EDB Postgres AI Hybrid Manager

リリース *1.4.0*

EnterpriseDB Corporation

2026 年 07 月 02 日

目 次

第 1 章　あなたにできること **3**
1.1　ここから始めます . . . 3

第 2 章　Hybrid Manager Innovation Release Notes **5**
2.1　Hybrid Manager デュアルリリース戦略 . . . 5
2.1.1　Innovation Release リリースノート . . . 5
2.1.1.1　Hybrid Manager 1.4.0 release notes . . . 6
2.1.1.1.1　リリースの詳細 . . . 7
2.1.1.2　Contributing Innovation Releases . . . 21
2.1.1.2.1　Hybrid Manager 2025.11.0 release notes . . . 21
2.1.1.2.2　Hybrid Manager 2025.12.0 release notes . . . 24
2.1.1.2.3　Hybrid Manager 2026.1.0 release notes . . . 27
2.1.1.2.4　Hybrid Manager 2026.2.0 release notes . . . 29
2.1.1.2.5　Hybrid Manager 2026.3.0 release notes . . . 31
2.1.1.2.6　Hybrid Manager 2026.4.1 release notes . . . 35
2.1.1.2.7　Hybrid Manager 2026.5.0 release notes . . . 46
2.1.1.2.8　Hybrid Manager 2026.5.1 release notes . . . 50

第 3 章　EDB Postgres AI Hybrid Manager **53**
3.1　なぜハイブリッドマネージャーを使うのか？ . . . 53
3.2　主要な機能 . . . 53
3.3　サポートされている環境 . . . 54
3.4　統合ソリューション . . . 54
3.5　詳細はこちら . . . 55
3.6　はじめる . . . 55
3.6.1　Hybrid Manager architecture . . . 55
3.6.1.1　スケーラビリティ . . . 56
3.6.1.2　バックアップアーキテクチャ . . . 56
3.6.1.3　高可用性と復元力 . . . 56
3.6.2　Supported database types . . . 56
3.6.2.1　分散高可用性 . . . 57
3.6.2.2　高度な高可用性 . . . 57
3.6.2.3　高可用性 . . . 57
3.6.2.4　単一ノード . . . 58
3.6.2.5　クラスタータイプのアーキテクチャ . . . 58
3.6.2.5.1　Distributed high availability . . . 58
3.6.2.5.2　Advanced high availability . . . 60
3.6.2.5.3　High availability . . . 61
3.6.2.5.4　Single-node cluster . . . 62
3.6.3　Known issues . . . 62

3.6.3.1 マルチ DC 63
3.6.3.1.1 アップグレード後のマルチ DC 構成の損失 63
3.6.3.1.2 マルチ DC セットアップの場所ドロップダウンリストが空です . . 63
3.6.3.1.3 プロジェクト分離のための専用オブジェクトストレージ 65
3.6.3.2 コアプラットフォームとリソース 65
3.6.3.2.1 OpenShift ユーザーのクロスストリームアップグレードの問題 . . 65
3.6.3.2.2 upm-beacon-agent 複雑な環境ではメモリ制限が不十分です . . . 65
3.6.3.2.3 バージョン 2026.2 へのアップグレード後に IdP 構成が失敗する . 65
3.6.3.2.4 Langflow シークレットが欠落しているため、accm-server の起動に失敗する 65
3.6.3.2.5 イメージをプルするためのハイブリッドマネージャーアプリでは kapp-controller サポートなし 66
3.6.3.2.6 スパイアが破損すると、ビーコンサーバーがクラッシュループに入る 66
3.6.3.2.7 SELinux が有効になっている RKE2 で SPIRE エージェントが起動に失敗する 66
3.6.3.2.8 重複するオペランドイメージが同じタグと SHA を共有する場合、イメージライブラリの検出に失敗します 67
3.6.3.3 データベースクラスターエンジン 67
3.6.3.3.1 max_connections がデフォルト以外の場合、3 ノード PGD クラスターの作成に失敗する 67
3.6.3.3.2 AHA 監視ノードのリソースがオーバープロビジョニングされています 67
3.6.3.3.3 HA クラスターは、ストリーミングレプリケーション証明書の認証に verify-full の代わりに verify-ca を使用します 67
3.6.3.3.4 2 番目のノードが遅すぎるため、大規模な HA クラスターに参加できません 67
3.6.3.3.5 2 つのデータグループと 1 つの監視グループを含む EDB Postgres 分散 PGD クラスターが正常ではありません 68
3.6.3.3.6 AKS デプロイメントは、リージョンごとにプライベートアクセスを備えた 1 つの PGD データグループに制限されています 68
3.6.3.3.7 RKE で明示的に選択されていない場合、ネットワークアクセスタイプはデフォルトの Private になります 68
3.6.3.3.8 ブートストラップ Job 作成が中断されると、CNPG オペレーターが無限リコンサイルループに入る 69
3.6.3.4 バックアップ リカバリー 69
3.6.3.4.1 リージョンをまたがってボリュームスナップショットリカバリーを使用する場合、レプリカクラスターの作成が失敗する 69
3.6.3.4.2 ボリュームスナップショットの復元は同じリージョンに制限されています 69
3.6.3.4.3 デフォルトの並列構成が原因で WAL アーカイブが遅い 69
3.6.3.4.4 transporter-db WAL ギャップのため、ディザスターリカバリー DR プロセスが失敗する場合がある 70
3.6.3.4.5 クラスターの完全なリカバリーは pg_ctl: server does not shut down エラーで失敗します 70
3.6.3.5 Agent Factory とモデル管理 70

3.6.3.5.1 プロファイルキャッシュを使用した nim-nvidia-nvclip モデルの展開の失敗 . . . 70
3.6.3.5.2 モデルプロファイルのオブジェクトストレージパスが空の場合、AI モデルクラスターの展開が停止する . . . 72
3.6.3.5.3 イノベーションリリースのアップグレード後にモデル構成設定がリセットされる . . . 72
3.6.3.5.4 ナレッジベースの資格情報を編集するとエラーページが表示される 72
3.6.3.5.5 PGD 監視ノードに aidb_users ロールが欠落しているため、AI 機能が防止されている . . . 73
3.6.3.5.6 GenAI Builder の構造とツールの実行失敗 . . . 73
3.6.3.5.7 パイプラインデザイナーが作成したナレッジベースは GenAI Builder に表示されません . . . 74
3.6.3.5.8 外部推論サービスパラメーターを編集すると、名前変更エラーが返されます . . . 74
3.6.3.5.9 無効な外部モデル名の検証メッセージがありません . . . 75
3.6.3.5.10 AIDB v6.1 を使用する PGD クラスターでパイプラインの作成が失敗する . . . 75
3.6.3.5.11 外部クラウドホスト OCR モデルは、HM コンソールを介して登録に失敗します . . . 75
3.6.3.6 アナリティクスと階層テーブル . . . 76
3.6.3.6.1 階層化パーティションテーブルの更新が PGAA エラーで失敗する 76
3.6.3.6.2 PostgreSQL 17 で一定の結果関係を使用して JOIN クエリーを実行すると、PGAA はサーバークラッシュ SIGSEGV を発生させます 76
3.6.3.7 HM コンソールと可観測性 . . . 77
3.6.3.7.1 Advanced High Availability および分散高可用性 PGD クラスターのフォールスポジティブバックアップアラート . . . 77
3.6.3.7.2 2 つの異なるイベントを表示するときにイベントマーカーが重なる 77
3.6.3.7.3 エステートページにアクセスするときの HTTP 431 “Request Header Fields Too Large” エラー . . . 77
3.6.3.7.4 タスクおよびクラスターのライフサイクルイベントの通知が見つからない . . . 77
3.6.3.7.5 WAL ストレージパーセントメトリックは、監視ページに誤った値を表示します . . . 78
3.6.3.7.6 GUC 設定は再起動するまで適用されない . . . 78
3.6.3.7.7 アラートしきい値の更新が断続的に有効にされない . . . 78
3.6.3.7.8 アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です . . . 78
3.6.3.7.9 アセットライブラリーアプリは表示名ではなくパッケージ名で並べ替えられました . . . 79
3.6.3.7.10 EPAS クラスターをデータベースバックエンドとして使用する場合、メタベースの展開に失敗する . . . 79
3.6.3.7.11 アセットライブラリーアプリ pgAdmin4 および Apache Superset は RHOS への展開に失敗する . . . 79
3.6.3.7.12 Airflow または Superset の暗号キーを編集すると、アプリが障害状態になる . . . 79
3.6.3.7.13 スーパーセットの展開は DuplicateTable エラーで失敗します . 80

3.6.3.7.14 アセットライブラリー起動アプリは、カスタムポータルポートに関係なく、常にポート 443 を使用します 80
3.6.3.7.15 アセットライブラリーアプリの CPU およびメモリメトリックには、失敗および完了したポッド状態が含まれます 80
3.6.3.7.16 サービスアカウントが見つからないため、アプリまたはフローの削除が停止する 80
3.6.3.7.17 ストレージクラスが異なる場合、RHOS でアセットライブラリーアプリが保留状態になることがある 81
3.6.3.8 クラスターとアップグレード 81
3.6.3.8.1 不変の `fluent-bit-k8s-events` StatefulSet が原因でバージョン 2026.4 へのアップグレードが失敗する 81
3.6.3.8.2 監視グループを使用した高度な高可用性および分散高可用性 PGD クラスターのメジャーバージョンのアップグレードが失敗する . . 82
3.6.3.8.3 ロギング名前空間の終了が原因でバージョン `2026.3` へのアップグレードが失敗する 82
3.6.3.9 Agent Factory 82
3.6.3.9.1 ナレッジベースデータは、PGD クラスターの非プライマリノードにレプリケートされません 82
3.6.3.9.2 hypopg 拡張機能が欠落しているため、インデックス分析アクションが失敗します 83
3.6.3.9.3 削除されたクラスターは AI コンポーネントに表示されたままです 83
3.6.3.9.4 必須 MCP ツールパラメーターの検証の欠落 83
3.6.3.9.5 一部の EDB コンポーネントツールは、互換性のない API 要求フォーマットのため機能しない 84
3.6.3.9.6 高度な高可用性と分散高可用性 PGD 監視グループがデータベース選択リストに表示される 84
3.6.3.9.7 テキスト検索は非表示フィールドに一致する場合があります . . . 84
3.6.3.9.8 Langflow MCP サーバーコンポーネントは HM フローホスティングでサポートされていません 84
3.6.3.9.9 EDB Knowledge Base コンポーネントを Langflow にドラッグすると 403 エラーを返します 84
3.6.3.9.10 データベースユーザーのグローバル変数を使用する場合、EDB Knowledge Base コンポーネントは接続に失敗します 85
3.6.3.9.11 空のナレッジベースに対してテストクエリーを実行すると、500 エラーが返されます 85
3.6.3.9.12 フローホスティングの展開は `FlowValidationError` で断続的に失敗します 85
3.6.3.9.13 EDB Model Server コンポーネントは、EDB Airman MCP コンポーネントと使用するとエージェント障害が発生します 85
3.6.3.9.14 `upm-ai-model-server-beaco` メモリ制限の不足による OOMKills 86
3.6.3.9.15 更新された Langflow フローの変更は、再公開後 HM にすぐに反映されません 86
3.6.3.10 マイグレーション 86
3.6.4 Supported Postgres distributions 86
3.6.4.1 EDB Postgres Advanced Server 86
3.6.4.2 EDB Postgres Extended Server 87
3.6.4.3 PostgreSQL 87

3.6.4.4 ディストリビューションの比較 . . . 87
3.6.4.5 ディストリビューション別のドキュメント . . . 88

第 4 章 Installing Hybrid Manager 89
4.1 ハイブリッドマネージャーのインストール . . . 89
4.1.1 step-by-step guide . . . 89
4.1.1.1 概要 . . . 89
4.1.1.2 フェーズ 1 アーキテクチャの計画 . . . 89
4.1.1.3 フェーズ 2 システム要件の収集 . . . 90
4.1.1.4 フェーズ 3Kubernetes クラスターの展開 . . . 90
4.1.1.5 フェーズ 4 環境の準備 . . . 91
4.1.1.6 フェーズ 5 ハイブリッドマネージャーのインストール . . . 91
4.1.1.7 フェーズ 6 インストール後のオプションの検討 . . . 91
4.1.1.8 HM の設定 . . . 92
4.1.1.8.1 Planning your architecture . . . 92
4.1.1.8.2 Gathering your system requirements . . . 108
4.1.1.8.3 Deploying your Kubernetes cluster . . . 130
4.1.1.8.4 Preparing your environment . . . 137
4.1.1.8.5 Install Hybrid Manager . . . 142
4.1.1.8.6 Exploring your post-installation options . . . 152
4.1.2 Configuring SOPs . . . 154
4.1.2.1 Syncing images to a local private registry . . . 154
4.1.2.1.1 前提条件 . . . 154
4.1.2.1.2 同期変数の構成 . . . 154
4.1.2.1.3 構成の更新 . . . 155
4.1.2.1.4 次のステップ . . . 156
4.1.2.2 Using the in-cluster container registry . . . 156
4.1.2.2.1 前提条件 . . . 156
4.1.2.2.2 セキュリティに関する考慮事項 . . . 157
4.1.2.2.3 HMRegistry への画像のロード . . . 158
4.1.2.2.4 HMRegistry を使用するように HybridControlPlane を構成する . . . 159
4.1.2.2.5 HM の外部のレジストリに戻す . . . 159
4.1.2.3 Configuring image discovery . . . 160
4.1.2.3.1 前提条件 . . . 160
4.1.2.3.2 imageDiscovery . . . 160
4.1.2.3.3 imagesetDiscoveryContainerRegistryURL . . . 161
4.1.2.3.4 imagesetDiscoveryAllowInsecureRegistry . . . 162
4.1.2.3.5 image_discovery_credentials_secret_name . . . 162
4.1.2.3.6 汎用/オンプレミス RKE2/OpenShift . . . 162
4.1.2.3.7 Azure Container RegistryACR . . . 163
4.1.2.3.8 AWS Elastic Container Registry ECR . . . 163
4.1.2.3.9 パッケージインストールの追加構成 . . . 164
4.1.2.3.10 Google Artifact Registry GAR . . . 164
4.1.2.3.11 GitLab コンテナレジストリ . . . 165
4.1.2.4 Customizing secrets . . . 166
4.1.2.4.1 Customizing GenAI Builder secrets . . . 166
4.1.2.4.2 Customizing Catalog secrets . . . 169

4.1.2.4.3 Customizing Migration Portal secrets for secure internal communication 170
4.1.2.5 Implementing TLS certificates . . . 176
4.1.2.5.1 ハイブリッドマネージャーポータルのカスタム x.509 証明書を設定する . . . 176
4.1.2.5.2 HM ポータルのカスタム cert-manager 発行者を設定する . . . 177
4.1.2.5.3 独自のプライベート認証局を使用する . . . 178
4.1.2.6 Configuring object storage . . . 179
4.1.2.6.1 プロバイダー構成 . . . 179
4.1.2.6.2 CA バンドルの提供 . . . 184
4.1.2.6.3 プロジェクトごとのオブジェクトストレージの構成 . . . 185
4.1.2.7 Deploying Hybrid Manager across multiple data centers . . . 185
4.1.2.7.1 Multi-DC for high availability . . . 186
4.1.2.7.2 Configuring a Hybrid Manager deployment across multiple data centers 190
4.1.2.8 Enabling Key Management Systems (KMS) for TDE . . . 196
4.1.2.8.1 Adding KMS support . . . 196
4.1.2.8.2 HashiCorp Vault . . . 198
4.1.2.8.3 AWS Key Management Service (KMS) . . . 201
4.1.2.8.4 Azure Key Vault (Key Management Service) . . . 202
4.1.2.8.5 Google Cloud Key Management Service (Cloud KMS) . . . 204
4.1.2.8.6 Deleting a key . . . 206
4.1.2.9 Optimizing service mesh memory . . . 206
4.1.2.9.1 Istio のメモリ要件を理解する . . . 206
4.1.2.9.2 推奨されるメモリ制限 . . . 206
4.1.2.9.3 Istio メモリ制限の構成 . . . 207
4.1.2.9.4 Istio のメモリ使用量のモニタリング . . . 208
4.1.2.10 API audit logging . . . 208
4.1.2.10.1 監査の対象となるもの . . . 209
4.1.2.10.2 API 監査ログの無効化 . . . 209
4.1.2.10.3 API 監査ログを再度有効にする . . . 209
4.1.2.10.4 監査ログの照会 . . . 210
4.1.2.10.5 ログフォーマット . . . 210
4.1.2.10.6 ログフィールド . . . 211
4.1.3 Migrating from bootstrap to operator installation . . . 211
4.1.3.1 概要 . . . 211
4.1.3.2 移行の課題を理解する . . . 211
4.1.3.2.1 Helm の所有権の競合 . . . 211
4.1.3.2.2 これが発生する理由 . . . 212
4.1.3.2.3 CRD 削除問題 . . . 212
4.1.3.3 edbctl で移行の準備をする . . . 212
4.1.3.3.1 変換を実行します . . . 213
4.1.3.4 1.7.0 未満の edbctl の手動フォールバック . . . 214
4.1.3.4.1 構成マッピングリファレンス . . . 214
4.1.3.4.2 移行スクリプトを使用して変換する . . . 215
4.1.3.4.3 Helm 所有権の移転マニュアル . . . 218
4.1.3.4.4 仕組み . . . 219
4.1.3.4.5 前提条件 . . . 219
4.1.3.4.6 ステップ 1 ブートストラップ展開を削除する . . . 221

4.1.3.4.7 ステップ 2 Helm の所有権を譲渡する - ServiceAccount 222
4.1.3.4.8 ステップ 3 Helm の所有権を譲渡する - CRD 222
4.1.3.4.9 ステップ 4Helm の所有権を譲渡する- クラスターロール 223
4.1.3.4.10 ステップ 5Helm の所有権を譲渡する- ClusterRoleBindings 223
4.1.3.4.11 ステップ 6Helm の所有権を譲渡する - ロール 224
4.1.3.4.12 ステップ 7Helm の所有権を譲渡する-ロールバインディング . . . 224
4.1.3.4.13 ステップ 8 Helm の所有権を譲渡する - サービス 225
4.1.3.4.14 ステップ 9 Helm の所有権を譲渡する - Webhook 構成 225
4.1.3.4.15 ステップ 10 所有権の移転を確認する 226
4.1.3.5 サポート 226
4.1.3.6 次のステップ 226
4.1.4 Troubleshooting Hybrid Manager installations 226
4.1.4.1 Understanding Preflight checks 226
4.1.4.1.1 前提条件 227
4.1.4.1.2 結果 227
4.1.4.1.3 プリフライトステータスの確認 227
4.1.4.1.4 状態を理解する 228
4.1.4.1.5 プリフライト障害のトラブルシューティング 228
4.1.4.2 Monitoring with Postflight checks 230
4.1.4.2.1 前提条件 230
4.1.4.2.2 結果 230
4.1.4.2.3 ポストフライトチェックの機能 230
4.1.4.2.4 ポストフライトステータスの確認 230
4.1.4.2.5 異常なチェックのトラブルシューティング 231
4.1.4.2.6 チェック間隔の設定 233

第 5 章 Using Hybrid Manager 235
5.1 どこから始めるべきか 235
5.2 ユーザー向け 235
5.3 Kubernetes 管理者の場合 236
5.3.1 Hybrid Manager Terminology 236
5.3.1.1 ハイブリッドマネージャー 236
5.3.1.1.1 プライベート Database-as-a-Service DBaaS 237
5.3.1.1.2 エンジニアードシステム 237
5.3.1.2 PG AI Console 237
5.3.1.2.1 プロジェクト 237
5.3.1.2.2 エステート 237
5.3.1.2.3 Launchpad 238
5.3.1.3 クラスター明確化 238
5.3.1.3.1 Kubernetes クラスター 238
5.3.1.3.2 Kubernetes ストレッチクラスター 238
5.3.1.3.3 Postgres クラスター 238
5.3.1.3.4 データベースクラスター 239
5.3.1.4 ハイブリッドマネージャーのデータベースクラスタータイプ 239
5.3.1.4.1 HM 管理対象クラスター 240
5.3.1.4.2 外部クラスター 240
5.3.1.4.3 自己管理クラスター 240

5.3.1.4.4 クラウドサービスプロバイダー管理のクラスター 241
5.3.1.4.5 外部 CNPG クラスター 241
5.3.1.5 その他の用語 241
5.3.1.5.1 データノード 241
5.3.1.5.2 Postgres インスタンス 241
5.3.2 Hybrid Manager console 242
5.3.2.1 Hybrid Manager コンソールとは何ですか? 242
5.3.2.2 HM コンソールでは何ができますか? 242
5.3.2.3 HM コンソールの構成 242
5.3.2.4 サードパーティサービスのローンチ 242
5.3.2.4.1 Exploring the Hybrid Manager console 243
5.3.2.4.2 Postgres clusters 245
5.3.2.4.3 OSS Library 248
5.3.3 For Users of Hybrid Manager 264
5.3.3.1 Projects 264
5.3.3.1.1 プロジェクトの作成 265
5.3.3.1.2 プロジェクトの詳細 265
5.3.3.1.3 プロジェクトの管理 266
5.3.3.1.4 投影されたオブジェクトストレージの構成 266
5.3.3.2 Working with clusters 274
5.3.3.2.1 何ができるでしょうか？ 275
5.3.3.3 Upgrading database clusters in Hybrid Manager 341
5.3.3.3.1 Postgres minor version upgrades in Hybrid Manager 342
5.3.3.3.2 Postgres major version upgrades 343
5.3.3.4 Migrating databases with Hybrid Manager 349
5.3.3.4.1 HM で移行機能を使用するにはどうすればよいですか? 350
5.3.3.4.2 EDB Data Migration Service はどのような種類の移行をサポートしていますか？ 350
5.3.3.4.3 EDB Data Migration Service はどのようなテクノロジーを使用しますか? 350
5.3.3.5 Asset management 552
5.3.3.5.1 HM Postgres とモデルアセットの管理 552
5.3.3.6 Backups and restores 558
5.3.3.6.1 Backups 558
5.3.3.6.2 Restores 564
5.3.3.7 EDB Hybrid Manager Chat Agent 567
5.3.3.7.1 サポートされている操作 567
5.3.3.7.2 モデル依存の機能 567
5.3.3.7.3 ドキュメントスコープ 567
5.3.3.7.4 次のステップ 567
5.3.3.8 Governance in Hybrid Manager 586
5.3.3.8.1 HM が提供するもの 586
5.3.3.8.2 前提条件 586
5.3.3.8.3 HM に接続する 586
5.3.3.8.4 詳細はこちら 587
5.3.3.9 Hybrid Manager CLI - edbctl 587
5.3.3.10 Interacting programmatically with HM 587

5.3.3.10.1 アクセスキー . . . 587
5.3.3.11 Managing identity providers . . . 593
5.3.3.11.1 Configuring your IdP in the HM console . . . 593
5.3.3.11.2 Rotating IdP connector encryption keys . . . 600
5.3.4 For Kubernetes Administrators . . . 602
5.3.4.1 Managing users in Hybrid Manager . . . 602
5.3.4.1.1 Managing native users for testing . . . 603
5.3.4.1.2 Managing authentication policy . . . 607
5.3.4.1.3 User roles and authorization . . . 608
5.3.4.2 Monitoring with Hybrid Manager . . . 615
5.3.4.2.1 監視アーキテクチャの理解 . . . 615
5.3.4.2.2 データを外部ツールに接続する . . . 617
5.3.4.2.3 関連トピック . . . 617
5.3.4.3 Upgrading Hybrid Manager . . . 706
5.3.4.3.1 LTS アップグレードパス . . . 707
5.3.4.3.2 クロスストリームアップグレードパス . . . 708
5.3.4.4 Hybrid Manager backup and disaster recovery (DR) . . . 728
5.3.4.4.1 Disaster recovery overview . . . 728
5.3.4.4.2 Configuring backups to support disaster recovery (DR) . . . 730
5.3.4.4.3 Hybrid Manager disaster recovery . . . 732
5.3.4.5 Enabling the Migration Portal AI Copilot . . . 743
5.3.4.5.1 サードパーティモデルクラウド . . . 744
5.3.4.5.2 セルフホストモデルオンプレミス . . . 744
5.3.4.6 Troubleshooting Hybrid Manager . . . 751
5.3.4.6.1 エラーコード . . . 751
5.3.4.6.2 エラーの詳細 . . . 752
5.3.4.7 Using query recommendations in Hybrid Manager . . . 768
5.3.4.8 ハイブリッドマネージャーでクエリ推奨事項を使用する . . . 768
5.3.4.8.1 クエリのレコメンデーションを使用するのは誰ですか? . . . 768
5.3.4.8.2 クエリレコメンデーションとは何ですか? . . . 768
5.3.4.8.3 クエリレコメンデーションを使用する理由 . . . 768
5.3.4.8.4 クエリの推奨事項を使用する場合 . . . 769
5.3.4.8.5 クエリレコメンデーションの仕組み . . . 769
5.3.4.8.6 クエリの推奨事項をチューニングプロセスに統合する . . . 769
5.3.4.8.7 ワークフロー クエリから最適化まで . . . 770
5.3.4.8.8 外部クラスターの要件 . . . 770
5.3.4.8.9 シナリオ例 . . . 770
5.3.4.8.10 サポートされているレコメンデーションタイプ . . . 771
5.3.4.8.11 関連情報 . . . 771

第 6 章 Agent Factory in Hybrid Manager 773
6.1 構築できる AI ソリューションの例 . . . 773
6.1.1 AI フローの構築 . . . 774
6.1.2 モデルの管理と提供 . . . 774
6.2 Agent Factory がハイブリッドマネージャーにどのように適合するか . . . 774
6.3 ハイブリッドマネージャーのソブリン AI . . . 774
6.4 はじめる . . . 775

6.4.1 Getting Started with AI Factory . . . 775
6.4.1.1 Getting started with Agent Factory on Hybrid Manager . . . 775
6.4.1.1.1 ビルドするもの . . . 775
6.4.1.1.2 始める前に . . . 776
6.4.1.1.3 オプション A — Langflow で AI フローを構築する . . . 776
6.4.1.1.4 オプション B —ナレッジベースパイプラインを作成する . . . 776
6.4.1.1.5 アプリケーションからモデルエンドポイントにアクセス . . . 776
6.4.1.1.6 監視と反復 . . . 776
6.4.1.2 Prerequisites for Agent Factory on Hybrid Manager . . . 777
6.4.1.2.1 エージェントファクトリーの有効化 . . . 777
6.4.1.2.2 インフラストラクチャ要件 . . . 777
6.4.1.2.3 レジストリ構成 . . . 777
6.4.1.2.4 次のステップ . . . 778
6.4.1.3 GPU Recommendations for Default NIM Models . . . 778
6.4.1.3.1 概要 . . . 778
6.4.1.3.2 デフォルトの NIM モデル . . . 779
6.4.1.3.3 最小 GPU 要件 . . . 779
6.4.1.3.4 クラウドマッピング . . . 779
6.4.2 Understand the AI Factory Architecture . . . 779
6.4.2.1 Agent Factory Architecture on Hybrid Manager . . . 779
6.4.2.1.1 アーキテクチャの概要 . . . 779
6.4.2.1.2 コアコンポーネント . . . 779
6.4.2.1.3 インフラストラクチャ . . . 781
6.4.2.2 Use Cases and Personas for Agent Factory on Hybrid Manager . . . 781
6.4.2.2.1 概要 . . . 781
6.4.2.2.2 主なユースケース . . . 781
6.4.2.2.3 ユーザーペルソナ . . . 783
6.4.2.2.4 業界固有の実装 . . . 784
6.4.2.2.5 実装の考慮事項 . . . 785
6.4.2.2.6 ユースケース別のスタートガイド . . . 786
6.4.2.2.7 追加リソース . . . 786
6.4.2.3 Frequently Asked Questions - Agent Factory on Hybrid Manager . . . 786
6.4.2.3.1 目次 . . . 786
6.4.2.3.2 プラットフォーム機能 . . . 787
6.4.2.3.3 インストールとセットアップ . . . 787
6.4.2.3.4 モデル管理 . . . 788
6.4.2.3.5 Langflow . . . 789
6.4.2.3.6 運用とメンテナンス . . . 789
6.4.2.3.7 障害対応 . . . 790
6.4.2.3.8 セキュリティとコンプライアンス . . . 791
6.4.2.3.9 パフォーマンスとスケーリング . . . 791
6.4.2.3.10 追加リソース . . . 792
6.4.3 Model Management . . . 792
6.4.3.1 Model Management on Hybrid Manager . . . 792
6.4.3.1.1 目的と利点 . . . 792
6.4.3.1.2 コアコンセプト . . . 793
6.4.3.1.3 実装ワークフロー . . . 794

6.4.3.1.4 デプロイされたモデルを使用する . . . 794
6.4.3.2 Deploy Agent Factory with Hybrid Manager UI . . . 795
6.4.3.2.1 概要 . . . 795
6.4.3.2.2 前提条件 . . . 795
6.4.3.2.3 ステップ 1 レジストリ認証を構成する . . . 796
6.4.3.2.4 ステップ 2 モデルライブラリーにアクセスする . . . 797
6.4.3.2.5 ステップ 3 モデルサーバークラスターを展開する . . . 797
6.4.3.2.6 ステップ 4 展開ステータスを監視する . . . 798
6.4.3.2.7 ステップ 5API アクセスを構成する . . . 799
6.4.3.2.8 ステップ 6 Gen AI アプリケーションをビルドする . . . 799
6.4.3.2.9 ステップ 7 モデルの更新と管理 . . . 799
6.4.3.2.10 Langflow フローの高可用性と分離 . . . 800
6.4.3.2.11 障害対応 . . . 800
6.4.3.2.12 追加リソース . . . 801
6.4.3.3 External inference services . . . 801
6.4.3.3.1 コンテキスト . . . 801
6.4.3.3.2 推論サービスリスト . . . 801
6.4.3.3.3 外部推論サービスを登録する . . . 802
6.4.3.3.4 推論サービスの詳細の取得 . . . 806
6.4.3.3.5 推論サービスパラメーターの更新 . . . 807
6.4.3.3.6 外部推論サービスの登録を解除する . . . 808
6.4.4 analytics . . . 809
6.4.4.1 Langflow . . . 809
6.4.4.1.1 前提条件 . . . 809
6.4.4.1.2 次に何をするか . . . 810
6.4.4.2 Pipeline Designer . . . 859
6.4.4.2.1 Pipeline Designer と AIDB の関係 . . . 859
6.4.4.2.2 前提条件 . . . 860
6.4.4.2.3 サポートされているクラスタータイプ . . . 860
6.4.5 Operate & Observe AI Factory . . . 895
6.4.5.1 Observability (Agent Factory on HM) . . . 895
6.4.5.1.1 ハイブリッドマネージャーの可観測性 . . . 895
6.4.5.1.2 メトリックとログへのアクセス . . . 896
6.4.5.1.3 一般的なコマンド . . . 896
6.4.5.1.4 Agent Factory メトリックの Grafana ダッシュボードの設定 . . . 896
6.4.5.1.5 重要なポイント . . . 897
6.4.5.2 Troubleshooting Agent Factory on Hybrid Manager . . . 897
6.4.5.2.1 診断の概要 . . . 897
6.4.5.2.2 クイックリファレンスリンク . . . 898
6.4.5.2.3 インフラストラクチャの問題 . . . 898
6.4.5.2.4 モデル提供の失敗 . . . 899
6.4.5.2.5 Langflow および Gen AI アプリケーションの問題 . . . 901
6.4.5.2.6 ログ分析 . . . 901
6.4.5.2.7 アラート設定 . . . 902
6.4.5.2.8 エスカレーションプロシージャ . . . 902
6.4.5.2.9 予防措置 . . . 903

第 7 章 Analytics in Hybrid Manager **905**
7.1 Configuring tiered storage . . . 905
7.1.1 クラスターの分析の管理 . . . 905
7.1.2 テーブルの表示 . . . 906
7.1.3 階層テーブルへの変換 . . . 906
7.1.4 アナリティクスへの複製 . . . 907
7.1.5 アナリティクスへのオフロード . . . 908
7.1.6 コンパクションのトリガー . . . 908
7.2 Managing catalogs . . . 909
7.2.1 クラスターのカタログの管理 . . . 909
7.2.2 クラスター全体で分析データのクエリー . . . 909

第 8 章 Sovereign Data and Agent Factory **911**
8.1 ハードウェアベースのシステムを選択する理由は何ですか? . . . 911
8.2 中身は何ですか . . . 911
8.3 サポートされているワークロード . . . 912
8.4 ハードウェア構成 . . . 912
8.5 Explore 機能 . . . 912
8.5.1 Concepts and architecture . . . 912
8.5.1.1 ノードタイプとロール . . . 913
8.5.1.1.1 管理ノード . . . 913
8.5.1.1.2 ハイブリッドマネージャーノード . . . 913
8.5.1.1.3 コンピューティングノード . . . 913
8.5.1.1.4 AI ノードオプショナル . . . 913
8.5.1.2 オーケストレーションとライフサイクル . . . 914
8.5.1.3 「ソブリン」とは何ですか？ . . . 914
8.5.1.4 サポートされているワークロードパターン . . . 914
8.5.2 Frequently asked questions . . . 915
8.5.2.1 クラウドサービスの代わりにこのシステムを使用するのはなぜですか? . . . 915
8.5.2.2 ハードウェアのインストールと保守は誰が行いますか? . . . 915
8.5.2.3 どのようなソフトウェアを実行しますか? . . . 915
8.5.2.4 このシステムをクラウドサービスに接続できますか? . . . 915
8.5.2.5 AI の使用がサポートされているモデルは何ですか? . . . 915
8.5.2.6 外部データベースをモニタできますか? . . . 916
8.5.2.7 システムをスケーリングするにはどうすればよいですか? . . . 916
8.5.2.8 サポートには何が含まれますか? . . . 916
8.5.3 Glossary and key terms . . . 916
8.5.3.1 管理ノード . . . 916
8.5.3.2 AI ノード . . . 916
8.5.3.3 エージェントワークフロー . . . 916
8.5.3.4 コントロールプレーン . . . 916
8.5.3.5 コンピューティングノード . . . 917
8.5.3.6 埋め込み . . . 917
8.5.3.7 EDB ハイブリッドマネージャー . . . 917
8.5.3.8 推論 . . . 917
8.5.3.9 KServe . . . 917
8.5.3.10 ライフサイクルサービス . . . 917

8.5.3.11 EDB Postgres Distributed PGD . . . 917
8.5.3.12 取得拡張生成 RAG . . . 917
8.5.3.13 ソブリン AI . . . 917
8.5.3.14 Vector ストア . . . 918
8.5.4 System specifications . . . 918
8.5.4.1 構成オプション . . . 918
8.5.4.2 構成 1 コアコンピューティング . . . 918
8.5.4.2.1 管理ノード . . . 918
8.5.4.2.2 コアコンピューティング – Hyper 2U サーバーサーバーごとに x6 . 919
8.5.4.3 構成 2: 高度な AI . . . 920
8.5.4.3.1 2-GPU Hyper 2U サーバーサーバーごとに x2 . . . 920
8.5.4.3.2 8-GPU B200 10U サーバーサーバーごとに x2 . . . 921
8.5.5 Common workloads and use cases . . . 922
8.5.5.1 クラウドでエンジニアドシステムを使用する理由は何ですか? . . . 922
8.5.5.1.1 データ主権とコンプライアンス . . . 922
8.5.5.1.2 規制された環境 . . . 923
8.5.5.1.3 Postgres + AI の統合プラットフォーム . . . 923
8.5.5.1.4 予測可能なパフォーマンスとコスト . . . 923
8.5.5.1.5 ライフサイクル管理されたインフラストラクチャ . . . 923
8.5.5.2 サポートされているワークロードパターン . . . 923
8.5.5.2.1 高可用性 Postgres . . . 923
8.5.5.2.2 ハイブリッド DBaaS . . . 923
8.5.5.2.3 エンベディングとベクターパイプライン . . . 924
8.5.5.2.4 取得拡張生成 RAG . . . 924
8.5.5.2.5 オンプレミスの推論 . . . 924
8.5.5.2.6 エージェント AI ワークフロー . . . 924
8.5.6 What to expect . . . 924
8.5.6.1 0〜21 日目: サイトの準備とインストール . . . 924
8.5.6.1.1 何が起こるか . . . 924
8.5.6.2 21〜28 日目: 最初のワークロードの展開 . . . 925
8.5.6.2.1 あなたにできること . . . 925
8.5.6.3 28〜90 日目 ライフサイクル管理とスケーリング . . . 925
8.5.6.3.1 含まれるもの . . . 925

EDB Postgres AI Hybrid Manager HM は、 EDB Postgres AI プラットフォームのコントロールプレーンです。Kubernetes が実行されている場所ならどこでも実行される**ソブリン AI、分析、および Postgres 管理ソリューション**です。データベースの展開、GenAI ワークロードの実行、および **クラウド**、 **オンプレミス** 、または **ハイブリッド環境**全体で Postgres ネイティブ分析のスケーリングのための統一エクスペリエンスを提供します。

現在、オンプレミスの **Red Hat OpenShift RHOS** 、 **Rancher RKE2** 、およびインクラウドの場合 **Amazon EKS** 、および **Google GKE の次の Kubernetes ディストリビューションでサポートされています。** 。

第1章　あなたにできること

Area	Capabilities
Database provisioning	コンソール、テンプレート、または宣言的マニフェストを使用して、高可用性の PostgreSQL または分散 Postgres クラスターを展開します。
AI integration	sovereign GenAI pipelines を実行し、KServe でモデルを提供し、インフラストラクチャによって管理される AI ナレッジベースを構築します。
Analytics	コールドデータを Delta Lake or Iceberg にオフロードし、Postgres 構文で照会し、ハイブリッドトランザクション/分析処理を有効にします。
Monitoring and observability	Prometheus、Grafana、Loki を活用する統合ダッシュボードを使用して、クエリの状態、リソースの使用状況、システムアラートに関する詳細な可視性を取得します。
Security and access management	ID プロバイダーと統合し、プロジェクトとクラスターの RBAC を定義し、データベースとサービスへの安全なアクセスを管理します。
Database migration	外部データベース Oracle、自己管理の Postgres から HM 管理クラスターにスキーマとデータを移行します。移行とスキーマ評価、データ転送スナップショット、および継続的レプリケーションストリーミングのための統合サービスを活用して、最小限のダウンタイムの移行を実現します。

1.1 ここから始めます

- *Install Hybrid Manager* → 完全な HM インストールをガイドする詳細なガイド。
- *Sovereign Data and Agent Factory* → ソブリンデータとエージェントファクトリーハードウェア
- *Using Hybrid Manager* → バックアップ/リストア、モニタリング、移行、クラスター作成、マルチ DC 展開など
- *Getting started with Agent Factory on Hybrid Manager* → ソブリン GenAI パイプライン、モデル提供、および取得システム
- *Analytics in Hybrid Manager* → デルタレイク、氷山、階層化ストレージ、およびレイクハウスパターン
- *Hybrid Manager 2025.11.0 release notes*

第2章　Hybrid Manager Innovation Release Notes

!!!info リリース周期の更新 HM は、 **LTS** と **イノベーションリリース**のデュアルリリース戦略に従います。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

Version	Release date
Hybrid Manager 1.4.0 release notes	24 June 2026

2.1 Hybrid Manager デュアルリリース戦略

Hybrid ManagerHM2025.11 では、新しい機能と安定性の向上をより迅速に予測可能な形で提供するように設計された新しいリリース戦略を採用しています。

従来のリリーススキーム 1.3.0 などのセマンティックバージョンを使用することに加えて、日付ベースのリリーススキーム 2025.11 などのバージョン番号を追加したことに注意してください。これをイノベーションリリースと呼んでいます。この変更は、継続的デリバリーとより明確なリリースリズムへの取り組みをサポートします。

- **イノベーションリリース月次** 2025.11 などのこれらのリリースは、毎月配信されます。彼らは、継続的なプラットフォームの強化、ワークフローの改善、安定性、信頼性、より小規模な反復機能の導入に焦点を当てています。これらは、開発の最先端を進み、製品をどこに持っていくかを見たい顧客とチームに最適です。
- **長期サポート LTS リリース年 2 回** 1.3.0 などのこれらのリリースは、年に 2 回配信されます。これらは、以前のイノベーションリリースの内容を、拡張サポートのコミットメントを必要とする展開のための安定した基盤に統合します。これらの LTS リリースは、リリース日から 15 か月間サポートされ、バグとセキュリティの修正を含む 1.3.1 などの月例パッチリリースでサポートされます。

ハイブリッドマネージャーの最新のプラットフォームの互換性については、

Platform Compatibility page を参照してください。

2.1.1 Innovation Release リリースノート

これらのノートでは、ハイブリッドマネージャーの 1.4LTS リリースで最高潮に達したイノベーションリリースをカバーしています。

Version	Release date
Hybrid Manager 2026.5.1 release notes	2 June 2026

次のページに続く

表 2 – 前のページからの続き

Version	Release date
Hybrid Manager 2026.5.0 release notes	19 May 2026
Hybrid Manager 2026.4.1 release notes	23 Apr 2026
Hybrid Manager 2026.3.0 release notes	12 Mar 2026
Hybrid Manager 2026.2.0 release notes	12 Feb 2026
Hybrid Manager 2026.1.0 release notes	15 Jan 2026
Hybrid Manager 2025.12.0 release notes	11 Dec 2025
Hybrid Manager 2025.11.0 release notes	07 Nov 2025

2.1.1.1 Hybrid Manager 1.4.0 release notes

Hybrid ManagerHM 1.4.0LTS は、Microsoft AKS サポートで展開の柔軟性を拡張し、新しい自動化エンジンで日常業務を簡素化し、パイプライン、移行、およびビルトインチャットボット全体の AI 機能を強化します。この長期サポートリリースには、イノベーションリリース 2025.11 から 2026.5.1 のすべての機能が統合されています。また、多数のバグ修正と生活の質の向上が含まれています。

発売日 2026 年 6 月 24 日

!!!note Postgres は、EDB でサポートされているすべてのディストリビューションを指します

- PostgreSQL
- EDB Enterprise Postgres - 以前は EDB Postgres Extended ServerPGE
- EDB Enterprise PostgresOracle 互換-以前の EDB Postgres Advanced Server
- ** *Azure Kubernetes Service (AKS) support* ** AI パイプライン、アナリティクスアクセラレーター、およびデータ移行サービスを完全にサポートしている HM を Microsoft AKS に展開できるようになりました。
- ** *Selective installations* ** インストール時またはその後に、必要な `scenarios` `core` 、`dbaas` 、`migration` 、`ai` 、および `analytics` のみを有効にすることにより、HM 展開をカスタマイズ、リソース使用量を削減し、コンポーネント管理を簡素化します。
- ** *Automations engine* ** 新しいオートメーションフレームワークは、ディスクのオートスケーリング、CPU とメモリのスケーリング、インデックスアプリケーション、マイナーおよびパッチデータベースバージョンの更新を含む日常的なデータベース操作を処理します。すべて構成可能なメンテナンスウィンドウとオプションのヒューマンインザループの承認。
- ** *OSS Library* ** コミュニティデータベースツール Apache Airflow、Apache Superset、Metabase、pgAdmin 4、pgBadger、および HammerDB を、構成可能なパラメーターと承認ワークフローを使用して、**アセットライブラリー* から HM プロジェクトに直接展開します。
- ** *What the Chat Agent can do* ** Postgres クラスター管理、エステートの参照、モニタリングとアラート、ヘルスレポートの取得、Oracle から Postgres への SQL 変換、および引用裏付けの回答付きのドキュメント Q&A 用の HM コンソールに組み込まれている会話アシスタント。OpenAI、Anthropic、Google Gemini のモデルと、HM を介して展開されたセルフホストモデルをサポートします。
- ** *Custom load balancer configuration* ** HM は、MetalLB アドレスプールの割り当てや Azure プライベートクラスターでの Azure Private Link の無効化などのユースケースにカスタムロードバランサー

のアノテーションをサポートしています。

- ** *Three-location cluster deployments* ** 同期レプリケーションを介したゼロデータ損失構成のために、Postgres クラスターを 3 つの地理的に離れた場所に展開します。HM は、クォーラムとスプリットブレイン防止のために 3 番目の場所で分散高可用性 DHAWitness Groups もサポートしています。
- ** *Unified schema migration* ** Oracle SQL 互換性評価と AI エージェントを介した API 駆動の自動化を使用して、HM を介して Oracle および Postgres ソースから Postgres の宛先にデータベーススキーマを移行します。
- ** *Model Context Protocol (MCP) integration* ** HM は MCP を介して API 機能を公開するようになり、AI エージェントと開発者ツールがエージェントワークフローの HM API を使用できるようになります。
- ** *Agent Factory enhancements* ** Visual AI Pipeline Designer、マルチステップコンポーザブルパイプライン、ハイブリッド検索、`summarize_text_aggregate()` SQL ファンクション、および EDB Airman MCP Server、EDB Embedded Models、EDB Vector Index を含む拡張 Langflow コンポーネント。
- ** *Langflow integration* ** Langflow は、HM Launchpad にネイティブに統合されています。HM 内で AI 搭載エージェントとワークフローを構築し、管理対象ワークロードとしてフローを展開し、Oracle および Postgres ソースから Postgres 宛先へのエンドツーエンドのデータベース移行に Langflow テンプレートを使用できます。
- ** *Enhanced migration risk assessment* ** レプリケーションスロットの状態、WAL 生成率、および危険なテーブル構成主キー、`REPLICA IDENTITY`、`TOAST`、およびデータ移行サービスでサポートされていないデータ型をカバーするソースデータベースの詳細な評価。
- ** *Advanced observability* ** 日次のクラスター状態レポート、スキーマ変更の検出、新しいクエリー検出、メトリック比較ツール、および強化されたモニタリングダッシュボード。
- ** *ARM architecture support* ** HM コンポーネントが ARM ベースのアーキテクチャで実行されるようになり、最新の ARM プラットフォームと Kubernetes ノードへの展開が可能になります。

2.1.1.1.1 リリースの詳細

Hybrid ManagerHM プラットフォーム

Type	Description
Platform support	Kubernetes 1.34 および OpenShift 4.21 のサポートを追加しました。
Platform support	HM の ARM64 アーキテクチャのサポートを追加し、展開オプションを ARM64 ベースの Kubernetes 環境に拡張します。サポートされている構成については、*Troubleshooting* を参照してください。
Feature	AI パイプラインとデータ移行サービス DMS の完全なサポートを備えた、Microsoft Azure Kubernetes ServiceAKS での HM 展開のサポートを追加しました。
Feature	HM コンソール **設定> ID プロバイダー** から構成可能な独自の ID プロバイダー IdP のサポートを追加しました。 HM は、IdP グループ属性に基づいてユーザーロールを自動的に割り当て、ユーザーグループをロールに宣言的にマッピングできるようになりました。

次のページに続く

表 3 – 前のページからの続き

Type	Description
Feature	情報クエリと Postgres クラスター管理のための HM コンソールの会話アシスタントである Chat Agent を追加しました。
Feature	HM はモデルコンテキストプロトコル MCP を介して API 機能を公開するようになりました。これにより、開発者と AI エージェントは、合理化されたデータベースエージェント構成のための Langflow との統合を含む、エージェントユースケースで HM API を使用できます。
Feature	HM コンソールのローンチパッドに Langflow が含まれ、プラットフォーム内で AI 搭載エージェントとワークフローを直接構築およびテストできます。
Feature	HM は選択的インストールをサポートするようになりました。インストール時以降に、必要な`scenarios` *core*、*dbaas*、*migration*、*ai*、および`analytics`のみを有効にすることにより、展開をカスタマイズ、リソース使用量を削減し、コンポーネント管理を簡素化します。
Feature	地理的に離れた 3 つの場所に Postgres クラスターを展開するためのサポートを追加し、同期レプリケーションを介したゼロデータ損失アーキテクチャを有効にします。 HM は、分散高可用性 DHAWitness グループを 3 番目の場所に展開して、クォーラムを維持し、自動フェイルオーバー中のスプリットブレインシナリオを防止することもサポートしています。
Feature	日常的なデータベース操作を自動化するための **Automations** エンジンを導入しました。使用可能な自動化には、データベースストレージの自動スケーリング、CPU とメモリの自動スケーリング、自動インデックスアプリケーション、自動マイナーおよびパッチイメージのアップグレードが含まれます。このフレームワークは、プロジェクトレベルのメンテナンスウィンドウ、詳細な実行制御、オプションのヒューマンインザループ承認、およびすべての自動アクティビティを集中監視するための統合 **タスクマネージャー** を特徴としています。
Feature	HM プロジェクトに直接展開できるコミュニティデータベースツールのカタログである OSS ライブラリーを紹介しました。初期カタログには、Apache Airflow、Apache Superset、Metabase、pgAdmin 4、pgBadger、および HammerDB が含まれています。アプリは、 **アセットライブラリー>アプリ** セクションから展開され、構成可能なパラメーター、オプションの展開スケジュール、および人間参加型の承認ワークフローを使用します。
Feature	HM コンソールのクラスター詳細ビューから直接分散高可用性クラスターの書き込みリーダースイッチオーバーを開始する機能を追加しました。
Feature	API 監査ログを追加しました。 HM は、イングレスゲートウェイを介してすべての認証された API 要求の構造化 JSON 監査ログエントリを出力し、コンプライアンスとセキュリティ監査のためにユーザー、セッション、API キー、および応答の詳細をキャプチャします。ログは、Grafana または`kubectl`を介して照会できます。
Feature	動的な PGD プレビューを追加し、クラスターのプロビジョニングおよび管理ワークフロー中に PGD 構成のリアルタイムプレビュー機能を有効にします。
Feature	プライマリ/スタンバイレプリケーション PSR クラスターの作成および編集操作の edbctl に`nodeSelector`および`tolerations`サポートを追加し、Postgres ワークロードの専用および分離されたノードプールを有効にします。

次のページに続く

表 3 – 前のページからの続き

Type	Description
Enhancement	高度なワークロード分離のための Postgres クラスタープロビジョニングとパッチ API に `nodeSelector`および `tolerations`のサポートを追加しました。
Enhancement	チャットエージェントは、選択リストの相互作用方法と別のパスワード送信パスを使用して、パスワードが大規模な言語モデルにアップロードされないようにしました。
Enhancement	Langflow フローをバンドルとして公開し、HM 内で展開するためのサポートを追加しました。
Enhancement	マルチロケーションマルチ DC 展開手順を簡素化しました。マルチ DC トポロジは、主に `primary.yaml`および `secondary.yaml`構成ファイルを介して制御されるようになり、標準の Helm プラクティスをサポートします。
Enhancement	クラスター詳細ビューには、分散高可用性クラスターのすべてのグループノードがリストされ、現在のトポロジとスイッチオーバー操作の前後のステータスを可視化します。
Enhancement	HM コンソールを介して Witness グループのネットワーク構成を変更できるようになりました。
Enhancement	HM イングレスロードバランサーのグローバル構成パラメーターを追加し、HM プラットフォームネットワーキングとトラフィック管理のより詳細な制御を提供します。
Enhancement	サポートされているプラットフォーム全体での EDB Postgres AI エージェント `beacon-agent`プロビジョニングにカスタムロードバランサーアノテーションのサポートを追加しました。
Enhancement	ロードバランサー構成の強化 `private`モードでは、HM デフォルトのプライベートアノテーションとともにカスタムアノテーションを使用できるようになりました。 `public`モードは、カスタムアノテーション、HM デフォルトプライベート、および HM デフォルトパブリックアノテーションをサポートしています。
Enhancement	クラスターを作成または編集するときにネットワークアクセスにカスタムロードバランサー構成オプションを追加し、読み取り書き込みおよび読み取り専用接続にカスタムロードバランサーサービスのアノテーションキーと値のペアを使用できます。
Enhancement	データ移行サービスの Kafka サービスを含む Kafka ロードバランサーでのカスタム Kubernetes Service アノテーションのサポートを追加し、BYO ネットワークおよびオンプレミスの展開シナリオを有効にします。
Enhancement	HM は、`edbctl project create-object-store`を使用した CLI を介した場合も含め、HM プロジェクトごとの専用オブジェクトストレージの構成をサポートするようになりました。
Enhancement	HM クラスター API は、プロビジョニングが失敗した場合の可視性を高め、セルフサービスの診断と解決までの時間の短縮を可能にします。
Enhancement	HA および DHA クラスターのオプションの読み取り専用接続文字列を追加して、読み取り操作をスタンバイレプリカまたは非書き込みリーダーノードにルーティングし、プライマリの負荷を削減します。

次のページに続く

表 3 – 前のページからの続き

Type	Description
Enhancement	Barman Cloud の統合を介して、HM バックアップ機能に高度な圧縮を追加しました。 WAL 圧縮がデフォルトで有効になり、ベースバックアップには Snappy 圧縮を、WAL ファイルには zstd を使用します。
Enhancement	統合システムデータベース *upm-system-db* を導入しました。
Enhancement	プラットフォームによって生成されたエージェントの自動タスクを含む、拡張されたイベントソースでデータベース通知システムを強化しました。監査ログテーブルのすべてのリソースタイプをクリックできるようになりました。
Enhancement	イメージ検出のレジストリプロバイダーとしての GitLab コンテナレジストリのサポートを追加しました。
Enhancement	イメージライブラリの **この画像を使用しているクラスター** ビューを強化して、ユーザーロールとプロジェクト権限に基づいて、アクセス権限があるクラスターのみを表示します。
Enhancement	HM オペレーター Helm チャートでの `kapp-controller` TLS 設定の構成のサポートを追加しました。カスタム CA 証明書信頼を設定するか、プライベートコンテナレジストリの TLS 検証を無効にできるようになりました。
Enhancement	llms.txt 規約に従って、機械読み取り可能な API インデックスを追加し、LLM 搭載コーディングアシスタントの使用可能なエンドポイントの合理的なリファレンスを提供します。
Change	*edb_stat_monitor.edbsm_normalized_query* GUC のデフォルト値を `false`から `true`に変更しました。述語で使用される値はデフォルトで HM に保存されなくなり、データのプライバシーが向上します。
Change	HM リリースから HM オペレーターを分離して、独立したバージョン管理と展開サイクルを有効にします。オペレーターは、操作にリモート方法の使用を強制し、ローカルに指定された方法をオーバーライドして、一貫したクラスター管理動作を保証します。
Change	HM を新しいバージョンにアップグレードするときに必要な中間手順を自動化しました。以前は、アップグレードには、パッチの適用、リソースの注釈、CRD の適用などの一連の Kubernetes 操作を手動で実行する必要がありました。ユーザーがターゲットバージョンを指定してアップグレードを開始すると、これらの手順が自動的に処理されるようになりました。
Change	HM プラグインを edbctl CLI に埋め込みました。 edbctl CLI には、*hm status*、*hm component suspend*、*hm component resume*、および `hm component retry`コマンドが含まれるようになりました。
Change	複数のデータベースを単一のデータベースに統合することにより、最小展開フットプリントを削減し、操作オーバーヘッドを削減します。
Change	オペレーター Helm チャートを介して有効になるオプションのクラスター内コンテナレジストリのサポートを追加しました。
Change	競合を回避するために CRD にアノテーションを付けるだけでなく、ターゲットプラットフォームに既にインストールされている場合に、prometheus-operator コンポーネントを無効にできるようになりました。

次のページに続く

表 3 – 前のページからの続き

Type	Description
Change	簡素化されたロードバランサーと NodePort 構成。カスタムリソースの単一のパラメーターを介してロードバランサーまたはノードポートを使用するように Istio サービスとデータベースサービスの両方を構成できるようになりました。
Change	プリフライトチェックを強化して、展開を開始する前にオブジェクトストレージの接続を検証し、誤ったオブジェクトストレージのシークレットによって発生する障害を防ぎます。
Change	`bdr.ddl_replication`のような以前は使用できなかった設定を含む、HM を介してすべての PGD GUC を構成するサポートを追加しました。
Change	プロビジョニング、バックアップ、保存場所、クラスターテンプレートを含むデータベースクラスター管理機能は、 `core`シナリオから分離された専用の`dbaas`シナリオによって制御されるようになりました。新規インストールには、デフォルトで`dbaas`が含まれます。 **2026.3 からアップグレードする既存のインストールは、Postgres クラスター管理機能を保持するために、`dbaas`をシナリオリストに明示的に追加する必要があります。 `dbaas`がないと、クラスター管理 API は 404 を返し、** Clusters**セクションは HM コンソールで非表示になります。
Deprecation	AHA – Advanced High AvailabilityPGD-S クラスタータイプを非推奨にします。クラスタータイプは HM コンソールで無効になっており、新しい AHA クラスタープロビジョニングを防ぎます。 AHA は MCP およびチャットボットコンポーネントでも非推奨であり、非推奨の通知が API ドキュメントに追加されました。既存の AHA クラスターは、引き続き通常に機能します。既存の AHA クラスターを PGD 単一分散グループ構成に移行する計画を立てます。
Deprecation	HM にバンドルされている Apache Spark コンポーネントを削除しました。外部 Spark クラスターへの接続は引き続きサポートされています。アクションは必要ありません。このコンポーネントは置き換えられません。
Deprecation	HM CRD の `beaconServer.additionalTrustDomains`フィールドを非推奨にしました。アクションは必要ありません。フィールドは下位互換性のために CRD によって受け入れられていますが、もう使用されません。値を設定しても効果はありません。このフィールドには置き換えられません。
Deprecation	*bootstrap* **および** *`edbpgai-bootstrap* Helm チャートのインストール方法を非推奨にします。 **必要なアクション** 統合された HM Helm チャートに移行します。 *Troubleshooting* を参照してください。
Deprecation	HM コンソールから Lakehouse Analytics Cluster 機能を削除しました。アクションは必要ありません。既存のクラスターは影響を受けず、別のクラスタータイプへの移行は必要ありません。分析機能は、分散高可用性 DHA クラスターと、PGAA 拡張機能が有効になっているシングルノード Postgres クラスターで PGAA を介して利用できるようになりました。
Deprecation	Lakehouse Analytics Cluster 機能に関連付けられた`upm-beacon-analytics-cnpg-plugin`コンポーネントを削除しました。アクションは必要ありません。代わりに DHA またはシングルノード Postgres クラスターで PGAA を使用してください。

エージェントファクトリー

Type	Description
Feature	すべての AIDB ナレッジベースを HM 内で直接作成、管理、および表示する機能を追加しました。
Feature	コンテキストウィンドウの制限や最適なユースケースを含む重要なモデルのメタデータを使用して、使用可能な AI モデルを確認し、HM インターフェイスから新しいものを直接追加する機能を追加しました。
Feature	Agent Factory が Postgres 18 をサポートするようになりました。
Feature	AI Pipelines には、Visual Pipeline Designer が含まれるようになりました。これは、HM に統合されている、AI データ準備パイプラインを開発、展開、および管理するための Web ベースのグラフィカルデザインツールです。
Feature	モデル提供のための外部サービスへのアクセスのサポートを追加し、HM、データパイプライン、および HM チャットボット内で独自のリモート AI モデルを接続して使用できます。
Feature	AIDB パイプラインフレームワークを拡張して、マルチステップのコンポーザブルパイプラインをサポートしました。プリペアラー機能たとえば、PDF からテキスト、HTML からテキストをチェーンして、あるステップの出力が次のステップにフィードし、ビジュアルパイプラインデザイナーを介して柔軟なパイプライン構成を可能にします。
Feature	`summarize_text_aggregate()`ファンクションを追加しました。
Feature	PDF から画像への変換パイプラインステップを AIDB に追加し、ページを画像に変換し、OCR PaddleOCR を適用してダウンストリームの埋め込みと取得用のテキストを抽出するパイプラインを介して複雑なテーブルレイアウトをサポートします。
Feature	EDB Knowledge Base コンポーネントと 3 つの新しいコンポーネントのハイブリッド検索サポートにより、Langflow の AIDB カバレッジを拡張しました。EDB Airman MCP Server。 EDB Embedded モデル、CPU ベースのテキストおよび画像の推論用。 EDB Vector Index、EDB ナレッジベースにベクトルインデックスを作成します。
Feature	AI Pipelines は、単一の共有ナレッジベースへの複数の互換性のあるパイプラインの書き込みをサポートするようになり、マルチプルのソースとフォーマットからのハイブリッドナレッジベースを有効にします。ベクターテーブルには、エンベディングごとにソースパイプラインを追跡するための新しい`pipeline_id`列が含まれています。
Feature	メモリ、CPU 割り当て、自動スケーリング、再起動ポリシーを含む構成可能な QoS 制約を使用して、HM によって管理される Kubernetes 環境に Langflow フローを展開するサポートを追加しました。展開されたフローには、完全な可観測性とアラート統合が含まれます。
Feature	AI パイプラインにハイブリッド検索機能を追加し、全文検索、ベクトル/セマンティック検索、述語フィルタリングを組み合わせました。一般的なユースケースの便利な機能とドキュメントの例が含まれています。
Enhancement	Langflow を使用して AI ワークフローを構築し、フローを HM プラットフォームでアセットとして公開し、スタンドアロンジョブとして実行できます。

次のページに続く

表 4 – 前のページからの続き

Type	Description
Enhancement	エラーログとデッドレターキューを追加して、AI パイプラインエラー処理を改善しました。どのファイルまたはレコードが成功し、パイプラインの実行中にどのエラーが発生したかを追跡します。
Enhancement	ノンブロッキングエラー処理を AI パイプラインに追加しました。パイプラインレベルのエラーが発生した場合は停止しますが、エラーが個々のレコードにのみ影響を与える場合、パイプラインは処理を継続するようになりました。`aidb.get_error_logs()`および`aidb.clear_error_logs()`を使用して、エラーを照会および管理します。
Enhancement	オンプレミスデータベースと外部 CloudNativePG クラスターの AI データパイプラインを作成するサポートを追加しました。
Enhancement	セカンダリの場所および HM プラットフォームの外部で実行されているクラスターを含む、HM によって管理される外部 Postgres クラスターの AIDB パイプラインサポートを追加しました。
Enhancement	AI パイプラインデザイナーは、静的なデフォルトを使用する代わりに使用可能なモデルを動的に取得するようになり、特定のプロジェクト、リソース、データベースの組み合わせに基づいてモデルを選択できます。
Enhancement	ビジュアル AI パイプラインデザイナーで直接 PDF to Image パイプラインステップを構成するサポートを追加しました フォーマット、DPI、ページ範囲、注釈レンダリングを含む
Enhancement	AI パイプラインデザイナーで AIDB 7 のサポートを追加しました。これには、既存のナレッジベーステーブルと PGD クラスターを使用したパイプラインのナレッジベースの検証の向上と修正が含まれます。
Enhancement	AIDB 7 のマルチパイプラインナレッジベースモデルをサポートするように AI Pipeline Designer を更新しました。ナレッジベースを複数のパイプラインに関連付けられるようになりました。パイプラインデザイナーは、完全なテーブル形式`vector_schema.vector_table`を使用して KB テーブル参照を表示します。パイプラインの作成は、既存の KB テーブルの選択をサポートします。KB リストビューには、関連するソースパイプラインがハイパーリンクで表示されます。
Enhancement	安全でない接続を使用した外部推論サービスを登録するためのサポートを追加しました。システムは、セキュリティリスクの明示的なユーザーの承認を必要とします。
Enhancement	新しいモデルプロバイダーとしての Hugging Face のサポートを追加しました。Hugging Face モデル名、オブジェクトストレージパス、またはその両方を介して構成可能。
Enhancement	無効なモデル名エラーメッセージ以前は [内部モデルクラスター] ページでのみ使用できましたが、外部推論サービス構成ページに拡張されました。
Enhancement	詳細なエラーメッセージを表示するように AI モデル作成フォームのエラー処理を強化し、モデル作成要件が満たされない場合のユーザーフィードバックを向上させました。
Enhancement	AI ワークフローの一時的な LLM エラーの自動再試行を追加し、推論プロバイダーが一時的な障害を返すときの信頼性が向上しました。

次のページに続く

表 4 – 前のページからの続き

Type	Description
Enhancement	AI ワークフローオーサリングのより厳密な制御のため、AI モデルマネージャーロールを持つユーザーの Langflow アクセスを制限しました。
Enhancement	HM コンソールでタグ名で AI モデルとサービスを検索できるようになりました。
Enhancement	マーメイドチャートを使用してメトリック API からデータをレンダリングするようにチャットボットを強化し、データベースメトリックと監視データの視覚化を向上させました。
Enhancement	GPT-5 バリアントミニ、ナノ、コーデックス、すべての Gemini モデル、Anthropic モデルを含むチャットボットでの追加 AI モデルファミリーのサポートを追加しました。
Enhancement	会話インターフェイスにモデル名が表示されるようになり、各会話でどの AI モデルが使用されているかに関する透明性が向上しました。
Enhancement	推論サービスを登録するときにヘルスチェック検証を追加しました。 HM は、登録中に無効なモデルエンドポイントを検出し、詳細なエラーメッセージを提供するようになりました。テストボタンを使用して登録された推論サービスをテストでき、新しいメトリックエンドポイントが Prometheus モニタリングの資格情報チェック結果を公開します。
Enhancement	知識カットオフの認識を HM ドキュメントスキルに追加しました。ドキュメントスナップショットにはコンパイルのタイムスタンプが含まれるようになり、AI アシスタントは、現在のドキュメントスナップショットでカバーされていない可能性のあるトピックに関する質問に回答するときに、適切な免責事項を表示します。
Change	マルチパイプラインナレッジベース機能のサポートの一環として、EDB Postgres AI エージェント`beacon-agent`からリゾルバーコンポーネントを削除し、AIDB 7 ビューを直接使用するようにエージェントを更新しました。
Change	新しいインスタンス固有のモデルリストのアプローチと互換性がないため、**ソブリン AI** セクションから AI パイプライン **デフォルト設定** ページを削除しました。

可観測性とモニタリング

Type	Description
Feature	**構成** と **セキュリティ** のプロアクティブな推奨カテゴリを追加しました。構成の推奨事項は、`shared_buffers`などの Postgres システムパラメーターを最適化することを提案します。セキュリティに関する推奨事項は、潜在的なセキュリティ問題をフラグし、データベース環境の完全性を継続的に監視するための実用的なガイダンスを提供します。
Feature	メモリ不足 OOM イベント、12 時間を超えるバックアップ、およびディスク/WAL 使用しきい値によって終了したモニタポッドに、新しいクリティカルアラートを追加しました。

次のページに続く

表 5 – 前のページからの続き

Type	Description
Feature	ユーザーは時系列チャートでイベントアノテーションを期間を選択するための区切り文字として使用できるようになり、特定のクラスターイベントに関連付けられたデータ範囲の分離と分析が簡単にできます。
Feature	クラスター状態レポートを追加しました。HM 管理の Postgres クラスターごとに 1 日 1 回自動的に生成されます。レポートは生のメトリックよりも長く保持され、長期的な状態の傾向を追跡し、異常を特定し、パフォーマンスのスナップショットを共有するために使用できます。チャットエージェントは、レポートを取得および比較できます。
Feature	チャットエージェントを介してスキーマデザインの推奨事項を追加しました。チャットエージェントは、実行中のワークロードへのスキーマの適合性を評価し、パフォーマンスを向上させるための実用的な推奨事項を提供します。
Feature	さまざまな Kubernetes クラスターにまたがる複数の外部 PGD グループを単一の論理クラスターとして監視するためのサポートを追加し、統合された可視性とトポロジ管理が有効になります。
Feature	これまでに監視対象データベースで実行されたことのないクエリの検出を追加しました。 HM は、検出を無効にし、過剰な通知を抑制するメカニズムを使用して、1 日以内に新しいクエリを識別します。
Feature	監視対象データベーススキーマの変更を自動的に識別し、アクティビティイベントをクラスターアクティビティフィードに出力するスキーマ変更検出を追加しました。これにより、予期しない変更を見つけ、通知を受信し、変更の前後のデータベース状態を比較できます。
Feature	2 つのユーザー定義の期間の間でデータベースメトリックとクエリ統計を比較するメトリック比較ツールを追加しました。横並びまたはオーバーレイのチャートビュー、フィルター、ズームコントロール、および強調表示された違いを使用します。
Enhancement	自己管理型 PGD クラスターに、レプリケーションマトリックスと遅延モニタリングの完全な機能パリティがあるようになりました。
Enhancement	垂直点線のアクティビティログをホストメトリックチャートに注釈を付ける API 機能を追加し、特定のクラスターイベントと正確な時点でのホストパフォーマンスを相関できるようにします。
Enhancement	GUC `query_advisor.exclude_schema_list`には、クエリアドバイザーの推奨事項から内部データベーススキーマを自動的に除外する事前定義されたデフォルト値が含まれるようになりました。
Enhancement	自己管理 EFM および PGD クラスターのモニタリングのサポートを追加しました。
Enhancement	**ディスクスループット** グラフには、より正確なモニタリングのために読み取りと書き込みのスループットが個別に表示されるようになりました。
Enhancement	HM は、クラスター API のポッドレベルのステータス条件を介して、より深い展開の洞察を提供するようになりました。
Enhancement	CNPG-GC クラスターを含む外部 PGD クラスターの可観測性サポートを拡張しました。

次のページに続く

表 5 – 前のページからの続き

Type	Description
Enhancement	到達不能な場合に 24 時間ごとに IT リンクへの再接続を試行する再試行ロジックを備えたビーコンサーバーの新しい `EnableEDBUsageReporting`構成オプションを追加しました。
Enhancement	ユーザーの変更、自動スケーリング、またはエージェントオートメーションによってクラスター仕様が更新されると、アノテーションが自動的に生成され、オブザーバビリティビューに表示されます。
Enhancement	Prometheus クエリ要求の構成可能な制限を追加して、広範すぎるクエリによるリソースの枯渇を防ぎます。ストアコンポーネントが `request-samples`および `request-series`制限をサポートするようになりました。
Enhancement	アノテーション API を追加して、モニタリングダッシュボードで選択したアクティビティイベントのマーキングを有効にし、自動操作の基盤を提供します。
Enhancement	EDB Postgres AI エージェントによって監視されるオンプレミスの Postgres インスタンスの WAL メトリック収集を追加しました。
Enhancement	取り込み前にデータベース接続を検証するための `validate`コマンドを EDB Postgres AI エージェント `beacon-agent`に追加しました。
Enhancement	HM は、自己管理の EDB フェールオーバーマネージャークラスターからクラスターステータスメトリックを収集および表示するようになりました。
Change	デフォルトのアラート評価間隔を 2 分に更新しました。
Change	Thanos、Prometheus、および Grafana 展開にトレースサポートを追加し、PromQL クエリの詳細なパフォーマンス分析と監視システムパフォーマンス構成のより適切な評価が可能になります。
Change	EDB Postgres AI エージェント展開から CPU 制限を削除して、調整を防止し、監視パフォーマンスを向上させます。
Change	Thanos Query でデフォルトで自動ダウンサンプリングが有効になり、必要に応じて低解像度データを自動的に使用することにより、大きな時間範囲のクエリパフォーマンスが向上します。
Change	サイドカーで消えたピアの古いメトリックシリーズを自動的に有効期限が切れるように Istio を構成し、ピアの出入りに伴う時間の経過による無制限のメトリックの増加を防ぎます。
Change	すべての接続とサービスがアクティブで EDB Postgres AI エージェントが完全に起動した後のみに準備ができているステータスを報告するようにビーコンの準備状況プローブを更新しました。
Change	チャートに送信される時間データに適切なタイムゾーン情報が含まれるか、正確なチャートレンダリングのためにユーザーの現地時間に変換されるようにバックエンドロジックを変更しました。

HM コンソール

Type	Description
Feature	AI を活用したクエリ生成をクエリエディターに追加しました。 **Generate with AI** を選択して、エステート全体のデータをターゲットとする自然言語プロンプトを使用して SQL クエリーを作成します。
Enhancement	PDF をダウンロードせずに HM コンソールでデータベースの状態レポートを直接表示する機能を追加しました。
Enhancement	Web ログインセッションのアイドルタイムアウトを含む HM 認証ポリシーの構成インターフェイスを **Settings** セクションに追加しました。
Enhancement	**Invalidate User Sessions** アクションを ID プロバイダー管理インターフェイスに追加し、管理者は特定の ID プロバイダーのすべてのアクティブなセッションを無効にできます。
Enhancement	ロール割り当てインターフェイスにフィルタを追加して、IdP グループマッピングを介して割り当てられたロールを除き、手動で割り当てられたロールのみを表示します。
Enhancement	`beacon.enterprisedb.com/exclude-storage-from-dbaas=true`ラベルを追加して、共有 Kubernetes 環境の **Create Cluster** ドロップダウンから特定の `StorageClass`または`VolumeSnapshotClass`リソースを除外します。
Enhancement	HM コンソールで、クラスターまたはグループ ID でクラスターをフィルターできるようになりました。
Enhancement	HM コンソールに検索可能なモデルサービスピッカーを追加しました。
Enhancement	HM コンソールにカスタムタイムゾーンで日付を表示できるようになりました。
Enhancement	以前はマウスのインタラクションのみをサポートしていたインタラクティブ要素にキーボードイベントハンドラーを追加することにより、HM コンソールでのキーボードのアクセシビリティが向上しました。

アナリティクス

Type	Description
Feature	HM コンソールから直接階層化テーブルを構成する機能を追加しました。ユーザーは、コンソール内で既存のテーブルを選択して、階層テーブル構造に変換できます。
Feature	Postgres Analytics Accelerator が Postgres 18 をサポートするようになりました。
Enhancement	階層テーブルの洞察と視覚メトリックを追加し、階層テーブル構成の視覚的な証拠を提供し、診断をコマンドラインから HM コンソールに移動します。
Enhancement	Postgres Analytics Accelerator は、Azure Data Lake Storage および Azure Blob Storage の Delta および Iceberg テーブルの読み取りおよび書き込みをサポートするようになりました。
Enhancement	HM に同梱されている PGAA バージョンを 1.9.1 に更新しました。

次のページに続く

表 7 – 前のページからの続き

Type	Description
Enhancement	Lakekeeper ウェアハウスの作成を強化して、AWS Security Token ServiceSTS をサポートしていないオブジェクトストレージ環境をサポートします。 S3 互換ストレージで最初の試行が失敗した場合、システムは STS を無効にしてウェアハウスの作成を自動的に再試行するようになりました。

マイグレーション

Type	Description
Platform support	HM 移行機能が、エアギャップオンプレミス環境 RKE および RHOS 展開で完全にサポートされるようになりました。
Feature	HM 管理クラスターのプロビジョニング、スキーマの移行、スナップショットとストリーミングデータの移行の実行を含む、Oracle および Postgres ソースから Postgres 宛先への AI を活用したエンドツーエンドのデータベース移行のための Langflow テンプレートを追加しました。 Oracle ソースでは、移行フローを実行する前に、移行ポータルを介してスキーマ変換が必要です。
Feature	HM コンソールの **ソース** および **宛先** 移行ページには、データ移行サービス DMS エージェント `cdcagent`が HM インスタンスに接続するために必要な `RW_SERVICE_HOST`環境変数が表示され、この値のセルフサービスの取得が可能になります。
Feature	ソースデータベーススキーマが EDB Postgres AI エージェント`beacon-agent`によって取り込まれると、HM は移行ポータルプロジェクトを自動的に作成するようになり、ほとんどの場合、移行ポータルプロジェクトを手動で作成する必要がなくなりました。
Feature	ソース Postgres データベースの論理レプリケーションスロットの状態を追跡するレプリケーションスロットヘルスモニタを追加し、ダウンロード可能なメトリックレポートで各スロットのヘルスステータス、再生ラグ、およびディスクリテンションを報告します。
Feature	スキーマ変更検出とは無関係に実行される専用のメトリックフェッチャーを追加し、定期的なポーリング間隔での WAL 生成レートなどのデータベースメトリックの継続的なモニタリングを有効にします。
Feature	WAL 生成への各テーブルの寄与を推定するテーブルごとのタプル変更率評価を追加し、ソース Postgres データベースのより詳細な移行計画を可能にします。
Feature	ソース Postgres データベースへの WAL 蓄積を監視するグローバル WAL 生成速度評価を追加し、移行スナップショット中のストレージ容量要件の推定に役立ちます。
Feature	Postgres テーブルの主キー、*REPLICA IDENTITY*、および`TOAST`構成の危険な組み合わせをチェックする移行評価を追加し、Data Migration ServiceDMS がサイレントに変更を見逃す可能性のあるシナリオを特定します。
Feature	DMS ベースの CDC でサポートされていないまたは危険な Postgres データ型を特定する移行評価を追加し、移行を開始する前に潜在的な互換性問題に関する早期警告を提供します。

次のページに続く

表 8 – 前のページからの続き

Type	Description
Feature	JSON 形式で移行ポータル評価レポートをダウンロードするためのパブリック API エンドポイントを追加しました。
Feature	多対多の関係でデータベースとスキーマに関連付けられているアプリケーションに関するメタデータを入力および表示する機能を追加し、移行の計画と優先順位付けのためのより良いコンテキストを提供します。
Feature	EDB Postgres AI エージェント *beacon-agent* が、Oracle `TNSNAMES.ORA` ファイル、CSV ファイル、または AWS RDS 自動検出から Oracle および Postgres データベース接続を自動的に構成する機能を追加しました。
Feature	HM を介して Oracle または Postgres ソースから Postgres 宛先に直接データベーススキーマを移行する統合スキーマ移行機能を追加しました。HM 管理および外部管理の宛先のサポート、および AI エージェント自動化のための API アクセス。
Feature	EDB Postgres Advanced Server との互換性について Oracle アプリケーション SQL ステートメントを評価する機能を追加し、Oracle のトレースファイルとシステムビューから抽出された SQL 、および HM コンソールまたは API を介して提供されるステートメントをサポートしています。
Feature	HM コンソールの移行の詳細ページに **メトリック** タブを追加しました。移行がアクティブである間、タブはその移行に事前フィルタリングされた Grafana ダッシュボードにリンクし、スループット行/秒、バイト/秒、バッチレイテンシー、コミットレイテンシー、およびバッチサイズの読み取り側と書き込み側の両方のバッチサイズを表示します。 DMS エージェント。
Enhancement	HM 移行には、移行 **エラー履歴** タブが含まれ、エラーソース、アクティビティステータス、タイムスタンプ、トラブルシューティングに役立つ完全なログ出力を含む詳細なエラーメッセージを提供します。
Enhancement	Oracle データベースの詳細ページには、移行ポータルプロジェクトによって報告されたデータベースの互換性を説明するスキーマ互換性の概要が含まれるようになりました。
Enhancement	移行ポータルには、依存オブジェクトとスキーマに関する情報が含まれ、リレーション参照の欠如など、関連する非互換性の問題を防ぎます。
Enhancement	Postgres データベースの詳細ページの **Assessment** タブには、移行の複雑さに影響を与える可能性のある構成項目をリストした新しいセクションが含まれます。インストールされた拡張機能、外部サーバー、外部テーブル、テーブルスペース、インストールされたプロシージャー言語。
Enhancement	**移行>移行** リストビューには、 **移行タイプ** および **移行の進行状況** 列が含まれるようになりました。 Migration Progress 列には、移行ごとのライブメトリックが含まれる合計テーブル、完了したスナップショットテーブル、および移行された累積行数が表示されます。移行の詳細ビューの **テーブル** タブには、テーブルごとのスナップショットステータス、マップされた列、移行された行、および移行速度行/秒が表示されます。
Enhancement	EDB Postgres AI エージェントと関連サービスを強化して、多数の外部データベースの取り込み、評価、および同時移行のサポートを強化しました。

次のページに続く

表 8 – 前のページからの続き

Type	Description
Enhancement	DMS は、HM 管理の Postgres 18 データベースへの移行、および外部管理の Postgres 18 データベースからの移行をサポートすることがテストおよび認定されています。
Enhancement	詳細ビューの移行評価は、Postgres ソースのレプリカ ID に関するより明確な診断を提供するようになりました。
Enhancement	[**Estate > Migrations**] タブでの移行ステータスの検索を改善して、結果が速くなりました。
Enhancement	進行中の移行のエラーログは、トラブルシューティングを簡単にするための分レベルのデータを提供するようになりました。
Enhancement	データの移行に使用できるソースデータベースのスキーマとテーブルをより詳細に制御するために、新しい DMS エージェント環境変数を追加しました。
Enhancement	テーブルスペースの使用状況を表示するセクションを Postgres データベースの詳細ページに追加し、各テーブルスペースのスキーマ別のオブジェクトタイプの数をリストします。
Enhancement	スナップショットフェーズ後まで主キーの適用を遅延することにより、DMS スナップショットのパフォーマンスが向上しました。複合主キーは、スナップショット後の順序で収集および適用され、スナップショットフェーズとストリーミングフェーズ間の重複排除がサポートされています。
Enhancement	大量ダウンロードをサポートするように移行ポータル評価レポート API を拡張しました。新しいエンドポイントは、複数のデータベースリソース ID を受け入れ、単一の JSONL ファイルで評価レポートを返します。
Enhancement	Postgres データベースが移行用に登録されると、そのスキーマが自動的に取り込まれ、解析され、移行ポータルのバックエンドデータベースに保存され、新しい API 動作を介して移行に利用できるようになります。
Enhancement	移行エージェントは、Oracle スキーマを抽出するときに上流のキャンセルシグナルを尊重するようになり、よりクリーンなシャットダウン動作が有効になります。
Enhancement	スナップショットと CDC レプリケーションモードの両方での `DATE`、*TIME*、*TIMESTAMP*、*INTERVAL*、*UUID*、*JSON*、*XML*、および `POINT` タイプに基づいて PostgreSQL `DOMAIN` タイプをレプリケートするサポートを追加しました。
Enhancement	`schema_and_data`移行のコンカレントスキーマファイナライズを追加しました。スナップショットステージが完了したら、**Finalize Schema Concurrently** を選択して、ノンブロッキング DDL バリアントを使用してバックグラウンドでインデックス、外部キー、およびその他のスキーマオブジェクトを適用できるため、カットオーバー時間を大幅に削減できます。
Enhancement	DMS での Postgres から Postgres へのスキーマの移行を拡張して、スキーマの移行中に Postgres 固有のデータベースオブジェクトカスタムドメインタイプを処理し、Postgres データベース間の完全なエンドツーエンドのスキーマ移行の正しい依存関係の順序を保証します。

次のページに続く

表 8 – 前のページからの続き

Type	Description
Enhancement	統一スキーマの移行、合理化されたエージェントコンポーネント、自動パスワード生成、データベースセットアップ、および Llama 3 Nemotron 49B および Nano 4B モデルのサポートを活用したアーキテクチャの改善により、Simple Migration Agent Langflow テンプレートを強化しました。
Enhancement	Simple Migration Agent Langflow テンプレートはパスワードなしで、グローバルシークレット管理を使用して、大規模言語モデル LLM に公開せずに資格情報を処理します。
Enhancement	移行スコープがスキーマのみに設定されている場合、テーブルのロードと検証をスキップするように移行作成フォームを最適化しました。
Change	改善された AI モデルで移行コパイロットを更新しました。GPT 5.2 がデフォルトモデルになり、代替として検証済みの Nemotron 49B Llama3 がサポートされています。
Change	EDB Postgres AI エージェント *beacon-agent* でのデータベースごとのバイナリパス構成のサポートを追加しました。 `beacon_agent.yaml`のデータベースエントリごとにカスタム`pg_bin_path`を指定できるようになりました。これにより、単一の`beacon-agent`が、異なる`pg_dump`および`pg_dumpall`バイナリを必要とする複数の PostgreSQL フレーバーに接続できます。
Change	DMS *publications* **にテーブルを登録する前に、***Postgres* **テーブル** *`REPLICA IDENTITY`* **のプリフライト検証を追加しました。主キーまたは適切な** *`REPLICA IDENTITY`* **設定のないテーブルは、移行の開始後に** *`UPDATE`*/`DELETE`操作を失敗させるのではなく、明確な修復手順でセットアップ時に拒否されます。
Change	PostgreSQL 18 *pg_dump* **および** *`pg_dumpall`* **出力解析のサポートを追加しました。新しい** *psql* **メタコマンド** *`restrict`* **および** unrestrict`および改善された ROLE オブジェクト分類を含みます。
Change	強化された Echo ベースコンテナイメージを使用するように移行コパイロットを更新し、セキュリティ体制を向上させ、脆弱性の露出を削減します。
Deprecation	`domain_name`パラメーターは、以前のイノベーションリリースで `componentsParameters["transporter-rw-service"]`から `globalParameters.dms_domain_name`に移動しました。 HM 1.3.x の最新パッチバージョンから直接アップグレードする場合、この移行は`hmupgrader`ツールによって自動的に処理されます。イノベーションリリーストラックをたどったユーザーは、この変更を既に適用しているでしょう。

2.1.1.2 Contributing Innovation Releases

2.1.1.2.1 Hybrid Manager 2025.11.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2025 年 11 月 07 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデル

の詳細については、*Hybrid Manager* デュアルリリース戦略 を参照してください。

- **GenAI Builder インターフェイス**ナレッジベースを作成および管理するためのビジュアルツールを導入し、HM 内で直接モデルのレビューと構成を有効にしました。
- **階層化テーブルの可観測性と構成** HM コンソールから直接階層化テーブル構成を有効にし、レイクハウスのクエリと構成の可観測性の強化を提供しました。
- **高度な高可用性および分散高可用性クラスターのスイッチオーバー** HM コンソールから直接書き込みリーダースイッチオーバーを開始する機能を追加しました。
- **移行の透明性とデバッグ**テーブルのステータス、リアルタイムの移行行数、エラーソース、ログ出力を含む、HM コンソールに詳細な移行の進行状況の可視性とエラー履歴を直接追加しました。
- **データベース構成とセキュリティの推奨事項** データベースの**推奨事項**タブを展開して、データベース構成とセキュリティのプロアクティブなチェックを含めます。

GenAI Builder UI でのモデルとナレッジベースの管理

ナレッジベース管理ユーザーは、HM 内で直接すべての AIDB ナレッジベースを作成、管理、および表示できるようになりました。データベースに作成された既存のナレッジベースはすべて、GenAI Builder インターフェイスから表示されるようになりました。

モデルのレビューと構成使用可能なモデルを確認し、インターフェイスから直接新しいモデルを追加できるようになりました。このインターフェイスは、コンテキストウィンドウの制限や最適なユースケースを含む重要なモデルメタデータを提供し、RAG パイプラインのモデル選択と構成を簡素化します。

アナリティクスとレイクハウスの統合

HM コンソールで構成可能な階層化テーブル階層化テーブルをセットアップする複雑なタスクが、単純な HM コンソールアクションになりました。ユーザーは、HM コンソール内で直接既存のテーブルを選択して、階層テーブル構造に変換できるようになりました。

階層テーブルの可観測性の向上階層テーブルとレイクハウスクエリの洞察とビジュアルメトリックを追加しました。これにより、階層テーブル構成のセットアップと予想される機能の視覚的な証拠が提供され、診断がコマンドラインから HM コンソールに移動します。

統合レイクハウスの統合 接続された外部 Apache Iceberg テーブルを、専用のレイクハウスまたはカタログページで表示できるようになりました。この統合されたビューにより、すべてのデータを一元的な場所で表示および管理できます。

合理化されたスイッチオーバー

HM コンソールからのスイッチオーバー Advanced High Availability および分散高可用性クラスターの書き込みリーダースイッチオーバーを、クラスターの詳細ビューから直接開始できるようになりました。これにより、エキスパートのみのタスクが低リスク操作に変わります。

ノードの可視性の強化クラスター詳細ビューには、高度な高可用性クラスターと分散高可用性クラスターのすべてのグループノードがリストされ、スイッチオーバー操作の前後の現在のトポロジとステータスへの重要な可視性を提供します。

移行の進行状況の可視化

HM コンソールの詳細な進行状況 HM は、包括的なリアルタイムの移行の進行状況情報を HM コンソールに直接表示するようになりました。

テーブルのステータスと行数移行される行のリアルタイム数や移行率など、移行されるすべてのテーブルのステータスを可視化します。これにより、ログや外部ツールを確認することなく、データ転送の進行状況を追跡できます。

移行エラー履歴

HM コンソールの詳細なエラーメッセージ HM 移行には、移行 **エラー履歴**タブが含まれるようになりました。これにより、エラーソース、アクティビティステータス、タイムスタンプ、トラブルシューティングに役立つ完全なログ出力を含む詳細なエラーメッセージが提供されます。

Postgres の移行評価の詳細

Postgres データベースの詳細ページの強化外部 Postgres データベースが移行のために登録および評価されると、**Estate** ビューは、拡張機能、外部サーバー、外部テーブル、プロシージャー言語、テーブルスペースに関する情報を含む拡張された詳細を提供するようになりました。

スキーマの互換性の概要

HM コンソールのスキーマ互換性の概要 Oracle データベースの詳細ページには、移行ポータルプロジェクトによって報告されたデータベースの互換性を説明する概要が含まれるようになりました。

Migration Portal 統合の機能強化

JSON ダウンロード移行ポータルスキーマ評価レポートの JSON 形式バージョンをダウンロードする機能を追加しました。

依存オブジェクトのレポートの改善移行ポータルには、依存オブジェクトとスキーマに関する情報が含まれ、関連する非互換性の問題を防止します。たとえば、“リレーション x は終了しない”。

プロアクティブなレコメンデーション

新しい推奨カテゴリ システムは、2 つの新しい重要なカテゴリ **構成** と**セキュリティ** の推奨事項を生成するようになりました。

構成のチューニング 推奨事項には、`shared_buffers` チューニングなどの Postgres システムパラメーターを最適化するための提案が含まれ、ボトルネックを防止し、管理対象データベースのパフォーマンスを向上させるのに役立ちます。

プロアクティブなセキュリティチェックこのタブは、潜在的なセキュリティ問題にフラグを立て、それらに対処するための実用的なガイダンスを提供し、データベース環境の完全性の継続的な監視を保証します。

2.1.1.2.2 Hybrid Manager 2025.12.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2025 年 12 月 11 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

- **プラットフォームの安定性と柔軟性** Microsoft Azure Kubernetes ServiceAKS にハイブリッドマネージャー HM を展開できるようになりました。このサポートは AI パイプラインまで拡張され、Postgres Analytics Accelerator で Azure Data Lake Storage および Blob Storage のテーブルを照会できるようになります。ただし、GenAI Builder のサポートと AKS の移行機能は、TDE 対応データベースでの Azure Key Vault の使用のサポートと同様に、将来のリリースで追加されます。
- **セキュリティと ID 制御の向上** HM コンソールから直接独自の ID プロバイダー IdP を構成および統合できるようになり、ユーザーアクセス管理を合理化できます。
- **移行の可視性と評価** Postgres データベース評価には、移行の複雑さの可視性を提供する追加の構成項目が含まれるようになりました。追加の HM コンソール情報により、Oracle および Postgres ソースのアクティブなデータ移行と完了したデータの移行をより適切に追跡できます。
- **大規模なデータベースの移行のサポート** EDB Postgres AI エージェントと DMS 構成を強化して、スキーマの取り込み、データベース評価、および多数の外部データベースの同時移行をより適切かつ効率的にサポートします。

機能と機能強化

Hybrid ManagerHM プラットフォーム

Type	Description
Feature	Microsoft Azure Kubernetes ServiceAKS での HM 展開のサポートを追加しました。
Feature	HM コンソール> **設定** > **ID プロバイダー** から独自の ID プロバイダーまたは IdP を構成する機能を追加しました。
Enhancement	マルチロケーションマルチ DC 展開手順を簡素化しました。現在、マルチ DC トポロジは、主に `primary.yaml`および `secondary.yaml`構成ファイルを介して制御され、標準の helm プラクティスをサポートしています。
Enhancement	HM は、自己管理の EDB フェールオーバーマネージャークラスターからクラスターステータスメトリックを収集および表示するようになりました。
Enhancement	HM は、HM プロジェクトごとの専用オブジェクトストレージの構成をサポートするようになりました。

マイグレーション

Type	Description
Feature	HM コンソールの **ソース** および **宛先** 移行ページが更新され、DMS エージェント `cdcagent`としてパッケージが HM インスタンスに正常に接続するために必要な`RW_SERVICE_HOST`環境変数設定が表示されます。この変数を表示すると、ユーザーは HM 管理者に連絡する代わりに、セルフサービスの方法でこの情報を取得できます。
Feature	Oracle データベーススキーマが EDB Postgres AI エージェント`beacon-agent`としてパッケージ化されてハイブリッドマネージャーに取り込まれるときに、HM は移行ポータルプロジェクトを自動的に作成するようになりました。この機能により、ほとんどの場合、ユーザーは移行ポータルプロジェクトを手動で作成する必要がなくなります。
Enhancement	Postgres データベースの詳細ページの **評価** タブが更新され、選択したデータベースの構成項目をリストする新しいセクションが含まれます。HM 管理データベースへの移行の複雑さに影響を与える可能性があります。次の新しいセクションが追加されました。 - インストールされた拡張機能 - 外部サーバー - 外部テーブル - テーブルスペース - インストールされたプロシージャー言語
Enhancement	**移行>移行** リストビューが **移行タイプ** および **移行進行状況** 列を追加して機能強化され、アクティブなすべての完了したデータ移行の移行進行状況のモニタリングが容易になりました。 **Migration Type** 列は、リストされた移行が **スナップショット** のみまたは **スナップショット+ストリーミング** タイプのデータ移行であるかどうかを示します。 **Migration Progress** 列は、移行のために選択されたテーブルの数、スナップショットフェーズを完了したテーブルの数、および移行された行数を示します。アクティブなデータの移行の場合、スナップショットされたテーブルと移行された行の数の値は、移行の進行につれて自動的に更新されます。
Enhancement	EDB Postgres AI エージェントと関連サービスを強化して、大量の外部データベースの取り込みと評価をより適切かつ効率的にサポートします。データ移行サービス DMS も改善され、大量の同時データ移行のサポートが向上しました。
Enhancement	データ移行サービス DMS は、HM 管理の Postgres 18*クラスターデータベースへの移行、および外部管理の Postgres 18* データベースへの移行をサポートすることがテストおよび認定されています。 HM 移行ポータルも更新され、Oracle データベースの EDB Enterprise PostgresOracle 互換 18 データベースへの評価と移行をサポートしました。

AI ファクトリー

Type	Description
Feature	AI パイプラインの Azure での HM 展開のサポートを追加しました。 GenAI Builder のサポートは、将来のリリースで追加される予定です。
Feature	AI Factory は、Postgres 18*をサポートするようになりました。
Feature	AI Pipelines には、HM に統合されている、AI データ準備パイプラインを開発、展開、および管理するための Web ベースのグラフィカルデザインツールである Visual Pipeline Designer が含まれるようになりました。
Feature	EDB は、Postgres の高度なオープンソースのベクトル検索拡張機能である VectorChord のパッケージ、セキュリティメンテナンス、およびサポートサービスを追加しました。 VectorChord パッケージは、EDB リポジトリを介して EDB 顧客がすぐに利用できるようになります
Feature	GenAI Builder には、ユーザーが環境とプロジェクト間でアプリケーションを転送できる機能である Resource Transfer が含まれるようになりました。
Feature	GenAI Builder は HM プロジェクトと統合され、ユーザーがエージェントアプリケーション開発プロジェクトでプライベートにコラボレーションできるようになりました。 以前の GenAI Builder データレイク、データソース、構造、ツール、その他の情報は、HM のデフォルトプロジェクトにあります。
Feature	GenAI Builder は、アシスタントおよびその他のプラットフォーム機能のプロジェクトごとの AI モデル構成をサポートするようになりました。
Enhancement	GenAI Builder SDK で構築されたアプリケーションの推奨展開パスに関する手順を追加しました。
Enhancement	GenAI Builder は、リソースの分離を向上させるために改善されたユーザー定義のツールと構造サンドボックスの恩恵を受けるようになりました。
Enhancement	GenAI Builder の Threads API は、大量操作をサポートするようになりました。
Enhancement	GenAI Builder ナレッジベースでは、国際文字エンコードセットのサポートが強化されました。

アナリティクスアクセラレーター

Type	Description
Feature	Postgres Analytics Accelerator は、Postgres 18*をサポートするようになりました。
Enhancement	Postgres Analytics Accelerator は、Azure Data Lake Storage および Azure Blob Storage の Delta および Iceberg テーブルの読み取りおよび書き込みをサポートするようになりました。

!!!note * Postgres は、EDB でサポートされているすべてのディストリビューションを指します

- PostgreSQL
- EDB Enterprise Postgres - 以前は EDB Postgres Extended ServerPGE
- EDB Enterprise PostgresOracle 互換-以前の EDB Postgres Advanced Server !!!

2.1.1.2.3 Hybrid Manager 2026.1.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2026 年 1 月 15 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

- **ロードバランサー構成の強化**ロードバランサーの新しいグローバル構成パラメーターを導入し、ネットワーキングとトラフィック管理のより詳細な制御を提供します。
- **Microsoft AKS のサポートの拡張** データ移行サービス DMS が Microsoft AKS を完全にサポートするようになり、Hybrid ManagerHM が Azure の管理対象 Kubernetes 環境に展開されているときに移行ワークロードを実行できます。
- **合理化された移行診断** [**エステート > 移行**] タブに登録された Postgres データベースの詳細ビューは、レプリカ ID に関するより明確な診断を提供するようになりました。[**移行**] タブには、移行ステータスの高性能検索機能もあり、大規模な環境全体の進行状況を追跡するのに役立ちます。
- **データベースの可観測性** [**Monitoring**] タブには、データベースサイズの積み上げグラフなど、データの視覚化が強化されました。
- **包括的なイベントと推奨事項の可視性**ユーザーのメニューからアクセス可能なグローバル **アクティビティログ**セクション内のカスタムおよびモデルクラスターイベントへのより深い可視性を追加しました。また、クエリアドバイザーを更新して、よりクリーンで関連性の高いクラスターパフォーマンスの推奨事項のために内部システムオブジェクトを自動的に除外します。

機能と機能強化

Hybrid ManagerHM プラットフォーム

Type	Description
Feature	Hybrid ManagerHM は、モデルコンテキストプロトコル MCP を介して API 機能を公開するようになりました。この統合により、開発者と AI エージェントは、エージェントのユースケースでハイブリッドマネージャー API を使用できるようになります。たとえば、これらの機能を Langflow とともに使用して、データベースエージェントの構成を合理化できます。
Feature	HM コンソールの Launchpad に Langflow が含まれるようになりました。この統合を使用して、HM プラットフォーム内で直接 AI 搭載エージェントとワークフローを構築およびテストできます。
Enhancement	ロードバランサーの新しいグローバル構成パラメーターを追加しました。
Enhancement	HM クラスター API は、プロビジョニングが失敗した場合の可視性を向上させ、セルフサービスの診断と解決までの時間の短縮を可能にします。
Enhancement	**設定 > ID プロバイダー** 管理ページのフィールドの処理を改善しました。

次のページに続く

表 13 – 前のページからの続き

Type	Description
Deprecation	グローバルパラメーター`TRANSPORTER_RW_SERVICE_DOMAIN_NAME`の名前が`GLOBAL.DMS_DOMAIN_NAME`に変更されます。アップグレードプロセス中に`GLOBAL.DMS_DOMAIN_NAME`が明示的に設定されていることを確認します。シークエンシャルなアップグレード戦略に従った場合、これは *Troubleshooting* 中に既にカバーされています。さらに、 *Troubleshooting* に記載されているように、すべての Transporter コンポーネントが HM の現在のバージョンに更新されていることを確認します。

可観測性とモニタリング

Type	Description
Enhancement	HM 可観測機能は、データベースサイズの積み上げグラフ、履歴イベントマーカー、およびクエリテーブルをチャート選択と同期する機能でアップグレードされました。
Enhancement	ホストメトリックチャートに注釈を付けるための API 機能を追加しました。クラスターの **モニタリング** タブのホスト履歴グラフ内に、アクティビティログを垂直点線として表示できるようになりました。これにより、特定のクラスターイベントを正確な時点でホストパフォーマンスの変動と関連付けることができます。
Enhancement	Grand Unified ConfigurationGUC `query_advisor.exclude_schema_list`に、事前定義されたデフォルト値が含まれるようになりました。これらのデフォルトでは、内部データベーススキーマが自動的に除外され、クエリアドバイザーの推奨事項クラスターの **推奨事項** タブに表示されるインデックスおよび統計メトリックがユーザー定義データに焦点を当てたままで、内部システムオブジェクトによって乱雑にならないようにします。
Enhancement	Debian での EDB Postgres AI エージェントのアンインストールがクリーンになり、警告が表示されなくなりました。
Enhancement	ユーザーのメニューからアクセスできるグローバル **アクティビティログ** セクションに、追加のイベントカスタムイベントとモデルクラスターイベントの可視性を追加しました。
Enhancement	メモリ不足 OOM イベント、12 時間を超えるバックアップ、およびディスク/WAL 使用しきい値によって終了したモニタポッドに、新しいクリティカルアラートを追加しました。自己管理型 PGD クラスターに、レプリケーションマトリックスと遅延モニタリングの完全な機能パリティがあるようになりました。
Enhancement	HM は、クラスター API のポッドレベルのステータス条件を介して、より深い展開の洞察を提供するようになりました。

マイグレーション

Type	Description
Feature	Data Migration ServiceDMS は、Microsoft AKS での展開を完全にサポートするようになりました。
Enhancement	移行の *Troubleshooting* **エステート > 移行** で利用可能な評価は、Postgres ソースのレプリカ ID に関するより明確な診断を提供するようになりました。
Enhancement	[**Estate > Migrations**] タブでの移行ステータスの検索を改善して、結果が速くなりました。
Enhancement	**移行 > 移行** セクションでプロジェクトの進行中の移行で利用可能なエラーログは、トラブルシューティングを容易にするための分レベルのデータを提供し、より迅速な問題の解決が可能になります。
Enhancement	データの移行に使用できるソースデータベースのスキーマとテーブルのより詳細な制御を提供する新しい DMS エージェント環境変数を追加しました。

2.1.1.2.4 Hybrid Manager 2026.2.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2026 年 2 月 12 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

- **選択的インストール** —必要な特定の機能のみを有効にすることにより、HM 展開をカスタマイズできるようになりました。これにより、リソースの使用量が大幅に削減され、必要な機能のみを展開することにより HM 展開の管理が簡素化されます。
- **統合されたネイティブチャットボット**このリリースでは、HM コンソール内にネイティブチャットインターフェイスを導入し、チャットを介して直接質問に回答し、操作に関するガイダンスを取得できます。
- **3 か所のクラスター展開** —アーキテクチャのサポートの強化により、Postgres データベースクラスターを 3 つの地理的に離れた場所に展開できるようになり、ゼロデータ損失構成が有効になります。
- **外部 AI モデルのサポート** —AI ファクトリーは外部モデルサービングをサポートするように拡張され、HM 内で独自のリモート AI モデルを接続して使用できます。
- **自己管理クラスターの統合モニタリング** —オブザーバビリティが自己管理の EFM および PGD クラスターに拡張され、HM コンソールを介してより幅広い自己管理データベースを監視できます。

機能と機能強化

Hybrid ManagerHM プラットフォーム

Type	Description
Feature	HM は選択的インストールをサポートするようになりました。インストール時以降に、必要な機能オブザーバビリティと DBaaS、移行、分析、AI のみを有効にすることにより、展開をカスタマイズします。これにより、リソースの使用量が削減され、コンポーネント管理が簡素化されます。
Feature	基本的な情報クエリ、直接 SQL 実行、およびデータベース管理をサポートする新しいネイティブチャットボットインターフェイスを追加しました。
Enhancement	地理的に離れた 3 つの場所に Postgres データベースクラスターを展開するためのサポートを追加し、レプリカクラスターまたはデータグループを他の場所に展開できる高度なアーキテクチャを有効にします。この更新により、同期レプリケーションを介したゼロデータ損失アーキテクチャが可能になります。さらに、HM は、クォーラムを維持し、自動フェイルオーバー中のスプリットブレインシナリオを防止するための 3 番目の場所への PGD-X Witness Group の展開をサポートするようになりました。

可観測性とモニタリング

Type	Description
Feature	ユーザーは時系列チャートでイベントアノテーションを、期間を選択するための区切り文字として使用できるようになりました。これにより、特定のクラスターイベントに関連するデータ範囲の分離と分析が簡単になります。
Enhancement	HM 内の自己管理 EFM および PGD クラスターのデータベースモニタリングのサポートを追加しました。 EDB Postgres AI エージェントは、メトリック、ログ、イベントをこれらの外部クラスターから中央の可観測性プラットフォームに収集し、HA クラスターの状態の統合ビューを提供します。

AI ファクトリー

Type	Description
Feature	モデル提供のための外部サービスへのアクセスのサポートが追加され、独自のリモート AI モデルに接続して使用できます。既存のモデルインフラストラクチャを使用して、複数のモデルをデータパイプライン、GenAI Builder アプリケーション、および HM チャットボットへの統合を簡素化します。

マイグレーション

Type	Description
Enhancement	テーブルスペースの使用状況を表示するための新しいセクションが Postgres データベースの詳細ページに追加されました。このビューは、各テーブルスペースのスキーマ別のオブジェクトタイプの数をリストし、ユーザーが外部管理の Postgres データベースの移行の複雑さを評価するのに役立ちます。
Platform support	HM 移行機能がエアギャップオンプレミス環境で完全にサポートされるようになりました。これにより、外部ネットワークアクセスを必要とせずに、RKE および RHOS 展開ですべての移行機能が正しく機能することが保証されます。
Platform support	HM 移行機能が Azure Kubernetes ServiceAKS で完全にサポートされるようになり、Azure 環境での展開で一貫した機能とパフォーマンスが保証されます。
Bug fix	HM 2025.12 リリースで導入された移行ポータルの問題を修正し、既存のデータベースをオンラインスキーマの移行の宛先として使用できませんでした。
Bug fix	移行ポータル分析機能を修正して、 DDL での `PIVOT`演算子の使用を識別し、互換性のない機能として報告するようになりました。
Bug fix	移行ポータル分析機能を修正して、互換性のない機能として`IS JSON CHECK CONSTRAINT`を不正確に報告しなくなりました。

2.1.1.2.5 Hybrid Manager 2026.3.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2026 年 3 月 12 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

- **HM オートメーションの紹介**このリリースでは、自律データベース管理への大きな一歩である `Automations` エンジンをデビューさせます。最初に使用可能なタスクは `Disk Auto-Scaling` で、これにより、システムはディスク使用量を監視し、プロジェクトレベルのメンテナンスウィンドウと人間参加者の承認に基づいて、プライマリストレージを動的に拡張できます。
- **AI Factory の新しい「Summarize」 SQL 集計ファンクション** 標準の SQL `GROUP BY` クエリで `summarize_text_aggregate()` ファンクションを使用できるようになりました。これにより、複数の行のテキストを単一の LLM 生成の応答に要約できます。エンジンは、マップ/リデュースアプローチを介して大量のテキストを自動的に処理し、モデルトークン制限に到達しないようにします。
- **自動化された日次のクラスター状態レポート**ハイブリッドマネージャーは、エステート内のすべてのクラスターに対して 1 日 1 回、包括的な状態レポートを自動的に生成するようになりました。これらのレポートは、生のメトリックよりも長く保持されるため、長期的な状態の傾向を追跡し、異常を特定し、パフォーマンスのスナップショットを共有できます。チャットエージェントにレポートを取得または比較するように依頼することもできます。
- **チャットボットを介したインタラクティブな AI の洞察** Hybrid Manager チャットボットが運用ワークフローにさらに深く統合されました。実行中のワークロードに基づいて自動スキーマデザインの推奨事項を提供し、単純な自然言語クエリを介して毎日生成されるクラスター状態レポートを取得および比較できるようになりました。
- **移行リスク評価の強化**データ移行のより高い成功率を保証するために、このリリースではソースデー

タベースの詳細な評価を追加します。これには、レプリケーションスロットの状態のモニタリングと、Debezium ベースの CDC に影響を与える可能性のある危険なテーブル構成プライマリキー、TOAST、およびデータ型の特定が含まれます。

- **高度な AI パイプライン機能** AI Factory が、マルチステップのコンポーザブルパイプラインをサポートするようになりました。ユーザーは、構造化 PDF の画像への変換、OCR によるテキストの抽出、LLM による結果の要約など、複雑なタスクをビジュアルパイプラインデザイナー内で直接チェーンできるようになりました。

機能と機能強化

Hybrid ManagerHM プラットフォーム

Type	Description
Feature	*Automations* `の紹介は、ユーザーが日常的なデータベース操作を自動化できる強力な新しいエンジンです。最初のリリースには、計画的な自動タスクのシリーズの最初のものである *`Disk auto-scaling* `が含まれています。新しいフレームワークは、タスクをスケジュールするプロジェクトレベルの *`Maintenance Windows*、詳細な実行制御、オプションのヒューマンインザループ承認、および単一の一元化インターフェイスからすべての自動アクティビティを監視する統合 `Task Manager`を備えています。
Enhancement	新しい `edbctl project create-object-store`コマンドを使用して、CLI を介してプロジェクトごとに専用オブジェクトストレージを構成するオプションを追加しました。新しいプロジェクトを作成するときに、デフォルトの共有オブジェクトストレージを使用するか、専用のオブジェクトストレージを構成するかを選択できるようになりました、Postgres バックアップのプロジェクトレベルの分離が保証されます。既存のプロジェクトは保護され、force フラグを使用して明示的に上書きしない限り影響を受けません。
Enhancement	HM に同梱されている PGAA バージョンを *1.7.0* **未満の** *edbctl* **の手動フォールバック** に更新しました。
Platform support	Kubernetes 1.34 および OpenShift 4.21 のサポートを追加しました。
Platform support	ARM ベースのアーキテクチャでハイブリッドマネージャーコンポーネントを実行するためのサポートを追加し、最新の ARM プラットフォーム ARM ベースの Kubernetes ノードやクラウドインスタンスなどへの展開が可能になります。
Bug fix	Azure ObjectStore 認証メカニズムをマネージド ID のみから DefaultAzureCredential に更新しました。これにより、複数の認証方法のサポートが有効になり、接続が成功するまでさまざまな資格情報が自動的に試行されます。
Bug fix	プライベート VPC で実行されているときにハイブリッドマネージャーがパブリックロードバランサーを作成しようとしたため、クラスタープロビジョニング障害が発生した問題を修正しました。環境が純粋なプライベートネットワークにある場合、システムはプライベートロードバランサーを正しく作成するようになりました。
Bug fix	システムが Postgres によって適用された最終評価値の代わりに未評価値を使用する GUC グランドユニファイド構成検証の問題を修正し、より正確な構成の検証を保証します。

次のページに続く

表 20 – 前のページからの続き

Type	Description
Bug fix	データベースファイルの非互換性のため、メジャーバージョンのアップグレードに続いて、プライマリクラスターへのレプリカクラスターの追加に失敗する問題を修正しました。
Bug fix	レプリケーション構成エンリッチメントロジックのエラーのため、レプリカクラスターのプロビジョニングが失敗した問題を修正しました。レプリカのライフサイクルが正常に完了するように、調整プロセスが更新されました。
Bug fix	小さなチャンクの高速ストリーミングを受信すると、チャットエージェント UI がコンテンツを損失する問題を修正しました。
Change	チャットエージェントが `nvidia/llama-3.3-nemotron-super-49b-v1.5`をサポートするようになりました。

AI ファクトリー

Type	Description
Feature	AIDB パイプラインフレームワークを拡張して、マルチステップのコンポーザブルパイプラインをサポートしました。 1 つのステップの出力が次のステップの入力にフィードするように、プリペアラーファンクション PDF からテキスト、HTML からテキストなどをチェーンできるようになり、ビジュアルパイプラインデザイナーを介して柔軟なパイプライン構成が可能になります。
Feature	`summarize_text_aggregate()`ファンクションを AIDB に追加し、標準の SQL GROUP BY クエリー内の単一の LLM 生成応答に、複数の行のテキストを要約できるようにします。このファンクションは、構成可能なトークン制限、カスタムプロンプトをサポートし、モデルのトークン制限を超える大量のテキストに対してマップリデュースアプローチを自動的に採用します。
Feature	PDF から画像への変換パイプラインステップを AIDB に追加しました。ページを画像に変換し、OCR PaddleOCR を適用して、ダウンストリームの埋め込みと取得に適したテキストを抽出するパイプラインを介して、複雑なテーブルレイアウトを含む PDF ドキュメントを処理できるようになりました。
Change	このバージョンから、GenAI Builder はデフォルトのオーケストレーションエンジンとして LangFlow を使用します。
Enhancement	無効なモデル名エラーメッセージ以前は [内部モデルクラスター] ページでのみ使用できましたが、外部推論サービス構成ページに拡張されました。
Bug fix	名前にダッシュが含まれる AIDB ボリュームは、識別子の二重引用符が原因で削除できない問題を修正しました。

可観測性とモニタリング

Type	Description
Feature	クラスター状態レポートを追加しました。これらは、エステート内のクラスターごとに 1 日 1 回自動的に生成されます。ヘルスレポートを調べて異常を特定したり、データベースパフォーマンスのスナップショットを他の人と共有したりできます。これらは生のメトリックやクエリ統計よりも長く保持されるため、長期間にわたって状態を追跡するために使用することもできます。チャットエージェントは、レポートを取得および比較することもできます。質問してください。
Feature	チャットエージェントを介してスキーマデザインの推奨事項を追加しました。チャットエージェントにスキーマデザインの推奨事項を依頼できるようになりました。スキーマが実行中のワークロードにどの程度適しているかを評価し、パフォーマンスを向上させるための実用的な推奨事項を提供します。
Feature	異なる Kubernetes クラスターにまたがる複数の外部 PGD グループを単一の論理クラスターとして管理するためのサポートを追加し、**Estate** セクションでの統合可視性とトポロジ管理が有効になります。
Enhancement	ディスクスループットチャートには、より正確なモニタリングのために読み取りと書き込みのスループットが個別に表示されるようになりました。
Enhancement	EDB Postgres AI エージェントコンテナイメージと Helm チャートが、エンタープライズリポジトリに加えて、community360 および標準のリポジトリに公開されるようになり、より広範な展開シナリオでのアクセスが簡素化されます。
Bug fix	レコメンドサービス API でデータベースフィルタが誤って必要とされる問題を修正しました。
Bug fix	モニタリングのノードステータスの合計に`initdb`および`join`ポッドが含まれ、誤ったカウントが発生する問題を修正しました。
Bug fix	外部 CNP クラスターのブレッドクラムパスが、予想される`/Projects/<project-name>/Postgres Clusters/<cluster-name>`では なく 、`/Projects/<project-name>/Detail/Details`と誤って表示される問題を修正しました。
Bug fix	多数のプロジェクトでエステートを処理する際に、要求ヘッダーが大きすぎるため、**Estate** エンドポイントへの API 要求が 431 エラーで失敗した問題を修正しました。

マイグレーション

Type	Description
Feature	ソース Postgres データベースの論理レプリケーションスロットの状態を追跡するレプリケーションスロットヘルスモニタを追加し、ダウンロード可能なメトリックレポートで各スロットのヘルスステータス、再生ラグ、およびディスクリテンションを報告します。
Feature	スキーマ変更検出とは無関係に実行される専用のメトリックフェッチャーを追加し、定期的なポーリング間隔での WAL 生成レートなどのデータベースメトリックの継続的なモニタリングを有効にします。

次のページに続く

表 23 – 前のページからの続き

Type	Description
Feature	WAL 生成への各テーブルの寄与を推定するテーブルごとのタプル変更率評価を追加し、ソース Postgres データベースのより詳細な移行計画を可能にします。
Feature	ソース Postgres データベースでの WAL 蓄積速度を監視するグローバル WAL 生成速度評価を追加し、移行スナップショット中のストレージ容量要件の推定に役立ちます。
Feature	Postgres テーブルの主キー、レプリカ IDENTITY、および TOAST 構成の危険な組み合わせをチェックする移行評価を追加し、Debezium CDC がサイレントに変更を見逃す可能性のあるシナリオを特定します。
Feature	Debezium ベースの CDC でサポートされていないまたは危険な Postgres データ型を特定する移行評価を追加し、移行を開始する前に潜在的な互換性問題に関する早期警告を提供します。
Feature	JSON 形式で移行ポータル評価レポートをダウンロードするためのパブリック API エンドポイントを追加し、GUI を使用する必要がなくなりました。
Enhancement	スナップショットフェーズ後まで主キーのアプリケーションを遅延することにより、データ移行サービス DMS スナップショットのパフォーマンスが向上しました。複合主キーがスナップショット後の順序で収集および適用されるようになり、 pg_dump および pg_restore と同等のパフォーマンスを達成します。スナップショットフェーズとストリーミングフェーズ間の重複排除のサポートも追加しました。
Bug fix	移行ポータルプロジェクトの作成の失敗通知を無視しても、失敗したプロジェクトエントリがバックエンドから削除されず、同じスキーマの新しい移行ポータルプロジェクトを作成できない問題を修正しました。
Bug fix	[**データベースの詳細**] ページに AWS RDS PostgreSQL および Aurora PostgreSQL データベースのスキーマがリストされていない問題を修正しました。EDB Postgres AI エージェントは、保護された `rdsadmin` システムデータベースに接続しようとしたため、インスタンス上のすべてのデータベースでスキーマ収集が失敗しました。

2.1.1.2.6 Hybrid Manager 2026.4.1 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2026 年 4 月 23 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager* **デュアルリリース戦略** を参照してください。

- **オートメーションエンジンの拡張** 新しい **Automations** タスクにより、推奨されるインデックスの自動適用、CPU とメモリの自動増加、マイナーおよびパッチデータベースの更新を自動化できます。すべて構成可能なメンテナンスウィンドウと人間による-ループの承認。
- **Langflow の AIDB カバレッジの拡大** 新しい Langflow コンポーネントは、EDB Knowledge Base コンポーネントと新しい EDB Airman MCP Server、EDB Embedded Models、EDB Vector Index コンポー

ネントのハイブリッド検索を含む AIDB 統合を拡張します。構成可能なリソース制約と完全なオブザーバビリティの統合を使用して、Langflow フローを管理対象ワークロードとして HM に展開することもできます。

- **Langflow 移行テンプレート**新しい AI 搭載の Langflow テンプレートは、HM 管理クラスタのプロビジョニング、スキーマの移行、スナップショットとストリーミングデータの移行の実行を含む、Oracle から Postgres へのエンドツーエンドの移行プロセスを自動化します。
- **統合スキーマの移行** Oracle SQL 互換性評価と AI エージェントを介した API 駆動の自動化のサポートにより、ハイブリッドマネージャーを介して Oracle または Postgres ソースから Postgres の宛先に直接データベーススキーマを移行できるようになりました。
- **プロアクティブなクエリとスキーマ変更の検出** HM は、監視対象データベースで初めて実行されるクエリを自動的に検出し、スキーマの変更を特定するようになりました両方とも 1 日以内- 予期しない変更を見つけ、潜在的な回帰を早期に調査するのに役立ちます。
- **``dbaas`` シナリオの分離**データベースクラスター管理機能は、`core` シナリオから分離された専用の `dbaas` シナリオによって制御されるようになりました。2026.3 からアップグレードする既存のインストールは、Postgres クラスター管理機能を保持するために `dbaas` をシナリオリストに明示的に追加する必要があります。
- **OSS ライブラリー** コミュニティデータベースツール Apache Airflow、Apache Superset、Metabase、pgAdmin4、pgBadger、および HammerDB を含むを、構成可能なライブラリー、スケジューリング、および承認ワークフロー。

リリースの詳細

Hybrid ManagerHM プラットフォーム

Type	Description
Feature	導入された OSS ライブラリーは、HM プロジェクトに直接展開できるコミュニティデータベースツールのカタログです。初期カタログには、Apache Airflow、Apache Superset、Metabase、pgAdmin 4、pgBadger、および HammerDB が含まれています。アプリは、 **アセットライブラリー>アプリ** セクションから展開され、構成可能なパラメーター、オプションの展開スケジュール、および人間参加型の承認ワークフローを使用します。展開されたアプリは、 **Estate > Apps** またはプロジェクトの **Apps** ビューから監視、アップグレード、および削除できます。
Feature	**Automations** エンジンを介して推奨されるデータベースインデックスを自動的に適用する機能を追加しました。データベース、スキーマ、またはテーブルごとに提案をフィルターし、明示的な人間の承認を得て、および/または **メンテナンス期間** 中に適用することを選択します。
Feature	使用量のしきい値を超えたときに CPU および/またはメモリリソースを自動的にスケールアップするように **Automations** エンジンで HM 管理クラスターを構成できるようになりました。

次のページに続く

表 24 – 前のページからの続き

Type	Description
Feature	**Automations** エンジンを介してマイナーおよびパッチデータベースの更新を自動化できるようになりました。優先するスケジュールとメンテナンスルールを設定して、手動の労力なしで HM 管理クラスターを最新の状態に保ちます。
Enhancement	Barman Cloud 圧縮アルゴリズムとの統合を介して、HM バックアップ機能に高度な圧縮が追加されました。 WAL 圧縮がデフォルトで有効になるようにして、バックアップパフォーマンスを向上させ、ストレージ消費を削減し、WAL ストレージがフルになるインシデントを防ぎます。ベースバックアップの Snappy 圧縮と WAL ファイルの zstd 圧縮はデフォルトです。
Enhancement	HA、AHA、および DHA クラスターにオプションの読み取り専用接続文字列を追加しました。専用の読み取り専用エンドポイントを有効にして、読み取り操作をスタンバイレプリカ HA または現在書き込みリーダー AHA および DHA として指定されていないノードにルーティングし、プライマリの負荷を削減します。 *Cluster setup* HA で構成するか、 *2 つのデータグループと 1 つの監視グループを含む EDB Postgres 分散 PGD クラスターが正常ではありません* AHA および DHA のデータグループごとに構成します。有効にすると、接続文字列はクラスターの *Troubleshooting* で利用できるようになります。
Enhancement	クラスタープロビジョニングとパッチ API でのノードセレクターと許容範囲のサポートを追加しました。追加のポッドスケジューリング制約を指定して、Kubernetes クラスター内の PostgreSQL ポッドを展開する場所を制御できるようになりました。これには、高度なワークロード分離シナリオで特定のノードをターゲットし、ノードテイントを許容する機能が含まれます。
Enhancement	EDB Postgres AI エージェント `beacon-agent`プロビジョニングにカスタムロードバランサーアノテーションのサポートを追加し、サポートされているプラットフォーム全体でデータベース展開のロードバランサーアノテーションを構成できます。この機能は、ロケーション機能レベルで有効または無効にできます。
Enhancement	この機能が EDB Postgres AI エージェント `beacon-agent`構成を介してロケーションでサポートされている場合、カスタムロードバランサーアノテーション metal-lb または任意のカスタムアノテーションを使用して API を介してクラスターをプロビジョニングする機能を追加しました。これは、このリリースの API のみの機能です。 UI および CLI のサポートは、次の LTS で追加されます。
Feature	IdP グループ属性に基づいて HM ユーザーロールを自動的に割り当てる機能を追加しました。宣言的な方法で LDAP ユーザー属性またはグループ名をロールにマッピングできるようになったため、 LDAP ディレクトリ内のユーザーにロールを手動で割り当てる必要がありません。
Enhancement	推論サービスを作成すると、チャットエージェントが自動的に利用できるようになりました。以前は、ブラウザを更新して Chat Agent を利用できるようにする必要がありました。

次のページに続く

表 24 – 前のページからの続き

Type	Description
Enhancement	拡張されたイベントソースによりデータベース通知システムが強化されました。通知システムは、プラットフォームによって生成されたエージェントの自動タスクを含む、HM システムによって記録されたイベントを通知できるようになりました。さらに、監査ログテーブルのすべてのリソースタイプがクリック可能になり、以前に使用できたタスクのみの機能を超えています。
Enhancement	イメージライブラリの **この画像を使用しているクラスター** ビューを強化して、ユーザーロールとプロジェクト権限に基づいて、アクセス権限があるクラスターのみを表示します。
Enhancement	イメージ検出のレジストリプロバイダーとしての GitLab コンテナレジストリのサポートを追加しました。
Enhancement	チャットエージェントを介して、高可用性、高度な高可用性、および分散高可用性ディストリビューションを含む HM 管理クラスターを作成できるようになりました。
Enhancement	監査ログエントリに API キー識別子を含めるようにアクティビティログを強化し、ゲートウェイでの ID メタデータの追跡を向上させて、セキュリティの監視と相関を向上させました。
Enhancement	チャットエージェントの機能とセキュリティが強化されました。チャットエージェントは、インタラクティブ情報の確実性を高める選択リストインタラクション方法を使用し、パスワードが大規模な言語モデルにアップロードされないことを保証する別のパスワード伝送パスを使用するようになりました。
Enhancement	`upm-flow-proxy`サービスを介して、公開されたフローバンドルを展開するためのサポートを追加しました。 HM プラットフォームは、クラウドストレージ S3、Azure、GCP の代わりに内部フロープロキシサービスから直接フローを取得できるようになり、公開されたフロー構成の合理的な展開が可能になります。
Enhancement	統合システムデータベース *upm-system-db* を導入しました。オペレータープリフライトチェックは、インストールまたはアップグレードの前に、必要なデータベース資格情報シークレット *db-upm*、*db-dex*、*db-langflow*、*db-lakekeeper*、*db-beacon-app*、*db-transporter*、`db-superuser`が存在することを検証するようになりました。これらのシークレットは、IR 2026.4 以降のバージョンをターゲットとする場合、`edbctl setup create-install-secrets`によって自動生成されます。
Enhancement	プロビジョニング、バックアップ、保存場所、クラスターテンプレートを含むデータベースクラスター管理機能は、 `core`シナリオから分離された専用の`dbaas`シナリオによって制御されるようになりました。シナリオが指定されない場合、新規インストールにはデフォルトで`dbaas`が含まれます。 2026.3 からアップグレードする既存のインストールは、オペレーターをアップグレードした後、Postgres クラスター管理機能を保持するために`dbaas`をシナリオリストに明示的に追加する必要があります。 `dbaas`がないと、クラスター管理 API は 404 を返し、 **Clusters** セクションは HM コンソールで非表示になります。

次のページに続く

表 24 – 前のページからの続き

Type	Description
Change	*edb_stat_monitor.edbsm_normalized_query* GUC のデフォルト値を `false`から `true`に変更しました。この変更により、述語で使用される値がデフォルトで HM に保存されないため、機密のクエリデータをデータベースから残したくない顧客のデータプライバシーが向上します。
Deprecation	Hybrid Manager にバンドルされている Apache Spark コンポーネントを削除しました。外部 Spark クラスターへの接続は引き続きサポートされていることに注意してください。
Deprecation	HM CRD の `beaconServer.additionalTrustDomains`フィールドを非推奨にしました。このフィールドは、*upm-beacon* Helm チャートでは使用されなくなり、専用の Helm 値構成に置き換えられました。
Security	AI モデルサーバー API へのアクセスを制限するための Langflow コンポーネントのネットワークセキュリティポリシーを強化しました。 Langflow は、モデルサーバークラスターと推論サービスの読み取り専用の検出 API にのみアクセスできるようになりましたが、作成、更新、削除などの危険な変更操作をブロックします。
Security	SPIRE サーバー監査ログが無効になっているため、SVID の発行、エントリの変更、フェデレーション操作などのセキュリティ関連イベントのログが防止されているセキュリティ問題を修正しました。 SPIRE サーバーでは、SOC2 および PCI DSS のコンプライアンス要件を満たすために監査ログが有効になりました。
Security	公開されたフローのプロジェクトスコープの承認が強化されました。 `upm-flow-proxy`はプロジェクトレベルの権限を適用し、ユーザーが承認されたプロジェクト内で公開されたフローにのみアクセスおよび変更できるようになりました。フローを作成するとき、ユーザーは指定されたすべてのプロジェクトにアクセスする必要があります。既存のフローを更新する場合、ユーザーは自分がアクセスできるプロジェクトのみを追加または削除できますが、スコープ外のプロジェクトは変更されません。
Bug fix	競合状態が原因で ClusterWrapper がソフトな削除フェーズでスタックする可能性がある問題を修正しました。
Bug fix	コンポーネントのデフォルトが `upm-beaco-ff-base`コンポーネントに適用されない問題を修正しました。これにより、 *usage_generator_interval*、*AES_KEY_ROTATION_INTERVAL*、および `SESSION_DURATION_SECONDS`パラメーターの構成値が欠落しているため、`accm-server`がクラッシュループを発生する可能性がありました。
Bug fix	PostgreSQL ステータスストアが存在しない場合、保護されていない null 参照エラーが原因で、Google Kubernetes EngineGKE、Amazon Elastic Kubernetes ServiceEKS、および Red Hat OpenShiftRHOS プラットフォームで読み取り専用接続テストが失敗する問題を修正しました。
Bug fix	`beaconServer.logLevel`がデフォルトで `info`になり、ユーザー構成値をオーバーライドする問題を修正しました。設定はユーザー構成値を正しく尊重するようになりました。

次のページに続く

表 24 – 前のページからの続き

Type	Description
Bug fix	SPIRE エージェントがクラスター DNS で解決できなかったノードホスト名を介して kubelet に接続しようとしたときの、DNS 解決障害が原因で RHOS 環境でポッドがクラッシュループしていた問題を修正しました。
Bug fix	EDB Postgres AI エージェント *beacon-agent* が、古い HM サーバーに対して新しいエージェントバージョンでサーバーを再登録するときに、紛らわしいエラーメッセージを生成した問題を修正しました。エージェントはバージョンの互換性をより適切に処理し、サーバー API が利用できない場合に、より明確なエラーレポートを提供するようになりました。
Bug fix	AI Pipelines がボリュームソースの削除されたソースレコードを正しく検出しなかったバグを修正しました。このバグは、孤立した宛先レコードが残される原因となりますが、現在修正されています。
Bug fix	`upm-beacon-server`は、spire コンポーネントが壊れているか、使用できない場合にクラッシュループに入ります。 spire は、セカンダリロケーションをサポートするように構成されたすべての HM 展開に必要であるため、壊れた spire コンポーネントは Postgres クラスターの展開を妨げます。
Bug fix	`edbctl setup create-install-secrets`コマンドの資格情報作成プロセスに非決定的な順序があり、名前空間とシークレットが実行ごとに異なる順序で作成される問題を修正しました。

AI ファクトリー

Type	Description
Feature	EDB ナレッジベースコンポーネントに追加されたハイブリッド検索サポートにより、Langflow の AIDB カバレッジを拡張し、テキストマッチングとベクトル類似性検索を組み合わせました。 3 つの新しいコンポーネントが追加されました。EDB Airman MCP Server。 EDB Embedded Models、 AIDB の埋め込みモデルを使用した CPU ベースのテキストと画像の推論用。 EDB Vector Index、EDB ナレッジベースにベクトルインデックスを作成します。
Feature	AI を活用したエンドツーエンドのデータベース移行のための Langflow テンプレートを追加しました。このテンプレートは、HM 管理クラスターのプロビジョニング、スキーマの移行、スナップショットとストリーミングデータの移行の実行を含む、Oracle から Postgres への完全な移行プロセスを自動化します。必要に応じて、移行フローをカスタマイズして、独自のソリューションを統合できます。この初期バージョンは、完全な互換性のソーススキーマを使用した単一データベースの移行をサポートし、 GPT-4o 以上の推論クラスモデルを必要とします。

次のページに続く

表 25 – 前のページからの続き

Type	Description
Feature	Postgres の AIDB 拡張機能の AI Pipelines コンポーネントは、単一の共有ナレッジベースへの複数の互換性のあるパイプラインの書き込みをサポートするようになり、それぞれが独立した処理ステップを持つマルチプルのソースとフォーマットマルチモデル画像とテキストを含むからのハイブリッドナレッジベースを有効にします。ベクターテーブルには、エンベディングごとにソースパイプラインを追跡するための新しい`pipeline_id`列が含まれ、取得ファンクションは元のパイプライン名を結果ます。
Feature	ハイブリッドマネージャーによって管理される Kubernetes 環境への Langflow フローの展開のサポートを追加しました。メモリ、CPU 割り当て、自動スケーリング、再起動ポリシーなどの構成可能なサービスの品質の制約を使用して Langflow フローを展開できるようになりました。展開されたフローには、完全な可観測性とハイブリッドマネージャーとのアラート統合が含まれ、展開、展開解除、およびステータスモニタリング機能を備えた HM インターフェイスを介して管理できます。
Feature	AIDB 拡張機能の AI Pipelines コンポーネントにハイブリッド検索機能を追加し、全文検索、ベクトル/セマンティック検索、述語フィルタリングを組み合わせました。以前は、ベクター/セマンティック検索のみがサポートされていました。カスタム SQL クエリを使用して、AI Pipelines ナレッジベースから高度な取得クエリを実行できるようになりました。これらのクエリには、任意のリレーショナルフィルタリングまたはサブクエリを含める場合があり、bm25 インデックスを使用して全文検索を実行することもできます。 AI Pipelines には、これらのクエリの構築に役立つ便利な機能が含まれており、ドキュメントには一般的なユースケースの例が含まれています。
Enhancement	エラーログとデッドレターキューを追加して、AI パイプラインエラー処理を改善しました。どのファイルまたはレコードが正常に処理され、パイプラインの実行中にエラーが発生したファイルを追跡できるようになりました。
Enhancement	ノンブロッキングエラー処理を AI パイプラインに追加しました。パイプラインは、エラーが個々のレコード不正な PDF ファイルや無効なデータにのみ影響を与える場合に処理を継続しますが、モデルの欠落や構成の問題などのパイプラインレベルのエラーでは停止します。新しい`aidb.get_error_logs()`および`aidb.clear_error_logs()`ファンクションを使用して、パイプラインエラーを照会および管理できます。
Enhancement	オンプレミスデータベースと外部 CloudNativePG クラスターの AI データパイプラインを作成するサポートを追加しました。非管理対象エステートのコントロールプレーンパイプラインを作成できるようになり、HM 管理データベースインスタンスを超えてパイプライン機能を拡張できます。
Enhancement	Hybrid ManagerHM プラットフォームによって管理される外部 Postgres クラスターの AIDB パイプラインサポートを追加しました。 HM のセカンダリロケーションで実行されているクラスター、および HM プラットフォームの外部で実行されているクラスターを含む、これらの外部クラスターで AIDB パイプラインを構築し、プライマリ場所からモデルおよび推論サービスにアクセスできるようになりました。

次のページに続く

表 25 – 前のページからの続き

Type	Description
Enhancement	AI パイプラインデザイナーは、静的なデフォルト設定を使用する代わりに、使用可能なモデルを動的に取得するようになりました。この変更により、同じユーザーエクスペリエンスを維持しながら、特定のプロジェクト、リソース、およびデータベースの組み合わせに基づいてモデルを選択できます。
Enhancement	ビジュアル AI パイプラインデザイナーで直接 PDF to Image パイプラインステップを構成するサポートを追加しました。デザイナーインターフェイスを介してフォーマット、DPI、ページ範囲、注釈レンダリングなどの変換オプションを設定できるようになりました。
Enhancement	ハイブリッドマネージャーでの EDB AIDB パイプライン拡張機能 v6.1.0 のサポートを追加し、最新のパイプライン機能と機能強化を AIDB ワークフローにもたらします。
Enhancement	LangFlow を使用して AI ワークフローを構築し、フローをアセットとして Hybrid Manager プラットフォームで公開できます。これらのフローは、プロジェクト内の選択した場所でスタンドアロンジョブとして実行できます。
Enhancement	新しいモデルプロバイダーとして Hugging Face のサポートを追加しました。プラットフォームに新しいモデルを追加するときに、ハギングフェイスモデル名、オブジェクトストレージパス、またはその両方を使用して AI モデルを構成できるようになりました。
Enhancement	詳細なエラーメッセージを表示するように AI モデル作成フォームのエラー処理を強化し、モデル作成要件が満たされない場合のユーザーフィードバックを向上させました。
Security	Langflow コンポーネントのリモートコード実行の脆弱性を修正したため、攻撃者は`build_public_tmp`エンドポイントの data パラメーターを介して任意のコマンドを実行できます。
Bug fix	ネットワークポリシーの制限により、Postgres ポッドがモデルサーバー API にアクセスできないため、RKE、GKE、および AKS プラットフォームのビジュアルパイプラインビルダーから AI モデルサービスをリストできない問題を修正しました。
Bug fix	AIDB Pipeline Designer のプロビジョニングページに、特定のプロジェクトからのみを表示するようにクラスターをフィルタリングするのではなく、すべてのプロジェクトのクラスターが表示される問題を修正しました。

可観測性とモニタリング

Type	Description
Feature	これまでに監視対象データベースで実行されたことのないクエリの検出を追加しました。 HM は、1 日以内に新しいクエリを識別するようになり、問題を発生させていないことを確認し、問題が発生している場合は根本原因を簡単に見つけられるようになりました。この機能には、すべてのクエリが最初に新しいものとして表示されるデータベースを最初に追加するときなど、検出を無効にし、過剰な通知を抑制するメカニズムが含まれています。

次のページに続く

表 26 – 前のページからの続き

Type	Description
Feature	1 日以内の監視対象データベーススキーマの変更を自動的に識別するスキーマ変更検出を追加しました。予期しないスキーマ変更を特定し、変更の前後でデータベースの状態を比較して、潜在的な回帰を特定できるようになりました。
Feature	HM コンソールに新しい比較ツールを追加して、2 つのユーザー定義の期間でデータベースメトリックとクエリ統計を比較できます。サイドバイサイドまたはオーバーレイされたチャートビューでパフォーマンスの違いを簡単に特定し、フィルターやズームコントロールを適用し、強調表示された違いを確認して時間の経過による変更の影響を理解できるようになりました。
Enhancement	モニタリングダッシュボードのアノテーション機能を拡張して、ユーザーが挿入したカスタムイベントをサポートしました。システム生成イベントとともに、興味深いプラットフォームとデータベースのアクティビティを手動でマークできるようになったため、変更をパフォーマンス傾向と関連付けることが簡単になります。
Enhancement	CNPG-GC クラスターを含む外部 PGD クラスターの可観測性サポートを拡張しました。以前のリリースで導入されたモニタリング機能を基に、HM 可観測機能を介して自己管理 PGD および CNPG-GC クラスターの両方からメトリックを監視できるようになりました。
Enhancement	ビーコンサーバーの新しい `EnableEDBUsageReporting`構成オプションを追加して、使用状況レポートをより明示的に制御します。ビーコンサーバーには、IT リンクが到達不能な場合に 24 時間ごとに再接続を試みる再試行ロジックが含まれ、エアギャップセットアップを含むさまざまな環境での操作の信頼性が向上します。
Enhancement	クラスター仕様が更新されると、ユーザーの変更、自動スケーリング、またはエージェントによるオートメーションによって、アノテーションが自動的に生成され、オブザーバビリティビューに表示されます。
Bug fix	ポートが有効なポート範囲外の 0 に指定されている場合、無効な Postgres DSN エラーのため EDB Postgres AI エージェント *beacon-agent* が起動に失敗した問題を修正しました。
Bug fix	メトリック時間と値が、HM コンソールのクラスターレベルとノードレベルのビュー間の不一致を示した問題を修正しました。バックエンドは、両方のビューで一貫したデータを返し、ノードレベルのメトリックのタイムスタンプを正しく配置するようになりました。
Bug fix	内部メトリック統計データベースの堅牢性とパフォーマンスを強化し、ストレージとメモリの使用量を最適化します。
Bug fix	WAL ストレージパーセントメトリックが、監視グラフで 100%を超える誤った値を表示する問題を修正しました。このメトリックは、プライマリノードのみによる WAL 使用量を正確に集計し、適切なパーセンテージ計算を提供するようになりました。

HM コンソール

Type	Description
Feature	AI を活用したクエリ生成をクエリエディターに追加しました。 **Generate with AI** を選択して、自然言語プロンプトを使用して SQL クエリーを作成できるようになりました。この機能は、クエリを許可されているエステート全体で有効な SQL ターゲティングデータを生成し、複雑な SQL 構文を書かずにデータの探索および分析が簡単になります。
Enhancement	PDF をダウンロードせずに UI でデータベースの状態レポートを直接表示する機能を追加しました。ダウンロードを決定する前に、レポートの内容をインラインでプレビューできるようになりました。
Enhancement	**Settings** セクションの HM 認証ポリシー用の HM コンソールに構成インターフェイスを追加しました。組織の所有者は、Web ログインセッションのアイドルタイムアウトを含む認証ポリシー設定を構成できるようになりました。
Enhancement	**Invalidate User Sessions** アクションを ID プロバイダー管理インターフェイスに追加しました。 HM コンソールを介して特定の ID プロバイダーのすべてのアクティブなユーザーセッションを無効にできるようになり、管理者がユーザーアクセス管理をより適切に制御できるようになりました。
Enhancement	ロールの割り当てインターフェイスにフィルタを追加して、現在のユーザーの権限を表示するときに手動で割り当てられたロールのみを表示します。ノイズを削減するために、IdP グループマッピングを介して割り当てられたロールがこのビューから除外されるようになりました。
Bug fix	HM コンソールでマシンユーザーを正しくフィルタリングできない問題を修正しました。フィルタリングでは、一般的なクエリパラメーターの代わりに適切な ID プロバイダーパラメーターを使用するようになりました。

アナリティクス

Type	Description
Enhancement	HM に同梱されている PGAA バージョンを 1.8 に更新しました。
Enhancement	Lakekeeper ウェアハウスの作成を強化して、AWS Security Token ServiceSTS をサポートしていないオブジェクトストレージ環境をサポートします。 S3 互換ストレージで STS を有効にした最初の試行が失敗した場合、システムは STS を無効にしてウェアハウスの作成を自動的に再試行し、Vultr の Ceph Object Gateway のようなプロバイダーとの互換性が向上しました。

マイグレーション

Type	Description
Feature	各データベースとスキーマを使用するアプリケーションに関するメタデータを入力および表示する機能を追加しました。アプリケーションを多対多の関係でデータベースおよびスキーマに関連付けることができ、データベースの移行の計画と優先順位付けにより良いコンテキストを提供します。アプリケーションのメタデータは、ハイブリッドマネージャーインターフェイスを介して手動で、またはハイブリッドマネージャー API を介してプログラムで入力できます。
Feature	EDB Postgres AI エージェント *beacon-agent* が、Oracle `TNSNAMES.ORA` ファイルや CSV ファイルなどの確立された接続情報のソースを使用して、Oracle および Postgres データベース接続を自動的に構成する機能を追加しました。RDS Postgres データベースを自動検出し、それらへのエージェント接続を構成するオプションも利用できます。
Feature	ハイブリッドマネージャーを介して Oracle または Postgres ソースから Postgres の宛先にデータベーススキーマを直接移行する統合スキーマ移行機能を追加し、外部ツールの必要性を排除します。このサービスは、HM 管理と外部管理の Postgres の両方の宛先をサポートし、AI エージェント自動化のための API アクセスを使用して、スキーマのみ、データのみ、または組み合わせた移行のデータ移行ワークフローと統合します。 Oracle から EDB Postgres Advanced Server に移行する場合、互換性のあるスキーマは移行ポータルから取得されます。
Feature	EDB Postgres Advanced Server との互換性について Oracle アプリケーションの SQL ステートメントを評価する機能を追加しました。 Oracle のトレースファイルとシステムビューから抽出された SQL ステートメント、およびユーザーインタフェースまたはプログラム API を介して直接提供される SQL ステートメントを評価できるようになりました。この機能には、MCP ツールのサポートが含まれ、ハイブリッドマネージャー移行アーキテクチャ内の新しいアプリケーション情報オブジェクトを導入して、 SQL ステートメントおよび評価レポートのコンテナとして機能します。
Enhancement	スキーマ解析、ストレージ、および移行機能を移行ポータルに追加しました。Postgres データベースが移行のためにハイブリッドマネージャーに登録されると、Postgres データベーススキーマが自動的に取り込まれ、解析され、移行ポータルのバックエンドデータベースに保存されます。これらのスキーマは、新しい API 動作で移行できます。
Enhancement	大量ダウンロードをサポートするように移行ポータル評価レポート API を拡張しました。新しいエンドポイントは、複数のデータベースリソース ID を受け入れ、指定されたすべてのリソースの評価レポートを単一の JSONL ファイルで結果、大規模なレポートのダウンロードが簡単になります。
Enhancement	移行エージェントは、Oracle スキーマを抽出するときに上流のキャンセルシグナルを尊重するようになりました。以前は、EDB Postgres AI エージェント `beacon-agent` が Oracle スキーマ抽出中に終了信号を受信すると、`sqlplus` プロセスが完了まで実行され続け、配信できなかった作業にリソースを浪費していました。エージェントは、キャンセル信号を受信したときに `sqlplus` サブプロセスを適切に終了するようになり、よりクリーンなシャットダウン動作が有効になります。

次のページに続く

表 29 – 前のページからの続き

Type	Description
Enhancement	*DATE*、*TIME*、*TIMESTAMP*、*INTERVAL*、*UUID*、*JSON*、*XML*、および`POINT`タイプに基づいて PostgreSQL `DOMAIN`タイプをレプリケートするサポートを追加しました。 `DOMAIN`タイプは、既存の PostgreSQL タイプを追加の制約でラップするユーザー定義タイプです。ターゲットが同じ`DOMAIN`を使用するか、基になるベースタイプのみを使用するかにかかわらず、スナップショットと CDC レプリケーションモードの両方が、これらの`DOMAIN`タイプを正しく処理するようになりました。
Bug fix	EDB Postgres AI エージェント`beacon-agent`が非特権ユーザーとして接続したときに、Oracle スキーマ収集が誤ったテーブル数を報告する問題を修正しました。
Bug fix	評価中に PostgreSQL データベースの DDL 抽出が失敗したときに、`GetMigrationDatabaseAssessmentData`ファンクションでヌル文字列例外が発生し、API 要求が 500 エラーを返す問題を修正しました。
Bug fix	特定の環境での Oracle 評価中にデータベース機能が収集されなかった問題を修正しました。
Bug fix	スキーマに依存関係がある場合に gRPC UI でエラーが発生する問題を修正しました。
Bug fix	**移行データベース** が空の状態にあるネストされたボタン要素が無効な DOM 構造の警告を発生させていた問題を修正しました。
Bug fix	ソーステーブルのロードが失敗したときに Create OLTP Migration フォームにエラーメッセージが表示されず、ユーザーに何が悪かったのかに関するフィードバックがない問題を修正しました。
Bug fix	PostgreSQL テーブルまたは列名に日本語、中国語、韓国語などのマルチバイト UTF-8 文字が含まれている場合、CDC レプリケーションが失敗する問題を修正しました。
Bug fix	リカバリーなしでワイドテーブルの移行が失敗する原因となった問題を修正しました。
Bug fix	Netty 依存関係をバンプした後、S3 にオブジェクトをアップロードするときにリーダーがエラーをスローする問題を修正しました。このエラーは、非同期 S3 クライアントの接続プールタイムアウトの問題が発生したために発生しました。これは、同期 S3 クライアントに切り替えることにより解決されました。

2.1.1.2.7 Hybrid Manager 2026.5.0 release notes

このイノベーションリリースには、機能と機能強化が含まれています。

発売日 2026 年 5 月 19 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager (HM) dual release strategy* を参照してください。

- **Postgres から Postgres へのスキーマの移行の拡張 :** データ移行サービス DMS は、スキーマの移行中に Postgres 固有のデータベースオブジェクトを処理するようになりました。カスタムドメインタイ

プやその他の Postgres ネイティブタイプ定義を含む、完全なエンドツー Postgres データベース間のスキーマの移行を終了します。

- **エージェントの移行の強化**移行エージェントの Langflow テンプレートは、HM のユニファイドスキーマ移行機能を活用する改善されたアーキテクチャを使用するようになり、Llama 3 Nemotron 49B および Nano 4B モデルのサポートが追加されました。
- **HM チャットボットの機能強化**チャットボットプロバイダーとして Anthropic Claude を追加し、引用裏付けの回答用の HM およびコミュニティ PostgreSQL ドキュメントをバンドルしました。
- **ロールマッピングの改善**単一のロールマッピングルールで複数のロールを割り当てできるようになり、ID プロバイダーグループからロールへの割り当てを合理化します。

リリースの詳細

AI ファクトリー

Type	Description
Enhancement	AI パイプラインデザイナーで AIDB 7 のサポートを追加しました。ナレッジベースの検証の向上と、既存のナレッジベーステーブルと EDB Postgres 分散 PGD クラスターとパイプラインに影響を与える問題の修正を含みます。詳細については、 AIDB 7 release notes を参照してください。
Enhancement	安全でない接続を使用した外部推論サービスを登録するためのサポートを追加しました。テスト環境などの署名された証明書を持たない外部モデルサービスを登録するときに、 **Allow insecure Connection** オプションを有効にできるようになりました。システムでは、登録を許可する前に、ユーザーのセキュリティリスクの明示的な承認が必要です。
Enhancement	無効なパラメーターを使用して AI フローをパブリッシュする際のエラー処理が強化されました。 UI には、汎用の **フローの公開に失敗しました** メッセージの代わりにサーバーからの詳細なエラーメッセージが表示されるようになり、パラメーター名前付けの問題を特定して修正するのに役立ちます。
Change	AI Pipelines **デフォルト設定** ページを **ソブリン AI** セクションから削除しました。デフォルト設定機能は、新しいインスタンス固有のモデルリストのアプローチと互換性がありません。
Bug fix	uvx 依存関係解決エラーによりモデルコンテキストプロトコル MCP サーバーの初期化が防止された EDBAirman コンポーネントの起動障害を解決しました。ランタイムには、必要な依存関係がすべて含まれます。

HM コンソール

Type	Description
Enhancement	`beacon.enterprisedb.com/exclude-storage-from-dbaas=true`ラベルを追加して、特定の `StorageClass`または `VolumeSnapshotClass`リソースを HM コンソールから除外します。共有 Kubernetes 環境では、このラベルを Postgres ワークロードに適していないストレージクラス GPU または HM 専用クラスなどに適用して、 **Create Cluster** ドロップダウンに表示されないようにできます。ラベルは次の HM ポーリングサイクルで有効になり、再起動は必要ありません。
Enhancement	ロールマッピングルールインターフェイスを強化して、単一のルールで複数のロールの選択をサポートします。ロールごとに個別のルールを作成するのではなく、複数のロール **プロジェクトビューアー** と **カタログデータライター** の両方を、単一のルールで同じグループと ID プロバイダーの組み合わせに割り当てることができるようになりました。
Enhancement	HM チャットボットのサポートされているプロバイダーとして Anthropic Claude を追加しました。チャットボットは、応答にマーメイド図をインラインでレンダリングするようになりました。 HM およびコミュニティ PostgreSQL 16/17/18 ドキュメントをチャットボットにバンドルして、引用付きで製品の質問に回答できるようにしました。

可観測性とモニタリング

Type	Description
Bug fix	Thanos Ruler は、デフォルトの Kubernetes サービスアカウントを使用しました。専用のサービスアカウントを使用するようになりました。

アナリティクス

Type	Description
Enhancement	HM に同梱されている PGAA バージョンを 1.9 に更新しました。

マイグレーション

Type	Description
Enhancement	`schema_and_data`移行のコンカレントスキーマファイナライズを追加しました。スナップショットステージが完了したら、移行の詳細ページで **Finalize Schema Concurrently** を選択して、ノンブロッキングデータ定義言語 DDL バリアントを使用して、バックグラウンドでインデックス、外部キー、およびその他のスキーマオブジェクトを適用できます。後で移行が完了すると、軽量の検証手順のみが残り、カットオーバー時間を大幅に削減します。

次のページに続く

表 34 – 前のページからの続き

Type	Description
Enhancement	DMS での Postgres から Postgres へのスキーマ移行を拡張して、スキーマの移行中に Postgres 固有のデータベースオブジェクトカスタムドメインタイプおよびその他の Postgres ネイティブタイプ定義を含むを処理します。 DMS は、これらのスキーマオブジェクトを依存する他のスキーマオブジェクトよりも前にターゲットに適用するようにし、Postgres データベース間の完全なエンドツーエンドのスキーマの移行を有効にします。
Enhancement	ユニファイドスキーマ移行機能を活用する改善されたアーキテクチャで Simple Migration Agent Langflow テンプレートを強化し、以前の移行ポータルアプローチを置き換えました。更新されたワークフローには、合理化されたエージェントコンポーネント、自動パスワード生成、およびデータベースのセットアップが含まれ、冗長なコンポーネントが削除されます。
Enhancement	Simple Migration Agent Langflow テンプレートに Llama 3 Nemotron 49B および Llama 3 Nemotron Nano 4B モデルのサポートを追加し、移行ワークフローのための追加の AI モデルオプションを提供します。
Enhancement	クラスター作成操作を大規模言語モデル LLM 処理の外部に移動し、グローバルシークレット管理を活用して資格情報の公開を防ぐことにより、Simple Migration Agent Langflow テンプレートのデータベースパスワードのセキュリティが強化されました。
Enhancement	移行スコープをスキーマのみに設定する場合、不要なテーブルのロードと検証をスキップするように移行作成フォームを最適化しました。フォームは、データを含む移行スコープのテーブルデータのみを取得するようになり、パフォーマンスとユーザーエクスペリエンスが向上しました。
Bug fix	MCP サーバーを選択した後、ページを更新するまで MCP ツールコンポーネントのツールのリストが更新されない Simple Migration Agent Langflow テンプレートの問題を修正しました。
Bug fix	EDB Postgres Advanced Server `PACKAGE`および `PACKAGE BODY`スキーマオブジェクトを EPAS ソースデータベースから EPAS 宛先データベースに移行するためのサポートを追加しました。 DMS は、 DMS が参照するテーブルの後にパッケージ本体を適用することを保証するための正しい依存関係の順序を含む、 スキーマの移行中にパッケージおよびパッケージ本体のステートメントを適切に処理するようになりました。
Bug fix	EPAS ソースプロジェクトから EPAS 非パブリック `SYNONYM`スキーマオブジェクトを移行するためのサポートを追加しました。 DMS は、スキーマの移行中に非パブリックシノニムステートメントを適切に処理するようになりました。 DMS は、`PUBLIC SYNONYM`オブジェクトを抽出して移行ポータルプロジェクトにアップロードしますが、宛先データベースへの移行はまだサポートしていません。
Bug fix	スキーマ移行機能が、テーブルを含まないスキーマのデータベースオブジェクトを移行できない問題を修正しました。移行プロセスは、テーブルが含まれるかどうかに関係なく、すべてのスキーマを処理するようになりました。

次のページに続く

表 34 – 前のページからの続き

Type	Description
Bug fix	特定のスキーマのデータ移行を無効にし、他のスキーマでは有効にしている場合、スキーマのみの移行が`PRIMARY KEY`制約とその依存外部キー、名前付けの`CHECK`制約をドロップする問題を修正しました。トランスポーターは、データ移行から除外されたテーブルの **ファイナライズスキーマ** フェーズ中に`PRIMARY KEY`制約を正しく適用するようになりました。
Bug fix	移行にこれらのスキーマが含まれない場合でも、DMS 検証がソースデータベース内のすべてのスキーマが正常な移行ポータルプロジェクトを持つことを誤って必要とする問題を修正しました。検証では、移行のテーブルリストが参照するスキーマのみをチェックするようになりました。
Bug fix	ソース SQL ファイルに Windows スタイルの CR LF 行末が含まれている場合、変換された SQL ファイルに行末が欠落している問題を修正しました。

2.1.1.2.8 Hybrid Manager 2026.5.1 release notes

このイノベーションリリースには、機能強化とバグ修正が含まれています。

発売日 2026 年 6 月 2 日

!!!info リリースペースの更新

HM は現在、**LTS** と **イノベーションリリース**のデュアルリリース戦略に従っています。この新しいモデルの詳細については、*Hybrid Manager (HM) dual release strategy* を参照してください。

Hybrid ManagerHM プラットフォーム

Type	Description
Enhancement	トランスポーターの Kafka サービスを含む Kafka ロードバランサーでのカスタム Kubernetes Service アノテーションのサポートを追加しました。カスタム専用ロードバランサーモードを使用する場合、 MetalLB アドレスプール選択に`metallb.universe.tf/address-pool`または GCP 共有 VPC 内部ロードバランサーサブネット構成に`networking.gke.io/internal-load-balancer-subnet`などのアノテーション、およびノードプールクラスを指定できるようになりました、BYO ネットワークとオンプレミスを有効にします展開シナリオ。
Platform support	Red Hat OpenShiftRHOS オペレーターを使用した展開用に認定された HM 2026.5.1。
Bug fix	Azure Blob Storage への WAL アーカイブが AKS、EKS、および GKE クラスターで HTTP 403 エラーを返した問題を修正しました。システムは、Azure Storage のバックアップ資格情報を正しく処理するようになりました。

AI ファクトリー

Type	Description
Enhancement	AI Model Manager ロールを持つユーザーの Langflow アクセスを制限しました。このロールのないユーザーは Langflow にアクセスできず、AI ワークフローオーサリングのより厳密な制御を提供します。
Enhancement	AI ワークフローの一時的な LLM エラーの自動再試行を追加し、推論プロバイダーが一時的な障害を返すときの信頼性が向上しました。
Enhancement	HM コンソールでタグ名で AI モデルとサービスを検索できるようになりました。
Change	Langflow フローキャッシュを一時ストレージに移動しました。キャッシュはポッドを再起動しても持続せず、起動時に自動的に再構築されます。
Bug fix	ソフト的に削除された AI モデルサービスが HM コンソールに再表示される問題を修正しました。

HM コンソール

Type	Description
Enhancement	HM チャットボットエージェントのサポートを、Gemini 2.x モデルを含むすべての外部推論プロバイダーに拡張しました。
Enhancement	HM コンソールで、クラスターまたはグループ ID でクラスターをフィルターできるようになりました。
Enhancement	HM コンソールに検索可能なモデルサービスピッカーを追加しました。
Bug fix	クラスター内レジストリを使用したインストールでマーケットプレイスパッケージの同期が失敗する問題を修正しました。クラスター内レジストリを使用する場合、パッケージが正しく同期するようになりました。
Bug fix	サポートされていないまたは空のストレージクラスでのクラスター作成が無期限にハングする問題を修正しました。操作はすぐに失敗し、情報エラーが表示されます。

可観測性とモニタリング

Type	Description
Change	デフォルトのアラート評価間隔を 2 分に更新しました。
Bug fix	オブザーバビリティダッシュボードの上位データベースサイズメトリックの誤った計算を修正しました。

マイグレーション

Type	Description
Enhancement	デフォルト拒否セキュリティモデルで移行ポータル API を強化しました。未登録のエンドポイントが HTTP 403 を返すようになり、API はテナントの所有権チェックを実行して、不正なテナント間アクセスを防止するようになりました。
Bug fix	特殊文字または大/小文字混合の名前を含むスキーマが、移行ポータルを介して正しく移行できない問題を修正しました。
Bug fix	Oracle から EDB Postgres Advanced Server スキーマの移行が正しく完了できなかった問題を修正しました。
Bug fix	ファントムまたは重複したスキーマが移行ポータルプロジェクトに表示される問題を修正しました。
Bug fix	最初のスキーマが空だった移行ポータルプロジェクトが完了できない問題を修正しました。

第3章　EDB Postgres AI Hybrid Manager

EDB Postgres AI Hybrid ManagerHM は、Kubernetes ベースのインフラストラクチャ全体での Postgres、ソブリン AI、および分析ワークロードのコントロールプレーンです。オンプレミス、クラウド、またはハイブリッド環境に展開するかにかかわらず、PostgreSQL クラスターをプロビジョニング、操作、および監視する一貫した統一方法を提供します。

ハイブリッドマネージャーは、データベースの展開から AI 分析に至るエンドツーエンドのデータワークフローを有効にし、すべてインフラストラクチャ内で管理されます。

3.1 なぜハイブリッドマネージャーを使うのか？

最新のデータプラットフォームでは、データベースのプロビジョニングだけではなく、以下をサポートする必要があります。

- Kubernetes ネイティブのスケーラビリティとオーケストレーション
- 復元力のある高可用性の展開
- AI と GenAI の統合
- オープンフォーマットを使用した Postgres ネイティブ分析
- ガバナンス、可観測性、およびアクセス制御

Hybrid Manager は、これらの要求を満たすために作成されました。ビルトインの Postgres の専門知識で Kubernetes 機能を拡張し、コンテナ化されたインフラストラクチャの柔軟性を犠牲にすることなくステートフルワークロードの操作を簡単にします。

3.2 主要な機能

Area	Description
Database provisioning	コンソール、テンプレート、または宣言的マニフェストを使用して、高可用性の PostgreSQL または分散 Postgres クラスターを展開します。
AI integration	*Troubleshooting* を実行し、KServe でモデルを提供し、インフラストラクチャによって管理される AI ナレッジベースを構築します。
Analytics	コールドデータを Delta Lake or Iceberg にオフロードし、Postgres 構文で照会し、ハイブリッドトランザクション/分析処理を有効にします。
Monitoring and observability	Prometheus、Grafana、Loki を活用する統合ダッシュボードを使用して、クエリの状態、リソースの使用状況、システムアラートに関する詳細な可視性を取得します。

次のページに続く

表 1 – 前のページからの続き

Area	Description
Security and access management	ID プロバイダーと統合し、プロジェクトとクラスターの RBAC を定義し、データベースとサービスへの安全なアクセスを管理します。
Database migration	外部データベース Oracle、自己管理の Postgres から HM 管理クラスターにスキーマとデータを移行します。移行とスキーマ評価、データ転送スナップショット、および継続的レプリケーションストリーミングのための統合サービスを活用して、最小限のダウンタイムの移行を実現します。

3.3 サポートされている環境

ハイブリッドマネージャーは Kubernetes で実行され、次の一般的な管理対象 Kubernetes クラウドサービスをサポートします。

- Microsoft Azure Kubernetes ServiceAKS
- Amazon Elastic Kubernetes Service EKS
- Red Hat OpenShift Service on AWS ROSA
- Google Kubernetes EngineGKE

HM は、オンプレミスの Kubernetes プラットフォームもサポートしています。

- Red Hat OpenShift RHOS
- SUSE Rancher RKE2

クラウドネイティブとオンプレミスの両方のアーキテクチャをサポートし、主権、コンプライアンス、柔軟なデータインフラストラクチャを構築できます。

3.4 統合ソリューション

Hybrid Manager は、EDB Postgres AI エコシステム全体のコア機能のロックを解除します。それは次のもので構成されています。

- *Getting Started with AI Factory* – プライベートデータを使用して GenAI ワークロードをビルドおよび実行します。
- *Analytics in Hybrid Manager* – Iceberg、Delta Lake、および階層化オフロードを使用してオープンフォーマット分析を有効にします。
- *Simple Migration Agent* – プラットフォーム内の移行ツールを使用して Postgres の採用を加速します。
- **モニタリングの実行** – Kubernetes とデータベースの洞察を備えたビルトインの可観測性スタックを使用して、オペレーションを監視します。
- **タスクおよびクラスターのライフサイクルイベントの通知が見つからない** – テンプレートの作成、アップグレードの実行、HA 設定の構成、およびデータの復元。

3.5 詳細はこちら

- *Architecture overview*
- *Supported Postgres distributions*
- *Supported database types*

3.6 はじめる

- *Install Hybrid Manager*
- Use Hybrid Manager

3.6.1 Hybrid Manager architecture

Hybrid Manager は、クラウドサービスプロバイダーのインフラストラクチャで実行されます。この文脈での「オンプレミス」には、組織のインフラストラクチャ内でホストされる物理サーバーベアメタル、仮想マシン、またはプライベートクラウドでの展開が含まれます。

Kubernetes クラスター内の 3 つの主な論理グループは次のとおりです。

1. **Kubernetes コントロールプレーン** Kubernetes コントロールプレーンは、コア Kubernetes コンポーネントで構成され、2 つまたは 3 つの Kubernetes 制御ノードによって実装されます。コアコンポーネントの例は、`kube-apiserver`、`etcd`、`kube-scheduler`、`kube-controller-manager`、およびクラウド設定の場合は `cloud-controller-manager` です。
2. **制御コンピューティンググループ** HM の制御コンピューティンググループは、2 または 3 つの Kubernetes ワーカーノードで実行されている 70 以上のポッドの論理グループです。Grafana、Loki、Thanos、オブザーバビリティの Prometheus、証明書管理の Cert Manager、ネットワーキングの Istio、ソフトウェアリリースを保護するためのトラストマネージャーなど、HM のコア管理コンポーネントを実装しています。

コンポーネントの 70 以上のポッドは、`kube-scheduler` がそれらを配置する方法に従って、ワーカーノード全体に分散されます。

必要に応じて、ワーカーノードを追加して、より多くの Database-as-a-Service リソースをサポートできます。

3. **データコンピューティンググループ** HM のデータコンピューティンググループは、名前空間によって編成された Postgres クラスターの論理グループです。`kube-scheduler` が配置すると、少なくとも 3 つの Kubernetes ワーカーノードに分散された多数のポッドで実行されます。

HM を使用してデータベースが追加されると、ワーカーノードの数を 3 つから増やすことができます。ただし、これを自動的に行うには、ワーカーノードを手動で追加するか、インフラストラクチャとともに自動スケーリングを構成する必要がある場合があります。

制御コンピューティンググループのワーカーノードとデータコンピューティンググループのワーカーノードは、両方ともスナップショット SAN または EKS で EBS を使用を使用するオンプレミスまたは Barman for the cloud を使用する S3 互換オブジェクトストアにバックアップされます。

3.6.1.1 スケーラビリティ

スケーラビリティをサポートするために、HM は Kubernetes の自動スケーリング機能と統合しています。これは、AWS のようなプラットフォームで、データコンピューティンググループのワーカーノードを、AWS のマネージドサービス Auto Scaling groups を使用して、必要に応じてクラスターに自動的に追加できることを意味します。ただし、HM の Kubernetes 自動スケーリング機能を無効にして、コストを制御できます。さらに、オンプレミス展開の自動スケーリング機能を有効にするには、追加のインフラストラクチャ構成が必要であることに注意してください。これは、AWS の Auto Scaling グループの場合と同様に、オンプレミス展開にはすぐに使える自動スケーリング機能がないためです。

included Grafana dashboards または using Kubernetes `kubectl commands <Tracing clusters back to their Kubernetes resources>` を使用して、データベースクラスターを実装するポッドとワーカーノードの分散とリソース使用量をモニタできます。

3.6.1.2 バックアップアーキテクチャ

HM は、データの完全性と復旧可能性を保証するための堅牢なバックアップオプションを提供します。データコンピューティンググループ内の各 HM ワーカーノードは、デフォルトで夜間バックアップを実行し、Barman クラウド用またはスナップショットバックアップを使用して保存されます。

スナップショットバックアップは、復旧時間を短縮するために利用できますが、ストレージレイヤーに互換性のある CSI ドライバーが必要ですクラウドベースのスナップショットとローカルディスクスナップショットの両方。論理ボリュームマネージャー LVM セットアップの制約を考慮すると、スナップショットはローカルリカバリに最適ですが、サイト間の共有アクセスが欠如しているため、マルチサイトまたはグローバルリカバリシナリオのセットアップで使用する場合、制限に直面する場合があります。

より堅牢なマルチアベイラビリティーゾーンのリカバリーのために、Barman クラウドバックアップは、マルチゾーンレプリケーションをネイティブにサポートする AWS S3 のようなオブジェクトストアに保存できます。または、ローカルオブジェクトストアでは、バックアップがリモートサイトにレプリケートされるように追加の構成が必要になる場合があります。ストレッチクラスターは、地理的に分散した場所全体でデータのレプリケーションと冗長性を有効にすることにより、これらの制限を緩和します。

3.6.1.3 高可用性と復元力

HM は、Kubernetes の実績のあるテクノロジーを活用して、自動フェイルオーバーメカニズムを介して高可用性を保証します。これらのメカニズムは、ストレッチクラスター内、アベイラビリティーゾーン全体、または同じリージョン内のゾーン間で機能します。このアーキテクチャは、スナップショットバックアップを介したより高速なバックアップと、大規模なデータセットのリカバリをサポートしていますが、マルチサイトのリカバリ機能は、基になるストレージインフラストラクチャに依存します。複数のラックまたはデータセンターにまたがるストレージエリアネットワーク SAN などの高度なストレージ構成を使用しているお客様の場合、HM はインフラストラクチャを活用して復元力を強化し、迅速なフェイルオーバーを保証できます。ただし、これらの構成はすべてのユーザーで共通ではない場合があります。

3.6.2 Supported database types

Hybrid ManagerHM で新しいクラスターを作成するときに適切な Postgres データベースクラスタータイプを選択することは、データベース環境を最適化するために重要です。EDB は、基本的なシングルノード展開から復元力の高い分散ソリューションに至るまで、さまざまなニーズに合わせて調整された幅広いソリューションを提供します。

3.6.2.1 分散高可用性

分散高可用性 DHA クラスターは、 EDB Postgres 分散 PGD を搭載し、マルチマスター論理レプリケーションを活用します。このアーキテクチャにより、マルチプルのリージョンにわたって真のアクティブ/アクティブソリューションが可能になり、最高レベルの可用性と復元力を提供します。

次の場合にこのオプションを選択します。

- 組織は、超高可用性とほぼゼロのダウンタイムを必要としています。
- アプリケーションは、ユーザーの低遅延アクセスを備えた分散展開を必要とします。
- ビジネスには、マルチプルのアクティブノードにわたる継続的なオペレーションとデータの一貫性が最重要であるミッションクリティカルなアプリケーションがあります。
- シナリオには、可用性に影響を与えることなく、中断のないオンラインデータベースのメンテナンスとインプレースメジャーバージョンのアップグレードが必要です。

3.6.2.2 高度な高可用性

高度な高可用性 AHA は、基本的なプライマリ/スタンバイセットアップを超えた堅牢な HA ソリューションを提供することに焦点を当てており、より高いアップタイム保証とより洗練されたリカバリメカニズムを目指しています。これには、さまざまな強化された HA 構成が包含され、多くの場合 EDB の専用ツールと専門知識を使用して、優れた復元力を提供します。DHA クラスターとは異なり、AHA クラスターは単一のリージョンを使用し、そのリージョン内で高可用性を提供するように設計されています。

次の場合にこのオプションを選択します。

- チームは、高度な EDB Postgres 分散機能を活用し、将来の完全な DHA への移行を可能にする DHA よりもシンプルなセットアップを好みます。
- ビジネスでは、複雑さと高いアップタイムのバランスが必要で、完全なマルチマスター分散アーキテクチャなしでより強力な復元力を提供します。
- 企業は、標準のプライマリ/スタンバイ構成を超える高可用性の強化が必要です。
- 組織は、より柔軟な展開オプションを使用して、計画的なメンテナンスと計画外の停止のダウンタイムを最小限に抑えようとしています。

3.6.2.3 高可用性

高可用性 HA プライマリ/スタンバイ構成には、通常、プライマリデータベースサーバーと 1 つ以上のスタンバイレプリカサーバーが関係します。スタンバイサーバーは、プライマリからデータの変更を継続的に受信し、プライマリに障害が発生した場合に引き継ぎます。このセットアップでは、物理的ストリーミングレプリケーションを使用します。

次の場合にこのオプションを選択します。

- アプリケーションはデータの耐久性と最小限のダウンタイムでの継続的なサービスを必要とし、厳密なアクティブ-アクティブマルチマスターは要件ではありません。
- ユースケースには、専用の *read-only connection endpoint* を介してホットスタンバイレプリカにクエリをオフロードすることで利益を得ることができる読み取り中心のワークロードがあります。

3.6.2.4 単一ノード

シングルノード展開は、スタンバイまたはレプリカサーバーなしで実行される単一の PostgreSQL データベースインスタンスで構成されます。これは最も単純な構成であり、クリティカルでないワークロードまたは開発環境に適しています。

このオプションは次の場合に選択します。

- 高可用性が懸念されない開発およびテスト環境。
- メンテナンスまたはアップグレード、または障害の場合のダウンタイムを許容できる非クリティカルアプリケーション。
- 何よりもシンプルさと最小限のインフラストラクチャコストを優先するユースケース。
- トラフィックとデータ量の少ない小規模なアプリケーション。

3.6.2.5 クラスタータイプのアーキテクチャ

- 分散高可用性 *DHA* クラスターのネットワーク遅延
- *Advanced high availability*
- *Langflow* フローの高可用性と分離
- 単一ノード

3.6.2.5.1 Distributed high availability

分散高可用性クラスターは、 EDB Postgres Distributed を搭載しています。マルチマスター論理レプリケーションを使用して、物理レプリケーションベースのシステムよりも高度なクラスター管理を提供します。分散高可用性クラスターを使用すると、複数のリージョンまたは単一のリージョンにクラスターを展開できます。リージョン全体の高可用性が要件であるユースケースの場合、分散高可用性を備えたクラスター展開は、2 つの異なるリージョンに最大 2 つのデータグループと、3 番目の別のリージョンの監視グループを提供できます。

この構成は、各データグループが書き込み操作を受け入れるように構成されているため、真のアクティブ-アクティブソリューションを提供します。

分散高可用性クラスターは、EDB Postgres Advanced Server と EDB Postgres Extended Server データベースディストリビューションの両方をサポートしています。

分散高可用性クラスターには、1 つまたは 2 つのデータグループが含まれます。データグループには、3 つのデータノード、または 2 つのデータノードと 1 つの監視ノードを含めることができます。任意の時点で、各グループのこれらのデータノードの 1 つがリーダーであり、書き込みを受け入れますが、残りは

shadow nodes と呼ばれます。非同期 commit scopes を使用する場合を除き、本番では 2 つのデータノードと 1 つの監視ノードを使用しないことをお勧めします。

PGD Connection Manager は、すべてのアプリケーショントラフィックを書き込みリーダーノードにルーティングできます。各グループには独自の書き込みリーダーがあり、分散データの競合が削減されます。PGD は、分散コンセンサスモデルを活用して、クラスター内のデータノードの可用性を判断します。リーダーが失敗または使用不可になると、PGD は新しい書き込みリーダーを選択し、アプリケーショントラフィックをリダイレクトします。EDB Postgres 分散のコア機能と合わせて、アプリケーショントラフィッ

クをリーダーノードにルーティングするこのメカニズムにより、高速なフェイルオーバーとスイッチオーバーが可能になります。

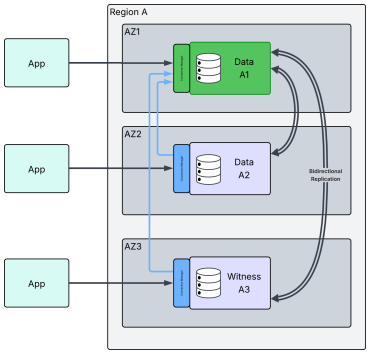
監視ノード/Witness グループはデータをホストしませんが、管理のために存在します。リージョンに障害が発生した場合など、コンセンサスを必要とする操作をサポートします。

!!!note 分散高可用性クラスターに対する操作は、計画的なライフサイクル操作中に 1 秒未満の中断を提供する EDB Postgres Distributed set-leader 機能を活用します。

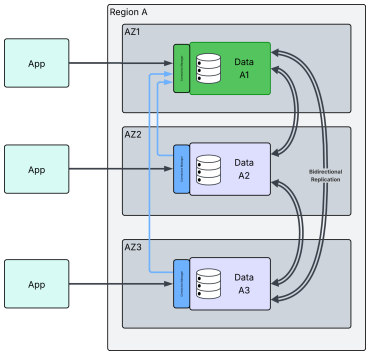
単一のデータの場所

単一のデータの場所を持つ構成には、1 つのデータグループと次のいずれかがあります。

- 1 つのリード、1 つのシャドウ、および監視ノードを備えた 2 つのデータノード、それぞれが別のアベイラビリティーゾーンにある



- 1 つのリードノードと 2 つのシャドウノードを備えた 3 つのデータノード、それぞれが別のアベイラビリティーゾーンにある



監視ノードを使用する複数のグループ

複数のデータの場所を含む構成には、各グループに 3 つのデータノードのいずれかを含む 2 つのデータグループがあります。

- 1 つのグループ内の 1 つのデータノードと 2 つのシャドウノード
- 別のグループの同じ構成
- 3 番目のグループの監視ノード

グループは現在同じリージョンクラスターが存在するリージョンにありますが、これは将来的に変更される可能性があります。

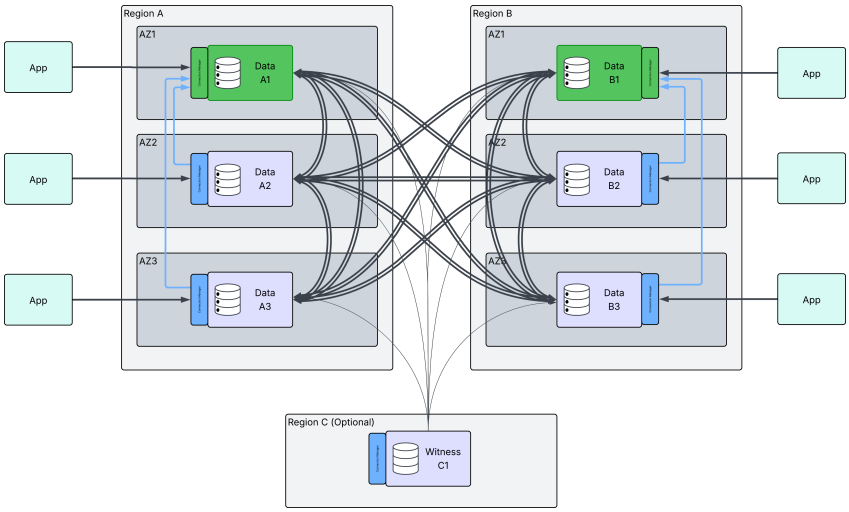


図 1: groups(2 x 3 data + 1 witness)

アドバイザリーロック

アドバイザリロックは Postgres ノード間でレプリケートされないため、シャドウノードで取得されたアドバイザリロックは、リーダーで取得されたアドバイザリロックと競合しません。アドバイザリーロックに依存するアプリケーションは、これらのトランザクションで読み取り専用のワークロードを使用しないことをお勧めします。

DHA クラスターは、データグループごとの読み取り専用接続もサポートし、現在書き込みリーダーとして指定されていないノードに読み取り専用操作をルーティングでき、パフォーマンスとワークロードの分離が向上します。これを有効にするには、*2* つのデータグループと *1* つの監視グループを含む *EDB Postgres* 分散 *PGD* クラスターが正常ではありません を参照してください。

詳細については

HM CLI を使用して分散高可用性クラスターを作成、取得、および管理する手順については、*Using the HM CLI* を参照してください。

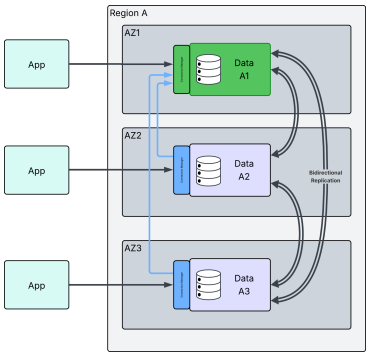
3.6.2.5.2 Advanced high availability

Advanced 高可用性 AHA クラスターは EDB Postgres Distributed を搭載し、PGD の Essential エディションをベースとしています。分散高可用性クラスターのトレースバック とは異なり、高度な高可用性クラスターは単一のリージョンを使用し、そのリージョンで高可用性を提供するように設計されています。

これらのクラスターは、分散高可用性クラスターよりも簡単なセットアップと少ない構成オプションを必要とするユーザー向けに設計されています。同時に、これらのユーザーは、EDB Postgres Distributed の高度な機能と、将来的に分散高可用性クラスターに簡単に移行できる機能の恩恵を受けます。

AHA クラスターは、マルチリージョン展開の複雑さなしで堅牢なフェイルオーバー機能を必要とするアプリケーションに最適です。高度な高可用性クラスターの基になるアーキテクチャは、分散高可用性クラスターのアーキテクチャと似ていますが、ユーザーが PGD 接続マネージャーすべてのノードに埋め込みを介して接続して、接続が現在の書き込みリーダーノードに送られるようにする必要があります。これにより、クラスター全体の競合を管理する必要がなくなり、アプリケーション開発が簡単になります。

この図は、接続マネージャーポートを介して PGD クラスターに接続するアプリケーションを示していま



す。接続マネージャーは、読み取りおよび書き込みトランザクションをグループ内の適切なノードにルーティングします。書き込みリーダーは、すべての書き込みトランザクションを処理する責任を負い、AZ1の上部に緑色で表示されます。

グループ内の他のノードは、書き込みリーダーからデータを複製する読み取り専用ノードです。読み取り専用ノードの接続マネージャー読み取り/書き込みポートに接続しているアプリケーションは、クエリと変更が書き込みリーダーにルーティングされます。常に、ノードは互いに通信し、データを複製して同期していることを確認します。

AHA クラスターは、EDB Postgres Advanced Server と EDB Postgres Extended Server データベースディストリビューションの両方をサポートしています。コミュニティ PostgreSQL ディストリビューションはサポートされていません。

AHA クラスターは、データグループごとの読み取り専用接続もサポートし、現在書き込みリーダーとして指定されていないノードに読み取り専用操作をルーティングでき、パフォーマンスとワークロードの分離が向上します。これを有効にするには、*2* つのデータグループと *1* つの監視グループを含む *EDB Postgres* 分散 *PGD* クラスターが正常ではありません を参照してください。

3.6.2.5.3 High availability

プライマリ/スタンバイの高可用性により、障害が発生した場合のダウンタイムを最小限に抑えます。プライマリ/スタンバイの高可用性クラスター 1 つの**プライマリ** と 1 つまたは 2 つの **スタンバイレプリカ**は、自動的に構成されます。スタンバイレプリカは、物理的ストリーミングレプリケーションを介して最新の状態に保たれます。

読み取り専用ワークロードが有効になっている場合、スタンバイレプリカは読み取り専用ワークロードを提供します。2 ノードクラスターでは、単一のスタンバイレプリカが読み取り専用のワークロードを処理します。3 ノードクラスターでは、両方のスタンバイレプリカが読み取り専用のワークロードを処理します。接続は、2 つのスタンバイレプリカへの接続ごとにランダムに作成されます。クラスターに 読み取り専用の接続情報 を構成することにより、この機能を有効にできます。これは、標準の読み取り/書き込み接続文字列とともに専用の読み取り専用エンドポイントを提供します。

アベイラビリティーゾーンを備えたクラウドリージョンでは、クラスターがゾーンを超えてプロビジョニングされ、データセンターに障害が発生した場合のフォールトトレランスを提供します。

プライマリが一時的または永続的に使用不可になった場合、スタンバイレプリカがプライマリになります。

着信クライアント接続は、常に現在のプライマリにルーティングされます。プライマリに障害が発生した場合、スタンバイレプリカがプライマリに昇格し、新しい接続が新しいプライマリにルーティングされます。古いプライマリが復旧すると、スタンバイレプリカとしてクラスターに再参加します。

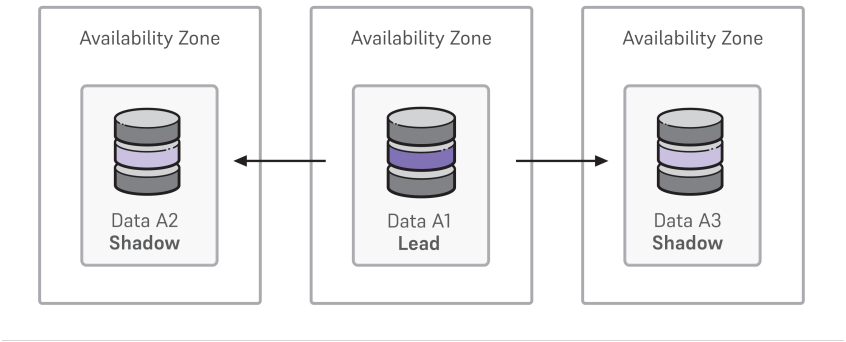


図 2: Primary/standby high availability

スタンバイレプリカ

デフォルトでは、レプリケーションは 1 つのスタンバイレプリカと同期し、もう 1 つとは非同期です。つまり、1 つのスタンバイレプリカは、クライアントが成功したコミットの確認を受信する前に、トランザクションレコードがディスクに書き込まれたことを確認する必要があります。

1 つのプライマリと 1 つのレプリカを持つクラスター 2 ノードプライマリ/スタンバイ高可用性クラスターでは、3 ノードと同じレベルの信頼性がないため、クラスターが書き込みに使用できなくなるリスクがあります。クラスター。ハイブリッドマネージャーは、2 ノードクラスターのメンテナンス操作中に同期レプリケーションを無効にして、書き込みの可用性を保証します。2 ノードクラスターのデフォルトの同期レプリケーションから、セッションごとまたはトランザクションごとに非同期レプリケーションに変更することもできます。

PostgreSQL では、`synchronous_commit` は `on` に設定され、`synchronous_standby_names` は `ANY 1 (replica-1, replica-2)` に設定されます。 `SET` または `ALTER` コマンドを使用して、トランザクションごと、セッションごと、ユーザーごと、またはデータベースごとにこの動作を変更できます。

ハイブリッドマネージャーは 1 つのノードにのみ同期してレプリケートするため、3 ノードクラスターの一部のスタンバイレプリカではレプリケーションラグが発生する場合があります。また、ハイブリッドマネージャーの同期レプリケーション構成をオーバーライドすると、スタンバイレプリカに一貫性がなくなります。

3.6.2.5.4 Single-node cluster

高可用性が主な関心事ではない非運用ユースケースの場合、高可用性を有効にしていないクラスター展開は、フェールオーバーまたは読み取り専用ワークロードのスタンバイレプリカのないプライマリを 1 つ提供します。

プライマリに回復不能な障害が発生した場合、バックアップからの復元が必要です。

3.6.3 Known issues

これらは、ハイブリッドマネージャー HM イノベーションリリースで特定された現在既知の問題と制限です。回避策は、これらの問題の影響を軽減するのに役立つ場合があります。これらの問題は積極的に追跡され、将来のリリースで解決される予定です。

図 3: Single node cluster

3.6.3.1 マルチ DC

3.6.3.1.1 アップグレード後のマルチ DC 構成の損失

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

回避策 HM アップグレードまたはコンポーネントを元に戻した後、マルチ DC セットアップスクリプトを再度実行して、必要な構成を再適用します。

3.6.3.1.2 マルチ DC セットアップの場所ドロップダウンリストが空です

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

回避策この高度な回避策には、API から返されるイメージセット情報の量を制限するクラスター管理者権限が必要です。これには、`upm-image-library` および `upm-beacon` ConfigMap のイメージ検出タグルールの変更と、それに続く関連ポッドの再起動が含まれます。

回避策の詳細

回避策は、画像ライブラリと EDB Postgres AI エージェントコンポーネントが使用する正規表現タグルールを変更して、インデックス付けされる画像タグの数を一時的に制限します。これにより、API 応答サイズが削減され、ロケーションをロードできるようになります。

1. `upm-image-library` ConfigMap を見つけます。

```
kubectl get configmaps -n upm-image-library | grep upm-image-library
# Example Output: upm-image-library-ttkt29fmf7 1 5d3h
```

2. ConfigMap を編集し、各イメージ検出ルール `edb-postgres-advanced`、`edb-postgres-extended`、`postgresql` の下の `tags` ルールを変更します。既存の正規表現を制限的な正規表現に置き換えます。

```
# Snippet of the YAML you will edit in the ConfigMap
"imageDiscovery": {
  "rules": {
    "(^|.*/)edb-postgres-advanced$": {
      "readme": "EDB postgres advanced server",
      "tags": [
        "^(?P<major>\d+)\.(?P<minor>\d+)-2509(?P<day>\d{2})(?P<hour>\d{2})(?P<minute>
↪\d{2})    (?:-(?P<pgdFlavor>pgdx|pgds))?(?:-(?P<suffix>full))?$"
      ]
    },
    # ... repeat for edb-postgres-extended and postgresql ...
  }
}
```

注釈

マルチ DC セットアップを実行している場合、プライマリ HM クラスターでこの手順を実行します。

3. イメージライブラリーポッドを再起動します。

```
kubectl rollout restart deployment upm-image-library -n upm-image-library
```

1. upm-beacon ConfigMap を取得して、 EDB Postgres AI エージェント構成を変更します。

```
kubectl get configmaps -n upm-beacon beacon-agent-k8s-config
```

5. ConfigMap `beacon-agent-k8s-config` を編集し、各 `postgres_repositories` エントリー `edb-postgres-advanced` 、`edb-postgres-extended` 、`postgresql` の下の `tag_regex` ルールを変更します。

```
# Snippet of the YAML you will edit in the ConfigMap
postgres_repositories:
  - name: edb-postgres-advanced
    description: EDB postgres advanced server
    tag_regex: "^(?P<major>\d+)\.(?P<minor>\d+)-2509(?P<day>\d{2})(?P<hour>\d{2})(?
 P<minute>\d{2})(?:-(?P<pgdFlavor>pgdx|pgds))?(?:-(?P<suffix>full))?$"
  # ... repeat for edb-postgres-extended and postgresql ...
```

EDB Postgres AI エージェントポッドを再起動します。

```
kubectl rollout restart deployment -n upm-beacon upm-beacon-agent-k8s
```

これらの手順を完了すると、画像データのサイズが削減されるため、ロケーション API 呼び出しが成功し、場所が HM コンソールに正しく表示されます。

3.6.3.1.3 プロジェクト分離のための専用オブジェクトストレージ

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

回避策新しいプロジェクト用に専用のオブジェクトストレージを作成し、プロジェクトの分離を有効にします。

回避策の詳細プロジェクトリソースを分離する必要がある場合は、プロジェクトごとに プロジェクト分離のための専用オブジェクトストレージ を構成できます。

3.6.3.2 コアプラットフォームとリソース

3.6.3.2.1 OpenShift ユーザーのクロスストリームアップグレードの問題

!!!tip HM バージョン 1.4.0 以降で解決されました。

回避策この問題に使用可能な回避策はありません。これを解決するために、現在 HM アプリケーションバージョンから Operator バージョンを切り離しているところです。

3.6.3.2.2 upm-beacon-agent 複雑な環境ではメモリ制限が不十分です

!!!tip HM バージョン 1.3.1 以降で解決されました。

回避策 Kubernetes 展開に手動でパッチを適用して、upm-beacon-agent ポッドのメモリリソース制限を増やします。

3.6.3.2.3 バージョン 2026.2 へのアップグレード後に IdP 構成が失敗する

説明: 場合によっては、バージョン 2026.1 から 2026.2 にアップグレードすると、ID プロバイダー IdP 構成が HM コンソールに **Expiring Soon** ステータスを表示する場合があります。これは、移行プロセス中の競合状態が原因で発生します。この問題はすべての環境で発生するわけではありませんが、トリガーされると、ユーザーは IdP プロバイダーを介して正常に認証できません。

回避策この状態を解決し、構成を正しく復元できるようにするには、クラスター管理者が内部アプリケーションデータベースを手動で更新する必要があります。

HM コンソールで IdP 統合を再セットアップする前に**、内部 app-db データベースで次の SQL ステートメントを実行します。

```
UPDATE upm_api_admin.idp SET deleted_at = NOW() WHERE deleted_at IS NULL;
```

3.6.3.2.4 Langflow シークレットが欠落しているため、accm-server の起動に失敗する

!!!ヒント

HM バージョン 2026.3 以降で解決されました。

回避策 AI 機能がインストールされていない環境で accm-server を起動できるようにするには、upm-beaco-ff-base 名前空間にプレースホルダーシークレットを手動で作成する必要があります。

次の kubectl コマンドを使用して、空のシークレットを作成できます。

```
kubectl create secret generic langflow-secret -n upm-beaco-ff-base --from-
↪literal=LANGFLOW_NOT_INSTALLED=true
```

シークレットが存在すると、accm-server ポッドはボリュームをマウントし、**Running** 状態に移行できるようになります。

3.6.3.2.5 イメージをプルするためのハイブリッドマネージャーアプリでは kapp-controller サポートなし

説明: kapp-controller は、イメージをプルするためのマネージド ID/IAM ロールをサポートしていません。

回避策クラウドネイティブのマネージド ID は現在サポートされていないため、ユーザーはレジストリ認証に SecretRef Kubernetes シークレットを使用して PackageRepository を手動で構成する必要があります。

3.6.3.2.6 スパイアが破損すると、ビーコンサーバーがクラッシュループに入る

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 upm-beacon-server は、spire コンポーネントが壊れているか、使用できない場合にクラッシュループに入ります。spire は、セカンダリロケーションをサポートするように構成されたすべての HM 展開に必要であるため、壊れた spire コンポーネントは Postgres クラスターの展開を妨げます。

回避策 kubectl アクセスを持つ管理者として、ビーコンサーバーの構成マップを編集して、spire 統合を無効にします。実行します

```
kubectl edit cm -n upm-beacon beacon-server-config
```

次に、次の 2 つの spiffe 関連の構成項目を設定します。

```
general:
  spiffe:
    enabled: false
  spiffeid_controller:
    disabled: true
```

セカンダリロケーションを接続する前に、これらの設定を元に戻します。

3.6.3.2.7 SELinux が有効になっている RKE2 で SPIRE エージェントが起動に失敗する

説明 RHEL または強制モードの SELinux を使用する他の Linux ディストリビューションで実行されている RKE2 クラスターでは、SPIRE エージェントポッドが create UDS listener: listen unix /tmp/spire-agent/public/spire-agent.sock: bind: permission denied エラーで CrashLoopBackOff 状態になります。これは、SELinux がコンテナプロセスが HostPath ボリューム /run/spire/agent-sockets に Unix ドメインソケットを作成することをブロックするために発生します。SELinux のない環境 Ubuntu ベースの RKE2 クラスターなどは、影響を受けません。

回避策すべての RKE2 クラスターノードで SELinux を無効にします。このためには、/etc/selinux/config で SELINUX=disabled を設定し、ノードをリブートします。リブート後、getenforce が Disabled と

表示されていることを確認します。SPIRE エージェントポッドは自動的に復旧しますが、ポッドが既に `CrashLoopBackOff` 状態にある場合、バックオフタイマーのため数分かかる場合があります。

3.6.3.2.8 重複するオペランドイメージが同じタグと SHA を共有する場合、イメージライブラリの検出に失敗します

!!!tip HM バージョン 2026.4 以降で解決されました。

説明 2 セットの EDB 公式 PostgreSQL オペランドイメージが同じタグと SHA ダイジェストが異なるリポジトリ名でアップロードされている場合、イメージライブラリ検出プロセスは失敗し、クラスタープロビジョニングに使用できるイメージはありません。

回避策重複したオペランドイメージの 1 セットを削除します。画像ライブラリは、残りのセットを正常に検出します。

3.6.3.3 データベースクラスターエンジン

3.6.3.3.1 `max_connections` がデフォルト以外の場合、3 ノード PGD クラスターの作成に失敗する

説明初期クラスタープロビジョニング中に構成パラメーター `max_connections` がデフォルト以外の値に設定されている場合、3 データノード PGD クラスターの作成は失敗します。

回避策 デフォルトの `max_connections` 値を使用して PGD 3 データノードクラスターを作成します。クラスターが正常にプロビジョニングされた後、値を更新します。

3.6.3.3.2 AHA 監視ノードのリソースがオーバープロビジョニングされています

説明監視ノードを使用する高度な高可用性 AHA クラスターの場合、監視ノードは大規模なデータノードの CPU、メモリ、およびディスク構成を誤って継承し、不必要なリソースのオーバープロビジョニングが発生します。

回避策 pgdgroup YAML 構成を手動で更新して、監視ノードに必要な最小限のリソースを指定および構成します。

3.6.3.3.3 HA クラスターは、ストリーミングレプリケーション証明書の認証に `verify-full` の代わりに `verify-ca` を使用します

説明: レプリカクラスターは、ストリーミングレプリケーション認証に、推奨される最も安全な `verify-full` の代わりに、それほど厳密ではない `verify-ca` 設定を使用します。基になる CloudNativePGCNP クラスターが、特定の環境 GKE ロードバランサーなどで `verify-full` に必要な IP サブジェクト代替名 IP SAN をサポートしていないため、これは現在必要です。

回避策 なし。修正は、IP SAN をサポートする基になる CNP コンポーネントに依存しています。

3.6.3.3.4 2 番目のノードが遅すぎるため、大規模な HA クラスターに参加できません

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

回避策単一のノードにデータをロードして、HA にスケーリングするベストプラクティスを回避します。代わりに、最初から HA クラスターにデータを直接ロードします。大規模なクラスターを HA 構成に復元するための回避策はありません。

3.6.3.3.5 2 つのデータグループと 1 つの監視グループを含む EDB Postgres 分散 PGD クラスターが正常ではありません

!!!tip HM バージョン 1.4.0 以降で解決されました。

3.6.3.3.6 AKS デプロイメントは、リージョンごとにプライベートアクセスを備えた 1 つの PGD データグループに制限されています

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 Azure は、標準ロードバランサー `PrivateLinkServicesPerLoadBalancerLimitReached` ごとに 8 つのプライベートリンクサービスのハード制限を適用します。プライベートアクセスを持つ各 PGD データグループは 5 つのロードバランサーサービスノードごとに 1 つとグループサービスとプロキシサービスを必要とするため、AKS ベースの HM 展開は、リージョンごとにプライベートアクセスを持つ 1 つの PGD データグループのみをホストできます。読み取り専用接続を有効にすると、データグループごとに 1 つの追加ロードバランサーサービスが追加され、使用可能なクォータがさらに削減されます。

プライベートアクセスを備えた完全な 2 データグループと監視クラスターは、標準のロードバランサー制限の 8 で 12 のプライベートリンクサービススロットを消費します。これにより、単一の場所の AKS でのマルチノード PGD ブートストラップが妨げられます。

回避策データグループを複数の場所に展開して、ロードバランサーのクォータを分散します。たとえば、1 つのデータグループと監視を 1 つの場所に展開し、2 番目のデータグループを別の場所に展開します。単一の HM インスタンス内の同じ AKS リージョンにプライベートアクセスを使用して複数の PGD データグループを作成しないでください。

3.6.3.3.7 RKE で明示的に選択されていない場合、ネットワークアクセスタイプはデフォルトの `Private` になります

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 RKE 環境では、2 つの関連するネットワークアクセスタイプの問題が存在します。

- 場所で **パブリック** と **プライベート**の両方のアクセスが有効になっている場合、ネットワークアクセスタイプフィールドにはデフォルト選択があります。オプションを明示的に選択しない場合、バックエンドはデフォルトの `Private` になります。
- 場所で **ノードポート**が有効になっている場合、バックエンドはクラスターのノードポートを正しく構成しますが、HM コンソールには代わりに `Private` としてアクセスタイプが表示されます。

回避策 RKE でクラスターを作成するときは、目的のネットワークアクセスタイプを常に明示的に選択します。将来のリリースに向けて修正が追跡されています。

3.6.3.3.8 ブートストラップ Job 作成が中断されると、CNPG オペレーターが無限リコンサイルループに入る

説明 PVC の作成後、ブートストラップ Job が作成される前に、オプティミスティックロックの競合などにより、クラスターリコンシリエーションが中断された場合、後続のリコンサイルは無限ループに入ります。オペレーターは、PVC が Ready ステータスに到達するのを待機しますが、これは Job なしでは発生せず、ループが無期限に継続します。

回避策影響を受ける PVC を削除して、オペレーターがそれを再作成し、ループを中断できるようにします。この問題は、新しく作成された PVC にのみ影響を与えるため、PV にはまだデータが含まれず、削除しても安全です。

```
kubectl delete pvc <pvc-name> -n <cluster-namespace>
```

3.6.3.4 バックアップ リカバリー

3.6.3.4.1 リージョンをまたがってボリュームスナップショットリカバリーを使用する場合、レプリカクラスターの作成が失敗する

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

回避策最初にプライマリクラスターから Barman バックアップを手動でトリガーします。次に、その Barman バックアップをボリュームスナップショットの代わりに使用して、クロスリージョンレプリカクラスターをプロビジョニングします。

3.6.3.4.2 ボリュームスナップショットの復元は同じリージョンに制限されています

説明 HM は、内部バックアップメカニズムを使用してレプリカクラスターのクロスリージョンデータの同期を自動的に処理しますが、ボリュームスナップショットからのクラスターの手動復元は、クロスリージョン操作をサポートしていません。ボリュームスナップショットを使用してクラスターを別のリージョンの場所に復元しようとすると、スナップショットは地理的にソースリージョンに制限されているため、操作は失敗します。

回避策別のリージョンの場所にデータを復元するには、ボリュームスナップショットの代わりに Barman バックアップを使用します。Barman バックアップは、リージョンを超えてアクセスできます。

3.6.3.4.3 デフォルトの並列構成が原因で WAL アーカイブが遅い

説明 wal.maxParallel のデフォルト設定は制限が多すぎるため、大量のデータのロード中に WAL アーカイブが遅くなります。これにより、アーカイブの準備ができた WAL ファイルのバックログが発生し、ディスクがいっぱいの状態が発生する可能性があります。このパラメーターは、HM コンソールではまだ設定できません。

回避策特定のバックアップオブジェクトストアの objectstores.barmancloud.cnpg.io Kubernetes リソースを手動で編集し、wal.maxParallel 値を 20 などに増やして、アーカイブを高速化します。

3.6.3.4.4 transporter-db WAL ギャップのため、ディザスターリカバリー DR プロセスが失敗する場合がある

!!!tip HM バージョン 1.4.0 以降で解決されました。

回避策復元を成功させるには、すぐに少なくとも 1 つのアーカイブされた WAL ファイルがある古いバックアップを選択して復元します。これにより、必要なトランザクションログをリカバリプロセスで利用できるようになります。

3.6.3.4.5 クラスターの完全なリカバリーは pg_ctl: server does not shut down エラーで失敗します

説明競合状態では、HM によって作成および管理される HA アーキテクチャを備えたクラスターへの復元に失敗する場合があります。完全復旧ポッドのエラーログには、次のようなエントリが含まれます。

```
pg_ctl: server does not shut down
pg_ctl: waiting for server to shut down.....................................
↪..... failed
controller: Error while deactivating instance err=error stopping PostgreSQL instance:␣
↪exit status 1
```

回避策最初にシングルノードクラスターに復元します。復元が完了したら、クラスターを HA アーキテクチャに拡張します。

3.6.3.5 Agent Factory とモデル管理

3.6.3.5.1 プロファイルキャッシュを使用した nim-nvidia-nvclip モデルの展開の失敗

説明展開プロセス中にプロファイルキャッシュが使用される場合、エージェントファクトリーで nim-nvidia-nvclip モデルの作成が失敗します。

回避策管理者は、必要なモデルプロファイルを NVIDIA レジストリからローカルマシンに手動でダウンロードする必要があります。次に、プロファイルファイルを HM のオブジェクトストレージパスに直接アップロードする必要があります。次に、特定の環境変数を使用して Kubernetes InferenceService YAML にパッチを適用して、失敗したネットワークダウンロードを試行する代わりに事前キャッシュされたファイルを使用するように強制することにより、モデルを展開します。

回避策の詳細

1. NGC API キーを使用して、NVIDIA Container Registry nvcr.io にログインします。

```
docker login nvcr.io -u $oauthtoken -p $NGC_API_KEY
```

2. Docker イメージをローカルマシンにプルします。

```
docker pull nvcr.io/nim/nvidia/nvclip:latest
```

3. ダウンロードしたプロファイル用のローカルディレクトリを準備します。

```
mkdir -p ./model-cache
chmod -R a+w ./model-cache
```

4. ターゲット GPU のプロファイルを選択します。

例、A100 GPU プロファイル `9367a7048d21c405768203724f863e116d9aeb71d4847fca004930b9b9584bb6`

5. コンテナを実行してプロファイルをダウンロードします。コンテナは CPU 専用モード `NIM\_CPU\_ONLY=1` で実行され、ダウンロードマシンでの GPU 固有の初期化問題を防ぎます。

```
export NIM_MANIFEST_
↪PROFILE=9367a7048d21c405768203724f863e116d9aeb71d4847fca004930b9b9584bb6
export NIM_CPU_ONLY=1

docker run -v ./model-cache:/opt/nim/.cache -u $(id -u) -e NGC_API_KEY -e NIM_CPU_
↪ONLY -e NIM_MANIFEST_PROFILE --rm nvcr.io/nim/nvidia/nvclip:latest
```

このコンテナは終了しません。ログに `Health method called` という行が表示されたら、実行を手動で停止する必要があります Ctrl+C 。これは、プロファイルのダウンロードが完了したことを確認します。

6. ローカルマシンから HM 展開が使用するオブジェクトストレージバケットにプロファイルをアップロードします。

```
gcloud storage cp -r ./model-cache gs://uat-gke-edb-object-storage/model-cache/nim-
↪nvidia-nvclip
```

!!!note

`gs://` パスを調整して、展開の構成済みオブジェクトストレージの場所と一致します。

7. HM コンソールを使用してモデル `nim-nvidia-nvclip` を作成し、**Model Profiles Path** フィールドを以前の場所たとえば`/model-cache/nim-nvidia-nvclip` などとして指定します。展開は、最初は失敗するか、スタックします。

8. HM Kubernetes クラスターから InferenceService YAML をエクスポートします。

9. 必要な環境変数 `NIM_IGNORE_MODEL_DOWNLOAD_FAIL` を、エクスポートした YAML の `spec.predictor.model` ブロックの env セクションに追加します。このフラグは、ローカルで利用可能なキャッシュアップロードしたファイルを使用し、ネットワークダウンロードの失敗を無視するように NIM コンテナに指示します。

```
# --- Snippet of the modified InferenceService YAML ---
spec:
  predictor:
    minReplicas: 1
    model:
      modelFormat:
        name: nim-nvidia-nvclip
      name: ""
      env:
```

(次のページに続く)

(前のページからの続き)

```
      - name: NIM_IGNORE_MODEL_DOWNLOAD_FAIL  # <-- ADD THIS LINE
        value: "1"                            # <-- ADD THIS LINE
      resources:
        # ... resource requests/limits ...
      runtime: nim-nvidia-nvclip
      storageUri: gs://uat-gke-edb-object-storage/model-cache/nim-nvidia-nvclip
# ---------------------------------------------------
```

10. kubectl を使用して変更した YAML を適用して、事前ダウンロードされたプロファイルを使用するように展開を強制します。

```
kubectl apply -f <modified-inference-service-file.yaml> -n <model-cluster-namespace>
```

オブジェクトストレージを使用して手動で提供したモデルプロファイルを使用して、ポッドが正常に起動します。

3.6.3.5.2 モデルプロファイルのオブジェクトストレージパスが空の場合、AI モデルクラスターの展開が停止する

!!!ヒント

HM バージョン 2025.11 以降で解決されました。

回避策 モデルクラスター展開を開始する前に、 **Model Profiles Path** フィールドで指定されたオブジェクトストレージパスに、正しい有効なプロファイルが含まれていることを確認します。

3.6.3.5.3 イノベーションリリースのアップグレード後にモデル構成設定がリセットされる

!!!tip HM はグリップテープの代わりに Langflow を使用するようになったため、この問題は適用されません。

説明 HM を `2025.11` から `2025.12` イノベーションリリースにアップグレードした後、既存のモデル構成設定が意図せずに空にリセットされます。これにより、`model_not_found` エラーが発生し、AI サービスへのアクセスが防止され、ナレッジベース KB が HM コンソールに表示されないなどの問題が発生します。

回避策 HM コンソールで、必要なモデル構成を手動で再入力または設定して、機能を復元します。

3.6.3.5.4 ナレッジベースの資格情報を編集するとエラーページが表示される

!!!tip HM はグリップテープの代わりに Langflow を使用するようになったため、この問題は適用されません。

説明: HM コンソールのパイプラインから作成されたナレッジベース KB を編集するときに、ユーザー名フィールドをスキップして、すぐにパスワードフィールドに移動すると、クライアントサイドの JavaScript エラー null のプロパティを読み取ることができません、**Unexpected Application Error!** ページが発生します。

回避策 エラーページが表示されないようにするには、 **Edit KB** を選択した直後、パスワードを入力する前に **Username** フィールドを入力します。

3.6.3.5.5 PGD 監視ノードに aidb_users ロールが欠落しているため、AI 機能が防止されている

説明: aidb_users ロールエージェントファクトリー機能に必要およびその関連拡張機能は、クラスターの初期化中に PGD クラスターの監視ノードに正常に複製されません。この問題は、単一の監視ノード監視グループではなくを使用する PGD クラスターに固有であり、必要なロールが不足しているためにエージェントファクトリーでエラーが発生します。

回避策必要なロールを手動でインストールし、エージェントファクトリーが機能できるようにするには、PGD 監視ノードで次の SQL コマンドを直接実行します。

```
begin;
set local bdr.ddl_replication = off;
SET LOCAL bdr.commit_scope = local;
create user aidb_users;
commit;
```

3.6.3.5.6 GenAI Builder の構造とツールの実行失敗

!!!ヒント

HM バージョン 2025.12 以降で解決されました。

```
[Errno -2] (Name or service not known)
```

回避策 GenAI Builder で構造を作成するときに、 **Create Structure** パネルで次の環境変数を設定します。

- **``GT_CLOUD_BASE_URL``** 完全なプロジェクトパスに設定する必要があります `https://<PORTAL URL>/apps/genai-builder/projects/<PROJECT ID>`
- **``REQUESTS_CA_BUNDLE``** 次の証明書パスに設定する必要があります。`/etc/ssl/certs/hcp.crt`

変数のスクリーンショット

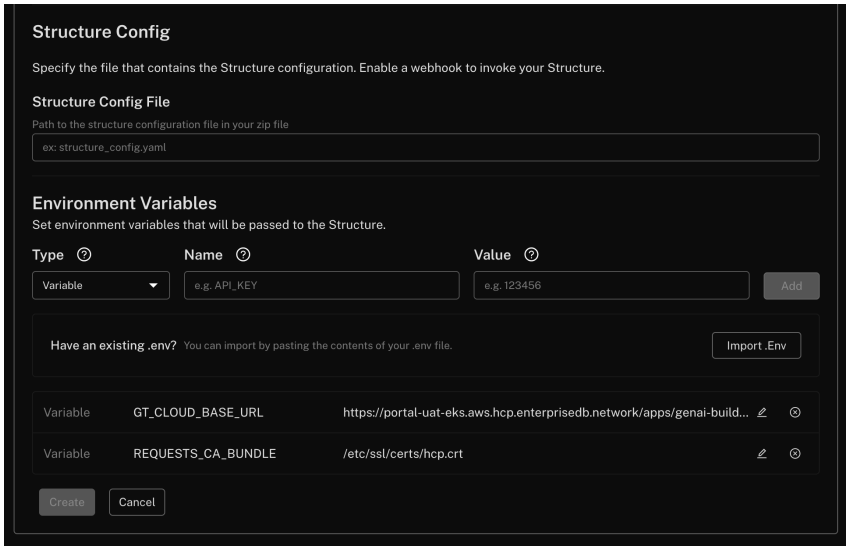


図 4: Screenshot of environment variables to be set in the UI

3.6.3.5.7 パイプラインデザイナーが作成したナレッジベースは GenAI Builder に表示されません

説明 HM 2025.12 では、パイプラインデザイナー PD で作成されたナレッジベース KB は visual_pipeline_user ロールに割り当てられます。これは、厳密な分離を適用するために行われ、HM ユーザーがデフォルトで SQL 作成の KB にアクセスできないようにします。その逆も可能です。ただし、この分離により、GenAI Builder はすぐにこれらの KB を照会できません。

回避策 PD で作成した KB を特定の PostgreSQL ユーザーアカウントと明示的に共有して、GenAI Builder でクエリ可能にします。

回避策の詳細

サンプルシナリオ

Alice は alice@acme.org として HM に接続し、データベースユーザー alice として PostgreSQL に接続します。彼女は PD KB を alice データベースユーザーと共有して、GenAI Builder エージェントが彼女の資格情報を使用してそれらを照会できるようにする必要があります。

1. 前提条件 ユーザーの存在を確認する

ターゲット PostgreSQL ユーザーこの例では alice が存在し、aidb_users ロールが割り当てられていることを確認します。ユーザーが存在しない場合は、次を実行します AIDB のインストールに EDB ドキュメントに従っていることを前提としています。

```
CREATE USER alice WITH PASSWORD  ****  **** ;
GRANT CONNECT ON DATABASE <some_db> TO alice;
GRANT CREATE ON SCHEMA <some_schema> TO alice;
GRANT aidb_users TO alice;
```

2. ロールアクセスの付与 回避策

ユーザーが PD で作成したナレッジベースを表示および照会できるようにするには、visual_pipeline_user ロールのメンバーシップを付与します。

```
GRANT visual_pipeline_user TO alice;
```

3. 設定

設定した alice 資格情報のユーザー名とパスワードを使用するように GenAI Builder エージェント構成を更新します。

3.6.3.5.8 外部推論サービスパラメーターを編集すると、名前変更エラーが返されます

!!!ヒント

HM バージョン 2026.3 以降で解決されました。

回避策外部推論サービスのパラメーターを更新するには、既存のサービスを削除し、目的の構成で新しいサービスを作成する必要があります。この問題は、特に外部モデルサービスに影響を与えますが、内部推論サービスには影響しません。

3.6.3.5.9 無効な外部モデル名の検証メッセージがありません

!!!ヒント

HM バージョン 2026.3 以降で解決されました。

両方の名前が有効であるためには、次のことを行う必要があります。

- 小文字の英数字、ハイフン- 、またはドット . のみで構成されます。
- 英数字で開始および終了します。

回避策登録時に指定されたすべての名前が文字要件を厳密に従っていることを確認します。外部サービスの登録後にページがハングしたように見える場合は、サービス名とモデル名に大文字またはサポートされていない特殊文字が含まれていないことを確認します。

3.6.3.5.10 AIDB v6.1 を使用する PGD クラスターでパイプラインの作成が失敗する

!!!tip AIDB 7.3.2 に同梱されている HM バージョン 1.4.0 以降で解決されています。

説明 HM バージョン 2026.3 では、 EDB Postgres DistributedPGD および AIDB v6.1 でエージェントファクトリーを使用する場合、レプリケートテーブルでのパイプラインの作成が失敗する場合があります。次のエラーが発生する場合があります。`ERROR: replicated relation <table_name> cannot have triggers that are owned by roles other than the relation owner (SQLSTATE 42501)` 。

これは、PGD では、トリガーとその関連トリガーハンドラーファンクションの両方が、テーブル通常は `visual_pipeline_user` を所有する同じロールが所有していることを必要とするために発生します。バージョン 2026.3 では、これらの機能はデフォルトで別の所有者に割り当てられる場合があります。

回避策この所有権の競合を解決するには、スーパーユーザーが AIDB トリガーハンドラーファンクションの所有権を `visual_pipeline_user` ロールに手動で譲渡する必要があります。

影響を受けるデータベースで次の SQL コマンドを実行します。

```
ALTER FUNCTION aidb.pipeline_background_trigger_handler OWNER TO visual_pipeline_user;
ALTER FUNCTION aidb.pipeline_live_trigger_handler OWNER TO visual_pipeline_user;
```

3.6.3.5.11 外部クラウドホスト OCR モデルは、HM コンソールを介して登録に失敗します

説明: `aidb.sync_hcp_models` ファンクションは、同期中にすべてのモデル URL に`/v1/infer` サフィックスを自動的に追加します。このサフィックスはローカルコンテナ化された NIM モデルに必要ですが、NVIDIA の PaddleOCR `.../cv/baidu/paddleocr` のような URL パスを使用するなどの外部クラウドホストモデルの接続を切断します。その結果、外部 OCR モデルは、HM コンソールを介して正常に構成または使用できません。

回避策 HM コンソール登録フローをバイパスし、`aidb.create_model` ファンクションを介して SQL を使用してデータベースにモデルを直接登録し、正しい URL を明示的に提供します。

オプション 1—リモート NVIDIA モデルのビルトインのデフォルト URL を使用します。

```
SELECT aidb.create_model(
    my_ocr_model,
```

(次のページに続く)

(前のページからの続き)

```
    nim_paddle_ocr,
    credentials => jsonb_build_object(api_key, current_setting(aidb.nvidia_nim_api_
↪key, true)),
    replace_credentials => true
);
```

オプション 2-カスタムの完全な URL を明示的に設定します。

```
SELECT aidb.create_model(
    my_ocr_model,
    nim_paddle_ocr,
    config => {"url":"https://ai.api.nvidia.com/v1/cv/baidu/paddleocr"}::JSONB,
    credentials => jsonb_build_object(api_key, current_setting(aidb.nvidia_nim_api_
↪key, true)),
    replace_credentials => true
);
```

3.6.3.6 アナリティクスと階層テーブル

3.6.3.6.1 階層化パーティションテーブルの更新が PGAA エラーで失敗する

!!!ヒント

HM バージョン 2026.2 以降で解決されました。

3.6.3.6.2 PostgreSQL 17 で一定の結果関係を使用して JOIN クエリーを実行すると、PGAA はサーバークラッシュ SIGSEGV を発生させます

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明: Postgres Analytics Accelerator PGAA 拡張機能がロードされているときに、一方が定数結果リレーションたとえば `SELECT aaa FROM v_dual LEFT JOIN bbb` または `SELECT aaa FROM (SELECT 1) d LEFT JOIN bbb` である `JOIN` クエリーを実行すると、SIGSEGV クラッシュが発生します。この問題は、PostgreSQL 17 のタイプの混乱が原因であり、3 つの異なるパスサブタイプが同じ `T_Result` パスタイプを共有します。

回避策 PGAA で結合と集計プッシュダウンを無効にします。

```
ALTER DATABASE <your_database> SET pgaa.enable_join_pushdown    = off;
ALTER DATABASE <your_database> SET pgaa.enable_groupby_pushdown = off;
```

Advanced 高可用性または分散高可用性 PGD クラスターを使用している場合、設定をすべてのノードにレプリケートします。

```
SELECT bdr.run_on_all_nodes($$
ALTER DATABASE <your_database> SET pgaa.enable_join_pushdown    = off;
ALTER DATABASE <your_database> SET pgaa.enable_groupby_pushdown = off;
$$);
```

3.6.3.7 HM コンソールと可観測性

3.6.3.7.1 Advanced High Availability および分散高可用性 PGD クラスターのフォールスポジティブバックアップアラート

!!!ヒント

HM バージョン 2026.2 以降で解決されました。

回避策 HM コンソールでの不必要な通知を回避するには、次のいずれかの方法を使用して、高度な高可用性および分散高可用性 PGD クラスターのこれらの特定のアラートをサイレントにできます。

- HM コンソールプロジェクトで、**Settings** > **Alerts** に移動し、`pem_time_since_last_backup_available_seconds` および `pem_backup_longer_than_expected` を沈黙させます。
- アラートマネージャー高度な高可用性および分散高可用性 PGD 環境をターゲットとするこれらの 2 つのアラートのサイレントルールを作成します。

3.6.3.7.2 2 つの異なるイベントを表示するときにイベントマーカーが重なる

!!!ヒント

HM バージョン 2026.3 以降で解決されました。

回避策イベントマーカーフィルターを使用して特定のイベントを表示し、イベントマーカーの重なりを回避します。

3.6.3.7.3 エステートページにアクセスするときの HTTP 431 “Request Header Fields Too Large”エラー

!!!ヒント

HM バージョン 2026.3 以降で解決されました。

回避策このエラーの影響を受けるユーザーの場合、組織所有者は次のいずれかのアクションを実行して要求ヘッダーのサイズを削減する必要があります。

- プロジェクトアクセスの取り消しアクティブに必要とされなくなったプロジェクトへの影響を受けるユーザーのアクセスを削除します。
- 未使用のプロジェクトの削除使用されなくなったプロジェクトを削除して、ユーザーのプロファイルに関連付けられている合計数を減らします。

3.6.3.7.4 タスクおよびクラスターのライフサイクルイベントの通知が見つからない

!!!tip HM バージョン 2026.4 以降で解決されました。ただし、回避策は 2026.3 へのアップグレードに引き続き適用されます。

説明: バージョン 2026.3 にアップグレードした後、ユーザーはクラスターライフサイクルイベントと自動化タスクトリガーされたスケーリングやメンテナンスなどのアプリ内通知と Webhook 通知の受信を停止する場合があります。この問題は、以前のバージョンの孤立したテーブルが新しいスキーマと競合し、アクティビティプロセッサの初期化を妨げた、データベースの移行 `005_activity_processor_setting.sql` の失敗が原因で発生します。新規インストールは影響を受けません。

回避策 HM 管理者は、不足している設定テーブルを手動で作成し、内部アプリケーションデータベースのタイムスタンプを初期化する必要があります。

upm-beaco-ff-base 名前空間にある app-db ポッド内で次のコマンドを実行します。

```
CREATE TABLE upm_notification_system.activity_processor_settings (
    last_processed_at TIMESTAMPTZ NOT NULL
);

INSERT INTO upm_notification_system.activity_processor_settings (last_processed_at)
VALUES (NOW());
```

3.6.3.7.5 WAL ストレージパーセントメトリックは、監視ページに誤った値を表示します

!!!tip HM バージョン 2026.4 以降で解決されました。

回避策 n/a

3.6.3.7.6 GUC 設定は再起動するまで適用されない

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明 shared_preload_libraries など、HM コンソールで構成された GUC 設定はすぐには適用されません。設定は、少なくとも 1 回の再起動が実行されるまで、以前の状態のままです。

回避策 HM コンソールで GUC 変更を適用した後、再起動をトリガーして、新しい設定を有効にします。

3.6.3.7.7 アラートしきい値の更新が断続的に有効にされない

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 HM コンソールで新しいアラートしきい値を保存した後—たとえば、メモリ使用量アラートしきい値を 90%から 85%に減らした後-変更は断続的に適用されず、しきい値は以前の値に戻ります。

回避策 アラート設定で、 **Notify After** フィールドが 0 より大きい値に設定されていることを確認します。現在 0 に設定されている場合、ゼロ以外の値に更新して保存します。

3.6.3.7.8 アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です

説明: すべてのアセットライブラリーアプリで、ユーザーがアクセス、展開、または管理するには、少なくとも**プロジェクトエディター** ロールが必要です。 **プロジェクト閲覧者**など、より低い権限ロールを持つユーザーは、アセットライブラリーアプリにアクセスできません。このロール要件は、すべてのアセットライブラリーアプリに適用され、現在アプリごとに構成できません。

回避策アセットライブラリーアプリにアクセスまたは管理する必要があるユーザーに、関連するプロジェクトの少なくとも**プロジェクトエディター**ロールが割り当てられていることを確認します。ロールの割り当ては、プロジェクト設定でプロジェクト所有者または組織の所有者が管理できます。

3.6.3.7.9 アセットライブラリーアプリは表示名ではなくパッケージ名で並べ替えられました

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 **アセットライブラリー**では、アプリは表示名ではなく内部パッケージ名で並べ替えられます。これにより、アルファベット順にブラウズしたときにアプリが予期しない位置に表示されます。たとえば、Apache Airflow と Apache Superset が予想通りに一緒に表示されない場合があります。

回避策なし。これが将来のリリースで解決されるまで、完全なアプリリストを手動で参照してアプリを見つけます。

3.6.3.7.10 EPAS クラスターをデータベースバックエンドとして使用する場合、メタベースの展開に失敗する

説明 選択したデータベースクラスターが EDB Postgres Advanced Server インスタンスの場合、 **アセットライブラリー**からのメタベースの展開は失敗します。Metabase は、Liquibase 移行エラー `ERROR: syntax error at or near "$"` を含む `CrashLoopBackOff` 状態になります。これは、Liquibase が EDB Postgres Advanced Server をコミュニティ PostgreSQL とは異なるデータベースタイプとして識別するために発生します。その結果、`dbms="postgresql"` `${TEXT.TYPE}` や`${TIMESTAMP_TYPE}` などにスコープされたプロパティ定義はスキップされますが、依存する変更セットは未解決のプレースホルダーで実行され、SQL 構文エラーが発生します。

回避策 EDB Postgres Advanced Server の代わりにコミュニティ PostgreSQL クラスターをメタベースのデータベースバックエンドとして使用します。EDB Postgres Advanced Server クラスターは、展開後も分析用のデータソースとしてメタベースに接続できます。

3.6.3.7.11 アセットライブラリーアプリ pgAdmin4 および Apache Superset は RHOS への展開に失敗する

!!!tip この問題は、HM バージョン 1.4 以降の Apache Superset で解決されました。pgAdmin4 では、この問題が引き続き発生しています。

説明: Red Hat OpenShift RHOS 環境では、pgAdmin4 および Apache Superset アセットライブラリーアプリケーションの展開が、OpenShift Security Context ConstraintSCC 違反のため失敗します。これらのアプリのアップストリームイメージは、OpenShift の制限された SCC ポリシーによって適用される許可 UID 範囲の外にある特定のユーザー ID たとえば、pgAdmin4 の `5050` と Apache Superset の init コンテナの `0` として実行されます。これは、上流のイメージの互換性の問題です。

回避策この問題が上流で解決されるまで、RHOS 環境に pgAdmin4 または Apache Superset を展開しないでください。

3.6.3.7.12 Airflow または Superset の暗号キーを編集すると、アプリが障害状態になる

!!!tip HM バージョン 2026.5 以降で解決されました。

説明初期展開後にアセットライブラリー UI を介して Airflow の `fernetKey` または `apiSecretKey` 、または Superset の `secretKey` を編集すると、アプリが障害状態になります。これらのキーは、接続、変数、セッション、CSRF トークンの暗号化を含む暗号化操作に使用され、基になる Helm チャートはキーのローテーションをサポートしていません。最初の作成後にこれらの値を変更すると、新しいキーと元のキーで暗号化されたデータの間で不一致が発生し、アプリケーションが中断されます。

回避策 最初のアプリの作成後は、 `fernetKey` 、`apiSecretKey` Airflow、または `secretKey` Superset を変更しないでください。キーの変更が原因でアプリが障害状態になった場合は、元のキー値で再展開して機能を復元します。

3.6.3.7.13 スーパーセットの展開は DuplicateTable エラーで失敗します

説明: アセットライブラリーからスーパーセットの展開は、`DuplicateTable` エラー `psycopg2.errors.DuplicateTable: relation "idx_user_id" already exists` で失敗する場合があります。これは、選択したバックエンドデータベースに、以前のスーパーセット展開からのテーブルが既に含まれている場合に発生します。

回避策スーパーセットを展開するときに、新しい空のデータベースを使用します。

3.6.3.7.14 アセットライブラリー起動アプリは、カスタムポータルポートに関係なく、常にポート 443 を使用します

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 HM が 443 以外のカスタム `portal_port` 値で構成されている場合、**Estate** のアプリまたはフロー URL は、構成されたポートの代わりにポート 443 を使用します。これにより、HM ポータルが標準の HTTPS ポートで公開されていない環境では、起動されたアプリまたはフローが到達不能になります。

回避策ポート 443 を、アクセスするアプリまたはフロー URL で構成済みの `portal_port` に手動で置き換えます。

3.6.3.7.15 アセットライブラリーアプリの CPU およびメモリメトリックには、失敗および完了したポッド状態が含まれます

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明: アセットライブラリーアプリに表示される CPU とメモリ使用量メトリックには、ジョブによって生成されたポッドなど、`Failed` または `Completed` 状態のポッドが含まれます。これらのメトリックは、アプリの起動フェーズ中にのみ使用できるため、メトリックが安定するまで、表示される数値はアプリの現在のリソース使用量を正確に反映していない場合があります。

回避策アプリの起動後、メトリックが安定して正しい状態を反映するまで 30 分間待ちます。

3.6.3.7.16 サービスアカウントが見つからないため、アプリまたはフローの削除が停止する

!!!tip HM バージョン 2026.5 以降で解決されました。

説明アプリまたはフローを削除すると、削除がエラー `Delete failed: Preparing kapp: Getting service account: serviceaccounts "beacon-app-installer" not found` でスタックする場合があります。これは、 `PackageInstall` リソースのファイナライザーが、存在しないサービスアカウントを使用しようとしているため、アプリの名前空間を削除できないために発生します。

回避策 スタックした名前空間を特定し、 `PackageInstall` およびそれに関連する `App` リソースからファイナライザーを手動で削除します。

1. `Terminating` でスタックしている名前空間を見つけます。

```
kubectl get ns --field-selector status.phase=Terminating
```

アプリ名前空間は、アプリケーション ID たとえば app-abc123xyz と一致します。

1. PackageInstall 名前を特定します。

```
kubectl get packageinstall -n <app-namespace>
```

名前は<app-namespace>-pkginstall 形式に従います。

1. PackageInstall および関連する App からファイナライザーを削除します App 名前は PackageInstall 名前と一致します。

```
kubectl patch packageinstall <name> -n <app-namespace> --type=json -p=[{"op": "remove
↪", "path": "/metadata/finalizers"}]
kubectl patch app <name> -n <app-namespace> --type=json -p=[{"op": "remove", "path":
↪"/metadata/finalizers"}]
```

1. 名前空間の削除が完了したことを確認します。

```
kubectl get ns <app-namespace>
```

3.6.3.7.17 ストレージクラスが異なる場合、RHOS でアセットライブラリーアプリが保留状態になることがある

説明クラスターのデフォルトのストレージクラスが spec.globalParameters.storage_class 値と異なる Red Hat OpenShiftRHOS 環境たとえば、デフォルトは lvms-vm コントロールプレーンノードによってバッキングされていますが、構成されたストレージクラスは lvms-bm (ワーカーノードによってバッキングされた) —pgAdmin4 などの永続ボリュームを必要とするアセットライブラリーアプリは、無期限に Pending 状態でスタックする場合があります。これは、アプリの PVC がクラスターのデフォルトのストレージクラスを継承するため、アプリポッドをスケジュールできるノードで使用可能な容量がない可能性があるために発生します。

回避策アプリの展開がスタックする環境では、アプリ名前空間で保留中の PVC を削除し、storageClassName を hcp.spec.globalParameters.storage_class の値に設定して再作成します。または、クラスターのデフォルトの StorageClass を hcp.spec.globalParameters.storage_class と一致するように変更して、すべての新しい PVC が自動的に正しいストレージを使用するようにします。

3.6.3.8 クラスターとアップグレード

3.6.3.8.1 不変の fluent-bit-k8s-events StatefulSet が原因でバージョン 2026.4 へのアップグレードが失敗する

説明 HM 2026.4 にアップグレードする場合、アップグレードプロセスは upm-fluent-bit の Failed to install component エラーで失敗する場合があります。これは、アップグレードが fluent-bit-k8s-events StatefulSet 仕様の不変フィールドを変更しようとするため、Kubernetes API サーバーが拒否するために発生します。エラーメッセージは StatefulSet.apps "fluent-bit-k8s-events" is invalid: spec: Forbidden: updates to statefulset

spec for fields other than 'replicas', 'ordinals', 'template', 'updateStrategy', 'persistentVolumeClaimRetentionPolicy' and 'minReadySeconds' are forbidden です。

回避策 2026.4 にアップグレードする前に、edb-observability 名前空間の fluent-bit-k8s-events StatefulSet を手動で削除します。オペレーターは、アップグレード中に正しい構成で再作成します。

```
kubectl delete statefulset fluent-bit-k8s-events -n edb-observability
```

3.6.3.8.2 監視グループを使用した高度な高可用性および分散高可用性 PGD クラスターのメジャーバージョンのアップグレードが失敗する

!!!ヒント

HM バージョン 2026.2 以降で解決されました。

回避策この特定のクラスター構成に対してインプレースメジャーバージョンのアップグレードを実行しないでください。代わりに、ターゲットメジャーバージョンを使用して新しい高度な高可用性または分散高可用性 PGD クラスターをプロビジョニングし、データ移行サービス DMS を使用して古いクラスターから新しいクラスターにスキーマとデータを移行します。

3.6.3.8.3 ロギング名前空間の終了が原因でバージョン 2026.3 へのアップグレードが失敗する

!!!tip HM バージョン 2026.4 以降で解決されました。ただし、回避策は 2026.3 へのアップグレードに引き続き適用されます。

回避策オプション 1 推奨

バージョン 2026.3 へのアップグレードを開始する前に、次のコマンドを実行して、クリーンアッププロセス中に名前空間を孤立させます。

```
kubectl annotate namespace logging kapp.k14s.io/delete-strategy=orphan --
↪overwrite=true
```

回避策オプション 2ff のアップグレードは既にスタックしています

アップグレードが既に開始され、logging 名前空間が Terminating 状態でスタックしている場合、次のコマンドを実行してファイナライザーを手動で削除し、アップグレードのブロックを解除します。

```
kubectl patch fluentbit/fluent-bit -n logging --type=merge -p {"metadata":{"finalizers
↪":[]}}
```

3.6.3.9 Agent Factory

3.6.3.9.1 ナレッジベースデータは、PGD クラスターの非プライマリノードにレプリケートされません

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明: PGD クラスターで AIDB 7.3.0 でエージェントファクトリーを使用する場合、ナレッジベースデータ aidb.knowledge_bases_v6 ビューで可視プライマリノードにのみ存在し、他のノードに自動的にレプリケートされません。これは、AIDB 7 で導入された aidb.knowledge_base_registry および aidb.knowledge_base_pipeline テーブルが PGD セットアップスクリプトに含まれていないために発生します。

回避策ナレッジベースを作成する前に、PGD クラスターで次の SQL コマンドを手動で実行します。

```
SELECT bdr.alter_sequence_set_kind(aidb.knowledge_base_registry_id_seq::regclass,␣
↪galloc, 1);
SELECT bdr.replication_set_add_table(aidb.knowledge_base_registry);
SELECT bdr.replication_set_add_table(aidb.knowledge_base_pipeline);
```

3.6.3.9.2 hypopg 拡張機能が欠落しているため、インデックス分析アクションが失敗します

説明: 必要な hypopg 拡張機能が標準の HM データベースイメージにないため、Agent Factory アクション `analyze_workload_indexes` および `analyze_query_indexes` は現在使用できません。これらのアクションを実行しようとすると、拡張機能がインストールされていないことを示すエラーが発生します。

回避策現在、管理対象クラスターにこの拡張機能をインストールする手動の回避策はありません。拡張機能が将来のイメージ更新に含まれるまで、これらの 2 つの特定のアクションは使用しないでください。

3.6.3.9.3 削除されたクラスターは AI コンポーネントに表示されたままです

!!!ヒント

HM バージョン 2026.2 以降で解決されました。

回避策 AI コンポーネントでクラスターを選択する前に、HM コンソールまたはクラスター API を介してそのステータスを確認します。現在「正常」または「Active 」状態にあるクラスターのみを選択してください。

3.6.3.9.4 必須 MCP ツールパラメーターの検証の欠落

説明 MCP ツール内のいくつかのアクションでは、構成中にフィールドを空のままにすることができます。これにより、検証警告ではなく実行エラーが発生します。これらのパラメーターが入力されない場合、ワークフローはビルドに失敗するか、「Unsupported object type」エラーを返す場合があります。

影響を受けるアクションとパラメーター

- `list_objects` および `get_object_details` Object Type フィールドは空であってはなりません。
- `explain_query` Analyze および `hypothetical_indexes` フィールドには、特定のデータ型を入力する必要があります。

回避策フローを実行する前に、次のパラメーターが手動で入力されていることを確認します。

- `list_objects` オブジェクトタイプ `table` 、`view` 、`sequence` 、`function` 、`stored procedure` 、または `extension` と入力します。
- `get_object_details` オブジェクトタイプ `table` 、`view` 、`sequence` 、または `extension` と入力します。
- `explain_query` 分析 `True` または `False` と入力します。
- `explain_query` `hypothetical_indexes` ツールモードを使用して、有効なリストを提供します。

3.6.3.9.5 一部の EDB コンポーネントツールは、互換性のない API 要求フォーマットのため機能しない

説明 EDB コンポーネント内の一部のツールは、互換性のない API 要求フォーマットのため、現在機能していません。デフォルトでは、コンポーネントは公開するツールが多すぎるため、API 制限を超え、エラーをトリガーする可能性があります。

回避策ツールモードを有効にし、特定のフィルタを適用して、コンポーネントによって公開されるツールの数を制限します。

3.6.3.9.6 高度な高可用性と分散高可用性 PGD 監視グループがデータベース選択リストに表示される

!!!ヒント

HM バージョン 2026.2 以降で解決されました。

回避策これらのコンポーネントを構成するときは、監視グループとしてラベル付けされたエントリを手動で無視します。データグループのみを選択して、有効な接続と正常なデータの取得を保証します。

3.6.3.9.7 テキスト検索は非表示フィールドに一致する場合があります

説明 HM コンソールの検索機能、たとえば **アクティビティログ**セクションを使用すると、システムはすべての基になるデータフィールドの全文検索を実行します。その結果、検索結果には、表示されている**アクティビティ名**またはその他の表示属性が検索用語と一致しないように見えるリソースまたはアクティビティログエントリが含まれる場合があります。

回避策これは、基になる検索デザインの結果であるため、現在この動作の回避策はありません。ユーザーは、検索結果には、プライマリログビューに表示されない内部アクションの詳細が含まれることに注意してください。

3.6.3.9.8 Langflow MCP サーバーコンポーネントは HM フローホスティングでサポートされていません

説明 Langflow MCP モデルコンテキストプロトコルサーバーコンポーネントを含むフローは、ハイブリッドマネージャーフローホスティング API を介して実行すると、次のエラーで失敗します。`Error building Component my_mcp: Error updating tool list: Langflow MCP server functionality is not available. This feature requires the full Langflow installation.` HM Flow Hosting ランタイム `lfx-runtime` は、軽量環境です。完全な Langflow インストールが含まれます。MCP サーバーコンポーネントを使用するフローは、正常に展開されたように見えますポッドは正常であると報告しますが、REST API を介して呼び出されると、実行時に失敗します。

回避策 HM Flow Hosting API を介した展開を目的としたフローに Langflow MCP サーバーコンポーネントを含めないでください。 Airman または HM MCP サーバーを実行する必要がある場合は、代わりに EDB Airman または EDB Platform コンポーネントを使用します。これらのコンポーネントは、lfx-runtime 環境と互換性があります。他の MCP サーバー機能の場合、代わりに完全な Langflow インタラクティブ環境を使用します。

3.6.3.9.9 EDB Knowledge Base コンポーネントを Langflow にドラッグすると 403 エラーを返します

!!!tip HM バージョン 2026.5 以降で解決されました。

説明 EDB ナレッジベースコンポーネントを Langflow にドラッグすると、左下隅にエラー `API call to .../projects/None/clusters returned 403` が表示されます。これは、コンポーネントのロード時に API 呼び出しにプロジェクト ID が欠落しているために発生します。

回避策 コンポーネントで **Hybrid Manager Project** ドロップダウンを選択し、 **リストの更新**を選択して使用可能なプロジェクトをロードします。プロジェクト ID が入力されると、エラーはクリアされます。

3.6.3.9.10 データベースユーザーのグローバル変数を使用する場合、EDB Knowledge Base コンポーネントは接続に失敗します

!!!tip HM バージョン 2026.5 以降で解決されました。

説明データベースユーザーフィールドのグローバル変数を使用して Langflow の EDB Knowledge Base コンポーネントを構成すると、接続試行が失敗します。初期接続セットアップ中に、このフィールドのグローバル変数値は正しく解決されません。

回避策グローバル変数ではなく生のテキスト値としてデータベースのユーザー名を入力します。

3.6.3.9.11 空のナレッジベースに対してテストクエリーを実行すると、500 エラーが返されます

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明 基になる埋め込みテーブルが空のナレッジベースに対して**ナレッジベース**リストビューからテストクエリーを実行すると、たとえば、KB パイプラインがまだ `Processing` 状態にあるときに、API は HTTP 500 エラー `ERROR: Query returned no data. Hint: The "<pipeline_name>" table is likely empty`。API は代わりに空の結果セットを返す必要があります。

回避策テストクエリを実行する前に、ナレッジベースパイプラインの処理が終了するまで待機します。データのインデックスが作成され、エンベディングテーブルにデータが設定されると、パイプラインステータスは `Processing` からアクティブ状態に変更されます。

3.6.3.9.12 フローホスティングの展開は `FlowValidationError` で断続的に失敗します

!!!ヒント

HM バージョン 2026.5 以降で解決されました。

説明 Langflow UI を介してフローを作成する場合、コンポーネントのシークレットまたは資格情報フィールドはグローバル変数を使用する必要があります。パスワードを直接貼り付けると、HM フローホスティングを介してフローを展開するときにエラーが発生します。これは、Langflow v0.19 からのすべてのコンポーネントのデフォルト動作です。

回避策 資格情報をグローバル変数に保存し、フロー構成で参照します。

3.6.3.9.13 EDB Model Server コンポーネントは、EDB Airman MCP コンポーネントと使用するとエージェント障害が発生します

!!!tip HM バージョン 1.4.0 以降で解決されました。

説明 EDB Airman MCP コンポーネントとともに EDB Model Server コンポーネントを使用する Langflow でエージェントフローをビルドすると、エージェントが `messages with role 'tool' must be a response to a preceding message with 'tool_calls'` のようなエラーで失敗する場合があります。この問題は断続的であり、他のモデルプロバイダーコンポーネントを使用する場合には発生しません。

回避策 EDB Model Server コンポーネントの代わりに OpenAI コンポーネントまたは汎用モデルコンポーネントを使用します。これらの代替モデルプロバイダーを使用するフローは、この問題の影響を受けません。

3.6.3.9.14 upm-ai-model-server-beaco メモリ制限の不足による OOMKills

!!!tip HM バージョン 2026.5 以降で解決されました。

説明: upm-ai-model-server-beaco ポッドは、重要な AI ワークロードがある環境での OOMKill が原因でクラッシュループに入る場合があります。このコンポーネントのデフォルトのメモリ割り当ては不十分であり、現在標準の Helm 値または HybridControlPlane CR を介して構成できません。

回避策 Kubernetes 展開に手動でパッチを適用して、upm-ai-model-server-beaco ポッドのメモリリソース制限をデフォルトの 256MB から 512MB に増やし、ポッドをモニタして安定性を確認します。

3.6.3.9.15 更新された Langflow フローの変更は、再公開後 HM にすぐに反映されません

説明: 以前にパブリッシュされた Langflow フローを HM Langflow エディターで編集し、アセットライブラリーに再公開すると、更新されたバージョンは HM にすぐに表示されません。変更が伝播されて表示されるまでに、最大 5 分の遅延が発生します。

回避策フローを再公開した後、約 5 分待って、変更が HM に反映されていることを確認します。

3.6.3.10 マイグレーション

HM 移行ポータル、データ移行ワークフロー、およびスキーマ取り込みワークフローに関連する既知の問題は、*Migrating databases with Hybrid Manager* ドキュメントの専用ページで維持されています。完全なリストについては、*Known issues, limitations, and notes* を参照してください。

3.6.4 Supported Postgres distributions

Hybrid ManagerHM で新しいデータベースクラスターを作成するときに適切な Postgres ディストリビューションを選択することは、データベース環境を最適化するために重要です。EDB は、オープンソースの柔軟性からエンタープライズグレードの機能、Oracle 互換性まで、さまざまなニーズに合わせた幅広いオプションを提供します。

3.6.4.1 EDB Postgres Advanced Server

EDB Postgres Advanced Server は、EDB で最も包括的な PostgreSQL ディストリビューションです。*advanced security* 、 *advanced SQL* 、高可用性、マルチマスターレプリケーションなどの機能をサポートします *2* **つのデータグループと** *1* **つの監視グループを含む** *EDB Postgres* **分散** *PGD* **クラスターが正常ではありません** およびオプションで Oracle compatibility 。要求の高いエンタープライズワークロードと、コストのかかる Oracle データベースからのシームレスな移行に最適です。

このオプションは次の場合に適しています。

- 移行コストと複雑さを削減するための高い互換性を求める Oracle データベースから移行する企業。
- 高度なセキュリティ、パフォーマンス、および管理ツールを必要とするミッションクリティカルなアプリケーションを持つ組織。
- 強化された監査、SQL インジェクション保護、堅牢な高可用性を含む包括的なエンタープライズグレードの機能を必要とするビジネス。

3.6.4.2 EDB Postgres Extended Server

EDB Postgres Extended Server は、エンタープライズ要件、特にデータセキュリティにおいて設計された特定の機能強化を提供します transparent data encryption などを介して。また、EDB の強力な高可用性マルチマスターレプリケーションソリューション *2 つのデータグループと 1 つの監視グループを含む EDB Postgres 分散 PGD クラスターが正常ではありません* を使用した高度なレプリケーションもサポートしています。

このオプションは次の場合に適しています。

- 強化されたデータセキュリティ機能を必要とする企業。
- EDB の高可用性およびマルチマスターレプリケーションソリューションを活用しようとしている組織。
- コミュニティ PostgreSQL との強い互換性を維持しながら、追加のエンタープライズ機能を備えた PostgreSQL エクスペリエンスを希望するユーザー。

3.6.4.3 PostgreSQL

オープンソースの PostgreSQL は、強力で信頼性の高い機能が豊富なリレーショナルデータベースシステムです。SQL 標準への強い準拠、堅牢なアーキテクチャ、データの完全性、およびグローバルコミュニティで知られています。

このオプションは次の場合に適しています。

- 費用対効果とオープンソースの自由を優先する組織。
- 高度なセキュリティ機能、マルチマスターレプリケーションソリューション、または Oracle の互換性を必要としないプロジェクト。
- 社内の PostgreSQL の専門知識とリソースとチームして、データベース環境を管理します。

3.6.4.4 ディストリビューションの比較

機能および機能ごとの使用可能なディストリビューションの比較については、以下を参照してください。

- *Feature support per distribution*
- *Advanced replication*
- *Advanced security*
- *Advanced SQL*
- *Advanced performance*
- *Oracle support*
- *Tools and extensions*
- *Deployment options*

3.6.4.5 ディストリビューション別のドキュメント

Postgres ディストリビューションを最大限に活用するには、そのドキュメントを参照してください。

- PostgreSQL
- *EDB Postgres Extended Server*
- *EDB Postgres Advanced Server*

第4章　Installing Hybrid Manager

4.1 ハイブリッドマネージャーのインストール

このセクションには、ハイブリッドマネージャー HM のインストールと構成に必要なすべてが含まれています。

- ** Installation guide ** —アーキテクチャ計画からインストール後の構成までをカバーする 6 フェーズのガイド。新規インストールの場合はここから開始します。
- ** **ステップ** *3* **タスク構成** ** — 高度な構成トピックの詳細な SOPs TLS、IdP、TDE の KMS、マルチ DC、画像同期など。
- ** *Migrate from bootstrap to operator* ** —オペレーター方法に移行する既存のブートストラップインストールの場合。
- ** *Troubleshooting Langflow in Hybrid Manager* ** — インストールの問題の診断と解決に関するガイダンス。

4.1.1 step-by-step guide

4.1.1.1 概要

このガイドでは、インフラストラクチャに Hybrid ManagerHM を計画、構成、およびインストールする手順を説明します。

プロセスは **6 つの時系列フェーズ**に分割されており、展開を続行する前に依存関係が満たされていることを確認します。

これらのフェーズを順番に実行してください。スキップする手順、最も重要な**フェーズ** *2* **システム要件の収集** または *Preparing your environment* は、インストールが失敗する最も一般的な原因です。

4.1.1.2 フェーズ 1 アーキテクチャの計画

> **重点ロール** CTO / アーキテクト / リードエンジニア

この最初のフェーズは、展開アーキテクチャの **デザイン**に関するものです。

それは、そのアーキテクチャアーキテクチャアーキテクチャ決定レコード ADR を文書化することで終わります。これは、後続のすべてのステップの青写真として機能します。

- **目標** HM の基本的なアーキテクチャを定義します。
- **主要なアクティビティ**
  - 適切な **Kubernetes フレーバー**、トポロジ、および展開場所を確認および選択します。

  - インフラストラクチャ要件と Kubernetes インストールフェーズ 2 および 3 の入力としてアーキテクチャ決定 ADR を文書化します。

- **ドキュメント** フェーズ *1* アーキテクチャの計画

顧客としてのあなたは、最終的に展開アーキテクチャを所有する必要があります。EDB のセールスエンジニアリング、プロフェッショナルサービス、サポートチーム、またはドキュメントを参照することができますが、最終的なアーキテクチャの決定はあなたのチームにあります。

4.1.1.3 フェーズ 2 システム要件の収集

重点ロールインフラストラクチャエンジニア/CSP 管理者/プラットフォームエンジニア

このフェーズでは、フェーズ 1 で定義したアーキテクチャをサポートするために必要なインフラストラクチャを**特定、準備、および検証する**ことに焦点を当てています。ここで、構成の準備を開始します。

- **目標** すべての前提条件を満たし、コア構成の入力を準備します。
- **主要なアクティビティ**
  - 選択したアーキテクチャの**ハードウェア、ネットワーキング、ストレージ、DNS 要件、およびその他のシステム要件**を特定します。
  - Kubernetes 展開のすべての前提条件が満たされていることを確認します。
  - 重要な構成エントリを設定する準備をします。 `HybridControlPlane` CR のハイブリッドマネージャー。
- **ドキュメント** フェーズ *2* システム要件の収集

EDB のセールスエンジニアリングとプロフェッショナルサービスは、セールスサイクル中または作業ステートメント SoW を介して、詳細な展開要件を伝達および明確にするための主要リソースです。公式ドキュメントには、包括的なチェックリストも提供されています。

4.1.1.4 フェーズ 3Kubernetes クラスターの展開

重点ロール Kubernetes エンジニア/CSP 管理者

このフェーズでは、以前の計画に基づいて、選択した Kubernetes 環境の展開を実行します。

- **目標**運用可能な Kubernetes クラスターを展開およびプロビジョニングします。
- **主要なアクティビティ**
  - 定義されたアーキテクチャとインフラストラクチャ要件に従って Kubernetes を展開します。
  - 必要な抽象化ノードプール、ストレージクラス、ネットワーキングを実装します。
  - インフラストラクチャ要件が満たされていることを検証し、重要な構成エントリを準備します。
- **ドキュメント** フェーズ *3Kubernetes* クラスターの展開

EDB サポートコンテキストお客様は、Kubernetes クラスターのインストールとライフサイクル操作を所有します。プロフェッショナルサービスは、作業ステートメント SoW を介して従事でき、サポートはナレッジベース記事を介して支援を提供できます。

4.1.1.5 フェーズ 4 環境の準備

重点ロールサイト信頼性エンジニア SRE/インフラストラクチャエンジニア

ここで最終的な準備が行われ、クラスターが HM を受信して実行するように最適に構成されます。

- **目標** 必要なセキュリティと構成リソースをプロビジョニングします。
- **主要なアクティビティ**
  - 重要な **シークレットと構成リソース** のプロビジョニング
  - ハイブリッドマネージャーインストールのクラスターの準備状況の詳細な検証を実行します。
- **ドキュメント** *Preparing your environment*

EDB サポートコンテキストハイブリッドマネージャーの最終準備は、クラスターのライフサイクルオペレーションと同様に、主に**顧客の責任** エンジニアドシステムを除く。**プロフェッショナルサービス**は、作業ステートメント SoW を介して従事でき、 **サポート**はナレッジベース記事を介して支援を提供できます。

4.1.1.6 フェーズ 5 ハイブリッドマネージャーのインストール

重点ロールサイト信頼性エンジニア SRE/インフラストラクチャエンジニア

これは最後の展開手順であり、準備した環境にハイブリッドマネージャーがインストールされます。

- **目標**ハイブリッドマネージャーを展開し、操作が成功したことを確認します。
- **主要なアクティビティ**
  - 構成ファイル、および環境と展開設定の最終検証。
  - ハイブリッドマネージャーの展開
  - インストールが成功し、操作の準備ができていることを確認します。
- **ドキュメント** *Installing Hybrid Manager*

4.1.1.7 フェーズ 6 インストール後のオプションの検討

重点ロール インフラストラクチャエンジニア/プラットフォーム管理者

システムが実行されると、このフェーズでは、AI ワークロード、外部モニタリング、移行支援などの高度な機能のロックを解除するために必要な「Day 2」構成をカバーします。

- **目標**プラットフォームをより広範なエコシステムと統合し、オプションモジュールを有効にします。
- **主要なアクティビティ**
  - **AI 機能:** S3 バケットと Fernet キーを接続して、GenAI Builder を有効にします。

  - **オブザーバビリティ:** 通知チャネル PagerDuty、電子メールを構成し、外部データベースモニタリング用にエージェントをインストールする.
  - **移行** 移行コパイロットを有効にして、スキーマの変換を支援します。

- **ドキュメント** *See Post-installation options*

4.1.1.8 HM の設定

いくつかの詳細な構成 SOPS は、インストール手順全体で参照され、特定の手順を参照するために役立つか、参照する必要があります。

1. *Syncing images to a local private registry*
2. *Using the in-cluster container registry*
3. *Configuring image discovery*
4. *Customizing secrets*
5. *Implementing TLS certificates*
6. Deploying HM across multiple data centers
7. *Configuring IdPs*
8. Enabling KMS for TDE
9. *Optimizing service mesh memory*
10. *Migrating from the bootstrap to operator method*

4.1.1.8.1 Planning your architecture

概要

ロール CTO / アーキテクト / リードエンジニア

前提条件

- **アーキテクチャプランで有効にするビジネス目標のリスト例バグデコンストレイン、目的のアップタイム、および目的のレイテンシー***)*

結果

- **ハイブリッドマネージャー** *HM* **アーキテクチャのトポロジ、局所性、および冗長モデルを定義するアーキテクチャ決定レコード** *ADR*。*(*少なくとも、ノート付きのアーキテクチャ図*)*
- *HM* **構成の初期入力 — ``***HybridControlPlane***``カスタムリソースマニフェスト。**

最終的に展開アーキテクチャを所有するのは顧客であるあなたです。EDB のセールスエンジニアリング、プロフェッショナルサービス、サポートチーム、またはドキュメントを参照することができますが、最終的なアーキテクチャの決定はあなたのチームにあります。

アーキテクチャの発見

アーキテクチャ検出の目標は、ハイブリッドマネージャー HM を正常に展開するために必要な決定をナビゲートおよび文書化することです。これらの決定は、 ** フェーズ *2* システム要件の収集 ** フェーズ 2 および ** *Preparing your environment* ** フェーズ 3 のブループリントを形成します。

付属の質問は、データベースレイヤーを超えて広がる幅広い考慮事項をカバーしています。このガイドは、2 つの観点から見る必要があります。

- **現在の状態**現在、既存のデータベースとアプリケーションのワークロードはどこにありますか？
- **目標の状態** HM をすぐにどこに展開する予定ですか、そして次の 1〜2 年でどこに拡張する予定ですか。

  推奨事項現在およびターゲット状態の図を取得して確認することは、このフェーズを完了するための最も効率的な方法です。

地域 HM はどこに住んでいますか？

データベースと依存アプリケーションの物理的または論理的な場所を理解することは、必要なアーキテクチャを決定するために重要です。

- **回答する質問**
  - 現在のデータベースソリューションは、 **クラウドリージョン CSP** または **物理データセンターオンプレミス**の観点からどこにありますか?
  - 依存アプリケーションのワークロードはどこにありますか
  - **依存関係の上流レイヤー** はありますか? それらはどこにありますか?
- **分析**
  - ローカリティは、展開の初期スコープたとえば、単一のクラウドリージョン対マルチリージョンを決定します。
  - 複数のリージョン、クラウド、またはハイブリッドクラウド環境にまたがる予定がある場合、**Postgres Distributed** は、適切なデータベースサービスの推奨事項である可能性があります。
  - 上流アプリケーションの局所性が、ネットワーク遅延を最小限に抑えるためのキーとなります。

ディザスターリカバリーホット/コールド

ディザスターリカバリー DR は、さまざまな場所でのビジネスの継続性を保証します。

- **回答する質問**
  - ビジネス継続のサブセットとしてのディザスターリカバリーはこれらの場所でどのように実行されますか、または特にディザスターリカバリーとして割り当てられた**追加のロケーション** はありますか?
  - **追加情報はありますか場所は特に DR** として割り当てられました?
  - DR 機能はどのようにして検証されますか**、どのくらいの頻度でしょうか。
- **分析**

  - 専用のセカンダリロケーションがあることは、強力なアーキテクチャ要件を示しています。
  - 正式な DR 実践が存在しない場合、HM DBaaS ファーウェイレプリカソリューションは新しい機能を提供する可能性があります。

アクティブネス アクティブ/パッシブ vs. アクティブ/アクティブ

アクティブ度は、重要なワークロードに分散ロケーションがどのように利用されているかを表します。

- **回答する質問**
  - 複数のロケーションがある場合、重要な依存ワークロードはこれらのシステムをどのように利用しますか?
  - トランザクション処理 OLTP に対して 1 つのロケーションがアクティブで、もう 1 つがパッシブですか?
  - OLTP に対して 1 つのロケーションがアクティブですか、そしてもう 1 つは分析処理 OLAP / BI のためにアクティブですか？
- **分析**
  - ターゲット状態が複数のデータベースインスタンスへの同時書き込みが必要な場合、場所全体で真の**アクティブ/アクティブ** 、そのマルチプルのため、 **Postgres Distributed** が必要なソリューションです。ライター機能..
  - 場所が受動的に待機しているコールドスタンバイか、アクティブに実行しているホットスタンバイかを理解することは、リソース要件と目標復旧時間 RTO を定義するのに役立ちます。
  - **ビジネス継続性**アクティブ/パッシブを中心としたアーキテクチャの選択、アクティブ/アクティブ、およびスタンバイモデルは、冗長システムの維持のコストと、ダウンタイム/データ損失に対する組織の許容度のバランスを取る必要があります。

これらのトピックは、アクティブ性の議論に自然に続き、アプリケーションエコシステムの全体像を完成させるのに役立ちます。

- 消費アプリケーションの観点からの入力トラフィックルーティング。
- さまざまなアプリケーションレイヤーでのレプリケーション。
- キャッシュレイヤーとデータベースに対するその相対的な場所。
- セッション要求たとえば、セッションレプリケーションはアプリケーションレイヤーで処理されますか？

ライフサイクル操作

運用慣行を理解することは、データベースサービスの管理に必要な Kubernetes 環境の複雑さを判断するのに役立ちます。

- **回答する質問**
  - **Blue/Green** または **Canary** などのライフサイクル操作パターンを利用していますか?
  - **DML/ DDL 更新** データとスキーマをどのように処理しますか) vs. **エンジンのアップグレード** メジャーバージョン?

  - どのような **プリプロダクション環境**ステージング、開発、テストが必要ですか。

- **分析**

  - ブルー/グリーン展開のようなプラクティスは、EDB のデータベースソリューションが提供するゼロダウンタイム機能とよく連携します。

  - プリプロダクション環境の数は、** フェーズ *2* システム要件の収集 で定義された合計クラスター数とリソースサイズに直接影響します。** 。

サポートされているプラットフォーム

HM と Kubernetes は 1 対 1 の関係にあります。各 HM 展開には、独自の専用の Kubernetes クラスターが必要です。Kubernetes クラスターは、現在のバージョンで HM 専用である必要があります。他のワークロードとの共有はサポートされていません。

- **Amazon EKS** Elastic Kubernetes Service
- **Microsoft Azure AKS** Azure Kubernetes Service
- **Google GKE** Google Kubernetes Engine
- **ROSA** AWS 上の Red Hat OpenShift Service
- **RHOS** Red Hat OpenShift Container Platform
- **Rancher RKE** Rancher Kubernetes Engine / RKE2

お客様は、Kubernetes クラスターの完全なライフサイクル管理、プロビジョニング、展開、アップグレード、スケーリングに対して責任を負います。

HM 分散リファレンスアーキテクチャは、最高レベルのスケールと最速の SLA を達成するための最終目標を表しています。通常、複数のデータセンターにまたがります。

図の凡例リファレンス

凡例は、アーキテクチャ図で使用される色と論理グループを定義します。

- **ローカリティ** 物理データセンターまたは地理的地域たとえば、“City 1”や”Data Center 1”などの最高レベルの物理的または論理グループ。
- **Kubernetes クラスター**プラットフォーム全体をホストする完全な Kubernetes 環境すべての CP およびワーカーノードを含む。
- **EDB HM:** コア HM コンポーネントの論理境界。これは通常、専用の Kubernetes 名前空間たとえば、コントロールプレーンとして実装されます。
- **コンピューティングマシン** Kubernetes ワーカーノードとして機能し、クラスターに CPU、メモリ、およびストレージを提供する仮想マシンたとえば vm01、vm02、vm03。
- **インフラストラクチャの抽象化**この重要なレイヤーは、基になる物理または仮想インフラストラクチャを抽象化する Kubernetes ネイティブリソースを表します。これらのリソースは、Kubernetes クラスターの環境によって提供される必要があります。

  - **例 1 type LoadBalancer** これは、外部ロードバランサーを要求する Kubernetes Service タイプです。パブリッククラウド環境 AWS、GCP、Azure などでは、これは管理対象サービスとして自

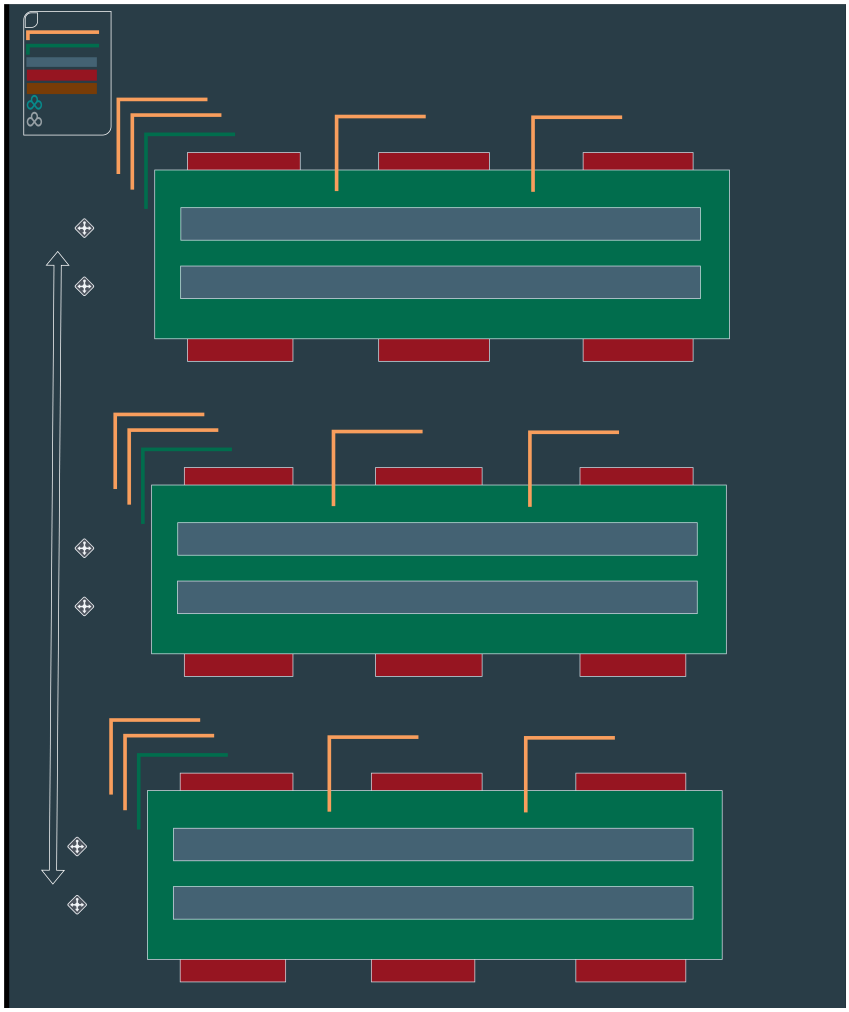


図 1: HM reference architecture

動的にプロビジョニングされます。オンプレミスまたはベアメタル展開では、これらのロードバランサー要求を満たすための MetalLB のようなソリューションを提供する必要があります。

- **例 2 StorageClass** このリソースは、「ブロックストレージ」および「オブジェクトストレージ」要件を抽象化します。Kubernetes ストレージ要求永続ボリューム要求を実際のプロビジョニングされたストレージハードウェアまたはソフトウェア local-pv、Ceph、vSphere、またはクラウドベースのディスクなどにマッピングします。

HM コントロールプレーン vs. データプレーン

Attribute	Control Plane (CP)	Data Plane (DP)
Role	Orchestration, configuration, and observability	Postgres cluster hosting and replication
Protocol	HTTP (REST / gRPC) via Istio service mesh	Postgres wire protocol (TCP) per PG cluster
Multi-location topology & traffic	Star — Primary is mandatory hub across locations; Primary pushes Control outbound to each Secondary, Secondaries return Logs to Primary	Optional mesh — locations may be fully isolated or directly peered for Postgres replication; no hub required
Lifecycle isolation	All CPs share management context; CP upgrades affect the full estate	Each DP is an independent failure, security, and upgrade domain
DR role	Secondary CPs provide management continuity	DPs are the replication targets; hot/cold standby model determined per deployment
Scenarios	core, migration, ai	dbaas, analytics, ai

プロトコルとイングレス

2 つのプレーンを区別する最も簡単な方法は、プロトコルによるものです。

コントロールプレーンは完全に EDB 製品であり、その内のすべてのコンポーネントは HTTP を話します。その HTTP トラフィックは、**Istio サービスメッシュ**によって提供および保護され、すべての CP サービスにわたって単一の集中化された入力サーフェスを提供します。認証、TLS、およびトラフィックポリシーはすべて、要求が CP コンポーネントに到達する前に Istio レイヤーで適用されます。

データプレーンは顧客プロビジョニングの Postgres であり、その内のすべてのクラスターは **Postgres ワイヤプロトコル PSQL** を話します。パスにはミドルウェアがありません。各 Postgres クラスターは、独自の専用 TCP 接続でリッスンします。これらの接続を集約する共有プロキシまたはメッシュはありません。これは意図的なデザインの選択です。クラスターごとの専用イングレスにより、顧客のワークロード間の可用性とセキュリティ分離境界の両方が最大化されます。

要するに **HTTP CP 入力は Istio を介して一元化されます。 PSQL DP ingress は、Postgres クラスターごとに専用です。**

ライフサイクルの分離

ライフサイクルの分離は、2 つのプレーン間の運用上で最も重要な違いです。

CP とそれが管理するすべての DP は、**独立したアップグレードライフサイクル**上にあります。私たちは、長期サポート LTS リリースのパッチを月次のリズムで公開しています。お客様はこれらを速やかに適用することを強くお勧めします。アーキテクチャによりこれが現実的です。CP にはパッチが適用されています。DP で Postgres を管理する組み込み CloudNativePG CNPG オペレーターが**インプレースインスタンスマネージャー更新**用に構成されているため、この分離は保持されます。オペレーターがアップグレードされると、各 Postgres ポッド内のインスタンスマネージャーバイナリがスワップされますが、実行中のポストマスターがトラフィックの処理を継続します。ポッドの再起動、スイッチオーバー、アプリケーション可視イベントはありません。

これらのマンスリー CP パッチには、2 つのカテゴリのコンテンツが含まれます。

1. **CP コンポーネントセキュリティパッチ** —オーケストレーションレイヤー自分自身を強化します。
2. **新しい Postgres コンテナイメージ** —Postgres エンジンバージョン、拡張機能、およびコンテナホスト OS の更新されたイメージ、すべてに現在のセキュリティパッチが適用されます。

CP で新しいイメージが利用できるようになったら、CP のオーケストレーション機能を使用してこれらのイメージを DP Postgres 展開にロールアウトするかは **お客様の決定**です。これにより、Postgres のアップグレードタイミングを顧客が完全に制御できながら、強化されたイメージを常にオンデマンドで利用できるようになります。

ルールの例外

CNPG オペレーターリリースのごく一部 — 歴史的に 10%未満 —アンチアフィニティルール、コンテナプローブ、または init コンテナの変更など、Postgres **ポッド仕様自分自身**を変更します。これらのリリースでは、Kubernetes は新しい仕様に一致するようにポッドを再作成する必要があり、インプレースインスタンスマネージャーの更新が有効になっている場合でも、CP アップグレードの一部として HA Postgres クラスターのローリングリスタートは避けられません。HM リリースにこれらの CNPG バージョンのいずれかが埋め込まれている場合、**EDB は HM アップグレードドキュメントおよび影響を受ける顧客へのアップグレード前ガイダンスでこれを明示的に呼び掛けています。これは、特定の HM アップグレードが完全に非対応かどうかを顧客が事前に知ることを意味します。破壊的、またはローリング Postgres の再起動がその一部として発生するかどうか。実際に必要な場合にのみメンテナンスウィンドウを計画できます。

可用性とディザスターリカバリー

CP の可用性と DP の可用性は独立した関心事であり、2 つのプレーンは異なる復旧戦略を保証します。

コントロールプレーン

CP は、顧客の Postgres トラフィックのデータパスにありません。CP の停止は、プロビジョニングされた Postgres クラスターへの接続を中断しません。実行中のワークロードは影響を受けずに継続されます。CP は、堅牢な DR 機能を備えています。規律ある運用実践定期的なバックアップ、文書化されたランブックにより、CP は **1 時間未満** で復元できます。

データプレーン

DP はライブ Postgres トラフィックを伝送するため、高可用性ターゲットには**障害ドメイン**全体、最適にはリージョン全体でアクティブなレプリケーションが必要です。DP の場所ネットワークピアリングがプロビ

ジョニングされる場所を超えた Postgres レプリケーションにより、DP はアプリケーションの接続に影響を与えることなく、リージョンに障害が発生しても生き残ることができます。

アベイラビリティーゾーン

どちらのプレーンも、 *ロケーション内の**アベイラビリティーゾーン**を最大限に利用できます。すべての HM 展開場所には、CP 要素、DP 要素、またはその両方が含まれる場合があります。バックアップを使用して、任意の場所をディザスターから復旧できます。違いは天井です。CP のリカバリーは数分〜1 時間で測定される復元オペレーションですが、DP の Postgres クラスターは、他の障害ドメインにライブレプリカを維持することにより、災害中に継続的に可用性を維持するアーキテクチャ能力を備えています。

デプロイメントアーキテクチャ

上記のこの凡例は、以下の展開アーキテクチャにも適用されます。

最小インストールでは、Kubernetes 制御ノードに HM コントロールプレーン CP をコロケーションします。

これは次の場合に完全に機能します。

- Postgres / Oracle エステートのビューの一元化。
- データベース移行機能。
- GenAI 管理対象 Postgres インスタンスの不足による機能の制限。

内部アーキテクチャ HM コントロールプレーン

HM は、Kubernetes クラスター内で実行される複数のコアマイクロサービスで構成されています。これらのコンポーネントを理解することは、リソースの割り当てとセキュリティ境界を計画する際に役立ちます。

- **GenAI** AI / ML 機能を提供します。このコンポーネントは有効になっている場合、** **フェーズ** *2* **システム要件の収集** ** での GPU 対応ワーカーノードの必要性を指示します。 - **参照** *GenAI in HM*
- **Postgres ライフサイクル操作**データベースの展開、スケーリング、更新を管理するオーケストレーションエンジン。- **参照** *Cluster Management*
- **テレメトリ**メトリックとログを収集します。このサービスでは、状態を報告するためにアウトバウンドネットワークアクセスが必要です。- **参照** *Monitoring with HM*
- **データベース移行アシスタント**外部ソースからプラットフォームへのデータの移動を促進します。 - **参照** *Migrating databases with HM*
- **エステート** HM DBAAS 内部システムと外部データベースを使用して作成するリソースのインベントリを管理します。- **参照** *Enable monitoring>On external database clusters*
- **フェデレーション**マルチロケーショントポロジ内の複数の HM インスタンス全体で安全な通信と承認を管理します。- **参照** *Configuring multiple data centers for HM*

アーキテクチャの依存関係

上記のアーキテクチャ図は、いくつかの外部コンポーネントを参照しています。** **フェーズ** *2* **システム要件の収集** ** でこれらの特定のハードウェア/ソフトウェア要件を確認している間、アーキテクチャデザインでの接続を考慮する必要があります。

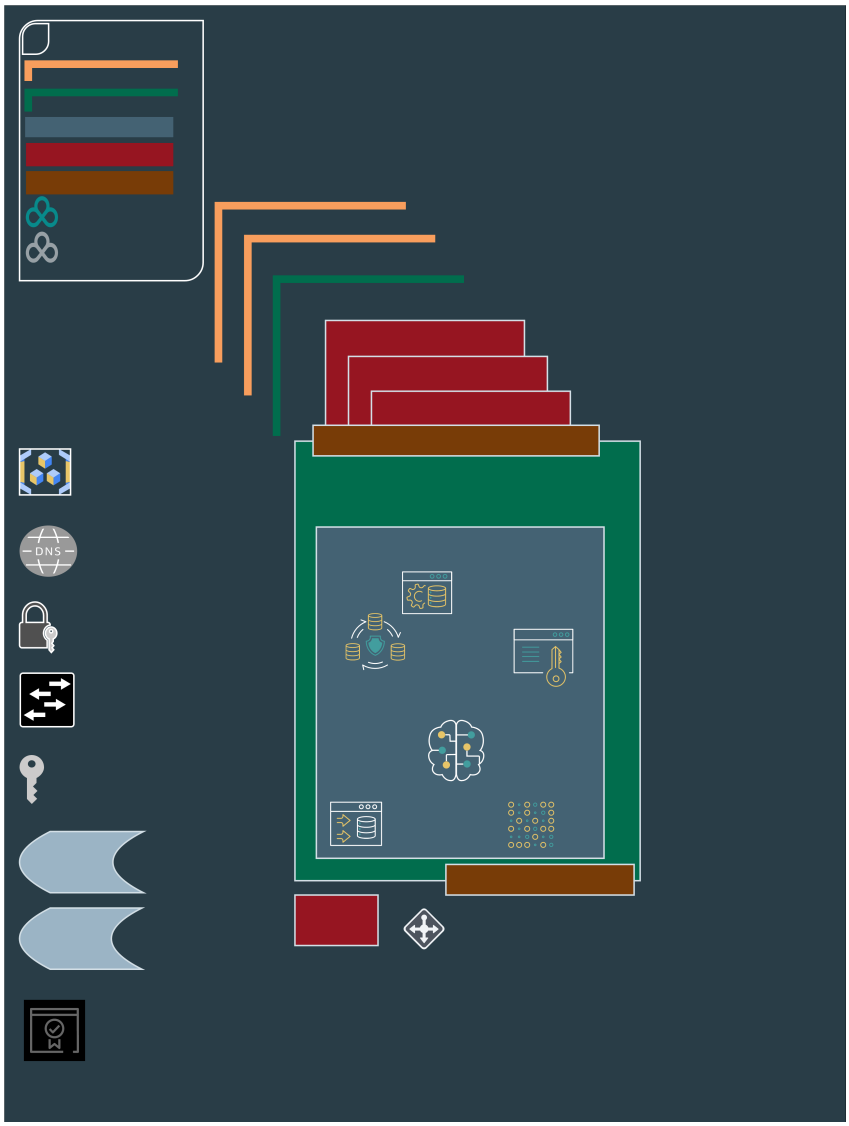


図 2: HM minimum

- **ID プロバイダー IdP** ユーザー認証に必要です。このアーキテクチャは、すべての人によるアクセスについて OIDC LDAP / SAML に依存しています。
- **キー管理サービス KMS** オプショナルセキュリティポリシーで透過的データ暗号化 TDE が要求される場合にのみ必要。
- **オブジェクトストレージ**システムの復元力に必要です。バックアップ、ログをホストし、マルチロケーショントポロジのデータレプリケーションを促進します。
- **ブロックストレージ**データベースのパフォーマンスに必要です。ストレージアーキテクチャは、Postgres データレイヤーの永続ボリューム PVC を提供する必要があります。
- **ローカルネットワーク** CP をデータプレーンに接続するファブリック。ここでのレイテンシーは、ローカリティの決定を促進します。
- **Container Registry** アプリケーションイメージの信頼できるソース。エアギャップデザインの場合、これはローカル同期レジストリを表します。

HM コントロールプレーン+データプレーン

HM CP の横にあるのは、HM データプレーン DP です。これは、実際のデータベースワークロードが存在する場所です。

- Postgres クラスター実際のデータベースインスタンスプライマリおよびスタンバイ。
- 拡張機能 PostGIS、PGVector、およびその他のデータベース拡張機能。
- バックアップエージェントオブジェクトストレージへの WAL アーカイブを管理するローカルツール Barman のような。

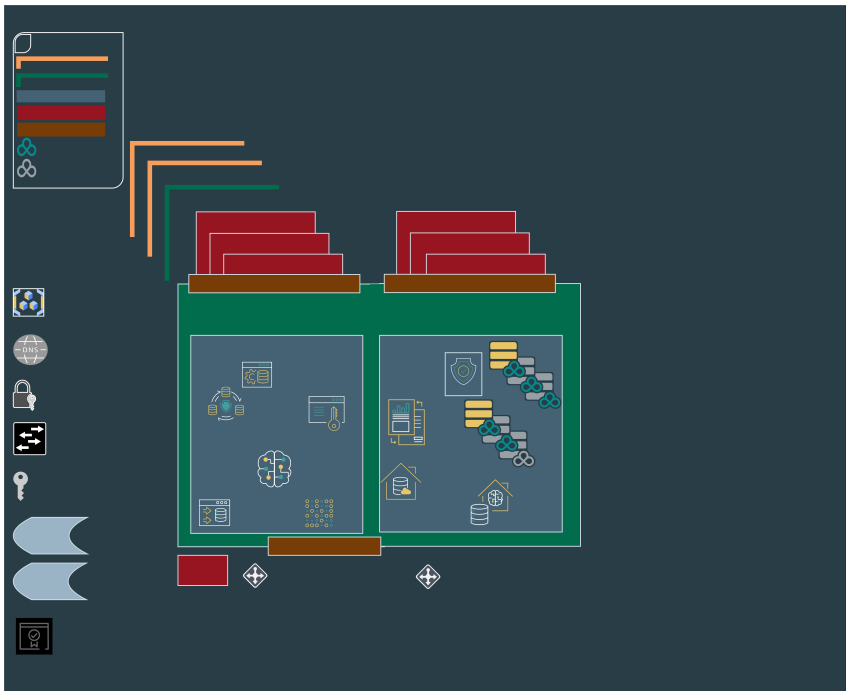


図 3: HM Data Plane

HM のフル機能の展開

このビューには、AI ワークロードの GPU アクセラレーションなどのリソースを含む、完全な機能の HM 展開が示されています。

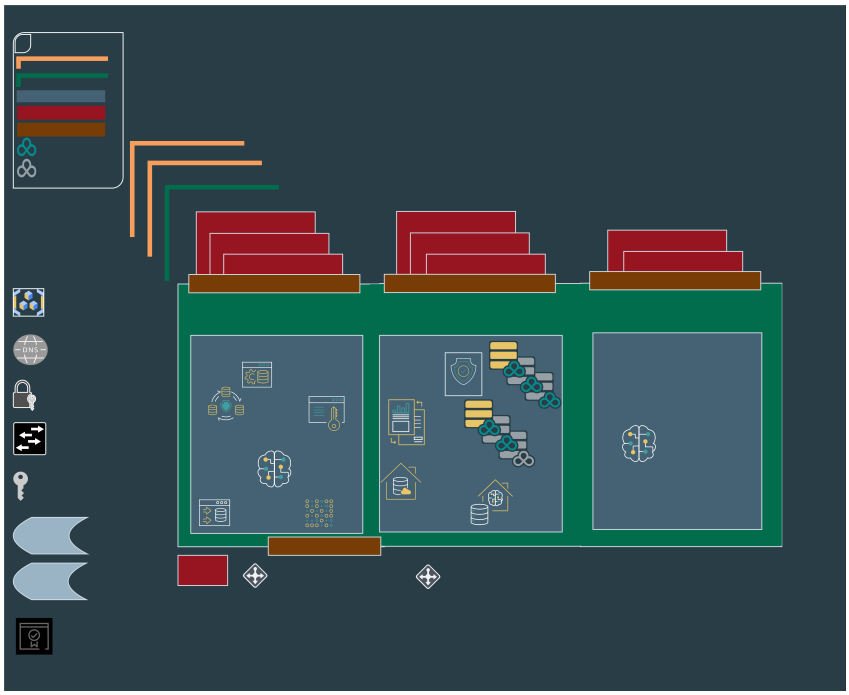


図 4: HM fully featured

EDB Hybrid Manager のマルチロケーショントポロジ

EDB ハイブリッドマネージャーは、2 つの異なるただし相補的なネットワークトポロジ-1 つはコントロールプレーン用、もう 1 つはデータプレーン用を介してマルチロケーション展開をサポートします。各トポロジは、そのプレーンのロールとプロトコルを反映します。

マルチロケーション機能は、ハブアンドスポークモデルに従った DBaaS オファリングです。

1. DBaaS オファリングとして、セカンダリ HM の機能セットは、プライマリと比べて低下します。
2. プライマリ HM はセカンダリを制御します。
3. 接続は、負荷分散されたエンドポイントを使用して確立されます。Submariner のようなネットワークメッシュサービスではありません。

コントロールプレーン スタートポロジ

コントロールプレーンは **スタートポロジ**で配置されます。指定された 1 つのプライマリ CP が中心にあり、最大 5 つのセカンダリ CP がそこから外側に放射状にあります。

トラフィックは厳密な方向パターンに従います。**制御フローはプライマリから各セカンダリへの外向き** - 構成、オーケストレーションコマンド、およびライフサイクル操作はすべてハブから発生します。**ログは**

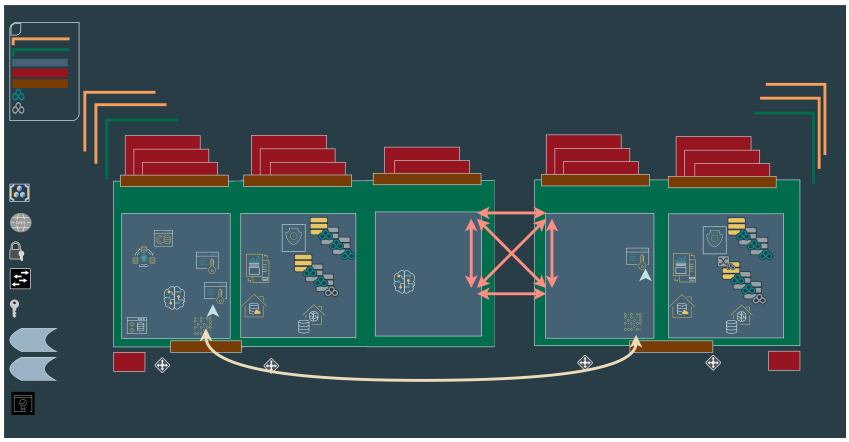


図 5: HM multi-location

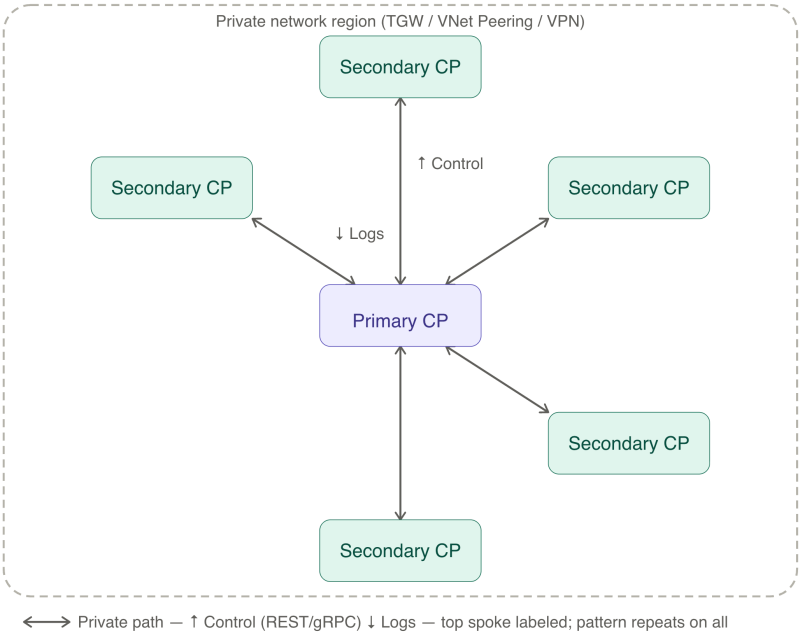


図 6: CP star topology

内向き、各セカンダリからプライマリに戻ります。セカンダリには横方向の可視性がありません。相互に制御できず、自己統治もできません。すべての権限はプライマリに集中されています。

このハブアンドスポーク権限モデルは重要な結果をもたらします。各セカンダリ CP はプライマリによって管理されますが、ホストするワークロードの点では自律しているため、セカンダリはまったく異なるエンゲージメントを提供できます。一般的な例は次のとおりです。

- **開発/テストと運用** —個別のセカンダリは、インフラストラクチャレベルで環境の分離を適用します
- **ある顧客と別の顧客** —ワークロードの分離が契約またはセキュリティ要件であるマルチテナントまたは MSP 展開
- **オンプレミスとクラウド** —単一のプライマリは、1 つ以上の CSP に展開された CP と並んで、プライベートデータセンターに展開された CP を管理できます。
- **地理的分散リージョン** - 各セカンダリは、単一の操作重心から管理される別の地理的リージョンに存在できます。スターは**プライベートネットワークパス**で排他的に接続されます。VPC / VNet を介してアクセスされる内部ロードバランサーエンドポイントピアリング、トランジットゲートウェイ、VPN、または同等のサービス。パブリックエンドポイントを介して到達可能なセカンダリはありません。

データプレーン オプションのメッシュトポロジ

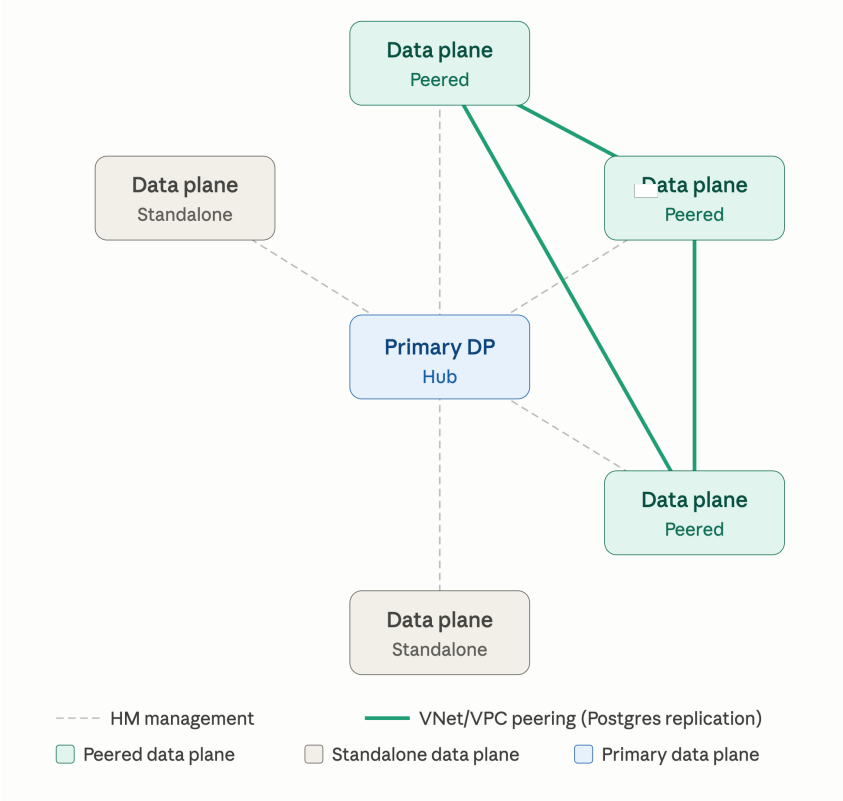


図 7: DP optional mesh topology

データプレーントポロジは **オプショナルメッシュ**です。2 つのデータプレーンの場所を完全に分離したままにするか、Postgres レベルのレプリケーションの VPC / VNet ピアリングを介して直接相互接続できます。ハブ、必須センター、および場所間の必要な接続はありません。

2 つのデータプレーンがピアリングされている場合、ネットワークパスは、各場所でホストされるクラスター間で**直接 Postgres レプリケーション**を有効にします。これは、EDB がサポートする 2 つの分散 Postgres トポロジの基礎です。

- **EDB Postgres 分散 PGD** - ロケーション全体のアクティブ/アクティブマルチライターレプリケーション。リージョン全体でゼロ RPO およびほぼゼロの RTO を必要とするアプリケーションに適しています。PGD documentation を参照してください。
- **CloudNativePG 分散** —CloudNativePG の分散トポロジを使用して、Kubernetes クラスターと障害ドメイン全体でストリーミングレプリケーション。CloudNativePG distributed documentation を参照してください。ピアリングされていないデータプレーンは、完全に独立したままです。別のセキュリティドメイン、別の障害ドメイン、別のアップグレードタイムライン。ピアリングは追加的です。他の目的で各プレーンの分離特性を変更せずに、レプリケーションパスを導入します。

メッシュのオプショナルの性質は、データプレーントポロジが、HM 自分自身からのアーキテクチャ指示によってではなく、顧客の**レプリケーションと可用性要件**によって完全に形成されることを意味します。

---

2 つのトポロジの関係

CP スターと DP メッシュは、同じセットの物理的またはクラウドの場所上の独立したオーバーレイです。ロケーションは、セカンダリ CP、データプレーン、またはその両方をホストできます。CP スターは、展開されるものとその管理方法を制御します。DP メッシュは、Postgres データが場所間を移動する方法を制御します。どちらのトポロジも他方をコンストレインしません。

	Control Plane	Data Plane
Shape	Star — mandatory hub	Optional mesh — peer-to-peer
Hub required	Yes — Primary CP	No
Cross-node traffic	Control out, Logs in	Postgres replication (when peered)
Isolation model	Secondaries are independent engagement boundaries	Each DP is an independent failure and security domain
Network requirement	Private path to Primary required for all Secondaries	Private path only between peered DPs

---

インストールシナリオの選択

HM は、デフォルトで包括的な機能を提供します。厳格なセキュリティ基準と組織のガバナンスを満たすために、HM は「シナリオ」を介したモジュールインストールをサポートしています。これにより、厳選された機能のサブセットを展開でき、セキュリティ監査のためのソフトウェアフットプリントを効果的に削減し、未承認またはライセンスのないコンポーネントを削除することにより UI を合理化できます。

!!!note

シナリオは、高度な運用展開計画を対象としています。パイロット、概念実証 PoC、および初期評価の場合、完全なインストールをして、最大限のオプションを維持し、すべての統合機能がテストに利用できるようにすることをお勧めします。この場合、`scenarios` 構成パラメーターを省略します。これにより、使用可能なすべてのシナリオがデフォルトで有効になります。

使用可能な 4 つのシナリオは次のとおりです。

Module	Description	Included capabilities
core	**必須** 。すべての展開の基礎レイヤー。	Estate management, Observability, DBaaS provisioning, and system services (Cert-manager, Istio, etc.).
migration	スキーマおよびデータ移行ツールを有効にします。	Migration Portal integration, Data Migration Service (DMS), and other migration services.
analytics	大規模なデータの処理とカタログ化。	Data Catalog.
ai	生成 AI を構築および提供するためのツール。	Sovereign AI, model serving (kserve), Langflow, and GenAI builders.
dbaas	データベースクラスター管理。	Postgres cluster management, including provisioning, scaling, and updates.

2026. 4.0 以降、 `dbaas` 、データベースクラスター管理機能は、 `core` シナリオから分離された専用の `dbaas` シナリオによって制御されるようになりました。2026.3 からアップグレードする既存のインストールは、Postgrescluster 管理機能を保持するために `dbaas` をシナリオリストに明示的に追加する必要があります。

インストールするシナリオを選択するときは、次のアーキテクチャおよび操作上の要素を検討します。

ファンクションフットプリントベースラインリソース使用量はシナリオ間で同様ですが、各シナリオは特定のサービスとエンドポイントを紹介します。インストールを必要なシナリオに制限すると、セキュリティ監査が簡素化され、運用環境の「攻撃面」が削減されます。

UI と機能のガバナンス HM コンソールは、無効なシナリオのナビゲーションリンクとツール GenAI Builder や Data Catalog などを動的に非表示にします。これにより、ユーザーは許可されライセンスされた機能とのみ対話できます。

デフォルト動作 `scenarios` パラメーターが構成から省略されている場合、システムはデフォルトですべてのシナリオをインストールして、下位互換性を維持します。

環境に完全なインストールが必要か、ターゲットを絞ったサブセットが必要かどうかを決定します。

!!!重要

現在、展開後のインストールシナリオの変更はサポートされていません。HM が展開されると、それらの機能を追加または削除することはできないため、必要な機能をすべてインストールしてください。

この検出プロセス中に行われる決定は、インストール構成のルートパラメーターの一部を直接決定します。

ファイルをまだ作成する必要はありませんが、**アーキテクチャ決定レコード**でこれらのキーの値を指定する必要があります。 **SRE/Admin** は、これらの入力を記録するか、 **フェーズ** *2* **システム要件の収集** の構成ファイルを開始/続行することにより、これらの入力に基づいてビルドし、および/またはこれらの値を使用して** **フェーズ** *4* **環境の準備** ** に構成ファイルをビルドします。

構成に必要な値

フェーズ 1〜2 で収集した情報とフィールド値は、 `edb-hcp-operator` Helm チャートで使用される `HybridControlPlane` カスタムリソースマニフェストに記録されます。

構成の詳細

Architecture decision	HybridControlPlane CR field	Example value
Kubernetes Platform	*spec.flavour*	*aks*, *eks*, *gke*, *rhos*
Target location	*spec.componentsParameters.upm-beacon.beacon_location_id*	*aws-us-east-1*
Provisioning mode	*spec.beaconAgent.provisioning.provider*	*azure*, *aws*, or *gcp*
Installation scenarios	*spec.scenarios* (YAML list)	*core*, *migration*, *ai*, *analytics*, *dbaas*

構成ファイルへの影響

次の YAML スニペットは、上記のアーキテクチャの決定が最終的な `HybridControlPlane` CR 構造にどのようにマッピングされるかを示しています。

```yaml
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  flavour: <Kubernetes_Flavor> # for example, rhos, rke2, aks, eks, gke
  imageRegistry: <Container_Registry_Domain>/pgai-platform
  version: <Version>
  scenarios: # Omit to install all scenarios by default
    - core
    - migration # remove if not needed
    - ai        # remove if not needed
    - analytics # remove if not needed
    - dbaas     # remove if not needed
  componentsParameters:
    upm-beacon:
      beacon_location_id: <Deployment_Location_Name>
  beaconAgent:
    provisioning:
      provider: <Provider_Name> # Azure, AWS, GCP
      openshift: <Boolean_Value> # Defaults to false, set to true if deploying on RHOS
```

次のフェーズ

アーキテクチャが定義され、理想的には参照用に ADR に記録されます。

** *Phase 2: Gathering your system requirements* → に進んで、インフラストラクチャが ADR のデザインと一致できることを確認します。**

4.1.1.8.2 Gathering your system requirements

概要

重点ロールインフラストラクチャエンジニア/CSP 管理者/プラットフォームエンジニア

前提条件

- **フェーズ *1* アーキテクチャの計画** 完成品

結果

** フェーズ 3 で使用するコンピューティング、ネットワーク、およびストレージインフラストラクチャの検証された高レベルのインベントリ**フェーズ *3*Kubernetes クラスターの展開** *

** フェーズ 4 で使用する準備ができているとして検証された多数の要件 *Preparing your environment* と HM 構成の初期入力-`HybridControlPlane` カスタムリソースマニフェスト。*

> **注釈**
>
> EDB のセールスエンジニアリングとプロフェッショナルサービスは、セールスサイクル中または作業ステートメント SoW を介して、詳細な展開要件を伝達および明確にするための主要なリソースです。公式ドキュメントには、包括的なチェックリストも提供されています。 !!!

次のフェーズ フェーズ 3 **フェーズ *3*Kubernetes クラスターの展開**

アーキテクチャへの接続

フェーズ 2 のために以下にリストされている要件は、すべてが汎用的なものではありません。多くは、**フェーズ *1* アーキテクチャの計画** で行われた決定の直接の結果です。

*** ローカリティの決定**により、コントロールプレーンとデータプレーン間のレイテンシー要件が決まります。

*** ディザスターリカバリー目標**は、特定のオブジェクトストレージ構成レプリケーション用の必要性を判断します。

*** アクティブ度アクティブ/アクティブ**は、 Postgres Distributed に必要な特殊なネットワーキングが必要かどうかを判断します。

> **重要**

このドキュメントは包括的な技術チェックリストとして機能しますが、 **EDB Professional Services** は、複雑な展開アーキテクチャを検証するための主要なリソースです。フェーズ 1 の「ターゲット状態」にマルチリージョンの冗長性または高スケールの要件が含まれる場合、インフラストラクチャを調達する前に作業ステートメント SoW を介してこれらの詳細な要件を明確にすることをお勧めします。 !!!

導入準備状況チェックリスト

このチェックリストを使用して、必要なインフラストラクチャコンポーネントをすべて利用できることを高レベルで確認します。

- 1. :ref:` <#1-management-workstation-bastion-host>`
- 2. :ref:` <EDB Platform>`
- 3. :ref:` <#3-compute-node-requirements>`
- 3.1 *3.1* コントロールプレーンノード
- 3.2 *3.2* データプレーンノード
- 3.3 *3.3 AI* モデルノード
- 4. :ref:` <#4-local-networking>`
- 5. :ref:` <#5-ingress>`
- 5.1 *5.1* ロードバランサーコントローラー
- 5.2 Domain Name Service
- 5.3 Portal certificate management/TLS
- 6. :ref:` <EDB Database>`
- 6.1 *6.1* ボリュームのスナップショット
- 6.2 *6.2* ハイブリッドマネージャー検出からのストレージの除外
- 7. :ref:` <#7-object-storage>`
- 8. :ref:` <#8-container-registry>`
- 9. :ref:` <#9-identity-provider-idp>`
- 10. :ref:` <#10-key-management-service-kms>`

1. 管理ワークステーション Bastion ホスト

オプショナル 厳密な要件。

展開をオーケストレーションするには、指定されたマシンが必要です。これは、ラップトップパブリッククラスターの場合またはクラウドベースの Bastion ホストプライベートクラスターの場合です。

> **注釈**
>
> Kubernetes API、ポータル、および PostgreSQL エンドポイントへの安全なプライベートアクセスには、**Bastion host**またはジャンプホスト/ボックスを強くお勧めします。このホストは、インストールと検証アクティビティのための専用の操作ワークステーションとして機能します。 EDB リモート DBA またはマネージドサービス契約の場合、専用の Bastion は**必須**です。 !!!

一般的な要件

*** オペレーティングシステム **Linux AMD64/ARM64 または macOS。**注:*Windows ユーザーは **WSL2** Linux 用 Windows サブシステムを使用する必要があります。

*** ネットワークアクセス ***** Kubernetes API: **クラスター API サーバー通常はポート 6443 へのネットワーク接続が必要です。*** インターネット/レジストリ** Container Registry に到達して、Helm チャートとイメージをプルできる必要があります。

*** ストレージ** 最小限構成ファイルと証明書を保持するのに十分な。

必要な工具在庫

この段階ではこれらのツールをインストールする必要はありません。ただし、ワークステーション環境で、次のフェーズフェーズ 3 で次のバイナリのインストールと実行が許可されていることを確認する必要があります。

コアインストールツール

- `edbctl` EDB ハイブリッドマネージャー CLI
- `kubectl` Kubernetes CLI + `cnpg` プラグイン
- `helm` Kubernetes パッケージマネージャー
- `yq` コマンドライン YAML プロセッサー

ユーティリティ

- `curl` ダウンロードユーティリティ
- `openssl` 証明書管理
- `htpasswd` 静的ユーザーを使用する場合にのみ必要

プラットフォーム CLI 環境に依存します

- `aws` AWS CLI
- `gcloud` Google Cloud CLI
- `oc` OpenShift CLI

操作ツールインストール後

- `pgdcli` Postgres Distributed 管理
- `sentinel` モニタリングとフェイルオーバー管理

2. Kubernetes プラットフォームの検証

プロビジョニングするクラスターが ** フェーズ *1* アーキテクチャの計画 ** フェーズで選択したプラットフォームと一致することを確認します。

サポートされているディストリビューションクラウドサービスプロバイダープラットフォーム Microsoft Azure AKS IR リリースストリームのみ *Amazon EKS* Google GKE * ROSA RedHat OpenShift Service on AWS IR リリースストリームのみ

オンプレミスプラットフォーム SUSE Rancher RKE2 * OCP RedHat OpenShift RHOS

LTS / IR バージョンごとの HM 互換性の詳細については、

Hybrid Manager platform compatibility を参照してください。

プロビジョニング制約

*** 専用クラスター**クラスターはハイブリッドマネージャー専用である必要があります。他のワークロードとのマルチテナントは**サポートされていません**。

*** 関係** 1:1 1 つのクラスターごとに 1 つの HM 展開。

*** ライフサイクル** Kubernetes レイヤーのプロビジョニング、アップグレード、およびスケーリングは自分の責任です。

構成の依存関係

プラットフォームの選択により、特定の構成値が決まります。

次のフェーズに向けてこれらを記録します。

Parameter	Custom Resource key	Required Value
System flavor	*spec.flavour*	*eks*, *gke*, *aks*, *rke2*, or *rhos*
OpenShift	*spec.beaconAgent.provisioning.openshift*	Set to *true* **only** if using Red Hat OpenShift.
OpenShift ingress domain	*spec.globalParameters.default_ingress_domain*	Required only for *rhos* (query via *oc get ingresses.config/cluster -o jsonpath={.spec.domain}*).
OpenShift console domain	*spec.componentsParameters.upm-istio-gateway.openshift_console_domain_name*	Required only for OpenShift; used for console linking and UX.

3. コンピューティングノードの要件

ワーカーノードが ** *Planning Architecture* ** フェーズで定義されたとおりに、目的の展開トポロジの最小サイズとフル機能を満たしていることを確認します。

ハイブリッドマネージャーコンポーネントには、**AMD64/x86-64 ノード**が必要です。PostgreSQL 自分自身は ARM64 で実行できますが、混合アーキテクチャクラスター ARM64 + AMD64 は、HM ではサポートされていません。

ノードロールの概要

Node type	Purpose	Count	Kubernetes nodes	Required label
Control plane	Runs HM control plane & telemetry.	3+	Control or Worker	*edbaiplatform.io/control-plane: "true"*
Data plane	Runs Postgres databases.	0 or 3+	Worker	*edbaiplatform.io/postgres: "true"*
AI model (GPU)	*Optional:* Runs AI/ML workloads.	0 or 2+	Worker	*nvidia.com/gpu: "true"*

ノードプール AWS ノードグループ、RHOS マシンセットなどを使用して、リソースを管理し、必要なラベル/テイントを適用します。

> **注釈**
>
> HM システムデータベース—コアプラットフォームデータベースと、ビーコンサーバー、移行ポータル、およびトランスポーター DMS サービスをバッキングするデータベース—は、データプレーンではなくコントロールプレーンノードで実行されます。データプレーンプールは、ユーザーがプロビジョニングされた Postgres クラスターのみをホストします。 !!!

ほとんどの場合、HM はクラウド環境の Kubernetes ワーカーノードで実行され、管理対象制御ノードへのアクセスがないため、EKS コントロールプレーンアクセスを参照してください。オンプレミスクラスターの制御ノード

see Kubernetes control plane documentation またはワーカーノードで実行できます。

3.1 コントロールプレーンノード

オプショナル 厳密な要件。

コントロールプレーンのリソース要件は、監視するデータベースの数に応じて増加します。注テレメトリースタックは、モニタリング負荷に応じてスケーリングします。

サイズガイドライン

Resource	Minimum (Up to 10 DBs)	Standard (10–50 DBs)	Large (>50 DBs)
CPU	8 vCPUs	16 vCPUs	16+ vCPUs
Memory	32 GB RAM	64 GB RAM	64+ GB RAM
Disk	100 GB SSD	200 GB SSD	>200 GB SSD

次のページに続く

表 7 – 前のページからの続き

Resource	Minimum (Up to 10 DBs)	Standard (10–50 DBs)	Large (>50 DBs)
Quantity	3 nodes	3 nodes	3+ nodes
Max pods/node	≥ 60	≥ 60	≥ 60

ポッド密度 ``maxPods`` 見落としやすいですが、CPU/メモリの問題ではありません。コントロールプレーンスタックはポッド密度です。テレメトリとサービスメッシュに加えて、**HM システムデータベースはコントロールで実行されます-plane nodes** ノードロールの概要 を参照してください。すべてのシナリオのインストールでは、**〜150 のポッド**がコントロールプレーンプールに配置されます。これは、最小 3 ノードで約 50 ポッド/ノードです。したがって、各コントロールプレーンノードは**少なくとも 60 のポッド** を許可する必要があります。

これは CPU やメモリとは無関係です。ノードには予備の CPU / RAM があっても、ポッドキャップに到達するとポッドを拒否できます。**AKS では、Azure CNI のデフォルトは ``maxPods=30``** であり、これは低すぎます。コントロールプレーンに固定されたポッドたとえば `upm-beacon-server` は、“*Too many pods*”で `Pending` のままで、インストールは停止します。制限** ノードプールの作成時** `az aks nodepool add … --max-pods 110` 。後で変更することはできません。または、Azure CNI オーバーレイデフォルト 250 を使用します。 GKE デフォルト 110 と EKSENI IP 制限ごとに十分なヘッドルームがありますが、上記の〜60 ポッド/ノードの図と比較して確認します。

3.2 データプレーンノード

オプショナル 厳密な要件。

データプレーンノードは、ユーザーがプロビジョニングした **PostgreSQL** データベースクラスターをホストするようにサイズを調整する必要があります。

*** サイズ設定** 予想されるデータベースワークロード、ストレージ要件、および**フェーズ** *1-***パイプラインオプション** で定義されたパフォーマンス SLA に完全に依存します。

3.3 AI モデルノード

オプショナル GenAI / Agent Factory 機能を利用する場合にのみ必要インストールシナリオ `ai` が有効になっています。

*** サイジング **現在の推奨事項では、** Nvidia B200 GPU**または同等のサポートされているハードウェアが必要です。

ノードの抽象化

HM コントロールプレーンが専用リソースで実行され、Postgres ワークロードが独立したリソースで実行されるように保証、または AI ワークロードの専用リソースを分割するには、**フェーズ** *3Kubernetes* **クラスターの展開** の前にこれらのノードプールに特定の構成を適用する必要があります。

ノードの割り当て可能なリソースの要件

ハイブリッドマネージャーコンポーネントには、スケジュールされているノードが満たす必要がある特定のリソース要件があります。ノードプールを定義するときは、割り当て可能なリソースが、実行する予定のハイブリッドマネージャーコンポーネントと Postgres クラスターの最小要件を満たしていることを確認します。各シナリオはハイブリッドマネージャーコンポーネントのセットに対応し、独自のリソース要件があります。

Scenario	Total CPU	Total Memory	Total Storage	Total Ephemeral Storage
ai	3366m	4749Mi	4Gi	250Mi
analytics	200m	456Mi	0	0
core	10020m	30445Mi	454706Mi	3Gi
dbaas	0	0	0	0
migration	2880m	6432Mi	54Gi	0

さらに、Hybrid Manager オペレーター自分自身には `260m CPU` および `576Mi memory` が必要です。

4. ローカルネットワーキング

ローカルネットワークの要件

HM は、ローカルネットワーキングロジックについて厳密な意見を持っていません。アベイラビリティーゾーン間で **低レイテンシーと高帯域幅**を提供する場合、主要なクラウドサービスプロバイダー AWS、Azure、GCP が提供する標準のネットワーキング機能で十分です。

*** オンプレミス** 高可用性を保証には、スイッチング、リンクボンディング、およびリンクの冗長性に関するベストプラクティスに従う必要があります。

*** プロトコル **** IPv4 のみ**。Kubernetes は IPv4 用に構成する必要があります。IPv6 は、ハイブリッドマネージャー用に完全には製品化されていません。IPv6 が必要な場合、内部通信が IPv4 のままであるように、ロードバランサーの背後でマスクする必要があります。

*** 定義されたネットワークアドレス空間 CIDR **クラスターは、次の個別の非オーバーラップ IP レンジで構成する必要があります。* ポッドネットワーク clusterCIDR すべてのポッドに IP が割り当てられる IP レンジ。**サービスネットワーク** *serviceCIDR* **すべての内部サービス** *ClusterIP* **に仮想** *IP* **が割り当てられる** *IP* **範囲。**ファンクション DNS および NTP クラスターの内部 DNS サービス通常は CoreDNS が実行されており、内部クラスターサービスたとえば kubernetes.default と外部アドレスの両方を解決できる必要があります。

コンテナネットワークインターフェイス CNI

機能的な CNI は、ポッド間ネットワーキングの厳密な要件です。

*** 例** キャリコ、シリウム、フランネル

*** コンテキスト **** データプレーン**の場合、ポッド密度ではなくデータベース I/O がボトルネックであるため、そこで CNI ポッド密度の調整が問題になることはほとんどありません。**コントロールプレーン**

は異なります。ポッド密度が高いため、CNI およびノードごとの maxPods 設定は、コントロールプレーンノードごとに少なくとも 60 のポッドを許可する必要があります。maxPods の設定に関するプラットフォーム固有のガイダンスについては、*3.1* コントロールプレーンノード を参照してください。Cilium は、必要に応じて、ポッド数が非常に多くなると、より効率的です。

*** 検証** CNI セットアップと CIDR の割り当てを確認します。

```
kubectl get nodes -o custom-columns=NAME:.metadata.name,PODCIDR:.spec.podCIDR
kubectl get svc
```

5.Ingress

インフラストラクチャ機能に合ったイングレス戦略を選択する必要があります。HM は、独自のポータルとサービスにロードバランサーまたは NodePort サービスを展開でき、さらに、作成する各 Postgres データベースに個々のロードバランサーと NodePort を展開できます。インフラストラクチャ機能に一致するイングレス戦略を選択します。HM は、独自のポータルとサービスにロードバランサーまたは NodePort サービスを展開できます。作成する Postgres データベースごとに、個々のロードバランサーとノードポートを展開することもできます。次の表は、DNS 管理、エンドポイントタイプ、および SSL ターミネーションのサポートされている組み合わせを定義しています。

Scenario	DNS Management	Endpoint Type	Portal SSL/TLS Termination
CSP LB (ELB, ALB)	Manual portal, dynamic postgres	Dynamic via Load Balancer Controller	Istio Ingress Gateway
F5 + F5 K8s Controller	Manual portal, dynamic postgres	Dynamic via Load Balancer Controller	Istio Ingress Gateway
MetalLB	Manual	Dynamic via Load Balancer Controller	Istio Ingress Gateway
NodePort (Conventional LB)	Manual (A record against LB)	Static (manual)	Istio Ingress Gateway
NodePort (No LB)	Manual (Round Robin against Nodes)	Static (manual)	Istio Ingress Gateway

5.1 ロードバランサーコントローラー

オプション性最適なエクスペリエンスを実現するために必要なオプション。

ロードバランサーコントローラの例は、AWS Load Balancer コントローラから MetalLB または DNS 機能とその F5 ロードバランサーコントローラと組み合わせた BigIP F5 まであります。

一般要件

*** ファイアウォールの注釈** 特定のプロバイダースキーム AWS インターネット接続を使用する場合、ロードバランサーを正しくプロビジョニングするための正しい resourceAnnotations がわかっています。

AWS **の例のアノテーション**

```
globalParameters:
  load_balancer_mode: public
  load_balancer_provider: aws
resourceAnnotations:
  - name: istio-ingressgateway
    kind: Service
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
```

MetalLB の例のアノテーション

```
globalParameters:
  load_balancer_mode: custom-only
resourceAnnotations:
  - name: istio-ingressgateway
    kind: Service
    annotations:
      metallb.io/address-pool: production-public-ips
```

構成の依存関係

Parameter	Custom Resource key	Value / Format
Enable LB	*spec.globalParameters.load_balancer_mode*	*"public"*, *"private"*, or *"disabled"*
Provider	*spec.globalParameters.load_balancer_provider*	*aws*, *azure*, or *gcp*
Annotations	*spec.resourceAnnotations[]*	Provider-specific list (see below).

`load_balancer_mode` パラメーターを使用すると、ポータルと Postgres データベースの両方のロードバランサーサービスを作成するように HM に指示できます。`disabled` に設定されている場合、ポータルと Postgres データベースの両方が NodePort を使用します。それ以外の場合、ロードバランサーを使用します。`public` および `private` 値は Postgres データベースにのみ影響を与え、ポータルロードバランサーには影響しません。同様に、`load_balancer_provider` パラメーターは Postgres クラスターにのみ適用されます。`resourceAnnotations` パラメーターは、HM ポータルにのみ関連します。

スポット検証

コントローラがアクティブであることを確認するには、次のコマンドを使用します。

```
kubectl get pods -A | grep -i load\|lb\|router\|metallb\|gateway
```

HM コントロールプレーンポート

ハイブリッドマネージャーが機能するには、ロードバランサーまたはファイアウォールで次のポートを開く必要があります。

Port	Protocol	Service	Description
80	TCP	*http2*	HM ポータル HTTP — HTTPS にリダイレクトします
443	TCP	*https*	HM ポータル HTTPS イングレス
9443	TCP	*grpc-tls*	ビーコン gRPC API
8444	TCP	*spire-tls*	Spire TLS
9445	TCP	*beacon-spire*	ビーコンスパイア TLS
9198	TCP	*kafka*	Kafka 移行シナリオに必要
24224	TCP	*fluent-bit*	クラスター間ログ転送マルチ DC のみ
10901	TCP	*thanos-query*	クラスター間メトリックマルチ DC のみ

Postgres ポートデータプレーン

適切なロードバランサーが設定されている場合、次は Postgres ポートです。

Port	Protocol	Description
5432	TCP	デフォルト PSQL
6432	TCP	PGD 接続マネージャー PSQL

SSL ターミネーション

HM は、TLS を使用して Istio ゲートウェイを介して公開されます。Istio の前にロードバランサーを使用して HM にアクセスしている場合、2 つのオプションがあります。

1. **SSL パススルー**変更せずに TCP トラフィックを転送するようにロードバランサーを構成します。
2. **TLS 終了**ロードバランサーでクライアントから TLS セッションを終了し、ロードバランサーに Istio との新しい TLS 接続を確立させます。

5.1.2 NodePort 代替

オプショナル *5.1* **ロードバランサーコントローラー** の代替。

ロードバランサーが利用できない場合、 **NodePort** 戦略を使用する必要があります。

*** 構成** HM 構成で `spec.globalParameters.load_balancer_mode: "disabled"` を構成し、`spec.globalParameters.node_port_domain` を定義する必要があります。

*** コンテキスト**デフォルトでは、**HM コンポーネントは特定の NodePort 値でサービスを公開します。**NodePort を直接使用するか、これらのポートを** MetalLB **ソフトウェアロードバランサーまたは**ハードウェアロードバランサー**でフロントして、よりわかりやすい DNS 名とフェールオーバーを向上させることができます。

デフォルトのノードポートの割り当て

入力に NodePort を使用する場合、次の特定のポートは **NodePort** を使用する必要があります。

*** コンストレイン** `node_port_domain` ベース DNS 名を定義する必要があります。

*** コンテキスト** HM コンポーネントは、特定の NodePort 値でサービスを公開します。NodePort を直接使用するか、これらのポートのフロントにハードウェアロードバランサーを使用できます。

構成の依存関係

Parameter	Custom Resource key	Value / Format
Disable LB	*spec.globalParameters.load_balancer_mode*	*"disabled"*
Domain	*spec.globalParameters.node_port_domain*	Postgres DNS domain (e.g., *pg.myorg.com*).

デフォルトのノードポートの割り当て

NodePort variable	Default port	LB port	Description
spec.componentsParameter istio-gateway.ingress_http_node	**32542**	80	HM ポータル HTTP
spec.componentsParameter istio-gateway.ingress_https_nod	**30288**	443	HM ポータル HTTPS
*spec.componentsParameter istio-gateway.ingress_grpc_tls_*	**30290**	9443	ビーコン gRPC TLS
*spec.componentsParameter istio-gateway.ingress_spire_tls_*	**30292**	8444	Spire TLS
spec.componentsParameter istio-gateway.ingress_beacon_sp	**30294**	9445	ビーコンスパイア TLS
spec.componentsParameter istio-gateway.ingress_thanos_qu	**30296**	10901	クラスター間メトリックマルチ DC のみ
*spec.componentsParameter istio-gateway.ingress_fluent_bit_*	**30298**	24224	クラスター間ログ転送マルチ DC のみ
spec.componentsParameter istio-gateway.enable_server_ses	n/a	n/a	サーバーストアドセッション *"true"* `または"false"`を有効にする

Postgres ノードポート

Postgres クラスターは、`node_port_domain` を使用し、ポート範囲 `30000`+ でプロビジョニングされたポートでインクリメントします。

5.2 DNS

オプション DNS 構成は、**最適なエクスペリエンスを実現するために必要なオプション** です。

クラスター内と外部アクセスの両方で適切な DNS 解決を保証する必要があります。

内部クラスター DNS および NTP

*** CoreDNS** クラスターの内部 DNS サービス通常は CoreDNS が実行されており、内部クラスターサービス `kubernetes.default` などと外部インターネットアドレスの両方を解決できる必要があります。

*** NTP:** ノード全体での時間同期を保証には、機能的 NTP が必要です。

*** Container Registry:** DNS サービスは、コンテナレジストリローカルまたは EDB の DNS エントリを解決できる必要があります。

構成の依存関係

次のドメインをプロビジョニングおよび制御する必要があります。これらは、次のフェーズでの HM 構成への入力です。

Domain Purpose	Custom Resource key	Description
Portal domain	*spec.globalParameters.portal_*	HM ポータル UI のホスト名。
Agent domain	*spec.componentsParameters.up* *beacon.server_host*	Beacon Server API が到達可能なホスト名。
Migration domain	*spec.globalParameters.dms_do*	Transporter 移行サービスの内部ドメイン。 `migration`インストールシナリオが有効になっていない限り必須。
NodePort base	*spec.globalParameters.node_p*	**シナリオ B のみ** NodePort を使用する場合の Postgres アクセスのベースドメイン。

NodePort アプローチの場合、上記の DNS エントリは、Kubernetes ノードでコントロールプレーンラベルが適用されるコンピューティング IP でラウンドロビンである必要があります。

DNS 構成戦略

DNS 構成は、Ingress の選択によって異なります。

シナリオ A ロードバランサー推奨

*** 依存関係** HM 構成で `spec.globalParameters.load_balancer_mode:` が`"public"` または`"private"` に設定されていることを確認します。

*** ポータル **AWS ELB または DNS 機能を備えた F5 のような最適なロードバランサーコントローラの場合、``istio-ingress`` の結果のロードバランサー IP をポイントするように DNS を手動で構成します。Kubernetes のイングレスサービスが HTTPS 要求のホスト名に従ってトラフィックを適切にルーティングできるように、ドメイン名は**ローカルルーティング可能な IP**に解決可能である必要があります。

*** Postgres** すべてのデータベースの DNS を手動で管理する必要はありません。

Postgres DNS レコードは、ロードバランサーコントローラがその Postgres クラスター専用の名前空間のサービスのステータスを更新する機能として、自動的に入力されます。

シナリオ B: ノードポートの代替

*** 前提条件** ロードバランサーを無効にし、Ingress セクションで `node_port_domain` を定義しました。

*** ポータル **DNS エントリは、** コントロールプレーンノード **``edbaiplatform.io/control-plane``** のラベル付けのノードのコンピューティング IP を指す**ラウンドロビン A レコード**である必要があります。

*** Postgres: **``node_port_domain``** 値は、すべての Postgres インスタンスのベース URL として使用されます。これは、Postgres クラスターが実行されている**ワーカーノード **の IP アドレスを指す**ラウンドロビン DNS レコード**である必要があります。

5.3 証明書管理

オプショナル 厳密に必須。

安全な通信には、ポータルと API エンドポイントの TLS 証明書が必要です。インストールする前に、証明書管理戦略を決定する必要があります。

決定マトリックス

Option	Description	Typical use case
A. Custom cert-manager issuer	既存の `cert-manager` 発行者を使用します。	Production (Recommended).
B. Customer CA	独自の CA を使用して内部証明書に署名します。	Enterprises with internal PKI.
C. Customer Certificate	独自の事前生成された x.509 証明書とプライベートキーを Kubernetes シークレットとして提供します。	Specific organization-issued cert.
D. Self-signed	インストーラーは自己署名証明書を生成します。	**Test/Non-Production only.**

戦略別の構成の依存関係

Strategy	Custom Resource key	Description
A. Cert Manager	*spec.globalParameters.portal_*	**必須。** リソース種類 `Issuer` または `ClusterIssuer`。
A. Cert Manager	*spec.globalParameters.portal_*	**必須。** リソース名たとえば *letsencrypt-prod*
B. BYO CA	*spec.globalParameters.ca_secr*	**必須。** CA シークレットの名前。
C. BYO Cert	*spec.globalParameters.portal_*	**必須。** 証明書シークレットの名前。
D. Self-signed	N/A	**デフォルトの動作** 構成は必要ありません。

6. ブロックストレージデータベース＆ログ

オプショナル 厳密に必須。

永続ストレージボリューム PVC を提供する Kubernetes `StorageClass` を定義する必要があります。

HM コントロールプレーンに使用されるデフォルトのストレージクラスは、HM 構成で設定されます。同じストレージクラスは、すべての Postgres インスタンスの展開のオプションでもあります。ただし、**データプレーン** Postgres 展開の場合、クラスター内に任意の数の追加ストレージクラスを確立できます。それらはすべて Postgres 展開で使用可能なオプションであり、重要なまたは集中的なワークロードに必要な柔軟性を提供します。

コントロールプレーンの要件

HM コントロールプレーンのブロックストレージの要求は比較的軽いです。 **AWS gp2** の機能とほぼ同等のストレージクラスで十分です。

データプレーンの要件 Postgres

Postgres ワークロードは、特定の I/O パターンによっては、特殊な構成が必要になる場合があります。

*** レイテンシーの感度**Postgres は、一般に生のスループットよりもレイテンシー、特にレイテンシーの* 一貫性*に対してより敏感です。

*** 例分析 **分析機能は、PVC ごとに約** 16,000 IOPS**容量で予想通りに実行されます。

*** 例 Azure** Azure では、標準レベルと比較して非常に一貫したレイテンシーのため、低 IOPS 容量をプロビジョニングする場合でも、IOPS 対応ストレージプレミアム SSD が非常に優先されます。

重要な考慮事項

ハイブリッドマネージャーとコンテナ化された Postgres は、**ローカルストレージ**を活用するのに特に適しています。従来のアプリケーションとは異なり、これらのソリューションは、ストレージプロバイダーによって Kubernetes クラスター全体で複製される PVC に厳密には依存しません。Postgres インスタンスが基になるノードおよびローカルディスクを失った場合、新しいポッドは古い PVC を再マウントする必要はありません。代わりに、ハイブリッドマネージャーの高可用性 HA ロジックは、ピアまたはバックアップから新しいノードを再構築します。

構成の依存関係

Parameter	Custom Resource key	Value / Format
Storage Class	*spec.globalParameters.storage_class*	The exact name of your K8s StorageClass.

検証

使用可能なストレージクラスを確認します。

```
kubectl get sc
```

6.1 ボリュームのスナップショット

*** オプション** 最適なエクスペリエンスに必要です。これがないとバックアップは制限されます。

基になるストレージプロバイダーは**ボリュームスナップショットをサポートする必要があります**。プラットフォームのバックアップおよびリカバリー機能を有効にします。

*** メカニズム** ハイブリッドマネージャーは、機能が Kubernetes API を介して公開されている場合、ボリュームスナップショットを自動的に利用します。

*** 実装ノート ***** クラウド: **通常は CSP の CSI ドライバー AWS EBS CSI によって処理されますが、多くの場合「スナップショットコントローラ」アドオンのインストールが必要です。** * オンプレミス:** ストレージバックエンドは CSI スナップショッター仕様をサポートする必要があり、それにマップする `VolumeSnapshotClass` を定義する必要があります。

検証

スナップショットクラスが構成され、使用可能なことを確認します。

```
kubectl get volumesnapshotclass
```

6.2 ハイブリッドマネージャー検出からのストレージの除外

共有 Kubernetes 環境では、一部の `StorageClasses` または `VolumeSnapshotClasses` は Postgres ワークロードたとえば、GPU または HM 専用クラスに適していない場合があります。デフォルトでは、検出されたすべてのクラスはハイブリッドマネージャー **Create Cluster** ドロップダウンに表示されます。これにより、オペレーターは Postgres クラスターが正常に展開されないクラスを選択する可能性があります。

ハイブリッドマネージャープロビジョニング UI からクラスを非表示にするには、値が `true` である `beacon.enterprisedb.com/exclude-storage-from-dbaas` ラベルを適用します。

```
kubectl label storageclass <name> beacon.enterprisedb.com/exclude-storage-from-
↪dbaas=true
kubectl label volumesnapshotclass <name> beacon.enterprisedb.com/exclude-storage-from-
↪dbaas=true
```

動作

*** オプトイン** 正確な値`"true"` のみがクラスを除外します。存在しないラベル、または他の値は、現在の動作を保持します。この機能は、既存の展開の no-op です。

*** デフォルトセーフ** 除外された `StorageClass` は、`storageclass.kubernetes.io/is-default-class` アノテーションが付いている場合でも、ハイブリッドマネージャーの検出され

たデフォルトに昇格することはありません。

*** 次のエージェントハートビートに有効になります。** エージェントを再起動する必要はありません。

*** UI のみ** ラベルは、ハイブリッドマネージャーのプロビジョニング UI からクラスを非表示にします。クラスを名前で参照するワークロードたとえば、カスタムリソースまたは直接 `kubectl apply` は、引き続きそれを使用できます。

クラスをプロビジョニング UI に復元するには、ラベルを削除します。

```
kubectl label storageclass <name> beacon.enterprisedb.com/exclude-storage-from-dbaas-
kubectl label volumesnapshotclass <name> beacon.enterprisedb.com/exclude-storage-from-
↪dbaas-
```

検証

現在除外されているクラスを監査します。

```
kubectl get sc,volumesnapshotclass -l beacon.enterprisedb.com/exclude-storage-from-
↪dbaas=true
```

7. オブジェクトストレージ

オプショナル 厳密に必須。

オブジェクトストレージは、ディザスターリカバリーとデータモビリティのバックボーンです。次の目的で使用されます。

- Velero バックアップクラスター DR
- Postgres WAL アーカイブポイントインタイムリカバリー
- GenAI / AIDB の非構造化データ
- Lake Keeper 用の寄木細工ファイル
- ログファイル

一般的な要件

*** プロトコル **完全な** S3 互換**である必要があります。

*** 一貫性** 複数の場所の展開では、オブジェクトストレージ構成はすべての場所で等しく利用できる必要があります。詳細については、*multi-dc guidance* を参照してください。

> **注釈**
>
> *default* 名前空間に `edb-object-storage` という名前の Kubernetes Secret をプロビジョニングする必要があります。 このシークレットの内容は、すべてのクラスターで同一である必要があります。 プロセスのこの時点では、*kubectl create secret* を実行する必要はありません。 ここでは、生の資格情報 API キー、証明書、パスワードを収集しているため、 **フェーズ** *4* **環境の準備** のクラスターに注入する準備が整います。 !!!

重要

オブジェクトストレージは完全にハイブリッドマネージャー専用である必要があります。 オブジェクトストレージに存在する以前のデータは、HM の正常なオペレーションを妨げます。 同様に、HM を削除して再インストールした場合、新しいインストールを開始する前にオブジェクトストレージを空にする必要があります。 !!!

認証戦略

プロバイダーがサポートする認証方法を選択します

Provider	Auth Method	Example Use Case
AWS (EKS/ROSA)	Workload Identity (IAM)	Native AWS integration (Recommended).
AWS (Other K8s)	Static Keys	Generic Kubernetes connecting to AWS S3.
Azure Blob	Static Keys	Connect via Connection String/Key.
GCP Storage	Static Keys	Base64-encoded Service Account JSON.
S3 Compatible	Static Keys	On-Premises (MinIO, Ceph, etc.).

構成の依存関係

構成ファイル

これらは、HM インストールの構成キーです。

8. コンテナレジストリ

> **オプショナル** ローカルコンテナレジストリは、**最適なエクスペリエンスを実現するために必要なオプション** です。

HM および Postgres イメージをホストするには、Kubernetes クラスターからアクセス可能なコンテナレジストリが必要です。

*** 目的** コントロールプレーンとデータプレーンに必要な特定のアプリケーションイメージを提供するため。

*** アクセス要件** クラスターは、このレジストリからイメージをプルできる必要があります。

*** 権限 **使用される認証資格情報には、** リポジトリのリスト **、** タグのリスト **、およびタグマニフェストの読み取り**に対する権限が必要です。

環境がパブリック EDB リポジトリにアクセスできない場合は、アーティファクトをプルし、`edbctl` を使用してローカルのプライベートレジストリにプッシュする必要があります。

レジストリアプローチの選択

2 つのレジストリオプションがあります。

*** 外部プライベートレジストリ推奨。** 組織の既存のレジストリを使用します—たとえば、AWS ECR、Google Artifact Registry、Azure Container Registry、または Harbor など。確立されたレジストリツール、ス

キャン、およびアクセス制御を使用する環境に最適です。*Syncing images to a local private registry* を参照してください。

*** クラスター内コンテナレジストリ。** EDB HCP Operator 自分自身に埋め込まれたレジストリ。ノードにパブリック IP がない完全にエアギャップクラスターにのみ適しています。*Using the in-cluster container registry* を参照してください。

ノードにパブリック IP がある場合は、クラスター内レジストリを使用しないでください。代わりに外部のプライベートレジストリを使用します。

Registry provider	Supported Auth	Recommended Auth
Azure (ACR)	Token, Basic	Token
Amazon (ECR)	EKS Managed Identity	EKS Managed Identity
Google (GAR)	Token, Basic	Basic
EDB Repo	Token	Token (PoC/Pilot only!)

準備

*** イメージをローカルコンテナレジストリに同期する**プライベートレジストリを使用する場合、`edbctl` を使用してイメージを同期する必要があります。* *SOP: Sync images to local registry**

** ガイド `edbctl` を使用してプラットフォームとオペレーターイメージをミラーリングします。*

*** シークレット** プル資格情報を使用して Kubernetes シークレットをプロビジョニングする必要があります。実装の詳細については、**フェーズ 4 環境の準備** を参照してください。

> **注釈**
>
> **イメージ検出の構成** イメージがローカルレジストリに同期したら、それらを見つけるようにインストーラーを構成する必要があります。 これにより、クラスターは正しい場所ローカルレジストリ対パブリックレジストリからアーティファクトをプルします。

- *SOP: Image discovery configuration*

** ガイダンス: レジストリミラーとプルポリシーの構成。*

次のフェーズフェーズ 3 での HM 構成用にこれらの値を収集します。

Parameter	Custom Resource key	Value / Format
Registry URL	*spec.imageRegistry*	*<Registry Domain>/pgai-platform*
Image version	*spec.version*	HM version tag (e.g., *"v2026.5.1"*).
Pull Secret	*spec.imagePullSecrets[]*	Kubernetes image pull secret references (for example, *[{name: edb-cred, namespace: edbpgai-bootstrap}]*).

次のページに続く

表 21 – 前のページからの続き

Parameter	Custom Resource key	Value / Format
Discovery Registry	*spec.beaconAgent.provisioning.imagesetDiscoveryC*	*<Registry Domain>/pgai-platform*
Discovery Auth	*spec.beaconAgent.provisioning.imagesetDiscoveryAu*	*<Auth Type>* (e.g., *token*, *basic*, *eks_managed_identity*)
TLS Validation	*spec.beaconAgent.provisioning.imagesetDiscoveryAl*	Set to *true* only if intentionally allowing insecure (no/invalid TLS) registry discovery.

9.ID プロバイダー IdP

ハイブリッドマネージャーは、ユーザー認証を直接提供することを目的としたものではありません**。外部 ID プロバイダー IdP に依存して、ユーザーアクセスを安全に管理します。

オプショナル 最適なエクスペリエンスとセキュリティのために必要です。

*** メカニズム** OIDC を介して既存のディレクトリサービスに接続します。

*** サポートされているバックエンド** LDAP または SAML。

Preparing your environment フェーズ 4 用に次の値とアーティファクトを準備する必要があります。

10. キー管理サービス KMS

Postgres データベースの **透過的データ暗号化 TDE** を有効にするには、外部キー管理サービス KMS を統合する必要があります。

*** オプショナル** 最適なセキュリティとコンプライアンスのために必要です。

*** メカニズム** ハイブリッドマネージャーは、クラウドプロバイダーまたは外部ボールトと統合して、暗号化キーを安全に管理します。

サポートされている認証

*** ワークロード ID** ネイティブクラウド認証 EKS / GKE / AKS で推奨。

*** 資格情報** Kubernetes シークレットに保存されている静的キー。

構成の依存関係

構成ファイル

フェーズ *4* ガイド の HM 構成用に次の値を準備する必要があります。

Parameter	Custom Resource key	Value / Format
TDE Providers	*spec.beaconAgent.transparentDataEncryptionMetho*	List of enabled methods (e.g., *passphrase*, *aws_kms*, *azure_kms*, *gcp_kms*, *hashicorp_vault*).

*** 認証シークレット** `credentials` モードを使用する場合、特定の Kubernetes シークレットを API キーでプロビジョニングする必要があります。

構成への影響

このドキュメントで詳細に説明されているインフラストラクチャの仕様は、フェーズ 1 でのアーキテクチャの決定と組み合わせて、次のフェーズで使用する値を決定します。

要件インベントリガイド

フェーズ 3 に進む前に、このインベントリガイドを使用して、必要な値をすべて決定していることを確認します。

`HybridControlPlane` CR キーは、要件ごとに指定されます。

一般&プラットフォーム

Source	Requirement	Custom Resource key	Condition
Phase 1	**K8s Flavor**	*spec.flavour*	**Required** (e.g., *eks*, *gke*, *aks*, *rhos*)
Phase 1	**Provider**	*spec.beaconAgent.pro*	**Required** (e.g., *AWS*, *GCP*)
Phase 1	**OpenShift**	*spec.beaconAgent.pro*	**Required** if using RHOS (*true*)
Phase 1	**Location**	*spec.componentsPara beacon.beacon_locat*	**Required** (String ID)
Phase 1	**Installation scenarios**	*spec.scenarios[]*	**Optional** . All scenarios enabled by default. Customize with *core* plus any of *migration*, *ai*, *dbaas* or *analytics*.

ネットワーキングとイングレス

Source	Requirement	Custom Resource key	Condition
Phase 2	**Portal Domain**	*spec.globalParameter*	**Required**
Phase 2	**Agent Domain**	*spec.componentsPara beacon.server_host*	**Required**

次のページに続く

表 24 – 前のページからの続き

Source	Requirement	Custom Resource key	Condition
Phase 2	**Migration Domain**	*spec.componentsPara rw-service.domain_name*	**Required**
Phase 2	**Migration URL**	*spec.componentsPara dp-agent.rw_service_url*	**Required**
Phase 2	**Load Balancer Mode**	*spec.globalParameter*	**Required** (*"public"*, *"private"*, or *"disabled"*)
Phase 2	**LB Annotations**	*spec.resourceAnnotat*	**Required** if using CSP LB (e.g., AWS)
Phase 2	**NodePort Domain**	*spec.globalParameter*	Required if load balancer mode is *"disabled"*

ストレージとレジストリ

Source	Requirement	Custom Resource key	Condition	
Phase 2	**Storage Class**	*spec.globalParameter*	**Required**	
Phase 2	**Registry URL**	*spec.imageRegistry*	**Required**	
Phase 2	**Image Version**	*spec.version*	**Required**	
Phase 2	**Discovery Reg.**	*spec.beaconAgent.pro*	**Required**	*rRegistryURL*
Phase 2	**Registry Auth**	*spec.beaconAgent.pro*	**Required**	*ationType*
Phase 2	**Insecure TLS**	*spec.beaconAgent.pro*	Optional (if using self-signed registry)	*ecureRegistry*

セキュリティと ID

Source	Requirement	Custom Resource key	Condition	
Phase 2	**IDP Secret**	*spec.componentsPara*	**Required** (if using OIDC)	*ıtSecret*
Phase 2	**IDP Config**	*spec.componentsPara*	**Required** (if using OIDC)	*Connectors*
Phase 2	**Static User**	*spec.componentsPara*	**PoC Only** (if no IdP)	*cPasswords.**
Phase 2	**TDE Keys**	*spec.beaconAgent.tra*	Optional (if using TDE)	

証明書 (戦略を 1 つ選択してください)

いずれか 戦略 A、B、C、または D を選択し、対応する値を記録します。

Strategy	Custom Resource key	Description
A. Cert Manager	*spec.globalParameters.portal_*	**必須。** リソース種類たとえば *ClusterIssuer*
A. Cert Manager	*spec.globalParameters.portal_*	**必須。** リソース名たとえば *letsencrypt-prod*
B. BYO CA	*spec.globalParameters.ca_secr*	**必須。** CA シークレットの名前。
C. BYO Cert	*spec.globalParameters.portal_*	**必須。** 証明書シークレットの名前。
D. Self-signed	N/A	**デフォルトの動作** 構成は必要ありません。

構成例

前の手順で行った決定に従って、構成ファイルはそれらの選択を反映する必要があります。

これは、HybridControlPlane CR の構成です。完全なフィールドリファレンスについては、**構成マッピングリファレンス** を参照してください。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  flavour: <Kubernetes> # e.g., eks, gke, rhos, rke2
  imageRegistry: <Container Registry Domain>/pgai-platform
  version: <Version>

  imagePullSecrets:
    - name: edb-cred
      namespace: edbpgai-bootstrap

  globalParameters:
    CONTAINER_REGISTRY_URL: <Container Registry Domain>/pgai-platform
    portal_domain_name: <Portal Domain>
    storage_class: <Block Storage>
    portal_certificate_issuer_kind: <ClusterIssuer>
    portal_certificate_issuer_name: <my-issuer>
    trust_domain: <Portal Domain>

  componentsParameters:
    upm-beacon:
      beacon_location_id: <Location>
      server_host: <Agent Domain>
    transporter-rw-service:
      domain_name: <Migration Domain>
    transporter-dp-agent:
      rw_service_url: https://<Migration Domain>/transporter

  beaconAgent:
    provisioning:
```

(次のページに続く)

(前のページからの続き)

```
    imagesetDiscoveryAuthenticationType: <Authentication Type for the Container␣
↪Registry>
    imagesetDiscoveryContainerRegistryURL: "<Container Registry Domain>/pgai-
↪platform"
  transparentDataEncryptionMethods:
    - <available_encryption_method>

scenarios:
  - core
  - <if required, migration>
  - <if required, ai>
  - <if required, analytics>
  - <if required, dbaas>

resourceAnnotations:
  - name: istio-ingressgateway
    kind: Service
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-scheme: internal
      service.beta.kubernetes.io/load-balancer-source-ranges: 10.0.0.0/8
```

次のフェーズ

これで、完全なインベントリと展開戦略、および HM 構成の初期入力 HybridControlPlane CR が完成しました。続行して構成のビルドを続けるか、これらの入力を使用して**フェーズ *4* 環境の準備** の最後にファイルをアセンブルします。

** *Phase 3: Deploying your Kubernetes cluster* → に進んで、Kubernetes クラスターをプロビジョニングおよび展開します。**

4.1.1.8.3 Deploying your Kubernetes cluster

概要

重点ロール Kubernetes エンジニア/CSP 管理者

前提条件

- **フェーズ *1* アーキテクチャの計画** 完成品
- **フェーズ *2* システム要件の収集** 完成品

結果

** ハイブリッドマネージャー HM のコンピューティング、ストレージ、およびネットワーキング要件をすべて満たす、実行中の到達可能な Kubernetes クラスター。*

> **注釈**
>
> **EDB support context:** お客様は、Kubernetes クラスターのインストールとライフサイクル操作を所有します。 プロフェッショナルサービスは、作業ステートメント SoW を介して従事でき、サポートはナレッジベース記事を介して支援を提供できます。 !!!

次のフェーズ フェーズ 4 *Preparing your environment*

プロビジョニングプロセスを開始する

システム要件を定義したら、インフラストラクチャをプロビジョニングする準備が整います。このフェーズでは、Hybrid ManagerHM プラットフォームをホストするベース Kubernetes クラスターの展開について説明します。

HM は主にプラットフォームに依存せず、主要なクラウドプロバイダーとオンプレミスディストリビューションをサポートしています。プラットフォーム固有の変動の場合、

管理ワークステーションをセットアップする

前のフェーズ **フェーズ** *2* **システム要件の収集**で計画したように、管理ワークステーション Bastion またはその他をプロビジョニングおよび構成します。

コアツールのインストール

展開をオーケストレーションするには、次の CLI ツールをインストールする必要があります。

1. **Kubernetes ツール ``kubectl`` および ``helm`` をインストールします。** 公式ガイドに従って、kubectl および Helm をインストールします。
2. **``edbctl`` EDB ハイブリッドマネージャー CLI をインストールします。**インストールスクリプトを実行して、バイナリをダウンロードします。

```
curl -sfL https://get.enterprisedb.com/edbctl/install.sh | sh -
```

3. **ユーティリティのインストール**

yq v4+、curl 、および openssl がパッケージマネージャーを介してインストールされていることを確認します。

```
# Example for Ubuntu/Debian
sudo apt-get update && sudo apt-get install -y curl openssl

# Example for MacOS
brew install yq curl openssl
```

4. **プラットフォーム CLI のインストール環境に依存します。**

パブリッククラウドに展開する場合は、認証に関連する CLI をインストールします。*** AKS:** Install Azure

*** AWS:** Install AWS CLI

*** Google Cloud:** Install gcloud SDK

*** OpenShift:** Install oc CLI

接続の検証

構成フェーズに進む前に、ワークステーションが必要なエンドポイントに到達できることを確認します。

1. **インターネット/レジストリアクセスの確認**

```
curl -I https://docker.enterprisedb.com/v2/
```

1. **クラウド ID の検証 AWS / GCP を使用する場合**

AWS の場合

```
    aws sts get-caller-identity
```
```または GCP の場合

```
gcloud auth list
```

```
1.  **ローカルツールの検証**
```

```
edbctl version
kubectl version --client
```

```
 ..  Note::
 ワークステーションが、インストール中に一時的なブートストラップクラスターをホストするための CPU および RAM の要件を満たしていることを確認します。 !!!

 ##  Kubernetes クラスターを展開する

 管理ワークステーションの構成が完了したら、前のフェーズで定義した関連仕様に従ってクラスターを展開する準備が整います。

 ###  展開パスを選択してください

 フェーズ 1 または 2 で選択した **Kubernetes フレーバー** と一致する以下のガイドを選択し、
↪:ref:`フェーズ 2 システム要件の収集 <フェーズ 2 システム要件の収集>` で定義されたシステム要件に従って展開を進めます。

 ###  パブリッククラウドプロバイダー

 .. csv-table::
:header: Provider,Official documentation,EDB Knowledge Base,Key considerations
:widths: 15,12,12,20
:align: left
```

(次のページに続く)
```

(前のページからの続き)

```
:class: longtable

**Azure (AKS)** , ` **Tutorial - Create an Azure Kubernetes Service (AKS) cluster**
 <https://learn.microsoft.com/en-us/azure/aks/tutorial-kubernetes-deploy-cluster>`__ ,
 ` **HM on Azure Guide**  <https://knowledge.enterprisedb.com/hc/en-us/articles/
↪24863144589980-Guidance-on-AKS-for-Hybrid-Manager>`__ ,""
**AWS (EKS)** , ` **Creating an EKS cluster**  <https://docs.aws.amazon.com/eks/
↪latest/userguide/create-cluster.html>`__ , ` **HM on EKS Guide**  <https://
↪knowledge.enterprisedb.com/hc/en-us/articles/24212491501468-Guidance-on-AWS-for-
↪Hybrid-Manager>`__ ,"Requires IAM OIDC provider, EBS CSI driver, and LoadBalancer␣
↪Controller."
**Google Cloud (GKE)** , ` **Creating a regional cluster**  <https://cloud.google.com/
↪kubernetes-engine/docs/how-to/creating-a-regional-cluster>`__ , ` **HM on GKE␣
↪Guide**  <https://knowledge.enterprisedb.com/hc/en-us/articles/24212699192732-
↪Guidance-on-GCP-for-Hybrid-Manager>`__ ,Requires Workload Identity enabled and␣
↪specific VPC firewall rules.
**Red Hat OpenShift Service on AWS (ROSA),:ref:`Troubleshooting <Getting started with␣
↪Pipeline Designer>` ,n/a,""

 ###  プライベートクラウドとオンプレミス

 .. csv-table::
:header: Distribution,Official documentation,EDB Knowledge Base,Key considerations
:widths: 15,12,12,20
:align: left
:class: longtable

**Rancher RKE2** ,:ref:`Troubleshooting <Langflow quickstart>` , ` **HM on RKE2␣
↪Guide**  <https://knowledge.enterprisedb.com/hc/en-us/articles/24212371387420-
↪Guidance-on-Rancher-for-Hybrid-Manager>`__ ,Requires manual setup of Longhorn (or␣
↪similar) storage and MetalLB for ingress.
**Red Hat OpenShift** ,:ref:`Troubleshooting <>` , ` **HM on OpenShift Guide**
↪<https://knowledge.enterprisedb.com/hc/en-us/articles/24212799018780-Guidance-on-
↪RHOS-for-Hybrid-Manager>`__ ,Requires specific SCC (Security Context Constraints)␣
↪and Route configuration.

 ###  ノードの抽象化を設定する

 ####  コントロールプレーン

 **必須ラベルとテイント**

 *** ラベル** `edbaiplatform.io/control-plane: "true"`

 *** Taint: *** キー: `edbaiplatform.io/control-plane`   *値: `"true"`* エフェクト:␣
```

(次のページに続く)

(前のページからの続き)

```
↪`NoSchedule`

 ```yaml
 spec:
     replicas: 3
     template:
     spec:
         metadata:
         labels:
             edbaiplatform.io/control-plane: "true"
         taints:
         - key: edbaiplatform.io/control-plane
             value: "true"
             effect: NoSchedule
```

!!!重要 コントロールプレーンプールの**ノードごとのポッド制限を少なくとも 60** maxPods に設定します。HM システムデータベースを含むコントロールプレーンスタックは、このプールに約 150 のポッドを配置します。これは、一部のプラットフォームでのデフォルトを超えます **AKS Azure CNI のデフォルトは ``maxPods=30``** で、CPU がメモリは十分です。AKS では、これはノードプールの作成時にのみ設定されます `az aks nodepool add … --max-pods 110`。*3.1* コントロールプレーンノード を参照してください。

必須ラベルとテイント

*** ラベル** edbaiplatform.io/postgres: "true"

*** Taint: *** キー: edbaiplatform.io/postgres **値**: ``"true"`` エフェクト: NoSchedule

```
#  Example Node Pool Specification
spec:
    replicas: 3 # Minimum recommended for High Availability
    template:
      metadata:
        labels:
          edbaiplatform.io/postgres: "true"
      spec:
        taints:
        - key: edbaiplatform.io/postgres
          value: "true"
          effect: NoSchedule
```
``` #### AI モデルノード

!!!note `ai` インストールシナリオが有効になっている展開にのみ必要です。このシナリオは、␣
↪`scenarios` 構成パラメーターを介して手動で除外されない限り、デフォルトで含まれます。

.. ::
* **サイズ設定**現在の推奨事項では、**Nvidia B200 GPU**または同等のサポートされているハード
```

(次のページに続く)
```

(前のページからの続き)

ウェアが必要です。

必須ラベルとテイント

*** ラベル** `nvidia.com/gpu: "true"`

*** Taint: *** キー:`nvidia.com/gpu` *値:`"true"`* 効果:`NoSchedule` ## クラスター仕様を検証する

クラスターの展開が完了したので、続行する前に次のことを行います。

1. :ref:`管理ワークステーションをセットアップする <管理ワークステーションをセットアップする>` が、次のフェーズ :ref:`フェーズ 4 環境の準備 <フェーズ 4 環境の準備>` の構成のすべての関連エンドポイントに到達できることを確認します。

1. 展開したクラスターが :ref:`フェーズ 2 システム要件の収集 <フェーズ 2 システム要件の収集>` で定義した関連仕様と一致することを確認します。

コンピューティングリソース

出典 :ref:`Gathering your system requirements>Compute (Node requirements)<Gathering your system requirements>`

*** ノード数** ノードプールが前のフェーズで決定されたものと一致することを確認します。

*** アーキテクチャ **すべてのノードが** AMD64 (x86_64) **アーキテクチャを使用していることを確認します。* ARM ベースのノードは、現在コントロールプレーンでサポートされていません。*

*** ノードごとのポッド **コントロールプレーンノードで** 各ポッド 60 以上のポッド **が許可されていることを確認します `kubectl get nodes -o jsonpath='{range .items[* ]}{.metadata.name}{"\t"}{.status.allocatable.pods}{"\n"}{end}'` 。 AKS Azure CNI では、デフォルトの 30 は低すぎるため、ノードプールの作成時にを上げる必要があります。

ノード抽象化の検証

インフラストラクチャをプロビジョニングした後、これらのコマンドを実行して、ノードが正しくラベル付けされ、汚染されていることを確認します。これにより、Hybrid Manager インストーラーがポッドを正しくスケジュールできるようになります。

1. **コントロールプレーンノードを検証します。**

```bash
    kubectl get nodes -l edbaiplatform.io/control-plane="true"
```

1. **テイントの検証**

```
kubectl get nodes -o json | jq .items[] | select(.metadata.labels["edbaiplatform.io/
↪control-plane"]=="true") | {name: .metadata.name, taints: (.spec.taints // [] |␣
↪map(select(.key=="edbaiplatform.io/control-plane")))}
```

1. データプレーンノードの検証

```
kubectl get nodes -l edbaiplatform.io/postgres="true"
```

1. テイントの検証

```
kubectl get nodes -o json | jq .items[] | {name: .metadata.name, labels: .metadata.
↪labels["edbaiplatform.io/postgres"], taints: (.spec.taints // [] | map(select(.key==
↪"edbaiplatform.io/postgres")))}
```

ブロックストレージ構成

出典 *Gathering your system requirements>6. Block storage (Database & logs)*

*** ストレージクラス** クラスターに、コントロールプレーン gp2 などのフェーズ 2 で指定した *Storage Class** default* があることを確認します。

*** 機能** 必要に応じて、データプレーン Postgres ワークロードの任意の数のストレージクラスを確立できます。これらは、HM を使用してデータベースクラスターをプロビジョニングするときに使用するオプションです。

Ingress

出典 *Gathering your system requirements>5. Ingress*

*** イングレスコントローラ** 選択したイングレスメカニズムロードバランサーまたはノードポートがアクティブで、必要なポートを公開していることを確認します。

*** DNS:** CoreDNS または同等のサービスが正常で、内部サービス名を解決していることを確認します。

展開後の検証

クラスターが `Active` または `Ready` になったら、これらの簡単なチェックを実行して、 **環境の準備**フェーズの準備ができていることを確認します。

API アクセスの検証

管理ワークステーションをセットアップする から、API サーバーに到達できることを確認します。

```
kubectl cluster-info
```

*** Success** Kubernetes コントロールプレーン URL を結果ます。

ストレージ機能の検証

有効なストレージクラスが存在し、デフォルトに設定されていることを確認します。

```
kubectl get sc
```

*** Success:** 出力は、名前に `(default)` を含むクラス `gp3` などをリストします。

ID の検証クラウドのみ

EKS / GKE / AKS を使用している場合、現在の IAM ID に管理者権限があることを確認します。

```
kubectl auth can-i create secrets --all-namespaces
```

*** 成功** 出力は `yes` です。

次のフェーズ

インフラストラクチャがプロビジョニングされました。これで、このクラスターに接続できます。

次に、イメージをローカルレジストリに同期し、シークレットのステージング、TLS の構成、関連する高度な機能を設定し、HM 構成を完成させます。

** *Proceed to Phase 4: Preparing your environment* →**

4.1.1.8.4 Preparing your environment

概要

重点ロールサイト信頼性エンジニア SRE/インフラストラクチャエンジニア

前提条件

- **フェーズ** *1* **アーキテクチャの計画** 完成品
- **フェーズ** *2* **システム要件の収集** 完成品
- **フェーズ** *3Kubernetes* **クラスターの展開** 完了&実行中

結果

- **必要なシークレット** *TLS*、**認証、ストレージ、その他の必要なシークレットで構成された** *Kubernetes* **クラスター**
- **選択した高度な機能の構成** *IDP*、**マルチロケーション、** *TDE* **の** *KMS)*
- **ハイブリッドマネージャー** *HM* **インストールの準備ができた最終化および検証済みの構成-** ``*HybridControlPlane*`` **カスタムリソースマニフェスト**

EDB サポートコンテキストハイブリッドマネージャーの最終準備は、クラスターのライフサイクルオペレーションと同様に、主に**顧客の責任** *Sovereign Systems* を除く 。 **プロフェッショナルサービス**は、作業ステートメント SoW を介して従事でき、 **サポート**はナレッジベース記事を介して支援を提供できます。

フェーズ 4 ガイド

Kubernetes クラスターが実行されたため、HM プラットフォーム用に準備する必要があります。

これには以下が含まれます。

1. リモート管理ワークステーションを介した管理アクセスを確立します。
2. 必要なアーティファクトとシークレットをクラスターにステージングする
3. アーキテクチャの決定を最終的な構成ファイルに変換する。

シークレットの作成

イメージプルシークレットを作成する

`edbctl` を使用して `ImagePullSecret` を作成します。これにより、Kubernetes クラスターは EDB レジストリにアクセスして、インストールに必要なイメージをプルできるようになります。

```
edbctl image-pull-secret create
```

その他の CLI オプションについては、 *edbctl image-pull-secret* を参照してください。

コンポーネントシークレットを作成する

`edbctl` を使用して、すべての HM コンポーネントが必要なシークレットを作成します `core`、`ai`、`migration`、`analytics`。

```
edbctl hybrid-manager create-install-secrets
```

その他の CLI オプションについては、*edbctl hybrid-manager* を参照してください。コンポーネントのシークレットをカスタマイズするには、*Customizing secrets* を参照してください。

オブジェクトストレージシークレットを作成する

object storage requirement を実装するには、インストール前に `default` 名前空間に `edb-object-storage` Kubernetes シークレットを作成する必要があります。プロバイダー固有の構成と完全なパラメーターリファレンスについては、*Configuring object storage* を参照してください。

TLS の実装

フェーズ *2* システム要件の収集 フェーズ中に選択した TLS ストラテジーを実装する必要があります。一部の TLS 戦略では、シークレットを作成する必要があります。

SOP: Implementing TLS certificates は、実装オプションについて説明します。

高度な機能の設定

マルチロケーションアーキテクチャの構成

複数の場所マルチ DC に HM を展開するには、プライマリおよびセカンダリクラスターが安全に通信できるようにするための大幅な準備が必要です。

トラストドメインを確立し、ストレージシークレットを同期し、セカンダリの場所を登録するように HM 内部ビーコンエージェントを構成する必要があります。

準備チェックリスト

1. **ネットワーク** クラスター間のポート `8444` SPIRE および `9445` Beacon での接続を確認します。
2. **ストレージ** すべてのクラスターで `edb-object-storage` シークレットを同期します。
3. **Identity:** クラスター間の信頼を許可するように SPIRE フェデレーションを構成します。
4. **構成** HM 構成で一意の `beacon_location_id` 値と信頼ドメインを定義します。

- ***SOP: Configuring multiple data centers for Hybrid Manager***
- **ガイド *SPIRE* フェデレーションと *DC* クロス配線をセットアップするための詳細な手順。**

TDE の KMS の実装

データベース TDE の保存時の暗号化が必要な場合は、キー管理サービス KMS プロバイダーを有効にし、今すぐキーを作成する必要があります。

- ***SOP: Adding KMS support***

使用している KMS に応じて、以下のガイドを参照して、対応するキーを作成します。

- *HashiCorp Vault*
- *AWS Key Management Service (KMS)*
- *Google Cloud Key Management Service (Cloud KMS)*

環境の準備の完了

バックアップフォルダーに名前を付けます

1. ディザスターリカバリー用の HM バックアップを保存する新しいフォルダーを作成します。
2. 一意のフォルダー名たとえば UUID ジェネレーターを使用して、`spec.globalParameters.internal_backup_folder` の下の HM 構成に提供します。
3. 文字列は、文字と数字で構成される 12 文字である必要がありますつまり `internal_backup_folder: "a7462dbc7106"` 。

必要なコンポーネント構成を追加します

次のコンポーネント固有のパラメーターを `spec.componentsParameters` の下の HM 構成に追加する必要があります。

```
spec:
  componentsParameters:
    edb-migration-portal:
      enable_pdb: "false"
    upm-api-pnxt-metric:
      log_level: "info"
```

(次のページに続く)

(前のページからの続き)

```
    tsdb_host: "thanos-query.monitoring.svc.cluster.local"
    tsdb_port: "9090"
  upm-api-pnxt-pd:
    log_level: "info"
  upm-beaco-ff-base:
    aes_key_rotation_interval: 2160h
    usage_generator_interval: 24h
  upm-beaco-usages:
    usage_generator_interval: "24h"
  upm-beacon:
    beacon_location_id: <your-location-id>
    server_host: <your-beacon-server-host>
  upm-istio-gateway:
    enable_server_session: "true"
    openshift_console_domain_name: <your-openshift-console-domain>
  upm-prometheus:
    memory: "4Gi"
  upm-thanos:
    memcached_cpu: "500m"
    memcached_memory: "2Gi"
    query_frontend_cpu: "200m"
    query_frontend_memory: "120Mi"
  upm-trust-manager:
    cert_manager_namespace: "cert-manager"
```

クラスター構成の完成

次に、HM インストールの最終構成を構築または更新します。

この構成は、フェーズ 2 の **システム要件**と、フェーズ 4 で準備した環境設定と機能構成を参照します。

HybridControlPlane カスタムリソースを作成する

Hybrid Manager インストールを `HybridControlPlane` カスタムリソースとして構成し、`edb-hcp-operator` チャートで使用します。バージョン管理と Kubernetes ネイティブのリソース処理を提供します。

- 完全な HybridControlPlane マニフェストテンプレートとフィールドマッピングリファレンスに関する ***SOP: Install with the Operator*** ガイド。

構成ファイル、シークレット、ロードバランサー、およびクラスターの準備状況の検証

展開に進む前に、構成ファイルとランタイム環境の両方を検証します。

構成構文の検証

続行する前に、構成ファイルを検証します。

kubectl ドライランを使用して、HybridControlPlane マニフェストを検証します。

```
kubectl apply --dry-run=client -f hybridmanager.yaml
```

このコマンドがエラーなしで完了した場合、マニフェスト構文は有効です。

このドライランは、クラスターの API 検出を介して HybridControlPlane 種類を解決するため、オペレーターのカスタムリソース定義 CRD を既にインストールしている必要があります。オペレーターは、これらの CRD を フェーズ 5 ハイブリッドマネージャーのインストール にインストールします。それまでは、コマンドは error: unable to recognize ... no matches for kind "HybridControlPlane" で失敗します。この検証は、オペレーターのインストールフェーズ 5 の **後**、実際にマニフェストを適用する前に実行します。このフェーズで YAML 構文のみをチェックするには、yq . hybridmanager.yaml などの YAML リンターを使用します。

最初に、Helm リポジトリの add および update を実行します。

```
helm repo add enterprisedb-edbpgai "https://downloads.enterprisedb.com/${EDB_
↪SUBSCRIPTION_TOKEN}/pgai-platform/helm/charts/"
helm repo update
```

次に、 Helm チャートに対して values.yaml を検証します。

```
helm template edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --values values.yaml \
  --version <target-helm-version> > /dev/null
```

このコマンドがエラーなしで完了した場合、YAML 構文は有効です。

シークレットを検証する

構成で参照されたシークレットがクラスター内に存在することを確認します。イメージプルシークレットをリストするには

```
edbctl image-pull-secret list
```

コンポーネントのシークレットをリストするには

```
edbctl hybrid-manager list-install-secrets
```

ロードバランサーの検証

クラスターが外部ロードバランサーサービスイングレスに必要をプロビジョニングできることを確認します。

```
kubectl create service loadbalancer test-lb --tcp=80:80
kubectl get svc test-lb
```

- **成功**ロードバランサーコントローラが機能している場合、test-lb サービスには外部 IP またはホスト名が割り当てられます。
- **クリーンアップ** `kubectl delete svc test-lb`

診断ツールを実行する

クラスターの準備状況を包括的にチェックするには、診断プラグインを使用します。

- **SOP: Use kubectl-edb-diagnostic** 。

次のフェーズ

管理ワークステーションの準備が整い、イメージの同期、シークレットの作成、高度な機能の構成、HM 構成の最終化、構成ファイルの検証が完了したため、HM をインストールする準備が整います。

** *Proceed to Phase 5: Installing Hybrid Manager* → **

4.1.1.8.5 Install Hybrid Manager

概要

重点ロールサイト信頼性エンジニア SRE/インフラストラクチャエンジニア

前提条件

- **フェーズ** *1* **アーキテクチャの計画** 完成品
- **フェーズ** *2* **システム要件の収集** 完成品
- **フェーズ** *3Kubernetes* **クラスターの展開** 実行および検証済み
- **フェーズ** *4* **環境の準備** 完了および検証済み

結果

** 完全に展開されたハイブリッドマネージャー HM プラットフォーム、Web ポータルからアクセスできます。*

> **注釈**
>
> **展開の責任** ハイブリッドマネージャー HM のインストールは、**顧客の SRE チーム**によって最適に実行され、顧客がそのライフサイクルオペレーションを所有します。 *Sovereign System* がある場合、EDB CX チームは HM インストールを実行します。 **プロフェッショナルサービス**および**セールスエンジニアリング**もサポートを提供することができ、**サポート**は利用可能なナレッジベース記事を提供しています。 !!!

次のフェーズ フェーズ 6 *Exploring post-installation options*

ハイブリッドマネージャーのインストール

環境を準備し、構成を検証すると、HM を展開する準備が整います。

このフェーズには、構成アーティファクトを Kubernetes クラスターに適用し、展開のモニタリングが含まれます。

ハイブリッドマネージャーは、 **EDB Postgres AI Operator** helm チャートと **HybridControlPlane Custom Resource** を使用してインストールされます。このアプローチは、バージョンの柔軟性、Kubernetes ネイティブのリソース管理、および GitOps サポートを提供します。

HybridControlPlane カスタムリソースについて

HybridControlPlane は、Hybrid Manager インストールの望ましい状態を定義する Kubernetes カスタムリソースです。

演算子アプローチの主な利点

1. **Kubernetes ネイティブ**カスタムリソースを使用して標準の Kubernetes パターンに従います
2. **バージョンの柔軟性**オペレーターバージョンとは無関係に、さまざまな HM バージョンをインストールできます。
3. **GitOps フレンドリー** CR マニフェストは GitOps ツール ArgoCD、Flux とシームレスに動作します
4. **宣言的** 演算子は、目的の状態と一致するように実際の状態を調整します

最終構成の検証

インストールする前に、以下の例をテンプレートとして使用して HybridControlPlane マニフェストを準備します。

HybridControlPlane マニフェストの例

次に、構造を示す完全な例を示します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
  annotations:
    biganimal.enterprisedb.io/deletion-protect: enabled
spec:
  # Basic configuration
  flavour: eks  # or aks, gke, rhos, rke
  imageRegistry: docker.enterprisedb.com/pgai-platform
  version: "v2026.5.1"

  # Source configuration
  source:
```

(次のページに続く)

(前のページからの続き)

```
    remote: true  # Use remote bootstrap image
    # OR
    # useLocalKustomizations: true  # Use operator embedded manifests

  # Image pull secrets
  imagePullSecrets:
    - name: edb-cred
      namespace: edbpgai-bootstrap

  # Global parameters
  globalParameters:
    CONTAINER_REGISTRY_URL: docker.enterprisedb.com/pgai-platform
    internal_backup_folder: "abc123xyz789"
    portal_domain_name: "portal.my-company.com"
    trust_domain: "portal.my-company.com"
    dms_domain_name: "migration.my-company.com"
    storage_class: "gp3"
    portal_certificate_issuer_kind: "ClusterIssuer"
    portal_certificate_issuer_name: "letsencrypt-prod"
    load_balancer_mode: "public"  # or "private" or "disabled"
    load_balancer_provider: "aws"  # or "gcp", "azure". Leave empty for self-managed␣
↪LB
    # node_port_domain: "pg.my-company.com"  # Required only if load_balancer_mode is
↪"disabled"

  # Component-specific parameters
  componentsParameters:
    upm-beacon:
      beacon_location_id: "us-east-1"
      server_host: "beacon.my-company.com"
    transporter-rw-service:
      domain_name: "migration.my-company.com"
    transporter-dp-agent:
      rw_service_url: "https://migration.my-company.com/transporter"
    upm-istio-gateway:
      cookie_aeskey: "your-16-or-32-byte-key"
    transporter-data-operator:
      fips_enabled: "false"

  # Beacon server configuration
  beaconServer:
    logLevel: info
    hybridManagerInstallation: "my-company-prod"
    additionalTrustDomains:
      - "secondary-location.my-company.com"
```

(次のページに続く)

(前のページからの続き)

```
# Beacon agent configuration
beaconAgent:
  accessKey: "your-access-key"
  logLevel: info
  providers:
    - appliance
  provisioning:
    imageDiscovery: true
    imagesetDiscoveryAuthenticationType: "basic"
    imagesetDiscoveryContainerRegistryURL: "docker.enterprisedb.com/pgai-platform"
    openshift: false
    operandSetVersions:
      - "1\.1\.[0-9]+"
    nodeSelector:
      kubernetes.io/os: linux
    topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: "DoNotSchedule"
        podSelector: "MatchDatabasePods"
  transparentDataEncryptionMethods:
    - passphrase
    - hashicorp_vault

# Cluster groups (for multi-DC deployments)
clusterGroups:
  role: standalone  # or primary/secondary
  # For multi-DC primary:
  # primary:
  #   domainName: "primary.my-company.com"
  # secondaries:
  #   - domainName: "secondary.my-company.com"

# Installation scenarios
scenarios:
  - core
  - migration
  - ai
  - analytics
  - dbaas

# Optional: Disable specific components
# disabledComponents:
#   - component-name
```

(次のページに続く)

(前のページからの続き)

```
  # Optional: provision an internal container registry (not for use except on fully␣
↪air-gapped clusters!)
  # hmRegistryRef:
  #   name: hm-registry

  # Optional: Resource annotations
  resourceAnnotations:
    - name: istio-ingressgateway
      kind: Service
      annotations:
        service.beta.kubernetes.io/aws-load-balancer-scheme: internal
        service.beta.kubernetes.io/load-balancer-source-ranges: "10.0.0.0/8"
```

EDB Postgres AI Operator のインストール

EDB Helm リポジトリを追加します

EDB Helm チャートリポジトリを登録し、ローカルキャッシュを更新します。<EDB_SUBSCRIPTION_TOKEN> を EDB Repos 2.0 トークンに置き換えます。

```
helm repo add enterprisedb-edbpgai \
  "https://downloads.enterprisedb.com/${EDB_SUBSCRIPTION_TOKEN}/pgai-platform/helm/
↪charts/"
helm repo update
```

名前空間を作成する

オペレーターの専用の名前空間を作成します。

```
kubectl create namespace edbpgai-bootstrap --dry-run=client -o yaml | kubectl apply -
↪f -
__OUTPUT__
namespace/edbpgai-bootstrap created
```

オペレーターヘルムチャートのインストール

`edbctl hm upgrade-operator` を使用してオペレーターをインストールします。このコマンドは、ターゲットレジストリで使用可能なオペレーターイメージタグをリストし、最新のものを選択し、チャートをインストールまたはアップグレードします。オペレーターバージョンを手動で選択する必要はありません。HM 1.4 から、オペレーターは HM リリースケイデンスから分離され、単一のオペレーターバージョンがマルチプルの HM ラインをサポートします。

推奨されるパスは、最初に HM アーティファクトオペレーターイメージ+チャートを含むをプライベートレジストリにミラーリングすることです *Syncing images to a local private registry* を参照してください - 次に、そのプライベートレジストリからオペレーターをインストールすることです。同期ステップ `LOCAL_REGISTRY_URI`、`LOCAL_REGISTRY_USER`、`LOCAL_REGISTRY_PWD` で定義されたのと同じ変数を再利用しファイル。

```
edbctl hm upgrade-operator \
  --registry-uri "${LOCAL_REGISTRY_URI}" \
  --registry-username "${LOCAL_REGISTRY_USER}" \
  --registry-password "${LOCAL_REGISTRY_PWD}"
__OUTPUT__
Latest HM Operator version found in "${LOCAL_REGISTRY_URI}": 2.0.0
Namespace "edb-hcp-operator-system" does not exist, creating it...
namespaces/edb-hcp-operator-system created
Creating image pull secret "edb-hcp-operator-pull-secret" in namespace "edb-hcp-
↪operator-system"...
secrets/edb-hcp-operator-pull-secret created
Installing HM Operator chart "edb-hcp-operator" (version 2.0.0) into namespace "edb-
↪hcp-operator-system"...
HM Operator chart "edb-hcp-operator" is now at version 2.0.0 in namespace "edb-hcp-
↪operator-system".
```

プライベートレジストリが自己署名証明書を使用する場合、--registry-insecure を追加します。追加の Helm 値を渡すたとえば、クラスター内 HM Registry コンポーネントを有効にするには、 --set を使用します。

```
edbctl hm upgrade-operator \
  --registry-uri "${LOCAL_REGISTRY_URI}" \
  --registry-username "${LOCAL_REGISTRY_USER}" \
  --registry-password "${LOCAL_REGISTRY_PWD}" \
  --set hmRegistry.enabled=true
```

プライベートレジストリを操作しない場合は、docker.enterprisedb.com/pgai-platform で--registry-uri を指定し、EDB Repos 2.0 トークンを--registry-password --registry-username pgai-platform を使用して渡すことにより、EDB Repos 2.0 から直接プルすることもできます。

オペレーターのインストールの確認

edbctl hm upgrade-operator は、戻る前にオペレーターポッドの準備ができるまで待機します。ダブルチェックしたい場合

```
kubectl get pods -n edb-hcp-operator-system
```

次のような出力が表示されるはずです。

```
NAME                                  READY   STATUS    RESTARTS   AGE
edb-hcp-operator-xxxxxxxxxx-xxxxx   1/1     Running   0          30s
```

手動 Helm フォールバック

edbctl < 1.7.0 がまだある環境の場合、以前の手動 Helm フローは引き続き機能します。*Syncing images to a local private registry* LOCAL_REGISTRY_URI 、LOCAL_REGISTRY_USER 、LOCAL_REGISTRY_PWD で定義されたものと同じプライベートレジストリ変数を再利用し

```
kubectl create namespace edbpgai-bootstrap --dry-run=client -o yaml | kubectl apply -
↪f -

#  Create a pull secret for your private registry:

kubectl create secret docker-registry edb-cred \
  --namespace edbpgai-bootstrap \
  --docker-server="${LOCAL_REGISTRY_URI}" \
  --docker-username="${LOCAL_REGISTRY_USER}" \
  --docker-password="${LOCAL_REGISTRY_PWD}" \
  --dry-run=client -o yaml | kubectl apply -f -

#  Install the operator using Helm:

helm template \
  --namespace edbpgai-bootstrap \
  --version <OPERATOR_VERSION> \
  --include-crds \
  edb-hcp-operator enterprisedb-edbpgai/edb-hcp-operator \
  --set "controllerManager.manager.image.repository=${LOCAL_REGISTRY_URI}/edb-hcp-
↪operator/manager" \
  --set "controllerManager.manager.image.tag=<OPERATOR_VERSION>" \
  --set "imagePullSecrets[0].name=edb-cred" \
  | kubectl apply -n edbpgai-bootstrap -f -
```

HM 1.4 の<OPERATOR_VERSION> は、2.x リネージの 2.0.0 以降です。インストールする HM リリースでサポートされているバージョンの *Hybrid Manager 2025.11.0 release notes* を確認します。

プライベートレジストリを操作しない場合は、 ${LOCAL_REGISTRY_URI} を docker.enterprisedb.com/pgai-platform に交換し、EDB Repos 2.0 トークンユーザー名 pgai-platform を使用します。

HybridControlPlane リソースを作成する

HM マニフェストの準備

構成を使用して hybridmanager.yaml という名前のファイルを作成します。example above をテンプレートとして使用し、環境の準備に基づいて値をカスタマイズします。

確認する重要なフィールド

1. ``**spec.flavour**`` Kubernetes ディストリビューション
2. ``**spec.imageRegistry**`` コンテナレジストリの URL

3. ``**spec.version**`` インストールする HM バージョン
4. ``**spec.globalParameters.portal_domain_name**`` ポータルの FQDN
5. ``**spec.beaconAgent.provisioning**`` プロビジョニング構成

HM リソースを適用する

ハイブリッドコントロールプレーンを展開します。

```
kubectl apply -f hybridmanager.yaml
__OUTPUT__
hybridcontrolplane.edbpgai.edb.com/edbpgai created
```

インストールの進行状況を監視する

HM リソースが作成されると、オペレーターは HM コンポーネントの調整と展開を開始します。

HM ステータスを確認する

HybridControlPlane リソースのステータスを監視します。

```
kubectl get hybridcontrolplane edbpgai -o yaml
```

status セクションを探します。

```
status:
  phase: deployed  # Can be: "", "deploying", "deployed", "failed", "deleting"
  components:
    - name: component-name
      state: success  # Can be: pending, success, failed, deleted
      version: "2026.1.0"
      timestamp: "2026-03-09T10:30:00Z"
  preflightChecks:
    phase: passed
    totalChecks: 10
    failedChecks: 0
  postflightChecks:
    phase: passed
    totalChecks: 8
    failedChecks: 0
```

kubectl を使用してモニタする

簡単なステータスビューを取得します。

```
kubectl get hcp
```

予想される出力

```
NAME      PHASE      PREFLIGHT   POSTFLIGHT
edbpgai   deployed   passed      passed
```

コンポーネントの展開を確認する

すべてのコンポーネントとそのステータスを表示します。

```
kubectl get hcp edbpgai -o jsonpath={.status.components[*].name} | tr   \n
__OUTPUT__
upm-beacon
upm-istio-gateway
transporter-rw-service
transporter-dp-agent
transporter-data-operator
```

障害が発生したコンポーネントを確認します。

```
kubectl get hcp edbpgai -o jsonpath={.status.components[?(@.state=="failed")]}
```

出力が返されない場合、すべてのコンポーネントは正常に展開されました。

ポストフライトをチェック

HM のポストフライトステータスを確認します。

```
kubectl get postflights.edbpgai.edb.com edbpgai -o yaml
__OUTPUT__
...
status:
  checkResults:
  - description: Check whether all pods belonging to the Hybrid Manager are ready.
    name: PodsChecker
    state: Passed
  - description: Check whether all internal databases belonging to the Hybrid Manager
      are ready.
    name: DBsChecker
    state: Passed
  - description: Check whether the latest Velero Backup has been completed.
    name: VeleroChecker
    state: Passed
  - description: Check whether all nodes are ready.
    name: NodesChecker
    state: Passed
  - description: Check whether all certificates belonging to the Hybrid Manager are
      valid.
    name: CertificatesChecker
```

(次のページに続く)

(前のページからの続き)

```
    state: Passed
  lastTransitionTime: "2026-01-27T08:50:25Z"
  phase: Healthy
```

探すべきもの

- ポストフライトは`→ Healthy`から移行する必要があります
- すべての`.status.checkResults`には`Passed`状態が必要です

> **注釈**
>
> HM フェーズが`failed`を示している場合、エラーメッセージのコンポーネントステータスを確認します。ポストフライトチェックが失敗した場合は、特定の失敗の理由について`checkResults`を確認します。 !!!

ハイブリッドコントロールプレーンの管理

構成の更新

HM 構成を更新するには

1. `hybridmanager.yaml`マニフェストを編集します
2. 変更を適用します。

```
kubectl apply -f hybridmanager.yaml
__OUTPUT__
hybridcontrolplane.edbpgai.edb.com/edbpgai configured
```

1. オペレーターは変更を調整および適用します

オペレーターログの確認

問題が発生した場合は、オペレーターログを確認してください。

```
kubectl logs -n edb-hcp-operator-system deployment/edb-hcp-operator-controller-
↪manager -f
```

HM バージョンのアップグレード

`edbctl hm upgrade-operator`レジストリで利用可能な最新のオペレーターバージョンを取得します。バージョンを手動で設定する必要はありません。次に、`HybridControlPlane`マニフェストの`spec.version`フィールドをターゲット HM バージョンと`kubectl apply`に更新します。オペレーターは HM リリースケイデンスから分離されているため、同じ`2.x`オペレーターがさまざまな HM バージョンをカバーします。アップグレード先の HM リリースでサポートされているオペレーターバージョンについては、*Hybrid Manager 2025.11.0 release notes* を参照してください。

ポータルアクセスの検証

インストールが正常に完了したら、HM コンソールにアクセスできることを確認します。

1. **ブラウザーを開き**、フェーズ *4* 環境の準備 中に定義したポータルドメインたとえば `https://portal.my-company.com` に移動します。
2. **証明書を受け入れます。**自己署名証明書を使用した場合、またはカスタム CA をまだ信頼していない場合、ブラウザーに警告が表示される場合があります。とりあえずそれを通り過ぎてください。
3. **ログイン** 環境の準備中に構成した **User-0** 資格情報を使用します。

成功 HM コンソールのランディングページが表示されます。

次のフェーズ

プラットフォームがインストールされ、実行されています。これで、最初の自己管理クラスターの接続やチームメンバーの追加など、インストール後の初期構成を実行する準備が整いました。

** *Proceed to Phase 6: Exploring your post-installation options* →**

4.1.1.8.6 Exploring your post-installation options

概要

> **ロール:** インフラストラクチャエンジニア/プラットフォーム管理者

前提条件

- フェーズ *1* アーキテクチャの計画 完成品
- フェーズ *2* システム要件の収集 完成品
- フェーズ *3Kubernetes* クラスターの展開 実行および検証済み
- フェーズ *4* 環境の準備 完了および検証済み
- フェーズ *5* ハイブリッドマネージャーのインストール 完了&検証済み

*ハイブリッドマネージャー HM ポータルへの管理者アクセス。

結果

** セキュリティ統合、AI 機能、モニタリングが有効になった完全に構成された HM 環境。*

次のフェーズ *Using Hybrid Manager*

HM コントロールプレーンが実行されたため、オプションだが重要な統合を有効にできます。プラットフォームはこれらなしでも機能しますが、これらを構成することにより、環境がセキュリティ、AI ワークロード、可観測性の点で実稼働の準備が整います。

ID プロバイダー IdP の構成

HM コンソールを介して *Configure your IdP integration* を使用して、ユーザーの SSO を有効にします。これは、運用環境で安全なアクセス管理を保証するための重要な手順です。

AI 機能のロックを解除する 完全な GenAI Builder セットアップ

- ***Guide: Gen AI Builder*** — `ai` インストールシナリオが必要ですデフォルトで有効。

通知を設定する オブザーバビリティ

重要なイベントクラスターフェールオーバー、バックアップの障害、または容量の問題などについてチームに警告を発します。HM では、通知チャンネルをグローバルまたはプロジェクトごとに構成できます。

サポートされているチャンネル

- 電子メール
- Webhook Slack、Microsoft Teams、汎用 JSON
- PagerDuty
- ***Guide: Notifications***

外部データベースを監視する オブザーバビリティ

HM は、管理していない Postgres データベース例 RDS、自己管理 EC2、またはオンプレミスインスタンスをモニタできます。

これを有効にするには、外部データベースホストに **EDB Postgres AI エージェント** をインストールする必要があります。

エージェントはメトリックとログを収集し、一元的な可視化のために HM コントロールプレーンに戻します。

- ***SOP: Monitor External Databases***

移行コパイロットを有効にする

`migration` インストールシナリオが必要ですデフォルトで有効。

AI 支援の **移行コパイロット**を使用するには、特定のシークレットと構成をコントロールプレーンに適用する必要があります。

この機能は、スキーマの分析と古いデータベースから EDB Postgres への変換を支援します。

- ***SOP: Enable Migration Co-pilot***

次のステップ

HM プラットフォームが完全に展開され、統合されました。ユーザーをオンボードし、クラスターのプロビジョニングを開始する準備が整いました。

** *Using Hybrid Manager* に進む →**

4.1.2 Configuring SOPs

4.1.2.1 Syncing images to a local private registry

関連するインストールフェーズ *Phase 4: Preparing the Environment*

4.1.2.1.1 前提条件

- **`edbctl CLI tool** <Hybrid Manager CLI - edbctl>` がインストールおよび構成されています。
- 有効な **EDB Repos 2.0 Token** 。
- ターゲットローカルプライベートレジストリへの書き込みアクセス例 AWS ECR、Google Artifact Registry、Harbor 。

イメージを内部的にホスティングすると、次のことが保証されます。

- **セキュリティとコンプライアンス**アーティファクトのスキャンと承認を制御します。
- **信頼性**展開は、外部のインターネット接続またはサードパーティのアップタイムに依存しません。
- **パフォーマンス** イメージをプルするクラスターノードの遅延の削減。

このプロセスでは、 `edbctl` ツールを使用して、SHA256 ダイジェストを保存しながら EDB Repos 2.0 からプライベートレジストリにアーティファクトをコピーします。

HM のソフトウェアスタックは EDB Repos 2.0 レジストリにプッシュされ、ローカルプライベートレジストリで使用するアーティファクトを提供します。RHOS または Rancher RKE2 オンプレミスシナリオの顧客管理の内部レジストリ、またはクラウドサービスの自己管理レジストリ CSP シナリオでの HM のプロバイダー CSP レジストリ AWS Elastic Container Registry ECR、 Google Cloud Artifact Registry GAR、または Azure Container Registry ACR 。

開始する前に、コンテナイメージをホストする準備ができている安全で承認されたプライベートレジストリがあることを確認します。

レジストリの URI と、書き込みアクセスを持つユーザー名とパスワードまたはトークンが必要です。さらに、インストールする EDB Postgres AI の特定のバージョンを確認します。

これらの資格情報を使用して、同期プロセスは、必要なすべてのアーティファクトを EDB Repos 2.0 からローカルのプライベートレジストリに直接コピーし、Helm チャートのインストールまたはアップグレードで利用できるようにします。

4.1.2.1.2 同期変数の構成

必須情報

- **リリースバージョン**インストールするハイブリッドマネージャー HM のタグ例 `v2026.5.1` 。
- **EDB Token:** `docker.enterprisedb.com` のアクセストークン。
- **ローカルレジストリ** 組織のレジストリの URI と資格情報。

1. EDB リリースバージョンを定義する

```
export EDBPGAI_RELEASE=<EDB-pgai-release-version>
```

2. EDB 資格情報を定義しますソース

```
export CS_EDB_TOKEN=<your-edb-repos-token>
export EDB_SOURCE_REGISTRY=pgai-platform
```

3. プライベートレジストリ資格情報の定義 宛先

```
export LOCAL_REGISTRY_URI=<your_local_container_registry_uri>
export LOCAL_REGISTRY_USER=<your_local_registry_user>
export LOCAL_REGISTRY_PWD=<your_local_registry_password>
```

クラウドレジストリ AWS ECR、Google Artifact Registry、または Azure ACR を使用している場合、ローカル環境が `aws ecr get-login-password` などを介して認証され、`LOCAL_REGISTRY_PWD` が有効なトークンを反映していることを確認します。

次のコマンドを実行して、プラットフォームイメージとオペレーターアーティファクトをワンステップで同期します。

```
edbctl image sync-to-local-registry \
    --destination-registry-uri "${LOCAL_REGISTRY_URI}" \
    --version "${EDBPGAI_RELEASE}" \
    --source-registry-username "${EDB_SOURCE_REGISTRY}" \
    --source-registry-password "${CS_EDB_TOKEN}" \
    --destination-registry-username "${LOCAL_REGISTRY_USER}" \
    --destination-registry-password "${LOCAL_REGISTRY_PWD}"
```

このコマンドは、次のアーティファクトをプライベートレジストリにミラーリングします。

- **プラットフォームイメージ** —ポータル、ビーコン、トランスポーター、およびその他の HM コンポーネント。
- **オペレーターマネージャーイメージ** — `edb-hcp-operator` コントローラーイメージ。
- **オペレーターバンドルイメージ** —OCP 認定バンドル、エアギャップ OpenShift クラスターでセルフホストの OperatorHub カタログをビルドするために使用されます。
- **オペレーターヘルムチャート** — `edb-hcp-operator` チャート、画像とともに OCI アーティファクトとしてミラーリングされます。

4.1.2.1.3 構成の更新

同期が完了したら、プライベートレジストリからイメージをプルするように HM を構成する必要があります。

`HybridControlPlane` CR を編集して、`spec.imageRegistry` をプライベートレジストリ URI に設定します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  imageRegistry: "<your_local_container_registry_uri>"
  # ... your other spec fields
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

4.1.2.1.4 次のステップ

イメージが同期され、構成が更新されたら、インストールを続行する準備が整います。

** フェーズ *4* 環境の準備 または フェーズ *5* ハイブリッドマネージャーのインストール に戻ります。**

4.1.2.2 Using the in-cluster container registry

関連するインストールフェーズ フェーズ *4* 環境の準備

4.1.2.2.1 前提条件

- フェーズ *2* システム要件の収集 を満たす実行中の Kubernetes クラスター。
- パブリック IP のないプライベートサブネットに展開されたすべてのクラスターノード *Security considerations for Langflow* を参照してください。
- クラスターに既にインストールされている EDB Postgres AI オペレーター *EDB Postgres AI Operator のインストール* を参照してください。
- :ref:``edbctl` を使用して移行ポータルシークレットを作成する <*edbctl* を使用して移行ポータルシークレットを作成する>` は、管理ワークステーションにインストールおよび構成されています。
- `kubectl` はクラスターに対してインストールされ、認証されました。
- `yq` コマンドライン YAML プロセッサー。

EDB Postgres AI オペレーターは、`HMRegistry` カスタムリソース定義で定義されたコンテナレジストリを埋め込み、オペレーター自分自身と一緒に実行できます。レジストリを有効にするために、オペレーターイメージを超えて追加のアーティファクトを同期する必要はありません。

このレジストリは、HM の外部のプライベートレジストリがない、または維持したくない**エアギャップまたは完全な内部展開**に使用します。オペレーターをインストールした後、EDB イメージをクラスター内レジストリに一度同期すると、プラットフォームの残りの部分がそこからプルします。

セキュリティと操作要件を満たす HM の外部のプライベートレジストリがある場合は、クラスター内レジストリよりも優先します。そのフローについては、*Syncing images to a local private registry* を参照してください。

4.1.2.2.2 セキュリティに関する考慮事項

> **警告**
>
> クラスター内レジストリは、認証なしで**プレーン HTTP**を介してイメージを提供し、 *NodePort* を介してクラスターノードに自分自身を公開します。

この設定は、**すべてのクラスターノードがパブリック IP のないプライベートサブネットにある**場合にのみ安全です。ノードがパブリック IP を持つ場合、レジストリとそれが保持するすべてのイメージは、資格情報なしでインターネットから到達可能です。

ノードにパブリック IP がある場合は、クラスター内レジストリを使用しないでください。*Syncing images to a local private registry* に文書されているように、HM の外部のプライベートレジストリを使用します。

`HMRegistry` は、ハイブリッドマネージャー HM ワークロードを対象としたクラスター内 Open Container InitiativeOCI イメージレジストリを展開するカスタムリソースです。オペレーターは、レジストリの ConfigMap、Deployment、PVC、および Service をプロビジョニングし、OpenShift では `image.config.openshift.io/cluster` にもパッチを適用して、クラスターのコンテナランタイムがレジストリの NodePort / ClusterIP エンドポイントを安全でないソースとして信頼します。

`hm-registry.yaml` という名前のローカルファイルを作成し、次のコンテンツを追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HMRegistry
metadata:
  name: hm-registry
  annotations:
    biganimal.enterprisedb.io/deletion-protect: "enabled"
spec:
  mode: Insecure
  resources:
    requests:
      cpu: 250m
      memory: 512Mi
    limits:
      cpu: 1000m
      memory: 4Gi
  storage:
    pvc:
      size: 120Gi
```

次に、実行します。

```
kubectl apply -f hm-registry.yaml
```

次の方法で HMRegistry の作成を追跡できるようになりました。

```
kubectl get hmregistries.edbpgai.edb.com
```

完了すると、これに似た出力が表示されます。

```
$ kubectl get hmregistries.edbpgai.edb.com hm-registry
NAME          PHASE   MESSAGE               CRI URI           KUBE URI
hm-registry   Ready   HMRegistry is ready   localhost:30183   hm-registry-service.
↪edbpgai-bootstrap.svc.cluster.local:5000
```

そして、オペレーターは、対応する ConfigMap、Deployment、PVC、およびサービスを edbpgai-bootstrap 名前空間で管理します。 これらは次の経由で表示できます。

```
kubectl get cm,deployment,pods,pvc,svc -n edbpgai-bootstrap
```

4.1.2.2.3 HMRegistry への画像のロード

必須情報

- **リリースバージョン**インストールするハイブリッドマネージャー HM のタグ例 v1.4.0 。
- **EDB Token:** docker.enterprisedb.com のアクセストークン。

1. EDB リリースバージョンを定義する

```
export EDBPGAI_RELEASE=<EDB-pgai-release-version>
```

2. 別のシェルで、次を実行します

```
kubectl port-forward svc/hm-registry-service 5000:5000 -n $EDB_OPERATOR_NAMESPACE
```

3. EDB 資格情報を定義しますソース

```
export CS_EDB_TOKEN=<your-edb-repos-token>
export EDB_SOURCE_REGISTRY=pgai-platform
```

4. 画像を HMRegistry に同期する

```
edbctl image sync-to-local-registry \
    --version "${EDBPGAI_RELEASE}" \
    --source-registry-username "${EDB_SOURCE_REGISTRY}" \
    --source-registry-password "${CS_EDB_TOKEN}" \
    --destination-registry-uri "localhost:5000" \
    --insecure-destination-registry
```

5. ctrl-c を使用して、別のシェルで実行されているポートフォワードを終了します。

4.1.2.2.4 HMRegistry を使用するように HybridControlPlane を構成する

次のように HybridControlPlane カスタムリソースを更新します。

```
kubectl patch hybridcontrolplanes edbpgai --type=merge --patch-file=/dev/stdin <<EOF
  spec:
    hmRegistryRef:
      name: hm-registry
EOF
```

次に、オペレーターは、 `hm-registry` カスタムリソースを自動的に検査してレジストリの `CRI URI` を見つけ、それに応じて `edpbgai` HybridControlPlane リソースの `spec.imageRegistry` キーを更新します。これ以上の構成は必要ありません。

4.1.2.2.5 HM の外部のレジストリに戻す

クラスター内レジストリを使用しない場合は、次のことを行う必要があります。

1. HybridControlPlane リソースから `hm-registry` への参照を削除します

```
kubectl patch hybridcontrolplanes edbpgai --type=json -p [{"op":"remove","path":"/
↪spec/hmRegistryRef"}]
```

2. HybridControlPlane リソースをコンテナレジストリ URI で更新します

```
kubectl patch hybridcontrolplanes edbpgai --type=merge --patch-file=/dev/stdin <<EOF
  spec:
    imageRegistry: <your_container_registry_uri>
EOF
```

Configuring image discovery を参照する必要がある場合があります。

3. クラスター内レジストリの削除保護を無効にする

```
kubectl patch hmregistries.edbpgai.edb.com hm-registry --type=merge --patch-file=/dev/
↪stdin <<EOF
  metadata:
    annotations:
      biganimal.enterprisedb.io/deletion-protect: disabled
EOF
```

4. `hm-registry` HMRegistry インスタンスを削除します

edb_notranlate_13 これは、 `edbpgai-bootstrap` 名前空間のクラスター内 HMRregistry に関連付けられているすべてを破壊します ConfigMap、Deployment、PVC、および Services。

4.1.2.3 Configuring image discovery

関連するインストールフェーズ *Phase 4: Preparing the Environment*

4.1.2.3.1 前提条件

- ***Synced images to your local container registry*** 完了済 - EDB イメージが設定されたプライベートレジストリが必要です。
- **レジストリコンプライアンス** レジストリは OCI compliant である必要があります。
- **権限** IAM ロール AWS / GCP またはレジストリアクセス制御オンプレミスを構成するためのアクセス。

概要

イメージ検出 は、**HM 内部ビーコンエージェント**内で実行されているバックグラウンドプロセスです。プライベートコンテナレジストリに接続し、利用可能な Postgres バージョンタグをスキャンし、ハイブリッドマネージャー HM に登録します。

イメージ検出により、新しいデータベースバージョンをレジストリに同期すると、ユーザーが手動介入なしでプロビジョニングに自動的に利用できるようになります。

イメージ検出の構成

この機能を有効にするには、レジストリの詳細を使用してビーコンエージェントを構成する必要があります。

`HybridControlPlane` CR を編集して、`spec.beaconAgent` の下にイメージ検出設定を追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  beaconAgent:
    provisioning:
      imageDiscovery: true
      imagesetDiscoveryContainerRegistryURL: "<your-private-registry-uri>"
      # Set to true only for self-signed/internal CA scenarios
      imagesetDiscoveryAllowInsecureRegistry: false
    # ... your other beaconAgent settings
```

4.1.2.3.2 imageDiscovery

このブールオプションは、イメージ検出機能を有効または無効にします。デフォルトは `false` です。

4.1.2.3.3 imagesetDiscoveryContainerRegistryURL

このオプションは、PostgreSQL コンテナイメージを検出するコンテナレジストリの URL を指定します。

Registry provider	Registry URL	Example
EDB Repo 2.0	*docker.enterprisedb.com/pgai-platform*	*docker.enterprisedb.com/pgai-platform*
Azure Container RegistryACR	*<registry-name>.azurecr.io*	*myregistry.azurecr.io*
Amazon Elastic Container Registry (ECR)	*<aws-account-id>.dkr.ecr.<aws-region-id>.amazonaws.com*	*123456079902.dkr.ecr.us-east-1.amazonaws.com*
Google Artifact Registry GAR	*<gcp-region-id>-docker.pkg.dev*	*us-east1-docker.pkg.dev*
GitLab コンテナレジストリ	*registry.gitlab.com*	*registry.gitlab.com*
MicroK8s Built-in Registry	*localhost:32000*	*localhost:32000*

imagesetDiscoveryContainerRegistryProvider

このオプションは、`imagesetDiscoveryContainerRegistryURL` にあるレジストリのプロバイダーを定義します。デフォルトは `oci` です。

Provider	Description
oci	コンテナレジストリプロバイダーが `/v2/_catalog`エンドポイントを含む完全な OCI compliant である場合に、これを設定します。
gitlab	コンテナレジストリプロバイダーが *GitLab* コンテナレジストリ の場合にこれを設定します。

レジストリプロバイダーごとに推奨されるプロバイダー値は次のとおりです。

Registry provider	Provider value
EDB Repo 2.0	*oci*
Azure Container RegistryACR	*oci*
Amazon Elastic Container Registry (ECR)	*oci*
Google Artifact Registry GAR	*oci*
GitLab コンテナレジストリ	*gitlab*
Harbor Registry	*oci*
MicroK8s Built-in Registry	*oci*

4.1.2.3.4 imagesetDiscoveryAllowInsecureRegistry

このオプションが有効になっている場合、証明書の検証なしで TLS 接続を確立できます。デフォルトは `false` です。

imagesetDiscoveryAuthenticationType

レジストリプロバイダーによってサポートされている認証タイプは次のとおりです。

Registry provider	Supported	Recommended
EDB Repo 2.0	*token*	*token*
Azure Container RegistryACR	*token*, *basic*	*token*
Amazon Elastic Container Registry (ECR)	*eks_managed_identity*	*eks_managed_identity*
Google Artifact Registry GAR	*token*, *basic*	*basic*
GitLab コンテナレジストリ	*token*	*token*
MicroK8s Built-in Registry	*none*	*none*

4.1.2.3.5 image_discovery_credentials_secret_name

この値は、イメージ検出のためのレジストリの資格情報を含む Kubernetes シークレットの名前です。デフォルトは「edb-cred」です。

レジストリ権限の構成

HM 内部ビーコンエージェントには、レジストリをスキャンするには特定の権限が必要です。

次のことができる必要があります。

- リポジトリのリスト
- リストタグ
- タグマニフェストの読み取り

以下からプラットフォームを選択して、必要な権限を構成します。

4.1.2.3.6 汎用/オンプレミス RKE2/OpenShift

標準の OCI レジストリ Harbor、Quay、Artifactory などを使用している場合、**Image Pull Secret** *を作成するときに指定したユーザー名/パスワードに、ターゲットリポジトリに対する `read` および `list` 権限があることを確認します。

レジストリのドキュメント例 Quay.io Permissions を参照して、サービスアカウントを構成します。

4.1.2.3.7 Azure Container RegistryACR

Kubernetes イメージプルシークレットとして使用される ACR トークンには、_repositories_pull_metadata_read スコープマップが必要です。

```
az acr token create --name <token-name> --registry <registry-name> \
    --scope-map "_repositories_pull_metadata_read"
```

4.1.2.3.8 AWS Elastic Container Registry ECR

ECR を使用する EKS で実行されている場合、 **EKS マネージド Identity** が必要な認証方法です。

AmazonEC2ContainerRegistryReadOnly ポリシーで IAM ロールを作成し、ビーコンのサービスアカウントに関連付ける必要があります。

このスクリプトを実行して、IAM ロールとポッド ID を構成します。

```
#  Configuration Variables

EKS_CLUSTER_NAME="<eks_cluster_name>"
EKS_CLUSTER_REGION="<eks_cluster_region>"
IMAGE_DISCOVERY_IAM_ROLE_NAME="<iam_role_name>"

#  1. Create Trust Policy

cat <<EOF > ./image-discovery-trust.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF

#  2. Create Role & Attach Policy
```

(次のページに続く)

(前のページからの続き)

```
aws iam create-role --role-name "${IMAGE_DISCOVERY_IAM_ROLE_NAME}" \
    --assume-role-policy-document file://image-discovery-trust.json

aws iam attach-role-policy --role-name "${IMAGE_DISCOVERY_IAM_ROLE_NAME}" \
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly

#  3. Associate with EKS Service Account

IMAGE_DISCOVERY_IAM_ROLE_ARN=$(aws iam get-role --role-name ${IMAGE_DISCOVERY_IAM_
↪ROLE_NAME} | jq -r .Role.Arn)

aws eks create-pod-identity-association --cluster-name "${EKS_CLUSTER_NAME}" \
    --namespace upm-beacon \
    --service-account upm-beacon-agent-k8s \
    --role-arn "${IMAGE_DISCOVERY_IAM_ROLE_ARN}" \
    --region "${EKS_CLUSTER_REGION}"
```

4.1.2.3.9 パッケージインストールの追加構成

パッケージをダウンロードして展開するために **パッケージのインストール**を許可するには、**kapp-controller** コンポーネントへの ECR アクセスを許可する必要があります。

kapp-controller は、アプリケーションの展開中に ECR から Helm チャートとパッケージマニフェストを取得します。この構成がないと、パッケージのインストールは資格情報エラーで失敗します。

kapp-controller のポッド ID の関連付けを追加します。

```
#  Use the same IAM role ARN from above

aws eks create-pod-identity-association --cluster-name "${EKS_CLUSTER_NAME}" \
    --namespace kapp-controller \
    --service-account kapp-controller-sa \
    --role-arn "${IMAGE_DISCOVERY_IAM_ROLE_ARN}" \
    --region "${EKS_CLUSTER_REGION}"
```

4.1.2.3.10 Google Artifact Registry GAR

GKE で実行されている場合、ビーコンが使用するサービスアカウントには、プロジェクトをリストし、アーティファクトを読み取る権限が必要です。

必要なロール

- `roles/artifactregistry.reader` 画像を読み取るため
- `roles/browser` 特にリポジトリを見つけるための `resourcemanager.projects.list`

このスクリプトを実行して、IAM バインディングを適用します。

```
#  Replace with your Project ID and Service Account Name

PROJECT_ID="<your-project-id>"
SERVICE_ACCOUNT_EMAIL="<service-account-name>@${PROJECT_ID}.iam.gserviceaccount.com"

#  Grant Reader Role

gcloud projects add-iam-policy-binding ${PROJECT_ID} \
    --member="serviceAccount:${SERVICE_ACCOUNT_EMAIL}" \
    --role="roles/artifactregistry.reader"

#  Grant Browser Role (Required for discovery)

gcloud projects add-iam-policy-binding ${PROJECT_ID} \
    --member="serviceAccount:${SERVICE_ACCOUNT_EMAIL}" \
    --role="roles/browser"
```

4.1.2.3.11 GitLab コンテナレジストリ

Kubernetes イメージプルシークレットとして使用される Personal access token には、次のスコープが必要です。

- `read_registry` —プライベートプロジェクトのコンテナレジストリイメージへの読み取り専用アクセスを許可します。
- `read_api` —プロジェクトとコンテナリポジトリへの読み取りアクセスを許可します。

詳細については、 GitLab Personal access token scopes を参照してください。

> **注釈**
>
> *legacy* トークンのみがサポートされています。 !!!

Kubernetes Secret の構成

最後のステップは、どの Kubernetes Secret にレジストリ資格情報が含まれているかを HM が知っていることを確認することです Cloud Identity を使用しない場合。

`spec.componentsParameters.upm-beacon` の下の `HybridControlPlane` CR の資格情報のシークレット名を確認または更新します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
```

(次のページに続く)

(前のページからの続き)

```
  componentsParameters:
    upm-beacon:
      image_discovery_credentials_secret_name: "edb-cred"
    # ... your other componentsParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

検証

これらの変更を適用した後、**HM-internal Beacon agent** ポッドのログを確認します。レジストリをスキャンし、イメージタグを検出していることを示すメッセージが表示されます。

4.1.2.4 Customizing secrets

デフォルトでは、Hybrid ManagerHM には、事前に構成されたシークレットが含まれており、すぐに使えるインストールが簡単にできます。ただし、これらのデフォルトのシークレットは、運用環境での使用を目的としたものではありません。ライブ環境では、カスタムシークレットを作成して、インストールが安全で一意なままであることを確認する必要があります。

HM シークレットは、インストールシナリオによって分類されています。どのインストールシナリオを有効にしているかに応じて、以下の対応するシークレットを構成する必要があります。

Scenario	Feature	Key secrets	Docs
ai	GenAI Builder	Fernet key, NVIDIA NGC credentials	*Customizing GenAI Builder secrets*
analytics	Catalog	Confounding key	*Customizing Catalog secrets*
migration	Migration Portal	DB accounts, Copilot HTTP/Metrics auth	*Customizing Migration Portal secrets for secure internal communication*

4.1.2.4.1 Customizing GenAI Builder secrets

`ai` インストールシナリオが有効になっている展開に必要です。これらのシークレットは、AI エージェントのモデルのプルとデータの暗号化を処理します。

NGC API キー NVIDIA NIM モデルをプルするために必要です。

Fernet キー GenAI スタック内の機密データの対称暗号化に使用されます。

オブジェクトストレージ DataLake 用の S3 互換バケット AWS、GCS などが必要です。

GenAI Builder シークレットの作成

!!!note ai インストールシナリオが有効になっている展開に必要です。このシナリオは、シナリオ構成 HybridControlPlane CR の spec.scenarios を介して手動で除外されない限り、デフォルトで含まれます。

NVIDIA NGC documentation の後に 1 つを作成して、モデルイメージのプルを有効にします。

1. edbctl CLI を使用して、Fernet キーシークレットと NGC キーシークレットを作成できます。

- 手動インストールの場合、次のコマンドを実行し、対話型プロンプトに従います。

```
edbctl hybrid-manager create-install-secrets --version <version> --scenario ai
```

- CI / CD パイプラインを介してインストールを実行している場合、対話型プロンプトを抑制する必要があります。これを行う方法は、edbctl バージョンによって異なります。

1. 上記で作成した NGC API キーが変数として利用できることを確認します。

2. 非対話型動作用に edbctl を構成します。

```
edbctl config set interactive_mode off
edbctl config set confirm_mode off
```

1. setup コマンドを実行します。

```
edbctl hybrid-manager create-install-secrets --version <version> --scenario ai
```

これにより、Fernet キーシークレットと、適切なレプリケーションアノテーションを使用して default 名前空間に nvidia-nim-secrets および ngc-cred シークレットが作成されます。

!!!note Fernet は、Python で使用される暗号化ライブラリです。対称的な暗号化/復号化を提供し、機密データを保存するために必要です。 .. ::

1. HM 管理者は、Fernet キーを安全に保管し、バックアップする必要があります。シークレット名と名前空間は、実行しているハイブリッドマネージャーのバージョンによって異なります。

バージョン 2026.2 以前の場合、ソリューションはグリップテープに基づいています。次のコマンドを使用して、シークレットを取得します。

```
kubectl get secret -n upm-griptape fernet-secret -o yaml
```

バージョン 2026.3 以降では、ソリューションは LangFlow に移行しました。シークレットはデフォルトの名前空間に保存されます。

```
kubectl get secret langflow-secret -n default -o yaml
```

キーは安全に保管してください。

1. ハイブリッドマネージャー展開に使用しているオブジェクトストレージに DataLake バケットを作成することにより、GenAI ビルダーの DataLake オブジェクトストレージを構成します。GenAI Builder は、それを使用して、構造、ツール、インデックス付きデータを保存します。

```
aws s3 mb s3://<your-datalake-bucket-name> -region     <your-region>
```

```
gsutil mb -l <your-region> gs://<your-datalake-bucket-name>
```

プロバイダーの管理コンソールまたは CLI を使用して、DataLake バケットの一意の名前を持つバケットを作成します。

1. バケットの次の情報をキャプチャします。後で GenAI ランチパッドアプリケーションを初めて使用するときに必要になります。コンソールには、DataLake バケットの構成の入力が求められます。これには、以下が必要です。
    - `DATA_LAKE_ROOT_BUCKET` DataLake で使用するバケットの名前。
    - `DATA_LAKE_S3_ACCESS_KEY` DataLake バケットへの接続に使用される access_key
    - `DATA_LAKE_S3_SECRET_ACCESS_KEY` DataLake バケットへの接続に使用される secret_access_key
    - `DATA_LAKE_S3_ENDPOINT_UR` DataLake バケットへの接続に使用されるエンドポイント URL
2. CORS 構成でバケットの設定を更新します。

バケットの設定を更新して、次の CORS configuration ます。

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE",
            "GET",
            "HEAD"
        ],
        "AllowedOrigins": [
            "https://${PORTAL_DOMAIN_NAME}"
        ],
        "ExposeHeaders": []
    }
]
```

`https://${PORTAL_DOMAIN_NAME}` は、ハイブリッドマネージャー用に構成されたドメインです。

GCS の S3 interoperability layer により、GenAI Builder は S3 互換オブジェクトストアとして GCS を使用できます。 1. GCS コンソールの **設定** の下で、s3 の相互運用性をオンにします。2. ストレージ管理者およびサービスアカウントトークン作成者ロールを使用して更新または create a service account します。3. サービスアカウントの HMAC key pair を作成します。4. ハイブリッドマネージャーエンドポイントをポイントする CORS 構成を使用して構成ファイルを作成します。

edb_notranlate_8 `https://${PORTAL_DOMAIN_NAME}` は、ハイブリッドマネージャー用に構成されたドメインです。 5. CORS 構成を、前に作成したバケットに適用します。

edb_notranlate_9 プロバイダーの管理コンソールまたは CLI を使用して、ハイブリッドマネージャーでクロスオリジンリソース共有 CORS を構成します。

4.1.2.4.2 Customizing Catalog secrets

`analytics` インストールシナリオが有効になっている展開に必要です。

交絡キーストレージ資格情報の暗号化に使用される重要なランダム化文字列。

警告このキーを失うと、管理対象レイクハウスデータにアクセスできなくなります。`pg-confounding-key` シークレットは常にバックアップしてください。

カタログシークレットの作成

!!!note `analytics` インストールシナリオが有効になっている展開に必要です。このシナリオは、シナリオ構成 `HybridControlPlane` CR の `spec.scenarios` を介して手動で除外されない限り、デフォルトで含まれます。

- 手動インストールの場合、次のコマンドを実行し、対話型プロンプトに従います。

edb_notranlate_0 - CI / CD パイプラインを介してインストールを実行している場合、対話型プロンプトを抑制する必要があります。これを行う方法は、`edbctl` バージョンによって異なります。

1. 非対話型動作用に `edbctl` を構成します。

```
edbctl config set interactive_mode off
edbctl config set confirm_mode off
```

1. setup コマンドを実行します。

edb_notranlate_2 !!!note-交絡キーは、少なくとも 32 バイト長のランダム化された文字列です。 - ハイブリッドマネージャー展開ごとに交絡キーを作成します。 .. ::

1. Hybrid Manager 管理者は、交絡キーを安全に保管し、バックアップする必要があります。

> **⚠ 警告**
>
> ディザスターシナリオでの交絡キーの損失は、Hybrid Manager データカタログで管理される Lakehouse データにアクセスするためのメカニズムが存在しない状況をもたらします。代わりに、管理者は新しいキーを作成して保存し、*upm-lakekeeper /lakekeeper* ワークロードを再起動し、既存のすべてのデータカタログを削除せずに注意深くリビルドする必要があります。この手順は非常に危険であり、EDB PG AI プロフェッショナルサービスチームのサポートが必要です。 キーを取得します。

`shell kubectl get secrets -n upm-lakekeeper pg-confounding-key -o yaml` 鍵は安全に保存します。

4.1.2.4.3 Customizing Migration Portal secrets for secure internal communication

migration インストールシナリオが有効になっている展開に必要です。これらのシークレットは、移行ポータル、AI コパイロット、および基になるデータベース間の内部通信を管理します。

edbctl はこれらを自動的に生成できますが、運用環境では多くの場合、特定のサービスアカウントの手動オーバーライドが必要です。

データベースアカウント内部サービスユーザーとスーパーユーザーの資格情報。

コパイロット認証安全なコンポーネント相互通信のための HTTP およびメトリック認証。

標準の *installation guide* には、edbctl を使用してこれらのカスタムシークレットを生成するための手順が既に含まれていますが、このページでは、代替としてのシークレットの手動作成と、既存のシークレットをオーバーライドして資格情報を更新またはローテーションする方法を含む詳細を提供します。

edbctl を使用して移行ポータルシークレットを作成する

!!!note migration インストールシナリオが有効になっている展開に必要です。このシナリオは、シナリオ構成 HybridControlPlane CR の spec.scenarios を介して手動で除外されない限り、デフォルトで含まれます。

- 手動インストールの場合、次のコマンドを実行し、対話型プロンプトに従います。

```
edbctl hybrid-manager create-install-secrets --version <version> --scenario migration
```

- CI / CD パイプラインを介してインストールを実行している場合、対話型プロンプトを抑制する必要があります。これを行う方法は、edbctl バージョンによって異なります。

1. 非対話型動作用に edbctl を構成します。

```
edbctl config set interactive_mode off
edbctl config set confirm_mode off
```

1. setup コマンドを実行します。

```
edbctl hybrid-manager create-install-secrets --version <version> --scenario migration
```

これにより、名前空間、カスタムシークレットが作成され、5 つのサービスアカウントすべての安全なランダムパスワードが生成されます。

- edb-migration-portal.db_secrets - edb-migration-portal.db_superuser_secrets - edb-migration-portal.copilot_secrets - edb-migration-copilot.db_secrets - edb-migration-copilot.metrics_auth_secrets

詳細については、 *Migration Portal* **のシークレットを理解する** を参照してください。

既存のシークレットの手動オーバーライド

特定のユーザー名またはパスワードが必要な場合は、シークレットを手動で作成できます。運用環境の場合、データを端末に公開しない適切な安全なプロセスを使用することをお勧めします。

シークレットの作成

1. 各サービスユーザーアカウントの安全なパスワードを生成するファンクションを作成します。

```
function generate_pw {
 dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64 | tr '+/' '-_' | tr -d
'='
}
```

1. シークレットに必要なすべてのユーザーとパスワードの環境変数を設定します。

```
HTTP_USERNAME=test_user_001
HTTP_PASSWORD=$(generate_pw)
DB_SUPERUSER_USERNAME=postgres
DB_SUPERUSER_PASSWORD=$(generate_pw)
PORTAL_DB_USERNAME=test_user_002
PORTAL_DB_PASSWORD=$(generate_pw)
COPILOT_DB_USERNAME=test_user_003
COPILOT_DB_PASSWORD=$(generate_pw)
METRICS_USERNAME=test_user_004
METRICS_PASSWORD=$(generate_pw)
```

1. カスタムシークレットを保存する名前空間を作成します。

```
kubectl create namespace edb-migration-copilot
kubectl create namespace edb-migration-portal
```

1. *database user credentials* を決定するシークレットを作成します。

```
kubectl create secret generic custom-db-secrets \
  --namespace=edb-migration-portal \
  --from-literal=username="${PORTAL_DB_USERNAME}" \
  --from-literal=password="${PORTAL_DB_PASSWORD}" \
  --type=kubernetes.io/basic-auth
```

1. *database superuser credentials* を決定するシークレットを作成します。

```
kubectl create secret generic custom-db-superuser-secrets \
  --namespace=edb-migration-portal \
  --from-literal=username="${DB_SUPERUSER_USERNAME}" \
  --from-literal=password="${DB_SUPERUSER_PASSWORD}" \
  --type=kubernetes.io/basic-auth
```

1. コパイロットと移行ポータルの間で *authenticates HTTP communication* というシークレットを作成します。

```
kubectl create secret generic custom-edb-migration-copilot-auth \
  --namespace=edb-migration-portal \
```

(次のページに続く)

(前のページからの続き)

```
  --from-literal=username=${HTTP_USERNAME} \
  --from-literal=password=${HTTP_PASSWORD} \
  --type=kubernetes.io/basic-auth
```

1. *copilot with access to the Migration Portal database* を提供するシークレットを作成します。

```
kubectl create secret generic custom-ragchew-db-secrets \
  --namespace=edb-migration-copilot \
  --from-literal=username="${COPILOT_DB_USERNAME}" \
  --from-literal=password="${COPILOT_DB_PASSWORD}" \
  --type=kubernetes.io/basic-auth
```

1. monitoring of the copilot instance を有効にするシークレットを作成します。

```
kubectl create secret generic custom-edb-migration-copilot-metrics-auth \
  --namespace=edb-migration-copilot \
  --from-literal=username=${METRICS_USERNAME} \
  --from-literal=password=${METRICS_PASSWORD} \
  --type=kubernetes.io/basic-auth
```

シークレットをインストールに適用する

デフォルトのシークレットをオーバーライドするには、構成で新しいシークレット名を参照します。

HybridControlPlane CR を編集して、spec.componentsParameters の下のカスタムシークレットを参照します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  componentsParameters:
    edb-migration-copilot:
      db_secrets: custom-ragchew-db-secrets
      metrics_auth_secrets: custom-edb-migration-copilot-metrics-auth
    edb-migration-portal:
      copilot_secrets: custom-edb-migration-copilot-auth
      db_owner: ${PORTAL_DB_USERNAME}
      db_secrets: custom-db-secrets
      db_superuser_secrets: custom-db-superuser-secrets
    # ... your other componentsParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

オペレーターは変更を調整し、シークレットオーバーライドを使用するようにインストールを構成します。

インストール後にシークレットをオーバーライドする場合は、CR を再適用して変更を有効にします。

シークレットをバックアップする

次のようなシークレットを取得できます。

```
kubectl get secret -n <namespace> <secret_name> -o yaml
```

このコマンドは、シークレットの内容を YAML 形式で出力するため、安全にコピーして保存できます。作成した各シークレットでこれを繰り返します。

Migration Portal のシークレットを理解する

各シークレットについて、**Description** でそれが必要な理由を説明します。 **構成キー**は、HybridControlPlane CR のどこを参照するかを示します。**パラメーター** は、デフォルト以外の値を設定する方法を示します。

カスタムシークレットを構成し、デフォルトのものをオーバーライドする方法の例については、シークレットをインストールに適用する を参照してください。

Migration Portal データベースアカウント

説明ハイブリッドマネージャーが使用するデータベースサービスアカウントを構成し、移行ポータルシステムデータとスキーマ評価結果を専用データベースに保存します。

構成キー spec.componentsParameters.edb-migration-portal.db_secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: <db-secret-name>
  namespace: edb-migration-portal
stringData:
  username: "<db username>"
  password: "<db password>"
type: kubernetes.io/basic-auth
```

パラメーター

- metadata.name 後で構成キーで参照するシークレットの名前。
- stringData.username データベースに接続するユーザー。また、このユーザーを edb-migration-portal の db_owner キーに入力します シークレットをインストールに適用する を参照してください。
- stringData.password このユーザーに生成されたパスワードを割り当てます。

Migration Portal データベースのスーパーユーザーアカウント

説明ハイブリッドマネージャーが移行ポータルとコパイロットデータベースをプロビジョニングするために使用するデータベーススーパーユーザーアカウントを構成します。

構成キー `spec.componentsParameters.edb-migration-portal.db_superuser_secrets`

```
apiVersion: v1
kind: Secret
metadata:
  name: <db-superuser-secret-name>
  namespace: edb-migration-portal
stringData:
  username: "postgres"
  password: "<db superuser password>"
type: kubernetes.io/basic-auth
```

パラメーター

- `metadata.name` 後で構成キーで参照するシークレットの名前。
- `stringData.username` 常に `postgres` である必要があります。
- `stringData.password` このユーザーに生成されたパスワードを割り当てます。

Migration Copilot Migration Portal からの HTTP アクセス

説明コパイロットと移行ポータル間の HTTP 通信の認証に使用される資格情報を定義します。

構成キー `spec.componentsParameters.edb-migration-portal.copilot_secrets`

```
apiVersion: v1
kind: Secret
metadata:
  name: <copilot-auth-secret>
  namespace: edb-migration-portal
stringData:
  username: "<http username>"
  password: "<http password>"
type: kubernetes.io/basic-auth
```

パラメーター

- `metadata.name` HTTP 要求の認証に使用されるシークレットの名前。後で構成キーで参照します。
- `stringData.username` コパイロットで認証するときに使用する移行ポータルのユーザー名を選択します。
- `stringData.password` このユーザーに生成されたパスワードを割り当てます。

移行コパイロットおよび移行ポータルのデータベースアカウント

説明コパイロットが移行ポータルデータベースと通信できるようにします。edb-migration-copilot コンポーネントは、初期化時に、このユーザーが作成され、コパイロットデータベースに正しい所有権と権限が設定されていることを保証します。

構成キー spec.componentsParameters.edb-migration-copilot.db_secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: <copilot_mp_secret_name>
  namespace: edb-migration-copilot
stringData:
  username: "<db username>"
  password: "<db password>"
type: kubernetes.io/basic-auth
```

パラメーター

- metadata.name コパイロットに移行ポータルデータベースへのアクセスを提供するために使用されるシークレットの名前。
- stringData.username 移行ポータルデータベースにアクセスするときに使用するコパイロットのユーザー名を選択します。
- stringData.password このユーザーに生成されたパスワードを割り当てます。

モニタリングと移行コパイロット

説明ハイブリッドマネージャーは、Grafana を介してコパイロット展開の状態を監視できます。

構成キー spec.componentsParameters.edb-migration-copilot.metrics_auth_secrets

```
apiVersion: v1
kind: Secret
metadata:
  name: <copilot-metrics-secret-name>
  namespace: edb-migration-copilot
stringData:
  username: <metrics-http-username>
  password: <metrics http password>
type: kubernetes.io/basic-auth
```

パラメーター

- metadata.name コパイロットの監視を許可するために使用されるシークレットの名前。
- stringData.username これを、移行ポータルからの移行コパイロット HTTP アクセス用に構成されたユーザー名とは異なるユーザー名に設定します。
- stringData.password このユーザーに生成されたパスワードを割り当てます。

4.1.2.5 Implementing TLS certificates

4.1.2.5.1 ハイブリッドマネージャーポータルのカスタム x.509 証明書を設定する

ハイブリッドマネージャー HM ポータルが使用する自己署名 x.509 証明書を独自に変更できます。

独自の証明書をインポートするには、サーバー証明書とプライベートキーを含む Kubernetes シークレットを作成する必要があります。

証明書を生成する

証明書例 `my-certificate.crt` とプライベートキー例 `my-certificate.key` の base64 エンコード文字列を生成します。

```
cat my-certificate.crt | base64
cat my-certificate.key | base64
```

yaml ファイルを作成する

次の内容を含むファイル例 `my-secret.yaml` を作成し、`<base64 encoded string>` を前の手順で生成された値に置き換えます。

```
apiVersion: v1
data:
  tls.crt: <base64 encoded string>
  tls.key: <base64 encoded string>
kind: Secret
metadata:
  name: my-portal-certificate
  namespace: default
  annotations:
    replicator.v1.mittwald.de/replicate-to: istio-system
type: kubernetes.io/tls
```

Kubernetes クラスターにシークレットを適用する

```
kubectl apply -n default -f ./my-secret.yaml
```

新しい証明書を使用するように HM を構成する

`HybridControlPlane` CR を編集して、`spec.globalParameters` の下に `portal_certificate_secret` パラメーターを追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
```

(次のページに続く)

(前のページからの続き)

```
  globalParameters:
    portal_certificate_secret: "my-portal-certificate"
    # ... your other globalParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

オペレーターは変更を調整し、ポータルの証明書構成を更新します。

シークレットのフォーマットの詳細については、 Istio documentation を参照してください。

4.1.2.5.2 HM ポータルのカスタム cert-manager 発行者を設定する

HM ポータルの証明書は、HM の内部証明書マネージャーでサポートされている x.509 発行者のいずれかを使用して生成および管理することもできます。Let’s Encrypt 証明書の ACME 発行者。

cert-manager Web サイトから直接選択した発行者のドキュメントに従って、設定できます。

> **注釈**
>
> 発行者ではなくクラスター発行者を設定することをお勧めします。発行者を設定する場合は、istio-system 名前空間で作成する必要があります。 !!!

ClusterIssuer が構成されたら、その名前を HM に渡します。

> **注釈**
>
> cert-manager 発行者の使用と独自の証明書シークレットの提供は相互に排他的です。発行者を構成する場合、CR で `spec.globalParameters.portal_certificate_secret` も設定しないでください。 !!!

`HybridControlPlane` CR を編集して、`spec.globalParameters` の下に発行者パラメーターを追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  globalParameters:
    portal_certificate_issuer_kind: "ClusterIssuer"  # Valid values are Issuer and␣
↪ClusterIssuer
    portal_certificate_issuer_name: "my-issuer"       # Your Issuer name
    # ... your other globalParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

オペレーターは変更を調整し、指定された発行者を使用するようにポータルを構成します。

4.1.2.5.3 独自のプライベート認証局を使用する

デフォルトでは、HM が使用するすべての証明書は、cert-manager を使用して内部認証局 CA によって署名されます。CA はインストール時に作成されます。

独自のプライベート CA を使用する場合は、このセクションの手順に従ってください。それ以外の場合、このセクションはスキップできます。

> **注釈**
>
> この方法を正常に使用するには、CA プライベートキーにアクセスする必要があります。 !!!

CA シークレットが存在しない場合は、作成します。

```
apiVersion: v1
data:
  ca.crt: <base64 encoded string> # real ca crt provided by customers
  tls.crt: <base64 encoded string> # real tls crt provided by customers
  tls.key: <base64 encoded string> # real tls key provided by customers
kind: Secret
metadata:
  name: my-custom-ca
  namespace: default
  annotations:
    replicator.v1.mittwald.de/replicate-to: cert-manager
type: kubernetes.io/tls
```

> **警告**
>
> my-custom-ca は、*global-ca-secret* を除き、選択した名前に変更できます。 次のコマンドを実行するときは、選択した名前と一致するようにしてください。 !!!

HybridControlPlane CR を編集して、spec.globalParameters の下に ca_secret_name パラメーターを追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  globalParameters:
```

(次のページに続く)

(前のページからの続き)

```
  ca_secret_name: "my-custom-ca"
  # ... your other globalParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

オペレーターは変更を調整し、プライベート CA を使用するように HM を構成します。

4.1.2.6 Configuring object storage

Hybrid ManagerHM をインストールする前に、`default` 名前空間に `edb-object-storage` という名前の Kubernetes シークレットを作成する必要があります。HM はこのシークレットを使用して、次のオブジェクトストレージにアクセスします。

- Postgres データベースのバックアップと WAL アーカイブ
- リストア操作
- 複数の場所への展開でのデータレプリケーション
- ログ Postgres および内部サービス
- メトリック Postgres および内部サービス

このシークレットは、インストール中に自動的に参照されます。バックアップおよびリカバリーの計画については、*Backup readiness* を参照してください。

複数の場所に展開する場合、`edb-object-storage` シークレットはすべての場所で同期する必要があります。詳細は、*Configuring multiple data centers* を参照してください。

!!!note オブジェクトストレージは完全にハイブリッドマネージャー専用である必要があります。バケットに既に存在するデータは、HM の正常なオペレーションを妨げます。HM を削除して再インストールする場合は、新しいインストールを開始する前にバケットを空にします。

シークレットを作成するには、HM をインストールする前に次の手順を完了します。

1. 以下の **プロバイダー構成** からプロバイダーの構成を選択し、必要な値を入力します。
2. YAML をファイルたとえば `object-storage-secret.yaml` に保存します。
3. クラスターに適用します。

```
kubectl apply -f object-storage-secret.yaml
```

4.1.2.6.1 プロバイダー構成

AWS IAM EKS および ROSA

EKS または ROSA で実行されている場合、ワークロード ID を使用して認証します。静的資格情報は必要ありません。

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage # cannot be changed
  namespace: default # cannot be changed
stringData:
  auth_type: workloadIdentity
  aws_region: <AWS_BUCKET_REGION>
  aws_role_arn: <PRIMARY_IDENTITY_ROLE_ARN>
  bucket_name: <AWS_BUCKET_NAME>
  secondary_role_arn: <SECONDARY_IDENTITY_ROLE_ARN>
  secondary_role_external_id: <SECONDARY_IDENTITY_EXTERNAL_ID>
```

パラメーター

Parameter	Required	Description
auth_type	Yes	`workloadIdentity`である必要があります。
aws_region	Yes	バケットが存在する AWS リージョン。例、*us-east-1*。
aws_role_arn	Yes	HM がバケットにアクセスするために引き受けるプライマリ IAM ロールの ARN。
bucket_name	Yes	S3 バケットの名前。
secondary_role_arn	No	クロスアカウントバケットアクセスのセカンダリロールの ARN。
secondary_role_external_id	No	セカンダリロールを引き受けるときに必要な外部 ID。

AWS 静的資格情報汎用 Kubernetes

ワークロード ID を使用せずに AWS S3 に接続する汎用 Kubernetes クラスターで実行されている場合、静的アクセスキーを使用します。

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage
  namespace: default
stringData:
  auth_type: credentials
  aws_region: <AWS_BUCKET_REGION>
  bucket_name: <AWS_BUCKET_NAME>
  aws_access_key_id: <AWS_ACCESS_KEY_ID>
  aws_secret_access_key: <AWS_SECRET_ACCESS_KEY>
  secondary_role_arn: <SECONDARY_IDENTITY_ROLE_ARN>
  secondary_role_external_id: <SECONDARY_IDENTITY_EXTERNAL_ID>
```

パラメーター

Parameter	Required	Description
auth_type	Yes	`credentials`である必要があります。
aws_region	Yes	バケットが存在する AWS リージョン。
bucket_name	Yes	S3 バケットの名前。
aws_access_key_id	Yes	IAM ユーザーまたはロールの AWS アクセスキー ID。
aws_secret_access_key	Yes	AWS シークレットアクセスキー。
secondary_role_arn	No	クロスアカウントバケットアクセスのセカンダリロールの ARN。
secondary_role_external_id	No	セカンダリロールを引き受けるときに必要な外部 ID。

Azure Blob Storage

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage
  namespace: default
stringData:
  provider: azure
  subscription_id: <AZURE_SUBSCRIPTION_ID>
  resource_group_name: <AZURE_RESOURCE_GROUP>
  storage_account_name: <AZURE_STORAGE_ACCOUNT>
  storage_account_container_name: <AZURE_STORAGE_CONTAINER>
  storage_account_key: <AZURE_STORAGE_KEY>
  region: <AZURE_REGION>
  client_id: <AZURE_CLIENT_ID>
  client_secret: <AZURE_CLIENT_SECRET>
  tenant_id: <AZURE_TENANT_ID>
```

パラメーター

Parameter	Required	Description
provider	Yes	`azure`である必要があります。
subscription_id	Yes	Azure サブスクリプション ID。
resource_group_name	Yes	ストレージアカウントを含む Azure リソースグループ。
storage_account_name	Yes	Azure ストレージアカウントの名前。

次のページに続く

表 35 – 前のページからの続き

Parameter	Required	Description
*storage_account_container_*	Yes	ストレージアカウント内の BLOB コンテナの名前。
storage_account_key	Yes	ストレージアカウントのアクセスキー。
region	Yes	ストレージアカウントの Azure リージョン。例、*eastus*。
client_id	Yes	認証に使用される Azure サービスプリンシパルのクライアント ID。
client_secret	Yes	Azure サービスプリンシパルのクライアントシークレット。
tenant_id	Yes	Azure Active Directory のテナント ID。

GCP オブジェクトストレージ

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage
  namespace: default
stringData:
  provider: gcp
  location_id: <GCP_BUCKET_REGION>
  project_id: <GCP_PROJECT_ID>
  bucket_name: <GCP_BUCKET_NAME>
  credentials_json_base64: <GCP_CREDENTIAL_BASE64>
```

パラメーター

Parameter	Required	Description
provider	Yes	`gcp`である必要があります。
location_id	Yes	バケットが存在する GCP リージョン。例、*us-central1*。
project_id	Yes	GCP プロジェクト ID。
bucket_name	Yes	GCS バケットの名前。
credentials_json_base64	Yes	GCP サービスアカウントの JSON キーファイルの Base64 エンコードされたコンテンツ。手順については、*Troubleshooting* を参照してください。

その他の S3 互換ストレージ

この構成は、オンプレミスまたは MinIO や Ceph などのサードパーティの S3 互換ストレージに使用します。S3 互換ストレージでは、AWS 静的資格情報 `auth_type: credentials` のみがサポートされています。ワークロード ID は、AWS 以外のプロバイダーは使用できません。

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage
  namespace: default
stringData:
  auth_type: credentials
  aws_endpoint_url_s3: <S3_ENDPOINT_URL>
  aws_access_key_id: <AWS_ACCESS_KEY_ID>
  aws_secret_access_key: <AWS_SECRET_ACCESS_KEY>
  bucket_name: <S3_BUCKET_NAME>
  aws_region: <S3_REGION>
  # Optional: use ONE of the following if your endpoint uses a private or self-signed␣
↪CA
  aws_ca_bundle: <CA_BUNDLE_PEM>          # raw PEM format
  # aws_ca_bundle_base64: <CA_BUNDLE_BASE64>  # base64-encoded PEM (alternative)
  # Set to true if server-side encryption is disabled on the bucket
  server_side_encryption_disabled: <true|false>
  # Required for on-premises storage that does not support checksum headers; added in␣
↪HM 2025.12
  aws_request_checksum_calculation: when_required
  aws_response_checksum_validation: when_required
```

パラメーター

Parameter	Required	Description
auth_type	Yes	`credentials`である必要があります。
aws_endpoint_url_s3	Yes	プロトコルを含む S3 互換エンドポイントの完全な URL。例、*https://minio.example.com*。
aws_access_key_id	Yes	S3 互換ストレージのアクセスキー ID。
aws_secret_access_key	Yes	シークレットアクセスキー。
bucket_name	Yes	バケットの名前。
aws_region	Yes	リージョン識別子。リージョンを使用しないストレージの場合、`us-east-1`などのプレースホルダーを使用します。
aws_ca_bundle	No	未加工の PEM 形式の CA バンドル。これまたは`aws_ca_bundle_base64`を使用します — 両方ではなく。 S3 エンドポイントがプライベートまたは自己署名 CA によって署名された証明書を使用する場合に必要です。
aws_ca_bundle_base64	No	Base64 エンコードされた PEM 文字列としての CA バンドル。これまたは`aws_ca_bundle`を使用します — 両方ではなく。手順については、*CA バンドルの提供* を参照してください。

次のページに続く

表 37 – 前のページからの続き

Parameter	Required	Description
*server_side_encryption_*	No	バケットでサーバーサイドの暗号化が無効になっている場合、 *true* `に設定します。デフォルト *`false*
*aws_request_checksum_*	No	AWS SDK がリクエストのチェックサムを計算する時期を制御します。チェックサムヘッダーをサポートしていないオンプレミスストレージの場合は、`when_required`に設定します。これにより、一部の S3 互換プロバイダーで見られる `x-amz-content-sha256`エラーが解決されます。 HM 2025.12 で追加されました。
aws_response_checksum	No	AWS SDK がレスポンスチェックサムを検証する時期を制御します。`aws_request_checksum_calculation`とともに`when_required`に設定します。 HM 2025.12 で追加されました。

4.1.2.6.2 CA バンドルの提供

その他の *S3* 互換ストレージ を使用しており、エンドポイントがプライベートまたは自己署名の認証局 CA によって署名された証明書を使用している場合、CA バンドルを提供する必要があります。シークレットの `aws_ca_bundle` または `aws_ca_bundle_base64` フィールドに追加します。2 つのフィールドがサポートされています—両方ではなく、いずれか一方を使用します。

Field	Format	When to use
aws_ca_bundle	Raw PEM text	You have the *.pem* or *.crt* file and can paste its contents directly.
aws_ca_bundle_base64	Base64-encoded PEM	You need to embed the bundle as a single-line string.

`aws_ca_bundle` raw PEM の使用

CA バンドルファイルの内容を値として直接貼り付けます。PEM ファイルは`-----BEGIN CERTIFICATE-----` で始まります。

```
stringData:
  aws_ca_bundle: |
    -----BEGIN CERTIFICATE-----
    MIIDazCCAlOgAwIBAgI...
    -----END CERTIFICATE-----
```

aws_ca_bundle_base64 base64 エンコードされた PEM の使用

シークレットで使用する前に、CA バンドルファイルをエンコードします。

Linux の場合

```
base64 -w 0 /path/to/ca-bundle.pem
```

macOS の場合

```
base64 -i /path/to/ca-bundle.pem
```

出力をコピーし、aws_ca_bundle_base64 の値として設定します。

!!!note CA バンドルに証明書チェーン中間+ルート CA が含まれる場合、エンコードする前に PEM ファイルにすべての証明書を含め、発行 CA からルートの順に連結します。

GCP configuration を使用している場合、シークレットに追加する前に、サービスアカウント JSON キーファイルを base64 文字列としてエンコードする必要があります次のコマンドを使用して、その値を生成します。

資格情報ファイルをエンコードするには

```
base64 -w 0 /path/to/service-account-key.json
```

macOS では、次を使用します。

```
base64 -i /path/to/service-account-key.json
```

4.1.2.6.3 プロジェクトごとのオブジェクトストレージの構成

デフォルトでは、すべてのプロジェクトは、インストール中に構成されたオブジェクトストレージを共有します。インストール後、そのプロジェクトに Postgres ワークロードを作成する前に、いつでも個々のプロジェクトの専用オブジェクトストレージを構成できます。これにより、各プロジェクトのバックアップと WAL データが分離されます。

プロジェクト ID の取得、正規化、シークレットの適用方法を含む詳細な手順については、*Configure dedicated object storage for a project* を参照してください。

4.1.2.7 Deploying Hybrid Manager across multiple data centers

高可用性要件を満たすように、複数のデータセンターにわたって HM インストールを構成できますマルチ DC トポロジ。このトポロジには、プライマリ HM と、複数の Kubernetes クラスターにわたる 1 つ以上のデータ専用 HM セカンダリが含まれます。

- 概要—マルチ DC の概念をレビューし、使用可能なマルチ DC トポロジを調査し、マルチ DC HM インストールのアーキテクチャレビューを提供します。
- *Configuring a multi-DC deployment* —複数のデータセンターにわたって HM インストールを構成し、さまざまな場所に Postgres クラスターをプロビジョニングする方法に関するステップバイステップガイドを提供します。

4.1.2.7.1 Multi-DC for high availability

複数のデータセンターマルチ DC にわたってハイブリッドマネージャー HM を展開すると、Postgres データベースクラスターの高可用性とディザスターリカバリーが提供されます。この設定により、単一のインターフェイスから異なる地理的な場所にあるデータベースを管理できます。

!!!重要 これは、HM インストール自分自身のリカバリーを除きます。これは、*back up and restore separately* する必要があります。

複数のデータセンターに HM を展開すると、Postgres データベースクラスターの高可用性とディザスターリカバリーが得られます。それにより、次のことができます。

- データベース損失 DR を生き延びる —1 つまたは 2 つのウォームセカンダリ Postgres データベースを準備しておきます。プライマリデータセンターが利用できない場合は、セカンダリの場所でレプリカをプロモートし、サービスを復元します。
- ダウンタイム RTO の最小化—複数の地理的に分散した書き込みエンドポイント DHA Postgres クラスターを使用するか、迅速なフェイルオーバーシングルノード、HA、および AHA Postgres クラスターを活用して、メンテナンスまたは計画外のサイトの停止中にサービスを迅速に復元します。
- データの保護 RPO - 2 番目のデータセンターへの継続的レプリケーションにより、単一サイトのバックアップのみと比較して、潜在的なデータ損失が削減されます。
- 爆風半径を縮小します—一方のデータセンターの障害、構成ミス、またはノイジーネイバーは、他方のデータセンターをダウンさせません。
- コンプライアンス/主権を満たす—制御を集中させながら、特定の地域または施設にコピーを保持します。
- 大規模な運用—読み取りトラフィックを分割し、アップグレードを段階的に行うか、データセンター全体でブルー/グリーンのカットオーバーを実行します。
- 一元管理—単一のプライマリ HM がすべてのリモート Kubernetes クラスターにわたってデータベースを管理し、地理的冗長性を提供しながら管理を簡素化します。

アーキテクチャとロール ハブアンドスポーク

HM は、ハブアンドスポークモデルを使用して、マルチプルのデータセンターを管理します。コントロールプレーン管理ソフトウェアとデータプレーン実際の Postgres データベースを区別することが重要です。

プライマリザ・ハブ

プライマリロケーションは「真実のソース」であり、完全な HM コントロールプレーンがアクティブな唯一のサイトです。次のプロセスは、プライマリまたはハブが所有しています。

- **HM コンソール UI** 単一のウィンドウ。プライマリの URL にログインして、すべてのリージョンのデータベースを管理します。
- **セントラルスケジューラー**プライマリは、データベースノードが存在する場所を決定し、セカンダリにプロビジョニング指示を送信します。
- **ID プロバイダー** SPIRE の信頼のルートとして機能し、すべてのセカンダリの場所に安全な ID を発行します。

- **オブザーバビリティハブ** Thanos と Loki を使用して、すべての場所からメトリックとログを単一のビューに集約します。

セカンダリスポーク

セカンダリは、管理ではなく実行のために設計された「リーン」インストールです。これらは独自の独立したコンソールを実行しませんが、プライマリまたはハブからの通信と管理を促進するコンポーネントを実行します。

- **HM 内部ビーコンエージェント** UI の代わりに、セカンダリは HM 内部ビーコンエージェントを実行します。このエージェントはプライマリにハートビートを返し、ローカル Kubernetes リソースノード、ストレージを中央スケジューラーで利用できるようにします。
- **ローカル実行**クラスターをセカンダリの場所に展開すると、プライマリはセカンダリのローカルオペレータに Postgres ポッドをプロビジョニングするように指示します。

データプレーンとストレージ

コントロールプレーンはハブアンドスポークですが、データは環境全体で同期されたままです。

- **Postgres クラスター** DC-A にプライマリ Postgres クラスターを展開し、DC-B および DC-C に非同期レプリカを展開できます。分散高可用性 PGD の場合、データグループと Witness ノードを 3 つの場所すべてに分散できます。
- **ユニファイドストレージ**すべての場所は、一貫したオブジェクトストア構成を共有する必要があります。これにより、あるデータセンターで生成されたバックアップと WAL ファイルは、別のデータセンターでリカバリーにすぐに利用できます。

トポロジ

次のトポロジを使用してマルチ DC セットアップを展開できます。

- **プライマリおよびデータのみの HM** これは、1 つのプライマリ HM と少なくとも 1 つのデータ専用 HM を使用した基本的なセットアップです。これにより、2 つの場所に Postgres クラスターを展開できます。
- **複数のデータ専用 HM を使用した統合管理**単一のプライマリ HM から複数のリモート Kubernetes クラスターにわたってデータベースを管理します。このアーキテクチャは、各ターゲットの場所で軽量のデータ専用のフットプリントを維持しながら、統合されたコントロールプレーンを提供します。
- **PGD 用の 3 つの HM** 最適な PGD 展開分散および高度な高可用性 Postgres クラスター、特に 2 つのデータグループと 1 つの監視グループを使用する場合、3 ロケーショントポロジをお勧めします。この構成により、適切な分散コンセンサスのためのクォーラムが保証されます。

復元力とフェイルオーバー

データセンター全体でさまざまなデータベーストポロジを組み合わせて使用できるため、サイトに障害が発生した場合の影響は異なります。フェールオーバーは、ハイブリッドマネージャー環境全体を一度に実行するのではなく、データベースクラスターごとに実行されます。次にいくつかの例を示します。

Database topology	Setup	Recovery action	RTO (Time)	RPO (Data)
Single-Node	No replicas, located in DC-A.	**Full restore:** Must recreate the cluster from object storage/backup.	**High:** depends on DB size and restore speed.	**Medium:** Up to the last backup/WAL archive.
Single-Node + Replica	Single-node in DC-A, replica in DC-B.	**Manual promotion:** Promote the replica in DC-B to a standalone primary.	**Low:** is driven by runbook and DNS updates.	**Very low:** Seconds (based on async replication lag).
HA cluster	Primary/Replic in DC-A, replica in DC-B.	**Manual promotion:** Promote the cross-DC replica in DC-B to primary.	**Low:** is driven by runbook and DNS updates.	**Very low:** Seconds (based on async replication lag).
DHA cluster	Data Groups in DC-A and DC-B, witness in DC-C.	**Automatic:** PGD maintains quorum; surviving nodes stay writable.	**Near zero:** Instant, limited only by DNS/LB TTL.	**Zero:** If using sync commit, otherwise near-zero.

リカバリーの違いは、マルチ DC セットアップと、特定のデータベースクラスターごとに選択したトポロジの両方に依存します。

ローカル HA とマルチ DC フェールオーバーディザスターリカバリーの観点から、シングルインスタンスと HA クラスターは両方とも、プライマリサイトに障害が発生した場合、セカンダリの場所へのプロモーションが必要です。ただし、HA クラスターローカルレプリカを使用すると、内部復元力が提供されます。同じデータセンター内のプライマリインスタンスの障害から自動的に復旧できます。ローカルスタンバイがない単一インスタンス展開では、データベースを完全に再起動するか、完全なクロスロケーションフェールオーバーを実行して、ローカルの中断から復旧する必要があります。

RTO 目標復旧時間これはダウンタイムです。手動プロモーションシングルノード+レプリカまたは HA クラスターの場合、RTO はフェールオーバープレイブックを実行できる速度によって制限されます。

RPO 目標復旧ポイントこれは、データの損失です。マルチ DC の標準である非同期レプリケーションの場合、これは通常数秒の RPO になります。同期レプリケーションの場合、ゼロ RPO を達成できますが、これにより、リモートデータセンターがデータを確認するのを待機するため、すべての書き込みに遅延が追加されます。

分散高可用性 DHA クラスターのネットワーク遅延

分散高可用性クラスター PGD を搭載したクラスターがコンセンサスの問題に対応する速さと、存在するネットワーク遅延の量の間には、固有のトレードオフがあります。レイテンシーが長くなると、より緩やかなコンセンサス設定が必要になり、障害の検出と対応に時間がかかる可能性があります。一部の Raft 設定は、HM コンソールから直接構成できない場合があります。

リージョンをまたがるクラウド環境と比較して、オンプレミスの密結合環境では、遅延はあまり問題になりません。高レイテンシーシナリオに対応するように、よりリラックスした Raft 設定を構成できます。

pg CLI を使用して、クラスターと対話し、コンソールに公開されていない構成設定、特に Raft 設定を調整できます。`bdr run on nodes` ファンクションを使用して、他のノードでコマンドを実行します。

マルチ DC 環境の HM バックアップと DR

HM は 2 つの方法を使用してデータを保護しますマルチ DC セットアップでは、これらのロールは固有です。

Barman (Cross-DC) バックアップはグローバルオブジェクトストアに保存されます。これは、場所を超えたディザスターリカバリーのための主要なツールです。

ボリュームスナップショットローカル特定のデータセンターにローカルなディスクベースのバックアップ。これらは、正常なサイト内での高速な「ポイントインタイム」リカバリーに使用します。

サイトの停止中に Hybrid Manager コントロールプレーンと Postgres データベースの両方を復旧できるように保証には、以下を確認してください。

- プライマリ HM メタデータ構成、ユーザー、クラスター定義がバックアップされ、プライマリデータセンターの外部に保存されていることを確認します。
- 共有バックアップにライブでアクセスするには、すべての HM インスタンスプライマリおよびセカンダリに同一の `edb-object-storage` シークレットが必要です。
- 場所ごとに *automated backup schedules* を個別に構成する必要があります。設定はサイト間で自動的に同期しません。
- 組織は、プライマリサイトの停止のために HM コンソールがダウンしている場合に *restore a database using the CLI* の方法を知っています。

現在の制限

これらは、マルチ DC 展開の現在の制限です。

- 最大 3 つの場所プライマリと 1 つのセカンダリ、またはプライマリと 2 つのセカンダリ。
- 手動フェイルオーバープライマリがダウンしている場合、セカンダリの場所のいずれかでレプリカを昇格させます。
- 同じクラウド/オンプレミスファミリークロス CSP マルチ DC はサポートされていません。

4.1.2.7.2 Configuring a Hybrid Manager deployment across multiple data centers

Postgres データベースを複数のデータセンターに展開できるようにするには、最初に HM インストールを 2 つまたは 3 つの異なる地理的場所またはリージョンに展開し、*Hub & Spoke or Primary & Secondary architecture* をサポートする必要があります。次に、これらのインストールを相互にリンクすると、単一の HM コンソールから管理できます。このガイドは、次のことに役立ちます。

- すべての HM インストール構成ファイルを作成し、展開します。
- 同じプロバイダー/オンプレミスファミリーで 2 つ以上の HM インストール HM Kubernetes クラスターを接続します。これらは異なる地理的リージョンにある場合があります。たとえば、プライマリと 1 つのセカンダリ、またはプライマリと 2 つのセカンダリを持つことができます。
- オブジェクトストレージ同一の `edb-object-storage` シークレットを調整するため、すべてのデータセンターでバックアップ/アーティファクトを使用できます。
- HM 内部エージェントビーコンまたは `upm-beacon-agent` を接続して、セカンダリが管理対象場所としてプライマリに登録し、そこでプロビジョニングできるようにします 9445/TCP。
- 一方のデータセンターにプライマリ Postgres クラスターを使用し、他のデータセンターにレプリカ Postgres クラスターを使用して Postgres トポロジを準備します。レプリカを昇格させることにより手動フェイルオーバーを実行します。

始める前に

マルチ DC 手順を開始する前に、以下についてよく理解してください。

- 概要

前提条件

アーキテクチャの前提条件

このマルチ DC セットアップは、単一のプライマリが複数のリーンセカンダリを管理するハブアンドスポークモデルに従います。

- ハブクラスタープライマリ HM をホストする 1 つの Kubernetes クラスター。このクラスターは、中央の管理プレーンおよび UI として機能します。
- スポーククラスターセカンダリ HM をホストする 1 つまたは 2 つの Kubernetes クラスター。これらは、データベースワークロードの実行ポイントとして機能します。
- ネットワーク接続
  - 8444/クラスター間でオープンした TCP SPIRE バンドルエンドポイント.
  - 9445/セカンダリ → プライマリからの TCP Beacon gRPC.
  - 同じプロバイダー/オンプレミスファミリー (クロスクラウドなし)。
- 共有オブジェクトストレージ標準の HM インストールでは、個々の Kubernetes クラスター HM インストールに専用のオブジェクトストレージが必要です。マルチ DC 展開の場合、このオブジェクトストレージはすべてのクラスターで共有されます。

必要な情報を収集する

1. HM インストール構成ファイルのコピーを 2 つまたは 3 つ準備します。2 つの場所に展開する場合は `primary.yaml` 、`secondary.yaml` 、3 つの場所に展開する場合は `primary.yaml` 、`secondaryA.yaml` 、および `secondaryB.yaml` と名前付けます。これらのファイルは `HybridControlPlane` CR マニフェストです。
2. ドメイン名

各 HM インストールは、専用のドメイン名で構成し、構成で `portal_domain_name` として設定する必要があります。このパラメーターは、プライマリとセカンダリで使用されます。

各 HM インストール構成に応じて 2 つまたは 3 つのインストールごとにこれらのドメイン名を作成し、この情報を記録します。

場所をまたがるオブジェクトストレージ

HM は、バックアップ、アーティファクト、WAL、および内部バンドルにオブジェクトストアを使用します。**マルチ DC では、すべての HM インストールが同じオブジェクトストア構成を使用する必要があります。**

キー要件

すべての HM インストールには、デフォルトの名前空間に `edb-object-storage` という名前の同一の Kubernetes シークレットが必要です。2 つの場所で実行している場合、このシークレットはプライマリとセカンダリに保存します。3 つのデータセンターがある場合、このシークレットをプライマリおよびセカンダリの場所に保存します。

パラメーターの一意性

このマルチデータセンター展開に関係する 2 つまたは 3 つのクラスターは同じオブジェクトストアを使用しているため、すべてのクラスターで特定のパラメーターが異なることを確認する必要があります。

Parameter	CR path	Requirement	Why it matters
beacon_location_id	*spec.componentsPara beacon.beacon_locat*	Must be unique per location.	Human-readable location identifier. Identifies the specific location in the HM console and API.
internal_backup_folder	*spec.globalParameter*	Must be unique per location AND match the format: *^[0-9a-z]{12}$*.	Separates database backups so that a restore doesn't pull the wrong data.
metrics_storage_prefix	*spec.globalParameter*	Must be unique per location.	Ensures observability data from each site is stored in its own directory.
logs_storage_prefix	*spec.componentsPara loki.logs_storage_pre*	Must be unique per location.	Prevents logs from different locations from overwriting each other in shared storage.

マルチ DC トポロジの構成

この手順では、ハブアンドスポーク関係を定義します。ハブプライマリはそのすべてのスポークセカンダリについて知っている必要があり、各スポークはハブに到達する方法を知っている必要があります。

次のスタンザを各構成ファイル primary.yaml および secondary.yaml に追加します。 HM インストールごとに前に設定した portal_domain_name を使用して、domainName 値を入力します。

primary.yaml で ロール primary を設定します。このクラスターはハブとして機能します。

secondary.yaml で ロール secondary を設定します。このクラスターはスポークとして機能し、primary.domainName を使用してマネージャーを見つけます。

各 HybridControlPlane CR の spec の下に clusterGroups スタンザを追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  clusterGroups:
    # Set to primary for your Hub, secondary for your Spokes
    role: (secondary|primary|standalone)
    primary:
      domainName: <primary portal domain>
    secondaries:
    - domainName: <secondary portal domain>
      # Add an additional entry here for additional locations
  # ... your other spec fields
```

コンポーネントのセットの削減スポーククラスターのみ

HM は多数の異なるコンポーネントで構成されており、セカンダリの場所では必要ないものもあります。フルセットをセカンダリの場所に正常にインストールできますが、パラメーター scenarios を core のみに設定し、HM コンソール UI を無効にすることにより、そのリストを削減することをお勧めします。

次のパラメーターをセカンダリロケーションの構成ファイルに追加します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  scenarios:
    - core
  disabledComponents:
```

(次のページに続く)

(前のページからの続き)

```
    - upm-ui
  # ... your other spec fields
```

1. 検証チェックリスト

- edb-object-storage シークレットは、すべての場所で同一である必要があります。データのみを比較。
- すべての場所でバケットをリスト/書き込みできます簡単なポッド/ジョブテスト。
- beacon_location_id 、internal_backup_folder 、metrics_storage_prefix 、および logs_storage_prefix は以下である必要がありますロケーションごとに一意。
- scenarios は、セカンダリロケーションの core に設定されています。
- disabledComponents には、セカンダリロケーション用にリストされた upm-ui コンポーネントがあります。

Hybrid Manager のインストール

構成ファイル primary.yaml および secondary.yaml 、または 3 か所のセットアップの場合は primary.yaml 、secondaryA.yaml および secondaryB.yaml を使用して、各クラスターにハイブリッドマネージャーをインストールします。完全なインストール手順については、フェーズ 5 ハイブリッドマネージャーのインストール に従ってください。

配線の検証

インストールが完了したら、ハブアンドスポークリンクを確認します。

1. プライマリで、管理対象のセカンダリの場所をリストします。

```
kubectl get location
```

各管理ロケーションは、 managed-<LOCATION_ID> managed-primary および managed-secondary などの名前付けの行として表示され、LOCATIONID の下にロケーション ID があり、最近の LASTHEARTBEAT があります。

```
NAME                SCOPE     LOCATIONID   LASTHEARTBEAT
managed-primary     managed   primary      2026-01-15T10:30:00Z
managed-secondary   managed   secondary    2026-01-15T10:30:00Z
```

1. SPIRE フェデレーションがすべての場所に存在することを確認します。

edb_notranlate_4 bundle endpoint profile: https_spiffe およびピアの:8444 URL との関係ピア信頼ドメインを示す federation list が表示されます。

プライマリでは、フェデレーションリストに 1 つまたは 2 つのエントリセカンダリごとに 1 つ表示されます。

各セカンダリインスタンスには、1 つのエントリプライマリへの関係のみが表示されます。

さまざまなデータセンターにまたがって Postgres データベースクラスターを作成する

HM はセカンダリの場所にプロビジョニングできるようになりましたが、実際のデータベーストポロジを選択して作成する必要があります。HM コンソールで、データベースクラスターを作成し、データベースノードごとに異なる場所を選択してください。

- シングルノードおよび高可用性データベースクラスターの場合、**リージョンをまたがってボリュームスナップショットリカバリーを使用する場合、レプリカクラスターの作成が失敗する** 中に、***Replica Clusters** tab* を使用して他の場所にレプリカデータベースクラスターを追加します。
- 高度な高可用性および分散高可用性データベースクラスターの場合、**リージョンをまたがってボリュームスナップショットリカバリーを使用する場合、レプリカクラスターの作成が失敗する** 中に、***Data Groups** tab > **Node Settings** > **Deployment Location*** オプションを使用してクラスターデータグループを場所を超えて分散します。

!!!note PGD の Witness ノードには、少なくとも 10GB のディスク領域が必要です。 .. ::
バックアップが両方のデータセンターから共有オブジェクトストアに書き込まれていることを確認します。

操作上の注意事項

- DB TLS は、SPIRE / Beacon プラットフォーム ID とは別です。ポリシーごとに PG TLS を構成します。
- 各 DC のストレージクラスが PG IOPS/レイテンシーを満たしていることを確認します。
- サイト間でレプリケーションポートを開きます。

検証エンドツーエンド

プライマリロケーションで、次のチェックを実行して、マルチ DC セットアップを検証します。

1. プライマリ/セカンダリクラスターの関係を検証します。

```
kubectl -n spire-system exec svc/spire-server -c spire-server -- \
/opt/spire/bin/spire-server federation list
```

1. セカンダリの場所が登録されていることを確認します。

```
kubectl get location
```

1. セカンダリへのプロビジョニングを検証する

- プライマリから、小規模なテストワークロードをセカンダリの場所に展開します。- テレメトリオプショナル Thanos ストアはフェデレーションピアを表示します。Loki クエリーは、secondary. - オブジェクトストレージからタグ付けされたログを返しますどちらのクラスターもバケットの読み取り/書き込みができます。シークレットは同じです。

手動フェイルオーバー

プライマリロケーションからセカンダリロケーションへのデータベースの手動フェイルオーバー手順

1. プライマリロケーションへの書き込みを一時停止しますメンテナンスモード/LB カットオーバー。
2. セカンダリの場所のレプリカをプライマリに昇格させます HM コンソールまたはスクリプトを使用して。
3. クライアント DNS / LB をセカンダリの場所にリダイレクトします。
4. 観察書き込みが成功したことを確認します。そして、レプリケーションロールが更新されます。
5. プライマリロケーションが元に戻ったら、新しいプライマリのレプリカとして再シードします。オプションで、後の削減を計画します。

!!!note セカンダリ DC で Postgres データベースを昇格させている間、HM コンソール自分自身はプライマリで実行されたままです。プライマリ DCHM が存在する場所が完全になくなった場合は、 *HM disaster recovery guide* に従って、他の場所で管理コンソールを復元します。

- カットオーバーのために DNS TTL を十分に低く保ちます。
- ダウンタイムを追跡して RTO を測定します。
- プロモーション後のバックアップを検証します。

障害対応

- 問題 **フェデレーション関係がありません**
  - 再生成およびクロス適用 `ClusterFederatedTrustDomain` CRs.
  - 8444 / TCP の到達可能性を確認します。
- 問題 **``kubectl get location`` にリストされていないセカンダリ**
  - 両側のビーコン値を再確認します。再起動 Beacon server/agent.
  - プライマリポータルへの 9445/TCP の到達可能性を確認します。トラストドメインが正しいです。
- 問題 **セカンダリ**
  - でオブジェクトストアアクセスが失敗する 再同期 `edb-object-storage` .
  - EKS / IRSA の場合セカンダリ OIDC がロールの信頼ポリシーに含まれていることを確認します。
- 問題 **テレメトリフェデレーションがありません**
  - 正しい-l プライマリ|セカンダリフラグと一意のプレフィックスで再インストールします。
  - Thanos `/api/v1/stores` および Loki 読み取り API を確認します。
- 問題 **レプリカの遅延/接続** - ネットワーク ACL / SG、TLS 証明書、およびストレージパフォーマンスを確認します。

付録 B — 簡単な日常チェック

- プライマリの kubectl get location は、セカンダリの準備ができていることを示します。
- Thanos / Loki フェデレーションは正常な場合有効な場合。
- オブジェクトストアの書き込みは両方の DC から成功します。
- SLO 内のレプリケーションラグ。

4.1.2.8 Enabling Key Management Systems (KMS) for TDE

新しいクラスターを作成するときに、データベースに保存されている encrypt the data に対して**透過的データ暗号化 TDE** を有効にできます。

secure the data encryption key には、選択したパスフレーズを入力するか、 HashiCorp Vault や AWS Key Management Service などのキー管理プロバイダーから独自のキーを持ってくることができます。

!!!note TDE は、バージョン 15 以降の EDB Postgres Advanced Server および EDB Postgres Extended Server で利用できます。

- *Adding KMS support*

次に、キー管理システムプロバイダーを構成します。

- *HashiCorp Vault*
- *AWS KMS*
- *Azure KMS*
- *Google Cloud KMS* **キーの作成**

4.1.2.8.1 Adding KMS support

ハイブリッドマネージャー HM を展開すると、HybridControlPlane カスタムリソースには、デフォルトで TDE 実装にパスフレーズまたは HashCorp Vault キーを使用する機能が含まれます。

```
# - transparentDataEncryptionMethods: List of TDE Encryption Methods for the agent.
#     Supported options: passphrase, aws_kms, azure_kms, gcp_kms, hashicorp_vault.
#     Default values: passphrase, hashicorp_vault.
transparentDataEncryptionMethods:
  - passphrase
  - hashicorp_vault
```

TDE に別のキー管理システムを使用する場合は、インストール中にこのセクションを編集できます。または、ハイブリッドマネージャーをインストールした後にキープロバイダーを追加できます。その場合、更新された構成で HybridControlPlane CR を再適用して、システムに新しい設定を入力する必要があります。CR を適用する方法に関するインフラストラクチャ固有の手順については、インストールガイドを参照してください。

HashiCorp Vault

ハイブリッドマネージャー構成には、すぐに使える HashCorp のサポートが含まれています。

AWS KMS

Amazon の KMS、Amazon Key Management サービスを使用するには、 `aws_kms` をリストに追加します。

```
# - transparentDataEncryptionMethods: List of TDE Encryption Methods for the agent.
#     Supported options: passphrase, aws_kms, azure_kms, gcp_kms, hashicorp_vault.
#     Default values: passphrase, hashicorp_vault.
transparentDataEncryptionMethods:
  - passphrase
  - hashicorp_vault
  - aws_kms
```

Azure KMS

Azure の KMS、Azure Key Vault を使用するには、 `azure_kms` をリストに追加します。

```
# - transparentDataEncryptionMethods: List of TDE Encryption Methods for the agent.
#     Supported options: passphrase, aws_kms, azure_kms, gcp_kms, hashicorp_vault.
#     Default values: passphrase, hashicorp_vault.
transparentDataEncryptionMethods:
  - passphrase
  - hashicorp_vault
  - azure_kms
```

Google KMS

Google の KMS クラウドキー管理サービスを使用する場合は、 `gcp_kms` をリストに追加します。

```
# - transparentDataEncryptionMethods: List of TDE Encryption Methods for the agent.
#     Supported options: passphrase, aws_kms, azure_kms, gcp_kms, hashicorp_vault.
#     Default values: passphrase, hashicorp_vault.
transparentDataEncryptionMethods:
  - passphrase
  - hashicorp_vault
  - gcp_kms
```

次のステップ

- *Configure HashiCorp Vault as the KMS for TDE*
- *Configure AWS Key Management Service as the KMS for TDE*
- *Configure Azure Key Vault as the KMS for TDE*
- *Configure Google Cloud Key Management Service as the KMS for TDE*

4.1.2.8.2 HashiCorp Vault

HashiCorp Vault キーを使用して TDE 対応クラスターを作成する前に、管理者は Hybrid Manager インスタンスの Vault をインストールして構成する必要があります。

この例では、ハイブリッドマネージャーと同じ Kubernetes 環境に Vault のカスタマイズバージョンをインストールし、Vault プロバイダーを構成して、キーを使用して TDE が暗号化を有効にした新しいデータベースクラスターをプロビジョニングできるようにします。

ベンダーニュートラルなキー管理ソリューションを目指している場合は、さまざまなクラウドプロバイダーとオンプレミス環境にわたって一貫した一元的なキー管理を提供し、ベンダーロックインを防止するため、HashCorp Vault を使用するのが現実的です。

Vault のインストール

ハイブリッドマネージャーを展開したのと同じ Kubernetes 環境上の Install HashiCorp Vault 必ず独自の `vault` 名前空間にインストールしてください。

1. Helm リポジトリを設定します。

```
helm repo add hashicorp https://helm.releases.hashicorp.com
```

2. インストールのカスタマイズオーバーライドを作成して、Vault がハイブリッドマネージャーのコントロールプレーンポート 8200 にインストールされるようにします。

```
   cat > helm-vault-raft-values.yml <<EOF
   injector:
    tolerations: |
      - key: "edbaiplatform.io/control-plane"
        effect: NoSchedule
        operator: Equal
        value: "true"

 affinity: |
   nodeAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - preference:
         matchExpressions:
         - key: edbaiplatform.io/control-plane
           operator: Exists
       weight: 1
server:
 tolerations: |
   - key: "edbaiplatform.io/control-plane"
     effect: NoSchedule
     operator: Equal
     value: "true"

 affinity: |
```

(次のページに続く)

(前のページからの続き)

```
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - preference:
        matchExpressions:
        - key: edbaiplatform.io/control-plane
          operator: Exists
      weight: 1
ha:
    enabled: true
    raft:
       enabled: true
       setNodeId: true
       config: |
          cluster_name = "vault-integrated-storage"
          storage "raft" {
             path    = "/vault/data/"
          }

          listener "tcp" {
             address = "[::]:8200"
             cluster_address = "[::]:8201"
             tls_disable = "true"
          }
          service_registration "kubernetes" {}
EOF
```

3. Install Vault using the override:

```shell helpm install -n vault vault bashicorp/vault –values helpm-vault-raft-values.yml –create-namespace

```
4. Wait until all pods reach the "Healthy" status before proceeding:

```shell kubectl -n vault get pods

edb_notranlate_5shell kubectl -n vault exec --stdin=true --tty=true vault-0 -- /bin/sh
```

2. Enable Kubernetes communication for the vault instance:

``` Shell Vault auth Enable kubernetes

edb_notranlate_7 Shell vault write auth/kubernetes/config token_reviewer_jwt= “$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)$” $kubernetes_host$ = $https$ : //{KUBERNETES_PORT_443_TCP_ADDR}:443 kubernetes_ca_cert=@/var/run/secrets/kubernetes.io/serviceaccount/ca.crt

```
4. Enable the *Transit* secrets engine to activate key management capabilities:
```

(次のページに続く)
```

(前のページからの続き)

```シェルボールトのシークレットはトランジットを有効にします
```

5. Create a new file that contains the Vault policy definition for the keys:

   In this example the file with the policy is called `tde.txt`.

```shell cat > /tmp/tde.txt << EOL path “transit/*” {Capacity = [“create”, “read”, “update”, “patch”, “delete”, “list”] } EOL
```

```
6.  Use the created file to update HashiCorp Vaults security policy:

``` シェルボールトポリシー書き込み tde /tmp/tde.txt
```

7. Review the default parameter values for the Transit engine (e.g. around key rotation, encryption method, etc.) and adjust according to your requirements. Alternatively, can use the vault write command during key creation in the following step to override the default values for a specific key.

8. Create a new key that uses the transit engine:

   Replace <my-tde-key> with a key name of your choice.

```shell vault write -f transit/keys/

edb_notranlate_12 Shell vault write auth/kubernetes/role/tde :raw-latex:`\bound`_service_account_names=“p-*” :raw-latex:`\bound`_service_account_namespaces=“p-*” :raw-latex:`policy`=tde ttl=72h

edb_notranlate_13 シェル終了 ```
```

Vault を構成し、キーを作成したので、PG AI コンソールのプロジェクトに追加します。

Vault キーのプロジェクトへの割り当て

コンソールを使用して、作成したキーをハイブリッドマネージャーのプロジェクトに追加します。

1. コンソールで、TDE 対応クラスターを作成するプロジェクトに移動します。プロジェクト内の複数のクラスターにキーを使用できますが、追加のハイブリッドマネージャープロジェクトごとに新しいキーをセットアップすることをお勧めします。

2. 左側のナビゲーションで、 **Settings** ☒ **Security** を選択し、 **+ Add a key** を選択します。

3. キーを構成する場所を選択します。

4. **選択したキー管理システムプロバイダー** の下で **HashiCorp Vault** を選択します。

5. **Vault address** に、HashiCorp Vault サーバーエンドポイントを入力します。

Vault が `vault` 名前空間、およびハイブリッドマネージャーと同じ Kubernetes 環境にインストールされている場合、Vault のデフォルトの Kubernetes エンドポイントは `http://vault.vault.svc.cluster.local:8200/` になります。設定が異なる場合は、次のように環境からエンドポイントを推定します<pod_name>.<namespace>.svc.cluster.<host>:<port_configured_for_vault> 。

6. **Vault Role** に、前に作成したロールの名前、`tde` を入力します。

7. **Vault Key** に、前に作成したキーの名前 <my-tde-key> を入力します。
```

8. 必要に応じて、 **(オプション) キーのフレンドリな名前を入力します**に、覚えやすい代替キーの名前を入力できます。

9. **Add Key** を選択します。

クラスターを作成するときに暗号化を有効にする場合、このキーを使用できるようになりました。追加されたキーは、*enable TDE during cluster creation* を実行するときにオプションとして表示されます。

4.1.2.8.3 AWS Key Management Service (KMS)

AWS Key Management Service キーを使用して TDE 対応クラスターを作成する前に、管理者は Hybrid Manager インスタンスの AWS KMS をインストールして構成する必要があります。

この例では、AWS で対称カスタマー管理キーを作成し、ハイブリッドマネージャーで KMS プロバイダーを構成して、そのキーを使用して TDE が有効にして、暗号化が有効になった新しいデータベースクラスターをプロビジョニングします。

Amazon Elastic Kubernetes ServiceEKS でハイブリッドマネージャーを実行している場合、AWS エコシステムのベストプラクティスとのネイティブ統合と調整を提供するため、キー管理に AWS KMS を使用することが現実的です。

前提条件

ハイブリッドマネージャーの `HybridControlPlane` CR で AWS KMS のパラメーターを設定していることを確認します。有効にするには、*Adding KMS support* を参照してください。

AWS KMS キーの作成

ハイブリッドマネージャーがインストールされているのと同じ AWS アカウントとリージョンで AWS コンソール、AWS CLI または AWS API を使用して Create a symmetric customer-managed key 。

新しいキーには次の構成を使用します。

- **キータイプ** 対称
- **キーの使用法** 暗号化と復号化
- **エイリアス**キーの使いやすいエイリアスを割り当てます機密情報は入力しないでください
- **キーポリシー** Kubernetes クラスターでハイブリッドマネージャー用に構成されたメイン IAM ユーザーまたはロールに、キーを暗号化および復号化する権限があることを確認します。

```
{
    "Sid": "Allow HM identity to use this KMS key",
    "Effect": "Allow",
    "Principal": {
        "AWS": "<IAM-user-or-role-arn>"
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt"
    ],
```

(次のページに続く)

(前のページからの続き)

```
    "Resource": "*"
}
```

顧客管理 KMS キーを作成したので、コンソールのプロジェクトに追加します。

AWS KMS キーのプロジェクトへの割り当て

作成したキーをプロジェクトに追加します。

1. コンソールで、TDE 対応クラスターを作成するプロジェクトに移動します。プロジェクト内の複数のクラスターにキーを使用できますが、追加のハイブリッドマネージャープロジェクトごとに新しいキーをセットアップすることをお勧めします。
2. 左側のナビゲーションで、 **Settings** ☒ **Security** を選択し、 **+ Add a key** を選択します。
3. キーを構成する場所を選択します。
4. **キー管理システムプロバイダーの選択** で **AWS** を選択します。
5. **Enter AWS Key Management System ARN** に、AWS KMS キー ARN を入力します。 AWS KMS ドキュメントの Find the key ID and key ARN を参照してください。
6. 必要に応じて、 **(オプション) キーのフレンドリな名前を入力します**に、覚えやすい代替キーの名前を入力できます。
7. **Add Key** を選択します。

クラスターを作成するときに暗号化を有効にする場合、このキーを使用できるようになりました。追加されたキーは、*enable TDE during cluster creation* を実行するときにオプションとして表示されます。

4.1.2.8.4 Azure Key Vault (Key Management Service)

Azure Key Vault キーを使用して TDE 対応クラスターを作成する前に、管理者は Hybrid Manager インスタンスの Azure Key Vault をインストールして構成する必要があります。

この例では、Azure Key Vault でカスタマー管理キーを作成し、ハイブリッドマネージャーで KMS プロバイダーを構成して、キーを使用して TDE が暗号化を有効にした新しいデータベースクラスターをプロビジョニングできるようにします。

Azure Kubernetes ServiceAKS でハイブリッドマネージャーを実行している場合、Azure エコシステムのベストプラクティスとのネイティブ統合と調整を提供するため、キー管理に Azure Key Vault を使用することが現実的です。

前提条件

ハイブリッドマネージャーの `HybridControlPlane` CR で Azure KMS のパラメーターを設定していることを確認します。有効にするには、*Adding KMS support* を参照してください。特に、`azure_kms` はエージェントの `transparentDataEncryptionMethods` リストに存在する必要があります。それ以外の場合、 **Azure** は、コンソールでキーを追加するときに**キー管理システムプロバイダーの選択**の下にオプションとして表示されません。

Azure Key Vault キーの作成

ハイブリッドマネージャーがインストールされているのと同じ Azure リージョンで Azure ポータル、Azure CLI、または Azure API を使用して Create a key in Azure Key Vault 。

新しいキーには次の構成を使用します。

- **キータイプ** RSA または EC ボールトでサポートされているとおり
- **キー操作**少なくとも次の許可された操作が有効になっていることを確認します。

Action	Purpose
encrypt / *decrypt*	Wrap and unwrap the data encryption key
wrapKey / *unwrapKey*	Key wrap operations
get / *read*	Read key metadata

キーへのワークロード ID アクセスの付与

Postgres クラスターは、Azure Key Vault キーによってバッキングされた TDE を使用します。クラスターのポッドに関連付けられているワークロード ID には、そのキーで暗号化操作を実行する権限が必要です。

ビルトインロール **Key Vault Crypto User** をワークロード ID に割り当てます。このロールは、TDE が必要とする操作のみを付与します。

Action	Purpose
Microsoft.KeyVault/vaults/key	Wrap the data encryption key
Microsoft.KeyVault/vaults/key	Unwrap the data encryption key
Microsoft.KeyVault/vaults/key	Key wrap operations
Microsoft.KeyVault/vaults/key	Key unwrap operations
Microsoft.KeyVault/vaults/key	Read key metadata

Azure CLI を使用してロールを割り当てます。

```
az role assignment create \
    --role "Key Vault Crypto User" \
    --assignee-object-id <WORKLOAD_IDENTITY_OBJECT_ID> \
    --assignee-principal-type ServicePrincipal \
    --scope "/subscriptions/<SUBSCRIPTION_ID>/resourceGroups/<RESOURCE_GROUP>/
↪providers/Microsoft.KeyVault/vaults/<VAULT_NAME>/keys/<KEY_NAME>"
```

割り当てを確認します。

```
az role assignment list \
    --assignee <WORKLOAD_IDENTITY_OBJECT_ID> \
    --scope "/subscriptions/<SUBSCRIPTION_ID>/resourceGroups/<RESOURCE_GROUP>/
↪providers/Microsoft.KeyVault/vaults/<VAULT_NAME>" \
    -o table
```

Role: Key Vault Crypto User を含む行が表示されます。

Azure Key Vault キーを作成したら、Hybrid Manager コンソールのプロジェクトに追加します。

Azure Key Vault キーのプロジェクトへの割り当て

作成したキーをプロジェクトに追加します。

1. コンソールで、TDE 対応クラスターを作成するプロジェクトに移動します。プロジェクト内の複数のクラスターにキーを使用できますが、追加のハイブリッドマネージャープロジェクトごとに新しいキーをセットアップすることをお勧めします。
2. 左側のナビゲーションで、 **Settings** -> **Security** を選択し、次に**+ Add a key** を選択します。
3. キーを構成する場所を選択します。
4. **選択したキー管理システムプロバイダー** で **Azure** を選択します。
5. **Key Vault Name** に、キーを保持する Azure Key Vault の名前を入力します。
6. **Key Name** に、Microsoft Azure アカウントのキーの名前を入力します。この名前は、Azure Key Vault のキーの名前と一致する必要があります。
7. **Key Version** に、暗号化と復号化に使用されるキーのバージョンを入力します。
8. 必要に応じて、 **(オプション) キーのフレンドリな名前を入力します**に、覚えやすい代替キーの名前を入力できます。
9. **Add Key** を選択します。

クラスターを作成するときに暗号化を有効にする場合、このキーを使用できるようになりました。追加されたキーは、*enable TDE during cluster creation* を実行するときにオプションとして表示されます。

4.1.2.8.5 Google Cloud Key Management Service (Cloud KMS)

Google Cloud Key Management サービスキーを使用して TDE 対応クラスターを作成する前に、管理者は Hybrid Manager インスタンスの Google Cloud Key Management をインストールして構成する必要があります。

この例では、Cloud KMS で顧客管理キーを作成し、ハイブリッドマネージャーでキープロバイダーを構成して、そのキーを使用して TDE が暗号化を有効にした新しいデータベースクラスターをプロビジョニングできるようにします。

Google Kubernetes EngineGKE でハイブリッドマネージャーを実行している場合、Google エコシステムのベストプラクティスとのネイティブ統合と調整を提供するため、キー管理に Cloud KMS を使用することが現実的です。

前提条件

ハイブリッドマネージャーの `HybridControlPlane` CR で Google Cloud KMS のパラメーターを設定していることを確認します。有効にするには、*Adding KMS support* を参照してください。

Google Cloud KMS キーの作成

ハイブリッドマネージャーがインストールされているプロジェクトとリージョンで Google Cloud コンソール、CLI、または API を使用して Create a customer-managed key 。キーは、特定の場所に関連付けられた 1 つのキーリングにのみ存在します。

新しいキーリングには次の構成を使用します。

- **キーリング名** キーリングの特定の名前を選択します
- **ロケーションタイプ** リージョン
- **リージョン**ハイブリッドマネージャーが展開されているのと同じリージョンを選択します

新しいキーリングに関連付けられたキーを作成します。

- **キー名** キーの特定の名前を選択します
- **保護レベル** ソフトウェア
- **目的** 対称暗号化/復号化

キーのデフォルト設定たとえば、キーローテーション、保護レベル、キーマテリアル、バージョンなどと adjust according to your requirements を確認します。

キーの権限を構成します。

- キー権限を編集し、ハイブリッドマネージャーが使用する Google サービスアカウントのアクセスを付与します。これは、ハイブリッドマネージャーのインストール中に構成されたアカウントです。
- 暗号化および復号化の権限が付与されていることを確認しますたとえば、Google ビルトインの Cloud KMS CryptoKey Encrypter および Decrypter ロールを使用します。

Google Cloud KMS キーが作成されたので、Hybrid Manager コンソールのプロジェクトに追加します。

Google Cloud KMS キーのプロジェクトへの割り当て

作成したキーをプロジェクトに追加します。

1. コンソールで、TDE 対応クラスターを作成するプロジェクトに移動します。プロジェクト内の複数のクラスターにキーを使用できますが、追加のハイブリッドマネージャープロジェクトごとに新しいキーをセットアップすることをお勧めします。
2. 左側のナビゲーションで、 **Settings** ☒ **Security** を選択し、 **+ Add a key** を選択します。
3. キーを構成する場所を選択します。
4. **選択したキー管理システムプロバイダー** で **Google Cloud** を選択します。
5. **Region** に、ハイブリッドマネージャーが展開されるリージョンを入力します。us-west-2.
6. **Project ID** に、キーリングを含むプロジェクトの ID を入力します。

7. **Keyring name** に、Google Cloud アカウントで定義されたキーリングの正確な名前を入力します。
8. **Key name** に、Google Cloud アカウントで定義された正確なキー名を入力します。
9. 必要に応じて、 **(オプション) キーのフレンドリな名前を入力します**に、覚えやすい代替キーの名前を入力できます。
10. **Add Key** を選択します。

クラスターを作成するときに暗号化を有効にする場合、このキーを使用できるようになりました。追加されたキーは、*enable TDE during cluster creation* を実行するときにオプションとして表示されます。

4.1.2.8.6 Deleting a key

キーが必要でなくなった場合は、削除できます。

以前にキーを追加したプロジェクトに移動します。左側のナビゲーションで、**Settings** ⊠ **Security** を選択し、キーを削除します。

!!!note

既存のクラスターの暗号化に使用されているキーは削除できません。

4.1.2.9 Optimizing service mesh memory

Hybrid Manager は、サービスメッシュとして Istio を使用して、Postgres クラスターと他のコンポーネント間のネットワークトラフィックを管理します。管理対象 Postgres クラスターの数が増加すると、Istio のメモリ要件が増加します。Istio のメモリ制限を調整して、予想されるワークロードに基づいてパフォーマンスを最適化できます。

4.1.2.9.1 Istio のメモリ要件を理解する

Istio のメモリ要件は、いくつかの要因によって異なります。

- **Postgres クラスターの数**クラスターが増えると、サービス検出とルーティングにより多くのメモリが必要になります
- **クラスターアーキテクチャ**複雑なアーキテクチャ複数のデータグループを持つ PGD クラスターなどは、単純なスタンドアロンクラスターよりも多くのメモリを必要とします
- **ネットワークトラフィック**ネットワークアクティビティが増加すると、メモリ使用量が増加します

デフォルトでは、ハイブリッドマネージャーは控えめなメモリ制限で Istio を構成します。環境の規模が拡大すると、これらの制限を調整して、パフォーマンスの低下またはメモリ不足エラーを防ぐ必要がある場合があります。

4.1.2.9.2 推奨されるメモリ制限

次の表は、管理する予定の Postgres クラスターの数に基づいた推奨メモリ制限を示しています。

Number of Postgres Clusters	Recommended Memory Limit
up to 10-20 clusters	512Mi
up to 30-40 clusters	1Gi
up to 60-70 clusters	1.5Gi
scale more if you need	> 1.5Gi

注釈

これらの推奨事項はガイドラインです。実際の要件は、クラスターアーキテクチャとネットワークトラフィックパターンによって異なる場合があります。 Istio のメモリ使用量を監視し、それに応じて調整します。 !!!

4.1.2.9.3 Istio メモリ制限の構成

`upm-istio` コンポーネントの `memory_limit` および `cpu_limit` 値を設定することにより、Istio メモリ制限を構成できます。

Tip

予想されるクラスター数の推奨値から始めます。 70 クラスターを超えてスケールする予定がある場合は、メモリ制限を比例して増やしますたとえば、追加 30 クラスターごとに約 512Mi 追加します。 !!!

`HybridControlPlane` CR を編集して、`spec.componentsParameters.upm-istio` の下の Istio メモリ設定を追加または更新します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  componentsParameters:
    upm-istio:
      # Memory limit recommendations:
      # - 512Mi for 10-20 Postgres clusters
      # - 1Gi for 30-40 Postgres clusters
      # - 1.5Gi for 60-70 Postgres clusters
      # Adjust based on your expected load and architecture
      memory_limit: "512Mi"
      cpu_limit: "500m"
    # ... your other componentsParameters
```

更新された CR を適用します。

```
kubectl apply -f hybridmanager.yaml
```

オペレーターは変更を調整し、Istio リソース制限を更新します。

4.1.2.9.4 Istio のメモリ使用量のモニタリング

メモリ制限を調整した後、Istio のパフォーマンスを監視して、構成がワークロードに適切であることを確認します。

Grafana ダッシュボードを使用する

ハイブリッドマネージャーの Launchpad で使用可能な Grafana ダッシュボードを介して Istio のメモリ使用量を監視します。

1. Hybrid Manager ポータルから、ユーザーメニューを開き、 **Launchpad** を選択します。
2. **Grafana** カードで、 **View service** を選択します。
3. **Dashboards > Kubernetes** に移動します。
4. **Kubernetes / Compute Resources / Workload** ダッシュボードを選択します。
5. フィルター

- **namespace** `istio-system`
- **workload** `istiod` Istio コントロールプレーンの場合

メモリ使用量パネルを監視して、使用量が一貫して構成した制限に近づいているかどうかを確認します。その場合、`memory_limit` 値を増やすことを検討します。

メモリ関連の問題の確認

Istio でメモリプレッシャーが発生している場合、次の症状が表示される場合があります。

- ポッドが頻繁に再起動します
- ポッドの説明の OOMKilled Out of Memory Killed イベント
- Postgres クラスター操作の応答時間の低下

ポッドのステータスとイベントを確認するには

```
kubectl get pods -n istio-system
kubectl describe pod <pod-name> -n istio-system
```

出力で `OOMKilled` またはメモリ関連のエラーを記載したイベントを探します。

4.1.2.10 API audit logging

EDB Postgres ® AI Hybrid ManagerHM は、イングレスゲートウェイを通過する認証された API 要求の構造化監査ログを生成できます。有効にすると、認証を必要とするすべての API 呼び出しで、誰が何に、い

つ、どこからアクセスしたかの詳細な記録を提供する JSON ログエントリが生成されます。この機能は、コンプライアンスレポート、セキュリティインシデントの調査、およびアクセス監査に役立ちます。

API 監査ログは、HM 2026.4 から利用できます。

4.1.2.10.1 監査の対象となるもの

HM は、次の条件が両方とも満たされる場合、要求の監査ログエントリを発行します。

- 要求には認証が必要です。匿名または除外エンドポイントは監査されません。
- 要求は、REST API /api/…、Iceberg カタログ API、launchpad アプリケーション API、または gRPC サービスエンドポイントなどの API パスをターゲットとしています。

次の要求タイプは監査されません。

- 静的リソース例 JavaScript、CSS、画像
- /api/ui および/api-docs を含む除外パス

4.1.2.10.2 API 監査ログの無効化

API 監査ログはデフォルトで有効になっています。これを無効にするには、インストール中に upm-istio-gateway コンポーネントの enable_api_audit 設定を"false" に設定します。

> **注釈**
>
> 受け入れられる値は、文字列`"true"` および`"false"` のみです。ブール型は使用できません。 !!!

HybridControlPlane カスタムリソースを編集して、spec.componentsParameters.upm-istio-gateway の下の enable_api_audit 設定を追加または更新します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  componentsParameters:
    upm-istio-gateway:
      enable_api_audit: "false"
```

更新されたカスタムリソースを適用します。

```
kubectl apply -f hybridmanager.yaml
```

4.1.2.10.3 API 監査ログを再度有効にする

API 監査ログを無効にした後、再度有効にするには、enable_api_audit を"true" に設定し、同じ手順を使用して再適用します。

4.1.2.10.4 監査ログの照会

監査ログエントリは、イングレスゲートウェイの標準ログ出力に書き込まれます。EDB-HM-AUDIT キーでフィルタリングすることにより、クラスターのログ集計ツールを使用してそれらを照会できます。

Grafana での監査ログの表示

Grafana は、HM の可観測性スタックに含まれており、Loki によって集計されたログを照会します。HM コンソールの *Launchpad* からサービスとしてアクセスできます。

> **注釈**
>
> Launchpad と Grafana にアクセスするには、プラットフォーム管理者ロールが必要です。詳細は、*User role permissions by use case* を参照してください。 !!!

Grafana で監査ログを表示するには

1. HM コンソールのユーザーメニューから、 **Launchpad** を選択します。
2. **Grafana** を選択します。
3. Grafana で、左側のペインから **Explore** を選択します。
4. **Loki** データソースを選択します。
5. LogQL クエリーを実行して、監査エントリをフィルタリングします。例

```
{namespace="istio-system", container="istio-proxy"} |= "EDB-HM-AUDIT" | json
```

kubectl を使用する

利用可能なログ集計ツールがない場合は、次のコマンドを使用してイングレスゲートウェイポッドからログを直接ストリーミングし、簡単な検査を行います。

```
kubectl logs -n istio-system -l istio=ingressgateway -c istio-proxy | grep EDB-HM-
↪AUDIT
```

4.1.2.10.5 ログフォーマット

各監査ログエントリは、EDB-HM-AUDIT キーの下にネストされた JSON オブジェクトです。このキーを使用して、Loki などのログ集計システムの監査エントリをフィルタリングできます。

ログエントリの例

```
{
  "EDB-HM-AUDIT": {
    "method": "GET",
    "uri": "/api/v1/projects?page=1",
    "user_id": "usr_abc123",
```

(次のページに続く)

(前のページからの続き)

```
    "user_name": "alice",
    "key_id": "ak_def456",
    "session_id": "aksid_a3f8c2e1b9d04f67",
    "request_id": "a1b2c3d4-e5f6-7890",
    "user_agent": "curl/7.68.0",
    "ip": "10.0.0.1",
    "status_code": 200
  }
}
```

4.1.2.10.6 ログフィールド

各監査ログエントリには、次のフィールドが含まれます。ノイズを削減するために、空の値または使用できない値を含むフィールドは省略されます。

フィールド |説明 ——|——– – method | HTTP メソッド GET 、POST 、PUT 、DELETE など。 uri |元の要求 URI、クエリ文字列を含む user_id |認証されたユーザー ID。認証が失敗した場合は空 401 応答など。user_name |認証されたユーザーの表示名。 key_id |アクセスキー ID。アクセスキーベースの認証の場合にのみ存在します。ブラウザまたは OIDC セッションの場合は省略されます。session_id |セッション識別子。アクセスキー要求の場合、このフィールドは、キーとクライアントコンテキストから派生した決定論的ハッシュであり、同じ自動クライアントからの要求間の相関を有効にします。ブラウザセッションの場合、このフィールドは OIDC セッション ID です。request_id |一意の要求トレース ID x-request-id 。監査エントリを上流のサービスログと相関させるのに役立ちます。user_agent |クライアントの User-Agent ヘッダー値。 ip | x-forwarded-for ヘッダーから抽出されたクライアント IP アドレス。status_code | HTTP 応答ステータスコードたとえば、200、401、403、500 など。grpc_status | gRPC ステータスコード。gRPC 要求の場合にのみ存在します。HTTP トランスポートは常に 200 を返します。実際の結果はここに反映されますたとえば、16 = 未認証。

4.1.3 Migrating from bootstrap to operator installation

4.1.3.1 概要

以前に **ブートストラップ方法** values.yaml を使用した edbpgai-bootstrap Helm チャートを使用してハイブリッドマネージャーをインストールし、**オペレーター方法** HybridControlPlane CR を使用した edb-hcp-operator Helm チャートに移行する場合、このガイドでは必要な手順を提供します。

ブートストラップをアンインストールするだけで、**すべての CRD カスタムリソース定義** が削除され、**カスケードしてすべてのデータベースクラスターを削除し、データを破棄します**。データの損失を回避するには、このガイドの CRD を保持した移行手順に従ってください。

4.1.3.2 移行の課題を理解する

4.1.3.2.1 Helm の所有権の競合

既存のブートストラップインストールに加えてオペレーターをインストールしようとすると、次のようなエラーが発生します。

```
Release "edb-hcp-operator" does not exist. Installing it now.
Error: Unable to continue with install: ServiceAccount "edb-hcp-operator-controller-
↪manager"
in namespace "edbpgai-bootstrap" exists and cannot be imported into the current␣
↪release:
invalid ownership metadata; annotation validation error: key "meta.helm.sh/release-
↪name"
must equal "edb-hcp-operator": current value is "edbpgai-bootstrap"
```

4.1.3.2.2 これが発生する理由

- ブートストラップインストールは、edbpgai-bootstrap Helm リリースが所有する Kubernetes リソースを作成します
- Operator インストールは、edb-hcp-operator Helm リリースが所有するリソースを作成しようとします
- Helm は、あるリリースが別のリリースのリソースの所有権を取得しないようにします
- どちらのインストールも、重複する名前空間とリソース名を使用します

4.1.3.2.3 CRD 削除問題

ブートストラップ helm チャートをアンインストールすると、**すべてのカスタムリソース定義 CRD が削除されます** 。これは**カスケードしてすべてのデータベースクラスターとカスタムリソースを削除します**。データが失われないように、これは回避する必要があります。

- ハイブリッドコントロールプレーン
- プリフライト
- ポストフライト

これらの CRD が削除されると、Kubernetes はこれらのカスタムリソースの**すべてのインスタンス**を自動的に削除し、HM プラットフォームを機能しなくします。

4.1.3.3 edbctl で移行の準備をする

ブートストラップインストールからオペレーターインストールに移行するための推奨される方法は、edbctl hm prepare-bootstrap-operator-conversion です。このコマンドは、単一の呼び出しで移行の両方の半分を実行します。

1. 現在の HybridControlPlane CR をローカル YAML ファイルデフォルト hybridmanager.yaml に**エクスポート**して、オペレーター管理マニフェストの開始点を設けます。
2. ブートストラップインストールされたリソース CRD、クラスターロール、クラスターロールバインディング、ロール、ロールバインディング、サービスアカウント、サービス、webhook 構成の **Helm 所有権を移行する** ため、後続の helm install /edbctl hm upgrade-operator は所有権エラーなしで既存のリソースをきれいに復元します。

CRD とカスタムリソースは保持されます。データベースクラスターは、移行中に実行され続けます。

4.1.3.3.1 変換を実行します

kubeconfig がブートストラップインストールされた HM が存在するクラスターをポイントした状態で、次を実行します。

```
edbctl hm prepare-bootstrap-operator-conversion
__OUTPUT__
Migration prepared. Next steps:
  1. Review the generated CR: hybridmanager.yaml
  2. Install the operator chart for release "edb-hcp-operator" into namespace
↪"edbpgai-bootstrap" — the transfer
     stamped Helm ownership for that namespace, so install there (not elsewhere):
       helm install edb-hcp-operator <chart> -n edbpgai-bootstrap
     or install the latest operator with edbctl:
       edbctl hm upgrade-operator --release-name edb-hcp-operator --namespace edbpgai-
↪bootstrap --registry-uri <uri> --registry-username <user> --registry-password
↪<password>
  3. Apply the reviewed CR: kubectl apply -f hybridmanager.yaml
```

デフォルト

- `--namespace edbpgai-bootstrap` —ブートストラップリリースが存在する名前空間。
- `--operator-namespace <same as --namespace>` —オペレーターチャートがインストールされる名前空間。所有権の移転はこの名前空間をスタンプするため、後続の `helm install` /`edbctl hm upgrade-operator` はそれをターゲットとする **必要** 。
- `--source-release edbpgai-bootstrap` —現在リソースを所有している Helm リリース。
- `--target-release edb-hcp-operator` — 所有権を譲渡する Helm リリース。
- `--output hybridmanager.yaml` —エクスポートされた CR が書き込まれるファイル。

インストールでデフォルト以外の名前を使用している場合、それらのいずれかをオーバーライドします。`--dry-run` を使用して、クラスターを変更せずに削除とアノテーションをプレビューします。

> **重要**
>
> 生成された `hybridmanager.yaml` を適用する前に確認します。特に

- **``spec.version``** ライブ CR から派生。オペレーターは独自のバージョンスキームを使用します。`spec.version` は、HM ブートストラップバージョンたとえば `v1.4.0` であり、オペレーター独自のバージョン `2.x` リネージの `2.0.0` 以降 ではありません。
- **``spec.imagePullSecrets``** 各エントリには `name` と `namespace` の両方が必要です。シークレットを `edbpgai-bootstrap` の外部に保存する場合、それらが正しい名前空間を参照していることを確認します。

1. 生成された `hybridmanager.yaml` を確認します。
2. 変換がスタンプした名前空間にオペレーターチャート —`edbctl hm upgrade-operator`

`--namespace edbpgai-bootstrap --registry-uri <REGISTRY_URI> --registry-username <USER> --registry-password <PASSWORD>` レジストリから最新のオペレーターバージョンを自動検出します。 edbctl `< 1.7.0` では、`helm install edb-hcp-operator <chart> -n edbpgai-bootstrap` にフォールバックします。

3. 確認した CR `kubectl apply -f hybridmanager.yaml` を適用します。エクスポートされたファイルは、ライブ CR がまだ保持しているブートストラップリリースの `meta.helm.sh/release-name=edbpgai-bootstrap` アノテーションを意図的に省略しています。ファイルを適用すると、その古いアノテーションが上書きされるため、後の `helm uninstall edbpgai-bootstrap` ブートストラップリリースのクリーンアップは、HybridControlPlane CR およびデータベースはそれでクラスター化されません。

4.1.3.4 1.7.0 未満の edbctl の手動フォールバック

edbctl < `1.7.0` がまだある環境の場合、以前の手動フローは引き続き機能します。このガイドの残りの部分では、参照用にこれを保存しますが、上記の `edbctl hm prepare-bootstrap-operator-conversion` パスは推奨されるパスです。

4.1.3.4.1 構成マッピングリファレンス

次の表は、ブートストラップ `values.yaml` から CR を手書きする必要がある場合に、`values.yaml` フィールドが `HybridControlPlane` CR フィールドにどのようにマッピングするかを示しています。

values.yaml field	HybridControlPlane field	Description
system	*spec.flavour*	Kubernetes ディストリビューション eks、aks、gke、rhos など
containerRegistryURL	*spec.imageRegistry*	コンテナレジストリ URL
bootstrapImageTag	*spec.version*	インストールする HM バージョン
imagePullSecrets	*spec.imagePullSecrets[]*	イメージプルシークレット参照。`values.yaml`では、これはシークレット名文字列のリストです。CR では、各エントリに `name`および `namespace`フィールドが必要です。
disabledComponents	*spec.disabledComponents[]*	無効にするコンポーネント
*parameters.global.**	*spec.globalParameters*	グローバル構成パラメーター
*parameters.<component>.**	*spec.componentsParameters.<*	コンポーネント固有のパラメーター
beaconServer	*spec.beaconServer*	ビーコンサーバーの構成
beaconAgent	*spec.beaconAgent*	ビーコンエージェントの構成
clusterGroups	*spec.clusterGroups*	マルチ DC クラスタートポロジ
scenarios	*spec.scenarios[]*	インストールシナリオコア、移行、ai、分析。 *values.yaml* **では、これはコンマ区切りの文字列たとえば** *`"core,migration,ai"* 。 CR では、それは YAML リストです。
resourceAnnotations	*spec.resourceAnnotations[]*	カスタムリソースアノテーション

次のページに続く

表 44 – 前のページからの続き

values.yaml field	HybridControlPlane field	Description
source.remote	*spec.source.remote*	マニフェストにリモートブートストラップイメージを使用する

4.1.3.4.2 移行スクリプトを使用して変換する

次のスクリプトを使用して、既存の values.yaml から HybridControlPlane CR を自動的に生成します。

前提条件 yq バージョン 4+をインストールする必要があります。

```
# !/bin/bash
#  convert-values-to-hcp.sh
#  Converts a bootstrap values.yaml to a HybridControlPlane Custom Resource manifest.
#
#  Usage: ./convert-values-to-hcp.sh <values.yaml> [output-file]
#    If output-file is omitted, prints to stdout.

set -euo pipefail

VALUES_FILE="${1:?Usage: $0 <values.yaml> [output-file]}"
OUTPUT_FILE="${2:-}"

if [ ! -f "$VALUES_FILE" ]; then
  echo "Error: File $VALUES_FILE not found." >&2
  exit 1
fi

if ! command -v yq &> /dev/null; then
  echo "Error: yq (v4+) is required. Install from https://github.com/mikefarah/yq" >&2
  exit 1
fi

#  Build the HybridControlPlane CR using yq

HCP=$(yq -n
  .apiVersion = "edbpgai.edb.com/v1alpha1" |
  .kind = "HybridControlPlane" |
  .metadata.name = "edbpgai" |
  .metadata.annotations."biganimal.enterprisedb.io/deletion-protect" = "enabled"
)

#  Basic configuration mapping

SYSTEM=$(yq .system // "" "$VALUES_FILE")
[ -n "$SYSTEM" ] && HCP=$(echo "$HCP" | yq ".spec.flavour = \"$SYSTEM\"")
```

(次のページに続く)

(前のページからの続き)

```
REGISTRY=$(yq .containerRegistryURL // "" "$VALUES_FILE")
[ -n "$REGISTRY" ] && HCP=$(echo "$HCP" | yq ".spec.imageRegistry = \"$REGISTRY\"")

VERSION=$(yq .bootstrapImageTag // "" "$VALUES_FILE")
[ -n "$VERSION" ] && HCP=$(echo "$HCP" | yq ".spec.version = \"$VERSION\"")

#  Source configuration

SOURCE_REMOTE=$(yq .source.remote // "" "$VALUES_FILE")
if [ -n "$SOURCE_REMOTE" ] && [ "$SOURCE_REMOTE" = "true" ]; then
  HCP=$(echo "$HCP" | yq ".spec.source.remote = true")
else
  HCP=$(echo "$HCP" | yq ".spec.source.useLocalKustomizations = true")
fi

#  Image pull secrets

PULL_SECRETS_COUNT=$(yq .imagePullSecrets | length // 0 "$VALUES_FILE" 2>/dev/null ||␣
↪echo "0")
if [ "$PULL_SECRETS_COUNT" -gt 0 ]; then
  SECRETS_JSON=$(yq -o=json .imagePullSecrets | [.[] | select(. != null and . != "")␣
↪| {"name": ., "namespace": "edbpgai-bootstrap"}] "$VALUES_FILE")
  HCP=$(echo "$HCP" | yq ".spec.imagePullSecrets = $SECRETS_JSON")
fi

#  Global parameters (parameters.global.* -> spec.globalParameters)

GLOBAL_PARAMS=$(yq .parameters.global // "" "$VALUES_FILE")
if [ -n "$GLOBAL_PARAMS" ] && [ "$GLOBAL_PARAMS" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.globalParameters = $(yq -o=json .parameters.global "
↪$VALUES_FILE")")
  [ -n "$REGISTRY" ] && HCP=$(echo "$HCP" | yq ".spec.globalParameters.CONTAINER_
↪REGISTRY_URL = \"$REGISTRY\"")
  [ -n "$VERSION" ] && HCP=$(echo "$HCP" | yq ".spec.globalParameters.HCP_VERSION = \"
↪$VERSION\"")
fi

#  Component-specific parameters (parameters.<component>.* -> spec.
↪componentsParameters.<component>)

COMPONENTS=$(yq .parameters | keys | .[] | select(. != "global") "$VALUES_FILE" 2>/
↪dev/null || true)
if [ -n "$COMPONENTS" ]; then
  while IFS= read -r component; do
```

(次のページに続く)

(前のページからの続き)

```
    HCP=$(echo "$HCP" | yq ".spec.componentsParameters.[\"$component\"] = $(yq -
↪o=json ".parameters.[\"$component\"]" "$VALUES_FILE")")
  done <<< "$COMPONENTS"
fi

#  Beacon server configuration

BEACON_SERVER=$(yq .beaconServer // "" "$VALUES_FILE")
if [ -n "$BEACON_SERVER" ] && [ "$BEACON_SERVER" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.beaconServer = $(yq -o=json .beaconServer "$VALUES_
↪FILE")")
fi

#  Beacon agent configuration

BEACON_AGENT=$(yq .beaconAgent // "" "$VALUES_FILE")
if [ -n "$BEACON_AGENT" ] && [ "$BEACON_AGENT" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.beaconAgent = $(yq -o=json .beaconAgent "$VALUES_FILE
↪")")
fi

#  Cluster groups

CLUSTER_GROUPS=$(yq .clusterGroups // "" "$VALUES_FILE")
if [ -n "$CLUSTER_GROUPS" ] && [ "$CLUSTER_GROUPS" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.clusterGroups = $(yq -o=json .clusterGroups "$VALUES_
↪FILE")")
fi

#  Installation scenarios

SCENARIOS=$(yq .scenarios // "" "$VALUES_FILE")
if [ -n "$SCENARIOS" ] && [ "$SCENARIOS" != "null" ]; then
  SCENARIOS_JSON=$(echo "$SCENARIOS" | tr , \n | sed s/^[[:space:]] *//;s/
↪[[:space:]]* $// | yq -o=json [.[] | select(. != "")] -)
  HCP=$(echo "$HCP" | yq ".spec.scenarios = $SCENARIOS_JSON")
fi

#  Disabled components

DISABLED=$(yq .disabledComponents // "" "$VALUES_FILE")
if [ -n "$DISABLED" ] && [ "$DISABLED" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.disabledComponents = $(yq -o=json .disabledComponents
↪"$VALUES_FILE")")
fi
```

(次のページに続く)

(前のページからの続き)

```
#  Resource annotations

ANNOTATIONS=$(yq .resourceAnnotations // "" "$VALUES_FILE")
if [ -n "$ANNOTATIONS" ] && [ "$ANNOTATIONS" != "null" ]; then
  HCP=$(echo "$HCP" | yq ".spec.resourceAnnotations = $(yq -o=json .
↪resourceAnnotations "$VALUES_FILE")")
fi

#  Output

if [ -n "$OUTPUT_FILE" ]; then
  echo "$HCP" > "$OUTPUT_FILE"
  echo "HybridControlPlane CR written to: $OUTPUT_FILE"
else
  echo "$HCP"
fi
```

使用法

```
#  Print the converted CR to stdout:

chmod +x convert-values-to-hcp.sh
./convert-values-to-hcp.sh values.yaml

#  Write to a file:

./convert-values-to-hcp.sh values.yaml hybridmanager.yaml
```

重要

生成されたマニフェストを適用する前に、注意深く確認します。

- **``spec.version``** ブートストラップ bootstrapImageTag は、オペレーター spec.version とは異なるバージョン管理スキームを使用する場合があります。
- **``spec.imagePullSecrets``** このスクリプトでは、すべてのシークレットが edbpgai-bootstrap 名前空間にあることを前提としています。シークレットが別の名前空間にある場合は調整します。

4.1.3.4.3 Helm 所有権の移転マニュアル

このアプローチは、既存のリソースの Helm 所有権を edbpgai-bootstrap リリースから edb-hcp-operator リリースに移行します。これにより、オペレーターは、何も再作成せずに既存のインフラストラクチャを採用および管理できます。

> **注釈**
>
> このメソッドは、**すべてのリソース** CRD、RBAC、Webhook、データベースクラスターを保持し、Helm 所有権のアノテーションを単に変更します。データベースクラスターは、移行中は実行されたままで、オペレーターはシームレスに管理を引き継ぎます。 !!!

4.1.3.4.4 仕組み

移行は、`edbpgai-bootstrap` から `edb-hcp-operator` までのすべてのリソースの `meta.helm.sh/release-name` アノテーションを変更することにより機能します。これは、これらのリソースがブートストラップリリースではなくオペレーターリリースに属することを Helm に示します。

このアプローチの主な利点

1. **リソースの削除なし** - CRD、ハイブリッドコントロールプレーン、プリフライト、およびポストフライトのリソースはそのままです
2. **Helm 競合なし** - オペレーターは、所有権エラーなしで既存のリソースを採用できます。
3. **最小限のダウンタイム** - 移行中に HM オペレーターポッドのみが影響を受けます。
4. **単純なロールバック** - 必要に応じてアノテーションを転送して戻すことができます
5. **クリーンな移行** - 検証後、ブートストラップ名前空間は安全に削除できます

転送されるもの

- カスタムリソース定義 CRD
- クラスターロールとクラスターロールバインディング
- ブートストラップ名前空間のロールとロールバインディング
- サービスアカウント
- サービス
- Webhook 設定

4.1.3.4.5 前提条件

- kubectl および helm CLI ツールへのアクセス
- `jq` コマンドライン JSON プロセッサーがインストールされている

次のスクリプトは、すべての移行手順を順番に実行します。単一のスクリプトとして実行するか、各ステップを手動で実行できます。

```
# !/bin/bash
set -e

echo "Starting migration from bootstrap to operator installation..."

echo "Step 1: Deleting bootstrap deployments..."
```

(次のページに続く)

(前のページからの続き)

```
kubectl delete deployments -n edbpgai-bootstrap --all

echo "Step 2: Transferring Helm ownership of ServiceAccount..."
kubectl annotate sa edb-hcp-operator-controller-manager \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite \
  -n edbpgai-bootstrap

echo "Step 3: Transferring Helm ownership of CRDs..."
for line in $(kubectl get crd | grep edbpgai.edb.com | awk {print $1}); do
  kubectl annotate crd $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done

echo "Step 4: Transferring Helm ownership of ClusterRoles..."
for line in $(kubectl get clusterroles -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate clusterrole $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done

echo "Step 5: Transferring Helm ownership of ClusterRoleBindings..."
for line in $(kubectl get clusterrolebindings -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate clusterrolebinding $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done

echo "Step 6: Transferring Helm ownership of Roles..."
for line in $(kubectl get roles -n edbpgai-bootstrap -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate role $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done

echo "Step 7: Transferring Helm ownership of RoleBindings..."
for line in $(kubectl get rolebindings -n edbpgai-bootstrap -o json | \
```

(次のページに続く)

(前のページからの続き)

```
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate rolebinding $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done

echo "Step 8: Transferring Helm ownership of Services..."
for line in $(kubectl get svc -n edbpgai-bootstrap -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate svc $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done

echo "Step 9: Transferring Helm ownership of Webhook Configurations..."
kubectl annotate mutatingwebhookconfiguration \
  edb-hcp-operator-mutating-webhook-configuration \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite

kubectl annotate validatingwebhookconfiguration \
  edb-hcp-operator-validating-webhook-configuration \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite

echo "migration completed successfully! Now the operator can be installed without␣
↪conflicts."
```

4.1.3.4.6 ステップ 1 ブートストラップ展開を削除する

ブートストラップ名前空間内のすべての展開を削除します。これにより、ブートストラップコントローラが停止します。

```
#  Delete all deployments in the bootstrap namespace

kubectl delete deployments -n edbpgai-bootstrap --all
__OUTPUT__
deployment.apps "edb-hcp-operator-controller-manager" deleted
deployment.apps "file-server" deleted
```

```
#  Verify deployments are removed

kubectl get deployments -n edbpgai-bootstrap
__OUTPUT__
No resources found in edbpgai-bootstrap namespace.
```

4.1.3.4.7 ステップ 2 Helm の所有権を譲渡する - ServiceAccount

サービスアカウントの所有権を転送します アノテーション

```
#  Re-annotate the service account to change Helm ownership

kubectl annotate sa edb-hcp-operator-controller-manager \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite \
  -n edbpgai-bootstrap
__OUTPUT__
serviceaccount/edb-hcp-operator-controller-manager annotated
```

4.1.3.4.8 ステップ 3 Helm の所有権を譲渡する - CRD

edbpgai.edb.com の下のすべての CRD の所有権を譲渡します。

```
#  Update Helm release annotation for all edbpgai.edb.com CRDs

for line in $(kubectl get crd | grep edbpgai.edb.com | awk {print $1}); do
  kubectl annotate crd $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done
__OUTPUT__
customresourcedefinition.apiextensions.k8s.io/hybridcontrolplanes.edbpgai.edb.com␣
↪annotated
customresourcedefinition.apiextensions.k8s.io/postflights.edbpgai.edb.com annotated
customresourcedefinition.apiextensions.k8s.io/preflights.edbpgai.edb.com annotated
```

```
#  Verify the annotations were updated

kubectl get crd | grep edbpgai.edb.com | head -5 | while read crd rest; do
  echo "CRD: $crd"
  kubectl get crd $crd -o jsonpath={.metadata.annotations.meta\.helm\.sh/release-name}
  echo ""
done
__OUTPUT__
CRD: hybridcontrolplanes.edbpgai.edb.com
```

(次のページに続く)

(前のページからの続き)

```
edb-hcp-operator
CRD: postflights.edbpgai.edb.com
edb-hcp-operator
CRD: preflights.edbpgai.edb.com
edb-hcp-operator
```

4.1.3.4.9 ステップ 4Helm の所有権を譲渡する- クラスターロール

クラスターロールの所有権を譲渡します。

```
#  Update Helm release annotation for ClusterRoles

for line in $(kubectl get clusterroles -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate clusterrole $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done
__OUTPUT__
clusterrole.rbac.authorization.k8s.io/create-hcp annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-hybridcontrolplane-editor-role␣
↪annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-hybridcontrolplane-viewer-role␣
↪annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-manager-role annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-metrics-auth-role annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-metrics-reader annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-postflight-admin-role annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-postflight-editor-role␣
↪annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-postflight-viewer-role␣
↪annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-preflight-admin-role annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-preflight-editor-role annotated
clusterrole.rbac.authorization.k8s.io/edb-hcp-operator-preflight-viewer-role annotated
```

4.1.3.4.10 ステップ 5Helm の所有権を譲渡する- ClusterRoleBindings

クラスターロールバインディングの所有権を譲渡します。

```
#  Update Helm release annotation for ClusterRoleBindings

for line in $(kubectl get clusterrolebindings -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
```

(次のページに続く)

(前のページからの続き)

```
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate clusterrolebinding $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite
done
__OUTPUT__
clusterrolebinding.rbac.authorization.k8s.io/create-hcp annotated
clusterrolebinding.rbac.authorization.k8s.io/edb-hcp-operator-manager-rolebinding␣
↪annotated
clusterrolebinding.rbac.authorization.k8s.io/edb-hcp-operator-metrics-auth-
↪rolebinding annotated
```

4.1.3.4.11 ステップ 6Helm の所有権を譲渡する - ロール

ブートストラップ名前空間のロールの所有権を譲渡します。

```
#  Update Helm release annotation for Roles

for line in $(kubectl get roles -n edbpgai-bootstrap -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate role $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done
__OUTPUT__
role.rbac.authorization.k8s.io/edb-hcp-operator-leader-election-role annotated
```

4.1.3.4.12 ステップ 7Helm の所有権を譲渡する-ロールバインディング

ブートストラップ名前空間の RoleBinding の所有権を譲渡します。

```
#  Update Helm release annotation for RoleBindings

for line in $(kubectl get rolebindings -n edbpgai-bootstrap -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate rolebinding $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done
__OUTPUT__
rolebinding.rbac.authorization.k8s.io/edb-hcp-operator-leader-election-rolebinding␣
```

(次のページに続く)

(前のページからの続き)

```
↪annotated
```

4.1.3.4.13 ステップ 8 Helm の所有権を譲渡する - サービス

ブートストラップ名前空間のサービスの所有権を譲渡します。

```
#  Update Helm release annotation for Services

for line in $(kubectl get svc -n edbpgai-bootstrap -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] ==
↪"edbpgai-bootstrap") | .metadata.name); do
  kubectl annotate svc $line \
    meta.helm.sh/release-name=edb-hcp-operator \
    --overwrite \
    -n edbpgai-bootstrap
done
__OUTPUT__
service/edb-hcp-operator-controller-manager-metrics-service annotated
service/edb-hcp-operator-webhook-service annotated
```

4.1.3.4.14 ステップ 9 Helm の所有権を譲渡する - Webhook 構成

Webhook 構成の所有権を譲渡します。

```
#  Update MutatingWebhookConfiguration

kubectl annotate mutatingwebhookconfiguration \
  edb-hcp-operator-mutating-webhook-configuration \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite
__OUTPUT__
mutatingwebhookconfiguration.admissionregistration.k8s.io/edb-hcp-operator-mutating-
↪webhook-configuration annotated
```

```
#  Update ValidatingWebhookConfiguration

kubectl annotate validatingwebhookconfiguration \
  edb-hcp-operator-validating-webhook-configuration \
  meta.helm.sh/release-name=edb-hcp-operator \
  --overwrite
__OUTPUT__
validatingwebhookconfiguration.admissionregistration.k8s.io/edb-hcp-operator-
↪validating-webhook-configuration annotated
```

4.1.3.4.15 ステップ 10 所有権の移転を確認する

リソースに正しい Helm 所有権があることを確認します。

```
#  Check a sample of resources

echo "Checking CRD ownership:"
kubectl get crd | grep edbpgai.edb.com | head -1 | awk {print $1} | xargs -I {} \
  kubectl get crd {} -o jsonpath={.metadata.annotations.meta\.helm\.sh/release-name}
echo ""

echo "Checking ClusterRole ownership:"
kubectl get clusterroles -o json | \
  jq -r .items[] | select(.metadata.annotations["meta.helm.sh/release-name"] == "edb-
↪hcp-operator") | .metadata.name | head -3
```

4.1.3.5 サポート

移行中に問題が発生した場合

- *troubleshooting guide* を確認します

*移行ログとエラーメッセージについて EDB サポートに連絡してください

*複雑な移行シナリオについては、EDB プロフェッショナルサービスに相談してください

4.1.3.6 次のステップ

移行が成功した後

- 進行中の管理については *operator installation guide* を確認します
- *updating HCP configuration* について知る
- 演算子を使用して *HM* バージョンのアップグレード を参照してください

4.1.4 Troubleshooting Hybrid Manager installations

Hybrid ManagerHM インストールには、インストールを検証したり、実行中の展開を監視したりするために使用できるプリフライトおよびポストフライトチェックが付属しています。これらのツールは、失敗したチェックのトラブルシューティングに役立つ詳細情報も提供します。

4.1.4.1 Understanding Preflight checks

Preflight チェックの結果を確認して、インストール関連のすべてのシークレットが有効であることを確認します。

重点ロールサイト信頼性エンジニア SRE/インフラストラクチャエンジニア

4.1.4.1.1 前提条件

- フェーズ *5* ハイブリッドマネージャーのインストール —インストールコマンドを実行しましたが、プロセスがスタックしているようです。

4.1.4.1.2 結果

- **ハイブリッドマネージャー *HM* のインストール中に、シークレット関連のブロッカーを診断および解決する機能。**

HM インストールを開始すると、オペレーターは `Preflight` カスタムリソースを自動的に作成します。このリソースは、必要なすべての Kubernetes シークレットが存在し、オペレーターがプラットフォームサービスの展開を続行する前に**正しく構造化されていることを検証します。

Preflight リソースは自分で作成しません。 `HybridControlPlane` リソースが作成された後、オペレーターは自動的に作成します。

`HybridControlPlane` マニフェストを適用すると、オペレーターは次のことを行います。

1. `HybridControlPlane` リソースを調整します。
2. デプロイメントのシークレット要件を使用して `Preflight` リソースを作成します。
3. 各シークレット要件を検証します。シークレットが存在することを確認し、シークレットのコンテンツに対して検証ルールを評価します。
4. プリフライトリソースステータスの結果を報告します。

すべての要件が合格した場合、オペレーターはプラットフォームサービスの展開を続行します。要件が失敗すると、**展開がブロックされ**、オペレーターは問題が解決されるまで 30 秒ごとに再試行します。

4.1.4.1.3 プリフライトステータスの確認

installation Helm command を実行した後にインストールがスタックしているように見える場合は、プリフライトステータスを検査します。

```
kubectl get preflight -o yaml
```

`status.secretRequirementsStatus` セクションには、チェックされた各シークレットの結果が表示されます。

```
status:
  observedGeneration: 1
  secretRequirementsStatus:
    - secret:
        name: hm-portal-bootstrap
        namespace: default
      state: Valid
    - secret:
        name: edb-object-storage
        namespace: default
```

(次のページに続く)

(前のページからの続き)

```
    state: Invalid
    messages:
      - "failed to pass the rule: aws-configurations"
  - secret:
      name: langflow-secret
      namespace: default
    state: SecretNotFound
```

4.1.4.1.4 状態を理解する

State	Meaning	Action required
Valid	Secret exists and passes all validation rules.	None.
Invalid	Secret exists but one or more validation rules failed.	Review the *messages* field to identify which rules failed, then fix the secret content.
SecretNotFound	The referenced secret does not exist.	Create the secret in the correct namespace with the required keys.

!!!警告

すべての シークレット要件が `Valid` 状態に到達するまで、展開はブロックされたままです。オペレーターは 30 秒ごとに自動的に再試行します。すべての要件が合格すると、手動介入なしで展開が続行されます。

各シークレット要件は、 **インストールシナリオ** `core` 、`migration` 、`ai` 、または `analytics` に関連付けられています。オペレーターは、`HybridControlPlane` の `spec.scenarios` で構成されたシナリオを使用して、チェックする要件をフィルタリングします。

- 構成が特定のシナリオ `core, ai` などを定義している場合、これらのシナリオに一致する要件のみが検証されます。他のシナリオ `migration` などの要件はスキップされます。
- 構成が `spec.scenarios` を定義していない場合、シナリオに関係なく、すべてのシークレット要件が検証されます。

これは、実際に展開するシナリオのシークレットを提供するだけでよいことを意味します。たとえば、`analytics` シナリオを使用しない場合、Lakekeeper の交絡キーシークレットを提供する必要はありません。

4.1.4.1.5 プリフライト障害のトラブルシューティング

インストールが停止しているようです

Helm コマンドは完了したが、ポッドが起動していない場合、プリフライトチェックが展開をブロックしている可能性があります。最初にプリフライトステータスを確認します。

```
kubectl get preflight -o yaml
```

`Valid` ではないエントリがないか、 `status.secretRequirementsStatus` を確認します。

シークレットが見つかりません

要件に SecretNotFound が示されている場合、シークレットが存在することを確認します。

```
kubectl get secret <secret-name> -n <namespace>
```

シークレットが存在しない場合は、*Preparing your environment* のドキュメントに従って作成します。名前と名前空間がプリフライトで想定されているものと正確に一致することを確認します。

検証ルールの失敗

要件が Invalid を示している場合、messages フィールドは、どのルールが失敗したかを示します。例

```
failed to pass the rule: aws-configurations
```

これは、シークレットが存在するが、予想される構造を満たさないことを意味します。シークレットを検査します。

```
kubectl get secret <secret-name> -n <namespace> -o yaml
```

シークレットのデータキーと値をルールメッセージと比較します。一般的な原因には次のようなものがあります。

- **欠落キー** —シークレットには必須データフィールドたとえば、AWS オブジェクトストレージの aws_region または bucket_name が欠落しています。
- **アノテーションの欠落** —一部のシークレットでは、シークレットのメタデータに特定のレプリケーターアノテーションを設定する必要があります。
- **間違ったシークレットタイプ**—たとえば、TLS シークレットには kubernetes.io/tls タイプが必要です。
- **空の値** — 必須フィールドは存在しますが、データが含まれません。

シークレットを修正した後、オペレーターは 30 秒以内に自動的に再評価します。

要件がチェックされていません

要件が検証されると予想されるが、ステータスに表示されない場合は、シナリオ構成を確認します。

1. HM インストールにどのシナリオが定義されているかを確認します。

```
kubectl get hybridcontrolplane <name> -o jsonpath={.spec.scenarios}
```

2. 確立されたシナリオが、*Planning architecture* フェーズ中に CTO/アーキテクト/リードエンジニアが確立したシナリオと一致することを確認します。コマンドの出力が予想されるシナリオと一致しない場合、それはインストールが計画通りに実行されず、インストール中に 1 つ以上のシナリオがスキップされたことを意味します。

4.1.4.2 Monitoring with Postflight checks

Postflight チェックの結果を確認して、HM 展開の準備ができていることを監視します。

重点ロール インフラストラクチャエンジニア/プラットフォーム管理者

4.1.4.2.1 前提条件

- フェーズ *5* ハイブリッドマネージャーのインストール —HM のインストールが完了および確認できました。展開が実行されており、その状態を確認するか、失敗したチェックのトラブルシューティングを行う必要があります。

4.1.4.2.2 結果

** 展開されたハイブリッドマネージャー HM プラットフォームが正常で、2 日目構成の準備ができていることの確認。*

Postflight カスタムリソースは、展開された HM スタックの自動ヘルスモニタリングを提供します。一連のビルトインチェックを定期的に実行し、全体的な状態を報告します。

> **注釈**
>
> *Understanding Preflight checks* とは異なり、 postflight はモニタリング機能を提供します。操作をブロックせず、自動修復をトリガーしません。トラブルシューティングの可視性を提供します。 !!!

4.1.4.2.3 ポストフライトチェックの機能

オペレーターは、 `Postflight` リソースを使用して、構成可能な間隔で 5 つのビルトインヘルスチェックを実行します。チェックは、`HybridControlPlane` が `deployed` フェーズに到達した後にのみ開始されます。各チェックは、ランニングプラットフォームのさまざまな側面を検証します。

Check	What it verifies
Pods	All HM pods are running and ready.
Databases	All internal EDB Postgres for Kubernetes clusters are ready and their latest backups completed.
Velero backups	The latest Velero backup completed successfully.
Nodes	All Kubernetes cluster nodes are ready.
Certificates	All cert-manager certificates are valid and ready.

all チェックがパスした場合、全体的なフェーズは `Healthy` です。**任意の** チェックが失敗した場合、フェーズは `Unhealthy` です。

4.1.4.2.4 ポストフライトステータスの確認

概要

インストールが正常かどうかを確認します。

```
kubectl get postflight
```

出力は次のようになります。

```
NAME                  PHASE     LAST CHECK
edbpgai-postflight    Healthy   2025-01-15T10:30:00Z
```

詳細な結果

フェーズが Unhealthy の場合、より詳細な結果を取得します。

```
kubectl get postflight <name> -o yaml
```

status.checkResults セクションには、個々の結果が表示されます。

```
status:
  phase: Unhealthy
  lastTransitionTime: "2025-01-15T10:30:00Z"
  checkResults:
    - name: PodsChecker
      description: "Check whether all pods belonging to the Hybrid Manager are ready."
      state: Passed
    - name: DBsChecker
      description: "Check whether all internal databases belonging to the Hybrid␣
↪Manager are ready."
      state: Passed
    - name: VeleroChecker
      description: "Check whether the latest Velero Backup has been completed."
      state: Failed
      messages:
        - "the latest velero backup is in a unhealthy phase: PartiallyFailed"
    - name: NodesChecker
      description: "Check whether all nodes are ready."
      state: Passed
    - name: CertificatesChecker
      description: "Check whether all certificates belonging to the Hybrid Manager␣
↪are valid."
      state: Passed
```

4.1.4.2.5 異常なチェックのトラブルシューティング

ポストフライトフェーズが Unhealthy の場合、 status.checkResults セクションを検査して、失敗したチェックとそれに関連するメッセージを特定します。

ポッドチェックが失敗する

1 つ以上の HM ポッドが準備完了状態にありません。障害が発生したポッドを調査します。

```
kubectl get pods -A | grep -v Running
kubectl describe pod <pod-name> -n <namespace>
kubectl logs <pod-name> -n <namespace>
```

一般的な原因には、イメージのプルの失敗、リソース制限、コンテナのクラッシュループが含まれます。

データベースチェックが失敗する

Kubernetes データベースクラスターの EDB Postgres の準備ができていないか、最新のバックアップに失敗しました。クラスターとバックアップのステータスを確認します。

```
kubectl get clusters.postgresql.k8s.enterprisedb.io -A
kubectl get backups.postgresql.k8s.enterprisedb.io -A --sort-by=.metadata.
↪creationTimestamp
```

一般的な原因には、ストレージの問題、接続障害、WAL アーカイブの問題が含まれます。

Velero バックアップチェックが失敗します

最新の Velero バックアップは、障害または部分的に障害が発生した状態です。

```
kubectl get backups.velero.io -n velero --sort-by=.metadata.creationTimestamp
kubectl describe backup <backup-name> -n velero
```

ノードチェックが失敗する

1 つ以上のクラスターノードの準備ができていません。これは、HM だけでなくクラスター全体に影響します。

```
kubectl get nodes
kubectl describe node <node-name>
```

一般的な原因には、ディスク圧力、メモリ圧力、ネットワーク到達不能状態、および kubelet の問題が含まれます。

証明書チェックが失敗します

cert-manager 証明書は発行されていないか、有効期限が切れています。

```
kubectl get certificates -A
kubectl describe certificate <cert-name> -n <namespace>
kubectl get certificaterequests -A
```

一般的な原因には、発行者の構成の問題、レート制限 ACME / Let’s Encrypt の場合、および DNS 検証の失敗が含まれます。

4.1.4.2.6 チェック間隔の設定

ポストフライトチェックが実行される間隔を変更することもできます。

spec.interval フィールドは、チェックを実行する頻度を制御します。フェーズが Healthy の場合、オペレーターはこの間隔を尊重し、実行が成功するたびに残りの期間を再キューイングします。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: Postflight
metadata:
  name: edbpgai-postflight
spec:
  hm:
    name: edbpgai
  interval: 30m
```

注釈

フェーズが `Unhealthy` に移行すると、ステータスがすぐに更新されます。間隔は、システムが正常な場合の再チェックのリズムにのみ影響します。 !!!

第5章　Using Hybrid Manager

このセクションでは、日常のクラスター管理から高可用性、ディザスターリカバリーまで、さまざまなユースケースで EDB Postgres AI ハイブリッドマネージャー HM を操作する方法を理解するのに役立ちます。

トピックは対象者ごとに構成されており、一般的なプラットフォームユーザーアプリケーションチーム、データベースエンジニア、プラットフォーム所有者から始まり、クラスターのセキュリティ、オブザーバビリティ、および統合を担当する Kubernetes 管理者まで続きます。

5.1 どこから始めるべきか

基本的なセクションから始めます。

- 用語 は、主要な HM 概念を紹介します。
- *Hybrid Manager overview* は、Postgres、Lakehouse、および AI ワークロードに重要なコンテキストを提供します。

そこから、自分の役割と責任に基づいてトピックを探索します。

5.2 ユーザー向け

これらのセクションでは、ハイブリッドマネージャーでの日常的なタスクについて説明します。

- *Migration Portal* プロジェクトの所有者 —アラートや使用状況レポートを含むプロジェクトレベルの設定を作成、管理、および構成します。
- Cluster Management —バックアップ、クエリ診断、テンプレートを含む Postgres クラスターを作成、監視、および管理します。
- Backup and Restore —バックアップスケジュール、復元ポイント、およびディザスターリカバリー構成を管理します。
- :ref:``edbctl` を使用して移行ポータルシークレットを作成する <*edbctl* を使用して移行ポータルシークレットを作成する>` —CLI を使用して、展開、構成、およびデータ操作を自動化します。
- ステップ *5API* アクセスを構成する — HM API を使用してプラットフォーム機能を自動化します。
- Postgres database upgrades - HM 管理の Postgres データベースを次のマイナーまたはメジャーバージョンにアップグレードします。
- *Simple Migration Agent* —Oracle、RDS、またはその他の Postgres インストールからスキーマとデータの移行を実行します。

- Image Management —コンテナイメージ、リポジトリ、および AI ナレッジベースアセットを管理します。
- Enable Migration Copilot —Copilot を使用したガイド付き移行ワークフローのロックを解除します。
- *What the Chat Agent can do* —平易な言語でハイブリッドマネージャーを操作します。エステートの参照、クラスターのプロビジョニング、パフォーマンスのトラブルシューティング、およびドキュメントの参照。
- *Governance in Hybrid Manager* —EDB エージェントガバナンスを HM と統合して、Postgres クラスターで動作している AI エージェントを監査する方法。

5.3 Kubernetes 管理者の場合

ハイブリッドマネージャーの下のインフラストラクチャを管理している場合

- *Managing users in Hybrid Manager* —ネイティブユーザーを構成します。
- IdP management —ID プロバイダーを構成し、IdP コネクタ暗号化キーをローテーションします。
- **モニタリングの実行**—エージェントをセットアップし、ビルトインダッシュボードを探索し、Prometheus と Grafana で可観測性を拡張します。
- **マルチ *DC* トポロジの構成** - 複数のデータセンター、プライマリおよびセカンダリの場所をセットアップします。
- Hybrid Manager upgrades - HM プラットフォームを次のマイナー/メジャー/イノベーションリリースバージョンにアップグレードします。
- HA/DR —実稼働クラスターの高可用性とディザスターリカバリーを構成します。
- *Troubleshooting Langflow in Hybrid Manager* —すべてのプラットフォーム領域にわたって問題を調査および解決します。

5.3.1 Hybrid Manager Terminology

これらの用語を確認して、ハイブリッドマネージャー HM 機能を理解および活用します。

5.3.1.1 ハイブリッドマネージャー

Hybrid Manager は、Postgres および Lakehouse データベースとソブリン AI システムの展開、管理、監視、自動化のための EDB のソリューションに与えられた名前です。

これは、最新のプライベートクラウド、パブリッククラウド、マルチクラウド、またはオンプレミスのインフラストラクチャを最大限に活用することにより、展開されたデータベースクラスターのライフサイクルを簡素化する Kubernetes ベースのソフトウェアスタックであり、開発者がプライベートを介してアプリケーションのデータベースを自己管理できますまたは Sovereign Database-as-a-ServiceDBaaS モデル。

5.3.1.1.1 プライベート Database-as-a-Service DBaaS

ハイブリッドマネージャーの重要な機能は、Kubernetes でデータベースをプロビジョニングおよび管理できる機能です。この機能は、Private Database-as-a-Service DBaaS と呼ばれます。

この機能により、ハイブリッドマネージャーは、インフラストラクチャ内の自己管理の内部データベースサービスになり、プライベート環境の制御とセキュリティを備えたパブリッククラウド DBaaS の機敏性を提供します。

このアプローチにより、開発者はセルフサービスのプロビジョニング、可観測性、およびトラブルシューティングを実行できます。

5.3.1.1.2 エンジニアードシステム

Engineered System は、ボックス内のハイブリッドマネージャーであり、ラックにスロットしてコンテナ化されたアプライアンスとして展開できるすぐに使える物理サーバーです。この展開方法は、ハイブリッドまたはマルチクラウドの世界でデータベース操作を合理化しようとしている組織に、柔軟でポータブルなソリューションを提供します。

5.3.1.2 PG AI Console

PG AI コンソールは、ハイブリッドマネージャーの集中的な対話ポイントをユーザーに提供する Web ベースのグラフィカルインターフェイスです。このダッシュボードから、データベースの展開と管理、外部データベースの接続と監視、データの移行の実行、ユーザー権限の管理、AI ワークロードの展開などを行うことができます。

PG AI Console の機能の概要については、 *About the PG AI Console* を参照してください。

5.3.1.2.1 プロジェクト

ハイブリッドマネージャーでは、 **プロジェクト**は、リソースの管理と作成のための分離を提供する基本的な組織単位として機能します。これらの個々のプロジェクト内でクラスターを作成および管理できます。プロジェクトを使用して、チームごとにエンジニアリング、オペレーション、データサイエンス、環境ごとに開発、ステージング、プロダクション、製品ごと、顧客ごとなどにリソースを管理します。

詳細については、 *Migration Portal* **プロジェクトの所有者** を参照してください。

5.3.1.2.2 エステート

ハイブリッドマネージャーでは、 **エステート**は、アクセスするすべてのデータベースリソースの基本的な概要として機能します。ハイブリッドマネージャーでデータベースまたは分析クラスターを作成した場合でも、モニタリングのために外部データベースに接続した場合でも、それはエステートの一部になります。

詳細については、 エステートページにアクセスするときの HTTP 431 "Request Header Fields Too Large"エラー を参照してください。

5.3.1.2.3 Launchpad

Launchpad は、ハイブリッドマネージャーの機能を強化するために使用できる多数のサードパーティサービスへのアクセスを提供します。

詳細については、 *Launchpad* を参照してください。

5.3.1.3 クラスター明確化

コンピューティングにおける **クラスター**とは、共通の目標を達成するために統合システムとして連携する相互接続されたコンポーネントのグループを指します。多くのテクノロジーがこのアプローチを採用するため、**クラスター**という用語は、複数の個々のコンポーネントが統合された方法で管理されるシステムを指すように広く普及しています。

5.3.1.3.1 Kubernetes クラスター

ハイブリッドマネージャーのスコープ内で、Kubernetes クラスターは、ハイブリッドマネージャーソフトウェア自分自身が自己完結型インスタンスとして展開される基盤インフラストラクチャを表します。Hybrid Manager は、この基になる Kubernetes クラスターの機能を活用して、Kubernetes で Postgres をオーケストレーションするためのツールとして CloudNativePG を使用して、Postgres データベースの展開を管理します。

**Postgres* cluster* または **Database* cluster* と混同しないでください。

5.3.1.3.2 Kubernetes ストレッチクラスター

これは、ハイブリッドマネージャーの展開にも使用できる Kubernetes クラスターのサブタイプです。

!!!note

この機能は、後のバージョンで追加される予定です。

通常、Kubernetes クラスターのすべてのノードは、単一のアベイラビリティーゾーンにあります。高可用性 HA、高度な高可用性 AHA、または分散高可用性 DHA が有効になっているハイブリッドマネージャー展開の場合、プラットフォームは、スタンバイノードによるシームレスなテイクオーバーを開始することにより、プライマリノードの障害を自動的に処理します。

ただし、このビルトインフェイルオーバーメカニズムは、アベイラビリティーゾーン全体が停止した場合には不十分です。ここで、**Kubernetes ストレッチクラスター**が重要になります。これらは、Kubernetes クラスターを複数のアベイラビリティーゾーンまたは地理的に離れた場所に拡張または「拡張」し、1 つのゾーンに障害が発生しても、他の場所の正常なノードがアプリケーションのワークロードを継続して提供できます。この設定では、AZ に依存しない共有ストレージを使用して、AZ 障害が発生している間のデータアクセスを維持する必要があります。

5.3.1.3.3 Postgres クラスター

Postgres では、クラスターは単一の *instance of Postgres* であり、Postgres サーバーとしても知られています。Postgres インスタンスのこの単一のオカレンスには、1 つ以上の異なるデータベースを含めることができます。たとえば、データベース *HR* 、 *FINANCE* 、および *LEGAL* を使用する MYCOMPANY のサーバー。

このコンテキストでの「クラスター」の使用があいまいになる可能性があるため、この用語はハイブリッドマネージャーのドキュメントでは使用されません。ここでは、Postgres インスタンスと呼びます。

Kubernetes cluster* または *Database* cluster と混同しないでください。

5.3.1.3.4 データベースクラスター

ハイブリッドマネージャーのドキュメントでは、**データベースクラスター**は、レプリケーション機能を使用してデータベースの復元力をサポートし、パフォーマンスを向上させる Postgres の複数のインスタンスを指します。このようなクラスターベースのシステムは、現代のエンタープライズ Postgres 操作の重要な機能であるため、ハイブリッドマネージャーはデータベースクラスターを中心に構築されています。このため、シングルノードの Postgres インスタンスでもクラスターとして表されます。

たとえば、MYCOMPANY の postgres サーバーにデータベース *HR* 、 *FINANCE* 、および *LEGAL* が含まれ、同じサーバーが冗長性とフェイルオーバーのためにレプリケートされる場合、システム全体をデータベースクラスターと呼びます。2 つのインスタンスを使用するシナリオでは、 *HR* 、 *FINANCE* 、および *LEGAL* を使用したデータベースサーバ A と、 *HR* 、 *FINANCE* 、および *LEGAL* を使用したレプリカサーバー B が存在します。サーバー A と B は同じデータを持っています。このレプリケーションセットアップの各サーバーは、独自のデータノードに存在します。

データベースクラスターは、ハイブリッドマネージャー内で作成するか、外部で作成できますが、可観測性のためにハイブリッドマネージャーに接続します。ハイブリッドマネージャーを作成または接続できるすべてのクラスタータイプの概要については、*Database cluster types in the Hybrid Manager* を参照してください。

Kubernetes cluster* または *Postgres* cluster と混同しないでください。

5.3.1.4 ハイブリッドマネージャーのデータベースクラスタータイプ

Hybrid Manager インスタンスは、複数のデータベースクラスターカテゴリをホスト、管理、および監視できる一元化されたガラス画面です。一部のデータベースクラスターはハイブリッドマネージャー内で作成され、完全に維持されますが、他のデータベースクラスターは外部で作成されますが、ハイブリッドマネージャーに接続され、監視と移行機能を提供します。

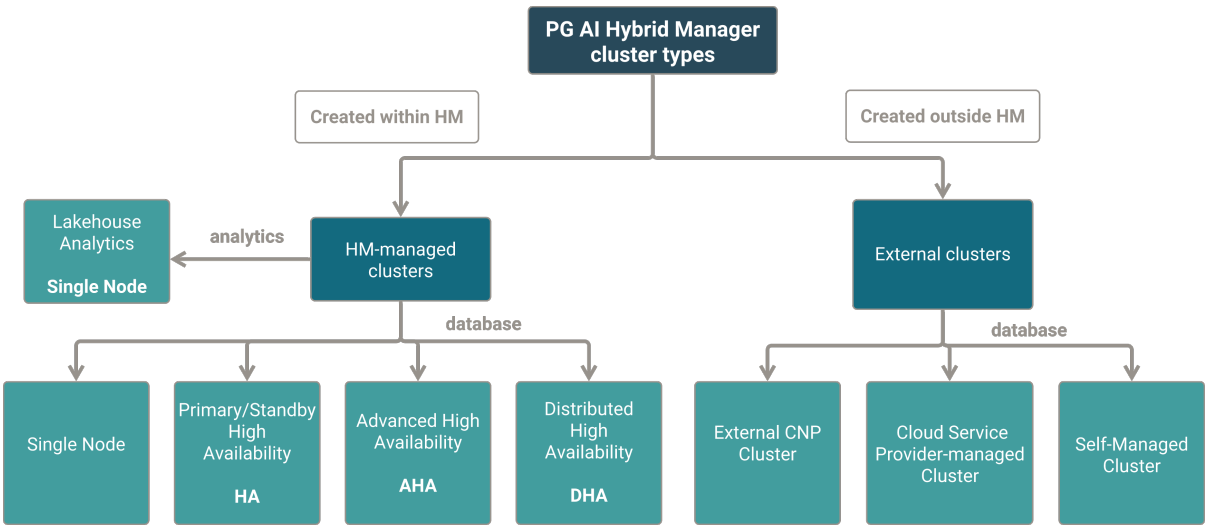


図 1: Cluster Types in Hybrid Manager

現在、次のことができます。

- *HM 管理対象クラスターと外部 CNPG クラスター* の作成

または、

- 外部クラスターの要件 を接続します。

5.3.1.4.1 HM 管理対象クラスター

HM 管理対象クラスターは、ハイブリッドマネージャーで作成したデータベースサーバーです。これらは、プロビジョニング機能を使用して、ハイブリッドマネージャーと同じ Kubernetes クラスターに展開されます。**Estate** ページでは、HM 管理クラスターの **Management** 列の下に **Hybrid Manager** ラベルがあります。

例 展開された PostgreSQL、 EDB Postgres Advanced Server、または EDB Postgres Extended Server を使用した単一ノード、プライマリ/スタンバイの高可用性 HA、Advanced High AvailabilityAHA、または分散高可用性 PGD データベースインスタンスハイブリッドマネージャーを使用して。

EDB は、次のデータベーストポロジオプションを提供します。

単一ノードこのオプションは、単一の Postgres インスタンスを作成します。詳細については、*Single-node cluster* を参照してください。

プライマリ/スタンバイの高可用性 HA とも呼ばれるこのトポロジオプションは、Postgres インスタンスをホストし、少なくとも 1 つの追加ノードでそれをレプリケートします。詳細については、*Primary/standby high availability cluster* を参照してください。

高度な高可用性 AHA とも呼ばれるこのトポロジは、単一リージョンのアベイラビリティーゾーンを超えて Postgres インスタンスとそのレプリカの少なくとも 2 つのグループをホストします。詳細については、*Advanced high availability cluster* を参照してください。

分散高可用性 DHA とも呼ばれるこのトポロジは、複数のリージョンのアベイラビリティーゾーンにまたがって Postgres インスタンスとそのレプリカの少なくとも 2 つのグループをホストします。詳細は、*Distributed high availability cluster* を参照してください。

5.3.1.4.2 外部クラスター

外部でプロビジョニングされたクラスターを接続できます。

5.3.1.4.3 自己管理クラスター

自己管理クラスターは、外部 Postgres または Oracle クラスター Hybrid Manager の外部で作成されます。これらは、ハイブリッドマネージャー以外のプロビジョニング機能などを使用して展開され、多くの場合、ハイブリッドマネージャーとは別のマシンに展開されます。

Estate ページでは、自己管理クラスターの **Management** 列の下に **Self-Managed** ラベルが表示されます。

Simple Migration Agent でのオブザーバビリティのために自己管理クラスターをハイブリッドマネージャーに接続し、*Data Migration Service* **読み取り専用** で移行を実行できます。

例 物理サーバー、仮想マシンなどに自分自身で展開された PostgreSQL、EDB Postgres Advanced Server、EDB Postgres Extended Server、または Oracle データベースインスタンス。仮想マシンの場合、クラウドインフラストラクチャで実行できますが、データベースクラスターのプロビジョニングはクラウドサービスプロバイダーではなくお客様が管理します。

!!!note

ハイブリッドマネージャーに接続する自己管理 Oracle データベースは、**Estate** > **Postgres** の下に表示されません。情報を表示するには、**Estate** > **Migrations** に切り替えます。

ハイブリッドマネージャーに接続する自己管理の Postgres クラスターは、**Estate** > **Postgres** と **Estate** > **Migrations** の両方に表示されます。

5.3.1.4.4 クラウドサービスプロバイダー管理のクラスター

クラウドサービスプロバイダー管理クラスターまたは CSP 管理クラスターは、Postgres インスタンスの 1 つ以上のオカレンスをホストする外部クラスターハイブリッドマネージャーの外部で作成されます。これらはクラウドサービスプロバイダー CSP プロビジョニング機能を使用して展開され、主要な操作タスクは CSP によって処理されますバックアップ、アップグレード、フェールオーバーなど。ただし、アーキテクチャ、開発者、およびアプリケーション固有のアクティビティは自分自身が処理します。

Estate ページでは、クラウドホスト型データベースクラスターの **Management** 列の下に **Cloud Hosted Databases** ラベルがあります。

connect CSP-managed clusters to Hybrid Manager for observability 、 *Data Migration Service* **読み取り専用** を使用して移行を実行できます。

例 AWS RDS の Postgres。

5.3.1.4.5 外部 CNPG クラスター

外部 CNPG データベースクラスターは、CloudNativePG を使用してハイブリッドマネージャーの外部の Kubernetes 環境で作成された外部データベースクラスターです。これらは、オープンソースの CloudNativePG、EDB CloudNativePG Cluster または EDB CloudNativePG Global Cluster のプロビジョニング機能を使用して展開されます。

エステート ページでは、外部 CNPG クラスターの **管理** 列の下に**外部 CNP クラスター** ラベルがあります。

例 CloudNativePG、EDB CloudNativePG クラスターまたは EDB CloudNativePG グローバルクラスターで展開された PostgreSQL、 EDB Postgres Advanced Server、または EDB Postgres Extended Server データベースインスタンス。

5.3.1.5 その他の用語

5.3.1.5.1 データノード

「データノード」という用語は、ハイブリッドマネージャーによって管理される Postgres データベースインスタンスをホストするように特に指定された Kubernetes ワーカーノードを参照するために使用されます。

5.3.1.5.2 Postgres インスタンス

複数のデータベースをホストできる Postgres サーバー。これは、そのインスタンスを実行するために必要なすべてのデータとファイルを含む特定のデータディレクトリで起動します。高可用性展開では、少なくとも 1 つの追加インスタンスが別のアベイラビリティーゾーンに複製されます。

5.3.2 Hybrid Manager console

5.3.2.1 Hybrid Manager コンソールとは何ですか?

Hybrid ManagerHM コンソールは、Postgres クラスターをプロビジョニングおよび管理するためのユーザーインターフェイスです。これは、データベースのライフサイクルを表示、監視、および管理するための集中的な場所です。

このコンソールでは、リソースを **プロジェクト**に整理します。プロジェクトは、組織に関連するリソースを関連付けるため、管理とアクセスが簡素化されます。

5.3.2.2 HM コンソールでは何ができますか?

コンソールを使用して、次のことができます。

- Postgres クラスターの作成、表示、および管理
- プロジェクトの作成、表示、および管理
- Migration Portal プロジェクトを作成し、移行を実行する
- 監視データベースの移行
- カタログの表示と管理
- モデルクラスターのステータスの表示
- アセットライブラリを管理する
- HM 管理対象および外部クラスターの監視
- Apache Airflow、Apache Superset、Metabase、pgAdmin 4、pgBadger などのオープンソースアプリを HM 内で直接展開および管理します。

5.3.2.3 HM コンソールの構成

コンソールは、HM のすべてのページに表示されるページ名を選択してアクセスする 3 つのメインページを中心に構成されています。

- ***Projects***、プロジェクトの作成と管理、クラスターの作成と管理、およびクラスターと各プロジェクトを管理するタスクを実行します。
- ***Estate***、リソースをモニタします。クラスターを作成するための開始点として使用することもできます。
- ***Asset Library***、コンテナイメージとナレッジベースを表示し、リポジトリを管理できます。

5.3.2.4 サードパーティサービスのローンチ

HM *Launchpad* を使用して、HM を終了せずに関連するサードパーティサービスを起動できます。

5.3.2.4.1 Exploring the Hybrid Manager console

コンソールへのアクセス

アカウントを取得したら、Hybrid ManagerHM コンソールにアクセスし、管理者が提供する `https://<your-domain>/` フォームの URL を使用します。管理者から提供された資格情報を使用してポータルにログインします。

デフォルトでは、コンソールが開き、アカウントの概要ページと、 EDB Postgres AI HM サービスの **単一ウィンドウ**の概要が表示されます。ポータルのどこからでもこのビューに戻るには、EDB ロゴを選択します。

このビューでは、 *Migration Portal* **プロジェクトの所有者** および エステートページにアクセスするときの HTTP 431 "Request Header Fields Too Large"エラー の概要を表示できます。

Projects 、 **Estate** 、および **Asset ライブラリー**タブを選択して、さまざまなリソースを監視および作成できるページに移動します。

プロジェクトでリソースを整理する

リソースを作成および管理するには、リソースを保存するプロジェクトが必要です。プロジェクトは、組織に関連するリソースを関連付けるため、管理とアクセスが簡素化されます。

詳細については、 *Migration Portal* **プロジェクトの所有者** を参照してください。

エステート内のリソースのモニタリング

HM コンソールでは、 *estate* は、アクセスできるすべてのリソースを指します。物理的資産が不動産の一部と考えられるのと同様に、データベースのライフサイクルに関連するリソースは HM 資産の一部です。HM での不動産の管理とは、リソースの状態を監視するか、安全に保つか、必要に応じて新しいものを作成するかどうかを管理することを意味します。

リソースをエステートとして表示することは、すべてのリソースの単一の統合ビューを提供するため、プロジェクトで表示することとは異なります。

リソースはさまざまなカテゴリにグループ化されています。

- ***Postgres clusters***
- ***Migrations***
- ***Catalogs***
- **OSS Library**

エステートの機能を探る

エステートの **Postgres** タブでは、次のことができます。

- HM 管理か外部かにかかわらず、Postgres クラスターを監視します。
- HM で作成した Postgres クラスターにアクセスします。テーブル内のデータベースクラスターを選択して、詳細ビューを開きます。
- 移行の宛先クラスターを作成します。

エステートの **移行** タブでは、次のことができます。

- HM に接続した *assessment of the databases* を表示して、移行の準備をします。
- 特定のデータベースを選択し、 *an assessment of its schemas* を表示します。
- 特定のデータベースと *perform a schema migration with Migration Portal integration* を選択します。
- 移行の宛先クラスターを作成し、データベースの移行を実行します。

不動産の **カタログ** タブでは、次のことができます。

- HM に接続されたまたは HM で作成された *Apache Iceberg ®* テーブルをすべて表示します。

エステートの **Apps** タブでは、次のことができます。

- OSS Library apps にアクセスして、顧客データを管理およびモニタします。

クイックアクションメニュー

エステートページのどこからでも、 **クイックアクション**メニューの一般的なアクションにアクセスできます。アクションには次のものが含まれます。

- **EDB Postgres AI クラスター** を作成します** 既存のプロジェクトで Create an EDB Postgres AI cluster 。
- **自己管理クラスターの追加** —新しい自己管理クラスターを既存のプロジェクトに追加します。

Launchpad

コンソールでは、Launchpad を使用して、HM の機能を強化するために使用できるサードパーティサービスにアクセスできます。たとえば、Launchpad から *Customizing Migration Portal secrets for secure internal communication* にアクセスできます。これらのサービスは、展開されているプラットフォームに関係なく、HM のすべてのユーザーが利用できます。

Launchpad にアクセスするには、ユーザーメニューから **Launchpad** を選択します。

利用可能なすべてのサービスが Launchpad に表示され、各サービスの簡単な説明が表示されます。検索フィルタは、探しているサービスを見つけるのに役立ちます。サービスには次のものが含まれます。

- *Agent Factory* **メトリックの** *Grafana* **ダッシュボードの設定**
- *Langflow FAQ*
- プロメテウス
- サノス - クエリ
- Thanos - ストアゲートウェイ
- Thanos - Compactor
- サノス - ルール
- アラートマネージャー
- EDB Migration Portal

ユーザーメニュー

ユーザーメニューから、一部のポータル関連のアクションと一部のアカウント関連のオプションに集中的にアクセスできます。

ポータル関連のオプション

- **Launchpad** —*Launchpad* を開き、HM にインストールされているサードパーティサービスにアクセスします。
- **設定** — 組織名 **プロフィール** を介して *set your monitoring notification preferences* を設定できます。
- **Projects** — Projects overview を開きます。
- **Tags** — **Tags** ページを開き、タグ名を追加し、ディスプレイ内のタグ/バッジの色を選択できます。
- **User Management** —ユーザーを追加し、ロールを割り当てできる *User Management* ページを開きます。
- **使用レポート** —ディスク IO、vCPU、メモリ、ディスクおよびアップタイムを含む *information about each cluster's use of resources* を表示します。
- **アクティビティログ** —HM で管理されているリソースの作成、削除、更新など、実行されたアクションのリストを表示します。

アカウントオプション

- **マイアカウント** —プロフィールの表示、アクセスキーの表示と管理、およびアカウント、組織、およびプロジェクトの通知の管理ができます。
- **ログアウト**

技術的な詳細へのアクセス

HM コンソールのページのフッターにある **About** を選択して、現在のセッションと環境に関する技術的な詳細にアクセスできます。“Troubleshooting Information”ビューは、EDB サポートが問題を診断するために必要な場合が多い重要な詳細を提供します。

バージョンとビルド —特定のハイブリッドマネージャー HM ソフトウェアリリースと内部ビルドタイムスタンプを識別します。

シナリオ — 環境に現在展開されているアクティブな機能 AI、分析、または移行などをリストします。

UI Version —コンソールインターフェイスの現在のバージョンを指定します。

組織 ID とセッション ID —組織と現在のブラウザセッションの一意の識別子が表示されます。

クライアントの詳細 - ブラウザーの種類、オペレーティングシステム、および言語設定を表示します。

5.3.2.4.2 Postgres clusters

アカウント内のすべてのプロジェクトのすべてのクラスターに関する情報を表示するには、**Estate** ページから **Postgres** タブを選択します。

概要統計の表示

Postgres タブの上部に、Postgres クラスターの次の側面を要約したカードを表示できます。

- **アクティブなアラート** —エステートで監視されているすべての Postgres クラスターにおける、高、中、および低の重大度のアラートの集計数。
- **上位 CPU 使用率** —エステートで監視されているすべての Postgres クラスターの中で、最も多くの CPU 使用率を示した上位 9 位のクラスター。カードの下部にあるコントロールを使用して、クラスターをページ移動できます。
- **リソース使用量** —エステートで監視されているすべての Postgres クラスターの集計された CPU、メモリ、およびストレージの使用量。

クラスターテーブル

クラスターテーブルは、プロジェクト全体で使用可能な Postgres クラスターのリストであり、パフォーマンス、状態、ステータス、およびその他のメトリックを監視するのに役立ちます。ビューを変更して表示する列の数を変更したり、フィルターを適用して目的のクラスターのみを表示したりできます。

表示する列の選択

デフォルトでは、テーブルには次の列が表示されます。

- Project Name — クラスターが属するプロジェクト。
- Cluster — クラスター名。
- Management — クラスターが管理される方法アプライアンスまたは自己管理。
- Cluster Type — クラスターのタイプ PGD、シングルノード、または HA。
- Status — クラスターの現在のステータス OK、Error、 注意、または Working 。
- Engine —このクラスターが実行している現在の Postgres ディストリビューションタイプ Postgres、EDB Postgres Advanced Server、または EDB Extended Server。
- Version —クラスターが実行している Postgres ディストリビューションのメジャーバージョン 13、14、15、16、または 17。
- Alerts — クラスターの高、中、および低の重大度のアラートの数。
- Health Scores —データベースインスタンスのインデックス、統計、構成、およびセキュリティの状態のスコアカード。
- TPS —クラスターによって行われた 1 秒ごとのトランザクションと、最近の傾向を要約した折れ線グラフ。
- CPU —クラスターによって現在使用されている CPU の割合と、最近の傾向を要約した折れ線グラフ。
- Memory —クラスターによって現在使用されているメモリの割合と、最近の傾向を要約した折れ線グラフ。
- Disk —クラスターが現在使用しているディスク領域の割合と、最近の傾向を要約した折れ線グラフ。
- Tags — クラスターに関連付けられたタグ。

表示する列を選択することもできます。歯車を選択して、上部に検索ボックスを備えたメニューを開きます。検索ボックスを使用して、テキスト一致を使用して表示する列をフィルタリングします。返された列名のリストから、テーブルに表示する列名を選択します。

これらは表示できる他の列です。

- # of Alerts — クラスターのアラートの合計数。
- # of Cores — クラスターが使用するコアの合計数。
- Conn.Usage — 接続の使用率。
- IO Wait % — 入力/出力タイプの待機イベントの割合。
- IOPS — クラスターの 1 秒ごとの入出力操作の数。
- カーネルバージョン — 自己管理クラスターのプラットフォームタイプ。
- Last Backup — クラスターが最後にバックアップされた時の ISO 8601 タイムスタンプ。
- OS バージョン —自己管理クラスターのオペレーティングシステムバージョン。
- Provider — CSP プロバイダーの ID。
- Region — アプライアンスクラスターのロケーション ID。
- Rep. Lag — ミリ秒単位のレプリケーションラグ。
- TDS — ギガバイト単位のデータベースの合計サイズ。

> **Tip**
>
> 列をデフォルトのセットにリセットするには、歯車メニューから**Reset Columns**を選択します。

表示するクラスターのフィルタリング

次の 2 つの手法のいずれかを使用して、表示されるクラスターのリストを絞り込んで、関心のあるクラスターを含めることができます。

- テーブルの **Search** ボックスを使用して、表示するクラスターを名前で検索します。
- **Filter** コントロールを使用して、次のプロパティに一致するクラスターを表示します。各プロパティには、プロパティに関連する特定のタイプまたは値を選択するためのサブメニューがあります。
- クラスタータイプ
- Engine
- ヘルススコアインデックス、統計、構成、セキュリティ)
- 削除されたクラスターを含む b_tran_16 ロケーション
- 管理 — ハイブリッドマネージャー - ハイブリッドマネージャー内で作成および管理されるクラスター HM).
  - クラウドホストデータベース-クラウドサービスプロバイダーによって管理される外部クラスター。

  - 外部 CNP クラスター-CloudNativePG のオープンソースまたは EDB バージョンを実行している外部クラスターたとえば、CloudNativePG グローバルクラスターの EDB Postgres AI または CloudNativePG の EDB Postgres AI Cluster).
  - 自己管理 —サーバー、VM などで実行されている外部クラスター。これは、外部 EDB Postgres 分散クラスターを除きます。
- Postgres バージョン
- プロジェクト名
- ステータス

データの古さの監視

テーブルのデータが更新される頻度を確認するには、*Metrics latency* の **Estate page > Postgres テーブル**で提供されている値を参照してください。

テーブルの下のメッセージは、テーブルが新しいデータで最後に更新された時刻を示します。

5.3.2.4.3 OSS Library

Hybrid ManagerHM は、HM コンソールから直接展開および管理できるオープンソースアプリケーションのセットを提供します。

使用可能なアプリケーション

Application	Version	License	Official documentation
Apache Airflow	v3.0.2	Apache-2.0 License	*Apache Airflow*
Apache Superset	v6.0.1rc1	Apache-2.0 License	アセットライブラリーアプリ *pgAdmin4* および *Apache Superset* は *RHOS* への展開に失敗する
Metabase	v0.58.7	GNU Affero general public license (AGPL)	*EPAS* クラスターをデータベースバックエンドとして使用する場合、メタベースの展開に失敗する
pgAdmin 4	v8.12	*Troubleshooting*	Troubleshooting
pgAdmin 4	v9.13	*Troubleshooting*	Troubleshooting
pgBadger	v13.2	PostgreSQL License	*pgBadger*
HammerDB	v5.0	GPL-3.0 license	*HammerDB*

デプロイメントとアーキテクチャ

アプリは、`.yaml` パッケージファイルを使用して、HM Kubernetes 環境内でコンテナ化されたアプリケーションとして展開されます。この展開により、データベーストラフィックが完全にプライベートネットワーク VPC / VNET およびセキュリティ境界内にとどまることが保証されます。このアーキテクチャは、ホリゾンタルスケーリングと外部 PostgreSQL アプリケーションデータベースをサポートして、高可用性を実現します。

重要

バックエンドデータベース接続を必要とするアプリを展開する場合、アプリごとに Postgres クラスター内の専用データベースを使用します。同じデータベースを共有するアプリは、初期化フェーズ中のスキーマの競合により移行が失敗するリスクがあります。 !!!

注釈

OSS ライブラリーアプリは、バックエンドアプリケーションデータベースとして Community PostgreSQL をサポートしています。 pgAdmin 4 は、 EDB Postgres Advanced Server もサポートしています。 !!!

Apache Airflow

重要

通常のアップストリームリリースでは、セキュリティ更新が配信されます。 Airflow インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

Apache Airflow ®は、 Hybrid ManagerHM 内で展開および管理されるオープンソースのワークフローオーケストレーションプラットフォームです。拡張可能な Python フレームワークは外部テクノロジーとの統合をサポートし、HM コンソールからアクセスできる Web ベースの UI は、ワークフローの実行とデバッグの可視性を提供します。Airflow は、シングルノードと分散展開の両方をサポートしています。

主な機能

- ワークフローの実行を監視、管理、およびデバッグするための Web ベースの UI。
- 有向非循環グラフ DAG を使用した Python ベースのワークフロー定義。
- cron ベースおよびイベント駆動のトリガーをサポートするビルトインスケジューラー。
- 外部システムおよびサービスと統合するための拡張可能な演算子ライブラリ。
- リポジトリから DAG ファイルを展開するための Git 同期。
- マルチプルのワーカーにわたる分散タスクの実行。

Apache Airflow のコードとしてのワークフロー

Airflow ワークフローは完全に Python で定義されています。

- パイプラインはコードで定義され、動的な DAG 生成とパラメーター化をサポートします。
- ビルトイン演算子は、カスタム統合用に拡張できます。
- Jinja テンプレートは、ランタイム変数置換でサポートされています。

サポートされているデータソース

Airflow は、Connections を介して PostgreSQL、MySQL、AWS、Snowflake を含むほぼすべてのデータソースに接続します。Airflow の Connection は、外部サービスに接続するために必要な資格情報やその他の情報を保存します。

要件

技術的要件

- Python v3.10+。
- **必須データベース** Airflow には、ワークフロー状態、タスクインスタンス、DAG 定義、および実行履歴を保存するためのデータベースが必要です。Airflow.
- を展開する前に、このデータベースを作成する必要があります。運用環境には、PostgreSQL v13 以降が推奨されます。インストールする前に、ホスト、ポート、データベース名、ユーザー名、およびパスワードを準備します。これらは、HM コンソールでクラスターの **Connect** タブから取得できます。
- Airflow は、MySQL 8.0.17+および MariaDB 10.2.2+もサポートしています。
- **オプションのストレージ** SSH キーを使用してパブリックまたはプライベート Git リポジトリから DAG を同期するための Git Sync 構成。Git 同期を使用する場合、リポジトリ URL、ブランチ名、SSH プライベートキーを準備します。

Airflow の展開

1. HM コンソールで、 **Asset ライブラリー** に移動し、 **Apps** を選択します。
2. **Apache Airflow** を選択し、 **Deploy** を選択します。
3. **Identity** の下に、展開の名前を入力し、ターゲットプロジェクトを選択します。
4. **Parameters** の下で、必須フィールドに入力します。フィールドの説明については、*HNSW* **パラメーターインデックスタイプ**= *hnsw* を参照してください。
5. **Deploy** を選択して Airflow をインストールします。

パラメーター

Parameter	Description	Required
data.metadataConnectio	外部 PostgreSQL 接続の詳細データベース名、ホスト、ポート、パスワード、ユーザー名、SSL モード `require`推奨、およびプロトコル `postgresql`	Yes
web.defaultUser	Airflow UI のデフォルトの管理ユーザー名。	Yes
web.password	Airflow UI のデフォルトの管理ユーザーパスワード。	Yes
dags.gitSync.enabled	Git リポジトリからの DAG の同期を有効にします。	No
dags.gitSync.repo	DAG を含む Git リポジトリ URL。	No

次のページに続く

表 2 – 前のページからの続き

Parameter	Description	Required
dags.gitSync.branch	同期元のブランチ。デフォルト `main`	No
dags.gitSync.subPath	DAG ファイルを含むリポジトリのサブフォルダー。デフォルト `dags`	No
dags.gitSync.sshKey	プライベート Git リポジトリへの認証のための SSH プライベートキー。	No
dags.gitSync.wait	Git 同期間隔の秒数。デフォルト `60`	No
dagProcessor.replicas	DAG プロセッサレプリカの数。 Airflow 3.0+のみ。最小 1、最大 3。	No
scheduler.replicas	スケジューラーレプリカの数。最小 1、最大 3。	No

注釈

HM に Apache Airflow を展開すると、HM コンソールの**Estate > Apps**またはプロジェクトの**Apps**ビューからアクセスできます。 !!!

展開後の構成

Airflow が展開されたら、次の手順を実行してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールから **Airflow** を開きます。
3. 展開中に `web.defaultUser` で設定した管理ユーザー名でログインします。
4. **Admin > Connections** に移動して、DAG が接続するデータベースまたはサービスの資格情報を追加します。
5. Git 同期を有効にした場合、DAG が Airflow UI に表示されることを確認します。

サポートリソース

Airflow OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- Airflow community —コミュニティからのピアツーピアサポート。
-

Airflow documentation —
  セルフサービスの展開ガイドと機能ガイド、および Airflow のインストールと構成に関する追加情報。

- Airflow GitHub repository —ソースコード、問題トラッカー、およびリリースノート。
- Best Practices — 効率的な DAG を開発および維持するためのガイダンス。
- Scheduler —スケジューラーのパフォーマンスを微調整するためのガイダンス。

-

Tutorials —
Airflow を使い始めるためのステップバイステップチュートリアル。

-

How-to Guides —一般的な Airflow タスクの実用的なガイド。

Apache Superset

重要

通常のアップストリームリリースでは、セキュリティ更新が配信されます。 Superset インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

Apache Superset は、データの視覚化、ダッシュボードの作成、およびデータベースの SQL ベースのデータ探索を提供するオープンソースのビジネスインテリジェンス Web アプリケーションです。

警告

スーパーセットは、Red Hat OpenShift RHOS をサポートしていません。 !!!

主な機能

- チャートを構築するためのノーコードインターフェイス。
- クエリー用の Web ベースの SQL エディター。
- カスタムディメンションとメトリックを定義するためのセマンティックレイヤー
- SQL データベースとデータエンジンのサポート。
- 棒グラフや地理空間ビジュアライゼーションを含むさまざまなビジュアライゼーションタイプ。
- データベースの負荷を削減するための構成可能なキャッシュレイヤー。
- 構成可能なセキュリティロールと認証オプション。
- プログラムによるカスタマイズのための API。
- クラウドネイティブアーキテクチャ。

Apache Superset コンポーネント

- **コアアプリケーション** — Python Flask バックエンド、API レイヤー、および React フロントエンドが含まれます。ユーザーがチャートまたはダッシュボードをロードすると、Superset は構成済みのデータウェアハウスを照会し、結果をビジュアライゼーションとしてレンダリングします。
- **メタデータデータベース** —チャート定義、ダッシュボード構成、ユーザー情報、およびログを保存します。PostgreSQL および MySQL は、本番使用をサポートしています。運用環境では SQLite を避けてください。一部のインストール方法では、デフォルトでそれを使用します。

- **キャッシュレイヤーオプショナル** —クエリ結果を保存することによりデータウェアハウスの負荷を削減し、ワーカープロセスのメッセージブローカーとして機能します。Redis が最も一般的な選択です。
- **ワーカーとビートオプショナル** —通常は Celery を介して、非同期タスクの実行とスケジュールを処理します。

キャッシングレイヤーとワーカーがこれらの機能にアクセスできるようにします。

- アラートとレポート
- キャッシング
- 非同期クエリ
- ダッシュボードのサムネイル

サポートされているデータソース

スーパーセットは、Python DB-API ドライバーと SQLAlchemy 方言を備えた SQL ベースのデータストアまたはデータエンジン Presto、Trino、Athena、および more からデータを照会できます。

要件

技術的要件

- **必須データベース**スーパーセットには、チャート定義、ダッシュボード構成、ユーザー情報、ログを保存するための外部データベースが必要です。Superset.
- を展開する前にこのデータベースを作成する必要があります。運用環境には PostgreSQL v13 以降をお勧めします。インストールする前に、ホスト、ポート、データベース名、ユーザー名、およびパスワードを準備します。これらは、HM コンソールでクラスターの **Connect** タブから取得できます。
- スーパーセットは MySQL v8.x+もサポートしています。
- **リソース** ワーカーポッドのデフォルトは 1Gi メモリと 250m CPU です。

スーパーセットの展開

1. **Asset ライブラリー** に移動し、HM コンソールで **Apps** を選択します。
2. **Apache Superset** を選択し、 **Deploy** を選択します。
3. **Identity** の下に、展開の名前を入力し、ターゲットプロジェクトを選択します。
4. **Parameters** の下で、必須フィールドに入力します。フィールドの説明については、*HNSW* **パラメーターインデックスタイプ=** *hnsw* を参照してください。
5. **Deploy** を選択してスーパーセットをインストールします。

パラメーター

Parameter	Description	Required
PostgreSQL Host	PostgreSQL サーバーのホスト名または IP アドレス。例 *postgres.database.svc.cluster.local*、*prod-postgres.example.com*、*10.0.1.50*	Yes
Database name	PostgreSQL データベースの名前。	Yes
Database username	Superset アプリケーションの PostgreSQL ユーザー。	Yes
Database password	データベース接続の PostgreSQL パスワード。	Yes
PostgreSQL Port	PostgreSQL 接続のポート番号。デフォルト `5432`	No
replicaCount	スーパーセットレプリカの数。デフォルト `1`	No
CPU limit (node)	スーパーセットノードの最大 CPU。デフォルト `1000m`	No
Memory limit (node)	スーパーセットノードの最大メモリ。デフォルト `2Gi`	No
CPU limit (worker)	スーパーセットワーカーの最大 CPU。デフォルト `1000m`	No
Memory limit (worker)	スーパーセットワーカーの最大メモリ。デフォルト `2Gi`	No
CPU request (worker)	スーパーセットワーカーの要求された CPU。デフォルト `500m`	No
Memory request (worker)	スーパーセットワーカーの要求されたメモリ。デフォルト `1Gi`	No

展開後の構成

スーパーセットが展開されたら、次の手順を実行してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールから **Superset** を開きます。
3. **Settings > Database Connection** に移動して、分析のためのデータソースを追加します。

サポートリソース

Superset OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- Superset community —コミュニティからのピアツーピアサポート。
- Superset documentation —セルフサービスの展開ガイドと機能ガイド、およびスーパーセットのインストールと構成に関する追加情報。
- Superset GitHub repository —ソースコード、問題トラッカー、およびリリースノート。

Metabase

重要

通常のアップストリームリリースでは、セキュリティ更新が配信されます。 Metabase インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

Metabase は、コミュニティにサポートされているオープンソースのビジネスインテリジェンスツールであり、Hybrid Manager のアプリとして利用できます。

主な機能

- SQL を必要としない質問ビルダー 。
- アドホッククエリーの SQL モード。
- インタラクティブなビジュアライゼーションとダッシュボード。
- データ探索と自動分析。
- 埋め込み可能なチャートとダッシュボード。
- 複数のデータベースバックエンド PostgreSQL、MySQL などのサポート。

要件

サポートされているデータソース

運用環境の場合、Metabase では、Metabase アプリケーションデータベースとして PostgreSQL を使用することをお勧めします。

- PostgreSQL プライマリ、および上流のメタベースイメージでサポートされているさまざまな SQL および NoSQL データベース。

展開する前に、PostgreSQL データベース接続の詳細ホスト、ポート、データベース名、ユーザー名、およびパスワードを準備します。これらは、HM コンソールのクラスターの **Connect** タブから取得できます。リソース制限を含む必須フィールドとオプショナルフィールドの完全なリストについては、*HNSW パラメーターインデックスタイプ= hnsw* を参照してください。

メタベースの展開

Metabase をハイブリッドマネージャー HM に直接インストールします。

1. HM コンソールで、 **Asset ライブラリー** に移動し、 **Apps** を選択します。
2. **Metabase** 、 **Deploy** を選択し、必須フィールドに入力します。フィールドの説明については、*HNSW パラメーターインデックスタイプ= hnsw* を参照してください。

パラメーター

Parameter	Description	Required
PostgreSQL Host	PostgreSQL サーバーのホスト名または IP アドレス。例 *postgres.database.svc.cluster.local*、*prod-postgres.example.com*、*10.0.1.50*	Yes
Database Name	PostgreSQL データベースの名前。 PostgreSQL サーバーに存在する必要があります。	Yes
Database User Name	Metabase アプリケーションの PostgreSQL ユーザー。データベースにアクセスする必要があります。	Yes
Database Password	データベース接続の PostgreSQL パスワード。	Yes
PostgreSQL Port	PostgreSQL 接続のポート番号。デフォルト `5432`	No
CPU request	メタベースポッドの要求された CPU。デフォルト `500m`	No
Memory request	メタベースポッドの要求されたメモリ。デフォルト `1Gi`	No
CPU limit	メタベースポッドの最大 CPU。デフォルト `1000m`	No
Memory limit	メタベースポッドの最大メモリ。デフォルト `2Gi`	No

注釈

HM にメタベースを展開すると、HM コンソールの**Estate > Apps**またはプロジェクトの**Apps**ビューからアクセスできます。 !!!

展開後の構成

Metabase が展開されたら、次の手順を実行してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動します。セットアップウィザードは、残りの構成手順をガイドします。
2. Metabase で管理アカウントを作成して、後続のすべてのユーザー追加、データベース接続、およびシステム構成を管理します。
3. セルフサービス分析、アプリケーションに埋め込まれた分析、またはその両方を選択します。これは後で変更できます。
4. データベースのホスト名、ポート、名前、および資格情報を入力します。この手順をスキップして、Metabase に含まれるサンプルデータベースを探索することもできます。

-

Adding and managing databases - Connecting to supported databases

1. 匿名の使用状況データの収集を許可するかどうかを選択します。Metabase は、データまたは質問結果に関するものを収集することはありません。

サポートリソース

Metabase OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- Metabase discussion forum —コミュニティからのピアツーピアサポート。
- Metabase documentation —セルフサービスの展開ガイドと機能ガイド、およびメタベースのインストールと構成に関する追加情報。
- Metabase GitHub repository —ソースコード、問題トラッカー、およびリリースノート。

pgAdmin 4 (v8.12)

> **重要**
>
> 通常のアップストリームリリースでは、セキュリティ更新が配信されます。 pgAdmin インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

pgAdmin 4 は、PostgreSQL データベースを管理するためのオープンソースの Web ベースのインターフェイスです。ハイブリッドマネージャーにアプリとして展開されると、データベースオブジェクトを管理し、クエリを実行するための PostgreSQL クラスターへのリモートアクセスを提供します。

> **警告**
>
> pgAdmin 4 は、Red Hat OpenShiftRHOS ではサポートされていません。 !!!

主な機能

- PostgreSQL データベースオブジェクトを管理するためのグラフィカルインターフェイス。
- 構文の強調表示、クエリ履歴、および実行プランの視覚化を備えたクエリツール。
- 統計、ログ、アクティビティを含むサーバーモニタリング。
- バックアップおよび復元ツール。
- スキーマ、ロール、および権限の管理。

要件

技術的要件

- Kubernetes クラスター。
- 永続ボリューム データストレージ用に 10Gi デフォルト。

- リソース 最小 512Mi メモリ、500m CPU 制限 1Gi メモリ、1000m CPU。

サポートされている PostgreSQL バージョン

- PostgreSQL 12〜17。
- EDB Advanced Server 12〜16。

pgAdmin 4 の展開

1. HM コンソールで、 **Asset ライブラリー** に移動し、 **Apps** を選択します。
2. **pgAdmin 4** を選択し、バージョンを選択して、 **Deploy** を選択します。
3. **Identity** の下に、展開の名前を入力し、ターゲットプロジェクトを選択します。
4. 必要に応じて、レプリカ数デフォルト 1 とストレージサイズを構成します。
5. **展開** を選択します。

> **注釈**
>
> HM に pgAdmin 4 を展開すると、HM コンソールで**Estate > Apps**またはプロジェクトの**Apps**ビューからアクセスできます。 !!!

展開後の構成

pgAdmin 4 が展開されたら、次の手順を完了してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールから **pgAdmin 4** を開きます。
3. マスターパスワードを設定します。
4. pgAdmin インターフェイス内で PostgreSQL サーバーを登録して、データベースの管理を開始します。

サポートリソース

pgAdmin OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- pgAdmin support page — pgAdmin 4 に関連するクエリのサポートを得ます。
-

pgAdmin documentation —
  セルフサービスの展開ガイドと機能ガイド、および pgAdmin のインストールと構成に関する追加情報。

- pgAdmin GitHub repository —ソースコード、問題トラッカー、およびリリースノート。

pgAdmin 4 (v9.13)

> **重要**
>
> 通常のアップストリームリリースでは、セキュリティ更新が配信されます。 pgAdmin インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

pgAdmin 4 は、PostgreSQL データベースを管理するためのオープンソースの Web ベースのインターフェイスです。ハイブリッドマネージャーにアプリとして展開されると、データベースオブジェクトを管理し、クエリを実行するための PostgreSQL クラスターへのリモートアクセスを提供します。

> **警告**
>
> pgAdmin 4 は、Red Hat OpenShiftRHOS ではサポートされていません。 !!!

主な機能

- PostgreSQL データベースオブジェクトを管理するためのグラフィカルインターフェイス。
- 構文の強調表示、クエリ履歴、および実行プランの視覚化を備えたクエリツール。
- 統計、ログ、アクティビティを含むサーバーモニタリング。
- バックアップおよび復元ツール。
- スキーマ、ロール、および権限の管理。

要件

技術的要件

- Kubernetes クラスター。
- 永続ボリューム データストレージ用に 10Gi デフォルト。
- リソース 最小 512Mi メモリ、500m CPU 制限 1Gi メモリ、1000m CPU。

サポートされている PostgreSQL バージョン

- PostgreSQL 13〜18。
- EDB Advanced Server 13〜18.

pgAdmin 4 の展開

1. HM コンソールで、 **Asset ライブラリー** に移動し、 **Apps** を選択します。
2. **pgAdmin 4** を選択し、バージョンを選択して、 **Deploy** を選択します。
3. **Identity** の下に、展開の名前を入力し、ターゲットプロジェクトを選択します。
4. 必要に応じて、レプリカ数デフォルト 1 とストレージサイズを構成します。

5. **展開** を選択します。

> **注釈**
>
> HM に pgAdmin 4 を展開すると、HM コンソールで**Estate > Apps**またはプロジェクトの**Apps**ビューからアクセスできます。 !!!

展開後の構成

pgAdmin 4 が展開されたら、次の手順を完了してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールから **pgAdmin 4** を開きます。
3. マスターパスワードを設定します。
4. pgAdmin インターフェイス内で PostgreSQL サーバーを登録して、データベースの管理を開始します。

サポートリソース

pgAdmin OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- pgAdmin support page — pgAdmin 4 に関連するクエリのサポートを得ます。
-

pgAdmin documentation —
  セルフサービスの展開ガイドと機能ガイド、および pgAdmin のインストールと構成に関する追加情報。

- pgAdmin GitHub repository —ソースコード、問題トラッカー、およびリリースノート。

pgBadger

> **重要**
>
> 通常のアップストリームリリースでは、セキュリティ更新が配信されます。 pgBadger インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

pgBadger は、HM クラスターから PostgreSQL ログを分析し、データベースアクティビティ、パフォーマンス、エラーに関する統計を含む HTML レポートを生成します。

主な機能

- パラレル ログ パース。
- インタラクティブなグラフを含む HTML レポート。
- クエリーの識別が遅い。

- 接続とセッションの統計。
- ロックとチェックポイント分析。
- 一時ファイルと自動バキュームのモニタリング。
- stderr、csvlog、および syslog ログ形式のサポート。

要件

技術的要件

- Kubernetes または OpenShift クラスター。
- 永続性レポートとアップロードされたログを保存するためのデフォルトの 1Gi ボリューム。

構成要件

- リテンション `reportRetentionDays` デフォルト 30 は、レポートを保持する期間を制御します。
- アップロード制限 `maxUploadSize` デフォルト 50MB は、分析できるログファイルのサイズを制限します。

サポートされているデータソース

- PostgreSQL ログファイルテキスト形式。

pgBadger の展開

1. **Asset ライブラリー** に移動し、HM コンソールで **Apps** を選択します。
2. **pgBadger** を選択し、予想されるログボリュームデフォルト 512Mi メモリに基づいてリソース制限を調整します。
3. **展開** を選択します。

> **注釈**
>
> HM に pgBadger を展開すると、HM コンソールの**Estate > Apps**またはプロジェクトの**Apps**ビューからアクセスできます。 !!!

展開後の構成

pgBadger が展開されたら、次の手順を完了してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールで **pgBadger Reports** インターフェイスを開きます。
3. PostgreSQL ログファイルをアップロードして、新しいビジュアルレポートを生成します。

サポートリソース

pgBadger OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- pgBadger support — pgBadger に関連するクエリのサポートを得ます。
- pgBadger documentation —セルフサービスの展開ガイドと機能ガイド、および pgBadger のインストールと構成に関する追加情報。
- pgBadger GitHub repository —ソースコード、問題トラッカー、およびリリースノート。

HammerDB

重要

通常のアップストリームリリースでは、セキュリティ更新が配信されます。 HammerDB インスタンスを最新の状態に維持して、最新のパッチを受信します。 !!!

HammerDB は、マルチプルのデータベースプラットフォームをサポートするオープンソースのデータベース負荷テストおよびベンチマークツールです。HammerDB は、ハイブリッドマネージャー HM に展開されると、既存の PostgreSQL クラスターに対して TPROC-C ベンチマークを実行し、スループット、レイテンシー、インフラストラクチャメトリックを含む詳細なパフォーマンスレポートを生成します。

注釈

HammerDB パッケージは PostgreSQL を展開しません。独自の PostgreSQL クラスターにスーパーユーザーまたは CREATEDB 特権を提供する必要があります。 !!!

主な機能

- HammerDB データベースの負荷テストおよびベンチマークツール。
- 構成可能なリソースとレプリカ。

要件

技術的要件

- **PostgreSQL クラスター** —ターゲット名前空間からアクセスできる既存の PostgreSQL インスタンス。
- スーパーユーザーまたは CREATEDB 特権が必要です HammerDB は独自のテストスキーマを作成します。
- `postgres://user:pass@host:5432/db?sslmode=require` 形式の接続文字列。

展開する前に、データベース接続の詳細ホスト、ポート、データベース名、ユーザー名、およびパスワードを準備します。これらは、HM コンソールのクラスターの **Connect** タブから取得できます。リソース制限を含む必須フィールドとオプショナルフィールドの完全なリストについては、*HNSW* **パラメーターインデックスタイプ**= *hnsw* を参照してください。

サポートされているデータベースバックエンド

Database	Release
Oracle (TimesTen)	12c / 18c / 19c / 21c
Microsoft SQL Server	2017 / 2019 / 2022
Db2	11.1 / 12.1
MySQL (Amazon Aurora)	5.7 / 8.0
MariaDB	10 / 11
PostgreSQL	10 / 11 / 12 / 13 / 14 / 15 / 16 / 17

HammerDB の展開

HammerDB を HM に直接インストールします。

1. HM コンソールで、 **Asset ライブラリー** に移動し、 **Apps** を選択します。
2. **HammerDB** を選択し、 **Deploy** を選択します。
3. **Identity** の下に、展開の名前を入力し、ターゲットプロジェクトを選択します。
4. **Parameters** の下で、必須フィールドに入力します。フィールドの説明については、*HNSW パラメーターインデックスタイプ= hnsw* を参照してください。
5. **Deploy** を選択して HammerDB をインストールします。

> **注釈**
>
> HM に HammerDB を展開すると、**Estate > Apps**またはプロジェクトの**Apps**ビューから HM コンソールにアクセスできます。 !!!

パラメーター

Parameter	Description	Required
connectionString	PostgreSQL 接続文字列。フォーマット `postgres://user:pass@host:5432/db?sslmode=require`。スーパーユーザーまたは CREATEDB 特権が必要です。	Yes
CPU limit	HammerDB ポッドの最大 CPU。デフォルト `1000m`	No
Memory limit	HammerDB ポッドの最大メモリ。デフォルト `2Gi`	No
size	HammerDB データとベンチマーク結果を保存するための永続ボリュームのサイズ。デフォルト `5Gi`	No

展開後の構成

HammerDB を展開したら、次の手順を実行してセットアップを完了します。

1. **Project** に移動し、 **Apps** タブを選択するか、 **Estate > Apps** に移動して、展開ステータスを確認します。ステータスに **Ready** と表示されたら、アプリを起動して続行します。
2. HM コンソールから **HammerDB** を開きます。
3. ベンチマークワークロードを構成し、PostgreSQL クラスターに対して TPROC-C テストを実行します。

サポートリソース

HammerDB OSS はコミュニティでサポートされているため、技術的なガイダンスと更新には次のリソースを使用してください。

- HammerDB GitHub repository —ソースコード、問題トラッカー、コミュニティからのピアツーピアサポート。
-

HammerDB documentation —
セルフサービスの展開ガイドと機能ガイド、および HammerDB のインストールと構成に関する追加情報。

-

HammerDB Blog —パフォーマンスの更新とガイド。

5.3.3 For Users of Hybrid Manager

5.3.3.1 Projects

Hybrid ManagerHM では、プロジェクトを使用して、組織に関連するリソースを関連付けます。プロジェクトにより、これらのリソースの管理とアクセスが簡素化されます。

プロジェクトのリストを表示するには、 **Projects** タブを選択します。各プロジェクトカードには次のものが表示されます。

- プロジェクト名
- プロジェクト内のクラスターの数
- プロジェクトにアクセスできるユーザーの数
- プロジェクトにタグを追加するためのコントロール、リソースを整理するのに役立つキーと値のペア
- プロジェクトで構成されたアクティブなクラウドサービスプロバイダー CSP のログ。

各カードの省略記号メニューから、 プロジェクトの詳細の表示時に実行できる機能にすばやくアクセスできます。

所有者権限がある場合は、このビューからプロジェクトを作成することもできます。

5.3.3.1.1 プロジェクトの作成

1. **Projects** タブを選択します。
2. **Create New Project** を選択します。
3. プロジェクトの名前を入力します。
4. オプションで、タグをプロジェクトに追加します。
5. **Create New Project** を選択します。

プロジェクトを作成すると、プロジェクトの詳細が表示されます。あなたは新しいプロジェクトの所有者であり、リソースの追加を開始できます。

5.3.3.1.2 プロジェクトの詳細

プロジェクトの詳細ビューでは、プロジェクト管理タスクを実行します。詳細ビューを開くには、プロジェクトを作成するか、**プロジェクト**ページのプロジェクトのリストからプロジェクトを選択します。このビューには、プロジェクト内の各クラスターの説明と、メモリ使用量、CPU 使用量、ストレージ使用量、ディスク IOPS とトランザクションレート、データベースサイズを含むメトリックが含まれています。

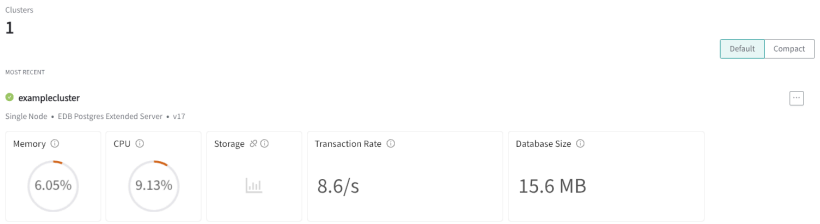


図 2: Project Overview

プロジェクトでは、次のことができます。

- Create, view, and manage clusters 。
- *Manage storage locations* 。
- カタログを追加します。
- *View and manage the users in the project* 。
- アクティビティログを表示およびダウンロードします。アクティビティログは、クラスターの作成と削除、ソフトウェアのインストールなど、現在のプロジェクトで実行されたすべてのアクションの記録です。
- Manage project settings 通知、セキュリティ、アラートを含む
- *View a usage report* 。
- View and manage database migrations 。
- *Access OSS Library apps* 。

クラスターごとに、カードの右上隅にある省略記号メニューを使用して、次のこともできます。

- クラスターを編集する に **Edit Cluster** を選択します。

- ** Credential Information を選択して、接続 URL、ユーザー名、データベース名を含むクラスターの接続の詳細を表示します。これらの詳細を使用して、Postgres クライアントからクラスターに接続できます。パスワードは表示されませんが、必要に応じてパスワードをリセットできます。詳細については、ステップ *2* 資格情報 を参照してください。

5.3.3.1.3 プロジェクトの管理

プロジェクトの名前を変更したり、削除したりできます。プロジェクトの管理 を参照してください。

5.3.3.1.4 投影されたオブジェクトストレージの構成

デフォルトでは、すべてのプロジェクトは、インストール中に構成された同じオブジェクトストレージを共有します。プロジェクトリソースを分離する必要がある場合は、プロジェクトごとに専用のオブジェクトストレージを構成できます。投影されたオブジェクトストレージの構成 を参照してください。

Managing projects

プロジェクトの名前を変更したり、削除したりできます。

プロジェクトの名前の変更

プロジェクトの一括ビューと **プロジェクト**タブでプロジェクトの名前を変更できます。プロジェクトのいずれかのリストから

1. プロジェクト名の右側にある省略記号メニューから、 **Edit** を選択します。
2. 新しいプロジェクト名を入力し、 **Update Project** を選択します。

プロジェクトの削除

プロジェクトの一括ビューと **プロジェクト**タブでプロジェクトを削除できます。

> **注釈**
>
> デフォルトプロジェクトは削除できません。ハイブリッドマネージャーは、組織ごとに 1 つのデフォルトプロジェクトを保持し、削除から保護します。

プロジェクトを削除する前に、プロジェクトにアクティブなリソースがない必要があります。アクティブなリソースには、環境にインストールされている機能に応じて、クラスター、データカタログ、マーケットプレイスアプリ、展開されたフロー、ナレッジベース、パイプラインが含まれます。プロジェクトにこれらのリソースがまだ含まれている場合、ハイブリッドマネージャーは削除をブロックし、最初に削除する必要があるリソースをリストします。

クラスターは、プロジェクトの削除ダイアログ内から削除できます。開始する前に、独自の管理ページから他のリソースタイプを削除します。

プロジェクトのいずれかのリストから

1. プロジェクト名の右側にある省略記号メニューから、 **Delete** を選択します。

削除の確認を求められます。

1. プロジェクトにクラスターが含まれている場合、プロジェクトを削除する前にクラスターを削除する必要があります。ディスプレイには、残りのクラスターが表示され、各クラスターの横にチェックボックスが表示されます。削除する各クラスターのチェックボックスを選択するか、**Select All** を選択します。

2. 削除するクラスターを選択したら、テキストボックスに `delete clusters` と入力して、削除を確認します。このテキストを入力すると、[**Delete All Clusters**] ボタンが有効になります。

3. **Delete All Clusters** を選択して、選択したクラスターを完全に削除します。

4. プロジェクトにアクティブなリソースが含まれていない場合、ダイアログボックスが表示され、チェックボックス付きプロジェクト名をリストします。チェックボックスを選択し、テキストフィールドに `delete project` と入力します。このテキストを入力すると、[**Delete Project**] ボタンが有効になります。

5. プロジェクトを削除するには、 **Delete Project** を選択します。

Settings

通知、セキュリティ、アラートなどのプロジェクト固有の設定を構成できます。**Settings** メニューを使用して、これらの設定にアクセスします。

- **プロファイル**を選択して、プロジェクトの名前をいつでも変更できます。プロジェクト名は、ハイブリッドマネージャー HM インスタンスで一意である必要があります。
- ***Notifications*** を選択して、プロジェクトの通知の送信方法を構成します。
- **Security** を選択して、透過的データ暗号化 TDE の外部管理キーをプロジェクトに追加します。後のクラスター作成中にこれらのキーを使用して、データの暗号化を追加できます。KMS 固有の手順については、 *Enabling KMS for TDE* を参照してください。
- ***Alerts*** を選択して、プロジェクトのアラート設定を表示し、アラートを送信する方法と時期を制御します。

Notifications

Hybrid ManagerHM を使用すると、受信する通知の種類をカスタマイズできます。2 種類の通知チャネルがあります。

- **ユーザー通知**電子メール、アプリ内の受信ボックス、またはその両方を介して通知を受信できます。これらの通知は、ロールで許可されている場合に有効にしているすべてのユーザーを対象としています。
- **システム通知**電子メールメールグループ、Webhook、ポケベルデューティ、または任意の組み合わせを介して通知を受信できます。これらの通知は、主に組織内のタスクを自動化するために、イベントごとに 1 回トリガーされます。

HM でのロールにより、通知設定へのアクセスが決まります。次の種類の通知が使用できます。

- アカウントユーザー通知チャンネル用。次の通知を受け取ることができます。
- 新しいプロジェクトロールがあなたに割り当てられました。
- プロジェクトロールがあなたから割り当て解除されました。

- 個人アクセスキーの有効期限が間もなくエキスパイアと。
- グローバルユーザーおよびシステム通知チャネル用組織の所有者にのみ可視。次の通知を受け取ることができます。
- 新しいプロジェクトが作成されました。
- マシンアクセスキーの有効期限が間もなくエキスパイア。
- プロジェクトユーザーおよびシステム通知チャネル用。次の通知を受け取ることができます。
- アクティブなクラスターは一時停止されました。
- 一時停止されたクラスターが再アクティブ化されました。

通知の設定

プロジェクトの所有者/編集者および組織の所有者/管理者は、表示できるイベントの通知を有効または無効にできます。通知は、アプリ内の受信ボックス、電子メール、またはその両方で受信できます。

ユーザーが行った通知設定は、そのユーザーにのみ適用されます。電子メール通知が有効になっている場合、電子メールはユーザーのログインに関連付けられた電子メールアドレスに送信されます。

システム通知チャネルは、組織の所有者がグローバルレベルで、またはプロジェクト所有者がプロジェクトレベルで構成できます。デフォルトでは、各組織のプロジェクトはグローバルレベルのシステム通知チャネルを継承します。ただし、プロジェクトレベルでシステム通知チャンネルを有効/無効または再構成して、デフォルト設定をオーバーライドすることができます。

グローバルレベルで通知を設定する

組織の所有者のみがグローバル通知を設定できます。

Webhook の場合

1. プロフィールメニューから、 **設定** を選択します。
2. **Notifications** タブを選択します。
3. **Webhook** を選択して、次を入力します。

- webhook のホスト URL b_tran_2 Webhook の方法-POST または PUT

1. オプションで、次の認証パラメーターのいずれかを提供します。

- 基本認証の場合 ユーザー名とパスワード
- ベアラートークン

1. **Save** を選択します。

PagerDuty の場合

PagerDuty 通知チャンネルを構成する前に

- 昇格した権限を持つ PagerDuty アカウントへのアクセスが必要です。
- PagerDuty サービスは、 Events API v2 統合を使用して作成および構成する必要があります。

- PagerDuty アカウントで、目的のサービスに移動し、その **統合**設定にアクセスします。
- 新しい Events API v2 統合を追加して構成します。
- 新しく作成した統合用に PagerDuty が提供した統合キールーティングキーをコピーします。

PagerDuty 通知チャンネルを構成するには

1. プロフィールメニューから、 **設定** を選択します。
2. **Notifications** タブを選択します。
3. **PagerDuty** を選択します。
4. ルーティングキー統合キーを設定します。
5. **Save** を選択します。

電子メールの場合

1. プロフィールメニューから、 **設定** を選択します。
2. **Notifications** タブを選択します。
3. **電子メール** を選択し、次のように入力します。

- 電子メールの宛先 - オプションで、システムメール通知の場合電子メールグループ
- 電子メールの送信元 - 電子メールを送信している電子メールの場合
- ホスト - 電子メールの URL サービス
- ポート - ポート email service
- ユーザー名 - 電子メールサービスのユーザー名
- パスワード - 電子メールサービスの

1. **Save** を選択します。

プロジェクトレベルで通知を設定する

プロジェクトレベルの通知設定は、グローバル設定をオーバーライドできます。各プロジェクトは、一意の構成を持つことができます。

1. **Projects** タブを選択します
2. 任意のプロジェクトを選択します。
3. プロジェクトの概要ページに移動します。
4. 左側のメニューで **Settings** を選択します。
5. **通知** を選択します。
6. 各システム通知チャネルに対して グローバルレベルで通知を設定する の下の手順に従います。

通知の表示

次のロールのユーザーは通知を表示できます。

- 組織の所有者はグローバル通知を表示できます。
- プロジェクトの所有者/編集者は、プロジェクトの通知を表示できます。
- すべてのユーザーが自分のアカウント通知を表示できます。

各通知は、HM で複数のロールが割り当てられているユーザーが属しているレベルとプロジェクトを示します。

アプリ内通知を表示するには、HM コンソールの上部にあるベルを選択します。通知を読んだり、未読としてマークしたり、アーカイブしたりできます。

メール通知を表示するには、設定したメールアドレスの受信箱を確認します。

通知設定の管理

1. プロフィールメニューから、 **My Account** を選択します。
2. **Notifications** タブを選択します。
3. **Account** 、**Global** 、または **Project** セクションまでスクロールして、管理するイベントを表示します。
4. セクションを展開して、イベントのリストを表示します。
5. 通知ごとに、次のことができます。 - イベント名の前にあるトグルボタンを使用して、イベントの通知の受信をオプトインまたはオプトアウトします。- チェックボックスを使用して、メールとアプリ内通知をカスタマイズします。

Alerts

アラートは、ハイブリッドマネージャー HM 可観測性エコシステムに不可欠です。Postgres クラスターのパフォーマンスに影響を与える重要なイベントについて積極的に通知します。重要なパフォーマンスメトリックを継続的に監視し、事前定義されたしきい値を超過するたびにアラートをトリガーします。この堅牢なアラートシステムは、HM エコシステムにシームレスに統合され、すべてのリージョンの Postgres クラスターとそのインフラストラクチャから収集されたリアルタイムデータを活用します。アラートは、HM で作成されたプロジェクトごとにデフォルトで有効になっています。

アラートシステムはタイムリーな通知を保証し、情報に基づいた迅速なアクションを可能にします。特定のしきい値と間隔を構成して、アラートエクスペリエンスをカスタマイズできます。モニタリングとアラートのこの統合により、潜在的な問題への迅速な対応が可能になり、最適なクラスターパフォーマンスを維持できます。

アラートを表示する

クラスターの詳細ページの **モニタリング**タブにあるアラートにアクセスし、特定のクラスターのアラートの詳細を表示できます。

HM クラスターの場合

1. HM コンソールで、 **Projects** を選択します。

2. リストからプロジェクトを選択します。
3. 左側のナビゲーションで、 **Clusters** を選択します。
4. 準備ができているクラスターを選択します。
5. クラスターの詳細ページで、 **Monitoring** タブを選択します。

外部 HM クラスターの場合

1. HM コンソールで、 **Estate** を選択します。
2. すべてのクラスターをリストしているテーブルに移動します。
3. **管理** が **自己管理**である準備ができているクラスターを選択します。
4. クラスターの詳細ページで、 **Monitoring** タブを選択します。

[**Monitoring**] タブには、アクティブなアラートリストが表に表示されます。このタブの上部に、アクティブなアラートの概要を表示できます。考えられる重大度レベルは、高、中、および低です。

次を使用して特定のアラートを検索できます。

- クエリベースの検索のための検索ボックス
- 特定の日付範囲内を検索する日付範囲セレクター
- 重大度レベルで検索するフィルタ

Node 列でノードを選択して、アラートがトリガーされたときの状態でチャートを表示します。ノードを選択すると、**Monitoring** タブのフィルタが適用され、関連するチャートが表示されます。

次のこともできます。

- **Download CSV** を選択して、アラートの詳細を CSV 形式でダウンロードします。
- 重大度、アラート開始時間、およびアラート名に基づいて、アラートリストを並べ替えます。
- **Refresh** を選択してリストを更新します。
- 列セレクターを使用して、テーブルに表示する列を選択します。

アラートの構成

プロジェクト管理者は、プロジェクトレベルでアラートを構成できます。アラートが構成および適用されると、システムがアラートを完全に処理するまでに数分かかります。

アラートを構成するには

1. HM コンソールの上部で、 **Projects** を選択します。
2. リストからプロジェクトを選択します。
3. 左側のナビゲーションで、 **Settings** を選択します。
4. **アラート** を選択します。

アラートごとに、 **Edit** を選択して設定します。

- Enable/Disable 特定のアラートを有効または無効にします。

- Notify After アラートがトリガーされるまでの時間を設定します。
- しきい値超過した場合にアラートをトリガーするパーセンテージまたは値を指定します。

これらのアラートはデータベースレベルで構成できます。

- 接続の使用状況
- 最近の WAL アーカイブは成功しませんでした
- ノードダウン
- バイト単位のレプリケーションラグ
- 合計接続数
- 非アクティブなレプリケーションスロットの合計
- トランザクションのロールバック
- WAL は成長し続けます
- WAL のディスク領域が不足しています

これらのアラートはホストレベルで構成できます。

- CPU 使用率
- ディスク IOPS
- ディスクの使用量
- メモリ使用量

Configure dedicated object storage for a project

デフォルトでは、すべてのプロジェクトは、インストール中に構成された同じオブジェクトストレージを共有します。プロジェクトリソースを分離する必要がある場合は、プロジェクトごとに専用のオブジェクトストレージを構成できます。

1. Hybrid Manager コンソール HM で、 **Projects** タブを選択します。
2. 専用オブジェクトストレージを構成するプロジェクトを選択します。
3. URL からプロジェクト ID を取得します。プロジェクト ID は、URL の`/projects/` 以降の文字列です。

たとえば、URL `https://hybrid-manager.example.com/projects/prj_Abc123` では、プロジェクト ID は `prj_Abc123` です。

4. すべてのアンダースコア_ をハイフン- に置き換えることによりプロジェクト ID を正規化し、すべての文字を小文字に変換します。例、`prj_Abc123` は `prj-abc123` になります。
5. Kubernetes クラスターにオブジェクトストレージシークレットを作成します。*Configuring object storage* のプロバイダー固有の `stringData:` ブロックを使用します。シークレット本文は、デフォルトの `edb-object-storage` シークレットと同じです。名前のみが異なり、`edb-object-storage-<normalized-project-id>` である必要があります。

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage-<normalized-project-id>
  namespace: default
stringData:
  # Provider-specific fields — see Configuring object storage.
  ...
```

- プロジェクトで Postgres ワークロードを作成する前に、専用のオブジェクトストレージを構成します。プロジェクトがオブジェクトストレージシークレットにバインドされると、関連付けを変更することはできません。
- **HM 展開に複数の場所がある場合、オブジェクトストレージシークレットをすべての場所に適用します。 .. ::**
  ## バインディングの検証

HM は、プロジェクトからストレージへのマッピングを `default` 名前空間の `slo-kv-store` ConfigMap に保存します。プロジェクトがどのオブジェクトストレージにバインドされているかを確認するには、次のコマンドを実行します。

```
kubectl get cm slo-kv-store -o yaml -n default
```

`data` セクションは、各プロジェクト ID をそのオブジェクトストレージシークレット名にマッピングします。例

```
data:
  prj_C9cgaIh5QAZdX977: edb-object-storage-prj-c9cgaih5qazdx977 # dedicated storage
  prj_EyHfzLGVDmQCsEC2: edb-object-storage # default storage
```

プロジェクト ID が ConfigMap に表示されない場合、プロジェクトはまだバインドされていません。バインディングは、プロジェクト内で最初の Postgres ワークロードが作成されるときに自動的に発生します。

Managed storage locations

管理対象ストレージの場所は、将来のリリースでプロジェクトのデータを保存するために使用されます。Hybrid ManagerHM コンソールで、プロジェクトと同じストレージを共有するストレージの場所を作成できます。ただし、HM サービスは管理対象ストレージ場所機能を使用していません。

Usage report

使用レポートページは、クラスターの次の履歴使用メトリックのビューを提供します。

Term	Definition	Measure
Cluster Name	The name of the cluster.	•

次のページに続く

表 7 – 前のページからの続き

Term	Definition	Measure
Cluster Type	The type of the cluster.	•
Database Type	The type of the database.	•
Disk Allocated (GiB-Hour)	The amount of disk storage allocated.	GiB-Hours
Disk IO (GiB)	The amount of data read from and written to disk storage.	GiB
Disk Used (GiB-Hour)	The amount of disk storage used.	GiB-Hours
Disk Used Percent	The total amount of time that the allocated storage (disk) has been available to your database system. This does not measure activity like Disk IO, but the total duration the storage has been provisioned and available.	Hours
Memory Allocated (GiB-Hour)	The amount of memory allocated.	GiB-Hours
Memory Used (GiB-Hour)	The amount of memory used.	GiB-Hours
Memory Used Percent	The amount of RAM (memory) consumed in hours versus the amount allocated.	•
Project	The name of the project.	•
Uptime (Hour)	The total amount of time the database instance has been continuously running and available.	Hour
vCPU Allocated (Core-Hour)	The amount of vCPU allocated.	Core-Hours
vCPU Used Percent	The number of virtual CPU hours consumed versus the number allocated.	•
vCPU Used (Core-Hour)	The amount of vCPU used.	Core-Hours

5.3.3.2 Working with clusters

Hybrid ManagerHM を使用して、データベースクラスターを作成および管理できます。データベースは、次の Postgres タイプのいずれかに基づくことができます。

- EDB Postgres Advanced Server
- EDB Postgres 拡張サーバー PGE
- PostgreSQL

クラスター管理には、新しいクラスターの作成、アクセスの制御、クラスターパフォーマンスの監視、およびクラスターの編集、復元、または削除などのライフサイクル操作の管理が含まれます。再利用可能な

テンプレートを作成して、今後のクラスターの作成を簡素化することもできます。

5.3.3.2.1 何ができるでしょうか？

- Create a cluster

事前定義されたテンプレートから新しいデータベースクラスターを作成する方法、または最初から構成する方法を学びます。

- Manage a cluster

構成の変更、バックアップの復元、クラスターの削除、テンプレートの作成などのキー管理タスクを実行します。

- *Manage database access*

ユーザー、ロール、および権限を設定して、クラスターにアクセスおよび接続できる人を制御します。

- View clusters

すべてのデータベースクラスターのステータス、構成、およびアクティビティを監視および確認します。

Creating a database cluster

テンプレートがある場合からデータベースクラスターを作成するか、クラスターをスクラッチから作成できます。

スタートポイント

クラスターを作成するための開始点は、 **Estate** ページまたはプロジェクトページのいずれかです。 **Estate** ページから開始する場合、データベースを作成するプロジェクトを指定します。プロジェクトページから開始すると、そのプロジェクトにデータベースを作成します。

Estate ページから開始するには

1. **Quick Actions > Create New Database Cluster** を選択します。

複数のプロジェクトがある場合、プロジェクトを選択するように求められます。

2. プロンプトが表示されたら、データベースを作成するプロジェクトを選択します。

プロジェクトを選択した後、またはプロジェクトが 1 つだけの場合、 **Create Cluster** *[パスの選択] ページから* ページが開きます。

プロジェクトページから開始するには、これらのアプローチのいずれかを使用できます。プロジェクトページに移動するには、**Projects** ページからプロジェクトを選択します。

- プロジェクトページで、 **Create New > Database Cluster** を選択します。
- プロジェクトページで、**Clusters > Postgres Cluster** を選択します。[**クラスター**] ページで、[**Create New > Database Cluster**] を選択します。

どちらの場合も、クラスターを作成するための **Choose your path** ページが開きます。

道を選択してください

[**Choose Your Path**] ページでは、事前定義されたテンプレートからクラスターを作成するか、カスタムビルドとしてスクラッチから作成するかを指定します。

- 使用可能なテンプレートからクラスターを作成するには、 **Use Template** を選択します。 *Creating a cluster from a template* を参照してください。
- 使用可能なテンプレートがない場合、またはクラスターを最初から作成する場合は、**Custom Build** を選択します。

Create New Template を選択して、ここからテンプレートの作成を開始することもできます。*Create a database cluster template* を参照してください。

クラスターを最初から作成する

Custom Build を選択したら、新しいクラスターの指定を開始できます。

Cluster Info タブで、クラスタータイプを選択します。

- シングルノード
- 高可用性
- 高度な高可用性
- 分散高可用性

作業するタブを選択するか、**Next** を使用してタブを順番に実行します。**DB 構成** および **追加設定** タブの代わりに **Advanced High Availability** または **Distributed High Availability** を選択した場合、***Data Groups*** タブが表示されます。

- ***Cluster Info***
- ***Cluster Settings***
- ***DB Configuration*** オプショナル
- ***Additional Settings*** オプショナル
- ***Replica Clusters*** シングルノードおよび高可用性クラスターの場合
- ***Data Groups*** 分散高可用性クラスター用

> **注釈**
>
> 構成しないオプションタブの設定はデフォルト値を使用します。

クラスターを作成する

タブの構成が終了したら、[**Create Cluster**] を選択します。設定が正しくないまたは欠落しているタブには、赤いインジケーターが表示されます。エラーを修正し、**Create Cluster** を再度選択します。

有効な設定で **Create Cluster** を選択すると、クラスターはプロビジョニングを開始し、クラスターのリストに戻ります。ポップアップはクラスターの作成を確認し、新しいクラスターがクラスターのリストに追

加されます。クラスターはバックグラウンドで作成されます。クラスターのリストで進行状況を確認できます。

クラスターの作成の進行中にクラスターの詳細を表示しようとすると、概要、バックアップ、ログ、プロパティのみを表示できます。

Cluster types to create

create your cluster を開始すると、[**Cluster Info**] タブが、作成するクラスターのタイプを選択するように求めます。このタブは、クラスターを編集しているときにも表示されます。

> **注釈**
>
> クラスターを編集するときは、シングルノードまたは高可用性クラスターと、高度なまたは分散高可用性クラスターを切り替えることはできません。

- ***Single Node*** は、1 つのプライマリとスタンバイレプリカを持たないクラスターを作成します。このタイプのクラスターは、高可用性が必要とされないテスト環境に適しています。PostgreSQL、EDB Postgres Extended Server、または EDB Postgres Advanced Server を実行するシングルノードクラスターを作成できます。
- ***High Availability*** は、異なるアベイラビリティーゾーンに 1 つのプライマリと複数のスタンバイレプリカを含むクラスターを作成します。PostgreSQL、 EDB Postgres Extended Server、または EDB Postgres Advanced Server を実行する高可用性クラスターを作成できます。

高可用性を選択すると、クラスターのスタンバイレプリカの数を指定するように求められます。デフォルトは 2 です。

- ***Advanced High Availability*** は、高速な自動フェイルオーバーを備えたシングルリージョン、マルチ AZ の復元力を提供します。オンラインアップグレード。プライマリ/スタンバイノードを介した論理レプリケーション。

Advanced High Availability を選択すると、タブ **DB Configuration** および **追加設定** は表示されません。代わりに [**Data Groups**] タブが表示されます。

- ***Distributed High Availability*** は、 EDB Postgres DistributedPGD を利用して、3 つのノードと 3 つのアベイラビリティーゾーンにまたがる最大 2 つのデータグループを使用してクラスターを作成します。この構成は、より高いパフォーマンスとより速いリカバリーを実現します。PostgreSQL、 EDB Postgres Extended Server、または EDB Postgres Advanced Server を実行する分散高可用性クラスターを作成できます。

分散高可用性 を選択すると、 **DB 構成** および **追加設定**タブが表示されません。代わりに [**Data Groups**] タブが表示されます。

詳細については、 **サポートされているクラスタータイプ** を参照してください。

Cluster setup

[**クラスター設定**] タブで、名前やパスワードなどのクラスター設定を構成します。

Cluster Name — クラスターの名前を入力します。

Password —クラスターのパスワードを入力します。これは、ユーザー edb_admin のパスワードです。このパスワードは、データベースに接続するために必要であり、ハイブリッドマネージャー HM には保存されません。

タグ — オプションで、 **+ Add Tag** を選択して、クラスターにタグを追加します。タグは、リソースを整理するのに役立つキーと値のペアです。

- 既存のタグを割り当てるには、 **タグ**の下の検索バーで、割り当てるタグが表示されるまでタグ名前の入力を開始します。クラスターに割り当てるタグを選択します。
- 新しいタグを追加するには、完全なタグ名を入力し、 **+ Add ``<tagname>``** を選択します。次に、タグの色を選択し、 **Create** を選択します。新しいタグがクラスターに割り当てられます。

シングルノードおよび高可用性クラスターの設定

次のオプションは、シングルノードおよび高可用性クラスターの場合にのみ**クラスター設定**タブに表示されます。他のクラスタータイプの場合は、 ***Data Groups*** タブでこれらのオプションを設定します。分散高可用性または高度な高可用性クラスターを作成する場合は、**Next: Data Groups** を選択してそのタブに移動します。

展開の場所 —データベースクラスターを展開する場所を選択します。リストの最初の場所は、EDB Postgres AI サービスをホストする Kubernetes クラスター内にあります。マルチ DC 環境でハイブリッドマネージャー HM が構成されている場合、データグループごとに異なる展開場所を選択して、データベースクラスターをマルチプルのデータセンターに分散できます。

データベースタイプ — 使用する Postgres のタイプを選択します。

1. **Postgres Type** フィールドで、使用する Postgres のタイプを選択します。

- ** EDB Postgres Advanced Server ** は、EDB の Oracle 互換データベース製品です。**マイグレーション** を表示します。 EDB Postgres Advanced Server は、すべてのクラスタータイプと互換性があります。
- ** EDB Postgres Extended Server ** は、高度な論理レプリケーションを使用する EDB の PostgreSQL 互換データベース製品です。
- ** PostgreSQL ** は、オープンソースのオブジェクトリレーショナルデータベース管理システムです。PostgreSQL は、シングルノードおよび高可用性クラスタータイプと互換性があります。

2. **Postgres Version** リストで、使用する Postgres のバージョンを選択します。
3. HM は、画像ライブラリに保存されている各 Postgres データベースの複数の画像をサポートしています。各イメージは、さまざまな拡張子を含むデータベースの構成です。クラスターに使用するイメージを選択します。詳細については、**アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です** を参照してください。通常、名前の最後に`-full` がないと、画像に拡張子はありません。名前の最後に`-full` を付けると、画像には使用可能なすべての拡張子が含まれます。

インスタンスサイズ —クラスターの CPU の数とメモリの量を選択します。選択できる CPU の数とメモリの量は、Kubernetes クラスターで使用可能なリソースによって異なります。

ストレージ

次のストレージ設定を指定できます。

先行書き込みログ用に別のストレージボリュームを使用します —デフォルトでは、HM はデータベースストレージとログ先行書き込み WAL ファイルに別のストレージボリュームを使用します。このデフォルトでは、最高のパフォーマンスと信頼性が得られます。チェックボックスをオフにして、WAL ファイルをデータベースストレージと同じボリュームに保存します。

デフォルトでは、HM はデータベースストレージと WAL ファイルに別個のストレージボリュームを使用します。このデフォルトでは、最高のパフォーマンスと復元力が得られます。このオプションをクリアすると、WAL ファイルはデータベースストレージと同じボリュームに保存されます。

データベースストレージ - **データベースストレージ**の下で、データベースストレージクラスとデータベースストレージボリュームのサイズを GiB 単位で選択できます。使用可能なストレージクラスは、基になるインフラストラクチャに設定した内容によって異なります。たとえば、EKS 展開の場合は gp2 および gp3、または RHOS 展開の場合は ssd-csi および standard-csi など。

Write-Ahead Logs Storage — **Write-Ahead Logs Storage** の下で、WAL ストレージクラスと WAL ストレージボリュームのサイズを GiB 単位で選択できます。使用可能なストレージクラスは、基になるインフラストラクチャに設定した内容によって異なります。たとえば、EKS 展開の場合は gp2 および gp3、または RHOS 展開の場合は ssd-csi および standard-csi など。

ネットワーキング

Connectivity Type では、プライベートネットワークデフォルトまたはパブリックネットワークを使用するかどうかを選択します。

- **プライベート**ネットワークでは、プライベートネットワーク内の IP アドレスのみがクラスターに接続できます。
- **パブリック**ネットワークとは、任意のクライアントがインターネットを介してクラスターのパブリック IP アドレスに接続できることを意味します。

接続読み取り専用

高可用性クラスターの場合、読み取り専用接続を有効にして読み取り専用操作をスタンバイレプリカにルーティングし、プライマリインスタンスのワークロードを削減し、全体的なパフォーマンスを向上させることができます。

> **注釈**
>
> 高可用性クラスターでのみ使用でき、シングルノードクラスターでは使用できません。

読み取り専用ワークロード —トグルを選択して、クラスターの専用の読み取り専用エンドポイントを有効にします。

有効にした場合、次の設定を構成します。

読み取り専用ワークロードネットワーク —読み取り専用エンドポイントにプライベートネットワークデフォルトまたはパブリックネットワークを使用するかどうかを選択します。

- **プライベート**ネットワークでは、プライベートネットワーク内の IP アドレスのみが読み取り専用エンドポイントを介して接続できます。
- **パブリック**ネットワーキングとは、任意のクライアントがインターネットを介して読み取り専用エンドポイントのパブリック IP アドレスに接続できることを意味します。

パブリック を選択した場合、オプションで**許可リストを使用してネットワークアクセスを制限し**、読み取り専用エンドポイントを介して接続を許可される IP レンジを指定できます。

有効にすると、クラスターの詳細の **Connect** タブで読み取り専用の接続文字列を利用できるようになります。

Database configuration parameters

シングルノードまたは高可用性クラスターを作成または編集するときは、[**DB Configuration**] タブを使用してデータベース構成パラメーターを指定します。

[**DB Configuration**] タブにリストされているデータベースパラメーターは、グランドユニファイド構成 GUC 変数とも呼ばれます。詳細については、What Is a GUC Variable? を参照してください。

パラメーターのリストは、クラスターを作成したときに [**クラスター設定**] タブで選択したデータベースのタイプに基づいて入力されます。データベースタイプのパラメーターの詳細については、

- PostgreSQL パラメーターについては、Postgres ドキュメントの

Setting Parameters および Server Configuration を参照してください。

- EDB Postgres Advanced Server の場合、

Summary of configuration parameters および Configuration parameters を参照してください。

- パラメーターが受け入れる値の種類の詳細については、

Parameter Names and Values
を参照してください。これらのタイプは、PostgreSQL と EDB Postgres Advanced Server パラメーターの両方に適用されます。

> **注釈**
>
> クラスターの作成が完了した後、データベースまたはサーバーパラメーターを変更できます。基になるハードウェアの物理的制限内でメモリおよび CPU 関連のデータベースパラメーターを構成してください。一部のパラメーターを変更するには、再起動が必要です。デフォルトの `wal_segment_size` は、PSR クラスター展開の場合は 64MB、PGD クラスター展開の場合は 128MB に設定されます。

パラメーターの変更

1. 次の方法を使用して、変更するパラメーターを見つけます。
   - 検索フィールドを使用してパラメーターを検索します。
   - **Show Only** リストのオプションを選択して、リストされたパラメーターをフィルターします。

- **Custom Values** —現在のセッションで、または以前のセッションで変更されたデフォルト値から変更されたパラメーターのみを表示します
- **現在編集中の値** —現在の編集セッション中に変更されたパラメーターのみを表示します。

2. パラメーター値フィールドに新しい値を入力します。エントリーには、リテラル値、または他のパラメーターに基づいて値を計算する *Formulas for parameter values* を指定できます。

> **警告**
>
> 黄色の感嘆符アイコンで示されるパラメーターは、変更を保存するときに再起動をトリガーします。再起動により、開いている接続がすべて終了され、完了までに数分かかります。新しい接続が確立されるまでにさらに数秒かかります。このプロセス中に、データベースに接続することはできません。

Backup and security settings

[**追加設定**] タブでは、バックアップ保持期間と一部のセキュリティ機能を指定できます。

バックアップ

バックアップ セクションの **保持期間**コントロールを使用して、デフォルトのデータベースバックアップの保持期間 30 日を変更します。保持期間は日、週、または月単位で構成できます。1〜180 日、1〜25 週間、または 1〜6 か月の範囲で保持期間を設定します。

バックアップ圧縮

HM は、 Barman Cloud プラグインを使用してデフォルトでバックアップを圧縮します。追加の構成は必要ありません。圧縮により、ストレージの消費量が削減され、バックアップパフォーマンスが向上するため、WAL ストレージのフルインシデントの防止に役立ちます。

デフォルトの圧縮アルゴリズムは次のとおりです。

- ベースバックアップ Snappy
- WAL アーカイブ zstd

それぞれが異なるアルゴリズムセットをサポートしているため、ベースバックアップと WAL アーカイブの圧縮アルゴリズムを個別に変更できます。

- ベースバックアップ none、Snappy、gzip、bzip2
- WAL アーカイブ none、Snappy、gzip、bzip2、lz4、xz、zstd

> **警告**
>
> WAL 圧縮を無効にすることは強くお勧めしません。 WAL ファイルを継続的に生成する PGD クラスター AHA および DHA の場合、WAL 圧縮に「none」を選択することは特にお勧めできず、WAL ストレージがいっぱいになるインシデントが発生する可能性があります。

パフォーマンス特性は環境によって異なるため、デフォルトから変更する前に、特定のワークロードに対して使用可能なアルゴリズムをテストして、どれが最も動作するかを判断します。

> **注釈**
>
> 圧縮アルゴリズムの変更は、後続のバックアップにのみ適用されます。既存のバックアップは影響を受けません。

セキュリティ

必要に応じて、 encrypt your cluster's data を実行できます。

このオプションは、バージョン 15 以降の EDB Postgres Advanced Server および EDB Postgres Extended Server で利用できます。

Transparent Data Encryption TDE) トグルを使用して、TDE を有効にします。

> **注釈**
>
> クラスターの作成後、TDE 構成を有効、無効、または変更することはできません。

次のオプションを設定できます。

- **パスフレーズを入力します** 。TDE 対応クラスターを作成するときに、パスフレーズを使用したこのオプション secures the data encryption key created by Hybrid Manager (HM) 。
- HM インスタンスの管理者が以前に *Adding KMS support* を使用していた場合、**Customer Key Management** を選択することもできます。次に、構成されたキーのいずれかを選択して、ハイブリッドマネージャーによって作成されたデータ暗号化キーを保護します。

TDE の詳細については、 Transparent Data Encryption を参照してください。

Creating replica clusters

[**Replica Clusters**] タブは、シングルノードまたは高可用性クラスターを作成または編集するときに使用できます。これを使用して、シングルノードおよび高可用性クラスターの_replica clusters_を作成および管理します。

レプリカクラスターは、通常、プライマリデータベースクラスターの読み取り専用コピーを作成して、読み取りクエリをオフロードし、パフォーマンスを向上させるために使用されます。プライマリクラスターとは別の地理的場所にレプリカクラスターを作成して、ディザスターリカバリーと高可用性を提供できます。

- **レプリカクラスター**は、プライマリクラスターからレプリケートする別の独立したデータベースクラスターです。ディザスターリカバリーまたは別の地理的場所からのトラフィックの処理に適しています。
- **読み取り専用接続**は、クエリをスタンバイレプリカにルーティングする同じクラスター内の専用の接続エンドポイントであり、別のクラスターを作成せずに読み取りワークロードの分散を有効にします。

代わりに HA クラスターの読み取り専用接続を構成するには、*Cluster setup* を参照してください。

1. [**クラスター情報**] タブで [**単一ノード**] または [**高可用性**] を選択した状態で、クラスターに名前とパスワードを割り当てた後、[**レプリカクラスター**] タブを選択します。

2. **ノード設定** タブを選択し、必要な情報を入力します。
3. **データベースパラメーター構成** タブを選択し、必要に応じてパラメーターを変更します。
4. *Backup and security settings* タブを選択し、必要に応じてパラメーター設定を変更します。
5. **Add Replica Cluster** を選択します。

新しいレプリカクラスターは、[**レプリカクラスター**] タブにリストされます。レプリカクラスターを展開して、レプリカクラスター設定を構成します。プライマリクラスターとは別の地理的場所にレプリカクラスターを作成して、ディザスターリカバリーと高可用性を提供できます。

必要な数のレプリカクラスターを作成します。

レプリカクラスターの管理

レプリカクラスターは編集および削除できます。

- **Edit** を選択して、レプリカクラスターを調整します。
- レプリカクラスターを削除するには、 **Delete** を選択します。
- レプリカクラスターを追加するには、 **Add Replica Cluster** を選択します。

Creating data groups

[**Data Groups**] タブは、分散高可用性または高度な高可用性クラスターを作成または編集するときに使用できます。それを使用して、高度な分散高可用性クラスターの**データグループ**を作成および管理します。データグループは、通常、グループ内でデータを複製し、アプリケーションクエリに応答するように構成されたデータノードのリージョンに近いグループです。分散高可用性クラスターは、相互に複製する 1 つ以上のデータグループで構成でき、地理分散クラスターが有効になります。

データグループの作成

1. **Cluster Info** タブで **Advanced High Availability** または **Distributed High Availability** を選択した状態で、 **Data Groups** タブを選択します。
2. ***Node Settings*** タブを選択し、必要な情報を入力します。
3. ***DB Configuration*** タブを選択し、必要に応じてパラメーターを変更します。
4. **Create Data Group** を選択します。

新しいデータグループは [**Data Groups**] タブにリストされます。クラスターアーキテクチャ図を展開して、データグループの概要を表示します。

別のデータグループを追加するには、 **Add Data Group** を選択します。

> **注釈**
>
> 2 つのデータグループを作成する場合、監視グループが自動的に追加されます。データグループと同様に監視グループを編集できますが、オプションはより制限されています。グループの名前、展開場所、データベースタイプ、バージョン、イメージ、インスタンスサイズを設定できます。

ノード設定

[**ノード設定**] タブを使用して、データグループを設定します。

ノード — グループ内のノードの構成を選択します。

推奨される 3 データノード構成または 2 データノード構成を選択できます。高度な高可用性クラスターの場合、1 ノード構成を作成できます。ノードアーキテクチャの詳細については、*Distribution high availability* を参照してください。

展開の場所 —展開するデータグループの場所を選択します。リストの最初の場所は、 EDB Postgres AI サービスをホストする Kubernetes クラスター内にあります。マルチ DC 環境でハイブリッドマネージャー HM が構成されている場合、データおよび監視グループごとに異なる展開場所を選択して、複数のデータセンターにデータベースクラスターを分散できます。

データベースタイプ — 使用する Postgres のタイプを選択します。

1. **Postgres Type** フィールドで、使用する Postgres のタイプを選択します。

   - EDB Postgres Advanced Server は、EDB の Oracle 互換データベース製品です。EDB Postgres Advanced Server は、すべてのクラスタータイプと互換性があります EDB Postgres Extended Server は、高度な論理レプリケーションを使用する EDB の PostgreSQL 互換データベース製品です。
   - PostgreSQL は、オープンソースのオブジェクトリレーショナルデータベース管理システムです。PostgreSQL は、シングルノードおよび高可用性クラスタータイプと互換性があります。

2. **Postgres Version** リストで、使用する Postgres のバージョンを選択します。

3. HM は、画像ライブラリに保存されている各 Postgres データベースの複数の画像をサポートしています。各イメージは、さまざまな拡張子を含むデータベースの構成です。クラスターに使用するイメージを選択します。詳細については、**アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です** を参照してください。通常、名前の最後に`-full` のない画像には拡張子はありません。名前の最後に`-full` を付けると、画像には PostGIS を除く使用可能なすべての拡張子が含まれます。

インスタンスサイズ —クラスターの CPU の数とメモリの量を選択します。選択できる CPU の数とメモリの量は、Kubernetes クラスターで使用可能なリソースによって異なります。

ストレージ

次のストレージ設定を指定できます。

先行書き込みログ用に別のストレージボリュームを使用します —デフォルトでは、HM はデータベースストレージとログ先行書き込み WAL ファイルに別のストレージボリュームを使用します。このデフォルトでは、最高のパフォーマンスと信頼性が得られます。チェックボックスをオフにして、WAL ファイルをデータベースストレージと同じボリュームに保存します。

データベースストレージ - **データベースストレージ**の下で、データベースストレージクラスとデータベースストレージボリュームのサイズを GiB 単位で選択できます。使用可能なストレージクラスは、基になるインフラストラクチャに設定した内容によって異なります。たとえば、EKS 展開の場合は gp2 および gp3、または RHOS 展開の場合は ssd-csi および standard-csi など。

Write-Ahead Logs Storage — **Write-Ahead Logs Storage** の下で、WAL ストレージクラスと WAL ストレージボリュームのサイズを GiB 単位で選択できます。使用可能なストレージクラスは、基になるインフラ

ストラクチャに設定した内容によって異なります。たとえば、EKS 展開の場合は gp2 および gp3、または RHOS 展開の場合は ssd-csi および standard-csi など。

ネットワーキング

Connectivity Type では、プライベートネットワークデフォルトまたはパブリックネットワークを使用するかどうかを選択します。

- **プライベート**ネットワークでは、プライベートネットワーク内の IP アドレスのみがクラスターに接続できます。
- **パブリック**ネットワークとは、任意のクライアントがインターネットを介してクラスターのパブリック IP アドレスに接続できることを意味します。

接続読み取り専用

高度な高可用性 および **分散高可用性**クラスターの場合、データグループごとに読み取り専用接続を有効にして、現在書き込みリーダーとして指定されていないノードに読み取り専用操作をルーティングし、負荷を削減できます。書き込みリーダーを強化し、全体的なパフォーマンスを向上させます。

読み取り専用ワークロード —トグルを選択して、クラスターの専用の読み取り専用エンドポイントを有効にします。

有効にした場合、次の設定を構成します。

読み取り専用ワークロードネットワーク —読み取り専用エンドポイントにプライベートネットワークデフォルトまたはパブリックネットワークを使用するかどうかを選択します。

- **プライベート**ネットワークでは、プライベートネットワーク内の IP アドレスのみが読み取り専用エンドポイントを介して接続できます。
- **パブリック**ネットワーキングとは、任意のクライアントがインターネットを介して読み取り専用エンドポイントのパブリック IP アドレスに接続できることを意味します。

パブリック を選択した場合、オプションで**許可リストを使用してネットワークアクセスを制限し**、読み取り専用エンドポイントを介して接続を許可される IP レンジを指定できます。

有効にすると、各データグループの読み取り専用接続文字列が、クラスターの詳細の **Connect** タブで利用できます。

バックアップ

バックアップ セクションの **保持期間**コントロールを使用して、デフォルトのデータベースバックアップの保持期間 30 日を変更します。保持期間は日、週、または月単位で構成できます。1〜180 日、1〜25 週間、または 1〜6 か月の範囲で保持期間を設定します。

セキュリティ

このセクションは、データベースタイプとして **EDB Postgres Advanced Server** または **EDB Postgres Extended Server** を選択し、バージョンとイメージを選択すると表示されます。ここでの変更はすべてのデータグループに適用されます。

透過的データ暗号化 TDE を有効にして、独自の暗号化キーを使用します。このオプションは、バージョン 15 以降の EDB Postgres Advanced Server および EDB Postgres Extended Server で利用できます。

Enable TDE チェックボックスを選択して **TDE** を有効にします。

TDE サポートの詳細については、Transparent Data Encryption を参照してください。

データベースパラメーター構成

DB Configuration タブを選択して、データベースのパラメーターを構成します。

Parameters セクションでは、必要に応じてデータベース構成パラメーターの値を更新できます。

[**DB Configuration**] タブにリストされているデータベースパラメーターは、グランドユニファイド構成 GUC 変数とも呼ばれます。詳細については、What Is a GUC Variable? を参照してください。

パラメーターのリストは、クラスターを作成したときに [**クラスター設定**] タブで選択したデータベースのタイプに基づいて入力されます。データベースタイプのパラメーターの詳細については、

- PostgreSQL パラメーターについては、Postgres ドキュメントの

Setting Parameters および Server Configuration を参照してください。

- EDB Postgres Advanced Server の場合、

Summary of configuration parameters および Configuration parameters を参照してください。

- パラメーターが受け入れる値の種類の詳細については、

Parameter Names and Values を参照してください。これらのタイプは、 PostgreSQL と EDB Postgres Advanced Server パラメーターの両方に適用されます。

> **注釈**
>
> クラスターの作成が完了した後、データベースまたはサーバーパラメーターを変更できます。一部のパラメーターを変更するには、再起動が必要です。

パラメーターの変更

1. これらの方法を使用して、変更するパラメーターを見つけます。
   - 検索フィールドを使用して特定のパラメーターを検索します。
   - **Show Only** リストのオプションを選択して、リストされたパラメーターをフィルターします。
     - **Custom Values** —現在のセッションで、または以前のセッションで変更されたデフォルト値から変更されたパラメーターのみを表示します
     - **現在編集中の値** —現在の編集セッション中に変更されたパラメーターのみを表示します。
2. パラメーター値フィールドに新しい値を入力します。エントリーには、リテラル値、または他のパラメーターに基づいて値を計算する *Formulas for parameter values* を指定できます。

> **警告**
>
> 黄色の感嘆符アイコンで示されるパラメーターは、変更を保存するときに再起動をトリガーします。再起動により、開いている接続がすべて終了され、完了までに数分かかります。新しい接続が確立されるまでにさらに数秒かかります。このプロセス中に、データベースに接続することはできません。

データグループの管理

データグループを複製、編集、および削除できます。

- グループを複製するには、 **Duplicate** を選択します。次に、新しいグループを編集して元のグループと区別します。マルチ DC 環境では、複製したデータグループに別の展開場所を選択できます。
- **Edit** を選択して、データグループを調整します。
- データグループを削除するには、 **Delete** を選択します。削除の確認を求められます。
- データグループを追加するには、 **Add Data Group** を選択します。

Creating a cluster from a template

プロジェクトにある場合は、テンプレートを使用してデータベースクラスターを作成できます。テンプレートを使用すると、時間を節約し、環境間の一貫性が保証されます。

テンプレートを作成するには、 *Create cluster templates* を参照してください。

スタートポイント

Choose Your Path ページまたは **Cluster Templates** ページのテンプレートからクラスターの作成を開始できます。

[パスの選択] ページから

Estate ページまたはプロジェクトページから **Choose Your Path** ページにアクセスできます。

エステート ページから

1. **Quick Actions > Create New Database Cluster** を選択します。

複数のプロジェクトがある場合、プロジェクトを選択するように求められます。

2. プロンプトが表示されたら、データベースを作成するプロジェクトを選択します。

プロジェクトを選択した後、またはプロジェクトが 1 つだけの場合、 **Choose Your Path** ページが開きます。

プロジェクトページから、これらのアプローチのいずれかを使用できます。プロジェクトページに移動するには、**Projects** ページからプロジェクトを選択します。

- プロジェクトページで、 **Create New > Database Cluster** を選択します。
- プロジェクトページで、**Clusters > Postgres Cluster** を選択します。[**クラスター**] ページで、[**Create New > Database Cluster**] を選択します。

Choose Your Path ページが開きます。

[Choose Your Path] ページのテンプレートを使用する

1. **Choose Your Path** ページで、 **Use Template** を選択します。
2. テンプレートを選択します。使用可能なテンプレートのリストから、ニーズに合ったものを選択します。
3. **Use Template** を選択します。

クラスター作成モードに入ります。 クラスターのカスタマイズと作成 を参照してください。

[クラスターテンプレート] ページから開始します

1. プロジェクトページの左側のナビゲーションで、 **Cluster > Cluster Templates** を選択します。
2. クラスターのベースとして使用するテンプレートの横にある **Use** アイコンを選択します。

クラスター作成モードに入ります。 クラスターのカスタマイズと作成 を参照してください。

クラスターのカスタマイズと作成

テンプレートを選択したら、作成しているクラスターの設定を確認および変更できます。詳細については、以下を参照してください。

- *Cluster types to create*
- *Cluster setup*
- *Database configuration parameters*
- *Backup and security settings*
- *Creating replica clusters* シングルノードおよび高可用性クラスターの場合
- *Creating data groups* 高度なまたは分散高可用性クラスター用

作業する別のテンプレートを選択することもできます。 **Cluster Templates** タブを選択して、使用可能なテンプレートのリストを表示します。

タブを使用してクラスターの構成が終了したら、[**Create Cluster**] を選択します。設定が正しくないまたは欠落しているタブには、赤いインジケーターが表示されます。エラーを修正し、**Create Cluster** を再度選択します。

有効な設定で **Create Cluster** を選択すると、クラスターはプロビジョニングを開始し、クラスターのリストに戻ります。ポップアップはクラスターの作成を確認し、新しいクラスターがクラスターのリストに追加されます。クラスターはバックグラウンドで作成されます。クラスターのリストで進行状況を確認できます。

クラスターの作成の進行中にクラスターの詳細を表示しようとすると、概要、バックアップ、ログ、プロパティのみを表示できます。

Creating a cluster programmatically

Postgres クラスターの作成を自動化することにより、再現可能なビルドを合理化できます。

Bash スクリプトとハイブリッドマネージャー API を使用して、再現可能なビルドの Postgres クラスターの作成を自動化します。すべての構成をソース管理にチェックして、変更を追跡し、ドキュメントを維持します。

マシンユーザーを作成する

1. HM コンソールの右上隅にあるドロップダウンメニューに移動します。
2. ドロップダウンメニューから **User Management** を選択します。
3. 右上隅にある **Add new user** を選択します。
4. **Add new user** が表示されるポップアップウィンドウに、名前を入力し、チェックボックス **Create access key** をチェックします。
5. フィールド **有効期限** に入力します。
6. ポップアップウィンドウで、[**Add user**] を選択します。
7. ポップアップウィンドウからアクセスキーをコピーします。
8. **キーは安全に保存されました** を選択します。

マシンにユーザー権限を付与します

> **注釈**
>
> マシンユーザー権限を正しく設定するには、左側のナビゲーションバーから Postgres クラスターを作成するターゲットプロジェクトを選択します。

2. 左側のナビゲーションバーで、 **Users** を選択します。
3. 検索バーで、作成したマシンユーザーを探します。
4. **Machine user** の右側にある鉛筆編集アイコンを選択します。
5. **役割の割り当て** を選択します。
6. **プロジェクトのロールの割り当て** の下で、チェックボックス**プロジェクトの所有者** をチェックします。
7. **Submit** を選択します。

env.sh ファイルを作成する

プロジェクト ID を `env.sh` ファイルと API ホストに追加することにより、シェルスクリプトの記述を簡素化できます。

コピーしたアクセスキーを `env.sh` という名前のファイルに追加します。

```
export HCP_ACCESS_KEY="baak_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
export HCP_API_ACCESS_KEY=${HCP_ACCESS_KEY}
export HCP_PROJECT_ID=prj_xxxxxxxxxxxxxx
export HCP_API_HOST="https://portal-uat-eks.aws.hcp.enterprisedb.network"
```

AIDB および PGFS を使用して Docker イメージを見つけます

場所とその場所で使用可能な Docker イメージのリストを取得するには、01-get-locations.sh というスクリプトを作成して実行します。次のコンテンツが表示されます。

```
# !/bin/bash

set -e
set -u
set -o pipefail

#  list locations

source env.sh

curl \
  --insecure \
  --header "Content-Type: Application/JSON" \
  --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
  --request GET \
  ${HCP_API_HOST}/api/v1/projects/${HCP_PROJECT_ID}/locations \
  | jq
```

次のスクリプトを実行します。

```
./01-get-locations.sh > locations.json
```

locations.json を調べて、pgExtensions 、aidb 、および pgfs の下にリストされているイメージを探します。

Postgres クラスターを作成する

前の手順で取得したイメージ名を含む create-psr-cluster.json という名前のファイルを作成します。

```
{
  "projectId": "${HCP_PROJECT_ID}",
  "psr": {
    "clusterName": "small-ai-api-001",
    "password": "111111111111",
    "locationId": "managed-devspatcher",
    "clusterData": {
```

(次のページに続く)

(前のページからの続き)

```
      "instances": 1,
      "resourceRequest": {
        "memory": "2Gi",
        "cpu": "2"
      },
      "storageConfiguration": {
        "primaryStorage": {
          "size": "4",
          "storageClass": "hcp-aws-gp3"
        },
        "walStorage": {
          "size": "4",
          "storageClass": "hcp-aws-gp3"
        }
      },
      "image": {
        "url": "upmdev.azurecr.io/postgresql:17.6-2510160831",
        "digest":
↪"sha256:6bda61e6d75c3b9d6a886c547d789c46b88ed807ae28d8d06fa1510a0cc5afae"
      },
      "networkAccessType": "NetworkAccessTypePublic"
    }
  }
}
```

create-psr-cluster.json を使用し、クラスターを作成するように 02-create-psr-cluster.sh というシェルスクリプトを作成します。

```
# !/bin/bash

set -e
set -o pipefail
set -u

source env.sh

cat create-psr-cluster.json | envsubst > /tmp/create-psr-cluster.json

curl \
  --insecure \
  --header "Content-Type: Application/JSON" \
  --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
  --data-binary "@/tmp/create-psr-cluster.json" \
  --request POST \
  ${HCP_API_HOST}/api/v1/projects/${HCP_PROJECT_ID}/clusters
```

シェルスクリプトを実行します。

```
./02-create-psr-cluster.sh | tee ./psr-cluster.json
```

HM コンソールでクラスター作成の進行状況を確認できます。

新しいクラスターに AIDB および PGFS 拡張機能を作成します

AIDB および PGFS の機能を、プログラムで作成したクラスターに拡張機能として追加できます。

PostgreSQL 認証資格情報を含めるように env.sh を構成する

以下を env.sh ファイルに追加します。例

```
export PGPASSWORD=your_postgresql_password
export PGUSER=edb_admin
export PGHOST=your-postgresql-hostname
export PGPORT=5432
export PGDATABASE=edb_admin
export PGSSLMODE=require
```

クラスター拡張機能を作成する

次の内容を使用して 03-create-ai-extensions.sh という名前の bash ファイルを作成します。

```
# !/bin/bash

set -e
set -o pipefail
set -u

source env.sh

psql -X <<EOF
create extension aidb cascade;
create extension pgfs cascade;
EOF
```

スクリプトを実行します。

```
$ chmod +x 03-create-ai-extensions.sh
$ ./03-create-ai-extensions.sh
```

クラスター検証

HM コンソールを介してクラスターの作成を確認した後、API を介して新しいクラスターを取得して、設定を確認できます。JSON 形式でクラスターを取得することは、必要な JSON フィールドを指定するのに

も役立ち、自動クラスター作成のブループリントとして機能し、各設定フィールドを個別に検索することを回避できます。

> **注釈**
>
> Create-cluster JSON と Get-cluster JSON は、異なる情報を含めることができる 2 つの異なる機能です。

```
# !/bin/bash

set -e
set -u
set -o pipefail

source env.sh

HCP_PG_CLUSTER_ID=p-fj3ki6e0x0

curl \
  --insecure \
  --header "Content-Type: Application/JSON" \
  --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
  --request GET \
  ${HCP_API_HOST}/api/v1/projects/${HCP_PROJECT_ID}/clusters/${HCP_PG_CLUSTER_ID} \
  | jq
```

HCP_PG_CLUSTER_ID はクラスター ID を指定します。

クラスターのコンテンツを JSON ファイルに保存します。

```
./get-existing-cluster.sh > existing-cluster.json
```

> **注釈**
>
> クラスター作成の必須フィールド、命名規則、および JSON 構造の詳細については、 *Using the HM API* を確認してください。 !!!

Managing clusters

いくつかの管理タスクは、ハイブリッドマネージャー HM のクラスターに関連付けられています。

クラスター作成タスク

クラスター作成プロセスの一部として、次のことを行うことができます。

- 新しいテンプレートのベースとして使用できる *Create a cluster template*
- *Create a formula* 特定の構成パラメーターの値として使用する。

既存のクラスターでの作業

create a cluster 後、クラスター名の右側にある 3 点メニューを使用して、次のことを行うことができます。

- クラスターを編集する 構成の大部分を変更します。
- クラスターパスワードを変更します。
- *Delete a cluster* 。
- クラスターをアップグレードします。マイナーアップグレードまたはメジャーアップグレードを実行できます。
- *Restore a deleted cluster* 。

メニューには、現在のプロジェクトユーザーが使用可能なオプションのみが含まれます。

Postgres クラスターの実行状況を分析する

クラスターの実行中に、Postgres クラスターが実行されている Kubernetes インフラストラクチャのステータスを確認する必要がある場合があります。これは、*tracing the cluster back to its Kubernetes resources* で行うことができます。たとえば、Grafana を使用して、リソースがどのように使用されているかをより深く理解できます。

Create a database cluster template

将来のクラスターのベースとして使用するクラスターテンプレートを作成できます。テンプレートの作成は、クラスターの作成に似ています。すべて同じ情報を指定する必要があります。

スタートポイント

開始点は、テンプレートを作成するプロジェクトのランディングページです。そこに到達するには、**プロジェクト** ページからプロジェクトを選択します。

プロジェクトページの左側のナビゲーションで、 **Clusters > Cluster Templates** を選択します。

Cluster Templates ページには、プロジェクト内のテンプレートがリストされます。

Create Template > Create Database Cluster Template を選択します。

この点から、クラスター構成の指定は、クラスターを作成する場合と同じです。**Next** を使用してタブを順番にステップ実行するか、作業するタブを選択できます。詳細については、以下を参照してください。

- *Cluster types to create*
- *Cluster setup*
- *Database configuration parameters*
- *Backup and security settings*
- *Creating data groups* 高度なまたは分散高可用性クラスター用

クラスターテンプレートの構成が終了したら、[**Create Template**] を選択します。

新しいテンプレートがテンプレートのリストに追加されます。

テンプレートの削除

テンプレートは、不要になった場合は **クラスターテンプレート**ページから削除できます。

> **警告**
>
> 削除されたテンプレートを復元または復元することはできません。

1. テンプレートを削除するプロジェクトのページの、左側のナビゲーションで、**クラスター > クラスターテンプレート** を選択します。
2. 削除するテンプレートの横にある **Delete** アイコンを選択します。
3. 確認ボックスに `delete template` と入力し、 **Delete** を選択します。

テンプレートの使用については、 *Creating a cluster from a template* を参照してください。

Deleting clusters

クラスターは、 **Postgres クラスター**ページおよびクラスターの詳細ビューから削除できます。

- プロジェクトページから、 **Clusters > Postgres Clusters** を選択します。削除するクラスターの横にある 3 点メニューから、 **Delete Cluster** を選択します。
- プロジェクトページから、 **Clusters > Postgres Clusters** を選択します。削除するクラスターを選択します。クラスターの詳細ビューから、**Quick Actions > Delete Cluster** を選択します。

削除の確認をするプロンプトが表示されたら、 **Delete** を選択します。

Editing a cluster

Postgres クラスターページまたはクラスターの詳細ビューからクラスターの編集を開始できます。

- **Postgres Clusters** ページで、編集するクラスターの横にある **Edit** を選択します。
- クラスターの詳細ビューを表示するには、 **Postgres Clusters** ページで、クラスター名前を選択します。 [**Cluster Details**] ページで、[**Quick Actions > Edit Cluster**] を選択します。

> **注釈**
>
> クラスターの作成が完了した後、データベースまたはサーバーパラメーターを変更できます。一部のパラメーターを変更するには、再起動が必要です。

クラスターの編集

Edit Cluster ページは、 **Create Cluster** ページに似ています。作業するタブを選択するか、 **Next** を使用してタブを順番に実行します。

- ***Cluster Info***
- ***Cluster Settings***
- ***DB Configuration*** オプショナル

- ***Additional Settings*** オプショナル
- ***Data Groups*** 分散高可用性クラスター用

> **注釈**
>
> クラスターを編集するときに、シングルノードまたは高可用性クラスターと、高度なまたは分散高可用性クラスターの間で切り替えることはできません

編集の保存と確認

クラスター構成の変更が完了したら、[**Save**] を選択します。

構成が有効な場合、クラスターに行う変更のリストが表示されます。変更によりサーバーが再起動される可能性があることを示す警告が表示される場合があります。**confirm Changes** を選択して変更をクラスターに適用し、元のページに戻ります。

> **注釈**
>
> ページの上部にある**戻る**矢印を選択すると、変更が破棄され、元のページに戻ります。

Editing a cluster programmatically

ハイブリッドマネージャー API を使用して、クラスター構成をプログラムで更新し、Postgres クラスターへの反復および監査可能な変更を有効にします。API は部分更新 PATCH モデルを使用するため、要求本文に明示的に含まれるフィールドのみが変更されます。

開始する前に、マシンユーザーと `env.sh` ファイルを構成します。初期設定の手順については、*Creating a cluster programmatically* を参照してください。

クラスター ID を `env.sh` に追加します。

```
export HCP_PG_CLUSTER_ID=p-fj3ki6e0x0
```

現在のクラスター構成の取得

クラスターにパッチを適用する前に、現在の状態を取得して、既存のフィールド値と `replicaPgId`、`groupId` などの識別子を確認します。これらは *PSR* **レプリカクラスターの管理** および *Managing PGD and AHA data groups* を管理する際に必要です。

`get-cluster.sh` というスクリプトを作成して実行します。

```
# !/bin/bash

set -e
set -o pipefail
set -u

```

(次のページに続く)

(前のページからの続き)

```
source env.sh

curl \
  --insecure \
  --header "Content-Type: application/json" \
  --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
  --request GET \
  ${HCP_API_HOST}/api/v1/projects/${HCP_PROJECT_ID}/clusters/${HCP_PG_CLUSTER_ID} \
  | jq
```

PSR クラスターにパッチを適用します

PSR クラスターを更新するには、変更するフィールドのみを含む JSON ファイルを作成します。次の例では、クラスターを 3 つのインスタンスにスケーリングし、そのリソース割り当てを更新します。

patch-psr-cluster.json という名前のファイルを作成します。

```
{
  "psr": {
    "instances": 3,
    "resourceRequest": {
      "memory": "8Gi",
      "cpu": "4"
    }
  }
}
```

patch-cluster.sh という名前のスクリプトを作成します。

```
# !/bin/bash

set -e
set -o pipefail
set -u

source env.sh

curl \
  --insecure \
  --header "Content-Type: application/json" \
  --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
  --data-binary "@${1}" \
  --request PATCH \
  ${HCP_API_HOST}/api/v1/projects/${HCP_PROJECT_ID}/clusters/${HCP_PG_CLUSTER_ID}
```

JSON ファイルを引数として渡して、スクリプトを実行します。

```
./patch-cluster.sh patch-psr-cluster.json
```

要求が成功すると、空の本文を含む HTTP 200 が返されます。変更は、Hybrid Manager コンソールで、または GET 要求でクラスターを取得することにより確認できます。

PSR レプリカクラスターの管理

PSR クラスターは、リモートの場所に展開された *read replica clusters* をサポートします。PSR パッチ要求に `replicaClusters` フィールドを含めると、API は宣言リストモデルを適用します。提供したリストは、その PSR クラスターの完全で権限のあるレプリカクラスターのセットになります。

次の例では、単一のパッチ要求で 3 つの変更を加えます。

- `managed-us-east-1` の既存のレプリカクラスターを保持し、構成を変更しません。
- `managed-eu-west-1` の既存のレプリカクラスターを編集し、2 つのインスタンスにスケーリングします。
- `managed-ap-southeast-1` に新しいレプリカクラスターを追加します。

`patch-psr-replicas.json` という名前のファイルを作成します。`replicaPgId` 値をクラスターの値に置き換えます。

```
{
  "psr": {
    "replicaClusters": [
      {
        "replicaPgId": "p-fj3ki6e0x0-b"
      },
      {
        "replicaPgId": "p-fj3ki6e0x0-c",
        "locationId": "managed-eu-west-1",
        "instances": 2
      },
      {
        "locationId": "managed-ap-southeast-1",
        "instances": 1,
        "resourceRequest": {
          "memory": "4Gi",
          "cpu": "2"
        },
        "storageConfiguration": {
          "primaryStorage": {
            "size": "10",
            "storageClass": "hcp-aws-gp3"
          }
        },
        "image": {
```

(次のページに続く)

(前のページからの続き)

```
          "url": "your-registry.example.com/postgresql:17.6-xxxxxxxxxx",
          "digest":
↪"sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
        },
        "backupRetentionPeriod": "30d",
        "backupSchedule": "0 0 ",
        "networkAccessType": "NetworkAccessTypePrivate"
      }
    ]
  }
}
```

PSR クラスターにパッチを適用します から `patch-cluster.sh` を実行し、レプリカファイルを引数として渡します。

```
./patch-cluster.sh patch-psr-replicas.json
```

注釈

構成を変更せずに既存のレプリカを保持する場合、その `replicaPgId` のみを提供するだけで十分です。他のすべてのフィールドは変更されません。

分散高可用性 PGD および高度な高可用性 AHA クラスターは、*2 つのデータグループと 1 つの監視グループを含む EDB Postgres 分散 PGD クラスターが正常ではありません* を使用して場所を超えてノードを編成します。PGD または AHA パッチ要求に `groups` フィールドを含める場合、API は PSR レプリカクラスターに使用されるのと同じ宣言リストモデルを適用します。

次の例では、PGD クラスターにパッチを適用します。更新されたインスタンス数を持つ 1 つの既存のデータグループを保持し、2 番目の場所に新しいデータグループを追加し、未参照のグループを削除します。

`patch-pgd-groups.json` という名前のファイルを作成します。 `groupId` 値をクラスターのものに置き換えます。

```
{
  "pgd": {
    "groups": [
      {
        "groupId": "p-sdl5klb2hu2-a",
        "locationId": "managed-us-east-1",
        "groupName": "us-east-data-group",
        "dataGroup": {
          "dataInstances": 3,
          "resourceRequest": {
            "memory": "8Gi",
            "cpu": "4"
```

(次のページに続く)

(前のページからの続き)

```
          }
        }
      },
      {
        "locationId": "managed-eu-west-1",
        "groupName": "eu-west-data-group",
        "dataGroup": {
          "dataInstances": 2,
          "resourceRequest": {
            "memory": "4Gi",
            "cpu": "2"
          },
          "storageConfiguration": {
            "primaryStorage": {
              "size": "20",
              "storageClass": "hcp-aws-gp3"
            }
          },
          "image": {
            "url": "your-registry.example.com/postgresql:17.6-xxxxxxxxxx",
            "digest":
↪"sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
          },
          "backupRetentionPeriod": "30d",
          "backupSchedule": "0 0 ",
          "networkAccessType": "NetworkAccessTypePrivate"
        }
      }
    ]
  }
}
```

PSR クラスターにパッチを適用します から patch-cluster.sh を実行し、グループファイルを引数として渡します。

```
./patch-cluster.sh patch-pgd-groups.json
```

- 構成を変更せずに既存の PGD データグループを保持する場合、その groupId のみを提供するだけで十分です。他のすべてのフィールドは変更されません。
- PGD クラスターに 2 つのデータグループがある場合、監視グループはプラットフォームによって自動的に管理され、Raft クォーラムを維持します。詳細については、*2* つのデータグループと *1* つの監視グループを含む *EDB Postgres* 分散 *PGD* クラスターが正常ではありません を参照してください。 !!!

Restoring clusters

削除されたクラスターをバックアップから復元できます。バックアップは、復元するクラスターと同じプロジェクトにある必要があります。復元プロセスは、技術的には、削除されたクラスターと同じ構成で新しいクラスターを作成します。これは実際には新しいクラスターであるため、新しい名前と新しいパスワードを指定する必要があります。

復元されたクラスターは、削除されたクラスターと同じプロジェクトに作成されます。バックアップが利用可能な限り、削除されたクラスターを復元できます。バックアップは、クラスターごとに設定された保持期間の間利用可能なままです。保持期間が経過すると、バックアップは削除されます。

バックアップからのクラスターの復元

1. **クラスター> Postgres クラスター** を選択します。
2. **Deleted** タブを選択します。
3. 復元する削除されたクラスターの横にある **Restore** アイコンを選択します。
4. **Restore Cluster** ダイアログボックスで、復元するバックアップを選択し、 **Restore** を選択します。

Restore Cluster ページが開きます。このページは、 **Create Cluster** ページに似ています。クラスター名前、クラスタータイプ、および復元するために選択したバックアップが表示されます。必要に応じて、クラスター名とクラスタータイプを変更できます。

5. 新しいパスワードを入力します。
6. [**クラスター設定**] タブで、オプションでクラスター設定を変更できます。 **Restore Type** の下で、実行する復元のタイプを選択します。
    - **完全な復元** - データベースの最後の完全なバックアップからクラスター全体を復元します。このオプションを選択すると、その下のリストから復元するバックアップを選択できます。
    - **ポイントインタイムリストア** - クラスターを特定のポイントインタイムに復元します。バックアップの保持期間内の日付と時刻を選択します。範囲は日付と時刻ピッカーの下に表示されます。
7. 必要に応じて他のタブを使用します。詳細については、*Cluster creation settings* を参照してください。
8. **Restore Cluster** を選択します。

Postgres Clusters ページが開き、復元操作の進行状況を確認できます。復元されたクラスターは、元のクラスターと同じプロジェクトにあります。そのクラスターと同じ構成ですが、新しい名前とパスワードがあります。

Tracing clusters back to their Kubernetes resources

Hybrid ManagerHM で Postgres クラスターを管理しているときに、Postgres クラスターが実行されている Kubernetes インフラストラクチャのステータスを確認する必要がある場合があります。これらは Kubernetes ワーカーノードです。たとえば、HM インターフェイスで問題が表示されているため、またはリソースが不足しているためにノードに障害が発生しているかどうかを確認することができます。

クラスターを Kubernetes リソースまでトレースするのに役立つ情報を取得するには、いくつかの方法があります。

- *HM を使用してクラスターをその名前空間に照合する* より

- *Grafana* ハードウェア使用率ダッシュボード より
- *Loki* と *Fluent Bit* を使用する
- あなたが管理者の場合 *kubectl* を使用して基になるクラスターリソースを確認する を使用する

分散高可用性クラスターをトレースバックする場合、高可用性またはシングルノードクラスターのトレースとのいくつかの違いに注意する必要があります。分散高可用性クラスターのトレースバック を参照

HM を使用してクラスターをその名前空間に照合する

HM を使用して作成された Postgres クラスターをその基になる Kubernetes リソースまでトレースバックする場合、最初に開始する場所は HM のクラスターの**概要** ページです。そのページに移動するには、 **Estate** ページで、クラスターを選択します。そこで、ブラウザーの URL の最後を見て、クラスターに関連付けられている Kubernetes 名前空間を見つけます。

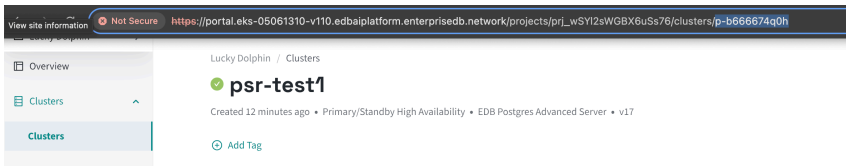


図 3: Browse cluster

この例では、クラスター psr-test1 は名前空間 `p-b666674Q0h` に関連付けられます。クラスターをそのリソースまでさらにトレースするには、関連する名前空間がパズルの重要な最初のピースです。それを使用すると、さらに多くのことを見つけることができます。

探索しているクラスターの名前空間を取得したら、次の目的はビルトインの Grafana ダッシュボードです。

Grafana ハードウェア使用率ダッシュボード

Grafana ダッシュボードにアクセスするには、HM ランチパッドを使用します。プロフィールメニューから、**Launchpad** を選択します。ラウンチパッドの **Grafana** タイルで、**View Service** を選択します。

Grafana のランディングページの **Search** フィールドに、`Hardware Utilization` と入力します。**ハードウェア使用率**ダッシュボードでは、基になる Kubernetes ワーカーノードと、一部で実行されている Postgres ワークロードを含む Kubernetes ポッドを表示できます。

ポッド

Postgres Pod Status セクションでは、調査している Postgres クラスターに関連付けられた名前空間を使用して、Postgres クラスターが関連付けられている名前空間と同じ指定で名前が始まるポッドを特定します。ワークロードがポッド間で分割されるため、これらのポッド名には`-1` 、`-2` 、`-3` などのサフィックスが付けられます。 Postgres Nodes テーブルから、State 列にポッドの状態を確認し、Node 列にポッドが実行されているノードを確認することもできます。

リソースの可用性

ポッドと基になる Kubernetes ワーカーノードを照合した後、**リソースの可用性**の下のノード名を使用して、これらのノードで使用可能なリソースに関する詳細情報を取得できます。

- **利用可能なコア**ワークロードが必要な分を使用した後に残っている利用可能なコアの数。

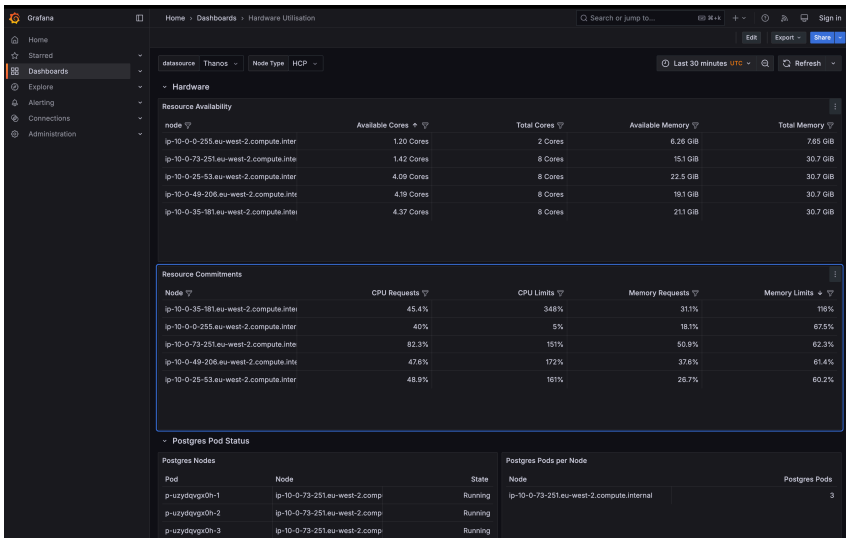


図 4: Hardware Utilization dashboard

- **合計コア**ワークロードから使用される前にノードが必要とするコアの合計数。ワーカーノードには複数のポッドが存在する場合があります。
- **利用可能なメモリ**ワークロードが必要な分を利用した後に利用可能なメモリの量。
- **合計メモリ**何も使用していないワークロードのないノードの合計メモリ。

リソースのコミットメント

Postgres ワークロードを調べたい Kubernetes ワーカーノードがある場合、ダッシュボードのもう 1 つの役立つセクションは**リソースコミットメント**テーブルです。ここでは、関心のあるポッドの基になるノードを含む、各ノードでどのような要求が実行されているかを確認できます。

- **CPU 要求**プロビジョニングされたワークロードによって要求されたノードの CPU 量、または HM によってプロビジョニングされています。
- **CPU 制限** ノード上のポッドからの CPU 制限の合計。
- **メモリ要求**プロビジョニングされたワークロードによって要求されたノードからのメモリ量、または HM によってプロビジョニングされています。
- **メモリ制限** ノード上のポッドからのメモリ制限の合計。

リクエスト

CPU とメモリ要求は両方とも、その上で実行されているポッドによるノードの要求を満たすことが保証されていますノードがオーバープロビジョニングされていないことを前提としています。これは、各ポッドが一度に 25%を要求する必要があるという意味ではありません。たとえば、1 つ以上がアイドル状態である可能性があり、その時点で必要な CPU ははるかに少なくなります。

ただし、これは、各ポッドが一度に CPU リソースの 25 %を使用する可能性があることを意味します。ノード A でポッド 1、2、および 3 のそれぞれに CPU 要求があり、3 つのポッドのそれぞれがノードで 25%の CPU 要求があるとします。ノード A で他のポッドが CPU 要求を行っていないことを考慮すると、ノードの合計要求要求は 75%であり、ノードが CPU 負荷を処理できることを意味します。

他のクラスターの他のポッドもノード A で要求を行っていた場合、たとえば、別のポッドがノードで 30 の要求を持っていた場合、ノードは 105 でオーバープロビジョニングされます。

リミット

CPU とメモリの制限は両方とも、ポッドが使用できるノードのリソースの CPU とメモリの上限です。ポッド 1、2、および 3 の制限がそれぞれ 50 の CPU であり、すべてノード A であると仮定します。これは、ノード A の CPU 制限が 150 であることを意味します。一度に 100 %を超える CPU を提供できないため、3 つのポッドすべてが 50 %にバーストすると、結果として調整されます。ポッド 1 が 50 の CPU 制限にバーストし、他の 2 つが通常のオペレーションでそれぞれ 25 %のみを使用している場合、ポッド 1 は必要な 50 %を取得します。

メモリの場合、ポッド 1、2、および 3 がノード A でそれぞれ 50%のメモリである同様のシナリオを検討します。3 つすべてが 50%のしきい値までバーストしようとします。ポッド内のコンテナは、Linux カーネルによって killed になる場合があります。

ハードウェア使用率ダッシュボードを使用した潜在的なリソース問題のトラブルシューティング

プロビジョニングを要求した Postgres クラスターがあり、85 %完了した状態で 15 分以上フリーズしているように見えるとします。HM コンソールから、Web ブラウザーの URL の最後のビットを使用して、クラスターの名前に関連付けられた名前空間を見つけることができます。

この場合、名前空間 `p-b666674Q0h` を調べているとします。**ハードウェア使用率**ダッシュボードを使用して、最初にその名前空間に関連付けられているポッド-ここでも `p-b666674Q0h-1` 、`p-b666674Q0h-2` 、および `p-b666674Q0h-3` とします-を見つけ、これらのポッドのステータスを確認できます。1 つ以上のポッドが `Pending` 状態になっている可能性があります。たとえば、 `p-b666674Q0h-1` および `p-b666674Q0h-2` が実行されている間、 `p-b666674Q0h-3` はまだ保留中の場合があります。次に、 **Postgres Pod Status** の下の Postgres ノードテーブルから `p-b666674Q0h-3` が実行されているノードを特定できます。

`p-b666674Q0h-3` がノード `ip-10-0-73-251.eu-west-2.compute.internal` で実行されていると仮定します。 **リソースの可用性**テーブルの `ip-10-0-73-251.eu-west-2.compute.internal` を確認すると、合計 8 コアのうち 0.5 コアが利用可能であり、合計 20.3 のうち 0.8Gi のメモリが利用可能なことがわかります。このことから、ノードには別のポッドのリソースがあまり残されていないと結論付けることができます。

Resource Commitments テーブルの `ip-10-0-73-251.eu-west-2.compute.internal` を確認すると、ノードに 129 の CPU 要求と 106 のメモリ要求があることに気づいたとしましょう。要求が 100%を超えていることを考慮すると、ノードがオーバープロビジョニングされていると結論付けることができます。これは、ポッド `p-b666674Q0h-3` が `Pending` 状態でスタックしている理由を説明しています。より多くのリソースがなければ、この Postgres クラスターは完了できません。HM を使用して、プロビジョニングするクラスターを編集して、必要な CPU とメモリを削減できます。プロビジョニングプロセスは続行されます。クラスターを小さくすることがニーズを満たさない場合は、より多くのデータベースワーカーノードを Kubernetes クラスターに手動で割り当てるか、オートスケーラーの最大ノードを割り当てて、より多くのプロビジョニングを行うように認識する必要があります。

Loki と Fluent Bit

Fluent Bit サービスログで Loki を使用して、ポッドとそのステータスに関する詳細情報を見つけることができます。このビューを見つけるには

1. Grafana のナビゲーションパネルから、 **Explore** を選択します。

2. **Outline** の横にあるメニューから、 **Loki** を選択します。
3. **Label filters** メニューを使用して、左側に **service_name** を選択し、右側に **fluent-bit** を選択します。
4. その名前空間に関連付けられているポッドの Loki の Fluent Bit ログで検索クエリを実行します。さらに分析するクラスターに関連付けられた名前空間をコピーして、**Line contains** フィールドに貼り付け、 **Run Query** を選択します。

検索した名前空間を含むログが収集された時間を確認できます。その下には、実際のログが表示されます。

Pretty JSON をオンにして、検索された名前空間の各インスタンスをログ全体で強調表示して表示します。

検索しているポッドに関連付けられている `Can't find target pod` など、検索している名前空間に関連するエラーメッセージが全体で表示される場合があります。または、`reason:  FailedScheduling` の下に、次のようなメッセージが表示される場合があります。

```
"0/43 nodes are available: 1 node(s) had untolerated taint (CriticalAddonsOnly: true),
 3 node(s) had untolerated taint (edbaiplatform.io/control-plane: true), 39␣
↪Insufficeint cpu, 4 Insuffcient memory. preemtion: 0/43 nodes are available: 39 No␣
↪preemption vitims found for upcoming pod, Preemption is not helpful for scheduling."
```

このメッセージは、ポッドが完了に失敗した正確な理由、つまり CPU とメモリの不足を示します。

分散高可用性クラスターのトレースバック

分散高可用性 DHA クラスターのトレースは、高可用性クラスターまたはシングルノードクラスターのトレースとは少し異なります。これは、DHA クラスターが 1 つ以上のデータグループに分割され、これらの各データグループにはグループごとに 2〜3 つのノードがあるためです。

たとえば、それぞれ 3 つのノードを持つ 2 つのデータグループ A および B を含む DHA クラスターがあるとします。クラスターのメインページの **Monitoring** タブに移動し、そのタブの **Monitoring** セクションに移動すると、2 番目の値をデフォルトの **クラスターレベル**から **ノードに変更できます。レベル** 。値を **ノードレベル**に変更した後、右側の最後のメニューを展開して、クラスターの 6 つのノードをそれぞれ表示できます。データグループ A の 1、2、および 3、およびデータグループ B の 1、2、および 3。

Kubernetes レベルに戻るには、Grafana の *Grafana* **ハードウェア使用率ダッシュボード** に移動します。

次に、 **ハードウェア使用率** ダッシュボードの左下近くにある **Postgres ノード** セクションを確認します。クラスターの **モニタリング**タブで、今表示した 6 つのノードは、`<id-of-cluster/namespace>-a-1-1` 、`<<id-of-cluster/namespace>-a-2-1>` などの形式でそれぞれのポッドで表されます。ポッドの指定は、ポッドが属するデータグループ A または B と、ポッドのどのノードである 1、2、または 3 を呼び出します。

ポッドを指定すると、実行しようとしているノードとポッドの状態の両方を確認できます。ノード ID を使用すると、ダッシュボードの ***Resource Availability*** および ***Resource Commitments*** セクションで、そのノードのリソースの可用性とリソースコミットメントの両方を表示できます。ノードに残っているコアの数、メモリ量などを確認できます。

シングルノードまたは HA クラスターと同様に、ここから一部のリソースの問題をトラブルシューティングできます。または、**Explore** に移動し、 *Loki* と *Fluent Bit* を使用して、関心のあるポッドのログレベルを調べることができます。

kubectl を使用して基になるクラスターリソースを確認する

Kubernetes へのアクセスを持つ管理者は、kubectl を使用して、Postgres クラスターの基になる Kubernetes ワーカーノードを分析できます。

まず、kubectl で HM に接続しているときに、最初に `kubectl get clusterwrappers` コマンドを使用します。このコマンドは、トレースバックしている Postgres クラスターの名前を、クラスターが属している名前空間と照合し、クラスターの経過期間と現在の状態に関する高レベルの情報を提供します。

```
kubectl get clusterwrappers
```

このコマンドは表示します。

- 名前空間ごとのクラスター。クラスターが関連付けられているプロジェクト ID を含む
- クラスターの名前空間と照合するために使用できるクラスター名
- クラスターの状態と年齢

```
NAME           PROJECTID              CLUSTERNAME        STATE     AGE
p-6gtrowr94d   prj_X9s5sGg5Zl2tDl1v   dbcluster-1        Healthy   2d2h
p-ha6mloybrf   prj_X9s5sGg5Zl2tDl1v   dbcluster-2        Healthy   5d23h
p-p93ou61mgw   prj_X9s5sGg5Zl2tDl1v   dbcluster-3        Healthy   6d
```

`dbcluster1` をその Kubernetes リソースまでトレースしようとしているとします。たとえば、`dbcluster1` が `p-6gtrowr94d` 名前空間にあることがわかりました。各 Postgres クラスターには、名前空間が 1 つだけあります。クラスターの状態は現在正常で、その経過期間は 2 日と 2 時間であることがわかります。

関心がある Postgres クラスターの名前空間の名前がわかっている場合は、それを使用して、その名前空間で実行されているポッドを表示できます。

```
kubectl get pods --namespace p-6gtrowr94d
```

すべてが正常に機能していると、出力として次のようなものが表示されます。

```
NAME             READY   STATUS    RESTARTS   AGE
p-6gtrowr94d-1   3/3     Running   0          2d2h
p-6gtrowr94d-2   3/3     Running   0          2d2h
p-6gtrowr94d-3   3/3     Running   0          2d2h
```

Postgres クラスター保留中

92%完了したプロビジョニング状態の HM を使用して作成した新しいクラスターがあるとします。新しいクラスターの名前は `psrtest-1` で、Kubernetes レベルで何が起こっているのかを確認したいと考えています。最初に `kubectl get clusterwrappers` を使用して、HM の PG クラスターの名前をその名前空間と照合します。

```
kubectl get clusterwrappers
__OUTPUT__
```

(次のページに続く)

(前のページからの続き)

```
NAME           PROJECTID              CLUSTERNAME        STATE     AGE
p-gime4xeamp   prj_X9s5sGg5Zl2tDl1v   psrtest-1          Pending   9m40s
```

クラスターのステータスが保留中であることは既にわかります。HM がまだプロビジョニング中であることを示したため、これは理にかなっています。クラスターの名前空間を取得したため、より詳しく確認し、ポッドのプロビジョニングの状況を確認することもできます。

```
kubectl get pods --namespace p-gime4xeamp
__OUTPUT__
AME                            READY   STATUS      RESTARTS   AGE
p-gime4xeamp-1                  0/3     Pending     0          7m17s
p-gime4xeamp-1-initdb-g6d9b    0/1     Completed   0          9m46s
```

1 つのポッドが正常に作成され、別のポッドがまだ作成中であることがわかります。現時点では、このクラスターが実際に停止しているようには見えません。まだプロビジョニング中です。

しばらくしても、ポッドの保留中ステータスがまだある場合は、リソースが使い果たされたと考える十分な理由があります。例

```
kubectl get pods --namespace p-gime4xeamp
__OUTPUT__
NAME                            READY   STATUS      RESTARTS   AGE
p-gime4xeamp-1                  0/3     Pending     0          28m
p-gime4xeamp-1-initdb-g6d9b    0/1     Completed   0          32m
```

リソースが不足したかどうかを確認するには、 `kubectl describe pod` を使用して、何が起こっているのかを正確に確認できます。

```
kubectl --namespace p-gime4xeamp describe pod p-gime4xamp-1
```

このコマンドは、ポッドに関する不要な多くの情報を結果ます。ただし、出力の最後では、何が問題になっているのかについての詳細を提供します。例、この場合、次のように表示されます。

```
Normal   NotTriggerScaleUp  4m15s (x181 over 34m)  cluster-autoscaler  pod didnt␣
↪trigger scale-up: 7 max node group size reached
```

これは、オートスケーラーが最大ノードグループサイズに到達したことを示しています。その結果、ポッドを完了できません。

> **警告**
>
> 別のクラスターに十分なリソースがある場合でも、同時に作成しようとする数が多すぎると、システムはすべてを一度に作成しようとします。この場合、負荷が分散されるため、どれも完全にプロビジョニングできない可能性があります。

リソースの追加

ポッドを完了できないような最大ノードグループサイズに到達したことを学習した後、最大ノードグループサイズを変更して、より大きくなります。AWS EKS 展開では、AWS コンソールに移動して EKS に移動してこれを行います。

1. 作業している EKS クラスターを選択し、 **Compute** タブを選択します。
2. **ノードグループ** セクションで、PGNodeGroup を選択し、**Edit** を選択します。
3. ノードグループの **最大サイズ**の値を以前よりも大きい値に変更します。HM は、クラスターのプロビジョニングを続行します。

一度に多くのクラスターを作成しようとした場合、それに応じて **最大**サイズを拡張する必要がある場合があります。

Formulas for parameter values

***DB Configuration** tab* のパラメーターに特定の値を入力することに加えて、一部のパラメーターでは、値を計算するための式を指定できます。演算子`+ - / * > >= < <= == != && || ! ? : ( )`を使用して、*shared buffer example* などの三項フォーミュラで整数および実数タイプのパラメーターのフォーミュラを使用できます。`?` および`:` を使用します。必要に応じて、`( )`を使用して操作の順序を指定します。数式で使用される GUC も、整数または実数型である必要があります。ターゲット GUC が整数型で実数でない場合、すべての算術演算は整数の結果に丸められた 64 ビット浮動小数点値で実行されます。

Hybrid ManagerHM には、方程式の作成を支援する **擬似** *GUC* と呼ばれるものもあります。これらの読み取り専用 GUC は次のとおりです。

- `cpu_cores` — クラスターノードを実行している VM が提供するコアの数。
- `disk_iops` —プライマリデータボリュームをバッキングするディスクに割り当てられた IOPS の数。
- `disk_size` —プライマリデータボリュームをバッキングするメモリ量 KB 単位。
- `ram` —クラスターノードを実行している VM によって提供されるメモリ量 KB 単位。
- `replica_count` — クラスター内のノードの合計数。

フォーミュラは、クラスターに対して選択した仕様に相対して、または他の非擬似 GUC に相対して値を計算できます。クラスターのサイズを変更すると、HM は擬似 GUC の新しい値を使用して数式を再計算します。

操作の順序

`*` および`/` は、演算子の優先順位が同じです。 `+` および`-` は、演算子の優先順位が同じです。 `*` および`/` は、`+` および`-` の前に評価されます。かっこを使用して、操作の追加の順序を指定します。

ユニット

GUC は、8 などの裸の数値か、GB などの単位を持つことができます。ユニットは、サイズや期間などのユニットの種類に分類されています。

ターゲット GUC のユニットの種類は、式で使用されるユニットの種類と一致する必要があります。たとえば、3 キロバイトと 2 秒を乗算する数式 `2s *3KB` は使用できませんが、スケールが異なる場合でも 2 つ

のサイズを乗算する数式 2KB* 8GB は使用できます。

ターゲット GUC の単位は、式で使用される単位と一致する必要があります。たとえば、2 秒に 4 日を掛けることはできますが、結果を KB 単位の GUC に割り当てることはできません。

ターゲット GUC に単位がある場合、方程式内の裸の数値はターゲット GUC のデフォルトの単位であるとみなされます。

再帰

再帰の深さは約 100 レベル以下をお勧めします。

例

shared_buffers パラメーターに特定の値を入力する代わりに、GUC 方程式を使用して、shared_buffers の値が RAM の特定のパーセンテージであるが、特定のサイズ以下であることを指定できます。例

```
((ram *0.25) > 8 GB) ? 8 GB : (ram* 0.25)
```

wal_buffers を shared_buffers の 3%、最大 16MB までにするとします。式は次のとおりです。

```
((shared_buffers *0.03) > 16MB) ? 16MB : (shared_buffers* 0.03)
```

フォーミュラは shared_buffers を参照し、この例では独自のフォーミュラがあります。この例は、GUC が他の GUC を参照できる方法を示しています。

一般的なエラーとエラー処理

不正な数式はエラーになります。次にいくつかの例を示します。

- 2 つの単位の数値の後に続く、例

```
work_mem = 10 GB kB
```

この構文では次のエラーが発生します。

```
unexpected token "kB" at end of input (line 1, col 7)
```

- 自分自身を参照する GUC を持つことにより、無限評価ループを作成します。例

```
effective_cache_size = effective_cache_size
```

この構文では次のエラーが発生します。

```
cycle in equation; GUC "effective_cache_size" already seen
```

- ユニットタイプの混合、例

```
work_mem = 10kB * 8s
```

この構文では次のエラーが発生します。

```
both size and time units used in calculation; cannot proceed
```

Managing database cluster access

ID プロバイダーを使用してデータベースクラスターアクセスハイブリッドマネージャーアクセスではなくを構成する場合、*LDAP authentication for database cluster access* を参照してください。

LDAP authentication for database cluster access

データベースクラスター認証用に LDAP を構成できます。

- 適用対象ハイブリッドマネージャーバージョン 1.2 以降

前提条件

- API を使用して LDAP 構成を使用してクラスターを展開する前に、環境に LDAP サーバーが存在する必要があります。
- ユーザーが LDAP を介して認証できる前に、ユーザーは PostgreSQL に存在する必要があります。

API を介して LDAP を構成する

LDAP サーバーをセットアップした後、API を使用してクラスターを展開し、LDAP 認証を構成します。

LDAP を使用する新しいクラスターの作成

REST API を使用する前に、有効なキーがあること、およびクラスター CRUD 操作を実行するための適切なロールが割り当てられていることを確認します。

新しいクラスターの場合、クラスターの作成中に curl を使用して LDAP サーバーを構成します。要求内の`${URL}`、`KEY`、`URL`、`project_id`、およびその他のパラメーターを更新します。

```
curl -H "x-access-key: ${KEY}" -H "Content-Type: application/json" --insecure -s
↪"https://.${URL} edbaiplatform.enterprisedb.network/api/v1/projects/prj_
↪pJ9Ug36492SQhdEQ/clusters" -d {"projectId": "prj_pJ9Ug36492SQhdEQ",
 "psr": {
   "clusterName": "ldap-cluster",
   "password": "111111111111",
   "locationId": "managed-devspatcher",
   "clusterData": {
     "instances": 1,
     "resourceRequest": {
       "requests": {
         "memory": "2Gi",
         "cpu": "1"
       }
     },
     "storageConfiguration": {
       "primaryStorage": {
         "size": "4"
       }
```

(次のページに続く)

(前のページからの続き)

```
    },
    "image": {
      "url": "<pg_image_url>",
      "digest": "<pg_image_sha>"
    },
    "ldapConfig": {
      "enabled": true,
      "config": {
        "hostname": "ldap-service.default.svc.cluster.local",
        "scheme": "ldap",
        "bindAuth": {
          "prefix": "uid=",
          "suffix": ",dc=example,dc=org"
        }
      }
    }
  }
}}
```

LDAP を使用するように既存のクラスターを変更する

既存のクラスターの場合、 PATCH を使用して LDAP 認証用に構成します。

```
curl -X PATCH -H "x-access-key: ${KEY}" -H "Content-Type: application/json" --
↪insecure -s "https://portal.eks-05220612-main.edbaiplatform.enterprisedb.network/
↪api/v1/projects/prj_sXNNRhwj1YddHslR/clusters/p-lfdghq5gxe" -d {"projectId": "prj_
↪sXNNRhwj1YddHslR","clusterId": "p-lfdghq5gxe",
"psr": {
  "ldapConfig": {
   "enabled": true,
   "config": {
    "hostname": "ldap-service.default.svc.cluster.local",
    "scheme": "ldap",
    "bindAuth": {
     "prefix": "uid=",
     "suffix": ",dc=example,dc=org"
    }
   }
  }
 }
}
```

検証

pg_hba.conf ファイルが認証用の LDAP エントリーで更新されたことを確認します。

```
#
#  LDAP CONFIGURATION (optional)

#
host all all 0.0.0.0/0 ldap ldapserver="ldap-server.default.svc.cluster.local"␣
↪ldapscheme="ldap" ldapprefix="uid=" ldapsuffix=",dc=example,dc=org"
```

バインドアンドサーチモード構成

search+bind モードを使用する場合、次のパラメーターが必要です。

- server
- prefix
- suffix

search+bind モードを使用する場合、次のパラメーターはオプショナル。

- searchFilter
- searchAttribute searchAttribute が指定されない場合、デフォルト値は uid に設定されます。
- scheme LDAP スキーム
- port

この例は、検索+バインドモードの構成を示しています。

```
postgresql:
  ldap:
    server: openldap.default.svc.cluster.local
    bindSearchAuth:
      baseDN: ou=org,dc=example,dc=com
      bindDN: cn=admin,dc=example,dc=com
      bindPassword:
        name: ldapBindPassword
        key: data
      searchAttribute: uid
```

バインドモード構成

バインドモードを使用する場合、次のパラメーターが必要です。

- server
- baseDN
- binDN

- bindPassword ldap パスワードを含むシークレット

バインドモードを使用する場合、次のパラメーターはオプショナル。

- scheme LDAP スキーム
- port

この例は、バインドモードの構成を示しています。

```
postgresql:
  ldap:
    server: "openldap.default.svc.cluster.local"
    bindAuth:
      prefix: "uid="
      suffix: ",dc=example,dc=org"
```

LDAP over TLS

TLS を使用して、PostgreSQL サーバーと LDAP サーバー間のトラフィックを暗号化できます。TLS 証明書を使用するように LDAP サーバーを構成できます。

SSL が構成されていない限り、PostgreSQL サーバーと PostgreSQL クライアント間の接続は暗号化されないままです。

詳細情報

- LDAP Authentication
-

LDAP Configuration

Automating database cluster operations

Hybrid ManagerHM から直接、日常的なデータベース操作を自動化できます。HM 管理のクラスターを手動で監視し、問題に対応する代わりに、スケジュールで実行、条件を検出、アクションを実行するルールベースの自動操作を構成します。あなたが管理し続けるためのビルトイン制御を提供します。

自動操作は、大規模な運用データベースを管理するチームのために設計されています。これらは手動介入を削減し、一貫した操作を適用し、統合された**タスクマネージャー** を介して完全な可視性を提供します。これらは、PostgreSQL、 EDB Postgres ExtendedPGE、 EDB Postgres Advanced Server など、サポートされているディストリビューションを実行している HM 管理クラスターで利用できます。

自動操作はクラスターごとに構成します。HA、AHA、DHA クラスターなど、複数のレプリカまたはデータグループを持つクラスターの場合、それぞれに個別に自動化を構成します。

概要

この機能は、すべての自動化された操作にわたって共有機能セットを提供します。

- **スケジュールされた実行** — 構成可能な間隔たとえば `5m` 、`1h` などを使用してオートメーションを実行する頻度を定義します。

- **クールダウン期間** —連続する実行間の最小時間を設定して、急速な繰り返しアクションを防ぎます。最小クールダウン時間は 5 分です。
- **オプションのメンテナンスウィンドウ** —必要に応じて、オペレーションスケジュールに合わせて、オートメーションの実行を定義した時間ウィンドウに制限できます。それ以外の場合、操作は条件が満たされるとすぐに実行されます。
- **ヒューマンインザループの承認** —オプションでユーザーに、実行前にタスクを確認および承認することを要求します。実稼働環境に推奨されます。
- **ロールベースのアクセス制御** —HM の RBAC システムは、ターゲットデータベースに適切な権限を持つユーザーのみがタスクを承認できるようにします。
- **通知の統合** — **アクティビティログ**のタスクイベントは、ユーザーまたはサードパーティシステムへの通知をトリガーします。プロジェクトレベルの Webhook を構成する方法については、通知を設定する オブザーバビリティ を参照してください。
- **タスクマネージャーの統合** —すべての自動化アクティビティはタスクマネージャーで追跡され、実行されたすべてのアクションの集中的な監査証跡を提供します。
- **一時停止と再開** —構成を削除せずにオートメーションを一時的に無効にします。

オートメーションを構成する場合

クラスター操作の自動化は、次の場合に適しています。

- **一貫した反復可能な操作が必要です。**オートメーションは、実行のたびに同じロジックを適用します。これにより、人的エラーが削減され、メンテナンスタスクがその場でではなくポリシーに従って実行されます。
- **ハンズオン実行せずに制御を維持したい。**承認ワークフローを使用すると、アクションが実行される前に人間のサインオフを要求でき、チームに作業を手動で実行せずに可視性と制御を提供します。

使用可能なオートメーション

- ** *Apply index recommendation* ** —最も有益な保留中のインデックス推奨をクラスターに自動的に適用します。HM は、書き込みパフォーマンスの影響と比較して節約された合計クエリ時間に基づいて、最適な推奨事項を決定します。
- ** *Automatic image upgrade* ** —Postgres のディストリビューションを、利用可能な最新のマイナーイメージバージョンに自動的にアップグレードします。
- ** *CPU auto-scaling* ** —平均使用率がしきい値を超えると、CPU を自動的にスケーリングします。
- ** *Database storage auto-scaling* ** —使用率がしきい値を超えると、クラスターのストレージを自動的に拡張し、ディスク全体の停止を防ぎます。
- ** *Memory auto-scaling* ** —平均使用率がしきい値を超えると、メモリを自動的にスケーリングします。

自動操作の構成

プロビジョニングではなく、クラスターが作成された後に、自動操作を構成する必要があります。

1. ターゲットクラスターを開き、[**Settings**] タブに移動します。
2. クラスターに複数のレプリカまたはデータグループがある場合は、**Automations** セクションの上部にあるドロップダウンを使用して、構成するレプリカまたはデータグループを選択します。
3. 設定するオートメーションの横にある **Configure** を選択します。
4. 一般的なオートメーションフィールドに入力します。

Field	Description
Display Name	このオートメーションの説明的な名前たとえば *Database storage auto-scaling – Production* 。
Interval	HM がディスク使用率たとえば *5m*、`1h`などを評価する頻度。
Cooldown Period	スケーリングイベントの後、再度評価するまでに待機する最小時間。これにより、使用量のスパイク中の急速な連続した拡張が防止されます。 AWS 展開の場合、ディスク拡張頻度に関するプラットフォームの制限のため、これを少なくとも `6h`に設定します。
Paused	構成を削除せずにオートメーションを一時的に無効にします。

1. 追加の構成パラメーターについては、 使用可能なオートメーション を参照してください。
2. オプションで、 **メンテナンスウィンドウ**を構成して、自動操作を実行できる時期を制限します。手順については、*Create a maintenance window* を参照してください。
3. オプションで、承認設定を構成します。

- **承認が必要** — オートメーションは **タスクマネージャー**に保留中のタスクを生成します。ユーザーはデータベース操作を実行する前に明示的に承認する必要があります。実稼働クラスターに推奨。- **承認設定>承認の説明** —操作要求の目的を説明する承認者に表示されるメッセージ。 - **承認設定>承認タイムアウト** —タスクの前に承認を待機する時間有効期限たとえば `24h` 、`72h`

  - **Require Reason** —有効にすると、承認者は承認または拒否するときに書面による理由を提供する必要があります。

- **承認は不要** —タスクは手動レビューなしで自動的に実行されます。

メンテナンス時間枠と承認要件を組み合わせることができます。その場合、タスクは実行前に承認され、開いているウィンドウ内にある必要があります。

1. **Create Automation** を選択して、オートメーションをアクティブにします。

共有構成

- *Create a maintenance window* —プロジェクトレベルで 1 回または繰り返しの時間ウィンドウを定義することにより、オートメーションタスクを実行できる時期を制限します。
- *Reviewing and approving automation tasks* — 保留中の自動化タスクを確認し、承認または拒否し、**タスクマネージャー** でタスク履歴をモニタします。

- 通知を設定する オブザーバビリティ —外部ツールでオートメーションアクティビティイベントを受信するように Webhook、電子メール、または Pager Duty を構成します。

制限事項

- クラスターがプロビジョニングされた後にのみ自動化を構成できます。
- 現在、HM コンソールでオートメーションタスクを作成後に削除することはできません。

Apply index recommendation

インデックス推奨の適用は、HM 管理対象クラスターにとって最も有益な保留中のインデックス推奨を選択し、構成された各間隔で適用する自動化です。HM は、書き込みパフォーマンスの影響と比較して節約された合計クエリ時間に基づいて、最適な推奨事項を決定します。

この自動化により、インデックス変更の特定、確認、実行という手動の労力が削減され、パフォーマンスの調整がワークロードの変化に追いつくことが保証されます。CPU やメモリのスケーリングとは異なり、インデックスの作成にはデータベースを再起動する必要はありませんが、インデックスの作成は大規模なデータベースまたはリソースに制限のあるデータベースでのクエリパフォーマンスに影響を与える可能性があるため、この自動化を注意深く計画してください。メンテナンスウィンドウを使用して、インデックス作成を実行する時期を制御し、承認要件を有効にして、チームが適用される前に各推奨事項を確認できます。

> **注釈**
>
> インデックスの推奨の適用は、HM 管理対象クラスターでのみ機能します。外部データベースはサポートされていません。

HA クラスターの場合、各レプリカで個別に構成します。論理レプリケーションは DDL 変更を同期しないため、インデックスはレプリカ間で自動的に伝播されません。

AHA および DHA クラスターの場合、1 つのデータグループで構成するだけで済みます。オートメーションはインデックスを書き込みリーダーに適用し、PGD は変更を他のすべてのデータグループに自動的にレプリケートします。

インデックス推奨パラメーターを適用する

configure an automated operation を行う場合、一般的なオートメーションフィールドに加えて、インデックス推奨の適用オートメーションの特定のパラメーターがあります。

Field	Description
Database Filter	自動化をクラスター内の特定のデータベースに制限します。複数の値を追加できます。すべてのデータベースを検討する場合、空のままにします。
Schema Filter	自動化を特定のスキーマに制限します。複数の値を追加できます。すべてのスキーマを検討する場合、空のままにします。複数のデータベースが選択されている場合、HM はすべてのデータベースで一致するスキーマ名を検索し、一致が見つかった場合に推奨を適用します。

次のページに続く

表 9 – 前のページからの続き

Field	Description
Table Filter	オートメーションを特定のテーブルに制限します。複数の値を追加できます。すべてのテーブルを検討するには、空のままにします。複数のデータベースまたはスキーマが選択されている場合、HM はそれらのすべてで一致するテーブル名を検索し、一致が見つかった場合に推奨を適用します。
Maintenance Window	このメンテナンス時間帯にインデックスの作成をスケジュールします。すぐに実行するには空のままにしてください。 *Create a maintenance window* を参照してください。
Approval	**承認が必要** を選択して、インデックス作成を実行する前にユーザーが承認する必要がある保留中のタスクを作成するか、 **承認は必要ありません** 。手動レビューなしでタスクを自動的に実行します。実稼働クラスターには **承認が必要** をお勧めします。

自動化が実行されるたびに、HM はデータベースクラスターの保留中のインデックスの推奨事項を評価し、最も有益な 1 つを選択します。次に、**タスクマネージャー**でタスクを作成し、構成に従って実行します。承認の有無にかかわらず、すぐにまたは次に開いているメンテナンス時間枠内で実行されます。後続の実行では、HM は次に最も有益な保留中の推奨事項を選択し、保留中の推奨事項がなくなるまで続行します。

タスクはタスクマネージャーに表示され、クラスターレベルの**タスクマネージャー**タブまたは左側のナビゲーションにあるプロジェクトレベルの**タスクマネージャー**からタスクのステータス保留中、完了、または拒否を監視できます。完全な監査可能性のために、すべての実行は**アクティビティログ** に記録されます。

Recommendations タブとこのオートメーションの連携方法

オートメーションは、クラスタービューの *Cluster recommendations* と同じレコメンデーションエンジンから描画します。そのタブの推奨事項のステータスは、オートメーションが選択する内容に直接影響します。

- オートメーションが特定の推奨事項を適用しないようにするには、**Recommendations** タブを開き、**Ignored** としてマークします。無視された推奨事項は自動化から除外され、再現されません。
- **タスクマネージャー**で保留中のタスクを承認しないことは、推奨事項が無視されたものとしてマークされません。オートメーションは、クールダウン期間が終了するとタスクを再作成します。レコメンデーションを完全にスキップするには、代わりに**レコメンデーション** タブで **Ignored** としてマークを付けます。

Automatic image upgrade

自動イメージアップグレードは、HM 管理対象クラスターを最新の状態に保つ自動化であり、最新の利用可能なマイナー Postgres イメージバージョンが利用可能になるたびに、それを自動的に適用します。この自動化により、手動介入を必要とせずにクラスターが最新の状態に保たれるため、資産全体でセキュリティパッチとバグ修正を一貫して維持するのに役立ちます。

オートメーションは、新しいマイナーバージョンのアップグレードのみ 18.1→18.2 などを検出します。同じマイナーバージョンの新しいイメージビルドまたはパッチは、自動的に検出されません。

注釈

Create a maintenance window を構成して、アップグレードの実行時期を制御することをお勧めします。クラスタータイプと構成によっては、イメージのアップグレードにクラスターノードのローリング再起動が含まれる場合があります。

自動イメージアップグレードパラメーター

configure an automated operation を行う場合、一般的なオートメーションフィールドに加えて、自動イメージアップグレードオートメーション用の特定のパラメーターがあります。

Field	Description
Maintenance Window	このメンテナンス時間帯にアップグレードをスケジュールします。新しいイメージバージョンが利用可能になったときにすぐに実行できるように、空のままにしておきます。 *Create a maintenance window* を参照してください。
Approval	**承認が必要** を選択して、アップグレードを実行する前にユーザーが承認する必要がある保留中のタスクを作成します。または、**承認は必要ありません** 手動レビューなしでタスクを自動的に実行します。

HM は、アクティブになると、各インターバルで新しいマイナーイメージバージョンをチェックします。新しいバージョンが利用可能な場合、**タスクマネージャー**でタスクを作成し、構成に従って実行します。承認の有無にかかわらず、すぐにまたは次に開いているメンテナンス時間枠内で実行します。

タスクステータス保留中、完了、またはリジェクトは、左側のナビゲーションにあるクラスターレベルの**タスクマネージャー** タブまたはプロジェクトレベルの**タスクマネージャー**からモニタできます。完全な監査可能性を実現するために、すべてのアップグレードイベントは**アクティビティログ** に記録されます。

ロールバック動作を含む、アップグレードが失敗した場合の動作については、**ロールバックの考慮事項** を参照してください。

CPU auto-scaling

CPU 自動スケーリングは、HM 管理クラスターの平均 CPU 使用率を監視し、使用率が構成されたしきい値を超えると CPU リソースを自動的にスケールアップします。この自動化により、手動介入を必要とせずに、データベースリソースの消費の増加に伴うパフォーマンスの低下と停止の可能性を防ぎます。

この自動化は CPU をスケールアップするだけであり、ダウンすることはありません。CPU 割り当てを削減する必要がある場合は、手動で行う必要があります。

> **警告**
>
> CPU スケーリングは、データベースの再起動をトリガーします。 *Create a maintenance window* を構成して、再起動が発生する時期を制御し、実稼働クラスターでの予期しないダウンタイムを回避します。

Automations セクションのドロップダウンを使用して、レプリカまたはデータグループごとにオートメーションを構成します。スケーリングは、選択したレプリカまたはデータグループ内のすべてのノードに適用されます。レプリカまたはデータグループにわたって調整されていません。他の場所で同じ動作が必要な場合は、それぞれを個別に構成します。

CPU オートスケーリングパラメーター

configure an automated operation を行う場合、一般的なオートメーションフィールドに加えて、CPU オートスケーリングオートメーションの特定のパラメーターがあります。

Field	Description
CPU Threshold	スケーリングイベント `80`などをトリガーする平均 CPU 使用率。最小 `1`、最大 `99`。
Lookback Window (minutes)	HM がしきい値たとえば *5* と比較する前に CPU 使用率を平均する分数。これにより、短期間のスパイクが平滑化されます。最小 `1`、最大 `60`。
Scaling Method	CPU リソースを増やす方法を選択します。 **Scale by quantity** は *Troubleshooting* で表される毎回固定量を追加します。 **Scale by Percentage** は、現在の値のパーセンテージで増加します。
Maximum CPU	CPU スケーリングの上限。オートメーションはこの値を超えてスケールすることはありません。完全なインクリメントがそれを超える場合、代わりに最大値にスケールします。ここに到達すると、それ以上のスケーリングは発生しません。
Maintenance Window	スケーリング操作と必要なデータベースの再起動が実行できるメンテナンスウィンドウ。 CPU のスケーリングには再起動が必要であるため、実稼働クラスターではこれを空のままにすることはお勧めできません。 *Create a maintenance window* を参照してください。
Approval	ユーザーが実行前に承認する必要がある保留中のタスクを **タスクマネージャー** で作成するには **Approval Required** を選択し、手動レビューなしでタスクを自動的に実行するには **Approval Not Required** を選択します。

アクティブになると、HM は、構成されたルックバックウィンドウを使用して、各インターバルでの平均 CPU 使用率を評価します。しきい値を超えると、**タスクマネージャー**でタスクが作成されます。タスクステータス保留中、完了、またはリジェクトは、左側のナビゲーションにあるクラスターレベルの**タスクマネージャー** タブまたはプロジェクトレベルの**タスクマネージャー** からモニタできます。

> **注釈**
>
> CPU スケーリングイベントの後、Postgres 構成を確認して、新しいリソース割り当てに最適化されていることを確認します。追加の CPU を最大限に活用するには、*max_parallel_workers* 、*max_parallel_workers_per_gather* 、*max_worker_processes* などのパラメーターを調整する必要がある場合があります。

Database storage auto-scaling

データベースストレージの自動スケーリングは、HM 管理対象クラスターのデータボリュームのストレージ使用率を監視します。使用量が構成されたしきい値を超えると、HM は基になる `PersistentVolumeClaim` PVC を自動的に拡張します。データベースをオンラインに維持し、ディスクフルイベントによる停止を防ぎます。

暴走を防ぐように、トリガーしきい値、スケーリング方法固定量またはパーセンテージ、および最大ディスクサイズを構成します。拡張はオンライン操作として実行されるストレージドライバーがサポートして

いる場合、全体的に Postgres データベースにアクセスしたままです。

> **注釈**
>
> データベースストレージの自動スケーリングは、プライマリデータボリュームにのみ適用されます。クラスターが別の WAL ボリュームを使用する場合、WAL ボリュームはスコープ内にありません。バックアップストレージも影響を受けません。

Automations セクションのドロップダウンを使用して、レプリカまたはデータグループごとにオートメーションを構成します。HM は、選択されたレプリカまたはデータグループ内の各ノードのデータボリュームを監視およびスケーリングします。オートメーションはレプリカまたはデータグループ間で調整されません。他の場所で同じ動作が必要な場合は、それぞれを個別に構成します。

データベースストレージの自動スケーリングパラメーター

configure an automated operation 、一般的なオートメーションフィールドに加えて、データベースストレージのオートスケーリングオートメーションの特定のパラメーターがあります。

Field	Description
Disk Threshold	スケーリングイベントたとえば *80* をトリガーするディスク使用率。
Scaling Method	ディスクサイズを増やす方法を選択します **Scale by quantity** は、毎回固定サイズを追加しますたとえば *10Gi*、*50Gi* **など。** ***Scale by Percentage*** **は、現在の容量のパーセンテージで増加しますたとえば** *20%* 。
Maximum Disk Size	スケーリングの上限。このサイズに到達すると、自動化の拡張が停止し、暴走的な成長を防ぎます。
Maintenance Window	ストレージドライバーがオンライン拡張をサポートしている場合、つまり操作でダウンタイムが発生しない場合、メンテナンスウィンドウフィールドを空のままにすることができ、しきい値を超えるとすぐにタスクが実行されます。
Approval	ユーザーが実行前に承認する必要がある保留中のタスクを **タスクマネージャー** で作成するには **Approval Required** を選択し、手動レビューなしでタスクを自動的に実行するには **Approval Not Required** を選択します。

HM は、アクティブになると、各インターバルでディスクの使用率を評価します。しきい値を超えると、**タスクマネージャー**でタスクが作成されます。タスクステータス保留中、完了、またはリジェクトは、左側のナビゲーションにあるクラスターレベルの**タスクマネージャー** タブまたはプロジェクトレベルの**タスクマネージャー** からモニタできます。

Memory auto-scaling

メモリの自動スケーリングは、HM 管理対象クラスターの平均メモリ使用率を監視し、使用率が構成されたしきい値を超えるとメモリリソースを自動的にスケールアップします。この自動化により、手動介入を必要とすることなく、データベースリソースの消費の増加に伴うパフォーマンスの低下を防ぎます。

警告

メモリーのスケーリングは、データベースの再起動をトリガーします。 *Create a maintenance window* を構成して、再起動が発生する時期を制御し、実稼働クラスターでの予期しないダウンタイムを回避します。

Automations セクションのドロップダウンを使用して、レプリカまたはデータグループごとにオートメーションを構成します。スケーリングは、選択したレプリカまたはデータグループ内のすべてのノードに適用されます。レプリカまたはデータグループにわたって調整されていません。他の場所で同じ動作が必要な場合は、それぞれを個別に構成します。

メモリのオートスケーリングパラメーター

configure an automated operation を行う場合、一般的なオートメーションフィールドに加えて、メモリオートスケーリングオートメーションの特定のパラメーターがあります。

Field	Description
Memory Threshold	スケーリングイベント`80`などをトリガーする平均メモリ使用率。最小`1`、最大`99`。
Lookback Window (minutes)	しきい値たとえば *5*`と比較する前に、*HM* がメモリ使用量を平均する分数。これにより、短期間のスパイクが平滑化されます。最小`1`、最大`60`。
Scaling Method	メモリリソースを増やす方法を選択します。 **Scale by quantity** は *Troubleshooting* で表されるたびに固定量を追加します。
Maximum Memory	メモリスケーリングの上限たとえば *32Gi* 。オートメーションはこの値を超えてスケールすることはありません。完全なインクリメントがそれを超える場合、代わりに最大値にスケールします。ここに到達すると、それ以上のスケーリングは発生しません。
Maintenance Window	スケーリング操作と必要なデータベースの再起動が実行できるメンテナンスウィンドウ。メモリのスケーリングには再起動が必要であるため、これを空のままにすることは、運用クラスターではお勧めできません。 *Create a maintenance window* を参照してください。
Approval	ユーザーが実行前に承認する必要がある保留中のタスクを **タスクマネージャー** で作成するには **Approval Required** を選択し、手動レビューなしでタスクを自動的に実行するには **Approval Not Required** を選択します。

HM は、アクティブになると、構成されたルックバックウィンドウを使用して各インターバルの平均メモリ使用率を評価します。しきい値を超えると、**タスクマネージャー**でタスクが作成されます。タスクステータス保留中、完了、またはリジェクトは、左側のナビゲーションにあるクラスターレベルの**タスクマネージャー** タブまたはプロジェクトレベルの**タスクマネージャー** からモニタできます。

注釈

メモリスケーリングイベントの後、Postgres 構成を確認して、新しいリソース割り当てに最適化されていることを確認します。追加メモリを最大限に活用するには、 *shared_buffers* 、 *effective_cache_size* 、 *work_mem* などのパラメーターを調整する必要がある場合があります。

Create a maintenance window

メンテナンスウィンドウを使用すると、オートメーションの実行を定義した期間に制限できます。条件が満たされた瞬間にタスクを実行する代わりに、ハイブリッドマネージャー HM は、クラスターが開いたウィンドウ内に収まるまで待機してから動作します。

プロジェクトレベルでメンテナンスウィンドウを構成でき、プロジェクト内のすべてのクラスターで共有および再利用できます。ウィンドウを作成したら、個々のオートメーションを構成するときにウィンドウを参照します。

ウィンドウが開いているときにタスクがトリガーされると、すぐに実行されます。その時点で開いているウィンドウがない場合、タスクは待機して、次の発生時に実行されます。

> **注釈**
>
> メンテナンス時間はオプショナル。操作でダウンタイムが必要ない場合たとえば、ストレージドライバーがオンライン拡張をサポートしている場合のストレージの自動増加など、自動化の構成時にメンテナンスウィンドウフィールドを空のままにすることができ、条件が満たされるとすぐにタスクが実行されます。

メンテナンスウィンドウを作成する

1. HM コンソールで、 **プロジェクト** に移動します。
2. **[設定] > [メンテナンス Windows]** に移動します。
3. **Create New Window** を選択し、次のフィールドに入力します。

Field	Description
Display Name	ウィンドウの説明的な名前たとえば `Weekend maintenance` または `Nightly – pre-prod` など。
Type	繰り返しスケジュールの場合は **Recurring (Cron)** を選択し、単一の発生の場合は **One-Time** を選択します。

Recurring (Cron) を選択した場合

Field	Description
Cron Expression	ウィンドウが開始される時期と、それが繰り返される頻度たとえば、毎週土曜日の午後 10 時に `0 22 ** 6` などを定義します。
Duration	ウィンドウが起動するたびに開いたままになる時間たとえば *2h* 。

ワンタイム を選択した場合

Field	Description
Start Time	ウィンドウが開いた日時 UTC。
End Time	ウィンドウが閉じた日時 UTC。

1. メンテナンスウィンドウを保存します。プロジェクト内のクラスターのオートメーションを構成するときに選択できるようになりました。

使用法とスコープ

ウィンドウをオートメーションに適用するには、オートメーション設定の**メンテナンスウィンドウ**ドロップダウンから選択します。メンテナンスウィンドウと承認要件を組み合わせることができます。その場合、タスクは実行前に承認され、開いているウィンドウ内にある必要があります。

メンテナンス時間はプロジェクトスコープです。プロジェクト内のすべてのオートメーションとクラスターで単一のウィンドウを再利用できますが、プロジェクト間で共有されません。複数のプロジェクトで同じスケジュールが必要な場合は、各プロジェクトに個別にウィンドウを作成します。将来的には、メンテナンスウィンドウは、自動化を超えて他の種類のクラスター操作に使用できるようになる予定です。

Reviewing and approving automation tasks

承認が必要でオートメーションを構成すると、ハイブリッドマネージャー HM はオペレーションをすぐに実行しません。代わりに、保留中のタスクを作成し、チームメンバーがそれを確認するのを待機します。**タスクマネージャー**は、そのレビューが行われる場所です。プロジェクト全体のすべての保留中、実行中、および完了したオートメーションタスクを一元的に表示するため、何をいつ実行するかを制御できます。HM API を使用してタスクをプログラム的に管理することもできます。これにより、承認フローをサードパーティのツールやワークフローと統合できます。

タスクマネージャーへのアクセス

タスクマネージャー は、2 つのレベルで使用できます。

- **プロジェクトレベル** — プロジェクト内の、左側のナビゲーションで、**Task Manager** を選択します。このビューには、プロジェクト内のすべてのクラスターの保留中、実行中、失敗した、および完了したタスクがすべて表示されます。
- **クラスターレベル** — プロジェクト内の、左側のナビゲーションで、**Clusters > Postgres Clusters** を選択し、クラスターを選択します。クラスタービューで、 **Task Manager** タブを選択します。このビューには、その特定のクラスターのタスクのみが表示されます。

タスクリストの読み取り

タスクリストの各行は、単一のオートメーションタスクを表します。次の列がデフォルトで表示されます。

Column	Description
Task Name	オートメーションタスクの名前。名前を選択して、タスクの詳細サイドバーを開きます。
Status	タスクの現在の状態。 タスクのステータスを理解する を参照してください。

次のページに続く

表 17 – 前のページからの続き

Column	Description
Created	タスクが作成されたとき。
Duration	タスクの実行にかかった時間。まだ実行されていないタスクにはダッシュ - が表示されます。
Source	タスクをトリガーした自動ルール。
Target	タスクのターゲットとなるクラスター。タスクマネージャーは、特定のクラスター内から表示する場合、この列を非表示にします。

タスクリストの上部にある設定アイコンを使用して、表示する列をカスタマイズできます。**Priority** はオプションの列であり、デフォルトでは表示されません。

タスクのステータスを理解する

各タスクは、 **ステータス** 列で追跡できるライフサイクルを移動します。

Status	Meaning
Pending	The task is waiting — either for approval (if the automation requires approval) or for its maintenance window to open.
Gate Satisfied	All conditions are met. The task executes when its maintenance window opens, or immediately if you didn't configure a window.
Running	The task is currently executing.
Completed	The task finished successfully.
Failed	The task couldn't run or encountered an error. When a specific failure reason is available, the status badge shows an error code instead of *Failed*. See 障害エラーコード .
Cancelled	A user or process canceled the task before it ran.

障害エラーコード

タスクが失敗すると、ステータスバッジに特定のエラーコードが表示され、その理由を示します。

Error code	Meaning
MaintenanceWindow-Failed	The maintenance window closed before the task could run. If the automation's cooldown period has passed, HM recreates the task on the next evaluation cycle.
ApprovalFailed	A user rejected the task, or the approval timeout expired before anyone approved it. If the automation's cooldown period has passed, HM recreates the task on the next evaluation cycle.
ExecutionError	The task started but encountered an error during execution. Check the task detail sidebar for the error message.
Timeout	The task exceeded the maximum allowed execution time.

タスクの承認または拒否

タスクが **保留中** ステータスであり、あなたの承認が必要な場合

1. タスク名またはそのステータスバッジを選択して、タスクの詳細サイドバーを開きます。
2. タスクの詳細を確認します。オートメーションタイプ、ターゲットクラスターまたはデータグループ、オートメーションの所有者が構成した承認の説明など。
3. **Approve** を選択してタスクの続行を許可するか、 **Deny** を選択して拒否します。
4. **Require Reason** でオートメーションを構成した場合、決定を確認する前に **Reason** フィールドに説明を追加します。

承認後、タスクはメンテナンスウィンドウを構成していない場合は **実行中**に、メンテナンスウィンドウが存在する場合は **ゲート満足度**に移動します。拒否後、HM はタスクをクローズします。オートメーションのクールダウン期間が経過した場合、HM は次の評価サイクルでタスクを再作成します。

> **注釈**
>
> タスクを承認するには、クラスターで基になる操作を実行するために必要なものと同じ権限が必要です。たとえば、自動イメージアップグレードの承認には、手動アップグレードを実行する場合と同じ権限が必要です。 HM の RBAC システムはこれを強制します。

保留中の承認タスクの通知

オートメーションタスクが **保留中**ステータスに移行し、承認が必要になると、HM はプロジェクトまたは組織レベルで構成したチャネルを介して通知を送信します。

- **アプリ内** —HM コンソールの右上隅のベルアイコンメニューに通知が表示されます。通知はタスクのステータスの変更を反映し、システムアクションに注意が必要であることを示します。ベルのアイコンを選択して表示し、**タスクマネージャー** に移動してタスクを確認および承認します。
- **外部チャネル** 電子メール、Pager Duty、または Webhook を構成している場合、HM はこれらのチャネルを介して保留中のタスクに関する追加コンテキストを含む通知を送信します。**Task Manager** に移動して、タスクを承認または拒否します。

どちらの場合も、 **タスクマネージャー**で承認を完了する必要があります。通知自分自身には、直接承認アクションは含まれません。

タスク履歴のモニタリング

タスクマネージャーは、完了した、失敗した、およびキャンセルされたすべてのタスクの履歴を保持します。これを使用して、何が、いつ、どのような結果で実行されたかを監査します。

リストを絞り込むには、使用可能なフィルタを使用します。

- **Status** — 1 つ以上のタスクフェーズでフィルターしますたとえば、`Pending` または `Failed` タスクのみを表示します。

- **日付範囲** — 結果を特定の時間ウィンドウに制限します。7、15、30、および 90 日のプリセット範囲を使用できるか、カスタム範囲を設定します。
- **検索** — 名前でタスクを検索します。

タスクを選択して、詳細サイドバーを開きます。*Auditing platform activity* は、すべてのタスクアクティビティを記録し、タスクだけでなくすべてのリソースタイプにわたる組織またはプロジェクト全体の監査証跡を提供します。

Cluster view

クラスタービューは、クラスターの構成、状態、モニタリング、クエリ診断、バックアップ、ログ、その他のプロパティを含むクラスターの詳細な概要を提供します。Cluster ビューを開くには、 Clusters テーブルからクラスターを選択します。

[クラスター] ビューには、クラスターの名前、作成時刻、クラスターの種類、使用される Postgres ディストリビューション、および Postgres ディストリビューションのバージョンが表示されます。作成時間は、クラスターが現在時刻と相対して作成された時間を示します。クラスター名前を選択して、クラスターの接続情報ページを開きます。

タグは、管理を簡単にするためにクラスターをグループ化するために使用されます。クラスタービューでは、タグがリンクのリストとして表示されます。タグを選択して、そのタグを持つすべてのリソースを表示する詳細ビューを開きます。

タグを追加または削除するには、 + を選択します。ダイアログボックスで、タグを追加および作成し、クラスターからタグを削除できます。

クラスタービューのタブは次のとおりです。

- **概要** — クラスターの簡単な概要。
- **接続** — クラスターの接続情報。
- **モニタリングの実行** — クラスターのモニタリング情報。
- *Diagnostics [***クエリーの診断と推奨事項***] タブ* — クラスターの診断情報を照会します。
- **マルチ** *DC* **環境の** *HM* **バックアップと** *DR* — クラスターのバックアップ情報。
- **ログと可観測性** — クラスターのログ。
- *Cluster Properties* — クラスターのプロパティ。
- *Health assessment reports* —データベースクラスターの状態評価レポートを生成、表示、ダウンロード、および比較します。

Cluster overview

[**Cluster Overview**] タブには、クラスターの接続文字列、アクティブなアラート、および最近のクラスターアクティビティの概要が表示されます。

クラスターが単一ノードまたはプライマリ/スタンバイの高可用性クラスターの場合、概要には最初に **接続文字列の取得** セクションが表示されます。

クラスターが分散高可用性クラスターの場合、概要には最初に *2* つのデータグループと *1* つの監視グループを含む *EDB Postgres* 分散 *PGD* クラスターが正常ではありません セクションが表示されます。

接続文字列の取得

Quick Connect セクションは、クラスター用に構成された接続文字列を含む完全な psql コマンドラインと、edb_admin データベースに接続するための edb_admin ロールを持つユーザーを提供します。このコマンドをコピーしてターミナルに貼り付けて、データベースに接続できます。データベースに接続するには、edb_admin ユーザーのパスワードが必要です。

データグループ

Data Groups の下で、クラスター内のデータグループと監視グループの概要を表示します。各グループは、ステータスバッジ、データグループの名前、および**データグループ** または **Witness Group** ラベルでリストされます。

ステータスバッジは、データグループのステータスを示します。白いチェックマークの付いた緑の円は、データグループが正常であることを示します。表示されるその他の情報には、作成からの時間、Postgres サーバータイプとバージョン、データグループ内のノード数が含まれます。

- データグループのノード設定とデータベース構成を編集するには、データグループ名の横にある **Edit Data Group** を選択します。
- データグループが複数ある場合、データグループを削除できます。すべてのデータグループを削除することはできません。省略記号メニューから **Delete Data Group** を選択します。
- 削除したデータグループを復元するには、省略記号メニューから **Restore Data Group** を選択します。
- 書き込みリーダースイッチオーバーを実行するには、省略記号メニューからスイッチオーバー書き込みリーダー を選択します。

データグループの **Quick Connect** の下で、読み取り-書き込みおよび読み取り専用接続の psql コマンドライン接続文字列を取得できます。

Read-write または **Read-only** を選択して、データグループと bdrdb データベースに接続する edb_admin ロールを持つユーザー用に構成された接続文字列を含む psql コマンドラインを表示します。データベースに接続するには、このコマンドをコピーしてターミナルに貼り付けます。これを行うには、edb_admin ユーザーのパスワードが必要です。

> **注釈**
>
> **読み取り専用**接続文字列は、Hybrid ManagerHM1.1 では使用できません。

クラスターアーキテクチャ図の表示を切り替えるには、 **Cluster Architecture Diagram** 見出しを選択します。この図は、データグループ、監視グループ、プロキシを含むクラスターのアーキテクチャと、それらの間の相互接続を示しています。

スイッチオーバー書き込みリーダー

高度な高可用性または分散高可用性クラスターデータグループ内で書き込みリーダーのスイッチオーバーを開始して、書き込みリーダーのロールをあるノードから別のノードにシームレスに転送します。このデータ損失ゼロオペレーションは、ダウンタイムを最小限に抑えるように設計されており、実行中のアプリケーションがトランザクションを失わずにスムーズな移行を保証します。

書き込みリーダーのスイッチオーバーは、いくつかの重要な目的に役立ちます。

- **計画的なメンテナンス**パッチ適用、OS の更新、またはハードウェアの交換のためにノードをオフラインにする前に、書き込みリーダーのロールを正常なノードにシフトし、必要なメンテナンスを実行して、ロールを元に戻します。
- **ルーチンテスト**定期的なディザスターリカバリー訓練の一部としてスイッチオーバーを実行して、プロセスを検証し、実際のイベント中に予想される機能を保証します。
- **ロードバランシング**書き込みリーダーロールを別のノードに移動して、クラスター全体でワークロードを分散するか、書き込みリーダーを特定のアプリケーショングループの近くに配置します。

詳細は Switching the write leader でご覧いただけます。

アクティブなアラート

現在アクティブなアラートの数は、高重大度、中重大度、および低重大度の 3 つのカテゴリにリストされます。表示されている数を選択すると、**クラスターモニタリング、アラート、およびオブザーバビリティ**タブが開き、アラートの詳細を表示できます。

クラスターアクティビティログ

クラスターに記録された最近のアクティビティには、作成、構成の更新、およびクラスターのライフサイクル内のその他のメジャーイベントが含まれます。アクティブな要素はここには表示されません。イベントの詳細については、*Cluster Logs* タブを選択してください。

HM は、HM 管理データベースのスキーマ変更を検出すると、クラスターアクティビティログにイベントエントリを自動的に挿入します。これらのエントリを使用して、スキーマ変更が発生した時期を追跡し、クラスターパフォーマンスの変化と相関させることができます。

> **注釈**
>
> クラスターアクティビティログは、自己管理クラスターでは使用できません。自己管理クラスターのスキーマ変更イベントは、代わりに**クラスターモニタリング、アラート、およびオブザーバビリティ**タブにイベントマーカーとして表示されます。

Cluster connections

Connect タブを使用して、クラスターの接続情報を取得し、クラスター内の `edb_admin` データベースに接続するための edb_admin ロールを持つユーザーを取得します。

クラスターが分散高可用性クラスターの場合、接続情報はクラスター内のデータグループごとに提供されます。

接続情報

次の接続情報を表示できます。

- **読み取り/書き込み URI** 読み取り/書き込みアクセスでクラスターに接続するための接続文字列。これは、**Quick Connect** セクションの psql コマンドラインと同じです。
- **Dbname** 接続するデータベースの名前。これは常に `edb_admin` です。
- **読み取り/書き込みホスト**読み取り/書き込みアクセスのためのクラスターのホスト名または IP アドレス。
- **Port** 接続するポート。これは常に `5432` です。
- **User** 接続するユーザー。これは常に edb_admin です。

これらの各フィールドをコピーして、データベースへの接続試行を開始するために使用できます。

読み取り専用の接続情報

クラスターを作成または編集するときに読み取り専用接続を有効にしている場合、[**Connect**] タブには読み取り専用の接続文字列も表示されます。この接続文字列を使用して、読み取り専用操作をスタンバイレプリカ HA クラスターの場合、または現在書き込みリーダーとして指定されていないノード AHA および DHA クラスターの場合にルーティングし、プライマリインスタンスの負荷を削減します。

次の読み取り専用の接続情報を表示できます。

- **読み取り専用 URI** 読み取り専用アクセスのためにクラスターに接続するための接続文字列。
- **読み取り専用ホスト**読み取り専用エンドポイントのホスト名または IP アドレス。

DHA クラスターの場合、読み取り専用の接続情報はデータグループごとに個別に提供されます。

読み取り専用接続を有効にするか、構成するには、HA クラスターの場合は *Cluster setup* 、AHA および DHA クラスターの場合は *2* **つのデータグループと** *1* **つの監視グループを含む** *EDB Postgres* **分散** *PGD* **クラスターが正常ではありません** を参照してください。

`.pgpass` blob

`.pgpass` blob をコピーして `.pgpass` ファイルに貼り付けることができるため、パスワードを入力せずにクラスターに接続できます。これは、スクリプトやアプリケーションなど、クラスターへの接続を自動化する場合に役立ちます。プレースホルダーを実際のパスワードに置き換えます。不正なアクセスを防ぐには、`.pgpass` ファイルに正しい権限が設定されていることを確認します。権限を `0600` ユーザーの読み取りおよび書き込みのみに設定します。

`.pgpass` ファイルの詳細については、PostgreSQL documentation を参照してください。

`.pg_service.conf` blob

`.pg_service.conf` blob をコピーして `.pg_service.conf` ファイルに貼り付けることができるため、**サービス名**で参照してクラスターに接続できます。これにより、クラスターへの接続を覚えやすくなります。`.pgpass` blob と組み合わせて、サービス名を指定して接続できます。`.pg_service.conf` ファイルの詳細については、

PostgreSQL documentation を参照してください。

pgpass および.pg_service.conf ファイルは、libpq を使用する Postgres クライアントアプリケーションでサポートされています。

Cluster monitoring

[**Monitoring**] タブは、ノードとクラスターの状態の詳細なビューを提供します。

モニタリング

クラスターの状態の概要については、**Monitoring** セクションを確認してください。テーブルの上部にあるモニタリングコントロールを使用して、テーブルに表示される内容を制御します。

表示されるデータの開始時期を指定できます。デフォルトは 15 分で、これは、過去 15 分間のデータを表示することを意味します。15 日〜5 分の値を指定できます。または、日付と時刻ピッカーを使用して、監視データをサンプリングするカスタム時間範囲を選択します。クラスターのデータが利用可能な場合にのみ時間範囲を選択できます。最初のメニューから時間を選択すると、指定した日付と時刻がクリアされます。

クラスター内のすべてのノードの集計としてデータを表示するかどうか**クラスタレベル** 、または各ノードとして表示するかどうか**ノードレベル**を指定することもできます。デフォルトでは、データをクラスターレベルで表示することです。

HM によって検出されたスキーマ変更イベントを含むイベントマーカーをオーバーレイするために、各監視カテゴリ**ホスト** 、 **接続** 、 **トランザクション** などの **表示イベント**を有効にします。 EDB Postgres AI エージェントを介して監視される自己管理クラスターの場合、スキーマ変更イベントはクラスターアクティビティログではなくイベントマーカーとして表示されます。

クラスターレベルでは、各リングチャートは、クラスター内のすべてのノードのデータの平均を示す単一のリングであり、グラフは同様の集計にもとづいています。

ノードレベルでは、各チャートは同心リングチャートです。各リングは、クラスター内のノードを表します。グラフはノードごとに個別に表示されます。

期間の比較

比較ビューでは、2 つの隣接しない期間間のモニタリングメトリックとクエリ統計を並べて比較できるため、変更、インシデント、またはワークロードのシフトの影響を理解できます。これを開くには、**Monitoring** タブから **+ New Comparison** を選択します。

参照期間 と **比較期間**を定義します。期間ごとに、日付と時刻ピッカーを使用して、または展開マーカーイベントを選択して、開始時刻と終了時刻を設定します。**Monitoring** および **Query Diagnostics** トグルを使用して、これらのセクションを比較に含めるまたは除外します。

ビューには、両方の期間のチャートが並べて表示され、それらの間のデルタバッジがパーセンテージの差を示します。グラフの下の**リファレンス期間と比較期間**セクションでは、両方の期間のクエリパフォーマンスを比較しています。結果のスコープは単一のクラスターに限定され、クラスターレベルとノードレベルのビューの両方をサポートします。

> **注釈**
>
> データの可用性は保持制限の対象です。モニタリングメトリックは最大 30 日間、クエリ統計は最大 7 日間保持されます。

アクティブなアラート

アクティブなアラートの概要には、アクティブなアラートの数が高重大度、中重大度、低重大度の 3 つのカテゴリで表示されます。

アラートの検索可能なテーブルビューでは、検索/フィルターバーを使用して次のことができます。

- テキスト検索用語でフィルタリングします。 **Search** ボックスに値を入力します。
- 時間範囲でフィルタリングします。 **From** および **To** 日付/時刻ピッカーを選択します。
- 重大度によるフィルタリング。 **Filter > Severity** を選択し、1 つ以上の使用可能な重大度を選択します。
- 開始時間、重大度、またはアラートで昇順または降順で並べ替えます。
- 稲妻アイコンを選択して、自動更新を選択します。
- 表示されている列を選択します。歯車を選択し、メニューからテーブルに表示する列を選択します。
- 現在表示されているアラートを CSV 形式でダウンロードします。ダウンロードアイコンを選択します。

ホスト

グラフに表示されるオペレーティングシステム統計の概要については、**Host** セクションを確認します。

- **Memory** Postgres プライマリノードのメモリの平均メモリ使用率。
- **CPU** Postgres プライマリノードの平均 CPU 割合。
- **Storage** Postgres プライマリノードに使用される合計ストレージ。
- **Disk IOPS** 一定期間における 1 秒ごとのディスクの読み取り、書き込み、および合計操作の合計数。
- **ディスクスループット** Postgres ノードの 1 秒ごとにディスクとの間で転送されるデータの合計量。
- **ネットワークアクティビティ** Postgres ノードの期間、1 秒ごとにネットワークカードとの間で転送されるデータの合計量。

接続

- **Connections** クライアントアプリケーションと Postgres データベース間のタイプ別の現在の接続数。
- **待機タイプ別の平均アクティブセッション**プライマリ Postgres ノードが各待機イベントタイプに費やした時間は、平均アクティブセッション AAS を使用して計算されます。
- **ブロックされたバックエンドの数**クラスター内のすべての Postgres ノードにわたってロックで待機しているバックエンドの合計数。

トランザクション

- **Tuples In** Postgres ノードの 1 秒ごとに挿入、更新、および削除されたタプルの合計数。
- **Tuples Out** Postgres ノードの 1 秒ごとに取得および返されたタプルの合計数。

- **トランザクションレート** Postgres ノードの 1 秒ごとのコミット/ロールバック/合計トランザクションの合計数。
- **バッファキャッシュヒット率**クラスター内のすべての Postgres ノードの平均バッファキャッシュヒット率。
- **最長実行トランザクション** Postgres ノードの 1 秒ごとの最長実行トランザクションの合計数。

クエリー

- **Query Rate** Postgres ノードの 1 秒ごとのクエリーの合計数。
- **Query Latency** クラスター内のすべての Postgres ノードのミリ秒単位の平均クエリレイテンシー。

ストレージ

- **データベースサイズ** 折れ線グラフプライマリ Postgres ノード全体のデータベースサイズの合計。
- **ディスク使用量**プライマリ Postgres ノードのディスク使用量のパーセンテージ。
- **WAL Size** プライマリ Postgres ノード全体の WAL ディレクトリの合計サイズ。
- **WAL 使用量** プライマリ Postgres ノードの WAL 使用率。
- **Live/Dead Tuples** Postgres ノードの Live/Dead タプルの合計数。
- **インデックス、テーブル、および一時的な推定サイズ** Postgres ノードのインデックス、テーブル、および一時の推定サイズ。

内部

- **最後の自動バキュームからの時間** Postgres ノードの秒単位の最後の自動バキュームからの時間。
- **Autovacuum Stats** Postgres ノードの 1 秒ごとの自動バキューム操作の合計数。
- **最後のチェックポイントからの時間** Postgres ノードの秒単位の最後のチェックポイントからの時間。
- **Checkpoint Stats** Postgres ノードの 1 秒ごとのチェックポイントの合計数。
- **最後の成功した WAL アーカイブからの時間** Postgres ノードの秒単位の最後の成功した WAL アーカイブからの時間。
- **WAL Archiving Stats** Postgres ノードの 1 秒あたりの WAL アーカイブ操作の合計数。

Cluster Query Diagnostics

クエリ診断

[Query Diagnostics] タブは、クラスターで実行されているワークロードに関する洞察を提供します。実行されたクエリに関する詳細な統計を収集するため、ユーザーは問題のある領域をすばやく特定できます。

[Query Diagnostics] タブは、4 つのグラフとクエリの表で構成され、クエリの実行に関するパフォーマンスの問題を診断するための役立つ統計を提供します。

チャート

チャートは次のとおりです。

- 「クエリレイテンシー」、クエリレイテンシーのパーセンタイル 50、90、および 99 を含む折れ線グラフ、
- 「実行されたクエリーの数」、実行されたクエリーの生の数を含む折れ線グラフ、
- クエリレイテンシーヒストグラム、クエリレイテンシー分布を含む棒グラフ、
- 「アクティブなセッションの数」アクティブなセッションの数と、I/O、CPU、IPC、およびロックの関連する待機状態を含む積み上げ面グラフ。

フィルタを使用すると、ユーザーは次の方法でデータを制限できます。

- 時間間隔、
- 開始日と終了日、
- 特定のノード。

クエリテーブル

グラフの下のテーブルには、収集されたすべてのクエリーとその情報と統計がリストされています。

- “Query ID” クエリの ID、
- “Query”、切り詰められたクエリテキスト、
- 「# of Executions」このクエリの実行数、
- “50th Percentile” クエリレイテンシーの 50 パーセンタイル
- 「90th Percentile」、クエリレイテンシーの 90 パーセンタイル、
- 「99th Percentile」 クエリレイテンシーの 99 パーセンタイル
- 「CPU %」 CPU での作業に費やされた時間の割合、
- 「Wait %」待機して費やされた時間つまりアクティブですが、CPU ではない時間の割合、
- “最小実行時間”、
- “最大実行時間”、
- “CPU AAS”、クエリによって CPU で費やされた平均アクティブセッション
- 「Wait AAS」、クエリーによって待機状態で費やされた平均アクティブセッション、
- “Total AAS”、クエリの実行に費やされた平均アクティブセッション、
- 「Count」、このクエリを少なくとも 1 回実行したユーザー、データベース、およびノードの数。

歯車アイコンを使用すると、列の表示/非表示が可能です。

フィルタを使用すると、ユーザーは次の方法でデータを制限できます。

- データベース,

- ユーザー。

右側のドロップダウンメニューを使用すると、テーブルの行を列ごとに昇順または降順で並べ替えることができます。

左側の検索バーを使用して、クエリテキストの部分文字列を検索できます。

クエリ診断について

クエリーとは何ですか?

クエリ診断を行うために、述語が削除された後、一意のクエリテキストとしてクエリを定義しました正規化とも呼ばれます。たとえば、`SELECT id FROM users WHERE name = "Bob"` は `SELECT id FROM users WHERE name = "Alice"` と同じクエリーです。これは、クエリーが別のデータベースで別のバージョンの `users` テーブルに対して実行される場合にも当てはまります。

平均アクティブセッション AAS とはどういう意味ですか?

平均アクティブセッションは、すべてのセッションにわたって特定の状態で費やされた時間を示すメトリックです。これは、すべてのセッションを合計して、目的の状態で費やされた各セッションの時間の割合として計算されます。

たとえば、単一のセッションが同じクエリを 1 時間継続して実行した場合、そのクエリはその時間に 1AAS を占有すると言われます。クエリの実行に 30 分のみを費やした場合、それは 0.5 AAS になります。

代わりに 10 のセッションがそれぞれ 30 分ずつクエリを実行している場合、それは 5 AAS になります。

平均アクティブセッション数は、「昨日のトップ 10 クエリーは何でしたか?」という質問に答えるために役立つメトリックです。定義では、最も高い AAS を持つ 10 のクエリーは、一日全体、すべてのセッションにわたって、最も時間を占めたクエリーとなります。AAS の場合、クエリが 1 日に 100 万回実行されたか、1 回だけ実行されたかは関係ありません。合計実行時間を測定します。

データはどこから来たのですか?

クエリ診断用のデータは、2 つの拡張機能から取得されます。1 つだけ使用可能な場合、機能が制限されます。どちらも使用できない場合、クエリ診断は無効になります。

- EDB Wait States は、セッション状態からサンプリングデータを収集します。これは、AAS および関連メトリックを設定するために使用されます。
- EDB Stat Monitor は、クエリ実行レベルでデータを収集します。デフォルトでは、すべてのクエリ実行が記録されます。これは、実行数を入力し、レイテンシーメトリックを照会するために使用されます。

クエリ診断には、実行されたすべてのクエリが表示されますか?

はい、EDB Stat Monitor が有効になり、`edbsm_sample_rate` がデフォルトの 1 に設定されている場合、クエリ診断には、`SET` などのユーティリティクエリや PGPLSQL プロシージャ内で実行されるクエリを除き、実行されたすべてのクエリが含まれます。

EDB Stat Monitor が有効になっていない場合、表示されるクエリは、EDB 待機状態によって観察されたクエリのみです。これには、大量の AAS に寄与するクエリが含まれる可能性がありますが、これは保証できません。

Cluster recommendations

おすすめ

[**Recommendations**] タブには、Postgres クラスターのパフォーマンスの向上に役立つスコアカードと推奨事項のリストが表示されます。

スコアカードと推奨事項は直接関係しています。クラスターは、推奨事項が表示される場合にのみ Excellent より低いスコアとなります。推奨事項を適用または却下すると、スコアに Excellent と示されます。

スコアカード

スコアカードでは、スコアは各評価カテゴリの特定のクラスターに割り当てられます。

- インデックス
- 統計
- 構成
- セキュリティ
- ワークロード

> **注釈**
>
> 現在、インデックス、統計、構成、およびセキュリティが利用できます。

通常、グレードは A〜F スケールで割り当てられます。

Indexes カテゴリの場合、可能なグレードは次のとおりです。

- Excellent (A) インデックスを調整するための推奨事項が生成されなかった場合
- 生成された推奨事項の数に応じて、良い B、Fair、C、および悪い D

インデックスの場合、スコアは、PSR の場合はプライマリノード、PGD / AHA の場合は書き込みリーダーのみを考慮して計算されます。さらに、スコアは、最もパフォーマンスの悪い PG クラスター内のデータベースに関連しており、このようなクラスターで作成されたすべてのデータベースの平均ではありません。

統計カテゴリの場合、可能なグレードは次のとおりです。

- Excellent (A) 統計を追加するための推奨事項が生成されなかった場合
- 良好 B 少なくとも 1 つの推奨事項が生成された場合

Configuration カテゴリの場合、可能なグレードは次のとおりです。

- Excellent (A) 構成パラメーターを調整するための推奨事項が生成されなかった場合
- 生成された推奨事項の数に応じて、良い B、Fair、C、悪い D、および Fail

セキュリティカテゴリの場合、可能なグレードは次のとおりです。

- Excellent (A) セキュリティを向上させるための推奨事項が生成されなかった場合
- 生成された推奨事項の数に応じて、良い B、Fair、C、悪い D、および Fail

ワークロードスコアカードのサポートは、後のリリースで追加される予定です。

> **注釈**
>
> “Unknown” U は、クラスターを評価するために必要な情報が利用できないことを意味します。

カテゴリを選択すると、表示されるテキストは、スコアが計算された方法に関するコンテキストを提供することに加えて、推奨事項の関連性を理解するのに役立ちます。

レーダーチャートには、すべてのカテゴリのクラスターのグローバルスコアが表示されます。

おすすめリスト

スコアカードの下には、スコアカードで選択したカテゴリに応じて、推奨事項のリストが表示されます。

- ステータス 推奨事項のステータス New、Ignored、Applied 。
- データベース名。
- ノード ID。
- サイズの増加インデックスの場合 推定インデックスサイズ。
- Absolute Benefit インデックスの場合このインデックスを適用することにより生み出される絶対的なコスト削減。
- 合計コストインデックスの場合このクエリを実行するための実行時間とプラン時間。
- コスト削減インデックスの場合 予想されるコスト削減の割合は、(abs_benefit/total_cost) * 100 として計算され、データベースで実行されたすべてのワークロードに相対しています。
- 重み統計の場合 統計を適用することの潜在的な利点の大まかな推定。
- 推奨時間 この推奨事項の作成時間。
- クエリ インデックスまたは統計を作成するクエリのテキスト。

列を表示/非表示にするには、歯車アイコンを使用します。

各行の CTA ボタンでは、次のことができます。

- 推奨事項を完了済みとしてマークします
- 推奨を無視する
- 推奨事項のテキストをコピーします

> **注釈**
>
> Web インターフェイスから推奨事項を適用することはできません。ターミナルまたは管理 UI でそれらを手動でコピーして実行する必要があります。

フィルタを使用して、データベース、ノード、またはステータスごとにデータを制限します。

推奨事項を理解する

推奨事項を適用/無視するものとしてマークします

applied オプションを使用して、推奨される変更を適用したことをハイブリッドマネージャー HM に伝えます。推奨事項を適用済みとしてマークすると、リストから消えます。ページが次にリロードされると、スコアが増加して変更を反映します。同じ推奨事項が再度表示される場合、これは変更が適用されないか、元に戻されたことを意味します。またスコアが下がってしまいます。

ignored オプションを使用して、推奨される変更を確認し、適用しないことにしたことを HM に伝えます。おすすめを無視するとしてマークを付けると、リストから消えます。次にページがリロードされると、スコアが増加します。同じ推奨事項を再度受け取った場合、それは無視され、再度表示されません。

このデータはどこから来たのですか?

インデックスと統計の推奨事項は、クエリアドバイザー拡張機能から取得されます。この拡張機能が監視対象データベースで有効になっていることを確認します。

構成とセキュリティの推奨事項は、Postgres 構成をソースとしています。

将来のリリースでは、履歴メトリックやクエリ診断を含む他のデータソースからレコメンデーションが生成されます。

Backup Management

ハイブリッドマネージャーの特定のクラスターの下の [Backups] タブは、その PostgreSQL クラスターのバックアップを管理するための集中ビューとコントロールパネルを提供します。このページは、ユーザーがバックアップライフサイクルを効率的かつ安全に追跡、検証、および管理するのに役立ちます。

目的

[Backups] タブは、選択したクラスターに関連付けられているすべてのバックアップアクティビティを提供するように設計されています。過去のバックアップの監視と新しいバックアップの開始の両方をサポートしています。運用およびサイト信頼性エンジニア SRE チームがデータ保護を保証し、必要に応じて復旧プロセスを促進するために特に役立ちます。

主な機能

- **バックアップステータスの監視**

バックアップが正常に完了したか失敗したか、タイムスタンプと期間を簡単に確認できます。

- **手動バックアップの開始** [Create Backup] ボタンを使用して、クラスターの即時バックアップをトリガーします。

- **バックアップメタデータの確認**

開始時間と停止時間、バックアップ方法ボリュームスナップショットなど、使用されるストレージ場所などの詳細な洞察を取得します。

- **バックアップからの復元**

復元アイコン⊠を使用して、特定のバックアップポイントからリカバリーを開始します。

- **バックアップ ID をコピーします**

コピーアイコン⧉を使用して、オートメーション、サポートケース、またはログで参照するためにバックアップ識別子をコピーします。

- **バックアップの削除**

削除アイコン🗑を使用して、不要になったバックアップを手動で削除します。

[backups] タブでは、次の列を含むバックアップジョブの詳細情報を提供します。

Column	Description
Status	バックアップが正常に完了した Completed、失敗、またはまだ実行中であることを示します。
Name	バックアップジョブのシステム生成の名前。
Start Time / Stop Time	バックアップが開始および終了した時刻を示すタイムスタンプ。
Duration	バックアップが完了するまでにかかった時間。
Location	バックアップの保存先たとえば managed-devpatcher。
Method	バックアップの取得に使用されるテクノロジーたとえば、 VolumeSnapshot 。

ユーザビリティ

このタブは、次の場合に使用します。

- 自動またはスケジュールされたバックアップの正常な完了の確認
- メジャーアップグレードまたはメンテナンスタスクの前にバックアップを実行する
- データベースを以前の既知の良好な状態に復元する
- 履歴バックアップアクティビティの監査

Cluster Logs

ハイブリッドマネージャーで選択したクラスターの Logs タブは、クラスターの内部操作、システムイベント、および拡張レベルのアクティビティに関するリアルタイムおよび履歴の洞察を提供します。これは、クラスター内の個々のポッドの状態をデバッグ、診断、および監視するための重要なツールです。

目的

[Logs] タブは、ユーザーがクラスター内の各ポッドのイベント、警告、およびシステムアクティビティをトレースできるようにする集中ロギングインターフェイスとして機能します。データベース管理者や Kubernetes オペレーターが、誤った構成の拡張機能、ハートビート障害、欠落コンポーネントなどの運用上の問題を迅速に特定するのに役立ちます。

主な機能

- **ポッドレベルのログの表示**ポッドごとにログを表示するため、クラスターの特定の部分に影響を与える問題を簡単に分離できます。
- **時間範囲によるフィルタリング**

日付セレクターを使用して、特定の時間フレーム内のログをフィルタリングし、インシデント調査中の焦点を向上させます。

- **イベントの検索とフィルタリング**イベントタイプまたはキーワードでフィルタリングすることにより、関連するログエントリを簡単に見つけます。
- **ログレベルインジケーター**

Info や Warn などのラベルを介して、各イベントの重大度をすばやく評価します。

- **エラーの詳細な説明**

ログには、欠落している拡張機能や失敗したクエリなどの詳細なエラーメッセージが含まれており、根本原因の分析に役立ちます。

[logs] タブには、次のクラウドを含むログの詳細情報が表示されます。

Column	Description
Timestamp	ログイベントが発生した時 UTC を示します。
Pod	イベントが発生したポッド。
Event	拡張エラーやハートビートチェックなどのイベントの簡単な説明。イベントの情報、警告などの重大度ラベルのレベルも提供します。

ユーザビリティ

- モニタリング拡張機能 query_advisor、edb_stat_monitor などがアクティブであり、正しく構成されているかどうかを確認します。
- 欠落または非機能のデータベース拡張機能に関連するエラーの調査。
- ポッドのハートビートなどの定期的なシステムレベルのイベントの追跡。
- クラスター動作の最近の変更または異常の監査。

Cluster Properties

[**プロパティ**] タブは、選択した PostgreSQL クラスターの構成設定とリソース割り当ての包括的な概要を提供します。このタブは、スケーリング、編集、トラブルシューティングなどのクラスター操作を実行する前に、技術仕様を確認したり、セットアップパラメーターを検証したりする必要があるユーザーに重要です。

目的

このタブは、クラスターに関連するすべての静的プロパティの単一参照ポイントとして機能します。コンピューティングリソース、ストレージ構成、ネットワークアクセス、PostgreSQL バージョンなど。データベース管理者、サポートエンジニア、および開発者に役立ちます。

- クラスターの容量と制限を理解する
- 環境がワークロード要件を満たしていることを確認します
- コンプライアンスと監査チェックの促進
- 構成の不一致または展開エラーを特定する

[プロパティ] タブでは、次のプロパティを使用してクラスター構成設定に関する詳細情報を提供します。

Property	Description
Cluster Type	クラスターがシングルノードまたは高可用性 HA であるかどうかを示します。
Instances	クラスターにデプロイされた PostgreSQL インスタンスの数。
PgType	データベースタイプ PostgreSQL などを指定します。
PostgreSQL Version	現在クラスターにインストールされている PostgreSQL のバージョン。
Image	クラスターのプロビジョニングに使用されるコンテナまたは VM イメージタグ。
CPU Cores	クラスターに割り当てられた仮想 CPU コアの合計数。
Memory	クラスターに割り当てられた合計メモリ。
Primary Storage Class	永続ボリューム要求によって使用される基になるストレージクラスたとえば *gp3* 。
Primary Storage Size	プライマリインスタンスに接続されているストレージボリュームのサイズ。
Network Access	クラスターがパブリックまたはプライベートにアクセスできるかどうかを示します。
Connections	接続モード例 *Read/write* について説明します。
Retention Period	バックアップとログが保持される期間たとえば、30 日。

追加機能

- **Edit Cluster** 「Edit Cluster」リンクをクリックすると、ユーザーはリソースの割り当てやネットワーク構成などの編集可能なクラスタープロパティを変更できます。一部のプロパティでは、クラスターの再起動または再プロビジョニングが必要になる場合があります。
- **クイックアクション**スケーリングや削除など、クラスターに関連する状況依存のオプションにアクセスします。

Health assessment reports

状態評価レポートは、特定の期間における PG クラスターノードのパフォーマンスと状態のポイントインタイムのスナップショットを提供します。これらを使用して、異常を特定し、システムが時間の経過とともにどのように進化するかを追跡し、一貫したアーティファクトを他の人と共有できます。

レポートは自動的に生成されます。クラスターがハイブリッドマネージャー HM に接続された直後にノードごとに 1 つのレポート、その後 24 時間ごとに 1 回。監視ノードを含むすべてのノードが追跡されます。レポートのスコープはクラスターまたはデータグループに限定されません。レポートは永続的に保存されるため、レポートの生成に使用された基になるデータがパージされた後でも利用可能なままです。

レポート内容

各レポートは固定時間ウィンドウをカバーし、セッション数、トランザクション統計、待機イベント、上位の SQL ステートメント、WAL、バッファメトリックなどのその期間の集計統計が含まれています。

レポートの表示とダウンロード

レポートにアクセスするには

1. HM コンソールで、 **Reports** を選択します。

レポートページには、不動産全体のすべてのレポートが日付でソートされてリストされます。プロジェクト、クラスター、日付範囲、およびレポート間隔でフィルタリングできます。

2. レポートをダウンロードするには、レポート行を選択し、 **Download Report** を選択します。レポートは PDF ファイルで入手できます。

クラスターの詳細ページから単一のクラスターにスコープされたレポートを表示することもできます。

2 つのレポートの比較

同じクラスターの 2 つのレポートを比較するように Chat Agent に依頼できます。これは、2 つのスナップショット間の変更を強調表示します。時間の経過とともにパフォーマンスがどのように変化したかを追跡するのに役立ちます。

チャットエージェントを使用してレポートを比較するには

1. HM で **Chat Agent** を開きます。
2. 特定のノードとクラスターのレポートを比較するようにチャットエージェントに依頼します。例**「クラスター *X* のプライマリノードの最後の *2* つのレポートを比較します。」**

Chat Agent は、2 つのレポートの違いの概要を結果ます。

> **注釈**
>
> チャットボット比較機能を使用するには、HM でチャットエージェントを有効にし、モデルを少なくともラージ L モデルとして構成する必要があります。手順については、 *EDB Hybrid Manager Chat Agent* を参照してください。

5.3.3.3 Upgrading database clusters in Hybrid Manager

アップグレードは、Postgres データベースのメジャーまたはマイナーアップグレードの両方にとって、データベース管理の重要な部分です。

- *Postgres major version upgrades*
- *Postgres minor version upgrades in Hybrid Manager*

> **Tip**
>
> **Hybrid Manager インストール**をアップグレードする場合は、 *Upgrading Hybrid Manager* を参照してください。

5.3.3.3.1 Postgres minor version upgrades in Hybrid Manager

マイナーバージョンのアップグレードは、すべてのクラスタータイプと Postgres データベースディストリビューション（コミュニティ PostgreSQL、EDB Postgres Extended Server、および EDB Postgres Advanced Server を含む）で同じ手順を使用します。

マイナーバージョンリリースには、バグ修正、セキュリティパッチ、および下位互換性の向上が含まれます。これらは通常、少なくとも 3 か月に 1 回発行されます。

Postgres のマイナーリリースは、新機能や重大な変更を導入しないため、リリースされたらすぐに採用することをお勧めします。Postgres クラスターが複数のマイナーバージョンを実行している場合でも、利用可能な最新のマイナーバージョンを採用することをお勧めします。

Hybrid ManagerHM で新しいバージョンが利用可能になる時期を完全に制御できます。検証および公開されると、新しいバージョンのイメージが HM のイメージライブラリで利用できるようになり、既存のクラスターでアップグレードに利用できます。

> **注釈**
>
> Postgres のマイナーバージョンのアップグレードには、データの再起動が必要です。最小限のダウンタイムが予想されます。

マイナーアップグレードは頻繁に行われ、リスクが低いのが理想ですが、ベストプラクティスは、アップグレードを続行する前に、常に最新のバックアップと復旧計画を実施することです。

開始するための前提条件は、HM が起動し、正常な状態で作成されたクラスターがあることです。

クラスターが正常な状態にあることを確認するには、クラスターのメインページから **Monitoring** タブを選択します。クラスターとデータベースのメトリックを確認します。すべてが動作しており、すべてのメトリックが定義された許容可能なパラメーターの範囲内にあることを確認します。

HM で作成した既存のインスタンス/クラスターの Postgres マイナーバージョンをアップグレードする

1. クラスターのメインページから、 **Quick Action > Upgrade Cluster** を選択します。次に、 **マイナーアップグレード** を選択します。
2. 優先する新しいマイナーバージョンのイメージを選択します。
3. アップグレードパスを確認します。確認するには、[**Continue**] を選択します。
4. アップグレードで予想される特定のパッケージの変更を確認します。すべてが正しい場合は、**Upgrade Cluster** を選択します。

Monitoring および **Logs** タブを使用して、プロセスをモニタし、予期しない動作のトラブルシューティングを行います。

> **注釈**
>
> まれに、特定の製品リリースのマイナーアップグレードが完了した後、追加の手順が推奨される場合があります。 リリースノートを参照して、あなたの場合にこれが必要かどうかを確認します。

ロールバックの考慮事項

このアプローチでは、自動ロールバックは利用できないため、ダウングレードの実行はお勧めしません。マイナーアップグレードを元に戻すには、優先マイナーバージョンを実行する新しいクラスターで最新のバックアップから復元し、アプリケーションのワークロードを新しいクラスターに切り替えます。次に、Postgres のマイナーアップグレードされたクラスターを削除するか、さらにトラブルシューティングを行うことができます。

5.3.3.3.2 Postgres major version upgrades

メジャーバージョンのアップグレードにより、PostgreSQL の最新バージョンで導入された新機能、機能強化、およびパフォーマンスの向上にアクセスできます。ただし、下位互換性のない変更が含まれる場合があるため、注意深い計画と検証が必要です。

メジャーバージョンのアップグレードは通常毎年発生し、新しい SQL 機能、パフォーマンスの強化、最新の標準との互換性の向上などの重要な進歩をもたらす可能性があります。

スムーズなアップグレードプロセスを保証には、新しいバージョンに対してアプリケーションと拡張機能をテストし、非推奨または変更された機能のリリースノートを確認することが重要です。

Hybrid ManagerHM を使用すると、メジャーバージョンのアップグレードを自分の裁量で管理できます。新しいメジャーバージョンが検証および公開されると、HM のイメージライブラリですぐに利用できるようになり、既存のクラスターの制御されたアップグレードが有効になります。

> **注釈**
>
> Postgres のメジャーバージョンのアップグレードには、データの再起動が必要です。ダウンタイムが予想されます。

続行する前に、クラスターの完全なバックアップを実行し、包括的な復旧計画を検証することを強くお勧めします。これにより、アップグレードプロセス中に互換性または操作の問題が発生した場合に、データを復元できます。

データベースクラスタータイプ別のメジャーバージョンのアップグレード

サポートされているクラスタータイプで Postgres メジャーバージョンをアップグレードできます。

- インプレース、オフライン *upgrade procedure* を使用して、シングルノードの高可用性クラスターをアップグレードできます。
- 高度な高可用性 AHA および分散高可用性 DHA クラスターは、インプレース、オンライン *upgrade procedure* をサポートします。
- また、DHA クラスターは、 *blue-green deployment approach* を使用したゼロダウンタイムのメジャーアップグレードもサポートしています。

ロールバックの考慮事項

メジャーバージョンのアップグレードでは自動ロールバックは利用できません。ダウングレードの実行はお勧めしません。メジャーアップグレードを元に戻すには、優先メジャーバージョンを実行している新しいクラスターで最新のバックアップから復元し、アプリケーションのワークロードを新しいクラスターに切り替えます。次に、Postgres メジャーアップグレードされたクラスターを削除するか、さらにトラブルシューティングを行うことができます。

Postgres major version upgrades on Hybrid Manager single-node or high-availability clusters

前提条件

- ハイブリッドマネージャー HM で作成されたクラスターが起動し、正常な状態にあることを確認します。クラスターが正常な状態にあることを確認するには、クラスターのメインページから **Monitoring** タブを選択します。クラスターとデータベースのメトリックを確認します。すべてが動作しており、すべてのメトリックが定義された許容可能なパラメーターの範囲内にあることを確認します。
- 続行する前に、クラスターの完全なバックアップを実行し、包括的な復旧計画を検証することを強くお勧めします。

HM で作成した既存のクラスターの Postgres メジャーバージョンをアップグレードする

1. クラスターのメインページから、 **Quick Actions > Upgrade Cluster** を選択します。次に、 **メジャーアップグレード** を選択します。
2. 優先する新しいメジャーバージョンのイメージを選択します。
3. アップグレードパスを確認します。 **続行** を選択します。
4. アップグレードで予想される特定のパッケージの変更を確認します。すべてが正しい場合は、**Upgrade Cluster** を選択します。
5. **Monitoring** および **Logs** タブを使用して、プロセスを監視し、予期しない動作のトラブルシューティングを行います。

> **注釈**
>
> まれに、特定の製品リリースのメジャーアップグレードが完了した後、追加の手順が推奨される場合があります。 リリースノートを参照して、あなたの場合にこれが必要かどうかを確認します。

Postgres major version upgrades on Hybrid Manager for advanced high availability or distributed high availability

アップグレードの考慮事項

- **アップグレードスキップ**直接メジャーバージョンスキップはサポートされていません。すべてのアップグレードは、一度に 1 つのメジャーバージョンごとにシークエンシャルに進行する必要があります。

正解 16 > 17 > 18

不正解 16 > 18

- **画像タグの一貫性**画像タグの一貫性を維持します。たとえば、`<old-image-version>-<distribution>_full`から`<new-image-version>-<distribution>_full`にアップグレードします。これを行わないと、パッケージ化された拡張機能のセットが変更される場合があります。
- **物理的結合**高度な高可用性 AHA および分散高可用性 DHA クラスターは、日々のデータ同期に論理レプリケーションを使用しますが、メジャーバージョンのアップグレードのプロセスには**物理的結合**が含まれます物理レプリケーションデータグループ間のバージョンの変更を容易にします。これにより、データグループをインプレースでアップグレードでき、一時的に別のメジャーバージョンを実行できます。ただし、メジャーバージョンが異なる新しいデータグループを追加することはできません。

前提条件

- ハイブリッドマネージャー HM で作成されたクラスターが起動し、正常な状態にあることを確認します。クラスターが正常な状態にあることを確認するには、クラスターのメインページから **Monitoring** タブを選択します。クラスターとデータベースのメトリックを確認します。すべてが動作しており、すべてのメトリックが定義された許容可能なパラメーターの範囲内にあることを確認します。
- 続行する前に、クラスターの完全なバックアップを実行し、包括的な復旧計画を検証することを強くお勧めします。

HM で作成した既存のクラスターの Postgres メジャーバージョンをアップグレードする

AHA または DHA クラスター複数のデータグループを持つクラスターのメジャーアップグレードの場合、すべてのデータと監視グループを順番にアップグレードする必要があります。グループの順序は重要ではありませんが、次のグループを開始する前に、各グループのアップグレードが完全に完了していることを確認します。

1. クラスターのメインページで、アップグレードする最初のデータグループの横にある省略記号メニューから **Upgrade Data Group** を選択します。次に、**メジャーアップグレード** を選択します。
2. 優先する新しいメジャーバージョンのイメージを選択します。
3. アップグレードパスを確認します。 **続行** を選択します。
4. アップグレードで予想される特定のパッケージの変更を確認します。すべてが正しい場合は、**Upgrade Cluster** を選択します。

5. アップグレードが適用された後、PGD CLI を使用して、データグループが正常な状態になることを確認できます。すべてのチェックのステータスが `Ok`

   であれば、アップグレードは成功です。

6. 最初のデータグループが正常にアップグレードされ正常になったら、クラスター全体が新しいバージョンになるまで、残りのデータグループと監視グループを 1 つずつアップグレードします。

> **注釈**
>
> まれに、特定の製品リリースのメジャーアップグレードが完了した後、追加の手順が推奨される場合があります。 リリースノートを参照して、あなたの場合にこれが必要かどうかを確認します。

Seamless major upgrades with blue-green deployments

メジャーアップグレードのブルーグリーンアプローチ

分散高可用性 DHA クラスターを使用すると、信頼性が高く安全な即時のメジャーアップグレードが可能です。この方法は、さまざまなデータグループに基づいて、分散データベースに blue-green deployment アプローチを使用することで構成されています。

- 既存の *blue* データグループは、現在の安定した環境たとえば、運用データベースを表します。
- 新しい *green* データグループがデータベースクラスターに作成され、次のメジャーバージョンにアップグレードされます。
- green データグループに接続することにより、アプリケーションで包括的なテストを実行します。新しい接続文字列を使用できます。
- すべてのクライアントワークロードをグリーンデータグループにリダイレクトします。
- アップグレードが完全に検証された後、青いデータグループを削除します。

前提条件

- ハイブリッドマネージャー HM 上の古いバージョンの青のデータグループを含む DHA クラスターが正常な状態で稼働していること。
- アセットライブラリで利用可能な次の Postgres メジャーバージョンを用意します。

一般的な推奨事項

- アップグレードプロセスを開始する前に、常に完全なバックアップを実行します。
- 可能であれば、実稼働前環境でアップグレードプロセスをテストします。
- アップグレードプロセスは短期間にたとえば同日に完了することをお勧めします。データグループで異なるメジャーバージョンを実行すると、予期しない非互換性の問題が発生する可能性があります。

クラスターのステータスを確認する

クラスターが正常な状態にあることを確認するには、クラスターのメインページから **Health Status** タブを選択します。次に、クラスターを確認します。

- **ラフトステータス** 。 **OK** と表示されていることを確認します。
- **レプリケーションスロットのステータス** 。 **OK** と表示されていることを確認します。
- **クロックスキュー** 。 **OK** と表示されていることを確認します。
- **プロキシステータス**。すべてのプロキシが起動していることを確認します。
- **ノードステータス** 。すべてのノードが起動していることを確認します。
- **トランザクションレート**。許容/予想されるパラメーターの範囲内であることを確認します。

次を含む各データグループとデータグループ内のノードに関する詳細を調査できます。

- データグループの **プロキシステータス**

- データグループの **ノードステータス**
- グループ内の各ノードの以下の詳細
  - 接続数
  - WAL サイズ
  - 使用メモリの割合
  - 使用ストレージの割合

最新の **レプリケーションステータス**の詳細を確認して、ノード間のレプリケーション遅延を確認できます。

すべてのデータが予想通りに同期されていることを確認するには、PGD CLI を使用して view data group state を実行するか、レプリケーションピアで deeper monitoring を実行できます。

これらのメトリックのいずれかが許容可能な/予想されるパラメーターの範囲外である場合、アップグレードプロセスを続行する前にこれらの問題に対処してください。

クラスターを編集する

アップグレードプロセスを開始するには、右上隅の **Quick Actions** メニューから **Edit Cluster** を選択します。次に

1. [**クラスター設定**] タブを選択し、パスワードを再度設定します。
2. [**Data Groups**] タブを選択し、古いバージョンの青色のデータグループを見つけて、**Duplicate** を選択します。これにより、既に持っているデータグループの下に新しいデータグループが表示されます。

グループの名前をデフォルトから変更し、 **ノード**アーキテクチャ 2〜3 ノードまたは 3〜2 ノード、および **展開場所** 、**データベースタイプ** 、を変更できます。 **インスタンスサイズ** 、**ストレージ** 、 **ネットワーク** 、**バックアップ** 、および**セキュリティ**設定。ただし、このアップグレードプロセスを使用するほとんどの場合、古いバージョンのデータグループを、同じ負荷を処理できる別の新しいバージョンのデータグループに置き換えようとしているため、**グループ名** を除き、すべてを同じにしておきます。。ただし、特定の状況で何かを変更する必要がある場合は、アップグレードプロセスのこの時点でそれを行うことができます。

3. **Create Data Group** を選択します。奇数のデータグループのみがある場合、デフォルト名 **Witness Group** を持つ監視グループがデータグループのセットに追加されます。Witness グループは、Raft コンセンサスを可能に保つために役立ちます。

偶数のデータグループを使用すると、いわゆる「スプリットブレイン」シナリオが発生する可能性があるため、これは 2 つのデータグループで必要です。この場合、グループの半分は 1 つの決定をしたいと考えており、残りの半分は別のことをしたいと考えています。監視グループは、これが発生するたびにタイブレーカーになることにより、このようなシナリオを防ぎます。

4. **Save** を選択して、監視グループの構成を保存します。

[**データグループ**] タブには、古い青のデータグループ、新しい緑のデータグループ、および監視グループの両方が表示されます。

5. [**Data Groups**] タブで、[**Save**] を選択します。

確認の変更ポップアップは、クラスターに行う変更と、変更を有効にするとクラスター全体でデータベースの再起動がトリガーされる可能性があることを示す警告をリストします。これは事実ですが、変更の適

用によりダウンタイムが発生することを意味するものではありません。この間、古いバージョンのデータグループの少なくとも 1 つのノードはアクティブなままです。

6. **confirm Changes** を選択します。基になるハードウェアとデータベースのサイズによっては、変更にかなりの時間がかかる場合があります。

緑のデータグループのデータを検証する

クラスター全体が クラスターのステータスを確認する にあることを確認します。

[**概要**] タブから、新しいデータグループの接続文字列をコピーできます。各データグループには異なる接続文字列が公開されるため、既存のデータグループに接続しているアプリケーションは、プロセスが完了するまで影響を受けません。新しい接続文字列を使用して、すべてのデータが同期されていることを検証するなどのテストを実行できます。

green データグループを次のメジャーバージョンにアップグレードする

1. **Overview** タブから、新しいデータグループを特定します。省略記号メニューから、**Upgrade data group** を選択します。
2. **Software Image** リストから、ターゲットバージョンを選択します。次のメジャーバージョンで使用可能な最新のマイナーにアップグレードすることをお勧めします。
3. **Continue** を選択します。アップグレードの詳細が表示されます。
4. **Upgrade Cluster** を選択して、データグループのアップグレードを開始します。プロセスが完了するまで数分待ちます。

アップグレードの確認

Overview および **Properties** タブには、データグループの新しいバージョンが表示されます。クラスターが クラスターのステータスを確認する にあることを確認します。

テストを計画する

ブルーの環境から切り替える前に、新しいアップグレードバージョンを使用してアプリケーションとクライアントを厳密にテストすることを強くお勧めします。

テストを実行するために書き込み操作またはスキーマの変更が必要な場合、それらは古いバージョンのデータグループにレプリケートされます。これを管理するには、enabling or disabling the replication subscriptions によってクラスターのレプリケーション構成を変更できます。

たとえば、次の SQL コマンドを使用して、新しいアップグレードバージョンのデータグループから古いバージョンのデータグループへの変更を無効にできます。古いバージョンのデータグループのノードに接続したときに実行する必要がある SQL コマンドは、新しいデータグループから変更をレプリケートするサブスクリプションを無効にします。

```
SELECT bdr.alter_subscription_disable(sub_name, true, true)
    FROM bdr.subscription s, bdr.node n, bdr.node_group ng
  WHERE n.node_group_id = ng.node_group_id
```

(次のページに続く)

(前のページからの続き)

```
    AND s.source_node_id = n.node_id
    AND ng.node_group_name = <new version data group name>;
```

スイッチオーバーを計画する

すべてのテストを実行したら、新しい接続文字列を使用してワークロードをグリーンデータグループに切り替える準備が整います。これには、アプリケーション、クライアント、またはインフラストラクチャサービスの変更が必要になる場合があります。

新しいバージョンと古いバージョンのデータグループ間の双方向同期を以前に無効にした場合、再度有効にする必要があります。

```
SELECT bdr.alter_subscription_enable(sub_name, true)
    FROM bdr.subscription s, bdr.node n, bdr.node_group ng
  WHERE n.node_group_id = ng.node_group_id
    AND s.source_node_id = n.node_id
    AND ng.node_group_name = <old version data group name>;
```

クラスターが クラスターのステータスを確認する にあることを確認します。

青いデータグループを削除します

アップグレード手順の最後のステップは、古いバージョンのデータグループを削除することです。クラスターページの**概要**タブで、削除する古いバージョンのデータグループの省略記号メニューから、**Delete Data Group** を選択します。

削除を確認します。持っている他の古いバージョンのデータグループまたは監視データグループが必要ない場合は、削除します。これでアップグレードは完了します。

クラスターが クラスターのステータスを確認する にあることを確認します。

ロールバックの考慮事項

このアプローチでは、古いバージョンのデータグループを削除した後、自動ロールバックは利用できません。メジャーアップグレードを元に戻すには、目的のメジャーバージョンを実行する新しいデータグループで最新のバックアップを復元し、アプリケーションとクライアントのワークロードをその新しいデータグループに切り替えることを検討します。

5.3.3.4 Migrating databases with Hybrid Manager

Hybrid ManagerHM は、外部データベースからお客様の環境への移行を実行するための統合ソリューションを提供します。この機能により、既存のデータ、スキーマ、アプリケーションを一元的な管理ハブに取り込むことができ、管理と監視が簡単になります。

データの移行、移行プロセスは主に EDB Data Migration ServiceDMS を活用します。移行ポータルや AI コパイロットなどの他の統合と連携して、データベースの評価、変換、移行のための包括的なワークフローを提供します。

5.3.3.4.1 HM で移行機能を使用するにはどうすればよいですか?

ソースデータベースと宛先データベースを HM コンソールに接続する一連の手順を使用して、移行プロセスを実行します。ワークフローを理解するには、*Agent Factory Architecture on Hybrid Manager* を参照してください。

5.3.3.4.2 EDB Data Migration Service はどのような種類の移行をサポートしていますか？

EDB DMS は、外部データベースから HM 管理データベースへ、外部データベースから他の外部データベースへ、HM 管理データベースから外部データベースへの移行をサポートします。考えられる移行パスのエンドツーエンドの例については、*Choosing your migration path* を参照してください。

サポートされているソースおよび宛先データベースについては、*Supported databases and operating systems* を参照してください。

サポートされている移行方法については、 **移行タイプ** を参照してください。

5.3.3.4.3 EDB Data Migration Service はどのようなテクノロジーを使用しますか?

DMS は、Apache Kafka とオープンソースの Debezium CDC プラットフォームに基づいて構築されています。変更データキャプチャ CDC とイベントストリーミングを使用して、ソースデータベースの行の変更が移行先にレプリケートされます。

Planning

Migration architecture

Hybrid ManagerHM は、さまざまなソースから HM 管理および外部管理の Postgres クラスターにデータベースを移行するための統合プラットフォームを提供します。移行プロセスは、スキーマ変換、データ転送、および継続的レプリケーションを処理するために連携するいくつかの主要コンポーネントに依存しています。

Oracle から

図 5: Migrations from Oracle databases

Oracle からの移行は、EDB Postgres Advanced Server クラスター HM 管理または外部管理をターゲットとするのに最適です。2 つのデータベースシステムの違いを考慮すると、これらの移行には、スキーマとデータの移行の両方を処理するための特定のコンポーネントのセットが必要です。EDB EDB Postgres Advanced Server は、この目的に最適化されています。

Oracle データベース移行するデータとスキーマを含むソースデータベース。

エージェント Oracle ソースに接続して、移行のスキーマを読み取りおよび抽出するコンポーネント。また、ソースを HM に登録すると、移行評価、およびスキーマの移行を有効にするために HM コンソールにソースデータベースのビューが提供されます。

リーダーモードの DMS エージェント Oracle ソースに接続して、移行用のデータを読み取りおよび抽出するコンポーネント。

ハイブリッドマネージャー HM 中央コントロールプレーン。

HM コンソールユーザーがすべての移行アクティビティを管理および監視する Web ベースのインターフェイス。

移行ポータル EDB Postgres Advanced Server との互換性を保証しながら、Oracle 固有のスキーマ評価、変換、および宛先データベースへのインポートを処理する HM のサービス。AI コパイロットを使用して、スキーマ関連の問題を支援することもできます。

データ移行サービス DMS 実際のデータ転送スナップショットと継続的変更レプリケーションストリーミングを実行するコアサービス。これには、HM 管理データベースへの移行のためのビルトインライタが含まれています。

HM 管理 EDB Postgres Advanced Server データベース HM によって作成および管理される宛先データベースクラスター。

ライタモードの DMS エージェント移行先データベースが HM の外部の自己管理データベースである場合にのみ必要なコンポーネント。CDC レコードを消費して、宛先データベースに変更を複製および適用します。

外部 EDB Postgres Advanced Server データベース自己管理データベースとも呼ばれる、移行の外部管理の宛先。このシナリオでは、宛先ホストに接続してデータを受信および適用するライタモードの追加の DMS エージェントコンポーネントが必要です。

移行を実行する方法を示すエンドツーエンドの例については、Oracle to HM-managed database を参照してください。

Postgres から

図 6: Migrations from Postgres databases

スキーマには既に互換性があるため、Postgres データベース間の移行はより合理化されたプロセスです。この移行は、主に効率的なデータ転送とレプリケーションに焦点を当てており、移行には特定のコンポーネントのセットが必要です。

外部 Postgres データベース移行するデータを含むソース Postgres データベース。

Migration Toolkit または pg_dump: スキーマ自分自身は変換を必要としないため、初期スキーマの移行に使用される外部ツール。

エージェントソースを HM に登録して、モニタリングのために HM コンソール内のソースデータベースのビューを提供するコンポーネント。

リーダーモードの DMS エージェントソース Postgres に接続してデータを読み取りおよび抽出するコンポーネント。

ハイブリッドマネージャー HM 中央コントロールプレーン。

HM コンソールユーザーがすべての移行アクティビティを管理および監視する Web ベースのインターフェイス。

データ移行サービス DMS 実際のデータ転送スナップショットと継続的変更レプリケーションストリーミングを実行するコアサービス。これには、HM 管理データベースへの移行のためのビルトインライタが含まれています。

HM 管理の Postgres データベース HM によって作成および管理される宛先データベースクラスター。 PostgreSQL、EDB Postgres Advanced Server、または EDB Postgres Extended Server です。

ライタモードの DMS エージェント移行先データベースが HM の外部の自己管理データベースである場合にのみ必要なコンポーネント。CDC レコードを消費して、宛先データベースに変更を複製および適用します。

外部 Postgres データベース自己管理データベースとも呼ばれる、移行の外部管理の宛先。このシナリオでは、宛先ホストに接続してデータを受信および適用する書き込みモードの追加の DMS エージェントコンポーネントが必要です。

移行を実行する方法を示すエンドツーエンドの例については、Self-managed Postgres to HM-managed database を参照してください。

Terminology

この用語は、EDB Data Migration Service DMS 機能を理解するために重要です。

変更データキャプチャ CDC

CDC は、_deltas_を計算することによりデータセットの変更を決定および追跡するために使用されるソフトウェアデザインパターンのセットです。EDB DMS は CDC を使用して、データベースクラスターから別のデータベースクラスターに変更をストリーミングします。

EDB DMS エージェント

DMS エージェントは、移行のためにデータベースを登録できるツールです。リーダーモードでは、データベースを移行のソースとして登録します。ライタモードでは、宛先として登録します。

ソースデータベースと宛先データベースを登録すると、DMS エージェントは DMS と連携してデータ移行を実行します。

> **注釈**
>
> Hybrid ManagerHM バージョン 1.3 以降では、 *cdcagent* としてパッケージ化された *Installing EDB Data Migration Service agent on Linux* が、古い DMS Reader および DMS Writer を置き換えます。

互換性を保証には、すべての HM 1.3 以降の展開で DMS エージェントを使用します。バージョン 1.2 以前の HM 展開の場合、個別にパッケージされた EDB DMS Reader および EDB DMS Writer を使用します。

リーダーモードの EDB DMS エージェント

リーダーモードで実行されている場合、DMS エージェントは Debezium を使用してソースデータベースと Kafka /オブジェクトストレージで CDC 操作を実行し、変更イベントを含むメッセージを生成します。

ソースマシンに DMS エージェントをインストールします。また、ソースデータベースが、管理アクセスがないクラウドプロバイダーによって管理されるマシンで実行されている場合、ソースデータベースへのアクセスがある補助マシンにインストールすることもできます。

次に、このモードでソースに接続して読み取ることができるため、あらゆる種類の移行を実行するときにリーダーモードで構成して起動します。

ライタモードの EDB DMS エージェント

ライタモードで実行されている場合、DMS エージェントは Kafka/Object ストレージから CDC レコードを消費して、宛先データベースに変更を複製および適用します。

移行の宛先が HM 管理データベースの場合、HM に付属の DMS コンポーネントには既にビルトインライタが含まれているため、ライタモードで EDB DMS エージェントをインストールして起動する必要はありません。

宛先が自己管理データベースの場合、宛先マシンに追加の DMS エージェントをインストールします。次に、ライタモードで構成して起動します。そのモードでは、自己管理の宛先に接続してデータを移行できます。

EDB Postgres AI エージェント

`beacon-agent` としてパッケージされ、以前は HM またはビーコンエージェントとして知られていた *HM で AWS RDS 展開をモニタするための EDB Postgres AI エージェントの設定*は、データベースの可観測性、モニタリング、使用状況レポートなどを提供するツールです。移行のコンテキストでは、EDB Postgres AI エージェントは、登録した各データベースの移行評価を実行します。また、後続のスキーマ移行のためのスキーマ抽出も実行します。Oracle ソースの場合、これは *Migration Portal 統合の機能強化* をフィードします。 Postgres ソースの場合、**スキーマとデータ** または **スキーマのみ**スコープを選択すると、統合スキーマ移行ワークフローが有効になります。

補助機

補助マシンは、データベース移行のために EDB DMS エージェントを実行する専用の Linux ベースのサーバーベアメタル、VM、またはローカルです。このマシンは移行エンジンとして機能し、ローカルまたはリモートにかかわらず、さまざまなソースおよび宛先データベースに接続できます。

このマシンは、ソースまたは宛先データベースがあるのと同じホスト、および HM が実行されているホストと同じにすることができます。唯一の要件は Linux オペレーティングシステムです。それが現在 EDB DMS エージェントをサポートしている唯一のプラットフォームであるためです。

移行タイプ

creating migrations の場合、 **Snapshot** または **Snapshot + Streaming** オプションを使用して移行を実行するかどうかを選択できます。

スナップショットの移行

Snapshot オプションを選択すると、DMS は特定の瞬間にデータベースのイメージをキャプチャします。このポイント・インタイム・コピーは、移行に使用されます。スナップショットが作成された後にソースデータベースで実行された新しい挿入と更新は、宛先データベースにレプリケートされません。

スナップショット+ストリーミングの移行

Snapshot + Streaming オプションを選択すると、DMS はスナップショットから開始してベースラインコピーを確立し、常に CDC を使用して後続の変更を適用します。このアプローチにより、移行中のデータの一貫性が保証されます。DMS は、*mark the migration as completed* を行うまで、ソースデータベースで実行されたデータ更新を宛先データベースにストリーミングし続けます。

移行スコープ

移行を作成するときに、移行にスキーマ、データ、またはその両方を含めるかどうかを決定するスコープを選択します。

データのみ

データのみを移行します。このオプションは、既にスキーマを宛先データベースに手動でインポートしている場合に使用します。スキーマは、移行を開始する前に設定する必要があります。

スキーマとデータ

単一の管理対象ワークフローでスキーマとデータを移行します。HM は、ソースデータベースの登録時に作成された移行ポータルプロジェクトからスキーマを自動的にインポートし、移行中に制約の順序を処理し、移行の完了後に遅延制約を再適用します。このスコープでは、ソースデータベースを EDB Postgres AI エージェントに登録する必要があります。

スキーマのみ

データを転送せずに、スキーマのみを移行します。データのみの移行を実行する前に、宛先データベース構造をセットアップするのに役立ちます。

移行先のタイプ

creating migrations の場合、管理対象 Postgres への移行を実行するか選択できます。

マネージド Postgres

管理対象 Postgres の宛先は、HM を介して作成および管理されるデータベースです。移行を実行しているプロジェクトで作成されたデータベースクラスターを移行先として選択できます。

外部 Postgres

外部 Postgres の宛先は、HM の外部で作成され、完全に管理するデータベースです。移行先として外部の Postgres データベースを選択するには、最初に *Configuring and running the EDB DMS agent* をインストールして構成し、ライタモードで構成して、DMS が自己管理データベースに接続し、書き込み操作を実行できるようにする必要があります。

Apache Kafka

Apache Kafka は、オープンソースの分散イベントストリーミングプラットフォームであり、高性能データパイプライン、ストリーミング分析、データ統合、およびミッションクリティカルアプリケーションに数千の企業が使用しています。EDB DMS は、Kafka を使用して、ソースから宛先データベースへのデータストリーミングを管理します。

Debezium

Debezium は、Red Hat コミュニティによってサポートされている CDC 用の Java ベースのオープンソースプラットフォームです。リーダーモードで実行されている場合、EDB DMS エージェントは Debezium を使用して読み取り操作を実行し、ソースデータベースでデータ変更をキャプチャします。

古い用語

EDB DMS リーダー

HM バージョン 1.2 以前では、 `cdcreader` としてパッケージ化された EDB Data Migration Service Reader は、Debezium を使用してソースデータベースで CDC 操作を実行し、Kafka を使用して変更イベントを含むメッセージを生成するツールです。

ソースに接続して読み取ることができるため、あらゆる種類の移行を実行するときに DMS Reader をインストールして構成します。移行の宛先が HM 管理データベースの場合、*EDB DMS Writer* をインストールする必要はありません。宛先が別の自己管理データベースの場合、*EDB DMS Writer* をインストールします。

EDB DMS Writer

HM バージョン 1.2 以前では、 `cdcwriter` としてパッケージ化された EDB Data Migration Service Writer は、CDC レコードを使用して宛先データベースに変更をレプリケートおよび適用するツールです。

自己管理の宛先データベースへの移行を実行するときに、宛先マシンに DMS Writer をインストールして構成します。自己管理の宛先に接続できるようになります。任意の種類の移行の場合、 *EDB DMS リーダー* もインストールします。

Migration features

Hybrid ManagerHM は、データベース移行のための包括的な一元化ソリューションを提供し、データ転送プロセスを簡素化および高速化するように設計された幅広い機能を提供します。

ソブリン移行ソリューション

オンプレミスまたはクラウド環境内に完全に展開された単一のプラットフォームから移行を実行します。このアプローチでは、データとインフラストラクチャを完全に制御でき、独自のネットワーク内での移行が主権的で安全であることが保証されます。

移行の評価

HM は、 EDB Postgres AI エージェントを活用して、Oracle データベースの移行に必要な複雑さの評価と推定作業量 LOE を生成します。登録されたソースデータベースを分析することにより、プラットフォームは潜在的な問題と移行の規模に関する早期の洞察を提供し、効果的な計画を支援します。

大量移行の評価

評価する大規模なデータベース資産がある場合、EDB Postgres AI エージェントの `configure-onprem-dbs` コマンドを使用すると、シングルステップで移行評価のために多くのデータベースを登録できます。データベースごとに接続文字列を手動で構成する代わりに、CSV ファイル、Oracle `tnsnames.ora` ファイルからインベントリをインポートしたり、AWS RDS から直接データベースを検出したりできます。 *Assessing multiple databases at once* を参照してください。

データベースとスキーマの管理

データベースの詳細ページからソースデータベースに関する詳細情報にアクセスします。そこでは、スキーマに関する情報と、移行評価要素の明確な内訳があります。

AI Copilot を使用した移行ポータル

移行ポータルは、Oracle から Postgres への移行のための強力なツールです。スキーマ、フラグ、非互換性の評価を自動化します。また、統合された AI コパイロットも提供し、複雑なスキーマの問題を迅速に特定して解決し、変換プロセスを合理化します。元々は EDB がホストする Web ツールでしたが、移行ポータルは HM に統合されました。この統合は、環境にサービスを展開し、展開の主権をサポートします。

柔軟なスキーマとデータ移行のオプション

移行スコープ

HM は、Oracle と Postgres の両方のソースのデータ移行ワークフローの一部として、自動スキーマの移行をサポートしています。移行を作成するときに、**スキーマとデータ** 、 **データのみ** 、または **スキーマのみ**を移行することを選択でき、転送されるものを完全に制御できます。

移行方法

ニーズに最適な移行方法を選択してください

- **スナップショット** — 大量移行のための高速なワンタイムデータ転送。
- **スナップショットとストリーミング** —最小限のダウンタイムを必要とする移行に適しています。変更データキャプチャ CDC を使用して、ソースから宛先データベースに進行中の変更を継続的にキャプチャおよびレプリケートします。

多様な移行先

このソリューションは、目的地を選択する際の柔軟性を提供します。HM でプロビジョニングされたデータベース HM 管理データベースまたは外部管理の Postgres データベース外部管理データベースに移行できます。

詳細な移行制御

移行する特定のデータベーステーブルと列を選択する機能を使用して、データの移行を正確に制御します。この機能を使用すると、必要なデータのみを移行できるため、移行時間とリソースの使用量を削減できます。

スキーマと列のマッピング

移行サービスは、柔軟なスキーママッピングを可能にし、宛先データベース内の別の名前を持つテーブルと列にデータを移行できます。この機能は、移行プロセス中にデータを再構成する場合に特に役立ちます。

Supported databases and operating systems

EDB Data Migration Service EDB DMS は、Hybrid ManagerHM の一部としてのみ利用できます。

サポートされているソースデータベース

次のデータベースバージョンがサポートされています。

Database version	Supported source
PostgreSQL 12 - 18	Y
EDB Postgres Advanced Server 12 - 18	Y
EDB Postgres Extended Server 12 - 18	Y
Oracle 19c	Y
Oracle 21c	Y
Oracle 23ai	Y
AWS RDS PostgreSQL 13-17	Y
AWS RDS Aurora (PostgreSQL compatible) 13-17	Y

サポートされている Postgres ソースデータベースには、外部管理自己管理データベースのみが含まれます。

注釈

DMS エージェントは、Oracle 21c および 23ai データベースの移行をサポートしています。ただし、Migration Portal は行いません。これは、 21c および 23ai データベースを移行するには、 スキーマの移行を手動で実行する必要があることを意味します。

Oracle

EDB DMS スタックでは、Oracle データベースでアーカイブログモードを有効にし、テーブルおよびデータベースレベルでサプリメンタルロギングデータを有効にする必要があります。詳細については、*Preparing Oracle source database* を参照してください。

コンテナデータベース CDB / PDB および非 CDB ソースがサポートされています。

Postgres

Postgres から Postgres へ移行する場合、クロスバージョンおよびクロスディストリビューションの移行がサポートされています。PostgreSQL 16 から PostgreSQL 17、または PostgreSQL 17 から EDB Postgres Advanced Server 17 など、PostgreSQL のあるバージョンから別のバージョンにデータを移行できます。例外は、ソースとして EDB Postgres Advanced Server から別のディストリビューション、たとえば、PostgreSQL または EDB Postgres Extended Server へ。

ソース Postgres インスタンスは、ベアメタルマシン、仮想マシン上、または AWS のようなクラウドサービスプロバイダー RDS または Aurora データベースでホストされる自己管理インスタンスにすることができます。

Postgres ソースには、レプリケーションを管理できるデータベースロールまたはユーザーが必要です。詳細については、*Preparing your AWS RDS/Aurora PostgreSQL source database* を参照してください。

サポートされている宛先データベース

DMS は、HM 管理データベース HM で作成されたクラスター上と Postgres を実行している自己管理データベースの両方への移行をサポートしています。

Database version	Supported destination
PostgreSQL 11 - 18	Y
EDB Postgres Advanced Server 11 - 18	Y
EDB Postgres Extended Server 11 - 18	Y

> **注釈**
>
> HM 管理データベースクラスターの場合、サポートはバージョン 16 および 17 のみです。

DMS は、表にリストされているすべてのデータベースへの移行をサポートしています。ただし、移行ポータルは、EDB Postgres Advanced Server バージョン 16 および 17 の場合のみスキーマの移行をサポートしています。バージョン 15 以前の場合、スキーマの移行を手動で実行する必要があります。

Oracle

EDB DMS は、Oracle 宛先への移行をまだサポートしていません。

Postgres

Postgres から Postgres に移行する場合、クロスバージョンおよびクロスディストリビューションの移行がサポートされています。たとえば、PostgreSQL 16 から 17、または PostgreSQL 17 から EDB Postgres Advanced Server 17 など。

Postgres の宛先には、レプリケーションを管理できるデータベースロールまたはユーザーが必要です。詳細については、*Preparing your HM-managed destination database* を参照してください。

サポートされているオペレーティングシステム

DMS エージェントは Linux で実行されます。詳細については、Installing EDB DMS agent を参照してください。

Known issues, limitations, and notes

これらは、次の既知の問題、制限、および注意事項です。

- **データ移行の制限**
- *Oracle* **の制限事項**
- *Postgres* **の制限**
- **スキーマの取り込みの既知の問題**
- *HM Migration Portal* **の既知の問題**

- その他の既知の問題

データ移行の制限

オブジェクト名の大文字と小文字の区別

> **Tip**
>
> HM バージョン 2025.11 以降で解決されました。

EDB DMS は、現在、大文字と小文字を区別する名前を持つスキーマ、テーブル、列の移行をサポートしていません。

アップグレードされた宛先クラスター

HM インスタンスをバージョン 1.3 以降にアップグレードした後、Hybrid ManagerHM バージョン 1.2 以前でプロビジョニングされたクラスターを移行先として使用することはできません。アップグレードした環境で新しいクラスターを作成して、有効な移行先として使用する必要があります。

単一の宛先への複数の移行の考慮事項と制限

DMS は、1 つのソースから 1 つの宛先データベースへの複数のデータ移行の実行をサポートしていますが、特定の制限が適用されます。

- **テーブルの使用法**各宛先テーブルは、一度に 1 つの移行でのみアクティブに使用できます。
- **一意のリソース ID 外部宛先への移行に適用**外部宛先へ複数の移行を実行する場合、`DBCONFIG_DATABASES_0__RESOURCEID=${Resource_ID}` パラメーターを使用してライタモードで EDB DMS エージェントの各インスタンスに一意のリソース ID を割り当てる必要があります。このパラメーターを設定すると、同時移行が同じテーブルを同時に使用しようとすることを防ぎます。これにより、データの損失または重複が発生する可能性があります。

> **注釈**
>
> この resourceID 要件は、HM 管理の Postgres インスタンスをターゲットとする移行には適用されません。 !!!

- **パフォーマンスプランニング外部宛先への移行に適用**一般的に *System specifications* でパフォーマンスは強力なままですが、同じ外部宛先データベースに 5 つ以上の移行を実行する場合、リソースの競合を防ぐために、追加の CPU、メモリの割り当てを検討します。、および宛先へのボリューム IOPS/スループット。これは、宛先が外部データベースの場合に特に重要です。
- **リソース管理**各移行ジョブでは、必要な DMS エージェントコンポーネントの専用の非同時使用が必要です。

リーダーモードの DMS エージェントソース) 同じソースデータベースからの複数の同時移行には、リーダーモードで EDB DMS エージェントの単一の実行インスタンスを使用しないでください。HM コンソールは、ソースデータベースが移行でアクティブになると、利用可能なリストから削除します。ただし、後

続の移行または別のソースデータベースからの同時移行には、リーダーモードで単一の DMS エージェントを使用できます。

ライタモードの DMS エージェント外部宛先移行が別の移行と同じ宛先を共有する場合でも、各移行にはライタモードの DMS エージェントの独自のインスタンスが必要です。

Postgres 固有のスキーマオブジェクトタイプは、スキーマの移行時に DMS によって移行されません

> **Tip**
>
> HM バージョン 2026.5 以降で解決されました。

説明 Postgres スキーマ移行のサポートは、テーブルのみを含む単純なデータベースに制限されています。DMS は現在一部のオブジェクトタイプを移行できず、オブジェクトが移行される順序に問題があります。

回避策スキーマを手動で移行し、データのみの移行モードで DMS を使用します。

Oracle の制限事項

サポートされていない Oracle データ型

限られた数の Oracle データ型と機能は、EDB DMS でサポートされていません。

サポートされているデータ型に関する詳細なコメントについては、Debezium documentation を参照してください。

サポートされていない Oracle データ型は次のとおりです。

- BFILE
- ロング
- LONG RAW
- RAW
- UROWID
- ユーザー定義型 REF、Varray、ネストされたテーブル
- ANYDATA
- XMLTYPE
- 空間

EDB DMS は、 BLOB、CLOB、または NCLOB 列を含む Oracle テーブルのレプリケート、これらにも主キー制約がある場合にのみサポートします。テーブルに主キー制約がない場合、ストリーミングレプリケーションは INSERT 操作のみをサポートします。

`Nan` 、`+INF` 、および`-INF` 値を含む可能性のある Oracle の `BINARY_FLOAT` および `BINARY_DOUBLE` タイプは、EDB DMS ではサポートされていません。

EDB DMS は、Oracle の `VARCHAR` や `CLOB` のようなテキストフィールドタイプのヌル文字 0x00 を含む値をサポートしていません。

Oracle 21c データベース

DMS エージェントは、Oracle 21c データベースの移行をサポートしています。ただし、Migration Portal は行いません。これは、 21c データベースを移行するには、スキーマの移行を手動で実行する必要があることを意味します。

Oracle XE の互換性制限

Oracle XE Express Edition は、データベースレベルでのサプリメンタルロギングの有効化をサポートしていません。これは変更データキャプチャ CDC のコア要件であるため、EDB DMS はソースデータベースとしての Oracle XE と完全な互換性がありません。

- 予想されるエラー Oracle XE を使用する場合、DMS エージェントリーダーは通常、次のような例外をログに記録します。- Debezium コンシューマー$NAME は例外で停止しました。変更イベントプロデューサーで例外が発生しました。このコネクタは停止します。
- 機能制限スナップショットのみの移行のみが機能すると予想されます。スナップショットが成功した場合でも、Debezium エラーメッセージがログに表示される場合があります。
- 使用ポリシー Oracle XE でのスナップショットのみの移行は、テストまたは評価の目的にのみ適しています。この構成は、運用ユースケースではサポートされていません。

移行された PL/ SQL ボディの非修飾オブジェクト参照

説明 DMS が Oracle トリガー、プロシージャ、ファンクション、およびパッケージを EDB Postgres Advanced Server に変換すると、変換された PL/ SQL ボディは Oracle ソースからの非修飾オブジェクト参照を保持します。

Oracle と EPAS は、非修飾名の解決が異なります。

- **Oracle** PL/ SQL 本文内の非修飾参照は、誰が呼び出したかに関係なく、定義者のスキーマオブジェクトを所有するスキーマに解決されます。
- **EPAS** 非修飾参照は、呼び出し元の `search_path` を使用して実行時に解決されます。呼び出したセッションの `search_path` に移行されたスキーマが含まれていない場合、実行時に参照は失敗します。

DMS エージェントは、各 DDL ステートメントの実行時に `search_path` を明示的に設定するため、スキーマの移行自分自身は成功します。障害は、移行後、アプリケーションまたはユーザーが別の `search_path` とのセッションから移行された PL/ SQL オブジェクトを呼び出すときに表面化します。

この制限は、移行スコープが **スキーマのみ** または **スキーマとデータ**である Oracle から EPAS への移行にのみ適用されます。これは、 **データのみ**の移行には影響しません。ここでは、移行が実行される前に顧客によってスキーマオブジェクトが作成され、DMS は PL/ SQL ボディを生成または変換しません。

例

この Oracle トリガーは、スキーマ接頭辞なしで `test_orders_seq` を参照します。

```
CREATE TRIGGER "DEBEZIUM"."TEST_ORDERS_TRG"
    BEFORE INSERT ON test_orders_legacy
    FOR EACH ROW
BEGIN
```

(次のページに続く)

(前のページからの続き)

```
    IF :NEW.order_id IS NULL THEN
        SELECT test_orders_seq.NEXTVAL INTO :NEW.order_id FROM dual;
    END IF;
END;
```

EPAS の DMS によって生成された DDL は、非修飾の参照を保持します。

```
CREATE OR REPLACE TRIGGER TEST_ORDERS_TRG
    BEFORE INSERT ON test_orders_legacy
    FOR EACH ROW
BEGIN
    IF :NEW.order_id IS NULL THEN
        SELECT test_orders_seq.NEXTVAL INTO :NEW.order_id FROM dual;
    END IF;
END;
```

移行後、動作は呼び出し元によって異なります。

- search_path = debezium トリガーとのセッションは正常に実行されます。
- デフォルト search_path = "$user", public でのセッション sequence "test_orders_seq" does not exist で失敗します。

回避策

環境に基づいて、次のアプローチのいずれかを選択します。

- **DDL のスキーマ修飾推奨**カットオーバーの前に、変換されたすべてのトリガー、プロシージャ、ファンクション、およびパッケージ本体を確認し、非修飾のオブジェクト参照にソーススキーマ名を接頭辞します。変更した DDL を宛先に適用します。このアプローチは、search_path への実行時依存関係を削除します。

次の修正スキーマは、トリガー本体内のシーケンス参照をスキーマ修飾します。トリガーヘッダーのテーブル参照は、オブジェクトの所有権を明示するように修飾されます。

```
CREATE OR REPLACE TRIGGER TEST_ORDERS_TRG
    BEFORE INSERT ON debezium.test_orders_legacy
    FOR EACH ROW
BEGIN
    IF :NEW.order_id IS NULL THEN
        SELECT debezium.test_orders_seq.NEXTVAL INTO :NEW.order_id FROM dual;
    END IF;
END;
```

- **ロールまたはデータベースのデフォルトの ``search_path`` を設定します**永続的なデフォルトを設定して、新しいセッションが search_path に移行したスキーマを自動的に含めます。

```
ALTER ROLE app_user SET search_path = debezium, public;
-- or, database-wide:
ALTER DATABASE mydb SET search_path = debezium, public;
```

- **接続時に ``search_path`` を設定します**移行したスキーマを search_path に含めるようにすべてのアプリケーション接続を構成します。接続文字列 options=-c search_path=debezium,public または接続後の最初のステートメントとして。

```
SET search_path = debezium, public;
```

明示的な search_path DBA セッション、モニタリングツール、または BI クライアントを使用せずに接続するアドホックセッションは、どの回避策を選択しても障害が発生するリスクがあります。

Postgres の制限

列でサポートされていないドメインタイプ定義

> **Tip**
>
> HM バージョン 2026.4 以降で部分的に解決されました。

EDB DMS は、次のデータ型ドメインのデータ型としてユーザー定義ドメインを含む列を含むテーブルの移行をサポートしていません。

- DATE イノベーションリリース HM 2026.4 でサポートを追加しました
- TIME イノベーションリリース HM 2026.4 でサポートを追加しました
- TIMESTAMP イノベーションリリース HM 2026.4 でサポートを追加しました
- INTERVAL イノベーションリリース HM 2026.4 でサポートが追加されました
- UUID イノベーションリリース HM 2026.4 でサポートが追加されました
- ENUM
- JSON イノベーションリリース HM 2026.4 でサポートが追加されました
- XML イノベーションリリース HM 2026.4 でサポートが追加されました
- POINT イノベーションリリース HM 2026.4 でサポートを追加しました
- LTREE
- TSVECTOR

TSVECTOR タイプの移行の回避策

TSVECTOR タイプ列の移行はサポートされていませんが、この回避策を適用して移行を有効にすることができます。たとえば、ソースデータベースのこの messages テーブルには、TSVECTOR 列が含まれています。

```
CREATE TABLE messages (
    title       text,
    body        text,
    tsv         tsvector
);
```

移行を開始する前に、宛先データベースに TSVECTOR 列なしでテーブルを作成します。

```
CREATE TABLE messages (
    title       text,
    body        text,
);
```

移行を実行するときは、 tsv 列チェックボックスをクリアして **Review your Tables** ページでテーブルを確認するときに、HM コンソールの TSVECTOR 列を除外する必要があります。

移行が完了すると、title および body 列がターゲットデータベースに移行されます。これで、TSVECTOR 列をターゲットテーブルに追加できます。

```
ALTER TABLE messages ADD COLUMN tsv TSVECTOR DEFAULT NULL;
```

次に、次の UPDATE ステートメントを実行して、移行されたすべてのデータの列を設定します。

```
UPDATE messages SET tsv = to_tsvector(english, CONCAT(title,  , body)) WHERE tsv IS␣
↪NULL;
```

拡張機能の移行の制限

Postgres ディストリビューション間を移行する場合、 EDB Postgres AI エージェントは、ソースデータベースにインストールされているすべての拡張機能を抽出し、スキーマの移行の一部として宛先に適用します。エージェントは、ホワイトリストも拒否リストも使用せずにソースにインストールされたものを抽出します。次の制限が適用されます。

- **データベースごとのスコープ**エージェントは、データベースごとに拡張機能を抽出します。同じ Postgres サーバー上の他のデータベースにインストールされた拡張機能は、含まれません。複数のデータベースから拡張機能を移行するには、データベースごとに 1 つの移行を実行します。
- **バージョンピンニング**エージェントは、ソースで見つかったバージョンと正確に各拡張機能をインストールします。その正確なバージョンが宛先で利用できない場合、スキーマの移行は失敗します。異なる Postgres バージョンまたはディストリビューション間を移行する場合、バージョンが利用可能な場合でも、拡張機能の動作が予期せず異なる場合があります。バージョン間の拡張機能の互換性は保証されていません。
- **FDW オブジェクトの移行**エージェントは、外部データラッパー FDW 拡張機能自分自身たとえば、postgres_fdw または oracle_fdw をインストールしますが、FDW オブジェクトは移行しません。外部サーバー、ユーザーマッピング、および外部テーブルはストリップされ、移行後に宛先で手動で再作成する必要があります。
- **スーパーユーザーを必要とする拡張機能**スーパーユーザーのインストールを必要とする拡張機能は、

スーパーユーザーを付与しない管理対象 Postgres サービス Amazon RDS または Aurora などの適用時にサイレントに失敗します。移行評価は、これらの拡張機能に `RequiresSuperuser` としてフラグを立てますが、これは移行の続行を妨げるものではありません。

- **固定スキーマ拡張機能**一部の拡張機能は、特定のスキーマにのみインストールでき、別のスキーマに配置できません。ソースにデフォルト以外のスキーマのこのような拡張機能がある場合、エージェントは宛先にそのスキーマ配置をレプリケートできません。
- **Postgres 10〜12 の拡張機能スーパーユーザー表示** Postgres 10〜12 では、移行評価は常に、そうでない場合でも、スーパーユーザーのインストールを必要とする拡張機能を表示します。これは表示のみの制限であり、移行自分自身には影響しません。

スキーマの取り込みの既知の問題

Postgres DDL 抽出の予約スキーマ名

EDB Postgres AI エージェントは、Postgres DDL 抽出中に内部で 2 つの合成スキーマ名を使用します。

- `__extensions__` — `CREATE EXTENSION` DDL ステートメントを保存します
- `__globals__` — グローバルオブジェクト `pg_dumpall` からのロール、権限、およびメンバーシップを保存します

これらの名前は予約されており、エージェントはそれらと競合するユーザスキーマを拒否します。ダブルアンダースコア規則は、実際のスキーマ名との衝突を回避するために意図的に選択されるため、これがほとんどのデータベースに影響を与える可能性は低いです。

EDB Postgres Advanced Server ソース用に取り込まれた追加スキーマ

> **Tip**
>
> HM バージョン 2025.11 以降で解決されました。

EDB Postgres AI エージェントに登録したソース EDB Postgres Advanced Server データベースのスキーマのリストを表示すると、明示的に作成したスキーマに直接対応しない追加スキーマが観察される場合があります。これらの追加スキーマは、Oracle PACKAGE オブジェクトを表す Oracle 互換性機能セットの一部として EDB Postgres Advanced Server によって内部生成されます。これは既知の問題であり、将来のリリースで修正される予定です。それまでは、これらの生成されたスキーマは、移行の目的で無視できます。

HM Migration Portal の既知の問題

移行ポータルプロジェクトの自動作成を無効にするには、回避策が必要です

> **Tip**
>
> HM バージョン 2026.1 以降で解決されました。バージョン 2026.1 以降を使用している場合は、*Migration Portal* プロジェクトの自動作成の無効化 を参照してください。

2025. 12 以前のバージョンには、内部 beacon-server サービスに影響を与える既知の問題があり、ConfigMap で autocreate_mp_projects: false パラメーターを設定しても、移行ポータルプロジェクトの自動作成が妨げられません。

回避策移行ポータルプロジェクトの自動作成を防止し、手動作成を強制するには、beacon-server ConfigMap 内でパラメーター max_automatic_project_creation_bytes を非常に低い値 1 などに設定します。この低いしきい値により、すべてのスキーマが制限を超え、自動作成プロセスが正常にバイパスされます。

1. beacon-server ConfigMap を編集します。

```
kubectl edit configmap beacon-server-config -n upm-beacon
```

1. beacon_server.yaml ブロックを見つけて、server の下の max_automatic_project_creation_bytes セクションを挿入または変更します。値を 1 などの非常に低い数値に設定して、自動作成がバイパスされるようにします。

```
apiVersion: v1
data:
  beacon_server.yaml: |
  # [...]
    server:
    # [...]
      migration_portal_configuration:
        max_automatic_project_creation_bytes: 1
```

1. beacon_server 展開を再起動して、変更を適用します。

```
kubectl rollout restart deployment upm-beacon-server -n upm-beacon
```

警告

upm-beacon-server は、EDB Postgres AI エージェントを使用して、HM インスタンス全体のすべてのデータベース登録とスキーマ取り込みアクティビティをサポートします。 *upm-beacon-server* を再起動すると、プロジェクトに関係なく、すべての進行中のデータベース登録とスキーマの取り込みが停止します。この再起動は他のユーザーと調整してください。

[Estate > Migrations] タブに既に表示されており、スキーマの取り込みが終了しているデータベースは、再起動の影響を受けません。 .. ::

1. 再起動したサービスのステータスをモニタできます。

```
kubectl get pods -n upm-beacon
```

注釈

プロジェクトを手動で作成した後に HM console warning for manual Migration Portal project creation persists が発生する現在別の既知の問題があります。

HM 警告は手動で移行ポータルプロジェクトを作成した後も持続します

> **Tip**
>
> HM バージョン 2026.1 以降で解決されました。

バージョン 2025.12 以前では、移行ポータルプロジェクトの自動作成が失敗すると、取り込まれたスキーマがデフォルトのサイズ制限 300 MB または別の指定された制限を超えているため、HM コンソールに永続的な警告が表示され、ユーザーに指示しますプロジェクトを手動で作成します。この警告は、ユーザーが Oracle スキーマに必要な移行ポータルプロジェクトを正常に作成した後でも、HM コンソールに表示され続けます。

自動移行ポータルプロジェクトの作成は、無効な `resource_id` 文字により失敗します

登録された Oracle データベースに指定された `resource_id` に無効な文字が含まれている場合、移行ポータルプロジェクトの自動作成は失敗しますエラーメッセージ **Failed to create** 。

この障害は、移行ポータルプロジェクト名の構築に `resource_id` 値が使用されるために発生します。現在、プロジェクトの作成を妨げるハイフン-などの特殊文字をサポートしていません。

回避策移行ポータルプロジェクトの自動作成を成功させるには、Oracle データベースの EDB Postgres AI エージェント構成で指定された `resource_id` 値が次の命名ルールに準拠する必要があります。

- 英数字 A〜Z、a〜z、0〜9 を含めることができます。
- アンダースコア_を含めることはできますが、アンダースコアで始めることはできません。
- 数字で始まることはできません。
- ハイフン-などの特殊文字を含めることはできません。

アップグレード後に移行ポータルデータベースポッドの起動に失敗する

> **Tip**
>
> HM バージョン 2026.2 以降で解決されました。

説明: 2026.1 イノベーションリリースへのアップグレード後、移行ポータルの PostgreSQL ポッドたとえば `mp-epas-16` 、`mp-epas-17` は、`CrashLoopBackOff` または `ImagePullBackOff` 状態になる場合があります。これは、展開が、リポジトリで使用できなくなった不正なイメージダイジェストまたはタグを参照しているために発生します。

回避策 エラーを解決し、移行ポータルを正常な状態に復元するには、`kubectl delete pod <pod-name>` を使用して影響を受けるポッドを手動で削除します。展開では、正しい更新されたイメージを使用してポッドが自動的に再作成されます。

移行ポータルを使用してスキーマを EPAS 18 に移行する際の SSL 接続エラー x25519

説明: 移行ポータルの **オンラインスキーマ移行**機能を使用して、FIPS 適用環境で EDB Postgres Advanced Server データベースバージョン 18 をターゲットとすると、接続が移行ポータル UI エラーで失敗する場合があります。`Failed to connect: Failed to initialize pool: SSL error: Cannot decode named group: x25519` . これは、ハイブリッドマネージャー移行ポータルで有効になっている FIPS 環境によるものです。

この問題は、Postgres 18 がデフォルトで `x25519` SSL グループに設定されているために発生します。これは、移行ポータルアプリケーションが使用する特定のコンポーネントで現在サポートされていません。この問題は、特に**オンライン移行**オプションに影響します。ここで、宛先データベースは、HM 管理 EPAS データベースクラスターまたは自己管理 EPAS データベースのいずれかです。

回避策接続障害を解決するには、宛先データベースクラスターでサポートされている SSL グループリストを手動で更新して、FIPS 準拠の NIST 曲線 `prime256v1` などのみを含める必要があります。

HM 管理の EPAS データベースクラスターバージョン 18 の構成例

1. HM コンソールで、宛先クラスターをホストするプロジェクトを選択します。
2. **Clusters > Postgres Clusters** を選択し、バージョン 18 クラスターを選択します。
3. **Quick Actions > Edit Cluster** を選択し、 **DB Configuration** タブに移動します。
4. `ssl_groups` パラメーターを見つけます。
5. 値をデフォルトから `prime256v1` に変更します。
6. 変更を保存し、クラスターが調整できるようにします。

クラスターが更新された後、移行ポータルのオンラインスキーマの移行で接続テストを再試行します。

自己管理 EPAS データベースバージョン 18 の構成例

1. `postgresql.conf` ファイルを編集し、 `ssl_groups` パラメーターを見つけて `prime256v1` に変更します。
2. `systemctl` 、`psql` 、または使用する方法で Postgres サービスをリスタートすることにより、新しい構成をロードします。

クラスターが更新された後、移行ポータルのオンラインスキーマの移行で接続テストを再試行します。

自動移行ポータルプロジェクトの作成は、「UNAVAILABLE: unresolved address」で失敗します

> **Tip**
>
> HM バージョン 2026.4 以降で解決されました。

説明移行ポータルプロジェクトの自動作成は、内部サーバーエラー `rpc error: code = Internal desc = Internal server error occurred: UNAVAILABLE: unresolved address` で失敗する場合があります。これは通常、ハイブリッドマネージャーコンポーネントと移行ポータルサービス間の一時的なネットワーク解決の問題が原因で発生します。

回避策このエラーが発生した場合は、次のいずれかの方法を使用して解決します。

- 手動作成ネットワーク状態が安定するまで 20〜40 分待って、HM コンソールを介して移行ポータルプロジェクトを手動で作成します。
- リソース ID のリセット `beacon_agent.yaml` 構成ファイルの `resource_id` を自分で割り当てた新しい ID に変更し、EDB Postgres AI エージェントを再起動するか、新しいプロジェクトを作成し、エージェントをその新しいプロジェクトにマップして、新しい同期をトリガーします。

不変の `ragchew-ai-vendor-secrets` シークレットが原因でバージョン 2026.4 へのアップグレードが失敗する

説明: HM をバージョン 2026.4 にアップグレードすると、`edb-migration-copilot` の `Failed to install component` エラーでアップグレードプロセスが失敗する場合があります。これは、アップグレードが `edb-migration-copilot` 名前空間の既存の `ragchew-ai-vendor-secrets` Kubernetes シークレットの `type` フィールドを変更しようとするために発生します。 `type` フィールドは、シークレットが作成されると不変であるため、Kubernetes API サーバーはエラー `Secret "ragchew-ai-vendor-secrets" is invalid: type: Invalid value: "Opaque": field is immutable` で更新を拒否します。

回避策 2026.4 にアップグレードする前に、 `edb-migration-copilot` 名前空間のシークレット `ragchew-ai-vendor-secrets` を手動で削除します。オペレーターは、アップグレード中に正しいタイプで再作成します。

```
kubectl delete secret ragchew-ai-vendor-secrets -n edb-migration-copilot
```

SQL ファイル評価は、ソースデータベースタイプがデフォルトで `Oracle_19c` に設定されます

> **Tip**
>
> HM バージョン 1.4.0 以降で解決されました。

説明アプリケーション移行評価の SQL ファイルをアップロードする場合、HM コンソールはソースデータベースのリソース ID をアップロードされたファイルに関連付けません。リソース ID が提供されないため、`upm-beacon` は、ソースデータベースバージョンを `UNKNOWN` に設定して評価要求を移行ポータルに送信します。移行ポータルは、評価を処理するときにソースデータベースタイプとしてデフォルトの `Oracle_19c` を設定します。その結果、評価は、実際のソースデータベースバージョンの互換性ルールを正確に反映しない場合があります。さらに、アプリケーション SQL 評価がエラーなしで完了するには、移行ポータルに少なくとも 1 つのプロジェクトが存在する必要があります。

回避策 SQL ファイルのアップロードを介してアプリケーション移行評価を開始する前に、移行ポータルに少なくとも 1 つのプロジェクトが存在することを確認します。ソースデータベースのリソース ID をアップロードされた SQL ファイルに関連付ける修正は、将来のリリースに向けて追跡されています。

その他の既知の問題

RHOS 環境のみでロードバランサーなしでサポートされているデータ移行

> **Tip**

HM バージョン 2026.1 以降で解決されました。

説明ハイブリッドマネージャー内で Kafka を公開するデータの移行に必要な主な方法は、ロードバランサーを介します。例外は RHOS 環境で、ネイティブルートを使用して Kafka を公開します。したがって、ロードバランサーが無効になっている非 RHOS 環境では、NodePort モードを使用して Kafka の公開をサポートするフォールバックメカニズムがないため、データ移行ワークフローは機能しません。

回避策 RKE などの非 RHOS 環境でデータ移行ワークフローを有効にするには、ロードバランサーを構成する必要があります。

Optimizing migration performance

EDB Data Migration Service DMS は、Hybrid ManagerHM に含まれています。これを活用して、既存のデータベース Oracle、自己管理の Postgres などから HM 管理のデータベースクラスターにデータを移行できます。EDB DMS は、スナップショットとスナップショット+ストリーミングの 2 つの移行方法をサポートしており、最小限またはゼロのダウンタイムでデータを移行できます。

HM での移行ワークロードを理解する

EDB DMS は、HM が展開される場所と同じ Kubernetes 環境オンプレミス、AWS、Red Hat OpenShift、および Google Cloud 内で実行されます。これは、移行操作に必要なワークロードのほとんどが Kubernetes クラスターによって管理および最適化されることを意味します。

HM を実行するための一般的な要件を満たす Kubernetes 環境は、スムーズな移行に役立つため、一般にリソースを大幅に微調整する必要はありません。ただし、大規模なデータセットの場合、特定のコンポーネントを最適化することによりプロセスを大幅に高速化できます。

リソースの割り当てについて考慮する必要がある主なコンポーネントは次のとおりです。

- **リーダーモードの EDB DMS エージェント**このコンポーネントは、ソースデータベースと同じマシンで、またはソースデータベースと HM に接続できる補助マシンで実行できます。
- **宛先データベースクラスター**これは、HM によって管理されるデータベースまたは別の外部の自己管理データベースです。ただし、HM 管理データベースに焦点を当てます。ここにデータが存在します。
- **ライタモードの EDB DMS エージェント**このコンポーネントは、移行先が外部自己管理データベースクラスターであり、そのデータベースのホストマシンで実行されている場合にのみ必要です。

スナップショットまたはスナップショット+ストリーミング移行を実行するかどうかも、パフォーマンスに異なる影響を与えます。

- **スナップショット移行**これらは主に、大量の既存のデータをソースデータベースから宛先に効率的に転送することに焦点を当てています。これらのパフォーマンスは、合計データサイズと持続的なディスク I/O に大きく影響されます。
- **スナップショット+ストリーミング移行**最初に既存のデータのスナップショットを転送し、次に進行中の変更トランザクションをソースから宛先まで継続的にストリーミングします。ここでのパフォーマンスは、持続トランザクションレート TPS とソースの書き込みアクティビティに追いつく機能によって左右されます。

大規模なデータセットの移行パフォーマンスの最適化

HM および DMS は、リソースの使用量を処理するように設計されています。 100 GB より小さいデータセットの場合、それ以上のチューニングなしで高速な移行を実行するように構成されます。ただし、100 GB を超えるデータセットの場合、いくつかの対象を絞った変更で大幅にパフォーマンスが向上し、移行が迅速に行われる可能性があります。

宛先データベースのパフォーマンスの調整

宛先データベースクラスター HM 管理データベースまたは自己管理データベースを最適化すると、リソース調整から最も重要なパフォーマンスの利点が得られます。

> **注釈**
>
> これらの推奨事項は、現在のテストにもとづいていますが、変更される場合があります。私たちは、移行パフォーマンスを向上させ、移行時間の削減に継続的に取り組んでいます。

ストレージの推奨事項

移行中、ストレージ IOPS は 2000 でピークに達し、平均スループットは約 300 MB/秒、ピークは最大 350 MB/秒です。これらの需要に対応するには

- WAL ボリュームとデータボリュームを分離します。データボリュームとログ先行書き込み WAL ボリュームの両方で、最小 2500〜3000 IOPS と少なくとも 200〜250 MB/秒のスループットを処理できるストレージボリュームを目指します。最適なパフォーマンスを得るには、WAL とデータに別個のボリュームを使用することを強くお勧めします。
- ユニファイドストレージと WAL ボリューム。ユニファイドストレージと WAL ボリュームは、パフォーマンスに悪影響を与える可能性があります。たとえば、3000 IOPS と 125 MB/秒のスループットで単一のボリュームを使用すると、移行時間が 40%以上増加する可能性があります。ユニファイドストレージと WAL ボリュームを使用する必要がある場合、300〜400 MB/秒のスループットをお勧めします。WAL がいっぱいになると、移行は失敗することに注意してください。

これを調整する方法

HM 管理データベースクラスターの場合

- クラスターのストレージ **公開されたバンドルの編集**によって WAL とデータボリュームの割り当てを調整します。
- デフォルトでは、HM 管理クラスターはデータと WAL ボリュームに別個のボリュームを使用しますが、クラスターの作成中に**先行書き込みログに別のストレージボリュームを使用する**オプションをクリアすると、後でこの設定を有効にすることはできません。
- ボリューム速度 IOPS またはスループットは、クラスターの基になるインフラストラクチャによって決まります。HM 管理のデータベースクラスターに移行する場合、クラスターで使用可能な特定のストレージクラス、およびそれに関連する IOPS およびスループット特性は、HM 管理者および基になるクラウドまたはオンプレミス環境によって決定されます。したがって、クラスターで使用可能なストレージオプションを確認し、HM 管理者と協力して、選択したストレージが推奨パフォーマンス基準を満たしていることを確認します。クラウドプロバイダーインフラストラクチャのプロビジョニング

の詳細については、Request Amazon EBS volume modifications または Google Cloud - About Persistent Disk performance を参照してください。

メモリの推奨事項

5 GB のメモリで十分ですが、量が多いほど、移行中の最適なパフォーマンスが促進されます。宛先データベースには少なくとも 10 GB のメモリを使用します。

これを調整する方法

HM 管理データベースクラスターの場合

- 公開されたバンドルの編集 クラスターのインスタンスサイズ **クラスター設定**の下で、メモリ割り当てを調整します。
- **DB Configuration** で `shared_buffers` 、`effective_cache_size` 、および `wal_buffers` パラメーターを調整することもできます。

CPU の推奨事項

宛先データベースクラスターには、少なくとも 4 つの CPU コアが必要です。コア数が少ないと、移行パフォーマンスに悪影響を与えます。

これを調整する方法

HM 管理データベースクラスターの場合、クラスターのインスタンスサイズに応じて CPU 割り当てを調整します公開されたバンドルの編集 。

> **注釈**
>
> 移行プロセス中にこれらの最適化を適用します。移行が完了した後、進行中のワークロード要件に基づいて、必要に応じてこれらの値を再調整できます。

広範なテストの結果、移行のパフォーマンスに影響を与える最大の要素はボリューム速度 IOPS およびスループットであることが示されています。CPU を推奨値より減らすとパフォーマンスが低下する可能性がありますが、ストレージの IOPS とスループットが低下するほどではありません。ストレージのスループットを半分にすると、移行時間が 40%以上増加する可能性がありますが、CPU 数を 2 コアに半分にすると、10〜15%増加する可能性があります。これらの推奨事項は、最高の移行パフォーマンスを生み出すように設計されています。より低い値を使用しても、移行は成功する場合がありますが、速度は遅くなります。

WAL 構成

WAL チェックポイントは、データの移行中に最もリソースを消費するタスクの 1 つです。デフォルトでは、`max_wal_size` は 1GB に設定されています。これは、WAL ファイルに書き込まれるすべての GB に対してチェックポイントが作成されることを意味します。移行中に、これにより、チェックポイントが 30〜60 秒ごとに発生し、宛先データベースにかなりの負荷がかかる場合があります。

移行中に `max_wal_size` を増やすことにより、宛先データベースの負荷を軽減できます。テストでは、`max_wal_size` が大きくても移行速度に影響を与えないことがわかりました。たとえば、20 GB `max_wal_size` を使用した移行は、1 GB 設定の移行と同じ時間かかります。ただし、20 GB 設定を使用した移行では、消費するリソースが約 50%少なくなります。

WAL ロギングにより、移行されたすべてのデータがチェックポイントを介して適切に保存されることが保証されることを考えると、いくつかの重要な機能がそれに依存しているため、`max_wal_size` を無効にすることは、ほとんどの EDB データベースで実行可能なオプションではありません。

これを調整する方法

HM 管理データベースクラスターの場合

クラスターの DB 構成 公開されたバンドルの編集 によって WAL 構成を調整します。`max_wal_size` パラメーターを検索し、新しいサイズを MB 単位で指定し、保存して構成を適用します。

EBD DMS エージェントのパフォーマンスの調整

EDB DMS エージェントの最適なパフォーマンスは、これらのコンポーネントがインストールされるホストマシンがこれらの仕様に従っている場合に達成されます。

リーダーモードの EDB DMS エージェント

CPU 3 CPU コア

メモリ 4 GB

ライタモードの EDB DMS エージェント

CPU: 2 CPU コア

メモリ 2.5 GB

共有ホストの考慮事項

最適な移行パフォーマンスを得るには、EDB DMS コンポーネントとソースデータベースが同じマシン/ホストで実行されている場合、リソースの割り当てを注意深く検討してください。

- **DMS エージェントはホストを共有します**両方のエージェントが同じマシン上にあり、1 つがリーダーモードで、2 つ目がライタモードである場合、組み合わせたリソース少なくとも 5 つの CPU コアと 6.5 GB メモリをプロビジョニングします。
- **DMS コンポーネントとソースデータベースはホストを共有します** DMS エージェントがソースデータベースと同じホスト上にある場合、ホストのリソースが大幅に増加して、データベースの操作ワークロードと移行の要求の両方に対応することを確認します。

大量移行の移行パフォーマンスの最適化

HM および DMS は、複数の同時移行を処理するように設計されています。ただし、大量の移行つまり、大量の同時または大規模な移行を実行する場合、Kafka ブローカーの数を増やすと、データ移行のパフォーマンスを大幅に向上させることができます。

> **Tip**
>
> データの移行中に Kafka ブローカーで高い CPU とメモリの使用量が観察された場合、デフォルトの DMS 構成は大量の移行には不十分です。 *Agent Factory* メトリックの *Grafana* ダッシュボードの設定 を

使用して、*transporter-kafka-operator* 名前空間の Kafka ポッドのリソース使用量を確認します。

Kafka ブローカー数を変更します

1. kubeconfig ファイルを使用して、HM Kubernetes クラスターに接続します。(特定のコマンドは、クラウド環境によって異なります。)

```
# AWS example:
aws eks update-kubeconfig --name eks-1234567890-main --region eu-west-2 --profile app-
↪dev
```

1. 変更を行う前に、Kafka クラスターが既にプロビジョニングされているかどうかを確認します。これは、以前にデータ移行を実行した場合に当てはまります。

```
kubectl get kafka -n transporter-kafka-operator
```

1. 既存の移行と Kafka クラスターを削除します。コマンド出力に Kafka リソースがリストされている場合、最初にすべての移行を削除して、クリーンな状態で開始する必要があります。

警告

Kafka クラスターは、通常、HM インスタンス全体でのすべてのデータ移行アクティビティをサポートするように展開されます。 Kafka リソースを削除すると、プロジェクトに関係なく、アクティブなデータの移行がすべて停止します。この変更を他のユーザーと調整します。 !!!

HM コンソールの **移行>移行**の下で、プロジェクト内のアクティブな完了した移行をすべて削除します。

テスト環境の場合

テストまたは開発環境の場合、kubectl を使用して Kafka クラスターを削除することもできます。ただし、この方法は、他のユーザーが使用している可能性のあるすべてのリソースを削除するため、非実稼働環境でのみお勧めします。

```
kubectl -n transporter-kafka-operator delete $(kubectl get strimzi -o name -n␣
↪transporter-kafka-operator)
```

1. transporter-dp-provisioner 名前空間の ConfigMap の名前を取得します。

```
kubectl get cm -n transporter-dp-provisioner
```

1. 編集のために ConfigMap を開きます。 <config_map_name> を、前の手順で見つけた名前に置き換えます。

```
kubectl edit cm transporter-dp-provisioner-<config_map_name> -n transporter-dp-
↪provisioner
```

1. ConfigMap で broker_number フィールドを見つけて、値を目的の数の Kafka ブローカーたとえば broker_number: 3 に変更します。

2. ConfigMap を保存します。新しい構成はリロードされ、約 10 秒以内に適用されます。

3. 変更したブローカー容量を使用する新しいデータの移行を開始できます。

Migratiog

Choose your migration path

各種類のデータベースの移行を実行する方法に関するエンドツーエンドの手順については、以下を参照してください。

- *AWS RDS PostgreSQL or AWS RDS Aurora (PostgreSQL compatible) database to an HM-managed database*
- *Migrate your Oracle database to an HM-managed database*
- *Migrate your self-managed Postgres database to a different self-managed Postgres database*
- *Migrate your self-managed Postgres database to an HM-managed database*

Migrate your Oracle database to an HM-managed database

これらは、Oracle データベースから Hybrid ManagerHM によって管理される EDB Postgres Advanced Server クラスターへのエンドツーエンドの移行に含まれるタスクです。このプロセスでは、2 つのエージェントを使用します。EDB EDB Postgres AI エージェント `beacon-agent` は、Oracle データベースを HM に登録し、評価と変換のためのスキーマを取り込みます、と実際のデータ移行を実行する EDB DMS エージェント `cdcagent` 。

1. ** *Configuring your auxiliary machine or bastion host* ** —Oracle ソースと HM 間のブリッジとして機能する補助マシンを構成します。このマシンは両方のエージェントをホストします。

2. ** *Preparing Oracle source database* ** —専用の移行ユーザーを作成し、スキーマ取り込みと変更データキャプチャ CDC をサポートするように Oracle データベースを構成します。

3. ** *Configuring the EDB Postgres AI agent to register your Oracle database* ** — EDB Postgres AI エージェントをインストールおよび構成して、Oracle データベースを HM に接続します。これにより、移行ポータルで使用する移行評価と自動スキーマ DDL 抽出が有効になります。

4. ** *Preparing your HM-managed destination database* ** — 移行したスキーマとデータを受信する HM 管理の EDB Postgres Advanced Server クラスターを設定します。

5. ** *Migrating Postgres schemas* ** — 移行ポータルを使用して、 EDB Postgres Advanced Server と互換性のある Oracle DDL を変換し、データの移行が開始される前に宛先データベースにスキーマをインポートします。

6. ** *Configuring and running the EDB DMS agent* ** — EDB DMS エージェントをリーダーモードで構成して、データ移行の Oracle ソースから変更を読み取るようにします。

7. ** *Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database* ** — HM コンソールでデータ移行を作成および実行します。

Configuring your auxiliary machine or bastion host

EDB Postgres AI エージェント `beacon-agent` および DMS エージェント `cdcagent` は、Oracle データベースサーバー自分自身または別の補助マシンまたは要塞ホストで実行できます。ほとんどの運用環境では、専用の補助マシンでエージェントを実行することが優先されます。これにより、データベースサーバーへの追加ソフトウェアのインストールが回避され、単一のエージェントインスタンスでマルチプルのサーバー上のデータベースに接続できます。

必要なソフトウェア

エージェントを実行する前に、補助マシンに次のソフトウェアをインストールしていることを確認します。

- **Java 17 以降** — DMS エージェント `cdcagent` に必要です。`java -version` で確認します。
- **Oracle Instant Client with SQL*Plus** —両方のエージェントに必要です。 EDB Postgres AI エージェントは、SQL *Plus を使用して Oracle ソースから DDL を抽出します。DMS エージェントは、変更データキャプチャ CDC 接続に Oracle JDBC ライブラリを使用します。

Download Oracle Instant Client し、 Basic と SQL* Plus パッケージの両方をインストールします。ソース Oracle データベースと互換性のあるバージョンをダウンロードしてください。

- **EDB Postgres AI エージェント** `beacon-agent` - データベース登録とスキーマ評価を処理します。*beacon-agent version compatibility matrix* を確認します。次に、 install a compatible version of the EDB Postgres AI agent 。
- **DMS エージェント** `cdcagent` - データの移行と CDC を処理します。*DMS agent version compatibility* を確認します。次に、 *install a compatible version of the DMS agent* 。

環境の構成

両方のエージェントが、実行時に Oracle ライブラリとバイナリを見つけられる必要があります。補助マシンで次の環境変数を設定し、`</path/to/instantclient>` を実際のインスタントクライアントディレクトリ `/opt/oracle/instantclient_19_19` などに置き換えます。これらの変数は、エージェントを実行するのと同じシェルセッションで設定する必要があります。永続的な構成の場合、それらをシェルプロファイルたとえば`~/.bashrc` または`~/.bash_profile` などに追加します。エージェントを systemd サービスとして実行する場合、代わりにサービスファイルにこれらの変数を設定します。

```
export ORACLE_HOME=</path/to/instantclient>
export PATH=$ORACLE_HOME:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME:$LD_LIBRARY_PATH
```

> **注釈**
>
> Oracle Instant Client の場合、SQL*Plus バイナリと共有ライブラリ `.so` ファイルは両方とも Instant Client のルートディレクトリに直接あります。*bin/* または `lib/` サブディレクトリにはありません。 *PATH* と `LD_LIBRARY_PATH` の両方を Instant Client ルートに設定します。

続行する前に設定を確認してください。

```
which sqlplus
```

出力には、バージョン番号を持つ Instant Client ディレクトリ内の SQL*Plus バイナリへのパスが表示されます。または、次を実行します。

```
sqlplus -V
```

これは、 SQL*Plus のバージョンを示しています。

ネットワーク接続

補助マシンとソースデータベースホスト間、および補助マシンとハイブリッドマネージャー間のネットワーク接続を確認します。必要な特定のポートは、後でエージェントを構成するときに確認します。

次のステップ

Preparing Oracle source database

Preparing Oracle source database

移行を開始する前に、Oracle ソースデータベースを構成し、移行専用ユーザーを作成します。どれだけ構成する必要があるかは、何を行う予定なのかによって異なります。

- **スキーマの取り込みと評価のみ** EDB Postgres AI エージェントデータベースレベルの変更を行う必要はありません。Oracle データディクショナリビュー特に `SELECT_CATALOG_ROLE` または `SELECT ANY DICTIONARY` を介して付与されたビューへの読み取りアクセスと `CREATE TABLE` 特権を持つ移行ユーザーを指定するだけです。評価スクリプトは、`CREATE TABLE` 特権を使用して、実行中に一時テーブルを作成します。データベースには永続的な変更は行われません。たとえば、移行をコミットする前に大規模な Oracle 資産を評価することが目的である場合は、**移行ユーザーの構成** に直接スキップします。
- **データ移行** DMS エージェント 追加のデータベースレベルの構成 - `ARCHIVELOG` モード、サプリメンタルログ、専用テーブルスペース、および移行ユーザーの拡張 LogMiner 特権が必要です。このページのすべてのセクションを完了します。

スキーマ評価とデータの移行の両方を実行する予定がある場合は、すべてのセクションを完了します。ここで作成する移行ユーザーは、両方のエージェントにサービスを提供します。

前提条件

- Oracle アーキテクチャ CDB / PDB マルチテナントと非 CDB を特定します。
- 移行するスキーマとテーブルの名前を収集します。

データベースの構成

ARCHIVELOG モードの有効化

移行ツールでは、データベースを `ARCHIVELOG` モードにして、転送中にデータ変更がキャプチャされるようにする必要があります。

1. SYSDBA としてデータベースに接続して、管理タスクを実行します。

CDB サービス名を使用してルートコンテナ CDB$ROOT に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<CDB_NAME> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCLCDB as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCL as sysdba
```

1. ステータスを確認します。

```
archive log list;
```

ステータスが NOARCHIVELOG の場合、有効にする必要があります。

1. 高速リカバリ領域 FRA を構成し、ログを有効にします。

Tip

***2_blank_5_*- 事前に高速リカバリ領域 FRA 用のディレクトリを作成します。*1_blank_4_*- *db_recovery_file_dest_size* は、データベースのハングを防ぐために、移行のロードのピーク中に少なくとも 24 時間の REDO ログを保持するのに十分な大きさであることを確認します。 .. Tip::**
edb_notranlate_5

注釈

db_recovery_file_dest および `db_recovery_file_dest_size` の設定は、Debezium が読み取るアーカイブログに必要な保存場所である Oracle の FRA を構成します。

これらのパラメーターは、データベースの安定性にとって重要です。運用環境に設定する前に、必ず DBA に相談してください。 .. ::
2_blank_1_1. データベースを再起動し、*ARCHIVELOG* を有効にします。

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

1. 変更を確認します。

```
archive log list;
```

サプリメンタルログの有効化

サプリメンタルロギングにより、REDO ログには、DMS エージェントが変更された行を識別するための十分な情報が含まれています。

1. SYSDBA としてデータベースに接続して、管理タスクを実行します。

CDB サービス名を使用してルートコンテナ CDB$ROOT に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<CDB_NAME> as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

1. サプリメンタルログを有効にします。

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

オプションで、十分な REDO ログ領域を確保します

REDO ログのサイズが小さすぎると、頻繁なログスイッチングが発生し、移行が遅くなるか不安定になる可能性があります。REDO ログがデータベースの変更ボリュームと比べて小さい場合は、移行を開始する前にサイズと数を増加します。

詳細情報…

データ移行プロセスは、2 つの主なフェーズで構成されています。データの一貫したスナップショットと、それに続く継続的な変更ストリーム。この変更ストリームは、LogMiner および Oracle データベースの REDO ログを利用しています。

REDO ログの変更の有効期間は制限されています。これは、ログのサイズと数、およびデータベースの変更のスループットによって決まります。REDO ログが小さすぎると、頻繁なログスイッチが発生する可能性があり、移行パフォーマンスに悪影響を与える可能性があります。

1. ルートコンテナまたはデータベースインスタンスに SYSDBA として接続します。

ルートコンテナに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<CDB_NAME> as sysdba
```

データベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

1. 現在のログファイル構成を表示するには、次の SQL クエリーを実行します。

```
SELECT GROUP#, TYPE, MEMBER FROM V_$LOGFILE;
__OUTPUT__
    GROUP# TYPE    MEMBER
---------- ------- --------------------------------------------------
         1 ONLINE  /opt/oracle/oradata/ORCLCDB/redo03.log
         2 ONLINE  /opt/oracle/oradata/ORCLCDB/redo01.log
         3 ONLINE  /opt/oracle/oradata/ORCLCDB/redo04.log
```

```
SELECT GROUP#, ARCHIVED, BYTES/1024/1024 MB, STATUS FROM V_$LOG;
__OUTPUT__
    GROUP# ARC  MB STATUS
---------- --- --- ----------------
         1 YES 2000 INACTIVE
         3 YES 2000 INACTIVE
         3 NO  2000 CURRENT
```

たとえば、出力には、それぞれのサイズが 2000MB の 3 つのロググループが表示される場合がありますが、これは多くの運用環境にとって小さすぎる場合があります。

1. 必要に応じて、次のようなコマンドを使用して REDO ログを調整できます。

```
ALTER DATABASE ADD LOGFILE GROUP 4 (/opt/oracle/oradata/ORCLCDB/redo04.log) SIZE 8G;
ALTER DATABASE ADD LOGFILE GROUP 5 (/opt/oracle/oradata/ORCLCDB/redo05.log) SIZE 8G;
ALTER DATABASE ADD LOGFILE GROUP 6 (/opt/oracle/oradata/ORCLCDB/redo06.log) SIZE 8G;
ALTER DATABASE ADD LOGFILE GROUP 7 (/opt/oracle/oradata/ORCLCDB/redo07.log) SIZE 8G;
ALTER SYSTEM ARCHIVE LOG CURRENT;
ALTER SYSTEM CHECKPOINT;
ALTER SYSTEM ARCHIVE LOG CURRENT;
ALTER SYSTEM CHECKPOINT;
ALTER SYSTEM ARCHIVE LOG CURRENT;
ALTER SYSTEM CHECKPOINT;
ALTER DATABASE DROP LOGFILE GROUP 1;
ALTER DATABASE DROP LOGFILE GROUP 2;
ALTER DATABASE DROP LOGFILE GROUP 3;
```

これらのコマンドは、それぞれが単一の REDO ログファイルを含む 4 つの 8GB REDO ロググループを作成します。

> **注釈**
>
> 常に DBA に相談して、本番環境に適切な REDO ログサイズを決定します。

テーブルの構成

テーブルレベルのサプリメンタルログの有効化

データベースレベルのロギング前の手順で有効にしたことは基盤を提供しますが、完全な列ロギングを必要とする個々のテーブルも指定する必要があります。これにより、DMS エージェントは、すべての行変更の_before_および_after_イメージを受信します。

- Oracle の場合、_schema_および_user_という用語は同じ意味で使用されます。データ移行の SQL コマンドを実行するときは、エラーを回避するために、必ずスキーマ名を正しいユーザーアカウントに設定してください。
- **Oracle XE の互換性制限については、** ***Limitations*** **を参照してください。 .. ::**
  <details><summary>詳細情報... </summary>

サプリメンタルロギングとは、_before_state などの Oracle REDO ログの追加情報のキャプチャを指します。この追加の REDO ログ情報は、EDB DMS のようなログベースのアプリケーションが変更イベントをキャプチャするために必要です。詳細については、Oracle の Supplemental Logging documentation を参照してください。

データの移行に Debezium を使用するには、データベースレベルとテーブルレベルの両方でサプリメンタルログを有効にする必要があります。これは、Debezium が変更データを正しくキャプチャするために重要です。

- **データベースレベルのサプリメンタルログ**は、変更データキャプチャ CDC の基礎です。これにより、すべての変更メッセージに追跡番号が含まれます。通常、これは変更された行の主キーです。これにより、どのレコードが変更されたかを一意に識別できます。これがないと、Debezium は変更イベントをソース行にリンクできません。
- **テーブルレベルのサプリメンタルログ**は、詳細情報を提供します。これにより、変更メッセージには行全体の変更前イメージ、つまり、変更が発生する前のすべての列の完全なコピーが含まれます。Debezium は変更前と後の状態を比較できるため、これは何が変更されたかを正確に理解するのに特に役立ちます。これがないと、Debezium は更新された列のみをキャプチャし、変更の完全なコンテキストをキャプチャしません。

1. データベースに接続します。

テーブルレベルのロギングは、データが存在するプラガブルデータベース PDB 内で有効にする必要があります。CDB サービス名を SYSDBA として使用して、データが存在する PDB に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<PDB_NAME> as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SERVICE_NAME_OR_SID> as sysdba
```

1. 特定のテーブルのロギングを有効にします。移行するすべてのテーブルに対してこのコマンドを実行する必要があります。

```
ALTER TABLE <SCHEMA_NAME>.<TABLE_NAME> ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

例

```
ALTER TABLE HR_APP.EMPLOYEES ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
ALTER TABLE FINANCE_APP.SALARIES ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

```
SELECT ALTER TABLE  || owner || . || table_name ||  ADD SUPPLEMENTAL LOG DATA (ALL)␣
↪COLUMNS;
FROM dba_tables
WHERE owner = YOUR_SCHEMA_1;
```

出力をコピーして実行して、スキーマ全体を一度に更新します。

データ移行サービス DMS と基になる Debezium エンジンには、内部メタデータ、ログマイニングの詳細、および追跡オフセットを保存するための専用のテーブルスペースが必要です。これにより、移行関連のオーバーヘッドがアプリケーションデータから分離されます。

初期サイズ 25MB はテスト用の最小値です。実稼働環境では、通常 512MB 以上が必要です。

ファイルパス サーバー上の.dbf ファイルの正しいディレクトリを決定するには、常にデータベース管理者 DBA に相談してください。

マルチテナント環境では、グローバルログマイニングのためにルートコンテナ CDB にテーブルスペースを作成し、テーブル固有の追跡のためにプラガブルデータベース PDB にテーブルスペースを作成する必要があります。

1. sysdba として CDB に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<CDB_NAME> as sysdba;
```

1. CDB テーブルスペースを作成します。

```
CREATE TABLESPACE <TS_NAME>
DATAFILE <CDB_PATH>/<TS_NAME>.dbf
SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

1. PDB に切り替えます。

```
ALTER SESSION SET CONTAINER = <PDB_NAME>;
```

1. PDB テーブルスペースを作成します。

```
CREATE TABLESPACE <TS_NAME>
DATAFILE <PDB_PATH>/<TS_NAME>.dbf
SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

Tip

両方の表領域に同じ名前たとえば *LOGMINER_TBS* を使用すると、後の構成が簡素化されます。 !!!

ログマイニングとメタデータ追跡の両方を処理するテーブルスペースを作成します。

```
CREATE TABLESPACE <TS_NAME>
DATAFILE <PATH>/<TS_NAME>.dbf
SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

詳細情報…

データの移行を実行するには、DMS とその基になる Debezium ライブラリの両方が専用のユーザーが必要です。このユーザーを作成する前に、必要なテーブルスペースを設定します。

CDB アーキテクチャの場合、2 つのテーブルスペースが必要です。

- **CDB レベルのテーブルスペース**このテーブルスペースは、Debezium の内部プロセス用です。これは、ログマイニングおよびコンテナデータベースレベルで実行する必要があるその他の管理操作に使用されます。
- **PDB レベルのテーブルスペース**これは、プラガブルデータベースに固有の Debezium 操作用です。その PDB 内でキャプチャしている個々のテーブルに関連するプロセスを処理します。

移行ユーザーの構成

移行プロセスを簡素化するために、単一の共通ユーザーマルチテナント CDB / PDB 構成の場合は C##または専用のシステムユーザー非 CDB 構成の場合を作成することをお勧めします。この統合ユーザーは、EDB Postgres AI エージェントを介したスキーマ評価と DMS エージェントを介したライブデータの移行の両方を処理します。

1. ルートコンテナまたはデータベースインスタンスに SYSDBA として接続します。

CDB サービス名を使用してルートコンテナ CDB$ROOT に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<CDB_NAME> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCLCDB as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

1. 移行用の新しいユーザーを作成します。

```
CREATE USER <C##_MIGRATION_USER> IDENTIFIED BY <MIGRATION_USER_PASSWORD> DEFAULT␣
↪TABLESPACE <TS_NAME> QUOTA UNLIMITED ON <TS_NAME> CONTAINER=ALL;
```

```
CREATE USER <MIGRATION_USER> IDENTIFIED BY <MIGRATION_USER_PASSWORD> DEFAULT␣
↪TABLESPACE <TS_NAME> QUOTA UNLIMITED ON <TS_NAME>;
```

1. 次のすべての特権を付与します。

```
-- Grant core session and container privileges
-- These grants are required for schema ingestion and assessment via the EDB Postgres␣
↪AI agent.
GRANT CREATE SESSION TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SET CONTAINER TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$DATABASE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT CONNECT TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT_CATALOG_ROLE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT CREATE TABLE TO <C##_MIGRATION_USER> CONTAINER=ALL;

-- Grant migration and log mining permissions
-- These additional grants are required for data migration via the DMS agent.
GRANT FLASHBACK ANY TABLE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ANY TABLE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT EXECUTE_CATALOG_ROLE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ANY TRANSACTION TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ANY DICTIONARY TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT LOGMINING TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT LOCK ANY TABLE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT CREATE SEQUENCE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT EXECUTE ON DBMS_LOGMNR TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT EXECUTE ON DBMS_LOGMNR_D TO <C##_MIGRATION_USER> CONTAINER=ALL;

-- Grant access to system views.
-- Note: several of these views are also accessible via SELECT_CATALOG_ROLE,
-- granted above. The explicit grants are intentional to ensure compatibility
-- across Oracle patch levels and to make the required access unambiguous.
GRANT SELECT ON V_$LOGMNR_LOGS TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$LOGMNR_CONTENTS TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$LOGFILE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$ARCHIVED_LOG TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$ARCHIVE_DEST_STATUS TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$TRANSACTION TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$VERSION TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$INSTANCE TO <C##_MIGRATION_USER> CONTAINER=ALL;
GRANT SELECT ON V_$LICENSE TO <C##_MIGRATION_USER> CONTAINER=ALL;

ALTER USER <C##_MIGRATION_USER> QUOTA UNLIMITED ON <TS_NAME> CONTAINER=ALL;
```

PDB に切り替えます。

```
ALTER SESSION SET CONTAINER = <PDB_NAME>;
```

ユーザーのデフォルトのテーブルスペースを設定します。

```
ALTER USER <C##_MIGRATION_USER> DEFAULT TABLESPACE <TS_NAME>;
```

```
-- Grant core session privileges
-- These grants are required for schema ingestion and assessment via the EDB Postgres␣
↪AI agent.
GRANT CREATE SESSION TO <MIGRATION_USER>;
GRANT SELECT ON V_$DATABASE to <MIGRATION_USER>;
GRANT CONNECT TO <MIGRATION_USER>;
GRANT SELECT_CATALOG_ROLE TO <MIGRATION_USER>;
GRANT CREATE TABLE TO <MIGRATION_USER>;

-- Grant migration and log mining permissions
-- These additional grants are required for data migration via the DMS agent.
GRANT FLASHBACK ANY TABLE TO <MIGRATION_USER>;
GRANT SELECT ANY TABLE TO <MIGRATION_USER>;
GRANT EXECUTE_CATALOG_ROLE TO <MIGRATION_USER>;
GRANT SELECT ANY TRANSACTION TO <MIGRATION_USER>;
GRANT SELECT ANY DICTIONARY TO <MIGRATION_USER>;
GRANT LOGMINING TO <MIGRATION_USER>;
GRANT LOCK ANY TABLE TO <MIGRATION_USER>;
GRANT CREATE SEQUENCE TO <MIGRATION_USER>;
GRANT EXECUTE ON DBMS_LOGMNR TO <MIGRATION_USER>;
GRANT EXECUTE ON DBMS_LOGMNR_D TO <MIGRATION_USER>;

-- Grant access to system views.
-- Note: several of these views are also accessible via SELECT_CATALOG_ROLE,
-- granted above. The explicit grants are intentional to ensure compatibility
-- across Oracle patch levels and to make the required access unambiguous.
GRANT SELECT ON V_$LOGMNR_LOGS TO <MIGRATION_USER>;
GRANT SELECT ON V_$LOGMNR_CONTENTS TO <MIGRATION_USER>;
GRANT SELECT ON V_$LOGFILE TO <MIGRATION_USER>;
GRANT SELECT ON V_$ARCHIVED_LOG TO <MIGRATION_USER>;
GRANT SELECT ON V_$ARCHIVE_DEST_STATUS TO <MIGRATION_USER>;
GRANT SELECT ON V_$TRANSACTION TO <MIGRATION_USER>;
GRANT SELECT ON V_$VERSION TO <MIGRATION_USER>;
GRANT SELECT ON V_$INSTANCE TO <MIGRATION_USER>;
GRANT SELECT ON V_$LICENSE TO <MIGRATION_USER>;

ALTER USER <MIGRATION_USER> QUOTA UNLIMITED ON <TS_NAME>;
```

1. 移行ユーザーが CDB ルートからソーステーブルを表示できることを確認します。これで行が返されない場合、付与は CONTAINER=ALL で適用されませんでした。上記のようにすべての特権を再付与します。

移行ユーザーとして CDB ルートに接続します。

```
sqlplus <C##_MIGRATION_USER>/<MIGRATION_USER_PASSWORD>@//<HOST>:<PORT>/<CDB_NAME>
```

次に、ソーステーブルが表示されることを確認します。

```
SELECT owner, table_name FROM cdb_tables WHERE owner = <SCHEMA_NAME>;
```

この手順は、非 CDB 構成には適用されません。

1. ソーステーブルに SELECT を付与します。

SYSDBA のまま PDB に切り替えます。

```
ALTER SESSION SET CONTAINER = <PDB_NAME>
```

sysdba としてデータベースインスタンスに接続していることを確認します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

1. 特定のテーブルに SELECT 権限を付与します。移行するすべてのテーブルに対してこのコマンドを実行する必要があります。

```
GRANT SELECT ON <SCHEMA_NAME>.<TABLE_NAME> TO <C##_MIGRATION_USER/MIGRATION_USER>
```

1. デフォルトのテーブルスペースを設定します。

CDB と PDB の両方でデフォルトの表領域を変更します。

```
-- In the CDB ROOT
ALTER SESSION SET CONTAINER = CDB$ROOT;
ALTER USER <C##_MIGRATION_USER> DEFAULT TABLESPACE <TS_NAME>;

-- In the PDB
ALTER SESSION SET CONTAINER = <PDB_NAME>;
ALTER USER <C##_MIGRATION_USER> DEFAULT TABLESPACE <TS_NAME>;
```

```
ALTER USER <MIGRATION_USER> DEFAULT TABLESPACE <TS_NAME>;
```

1. SQL を終了し、移行ユーザーで簡単な接続テストを実行します。

```
sqlplus <C##_MIGRATION_USER>@//<HOST>:<PORT>/<PDB_NAME>
```

```
sqlplus <MIGRATION_USER>/<MIGRATION_USER_PASSWORD>@//<HOST>:<PORT>/<SID_OR_SERVICE>
```

1. 移行ユーザーがソーステーブルからデータを読み取ることができることを確認します。

```
SELECT * FROM <SCHEMA_NAME>.<TABLE_NAME> WHERE ROWNUM <= 5;
```

```
SELECT * FROM <SCHEMA_NAME>.<TABLE_NAME> WHERE ROWNUM <= 5;
```

データベースとユーザー構成の検証

データベース構成と作成されたユーザーがエージェントで機能するためのすべての要件を満たしていることを確認します。

1. /opt/cdcagent/reader/ の EDB DMS エージェントフォルダーに移動します。
2. <schema_name>.<table_name> 形式で移行する必要があるすべてのテーブルの名前を含む配列を作成します。例

```
arr=(test1.table1 test1.table2 test1.table3 test1.table4)
```

1. 必要なパラメーターを使用してスクリプトを実行します。例

```
ODB_SYSDBA_USERNAME=sys ODB_SYSDBA_PASSWORD=password ODB_HOST=localhost ODB_PORT=1521␣
↪ORACLE_SERVICENAME=ORCLPDB1 ODB_DBZ_USERNAME=C##MIG_USER2 ODB_DBZ_PASSWORD=migrator_
↪password LOG_MINER_TBS_NAME=LOGMINER_TBS ./oracleConfigValidation.sh "${arr[@]}"
```

注釈

ORACLE_SERVICENAME を PDB サービス名に設定します。 !!!

```
ODB_SYSDBA_USERNAME=sys ODB_SYSDBA_PASSWORD=password ODB_HOST=localhost ODB_PORT=1521␣
↪ORACLE_SERVICENAME=ORCL ODB_DBZ_USERNAME=MIG_USER2 ODB_DBZ_PASSWORD=migrator_
↪password LOG_MINER_TBS_NAME=LOGMINER_TBS ./oracleConfigValidation.sh "${arr[@]}"
```

注釈

ORACLE_SERVICENAME を SID またはサービス名に設定します。 !!!

次のような応答が得られます。

```
 ***  [Transporter] - Validate system user connectivity
[Pass] Provide username and password of system user can connect to database.

  ***  [Transporter] - Validate debezium user connectivity
[Pass] Provide username and password of debezium user can connect to database.

  ***  [Transporter] - Validate Archive Log mode
[Pass] Database log mode is Archive Mode.

  ***  [Transporter] - Validate fast recovery area parameters
[Pass] Recovery area path was set.
```

(次のページに続く)

(前のページからの続き)

```
[Pass] Recovery area size limit meets requirements.
[Pass] Recovery area remaining size meets requirements.

  ***  [Transporter] - Validate if database supplemental logging is enabled
[Pass] Supplemental logging check for database is passed.

  ***  [Transporter] - Validate if table supplemental logging is enabled
[Pass] All source tables enable supplemental logging.

  ***  [Transporter] - Validate redo log count&size
[Pass] Redo log count meets the requirements.
[Pass] Redo log size meets the requirements.
[Pass] Redo logs are all active.

  ***  [Transporter] - Validate if tablespace for migration with Limited Privileges␣
↪were created
[Pass] LOGMINER_TBS tablespace has enough datafiles.
[Pass] LogMiner DataFile Size meet the requirements.
[Pass] LogMiner DataFile Max Size meet the requirements.
[Pass] LogMiner DataFile are auto-extendable.

  ***  [Transporter] - Validate if user for migration with Limited Privileges were␣
↪created
Oracle version is 19, which is 12c or newer.
[Pass] This user has all sys privileges that migration needs.
[Pass] This user has all role privileges that migration needs.
[Pass] This user has all select privilege on specific tables that migration needs.
[Pass] This user has all execute privilege on specific tables that migration needs.

  ***  [Transporter] - Validate if user for migration with LOGMINER_TBS tablespace
[Pass] User for migration use LOGMINER_TBS tablespace.
```

データベースは CDC 移行の準備が整いました。

注釈

ここで推奨されているようにソースデータベース設定を変更することにより、*[Failed]* または`[Suggestion]`ステータスに対処します。 *[Failed]* チェックは、DMS がデータ移行を実行する前に解決する必要がある問題をブロックしています。 *[Suggestion]* チェックでは移行が許可されますが、対処しないでおくとパフォーマンスに悪影響を与える可能性があります。

詳細情報

- Debezium が Oracle REDO ログから変更をキャプチャする方法と、それに必要な特定のデータベース構成を理解するには、

Debezium Oracle Connector ドキュメントを参照してください。

- Oracle の

Enabling the Fast Recovery Area を参照してください。

次のステップ

Configuring the EDB Postgres AI agent to register your Oracle database

Configuring the EDB Postgres AI agent to register your Oracle database

EDB Postgres AI エージェント `beacon-agent` としてパッケージ化して、HM との安全な接続を確立し、Oracle ソースからスキーマと移行評価データを取得します。

前提条件

- **マシンユーザーとアクセスキー** *Creating a machine user* で作成されます。アクセスキーを環境変数に保存します。

```
export BEACON_AGENT_ACCESS_KEY="baak_your_key_here"
```

- **SQL*Plus** を使用した Oracle インスタントクライアント EDB Postgres AI エージェントは、SQL *Plus を使用して Oracle ソースから DDL を抽出します。 Oracle Instant Client と SQL* Plus をインストールしていることを確認し、補助マシンに必要な環境変数を構成します。手順については、*Configuring your auxiliary machine or bastion host* を参照してください。

> **注釈**
>
> エージェントは、実行時に `beacon_agent.yaml` で参照される環境変数を置き換えるため、エージェントを実行する前に環境で `$BEACON_AGENT_ACCESS_KEY` を設定するだけで十分です。環境変数は、YAML の静的値よりも常に優先されます。

環境変数を設定するための推奨事項

`$BEACON_AGENT_ACCESS_KEY` や `$DSN` などの機密データは、シークレットマネージャーで管理される環境変数として扱うことをお勧めします。この戦略により、機密の資格情報がプレーンテキストの構成ファイルに保存されるのを防ぎます。環境変数を使用する場合、EDB Postgres AI エージェントの実行中にそれらが利用できることを確認してください。

実稼働環境のための最も安全なアプローチについて、組織の IT 部門に相談してください。

短期間の移行またはテストとデモの場合、アクティブなターミナルセッションで `export` コマンドを使用できます。

HM 接続パラメーターの取得

エージェントには、HM インストールからの特定の値が必要です。次のことについては、管理者に連絡してください。

- **ビーコンサーバーホスト名** beacon_server の内部エンドポイント。

HM インストールを構成するときに、管理者/インストーラーは beacon_server サービスの内部エンドポイントを指定する必要があります。HM インスタンスの管理者またはインストーラーに、インストールに使用される Helm チャート構成ファイルで beacon_server サービスに設定したホスト名 URL を提供するように依頼します。このために、values.yaml ファイルの parameters.upm-beacon.server_host または BEACON_SERVICE_DOMAIN_NAME に設定した値を検索します。この値は、後で EDB Postgres AI エージェントを構成するために必要になります。

- **ルート証明書パス必要な場合** HM インスタンスが自己署名または内部 CA を使用する場合、ca_public.crt ファイルを取得します。エージェントマシンたとえば/etc/beacon/ca_public.crt に保存します。

EDB Postgres AI エージェントには、HM インスタンスの beacon_server サービスへの信頼できる接続が必要です。 EDB Postgres AI エージェントを実行しているサーバーが HM の TLS 証明書組織のセキュリティインフラストラクチャによって管理されているを本質的に信頼していない場合、HM インスタンスの管理者またはインストーラーに証明書を含むファイルを提供するように依頼してください。この証明書は、EDB Postgres AI エージェントを実行しているマシンに保存します。後でこの証明書へのディレクトリパスを提供して、EDB Postgres AI エージェントを構成する必要があります。

beacon_server およびルート証明書の詳細については、*Administrative tasks* を参照してください。

接続の検証

HM 接続パラメーターを取得したら、補助マシンが Oracle ソースとハイブリッドマネージャーの両方に到達できることを確認します。次のポートを開く必要があります。

Destination	Port	Used by
Oracle host	1521	Both agents (database connection)
HM *beacon_server*	9443	EDB Postgres AI agent
HM DMS service (*RW_SERVICE_HOST*)	443 (HTTPS)	DMS agent

nc を使用して、各ホストとポートへの接続を確認します。

```
#  Oracle Listener

nc -zv <oracle-host-name-or-ip> 1521

#  HM beacon server

nc -zv <beacon_server_hostname> 9443
```

(次のページに続く)

(前のページからの続き)

```
#  HM DMS service

nc -zv <rw-service-hostname> 443
```

接続が成功すると succeeded! が返されます。 Connection refused または Operation timed out が表示される場合、ポートは到達不能であるため、続行する前に開く必要があります。

ソースデータベース接続パラメーターの定義

データソース名 DSN は、Oracle ソースの接続文字列です。

共通ユーザー C##で始まるを使用して、特定のプラガブルデータベース PDB サービスに接続します。

> **注釈**
>
> ユーザー名の#文字は、%23 として URL エンコードする必要があります。

フォーマット

```
DSN="oracle://<C##_MIGRATION_USER>:<MIGRATION_USER_PASSWORD>@<HOST>:<PORT>/<PDB_
↪SERVICE_NAME>"
```

例

```
DSN="oracle://C%23%23_MIG:password@localhost:1521/ORCLPDB1"
```

標準のローカルユーザーを使用して、データベースインスタンスの SID またはサービス名に接続します。

フォーマット

```
DSN="oracle://<MIGRATION_USER>:<MIGRATION_USER_PASSWORD>@<HOST>:<PORT>/<SERVICE_NAME_
↪OR_SID>"
```

例

```
DSN="oracle://MIG_USER:password@localhost:1521/ORCL"
```

```
DSN="oracle://oracle:pa%24%24word@localhost:1521/ORCLPDB1"
```

構成ファイルの準備

エージェントの構成ファイルを準備します。エージェントは/etc/beacon/ または${HOME}/.beacon/ で beacon_agent.yaml を検索します。

1. ホームディレクトリに EDB Postgres AI エージェント構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

> **注釈**
>
> エージェントは`/etc/beacon/` または`${HOME}/.beacon/` で`beacon_agent.yaml` を検索します。最終的なフォールバックとして、実行されたディレクトリを検索します。 !!!

1. このディレクトリ内に、ファイル beacon_agent.yaml を作成します。

```
touch ${HOME}/.beacon/beacon_agent.yaml
```

1. 次のテンプレートをコピーして、新しいファイルに貼り付けます。

このテンプレートでは、 $BEACON_AGENT_ACCESS_KEY および$DSN は、前に構成した環境変数です。テンプレート内のすべての< >プレースホルダーを置き換えます。詳細は、*EDB DMS agent parameter reference* を参照してください。

```
---
agent:
  access_key: $BEACON_AGENT_ACCESS_KEY
  beacon_server: <beacon_server_hostname>:9443
  project_id: <your_project_id>
  providers:
    - "onprem"
  root_ca_path: ""
  # In most HM deployments, you need to set this to the path of the ca_public.crt
  # file obtained from your HM administrator (for example, /etc/beacon/ca_public.crt).
  # Leave empty only if the HMs certificate is already trusted by the systems CA␣
↪bundle.
  schema_providers:
    - "onprem-schema"
    # Include "onprem-schema" to enable schema ingestion and migration assessment.
    # Remove this section if you dont want schema information to be collected.
provider:
  onprem:
    databases:
      - resource_id: "<database_resource_identifier>"
        # Controls how the auxiliary machine appears in HM Estate.
        dsn: $DSN
        schema:
          enabled: true
          poll_interval: 15s
        tags:
          - "<tag_names>"
general:
  logging:
    level: info
    # Use debug during initial setup or troubleshooting for more verbose output.
```

1. ファイルを保存します。

EDB Postgres AI エージェントの実行

1. Oracle Instant Client 環境変数が、エージェントを実行するのと同じシェルセッションで設定されていることを確認します。

```
export ORACLE_HOME=</path/to/instantclient>
export PATH=$ORACLE_HOME:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME:$LD_LIBRARY_PATH
```

1. サービスを開始する前に、接続と構文を確認します。

```
beacon-agent validate
```

1. エージェントを実行します。

```
beacon-agent
```

Tip

他のエージェントモードとオプションについては、 agent CLI を参照してください。 !!!

1. エージェントがハイブリッドマネージャー HM へのデータの送信を開始するときに、取り込みプロセスを監視します。数分後、Oracle データベースが HM コンソールの**エステート>移行** の下に表示されます。

重要

エージェントは、テーブルと列の変更を動的に検出します。ただし、ビュー、プロシージャー、またはファンクションを変更するには、エージェントを再起動してメタデータを再取り込む必要があります。

本番使用の場合は、 *Running the agent as a service* を参照してください。

移行評価の検証

Oracle データベースがエージェントに認識され、移行の準備ができていることを確認するには、HM コンソールで **Estate > Migrations** に移動します。

データベースがテーブルに表示される場合、次のリソースにアクセスできます。

- **Estate** ページの **マイグレーション** の **移行評価**。データベーススキーマ数、コア数、移行の複雑さ、および作業レベルの概要を提供します。
- ** *Viewing migration assessment details* ** [Migrations **] タブから特定のデータベースを選択してアクセスできます。このビューは、** スキーマ 、**グローバル** 評価、**および移行を実行するための他の HM ツールに直接接続する**移行**ハブの拡張分析を提供します。

- ビルトインの移行ポータルでのスキーマの移行に使用できる**自動スキーマ DDL 抽出**。取り込みが完了すると、データベースごとに *Migration Portal automatically creates a project* 。これを使用して、互換性のないスキーマを変換し、宛先データベースに移行できます。

障害対応

エージェントが HM または Oracle データベースへの接続に失敗する理由はいくつかあります。ここでは、トラブルシューティングに使用できるツールをいくつか紹介します。

beacon-agent validate コマンド

`beacon-agent validate` コマンドは、一般的な接続と構成の問題を特定するための役立つ手順です。

```
beacon-agent validate
```

validate コマンドは、次のことをチェックします。

- **サーバー接続** エージェントが構成済みの `beacon_server` 通常はポート `9443` に接続および認証できることを確認します。
- **データベース接続**構成されている各データベース DSN への接続を試行します。これは完全な接続テストであり、単なる構文または URL エンコーディングのチェックではありません。
- **構成表示** TLS 設定、プロジェクト ID、プロバイダー、およびデータベースごとの設定を表示し、構成の誤りを特定するのに役立ちます。

beacon-agent ログ

エージェントが取り込み中にエラーを報告した場合、詳細についてはログを確認できます。エージェントをフォアグラウンドで実行している場合、ログはターミナルに出力されます。エージェントを systemd サービスとして設定する場合、`journalctl` を使用してログを表示します。

次に、ログに表示される一般的なエラーメッセージをいくつか示します。

Error	Cause	Fix
sqlplus is not installed or not found in PATH	Oracle Instant Client environment variables are not set or set incorrectly in the current session.	Set *PATH* and *LD_LIBRARY_PATH* to the Instant Client root directory and verify with *which sqlplus*. See *Configuring your auxiliary machine or bastion host* .
extractor permissions check failed: sqlplus execution failed: exit status 127	*sqlplus* binary is found but cannot load its shared libraries.	Ensure *LD_LIBRARY_PATH* points to the Instant Client root directory (not a *lib/* subdirectory).

次のページに続く

表 26 – 前のページからの続き

Error	Cause	Fix
tls: failed to verify certificate: x509: certificate signed by unknown authority	The agent cannot verify HM's TLS certificate because the CA is not trusted by the system.	Set *root_ca_path* in *beacon_agent.yaml* to the path of the *ca_public.crt* file obtained from your HM administrator. If you cannot obtain the certificate, you can set *plaintext: true* in the agent config as a temporary workaround (not recommended for production).
rpc error: code = Unknown desc = no rows in result set	The beacon-agent version is incompatible with the server.	Upgrade to beacon-agent v1.384.0 or later.

エージェントをサービスとして実行する

継続的なメトリックストリーミングのために、そして Oracle データベースが **HM コンソール>エステート**に表示されたままであることを確認するには、systemd サービスとして実行するようにエージェントを構成します。サービスとして実行すると、システムの起動時に自動的に起動し、エラーが発生した後に再起動します。

サービスがないと、エージェントは手動で実行している間のみメトリックをストリーミングします。データベースは、エージェントがストリーミングを停止してから 72 時間後に HM 不動産から消えます。

> **注釈**
>
> *beacon_agent.yaml* たとえば *$BEACON_AGENT_ACCESS_KEY* や `$DSN` などで参照される環境変数が、実行時にサービスにアクセスできることを確認します。

1. サービスファイルを作成します。この例では `vi` を使用しますが、任意のテキストエディタを使用できます。

```
sudo vi /usr/lib/systemd/system/beacon-agent.service
```

オペレーティングシステム別のデフォルトのサービスファイルパス

Operating system	Default path
RHEL	*/usr/lib/systemd/system/beacon-agent.service*
Rocky Linux	*/usr/lib/systemd/system/beacon-agent.service*
AlmaLinux	*/usr/lib/systemd/system/beacon-agent.service*
Ubuntu	*/etc/systemd/system/beacon-agent.service*
Debian	*/lib/systemd/system/beacon-agent.service*

1. 次のテンプレートをファイルに貼り付けて、すべての< >プレースホルダーを置き換えます。

```
[Unit]
Description=Agent
```

(次のページに続く)

(前のページからの続き)

```
After=network.target

[Service]
Type=simple
User=<user>
WorkingDirectory=</home/user/>
Environment="PATH=</path/to/oracle/instantclient>:</user/binaries>:</system/binaries>"
Environment="ORACLE_HOME=</path/to/oracle/instantclient>"
ExecStart=</usr/local/bin/beacon-agent>
Restart=on-failure
RestartSec=60

[Install]
WantedBy=multi-user.target
```

Setting	Placeholder	Description
User=	*<user>*	エージェントを実行するシステムユーザー。
WorkingDirectory=	*</home/user/>*	`.beacon`フォルダーが保存されるディレクトリ。パスはオペレーティングシステムによって異なります。
Environment=	*"PATH=</path/to/oracle/instal*	Oracle 実行可能ファイルを含むすべてのディレクトリ。エージェントは実行時にスキーマ抽出のために`sqlplus`を見つけることができます。
Environment=	*"ORACLE_HOME=</path/to/oracle*	Oracle Instant Client のルートディレクトリ。スキーマ抽出のために`sqlplus`が必要。
ExecStart=	*</usr/local/bin/beacon-agent>*	`beacon-agent`バイナリへのフルパス。パスは、エージェントがインストールされた方法と場所によって異なります。

1. systemd をリロードし、サービスを有効にして開始します。

```
sudo systemctl daemon-reload
sudo systemctl enable beacon-agent.service
sudo systemctl start beacon-agent.service
```

1. ログを確認して、サービスが正常に開始されたことを確認します。

```
journalctl -u beacon-agent.service
```

関連トピック

- スキーマ取り込みのより詳細な制御たとえば、スキーマを除外または含める場合、*Controlling schema ingestion* を参照してください。
- 管理者がルート証明書と `beacon_server` の値を取得する方法に関するガイダンスが必要な場合は、

Administrative tasks を参照してください。

次のステップ

Preparing your HM-managed destination database

Preparing your HM-managed destination database

Oracle の移行の宛先として機能する HM 管理データベースクラスターを準備します。

> **注釈**
>
> 宛先データベースクラスターは、 ** EDB Postgres Advanced Server**ディストリビューションを使用する必要があります。 HM のビルトインスキーマ移行機能移行ポータルと EDB DMS では、Oracle から移行する場合に EDB Postgres Advanced Server が必要です。

ハイブリッドマネージャー HM に宛先データベースクラスターがまだない場合は、作成します。

1. HM コンソールで、データベースクラスターを作成する "プロジェクトを選択できません" — ポータルドメインはデフォルトで API エンドポイントになります を選択します。
2. プロジェクトで、 **Create New** を選択し、 **Database Cluster** から *Create a database cluster template* を選択します。ディストリビューションを選択するようにプロンプトが表示されたら、**EDB Postgres Advanced Server** を選択します。

> **注釈**
>
> EDB Postgres Advanced Server 18 以降のクラスターを作成している場合、 *ssl_groups* データベースパラメーターを更新して、デフォルト値から `X25519` グループを削除する必要があります。 *X25519* は FIPS に準拠しておらず、移行ポータルの宛先データベースへの接続には FIPS が必要です。 !!!

1. **Clusters** ページで、クラスターを選択し、 **Quick Connect** オプションを使用してターミナルからデータベースにアクセスします。
2. 移行の宛先として使用する空のデータベースを作成します。または、デフォルトのデータベース `edb_admin` を使用できます。

すでに HM 管理の EDB Postgres Advanced Server データベースクラスターがある場合は、移行の宛先として使用できます。

続行する前に、次のことを確認します。

- クラスターは **EDB Postgres Advanced Server** ディストリビューションを使用します。別のディストリビューションを使用する場合、代わりに新しい EDB Postgres Advanced Server クラスターを作成します。
- 移行するテーブル、および移行中に生成される負荷は、クラスター上の既存のワークロードを干渉しません。

次のステップ

Migrating Postgres schemas

Migrating schema from Oracle to an HM-managed database

データを移行する前に、スキーマが宛先データベースに存在する必要があります。これを処理するには 2 つの方法があり、どちらも移行ポータルプロジェクトを開き、互換性の問題を解決することから始まります。

- **簡素化**移行ポータルを使用して非互換性のみを解決し、HM に移行の一部としてスキーマをインポートさせます。HM は、コンストレインのオーダーを自動的に管理します。手動のコンストレイン処理は必要ありません。移行を作成するときに、移行スコープとして**スキーマとデータ** を選択します。
- **手動**移行ポータルを使用して非互換性を解決し、スキーマを宛先に直接インポートします。制約の順序付けを自分自身で管理し、データの移行が完了した後、遅延制約を再適用する必要があります。移行を作成するときに移行スコープとして**データのみ** を選択します。

3 番目のスコープオプション **スキーマのみ**は、HM で移行を作成するときに使用できます。テーブルデータを移行せずにデータベース構造を移行する場合、これを使用します。たとえば、完全な移行をコミットする前に宛先のスキーマを検証します。

Migration Portal プロジェクトを開く

EDB Postgres AI エージェントは、Oracle ソースから DDL を自動的に抽出し、移行ポータルで利用できるようにします。移行ポータルを使用して、Oracle スキーマを変換および移行します。

前提条件

- EDB Postgres AI エージェントを介して *Configuring the EDB Postgres AI agent to register your Oracle database* します。
- オプション HM 管理者 *enabled and configured the AI Copilot* 移行ポータルでのスキーマの非互換性の解決を支援します。

プロジェクトへのアクセス

Oracle データベースを EDB Postgres AI エージェントに登録すると、移行ポータルプロジェクトが**自動的に作成されます** `beacon_agent.yaml` の `resource_id` に基づいて、登録されたデータベースごとに 1 つです。

> **警告**
>
> 移行ポータルプロジェクトの作成後に行われた DDL の変更新しいスキーマ、新しいテーブル、更新されたテーブル、または削除されたテーブルを含む-は、自動作成プロジェクトに反映されません。

プロジェクトを見つけるには

1. Oracle データベースの詳細ビューに移動します。次のいずれかのパスからそこに到達できます。

a. **エステート>移行>[あなたのデータベース]** 。

b. **[自分のプロジェクト] > [移行] > [データベース] > [自分のデータベース]** 。

1. **Schemas** タブを選択します。
2. **MP Project** 列のプロジェクトリンクを選択して、このデータベースの移行ポータルのプロジェクトスコープのワークスペースを開きます。

プロジェクトリンクが **MP Project** 列に表示されない場合、プロジェクトは自動的に作成されません。以下のセクションを展開して、手動で作成します。

移行ポータルプロジェクトの手動作成

次の場合、プロジェクトの自動作成はトリガーされません。

- 取り込まれた DDL が 300 MB のデフォルト制限または管理者が構成した *customized limit* を超えています。
- Hybrid Manager 管理者 *disabled automatic project creation* 。

これらの場合、プロジェクトを手動で作成します。

1. **Estate** ページで、 **Migrations** タブを選択し、データベースを選択します。
2. 詳細ビューで、 **Schemas** タブを選択し、 **New MP Project** またはスキーマの横にあるプラス記号を選択します。

> **Tip**
>
> **Dependencies**列は、現在のスキーマがどのスキーマに依存しているかを示します。スキーマとそのすべての依存関係を単一のプロジェクトにグループ化して、参照先のオブジェクトが宛先に存在することを確認します。 !!!

1. プロジェクトの詳細を入力し、 **Continue** を選択します。
2. 含めるスキーマを選択し、 **Create New Project** を選択します。

> **注釈**
>
> プロジェクトの作成後に、スキーマを追加することはできません。別のプロジェクトを使用して、追加のスキーマを同じ宛先に移行します。 !!!

HM は、バックグラウンドで移行ポータルにスキーマをロードします。準備が完了すると、プロジェクトリンクが表示されます。それを選択して移行ポータルを開き、AI コパイロットまたはクイックヘルプを使用して非互換性の解決を開始します。

スキーマの変換と互換性問題の解決

移行ポータルを使用して、Oracle DDL を EDB Postgres Advanced Server と互換性のあるものに変換します。AI コパイロットとクイックヘルプを使用して、変換中に発生した互換性の問題を解決します。追加リソース

- Using Migration Portal
- *Enabling and configuring the AI Copilot in HM* 管理者タスク
- Migration Portal's AI Copilot

-

Performing a schema assessment

> **注釈**
>
> 移行された PL/ SQL 本体トリガー、プロシージャ、ファンクション、パッケージは、Oracle ソースからの非修飾オブジェクト参照を保持します。インポートする前に、変換された DDL でこれらの参照を確認して、宛先での実行時障害を回避します。詳細と回避策については、 **移行された *PL/ SQL* ボディの非修飾オブジェクト参照** を参照してください。

スキーマのインポート

移行ポータルのすべての非互換性を解決した後、*Configuring and running the EDB DMS agent* に進みます。移行を作成するときに、スコープとして**スキーマとデータ**を選択します。HM は、正しい順序で適用された制約で変換されたスキーマをインポートし、データの移行が完了すると遅延制約を自動的に再適用します。

移行ポータルの互換性の問題をすべて解決した後、スキーマを宛先データベースにインポートします。

コンストレインの処理

2 段階で制約をインポートして、移行中のパフォーマンスの低下を回避します。

- **今すぐ含める** データ移行前プライマリキー、一意、および `NOT NULL` コンストレイン。 EDB DMS では、主に 1 回だけの配信を保証するために、主キーと一意の制約が必要です。`NOT NULL` コンストレインがパフォーマンスに与える影響はごくわずかです。
- **今すぐ除外し、後で再適用** データの移行後外部キー、チェック、および除外コンストレイン。これらは、データがアクティブに移行されているときに不必要な CPU とメモリのオーバーヘッドを発生させます。

以下の手順に従う場合、移行ポータルで **コンストレイン**を無効にし、除外したものを記録して、移行の完了後にそれらを再適用できます。

> **注釈**
>
> 主キーまたは一意制約のないテーブルの場合、EDB DMS は少なくとも 1 回の配信のみを提供します。移行の検証中に重複排除を実行できます。

<Tab title=“オンライン移行推奨”“>

スキーマ、テーブル、または列の名前を変更する必要がない場合に、このオプションを使用します。

1. **移行先 > オンライン移行** を選択します。
2. スキーマを選択します。データベースの下のオブジェクトリストで、**コンストレイン** を無効にします。 **Next** を選択します。
3. 宛先クラスターの詳細を入力します。

Clusters > Postgres Clusters > [cluster name] > Connect の下でクラスターの情報を見つけます。

1. **Test Connection** を選択して宛先への接続を確認し、 **Save and continue** を選択します。
2. **Next** を選択し、インポートが完了するまで待ちます。
3. 宛先データベースに接続し、スキーマが存在することを確認します。

スキーマ、テーブル、または列を宛先の別の名前にマップする必要がある場合に、このオプションを使用します。

1. **移行先 > オフライン移行** を選択します。
2. スキーマを選択します。データベースの下のオブジェクトリストで、**コンストレイン** を無効にします。 **Next** を選択します。
3. **Download SQL file** を選択します。
4. ダウンロードした DDL でオブジェクト名を直接編集します。名前を変更したオブジェクトが従属ビュー、コンストレイン、またはその他の参照でも更新されることを確認します。詳細は、*Mapping schemas, tables, and columns* を参照してください。
5. 宛先データベースに接続し、変更した DDL ファイルを適用します。

関連トピック

- *Migration Portal's project-scoped workspace*

次のステップ

Configuring and running the EDB DMS agent

Configuring and running the EDB DMS agent

EDB DMS エージェントは、Oracle ソースに接続し、Data Migration ServiceDMS と連携してデータとスキーマの移行を実行します。 **データのみ**、 **スキーマのみ** 、および **スキーマとデータ**の移行で実行されている必要があります。実行する前に、HM で移行資格情報を作成し、Oracle 接続の詳細とともにエージェントに渡す必要があります。

移行資格情報の作成

移行資格情報は、DMS エージェントをハイブリッドマネージャー HM に対して認証します。HM プロジェクトごとに 1 セットの資格情報が必要です。

1. HM コンソールで、移行に使用しているプロジェクトを選択します。
2. **移行>資格情報** を選択します。
3. **Create Migration Credential** を選択し、名前を入力して、 **Create Migration Credential** を選択します。 `credentials.zip` ファイルがマシンにダウンロードされます。

> **注釈**
>
> 資格情報は安全に保存します。それらを失った場合は、このセットを取り消し、新しいセットを作成する必要があります。 !!!

1. ダウンロードしたファイルを解凍し、資格情報フォルダーを DMS エージェントをインストールした補助マシンたとえば$HOME/credentials/ にコピーします。別のパスを使用する場合は、スクリプト内の CREDENTIAL_DIRECTORY_PATH を一致するように更新します。

フォルダーには次のものが含まれています

DMS エージェントの構成

1. /opt/cdcagent/run-cdcagent.sh を書き込み権限で開きます。
2. 必要な変数を設定します。オプションのパラメーターはデフォルトでコメントアウトされています。環境に適用されるものはコメントを外して構成します。

詳細は、 *EDB DMS agent parameter reference* を参照してください。

```
#!/bin/bash -e
# run-cdcagent.sh
#
# This script provides a convenient place to specify
# environment variables used to configure the
# EDB Data Migration Service Agent.
#

##########################################
# DMS Agent General Configuration       #
##########################################

# This ID is used to identify DMS Agent
# and is specified by the user.
export DBCONFIG_ID=<source_name_of_choice>

# Determine the run mode of the DMS Agent
# The run mode can be set to either reader or writer
# reader mode is used to read changes from a source database
# writer mode is used to write changes to a target database
export RUN_MODE=reader

# This is the DMS backend service used by the Agent
# Obtain this value from the HM console Sources or Destinations migration pages
export RW_SERVICE_HOST=<rw_service_host>

##########################################
# DMS Agent TLS Configuration           #
##########################################

# You need to create migration credentials in EDB Postgres AI platform
# and set the path of the credentials directory
export CREDENTIAL_DIRECTORY_PATH=$HOME/credentials
```

(次のページに続く)

(前のページからの続き)

```
########################################
# DMS Agent DB Configuration           #
########################################

# You can increase the index to configure more than
# one database for the DMS Agent
export DBCONFIG_DATABASES_0__TYPE=ORACLE
export DBCONFIG_DATABASES_0__HOSTNAME=<oracle_host>
export DBCONFIG_DATABASES_0__PORT=1521

# For CDB/PDB: use CDB_NAME/PDB_NAME (for example, ORCLCDB/ORCLPDB1).
# The agent splits on "/" to identify the CDB and PDB separately.
# Using only the PDB name will cause it to be treated as the CDB, leaving the PDB␣
↪unset.
# For non-CDB: use the SID or service name (for example, ORCL)
export DBCONFIG_DATABASES_0__CATALOG=<CDB_NAME/PDB_NAME or SID>

# Use your Oracle migration user (not SYS)
export DBCONFIG_DATABASES_0__USERNAME=<migration_user>

# The password env can be set without specifying it here
# but the env structure looks like this
export DBCONFIG_DATABASES_0__PASSWORD=<password>

# Required for Schema and data or Schema only scope.
# Must match resource_id in beacon_agent.yaml.
export DBCONFIG_DATABASES_0__RESOURCEID=<resource_id>

# Table filters for databases, only tables passing the filter will be discovered and␣
↪therefore can be chosen for migration:
# - Format: a comma-separated list of tokens in the form `schema.table`
# - Wildcard: use ` *` for schema or table (for example: `SCHEMA1.* ` or `*.TABLE1`)
# - Semantics:
#    * If `includeList` is absent/empty -> all tables are included
#    * If `includeList` is present -> only tables matching at least one include␣
↪pattern are considered
#    * If `excludeList` is absent/empty -> no table is excluded
#    * If `excludeList` is present -> only tables matching at least one exclude␣
↪pattern are considered
#    * `excludeList` has higher priority: a table matching any exclude pattern is␣
↪always excluded
# Examples:
#export DBCONFIG_DATABASES_0__INCLUDELIST=SCHEMA1.*,SCHEMA2.TABLE1
#export DBCONFIG_DATABASES_0__EXCLUDELIST=SCHEMA1.IGNORED_TABLE
```

(次のページに続く)

(前のページからの続き)

```
# SSL mode: STRICT (verify-full), DEFAULT (require), WEAK (prefer/allow plaintext)
#export DBCONFIG_DATABASES_0__SSLMODE=DEFAULT

# TCP connection settings (for dead connection detection)
# Socket timeout in seconds (default: 600 for writer, 60 for reader)
#export DBCONFIG_DATABASES_0__SOCKETTIMEOUT=600
# Enable TCP keepalive probes (default: true)
#export DBCONFIG_DATABASES_0__TCPKEEPALIVE=true
# Writer-mode TCP keepalive timing for long-running operations
# used during PK application and post-snapshot deduplication
# Seconds before first keepalive probe (default: 60)
#export DBCONFIG_DATABASES_0__TCPKEEPALIVESIDLE=60
# Seconds between keepalive retries (default: 10)
#export DBCONFIG_DATABASES_0__TCPKEEPALIVESINTERVAL=10
# Max failed probes before connection is dead (default: 6)
#export DBCONFIG_DATABASES_0__TCPKEEPALIVESCOUNT=6

# Escape hatch for arbitrary JDBC driver properties (format: key=value;key=value)
# These are applied after defaults but before explicit settings (SSLMODE,␣
↪SOCKETTIMEOUT, etc.)
# Oracle examples: oracle.net.encryption_client=REJECTED;v$session.program=DMSAgent
#export DBCONFIG_DATABASES_0__JDBCOPTIONS=oracle.net.encryption_client=REJECTED;v
↪$session.program=DMSAgent

# You can increase the index to configure more than one database for the DMS Agent:
#export DBCONFIG_DATABASES_1__TYPE=ORACLE
#export DBCONFIG_DATABASES_1__HOSTNAME=<oracle_host>
#export DBCONFIG_DATABASES_1__PORT=1521
#export DBCONFIG_DATABASES_1__CATALOG=<pdb_or_sid>
#export DBCONFIG_DATABASES_1__USERNAME=<migration_user>
#export DBCONFIG_DATABASES_1__PASSWORD=<password>
#export DBCONFIG_DATABASES_1__RESOURCEID=<resource_id>

########################################
# Optional Parameters Below            #
########################################

# FIPS mode can be enabled by setting the FIPS_MODE env variable to true
#export FIPS_MODE=true

# Enable transformation for null character(0x00) in Oracle text-based types: VARCHAR2,
 LONG, CLOB, NCLOB
#export TRANSFORMATION_ENABLE_NULLCHAR_REPLACEMENT=false
# Customize the replacement string for null character(0x00)
```

(次のページに続く)

(前のページからの続き)

```
#export TRANSFORMATION_NULLCHAR_REPLACEMENT=""
# Enable transformation for special characters in column names such as: $ and -
#export TRANSFORMATION_ENABLE_SPECIAL_COLUMN_NAME_ENCODING=false

# Configure logging
# Loglevel for the entire Quarkus application and underlying components
# WARNING: Enabling this loglevel is extremely verbose
#export QUARKUS_LOG_LEVEL=DEBUG
# Loglevel for the EDB package
# Preferred mode for debugging issues related to the logic of the agent
#export QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL=DEBUG

# Passthrough debezium configuration properties
# You can set any Debezium configuration property by prefixing it with DBZ_
# for example, to change max.batch.size:
# debezium supported properties can be found here:
# for oracle: https://debezium.io/documentation/reference/stable/connectors/oracle.
↪html#oracle-connector-properties
# for postgres: https://debezium.io/documentation/reference/stable/connectors/
↪postgresql.html#postgresql-connector-properties
#export DBZ_MAX_BATCH_SIZE=204800

cd "$(dirname "$0")"

echo "Starting in mode: $RUN_MODE"

case "$RUN_MODE" in
  reader)
    java -jar ./reader/quarkus-run.jar
    ;;
  writer)
    java -jar ./writer/quarkus-run.jar
    ;;
  *)
    echo "Unknown RUN_MODE: $RUN_MODE"
    exit 1
    ;;
esac
```

DMS エージェントの実行

DMS エージェントディレクトリに移動し、構成スクリプトを実行します。

```
cd /opt/cdcagent
./run-cdcagent.sh
```

スクリプトは、必要なすべての環境変数をエクスポートし、エージェントプロセスを開始します。エージェントが実行されると、Oracle データベースを HM のソースとして登録します。次のセクションに進んで、正しく表示されることを確認します。

HM でのソースの検証

エージェントが実行されたら、Oracle データベースが HM にソースとして表示されることを確認します。

1. HM コンソールで、プロジェクトを選択します。
2. **移行>ソース** を選択します。

Oracle データベースは、名前、タイプ、エージェント ID、およびステータスとともにリストされます。このページのすべてのフィールドの説明については、**デプロイメントリソースを理解する** を参照してください。

次のステップ

Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database

Migrating data from Oracle to an HM-managed database

EDB DMS エージェントが実行され、スキーマがインポートされるか、HM インポートの準備が整うと、HM コンソールで移行を作成する準備が整います。このページでは、移行の作成、その進行状況の監視、ファイナライズ、および移行後のコンストレインの処理という完全な移行ライフサイクルを説明します。

以下の手順は、前のページで選択したアプローチによって異なります。選択したものと一致するタブを選択します。

移行の作成

移行、Oracle ソースと HM によって管理される宛先データベースクラスター間の同期が確立されます。

1. HM コンソールで、移行に使用しているプロジェクトを選択します。
2. **移行>移行** を選択します。
3. **Create Migration** を選択します。
4. 移行に意味のある名前を割り当てます—たとえば、`oracle-prod-migration`。
5. 移行のソースを選択します。これは、DMS エージェントの構成時に設定した `DBCONFIG_ID` に対応します。
6. **Scope** で、アプローチに一致するオプションを選択します。

スキーマとデータ を選択します。HM は、解決された移行ポータルプロジェクトから変換されたスキーマをインポートし、初期スナップショットを高速に保つために制約なしでテーブルを作成し、スナップショットが完了した後に主キーと一意の制約を適用し、正確に 1 回配信に必要、残りのスキーマを同時に適用しますストリーミング**ファイナライズスキーマコンカレントリー**を介して、またはカットオーバー後の **スキーマファイナライズ**ステージで自動的に。 **ストリーミング移行の仕上げ** を参照してください。

データのみを選択します。スキーマは、移行ポータルを介して宛先に既にインポートされています。

> **注釈**
>
> テーブルデータを移行せずにデータベース構造を移行する場合、**スキーマのみ**を使用できます。 !!!

1. 移行タイプを選択します。

   - *Snapshot* —データを 1 回コピーして完了します。これは、1 回限りのオフライン移行に使用します。
   - *Snapshot + Streaming* —初期スナップショットをコピーし、ソースから変更を継続的にストリーミングします。これを使用して、ダウンタイムを最小限に抑えます。

2. **Destination Type** の下で、 **HM-managed Postgres** を選択し、宛先データベースクラスターとデータベースを選択します。次に、**Next** を選択します。

3. **Review your Tables** タブで、移行に含めるテーブルと列を選択します。テーブルを展開して、その列とオプションで *map source columns to different destination column names* を表示します。列マッピングは、 **データのみ**スコープでのみ使用できます。

4. **Create Migration** を選択します。

HM コンソールには、 *new migration* が表示されます。**スナップショット+ストリーミング** を選択した場合、移行が **実行中**状態を示している間、EDB DMS エージェントは継続的にデータ変更をストリーミングします。

> **注釈**
>
> EDB DMS エージェントは、データ変更のみをストリーミングします。DDL 変更 DDL 変更はありません。

進行状況のモニタリング

移行リストビュー

Migrations ページには、プロジェクト内のすべての移行の概要が表示されます。**移行の進行状況** 列には、移行ごとに 3 つのメトリックが表示されます。

- **テーブルの合計数** — この移行に含まれるテーブルの合計
- **スナップショットテーブルが完了しました** —スナップショットステージが終了したテーブル
- **移行された行** — すべてのテーブルで移行された累積行数

移行の詳細ビュー

リストから移行を選択して、3 つのタブがある詳細ページを開きます。

- **概要** - 現在の移行状態と、ソースおよび宛先データベースに関する一般的な情報を表示します。**スナップショット+ストリーミング** の移行の場合、ステータスが**ストリーミング** のときに移行はキャッチアップされます。**スナップショット** の移行の場合、ステータスが **Completed** になると完了します。

- **Tables** —移行のために選択されたすべてのテーブルを表示します。テーブルごとに、ソースおよび宛先テーブル、スキーマ名、スナップショットステータス、マップされた列、移行された行、および移行速度を確認できます。**+** アイコンを選択してテーブルを展開し、列レベルのマッピングを表示します。
- **Error History** —この移行についてログに記録されたすべてのエラーをリストします。 **i** アイコンを選択して、特定のエラーの完全なエラーログを開きます。移行が停止または失敗した場合、ここでのアクティブなエラーは、トラブルシューティングを開始する場所を示します。

DMS エージェントログ

HM コンソールは、すべての DMS エージェントエラーを表示するわけではありません。移行がスタックしているか、進行中のテーブルがない場合は、`run-cdcagent.sh` を実行したターミナルで `ERROR` ログエントリを確認します。多くの場合、これらには根本的な原因が含まれています。一般的な問題には、DMS エージェントと HM 間のバージョンの非互換性、および接続または資格情報の障害が含まれます。

ストリーミング移行の仕上げ

> **注釈**
>
> このセクションは、**スナップショット+ストリーミング**の移行にのみ適用されます。 **スナップショット**の移行の場合、DMS は移行を自動的に完了としてマークします。

カットオーバー前のキャッチアップのモニタリング

ソースへの書き込みを停止する前に、宛先に追いつくまでの時間を与えます。**移行の詳細**ページを使用して、行数とレプリケーションラグを追跡します。デスティネーションが一貫してソースに近づいた後にのみ、カットオーバーを開始します。

スキーマのファイナライズを同時に行うことをお勧めします

Schema および data スコープを使用した場合、HM はストリームの実行中にほとんどのスキーマオブジェクトを適用できます。プライマリキー、一意制約、インデックス `CREATE INDEX CONCURRENTLY` でビルドされるため、ライブ書き込みがブロックされません、外部キー (`NOT VALID`)、チェックコンストレイン、`NOT NULL` コンストレインなどとして追加されます。外部キー検証、シーケンス同期、およびその他の軽量タスクは、カットオーバー後の**スキーマファイナライズ** ステージで実行されます。

この作業を実行すると、読み取り専用のカットオーバーウィンドウが同時に縮小されます。テーブルが大きくなり、インデックスの数が増えるほど、節約される時間は長くなります。

すべてのテーブルがスナップショットを完了し、移行がストリーミングフェーズに入ると、**移行の詳細** ページに、**ファイナライズスキーマコンカレントリー**ボタンを備えた **スキーマファイナライズ可能**アラートが表示されます。ボタンはストリーミング中は利用可能なままであるため、目的地が追いつくまでにどれだけ近いかに基づいて、瞬間を選択できます。

1. **Migration Details** ページで **Finalize Schema Concurrently** を選択します。
2. HM は、ストリーミングが継続されている間、遅延オブジェクトを適用します。移行は**実行中** 状態のままで、進行状況サブステップには**ファイナライズスキーマコンカレントリー**というラベルが付いています。

サブステップが完了したら、カットオーバーに進みます。

> **注釈**
>
> **ファイナライズスキーマコンカレントリー**は、ストリーミングが有効になっている**スキーマおよびデータ**スコープで作成された移行でのみ使用できます。

この手順をスキップすると、移行を完了としてマークした後、同じオブジェクトが連続して適用されます。スキーマの最終化ステージを参照してください。この場合、インデックスは非並列で構築され、制約は排他的ロックの下で検証されるため、カットオーバーウィンドウは大きくなります。

手動パスには適用されません。移行前にスキーマを適用し、カットオーバー後に遅延コンストレインを手動で再適用します。遅延コンストレインの適用とデータの検証 を参照してください。

カットオーバーの実行

トラフィックをソースから宛先にスイッチする準備ができたら

1. ソースデータベースのすべての書き込みを停止します。
2. HM コンソールで、デスティネーションが最後のソース変更を適用したことを確認します。
3. HM コンソールで、プロジェクトで **移行>移行** を選択します。
4. **Migration Details** ページで、 **Mark as Completed** を選択します。

これにより、ストリーミングプロセスが停止します。

スキーマの最終化ステージ

スキーマおよびデータ スコープで作成された移行の場合、移行は **Mark as Completed** の後、自動的に **スキーマファイナライズ**ステージに入ります。このステージで実行されるものは、最初にコンカレントファイナライズを実行したかどうかによって異なります。

- **同時ファイナライズを実行した場合** HM は、軽量の残りの作業外部キー検証、シーケンス同期、およびその他の遅延メタデータタスクを実行します。このステージは短いです。
- **同時ファイナライズをスキップした場合** HM は、完全な遅延セットプライマリキー、一意制約、インデックス、外部キー、チェック制約、`NOT NULL` などを適用した後、上記の軽量タスクを適用します。このステージにはかなりの時間がかかり、ソースが読み取り専用であるときに実行されます。

移行は、 **スキーマのファイナライズ**ステージが完了した後にのみ完全に完了します。進行状況は、 **移行の詳細**ページに表示されます。

データのみスコープでは、自動スキーマファイナライズは実行されません。遅延コンストレインの適用とデータの検証 に進み、移行前に延期した外部キー、チェック、および除外制約を再適用します。

遅延コンストレインの適用とデータの検証

外部キー、チェック、および除外コンストレインの再適用

HM は、 **スキーマファイナライズ**ステージ中に、すべての遅延コンストレインと同期シーケンスを自動的に適用しました。手動操作は必要ありません。スキーマの最終化ステージ を参照してください。

スキーマのインポート中に、外部キー、チェック、および除外制約を除外して、データの移行中のパフォーマンスのオーバーヘッドを回避しました。移行が完了したので、宛先データベースに接続し、これらの制約を再適用します。

主キー、一意、`NOT NULL` 制約は、移行前に既に適用されており、アクションは必要ありません。

必要に応じて行の重複を削除する

主キーも一意制約もなかったテーブルの場合、EDB DMS エージェントは少なくとも 1 回の配信を提供します。これにより、重複行が発生する可能性があります。検証手順の一部として重複排除を実行します。

移行したデータの検証

ソースデータベースと宛先データベースを比較して、すべてのデータが正常に移行されたことを確認します。これには、LiveCompare を使用できます。

カットオーバー前のアプリケーション SQL の評価

データベースの移行が完了したら、アプリケーションを EPAS に切り替える前の次のステップは、アプリケーションが実行する SQL クエリーにターゲットデータベースと互換性があることを確認することです。スキーマとデータの移行が成功した場合でも、コード、ORM、またはスクリプトに埋め込まれたアプリケーションレベルの SQL が変更なしで動作することは保証されません。

HM のアプリケーション評価を使用して、互換性のないクエリを特定し、アプリケーションのカットオーバー前に提案された変換を確認します。概要については アプリケーションの評価 を参照するか、*Creating an application record* に直接アクセスして開始します。

関連トピック

- LiveCompare —移行の完全性を確認するための行レベルのデータ比較ツール
- 用語 —スナップショット、ストリーミング、およびその他の移行概念の定義
- アプリケーションの評価—アプリケーションをカットオーバーする前に、EPAS とアプリケーション SQL の互換性を評価します

Migrate your AWS RDS/Aurora PostgreSQL database to an HM-managed database

これらは、AWS RDS for PostgreSQL または Aurora PostgreSQL データベースから Hybrid ManagerHM によって管理されるデータベースクラスターへのエンドツーエンドの移行に含まれるタスクです。このプロセスでは、2 つのエージェントを使用します。データベースを HM に登録し、評価用のスキーマを取り込む EDB Postgres と、実際のデータ移行を実行する EDB DMS エージェント `cdcagent` 。

1. ** *Configuring your auxiliary machine or bastion host* ** —両方の移行エージェントをホストするマシンを構成します。このマシンは、AWS RDS または Aurora PostgreSQL ソースと HM 間のブリッジと

して機能します。

2. ** *Preparing your AWS RDS/Aurora PostgreSQL source database* ** —必要なユーザーを作成し、スキーマ抽出権限を構成し、AWS DB クラスターパラメータグループを介して論理レプリケーションを有効にし、構成を検証します。

3. ** *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* ** — エージェントをソースデータベースと HM に接続して、スキーマの取り込みと移行評価を有効にします。

4. ** *Preparing your HM-managed destination database* ** —移行されたスキーマとデータを受信する HM 管理データベースクラスターを作成または特定します。

5. ** *Migrating Postgres schemas* ** —HM がスキーマの移行と制約の順序を自動的に処理する簡素化パスか、自分でスキーマをエクスポート、準備、およびインポートする手動パスを選択します。

6. ** *Configuring and running the EDB DMS agent* ** —移行資格情報を作成し、リーダーモードで DMS エージェントを構成して AWS RDS または Aurora PostgreSQL ソースから変更を読み取り、ソースが HM に表示されることを確認します。

7. ** *Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database* ** — HM コンソールでデータ移行を作成して実行し、その進行状況を監視し、ストリーミングをファイナライズし、移行後のコンストレインを適用します。

Configuring your auxiliary machine or bastion host

EDB Postgres AI エージェント `beacon-agent` とデータ移行サービス DMS エージェント `cdcagent` は、Postgres データベースサーバー自分自身または別の補助マシンまたは要塞ホストで実行できます。ほとんどの運用環境では、専用の補助マシンでエージェントを実行することが優先されます。これにより、データベースサーバーへの追加ソフトウェアのインストールが回避され、単一のエージェントインスタンスでマルチプルのサーバー上のデータベースに接続できます。

必要なソフトウェア

エージェントを実行する前に、補助マシンに次のソフトウェアをインストールしていることを確認します。

- **Java 17 以降** — DMS エージェント `cdcagent` に必要です。`java -version` で確認します。
- **EDB Postgres AI エージェント** `beacon-agent` データベース登録、スキーマ評価、およびスキーマの移行に必要です。*beacon-agent version compatibility matrix* を確認します。次に、 install a compatible version of the EDB Postgres AI agent 。
- **DMS エージェント** `cdcagent` - データの移行と変更データキャプチャ CDC を処理します。インストールする前に、*DMS agent version compatibility* を確認してください。次に、 *install a compatible version of the DMS agent* 。

環境の構成

スキーマの取り込みを有効にすると、`beacon-agent` は `pg_dump` を使用して Postgres ソースから DDL を抽出します。Postgres クライアントツールパッケージには、`pg_dump` ユーティリティが含まれています。マシンで既に利用可能な場合があります。最初に確認します

```
pg_dump --version
```

コマンドが見つからない場合は、ソースデータベースディストリビューションと一致する install the client tools 。

インストールしたら、pg_dump バイナリが 2 つの互換性ルールを満たしていることを確認します。

- **バージョンルール** — pg_dump メジャーバージョンは、ソースデータベースのメジャーバージョン以上である必要があります。たとえば、ソースが Postgres 16 を実行している場合、pg_dump 16 以降が必要です。エージェントは起動時にこれを検証し、バージョンに互換性がない場合はエラーを報告します。
- **ディストリビューションルール** — pg_dump バイナリは、ソースデータベースと同じ Postgres ディストリビューションから取得する必要があります。各ディストリビューションの pg_dump は、ディストリビューション固有のカタログテーブルにアクセスします。一致しないバイナリたとえば、EDB Postgres Advanced Server ソースに対するコミュニティ PostgreSQL pg_dump を使用すると、予想されるカタログオブジェクトが存在しないため失敗します。

ホストでの複数の Postgres バージョンの管理

同じマシンに複数の Postgres バージョンがある場合は、正しい pg_dump が PATH の最初に表示されることを確認します。

```
which pg_dump
pg_dump --version
```

Debian および Ubuntu システムでは、update-alternatives を使用してアクティブなバージョンを管理します。

```
sudo update-alternatives --config pg_dump
```

複数のディストリビューションまたはバージョンからのデータベースの登録

単一の beacon-agent インスタンスは、さまざまな Postgres ディストリビューションたとえば、コミュニティ PostgreSQL および EDB Postgres Advanced Server またはさまざまなメジャーバージョンからデータベースを同時に登録できます。バージョンと配布ルールはデータベースごとに適用されるため、各データベース接続は異なる pg_dump バイナリを必要とする場合があります。

必要なすべての pg_dump バイナリを補助マシンにインストールします。後で、 beacon_agent.yaml 構成ファイルの pg_bin_path 設定を使用して、各データベースエントリを正しいバイナリディレクトリをポイントできます。設定されると、その特定のデータベースのシステム PATH ルックアップをオーバーライドします。省略すると、エージェントは PATH で最初に見つかった pg_dump にフォールバックします。例

```
provider:
  onprem:
    databases:
      - resource_id: "pg16-community"
        dsn: $DSN_PG16
```

(次のページに続く)

(前のページからの続き)

```
      pg_bin_path: /usr/lib/postgresql/16/bin
      schema:
        enabled: true
    - resource_id: "epas17"
      dsn: $DSN_EPAS17
      pg_bin_path: /usr/edb/as17/bin
      schema:
        enabled: true
```

資格情報と pgpass 動作を理解する

beacon-agent を実行する OS ユーザーは、pg_dump を実行するユーザーと同じです。プロセス引数でデータベース資格情報の公開を回避するため、エージェントは、制限的なアクセス許可 0600 でパスワードを一時 pgpass ファイルに書き込み、パスワードを含まない接続文字列を pg_dump に渡します。エージェントは OS 一時ディレクトリにファイルを作成し、各抽出が完了すると自動的に削除します。

> **注釈**
>
> エージェントプロセスが予期せず強制終了された場合たとえば *SIGKILL* 、エージェントは一時 pgpass ファイルをクリーンアップしない場合があります。エージェントを実行する OS ユーザーのみがファイルを読み取ることができます。 OS は、リブート時、またはエージェントがコンテナで実行されている場合はコンテナの再起動時にそれを削除します。孤立したファイルを確認するには、*/tmp* の `.pgpass-*` と一致するファイルを探します。

次のステップ

Preparing your AWS RDS/Aurora PostgreSQL source database

Preparing your AWS RDS/Aurora PostgreSQL source database

移行を開始する前に、AWS RDS または Aurora PostgreSQL ソースデータベースを構成し、必要なユーザーを作成します。何を構成する必要があるかは、目標によって異なります。

- EDB Postgres AI エージェントを使用した **スキーマ取り込みのみ**ハイブリッドマネージャー HM でマシンユーザーを作成し、補助マシンでエージェントの専用 OS ユーザーを設定し、スキーマ抽出権限を持つ移行ユーザーを作成します。WAL レベルのデータベースの変更は必要ありません。
- Data Migration Service DMS エージェントを使用した **データ移行* AWS DB クラスターパラメータグループを介して論理レプリケーションを有効にし、必要なレプリケーション特権を付与します。

スキーマの取り込みとデータの移行の両方を実行する予定がある場合は、すべてのセクションを完了します。スキーマまたはデータの移行のみを行う予定がある場合、各セクションでユースケースに適用できるかどうかについて説明します。

> **注釈**

AWS RDS および Aurora PostgreSQL はマネージドサービスであるため、 *psql* を使用して直接または `postgresql.conf` を編集して、データベース構成を変更することはできません。 WAL レベルの設定は、AWS DB クラスターパラメーターグループを介して適用する必要があります。

移行ユーザーの作成

Tip

EDB Postgres AI エージェントを使用したスキーマの取り込みと DMS エージェントを使用したデータ移行の両方に必要です。

EDB Postgres AI エージェントと DMS エージェントの両方にサービスを提供する単一の移行ユーザーを作成します。1 つのユーザーを使用すると、エージェント全体の資格情報の管理が簡素化されます。

データベース管理アカウントを使用して、RDS または Aurora PostgreSQL ソースデータベースに接続し、移行ユーザーを作成します。

```
CREATE USER <migration_user> WITH LOGIN PASSWORD <migration_user_password>;
```

スキーマ取り込み権限の付与

Tip

EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

1. EDB Postgres AI エージェントにソースデータベースへの接続アクセスを付与します。

```
GRANT CONNECT ON DATABASE <source_database> TO <migration_user>;
```

<source_database> は、評価するデータベースの名前です。エージェントを同じサーバー上の追加データベースに接続する予定がある場合は、各データベースに CONNECT を付与します。

1. 完全なスキーマ抽出に必要なロールを付与します。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_data TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

これら 3 つのロールを組み合わせると、所有権や特権情報を含む完全なスキーマの抽出が可能になります。pg_read_all_data のみが付与されている場合、エージェントは、所有権と特権情報を使用せずに、スキーマオブジェクト定義に制限された部分抽出を実行します。

pg_read_all_data は、Postgres 10〜13 では使用できません。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

データ移行の権限の付与

> **Tip**
>
> DMS エージェントを使用したデータ移行に必要です。

1. RDS および Aurora でのレプリケーション特権を提供する rds_replication ロールを付与します。

```
GRANT rds_replication TO <migration_user>;
```

1. 移行するテーブルを含む各スキーマで移行ユーザーの使用を付与します。

```
GRANT USAGE ON SCHEMA <db_schema> TO <migration_user>;
```

1. ソーステーブルに SELECT を付与します。スキーマ全体の場合

```
GRANT SELECT ON ALL TABLES IN SCHEMA <db_schema> TO <migration_user>;
```

個々のテーブルの場合

```
GRANT SELECT ON <table_name> TO <migration_user>;
```

1. DMS エージェントがパブリケーションを作成できるように、ソースデータベースに CREATE を付与します。

```
GRANT CREATE ON DATABASE <pg_db_name> TO <migration_user>;
```

スキーマ取り込みの準備 EDB Postgres AI エージェント

> **Tip**
>
> EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

エージェントの専用 OS ユーザーの作成

beacon-agent を実行するための補助マシンに専用の OS ユーザーを作成します。この戦略により、必要のないエージェント権限の付与が回避されます。これは Linux 固有の例であり、正確なコマンドはディストリビューションによって異なる場合があります。

```
sudo useradd <agent_os_user>
```

拡張機能の移行の準備 EDB Postgres AI エージェント

> **Tip**

ソースデータベースに拡張機能がインストールされている場合、 EDB Postgres AI エージェントを使用したスキーマの移行に必要です。

スキーマの移行の一部として、 EDB Postgres AI エージェントは、ソースデータベースにインストールされているすべての拡張機能を自動的に抽出し、他のスキーマ DDL が実行される前に宛先に適用します。拡張機能のインストールに失敗した場合、スキーマ全体の移行は中止されます。

スキーマの移行を実行する前に、ソースにインストールされている拡張機能を確認し、それらが同じバージョンの宛先で利用できることを確認します。一致するバージョンで拡張機能が利用できない場合、スキーマの移行は失敗します。異なるメジャー Postgres バージョンにまたがって移行する場合、移行先で利用可能な拡張機能バージョンはソースのものと一致しない場合があり、移行障害が発生する可能性があります。

注釈

管理対象 Postgres サービス Amazon RDS または Aurora などでは、スーパーユーザーがインストールを必要とする拡張機能は、適用時にサイレントに失敗します。移行の前に、スーパーユーザー要件についてソース拡張機能を確認します。

```
SELECT name, installed_version
FROM pg_available_extensions
WHERE installed_version IS NOT NULL;
```

バージョンを比較する宛先で同じクエリーを実行します。

移行の完了後、ソースが外部データラッパー FDW 拡張機能を使用している場合、エージェントは FDW 拡張機能自分自身をインストールしますが、FDW オブジェクトは移行しません。宛先で外部サーバー、ユーザーマッピング、および外部テーブルを手動で再作成します。

拡張機能の移行の制限の完全なリストについては、**拡張機能の移行の制限** を参照してください。

データ移行の準備 DMS エージェント

Tip

DMS エージェントを使用したデータの移行に必要です。

論理レプリケーションの有効化

DMS エージェントを使用するには、ソースインスタンスで論理レプリケーションを有効にする必要があります。RDS と Aurora はマネージドサービスであるため、 `postgresql.conf` を直接編集するのではなく、AWS DB クラスターパラメーターグループを介してこの機能を有効にします。

1. AWS マネジメントコンソール、AWS CLI、または RDS API を使用した Create a DB cluster parameter group 。パラメーターグループファミリーが、インスタンスが実行されている Postgres バージョンと一致することを確認します。
2. Associate the parameter group with your source RDS or Aurora instance 。

3. パラメーターグループ

   - の Modify the parameters `rds.logical_replication` を 1 に設定します。これにより、`wal_level` が `logical` に自動的に設定されます。
   - 適切な WAL LSN のライフタイムを得るには、`max_wal_size` を少なくとも 8 GB に設定します。

> **注釈**
>
> Aurora PostgreSQL では、*max_wal_size* を変更することはできません。AWS によって管理されています。このパラメーターは、RDS PostgreSQL インスタンスでのみ構成可能です。 !!!

`max_wal_size` の設定が小さすぎると、DMS エージェントがストリーミングできる前に Postgres が WAL エントリをドロップし、移行が妨げられるか、速度が低下する可能性があります。

1. Reboot your RDS instance 変更を適用します。
2. データベースに接続し、実行することにより、設定が有効になったことを確認します。

```
SHOW wal_level;
SHOW max_wal_size;
```

`wal_level` は `logical` を返す必要があります。

レプリケーショングループロールの作成

Postgres パブリケーションを自動作成するには、移行ユーザーがソーステーブルを所有している必要があります。これらのテーブルは既に別のロールが所有しているため、現在のテーブル所有者と移行ユーザーの両方が属するレプリケーショングループロールを作成します。

```
CREATE ROLE <replication_group>;
GRANT <replication_group> TO <migration_user>;
GRANT <replication_group> TO <original_owner>;
ALTER TABLE <table_name> OWNER TO <replication_group>;
```

そこで

- `<migration_user>` は、移行のために作成した Postgres ユーザーです。
- `<original_owner>` は、テーブルの元の所有者です。
- `<replication_group>` は、パブリケーション自動作成のソーステーブルを所有する共有ロールです。
- `<table_name>` は、移行するテーブルの名前です大文字と小文字が区別されます。

すべてのスキーマにわたって、移行する予定のテーブルごとに `ALTER TABLE ... OWNER TO <replication_group>` を実行する必要があります。データベース内のすべてのテーブルの所有権を一度に譲渡するには

```
DO $$
DECLARE
```

(次のページに続く)

(前のページからの続き)

```
  s text;
  t text;
BEGIN
  FOR s, t IN
    SELECT <schemaname>, <tablename> FROM pg_tables
    WHERE <schemaname> NOT IN (pg_catalog, information_schema)
  LOOP
    EXECUTE ALTER TABLE  || quote_ident(s) || . || quote_ident(t) ||  OWNER TO
↪<replication_group>;
  END LOOP;
END;
$$;
```

構成の検証

DMS エージェントには、ソースデータベースが CDC 移行のすべての要件を満たしているかどうかを確認する検証スクリプトが含まれています。

1. DMS エージェントフォルダーに移動します。

```
cd /opt/cdcagent/reader/
```

1. 移行するすべてのテーブルの配列を<schema_name>.<table_name> 形式で作成します。

```
arr=(schema1.table1 schema1.table2 schema2.table1)
```

1. 検証スクリプトを実行します。

```
PG_USERNAME=postgres PG_PASSWORD=<password> PG_HOST=<rds-endpoint> PG_PORT=5432 DB_
↪NAME=<db_name> DBZ_USERNAME=<migration_user> DBZ_PASSWORD=<migration_user_password>␣
↪./postgresConfigValidation.sh "${arr[@]}"
```

合格結果は次のようになります。

```
  ***  [Transporter] - Get PostgreSQL Major Version
PostgreSQL major version: 17
  ***  [Transporter] - Validate WAL Level
wal_level:  logical
[Pass] wal_level is logical.
  ***  [Transporter] - Validate max WAL senders
max_wal_senders: 10
[Pass] max_wal_senders is at least 1.
  ***  [Transporter] - Validate max replication slots
max_replication_slots: 10
[Pass] max_replication_slots is at least 1.
  ***  [Transporter] - Validate max WAL size
```

(次のページに続く)

(前のページからの続き)

```
max_wal_size: 8 GB
[Pass] max_wal_size is set to the recommended value of 8GB or higher.
  ***  [Transporter] - Validate checkpoints
checkpoints_timed: 100
checkpoints_req: 5
[Pass] Timed checkpoints are more frequent than requested checkpoints.
  ***  [Transporter] - Check <migration_user> user role
[Pass] User <migration_user> is present
[Pass] User <migration_user> has replication permission
  ***  [Transporter] - Check SELECT privilege on the tables to be migrated
[Pass] User <migration_user> has select privilege on all tables to be migrated.
  ***  [Transporter] - Check CREATE privilege on database
[Pass] Role <migration_user> has CREATE privilege on database <db_name>.
```

注釈

続行する前に`[Failed]` または`[Suggestion]` ステータスに対処します。 *[Failed]* チェックは、DMS エージェントがデータ移行を実行する前に解決が必要な問題をブロックしています。 *[Suggestion]* チェックでは移行が許可されますが、対処しないでおくとパフォーマンスに悪影響を与える可能性があります。

次のステップ

Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database

Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database

EDB Postgres AI エージェント `beacon-agent` としてパッケージ化して、HM との安全な接続を確立し、Postgres ソースからスキーマと移行評価データを取得します。

前提条件

- **マシンユーザーとアクセスキー** *Creating a machine user* で作成されます。アクセスキーを環境変数に保存します。

```
export BEACON_AGENT_ACCESS_KEY="baak_your_key_here"
```

- **pg_dump** EDB Postgres AI エージェントは pg_dump を使用して Postgres ソースから DDL を抽出します。 pg_dump が補助マシンにインストールおよび構成されていることを確認します。インストール手順、互換性ルール、および資格情報のセットアップについては、*Configuring your auxiliary machine or bastion host* を参照してください。

注釈

エージェントは、実行時に`beacon_agent.yaml` で参照される環境変数を置き換えるため、エージェントを実行する前に環境で`$BEACON_AGENT_ACCESS_KEY` を設定するだけで十分です。環境変数は、

YAML の静的値よりも常に優先されます。

環境変数を設定するための推奨事項

$BEACON_AGENT_ACCESS_KEY や$DSN などの機密データは、シークレットマネージャーで管理される環境変数として扱うことをお勧めします。この戦略により、機密の資格情報がプレーンテキストの構成ファイルに保存されるのを防ぎます。環境変数を使用する場合、EDB Postgres AI エージェントの実行中にそれらが利用できることを確認してください。

実稼働環境のための最も安全なアプローチについて、組織の IT 部門に相談してください。

短期間の移行またはテストとデモの場合、アクティブなターミナルセッションで export コマンドを使用できます。

HM 接続パラメーターの取得

エージェントには、HM インストールからの特定の値が必要です。次のことについては、管理者に連絡してください。

- **ビーコンサーバーホスト名** beacon_server の内部エンドポイント。

HM インストールを構成するときに、管理者は beacon_server サービスの内部エンドポイントを指定する必要があります。HM インスタンスの管理者またはインストーラーに、インストールに使用される Helm チャート構成ファイルで beacon_server サービスに設定したホスト名 URL を提供するように依頼します。values.yaml ファイルの parameters.upm-beacon.server_host または BEACON_SERVICE_DOMAIN_NAME に設定した値を検索します。この値は、後で EDB Postgres AI エージェントを構成するために必要になります。

- **ルート証明書パス必要な場合** HM インスタンスが自己署名または内部 CA を使用する場合、ca_public.crt ファイルを取得します。エージェントマシンたとえば/etc/beacon/ca_public.crt に保存します。

EDB Postgres AI エージェントには、HM インスタンスの beacon_server サービスへの信頼できる接続が必要です。 EDB Postgres AI エージェントを実行しているサーバーが HM の TLS 証明書組織のセキュリティインフラストラクチャによって管理されているを本質的に信頼していない場合、HM インスタンスの管理者またはインストーラーに証明書を含むファイルを提供するように依頼してください。後でこの証明書へのディレクトリパスを提供して、EDB Postgres AI エージェントを構成する必要があります。

beacon_server およびルート証明書の詳細については、*Administrative tasks* を参照してください。

接続の検証

HM 接続パラメーターを取得したら、補助マシンが Postgres ソースと HM の両方に到達できることを確認します。次のポートを開く必要があります。

Destination	Port	Used by
Postgres host	5432 (default for PostgreSQL, AWS RDS, and Aurora); 5444 (default for EDB Postgres Advanced Server)	Both agents (database connection)
HM *beacon_server*	9443	EDB Postgres AI agent
HM DMS service (*RW_SERVICE_HOST*)	443 (HTTPS)	DMS agent

注釈

ポート 5432 および 5444 はデフォルトです。 Postgres データベースが別のポートでリッスンするように構成されている場合、代わりにそのポートを使用し、ネットワークでオープンしていることを確認します。

nc を使用して、各ホストとポートへの接続を確認します。

```
#  Postgres source

nc -zv <postgres-host-name-or-ip> 5432

#  HM beacon server

nc -zv <beacon_server_hostname> 9443

#  HM DMS service

nc -zv <rw-service-hostname> 443
```

接続が成功すると `succeeded!` が返されます。 `Connection refused` または `Operation timed out` が表示される場合、ポートは到達不能であるため、続行する前に開く必要があります。

ソースデータベース接続パラメーターの定義

データソース名 DSN は、Postgres ソースの接続文字列です。

フォーマット

```
DSN="postgresql://<migration_user>:<migration_user_password>@<host>:<port>/<database_
↪name>"
```

エージェントを実行またはサービスを構成する前に、DSN を環境変数として設定します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb"
```

データベースの SSL が無効になっている場合、 `?sslmode=disable` を追加します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb?
↪sslmode=disable"
```

RDS インスタンスエンドポイントをホストとして使用します。RDS はデフォルトで SSL を適用するため、?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@myinstance.abc123.us-east-1.rds.
↪amazonaws.com:5432/mydb?sslmode=require"
```

Aurora クラスターエンドポイントをホストとして使用します。Aurora はデフォルトで SSL を適用するため、 ?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@my-cluster.cluster-abc123.us-east-1.
↪rds.amazonaws.com:5432/mydb?sslmode=require"
```

```
DSN="postgresql://migration_user:pa%24%24word@postgres-host:5432/mydb"
```

構成ファイルの準備

1. ホームディレクトリに EDB Postgres AI エージェント構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

> **注釈**
>
> エージェントは`/etc/beacon/` または`${HOME}/.beacon/` で`beacon_agent.yaml` を検索します。最終的なフォールバックとして、実行されたディレクトリを検索します。 !!!

1. このディレクトリ内に、ファイル beacon_agent.yaml を作成します。

```
touch ${HOME}/.beacon/beacon_agent.yaml
```

1. 次のテンプレートをコピーして、新しいファイルに貼り付けます。

このテンプレートでは、 $BEACON_AGENT_ACCESS_KEY および$DSN は、前に構成した環境変数です。テンプレート内のすべての< >プレースホルダーを置き換えます。詳細は、*EDB DMS agent parameter reference* を参照してください。

```
---
agent:
  access_key: $BEACON_AGENT_ACCESS_KEY
  beacon_server: <beacon_server_hostname>:9443
  project_id: <your_project_id>
  providers:
    - "onprem"
  # In most HM deployments, set `root_ca_path` to the path of the ca_public.crt
```

(次のページに続く)

(前のページからの続き)

```
  # file obtained from your HM administrator (for example, /etc/beacon/ca_public.crt).
  # Leave empty only if HMs certificate is already trusted by the systems CA bundle.
  root_ca_path: ""
  schema_providers:
    # Include "onprem-schema" to enable schema ingestion and migration assessment.
    # Remove this section if you dont want schema information to be collected.
    - "onprem-schema"
provider:
  onprem:
    databases:
      # `resource_id` controls how this database appears in HM Estate.
      - resource_id: "<database_resource_id>"
        dsn: $DSN
        # pg_bin_path: /usr/lib/postgresql/16/bin
        # Optional. Absolute path to the directory containing pg_dump and pg_dumpall.
        # Set when registering databases from different Postgres distributions/
↪versions.
        schema:
          enabled: true
          poll_interval: 15s
general:
  logging:
    # `info` is simplified, use debug during initial setup or
    # troubleshooting for more verbose output.
    level: info
```

1. ファイルを保存します。

EDB Postgres AI エージェントの実行

1. エージェントを起動する前に、接続と構成構文を確認します。

```
beacon-agent validate
```

1. エージェントを実行します。

```
beacon-agent
```

> **Tip**
>
> 他のエージェントモードとオプションについては、 Agent CLI を参照してください。 !!!

1. エージェントが HM へのデータの送信を開始するときに、取り込みプロセスを監視します。数分後、Postgres データベースが HM コンソールの **Estate > Migrations** の下に表示されます。

重要

エージェントは、テーブルと列の変更を動的に検出します。ただし、ビュー、プロシージャー、またはファンクションを変更するには、エージェントを再起動してメタデータを再取り込む必要があります。

本番使用の場合は、 *Running the agent as a service* を参照してください。

移行評価の検証

Postgres データベースがエージェントに認識され、移行の準備ができていることを確認するには、HM コンソールで **Estate > Migrations** に移動します。

データベースがテーブルに表示される場合、次のリソースにアクセスできます。

- **Estate** ページの **マイグレーション** の **移行評価**。データベーススキーマ数、コア数、移行の複雑さ、および作業レベルの概要を提供します。
- **Migrations** タブから特定のデータベースを選択することによりアクセスできる *Viewing migration assessment details* 。このタブは、**スキーマ** の拡張分析、グローバル **評価**、および移行を実行するための他の HM ツールに直接接続する **移行**ハブを提供します。
- ビルトインの移行ポータルでのスキーマの移行に使用できる**自動スキーマ DDL 抽出**。取り込みが完了すると、移行ポータルはデータベースごとにプロジェクトを自動的に作成します。これを使用して、スキーマを準備し、宛先データベースに移行できます。

障害対応

エージェントが HM または Postgres データベースへの接続に失敗する理由はいくつかあります。ここでは、トラブルシューティングに使用できるツールをいくつか紹介します。

`beacon-agent validate` コマンド

`beacon-agent validate` コマンドは、一般的な接続と構成の問題を特定するための役立つ手順です。

```
beacon-agent validate
```

validate コマンドは、次のことをチェックします。

- **サーバー接続** エージェントが構成済みの `beacon_server` 通常はポート `9443` に接続および認証できることを確認します。
- **データベース接続**構成されている各データベース DSN への接続を試行します。これは完全な接続テストであり、単なる構文または URL エンコーディングのチェックではありません。
- **構成表示** TLS 設定、プロジェクト ID、プロバイダー、およびデータベースごとの設定を表示し、構成の誤りを特定するのに役立ちます。

beacon-agent ログ

エージェントが取り込み中にエラーを報告した場合、詳細についてはログを確認できます。フォアグラウンドでエージェントを実行している場合、ログはターミナルに出力されます。エージェントを systemd サービスとして設定する場合、`journalctl` を使用してログを表示します。

次に、ログに表示される一般的なエラーメッセージをいくつか示します。

Error	Cause	Fix
tls: failed to verify certificate: x509: certificate signed by unknown authority	The agent can't verify HM's TLS certificate because the CA isn't trusted by the system.	Set *root_ca_path* in *beacon_agent.yaml* to the path of the *ca_public.crt* file obtained from your HM administrator. If you can't obtain the certificate, you can set *plaintext: true* in the agent config as a temporary workaround (not recommended for production).
rpc error: code = Unknown desc = no rows in result set	The beacon-agent version is incompatible with the server.	Upgrade to beacon-agent v1.384.0 or later.

エージェントをサービスとして実行する

継続的なメトリックストリーミングのために、また Postgres データベースが **HM コンソール>エステート**に表示されたままであることを確認するには、systemd サービスとして実行するようにエージェントを構成します。サービスとして実行すると、システムの起動時に自動的に起動し、エラーが発生した後に再起動します。

サービスがないと、エージェントは手動で実行している間のみメトリックをストリーミングします。データベースは、エージェントがストリーミングを停止してから 72 時間後に HM 不動産から消えます。

> **注釈**
>
> *beacon_agent.yaml* たとえば *$BEACON_AGENT_ACCESS_KEY* や `$DSN` などで参照される環境変数が、実行時にサービスにアクセスできることを確認します。スキーマ抽出資格情報の `.pgpass` ファイルを構成した場合 **環境の構成** で説明しているように、そのファイルが `User=` で指定されたユーザーのホームディレクトリに存在することを確認します。

1. サービスファイルを作成します。この例では `vi` を使用しますが、任意のテキストエディタを使用できます。

```
sudo vi /usr/lib/systemd/system/beacon-agent.service
```

オペレーティングシステム別のデフォルトのサービスファイルパス

Operating system	Default path
RHEL	*/usr/lib/systemd/system/beacon-agent.service*
Rocky Linux	*/usr/lib/systemd/system/beacon-agent.service*
AlmaLinux	*/usr/lib/systemd/system/beacon-agent.service*
Ubuntu	*/etc/systemd/system/beacon-agent.service*
Debian	*/lib/systemd/system/beacon-agent.service*

1. 次のテンプレートをファイルに貼り付けて、すべての< >プレースホルダーを置き換えます。

```
[Unit]
Description=EDB Postgres AI Agent
After=network.target

[Service]
Type=simple
User=<agent_os_user>
WorkingDirectory=</home/agent_os_user/>
ExecStart=</usr/local/bin/beacon-agent>
Restart=on-failure
RestartSec=60

[Install]
WantedBy=multi-user.target
```

Setting	Placeholder	Description
User=	*<agent_os_user>*	エージェントを実行する OS ユーザー。ソースデータベースの準備時に作成した専用の OS ユーザーを使用します。
WorkingDirectory=	*</home/agent_os_user/>*	`.beacon`フォルダーが保存されるディレクトリ。これは通常、`<agent_os_user>`のホームディレクトリです。
ExecStart=	*</usr/local/bin/beacon-agent>*	`beacon-agent`バイナリへのフルパス。パスは、エージェントがインストールされた方法と場所によって異なります。

1. systemd をリロードし、サービスを有効にして開始します。

```
sudo systemctl daemon-reload
sudo systemctl enable beacon-agent.service
sudo systemctl start beacon-agent.service
```

1. ログを確認して、サービスが正常に開始されたことを確認します。

```
journalctl -u beacon-agent.service
```

関連トピック

- スキーマ取り込みのより詳細な制御たとえば、スキーマを除外または含める場合、*Controlling schema ingestion* を参照してください。
- 構成パラメーターの完全なセットについては、*EDB DMS agent parameter reference* を参照してください。
- 管理者がルート証明書または `beacon_server` 値の取得に関するガイダンスが必要な場合は、*Administrative tasks* を参照してください。

次のステップ

Preparing your HM-managed destination database

Preparing your HM-managed destination database

移行の宛先として機能する HM 管理データベースクラスターを準備します。

ハイブリッドマネージャー HM に宛先データベースクラスターがまだない場合は、作成します。

1. HM コンソールで、データベースクラスターを作成する "プロジェクトを選択できません" — ポータルドメインはデフォルトで API エンドポイントになります を選択します。
2. プロジェクトで、 **Create New** を選択し、 **Database Cluster** から *Create a database cluster template* を選択します。
3. **Clusters** ページで、クラスターを選択し、 **Quick Connect** オプションを使用してターミナルからデータベースにアクセスします。
4. 移行の宛先として使用する空のデータベースを作成します。または、デフォルトのデータベース `edb_admin` を使用できます。

すでに HM 管理のデータベースクラスターがある場合は、それを移行の宛先として使用できます。

続行する前に、移行するテーブルと移行中に生成される負荷がクラスター上の既存のワークロードを干渉しないことを確認してください。

次のステップ

Migrating Postgres schemas

Migrating Postgres schemas

データの移行を開始する前に、スキーマの移行を処理する方法を選択します。HM は 2 つのアプローチをサポートしています。

- **簡素化推奨** HM は、データ移行セットアップの一部としてスキーマの移行と制約処理を自動的に管理します。
- **手動** Migration Toolkit または `pg_dump` を使用して自分自身でスキーマをエクスポートおよびインポートし、各ソーステーブルのキー構造に基づいて制約を処理します。

手動でのスキーマの準備は必要ありません。 HM は、後で **スキーマとデータ**スコープを選択すると、スキーマの移行と制約の順序付けを自動的に処理します。続行する前に、次を確認します。

- EDB Postgres AI エージェントはソーススキーマを取り込み、データベースには移行ポータルプロジェクトが存在します。確認するには、HM コンソールで **Estate > Migrations** に移動し、**Database Name** 列でデータベースを選択し、 **MP Project** 列にプロジェクトが表示されることを確認します。
- 後の手順で DMS リーダーエージェントを構成するときに、`DBCONFIG_DATABASES_<N>__RESOURCEID` を `beacon_agent.yaml` と同じ `resource_id` 値に設定します。 **スキーマとデータ** および**スキーマのみ** スコープは、これがないと失敗します。

次のステップに進みます。

データの移行に EDB DMS を使用する前に、スキーマを宛先データベースにエクスポートおよびインポートします。DMS はデータのみを移行します。移行を開始する前に、宛先データベースにスキーマを設定する必要があります。

コンストレインの処理

制約を処理する方法は、制約タイプと各ソーステーブルのキー構造によって異なります。

Constraint type	Action	Notes
Primary key	今すぐ除外する — DMS は自動的に適用されます	The DMS writer applies the primary key when the migration enters the streaming phase, using it to deduplicate snapshot data.
Unique key (source table has a primary key)	今すぐ除外、ストリーミング開始後に適用	Primary key-based deduplication ensures no duplicate rows remain when you apply unique constraints after streaming starts.
Unique key (source table has no primary key)	現在サポートされていません	Use the simplified path instead — select the **Schema and data** scope when creating the migration to let HM handle schema migration automatically.
NOT NULL	今すぐ含める	*NOT NULL* validation adds no meaningful overhead to the data load, unlike foreign key and check constraint validation.
Foreign key, check, and exclusion	今すぐ除外、移行後に再適用	These cause unnecessary CPU and memory overhead while data is actively being migrated.

注釈

主キーまたは一意キーのないテーブルの場合、移行の前後にキー制約は必要ありません。 DMS は、これらのテーブルをキーレスとして扱い、少なくとも 1 回の配信を提供します。移行後の検証の一部として、行の重複を削除します。

スキーマのエクスポート

EDB Migration Toolkit を使用してスキーマをエクスポートすることをお勧めします。MTK の offline migration 機能は、直接接続せずに宛先に適用できる SQL スクリプトを生成します。他の有効なエクスポート方法には、`pg_dump` /`pg_restore` 、pgAdmin、または `psql` などがあります。

Migration Toolkit を使用する場合

- `-offlineMigration` オプションを使用して SQL スクリプトを生成します。
- `-schemaOnly <schema_scope>` オプションを使用して、スキーマのみをエクスポートします。
- インポートする前に、生成されたスクリプトから主キー、一意、外部キー、チェック、および除外コンストレインを除外コメントアウトします。
- オブジェクトを宛先の別の名前にマップする予定がある場合は、インポートする前に DDL スクリプトで直接編集します。名前が変更されたオブジェクトを参照するすべての依存オブジェクトビュー、コンストレイン、およびその他の参照を更新します。詳細は、*Mapping schema, tables, and columns* を参照してください。

スキーマのインポート

DDL を準備し、遅延制約を除外した後、宛先データベースに接続し、生成された offline migration script を実行して SQL ファイルを適用します。

> **注釈**
>
> 移行が完了した後、除外された一意の外部キー、チェック、および除外制約を適用します。 DMS ライタは、主キーコンストレインを自動的に適用します。手動で再適用しないでください。除外されたコンストレインを追跡して、必要なときに再適用できます。

次のステップ

Configuring and running the EDB DMS agent

Configuring and running the EDB DMS agent

EDB DMS エージェントは、AWS RDS または Aurora PostgreSQL ソースに接続し、Data Migration ServiceDMS と連携してデータ移行を実行します。実行する前に、HM で移行資格情報を作成し、ソースデータベースの詳細でエージェントを構成します。

前提条件

- DMS エージェントを実行しているマシンに Java / OpenJDK 17 以降がインストールされていることを確認します。
- *Install the EDB DMS agent* 。

移行資格情報の作成

移行資格情報は、DMS エージェントをハイブリッドマネージャー HM に対して認証します。HM プロジェクトごとに 1 セットの資格情報が必要です。

1. HM コンソールで、移行に使用しているプロジェクトを選択します。
2. **移行>資格情報** を選択します。
3. **Create Migration Credential** を選択し、名前を入力して、 **Create Migration Credential** を選択します。 `credentials.zip` ファイルがマシンにダウンロードされます。

> **注釈**
>
> 資格情報は安全に保存します。それらを失った場合は、このセットを取り消し、新しいセットを作成する必要があります。 !!!

1. ダウンロードしたファイルを解凍し、資格情報フォルダーを DMS エージェントをインストールしたマシンたとえば`$HOME/credentials/` にコピーします。別のパスを使用する場合は、スクリプト内の `CREDENTIAL_DIRECTORY_PATH` を一致するように更新します。

フォルダーには次のものが含まれています

DMS エージェントの構成

1. `/opt/cdcagent/run-cdcagent.sh` を書き込み権限で開きます。
2. 必要な変数を設定します。オプションのパラメーターはデフォルトでコメントアウトされています。環境に適用されるものはコメントを外して構成します。

詳細は、 *EDB DMS agent parameter reference* を参照してください。

```
#!/bin/bash -e
# run-cdcagent.sh
#
# This script provides a convenient place to specify
# environment variables used to configure the
# EDB Data Migration Service Agent.
#

##########################################
# DMS Agent General Configuration       #
##########################################

# This ID is used to identify DMS Agent
# and is specified by the user.
export DBCONFIG_ID=<source_name_of_choice>

# Determine the run mode of the DMS Agent
# The run mode can be set to either reader or writer
```

(次のページに続く)

(前のページからの続き)

```
# reader mode is used to read changes from a source database
# writer mode is used to write changes to a target database
export RUN_MODE=reader

# This is the DMS backend service used by the Agent
# Obtain this value from the HM console Sources or Destinations migration pages
export RW_SERVICE_HOST=<rw_service_host>

##########################################
# DMS Agent TLS Configuration           #
##########################################

# You need to create migration credentials in EDB Postgres AI platform
# and set the path of the credentials directory
export CREDENTIAL_DIRECTORY_PATH=$HOME/credentials

##########################################
# DMS Agent Source DB Configuration     #
##########################################

# A sample configuration to create a single Postgres database connection:
export DBCONFIG_DATABASES_0__TYPE=POSTGRES
export DBCONFIG_DATABASES_0__HOSTNAME=<RDS_endpoint>
export DBCONFIG_DATABASES_0__PORT=<5432>
# The CATALOG is the database name
export DBCONFIG_DATABASES_0__CATALOG=<source>
export DBCONFIG_DATABASES_0__USERNAME=<postgres>
# The password env can be set without specifying it here
# but the env structure looks like this
export DBCONFIG_DATABASES_0__PASSWORD=<password>
# Required for Schema and data or Schema only scope.
# Must match resource_id in beacon_agent.yaml.
#export DBCONFIG_DATABASES_0__RESOURCEID=<resource_id>

# You can increase the index to configure more than
# one database for the DMS Agent
#export DBCONFIG_DATABASES_1__TYPE=POSTGRES
#export DBCONFIG_DATABASES_1__HOSTNAME=<RDS_endpoint>
#export DBCONFIG_DATABASES_1__PORT=<5432>
#export DBCONFIG_DATABASES_1__CATALOG=<source>
#export DBCONFIG_DATABASES_1__USERNAME=<postgres>
#export DBCONFIG_DATABASES_1__PASSWORD=<password>
#export DBCONFIG_DATABASES_1__RESOURCEID=<resource_id>

##########################################
```

(次のページに続く)

(前のページからの続き)

```
# Optional Parameters Below              #
#########################################

# FIPS mode can be enabled by setting the FIPS_MODE env variable to true
#export FIPS_MODE=true

# Configure logging
# Loglevel for the entire Quarkus application and underlying components
# WARNING: Enabling this loglevel is extremely verbose
#export QUARKUS_LOG_LEVEL=DEBUG
# Loglevel for the EDB package
# Preferred mode for debugging issues related to the logic of the agent
#export QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL=DEBUG

cd "$(dirname "$0")"

echo "Starting in mode: $RUN_MODE"

case "$RUN_MODE" in
  reader)
    java -jar ./reader/quarkus-run.jar
    ;;
  writer)
    java -jar ./writer/quarkus-run.jar
    ;;
  *)
    echo "Unknown RUN_MODE: $RUN_MODE"
    exit 1
    ;;
esac
```

注釈

DBCONFIG_DATABASES_0__HOSTNAME を、AWS コンソールの**接続とセキュリティ**の下にある RDS エンドポイントに設定します。編集された例は `foo_database.cb7asw1xxxxx.eu-west-0.rds.amazonaws.com` です。 !!!

DMS エージェントの実行

DMS エージェントディレクトリに移動し、構成スクリプトを実行します。

```
cd /opt/cdcagent
./run-cdcagent.sh
```

スクリプトは、必要なすべての環境変数をエクスポートし、エージェントプロセスを開始します。エージェ

ントは実行されると、AWS RDS または Aurora PostgreSQL データベースを HM のソースとして登録します。次のセクションに進んで、正しく表示されることを確認します。

HM でのソースの検証

エージェントが実行されたら、データベースが HM にソースとして表示されることを確認します。

1. HM コンソールで、プロジェクトを選択します。
2. **移行>ソース** を選択します。

データベースは、名前、タイプ、エージェント ID、およびステータスとともにリストされます。このページのすべてのフィールドの説明については、デプロイメントリソースを理解する を参照してください。

次のステップ

Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database

Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database

EDB DMS エージェントが実行され、スキーマがインポートされるか、HM インポートの準備が整うと、HM コンソールで移行を作成する準備が整います。このページでは、完全な移行ライフサイクルについて説明します。移行の作成、その進行状況の監視、ファイナライズ、および移行後の制約の適用。

以下の手順は、前のページで選択したアプローチによって異なります。選択したものと一致するタブを選択します。

移行の作成

移行は、Postgres ソースと HM によって管理される宛先データベースクラスター間の同期を確立します。

1. HM コンソールで、移行を管理するプロジェクトを選択します。
2. プロジェクトで、 **移行>移行** を選択します。
3. [**移行**] ページで、[**Create New Migration**] を選択します。
4. **Create OLTP Migration** ページで、名前を移行に割り当てます。
5. 移行のソースを選択します。 DMS エージェントの ID は、 **Source** メニューにリストされています。
6. **Scope** の下で、アプローチに一致するオプションを選択します。

スキーマとデータ を選択します。このオプションを選択すると、 HM:

1. データベースを EDB Postgres AI エージェントに登録したときに自動的に作成された移行ポータルプロジェクトからスキーマをソースします。
2. 制約なしで宛先にテーブルを作成するため、初期スナップショットコピーは作成されません制約の検証により遅くなります。
3. スナップショットが完了したら、主キーと一意の制約があるテーブルの場合を適用して、DMS がストリーミング変更を 1 回だけ配信できます。PK または UK のないテーブルは少なくとも 1 回の配信にフォールバックし、検証中に重複排除が必要です。

4. カットオーバーまで残りのスキーマインデックス、外部キー、チェック制約、NOT NULL などを遅延します。オプションで、この作業のほとんどを**ファイナライズスキーマコンカレントリー**を介してストリーミングと同時に適用して、カットオーバーウィンドウを縮小できます。 ストリーミング移行の仕上げ を参照してください。

5. 移行を完了とマークした後、遅延外部キーを検証し、シーケンスを同期し、残りの軽量ファイルを適用します最終の**スキーマファイナライズ** ステージのタスク。

> **注釈**
>
> テーブルデータを移行せずにデータベース構造を移行する場合は、 **スキーマのみ**を使用します。!!!

データのみを選択します。スキーマは、前の手順で既に手動でインポートされています。

1. スナップショットタイプを選択します *Snapshot* または *Snapshot + Streaming* 。

2. **Destination Type** で、移行用の HM 管理の宛先データベースクラスターとデータベースを選択します。次に、**Next** を選択します。

3. **Review your Tables** タブで、移行に含めるテーブルと列を選択します。テーブルを展開して、その列とオプションで *map source columns to different destination column names* を表示します。列マッピングは、 **データのみ**スコープでのみ使用できます。

4. **Create Migration** を選択します。

HM コンソールには、 *new migration* が表示されます。**スナップショット+ストリーミング** を選択した場合、移行が **実行中**状態を示している間、DMS エージェントは継続的にデータ変更をストリーミングします。

> **注釈**
>
> DMS エージェントは、データ変更のみをストリーミングします。DDL 変更 DDL 変更はストリーミングしません。

進行状況のモニタリング

移行リストビュー

Migrations ページには、プロジェクト内のすべての移行の概要が表示されます。**移行の進行状況** 列には、移行ごとに 3 つのメトリックが表示されます。

- **テーブルの合計数** — この移行に含まれるテーブルの合計
- **スナップショットテーブルが完了しました** —スナップショットステージが終了したテーブル
- **移行された行** — すべてのテーブルで移行された累積行数

移行の詳細ビュー

リストから移行を選択して、3 つのタブがある詳細ページを開きます。

- **概要** - 現在の移行状態と、ソースおよび宛先データベースに関する一般的な情報を表示します。**スナップショット+ストリーミング** の移行の場合、ステータスが**ストリーミング** のときに移行はキャッチアップされます。**スナップショット** の移行の場合、ステータスが **Completed** になると完了します。
- **Tables** —移行のために選択されたすべてのテーブルを表示します。テーブルごとに、ソースおよび宛先テーブル、スキーマ名、スナップショットステータス、マップされた列、移行された行、および移行速度を確認できます。**+** アイコンを選択してテーブルを展開し、列レベルのマッピングを表示します。
- **Error History** —この移行についてログに記録されたすべてのエラーをリストします。 **i** アイコンを選択して、特定のエラーの完全なエラーログを開きます。移行が停止または失敗した場合、ここでのアクティブなエラーは、トラブルシューティングを開始する場所を示します。

DMS エージェントログ

HM コンソールは、すべての DMS エージェントエラーを表示するわけではありません。移行がスタックしているか、進行中のテーブルがない場合は、`run-cdcagent.sh` を実行したターミナルで `ERROR` ログエントリを確認します。多くの場合、これらのエントリには、根本的な原因が含まれています。一般的な問題には、DMS エージェントと HM 間のバージョンの非互換性や接続または資格情報の障害が含まれます。

ストリーミング移行の仕上げ

> **注釈**
>
> このセクションは、**スナップショット+ストリーミング**の移行にのみ適用されます。 **スナップショット**の移行の場合、DMS は移行を自動的に完了としてマークします。

カットオーバー前のキャッチアップのモニタリング

ソースへの書き込みを停止する前に、宛先に追いつくまでの時間を与えます。**移行の詳細**ページを使用して、行数とレプリケーションラグを追跡します。デスティネーションが一貫してソースに近づいた後にのみ、カットオーバーを開始します。

スキーマのファイナライズを同時に行うことをお勧めします

Schema および data スコープを使用した場合、HM はストリームの実行中にほとんどのスキーマオブジェクトを適用できます。プライマリキー、一意制約、インデックス `CREATE INDEX CONCURRENTLY` でビルドされるため、ライブ書き込みがブロックされません、外部キー (`NOT VALID`)、チェックコンストレイン、`NOT NULL` コンストレインなどとして追加されます。外部キー検証、シーケンス同期、およびその他の軽量タスクは、カットオーバー後の**スキーマファイナライズ** ステージで実行されます。

この作業を実行すると、読み取り専用のカットオーバーウィンドウが同時に縮小されます。テーブルが大きくなり、インデックスの数が増えるほど、節約される時間は長くなります。

すべてのテーブルがスナップショットを完了し、移行がストリーミングフェーズに入ると、**移行の詳細** ページに、 **ファイナライズスキーマコンカレントリー**ボタンを備えた **スキーマファイナライズ可能**アラートが

表示されます。ボタンはストリーミング中は利用可能なままであるため、目的地が追いつくまでにどれだけ近いかに基づいて、瞬間を選択できます。

1. **Migration Details** ページで **Finalize Schema Concurrently** を選択します。
2. HM は、ストリーミングが継続されている間、遅延オブジェクトを適用します。移行は**実行中** 状態のままで、進行状況サブステップには**ファイナライズスキーマコンカレントリー**というラベルが付いています。

サブステップが完了したら、カットオーバーに進みます。

> **注釈**
>
> **ファイナライズスキーマコンカレントリー**は、ストリーミングが有効になっている**スキーマおよびデータ**スコープで作成された移行でのみ使用できます。

この手順をスキップすると、移行を完了としてマークした後、同じオブジェクトが連続して適用されます。スキーマの最終化ステージを参照してください。この場合、インデックスは非並列で構築され、制約は排他的ロックの下で検証されるため、カットオーバーウィンドウは大きくなります。

手動パスには適用されません。移行前にスキーマを適用し、カットオーバー後に遅延コンストレインを手動で再適用します。遅延コンストレインの適用とデータの検証 を参照してください。

カットオーバーの実行

トラフィックをソースから宛先にスイッチする準備ができたら

1. ソースデータベースのすべての書き込みを停止します。
2. HM コンソールで、デスティネーションが最後のソース変更を適用したことを確認します。
3. **Migration Details** ページで、 **Mark as Completed** を選択します。

これにより、ストリーミングプロセスが停止します。

スキーマの最終化ステージ

スキーマおよびデータ スコープで作成された移行の場合、移行は **Mark as Completed** の後、自動的に **スキーマファイナライズ**ステージに入ります。このステージで実行されるものは、最初にコンカレントファイナライズを実行したかどうかによって異なります。

- **同時ファイナライズを実行した場合** HM は、軽量の残りの作業外部キー検証、シーケンス同期、およびその他の遅延メタデータタスクを実行します。このステージは短いです。
- **同時ファイナライズをスキップした場合** HM は、完全な遅延セットプライマリキー、一意制約、インデックス、外部キー、チェック制約、`NOT NULL` などを適用した後、上記の軽量タスクを適用します。このステージにはかなりの時間がかかり、ソースが読み取り専用であるときに実行されます。

移行は、 **スキーマのファイナライズ**ステージが完了した後にのみ完全に完了します。進行状況は、 **移行の詳細**ページに表示されます。

データのみスコープでは、自動スキーマファイナライズは実行されません。遅延コンストレインの適用とデータの検証 に進み、移行前に延期した外部キー、一意、チェック、および除外制約を再適用します。

遅延コンストレインの適用とデータの検証

遅延コンストレインの再適用

HM は、 **スキーマファイナライズ**ステージ中に、すべての遅延コンストレインと同期シーケンスを自動的に適用しました。手動操作は必要ありません。スキーマの最終化ステージ を参照してください。

移行の初めに、外部キー、一意、チェック、および除外制約を除いたスキーマを準備およびインポートしました。宛先データベースに接続し、これらのコンストレインを再適用します。

主キー制約は、移行がストリーミングフェーズに入ったときに、DMS ライタによって自動的に適用されました。`NOT NULL` コンストレインは移行前に適用され、現在アクションは必要ありません。

必要に応じて行の重複を削除する

主キーも一意制約もなかったテーブルの場合、EDB DMS エージェントは少なくとも 1 回の配信を提供します。これにより、重複行が発生する可能性があります。検証手順の一部として重複排除を実行します。

移行したデータの検証

ソースデータベースと宛先データベースを比較して、すべてのデータが正常に移行されたことを確認します。これには、LiveCompare を使用できます。

関連トピック

- LiveCompare —移行の完全性を確認するための行レベルのデータ比較ツール
- 用語 —スナップショット、ストリーミング、およびその他の移行概念の定義

Migrate your self-managed Postgres database to a different self-managed Postgres database

これらは、ハイブリッドマネージャー HM のデータ移行サービス DMS を使用して、自己管理の Postgres データベースから別の自己管理の Postgres データベースへのエンドツーエンドの移行に関係するタスクです。宛先は HM 管理されていないため、この移行には 2 つの DMS エージェントインスタンスが必要です。ソースから変更をストリーミングするリーダーと、宛先に適用するライタ。

1. ** *Configuring your auxiliary machine or bastion host* ** —両方の移行エージェントをホストするマシンを構成します。このマシンは、Postgres ソースと HM 間のブリッジとして機能します。
2. ** *Preparing your AWS RDS/Aurora PostgreSQL source database* ** — 必要なユーザーを作成し、 EDB Postgres AI エージェントを使用したスキーマの取り込みと DMS リーダーエージェントを使用したデータ移行用に Postgres データベースを構成します。
3. ** *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* ** — エージェントをソースデータベースと HM に接続して、スキーマの取り込みと移行評価を有効にします。
4. ** *Preparing your HM-managed destination database* ** —自己管理の Postgres データベースを作成し、SSL と CDC 移行ロールで構成して、移行されたデータを受信します。
5. ** *Migrating Postgres schemas* ** —HM がスキーマの移行と制約の順序を自動的に処理する簡素化パスか、自分でスキーマをエクスポート、準備、およびインポートする手動パスを選択します。

6. ** *Configuring and running the EDB DMS agent* ** —移行資格情報を作成し、ソースのリーダーエージェントと宛先のライタエージェントを構成し、両方が HM に表示されることを確認します。

7. ** *Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database* ** — HM コンソールでデータ移行を作成して実行し、その進行状況を監視し、ストリーミングをファイナライズし、移行後のコンストレインを適用します。

Configuring your auxiliary machine or bastion host

EDB Postgres AI エージェント `beacon-agent` とデータ移行サービス DMS エージェント `cdcagent` は、Postgres データベースサーバー自分自身または別の補助マシンまたは要塞ホストで実行できます。ほとんどの運用環境では、専用の補助マシンでエージェントを実行することが優先されます。これにより、データベースサーバーへの追加ソフトウェアのインストールが回避され、単一のエージェントインスタンスでマルチプルのサーバー上のデータベースに接続できます。

必要なソフトウェア

エージェントを実行する前に、補助マシンに次のソフトウェアをインストールしていることを確認します。

- **Java 17 以降** — DMS エージェント `cdcagent` に必要です。`java -version` で確認します。
- **EDB Postgres AI エージェント** `beacon-agent` データベース登録、スキーマ評価、およびスキーマの移行に必要です。*beacon-agent version compatibility matrix* を確認します。次に、 install a compatible version of the EDB Postgres AI agent 。
- **DMS エージェント** `cdcagent` - データの移行と変更データキャプチャ CDC を処理します。インストールする前に、*DMS agent version compatibility* を確認してください。次に、 *install a compatible version of the DMS agent* 。

環境の構成

スキーマの取り込みを有効にすると、`beacon-agent` は `pg_dump` を使用して Postgres ソースから DDL を抽出します。Postgres クライアントツールパッケージには、`pg_dump` ユーティリティが含まれています。マシンで既に利用可能な場合があります。最初に確認します

```
pg_dump --version
```

コマンドが見つからない場合は、ソースデータベースディストリビューションと一致する install the client tools 。

インストールしたら、`pg_dump` バイナリが 2 つの互換性ルールを満たしていることを確認します。

- **バージョンルール** — `pg_dump` メジャーバージョンは、ソースデータベースのメジャーバージョン以上である必要があります。たとえば、ソースが Postgres 16 を実行している場合、`pg_dump` 16 以降が必要です。エージェントは起動時にこれを検証し、バージョンに互換性がない場合はエラーを報告します。
- **ディストリビューションルール** — `pg_dump` バイナリは、ソースデータベースと同じ Postgres ディストリビューションから取得する必要があります。各ディストリビューションの `pg_dump` は、ディストリビューション固有のカタログテーブルにアクセスします。一致しないバイナリたとえば、EDB Postgres Advanced Server ソースに対するコミュニティ PostgreSQL `pg_dump` を使用すると、予想されるカタログオブジェクトが存在しないため失敗します。

ホストでの複数の Postgres バージョンの管理

同じマシンに複数の Postgres バージョンがある場合は、正しい pg_dump が PATH の最初に表示されることを確認します。

```
which pg_dump
pg_dump --version
```

Debian および Ubuntu システムでは、update-alternatives を使用してアクティブなバージョンを管理します。

```
sudo update-alternatives --config pg_dump
```

複数のディストリビューションまたはバージョンからのデータベースの登録

単一の beacon-agent インスタンスは、さまざまな Postgres ディストリビューションたとえば、コミュニティ PostgreSQL および EDB Postgres Advanced Server またはさまざまなメジャーバージョンからデータベースを同時に登録できます。バージョンと配布ルールはデータベースごとに適用されるため、各データベース接続は異なる pg_dump バイナリを必要とする場合があります。

必要なすべての pg_dump バイナリを補助マシンにインストールします。後で、 beacon_agent.yaml 構成ファイルの pg_bin_path 設定を使用して、各データベースエントリを正しいバイナリディレクトリをポイントできます。設定されると、その特定のデータベースのシステム PATH ルックアップをオーバーライドします。省略すると、エージェントは PATH で最初に見つかった pg_dump にフォールバックします。例

```
provider:
  onprem:
    databases:
      - resource_id: "pg16-community"
        dsn: $DSN_PG16
        pg_bin_path: /usr/lib/postgresql/16/bin
        schema:
          enabled: true
      - resource_id: "epas17"
        dsn: $DSN_EPAS17
        pg_bin_path: /usr/edb/as17/bin
        schema:
          enabled: true
```

資格情報と pgpass 動作を理解する

beacon-agent を実行する OS ユーザーは、pg_dump を実行するユーザーと同じです。プロセス引数でデータベース資格情報の公開を回避するため、エージェントは、制限的なアクセス許可 0600 でパスワードを一時 pgpass ファイルに書き込み、パスワードを含まない接続文字列を pg_dump に渡します。エージェントは OS 一時ディレクトリにファイルを作成し、各抽出が完了すると自動的に削除します。

> **注釈**
>
> エージェントプロセスが予期せず強制終了された場合たとえば *SIGKILL* 、エージェントは一時 pgpass ファイルをクリーンアップしない場合があります。エージェントを実行する OS ユーザーのみがファイルを読み取ることができます。 OS は、リブート時、またはエージェントがコンテナで実行されている場合はコンテナの再起動時にそれを削除します。孤立したファイルを確認するには、*/tmp* の `.pgpass-*` と一致するファイルを探します。

次のステップ

Preparing your AWS RDS/Aurora PostgreSQL source database

Preparing your Postgres source database

移行を開始する前に、Postgres ソースデータベースを構成し、必要なユーザーを作成します。何を構成する必要があるかは、目標によって異なります。

- EDB Postgres AI エージェントを使用した **スキーマ取り込みのみ**ハイブリッドマネージャー HM でマシンユーザーを作成し、補助マシンでエージェントの専用 OS ユーザーを設定し、スキーマ抽出権限を持つ移行ユーザーを作成します。WAL レベルのデータベースの変更は必要ありません。
- Data Migration ServiceDMS エージェントを使用した **データの移行* WAL 設定を構成し、必要なレプリケーション特権を付与します。

スキーマの取り込みとデータの移行の両方を実行する予定がある場合は、すべてのセクションを完了します。スキーマまたはデータの移行のみを行う予定がある場合、各セクションでユースケースに適用できるかどうかについて説明します。

移行ユーザーの作成

> **Tip**
>
> EDB Postgres AI エージェントを使用したスキーマの取り込みと DMS エージェントを使用したデータ移行の両方に必要です。

EDB Postgres AI エージェントと DMS エージェントの両方にサービスを提供する単一の移行ユーザーを作成します。1 つのユーザーを使用すると、エージェント全体の資格情報の管理が簡素化されます。

データベース管理アカウントを使用して、ソース Postgres データベースに接続し、移行ユーザーを作成します。

```
CREATE USER <migration_user> WITH REPLICATION LOGIN PASSWORD <migration_user_password>
↪;
```

`REPLICATION` は、DMS エージェントがレプリケーションスロットを作成し、WAL から読み取るために必要です。スキーマの取り込みのみを実行する場合は、省略できますが、ここで含めるということは、後でデータの移行を追加する場合にユーザーを変更する必要がないことを意味します。

スキーマ取り込み権限の付与

> **Tip**
>
> EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

1. EDB Postgres AI エージェントにソースデータベースへの接続アクセスを付与します。

```
GRANT CONNECT ON DATABASE <source_database> TO <migration_user>;
```

<source_database> は、評価するデータベースの名前です。エージェントを同じサーバー上の追加データベースに接続する予定がある場合は、各データベースに CONNECT を付与します。

1. 完全なスキーマ抽出に必要なロールを付与します。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_data TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

これら 3 つのロールを組み合わせると、所有権や特権情報を含む完全なスキーマの抽出が可能になります。pg_read_all_data のみが付与されている場合、エージェントは、所有権と特権情報を使用せずに、スキーマオブジェクト定義に制限された部分抽出を実行します。

pg_read_all_data は、Postgres 10〜13 では使用できません。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

1. Postgres 16 以前を使用している場合は、エージェントが Postgres ログファイルの場所を検出できるように次も付与します。

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_current_logfile() TO <migration_user>;
```

データ移行の権限の付与

> **Tip**
>
> DMS エージェントを使用したデータの移行に必要です。

1. 移行するテーブルを含む各スキーマで移行ユーザーの使用を付与します。

```
GRANT USAGE ON SCHEMA <db_schema> TO <migration_user>;
```

1. ソーステーブルに SELECT を付与します。スキーマ全体の場合

```
GRANT SELECT ON ALL TABLES IN SCHEMA <db_schema> TO <migration_user>;
```

個々のテーブルの場合

```
GRANT SELECT ON <table_name> TO <migration_user>;
```

1. DMS エージェントがパブリケーションを作成できるように、ソースデータベースに CREATE を付与します。

```
GRANT CREATE ON DATABASE <pg_db_name> TO <migration_user>;
```

スキーマ取り込みの準備 EDB Postgres AI エージェント

> **Tip**
>
> EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

エージェントの専用 OS ユーザーの作成

beacon-agent を実行するための補助マシンに専用の OS ユーザーを作成します。この戦略により、必要のないエージェント権限の付与が回避されます。これは Linux 固有の例であり、正確なコマンドはディストリビューションによって異なる場合があります。

```
sudo useradd <agent_os_user>
```

拡張機能の移行の準備 EDB Postgres AI エージェント

> **Tip**
>
> ソースデータベースに拡張機能がインストールされている場合、 EDB Postgres AI エージェントを使用したスキーマの移行に必要です。

スキーマの移行の一部として、 EDB Postgres AI エージェントは、ソースデータベースにインストールされているすべての拡張機能を自動的に抽出し、他のスキーマ DDL が実行される前に宛先に適用します。拡張機能のインストールに失敗した場合、スキーマ全体の移行は中止されます。

スキーマの移行を実行する前に、ソースにインストールされている拡張機能を確認し、それらが同じバージョンの宛先で利用できることを確認します。一致するバージョンで拡張機能が利用できない場合、スキーマの移行は失敗します。異なるメジャー Postgres バージョンにまたがって移行する場合、移行先で利用可能な拡張機能バージョンはソースのものと一致しない場合があり、移行障害が発生する可能性があります。

```
SELECT name, installed_version
FROM pg_available_extensions
WHERE installed_version IS NOT NULL;
```

バージョンを比較する宛先で同じクエリーを実行します。

移行の完了後、ソースが外部データラッパー FDW 拡張機能を使用している場合、エージェントは FDW 拡張機能自分自身をインストールしますが、FDW オブジェクトは移行しません。宛先で外部サーバー、ユー

ザーマッピング、および外部テーブルを手動で再作成します。

拡張機能の移行の制限の完全なリストについては、拡張機能の移行の制限 を参照してください。

データ移行の準備 DMS エージェント

> **Tip**
>
> DMS エージェントを使用したデータ移行に必要です。

WAL 設定の構成

DMS エージェントは、Postgres 論理デコードを使用して、ログ先行書き込み WAL から変更をキャプチャします。`postgresql.conf` で次のパラメーターを構成します。

1. `wal_level` を `logical` に設定して論理デコードを有効にします。

```
wal_level = logical
```

1. `max_wal_senders` を少なくとも 1 に設定します。他のストリーミングクライアントがデータベースに接続する場合、適切な値について DBA に相談してください。

```
max_wal_senders = 1
```

1. `max_replication_slots` を少なくとも 1 に設定します。組織が Postgres レプリケーションを使用している場合、これは高くなる可能性があります。

```
max_replication_slots = 1
```

1. 実稼働トラフィックが WAL サイズに基づいて要求されたチェックポイントではなく、ほとんどの場合時間指定されたチェックポイントを生成するように、`max_wal_size` を十分に大きく設定します。少なくとも 8 GB をお勧めします。

```
max_wal_size = 8GB
```

`max_wal_size` の設定が小さすぎると、DMS エージェントがストリーミングできる前に Postgres が WAL エントリをドロップし、移行が妨げられるか、速度が低下する可能性があります。DBA に相談して、環境に適切なサイズを決定します。

1. データベースサーバーを再起動して、変更を適用します。

レプリケーショングループロールの作成

Postgres パブリケーションを自動作成するには、移行ユーザーがソーステーブルを所有している必要があります。これらのテーブルは既に別のロールが所有しているため、現在のテーブル所有者と移行ユーザーの両方が属するレプリケーショングループロールを作成します。

```
CREATE ROLE <replication_group>;
GRANT <replication_group> TO <migration_user>;
```

(次のページに続く)

(前のページからの続き)

```
GRANT <replication_group> TO <original_owner>;
ALTER TABLE <table_name> OWNER TO <replication_group>;
```

そこで

- <migration_user> は、移行のために作成した Postgres ユーザーです。
- <original_owner> は、テーブルの元の所有者です。
- <replication_group> は、パブリケーション自動作成のソーステーブルを所有する共有ロールです。
- <table_name> は、移行するテーブルの名前です大文字と小文字が区別されます。

すべてのスキーマにわたって、移行する予定のテーブルごとに ALTER TABLE ... OWNER TO <replication_group> を実行する必要があります。データベース内のすべてのテーブルの所有権を一度に譲渡するには

```
DO $$
DECLARE
  s text;
  t text;
BEGIN
  FOR s, t IN
    SELECT <schemaname>, <tablename> FROM pg_tables
    WHERE <schemaname> NOT IN (pg_catalog, information_schema)
  LOOP
    EXECUTE ALTER TABLE  || quote_ident(s) || . || quote_ident(t) ||  OWNER TO
↪<replication_group>;
  END LOOP;
END;
$$;
```

SSL の構成

ソースデータベースサーバーを accept SSL connections に構成します。DMS エージェントは、ソースに接続するために SSL が必要です。サーバー証明書とサーバープライベートキーを作成します。OpenSSL のようなツールでプライベートキーは生成できます。

構成の検証

DMS エージェントには、ソースデータベースが CDC 移行のすべての要件を満たしているかどうかを確認する検証スクリプトが含まれています。

1. DMS エージェントフォルダーに移動します。

```
cd /opt/cdcagent/reader/
```

1. 移行するすべてのテーブルの配列を<schema_name>.<table_name> 形式で作成します。

```
arr=(schema1.table1 schema1.table2 schema2.table1)
```

1. 検証スクリプトを実行します。

```
PG_USERNAME=postgres PG_PASSWORD=<password> PG_HOST=<host> PG_PORT=5432 DB_NAME=<db_
↪name> DBZ_USERNAME=<migration_user> DBZ_PASSWORD=<migration_user_password> ./
↪postgresConfigValidation.sh "${arr[@]}"
```

合格結果は次のようになります。

```
  ***  [Transporter] - Get PostgreSQL Major Version
PostgreSQL major version: 17
  ***  [Transporter] - Validate WAL Level
wal_level:  logical
[Pass] wal_level is logical.
  ***  [Transporter] - Validate max WAL senders
max_wal_senders: 10
[Pass] max_wal_senders is at least 1.
  ***  [Transporter] - Validate max replication slots
max_replication_slots: 10
[Pass] max_replication_slots is at least 1.
  ***  [Transporter] - Validate max WAL size
max_wal_size: 8 GB
[Pass] max_wal_size is set to the recommended value of 8GB or higher.
  ***  [Transporter] - Validate checkpoints
checkpoints_timed: 100
checkpoints_req: 5
[Pass] Timed checkpoints are more frequent than requested checkpoints.
  ***  [Transporter] - Check <migration_user> user role
[Pass] User <migration_user> is present
[Pass] User <migration_user> has replication permission
  ***  [Transporter] - Check SELECT privilege on the tables to be migrated
[Pass] User <migration_user> has select privilege on all tables to be migrated.
  ***  [Transporter] - Check CREATE privilege on database
[Pass] Role <migration_user> has CREATE privilege on database <db_name>.
```

注釈

続行する前に`[Failed]` または`[Suggestion]` ステータスに対処します。 *[Failed]* チェックは、DMS エージェントがデータ移行を実行する前に解決が必要な問題をブロックしています。 *[Suggestion]* チェックでは移行が許可されますが、対処しないでおくとパフォーマンスに悪影響を与える可能性があります。

次のステップ

Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database

Configuring the EDB Postgres AI agent to register your Postgres database

EDB Postgres AI エージェント `beacon-agent` としてパッケージ化して、HM との安全な接続を確立し、Postgres ソースからスキーマと移行評価データを取得します。

前提条件

- **マシンユーザーとアクセスキー** *Creating a machine user* で作成されます。アクセスキーを環境変数に保存します。

```
export BEACON_AGENT_ACCESS_KEY="baak_your_key_here"
```

- **pg_dump** EDB Postgres AI エージェントは `pg_dump` を使用して Postgres ソースから DDL を抽出します。 `pg_dump` が補助マシンにインストールおよび構成されていることを確認します。インストール手順、互換性ルール、および資格情報のセットアップについては、*Configuring your auxiliary machine or bastion host* を参照してください。

> **注釈**
>
> エージェントは、実行時に `beacon_agent.yaml` で参照される環境変数を置き換えるため、エージェントを実行する前に環境で `$BEACON_AGENT_ACCESS_KEY` を設定するだけで十分です。環境変数は、YAML の静的値よりも常に優先されます。

環境変数を設定するための推奨事項

`$BEACON_AGENT_ACCESS_KEY` や`$DSN` などの機密データは、シークレットマネージャーで管理される環境変数として扱うことをお勧めします。この戦略により、機密の資格情報がプレーンテキストの構成ファイルに保存されるのを防ぎます。環境変数を使用する場合、EDB Postgres AI エージェントの実行中にそれらが利用できることを確認してください。

実稼働環境のための最も安全なアプローチについて、組織の IT 部門に相談してください。

短期間の移行またはテストとデモの場合、アクティブなターミナルセッションで `export` コマンドを使用できます。

HM 接続パラメーターの取得

エージェントには、HM インストールからの特定の値が必要です。次のことについては、管理者に連絡してください。

- **ビーコンサーバーホスト名** `beacon_server` の内部エンドポイント。

HM インストールを構成するときに、管理者は `beacon_server` サービスの内部エンドポイントを指定する必要があります。HM インスタンスの管理者またはインストーラーに、インストールに使用される Helm チャート構成ファイルで `beacon_server` サービスに設定したホスト名 URL を提供するように依頼します。`values.yaml` ファイルの `parameters.upm-beacon.server_host` または `BEACON_SERVICE_DOMAIN_NAME`

に設定した値を検索します。この値は、後で EDB Postgres AI エージェントを構成するために必要になります。

- **ルート証明書パス必要な場合** HM インスタンスが自己署名または内部 CA を使用する場合、`ca_public.crt` ファイルを取得します。エージェントマシンたとえば`/etc/beacon/ca_public.crt` に保存します。

EDB Postgres AI エージェントには、HM インスタンスの `beacon_server` サービスへの信頼できる接続が必要です。 EDB Postgres AI エージェントを実行しているサーバーが HM の TLS 証明書組織のセキュリティインフラストラクチャによって管理されているを本質的に信頼していない場合、HM インスタンスの管理者またはインストーラーに証明書を含むファイルを提供するように依頼してください。後でこの証明書へのディレクトリパスを提供して、EDB Postgres AI エージェントを構成する必要があります。

`beacon_server` およびルート証明書の詳細については、*Administrative tasks* を参照してください。

接続の検証

HM 接続パラメーターを取得したら、補助マシンが Postgres ソースと HM の両方に到達できることを確認します。次のポートを開く必要があります。

Destination	Port	Used by
Postgres host	5432 (default for PostgreSQL, AWS RDS, and Aurora); 5444 (default for EDB Postgres Advanced Server)	Both agents (database connection)
HM *beacon_server*	9443	EDB Postgres AI agent
HM DMS service (*RW_SERVICE_HOST*)	443 (HTTPS)	DMS agent

注釈

ポート 5432 および 5444 はデフォルトです。 Postgres データベースが別のポートでリッスンするように構成されている場合、代わりにそのポートを使用し、ネットワークでオープンしていることを確認します。

nc を使用して、各ホストとポートへの接続を確認します。

```
#  Postgres source

nc -zv <postgres-host-name-or-ip> 5432

#  HM beacon server

nc -zv <beacon_server_hostname> 9443

#  HM DMS service
```

(次のページに続く)

(前のページからの続き)

```
nc -zv <rw-service-hostname> 443
```

接続が成功すると succeeded! が返されます。 Connection refused または Operation timed out が表示される場合、ポートは到達不能であるため、続行する前に開く必要があります。

ソースデータベース接続パラメーターの定義

データソース名 DSN は、Postgres ソースの接続文字列です。

フォーマット

```
DSN="postgresql://<migration_user>:<migration_user_password>@<host>:<port>/<database_
↪name>"
```

エージェントを実行またはサービスを構成する前に、DSN を環境変数として設定します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb"
```

データベースの SSL が無効になっている場合、 ?sslmode=disable を追加します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb?
↪sslmode=disable"
```

RDS インスタンスエンドポイントをホストとして使用します。RDS はデフォルトで SSL を適用するため、?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@myinstance.abc123.us-east-1.rds.
↪amazonaws.com:5432/mydb?sslmode=require"
```

Aurora クラスターエンドポイントをホストとして使用します。Aurora はデフォルトで SSL を適用するため、 ?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@my-cluster.cluster-abc123.us-east-1.
↪rds.amazonaws.com:5432/mydb?sslmode=require"
```

```
DSN="postgresql://migration_user:pa%24%24word@postgres-host:5432/mydb"
```

構成ファイルの準備

1. ホームディレクトリに EDB Postgres AI エージェント構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

注釈

エージェントは`/etc/beacon/`または`${HOME}/.beacon/`で`beacon_agent.yaml`を検索します。最終的なフォールバックとして、実行されたディレクトリを検索します。 !!!

1. このディレクトリ内に、ファイル beacon_agent.yaml を作成します。

```
touch ${HOME}/.beacon/beacon_agent.yaml
```

1. 次のテンプレートをコピーして、新しいファイルに貼り付けます。

このテンプレートでは、 $BEACON_AGENT_ACCESS_KEY および$DSN は、前に構成した環境変数です。テンプレート内のすべての< >プレースホルダーを置き換えます。詳細は、*EDB DMS agent parameter reference* を参照してください。

```
---
agent:
  access_key: $BEACON_AGENT_ACCESS_KEY
  beacon_server: <beacon_server_hostname>:9443
  project_id: <your_project_id>
  providers:
    - "onprem"
  # In most HM deployments, set `root_ca_path` to the path of the ca_public.crt
  # file obtained from your HM administrator (for example, /etc/beacon/ca_public.crt).
  # Leave empty only if HMs certificate is already trusted by the systems CA bundle.
  root_ca_path: ""
  schema_providers:
    # Include "onprem-schema" to enable schema ingestion and migration assessment.
    # Remove this section if you dont want schema information to be collected.
    - "onprem-schema"
provider:
  onprem:
    databases:
      # `resource_id` controls how this database appears in HM Estate.
      - resource_id: "<database_resource_id>"
        dsn: $DSN
        # pg_bin_path: /usr/lib/postgresql/16/bin
        # Optional. Absolute path to the directory containing pg_dump and pg_dumpall.
        # Set when registering databases from different Postgres distributions/
↪versions.
        schema:
          enabled: true
          poll_interval: 15s
general:
  logging:
    # `info` is simplified, use debug during initial setup or
    # troubleshooting for more verbose output.
    level: info
```

1. ファイルを保存します。

EDB Postgres AI エージェントの実行

1. エージェントを起動する前に、接続と構成構文を確認します。

```
beacon-agent validate
```

1. エージェントを実行します。

```
beacon-agent
```

> **Tip**
>
> 他のエージェントモードとオプションについては、 Agent CLI を参照してください。 !!!

1. エージェントが HM へのデータの送信を開始するときに、取り込みプロセスを監視します。数分後、Postgres データベースが HM コンソールの **Estate > Migrations** の下に表示されます。

> **重要**
>
> エージェントは、テーブルと列の変更を動的に検出します。ただし、ビュー、プロシージャー、またはファンクションを変更するには、エージェントを再起動してメタデータを再取り込む必要があります。

本番使用の場合は、 *Running the agent as a service* を参照してください。

移行評価の検証

Postgres データベースがエージェントに認識され、移行の準備ができていることを確認するには、HM コンソールで **Estate > Migrations** に移動します。

データベースがテーブルに表示される場合、次のリソースにアクセスできます。

- **Estate** ページの **マイグレーション** の **移行評価**。データベーススキーマ数、コア数、移行の複雑さ、および作業レベルの概要を提供します。
- **Migrations** タブから特定のデータベースを選択することによりアクセスできる *Viewing migration assessment details* 。このタブは、**スキーマ** の拡張分析、グローバル **評価**、および移行を実行するための他の HM ツールに直接接続する **移行**ハブを提供します。
- ビルトインの移行ポータルでのスキーマの移行に使用できる**自動スキーマ DDL 抽出**。取り込みが完了すると、移行ポータルはデータベースごとにプロジェクトを自動的に作成します。これを使用して、スキーマを準備し、宛先データベースに移行できます。

障害対応

エージェントが HM または Postgres データベースへの接続に失敗する理由はいくつかあります。ここでは、トラブルシューティングに使用できるツールをいくつか紹介します。

beacon-agent validate コマンド

`beacon-agent validate` コマンドは、一般的な接続と構成の問題を特定するための役立つ手順です。

```
beacon-agent validate
```

validate コマンドは、次のことをチェックします。

- **サーバー接続** エージェントが構成済みの `beacon_server` 通常はポート `9443` に接続および認証できることを確認します。
- **データベース接続**構成されている各データベース DSN への接続を試行します。これは完全な接続テストであり、単なる構文または URL エンコーディングのチェックではありません。
- **構成表示** TLS 設定、プロジェクト ID、プロバイダー、およびデータベースごとの設定を表示し、構成の誤りを特定するのに役立ちます。

beacon-agent ログ

エージェントが取り込み中にエラーを報告した場合、詳細についてはログを確認できます。フォアグラウンドでエージェントを実行している場合、ログはターミナルに出力されます。エージェントを systemd サービスとして設定する場合、`journalctl` を使用してログを表示します。

次に、ログに表示される一般的なエラーメッセージをいくつか示します。

Error	Cause	Fix
tls: failed to verify certificate: x509: certificate signed by unknown authority	The agent can't verify HM's TLS certificate because the CA isn't trusted by the system.	Set *root_ca_path* in *beacon_agent.yaml* to the path of the *ca_public.crt* file obtained from your HM administrator. If you can't obtain the certificate, you can set *plaintext: true* in the agent config as a temporary workaround (not recommended for production).
rpc error: code = Unknown desc = no rows in result set	The beacon-agent version is incompatible with the server.	Upgrade to beacon-agent v1.384.0 or later.

エージェントをサービスとして実行する

継続的なメトリックストリーミングのために、また Postgres データベースが **HM コンソール>エステート**に表示されたままであることを確認するには、systemd サービスとして実行するようにエージェントを構成します。サービスとして実行すると、システムの起動時に自動的に起動し、エラーが発生した後に再起動します。

サービスがないと、エージェントは手動で実行している間のみメトリックをストリーミングします。データベースは、エージェントがストリーミングを停止してから 72 時間後に HM 不動産から消えます。

注釈

beacon_agent.yaml たとえば *$BEACON_AGENT_ACCESS_KEY* や `$DSN` などで参照される環境変数が、実行時にサービスにアクセスできることを確認します。スキーマ抽出資格情報の `.pgpass` ファイルを構成した場合 **環境の構成** で説明しているように、そのファイルが `User=` で指定されたユーザーのホームディレクトリに存在することを確認します。

1. サービスファイルを作成します。この例では `vi` を使用しますが、任意のテキストエディタを使用できます。

```
sudo vi /usr/lib/systemd/system/beacon-agent.service
```

オペレーティングシステム別のデフォルトのサービスファイルパス

Operating system	Default path
RHEL	*/usr/lib/systemd/system/beacon-agent.service*
Rocky Linux	*/usr/lib/systemd/system/beacon-agent.service*
AlmaLinux	*/usr/lib/systemd/system/beacon-agent.service*
Ubuntu	*/etc/systemd/system/beacon-agent.service*
Debian	*/lib/systemd/system/beacon-agent.service*

1. 次のテンプレートをファイルに貼り付けて、すべての< >プレースホルダーを置き換えます。

```
[Unit]
Description=EDB Postgres AI Agent
After=network.target

[Service]
Type=simple
User=<agent_os_user>
WorkingDirectory=</home/agent_os_user/>
ExecStart=</usr/local/bin/beacon-agent>
Restart=on-failure
RestartSec=60

[Install]
WantedBy=multi-user.target
```

Setting	Placeholder	Description
User=	*<agent_os_user>*	エージェントを実行する OS ユーザー。ソースデータベースの準備時に作成した専用の OS ユーザーを使用します。

次のページに続く

表 37 – 前のページからの続き

Setting	Placeholder	Description
WorkingDirectory=	*</home/agent_os_user/>*	`.beacon`フォルダーが保存されるディレクトリ。これは通常、`<agent_os_user>`のホームディレクトリです。
ExecStart=	*</usr/local/bin/beacon-agent>*	`beacon-agent`バイナリへのフルパス。パスは、エージェントがインストールされた方法と場所によって異なります。

1. systemd をリロードし、サービスを有効にして開始します。

```
sudo systemctl daemon-reload
sudo systemctl enable beacon-agent.service
sudo systemctl start beacon-agent.service
```

1. ログを確認して、サービスが正常に開始されたことを確認します。

```
journalctl -u beacon-agent.service
```

関連トピック

- スキーマ取り込みのより詳細な制御たとえば、スキーマを除外または含める場合、*Controlling schema ingestion* を参照してください。
- 構成パラメーターの完全なセットについては、*EDB DMS agent parameter reference* を参照してください。
- 管理者がルート証明書または `beacon_server` 値の取得に関するガイダンスが必要な場合は、*Administrative tasks* を参照してください。

次のステップ

Preparing your HM-managed destination database

Preparing your self-managed destination database

移行の宛先として機能する自己管理の Postgres データベースを準備します。

宛先として使用する自己管理データベースが既にある場合、 *SSL の構成*にスキップできます。続行する前に、移行するテーブルと移行中に生成される負荷がサーバー上の既存のワークロードを干渉しないことを確認してください。

宛先データベースの作成

宛先データベースがまだない場合は、作成します。この例では、選択した Postgres データベースディストリビューションをインストールし、サーバーを初期化していることを前提としています。

1. 宛先の Postgres サーバーに接続します。

```
psql -p <PG_PORT> -U <PG_USERNAME>
```

1. 移行先として使用する空のデータベースを作成します。この例では、宛先データベースは tr_target です。

```
CREATE DATABASE tr_target;
```

SSL の構成

SSL 接続を受け入れるように宛先の Postgres サーバーを構成して、ライタモードの EDB DMS エージェントが安全に接続できるようにします。SSL を有効にするには、サーバー証明書とサーバープライベートキーを作成しますたとえば、OpenSSL を使用して。

次の例では、Ubuntu 上の PostgreSQL 17 サーバーの OpenSSL を使用して自己署名証明書とキーを作成します。bin および data ディレクトリは、OS、 *Supported Postgres distributions* 、およびバージョンによって異なる場合があります。

1. SSL 証明書とキーを生成します。

```
openssl req -new -x509 -days 365 -nodes -out server.crt -keyout server.key
```

ターミナルのプロンプトに従って、ファイルを作成します。

1. 作成したファイルを Postgres データディレクトリに移動します。

```
sudo cp server.crt server.key /var/lib/postgresql/17/<data_directory>
```

1. 必要な権限を設定します。

```
sudo chmod 600 /var/lib/postgresql/17/<data_directory>/server.key
sudo chmod 600 /var/lib/postgresql/17/<data_directory>/server.crt
sudo chown postgres:postgres /var/lib/postgresql/17/<data_directory>/server.crt /var/
↪lib/postgresql/17/<data_directory>/server.key
```

1. postgresql.conf で SSL を有効にします。

```
ssl = on
ssl_cert_file = server.crt
ssl_key_file = server.key
```

1. pg_hba.conf でローカル SSL 接続を許可します。

```
hostssl all             all             0.0.0.0/0               md5
```

1. サーバーを再起動して変更を適用します。

```
/usr/lib/postgresql/17/bin/pg_ctl -D /var/lib/postgresql/17/<data_directory> restart
```

移行ロールの作成

ライタモードの EDB DMS エージェントには、宛先データベースへの書き込みアクセスが必要です。既存のスーパーユーザーを使用するか、専用の移行ロールを作成できます。この例では、専用のロールを作成します。

1. ログインアクセスを持つロールを作成します。

```
CREATE ROLE <MIGRATION_ROLE_WRITER> WITH LOGIN PASSWORD <ROLE_PASSWORD>;
```

<MIGRATION_ROLE_WRITER> は、宛先データベースへのアクセスに使用するロールの名前です。

1. レプリケーションセッションの制御を付与します。

```
GRANT SET ON PARAMETER session_replication_role TO <MIGRATION_ROLE_WRITER>;
```

> **注釈**
>
> レプリケーションセッションの制御の付与は、Postgres 15 以降でのみサポートされています。宛先データベースが Postgres 14 以前である場合、代わりにスーパーユーザーを使用します。 !!!

次の手順で構成するときに、 DBCONFIG_DATABASES_0__USERNAME および DBCONFIG_DATABASES_0__PASSWORD 値を介してこのロールを DMS ライタエージェントに提供します。

次のステップ

Migrating Postgres schemas

Migrating Postgres schemas

データの移行を開始する前に、スキーマの移行を処理する方法を選択します。HM は 2 つのアプローチをサポートしています。

- **簡素化推奨** HM は、データ移行セットアップの一部としてスキーマの移行と制約処理を自動的に管理します。
- **手動** Migration Toolkit または pg_dump を使用して自分自身でスキーマをエクスポートおよびインポートし、各ソーステーブルのキー構造に基づいて制約を処理します。

手動でのスキーマの準備は必要ありません。 HM は、後で **スキーマとデータ**スコープを選択すると、スキーマの移行と制約の順序付けを自動的に処理します。続行する前に、次を確認します。

- EDB Postgres AI エージェントはソーススキーマを取り込み、データベースには移行ポータルプロジェクトが存在します。確認するには、HM コンソールで **Estate > Migrations** に移動し、**Database Name** 列でデータベースを選択し、 **MP Project** 列にプロジェクトが表示されることを確認します。
- 後の手順で DMS リーダーエージェントを構成するときに、DBCONFIG_DATABASES_<N>__RESOURCEID を beacon_agent.yaml と同じ resource_id 値に設定します。 **スキーマとデータ** および**スキーマのみ** スコープは、これがないと失敗します。

次のステップに進みます。

データの移行に EDB DMS を使用する前に、スキーマを宛先データベースにエクスポートおよびインポートします。DMS はデータのみを移行します。移行を開始する前に、宛先データベースにスキーマを設定する必要があります。

コンストレインの処理

制約を処理する方法は、制約タイプと各ソーステーブルのキー構造によって異なります。

Constraint type	Action	Notes
Primary key	今すぐ除外する — DMS は自動的に適用されます	The DMS writer applies the primary key when the migration enters the streaming phase, using it to deduplicate snapshot data.
Unique key (source table has a primary key)	今すぐ除外、ストリーミング開始後に適用	Primary key-based deduplication ensures no duplicate rows remain when you apply unique constraints after streaming starts.
Unique key (source table has no primary key)	現在サポートされていません	Use the simplified path instead — select the **Schema and data** scope when creating the migration to let HM handle schema migration automatically.
NOT NULL	今すぐ含める	*NOT NULL* validation adds no meaningful overhead to the data load, unlike foreign key and check constraint validation.
Foreign key, check, and exclusion	今すぐ除外、移行後に再適用	These cause unnecessary CPU and memory overhead while data is actively being migrated.

注釈

主キーまたは一意キーのないテーブルの場合、移行の前後にキー制約は必要ありません。 DMS は、これらのテーブルをキーレスとして扱い、少なくとも 1 回の配信を提供します。移行後の検証の一部として、行の重複を削除します。

スキーマのエクスポート

EDB Migration Toolkit を使用してスキーマをエクスポートすることをお勧めします。MTK の offline migration 機能は、直接接続せずに宛先に適用できる SQL スクリプトを生成します。他の有効なエクスポート方法には、`pg_dump` /`pg_restore` 、pgAdmin、または `psql` などがあります。

Migration Toolkit を使用する場合

- `-offlineMigration` オプションを使用して SQL スクリプトを生成します。
- `-schemaOnly <schema_scope>` オプションを使用して、スキーマのみをエクスポートします。
- インポートする前に、生成されたスクリプトから主キー、一意、外部キー、チェック、および除外コンストレインを除外コメントアウトします。

- オブジェクトを宛先の別の名前にマップする予定がある場合は、インポートする前に DDL スクリプトで直接編集します。名前が変更されたオブジェクトを参照するすべての依存オブジェクトビュー、コンストレイン、およびその他の参照を更新します。詳細は、*Mapping schema, tables, and columns* を参照してください。

スキーマのインポート

DDL を準備し、遅延制約を除外した後、宛先データベースに接続し、生成された offline migration script を実行して SQL ファイルを適用します。

> **注釈**
>
> 移行が完了した後、除外された一意の外部キー、チェック、および除外制約を適用します。 DMS ライタは、主キーコンストレインを自動的に適用します。手動で再適用しないでください。除外されたコンストレインを追跡して、必要なときに再適用できます。

次のステップ

Configuring and running the EDB DMS agent

Configuring and running the EDB DMS agent

宛先は自己管理の Postgres データベースであるため、この移行には 2 つの DMS エージェントインスタンスが必要です。1 つはリーダーモードでソースから変更を読み取り、もう 1 つはライタモードでそれらを宛先に適用します。移行を作成する前に、両方をインストールして構成します。

前提条件

- DMS エージェントを実行しているマシンに Java / OpenJDK 17 以降がインストールされていることを確認します。
- *Install the EDB DMS agent* 。

移行資格情報の作成

移行資格情報は、DMS エージェントをハイブリッドマネージャー HM に対して認証します。HM プロジェクトごとに 1 セットの資格情報が必要で、リーダーエージェントとライタエージェントの両方が同じ資格情報を使用します。

1. HM コンソールで、移行に使用しているプロジェクトを選択します。
2. **移行>資格情報** を選択します。
3. **Create Migration Credential** を選択し、名前を入力して、 **Create Migration Credential** を選択します。 `credentials.zip` ファイルがマシンにダウンロードされます。

> **注釈**

資格情報は安全に保存します。それらを失った場合は、このセットを取り消し、新しいセットを作成する必要があります。 !!!

1. ダウンロードしたファイルを解凍し、資格情報フォルダーを DMS エージェントをインストールした各マシンたとえば$HOME/credentials/ にコピーします。別のパスを使用する場合は、一致するように各スクリプトの CREDENTIAL_DIRECTORY_PATH を更新します。

フォルダーには次のものが含まれています

リーダー DMS エージェントの構成

リーダーエージェントは、ソース Postgres データベースに接続し、変更を HM にストリーミングします。

1. 書き込み権限を持つリーダーマシンで/opt/cdcagent/run-cdcagent.sh を開きます。
2. 必要な変数を設定します。オプションのパラメーターはデフォルトでコメントアウトされています。環境に適用されるものはコメントを外して構成します。

詳細は、 *EDB DMS agent parameter reference* を参照してください。

```
#!/bin/bash -e
# run-cdcagent.sh
#
# This script provides a convenient place to specify
# environment variables used to configure the
# EDB Data Migration Service Agent.
#

##########################################
# DMS Agent General Configuration       #
##########################################

# This ID is used to identify DMS Agent
# and is specified by the user.
export DBCONFIG_ID=<source_name_of_choice>

# Determine the run mode of the DMS Agent
# The run mode can be set to either reader or writer
# reader mode is used to read changes from a source database
# writer mode is used to write changes to a target database
export RUN_MODE=reader

# This is the DMS backend service used by the Agent
# Obtain this value from the HM console Sources or Destinations migration pages
export RW_SERVICE_HOST=<rw_service_host>

##########################################
# DMS Agent TLS Configuration           #
```

(次のページに続く)

(前のページからの続き)

```
##########################################

# You need to create migration credentials in EDB Postgres AI platform
# and set the path of the credentials directory
export CREDENTIAL_DIRECTORY_PATH=$HOME/credentials

##########################################
# DMS Agent DB Configuration            #
##########################################

# A sample configuration to create a single Postgres database connection:
export DBCONFIG_DATABASES_0__TYPE=POSTGRES
export DBCONFIG_DATABASES_0__HOSTNAME=<localhost/host>
export DBCONFIG_DATABASES_0__PORT=<5432>
# The CATALOG is the database name
export DBCONFIG_DATABASES_0__CATALOG=<source>
export DBCONFIG_DATABASES_0__USERNAME=<postgres>
# The password env can be set without specifying it here
# but the env structure looks like this
export DBCONFIG_DATABASES_0__PASSWORD=<password>
# Required for Schema and data or Schema only scope.
# Must match resource_id in beacon_agent.yaml.
#export DBCONFIG_DATABASES_0__RESOURCEID=<resource_id>

# You can increase the index to configure more than
# one database for the DMS Agent
#export DBCONFIG_DATABASES_1__TYPE=POSTGRES
#export DBCONFIG_DATABASES_1__HOSTNAME=<localhost/host>
#export DBCONFIG_DATABASES_1__PORT=<5432>
#export DBCONFIG_DATABASES_1__CATALOG=<source>
#export DBCONFIG_DATABASES_1__USERNAME=<postgres>
#export DBCONFIG_DATABASES_1__PASSWORD=<password>
#export DBCONFIG_DATABASES_1__RESOURCEID=<resource_id>

##########################################
# Optional Parameters Below             #
##########################################

# FIPS mode can be enabled by setting the FIPS_MODE env variable to true
#export FIPS_MODE=true

# Configure logging
# Loglevel for the entire Quarkus application and underlying components
# WARNING: Enabling this loglevel is extremely verbose
#export QUARKUS_LOG_LEVEL=DEBUG
```

(次のページに続く)

(前のページからの続き)

```
# Loglevel for the EDB package
# Preferred mode for debugging issues related to the logic of the agent
#export QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL=DEBUG

cd "$(dirname "$0")"

echo "Starting in mode: $RUN_MODE"

case "$RUN_MODE" in
  reader)
    java -jar ./reader/quarkus-run.jar
    ;;
  writer)
    java -jar ./writer/quarkus-run.jar
    ;;
  *)
    echo "Unknown RUN_MODE: $RUN_MODE"
    exit 1
    ;;
esac
```

リーダー DMS エージェントの実行

DMS エージェントディレクトリに移動し、構成スクリプトを実行します。

```
cd /opt/cdcagent
./run-cdcagent.sh
```

スクリプトは、必要なすべての環境変数をエクスポートし、エージェントプロセスを開始します。リーダーエージェントは、実行されると、Postgres ソースを HM に登録します。次のセクションに進んで、正しく表示されることを確認します。

HM でのソースの検証

1. HM コンソールで、プロジェクトを選択します。
2. **移行>ソース** を選択します。

データベースは、名前、タイプ、エージェント ID、およびステータスとともにリストされます。このページのすべてのフィールドの説明については、デプロイメントリソースを理解する を参照してください。

ライタ DMS エージェントの構成

ライタエージェントは、宛先の Postgres データベースに接続し、HM からのストリーミング変更を適用します。DMS エージェントのライタインスタンスをインストールしたマシンで構成します。

1. 書き込み権限でライタマシンで/opt/cdcagent/run-cdcagent.sh を開きます。

2. 必要な変数を設定します。オプションのパラメーターはデフォルトでコメントアウトされています。環境に適用されるものはコメントを外して構成します。

詳細は、 *EDB DMS agent parameter reference* を参照してください。

```
#!/bin/bash -e
# run-cdcagent.sh
#
# This script provides a convenient place to specify
# environment variables used to configure the
# EDB Data Migration Service Agent.
#

##########################################
# DMS Agent General Configuration       #
##########################################

# This ID is used to identify DMS Agent
# and is specified by the user.
export DBCONFIG_ID=<destination_name_of_choice>

# Determine the run mode of the DMS Agent
# The run mode can be set to either reader or writer
# reader mode is used to read changes from a source database
# writer mode is used to write changes to a target database
export RUN_MODE=writer

# This is the DMS backend service used by the DMS Agent
# Obtain this value from the HM console Sources or Destinations migration pages
export RW_SERVICE_HOST=<rw_service_host>

##########################################
# DMS Agent TLS Configuration           #
##########################################

# You need to create migration credentials in EDB Postgres AI platform
# and set the path of the credentials directory
export CREDENTIAL_DIRECTORY_PATH=$HOME/credentials

##########################################
# DMS Agent DB Configuration            #
##########################################

# A sample configuration to create a single Postgres database connection:
export DBCONFIG_DATABASES_0__TYPE=POSTGRES
export DBCONFIG_DATABASES_0__HOSTNAME=<localhost/host>
export DBCONFIG_DATABASES_0__PORT=<5432>
```

(次のページに続く)

(前のページからの続き)

```
# The CATALOG is the database name
export DBCONFIG_DATABASES_0__CATALOG=<destination_db_name>
export DBCONFIG_DATABASES_0__USERNAME=<MIGRATION_ROLE_WRITER>
# The password env can be set without specifying it here
# but the env structure looks like this
export DBCONFIG_DATABASES_0__PASSWORD=<password>

# You can increase the index to configure more than
# one database for the DMS Agent
#export DBCONFIG_DATABASES_1__TYPE=POSTGRES
#export DBCONFIG_DATABASES_1__HOSTNAME=<localhost/host>
#export DBCONFIG_DATABASES_1__PORT=<5432>
#export DBCONFIG_DATABASES_1__CATALOG=<destination_db_name>
#export DBCONFIG_DATABASES_1__USERNAME=<MIGRATION_ROLE_WRITER>
#export DBCONFIG_DATABASES_1__PASSWORD=<password>

##########################################
# Optional Parameters Below              #
##########################################

# FIPS mode can be enabled by setting the FIPS_MODE env variable to true
#export FIPS_MODE=true

# Configure logging
# Loglevel for the entire Quarkus application and underlying components
# WARNING: Enabling this loglevel is extremely verbose
#export QUARKUS_LOG_LEVEL=DEBUG
# Loglevel for the EDB package
# Preferred mode for debugging issues related to the logic of the agent
#export QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL=DEBUG

cd "$(dirname "$0")"

echo "Starting in mode: $RUN_MODE"

case "$RUN_MODE" in
  reader)
    java -jar ./reader/quarkus-run.jar
    ;;
  writer)
    java -jar ./writer/quarkus-run.jar
    ;;
  *)
    echo "Unknown RUN_MODE: $RUN_MODE"
    exit 1
```

(次のページに続く)

(前のページからの続き)

```
    ;;
esac
```

ライタ DMS エージェントの実行

DMS エージェントディレクトリに移動し、構成スクリプトを実行します。

```
cd /opt/cdcagent
./run-cdcagent.sh
```

ライタエージェントが実行されると、Postgres の宛先を HM に登録します。次のセクションに進んで、正しく表示されることを確認します。

HM での宛先の検証

1. HM コンソールで、プロジェクトを選択します。
2. **移行>宛先** を選択します。

データベースは、名前、タイプ、エージェント ID、およびステータスとともにリストされます。このページのすべてのフィールドの説明については、多様な移行先 を参照してください。

次のステップ

Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database

Migrating data from a self-managed Postgres database to a different self-managed Postgres database

両方の DMS エージェントが実行され、スキーマがインポートされるか、HM インポートの準備が整うと、HM コンソールで移行を作成する準備が整います。このページでは、完全な移行ライフサイクルについて説明します。移行の作成、その進行状況の監視、ファイナライズ、および移行後の制約の適用。

以下の手順は、前のページで選択したアプローチによって異なります。選択したものと一致するタブを選択します。

移行の作成

移行により、Postgres ソースと自己管理の宛先データベース間の同期が HM を介して確立されます。

1. HM コンソールで、移行を管理するプロジェクトを選択します。
2. プロジェクトで、 **移行>移行** を選択します。
3. [**移行**] ページで、[**Create New Migration**] を選択します。
4. **Create OLTP Migration** ページで、名前を移行に割り当てます。
5. 移行のソースを選択します。リーダー DMS エージェントの ID は、 **Source** メニューにリストされています。
6. **Scope** の下で、アプローチに一致するオプションを選択します。

スキーマとデータ を選択します。このオプションを選択すると、 HM:

1. データベースを EDB Postgres AI エージェントに登録したときに自動的に作成された移行ポータルプロジェクトからスキーマをソースします。
2. 制約なしで宛先にテーブルを作成するため、初期スナップショットコピーは作成されません制約の検証により遅くなります。
3. スナップショットが完了したら、主キーと一意の制約があるテーブルの場合を適用して、DMS がストリーミング変更を 1 回だけ配信できます。PK または UK のないテーブルは少なくとも 1 回の配信にフォールバックし、検証中に重複排除が必要です。
4. カットオーバーまで残りのスキーマインデックス、外部キー、チェック制約、`NOT NULL` などを遅延します。オプションで、この作業のほとんどを**ファイナライズスキーマコンカレントリー**を介してストリーミングと同時に適用して、カットオーバーウィンドウを縮小できます。 ストリーミング移行の仕上げ を参照してください。
5. 移行を完了とマークした後、遅延外部キーを検証し、シーケンスを同期し、残りの軽量ファイルを適用します最終の**スキーマファイナライズ** ステージのタスク。

> **注釈**
>
> テーブルデータを移行せずにデータベース構造を移行する場合は、 **スキーマのみ**を使用します。!!!

データのみを選択します。スキーマは、前の手順で既に手動でインポートされています。

1. スナップショットタイプを選択します ***Snapshot*** または ***Snapshot + Streaming*** 。
2. **Destination Type** で、移行の宛先として自己管理インスタンスを選択します。ライタ DMS エージェントの ID は、**Destination** メニューにリストされています。次に、 **Next** を選択します。
3. **Review your Tables** タブで、移行に含めるテーブルと列を選択します。テーブルを展開して、その列とオプションで *map source columns to different destination column names* を表示します。列マッピングは、 **データのみ**スコープでのみ使用できます。
4. **Create Migration** を選択します。

HM コンソールには、 *new migration* が表示されます。**スナップショット+ストリーミング**を選択した場合、DMS エージェントは、移行が **実行中**状態を示している間、データの変更を継続的にストリーミングします。

> **注釈**
>
> DMS エージェントは、データ変更のみをストリーミングします。DDL 変更 DDL 変更はストリーミングしません。

進行状況のモニタリング

移行リストビュー

Migrations ページには、プロジェクト内のすべての移行の概要が表示されます。**移行の進行状況** 列には、移行ごとに 3 つのメトリックが表示されます。

- **テーブルの合計数** — この移行に含まれるテーブルの合計
- **スナップショットテーブルが完了しました** —スナップショットステージが終了したテーブル
- **移行された行** — すべてのテーブルで移行された累積行数

移行の詳細ビュー

リストから移行を選択して、3 つのタブがある詳細ページを開きます。

- **概要** - 現在の移行状態と、ソースおよび宛先データベースに関する一般的な情報を表示します。**スナップショット+ストリーミング** の移行の場合、ステータスが**ストリーミング** のときに移行はキャッチアップされます。**スナップショット** の移行の場合、ステータスが **Completed** になると完了します。
- **Tables** —移行のために選択されたすべてのテーブルを表示します。テーブルごとに、ソースおよび宛先テーブル、スキーマ名、スナップショットステータス、マップされた列、移行された行、および移行速度を確認できます。**+** アイコンを選択してテーブルを展開し、列レベルのマッピングを表示します。
- **Error History** —この移行についてログに記録されたすべてのエラーをリストします。 **i** アイコンを選択して、特定のエラーの完全なエラーログを開きます。移行が停止または失敗した場合、ここでのアクティブなエラーは、トラブルシューティングを開始する場所を示します。

DMS エージェントログ

HM コンソールは、すべての DMS エージェントエラーを表示するわけではありません。移行がスタックしているか、進行中のテーブルがない場合は、`run-cdcagent.sh` を実行したターミナルで `ERROR` ログエントリを確認します。多くの場合、これらには根本的な原因が含まれています。一般的な問題には、DMS エージェントと HM 間のバージョンの非互換性、および接続または資格情報の障害が含まれます。

ストリーミング移行の仕上げ

> **注釈**
>
> このセクションは、**スナップショット+ストリーミング**の移行にのみ適用されます。 **スナップショット**の移行の場合、DMS は移行を自動的に完了としてマークします。

カットオーバー前のキャッチアップのモニタリング

ソースへの書き込みを停止する前に、宛先に追いつくまでの時間を与えます。**移行の詳細**ページを使用して、行数とレプリケーションラグを追跡します。デスティネーションが一貫してソースに近づいた後にのみ、カットオーバーを開始します。

スキーマのファイナライズを同時に行うことをお勧めします

Schema および data スコープを使用した場合、HM はストリームの実行中にほとんどのスキーマオブジェクトを適用できます。プライマリキー、一意制約、インデックス `CREATE INDEX CONCURRENTLY` でビルドされるため、ライブ書き込みがブロックされません、外部キー (`NOT VALID`)、チェックコンストレイン、`NOT NULL` コンストレインなどとして追加されます。外部キー検証、シーケンス同期、およびその他の軽量タスクは、カットオーバー後の**スキーマファイナライズ** ステージで実行されます。

この作業を実行すると、読み取り専用のカットオーバーウィンドウが同時に縮小されます。テーブルが大きくなり、インデックスの数が増えるほど、節約される時間は長くなります。

すべてのテーブルがスナップショットを完了し、移行がストリーミングフェーズに入ると、**移行の詳細** ページに、 **ファイナライズスキーマコンカレントリー**ボタンを備えた **スキーマファイナライズ可能**アラートが表示されます。ボタンはストリーミング中は利用可能なままであるため、目的地が追いつくまでにどれだけ近いかに基づいて、瞬間を選択できます。

1. **Migration Details** ページで **Finalize Schema Concurrently** を選択します。
2. HM は、ストリーミングが継続されている間、遅延オブジェクトを適用します。移行は**実行中** 状態のままで、進行状況サブステップには**ファイナライズスキーマコンカレントリー**というラベルが付いています。

サブステップが完了したら、カットオーバーに進みます。

> **注釈**
>
> **ファイナライズスキーマコンカレントリー**は、ストリーミングが有効になっている**スキーマおよびデータ**スコープで作成された移行でのみ使用できます。

この手順をスキップすると、移行を完了としてマークした後、同じオブジェクトが連続して適用されます。スキーマの最終化ステージを参照してください。この場合、インデックスは非並列で構築され、制約は排他的ロックの下で検証されるため、カットオーバーウィンドウは大きくなります。

手動パスには適用されません。移行前にスキーマを適用し、カットオーバー後に遅延コンストレインを手動で再適用します。遅延コンストレインの適用とデータの検証 を参照してください。

カットオーバーの実行

トラフィックをソースから宛先にスイッチする準備ができたら

1. ソースデータベースのすべての書き込みを停止します。
2. HM コンソールで、デスティネーションが最後のソース変更を適用したことを確認します。
3. **Migration Details** ページで、 **Mark as Completed** を選択します。

これにより、ストリーミングプロセスが停止します。

スキーマの最終化ステージ

スキーマおよびデータ スコープで作成された移行の場合、移行は **Mark as Completed** の後、自動的に **スキーマファイナライズ**ステージに入ります。このステージで実行されるものは、最初にコンカレントファイナライズを実行したかどうかによって異なります。

- **同時ファイナライズを実行した場合** HM は、軽量の残りの作業外部キー検証、シーケンス同期、およびその他の遅延メタデータタスクを実行します。このステージは短いです。
- **同時ファイナライズをスキップした場合** HM は、完全な遅延セットプライマリキー、一意制約、インデックス、外部キー、チェック制約、`NOT NULL` などを適用した後、上記の軽量タスクを適用します。このステージにはかなりの時間がかかり、ソースが読み取り専用であるときに実行されます。

移行は、 **スキーマのファイナライズ**ステージが完了した後にのみ完全に完了します。進行状況は、 **移行の詳細**ページに表示されます。

データのみスコープでは、自動スキーマファイナライズは実行されません。遅延コンストレインの適用とデータの検証 に進み、移行前に延期した外部キー、一意、チェック、および除外制約を再適用します。

遅延コンストレインの適用とデータの検証

遅延コンストレインの再適用

HM は、 **スキーマファイナライズ**ステージ中に、すべての遅延コンストレインと同期シーケンスを自動的に適用しました。手動操作は必要ありません。スキーマの最終化ステージ を参照してください。

移行の初めに、外部キー、一意、チェック、および除外制約を除いたスキーマを準備およびインポートしました。宛先データベースに接続し、これらのコンストレインを再適用します。

主キー制約は、移行がストリーミングフェーズに入ったときに、DMS ライタによって自動的に適用されました。`NOT NULL` コンストレインは移行前に適用され、現在アクションは必要ありません。

必要に応じて行の重複を削除する

主キーも一意制約もなかったテーブルの場合、EDB DMS エージェントは少なくとも 1 回の配信を提供します。これにより、重複行が発生する可能性があります。検証手順の一部として重複排除を実行します。

移行したデータの検証

ソースデータベースと宛先データベースを比較して、すべてのデータが正常に移行されたことを確認します。これには、LiveCompare を使用できます。

関連トピック

- LiveCompare —移行の完全性を確認するための行レベルのデータ比較ツール
- 用語 —スナップショット、ストリーミング、およびその他の移行概念の定義

Migrate your self-managed Postgres database to an HM-managed database

これらは、自己管理の Postgres データベースから Hybrid ManagerHM によって管理されるデータベースクラスターへのエンドツーエンドの移行に関係するタスクです。このプロセスでは、2 つのエージェントを使用します。EDB EDB Postgres AI エージェント `beacon-agent` は、Postgres データベースを HM に登録し、評価用のスキーマを取り込みます、と実際のデータ移行を実行する EDB DMS エージェント `cdcagent` 。

1. ** *Configuring your auxiliary machine or bastion host* ** —両方の移行エージェントをホストするマシンを構成します。このマシンは、Postgres ソースと HM 間のブリッジとして機能します。
2. ** *Preparing your AWS RDS/Aurora PostgreSQL source database* ** — 必要なユーザーを作成し、 EDB Postgres AI エージェントを使用したスキーマの取り込みと DMS エージェントを使用したデータ移行用に Postgres データベースを構成します。
3. ** *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* ** — エージェントをソースデータベースと HM に接続して、スキーマの取り込みと移行評価を有効にします。
4. ** *Preparing your HM-managed destination database* ** —移行されたスキーマとデータを受信する HM 管理データベースクラスターを作成または特定します。
5. ** *Migrating Postgres schemas* ** —HM がスキーマの移行と制約の順序を自動的に処理する簡素化パスか、自分でスキーマをエクスポート、準備、およびインポートする手動パスを選択します。
6. ** *Configuring and running the EDB DMS agent* ** —移行資格情報を作成し、Postgres ソースから変更を読み取るようにリーダーモードで DMS エージェントを構成し、ソースが HM に表示されることを確認します。
7. ** *Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database* ** — HM コンソールでデータ移行を作成して実行し、その進行状況を監視し、ストリーミングをファイナライズし、移行後のコンストレインを適用します。

Configuring your auxiliary machine or bastion host

EDB Postgres AI エージェント `beacon-agent` とデータ移行サービス DMS エージェント `cdcagent` は、Postgres データベースサーバー自分自身または別の補助マシンまたは要塞ホストで実行できます。ほとんどの運用環境では、専用の補助マシンでエージェントを実行することが優先されます。これにより、データベースサーバーへの追加ソフトウェアのインストールが回避され、単一のエージェントインスタンスでマルチプルのサーバー上のデータベースに接続できます。

必要なソフトウェア

エージェントを実行する前に、補助マシンに次のソフトウェアをインストールしていることを確認します。

- **Java 17 以降** — DMS エージェント `cdcagent` に必要です。`java -version` で確認します。
- **EDB Postgres AI エージェント** `beacon-agent` データベース登録、スキーマ評価、およびスキーマの移行に必要です。*beacon-agent version compatibility matrix* を確認します。次に、 install a compatible version of the EDB Postgres AI agent 。
- **DMS エージェント** `cdcagent` - データの移行と変更データキャプチャ CDC を処理します。インストールする前に、*DMS agent version compatibility* を確認してください。次に、 *install a compatible version of the DMS agent* 。

環境の構成

スキーマの取り込みを有効にすると、beacon-agent は pg_dump を使用して Postgres ソースから DDL を抽出します。Postgres クライアントツールパッケージには、pg_dump ユーティリティが含まれています。マシンで既に利用可能な場合があります。最初に確認します

```
pg_dump --version
```

コマンドが見つからない場合は、ソースデータベースディストリビューションと一致する install the client tools 。

インストールしたら、pg_dump バイナリが 2 つの互換性ルールを満たしていることを確認します。

- **バージョンルール** — pg_dump メジャーバージョンは、ソースデータベースのメジャーバージョン以上である必要があります。たとえば、ソースが Postgres 16 を実行している場合、pg_dump 16 以降が必要です。エージェントは起動時にこれを検証し、バージョンに互換性がない場合はエラーを報告します。
- **ディストリビューションルール** — pg_dump バイナリは、ソースデータベースと同じ Postgres ディストリビューションから取得する必要があります。各ディストリビューションの pg_dump は、ディストリビューション固有のカタログテーブルにアクセスします。一致しないバイナリたとえば、EDB Postgres Advanced Server ソースに対するコミュニティ PostgreSQL pg_dump を使用すると、予想されるカタログオブジェクトが存在しないため失敗します。

ホストでの複数の Postgres バージョンの管理

同じマシンに複数の Postgres バージョンがある場合は、正しい pg_dump が PATH の最初に表示されることを確認します。

```
which pg_dump
pg_dump --version
```

Debian および Ubuntu システムでは、update-alternatives を使用してアクティブなバージョンを管理します。

```
sudo update-alternatives --config pg_dump
```

複数のディストリビューションまたはバージョンからのデータベースの登録

単一の beacon-agent インスタンスは、さまざまな Postgres ディストリビューションたとえば、コミュニティ PostgreSQL および EDB Postgres Advanced Server またはさまざまなメジャーバージョンからデータベースを同時に登録できます。バージョンと配布ルールはデータベースごとに適用されるため、各データベース接続は異なる pg_dump バイナリを必要とする場合があります。

必要なすべての pg_dump バイナリを補助マシンにインストールします。後で、 beacon_agent.yaml 構成ファイルの pg_bin_path 設定を使用して、各データベースエントリを正しいバイナリディレクトリをポイントできます。設定されると、その特定のデータベースのシステム PATH ルックアップをオーバーライドします。省略すると、エージェントは PATH で最初に見つかった pg_dump にフォールバックします。例

```
provider:
  onprem:
    databases:
      - resource_id: "pg16-community"
        dsn: $DSN_PG16
        pg_bin_path: /usr/lib/postgresql/16/bin
        schema:
          enabled: true
      - resource_id: "epas17"
        dsn: $DSN_EPAS17
        pg_bin_path: /usr/edb/as17/bin
        schema:
          enabled: true
```

資格情報と pgpass 動作を理解する

beacon-agent を実行する OS ユーザーは、pg_dump を実行するユーザーと同じです。プロセス引数でデータベース資格情報の公開を回避するため、エージェントは、制限的なアクセス許可 0600 でパスワードを一時 pgpass ファイルに書き込み、パスワードを含まない接続文字列を pg_dump に渡します。エージェントは OS 一時ディレクトリにファイルを作成し、各抽出が完了すると自動的に削除します。

> **注釈**
>
> エージェントプロセスが予期せず強制終了された場合たとえば *SIGKILL* 、エージェントは一時 pgpass ファイルをクリーンアップしない場合があります。エージェントを実行する OS ユーザーのみがファイルを読み取ることができます。 OS は、リブート時、またはエージェントがコンテナで実行されている場合はコンテナの再起動時にそれを削除します。孤立したファイルを確認するには、*/tmp* の `.pgpass-*` と一致するファイルを探します。

次のステップ

Preparing your AWS RDS/Aurora PostgreSQL source database

Preparing your Postgres source database

移行を開始する前に、Postgres ソースデータベースを構成し、必要なユーザーを作成します。何を構成する必要があるかは、目標によって異なります。

- EDB Postgres AI エージェントを使用した **スキーマ取り込みのみ**ハイブリッドマネージャー HM でマシンユーザーを作成し、補助マシンでエージェントの専用 OS ユーザーを設定し、スキーマ抽出権限を持つ移行ユーザーを作成します。WAL レベルのデータベースの変更は必要ありません。
- Data Migration ServiceDMS エージェントを使用した **データの移行* WAL 設定を構成し、必要なレプリケーション特権を付与します。

スキーマの取り込みとデータの移行の両方を実行する予定がある場合は、すべてのセクションを完了します。スキーマまたはデータの移行のみを行う予定がある場合、各セクションでユースケースに適用できるかどうかについて説明します。

移行ユーザーの作成

> **Tip**
>
> EDB Postgres AI エージェントを使用したスキーマの取り込みと DMS エージェントを使用したデータ移行の両方に必要です。

EDB Postgres AI エージェントと DMS エージェントの両方にサービスを提供する単一の移行ユーザーを作成します。1 つのユーザーを使用すると、エージェント全体の資格情報の管理が簡素化されます。

データベース管理アカウントを使用して、ソース Postgres データベースに接続し、移行ユーザーを作成します。

```
CREATE USER <migration_user> WITH REPLICATION LOGIN PASSWORD <migration_user_password>
↪;
```

REPLICATION は、DMS エージェントがレプリケーションスロットを作成し、WAL から読み取るために必要です。スキーマの取り込みのみを実行する場合は、省略できますが、ここで含めるということは、後でデータの移行を追加する場合にユーザーを変更する必要がないことを意味します。

スキーマ取り込み権限の付与

> **Tip**
>
> EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

1. EDB Postgres AI エージェントにソースデータベースへの接続アクセスを付与します。

```
GRANT CONNECT ON DATABASE <source_database> TO <migration_user>;
```

<source_database> は、評価するデータベースの名前です。エージェントを同じサーバー上の追加データベースに接続する予定がある場合は、各データベースに CONNECT を付与します。

1. 完全なスキーマ抽出に必要なロールを付与します。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_data TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

これら 3 つのロールを組み合わせると、所有権や特権情報を含む完全なスキーマの抽出が可能になります。pg_read_all_data のみが付与されている場合、エージェントは、所有権と特権情報を使用せずに、スキーマオブジェクト定義に制限された部分抽出を実行します。

pg_read_all_data は、Postgres 10〜13 では使用できません。

```
GRANT pg_monitor TO <migration_user>;
GRANT pg_read_all_settings TO <migration_user>;
```

1. Postgres 16 以前を使用している場合は、エージェントが Postgres ログファイルの場所を検出できるように次も付与します。

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_current_logfile() TO <migration_user>;
```

データ移行の権限の付与

> **Tip**
>
> DMS エージェントを使用したデータの移行に必要です。

1. 移行するテーブルを含む各スキーマで移行ユーザーの使用を付与します。

```
GRANT USAGE ON SCHEMA <db_schema> TO <migration_user>;
```

1. ソーステーブルに SELECT を付与します。スキーマ全体の場合

```
GRANT SELECT ON ALL TABLES IN SCHEMA <db_schema> TO <migration_user>;
```

個々のテーブルの場合

```
GRANT SELECT ON <table_name> TO <migration_user>;
```

1. DMS エージェントがパブリケーションを作成できるように、ソースデータベースに CREATE を付与します。

```
GRANT CREATE ON DATABASE <pg_db_name> TO <migration_user>;
```

スキーマ取り込みの準備 EDB Postgres AI エージェント

> **Tip**
>
> EDB Postgres AI エージェントでのスキーマの取り込みに必要です。

エージェントの専用 OS ユーザーの作成

beacon-agent を実行するための補助マシンに専用の OS ユーザーを作成します。この戦略により、必要のないエージェント権限の付与が回避されます。これは Linux 固有の例であり、正確なコマンドはディストリビューションによって異なる場合があります。

```
sudo useradd <agent_os_user>
```

拡張機能の移行の準備 EDB Postgres AI エージェント

> **Tip**
>
> ソースデータベースに拡張機能がインストールされている場合、 EDB Postgres AI エージェントを使用したスキーマの移行に必要です。

スキーマの移行の一部として、 EDB Postgres AI エージェントは、ソースデータベースにインストールされているすべての拡張機能を自動的に抽出し、他のスキーマ DDL が実行される前に宛先に適用します。拡張機能のインストールに失敗した場合、スキーマ全体の移行は中止されます。

スキーマの移行を実行する前に、ソースにインストールされている拡張機能を確認し、それらが同じバージョンの宛先で利用できることを確認します。一致するバージョンで拡張機能が利用できない場合、スキーマの移行は失敗します。異なるメジャー Postgres バージョンにまたがって移行する場合、移行先で利用可能な拡張機能バージョンはソースのものと一致しない場合があり、移行障害が発生する可能性があります。

```
SELECT name, installed_version
FROM pg_available_extensions
WHERE installed_version IS NOT NULL;
```

バージョンを比較する宛先で同じクエリーを実行します。

移行の完了後、ソースが外部データラッパー FDW 拡張機能を使用している場合、エージェントは FDW 拡張機能自分自身をインストールしますが、FDW オブジェクトは移行しません。宛先で外部サーバー、ユーザーマッピング、および外部テーブルを手動で再作成します。

拡張機能の移行の制限の完全なリストについては、拡張機能の移行の制限 を参照してください。

データ移行の準備 DMS エージェント

> **Tip**
>
> DMS エージェントを使用したデータ移行に必要です。

WAL 設定の構成

DMS エージェントは、Postgres 論理デコードを使用して、ログ先行書き込み WAL から変更をキャプチャします。`postgresql.conf` で次のパラメーターを構成します。

1. `wal_level` を `logical` に設定して論理デコードを有効にします。

```
wal_level = logical
```

1. `max_wal_senders` を少なくとも 1 に設定します。他のストリーミングクライアントがデータベースに接続する場合、適切な値について DBA に相談してください。

```
max_wal_senders = 1
```

1. max_replication_slots を少なくとも 1 に設定します。組織が Postgres レプリケーションを使用している場合、これは高くなる可能性があります。

```
max_replication_slots = 1
```

1. 実稼働トラフィックが WAL サイズに基づいて要求されたチェックポイントではなく、ほとんどの場合時間指定されたチェックポイントを生成するように、max_wal_size を十分に大きく設定します。少なくとも 8 GB をお勧めします。

```
max_wal_size = 8GB
```

max_wal_size の設定が小さすぎると、DMS エージェントがストリーミングできる前に Postgres が WAL エントリをドロップし、移行が妨げられるか、速度が低下する可能性があります。DBA に相談して、環境に適切なサイズを決定します。

1. データベースサーバーを再起動して、変更を適用します。

レプリケーショングループロールの作成

Postgres パブリケーションを自動作成するには、移行ユーザーがソーステーブルを所有している必要があります。これらのテーブルは既に別のロールが所有しているため、現在のテーブル所有者と移行ユーザーの両方が属するレプリケーショングループロールを作成します。

```
CREATE ROLE <replication_group>;
GRANT <replication_group> TO <migration_user>;
GRANT <replication_group> TO <original_owner>;
ALTER TABLE <table_name> OWNER TO <replication_group>;
```

そこで

- <migration_user> は、移行のために作成した Postgres ユーザーです。
- <original_owner> は、テーブルの元の所有者です。
- <replication_group> は、パブリケーション自動作成のソーステーブルを所有する共有ロールです。
- <table_name> は、移行するテーブルの名前です大文字と小文字が区別されます。

すべてのスキーマにわたって、移行する予定のテーブルごとに ALTER TABLE ... OWNER TO <replication_group> を実行する必要があります。データベース内のすべてのテーブルの所有権を一度に譲渡するには

```
DO $$
DECLARE
  s text;
  t text;
BEGIN
  FOR s, t IN
    SELECT <schemaname>, <tablename> FROM pg_tables
    WHERE <schemaname> NOT IN (pg_catalog, information_schema)
```

(次のページに続く)

(前のページからの続き)

```
  LOOP
    EXECUTE ALTER TABLE  || quote_ident(s) || . || quote_ident(t) ||  OWNER TO
↪<replication_group>;
  END LOOP;
END;
$$;
```

SSL の構成

ソースデータベースサーバーを accept SSL connections に構成します。DMS エージェントは、ソースに接続するために SSL が必要です。サーバー証明書とサーバープライベートキーを作成します。OpenSSL のようなツールでプライベートキーは生成できます。

構成の検証

DMS エージェントには、ソースデータベースが CDC 移行のすべての要件を満たしているかどうかを確認する検証スクリプトが含まれています。

1. DMS エージェントフォルダーに移動します。

```
cd /opt/cdcagent/reader/
```

1. 移行するすべてのテーブルの配列を<schema_name>.<table_name> 形式で作成します。

```
arr=(schema1.table1 schema1.table2 schema2.table1)
```

1. 検証スクリプトを実行します。

```
PG_USERNAME=postgres PG_PASSWORD=<password> PG_HOST=<host> PG_PORT=5432 DB_NAME=<db_
↪name> DBZ_USERNAME=<migration_user> DBZ_PASSWORD=<migration_user_password> ./
↪postgresConfigValidation.sh "${arr[@]}"
```

合格結果は次のようになります。

```
  ***  [Transporter] - Get PostgreSQL Major Version
PostgreSQL major version: 17
  ***  [Transporter] - Validate WAL Level
wal_level:  logical
[Pass] wal_level is logical.
  ***  [Transporter] - Validate max WAL senders
max_wal_senders: 10
[Pass] max_wal_senders is at least 1.
  ***  [Transporter] - Validate max replication slots
max_replication_slots: 10
[Pass] max_replication_slots is at least 1.
  ***  [Transporter] - Validate max WAL size
max_wal_size: 8 GB
```

(次のページに続く)

(前のページからの続き)

```
[Pass] max_wal_size is set to the recommended value of 8GB or higher.
  ***  [Transporter] - Validate checkpoints
checkpoints_timed: 100
checkpoints_req: 5
[Pass] Timed checkpoints are more frequent than requested checkpoints.
  ***  [Transporter] - Check <migration_user> user role
[Pass] User <migration_user> is present
[Pass] User <migration_user> has replication permission
  ***  [Transporter] - Check SELECT privilege on the tables to be migrated
[Pass] User <migration_user> has select privilege on all tables to be migrated.
  ***  [Transporter] - Check CREATE privilege on database
[Pass] Role <migration_user> has CREATE privilege on database <db_name>.
```

注釈

続行する前に`[Failed]` または`[Suggestion]` ステータスに対処します。 *[Failed]* チェックは、DMS エージェントがデータ移行を実行する前に解決が必要な問題をブロックしています。 *[Suggestion]* チェックでは移行が許可されますが、対処しないでおくとパフォーマンスに悪影響を与える可能性があります。

次のステップ

Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database

Configuring the EDB Postgres AI agent to register your Postgres database

EDB Postgres AI エージェント `beacon-agent` としてパッケージ化して、HM との安全な接続を確立し、Postgres ソースからスキーマと移行評価データを取得します。

前提条件

- **マシンユーザーとアクセスキー** *Creating a machine user* で作成されます。アクセスキーを環境変数に保存します。

```
export BEACON_AGENT_ACCESS_KEY="baak_your_key_here"
```

- **pg_dump** EDB Postgres AI エージェントは pg_dump を使用して Postgres ソースから DDL を抽出します。 pg_dump が補助マシンにインストールおよび構成されていることを確認します。インストール手順、互換性ルール、および資格情報のセットアップについては、*Configuring your auxiliary machine or bastion host* を参照してください。

注釈

エージェントは、実行時に`beacon_agent.yaml` で参照される環境変数を置き換えるため、エージェントを実行する前に環境で `$BEACON_AGENT_ACCESS_KEY` を設定するだけで十分です。環境変数は、YAML の静的値よりも常に優先されます。

環境変数を設定するための推奨事項

$BEACON_AGENT_ACCESS_KEY や$DSN などの機密データは、シークレットマネージャーで管理される環境変数として扱うことをお勧めします。この戦略により、機密の資格情報がプレーンテキストの構成ファイルに保存されるのを防ぎます。環境変数を使用する場合、EDB Postgres AI エージェントの実行中にそれらが利用できることを確認してください。

実稼働環境のための最も安全なアプローチについて、組織の IT 部門に相談してください。

短期間の移行またはテストとデモの場合、アクティブなターミナルセッションで export コマンドを使用できます。

HM 接続パラメーターの取得

エージェントには、HM インストールからの特定の値が必要です。次のことについては、管理者に連絡してください。

- **ビーコンサーバーホスト名** beacon_server の内部エンドポイント。

HM インストールを構成するときに、管理者は beacon_server サービスの内部エンドポイントを指定する必要があります。HM インスタンスの管理者またはインストーラーに、インストールに使用される Helm チャート構成ファイルで beacon_server サービスに設定したホスト名 URL を提供するように依頼します。values.yaml ファイルの parameters.upm-beacon.server_host または BEACON_SERVICE_DOMAIN_NAME に設定した値を検索します。この値は、後で EDB Postgres AI エージェントを構成するために必要になります。

- **ルート証明書パス必要な場合** HM インスタンスが自己署名または内部 CA を使用する場合、ca_public.crt ファイルを取得します。エージェントマシンたとえば/etc/beacon/ca_public.crt に保存します。

EDB Postgres AI エージェントには、HM インスタンスの beacon_server サービスへの信頼できる接続が必要です。 EDB Postgres AI エージェントを実行しているサーバーが HM の TLS 証明書組織のセキュリティインフラストラクチャによって管理されているを本質的に信頼していない場合、HM インスタンスの管理者またはインストーラーに証明書を含むファイルを提供するように依頼してください。後でこの証明書へのディレクトリパスを提供して、EDB Postgres AI エージェントを構成する必要があります。

beacon_server およびルート証明書の詳細については、*Administrative tasks* を参照してください。

接続の検証

HM 接続パラメーターを取得したら、補助マシンが Postgres ソースと HM の両方に到達できることを確認します。次のポートを開く必要があります。

Destination	Port	Used by
Postgres host	5432 (default for PostgreSQL, AWS RDS, and Aurora); 5444 (default for EDB Postgres Advanced Server)	Both agents (database connection)
HM *beacon_server*	9443	EDB Postgres AI agent

次のページに続く

表 39 – 前のページからの続き

Destination	Port	Used by
HM DMS service (*RW_SERVICE_HOST*)	443 (HTTPS)	DMS agent

注釈

ポート 5432 および 5444 はデフォルトです。 Postgres データベースが別のポートでリッスンするように構成されている場合、代わりにそのポートを使用し、ネットワークでオープンしていることを確認します。

nc を使用して、各ホストとポートへの接続を確認します。

```
#  Postgres source

nc -zv <postgres-host-name-or-ip> 5432

#  HM beacon server

nc -zv <beacon_server_hostname> 9443

#  HM DMS service

nc -zv <rw-service-hostname> 443
```

接続が成功すると succeeded! が返されます。 Connection refused または Operation timed out が表示される場合、ポートは到達不能であるため、続行する前に開く必要があります。

ソースデータベース接続パラメーターの定義

データソース名 DSN は、Postgres ソースの接続文字列です。

フォーマット

```
DSN="postgresql://<migration_user>:<migration_user_password>@<host>:<port>/<database_
↪name>"
```

エージェントを実行またはサービスを構成する前に、DSN を環境変数として設定します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb"
```

データベースの SSL が無効になっている場合、 ?sslmode=disable を追加します。

```
export DSN="postgresql://migration_user:password@postgres-host:5432/mydb?
↪sslmode=disable"
```

RDS インスタンスエンドポイントをホストとして使用します。RDS はデフォルトで SSL を適用するため、?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@myinstance.abc123.us-east-1.rds.
↪amazonaws.com:5432/mydb?sslmode=require"
```

Aurora クラスターエンドポイントをホストとして使用します。Aurora はデフォルトで SSL を適用するため、 ?sslmode=require を追加します。

```
export DSN="postgresql://migration_user:password@my-cluster.cluster-abc123.us-east-1.
↪rds.amazonaws.com:5432/mydb?sslmode=require"
```

```
DSN="postgresql://migration_user:pa%24%24word@postgres-host:5432/mydb"
```

構成ファイルの準備

1. ホームディレクトリに EDB Postgres AI エージェント構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

> **注釈**
>
> エージェントは `/etc/beacon/` または `${HOME}/.beacon/` で `beacon_agent.yaml` を検索します。最終的なフォールバックとして、実行されたディレクトリを検索します。 !!!

1. このディレクトリ内に、ファイル beacon_agent.yaml を作成します。

```
touch ${HOME}/.beacon/beacon_agent.yaml
```

1. 次のテンプレートをコピーして、新しいファイルに貼り付けます。

このテンプレートでは、 $BEACON_AGENT_ACCESS_KEY および$DSN は、前に構成した環境変数です。テンプレート内のすべての< >プレースホルダーを置き換えます。詳細は、*EDB DMS agent parameter reference* を参照してください。

```
---
agent:
  access_key: $BEACON_AGENT_ACCESS_KEY
  beacon_server: <beacon_server_hostname>:9443
  project_id: <your_project_id>
  providers:
    - "onprem"
  # In most HM deployments, set `root_ca_path` to the path of the ca_public.crt
  # file obtained from your HM administrator (for example, /etc/beacon/ca_public.crt).
  # Leave empty only if HMs certificate is already trusted by the systems CA bundle.
  root_ca_path: ""
  schema_providers:
```

(次のページに続く)

(前のページからの続き)

```
    # Include "onprem-schema" to enable schema ingestion and migration assessment.
    # Remove this section if you dont want schema information to be collected.
    - "onprem-schema"
provider:
  onprem:
    databases:
      # `resource_id` controls how this database appears in HM Estate.
      - resource_id: "<database_resource_id>"
        dsn: $DSN
        # pg_bin_path: /usr/lib/postgresql/16/bin
        # Optional. Absolute path to the directory containing pg_dump and pg_dumpall.
        # Set when registering databases from different Postgres distributions/
↪versions.
        schema:
          enabled: true
          poll_interval: 15s
general:
  logging:
    # `info` is simplified, use debug during initial setup or
    # troubleshooting for more verbose output.
    level: info
```

1. ファイルを保存します。

EDB Postgres AI エージェントの実行

1. エージェントを起動する前に、接続と構成構文を確認します。

```
beacon-agent validate
```

1. エージェントを実行します。

```
beacon-agent
```

Tip

他のエージェントモードとオプションについては、 Agent CLI を参照してください。 !!!

1. エージェントが HM へのデータの送信を開始するときに、取り込みプロセスを監視します。数分後、Postgres データベースが HM コンソールの **Estate > Migrations** の下に表示されます。

重要

エージェントは、テーブルと列の変更を動的に検出します。ただし、ビュー、プロシージャー、またはファンクションを変更するには、エージェントを再起動してメタデータを再取り込む必要があります。

本番使用の場合は、 *Running the agent as a service* を参照してください。

移行評価の検証

Postgres データベースがエージェントに認識され、移行の準備ができていることを確認するには、HM コンソールで **Estate > Migrations** に移動します。

データベースがテーブルに表示される場合、次のリソースにアクセスできます。

- **Estate** ページの **マイグレーション** の **移行評価**。データベーススキーマ数、コア数、移行の複雑さ、および作業レベルの概要を提供します。
- **Migrations** タブから特定のデータベースを選択することによりアクセスできる *Viewing migration assessment details* 。このタブは、**スキーマ** の拡張分析、グローバル **評価**、および移行を実行するための他の HM ツールに直接接続する **移行**ハブを提供します。
- ビルトインの移行ポータルでのスキーマの移行に使用できる**自動スキーマ DDL 抽出**。取り込みが完了すると、移行ポータルはデータベースごとにプロジェクトを自動的に作成します。これを使用して、スキーマを準備し、宛先データベースに移行できます。

障害対応

エージェントが HM または Postgres データベースへの接続に失敗する理由はいくつかあります。ここでは、トラブルシューティングに使用できるツールをいくつか紹介します。

`beacon-agent validate` コマンド

`beacon-agent validate` コマンドは、一般的な接続と構成の問題を特定するための役立つ手順です。

```
beacon-agent validate
```

validate コマンドは、次のことをチェックします。

- **サーバー接続** エージェントが構成済みの `beacon_server` 通常はポート `9443` に接続および認証できることを確認します。
- **データベース接続**構成されている各データベース DSN への接続を試行します。これは完全な接続テストであり、単なる構文または URL エンコーディングのチェックではありません。
- **構成表示** TLS 設定、プロジェクト ID、プロバイダー、およびデータベースごとの設定を表示し、構成の誤りを特定するのに役立ちます。

`beacon-agent` ログ

エージェントが取り込み中にエラーを報告した場合、詳細についてはログを確認できます。フォアグラウンドでエージェントを実行している場合、ログはターミナルに出力されます。エージェントを systemd サービスとして設定する場合、`journalctl` を使用してログを表示します。

次に、ログに表示される一般的なエラーメッセージをいくつか示します。

Error	Cause	Fix
tls: failed to verify certificate: x509: certificate signed by unknown authority	The agent can't verify HM's TLS certificate because the CA isn't trusted by the system.	Set *root_ca_path* in *beacon_agent.yaml* to the path of the *ca_public.crt* file obtained from your HM administrator. If you can't obtain the certificate, you can set *plaintext: true* in the agent config as a temporary workaround (not recommended for production).
rpc error: code = Unknown desc = no rows in result set	The beacon-agent version is incompatible with the server.	Upgrade to beacon-agent v1.384.0 or later.

エージェントをサービスとして実行する

継続的なメトリックストリーミングのために、また Postgres データベースが **HM コンソール>エステート**に表示されたままであることを確認するには、systemd サービスとして実行するようにエージェントを構成します。サービスとして実行すると、システムの起動時に自動的に起動し、エラーが発生した後に再起動します。

サービスがないと、エージェントは手動で実行している間のみメトリックをストリーミングします。データベースは、エージェントがストリーミングを停止してから 72 時間後に HM 不動産から消えます。

> **注釈**
>
> *beacon_agent.yaml* たとえば *$BEACON_AGENT_ACCESS_KEY* や `$DSN` などで参照される環境変数が、実行時にサービスにアクセスできることを確認します。スキーマ抽出資格情報の `.pgpass` ファイルを構成した場合 **環境の構成** で説明しているように、そのファイルが `User=` で指定されたユーザーのホームディレクトリに存在することを確認します。

1. サービスファイルを作成します。この例では `vi` を使用しますが、任意のテキストエディタを使用できます。

```
sudo vi /usr/lib/systemd/system/beacon-agent.service
```

オペレーティングシステム別のデフォルトのサービスファイルパス

Operating system	Default path
RHEL	*/usr/lib/systemd/system/beacon-agent.service*
Rocky Linux	*/usr/lib/systemd/system/beacon-agent.service*
AlmaLinux	*/usr/lib/systemd/system/beacon-agent.service*
Ubuntu	*/etc/systemd/system/beacon-agent.service*
Debian	*/lib/systemd/system/beacon-agent.service*

1. 次のテンプレートをファイルに貼り付けて、すべての< >プレースホルダーを置き換えます。

```
[Unit]
Description=EDB Postgres AI Agent
After=network.target

[Service]
Type=simple
User=<agent_os_user>
WorkingDirectory=</home/agent_os_user/>
ExecStart=</usr/local/bin/beacon-agent>
Restart=on-failure
RestartSec=60

[Install]
WantedBy=multi-user.target
```

Setting	Placeholder	Description
User=	*<agent_os_user>*	エージェントを実行する OS ユーザー。ソースデータベースの準備時に作成した専用の OS ユーザーを使用します。
WorkingDirectory=	*</home/agent_os_user/>*	`.beacon`フォルダーが保存されるディレクトリ。これは通常、`<agent_os_user>`のホームディレクトリです。
ExecStart=	*</usr/local/bin/beacon-agent>*	`beacon-agent`バイナリへのフルパス。パスは、エージェントがインストールされた方法と場所によって異なります。

1. systemd をリロードし、サービスを有効にして開始します。

```
sudo systemctl daemon-reload
sudo systemctl enable beacon-agent.service
sudo systemctl start beacon-agent.service
```

1. ログを確認して、サービスが正常に開始されたことを確認します。

```
journalctl -u beacon-agent.service
```

関連トピック

- スキーマ取り込みのより詳細な制御たとえば、スキーマを除外または含める場合、*Controlling schema ingestion* を参照してください。
- 構成パラメーターの完全なセットについては、*EDB DMS agent parameter reference* を参照してください。
- 管理者がルート証明書または `beacon_server` 値の取得に関するガイダンスが必要な場合は、*Administrative tasks* を参照してください。

次のステップ

Preparing your HM-managed destination database

Preparing your HM-managed destination database

移行の宛先として機能する HM 管理データベースクラスターを準備します。

ハイブリッドマネージャー HM に宛先データベースクラスターがまだない場合は、作成します。

1. HM コンソールで、データベースクラスターを作成する "プロジェクトを選択できません" — ポータルドメインはデフォルトで API エンドポイントになります を選択します。
2. プロジェクトで、 **Create New** を選択し、 **Database Cluster** から *Create a database cluster template* を選択します。
3. **Clusters** ページで、クラスターを選択し、 **Quick Connect** オプションを使用してターミナルからデータベースにアクセスします。
4. 移行の宛先として使用する空のデータベースを作成します。または、デフォルトのデータベース `edb_admin` を使用できます。

すでに HM 管理のデータベースクラスターがある場合は、それを移行の宛先として使用できます。

続行する前に、移行するテーブルと移行中に生成される負荷がクラスター上の既存のワークロードを干渉しないことを確認してください。

次のステップ

Migrating Postgres schemas

Migrating Postgres schemas

データの移行を開始する前に、スキーマの移行を処理する方法を選択します。HM は 2 つのアプローチをサポートしています。

- **簡素化推奨** HM は、データ移行セットアップの一部としてスキーマの移行と制約処理を自動的に管理します。
- **手動** Migration Toolkit または `pg_dump` を使用して自分自身でスキーマをエクスポートおよびインポートし、各ソーステーブルのキー構造に基づいて制約を処理します。

手動でのスキーマの準備は必要ありません。 HM は、後で **スキーマとデータ**スコープを選択すると、スキーマの移行と制約の順序付けを自動的に処理します。続行する前に、次を確認します。

- EDB Postgres AI エージェントはソーススキーマを取り込み、データベースには移行ポータルプロジェクトが存在します。確認するには、HM コンソールで **Estate > Migrations** に移動し、**Database Name** 列でデータベースを選択し、 **MP Project** 列にプロジェクトが表示されることを確認します。
- 後の手順で DMS リーダーエージェントを構成するときに、`DBCONFIG_DATABASES_<N>__RESOURCEID` を `beacon_agent.yaml` と同じ `resource_id` 値に設定します。 **スキーマとデータ** および**スキーマのみ** スコープは、これがないと失敗します。

次のステップに進みます。

データの移行に EDB DMS を使用する前に、スキーマを宛先データベースにエクスポートおよびインポートします。DMS はデータのみを移行します。移行を開始する前に、宛先データベースにスキーマを設定する必要があります。

コンストレインの処理

制約を処理する方法は、制約タイプと各ソーステーブルのキー構造によって異なります。

Constraint type	Action	Notes
Primary key	今すぐ除外する — DMS は自動的に適用されます	The DMS writer applies the primary key when the migration enters the streaming phase, using it to deduplicate snapshot data.
Unique key (source table has a primary key)	今すぐ除外、ストリーミング開始後に適用	Primary key-based deduplication ensures no duplicate rows remain when you apply unique constraints after streaming starts.
Unique key (source table has no primary key)	現在サポートされていません	Use the simplified path instead — select the **Schema and data** scope when creating the migration to let HM handle schema migration automatically.
NOT NULL	今すぐ含める	*NOT NULL* validation adds no meaningful overhead to the data load, unlike foreign key and check constraint validation.
Foreign key, check, and exclusion	今すぐ除外、移行後に再適用	These cause unnecessary CPU and memory overhead while data is actively being migrated.

注釈

主キーまたは一意キーのないテーブルの場合、移行の前後にキー制約は必要ありません。 DMS は、これらのテーブルをキーレスとして扱い、少なくとも 1 回の配信を提供します。移行後の検証の一部として、行の重複を削除します。

スキーマのエクスポート

EDB Migration Toolkit を使用してスキーマをエクスポートすることをお勧めします。MTK の offline migration 機能は、直接接続せずに宛先に適用できる SQL スクリプトを生成します。他の有効なエクスポート方法には、`pg_dump` /`pg_restore` 、pgAdmin、または `psql` などがあります。

Migration Toolkit を使用する場合

- `-offlineMigration` オプションを使用して SQL スクリプトを生成します。
- `-schemaOnly <schema_scope>` オプションを使用して、スキーマのみをエクスポートします。
- インポートする前に、生成されたスクリプトから主キー、一意、外部キー、チェック、および除外コンストレインを除外コメントアウトします。
- オブジェクトを宛先の別の名前にマップする予定がある場合は、インポートする前に DDL スクリプトで直接編集します。名前が変更されたオブジェクトを参照するすべての依存オブジェクトビュー、

コンストレイン、およびその他の参照を更新します。詳細は、*Mapping schema, tables, and columns* を参照してください。

スキーマのインポート

DDL を準備し、遅延制約を除外した後、宛先データベースに接続し、生成された offline migration script を実行して SQL ファイルを適用します。

> **注釈**
>
> 移行が完了した後、除外された一意の外部キー、チェック、および除外制約を適用します。 DMS ライタは、主キーコンストレインを自動的に適用します。手動で再適用しないでください。除外されたコンストレインを追跡して、必要なときに再適用できます。

次のステップ

Configuring and running the EDB DMS agent

Configuring and running the EDB DMS agent

EDB DMS エージェントは Postgres ソースに接続し、Data Migration ServiceDMS と連携してデータ移行を実行します。実行する前に、HM で移行資格情報を作成し、ソースデータベースの詳細でエージェントを構成します。

前提条件

- DMS エージェントを実行しているマシンに Java / OpenJDK 17 以降がインストールされていることを確認します。
- *Install the EDB DMS agent* 。

移行資格情報の作成

移行資格情報は、DMS エージェントをハイブリッドマネージャー HM に対して認証します。HM プロジェクトごとに 1 セットの資格情報が必要です。

1. HM コンソールで、移行に使用しているプロジェクトを選択します。
2. **移行>資格情報** を選択します。
3. **Create Migration Credential** を選択し、名前を入力して、 **Create Migration Credential** を選択します。 `credentials.zip` ファイルがマシンにダウンロードされます。

> **注釈**
>
> 資格情報は安全に保存します。それらを失った場合は、このセットを取り消し、新しいセットを作成する必要があります。 !!!

1. ダウンロードしたファイルを解凍し、資格情報フォルダーを DMS エージェントをインストールしたマシンたとえば$HOME/credentials/ にコピーします。別のパスを使用する場合は、スクリプト内の CREDENTIAL_DIRECTORY_PATH を一致するように更新します。

フォルダーには次のものが含まれています

DMS エージェントの構成

1. /opt/cdcagent/run-cdcagent.sh を書き込み権限で開きます。
2. 必要な変数を設定します。オプションのパラメーターはデフォルトでコメントアウトされています。環境に適用されるものはコメントを外して構成します。

詳細は、 *EDB DMS agent parameter reference* を参照してください。

```
#!/bin/bash -e
# run-cdcagent.sh
#
# This script provides a convenient place to specify
# environment variables used to configure the
# EDB Data Migration Service Agent.
#

##########################################
# DMS Agent General Configuration       #
##########################################

# This ID is used to identify DMS Agent
# and is specified by the user.
export DBCONFIG_ID=<source_name_of_choice>

# Determine the run mode of the DMS Agent
# The run mode can be set to either reader or writer
# reader mode is used to read changes from a source database
# writer mode is used to write changes to a target database
export RUN_MODE=reader

# This is the DMS backend service used by the Agent
# Obtain this value from the HM console Sources or Destinations migration pages
export RW_SERVICE_HOST=<rw_service_host>

##########################################
# DMS Agent TLS Configuration           #
##########################################

# You need to create migration credentials in EDB Postgres AI platform
# and set the path of the credentials directory
export CREDENTIAL_DIRECTORY_PATH=$HOME/credentials
```

(次のページに続く)

(前のページからの続き)

```
########################################
# DMS Agent DB Configuration           #
########################################

# A sample configuration to create a single Postgres database connection:
export DBCONFIG_DATABASES_0__TYPE=POSTGRES
export DBCONFIG_DATABASES_0__HOSTNAME=<localhost/host>
export DBCONFIG_DATABASES_0__PORT=<5432>
# The CATALOG is the database name
export DBCONFIG_DATABASES_0__CATALOG=<source>
export DBCONFIG_DATABASES_0__USERNAME=<postgres>
# The password env can be set without specifying it here
# but the env structure looks like this
export DBCONFIG_DATABASES_0__PASSWORD=<password>
# Required for Schema and data or Schema only scope.
# Must match resource_id in beacon_agent.yaml.
#export DBCONFIG_DATABASES_0__RESOURCEID=<resource_id>

# You can increase the index to configure more than
# one database for the DMS Agent
#export DBCONFIG_DATABASES_1__TYPE=POSTGRES
#export DBCONFIG_DATABASES_1__HOSTNAME=<localhost/host>
#export DBCONFIG_DATABASES_1__PORT=<5432>
#export DBCONFIG_DATABASES_1__CATALOG=<source>
#export DBCONFIG_DATABASES_1__USERNAME=<postgres>
#export DBCONFIG_DATABASES_1__PASSWORD=<password>
#export DBCONFIG_DATABASES_1__RESOURCEID=<resource_id>

########################################
# Optional Parameters Below            #
########################################

# FIPS mode can be enabled by setting the FIPS_MODE env variable to true
#export FIPS_MODE=true

# Configure logging
# Loglevel for the entire Quarkus application and underlying components
# WARNING: Enabling this loglevel is extremely verbose
#export QUARKUS_LOG_LEVEL=DEBUG
# Loglevel for the EDB package
# Preferred mode for debugging issues related to the logic of the agent
#export QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL=DEBUG

cd "$(dirname "$0")"
```

(次のページに続く)

(前のページからの続き)

```
echo "Starting in mode: $RUN_MODE"

case "$RUN_MODE" in
  reader)
    java -jar ./reader/quarkus-run.jar
    ;;
  writer)
    java -jar ./writer/quarkus-run.jar
    ;;
  *)
    echo "Unknown RUN_MODE: $RUN_MODE"
    exit 1
    ;;
esac
```

DMS エージェントの実行

DMS エージェントディレクトリに移動し、構成スクリプトを実行します。

```
cd /opt/cdcagent
./run-cdcagent.sh
```

スクリプトは、必要なすべての環境変数をエクスポートし、エージェントプロセスを開始します。エージェントが実行されると、Postgres データベースを HM のソースとして登録します。次のセクションに進んで、正しく表示されることを確認します。

HM でのソースの検証

エージェントが実行されたら、Postgres データベースが HM のソースとして表示されることを確認します。

1. HM コンソールで、プロジェクトを選択します。
2. **移行>ソース** を選択します。

データベースは、名前、タイプ、エージェント ID、およびステータスとともにリストされます。このページのすべてのフィールドの説明については、**デプロイメントリソースを理解する** を参照してください。

次のステップ

Migrating data from an AWS RDS or Aurora PostgreSQL database to an HM-managed database

Migrating data from a self-managed Postgres database to an HM-managed database

EDB DMS エージェントが実行され、スキーマがインポートされるか、HM インポートの準備が整うと、HM コンソールで移行を作成する準備が整います。このページでは、完全な移行ライフサイクルについて説明します。移行の作成、その進行状況の監視、ファイナライズ、および移行後の制約の適用。

以下の手順は、前のページで選択したアプローチによって異なります。選択したものと一致するタブを選択します。

移行の作成

移行は、Postgres ソースと HM によって管理される宛先データベースクラスター間の同期を確立します。

1. HM コンソールで、移行を管理するプロジェクトを選択します。
2. プロジェクトで、 **移行>移行** を選択します。
3. [**移行**] ページで、[**Create New Migration**] を選択します。
4. **Create OLTP Migration** ページで、名前を移行に割り当てます。
5. 移行のソースを選択します。 DMS エージェントの ID は、 **Source** メニューにリストされています。
6. **Scope** の下で、アプローチに一致するオプションを選択します。

スキーマとデータ を選択します。このオプションを選択すると、 HM:

1. データベースを EDB Postgres AI エージェントに登録したときに自動的に作成された移行ポータルプロジェクトからスキーマをソースします。
2. 制約なしで宛先にテーブルを作成するため、初期スナップショットコピーは作成されません制約の検証により遅くなります。
3. スナップショットが完了したら、主キーと一意の制約があるテーブルの場合を適用して、DMS がストリーミング変更を 1 回だけ配信できます。PK または UK のないテーブルは少なくとも 1 回の配信にフォールバックし、検証中に重複排除が必要です。
4. カットオーバーまで残りのスキーマインデックス、外部キー、チェック制約、`NOT NULL` などを遅延します。オプションで、この作業のほとんどを**ファイナライズスキーマコンカレントリー**を介してストリーミングと同時に適用して、カットオーバーウィンドウを縮小できます。 ストリーミング移行の仕上げ を参照してください。
5. 移行を完了とマークした後、遅延外部キーを検証し、シーケンスを同期し、残りの軽量ファイルを適用します最終の**スキーマファイナライズ** ステージのタスク。

> **注釈**
>
> テーブルデータを移行せずにデータベース構造を移行する場合は、 **スキーマのみ**を使用します。!!!

データのみを選択します。スキーマは、前の手順で既に手動でインポートされています。

1. スナップショットタイプを選択します ***Snapshot*** または ***Snapshot + Streaming*** 。
2. **Destination Type** で、移行用の HM 管理の宛先データベースクラスターとデータベースを選択します。次に、**Next** を選択します。
3. **Review your Tables** タブで、移行に含めるテーブルと列を選択します。テーブルを展開して、その列とオプションで *map source columns to different destination column names* を表示します。列マッピングは、 **データのみ**スコープでのみ使用できます。

4. **Create Migration** を選択します。

HM コンソールには、*new migration* が表示されます。**スナップショット+ストリーミング** を選択した場合、移行が **実行中**状態を示している間、DMS エージェントは継続的にデータ変更をストリーミングします。

> **注釈**
>
> DMS エージェントは、データ変更のみをストリーミングします。DDL 変更 DDL 変更はストリーミングしません。

進行状況のモニタリング

移行リストビュー

Migrations ページには、プロジェクト内のすべての移行の概要が表示されます。**移行の進行状況** 列には、移行ごとに 3 つのメトリックが表示されます。

- **テーブルの合計数** — この移行に含まれるテーブルの合計
- **スナップショットテーブルが完了しました** —スナップショットステージが終了したテーブル
- **移行された行** — すべてのテーブルで移行された累積行数

移行の詳細ビュー

リストから移行を選択して、3 つのタブがある詳細ページを開きます。

- **概要** - 現在の移行状態と、ソースおよび宛先データベースに関する一般的な情報を表示します。**スナップショット+ストリーミング** の移行の場合、ステータスが**ストリーミング** のときに移行はキャッチアップされます。**スナップショット** の移行の場合、ステータスが **Completed** になると完了します。
- **Tables** —移行のために選択されたすべてのテーブルを表示します。テーブルごとに、ソースおよび宛先テーブル、スキーマ名、スナップショットステータス、マップされた列、移行された行、および移行速度を確認できます。**+** アイコンを選択してテーブルを展開し、列レベルのマッピングを表示します。
- **Error History** —この移行についてログに記録されたすべてのエラーをリストします。 **i** アイコンを選択して、特定のエラーの完全なエラーログを開きます。移行が停止または失敗した場合、ここでのアクティブなエラーは、トラブルシューティングを開始する場所を示します。

DMS エージェントログ

HM コンソールは、すべての DMS エージェントエラーを表示するわけではありません。移行がスタックしているか、進行中のテーブルがない場合は、`run-cdcagent.sh` を実行したターミナルで `ERROR` ログエントリを確認します。多くの場合、これらのエントリには、根本的な原因が含まれています。一般的な問題には、DMS エージェントと HM 間のバージョンの非互換性や接続または資格情報の障害が含まれます。

ストリーミング移行の仕上げ

> **注釈**
>
> このセクションは、**スナップショット+ストリーミング**の移行にのみ適用されます。 **スナップショット**の移行の場合、DMS は移行を自動的に完了としてマークします。

カットオーバー前のキャッチアップのモニタリング

ソースへの書き込みを停止する前に、宛先に追いつくまでの時間を与えます。**移行の詳細**ページを使用して、行数とレプリケーションラグを追跡します。デスティネーションが一貫してソースに近づいた後にのみ、カットオーバーを開始します。

スキーマのファイナライズを同時に行うことをお勧めします

Schema および data スコープを使用した場合、HM はストリームの実行中にほとんどのスキーマオブジェクトを適用できます。プライマリキー、一意制約、インデックス `CREATE INDEX CONCURRENTLY` でビルドされるため、ライブ書き込みがブロックされません、外部キー (`NOT VALID`)、チェックコンストレイン、`NOT NULL` コンストレインなどとして追加されます。外部キー検証、シーケンス同期、およびその他の軽量タスクは、カットオーバー後の**スキーマファイナライズ** ステージで実行されます。

この作業を実行すると、読み取り専用のカットオーバーウィンドウが同時に縮小されます。テーブルが大きくなり、インデックスの数が増えるほど、節約される時間は長くなります。

すべてのテーブルがスナップショットを完了し、移行がストリーミングフェーズに入ると、**移行の詳細** ページに、 **ファイナライズスキーマコンカレントリー**ボタンを備えた **スキーマファイナライズ可能**アラートが表示されます。ボタンはストリーミング中は利用可能なままであるため、目的地が追いつくまでにどれだけ近いかに基づいて、瞬間を選択できます。

1. **Migration Details** ページで **Finalize Schema Concurrently** を選択します。
2. HM は、ストリーミングが継続されている間、遅延オブジェクトを適用します。移行は**実行中** 状態のままで、進行状況サブステップには**ファイナライズスキーマコンカレントリー**というラベルが付いています。

サブステップが完了したら、カットオーバーに進みます。

> **注釈**
>
> **ファイナライズスキーマコンカレントリー**は、ストリーミングが有効になっている**スキーマおよびデータ**スコープで作成された移行でのみ使用できます。

この手順をスキップすると、移行を完了としてマークした後、同じオブジェクトが連続して適用されます。スキーマの最終化ステージを参照してください。この場合、インデックスは非並列で構築され、制約は排他的ロックの下で検証されるため、カットオーバーウィンドウは大きくなります。

手動パスには適用されません。移行前にスキーマを適用し、カットオーバー後に遅延コンストレインを手動で再適用します。遅延コンストレインの適用とデータの検証 を参照してください。

カットオーバーの実行

トラフィックをソースから宛先にスイッチする準備ができたら

1. ソースデータベースのすべての書き込みを停止します。
2. HM コンソールで、デスティネーションが最後のソース変更を適用したことを確認します。
3. **Migration Details** ページで、 **Mark as Completed** を選択します。

これにより、ストリーミングプロセスが停止します。

スキーマの最終化ステージ

スキーマおよびデータ スコープで作成された移行の場合、移行は **Mark as Completed** の後、自動的に **スキーマファイナライズ**ステージに入ります。このステージで実行されるものは、最初にコンカレントファイナライズを実行したかどうかによって異なります。

- **同時ファイナライズを実行した場合** HM は、軽量の残りの作業外部キー検証、シーケンス同期、およびその他の遅延メタデータタスクを実行します。このステージは短いです。
- **同時ファイナライズをスキップした場合** HM は、完全な遅延セットプライマリキー、一意制約、インデックス、外部キー、チェック制約、`NOT NULL` などを適用した後、上記の軽量タスクを適用します。このステージにはかなりの時間がかかり、ソースが読み取り専用であるときに実行されます。

移行は、 **スキーマのファイナライズ**ステージが完了した後にのみ完全に完了します。進行状況は、 **移行の詳細**ページに表示されます。

データのみスコープでは、自動スキーマファイナライズは実行されません。遅延コンストレインの適用とデータの検証 に進み、移行前に延期した外部キー、一意、チェック、および除外制約を再適用します。

遅延コンストレインの適用とデータの検証

遅延コンストレインの再適用

HM は、 **スキーマファイナライズ**ステージ中に、すべての遅延コンストレインと同期シーケンスを自動的に適用しました。手動操作は必要ありません。スキーマの最終化ステージ を参照してください。

移行の初めに、外部キー、一意、チェック、および除外制約を除いたスキーマを準備およびインポートしました。宛先データベースに接続し、これらのコンストレインを再適用します。

主キー制約は、移行がストリーミングフェーズに入ったときに、DMS ライタによって自動的に適用されました。`NOT NULL` コンストレインは移行前に適用され、現在アクションは必要ありません。

必要に応じて行の重複を削除する

主キーも一意制約もなかったテーブルの場合、EDB DMS エージェントは少なくとも 1 回の配信を提供します。これにより、重複行が発生する可能性があります。検証手順の一部として重複排除を実行します。

移行したデータの検証

ソースデータベースと宛先データベースを比較して、すべてのデータが正常に移行されたことを確認します。これには、LiveCompare を使用できます。

関連トピック

- LiveCompare —移行の完全性を確認するための行レベルのデータ比較ツール
- 用語 —スナップショット、ストリーミング、およびその他の移行概念の定義

References

DMS agent version compatibility

DMS エージェント `cdcagent` は、ハイブリッドマネージャー HM とは無関係にバージョン付けおよびリリースされます。互換性のないバージョンを使用すると、サイレント障害が発生する可能性があります。HM コンソールはデータベースを正しく認識しますが、問題は移行が開始された場合にのみ表示されます。

この表は、各 HM リリースに必要な最小 `cdcagent` バージョンを示しています。最小よりも新しいバージョンの使用がサポートされており、お勧めします。

HM version	Minimum *cdcagent* version
1.4.0 (LTS)	1.13.12
2026.5 (IR)	1.12.40
2026.4 (IR)	1.12.8
2026.3 (IR)	1.10.21
2026.2 (IR)	1.9.4
1.3.x (LTS)	1.3.23

DMS エージェントをインストールまたはアップグレードするには、Installing the DMS agent を参照してください。

EDB DMS agent parameter reference

DMS エージェントの一般的な構成

これらのパラメーターは、EDB DMS エージェントが次の目的で必要な情報を定義します。

- 移行に使用している Hybrid ManagerHM インスタンスとプロジェクトにアクセスします
- ソースデータベースに接続してデータを収集するか、宛先データベースに接続してデータを移行します

> **注釈**
>
> DMS エージェントインスタンスは、リーダーまたはライタモードでのみ実行できます。

DBCONFIG_ID

ソース/宛先を識別するために割り当てる名前。この名前は、後で HM コンソールの**ソース** タブの **Migrate** セクションにソースとして、または **Destinations** タブに宛先として表示されます。

ID の場合

- 最大文字長は 255 文字です。
- 小文字、大文字、数字、アンダースコア_、ハイフン-が使用できます。他の特殊文字はサポートされていません。
- ID はプロジェクト内で一意である必要があります。同じプロジェクト内の 2 つ以上のソース/デスティネーションに同じ ID を使用しないでください。

RUN_MODE

EDB DMS エージェントを実行する方法を決定します。DMS エージェントを使用して HM を移行ソースに接続している場合、`reader` に設定します。 HM を外部の自己管理の移行先に接続する場合、`writer` に設定します。

移行の宛先が自己管理の Postgres データベースである場合にのみ、DMS エージェントをライタモードで実行します。

RW_SERVICE_HOST

EDB DMS エージェントが移行を実行するために使用する DMS の URL を指定します。この値は展開スコープです。特定の HM 展開内のすべてのプロジェクトに同じ URL が適用されます。この値は、HM コンソールの**ソース** または **デスティネーション** 移行ページから取得します。

`RW_SERVICE_HOST` を取得するには

1. HM コンソールで、移行に使用するプロジェクトを選択します。
2. 左側のナビゲーションで **Migrate** を選択し、 **Sources** または **Destinations** を選択します。

Sources と **Destinations** ページの両方に、ページの上部のバナーに `RW_SERVICE_HOST` が表示されます。

1. コピーアイコンを使用して値を取得し、EDB DMS エージェントの構成に使用します。

CREDENTIAL_DIRECTORY_PATH

Configuring and running the EDB DMS agent 資格情報フォルダーのディレクトリパス。

DMS エージェントデータベース構成

これらのパラメーター `DBCONFIG_DATABASES` セクションは、EDB DMS エージェントが移行のために正しいデータベースに接続するために必要なソース/宛先データベース情報のリストを定義します。

複数のデータベースを移行するように EDB DMS エージェントを構成できます。`DBCONFIG_DATABASES_0__TYPE` セクションは、最初のデータベースの情報を区切ります。`DBCONFIG_DATABASES_1__TYPE` を使用して、2 番目のデータベースにデータを提供できます。インデックスを手動で増やして、EDB DMS エージェント `DBCONFIG_DATABASES_2__TYPE`、`DBCONFIG_DATABASES_3__TYPE` にセクションを追加します。

DBCONFIG_DATABASES_0__TYPE

ソース/宛先データベースタイプ。

EDB DMS エージェントは、ソースデータベースタイプリーダーモードとして `ORACLE` および `POSTGRES` をサポートしますが、宛先タイプライタモードとしては `POSTGRES` のみをサポートします。

DBCONFIG_DATABASES_0__HOSTNAME

ソース/宛先データベースのホスト名。

> **注釈**
>
> RDS データベースの場合、 *DBCONFIG_DATABASES_0__HOSTNAME* 値が、AWS コンソールの**接続とセキュリティ**の下にある RDS エンドポイントを指していることを確認します。編集された例は `foo_database.cb7asw1xxxxx.eu-west-0.rds.amazonaws.com` です。

DBCONFIG_DATABASES_0__PORT

ソース/宛先データベースのポート。

DBCONFIG_DATABASES_0__CATALOG

ソース/宛先データベースサーバーのデータベース名。

Oracle ソースの場合

- **CDB/PDB** `CDB_NAME/PDB_NAME` `ORCLCDB/ORCLPDB1` などを使用します。エージェントは/ で分割して、CDB と PDB を個別に識別します。PDB 名のみを使用すると、CDB として扱われ、PDB は設定されないままになります。
- **非 CDB** SID またはサービス名を使用しますたとえば、 `ORCL`

DBCONFIG_DATABASES_0__USERNAME

ソース/宛先データベースのデータベースユーザー名。専用の移行ユーザーを使用します。EDB EDB Postgres AI エージェント用に作成したユーザーと同じユーザーにすることができます。

> **注釈**
>
> Oracle ソースの場合、 *SYS* または他の特権システムアカウントではなく、専用の移行ユーザーを使用します。必要な移行ユーザーの作成手順については、 *Preparing Oracle source database* を参照してください。

DBCONFIG_DATABASES_0__PASSWORD

ソース/宛先データベースのデータベースユーザー名のパスワード。スクリプトで指定せずにパスワードの環境変数を設定できますが、変数は次のフォーマットに従う必要があります。`export DBCONFIG_DATABASES_0__PASSWORD=password`

DBCONFIG_DATABASES_0__RESOURCEID

このデータベースを HM に登録された対応物と関連付けるユーザー定義の識別子。

以前にこのデータベースを EDB Postgres AI エージェントに登録した場合、この値は `beacon_agent.yaml` でこのデータベースに設定された `resource_id` と**一致する必要があります**。これは、HM が DMS エージェントソースを登録済みデータベースと関連付ける方法です。不一致があると、それらは別の無関係なエントリとして表示されます。

DBCONFIG_DATABASES_0__INCLUDELIST

移行に使用できるテーブルを制御する `schema.table` トークンのコンマ区切りリスト。スキーマまたはテーブル名のワイルドカードとして* を使用しますたとえば、 `SCHEMA1.*` または`*.TABLE1` など。

存在しないまたは空の場合、すべてのテーブルが含まれます。 `EXCLUDELIST` も設定されている場合、除外が優先されます。

DBCONFIG_DATABASES_0__EXCLUDELIST

常に移行から除外するテーブルの `schema.table` トークンのコンマ区切りリスト。 * をワイルドカードとして使用します。`INCLUDELIST` よりも優先されます。除外パターンに一致するテーブルは、含めるルールに関係なく、常に除外されます。

DBCONFIG_DATABASES_0__SSLMODE

データベース接続の SSL/TLS 動作を制御します。受け入れられる値

- `STRICT` — verify-full; 有効で信頼できる証明書が必要です
- `DEFAULT` — 暗号化が必要ですが、証明書を検証しません
- `WEAK` — 暗号化を優先しますが、平文フォールバックを許可します

DBCONFIG_DATABASES_0__SOCKETTIMEOUT

データベース接続の秒単位のソケットタイムアウト。デッド接続の検出に使用されます。デフォルトライタモードの場合は `600` 、リーダーモードの場合は `60` 。

DBCONFIG_DATABASES_0__TCPKEEPALIVE

データベース接続で TCP キープアライブプローブを有効にします。デフォルト `true`

DBCONFIG_DATABASES_0__TCPKEEPALIVESIDLE

最初の TCP キープアライブプローブが送信されるまでの非アクティブな秒数。PK アプリケーションやスナップショット後の重複排除などの長時間実行されるライタ操作中に使用されます。デフォルト `60`

DBCONFIG_DATABASES_0__TCPKEEPALIVESINTERVAL

TCP キープアライブ再試行プローブ間の秒数。デフォルト `10`

DBCONFIG_DATABASES_0__TCPKEEPALIVESCOUNT

接続がデッドとみなされるまでの、失敗した TCP キープアライブプローブの最大数。デフォルト `6`

DBCONFIG_DATABASES_0__JDBCOPTIONS

データベース接続に渡す任意の JDBC ドライバープロパティ。`key=value;key=value` 形式で。これらはデフォルトの後、`SSLMODE` や `SOCKETTIMEOUT` のような明示的な設定の前に適用されます。

Oracle の例

- ネットワーク暗号化ネゴシエーションを無効にする `oracle.net.encryption_client=REJECTED`
- セッションプログラム名を設定します `v$session.program=DMSAgent`

オプショナルパラメーター

FIPS_MODE

FIPS 標準に準拠する必要がある場合は、環境変数 `export FIPS_MODE=true` を true に設定して、DMS エージェントの FIPS モードを有効にします。

TRANSFORMATION_ENABLE_NULLCHAR_REPLACEMENT

Oracle 固有。テキストベースの列タイプ `VARCHAR2` 、`LONG` 、`CLOB` 、および `NCLOB` の Null 文字 `0x00` の置換を有効にします。デフォルト `false`

`TRANSFORMATION_NULLCHAR_REPLACEMENT` とともに使用して、置換文字列を指定します。

TRANSFORMATION_NULLCHAR_REPLACEMENT

Oracle 固有。 `TRANSFORMATION_ENABLE_NULLCHAR_REPLACEMENT` が有効になっているときにヌル文字 `0x00` を置き換えるために使用される文字列。デフォルトは空の文字列です。

TRANSFORMATION_ENABLE_SPECIAL_COLUMN_NAME_ENCODING

Oracle 固有。列名の特殊文字`$` および`-` のエンコードを有効にします。デフォルト `false`

DBZ_*

Debezium プロパティのパススルー。 `DBZ_` を接頭辞として付けることにより、Debezium コネクタプロパティをオーバーライドできます。たとえば、`max.batch.size` を設定するには、 `export DBZ_MAX_BATCH_SIZE=204800` を使用します。

サポートされているプロパティの完全なリストについては、Debezium ドキュメントを参照してください。

- Oracle connector properties
-

PostgreSQL connector properties

QuarkUS_LOG_LEVEL

基になるすべてのフレームワークとサードパーティのライブラリを含む Quarkus アプリケーション全体のデフォルトのログレベルを設定します。

アプリケーションエコシステム全体内の問題を診断する必要がある場合を除き、これを DEBUG または TRACE に設定することは避けてください。この方法を使用すると、過度に詳細で分析が困難なログが生成されます。集中的なデバッグには、代わりに QUARKUS_LOG_CATEGORY__COM_ENTERPRISEDB__LEVEL を使用します。

QUARKUS_LOG_CATEGORY COM_ENTERPRISEDB LEVEL

com.enterprisedb パッケージ内のクラスに専用のログレベルを設定します。

これは、デバッグに推奨されるパラメーターです。DMS エージェントのコアロジックスキーマ抽出、移行評価などに関連するコンポーネントの詳細なログを取得するには、このパラメーターを DEBUG または TRACE に設定します。これにより、有用な出力が分離され、サードパーティライブラリからのメッセージが除外されます。

Installing EDB Data Migration Service agent on Linux

> **注釈**
>
> Hybrid Manager バージョン 1.3 以降を使用して EDB DMS エージェントをインストールおよび構成します。 Hybrid Manager 1.2 以前のバージョンの場合、EDB DMS Reader および EDB DMS Writer を使用してください。

リンクを選択して、該当するインストール手順にアクセスします。

Linux x86-64 (amd64)

Red Hat Enterprise Linux RHEL とその派生

- *Installing EDB Data Migration Service agent on RHEL 9 or OL 9 x86_64* 、*Installing EDB Data Migration Service agent on RHEL 8 or OL 8 x86_64*
- *Installing EDB Data Migration Service agent on RHEL 9 or OL 9 x86_64* 、*Installing EDB Data Migration Service agent on RHEL 8 or OL 8 x86_64*
- *Installing EDB Data Migration Service agent on AlmaLinux 9 or Rocky Linux* 、 *Installing EDB Data Migration Service agent on AlmaLinux 8 or Rocky Linux*
- *Installing EDB Data Migration Service agent on AlmaLinux 9 or Rocky Linux* 、 *Installing EDB Data Migration Service agent on AlmaLinux 8 or Rocky Linux*

SUSE Linux Enterprise SLES

- *Installing EDB Data Migration Service agent on SLES 15 x86_64*

Debian とデリバティブ

- *Installing EDB Data Migration Service agent on Ubuntu 24.04 x86_64* 、 *Installing EDB Data Migration Service agent on Ubuntu 22.04 x86_64*
- *Installing EDB Data Migration Service agent on Debian 12 arm64*、*Installing EDB Data Migration Service agent on Debian 11 x86_64*

Linux Installing EDB Data Migration Service agent on Linux AArch64 (ARM64)

Debian とデリバティブ

- *Installing EDB Data Migration Service agent on Debian 12 arm64*

Installing EDB Data Migration Service agent on Linux x86 (amd64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing EDB Data Migration Service agent on RHEL 9 or OL 9 x86_64*
- *Installing EDB Data Migration Service agent on RHEL 8 or OL 8 x86_64*
- *Installing EDB Data Migration Service agent on RHEL 9 or OL 9 x86_64*
- *Installing EDB Data Migration Service agent on RHEL 8 or OL 8 x86_64*
- *Installing EDB Data Migration Service agent on AlmaLinux 9 or Rocky Linux*
- *Installing EDB Data Migration Service agent on AlmaLinux 8 or Rocky Linux*
- *Installing EDB Data Migration Service agent on AlmaLinux 9 or Rocky Linux*
- *Installing EDB Data Migration Service agent on AlmaLinux 8 or Rocky Linux*

SUSE Linux Enterprise SLES

- *Installing EDB Data Migration Service agent on SLES 15 x86_64*

Debian とデリバティブ

- *Installing EDB Data Migration Service agent on Ubuntu 24.04 x86_64*
- *Installing EDB Data Migration Service agent on Ubuntu 22.04 x86_64*
- *Installing EDB Data Migration Service agent on Debian 12 arm64*
- *Installing EDB Data Migration Service agent on Debian 11 x86_64*

Installing EDB Data Migration Service agent on Debian 11 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント `cdcagent` としてパッケージ化されたものをインストールします。

```
sudo apt-get install cdcagent
```

Installing EDB Data Migration Service agent on Debian 12 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。

4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo apt-get install cdcagent
```

Installing EDB Data Migration Service agent on AlmaLinux 8 or Rocky Linux

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo dnf install cdcagent
```

Installing EDB Data Migration Service agent on AlmaLinux 9 or Rocky Linux

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo dnf install cdcagent
```

Installing EDB Data Migration Service agent on RHEL 8 or OL 8 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo dnf install cdcagent
```

Installing EDB Data Migration Service agent on RHEL 9 or OL 9 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
dnf repolist | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント `cdcagent` としてパッケージ化されたものをインストールします。

```
sudo dnf install cdcagent
```

Installing EDB Data Migration Service agent on SLES 15 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
zypper lr -E | grep enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。

4. EDB リポジトリの設定手順に従ってください。

- 必要な SUSE モジュールをアクティブにします。

```
sudo SUSEConnect -p PackageHub/15.7/x86_64
```

- メタデータを更新します。

```
sudo zypper refresh
```

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo zypper install cdcagent
```

Installing EDB Data Migration Service agent on Ubuntu 22.04 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント cdcagent としてパッケージ化されたものをインストールします。

```
sudo apt-get install cdcagent
```

Installing EDB Data Migration Service agent on Ubuntu 24.04 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント `cdcagent` としてパッケージ化されたものをインストールします。

```
sudo apt-get install cdcagent
```

Installing EDB Data Migration Service agent on Linux AArch64 (ARM64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Debian とデリバティブ

- *Installing EDB Data Migration Service agent on Debian 12 arm64*

Installing EDB Data Migration Service agent on Debian 12 arm64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリが存在するかどうかを判断するには、次のように入力します。

```
apt-cache search enterprisedb
```

出力が生成されない場合、リポジトリはインストールされます。

EDB リポジトリをセットアップするには

1. EDB repositories に移動します .
2. EDB リポジトリへのアクセスを提供するボタンを選択します。
3. ダウンロードするプラットフォームとソフトウェアを選択します。
4. EDB リポジトリの設定手順に従ってください。

パッケージをインストールする

EDB DMS エージェント `cdcagent` としてパッケージ化されたものをインストールします。

```
sudo apt-get install cdcagent
```

Using the HM console to view migration assessments

Hybrid ManagerHM の主な使用の 1 つは、データベースの移行の計画と実行に役立つことです。

外部 Oracle または Postgres データベースを EDB Postgres AI エージェントたとえば *Configuring the EDB Postgres AI agent to register your Oracle database* に登録できます。 **Estate** ページからアクセスできる **移行**タブには、登録した外部データベースと、計画に使用できる移行評価がリストされます。

これは、外部データベースを HM に移行する際の HM コンソールの最初の停止です。

Migration テーブル

デフォルト列

テーブルのデフォルトビューには、分析されたデータベースごとに次の情報が含まれています。

データベース名 評価されたデータベースの名前。

データベースタイプデータベースタイプ。現在サポートされているデータベースタイプについては、**スキーマの互換性の概要** を参照してください。

リソース ID: `beacon_agent.yaml` 構成でデータベースに割り当てたリソース ID または `databases.resource_id` 。詳細については、 EDB Postgres AI エージェントの *EDB DMS agent parameter reference* を参照してください。

バージョンデータベースバージョン。現在サポートされているバージョンについては、**スキーマの互換性の概要** を参照してください。

 # or Cores: データベースに割り当てられたコアの数。

# of Schemas: DSN ユーザーがアクセスできる評価済みデータベース内のスキーマの数。DSN ユーザーが `SELECT` 特権を持つ少なくとも 1 つのテーブルが含まれています。 DSN ユーザーは、EDB Postgres AI エージェントの構成中に指定したソースデータベースユーザーです。

- Oracle データベースでは、ビルトイン `SYS` および `SYSTEM` スキーマは列数から除外されます。

- DSN ユーザーのスキーマは、テーブルがない場合、通常除外されます。ただし、既知の問題により、DDL エクストラクターが抽出中に一時テーブルを作成するときに含まれる場合があります。
- EDB Postgres AI エージェントは、実行中にテーブルと列のスキーマ変更を検出します。ただし、ビューまたはプロシージャへの変更は、EDB Postgres AI エージェントを再起動するまで検出されません。

Engine: この列は Postgres データベースにのみ適用され、 PostgreSQL、EDB Postgres Advanced Server、または EDB Postgres Extended Server などのデータベースサーバーディストリビューションを示します。

複雑度この値は Oracle スキーマに対してのみ生成され、データベースを HM に移行するための複雑さのレベルを提供します。可能な値は次のとおりです。

- **非常に簡単** Oracle スキーマには互換性のない機能やオブジェクトが含まれておらず、データモデルは EDB Postgres Advanced Server と完全な互換性があります。
- **簡単**移行は比較的簡単です。データベースの構造と機能には互換性があり、最小限の調整が必要です。
- **中**中程度の移行作業が必要です。一部のデータベース構造または機能は、 EDB Postgres Advanced Server で同様の機能を取得するために変更を必要とする場合があります。
- **ハード** 多大な努力と専門知識が必要です。Oracle のデータベース構造、機能またはカスタムコードには、移行中に対処する必要がある大きな違いまたは複雑さがあります。
- **非常に難しい** 移行は非常に複雑で、要求が高いです。 Oracle と EDB Postgres Advanced Server データベースの構造、機能、カスタムコードの大幅な違いには、データ、オブジェクト、ステートメントを変換できるように専門知識が必要です。コードを書き直すか、アーキテクチャの再設計が必要になる場合があります。

移行 LoE この値は、Oracle スキーマについてのみ生成され、スキーマを変換し、データを HM に移行するために必要な営業日 8 時間単位の労力レベルを表します。この推定は、Oracle から Postgres への移行を専門とする EDB のプロフェッショナルサービスのメンバーによって適用される労力に対応しています。この値は、データベース機能、スキーマの複雑さ、データボリューム、および日数を見積もるその他の要素を含むデータベースおよびスキーマレベルの評価を考慮します。

ステータス移行の現在の状態。データベースの *Viewing migration assessment details* からステータスを手動で更新して、進行状況を反映できます。

その他の列

歯車アイコンを使用して、追加の列を表示できます。

最終評価データベースが最後に評価されて、分析が最新であるかどうかを判断してから経過した期間。

管理データベースが自己管理、クラウドサービスプロバイダー、または HM によって管理されているかどうかを示します。

プロジェクト名 EDB Postgres AI エージェントの構成に使用されたプロジェクト。これは、`agent.project_id` でエージェントに割り当てるプロジェクトです。詳細については、 EDB Postgres AI エージェント *EDB DMS agent parameter reference* を参照してください。

タグ `databases.tags` の特定のデータベースの移行に割り当てられたラベル。詳細については、 EDB Postgres AI エージェント *EDB DMS agent parameter reference* を参照してください。

TDS: GB 単位の合計データベースサイズ。これには、データベースデータのみが含まれます。メタオブジェクト、一時ファイル、ログファイルなど用に、移行先データベースに追加の領域を計画します。

移行テーブルビューの表示とフィルタリング

Estate ページの **Migration** タブには、 EDB Postgres AI エージェントに登録した後に HM で移行できるすべてのデータベースのテーブルベースの分析が表示されます。評価結果の表示またはデータベースのフィルタリング方法をカスタマイズするには、次のことができます。

- キーワードを入力して特定のデータベースを検索します。 **Search** ファンクションは、キーワードを使用して、 **リソース ID** 、**プロジェクト名** 、および **データベースタイプ**値と一致するデータベースを表示します。
- **Filter** セレクターを使用して、Postgres バージョンとプロジェクト名でデータベースを表示します。
- 結果を新しいものから古いもの、古いものから新しいもの、およびアルファベット順に並べます。
- 稲妻アイコンを使用して、テーブルを更新する頻度を選択します。アイコンが有効になっていない場合、テーブルは手動で更新しない限り更新されません。**自動更新**が有効になっている場合、テーブルは 30 秒ごとに更新されます。または、事前構成された間隔のいずれかを選択できます。
- 歯車アイコンを使用して、[**移行**] タブに表示される列をカスタマイズします。列をデフォルトの状態にリセットすることもできます。
- テーブルのサイズを画面に合わせて調整します。

関連トピック

- **スキーマの互換性の概要** を参照して、移行評価がサポートされているデータベースを理解します。
- データが EDB Postgres AI エージェントによってどのように保存されるかを理解するには、*Data security and sovereignty* を参照してください。

Migration assessment support

自己管理データベースを EDB Postgres AI エージェントに登録した後、HM は移行評価を生成します。評価は現在、次のデータベースバージョンでサポートされています。

> **注釈**
>
> 現在、Postgres 評価は高レベルの移行の概要を提供します。 Oracle データベースでは、より詳細な分析が利用できます。

Oracle

- Oracle 11g
- Oracle 12c
- Oracle 18c
- Oracle 19c

新しいバージョン 21c、23c なども評価される可能性がありますが、まだ完全に認定されていません。

Postgres

- PostgreSQL 13-18
- EDB Postgres Advanced Server 13-18
- EDB Postgres 拡張サーバー 13-18

Viewing migration assessment details

ハイブリッドマネージャー HM の **エステート** ページの **移行**タブからデータベースを選択すると、詳細ビューが開きます。このビューを使用して、外部データベースからの移行を計画および実行します。**Schemas** および **Assessment** タブには、ソースデータベースが Postgres または Oracle のいずれであるかに応じて、異なる情報が表示されます。

ステータス

データベースの詳細ページの上部に、データベース名の横にステータスセレクターが表示されます。これを使用して、データベースの移行の現在の状態を更新し、移行ワークフローを追跡し、自分とチームメンバーを明確にします。

次のステータスから選択できます。

- アセスメント デフォルト
- 企画
- 移行しないことにした
- 進行中
- 完了

スキーマ

[**Schemas**] タブは、接続したデータベースで見つかったスキーマの概要を提供します。データベースのサマリーメトリックと、少なくとも 1 つのテーブルで SELECT アクセスを付与したスキーマのリストが表示されます。少なくとも 1 つのテーブルを含むスキーマのみが表示されます。

この情報は、移行作業の範囲を調整し、スキーマの移行を効果的に計画するのに役立ちます。

> **重要**
>
> EDB Postgres AI エージェントは、実行中にテーブルと列のスキーマ変更を検出します。ただし、ビューまたはプロシージャへの変更は、 EDB Postgres AI エージェントを再起動するまで検出されません。

サマリーカード

- 選択したデータベースのスキーマの合計。
- 選択したデータベースのテーブルの合計数。
- 選択したデータベースのオブジェクトの合計数。

スキーマテーブル

- **Name:** スキーマの名前。
- **# of Tables:** スキーマ内のテーブルの数。
- **# of Objects:** スキーマ内のオブジェクトの数。
- **MP プロジェクト**このスキーマに関連付けられている移行ポータルプロジェクトへのリンク。Postgres データベースの移行ポータルプロジェクトは、 EDB Postgres AI エージェントによって自動的に作成されます。

行を選択すると展開され、以下が表示されます。

スキーマ互換性の概要移行ポータルから報告されたスキーマオブジェクトの互換性ステータスを示します。障害、システム修復、ユーザー修復、および合格したオブジェクトの数、および合計オブジェクト数と合格した割合が含まれます。

移行ポータルプロジェクトテーブルスキーマに関連付けられている移行ポータルプロジェクトをリストします。プロジェクト名、ターゲット EDB Postgres Advanced Server バージョン、作成日、使用可能なアクションを含みます。

サマリーカード

- 選択したデータベースのスキーマの合計。
- 選択したデータベースのテーブルの合計数。
- 選択したデータベースのオブジェクトの合計数。
- すべてのスキーマの複雑さの概要、複雑さごとに色分けされています Very Easy、Easy、Medium、Hard、Very Hard。
- 8 時間勤務日で定義された日の範囲 0〜5、6〜15、16〜25、26〜40、40+日によって色分けされた移行作業レベル。

スキーマテーブル

- **Name:** スキーマの名前。
- **最新の評価** 最新の評価のステータス。使用可能な値は、Not Started、In Progress、Complete、および Error です。
- **最終評価** 最終評価のタイムスタンプ。 EDB Postgres AI エージェントが新しいスキーマまたは既存のスキーマの更新を検出した場合、評価は再生成または更新されます。
- **# of Tables:** スキーマ内のテーブルの数。

- **# of Objects:** スキーマ内のオブジェクトの数。
- **複雑度** 詳細なオブジェクトレベルの分析に基づいて、スキーマを EDB Postgres Advanced Server に変換することがいかに難しいかを示します。指定できる値は、Very Easy、Easy、Medium、Hard、および Very Hard です。スキーマレベルの分析はより詳細であるため、単一のスキーマの複雑さはデータベース全体の複雑さよりも高くなる場合があります。
- **移行 LoE** 詳細なオブジェクトレベルの分析に基づいて、スキーマの移行にかかる日数を 8 時間営業日で示します。
- **依存関係**現在のスキーマが依存するスキーマの名前。依存関係を含むスキーマとその依存関係を同じバッチで移行ポータルプロジェクトを使用して、参照されているすべてのオブジェクトが宛先データベースに存在することを確認します。
- **MP プロジェクト**このスキーマに関連付けられている移行ポータルプロジェクトへのリンク。新しいプロジェクトを作成するには、表の上にある**+ New MP Project** を選択し、必要な詳細を提供します。HM は、移行ポータル**プロジェクトスコープのワークスペースとユーザーのワークスペース** を使用してバックグラウンドでプロジェクトを作成します。後で ***Migration*** タブからアクセスできます。

行を選択すると展開され、以下が表示されます。

スキーマ互換性の概要移行ポータルから報告されたスキーマオブジェクトの互換性ステータスを示します。障害、システム修復、ユーザー修復、および合格したオブジェクトの数、および合計オブジェクト数と合格した割合が含まれます。

移行ポータルプロジェクトテーブルスキーマに関連付けられている移行ポータルプロジェクトをリストします。プロジェクト名、ターゲット EDB Postgres Advanced Server バージョン、作成日、使用可能なアクションを含みます。Actions 列からプロジェクトを削除できます。

アセスメント

[**Assessment**] タブは、接続されたデータベースの移行評価を提供し、作業の範囲を調整し、データベースオブジェクトの移行を計画するのに役立つ洞察を提供します。このセクションには、主要なメトリックと、生成された分析に寄与する要素の詳細な内訳を含む評価の概要が含まれています。

`.json` フォーマットで評価結果をエクスポートするには、 **Download Report** を選択します。

評価の詳細データベースの高レベルの移行評価の概要と、評価で使用される重要な情報を表示します。Postgres データベースの場合、これにはデータベース名、エンジン、バージョン、サイズ、オブジェクト数、テーブル数、スキーマ数が含まれます。Oracle データベースの場合、移行の複雑さ、使用される Oracle データベース機能、使用される Oracle PL/ SQL 機能、データベース移行 LoE、CPU 数、評価されたスキーマの数、評価タイプ、最終評価タイムスタンプも含まれます。

インストールされた拡張機能選択したデータベースにインストールされている外部データベースラッパーを含む拡張機能をリストします。データベースを移行する前に、EDB Postgres extensions ドキュメントを確認して、EDB サポートされている各データベースでサポートされている拡張機能を判断します。

Foreign Servers 選択したデータベースで作成された外部サーバーと、それらの構成に使用される対応する外部データラッパー FDW をリストします。移行する前に、各 FDW が宛先データベースと互換性があること、および宛先が外部サーバーに到達できることを確認します。

外部テーブル選択したデータベースの評価に含まれるスキーマの外部テーブルの数をリストします。スキーマおよび外部サーバーごとにグループ化されています。

テーブルスペーステーブルスペース名前、所有者、場所を含む、選択したデータベース内のテーブルスペースをリストします。

テーブルスペーススキーマ使用状況の概要選択したデータベース内の非ビルトイン Postgres テーブルスペース `pg_default` および `pg_global` を除く、およびスキーマごとにこれらのテーブルスペースに作成されたテーブル、インデックス、またはマテリアライズドビューオブジェクトの数をリストします。評価。

完全なレプリカ ID のないスキーマ評価に含まれる各スキーマのテーブルで、主キー、一意キー、またはレプリカ ID が `FULL` に設定されていないテーブルの数をリストします。この条件は、変更データキャプチャ CDC ストリーミングデータの移行を実行する機能に影響を与えます。

インストールされたプロシージャル言語選択したデータベースにインストールされているプロシージャル言語をリストします。アプリケーション要件をサポートするためにこれらの言語が必要な場合は、データベースを移行する前に、EDB Postgres extensions ドキュメントを確認して、EDB サポートデータベースでの使用がサポートされている言語を判断してください。また、インストールされた言語のいずれかが信頼できない言語の場合、その使用がセキュリティポリシーに準拠し、サポートされているかどうかを判断します。

互換性の概要データベース機能とスキーマオブジェクト、および全体的な移行の複雑さと作業量の評価に寄与するその他の要素の高レベルの互換性の概要を表示します。メトリックには、合計スキーマ、評価された PL/ SQL オブジェクトの合計、互換性オブジェクト、既知の回避策と互換性のないオブジェクトとオカレンス、さらなる分析が必要な互換性のないオブジェクトとオカレンス、使用されている Oracle データベース機能、データベースサイズ、PL/ SQL 行、ラージオブジェクトの使用状況、パーティションが含まれます使用されるタイプ、集計クエリ、および SQL カウント。

使用される Oracle PL/ SQL 機能 Oracle ビルトイン機能を含むすべての PL/ SQL オブジェクトをリストします。表には、機能ごとに、オブジェクト数、発生数、EPAS 互換かどうか Yes/No、および移行の複雑度が示されています。

使用される Oracle データベース機能データベースで使用されているすべての Oracle データベース機能をリストします。移行の複雑さ、日単位の移行作業レベル LoE など。

オブジェクトの互換性のない PL/ SQL 機能データベースオブジェクトで見つかった互換性のない PL/ SQL 機能をすべてリストします。表には、エントリごとに、スキーマ名、オブジェクトタイプファンクション、パッケージなど、オブジェクト名、互換性のない PL/ SQL 機能、出現数、移行の複雑度が表示されます。

移行

Migration タブは、選択したデータベースの移行関連のタスクを実行するのに役立ちます。

移行ワークフロー

宛先の作成クラスター作成ワークフローを開きます。ここで、移行の宛先として機能する新しいデータベースクラスターを設定できます。詳細については、*Creating a database cluster* を参照してください。

移行スキーマデータベーススキーマとデータの両方をターゲットクラスタに移動する移行を定義します。**Migrate Schema(s)** を選択して開始します。

データの移行データの整合性を維持し、ダウンタイムを最小限に抑えながら、完全なデータセットを新しいクラスターに転送します。**Migrate Data** を選択して開始します。エンドツーエンドの例については、*Choose your migration path* を参照してください。

アクティブな移行

選択したデータベースに関連付けられている移行をリストします。 **View All Migrations** を選択して、資産全体のすべての移行を表示します。

テーブルには次の列が含まれます。

- **Name:** 移行の名前、移行の詳細ページにリンクされています。
- **ソース** 移行される外部ソースデータベース。
- **宛先** ターゲットデータベースとクラスター。
- **スナップショットタイプ**使用されるスナップショットのタイプ。使用可能な値は **Snapshot** および **Snapshot + Streaming** です。
- **Scope:** 移行のスコープ。使用可能な値は **スキーマのみ** 、**データのみ** 、および **スキーマ+データ** です。
- **移行の進行状況**移行の 3 つのメトリック、合計テーブル、完了したスナップショットテーブル、および移行された行を表示します。スキーマのみの移行の場合は表示されません。
- **ステータス** 移行の現在のステータス。

Assessing multiple databases at once

大規模なデータベース資産数十の Oracle または Postgres データベースがある場合、DSN 文字列を手動で構成し、それぞれに `beacon_agent.yaml` エントリを追加するのは、時間がかかり、エラーが発生しやすくなります。 EDB Postgres AI エージェントの `configure-onprem-dbs` コマンドは、既存のデータベースインベントリをファイルから読み取りまたは AWS から直接データベースを検出し、正しい DSN エントリで `beacon_agent.yaml` に自動的に入力することにより、この問題を解決します。

設定された構成でエージェントを起動すると、エージェントはすべての登録データベースを HM に登録し、それぞれの移行評価を生成します。`configure-onprem-dbs` への大量登録は、個々の移行プロジェクトにコミットする前に資産全体の移行準備状況レポートが必要な場合に、推奨される開始点です。

> **注釈**
>
> *configure-onprem-dbs* は、*beacon_agent.yaml* の `provider.onprem.databases` セクションに DSN エントリのみを移入します。完全なエージェント構成の生成、エージェントの起動、または HM への接続はしません。

サポートされている入力ソース

`configure-onprem-dbs` コマンドは、3 つのソースから構成を設定できます。

- CSV ファイル
- Oracle `tnsnames.ora` ファイル
- AWS RDS ディスカバリ

完全なコマンドリファレンス

使用可能なすべてのフラグ---region 、--rds-engine 、--tag-filter などの RDS 固有のオプションを含む-については、 EDB Postgres AI エージェント CLI リファレンスの :ref:``configure-onprem-dbs` を使用して DSN エントリを設定する <*configure-onprem-dbs* を使用して DSN エントリを設定する>` を参照してください。

関連トピック

- 移行評価がサポートされている Oracle および Postgres のバージョンについては、*Migration assessment support* を参照してください。
- はじめに *Enrolling multiple databases for assessment* を参照してください。

Enrolling multiple databases for assessment

EDB Postgres AI エージェントを設定および実行して、単一のワークフローでの移行評価のために複数のデータベースを登録します。

前提条件

これらの前提条件は、ソースデータベースタイプに関係なく適用されます。

- **EDB Postgres AI エージェントは補助マシンまたは要塞ホストにインストールされています**。補助マシンの構成方法は、ソースデータベースタイプによって異なります。*Configuring your auxiliary machine or bastion host* 、 *Configuring your auxiliary machine or bastion host* 、および *Configuring your auxiliary machine or bastion host* を参照してください。
- **アクセスキーを持つマシンユーザー** 。 `estate ingester` ロールと有効なアクセスキーを持つ HM マシンユーザーが必要です。*Creating a machine user* を参照してください。

登録する前に、 **データベースごとに個別に**、次の要件を完了する必要があります。エージェントのインストールとマシンユーザーは一度セットアップされ、すべてのデータベースで再利用されます。

- **Oracle ソース** - 各ソースデータベースには、適切な読み取り権限を持つ専用の移行ユーザーが必要です。*Preparing Oracle source database* を参照してください。
- **Postgres ソース** —各ソースデータベースには、十分な読み取りアクセスを持つユーザーが必要です。*Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* 自己管理 Postgres または *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* AWS RDS /Aurora 管理の PostgreSQL を参照してください。

> **注釈**
>
> データベースは、エージェントが接続できる限り HM の**Estate > Migrations**の下に表示されます。完全な特権セットがないと、スキーマ数、複雑さスコア、および評価データは表示されません。 !!!

beacon_agent.yaml の準備

この手順を処理する方法は、構成ファイルが既に存在するかどうかによって異なります。

既に ``beacon_agent.yaml`` がありますので、この手順をスキップします。次のステップでは、既存のファイルに--output-yaml をポイントすると、configure-onprem-dbs は新しいデータベースエントリをそれにマージし、他のすべての設定は変更しないままにします。

``beacon_agent.yaml`` はまだありません — 2 つのオプションがあります。

- 最初に完全な構成ファイル HM 接続パラメーター、プロジェクト ID などをセットアップし、provider.onprem.databases セクションを空のままにします。次に、次のステップで configure-onprem-dbs を使用してデータを設定します。ソースデータベースタイプの構成ガイドに従って、ファイルを作成します。
- **Oracle** — *Configuring the EDB Postgres AI agent* を参照してください .
- **自己管理 Postgres** — *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* を参照してください .
- **AWS RDS / Aurora** — *Configuring the EDB Postgres AI agent to register your AWS RDS/Aurora PostgreSQL database* を参照してください。
- 最初に configure-onprem-dbs を実行し、後でエージェント構成を追加します。このコマンドは、DSN エントリを含む provider.onprem.databases セクションのみを含む最小限のファイルを作成します。次に、上記と同じガイドに従って、HM 接続パラメーターとその他の設定を手動で追加する必要があります。

configure-onprem-dbs を使用して DSN エントリを設定する

configure-onprem-dbs を実行してデータベースインベントリを読み取り、作成した構成ファイルに provider.onprem.databases エントリを追加します。入力ソースに一致する例を選択します。

プロンプト中に、各データベースの資格情報を個別に入力したり、エントリをスキップしてプレースホルダーを残したり、残りのすべてのエントリに同じ資格情報を適用したり、残りのすべてのエントリを一度にスキップしたりできます。実行後もプレースホルダー資格情報を使用しているエントリは警告にリストされます。エージェントを起動する前に出力ファイルに置き換えます。

CSV ファイルを使用すると、任意のソースからデータベースを登録できます。Oracle Enterprise Manager、別の構成管理データベース CMDB または資産管理ツールからエクスポートしたり、手動で組み立てたりできます。ファイルは特定の形式に従う必要があります。EDB Postgres AI エージェント CLI リファレンスの CSV file format を参照してください。

データベースインベントリを含む CSV ファイルを準備します。例

```
#  Database inventory

resource_id,db_type,host,port,db_name,username,password,schema_enabled,poll_interval,
↪tags
ORACLE_PROD_01,oracle,oracle-prod.example.com,1521,PRODPDB,<username>,<password>,true,
↪30s,production|critical
ORACLE_STAGING_01,oracle,oracle-staging.example.com,1521,STAGEPDB,<username>,
```

(次のページに続く)

(前のページからの続き)

```
↪<password>,true,1m,staging
ORACLE_DEV_01,oracle,oracle-dev.example.com,1521,DEVPDB,<username>,<password>,false,
↪5m,development
PG_PROD_01,postgresql,postgres-prod.example.com,5432,appdb,<username>,<password>,true,
↪1m,production|postgres
```

次に、次を実行します。

```
beacon-agent configure-onprem-dbs \
  --source-type csv \
  --input databases.csv \
  --output-yaml ${HOME}/.beacon/beacon_agent.yaml \
  --interactive
```

--interactive を使用して、プレースホルダー値を持つエントリの資格情報の入力を求めます。

Oracle tnsnames.ora ファイルから

Oracle Net Services を使用する場合、tnsnames.ora ファイルを直接渡すことができます。コマンドは、各エントリを解析し、ホスト、ポート、およびサービス名または SID を抽出して DSN を構成します。Oracle 環境で tnsnames.ora を使用している場合、それを直接渡します。tnsnames.ora には資格情報が含まれていないため、実行中に--interactive を使用して資格情報を提供します。

```
beacon-agent configure-onprem-dbs \
  --source-type tns \
  --input /etc/oracle/network/admin/tnsnames.ora \
  --output-yaml ${HOME}/.beacon/beacon_agent.yaml \
  --interactive
```

コマンドは各エントリを解析し、ホスト、ポート、およびサービス名または SID から DSN を構成します。例、このエントリの場合

```
PROD_DB =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = oracle-prod.example.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVICE_NAME = PRODPDB)
    )
  )
```

エージェントは次を生成します oracle://username:password@oracle-prod.example.com:1521/PRODPDB

AWS RDS ディスカバリーから

データベースが AWS RDS または Aurora で実行されている場合、コマンドはローカルファイルを使用せずにそれらを自動的に検出できます。標準の AWS 資格情報チェーン環境変数、`~/.aws/credentials`、または IAM ロールを使用し、1 つまたはすべてのリージョンをスキャンします。エンジンタイプまたはリソースタグでフィルタリングできます。

サポートされている AWS RDS エンジン `postgres`、`aurora-postgresql`、`oracle-ee`、`oracle-ee-cdb`、`oracle-se2`、`oracle-se2-cdb`。

特定のリージョンでサポートされているすべての AWS RDS および Aurora インスタンスを検出するには

```
beacon-agent configure-onprem-dbs \
  --source-type rds \
  --region us-east-1 \
  --output-yaml ${HOME}/.beacon/beacon_agent.yaml \
  --interactive
```

検出を特定のエンジンタイプに絞り込み、リソースタグでフィルタリングするには

```
beacon-agent configure-onprem-dbs \
  --source-type rds \
  --region us-east-1 \
  --rds-engine oracle-ee \
  --rds-engine oracle-se2 \
  --tag-filter Environment=Production \
  --output-yaml ${HOME}/.beacon/beacon_agent.yaml \
  --interactive
```

エージェントの確認と起動

エージェントを起動する前に、出力ファイルでプレースホルダーの資格情報を使用しているエントリがないかを確認します。

```
grep -n "CHANGEME" ${HOME}/.beacon/beacon_agent.yaml
```

プレースホルダーが残っている場合、それらを置き換える 2 つのオプションがあります。

- **すべてのデータベースは同じ資格情報を共有します** — `sed` コマンドを使用して、1 つのステップですべてのプレースホルダーを置き換えます。

```
sed -i "s/CHANGEME_USERNAME:CHANGEME_PASSWORD/<username>:<password>/g" ${HOME}/.
↪beacon/beacon_agent.yaml
```

パスワードが環境変数に保存されている場合、それを直接参照できます。

```
sed -i "s/CHANGEME_USERNAME:CHANGEME_PASSWORD/<username>:$MY_DB_PASSWORD/g" ${HOME}/.
↪beacon/beacon_agent.yaml
```

- **各データベースには異なる資格情報があります** —出力ファイルを手動で編集し、各プレースホルダーを正しい資格情報に置き換えるか、`configure-onprem-dbs` を`--interactive` で再実行して、各データベースを個別にステップ実行します。

残りのプレースホルダーを実際の資格情報に置き換えて、ソースデータベースタイプのガイドに従って構成を検証し、エージェントを起動します。

- **Oracle** — *EDB Postgres AI* **エージェントの実行** を参照してください。
- **自己管理 Postgres** — *EDB Postgres AI* **エージェントの実行** を参照してください。
- **AWS RDS / Aurora** — *EDB Postgres AI* **エージェントの実行** を参照してください。

Using the HM console to assess your applications

Hybrid ManagerHM コンソールには、アプリケーションの SQL ステートメントを評価する機能が含まれています。この評価では、Oracle データベースの移行の範囲で、Oracle アプリケーション SQL ステートメントと EDB Postgres Advanced Server との互換性を評価できます。

HM の移行ツールはデータベーススキーマとデータに焦点を当てていますが、この機能は移行のアプリケーション側に対応します。アプリケーションコードと、それらが存在するデータベースの関係を管理する構造化された方法を提供し、アプリケーションによって実行される SQL クエリーが EDB Oracle 互換環境で正しく機能することを保証します。

概要

データベースの移行がデータベース自分自身に制限されることはほとんどありません、それはアプリケーションの移行です。スキーマが正常に移行された場合でも、アプリケーションコード Java、.NET などまたは外部スクリプト内に埋め込まれた SQL クエリーには、ターゲットデータベースで失敗する Oracle 固有の構文が含まれる場合があります。

これらの非互換性を早期に特定することにより、次のことができます。

- **リスクの削減**アプリケーションのテストフェーズの前に、コードレベルの問題を発見します。
- **正確な計画**アプリケーションのリファクタリングに必要な労力を見積もります。
- **変換の自動化** EDB の変換エンジンを活用して、複雑な Oracle SQL を EPAS 互換の構文に自動的に変換します。

HM のアプリケーションレコード

HM では、 *Creating an application record* ができます。レコードは、データベースを使用してソフトウェアに関するメタデータを保存し、評価が必要な SQL アーティファクトトレースファイル、スクリプト、または抽出されたクエリーを保持する論理コンテナとして機能します。

主な機能には次のものが含まれます。

- **多対多のマッピング** 1 つのアプリケーションを複数のスキーマ/データベースに関連付け、現実のマイクロサービスまたはモノリシックアーキテクチャを反映します。
- **SQL 変換** Oracle DML、DCL、TCL、および DDL を EPAS 互換バージョンに自動的に変換します。

アプリケーションの評価

アプリケーションレコードを作成し、SQL アーティファクトをアップロードすると、HM はアプリケーション評価を自動的に実行します。

1. **メタデータ定義を使用したアプリケーションレコードの作成**アプリケーションの目的、技術スタック、およびアクセスするデータベースを定義します。
2. **SQL インジェスト** SQL アーティファクト Oracle トレース `.trc` 、EDB Query Extractor ファイル、またはプレーン SQL をアップロードします。各ファイルタイプを生成する手順については、*Preparing SQL files for assessment* を参照してください。
3. **評価** システムは EDB PLTE エンジンを使用してクエリーを解析し、非互換性を特定します。次に、*Reviewing application assessment results* を実行できます。

サポートされているインポート形式

HM は、SQL ステートメントを収集するための 3 つの主な方法をサポートしています。

File Type	Description	Extension
Oracle trace	Oracle トレースユーティリティによって生成された標準ファイル。	*.trc*
EDB Query Extractor	EDB スクリプトを使用して Oracle Unified Auditing から抽出された SQL ステートメント。	*.out*
Plain SQL	標準の SQL ステートメントを含む手動スクリプトまたはユーティリティファイル。	*.sql*

各ファイルタイプの準備に関する詳細な手順については、*Preparing SQL files for assessment* を参照してください。

Preparing SQL files for assessment

Hybrid ManagerHM は、アプリケーション評価のために 3 種類の SQL 入力ファイルを受け入れます。それぞれがアプリケーションの SQL を異なる方法でキャプチャし、最適な選択は、アプリケーションのアーキテクチャと使用する Oracle 機能によって異なります。

File type	Best for	Extension
Generating Oracle trace files	Applications where you can enable Oracle SQL tracing against a test or nonproduction database. Provides the most complete picture of runtime SQL, including bind variable values.	*.trc*
Using the EDB Query Extractor	Applications where Oracle tracing isn't available but Unified Auditing is enabled. Captures SQL persistently across restarts.	*.out*

次のページに続く

表 46 – 前のページからの続き

File type	Best for	Extension
Preparing plain SQL files	Existing SQL scripts — initialization scripts, maintenance jobs, or SQL extracted from source code — that don't require any Oracle-side configuration.	*.sql*

既知の制限

- **SQL*Plus commands (plain SQL files only):** HM recognizes the following SQL*Plus commands:
  - `EXIT`
  - `SHOW ERRORS` (`SHOW ERR`)
  - `START`
  - `WHENEVER`
  - `EXEC`
  - `DEFINE`
  - `SET`
  - `COLUMN`
  - `VARIABLE`

HM は、プレーン SQL ファイルを処理する際に、他の SQL *Plus コマンドをサポートしていません。これらは、解析エラーが発生するか、サイレントにスキップされる場合があります。解決するアクティブなセッションがないため、HM は実際のデータベースに対してサポートされているコマンドを検証しません。

- **スキーマコンテキストがありません** HM は現在スキーマコンテキストを使用して、変換された SQL ステートメントを検証しません。アプリケーションの**概要** タブの **スキーマ名**列は、すべてのファイルタイプで空に表示されます。将来のリリースでは、トレースファイルと EDB Query Extractor ファイルのこれを解決する予定です。プレーン SQL ファイルの場合、ファイルはライブ Oracle セッションにリンクしていないため、永続的なままです。
- **MP Project 列** アプリケーションの **Overview** タブの **MP Project** 列は、すべてのファイルタイプで空に表示されます。将来のリリースではこれは解決されるでしょう。
- **プロシージャとファンクションクエリ EDB Query Extractor ファイルのみ** EDB Query Extractor は、ストアドプロシージャまたはファンクション内から実行されたクエリを、評価結果の個別のステートメントとしてキャプチャします。

関連トピック

- *Creating an application record* —アプリケーションレコードを設定し、評価用の SQL ファイルをアップロードします。
- *Reviewing application assessment results* —互換性レポートを解釈し、修正を計画します。
- Application assessment overview —HM がアプリケーション SQL とサポートされているファイル形式を評価する方法を理解します。

Generating Oracle trace files

Oracle SQL トレースファイルは、バインド変数値や待機イベントを含む、セッションまたはユーザーが実行したすべての SQL 文を記録します。これらは、Hybrid ManagerHM に、アプリケーションのランタイム SQL の最も完全な画像を提供します。

前提条件

- Oracle データベースへの DBA レベルのアクセス権がある。
- Oracle 診断先 `diagnostic_dest` に十分なディスク領域がある。トレースは、特にビジーなアプリケーションの場合、大量のデータをすばやく生成できます。

SYSDBA としてデータベースに接続する

アプリケーションデータが存在するプラガブルデータベース PDB に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<PDB_NAME> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCLPDB1 as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCL as sysdba
```

ログオントリガーの作成

ログオントリガーは、特定のデータベースユーザーが接続するたびに、トレースを自動的に開始します。`YOUR_APP_USER` を、アプリケーションが接続する Oracle スキーマに置き換えます。

```
CREATE OR REPLACE TRIGGER trace_user_logon
AFTER LOGON ON DATABASE
BEGIN
    IF (USER = YOUR_APP_USER) THEN
        DBMS_SESSION.SET_IDENTIFIER(YOUR_APP_USER);
        EXECUTE IMMEDIATE ALTER SESSION SET TRACEFILE_IDENTIFIER = APP_TRACE;
        DBMS_SESSION.SESSION_TRACE_ENABLE(waits => TRUE, binds => TRUE);
    END IF;
END;
/
```

複数のユーザーをトレースするには、`IN` リストを使用します。

```
CREATE OR REPLACE TRIGGER trace_user_logon
AFTER LOGON ON DATABASE
BEGIN
    IF (USER IN (APP_USER_1, APP_USER_2)) THEN
        EXECUTE IMMEDIATE ALTER SESSION SET TRACEFILE_IDENTIFIER =  || USER || _TRACE;
        DBMS_SESSION.SESSION_TRACE_ENABLE(waits => TRUE, binds => TRUE);
    END IF;
END;
/
```

トレースがアクティブであることの確認

アプリケーション全体を実行する前に、トリガーが機能していることを確認してください。YOUR_APP_USER として接続して実行します。

```
SELECT sysdate FROM dual;
```

次に、トレースディレクトリで新しいまたは更新された.trc ファイルを確認します。表示されている場合、トレースは正常に機能しています。

アプリケーションの実行

トレースを確認したら、Oracle データベースに対してアプリケーションを実行します。代表的なカバレッジを取得するには、完全なテストスイートを実行するか、アプリケーションが運用環境で使用する主要なワークフローを演習します。

トレースファイルの収集

トレースディレクトリを見つけるには、次を実行します。

```
SELECT value FROM v$diag_info WHERE name = Diag Trace;
```

これにより、Oracle のトレースディレクトリへのフルパスが返されます。これは、次の構造に従います。

```
<ORACLE_BASE>/diag/rdbms/<DB_NAME>/<INSTANCE_NAME>/trace/
```

例 /u01/app/oracle/diag/rdbms/orcl/orcl/trace/

そのディレクトリで実行されているアプリケーションのトレースファイルは、インスタンスと TRACEFILE_IDENTIFIER たとえば orcl_ora_12345_APP_TRACE.trc にちなんで付けられた名前です。これらのファイルを収集し、HM にアップロードします。

> **注釈**
>
> このステップの後、ログオントリガーはアクティブなままです。トレースをキャプチャする必要がなくなったときに削除して、今後の接続でトレースファイルの蓄積を回避します。

```
DROP TRIGGER trace_user_logon;
```

- *Preparing SQL files for assessment* —サポートされているすべての SQL ファイルタイプの概要。
- *Using the EDB Query Extractor* — Oracle Unified Auditing を使用した代替キャプチャ方法。
- *Preparing plain SQL files* —Oracle 側の構成を使用せずに既存の SQL スクリプトをアップロードします。
- *Creating an application record* — トレースファイルをアップロードし、評価を開始します。

Using the EDB Query Extractor

EDB Query Extractor は、インメモリで再起動ごとにクリアできる V$SQL とは異なり、監査レコードの永続的ストレージを提供する Oracle Unified Auditing を使用して、アプリケーションで実行された SQL をキャプチャします。エクストラクターは、キャプチャした SQL を Hybrid ManagerHM の評価エンジンと互換性のある.out ファイルにエクスポートします。

> **注釈**
>
> EDB Query Extractor は、現在リクエストに応じて利用できます。 EDB Support に連絡して、スクリプトを入手します。セルフサービスのダウンロードは、将来の HM リリースに含まれる予定です。

前提条件

- **Oracle Unified Auditing はデータベースで有効になっています。** Oracle 12c 以降では、統合監査は混合モードでデフォルトで利用できます。データベースでアクティブであることを確認するには、次を実行します。

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = Unified Auditing;
```

TRUE の結果は、純粋な統合監査が有効になっていることを意味します。FALSE の結果は、混合モードがアクティブであることを意味します。これは引き続き統合監査をサポートし、このワークフローには十分です。統合監査を利用できない場合、

Oracle documentation on enabling Unified Auditing を参照してください。

- **あなたまたは DBA は、 ``AUDIT SYSTEM`` 特権を持っています。**この権限は、以下の手順で監査ポリシーを作成および有効にするために必要です。SYSDBA ユーザーはデフォルトでこれを持っています。SYSDBA として接続していない場合は、AUDIT SYSTEM 特権を付与するか、あなたに代わってポリシー設定を実行するように DBA に依頼してください。

データベースへの接続

アプリケーションデータが存在するプラガブルデータベース PDB に接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<PDB_NAME> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCLPDB1 as sysdba
```

サービス名または SID を使用してデータベースインスタンスに接続します。

```
sqlplus sys/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE> as sysdba
```

例

```
sqlplus sys/password123@//localhost:1521/ORCL as sysdba
```

監査ポリシーの設定

1. すべての SQL アクションをキャプチャする監査ポリシーを作成します。

```
CREATE AUDIT POLICY trace_style_policy
    ACTIONS ALL;
```

1. アプリケーションのデータベースユーザーのポリシーを有効にします。YOUR_APP_USER を、アプリケーションが接続する Oracle スキーマに置き換えます。パフォーマンスへの影響と出力ボリュームを制限するために、すべてのユーザーではなく特定のユーザーに対してポリシーを有効にすることを強くお勧めします。

```
AUDIT POLICY trace_style_policy BY YOUR_APP_USER;
```

監査ポリシーが機能していることの確認

アプリケーションを実行する前に、Oracle が監査対象ユーザーの SQL を記録していることを確認してください。

1. YOUR_APP_USER としてデータベースに接続します。

```
sqlplus YOUR_APP_USER/<PASSWD>@//<HOST>:<PORT>/<PDB_NAME>
```

```
sqlplus YOUR_APP_USER/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE>
```

1. 一対のテストステートメントを実行して監査レコードを生成します。

```
CREATE TABLE junk(a int);
DROP TABLE junk;
```

1. SYSDBA として再接続し、 UNIFIED_AUDIT_TRAIL を照会して、レコードがキャプチャされたことを確認します。

```
SELECT SQL_TEXT, ACTION_NAME, STATEMENT_ID, SESSIONID, EVENT_TIMESTAMP
FROM UNIFIED_AUDIT_TRAIL
WHERE DBUSERNAME = UPPER(YOUR_APP_USER)
  AND (OBJECT_SCHEMA IS NULL
```

(次のページに続く)

(前のページからの続き)

```
      OR OBJECT_SCHEMA NOT IN (SYS, SYSTEM, AUDSYS, USERENV, GSMADMIN_INTERNAL))
  AND SQL_TEXT IS NOT NULL
  AND SESSIONID IS NOT NULL
  AND STATEMENT_ID IS NOT NULL;
```

CREATE TABLE および DROP TABLE ステートメントが結果に表示される場合、ポリシーは正常に機能しています。

EDB Query Extractor 特権の付与

EDB Query Extractor を実行する前に、監査対象ユーザーに監査証跡を読み取るために必要な特権を付与します。

```
GRANT AUDIT_VIEWER TO YOUR_APP_USER;
GRANT SELECT ON UNIFIED_AUDIT_TRAIL TO YOUR_APP_USER;
GRANT SELECT ON sys.v_$instance TO YOUR_APP_USER;
```

アプリケーションの実行

監査ポリシーを確認したら、アプリケーションまたはテストスイートを実行します。Oracle は、監査対象ユーザーが実行したすべての SQL ステートメントを UNIFIED_AUDIT_TRAIL に記録します。

SQL の抽出

EDB Query Extractor スクリプト edb_queries_extractor.sql を YOUR_APP_USER として実行して、キャプチャされた SQL を UNIFIED_AUDIT_TRAIL から.out ファイルにエクスポートします。

```
sqlplus YOUR_APP_USER/<PASSWD>@//<HOST>:<PORT>/<PDB_NAME>
```

```
sqlplus YOUR_APP_USER/<PASSWD>@//<HOST>:<PORT>/<SID_OR_SERVICE>
```

次に、スクリプトを実行します。

```
@/path/to/edb_queries_extractor.sql
```

スクリプトは、次の情報の入力を求めます。

- **出力ファイルの場所** — .out ファイルが書き込まれるディレクトリ。デフォルトは現在のディレクトリです。
- **Username** —監査レコードをエクスポートする Oracle ユーザー。現在のユーザーのデフォルト。
- **監査ポリシー** —フィルタリングする統合監査ポリシー名のコンマ区切りリスト。すべてのポリシーを含めるには、空白のままにします。
- **クライアントプログラムまたは識別子** —アプリケーションのソースコードで設定されたクライアントプログラム名またはクライアント識別子たとえば MyPythonApp 。すべてを含めるには空白のままにします。

- **開始タイムスタンプ** —キャプチャウィンドウの開始、DD-MM-YYYY HH24:MI:SS 形式。デフォルト 01-01-1970
- **終了タイムスタンプ** —キャプチャウィンドウの終了、DD-MM-YYYY HH24:MI:SS 形式。デフォルト現在の時間。

スクリプトは、指定されたディレクトリたとえば/tmp/YOUR_APP_USER_260414133704.out の USERNAME_YYMMDDHHMMSS.out という名前のファイルに出力を書き込みます。

監査ポリシーの無効化

抽出後、監査ポリシーを無効にして新しいレコードのキャプチャを停止します。

```
NOAUDIT POLICY trace_style_policy BY YOUR_APP_USER;
```

生成された.out ファイルを評価のために HM にアップロードします。

関連トピック

- *Preparing SQL files for assessment* —サポートされているすべての SQL ファイルタイプの概要。
- *Generating Oracle trace files* — Oracle SQL トレースを使用した代替キャプチャ方法。
- *Preparing plain SQL files* —Oracle 側の構成を使用せずに既存の SQL スクリプトをアップロードします。
- *Creating an application record* —エクストラクター出力をアップロードし、評価を開始します。

Preparing plain SQL files

すでに SQL スクリプトたとえば、データベースの初期化、メンテナンスジョブ、またはソースコードファイルに埋め込まれた SQL に使用されるスクリプトがある場合は、Oracle 側の構成を使用せずにプレーンの.sql ファイルとして直接アップロードできます。

フォーマット要件

プレーン SQL ファイルを正しく解析するには、ハイブリッドマネージャー HM の次のフォーマットルールに従う必要があります。

- すべての SQL ステートメントはセミコロン; で終わる必要があります。
- すべての PL/ SQL ブロック無名ブロック、ストアドプロシージャ、トリガーは、独自の行でスラッシュ/ で終わり、前に空白行が必要です。
- ファイルは UTF-8 エンコードを使用する必要があります。

例

```
SELECT employee_id, first_name FROM employees WHERE department_id = 10;

BEGIN
    UPDATE employees SET salary = salary * 1.1 WHERE department_id = 10;
```

(次のページに続く)

(前のページからの続き)

```
    COMMIT;
END;

/
```

関連トピック

- *Preparing SQL files for assessment* —サポートされているすべての SQL ファイルタイプの概要。
- *Generating Oracle trace files* — Oracle SQL トレースを使用して、ランタイム SQL をキャプチャします。
- *Using the EDB Query Extractor* — Oracle Unified Auditing を使用してランタイム SQL をキャプチャします。
- *Creating an application record* — SQL ファイルをアップロードし、評価を開始します。

Creating an application record

アプリケーションレコードは、ハイブリッドマネージャー HM がソフトウェアアプリケーションを表す方法です。アプリケーションを依存するソースデータベースとスキーマにマップし、アップロードする SQL ファイルを保持して EDB Postgres Advanced Server との互換性を評価します。アプリケーションレコードの作成は、アプリケーション評価ワークフローの最初のステップです。

前提条件

アプリケーションレコードを作成する前に、以下を確認してください。

- アプリケーションを作成する HM プロジェクトにアクセスできること。
- あなたのユーザーロールには、その HM プロジェクトの `migration_applications` 権限が含まれています。
- SQL アーティファクトをアップロードする準備ができました。受け入れられるファイルタイプについては、**サポートされているインポート形式** を参照してください。最初に生成する必要がある場合は、*Preparing SQL files for assessment* を参照してください。

アプリケーションレコードの作成

HM コンソールは、アプリケーションの一般的なメタデータを定義し、移行データベースに関連付ける手順をガイドします。

1. HM で、プロジェクトを開きます。
2. 左側のナビゲーションで、 **Migrate** > **Applications** に移動します。
3. **+新しいアプリケーション** を選択します。
4. アプリケーションのメタデータを入力します。

Field	Required	Description
Name	Yes	アプリケーションの説明的な名前。この名前は、**Applications** テーブルと詳細ページに表示されます。
Description	No	アプリケーションの機能またはそのビジネス目的の簡単な説明。
Target database version	Yes	SQL が評価される EPAS バージョン、正しい互換性ルールが適用されることを確認します。このアプリケーションにアップロードするすべてのファイルのデフォルトになるため、アップロード時に再度選択する必要はありません。
Programming language	Yes	アプリケーションがデータベースと対話するために使用するプライマリ言語またはフレームワークたとえば、Java、Go (sqlx)、Python など。

1. アプリケーションをソースデータベースにリンクする場合は、オプションで **Next: Databases (Optional)** ステップに進みます。
2. オプションで、含める各データベースの横にあるチェックボックスを選択します。ページ全体で複数のデータベースを選択できます。

> **注釈**
>
> ソースデータベースのリンクはオプショナル。リンクしない場合、HM は Oracle 19c をソースデータベースバージョンとして想定します。 SQL に Oracle 21c 固有の構文が含まれる場合、これは評価精度に影響を与える可能性があります。最初に Oracle データベースを登録するには、 *Configuring the EDB Postgres AI agent to register your Oracle database* を参照してください。 .. Note:: 1. **Create application** を選択します。

HM はアプリケーションレコードを作成し、アプリケーションの詳細ページに移動します。

SQL ファイルのアップロード

アプリケーションレコードを作成した後、評価のために SQL アーティファクトをアップロードします。

1. アプリケーションの詳細ページで、[**概要**] タブに移動します。
2. **SQL File Assessments** セクションで、 **Upload New File(s)** を選択します。
3. ファイルをアップロードエリアにドラッグアンドドロップするか、ファイルピッカーを使用して選択します。1 回のアップロード操作で最大 20 のファイルを選択できます。個々のファイルは 10MB 以下である必要があります。ファイルが制限を超える場合、アップロード時にエラーメッセージが表示され拒否されます。
4. ファイルのリストを確認します。事前検証エラーはここに表示されます。
5. **Next: Databases** を選択して続行します。
6. アップロードされたファイルごとに、次の設定を構成します。

Field	Required	Description
Source database	No	ファイルの SQL が抽出された移行データベース。ファイルをソースデータベースにリンクすると、HM はクエリを特定のスキーマにマップして、より正確な評価を行うことができます。
Target database version	Yes	たとえば、EPAS 18 に対して評価する EPAS バージョン。このアクションは、アプリケーションレコードレベルですべての評価対象データベースのターゲットバージョンを事前入力します。
Oracle User UIDs	No	オブジェクトが SQL ファイルに表示される 1 つ以上の Oracle ユーザー ID。評価の範囲を関連するスキーマ所有者に設定するために使用されます。

1. **確認**を選択してファイルをアップロードし、評価のためにキューに入れます。

ファイルをアップロードすると、HM は自動的に評価を実行します。*Reviewing application assessment results* を参照して、互換性レポートの解釈、互換性のないクエリーの特定、およびアプリケーションを移行する前に修復を計画する方法を理解します。

関連トピック

- *Preparing SQL files for assessment* — Oracle トレースファイル、EDB Query Extractor 出力を生成するか、アップロード用のプレーン SQL ファイルを準備します。
- *Reviewing application assessment results* —評価出力を解釈し、互換性のないクエリーを調査し、アプリケーションの移行前に修復を計画します。
- Application assessment overview —HM がアプリケーション SQL とサポートされているファイル形式を評価する方法を理解します。
- マイグレーション —アプリケーション評価と並行して、スキーマレベルおよびデータベースレベルの評価を確認します。

Reviewing application assessment results

Creating an application record 後、Hybrid ManagerHM は自動的に評価を実行し、互換性レポートを生成します。このページでは、アプリケーションを移行する前に、これらの結果を読み取り、それらを使用して非互換性を修復する方法について説明します。

SQL Assessment タブを開く

1. HM で、 **移行 > アプリケーション** に移動します。
2. 確認するアプリケーションを選択します。
3. アプリケーションの詳細ページで、 **SQL Assessment** タブを選択します。

[**SQL Assessment**] タブは、アプリケーションにアップロードされたすべての SQL ファイルの結果を集計します。

評価の詳細の確認

SQL Assessment タブの上部にある **Assessment Details** セクションには、評価の実行に関する機能互換性の概要と主要なメタデータが提供されます。

機能互換性カード

互換性カードは、SQL ファイルの Oracle 固有の機能が互換性ステータスごとにどのように内訳を示します。

Card	Description
Compatible features	EDB Postgres Advanced Server で変更なしで動作する機能。
Features with available workaround	ネイティブ互換性はありませんが、適用できる既知の確立された回避策がある機能。
Features with new workaround	新しい回避策が特定された機能。採用する前に追加のレビューが必要になる場合があります。
Features with no workaround	自動的に変換できず、既知の回避策がない機能。これらの項目は手動での修正が必要です。

評価メタデータ

カードの下のメタデータストリップは、評価実行のコンテキストを示します。

Field	Description
Application name	アプリケーションレコードの名前。
Source database(s)	このアプリケーションにリンクされている移行データベース。
Target database version(s)	評価が実行された EPAS バージョン。
Total input files provided	このアプリケーションにアップロードされた SQL ファイルの数。
Total queries parsed	すべてのファイルで正常に解析された SQL ステートメントの数。
Total queries with suggested repairs	HM が提案された変換を生成したクエリの数。
Last assessment timestamp	最新の評価が完了したとき。

評価レポートのダウンロード

完全な評価を共有またはアーカイブするには、**レポートのダウンロード**を選択します。レポートには、ファイル全体のすべての結果の完全な概要が含まれており、オフラインレビュー、チームの引き継ぎ、または移行計画のドキュメントに役立ちます。

変換されたクエリーのダウンロード

Download Transformed Queries を選択して、アプリケーションのすべての変換クエリを単一のアーカイブとしてエクスポートします。この出力は次の目的で使用できます。

- HM が EPAS の特定のクエリをどのように適応させたかを検査します。
- Feed は、移行が完了する前にステートメントをテスト環境に変換しました。
- 開発チームとレビュープロセスをシードします。

互換性のない機能のレビュー

非互換機能 テーブルは、SQL ファイルにある EPAS と直接互換性のない Oracle 固有の機能を識別します。個々のクエリを掘り下げる前に、このテーブルを使用して、コードベース全体の問題のカテゴリを理解します。

Column	Description
Feature	互換性のない Oracle 固有の構文、ファンクション、または構成体。
File	この機能が表示される SQL ファイル。
Oracle User UIDs	アップロード時に提供される場合、この機能に関連付けられている Oracle スキーマ所有者。
Workaround Status	この機能の現在の修復状態。
Queries	この機能を使用するファイル内のクエリーの数。
Occurrences	ファイル内のすべてのクエリーにわたってこの機能が出現する合計回数。
Query ID	フィーチャが見つかった特定のクエリーへの参照。

互換性のないクエリーの確認

In stable Queries テーブルは、アプリケーションを EPAS に対して実行する前に変更が必要な SQL ステートメントのクエリごとの内訳を提供します。このテーブルは、アプリケーション側の修復作業の範囲を理解するための主要なツールです。

各行は、1 つ以上の非互換性を持つ単一の SQL ステートメントを表します。ここにリストされているクエリーはそのままでは正しく実行されず、アプリケーションのカットオーバーの前にアクションが必要です。

Column	Description
Query ID	評価結果のクエリの安定した識別子。
File	このクエリーを含む SQL ファイル。
Oracle User UIDs	アップロード時に提供される場合、このクエリーに関連付けられている Oracle スキーマ所有者。
Query	SQL ステートメントのプレビュー。
# of Incompatible Features	このクエリーで検出された個別の互換性のない機能の数。
Occurrences	アップロードされたファイル全体でこのクエリが表示される回数。

クエリを展開して変換と修復ガイダンスを表示する

HM がクエリーを修復できた場合は、そのクエリー行の + 記号を使用して展開します。展開されたパネルには次のものが表示されます。

- **変換されたクエリ** —元の SQL の HM の EPAS 互換バージョン。この出力は、アプリケーションコードの更新に使用できる代替案です。
- **Repair handlers** 変換中に行われた特定の変更に関する詳細。
- どのような Oracle コンストラクトが変更されたか。
- どの変換ルールが適用されました。
- 手動の注意を必要とする残りのステップ。

互換性のないすべてのクエリーを自動的に変換できるわけではありません。HM が変換を生成できない場合、修復ハンドラーノートに変更が必要なものが記載されているため、アプリケーションコードで手動で修正を行うことができます。

アプリケーションの移行前の非互換性の修正

評価結果を確認した後、カットオーバーの前に、その結果を使用してアプリケーションを適応させます。一般的な修復ワークフローは次のとおりです。

1. **互換性のない機能を使用したトリアージ。 互換性のない機能**テーブルから始めて、問題のカテゴリを理解します。 **Workaround Status** 列を使用して、優先順位を付けます。 **None / Needs Investigation** とマークされた機能には既知の回避策がなく、最も注意が必要です。一方、**Available** および **New Workaround** エントリは修復パスが確立されています。少数の機能タイプが互換性のないクエリの大部分を占めることがよくあります。これらのパターンを特定すると、クエリごとにではなく一括で修復を計画できます。
2. **変換されたクエリーを確認します。 In stable Queries** テーブルの互換性のないクエリーごとに、行を展開して、元の Oracle SQL と変換された EPAS バージョンを比較します。HM の変換が正しい場合、アプリケーションコードに直接採用できます。
3. **自動変換なしでクエリを処理します。**変換が利用できない場合、展開された行の **Repair Handler Applied** フィールドは、試行された内容とまだ手動の注意が必要なものを説明します。それに応じてアプリケーションコードを更新し、非運用 EPAS 環境に対して再テストします。
4. **非実稼働環境でテストします。**実稼働を移行する前に、移行したデータとスキーマを使用して非実稼働 EPAS クラスターに対して更新されたアプリケーションを実行します。展開された各クエリ行の**# Leftover 非互換性フィーチャ**数を使用して、すべての問題が解決されたことを確認します。カットオーバーの前に、残っている実行時エラーを解決します。

> **注釈**
>
> 本番のカットオーバーウィンドウの前に SQL の非互換性に対処すると、リスクが削減され、メンテナンスウィンドウが短縮されます。これらの手順のシーケンスに関するガイダンスについては、Oracle to HM-managed migration guide を参照してください。

関連トピック

- *Preparing SQL files for assessment* — Oracle トレースファイル、EDB Query Extractor 出力を生成するか、アップロード用のプレーン SQL ファイルを準備します。
- *Creating an application record* —アプリケーションレコードを設定し、評価用の SQL ファイルをアップロードします。
- Application assessment overview —HM がアプリケーション SQL とサポートされているファイル形式を評価する方法を理解します。
- マイグレーション—アプリケーションレベルの所見とともにスキーマレベルおよびデータベースレベルの評価を確認します。

Migrating schemas

スキーマの移行は、データベースの移行において重要な手順です。プロセスの複雑さはデータベースシステムによって異なるため、使用するツールはソースデータベースによって異なります。

Oracle ソース

Oracle ソースの場合、Oracle と PostgreSQL の違いにより、スキーマの移行はより複雑になります。このプロセスには、直接互換性のない構文、データ型、およびファンクションを処理するための専用の変換ステップが必要です。

このプロセスを合理化するために、Hybrid ManagerHM は統合された移行ポータルを提供します。このサービスは、Oracle スキーマの評価を自動化し、非互換性のフラグを立て、複雑なスキーマの問題の解決に役立つ AI コパイロットが含まれています。

Oracle schema migration

Postgres ソースデータベース

Postgres ソース PostgreSQL、 EDB Postgres Advanced Server、または EDB Postgres Extended Server オンプレミスまたはクラウドディストリビューションからスキーマを移行するのは、より簡単なプロセスです。ソースと HM 管理の宛先は両方とも同じテクノロジーに基づいているため、スキーマに互換性があり、変換の必要がありません。

このタスクには標準の PostgreSQL ツールを使用できます。Migration ToolkitMTK または pg_dump を使用して、ソースデータベースからスキーマを手動でエクスポートし、HM 管理の宛先クラスターにインポートすることをお勧めします。

3. Postgres クラスターを復元する

Using Migration Portal to assess, convert, and migrate Oracle schemas

移行ポータルは、Oracle データベーススキーマを評価、分析、変換、および EDB Postgres Advanced Server に移行するのに役立ちます。web-based service として、または Hybrid ManagerHM のアプリケーションとして利用できます。

ハイブリッドマネージャー移行ポータル vs Web ベースの移行ポータル

移行ポータルの HM 展開の一部の機能は一意であるか、Web ベースのバージョンと機能が異なります。

AI コパイロット

> **注釈**
>
> HM の移行ポータルでこの機能を手動で有効にする必要があります。 Web ベースのバージョンでは、すぐに使えるように含まれています。

AI Copilot は、移行ポータルでのスキーマ評価と変換中に発生する可能性のある問題の解決を支援する AI 駆動のチャットボットツールです。サードパーティの AI サービスを使用し、EDB の製品ドキュメントとサポートナレッジベースの両方でトレーニングされて、ユースケース固有のソリューションを提供します。

AI コパイロットは、HM ではデフォルトで有効になっていません。

- 移行ポータルクエリーに AI コパイロットを使用するには、HM 管理者に enable and configure it with the AI/LLM of your organization を依頼します。
- 詳細については、 AI Copilot を参照してください。

プロジェクトコラボレーション

> **注釈**
>
> プロジェクトコラボレーション機能は、ハイブリッドマネージャー移行ポータルでのみ利用できます。

移行ポータルは、共同 DDL 編集をサポートし、同じ HM プロジェクト内の複数のユーザーが同じスキーマファイルで同時に作業できるようにします。あなたが編集している DDL を別のユーザーが変更すると、あなたの変更は競合解決ビューをトリガーします。このビューには、他のユーザーの更新が **Update** ビューに表示され、自分の編集が **Target** ビューに表示されます。次に、 **Use Updated** を選択して他のユーザーの修正を採用するか、 **Reassess** を選択して独自の変更を適用および保存し、以前の更新を上書きできます。このアプローチにより、効率的な同時スキーマ互換性調整が保証されます。

ポータルアクセス

移行ポータルインターフェイスへのアクセスは、使用しているバージョンによって異なります。

- EDB portal を介して Web ベースの移行ポータルにアクセスします。
- HM コンソールから HM に組み込まれている移行ポータルにアクセスします。
- *Assessing schemas for HM projects* を使用して、 EDB Postgres AI エージェントによってロードされたスキーマを移行します。
- *Assessing schemas for external projects* を使用して、外部提供されたスキーマを移行します。

ユーザー権限とロールベースのアクセス

- Web ベースの移行ポータルは、各 EDB アカウントに専用のワークスペースを提供します。つまり、他のユーザーとプロジェクトを共有することはできません。
- HM に組み込まれている移行ポータルは、プロジェクトスコープの共同アクセスを提供します。特定の HM プロジェクトへのアクセスを持つユーザーは、それに関連付けられている移行ポータルプロジェクトへのアクセスを取得します。
- HM では、移行ポータルのユーザー権限は HM を介して直接管理されます。たとえば、Viewer 権限を持つユーザーは、スキーマを移行したり、DDL を再評価したりすることはできません。

Accessing Migration Portal

Hybrid ManagerHM の移行ポータルは、使用する方法に応じて異なる方法でアクセスする 2 つの異なるワークスペースを提供します。

プロジェクトスコープのワークスペースとユーザーのワークスペース

HM の展開には、2 種類のワークスペースを持つ移行ポータルのビルトインインスタンスが含まれます。

- 移行ポータルの **プロジェクトスコープワークスペース** は、 *HM で AWS RDS 展開をモニタするための EDB Postgres AI エージェントの設定*によってロードされたスキーマの HM コンソールで作成された移行ポータルプロジェクトを管理する場所です。

この共有ポータル環境は、移行ポータルプロジェクトを HM プロジェクトと直接調整します。この共有ポータルでは、自分と同じ HM プロジェクトの一部である他のユーザーが作成したすべての移行ポータルプロジェクトを表示できます。ただし、他の HM プロジェクトで実行された移行から移行ポータルプロジェクトを表示することはできません。

- 移行ポータルの **ユーザーのワークスペース**は、分析、変換、および移行のために手動でアップロードする外部スキーマの移行プロジェクトを作成および管理する場所です。

これは共有環境ではありません。このワークスペースで作成された移行ポータルプロジェクトには、自分だけがアクセスできます。

この分離により、移行ポータルは、さまざまな種類と規模の移行作業に対応できます。

- 組織化された大規模なエンタープライズ主導のプロジェクトプロジェクトスコープ
- より柔軟、小規模、または個人主導の移行ユーザーのワークスペース

アクセス

移行ポータルには、さまざまな目的に応じた 2 種類のワークスペースがあります。

- EDB Postgres AI エージェントによって HM にロードされたスキーマのプロジェクトを作成および管理するには、*Assessing schemas for HM projects* を参照してください。これは、移行ポータルへのプロジェクトスコープのアクセスを使用します。
- 外部提供された DDL ファイルを手動でアップロードし、移行ポータルでスキーマを評価するには、*Assessing schemas for external projects* を参照してください。これには、ユーザーのワークスペースから移行ポータルへのアクセスが使用されます。

Assessing schemas for HM projects

移行ポータルの協調的な **プロジェクトスコープのワークスペースとユーザーのワークスペース** を使用して、データベースの移行のコンテキストでスキーマを変換および移行します。HM は、HM コンソールを使用して移行ポータルプロジェクトを作成できる統合を提供します。このプロジェクトでは、データベースの移行の実行中に、 *HM で AWS RDS 展開をモニタするための EDB Postgres AI エージェントの設定* によってプリロードされたソースデータベーススキーマを追加できます。

Oracle データベースのみが移行ワークフローへの移行ポータルの統合を含みます。スキーマは宛先データベースに移行する前に互換性評価と多くの場合変換が必要です。

外部提供された DDL スキーマファイルをアップロードして評価するには、*use the user's workspace* 。

前提条件

- HM を使用した *Register an Oracle database*
- **オプション**移行ポータルでのスキーマの非互換性の解決を促進するには、HM 管理者は enable and configure the AI Copilot with the AI/LLM of your organization する必要があります。

Migration Portal へのアクセス

Oracle データベースの場合のみ、移行ポータル統合を使用してスキーマを評価および移行します。

1. **Estate** ページで、 **Migrations** タブを選択します。
2. データベースの詳細ビューを開くには、移行するデータベースの名前を選択します。

1 つ、複数、またはすべてのスキーマのスキーマ移行プロジェクトを既に作成している場合、HM コンソールの**移行ポータルプロジェクト**列にプロジェクトの名前のリンクが表示されます。リンクを選択して、そのプロジェクトのすべてのスキーマを含む移行ポータルワークスペースを開きます。

スキーマ移行のプロジェクトをまだ作成していない場合は、グローバルの **New Migration Portal Project** ボタンまたはスキーマスコープの **+** 記号を選択します。この選択により、*create a Migration Portal* プロジェクトへの新しいワークフローが開始されます。

Migration Portal プロジェクトの作成

EDB Postgres AI エージェントを介して Oracle データベースをハイブリッドマネージャーに登録した場合、すべてのデータベーススキーマと依存関係に対して移行ポータルプロジェクトが**自動的に作成されます**。登録されたデータベースごとに `beacon-agent.yaml` で確立された `resource_id` ごとにプロジェクトを作成します。登録後に追加されたスキーマは、移行ポータルプロジェクトに反映されないことに注意してください。

移行ポータルが登録したデータベースのプロジェクトを自動的に作成しなかった場合、プロジェクトを手動で作成します。これは、次の場合に当てはまります。

- 取り込まれた DDL のサイズがデフォルト制限の 300 MB を超えています。
- 取り込まれた DDL のサイズが *Migration Portal* **プロジェクトの最大サイズのカスタマイズ** を超えています。
- Migration Portal プロジェクトの自動作成を無効にしました。

Migration Portal プロジェクトの手動作成

自動プロジェクト作成が無効または失敗した場合、移行ポータルプロジェクトを手動で作成します。

1. **Estate** ページで、 **Migrations** タブを選択します。
2. データベースの詳細ビューを開くには、移行するデータベースの名前を選択します。
3. データベースの詳細ビューで、 **Schemas** タブを選択します。
4. **New Migration Portal Project** または、移行するスキーマの 1 つの横にあるプラス記号を選択します。

> **Tip**
>
> **Dependencies**列には、現在のスキーマが依存する他のスキーマが表示されます。スキーマとそのすべての依存関係を単一の移行ポータルプロジェクトにグループ化して、参照されるすべてのオブジェクトが宛先データベースに存在することを確認することをお勧めします。

1. ポップアップウィンドウで、スキーマの評価、変換、および移行をホストする移行ポータルプロジェクトの詳細を入力します。**続行** を選択します。
2. この移行ポータルプロジェクトに追加するスキーマをすべて選択します。EDB Postgres AI エージェントが正常に取り込んだスキーマと、別の移行ポータルプロジェクトの一部になっていないスキーマのみを選択できます。

> **注釈**
>
> 既存のプロジェクトにスキーマを後で追加することはできません。ただし、別のプロジェクトを使用して、他のスキーマを同じ宛先データベースに移行できます。

1. **Create New Project** を選択します。

これにより、バックグラウンドで、移行ポータルプロジェクトが作成され、分析のためにソーススキーマがロードされます。

HM がスキーマのロードと移行ポータルでのプロジェクトの作成が終了すると、プロジェクトの名前のリンクが追加されます。リンクを選択して Migration Portal を開きます。

Migration Portal での作業

選択したスキーマは、移行ポータルによってアップロードされ、評価されました。AI Copilot または QuickHelp を使用して、非互換性の解決に役立ちます。

1. すべての互換性問題を解決したら、[**Migrate to**] を選択します。
2. **移行** オプションを選択します推奨。
3. 移行するスキーマを選択しますが、データベースの下のオブジェクトのリストから**コンストレイン**をクリアします。 **Next** を選択します。

> **注釈**
>
> コンストレインを無効にする理由およびその他の考慮事項の詳細については、*Migrating Postgres schemas* を参照してください。

1. **Connect to your cluster** ページで、宛先として使用するデータベースクラスターのホスト名を入力します。

> **注釈**
>
> HM 管理データベースクラスターに移行する場合、**クラスター**ページから読み取り/書き込みホストおよびその他の接続データを見つけることができます。クラスターを選択し、**Connect**を選択します。

1. **Next** を選択し、スキーマが正常にインポートされるまで待ちます。
2. ターゲットデータベースクラスターに接続し、スキーマが移行されたことを確認します。

Migration Portal プロジェクトの削除

移行ポータルのプロジェクトスコープのワークスペースを使用する場合、統合ポータルではなく、HM コンソールを介してのみプロジェクトを削除できます。

1. **Estate** ページから、 **Migrations** タブを選択します。
2. 移行したデータベースの名前を選択します。
3. データベースの詳細ビューで、 **Schemas** タブを選択します。スキーマと割り当てられた移行ポータルプロジェクトのリストが表示されます。
4. 削除する移行ポータルプロジェクトに割り当てられたスキーマを見つけ、同じ行の**+** プラス記号を選択します。

> **注釈**
>
> プロジェクトを使用して複数のスキーマを評価している場合、割り当てられたスキーマのいずれかからプロジェクトを削除すると、すべてのスキーマのプロジェクトが削除されます。

5. 展開されたビューの **Actions** の下で、ゴミ箱アイコンを選択します。
6. プロジェクトを完全に削除するには、 **delete project** と入力し、**confirm Delete** を選択します。

ユーザーのワークスペースに切り替える

移行ポータルでスキーマの作業中に、手動 DDL アップロードのために **プロジェクトスコープのワークスペースとユーザーのワークスペース** に切り替えることができます。

これを行うには、 **ユーザーのワークスペースに切り替える** を選択します。

ただし、このワークスペースには、プロジェクトの作成中に選択したスキーマは表示されません。これは、タスクをすばやく切り替える方法としてのみ機能します。詳細については、*Managing external schemas for*

external DDL を参照してください。

関連トピック

- 宛先データベースに変換されたスキーマの

Perform an online or offline migration 。

- Migration Portal documentation ポータルの使用方法に関する推奨事項とガイドラインについては。
- *Controlling Migration Portal project creation* プロジェクトの最大サイズをカスタマイズするか、プロジェクトの自動作成を無効にする。

Assessing schemas for external projects

移行ポータルの **プロジェクトスコープのワークスペースとユーザーのワークスペース** を使用して、Oracle データベースの移行用に外部から提供された DDL ファイルを評価、変換、および移行します。

EDB Postgres AI エージェント *access Migration Portal from the detailed view of the Oracle databases* を活用するハイブリッドマネージャー HM 移行のスコープで、移行ポータルのプロジェクトスコープのワークスペースにアクセスします。

前提条件

- **オプション**移行ポータルでのスキーマの非互換性の解決を促進するには、HM 管理者は enable and configure the AI Copilot with the AI/LLM of your organization する必要があります。

Migration Portal へのアクセス

Launchpad を使用して、Migration Portal にアクセスします。

1. *Access the Launchpad* 。
2. Migration Portal タブを開くには、 **Migration Portal** タイルにカーソルを合わせ、 **View Service** を選択します。

次のステップ

Migration Portal を使用して次のことを行います。

- 非互換性を解決し、スキーマの変換を完了します。
- 宛先データベースに変換されたスキーマの

Perform an online or offline migration 。

使用方法に関する推奨事項とガイドラインについては、Migration Portal documentation を参照してください。

Controlling Migration Portal project creation

EDB Postgres AI エージェントを介して Oracle データベースをハイブリッドマネージャー HM に登録した場合、すべてのデータベーススキーマと依存関係に対して移行ポータルプロジェクトが**自動的に作成されます** 。

データベースごとのスキーマサイズがデフォルトの最大サイズ 300MB を超えない限り、登録されたデータベースごとに beacon-agent.yaml で確立された resource_id ごとにプロジェクトを作成します。

> **注釈**
>
> 登録後に追加されたスキーマは、移行ポータルプロジェクトに反映されないことに注意してください。

Migration Portal プロジェクトの最大サイズのカスタマイズ

移行ポータルプロジェクトのデフォルトの最大サイズは 300MB です。取り込まれたスキーマがそれより大きい場合、移行ポータルは *Migration Portal* プロジェクトの手動作成を必要とします。ただし、たとえば、次の場合、自動的に作成されるプロジェクトに別の最大サイズを設定することが優先されるシナリオもあります。

- プロジェクトサイズの制御
- コストとリソースの管理
- ワークフロー制御

自動的に作成される移行ポータルプロジェクトの最大サイズを変更するには、HM Kubernetes クラスターで beacon_server ConfigMap を編集し、サーバー展開を再起動する必要があります。beacon_server は、移行ポータルプロジェクト構成を制御するサービスです。

1. ConfigMap を編集します。

```
kubectl edit configmap beacon-server-config -n upm-beacon
```

1. beacon_server.yaml ブロックを見つけて、server の下の max_automatic_project_creation_bytes セクションを挿入または変更します。

```
apiVersion: v1
data:
  beacon_server.yaml: |
  # [...]
    server:
    # [...]
      migration_portal_configuration:
        max_automatic_project_creation_bytes: 150000000
```

> **注釈**
>
> *max_automatic_project_creation_bytes* の値設定は、*schema_max_file_size* の構成値デフォルトで 300MB 以下である必要があります。例、

```
apiVersion: v1
data:
```

(次のページに続く)

(前のページからの続き)

```
  beacon_server.yaml: |
  # [...]
    server:
    # [...]
      stream_configuration:
        schema_max_file_size: 150000000
      migration_portal_configuration:
        max_automatic_project_creation_bytes: 150000000
```

```
kubectl rollout restart deployment upm-beacon-server -n upm-beacon
```

> **警告**
>
> *upm-beacon-server* は、EDB Postgres AI エージェントを使用して、HM インスタンス全体ですべてのデータベース登録とスキーマ取り込みアクティビティをサポートします。 *upm-beacon-server* を再起動すると、プロジェクトに関係なく、すべての進行中のデータベース登録とスキーマの取り込みが停止します。この再起動は他のユーザーと調整してください。

[Estate > Migrations] タブに既に表示されており、スキーマの取り込みが終了しているデータベースは、再起動の影響を受けません。 .. ::

1. 再起動したサービスのステータスをモニタできます。

```
kubectl get pods -n upm-beacon
```

Migration Portal プロジェクトの自動作成の無効化

自動プロジェクトの作成は、すぐに始めるのに便利ですが、次のような手動管理が優先されるシナリオもあります。

- コストとリソースの管理
- ステージング環境と開発環境
- ワークフロー制御と監査

移行ポータルプロジェクトの自動作成を無効にするには、HM Kubernetes クラスターの `beacon_server` ConfigMap を編集し、サーバー展開を再起動する必要があります。`beacon_server` は、移行ポータルプロジェクト構成を制御するサービスです。

1. ConfigMap を編集します。

```
kubectl edit configmap beacon-server-config -n upm-beacon
```

1. `beacon_server.yaml` ブロックを見つけて、`server` の下の `migration_portal_configuration` セクションを挿入または変更します。

```
apiVersion: v1
data:
  beacon_server.yaml: |
  # [...]
    server:
    # [...]
      migration_portal_configuration:
        disable_auto_create_mp_projects: true
```

1. beacon_server 展開を再起動して、変更を適用します。

```
kubectl rollout restart deployment upm-beacon-server -n upm-beacon
```

> **警告**
>
> *upm-beacon-server* は、EDB Postgres AI エージェントを使用して、HM インスタンス全体のすべてのデータベース登録とスキーマ取り込みアクティビティをサポートします。 *upm-beacon-server* を再起動すると、プロジェクトに関係なく、すべての進行中のデータベース登録とスキーマの取り込みが停止します。この再起動は他のユーザーと調整してください。

[Estate > Migrations] タブに既に表示されており、スキーマの取り込みが終了しているデータベースは、再起動の影響を受けません。 .. ::
関連トピック

- *Migrating Oracle databases with HM* 。
- *Controlling schema ingestion* 。
- Migration Portal documentation ポータルの使用方法に関する推奨事項とガイドラインについては。

Using the HM Console to perform data migration

データの移行を実行するには、HM コンソールが提供するデータ移行機能を活用できます。このセクションでは、プロジェクトの左側のナビゲーションの **Migrate** セクションで使用可能なオプションを説明する参考資料を提供します。

- **デプロイメントリソースを理解する**リーダーモードで DMS エージェントを使用して追加した移行データベースソースを表示および管理します。
- **多様な移行先** ライタモードで DMS エージェントを使用して追加した移行データベース [宛先] を表示および管理します。
- **マイグレーション**既存のデータ移行を表示するか、既存のソースから既存の宛先への新しいデータ移行を開始します。
- **ステップ *2* 資格情報**ソースと宛先を DMS エージェントに接続するために必要なプロジェクトスコープの資格情報を取得します。

Sources

ソースは、*Data Migration Service or DMS* を使用してハイブリッドマネージャーに接続したデータベースであるため、移行のソースとして使用できます。データベースをこのビューに表示するには、**リーダーモードの *EDB DMS* エージェント** をインストール、構成、および実行する必要があります。

このビューには、プロジェクト内で構成されているすべてのソースが表示されます。

各ソースのビューには、データベース名、データベースタイプ Postgres または Oracle、cdcagent ID、現在の **ステップ *4* 展開ステータスを監視する** 、ソースタイプなどが表示されます。

ステータス

HM は、ソースの次のステータスを表示できます。

Status	Description
Creating	データベースソース接続は初期化中です。これは過渡状態です。
Waiting for tables metadata	EDB DMS エージェントは、テーブルに関する情報たとえば、名前、列、データ型を取得しています。ソースがこの状態でスタックしている場合、データベースに接続の問題がある可能性があります。
Ready to start a new migration	EDB DMS エージェントはソースデータベースを正常に読み取ることができ、移行に使用できます。
Running	このソースを使用した移行プロセスは現在アクティブであり、データは転送中です。
Paused	このソースを使用している進行中の移行プロセスは、一時的に停止されました。
Stopping	このソースで現在実行中の移行は終了中です。
Connectivity error	EDB DMS エージェントは、外部管理データベースへの接続を確立できません。これは、誤った接続の詳細ホスト、ポート、データベース名、ユーザー名、パスワードまたはネットワークの問題が原因である可能性があります。
Error	未指定の問題。`cdcagent`ログを調査して、特定の原因を理解します。

注釈

アクティブな移行実行中または一時停止を含む任意の状態の一部であるソースまたは宛先は、別の移行に使用できません。ソースまたは宛先を再利用する前に、既存の移行を完了するか削除する必要があります。

Destinations

宛先は、*Data Migration Service or DMS* を使用してハイブリッドマネージャーに接続した外部管理データベースであるため、移行の宛先として使用できます。データベースをこのビューに表示するには、**ライタモードの *EDB DMS* エージェント** をインストールして構成する必要があります。

このビューには、プロジェクト内で設定されているすべての宛先が表示されます。

各宛先のビューには、データベース名、データベースタイプ、cdcagent ID、現在の **ステップ *4* 展開ステー**

タスを監視する 、宛先タイプなどが表示されます。

ステータス

HM は、宛先の次のステータスを表示できます。

Status	Description
Waiting for tables metadata	EDB DMS エージェントは、テーブルに関する情報たとえば、名前、列、データ型を取得しています。宛先がこの状態でスタックしている場合、データベースに接続の問題がある可能性があります。
Ready to start a new migration	EDB DMS エージェントは宛先データベースに正常にアクセスでき、移行に使用できます。
Running	この宛先を使用した移行プロセスは現在アクティブであり、データが転送されています。
Paused	この宛先を使用している進行中の移行プロセスは、一時的に停止されました。
Connectivity error	EDB DMS エージェントは、外部管理データベースへの接続を確立できません。これは、誤った接続の詳細ホスト、ポート、データベース名、ユーザー名、パスワードまたはネットワークの問題が原因である可能性があります。
Error	未指定の問題。 HM コンソールと `cdcagent` ログを調査して、特定の原因を理解します。

注釈

アクティブな移行実行中または一時停止を含む任意の状態の一部であるソースまたは宛先は、別の移行に使用できません。ソースまたは宛先を再利用する前に、既存の移行を完了するか削除する必要があります。

Migrations

移行は、ハイブリッドマネージャー HM のプロジェクトで開始でき、データベースから別のデータベースにデータを移動するプロセスです。これらのプロセスには、外部 Oracle データベースの HM 管理クラスタおよび外部管理の Postgres インスタンスなどへの移行が含まれます。

既存の移行の表示

1 つ以上の移行を作成すると、このビューには、プロジェクト内のすべての移行が表形式で表示されます。テーブルには次の列が含まれています。

- **Name** 移行に割り当てた名前。
- **Source** ソースの名前。 **Sources** ページで構成します。
- **宛先** 宛先の名前。HM 管理対象クラスター内のデータベースまたは**宛先** ページで構成された宛先のいずれか。
- **スナップショットタイプ** 移行ごとに構成されたレプリケーションモード

- **Snapshot** 移行はスナップショットのみを実行します。ソースに対する後続の変更は、宛先にレプリケートされません。
- **スナップショット+ストリーミング**移行は初期スナップショットを実行し、ソースデータベースから進行中の変更のストリーミングを継続します。
- **移行の進行状況**この列は、移行がどの程度進行したかを要約します。移行ごとに 3 つのメトリックが表示されます。
- **テーブルの合計数** この移行に含まれるテーブルの合計数 .
- **スナップショットテーブルが完了しました**スナップショットステージを終了したテーブルの数。
- **移行行** この移行のすべてのテーブルで移行された行の累積数。
- **Status** 移行の現在のステータス。値は次のいずれかです。`Creating`、`Starting`、`Running`、`Pausing`、`Paused`、`Error`、`Completed`、`Failed`、`Deleting`、または `Deleted`。
- **Delete** 各行の最後にあるゴミ箱アイコンを使用して、移行を削除できます。

ページの右上にある **Sort** ドロップダウンを使用して、 **作成日**で移行のリストを並べ替えることができます。デフォルトでは、リストは新しいものから古い作成日でソートされます。

Create Migration を使用して、新しい移行を作成できます。詳細については、移行の作成 を参照してください。

ステータス

HM は、移行の次のステータスを表示できます。

Status	Description
Creating	HM は、移行を実行するために必要なリソースたとえば、Kafka ノードなどを作成しています。
Starting	実行の初期フェーズが開始され、EDB DMS エージェントがソースおよび/または宛先データベースに接続します。
Running	移行はアクティブで、データは転送中です。
Pausing	移行は一時的に停止されています。
Paused	移行は一時的に停止されましたが、再開できます。
Error	未指定の問題。 **Migrations** の下で、移行の名前を選択して詳細ページを開きます。原因の詳細については、 **Error log** セクションを参照してください。
Completed	移行は正常に完了しました。
Failed	移行でエラーが発生し、完了できませんでした。 HM コンソールログを調査して、特定の原因を理解します。
Deleting	移行は終了されています。
Deleted	移行は正常に終了および削除されました。

注釈

アクティブな移行実行中または一時停止を含む任意の状態の一部であるソースまたは宛先は、別の移行に使用できません。ソースまたは宛先を再利用する前に、既存の移行を完了するか削除する必要があります。

移行の詳細ページ

このページに表示される各移行について、移行の名前を選択することにより、より詳細なビューを開くことができます。

Mark as Completed オプション移行の作成 のときに ***Snapshot + Streaming*** オプションを選択した場合、**Mark as Completed** を使用してストリーミングプロシージャを停止できます。

[概要] タブ

[**概要**] タブには、移行のステータス、移行の種類、ソースおよび宛先データベース情報、およびトラブルシューティングに役立つエラーログに関する追加情報が含まれています。

[テーブル] タブ

[**Tables**] タブは、移行内の個々のテーブルの進行状況に関する詳細情報を提供します。

1. **移行の概要**

タブの上部の概要セクションには、移行の主要なメトリックが表示されます。

- **Total schemas** : この移行に含まれるスキーマの合計数 .
- **Total tasks** テーブルの合計数この移行に含まれます .
- **Snapshot completed** スナップショットステージを終了し、ストリーミングの準備ができているテーブルの数該当する場合。

2. **検索とフィルタリングバー**

フィルタリングバーを使用すると、テーブルリストを絞り込んで特定のデータセットに焦点を当てたり、移行のボトルネックをトラブルシューティングしたりできます。

- **グローバル検索**検索バーを使用して、スキーマまたはテーブル名に含まれるキーワードまたはフレーズを入力して、特定のテーブルを見つけます。
- **スナップショットステータスフィルタ**ステータスフィルタドロップダウンを使用して、**Pending** 、**In Progress** 、 **Completed** 、または **Failed** などの移行ステータスに基づいてテーブルを表示します。

3. **ソートドロップダウン**

並べ替えドロップダウンは、選択した順序に従ってテーブルリストの表示順序を調整します。デフォルトでは、テーブルはスキーマとテーブル名前のアルファベット順にソートされます。次のようなさまざまな基準に基づいてソート順を変更できます。

- **テーブル名** 名前のアルファベット順にテーブルを並べ替えます。
- **スナップショットステータス**現在のスナップショットステータスに基づいてテーブルを並べ替えます。
- **移行済み rows** 移行された行数でテーブルを並べ替えます。

4. **テーブルの詳細ビュー**

このセクションでは、移行内のすべてのテーブルの行レベルの粒度を提供します。

テーブルごとに、ソースおよび宛先のスキーマとテーブル名、スナップショットステータス、マップされた列と移行された行の数、および進行中の移行の移行率を表示できます。スナップショットステータスは、スナップショットのみのジョブの場合は全体的な移行状態、またはスナップショット+ストリーミングジョブの場合はスナップショットフェーズステータスを反映します。各行は展開可能なため、ソースから宛先までの詳細な列レベルのマッピングを表示できます。

[エラー履歴] タブ

[**Errorhistory**] タブには、この移行中に DMS がログに記録したすべてのエラーのリストが表示されます。エラーごとに、要約されたエラーメッセージ、エラーソースリーダーまたはライタモードの DMS エージェント、Kafka など、エラーがアクティブかどうか、およびエラータイムスタンプを表示できます。エラーの詳細を取得するには、情報アイコンを選択して、完全なエラーログを含むビューを開きます。特定の問題を分離するのに役立つように、**Errorhistory** タブには、高度なフィルタリングと時間範囲選択機能が含まれています。

- **グローバル検索**検索バーを使用して、エラーメッセージに含まれるキーワードまたはフレーズを入力して、特定のエラーを見つけます。
- **柔軟な時間範囲**時間範囲セレクターを使用して、事前定義された間隔過去 7 日間、過去 30 日、過去 90 日間に基づいてエラーをフィルタリングするか、カスタム日付と時刻範囲を指定します。
- **高度なフィルタリング**フィルタバーを使用して、エラー属性に基づいてリストを絞り込みます。次の方法を使用してフィルタリングできます。
- **Errorsource** リーダーまたはライタモードの DMS エージェント、Kafka など、特定のコンポーネントから発生したエラーのみを表示します。
- **Active** ステータスがアクティブ間ですばやくフィルタリングしますおよび解決されたエラー。

移行の作成

移行を作成するには、 **Create Migration** を選択します。ステップバイステップの例については、自己管理の Postgres から HM ガイドの 移行の作成 、または移行ガイドの同等のセクションを参照してください。

移行中のマッピングスキーマ、テーブル、および列

データのみスコープで移行を作成する場合、データ移行サービス DMS を使用すると、ソーススキーマ、テーブル、および列名をさまざまな宛先スキーマ、テーブル、および列にマッピングできます。この方法は、移行中にデータベースを再構成する場合に特に役立ち、テーブル名を新しい命名規則に揃えたり、明確にしたりできます。

列マッピングは、 **スキーマとデータ** または **スキーマのみ**スコープでは使用できません。これらのスコープでは、HM は、ソースデータベースから作成した移行ポータルプロジェクトを使用して、スキーマの移行を自動的に推進します。HM は宛先オブジェクト自分自身を作成するため、その名前を制御します。構成した列の名前の変更は移行定義に保存されますが、スコープを**データのみ** に切り替えない限り有効になりません。

前提条件

DMS では、データの移行が開始される前に、すべての宛先オブジェクトが宛先データベースに既に存在する必要があります。ソースオブジェクトスキーマ、テーブル、または列を宛先の別の名前たとえば、ソーステーブル `loc` から宛先テーブル `location` にマッピングすることによりデータを再構成する予定がある場合、宛先オブジェクト `location` がデータの移行を開始する前に、宛先データベースで手動で作成または名前変更します。

- **スキーマ移行ツール** スキーマのインポートに Migration Toolkit または pg_dump / pg_restore のようなユーティリティを使用する場合、オブジェクトはソースと同じ名前で作成されます。次に、目的のマッピングと一致するように、宛先内のオブジェクトの名前を手動で変更する必要があります。たとえば、移行ツールキットのオフラインオプションまたは pg_dump を使用して SQL スクリプトを生成し、スクリプト内のオブジェクト名を直接編集し、変更したスクリプトを宛先に適用できます。
- **手動/代替作成**宛先オブジェクトを手動で作成する場合、マップされたすべての列と互換性のあるデータ型でオブジェクトが定義されていることを確認します。DMS は別の名前のオブジェクトにデータを移行できますが、宛先オブジェクトの構造はソースデータと一致する必要があります。

オブジェクトマッピング

1. HM コンソールで **Migrate** > **Migrations** > **Create Migration** から **Create Migration** ウィンドウを開きます。詳細については、:ref:``edbctl` を使用して移行ポータルシークレットを作成する <*edbctl* を使用して移行ポータルシークレットを作成する>` を参照してください。
2. 移行の **Tables** タブに移動します。
3. スキーマを名前変更した宛先スキーマにマップするには
    1. リスト内でスキーマの名前を変更するテーブル<schema>.<table> を見つけます。
    2. <schema>.<table> 名前の横にあるペンアイコンを選択します。
    3. スキーマの新しい名前を入力します。
4. 名前を変更した宛先テーブルにテーブルをマップするには
    1. テーブルの名前を変更するリスト<schema>.<table> 内のテーブルを見つけます。
    2. <schema>.<table> 名前の横にあるペンアイコンを選択します。
    3. テーブルの新しい名前を入力します。
5. 名前を変更した宛先列に列をマップするには
    1. ドロップダウン矢印でリスト<schema>.<table> のいずれかを展開します。
    2. **ソース列** リストで名前を変更する列を見つけます。
    3. **宛先列** の列の新しい名前。

このプロセスでは、移行定義の一部としてスキーマ、テーブル、および列をマップできます。

Credentials

移行を実行するには資格情報が必要です。移行のソースまたは宛先をハイブリッドマネージャー HM に追加する前に、移行の資格情報を作成する必要があります。次に、それらを使用して、EDB DMS エージェントが HM に接続できるようにします。

Credentials ビューには、プロジェクト内のすべての資格情報が表形式で表示されます。各資格情報のエントリには、その名前、ステータス、作成日、および有効期限が表示されます。

移行資格情報の作成

EDB DMS エージェントの構成の一部として、移行資格情報を作成します。ステップバイステップの例については、自己管理の Postgres から HM ガイドの **移行資格情報の作成** 、または移行ガイドの同等のセクションを参照してください。

解凍された資格情報フォルダーには、次のファイルが含まれています。

- `int.crt`
- `int.truststore.p12`
- `client.keystore.p12`
- `client-key.pem`
- `client-cert.pem`

Using the Migration API

Migration API は、データベースの移行タスクを自動化するための重要なツールです。プログラム的に移行を管理および監視でき、移行ワークフローを既存の CI / CD パイプラインまたは自動スクリプトに統合する強力な方法を提供します。UI をナビゲートする代わりに、API を使用して、移行プロジェクトの管理、詳細なスキーマレポートの表示、データの移行ステータスを確認し、その他の多くの管理タスクを実行できます。

この API は、大量の移行を処理するか、より広範な自動化戦略に統合するためのスケーラブルで自動化された方法を必要とする開発者、DevOps エンジニア、および DBA を対象としています。

Hybrid ManagerHM プラットフォームは、評価、スキーマ変換、データ移行を含むエンドツーエンドのデータベース移行をサポートするためのより広範な API セットを提供します。ただし、**例** には、サブセットのみが表示されます。

例

Migration Portal プロジェクトの作成

この API エンドポイントは、移行ポータルで新しい移行プロジェクトを作成するために使用されます。これは、JSON ペイロードを送信してプロジェクトのプロパティと構成を定義する `POST` 要求です。

- **パス** `/api/v1/projects/prj_LTIfqp81u6MslAk4/migration-portal-project`
- **メソッド** `POST`

URL パスパラメーター

- prj_LTIfqp81u6MslAk4 プロジェクトのスコープの一意の識別子。

```
Request Body:
{
    "projectId": "prj_LTIfqp81u6MslAk4",
    "resourceId": "oracleSource",
    "schemaIds": [
        "cea06ec8-8641-4dff-9631-4b19fcba5212",
        "5770b6fa-295c-40af-bd02-1acbe02c9eb6"
    ],
    "mpProjectName": "my_project_name",
    "applicationInterface": "JDBC",
    "targetDbVersion": "EPAS_17",
    "useUniqueSchemaObjectSuffixes": true,
    "useOracleDefaultCase": true,
    "assessDefaultProfile": true,
    "description": "My Project Description (Optional)"
}
```

要求本文の説明

- projectId HM 内のプロジェクトの一意の識別子。
- resourceId ソースデータベースの一意の識別子たとえば oracleSource 。
- schemaIds 移行プロジェクトに含めるスキーマの一意の ID の配列。
- mpProjectName 新しい移行ポータルプロジェクトに割り当てるカスタム名。
- applicationInterface ソースデータベースへの接続に使用されるインターフェイスのタイプたとえば、JDBC 、ODBC 、.NET 、OCI 、ProC 、Other など。
- targetDbVersion 移行先のターゲットデータベースのバージョンたとえば、EDB Postgres Advanced Server バージョン 17 の場合は EPAS_17 。
- useUniqueSchemaObjectSuffixes true に設定すると、ターゲットスキーマ内のオブジェクト名に一意のサフィックスを持つことを保証するブール値、競合を回避します。
- useOracleDefaultCase true に設定されている場合、オブジェクト名に大文字を使用する Oracle のデフォルト動作を尊重するブール値。
- assessDefaultProfile true に設定されている場合、評価にデフォルトプロファイルを含むブール値。
- description 移行プロジェクトのカスタム、オプションの説明。

Migration Portal プロジェクトの削除

この API エンドポイントは、既存の移行プロジェクトを移行ポータルから削除します。これは、ID で特定のプロジェクトをターゲットとする DELETE 要求です。

- **パス**/api/v1/projects/prj_LTIfqp81u6MslAk4/migration-portal-project/42

- メソッド DELETE

URL パスパラメーター

- prj_LTIfqp81u6MslAk4 プロジェクトのスコープの一意の識別子。
- 42 削除する特定の移行ポータルプロジェクトの一意の ID。この ID は、プロジェクトが正常に作成された後に割り当てられます。

5.3.3.5 Asset management

Hybrid ManagerHM コンソールの Asset Management セクションは、クラスターで使用可能な資産を管理するためのツールを提供します。これには、コンテナイメージと AI モデル、およびこれらのアセットを含むリポジトリの管理が含まれます。

5.3.3.5.1 HM Postgres とモデルアセットの管理

- アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です

HM コンソールでアセットを管理するためのメインページ。コンテナイメージとナレッジベースを表示および管理できます。

- コンテナイメージ

イメージの詳細を検索、フィルタリング、表示することにより、アセットライブラリ内のコンテナイメージを探索および管理できます。

- *Manage repositories in the asset library*

リポジトリの登録とルールを管理することにより、アセットライブラリ内のリポジトリと画像の可視性を制御します。

- ナレッジベースと検索

クラスターで使用する AI モデルを含むナレッジベースの作成と管理を含む、アセットライブラリの AI モデルを管理します。

Asset library

アセットライブラリは、ハイブリッドマネージャークラスターで使用可能なコンテナイメージとモデルを管理する場所です。

アセットライブラリへのアクセス

アセットライブラリにアクセスするには、Hybrid ManagerHM コンソールの上部にある **Asset ライブラリー**タブを選択します。

アセットライブラリーページ

アセットライブラリーページには、ライブラリで使用可能な画像の概要数、検索およびフィルタバー、およびライブラリで使用可能な主要画像のリストが表示されます。

アセットライブラリは、 **コンテナイメージ** と **モデル**の 2 つのメインセクションに分割されています。各セクションでは、HM クラスターで使用可能なアセットを表示および管理できます。

コンテナイメージ

アセットライブラリの **Container Images** セクションでは、HM クラスターで使用可能なコンテナイメージを表示および管理できます。これには、特定の画像の検索、さまざまな基準で画像リストをフィルタリング、および各画像に関する詳細情報を表示する機能が含まれます。このタブの使用方法の詳細については、コンテナイメージ を参照してください。

モデル

モデルは、HM で使用できる AI モデルです。これらのモデルを特定のデータセットで使用して、自然言語処理、画像認識などを含むさまざまなタスクを実行できます。このタブの使用方法の詳細については、ナレッジベースと検索 を参照してください。

Container images in the asset library

アセットライブラリの **Container Images** セクションでは、ハイブリッドマネージャー HM クラスターで使用可能なコンテナイメージを表示および管理できます。

コンテナイメージ

コンテナイメージは、さまざまな拡張機能を含むデータベースの構成です。各画像は、名前と画像タグによって識別されます。

名前ごとに、イメージにロードされた拡張機能のさまざまなバージョンとさまざまな組み合わせを表す複数のイメージタグが存在する場合があります。

たとえば、公式 HM イメージには、イメージ名が `postgresql` と、バージョン 17.4 のコミュニティ PostgreSQL イメージを表すイメージタグ `17.4-2503191025` があります。 `2503191025` は、イメージのビルド日時 2025/03/19 10:25 です。後置 `-full` は、このイメージに HM Postgres ディストリビューションで使用可能なすべての拡張機能が含まれていることを示します。

このタグ形式は公式 HM 画像に固有であり、他の画像とは異なる場合があります。イメージ名は、Postgres の特定のフレーバーなど、画像の種類を識別するために使用されます。イメージタグは、イメージの特定のバージョンを識別するために使用されます。

画像統計

ライブラリで使用可能な画像の数の概要がタイプごとにリストされています。画像の種類は次のとおりです。

- **合計イメージタグ**ライブラリで使用可能なイメージタグの合計数。タグは画像の特定のバージョンを表します。
- **PostgreSQL Images** ライブラリで使用可能な Postgres イメージの数。

- **AI モデル画像** ライブラリで使用可能な AI モデル画像の数。

コンテナイメージの検索とフィルターバー

このビューでは、 **Search** フィールドを使用して、イメージリポジトリの名前または説明の一部である検索用語を入力します。**Filter** フィールドを使用して、タイプ Postgres または AI モデルでリポジトリをフィルタリングします。

Sort フィールドを使用して、名前、説明、タイプ、またはリポジトリが更新された日でリポジトリを並べ替えます。

検索とフィルターバーで、**Manage Repositories** を選択して、ライブラリで使用可能なリポジトリを管理します。ダイアログボックスを使用して、リポジトリを追加、編集、または削除します。**Manage Repositories** ページでは、アセットライブラリで使用可能なリポジトリとイメージを管理できます。リポジトリの管理方法の詳細については、*Manage repositories in the asset library* を参照してください。

コンテナイメージカード

各イメージリポジトリは、次の情報を含むカード形式で表示されます。

- **Name** イメージリポジトリの名前。
- **Description** イメージリポジトリの簡単な説明。
- **タグ**この画像リポジトリに関連付けられた HM タグ。イメージリポジトリにタグが関連付けられていない場合、**Add Tag** が表示されます。
- **Updated** イメージリポジトリが最後に更新されてからの日数。

イメージリポジトリの名前を選択して、そのイメージの詳細を含むページに移動します。

画像の詳細

画像の **画像の詳細** ページには、 **概要** と **イメージタグ**の 2 つのタブがあります。 **Add Tag** ボタンを使用して、HM タグを画像リポジトリに追加できます。既存のタグはボタンの前に表示され、タグを選択して詳細を表示できます。

[概要] タブ

[**概要**] タブは、以下を含むイメージリポジトリの概要を提供します。

- **README** イメージリポジトリのテキストによる説明。これには、画像の使用方法、含まれる内容、その他の役立つ詳細など、画像に関する関連情報が含まれます。
- **最近の画像タグ**この画像リポジトリで最近作成された 10 個の画像タグ。各タグを選択して、そのタグの**イメージタグの詳細** ページに移動できます。

クラスターを使用してこのイメージセクションには、このリポジトリのイメージを使用しているエステート内のクラスターがリストされます。このイメージを使用しているクラスターがない場合、「No Data」というメッセージが表示されます。

検索とフィルターバーを使用して、クラスターを検索します。 **Search** フィールドを使用して、クラスターの名前または説明の一部である検索用語を入力します。**Filter** フィールドを使用して、属性クラスタータイプ、クラスターアーキテクチャ、Postgres フレーバー、および Postgres バージョンでクラスターをフィ

ルタリングします。**Sort** フィールドを使用して、名前、説明、またはクラスターが作成された日で並べ替えるクラスターを表示します。

クラスター名を選択して、そのクラスターの **クラスターの詳細の概要**ページに移動します。

画像タグ

Image Tags タブを選択して、このイメージリポジトリに関連付けられているすべてのイメージタグのリストを表示します。

各タグは、次の情報を含む表形式で表示されます。

- **Name** 画像タグの名前。
- **Created** 画像タグが作成された日時。
- **Last Pushed** イメージタグがリポジトリにプッシュされた日時。
- **プラットフォームアーキテクチャ**イメージタグが使用可能なプラットフォームアーキテクチャ、X86 または AMD64 のいずれか。
- **SHA** 画像の SHA256 ハッシュ。画像の一意の識別子。これを使用して、他のリポジトリと比較してイメージを検証できます。ハッシュの最後の 8 桁が表示されます。これらの数字にカーソルを合わせると、完全なハッシュが表示されます。**Copy** ボタンを使用して、完全なハッシュをコピーできます。

イメージタグのリストをナビゲートするには、テーブルの上部にある検索とフィルタバーを使用できます。**Search** フィールドに、画像タグ名の一部である検索用語を入力します。**Filter** フィールドを使用して、属性 EDB 公式リリース、利用可能な場所、プラットフォームアーキテクチャによってイメージタグをフィルタリングします。**Sort** フィールドを使用して、バージョン、作成日、または最後にプッシュされた日でイメージタグを並べ替えます。

イメージタグの名前を選択して、そのタグの **イメージタグの詳細**ページに移動します。

イメージタグの詳細

Image Tag Details ページには、特定の画像タグに関する最も詳細な情報が表示されます。

ページの上部にあるのは、イメージタグの名前です。その下は、このイメージタグに関連付けられている HM タグです。これには、EDB 公式タグ、プラットフォームによって生成された透過的データ暗号化タグ、および作成したタグが含まれる場合があります。**Add Tag** を選択して、イメージタグにタグを追加または削除し、必要に応じて新しいタグを作成します。

次は次のとおりです。

- **README** イメージリポジトリのテキスト説明。これには、画像の使用方法、含まれる内容、およびその他の役立つ詳細など、画像に関する関連情報が含まれます。
- 次の情報を表示するイメージ属性パネル
  - **バージョン** イメージタグのバージョン
  - **ベースイメージ** イメージの基になる OS イメージ
  - **イメージ SHA** : image.
  - の SHA256 ハッシュリポジトリへ。

- **Created** イメージタグが作成された日時。
- **プラットフォームアーキテクチャ**イメージタグが使用可能なプラットフォームアーキテクチャ、X86 または AMD64 のいずれか。
- **Transparent Data Encryption** イメージタグが透過的データ暗号化 TDE をサポートするかどうかを示します。

Deployment Locations というラベルのフィールドは、イメージタグが利用可能な場所をリストしています。

表には、このイメージタグに含まれる拡張子が示されています。各拡張機能は、名前、バージョン、および説明とともに表示されます。**Search** フィールドを使用して、名前または説明で特定の拡張機能を検索します。このテーブルを名前でソートできます。

このイメージタグを使用しているクラスターセクションには、このイメージタグを使用しているエステート内のクラスターがリストされています。

Manage repositories in the asset library

基になるリポジトリへの接続はシステム管理者が構成しますが、アセットライブラリで使用可能なリポジトリを管理できます。

これは、特定のパターンに一致するリポジトリとタグ付けされたイメージをシステムに含められるようにすることで実現できます。各パターンはルールとして定義されます。次に、どのリポジトリをアセットライブラリに含めるか、およびどのイメージをハイブリッドマネージャー HM クラスターで使用できるかを制御できます。

デフォルトでは、アセットライブラリには、すべての公式 HM リポジトリと、それらのリポジトリ内のすべてのタグ付け画像が含まれます。

アセットライブラリで使用可能なリポジトリとイメージをより詳細に制御するには、プライベートレジストリでホストできる独自のプライベートリポジトリを管理できます。これにより、レジストリに含まれる内容を完全に制御でき、独自のイメージを追加して管理できます。どのルートを選択しても、アセットライブラリのリポジトリ包含ルールが動作して、アセットライブラリで使用可能なものを制御します。

リポジトリの管理

アセットライブラリの **Container Images** タブで、**Manage Repositories** を選択します。 **Manage Repositories** ページを使用して、アセットライブラリで使用可能なリポジトリ登録を追加、編集、または削除します。

現在アセットライブラリに登録されているリポジトリのテーブルには、次の列があります。

- **Repository Name Regex** リポジトリの名前が一致する必要がある正規表現。
- **ルールの数** このリポジトリ登録に関連付けられているルールの数。
- **Created** リポジトリ登録が作成された日時。

データが入力されると、各行には、リポジトリ登録を編集するための **Edit** ボタンと、リポジトリ登録を削除するための **Delete** ボタンが含まれます。

検索フィールドを使用して、特定のリポジトリ登録を名前で検索します。ソートコントロールを使用して、リポジトリ名または作成日でテーブルをソートします。

画面の右上隅にある **Register New Repository** を選択して、リポジトリ登録を追加します。

新しいリポジトリの登録

Register New Repository を選択して、新しいリポジトリ登録の詳細を入力できるダイアログボックスを開きます。

- **Repository Name Regex** リポジトリの名前と一致する正規表現。これは、アセットライブラリに含まれるリポジトリをフィルタリングするために使用されます。

このフィールドは、アセットライブラリに含めるリポジトリの名前と一致する必要があります。たとえば、`edb-postgres-advanced-` で始まるすべてのリポジトリを含める場合は、正規表現として`^edb-postgres-advanced-.*$` を入力します。プレーンテキストを入力できますが、正規表現として扱われます。正規表現を使用して、リポジトリ名が目的のパターンと一致することを確認することをお勧めします。

- **ルール**このリポジトリ登録に関連付けられているルールのリスト。各ルールは、画像に関連付けられているタグと一致する正規表現で構成されています。ルールは、アセットライブラリに含まれる画像をフィルタリングするために使用されます。たとえば、タグ `latest` を持つすべての画像を含める場合は、正規表現として`^latest$` と入力します。プレーンテキストを入力できますが、正規表現として扱われます。正規表現を使用して、タグが目的のパターンと正確に一致することを確認することをお勧めします。

ルールを追加するには、 **Add Rule** を選択します。これにより、ルールのリストに新しい行が追加されます。ルールを削除するには、ルールの横にある **Delete** を選択します。

新しいリポジトリ登録の詳細を入力したら、[**Register Repository**] を選択して登録を保存します。新しいリポジトリの登録は、登録済みリポジトリのリストに追加され、アセットライブラリで利用できるようになります。システムはリポジトリをスキャンして、定義したルールと一致するイメージをスキャンする必要があるため、これには数分かかる場合があります。

操作が完了すると、 **Manage Repositories** ページに戻ります。登録済みリポジトリのリストに新しいリポジトリの登録が表示されます。

アセットライブラリーアプリには、プロジェクト所有者またはプロジェクト編集者のロールが必要です に戻って、新しいリポジトリールールの効果と使用可能になったイメージを確認します。

Knowledge bases in the asset library

アセットライブラリの **ナレッジベース**ページでは、ハイブリッドマネージャー HM クラスターで使用できる AI モデルのコレクションを管理できます。

Knowledge Bases ページは、検索とフィルターバー、並べ替えコントロール、およびナレッジベースのテーブルで構成されています。各ナレッジベースは独自の行に表示され、次の情報が表示されます。

- **Name** ナレッジベースの名前。
- **ソースデータ** ナレッジベースの構築の元になるソースデータ。
- **データベース** ナレッジベースの構築元のデータベース。
- **クラスター** ナレッジベースが構築されるクラスター。
- **モデル** ナレッジベースの構築に使用される AI モデル。

5.3.3.6 Backups and restores

バックアップと復元の基本的なタスクは次のとおりです。

- マルチ *DC* 環境の *HM* バックアップと *DR*
- *Backups and restores*

5.3.3.6.1 Backups

Hybrid ManagerHM は、さまざまなテクノロジーを使用したコンポーネントのバックアップおよびリカバリー機能を備えています。2 つのバックアップ戦略を提供します。

- Postgres バックアップ データをバックアップします
- HM バックアップ HM システムのコア機能をバックアップします

Postgres バックアップデータプレーン

Postgres バックアップは、Postgres データベースに保存されているユーザーデータの保護に焦点を当てています。

- ベースバックアップは、すべての HM 展開で利用できます。
- スナップショットバックアップは、クラウドサービスプロバイダー CSP 環境と、適切なストレージとドライバーが構成されているオンプレミス展開で利用できます。

Postgres バックアップは、ログ先行書き込み WAL アーカイブなどの堅牢なデータベース機能を活用して、きめの細かいポイントインタイムリカバリー PITR を提供します。このアプローチでは、きめの細かい目標リカバリーポイント RPO が可能になります。

> **注釈**
>
> ローカルストレージは、スナップショットバックアップで現在サポートされていません。

HM バックアップコントロールプレーン

HM バックアップは、メタデータやシステムデータベースを含む HM システムを保護します。これらは別のスケジュールで動作し、Postgres バックアップとは異なるバックアップファイルを生成します。HM バックアップはファイルシステムベースであり、デフォルトでメタデータの時間ごとのバックアップとシステムの Postgres データベースのバックアップを行います。

Postgres バックアップとは異なり、HM バックアップは、WAL アーカイブを介した継続的なバックアップを提供しません。したがって、復旧は時間ごとまたはユーザー定義のバックアップ間隔に制限されます。

主な違いとリカバリーの影響

これらの違いにより、RPO と目標復旧時間 RTO は、ユーザーデータ Postgres と HM システムで異なります。

- Postgres データは、PITR および WAL アーカイブにより強力な RPO 機能を提供し、非常に最近のポイント・インタイムへのリカバリーを可能にする可能性があります。

- HM システムの場合、RPO はユーザー変更可能なバックアップスケジュールに制限されます。デフォルトでは時間ごとに。

Component	RPO	RTO	Method	Location
HM	1 hour (default)	30 mins or less	Velero	Object storage
Postgres data plane	5 minutes or less	Based on data size, backup method, and hardware.	Snapshot or base backup	Object storage

データサイズ

ベースバックアップを使用する Postgres クラスターの場合、500 GB 未満のデータサイズが有効なサイズ制限であると予想されます。ハードウェアのパフォーマンスによっては、最大 1 TB になる場合があります。

スナップショットバックアップを使用する Postgres クラスターの場合、1 TB を超えるデータサイズが、サービス品質に悪影響を与えるバックアップおよび復元プロセスなしでサポートされる予定です。

バックアップの可搬性とディザスターリカバリー DR

最も単純な構成では、たとえば TopoLVM バージョン 1.1 の GA では使用できません。スナップショットのバックアップとリカバリーはローカルのみであることが想定されており、スナップショットバックアップを別のアプライアンス、データセンター、または場所に移動できないことを意味します。この制限により、スナップショットの最も基本的な実装を使用した DR が効果的に除外されます。

HM がエンタープライズストレージと、スナップショットをサポートする CSI ドライバーで構成されている場合、HM はその機能を利用してクロスロケーション DR を提供できます。

スナップショットの可用性

ボリュームスナップショットは、Kubernetes 互換のボリュームスナップショット機能を提供するすべての環境でサポートされています。

スナップショットのトランスポート性

スナップショットのトランスポート可能性は、特定の CSI ドライバー実装に依存します。

AWS の詳細

- デフォルトでは、AWS ボリュームスナップショットはトランスポートできません。
- 同じ AWS リージョン内の新しいホストへの復元は可能ですが、クロスリージョンの復元には追加の手動手順が必要です。

TopoLVM の詳細

TopoLVM スナップショットはトランスポート不可で、別のホストに復元できません。

構成

次のストレージ、オブジェクトストレージ、およびバックアップを構成できます。

- シングルノードクラスター

・プライマリ/スタンバイクラスター

- EDB Postgres 分散 PGD クラスター

シングルノードクラスター

シングルノードクラスターのバックアップには、次の特徴がある場合があります。

- シングルノードクラスターは、HM を使用して展開されます。
- 保持期間とバックアップ時間を変更するか、デフォルト値を使用できます。
- 自動バックアップは、クラスター展開後、毎日発生します。
- 手動バックアップを実行できます。

プライマリ/スタンバイクラスターのバックアップ

プライマリ/スタンバイクラスターのバックアップには、次の特徴がある場合があります。

- プライマリ/スタンバイクラスターは、HM を使用して展開されます。
- 保持期間とバックアップ時間を変更するか、デフォルト値を使用できます。
- 自動バックアップは、クラスター展開後、毎日発生します。
- 手動バックアップを実行できます。
- バックアップはレプリカプライマリ/セカンダリから取得されます。

バックアップ PGD クラスター

PGD クラスターのバックアップには、次の特徴がある場合があります。

- PGD クラスターは、HM を使用して展開されます。
- 保持期間とバックアップ時間を変更するか、デフォルト値を使用できます。
- 自動バックアップは、クラスター展開後、毎日発生します。
- 手動バックアップを実行できます。
- バックアップはレプリカプライマリ/セカンダリから取得されます。

バックアップ圧縮

バックアップ圧縮は、オブジェクトストレージに保存する前に圧縮アルゴリズムを使用してバックアップデータのサイズを削減します。HM は、 Barman Cloud プラグインのアルゴリズムを使用して、デフォルトで Postgres バックアップを圧縮します。圧縮は、ベースバックアップと WAL アーカイブに個別に適用されます。

- ベースバックアップは、デフォルトで Snappy 圧縮を使用します。
- WAL アーカイブは、デフォルトで zstd 圧縮を使用します。

圧縮により、バックアップパフォーマンスが向上し、ストレージ消費量が削減されます。これは、すべての新しいクラスターで自動的に有効になります。アクティブにするために追加の構成は必要ありません。

クラスターの作成または編集時に、 *Backup and security settings* タブで各バックアップタイプの圧縮アルゴリズムを個別に変更できます。

HM バックアップ

HM バックアップには、HM システムを復元するために必要なメタデータが含まれています。ユーザーデータベースのバックアップは含まれません。HM バックアップは、Velero を使用してオブジェクトストレージにバックアップします。デフォルトのバックアップスケジュールは 1 時間ごとです。スケジュールを調整できます。

デフォルトのスケジュールを使用する場合、RPO は 1 時間です。これは、クラッシュが発生した場合に最大 1 時間の HM データが失われることが予想されることを意味します。ユーザーデータベースは Postgres の継続的なバックアップで保護されているため、これはユーザーデータベースレベルでのデータ損失を意味するものではありません。

Postgres のバックアップの取得

HM コンソール、API、または CLI を使用して、Postgres バックアップを取得できます。API または CLI を使用して Postgres バックアップを取得するには、HM コンソールの API ドキュメントリンクを参照してください。

HM コンソールを使用して Postgres バックアップを取得するには

1. **Backups** タブを選択して、既存のバックアップを表示します。
2. **Create Backup** を選択します。
3. 場所、バックアップ名、およびバックアップ方法を入力し、 **OK** を選択します。

HM バックアップの取得

HM システムのバックアップを構成するには、*Hybrid Manager disaster recovery* を参照してください。

用語

次の用語は、Postgres および HM でのバックアップとリカバリーに使用されます。

バックアップ

元の場所からオブジェクトストレージのようなセカンダリの場所への 1 つ以上のファイルのコピー。バックアップは、コンポーネントタイプたとえば、コントロールプレーンとデータプレーンなど、展開タイプたとえば、CSP、エンジニアドシステム、または RHOS などによって異なります。

バックアップ保持期間

データのバックアップコピーが削除または上書きされるまでに保存される時間。バックアップを復旧のために利用できる期間を定義します。

Barman

主に EDB によって維持およびサポートされている open-source Postgres backup tool 。単独で使用される場合、Barman という用語は、 *Barman* クラウドユーティリティスクリプト ではなく、Barman サーバーまたはコアバックアップツールを指します。Barman は、エンタープライズクラスのバックアップとリカバリーに必要な Postgres のバックアップとリカバリープリミティブの多くを自動化します。Barman は、インクリメンタルバックアップ Postgres が提供したブロックレベルのインクリメンタルバックアップに対して、RPO=0 またはほぼゼロを達成するための WAL ストリーミング、バックアップレポート、自動バックアップファイル保持管理、その他の便利な機能などの革新的な機能を提供します。

Barman クラウドユーティリティスクリプト

Barman サーバーとは別に使用できる collection of scripts 。彼らの目的は、Postgres バックアップを S3 互換オブジェクトストレージにコピーすることです。クラウドの Barman は、CNP オペレーターと EDB Cloud Service が Postgres バックアップをオブジェクトストレージにコピーするために使用されます。

圧縮アルゴリズム

バックアップデータの圧縮に使用される方法。サポートされているアルゴリズムは、ベースバックアップと WAL アーカイブで異なります。

- ベースバックアップ none、Snappy、gzip、bzip2
- WAL アーカイブ none、Snappy、gzip、bzip2、lz4、xz、zstd

パフォーマンス特性は、環境とワークロードによって異なります。

ベースバックアップ

Postgres バックアップユーティリティ pg_basebackup Postgres インスタンスの復元に必要なファイルのバックアップまたはファイルコピーを自動化します。

クラウド

パブリッククラウドプロバイダーつまり AWS、Microsoft Azure、および Google Compute PlatformGCP。

クラスター

単一の Postgres インスタンス Postgres クラスターとしても知られるデータベースのコレクション。HM では、展開タイプたとえば、シングルノード、PSR、PGD クラスターなども参照します。

クラウドネイティブ PostgresCNP

open-source Postgres operator EDB によって開発および保守されています。HM の Kubernetes での Postgres の基盤を提供します。

Postgres クラスター

Postgres インスタンスは、Postgres クラスターを管理します。データベースクラスターは、共通のファイルシステムの場所データディレクトリに保存されているデータベースのコレクションです。

pg_dump

データのエクスポートおよびインポートユーティリティ *dump* とも呼ばれます。データベースの一部または全体をエクスポートする機能を提供しますが、エンタープライズクラスのスケーラブルなバックアップツールではありません。ただし、pg_dump は、テストデータベースの作成、データのアーカイブ、データベースから別のデータベースまたは DBaaS プロバイダーへの移行など、さまざまな目的でデータをエクスポートするために広く使用されています。HM ユーザーは pg_dump を使用して Postgres データにアクセスします。pg_dump ユーティリティは Postgres クライアントとともにインストールされ、ユーザーのクライアントマシンから使用できます。

リカバリー

バックアップデータファイルと WAL ファイルに記録された変更を適用することにより、復元データを一貫性のある最新の状態にするプロセス。

目標リカバリポイント RPO

リカバリーに使用できるデータベースタイムラインのポイントまたは時間たとえば 5 分。これは、一般的に「どれくらいのデータを失ってもよいですか?」という質問に答えるものだと言われています。RPO は、時間的に測定された、中断後に組織が耐えることができるデータ損失の最大許容量です。システムとデータを復旧する必要がある時点を定義します。

目標復旧時間 RTO

中断後に組織が耐えることができるダウンタイムの最大許容量。システムとデータを復元する必要がある目標時間を定義します。たとえば、実稼働データベースの RTO は 30 分である場合があります。これは、復元とリカバリーを実行するダウンタイムが 30 分を超えてはいけないことを意味します。

リストア

復旧の準備として、バックアップ場所から元の場所または代替の場所にファイルをコピーするプロセス。例は、バックアップファイルをオブジェクトストレージからデータベースサーバーにコピーすることです。

スナップショット

データのボリュームレベルのスナップショットまたはファイルシステムレベルのバックアップ。スナップショットは、使用可能な最も速く、最もスケーラブルなバックアップおよびリカバリー方法です。スナップショット機能は、HM 展開タイプとトランスポート性などのストレージ機能の可用性によって異なります。

トランスポータブルボリュームのスナップショット

別のストレージシステム、場所、またはクラウドリージョンに移動またはコピーできるデータのポイントインタイムコピー。このコピーにより、ディザスターリカバリー、データ移行、およびオフサイトバックアップが可能になります。

トランスポート不能なボリュームのスナップショット

特定のストレージシステムまたはデータが作成された場所に制限されるデータのポイントインタイムコピー。これは、主にそのシステムのローカルデータの復旧に使用されます。

5.3.3.6.2 Restores

Postgres バックアップの復元

次を使用して復元できます。

- Hybrid ManagerHM コンソール
- API
- CLI

HM コンソールを使用した復元

1. **Backups** タブを選択して、復元可能な既存のバックアップを表示します。
2. 復元するバックアップを選択します。
3. 復元するクラスタータイプを選択します。
4. オプションで、リストアを構成します。
    - **クラスター設定**で、リストアされたクラスターの一意の名前を指定します。
    - **リストアタイプ** の場合、**完全なリストア** または**ポイントインタイムリストアを選択します。**ポイントインタイムリストア**を選択した場合は、日付ピッカーを使用して、復元するポイントを選択します。-** DB 構成 **および**追加設定**では、優先するデータベース構成を選択します。
5. **Restore Cluster** を選択します。復元されたクラスターは、コンソールの **クラスター**リストに表示されます。

API を使用した復元

API を使用して復元するには、次のものが必要です。

- 復元構成を指定する JSON ファイル。
- カールを使用して JSON ファイルを呼び出すシェルスクリプト。

Tip

EDB GitHub repository でポイントインタイムリストアのファイルテンプレートを取得できます。リストアのテンプレートには、*pitr_* プレフィックスが付いています。

API を使用してポイント インタイムに復元するには

1. JSON ファイルで、次のパラメーターを提供します。

`projectID` クラスターの URL から `projectID` を取得します。たとえば、URL が https://portal/projects/prj_6EOsaHpfS8RLshWH/clusters の場合、`projectID` は `prj_6EOsaHpfS8RLshWH` です。

`source_cluster_id` クラスターの URL から `source_cluster_id` を取得します。たとえば、URL が https://portal/projects/prj_6EOsaHpfS8RLshWH/clusters/p-nubw4gnbv9>の場合、`source_cluster_id` は `p-nubw4gnbv9` です。

`point_in_time` クラスターを復元するポイント・インタイムのタイムスタンプ値を提供します。

復元するクラスターに必要に応じて、他の構成パラメーターの追加情報を提供します。例

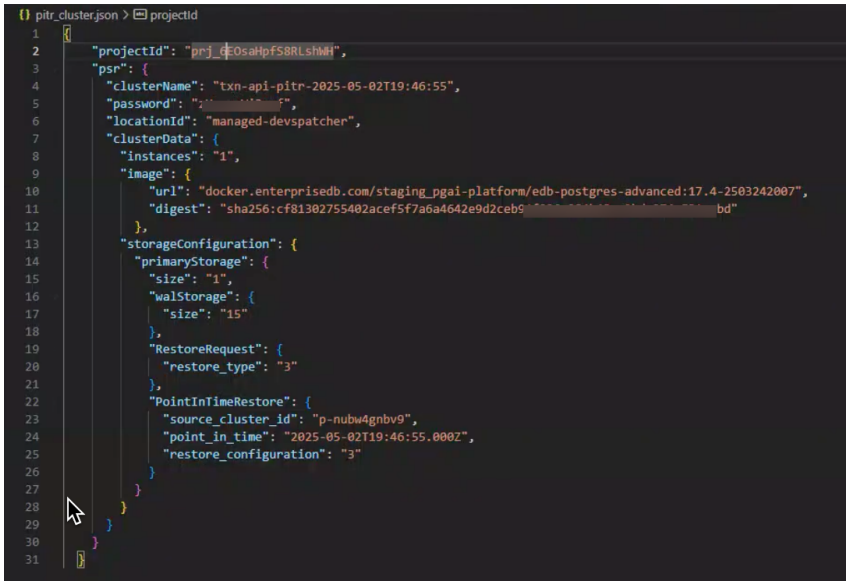
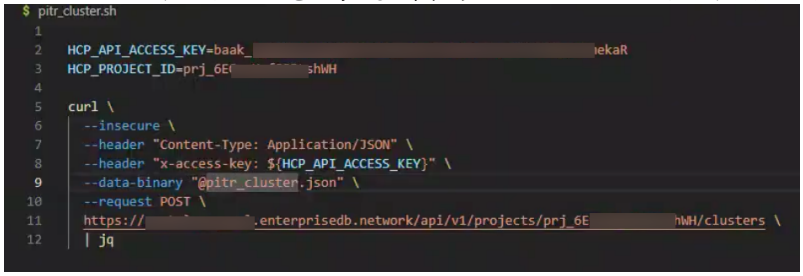


図 7: PGAIHM Restore via API JSON file

1. JSON ファイルを呼び出すシェルスクリプトを作成します。例



2. シェルスクリプトを実行します。スクリプトが出力としてクラスター ID を返した場合、復元されたクラスターはコンソールの**クラスター** リストに表示されます。例

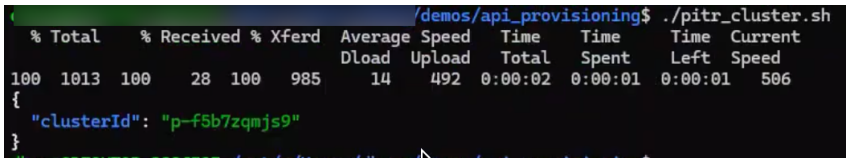


図 8: PGAIHM Restore via API shell file conf

CLI を使用した復元

CLI を使用してポイント インタイムに復元できます。CLI を使用して復元するには、次のものが必要です。

- リストア構成を指定する YAML ファイル。
- CLI で `edbctl` を使用して YAML ファイルを呼び出すには。

1. YAML ファイルで、次のパラメーターを提供します。

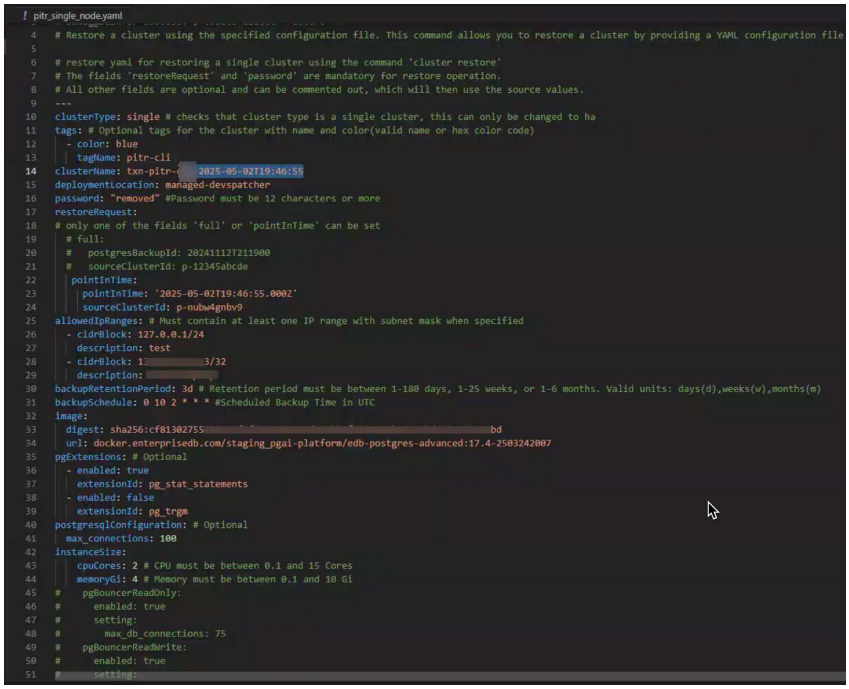


図 9: HM Restore via CLI YAML file

> **Tip**
>
> resume コマンドで使用できるパラメーターの詳細については、 CLI で `edbctl cluster restore -h` を使用してヘルプを表示します。

1. YAML ファイルを指定して CLI コマンドを実行します。クラスターの復元が開始されたことを示すメッセージが返されます。例

図 10: PGAIHM Restore via CLI YAML file

5.3.3.7 EDB Hybrid Manager Chat Agent

EDB Hybrid ManagerHM **Chat Agent** は、わかりやすい言語で HM を操作するのに役立つ会話アシスタントです。資産について質問し、Postgres クラスターをプロビジョニングおよび検査し、アクティブなアラートを検索、パフォーマンスのトラブルシューティング、および HM ドキュメントの参照を行います。すべて HM コンソールを離れることなく。

5.3.3.7.1 サポートされている操作

チャットエージェントは、エステートの参照、クラスターライフサイクル管理、モニタリングとアラート、構成の推奨事項、クラスターヘルスレポート、Oracle から Postgres への SQL 変換、データ移行サービスのステータス、およびドキュメント Q&A をカバーします。すべて HM 権限の範囲内です。

プロンプトとスコープ境界の例を含む完全なカタログについては、*What the Chat Agent can do* を参照してください。

5.3.3.7.2 モデル依存の機能

Chat Agent の実効機能は、管理者が構成するモデルの強度に応じて拡張します。小さなモデルが HM ドキュメントの質問に答えます。強力なモデルでは、エステートの参照、モニタリング、Postgres クラスターライフサイクル管理、クラスター状態レポート、およびスキーマデザインの推奨事項を追加します。モデル選択のガイダンスについては、**モデルの強度階層** を参照してください。

5.3.3.7.3 ドキュメントスコープ

Chat Agent は、各リリースに同梱されているバージョン化されたドキュメントスナップショットから HM の質問に回答します。引用動作と知識カットオフの詳細については、*HM ドキュメント Q&A* を参照してください。

5.3.3.7.4 次のステップ

- *What the Chat Agent can do* - エリアごとにグループ化された完全な機能カタログ、プロンプトの例。
- *Configuring the Chat Agent* —推論サービスの設定とモデルの選択を行う HM 管理者用。
- *Chat Agent security and data handling* —承認モデル、監査、およびデータの処理方法。
- *Troubleshooting Langflow in Hybrid Manager* — 一般的な問題とその解決方法。

What the Chat Agent can do

チャットエージェントは、HM エリアごとにグループ化され、平易な言語でハイブリッドマネージャー HM を操作するのに役立ちます。使用可能な機能は、構成されたモデルの強度層によって異なります。**モデルの強度階層** を参照してください。すべての機能は、HM 権限とプロジェクトスコープを尊重します。*Chat Agent security and data handling* を参照してください。

HM ドキュメント Q&A

モデルサービング、レイクハウス分析、移行など、HM に関する概念的な質問やハウツーの質問に答えます。回答は、各 HM リリースに同梱される公式 HM ドキュメントのスナップショットから得られます。Chat Agent は、読み取ったページを引用します。 Chat Agent がバンドルされたスナップショットで回答を見つ

けられない場合、その旨を示し、公式 EDB Hybrid Manager ドキュメントサイトにリダイレクトされます。ここで、このリリース後に追加されたコンテンツを確認できます。

試してください

- *HM* でログファイルはどのように制御されますか*?*
- *PGD* のコミットスコープとは何ですか*?*いつ使用する必要がありますか*?*

資産、プロジェクト、ユーザー、ロールを管理する

HM エステート、プロジェクト、組織とプロジェクトユーザー、ロールの割り当て、およびアクティビティフィードを参照および管理します。チャットエージェントではユーザーを削除したり、ID プロバイダーを構成したりすることはできません。IdP セットアップには *Configuring your IdP in the HM console* を使用します。

試してください

- プロジェクトをリストします。
- ``*Editor*`` ロールをプロジェクト ``*payments*`` の ``*alice@example.com*`` に追加します。

Postgres クラスターのライフサイクル管理

これらのトポロジでの Postgres クラスターをリスト、検査、作成、削除、および復元します。単一ノード、高可用性 HA、および分散高可用性 DHA。リストアは、ポイントインタイムリカバリー PITR およびターゲットデータベースのリストアをサポートしています。Chat Agent はバックアップを検査し、サポートされている Postgres イメージバージョンをリストします。長時間実行されるアクションはバックグラウンドタスクとして実行されます。Chat Agent はタスクリファレンスを返し、進行状況を確認できます。

試してください

- 開発用の小さな *Postgres* クラスターを作成したいです。
- 最新のバックアップからクラスター ``*orders-prod*`` を復元し、新しいクラスターに ``*orders-restored*`` . という名前を付けます。

クラスターモニタリング、アラート、およびオブザーバビリティ

標準クラスターメトリックの現在または履歴値を取得します。使用可能なメトリックは、CPU、メモリ、ストレージ、I/O、スループット、ネットワーク、接続、トランザクション、クエリレートとレイテンシー、バッファキャッシュヒット率、上位データベース、先行書き込みログ、チェックポイントをカバーします。時間の経過とともにメトリックをグラフとしてレンダリングします。クラスターごとまたはプロジェクトごとにアクティブなアラートを表示します。Chat Agent は、HM コンソールの **クラスターモニタリング**ページをミラーリングします。ダッシュボードの作成または変更、アラートの確認、または沈黙のアラートはありません。これらには HM コンソールを使用します。

試してください

- 過去 *24* 時間に ``*customers-prod*`` の *CPU* 使用率をプロットします。
- クラスター ``*customers-prod*`` のストレージの使用状況はどうですか*?*

Kubernetes ワークロードのオブザーバビリティ

HM の可観測性スタックを介して、クラスターの背後にある Kubernetes ワークロードを検査します。Prometheus からポッドと名前空間のメトリックを照会し、Loki でログを読み取り、使用可能な Grafana データソースをリストします。

試してください

- **名前空間 ``*p-0hrzzicgpq-a*`` の最近のエラーを追跡します。**

推奨事項と勧告読み取り専用

クラスターの構成、インデックス、統計、セキュリティの推奨事項に加えて、プロジェクト全体の集計された状態スコアを取得します。Chat Agent は確認のための推奨事項を浮上させます - HM コンソールから適用します。

試してください

- ``*orders-prod*`` **に保留中のインデックスの推奨事項はありますか***?*
- **クラスター ``*orders-prod*`` のセキュリティ推奨事項の概要です。**

クラスター状態レポート

クラスター状態レポートをリスト、取得、分析、および比較します。HM は、クラスターごとに 1 日 1 つのレポートを生成します。新しく作成されたクラスターには、作成日のレポートがありません。

試してください

- ``*customers-prod*`` **の昨日と今日のヘルスレポートを比較し、何が変わったのかを教えてください。**

スキーマデザインの推奨事項

データ型、正規化、インデックス、パーティショニング、および制約に関するテーブルレベルの推奨事項を取得します。Chat Agent は、テーブルの構造と、それに対して記録されたリソースを消費する上位のクエリー `pg_stat_statements` からを読み取り、Postgres および EDB のベストプラクティスカタログを適用します。記録されたクエリアクティビティがない場合、推奨事項はテーブル定義からのスキーマのみの修正にフォールバックします。

試してください

- ``*customers-prod*`` **、データベース** ``*app*``. のテーブル `public.orders` のスキーマの改善を提案します。

Lakehouse Analytics カタログの参照読み取り専用

プロジェクトに登録されている Iceberg および Lakehouse カタログをリストおよび検査します。HM コンソールからカタログを作成、更新、および削除します。

試してください

- *Iceberg* **のカタログをリストします。**

AI Factory インベントリ読み取り専用

プロジェクト内の AI Factory リソース-登録モデル、推論サービス、ナレッジベース、パイプライン、フローを参照します。Chat Agent は、これらのリソースをリストおよび検査します。HM コンソールからそれらを作成および構成します。

試してください

- **推論サービスをリストします**
- **プロジェクト** *``payments``*. **のナレッジベースをリストします。***

使用状況分析

組織とプロジェクトの集計されたリソース使用分析 Postgres クラスターが消費するコンピューティング時間、メモリ、ストレージなどを確認します。期間ごとの比較を含みます。使用量は、リソース消費のみをカバーし、コストは含まれません。

試してください

- **組織全体で今月使用したコンピューティングとストレージの量はどれくらいですか***?*
- **今月のプロジェクト** *``payments``* **のリソース使用量をクラスター別に表示します。**

Data Migration Service 読み取り専用

既存の移行、ソース、宛先、テーブルマッピング、エラー履歴、および評価レポートを表示します。移行を開始、一時停止、再開、または削除するには、HM コンソールを使用します。Chat Agent は適切なページを示します。

試してください

- **プロジェクト** *``migrations-prod``* **の失敗した移行と、それぞれの最新のエラーを表示します。**

Oracle から Postgres への SQL 変換

Oracle SQL を EDB Postgres 互換 SQL に変換します。Oracle からの移行に先立って、クエリーまたはスキーマ定義を準備する場合に役立ちます。

試してください

- **この** *Oracle SQL* **の翻訳** *``SELECT NVL(name, 'unknown') FROM customers …``*

一般的なコンストレイン

これらのルールはすべての機能に適用されます。

- **HM のみの範囲。**トピック外の質問一般知識、数学、その他の製品は範囲外です。
- **確認ゲート。** 作成、更新、削除、ドロップ、DDL、および DML 操作は、実行前に明示的な *yes* が必要です。
- **Permission-honest.** permission- denied エラーが発生すると、チャットエージェントは再試行または回避するのではなく、停止してエラーを報告します。

- **ステータスはステータスチェックから来ています。**リソースのリストは、それが正常またはアクティブであることを意味するものではありません— Chat Agent は、必要に応じてステータスエンドポイントを呼び出します。

これらの保証の背後にあるセキュリティモデルについては、*Chat Agent security and data handling* を参照してください。

関連項目

- *Configuring the Chat Agent* —モデル構成、サポートされているプロバイダー、および階層選択。
- *Chat Agent security and data handling* —これらの機能の背後にある承認モデル、監査、およびデータ処理。
- *Troubleshooting Langflow in Hybrid Manager* — 一般的な問題とその解決方法。

Configuring the Chat Agent

EDB Hybrid ManagerHM チャットエージェントは、HM とともに配信され、HM 管理の推論サービスを介して提供するチャット完了モデルを利用しています。HM 管理者として、そのモデルの推論サービスを作成または登録します。その後、組織内の誰もが HM コンソールから Chat Agent を使用できます。

このページは HM 管理者用です。 Chat Agent の使用方法については、**概要** を参照してください。

始める前に

これらの前提条件を確認します。

- HM がインストールされ、到達可能。
- **AI シナリオ**が HM 展開で有効になっています。チャットエージェントは AI シナリオの一部であり、AI シナリオがインストールされていない場合、輝きのアイコンは非表示になっています。これを有効にするには、`HybridControlPlane` マニフェストの `scenarios` の下に `ai` がリストされていることを確認します。詳細は、*HM installation scenarios* を参照してください。
- HM AI Factory が利用でき、HM コンソールからモデルサービングとモデルライブラリーに到達できます。AI Factory のドキュメントについては、**モデルサービング** および **ステップ 2 モデルライブラリーにアクセスする** を参照してください。
- **AI Model Manager** 組織ロールを持っています。このロールでは、AI モデルの登録、モデル機能の適用、および推論サービスの作成ができます。完全なロールマトリックスについては、*User role permissions by use case* を参照してください。
- 少なくとも 1 つのチャット完了モデルサードパーティ API またはセルフホストエンドポイントの資格情報と接続の詳細がある。**サポートされているモデルプロバイダー** を参照してください。
- チャット完了モデルは、 **少なくとも 128K トークン**のコンテキストウィンドウをサポートしています。 Chat Agent は、ターンごとにスキル指示、ツールスキーマ、および会話履歴をロードします。コンテキストウィンドウが小さいモデルには、十分なコンテキストがありません。

重要

Chat Agent は、ライブ HM エステートに関する質問に回答します。データ処理ポリシーが組織の要件と一致するプロバイダーを選択します。セルフホストモデルは、プロンプトと応答をインフラストラクチャ内に保持します。

Chat Agent は、HM 管理の推論サービスを介して提供されるチャット完了モデルを利用しています。チャットエージェントでモデルを利用できるようにするには、HM コンソールで **Estate > Inference Services** に移動し、 **Quick Actions** メニューを開き、モデルが実行される場所に応じて、次の 2 つのパスのいずれかを選択します。

- **ローカル推論サービスの作成** — HM ローカル **アセットライブラリー**からモデルを展開および提供します。プロンプトと応答をインフラストラクチャ内に保持したい場合に推奨します。
- **外部推論サービスの登録** —クラスターの外部で実行されているモデルエンドポイントを HM に向けます。例には、サードパーティのクラウド API OpenAI、Anthropic、Gemini または別の HM クラスター上のセルフホストエンドポイントが含まれます。すでに他の場所でモデルを操作しており、HM で使用したい場合にお勧めします。

ローカル推論サービスの作成

モデルを HM **アセットライブラリー**に追加し、そのエントリから推論サービスを作成します。

1. **Asset ライブラリー > Models** で、**+ Add New Model** を選択し、モデルの詳細を入力します。 ``**chatbot-gen-content**`` を **Functions** フィールドに追加します。完全なフォームリファレンスについては、*How to deploy AI Models from the Model Library* を参照してください。
2. **Estate > Inference Services** に移動します。
3. **Quick Actions** から、**Create Local Inference Service** または空の状態の **Create Local Inference Service** ボタンを選択します。
4. 前の手順で追加したモデルを選択し、展開フォームに記入します。
5. 推論サービスが **Ready** 状態に到達するまで、送信して待機します。
6. (オプション) このモデルのチャットエージェントの動作をカスタマイズするには、追加のモデル機能を適用します。**モデルファンクションの構成** を参照してください。

次回ユーザーが Chat Agent を開くと、モデルがモデルセレクターにリストされます。

外部推論サービスの登録

HM を、クラスターの外部で実行されているモデルエンドポイント-たとえば、OpenAI の API、Anthropic 互換エンドポイント、Google Gemini エンドポイント、または別の HM クラスター上のセルフホストエンドポイントなどに向けます。

1. **Estate > Inference Services** に移動します。
2. **Quick Actions** から、 **Register external Inference Service** を選択します。
3. 外部エンドポイントの接続の詳細を提供します。必須フィールドは、URL、API プロトコル OpenAI v1、Anthropic Messages、または Google Gemini、および API キーです。
4. ``**chatbot-gen-content**`` ファンクションが適用されていることを確認します。

5. 登録が **Ready** 状態に達するまで、送信して待ちます。
6. (オプション) このモデルのチャットエージェントの動作をカスタマイズするには、追加のモデル機能を適用します。**モデルファンクションの構成** を参照してください。

外部サービスの詳細については、AI Factory ドキュメントの **モデルサービング** を参照してください。

モデルファンクションの構成

このオプションのステップは、**ローカル推論サービスの作成** および *HM* **への外部推論サービスの登録** の最後で参照されます。これらの機能は、適用されているモデルのチャットエージェントの動作を変更します。**アセットライブラリー>モデル** で、モデルの **ファンクション**フィールドを編集して、それらを適用または更新します。

Function	Value	Purpose
chatbot-default-thinking-level	*low*, *medium*, or *high*	Sets the model's default reasoning effort.
chatbot-model-tier	*S*, *M*, *L*, or *XL*	Overrides the auto-detected strength tier.

思考または推論モードをサポートするモデルでのみ `chatbot-default-thinking-level` を設定します。設定しない場合、モデルは独自のデフォルトの思考レベル通常は `medium` 、モデルの仕様ごとに。このデフォルトは、ほとんどのタスクでうまく機能します。複雑なタスクで一貫性のないチャットエージェントの動作が表示されない場合、またはより迅速な応答が必要な場合を除き、ファンクションを設定しないままにしてください。より複雑なタスクには `high` に上げますか、応答を高速化するには `low` に下げます。

> **警告**
>
> 一部の推論モデルは、要求ごとの思考レベル設定を受け入れません。これらのモデルでは、 *chatbot-default-thinking-level* を適用すると、モデルがエラーを結果、Chat Agent が応答を停止します。この機能を設定する前に、モデルが選択したレベルを受け入れられることを確認してください。

構成の検証

1. HM コンソールで、少なくとも 1 つのプロジェクトに属するユーザーとしてサインインします。
2. 右上の sparkle アイコンを選択します。
3. 入力にヘッダー **HM AI** とプレースホルダー **Ask me anything**…が表示されたパネルが開くことを確認します。
4. 複数のモデルを登録した場合は、 **モデル**ピッカーを選択し、モデルが表示されることを確認します。
5. 次のような簡単なプロンプトを送信します。*What can you do?*
6. ストリーミング応答が数秒以内に始まることを確認します。

スパークルアイコンが見つからない場合、またはパネルを開くことができない場合は、*Troubleshooting Langflow in Hybrid Manager* を参照してください。

チャットエージェントへのアクセス

チャットエージェントは、HM コンソールのすべての認証されたページで利用できます。

1. HM ヘッダーの右上にある虹の輝きアイコンを探します。
2. アイコンを選択して Chat Agent を開きます。

アイコンが見つからないまたは無効になっている場合は、*Troubleshooting Langflow in Hybrid Manager* を参照してください。

Chat Agent は 2 つのレイアウトで開きます。

- **フルスクリーン** —デフォルト。会話リストが左側に表示されます。
- **フローティングパネル** —HM コンソールの右下にドッキングされたコンパクトなドラッグ可能なパネル。展開または縮小ボタンを使用して、モードを切り替えます。

チャットエージェントを使用する

質問する

1. 管理者が複数のモデルを登録している場合、ピッカーから **モデル**を選択します。モデルは、どの機能を使用できるかを決定します。**モデル依存の機能** を参照してください。
2. **Ask me anything** 入力に質問を入力し、 **Enter** を押します。
3. **Shift+Enter** を使用して、メッセージ内に改行を追加します。
4. Chat Agent はチャットウィンドウに応答をストリーミングします。ドキュメントの回答にはインライン引用が含まれるため、ソースページを直接開くことができます。
5. stop ボタンを選択して、生成を早期に中断します。

要求を完了する前に Chat Agent が詳細情報たとえば、ターゲットプロジェクトや接続パスワードなどを必要とする場合、Chat Agent は一時停止し、構造化フォームを表示します。フォームに入力して続行するか、キャンセルまたは無関係のメッセージを入力して閉じます。

ページコンテキストを使用する

クラスター、プロジェクト、またはその他のリソースページから Chat Agent を開くと、そのページのコンテキストが自動的に取得されます。プロジェクトとクラスター ID は入力の上に小さなタグとして表示されるため、名前を付けずにそのリソースについて質問できます。たとえば、_Show me the active attempts_は、現在表示しているクラスターに適用されます。

会話を管理する

各会話は HM 組織に保存されるため、後で戻ることができます。 **New Chat** を選択して新しい会話を開始するか、メニューアイコンでサイドバーを開き、以前のチャットに戻ります。

チャットエージェントは、新しい会話ごとに短い自動生成のタイトルを付けます。サイドバーから、会話の**名前を変更** したり、 **削除**したりすることもできます。削除は元に戻すことはできません。

会話のプライバシーの詳細については、 **会話のストレージと分離** を参照してください。

おすすめ

- 大規模モデルから始める —大規模は、構造化フォームと中層機能のフルセットで Postgres クラスターライフサイクル管理をカバーする最下位層です。`gemini-2.5-pro` 、`gpt-5` 、および `gpt-oss-120b` はこの層にあります。
- プロンプトに規制されたデータまたは独自のデータが含まれる場合、セルフホストモデルを選択します。この選択により、チャットエージェントのプロンプトと応答がインフラストラクチャ内に保持されます。
- ユーザーが会話ごとに層を選択できるように複数のモデルを登録します。たとえば、ドキュメントルックアップ用に小さいモデルを登録し、クラスター作成フロー用に大きいモデルを登録します。

サポートされているモデルプロバイダー

チャットエージェントは、3 つの API プロトコルを介してモデルと通信します。これらのプロトコルのいずれかを通信するプロバイダーサードパーティまたはセルフホスト型は動作します。EDB は、AI モデルを構築または出荷しません。すべての HM ホストモデルは、`gpt-oss-20b` や `nvidia/llama-3.3-nemotron-super-49b-v1.5` など、Hugging Face または NVIDIA から HM によってダウンロードされたサードパーティモデルです。

Protocol	Compatible providers and runtimes
OpenAI v1	OpenAI, Azure OpenAI, NVIDIA NIM containers, Amazon Bedrock (via OpenAI-compatible gateway), vLLM and other OpenAI-compatible self-hosted servers
Anthropic Messages	Anthropic API, Anthropic-compatible gateways and proxies that preserve the Messages API surface
Google Gemini	Google AI Studio (Gemini API), Vertex AI with the Gemini API surface

モデルの強度階層

Chat Agent は、モデルの機能に基づいて、登録されたすべてのモデルを**強度層**小、中、大、または特大に分類します。モデルの層により、そのモデルが選択されたときに使用可能なチャットエージェントスキルが決まります。

強さ段階

4 つの層は、大まかなモデルのスケールと機能に対応しています。

- Small — 約 7〜30B のパラメータークラス。
- Medium — 約 30〜100B パラメータークラス。
- Large — `gpt-4.1` 、`gemini-2.5-pro` 、`gpt-5` 、および `gpt-oss-120b` クラス。
- Extra Large — 現在のフラッグシップモデル クロード 4.5+ファミリー `claude-haiku-4-5` を含む `gpt-5.2` 以降、`gpt-5.3-codex` 、`gemini-3.1-pro-preview` 、および同等のフロンティアモデルを含む。

以下の認識されたリストにないモデルは、デフォルトの Small になります。

認識されたモデル

Model	Tier
gemini-2.5-flash-lite	Small
gemini-3.1-flash-lite	Small
gpt-4.1-nano	Small
gpt-4o	Small
gpt-4o-mini	Small
gpt-5-nano	Small
gpt-oss-20b	Small
nvidia/llama-3.3-nemotron-super-49b-v1	Small
nvidia/llama-3.3-nemotron-super-49b-v1.5	Small
nvidia/NVIDIA-Nemotron-3-Nano-30B-A3B-BF16	Small
nvidia/Nemotron-3-Nano-Omni-30B-A3B-Reasoning-BF16	Small
nvidia/Nemotron-Cascade-2-30B-A3B	Small
gemini-2.5-flash	Medium
gpt-4.1-mini	Medium
gpt-5-mini	Medium
gpt-5.4-nano	Medium
o4-mini	Medium
qwen/qwen3-next-80b-a3b-instruct	Medium
qwen/qwen3-next-80b-a3b-thinking	Medium
gemini-2.5-pro	Large
gemini-3-flash-preview	Large
gpt-4.1	Large
gpt-5	Large
gpt-5.1	Large
gpt-5.4-mini	Large
gpt-oss-120b	Large
o3	Large
nvidia/llama-3.1-nemotron-ultra-253b-v1	Large
nvidia/NVIDIA-Nemotron-3-Super-120B-A12B-BF16	Large
qwen/qwen3.5-122b-a10b	Large
claude-haiku-4-5	Extra Large
claude-sonnet-4-5	Extra Large
claude-sonnet-4-6	Extra Large
claude-opus-4-1	Extra Large
claude-opus-4-5	Extra Large
claude-opus-4-6	Extra Large
claude-opus-4-7	Extra Large
gemini-3.1-pro-preview	Extra Large
gpt-5.2	Extra Large
gpt-5.3-codex	Extra Large
gpt-5.4	Extra Large

次のページに続く

表 59 – 前のページからの続き

Model	Tier
gpt-5.4-pro	Extra Large
gpt-5.5	Extra Large
gpt-5.5-pro	Extra Large
deepseek-ai/deepseek-v4-flash	Extra Large
deepseek-ai/deepseek-v4-pro	Extra Large
meta/llama-4-maverick-17b-128e-instruct	Extra Large
minimaxai/minimax-m2.7	Extra Large
qwen/qwen3-coder-480b-a35b-instruct	Extra Large
qwen/qwen3.5-397b-a17b	Extra Large
z-ai/glm5	Extra Large
z-ai/glm-5.1	Extra Large

次のエイリアスは、同じ正規モデルとして認識されます。

- Amazon Bedrock の人体モデル — クロスリージョンプレフィックス `us.`、`eu.`、`jp.`、`au.`、`global.` およびプロバイダープレフィックスバリアント `anthropic.<model>-<date>-v1:0`。
- OpenAI `gpt-oss-20b` および `gpt-oss-120b` —プロバイダープレフィックス付きバリアント `openai/<model>`、`openai.<model>-1:0`、`gpt-oss:<size>`。

未認識のモデルへの階層の割り当て

名前が上記の表の行と一致せず、エイリアスルールの対象になっていないモデルは、デフォルトで Small になります。その結果、より高度なスキルの多くが使用できなくなります。

この設定をオーバーライドするには、次のいずれかの値を使用して ``**chatbot-model-tier**`` ファンクションをアセットライブラリーのモデルに追加します。

Function value	Tier
S	Small
M	Medium
L	Large
XL	Extra Large

モデルの実際の機能に最適な値を使用します。保存後、Chat Agent はデフォルトの Small に設定される代わりに、設定された階層を優先します。

各ティアで利用可能なスキル

スキルは、選択したモデルのティアがスキルの最小ティア以上の場合に利用できます。上位層は、既存の機能を削除せずに機能を追加します。

Skill / capability	Minimum tier
HM documentation Q&A (concepts, architecture, how-to, with citations)	Small
Browse and manage estate, projects, organization users, project users, organization roles, and project roles	Medium
Project and organization activity feeds	Medium
Iceberg and Lakehouse catalog inventory (list and inspect, read-only)	Medium
Kubernetes workload observability — Grafana data sources, Prometheus queries, Loki queries	Medium
Cluster realtime and historical metrics, charts, and available metric metadata	Medium
Active alerts (per cluster, per project), aggregated health scores	Medium
Configuration, index, statistics, and security recommendations	Medium
Data Migration Service status — sources, destinations, mappings, errors, assessments	Medium
Oracle-to-Postgres SQL translation	Medium
Resource-usage analytics (compute, memory, and storage consumption)	Medium
Postgres Cluster lifecycle management	Medium
AI Factory inventory — models, inference services, knowledge bases, pipelines, and flows	Large

次のページに続く

表 61 – 前のページからの続き

Skill / capability	Minimum tier
Image library and deployable Postgres versions per location	Large
Read-only database, schema, and cluster-configuration inspection	Large
Cluster health reports — list, fetch, analyze, compare	Large
Postgres Cluster lifecycle management with structured follow-up questions	Large
Schema design recommendations driven by top running queries	Extra Large

Small モデルでは、HM ドキュメント Q&A のみを提供します。Postgres クラスターのライフサイクル管理は中層から始まり、大規模以上に構造化されたフォローアップの質問が追加されます。クラスター状態レポートには大規模モデルが必要であり、スキーマデザインの推奨事項には特大モデルが必要です。各層のサポート内容の完全な内訳については、*What the Chat Agent can do* を参照してください。

組織のチャットエージェントの無効化

HM コンソールから Chat Agent を削除するには、`chatbot-gen-content` ファンクションを備えているすべてのモデルから `chatbot-gen-content` ファンクションを削除します。チャットエージェントプロバイダーが登録されない場合、スパークルアイコンは表示されますが無効になり、ツールチップはモデルサービスを構成するようにユーザーに指示します。

スパークルアイコンを完全に削除するには、インストール時に AI シナリオを無効にします。*HM installation scenarios* を参照してください。

次のステップ

- *What the Chat Agent can do* —エリアごとに編成されたユーザーに表示される機能カタログ。
- *Chat Agent security and data handling* — 承認モデル、監査、およびデータ処理。
- *Troubleshooting Langflow in Hybrid Manager* —一般的な構成の問題とその解決方法。

Chat Agent security and data handling

ハイブリッドマネージャー HM チャットエージェントは、HM コンソールを駆動するのと同じ API の利便性レイヤーです。新しい特権モデルを導入せず、独自の権限で動作しません。

承認モデル

Chat Agent は特権サービスアカウントではありません。 HM ID とプロジェクトコンテキストは、それが行うすべての要求とともに移動します。Chat Agent は、認証されたユーザーとして、既に呼び出すことができるのと同じ API に対して実行されます。表示できないリソースを表示したり、実行権限のないアクションを実行したりすることはできません。検証されたユーザー ID のない要求は拒否されます。

組織が権限を削除したり、プロジェクトメンバーシップを取り消したり、ユーザーを無効にしたりすると、これらの変更は Chat Agent にすぐに適用されます。 Chat Agent 固有の構成は必要ありません。

シークレットの取り扱い

Chat Agent は入力としてシークレットが必要な場合、非表示の入力を使用した構造化フォームを介して値を収集します。例は、Postgres クラスターのライフサイクル管理中のデータベースパスワードです。フォーム収集された値は HM 内に残ります。HM はそれをチャット完了モデルに送信したり、会話のトランスクリプトに含めたりすることはありません。

この保証は、構成されたモデルがラージまたはエクストララージティアの場合にのみ適用されます。これらの層は、パスワードなどの入力に対して構造化されたフォローアップの質問をします。下位層では、Chat Agent は自由形式のプロンプトを使用します。チャットに入力したシークレットは、プロンプトの残りの部分とともにモデルをパススルーします。**各ティアで利用可能なスキル** を参照してください。

監査ログ

チャットエージェントが実行するすべてのアクションは、HM コンソールおよび API アクティビティとともに HM アクティビティログに記録されます。**アクティビティログ** を参照してください。

会話のストレージと分離

会話は HM 組織内に保存されるため、後で再開できます。各会話はその作成者にプライベートです。組織の所有者やプロジェクト管理者を含む他の人は、Chat Agent またはその API を介してそのメッセージをリスト、ロード、またはダウンロードすることはできません。HM アクティビティログには、プロンプトや応答ではなく、アクションが記録されます。

チャットエージェントの会話サイドバーから会話を削除し、自動生成された件名の名前を変更できます。会話を削除した後、復元することはできません。

クラスターの詳細ページから実行したアドホック SQL クエリーは保存されません。ページを更新するか閉じると、これらはクリアされます。

プロンプトとモデルプロバイダー

プロンプトとチャットエージェントの応答は、管理者が構成したチャット完了モデルを通過します。セルフホストモデルでは、どちらもインフラストラクチャ内にあります。サードパーティモデルでは、インフラストラクチャを離れてベンダーに到達します。データ処理と保持ポリシーが組織の要件と一致するプロバイダーを選択します。

モデルをデータパスの一部として扱います。構造化フォームを介して収集された機密値は HM 内にあります。**シークレットの取り扱い** を参照してください。チャットに入力したその他の内容はモデルに到達します。

知識のカットオフ

Chat Agent は、各リリースに同梱されているバージョン化されたドキュメントスナップショットから HM の質問に回答します。スナップショットの仕組み、引用動作、および知識カットオフの処理については、*HM ドキュメント Q&A* を参照してください。

関連項目

- *What the Chat Agent can do* — スコープ境界を含む完全な機能カタログ。
- *Configuring the Chat Agent* —セルフホストとサードパーティプロバイダーの考慮事項を含むモデル構成。
- *Troubleshooting Langflow in Hybrid Manager* — 一般的な問題とその解決方法。

Chat Agent schema design recommendations

Hybrid ManagerHM Chat Agent は、テーブルスキーマとクエリワークロードを分析し、スキーマデザインの改善を提案できます。推奨事項は証拠ベースであり、Postgres および EDB のベストプラクティスに基づいています。

レコメンデーションは会話形式で配信されます。特定のテーブルについてチャットエージェントに質問すると、実行可能な提案が返されます。スキーマデザインの推奨事項は、[**Recommendations**] タブには表示されません。チャットエージェントを介してのみ利用できます。

前提条件

- ターゲットクラスターが HM に登録されている。HM 管理対象クラスターと自己管理クラスターの両方がサポートされています。
- クラスターでモニタリングが有効になっています。チャットエージェントは、管理対象クラスタと自己管理クラスタの両方で使用できる HM の監視データを介してクエリ統計とスキーマメタデータを読み取ります。データを監視しない場合、推奨事項はスキーマのみの分析にフォールバックします。

レコメンデーションタイプ

Chat Agent は、管理対象クラスターと自己管理クラスターの両方の DDL レベルの向上に焦点を当てています。データにアクセスするクエリーの変更ではなく、データの定義および保存方法の変更です。Postgres および EDB のベストプラクティスカタログを適用するため、返される提案はテーブルとそのワークロードによって異なります。共用エリアには次のものが含まれます。

- 正規化と構造 —パフォーマンスや保守性に影響を与える可能性のある繰り返しグループ、部分的な依存関係、または過度の非正規化などの問題にフラグを立てます。
- データ型 — 列のより適切なタイプを提案します。たとえば、`TIMESTAMP` よりは `TIMESTAMPTZ` 、money は `NUMERIC` 、または `CHAR(n)` /`VARCHAR(n)` よりは `TEXT` です。
- コンストレインとキー — 主キー、外部キー、`NOT NULL` 、`CHECK` 、および `UNIQUE` コンストレインが欠落している場合、または強化の可能性がある場合に推奨します。
- インデックス付け —アクセスパスの欠落しているインデックスインデックスなしの外部キー列などと適切なインデックスタイプを特定し、未使用または冗長なものにフラグを立てます。

- パーティショニング —クエリパターンとデータ分散に基づいて、レンジ、リスト、またはハッシュパーティショニングから利益を得るテーブルを特定します。
- ワークロード固有のチューニング — `fillfactor` 設定やホット列とコールド列の分離など、更新が多い、挿入が多い、または更新/挿入指向のテーブルの調整。
- EDB 拡張機能 —価値を追加する場合、ユースケースに合わせた EDB 拡張機能をお勧めします。たとえば、分析または追加専用ワークロードの EDB Advanced Storage Pack テーブルアクセス方法。

厳選された自動適用可能なインデックスの推奨事項については、**推奨事項と勧告読み取り専用** を参照してください。

スキーマの推奨事項の仕組み

スキーマの推奨事項を聞いたとき、チャットエージェントは次のことを行います。

- テーブルの DDL 列定義、制約、インデックス、およびストレージ設定を読み取ります。
- テーブルに対して実行されている上位のクエリーを取得します。デフォルトでは、過去 6 時間を見ます。プロンプトで別のウィンドウを指定できます。
- Postgres および EDB ベストプラクティスカタログをスキーマとクエリデータに適用します。
- 具体的な証拠に基づいた推奨事項をそれぞれの説明とともに返します。

指定されたウィンドウにクエリワークロードがない場合、Chat Agent はテーブル定義のみから派生したスキーマのみの推奨事項にフォールバックします。

プロンプトの例

- **``*customers-prod*``、データベース ``*app*``.** のテーブル `public.orders` のスキーマの改善を提案します。
- **過去 *24* 時間のクエリデータを使用して、``*orders-prod*`` の ``*public.line_items*`` のスキーマを確認します。**
- **``*analytics-db*`` の ``*public.events*`` にパーティショニングの機会はありますか***?***

制限事項

- 推奨事項は DDL レベルの改善のみをカバーしています。クエリレベル DML の改善は範囲外です。
- スキーマの推奨事項は自動適用されません。インデックスの推奨事項とは異なり、手動または独自の展開パイプラインを介して実装する必要があります。
- 推奨事項は [**推奨事項**] タブに表示されず、クラスター状態スコアにも含まれません。
- クエリデータが増えると精度が向上します。ワークロード履歴がほとんどない、新しく作成されたテーブルまたはクラスターの場合、推奨事項はスキーマ構造のみに基づいています。

Troubleshooting the Chat Agent

これらの修正を使用して、ユーザーと管理者のための一般的なハイブリッドマネージャー HM チャットエージェントの問題を解決します。ここで問題が表示されない場合は、HM コンソールに表示される要求トレース ID を使用して EDB サポートに連絡してください。

スパークルアイコンは表示されません

問題 レインボースパークルアイコンが HM ヘッダーからありません。

原因 AI シナリオがインストールされていません。 Chat Agent は AI シナリオの一部です。

修正: AI シナリオがインストールまたはアップグレード時に選択されたことを HM 管理者に確認してください。*HM installation scenarios* を参照してください。

スパークルアイコンはツールチップでは無効になっています

問題アイコンは表示されますが、グレー表示されています。カーソルを合わせると、**管理者はチャットエージェントを有効にするように推論サービスを構成する必要があります**—または別の理由などのツールチップが表示されます。

原因 1: チャット完了モデルは、チャットエージェントプロバイダーとして登録されていません。

修正 1: 管理者はチャット完了モデルを登録し、`chatbot-gen-content` ファンクションを適用する必要があります。*Configuring the model* を参照してください。

原因 2: 登録された推論サービスの準備ができていません。

修正 2: Estate > Inference Services を開き、**Status** 列に **Ready** と表示されていることを確認します。チャットエージェントは、**Ready** 状態の推論サービスのみを表示します。他の状態たとえば **Unknown** のサービスは、Chat エージェントで使用できません。

原因 3: Chat Agent サービスは利用できません。

修正 3: ツールチップには理由が含まれています。少し待って再試行するか、サポートする HM ワークロードが正常であることを管理者に確認してください。

生成がストールまたは失敗する

問題 **Waiting**…プレースホルダーバブルが長時間残るか、応答が `Request failed, please try again!` で終了する

原因 1: 構成済みモデルは到達不能か、使用制限に到達しています。

修正 1: 複数のモデルが登録されている場合、モデルセレクターから別のモデルを試してください。それ以外の場合は、**AI Factory > Model Serving** *Monitor InferenceService* でモデルサーバーのステータスを確認します。

原因 2: プロンプトが長すぎます。各ユーザーメッセージの上限は 10,240 文字です。

修正 2: プロンプトをトリミングして、再試行します。

原因 3: ネットワーク外向きがブロックされています。

修正 3: サードパーティモデルの場合、HM クラスターにプロバイダーの API へのアウトバウンドアクセスがあることを確認します。

原因 4 モデルは、構成された `chatbot-default-thinking-level` を拒否します。すべての推論モデルが要求ごとの思考レベル設定を受け入れられるわけではないため、いずれかが渡されるとターンは失敗します。

修正 4: モデルの **ファンクション**フィールドから `chatbot-default-thinking-level` を削除してモデルのデフォルトにフォールバックするか、要求ごとのシンキングレベル構成をドキュメントでサポートするモデルに切り替えます。**モデルファンクションの構成** を参照してください。

Chat Agent が私の質問に答えてくれません

問題 チャットエージェントは、HM トピックでのみ役立つと言います。

原因チャットエージェントは HM 関連の質問にのみ回答します。トピック外の質問一般知識、数学、クリエイティブライティング、およびその他の製品は、範囲外です。

修正: あなたの質問を HM エステート、HM 機能、または HM ドキュメントに関するものに言い換えます。

チャットエージェントはアクションを実行できないと言います

問題 チャットエージェントが `403` または `not authorized` エラーを報告して停止するか、アクションを実行する権限がないと言います。

原因 Chat Agent は自分のユーザーアカウントとして実行されます。HM コンソールでアクションを実行するロールがない場合、Chat Agent は同じ回答を取得します。

修正します

- 適切な組織またはプロジェクトのロールを保持していることを管理者に確認します。
- アクションの対象となるプロジェクトのメンバーであることを確認します。
- 承認ワークフローを必要とするアクションタイプ一部の自動タスクなどの場合、管理者が承認するまで待ちます。

承認モデル を参照してください。

Chat Agent がクラスターまたはプロジェクトを認識しない

問題チャットエージェントは、存在することがわかっているプロジェクトまたはクラスターが見つからないと言います。

原因 1: あなたは、リソースを所有するプロジェクトのメンバーではありません。 Chat Agent は、アクセスできるプロジェクト内のリソースのみを参照します。

修正 1: 管理者にプロジェクトに追加するように依頼します。

原因 2 リソース名のスペルを間違えました。リソース名はプロジェクト内で一意です。Chat Agent は、最初にリソースをリストすることにより名前を ID に解決しますが、タイプミスにより一致するものが見つかりません。

修正 2: HM コンソールからリソース名をコピーし、再試行します。

チャットエージェントは、トピックのナレッジより前の日付を言います

問題チャットエージェントは、あなたが質問したトピックは、バンドルされたドキュメントのスナップショットより新しいと言います。

原因各 HM リリースには、ナレッジカットオフ日がスタンプされた公式 HM ドキュメントのスナップショットが同梱されています。Chat Agent は、ドキュメントスナップショットに回答が含まれていない場合にレポートします。

修正新しい情報については、公式 EDB Hybrid Manager ドキュメントサイトを参照してください。より新しい製品内スナップショットの恩恵を得るには、HM をより新しいリリースにアップグレードします。

チャットエージェントはフォローアップの質問をし続けます

問題 チャットエージェントは、要求を完了する前に詳細情報を求め続けます。

原因一部の Chat Agent フローは、事前定義された入力シーケンスに従って、特定のタスクを完了します。Postgres クラスターのライフサイクル管理は一例です。Chat Agent が必要なパラメータをすべて持っていない場合、不完全なデータを推測したり続行したりするのではなく、必要になるまで質問し続けます。

修正: 要求された情報を入力して続行するか、フォームをキャンセルするか、チャットに無関係のメッセージを入力します。質問がクリアされます。

チャートはレスポンスでレンダリングされません

問題応答には、レンダリングされたチャートの代わりに生のチャート構文または不正なプレースホルダーが表示されます。

原因チャート生成では、モデルが正確な構造化出力を発行する必要があります。小さなモデルは多くの場合、構文に問題があり、レンダラーが表示できないチャートを生成します。

修正: 管理者が複数のモデルを登録している場合、モデルピッカーから Large または Extra Large モデルに切り替えます。それ以外の場合は、管理者により強力なモデルを登録するように依頼します。**モデルの強度階層** を参照してください。

能力の低下

問題チャットエージェントはクラスター作成を提供せず、構造的なフォローアップの質問をせず、これまでに見た完全なインタラクティブフローを開始しません。

原因チャットエージェント機能は、構成されたモデルの強度に応じてスケーリングします。Small または Medium モデルは、フロンティアモデルよりも少ないフローを処理し、構造化されたフォローアップフォームはモデルがサポートしている場合にのみ使用されます。

修正: 管理者が複数のモデルを登録している場合、モデルピッカーからより強いモデルを選択するか、管理者により強いモデルを登録するように依頼します。**モデルの強度階層** を参照してください。

サポートへの連絡

問題が解決しない場合は、EDB Support に連絡する前にこれらの詳細をキャプチャしてください

- HM トラブルシューティング情報 — HM コンソールフッターから **About** を開き、 **Copy parameters** を選択して、バージョン、ビルド、シナリオ、組織、セッション、およびプラットフォームの詳細をキャプチャします。
- 使用中のモデルとモデルプロバイダー既知の場合。
- 送信したプロンプトと受け取った応答のコピー。

5.3.3.8 Governance in Hybrid Manager

EDB Agent Governance は、AI エージェントが Postgres データと相互作用する方法を監査および制御するスタンドアロンアプリケーションです。エージェントセッションを再構成し、エージェントが実行したすべての SQL ステートメントを検査し、完全な監査証跡を表示します。複数の上流ログソースで動作します。Hybrid ManagerHM もその 1 つです。

このページでは、HM 固有の統合のみを説明します。アプリケーション自分自身の使用方法データソースの接続、セッションリストの読み取り、セッションの詳細の検査、および展開については、

EDB Agent Governance documentation を参照してください。

5.3.3.8.1 HM が提供するもの

HM インスタンスで EDB エージェントガバナンスをポイントすると、HM は、アプリケーションが AI エージェントアクティビティを検出して読み取るために必要な部分を提供します。

- **プロジェクトとクラスターの検出** - アプリケーションは、HM データプレーンでプロジェクトと Postgres クラスターを自動検出するため、手動構成なしで監査ターゲットとして表示されます。
- **HM の Loki パイプラインを介したログアクセス** — HM は、モニタリングの実行 の一部として Postgres ログを Loki に収集しています。アプリケーションは、同じパイプラインを照会して、AI エージェントが生成するクエリログを取得します。
- **マシンユーザー API キー認証** —アプリケーションはマシンユーザー API キーを使用して HM に対して認証するため、そのキーに読み取りが許可されているもののみを読み取ります。

5.3.3.8.2 前提条件

HM 管理クラスターの AI エージェントアクティビティを EDB エージェントガバナンスに表示するには

- クラスターでは Postgres ステートメントログが有効になっています `log_statement = 'all'` または `log_min_duration_statement = 0`。
- AI エージェントワークロードは、トレースが有効になった Airman モデルコンテキストプロトコル MCP サーバーを介して実行されるため、各クエリーはタグ付けされ、セッションに関連付けられます。
- HM インスタンスのマシンユーザー API キーがあること。HM 管理ポータルで作成します。

5.3.3.8.3 HM に接続する

EDB エージェントガバナンスで、タイプ **HM** のインスタンスを追加し、HM ベース URL とマシンユーザー API キーを提供します。次に、アプリケーションはプロジェクトとクラスターを検出し、セッションデータの同期を開始します。完全なプロシージャについては、

Connecting data sources を参照してください。

5.3.3.8.4 詳細はこちら

- EDB Agent Governance — 完全な製品ドキュメント。
- Connecting data sources — HM インスタンスの登録。
- Securing access and handling credentials — 認証と資格情報の処理。
- モニタリングの実行 — アプリケーションが読み取る HM の Loki ロギングパイプライン。

5.3.3.9 Hybrid Manager CLI - edbctl

edbctl は、 EDB Postgres AI Hybrid ManagerHM の専用コマンドラインインターフェイス CLI です。これは、HM リソースを管理し、ターミナル環境を介して高度なデータベースクラスター操作を実行するための主要な管理ツールとして機能します。

edbctl は、構成とリソース管理の両方の統合エントリポイントを提供することにより、管理者はさまざまな環境にわたってデータベース資産のライフサイクルを自動化および合理化できます。

edbctl は、独自の独立したバージョン管理ライフサイクルを持つスタンドアロンパッケージとして配布されることに注意することが重要です。したがって、そのバージョン番号はホストマネージャー HM のバージョン番号をミラーしません。

インストール、互換性、使用法などの詳細については、

edbctl documentation を参照してください。

5.3.3.10 Interacting programmatically with HM

システム管理者と開発者は、 EDB Postgres AI Hybrid ManagerHM API を使用して、管理タスクのために HM と直接統合できます。これらのタスクには、クラスターの管理プロビジョニング、プロビジョニング解除、およびスケーリング、管理操作の自動化、および移行の処理が含まれます。

詳細については、 EDB Postgres AI ポータルの API ドキュメントを参照してください。

1. Hybrid Manager コンソールの URL に移動し、資格情報を使用してログインします。
2. ページの下部で `API` リンクを探します。
3. `API` リンクを選択して、インタラクティブ API エクスプローラーを開きます。

使用可能なすべてのエンドポイントを表示し、必須パラメーターを確認し、ブラウザーから直接サンプル要求を実行することもできます。

5.3.3.10.1 アクセスキー

アクセスキーは、HM API への安全なプログラムによるアクセスを提供します。これにより、ユーザーは毎回対話型ログインを必要とせずに、HM サービスを認証および対話できます。

主な目的

- 認証 — API を呼び出すユーザーまたはシステムの ID を検証します。

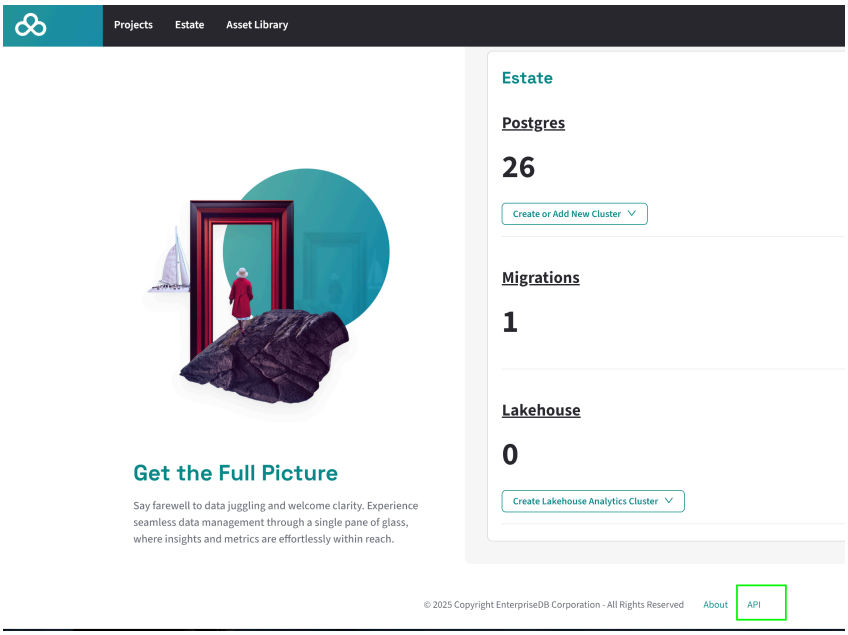


図 11: API Docs

- 自動化 —スクリプト、ツール、またはアプリケーションは、手動介入なしでクラスターのプロビジョニング、スケーリング、モニタリングなどのタスクを安全に実行できます。
- アクセス制御 —アクションを特定のユーザーまたはマシンに関連付け、監査証跡と権限の適用を有効にします。
- セキュリティ —自動化スクリプトにユーザー名とパスワードを埋め込む必要性を回避します。

詳細については、 *Create and manage Access key* を参照してください。

Access key

アクセスキーは、ユーザーがハイブリッドマネージャー HM API と直接対話するための認証を提供します。各アクセスキーは単一のユーザーにリンクされ、作成後に変更できません。

マシンユーザーまたは通常のユーザーのアクセスキーを作成できます。組織の所有者は、マシンユーザーのアクセスキーを管理できます。通常のユーザーは自分のアクセスキーを管理できます。

各キーには、1〜365 日の有効期限が作成時に設定される必要があります。アクセスキーの有効期限が切れたら、または紛失した場合、削除して新しいものを作成する必要があります。

User type	Quota - max keys	Keys created and managed
Machine user	2	Two access keys, created while *Creating a machine user* . Optionally, can be individually added from the **Access keys** tab on the machine user's details page using the **Create New Key** option.
Normal user	1	One access key, ユーザー作成スキーマのテーブル and managed by the user from their home page.

次のページに続く

表 62 – 前のページからの続き

User type	Quota - Keys created and managed max keys

アクセスキーはユーザーから取り消すことができます。キーを取り消しても、リンクされたユーザーのロールまたはその他の認証プロセスには影響しません。

個人用アクセスキーを作成します

1. HM コンソールにログインします。
2. 右上の組織名の横にあるメニューから、 **My Account** を選択します。
3. **My Account** ページから、 **Access Keys** タブを選択します。
4. [**Access Keys**] タブで、[**Create New Key**] を選択します。
5. **Access Key Name** の値を入力します。
6. **有効期限** の値を 1〜365 の範囲で入力します。
7. **Create Key** を選択します。

生成されたアクセスキーはユーザーに提供されます。

このアクセスキーをコピーし、安全な場所に保存します。アクセスキーは、作成した場合にのみ使用できます。アクセスキーを紛失した場合は、削除して新しいものを作成する必要があります。

アプリ内の受信ボックスまたはメール通知を有効にして、個人キーの有効期限が間もなくエキスパイアと通知されるようにできます。詳細については、*Managing notifications* を参照してください。

アクセスキーの管理

Users ページでマシンユーザーのアクセスキーを管理できます。

アクセスキーの詳細を表示するには

1. 右上の組織名の横にあるメニューから、 **User Management** を選択します。
2. アクセスキーの詳細を表示するユーザーの横にある **Edit** を選択します。
3. ユーザーのページにある **Access Keys** タブを表示します。各行は、アクセスキーの詳細を提供します。
    - **Name** — アクセスキーの名前。
    - **Access Key** — 半分暗号化された形式のアクセスキー。
    - **Expiry at** — アクセスキーの有効期限。
    - **作成日** — アクセスキーの作成日。

キーを削除するには、キーの横にある **Delete** を選択します。削除の確認を求められます。

キーを作成するには、 **Access Keys** タブの右上にある **Create New Key** を選択します。

アクセスキーを使用する

HM API は、呼び出し元が要求でアクセスキーを使用しているかどうかを検出します。呼び出し元がアクセスキーを使用する場合、API は以下をチェックします。

- アクセスキーの有効性。
- アクセスを要求しているユーザーの組織。
- ユーザーの権限。

次に、アクセスを承認します。詳細については、*Using the access key* を参照してください。

Using Access Keys to access GenAI Builder

GenAI Builder API と対話するには、アクセスキーを使用して要求を認証する必要があります。このキーは、ベアラートークンまたは `x-access-key` として要求ヘッダーに提供できます。

1. アクセスキーを生成する

UI または CLI を介したハイブリッドマネージャーの *Create an access key*

重要 キーは生成後に表示されないため、安全に保存してください。

2. API 要求でアクセスキーを使用する

GenAI Builder API エンドポイントのベース URL は、通常、`${BASE_URL}/launchpad/apps/genai-builder/api` として構造化されています。ここで `${BASE_URL}` は、Hybrid Manager インスタンスのドメインまたは IP アドレスを表します。

`curl` のようなツールを使用して、API アクセスをテストできます。アクセスキーは、2 つの方法のいずれかで HTTP ヘッダーに含めて送信する必要があります。

- **ベアラートークンとして** `Authorization: Bearer ${access_key}`
- **x-アクセスキーとして** `x-access-key: ${access_key}`

次に、 `x-access-key` ヘッダーを使用してユーザーのリストを取得する例を示します。

```
curl -H "x-access-key: ${access_key}" \
     -k "${BASE_URL}/launchpad/apps/genai-builder/api/users"
```

そこで、

- `${access_key}` は、実際のアクセスキーです。
- `${BASE_URL}` は、Hybrid Manager インスタンスのドメインまたは IP アドレスです。 **注** `-k` オプションを使用すると、`curl` は自己署名 TLS 証明書を操作できます。信頼できる証明書を使用している場合は、削除します。

API 応答の例 ユーザーの取得

上記の `curl` の例に示すように、ユーザーエンドポイントへの要求が成功すると、次のような JSON 応答を受け取る場合があります。

```
{
    "users": [
        {
            "user_id": "d9e03312-638d-52b6-8d6f-f5b99f5f2480",
            "created_at": "2025-06-25T19:27:32.642823Z",
            "updated_at": "2025-06-26T08:50:42.122521Z",
            "organizations": [
                "060e360b-0364-47b6-adc2-f62ac0feae41"
            ],
            "name": "owner@mycompany.com",
            "email": "owner@mycompany.com"
        }
    ]
}
```


Using the HM API

HM API を使用して、クラスターのプロビジョニング、プロビジョニング解除、スケーリングなどの管理アクティビティを行うために HM と直接統合します。

API リファレンスドキュメントは、 EDB Postgres AI ポータルから入手できます。

前提条件

API を呼び出す前に、アクセスキーを作成します。詳細については、*Access key* を参照してください。

API を呼び出す

HM API で認証するには、アプリケーションは、取得したアクセスキーを各 HTTP 要求の x-access-key ヘッダーに含める必要があります。例

```
curl --request GET \
  --url "https://portal.edb-hcp.com/api/v1/projects" \
  --header "x-access-key: $ACCESS_KEY"
```

ACCESS_KEY 変数は、コンソールからコピーしたアクセスキーのテキストです。

応答例


```
{
  "data": [
    {
      "projectId": "string",
      "projectName": "string",
      "userCount": 0,
      "clusterCount": 0,
      "tags": []
```


(次のページに続く)

(前のページからの続き)

```
    }
  ]
}
```

Using MCP tools

Hybrid Manager は、AI エージェントと MCP 互換クライアントがプログラムでプラットフォームと対話できるモデルコンテキストプロトコル MCP サーバーを公開します。MCP サーバーは、エージェントワークフロー自動移行、クラスター管理、オブザーバビリティクエリーなどを大規模な言語モデル LLM または AI パイプラインによって駆動できるようにします。

MCP サーバーは自己文書です。MCP クライアントがエントリーポイントに接続すると、個別の API リファレンスドキュメントを必要とせずに、使用可能なすべてのツールとその説明を自動的に検出できます。

MCP サーバーへの接続

MCP サーバーは、ポータル API ゲートウェイを介して利用できます。次のエントリーポイント URL を使用し、<portal-url> を HM 環境のポータル URL に置き換えます。

```
https://<portal-url>/api/mcp
```

MCP サーバーへの要求には、次のヘッダーが含まれる必要があります。

Header	Value
Authorization	*Bearer <access-key>*
Accept	*application/json, text/event-stream*

ポータルアクセスキーを使用します。十分な権限が必要です。アクセスキーの作成方法については、*Access key* を参照してください。

使用可能なツールカテゴリ

MCP サーバーは、次の機能領域にわたってツールを公開します。

- クラスター管理 —単一ノード、高可用性、高度な高可用性、および分散高可用性トポロジでの Postgres クラスター PG、PGE、および EPAS を作成、表示、スケーリング、更新、および削除し、クラスターテンプレート、バックアップ、およびメンテナンスウィンドウを管理します。
- 可観測性とモニタリング —ヘルススコア、現在および履歴メトリック、アクティブなアラート、ログ、および Grafana ダッシュボードを取得します。
- 推奨事項 —クラスターのインデックス、統計、構成調整、およびセキュリティの推奨事項を取得します。
- 移行 — 評価、スキーマ変換、 Oracle SQL 変換、および移行ライフサイクル作成、一時停止、再開、完了を含むデータベースの移行をエンドツーエンドで管理します。

- レイクハウスと分析 —レイクハウスワークロードのカタログ、階層テーブル、およびデータオフロードを管理します。
- Agent Factory —推論サービス、モデル、モデルサーバークラスター、ナレッジベース、パイプライン、および Langflow ベースの AI フローを構成します。
- ID とアクセス —アクセスキー、マシンユーザー、ID プロバイダー IdP、および組織とプロジェクトのロールを管理します。
- プロジェクトと組織 —プロジェクト、タグ、場所を作成および管理し、使用状況およびアクティビティログを表示します。
- データベースのイントロスペクション —MCP インターフェイスを介して直接スキーマ、テーブル、パーティション、DDL を参照し、クエリパフォーマンスを分析します。

Pg Airman MCP を使用します。

5.3.3.11 Managing identity providers

Hybrid ManagerHM は、HM コンソールでユーザーを認証するための LDAP、SAML、OIDC などの外部 ID プロバイダー IdP をサポートしています。

- IdP 統合を設定および管理するには、 *Configuring your IdP in the HM console* を参照してください。
- IdP コネクタのシークレットを保護する暗号化キーをローテーションするには、*Rotating IdP connector encryption keys* を参照してください。

5.3.3.11.1 Configuring your IdP in the HM console

Hybrid ManagerHM コンソールの ID プロバイダー IdP 構成ワークフローは、統合プロセスを自動化します。このワークフローを使用する場合、Helm チャートを手動で変更したり、ワークロードを再起動したりする必要はありません。ワークフローにより、展開の一貫性が高まり、ダウンタイムが削減されます。

> **重要**
>
> HM で IdP を構成するには、HM 組織の所有者である必要があります。

SAML IdP の構成

Okta、Microsoft Entra ID、Google Workspace、Keycloak など、SAML 2.0 をサポートする IdP を使用できます。

> **重要**
>
> HM で SAML アイデンティティプロバイダーの追加を開始すると、システムは一意の**アサーションコンシューマーサービス URL**および**オーディエンス URI**値を生成します。これらの値は、新しい構成を開始するたびに変更されます。ページから移動せずに、1 つのセッションで IdP 構成と HM 構成の両方を完了する必要があります。

始める前に

1. 別のブラウザタブで IdP 管理コンソールを開きます。
2. HM で、SAML IdP 構成を開始して、**アサーションコンシューマーサービス URL** と **オーディエンス URI** を取得します。
3. これらの値を IdP 構成にすぐにコピーします。
4. IdP セットアップを完了し、シングルサインオン SSO URL と証明書を取得します。
5. [HM] タブに戻り、開いたままにして、構成を完了します。

HM での SAML 構成の開始

1. HM コンソールで、 **設定 > Identity Providers** を選択します。
2. **ID プロバイダーの追加** を選択します。
3. **SAML** タブを選択します。
4. 次の値をコピーします IdP にこれらが必要です。

- **アサーションコンシューマサービス URL**
- **オーディエンス URISP エンティティ ID**

> **注釈**
>
> HM 構成が完了するまで、このページを閉じたり別の場所に移動したりしないでください。

ID プロバイダーの構成

別のブラウザタブで、次の設定を使用して IdP に新しい SAML 2.0 アプリケーションを作成します。

1. SAML 統合の名前を入力します。
2. HM からの **アサーションコンシューマーサービス URL** を**シングルサインオン URL** フィールドに入力します。
3. HM からの **オーディエンス URI** を **Audience URI SP Entity ID)** フィールドに入力します。
4. 要件に基づいて **NameID Format** を設定します。詳細は、*NameID 形式* を参照してください。
5. SAML アサーションにユーザー情報を含めるように属性マッピングを構成します。

- **Username** はユーザーの一意の識別子です。
- **Email** はユーザーのメールアドレスです。

> **注釈**
>
> 属性名は IdP によって異なります。 SAML 属性ステートメントの構成については、IdP のドキュメントを参照してください。 !!!

1. 統合を保存します。
2. IdP が提供した **シングルサインオン URL** SSO URL またはログイン URL とも呼ばれますをコピーします。
3. IdP が提供した **署名証明書** を.pem または.cer ファイルとしてダウンロードします。

HM での SAML 構成の完了

HM ブラウザタブに戻り、次を入力します。

1. **Name** フィールドに、この ID プロバイダーの一意の名前を入力します。
2. **Description** フィールドに、この ID プロバイダーのオプションの説明を入力します。
3. **Single Sign-On URL** に、IdP からの SSO URL を入力します。
4. **Signature Certificate File** の下で、IdP から署名証明書をアップロードします。
5. **NameID Format** で、IdP 構成と一致する形式を選択します。
6. **ユーザー名属性** と **電子メール属性** の値を入力します。

- **Username Attribute** は、ユーザー名を含む属性名たとえば name .
- **Email Attribute** はメールアドレスたとえば email を含む属性名です。

1. オプショナル *Automatic role assignment with role mapping rules* を有効にするには、グループ属性マッピングを構成します。ここで、

- **Groups Attribute** は、ユーザーのグループメンバーシップを含む SAML 属性名ですたとえば、groups または memberOf
- **Groups Delimiter** は属性値内の複数のグループを区切る区切り文字。IdP がグループを別の属性値として送信する場合、空のままにします。

> **注釈**
>
> SAML アサーションにグループメンバーシップを含めるように IdP を構成します。正確な構成は IdP によって異なります。グループ属性ステートメントの構成については、IdP のドキュメントを参照してください。 !!!

1. **Save** を選択します。

NameID 形式

IdP 構成と一致する NameID 形式を選択します。

Format	Description	Use case
Email Address	ユーザーのメールアドレス	Most common. Use when your IdP identifies users by email.

次のページに続く

表 64 – 前のページからの続き

Format	Description	Use case
Persistent	ユーザーの永続的で不透明な識別子	When the IdP assigns stable user identifiers that aren't email addresses.
Transient	現在のセッションでのみ有効な一時識別子	When you require privacy and can't share a persistent identifier.
X.509 Subject Name	X.509 証明書のサブジェクト名	When using certificate-based authentication.

Tip

どの形式を使用するかわからない場合は、最も一般的に使用される形式である**メールアドレス**から始めます。

LDAP IdP の構成

OpenLDAP、Microsoft Active Directory、またはその他の LDAP 互換ディレクトリなどの LDAP ディレクトリサーバーに対してユーザーを認証するように HM を構成できます。

要件

HM で LDAP を構成する前に、LDAP 管理者から次の情報を収集します。

- LDAP サーバーのホスト名とポート
- 読み取りアクセスを持つサービスアカウントのバインド DN とパスワード
- ユーザー検索のベース DN
- ユーザーの属性名 ユーザー名、メールアドレス、名前

HM での LDAP IdP の構成

1. HM コンソールで、 **設定 > Identity Providers** を選択します。
2. **ID プロバイダーの追加** を選択します。
3. **LDAP** タブを選択します。
4. **Name** フィールドに、この ID プロバイダーの一意の名前を入力します。
5. **Description** フィールドに、オプションの説明を入力します。
6. LDAP サーバー接続設定を入力します。

- **Host** は、LDAP サーバーのホスト名です。例えば `ldap.example.com` .
- **Port** は LDAP サーバーポートです。 LDAP の場合は `389` 、LDAPS の場合は `636` を使用します。

- **Scheme** は接続プロトコルです。暗号化されない場合は `ldap` 、TLS 暗号化された接続の場合は `ldaps` を選択します。
- **Use Custom Root CA** は、 **Scheme** が `ldaps` の場合にのみ表示されるトグルです。HM が LDAP サーバーに接続するために内部または自己署名 CA を信頼する必要がある場合に有効にします。
- **カスタムルート CA ファイル**は、LDAP サーバーの TLS 証明書を検証するために使用される CA 証明書を含むテキストファイルです。単一の CA 証明書または証明書のバンドルを含めることができます。複数の`-----BEGIN CERTIFICATE-----` / `-----END CERTIFICATE-----` ブロックが連結されたもの。これは、LDAP サーバーが中間 CA を含むチェーンを提示する場合に役立ちます。サーバー証明書に署名した CA または CA チェーンのみをアップロードします。サーバー証明書またはプライベートキーはアップロードしないでください。

1. バインド資格情報を入力します。 ここで

- **Bind DN** は、ディレクトリの検索に使用されるサービスアカウントの識別名です。例えば `cn=admin,dc=example,dc=com` .
- **Bind Password** は、バインド DN サービスアカウントのパスワードです。

1. ユーザー検索設定を入力します。 ここで

- **Base DN** は、ユーザー検索のベース識別名です。例えば `ou=People,dc=example,dc=com` .
- **Filter** は、ユーザーエントリを検索するための LDAP フィルタです。例えば `(objectClass=inetOrgPerson)` .
- **Username** は、ユーザー名マッチングに使用される属性です。例えば `uid` または `cn` .
- **ID 属性** は、HM ユーザー ID にマッピングする属性です。例えば `uid` または `cn` .
- **電子メール属性**は、HM ユーザーの電子メールにマッピングする属性です。例えば `mail` .
- **Name 属性** は、HM ユーザー表示名にマップする属性です。例 `cn` または `displayName`

1. オプショナル *Automatic role assignment with role mapping rules* を有効にするには、グループ検索設定を構成します。 ここで

- **Base DN** は、グループ検索のベース識別名です。例えば `ou=Groups,dc=example,dc=com` .
- **Filter** は、グループエントリを検索するための LDAP フィルタです。例えば `(objectClass=groupOfNames)` .
- **Name 属性** は、グループ名にマップする属性です。例えば `cn` .
- **User Matchers** は、ユーザーをグループにマッチさせる方法を定義します。各マッチャーは次を指定します。- **User Attribute** は、一致するユーザー属性です。例えば `DN` または `uid` . - **グループ属性**は、ユーザー参照を含むグループ属性です。例 `member` または `memberUid`

`groupOfNames` : - を使用した OpenLDAP の構成例 ベース DN `ou=Groups,dc=example,dc=com` - フィルター `(objectClass=groupOfNames)` - 名前属性 `cn` - ユーザーマッチャーユーザー属性 `DN` 、グループ属性 `member`

Active Directory の構成例 - ベース DN `CN=Users,DC=example,DC=com` - フィルター `(objectClass=group)` - 名前属性 `cn` - ユーザーマッチャー ユーザー属性 `DN` 、グループ属性 `member`

1. **Save** を選択します。

OIDC IdP の構成

Okta、Microsoft Entra ID、Google Workspace、Keycloak など、OpenID ConnectOIDC をサポートする ID プロバイダーを使用できます。

HM での OIDC 構成の開始

1. HM コンソールで、 **設定 > Identity Providers** を選択します。
2. **ID プロバイダーの追加** を選択します。
3. **OIDC** タブを選択します。
4. HM がページに表示する **リダイレクト URI** をコピーします。これは IdP に必要です。

ID プロバイダーの構成

別のブラウザタブで、OIDC プロバイダーに新しいアプリケーションを登録します。

1. 新しい OIDC または OAuth 2.0 アプリケーションを作成します。
2. アプリケーションの名前を入力します。
3. **リダイレクト URI** コールバック URL とも呼ばれますを、HM からコピーした値に設定します。
4. ID トークンまたは UserInfo エンドポイントに `name` および `email` クレームを含めるようにアプリケーションを構成します。
5. アプリケーションを保存します。
6. IdP から次の値を記録します。

- **発行者 URL** たとえば、 `https://accounts.google.com`)
- **クライアント ID**
- **クライアントシークレット**

1. HM にアクセスするユーザーを登録したアプリケーションに割り当てます。

HM での OIDC 構成の完了

HM ブラウザタブに戻り、次を入力します。

1. **Name** フィールドに、この ID プロバイダーの一意の名前を入力します。
2. **Description** フィールドに、オプションの説明を入力します。
3. OIDC プロバイダーからの値を入力します。 ここで

- **発行者 URL** は、OIDC プロバイダーの発行者 URL です。
- **クライアント ID** は、IdP が登録したアプリケーションに割り当てるクライアント ID です。
- **クライアントシークレット**は、IdP が登録済みアプリケーションに割り当てるクライアントシークレットです。

1. オプショナル *Automatic role assignment with role mapping rules* を有効にするには、グループ要求設定を構成します。ここで - **Groups Key** は、ID トークンまたは UserInfo 応答にユーザーのグループメンバーシップを含む要求名です。例 `groups`

> **注釈**
>
> ID トークンまたは UserInfo 応答にグループメンバーシップ要求を含めるように OIDC プロバイダーを構成します。正確な構成はプロバイダーによって異なります。グループクレームの構成については、プロバイダーのドキュメントを参照してください。 !!!

1. **Save** を選択します。

IdP 統合のテスト

ID プロバイダーを構成した後、ビルトインテスト機能を使用して、統合が正しく機能しているかどうかを確認できます。このテストは、構成された IdP を介してサインインを試行し、IdP が `email` や `name` などのユーザー属性を正しく返すことを検証します。

1. HM コンソールで、 **設定 > Identity Providers** を選択します。
2. テストする IdP を選択します。
3. **Test Connection** を選択します。
4. IdP からのサインインプロンプトに従います。認証が完了すると、HM コンソールに、統合が正しく機能するかどうかを示すテスト結果が表示されます。

IdP の管理

HM では、編集または削除して IdP を管理し、許可されたユーザーのみにアクセスを制限するようにユーザーを割り当てます。

IdP を介してユーザーにアクセスを付与する

HM は、ユーザーが最初にログインしたときに IdP に自動的に関連付けます。ユーザーにアクセスを許可するには

1. IdP の管理コンソールで、HM へのアクセスを付与するユーザーを割り当てます。
2. **クイックサインイン URL** をユーザーと共有します。

- HM コンソールで、 **設定 > Identity Providers** を選択します。
- IdP を選択し、 **クイックサインインをコピーします URL** 。

1. ユーザーがこの URL を使用してログインすると、HM は自動的に登録し、IdP に関連付けます。

HM からの IdP の削除

HM から IdP を削除する前に、その IdP を介してログインしたすべてのユーザーを削除する必要があります。HM は、関連付けられたすべてのユーザーを削除するまで、IdP の削除オプションを無効にします。

1. HM コンソールで、 **User Management** を選択します。

2. 削除する IdP に関連付けられているすべてのユーザーを特定して削除します。
3. **設定** > **ID プロバイダー** を選択します。
4. 削除する IdP を選択し、 **Delete** を選択します。
5. 削除を確認します。

5.3.3.11.2 Rotating IdP connector encryption keys

Hybrid ManagerHM は、`hm-portal-bootstrap` Kubernetes シークレットの `fernet-key` フィールドに保存されている Fernet キーを使用して、アイデンティティプロバイダー IdP コネクタ構成の機密フィールドクライアントシークレットやバインドパスワードなどの値を暗号化します。セキュリティ体制の一部として、このキーを定期的にローテーションするか、キーが公開された疑いがある場合はすぐにローテーションします。

`fernet-key` はキーのコンマ区切りリストを受け入れるため、ダウンタイムなしでローテーションできます。HM は常にリストの最初のキーで暗号化しますが、リスト内の任意のキーで保護されたデータを復号化できます。これは、新しいキーをフロントに追加し、HM にそれを使用させ、既存のコネクタを再暗号化し、その後にのみ古いキーを削除できることを意味します。サービス全体を実行したままにします。

始める前に

- `default` 名前空間のシークレットの読み取りおよび編集の権限と、HM を実行しているクラスターへの `kubectl` アクセスが必要です。
- 新しいキーの生成に使用するマシンに `openssl` をインストールする必要があります。
- 現在の Fernet キーのバックアップを実行し、安全な場所に保存します。例を提供しますが、好みの方法を使用できます。
- ローテーションには、各コネクタを再保存し、HM の ID サービスを再起動する必要があるため、メンテナンスウィンドウに実行します。

> **重要**
>
> Fernet で暗号化された値は、それを生成したキーでのみ読み取ることができます。データがまだ保護されている限りリストにキーを保持します。すべてのコネクタを再暗号化する前にキーを削除すると、これらのコネクタのシークレットが完全に回復不可になります。

キーのローテーション

1. 現在のキーを読み取り、バックアップします。この値をシークレットマネージャーにコピーします。これは、バックアップであり、次の手順で参照される「現在のキー」の両方です。

```
kubectl get secret hm-portal-bootstrap -n default \
  -o jsonpath={.data.fernet-key} | base64 -d
```

1. 新しいキーを生成します。Fernet キーは、URL セーフ、base64 エンコードされた 32 バイト値です。出力をコピーします。

```
openssl rand -base64 32 | tr +/ -_
```

1. 現在のキーはそのままで、新しいキーを追加します。編集のためにシークレットを開きます。

```
kubectl edit secret hm-portal-bootstrap -n default
```

エディターが開くと、`fernet-key` が base64 エンコード値として `data:` の下にリストされています。手動でエンコードせずにプレーンテキストのキーを提供するには、`data:` ブロックから `fernet-key` 行を削除し、代わりにトップレベルの `stringData:` ブロックを追加し、新しいキー **最初に**、現在のキーを 2 番目にリストします。

```
apiVersion: v1
kind: Secret
metadata:
  name: hm-portal-bootstrap
  # ⋯
data:
  static-passwords.yaml: <unchanged>
stringData:
  fernet-key: "<new-key>,<current-key>"
```

保存してエディターを閉じます。HM は、新しいキーリストの最初に使用して暗号化し、現在のキーで既存のデータを読み取ります。

> **注釈**
>
> *stringData* は`fernet-key` のみをオーバーライドするため、シークレットの残りの部分 *static-passwords.yaml* を含む は変更されません。 API サーバーは`stringData` を`data` に移動してエンコードするため、次回シークレットを編集するときに、*stringData* ではなく `data:` の下に `fernet-key` が再び表示されます更新された値を保持しています。 !!!

1. 変更が `upm-dex` 名前空間に複製されると、HM の ID サービスは自動的に再起動します。ポッドが再起動し、実行されていることを確認します。

```
kubectl get pods -n upm-dex -l app.kubernetes.io/name=dex
```

認証は中断せずに動作し続けます。

1. 新しいキーを使用して既存のコネクタのシークレットを再暗号化します。HM コンソールで、設定済みの各 IdP コネクタを開き、再度保存します。*Configuring your IdP in the HM console* を参照してください。コネクタを保存すると、現在の新しいキーで機密フィールドが再暗号化されます。

> **注釈**
>
> コネクタを再保存するまで、そのシークレットは古いキーで保護されたままです。次の手順に進む前に、すべてのコネクタを再保存します。 !!!

1. 古いキーを削除します。すべてのコネクタが再保存されたら、シークレットを再度編集します。

```
kubectl edit secret hm-portal-bootstrap -n default
```

`data:` ブロックから `fernet-key` 行を削除し、新しいキーのみを含む `stringData:` ブロックを追加します。

```
stringData:
  fernet-key: "<new-key>"
```

保存してエディターを閉じます。

> **警告**
>
> すべてのコネクターを再保存するまで、古いキーを削除しないでください。コネクタのシークレットが古いキーで保護されているときに削除すると、HM はそのシークレットを読み取ることができず、影響を受ける IdP ログインは機能を停止します。 !!!

ローテーションの検証

HM の ID サービスが再起動したら、ポッドが実行されていることを確認します。

```
kubectl get pods -n upm-dex -l app.kubernetes.io/name=dex
```

次に、構成済みの各 IdP を介してサインインして、認証が成功したことを確認します。

古いキーを削除した後にログインが失敗した場合、そのコネクタは時間内に再暗号化されませんでした。復旧するには、シークレットを再度編集し、バックアップした古いキーを使用して `stringData.fernet-key` を`"<new-key>,<old-key>"` に新しいキーを最初に設定します。HM の ID サービスは自動的に再起動します。サインインしてログインが機能することを確認し、影響を受けるコネクタを HM コンソールに再保存して、古いキーを再度削除します。

5.3.4 For Kubernetes Administrators

5.3.4.1 Managing users in Hybrid Manager

Hybrid ManagerHM は、ネイティブユーザーまたは LDAP、SAML、OIDC などの外部 ID プロバイダー IdP を使用して HM コンソールの認証をサポートします。

HM の初期ブートストラップ中に、管理者は HM コンソールアクセス用のネイティブスーパーユーザーを設定します。このネイティブスーパーユーザーを使用して、コンソールに初めてサインインします。

テストまたはデモ環境の場合、必要に応じてネイティブユーザーを追加することにより、操作を簡素化できます。

- 詳細については、 *Creating native users for testing* を参照してください。

システムロールをさまざまなユーザーに割り当てるために、HM はロールベースのアクセス制御 RBAC モデルを使用します。

- 詳細については、 *User roles and authorization* を参照してください。

運用展開の場合、 LDAP、SAML、OIDC、または別の supported connector のような外部 IdP を構成することを強くお勧めします。これにより、堅牢で安全なユーザー管理が保証されます。インストール時に IdP を構成します。選択した IdP を構成し、組織の新しい管理者を追加した後、ネイティブユーザーログインを無効にします。

- 詳細については、 *Configuring your IdP in the HM console* を参照してください。

組織全体にセキュリティとガバナンス要件を適用するために、HM では、組織の所有者がアクセスキーの有効期間、Web セッションの期間、マシンユーザーとアクセスキーの最大数などの認証ポリシー設定を構成できます。

- 詳細については、 *Managing authentication policy* を参照してください。

5.3.4.1.1 Managing native users for testing

テストまたはデモ環境の場合、ネイティブユーザーを使用して操作を簡素化できます。ネイティブユーザーのライフサイクルを管理するには、次のことができます。

- *Changing password for native users* 。
- *Creating native users for testing* 。

運用展開の場合、 *Configuring your IdP in the HM console* を強くお勧めします。IdP を構成すると、次のことができます。

- *Disabling native user login* 。

Changing password for native users

セキュリティ違反が発生した場合、ネイティブユーザーのパスワードを変更できます。

HM は、ネイティブユーザー資格情報を hm-portal-bootstrap Kubernetes シークレットに保存します。パスワードを変更するには、このシークレットを新しい bcrypt ハッシュ化したパスワードで更新します。

ネイティブユーザーパスワードの変更

1. 新しい bcrypt ハッシュ化されたパスワードを生成します。$password には、新しいパスワードが含まれます。

```
NEW_HASH=$(echo -n "${password}" | htpasswd -BinC 10 admin | cut -d: -f2)
echo "Password hash: $NEW_HASH"
```

出力には、新しいハッシュされたパスワードが表示されます。例

```
Password hash: $2y$10$YDHMEnc8VZKnhvze538ePOLKol7dFH.S7IgpqC3.X3R/3P2Q9zRcG
```

1. 現在のシークレットコンテンツを取得します。

```
kubectl get secret hm-portal-bootstrap -n default -o jsonpath={.data.static-passwords\
↪.yaml} | base64 -d > static-passwords.yaml
```

1. static-passwords.yaml ファイルを編集し、更新するユーザーの hash 値を置き換えます。

```
staticPasswords:
  - email: "owner@mycompany.com"
    hash: "$2y$10$YDHMEnc8VZKnhvze538ePOLKol7dFH.S7IgpqC3.X3R/3P2Q9zRcG"
    username: "Owner MyCompany"
    userID: "0fa134d4-b3ee-4503-ad90-91044dc37061"
```

重要

hash 値のみを変更します。 *userID* は変更しないでください。 *userID* を変更すると、新しいユーザー ID が作成され、同じメールアドレスと名前を持つ重複ユーザーが作成され、元のユーザーに関連付けられている既存のデータを孤立させます。 !!!

1. 変更したコンテンツでシークレットを更新します。

```
kubectl patch secret hm-portal-bootstrap -n default \
  --type merge \
  -p "{\"data\":{\"static-passwords.yaml\":\"$(base64 < static-passwords.yaml | tr -d␣
↪\n)\"}}"
```

Kubernetes はシークレットを upm-dex 名前空間に自動的にレプリケートし、Dex HM を ID プロバイダーに接続する IdP プロキシサービスが自動的に再起動して変更を適用します。

1. Dex ポッドが再起動し、実行されていることを確認します。

```
kubectl get pods -n upm-dex -l app.kubernetes.io/name=dex
```

新しいパスワードでサインインできます。

Creating native users for testing

テストまたはデモ環境の場合、ネイティブユーザーを追加することにより操作を簡素化できます。運用展開の場合、*Configuring your IdP in the HM console* を強くお勧めします。

HM は、ネイティブユーザー資格情報を hm-portal-bootstrap Kubernetes シークレットに保存します。新しいユーザーを追加するには、このシークレットに資格情報を追加します。

ネイティブユーザーの追加

1. 新しいユーザーの一意のユーザー ID と bcrypt ハッシュされたパスワードを生成します。${password} にはパスワードが含まれます。

```
NEW_USER_ID=$(uuidgen | tr [:upper:] [:lower:])
NEW_HASH=$(echo -n "${password}" | htpasswd -BinC 10 admin | cut -d: -f2)
echo "User ID: $NEW_USER_ID"
echo "Password hash: $NEW_HASH"
```

1. 現在のシークレットコンテンツを取得します。

```
kubectl get secret hm-portal-bootstrap -n default -o jsonpath={.data.static-passwords\
↪.yaml} | base64 -d > static-passwords.yaml
```

1. static-passwords.yaml ファイルを編集し、新しいユーザーを staticPasswords リストに追加します。

```
staticPasswords:
  - email: "owner@mycompany.com"
    hash: "$2y$10$FWXeALAmT/axI/FIos/NtOVEldynfCx/bqDOAJ9EbiKeq5NuDR.XO"
    username: "Owner MyCompany"
    userID: "c234d951-dee1-4e3d-9f1d-108dabe339ce"
  - email: "newuser@example.com"
    hash: "$2y$10$FIvL7GWymJtGZo5uTMrWsOTuifUmwJQg8JtHJQxuLkT3os2Ozpo1C"
    username: "New User"
    userID: "60138e10-9552-4922-afe7-044c2250a23d"
```

重要

ユーザーの `userID` は、割り当てられた後に変更しないでください。これを変更すると、新しいユーザー ID が作成され、同じメールアドレスと名前を持つ重複ユーザーが作成され、元のユーザーに関連付けられている既存のデータを孤立させます。 !!!

1. 変更したコンテンツでシークレットを更新します。

```
kubectl patch secret hm-portal-bootstrap -n default \
  --type merge \
  -p "{\"data\":{\"static-passwords.yaml\":\"$(base64 < static-passwords.yaml | tr -d␣
↪\n)\"}}"
```

1. Kubernetes はシークレットを upm-dex 名前空間に自動的にレプリケートし、Dex 展開が自動的に再起動して変更を適用します。
2. Dex ポッドが再起動し、実行されていることを確認します。

```
kubectl get pods -n upm-dex -l app.kubernetes.io/name=dex
```

新しいユーザーは、自分のメールアドレスとパスワードを使用して HM コンソールにサインインできます。

パラメーター

これらの設定は、portal.authentication.staticPasswords エントリーのユーザー用です。各ユーザーには、email 、hash 、username 、および userID の値が必要です。

Parameter	Description
email	ユーザーのメールアドレス。 HM コンソールにアクセスするためのユーザーのログイン識別子としても機能します。
hash	パスワードストアの Bcrypt ハッシュ化されたユーザーパスワード。この値を生成するには、 `echo ${password} I htpasswd -BinC 10 userA I cut -d: -f2`を使用します。ここで、実際のパスワードを `${password}`変数に保存します。 `userA`は、パスワードのハッシュプロセス中に使用されるユーザー名を表します。 HM は構成内の他の場所で使用しないため、任意のテキストにすることができます。 HM は、結果のハッシュのみを使用します。
userID	静的パスワードの各ユーザーには、一意の `userID`が必要です。 UUID ツールで生成するか、ランダムな文字列を手動で割り当てることができますが、一度設定したら決して変更しないでください。 `userID`を変更すると、新しい ID が作成され、ユーザーが複製され、既存のデータが孤立します。
username	ユーザーの一意のユーザー名。これは、HM コンソールにログインするためのプライマリ識別子です。 `email`と同じでもかまいません。

Disabling native user login

ネイティブユーザーログインを無効にする前に、少なくとも 1 人の他の IdP ユーザーが組織所有者ロールがあることを確認してください。新しい組織の所有者をまだ割り当てていない場合は、割り当てます。

新しい組織の所有者の割り当て

1. 組織の所有者になる IdP ユーザーとして Hybrid ManagerHM コンソールに少なくとも 1 回ログインします。これにより、HM コンソールは新しいユーザーのインデックスを作成します。次に、ログアウトします。
2. HM の現在のネイティブ所有者たとえば `owner@mycompany.com` としてログインします。
3. 右上のプロファイルメニューから、 **User Management** を選択します。新しい所有者になるユーザーがリストにあることを確認します。
4. ユーザーを編集し、所有者ロールを割り当てます。

ネイティブユーザーの無効化

1. 現在のシークレットコンテンツを取得します。

```
kubectl get secret hm-portal-bootstrap -n default -o jsonpath={.data.static-passwords\
↪.yaml} | base64 -d > static-passwords.yaml
```

1. `static-passwords.yaml` ファイルを編集し、 `staticPasswords` を空のリストに設定します。

```
staticPasswords: []
```

1. 変更したコンテンツでシークレットを更新します。

```
kubectl patch secret hm-portal-bootstrap -n default \
  --type merge \
  -p "{\"data\":{\"static-passwords.yaml\":\"$(base64 < static-passwords.yaml | tr -d␣
↪\n)\"}}"
```

1. Kubernetes はシークレットを upm-dex 名前空間に自動的にレプリケートし、Dex は自動的に再起動して変更を適用します。

ネイティブユーザーログインを無効にしましたが、ユーザーアカウントは引き続き**ユーザー管理**ビューに表示されます。構成済みの IdP を介して認証したユーザーのみが HM コンソールにアクセスできます。

5.3.4.1.2 Managing authentication policy

組織の所有者は、認証ポリシーを構成して、組織のセキュリティ設定を制御できます。これらの設定では、ユーザーとマシンユーザーがアカウントにアクセスする方法を管理できます。たとえば、アクセスキーの有効期間を制限すると、キーの漏洩が削減され、マシンユーザーの数に制限を設けることで最小特権アクセスがサポートされます。

認証ポリシーの構成

> **重要**
>
> **組織所有者**ロールを持つユーザーのみが認証ポリシー設定を表示および更新できます。

認証ポリシー設定を更新するには

1. Hybrid ManagerHM コンソールで、 **Settings** に移動します。
2. **Authentication** タブを選択します。
3. 構成する値を入力します。各設定の詳細については、 パラメーター値を参照してください。
4. **Save** を選択して変更を適用します。

> **注釈**
>
> ポリシーの変更は、既存のセッション、マシンユーザー、またはアクセスキーに影響を与えません。変更は、新しいセッション、新しく作成されたマシンユーザーとアクセスキーにすぐに有効になります。アクセスキーまたはマシンユーザーのクォータを減らしても、既にプロビジョニングされたマシンユーザーとアクセスキーには影響しません。

パラメーター値

これらの設定は、組織の **設定>認証** から利用できます。

Setting	Description	Default	Valid range
Maximum Machine Users	組織で許可されるマシンユーザーの最大数。	4096	1 to 2147483647
Maximum Access Keys per Machine User	マシンユーザーごとに作成できるアクセスキーの最大数。	2	1 to 2147483647
Maximum Access Keys per User	通常の人間のユーザーアカウントごとに作成できるアクセスキーの最大数。	1	1 to 2147483647
Maximum Access Key Duration	アクセスキーの最大有効期間日単位。キーはこの期間の後に有効期限が切れます。	365	1 to 2147483647
Maximum Web Session Duration	Web セッションの最大時間秒単位。ユーザーはこの期間が経過するとログアウトされます。	86400	1 to 2147483647

5.3.4.1.3 User roles and authorization

複数のユーザーロールを Hybrid ManagerHM ユーザーに割り当てることができます。参照

- *User role permissions by use case*
- *User role descriptions*
- *Automatic role assignment with role mapping rules*

User role descriptions

Hybrid ManagerHM には、人間ユーザーネイティブユーザーまたは IDP ユーザーとマシンユーザーの 2 種類のユーザーがいます。

マシンユーザーのみがエステートインジェスターになれることを除き、両方のタイプのユーザーにデフォルトのロールを割り当てることができます。

ロールの 2 つの主要なカテゴリは、組織レベルとプロジェクトレベルです。

組織レベルのロール

HM コンソールのナビゲーションバーの右上にあるユーザプロファイルメニューの**ユーザー管理**オプションを使用して、これらのロールを人間またはマシンユーザーに割り当てることができます。

組織管理者

このロールは、組織レベルでの読み取りアクセスを提供し、ユーザーは組織全体の情報と設定を表示できます。

組織の所有者

このロールの目的は、高いレベルで組織を管理することです。組織の所有者はプロジェクトを作成し、組織レベルのロールを他のユーザーに割り当てることができます。

プラットフォーム管理者

このロールは、HM の基になるプラットフォームコンポーネントにアクセスして管理する必要があるユーザーを対象としています。プラットフォームの管理および監視ツールにアクセスできます。

AI モデルマネージャー

このロールは、HM の AI モデルとモデルサービス機能を使用するユーザーを対象とており、その機能へのフルアクセスが付与されます。また、Langflow フロービルダー UI へのアクセスも許可します。

プロジェクトレベルのロール

これらのロールは、人間またはマシンユーザー Estate Ingester を除くに割り当てることができます。プロジェクトを表示するときに、左側のナビゲーションで **Users** を選択します。

プロジェクト所有者

特定のプロジェクトでは、プロジェクト所有者ロールが最高レベルの権限を持っています。彼らはそのプロジェクト内のすべてのアクションを実行でき、プロジェクトレベルのロールを他のユーザーに割り当てます。

プロジェクトエディター

このロールは、プロジェクト内のデータをアクティブに作業する必要があるユーザーを対象としています。Project Editor ロールを持つユーザーは、データの読み取りおよび書き込みアクセスがあります。

プロジェクトビューア

プロジェクト内のデータを観察することだけが必要なユーザーに、Project Viewer ロールは読み取り専用アクセスを提供します。

エステートインジェスター

このロールはマシンユーザー用であり、プロジェクトでエステートの取り込みを実行するためのアクセスを提供します。

パイプラインエディター

このロールは、プロジェクトで AI パイプラインとナレッジベースを作成および管理する必要があるユーザーが対象です。

カタログデータリーダー

このロールを持つユーザーは、プロジェクトスコープのカタログの Iceberg 名前空間/テーブル/ビューを読み取ることができます。

カタログデータライター

このロールにより、ユーザーは、プロジェクト内のすべてのプロジェクトスコープのカタログの Iceberg 名前空間/テーブル/ビューの読み取り、書き込み、および削除ができます。

Migration Portal プロジェクトの所有者

このロールは、HM プロジェクトで移行ポータルプロジェクトを管理するユーザーを対象としています。これらのプロジェクトを作成、読み取り、更新、および削除できます。

Migration Portal プロジェクトエディター

このロールを持つユーザーは、HM プロジェクトの移行ポータルプロジェクトを読み取りおよび更新できます。

Migration Portal プロジェクトビューア

このロールは、HM プロジェクトの移行ポータルプロジェクトへの読み取り専用アクセスを提供します。

Automatic role assignment with role mapping rules

ロールマッピングルールを使用すると、アイデンティティプロバイダー IdP 属性に基づいて、ユーザーにロールを自動的に割り当てることができます。ユーザーがログインすると、ハイブリッドマネージャー HM は構成されたルールを評価し、手動介入なしで適切なロールを割り当てます。

この機能は、次のことを行う組織に役立ちます。

- IdP グループメンバーシップに基づいてロールを自動的に付与します。
- 手動ロール割り当てのオーバーヘッドを削減します。
- 同じ IdP グループのユーザー間で一貫したロールの割り当てを保証します。

ロールマッピングの仕組み

ユーザーが外部 IdP を介して認証すると、HM はその IdP ID とグループメンバーシップに関する情報を受信します。ロールマッピングルールは、これらの属性に基づいて HM がどのロールを割り当てるかを定義します。

各ルールは次で構成されます。

- **条件**適用するルールと一致する必要がある基準たとえば、IdP 識別子とグループメンバーシップなど。
- **ロール** すべての条件が一致した場合に割り当てるロール。

ルールの評価

- 単一のルール内のすべての条件は AND ロジックと一致する必要があります。
- 複数のルールは独立して評価されます OR ロジック。
- ユーザーは、すべての一致ルールからロールのユニオンを受け取ります。

ロールの割り当てソース

ユーザーは 2 つのソースからロールを持つことができます。

- **手動**管理者が HM コンソールを介して直接割り当てるロール。管理者はこれらのロールを変更または削除できます。
- **マッピング** HM がロールマッピングルールを介して自動的に割り当てるロール。HM コンソールでこれらのロールを変更することはできません。ユーザーのマップされたロールを変更するには、IdP でのロールマッピングルールまたはユーザーのグループメンバーシップを更新します。

前提条件

ロールマッピングルールを構成する前に

1. グループ属性マッピングを有効にして外部 IdP SAML、OIDC、または LDAP を構成します。
2. ユーザーが IdP で構成されたグループメンバーシップを持っていることを確認します。
3. 組織の所有者または適切な管理者権限を持っています。

> **重要**
>
> グループメンバーシップ情報を取得するように HM IdP 統合を構成する場合、各ユーザーは IdP 内の少なくとも 1 つのグループのメンバーである必要があります。グループメンバーシップを持たないユーザーは、HM にサインインできません。

ロールマッピングルールの構成

ロールマッピングルールは 2 つのレベルで構成できます。

- **組織レベル**組織スコープのロール組織管理者、組織所有者などを割り当てます。
- **プロジェクトレベル**プロジェクトスコープのロールを割り当てますプロジェクト所有者、プロジェクト編集者、プロジェクトビューアなど

組織レベルのルールの作成

> **重要**
>
> 組織所有者のみが組織レベルのロールマッピングルールを作成または編集できます。

1. HM コンソールで、 **User Management** に移動します。

2. **Role Mapping** タブを選択します。
3. **Add Role Mapping Rule** を選択します。
4. ルール条件を構成します。

Condition	Description
Identity Provider	このルールが適用される IdP を選択します。
Group	ユーザーが所属する必要がある IdP グループ名を入力します。

1. 条件が一致した場合に割り当てる組織ロールを選択します。
2. **Save** を選択します。

プロジェクトレベルのルールの作成

> **重要**
>
> 組織所有者またはプロジェクト所有者のみがプロジェクトレベルのロールマッピングルールを作成または編集できます。

プロジェクトレベルのロールマッピングルールは、既存のプロジェクトに、またはプロジェクトの作成中に追加できます。

既存のプロジェクトへのルールの追加

1. HM コンソールで、プロジェクトに移動します。
2. **Users** に移動します。
3. **Role Mapping** タブを選択します。
4. **Add Role Mapping Rule** を選択します。
5. ルール条件 ID プロバイダーとグループを構成します。
6. 条件が一致した場合に割り当てるプロジェクトロールを選択します。
7. **Save** を選択します。

プロジェクトの作成中にルールを追加する

1. 新しいプロジェクトを作成するときに、 **自動ロールの割り当て**を有効にします。
2. プロジェクトセットアップの一部としてロールマッピングルールを構成します。

> **注釈**
>
> ロールマッピングルールを介して HM が割り当てたロールは、HM コンソールで変更できません。ユーザーのマップされたロールを変更するには、IdP でのロールマッピングルールまたはユーザーのグルー

プメンバーシップを更新します。

構成例

IT 管理者への組織所有者ロールの割り当て

組織レベルのルールを作成します。

- **ID プロバイダー** 企業の SAML IdP
- **グループ** `IT-Admins`
- **ロール** 組織の所有者

HM は、`IT-Admins` グループ内のすべてのユーザーがログイン時に、組織所有者ロールを自動的に割り当てます。

チームメンバーシップに基づいてプロジェクトのロールの割り当て

データ分析プロジェクトのプロジェクトレベルのルールを作成します。

ルール 1 - データエンジニアは編集者アクセスを取得します。

- **ID プロバイダー** 企業 LDAP
- **グループ** `data-engineering`
- **ロール** プロジェクト編集者

ルール 2 - データアナリストはビューアアクセスを取得します。

- **ID プロバイダー** 企業 LDAP
- **グループ** `data-analysts`
- **ロール** プロジェクト閲覧者

`data-engineering` および `data-analysts` グループの両方のメンバーであるユーザーは、Project Editor と Project Viewer の両方のロールを受け取ります。

ロールの調整

ロールマッピングルールを変更すると、HM は影響を受けるユーザーのロールの割り当てを自動的に調整します。

- **ルールの追加**新しいルールに一致するユーザーは、次回のログイン時に設定されたロールを受け取ります。
- **ルールの削除** HM は、削除されたルールからマッピングに割り当てられたロールを取り消します。手動で割り当てられたロールは変更されません。
- **ルールの変更** HM は、以前の割り当てを削除し、更新されたルールに基づいて新しい割り当てを適用します。

HM は、ルールが変更されると、以前にログインしたユーザーのロールをすぐに調整します。ログインしたことがないユーザーは、最初のログイン時に適切なロールの割り当てを受け取ります。

ユーザーセッションの無効化

IdP のグループメンバーシップを更新する場合たとえば、新しいグループにユーザーを追加すると、現在ログインしているユーザーは、再度ログインするまで更新されたグループ情報を自動的に受信しません。

ユーザーが現在のグループメンバーシップに基づいて正しいロールを受け取れるように、セッションを無効にして再認証を強制できます。

1. HM コンソールで、 **設定 > Identity Providers** に移動します。
2. ユーザーを再認証する必要がある IdP を選択します。
3. **Invalidate Sessions** を選択します。
4. アクションを確認します。

HM は、その IdP からユーザーのアクティブなセッションをすべて終了します。ユーザーは再度ログインする必要があります。この時点で、認証トークンには IdP からの現在のグループメンバーシップが含まれます。

> **注釈**
>
> セッションの無効化は、グループが変更されたユーザーだけでなく、選択した IdP のすべてのユーザーに影響を与えます。 IdP で重要なグループメンバーシップを変更した後、この機能を使用します。

ロールマッピングルールの削除

ロールマッピングルールを削除する場合

1. HM は、影響を受けるユーザーからマッピングを介してのみ割り当てられたロールを自動的に取り消します。
2. 手動で割り当てられたロールはそのまま残ります。
3. 変更は、以前にログインしたことがあるユーザーにすぐに有効になります。

> **注釈**
>
> ロールマッピングルールを削除しても、ユーザーのログイン機能には影響しません。そのルールから自動的に割り当てられたロールのみを削除します。

アクティビティログ

ロールマッピングルールの変更は、アクティビティログに記録されます。ルールを追加または削除すると、HM は、スコープが組織またはプロジェクトであるかどうか、および構成されているルールの数を示す更新アクションをログに記録します。

設定 > アクティビティログに移動して、組織アクティビティログのロールマッピングアクティビティを表示できます。プロジェクト固有のロールマッピングの変更は、プロジェクトに移動し**アクティビティログ**を選択することにより、プロジェクトのアクティビティログで表示することもできます。

User role permissions by use case

これらのユーザーロールの承認は、ロールベースのアクセス制御 RBAC モデルに従い、特定のスコープ 1 つのプロジェクト内または 1 つのアカウント内のいずれかに制限が適用されます。

次のリストは、Postgres クラスターデータベースの承認をカバーしていません。

現在、カスタムロールを作成することはできません。これらの 14 の事前定義されたロールのみが使用できます。

マシンユーザーのみが自己管理クラスターデータの取り込みに割り当てることができます。

次の機能は、割り当てられたロールに関係なく、認証されたすべてのユーザーが利用できます。

- チャットボットへのアクセス。ユーザーがチャットボットを介して対話するには、関連するリソース Postgres クラスター、AI モデル、パイプラインなどへのアクセスを付与するロールが引き続き必要です。
- Postgres イメージリポジトリとリポジトリルールを管理します。

5.3.4.2 Monitoring with Hybrid Manager

Hybrid ManagerHM を使用すると、さまざまなメトリックホストメトリック、Kubernetes メトリック、Postgres データベースデータ、クエリデータ、ロギングデータを収集できます。収集されたデータを HM コンソールまたは Grafana インスタンスで視覚化して、傾向の分析、アラートの構成、データベースの効率の向上、トラブルシューティングの実行などを行うことができます。

5.3.4.2.1 監視アーキテクチャの理解

HM には、モニタリング機能を提供する 3 つの主要な内部パイプラインがあります。

- Prometheus と Thanos はメトリックを収集します。
- Loki はログを収集します。
- Stats Collector はクエリデータを収集します。

これらのサービスはすべて、独自の専用ストアが付属しています。これは、収集されたデータが HM に保持されることを意味します。これらのサービスとストアはデフォルトで展開されます。

これらの収集されたメトリックは、2 つのインターフェイスを介して利用できるようになります。

- **Grafana** ダッシュボード。
- **HM console** たとえば、 **Estate** タブまたは **Cluster** view 。

データベースモニタリングの有効化

HM に付属する監視サービスは、監視するデータベースへのアクセスと、Stats Collector を有効にするための拡張機能が必要です。

- HM 管理クラスター HM で作成の場合、デフォルト設定は監視スタックへのアクセスを提供します。
- 外部自己管理クラスターの場合、データ収集を有効にするには、各データベースを registered with the EDB Postgres AI agent にする必要があります。デフォルトでは、エージェントは Prometheus、Thanos、

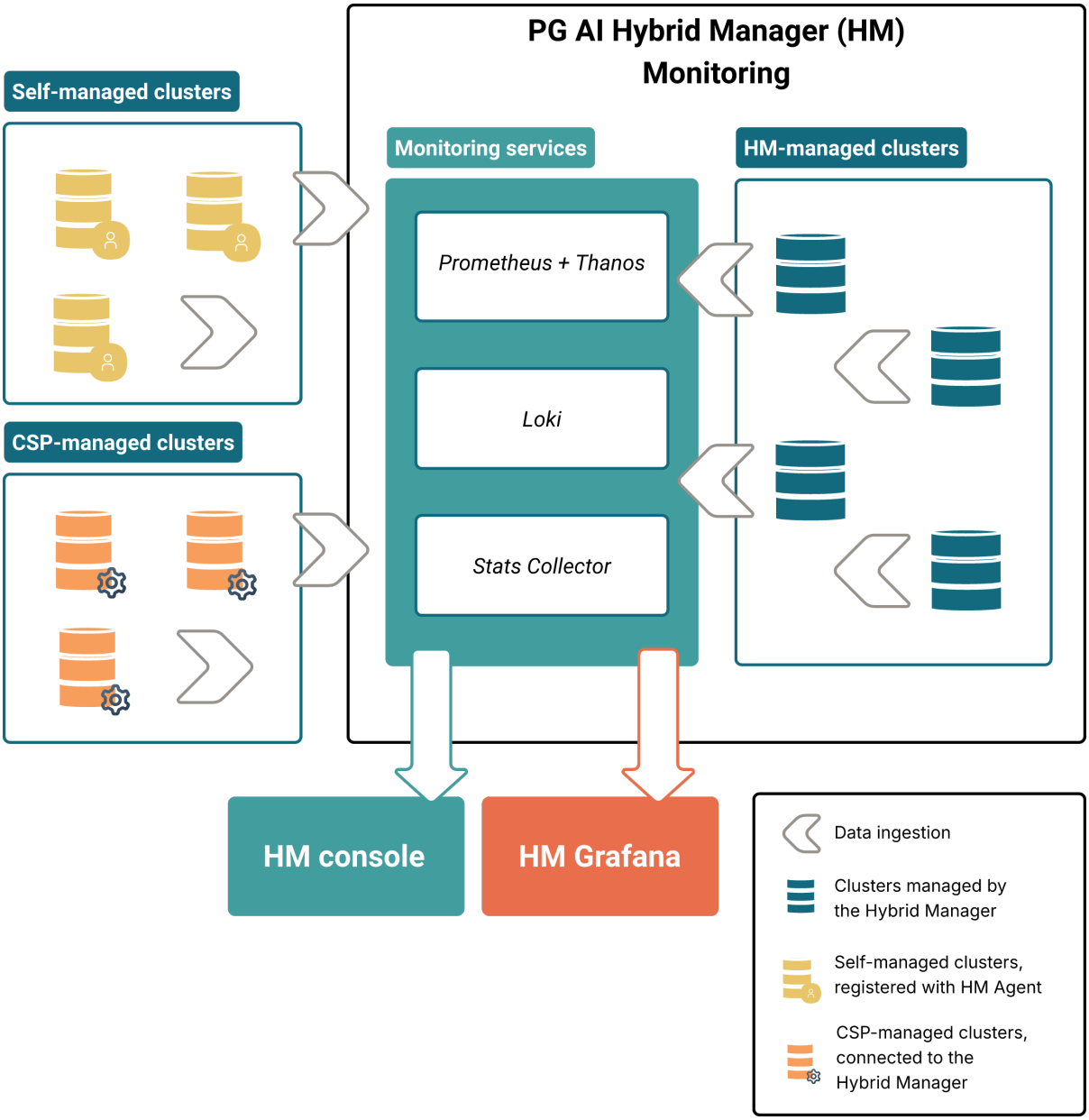


図 12: Monitoring infrastructure in Hybrid Manager

および Loki データを収集するための構成を提供します。さらに、エージェントは構成パラメーターを提供するため、Stats Collector クエリデータの収集に必要な拡張機能、つまり `edb_wait_states_data` 、`edb_stat_monitor` 、`edb_query_advisor` 、および `pg_stat_statements` を有効または無効にするかどうかを制御できます。

- AWS RDS などの外部クラウドサービスプロバイダー CSP 管理データベースの場合、HM は Kubernetes `secret` リソースを介してデータベースにアクセスして、データ収集を有効にする必要があります。詳細については、Monitoring CSP-managed databases を参照してください。 Stats Collector を使用したクエリデータは、CSP 管理データベースでは現在使用できません。

データの取り込みとスコープの監視

さまざまなクラスタータイプに対して HM のアクセスレベルが異なるため、データが取り込まれる方法、および収集されるデータの範囲は異なります。

- HM 管理クラスター HM で作成の場合、データベースはホストメトリック、Postgres メトリック、クラスターメトリック、クエリメトリック、レコメンデーションに必要なデータ、およびログを、対応する Prometheus、Thanos、Loki、および Stats Collector ストアにプッシュします。
- 外部自己管理データベースの場合、エージェントはホストメトリック、Postgres メトリック、クラスターメトリック、クエリメトリック、レコメンデーションに必要なデータ、およびログを構成可能な間隔で対応する Prometheus、Thanos、および Stats Collector ストアにプッシュします。
- AWS RDS などの外部 CSP 管理データベースの場合、HM は HM からデータベースへの接続を活用して、ホストおよび Postgres のメトリックを中央ストアにプルします。Stats Collector を使用したクエリデータは、CSP 管理データベースでは現在使用できません。
- 自動コンプライアンスの場合 HM サーバーは、コアベースのライセンス付与とすべてのオンプレミスサーバーにわたる資格追跡のために使用レポートを EDB に送信し、手動監査を排除します。

詳細については、 *Metrics support scope* を参照してください。

監視リフレッシュ頻度

HM によって収集されるメトリックの更新間隔は、メトリックとデータベースタイプによって異なります。この違いは、データ収集のさまざまな方法と、Prometheus / Thanos、Loki、および Stats Collector サービスの統合によるものです。

メトリックが更新される頻度と頻度をカスタマイズする方法については、*Metrics refresh intervals* を参照してください。

5.3.4.2.2 データを外部ツールに接続する

現在、EDB は、HM によって収集されたメトリックとログデータの外部ツールまたはエンタープライズシステムへのプラグインをサポートしていません。

5.3.4.2.3 関連トピック

- *Enabling monitoring* ☒エージェントをインストールするか、データベースを HM に接続して、外部データベースでのモニタリングを有効にします。
- *Enabling HM server monitoring* ☒ HM サーバーでの使用レポートを有効にします。

- *Metrics refresh intervals* ☒メトリックが更新される頻度と、頻度をカスタマイズする方法を理解します。
- Monitoring with Grafana ☒ Grafana を使用してデータベースをモニタします。
- Monitoring with HM console ☒ HM コンソールを使用してデータベースをモニタします。
- 通知を設定する オブザーバビリティ ☒ 通知を構成します。
- アクティビティログ ☒組織またはプロジェクト全体でプラットフォームイベントを監査します。

Enabling monitoring

HM 管理のデータベースクラスター

Hybrid Manager (HM)-managed database clusters は HM と同じ環境で作成されているため、それらのリソースは既にアクセス可能です。したがって、これらのクラスターはデフォルトで監視され、展開が完了するとすぐに監視が開始されます。

外部データベースクラスターの場合

外部データベースクラスターの場合 は、クラウドサービスプロバイダー CSP またはお客様によって管理されます。このため、データベースクラスターと HM インスタンス間の接続を作成するには、追加の手順が必要です。

モニタできる外部データベースには 2 種類があります。

- **自己管理展開のモニタリング**

enable monitoring on self-managed deployments には、エージェントをインストールし、外部自己管理データベースを登録する必要があります。

例自己管理の Postgres データベースまたはベアメタルにある EDB フェールオーバーマネージャーのようなクラスターマネージャー、または CSP によって管理されていないクラウドでホストされているもの EC2 インスタンスの Postgres データベースなど。

- **CSP 管理データベースの監視**

enable monitoring on CSP-managed databases 、データベースを HM に接続します。

例 Amazon RDS。

> **注釈**
>
> データベースをエージェントに *assess them for migration* に登録することもできます。

Monitoring self-managed deployments

ベアメタルまたは仮想マシンに展開されている自己管理の外部 Postgres インスタンスをハイブリッドマネージャー HM からモニタするには、最初に EDB Postgres AI エージェントをインストールする必要があります。エージェントは、モニタリングと可観測性メトリックを HM に提供します。

自己管理の Postgres インスタンスのモニタリング、クエリ診断、および推奨事項を行うには

1. *Create an OS user to run the agent.*
2. *Prepare your database.*
3. *Create a machine user.*
4. Install the agent.
5. *Configure the agent.*
6. *Run the agent.*
7. *Configure the agent to run as a service.*

注釈

可観測性機能を完全に活用するには、自己管理データベースクラスターを使用してホストマシンごとに 1 つのエージェントを展開することをお勧めします。次に、専用の DSN 文字列を使用して各データベースに接続するようにそのホスト上のエージェントを構成できます。 DSN 構成の詳細は、 *Configuring your agent* で提供されています。

エージェントの構成に必要な管理タスクの詳細については、*Administrative tasks* を参照してください。

一般的なトラブルシューティングのヘルプについては *Troubleshooting Langflow in Hybrid Manager* を参照し、エージェントによってデータが保存される方法を理解するには *Data security and sovereignty* を参照してください。

Create an OS user to run the agent

エージェントを構成および実行するための専用のオペレーティングシステムユーザーを作成することをお勧めします。これは、機能する必要のないエージェントに権限を付与することを回避するため、セキュリティのベストプラクティスです。

ユーザーを作成するには

```
sudo useradd <agent_os_user>
```

エージェント OS ユーザーを Postgres ユーザーのグループに追加することをお勧めします。これにより、後でエージェントに Postgres ログファイルへのアクセスを許可することが簡単になります。

```
sudo usermod -aG postgres <agent_os_user>
```

Preparing your Postgres database for monitoring

監視ユーザー

エージェントを構成するときは、最初に監視する自己管理データベースに接続し、エージェントがデータベースへの接続に使用するユーザーを作成する必要があります。このユーザーは、pg_monitor ロールと pg_read_all_data ロールを持つ Postgres データベースユーザーで、ターゲットデータベースに対する CONNECT 特権があります。

この例では、ユーザー `agent_user` を作成します。

```
CREATE USER agent_user WITH PASSWORD <password>;
GRANT CONNECT ON DATABASE <target_database> TO agent_user;
GRANT pg_monitor, pg_read_all_data TO agent_user;
```

<target_database> は、モニタするデータベースの名前です。

Postgres 16 以前を使用している場合、次の権限を付与して、エージェントが Postgres ログファイルの場所を検出できるようにする必要があります。

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_current_logfile() TO agent_user;
```

エージェントユーザーが、接続する予定のデータベースにログインできることを確認する必要があります。たとえば、hr というデータベースに接続する予定がある場合

```
GRANT CONNECT ON DATABASE hr TO USER agent_user;
```

edb_internals スキーマと必要な拡張機能

Hybrid ManagerHM の完全なモニタリングと最適化機能を活用するには、Postgres にインストルメンテーションを追加する拡張機能を有効にする必要があります。これらの拡張機能は、**Query Diagnostics** と **Recommendations** タブ、および一部のメトリックを有効にするために必要です。

拡張機能をインストールするには

1. プラットフォームおよび Postgres ディストリビューションの正しいコマンドについては、拡張機能ごとのインストール手順を参照してください。

- EDB Wait States
- EDB Query Advisor
- EDB Stat Monitor

さらに、パブリックスキーマで pg_stat_statements を有効にする必要があります。pg_stat_statements は Postgres とともにインストールされるため、追加のインストールは必要ありません。

例、RHEL 9 で EDB Advanced Server 17 をモニタするには

```
dnf install edb-as17-server-edb_wait_states \
            edb-as17-query-advisor \
            edb-as17-stat_monitor
```

1. データベースに接続しているときに、必要なプリロードモジュールを Postgres shared_preload_libraries に追加します。例、shared_preload_libraries の現在値が空の場合

```
ALTER SYSTEM SET
shared_preload_libraries="$libdir/pg_stat_statements","$libdir/edb_wait_states", "
↪$libdir/query_advisor","$libdir/edb_stat_monitor";
```

1. スーパーユーザーとして、エージェントが接続するデータベースにスキーマを作成し、pg_stat_statements 拡張機能と前の手順でインストールした拡張機能を有効にします。

```
CREATE SCHEMA edb_internals;
CREATE EXTENSION edb_wait_states WITH SCHEMA edb_internals;
CREATE EXTENSION query_advisor WITH SCHEMA edb_internals;
CREATE EXTENSION edb_stat_monitor WITH SCHEMA edb_internals;
CREATE EXTENSION pg_stat_statements;
```

1. edb_internals スキーマを検索パスに追加します。例、スーパーユーザーの場合

```
ALTER SYSTEM SET
search_path="$user", public, edb_internals;
```

1. エージェントユーザーにこのスキーマへのアクセスを付与します。

```
GRANT USAGE ON SCHEMA edb_internals TO agent_user;
GRANT SELECT ON ALL TABLES IN SCHEMA edb_internals TO agent_user;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA edb_internals TO agent_user;
```

Postgres ロギング構成

Postgres ログを収集するには、Postgres ロギングコレクターを有効にし、エージェントにログファイルを読み取る権限が必要です。これを行う最も簡単な方法は、Postgres ユーザーとしてエージェントを実行することです。ただし、セキュリティのベストプラクティスとして、 *Create an OS user to run the agent* を使用してエージェントを実行することをお勧めします。

エージェント OS ユーザーがログファイルを読めるようにするには、ログがデータディレクトリの外部にあり、適切な権限を持っている必要があります。たとえば、postgresql.conf で次のパラメーターを設定すると、ログが/var/log/postgres-lc に書き込まれ、所有者のグループのメンバーであるすべてのユーザーが読み取ることができます。

```
logging_collector=on
log_directory=/var/log/postgres-lc
log_file_mode=0640
log_destination = jsonlog
```

新しいロギングの場所は、Postgres ユーザーが所有し、そのユーザーのグループが読み取り可能である必要があります。エージェント OS ユーザーは同じグループのメンバーであるため、ログディレクトリへのアクセスも取得します。例

```
mkdir /var/log/postgres-lc
chown postgres:postgres /var/log/postgres-lc
chmod 0750 /var/log/postgres-lc
```

これらの変更が完了したら、Postgres を再起動します。

Creating a machine user

Hybrid ManagerHM コンソールで、エージェントがエステートにデータを取り込むために使用できるマシンユーザーを作成します。

マシンユーザーの作成

1. 右上隅にあるプロフィールアイコンを選択します。
2. **User Management > Add New User** を選択します。
3. **Add New User** ページで、新しいユーザーの名前を入力します。
4. オプションで、ユーザーの電子メールアドレスを入力します。
5. **Create Access Key** を選択します。フォームが展開され、**アクセスキー名** と **有効期限** 時間を入力できます。
6. **Add User** を選択します。
7. アクセスキーは安全に保存します。キーに再度アクセスすることはできません。**キーは安全に保存されました** を選択します。

ロールの割り当て

1. **Projects** タブを選択して、すべてのプロジェクトを表示します。
2. データベースをモニタするプロジェクトを選択します。
3. 左側のナビゲーションで **Users** オプションを選択し、作成したばかりのマシンユーザーを見つけます。
4. ユーザーエントリの編集ボタンペンアイコンを選択します。
5. **役割の割り当て** を選択します。
6. **Assign Project Roles** ウィンドウで、 **estate ingester** を選択します。
7. **Submit** を選択します。

新しいマシンユーザーは、エージェントからデータを取り込む準備が整います。

Configuring the EDB Postgres AI agent

EDB Postgres AI エージェント `beacon-agent` として展開を構成して、HM に接続し、ソースデータベースから監視データを取得できるようにします。

> **注釈**
>
> エージェントを使用して *assess your Oracle database for migration* を行うこともできます。

前提条件

- Install the agent 。
- あなたは *Creating a machine user* に割り当てられました。アクセスキーを$BEACON_AGENT_ACCESS_KEY という名前の環境変数に保存して、 EDB Postgres AI エージェントが後でアクセスできるようにします。
- 移行ワークフローの一部としてスキーマの取り込みを使用する予定がある場合、エージェントを実行しているマシンに pg_dump をインストールする必要があります。インストール手順と互換性ルールについては、*Configuring your auxiliary machine or bastion host* を参照してください。

EDB Postgres AI 接続パラメーターを取得する

EDB Postgres AI エージェントには、HM インストール中に設定される特定の構成値が必要です。あなた自身はこれらにアクセスできません。HM インスタンスの管理者またはインストーラーに連絡して、次の値を要求します。

- **ビーコンサーバーホスト名** HM インストールを構成するときに、管理者/インストーラーは beacon_server サービスの内部エンドポイントを指定する必要があります。HM インスタンスの管理者またはインストーラーに、インストールに使用される Helm チャート構成ファイルで beacon_server サービスに設定したホスト名 URL を提供するように要求します。このためには、values.yaml ファイルの parameters.upm-beacon.server_host または BEACON_SERVICE_DOMAIN_NAME に設定した値を検索する必要があります。この値は、後で EDB Postgres AI エージェントを構成するために必要になります。
- **ルート証明書パス必要な場合** EDB Postgres AI エージェントには、HM インスタンスの beacon_server サービスへの信頼できる接続が必要です。エージェントを実行しているサーバーが HM の TLS 証明書組織のセキュリティインフラストラクチャによって管理されているを本質的に信頼していない場合は、HM インスタンスの管理者またはインストーラーに証明書を含むファイルを提供するように要求します。この証明書は、EDB Postgres AI エージェントを実行しているマシンに保存します。後で EDB Postgres AI エージェントを構成するには、この証明書へのディレクトリパスを提供する必要があります。

beacon_server およびルート証明書の詳細については、*Administrative tasks* を参照してください。

ソースデータベース接続パラメーターを定義する

DSN は、 EDB Postgres AI エージェントがソースデータベースに接続するために使用する接続文字列です。環境変数$DSN 推奨またはダイレクト URL 形式を使用して、 clusters.nodes.dsn パラメーターを設定します。

データベースへの接続の DSN 文字列は、次の形式を使用する必要があります。

```
DSN="<postgresql/oracle>://<user>:<password>@<host>:<port>/<database_name>"
```

そこで

- <postgresql/oracle> は、登録するデータベースのタイプであり、すべてのタイプの Postgres データベース EDB Postgres、AWS RDS PostgreSQL などの場合は postgresql 、Oracle データベースの場合は oracle のいずれかです。

- <username> および<password> は、データベースの準備時に権限を付与したユーザーに対応します。
- <host> は、データベースが実行されているインスタンスのホスト名です。
- <port> は、データを取得するデータベースのポート番号です。
- <database_name> は、データを取得するデータベースの名前です。

例

ほとんどの運用環境では、SSL が有効になっていて、ネットワークトラフィックが暗号化されます。

パスワード password で認証されたユーザー postgres として、SSL を有効にしてポート 5432 のデータベース postgres に接続するには、次を指定します。

```
DSN="postgresql://postgres:password@localhost:5432/postgres"
```

Postgres データベースが SSL を無効にして構成されている場合、または暗号化されない接続が必要な場合、DSN 文字列に?sslmode=disable を追加します。

たとえば、パスワード password で認証されたユーザー postgres として、SSL を無効にしてポート 5432 のデータベース postgres に接続するには、次のように指定します。

```
DSN="postgresql://postgres:password@localhost:5432/postgres?sslmode=disable"
```

```
DSN="postgresql://postgres:pa%24%24word@localhost:5432/postgres?sslmode=disable"
```

構成ファイルの準備

1. *OS user you previously created* に変更して、エージェント構成を実行します。
2. ホームディレクトリに構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

この場所が便利でない場合は、次のいずれかを使用することもできます。

- /etc/beacon - beacon-agent コマンドを実行するディレクトリ。

エージェントは、/etc/beacon で始まり、次に${HOME}/.beacon で構成ファイルを検索します。最終的なフォールバックとして、実行されたディレクトリを検索します。

1. このディレクトリ内に、新しいファイル beacon_agent.yaml を作成します。

```
touch .beacon/beacon_agent.yaml
```

1. 次のテンプレートをコピーして、新しいファイルに貼り付けます。このテンプレートは、すべての監視オプションを有効にし、移行評価を生成するように設計されています。

このテンプレートでは、 $BEACON_AGENT_ACCESS_KEY および$DSN は、前に構成した環境変数です。テンプレート内のすべての< >プレースホルダーを *EDB DMS agent parameter reference* に置き換えます。

``yaml # EDB Postgres AI エージェントの構成`

agent.beacon_server	*<beacon_server>:9443*	Host URL that allows the agent to connect to the correct service endpoint. Obtain the host URL of your beacon server from an administrator, the port is fixed as 9443. See *Administrative tasks* for more information.
agent.project_id	*<project_id>*	To obtain your project's ID, go to the HM console, select **Projects** , and select your project. Your browser displays a URL like *https://portal.example.com/projects/prj_5Wuqyl5JtpvxjiYC/clusters*. The project ID is the identifier starting with "prj_" in the URL of that page. In this example: *prj_5Wuqyl5JtpvxjiYC*.
agent.providers	N/A	Set to - *"onprem"*.
agent.settings_providers	N/A	Set to - *"onprem-settings"* to enable recommendations for Postgres configurations.
agent.schema_providers	N/A	Set to - *"onprem-schema"* to enable schema ingestion and migration assessment capabilities. Remove this section if you don't want to allow schema information to be collected.
agent.root_ca_path	N/A	Set to *""* if your organization has configured the machine where EDB Postgres AI agent runs to trust HM's TLS certificate. Otherwise, obtain the root certificate from an administrator and store it locally. Then, set this value to the path where you stored it. See *Administrative tasks* for more information.
agent.plaintext	*false*	Disables TLS when set to *true*. Defaults to *false*. In almost all cases, this should remain *false* or unset. Only set to *true* for local testing or when connecting to an endpoint that does not use TLS.
agent.feature_flag_interv	*10m*	How often the agent syncs feature flags with the server. Defaults to *10m*. In almost all cases, the default is preferred.
general.logging.level	N/A	Set to *"info"* or *"trace"* to obtain logging information. Start with *"info"* and only use *"trace"* if required, as it is extremelly verbose.
general.metrics.push.[...]	N/A	Enables pushing metrics to the HM console. The *push_endpoint* is the same as *agent.beacon_server*. The *root_ca_path* is the same as *agent.root_ca_path*.
general.metrics.usage.[...]	N/A	Enables storing usage data (and reporting usage data, *Troubleshooting*). Your *company_code* can be found on the EDB Support Portal .

次のページに続く

表 68 – 前のページからの続き

provider.onprem.check_c	N/A	Enables verifying all DSN connections and reports any errors to the terminal by indicating the source of the error with the resource ID during runtime. The check is performed when the EDB Postgres AI agent runs without any CLI subcommands/options , or when it runs with the *--config* subcommand. The default is set to false.
dispatcher.location_id	<*location_id*>	A unique identifier used by the EDB Postgres AI for this agent/location. Consider setting this to the same value as *provider.onprem.host.resource_id* for convenience.

これらの設定は、データベース構成に関連します。すべての `clusters.*` パラメーターは、YAML ファイルの `provider.onprem` の下にネストされています。

YAML database settings	Placeholder	Guidelines
clusters.resource_id	<*cluster_resource_id*>	Assign a unique identifier to your database cluster. Encase in double quotes. You must begin identifiers with an alphanumeric character (*A-Z, a-z, 0-9*); in between you may also include hyphens (-), underscores (_) and periods (.). You must not use any other characters. Example: *An_I.D.-1*
clusters.name	*"<instance_name>"*	Assign a unique name to your database cluster.
clusters.manager	*"other"*	Collects statistics for cluster manager. Possible values are *"patroni"*, *"repmgr"*, *"efm"*, *"pgd"*, *"other"*.
clusters.pgd_cluster.bdr_dat	<*bdr_database_name*>	Required when *clusters.manager* is set to *"pgd"*. Specifies the name of the BDR database in the PGD cluster.
clusters.nodes.resource_id	<*node_resource_id*>	Assign a unique identifier to your database node. Single-node instances require a single entry here. Multi-node instances require an entry per node. Encase in double quotes. Reuse the same id when configuring the DMS agent for migrations. This consistency is essential for the DMS to correctly correlate the resource across both the EDB Postgres AI agent and DMS agent components. You must begin identifiers with an alphanumeric character (*A-Z, a-z, 0-9*); in between you may also include hyphens (-), underscores (_) and periods (.). You must not use any other characters. Example: *An_I.D.-1*

次のページに続く

表 69 – 前のページからの続き

YAML database settings	Placeholder	Guidelines
clusters.nodes.dsn	N/A	Connection string or DSN that provides access to your remote database. The DSN must follow the format: *<database_type>://<user>:<password>@<host>:<port>/<database_n* The *database_type* can be postgresql or oracle.
*clusters.nodes.disable_host_*	N/A	Set to *true* to prevent this database being associated with the host on which the agent is running. For example, if the agent is running on a different server.
clusters.nodes.tags	N/A	Assign tags for resource labeling. Encase in double quotes and introduce each tag with a dash.
clusters.nodes.settings.enabl	N/A	Set to *true* to enable collecting database configuration data.
clusters.nodes.metrics.[...]	N/A	Controls *Fine-tuning monitoring data collection* the agent ingests, and how they are ingested.

これらの設定はホストデータに関連しています。すべての `host.*` パラメーターは、YAML ファイルの `provider.onprem` の下にネストされています。

YAML database settings	Placeholder	Guidelines
host.metrics.disabled	N/A	Set to *true* to disable the collection of host metrics from the host on which the agent is located. For example, when the agent host is not the same as the database cluster host.
host.resource_id	*<host_id>*	Assign an identifier to the host on which the agent is running, by default this will be populated by the hostname detected during setup. This will link your host machine to an identifier. Encase in double quotes. Must be unique within the project.

これらの設定は、スキーマの取り込みに関連します。`clusters.nodes.schema.*` パラメーターは、YAML ファイルの `provider.onprem` の下にネストされています。

YAML database schema settings	Placeholder/Value	Guidelines
provider.onprem.schema	N/A	Maximum number of workers the EDB Postgres AI agent starts in parallel to extract schema from the source database server(s). You can control CPU, memory and disk space usage by the EDB Postgres AI agent witht this parameter. See 複数の同時スキーマ取り込みの実行 for more information. The default is set to 10.
clusters.nodes.schema.enabl	N/A	Set to *true* to enable schema ingestion for the database. Set to *false* to disable it.
clusters.nodes.schema.poll_i	N/A	Defines how frequently the agent checks for schema changes. Provide a value in seconds or minutes (for example, 15 s or 1 m).
clusters.nodes.schema.filter	N/A	Use this block to specify which schemas to include or exclude from ingestion. It contains the *mode* and *names* parameters.
clusters.nodes.schema.filter.	*include* or *exclude*	Specifies the filtering behavior. Use *include* to only ingest listed schemas, or *exclude* to ingest all schemas except those listed.
clusters.nodes.schema.filter.	*<schema_name>*	A list of strings representing the names of the schemas to be filtered. This list works in conjunction with the mode setting.

関連トピック

- *Fine-tuning monitoring data collection* ☒モニタリング機能をカスタマイズ、それらのサブセットを有効/無効にする方法を学びます。
- *Configuring multiple databases* ☒同じエージェント構成ファイルを使用して、複数のデータベースに接続し、データを取得する方法を学習します。
- *Monitoring the EFM Cluster* ☒ 同じエージェント構成ファイルを使用して EDB Failover ManagerEFM クラスターをモニタする方法を学習します。
- *Adding usage reporting for Hybrid Manager-monitored databases* ☒製品の使用状況に関する基本的な洞察を EDB に提供することにより、モニタリング取り組みを補完する方法を学びます。
- *Administrative tasks* ☒ HM インスタンスの管理者がルート証明書と `beacon_server` の値を取得する方法を学習します。
- *Use the agent to assess your Oracle database for migration* ☒ EDB Postgres AI エージェントを使用して、Oracle データベースを評価し、移行ポータルで変換するためのスキーマを抽出します。
- *Perform multiple concurrent schema ingestions* ☒ EDB Postgres AI エージェント構成を調整して、EDB Postgres AI エージェントのリソース使用率を調整し、大量のデータベースの取り込みをサポートします。

Monitoring the EFM Cluster

EDB フェールオーバーマネージャーをモニタするには、efm ノードごとに 1 つの EDB Postgres AI エージェントを展開します監視ノードを除きます。 EDB Postgres AI エージェントは、 `efm` cli を活用し、`efm cluster-status-json <cluster_name>` コマンドを実行して、フェールオーバーマネージャーの監視情報を収集します。

マネージャータイプを `efm` として使用するクラスターとノードの構成

- マネージャータイプを `efm` として設定します
- `cluster_name` フィールドを使用して、Failover Manager クラスターの名前を指定します。クラスター名は、クラスタープロパティファイルの名前の接頭辞です。たとえば、クラスタープロパティファイルの名前が efm.properties の場合、クラスター名は efm になります。

```
#  efm node1 beacon agent config

...
clusters:
  - resource_id: selfmanaged_efm_cluster_id
    name: "<pg_cluster_name1>"
    manager: "efm"
    efm:
      cluster_name: "<efm_cluster_name>"
    nodes:
      - resource_id: <efm_node_resource_id_1>
        dsn: "$DSN1.1"
        ...
```

efm ノードごとに 1 つの EDB Postgres AI エージェントを展開する必要があるため、ノードリストには 1 つのエントリのみが含まれます。

```
#  efm node2 beacon agent config

...
clusters:
  - resource_id: selfmanaged_efm_cluster_id
    name: "<pg_cluster_name1>"
    manager: "efm"
    efm:
      cluster_name: "<efm_cluster_name>"
    nodes:
      - resource_id: <efm_node_resource_id_2>
        dsn: "$DSN1.2"
        ...
```

> **注釈**
>
> *clusers ～resource_id* が efm ノードのすべてのエージェント構成で同じであることを確認します。

Fine-tuning monitoring data collection

Configuring the agent で提供されるテンプレートは、すべてのモニタリング機能を有効にします。有効になっている機能を微調整できます。

時系列メトリック [モニタリング] タブ

構成ファイルの `general.metrics.push` セクションは、時系列メトリックの Hybrid ManagerHM の Prometheus インスタンスへのプッシュを制御します。これらのメトリックは、[**Monitoring**] タブに入力されます。これを無効にするには、`general.metrics.push.enabled` を `false` に設定します。

Postgres ログ [Logs] タブ

`general.metrics.push.include_logs` を `true` に設定して、ログ収集を有効にします。この値を `false` に設定することにより、エージェントレベルで無効にします。さらに、Postgres ログを収集する `provider.onprem.clusters.nodes` のエントリごとに、`logs.disabled` ～`false` を設定します。

デフォルトでは、エージェントは Postgres ファンクション `pg_current_logfile()` を呼び出すことにより、ログファイルの場所を検出しようとします。別の場所にログを保存するように Postgres インスタンスを構成した場合など、ログファイルの場所を明示的に指定する場合は、`<logs_directory>` をログファイルを保存するディレクトリへのパスに置き換えます。

たとえば、ファイルパス `/var/log/postgres-lc` から拡張子 `.log` を持つログを収集するには、パターン `/var/log/postgres-lc/*.log` を使用できます。

```
clusters:
  - resource_id: selfmanaged_postgres
    ...
    nodes:
      - resource_id: selfmanaged_postgres_node1
        ...
        logs:
          disabled: false
          mode: file
          file:
            paths: <logs_directory>/*log
```

Diagnostics [クエリーの診断と推奨事項] タブ

`provider.onprem.clusters.nodes` の各エントリの下の `metrics.stats` セクションは、クエリ分析と最適化のためのより詳細な診断データの収集を制御します。

`metrics.stats.enabled` を `false` に設定することにより、このようなデータ収集をすべて無効にできます。または、適切な `enabled` パラメーターを設定することにより、各要素を有効または無効にすることができます。

- `metrics.stats.wait_states` は、EDB 待機状態からセッション状態をサンプリングすることに基づいて、クエリーデータの収集を制御します。
- `metrics.stats.query_text` は、 EDB Stat Monitor からのクエリ統計の収集を制御します。

これらのオプションを両方とも無効にすると、[**Query Diagnostics**] タブが無効になります。

- `metrics.stats.recommendations` は、EDB クエリアドバイザーからのインデックスと統計の推奨事項の収集を制御します。無効にすると、**Recommendations** タブにはインデックスと統計の推奨事項が表示されませんが、他の推奨事項は表示されます。
- `metrics.stats.query_text` は、 EDB 待機状態と EDB Stat Monitor の両方からクエリテキストの収集を制御します。無効にすると、クエリテキストは収集されず、**クエリ診断** にクエリ ID のみが表示されます。

Configuring multiple databases

可観測性機能を完全に活用するには、自己管理データベースクラスターを使用してホストマシンごとに 1 つのエージェントを展開することをお勧めします。次に、同じホスト上の複数のデータベースクラスターに接続するようにこの単一のエージェントを構成できます。

複数のクラスターとノードの構成

エージェントは `clusters` セクションを使用して、ホスト上のすべての監視対象データベースクラスターを定義します。

- **クラスターエントリ**クラスターリストの各項目は、単一のデータベースクラスターたとえば、プライマリデータベースとそのレプリカ、またはシングルノードインスタンスを表します。
- **ノードエントリ**各クラスター内のノードリストは、そのクラスターに属する個々のデータベースインスタンスを指定します。

シングルインスタンスデータベースの場合、ノードリストには 1 つのエントリが含まれます。

マルチノードクラスタープライマリ/レプリカ設定のような場合、ノードリストには、すべてのインスタンスのエントリを含めることができます。

```
clusters:
  - resource_id: selfmanaged_postgres_1
    name: "<instance_name1>"
    manager: "other"
    nodes:
      - resource_id: <node_resource_id_1.1>
        disable_host_association: false
        dsn: "$DSN1.1"
        ...
      - resource_id: <node_resource_id_1.2>
        disable_host_association: false
        dsn: "$DSN1.2"
        ...
  - resource_id: selfmanaged_postgres_2
```

(次のページに続く)

(前のページからの続き)

```
  name: "<instance_name2>"
  manager: "other"
  nodes:
    - resource_id: <node_resource_id_2.1>
      disable_host_association: true
      dsn: "$DSN2.1"
  ...
```

単一のエージェントインスタンスは、異なるホストマシンにまたがるデータベースを追跡できますが、その設定は、エージェントと同じマシン上にないデータベースに限定的なホストレベルのメトリックを提供します。

注釈

現在、監視データは DSN で指定されたデータベースからのみ収集されます。たとえば、 *postgres* データベースを指定した場合、データベースサイズはこのデータベースのサイズのみを反映します。同じ Postgres インスタンス内の他のデータベースをモニタするには、複数のインスタンスを指定するのと同じ方法でそれらを個別に指定する必要があります。

Schema change detection

スキーマ変更検出は、追加または削除テーブル、変更された列、新しいまたは削除されたインデックス、制約の変更など、データベーススキーマの変更を自動的に特定し、ハイブリッドマネージャー HM コンソールにアクティビティイベントとして報告します。

スキーマ変更の検出は 2 つの段階で動作します。

1. **スキーマ収集** EDB Postgres AI エージェントは定期的に監視対象データベースをポーリングし、現在のスキーマ状態をビーコンサーバーに送信します。ポーリング頻度は `schema.poll_interval` デフォルト `1m` によって制御されます。
2. **変更検出**ビーコンサーバーは、受信した各スキーマスナップショットを、以前に保存されたバージョンと比較します。違いが見つかると、サーバーはスキーマ変更イベントをアクティビティサービス `accm-server` に発行し、HM アクティビティログに記録します。

データベースが最初に登録された後、 **10 分の猶予期間**が適用されます。この期間中、新しく検出されたテーブルは変更として報告されず、これにより、最初のスキーマの取り込みが大量の「テーブル追加」イベントを生成するのを防ぎます。

検出された変更タイプ

- テーブル追加または削除
- 列の追加、削除、または変更データ型、NULL 可能性、デフォルト値
- インデックスの追加または削除
- コンストレインの追加または削除

- テーブルタイプが変更されました

展開タイプ別のスキーマ変更検出の有効化

スキーマ変更の検出には、スキーマ収集がアクティブである必要があります。これを有効にする方法は、展開タイプによって異なります。

オンプレミスデータベース

オンプレミス EDB Postgres AI エージェントで監視されるデータベースの場合、`beacon-agent.yaml` でスキーマ収集を **明示的に有効にする**必要があります。各ノードエントリに `schema.enabled` から `true` を設定します。

```
provider:
  onprem:
    clusters:
      - resource_id: "<cluster_resource_id>"
        nodes:
          - resource_id: "<node_resource_id>"
            dsn: $DSN
            schema:
              enabled: true
              poll_interval: 1m
```

スキーマ収集が有効になると、展開タイプに関係なくサーバー側で変更検出が自動的に行われ、追加の構成は必要ありません。`schema.enabled` が設定されていないか、`false` の場合、スキーマデータは収集されず、変更は検出できません。

HM 管理対象クラスターと外部 CNPG クラスター

HM 管理の Postgres クラスターと外部 CloudNativePG クラスターの場合、EDB Postgres AI エージェントはデータベースとともにサイドカーコンテナとして実行されます。スキーマ収集は**デフォルトで有効になっています** 。

RDS などの外部クラウドデータベース

EDB Postgres AI エージェントを介して監視される外部クラウドホストデータベースの場合、オンプレミスデータベースと同じ構成に従います。関連するノードエントリの EDB Postgres AI エージェント構成で `schema.enabled: true` を設定します。

制限事項

- EDB Postgres AI エージェントは、実行中に **テーブルと列**への変更を検出します。ビューまたはプロシージャへの変更は、 EDB Postgres AI エージェントを再起動するまで検出されません。
- スキーマ変更の検出は、イベントを受信できるアクティビティサービス `accm-server` に依存しています。詳細は、 アクティビティサービスの使用不可 を参照してください。

アクティビティサービスの使用不可

スキーマ変更イベントおよびその他のビーコンテレメトリは、アクティビティサービス accm-server を介してプラットフォームに配信されます。アクティビティサービスが低下するか、利用できない場合、イベントは EDB Postgres AI エージェントからの送信に失敗します。これは予想されるプラットフォームの動作であり、製品の欠陥ではありません。アクティビティサービスが復旧すると、イベント配信が自動的に再開されます。

> **注釈**
>
> 監視対象データベース自分自身で失われるスキーマ変更データはありません。アクティビティサービスが再び利用できるようになるまで、HM コンソールへの検出された変更のレポートのみが遅延されます。 EDB Postgres AI エージェントはスキーマスナップショットの収集を継続し、アクティビティサービスが回復すると、サーバーは変更を検出して報告します。

Controlling schema ingestion

beacon-agent.yaml により、エージェントは、移行ユーザーが権限を持ち、少なくとも 1 つのテーブルを含むすべてのスキーマを取り込むことができます。

さらに構成を追加して、スキーマの取り込みを制御できます。

インジェスト用のスキーマのサブセットの定義

エージェントにすべてのスキーマを取り込ませたくない場合は、databases.schema.filter オプションを使用してサブセットを定義できます。

スキーマフィルターを使用する

エージェントがデータベースからどのスキーマを取り込むかを制御するには、beacong-agent.yaml の databases.schema 構成ブロック内に filter パラメーターを追加します。これにより、含めるまたは除外するスキーマのセットを指定できます。

Filter パラメーターには 2 つの設定が必要です。

- mode フィルタリング動作を定義します。 include または exclude のいずれかに設定する必要があります。 include に設定されている場合、エージェントは名前リストにリストされているスキーマのみを取り込みます。exclude に設定されている場合、エージェントは名前リストにリストされているものを除くすべてのスキーマを取り込みます。
- names フィルターを適用するスキーマ名のリスト。名前は大文字と小文字が区別されるため、スキーマがソースデータベースに保存される方法と同じ大文字と小文字を名前に使用してください。

> **注釈**
>
> エージェントには、Oracle と Postgres の両方にデフォルトの 除外されるシステムスキーマ があり、データベースの移行とは無関係であるため、取り込みから自動的に除外されます。 *exclude* フィルターを使用する場合、このデフォルトリストは指定した追加スキーマと組み合わせられます。 *include* フィルタ

の場合、エージェントは明示的に名前付けたスキーマを取り込みます。デフォルトで除外されるシステムスキーマを含みます。

インクルードの例

```
[...]
provider:
  onprem:
    clusters:
      - resource_id: "<cluster_resource_id>"
        [...]
        nodes:
          - resource_id: "<node_resource_id>"
            dsn: $DSN
            schema:
              enabled: true
              poll_interval: 15s
              filter:
                mode: "include"
                names:
                  - "<schema_name1>"
                  - "<schema_name2>"
            tags:
              - "<tag_names>"
```

例の除外

```
[...]
provider:
  onprem:
    clusters:
      - resource_id: "<cluster_resource_id>"
       [..]
       nodes:
         - resource_id: "<node_resource_id>"
           dsn: $DSN
           schema:
             enabled: true
             poll_interval: 15s
             filter:
               mode: "exclude"
               names:
                 - "<schema_name1>"
                 - "<schema_name2>"
           tags:
```

(次のページに続く)

(前のページからの続き)

```
            - "<tag_names>"
```

除外されるシステムスキーマ

エージェントは、デフォルトで次のシステムスキーマを除外します。

Postgres および EDB Postgres Extended Server

- information_schema
- pg_catalog
- pg_toast

EDB Postgres Advanced Server

- sys

Oracle

- アノニマス
- APEX_PUBLIC_USER
- APEX_030200
- APEX_040000
- APEX_040200
- APPQOSYS
- AUDSYS
- CTXSYS
- DMSYS
- DBSNMP
- DBSFWUSER
- デモ
- DIP
- DVF
- DVSYS
- EXFSYS
- FLOWS_FILES
- FLOWS_020100
- フランク
- GGSYS

- GSMADMIN_INTERNAL
- GSMCATUSER
- GSMROOTUSER
- GSMUSER
- LBACSYS
- MDDATA
- MDSYS
- MGMT_VIEW
- OJVMSYS
- OLAPSYS
- ORDPLUGINS
- ORDSYS
- ORDDATA
- OUTLN
- ORACLE_OCM
- OWBSYS
- OWBYSS_AUDIT
- PDBADMIN
- RMAN
- REMOTE_SCHEDULER_AGENT
- SI_INFORMTN_SCHEMA
- SPATIAL_CSW_ADMIN_USR
- SPATIAL_WFS_ADMIN_USR
- SQLTXADMIN
- SQLTXPLAIN
- SYS
- SYSBACKUP
- SYSDG
- SYSKM
- SYSRAC
- システム

- SYSMAN
- TSMSYS
- WKPROXY
- WKSYS
- WK_TEST
- WMSYS
- XDB
- XS$NULL

複数の同時スキーマ取り込みの実行

複数の Oracle データベースからスキーマを取り込むように EDB Postgres AI エージェントを構成する場合、 `beacon-agent.yaml` 構成ファイルの `schema_export_max_workers` パラメーターを使用して、EDB Postgres AI エージェントが並列実行されるスキーマエクスポートジョブの数を制御できます。

`schema_export_max_workers` は、EDB Postgres AI エージェントが構成済みのすべての Oracle データベースで実行される同時スキーマエクスポートの最大数を設定します。アクティブなワーカーは、EDB Postgres AI エージェントを実行する補助マシンの CPU、メモリ、および一部のディスク領域を消費するため、このパラメーターを使用して、リソース消費を調整しながら、大量のスキーマの取り込みを高速化または低速化できます。

`schema_export_max_workers` は、`beacon-agent.yaml` の `provider.onprem` セクションにあります。

```
... # other beacon-agent.yaml configurations
provider:
  onprem:
    runner:
      enabled: true
    schema_export_max_workers: 10
```

max servers パラメーターを使用する

初期スキーマインジェストを構成するときは、デフォルトの `schema_export_max_workers: 10` から始めます。次に、EDB Postgres AI エージェントを実行している補助マシン CPU、メモリ、ディスク I/O、ネットワークを監視します。

補助マシンが十分に活用されておらず、複数のデータベースのより速い取り込みが必要な場合、またはデータベースが [**エステート>移行**] タブに表示されるまでに時間がかかりすぎていることに気づいた場合、値を増やして影響を観察します。

リソースの使用量が一貫して高いか、競合を引き起こしている場合たとえば、CPU、メモリ、ディスク使用量のスパイクに気づいた場合、値を減らし、システムが安定することを確認します。これにより、合計の取り込み時間は遅くなりますが、EDB Postgres AI エージェントホストでのピークリソース使用量が削減されます。

変更するたびに、エージェントが自動リロードを使用するサービスとして実行されていない場合は、エージェントを再起動します。

Adding usage reporting for Hybrid Manager-monitored databases

すべて統合エージェント展開から製品の使用状況に関する貴重な洞察を EDB に提供することにより、モニタリング活動を補完できます。使用レポートが有効になっている場合、エージェントは基本的な使用データを収集して、EDB が顧客をより深く理解できるようにします。有効にした場合にのみ EDB にデータを送信します。

インターネットに接続された環境の場合

1. *agent configuration example* では、 `beacon_agent.yaml` ファイルに既に `general.metrics.usage.output.file.enabled` が `true` に設定されています。この設定により、エージェントは使用データのローカルコピーをファイル `beacon_usage.json` に保存できます。
2. このファイルを確認して、情報が正確で、EDB と共有したくないデータが含まれていないことを確認します。
3. `general.metrics.usage.output.http.enabled` を `true` に設定して、EDB へのレポートを有効にします。

エアギャップ環境

1. *agent configuration example* では、 `beacon_agent.yaml` ファイルに既に `general.metrics.usage.output.file.enabled` が `true` に設定されています。この設定により、エージェントは使用データのローカルコピーをファイル `beacon_usage.json` に保存できます。
2. このファイルを確認して、情報が正確で、EDB と共有したくないデータが含まれていないことを確認します。
3. `beacon_usage.json` レポートファイルを、インターネットにアクセスできるマシンにコピーします。

Running the agent

エージェントを実行します。

```
beacon-agent
```

> **Tip**
>
> 他のエージェントモードとオプションについては、 Agent CLI を参照してください。

エージェントはハイブリッドマネージャー HM へのデータの送信を開始します。ログに従って、データの取り込みプロセスを監視します。数分後、エージェントからのデータが HM コンソールに表示されます。

- **Estate** セクションの **Postgres** タブには、Postgres データベースと要約されたデータベース情報がリストされます。いずれかのデータベースを選択すると、より詳細なデータベースの概要と監視機能が提供されます。

- **Estate > Migrations** タブには、外部データベースたとえば Oracle と、要約されたデータベースと移行情報がリストされます。いずれかのデータベースを選択すると、より詳細な移行評価と移行機能が提供されます。

> **重要**
>
> エージェントは、実行中にテーブルと列のスキーマ変更を検出します。ただし、ビューまたはプロシージャへの変更は、エージェントを再起動するまで検出されません。

本番使用の場合、エージェントを *run as a service* に構成します。

Running the agent as a service

継続的なメトリックストリーミングのために、データベースがハイブリッドマネージャー HM エステートで可視されたままであるように保証には、サービスとして実行するようにエージェントを構成します。サービスとして実行すると、システムの起動時に自動的に起動し、エラーが発生した後に再起動します。

エージェントをサービスとして設定しない場合、手動で実行した場合にのみメトリックがストリーミングされます。さらに、データベースは、エージェントがストリーミングを停止してから 72 時間後に HM 不動産から消えます。

サービスとして設定する前に、エージェントのインストールと構成を完了します。

> **注釈**
>
> エージェントパッケージの将来のバージョンでは、これが自動的に設定される可能性があります。

この例では、RHEL 9 マシンで実行されるようにエージェントを構成します。ただし、値を置き換える時期がわかるように、オペレーティングシステム支援を提供します。

1. 選択したテキストエディタを使用してファイルを作成します。このコマンド例では、vi エディターを使用します。

```
sudo vi /usr/lib/systemd/system/beacon-agent.service
```

AWS EC2 インスタンスの他のデフォルトのシステムディレクトリは次のとおりです。

Operating system	Default system directory
RHEL	/usr/lib/systemd/system/beacon-agent.service
Rocky Linux	/usr/lib/systemd/system/beacon-agent.service
AlmaLinux	/usr/lib/systemd/system/beacon-agent.service
Ubuntu	/etc/systemd/system/beacon-agent.service
Debian	/lib/systemd/system/beacon-agent.service

1. 次のテンプレートを `beacon-agent.service` ファイルに貼り付けて、プレースホルダー < > の情報を置き換えます。

```
# TEMPLATE

[Unit]
Description=Agent

After=network.target

[Service]
Type=simple

User=<user>

WorkingDirectory=</home/user/>

ExecStart=</usr/local/bin/beacon-agent>

Restart=on-failure
RestartSec=60

[Install]
WantedBy=multi-user.target
```

Service settings	Placeholder to replace	Guidelines
User=	*<user>*	Provide the name of the system user that will run the agent.
WorkingDirectory=	*</home/user/>*	Set the working directory to where you stored the *.beacon* directory. The path depends on the operating system.
ExecStart=	*</usr/local/bin/beacon-agent>*	Provide the path to the directory where the agent was installed. The path depends on the operating system.

1. 次のコマンドを実行して、新しいサービスをリロード、有効化、および開始します。これらのコマンドは、前述のすべてのオペレーティングシステムに適用されます。

```
sudo systemctl daemon-reload
sudo systemctl enable beacon-agent.service
sudo systemctl start beacon-agent.service
```

1. エージェントはサービスとして実行されています。ログを確認します。

```
journalctl -u beacon-agent.service
```

Troubleshooting

EDB Postgres AI エージェントの構成に問題がある場合、または実行しようとしたときにエラーが発生した場合は、トラブルシューティングを行うことができます。各 EDB Postgres AI エージェント機能を個別にテストして、問題の原因を絞り込みます。

EDB Postgres AI エージェントからハイブリッドマネージャーへの接続のテスト

EDB Postgres AI エージェントとハイブリッドマネージャー HM 間の接続があることを確認するには、EDB Postgres AI エージェントを実行します。

```
beacon-agent --config
```

他の EDB Postgres AI エージェントモードとオプションについては、EDB Postgres AI Agent CLI を参照してください。

出力にエラーが表示されない場合、HM への接続は正常に機能しています。問題は、データベースへの接続に関係している可能性があります。

出力にエラーが表示される場合、HM への接続は正常に機能していません。YAML ファイルの構文、アクセスキー、CA パスとコンテンツ、DSN 文字列、およびプロジェクト ID を次々に確認して、設定が正しいことを確認します。

エラーメッセージと解決策

`"fail fetching server version: pq: SSL is not enabled on the server"`

接続しているデータベースの SSL が無効になっている場合、暗号化されていない接続を使用するように EDB Postgres AI エージェントに明示的に指示する必要があります。これを行うには、設定`?sslmode=disable`を *EDB Postgres AI Agent's configuration* で使用される DSN に追加します。例

```
export DSN="postgresql://postgres:password@localhost:5432/postgres?sslmode=disable"
```

スキーマ変更イベントがハイブリッドマネージャーに表示されない

スキーマ変更イベントまたはその他の EDB Postgres AI エージェントテレメトリが HM コンソールに表示されない場合、アクティビティサービス `accm-server` は機能が低下しているか、使用できない可能性があります。

`accm-server` ポッドのステータスを確認するには、次のコマンドを実行します。

```
kubectl get pods -A | grep accm
upm-beaco-ff-base               accm-server-8c5cc7676-f2zv2
           2/2     Running     0             8h
```

症状

- EDB Postgres AI エージェントが正常に実行されているにもかかわらず、スキーマの変更は HM コンソールに反映されません。
- 他のイベント駆動機能新しいクエリ検出なども更新されません。

- EDB Postgres AI エージェントログには、プラットフォームへのイベントの送信中に次のようなエラーが表示されます。

```
{"level":"warn","error":"rpc error: code = Unavailable desc = no healthy upstream",
↪"project_id":"prj_xxxx","cluster_id":"p-xxxxx","table":"edb_admin.public.test_18653
↪","time":1781518934,"msg":"failed to emit schema change activity"}
```

解像度

これは、アクティビティサービスがダウンしているときに予想される動作であり、EDB Postgres AI エージェントまたは監視対象データベースの欠陥ではありません。 EDB Postgres AI エージェントは、ローカルで変更を検出し続け、配信のためにイベントをキューに入れます。アクティビティサービスが回復すると、キューに入れられたイベントが自動的に配信され、HM コンソールが更新されます。

問題が解決しない場合は、HM Kubernetes 展開で `accm-server` ポッドが実行されており、正常であることを確認します。

Administrative tasks

エージェントを構成するには、ユーザーが自分自身では取得できない一部のリソースにアクセスする必要があります。ただし、管理者は、Hybrid ManagerHM のインストール中またはその後に証明書とホスト関連の設定を構成する権利と権限を持っているため、これらのリソースを利用できます。

`beacon_server` とは何ですか？

エージェントを構成するには、ユーザーが `beacon_agent.yaml` ファイルで `beacon_server` 値を設定する必要があります。これは、ホストとポートの 2 つの値で構成される文字列です。

ホストは、インストールまたは `values.yaml` に使用される Helm チャート構成ファイルで提供されている URL です。`upm-beacon.server_host` または `BEACON_SERVICE_DOMAIN_NAME` に設定された値を検索して、URL を取得します。ポートは 9443 に固定されています。

ルート認証局 CA とは何ですか?

エージェントは、TLS を使用して `beacon_server` URL への安全な接続を確立します。したがって、エージェントが実行されているマシンは、`beacon_server` が提供するサーバー証明書を信頼する必要があります。通常、この信頼は組織の既存のセキュリティインフラストラクチャによって既に管理されています。内部サーバー証明書の署名を担当する認証局 CA は、ネットワーク全体で信頼できる CA として構成する必要があります。これにより、エージェントを実行しているサーバーを含む環境内のサーバーが、この CA が発行した証明書を本質的に信頼することが保証されます。

これに当てはまらない場合、ユーザーは `beacon_agent.yaml` ファイルで `root_ca_path` 値を設定できます。これは、ユーザーが EDB Postgres AI Platform サーバー証明書を検証できるルート証明書を保存したディレクトリです。ただし、非管理ユーザーは証明書にアクセスできないため、それらに証明書を提供する必要があります。

ルート証明書を含む Kubernetes シークレットを表示するには

```
#  The certificate is base64 encoded as the data field "public.crt"

kubectl get secret -n upm-beacon beaconator-ca-bundle -o json
```

ルート証明書のみを印刷する読みやすい出力については、 jq および base64 をインストールして実行していることを確認します。

```
#  If `jq` and `base64` are available

kubectl get secret -n upm-beacon beaconator-ca-bundle -o json | jq -r .data."public.
↪crt" | base64 --decode
```

このコマンドは、証明書の内容を出力します。安全なチャネルを介してそれらを必要とするユーザーに送信します。ユーザーは crt 形式でローカルに保存する必要があります。

Data security and sovereignty

ハイブリッドマネージャー HM で外部データベースをモニタするエージェントを展開すると、セキュリティとコンプライアンス要件に合わせて、モニタリングと評価のために収集されたデータの制御を維持します。

限定的なデータ収集

エージェントは次のデータを収集します。

- **Estate** ビューにメトリックを表示するための一般的なデータベースメタデータ。

Postgres データベースの場合

- **外部クラウドホスト** *OCR* **モデルは、** *HM* **コンソールを介して登録に失敗します** および *Agent Factory* **メトリックの** *Grafana* **ダッシュボードの設定** を介して可観測機能を提供するデータベースログとメトリック。

Oracle データベースの場合

- 移行の複雑さと作業レベルの自動評価を生成するデータベースデータは、[**移行**] タブの **エステート** セクションに表示されます。
- スキーマ変換にそのツールを使用する場合、移行ポータルによる最終的な取得と使用のためのスキーマ DDL 。

> **注釈**
>
> 現在、移行評価データとスキーマ DDL データは、Oracle データベースでのみ収集されます。新しいリリースでは、外部の Postgres およびクラウドサービスプロバイダー CSP データベースの移行評価とスキーマ DDL データも収集します。

当社は、これらの目的に厳密に必要な範囲を超えてデータを収集することはありません。

データの保存場所

収集されたデータは、選択した環境内に残ります。HM はお客様のデータ主権を尊重します。たとえば、HM を独自のプレミスに展開した場合、監視対象データベースから収集されたデータはすべてオンプレミスインフラストラクチャ内に残ります。HM インスタンスが EKS のようなクラウドプラットフォームに展開されている場合、収集されたデータは特定のクラウド環境に安全に存在します。

Installing the Agent

エージェントのインストール方法については、*Vault* **のインストール** を参照してください。

Monitoring cloud service provider-managed Postgres deployments

完全な不動産表現サポートを備えたクラウドサービスプロバイダー CSP によって管理される環境で、ハイブリッドマネージャー HM の外部で実行されている PostgreSQL データベースを簡単に監視できます。

現在、次の外部 CSP をモニタできます。

- *Monitoring AWS RDS deployments*

Monitoring AWS RDS deployments

ハイブリッドマネージャー HM を使用して、アマゾンウェブサービス AWS リレーショナルデータベースサービス RDS 展開をモニタできます。

前提条件

HM が存在する Kubernetes 環境にアクセスし、HM クラスター全体の設定を変更するための十分な管理者権限が必要です。

このアクセス要件は、必要な特権がより少ない Postgres の *Monitoring self-managed deployments* 展開など、HM を使用した他の外部データベースモニタリング状況とは異なります。

また、Kubernetes シークレットに保存されているユーザー資格情報たとえば `example-beacon-csp-credentials` に、監視対象の AWS RDS 環境に対する十分な権限があることを確認する必要があります。これらの権限がないと、システムは使用可能なすべてのメトリックを取得できません。

これを行うには、次の最小限の権限を持つ IAM ユーザーの AWS アクセスキーを取得する必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowInstanceAndLogDescriptions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBInstances",
                "rds:DescribeDBLogFiles"
            ],
```

(次のページに続く)

(前のページからの続き)

```
            "Resource": [
                "arn:aws:rds: *:* :db:*"
            ]
        },
        {
            "Sid": "AllowMaintenanceDescriptions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribePendingMaintenanceActions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowGettingCloudWatchMetrics",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowRDSUsageDescriptions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowQuotaDescriptions",
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowInstanceTypeDescriptions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
```

(次のページに続く)

(前のページからの続き)

```
            "Sid": "AllowInstanceFilterByTags",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        }
    ]
}
```

HM で AWS RDS 展開をモニタするための EDB Postgres AI エージェントの設定

HM ですべての AWS RDS 展開をモニタするようにエージェントを設定できます。

> **注釈**
>
> RDS リージョンを登録すると、システムはその RDS リージョン内にあるすべてのデータベースを追加します。これにより、複数のリソースが不動産に追加される場合があります。これらはすべて 1 つのプロジェクトの下にあるため、ユーザーとしてすべての RDS データベースにアクセスできるか、それらのいずれにもアクセスできません。

1. `example-beacon-csp-credentials` シークレットを追加します。

`example-beacon-csp-credentials` シークレットを HM Kubernetes クラスターの `upm-beacon` 名前空間に追加します。

例 `yaml` サンプル

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
    name: example-beacon-csp-credentials
    namespace: upm-beacon
stringData:
    AWS_ACCESS_KEY_ID: <your_access_key_id>
    AWS_SECRET_ACCESS_KEY: <your_secret_access_key>
    AWS_SESSION_TOKEN: <your_session_token>
    AWS_REGION: example-us-east-1
    BEACON_PROVIDER_AWS_CONSTANT_REGIONS: example-us-east-1
    BEACON_PROVIDER_AWS_CONSTANT_PROJECT_ID: <your_project_id>
    BEACON_AGENT_PROVIDERS: appliance,aws
    BEACON_PROVIDER_AWS_MODE: constant
    BEACON_PROVIDER_AWS_CONSTANT_SERVICES: rds
EOF
```

2. エージェントと外部メトリックレシーバーを再起動します。

```
kubectl -n upm-beacon rollout restart deployment upm-beacon-agent-k8s
kubectl -n upm-beacon rollout restart deployment upm-beacon-external-metrics-receiver
```

Estate ページに AWS RDS 展開が表示されます。データベースへの入力が開始されると、5 つのメトリックを表示できます。

- CPU
- ネットワークスループット
- 接続
- ディスク IOPS
- ディスクスループット

HM での AWS RDS モニタリングの無効化

AWS RDS 展開のモニタリングをオフにできます。

1. `beacon-csp-credentials` シークレットを削除します。

```
kubectl -n upm-beacon delete secret beacon-csp-credentials
```

2. エージェントと外部メトリックレシーバーを再起動します。

```
kubectl -n upm-beacon rollout restart deployment upm-beacon-agent-k8s
kubectl -n upm-beacon rollout restart deployment upm-beacon-external-metrics-receiver
```

エステートで AWS RDS 展開は表示されなくなり、HM でモニタリングされなくなります。

Monitoring standalone Kubernetes clusters

EDB Postgres ® AI for CloudNativePG™ または CloudNativePG™ クラスターの集中モニタリング、メトリック収集、およびリソース管理に EDB Postgres AI エージェントを使用し、Hybrid ManagerHM コンソールに統合します。これらのクラスターを HM コンソールと統合することにより、クラウドネイティブ環境の統合ビューを取得します。この統合は、Helm チャートを使用して展開されます。

EDB Postgres AI エージェントをどこでも実行するように構成できます。特定の機能をアクティブにするには、1 つ以上のロール Webhook、プロバイダー、ディスパッチャー、またはランナーを割り当てます。これらの機能の任意の組み合わせを同時に実行するように単一のエージェントを構成できます。

アーキテクチャ

Kubernetes 展開の監視を実装するアーキテクチャは、次のコンポーネントで構成されています。

EDB Postgres AI エージェント

EDB Postgres AI エージェントは、リソースの検出とモニタリングのための集中ツールとして機能し、メトリックの収集と転送を合理化します。

OpenTelemetry コレクターオプショナル

OpenTelemetryOTEL コレクターは、ログを収集し、環境からテレメトリデータをエクスポートする集中化されたデータパイプラインとして機能します。

主な機能

- コンテナメトリック cAdvisor cAdvisor サービスは、アクティブなコンテナからリソース使用データ CPU、メモリを収集および処理します。
- Kubelet メトリック Kubernetes ノードとポッドの状態とパフォーマンスを監視します。
- アプリケーションログデータベースアプリケーションからログを集計および転送します。
- カスタムメトリック特定のアプリケーションのニーズに合わせて調整されたユーザー定義メトリックの収集が可能です。
- オプションの Kube 状態メトリック Kubernetes オブジェクトの状態に基づいてメトリックを生成します。

Requirements to install the EDB Postgres AI agent

このセクションでは、外部 CloudNativePG™、 CloudNativePG™クラスターの EDB Postgres ® AI、および CloudNativePG™グローバルクラスタークラスターの EDB Postgres Postgres ® AI をモニタするようにハイブリッドマネージャー HM インストールを構成するための要件をリストしています。

HM コンソールの要件

次の方法で HM コンソールインスタンスにアクセスする必要があります。

- マシンユーザー HM クラスターの準備ができたら、 マシンユーザーを作成する 。
- エステートインジェストロール作成したマシンユーザーに *estate ingest role* を割り当てます。
- アクセスキーマシンユーザーの作成中に生成されたアクセスキーを保存します。
- プロジェクト ID エージェントがメトリックを報告するプロジェクト ID をメモします。

必要な実行可能ファイル

このドキュメントのコマンドを実行するには、次の実行可能ファイルが必要です。

- kubectl
- ヘルム
- envsubst

kubectl の cnp および ns プラグインもお勧めします。

サポートされている演算子

次のいずれかのオペレーターによって管理される Postgres クラスターを使用した既存の外部 Kubernetes クラスター HM がインストールされる Kubernetes クラスター以外が必要です。

- EDB Postgres ® AI for CloudNativePG™ Cluster (formerly EDB Postgres for Kubernetes)
- EDB Postgres ® AI for CloudNativePG™ Global Cluster (formerly EDB Postgres Distributed for Kubernetes)
- CloudNativePG™

Kubernetes クラスター要件

Postgres データベースを実行している Kubernetes クラスターも、これらの要件を満たしていることを確認します。

- Kubernetes バージョン 1.21 以降。
- Helm バージョン 3.8 以降。
- クラスターアクセス名前空間、RBAC リソース、および展開を作成する管理者権限。
- メトリックをモニタするには Kubernetes オペレーターが必要です。

Cert-manager

展開を完了するには、クラスターに cert-manager をインストールする必要があります。

cert-manager をインストールするには

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.9.
↪1/cert-manager.crds.yaml
```

必須の環境値

展開を完了するために必要な次の環境値をエクスポートします。展開中に環境値を明示的に提供する場合、この手順をスキップできます。

- `TARGET_NS` - 既存の CNPG クラスターまたは PGD4K の PGD グループの名前空間。
- `AGENT_NS` - EDB Postgres AI エージェントをインストールする名前空間。
- `HM_ACCESS_KEY` - Hybrid Manager のアクセスキー。
- `HM_PROJECT_ID` - HM のプロジェクト ID。
- `HM_CACERT` - HM のルート CA 証明書。 `rootCAEnabled` が `true` デフォルトの場合、このガイドで説明する構成である場合に必要です。チェーンから提供される任意の証明書で動作します。代わりに、システム CA プールを介して HM を信頼するように `rootCAEnabled: false` を設定する場合、または検証をスキップする場合、この値と **名前空間とシークレット** の `beacon-server-ca` シークレットを省略できます。 *TLS* **構成オプション** を参照してください。

- HM_URL - ポートを含むビーコンサーバー URL
- EDB_REPO - サブスクリプションと一致する EDB リポジトリの名前 enterprised、standard または community360
- EDB_SUBSCRIPTION_TOKEN - EDB トークン。
- POD_SELECTOR - EDB オペレーターの場合は k8s.enterprisedb.io/podRole=instance に、コミュニティ CNPG オペレーターの場合は cnpg.io/podRole=instance に設定します。
- AGENT_VERSION - HM 2026.2 の v1.337.18 に設定します。

例

```
export TARGET_NS=cnpg
export AGENT_NS=upm-beacon
export HM_ACCESS_KEY=baak_1DMvAduJeigwIcCJyk2pc6BxllJxAbjFa5Nlc1c680lqSj0sj
export HM_PROJECT_ID=prj_3AlRuOvlbUZafFY2
export HM_CACERT=$(cat source/k8s-agent/ca.cert)
export HM_URL=beacon-pm-test.edbhcp.com:9443
export EDB_REPO=enterprise
export POD_SELECTOR=k8s.enterprisedb.io/podRole=instance
export AGENT_VERSION=v1.337.18
```

名前空間とシークレット

インストールする前に、次の名前空間とシークレットを作成します。

- 名前空間 EDB Postgres AI エージェントを分離するために使用される論理パーティション${AGENT_NS}。
- ビーコンサーバー CA シークレット HM に接続するための TLS 証明書が含まれています。
- ビーコンアクセスキー aecret クラスターの HM アクセスキーが含まれています。
- イメージプルのシークレットエージェントおよび OpenTelemetryOTEL コレクターイメージをプルするための資格情報が含まれています。

構成に必要な名前空間とシークレットを作成します。

```
kubectl create namespace "${AGENT_NS}"
kubectl create secret generic beacon-server-ca --from-literal=ca.crt="${HM_CACERT}" -
↪n "${TARGET_NS}"
kubectl create secret generic beacon-server-ca --from-literal=ca.crt="${HM_CACERT}" -
↪n "${AGENT_NS}"
kubectl create secret generic beacon-access-key --from-literal=access-key="${HM_
↪ACCESS_KEY}" -n "${TARGET_NS}"
kubectl get secrets -n "$AGENT_NS" -oyaml beacon-agent-pull-secret | grep -v ^\
↪s*namespace:\s | kubectl apply -n "$TARGET_NS" -f -
```

注釈

2 つの `beacon-server-ca` シークレットは、 *rootCAEnabled* が `true` デフォルトの場合にのみ必要です。 *rootCAEnabled: false* を設定する場合、作成をスキップします。 *TLS* **構成オプション** を参照してください。 !!!

Installing the EDB Postgres AI agent

インストール方法

重要

外部 Kubernetes クラスターの EDB Postgres AI エージェントのインストールを続行する前に、 **必須の環境値** を設定します。

Environment	Configuration	OpenShift Flag
KIND, GKE, EKS, AKS	Uses image defaults (numeric or named)	openshift.enabled: false (default)
OpenShift (RHOS)	Compatible via runAsUser: null (SCC assigns ID)	openshift.enabled: false/true (optional)

キーポイント

- **ユニバーサル互換性** チャートのデフォルトは `runAsUser: null` であるため、OpenShift SCC は有効な UID を自動的に割り当てることができますが、標準の Kubernetes はイメージのデフォルトユーザーにフォールバックします。
- **セキュリティコンテキスト** runAsNonRoot 戦略は、標準の Kubernetes 検証ロジックと OpenShift の名前付けユーザーサポートとの互換性を確保するためにコンポーネントによって異なります。
- **OTEL コレクター**明示的な数値 ID なしでイメージのデフォルトユーザー非ルートを使用して、厳密な K8s 検証と OpenShift SCC の両方を満たします。

Helm チャートのダウンロードと値ファイルのカスタマイズ

1. EDB サブスクリプショントークンを取得します。
2. EDB リポジトリから EDB Postgres AI エージェントチャートをダウンロードします。

```
helm repo add enterprisedb-$EDB_REPO "https://downloads.enterprisedb.com/$EDB_
↪SUBSCRIPTION_TOKEN/$EDB_REPO/helm/charts/"
```

3. Hybrid ManagerHM アクセス資格情報を含む `custom-values.yaml` ファイルを作成します。

```
# Required: Your HM access credentials
agentConfig:
```

(次のページに続く)

(前のページからの続き)

```
  accessKey: ${HM_ACCESS_KEY}
  projectID: ${HM_PROJECT_ID}
  beaconServer:
    host: ${HM_URL}
    plaintext: false
  metrics:
    push:
      endpoint: ${HM_URL}
  webhook:
    enabled: true
    podSelector: ${POD_SELECTOR}
# Required: Docker registry credentials for EDB images
image:
  repository: docker.enterprisedb.com/${EDB_REPO}/upm-beacon-agent
  tag: ${AGENT_VERSION}
  imageCredentials:
    registry: docker.enterprisedb.com
    username: ${EDB_REPO}
    password: ${EDB_SUBSCRIPTION_TOKEN}

# OTEL collector image
otelImage:
  repository: "docker.enterprisedb.com/${EDB_REPO}/upm-beacon-external-metrics-
↪receiver"
  tag: ${AGENT_VERSION}

# Optional: Enable additional monitoring components
additionalCharts:
  cert-manager:
    install: false
  opentelemetry-collector:
    install: true # Enable for metrics collection
  kube-state-metrics:
    install: true # Enable for Kubernetes state metrics
  prometheus-node-exporter:
    install: true # Enable for node metrics (requires privileged access)

# Security context for the agent
securityContext:
  runAsUser: 65532
  runAsGroup: 65532
podSecurityContext:
  fsGroup: null

# kube-state-metrics uses the image default user
```

(次のページに続く)

(前のページからの続き)

```
kube-state-metrics:
  podSecurityContext: null

# Required if opentelemetry-collector.install is true:
opentelemetry-collector:
  image:
    repository: "docker.enterprisedb.com/${EDB_REPO}/upm-beacon-external-metrics-
↪receiver"
    tag: ${AGENT_VERSION}
  podSecurityContext:
    seccompProfile:
      type: RuntimeDefault
  securityContext:
    runAsUser: 65532
    runAsGroup: 65532
    runAsNonRoot: true
    allowPrivilegeEscalation: false
    capabilities:
      drop:
        - ALL
```

他の構成パラメーターとオプションについては、*Configuration reference* を参照してください。

4. helm チャートを適用します。

- 標準の Kubernetes GKE、EKS、AKS、KIND の場合

```
helm upgrade --install beacon-agent-k8s \
 enterprisedb-$EDB_REPO/beacon-agent-k8s \
 --create-namespace \
 --namespace upm-beacon \
 --values custom-values.yaml
```

- OpenShift クラスターの場合、`--set openshift.enabled=true` フラグを追加します。

```
helm upgrade --install beacon-agent-k8s \
 enterprisedb-$EDB_REPO/beacon-agent-k8s \
 --create-namespace \
 --namespace upm-beacon \
 --set openshift.enabled=true \
 --values custom-values.yaml
```

重要

OTEL コレクターを使用する場合に SCC を付与します。

```
oc adm policy add-scc-to-user privileged \
  system:serviceaccount:upm-beacon:beacon-agent-k8s-opentelemetry-collector
kubectl -n upm-beacon rollout restart daemonset beacon-agent-k8s-opentelemetry-
↪collector-agent
```

```
kubectl get secrets -n "$AGENT_NS" -oyaml beacon-agent-pull-secret | grep -v ^\
↪s*namespace:\s | kubectl apply -n "$TARGET_NS" -f -
```

監視拡張機能を使用した外部クラスターの展開

EDB Postgres AI エージェントは、監視対象クラスターに存在する必要がある拡張機能からクエリーと待機状態のメトリックを収集します。データベース`${PG_DATABASE}` および `postgresql` 設定で次の `bootstrap.initdb.postInitApplicationSQL` コマンドを使用して外部クラスターを展開して、クラスターが初期化されたときに監視拡張機能が作成およびプリロードされるようにします。

注釈

マニフェストが `bootstrap.initdb.owner` を設定していないため、CNPG は所有者をデフォルトのデータベース名にします。その結果、CNPG は、ターゲットデータベース `${PG_DATABASE}` とまったく同じ名前で非特権の Postgres ロール user を作成し、そのデータベースの所有者にします。 init SQL の許可は、*${PG_DATABASE}* 名前でこのロールを参照します。

- `${PG_DATABASE}` に `edb_internals` スキーマを作成し、それに `USAGE` をデータベース所有者ロール`${PG_DATABASE}`、CNPG がデータベースと同じ名前で作成します。および `pg_monitor` に付与します。
- そのスキーマに `edb_stat_monitor` 、`query_advisor` 、および `edb_wait_states` 拡張機能を作成します。
- データベース `search_path` および `timezone` を設定します。

`postgresql` ブロックは `pg_stat_statements` を有効にし、`shared_preload_libraries` を介して `aidb` 、`edb_stat_monitor` 、`query_advisor` 、および `edb_wait_states` をプリロードします。

重要

${PG_IMAGE} は、 *aidb* 、*edb_stat_monitor* 、*query_advisor* 、および `edb_wait_states` 拡張機能をバンドルする EDB Postgres AI イメージである必要があります。コミュニティマニフェストは、アップストリームの CloudNativePG オペレーター API を使用しますが、引き続きこれらの EDB 拡張機能が必要です。

イメージ名は、実行する Postgres ディストリビューションに応じて、`postgresql` 、`edb-postgres-advanced` 、または `edb-postgres-extended` になります。

```
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
```

(次のページに続く)

(前のページからの続き)

```
metadata:
  name: ${PG_CLUSTER_NAME}
  namespace: ${TARGET_NS}
  labels:
    k8s.enterprisedb.io/beacon: enabled
spec:
  backup:
    target: primary
  plugins:
    - name: barman-cloud.cloudnative-pg.io
      isWALArchiver: true
      parameters:
        barmanObjectName: ${PG_CLUSTER_NAME}-bck
  bootstrap:
    initdb:
      database: ${PG_DATABASE}
      postInitApplicationSQL:
        - CREATE SCHEMA IF NOT EXISTS edb_internals
        - GRANT USAGE ON SCHEMA edb_internals TO ${PG_DATABASE}
        - GRANT USAGE ON SCHEMA edb_internals TO pg_monitor
        - CREATE EXTENSION IF NOT EXISTS edb_stat_monitor WITH SCHEMA edb_internals
        - CREATE EXTENSION IF NOT EXISTS query_advisor WITH SCHEMA edb_internals
        - CREATE EXTENSION IF NOT EXISTS edb_wait_states WITH SCHEMA edb_internals
        - ALTER DATABASE ${PG_DATABASE} SET search_path TO "\$user", public, edb_
↪internals
        - ALTER DATABASE ${PG_DATABASE} SET timezone TO UTC
  imageName: ${PG_IMAGE}
  postgresql:
    parameters:
      pg_stat_statements.max: "10000"
      pg_stat_statements.track: all
    shared_preload_libraries:
      - aidb
      - edb_stat_monitor
      - query_advisor
      - edb_wait_states
  instances: 2
  storage:
    size: 1Gi
  inheritedMetadata:
    labels:
      k8s.enterprisedb.io/beacon: enabled
```

CloudNativePG の EDB Postgres AI エンタープライズオペレーター

エンタープライズマニフェストは EDB オペレーター API を使用し、EDB レジストリの imagePullSecrets エントリを追加して、public スキーマの USAGE をデータベース所有者ロール${PG_DATABASE} および pg_monitor に付与します。

```
apiVersion: postgresql.k8s.enterprisedb.io/v1
kind: Cluster
metadata:
  name: ${PG_CLUSTER_NAME}
  namespace: ${TARGET_NS}
  labels:
    k8s.enterprisedb.io/beacon: enabled
spec:
  backup:
    target: primary
  plugins:
    - name: barman-cloud.cloudnative-pg.io
      isWALArchiver: true
      parameters:
        barmanObjectName: ${PG_CLUSTER_NAME}-bck
  bootstrap:
    initdb:
      database: ${PG_DATABASE}
      postInitApplicationSQL:
        - CREATE SCHEMA IF NOT EXISTS edb_internals
        - GRANT USAGE ON SCHEMA edb_internals TO ${PG_DATABASE}
        - GRANT USAGE ON SCHEMA edb_internals TO pg_monitor
        - GRANT USAGE ON SCHEMA public TO ${PG_DATABASE}
        - GRANT USAGE ON SCHEMA public TO pg_monitor
        - CREATE EXTENSION IF NOT EXISTS edb_stat_monitor WITH SCHEMA edb_internals
        - CREATE EXTENSION IF NOT EXISTS query_advisor WITH SCHEMA edb_internals
        - CREATE EXTENSION IF NOT EXISTS edb_wait_states WITH SCHEMA edb_internals
        - ALTER DATABASE ${PG_DATABASE} SET search_path TO "\$user", public, edb_
↪internals
        - ALTER DATABASE ${PG_DATABASE} SET timezone TO UTC
  imagePullSecrets:
    - name: edb-pull-secret
  imageName: ${PG_IMAGE}
  postgresql:
    parameters:
      pg_stat_statements.max: "10000"
      pg_stat_statements.track: all
    shared_preload_libraries:
      - aidb
      - edb_stat_monitor
```

(次のページに続く)

(前のページからの続き)

```
      - query_advisor
      - edb_wait_states
  inheritedMetadata:
    labels:
      k8s.enterprisedb.io/beacon: enabled
  instances: 2
  storage:
    size: 1Gi
```

kubectl apply -f <manifest>.yaml を使用してマニフェストを適用します環境値でパラメーター化する場合、最初に envsubst を使用します。

OpenShift で使用する場合、上流のプラグインマニフェストは restricted-v2 SCC によって拒否されます。古い seccomp アノテーションと runAsUser: 10001 を設定するためです。両方を剥がした状態で取り付けます。エンタープライズ EDB オペレーターを使用する場合、postgresql.k8s.enterprisedb.io API グループの clusters および clusters/finalizers で patch /update でプラグイン get /list /update を付与する必要もあります。上流プラグインはコミュニティ postgresql.cnpg.io グループでのみこれを付与するため、それがないとクラスターはリコンシルしません cannot set blockOwnerDeletion … can't set finalizers 。

このセクションでは、単一のクラスターにパッチを適用する方法を示します。以下の手順を適応して、多くのクラスターにパッチを適用できます。

1. パッチを適用するターゲットリソースを定義します。エージェントを追加する CloudNativePG™クラスターまたは PGD グループ CloudNativePG™クラスターの EDB Postgres ® AI の場合の TARGET_NAME をエクスポートします。

```
export TARGET_UID="$(kubectl get "${TARGET_TYPE}" "${TARGET_NAME}" -n "${TARGET_NS}"␣
↪-o jsonpath={.metadata.uid})"
```

2. 構成パッチを作成します。このテンプレートは、必要なラベルを追加し、エージェントが必要なイメージをプルできるようにします。

```
PATCH_CONTENT=$(cat <<EOF
metadata:
  labels:
    k8s.enterprisedb.io/beacon: enabled
spec:
  imagePullSecrets:
  - name: beacon-agent-pull-secret
  inheritedMetadata:
    labels:
      k8s.enterprisedb.io/beacon: enabled
      k8s.enterprisedb.io/beacon-project-id: "${HM_PROJECT_ID}"
      k8s.enterprisedb.io/beacon-resource-id: "${TARGET_NAME}-${TARGET_UID}"
EOF
)
```

3. 次のコマンドを実行して、構成の変更をリソースにマージします。

```
kubectl patch pgdgroup "${TARGET_NAME}" -n "${TARGET_NS}" --type=merge -p "$PATCH_
↪CONTENT"
```

4. クラスターを再起動してエージェントをオンラインにします。

- **CNPG クラスターの場合**

```
kubectl cnp restart "$TARGET_NAME"
```

- **EDB Postgres® AI for CloudNativePG™ Cluster クラスターの場合**

グループ内の各ノードを再起動します。

```
kubectl cnp restart region-a-1
kubectl cnp restart region-a-2
kubectl cnp restart region-a-3
```

外部 Kubernetes クラスターのセキュリティコンテキストの構成

標準の Kubernetes KIND、GKE、EKS、AKS の場合

ユーザー ID とグループ ID を含むデフォルトの安全設定で `custom-values.yaml` を構成します。

```
securityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop:
      - ALL
  privileged: false
  readOnlyRootFilesystem: true
  runAsNonRoot: true
  runAsUser: 10000
  runAsGroup: 10000

podSecurityContext:
  seccompProfile:
    type: RuntimeDefault
```

OpenShift Kubernetes クラスターの場合

1. OpenShift モードを有効にして、SCC 互換性のセキュリティコンテキストを自動的に構成します。
2. `custom-values.yaml` ファイルで OpenShift サポートを有効にします。

```
openshift:
  enabled: true
```

値が `openshift.enabled: true` に設定されている場合、チャートは自動的に次のことを行います。

- コンテナセキュリティコンテキストから runAsUser および runAsGroup を省略し、SCC がそれらを割り当てられるようにします。 - capabilities.drop 、readOnlyRootFilesystem などの他のセキュリティ設定を保存します。 - ディストリビューションイメージ検証に必要な OTEL コレクタの数値 UID 65532 を使用します。

3. または、コマンドラインで-f custom-values.yaml フラグを使用して構成を指定します。

```
securityContext:
  allowPrivilegeEscalation: false
  capabilities:
    drop:
      - ALL
  privileged: false
  readOnlyRootFilesystem: true
  runAsNonRoot: true
```

4. OpenShift が自動的に割り当てるため、以下のフィールドを空のままにしておきます。

```
runAsUser:
runAsGroup:
```

5. OpenShift に展開するときは、プラットフォームがボリューム権限を自動的に管理できるように、fsGroup フィールドが未定義のままであることを確認します。

```
podSecurityContext:
  seccompProfile:
    type: RuntimeDefault
```

6. OpenShift の互換性を保証には、制限されたセキュリティコンテキストで OTEL コレクタサブチャートを構成します。

```
opentelemetry-collector:
  podSecurityContext:
    seccompProfile:
      type: RuntimeDefault
  securityContext:
    runAsNonRoot: true
    allowPrivilegeEscalation: false
    capabilities:
      drop:
        - ALL
```

7. セキュリティコンテキストを構成して、ルートアクセスを禁止し、ファイルシステムグループを介してボリューム権限を管理し、デフォルトの seccomp プロファイルを使用してシステムコールを制限します。

```
securityContext:
  enabled: true
```

(次のページに続く)

(前のページからの続き)

```
  runAsNonRoot: true
  runAsUser:
  runAsGroup:
  fsGroup:
  seccompProfile:
    type: RuntimeDefault
```

注釈

デプロイメントに標準化されたセキュリティコンテキストを実装すると、多様なコンテナプラットフォーム全体でのシームレスな互換性とセキュリティの強化が保証されます。

- Distroless イメージのサポート UID / GID `65532` は、distroless イメージの標準の非 root ユーザーであり、従来の`/etc/passwd` ファイルがありません。
- 数値検証 `runAsNonRoot: true` 指示子でユーザー ID を正しく検証するには、特定の数値 ID を設定する必要があります。
- **OpenShift の互換性 OpenShift の特権 SCC は任意の UID を許可しますが、 65532 を使用することにより、OpenShift と標準の Kubernetes 環境の両方で一貫した安全な ID が保証されます。 .. ::**
  主要な構成の注意事項
- OpenShift ID 管理 `runAsUser` 、`runAsGroup` 、および `fsGroup` フィールドは空のままで、OpenShift の SCC がプロジェクトに正しい UID および GID 範囲を自動的に割り当てます。
- セキュリティコンテキストのアクティブ化 `enabled: true` 設定により、サブチャートは内部デフォルトにフォールバックするのではなく、指定されたセキュリティ設定を適用します。
- 非ルートの適用 `runAsNonRoot: true` フラグはアクティブなままで、プラットフォームによって ID が動的に割り当てられる場合でも、コンテナが root 権限で実行されないことを保証します。
- Seccomp の強化 `RuntimeDefault` プロファイルを利用すると、コンテナがコンテナランタイムのデフォルトのセキュア syscall フィルタを使用します。

8. 読み取り専用ファイルシステムを適用し、特権エスカレーションを無効にし、デフォルトの Linux 機能をすべて削除することにより、コンテナレベルのセキュリティコンテキストを強化します。

```
containerSecurityContext:
  readOnlyRootFilesystem: true
  allowPrivilegeEscalation: false
  capabilities:
    drop:
      - ALL
```

または、コマンドラインを介してこれらの設定を適用できます。

```
helm upgrade --install beacon-agent-k8s \
  . \
```

(次のページに続く)

(前のページからの続き)

```
--namespace upm-beacon \
--create-namespace \
# ... your other flags ...
--set securityContext.runAsUser=null \
--set securityContext.runAsGroup=null \
--set podSecurityContext.fsGroup=null \
--set opentelemetry-collector.securityContext.runAsUser=null \
--set opentelemetry-collector.securityContext.runAsGroup=null \
--set kube-state-metrics.podSecurityContext=null
```

OpenShift の追加要件

OpenShift は、特定の UID 範囲を各名前空間に割り当てます。例 1000880000-1000889999

明示的な runAsUser 10000 などを指定すると、多くの場合 UID 範囲外になり、SCC 違反がトリガーされます。OpenShift のアドミッションコントローラが許可範囲から適切な値を自動的に割り当てるように、これらの値を空のままにします。

コンポーネントごとの推奨される runAsUser 構成

この表は、さまざまな SCC との互換性を保証するために、 EDB Postgres AI エージェントコンポーネントに必要な runAsUser 設定の概要を示しています。

Component	runAsUser	Reason
beacon-agent	Empty	Uses restricted SCC; OpenShift assigns UID from namespace range
kube-state-metrics	Empty	Uses restricted SCC; OpenShift assigns UID from namespace range
opentelemetry-collector	65532	Uses privileged SCC for *hostPath* or *hostPorts*.
opentelemetry-collector	65532	Requires explicit numeric UID because *runAsNonRoot: true* cannot verify non-numeric users like *nonroot*. Privileged SCC allows any UID, so *65532* works.

デーモンセットの SCC の付与

OpenShift、`opentelemetry-collector` および `prometheus-node-exporter` サブチャートでノードレベルのメトリックとログを収集するには、`hostPath` 、`hostPorts` 、および `hostNetwork` にアクセスするための昇格された権限が必要です。以下のコマンドを使用して、特権 SCC を各サービスアカウントに付与します。

注釈

OpenShift にはビルトインのノードエクスポーターが含まれているため、展開が既存のモニタリングと競合する可能性があります。代わりにプラットフォームのネイティブエクスポーターの使用を検討します。

1. 名前空間を作成しますまだ存在しない場合。

```
oc new-project upm-beacon
```

2. 特権 SCC を付与します。

```
oc adm policy add-scc-to-user privileged \
  system:serviceaccount:upm-beacon:beacon-agent-k8s-opentelemetry-collector

oc adm policy add-scc-to-user privileged \
  system:serviceaccount:upm-beacon:beacon-agent-k8s-prometheus-node-exporter
```

3. Helm チャートを展開または更新します。

```
helm upgrade --install beacon-agent . -f custom-values.yaml -n upm-beacon
```

4. デーモンセットを再起動します。

```
kubectl -n upm-beacon rollout restart daemonset beacon-agent-k8s-opentelemetry-
↪collector-agent
kubectl -n upm-beacon rollout restart daemonset beacon-agent-k8s-prometheus-node-
↪exporter
```

注釈

OpenShift では、*prometheus-node-exporter* は OpenShift のビルトイン `node-exporter` と競合する可能性があります。たとえば、ポート 9100 が既に使用されている場合。 *additionalCharts.prometheus-node-exporter.install: false* を設定し、cadvisor / kubelet スクレイピングを介してコンテナメトリックについては `opentelemetry-collector` に依存することをお勧めします。

フルロードの OpenShift クラスターでは、ノードごとの kubelet および cAdvisor スクレイプは、HM メトリックレシーバーの 4 MiB gRPC メッセージ制限を超える OTLP バッチを生成できます。それが発生すると、レシーバーは `ResourceExhausted` でバッチを拒否し、 **その中のすべてのデータポイントがドロップされる**ため、コレクタポッドが正常であっても、HM は CPU、メモリ、またはネットワークメトリックを表示しません。

コレクターのバッチプロセッサをキャップして、各プッシュが制限内に収まるようにします。

```
opentelemetry-collector:
  config:
    processors:
      batch:
```

(次のページに続く)

(前のページからの続き)

```
      send_batch_size: 1000
      send_batch_max_size: 1500
```

またはコマンドラインを介して

```
- -set opentelemetry-collector.config.processors.batch.send_batch_size=1000 \
- -set opentelemetry-collector.config.processors.batch.send_batch_max_size=1500
```

Configuration reference

コア構成オプション

Parameter	Description	Default	Required
agentConfig.accessKey	HM マシンユーザーアクセスキー	""	Yes
agentConfig.projectID	HM プロジェクト ID	""	Yes
agentConfig.beaconServer.host	HM サーバーホスト名	""	Yes
agentConfig.locationID	エージェントの場所識別子	*"standalone-k8s"*	Yes
namespace	Kubernetes 名前空間	*"upm-beacon"*	No
image.repository	ビーコンエージェントイメージ	""	Yes
image.tag	ビーコンエージェントのバージョン	""	Yes
otelImage.repository	OTEL コレクターイメージ	""	Yes (if OTEL enabled)
otelImage.tag	OTEL コレクターバージョン	""	Yes (if OTEL enabled)
additionalCharts.cert-manager.install	cert-manager サブチャートを有効にする	*false*	No
additionalCharts.opentelen collector.install	opentelemetry-collector サブチャートを有効にする	*false*	No
additionalCharts.kube-state-metrics.install	kube-state-metrics サブチャートを有効にする	*false*	No
additionalCharts.prometheus-node-exporter.install	prometheus-node-exporter サブチャートを有効にする	*false*	No
opentelemetry-collector.image.repository	OTEL コレクターイメージリポジトリ	""	Yes (if OTEL enabled)
opentelemetry-collector.image.tag	OTEL コレクターイメージタグ	""	Yes (if OTEL enabled)

TLS 構成オプション

これらのオプションは、エージェントがハイブリッドマネージャー HM ビーコンサーバー証明書を信頼する方法を制御します。デフォルト `rootCAEnabled: true` では、エージェントは認証局 CA シークレットをマウントし、その証明書を使用して HM を検証します。

Parameter	Description	Default	Required
rootCAEnabled	*true* `の場合、*CA* シークレットをマウントし、エージェントの *`root_ca_path* `を設定します。 *CA* シークレットをマウントせずにシステム *CA* プールを介して *HM* を信頼するか、検証をスキップするには *`false* に設定します。	*true*	No
rootCASecretName	HM ルート CA `ca.crt`を保持するシークレットの名前。`rootCAEnabled`が `true`の場合に存在する必要があります。	*beacon-server-ca*	No
rootCA	PEM エンコードされたルート CA。設定すると、チャートは既存のものを必要とする代わりに、CA シークレットを作成します。	*""*	No
agentConfig.beaconServer.rootCAPa	CA 証明書がマウントおよび読み取られるポッド内のパス。`rootCAEnabled`が `true`の場合にのみ使用されます。	*/etc/ssl/certs/ca-certificates.crt*	No
agentConfig.beaconServer.tlsInsecu	`true`の場合、エージェントは HM ビーコンサーバー証明書の TLS 検証をスキップします。通常、テストのために `rootCAEnabled: false`と組み合わせます。	*false*	No
agentConfig.beaconServer.plaintext	`true`の場合、エージェントは TLS なしで HM に接続します。	*false*	No

注釈

rootCAEnabled: false の場合、名前空間とシークレット で説明している `beacon-server-ca` シークレットを作成する必要はありません。エージェントはシステム CA プールに依存するか、*tlsInsecureSkipVerify: true* の場合に検証をスキップし、CA ボリュームマウントはエージェント、OpenTelemetryOTEL コレクター、または CloudNativePG™サイドカーポッドに注入されません。 !!!

Monitoring with the Hybrid Manager console

Hybrid ManagerHM コンソールを使用して、データベースメトリックの包括的なセットにアクセスできます。可用性は、メトリックとクラスタータイプによって異なります。

- *Support scope* クラスターでサポートされているメトリックの種類について説明します。
- *Metrics latency* メトリックが更新される頻度について学びます。
- *2 つの異なるイベントを表示するときにイベントマーカーが重なる*イベントマーカーと、クラスターで監視されているデータに関する追加情報を提供する方法について学びます。

Metrics support

Hybrid ManagerHM は、幅広いメトリックを収集および表示して、データベースクラスターを包括的な可視性を提供します。

HM コンソールに表示できるメトリックの可用性は、メトリックとクラスタータイプによって異なります。

概要の概要

この概要には、データベースクラスタータイプごとに使用可能なメトリックが示されています。

	ホストメトリック	*Postgres* メトリック	クラスターメトリック	クエリメトリック	*GPU Recommendations for Default NIM Models*	ログと可観測性
HM-managed cluster	Yes	Yes	Yes	Yes	Yes	Yes
External: **Self-managed cluster**	Yes	Yes	Yes	Yes	Yes	Yes
External: **CNPG database cluster**	Yes	Yes	Yes	Yes	Yes	Yes
External: **AWS RDS database**	Yes	partial support	No	No	No	No

- 現在、ハイブリッドマネージャーは、EDB CloudNativePG およびオープンソースの CloudNativePG クラスターのモニタリングをサポートしています。
- **モニタリングは、外部の自己管理 EDB Postgres 分散クラスターではサポートされていません。 .. ::** ## 詳細な概要

ホストメトリック

ホストメトリックには、データベースインスタンスをホストする基になるサーバーインフラストラクチャのリソース使用率が表示されます。それらは、**Estate** ビューと、クラスタービューの **Host** セクションの **Monitoring** タブに表示されます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Memory	Yes	Yes	Yes	Yes
CPU	Yes	Yes	Yes	Yes
Storage	Yes	Yes	Yes	Yes
Disk IOPS	Yes	Yes	Yes	Yes
Disk throughput	Yes	Yes	Yes	Yes
Network activity	Yes	Yes	Yes	Yes

重要

メモリ、CPU、およびストレージメトリックは、ホスティングノード上の Postgres プロセスの平均使用率を示します。 HM 管理対象クラスターでシステム全体の使用状況を監視するには、 *Agent Factory* メトリックの *Grafana* ダッシュボードの設定 を使用します。システム全体の監視は、外部クラスターで現在サポートされていません。

Postgres メトリック

Postgres メトリックは、Postgres データベースサーバーに固有のパフォーマンスと操作統計を示します。それらは **Estate** ビューとクラスタービューの ***Monitoring*** タブに表示されます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Connections				
Connections	Yes	Yes	Yes	Yes
AAS by Wait Type	Yes	Yes	Yes	No
Nr. of blocked backends	Yes	Yes	Yes	No
Transactions				
Tuples in	Yes	Yes	Yes	Yes
Tuples out	Yes	Yes	Yes	Yes
Transactions rate	Yes	Yes	Yes	Yes
Buffer cache hit ration	Yes	Yes	Yes	Yes
Longest running transaction	Yes	Yes	Yes	No
Queries				
Query latency	Yes	Yes	Yes	No
Query rate	Yes	Yes	Yes	No
Storage				
Database size	Yes	Yes	Yes	No

次のページに続く

表 80 – 前のページからの続き

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Disk usage	Yes	Yes	Yes	No
WAL size	Yes	Yes	Yes	Yes
WAL usage	Yes	Yes	Yes	No
Live/Dead tuples	Yes	Yes	Yes	No
Index/table/temp estimated size	Yes	Yes	Yes	Yes
Internals				
Time since last autovacuum	Yes	Yes	Yes	Yes
Autovacuum Stats	Yes	Yes	Yes	Yes
Time since last checkpoint	Yes	Yes	Yes	Yes
Checkpoint stats	Yes	Yes	Yes	Yes
Time since last successful WAL archive	Yes	Yes	Yes	No
WAL archiving stats	Yes	Yes	Yes	Yes

クラスターメトリック

クラスターメトリックは、ホストおよび postgres メトリックであり、マルチプルのノード HA、AHA、および DHA トポロジにまたがるデータベース展開用に個別および集計されたものです。それらは、**Estate** ビューと、クラスタービューの ***Monitoring*** タブの **Topology** および **Replication** セクションに表示されます。クラスターメトリックは、単一ノード展開の場合には表示されません。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
RAFT status (AHA, and DHA)	Yes	No	No	No
Replication slot status	Yes	Yes	Yes	No
Clock skew	Yes	Yes	Yes	No
Node status	Yes	Yes	Yes	No
Transaction rate	Yes	Yes	Yes	No
Connections (total)	Yes	Yes	Yes	No
Connections (per node)	Yes	Yes	Yes	No
Memory (per node)	Yes	Yes	Yes	No
Storage (per node)	Yes	Yes	Yes	No
WAL size (per node)	Yes	Yes	Yes	No
Replication status	Yes	Yes	Yes	No

クエリメトリック

クエリメトリックは、実行されているデータベース Postgres クエリのパフォーマンスと特性に関する洞察を提供します。それらは、クラスタービューの *Query diagnostics* タブで確認できます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Query latency	Yes	Yes	Yes	No
Nr. of queries executed	Yes	Yes	Yes	No
Query latency histogram	Yes	Yes	Yes	No
Nr. of active sessions	Yes	Yes	Yes	No

推奨事項

レコメンデーションは、観察されたメトリックに基づいて、データベース環境を改善するためのアクションと最適化を提案しますインデックスの改善など。それらは、クラスタービューの**推奨事項** タブで確認できます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Indexes	Yes	Yes	Yes	No
Statistics	Yes	Yes	Yes	No
Configuration	Yes	Yes	Yes	No
Security	Yes	Yes	Yes	No
Workload	No	No	No	No

ログ

Logs は、トラブルシューティングと監査のためにデータベースシステムのさまざまなコンポーネントからの生および処理されたログデータを表示します。それらは、クラスタービューの *Logs* タブで確認できます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Logs	Yes	Yes	Yes	No

Metrics latency

Hybrid ManagerHM は、広範囲の可観測性データを収集および表示し、データベースクラスターの包括的な可視性を提供します。

これらのメトリックの更新間隔は、メトリックとデータベースタイプによって異なります。この違いは、データの収集と統合のさまざまな方法によるものです。

データレイテンシーの理解

HM コンソールでデータベースメトリックを表示すると、表示されるデータは 2 段階のプロセスの結果であり、それぞれに独自の更新間隔があります。両方の手順を理解することは、メトリックがどの程度最新であるかを理解するのに役立ちます。

バックエンドの更新これは最初のステップであり、HM の内部サービスがデータベースクラスターから更新されたメトリックとデータをアクティブに収集します。クラスターが HM で管理されているか外部データベースで管理されているかにかかわらず、バックエンドシステムは常に最新の情報を収集しています。このデータは、HM の中央データストアに保存されます。

HM コンソールの更新これは 2 番目のステップであり、HM コンソールが中央のデータストアを呼び出して、最後に収集されたデータを取得して表示します。

HM コンソールに表示されるデータの実際のレイテンシーは、これら 2 つの更新間隔の組み合わせです。画面上のデータは、これらの 2 つの間隔の合計より古くない情報を反映しています。多くの場合、それははるかに新鮮であり、場合によってはわずか数秒前です。

リフレッシュ間隔

バックエンドの更新 HM の内部サービスは次のようにデータストアを更新します。間隔は、データベースクラスタータイプとメトリックコンテキストによって異なります。

	HM-managed	External: self-managed	External: CSP-managed	External: EDB CNPG
Estate page ☒ Postgres table	20s	5m	5m	5m
Cluster view ☒ Monitoring tab	15s	15s	30s	15s
Cluster view ☒ Query Diagnostics tab	1m	1m	na	1m
Cluster view ☒ Recommendations tab	30m	30m	na	30m
Cluster view ☒ Logs tab	1s	1s	na	1s

これらの間隔は、自己管理および CSP 管理のクラスターで構成できます。自己管理クラスターのデータ収集の頻度をカスタマイズするには、*EDB Postgres AI agent configuration YAML* で間隔を定義します。 CSP 管理対象クラスターの場合、*beacon-csp-credentials* を介してサポートされているメトリックの間隔を定義します。

HM コンソールの更新 HM コンソールは、次の頻度ですべてのクラスタータイプのデータを更新します。

	Console refresh interval
Estate page ☒ Postgres table	1m
Cluster view ☒ Monitoring tab	30s
Cluster view ☒ Query Diagnostics tab	Manual browser refresh
Cluster view ☒ Recommendations tab	Manual browser refresh
Cluster view ☒ Logs tab	10s

注釈

データは、ブラウザページを明示的にリロードした場合にのみ更新されます。

Event markers

イベントマーカーは、アクティビティログイベントをハイブリッドマネージャー HM の時系列チャートに視覚的なインジケーターとして表示します。これらは、メトリックの変更を特定のクラスターイベントと相関させるために、または分析する期間を選択するときの境界として使用します。

マーカータイプ

イベントマーカーは、イベントのタイムスタンプに垂直線として表示されます。構成の変更やアラートなど、単一の瞬間をマークするために使用します。

イベントマーカーの表示

イベントマーカーは、 **Monitoring** タブのモニタリングチャートに表示されます。クラスターまたはノードレベルで表示できます。

イベントマーカーを表示するには

1. HM コンソールで、プロジェクトを選択します。
2. **クラスター> Postgres クラスター** を選択します。
3. **Monitoring** タブを選択します。
4. **All Events** フィルタから、表示するイベントタイプを選択します。

マーカーにカーソルを合わせると、そのメタデータ名前、タイプ、説明が表示されます。

マーカーの切り替え

All Events フィルターを使用して、セクション、チャート、またはマーカータイプごとにマーカーを表示または非表示にします。マーカーの可視性は、タイプごとに独立して制御できます。

注釈

アラートはイベントマーカーとして表示されますが、アクティビティログには記録されません。アラートマーカーは、アラートの開始時間のみを反映します。根本的な問題が解決されると、マーカーは消えます。

時間区切り文字としてマーカーを使用する

日付と時刻ピッカーの代わりに、イベントマーカーを使用して、監視チャートに表示される時間範囲を定義できます。時間範囲の開始または終了としてマーカーを使用し、いずれかの側の範囲を拡張するオプションの時間バッファを使用します。一方のマーカーと他方の日付時刻をペアにすることもできます。

イベントマーカーを時間区切り文字として使用するには

1. モニタリングツールバーで、 **Start** または **End** ピッカーを選択します。
2. **Event** タブを選択します。
3. リストからイベントを検索または選択します。
4. (オプション) 他のピッカーでも繰り返すか、 **日付**タブを使用して日付時刻を手動で入力します。
5. (オプション) **タイムバッファ**を選択し、期間を選択していずれかの側の範囲を拡張します。
6. **Apply** を選択します。

システム生成のアノテーションの表示

HM は、特定のイベントの注釈を自動的に生成するため、手動アクションなしで監視チャートに表示されます。これらのイベントには、クラスター構成の更新が含まれます。ユーザー、自動スケーリング、またはエージェントオートメーションによってトリガーされるかどうか。

カスタムイベントマーカーの作成

POST 要求を/api/v1/projects/<project_id>/custom-event に送信することにより、任意の時点でカスタムイベントマーカーを作成できます。

```
{
  "message": "The text to attach to your event",
  "clusterId": "<cluster_id>"
}
```

注釈

clusterId フィールドはオプショナル。これを含める場合、イベントはアクティビティログに、およびそのクラスターのチャートに注釈として表示されます。省略すると、イベントはアクティビティログにのみ表示されます。

Grafana

Grafana を使用すると、ハイブリッドマネージャー HM 展開、HM 管理対象データベースクラスター、および HM に接続されているその他の外部クラスターに関するメトリックを視覚化および分析できます。パフォーマンスの監視、問題のトラブルシューティング、およびリソースの使用率に関する洞察を得ることができます。

HM 管理対象クラスターの場合、Grafana はすぐにメトリックを収集します。外部クラスターの場合、Monitoring external databases で説明しているように、最初に EDB Postgres AI エージェントをインストールする

必要があります。

Grafana のサポート

Grafana ダッシュボードは次のようにサポートされています。

- HM 管理クラスター すぐに使える完全なサポート。
- 自己管理クラスター クラスターが EDB Postgres AI エージェントを介して接続された後の部分的なサポート。
- CNP クラスター クラスターが EDB Postgres AI エージェントを介して接続された後の部分的なサポート。
- RDS クラスタークラスターが Kubernetes シークレットを介して接続された後の部分的なサポート。

Grafana へのアクセス

Launchpad を介してビルトイン Grafana インスタンスにアクセスできます。

結果のフィルタリング

HM によって管理されるすべてのクラスターには ID があり、名前空間とそのポッドまで追跡できます。クラスター ID 名前空間に相当するまたはポッド名で Grafana のメトリックをフィルタリングおよび分析し、トラブルシューティングとリソースプランニング操作を実行できます。

Kubernetes インフラストラクチャがクラスターにどのように関連するかを理解するには、*Tracing clusters back to their Kubernetes resources* を参照してください。より具体的には

- *ID and namespace for a cluster* の取得方法を理解します。
- クラスター/名前空間で **ポッドチェックが失敗する** を認識する方法を理解します。

関連トピック

- *Grafana* **ダッシュボードを使用する**
- Grafana's Explore option

Using Grafana dashboards

Hybrid ManagerHM に同梱されている Grafana サービスには、さまざまな HM リソースをモニタするように設計されたデフォルトのダッシュボードが付属しています。または、独自のカスタムダッシュボードを作成して、組織の要件を満たす特定のデータを表示します。

デフォルトの Grafana ダッシュボード

HM に同梱されている Grafana サービスには、次のカテゴリにグループ化されたデフォルトのダッシュボードが付属しています。

- *Kubernetes dashboards*
- *Loki dashboards for logging monitoring*

- *Postgres dashboards*
- *Telemetry dashboards*
- *Other dashboards*

> **注釈**
>
> ダッシュボードの説明では、HM の Grafana に含まれるデフォルトのダッシュボードの一般化および例示的な概要を提供します。特定のメトリックの可用性は、環境によって異なります。

カスタムダッシュボード

Grafana のドキュメント create your own dashboard を参照してください。

> **警告**
>
> カスタマイズされた Grafana ダッシュボードのバックアップがあり、バージョン管理リポジトリに保存されていることを確認します。カスタマイズされた Grafana ダッシュボードは、HM アップグレード、ノードの Kubernetes バージョンのアップグレード、または一般的なノードの障害など、Grafana インスタンスが更新または再展開されるたびに削除されます。

ダッシュボードの作成、使用、および管理

ダッシュボードの作成、使用、および管理の方法については、Grafana documentation を参照してください。ここにいくつかの役立つリンクを示します。

-

Create a dashboard 。

-

Import dashboards 。

-

Modify dashboard settings 。

- Manage dashboards 。

-

Share dashboards 。

- Use Explore データの照会、収集、および分析。
- Use Alerting 特定のイベントを監視およびフラグを立てるようにシステムを設定します。

Kubernetes dashboards

これらのダッシュボードは、ハイブリッドマネージャー HM を強化する基になる Kubernetes インフラストラクチャの状態とパフォーマンスに関する洞察を提供します。

これらは、Grafana の公式ライブラリからの Kubernetes mixins と呼ばれる事前に構築されたダッシュボードです。

Kubernetes / API サーバー

> **注釈**
>
> Grafana は API サーバーへのノードレベルのアクセスを必要とするため、オンプレミスの HM 展開のデータのみが表示されます。

説明クラスターとのすべての相互作用のための重要なコンポーネントである Kubernetes API サーバーのパフォーマンスと可用性を監視します。

メトリックこれらのメトリックには、読み取りおよび書き込みの可用性、ワークキューのレイテンシー、読み取りおよび書き込みエラー、リソース使用率などが含まれます。

使用法これを使用して、API レイテンシー問題または可用性問題を特定します。

Kubernetes /コンピューティングリソース *

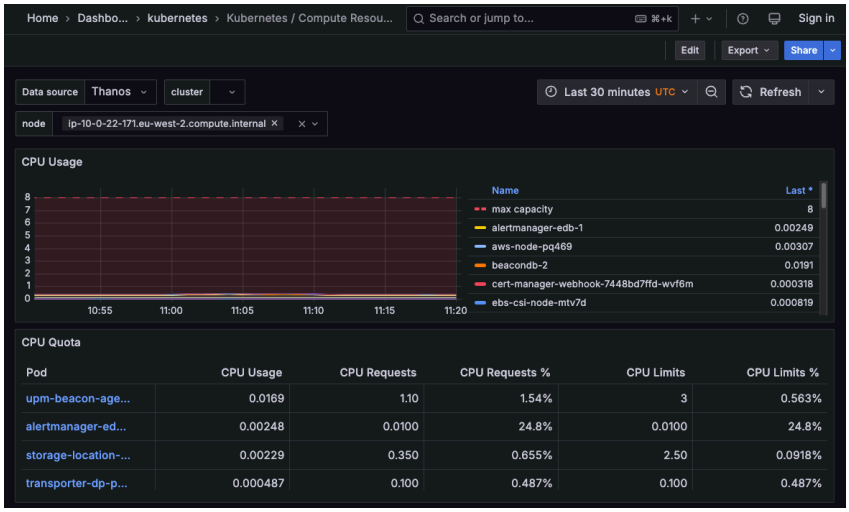


図 13: Compute Resources by Node and Pods

コンピューティングリソースは、さまざまなビューを提供する複数のダッシュボードに表示されます。**クラスター** 、 **名前空間ポッド** 、 **名前空間ワークロード** 、**ノードポッド** 、 **ポッド** 、または **ワークロード** 。

- **Cluster** には、Kubernetes クラスター全体のリソース使用率と、名前空間ごとの使用率が表示されます。
- **Namespace (Pods)** は、ポッド/サービスの消費に焦点を当てて、Kubernetes 名前空間でフィルタリングできるリソースの使用率を表示します。
- **Namespace (Workloads)** は、個々のワークロードの消費に焦点を当てて、Kubernetes 名前空間および

ワークロードタイプ展開、デーモンセット、ステートフルセットなどによってフィルタリングできるリソース使用率を表示します。

- **Node (Pods)** は、個々の Kubernetes ノードでフィルタリングできるリソースの使用率を示し、それらで実行されているポッドごとに内訳。
- **Pod** は、特定の名前空間の個々の Kubernetes ポッドの詳細なリソース使用量を提供します。
- **Workload** は、特定の名前空間の特定の Kubernetes ワークロード展開ワークロードなどの詳細なリソース使用量を示します。

説明 Kubernetes クラスターレベルでのリソース使用率を表示します。各サブダッシュボードは異なる方法でデータを表示し、さまざまな Kubernetes レベルに焦点を当て、さまざまなフィルタリングオプションを提供します。

メトリックメトリックには、CPU、メモリ、帯域幅、パケットレート、IOPS、スループットなどが含まれます。

使用法 これらのダッシュボードは次の目的で使用します。

- すべてのワークロードのリソース消費を理解する
- リソースを大量に消費するアプリケーションまたはチームを特定する
- アプリケーション内の特定のワークロード、およびノード内の特定のポッドのリソースフットプリントを理解する
- 特定のアプリケーションインスタンスのパフォーマンス問題のトラブルシューティング
- 特定のアプリケーションコンポーネントのパフォーマンスを監視する

Kubernetes / Kubelet

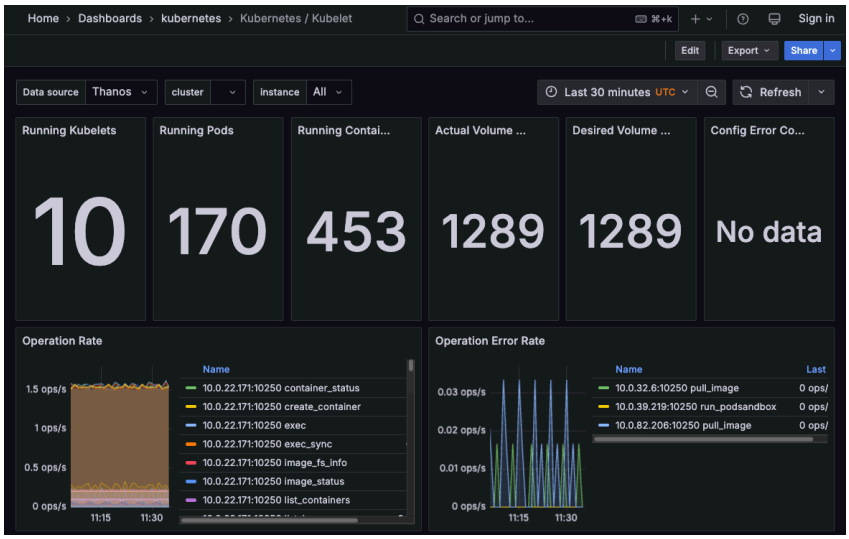


図 14: Kubelet dashboard

説明コンテナの管理を担当する、各 Kubernetes ノードで実行されている Kubelet サービスの状態とパフォーマンスを監視します。すべての kubelet の集計パフォーマンスを確認し、kubelet インスタンスでフィルタリングし、特定の kubelet 操作にドリルダウンできます。

メトリックメトリックには、kubelet 数、ポッド数、コンテナー数、操作率、操作期間、ポッドライフサイクル操作率、操作エラー率などが含まれます。

使用法これを使用して、ノードレベルのエージェントアクティビティ、操作動作、および遅延を特定し、特定のノードのパフォーマンス問題のトラブルシューティングを行います。

Kubernetes / ネットワーキング *

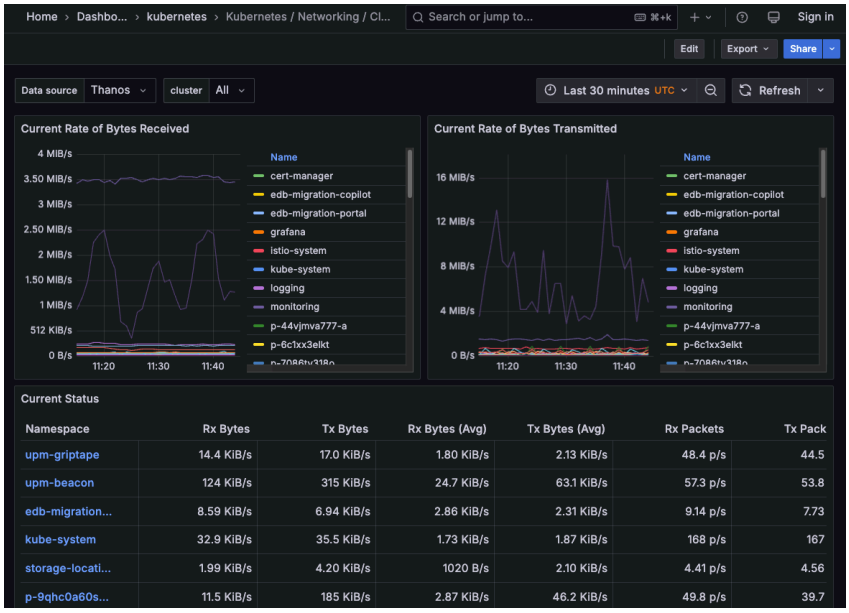


図 15: Networking overall cluster dashboard

ネットワークリソースは、さまざまなビューを提供する複数のダッシュボードに表示されます。

- **Cluster** には、Kubernetes クラスター全体および名前空間ごとのネットワーク使用率が表示されます。
- **Namespace (Pods)** は、Kubernetes 名前空間でフィルタリングできるネットワークトラフィックメトリックを、その名前空間のポッドに焦点を当てて表示します。
- **Namespace (Workloads)** は、個々のワークロードトラフィックに焦点を当てて、Kubernetes 名前空間およびワークロードタイプ展開、デーモンセット、ステートフルセットなどによってフィルタリングできるネットワークトラフィックメトリックを示します。
- **Pod** は、特定の名前空間内の個々の Kubernetes ポッドの詳細なネットワークトラフィックメトリックを提供します。
- **Workload** は、特定の Kubernetes ワークロード展開ワークロードなどの詳細なネットワークトラフィックメトリックを表示します。

説明 Kubernetes クラスターレベルでのネットワーク使用状況を表示します。各サブダッシュボードは異なる方法でデータを表示し、さまざまな Kubernetes レベルに焦点を当て、さまざまなフィルタリングオプションを提供します。

メトリックメトリックには、送受信されたバイト数、送受信されたパケットのレート、送受信帯域幅などが含まれます。

使用法 これらのダッシュボードは次の目的で使用します。

- 全体的なネットワーク通信パターン、トラフィック、スループット、および潜在的なボトルネックを理解する
- 特定のアプリケーションのネットワークパターンを分析する
- アプリケーション内の特定のワークロードのリソースフットプリントを理解する
- アプリケーション内の特定のポッドとのネットワーク接続問題のトラブルシューティング
- 特定のアプリケーションコンポーネントのネットワークパフォーマンスを監視する

Kubernetes / 永続ボリューム

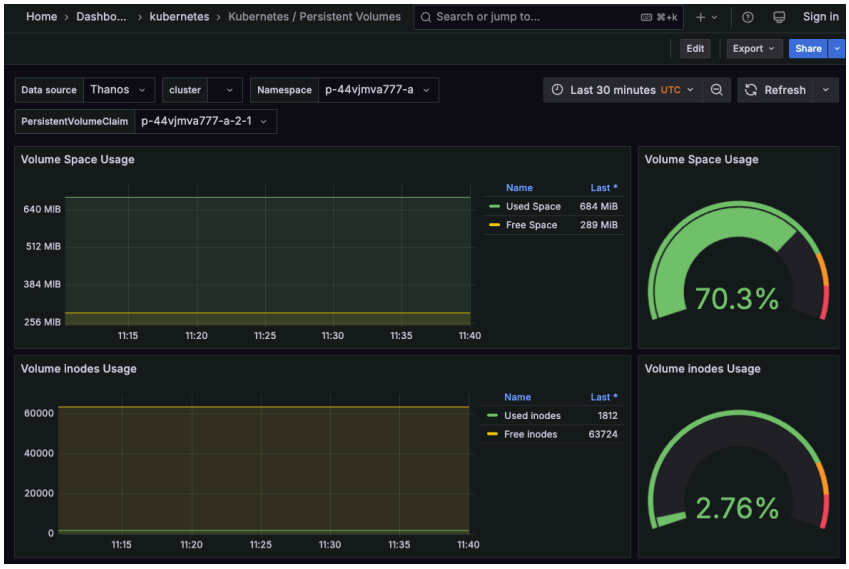


図 16: Persistent Volumes dashboard

説明 Kubernetes クラスター内の永続ボリューム PVs の使用状況とパフォーマンスを監視します。名前空間および永続ボリューム要求ごとに結果をフィルタリングできます

PVCs 。

メトリック メトリックには、空き領域と使用済み領域が含まれます。

使用法これを使用して、ストレージのボトルネックと容量の問題を特定し、ストレージ拡張の計画を立て、ストレージコストを監視します。

Kubernetes / StatefulSets

説明 ステートフルアプリケーションを管理する Kubernetes StatefulSets の特定のモニタリングを提供します。名前空間=アプリケーションとステートフルセット名で結果をフィルタリングできます。

メトリックメトリックには、CPU、メモリ、ネットワークの消費量、およびレプリカが含まれます。

使用法これを使用して、ステートフルアプリケーションの安定性、パフォーマンス、およびリソース消費をモニタし、ステートフルアプリケーションが目的のレプリカを満たしているかどうかを追跡します。

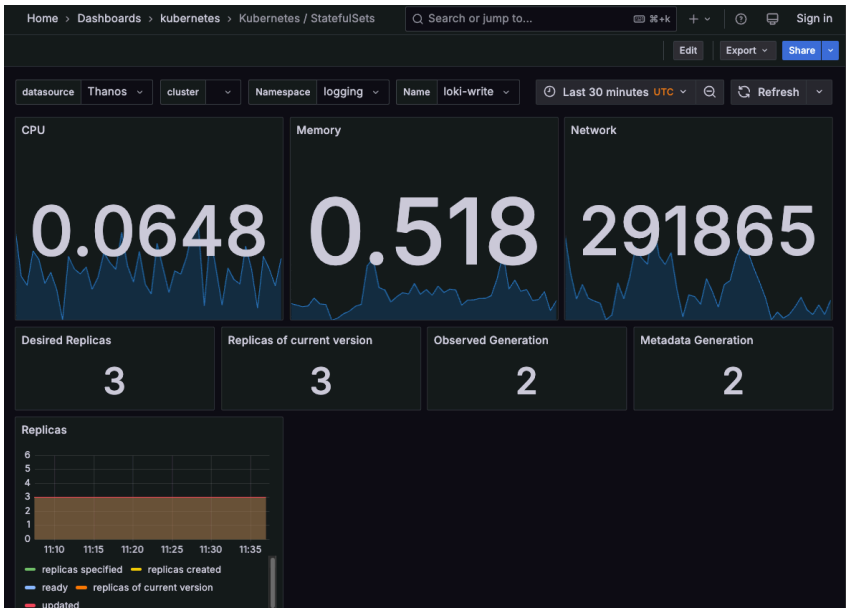


図 17: StatefulSets dashboard

Kubernetes CoreDNS

図 18: CoreDNS dashboard

説明 CoreDNS サービスのパフォーマンスと状態を監視します。これは、Kubernetes クラスターで DNS 解決を提供し、ポッドとサービス内の通信を有効にします。

メトリックメトリックには、プロトコル別の DNS 要求レート、クエリタイプ、DNS ゾーン、サイズ、応答コード、応答期間、応答サイズなどが含まれます。

使用法これを使用して、CoreDNS コンポーネントの信頼性と効率をモニタし、サービス間の検出と通信の問題を特定します。

Loki dashboards for logging monitoring

これらのダッシュボードは、統合された Loki ログシステムに関連しており、ログを視覚化および分析できます。Loki が操作を実行する場所であるため、視覚化は `logging` 名前空間のみに制限されます。

Loki ダッシュボードは、Grafana の公式ライブラリからの Loki mixins と呼ばれる事前構築されたダッシュボードです。

ロキ / チャンク

図 19: Loki Chunks dashboard

説明 Loki のログデータチャンク単一のログストリームに属するログエントリのグループのストレージと管理を監視します。

メトリックメトリックには、アクティブなチャンク、パーセンタイル使用率、チャンクごとのログエントリ、チャンクライフサイクルイベントフラッシュ操作、チャンクサイズ、チャンク年齢、チャンクごとのバイトなどが含まれます。

使用法潜在的なストレージの非効率を特定し、チャンクのライフサイクルパターンを理解し、チャンク管理とクエリパフォーマンスに関連する問題のトラブルシューティングを支援します。

Loki / 削除

説明保持ポリシーに基づいて、Loki のログ削除プロセスのパフォーマンスとステータスを追跡します。デフォルトの Loki 保持期間は 30 日です。

メトリック削除期間、削除エラー、削除されたチャンク数、削除の影響を受けるシリーズの数、および削除キューの長さ。

使用法保持ポリシーが正しく適用されていることを確認し、削除プロセスの効率を監視し、ログの削除を妨げているエラーを特定できます。これを使用して、ログが予想通りにプルーニングされていることを確認します。

Loki / Logs

説明 Loki によって取り込まれたログを表示およびフィルタリングできます。

メトリック ログ、取り込まれたログのレートなど。

使用法ログデータの視覚的な概要を提供し、傾向とエラーをすばやく識別できます。これは、リアルタイムログ分析とトラブルシューティングに使用します。

Loki / オペレーション

図 20: Loki Operational dashboard

説明 Loki システムの運用の状態とパフォーマンスに関する洞察を提供します。

メトリック 1 秒ごとのクエリーとプッシュ数、クエリーとプッシュのレイテンシー、テナントごとのログ行、テナントごとの MB 使用量、Loki コンポーネントの CPU 使用率インジェスター、ディストリビューター、クエリア、Loki コンポーネントのメモリ使用量、ストリーム、チャンク時間、その他。

使用法リソース消費と内部の状態を監視することにより、Loki ログシステムの安定性とパフォーマンスを保証します。Loki の潜在的なボトルネックまたはリソースの枯渇を特定します。

Loki / 読み取り

説明 Loki に対するログ読み取りクエリのパフォーマンスを監視します。

メトリック 1 秒ごとの Loki 読み取りクエリー、読み取りオペレーションごと、ポッドごとのレイテンシー。

使用法 ログクエリーに対する Loki の応答性を評価できます。

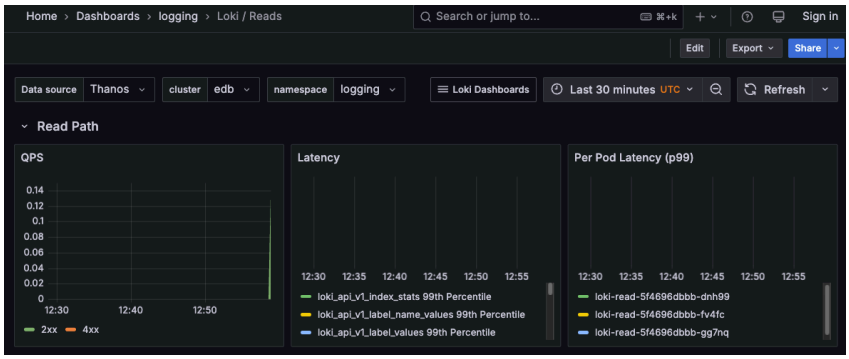


図 21: Loki Reads dashboard

Loki / Reads リソース

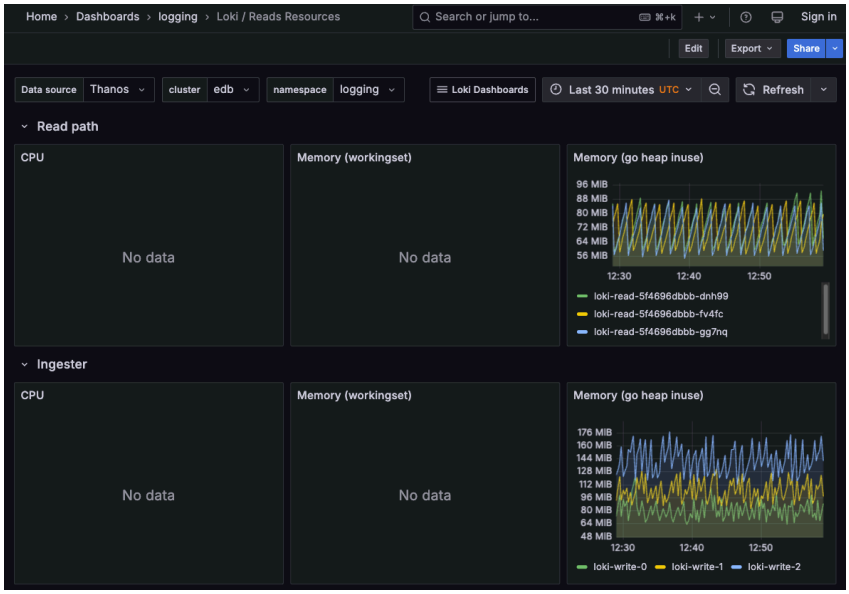


図 22: Loki Reads Resources dashboard

説明 Loki 読み取り操作によるリソース消費 CPU、メモリを表示します。

メトリック個々の Loki 読み取りポッドおよび Loki に取り込まれたポッドによって使用されているメモリ。

使用法ログ読み取りクエリのリソースオーバーヘッドを理解するのに役立ちます。

Loki / リテンション

説明 Loki のログ保持ポリシーとその適用に関する詳細なメトリックを提供します。

メトリック最後に成功した圧縮オペレーション、期間とステータス別の圧縮オペレーション、Loki バックエンドポッドが使用しているメモリ保持用。

使用法構成済みの保持ポリシーがアクティブであることを確認し、Loki のストレージ消費への影響を理解できます。

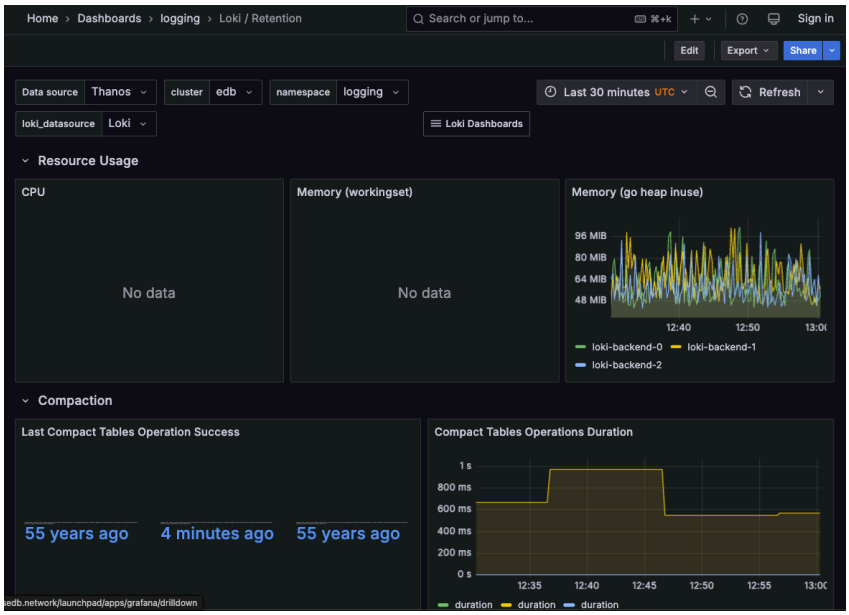


図 23: Loki Retention dashboard

Loki / Writes

図 24: Loki Writes dashboard

説明 Loki へのログ取り込み書き込みクエリのパフォーマンスを監視します。

メトリック 1 秒ごとの Loki 書き込みクエリー、書き込みオペレーションごと、ポッドごとのレイテンシー。

使用法遅い取り込み速度またはアプリケーションから Loki へのログ転送パイプラインでドロップされたログを特定するのに役立ちます。

Loki / Writes リソース

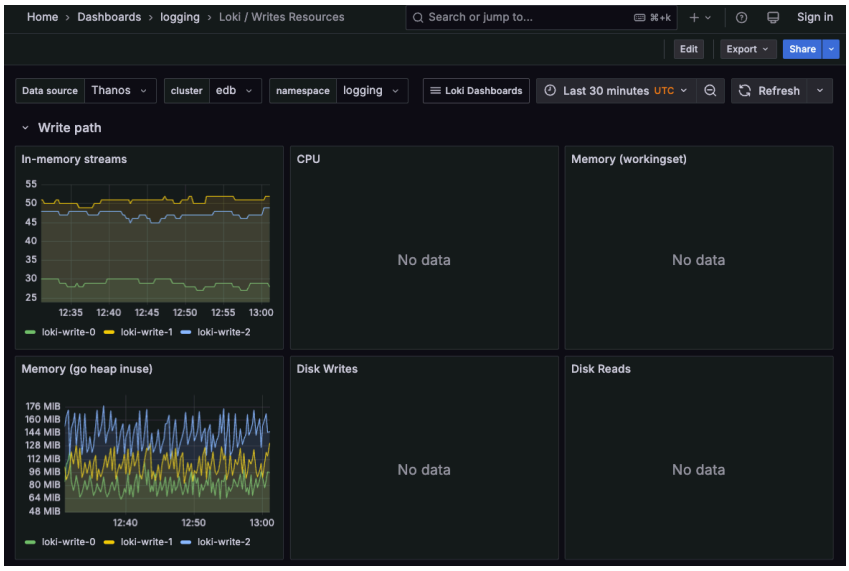


図 25: Loki Writes Resources dashboard

説明 Loki 書き込み操作によるリソース消費 CPU、メモリを表示します。

メトリック個々の Loki 書き込みポッドおよび Loki 取り込みポッドによって使用されているメモリ。

使用法ログ書き込みクエリのリソースオーバーヘッドを理解するのに役立ちます。

ロキデータ

図 26: Loki Data dashboard

説明これは、EDB によって設計されたダッシュボードで、Loki が管理しているログデータの量の高レベルのビューを提供します。

メトリック Loki の取り込みレートとバイト単位の合計保存チャンク。

使用法このダッシュボードは、キャパシティプランニング、データの増加の理解、およびロギング関連のストレージコストの管理に使用できます。

Postgres dashboards

これらのダッシュボードは、Hybrid ManagerHM で管理または監視する Postgres データベースのパフォーマンスと状態に関する洞察を提供するように EDB によって設計されました。

ハードウェア使用率

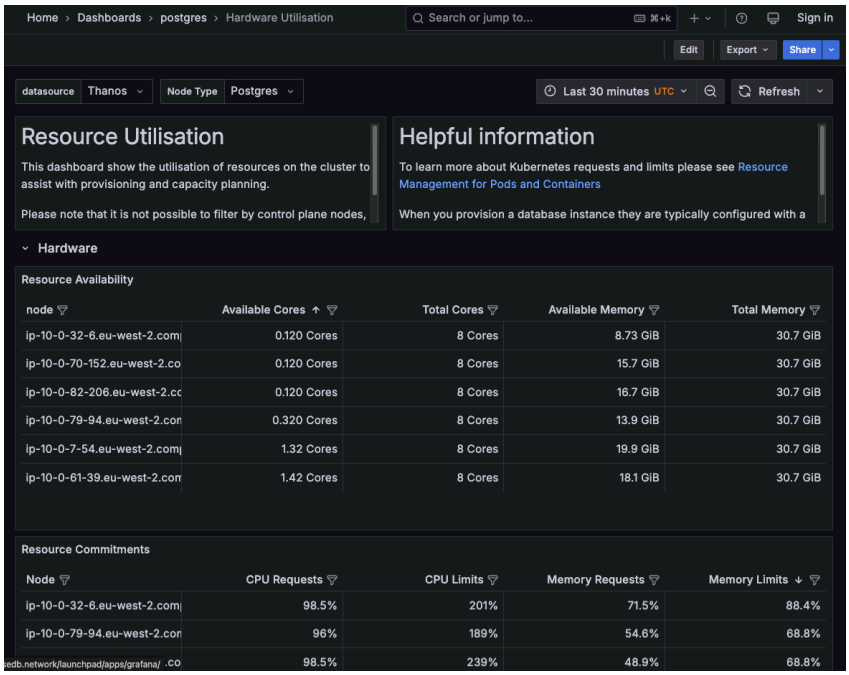


図 27: Postgres Hardware Utilisation dashboard

説明 Postgres インスタンスをホストするノードの基になるハードウェアリソース使用コア、メモリ、CPU を表示します。

メトリックノードごとの使用可能なコアと合計コア、ノードごとの使用可能なメモリと合計メモリ、ノードごとの CPU 要求と制限、ノードごとの Postgres ポッド数、ポッドステータスなど。

使用法これを使用して、キャパシティのボトルネックを特定し、オーバーサブスクライブしていないか、ノードがオーバースケジュールされているかどうかを確認します。

Kubernetes イベント

説明 Postgres のポッドと展開に関連する Kubernetes イベントを表示します。過去 1 時間のイベントが表示されます。したがって、ワークロードがスムーズに実行され、ポッドの復元に Kubernetes イベントが必要ない場合、指定された名前空間のイベントが表示されない可能性があります。

メトリック YAML 形式のイベントのタイプたとえば、ポッドの作成、削除、エラーなど。

使用法 Postgres インスタンスに影響を与える可能性のある Kubernetes 操作にコンテキストを提供します。失敗した展開、ポッドの再起動、またはデータベースの可用性に影響を与えるその他の Kubernetes レベル

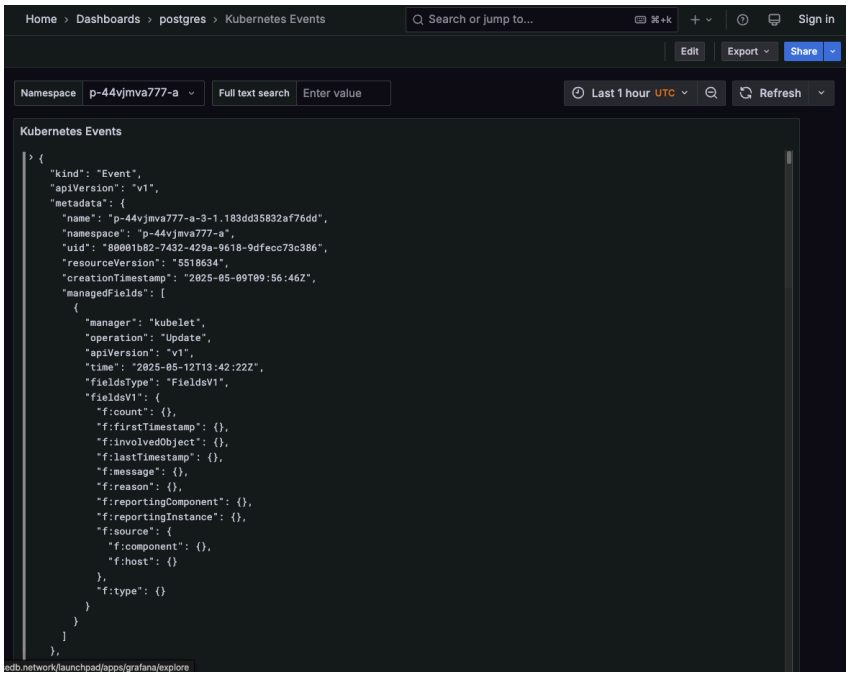


図 28: Postgres Kubernetes Events dashboard

の問題の診断に役立ちます。

PGD の概要

分散高可用性 DHA または PGD クラスターのみ。

説明レプリケーション機能に焦点を当てて、分散高可用性 PGD クラスターのパフォーマンスの概要を提供します。

メトリック PGD および BDR のバージョン、データノードのステータス、バージョン、Raft の状態とメトリック、BDR ワーカーの状態とメトリック、レプリケーションステータス、接続、スループットなど。

使用法クラスターの状態と同期ステータスを一元的にビューできます。遅延または異常なノードを特定し、分散セットアップの全体的なパフォーマンスを監視します。

Postgres エステート統計

説明 実行しているクラスターのフリート全体のビューを提供します。

メトリック実行中のすべてのクラスターのリストと数、その経過期間、および最新の復元スタンプ。

使用法 エステートの下のクラスターの高レベルの概要を提供します。

Postgres ログ

説明: Postgres サーバーログを表示できます。

メトリック テキスト形式のログ。

使用法 Postgres ログデータの視覚的な概要を提供し、傾向とエラーをすばやく識別できるようにします。これは、リアルタイムログ分析とトラブルシューティングに使用します。

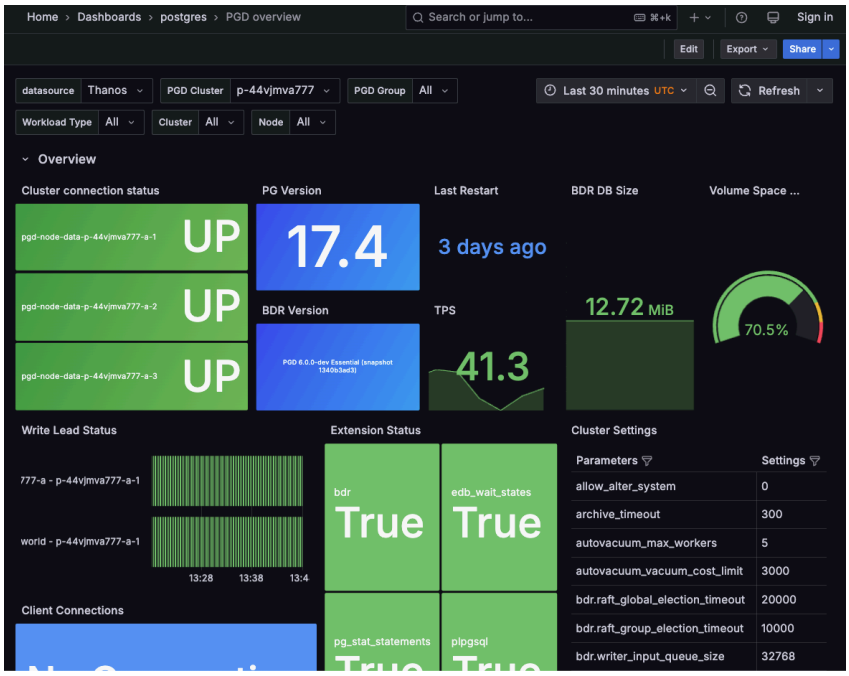


図 29: Postgres PGD Overview dashboard

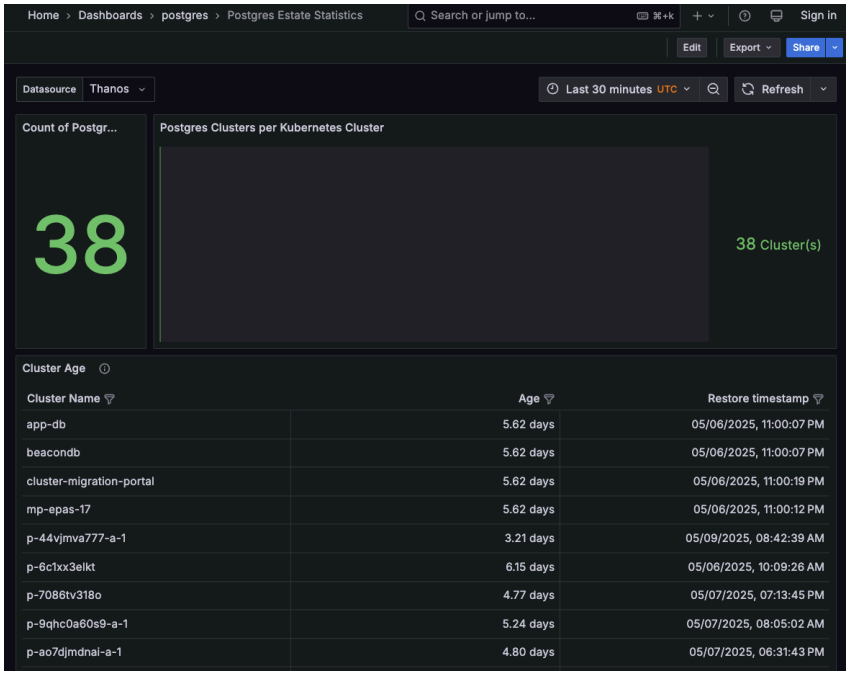


図 30: Postgres Estate Statistics dashboard

Postgres ステータス

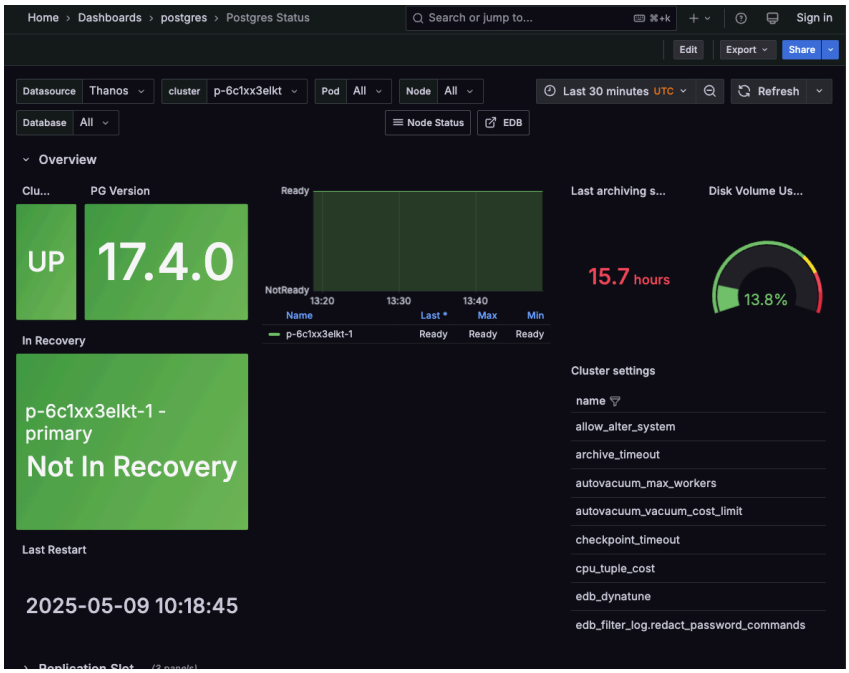


図 31: Postgres Status dashboard

すべてのクラスタータイプシングルノードおよびマルチプルノードの場合。

説明 個々の Postgres インスタンスのパフォーマンスの概要を提供します。

メトリッククラスターステータス、Postgres バージョン、ポッドステータス、レプリケーションステータスと遅延、接続、スループット、リソース使用率、WAL ファイリングステータスなど。

使用法個々の Postgres データベースの状態とアクティビティの一元的なビューを提供します。高負荷、接続の問題、レプリケーションラグ、または調査が必要なその他のパフォーマンス問題があるインスタンスを特定します。

Telemetry dashboards

これらのダッシュボードは、Hybrid ManagerHM プラットフォームとその基になるコンポーネントに関する洞察を提供します。

FluentBit

これは、Grafana の公式ライブラリからの事前ビルドされた Fluent Bit mixin です。

説明ノードからログを収集し、Loki に送信するために使用される FluentBit インスタンスに関するメトリックを提供し、Loki は Hybrid Manager に通知します。

メトリック入力および出力処理レート、入力および出力レコードレート、失敗およびエラー率など。

使用法 FluentBit でパフォーマンスのボトルネック、リソースの枯渇、または HM コンポーネントからログの収集と送信中にエラーが発生しているかどうかを特定するのに役立ちます。

Grafana の概要

説明さまざまなソースからメトリックを収集し、一箇所に表示するプラットフォームである Grafana インスタンスに関するメトリックを提供します。

メトリック アラートの発生、1 秒ごとの要求、レイテンシーなど。

使用法これらのダッシュボードの表示に使用している Grafana インスタンスの状態とパフォーマンスをモニタできます。Grafana が過負荷になっているか、問題が発生しているかどうかを特定します。

Memcached

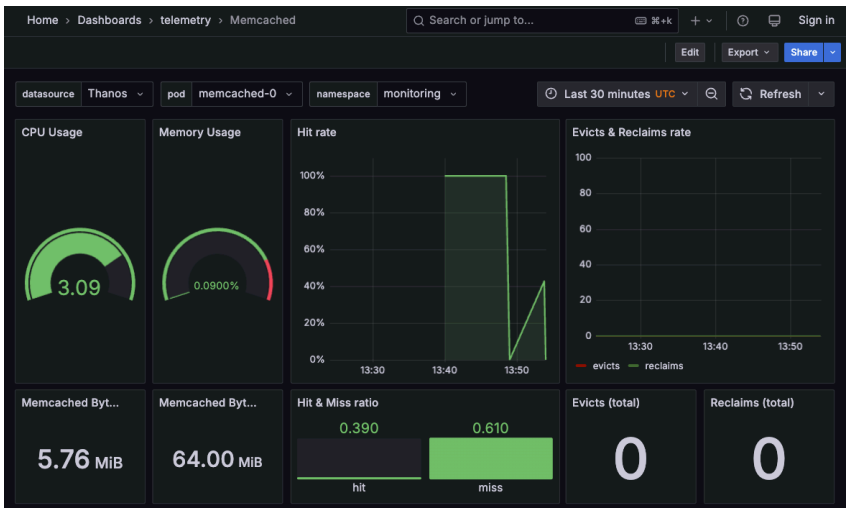


図 32: Telemetry Memcached dashboard

説明 HM インフラストラクチャの Memcached モニタリングサービスの状態とパフォーマンスを監視し、リソースの使用率、キャッシュ効率、およびエビクション率に関する洞察を提供します。

メトリック CPU とメモリの使用量、ヒット/ミス率、削除と再利用率、memcached バイト数など。

使用法このダッシュボードを使用して、Memcached インスタンスのリソース消費を理解し、キャッシュの有効性ヒット率を評価し、メモリプレッシャーのために項目がエビクトまたは再利用される頻度をモニタします。

ノードエクスポーター*

これらは、Grafana の公式ライブラリから事前にビルドされた node mixins です。

ノードエクスポーターメトリックは、さまざまなビュー **ノード** 、 **USE Method / Cluster** 、および **USE Method / Node** を提供する複数のダッシュボードに表示されます。

- **ノード**は、特定のノードの集計されたリソース使用率を提供します。CPU 使用率、負荷、メモリの種類などに分類されます。
- **USE Method / Cluster** は、Kubernetes クラスター全体の集計されたリソース使用率をノードごとに色付けで表示します。
- **USE Method / Node** は、ノードごとの合計リソース使用率と、不揮発性メモリデバイスのいくつかのメトリックを表示します。

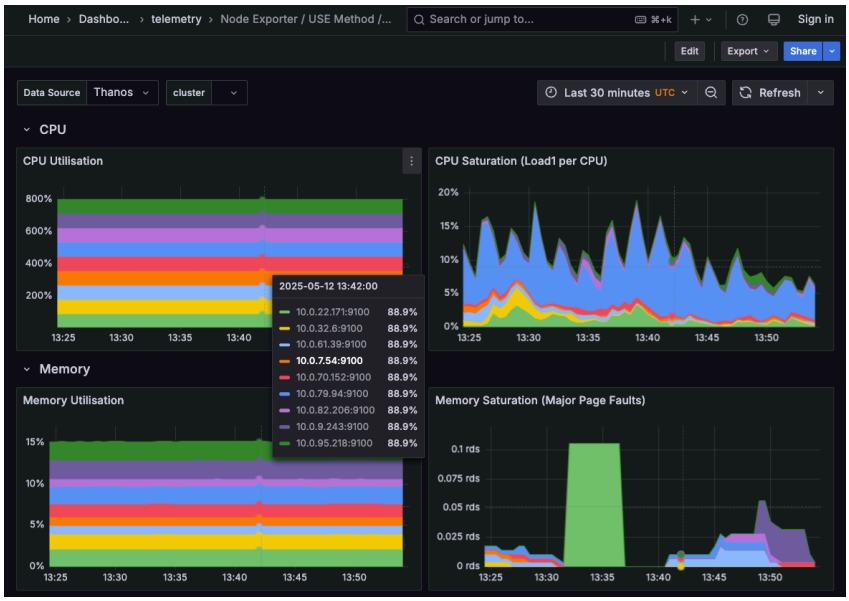


図 33: Telemetry Node Exporter dashboards

説明 HM コンポーネントを実行しているノードのハードウェアメトリックを表示します。

メトリック CPU 使用率、負荷平均、メモリ使用量、ディスク使用量など。

使用法 HM をホスティングするサーバーのハードウェアリソースのいくつかのビューを提供します。ノードレベルでリソースのボトルネックを特定するのに役立ちます。

Prometheus の概要

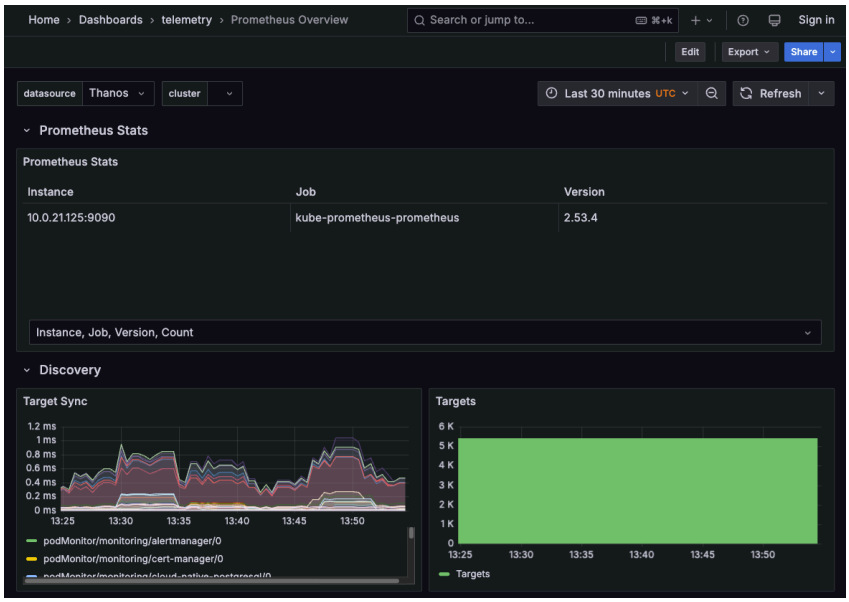


図 34: Telemetry Prometheus dashboard

これは、Grafana の公式ライブラリからの事前ビルドされた Prometheus mixin です。

説明 HM でメトリックを収集および出荷するために使用される Prometheus モニタリングシステムの状態とパフォーマンスを監視します。

メトリック Prometheus 統計、検出とターゲット、スクレイピング間隔と失敗、クエリレートなど。

使用法 Prometheus メトリック収集エンジンの状態を監視できます。Prometheus の問題により、監視データが不完全または遅延される可能性があります。

Thanos *

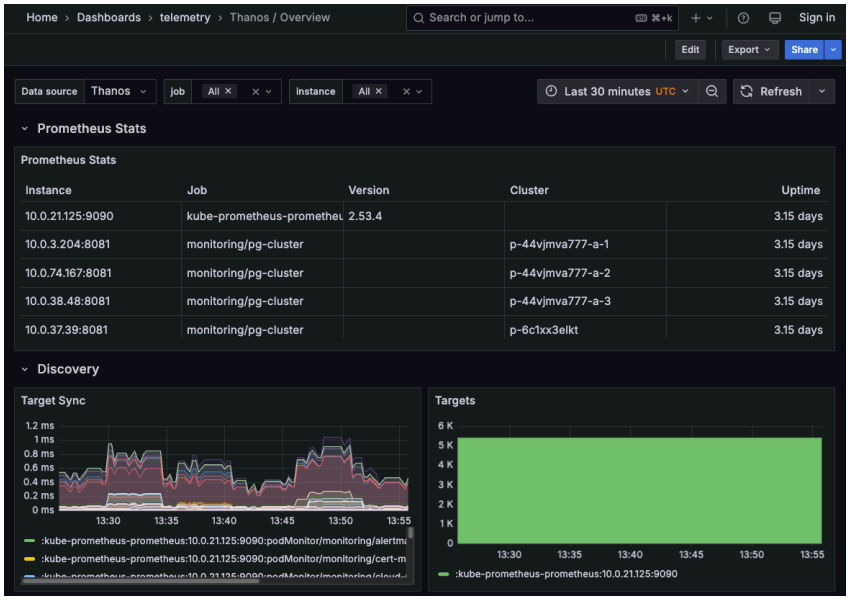


図 35: Telemetry Thanos dashboards

これらは、Grafana の公式ライブラリから事前にビルドされた Thanos mixins です。

Thanos メトリックは、 **Compact** 、 **Overview** 、 **Query** 、**Receive** 、および **Store** などのさまざまなビューを提供する複数のダッシュボードに表示されます。

- **Compact** は、Prometheus メトリックの長期保存とダウンサンプリングに使用されるコンポーネントである Thanos コンパクターをモニタします。
- **概要** は、HM での Thanos 展開の高レベルの概要を提供します。
- **Query** は Thanos クエリ API をモニタし、レートと期間メトリック、エラーなどを提供します。
- **Receive** は Thanos 着信要求をモニタし、レートと期間メトリック、エラーなどを提供します。
- **Store** は、Thanos ストレージジョブ、バケット、ブロック、キャッシュ操作、および関連するエラーをモニタします。

説明 Thanos の長期メトリックストレージシステムのパフォーマンスと状態を監視します。

メトリック Thanos ジョブの統計、スクレイピングインターバル期間、コンパクション、クエリー、着信要求とストアプロシージャー、プロシージャーのレイテンシー、エラー、CPU とメモリの使用量に関するメトリック。

使用法メトリックの長期ストレージが Thanos によって効率的に管理されていることを確認するのに役立ちます。コンパクターに問題があると、ストレージの非効率またはデータの損失が発生する可能性があります。Thanos のクエリと着信要求の処理が遅いと、Grafana ダッシュボードなどの応答性に影響を与える可能性があります。

Other dashboards

電力消費 電力使用量と二酸化炭素排出量

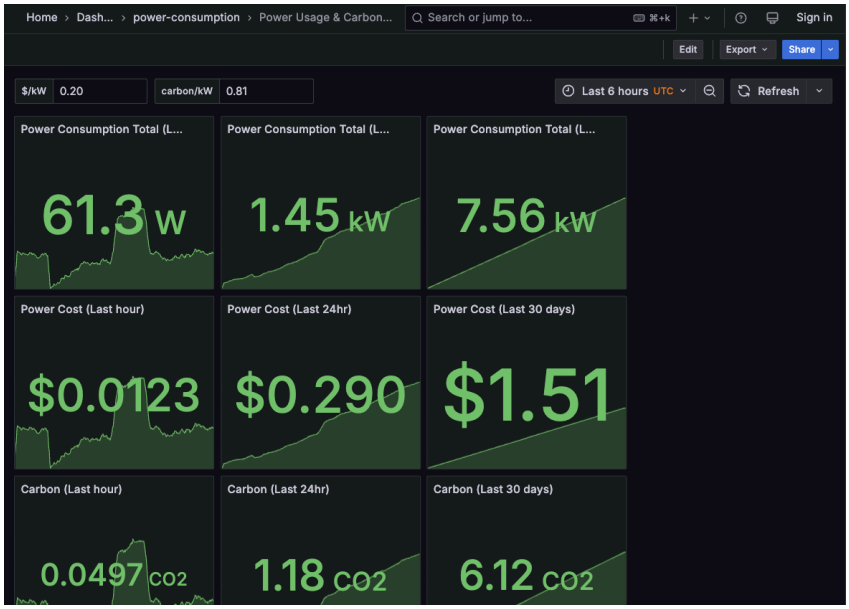


図 36: Power usage and carbon footprint dashboard

説明ハイブリッドマネージャー HM インフラのエネルギー消費と推定二酸化炭素フットプリントを監視します。これを使用して、環境への影響を追跡し、最適化する可能性があります。

このダッシュボードは、re:cinq の Cloud CPU Energy Consumption 計算方法と同様の手順に従って EDB によって設計され、ソースデータとして TEADS dataset から無料で利用可能な電力測定値を使用します。

メトリック過去 1 時間、24 時間、または 30 日間のグラフに分類された電力消費量ワットまたはキロワット単位、推定二酸化炭素排出量、および推定電力コスト。

使用法 HM インフラストラクチャの環境への影響を視覚化し、エネルギー使用の傾向を追跡し、コストと環境への影響を削減するために最適化する領域を特定できる可能性があります。

インフラストラクチャ:cert-manager

説明 TLS 証明書の管理と発行を自動化する Kubernetes アドオンである cert-manager の状態とパフォーマンスを監視します。これを使用して、HM での証明書管理が適切に機能することを確認します。

メトリック発行された証明書の数、証明書要求ステータス保留中、準備完了、失敗)、証明書の有効期限など。

使用法 HM の TLS 証明書が正しく管理され、それらに依存するアプリケーションが有効で最新の証明書を持つことを保証します。停止が発生する前に、証明書の発行または更新に関する潜在的な問題を特定します。

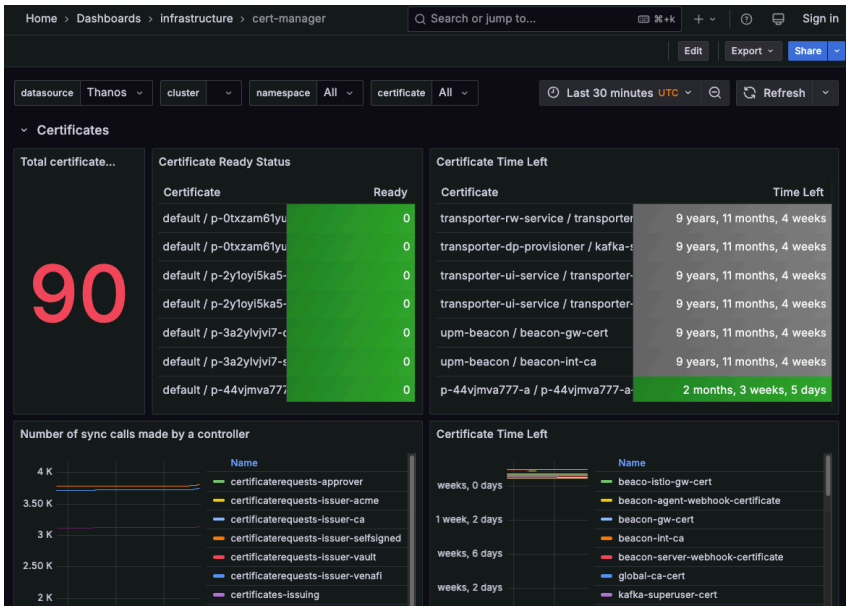


図 37: Cert-manager dashboard

インフラストラクチャ Velero の統計

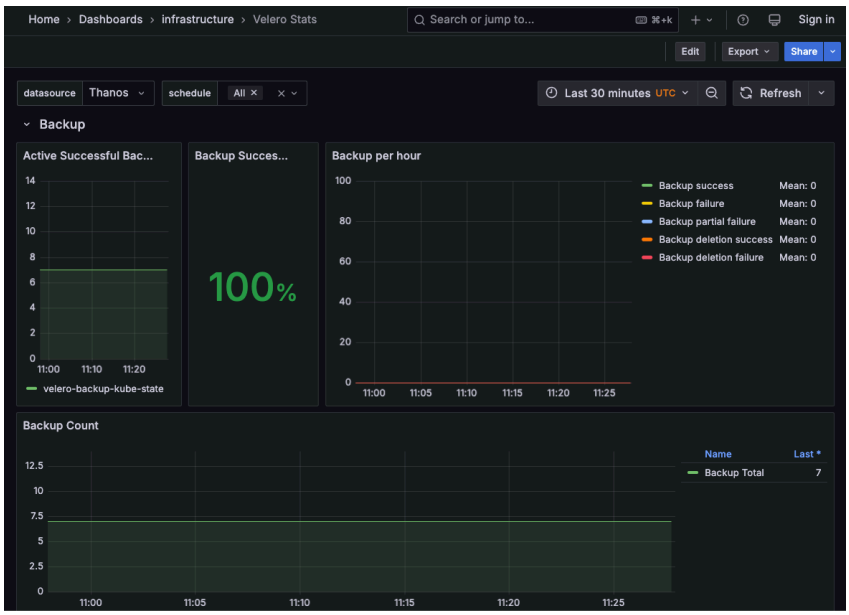


図 38: Velero dashboard

説明 Kubernetes リソースと永続ボリュームのバックアップおよび復元ツールである Velero の状態とパフォーマンスを監視します。これを使用して、HM でのバックアップのステータスと成功を追跡します。

メトリック作成されたバックアップの数、実行されたリストアの数、バックアップとリストア操作のステータス完了、失敗、進行中、バックアップサイズ、リストア期間、バックアップまたはリストア中に発生したエラー、スケジュールステータス。

使用法 HM でのバックアップおよび復元操作の可視性を提供します。バックアップの成功を確認し、復元プロセスを監視し、ディザスターリカバリー戦略の問題を特定できます。

Grafana’s Explore option

Grafana は、Loki によって収集されたログを照会できる **Explore** オプションを提供します。

特定のクラスターからログを照会するには

1. *cluster's ID and namespace* を取得します。 `cnp_instance_id` を選択します。
2. **Label filters** メニューで、 **cnp_instance_id** を選択します。
3. **Select value** メニューで、ポッドの ID を選択します。
4. そのインスタンスのログで特定のキーワードたとえば、「error」や「Exception」を検索するようにクエリーを作成します。

Troubleshooting in Grafana

Grafana ダッシュボードの `no healthy upstream` 警告は、サービスが目的のバックエンドに接続できないことを示しています。この問題は一時的なか、長期間持続する可能性があり、モニタリングおよびデバッグツールの使いやすさに影響を与えます。一般的な原因を理解することは、これらの問題の診断と解決に役立ちます。

潜在的な原因

ポッドのエビクションまたは移動

Karpenter のような動的なノードプロビジョニングとポッドスケジューリングツールは、ノード間でポッドを移動する場合があります。このプロセス中に、ポッドが一時的に使用できなくなる場合があり、再スケジュールされ、新しいノードで準備が整うまで、`no healthy upstream` エラーが発生します。

リソースのオーバースケジュールとメモリ不足 OOM エラー

サービス、特に複雑または大規模なクエリを処理するサービス Loki などは、大量のメモリを消費する可能性があります。通常、これらのツールはクエリを制約しません。彼らは、メモリ割り当てで成功しない可能性のあるクエリを特定してフラグを立てるのではなく、テラバイトのログまたはギガバイトのメトリックと OOM を返す可能性のあるクエリを実行しようとします。

クラスターノードがオーバーサブスクライブされている場合、またはポッドが適切なリソース制限で構成されていない場合、ホストのメモリ圧力が原因で Kubernetes はそれらを強制終了する場合があります。ポッドがすぐに回復した場合でも、OOM イベント中の短期間の使用不可により、`no healthy upstream` 警告がトリガーされる場合があります。

一時的なネットワークの問題または接続の終了

断続的なネットワーク問題、接続リセット、またはヘッダーを受信する前の突然の接続終了も `no healthy upstream` として発生する場合があります。

サービスの使用不可または構成の誤り

一時的な問題ではあまり一般的ではありませんが、 `no healthy upstream` メッセージは、バックエンドサービスが異常である、実行されていない、または誤って構成されていることを示し、Istio がトラフィックを正常にルーティングできないことを示している場合があります。

トラブルシューティングの手順

1. **クエリの最適化**これらの特定のバックエンドの OOM の最も一般的な直接の原因に対処します。Loki、Thanos、および Prometheus はクエリをガードレールしないため、これらのバックエンドが失敗し始めたときに最初に確認することは、広すぎるまたは複雑なクエリが OOM 問題を発生させているかどうかです。

時間ウィンドウを短縮するか、より少ないログメトリックのセットを選択するか、より具体的なフィルタリングストリームセレクター、ラインマッチング、JSON クエリー、正規表現式の回避を使用して、クエリーを絞り込んでみてください。詳細については、Query best practices を参照してください。

1. **リソースの使用率を監視する**監視ワークロード Loki、Thanos、Prometheus のリソース使用量を確認します。*resource utilization metrics for the affected pods* とその基になるノード CPU、メモリを確認します。これは、リソースのオーバースケジュールまたはキャパシティの不足が根本原因であるかどうかを特定するのに役立ちます。例、 *Loki dashboards for logging monitoring* を確認します。ポッドのリソース **推奨されるメモリ制限** や **リクエスト** など、構成済みの **リソースのコミットメント** を確認します。

リソースの使用率が高いように見え、OOM エラーが持続する場合は、ポッドのリソース要求/制限またはノードのキャパシティを調整することを検討します。ノードに十分なメモリ容量があるが、ポッドがアンダーサブスクライブされている場合、ポッドのメモリ制限を増やすことを検討します。ポッドがオーバーサブスクライブされ、ノードがサブスクライブされたポッドをホストするためにより多くのメモリ容量を必要とする場合、ノードのメモリを増やすことを検討します。

1. **Kubernetes イベントとログを確認します**ポッドの削除、スケジュール障害、またはノードの変更については、Kubernetes イベント Grafana's Explore option を確認します。ログは、ポッドの状態と特定の OOM イベントを確認するための良いソースです。これらは、どのポッドとプロセスに障害が発生しているかを理解するのに役立ちます。

2. **ネットワークの状態を確認します。**これはより一般的なチェックであり、OOM 問題の主な原因になる可能性は低いです。基になる *network infrastructure* の状態を確認します。

Enabling usage reporting in the HM server

Hybrid ManagerHM サーバーは、使用状況テレメトリを EDB に送信して、コアベースのライセンスと資格を自動化します。これにより、手動の監査または追跡を必要とせずに、環境がプロビジョニング制限に準拠した状態を維持できます。このレポートは、アクティブな監視に HM を使用していない場合でも、すべてのオンプレミスサーバーで機能します。

HM サーバーを使用した使用状況レポートはオプトインサービスであるため、デフォルトでは有効になっていません。データは、明示的に有効にした場合にのみ EDB に送信されます。有効にすると、HM サーバーは、デフォルトで 24 時間ごとに、HM サーバーが最初に実行されたときから新しいレポートを送信します。

どのような種類のデータが収集されますか?

使用状況データを送信するように構成されている場合、HM サーバーは、環境メタデータとリソースの使用状況を含む小さなレポートを生成します。これには、次のような情報が含まれます。

- オペレーティングシステムの詳細
- メモリと CPU の使用率
- インストールされたデータベースディストリビューションたとえば、PostgreSQL、 EDB Postgres Advanced Server、 EDB Postgres Extended Server とバージョン

> **重要**
>
> このレポートには、機密データまたはユーザーを識別できる情報は含まれません。

データはどこに送信されますか?

構成すると、レポートデータは EDB 管理エンドポイントである https://hm-usage.enterprisedb.com に送られます。

HM サーバーでの使用状況レポートを有効にするにはどうすればよいですか?

ハイブリッドマネージャーを使用している場合、HM サーバーの構成の一部としてレポートを有効にできます。*HM* サーバー使用状況レポートの有効化 を参照してください。

データベースサーバー PostgreSQL、EPAS、PGE のスタンドアロン展開があり、HM を使用したくない場合は、Enabling usage reporting for standalone databases を参照してください。

エアギャップ環境で動作している場合は、 EDB Postgres AI エージェントの構成ページ内で JSON ファイルとして使用レポートを作成し、**Tools** セクションの下の **PGAI Agent Reporting** を介してアップロードできます。 **EDB カスタマーポータル**の左側のペイン。

HM サーバー使用状況レポートの有効化

次の方法で HM の使用状況レポートを有効にできます。

オペレーターを介した HM サーバー使用状況レポートを有効にする

テレメトリとサーバーメトリック収集を有効にするには、特定の会社とインストールの詳細を使用して `HybridControlPlane` カスタムリソースを構成する必要があります。

要件

- オペレーターがインストールされているクラスターへのアクセス。
- 一意の企業コードは、管理者によって提供されます。

使用状況レポートを有効にする 演算子

1. HybridControlPlane リソースを編集のために開き、kubectl を使用して既存のコントロールプレーン構成を編集します。

```
kubectl edit hybridcontrolplane <resource-name> -n <namespace>
```

1. spec セクションの下で、globalParameters ブロックに会社コードが含まれていることを確認します。

```
spec:
  globalParameters:
    company_code: "YOUR_COMPANY_CODE"
```

1. enableEDBUsageReporting および hybridManagerInstallation フィールドを beaconServer セクションに追加します。

```
spec:
  beaconServer:
    enableEDBUsageReporting: true
    hybridManagerInstallation: <install-name>
```

1. 変更を保存します。オペレーターは、更新された仕様を自動的にリコンサイルします。HybridControlPlane が正常な状態になると、HM サーバーはテレメトリサービスの初期化を開始します。

重要

***2_blank_4_- enableEDBUsageReporting* が `false` に設定されている場合、使用データは収集されず、EDB.`1_blank_3_-` に送信されません。 *enableEDBUsageReporting* が true に設定されている場合、フィールド *company_code* および `hybridManagerInstallation` は必要です。いずれかのフィールドが欠落している場合、HM カスタムリソースは障害状態に移行し、使用状況レポートが有効になっているときにこれらのフィールドが必要であることを示すエラーを返します。 .. important::**
Helm チャートを介して HM サーバーの使用状況レポートを有効にする

Helm 展開中にテレメトリとサーバーメトリック収集を有効にするには、構成でグローバル企業 ID と Beacon Server ステータスを定義する必要があります。

要件

- 機能的な Helm 環境。
- 一意の企業コードは、管理者によって提供されます。

使用状況レポートを有効にする Helm

1. 特定の展開に使用される values.yaml 構成ファイルを見つけます。
2. parameters.global セクションの下に、会社コードを追加します。

```
parameters:
  global:
    company_code: "YOUR_COMPANY_CODE"
```

1. enableEDBUsageReporting を true に、hybridManagerInstallation を<install-name> に設定して、サーバーメトリック収集を有効にします。

```
beaconServer:
  enableEDBUsageReporting: true
  hybridManagerInstallation: <install-name>
```

1. 初めてインストールする場合、変更を適用します。

```
helm install <release-name> <chart-path> -f values.yaml
```

既存のインストールを更新するには、 helm upgrade コマンドを実行します。

```
helm upgrade <release-name> <chart-path> -f values.yaml
```

- enableEDBUsageReporting が false に設定されている場合、使用データは収集されず、EDB に送信されません。
- enableEDBUsageReporting が true に設定されている場合、フィールド company_code および hybridManagerInstallation は必須です。使用状況レポートが有効になっているときにいずれかの値が欠落している場合、Helm はエラーで失敗します。!!!

Notifications

Hybrid ManagerHM を使用すると、受信する通知の種類をカスタマイズできます。2 種類の通知チャネルは次のとおりです。

ユーザー通知電子メール、HM インボックス、またはその両方を介して通知を受信できます。これらの通知は、それらを有効にしているすべてのユーザー、および自分のロールで許可されている場合に対象となります。

システム通知 電子メールメールグループ、Webhook、Pager Duty、または任意の組み合わせを介して通知を受信できます。これらの通知は、イベントごとに 1 回トリガーされます。システム通知は、クラスターライフサイクルの変更作成、更新、削除、セキュリティとインデックスの推奨事項、自動タスクアクティビティ、その他のプラットフォームイベントを含むアクティビティログイベントによって駆動されます。

HM でのユーザーのロールにより、通知設定へのアクセスの範囲が決まります。通知は次の設定セクションに分類されます。

- アカウント ユーザー通知チャンネル用。
- グローバル ユーザーおよびシステム通知チャネル用。
- プロジェクト ユーザーおよびシステム通知チャネル用。

[アカウント設定] セクションの通知は次のとおりです。

- あなたに割り当てられた新しいプロジェクトロール

- プロジェクトロールがあなたから割り当て解除されました
- 個人アクセスキーの有効期限が間もなくエキスパイア

グローバル設定セクションの通知は次のとおりです。

- 新しいプロジェクトが作成されました
- マシンアクセスキーの有効期限が間もなくエキスパイア

> **注釈**
>
> このセクションは、組織の所有者のみが表示されます。

[プロジェクト設定] セクションの通知は次のとおりです。

- アクティブなクラスターが一時停止
- 一時停止したクラスターが再アクティブ化されました
- クラスターメンテナンスのアップグレードの失敗
- プロジェクトに招待されたユーザー
- サードパーティのモニタリング統合の失敗

通知の設定

プロジェクトの所有者/編集者および組織の所有者/管理者は、自分に表示されるイベントの通知を有効/無効にできます。HM インボックス、電子メール、またはその両方で通知を受信できます。

ユーザーが行った通知設定は、そのユーザーにのみ適用されます。電子メール通知が有効になっている場合、電子メールはユーザーのログインに関連付けられた電子メールアドレスに送信されます。

組織レベルおよびプロジェクトレベルの通知の場合、HM は、イベントのターゲット視聴者と各ユーザのロールおよび有効になっている通知設定を相互参照することにより、各通知の受信者を決定します。両方の条件を満たす必要があります。ユーザーのロールは、その通知タイプを受信できるように資格を付与し、設定で有効にする必要があります。アカウントレベルの通知は、常に関係する個々のユーザーにのみ配信されます。

システム通知チャネルは、組織の所有者がグローバルレベルで、またはプロジェクト所有者がプロジェクトレベルで構成できます。デフォルトでは、各組織のプロジェクトはグローバルレベルのシステム通知チャネルを継承します。ただし、プロジェクトレベルでシステム通知チャンネルを有効/無効または再構成して、デフォルト設定をオーバーライドすることができます。

通知設定はスコープ階層に従います。グローバル設定はすべてのプロジェクトに適用され、プロジェクトレベルの設定はそれらをオーバーライドします。ニーズに合った最小のスコープで構成します。組織全体のデフォルトの場合はグローバル、プロジェクト固有のルーティングの場合はプロジェクトレベル。

グローバルレベルで通知を設定する

組織の所有者のみがグローバルレベルの通知を設定できます。

Webhook

1. HM コンソールで、プロフィールメニューから **Settings** を選択します。
2. **Notifications** タブを選択します。
3. **Webhook** の横にある edit アイコンを選択し、詳細を提供します。

- **Host** ウェブフックのエンドポイント URL。
- **Method** 要求の HTTP メソッド POST または PUT.
- **HTTP ヘッダー** オプショナルキーと値のペアとして、要求に含めるカスタム HTTP ヘッダー。
- **Authentication** 認証タイプを選択し、資格情報を提供します。 - **Basic Auth** : オプショナルのユーザー名とパスワード。 - **Bearer Token** ベアラートークンを提供します。

1. **Save** を選択します。

ポケベル当番

Pager Duty 通知チャネルを設定する前に

- 昇格した権限を持つ Pager Duty アカウントへのアクセスが必要です。
- Events API v2 統合を使用して Pager Duty サービスを作成および構成する必要があります。
- Pager Duty アカウントで、目的のサービスに移動し、その **統合**設定にアクセスします。
- Events API v2 統合を追加し、要件に従って構成します。
- 新しく作成した統合のために Pager Duty が提供した統合キールーティングキーをコピーします。

次に、チャネルを構成できます。

1. HM コンソールで、プロフィールメニューから **Settings** を選択します。
2. **Notifications** タブを選択します。
3. **Pager Duty** の横にある edit アイコンを選択し、 **ルーティングキー**を設定します。
4. **Save** を選択します。

電子メール

1. HM コンソールで、プロフィールメニューから **Settings** を選択します。
2. **Notifications** タブを選択します。
3. **電子メール** の横にある編集アイコンを選択し、詳細を入力します。

- **電子メールの宛先**オプションで、システム通知用のグループメールアドレス。HM は、チャネルを構成した人に関連するイベントだけでなく、このスコープのすべてのシステム通知をこのアドレスに送信します。個人のメールアドレスではなく、共有の受信ボックスまたは配布リストを使用します。

- **Email From** 通知を送信するメールアドレス。
- **Host** メールサービスの URL。
- **Port** メールサービスのポート.
- **Username** メールサービスのユーザー名.
- **Password** メールサービスのパスワード。

> **注釈**
>
> **電子メールの宛先**が設定されている場合、HM は対象となる個々のユーザーに加えて、そのアドレスに通知を配信します。彼らの代わりにではありません。 **Email To**が空白のままの場合、資格のある個々のユーザーのみが HM アカウントのメールアドレスでメール通知を受信します。 !!!

1. **Save** を選択します。

プロジェクトレベルで通知を設定する

プロジェクトレベルの通知設定は、グローバル設定をオーバーライドできます。各プロジェクトは、一意の構成を持つことができます。

1. HM コンソールで、 **Projects** タブを選択します。
2. 任意のプロジェクトを選択します。
3. プロジェクトの概要ページに移動します。
4. 左側のナビゲーションで、 **Settings** を選択します。
5. **通知** を選択します。
6. **ステップ** *9 Helm* **の所有権を譲渡する** - *Webhook* **構成** 、 **ポケベル当番** 、または**電子メール** の説明に従って設定を指定します。

HM コンソールで通知を表示する

次のロールのユーザーは通知を表示できます。

- 組織の所有者はグローバル通知を表示できます。
- プロジェクトの所有者/編集者は、プロジェクトの通知を表示できます。
- すべてのユーザーが自分のアカウント通知を表示できます。

各通知は、HM で複数のロールが割り当てられているユーザーが属しているレベルとプロジェクトを示します。

通知を表示するには、HM コンソールの上部にあるベルアイコンを選択します。通知を読んだり、未読としてマークしたり、アーカイブしたりできます。

一部の通知、特にタスク関連の通知には、影響を受けるリソースへの直接リンクが含まれています。必要なアクセス権限がある場合、リンクを選択すると、関連するページに移動します。

未読のブロードキャスト通知-メンテナンス時間などのプラットフォーム全体のお知らせ-がある場合、コンソールの上部に永続的なバナーが表示されます。複数の未読のブロードキャストがある場合、バナーはスタックします。バナーで**既読としてマーク** を選択して非表示にします。

通知設定の管理

1. HM コンソールで、プロフィールメニューから **My Account** を選択します。
2. **Notifications** タブを選択します。通知は、使用可能なアカウント、組織、およびプロジェクトごとにグループ化されています。
3. **Account** 、**Global** 、または **Project** セクションまでスクロールして、管理するイベントを表示します。
4. エンティティの横にある右矢印を選択して、イベントのリストを展開します。
5. 通知ごとに、次のことができます。 - イベント名の横にあるトグルを使用して、そのイベントの通知の受信をオプトインまたはオプトアウトします。- イベントトグルが有効になっている場合、各チャネル電子メール、アプリ内、Webhook、Pager Duty のチェックボックスを選択して、その通知を配信するチャネルを制御します。

Alerts

アラートは、Postgres クラスターのパフォーマンスに影響を与える重要なイベントについて積極的に通知する Hybrid ManagerHM 可観測性エコシステムに不可欠です。重要なパフォーマンスメトリックを継続的に監視し、事前定義されたしきい値を超過するたびにアラートをトリガーします。この堅牢なアラートシステムは、HM エコシステムにシームレスに統合され、すべてのリージョンの Postgres クラスターとそのインフラストラクチャから収集されたリアルタイムデータを活用します。アラートは、HM で作成されたプロジェクトごとにデフォルトで有効になっています。

アラートシステムは、直感的な通知を介してタイムリーな通知を保証し、情報に基づいた迅速なアクションを可能にします。包括的なアラート構成オプションにより、特定のしきい値と間隔を構成して、アラートエクスペリエンスをカスタマイズできます。モニタリングとアラートのこのシームレスな統合により、完全な可視性が提供され、潜在的な問題への迅速な対応が可能になり、最適なクラスターパフォーマンスを維持できます。

アラートを表示する

クラスターの詳細ページの **モニタリング**タブにあるアラートにアクセスし、特定のクラスターのアラートの詳細を表示できます。

- HM クラスターの場合

1. HM コンソールで、 **Projects** を選択します。
2. リストからプロジェクトを選択します。
3. 左側のナビゲーションで、 **Clusters** を選択します。
4. 準備ができているクラスターを選択します。
5. On クラスターの詳細ページで、 **Monitoring** タブを選択します。

- 外部 HM クラスターの場合

1. HM コンソールで、 **Estate** を選択します。
2. すべてのクラスターをリストしているテーブルに移動します。
3. 管理が自己管理である準備ができているクラスターを選択します。
4. クラスターの詳細ページで、 **モニタリング** タブ。

[**Monitoring**] タブには、アクティブなアラートリストが表に表示されます。このタブの上部には、アクティブなアラートの概要が表示され、重大度レベルが高、中、および低に設定されています。

次を使用して特定のアラートを検索できます。

- クエリベースの検索のための検索ボックス
- 特定の日付範囲ベースの検索の日付範囲セレクター
- 重大度レベルベースの検索のフィルタ
- 重大度、アラート開始時間、およびアラート名に基づいてアラートリストを並べ替える並べ替えユーティリティ
- アラートリストを更新するための更新ボタン
- 設定に従ってテーブルに表示する列を選択する列セレクター
- アラートリストを CSV 形式でダウンロードするための Download ボタン

アラートを手動で更新するには、 **Refresh** を選択します。最後の更新間隔は、アラートテーブルの下部に表示されます。

Node 列でノードを選択して、アラートがトリガーされたときの状態でチャートを表示します。ノードを選択すると、**Monitoring** タブのフィルタが適用され、関連するチャートが表示されます。

CSV のダウンロード — 右上隅の [Download] ボタンを使用して、アラートの詳細を CSV 形式でダウンロードできます。CSV ファイルには、次の列が含まれています。

- アラートの説明
- スタート時間
- アラートの重大度
- アラートをトリガーするメトリック

・ アラートが発生したノード

- アラートしきい値
- クラスター ID
- プロジェクト ID

アラートの構成

プロジェクトレベルでアラートを構成できます。プロジェクト管理者はアラートを構成できます。アラートが構成および適用されると、システムがアラートを完全に処理するまでに数分を必要とします。

アラートを構成するには

1. HM コンソールの上部で、 **Projects** を選択します。
2. リストからプロジェクトを選択します。
3. 左側のナビゲーションで、 **Settings** を選択します。
4. **Alerts** を選択して、アラート構成にアクセス/編集します。

アラートごとに、[**Edit**] ボタンを選択して次のことを行います。

- 有効/無効 —必要に応じて、特定のアラートをアクティブまたは非アクティブにします。
- Notify After — アラートがトリガーされるまでの時間を設定します。
- しきい値 —超過したときにアラートをトリガーするパーセンテージまたは値を指定します。

デフォルト構成のアラートのリスト

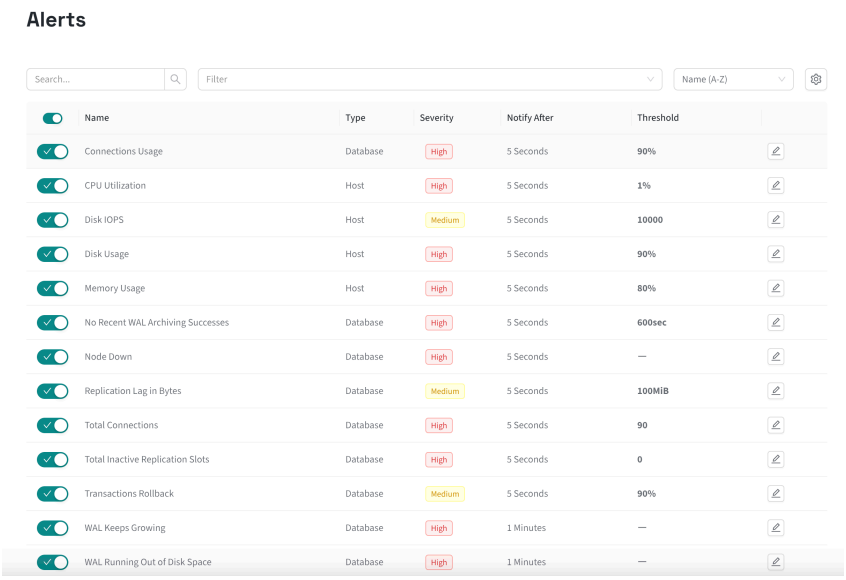


図 39: Alerts

これらのアラートはデータベースレベルで設定できます。

- 接続の使用状況
- 最近の WAL アーカイブは成功しませんでした
- ノードダウン
- バイト単位のレプリケーションラグ
- 合計接続数
- 非アクティブなレプリケーションスロットの合計
- トランザクションのロールバック
- WAL は成長し続けます
- WAL のディスク領域が不足しています

これらのアラートはホストレベルで設定できます。

- CPU 使用率
- ディスク IOPS
- ディスクの使用量
- メモリ使用量
- アラート構成に加えられた変更は、自動的に保存されます。
- 技術的な制限のため、すべてのアラートは外部 HM クラスターでは使用できません。

Auditing platform activity

アクティビティログは、ハイブリッドマネージャー HM 環境で発生したすべてのことの時系列の記録です。クラスターの変更、自動化タスクの実行、ユーザーアクション、アクセス制御の変更など。これを使用して、何が変更されたか、誰がいつ変更したかを理解します。

アクティビティログ は、 通知を設定する オブザーバビリティ のイベントソースでもあります。**アクティビティログ**がオートメーションタスクの完了や推奨事項の HM 適用などのプラットフォームイベントを記録すると、HM はそのイベントを使用して、プロジェクトまたは組織レベルで構成した通知チャネルをトリガーします。

アクティビティログへのアクセス

アクティビティログ は、2 つのレベルで利用できます。

- **プロジェクトレベル** — プロジェクト内の、左側のナビゲーションで、**アクティビティログ**を選択します。このビューには、そのプロジェクトにスコープされるすべてのイベントが表示され、プロジェクトロールを持つすべてのユーザーが利用できます。
- **組織レベル** — 組織ナビゲーションで、プロファイル名を選択し、**アクティビティログ**を選択します。このビューには、組織内のすべてのプロジェクトのイベントが表示され、ソースを識別するための**プロジェクト** 列が表示されます。 **組織所有者** または**組織管理者** ロールが必要です。

ログの読み取り

アクティビティログの各行は、単一のイベントを表します。列は次のとおりです。

Column	Description
Activity Name	何が起こったのか — クラスターの作成、タスクの承認、またはユーザーロールの変更など。タスク関連のイベントの場合、識別しやすいようにアクティビティ名にタスク名が含まれます。
User	イベントをトリガーしたユーザーまたはシステムプロセス。
Date	イベントが発生したとき。
Resource	影響を受けるオブジェクトのタイプ。一般的な値は次のとおりです。 **クラスター** 、 **クラスターテンプレート** 、 **カタログ** 、 **タスク** 、 **タスク承認** 、 **ユーザーロール** 、 **プロジェクト** 、 **ID プロバイダー** 、 **通知資格情報** 、および **ロールマッピングルール** 。組織レベルのビューには、 **アクセスキー** 、 **ユーザー** 、 **組織** 、 **タグ** イベントなどの追加リソースも表示されます。

次のページに続く

表 87 – 前のページからの続き

Column	Description
Project	組織レベルのビューのみ イベントが属するプロジェクト。

フィルタリングと検索

ログの上部にあるフィルタを使用して、結果を絞り込みます。

- **日付範囲** —7 日、15 日、および 30 日のプリセット範囲を使用できるか、カスタム範囲を設定します。
- **リソース** リソースタイプたとえば、 **クラスター** または**タスクイベント**のみを表示し、オプションでそのタイプ内の特定のリソースによってフィルターします。
- **プロジェクト** — 組織レベルのビューのみ 1 つ以上の特定のプロジェクトにイベントをフィルタリングします。
- **検索** — 名前またはリソースでイベントを検索します。

アクティビティデータのエクスポート

Download CSV を選択して、現在のフィルタリングされたビューを CSV ファイルとしてダウンロードします。これは、コンプライアンス監査、または HM 外部の関係者とアクティビティ履歴を共有する場合に役立ちます。

アクティビティを通知に接続する

プロジェクトまたは組織レベルで通知チャネル Webhook、Pager Duty、または電子メールを構成している場合、 **アクティビティログ**レコードのイベントはシステム通知をトリガーできます。これは、**アクティビティログ**を手動で監視する必要がないことを意味します。チャンネルを構成すると、HM が関連するイベントを外部ツールに自動的にプッシュします。

チャンネルの構成と受信するイベントの管理の手順については、通知を設定する オブザーバビリティ を参照してください。

5.3.4.3 Upgrading Hybrid Manager

Hybrid ManagerHM は、 *Hybrid Manager* デュアルリリース戦略 **LTS 長期サポート** および**イノベーションリリース IR** を提供します。これらのストリームは異なるバージョン化ロジックに従うため、それらの間を移動するためにサポートされているパスは、現在のバージョンとターゲット環境によって異なります。

> **Tip**
>
> **Postgres データベースクラスター**をアップグレードする場合は、*Upgrading database clusters in Hybrid Manager* を参照してください。

イノベーションリリースは、日付ベースのバージョン管理スキーム例 **2025.11** に従います。これらのリリースは、最新の機能とプラットフォームの強化をより速いリズムで提供します。

ただし、これらのリリースは無限ではなく、次の長期サポート LTS マイナーバージョンに収束するように設計されています。

シークエンシャル要件 IR ストリーム内で毎月アップグレードする必要がありますたとえば、2025.11→2025.12。

統合ポイント年に 2 回、IR ストリームが「統合」されます。統合ポイント後に更新を受信し続けるには、新しくリリースされた LTS バージョン 1.4.0 などに移行する必要があります。これは、来月の IR の基盤として機能します。

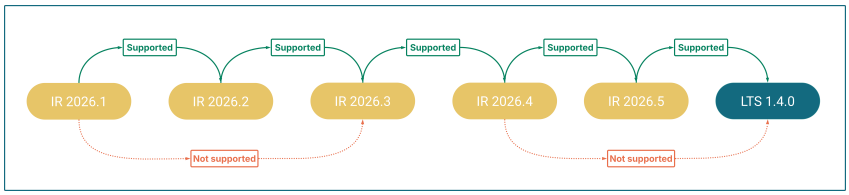


図 40: IR sequential upgrade

5.3.4.3.1 LTS アップグレードパス

LTS リリースは、標準のセマンティックバージョニング `major.minor.patch`、例、**1.3.x** に従います。これらのバージョンは安定性に焦点を当てており、長いサポートライフサイクルを必要とする実稼働環境を対象としています。各 LTS リリースは、リリース日から 15 か月間サポートされ、バグとセキュリティ修正を含む毎月のパッチリリースを受け取ります。

マイナーバージョンとパッチバージョンのアップグレード

マイナーバージョンからすぐに次のマイナーリリースたとえば **1.3.x** から **1.4.y** などにアップグレードできます。ただし、続行する前に、現在のマイナーリリースの利用可能な最新のパッチバージョンを実行していることを最初に確認する必要があります。

同じマイナーバージョン内では、パッチのアップグレードは柔軟です。中間パッチをインストールする必要なく、任意のパッチバージョンから上位のパッチバージョン 1.3.1 から 1.3.5 などにアップグレードできます。

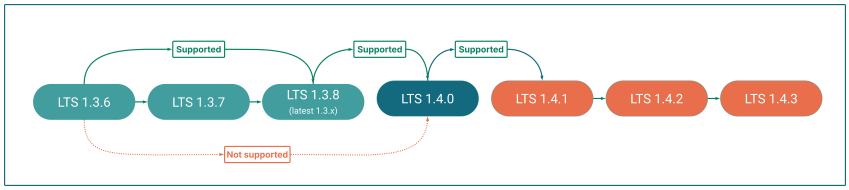


図 41: LTS minor and patch releases

メジャーバージョンのアップグレード

新しいメジャーバージョン **2.0** がリリースされた場合、アップグレードを実行するには、利用可能な最新の **1.x** マイナーリリースを実行している必要があります。

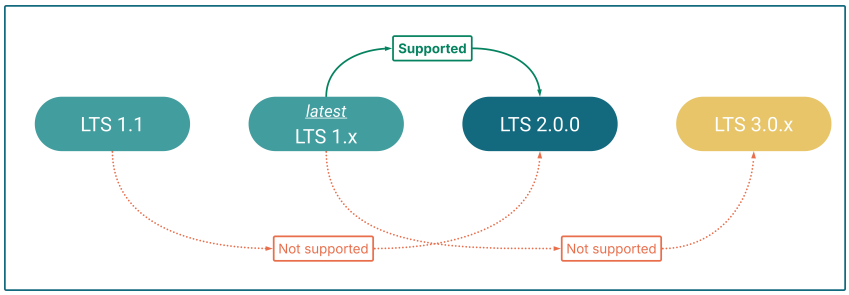


図 42: IR sequential upgrade

5.3.4.3.2 クロスストリームアップグレードパス

LTS とイノベーションリリース間の移動は可能ですが、バージョン番号とプラットフォームの制限に関する特定の制約の対象となります。

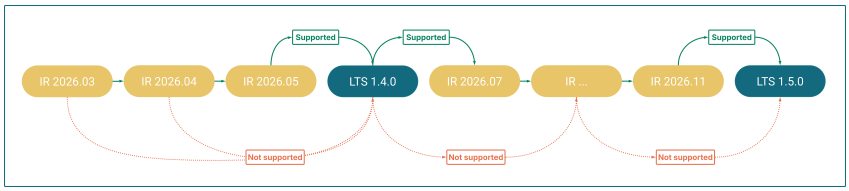


図 43: IR sequential upgrade

LTS へのイノベーションリリース

サイクル 1 年に 2 サイクルの **最終イノベーションリリース**から、これらの機能を統合する **LTS リリース**に移動することはできますが、他のイノベーションリリースから LTS リリースに移動することはできません。。

現在のイノベーションリリースが LTS バージョンに移行できるかどうかを判断するには、各新しいリリースに提供されている文書化されたアップグレード手順を参照してください。これらのリリース固有の手順は、バージョンがイノベーションリリースであるかどうか、および LTS ストリームへの移行をサポートするかどうかを明示的に示します。

LTS からイノベーションリリース

LTS リリースをその直後のイノベーションリリースにアップグレードできます。

LTS バージョンから任意またはそれ以降のイノベーションリリースにジャンプすることはできません。最初に直接の後継者に移動し、次にシークエンシャルな IR アップグレードパスをたどる必要があります。

> **警告**
>
> LTS リリースからイノベーションリリースに移行すると、次の統合ポイントまで LTS リリースに戻すことはできません。以下のイノベーションリリースから LTS へを参照してください。統合ポイントは年に 2 回発生します。

HM 管理の Postgres データベースクラスターへのアップグレードの影響は、特定のアップグレードパスと更新されるコンポーネントによって異なります。

多くのアップグレードはデータベーストラフィックを中断しないように設計されていますが、特定のシステムの更新は、HM 管理の Postgres データベースクラスターの自動再起動をトリガーする場合があります。予期しないサービスの中断を回避するには、アップグレードパスの特定の要件を確認します。

Upgrading from 2026.3 to the 2026.4 Innovation Release

2026. 3 から 2026.4 へのアップグレード。

前提条件

- Kubernetes クラスター `kubectl` 、`helm` への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは `values.yaml` ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール
- `helm` 最新バージョンを推奨
- `kubectl`
- `edbctl` バージョン `1.4.0` 以降または `main` からのビルド

アップグレード 2026.3 → 2026.4

このリリースでは、`dbaas` シナリオを導入し、`upm-system-db` の新しいシークレットが必要で、データの破損とリソースの競合を防ぐための必須のアップグレード前の手順が含まれています。

upm-system-db の新しいシークレットを作成する

アップグレードする前に、 `upm-system-db` コンポーネントに必要な新しいシークレットを作成します。

```
edbctl setup create-install-secrets --version v2026.4.0
```

> **注釈**
>
> このコマンドには、*edbctl* バージョン *1.4.0* または`main` からのビルドが必要です。以前のバージョンは、*--version* フラグをサポートしていません。

2026. 4 のラベルの変更により、既存の `fluent-bit` StatefulSet と DaemonSet が不変になります。アップグレードが失敗しないように、アップグレードの前に両方のリソースを削除します。

```
kubectl delete statefulset fluent-bit-k8s-events -n edb-observability
kubectl delete daemonset fluent-bit -n edb-observability
```

オペレーターは、アップグレード中にそれらを自動的に再作成します。

統計コレクターデータベースを削除します

> **警告**
>
> アップグレードを実行する前にこの手順を完了します。 *upm-api-stats-collector* コンポーネントは、このリリースの Bluefin 拡張機能バージョンをバンプして、重大な WAL ファイルの問題に対処します。既存のデータは、新しいバージョンと互換性がないため、移行できません。最初にクラスターを削除せずにアップグレードすると、データの破損が発生します。

統計コレクターは 7 日間のテレメトリーデータのみを保持するため、クラスターの削除は、ほとんどの展開で無停止です。このデータを保存する必要がある場合は、続行する前に EDB Support に連絡してください。

```
kubectl -n upm-api-stats-collector delete cluster stats-collector-db
```

クラスターがすぐに終了しない場合は、ポッドを強制削除します。

```
kubectl delete pod --force -n upm-api-stats-collector -l cnpg.io/cluster=stats-
↪collector-db
```

1. 続行する前に、関連するすべてのリソースが削除されていることを確認します。

```
kubectl -n upm-api-stats-collector get cluster stats-collector-db
kubectl -n upm-api-stats-collector get pod stats-collector-db-1
kubectl -n upm-api-stats-collector get pvc stats-collector-db-1
```

リソースがなくなると、各コマンドは NotFound を結果ます。3 つすべてが NotFound を返した場合のみ続行します。

アップグレードを実行する

Helm ベースのインストールの場合

1. Helm リポジトリを更新します。

```
helm repo update
```

1. SKU が許可する場合は、values.yaml の scenarios フィールドに dbaas を追加します。

```
scenarios: "core,migration,ai,analytics,dbaas"
```

1. アップグレードを実行します。

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --version 2026.4.0 \
  --values values.yaml \
  --namespace edbpgai-bootstrap \
  --wait
```

オペレーターベース CRD インストールの場合

dbaas シナリオは、2026.3 CRD スキーマには存在せず、2026.3 オペレーターの実行中に追加できません。最初にオペレーターをアップグレードしてから、HybridControlPlane リソースを更新します。

1. edb-hcp-operator Helm チャートをバージョン 2026.4.0 にアップグレードします。

```
helm template \
  --namespace edbpgai-bootstrap \
  --install \
  --version 2026.4.0 \
  --include-crds \
  edb-hcp-operator enterprisedb-edbpgai/edb-hcp-operator \
  --set "controllerManager.manager.image.repository=docker.enterprisedb.com/pgai-
↪platform/edb-hcp-operator/manager" \
  --set "controllerManager.manager.image.tag=2026.4.0" \
  --set "imagePullSecrets[0].name=edb-cred" \
  | kubectl apply -f -
```

1. バージョンを更新し、SKU が許可する場合は、単一の patch コマンドで dbaas シナリオを追加します。

```
kubectl patch hcp edbpgai --type json \
  --patch [{"op":"add", "path":"/spec/scenarios/-", "value":"dbaas"}, {"op":"replace",
↪ "path":"/spec/version", "value":"v2026.4.0"}]
```

dbaas を有効にしていない場合は、バージョンのみを更新します。

```
kubectl patch hcp edbpgai --type json \
  --patch [{"op":"replace", "path":"/spec/version", "value":"v2026.4.0"}]
```

dbaas が有効になっていない場合に何が起こるかについては、*dbas* **シナリオ** を参照してください。

1. リソースに注釈を付けることにより、アップグレードをトリガーします。

```
kubectl annotate hybridcontrolplane edbpgai --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

1. オペレーターはバージョンの変更を検出し、ローリング更新を開始します。次の方法で進行状況をモニタします。

```
kubectl get hybridcontrolplane edbpgai -w
```

ビーコンサーバーの使用状況テレメトリを有効にする

2026. 4 では、使用状況サーバーテレメトリ収集を有効にするための新しいブールフィールドを導入しました。これを有効にするには、既存の必須フィールドの横に enableEDBUsageReporting を true に設定します。

オペレーターベースのインストールの場合、HybridControlPlane CR を更新します。

```
spec:
  beaconServer:
    enableEDBUsageReporting: true
    hybridManagerInstallation: "<your-installation-name>"
  globalParameters:
    company_code: "<your-company-code>"
```

Helm ベースのインストールの場合、同等の値を values.yaml に追加します。

```
beaconServer:
  enableEDBUsageReporting: true
  hybridManagerInstallation: "<your-installation-name>"
parameters:
  global:
    company_code: "<your-company-code>"
```

構成を更新した後、再度アノテーションを付けて変更を適用します。

```
kubectl annotate hybridcontrolplane edbpgai --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

dbas シナリオ

dbaas シナリオは 2026.4 の新しいシナリオで、core とは別です。クラスターのプロビジョニング、バックアップ、保存場所、テンプレートを含むデータベースクラスター管理が有効になります。使用可能なすべてのシナリオの概要については、フェーズ *1* アーキテクチャの計画 を参照してください。

> **⚠ 警告**
>
> 2026.3 からアップグレードする既存のインストールは、シナリオリストに `dbaas` を明示的に追加して、Postgres クラスター管理機能を保持する必要があります。これがないと、クラスター UI は非表示になり、クラスター管理 API は `404` を返します。既存のクラスターは実行を継続しますが、プラットフォームから管理できなくなります。

- クラスター管理 API は 404 を結果ます。
- **Clusters** セクションは HM コンソールに非表示になっています。
- 既存の Postgres クラスターは実行され続けますが、プラットフォームから管理できなくなります。

CR にパッチを適用し、リコンシリエーションを再トリガーすることにより、アップグレード後にいつでも dbaas を追加できます。

```
kubectl patch hcp edbpgai --type json \
  -p [{"op":"add", "path":"/spec/scenarios/-", "value":"dbaas"}]

kubectl annotate hybridcontrolplane edbpgai \
  --overwrite edbpgai.com/ready-for-upgrade=true
```

障害対応

edb-migration-copilot のインストールに失敗する

アップグレードが次のようなエラーで失敗した場合

```
failed to execute kapp command, error: update secret/ragchew-ai-vendor-secrets (v1)␣
↪namespace: edb-migration-copilot
```

競合するシークレットを削除し、アップグレードを再トリガーします。

```
kubectl delete secret -n edb-migration-copilot ragchew-ai-vendor-secrets
kubectl annotate hybridcontrolplane edbpgai --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

Upgrading from 2026.2 to the 2026.3 Innovation Release

2026. 2 から 2026.3 へのアップグレード。

前提条件

- Kubernetes クラスター kubectl 、helm への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは values.yaml ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール
- helm 最新バージョンを推奨
- kubectl

アップグレード 2026.2 → 2026.3

このリリースでは、テレメトリコンポーネントを新しい名前空間に移動します。これには、アップグレードの前に手動の手順が必要です。

テレメトリー名前空間の準備

アップグレードする前に、既存の logging 名前空間にアノテーションを付けて、テレメトリコンポーネントが新しい名前空間にきれいに移動することを確認します。

```
kubectl annotate namespace logging kapp.k14s.io/delete-strategy=orphan --
↪overwrite=true
```

アップグレードを実行する

Helm ベースのインストールの場合

1. Helm リポジトリを更新します。

```
helm repo update
```

1. アップグレードを実行します。

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --version 2026.3.0 \
  --values values.yaml \
  --namespace edbpgai-bootstrap \
  --wait
```

オペレーターベース CRD インストールの場合

オペレーターを介してインストールが管理されている場合、HybridControlPlane リソース内のバージョン仕様を更新します。

1. HybridControlPlane リソースを編集します。

```
kubectl edit hybridcontrolplane <resource-name>
```

<resource-name> は、インストール中に割り当てた名前です。

1. YAML エディターで、spec.version フィールドを見つけて、2026.3.0 に更新します。
2. 変更を保存し、リソースに注釈を付けてリコンシリエーションを強制します。

```
kubectl annotate hybridcontrolplane <resource-name> --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

1. オペレーターはバージョンの変更を検出し、ローリング更新を開始します。次の方法で進行状況をモニタします。

```
kubectl get hybridcontrolplane <resource-name> -w
```

クリーンアップ

アップグレードが完了したら、logging 名前空間を削除します。

```
kubectl delete namespace logging
```

Upgrading from 2026.1 to the 2026.2 Innovation Release

2026. 1 から 2026.2 へのアップグレード。

前提条件

- Kubernetes クラスター kubectl 、helm への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは values.yaml ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール
- helm 最新バージョンを推奨
- kubectl 、 yq 、 base64 、 jq

アップグレード 2026.1 → 2026.2

このリリースでは、再設計されたアイデンティティプロバイダ IdP 構成システムと、手動のアップグレード前および後の手順を必要とする新しいカスタムリソース定義 CRD を導入しています。

以前のリリースからの変更 IdP の移行

- config.yaml は、以前に upm-dex シークレットからロードされましたが、ConfigMap からマウントされます。静的コネクタ構成はサポートされなくなりました。コネクタは、IdP Management API を使用してプロビジョニングする必要があります。$DB_USER や$DB_PASSWORD のような環境変数値は、upm-db-url シークレットから取得され、upm-dex 名前空間に自動的に伝播されます。
- 新しい hm-portal-bootstrap シークレットを default 名前空間に作成する必要があります。これは、リローダーアノテーションを介して upm-dex 名前空間に自動的に複製されます。これには以下が含まれます。
- fernet-key — 機密コネクタ構成フィールドの暗号化キー
- static-passwords.yaml —ブートストラップアクセスの初期ユーザーアカウント
- 新しい CRD は 2026.2 で導入され、Helm は CRD 変更をスキップするため、アップグレードの前に手動で適用する必要があります。
- 移行が成功した後、 upm-dex および dex-fernet-key シークレットを手動で削除します。

1. 現在の IdP 設定をバックアップします

HM ポータルから、所有者アカウントでログインし、https://<portal_dns_name>/settings/identity-providers に移動します。IdP ごとに、画面に部分的にのみ表示されているフィールドを含むすべての設定を保存します。

> **警告**

HM IdP API は、*clientSecret* OIDC や `BindPassword` LDAP などの機密フィールドを結果ずます。続行する前に、実際の ID プロバイダー Okta、Google などからこれらの値を直接取得します。 !!!

- EDB Internal の場合 upm-dex シークレットを直接バックアップします。

```
kubectl get secret upm-dex -n upm-dex -o yaml > upm-dex-backup.yaml
```

- オプションで、CLI から IdP コネクタ構成をエクスポートします。

```
kubectl get hcp edbpgai -o json | jq .spec.pgai.portal.authentication.idpConnectors >␣
↪idps.json
```

1. 既存の IdP 構成から証明書ファイルを生成する

既存の IdP コネクタから証明書を抽出します。

```
kubectl get hybridcontrolplane edbpgai -o yaml \
    | yq .spec.pgai.portal.authentication.idpConnectors[0].config.caData \
    | base64 -d > okta-cert.pem
```

注釈

複数のコネクタがある場合は、配列インデックスを調整します。 *okta-cert.pem* はそのままにしておきます。ステップ 8 で必要になります。

1. dex-fernet-key シークレットから fernet キーを抽出します。

```
DEX_FERNET_KEY=$(kubectl get secret -n upm-dex dex-fernet-key \
    -o jsonpath={.data.fernet-key} | base64 -d)
```

1. upm-dex シークレットのデコードされた config.yaml から静的ユーザーパスワードを抽出します。

```
staticPasswords=$(kubectl get secret -n upm-dex upm-dex -o yaml \
    | yq .data."config.yaml" | base64 -d \
    | yq .staticPasswords)
```

重要

各ユーザーの `userID` を抽出されたとおりに移行します。これを変更すると、新しいユーザー ID が作成され、同じメールアドレスと名前を持つ重複ユーザーが作成され、元のユーザーに関連付けられている既存のデータを孤立させます。 !!!

1. 手順 3 および 4 の値を使用して、default 名前空間に hm-portal-bootstrap シークレットを作成します。

```
kubectl create secret generic hm-portal-bootstrap \
    --namespace default \
    --from-literal=fernet-key="$DEX_FERNET_KEY" \
    --from-literal=static-passwords.yaml="staticPasswords:
$staticPasswords"

# Add the replicator annotation
kubectl annotate secret hm-portal-bootstrap --namespace default \
    replicator.v1.mittwald.de/replicate-to=upm-dex
```

レプリケーターアノテーションにより、upm-dex 名前空間に伝播されます。

1. ご使用の環境に適切な方法を使用して、新しい CRD を適用します。

> **⚠ 警告**
>
> Helm は、アップグレード中に CRD の変更をスキップします。これらはアップグレードする前に適用する必要があります。 !!!

リモート CS Helm リポジトリの Helm パッケージを使用する場合

```
helm pull edbpgai/edb-hcp-operator --version 2026.2.0 --untar
kubectl apply -f edb-hcp-operator/crds/preflight-crd.yaml
kubectl apply -f edb-hcp-operator/crds/postflight-crd.yaml
```

EDB Internal — 一時クラスターの場合

```
git clone -b 2026.2 git@github.com:EnterpriseDB/edb-hcp-operator.git
cd edb-hcp-operator/deploy/charts/edb-hcp-operator/crds
kubectl apply -f preflight-crd.yaml
kubectl apply -f postflight-crd.yaml
```

- Helm 管理の CRD に注釈を付けます。

```
kubectl annotate crd preflights.edbpgai.edb.com postflights.edbpgai.edb.com \
meta.helm.sh/release-name=edbpgai-bootstrap \
meta.helm.sh/release-namespace=edbpgai-bootstrap \
--overwrite

kubectl label crd preflights.edbpgai.edb.com postflights.edbpgai.edb.com \
app.kubernetes.io/managed-by=Helm \
--overwrite
```

1. アップグレードを実行する

ブートストラップ方法のインストール

1. Helm リポジトリを更新します。

```
helm repo update
```

1. アップグレードを実行します。

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
--version 2026.2.0 \
--values values.yaml \
--namespace edbpgai-bootstrap \
--wait
```

演算子メソッドのインストール

Operator 方法を使用したインストールの場合、HybridControlPlane リソース内のバージョン仕様を更新する必要があります。

1. HybridControlPlane リソースを編集します。

```
kubectl edit hybridcontrolplane <resource-name>
```

<resource-name> は、インストール中に割り当てた名前です。

1. YAML エディターで、spec.version または同等のバージョンフィールドを見つけて、2026.2.0 に更新します。
2. 変更を保存します。リソースにアノテーションを付けることにより、調整を強制します。

```
kubectl annotate hybridcontrolplane <resource-name> --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

1. オペレーターはバージョンの変更を検出し、ローリング更新を開始します。次の方法で進行状況を監視できます。

```
kubectl get hybridcontrolplane <resource-name> -w
```

1. 自動検出された IdP 構成を削除します。

アップグレード後、アプリケーションデータベースに接続して実行します。

```
kubectl exec svc/app-db-rw -c postgres -n upm-beaco-ff-base -- \
  psql -U postgres upm -c \
  "UPDATE upm_api_admin.idp SET deleted_at = NOW() WHERE deleted_at IS NULL;"
```

1. IdP Management API を使用してコネクタを移行します。

続行する前に、 hm-portal-bootstrap が upm-dex 名前空間に存在し、upm-dex ポッドが実行されていることを確認します。

```
kubectl get pods -n upm-dex
```

ポータル UI の使用

1. ネイティブ資格情報でログインし、 **Settings** → **Identity Providers** に移動します。
2. 次の値を使用して新しい IdP 構成を作成します。

Field	Value
Name	*okta* _(ensures the existing user config is maintained)_
Assertion Consumer Service URL	*https://<Portal URL>/auth/callback/okta*
Audience URI	*https://<Portal URL>/auth/samlentity/okta*
Connector ID	*okta*
Single Sign-On URL	From your backup in step 1
Signature Certificate File	*okta-cert.pem* — generated in step 2
NameID Format	*PERSISTENT*
User Attribute	*name*
Email Attribute	*email*

1. **Save** をクリックします。

> **注釈**
>
> **保存とテスト**は、IT が IdP の ACS URL とオーディエンス URI を新しい`/okta`サフィックスで更新するまで、多くの場合失敗します。 IT 部門にチケットを発行して、テストの前にこの更新を要求します。 !!!

REST API の使用

手順 1 で保存した設定を使用して、要求本文を入力します。

```
curl -i -k -X POST https://<portal_url>/api/v1/idp \
--header x-access-key: <HM_org_owner_access_key> \
--header Content-Type: application/json \
--data {
    "general": {
    "name": "okta",
    "description": "Okta Identity Provider",
    "protocol": "OIDC"
    },
    "oidcSettings": {
    "issuer": "https://<your_okta_domain>.okta.com",
    "clientId": "your_client_id",
    "clientSecret": "your_client_secret"
    },
```

(次のページに続く)

(前のページからの続き)

```
    "dexConnectorId": "okta"
}
```

1. Dex をリスタートします。

Dex 展開を再起動して、すべての構成の変更を適用します。

```
kubectl rollout restart deploy upm-dex -n upm-dex
```

1. 移行を検証する
2. 静的ユーザーネイティブ資格情報ログインが動作することを確認します。
3. IdP 構成のログインが動作することを確認します。

クリーンアップ

両方のログインが機能することを確認した後、古いシークレットを削除します。

```
kubectl delete secret upm-dex -n upm-dex
kubectl delete secret dex-fernet-key -n upm-dex
```

Upgrading from 2025.12 to the 2026.1 Innovation Release

2025. 12 から 2026.1 へのアップグレード。

前提条件

- Kubernetes クラスター `kubectl` 、`helm` への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは `values.yaml` ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール - `helm` 最新バージョンを推奨

アップグレード 2025.12 → 2026.1

このリリースでは、新しい認証シークレットを導入し、いくつかのビーコンパラメーターを非推奨にします。

Langflow シークレットを作成する

2026. 1 チャートをインストールする前に、`default` 名前空間に `langflow-secret` シークレットを手動で作成する必要があります。

2027. Langflow シークレットを作成します。

```
kubectl create secret generic langflow-secret -n default \
  --from-literal LANGFLOW_SUPERUSER=langflow \
  --from-literal LANGFLOW_SECRET_KEY="$(python3 -c "from secrets import token_urlsafe;
 print(f{token_urlsafe(32)})")" \
  --from-literal LANGFLOW_SUPERUSER_PASSWORD="$(openssl rand -base64 24 | tr -dc a-zA-
↪Z0-9 | cut -c1-16)"
```

このコマンド

- Langflow 暗号化に必要な base64url エンコードされたアプリケーション秘密キー LANGFLOW_SECRET_KEY を生成します 32 バイトの安全なトークンが必要です。 - シークレットを langflow スーパーユーザー LANGFLOW_SUPERUSER に関連付けます。 - langflow ユーザー LANGFLOW_SUPERUSER_PASSWORD の安全なパスワードを生成します。

1. 作成したシークレットが upm-langflow および upm-beaco-ff-base 名前空間と同期していることを確認します。

```
kubectl annotate secret langflow-secret -n default replicator.v1.mittwald.de/
↪replicate-to="upm-langflow,upm-beaco-ff-base"
```

または、シークレットを手動で作成できます。

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: langflow-secret
  namespace: default
  annotations:
    replicator.v1.mittwald.de/replicate-to: "upm-langflow,upm-beaco-ff-base"
type: Opaque
stringData:
  LANGFLOW_SECRET_KEY: <generated_langflow_secret_key>
  LANGFLOW_SUPERUSER: langflow
  LANGFLOW_SUPERUSER_PASSWORD: <generated_superuser_password>
    EOF
```

values.yaml の更新

HM インストール構成ファイルまたは values.yaml で、次の非推奨フィールドを新しいフィールドに置き換えます。

Remove field:	Replace with:
beaconAgent.provisioning.provider	*parameters.global.load_balancer_provider*

次のページに続く

表 89 – 前のページからの続き

Remove field:	Replace with:
beaconAgent.provisioning.loadBala	*parameters.global.load_balancer_mode*
beaconAgent.provisioning.nodePor	*parameters.global.node_port_domain*

注釈

これらのフィールドを置き換えない場合、HM インストールのアップグレードは引き続き成功しますが、将来的に古いパラメーターと新しいパラメーターの両方に異なる値を設定している場合、新しいパラメーターが古いパラメーターより優先されます。

scenarios スタンザの削除マルチ DC 環境の場合のみ

マルチ DC 環境を実行している場合複数のデータセンターに HM を展開した場合、アップグレードの前に `scenarios` スタンザを削除する必要があります。HM がシングル DC インスタンスの場合、このセクションをスキップして、アップグレードを実行する に直接移動します。

`v2026.1.0` の `scenarios` オプションのスキーマ変更のため、オペレーターは `v2025.12.0` から値を正しくマップできません。これにより、意図しないクリーンアップルーチンがトリガーされます。これを緩和するには、インストールで `scenarios` パラメーターを使用している場合、バージョンアップグレードを開始する前にシナリオスタンザを削除する必要があります。アップグレード後に、スタンザを再適用できます。

新しいスキーマの変更とシナリオインストールオプションは *Deploying Hybrid Manager across multiple data centers* で見つけることができます。

これらの手順に従わないと、オペレーターは既存の構成を無視し、Postgres クラスターを含むすべてのコンポーネントをアンインストールします。

Helm ベースのインストールの場合

1. プライマリインスタンス Hub の `values.yaml` ファイルを開きます。
2. `scenarios:` スタンザ全体を削除します。
3. 構成をセカンダリインスタンススポークに再適用します。

セカンダリインスタンスがリコンサイルを完了し、正常な `Ready` または `Deployed` 状態であることを確認するまで、`v2026.1.0` アップグレードを続行しないでください。

オペレーターベース CRD インストールの場合

1. `HybridControlPlane` リソースを編集します。

```
kubectl edit hybridcontrolplane <resource-name>
```

`<resource-name>` は、インストール中に割り当てた名前です。

1. 仕様からシナリオセクションを削除します。

2. リソースにアノテーションを付けることにより、リコンシリエーションを強制します。

```
kubectl annotate hybridcontrolplane <resource-name> --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

セカンダリインスタンスがリコンサイルを完了し、正常な Ready または Deployed 状態であることを確認するまで、v2026.1.0 アップグレードを続行しないでください。

アップグレードを実行する

Helm ベースのインストールの場合

1. Helm リポジトリを更新します。

```
helm repo update
```

1. アップグレードを実行します。

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --version 2026.1.0 \
  --values values.yaml \
  --namespace edbpgai-bootstrap \
  --wait
```

オペレーターベース CRD インストールの場合

オペレーターを介して管理されるインストールの場合、HybridControlPlane リソース内のバージョン仕様を更新する必要があります。

1. カタログから edbpgai-bootstrap プロジェクトを更新します。

Openshift の例

- OpenShift Console で **Operators** > **OperatorHub** に移動します。
- **EDB Postgres AI** を検索します。
- 画面のウィザードに従って、edbpgai-bootstrap プロジェクトに展開します。

1. HybridControlPlane リソースを編集します。

```
kubectl edit hybridcontrolplane <resource-name>
```

<resource-name> は、インストール中に割り当てた名前です。

1. YAML エディターで、spec.version または同等のバージョンフィールドを見つけて、2026.1.0 に更新します。

2. 変更を保存します。リソースにアノテーションを付けることにより、調整を強制します。

```
kubectl annotate hybridcontrolplane <resource-name> --overwrite edbpgai.com/ready-for-
↪upgrade=true
```

1. オペレーターはバージョンの変更を検出し、ローリング更新を開始します。次の方法で進行状況を監視できます。

```
kubectl get hybridcontrolplane <resource-name> -w
```

シナリオスタンザを再追加しますマルチ DC 環境の場合のみ

アンインストールバグを回避するために以前にシナリオ: スタンザを削除した場合、再統合できるようになりました。プライマリハブとセカンダリの場所スポークのインストール構成を更新します。

Upgrading from 2025.11 to the 2025.12 Innovation Release

2025. 11 から 2025.12 へのアップグレード。

前提条件

- Kubernetes クラスター kubectl 、helm への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは values.yaml ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール
- helm 最新バージョンを推奨
- cmctl Cert-Manager CLI) - _2025.11+_に必要

アップグレード 2025.11 → 2025.12

このリリースでは、新しい認証シークレットを導入し、データ移行サービス DMS 構成を更新し、いくつかのビーコンパラメーターを非推奨にします。

Fernet シークレットを作成する

2025.12 チャートをインストールする前に 、dex-fernet-key を手動で作成する必要があります。

```
FERNET_KEY=$(head -c 32 /dev/urandom | base64 | tr +/ -_)

kubectl create secret generic dex-fernet-key \
  --from-literal=fernet-key="$FERNET_KEY" \
  -n upm-dex
```

values.yaml の更新

1. **DMS ドメインを ``values.yaml`` . に追加します**

新しいグローバルパラメーター dms_domain_name を追加します。

```
parameters:
global:
  dms_domain_name: <your-hm-dms-domain-name>
```

1. **非推奨のフィールドを削除します。**

values.yaml から次のフィールドを削除します。

- beaconAgent.plaintext
- beaconAgent.beaconServerAddress
- beaconAgent.tlsInsecure
- beaconAgent.inCluster

重要なインフラストラクチャの回避策

Postgres ワークロードが正しくスケジュールされるように保証には、特定のノードプールにテイントを追加する必要があります。

- **テイントキー** edbaiplatform.io/postgres
- **値** 'true'
- **効果** NoSchedule

アップグレードを実行する

1. **Helm リポジトリを更新します。**

```
helm repo update
```

1. **アップグレードを実行します。**

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --version 2025.12.0 \
  --values values.yaml \
  --namespace edbpgai-bootstrap \
  --wait
```

アップグレード後の修正

1. **SPIRE を再初期化します信頼ドメインが変更された場合。**
2. 12 では、 trust_domain パラメーターが削除され、デフォルトの portal_domain_name に設定されます。バージョン 1.3.0 で設定した trust_domain がポータルドメインと異なる場合、SPIRE は中断します。

- 修正

```
kubectl scale statefulset spire-server -n spire-system --replicas=0
kubectl delete pvc -l app=spire-server -n spire-system
kubectl scale statefulset spire-server -n spire-system --replicas=1
```

1. モデルの再構成

アップグレード後、ナレッジベース KB が UI から消える場合があります。回避するには、ポータルにログインし、**Model Configuration** ページに移動して、モデル設定を再保存します。

Upgrading from 1.3.x to the 2025.12 Innovation Release

1. 3.x から 2025.11 へのアップグレード。

前提条件

- Kubernetes クラスター `kubectl` 、`helm` への管理アクセス。
- 現在のインストールに使用される HM インストール構成ファイルまたは `values.yaml` ファイルへのアクセス。現在のインストール構成の内容を保存するには

```
helm get values edbpgai-bootstrap -n edbpgai-bootstrap > values.yaml
```

- 必要なツール
- `helm` 最新バージョンを推奨
- `cmctl` Cert-Manager CLI) - _2025.11+_に必要

アップグレード 1.3.x → 2025.11

このリリースでは、認証局 CA TTL が 10 年に増加し、OpenShift での Istio 実装がリファクタリングされました。

証明書の更新 必須

内部自己署名 CA が更新されました。新しい 10 年 TTL を適用するには、証明書を再生成する必要があります。

1. `cmctl` `brew install cmctl` などをインストールします。
2. 証明書を再生成します。

```
# Global CA
cmctl renew -n cert-manager global-ca-cert

# Leaf Certificates
cmctl renew -n upm-beacon beacon-int-ca
cmctl renew -n istio-system beaco-istio-gw-cert
```

OpenShift の詳細 Istio の移行

> **注釈**
>
> Red Hat OpenShiftRHOS で実行していない場合、この手順をスキップできます。

HM は、OLM 管理の Maistra Service Mesh を RHOS で使用しなくなりました。アップグレードの前に、古いリソースを手動でクリーンアップする必要があります。

1. オペレーターをスケールダウンします。

```
kubectl scale deployment --replicas 0 edb-hcp-operator-controller-manager -n edbpgai-
↪bootstrap
```

1. 古い Maistra リソースを削除します。

```
kubectl delete servicemeshmemberrolls.maistra.io -n istio-system default
kubectl delete servicemeshcontrolplanes.maistra.io -n istio-system upm
```

1. メンバーの一括削除

```
kubectl get servicemeshmember -A -ojsonpath={range .items[*]}{"kubectl delete␣
↪servicemeshmember "}{@.metadata.name}{" -n "}{@.metadata.namespace}{"\n"} {end} |␣
↪while read -r line; do eval $line; done
```

> **重要**
>
> この時点ではオペレーターを再開しないでください。アップグレードを続行します。 Helm アップグレードは、オペレーターを自動的にスケールアップします。

アップグレードを実行する

1. リポジトリを更新します。

```
helm repo update
```

1. アップグレードを実行します。

```
helm upgrade --install edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap \
  --version 2025.11.0 \
  --values values.yaml \
  --namespace edbpgai-bootstrap \
  --wait
```

5.3.4.4 Hybrid Manager backup and disaster recovery (DR)

Hybrid ManagerHM インストールとクラスターのディザスターリカバリー DR について説明します。このセクションには、次のトピックが含まれています。

- *Multi-DC for high availability* ディザスターリカバリーの手順と要件の概要を提供します。
- *Configuring backups to support disaster recovery (DR)* 目標復旧時間 RTO と目標復旧ポイント RPO 時間を考慮しながら、DR のバックアップ要件を満たしていることを確認する方法について説明します。
- *Hybrid Manager disaster recovery* Velero を使用して HM インスタンスと HM 管理の Postgres クラスターのディザスターリカバリー実行を実行するためのステップバイステップガイドを提供します。これには、バックアップから HM コントロールプレーンとデータプレーンの復元と、復元プロセスの検証が含まれます。

5.3.4.4.1 Disaster recovery overview

災害はハイブリッドマネージャー HM に影響を与え、使用できなくなる可能性があります。たとえば、基になるクラウドサービスプロバイダー CSP リージョンの使用不可、またはハードウェアを使用不可にするデータセンターの停止が発生する可能性があります。

ディザスターリカバリー DR オプションを使用すると、使用可能なバックアップからデータベースをポイントインタイムに復元できます。これには、壊滅的な障害の後に、**コントロールプレーン** HM コンソールと HM コンポーネントと**データプレーン** HM 管理の Postgres クラスターを復旧することが含まれます。

Postgres データベースのリアルタイムフェイルオーバーを提供する高可用性 HA とは異なり、DR は、元の HM 環境が復旧できない場合に開始される手動復元プロセスです。

リカバリーシナリオ

- **HM を元の場所に復元します** 2 つのデータセンター DC1、DC2 があり、HM は DC1 で実行されています。HM をオブジェクトストレージから DC1 に復元する必要があります。
- **HM を別の場所に復元する** 2 つのデータセンター DC1、DC2 があり、HM は DC1 で実行されています。HM をオブジェクトストレージから DC2 に復元する必要があります DC1 が予期しない期間ダウンしているため。代替の場所 DC2 は、DC1 が使用するのと同じまたはレプリケートされたオブジェクトストレージにアクセスする必要があります。

コアコンポーネント

DR は 3 つのコンポーネントに依存しています。

1. **Velero** ―Kubernetes リソース定義、シークレット、および構成マップをバックアップする HM インストールに含まれるオープンソースツール。HM インストールの構成と状態、およびそれを使用して作成された Postgres データベースクラスターの定義をキャプチャします。
2. **オブジェクトストレージ** ―HM インストールとその Postgres データベースクラスターの Velero バックアップを保持する S3 互換バケット AWS S3、GCS など。
3. **HM コンソール** ―これを使用して、Postgres データベースクラスターを再プロビジョニングし、データを戻します。

スコープ 2 レイヤー復元戦略

DR を理解するには、復旧するデータの 2 つのレイヤーを区別する必要があります。

Layer	Content	Recovery method
Control plane	HM console and its applications, general settings, user accounts, and database definitions.	Velero restore, reinstalls Kubernetes definitions into the new HM installation
Data plane	Actual Postgres tables, rows, and indexes from database clusters created with HM.	HM console restore, leverages Point-in-Time Recovery (PITR) to roll data forward.

両方のデータレイヤーのバックアップは、Velero で管理される S3 互換バケットに保存されます。HM の初期構成とインストール中に、バックアップの場所と頻度を定義します。

データリカバリーの考慮事項

- **移行** DR 手順では、アクティブな移行機能は復元されません。HM 内部の `transporter-db` および `migration-db` データベースが復元され、監査と記録保持のための移行履歴を保存しますが、アクティブでディザスターの影響を受けた移行を再作成する必要があります。
- **クエリの診断と推奨事項** クラスタービューの HM コンソール> **クエリ診断** および **推奨事項**タブに表示されるデータは、ディザスター後は復旧できません。このデータは一時的なものであるため、バックアップから意図的に除外されています。HM が復元されると、Postgres クラスターの現在の状態を反映する新しいデータの生成が自動的に開始されます。
- **モニタリング**履歴メトリック Thanos データクラスタのビューの HM コンソール> **モニタリング**タブに表示されますは、回復可能です。メトリックは、ディザスターリカバリー手順に従って、自動的に再表示されます。

ワークフロー 停止から復旧まで

ディザスターが発生すると、リカバリは次のシーケンスに従います。

1. **インフラストラクチャの準備** —正常なリージョンまたはデータセンターに新しい Kubernetes クラスターを展開し、それに HM を展開します。
2. **環境の復元** —Velero を使用して、バケットから HM 構成と Postgres データベースクラスター定義をプルします。これにより、HM コンソールと以前のデータベースのレコードが戻ります。
3. **データベースの再プロビジョニング** —復元された HM コンソールにログインします。データベースは **Deleted** または **Missing** として表示されます。 HM コンソールの **Restore** ファンクションを使用して、オブジェクトストレージ内の元のデータをポイントする新しい Postgres データベースクラスターをスピンアップします。

このアーキテクチャが機能するには、災害が発生する **前**に以下が当てはまる必要があります。

- **オフサイトバックアップ** S3 互換バケットは別のリージョンまたは物理サイトにレプリケートする必要があります。データセンターでバケットが死んだ場合、DR は不可能です。詳細については、場所

をまたがるオブジェクトストレージ を参照してください。

- **一貫した資格情報**古いバックアップを「参照」するには、新しい HM インスタンスが元のオブジェクトストレージバケットへの同じ IAM 権限またはシークレットアクセスを持っている必要があります。

Configuring backups to support disaster recovery (DR) 。

5.3.4.4.2 Configuring backups to support disaster recovery (DR)

ディザスターリカバリー DR の成功は、堅牢なバックアップ戦略に直接関係します。HM はデフォルトのバックアップ構成を提供しますが、補助スケジュールを評価および実装して、環境が特定の目標復旧時間 RTO および目標復旧ポイント RPO 要件と一致していることを確認する必要があります。HM バックアップは Velero で処理されます。

RTO と RPO

復元を実行する機能、および関連する RTO および RPO は、バックアップの頻度とサイズによって異なります。

- **RTO** は、新しい HM インスタンスを展開し、利用可能なバックアップから HM コントロールプレーンとデータプレーン Postgres データベースクラスターを復元するのにかかる時間によって決定されます。RTO は、Postgres クラスターのサイズ、Postgres クラスターの数、および HM コントロールプレーンデータのサイズの影響を受けます。
- **RPO** は、バックアップの頻度とバックアップライフサイクルポリシーによって決定されます。Postgres クラスターの定義などの重要なデータは、Kubernetes オブジェクトとして保存され、Velero バックアップに含まれます。デフォルトでは、このバックアップは、デフォルトの `velero-backup-kube-state` スケジュールで定義されているように、毎日 23 時に実行されます。RPO でより頻繁なバックアップが必要な場合は、追加のカスタムバックアップスケジュールを作成して、デフォルトのスケジュールを補完します。

> **注釈**
>
> RTO および RPO 目標は組織のサービスレベル要件によって定義されますが、災害中に達成される実際の時間とデータリカバリポイントは、データ量とこの手順の手動の性質によって異なります。定期的なディザスターリカバリー演習を実行して、これらの手動手順で定義した RTO および RPO 目標を達成できることを検証することを強くお勧めします。

インストール中に、以下を保存する HM インスタンスの **その他の *S3* 互換ストレージ** を構成します。

- 内部バックアップ HM コントロールプレーンデータ
- Postgres バックアップ Postgres データベースバックアップまたはデータプレーンデータ

バケットには、これらのバックアップのデフォルトの保存場所があります。ただし、HM は、同じバケット内のカスタム保存場所**管理対象ストレージ場所**へのバックアップデータの保存をサポートしています。カスタム格納場所を使用する場合、リージョンの停止中に利用できるように、これらのディレクトリがクロスリージョンレプリケーション戦略に含まれていることを確認します。

このデータはすべて、災害後に利用できる必要があります。データの重要性と復旧できる災害のレベルによっては、HM インスタンスが存在する CSP リージョンまたは物理データセンターの外部にこのデータを

レプリケートする必要があります。

> **Tip**
>
> バケットを使用する場合、クロスリージョンレプリケーションを使用してレプリケーションを実現できます。例、AWS S3 バケットの場合、 cross-region replication を参照してください。

デフォルトのバックアップ構成

HM は、二層バックアップ戦略を採用して、コントロールプレーンとデータプレーンの両方を保護します。

- **Postgres データベースデータ** —継続的なバックアップを介して管理されます。これにより、クラスターが復元されると、データを特定のポイント・インタイムに復元できますが、バックアップライフサイクルポリシーによってのみ制限されます。
- **HM 構成データ** —Velero を介して管理。これにより、Postgres クラスターの定義を含む環境の「ブループリント」がキャプチャされます。デフォルトでは、これは `velero-backup-kube-state` スケジュールを介して毎日 23:00 に発生します。

RPO がクラスター定義のより頻繁なスナップショットを必要とする場合たとえば、クラスターを頻繁に作成または削除する場合、カスタム Velero スケジュールを作成できます。

> **危険**
>
> HM ソフトウェアの更新によって上書きされる可能性があるため、デフォルトのスケジュールは変更しないでください。

カスタムバックアップスケジュールの追加

次のコマンドは、デフォルトのシステムバックアップに加えて、6 時間ごとに実行される補助バックアップスケジュールを作成します。これにより、ビルトインの `velero-backup-kube-state` スケジュールを妨げずに、メタデータのスナップショットをより頻繁に作成できます。実際の Postgres データは、オブジェクトストレージの継続的なバックアップストリームによって既に保護されているため、`--snapshot-volumes=false` が使用されることに注意してください。

```
velero create schedule <backup-name> --schedule="0  */6 " --include-namespaces=*  --
↪include-resources=* --snapshot-volumes=false --ttl=168h --skip-immediately=false
```

バックエンドでバックアップスケジュールがどのように構成されたかを確認するには、次のコマンドを実行します。

```
velero get schedules <backup-name> -o yaml | yq .spec
```

これは、次の出力を返すはずです。

```
schedule: 0 */6
skipImmediately: false
template:
```

(次のページに続く)

(前のページからの続き)

```
  csiSnapshotTimeout: 0s
  hooks: {}
  includedNamespaces:
    - "*"
  includedResources:
    - "*"
  itemOperationTimeout: 0s
  metadata: {}
  snapshotVolumes: false
  ttl: 168h0m0s
useOwnerReferencesInBackup: false
```

次のトピック

Hybrid Manager disaster recovery 。

5.3.4.4.3 Hybrid Manager disaster recovery

ディザスターリカバリー DR 手順は、HM インストールと HM 管理の Postgres クラスターを復旧するために実行する必要がある一連の手動手順として定義されます。

> **警告**
>
> 組織の DR 手順を有効にするには、常にテストと更新を行う必要があります。

始める前に

DR 手順を開始する前に、次のことを理解していることを確認します。

- *Multi-DC for high availability*
- *Configuring backups to support disaster recovery (DR)*

前提条件

- 新しい HM インスタンスが展開および実行されています。障害が発生したか、使用できなくなった古いインスタンスと同じバージョンを実行している必要があります。
- 古い使用できない HM インスタンスでクラスターをビルドするために使用されるコンテナイメージは、新しいもので利用できます。

必要なツール

次のツールがワークステーション環境または *1.* **管理ワークステーション** *Bastion* **ホスト** で利用できることを確認します。

- Velero CLI
- `edbctl` EDB ハイブリッドマネージャー CLI - *Hybrid Manager CLI - edbctl* を参照してください

- jq コマンドライン JSON プロセッサー
- yq コマンドライン YAML プロセッサー

1. 新しい HM インスタンスでバックアップを利用できるようにする

最初のステップでは、破損した HM インスタンスのバックアップを新しい HM インスタンスのリンクストレージ新しいバケットにコピーすることにより、使用できない HM インスタンスのバックアップ「古いバックアップ」に新しい HM インスタンスから到達できるようにします。

1. 新しいバケットのバケット名、バックアップ ID、およびリージョンを取得します。これらを、このガイド全体のコマンドで使用する環境変数として保存します。

> **重要**
>
> これらの変数はセッション固有です。新しいターミナルタブを開くか、セッションがタイムアウトすると、変数が失われ、後続のコマンドは失敗します。これらの値の再入力を回避するには、これらの export コマンドを小さなシェルスクリプト `dr-env.sh` などに保存できます。次に、そのファイルを新しいターミナルウィンドウで「ソース」して、環境をすぐにリロードできます .. Important:: edb_notranlate_0

古いバケット値を取得するにはどうすればよいですか?

古いバケット値を取得するには

1. CSP のコンソール/ダッシュボードに移動します > Buckets.
2. 古い HM インスタンスのバックアップにリンクされているバケットを見つけて選択します。
3. edb-internal-backups フォルダーを参照します。そのフォルダー内には、バックアップ ID を持つサブフォルダーがあります。4be7a1c8c9f0.

EKS の例

これは、EKS に展開された HM インスタンスの環境変数を設定する例です。

```
export OLD_BUCKET=eks-1105143903-2511-edb-postgres
export OLD_BACKUP_ID=a7462dbc7106
export NEW_BUCKET=eks-1105155418-2511-edb-postgres
export NEW_REGION=eu-west-3
```

2. 古いバケットから新しいバケットにデータをコピーするには、最初にソースフォルダーとターゲットフォルダーの名前を見つけてメモする必要があります。次のフォルダーとそのコンテンツをコピーする必要があります。
   - **内部 EDB バックアップフォルダー** - 古いバケット edb-internal-backups/<random-string> の内部バックアップフォルダーは、新しい HM インスタンスでは異なる<random-string> .
   - **Postgres クラスターバックアップフォルダー** — customer-pg-backups .
   - **定義されたカスタムストレージ場所に対応するフォルダー** — HM コンソールで **管理ストレージロケーション**を利用する場合 Postgres クエリのオフロードなど、対応することを確認する必要がありますフォルダーが古い S3 互換バケットから新しいバケットにコピーされます。定義は

Velero を介して復元されますが、これらのカスタムフォルダー内の実際のデータは、新しいターゲットバケットに手動で移行する必要があります。

3. 優先するツールを使用して、古いバックアップを新しいバケットにコピーします。次に、クラウドサービスプロバイダーの CLI を使用してバケット間でデータを移動する例をいくつか示します。

```
aws s3 cp --recursive s3://${OLD_BUCKET}/edb-internal-backups/${OLD_BACKUP_ID} s3://$
↪{NEW_BUCKET}/edb-internal-backups
aws s3 cp --recursive s3://${OLD_BUCKET}/customer-pg-backups s3://${NEW_BUCKET}/
↪customer-pg-backups
```

追加の **管理対象ストレージ場所**を構成している場合、同じ方法を使用してこれらのフォルダーをコピーします。

```
gcloud storage cp gs://${OLD_BUCKET}/edb-internal-backups/${OLD_BACKUP_ID} gs://$
↪{NEW_BUCKET}/edb-internal-backups --recursive
gcloud storage cp gs://${OLD_BUCKET}/customer-pg-backups gs://${NEW_BUCKET}/customer-
↪pg-backups --recursive
```

追加の **管理対象ストレージ場所**を構成している場合、同じ方法を使用してこれらのフォルダーをコピーします。

4. 新しいカスタムリソース定義を作成し、新しい HM インスタンスに適用することにより、新しい HM インスタンスにコピーしたバックアップをロードします。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  annotations:
    appliance.enterprisedb.com/s3-prefixes: edb-internal-backups/velero
  labels:
    appliance.enterprisedb.com/storage-credentials: bound
  name: recovery
  namespace: velero
spec:
  accessMode: ReadOnly
  config:
    insecureSkipTLSVerify: "false"
    region: ${NEW_REGION}
    s3ForcePathStyle: "true"
  default: false
  objectStorage:
    bucket: ${NEW_BUCKET}
    prefix: edb-internal-backups/velero
  provider: aws
EOF
```

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  annotations:
    appliance.enterprisedb.com/s3-prefixes: edb-internal-backups/velero
  labels:
    appliance.enterprisedb.com/storage-credentials: bound
  name: recovery
  namespace: velero
spec:
  accessMode: ReadOnly
  credential:
    key: gcp
    name: gcs-credentials
  default: false
  objectStorage:
    bucket: ${NEW_BUCKET}
    prefix: edb-internal-backups/velero
  provider: gcp
EOF
```

5. 新しい保存場所が利用できることを確認します。

```
velero get backup-locations
```

ステータスが Available でない場合は、Velero ポッドログで S3 バケットの権限エラーを確認します。

6. バックアップも利用できることを確認します。

```
velero get backups --selector velero.io/storage-location=recovery
```

7. 復元するバックアップを選択します。複数のバックアップを使用できるため、ニーズに最も適したものを選択してください。災害が発生する前の最新のバックアップ。リストアには両方が必要であるため、Velero バックアップ名と日付と時刻 UTC をメモします。たとえば、

```
NAME                                        STATUS      ERRORS   WARNINGS   CREATED
                    EXPIRES   STORAGE LOCATION   SELECTOR
velero-backup-kube-state- **20241216154403**    Completed   0        0          2024-
↪12-16 16:44:03 0100 CET   5d        recovery           \<none\>
```

> **注釈**
>
> タイムスタンプ値は、以下の手順では*リカバリ日*と呼ばれます。

8. (オプショナル)HM を使用して AI ワークロードを管理していた場合、例 GenAI Builder を使用して、オブジェクトストアファイルと CORS 構成も古いバケットから新しいバケットにコピーします。

```
export OLD_BUCKET_DATALAKE=<old-bucket-name>
export NEW_BUCKET_DATALAKE=<new-bucket-name>
```

```
# Copy data lake objects from old bucket to new bucket
aws s3 cp --recursive s3://${OLD_BUCKET_DATALAKE}/ s3://${NEW_BUCKET_DATALAKE}/
# Copy CORS configuration from old bucket to new bucket
aws s3api get-bucket-cors --bucket ${OLD_BUCKET_DATALAKE} --output json > cors-config.
↪json
aws s3api put-bucket-cors --bucket ${NEW_BUCKET_DATALAKE} --cors-configuration file://
↪cors-config.json
```

```
# Copy data lake objects from old bucket to new bucket
gcloud storage cp "gs://${OLD_BUCKET_DATALAKE}/**" gs://${NEW_BUCKET_DATALAKE}/ --
↪recursive
# Copy CORS configuration from old bucket to new bucket
gcloud storage buckets describe gs://${OLD_BUCKET_DATALAKE} --format="json" | jq .
↪cors_config > cors-config.json
gcloud storage buckets update gs://${NEW_BUCKET_DATALAKE} --cors-file=cors-config.json
```

2. リカバリ手順

HM 内部データベースの復元

古いバックアップが新しいバケットで利用できるようになったら、HM 内部データベースを復元できます。これらは HM が使用するバックエンドサービスであり、HM インスタンスを完全に復元するために必要です。使用している HM バージョン、および展開したインストールシナリオによっては、データベースのリストが異なる場合があります。

1. これらのプロセスを簡素化するには、新しい HM インストールをポイントする kubeconfig で次のスクリプトを実行します。

patch-clusters.sh

スクリプトの詳細

このパッチスクリプトは以下を処理します。

- 2 つのディレクトリを生成しながら、HM 内部データベースクラスターマニフェストを YAML ファイルに保存します。-新しい HM インストールのデータベースクラスターの現在の状態を含む 1 つのディレクトリ `old-cluster-configs` インストール後のデフォルト構成。 - 同じファイルを使用した `new-cluster-configs` と呼ばれる別のファイル。スクリプトは必要なパッチを実行して、HM 内部データベースがバックアップからのデータの使用を開始します。

2. すべての HM 内部データベースクラスターのリコンシリエーションを一時停止して、オペレーターがデフォルトで再作成せずに、データベースクラスターの古いカスタムリソース定義 CRD を安全に削除できるようにします。

```
for CLUSTER in $(kubectl get clusters.postgresql.k8s.enterprisedb.io -A -o json | jq -
↪rc .items[].metadata | select((.name | test("^p-") | not) and (.name != "stats-
↪collector-db")) | {namespace: .namespace} | uniq)
do
  NAMESPACE=$(echo "${CLUSTER}" | jq -rc .namespace)
  if [[ "${NAMESPACE}" == "upm-system-db" ]]
  then
    COMPONENT="upm-app-db"
  else
    COMPONENT="${NAMESPACE}"
  fi
  edbctl hm suspend-component "${COMPONENT}"
done
```

3. コンポーネントが正しくサスペンドされたことを確認します。

```
edbctl hm status
```

4. 新しい HM インスタンスのインストール中に作成された HM 内部データベースクラスターを削除して、バックアップから復元される HM 内部データベースクラスター用の領域を確保します。

```
for CONFIG in $(find new-cluster-configs -type f)
do
    kubectl delete -f $CONFIG
done
```

5. 新しい HM インスタンスのインストール中に作成されたバックアップ領域をクリーンアップして、復元する古いバックアップとの混同を避けます。

```
for CONFIG in $(find new-cluster-configs -type f)
do
    NAME=$(yq .metadata.name $CONFIG)
    NAMESPACE=$(yq .metadata.namespace $CONFIG)
    # Try to get PREFIX from cluster config first, fallback to ObjectStore if it fails
    PREFIX=$(yq .spec.backup.barmanObjectStore.destinationPath | downcase $CONFIG 2>/
↪dev/null)
    if [ -z "$PREFIX" ] || [ "$PREFIX" = "null" ]; then
      # destinationPath doesnt exist in the cluster config (after CNPG-I barman␣
↪plugin migration)
      # Get it from the ObjectStore resource instead
      PREFIX=$(kubectl -n $NAMESPACE get objectstores.barmancloud.cnpg.io $NAME -o␣
↪yaml | yq .spec.configuration.destinationPath | downcase)
    fi
    aws s3 rm --recursive ${PREFIX}/${NAME}
done
```

```
for CONFIG in $(find new-cluster-configs -type f)
do
    NAME=$(yq .metadata.name $CONFIG)
    NAMESPACE=$(yq .metadata.namespace $CONFIG)
    # Try to get PREFIX from cluster config first, fallback to ObjectStore if it fails
    PREFIX=$(yq .spec.backup.barmanObjectStore.destinationPath | downcase $CONFIG 2>/
↪dev/null)
    if [ -z "$PREFIX" ] || [ "$PREFIX" = "null" ]; then
      # destinationPath doesnt exist in the cluster config (after CNPG-I barman␣
↪plugin migration)
      # Get it from the ObjectStore resource instead
      PREFIX=$(kubectl -n $NAMESPACE get objectstores.barmancloud.cnpg.io $NAME -o␣
↪yaml | yq .spec.configuration.destinationPath | downcase)
    fi
    gsutil -m rm -r ${PREFIX}/${NAME}
done
```

6. hm-portal-bootstrap シークレットをバックアップのバージョンに置き換えます。

> **注釈**
>
> この手順は、*2026.2* 以降の HM のバージョンにのみ適用されます。

新しい HM インスタンスに対して `kubectl get secret hm-portal-bootstrap -n default` コマンドを実行し、シークレットが存在しない場合は、 `2026.2` より前のバージョンの HM のバージョンを使用しているため、この手順をスキップできます。

新しい HM インスタンスの hm-portal-bootstrap シークレットは、古い HM インスタンスにあったバージョンに置き換える必要があります。このシークレットには、機密構成データの暗号化に使用される Fernet キーと、初期ブートストラップアクセスに使用される静的パスワードが含まれています。古い環境の正しい Fernet キーがないと、HM は、次の手順で復元されるデータの一部を復号化することはできません。

この手順には、 `default` 名前空間で見つかった単一のシークレットの復元が含まれます。Velero は、通常、特定の名前空間の特定のタイプのすべてのリソースを復元するために使用されるため、これを十分にサポートしていません。したがって、この復元を実行する唯一の方法は、Velero バックアップ全体をダウンロードし、特定のファイルを抽出することです。

特定のファイルを選択するには、最初に使用するバックアップを選択する必要があります。次のコマンドを使用して、使用可能なすべてのバックアップをリストすることから始めます。

```
velero get backups -o json --selector velero.io/storage-location=recovery \
| jq -rc (["Name", "Timestamp"]), (.items // [.] | .[] | [.metadata.name, .metadata.
↪creationTimestamp]) | @tsv \
| column -t -s "$(printf \t)"
```

このリストからバックアップを選択し、通常は最新のファイルを選択し、選択したファイルの名前を指す次の変数をエクスポートします。

```
export BACKUP_NAME=<selected-name>
```

次に、完全なバックアップをダウンロードして抽出します。

```
mkdir "${BACKUP_NAME}"
velero backup download "${BACKUP_NAME}" -o "${BACKUP_NAME}/backup.tar.gz"
tar -xzf "${BACKUP_NAME}/backup.tar.gz" -C "${BACKUP_NAME}"
```

ファイル${BACKUP_NAME}/resources/secrets/namespaces/default/hm-portal-bootstrap.json が存在することを確認します。次に、既存の hm-portal-boostrap シークレットを削除し、ダウンロードしたばかりのバックアップからのシークレットに置き換えます。

```
kubectl delete secret hm-portal-bootstrap -n default
kubectl apply -f "${BACKUP_NAME}/resources/secrets/namespaces/default/hm-portal-
↪bootstrap.json"
```

次のコマンドを使用して、シークレットが存在することを確認します。

```
kubectl get secret hm-portal-bootstrap -n default
```

また、これらのファイルは再度必要ないため、ダウンロードしたバックアップのローカルコピーを削除できます。

```
rm -rf "${BACKUP_NAME}"
```

7. バックアップデータで再作成するすべての HM 内部データベースクラスターの YAML ファイルを適用します。

```
for CONFIG in $(find new-cluster-configs -type f)
do
    kubectl apply -f $CONFIG
done
```

kubectl get clusters -A を使用して、リストアの進行状況を監視できます。

HM サービスを再起動します

すべての HM 内部データベースクラスターが正常に復元され、正常な状態が報告されたら、管理サーバーのこの 1 回の再起動を実行して HM コンソールを更新します。

```
kubectl delete pods $(kubectl get pods -n upm-beaco-ff-base | grep ^accm-server | awk
↪{print $1}) -n upm-beaco-ff-base
```

新しいポッドが **Running** 状態に到達するまで待ちます。この時点で、HM コンソールは利用できますが、HM 管理の Postgres クラスターはまだ表示されません。

Velero プラグインの構成

Velero プラグインは、復元中に Kubernetes リソースの変換を処理します。最も重要なのは、Postgres クラスターがデータ復旧を手動でトリガーできる状態で復元されることを保証します。

1. 使用可能なバックアップをリストし、優先する復旧ポイントの **名前** と**タイムスタンプ** をメモします。

```
velero get backups -o json --selector velero.io/storage-location=recovery \
| jq -rc (["Name", "Timestamp"]), (.items // [.] | .[] | [.metadata.name, .metadata.
↪creationTimestamp]) | @tsv \
| column -t -s "$(printf \t)"
```

2. 環境変数をエクスポートします。

```
export BACKUP_TIMESTAMP=<recovery-date in YYYY-MM-DDTHH:MM:SSZ format>
# The BACKUP_NAME may already be available in your terminal from an earlier step.
export BACKUP_NAME=<selected-name>

# These environment variables should already be available in your terminal:
export OLD_BUCKET=<old-bucket-name>
export NEW_BUCKET=<new-bucket-name>
```

重要

BACKUP_TIMESTAMP は、前の手順で見つかった正確な ISO タイムスタンプである必要があります。

例 *2024-12-16T15:44:03Z* .. ::

3. *ConfigMap* を作成して適用して、Velero プラグインを構成します。

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-plugin-for-edbpgai
  namespace: velero
  labels:
    velero.io/plugin-config: ""
    enterprisedb.io/edbpgai-plugin: RestoreItemAction
data:
  # configure disaster recovery mode, so restored items are transformed as needed
  drMode: "true"
  # configure a date corresponding to the velero backup date. Note the format!
  drDate: "${BACKUP_TIMESTAMP}"
  # old and new buckets for internal custom storage locations
  oldBucket: ${OLD_BUCKET}
  newBucket: ${NEW_BUCKET}
EOF
```

リソースを復元する

1. 次の Velero 復元を適用して、 **管理ストレージ場所**を復元します。これには、デフォルトの managed-devspatcher ロケーションと追加のカスタム定義のロケーションが含まれます。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-1-storagelocations
  namespace: velero
spec:
  backupName: "${BACKUP_NAME}"
  includedResources:
   - storagelocations.biganimal.enterprisedb.com
  includeClusterResources: true
  labelSelector:
    matchLabels:
      biganimal.enterprisedb.io/reserved-by-biganimal: "false"
EOF
```

2. 次の Velero 復元リソースマニフェストを構成および適用して、クラスターラッパーを復元します。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-2-clusterwrappers
  namespace: velero
spec:
  backupName: "${BACKUP_NAME}"
  includedResources:
   - clusterwrappers.beacon.enterprisedb.com
  restoreStatus:
    includedResources:
     - clusterwrappers.beacon.enterprisedb.com
EOF
```

3. 復元の進行状況を監視します。次のカスタムリソース CR は clusterwrappers に依存しているため、最初に clusterwrappers が復元されるまで待機する必要があります。対応する clusterwrapper が見つからない場合、HM は他の CR を削除する可能性があります。

```
velero get restore restore-2-clusterwrappers
```

4. クラスターラッパーが復元された後、次の Velero リソースマニフェストを構成および適用して、バックアップラッパーを復元します。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-3-backupwrappers
  namespace: velero
spec:
  backupName: "${BACKUP_NAME}"
  includedResources:
   - backupwrappers.beacon.enterprisedb.com
  restoreStatus:
    includedResources:
     - backupwrappers.beacon.enterprisedb.com
EOF
```

5. 次の Velero リソースマニフェストを構成および適用して、Griptape、Lakekeeper、および Dex シークレットを復元します。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-4-required-secrets
  namespace: velero
spec:
  backupName: "${BACKUP_NAME}"
  includedNamespaces:
  - upm-griptape
  - upm-lakekeeper
  - upm-dex
  includedResources:
  - secrets
  includeClusterResources: false
EOF
```

6. オプショナル AI ワークロードを実行している場合、次の Velero 復元リソースマニフェストを構成および適用して、kserve リソースを復元します。

```
kubectl apply -f - <<EOF
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-5-kservecrs
  namespace: velero
spec:
  backupName: "${BACKUP_NAME}"
```

(次のページに続く)

(前のページからの続き)

```
  includedResources:
  - clusterservingruntimes.serving.kserve.io
  - inferenceservices.serving.kserve.io
EOF
```

7. すべての復元を監視し、完了するまで待ちます。

```
velero get restores
```

3. Postgres クラスターを復元する

クラスターメタデータは復元されましたが、HM 管理の Postgres クラスターを手動で再プロビジョニングしてデータにリンクする必要があります。

1. HM コンソールで、 Databases セクションに移動します。元のクラスターが**削除** のステータスでリストされます。
2. 目的のクラスターを選択し、[**Restore**] ボタンを見つけます。プロンプトに従って、新しいクラスターを作成します。このプロセス中に、システムは以前のバックアップを使用して、新しいインスタンスを設定します。
3. プロビジョニングが完了したら、データが元の状態と一致することを確認します。

同じ手順を適用して、セカンダリの場所で構成した Postgres クラスターを復元できます。

> **注釈**
>
> AI コンポーネント Launchpad セクションの GenAI Builder UI などは、復元が開始されると HM コンソールに自動的に再表示されます。コンテナイメージとプロファイルのサイズが大きいため、同期には時間がかかる場合があります。

これで復元手順は完了しました。リカバリーを成功させるには、データの完全性を確認することをお勧めします。新しくプロビジョニングされた Postgres クラスターにログインし、いくつかのテストクエリーを実行して、データが最新でアクセス可能であることを確認します。

> **Tip**
>
> DR ドリルの一部としてこれを実行する場合、データベースと AI レイヤーの両方の合計「復元時間」TTR を内部的に文書化して、復旧目標 RTO を調整するのに役立ちます。 !!!

5.3.4.5 Enabling the Migration Portal AI Copilot

Hybrid ManagerHM を展開した後、外部データベーススキーマを移行するための Launchpad ツール Migration Portal は、すぐに機能します。

ただし、移行ポータル内のスキーマ移行の互換性問題の解決に役立つチャットボットツール AI Copilot はデフォルトで有効になっていないため、有効にするには追加の構成が必要です。

OpenAI、Azure OpenAI、または代替 AI ベンダーを使用して AI コパイロットを構成できます。または、EDB Agent Factory 機能を使用して HM で以前に作成したセルフホストモデルと連携するように AI コパイロットを構成できます。これにより、サードパーティベンダーのすぐに使用できるモデルを使用したり、データと環境を完全に制御したりできる柔軟性が得られます。

図 44: Model options for the Migration Portal AI Copilot

5.3.4.5.1 サードパーティモデルクラウド

ワークフロー移行ポータルの使用中に、移行ポータル AI コパイロットにプロンプトを送信して、スキーマ変換に関する情報とヘルプを取得できます。AI コパイロットは、インターネットを介して選択した外部のサードパーティクラウドベンダー OpenAI などと通信します。サードパーティベンダーは、独自のモデルを使用してプロンプトを処理し、回答を生成します。

モデルサードパーティベンダーは、チャット完了モデルや埋め込みモデルを含むコア AI 機能を提供します。Azure OpenAI の場合、引き続き埋め込みモデルを展開する必要があります。

利点このアプローチでは、AI インフラストラクチャを管理する必要がなく、非常に強力で継続的に更新されるモデルへのアクセスを提供します。

考慮事項このオプションには、中断のないインターネット接続が必要です。ライセンスによっては、データはベンダーに保存される場合があります。また、チャット完了モデルと埋め込みモデルが必要とする定期的なヘルスチェック要求によって発生するコストとクエリごとの料金を検討します。

5.3.4.5.2 セルフホストモデルオンプレミス

ワークフロー移行ポータルの使用中に、移行ポータル AI コパイロットにプロンプトを送信して、スキーマ変換に関する情報とヘルプを取得できます。AI コパイロットは、HM 環境でローカルにホストされているモデルと通信します。これらのモデルは、EDB Agent Factory 機能を使用して組織が以前に作成したものです。セルフホストモデルは、プロンプトを処理し、回答を生成します。

モデル EDB Agent Factory を使用してこれらのモデルを作成および管理します。これらは、NVIDIA NIM マイクロサービスを使用して提供される Llama 3 のようなモデルです。

利点このアプローチは、安全性の高いエアギャップ環境を必要とする組織に適しています。すべてのデータ処理がインフラストラクチャ内に残り、厳格なコンプライアンスとプライバシー要件を満たします。

考慮事項 AI インフラストラクチャは自分自身が管理します。つまり、組織に AI の専門知識が必要です。また、セルフホストモデルには、サードパーティベンダーが提供するビルトインのコンテンツフィルタリングと安全対策がありません。モデルによって生成された安全でないまたは有害なコンテンツに対する責任および潜在的な法的責任は、ユーザーまたは組織にあります。また、AI ワークロードの実行に必要な高価な GPU ノードのリソース消費も考慮する必要があります。

Enabling a third-party model for the Migration Portal AI Copilot

AI Copilot のインタラクションにサードパーティのクラウドベンダーを活用するために、これらの例では OpenAI および Azure OpenAI を使用します。

前提条件

ベンダーの AI ソリューションを使用して移行ポータル AI コパイロットを展開するために環境が必要なリソースを準備します。

- 特定の AI ベンダーのサービスへのアクセス資格情報が必要です。
  - OpenAI を使用している場合、組織の ID と API キーが必要です。
  - Azure OpenAI を使用している場合、組織のエンドポイント URL と API キーが必要です。また、Azure OpenAI インスタンスで、次の展開を作成します。
    - gpt-4o 展開:
      - 展開名: `gpt-4o`
      - 推奨バージョン: `2024-11-20`
    - text-embedding-3-small 展開:
      - 展開名: `text-embedding-3-small`
      - Suggestedb
      -

AI コパイロットの有効化

Hybrid ManagerHM をインストールする前または後に、AI Copilot を有効にできます。

1. `edb-migration-copilot` が存在するかどうかを確認します。

```
kubectl get namespaces edb-migration-copilot
```

名前空間は、HM のインストール中に作成されます。HM をインストールする前に AI Copilot を有効にする場合、事前に名前空間を作成する必要があります。

2. `edb-migration-copilot` 名前空間がまだ存在しない場合は、作成します。

```
kubectl create ns edb-migration-copilot
```

3. 次の環境変数を設定して、AI コパイロットを既存の AI アカウントにリンクします。プレースホルダーを AI ベンダーのアカウント情報に置き換えます。

```
export OPENAI_ORG_ID=<openai_org_id>   # organisation id for the openai account
export OPENAI_API_KEY=<openai_api_key> # api key for accessing the openai account
```

```
export AZURE_OPENAI_ENDPOINT=<azure_openai_endpoint> # endpoint for the azure openai␣
↪instance
export AZURE_OPENAI_API_KEY=<azure_openai_api_key>   # api key for accessing the␣
↪azure openai instance
```

4. `edb-migration-copilot` 名前空間にシークレットを追加して、ベンダーの資格情報を構成します。

```
kubectl create secret generic ai-vendor-secrets \
  --namespace=edb-migration-copilot \
  --type=opaque \
  --from-literal=AI_VENDOR=OPENAI \
  --from-literal=OPENAI_ORG_ID="${OPENAI_ORG_ID}" \
  --from-literal=OPENAI_API_KEY="${OPENAI_API_KEY}"
```

```
kubectl create secret generic ai-vendor-secrets \
  --namespace=edb-migration-copilot \
  --type=opaque \
  --from-literal=AI_VENDOR=AZURE_OPENAI \
  --from-literal=AZURE_OPENAI_ENDPOINT="${AZURE_OPENAI_ENDPOINT}" \
  --from-literal=AZURE_OPENAI_API_KEY="${AZURE_OPENAI_API_KEY}"
```

5. 次の Helm 値で `migration-portal-values.yaml` というファイルを作成して、デフォルトの AI ベンダーシークレットを前の手順で作成したシークレットでオーバーライドします。

```
parameters:
  edb-migration-copilot:
    ai_vendor_secrets: ai-vendor-secrets
```

   - `ai_vendor_secrets` で参照されるシークレットは、以前に作成したシークレットの名前と一致する必要があります。

```
parameters:
  edb-migration-copilot:
    ai_vendor_secrets: ai-vendor-secrets
    chat_model: gpt-4o
```

   - `ai_vendor_secrets` で参照されるシークレットは、以前に作成したシークレットの名前と一致する必要があります。- `chat_model` で指定された LLM バージョンは、AI コパイロットの推奨バージョンであるため、デフォルトで `gpt-4o` です。

6. インストールとセットアップ 中に `helm upgrade` コマンドを実行するときにこのファイルを参照するため、移行ポータルのカスタマイズされた値が使用されます。後続の `helm upgrade` を呼び出すたびに、このオプションを使用します。それ以外の場合、値はベンダーのデフォルトでオーバーライドされます。

```
helm upgrade \
  -n edbpgai-bootstrap \
  --install \
  ...
  -f migration-portal-values.yaml \
  ...
```

Enabling a self-hosted model for the Migration Portal AI Copilot

セルフホストエージェントファクトリーモデルを使用して、AI コパイロットを提供できます。この例では、NVIDIA NIM を使用して要求を処理し、Llama 3 を使用して処理し、回答を生成します。

- チャット完了 nvidia/llama-3_3-nemotron-super-49b-v1
- 埋め込み nvidia/llama-3.2-nv-embedqa-1b-v2

> **警告**
>
> Migration Portal AI Copilot でセルフホストモデルを使用する場合、考慮すべき安全性の重要な影響があります。

OpenAI amd Azure OpenAI などのサードパーティベンダーが提供するモデルには、モデルが安全でないコンテンツに応答、生成、または寄与するリスクを削減するように設計されたコンテンツフィルタリングおよびその他の安全対策が含まれています。セルフホストモデルを使用する場合、これらの保護は存在しません。

さらに、モデルをホストしているため、不安全な動作に関連するリスクと潜在的な責任に対する責任を負います。

前提条件

セルフホストソリューションを使用して移行ポータル AI コパイロットを展開するために環境が必要なリソースを準備します。

- HM 環境への管理アクセスがあります。
- 組織は、HM *Getting started with Agent Factory on Hybrid Manager* で *chat completion and a text embeddings model* を作成しました。彼らは、環境変数として設定できる各モデルのエンドポイントを提供しています。

```
export COMPLETIONS_SVC=llama-3-3-nemotron-super-49b-v1
export EMBEDDINGS_SVC=llama-3-2-nv-embedqa-1b-v2

export COMPLETIONS_ENDPOINT=$(kubectl get inferenceservice $COMPLETIONS_SVC -o␣
↪jsonpath={.status.url})
export EMBEDDINGS_ENDPOINT=$(kubectl get inferenceservice $EMBEDDINGS_SVC -o␣
↪jsonpath={.status.url})
```

AI コパイロットの有効化

1. `edb-migration-copilot` 名前空間が存在するかどうかを確認します。

```
kubectl get namespaces edb-migration-copilot
```

名前空間は、HM のインストール中に作成されます。HM をインストールする前に AI Copilot を有効にする場合、事前に名前空間を作成する必要があります。

1. edb-migration-copilot 名前空間がまだ存在しない場合は、作成します。

```
kubectl create ns edb-migration-copilot
```

1. 次の環境変数を設定して、シークレットをモデルエンドポイントにリンクします。

```
export OPENAI_API_BASE=${COMPLETIONS_ENDPOINT}/v1
export OPENAI_EMBEDDINGS_API_BASE=${EMBEDDINGS_ENDPOINT}/v1
export OPENAI_API_KEY=<openai api key> # set to a placeholder value like `noop` if␣
↪models are deployed in a way that no key is required
```

注釈

AI コパイロットは、OpenAI 互換 API を使用して、セルフホストモデルを含むすべてのモデルと通信します。そのため、クエリを提供するために別のモデルを使用している場合でも、一部の構成パラメーターの名前に `openai` が含まれます。

1. ai-vendor-secrets シークレットを作成し、モデルのエンドポイントを指すように構成します。

```
kubectl create secret generic ai-vendor-secrets \
    --namespace=edb-migration-copilot \
    --type=opaque \
    --from-literal=AI_VENDOR=NIM \
    --from-literal=RAGCHEW_OPENAI_API_BASE="${OPENAI_API_BASE}" \
    --from-literal=RAGCHEW_OPENAI_EMBEDDINGS_API_BASE="$    {OPENAI_EMBEDDINGS_API_
↪BASE}" \
    --from-literal=OPENAI_API_KEY="${OPENAI_API_KEY}"
```

1. 次の Helm 値で migration-portal-values.yaml というファイルを作成して、デフォルトの AI ベンダーシークレットを前の手順で作成したシークレットでオーバーライドします。

```
parameters:
  edb-migration-copilot:
    ai_vendor_secrets: ai-vendor-secrets
```

1. HM インストールファイルを更新して、AI Copilot 構成を含めます。インストールに使用した YAML 値を更新するか、AI Copilot 構成パラメーターを使用して helm upgrade コマンドを実行します。

この構成ブロックは、Helm で HM のインストールに使用した values.yaml ファイル、または operator で HM のインストールに使用した CRD に追加します。

```
[...]
parameters:
  edb-migration-copilot:
    ai_vendor_secrets: ai-vendor-secrets                    # Allows the default AI␣
↪vendor secrets to be overridden by the secret created above. This  *must*  match␣
↪the name of the above secret.
```

(次のページに続く)

(前のページからの続き)

```
    chat_model: nvidia/llama-3.3-nemotron-super-49b-v1      # This must match the␣
↪model name as listed by `${COMPLETIONS_ENDPOINT}/v1/models`.
    embeddings_model: nvidia/llama-3.2-nv-embedqa-1b-v2     # This must match the␣
↪model name as listed by `${EMBEDDINGS_ENDPOINT}/v1/models`.
    chat_model_profile: llama3                              # Enables prompt rules␣
↪and other parameters which can help improve the quality of responses from chat␣
↪completion models of the llama3 family.
    embeddings_dimension: "2048"                            # This must match the size␣
↪of the vectors generated by the embeddings model.
    tokenizer: huggingface                                  # Causes tokenizers from␣
↪the Hugging Face Transformers library to be used for token counting.
    tokenizer_model: nvidia/llama-3.3-nemotron-super-49b-v1 # Sets the specific␣
↪tokenizer to use, ensuring it corresponds to a pretrained model on Hugging Face Hub.
    chat_request_max_tokens: "1024"                         # Limits the total number␣
↪of requested tokens to prevent excessively-verbose responses.
    default_similarity_limit: "10"                          # Limits the number of␣
↪contextual chunks included in queries to prevent overwhelming the model with input.
    stream_no_stop_sequence: "true"                         # Prevents OpenAI-specific␣
↪stop sequences being sent to the model. This should always be set when using NIM␣
↪models in order to avoid duplicated final chunks in response streams.
[...]
```

次の構成パラメーターを含めて helm upgrade コマンドを実行します。後続の helm upgrade を呼び出すたびに、これらのパラメーターを含めることを忘れないでください。それ以外の場合、値はデフォルトでオーバーライドされます。

```
helm upgrade \
    -n edbpgai-bootstrap \
    --install \
    [...]
    --set parameters.edb-migration-copilot.ai_vendor_secrets=ai-vendor-secrets \
    --set parameters.edb-migration-copilot.chat_model=nvidia/llama-3.3-nemotron-super-
↪49b-v1 \
    --set parameters.edb-migration-copilot.embeddings_model=nvidia/llama-3.2-nv-
↪embedqa-1b-v2 \
    --set parameters.edb-migration-copilot.chat_model_profile=llama3 \
    --set parameters.edb-migration-copilot.embeddings_dimension="2048" \
    --set parameters.edb-migration-copilot.tokenizer=huggingface \
    --set parameters.edb-migration-copilot.tokenizer_model=nvidia/llama-3.3-nemotron-
↪super-49b-v1 \
    --set parameters.edb-migration-copilot.chat_request_max_tokens="1024" \
    --set parameters.edb-migration-copilot.default_similarity_limit="10" \
    --set parameters.edb-migration-copilot.stream_no_stop_sequence="true" \
   [...]
```

1. edb-migration-copilot サービスを再起動して、システムとの新しい値の調整をトリガーします。

```
kubectl rollout restart edb-migration-copilot -n edb-migration-copilot
```

エアギャップインストールの追加構成

エアギャップ環境で実行されている場合、移行ポータル AI コパイロットは、Hugging Face Hub から事前トレーニングされたトークナイザーデータを取得しようとすると失敗します。

代わりにトークナイザーデータのローカルスナップショットを使用するように次のパラメーターを設定します。

次の `airgapped_mode` パラメーターを、以前に文書化した構成ブロック、Helm で HM のインストールに使用した `values.yaml` ファイル、またはオペレーターで HM のインストールに使用した CRD に含めます。

```
[...]
parameters:
  edb-migration-copilot:
    airgapped_mode: "true" # Forces a local snapshot of tokenizer data to be used␣
↪instead of fetching from Hugging Face Hub.
                           # Only works if `tokenizer` and `tokenizer_model` are set␣
↪to `huggingface` and
                           #`nvidia/llama-3.3-nemotron-super-49b-v1` respectively.
[...]
```

次の `airgapped_mode` パラメーターを、以前に文書化した `helm upgrade` コマンドに含めます。後続の `helm upgrade` を呼び出すときは、すべてのパラメーターを含めることを忘れないでください。それ以外の場合、値はデフォルトでオーバーライドされます。

```
helm upgrade \
    -n edbpgai-bootstrap \
    --install \
    [...]
    --set parameters.edb-migration-copilot.airgapped_mode="true" \
    [...]
```

`edb-migration-copilot` サービスを再起動します。

重要

移行ポータル AI コパイロットには、*nvidia/llama-3.3-nemotron-super-49b-v1* 事前トレーニング済みトークナイザーのトークナイザーデータのみが同梱されています。 *tokenizer_model* を他のモデルに設定して `airgapped_mode: "'true"'` を使用すると、移行ポータル AI コパイロットが失敗します。

5.3.4.6 Troubleshooting Hybrid Manager

Hybrid ManagerHM で一般的な問題をトラブルシューティングできます。

5.3.4.6.1 エラーコード

Error Code	Category	Error Message	Explanation	Potential solution
na	Installation	[upm-istio] - failed to install components: [upm-istio]	An error from bootstrap: the LoadBalancer service is pending.	Fix the loadbalancer IAM role first, then delete the LB pod to trigger re-creation.
na	Installation	upm-beaco-ff-base install failed		kubectl edit sc gp2
na	Installation	upm-thanos installation failed		
na	Installation	Failed to pull image		
na	Installation	Loki pods crash	The loki pods crashed and the installation failed.	
na	Portal Login	No healthy upstream	“No healthy upstream” is returned after the login.	
401	Portal Login	401 Access Denied	401 error returned when logging into the portal after the installation.	
500	Portal Login	HTTP Error 500	The back ingress gateway can't access incoming connections.	
na	Database Provisioning	Postgres Cluster Provisioning Stuck at X% Complete		
na	Database Provisioning	CPU and Memory range is much smaller than the nodegroup		
na	Database Update	Disk scale up	The new value is shown on cluster yaml but not applied on pvc/pv.	Edit the storageclass.

5.3.4.6.2 エラーの詳細

[upm-istio] - コンポーネントのインストールに失敗しました: [upm-istio]

詳細なエラー

```
5:03:42AM: ---- waiting on 1 changes [4/5 done] ----
5:03:42AM: ongoing: reconcile service/istio-ingressgateway (v1) namespace: istio-
↪system
5:03:42AM:  ^ Load balancer ingress is empty
{"level":"error","msg":"failed to install component: failed to execute kapp command,␣
↪error: Timed out waiting after 15m0s for resources: [service/istio-ingressgateway␣
↪(v1) namespace: istio-system], message: "}
{"level":"error","msg":"Failed to install components: [upm-istio]"}
{"level":"error","msg":"Installation failed, error: failed to install components:␣
↪[upm-istio]"}
```

```
k get svc -n istio-system
```

```
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
                           AGE
istio-ingressgateway   LoadBalancer   172.20.184.80    <pending>     80:30270/TCP,
↪443:31858/TCP,9443:32692/TCP   3h3m
istiod                 ClusterIP      172.20.146.223   <none>        15010/TCP,15012/
↪TCP,443/TCP,15014/TCP       3h3m
```

```
k describe svc istio-ingressgateway -n istio-system
```

```
Events:
  Type     Reason            Age    From     Message
  ----     ------            ----   ----     -------
  Warning  FailedBuildModel  53m    service  Failed build model due to operation␣
↪error Elastic Load Balancing v2: DescribeLoadBalancers, get identity: get␣
↪credentials: failed to refresh cached credentials, failed to retrieve credentials,
 operation error STS: AssumeRoleWithWebIdentity, https response error StatusCode: 403,
 RequestID: 07cc71c0-3113-4b4e-be7f-3de3b475c49b, api error AccessDenied: Not␣
↪authorized to perform sts:AssumeRoleWithWebIdentity
```

解像度

ロードバランサーの IAM ロールを修正し、LB ポッドを削除します。

```
k delete pod aws-load-balancer-controller-57ccd8bc77-f8lrk -n kube-system
k delete pod aws-load-balancer-controller-57ccd8bc77-j58c2 -n kube-system
k get svc -n istio-system
```

```
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP
                                                PORT(S)
       AGE
istio-ingressgateway    LoadBalancer   172.20.184.80    k8s-istiosys-istioing-
↪38d88046aa-37454e78c9648556.elb.us-east-1.amazonaws.com   80:30270/TCP,443:31858/
↪TCP,9443:32692/TCP   3h8m
istiod                  ClusterIP      172.20.146.223   <none>
                                                15010/TCP,15012/TCP,443/TCP,15014/
↪TCP       3h8m
```

upm-beaco-ff-base のインストールに失敗しました

詳細なエラー

```
☒
NAME                     READY   STATUS    RESTARTS   AGE
app-db-1-initdb-k4nbl    0/1     Pending   0          35m
```

```
kc get pvc -n upm-beaco-ff-base
```

```
NAME       STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS  ␣
↪VOLUMEATTRIBUTESCLASS   AGE
app-db-1   Pending                                                    <unset>
         35m
```

```
kc describe pvc app-db-1 -n upm-beaco-ff-base
```

```
Name:          app-db-1
Namespace:     upm-beaco-ff-base
StorageClass:
Status:        Pending
Volume:
…….
Events:
  Type    Reason         Age                  From                         Message
  ----    ------         ----                 ----                         -------
  Normal  FailedBinding  64s (x142 over 36m)  persistentvolume-controller  no␣
↪persistent volumes available for this claim and no storage class is se
```

解像度

kubectl edit sc gp2 を実行し、次のアノテーションを追加します。storageclass.kubernetes.io/is-default-class: "true"

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
```

Upm-thos のインストールに失敗しました

詳細なエラー

解像度

1. クラッシュループ内のポッドを確認します。
2. ログを確認して、権限関連のエラーを見つけます。

```
kubectl get pod -n monitoring
```

シークレットが正しいかどうかを確認します。

```
kubectl get secret -n monitoring
```

シークレットは、edb-object-storage で定義したものと一致する必要があります。

```
kubectl get secret -n monitoring  thanos-objstore-secret -o yaml
```

```
 apiVersion: v1
data:
  objstore.yml:␣
↪dHlwZTogUzMKY29uZmlnOgogIGJ1Y2tldDogYXBwbGlhbmNlLXNoYW50ZXN0LWVrcy1jbHVzdGVyLWVkYi1wb3N0Z3JlcwogIHJl

~ ☒  echo␣
↪dHlwZTogUzMKY29uZmlnOgogIGJ1Y2tldDogYXBwbGlhbmNlLXNoYW50ZXN0LWVrcy1jbHVzdGVyLWVkYi1wb3N0Z3JlcwogIHJl
↪| base64 -d
```

シークレットを修正するには

```
kubectl delete secret thanos-objstore-secret -n monitoring

kubectl delete pod storage-location-operator-controller-manager-xxxxxx -n storage-
↪location-operator
```

Helm アップグレードを再実行し、シークレット thanos-objstore-secret が正しいことを確認します。

```
kubectl delete pod <pod in crashloop> -n monitoring
```

イメージのプルに失敗しました

詳細なエラー

```
NAME                                                READY   STATUS                  ␣
↪RESTARTS   AGE
edbpgai-bootstrap-job-v1.0.6-appl-d12cvm-q2pbg   0/1     Init:AssetPullBackOff   0
      2m49s

Events:
  Type     Reason     Age               From               Message
  ----     ------     ----              ----               -------
  Normal   Scheduled  53s               default-scheduler  Successfully assigned␣
↪edbpgai-bootstrap/edbpgai-bootstrap-job-v1.0.6-appl-d12cvm-q2pbg to i-
↪001f23a316e13e905
  Warning  Failed     22s               kubelet            Failed to pull image
↪"docker.enterprisedb.com/staging_pgai-platform/edbpgai-bootstrap/bootstrap-eks:v1.0.
↪6-appl": rpc error: code = DeadlineExceeded desc = failed to pull and unpack image
↪"docker.enterprisedb.com/staging_pgai-platform/edbpgai-bootstrap/bootstrap-eks:v1.0.
↪6-appl": failed to resolve reference "docker.enterprisedb.com/staging_pgai-platform/
↪edbpgai-bootstrap/bootstrap-eks:v1.0.6-appl": failed to do request: Head "https://
↪docker.enterprisedb.com/v2/staging_pgai-platform/edbpgai-bootstrap/bootstrap-eks/
↪manifests/v1.0.6-appl": dial tcp 18.67.76.32:443: i/o timeout
  Warning  Failed     22s               kubelet            Error: ErrAssetPull
  Normal   BackOff    22s               kubelet            Back-off pulling image
↪"docker.enterprisedb.com/staging_pgai-platform/edbpgai-bootstrap/bootstrap-eks:v1.0.
↪6-appl"
  Warning  Failed     22s               kubelet            Error: AssetPullBackOff
  Normal   Pulling    7s (x2 over 52s)  kubelet            Pulling image "docker.
↪enterprisedb.com/staging_pgai-platform/edbpgai-bootstrap/bootstrap-eks:v1.0.6-appl"
```

解像度

1. AWS コンソールで、関連するサブネットを編集し、**パブリック IPv4 アドレスの自動割り当て** を有効にします。
2. ノードを削除して、EKS に変更をピックアップさせます。

```
k get node
```

```
NAME                  STATUS   ROLES    AGE   VERSION
i-001f23a316e13e905   Ready    <none>   19h   v1.31.4-eks-0f56d01
i-012fd44b2f575a514   Ready    <none>   20h   v1.31.4-eks-0f56d01
i-076d6a447c6551a21   Ready    <none>   20h   v1.31.4-eks-0f56d01
```

```
k delete node i-001f23a316e13e905 i-012fd44b2f575a514 i-076d6a447c6551a21
```

```
node "i-001f23a316e13e905" deleted
node "i-012fd44b2f575a514" deleted
node "i-076d6a447c6551a21" deleted
```

```
kgp -n edbpgai-bootstrap
```

```
NAME                                                READY   STATUS    RESTARTS   AGE
edbpgai-bootstrap-job-v1.0.6-appl-5ee7l7-xzfqt      1/1     Running   0          118s
```

Loki ポッドのクラッシュ

詳細なエラー

```
k get pods -n logging
```

```
NAME                          READY   STATUS             RESTARTS         AGE
loki-backend-0                1/2     CrashLoopBackOff   16 (2m1s ago)    59m
loki-read-5b8555fb64-9srmq    0/1     CrashLoopBackOff   16 (2m40s ago)   59m
loki-read-5b8555fb64-fw6ls    0/1     CrashLoopBackOff   16 (2m7s ago)    59m
loki-read-5b8555fb64-nw2vq    0/1     CrashLoopBackOff   16 (2m17s ago)   59m
loki-write-0                  0/1     CrashLoopBackOff   16 (2m37s ago)   59m
loki-write-1                  0/1     CrashLoopBackOff   16 (2m5s ago)    59m
loki-write-2                  0/1     CrashLoopBackOff   16 (2m16s ago)   59m
```

```
k logs loki-write-0 -n logging
```

```
level=info ts=2025-02-26T09:32:35.173785968Z caller=main.go:126 msg="Starting Loki"␣
↪version="(version=release-3.1.x-89fe788, branch=release-3.1.x, revision=89fe788d)"
level=info ts=2025-02-26T09:32:35.173821129Z caller=main.go:127 msg="Loading␣
↪configuration file" filename=/etc/loki/config/config.yaml
level=info ts=2025-02-26T09:32:35.174421237Z caller=server.go:352 msg="server␣
↪listening on addresses" http=[::]:3100 grpc=[::]:9095
level=info ts=2025-02-26T09:32:35.175391511Z caller=memberlist_client.go:435 msg=
↪"Using memberlist cluster label and node name" cluster_label= node=loki-write-0-
↪a0cfbc5d
level=info ts=2025-02-26T09:32:35.17599149Z caller=shipper.go:160 index-store=tsdb-
↪2020-05-15 msg="starting index shipper in WO mode"
level=info ts=2025-02-26T09:32:35.176072171Z caller=table_manager.go:136 index-
↪store=tsdb-2020-05-15 msg="uploading tables"
level=info ts=2025-02-26T09:32:35.176269663Z caller=head_manager.go:308 index-
↪store=tsdb-2020-05-15 component=tsdb-head-manager msg="loaded wals by period"␣
↪groups=0
level=info ts=2025-02-26T09:32:35.176305024Z caller=manager.go:86 index-store=tsdb-
↪2020-05-15 component=tsdb-manager msg="loaded leftover local indices" err=null␣
```

(次のページに続く)

(前のページからの続き)

```
↪successful=true buckets=0 indices=0 failures=0
level=info ts=2025-02-26T09:32:35.176325994Z caller=head_manager.go:308 index-
↪store=tsdb-2020-05-15 component=tsdb-head-manager msg="loaded wals by period"␣
↪groups=1
level=info ts=2025-02-26T09:32:35.180393332Z caller=module_service.go:82 msg=starting␣
↪module=server
level=info ts=2025-02-26T09:32:35.180477643Z caller=module_service.go:82 msg=starting␣
↪module=memberlist-kv
level=error ts=2025-02-26T09:32:35.180526964Z caller=loki.go:524 msg="module failed"␣
↪module=memberlist-kv error="starting module memberlist-kv: invalid service state:␣
↪Failed, expected: Running, failure: service memberlist_kv failed: failed to create␣
↪memberlist: Failed to get final advertise address: no private IP address found, and␣
↪explicit IP not provided"
level=error ts=2025-02-26T09:32:35.180549544Z caller=loki.go:524 msg="module failed"
 module=ring error="failed to start ring, because it depends on module memberlist-kv,␣
↪which has failed: invalid service state: Failed, expected: Running, failure:␣
↪starting module memberlist-kv: invalid service state: Failed, expected: Running,␣
↪failure: service memberlist_kv failed: failed to create memberlist: Failed to get␣
↪final advertise address: no private IP address found, and explicit IP not provided"
level=error ts=2025-02-26T09:32:35.180582375Z caller=loki.go:524 msg="module failed"␣
↪module=store error="failed to start store, because it depends on module memberlist-
↪kv, which has failed: invalid service state: Failed, expected: Running, failure:␣
↪starting module memberlist-kv: invalid service state: Failed, expected: Running,␣
↪failure: service memberlist_kv failed: failed to create memberlist: Failed to get␣
↪final advertise address: no private IP address found, and explicit IP not provided"
level=error ts=2025-02-26T09:32:35.180588655Z caller=loki.go:524 msg="module failed"␣
↪module=ingester error="failed to start ingester, because it depends on module␣
↪analytics, which has failed: context canceled"
level=error ts=2025-02-26T09:32:35.180593435Z caller=loki.go:524 msg="module failed"␣
↪module=distributor error="failed to start distributor, because it depends on module␣
↪analytics, which has failed: context canceled"
level=error ts=2025-02-26T09:32:35.180597415Z caller=loki.go:524 msg="module failed"␣
↪module=analytics error="failed to start analytics, because it depends on module␣
↪memberlist-kv, which has failed: invalid service state: Failed, expected: Running,␣
↪failure: starting module memberlist-kv: invalid service state: Failed, expected:␣
↪Running, failure: service memberlist_kv failed: failed to create memberlist: Failed␣
↪to get final advertise address: no private IP address found, and explicit IP not␣
↪provided"
level=info ts=2025-02-26T09:32:35.180687877Z caller=modules.go:1832 msg="server␣
↪stopped"
level=info ts=2025-02-26T09:32:35.180702097Z caller=module_service.go:120 msg="module␣
↪stopped" module=server
level=info ts=2025-02-26T09:32:35.180708417Z caller=loki.go:508 msg="Loki stopped"␣
↪running_time=85.855531ms
failed services
```

(次のページに続く)

(前のページからの続き)

```
github.com/grafana/loki/v3/pkg/loki.(*Loki).Run
    /src/loki/pkg/loki/loki.go:566
main.main
    /src/loki/cmd/loki/main.go:129
runtime.main
    /usr/local/go/src/runtime/proc.go:271
runtime.goexit
    /usr/local/go/src/runtime/asm_amd64.s:1695
level=error ts=2025-02-26T09:32:35.180772078Z caller=log.go:216 msg="error running␣
↪loki" err="failed services\ngithub.com/grafana/loki/v3/pkg/loki.(*Loki).Run\n\t/src/
↪loki/pkg/loki/loki.go:566\nmain.main\n\t/src/loki/cmd/loki/main.go:129\nruntime.
↪main\n\t/usr/local/go/src/runtime/proc.go:271\nruntime.goexit\n\t/usr/local/go/src/
↪runtime/asm_amd64.s:1695"
```

解像度

CIDR が 10 ではないため、 Loki からの既知の問題があります。*

1. VPC の IPv4 CIDR を確認します。たとえば、11.1.0.0/20 に設定されている場合、10.1.0.0/20 に設定します。

健全な上流はない

詳細なエラー

upm-dex のポッドはクラッシュループしています。ポッドログを確認して、エラーの詳細を表示します。この場合、pgai.portal.authentication.staticPasswords は正しく構成されていないため、正しい構成を更新し、展開を再実行します。

```
upm-dex                       upm-dex-66b96b5857-pdcg2
       1/2     CrashLoopBackOff   206 (3m34s ago)   17h

kubectl logs upm-dex-66b96b5857-pdcg2 -n upm-dex
error parse config file /tmp/dex.config.yaml-565347154: error unmarshaling JSON:␣
↪malformed bcrypt hash: crypto/bcrypt: hashedSecret too short to be a bcrypted␣
↪password
```

解像度

upgrade コマンドに 4 パラメーターを追加しました。

```
helm upgrade \
- n edbpgai-bootstrap \
- -install \
- -set parameters.global.portal_domain_name="${PORTAL_DOMAIN_NAME}" \
- -set parameters.global.dms_domain_name="$DMS_DOMAIN_NAME" \
```

(次のページに続く)

(前のページからの続き)

```
- -set parameters.global.load_balancer_provider="$LOAD_BALANCER_PROVIDER" \
- -set parameters.global.load_balancer_mode="$LOAD_BALANCER_MODE" \
- -set parameters.global.node_port_domain="$NODE_PORT_DOMAIN" \
- -set parameters.upm-beacon.server_host="${BEACON_SERVICE_DOMAIN_NAME}" \
- -set parameters.upm-beaco-ff-base.cookie_aeskey="${AES_256_KEY}" \
- -set system="eks" \
- -set remoteContainerRegistryURL="${REGISTRY_PACKAGE_URL}" \
- -set internalContainerRegistryURL="${REGISTRY_PACKAGE_URL}" \
- -set bootstrapAsset="${REGISTRY_PACKAGE_URL}/edbpgai-bootstrap/bootstrap-eks:$
↪{EDBPGAI_BOOTSTRAP_IMAGE_VERSION}" \
- -set pgai.portal.authentication.staticPasswords[0].email="owner@mycompany.com" \
- -set pgai.portal.authentication.staticPasswords[0].hash=$2y$10
↪$STTzuGJa3PydqHvlzi2z6OgDU1G/JLTqiuGblH.RemIutWxkztN5m \
- -set pgai.portal.authentication.staticPasswords[0].username="owner@mycompany.com" \
- -set pgai.portal.authentication.staticPasswords[0].userID="c5998173-a605-449a-a9a5-
↪4a9c33e26df7" \
- -version "${EDBPGAI_BOOTSTRAP_HELM_CHART_VERSION}" \
edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap
```

401 アクセスが拒否されました

詳細なエラー

解像度

config.yaml ファイルを確認します。この場合、staticPasswords セクションは適切に設定されていません。正しい値を使用して helm upgrade を再実行します。

```
k get secret upm-dex -n upm-dex -o yaml
```

```
apiVersion: v1
data:
  config.yaml:␣
↪aXNzdWVyOiBodHRwczovL3BvcnRhbC5iYXN1cHBvcnQub3JnL2F1dGgKc3RvcmFnZToKICB0eXBlOiBwb3N0Z3JlcwogIGNvbmZp
…

echo "xxx" | base64 -d
staticPasswords:
  - email: owner@mycompany.com
    hash: XXX
    userID: c5998173-a605-449a-a9a5-4a9c33e26df7
    username: owner@mycompany.com
```

別の方法として、次の 1 行コマンドを使用します。

```
k get secret -n upm-dex upm-dex -ojsonpath={.data.config\.yaml} | base64 -d
```

HTTP エラー 500

詳細なエラー

バックイングレスゲートウェイは着信接続にアクセスできません。パブリックネットワークから Web ページにアクセスしている場合、https が許可される唯一のプロトコルであるため、ポート 443 を介して行う必要があります。

解像度

ingressgateway のステータスを確認します。 ingressgateway は istio-ingressgateway です。

```
k get all -n istio-system
```

```
NAME                                        READY   STATUS    RESTARTS   AGE
pod/istio-ingressgateway-7956f6d57c-jd8jm   1/1     Running   0          14m
pod/istio-ingressgateway-7956f6d57c-jmmv8   1/1     Running   0          14m
pod/istio-ingressgateway-7956f6d57c-r65tk   1/1     Running   0          14m
pod/istiod-7f7bbcdcbb-fzggv                 1/1     Running   0          3h26m
pod/istiod-7f7bbcdcbb-mxjpz                 1/1     Running   0          3h26m

NAME                           TYPE           CLUSTER-IP       EXTERNAL-IP
                                                          PORT(S)
               AGE
service/istio-ingressgateway   LoadBalancer   172.20.92.132    k8s-istiosys-istioing-
↪987da680df-8452f790b4602f2c.elb.us-east-1.amazonaws.com   80:30950/TCP,443:32604/
↪TCP,9443:31744/TCP   3h26m
service/istiod                 ClusterIP      172.20.112.158   <none>
                                                          15010/TCP,15012/TCP,443/TCP,
↪15014/TCP       3h26m

NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/istio-ingressgateway    3/3     3            3           3h26m
deployment.apps/istiod                  2/2     2            2           3h26m

NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/istio-ingressgateway-68fcd46fff    0         0         0       84m
replicaset.apps/istio-ingressgateway-7956f6d57c    3         3         3       14m
replicaset.apps/istio-ingressgateway-866d9bd74b    0         0         0       3h26m
replicaset.apps/istiod-7f7bbcdcbb                  2         2         2       3h26m
```

ゲートウェイ構成に構成エラーがないかを確認します。ゲートウェイルールは 443 ポートでリッスンするように既に設定されています。

```
k get gateway/upm-portal -n istio-system -oyaml
```

```
apiVersion: networking.istio.io/v1
kind: Gateway
….
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - portal.basupport.org
    port:
      name: http
      number: 80
      protocol: HTTP
    tls:
      httpsRedirect: true
  - hosts:
    - portal.basupport.org
    port:
      name: https-443
      number: 443
      protocol: HTTPS
  …
```

次に、展開ログを確認します。

```
2025-02-20T03:42:32.203112Z     warning envoy main external/envoy/source/server/
↪server.cc:835   Usage of the deprecated runtime key overload.global_downstream_max_
↪connections, consider switching to `envoy.resource_monitors.downstream_connections`␣
↪instead.This runtime key will be removed in future.      thread=15
2025-02-20T03:42:32.203389Z     warning envoy main external/envoy/source/server/
↪server.cc:928   There is no configured limit to the number of allowed active␣
↪downstream connections. Configure a limit in `envoy.resource_monitors.downstream_
↪connections` resource monitor.   thread=15
2025-02-20T03:42:32.211729Z     info    xdsproxy        connected to delta upstream␣
↪XDS server: istiod.istio-system.svc:15012   id=1
2025-02-20T03:42:32.268369Z     info    ads     ADS: new connection for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system-1
2025-02-20T03:42:32.268437Z     info    cache   returned workload certificate from␣
↪cache        ttl=23h59m59.731565472s
2025-02-20T03:42:32.268890Z     info    ads     SDS: PUSH request for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system resources:1 size:4.0kB resource:default
2025-02-20T03:42:32.279500Z     info    ads     ADS: new connection for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system-2
```

(次のページに続く)

(前のページからの続き)

```
2025-02-20T03:42:32.279654Z     info    cache   returned workload certificate from␣
↪cache        ttl=23h59m59.72034871s
2025-02-20T03:42:32.279711Z     info    ads     SDS: PUSH request for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system resources:1 size:4.0kB resource:default
2025-02-20T03:42:32.280287Z     info    ads     ADS: new connection for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system-3
2025-02-20T03:42:32.280372Z     info    cache   returned workload trust anchor from␣
↪cache       ttl=23h59m59.719629366s
2025-02-20T03:42:32.280503Z     info    ads     SDS: PUSH request for node:istio-
↪ingressgateway-68fcd46fff-g4nth.istio-system resources:1 size:1.1kB resource:ROOTCA
2025-02-20T03:42:32.320657Z     info    wasm    fetching image staging_pgai-platform/
↪upm-beaco-filters/upm-oidc from registry docker.enterprisedb.com with tag v1.1.44
2025-02-20T03:42:32.323034Z     info    wasm    fetching image staging_pgai-platform/
↪upm-beaco-filters/upm-error-transformer from registry docker.enterprisedb.com with␣
↪tag v1.1.44
2025-02-20T03:42:32.323073Z     info    wasm    fetching image staging_pgai-platform/
↪upm-beaco-filters/upm-authz-checker from registry docker.enterprisedb.com with tag␣
↪v1.1.44
2025-02-20T03:42:35.074684Z     warning envoy wasm external/envoy/source/extensions/
↪common/wasm/context.cc:1198 wasm log: error parsing plugin configuration: Error(
↪"aes_key: Invalid Length, got 3 bytes, expected 32", line: 1, column: 12)  ␣
↪thread=15
2025-02-20T03:42:35.074712Z     error   envoy wasm external/envoy/source/extensions/
↪common/wasm/wasm.cc:110     Wasm VM failed Failed to configure base Wasm plugin    ␣
↪thread=15
2025-02-20T03:42:35.075923Z     critical        envoy wasm external/envoy/source/
↪extensions/common/wasm/wasm.cc:474     Plugin configured to fail closed failed to␣
↪load thread=15
2025-02-20T03:42:35.076559Z     warning envoy config external/envoy/source/extensions/
↪config_subscription/grpc/delta_subscription_state.cc:269  delta config for type.
↪googleapis.com/envoy.config.core.v3.TypedExtensionConfig rejected: Unable to create␣
↪Wasm HTTP filter istio-system.upm-oidc        thread=15
2025-02-20T03:42:35.076572Z     warning envoy config external/envoy/source/extensions/
↪config_subscription/grpc/grpc_subscription_impl.cc:138    gRPC config for type.
↪googleapis.com/envoy.config.core.v3.TypedExtensionConfig rejected: Unable to create␣
↪Wasm HTTP filter istio-system.upm-oidc thread=15
2025-02-20T03:42:35.076577Z     warning envoy config external/envoy/source/extensions/
↪config_subscription/grpc/grpc_subscription_impl.cc:138    gRPC config for type.
↪googleapis.com/envoy.config.core.v3.TypedExtensionConfig rejected: Unable to create␣
↪Wasm HTTP filter istio-system.upm-oidc thread=15
2025-02-20T03:42:35.076584Z     warning envoy config external/envoy/source/extensions/
↪config_subscription/grpc/grpc_subscription_impl.cc:138    gRPC config for type.
↪googleapis.com/envoy.config.core.v3.TypedExtensionConfig rejected: Unable to create␣
↪Wasm HTTP filter istio-system.upm-oidc thread=15
2025-02-20T03:42:35.189419Z     info    Readiness succeeded in 3.054667763s
```

(次のページに続く)

(前のページからの続き)

```
2025-02-20T03:42:35.189690Z     info    Envoy proxy is ready
```

ここでのエラーは、wasm plugin が正常にブートストラップされなかったために発生したため、ingressgateway は実際には準備ができていません。

```
k get pod -listio=ingressgateway -n istio-system
```

```
Enter MFA code for arn:aws:iam::121858151946:mfa/zane.zhou@enterprisedb.com:
NAME                                    READY   STATUS    RESTARTS   AGE
istio-ingressgateway-7956f6d57c-jd8jm   1/1     Running   0          72m
istio-ingressgateway-7956f6d57c-jmmv8   1/1     Running   0          72m
istio-ingressgateway-7956f6d57c-r65tk   1/1     Running   0          72m
```

しかし、ポッドでは、準備状況チェックが実際に失敗したことがわかります。

```
k describe pod/istio-ingressgateway-7956f6d57c-jd8jm -n istio-system
```

```
Warning  Unhealthy  20m (x2 over 20m)  kubelet            Readiness probe failed: Get
↪"http://10.0.30.134:15021/healthz/ready": dial tcp 10.0.30.134:15021: connect:␣
↪connection refused
```

この問題は、wasm plugin の構成の誤りによって発生します。aes_key が正しく設定されていません。構成を修正し、ingressgateway を再展開します。

```
k edit WasmPlugin/upm-oidc
```

次に、次の aes_key をデフォルト値または実際の運用環境でのカスタム生成値に設定します。

```
pluginConfig:
  aes_key: rzkutHl8NJNztPMEJYykZouHslNiA7xmIXH+58ISUVo=
```

次に、展開をリスタートします。

```
k rollout restart deployment/istio-ingressgateway -n istio-system
```

Postgres クラスタープロビジョニングが X%完了で停止する

詳細なエラー

単一ノードクラスターの作成が 87%完了で停止しました。

解像度

CPU とメモリの範囲はノードグループよりもかなり小さいです

詳細なエラー

ノードグループは m5.4xlarge で、これは 16vCPU と 64GB RAM、ノードなしです。クラスターの作成時に、コンソールに表示される範囲は CPU 0〜3.92 コアとメモリ 0〜14.3 Gi です。

解像度

これは、サイズ 0 のノードプールによるものです。ノードプールにノードがある場合、問題ありません。しかし、問題が発生したときにノードを作成すると、機能しません。ノードが作成された後も、コンソールには同じ値が表示されます。

保留中 https://enterprisedb.atlassian.net/browse/UPM-45883

ディスクのスケールアップ

詳細なエラー

コンソールのスケールディスク。新しい値はクラスター yaml に表示されますが、pvc / pv には適用されません。

```
kubectl get cluster p-2hgee2782y -o yaml
```

```
storage:
  resizeInUseVolumes: true
  size: 5Gi
```

```
k get pv
```

```
pvc-4a35b3ba-507e-4d4c-9017-360326e57b9d   2Gi    RWO            Delete          ␣
↪Bound    p-2hgee2782y/p-2hgee2782y-1
```

解像度

CNP オペレータログには次のエラーが含まれています。

```
kubectl logs postgresql-operator-controller-manager-7cc67597df-4ggdf -n postgresql-
↪operator-system
```

```
{
  "level": "error",
  "ts": "2025-01-17T10:02:35.739693483Z",
  "msg": "Reconciler error",
  "controller": "cluster",
  "controllerGroup": "postgresql.k8s.enterprisedb.io",
  "controllerKind": "Cluster",
  "Cluster": { "name": "p-2hgee2782y", "namespace": "p-2hgee2782y" },
  "namespace": "p-2hgee2782y",
```

(次のページに続く)

(前のページからの続き)

```
  "name": "p-2hgee2782y",
  "reconcileID": "2de2ade7-2bf7-45f4-a2af-3f135a4f7ad9",
  "error": persistentvolumeclaims "p-2hgee2782y-1" is forbidden: only dynamically␣
↪provisioned pvc can be resized and the storageclass that provisions the pvc must␣
↪support resize,
  "stacktrace": "sigs.k8s.io/controller-runtime/pkg/internal/controller.(␣
↪*Controller[...]).reconcileHandler\n\tcloud-native-postgres/vendor/sigs.k8s.io/
↪controller-runtime/pkg/internal/controller/controller.go:316\nsigs.k8s.io/
↪controller-runtime/pkg/internal/controller.(* Controller[...]).processNextWorkItem\
↪n\tcloud-native-postgres/vendor/sigs.k8s.io/controller-runtime/pkg/internal/
↪controller/controller.go:263\nsigs.k8s.io/controller-runtime/pkg/internal/
↪controller.(*Controller[...]).Start.func2.2\n\tcloud-native-postgres/vendor/sigs.
↪k8s.io/controller-runtime/pkg/internal/controller/controller.go:224",
}
```

storageclass ～allowVolumeExpansion: true を確認・編集します。

```
kubecl edit storageclass gp2
```

```
k get storageclass gp2 -o yaml
```

```
allowVolumeExpansion: true
```

CNP オペレーターポッドを削除して、変更を適用します。

```
kubectl  delete pod postgresql-operator-controller-manager-7cc67597df-tgl7b -n␣
↪postgresql-operator-system
```

```
kubectl get pv
```

```
pvc-4a35b3ba-507e-4d4c-9017-360326e57b9d   5Gi        RWO            Delete           ␣
↪Bound    p-2hgee2782y/p-2hgee2782y-1
```

stats-collector-db ディスクの問題

ほとんどの場合、 stats-collector-db でディスクの問題が発生している場合、次のいずれかが発生しています。

1. ディスクが容量に近づいていることを示すアラートマネージャーのアラート。これは、ディスクがまだいっぱいになっていないため、クラスターがまだ実行されている場合に発生します。
2. ポッドがクラッシュループにあることを示すアラートマネージャーのアラート。これは、ディスクがいっぱいであるためにクラスターがクラッシュした場合に発生します。また、クラスターの**モニタリング** タブでは、関連するディスク統計を取得できません。

シナリオ 1 クラスターの実行中にディスクの使用率が高くなる

クラスターは正常に動作しているが、ディスク使用量が高いことに気付いた場合、2 つのオプションがあります。

ソリューション A データ保持期間を短縮する

データ保持期間を短縮することは、保存するデータの量を減らして、ディスク使用量を削減する簡単な方法です。デフォルトの保持期間は 168 時間 7 日です。このソリューションを使用して、保持期間を 96 時間 4 日に短縮します。

保持期間を短縮すると、利用可能なディスクが約 42%増加します。リテンション率を低下させるには

1. プラットフォームへの kubectl アクセスを持つ端末から次のコマンドを使用して、ConfigMap を編集します。

```
kubectl -n upm-api-stats-collector edit cmd
upm-api-stats-collector-configmap-<your-config-map-string>
```

1. ConfigMap で、retention-period パラメーターを見つけ、その値を 168h から 96h に変更します。
2. upm-api-stats-collector を再起動します。

```
kubectl rollout restart deploy/upm-api-stats-collector -n upm-api-stats-collector
```

これらの手順を完了すると、DB の使用量は安定した範囲内に留まり、Alert Manager のアラートは消えます。

解決策 B ディスクボリュームサイズを増やす

7 日間のデータを保持する必要がある場合、問題を解決するには、ディスクのストレージ容量を増加する必要があります。

1. クラスター構成を編集します。

```
kubectl edit cluster stats-collector-db -n upm-api-stats-collector
```

1. stats-collector-db 仕様を見つけて、 .spec.storage.size の値を優先するサイズに増やします。
2. upm-api-stats-collector を再起動します。

```
kubectl rollout restart deploy/upm-api-stats-collector -n upm-api-stats-collector
```

シナリオ 2 ディスクがいっぱいで、クラスターがクラッシュする

ディスクがいっぱいである場合、クラスターがクラッシュループに陥ってしまう可能性があります。例

```
upm-api-stats-collector      stats-collector-db-1 0/1     CrashLoopBackOff   197␣
↪(4m34s ago)   4d7h
```

この問題に対処するには 2 つのオプションがあります。

解決策 A データベースを再作成します

履歴統計データを保持する必要がない場合

1. stats-collector-db クラスターを削除します。

```
kubectl delete cluster stats-collector-db -n upm-api-stats-collector
```

1. S3 バケットから stats-collector-db バックアップフォルダーを削除します。

```
S3/edb-internal-backups/<your-S3-string>/databases/transporter-db/
```

1. helm upgrade を実行して、デフォルト設定で空のデータベースを再作成します。

```
helm upgrade -n edbpgai-bootstrap \
- -install -f ./values.yaml \
- -version "${EDB_PLATFORM_VERSION}" \
edbpgai-bootstrap "${HCP_HELM_REPO_NAME}/edbpgai-bootstrap
```

1. upm-api-stats-collector を再起動します。

```
kubectl rollout restart deploy/upm-api-stats-collector -n upm-api-stats-collector
```

解決策 B ディスクボリュームサイズを増やす

既存のデータを保持する必要がある場合は、ディスクのボリュームサイズを増やす必要があります。

1. クラスター構成を編集します。

```
kubectl edit cluster stats-collector-db -n upm-api-stats-collector
```

1. stats-collector-db 仕様を見つけて、 .spec.storage.size の値を増加します。
2. クラッシュループ内のポッドを削除します。
3. ump-api-stats-collector を再起動します。

```
kubectl rollout restart deploy/upm-api-stats-collector -n upm-api-stats-collector
```

再起動後、データベースのボリュームが大きくなり、クラッシュループが解決されます。

> **警告**
>
> ボリュームサイズをデフォルトより大きくすると、バージョン`1.2.x` を実行しているクラスターのパッチバージョンのアップグレードがブロックされます。

ディスクボリュームサイズを増やしたものの、HM v1.2.x でマイナーバージョンのアップグレードを実行する必要がある場合は、最初にデータベースのサイズをデフォルトのサイズ 50GB に戻す必要があります。データベースのサイズ変更については、サポートチームに連絡してください。!!!

5.3.4.7 Using query recommendations in Hybrid Manager

5.3.4.8 ハイブリッドマネージャーでクエリ推奨事項を使用する

ハイブリッドマネージャーのクエリの推奨事項は、データベースワークロードのパフォーマンスを継続的に向上させるのに役立ちます。

詳細なクエリ診断を実用的な推奨事項と関連付けることにより、ハイブリッドマネージャーでは次のことが可能になります。

- システムの速度を低下させているクエリを特定します。
- これらのクエリを最適化する方法を理解します。
- クエリレイテンシーとリソース消費を削減するアクションを実行します。

5.3.4.8.1 クエリのレコメンデーションを使用するのは誰ですか?

クエリの推奨事項は次の場合に役立ちます。

- **DBA およびプラットフォームエンジニア**データベースパフォーマンスの調整。
- **データエンジニア** データパイプラインとバッチジョブを最適化します。
- **開発者** アプリケーションのパフォーマンスのボトルネックを診断する。
- **SRE とオブザーバビリティチーム**環境全体でデータベースの状態をモニタリング。

5.3.4.8.2 クエリレコメンデーションとは何ですか?

ハイブリッドマネージャーは、詳細なクエリ診断を収集し、分析します。このプラットフォームは、次のようなワークロードを最適化する機会を特定します。

- データ取得パフォーマンスを向上させるための **インデックスの推奨事項**。
- クエリプランニングと実行の精度を向上させるための **統計の推奨事項** 。

5.3.4.8.3 クエリレコメンデーションを使用する理由

クエリの推奨事項は次のことに役立ちます。

- 最小限の手動調整でクエリのパフォーマンスを向上させます。
- 最も高いインパクトを提供するクエリーにチューニング作業を集中します。
- 進化するワークロードに継続的に適応する —クエリパターンとデータ量が変更されると、最適化のニーズも変化します。
- データベースの負荷、インフラストラクチャコスト、およびユーザー側のレイテンシーを削減します。
- 運用上の問題に反応するのではなく、パフォーマンスをプロアクティブに管理します。
- データ駆動の洞察に基づいて、構造化された調整ワークフローを構築します。

5.3.4.8.4 クエリの推奨事項を使用する場合

定期的なチューニングとパフォーマンス管理プロセスの一部としてクエリの推奨事項を使用します。

- 新しいアプリケーションの **初期オンボーディング** 中。
- **データベースを移行**新しいプラットフォームまたはバージョンに移行する場合。
- 重要な **スキーマまたはアプリケーションロジックの変更** の後。
- **パフォーマンス低下** または **キャパシティ傾向** を監視する場合。
- **進行中のデータベースの状態レビュー** の一部として —コードを変更せずにクエリパターンは進化します。
- **リリース検証** 中 —新しいワークロードパターンが調整の機会を導入したかどうかを確認します。

5.3.4.8.5 クエリレコメンデーションの仕組み

ハイブリッドマネージャーは、次を含むクエリ診断データを収集します。

- クエリーレイテンシーと実行パターン。
- 待機状態の分析
- ワークロードの傾向とクエリ頻度。
- **Query Diagnostics** タブ —ここには、個々のクエリとインラインで推奨事項が表示されます。
- **Recommendations** タブ —選択したデータベースまたはクラスターの現在の推奨事項がすべてリストされます。

問題のあるクエリーの診断から、関連する推奨事項の理解と適用までシームレスに移行できます。

5.3.4.8.6 クエリの推奨事項をチューニングプロセスに統合する

ハイブリッドマネージャーの推奨事項をより広範な調整ワークフローに組み込むことができます。

- **毎日または毎週のレビュー** [推奨事項] タブを定期的にチェックして、新しい機会を特定します。
- **ターゲットを絞ったチューニング**クエリ診断でフラグが立てられた高レイテンシーまたは高頻度のクエリに焦点を当てます。インパクトの高い推奨事項欠落しているインデックスなどを適用することから始めます。
- **トレンド分析**推奨事項を適用した後にクエリパフォーマンスがどのように進化するかを監視します。ハイブリッドマネージャーは、調整作業が効果的かどうかを追跡するのに役立ちます。
- **チーム間のコラボレーション**開発者、DBA、およびプラットフォームエンジニアは、同じ推奨事項を使用して、調整作業の優先順位を付け、調整できます。
- **最適化バックログ**未解決の推奨事項を、キャパシティプランニングサイクル中に取り組む調整作業のバックログとして扱います。

5.3.4.8.7 ワークフロー クエリから最適化まで

クエリーのレコメンデーションを使用するには

1. クラスターの **Query Diagnostics** タブを開きます。
2. 時間範囲を選択し、クエリパフォーマンスを確認します。
3. 関連する推奨事項を持つクエリを探します。これらはクエリリストに視覚的に示されます。
4. 推奨事項を選択して [**Recommendations**] タブを開き、関連する推奨事項が強調表示されます。
5. 以下を含む推奨事項の詳細を確認します。

- 選択したクエリーへの影響.
- メリットのあるその他のクエリー
- 最適化の適用に関するガイダンス。

6. 推奨事項を適用しますたとえば、インデックスを作成するか、統計を更新します。
7. クエリーを経時的に監視して、パフォーマンスの向上を検証します。

5.3.4.8.8 外部クラスターの要件

外部非ハイブリッドマネージャー管理対象クラスターのクエリの推奨事項を表示するには

- **Query Diagnostics* 拡張機能を有効にする必要があります。
- **Query Advisor** 拡張機能を有効にする必要があります。
- ハイブリッドマネージャー **エージェント**がインストールされ、実行されている必要があります。

これらのコンポーネントがないと、ハイブリッドマネージャーは必要なデータを収集して、推奨事項を生成できません。

5.3.4.8.9 シナリオ例

次に、クエリの推奨事項が値を提供する一般的な調整シナリオを示します。

- **遅いアプリケーションページ**頻繁に使用され、高いレイテンシーのクエリが **クエリ診断**で特定されます。推奨事項は、フィルタ列にインデックスを追加することをお勧めします。適用されると、応答時間が大幅に向上します。
- **ETL ジョブ最適化**最近のデータの増加により、バッチプロセスの実行が遅くなりました。クエリ診断は、集計クエリのパフォーマンスが低下していることを示しています。統計更新の推奨事項により、プランナーの精度が向上し、ジョブのパフォーマンスが復元されます。
- **キャパシティプランニング**ハードウェアをスケーリングする前に、推奨事項を確認して、コンピューティングを追加せずにチューニングでボトルネックを緩和できるかどうかを確認します。
- **プロアクティブなチューニング**日常的なチューニングサイクル中に、重要なデータベース全体の推奨事項を確認し、改善を体系的に適用します。その結果、高価なクエリーを最適化することにより、データベース全体の負荷が削減されます。

- **リリース検証**新しいアプリケーション機能を展開した後、新しいクエリパターンがチューニングの機会が表面化したかどうかを確認します。推奨事項を早期に適用すると、将来のパフォーマンスの問題が防止されます。

5.3.4.8.10 サポートされているレコメンデーションタイプ

ハイブリッドマネージャーは現在以下をサポートしています

- **インデックスの推奨事項** —頻繁に使用される列に不足しているインデックスを追加するなど。
- **統計の推奨事項** —古い統計を更新してプランナーの精度を向上させるなど。

将来のリリースでは、追加の推奨タイプが計画されています。

5.3.4.8.11 関連情報

- *Cluster Query Diagnostics*
- *Cluster recommendations*

第6章　Agent Factory in Hybrid Manager

Hybrid Manager の **エージェントファクトリーワークロード**は、スケーラブルな AI および機械学習機能を Hybrid Manager プラットフォームにもたらします。Postgres、ベクトル検索、モデルサービング、AI パイプラインにわたって緊密に統合されているため、ハイブリッドマネージャーが管理するクラスターとデータ全体で AI を運用できます。

ハイブリッドマネージャーの Agent Factory を使用すると、次のことができます。

- 内部または外部向けに Langflow を使用して AI フローとエージェントを構築します。
- 統合された KServe ベースの推論と GPU アクセラレーションを使用して、AI モデルを大規模に提供します。
- ナレッジベースを作成し、ベクトル検索を使用して検索拡張生成 RAG を実行します。
- モデルライブラリを管理し、ハイブリッドマネージャーガバナンスの下で信頼できるモデルを展開します。
- AI 機能をアプリケーションとデータパイプラインに直接統合します。

6.1 構築できる AI ソリューションの例

ハイブリッドマネージャーのエージェントファクトリーは、多くのドメインにわたってソリューションを有効にします。

- エンタープライズ検索およびナレッジアシスタント企業文書、データベース、イントラネットコンテンツと統合する RAG ベースのアシスタントを構築します。
- カスタマーサポートチャットボット独自のデータとドメイン固有のモデルを活用したアシスタントを展開し、会話型 AI とナレッジベースを組み合わせます。
- AI 駆動型データアプリ アプリの AI 搭載エンドポイント - セマンティック検索、レコメンデーション、類似性検索、または自然言語クエリーを公開します。
- 内部ツールのためのオペレーショナル AI DevOps、カスタマーサクセス、HR オートメーション、セールスイネーブルメントなどを支援する AI エージェントとツールを構築します。
- ドメイン固有のモデル提供独自のまたは微調整されたモデル LLM、埋め込みモデル、ランキングモデルを、ハイブリッドマネージャークラスター内のスケーラブルな推論サービスとして提供します。

単一のアシスタントまたはモデルエンドポイントで小規模にスタートし、Agent Factory アーキテクチャを使用して完全な AI 搭載アプリケーションにスケーリングできます。

ハイブリッドマネージャーワークロードの## Agent Factory

ハイブリッドマネージャーは、エージェントファクトリーの機能のフルセットをサポートし、コントロールプレーンに緊密に統合されています。

6.1.1 AI フローの構築

- *Langflow FAQ* —HM 管理モデル、ナレッジベース、Postgres クラスターに接続された EDB コンポーネントを使用して、AI パイプラインとフローを構築および展開する
- *Pipeline Designer concepts* —ノーコードのビジュアルインターフェイスを介して AIDB データパイプラインとナレッジベースを作成する

6.1.2 モデルの管理と提供

- ステップ *2* モデルライブラリーにアクセスする — モデルアセットを管理および統治する
- モデルサービング — Kubernetes でのモデル推論サービスの展開とスケーリング
- *GPU resource management* — サービング用の GPU 容量を構成および割り当てます

6.2 Agent Factory がハイブリッドマネージャーにどのように適合するか

Hybrid Manager は、データベース、分析、AI ワークロードのコントロールプレーンです。Agent Factory は、他の HM 機能を補完します。

- データベース Postgres クラスターをプロビジョニングおよび管理します。Agent Factory は、ベクター検索、RAG、およびアプリケーションエンドポイントのためにこれらに接続します。
- 分析 データを Iceberg/Delta にオフロードし、 SQL 経由でクエリします。Agent Factory のパイプラインと Vector Engine は、取得のためにこれらのデータセットと統合します。
- オペレーション HM の可観測性とガバナンスを使用して、データベースと分析とともに AI フロー、パイプライン、およびモデル提供をモニタします。

HM のエージェントファクトリーの前提条件

- Agent Factory が有効になり、Langflow、Pipeline Designer、および Model Serving の資格が必要なハイブリッドマネージャープロジェクト。
- 提供に必要な GPU リソースへのアクセス。 *GPU resource management* を参照してください。

6.3 ハイブリッドマネージャーのソブリン AI

ハイブリッドマネージャーのエージェントファクトリーは、データと同じ場所で AI ワークロードを実行します。

- **ご使用の環境**モデル提供、パイプライン、およびベクター検索は、Hybrid Manager プロジェクトおよび Kubernetes クラスター内で直接実行されます。

- **あなたのデータ。**ソースドキュメント、埋め込み、および推論結果は、明示的に構成しない限り、環境を離れることはありません。
- **あなたのモデル。**モデルイメージはモデルライブラリーでキュレートされ、クラスター内のエンドポイントから提供されます。
- **あなたの可視性。**モデルログと AI ワークロードメトリックは、Postgres と分析に使用するのと同じプラットフォームを介して観察できます。

これは、プロンプト、ドキュメント、および埋め込みがネットワークを離れ、サードパーティ環境で処理される外部 AI サービスとは異なります。ハイブリッドマネージャーを使用すると、すべての取得、生成、およびストレージがプロジェクトとインフラストラクチャ内で発生します。

高セキュリティの展開の場合、ハイブリッドマネージャーは完全にエアギャップで動作できます。モデルイメージはプリロードされ、すべてのワークロードは完全に制御されたクラスター内にあります。エアギャップセットアップについては、**前提条件** を参照してください。

6.4 はじめる

- HM の Agent Factory は初めてですか？ここから始めます *Getting started with Pipeline Designer*
- 環境要件の確認 **前提条件**
- 現実世界のパターンを探索する *Use Cases and Personas for Agent Factory on Hybrid Manager*
- アーキテクチャの理解 *Agent Factory Architecture on Hybrid Manager*

6.4.1 Getting Started with AI Factory

6.4.1.1 Getting started with Agent Factory on Hybrid Manager

Agent Factory は、Hybrid Manager の AI と機械学習のワークロードです。これにより、NVIDIA NIM モデルを独自の GPU インフラストラクチャに展開して提供し、Langflow で AI フローとエージェントを構築し、Pipeline Designer でナレッジベースと RAG パイプラインを作成し、モデルエンドポイントをアプリケーションに公開できます。すべてソブリン Kubernetes 環境内で。

このページでは、何かを実行するための最速パスを説明します。

6.4.1.1.1 ビルドするもの

ハイブリッドマネージャーの Agent Factory は、2 つの開始パスをサポートしています。

- ナレッジベースに基づいて、Langflow で AI フローを構築します。
- パイプラインデザイナーを介してナレッジベースパイプラインを作成します。

どちらのパスも展開されたモデルクラスターが必要です。まだ行っていない場合は、そこから開始します。

6.4.1.1.2 始める前に

以下のパスをたどる前に、これらの手順を順番に完了します。Langflow と Pipeline Designer がモデルを使用できる前に、Agent Factory を有効にし、モデルクラスターを実行する必要があります。

- プロジェクトで Agent Factory を有効にする **エージェントファクトリーの有効化**
- GPU ノードグループのプロビジョニング *GPU Recommendations for Default NIM Models*
- モデルライブラリーからモデルクラスターを展開する *Deploy Agent Factory with Hybrid Manager UI*

6.4.1.1.3 オプション A — Langflow で AI フローを構築する

Langflow は、HM に同梱されているビジュアルフロービルダーです。これを使用して、EDB コンポーネントナレッジベース、埋め込みモデル、LLM、Postgres データベースを、対話型で実行または呼び出し可能サービスとして展開できる AI フローとエージェントに接続します。

- *Langflow quickstart* —エンドツーエンドのウォークスルーナレッジベースのセットアップ、コンポーネントの配線、およびプレイグラウンドでのフローの実行。

6.4.1.1.4 オプション B —ナレッジベースパイプラインを作成する

Pipeline Designer は、AIDB データパイプラインを構築するための HM コンソールのノーコードウィザードです。それを使用して、ドキュメントを取り込み、埋め込みを生成し、セマンティック検索または RAG の準備ができているベクターナレッジベースに保存します。

- *Getting started with Pipeline Designer*
- *Creating pipelines*

6.4.1.1.5 アプリケーションからモデルエンドポイントにアクセス

HM を介して展開されたモデルは、OpenAI 互換の REST エンドポイントを公開します。Langflow または Pipeline Designer を介せずに、アプリケーション、スクリプト、または外部ツールから直接呼び出すことができます。

- *Access KServe endpoints*
- *Quickstart (Python)*

6.4.1.1.6 監視と反復

モデルとフローが実行されたら、HM のビルトイン可観測性を使用して、推論レイテンシー、GPU 使用率、およびフローの状態を追跡します。

- *Observability (Agent Factory on HM)*
- *Monitor InferenceService*

6.4.1.2 Prerequisites for Agent Factory on Hybrid Manager

6.4.1.2.1 エージェントファクトリーの有効化

ハイブリッドマネージャーでエージェントファクトリーを使用する前に、HM のインストール中に有効にする必要があります。

Agent Factory 固有の有効化手順については、*Creating other necessary secrets* を参照してください。

6.4.1.2.2 インフラストラクチャ要件

Kubernetes Cluster Foundation

Agent Factory には、AI ワークロード用の十分なリソースを備えた適切に構成された Hybrid Manager Kubernetes クラスターが必要です。クラスターは GPU スケジューリングをサポートし、さまざまなワークロードタイプに合わせて構成された適切なノードグループが必要です。

クラスター要件

- GPU デバイスプラグインをサポートする Kubernetes 1.27 以降
- GPU ノード管理のためにインストールされた NVIDIA GPU オペレーター
- オーケストレーションコンポーネントに十分な CPU とメモリ
- サービスメッシュ通信をサポートするネットワークポリシー

GPU ノード構成

GPU リソースは、モデル提供操作に不可欠です。ノード構成は、モデル要件および予想されるワークロード特性と一致する必要があります。

GPU ノードの要件

- CUDA 12.1+をサポートする NVIDIA GPU
- `nvidia.com/gpu=true` でラベル付けされた GPU ノード
- 専用スケジューリング用の GPU テイント `nvidia.com/gpu`
- ターゲットモデルサイズに十分な GPU メモリ

モデルごとの GPU 要件とクラウドノードの推奨事項については、*GPU Recommendations for Default NIM Models* を参照してください。

6.4.1.2.3 レジストリ構成

インターネットに接続された展開

インターネットにアクセスできるクラスターの場合、追加のイメージミラーリングは必要ありません。クラスターが `nvcr.io` に出力されていることを確認して、HM がモデルの展開中に NVIDIA NIM イメージをプルできるようにします。

エアギャップ展開

インターネットアクセスのない環境の場合、ハブガイドを使用してイメージと展開アセットを事前に準備し、ハイブリッドマネージャーでプライベートレジストリアクセスを構成します。

モデルイメージの移行

これらのハブ参照に従って、必要なモデルイメージを準備してプライベートレジストリにミラーリングします。

- プライベートレジストリとイメージガバナンス **ステップ 2 モデルライブラリーにアクセスする**
- KServe マニフェストと展開フロー *Using NVIDIA NIM in your environment*

モデルレジストリの更新

デフォルトのモデル参照を更新して、プライベートレジストリをポイントします。モデルライブラリー構成と HM API /コンソールパスを操作します。

プロファイルキャッシュ NIM

一部の NVIDIA NIM モデルは、オフライン操作のためにローカルで利用できる必要があるランタイムプロファイルを使用します。プロファイルの検出とキャッシュ戦略については、NVIDIA のドキュメントに従ってください。

- NVIDIA NIM ドキュメント https://docs.nvidia.com/nim/
- ハブ使用パターン *Using NVIDIA NIM in your environment*
- *Air-gapped cache*

6.4.1.2.4 次のステップ

前提条件を満たしたら、次のことに進みます。

1. *GPU Recommendations for Default NIM Models* を確認して、ノードグループを計画します。
2. 最初のモデル *Deploy Agent Factory with Hybrid Manager UI* を展開します。
3. 最初の AI フロー *Getting started with Pipeline Designer* をビルドします。

アーキテクチャコンテキストについては、*Agent Factory Architecture on Hybrid Manager* を参照してください。

6.4.1.3 GPU Recommendations for Default NIM Models

6.4.1.3.1 概要

ハイブリッドマネージャーからは、AI モデルの 2 つの主なコンシューマーが存在します。

- AI ナレッジベースを作成および維持するための PG.AI Knowledge BaseAIDB Postgres 拡張機能。
- AI パイプラインとエージェントフローを構築するための Langflow。

6.4.1.3.2 デフォルトの NIM モデル

Model type	NIM model	NVIDIA NIM documented resource requirements
Text completion	nvidia/llama-3.3-nemotron-super-49b-v1	4 × L40S
Text embeddings	nvidia/llama-3.2-nemoretriever-300m-embed-v1	1 × L40S
Image embeddings	nvclip	1 × L40S
OCR	paddleocr	1 × L40S
Text reranking	llama-3.2-nv-rerankqa-1b-v2	1 × L40S

6.4.1.3.3 最小 GPU 要件

上記のデフォルトモデルに基づいて、それらを同時に実行できる最小は 8 × L40S GPU です。

6.4.1.3.4 クラウドマッピング

- AWS EKS 2 × `g6e.12xlarge` ノードを含むノードグループを推奨します。
- GCP GKE 2 × `a2-highgpu-4g` ノードを含むノードプールを推奨します。

注 GCP は、L40S GPU を提供していません。 A100 GPU を搭載した推奨 A2 ノードは、上記の NIM モデルでサポートおよび文書化されています。

6.4.2 Understand the AI Factory Architecture

6.4.2.1 Agent Factory Architecture on Hybrid Manager

6.4.2.1.1 アーキテクチャの概要

Agent Factory は、Hybrid Manager の Kubernetes インフラストラクチャ内のコンテナ化されたサービスのコレクションとして展開し、統合された *model governance* 、 *inference serving* 、および Langflow ベースの AI フロー開発を介してソブリン AI 機能を提供します。このアーキテクチャは、ローカル GPU リソースとオブジェクトストレージを活用して、顧客制御の Kubernetes クラスター内ですべての AI ワークロードを処理することにより、完全なデータ主権を保証します。

システムは、ガバナンスとオーケストレーションのためのコントロールプレーン、モデルサービングとフロー実行のランタイムレイヤー、モデルアーティファクトと **ナレッジベースと検索** のストレージレイヤーの 3 つのアーキテクチャレイヤーで動作します。これらのレイヤーは、Kubernetes API とカスタムリソースを介して統合され、プロジェクトとワークロード間の分離を維持しながら統合管理を提供します。

6.4.2.1.2 コアコンポーネント

モデルライブラリー

ステップ *2* モデルライブラリーにアクセスする は、プラットフォーム全体でモデルのライフサイクルとガバナンスを管理するコントロールプレーンサービスとして動作します。このサービスは、承認されたモデルの集中レジストリを維持し、モデルが運用環境に到達する前にセキュリティとコンプライアンスポリシーを適用します。

ライブラリは、いくつかの相互接続されたサービスで構成されています。

- 外部コンテナレジストリを監視するレジストリ同期サービス
- 組織ガバナンスルールに対してモデルを評価するポリシーエンジン
- モデルバージョン、パフォーマンスベンチマーク、および承認を追跡するメタデータサービス
- オブジェクトストレージバックエンドでモデルアーティファクトを管理するストレージインターフェイス

モデルのメタデータは、ハイブリッドマネージャーによって管理される PostgreSQL データベースに持続し、他のプラットフォームデータとの一貫性を保証します。このライブラリは、Kubernetes カスタムリソースを介してモデルをプロジェクト名前空間に公開し、一元的なガバナンスを維持しながら宣言的モデルの展開を可能にします。*Model Library explained* も参照してください。

推論サーバーインフラストラクチャ

推論サーバーは、プロジェクト名前空間内の KServe *InferenceServices* として展開され、特殊なコンテナポッドを介してスケーラブルな **モデルサービング** を提供します。これらのポッドは、さまざまなフレームワークとハードウェア構成に最適化されたモデルランタイムエンジンをカプセル化します。

推論ポッド構成には次のものが含まれます。

- モデルランタイムコンテナ
- GPU 割り当て、メモリ制限、および CPU 要件を定義するリソース仕様 *Setup GPU* および *Update GPU resources* を参照してください。
- モデルストレージと構成データに接続するボリュームマウント
- エンドポイント構成とランタイムパラメーターを含む環境変数
- liveness および readness プローブのヘルスチェック定義

自動スケーリング構成は、要求レイテンシー、GPU 使用率、キューの深さなどのメトリックに応答し、パフォーマンス目標を満たしながら最適なリソース使用率を保証します。デプロイメントオプションについては、**モデルの展開** および *Configure ServingRuntime* を参照してください。

Langflow ランタイム

Langflow FAQ は、ハイブリッドマネージャーの管理ワークロードとして実行され、フローを呼び出し可能サービスに変換するためのビジュアルフロービルダーと展開ライフサイクルを提供します。ランタイムはコンテナ化され、展開されたフローは分離された Kubernetes 名前空間でホストされます。

HM 内の Langflow アーキテクチャには以下が含まれます。

- フローがビルドおよびテストされる共有 Langflow エディタ環境
- 公開されたフローを長時間実行サービスとしてホストする展開ごとのランタイムコンテナ
- HM 管理リソースにフローを接続する EDB コンポーネント EDB Model Server、EDB Embeddings、EDB Knowledge Base など
- Kubernetes シークレットとして管理され、各展開の名前空間にスコープされる状態と資格情報

フローはクラスターローカルサービス DNS を介してモデルエンドポイントにアクセスし、Langflow ランタイム、モデルサーバーポッド、および Postgres クラスター間のすべてのトラフィックはプロジェクト名前空間内に残ります。完全なコンポーネントとデプロイメントリファレンスについては、*Langflow FAQ* を参照してください。

ストレージ

Agent Factory は、モデルアーティファクト、データセット、ナレッジベースにオブジェクトストレージを使用し、プライマリストレージバックエンドとして MinIO またはクラウドプロバイダーサービス S3、Azure Blob、GCS を使用します。これにより、コンピューティングがストレージから分離され、独立したスケーリングとコストの最適化が可能になります。

ストレージアクセスは、サービスアカウントの資格情報またはクラウドプロバイダーの ID メカニズムを介した認証を備えた標準化された S3 API を介して発生します。永続的なボリューム要求は、頻繁にアクセスされるモデルのローカルキャッシュを提供し、ネットワークオーバーヘッドを削減し、推論レイテンシーを改善します。

6.4.2.1.3 インフラストラクチャ

Agent Factory は、標準の Kubernetes プリミティブ GPU デバイスプラグイン、KServe、オブジェクトストレージ、およびサービスメッシュで実行されます。GPU セットアップ、ネットワークポリシー、HA 構成、およびモニタリングの詳細については、ハブリファレンスを参照してください。

- *GPU setup*
- *Create InferenceService*
- *Configure ServingRuntime*
- *Monitor InferenceService*

6.4.2.2 Use Cases and Personas for Agent Factory on Hybrid Manager

6.4.2.2.1 概要

ハイブリッドマネージャーのエージェントファクトリーは、主権 AI インフラストラクチャを介して多様な組織のユースケースを可能にします。組織はこれらの機能を活用して、Kubernetes 環境内のデータとモデルの完全な制御を維持しながら、インテリジェントなアプリケーションを構築します。

6.4.2.2.2 主なユースケース

内部ナレッジマネージメント

組織は AI アシスタントを展開し、従業員に内部文書、ポリシー、組織の知識への自然言語アクセスを提供します。

実装パターン

金融サービス企業は、コンプライアンス文書、規制当局への申告、および内部手順をナレッジベースにインデックス付けします。従業員は会話型インターフェイスを使用してこれらのシステムを照会し、データ主権を維持しながらソース引用付きの正確な回答を受け取ります。

技術コンポーネント

- SharePoint、Confluence、およびデータベースコンテンツのインデックスを作成するナレッジベース
- クエリの理解と応答生成のための LLM エンドポイント
- アクセス制御のためのロールベースのフィルタリングを備えた検索システム

実装ガイダンスについては、 *Getting started with Pipeline Designer* を参照してください。

カスタマーサービスオートメーション

AI を活用したカスタマーサポートシステムは、日常的な問い合わせを処理しながら、複雑な問題を人間のエージェントにエスカレーションします。

実装パターン

E コマース企業は、安全な API を介して注文システム、製品カタログ、返品ポリシーにアクセスするアシスタントを展開します。これらのアシスタントは、品質保証のための完全な監査証跡を維持しながら、顧客からの問い合わせの大部分を自動的に解決します。

技術コンポーネント

- 注文管理システム統合のための REST API ツール
- 製品ドキュメントやよくある質問を含むナレッジベース

実用的な例については、 *Langflow quickstart* を参照してください。

Enterprise RAG Systems

検索拡張生成アプリケーションは、組織データの LLM 応答を基礎付け、精度を維持しながら幻覚を削減します。

実装パターン

医療機関は、臨床ガイドライン、研究論文、および治療プロトコールを介して RAG システムを実装しています。医療専門家は、ソース資料への引用付きで証拠に基づいた推奨事項を受け取り、トレーサビリティを保証しながら臨床上の意思決定をサポートします。

技術コンポーネント

- pgvector を使用して PostgreSQL に保存されたベクター埋め込み
- セマンティックマッチングとキーワードマッチングを組み合わせたハイブリッド検索
- 結果の最適化のためのモデルの再ランキング

アーキテクチャの詳細については、 ナレッジベースと検索 を参照してください。

AI 対応 API

組織は、既存のアプリケーションと統合するための安全な API を介して AI 機能を公開します。

実装パターン

ソフトウェア会社は、標準の REST インターフェイスを介してアクセスできるモデルエンドポイントを展開することにより、製品内のインテリジェント機能を提供します。これらの API は、予測可能なパフォーマンス特性を維持しながら、ドキュメントの要約、感情分析、コンテンツ生成などのタスクを処理します。

技術コンポーネント

- OpenAI 互換エンドポイントを公開する InferenceServices
- 認証とレート制限を備えた API ゲートウェイ
- 高可用性のためのモデルレプリカ間のロードバランシング

デプロイメント手順については、 *Model Serving Guide* を参照してください。

6.4.2.2.3 ユーザーペルソナ

プラットフォームエンジニア

プラットフォームエンジニアは、エージェントファクトリーのインフラストラクチャを管理し、信頼性の高いモデル提供と最適なリソースの使用率を保証します。

主な責任

- GPU ノードとリソース割り当ての構成
- InferenceServices の展開と監視
- モデルライブラリとレジストリ接続を管理する
- セキュリティポリシーとネットワーク構成の実装

主要なワークフロー

1. *GPU Setup Guide* に従って GPU リソースをプロビジョニング
2. *Model Library Documentation* ごとにモデルレジストリを構成する
3. *Observability (Agent Factory on HM)* を使用して推論パフォーマンスをモニタする

データサイエンティスト

データサイエンティストはエージェントファクトリーを活用して、インフラストラクチャの複雑さを管理せずにモデルを展開し、インテリジェントなアプリケーションを構築します。

主な責任

- ナレッジベースの作成と最適化
- RAG アプリケーションの取得戦略の構成
- モデルのパフォーマンスと精度を評価する
- デザインアシスタントの動作とツールの統合

主要なワークフロー

1. *Getting started with Pipeline Designer* を使用してナレッジベースパイプラインを構築する
2. Langflow フローでの取得の構成 *EDB Knowledge Base*
3. フローをサービスとして展開する *Flow deployment*

アプリケーション開発者

開発者は、API と SDK を介して AI 機能をビジネスアプリケーションに統合します。

主な責任

- モデルエンドポイントをアプリケーションに統合する
- AI を活用した機能のカスタムツールを構築
- エラー条件とフォールバック戦略を処理します

主要なワークフロー

1. *KServe Integration* を介してモデルエンドポイントにアクセス
2. Python からモデルを呼び出します *Python Client Guide*
3. 展開された Langflow フローを HTTP または MCP で呼び出します *Accessing deployed flows*

ビジネスアナリスト

ビジネスアナリストは、Langflow のビジュアルフロービルダーを使用して、コーディングの専門知識なしで AI アプリケーションを作成します。

主な責任

- フロー動作と知識ドメインを定義する
- データソース接続の構成
- フロー応答のテストと検証
- 使用パターンと効果を監視する

主要なワークフロー

1. Langflow ビジュアルエディター *Langflow quickstart* を使用してフローをビルドします
2. 事前に構築されたテンプレートを開始点として使用します *Langflow templates*
3. 他の人が使用するフローを公開および展開します *Flow deployment*

6.4.2.2.4 業界固有の実装

金融サービス

ユースケース取引パターンを分析し、必要なレポートを生成する規制コンプライアンスアシスタント。

コンポーネント

- 規制文書のナレッジベース
- トランザクションデータベースにアクセスするツール
- コンプライアンス追跡のための監査ログ

ヘルスケア

ユースケース証拠に基づいた治療の推奨を提供する臨床決定支援システム。

コンポーネント

- 医学文献とガイドラインに関する RAG
- 電子医療記録との統合
- 患者データの厳密なアクセス制御

製造業

ユースケースセンサーデータと技術マニュアルを使用して、機器の問題を診断するメンテナンスアシスタント。

コンポーネント

- 機器ドキュメントのナレッジベース
- IoT センサーからのリアルタイムデータの統合
- 予知保全モデルのエンドポイント

6.4.2.2.5 実装の考慮事項

スケーラビリティープランニング

さまざまなユースケースには、さまざまなレベルのスケールが必要です。

- **大規模なカスタマーサービス**自動スケーリングを備えた複数のモデルレプリカ
- **内部ナレッジシステム** キャッシュ最適化による中程度のスケーリング
- **特殊なアシスタント** 専用のリソースを使用した単一のインスタンス

セキュリティ要件

ユースケースはセキュリティ構成を決定します。

- **パブリック向けアプリケーション** 認証を使用した外部イングレス
- **内部システム** RBAC を使用したクラスターローカルアクセス
- **規制対象業界** 監査ログと暗号化の強化

パフォーマンスの最適化

ユースケースの特性に基づいて最適化する

- **リアルタイムアプリケーション**エッジキャッシングを使用して提供する低レイテンシーモデル
- **バッチ処理** より大きなバッチサイズでのより高いスループット
- **インタラクティブアシスタント**バランスの取れたレイテンシーとスループット

6.4.2.2.6 ユースケース別のスタートガイド

ナレッジマネージメントの場合

1. ナレッジベースパイプライン *Getting started with Pipeline Designer* を作成します
2. フローからナレッジベースを照会します *EDB Knowledge Base*
3. RAG フローをエンドツーエンドでビルドする *Langflow quickstart*

カスタマーサービス用

1. 会話フローを構築します *Langflow quickstart*
2. エージェントフローのリファレンスとして Simple Migration Agent テンプレートを試してください *Langflow templates*
3. フローを呼び出し可能サービスとして展開します *Flow deployment*

API 開発の場合

1. *Deploy Agent Factory with Hybrid Manager UI* を介してモデルを展開する
2. エンドポイントにアクセスします *Python Client Guide*
3. パフォーマンスを監視する *Observability (Agent Factory on HM)*

6.4.2.2.7 追加リソース

関連ドキュメント

- *Hybrid Manager Overview*
- *Agent Factory Architecture on Hybrid Manager*
- *Langflow FAQ*
- *Pipeline Designer concepts*

6.4.2.3 Frequently Asked Questions - Agent Factory on Hybrid Manager

6.4.2.3.1 目次

- プラットフォーム機能
- インストールとセットアップ
- *Model Management*
- *Langflow FAQ*
- 運用とメンテナンス
- *Troubleshooting Langflow in Hybrid Manager*
- セキュリティとコンプライアンス

- パフォーマンスとスケーリング

6.4.2.3.2 プラットフォーム機能

ハイブリッドマネージャーはどのようなタイプのモデルをサポートしていますか?

ハイブリッドマネージャーは、NVIDIA NIM コンテナを介してのみ大規模言語モデル LLM 展開をサポートします。従来の機械学習モデル分類、回帰、時系列予測は、このリリースではサポートされていません。

サポートされている NVIDIA NIM モデルカテゴリ

- **テキスト生成** チャットおよび完了タスク用の大規模な言語モデル
- **テキスト埋め込み**セマンティック検索および RAG アプリケーションのモデル
- **Text Reranking** 検索結果最適化のモデル
- **マルチモーダルモデル** CLIP および OCR 機能を含むビジョンモデル

カスタムモデルを展開できますか?

カスタムモデルは、ハイブリッドマネージャーと互換性を保つために NVIDIA NIM コンテナとしてパッケージ化する必要があります。標準の機械学習フレームワーク scikit-learn、XGBoost、従来の ML の TensorFlow はサポートされていません。カスタム LLM は、NIM コンテナ仕様と API 標準に準拠している場合に、展開できます。

カスタム NIM 展開手順については、 *Private Registry Integration* を参照してください。

Agent Factory とクラウド AI サービスの違いは何ですか?

Agent Factory は、AI 操作に対する完全な主権を提供します。

- モデルは Kubernetes インフラストラクチャ内で実行されます
- データは組織の境界内に残ります
- 推論のための外部 API 依存関係はありません
- 規制コンプライアンスのための完全な監査証跡

6.4.2.3.3 インストールとセットアップ

インフラストラクチャの最小要件は何ですか?

コア要件

- NVIDIA GPU オペレーターを使用した Kubernetes 1.27+
- NIM コンテナ L40S、A100、H100 と互換性のある NVIDIA GPU
- モデルアーティファクト用の 100GB 以上のオブジェクトストレージ
- NVIDIA NGC レジストリまたはエアギャップ構成へのネットワーク接続

NIM モデルタイプ別の GPU 要件

- テキスト補完 Nemotron Super 49B 4 x L40S GPU

- テキスト埋め込み 1 x L40S GPU
- テキストのリランキング 1 x L40S GPU
- ビジョンモデル 1 x L40S GPU

包括的な仕様については、 *Prerequisites for Agent Factory on Hybrid Manager* を参照してください。

NIM モデルの GPU ノードを構成するにはどうすればよいですか?

GPU ノードの準備には以下が含まれます。

1. クラスターに NVIDIA GPU オペレーターをインストールする
2. `nvidia.com/gpu=true` で GPU ノードにラベルを付ける
3. 専用のスケジューリングに GPU テイントを適用する
4. NIM 要件の CUDA 互換性を確認する

GPU Setup Documentation で利用可能な詳細な手順。

Agent Factory はエアギャップ環境で動作できますか?

エアギャップ展開には事前の準備が必要です。

1. NVIDIA NIM イメージをプライベートレジストリにミラーリングする
2. モデルプロファイルのダウンロードとキャッシュ
3. プロファイルをオブジェクトストレージにアップロードする
4. プライベートレジストリアクセス用にモデルライブラリーを構成する

エアギャップ展開 に文書化されている手順を完了します。

6.4.2.3.4 モデル管理

NVIDIA NIM モデルを展開するにはどうすればよいですか?

NIM モデル展開ワークフロー

1. HM コンソールでモデルライブラリーにアクセスします
2. カタログから NVIDIA NIM モデルを選択します
3. リソースの構成 GPU 割り当て、メモリ、レプリカ
4. InferenceService をプロジェクト名前空間に展開する
5. 生成されたエンドポイントを介したアクセス

ステップバイステップガイド *Create InferenceService* 。

デフォルトで使用可能な NVIDIA NIM モデルはどれですか?

デフォルトの NIM モデル

- **llama-3.3-nemotron-super-49b-v1** 高度な推論とチャット
- **llama-3.2-nemoretriever-300m-embed-v1** テキスト埋め込み
- **llama-3.2-nv-rerankqa-1b-v2** クエリドキュメントのリランキング
- **nvclip** マルチモーダル埋め込み
- **paddleocr** 光学式文字認識

NIM モデルバージョンを管理するにはどうすればよいですか?

バージョン管理戦略

- モデルライブラリーは、各 NIM イメージのバージョンタグを維持します
- ブルーグリーン展開により、ゼロダウンタイムの更新が可能
- カナリア展開により、段階的なトラフィックのシフトが可能
- InferenceService 構成の更新を介したロールバック

6.4.2.3.5 Langflow

Langflow は、ハイブリッドマネージャーの AI フロービルダーであり、以前の Griptape ベースの Gen AI Builder を置き換えます。HM 管理モデル、ナレッジベース、Postgres クラスターに直接接続する EDB コンポーネントを使用して、AI パイプラインとフローを構築します。

Langflow 固有の質問コンポーネント、グローバル変数、ナレッジベース、モデルの使用状況、フロープロモーション、および一般的なエラーについては、*Langflow FAQ* を参照してください。

6.4.2.3.6 運用とメンテナンス

NIM モデルのパフォーマンスをモニタするにはどうすればよいですか?

モニタリングには以下が含まれます。

メトリックコレクション

- 推論レイテンシーの Prometheus メトリック
- GPU 使用率とメモリ消費量
- トークン生成のスループット
- 要求の成功率

ビジュアライゼーション

- HM コンソールに統合された Grafana ダッシュボード
- モデル固有のメトリックのカスタムパネル
- SLA 違反のアラート構成

参照 *Observability (Agent Factory on HM)* 。

NIM モデルの更新はどのように処理すればよいですか?

運用展開の更新手順

1. **検証** 開発名前空間に新しいバージョンを展開します
2. **テスト** パフォーマンステストと精度テストを実行します
3. **展開** カナリアまたはブルーグリーン戦略を実装します
4. **モニタリング** 移行中のメトリックの追跡
5. **決定** メトリックに基づいて完全なロールアウトまたはロールバック

6.4.2.3.7 障害対応

NIM モデルの起動に失敗します-診断手順は?

一般的な初期化障害

1. **GPU の使用不可** GPU リソースがモデル要件と一致することを確認します。
2. **イメージのプルの失敗** NGC 資格情報とネットワーク接続を確認します
3. **プロファイルキャッシュがありません**エアギャップセットアップでプロファイルを利用できることを確認
4. **メモリ不足** モデルサイズのメモリ割り当てを検証する

診断コマンド

```
kubectl describe inferenceservice <name> -n <namespace>
kubectl logs <pod-name> -n <namespace>
kubectl get events -n <namespace>
```

高い推論レイテンシー - 最適化戦略?

パフォーマンス最適化のアプローチ

- **バッチ処理** スループットを最適化するためにバッチサイズを増加します
- **モデル量子化** サポートされている場合は INT8 量子化を使用します
- **応答キャッシュ**頻繁なクエリーをアプリケーションレイヤーでキャッシュします
- **ホリゾンタル スケーリング**負荷分散のために追加のレプリカを展開します

RAG アプリケーションの取得品質は低いですか?

正しい埋め込みモデルが使用されていること、チャンクサイズがコンテンツに適切であること、およびすべてのドキュメントがインデックス化されていることを確認します。AIDB ナレッジベース診断を含む Langflow 固有のガイダンスについては、*RAG* フローでの取得品質を向上させるにはどうすればよいですか? を参照してください。

6.4.2.3.8 セキュリティとコンプライアンス

アクセス制御はどのように実装すればよいですか?

AI リソースのロールベースのアクセス制御には、次のものが含まれます。

- 名前空間とリソース権限のための Kubernetes RBAC
- 展開承認のためのモデルライブラリーのアクセス制御
- 外部エンドポイントアクセスの API キー管理
- サービス間通信のネットワークポリシー

どのような暗号化が実装されていますか?

暗号化カバレッジ

- **保存時** Kubernetes シークレット暗号化、データベース暗号化
- **転送中** API 呼び出しの TLS、サービスメッシュ内の mTLS
- **モデルアーティファクト** 暗号化されたオブジェクトストレージ
- **ナレッジベース** PostgreSQL の暗号化されたベクターストレージ

どの操作が監査されますか?

監査ログキャプチャ

- NIM モデルの展開と構成の変更
- 推論要求設定可能な詳細レベル
- ナレッジベースのクエリと更新
- AI リソースの管理操作

6.4.2.3.9 パフォーマンスとスケーリング

リソースクォータはどのように構成すればよいですか?

リソースクォータは、名前空間レベルでのリソースの枯渇を防ぎます。プロジェクト要件と利用可能なインフラストラクチャ容量に基づいて、GPU クォータ、メモリ制限、およびストレージ制約を構成します。

垂直方向と水平方向のどちらの場合にスケーリングする必要がありますか?

ホリゾンタル スケーリング 追加のレプリカ

- 高い同時要求量
- ステートレス推論ワークロード
- 負荷分散の要件

垂直スケーリング インスタンスごとのリソースの増加

- 大規模なモデルのメモリ要件

- バッチ処理の最適化
- 単一要求のレイテンシーの最小化

ハイブリッドマネージャーはどのようなモデルサイズをサポートできますか?

モデルサイズのコンストレイン

- **シングル GPU** 最大 13B パラメーターのモデル
- **マルチ GPU** テンソル並列処理を使用して最大 70B+パラメーターをモデル化
- **メモリ制限** GPU ごとに 80GB A100、48GB L40S

NVIDIA NIM は、利用可能なリソースに基づいて、モデルのシャーディングと並列処理を自動的に処理します。

6.4.2.3.10 追加リソース

はじめに

- *Prerequisites for Agent Factory on Hybrid Manager*
- *Agent Factory Architecture on Hybrid Manager*

実装ガイド

- モデルサービング
- *Langflow FAQ*
- ナレッジベースと検索 AIDB
- Data pipelines AIDB

トラブルシューティングリソース

- *Model Verification*

ここで対処されていない問題については、EDB サポートに連絡するか、*Troubleshooting Langflow in Hybrid Manager* を参照してください。

6.4.3 Model Management

6.4.3.1 Model Management on Hybrid Manager

6.4.3.1.1 目的と利点

ハイブリッドマネージャー内のモデル管理は、Kubernetes インフラストラクチャで実行されている AI モデルに一元的なガバナンスと展開機能を提供します。このシステムにより、組織はエンタープライズグレードの モデルサービング インフラストラクチャを活用しながら、AI 機能の完全な制御を維持できます。

この統合は、AI を大規模に展開する組織の重要な要件に対応します。承認されたレジストリを介したモデルガバナンス、GPU アクセラレーションを使用したスケーラブルな推論サービング、およびハイブリッドマネージャーのコントロールプレーンを介した統合管理。制御されたインフラストラクチャ内でモデルを実行することにより、最先端の AI 機能にアクセスしながら、データ主権を維持します。

6.4.3.1.2 コアコンセプト

モデルライブラリー

ステップ *2* モデルライブラリーにアクセスする は、AI モデルイメージの集中ガバナンスシステムとして機能します。Hybrid Manager のアセットライブラリーストラクチャ内で動作し、運用展開の準備ができている検証済みモデルの厳選されたビューを提供します。

ライブラリは、多段階のガバナンスを実装します。

- 信頼できるコンテナレジストリからの自動同期
- セキュリティスキャンと脆弱性評価
- 組織ポリシーに基づいた承認ワークフロー
- バージョン管理とドキュメントのメタデータ管理

ライブラリ内のモデルは、Langflow フロー、パイプラインデザイナーナレッジベース、カスタム推論アプリケーションを含むすべてのエージェントファクトリー機能を強化します。ライブラリのガバナンスフレームワークを通じて検証されたモデルのみが運用環境に到達できます。

モデルサービング

モデルサービング は、Kubernetes クラスター内の KServe を使用して、承認されたモデルをスケーラブルな推論エンドポイントに変換します。このインフラストラクチャは、自動スケーリング、状態管理、およびリソースの最適化を備えた実稼働グレードのモデルの展開を提供します。

主要なサービス提供機能には次のものが含まれます。

- デプロイされたモデルエンドポイントを定義する ***InferenceService*** リソース
- さまざまなモデルフレームワーク用に最適化された ***ServingRuntime*** 構成
- 高パフォーマンスの推論のための GPU の割り当てとスケジューリング —*GPU Recommendations for Default NIM Models* を参照してください
- 認証を使用した内部および外部エンドポイントアクセス

管理インターフェイス

Hybrid Manager は、Web コンソールを介して統合管理を提供し、完全な構成可能性を維持しながら Kubernetes の複雑さを抽象化します。このインターフェイスにより、次のことが可能になります。

- ライブラリからサービングまでモデル展開のビジュアルワークフロー
- リソースの割り当てとスケーリング構成
- 推論メトリックと GPU 使用率のモニタリングダッシュボード
- アクセス制御とエンドポイント管理

6.4.3.1.3 実装ワークフロー

モデル登録

組織は、信頼できるモデルソースへのリポジトリ接続を構成することから始めます。モデルライブラリーは、定義されたルールに基づいて外部レジストリと同期し、新しいモデルバージョンを自動的に検出および検証します。

```
External Registry → Repository Rules → Security Scanning → Model Library
```

リポジトリルールは、環境に入るモデルを決定し、インジェスト時点で組織ポリシーを実装します。この自動化されたアプローチにより、ガバナンス標準を維持しながら、手動オーバーヘッドが削減されます。

モデルの展開

検証されたモデルは、サービスインフラストラクチャを構成するガイド付きワークフローを介して展開されます。

1. **モデル選択**バージョン、パフォーマンス特性、リソース要件を含むメタデータを使用してライブラリ内の使用可能なモデルを参照します。
2. **実行時構成**モデルフレームワーク vLLM、TensorRT-LLM、カスタム用に最適化された *ServingRuntimes* を選択または作成します
3. **リソース割り当て**予想されるワークロードに基づいて、GPU、メモリ、および CPU 要件を定義します
4. **エンドポイント構成**認証を使用した内部クラスターアクセスまたは外部 API エンドポイントを設定します *Access KServe endpoints* を参照してください。

システムは、KServe が管理する InferenceService リソースを作成し、ポッドのスケジューリング、ヘルスモニタリング、トラフィックルーティングを自動的に処理します。

運用管理

導入されたモデルは、要求に基づいた自動スケーリングを使用した継続的な監視の下で動作します。ハイブリッドマネージャーは、次のことを介して可視性を提供します。

- レイテンシーやスループットを含むリアルタイムの推論メトリック
- リソース最適化のための GPU 使用率追跡
- プロアクティブなメンテナンスのエラー率と状態
- リソース消費に基づいたコスト分析

6.4.3.1.4 デプロイされたモデルを使用する

モデルクラスターが実行されたら、それを使用する 3 つの方法があります。

アプリケーションから

アプリケーションは、KServe InferenceServices を介して展開されたモデルエンドポイントを直接呼び出します。各エンドポイントは、チャット完了、埋め込み、またはその他の推論タスク用の標準の OpenAI 互換 REST API を公開します。

- *Access KServe endpoints*
- *Quickstart (Python)*

AIDB から SQL パターン

AIDB を使用すると、 SQL からモデルを呼び出し、Postgres 内で直接利用できるようになります。パイプラインを埋め込んだり、データベース内推論を有効にしたりするのに役立ちます。

- *Using models via OpenAI-compatible SQL calls*
- *Using NVIDIA NIM in AIDB*

ハイブリッドマネージャーの詳細

モデルがハイブリッドマネージャーを介して展開される場合

- **サービス URL.** 各モデルは、HM プロジェクト内の内部 KServe エンドポイントとして公開されます。URL は、モデルライブラリーまたはモデル提供の詳細ページに表示されます。
- **認証.** エンドポイントはプラットフォームによって保護されています。同じプロジェクト内で実行されているアプリケーションは、それらに直接到達できます。外部アクセスの場合、HM 入力およびプロジェクトスコープの資格情報を使用して認証を構成します。
- **可観測性。**要求とログは HM 可観測性にフローし、使用メトリック、レイテンシー、エラー追跡を提供します。

6.4.3.2 Deploy Agent Factory with Hybrid Manager UI

6.4.3.2.1 概要

このガイドでは、ハイブリッドマネージャー HM Web インターフェイスを介した Agent Factory コンポーネントの展開手順を提供します。この展開プロセスには、GPU インフラストラクチャのセットアップ、NVIDIA NIM モデルの展開、および Gen AI アプリケーションの構成が含まれます。

6.4.3.2.2 前提条件

インフラストラクチャ要件

Agent Factory コンポーネントを展開する前に

- GPU ノードが構成された Kubernetes クラスター
- NVIDIA GPU オペレーターのインストール
- NVIDIA NGC レジストリまたはプライベートレジストリへのアクセス
- モデルプロファイルのオブジェクトストレージエアギャップ展開

GPU ノード構成

GPU ノードが NIM 要件を満たしていることを確認します。*GPU Recommendations for Default NIM Models* を参照してください。

6.4.3.2.3 ステップ 1 レジストリ認証を構成する

インターネットに接続された展開

インターネットアクセスのあるクラスターの場合、NVIDIA NGC 認証を構成します。

1. NVIDIA NGC Portal から NGC API キーを取得します

2. HM UI を介して認証シークレットを作成します。 - **Project Settings** > **Secrets** に移動します。 - NGC_API_KEY を使用して `nvidia-nim-secrets` を作成します。 - `ngc-cred` Docker レジストリシークレットを作成します。

または、kubectl を使用します。

```
NGC_API_KEY=<your-ngc-api-key>

#  Create runtime secret

kubectl -n default create secret generic nvidia-nim-secrets \
    --from-literal=NGC_API_KEY=${NGC_API_KEY}

kubectl -n default annotate secret nvidia-nim-secrets \
    replicator.v1.mittwald.de/replicate-to=m-.*

#  Create image pull secret

kubectl -n default create secret docker-registry ngc-cred \
    --docker-server=nvcr.io \
    --docker-username=$oauthtoken \
    --docker-password=${NGC_API_KEY}

kubectl -n default annotate secret ngc-cred \
    replicator.v1.mittwald.de/replicate-to=m-.*
```

エアギャップ展開

インターネットアクセスのない環境の場合

1. **NIM イメージのミラーリング** skopeo を使用して、必要なイメージをプライベートレジストリにコピーします。

2. **モデル URL の更新**プライベートレジストリの場所を参照するように HM を構成します。

3. **キャッシュプロファイル**モデルプロファイルをダウンロードしてオブジェクトストレージに保存します。

4. **ストレージパスの構成**モデルの展開中にキャッシュされたプロファイルを参照します。

詳細な手順については、 *Air-Gapped Configuration* を参照してください。

6.4.3.2.4 ステップ 2 モデルライブラリーにアクセスする

Model ライブラリーインターフェイスに移動します。

1. Hybrid Manager コンソールにログインします。
2. プロジェクトセレクターからプロジェクトを選択します。
3. **Agent Factory** → **Model ライブラリー** に移動します。

モデルライブラリーには、使用可能な NVIDIA NIM モデルが表示されます。

- **llama-3.3-nemotron-super-49b-v1** 高度な推論とチャット機能 128K コンテキスト
- **llama-3.2-nemoretriever-300m-embed-v1** 高品質のテキスト埋め込み
- **llama-3.2-nv-rerankqa-1b-v2** 多言語クエリドキュメントのリランキング
- **paddleocr** 超軽量 OCR システム
- **nvclip** 画像とテキストのマルチモーダル埋め込み

6.4.3.2.5 ステップ 3 モデルサーバークラスターを展開する

導入用モデルの選択

1. モデルライブラリーで **Deploy Model** をクリックします。
2. 利用可能なオプションからターゲットモデルを選択します。
3. モデル要件とドキュメントリンクを確認します。

展開パラメーターの構成

モデルサーバークラスター設定を構成します。

インスタンス構成

- **サーバーインスタンス** デフォルト 1 高可用性のための増加
- **最小インスタンス** 1 サポートされている場合はスケールからゼロの 0
- **最大インスタンス** 負荷要件に基づきます

リソースの割り当て

- **メモリ** モデル要件に基づいて構成します
- テキスト補完モデル 64-128 GB
- 組み込みモデル 32-64 GB
- **CPU** 通常はデフォルト値で十分
- **GPU** 文書化された要件と一致する

- llama-3.3-nemotron 4 GPU
- その他のモデル 1 GPU 通常

スケーリング構成

- **同時要求のしきい値** 自動スケーリングトリガーの構成
- **Scale to Zero** サポートされている場合に有効にします初期リリースでは使用できない場合があります

デプロイモデル

1. 構成の概要を確認します。
2. **Deploy** をクリックして、展開を開始します。
3. モデルサーバークラスタービューで展開ステータスを監視します。

展開により作成されるもの

- KServe InferenceService リソース。
- リソースが構成された予測子ポッド。
- モデルアクセスのサービスエンドポイント。

6.4.3.2.6 ステップ 4 展開ステータスを監視する

View Model Server クラスター

Agent Factory → **Model Server Clusters** に移動して、以下を表示します。

- クラスター表示名
- ロードバランサーのイングレス URI
- 展開ステータス アクティブ/正常、保留中、失敗
- モデルの詳細とタグ
- リソース使用率メトリック

アクセスクラスターの詳細

表示するクラスター名をクリックします。

- 詳細な構成パラメーター
- リアルタイムの健康メトリック
- Grafana ダッシュボードの統合
- 推論レイテンシーチャート
- システム状態インジケーター

6.4.3.2.7 ステップ 5API アクセスを構成する

API トークンを生成する

デプロイされたモデルへの外部アクセスの場合

1. **Agent Factory** → **API Token Management** に移動します
2. **Create Token** をクリックします
3. トークン参照名を提供します
4. 生成されたトークンを安全に保存する

API トークンにより、次のことが可能になります。

- RAG アプリケーション認証
- 外部サービスの統合
- プログラマティックモデルアクセス

アクセスエンドポイント

モデルは OpenAI 互換のエンドポイントを公開します。

内部アクセス クラスター内

```
http://<service-name>.<namespace>.svc.cluster.local/v1/chat/completions
```

外部アクセス イングレスを使用する場合

```
https://<ingress-url>/v1/chat/completions
```

6.4.3.2.8 ステップ 6 Gen AI アプリケーションをビルドする

Hybrid Manager での Gen AI アプリケーションのビルドは、 **Langflow** を介して実行され、HM コンソールの **Launchpad** からアクセスできます。Langflow は、HM 管理モデル、ナレッジベース、Postgres クラスターに直接接続する EDB コンポーネントを備えたビジュアルフローエディターを提供します。

最初の Gen AI アプリケーションをビルドするための段階的なウォークスルーについては、*Langflow quickstart* を参照してください。

EDB コンポーネント、フローの展開、およびインスタンス間のフローの共有の完全なリファレンスについては、*Langflow FAQ* を参照してください。

6.4.3.2.9 ステップ 7 モデルの更新と管理

モデルサーバークラスターの編集

実行中のクラスターを変更するには

1. **Model Server Clusters** に移動します
2. 編集するクラスターを選択します

3. パラメーターを調整します。

- インスタンス数
- リソースの割り当て
- スケーリングしきい値

1. 変更を適用するローリング更新をトリガーします

ローリングアップデート

更新は、次のことを介して可用性を維持します。

- 再起動前の接続ドレイン
- インスタンス間の状態検証
- 失敗時の自動ロールバック

6.4.3.2.10 Langflow フローの高可用性と分離

Langflow フローの高可用性、プロジェクトレベルの分離、安定したルーティング、およびアクセス制御は、Langflow フロービルダーのプロパティではなく、**展開**フローのプロパティです。フロービルダーは単一の共有インスタンスです。そこで実行するフローはプロジェクトごとに分離されず、独立してスケールすることはできません。独自のエンドポイントを使用して、分離された高可用性の独立してスケーラブルなサービスとしてフローを実行するには、公開および展開します。*Flow deployment* を参照してください。これには、デプロイされたフローのレプリカ、エンドポイント、アクセスレベル、および資格情報の処理が含まれます。

Langflow フローが依存するモデルサーバークラスターの場合、 **Estate** → **InferenceServices** → 推論サービスを選択 → **Edit** → **Inference Service Instances** で実行中のインスタンスの数を調整します。

6.4.3.2.11 障害対応

一般的な展開の問題

モデルの開始に失敗します

- GPU の可用性が要件と一致することを確認します
- レジストリ認証シークレットの確認
- ポッドイベントとログを確認する

高い推論レイテンシー

- バッチサイズパラメーターを調整する
- GPU 割り当てを増加します
- 負荷分散のためのスケールレプリカ

API トークンの問題

- トークンの有効期限がまだ残っていることを確認します

- ネットワークポリシーの確認
- イングレス構成の確認

詳細な診断については、 *Troubleshooting Langflow in Hybrid Manager* を参照してください。

6.4.3.2.12 追加リソース

リファレンスドキュメント

- *GPU Setup Guide*
- *Create InferenceService*
- *Model Serving Concepts*

NVIDIA モデルドキュメント

- Llama 3.3 Nemotron
- Arctic Embed
- NV-ReRankQA
- PaddleOCR
- NV-CLIP

6.4.3.3 External inference services

6.4.3.3.1 コンテキスト

外部推論サービスは、ハイブリッドマネージャー HM を、クラスターの外部でホストされるリモートモデルプロバイダー OpenAI、Google Gemini、Anthropic、NVIDIA NIM などに接続します。プロバイダーの URL、モデル名、および API キーを 1 回構成します。HM は、資格情報を Kubernetes シークレットに保存し、すべてのダウンストリーム要求の認証を透過的に処理します。

このページでは以下について説明します。

- **推論サービスリスト** の表示
- 新しい外部推論サービスの登録
- 登録したサービスの詳細の取得
- 登録済みサービスの更新
- サービスの登録解除

6.4.3.3.2 推論サービスリスト

HM プロジェクトの **Estate** → **Inference Services** メニューから **Inference Services list** ページを開きます。

リストには、各サービスの名前、モデル、および現在のステータスが表示されます。ステータスが**準備完了** でない場合、短い **ステータスメッセージ**で原因が説明されます。

ステータス

Status	Meaning
Ready	The service is healthy and accepting requests.
Failed	The most recent health check failed. The service detail page shows a **Status Message** with the diagnosis (for example, *API key is missing or invalid*, *endpoint unreachable*, or *account quota exhausted*).
Pending	The endpoint hasn't yet been verified by the background health check. Typically seen during the first few minutes after an HM restart, while the cache warms up. The **Status Message** reads *Remote connection check is pending — upstream has not yet been verified*.
Unknown	The underlying Kubernetes resources haven't reported a status yet — usually a transient state right after creation.

ステータスは、バックグラウンドのヘルスチェックによって 5 分ごとに更新されます。サービスを作成または編集すると、接続プローブがインラインで実行されるため、結果はすぐに反映されます。

6.4.3.3.3 外部推論サービスを登録する

HM プロジェクトで **Estate** → **Quick Actions** → **Register external Inference Service** に移動します。

> **Tip**
>
> **Quick Actions**メニューを介して、推論サービスリストページからフォームに到達することもできます。

前提条件

登録する前に、次のことを確認してください。

- プロバイダーのベース URL スキームとホスト、後続の`/v1`なし。
- プロバイダーが想定するとおりのモデル名大文字と小文字が区別されます。
- プロバイダーの有効な API キー。
- HM クラスターから上流のホスト名までのネットワーク到達可能性。HM 管理者は、プロバイダーのドメインへの下りを許可する必要がある場合があります。

プロバイダー別のクイックリファレンス

この表を使用して、最も一般的なプロバイダーのフォームに入力します。列は、以下で説明するフィールドに直接マッピングされます。

Provider	Model base URL	API protocol version	Example model name	Notes
OpenAI	*https://api.openai.com*	*OPENAI_V1*	*gpt-4o-mini*	登録時に検証される API キー。
OpenRouter	*https://openrouter.ai/api*	*OPENAI_V1*	*openai/gpt-4o-mini*	登録時に API キーが検証されませんでした。
Google Gemini	*https://generativelanguage.g*	*GEMINI_V1_BETA*	*gemini-2.5-pro*	登録時に検証される API キー。
Anthropic	*https://api.anthropic.com*	*ANTHROPIC_V*	*claude-sonnet-4-5*	登録時に検証される API キー。
NVIDIA NIM	*https://integrate.api.nvidia.c*	*OPENAI_V1*	*meta/llama-3.1-8b-instruct*	登録時に API キーが検証されませんでした。
Hugging Face	*https://router.huggingface.co*	*OPENAI_V1*	*openai/gpt-oss-120b*	メインサイト `huggingface.co`ではなく、ルーターサブドメイン `router.huggingface.co`を使用します。登録時に API キーが検証されませんでした。
Self-hosted / vLLM	Your internal service URL, for example *http://vllm-svc.inference:8000*	*OPENAI_V1*	The model name your vLLM server is serving	vLLM が認証されずに実行される場合、登録時に API キーが検証されません。
Other HM (federation)	Another HM's external inference endpoint URL	*OPENAI_V1*	Same model name as on the upstream HM	**安全でない接続を許可** を有効にします。以下の *Allow Insecure Connection* **オプショナル、デフォルトオフ** を参照してください。

ここにリストされていないプロバイダーについては、以下の **フォームフィールド** を参照してください。

フォームフィールド

外部サービス名必須

HM 内のこのサービスの一意の識別子。DNS スタイルの命名ルールに従う必要があります。

- 小文字と数字のみ。
- ハイフン - セグメント内では使用できますが、先頭または末尾では使用できません。
- ドット . はセグメント区切り文字として使用できます。
- 大文字、アンダースコア、またはスペースは使用できません。
- 最大 63 文字。

例 `openai-gpt-4o-mini` 、`azure.gpt-4o.prod`

タグオプショナル

既存の HM タグを再利用してサービスをグループ化およびフィルタリングします。タグは要求ルーティングまたは認証に影響を与えません。

モデル名必須

プロバイダーによって文書化されている、上流プロバイダーが想定する正確な識別子。この値は大文字と小文字が区別されます。スラッシュ区切りの名前 `meta/llama-3.1-8b-instruct` や `openai/gpt-4o-mini` などがサポートされています。例については、上記の **プロバイダー別のクイックリファレンス** テーブルを参照してください。

API キーほとんどのプロバイダーに必要

API キーのみ — `Authorization: Bearer` …プレフィックスは含めないでください。 HM は、選択した **API プロトコルバージョン**に基づいて、正しい認証ヘッダーを自動的に追加します。

モデルベース URL 必須

プロバイダーの API のスキームとホストおよび必要なパス接頭辞。一般的なプロバイダーの正しい値については、上記の **プロバイダー別のクイックリファレンス** テーブルを参照してください。

`/v1` は含めないでください —コンシューマアプリケーションは`/v1` または Gemini の場合は`/v1beta` 自分自身を追加します。ここに`/v1` を含めると、404 を返す`/v1/v1/chat/completions` などのパスが重複して発生します。後続のスラッシュは許容され、自動的に削除されます。

> **Tip**
>
> セルフホスト vLLM サーバーは、モデルごとのエンドポイント `/v1/models/{name}` で `404` を返します。これにより、通常登録プローブが失敗します。 HM は、*GET /v1/models* にフォールバックし、リスト内のモデル名を照合することによりこれを処理するため、vLLM 提供モデルは正常に登録されます。

ファンクションオプショナル、複数選択

コンシューマアプリケーションが使用可能なモデルを検出するときにフィルタリングする機能タグ。HM のビルトインコンシューマには、以下の事前定義された値を使用します。独自のアプリケーションの場合、任意の文字列が有効です。

Built-in consumer	Required function tag
HM chatbot	*chatbot-gen-content*
AIDB pipeline step	The matching *aidb-** tag (see your AIDB pipeline documentation)

独自のモデル選択を実行するカスタムアプリケーションにのみサービスを公開する場合、このフィールドを空のままにします。

API プロトコルバージョン必須

要求本文の形式と送信認証ヘッダーの両方を制御します。プロバイダーのネイティブ API と一致するオプションを選択します。

Option	Request body shape	Auth header sent	Use for
OPENAI_V1	OpenAI Chat Completions	*Authorization: Bearer <key>*	OpenAI, NVIDIA NIM, vLLM, OpenRouter, any OpenAI-compatible endpoint
GEMINI_V1_BETA	Google Gemini	*x-goog-api-key: <key>*	Google Gemini native API only
ANTHROPIC_V1	Anthropic Messages	*x-api-key: <key>* + *anthropic-version: 2023-06-01*	Anthropic Claude

Allow Insecure Connection オプショナル、デフォルトオフ

上流への発信呼び出しで TLS 証明書検証を無効にします。これは、アップストリームが自己署名証明書、または HM クラスターによって信頼されていない CA によって署名された証明書を使用する場合にのみ有効にします。

別の HM の外部推論エンドポイントたとえば、HM1 を介してフェデレートする HM2 を登録するときにこのトグルを使用します。HM1 の入力は通常、自己署名証明書を提示します。これにより、登録プローブとランタイム転送の両方で TLS エラーが発生します。HM 対 HM フェデレーションでは、両端が API プロトコルバージョンとして `OPENAI_V1` を使用することも必要です。

> **警告**
>
> この設定は作成専用です。登録後に切り替えることはできません。変更する必要がある場合は、サービスを削除して再登録します。これは、開発環境または信頼できる自己署名証明書の場合にのみ有効にします。TLS 検証を無効にすると、セキュリティが低下します。

[Register] をクリックした後

HM は、インフラストラクチャを作成する前にエンドポイントを検証します。OpenAI `OPENAI_V1` 、Anthropic `ANTHROPIC_V1` 、および Google Gemini `GEMINI_V1_BETA` の場合、HM は到達可能性と資格情報の有効性の両方をチェックするライブ接続プローブを実行します。エンドポイントに到達できないか、API キーが拒否された場合、登録はエラーですぐに失敗します。リソースは作成されません。

> **注釈**
>
> NVIDIA NIM、HuggingFace、OpenRouter、認証なしで実行されるセルフホスト vLLM など、一部の `OPENAI_V1` プロバイダーの場合、モデルエンドポイントは API キーを必要としません。接続プローブは引き続き実行されますが、間違った API キーは引き続き HTTP 200 を返す場合があります。これらのプロバイダーの登録時に、キーの有効性は保証されていません。

登録後、ステータスはバックグラウンドのヘルスチェックによって 5 分ごとに更新されます。上流が到達不能になるか、資格情報を拒否し始めると、次の更新ティックでサービスは **Failed** 説明的な **ステータスメッセージ** を使用して変わります。

サービスを利用する

サービスの準備が完了すると、次のことが利用できます。

- **HM chatbot** — チャットボットは、`chatbot-gen-content` でタグ付けされたサービスを自動的に選択します。
- **Pipeline Designer** —登録された外部モデルは、HM ホストモデルとともにモデルピッカーに表示されます。詳細については、*External inference services* を参照してください。
- **Gen AI Builder** — モデルは、登録されると、Gen AI Builder パイプラインの推論ターゲットとして利用できます。

6.4.3.3.4 推論サービスの詳細の取得

Inference Services リストのサービス名をクリックして、その詳細ビューを開きます。ここには、サービスの構成、現在のステータス、および使用可能なアクションが表示されます。

詳細

Field	Description
External Service Name	登録時に割り当てられた一意の識別子。
Model name	上流プロバイダーに転送されるモデル識別子。
Model Base URL	プロキシの要求をルーティングする上流エンドポイント。
API Protocol Version	使用中の要求フォーマットと認証ヘッダー *OPENAI_V1*、*GEMINI_V1_BETA*、または `ANTHROPIC_V1` 。
Functions	現在サービスに割り当てられている機能タグ。
Allow Insecure Connection	アウトバウンド呼び出しの TLS 証明書検証を無効にするかどうか。
Status	サービスの現在の状態 **Ready** 、 **Failed** 、 **Pending** 、または **Unknown** 。
Status Message	ステータスを伴う人間が判読可能な説明。サービスが **準備ができている** 場合は空です。 **Failed** の場合、プローブ診断たとえば `API key is missing or invalid (status 401 from ...)` または `endpoint unreachable: ...` を搬送します。 **保留中** のときにキャッシュウォームアップテキストを運びます。

> **注釈**
>
> API キーはサービスの詳細ページに表示されません。これを置き換えるには、 **Quick Actions → Edit Service** を開き、新しい値を入力します。 HM は、基になるシークレットを自動的にローテーションします。

ステータスメッセージからのトラブルシューティング

サービスが **Failed** になると、 **ステータスメッセージ**が修正内容を示します。よくあるパターン

Status message	Likely cause	What to do
API key is missing or invalid	Wrong or expired API key.	Open **Edit** and replace the **API Key** value.
API key is missing/invalid or account credits are exhausted (Anthropic)	Anthropic returns the same status for a bad key and for depleted credits.	Verify the key first; if it's correct, top up the Anthropic account.
account quota exhausted — check billing and credits (OpenAI)	OpenAI billing balance is empty.	Top up the OpenAI account; no change to the HM service is needed.
API key lacks permission for this endpoint	Key is valid but doesn't have access to the model (project scoping, tier).	Grant model access on the provider's console, or use a different key.
endpoint not found — check the base URL and model name	Typo in **Model Base URL** or wrong **model name** .	Verify both values. Make sure Base URL doesn't end with */v1*.
endpoint unreachable: ... x509: ...	TLS error against an untrusted certificate.	Delete and re-register with **Allow Insecure Connection** enabled (the toggle is create-only).
endpoint unreachable: ... (other)	Network egress blocked, DNS failure, or upstream down.	Check egress rules from the HM cluster and confirm the provider is online.
rate limited by upstream	Provider is throttling requests.	Wait and retry; recurring rate limits indicate the upstream account tier needs upgrading.
upstream is temporarily unavailable / upstream server error	Provider-side outage.	Wait; the next refresh tick will recover automatically once upstream is healthy.
Remote connection check is pending — upstream has not yet been verified	Background health check hasn't run yet (typically after an HM restart).	Wait up to 5 minutes for the next refresh tick.

6.4.3.3.5 推論サービスパラメーターの更新

一部のパラメーターは登録後に更新できます。サービスの登録解除と再登録が必要な場合もあります。

変更できるものとできないもの

Field	Editable after registration?
Functions	Yes
API Key	Yes — HM rotates the underlying Kubernetes secret automatically
External Service Name	No — delete and re-register
Model name	No — delete and re-register
Model Base URL	No — delete and re-register
API Protocol Version	No — delete and re-register
Allow Insecure Connection	No — delete and re-register

注釈

API プロトコルバージョンは登録後にロックされています。各プロトコルは、異なる認証ヘッダーで異なるエンドポイントパスをプローブし、登録時に実行された接続プローブは、元のプロトコルが**モデルベース URL**および**モデル名**と一致するためのみに合格しました。これら 2 つのフィールドは変更できないため、別のプロトコルに切り替えようとすると、接続プローブは失敗し、更新は拒否されます。別のプロトコルを使用するには、サービスの登録を解除し、新しいものを登録します。

サービスパラメーターを編集する方法

サービスの詳細ページを開き、 **Quick Actions** → **Edit Service** を選択するか、 **Inference Services** リストの鉛筆アイコンをクリックします。

注釈

HM は、更新を適用する前に接続プローブを実行します。エンドポイントに到達できないか、新しい API キーが拒否された場合、更新は失敗し、変更は適用されません。

6.4.3.3.6 外部推論サービスの登録を解除する

警告

登録解除は永続的です。関連するすべての Kubernetes リソース名前空間、シークレット、*ServingRuntime* 、*InferenceService* はすぐに削除されます。このアクションは元に戻すことができません。

登録を解除する方法

サービスを削除するには、サービスの詳細ページを開き、 **Quick Actions** → **Deregister external Inference Service** を選択するか、 **Inference Services** リストのゴミ箱アイコンをクリックします。

サービスが現在 1 つ以上のパイプラインによって参照されている場合、HM は削除をブロックします。最初にこれらのパイプラインを削除または更新してから、再試行します。

削除が成功すると、HM は次のことを行います。

1. サービスをバッキングするすべての Kubernetes リソース API キーシークレットを含むを削除します。
2. データベースからサービスレコードを削除します。
3. サービスに関連付けられているすべてのタグをクリアします。

6.4.4 analytics

6.4.4.1 Langflow

Langflow は、AI パイプラインとモデルコンテキストプロトコル MCP サーバーを構築および展開するためのオープンソースツールです。エージェントやモデルコンテキストプロトコル MCP などの重要な AI 機能をサポートし、特定の大規模言語モデル LLM やベクターストアを使用する必要はありません。

EDB は、上流に 2 つの追加を含むハイブリッドマネージャーの Langflow を出荷しています。

- ** *EDB components* ** フローを HM 管理リソースナレッジベース、ベクトルインデックス、データベース、モデルサーバークラスター、ローカル `aidb` モデル、および HM プラットフォームツールに接続する **EDB** フォルダーの Langflow コンポーネントのセット。
- ** *Flow deployment* ** フローを HM でホストされる長期実行の呼び出し可能サービスに変換するための公開および展開ライフサイクル。

6.4.4.1.1 前提条件

Langflow で EDB コンポーネントを使用する前に、必要な HM ロールとアクセスキーが必要です。HM コンソールでユーザーと権限を管理できます。

ロール

- **EDB Embeddings** および **EDB Model Server** には、組織レベルでの AI Model Manager ロールが必要です。
- プロファイル > **ユーザー管理** > **組織ロールの割り当て** > **AI モデルマネージャー** 。
- **EDB Knowledge Base** 、**EDB Database** 、**EDB Vector Index** 、**EDB Embedded Models** 、および **EDB Airman MCP** は、所有するプロジェクトの Project Viewer ロールが必要です関連するクラスター
- プロジェクト > **ユーザー** > **プロジェクトロールの割り当て** > **プロジェクトビューアー** 。
- **EDB Platform** は、特定の HM ロールの割り当ては必要ありません。フローを実行している人の ID、フロービルダーのログインユーザー、または展開されたフローの呼び出し者の ID を使用します。ツール呼び出しは、その ID の権限で実行されます。

アクセスキー

- マシンユーザーを作成するか、プロファイルの [マイアカウント] オプションを介して、アクセスキーを作成します。`HM_API_KEY` という名前の Global Variable として Langflow に保存して、EDB コンポーネントが自動的に選択します。

6.4.4.1.2 次に何をするか

- *Langflow quickstart* ハイブリッドマネージャーの Langflow で最初の RAG フローをビルドします。
- *EDB components* 前提条件、フィールド、使用法を含む EDB フォルダー内の各コンポーネントのリファレンス。
- *Using models in Langflow* Langflow フローから EDB ホストモデルエンドポイントを使用する方法。
- *Flow deployment* フローのバンドルの公開、公開されたフローの展開、および展開されたフローへのアクセス。
- *Langflow templates* 一般的なハイブリッドマネージャーワークフローの事前構築されたフローテンプレート。
- *Langflow FAQ* Langflow フローの構築、構成、および操作に関するよくある質問。
- *Troubleshooting Langflow in Hybrid Manager* Langflow フローと EDB コンポーネントの一般的な問題の診断手順と修正。

Langflow quickstart

このクイックスタートでは、ハイブリッドマネージャーの Langflow で単純な取得拡張生成 RAG フローを構築する手順を説明します。最終的には、ユーザーの質問を受け入れ、ナレッジベースから関連コンテンツを取得し、LLM から根拠のある回答を返すワークフローが作成され、すべて HM クラスター内で実行されます。

前提条件

ロールとアクセスキーの設定の Langflow prerequisites を完了し、以下があることを確認します。

- モデルを提供するために展開された GPU ノード。*GPU Recommendations for Default NIM Models* を参照してください。
- 生成用 LLM を使用してモデルライブラリーから展開された少なくとも 1 つのモデルクラスター。*Model Management on Hybrid Manager* を参照してください。このクイックスタートのナレッジベースでは、データベース CPU で実行される AIDB `bert` 埋め込みモデルを使用します。埋め込みモデルクラスターは必要ありません。
- クエリするコンテンツを含むテーブルを含む HM プロジェクトの Postgres クラスター。そのクラスターのデータベースのユーザー名とパスワードを準備します。ステップ 2 で必要になります。

データの準備: ナレッジベースを作成します

Langflow を開く前に、HM Postgres クラスターにナレッジベースを作成して、フローが取得するコンテンツを持つようにします。

これは、psql またはクラスターに接続している SQL クライアントを使用して実行できます。接続の詳細を確認するには、HM コンソールでクラスターに移動し、**Connect** を選択します。

テーブル名と列を独自のデータに置き換えて、次を実行します。

```
CREATE EXTENSION IF NOT EXISTS aidb CASCADE;

SELECT aidb.create_table_knowledge_base(
    name                => my_knowledge_base,
    model_name          => bert,
    source_table        => public.my_content_table,
    source_data_column => content,
    source_data_format => Text
);

SELECT aidb.bulk_embedding(my_knowledge_base);
```

ナレッジベースの準備ができていることを確認します。

```
SELECT * FROM aidb.kbstat;
```

注釈

このクイックスタートでは、AIDB SQL を直接使用してナレッジベースを作成します。 UI 駆動のアプローチを希望する場合は、 *Pipeline Designer concepts* を介してナレッジベースを作成することもできます。

ステップ 1Langflow にアクセスする

1. HM コンソールにログインし、プロジェクトを選択します。
2. ナビゲーションで **Launchpad** を選択し、 **Langflow** を開きます。

ステップ 2 グローバル変数を設定する

EDB コンポーネントは、Langflow グローバル変数から資格情報を読み取ります。これらを一度設定すると、フロー内のすべてのコンポーネントで再利用できます。

1. Langflow で、 **Profile** → **Settings** → **Global Variables** に移動します。
2. 次の変数を作成します。

Variable name	Value
HM_API_KEY	Your HM machine user access key.
HM_DB_USER	Postgres user for your HM cluster.
HM_DB_PASSWORD	Password for the Postgres user.

手順については、 Configure Global Variables を参照してください。

ステップ 3 空のフローを作成する

1. Langflow ホーム画面で、 **New Flow** → **Blank Flow** を選択します。

ステップ 4 チャット入力を追加する

1. Langflow キャンバスのコンポーネントパネルで **Inputs** を選択します。
2. **Chat Input** をキャンバスにドラッグします。これはユーザーの質問をキャプチャします。

ステップ 5EDB Knowledge Base コンポーネントを追加する

1. コンポーネントパネルで **EDB** を選択し、 **EDB Knowledge Base** をキャンバスにドラッグします。
2. コンポーネントを構成します。

- **HM マシンユーザーキー** → `HM_API_KEY` グローバル変数を選択します。
- **Hybrid Manager Project** → プロジェクトを選択します。
- **Hybrid Manager Database** → ナレッジを含むクラスターを選択します base.
- **データベース接続タイプ** → 適切な接続タイプを選択します。
- **データベースのユーザー** → `HM_DB_USER` グローバル変数を選択します。
- **データベースのパスワード** → `HM_DB_PASSWORD` グローバル変数を選択します。
- **データベース名** → ナレッジベースが作成されたデータベースを選択します。
- **ナレッジベース** → `my_knowledge_base` 前提条件で作成を選択します。

1. **Chat 入力** → **EDB Knowledge Base** を接続します。これにより、ユーザーの質問が **Query Value** フィールドに自動的に配線されます。

ステップ 6 プロンプトテンプレートを追加する

プロンプトテンプレートがないと、モデルは質問とともに取得されたチャンクの生のダンプを受信します。テンプレートは、入力をモデルが従うことができるプロンプトに構造化します。

1. コンポーネントパネルで **プロンプト** を選択し、**プロンプトテンプレート** をキャンバスにドラッグします。
2. **テンプレート** フィールドに、次のように入力します。

```
Answer the question using only the context below. If the answer isnt in the context,␣
↪say you dont know.

Context:
{context}
```

(次のページに続く)

(前のページからの続き)

```
Question:
{question}
```

1. **EDB Knowledge Base** → **Prompt Template** `context` 変数を接続します。
2. **チャット入力** → **プロンプトテンプレート** `question` 変数を接続します。

ステップ 7EDB Model Server コンポーネントを追加する

1. コンポーネントパネルで **EDB** を選択し、 **EDB Model Server** をキャンバスにドラッグします。
2. コンポーネントを構成します。

- **HM マシンユーザーキー** → `HM_API_KEY` グローバル変数を選択します。
- **Hybrid Manager Model Server Cluster Instance** → 前提条件に展開された LLM モデルクラスターを選択します。

1. **プロンプトテンプレート** → **EDB Model Server** を接続します。

ステップ 8 チャット出力を追加する

1. コンポーネントパネルで **出力** を選択し、 **チャット出力**をキャンバスにドラッグします。
2. **EDB Model Server** → **Chat Output** を接続します。

ステップ 9 フローの実行とテスト

1. 上部のツールバーで **Playground** を選択して、インタラクティブテストパネルを開きます。
2. ソースデータに関連する質問を入力し、 **Enter** を押します。
3. 応答がコンテンツに基づいていることを確認します。関連する行を取得する EDB ナレッジベース、コンテキストと質問をフォーマットするプロンプトテンプレート、および状況に応じた回答を生成する EDB モデルサーバーが表示されるはずです。
4. 実行後、すべてのコンポーネントが緑色のステータスインジケータを表示することを確認します。

エラーを表示するコンポーネントがある場合は、それを選択して構成を確認します。よくある問題

- **EDB ナレッジベースコンポーネントではプロジェクトを選択できません** - ポータルドメインはデフォルトで API エンドポイントになっています。コンポーネントで **Controls** を選択し、 **Hybrid Manager URL** を `http://istio-internalgateway.istio-system.svc.cluster.local:7080` に設定します。
- **不正または有効期限が切れたアクセスキー** — `HM_API_KEY` グローバル変数が有効なマシンユーザーキーを保持していることを確認します。

- **不一致のデータベース接続** —データベースユーザー、パスワード、およびデータベース名がクラスターで構成されているものと一致することを確認します。
- **モデルクラスターがアクティブではありません** — **Agent Factory** → **Model Server Clusters** でクラスターステータスを確認します。

詳細なヘルプについては、 *Troubleshooting Langflow in Hybrid Manager* を参照してください。

次のステップ

- **コンポーネントを追加する** —ナレッジベースとともに構造化データを照会する **EDB Database** コンポーネント、またはベクトルインデックスを管理するための **EDB Vector Index** コンポーネントを使用してフローを拡張します。 EDB Airman MCP、EDB Platform、EDB Embedded Models を含むフルセットについては、*EDB components* を参照してください。
- **フローの展開** —バンドルとして公開し、独自のエンドポイント、プロジェクトレベルの分離、および独立したスケーリングを備えた管理対象呼び出し可能サービスとして展開します。*Flow deployment* を参照してください。
- **フローの依存関係を監視する** —モデルクラスターと Langflow ポッドの Prometheus アラートルールと Grafana ダッシュボードを設定します。*Observability (Agent Factory on HM)* を参照してください。

EDB components

Langflow の **EDB** フォルダーには、ハイブリッドマネージャー HM によって管理されるリソースにフローを接続するコンポーネントが含まれています。ナレッジベース、データベース、ベクトルインデックス、埋め込みおよびモデルサーバークラスター、埋め込みモデル、MCP ツール。

Component	Connects flow to	Required roles
EDB Knowledge Base	An AIDB knowledge base in an HM cluster	Project Viewer
EDB Vector Index	Vector indexes on Postgres tables (pgvector)	Project Viewer
EDB Database	A Postgres database in an HM cluster	Project Viewer
EDB Airman MCP	The Airman MCP server's tools	None (requires valid PG connection string)
EDB Platform	HM platform MCP tools for the current user	Depends on operations invoked
EDB Embeddings	An HM-hosted embedding model	AI Model Manager
EDB Model Server	An HM-hosted inference model (LLM or other)	AI Model Manager
EDB Embedded Models	AIDB embedded models for CPU-based inference	Project Viewer

ロール要件とグローバル変数の設定については、親の *Langflow FAQ* ページを参照してください。

EDB Knowledge Base

EDB Knowledge Base コンポーネントは、ナレッジベースのソーステーブル列にオプションの構造化フィルタリングを使用して、ハイブリッドマネージャー HM クラスター内の既存の AIDB knowledge base に対してセマンティック類似性検索を実行します。

適切なコンポーネントの選択

EDB Knowledge Base コンポーネントは、次の場合に使用します。

- 生成ステップに渡す前に、関連するパッセージを類似性によって取得する RAG フローを構築しています。
- オンデマンドでナレッジベースの情報を検索するエージェントツールが必要です。コンポーネントはツールモード互換性があります。
- ハイブリッド検索が必要です。ナレッジベースのソーステーブル列の構造化フィルタとセマンティックランキングを組み合わせます。

次の場合に別のコンポーネントを使用します。

- 既存のベクター列で pgvector インデックス構造を管理する必要があります HNSW または IVFFlat インデックスの作成、ドロップ、リビルド。*EDB Vector Index* を使用します。
- 類似性検索ではなく、データベースへの直接 SQL アクセスが必要です。*EDB Database* を使用します。
- ナレッジベースはまだ存在しません。*Pipeline Designer concepts* を介して、またはクラスターに対して `aidb.create_table_knowledge_base()` を呼び出して、作成します。

前提条件

- クラスターを所有するプロジェクトの Project Viewer ロール
- プロジェクト > **ユーザー** > **プロジェクトロールの割り当て** > **プロジェクトビューアー** 。
- 少なくとも 1 つの AIDB ナレッジベースを含む HM クラスター。*Pipeline Designer concepts* を介してまたは直接 `aidb.create_table_knowledge_base()` を介して作成します。ナレッジベースのソースは、テーブルフィルタリングサポートされている場合またはボリュームセマンティックのみである場合があります。
- HM マシンユーザーアクセスキーと Langflow に Global Variables として保存したデータベースユーザー/パスワード。

入力

接続

Field	Type	Required	Default	Notes
Connection Mode	Dropdown	Yes	Hybrid Manager	*Hybrid Manager*`または`*Direct Connection*。後続の接続フィールドを切り替えます。
Hybrid Manager URL	Text	No		高度な。デフォルトのHM URLをオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Project	Dropdown	Yes		HMプロジェクトから設定されます。
Hybrid Manager Database	Dropdown	Yes		ナレッジベースを含むクラスター。
Database Group	Dropdown	No		クラスターグループたとえば、プライマリまたはリーダー。
Database Connection Type	Dropdown	Yes		たとえば、読み取りまたは読み取り/書き込み。
User for the database	Text	Yes	*HM_DB_USER*	Postgres のロール。`HM_DB_USER`グローバル変数のデフォルト。
Password for the database	Secret	Yes	*HM_DB_PASSWO*	`HM_DB_PASSWORD`グローバル変数のデフォルト。
Default database name override	Text	No		高度な。データベースをリストするときにクラスターのデフォルトをオーバーライドします。
Database Name	Dropdown	Yes		AIDBナレッジベースを保持するデータベース。

ダイレクト接続モードでは、HMクラスターフィールドは**データベースホスト**および**データベースポート**に置き換えられます。ユーザー、パスワード、および名前のフィールドは残ります。

ナレッジベースとクエリ

Field	Type	Required	Default	Notes
Knowledge Base	Dropdown	Yes		選択したデータベースの `aidb.knowledge_bases` から入力されます。作成したばかりの場合は、更新ボタンを使用します。
Limit results	Integer	Yes	*10*	返される最大行数。
Query Value	Multiline text	Yes		検索する自然言語クエリー。ツールモード互換性 エージェントは、これをツール引数として使用してコンポーネントを呼び出します。

検索モード

Field	Type	Required	Default	Notes
Search Mode	Dropdown	Yes	*Vector Only*	*Vector Only* `は、` `aidb.retrieve_*` `を介して純粋なセマンティック検索を実行します。` *Vector + Filter* `は、ナレッジベースのソーステーブルに対して` *WHERE* `句を追加します。 KB のソースがボリュームの場合、` *Vector Only* `のみが提供されます。 :ref:` `ハイブリッド検索<ハイブリッド検索>` を参照してください。

Filter Vector + Filter のみ

これらのフィールドは、 **検索モード** が **ベクター+フィルター**であり、ナレッジベースにテーブルソースがある場合にのみ表示されます。

Field	Type	Required	Default	Notes
Filter Column	Dropdown	Yes		ソーステーブルの列から設定されます。
Operator	Dropdown	Yes	=	フィルタ演算子 を参照してください。

次のページに続く

表 14 – 前のページからの続き

Field	Type	Required	Default	Notes
Filter Value	Text	Yes		比較する値。 *IN* `の場合、コンマ区切りリストを指定します。 `*IS NULL* / `IS NOT NULL`では無視されます。

出力オプション

Field	Type	Required	Default	Notes
Include Text	Boolean	No	*false*	**Vector Only** モードで適用されます。オンの場合、結果には各ヒットのソーステキスト列が含まれます *aidb.retrieve_text* `を使用します。オフの場合、キーと距離のみが返されます `*aidb.retrieve_key* 。**Vector + Filter** の下では効果がなく、常にソーステキスト列が返されます。
Include Columns	Boolean	No	*true*	高度な。出力にソーステーブルの列のメタデータを含めます。ツールモード互換性。
Add Error	Boolean	No	*false*	高度な。結果のクエリエラーを発生させる代わりに、表面化します。ツールモード互換性。

出力

Output	Type	Carries
Result Table	DataFrame	One row per match, with *key*, optional *value* (text), and *distance*. Distance is computed using the knowledge base's configured metric (Cosine, L2, or Inner Product).

ハイブリッド検索

Vector + Filter は、ナレッジベースのソーステーブルに対して構造化された `WHERE` 句とセマンティックの類似性を組み合わせます。トップ K のセマンティックがデータのサブセット**内** と一致する場合に使用します。例、「`kind = 'policy'` タグ付けの文書で最も近い一致」。

仕組み

- コンポーネントは、KB のベクトルテーブルとソースキー列のソーステーブルを JOIN します。
- フィルタは、order-by-distant の前の `WHERE` 句で適用されます。
- HNSW インデックス付きベクトルの場合、コンポーネントは `hnsw.iterative_scan = relaxed_order` を設定するため、インデックスが積極的にプルーニングされすぎる場合でも、フィルタリングされた検索は有用な結果を結果ます。

ハイブリッド検索は、ナレッジベースのソースがテーブルの場合にのみ使用できます。ソースがボリュームの場合、**検索モード** ドロップダウンは **ベクターのみ**に制限され、フィルタフィールドは非表示のままです。

フィルタ演算子

Operator	Behavior
=, !=, >, <, >=, <=	標準 SQL の比較。数値は引用符なしで渡されます。数値以外の値は引用符で囲まれます。
LIKE	SQL `LIKE`パターンマッチ。 **Filter Value** で`%`ワイルドカードを使用します。
IN	**Filter Value** のコンマ区切り値、例 *policy, contract, memo*。
IS NULL, IS NOT NULL	**フィルタ値** は無視されます。

EDB Vector Index

EDB Vector Index コンポーネントは、既存のテーブルの pgvector インデックスを管理します。新しいインデックスの作成、既存のインデックスの削除、再構築、または単に存在するかどうかを確認できます。

適切なコンポーネントの選択

EDB Vector Index コンポーネントは、次の場合に使用します。

- `vector` 列を含む Postgres テーブルが既にあり、インデックスを追加または変更して、類似性検索を高速化したいと考えています。
- インデックスタイプ HNSW 対 IVFFlat を切り替えるか、既存の列の距離メトリックを変更したい。
- 他の操作を実行する前に、特定のインデックスが配置されているかどうかをフローから確認したい。

次の場合に別のコンポーネントを使用します。

- インデックス構造を管理するのではなく、ナレッジベースを照会したい。*EDB Knowledge Base* を使用します。
- データベースへの生の SQL アクセスが必要です。*EDB Database* を使用します。
- テーブルには `vector` 列がまだありません。コンポーネントは、既存のベクター列のインデックスのみを管理します。最初に列を追加します `aidb` または独自の DDL を使用して

前提条件

- ターゲットテーブルに対する `CREATE INDEX` 特権を持つ Postgres ロール。
- `pgvector` 拡張機能がインストールされた HM クラスターと、少なくとも 1 つの `vector` 列を含むテーブル。
- HM マシンユーザーアクセスキーと Langflow に Global Variables として保存されたデータベースユーザー/パスワード。

入力

接続

Field	Type	Required	Default	Notes
Hybrid Manager URL	Text	No		高度な。デフォルトの HM URL をオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Project	Dropdown	Yes		HM プロジェクトから設定されます。
Hybrid Manager Database	Dropdown	Yes		ターゲットテーブルを含むクラスター。
Database Group	Dropdown	No		クラスターグループたとえば、プライマリまたはリーダー。
Database Connection Type	Dropdown	Yes		たとえば、読み取りまたは読み取り/書き込み。
User for the database	Text	Yes	*HM_DB_USER*	`HM_DB_USER`グローバル変数のデフォルト。
Password for the database	Secret	Yes	*HM_DB_PASSWO*	`HM_DB_PASSWORD`グローバル変数のデフォルト。
Default database name override	Text	No		高度な。データベースをリストするときにクラスターのデフォルトをオーバーライドします。

次のページに続く

表 18 – 前のページからの続き

Field	Type	Required	Default	Notes
Database Name	Dropdown	Yes		ベクター列を含むテーブルを含むデータベース。

ターゲット

Field	Type	Required	Default	Notes
Table	Dropdown	Yes		選択したデータベースのテーブル、システムスキーマを除きます。
Vector Column	Dropdown	Yes		選択したテーブルの`vector`タイプの列から入力されます。
Distance Operation	Dropdown	Yes	*Cosine*	*Cosine*、*L2*、*InnerProduct*`のいずれか。*pgvector* **オペレータークラス** `*vector_cosine_ops*、*vector_l2_ops*、`vector_ip_ops`にマップします。
Index Type	Dropdown	Yes	*hnsw*	*hnsw*`または`*ivfflat*。
Operation	Dropdown	Yes	*Create*	*Create*、*Drop*、*Rebuild*、*Check Exists*`のいずれか。*:ref:*`オペレーション<オペレーション>` を参照してください。

HNSW パラメーターインデックスタイプ= hnsw

Field	Type	Required	Default	Notes
HNSW M (Max Connections)	Integer	No	*16*	高度な。ノードごとの最大接続数。
HNSW ef_construction	Integer	No	*64*	高度な。インデックス構築中の動的候補リストのサイズ。

IVFFlat パラメーターインデックスタイプ = ivf flat

Field	Type	Required	Default	Notes
IVFFlat Lists	Integer	No	*100*	高度な。逆引きリストの数。

出力

Output	Type	Carries
Result	Data	Operation outcome. Always includes *success*, *operation*, *table*, *column*, *index_name*, and *message*. *Check Exists* adds *exists*. *Create* adds *index_type* and the generated *sql* when it builds a new index, and *already_existed* when an index was already present (set both on an idempotent same-type match and on a different-type conflict).

オペレーション

Operation	Behavior
Create	インデックスが存在しない場合は作成します。べき等 同じタイプのインデックスがこの列に既に存在する場合、操作は再作成せずに成功を結果ます。列に別のタイプのインデックスが存在する場合、操作は失敗し、最初に`Drop`するように求められます。
Drop	インデックスを削除します。冪等`DROP INDEX IF EXISTS`を使用します。
Rebuild	既存のインデックスで`REINDEX INDEX`を実行します。インデックスが存在しない場合は失敗します。
Check Exists	`exists: true/false`を返します。データベースは変更しません。

インデックスの命名

コンポーネントは、テーブルと列から安定した予測可能なインデックス名 `idx_<table>_<column>_<index_type>` スキーマ接頭辞なしを生成します。その命名規則が必要な場合は、このコンポーネントを介してインデックスを選択します。それ以外の場合は、SQL で管理します。

EDB Database

EDB Database コンポーネントは、HM プロジェクト、クラスター、および接続タイプを Postgres 接続 URL に解決し、その URL を出力として公開します。ダウンストリームコンポーネント SQL クエリー、MCP ツールラッパー、カスタムコードは、URL を使用して接続します。

適切なコンポーネントの選択

次の場合に EDB Database コンポーネントを使用します。

- HM クラスター内のデータベースの Postgres 接続 URL が必要です。HM プロジェクトと接続設定から派生します。
- その URL を、データベースに直接接続する別のコンポーネント SQL-as-tools の *EDB Airman MCP* またはカスタムフローコンポーネントに接続する必要があります。

次の場合に別のコンポーネントを使用します。

- 既存のナレッジベースに対する類似性検索が必要です。*EDB Knowledge Base* を使用します。
- pgvector インデックス構造を管理したい。 *EDB Vector Index* を使用します。
- エージェントの MCP ツールとして SQL が必要です。*EDB Airman MCP* を使用しますこのコンポーネントからの URL 出力を受け入れます。

前提条件

- クラスターを所有するプロジェクトの Project Viewer ロール
- プロジェクト > **ユーザー** > **プロジェクトロールの割り当て** > **プロジェクトビューアー** 。
- ユーザーがアクセスできる HM クラスター。*Create clusters* を参照してください。
- HM マシンユーザーアクセスキーと Langflow に Global Variables として保存したデータベースユーザー/パスワード。

入力

Field	Type	Required	Default	Notes
Hybrid Manager URL	Text	No		高度な。デフォルトの HM URL をオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Project	Dropdown	Yes		HM プロジェクトから設定されます。
Hybrid Manager Database	Dropdown	Yes		データベースを含むクラスター。
Database Group	Dropdown	No		クラスターグループたとえば、プライマリまたはリーダー。
Database Connection Type	Dropdown	Yes		たとえば、読み取りまたは読み取り/書き込み。
User for the database	Text	Yes	*HM_DB_USER*	`HM_DB_USER`グローバル変数のデフォルト。

次のページに続く

表 24 – 前のページからの続き

Field	Type	Required	Default	Notes
Password for the database	Secret	Yes	*HM_DB_PASSWO*	`HM_DB_PASSWORD`グローバル変数のデフォルト。
Default database name override	Text	No		高度な。データベースをリストするときにクラスターのデフォルトをオーバーライドします。
Database Name	Dropdown	No		接続するデータベース。ユーザーのデフォルトを使用する場合は、空白のままにします。

注釈

ポータルドメインがデフォルトで API であるためプロジェクトを選択できない場合は、EDB データベースとナレッジベースコンポーネントの上部にある**Controls**を選択して、*http://istio-internalgateway.istio-system.svc.cluster.local:7080* を**ハイブリッドマネージャー URL**として構成します。

Output	Type	Carries
Database URL	Message	The fully resolved Postgres connection string (*postgresql://user:password@host:port/database*), URL-encoded as needed. Pass this into components that take a database URL, such as *EDB Airman MCP* .

EDB Airman MCP

EDB Airman MCP コンポーネントは、

pg-airman-mcp によって支援される Postgres MCP サーバーを起動し、そのツールをエージェントに表示します。エージェントは、これらのツールを呼び出して、個々の SQL 操作を手動で接続せずに、データベースを読み取るか、無制限モードで書き込むことができます。

適切なコンポーネントの選択

EDB Airman MCP コンポーネントは、次の場合に使用します。

- フローの実行中にエージェントがデータベースを照会できるようにする必要があります行の検索、テーブルの概要、クエリによる質問に回答します。

- デフォルトで安全な動作が必要な場合。無制限のアクセスを明示的にオプトインしない限り、エージェントは読み取り専用クエリのみを実行します。
- 各フロー実行は、監査のために呼び出した HM セッションにトラックバックしたい。

次の場合に別のコンポーネントを使用します。

- 他のコンポーネントに渡すには、固定データベース接続が必要ですエージェント用のツールではありません。*EDB Database* を使用します。
- SQL ではなくエージェントツールとしてプラットフォームレベルの操作プロジェクト、クラスター、ナレッジベースが必要です。*EDB Platform* を使用します。
- 任意の SQL ではなく、ナレッジベースに対する類似性検索が必要です。*EDB Knowledge Base* を使用します。

前提条件

- データベースクラスターを所有するプロジェクトの Project Viewer ロール。
- 到達可能な Postgres データベース URL *EDB Database* コンポーネントの `Database URL` 出力から、または直接提供されます。
- **ツール**を消費するフロー内のエージェントコンポーネントの唯一の出力はツールです。

入力

Field	Type	Required	Default	Notes
Database URL	Text	Yes		Postgres 接続文字列。通常、 *EDB Database* コンポーネントの `Database URL`出力から配線されます。
Purpose	Text	No		`AIR-MAN_MCP_PURPOSE`として MCP サーバーに渡されるオプションのフリーテキストヒント。これを使用して、エージェントがデータベースで行うべきことを説明します。

次のページに続く

表 26 – 前のページからの続き

Field	Type	Required	Default	Notes
Unrestricted Access Mode	Boolean	No	*false*	高度な。オフデフォルト場合、MCP サーバーは制限モードで実行され、安全な読み取り専用クエリのみが許可されます。オンの場合、エージェントはデータベースロールで許可されている SQL を実行できます。
MCP Server Name	Text	No		高度な。登録された MCP サーバー名をオーバーライドします。デフォルトでは、名前はフローノード ID から取得されるため、ビルド間で安定しています。

出力

Output	Type	Carries
Toolset	Tool	The MCP tools exposed by *pg-airman-mcp* for the configured database. Connect to an agent's Tools input.

アクセスモード

無制限のアクセスモードトグルは、エージェントができることを制御します。

Mode	Behavior
Restricted (default)	MCP サーバーは、安全な読み取り専用クエリーのみを受け入れます。変更ステートメント *INSERT*、*UPDATE*、*DELETE*、 DDL は、データベースロールで許可されている内容に関係なく、MCP レイヤーで拒否されます。
Unrestricted	MCP サーバーは、接続されたデータベースロールが実行できる SQL を受け入れます。ロール自分自身の特権が唯一の上限になります。

制限付きモードが安全なデフォルトです。エージェントに書き込みを行う正当な理由がある場合にのみオフにして、必要な最小限の特権を持つデータベースロールと組み合わせます。

セッションの伝播

コンポーネントが HM 展開されたフロー内で実行されると、呼び出したユーザーの HM セッション ID が各ツール呼び出しで MCP サーバーに転送されます。ツール呼び出しは、元のセッションまでトレース可能であり、監査に役立ちます、MCP アクティビティと HM ゲートウェイログを関連付けます。

EDB Platform

EDB Platform コンポーネントは、フローをハイブリッドマネージャー HM プラットフォームの MCP ツールサーフェスに接続します。呼び出したユーザーに代わって HM のプラットフォームレベルの操作をエージェントに表示するため、エージェントはユーザーの HM リソースをリスト、検査、および行動できます。

適切なコンポーネントの選択

EDB Platform コンポーネントは、次の場合に使用します。

- フローの実行中にエージェントが発信者の HM 環境プロジェクト、クラスター、ナレッジベースに関する情報を検索できるようにします。
- 現在の HM ユーザーの権限に範囲を限定した MCP ツールとして利用可能なプラットフォーム操作が必要です。

次の場合に別のコンポーネントを使用します。

- エージェントに特定の Postgres データベースに対して SQL を実行させたい。*EDB Airman MCP* を使用します。
- プラットフォームナビゲーションではなく、ナレッジベースの類似性検索が必要です。*EDB Knowledge Base* を使用します。

前提条件

- フローを実行する ID には、エージェントが呼び出すプラットフォーム操作に対する HM 権限が必要です。フロービルダーでは、これはログインしたユーザーです。展開されたフローでは、それは呼び出し元の ID ユーザーまたはマシンユーザーです。ツール呼び出しは、その ID の権限で実行されます。

入力

Field	Type	Required	Default	Notes
Tool	Dropdown	Yes		更新ボタンを選択すると、HM プラットフォーム MCP サーバーから設定されます。呼び出す特定のツールを選択します。ツールの引数に応じて、追加フィールドが表示されます。

ツールを選択すると、コンポーネントは MCP プロキシからツールの入力スキーマを読み取り、ドロップダウンの下の引数ごとに 1 つのフィールドをレンダリングします。フィールドタイプとラベルはツール定義から取得されます。

出力

Output	Type	Carries
Response	DataFrame	The selected tool's result, parsed as a DataFrame. JSON outputs are flattened to columns. Errors are returned as a row with an *error* column rather than raised.

コンポーネントがエージェントのツールプロバイダーとして使用される場合、直接呼び出されるのではなく、プロキシが公開するすべてのツールはエージェントに登録され、**Tool** ドロップダウンは参照されません。

注

- ツールは、実行中の HM 側 MCP プロキシから取得されます。使用可能なツールとその引数スキーマは、HM プラットフォームの機能が進化するにつれて変更される可能性があります。
- 各ツール呼び出しは、呼び出したユーザーの HM ID とセッションを伝えるため、プラットフォームサイドの監査ログはアクションを共有サービスアカウントではなくユーザーに帰します。

EDB Embeddings

EDB Embeddings コンポーネントは、ハイブリッドマネージャー HM モデルサーバークラスターで実行されているエンベディングモデルにフローを接続し、ダウンストリームコンポーネントベクターストア、リトリーバーがテキストをエンコードするために使用できる Embeddings モデルオブジェクトを生成します。

適切なコンポーネントの選択

次の場合に EDB Embeddings コンポーネントを使用します。

- HM ホストモデルサーバークラスター OpenAI 互換または NVIDIA NIM で実行されているモデルによって生成されたベクトル埋め込みが必要です。
- エンベディングモデルを想定するベクターストア、レトリーバー、または類似性検索コンポーネントをフィードしている。

次の場合に別のコンポーネントを使用します。

- `aidb` を使用して、データベースクラスターでローカルに CPU ベースの埋め込みを生成したい。*EDB Embedded Models* を使用します。
- エンベディングではなく、モデルからの生成応答が必要です。*EDB Model Server* を使用します。

前提条件

- 埋め込みモデルが展開された HM モデルサーバークラスター。作成方法については、*Model Management on Hybrid Manager* を参照してください。通常、 *GPU Recommendations for Default NIM Models* が必要です。
- Langflow に Global Variable デフォルト名 `HM_API_KEY` として保存された HM マシンユーザーアクセスキー。

入力

接続

Field	Type	Required	Default	Notes
Hybrid Manager URL	Text	No		高度な。デフォルトの HM URL をオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Model Server Cluster Instance	Dropdown	Yes		呼び出すモデルサーバークラスター。 HM モデルクラスターから設定されます。
External Ingress	Boolean	No		高度な。クラスター内サービスではなく外部イングレスを介してルーティングします。

モデル

Field	Type	Required	Default	Notes
API Client	Dropdown	No	*OpenAI*	高度な。 EDB Model Server の OpenAI 互換 API の `OpenAI`。 *NVIDIA* NVIDIA 固有の埋め込み機能を必要とするネイティブ NIM エンドポイントに接続する場合にのみ。

次のページに続く

表 32 – 前のページからの続き

Field	Type	Required	Default	Notes
Model	Dropdown	Yes		選択したモデルサーバークラスターから設定されます。リストが空の場合は、更新ボタンを使用します。

API Client = OpenAI デフォルト

Field	Type	Required	Default	Notes
Dimensions	Integer	No		高度な。出力ベクトルの次元数。特定のモデルでのみサポートされています。モデルのデフォルトを使用する場合は、空白のままにします。
Max Retries	Integer	No	*3*	高度な。失敗した埋め込み呼び出しの最大再試行数。
Chunk Size	Integer	No	*1000*	高度な。 1 つのバッチで送信するテキストの最大数。
Model Kwargs	Dict	No		高度な。 OpenAI 埋め込みクライアントに渡される追加のキーワード引数。

API クライアント=NVIDIA

Field	Type	Required	Default	Notes
Model Temperature	Float	No	*0.1*	高度な。モデル温度。

OpenAI 固有のフィールドは、 **API Client** が `NVIDIA` の場合、およびその逆の場合に非表示になります。クライアントを切り替えてもフィールド値は保持されるため、ラウンドトリップによって設定がワイプされません。

出力

Output	Type	Carries
Embedding Model	Embeddings	A LangChain *Embeddings* instance pointing at the selected HM model server cluster. Pass this into a vector store or retriever.

EDB Model Server

EDB Model Server コンポーネントは、ハイブリッドマネージャー HM モデルサーバークラスターで実行されている推論モデルにフローを接続します。チャットスタイルの言語モデル、推論モデル、および HM に展開したその他の推論エンドポイントに使用します。

適切なコンポーネントの選択

EDB Model Server コンポーネントは、次の場合に使用します。

- HM ホストモデルサーバークラスターで実行されているモデルからの言語モデル応答が必要です。
- `LanguageModel` オブジェクトをエージェントまたはダウンストリームチェーンにプラグインする必要があります。
- OpenAI 互換エンドポイントデフォルトまたはネイティブ NVIDIA NIM エンドポイントを使用している。

次の場合に別のコンポーネントを使用します。

- テキスト応答ではなく埋め込みが必要です。 *EDB Embeddings* を使用します。
- CPU ベースの推論をモデルサーバーではなくデータベースクラスターで実行したい。*EDB Embedded Models* を使用します。

前提条件

- ターゲットモデルが展開された HM モデルサーバークラスター。作成方法については、*Model Management on Hybrid Manager* を参照してください。通常、 *GPU Recommendations for Default NIM Models* が必要です。
- Langflow に Global Variable デフォルト名 `HM_API_KEY` として保存された HM マシンユーザーアクセスキー。

入力

プロンプト

Field	Type	Required	Default	Notes
Input	Message	No		モデルに送信するユーザーメッセージ。上流の Chat 入力または他のテキスト生成コンポーネントから接続します。
System Message	Multiline text	No		モデルに渡されるシステムプロンプト。
Stream	Boolean	No	*false*	高度な。モデルの応答トークンが到着したら、ストリーミングします。ストリーミングエンドポイントを公開するモデルでのみサポートされています。

接続

Field	Type	Required	Default	Notes
Hybrid Manager URL	Text	No		高度な。デフォルトの HM URL をオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Model Server Cluster Instance	Dropdown	Yes		呼び出すモデルサーバークラスター。 HM モデルクラスターから設定されます。
External Ingress	Boolean	No		高度な。クラスター内サービスではなく外部イングレスを介してルーティングします。

モデル

Field	Type	Required	Default	Notes
API Client	Dropdown	No	*OpenAI*	高度な。 `OpenAI`は、エージェントフローでの正しいツール呼び出し動作に必要です。`NVIDIA`は、詳細な思考などの NVIDIA 固有の機能を必要とするネイティブ NIM エンドポイントに接続する場合にのみ。
Model Name	Dropdown	Yes		選択したモデルサーバークラスターから設定されます。リストが空の場合は、更新ボタンを使用します。
Temperature	Slider	No	*0.1*	ランダム性を制御します。レンジ `0–1`、ステップ `0.01`。
Max Tokens	Integer	No		高度な。生成する最大トークン。レンジ `0–128000`。無制限の場合は `0`に設定します。
Seed	Integer	No	*1*	高度な。再現性をコントロールします。
Default model query	Text	No		高度な。入力がコンポーネントに接続されていないときに送信されます。空の入力を受け入れないモデルをテストする場合に役立ちます。

API Client = OpenAI デフォルト

Field	Type	Required	Default	Notes
Model Kwargs	Dict	No		高度な。 OpenAI チャットクライアントに渡される追加のキーワード引数。
JSON Mode	Boolean	No	*false*	高度な。モデルが JSON を返すように強制します。
Max Retries	Integer	No	*5*	高度な。失敗した要求の最大再試行数。

次のページに続く

表 39 – 前のページからの続き

Field	Type	Required	Default	Notes
Timeout	Integer	No	*700*	高度な。要求ごとのタイムアウト秒単位。

API クライアント=NVIDIA

Field	Type	Required	Default	Notes
Detailed Thinking	Boolean	No	*false*	高度な。モデルの詳細な思考プロセスを結果ます。NVIDIA 推論モデルでのみサポートされています。

OpenAI 固有のフィールドは、**API Client** が `NVIDIA` の場合、およびその逆の場合に非表示になります。クライアントを切り替えても、フィールド値は保持されます。

出力

Output	Type	Carries
Model Response	Message	The model's text response to the input prompt.
Language Model	LanguageModel	A LangChain *Language-Model* instance pointing at the selected HM model server cluster. Pass this into an agent or chain that needs a model handle.

EDB Embedded Models

EDB Embedded Models コンポーネントは、AIDB 拡張機能のビルトインモデルを使用して、Postgres クラスター内で推論を実行します。外部の推論サービスは関係しません。埋め込みと小型モデルのテキスト生成はデータベース CPU で実行され、ダウンストリームの Postgres 操作に必要な場所に既にあるデータを生成します。

適切なコンポーネントの選択

次の場合に EDB Embedded Models コンポーネントを使用します。

- 外部ネットワーク呼び出しを使用せずに、クラスターでローカルに生成されるテキストまたは画像の埋め込み。
- 大規模なフロンティアモデルを必要としないタスクの場合、データの近くにある小規模モデルテキスト生成 T5、Llama-Instruct、Qwen2、SmolLM2 が必要です。

- 単一の埋め込みステップと同じクラスター内のベクターストレージを組み合わせたフローを構築しています。

次の場合に別のコンポーネントを使用します。

- ハイブリッドマネージャーホスト型の大規模モデル推論エンドポイントが必要です。*EDB Model Server* または *EDB Embeddings* を使用します。
- 新しいエンベディングを計算するのではなく、既存のナレッジベースから取得したい。*EDB Knowledge Base* を使用します。

前提条件

- aidb 拡張機能がインストールされた HM クラスター。
- 少なくとも 1 つのサポートされている組み込みモデルが `aidb.list_models()` に登録されています。サポートされているプロバイダー `bert_local` 、`clip_local` 、`t5_local` 、`llama_instruct_local` 、`qwen2_local` 、`smollm2_local` 。
- HM マシンユーザーアクセスキーと Langflow に Global Variables として保存したデータベースユーザー/パスワード。

入力

接続

Field	Type	Required	Default	Notes
Connection Mode	Dropdown	Yes	Hybrid Manager	*Hybrid Manager*`または`*Direct Connection*。後続の接続フィールドを切り替えます。
Hybrid Manager URL	Text	No		高度な。デフォルトの HM URL をオーバーライドします。
HM Machine User Key	Secret	Yes	*HM_API_KEY*	`HM_API_KEY`という名前のグローバル変数のデフォルト。
Hybrid Manager Project	Dropdown	Yes		HM プロジェクトから設定されます。
Hybrid Manager Database	Dropdown	Yes		`aidb`を実行しているクラスター。
Database Group	Dropdown	No		クラスターグループたとえば、プライマリまたはリーダー。
Database Connection Type	Dropdown	Yes		たとえば、読み取りまたは読み取り/書き込み。
User for the database	Text	Yes	*HM_DB_USER*	`HM_DB_USER`グローバル変数のデフォルト。

次のページに続く

表 42 – 前のページからの続き

Field	Type	Required	Default	Notes
Password for the database	Secret	Yes	*HM_DB_PASSWO*	`HM_DB_PASSWORD`グローバル変数のデフォルト。
Default database name override	Text	No		高度な。データベースをリストするときにクラスターのデフォルトをオーバーライドします。
Database Name	Dropdown	Yes		`aidb`がインストールされたデータベース。

ダイレクト接続 モードでは、HM クラスターフィールドは**データベースホスト** および **データベースポート**に置き換えられます。

モデル選択

Field	Type	Required	Default	Notes
Model	Dropdown	Yes		選択したデータベースに対して`aidb.list_models()`から入力されます。モデルを登録したばかりの場合は、更新ボタンを使用します。
Model Capability	Dropdown	Yes		使用可能な機能は、選択したモデルによって異なります。 機能 を参照してください。

機能固有の入力

機能ドロップダウンに続くフィールドは、 **モデル機能**の設定によって異なります。

モデル機能 が **テキスト埋め込み** の場合

Field	Type	Required	Default	Notes
Text Input	Message/Data	No		埋め込むテキスト。空の場合、コンポーネントはエンベディングモデル自分自身を出力しますベクターストアに接続するのに役立ちます、出力 を参照してください。

モデル機能 が **テキスト生成** の場合

Field	Type	Required	Default	Notes
Prompt	Message	Yes		モデルに送信するプロンプト。
Temperature	Slider	No	*0.1*	高度な。ランダム性を制御します。レンジ`0–1`。
Max Tokens	Integer	No		高度な。生成する最大トークン。レンジ`1–4096`。

モデル機能 が **画像埋め込み** の場合

Field	Type	Required	Default	Notes
Image	Data/Message	Yes		埋め込む画像。ファイルコンポーネントまたは上流の画像生成コンポーネントを接続します。

出力

出力は機能によって異なります。

Capability	Output	Type	Carries
Text Embedding (with text supplied)	**Embedding**	Data	*{input, embedding, model, embedding_dim}* for the supplied text.
Text Embedding (text empty)	**Embedding**	Data	The embeddings model itself, suitable for passing to a vector store component.
Image Embedding	**Embedding**	Data	*{input, embedding, model, embedding_dim}* for the supplied image.
Text Generation	**Response**	Message	The model's generated text.

機能

Model Capability ドロップダウンには、選択したモデルの機能のみがリストされます。ビルトインモデルのマッピング

Model	Capabilities
bert_local	テキストの埋め込み
clip_local	テキスト埋め込み、画像埋め込み
t5_local	テキスト埋め込み、テキスト生成
llama_instruct_local	テキストの生成
qwen2_local	テキストの生成
smollm2_local	テキストの生成

モデルに 1 つの機能がある場合、ドロップダウンは非表示になり、機能が自動的に選択されます。

注釈

- 推論は、HM コントロールプレーンやモデルサーバークラスターではなく、データベースクラスターの CPU で実行されます。大規模なワークロードは、そのクラスターの CPU とメモリで通常のデータベースクエリーと競合します。 - コンポーネントは、内部で `aidb.encode_text` 、*aidb.encode_text_batch* **テキスト入力**が空の場合、ベクターストア出力パスへのバッチ埋め込み)、*aidb.encode_image* 、および `aidb.decode_text` を使用します。各機能の詳細については、AIDB documentation を参照してください。

Using models in Langflow

Langflow でモデルが使用される方法

ハイブリッドマネージャーでは、Langflow は 2 つの EDB コンポーネント **EDB Model Server** と **EDB Embeddings** を介して AI モデルに接続します。どちらのコンポーネントも、ハイブリッドマネージャーによって管理および提供されるモデルエンドポイントと通信します。デフォルトでは外部 API 呼び出しは行われません。

- **EDB Model Server** —展開された LLM にプロンプトを送信し、生成された応答を返します。これは、フロー内の会話のステップ、要約、および推論に使用します。
- **EDB Embeddings** —展開された埋め込みモデルを使用して、テキストをベクトル表現に変換します。これは、フロー内から取得ステップを構築するか、ナレッジベースを設定するときに使用します。

どちらのコンポーネントも、使用前に HM プロジェクトにモデルクラスターを展開する必要があります。設定手順については、*Model Management on Hybrid Manager* を参照してください。

EDB Model Server コンポーネントを使用する

1. Langflow で、左側のコンポーネントパネルで **EDB** を選択します。
2. **EDB Model Server** をキャンバスにドラッグします。
3. コンポーネントで、 **HM Machine User Key** をアクセスキーのグローバル変数に設定します。

4. **Hybrid Manager Model Server Cluster Instance** の下で、使用する展開されたモデルクラスターを選択します。
5. コンポーネントの入力をフローの上流ステップたとえば **チャット入力**または取得結果などに接続します。
6. 出力を **チャット出力** またはフローの次のステップに接続します。
7. **Play** アイコンを選択してコンポーネントを実行します。右下隅の **Model Response** を選択して、出力を検査します。

ドロップダウンで使用可能なモデルは、モデルライブラリーから HM プロジェクトに展開されたモデルです。ハイブリッドマネージャーは内部エンドポイントを管理します。URL を手動で構成する必要はありません。

EDB Embeddings コンポーネントを使用する

1. Langflow で、左側のコンポーネントパネルで **EDB** を選択します。
2. **EDB Embeddings** をキャンバスにドラッグします。
3. **HM Machine User Key** をアクセスキーのグローバル変数に設定します。
4. **Hybrid Manager Model Server Cluster Instance** の下で、埋め込みモデルクラスターを選択します。
5. コンポーネントの **Embedding Model** 出力を、エンベディングモデルを想定するダウンストリームの Langflow ベクターストアまたはレトリーバーコンポーネントに接続します。

EDB Embeddings は、ベクトルを直接生成するのではなく、LangChain `Embeddings` モデルオブジェクトを生成します。ダウンストリームコンポーネントはそれを使用して、クエリまたはインデックス作成時にテキストをエンコードします。独自のテキスト入力はありません。

ナレッジベースと検索

組織のデータのモデル応答を基礎付けるには、EDB Model Server と **EDB Knowledge Base** コンポーネントを組み合わせます。

1. AIDB SQL または *Pipeline Designer concepts* を使用して、HM プロジェクトにナレッジベースを設定します。
2. **EDB Knowledge Base** コンポーネントをフローに追加し、運用プロジェクト、クラスター、およびナレッジベース名を使用して構成します。
3. EDB ナレッジベースと EDB モデルサーバーの間に **プロンプトテンプレート**コンポーネントを追加します。 EDB Knowledge Base 出力をテンプレートのコンテキスト変数に接続し、ユーザーのクエリを質問変数に接続します。テンプレート出力を EDB Model Server 入力に接続します。

このパターンは、HM クラスター内で取得拡張生成 RAG を完全に実装します。ソースドキュメント、埋め込み、およびモデル呼び出しはすべて環境内にあります。

完全な構成のウォークスルーについては、 *EDB Knowledge Base* を参照してください。

環境とサービスの検出

Langflow がハイブリッドマネージャー内で実行される場合

- モデルエンドポイントは HM モデルライブラリーから自動的に検出されます。構成する外部 URL はありません。
- 認証は、グローバル変数として設定した HM マシンユーザーキーを介して処理されます。
- Langflow、モデルサーバーポッド、および Postgres クラスター間のすべてのトラフィックは、プロジェクトの名前空間内に残ります。

ポータルドメインがデフォルトの API エンドポイントであるため、コンポーネントがプロジェクトを解決できない場合は、コンポーネントで **Controls** を選択し、 **Hybrid Manager URL** を `http://istio-internalgateway.istio-system.svc.cluster.local:7080` に設定します。

Flow deployment

Hybrid ManagerHM は、エディタで構築したフローを長期実行の呼び出し可能サービスに変換する展開ライフサイクルで Langflow を拡張します。フローは 3 つのステージを介して移動します。

1. ** *Publishing bundles* ** Langflow プロジェクトから 1 つ以上のフローを HM アセットライブラリーの不変バンドルにパッケージ化します。
2. ** *Deploying flows* ** 独自のパラメーター値、リソース制限、エンドポイントを使用して、HM プロジェクトの管理対象ワークロードとして公開されたバンドルをインスタンス化します。
3. ** *Accessing deployed flows* ** HTTP または MCP を介してフローを呼び出します。アクセスは、HM の他の場所で使用されるのと同じ HM マシンユーザーアクセスキーによって認証されます。

Langflow エディターのフローは、ビルドしたユーザーにプライベートであり、共有エディター環境で実行されます。それと比較して、展開されたフローには次の点があります。

- 安定した HTTP エンドポイントと MCP サーバーエンドポイント、どちらも HM ゲートウェイを介してルーティングされます。
- 独自のパラメーター値、エディターで使用されるビルダーとは無関係に、展開時に提供されます。シークレットであるパラメーター値パスワード、API キー、またはフローがデータベース、モデル API、またはその他のダウンストリームリソースに到達するために使用するトークンは、展開の名前空間に Kubernetes シークレットとして保存されます。
- エディターを開く必要のない独立したライフサイクル展開、更新、展開解除。
- バージョニング各展開は、不変のバンドルバージョンから作成されます。バンドルを更新すると、新しいバージョンが作成されます。展開では、更新するまで展開されたバージョンが実行され続けます。
- アクセス制御展開されたフローは、グローバルにまたはプロジェクトごとに利用できるようにでき、アクセスは特定のハイブリッドマネージャープロジェクトのセットに制限されます。

コンセプト

Term	Meaning
Published flow	A versioned bundle of one or more Langflow flows in the HM Asset Library. Visible at **Asset Library > Published Flows** (*/asset-library/published-flows*). Immutable per version.
Bundle	Synonym for "published flow" when emphasizing that it can contain multiple flows.
Parameter	A value in the flow that uses a Langflow ステップ 2 グローバル変数を設定する. Parameters are deduplicated across all flows in a bundle by variable name.
Credential	A parameter whose input is a Langflow secret field (*SecretStrInput*), typically a password, API key, or token the flow uses to reach a downstream resource such as a database or model API. Not a credential for accessing the deployed flow itself (that is HM machine-user authentication, see *Accessing deployed flows*). Credentials never carry defaults at publish time. Their values are supplied at deploy time and stored as Kubernetes secrets.
Access level	Either **project** (visible only to users with access to one of a chosen list of HM projects) or **global** (visible to all projects). Configured at publish time, editable later.
Deployed flow	A running instantiation of a published flow in an HM project. Visible at **Estate > Flows** (*/estate/flows*). Each deployed flow is isolated in its own Kubernetes namespace.
Runtime	The Langflow runtime container that hosts a deployed flow. Exposes an HTTP API and an MCP server, both routed through the HM gateway.

必要なロール

パブリッシュ/展開ライフサイクルの権限は *User role permissions by use case* で定義されています。

Action	Required role
公開されたフローを表示する	Project Viewer (project-scoped) or any organization role (global)
フローの公開/新しいバージョンの作成	Project Admin or Project Member, for each project granted access
公開されたフローの編集デフォルト、アクセスリスト	Project Admin or Project Member
公開されたフローバージョンを削除する	Organization Admin
公開されたフローを展開する	Project Admin or Project Member
導入されたフローの表示	Project Viewer
導入されたフローの編集/展開解除	Project Admin or Project Member

次に何をするか

- *Publishing bundles* パッケージは Langflow からアセットライブラリーバンドルにフローします。
- *Deploying flows* 公開されたバンドルを、独自のパラメーター値とリソースを使用して実行中の展開に変換します。
- *Accessing deployed flows* HTTP または MCP を介して展開されたフローを呼び出します。

Publishing bundles

バンドル 「公開されたフロー」とも呼ばれるは、HM アセットライブラリー内の 1 つ以上の Langflow フローの不変パッケージであり、名前とバージョンで識別されます。公開は、flow deployment lifecycle の最初のステップです。

パブリッシュダイアログを開く

パブリッシュダイアログには 2 つのエントリーポイントがあり、どちらも同じ 3 ステップのサイドバーを開きます。

- **Langflow エディターから。** フローの編集中に、 **Share > Publish to Asset ライブラリー** を選択します。 HM UI が開き、手順 1 で既に選択した現在のフローが表示されます。
- **アセットライブラリーから.** HM コンソールで **Asset ライブラリー > Published Flows** に移動し、 **+ Add New Flow** を選択します。事前に選択されたフローはありません。

パブリッシュダイアログ

ステップ 1 フローを選択する

1. サイドバーが開き、現在の Langflow プロジェクトのすべてのフローのチェックリストが表示されます。各行には、フロー名と左側にチェックボックスが表示されます。
2. Langflow エディターからダイアログを開いた場合、編集していたフローは既にチェックされています。公開するフローであることを確認します。
3. 同じバンドルに追加のフローを含めるには、それらのチェックボックスをオンにします。バンドルには複数のフローを含めることができます。それらはすべて単一のユニットとして一緒に展開されます。
4. 選択からフローを削除するには、そのチェックボックスのチェックを外します。
5. 少なくとも 1 つのフローが選択されていることを確認します。少なくとも 1 つのフローがチェックされるまで、**Next** ボタンは無効のままです。
6. **Next** を選択して、手順 2 に進みます。

注釈

バンドル内の複数のフローが同じ Langflow グローバル変数を参照する場合、公開されたバンドル内の単一のパラメーターを共有します。そのパラメーターに設定したデフォルトは、それを使用するすべてのフローに適用されます。

ステップ 2 構成する

Field	Required	Notes
Name	Yes	最大 64 文字。アセットライブラリーの表示名として、および展開されたフローの ID の基礎として使用されます。
Description	No	フリーテキスト。公開されたフローの詳細ビューに表示されます。
Version Tag	Yes	最大 128 文字。不透明な文字列 — HM はそれを semver として解釈しません。 *latest* **`をリテラルタグとして使用することは避けてください。** *HM* **は、常に最新の公開バージョンに解決されるエイリアスとして予約します。スキーム** *`1.0.0、2026-05-28、*`rc1`を選択し、一貫して使用します。
Parameters	Conditional	重複排除されたグローバル変数ごとに 1 つのエントリ。パラメーターと資格情報 を参照してください。
Access Level	Yes	**プロジェクト** デフォルトまたは **グローバル** 。 アクセスレベル を参照してください。

Name は公開後は不変であることに注意してください。名前を変更するには、新しいバージョンを公開します。すべての必須フィールドが入力されたら、[**Next**] を選択して公開する前に確認します。

ステップ 3: 確認

確認手順には、構成したすべてが表示されます。 **Publish** を選択すると、バンドルがアセットライブラリーに書き込まれ、公開されたフローリストに戻ります。新しいバンドルは最初のバージョンです。同じ名前での後続のパブリッシュにより、同じバンドルの新しいバージョンが作成されます。

パラメーターと資格情報

パラメーターは、 Langflow global variable を使用するフロー内の値です。HM は、パブリッシュ時に各フローの定義からそれらを抽出し、ダイアログのステップ 2 に表示します。

パラメーターがどのように処理されるかは、その Langflow 入力タイプによって異なります。

Langflow input type	Treated as	Default at publish time	Value at deploy time
SecretStrInput	**Credential**	Not allowed. The input is read-only in the publish dialog.	**Required** . Supplied per deployment, stored as a Kubernetes secret in the deployment's namespace.
Anything else	**Non-credential parameter**	Optional default value	If no default was set and no value is supplied at deploy time, the flow runs without one.

資格情報のデフォルトは、API を介して指定した場合でも、永続化されることはありません。ハイブリッドマネージャーは、公開時に資格情報パラメーターのデフォルトを削除します。これにより、バンドルをプロジェクト間で共有しても安全に保たれます。

重複排除

パラメーターは、グローバル変数名によってバンドル全体で重複排除されます。バンドル内の 2 つのフローがそれぞれ `DB_USER` を参照する場合、パブリッシュダイアログには単一の `DB_USER` パラメーターが表示され、設定したデフォルトは両方のフローで再利用されます。バンドルは、どのフローが各パラメーターを参照するかを記録するため、展開はすべてのコンシューマーに同じ値を配線できます。

アクセスレベル

アクセスレベルは、アセットライブラリーでバンドルを表示できる人、および展開できる人を制御します。

- **プロジェクト**デフォルト。少なくとも 1 つのプロジェクトを選択する必要があります。バンドルは、リストされたプロジェクトの適切なロールを持つユーザーに表示されます。プロジェクトバンドルをグローバルに昇格させたり、プロジェクト間で移動するには、現在リストにあるすべてのプロジェクトにアクセスする必要があります。
- **グローバル**。組織内のすべてのユーザーが閲覧できます。慎重に使用してください。グローバルになると、プロジェクトの deploy ロールを持つ人は誰でもバンドルをインスタンス化できます。

アクセスはサーバーサイドで適用されます。リストと読み取り専用は、`allowed_projects` が要求元のユーザーの許可されたプロジェクトとオーバーラップするバンドル、またはアクセスレベルがグローバルであるバンドルをリストします。

公開されたバンドルの編集

Asset ライブラリー > Published Flows から、変更するバージョンの行の鉛筆アイコンを選択します。編集サイドバーが開きます。以下は編集可能です。

- **説明**

- **アクセスレベル** プロジェクト ↔グローバルおよび **許可プロジェクト**リスト
- 非資格パラメーターの **パラメーターのデフォルト**

以下は、特定のバージョンで不変です。

- **名前**
- **バージョンタグ**
- **バンドル内のフローのセットとその定義**

不変フィールドを変更するには、バンドルの新しいバージョンを公開します。

公開されたバンドルの削除

バンドルバージョンを削除すると、アセットライブラリーから削除されます。バージョンから作成された既存の展開は、実行され続けます。これらは、バンドルのダウンロードしたコピーを保持し、展開が明示的に更新されない限り再取得しません。削除は許可された操作です**必要なロール** を参照してください。

公開されたバンドルを実行中の展開に変更するには、*Deploying flows* を参照してください。

Deploying flows

デプロイにより、 *Publishing bundles* は、ハイブリッドマネージャー HM プロジェクトでホストされる実行中の Langflow ランタイムに変わります。各展開は、独自のパラメーター値、リソース構成、およびエンドポイントを持つ独立したワークロードです。

デプロイメントの開始

Asset ライブラリー > Published Flows に移動します。デプロイするバンドルの行で、 **Deploy** アイコンを選択します。 「 Deploy 」ダイアログが開きます。

Estate > Flows から **+ Add Flow** を選択し、バンドルを選択することにより、同じアクションに到達することもできます。

Deploy ダイアログ

ダイアログには 4 つのステップがあります。

ステップ 1 構成する

Field	Required	Notes
Display Name	Yes	最大 63 文字。エステート UI および Kubernetes リソース名で展開されたフローの名前として使用されます。

次のページに続く

表 53 – 前のページからの続き

Field	Required	Notes
Project	Yes	単一のプロジェクト。ドロップダウンには、展開されたフローを作成する権限があるプロジェクトのみがリストされます。デプロイされたフローは、1 つのプロジェクトにのみ存在します。
Network Access Type	Yes	パブリックかプライベートか。フローの HM ゲートウェイルートが HM ネットワークの外部から到達可能かどうかを判断します。
Runtime Configuration	Yes	フローのパラメーターたとえば、サービスレベル構成などとは別に、Langflow ランタイム自分自身に適用される設定。

展開は 1 つのプロジェクトに属し、作成後に移動することはできないことに注意してください。

Next を選択して、手順 2 に進みます。

ステップ 2 資格情報

バンドル内のすべての資格情報パラメーター Langflow 入力タイプが `SecretStrInput` であるパラメーターがここにリストされています。展開を続行する前に、それぞれを設定する必要があります。

資格情報の値は、Kubernetes シークレットとして展開の名前空間に保存され、環境変数としてランタイムコンテナにマウントされます。これらは展開後は表示されません。Estate UI はそれらを編集します。

公開時にデフォルトが指定された非資格情報パラメーターはそれらのデフォルトを継承しますが、ここでオーバーライドできます。

各資格情報フィールドのプロダクション値を入力します。フィールド名は、フローで使用される Langflow グローバル変数名と一致します `HM_DB_PASSWORD` または `HM_API_KEY` など。すべての必須フィールドに値が設定されるまで、 **Next** ボタンは無効のままです。

Next を選択して、手順 3 に進みます。

ステップ 3 タスク構成

デプロイメントのリソース制約を設定します。フィールドは、基になる Kubernetes リソースモデルのサブセットを表面化します。

Setting	Default	Range
Replicas	1	1–5
CPU request / limit	500m request, 1000m limit	Configurable
Memory request / limit	512Mi request, 1Gi limit	Configurable

運用ワークロードの場合、**レプリカ** を少なくとも 2 に設定して、ポッドが再起動してもフローが利用可能なままになるようにします。CPU とメモリの値が不明な場合は、デフォルトから始めて、**エステート>フロー** で観察された使用状況に基づいて調整します。

Next を選択して、レビューステップに進みます。

ステップ 4: 確認

パスワードがマスクされた場合の、手順 1〜3 のすべての設定の概要。**Deploy** を選択する前に、すべての資格情報フィールドがマスクされた値を表示していることを確認します。空白のフィールドは、手順 2 で資格情報が入力されず、展開の開始に失敗することを意味します。**Back** を使用して、欠損値を返して埋めます。

Deploy を選択すると、展開が送信されます。資格情報の値は Kubernetes シークレットにすぐに書き込まれ、ワークロードの調整が開始されます。新しい展開は、ステータス**プロビジョニング** で **エステート>フロー**に表示されます。プロビジョニングは通常、数分以内に完了します。ステータスが **Ready** に変わると、フローの HTTP および MCP エンドポイントはライブになります。

デプロイメントリソースを理解する

各展開は、独自の分離された名前空間に Kubernetes リソースのセットをプロビジョニングします。

Resource	Purpose
Deployment	Runs the Langflow runtime container. Replicated per the configured replica count.
Service	Exposes the runtime's HTTP API and MCP server inside the cluster. The HM gateway routes external traffic to this Service.
Secret	Holds all parameter values (credentials and non-credentials). Mounted into the runtime container as environment variables.
ServiceAccount	Pod identity used for secure communication with internal HM services.
NetworkPolicy	Restricts pod egress to DNS, the HM services the runtime needs at startup, public HTTPS (port 443), and Postgres (port 5432).

> **注釈**
>
> ネットワークポリシーの許可リスト DNS、HTTPS/443、Postgres/5432 外部のプロトコルまたはポートでホストに到達する必要があるフローは、接続に失敗します。フローで追加の出力が必要な場合は、HM 管理者に連絡してください。

ポッドの起動時に、ランタイムは、HTTP および MCP エンドポイントの準備が整う前に、バンドル展開が

作成されたバージョンに固定されますをダウンロードします。したがって、コールドスタートにはわずかな遅延が発生します。後続の再起動では、同じダウンロード手順を繰り返します。

デプロイメントの管理

- **展開パラメーターの編集。 Estate > Flows** から、展開されたフローを開き、パラメーター値を編集します。保存すると、展開のパラメーター Secret が書き換えられ、ランタイムポッドのローリングリスタートがトリガーされます。ランタイムは、起動時にバンドルを再ダウンロードします展開が作成されたバージョンに固定されたままです。
- **バンドルバージョンの更新。**展開されたフローは、展開されたバージョンに固定されます。同じバンドルの新しいバージョンを実行するには、展開を編集してバージョンを変更するか、展開を解除して新しいバージョンから再度展開します。新しいバンドルバージョンが公開されても、既存の展開は自動アップグレードされません。
- **Undeploy.** エステート UI から、展開されたフローを削除します。HM は、パラメーター値を保持するシークレットを含む展開のリソースを削除します。フローの HM ゲートウェイルートが削除されます。

HTTP または MCP を介してデプロイされたフローを呼び出すには、*Accessing deployed flows* を参照してください。

Accessing deployed flows

Deploying flows は、Langflow ランタイム API と MCP サーバーの 2 つのエンドポイントを公開します。どちらも Hybrid ManagerHM ゲートウェイを介してルーティングされます。クライアントは、HM マシンユーザーアクセスキーを使用してゲートウェイに対して認証します。

エンドポイントの検索

1. HM コンソールで、 **Estate > Flows** `/estate/flows` に移動します。
2. **Status** 列に **Ready** と表示されるまで待ちます。**プロビジョニング** や **Starting** などの以前のステータスは、ランタイムがまだ要求を受け入れていないことを意味します。
3. 行を選択して詳細サイドバーを開きます。
4. **Endpoints** セクションまでスクロールし、必要なエンドポイントの URL をコピーします。

> **注釈**
>
> これらの URL を手動で構成しないでください。ルート ID は展開時に HM によって割り当てられ、予測可能ではありません。

Endpoint label	Purpose
Langflow	The Langflow runtime API for listing and running flows in the bundle.

次のページに続く

表 56 – 前のページからの続き

Endpoint label	Purpose
MCP	Model Context Protocol server endpoint. Used by MCP-capable clients (LLM agents, IDEs) to discover and invoke flows as tools.

各ルートは、同じプロジェクトとアプリの下に独自のルート ID があります。

```
https://<hm-portal-host>/projects/<project-id>/apps/<app-id>/routes/<route-id>
```

たとえば、展開された単一のフローは以下を公開する可能性があります。

```
https://<hm-portal-host>/projects/prj_abc/apps/app-xyz/routes/a3528a9d   # HTTP␣
↪runtime
https://<hm-portal-host>/projects/prj_abc/apps/app-xyz/routes/538275d3   # MCP server
```

<project-id> 、<app-id> 、および 2 つの<route-id> 値は、展開時に HM によって割り当てられ、展開の有効期間にわたって安定しています。

認証

デプロイされたフローへのすべての要求には、x-access-key ヘッダーに HM マシンユーザーアクセスキーを含める必要があります。HM ゲートウェイはこれを強制します。ランタイムポッド自分自身は認証を処理しません。

```
curl -L https://<hm-portal-host>/projects/<project-id>/apps/<app-id>/routes/<route-id>
↪ \
  -H x-access-key: <your-machine-user-key>
```

アクセスキーを作成するには、**マシンユーザーを作成する** を参照してください。マシンユーザーには、展開されたフローのプロジェクトで適切なロールが必要です。**必要なロール** を参照してください。

HTTP ランタイム API

HTTP ランタイムルートのルートに対する GET は、展開されたバンドルと使用可能なエンドポイントのプレーンテキストインデックスを結果ます。

```
Welcome to Langflow Runtime
Loaded flows:
  - My Flow (4944038b-f66c-4358-bb31-0f0b5bb865fd)
  - Another Flow (5cb0ac73-8de9-4f65-96f5-0fab4dd6a195)
HTTP endpoints:
  GET  /flows              - List available flows
  POST /flows/{id}/run     - Execute a flow
  GET  /health             - Liveness probe
  GET  /ready              - Readiness probe
  GET  /docs               - OpenAPI docs
```

Endpoint	Method	Purpose
/	GET	Index. Human-readable list of loaded flows and endpoints.
/flows	GET	JSON list of flows in the deployed bundle, with their IDs and names.
/flows/{id}/run	POST	Execute the flow with the given ID. Request body carries the flow's inputs. Response carries its outputs. See */docs* for the exact schema.
/health	GET	Liveness probe. Used by Kubernetes. Safe for external monitoring.
/ready	GET	Readiness probe. Returns 200 when the runtime has loaded all flows and is ready to serve requests.
/docs	GET	Interactive OpenAPI documentation for the runtime API.

`POST /flows/{id}/run` の権限のある要求/レスポンスシェイプについては、特定の展開に対して`/docs` を開きます。スキーマは、公開したフローの入力と出力を反映します。

例フローのリスト

```
curl -L https://<hm-portal-host>/projects/<project-id>/apps/<app-id>/routes/<route-id>
↪/flows \
  -H x-access-key: <your-machine-user-key>
```

例フローを実行する

```
curl -L -X POST \
  https://<hm-portal-host>/projects/<project-id>/apps/<app-id>/routes/<route-id>/
↪flows/<flow-id>/run \
  -H x-access-key: <your-machine-user-key> \
  -H content-type: application/json \
  -d { "inputs": { "input_value": "hello" } }
```

正確な要求本文は、フロー内の入力コンポーネントによって異なります。ブラウザで`/docs` を開き、ブラウザ拡張機能またはプロキシを介して `x-access-key` を設定して、生成されたスキーマを確認します。

MCP サーバーエンドポイント

MCP サーバールートは、展開されたバンドルのフローをモデルコンテキストプロトコルツールとして公開します。MCP ルート URL で MCP 対応クライアント LLM エージェント、IDE 統合をポイントし、HTTP と同じ `x-access-key` ヘッダーを提供します。クライアントは、各フローを MCP ツールとして検出し、構造化された引数を使用して呼び出すことができます。

MCP プロトコル自分自身については、 MCP specification を参照してください。

ログと可観測性

デプロイされたフローからのログとメトリックは、HM の可観測性スタックに移動し、標準の HM Grafana ダッシュボードを介して表示されます。Prometheus アラートルールについては *Observability (Agent Factory on HM)* を、Langflow ポッドとモデルクラスターパフォーマンスについては Grafana ダッシュボードの設定を参照してください。

一般的なエラー

Symptom	Likely cause
401 Unauthorized from the gateway	Missing or invalid *x-access-key*, or the machine user does not have the role required to access the project's deployed flows.
404 Not Found on the runtime route	Wrong route ID, or the deployment has been undeployed. Re-check the URL in the Estate UI.
/ready returns 503	The runtime is still downloading the bundle at startup, or hasn't finished loading flows yet. Wait and retry. If it persists, check the deployment's logs.
Flow run returns a credential error from a downstream resource	A credential parameter (such as a DB password or model API key) was not supplied at deploy time, or the secret it carries is no longer valid against the downstream system. Edit the deployed flow's parameters from the Estate UI.

Langflow templates

Hybrid Manager の Langflow には、一般的なワークフロー用の事前構築されたフローテンプレートが同梱されています。テンプレートは作業的な開始点を提供します。テンプレートギャラリーから 1 つを開き、いくつかの入力を構成し、プレイグラウンドで実行します。

テンプレートにアクセスするには、HM コンソールの **Launchpad** から Langflow を開き、 **New Flow** を選択して、 **All Templates** を選択します。

Template	What it does
Simple Migration Agent	Automates database migration into Hybrid Manager: creates a destination cluster, migrates the schema, and kicks off data migration.

Simple Migration Agent

Simple Migration Agent は、データベースを Hybrid Manager に移行するプロセスを自動化する事前構築された Langflow テンプレートです。LLM を使用して、移行手順をオーケストレーションします。HM プロジェクトに宛先クラスターを作成し、スキーマを移行し、データの移行を開始します。データの移行が実行されると、エージェントのジョブが完了します。ハイブリッドマネージャーで進行状況を直接監視します。

> **重要**
>
> Simple Migration Agent は、運用環境での使用はお勧めできません。非実稼働環境でのハイブリッドマネージャーへの Postgres から Postgres の移行に最適です。

エージェントを実行する前に、4 つの前提条件すべてを満たしている必要があります。

1. Langflow で **LLM API キー**を構成します。エージェントには、移行手順を解釈および調整するための LLM が必要です。OpenAI をお勧めします。Langflow では、新しいプロジェクトを作成する前に、**プロファイル > 設定 > モデルプロバイダー** に移動し、OpenAI API キーを入力します。
2. ソースデータベースホストに **ビーコンエージェント**をインストールして登録します。ビーコンエージェントはソースデータベースを検出し、登録されたリソースとして HM コンソールに表示されます。続行する前に、HM コンソールのプロジェクトの下にソースデータベースが表示されることを確認してください。
3. **DMS エージェントをインストール**し、ソースデータベースが移行ソースとして表示されることを確認します。DMS Data Migration Service エージェントは、実際のデータ転送を処理します。ソースがリストされ、HM で移行ソースとして利用できることを確認します。
4. **一致するリソース ID を確認します。** DMS エージェント接続と EDB Postgres AI エージェント接続に使用されるリソース ID は、同じである必要があります。リソース ID が一致しないと、エージェントに障害が発生します。テンプレートを実行する前にこれを確認してください。

Simple Migration Agent を実行する

1. HM コンソールの **Launchpad** から Langflow を開きます。
2. **新しいフロー > すべてのテンプレート** を選択します。
3. テンプレートギャラリーから **Simple Migration Agent** を選択します。
4. プロジェクトを作成する前に、 **モデルプロバイダー** が **OpenAI** または選択したプロバイダーに設定されており、API キーが構成されていることを確認します。エージェントが続行するには、動作している LLM 接続が必要です。
5. **Create** を選択して、キャンバスでフローを開きます。
6. キャンバスの左側の **テキスト入力**コンポーネントで、次の形式で必要な入力を提供します。

```
project id: <your-project-id>
source database resource id: <your-source-database-resource-id>
```

- **project id** —デスティネーションクラスターが作成される HM プロジェクトの ID。 - **ソースデータベースリソース ID** —ビーコンエージェントおよび DMS エージェントによって登録されたソースデータベースのリソース ID。これは、両方のエージェント構成で使用される ID と一致する必要があります前提条件 を参照してください。

オプションで、`target cluster name` フィールドを追加して、新しいクラスターを作成する代わりに、既存のクラスターを移行ターゲットとして指定します。

```
project id: <your-project-id>
source database resource id: <your-source-database-resource-id>
target cluster name: <your-target-cluster-name>
```

1. オプションで、 **Cluster Creation Wrapper** コンポーネントで、表示されたテキストボックスに、宛先クラスター構成として使用するクラスターテンプレートの名前を入力します。空のままにするか、テンプレートが存在しない場合、エージェントはデフォルトのシングルノード EDB PaaS Postgres 18 クラスターに設定されます。

2. **Run** を選択します。エージェントは、次の手順を自動的にオーケストレーションします。

- 指定されたテンプレート、または有効なテンプレートが提供されない場合はシングルノード EDB PaaS Postgres 18 クラスターを使用して、HM プロジェクトに新しい宛先クラスターを作成します。
- ソースデータベースからスキーマを移行します destination cluster.
- データの移行をキックオフします。

1. データの移行が開始されたら、 **Hybrid Manager** に移動して、移行の進行状況を監視します。アクティブな移行は、プロジェクトの移行ステータスビューの下に表示されます。

> **注釈**
>
> データの移行が開始されると、Langflow エージェントのロールが終了します。その後の監視と管理はすべて、Langflow ではなくハイブリッドマネージャーで発生します。

エージェントを実行する前に、これらの制限を確認してください。

スキーマの互換性が必要です。エージェントは、スキーマの互換性を検証または修復しません。ソーススキーマに互換性のないオブジェクトたとえば、サポートされていないデータ型、宛先で使用できない拡張機能、または名前付けの競合がある場合、移行は失敗します。エージェントを実行する前にすべてのスキーマの問題を解決します。移行ポータル AI または EDB Copilot を使用して、最初にスキーマの互換性を評価および修正します。

デフォルトで新しい宛先クラスターを作成します。エージェントは、有効なクラスターテンプレート名が**クラスター作成ラッパー**コンポーネントで提供されていない限り、移行ターゲットとして新しいシングルノード EDB PaaS クラスター Postgres 18 をプロビジョニングします。`target cluster name` を **テキスト入力**コンポーネントに追加することにより、既存のクラスターをターゲットとして指定することもできます。

**宛先はハイブリッドマネージャー内である必要があります。エージェントは、HM の外部の外部宛先に移行できません。デスティネーションクラスターは、HM プロジェクト内に存在する必要があります。

実稼働用ではありません。このテンプレートは、評価および非実稼働の移行用に設計されています。これは、ハイブリッドマネージャーへの単純な Postgres から Postgres の移行に最適です。

Security considerations for Langflow

このページでは、フローがフロービルダーから展開されたフローに移動するときに資格情報がどのように処理されるかについて説明します。

グローバル変数の資格情報

フロービルダーのグローバル変数は、資格情報と機密でない構成値を保存するために使用されます。各ステージで資格情報が存在する場所と、それに伴うリスクを理解します。

フロービルダーで

- グローバル変数は暗号化され、Langflow データベースに保存されます。そのデータベースはフロービルダー内のすべてのフローで共有され、すべてのフローとすべてのユーザーのグローバル変数は同じキーで暗号化されます。ユーザーはお互いのグローバル変数を見ることはできませんが、分離はありません。データベースまたは暗号化キーが侵害されると、すべてのユーザーのグローバル変数が公開されます。

フローが公開されたとき

- グローバル変数値は、公開されたバンドルに含まれません。変数名が抽出され、パラメーターとしてバンドルに保存されます。公開時に汎用パラメーターのデフォルト値を設定できます。資格情報パラメーターに値は設定できません。

フローが展開される場合

- 展開時にパラメーターの値を指定します。汎用パラメーターはデフォルトを使用できますが、資格情報パラメーターには値を指定する必要があります。これらの値は、その展開されたフローによってのみ使用される分離された名前空間に Kubernetes シークレットとして保存されます。

おすすめ

- フロービルダーでは、非機密リソースたとえば開発者またはテストデータベースとモデル展開などへの資格情報にのみグローバル変数を使用します。
- フローを展開する場合にのみ運用資格情報を追加します。各展開の資格情報値は、そのフローに分離された Kubernetes シークレットとして保存されます。

関連ページ

- *Sovereign AI on Hybrid Manager* —データの常駐とエアギャップアーキテクチャの概念。
- *Flow deployment* —展開された Langflow フローのプロジェクトレベルの分離、スケーリング、およびアクセス制御。

Langflow FAQ

RAG フローでの取得品質を向上させるにはどうすればよいですか?

悪い検索結果は、通常、次のいずれかが原因で発生します。

- **間違った埋め込みモデル** —ナレッジベースの埋め込みの作成に使用されるモデルは、クエリ時に使用されるモデルと一致する必要があります。`aidb.create_table_knowledge_base` で使用される `model_name` を確認します。
- **チャンクサイズ**—ソースドキュメントが大きなチャンクとして取り込まれた場合、取得スコアにノイズが多くなる場合があります。`aidb.chunk_text_config` を使用して、より小さいチャンクでインデックスを再作成します。
- **Top-k too low** — EDB Knowledge Base コンポーネントは、上位 k の最も類似した結果を結果ます。関連するコンテンツが見逃される場合、これを増やします。
- **インデックスが最新ではありません** — 最後の `aidb.bulk_embedding` 実行以降にソースデータが変更されている場合、取得されたコンテンツは古い場合があります。大量の埋め込みを再実行するか、インクリメンタル同期を設定します。

EDB Model Server と EDB Embeddings の違いは何ですか？

EDB Model Server はプロンプトを LLM に送信し、テキスト応答を返します。生成、要約、および推論の手順に使用します。**EDB Embeddings** は、埋め込みモデルを使用してテキストをベクトルに変換します。類似度を計算するか、Postgres にベクトルを保存する必要がある場合に使用します。どちらも HM プロジェクトにモデルクラスターを展開する必要がありますが、異なるモデルタイプ LLM 対埋め込みモデルを使用します。

“プロジェクトを選択できません” — ポータルドメインはデフォルトで API エンドポイントになります

影響を受ける EDB データベースまたはナレッジベースコンポーネントの上部にある **Controls** を選択し、**Hybrid Manager URL** を次のように設定します。

```
http://istio-internalgateway.istio-system.svc.cluster.local:7080
```

Troubleshooting Langflow in Hybrid Manager

コンポーネント構成エラー

フローをインポートした後、コンポーネントに空のまたは無効なフィールドが表示される

原因 エクスポートされたフロー.json ファイルには、HM マシンのユーザーキー、パスワード、またはグローバル変数値が含まれません。これらは、エクスポートされた Langflow インスタンスにスコープされます。

修正

1. 新しい Langflow インスタンスで、 **プロファイル** → **設定** → **グローバル変数**に移動し、プロダクション値を使用してマシンユーザーキー、データベースユーザー、およびデータベースパスワードの変数を作成します。
2. 各 EDB コンポーネントを開き、 **HM Machine User Key** を正しいグローバル変数に設定します。
3. 正しいリソースを指すようにプロジェクト、クラスター、およびデータベースの選択を再構成します。

“プロジェクトを選択できません” — ポータルドメインはデフォルトで API エンドポイントになります

原因 EDB コンポーネントは、ハイブリッドマネージャー URL をフロントエンドではなく内部 API エンドポイントに解決しています。

修正影響を受ける EDB データベースまたはナレッジベースコンポーネントの上部にある **Controls** を選択し、 **Hybrid Manager URL** を次のように設定します。

```
http://istio-internalgateway.istio-system.svc.cluster.local:7080
```

EDB Knowledge Base コンポーネント

空または無関係の取得結果

症状コンポーネントはエラーなしで実行されますが、結果を返さない、またはクエリと一致しない結果を返します。

診断手順

1. ナレッジベースが存在し、入力されていることを確認します。

```
SELECT * FROM aidb.kbstat WHERE name = your_knowledge_base_name;
```

1. `embeddings` がゼロより大きいことを確認します。そうでない場合は、次を実行します。

```
SELECT aidb.run_pipeline(your_knowledge_base_name);
```

1. コンポーネントの **データベース名** および **ナレッジベース**フィールドが、クラスターで作成されたものと正確に一致することを確認します。

一般的な原因

- ナレッジベースは作成されましたが、 `aidb.run_pipeline` は実行されませんでした。
- ソースデータは、最後の埋め込み実行後に更新されました — `aidb.run_pipeline` を再実行して同期します。
- ナレッジベースの作成に使用された埋め込みモデルは、現在展開されているものと一致しません。 `aidb.create_pipeline(...)` の `aidb.knowledge_base_config(...)` に渡されたモデルは、まだアクティブなモデルクラスターに対応している必要があります。

Postgres クラスターへの接続障害

症状 コンポーネントは実行時に接続エラーを表示します。

診断手順

1. マシンユーザーがプロジェクトに対する **Project Viewer** ロールがあることを確認します。

- **プロジェクト** → **ユーザー** → **プロジェクトロールの割り当て** → **プロジェクトビューア**

1. HM コンソールで Postgres クラスターがアクティブであることを確認します。
2. **データベース接続タイプ** 、 **データベースのユーザー** 、**データベースのパスワード** 、および **データベース名**がすべてクラスターで構成されているものと一致することを確認します。

EDB Model Server コンポーネント

モデルクラスターはドロップダウンで使用できません

原因現在の HM プロジェクトにモデルクラスターが展開されていないか、マシンユーザーに **AI Model Manager** ロールがありません。

修正

1. HM コンソールの **Agent Factory** → **Model ライブラリー**からモデルクラスターを展開します。 *Model Management on Hybrid Manager* を参照してください。
2. マシンユーザーが組織レベルで **AI Model Manager** を持っていることを確認します。

- **プロファイル** → **ユーザー管理** → **組織ロールの割り当て** →**AI モデルマネージャー**

コンポーネントは実行されますが、応答またはエラーが返されません

症状 EDB Model Server コンポーネントはエラーで終了するか、空の応答を返します。

診断手順

1. **Agent Factory** → **Model Server Clusters** でモデルクラスターのステータスを確認します。要求を処理する前に、クラスターは**アクティブ / 正常** を表示する必要があります。
2. クラスターが起動している場合 ステータス保留中、準備が整うまで待って再試行します。
3. Hybrid Manager Grafana インスタンスを使用して、クラスターのポッドログで初期化エラーを確認します。ダッシュボードへのアクセスについては、*Observability (Agent Factory on HM)* を参照してください。
4. コンポーネントの **Input** フィールドが上流コンポーネントに接続されており、フローの実行時に空でないことを確認します。

コンテキストウィンドウを超過しました

症状モデルは、トークン制限または切り捨てられた入力を記載したエラーを返します。

修正 モデルに渡されるコンテキストの量を減らします。フローが EDB ナレッジベースの結果をモデルに直接パイプする場合

- EDB ナレッジベースコンポーネントの **Limit results** 値を下げて、取得するチャンクの数を減らします。
- ソースドキュメントでチャンクステップ AIDB `aidb.chunk_text_config` を使用して、インデックスを作成する前に個々のチャンクサイズを削減します。

EDB Embeddings コンポーネント

ドロップダウンで使用可能なモデルクラスターがありません

EDB Model Server と同じ原因と修正 —上記の **モデルクラスターはドロップダウンで使用できません** を参照してください。

埋め込み生成が失敗する

診断手順

1. モデルクラスターが LLM ではなく埋め込みモデルを使用していることを確認します。2 つは互換性がありません。
2. **Agent Factory** → **Model Server Clusters** で、クラスターがアクティブであることを確認します。
3. Hybrid Manager Grafana インスタンスを使用してポッドログを確認します。

フローレベルの問題

フローは Playground で動作しますが、別のハイブリッドマネージャーインスタンスに移動すると失敗します

考えられる原因

- 実稼働 Langflow インスタンスのグローバル変数が設定されていないか、不正な値を参照します。
- 実稼働マシンのユーザーには、異なるまたは欠落しているロールの割り当てがあります。
- 運用ナレッジベースまたはモデルクラスターには、開発のものとは異なる名前があります。

すべての EDB コンポーネントが正しいプロジェクト、クラスター、データベースを指していること、およびそのグローバル変数が有効な値を保持していることを確認します。デプロイ時に設定されるパラメーターと資格情報を使用して管理サービスとしてフローを実行するには、*Flow deployment* を参照してください。

フローは実行されますが、一貫性のない回答または低品質の回答が返されます

取得品質から始めます —RAG フローの回答品質の問題のほとんどは、モデルではなく EDB ナレッジベースコンポーネントが返すものに遡ります。 **空または無関係の取得結果** と *Langflow FAQ* の取得品質ガイダンスを参照してください。

その他のヘルプ

- *Langflow FAQ* —コンポーネント、グローバル変数、およびフローの展開に関するよくある質問。
- *Observability (Agent Factory on HM)* —Langflow ポッドの状態とモデルクラスターパフォーマンスの Grafana ダッシュボードと Prometheus モニタリング。
- *Troubleshooting Langflow in Hybrid Manager* —インフラレベルの問題 GPU、InferenceService、ストレージ。

6.4.4.2 Pipeline Designer

パイプラインデザイナー は、AI Database (AIDB) データパイプラインを構築および管理するためのハイブリッドマネージャー HM のビジュアルインターフェイスです。AI および ML 変換ステップを介してデータを処理するパイプラインを作成し、オプションで取得拡張生成 RAG 用のベクトルナレッジベースに結果を保存するためのガイド付きのノーコードワークフローを提供します。

パイプラインデザイナーを使用すると、次のことができます。

- ビジュアルウィザードを使用して、シングルステップまたはマルチステップのデータパイプラインを作成します。
- セマンティック検索用の埋め込みベクトルストレージを使用してナレッジベースを構築します。これには、マルチプルのパイプラインが単一のナレッジベースをフィードするマルチパイプラインナレッジベースを含みます。
- 処理モードライブ、バックグラウンド、またはオンデマンドを構成します。
- パイプラインステータス、レコード数、およびエラー条件を監視します。
- ビルトインクエリツールを使用してナレッジベースの取得をテストします。
- HM ホストモデル、HM 外部推論サービスプロキシ、AIDB ネイティブモデル、またはビルトインのデータベースモデルを使用します。
- プライマリ HM クラスター、セカンダリ HM の場所、または自己管理の Postgres インスタンスでパイプラインを実行します。

6.4.4.2.1 Pipeline Designer と AIDB の関係

Pipeline Designer は、AIDB extension のグラフィカルフロントエンドです。デザイナーを介して作成したすべてのパイプラインは、AIDB SQL 呼び出し `aidb.create_pipeline` 、`PipelineStepOperation` 、`PipelineAutoProcessingMode` 、および Postgres データベースで実行されている関連ファンクションに直接変換されます。Pipeline Designer は、独自の概念的語彙を導入しません。ビジュアルインターフェイスを介して AIDB パイプライン、ナレッジベース、およびモデルを表示します。基本的な概念については、AIDB documentation を参照してください。

Pipeline Designer は、作成するパイプラインのみを管理します。SQL パイプラインデザイナー UI の外部を介して直接作成されたパイプラインは、デザイナーインターフェイスに表示されません。この分離が存在する理由の詳細については、*VPU and permissions* を参照してください。

6.4.4.2.2 前提条件

最初のパイプラインを作成する前に、**セットアップの前提条件** の設定手順を完了してください。セットアップチェックリストには、拡張機能のインストール、クラスター構成、プラットフォームロール、モデル登録、およびソーステーブルの権限が含まれます。*Getting started with Pipeline Designer* ページには、パイプラインデザイナーをすぐに試すために使用できるソーステーブルの例も含まれています。

Pipeline Designer は、完全な AIDB 機能セットのサブセットを提供します。主な制限には、UI でまだサポートされていないボリュームベースのソースが含まれます。完全なリストについては、*Limitations* を参照してください。

6.4.4.2.3 サポートされているクラスタータイプ

パイプラインデザイナーは、次のクラスター構成をサポートしています。

- **HM 管理クラスター**プライマリ/スタンバイレプリケーション PSR および Postgres DistributedPGD/AHA プライマリの場所のハイブリッドマネージャーによってプロビジョニングおよび管理されるクラスター。
- **セカンダリ HM ロケーション**マルチロケーション展開の追加の HM ロケーションで実行されている HM 管理クラスター。
- **自己管理の Postgres インスタンス** AIDB がインストールされている EDB Postgres AI エージェント `beacon-agent` を介して HM に登録された単一インスタンスの Postgres サーバー。外部 PGD クラスターと Cloud Native PostgresCNP クラスターはサポートされていません。

サポートされている最小の Postgres メジャーバージョンは 16 で、サポートされている最小の AIDB 拡張バージョンは 7.2.0 です。これらのクラスタータイプでの実行の詳細については、*Executing pipelines* を参照してください。

- *Getting started with Pipeline Designer* 環境をセットアップし、最初のパイプラインを作成します。
- *Pipeline Designer concepts* パイプライン構造、マルチステップデザイン制約、および処理モード選択について学びます。
- *Creating pipelines* パイプライン作成ウィザードをウォークスルーします。
- **ナレッジベースと検索**ベクターストレージ、モニタリング、およびクエリーを理解します。
- *External inference services* 外部モデルプロバイダーを登録します。
- *Executing pipelines* さまざまなクラスタータイプでパイプラインを実行します。
- *VPU and permissions* 権限モデルを理解します。
- *Limitations* デザインコンストレインと電流制限。

Getting started with Pipeline Designer

以下のセットアップ手順と 2 つの完全なパイプライン例に従って、パイプラインデザイナーを実行します。この例では、テキスト要約パイプラインとナレッジベースパイプラインをカバーし、ソーステーブルの準備から結果検証までのコアワークフローをガイドします。

セットアップの前提条件

パイプラインデザイナーには、`visual_pipeline_user` VPU と呼ばれる専用の Postgres ロールが必要です。パイプラインデザイナーは、モデルレジストリの照会、宛先テーブルの作成、パイプラインステップの実行を含むすべてのパイプライン操作を実行するために使用します。このロールは自動的に作成されません。パイプラインデザイナーを使用する前に、手動で作成する必要があります。モデルピッカーは VPU として AI Database AIDB モデルレジストリを照会するため、ロールが存在しない場合、モデルピッカーは空で、パイプラインの作成は続行できません。設定手順については、Creating VPU role を参照してください。AIDB 拡張機能は、クラスタータイプに応じて、プリインストールされる場合とされない場合があります。

Deployment type	Extension installed at provisioning	Model picker populated before first pipeline
Postgres Distributed (PGD) managed	Yes	No (empty until VPU exists)
Primary/Standby Replication (PSR) managed	No	No (empty until extension and VPU exist)
Self-managed (any)	No	No (empty until extension and VPU exist)

以下の手順は依存関係によって並べられているため、新しいデータベースをセットアップするとき、または既存の展開にソーステーブルを追加するときに適用される手順を特定できます。

Step	Scope	Applies to	Re-run when
shared_preload_libraries	Instance	Self-managed	PG major upgrade
max_worker_processes	Instance	All	Pipeline concurrency growth
CREATE EXTENSION aidb CASCADE	Database	PSR managed, self-mgd	New database
aidb.bdr_setup()	Database	PGD self-managed	New database, extension upgrade
GRANT CREATE ON SCHEMA public	Database	All	New database
beacon-agent DDL privileges	Database	Self-managed	New cluster registration
CREATE ROLE visual_pipeline_user + *aidb_users* grant	Instance	All	Never (one-time)
Hybrid Manager (HM) roles	Platform	All	New user onboarding
Inference service registration	Platform	All	New model provider
Source table grants	Table	All	New tables, schema changes
Pipeline object creation	Pipeline	All	Automated (no manual action)

パイプラインデザイナーを初めてセットアップする場合は、以下のすべてのセクションを順番に実行します。

既存のクラスター上の新しいデータベースにパイプラインデザイナーを追加する場合、パイプラインコンカレンシーが変更された場合は **クラスターレベルのセットアップ** から開始し、**データベースごとのセットアップ** まで進みます。ロールがインスタンスに既に存在する場合、Creating VPU role をスキップします。

新しいソーステーブルを既存のパイプライン展開に追加するだけの場合、**ソーステーブルごとのセットアップ** に直接移動します。

クラスターレベルのセットアップ

Postgres インスタンスごとに 1 回、クラスターレベルの設定を適用します。Postgres のメジャーアップグレードの後、またはパイプラインの同時実行数をスケーリングする場合、これらの設定を再利用します。

- 自己管理クラスターでは、AIDB 拡張機能を作成する前に、`postgresql.conf` の `shared_preload_libraries` に `aidb` を追加し、Postgres を再起動する必要があります。

```
shared_preload_libraries = aidb
```

完全な自己管理構成手順については、AIDB configuration documentation を参照してください。

AIDB は Postgres バックグラウンドワーカーを使用して、バックグラウンド処理モードでパイプラインを実行します。デフォルトの `max_worker_processes` 値 8 は、複数の同時パイプラインを使用するワークロードには不十分である場合があります。ワーカースロットが不十分であると、パイプラインがサイレントに起動に失敗します。

> **Tip**
>
> Pipeline Designer はエラーも再試行も表示しないため、パイプラインは永久にスタックしているように見えます。パイプラインが予期せずアイドル状態になった場合は、早めに `max_worker_processes` をチェックしてください。

```
max_worker_processes = 64
```

推奨値は、バックグラウンドワーカーを消費する同時パイプラインおよびその他の拡張機能の数によって異なります。値 64 は、パイプラインデザイナーワークロードの適切な開始点です。

データベースごとのセットアップ

パイプラインデザイナーを使用するデータベースごとに、このセクションのすべての手順を繰り返します。既存のクラスターに新しいデータベースを作成する場合、ここに戻ります。

PGD 管理対象クラスターでは、AIDB 拡張機能はクラスタープロビジョニング中に自動的に作成されます。手動での拡張機能のインストールは必要ありません。

PSR 管理対象および自己管理クラスターでは、プロビジョニング時に拡張機能は作成されません。Pipeline Designer モデルピッカーにデータを入力するモデル同期には、使用可能なモデルを検出するための拡張機能が必要です。これがないと、モデルピッカーは空で、パイプラインの作成は続行できません。パイプラインデザイナーを使用する前に拡張機能を手動でインストールします。

```
CREATE EXTENSION IF NOT EXISTS aidb CASCADE;
```

```
SELECT aidb.bdr_setup();
```

管理対象 PGD クラスターでは、この手順はプロビジョニング時に自動的に実行されます。`ALTER EXTENSION aidb UPDATE;` で AIDB 拡張機能をアップグレードした後、`bdr_setup()` を再実行します。移行スクリプトが既に呼び出している場合、このファンクションは no-op であるため、各アップグレードの後に無条件に実行するのが安全です。

```
GRANT CREATE ON SCHEMA public TO visual_pipeline_user;
```

テーブルごとの付与は **ソーステーブルごとのセットアップ**で構成されます。オンデマンドモードの場合、これは `SELECT` 付与です。ライブモードおよびバックグラウンドモードの場合、VPU にソーステーブルの所有者ロールを付与する必要があります。必要な付与の完全なセットについては、*VPU* **権限の構成** を参照してください。これらの付与がないと、パイプラインの作成または実行は権限エラーで失敗します。

AIDB documentation を参照してください。

`visual_pipeline_user` ロールは、パイプラインデザイナーがすべてのパイプライン操作に使用するクラスター全体の Postgres ロールです。モデルピッカーは VPU として AIDB モデルレジストリを照会するため、パイプラインデザイナーを開く前に存在する必要があります。ロールが見つからない場合、モデルピッカーは空で、パイプラインの作成は続行できません。

Postgres インスタンスごとに 1 回ロールを作成します。VPU が参加する必要がある `aidb_users` ロールは AIDB 拡張機能によって作成されるため、最初に拡張機能が少なくとも 1 つのデータベースに存在する必要があります。

- **Postgres 16 以前**では、バックグラウンドワーカー認証がロールベースのログインを使用するため、VPU は `LOGIN` 特権が必要です。

```
CREATE ROLE visual_pipeline_user LOGIN;
GRANT aidb_users TO visual_pipeline_user;
GRANT visual_pipeline_user TO edb_beacon;
```

- **Postgres 17 以降**では、バックグラウンドワーカーの認証が依存しないため、`LOGIN` は必要ありません。

```
CREATE ROLE visual_pipeline_user;
GRANT aidb_users TO visual_pipeline_user;
GRANT visual_pipeline_user TO edb_beacon;
```

この手順は 1 回限りのインスタンスレベルであり、すべての展開タイプ PGD 管理、PSR 管理、および自己管理に適用されます。データベースまたはテーブルを追加するときにそれを繰り返す必要はありません。VPU が必要とするテーブルごとのグラントの完全なセットについては、*VPU* **権限の構成** を参照してください。

プラットフォームレベルのセットアップ

プラットフォーム構成は、 Postgres ではなく EDB Postgres AI ポータルで実行されます。データベースまたはテーブルを追加するときにこれらの手順を繰り返す必要はありません。

2 つの HM ロールは、パイプラインデザイナー機能へのアクセスを制御します。

1. **パイプラインエディタープロジェクトロール。**プロジェクト内のパイプラインを作成、編集、および削除するために必要です。
2. **AI Model Manager 組織ロール。** [**設定**] タブにアクセスするために必要です。外部推論サービスが登録および管理されます。

パイプラインデザイナーでの作業を開始する前に、関連するユーザーにこれらのロールが割り当てられていることを確認します。

モデル支援ステップ SummarizeText や KnowledgeBase などを使用するパイプラインを作成する前に、少なくとも 1 つの推論サービスを登録する必要があります。登録された推論サービスがないと、パイプラインデザイナーのモデルピッカーには提供するモデルがありません。使用可能な登録パス HM 外部推論プロキシと AIDB ネイティブ登録については、*External inference services* を参照してください。

ソーステーブルごとのセットアップ

新しいソーステーブルをパイプラインに追加するたびに VPU アクセスを付与し、スキーマが変更されたときに付与を更新しますたとえば、テーブルが別のスキーマに移動したり、新しいテーブルが追加されたとき。

必要な付与 オンデマンドモードの SELECT 付与、またはライブおよびバックグラウンドモードのソーステーブルの所有者ロールおよびトポロジ固有の指示パブリックスキーマ、カスタムスキーマ、および PGD クラスターについては、*VPU* **権限の構成** を参照してください。

パイプラインごとのセットアップ

パイプラインレベルのデータベースオブジェクトは自動的に管理されます。パイプラインデザイナーを介してパイプラインを作成すると、ビーコンエージェントは、宛先テーブル、ナレッジベースベクトルテーブル、トリガーファンクション、パイプライン状態追跡テーブルの作成を含む、パイプラインごとのすべてのデータベース操作を処理します。パイプラインごとの手動設定は必要ありません。VPU が作成するオブジェクトの詳細については、*VPU* **によって作成されたオブジェクト** を参照してください。

例テキストの要約

この例では、ソーステーブルからテキストコンテンツを取得し、要約用のテキスト補完モデルを介して実行し、要約を宛先テーブルに書き込むシングルステップパイプラインを作成します。

ソーステーブルを作成する

データベースに接続し、サンプルテキストコンテンツを含むテーブルを作成します。

```
CREATE TABLE source_table_st (
    id INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL
```

(次のページに続く)

(前のページからの続き)

```
);

INSERT INTO source_table_st (content) VALUES
(I dont want a babysitter. I am eleven years old. My babysitter is only three years␣
↪older than I am," she loudly yelled to her Mom. Now, she really wished she had␣
↪somebody with her as she heard the clicking, scratching noises outside of the␣
↪living room window. "This is silly. Its probably the storm," the girl said. She␣
↪regretted watching the horror show she had been tuned into for the last half hour.␣
↪As she searched for the remote to turn off the vampire movie, the front door blew␣
↪open with a thunderous noise. Carla whirled around to see a dark image.),
(There are times when the night sky glows with bands of color. The bands may begin as␣
↪cloud shapes and then spread into a great arc across the entire sky. They may fall␣
↪in folds like a curtain drawn across the heavens. The lights usually grow brighter,␣
↪then suddenly dim. During this time the sky glows with pale yellow, pink, green,␣
↪violet, blue, and red. These lights are called the Aurora Borealis. Some people␣
↪call them the Northern Lights. Scientists have been watching them for hundreds of␣
↪years. They are not quite sure what causes them. In ancient times people were␣
↪afraid of the Lights. They imagined that they saw fiery dragons in the sky. Some␣
↪even concluded that the heavens were on fire.);
```

ソーステーブルへの VPU アクセスを付与します。この例ではバックグラウンド処理を使用するため、VPU にソーステーブルの所有者ロールを付与する必要があります。

```
GRANT <source_table_owner_role> TO visual_pipeline_user;
```

<source_table_owner_role> を、source_table_st を所有するロールに置き換えます。代わりにオンデマンドモードを使用する場合は、SELECT ON source_table_st TO visual_pipeline_user のみを付与します。この例では、パブリックスキーマをカバーします。カスタムスキーマまたは PGD クラスターについては、*VPU* **権限の構成** を参照してください。

パイプラインを作成する

1. EDB Postgres AI ポータルで、**Sovereign AI > Pipelines** に移動し、**Create Pipeline** をクリックします。
2. パイプライン名、たとえば summarize_demo を入力します。
3. **クラスター** と **データベース** を選択します。
4. **処理タイプ** を **背景** に設定します。
5. **Next** を選択します。
6. **スキーマ** public 、 **テーブル** source_table_st 、**キー列** id 、および **データ列** content を選択します。
7. ソースと宛先間の矢印の **+** ボタンを選択し、 **SummarizeText** を選択します。
8. **SummarizeText** ステップ構成で、モデルピッカーから**テキスト補完モデル** を選択します。使用可能なモデルは、**プラットフォームレベルのセットアップ** 中に登録した推論サービスによって異なります。

9. **展開** を選択します。

パイプラインは **Creating** ステータスでパイプラインリストに表示され、セットアップが完了すると **Up To Date** に移行します。

> **注釈**
>
> バックグラウンド処理は展開後に自動的に実行され、手動介入なしで新しい行および既存の行を処理します。ライブモードとオンデマンドモードについては、 処理モード を参照してください。

バックグラウンド処理を使用すると、パイプラインは展開後に自動的に処理を開始します。ステータスが **Creating** から **Up To Date** に変わるまで、パイプラインリストを監視します。これは、すべてのソースレコードが処理されたことを示します。

パイプラインが **Up To Date** に到達したら、宛先テーブルを照会して結果を確認します。

```
SELECT * FROM pipeline_summarize_demo;
```

宛先テーブルには、ソースレコードごとに 1 行が含まれ、 content 列はモデルのサマリーに置き換えられます。たとえば、オーロラに関する一節は、オーロラ、その外観、歴史に関する重要なポイントをキャプチャした要約を生成します。

例 ナレッジベースを構築する

この例では、製品の説明をセグメントにチャンクし、ベクター埋め込みを生成し、セマンティック検索のためにナレッジベースに保存する 2 ステップのパイプラインを作成します。このワークフローは、パイプラインデザイナーで RAG 対応ベクターストアを構築するための標準パターンです。

ソーステーブルを作成する

データベースに接続し、製品データを含むテーブルを作成します。

```
CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    product_name TEXT NOT NULL,
    description TEXT
);

INSERT INTO products (product_name, description) VALUES
(Classic Burger, A juicy beef patty with lettuce, tomato, and pickles on a toasted␣
↪sesame bun),
(Margherita Pizza, Wood-fired pizza with San Marzano tomatoes, fresh mozzarella, and␣
↪basil),
(Caesar Salad, Crisp romaine lettuce with parmesan, croutons, and house-made Caesar␣
↪dressing),
(Fish Tacos, Grilled mahi-mahi with cabbage slaw, lime crema, and pickled onions),
(Falafel Wrap, Crispy chickpea falafel with tahini, pickled turnips, and fresh herbs␣
↪in warm pita),
```

(次のページに続く)

(前のページからの続き)

```
(Pad Thai, Stir-fried rice noodles with shrimp, peanuts, bean sprouts, and tamarind␣
↪sauce),
(Beef Burrito, Seasoned beef with rice, black beans, cheese, and salsa in a flour␣
↪tortilla),
(Mushroom Risotto, Creamy arborio rice with mixed wild mushrooms, parmesan, and␣
↪truffle oil),
(Lamb Kebab, Spiced lamb skewers with grilled vegetables, garlic yogurt, and crisp␣
↪pitta);
```

ソーステーブルへの VPU アクセスを付与します。この例ではバックグラウンド処理を使用するため、VPU にソーステーブルの所有者ロールを付与する必要があります。

```
GRANT <source_table_owner_role> TO visual_pipeline_user;
```

<source_table_owner_role> を、products を所有するロールに置き換えます。代わりにオンデマンドモードを使用する場合は、SELECT ON products TO visual_pipeline_user のみを付与します。これはパブリックスキーマをカバーします。カスタムスキーマまたは PGD クラスターについては、*VPU* **権限の構成** を参照してください。

パイプラインを作成する

1. EDB Postgres AI ポータルで、 **Sovereign AI > Pipelines** に移動し、**Create Pipeline** を選択します。
2. パイプライン名、たとえば product_kb を入力します。
3. **クラスター** と **データベース** を選択します。
4. **処理タイプ** を **背景** に設定します。
5. **Next** を選択します。
6. **スキーマ** public 、 **テーブル** products 、 **キー列** id 、および **データ列** description を選択します。
7. + ボタンを選択し、 **ChunkText** を選択します。**目的のチャンク長** たとえば 200 を設定します。このような短い説明の場合、チャンクステップは分割せずにそれらを通過させますが、後で長いテキストを処理する可能性のあるパイプラインにそれを含めることをお勧めします。
8. ChunkText ステップの後に + ボタンを選択し、 **KnowledgeBase** を選択します。
9. **KnowledgeBase** ステップ構成で、モデルピッカーから**テキスト埋め込みモデル** を選択し、 **データ形式** を **テキスト**に設定します。
10. **展開** を選択します。

> **注釈**
>
> バックグラウンド処理は展開後に自動的に実行され、手動介入なしで新しい行および既存の行を処理します。ライブモードとオンデマンドモードについては、 **処理モード** を参照してください。

バックグラウンド処理を使用すると、展開後にパイプラインが自動的に実行されます。パイプラインステータスが **Up To Date** に達するまで待ち、ビルトインクエリツールを使用してナレッジベースを確認します。

1. **Sovereign AI > Knowledge Bases** に移動します。
2. パイプラインによって作成されたナレッジベースを選択しますその名前はパイプライン名に由来しています。
3. **クエリ** を選択します。
4. 検索用語たとえば `seafood` または `spicy` を入力します。
5. **Run** を選択します。

クエリツールは、意味的に最も似ている製品の説明とその距離スコアを結果ます。たとえば、**seafood** というワードが説明に表示されない場合でも、 **seafood** を検索すると、最も近い一致としてフィッシュタコスエントリが返されます。これは、ベクトル埋め込みがキーワードマッチングに依存するのではなく、意味論的な意味をキャプチャする方法を示しています。

データベースレベルで生のパイプライン出力を確認することもできます。

```
SELECT * FROM pipeline_product_kb;
```

ナレッジベースは、 *Langflow FAQ* エージェントなどのダウンストリームアプリケーションで使用できるようになりました。統合の詳細については、*Langflow* **でナレッジベースを使用する** を参照してください。

次のステップ

- *Pipeline Designer concepts* パイプライン構造、マルチステップデザイン制約、および処理モード選択について学びます。
- *Creating pipelines* すべてのステップタイプとモデル選択を含む、パイプライン作成ウィザードの完全なリファレンス。
- **ナレッジベースと検索**ナレッジベースを監視し、レコード数を理解し、テスト取得。
- *VPU and permissions* `visual_pipeline_user` ロール、付与の詳細、およびセキュリティの考慮事項を理解します。
- *Limitations* 現在のデザインコンストレインと既知の問題。

Pipeline Designer concepts

Pipeline Designer は、 AI Database (AIDB) pipelines のビジュアルインターフェイスです。ガイド付きウィザードを介して AIDB 機能を表示および制約します。以下のコンセプトは、デザイナーが適用する内容とその理由をカバーしています。基本的なパイプライン、ナレッジベース、およびモデルの概念については、AIDB documentation を参照してください。

パイプライン構造

パイプラインは、データをソーステーブルから宛先に変換する 1〜10 の処理ステップのシーケンスです。*Creating pipelines* を介してソース、ステップ、および宛先を構成します。

パイプライン名は小文字であり、文字で始まり、文字、数字、アンダースコアのみを含む必要があります。最大長は 46 文字です。AIDB は、内部サフィックス用の Postgres 識別子制限の最大 17 文字を予約しています。これらのコンストレインは、*Creating pipelines* にパイプライン名を入力すると検証されます。

ステップタイプ

Pipeline Designer は、 AIDB pipeline steps および knowledge base 機能をパイプラインステップとして公開します。各操作の内容、そのパラメーター、およびその入出力タイプの詳細については、AIDB pipeline steps documentation を参照してください。

Step type	Input	Output	Use case
ParseHTML	Text (HTML)	Text	Strip HTML markup and extract clean text
ParsePDF	Bytes (PDF)	Text	Extract text content from PDF files
PdfToImage	Bytes (PDF)	Bytes (image)	Convert PDF pages to images, typically before OCR
PerformOCR	Bytes (image)	Text	Extract text from scanned images or image-based PDFs
ChunkText	Text	Text	Split long text into smaller segments
SummarizeText	Text	Text	Condense text using a completion model
KnowledgeBase	Text	Vector	Generate embeddings and store in a vector knowledge base

マルチステップパイプライン

パイプラインは、最大 10 のステップをチェーンできます。パイプラインキャンバスでは、ステップは**テーブルソース** と **宛先テーブル**エンドポイントの間に左から右に配置されたカードとして表示され、その位置にステップを挿入するための各カードのペアの間に**+** コントロールがあります。 。

デザイナーによって適用された順序制約

- 生のドキュメントデータ ParsePDF、PdfToImage を使用するステップを最初に配置する必要があります。PerformOCR はバイナリ入力も受け入れますが、パイプラインの後半たとえば、PdfToImage の後などに表示される場合があります。
- KnowledgeBase ステップが存在する場合、最後に配置する必要があります。
- 中間ステップ ChunkText、SummarizeText、ParseHTML は、最初と最後の位置の間で任意の順序で表示できます。

タイプの互換性

各ステップは特定のタイプテキストまたはバイトの出力を生成し、次のステップはそのタイプを入力として受け入れる必要があります。デザイナーはこれを自動的に検証し、互換性のないステップシーケンスが展開されるのを防ぎます。たとえば、バイトをテキストに変換する中間 OCR ステップがなければ、PdfToImage ステップバイト出力を生成する直後に ChunkText ステップテキスト入力を想定することはできません。

一般的なマルチステップパターン

Pattern	Steps	Use case
PDF to knowledge base	ParsePDF, ChunkText, KnowledgeBase	Index PDF documents for semantic search
Image OCR to knowledge base	PerformOCR, ChunkText, KnowledgeBase	Extract and index text from scanned documents
HTML processing	ParseHTML, ChunkText, SummarizeText	Clean, chunk, and summarize web content
Full document pipeline	ParsePDF, ChunkText, SummarizeText, KnowledgeBase	Parse, chunk, summarize, and index documents

パイプラインは 10 ステップに制限されており、作成後にステップ構造を変更することはできません。詳細は、**パイプラインの作成と構造** を参照してください。

処理モード

パイプラインの作成中に、3 つの AIDB 自動処理モードのいずれかを選択します。作成後に、パイプラインを再作成せずにモードを変更できます。

Mode	How it works	When to use
On Demand (default)	Processes rows only when triggered manually via SQL	Testing, ad-hoc runs, or pipelines where you control exactly when processing occurs
Live	Processes rows immediately as they are inserted or updated, via database triggers	Low-latency use cases where results must reflect writes with minimal delay
Background	Processes rows asynchronously in batches on a configurable interval	Most production workloads. Handles large backlogs and continuous ingestion without blocking writes

バックグラウンドモードでは、デザイナーは 2 つの追加コントロール、バッチサイズデフォルト 100 と同期間隔デフォルト 30 秒、1 秒〜2 日の範囲を公開します。各モードの動作の完全な説明については、AIDB auto-processing documentation を参照してください。

データフローとステップ結果

パイプラインが実行されると、データは AIDB の内部エンベロープを介してあるステップから次のステップに流れます。AIDB がデータリネージ、チャンク識別子、およびステップツーステップデータ転送を処理する方法の詳細については、AIDB pipelines documentation を参照してください。

パイプライン作成ウィザードでは、各中間ステップに **ステップ結果の保存**トグルデフォルトで有効になっています。有効にすると、ステップの出力は宛先テーブルに永続化され、最終出力だけでなくパイプラインの各ステージで中間結果を検査できます。トグルは最後のステップでは非表示になり、出力は常にパイプ

ラインの宛先に書き込まれます。このトグルがウィザードで表示される場所については、*Creating pipelines* を参照してください。

Creating pipelines

Pipeline Designer は、2 フェーズウィザードを使用してパイプラインを作成します。最初のフェーズでは、パイプラインのターゲットクラスター、データベース、および処理オプションを構成します。2 番目のフェーズでは、ソーステーブル、処理ステップ、および宛先を定義します。

フェーズ 1-パイプラインオプション

1. Hybrid ManagerHM プロジェクトで **Sovereign AI > Pipelines** に移動します。
2. **Create New Pipeline** を選択します。
3. **パイプライン名**を入力します。名前は小文字で、文字で始まり、文字、数字、アンダースコアのみを含み、長さは最大 46 文字である必要があります。
4. **クラスター**を選択します。ドロップダウンには、プロジェクト内のすべての資格のあるクラスターが表示されます。

- **HM 管理クラスター**プライマリ/スタンバイレプリケーション PSR または Postgres DistributedPGD/AHA) PGD クラスターの場合、データグループも選択します。Witness グループは除外されます。書き込みリーダーノードのみがパイプライン実行の対象です。
- **自己管理の Postgres インスタンス** EDB Postgres AI エージェント `beacon-agent` を介して HM に登録された単一インスタンスの Postgres サーバー。外部 PGD クラスターとクラウドネイティブ PostgresCNP クラスターはサポートされていません。
- Postgres 16 以降を実行しているクラスターのみが表示されます。レプリカクラスターは除外されます。

1. 選択したクラスターで **データベース** を選択します。
2. **処理タイプ** を構成します。

- **オンデマンド** デフォルト自動処理なし。実行を手動でトリガーします。デプロイ後、オンデマンドパイプラインは、初めて実行するまで「結果なし」のステータスを表示します。
- **ライブ** 行の変更によってトリガーされるリアルタイム処理。**処理モード** .
- を参照してください **バックグラウンド**構成可能な間隔での非同期バッチ処理。

1. バックグラウンドを選択した場合、 **バッチ処理サイズ** デフォルト 500 と**同期インターバル** デフォルト 15 時間を構成します。
2. **Next** を選択して、ステップ構成に進みます。

フェーズ 2-ステップ構成

ステップデザイナーには、2 つの固定エンドポイントを持つビジュアルパイプラインが表示されます。左側の**テーブルソース** カードと、右側の **宛先テーブル**カード。これらのエンドポイント間にステップを追加することにより、パイプラインを構築します。

ソースの構成

1. **テーブルソース** カードを選択します。
2. ソーステーブルを含む **スキーマ** を選択します。
3. パイプラインソースとして使用する **ソーステーブル** を選択します。
4. **ソースキー列**これは、各行を識別する一意の値を持つ列通常は主キーまたは一意の識別子です。
5. **ソースデータ列**処理するコンテンツテキスト、HTML、PDF バイト、またはイメージバイトを含む列を選択します、最初のステップに応じて。

> **注釈**
>
> *visual_pipeline_user* ロールには、ソーステーブルへのアクセスが必要です。オンデマンドモードでは、*SELECT* が必要です。ライブモードおよびバックグラウンドモードでは、 *visual_pipeline_user* にソーステーブルの所有者ロールを付与する必要があります。パイプラインの作成中に権限エラーが発生した場合は、必要な権限について *VPU and permissions* を参照してください。

1. パイプライン内の 2 つのノードを接続する矢印の **+** ボタンを選択します。
2. ドロップダウンからステップタイプを選択します。使用可能なステップタイプは、パイプライン内の位置と、先行するステップの出力タイプによって異なります。
   - **最初のステップ** ParsePDF、PerformOCR、PdfToImage、ParseHTML、ChunkText、SummarizeText、または KnowledgeBase それが唯一のステップの場合。
   - **中間ステップ** ParseHTML、 ChunkText、 SummarizeText 前述のステップでテキスト出力を生成する場合、または PerformOCR 前述のステップでバイナリイメージ出力を生成する場合
   - **最後のステップ**互換性のあるステップ。ベクターストレージが必要な場合、KnowledgeBase は最後のステップである必要があります。
3. ステップのパラメーターを構成します。
   - **ChunkText** 目的のチャンク長必須とオプションで最大長を設定します。
   - **SummarizeText** モデルピッカーからテキスト補完モデルを選択します。
   - **PerformOCR** モデルピッカーから OCR モデルを選択します。

   -** KnowledgeBase **オプションで既存の** ナレッジベーステーブル名**を選択して、既存のナレッジベースにデータを追加します :ref:`マルチパイプラインのナレッジベース <マルチパイプラインのナレッジベース>` を有効にします。フィールドを空白のままにして、新しいナレッジベースを作成します。** データフォーマット **テキストまたはイメージを選択します。
   - 新しいナレッジベースを作成するときに、** モデルサービス**埋め込みモデルを選択し、オプションで** 距離演算子 **とインデックスタイプ を構成します** .
   - 既存のナレッジベースに追加する場合、モデル、距離演算子、およびインデックスタイプは既存のナレッジベースから継承され、表示されません。
4. 最大 10 ステップまで繰り返してステップを追加します。

各中間ステップには、**Save Step Results** トグルデフォルトで有効になっており、ステップの出力を宛先テーブルに永続化し、各ステージで結果を検査できます。詳細は、データフローとステップ結果 を参照してください。

ステップ注文の管理

Pipeline Designer は、ステップの順序をリアルタイムで検証します。無効なステップ配置には視覚的にフラグが立てられます。ルールは次のとおりです。

- 生のドキュメント入力 ParsePDF、PdfToImage を使用するステップが最初のステップである必要があります。PerformOCR はバイナリ画像入力を受け入れ、画像出力 PdfToImage などを生成するステップの最初または後に現れることができます。
- KnowledgeBase は最後のステップである必要があります。
- 各ステップの出力タイプは、次のステップの予想される入力タイプと一致する必要があります。

パイプラインの作成中に、各ステップカードの左右の矢印コントロールを使用してステップを並べ替えたり、削除コントロールを選択してステップを削除したりできます。デプロイが完了すると、ステップ構造が修正されます。デプロイ後のパイプラインの管理 を参照してください。

モデルの選択

モデルを必要とするステップ KnowledgeBase、SummarizeText、PerformOCR には、ターゲットデータベースの AI Database AIDB モデルレジストリに登録されているすべてのモデルをリストするモデルピッカーが表示されます。ピッカーは、構成されるステップに基づいて、ファンクションタイプ埋め込み、テキスト補完、OCR によってフィルタリングします。ピッカーはモデル名のフラットなリストを提示し、ソース HM ホスト、外部、またはデータベースごとにモデルを分類しません。

すべての登録済みモデルがすべてのクラスターから到達できるわけではありません。HM ホストおよび HM プロキシモデルには、プライマリロケーションのモデル提供エンドポイントへのネットワーク接続が必要です。これは、プライマリロケーションクラスターでのみデフォルトで利用できます。セカンダリロケーションまたは自己管理インスタンスのクラスターは、*AIDB* ネイティブモデルの登録 SQL を介して直接登録、外部プロバイダーを呼び出します またはビルトインモデル AIDB に同梱され、ローカルで実行されますを使用する必要があります。デプロイメントタイプ別の完全なモデルの可用性の内訳については、モデルがパイプラインステップに到達する方法 を参照してください。

パイプラインの展開

1. すべてのステップが構成され、パイプラインが有効であることを確認します赤いインジケーターなし。
2. **宛先テーブル** 名はデフォルトの `pipeline_{your_pipeline_name}` です。KnowledgeBase ステップで終わるパイプラインの場合、これはベクターストレージテーブルになります。
3. **展開** を選択します。

パイプラインデザイナーは、パイプライン、関連するデータベースオブジェクト宛先テーブル、トリガー、またはバックグラウンド処理アーティファクトを作成し、そのステータスの監視を開始します。パイプラインは、「Creating」ステータスでパイプラインリストに表示されます。セットアップが完了すると、オペレーションステータスに移行します。

KnowledgeBase パイプラインの作成

ほとんどのパイプラインは、RAG 対応ベクターストアを構築するための標準パターンである KnowledgeBase ステップで終了します。このようなパイプラインを展開すると

1. AIDB は、エンベディングを保存するためのベクトルテーブルを作成します。
2. ナレッジベースエントリが AIDB のナレッジベースレジストリに登録されます。
3. ナレッジベースは、 *Langflow FAQ* エージェントなどのダウンストリームアプリケーションで使用できます。Pipeline Designer の **ナレッジベースと検索** ビューは、テストと検証のためのクエリインターフェイスを提供します。

ナレッジベースステップを使用してパイプラインを作成するには

1. 上記のウィザードに従ってください。
2. **ChunkText** ステップを追加して、ソーステキストをセグメントに分割します。
3. 最後のステップとして **KnowledgeBase** ステップを追加します。
4. KnowledgeBase ステップで、**Knowledge Base Table Name** を空白のままにして、新しいナレッジベースを作成するか、既存のナレッジベースを選択してそれに追加します。**テキスト埋め込みモデル** を選択し、データ形式を **テキスト**に設定します。
5. パイプラインを展開します。

KnowledgeBase の最終ステップのないパイプラインたとえば、テキストを宛先テーブルにチャンクまたは要約するだけのパイプラインなども可能ですが、一般的ではありません。

PDF または画像ソースの場合、ChunkText の前に ParsePDF または PerformOCR ステップを追加して、バイナリデータをテキストに変換します。

デプロイ後のパイプラインの管理

パイプラインが作成されたら

- **ステップ構造は修正されました。** **ステップは作成後に変更することはできません** を参照してください。
- **処理パラメーターは編集可能です。** パイプラインの詳細ページから **Actions > Edit Pipeline Options** を選択することにより、処理タイプ、バッチサイズ、および同期間隔を変更できます。
- **SQL を介して手動実行が利用できます。**処理モードに関係なく、パイプラインのデータベースで `SELECT aidb.run_pipeline('pipeline_name');` を呼び出して、手動実行をトリガーする必要があります。パイプラインデザイナーは、現在手動実行用の UI ボタンを提供していません。

Langflow との統合

Langflow FAQ は、EDB Postgres AI の AI エージェントのフロー構築インターフェイスです。 Pipeline Designer で作成されたナレッジベースは、これらのエージェントのデータソースとして機能できます。Langflow KB コンポーネントは、Pipeline Designer のビルトインクエリツールと同じベクトル取得メカニズムを使用して AIDB ナレッジベースを照会します。Langflow フローはユーザー設定のデータベース資格情報を使用するため、フローで構成された Postgres ロールは、ナレッジベースの基になるテーブルにアクセスできる必

要があります。権限の詳細については *Langflow でナレッジベースを使用する* を参照し、ビルディングエージェントフローの詳細については *Langflow documentation* を参照してください。

Knowledge bases

ナレッジベースは、最後のステップが KnowledgeBase であるパイプラインによって作成されるベクターストアです。ソースデータから生成されたエンベディングを保持し、取得拡張生成 RAG およびその他の AI ワークロードのセマンティック類似性検索をサポートします。

ナレッジベースリスト

Sovereign AI > Knowledge Bases に移動して、プロジェクト内のすべてのナレッジベースを表示します。各エントリは次を示します。

- **Name** パイプライン名から派生したナレッジベース識別子。
- **オリジン** ナレッジベースが存在するクラスターとデータベース。
- **Model** ベクトルを生成するために使用される埋め込みモデル。
- **ソースパイプライン**このナレッジベースにデータをフィードするパイプライン。ナレッジベースには、1 つ以上のソースパイプラインを持つことができます。3 つ以上のパイプラインが関連付けられている場合、リストには最初の 3 つがナレッジベースの詳細ページへの**+N more** リンクとともに表示されます。
- **ソースレコード** ソーステーブル全体の行の合計。
- **Result Records** 保存されたベクトル埋め込みの数。
- **未処理レコード** まだ処理されていないソース行。

ナレッジベースの詳細

ナレッジベースを選択して、その詳細ページを表示します。

レコード数

Metric	Description
Source records	ソーステーブルの合計行数
Embeddings	ナレッジベースに保存されているベクトル埋め込みの数
Unprocessed records	まだ処理されていないソース行

パイプラインが予想通りに動作している場合、未処理カウントはゼロバックグラウンドおよびライブモードの場合、ソースカウントからエンベディングを引いたものと同じですオンデマンドモードの場合。

ソースパイプライン

詳細ページには、このナレッジベースにデータをフィードするすべてのパイプラインがリストされています。ナレッジベースは、複数のパイプライン N:1 関係からデータを受信できるため、さまざまなソースおよび処理チェーンからのデータを単一の検索可能なナレッジベースに集約できます。

マルチパイプラインのナレッジベース

パイプラインデザイナーは、同じナレッジベースへの複数のパイプラインの書き込みをサポートしています。これにより、さまざまなソーステーブルからデータを集計したり、さまざまな処理チェーンたとえば、PDF ドキュメント用のパイプラインと HTML コンテンツ用のパイプラインなどを単一のナレッジベースに適用して、統合セマンティック検索ができます。

既存のナレッジベースにパイプラインを追加するには

1. *Creating pipelines* に従って新しいパイプラインを作成します。
2. 最後のステップとして **KnowledgeBase** ステップを追加します。
3. **Knowledge Base Table Name** フィールドで、追加する既存のナレッジベースを検索および選択します。
4. **データ形式**テキストまたはイメージを選択します。モデル、距離演算子、およびインデックスタイプは、既存のナレッジベースから継承されます。
5. パイプラインを展開します。

新しいパイプラインのデータは、既存のナレッジベースのベクターテーブルに追加されます。ナレッジベースリストと詳細ページの両方に、各ナレッジベースに関連付けられているすべてのソースパイプラインが表示されます。

テストの取得

ナレッジベースの詳細ページには、SQL を書かずにセマンティック検索をテストするためのビルトインクエリツールが含まれています。

テストクエリーの実行

1. ナレッジベースの詳細ページで、 **Run Test Query** を選択します。
2. **Perform Test Query Retrieval** ダイアログで、**Search Query** フィールドに検索テキストを入力します。
3. オプションで **結果の数** パラメーターを調整します。
4. **[Run Query]** をクリックします。

クエリツールは、ナレッジベースの埋め込みモデルを使用してテキストをエンコードし、ベクトル類似性検索を実行し、上位に一致するレコードとその距離スコアを結果ます。距離スコアが低いほど、より近い意味的一致を示します。結果は、クエリフィールドの下の同じダイアログに表示され、一致する各レコードは、ソースキーとその距離スコアを含む行として表示されます。

結果の解釈

各結果には以下が含まれます。

- **ソースキー**元のソース行のキー列値。結果をソースデータにトレースできます。
- **距離**ナレッジベースの構成済みの距離演算子デフォルトでは L2 を使用した、クエリベクトルと結果ベクトル間の類似距離。

テストクエリーは、パイプラインが意味のあるエンベディングを生成していること、およびチャンク戦略が適切な粒度のセグメントを生成することを検証するのに役立ちます。結果が悪く見える場合は、ChunkTextステップの目的の長さを調整するか、別の埋め込みモデルを試行することを検討します。

ナレッジベースのライフサイクル

ナレッジベースは、KnowledgeBase ターミナルステップを使用してパイプラインを展開すると、自動的に作成されます。パイプラインデザイナーを介して独立して作成することはできません。

ナレッジベースステップを持つパイプラインを削除すると、ナレッジベースは他のパイプラインがフィードしていない場合にのみ削除されます。他のパイプラインがナレッジベースを参照している場合、削除されたパイプラインの関連付けのみが削除されます。削除されたパイプラインがナレッジベースに関連付けられた最後のパイプラインである場合、ナレッジベースとそのベクターテーブルも削除され、保存されているすべてのエンベディングが完全に削除されます。

> **⚠ 警告**
>
> ナレッジベースの削除は元に戻せません。すべてのベクターデータは完全に削除されます。ナレッジベースが *Langflow FAQ* フローまたは他のコンシューマによって使用される場合、これらの統合は中断します。

Langflow でナレッジベースを使用する

パイプラインデザイナーを介して作成されたナレッジベースは、Langflow-based Gen AI builder からアクセスできます。 Langflow KB コンポーネントは、 Pipeline Designer クエリツールを強化するのと同じ **ナレッジベースと検索** ファンクションを使用して AIDB ナレッジベースに接続します。

Langflow でパイプラインデザイナーのナレッジベースを使用するには

1. 上記のように、パイプラインデザイナーを介してナレッジベースを作成および設定します。
2. Langflow で、KB コンポーネントをフローに追加します。
3. データベース接続の詳細とナレッジベース名を使用してコンポーネントを構成します。
4. コンポーネントは、フローの入力に基づいて、意味的に同様のコンテンツのナレッジベースを照会します。

障害対応

パイプラインの実行後は埋め込みなし

ソースレコード数が正の場合、埋め込みカウントはゼロのままである場合

1. パイプラインのステータスでエラーを確認します。「Has Errors」または「Failed」ステータスは、処理の問題を示します。
2. エンベディングモデルがパイプラインのクラスターから到達可能であることを確認します。ハイブリッドマネージャー HM ホスト対象および HM プロキシモデルには、プライマリロケーションクラスターからのみ到達可能です。パイプラインがセカンダリの場所または自己管理クラスターで実行

されている場合、これらのモデルは失敗します。モデルがパイプラインステップに到達する方法 を参照してください。

3. プライマリロケーションで HM ホストモデルを使用している場合、KServe InferenceService が正常であることを確認します。
4. ソースデータ列に、構成したパイプラインステップの有効なコンテンツが含まれていることを確認します。
5. AIDB エラーログで詳細なエラーメッセージを確認します。データベースに接続し、`SELECT * FROM aidb.get_error_logs('your_pipeline_name');` を実行します。

未処理数が減少しない

未処理の数が静的な場合

1. 処理モードがオンデマンドではないことを確認します。オンデマンドの場合、手動実行をトリガーするか、バックグラウンドまたはライブに切り替えます。
2. バックグラウンドモードの場合、最後のデータ変更から同期間隔が経過したことを確認します。
3. 処理を完全に停止するパイプラインブロッキングエラーがないかを確認します。

取得品質が低い

テストクエリーが無関係な結果を返した場合

1. チャンク戦略を確認します。大きすぎるチャンクまたは小さすぎるチャンクは、最適でないエンベディングを生成します。
2. 別の埋め込みモデルを検討します。一部のモデルは、特定のコンテンツタイプでパフォーマンスが向上します。
3. ソースデータがクリーンで関連性があることを確認します。HTML タグ、定型テキスト、または破損したコンテンツは、埋め込み品質を低下させます。チャンクする前に ParseHTML ステップを追加することを検討します。

External inference services

Pipeline Designer は、ハイブリッドマネージャー HM ホストモデルに加えて、外部 AI プロバイダー OpenAI、AWS Bedrock、NVIDIA NIM、およびその他の API 互換エンドポイントからのモデルを使用できます。2 つの異なるメカニズムにより外部モデルが利用可能になり、どちらが適用されるかは、クラスターが HM コントロールプレーンへの接続があるかどうかによって異なります。

配信メカニズムを理解することは、どのクラスターが特定のモデルを使用できるか、および資格情報を管理する方法を決定するため、重要です。

外部モデルが AIDB に到達する方法

Pipeline Designer を強化する Postgres 拡張機能である AI Database AIDB は、モデルレジストリを維持します。パイプラインステップは、このレジストリから名前でモデルを参照します。外部モデルは、2 つの独立したパスを介してレジストリに入ることができます。

HM 外部推論サービスプロキシ

ハイブリッドマネージャー UI ソブリン AI >モデルサービス利用可能な場合を介して外部推論サービスを登録すると、HM はプライマリ場所に管理対象プロキシを展開し、推論要求を外部プロバイダーの API エンドポイントに転送します。HM は、データベースに保存されないプロバイダーの API 資格情報を管理します。

AIDB の観点から、プロキシは HM ホストモデルと区別できません。AIDB は、それを自動的に検出し、同じ方法で推論要求をルーティングします。推論が最終的に外部プロバイダーに到達するという事実は、AIDB およびパイプラインデザイナーにとって透過的です。

このパスの主な特徴

- **HM コントロールプレーンによって管理されます。**登録、資格情報のストレージ、およびライフサイクル管理は、HM UI およびバックエンドサービスを介して発生します。
- **HM によって管理される資格情報。**プロバイダーの API キーまたは同等のものは、HM インフラストラクチャによって保存および注入されます。AIDB は、外部プロバイダーの資格情報を直接処理しません。
- **HM ホストモデルと同じ到達可能性制約。**プロキシはプライマリの場所で実行されるため、クラスターがこのパスを使用するには、プライマリへのネットワーク接続が必要です。外部プロバイダー自分自身はインターネットにアクセスできますが、AIDB からのトラフィックはプロバイダーに直接ではなく、プロキシを介して送信されます。デフォルトの展開では、セカンダリロケーションと自己管理クラスターにはこの接続がないため、代わりに AIDB ネイティブの登録を使用する必要があります。

HM への外部推論サービスの登録

1. Hybrid Manager プロジェクトの *Model Management on Hybrid Manager* に移動します。
2. **[外部サービスの登録]** を選択します。
3. 登録フォームに入力します。
    - **サービス名** このサービス接続の説明的な名前。
    - **プロバイダー** 外部プロバイダータイプたとえば、OpenAI、AWS Bedrock、NVIDIA NIM
    - **エンドポイント URL** 外部サービスの API エンドポイント。
    - **API プロトコルバージョン**プロバイダーと通信するときに使用するプロトコルバージョン。ほとんどのプロバイダーでは、デフォルト値が正しいです。プロバイダーが必要とする場合、または特定の API リビジョンにピン付けする必要がある場合にのみ、特定のバージョンを選択します。
    - **資格情報** プロバイダーの認証資格情報。これは通常、API キーです。AWS Bedrock の場合、これにはアクセスキーとシークレットが含まれる場合があります。
4. **Register** を選択します。

サービスが登録され、そのモデルは次のモデル同期サイクル通常数分以内に利用できるようになります。

HM での外部サービスの管理

ソブリン AI モデル提供インターフェイスから、次のことができます。

- **登録されたサービスの表示**ステータスやアクティビティログを含むすべての外部推論サービス接続を表示します。
- **サービスの編集**エンドポイント URL、資格情報、または API プロトコルバージョンを更新します。
- **サービスの登録を解除します**外部サービス接続を削除します。モデルは、次のモデル同期サイクル中にクリーンアップされます。

> **警告**
>
> 外部サービスの登録を解除すると、そのサービスのモデルを使用するパイプラインは、次回の実行で失敗します。登録を解除する前に、サービスのモデルに依存しているアクティブなパイプラインがないことを確認するか、別のモデルを使用するようにこれらのパイプラインを更新します。

AIDB ネイティブモデルの登録

AIDB には、HM とは無関係に動作する独自のモデル登録メカニズムがあります。`aidb.create_model()` SQL ファンクションは、外部プロバイダーの API エンドポイントを直接ポイントする、外部データラッパー FDW によってバッキングされたモデルエントリを作成します。パイプラインステップが AIDB ネイティブモデルを使用する場合、Postgres プロセスは中間プロキシを使用せずにプロバイダーに直接接続します。

このパスの主な特徴

- **SQL レベル、どこでも動作します。**AIDB 拡張機能を実行している Postgres クラスターは、HM インフラストラクチャが存在するか到達可能かどうかに関係なく、この方法でモデルを登録できます。
- **資格情報は FDW ユーザーマッピングとして保存されます。**API キーは `CREATE USER MAPPING FOR PUBLIC` を使用して保存され、すべてのデータベースユーザーがアクセスできます。資格情報のスコープは、個々のモデルではなく、プロバイダー外部サーバーに限定されます。同じプロバイダーを使用するすべてのモデルは、単一の資格情報のセットを共有します。
- **直接ネットワークアクセスが必要です。**Postgres プロセスは、外部プロバイダーの API エンドポイントに到達できる必要があります。プロキシや仲介者はいません。
- **HM UI 管理なし。**この方法で登録されたモデルは、HM モデル提供インターフェイスに表示されません。これらは完全に SQL によって管理されます。

`aidb.create_model()` 、プロバイダータイプ、および資格情報構成の詳細については、AIDB モデルレジストリのリファレンスドキュメントを参照してください。

クラスターに適切なメカニズムの選択

HM 外部推論プロキシは、プライマリの場所で実行されます。プライマリの場所の外部のクラスターは、プライマリのモデル提供エンドポイントへのネットワーク接続がある場合にのみ使用できます。デフォルトでは利用できません。Postgres プロセスはプロバイダーに直接接続するため、AIDB 拡張機能を備えたクラスターで機能します。

Deployment type	HM external inference proxy	AIDB-native registration
Primary location	Yes	Yes
Secondary location or self-managed	Not by default [1]	Yes

[1] プライマリロケーションのモデル提供エンドポイントへのネットワーク接続が必要です。これは、デフォルトの展開には存在しません。

完全なモデルの可用性の状況 HM ホスト型 KServe モデルおよびビルトイン AIDB モデルを含むについては、モデルがパイプラインステップに到達する方法 を参照してください。

パイプラインステップでの外部モデルの使用

どのメカニズムで登録されたかに関係なく、外部モデルは、HM ホストモデルおよびデータベース上のモデルとともにパイプラインデザイナーのモデルピッカーに表示されます。パイプラインデザイナーは登録パスを区別せず、構成されているステップと互換性のある AIDB モデルレジストリ内のすべてのモデルを表示します。

モデルを必要とするステップを構成する場合

- **KnowledgeBase ステップ**外部テキスト埋め込みモデル OpenAI 埋め込みモデルなどは、埋め込みモデルリストに表示されます。
- **SummarizeText ステップ**外部テキスト補完モデル OpenAI 補完モデルなどが言語モデルリストに表示されます。
- **OCR ステップの実行**プロバイダーが互換性のあるエンドポイントを提供している場合、外部 OCR モデルは OCR モデルリストに表示されます。

モデルピッカーはファンクションタイプでフィルタリングするため、構成するステップと互換性のあるモデルのみが表示されます。

モデルがサポートするファンクションタイプは、そのプロバイダーによって決定されます。HM UI を介して外部推論サービスを登録する場合、選択した **プロバイダ**たとえば、OpenAI または NVIDIA NIM によって、サービスのモデルがどのファンクションタイプを公開するかが決まります。同様に、モデルを `aidb.create_model()` に登録すると、プロバイダー引数 2 番目のパラメーターがファンクションタイプを制御します。たとえば、`openai_embeddings` プロバイダーに登録されたモデルはテキスト埋め込みをサポートしていますが、`nim_paddle_ocr` に登録されたモデルは OCR をサポートしています。ファンクションタイプを構成するための個別の手順はありません。彼らはプロバイダーからに従います。

資格情報の処理

外部サービスの資格情報は、登録パスに関係なく、すべてのデータベースユーザーで共有されます。HM 登録された外部サービスは、資格情報を Kubernetes シークレットプロキシモデルが使用し、同期後、データベース内の FDW ユーザーマッピングとして保存します。AIDB ネイティブモデルは、資格情報を FDW ユーザーマッピングとして直接保存します。どちらの場合も、ユーザーごとまたはパイプラインごとの資格情報の分離はありません。資格情報共有モデルとそのセキュリティへの影響の詳細については、共有モデル資格情報 を参照してください。

Executing pipelines

パイプライン実行は、ソースデータを読み取り、構成された処理ステップで実行し、宛先に結果を書き込むプロセスです。以下のセクションでは、実行をトリガーする方法、処理モードが実行動作に影響を与える方法、およびさまざまなクラスタータイプで実行がどのように動作するかについて説明します。

実行フロー

パイプラインが実行されると自動または手動でトリガーされ、次のシーケンスが発生します。

1. パイプライン操作は *VPU and permissions* ロールの下で実行されます。メカニズムは、パイプラインがトリガーされる方法によって異なります。
    - 手動実行およびパイプライン管理操作の場合、EDB Postgres AI エージェント `beacon-agent` はトランザクションを開き、`SET LOCAL ROLE` .
    - を使用して VPU に切り替えます。バックグラウンドモードの場合、AI データベース AIDB バックグラウンドワーカーパイプラインの所有者ロールとしてデータベースに直接接続します。
    - ライブモードの場合、VPU が所有するパイプラインごとの SECURITY DEFINER ラッパーファンクションは、呼び出し元のセッションのロールに関係なく、VPU の特権で実行されます。
2. AIDB は、ソーステーブルから保留中の行を読み取ります。
3. 各ステップはデータをシークエンシャルに処理します。1 つのステップの出力は、次のステップの入力になります。
4. いずれかのステップでレコードレベルのエラーたとえば、不正な PDF が発生した場合、エラーはログに記録され、パイプラインは残りのレコードで続行されます。パイプラインレベルのエラーたとえば、モデル到達不能などにより、実行が停止されます。
5. 最終出力は、宛先テーブル、または最後のステップが KnowledgeBase の場合はナレッジベースベクターテーブルに書き込まれます。

パイプラインの手動実行

処理モードに関係なく、パイプラインの実行を手動でトリガーできます。

1. パイプラインのデータベースに接続します。
2. `SELECT aidb.run_pipeline('your_pipeline_name');` を実行します。
3. パイプラインは、保留中のソースレコードをすべて処理します。

> **注釈**
>
> Pipeline Designer は、現在手動実行用の UI ボタンを提供していません。上記の SQL ファンクションを使用して、オンデマンドでパイプラインの実行をトリガーします。

手動実行は次の場合に役立ちます。

- 自動処理を有効にする前に、新しく作成したパイプラインのテスト。
- 以前のオンデマンドパイプラインを再度有効にした後のレコードのバックログの処理。

- 自動処理が必要ないワンタイムバッチ操作。

自動実行モード

ライブモード

ライブモードでは、 AIDB は、すべての `INSERT` 、`UPDATE` 、または `DELETE` で起動するトリガーをソーステーブルにインストールします。パイプラインはトリガートランザクション内の影響を受ける行を処理するため、ソーステーブルに書き込むアプリケーションは、書き込みが返される前にパイプライン処理の完全なレイテンシーを発生します。

特徴

- ソーステーブルへの書き込みは、パイプラインが影響を受ける行の処理を終了するまでブロックされます。アプリケーションは、書き込みごとにパイプライン処理のコストを支払います。
- 遅い推論モデル呼び出したとえば、埋め込みまたは要約は、アプリケーションの書き込みパスにレイテンシを直接追加します。
- 書き込み直後にデータを埋め込むか、処理する必要がある、少量のレイテンシーの影響を受けるシナリオに適しています。

バックグラウンドモード

バックグラウンドモードでは、AIDB のバックグラウンドワーカー Postgres インスタンス内で実行されており、構成可能なスケジュールで未処理の行をバッチで処理します。アプリケーショントランザクションはすぐに完了します。パイプライン処理は独立して発生します。

特徴

- ソーステーブルへのアプリケーションの書き込みは、パイプライン処理の影響を受けません。結果が表示されるまでに遅延同期間隔によって決まります。
- バッチサイズと同期間隔は構成可能ですデフォルトバッチごとに 500 行、15 時間間隔。
- ほとんどの運用ワークロードに推奨されます。
- バックグラウンドワーカーは、HM コントロールプレーンではなく、データベースクラスターで実行されます。Postgres DistributedPGD クラスターでは、書き込みリーダーノードで実行されます。

展開タイプ

Pipeline Designer は、3 つのハイブリッドマネージャー HM 展開タイプすべてのクラスターでパイプラインを作成および管理できます。使用可能なモデルアクセスパスが決定されるため、これらのカテゴリを理解することは役立ちます。完全な内訳については、以下の **モデルがパイプラインステップに到達する方法** を参照してください。

HM **ロケーション**は、HM コントロールプレーンによって管理される Kubernetes 環境です。HM は、ハブアンドスポークトポロジで複数のロケーションをサポートします。

- **プライマリロケーション**コントロールプレーン、KServe モデル提供、外部推論サービスプロキシ、およびすべてのソブリン AI サービスを含む完全なプラットフォームスタックが実行される最初の HM 管理環境。プライマリロケーションのクラスターには、すべての機能への完全なアクセスがあります。

- **セカンダリの場所。**マルチデータセンター展開における追加の HM 管理環境。セカンダリロケーションは core インストールシナリオを実行します。これには、データベースクラスターと重要なサービスが含まれますが、AI シナリオ KServe、モデルサービングは含まれません。これらは、プライマリの場所へのコントロールプレーンの接続を維持します。
- **自己管理クラスター。**顧客管理のシングルインスタンスの Postgres サーバーは、EDB Postgres AI エージェント beacon-agent を介して HM に登録されています。外部 PGD クラスターと Cloud Native PostgresCNP クラスターはサポートされていません。自己管理クラスターは HM によってプロビジョニングされず、デフォルトではプライマリの場所へのネットワーク接続がありません。その接続がなければ、AIDB ネイティブのビルトインモデルのみが使用できます。詳細は、モデルがパイプラインステップに到達する方法 を参照してください。

サポートされているクラスター

Pipeline Designer は、HM 管理のクラスタープライマリとセカンダリの両方の場所と自己管理の Postgres インスタンスでパイプラインを管理します。パイプラインを作成するときは、展開タイプに関係なく、エステートドロップダウンからターゲットクラスターを選択します。Pipeline Designer は、各クラスター上の EDB Postgres AI エージェントを介してデータベース、スキーマ、テーブル、および使用可能なモデルを検出します。

自己管理クラスターは、追加の前提条件を満たす必要があります。 Postgres 16 以降、AIDB 拡張機能のインストールと構成、および HM コントロールプレーンに接続された EDB Postgres AI エージェントが必要です。

モデルがパイプラインステップに到達する方法

推論を実行するパイプラインステップ KnowledgeBase、SummarizeText、PerformOCR は、ターゲットデータベースの AIDB モデルレジストリから名前でモデルを参照します。パイプラインデザイナーは、パイプラインレベルでモデルソースを区別しません。モデルピッカーには、ファンクションタイプが構成するステップと一致する登録済みモデルがすべて表示されます。ただし、モデルは 3 つの異なるパスを介してレジストリに入力され、それぞれのパスは、パイプラインのクラスターが展開トポロジ内のどこにあるかに応じて、異なる可用性特性を持っています。

HM ホスト型 KServe モデルパス 1

HM の Model Serving を介して展開されたモデルは、プライマリロケーションの Kubernetes クラスターで実行されます。AIDB は、これらのモデルを自動的に検出および登録し、パイプラインステップを作成するときに使用できるようにします。パイプラインステップがこのようなモデルを呼び出すと、AIDB は推論要求をプライマリロケーションにあるモデルのサービングエンドポイントにルーティングします。

可用性プライマリ場所ではデフォルトで利用できます。セカンダリロケーションまたは自己管理インスタンスのクラスターは、プライマリロケーションのモデル提供エンドポイントへのネットワーク接続がある場合にのみこれらのモデルを使用できます。この接続は、デフォルトの展開には存在しません。

HM 外部推論サービスプロキシパス 2

HM UI を介して外部プロバイダー OpenAI、AWS Bedrock、NVIDIA NIM を登録すると、HM はプライマリ場所に管理対象プロキシを展開し、推論要求を外部プロバイダーに転送し、ユーザーに代わって資格情報管理を処理します。AIDB は、ローカルにホストされるモデルパス 1 の場合とまったく同様に、このプロキシを検出してアドレス指定します。

可用性パス 1 と同じ制約。プロキシはプライマリの場所で実行されるため、クラスターが使用するにはプライマリへのネットワーク接続が必要です。外部プロバイダー自分自身はインターネットにアクセスできますが、AIDB のトラフィックはプロバイダーに直接ではなく、プロキシを介して送信されます。登録と管理の詳細については、*HM* **外部推論サービスプロキシパス** *2* を参照してください。

AIDB ネイティブモデル登録パス 3

`aidb.create_model()` SQL ファンクションは、プロバイダーの API エンドポイントを指す外部データラッパー FDW によってバッキングされ、AIDB モデルレジストリにモデルを直接登録します。パイプラインステップが AIDB ネイティブモデルを呼び出すと、Postgres プロセスは中間プロキシを使用せずにプロバイダーに直接接続します。資格情報は FDW ユーザーマッピング `CREATE USER MAPPING FOR PUBLIC` として保存されます。

可用性 HM 接続に関係なく、AIDB 拡張機能を実行しているクラスター。これは、プライマリの場所に到達できないクラスターで使用可能な唯一のモデルアクセスパスです。プロバイダー、構成、および資格情報管理の詳細については、

External inference services: AIDB-native model registration および *AIDB model registry reference documentation* を参照してください。

展開タイプ別のモデルの可用性

Deployment type	HM-hosted KServe (path 1)	HM external inference proxy (path 2)	AIDB-native registration (path 3)
Primary location	Yes	Yes	Yes
Secondary location (or self-managed with HM network)	Not by default [1]	Not by default [1]	Yes
Self-managed without HM network	Not by default [1]	Not by default [1]	Yes

[1] プライマリロケーションのモデル提供エンドポイントへのネットワーク接続が必要です。これは、デフォルトの展開には存在しません。

パス 1 および 2 では、プライマリロケーションのモデル提供エンドポイントへのネットワーク接続が必要です。これは、セカンダリロケーションまたは自己管理クラスターにデフォルトで存在しません。Postgres プロセスはプロバイダーエンドポイントに直接接続するため、パス 3 はどこでも機能します。

ビルトインモデル

AIDB には、Postgres プロセス内でローカルに実行され、ネットワークアクセスを必要としないいくつかのビルトインモデル bert、clip、t5、llama も同梱されています。これらは、AIDB 拡張機能がインストールされているすべてのクラスターで利用できます。

レプリケーションとフェイルオーバー

パイプラインの実行動作は、展開タイプではなくクラスターアーキテクチャに依存します。クラスターがプライマリロケーション、セカンダリロケーション、または自己管理にあっても、同じメカニズムが適用されます。

- **PGD クラスター**は、書き込みリーダーノードでパイプラインを実行します。パイプラインのメタデータ構成、状態、メトリックは、BDR 双方向レプリケーション論理レプリケーションを介してすべてのノードにレプリケートされますが、実行は書き込みリーダーでのみ発生します。書き込みリーダーがスイッチオーバーすると、新しいリーダーが実行を再開します。バックグラウンドモードでは、AIDB のバックグラウンドワーカーが新しいリーダーで起動し、未処理の行を選択します。ライブモードでは、トリガー機能がすべてのノードに複製され、新しい書き込みリーダーで自動的に起動されます。新しいリーダーが選出される間のスイッチオーバー中に、短いギャップが発生する場合があります。
- **プライマリ/スタンバイレプリケーション PSR クラスター**は、単一の書き込み可能なノードでパイプラインを実行します。昇格したスタンバイへのフェールオーバーでは、新しいプライマリが操作可能になり、AIDB のバックグラウンドワーカーが起動すると、バックグラウンド処理が再開されます。ライブモードトリガーは昇格したスタンバイに存在し、通常に起動します。フェールオーバーウィンドウがある場合に書き込まれた行は、通常の未処理行バックログの一部として処理されます。

モニタリングの実行

パイプラインの実行ステータスは、パイプラインリストと詳細ページに表示されます。

- **ステータスインジケーター**パイプラインが最新、処理中、古い、エラーがある、または失敗したかどうかを示します。
- **レコード数**ソースレコード、宛先レコードまたは KB パイプラインの埋め込み、および未処理のレコード。

エラー状態のパイプラインには、リストビューに Configuration Error バッジが表示されます。

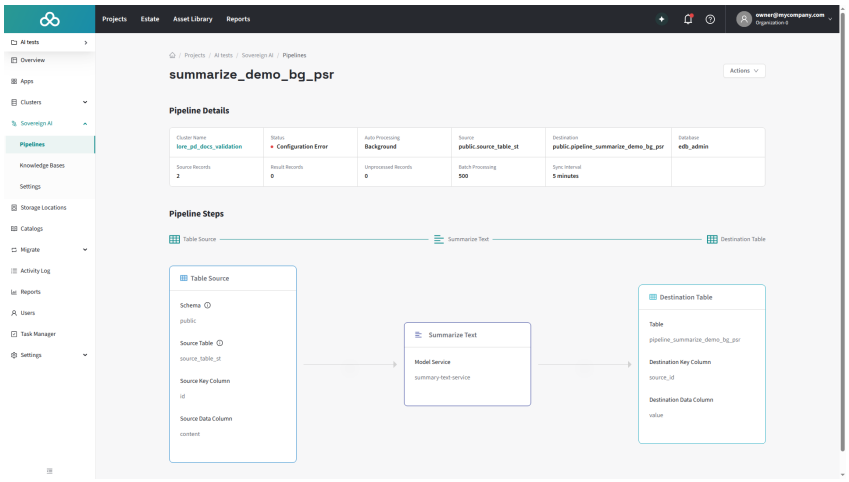


図 1: Pipeline list entry showing a Configuration Error status badge

- **エラーログ**詳細なエラー情報については、AIDB エラーログを直接照会します。

```
SELECT * FROM aidb.get_error_logs(your_pipeline_name);
```

エラーログクエリーとエラーカテゴリの詳細については、AIDB error log documentation を参照してください。

エラーは次のように分類されます。

- **RecordTemporary / RecordPermanent** 個々のレコードに影響を与えるエラー。パイプラインは他のレコードの処理を継続します。
- **PipelineTemporary / PipelinePermanent** パイプライン全体に影響を与えるエラー。問題が解決されるまで、実行は停止します。

VPU and permissions

パイプラインデザイナーは、 `visual_pipeline_user` VPU と呼ばれる専用の Postgres ロールを使用して、AI データベース AIDB 拡張機能に対してすべてのパイプライン操作を実行します。VPU の動作を理解することは、パイプラインデザイナーに表示されるパイプライン、パイプラインがアクセスできるテーブル、および権限の構成に直接影響を与えるため、重要です。

VPU とは何ですか?

`visual_pipeline_user` は、パイプラインデザイナーを使用する前に前提条件として手動で作成する Postgres ロールです。ロールは自動的に作成されません。設定手順については、*Creating VPU role* を参照してください。ロールは、作成されると基本的な接続を持っていますが、データの読み取りまたはパイプラインオブジェクトの作成に必要な権限がありません。これらの権限は自分自身で構成する必要があります *VPU* **権限の構成** を参照してください。パイプライン管理操作パイプラインの作成、更新、および削除は、このロールの下で実行されます。エージェントはトランザクションを開き、`SET LOCAL ROLE visual_pipeline_user` を使用して ID を切り替えます。パイプラインの実行も VPU として実行されますが、メカニズムは処理モードによって異なります。詳細については、**実行フロー** を参照してください。

VPU は 2 つの理由から存在します。

1. **特権の分離。** パイプライン SQL は、スーパーユーザーまたは管理者権限では実行されません。パイプライン操作が AIDB 拡張機能で予期しない動作をトリガーした場合、爆発半径は VPU がアクセスを許可されている範囲に制限されます。
2. **パイプラインデザイナーの可視性制御。** Pipeline Designer には、VPU が所有するパイプラインとナレッジベースのみが表示されます。この分離により、パイプラインデザイナーを介して管理されるパイプラインが、他の手段 SQL、スクリプト、サードパーティツールなどで作成されたパイプラインとは区別されます。Pipeline Designer は、自分が作成していないパイプラインを管理しようとせず、互換性のない構成との潜在的な競合を回避します。

環境セットアップ手順拡張機能のインストール、クラスター構成、プラットフォームロール、およびソーステーブルの付与については、**セットアップの前提条件** を参照してください。

VPU が可視性に与える影響

パイプラインデザイナーを介してパイプラインが作成されると、AIDB はパイプラインの `owner_role` を `visual_pipeline_user` としてスタンプします。 EDB Postgres AI エージェントは、AIDB からパイプラインデータとナレッジベースデータを定期的に収集し、`owner_role = 'visual_pipeline_user'` 用にフィルタリングします。一致するオブジェクトのみがハイブリッドマネージャー HM コントロールプレーンに報告され、パイプラインデザイナーに表示されます。

これは直接的な影響があります。パイプラインデザイナーの外部で作成されたパイプラインとナレッジベースは、パイプラインデザイナーに表示されません。SQL を介してパイプラインを作成する場合たとえば、edb_admin ロールとして psql を使用すると、データベースレベルで正しく機能しますが、パイプラインデザイナーのパイプラインリストまたはナレッジベースビューには表示されません。

作成後にパイプラインの owner_role を変更する AIDB ファンクションはありません。SQL で作成したパイプラインをポータルに表示する必要がある場合は、VPU ロールを引き受けているときに作成する必要があります。ロールのスイッチを実行するには、Postgres ロールに VPU GRANT visual_pipeline_user TO your_role が付与されている必要があります。

```
- - SET LOCAL ROLE scopes the identity change to the current transaction.
BEGIN;
SET LOCAL ROLE visual_pipeline_user;
SELECT aidb.create_pipeline(...);
COMMIT;
```

注釈

この方法で作成されたすべてのパイプラインには、どのユーザーが `SET LOCAL ROLE` を実行したかに関係なく、*owner_role = 'visual_pipeline_user'* でスタンプされます。パイプラインのメタデータには、ユーザーごとの属性はありません。 *SET ROLE* セッションスコープではなく `SET LOCAL ROLE` トランザクションスコープを使用して、後続の SQL が誤って VPU として実行されるのを防ぎます。

VPU 権限の構成

VPU は、データテーブルに自動的にアクセスしたり、パイプラインオブジェクトを作成したりする機能を持っていません。以下の概要で説明するように、VPU が必要とする権限は、パイプラインの処理モードによって異なります。必要な権限が不足している場合、パイプラインの作成または実行は権限エラーで失敗します。

必要な VPU の権限

1. **宛先テーブルを作成します。**AIDB は、ソーステーブルと同じスキーマに宛先テーブル pipeline_<name> という名前を作成します。VPU には、そのスキーマに CREATE ON SCHEMA が必要です。この許可はすべての処理モードに必要です。
2. **ソースデータを読み取り、処理トリガーを作成します。** Grant VPU がソーステーブルを読み取るニーズは、処理モードによって異なります。

- **オンデマンドモード**はソーステーブルを読み取りますが、トリガーは接続しません。VPU は、ソーステーブルで SELECT のみが必要です。

```
GRANT SELECT ON <schema>.<source_table> TO visual_pipeline_user;
```

- **ライブモードとバックグラウンドモード**は、新しい行または更新された行が自動的に処理されるように、ソーステーブルにトリガーをアタッチします。これらのモードの場合、VPU にソーステーブルの所有者ロールのメンバーシップを付与する必要があります。

```
GRANT <source_table_owner_role> TO visual_pipeline_user;
```

この単一の付与により、VPU に読み取りアクセスと、ソーステーブルの処理トリガーを作成および削除するために必要な特権の両方が付与されます。<source_table_owner_role> を、ソーステーブルを所有するロールに置き換えます。

Postgres ではテーブルの所有者のみがトリガーをドロップできるため、より狭い TRIGGER 付与ではなく、所有者ロールのメンバーシップが必要です。 TRIGGER 特権では、トリガーの作成はできますが、削除はできないため、VPU は、パイプラインが更新または削除されたときにトリガーをクリーンアップできません。

以下のサブセクションでは、トポロジごとの正確な GRANT ステートメントを示しています。

パブリックスキーマのテーブル

Postgres 15 以降では、public スキーマの CREATE は、デフォルトで PUBLIC に付与されません。したがって、VPU には、クラスターが管理対象または自己管理対象であるかに関係なく、明示的な CREATE ON SCHEMA public 付与が必要です。 aidb_users ロールは、AIDB 拡張オブジェクトへのアクセスを付与しますが、ユーザー作成テーブルの SELECT 、スキーマレベルの CREATE 、または処理トリガーの作成および削除に必要な特権はカバーしていません。

管理者としてデータベースに接続し、処理モードの許可を実行します。

オンデマンドモードの場合

```
GRANT SELECT ON public.<source_table> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA public TO visual_pipeline_user;
```

ライブモードとバックグラウンドモードの場合

```
GRANT <source_table_owner_role> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA public TO visual_pipeline_user;
```

<source_table> を実際のテーブル名に置き換え、<source_table_owner_role> をソーステーブルを所有するロールに置き換えます。

ユーザー作成スキーマのテーブル

カスタムスキーマのテーブルの場合、VPU アクセスを明示的に付与する必要があります。管理者としてデータベースに接続し、処理モードの許可を実行します。

オンデマンドモードの場合

```
GRANT USAGE ON SCHEMA <schema> TO visual_pipeline_user;
GRANT SELECT ON <schema>.<source_table> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA <schema> TO visual_pipeline_user;
```

ライブモードとバックグラウンドモードの場合

```
GRANT USAGE ON SCHEMA <schema> TO visual_pipeline_user;
GRANT <source_table_owner_role> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA <schema> TO visual_pipeline_user;
```

<schema> および<source_table> を実際のスキーマとテーブルの名前に置き換え、<source_table_owner_role> をソーステーブルを所有するロールに置き換えます。 USAGE により、VPU はスキーマ内のオブジェクトを認識できます。 CREATE を使用すると、そこに宛先テーブルを作成できます。

重要

ライブモードおよびバックグラウンドモードの場合、ソーステーブル`GRANT <source_table_owner_role> TO visual_pipeline_user` を所有するロールのみを付与します。広範な管理ロールで VPU メンバーシップを付与しないでください。一般的だが広範すぎるショートカットは `GRANT edb_admin TO visual_pipeline_user` で、これは効果的にデータベース全体への VPU 管理者レベルのアクセスを付与し、VPU が提供するように設計されている特権分離を無効にします。

PGD クラスターのテーブル

Postgres DistributedPGD クラスターは、他のトポロジと同じ許可に加えて、追加の BDR 双方向レプリケーションの考慮事項が必要です。すべての許可は、書き込みリーダーだけでなく、すべての BDR ノードで一貫している必要があります。書き込みリーダーには存在するが、サブスクライバノードにない付与は、そのノードでトリガーが起動されたときにレプリケーション障害を発生させます。

すべての BDR ノードにベースグラントを適用するか、レプリケートされた DDL メカニズムを使用します。

オンデマンドモードの場合

```
- - Run on each BDR node:
GRANT USAGE ON SCHEMA <schema> TO visual_pipeline_user;
GRANT SELECT ON <schema>.<source_table> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA <schema> TO visual_pipeline_user;
```

AIDB 7.3.2 以降を実行している PGD クラスターのライブおよびバックグラウンドモードの場合

```
- - Run on each BDR node:
GRANT USAGE ON SCHEMA <schema> TO visual_pipeline_user;
GRANT <source_table_owner_role> TO visual_pipeline_user;
GRANT CREATE ON SCHEMA <schema> TO visual_pipeline_user;
```

7. 3.2 より前の AIDB を実行している PGD クラスターのライブおよびバックグラウンドモードの場合、所有者ロールを付与する代わりに、ソーステーブルの所有権を VPU に転送します。

```
- - Run on each BDR node:
GRANT USAGE ON SCHEMA <schema> TO visual_pipeline_user;
ALTER TABLE <schema>.<source_table> OWNER TO visual_pipeline_user;
GRANT CREATE ON SCHEMA <schema> TO visual_pipeline_user;
```

<schema> および<source_table> を実際のスキーマとテーブルの名前に置き換え、<source_table_owner_role> をソーステーブルを所有するロールに置き換えます。 public のテーブルの場合、USAGE 付与を省略します。

PGD クラスターでのトリガーハンドラーの所有権

ライブまたはバックグラウンド処理を有効にすると、AIDB はパイプラインごとに SECURITY DEFINER ラッパーファンクションを作成し、その所有者をパイプラインの owner_role ポータルで作成したパイプラインの場合は visual_pipeline_user に設定します。BDR は、レプリケートテーブルにアタッチされたトリガーファンクションに所有権制約を適用します。トリガーファンクションの所有者がすべてのノード上のリレーションのレプリケーション制約を満たすことを確認します。

AIDB 7.3.2 以降では、ソーステーブルの所有者ロール GRANT <source_table_owner_role> TO visual_pipeline_user を VPU に付与すると、これらの制約が満たされます。7.3.2 より前の AIDB バージョンでは、所有者ロールを継承することによりソーステーブルの所有権を VPU に付与するだけでは、十分ではありません。ソーステーブルの VPU 所有者を明示的に作成する必要があります。

```
ALTER TABLE <schema>.<source_table> OWNER TO visual_pipeline_user;
```

これは、VPU をソーステーブルの所有者にするため、所有者ロールの付与よりも広範です。7.3.2 より前の AIDB バージョンで PGD 固有の回避策としてのみ使用します。基本的なコンストレインの詳細については、*PGD* クラスターでのトリガーファンクションの所有権 を参照してください。

セキュリティと分離の考慮事項

前述のセクションでは、VPU のセットアップと権限の付与について説明しました。以下のトピックでは、セキュリティ、資格情報の分離、およびオブジェクトの所有権に関する VPU の共有 ID モデルのアーキテクチャへの影響について説明します。

共有 ID

パイプラインデザイナーで作成されたすべてのパイプラインは、同じ Postgres ロールとして実行されます。データベースレベルでの HM ユーザーごとの分離はありません。HM ユーザー A のパイプラインと HM ユーザー B のパイプラインは、両方とも visual_pipeline_user として実行され、同じテーブルのセットにアクセスします。ユーザーのパイプライン間の分離は、Postgres レベルではなく、HM プロジェクトレベルでパイプラインデザイナーのアクセス制御を介して管理されます。

これは、Postgres レベルでユーザーごとの監査証跡がないことも意味します。データベースログのすべてのパイプラインアクティビティは、visual_pipeline_user による操作として表示されます。

共有モデル資格情報

モデル資格情報外部推論サービスの API キーは FDW ユーザーマッピングとして保存され、すべてのデータベースユーザーで共有されます。同じデータベース上のパイプラインは、どの HM ユーザーが作成したかに関係なく、登録されたモデルを使用できます。パイプラインごとまたはユーザーごとの資格情報の分離はありません。

VPU によって作成されたオブジェクト

パイプラインデザイナーがパイプラインを作成すると、VPU はいくつかのデータベースオブジェクトの所有者になります。

- パイプラインの宛先テーブルたとえば `pipeline_my_pipeline` 。
- ナレッジベースベクターテーブルパイプラインに KnowledgeBase ステップが含まれる場合。
- パイプラインごとの SECURITY DEFINER ラッパーファンクションライブおよびバックグラウンド処理モードの場合。これらのラッパーは、実際の AIDB トリガーハンドラーを呼び出し、VPU によって所有されるため、呼び出したロールに関係なく VPU の特権で実行されます。
- AIDB によって内部的に使用されるパイプライン状態追跡テーブル。

これらのオブジェクトは、パイプラインデザイナーを介してパイプラインが削除されるまで持続します。

一般的な VPU の問題

パイプラインの作成中に「Permission denied for table」

原因 VPU には、ソーステーブルまたはそのスキーマに必要な権限が 1 つ以上ありません。オンデマンドモードの場合、これは `SELECT` または `CREATE ON SCHEMA` です。ライブモードおよびバックグラウンドモードの場合、ソーステーブルの所有者ロール `GRANT <source_table_owner_role> TO visual_pipeline_user` または `CREATE ON SCHEMA` のメンバーシップです。

修正 *VPU* **権限の構成** の説明に従って、必要な権限を付与します。

SQL を介して作成されたパイプラインはパイプラインデザイナーに表示されません

原因 パイプラインは `visual_pipeline_user` 以外のロールで作成されたため、その `owner_role` はパイプラインデザイナーのフィルターと一致しません。

修正 `aidb.create_pipeline()` を呼び出す前に、`SET LOCAL ROLE visual_pipeline_user` を使用してパイプラインを再作成します。完全な手順については、*VPU* **が可視性に与える影響** を参照してください。

パイプラインは動作しますが、ナレッジベースが表示されません

原因 上記と同じ。ナレッジベースは、パイプラインから `owner_role` を継承します。パイプラインが VPU で作成されていない場合、パイプラインも KB もパイプラインデザイナーに表示されません。

PGD クラスターでトリガーの作成が失敗する

原因 AIDB がライブまたはバックグラウンド処理を有効にすると、VPU が所有するパイプラインごとの SECURITY DEFINER ラッパーファンクションを作成します。BDR、トリガーファンクションの所有者がソーステーブルのレプリケーション制約を満たす必要があります。VPU の BDR ノードで必要な許可が不足している場合、トリガーの作成は失敗します。

修正 *PGD* **クラスターのテーブル** の説明に従って、すべての PGD ノードに一貫した `GRANT` ステートメントを適用します。7.3.2 より前の AIDB バージョンでは、所有者ロールの付与では十分ではありません。ソーステーブルの所有権を VPU に転送するための *PGD* **クラスターでのトリガーハンドラーの所有権** フォールバックを参照してください。

Limitations

以下のセクションでは、現在のパイプラインデザイナー実装の意図的なデザイン境界と利用可能な回避策に関する既知の問題について説明します。リリース固有の詳細については、*Hybrid Manager release notes* を参照してください。

パイプラインの作成と構造

パイプラインごとに最大 10 ステップ

パイプラインは、最大 10 の処理ステップをサポートします。

ステップは作成後に変更することはできません

パイプラインが展開されると、そのステップ構造ステップタイプ、順序、構成が固定されます。ステップを追加、削除、または並べ替えることはできません。パイプラインの構造を変更するには、パイプラインを削除して、新しいものを作成します。

パイプライン名の長さ制限

パイプライン名は 46 文字までに制限されています。AIDB は、内部オブジェクト名サフィックスステートテーブル、トリガー、エラーログの Postgres 識別子制限の最大 17 文字を予約しています。

宛先テーブル名の衝突

`pipeline_{your_pipeline_name}` という名前のテーブルがターゲットスキーマに既に存在し、登録されたナレッジベースベクターテーブルでない場合、パイプラインの作成は失敗します。既存のテーブルの名前を変更するか、別のパイプライン名を選択します。

外部 PGD および CNP クラスターはサポートされていません

自己管理クラスターのサポートは、単一インスタンスの Postgres サーバーに制限されています。外部 PGD Postgres Distributed クラスターとクラウドネイティブ PostgresCNP オペレーター管理クラスターは、EDB Postgres AI エージェントを介してハイブリッドマネージャー HM に登録されている場合でも、パイプラインターゲットとしての資格はありません。

マルチステップパイプライン

ChunkText のオーバーラップ設定は使用できません

ChunkText ステップの `overlap_length` パラメーター連続するチャンク間でテキストが重なる量を制御しますは、AI データベース AIDB でサポートされていますが、パイプラインデザイナーではまだ構成できません。チャンクのオーバーラップが必要な場合は、SQL を介してパイプラインを直接作成します。パラメーターの詳細については、ChunkText step reference を参照してください。

リモートクラスターでの推論ルーティング

HM ホストおよび HM プロキシモデルには、プライマリロケーションの接続が必要です

HM ホスト型 KServe モデルと HM 外部推論サービスプロキシは、両方ともプライマリ場所で実行されます。セカンダリロケーションまたは自己管理インスタンスのクラスターは、デフォルトではプライマリロケーションのモデル提供エンドポイントへのネットワーク接続がありません。これらの 2 つのパスのモデルは AIDB モデルレジストリに登録されているためモデルピッカーに表示されますが、プライマリの場所に到達できないクラスターでパイプラインが実行されている場合、実行時に失敗します。

解決策プライマリロケーションの接続のないクラスター上のパイプラインの場合、`aidb.create_model()` Postgres プロセスからプロバイダーに直接連絡するを介して登録された AIDB ネイティブモデルまたはローカルで実行されるビルトイン AIDB モデルを使用します。完全なモデルの可用性の内訳については、**モデルがパイプラインステップに到達する方法** を参照してください。

処理モード

ライブモードはアプリケーションの書き込みをブロックします

ライブ処理は、トリガーとなるトランザクション内で実行されます。ソーステーブルの `INSERT` 、`UPDATE` 、または `DELETE` は、パイプラインが影響を受ける行の処理を終了するまで返されません。推論モデル呼び出し SummarizeText、KnowledgeBase 埋め込みを含むステップの場合、これにより、アプリケーションの書き込みパスに大幅な遅延が追加される可能性があります。

推奨リアルタイム処理が厳密に必要な場合を除き、運用ワークロードにはバックグラウンドモードを使用します。

バックグラウンドワーカーのオーバーヘッド

バックグラウンド処理ワーカーは、クラスター上のすべてのデータベースをポーリングして、バックグラウンドモードのパイプラインを見つけます。多くのデータベースまたは多くのパイプラインを使用する展開では、このポーリングにより測定可能なオーバーヘッドが発生する可能性があります。

推奨ニアリアルタイム処理が必要とされないパイプラインの `background_sync_interval` を増やして、ポーリング頻度を減らします。構成の詳細については、AIDB background worker documentation を参照してください。

ナレッジベース

スタンドアロンのナレッジベースの作成なし

ナレッジベースは、KnowledgeBase ターミナルステップを使用してパイプラインを展開する副作用としてのみ作成できます。パイプラインデザイナーには、スタンドアロンの「ナレッジベースの作成」操作がありません。

KnowledgeBase のステップだけで有用な結果を得るには、あまり十分ではありません。生のテキストデータは、コンテンツを埋め込み用に適切なサイズのセグメントに分割するために、通常、KnowledgeBase ステップの前に少なくとも 1 つの ChunkText ステップが必要です。KnowledgeBase ステップのみを含むパイプラインは、ソース行全体を単一のベクトルとして埋め込もうとしますが、モデルトークンの制限を超えたり、取得品質が低下したりする可能性があります。

モデルとアクセス制御

ユーザーごとのモデル分離なし

モデル資格情報は、クラスター上のすべてのデータベースユーザーで共有されます。ユーザーごとまたはパイプラインごとの資格情報の分離はありません。詳細は、**共有モデル資格情報** を参照してください。

既知の問題

PGD クラスターでのトリガーファンクションの所有権

7. 3.2 より前の AIDB では、`visual_pipeline_user` は、ロール継承を介して所有権を取得するのではなく、トリガーハンドラーファンクションを直接所有する必要があります。直接の所有権がないと、AIDB トリガーハンドラーファンクション `aidb.pipeline_background_trigger_handler` および `aidb.pipeline_live_trigger_handler` は、PGD クラスターでライブおよびバックグラウンド処理モードが失敗します。

回避策 トリガーハンドラーファンクションの所有権を `visual_pipeline_user` に転送します。

```
ALTER FUNCTION aidb.pipeline_background_trigger_handler OWNER TO visual_pipeline_user;
ALTER FUNCTION aidb.pipeline_live_trigger_handler OWNER TO visual_pipeline_user;
```

この問題は、将来の AIDB リリースで解決される予定です。

PGD クラスターでのナレッジベースのレプリケーション

7. 3.2 より前の AIDB では、`bdr_setup()` ファンクションは BDR レプリケーションにナレッジベーステーブルを含みません。これらのテーブルがないと、書き込みリーダーで作成されたナレッジベースレジストリエントリとナレッジベース-パイプラインの関連付けは、他の PGD ノードにレプリケートされません。この制限は、新しいリーダーにナレッジベースまたはそのパイプラインバインディングのレコードがない場合、書き込みリーダーのスイッチオーバー後に不一致が発生する可能性があります。

回避策 PGD クラスターで次の SQL を実行して、不足しているテーブルとシーケンスをレプリケーションに追加します。

```
SELECT bdr.alter_sequence_set_kind(aidb.knowledge_base_registry_id_seq::regclass,␣
↪galloc, 1);
SELECT bdr.replication_set_add_table(aidb.knowledge_base_registry);
SELECT bdr.replication_set_add_table(aidb.knowledge_base_pipeline);
```

このバグは AIDB 7.3.2 で解決されています。

6.4.5 Operate & Observe AI Factory

6.4.5.1 Observability (Agent Factory on HM)

6.4.5.1.1 ハイブリッドマネージャーの可観測性

ハイブリッドマネージャーにエージェントファクトリーコンポーネントを展開すると、プラットフォームは **Model Serving** KServe と **Langflow** の両方のワークロードの統合されたオブザーバビリティを提供します。

モデルサービング KServe

- プラットフォームのダッシュボードとログを使用して、**InferenceServices** と GPU ワークロードを監視します。
- メトリックには、要求レイテンシー、スループット、GPU / CPU 使用率、およびエラー率が含まれます。
- *Monitor InferenceService* を参照してください

Langflow

- Prometheus と Grafana を介して、Langflow ログ、ポッドの状態、レプリカ数、およびフローの可用性をモニタします。
- *Troubleshooting Langflow in Hybrid Manager* を参照してください

6.4.5.1.2 メトリックとログへのアクセス

- **ダッシュボード.** ハイブリッドマネージャーは、ビルトインの可観測性ダッシュボード Grafana を介してエージェントファクトリーメトリックを表示します。
- **ログ.** `kubectl logs` または HM UI ログビューアを使用して、Langflow ポッド、KServe InferenceServices、およびレトリーバージョブを検査します。

6.4.5.1.3 一般的なコマンド

クラスターレベルのトラブルシューティングまたはカスタムダッシュボードの場合、メトリックとログを直接取得することもできます。

```
#  Check logs for a running InferenceService

kubectl logs -n <project-namespace> svc/<model-service-name>

#  Port-forward Grafana if running locally

kubectl port-forward -n observability svc/grafana 3000:3000
```

6.4.5.1.4 Agent Factory メトリックの Grafana ダッシュボードの設定

Hybrid Manager には、オブザーバビリティスタックに Grafana インスタンスが含まれます。Launchpad からアクセスするか、エージェントファクトリー固有のメトリックのダッシュボードを構成できます。

グラファナにアクセス

ハイブリッドマネージャーポータルの Launchpad に移動し、Grafana タイルをクリックします。

推奨されるダッシュボードパネル

次のパネルを使用してダッシュボードを構築またはインポートして、エージェントファクトリーの状態を一目でカバーします。

Panel	Metric	Visualisation
Inference request rate	*rate(kserve_request_duration_seconds_count[5m])*	Time series
P95 inference latency	*histogram_quantile(0.95, rate(kserve_request_duration_seconds_bucket[5m]))*	Time series
Inference error rate	*rate(kserve_request_duration_seconds_count{respon*	Time series
Model server replica count	*kube_deployment_status_replicas_ready{deploymen*	Stat
Langflow replica count	*kube_deployment_status_replicas_ready{deploymen*	Stat
GPU utilisation	*DCGM_FI_DEV_GPU_UTIL*	Gauge

事前に構築されたダッシュボードをインポートする

KServe は、コミュニティ Grafana ダッシュボードを公開しています。インポートするには

1. Grafana で、 **Dashboards** → **Import** に移動します。
2. ダッシュボード ID を入力するか、 KServe community dashboards から JSON を貼り付けます。
3. Prometheus データソースを選択します。
4. **Import** を選択します。

6.4.5.1.5 重要なポイント

- **モデルサービス** → 完全なメトリックを使用して KServe サービスとして監視されます。
- **Langflow** → ポッドの状態、レプリカ数、およびフローの可用性は、Prometheus および Grafana を介して監視されます。
- **Hybrid Manager** は、統合された可観測性レイヤーを提供します。個別の監視スタックを構成する必要はありません。
- **エンタープライズモニタリング** → カスタム Grafana ダッシュボードでビルトインスタックを拡張します。

6.4.5.2 Troubleshooting Agent Factory on Hybrid Manager

6.4.5.2.1 診断の概要

このガイドでは、Hybrid Manager 環境内の Agent Factory コンポーネントの体系的なトラブルシューティング手順を提供します。問題は通常、インフラストラクチャの問題、モデル提供の失敗、およびアプリケーションレベルのエラーの 3 つのカテゴリに分類されます。

6.4.5.2.2 クイックリファレンスリンク

- *Langflow FAQ*
- *GPU Setup Guide*
- *InferenceService Creation*
- *Troubleshooting Langflow in Hybrid Manager*

6.4.5.2.3 インフラストラクチャの問題

GPU リソースの問題

症状

- InferenceService ポッドは保留中のままです
- 「不十分な nvidia.com/gpu」イベント
- モデルの初期化タイムアウト

診断手順

```
#  Check GPU node availability

kubectl get nodes -l nvidia.com/gpu=true

#  Verify GPU resource allocation

kubectl describe node <gpu-node-name> | grep -A 5 "Allocated resources"

#  Check GPU driver status

kubectl logs -n gpu-operator nvidia-driver-daemonset-<pod>
```

一般的な解決策

GPU ラベルの欠落 GPU を備えたノードは適切にラベル付けする必要があります。

```
kubectl label nodes <node-name> nvidia.com/gpu=true
```

GPU テイントの問題専用 GPU スケジューリングのテイント構成を確認します。

```
kubectl taint nodes <node-name> nvidia.com/gpu=NoSchedule
```

ドライバーの互換性 NVIDIA ドライバーバージョンが NIM コンテナの CUDA 要件と一致することを確認します。gpu-operator 名前空間のドライバーログに初期化エラーがないか確認します。

ストレージアクセス障害

症状

- モデルのダウンロードの失敗
- “ImagePullBackOff”ステータス
- エアギャップ環境でのプロファイルキャッシュエラー

診断手順

```
#  Check secret configuration

kubectl get secret nvidia-nim-secrets -n default -o yaml

#  Verify image pull secret

kubectl get secret ngc-cred -n <namespace> -o yaml

#  Test registry connectivity

kubectl run test-pull --image=nvcr.io/nim/nvidia/nvclip:latest --dry-run=client
```

解決戦略

レジストリ認証 認証が失敗した場合は、NGC 資格情報を再作成します。

```
kubectl delete secret ngc-cred -n default
kubectl create secret docker-registry ngc-cred \
  --docker-server=nvcr.io \
  --docker-username=$oauthtoken \
  --docker-password=<NGC_API_KEY> \
  -n default
```

エアギャップ環境オブジェクトストレージのプロファイルキャッシュの可用性と、モデル展開仕様の正しいパス構成を確認します。

6.4.5.2.4 モデル提供の失敗

InferenceService の準備ができていません

症状

- InferenceService は「NotReady」ステータスを示します
- 予測ポッドのクラッシュまたは再起動
- ヘルスチェックの失敗

調査コマンド

```
#  Check InferenceService status

kubectl get inferenceservice <name> -n <namespace>

#  Examine detailed conditions

kubectl describe inferenceservice <name> -n <namespace>

#  Review pod logs

kubectl logs <predictor-pod> -n <namespace> -c kserve-container
```

一般的な原因と修正

メモリ不足 NIM モデルには、大量のメモリが必要です。ポッドリソース要求を確認します。

```
kubectl describe pod <predictor-pod> -n <namespace> | grep -A 3 "Requests"
```

必要に応じて、InferenceService 仕様のメモリ割り当てを増加します。

モデルの読み込みタイムアウト大規模なモデルは、デフォルトの初期化タイムアウトを超過する場合があります。InferenceService 構成の readness プローブ設定を調整します。

プロファイルの不一致キャッシュされたプロファイルが GPU アーキテクチャと一致することを確認します。互換性のあるプロファイルをリストし、キャッシュの内容を確認します。

高い推論レイテンシー

症状

- 応答時間が SLA 要件を超過
- 予想を下回るトークン生成レート
- GPU の十分な使用率

パフォーマンス分析

```
#  Monitor GPU utilization

kubectl exec <predictor-pod> -n <namespace> -- nvidia-smi

#  Check request metrics

kubectl port-forward -n <namespace> svc/<service-name> 9090:9090

#  Access metrics at localhost:9090/metrics
```

最適化アプローチ

バッチサイズチューニング ServingRuntime 構成のバッチ処理パラメーターを調整して、スループットを向上させます。

レプリカのスケーリング負荷を分散するために InferenceService レプリカを追加します。

```
kubectl scale inferenceservice <name> -n <namespace> --replicas=3
```

リソース割り当て GPU メモリ割り当てがモデル要件と一致することを確認します。GPU メモリが不十分であると CPU フォールバックが強制され、パフォーマンスに重大な影響を与えます。

6.4.5.2.5 Langflow および Gen AI アプリケーションの問題

Langflow フローと EDB コンポーネントに関する問題-EDB ナレッジベースの取得失敗、EDB モデルサーバーエラー、フローインポート後のコンポーネント構成、およびフローレベルの品質問題を含む-については、*Troubleshooting Langflow in Hybrid Manager* を参照してください。

6.4.5.2.6 ログ分析

ログの場所

Agent Factory コンポーネントは、複数のレベルでログを生成します。

システムログ

- Kubernetes イベント `kubectl get events -n <namespace>`
- ノードログ GPU ノードの`/var/log/messages` または journalctl

アプリケーションログ

- InferenceService 予測ポッドコンテナログ
- Langflow フローアプリケーションポッドログ
- モデルライブラリー HM コントロールプレーンログ

メトリックとモニタリング

- Prometheus メトリック HM モニタリングスタックから利用可能
- カスタムダッシュボード HM コンソールの Grafana を介してアクセスできます

ログ集計

包括的な分析のための集中ロギングを構成します。

```
#  Stream logs from multiple components

kubectl logs -f -l app=inference-service -n <namespace> --all-containers

#  Export logs for analysis
```

(次のページに続く)

(前のページからの続き)

```
kubectl logs <pod> -n <namespace> --since=1h > diagnostic.log
```

6.4.5.2.7 アラート設定

クリティカルアラート

重要な状態のモニタリングアラートを構成します。

GPU の可用性 GPU ノードが使用不可になった場合、または GPU の割り当てが失敗した場合のアラート。

モデルの状態 InferenceService の準備状況と再起動頻度をモニタします。

パフォーマンスの低下推論レイテンシーのパーセンタイルとトークン生成レートを追跡します。

リソースの枯渇メモリ圧力、GPU メモリの飽和、またはストレージ容量に関するアラート。

アラート統合

HM アラートマネージャー構成を介して、アラートを組織の通知システムと統合します。

6.4.5.2.8 エスカレーションプロシージャ

サポートリソース

内部のトラブルシューティングが不十分であることが判明した場合

1. **ドキュメントレビュー**
   - *Agent Factory Hub* に相談します
   - モデル固有のドキュメントを確認します
2. **コミュニティリソース**
   - モデル固有の問題に関する NVIDIA NIM ドキュメント
   - インフラストラクチャの問題を提供するための KServe コミュニティ
3. **EDB サポート**
   - HM サポートツールを使用して診断バンドルを収集する
   - 関連するログ、構成、エラーメッセージを含めます
   - 参照固有のコンポーネントバージョンと展開仕様

診断情報の収集

包括的な診断情報を準備します。

```
#  Generate support bundle

kubectl cluster-info dump --output-directory=./cluster-dump
```

(次のページに続く)

(前のページからの続き)

```
#  Collect Agent Factory specifics

kubectl get inferenceservice -A -o yaml > inferenceservices.yaml
kubectl get pods -A -l serving.kserve.io/inferenceservice -o wide > serving-pods.txt
kubectl describe nodes -l nvidia.com/gpu=true > gpu-nodes.txt
```

6.4.5.2.9 予防措置

定期的な健康診断

プロアクティブなモニタリングを実装します。

- 毎週の GPU ドライバーとオペレーターのステータス検証
- 毎日の推論サービスの状態評価
- 継続的なパフォーマンスベースライン追跡

キャパシティプランニング

リソースの傾向を監視します。

- GPU 使用率パターン
- メモリ消費量の増加
- ストレージ使用量の予測
- リクエスト量の傾向

アップデート管理

構成要素通貨を維持します。

- NVIDIA NIM モデルの更新を追跡します
- セキュリティアドバイザリの監視
- 更新のためのメンテナンス時間枠を計画する
- 非実稼働環境でのテスト更新

第7章　Analytics in Hybrid Manager

テーブル管理、階層化ストレージ構成、カタログの探索などの分析管理を、ハイブリッドマネージャー HM コンソールから直接実行します。

AHA および DHA クラスターには、Postgres が Delta Lake、Apache Iceberg、Parquet のようなオープンテーブル形式で保存されている大規模なデータを照会できる高性能拡張機能である Postgres Analytics Accelerator PGAA が含まれています。

プロジェクトにシングルノード AHA または最小限の DHA クラスターを展開することにより、HM コンソールからクラスターの分析をエンドツーエンドで管理します。専用クラスターは、HM プロジェクトの AHA および DHA クラスターにわたるすべての分析テーブルの統合ビューを提供し、単一のインターフェイスから利用可能なテーブル管理、階層化ストレージ構成、およびカタログ探索を提供します。

概念、アーキテクチャ、およびリファレンスドキュメントについては、PGAA documentation を参照してください。

7.1 Configuring tiered storage

Postgres テーブルをオブジェクトストレージにレプリケートしてオフロードし、クラスターからデータを移行せずに標準の Postgres SQL で照会します。 Advanced High Availability AHA および分散高可用性 DHA は、ハイブリッドマネージャー HM によって管理される EDB Postgres 分散 PGD ベースのクラスタータイプで、Postgres Analytics Accelerator PGAA がプリインストールおよび自動的に構成されます。

概念、アーキテクチャ、およびリファレンスドキュメントについては、PGD ドキュメントの *Using tiered storage* を参照してください。

7.1.1 クラスターの分析の管理

AHA および DHA クラスターのオブジェクトストレージは、HM インストールで *creating a storage secret* として構成されます。HM は、そのオブジェクトストレージによってバッキングされるデータグループごとに Iceberg カタログを自動的にプロビジョニングし、アナリティクスの書き込み先として構成します。PGAA はプリインストールされており、カタログはすぐに使用できます。手動セットアップは必要ありません。**Estate > Catalogs** でカタログを表示および管理できます。または、クラスター間のクエリ用に追加の共有カタログを作成できます。詳細は、*Managing catalogs* を参照してください。

Actions メニューから **Enable Analytics** および **Disable Analytics** を使用して、データグループレベルで分析を制御します。分析を無効にすると、そのデータグループ内のすべてのテーブルでの新しいレプリケーションとオフロード操作ができなくなります。既存のオフロードテーブルにはアクセスしたままで、カタログ構成は保持されます。

7.1.2 テーブルの表示

クラスターの詳細ビューの **テーブル**タブには、クラスター内のすべてのテーブルと、そのタイプ、レプリケーションステータス、および使用可能なアクションがリストされます。

!!!note デフォルトでは、`edb.collect_table_statistics` 構成パラメーターは `pgaa` に設定され、Iceberg テーブルのみの統計を収集します。ヒープテーブルと HTAP テーブルを含めるには、**Edit Cluster > Data Groups > DB Configuration** を介して `all` に設定します。クラスターを再起動する必要はありません。

各テーブルには、次のいずれかのタイプがあります。

- **Heap** — PGAA メタデータを含まない標準の Postgres テーブル。
- **HTAP** ハイブリッドトランザクションおよび分析処理 - 分析レプリケーションが有効になっているヒープテーブル。データがオブジェクトストレージに継続的にレプリケートされている間、テーブルは完全なトランザクションのままです。アナリティクスへの複製 を参照してください。
- **Iceberg** —オブジェクトストレージに完全に保存されたコールド分析テーブル。ローカルコピーは切り詰められました。アナリティクスへのオフロード を参照してください。
- **Tiered** —PGD AutoPartition によって管理されるパーティションテーブル、コールドパーティションは Iceberg です。ホットパーティションは、リアルタイム分析レプリケーションが有効になっている場合は HTAP であり、そうでない場合はヒープです。階層テーブルへの変換 を参照してください。

テーブル行を選択すると、タブの上部に概要パネルが表示され、そのテーブルのストレージメトリックとして、ディスク上の行数と合計サイズ、およびオブジェクトストレージの行数と合計サイズが表示されます。これらのメトリックは、両方のストレージ階層にわたるテーブルの現在の状態を反映しています。

7.1.3 階層テーブルへの変換

ヒープテーブルを階層化テーブルに変換して、PGD AutoPartition によって管理されるレンジパーティションテーブルとして再構成します。コールドパーティションは、分析オフロード期間を過ぎて古くなると、Iceberg 形式に自動的にオフロードされます。ホットパーティションは HTAP であり、リアルタイム分析レプリケーションが有効になっている場合はオブジェクトストレージに継続的にレプリケートされ、そうでない場合はヒープとして残ります。

テーブルは、変換する前に次の要件を満たしている必要があります。

- 主キーがあります。
- パーティション列は、主キーの一部でもある `DATE` または `TIMESTAMP` 列です。
- 外部キーコンストレインまたはシーケンスはありません。変換後にこれらを削除して再作成します。

テーブルを変換するには

1. HM コンソールで、 **Estate > Postgres** に移動し、クラスターを選択します。
2. **Tables** タブを選択します。
3. テーブル行で、3 点メニューを開き、 **Convert to Tiered Table** を選択します。
4. パーティショニングオプションを入力します。

- **Partition Column** - パーティション化する `DATE` または `TIMESTAMP` 列

- **Initial Lower Bound** - 最初のパーティションの開始日。
- **アナリティクスオフロード期間** — Iceberg.
- にオフロードされるまでにパーティションがホットなままになる時間**リアルタイム分析レプリケーションを有効にする** —有効にすると、ホットパーティションが HTAP テーブルとしてオブジェクトストレージに継続的にレプリケートされます。無効にすると、ホットパーティションはヒープテーブルとして残り、オフロードウィンドウからエージングアウトすると一括オフロードされます。
- **Partition Increment** - 各パーティションの時間間隔。
- **Minimum Advance Partitions** - 最小数
- **最大事前パーティション** - 維持する将来のパーティションの最大数。

5. **Convert to Tiered Table** を選択し、変換が完了するまで待ちます。

変換後、テーブルタイプは **階層**に変更されます。テーブルの行を展開すると、個々のパーティションとそのタイプ Heap、HTAP、または Iceberg および境界式が表示されます。[**Tables**] タブには、パーティションタイプ間の内訳を含むパーティション分布グラフも表示されます。パーティションのライフサイクルは、変換後に PGD AutoPartition によって自動的に管理されます。

HM コンソールは、変換後に階層化テーブルを変更するアクションを提供していません。オフロード期間やパーティションのインクリメントなどの設定を変更するには、データベースクライアントで直接 `bdr.autopartition()` を再実行します。階層化テーブル設定の変更の詳細については、PGD ドキュメントの *Implementing tiered tables* を参照してください。

7.1.4 アナリティクスへの複製

ヒープテーブルでレプリケーションを有効にして、オブジェクトストレージへの継続的レプリケーションを開始します。テーブルタイプは **HTAP** に変更され、データの変更はクラスターの事前構成されたデータグループカタログにほぼリアルタイムでストリーミングされます。カタログ名は確認ダイアログに表示されます。

レプリケーションを有効にする前に、テーブルが次の要件を満たしている必要があります。

- 主キーがあります。

レプリケーションを有効にするには

1. HM コンソールで、 **Estate > Postgres** に移動し、クラスターを選択します。
2. **Tables** タブを選択します。
3. テーブル行で、3 点メニューを開き、 **Enable Replication** を選択します。
4. 確認ダイアログを確認し、 **Enable Replication** を選択します。

初期同期が完了すると、テーブルタイプが **HTAP** に変更され、レプリケーションステータスが **Enabled** と表示されます。

テーブルが HTAP になると、次のアクションが使用できます。

- **Disable Replication** —レプリケーションを停止し、PGAA メタデータを削除し、テーブルをヒープに戻します。以前にオブジェクトストレージにレプリケートされたデータは、自動的に削除されず、カタログに残ります。

- **オフロードテーブル** —テーブルをコールドの Iceberg テーブルに変換し、ローカルコピーを切り捨てます。
- **トリガーコンパクション** —オブジェクトストレージに断片化された Parquet ファイルを統合して、クエリパフォーマンスを向上させ、ストレージオーバーヘッドを削減します。

!!!note 以前にレプリケーションが無効になっていたテーブルでレプリケーションを有効にすると、カタログのメタデータが残っているため失敗します。エラーを解決するには、`purge_analytics_target` パラメーターを使用してデータベースクライアントで直接レプリケーションを再度有効にします。

```
CALL pgaa.enable_analytics_replication(<schema>.<table>::regclass, true);
```

PGAA ドキュメントの pgaa.enable_analytics_replication を参照してください。

7.1.5 アナリティクスへのオフロード

HTAP テーブルをオフロードしてコールドの Iceberg テーブルに変換し、ローカルコピーを切り捨て、分析コピーのみをオブジェクトストレージに残します。確認ダイアログには、データが保存される宛先カタログクラスターの事前構成されたデータグループカタログが表示されます。

テーブルは、オフロードする前に次の要件を満たしている必要があります。

- 他のテーブルは、それを指す外部キーコンストレインを持つことはできません。

テーブルをオフロードするには

1. HM コンソールで、 **Estate > Postgres** に移動し、クラスターを選択します。
2. **Tables** タブを選択します。
3. HTAP テーブル行で、3 点メニューを開き、 **Offload Table** を選択します。
4. 確認ダイアログを確認し、 **Offload** を選択します。

レプリケーションが無効になると、テーブルタイプが **Iceberg** に変更されます。Iceberg テーブルは読み取り専用で、直接書き込みをサポートしていません。`INSERT` 、`UPDATE` 、および `DELETE` 操作はサポートされていません。テーブルに再度書き込むには、最初に **HTAP** に復元します。

Iceberg テーブルからは、次のことができます。

- **Restore Table** — データをローカルストレージに移動します。テーブルは **HTAP** に戻ります。オブジェクトストレージのコピーは、自動的に削除されません。
- **トリガーコンパクション** —オブジェクトストレージに断片化された Parquet ファイルを統合して、クエリパフォーマンスを向上させ、ストレージオーバーヘッドを削減します。

7.1.6 コンパクションのトリガー

コンパクションを使用して、オブジェクトストレージ内の断片化された Parquet ファイルを、より大きな最適化されたファイルに統合します。コンパクションにより、クエリパフォーマンスが向上し、ストレージオーバーヘッドが削減されます。コンパクションは、 HTAP および Iceberg テーブルで利用できます。

圧縮をトリガーするには

1. HM コンソールで、 **Estate > Postgres** に移動し、クラスターを選択します。
2. **Tables** タブを選択します。
3. テーブル行で、3 点メニューを開き、 **Trigger Compaction** を選択します。
4. 確認ダイアログを確認し、 **Trigger Compaction** を選択します。

7.2 Managing catalogs

Iceberg カタログをクラスターにアタッチして、クラスター間でデータを移動せずにエステート全体で分析データを照会します。Hybrid ManagerHM は、 *object storage configured at installation* によって支援される Advanced High Availability AHA または分散高可用性 DHA クラスター内のデータグループごとに Iceberg カタログを自動的にプロビジョニングします。すべてのレプリケーションおよびオフロード操作は、このカタログに書き込みます。**Estate > Catalogs** の下でカタログを表示します。

Iceberg カタログの概念については、 Postgres Analytics AcceleratorPGAA ドキュメントの Integrating with Iceberg catalogs を参照してください。

7.2.1 クラスターのカタログの管理

クラスターの詳細ビューの **カタログ**タブから、AHA または DHA クラスターに関連付けられているカタログを表示します。次のアクションを使用して、テーブルメタデータをクエリーに利用できるようにします。

- **Import** —テーブルメタデータをカタログからクラスターにインポートします。ポイントインタイムスナップショットに **Import** を使用し、メタデータを更新するたびにそれを再度実行します。
- **Enable Sync** —テーブルメタデータをカタログからクラスターに継続的に同期し、新しいものがカタログに追加されるたびに、クラスターに分析テーブルを作成します。同期ステータスが **Enabled** に変更されます。テーブルがアクティブに変更され、クエリに常に最新の状態を反映したい場合は、**Enable Sync** を使用します。

7.2.2 クラスター全体で分析データのクエリー

同じプロジェクト内の別のクラスターからカタログを接続して、PGAA を使用してクラスターからデータを照会します。たとえば、AHA クラスターからデータをオフロードし、DHA クラスターから照会します。その逆も可能です。

カタログを添付するには

1. クラスターの **カタログ** タブに移動し、 **追加カタログ > 管理対象カタログの追加**を選択し、カタログを検索します。プロジェクトに既に存在するカタログのみが検索結果に表示されます。
2. **Import** または **Enable Sync** を使用して、カタログのテーブルをクエリーに利用できるようにします。

カタログ添付テーブルは、ソースクラスターでのタイプに関係なく、インポートクラスターでは **Iceberg** タイプとして表示されます。これらは読み取り専用であり、`INSERT` 、`UPDATE` 、または `DELETE` をサポートしていません。

別のクラスターのカタログで管理されている Iceberg テーブルで **Restore Table** を使用して、ローカルヒープテーブルとして自分のクラスターにデータをコピーします。ソースクラスターとそのデータは影響を受けませんが、復元されたテーブルはカタログにリンクされません。

!!!警告別のクラスターのカタログからテーブルを復元しても、元のスキーマは保持されません。Iceberg の型システムは Postgres のサブセットであるため、制約主キー、`NOT NULL`、一意が失われ、列型は汎用の同等のものに強制され、デフォルト値が削除されます。

第8章　Sovereign Data and Agent Factory

Sovereign Data と Agent Factory は、EDB と Supermicro ®が共同設計した完全に統合されたハードウェアおよびソフトウェアプラットフォームです。これにより、組織は外部に依存せずに、プロダクショングレードの Postgres ®および　AI ワークロードを独自のデータセンターに展開できます。

この設計されたシステムは、主権、セキュリティ、およびスケールのために最適化されています。事前に組み立てられ、ライフサイクル管理されており、インストールするとすぐに高度な Postgres、vector、AI パイプラインを実行できます。

8.1 ハードウェアベースのシステムを選択する理由は何ですか?

クラウドファーストは常にオプションであるとは限りません。ソブリンデータとエージェントファクトリーは、次のような環境に存在します。

- データとモデルの制御は法的または運用上必要です
- クラウド GPU アクセスが一貫していない、オーバーサブスクライブ、または制限されている
- ソブリン AI は、外部推論またはデータ移動なしを必要とします
- Postgres、vector、AI にわたる統合された可観測性が重要です
- ライフサイクルの信頼性はハードウェアとソフトウェア全体で重要

このシステムは、ハードウェア、オーケストレーションプラットフォーム、サポートされているランタイムを密に結合することにより、最新のワークロードから摩擦を取り除くため、インフラストラクチャではなく結果に集中できます。

8.2 中身は何ですか

ラックマウントシステムとしてデータセンターに提供されるプラットフォームには、次のものが含まれます。

- 検証済みの Supermicro サーバー
- ライフサイクル管理された Kubernetes とハイブリッドマネージャー
- ビルトイン HA Postgres オーケストレーション
- pgvector、埋め込みパイプライン、およびモデルサービング機能
- オプションの GPU ベースの AI 推論とエージェントランタイム
- 完全なモニタリング、パッチ適用、およびバージョンアップグレードの自動化

すべての構成は、アクセスを配信する前に処理されます。インストールしたら、ハイブリッドマネージャー Web インターフェイスを使用して、Postgres クラスター、RAG パイプライン、およびエージェントアプリケーションの展開をすぐに開始します。

8.3 サポートされているワークロード

さまざまなソブリンまたはオフライン対応ワークロードを実行できます。

- 高可用性 Postgres データベース
- ハイブリッド内部 DBaaS 環境
- ベクトル検索と埋め込みパイプライン
- ローカル LLM を使用した取得拡張世代 RAG
- GPU アクセラレーションを使用したオンプレミス AI 推論
- 内部チャットボット、コパイロット、エージェントオーケストレーション

例については、 *Common workloads and use cases* を参照してください。

8.4 ハードウェア構成

2 つの検証済み構成を提供します。

- **コアコンピューティング** —高性能 Postgres およびベクターワークロード用。
- **Advanced AI** —推論およびエージェントベースのシステム用の GPU 対応ノードを追加します。

どちらの構成も拡張可能で、完全にサポートされています。詳細は、*System specifications* を参照してください。

8.5 Explore 機能

- *Concepts and architecture*
- *Glossary and key terms*
- *What to expect during setup*
- *Common workloads and use cases*
- *Langflow FAQ*

ライフサイクル操作の詳細については、 *Using Hybrid Manager* を参照してください。

8.5.1 Concepts and architecture

Sovereign Data と Agent Factory は、物理インフラストラクチャ、オーケストレーションソフトウェア、ライフサイクル自動化を単一のシステムに組み合わせ、データセンターに事前に組み立ておよびインストールします。

システムアーキテクチャとコアロールを理解することは、Postgres ワークロードと AI パイプラインがどのように展開、管理、およびスケーリングされる方法を明確にするのに役立ちます。

前提条件として、 *System specifications* の構成の詳細を参照してください。

8.5.1.1 ノードタイプとロール

システムには、それぞれが明確な責任と構成を持つ複数のノードロールが含まれています。

8.5.1.1.1 管理ノード

管理ノードは、ソブリンデータとエージェントファクトリーの追加ソフトウェアパッケージ、ライセンス、およびサポートツールを提供します。インストールごとに 1 つの管理ノードが必要です。

8.5.1.1.2 ハイブリッドマネージャーノード

3 つのハイブリッドマネージャーノードは、高可用性コントロールプレーンを形成します。これらは、Kubernetes のスケジューリング、クラスターライフサイクル、オブザーバビリティ、および API サービスを管理します。

ファンクションには次のものが含まれます。

- データベースクラスター展開の管理
- Postgres、OS、および GPU ワークロードからの 200 以上のメトリックの監視
- アップデートとパッチの調整
- 構成 UI と API の提供

これらのノードは、データワークロードの実行には使用されません。

8.5.1.1.3 コンピューティングノード

コンピューティングノードは、Postgres データベース、分析パイプライン、またはベクターワークロードを実行するために使用されます。これらのノードは以下をサポートしています

- EDB Postgres Advanced Server または Extended Server
- EDB Postgres 分散オプショナル
- Apache Iceberg ®などのレイクハウスフォーマット
- Python ベースの埋め込みまたは変換パイプライン

8.5.1.1.4 AI ノードオプショナル

AI ノードは、大規模なモデルの推論とエージェントの実行をサポートする GPU 対応サーバーです。システムには 2 つのタイプが含まれます。

- 中規模モデルサービング NVIDIA Ada L40S 用の **2-GPU ノード**
- LLM およびエージェントランタイム用の **8-GPU ノード** B200 GPU を搭載した NVIDIA HGX

これらのノードは KServe と統合して、オンプレミスのモデル提供と推論ルーティングを提供します。

8.5.1.2 オーケストレーションとライフサイクル

システムは、ハイブリッドマネージャーによって管理されるプリインストールされた Kubernetes クラスターを実行します。Kubernetes には直接アクセスしません。代わりに、ハイブリッドマネージャーは以下を行うための Web ベースのインターフェイスを提供します。

- 高可用性を備えた Postgres クラスターを展開する
- ベクターパイプラインまたは AI ワークフローのインストールと監視
- ワークロードを追加ノードにスケーリング
- パフォーマンスとリソースの使用状況を監視する

EDB は、ソフトウェアとハードウェアの両方にライフサイクルサービスを提供します。

- メジャーおよびマイナーバージョンのアップグレード
- セキュリティパッチ
- ハードウェアファームウェアと BIOS のアップデート
- 自動クラスターリバランスと再起動処理

8.5.1.3 「ソブリン」とは何ですか？

この文脈での主権とは、次のことを意味します。

- すべてのデータとモデルは物理的環境にあります
- 外部クラウドへの依存関係やデータ共有はありません
- あなたがハードウェアを所有し、ネットワークアクセスを制御します
- 規制対象業界のための完全に監査対応の隔離された環境

ソブリンデータとエージェントファクトリーは以下に適しています。

- 政府または国家のデータインフラストラクチャ
- ローカルデータ規制の対象となる金融サービス
- 厳しいクラウド制限または機密 IP を持つ企業

8.5.1.4 サポートされているワークロードパターン

各構成は、さまざまなユースケースをサポートしています。

ワークロード |説明 |コアコンピューティング |高度な AI ——— |————— -|————— –|——— — - 高可用性 Postgres | 3 ノード HA クラスターを実行する | ☒ | ☒ エンベディング|内部データをベクター形式に変換する | ☒ | ☒取得拡張生成 RAG | Postgres + LLM を組み合わせる | ☒ | ☒ 推論 | GPU を使用してモデルをローカルに提供する | ☒ | ☒エージェントワークフロー |自律的に動作する AI エージェントを構築する | ☒ | ☒

構成の詳細については、 *System specifications* を参照してください。

8.5.2 Frequently asked questions

8.5.2.1 クラウドサービスの代わりにこのシステムを使用するのはなぜですか?

Sovereign Data と Agent Factory は、インフラストラクチャの完全な物理的および運用的な制御を提供します。次の目的に設計されています。

- データをオンプレミスに残しておく必要がある環境
- 一貫した予測可能なパフォーマンスを必要とするワークロード
- AI 機能が必要だが、クラウド API または GPU サービスを使用できない組織
- 単一のプラットフォームからの統合された Postgres と AI オーケストレーションを必要とするチーム

8.5.2.2 ハードウェアのインストールと保守は誰が行いますか?

Supermicro と EDB は、共同でお客様のサイトにシステムをインストールします。すべてのハードウェア、ファームウェア、および OS レベルのメンテナンスは、Postgres、ハイブリッドマネージャー、および AI ツールのサポートとともに、契約の下でカバーされています。

8.5.2.3 どのようなソフトウェアを実行しますか?

システムは以下を実行します。

- EDB Postgres Advanced Server または Extended Server
- オーケストレーションとオブザーバビリティのためのハイブリッドマネージャー
- Agent Factory コンポーネント
- ベクターおよび分析ワークロード用の pgvector および Apache Arrow/Iceberg
- モデルサービング用のコンテナと KServe
- オプション マルチノード HA 用の EDB Postgres 分散

8.5.2.4 このシステムをクラウドサービスに接続できますか?

できますが、必須または想定されていません。プラットフォームは、完全に自己完結型になるように設計されています。内部ポリシーに応じて、アウトバウンドトラフィックを選択的に有効にすることができます。

8.5.2.5 AI の使用がサポートされているモデルは何ですか?

システムは、次のような標準のコンテナ化モデルをサポートします。

- `bge-base` 、`instructor-xl` 、`e5` のようなモデルの埋め込み
- `LLaMA2` 、`Mistral` 、`Mixtral` のような言語モデル
- タスク固有のモデル要約、分類など

カスタムモデルは、推論コンテナ形式に準拠している限りにおいて展開できます。

8.5.2.6 外部データベースをモニタできますか?

はい。ハイブリッドマネージャーは、オブザーバーモードで外部 Postgres クラスターのメトリックを監視および表示できます。ただし、承認されたソブリンデータおよびエージェントファクトリーノードのみが内部クラスターに追加し、ライフサイクルワークフローで管理できます。

8.5.2.7 システムをスケーリングするにはどうすればよいですか?

追加のコンピューティングノードまたは GPU 対応 AI ノードを注文してラックにできます。

8.5.2.8 サポートには何が含まれますか?

- ソフトウェアスタック EDB の 24 時間 365 日のグローバルサポート
- ハードウェアとファームウェアのサポート
- ハイブリッドマネージャーを介して管理されるセキュリティパッチとバージョンのアップグレード
- 拡張、更新、またはカスタム構成のライフサイクル支援

8.5.3 Glossary and key terms

これらの重要な用語とアーキテクチャコンポーネントは、ソブリンデータとエージェントファクトリーのドキュメント全体で使用されています。

8.5.3.1 管理ノード

システムの中央制御ノード。ハイブリッドマネージャーインターフェイス、認証サービス、およびオーケストレーションロジックをホストします。

8.5.3.2 AI ノード

モデル推論、ベクトル処理、およびエージェント実行に使用される GPU 対応サーバー。オプションの AI ワークロード構成の一部。

8.5.3.3 エージェントワークフロー

AI エージェントが内部ツール、メモリ、フィードバックループを使用して独立してタスクを実行するマルチステップの自動化パターン。多くの場合、LLM およびベクターデータに基づいて構築されます。

8.5.3.4 コントロールプレーン

クラスターのスケジューリング、モニタリング、ライフサイクル操作、およびアクセス制御を担当する 3 つのハイブリッドマネージャーノードのセット。データワークロードとは無関係に動作します。

8.5.3.5 コンピューティングノード

Postgres クラスター、分析、またはベクターパイプラインを実行するために使用される汎用の高性能ノード。必要に応じてスケールアウトできます。

8.5.3.6 埋め込み

構造化または非構造化データを、検索や分類などの AI ワークフローに適したベクトル表現に変換するプロセス。

8.5.3.7 EDB ハイブリッドマネージャー

Kubernetes 環境でデータベースと AI ワークロードをオーケストレーションするための EDB の管理レイヤークラスターの展開、システムの監視、およびライフサイクル操作の管理に使用されます。

8.5.3.8 推論

機械学習モデルから出力を生成する行為。このシステムでは、推論は、KServe および NVIDIA NIM コンテナを使用して専用の GPU ノードで実行されます。

8.5.3.9 KServe

推論ワークロードを実行するために使用されるオープンソースのモデル提供フレームワーク。LLM を提供し、モデルを埋め込むために、ハイブリッドマネージャーと GPU ノードに統合されています。

8.5.3.10 ライフサイクルサービス

セキュリティパッチ、OS 更新、Postgres のアップグレード、ファームウェア管理を含む、ハードウェアとソフトウェアの両方の継続的なサポートと管理。

8.5.3.11 EDB Postgres Distributed PGD

EDB のマルチノード、高可用性 Postgres ソリューション。グローバルに分散または同期のマルチリージョンクラスターを必要とする顧客のオプション。

8.5.3.12 取得拡張生成 RAG

構造化データ多くの場合 Postgres が取得されて LLM に渡されて、自然言語応答を生成するワークロードパターン。埋め込み機能と推論機能が必要です。

8.5.3.13 ソブリン AI

すべてのデータ、モデル、実行環境が、必要に応じて完全にエアギャップされた顧客制御のスペース内に物理的および論理的に残る AI 展開モデル。

8.5.3.14 Vector ストア

埋め込みベクトルを保存し、類似性検索をサポートするデータベースまたはインデックス。Postgres と pgvector は、このシステムでローカルベクターストレージに使用されます。

8.5.4 System specifications

Sovereign Data および Agent Factory システムは、Core Compute と Advanced AI の 2 つの構成層で利用できます。各システムは事前に組み立てられ、完全なソフトウェアとハードウェアのサポートでライフサイクル管理されています。

8.5.4.1 構成オプション

Configuration	Description	Total nodes
Core Compute	高性能 Postgres およびハイブリッド DBaaS ワークロード用に設計	7
Advanced AI	AI および推論ワークロード用の GPU ノードでコアコンピューティングを拡張	11

8.5.4.2 構成 1 コアコンピューティング

含まれるもの

- 1x 管理ノード
- 3x ハイブリッドマネージャー/コントロールプレーンノード
- 3x Hyper 2U コンピューティングノード

8.5.4.2.1 管理ノード

Category	Product / Model	Description	Qty
System	SYS-111C-NR	1U X13SEDW-F サーバー	1
Processor	P4X-EMR4514Y-SRN6J-MC	Intel Xeon Gold 4514Y 16 コア@ 2.0GHz	1
Memory	MEM-DR516L-SL01-ER56	16GB DDR5-5600 1RX8 LP ECC RDIMM 合計 128GB	8
Storage	HDS-MUN-MTFDKCC3T2TFS1BC	Micron 7450 MAX 3.2TB NVMe PCIe 4.0 U.3	2
Storage Controller	AOC-VROCSTNMOD	Intel VROC RSTe 標準 HW キー	1

次のページに続く

表 2 – 前のページからの続き

Category	Product / Model	Description	Qty
Networking	AOC-AG-I2M-O	AIOM デュアルポート 1GbE RJ45	1
Networking	AOC-S25G-I2S-O	標準ロープロファイル 25G デュアルポート SFP28	1

8.5.4.2.2 コアコンピューティング – Hyper 2U サーバーサーバーごとに x6

Category	Product / Model	Description	Qty
System	SYS-221H-TNR	ハイパー 2U サーバーシャーシ、8x2.5 インチベイ、X13DEM マザーボード	1
Processor	R6538N-SRN6V-MCC	Intel Xeon Platinum 6538N 32 コア@ 2.1GHz	2
Cooling	SNK-P0087V	X13 プラットフォーム用のパッシブ CPU ヒートシンク	2
Memory	MEM-DR532L-SL01-ER56	32GB DDR5-5600 2Rx8 LP ECC RDIMM 合計 1024GB	32
Storage	HDS-SMN0-MZ1L2960HCJR	Samsung PM9A3 960GB NVMe PCIe 4.0 M.2	2
Storage	HDS-TUN-KCD8XPUG15T3	Kioxia CD8-P 15.3TB NVMe PCIe 5.0 2.5 インチ	4
Storage	HDS-TUN-KCD8XPUG1T92	Kioxia CD8-P 1.92TB NVMe PCIe 5.0 2.5 インチ	2
Storage Controller	AOC-VROCPREMOD	Intel VROC プレミアムコントローラ RAID 0、1、5、10	1
Networking	AOC-A100G-M2CM-O	AIOM 2 ポート 100GbE QSFP28、Mellanox CX-6 DX	2
Power	PWS-2K63A-1R	1U 2600W 冗長電源	2

次のページに続く

表 3 – 前のページからの続き

Category	Product / Model	Description	Qty
Cooling	FAN-0239L4	80x80x38 mm、16.8K RPM、中間冷却ファン	4
Expansion	RSC-H-68G5 / RSC-H-6G5L	1U LHS ハイパーライザーカード PCIe 5.0 x16/x8、PCIe 5.0 x16	2
Security	AOM-TPM-9672V-O	TPM 2.0 モジュール	1

8.5.4.3 構成 2: 高度な AI

すべてのコアコンピューティングコンポーネントに加えて、以下が含まれます。

- 2x 2-GPU Hyper 2U サーバー
- 2x 8-GPU B200 10U サーバー

8.5.4.3.1 2-GPU Hyper 2U サーバーサーバーごとに x2

Category	Product / Model	Description	Qty
System	SYS-221H-TNR	ハイパー 2U サーバーシャーシ、8x2.5 インチベイ、X13DEM マザーボード	1
Processor	R6538N-SRN6V-MCC	Intel Xeon Platinum 6538N 32 コア@ 2.1GHz	2
Cooling	SNK-P0087V	X13 プラットフォーム用 1U パッシブ CPU ヒートシンク	2
Memory	MEM-DR532L-SL01-ER56	32GB DDR5-5600 2Rx8 LP ECC RDIMM 合計 1024GB	32
GPU	GPU-NVL40S	NVIDIA Ada L40S 48GB GDDR6 PCIe Gen 4	2

次のページに続く

表 4 – 前のページからの続き

Category	Product / Model	Description	Qty
Storage	HDS-SMN0-MZ1L2960HCJR	Samsung PM9A3 960GB NVMe PCIe 4.0 M.2	2
Storage	HDS-TUN-KCD8XPUG15T3	Kioxia CD8-P 15.3TB NVMe PCIe 5.0 2.5 インチ	4
Storage	HDS-TUN-KCD8XPUG1T92	Kioxia CD8-P 1.92TB NVMe PCIe 5.0 2.5 インチ	2
Storage Controller	AOC-VROCPREMOD	Intel VROC プレミアムコントローラ RAID 0、1、5、10	1
Networking	AOC-A100G-M2CM-O	AIOM 2 ポート 100GbE QSFP28、Mellanox CX-6 DX	2
Power	PWS-2K63A-1R	1U 2600W 冗長電源	2
Cooling	FAN-0239L4	80x80x38 mm、16.8K RPM、中間冷却ファン	4
Cooling	MCP-310-8294...	GPU プラスチックエアシュラウド	2
Expansion	RSC-H-68G5 / RSC-H-6G5L	1U LHS ハイパーライザーカード PCIe 5.0 x16/x8、PCIe 5.0 x16	2
Security	AOM-TPM-9672V-O	TPM 2.0 モジュール	1

8.5.4.3.2 8-GPU B200 10U サーバーサーバーごとに x2

Category	Product / Model	Description	Qty
System	SYS-A21GE-NBRT-G1	Supermicro 10U B200 Gold シリーズ GPU サーバー	1
Processor	Intel ® Xeon ® Platinum 8570	56 コア@ 2.10GHz 350W	1
Memory	DDR5 ECC RDIMM	96GB DDR5 5600MHz ECC RDIMM 合計 3072GB	32

次のページに続く

表 5 – 前のページからの続き

Category	Product / Model	Description	Qty
GPU	NVIDIA HGX B200	8x B200 180GB GPU を搭載したベースボード	1
DPU	NVIDIA BlueField-3	BF3220 DPU 200GbE デュアルポート、暗号化対応	1
Storage	PM9A3 M.2	1.9TB M.2 NVMe	1
Storage	7450 PRO 2.5"	3.8TB NVMe PCIe 4.0 SSD 1 DWPD	1
*Networking	AOC-1	• 400 ギガビット MCX75310AA NEAT OSFP アダプター	1
Onboard	—	2x RJ45 10Gbase-T ポート	1
Security	AOM-TPM-9670V-P	TPM 2.0 モジュール	1

8.5.5 Common workloads and use cases

Sovereign Data and Agent Factory は、データ、インフラストラクチャ、AI の実行の厳密な制御を必要とする組織のために専用に構築されています。ローカルで管理されながら、幅広いワークロードをサポートします。

8.5.5.1 クラウドでエンジニアドシステムを使用する理由は何ですか?

多くの最新のワークロードはクラウド環境で実行できますが、すべての組織がクラウド環境に依存できるわけではありません、または依存する必要があります。Sovereign Data と Agent Factory は、次のニーズを満たすために存在します。

8.5.5.1.1 データ主権とコンプライアンス

- すべてのデータ、モデル、ワークロードは物理的インフラストラクチャ内に残ります。
- 外部 API やサードパーティのストレージは必要ありません。
- 金融、政府、医療、国家安全保障分野での展開を可能にします。

8.5.5.1.2 規制された環境

- インターネットへの依存を制限して動作します。
- 導入ライフサイクルをサポートします。
- 以前は閉ざされていた環境で LLM および AI のユースケースを有効にします。

8.5.5.1.3 Postgres + AI の統合プラットフォーム

- HA Postgres、ベクトル検索、および GPU 推論を 1 つのシステムに組み合わせ、組織によって制御されます。
- オーケストレーション、モニタリング、およびサポートを簡素化します。
- 断片化したソリューションと重複したインフラストラクチャを排除します。

8.5.5.1.4 予測可能なパフォーマンスとコスト

- 下り料金、調整、または予測できない GPU キューはありません。
- 既知のハードウェア全体で最適化された I/O とコンピューティング。
- 総所有コストが明確で制御されています。

8.5.5.1.5 ライフサイクル管理されたインフラストラクチャ

- 共同 EDB + Supermicro サポート契約。
- ハードウェアとソフトウェアのパッチが統合および調整されました。
- 顧客主導のインフラストラクチャ管理または DevOps に依存しません。

8.5.5.2 サポートされているワークロードパターン

次のワークロードは、Sovereign Data および Agent Factory システムでエンドツーエンドでサポートされています。

8.5.5.2.1 高可用性 Postgres

- 自動フェイルオーバーを使用してミッションクリティカルなシステムオブレコードを実行します。
- 可観測性とライフサイクルが焼き付けられます。

8.5.5.2.2 ハイブリッド DBaaS

- 1 つのインターフェイスからマルチプルのクラスターを使用して内部チームにサービスを提供します。
- ロールベースのアクセス制御とクラスターレベルの可視性。

8.5.5.2.3 エンベディングとベクターパイプライン

- ローカルモデルを使用してデータをエンベディングに変換します。
- Postgres の pgvector を使用して保存および照会します。
- API キーまたはクラウドエンドポイントは必要ありません。

8.5.5.2.4 取得拡張生成 RAG

- 構造化内部データを照会し、LLM に渡します。
- 独自の境界内で取得と推論の両方をホストします。

8.5.5.2.5 オンプレミスの推論

- 専用の GPU ハードウェアで生成モデルおよび埋め込みモデルを提供します。
- 標準のコンテナ NVIDIA NIM などを使用して、モデル提供を抽象化します。

8.5.5.2.6 エージェント AI ワークフロー

- 内部ツールとデータを使用して、マルチステップのタスクを実行する AI エージェントをビルドします。
- 外部 API やデータ漏洩はありません。
- Slack、カスタムダッシュボード、または内部ツールにインターフェイスを展開します。

8.5.6 What to expect

ここでは、最初の 90 日間、配信からアクティブな使用とサポートまで、ソブリンデータとエージェントファクトリーシステムを受信、セットアップ、および操作する際に想定されることを示します。

8.5.6.1 0〜21 日目: サイトの準備とインストール

注文が確認されたら、サイトの準備状況アンケートに回答して、インストールをガイドします。調査には以下が含まれます。

- 電源、ラック、およびケーブル配線の詳細
- ネットワーキングとセキュリティ設定
- 物理的アクセスと連絡先の調整

8.5.6.1.1 何が起こるか

- Supermicro は、選択したデータセンターに完全なシステムを出荷します。
- EDB および Supermicro チームは、オンサイトのラッキング、電源投入、および検証を実行します。
- 設定に基づいて設定が完了します。
- Hybrid Manager ポータルへのログイン資格情報と URL アクセスを受け取ります。

8.5.6.2 21〜28 日目: 最初のワークロードの展開

システムがライブになると、すぐにワークロードの起動を開始できます。すべての操作は、Hybrid Manager ポータルを介して実行されます。

8.5.6.2.1 あなたにできること

- 高可用性 3 ノード Postgres クラスターを展開します。
- Postgres と pgvector を使用して、埋め込みジョブとベクターパイプラインを起動します。
- GPU ノードを使用して、 NVIDIA NIM マイクロサービスを介してローカルにモデルを提供します。
- RAG スタイルのアプリケーションと内部チャットボットを作成します。

Kubernetes に直接アクセスする必要はありません。HM ポータルを介してすべてを管理します。

8.5.6.3 28〜90 日目 ライフサイクル管理とスケーリング

ワークロードの進化に応じて、システムは EDB および Supermicro からの完全なライフサイクルサポートでスケーリングと更新の準備が整います。

8.5.6.3.1 含まれるもの

- Postgres や AI ツールを含む定期的な OS とソフトウェアの更新。
- コンピューティングまたは GPU ノードを追加することによるオプションの拡張。
- Supermicro の 4 時間のゴールド SLA によるハードウェアサポート。
- すべてのレイヤーにわたるビルトインの可観測性と診断。

ソフトウェアとハードウェアの両方が共同サポート契約の対象となっている間、ハイブリッドマネージャーを介してシステムの操作を継続できます。