

EDB Postgres AI Hybrid Manager

リリース *1.2*

EnterpriseDB Corporation

2025 年 06 月 25 日

目 次

第 1 章 EDB Postgres AI Hybrid Manager **3**
1.1 Hybrid Manager の主な機能 . . . 3

第 2 章 Hybrid Manager Release Notes **5**
2.1 Hybrid Manager 1.2 release notes . . . 5
2.1.1 リリースハイライト . . . 5
2.1.2 展開オプション . . . 5
2.1.3 新機能と機能強化 . . . 6
2.1.4 可観測性、モニタリング、インサイト . . . 7
2.1.5 サステナビリティ ESG レポートプレビュー . . . 7
2.1.6 プラットフォームの改善 . . . 8
2.1.7 リソース . . . 8

第 3 章 EDB Postgres AI Hybrid Manager **9**
3.1 主要な機能 . . . 9
3.2 統合ソリューション . . . 9
3.3 次のステップ . . . 10
3.3.1 Hybrid Manager Architecture . . . 10
3.3.2 Supported Database Types . . . 12
3.3.3 Supported Postgres Distributions . . . 19

第 4 章 Installing the PG AI Hybrid Manager **21**
4.1 PG AI Hybrid Manager のインストール . . . 21
4.2 計画 . . . 21
4.2.1 Kubernetes 汎用 . . . 21
4.3 プリインストールされたハードウェアソリューションを検討していますか? . . . 21
4.3.1 Kubernetes planning and minimum recommended requirements . . . 22
4.3.2 Install Hybrid Manager on Red Hat OpenShift . . . 29
4.3.3 Install Hybrid Manager on AWS EKS . . . 46
4.3.4 Install Hybrid Manager on GCP GKE . . . 80
4.3.5 Customizations . . . 96

第 5 章 Using Hybrid Manager **107**
5.1 タスク . . . 107
5.1.1 Hybrid Manager Terminology . . . 107
5.1.2 Hybrid Manager Console . . . 113
5.1.3 For Users . . . 121
5.1.4 For Kubernetes Administrators . . . 215

第 6 章 AI Factory in Hybrid Manager **321**
6.1 ラーニングパス . . . 321

6.1.1 Gen AI ワークロード . . . 321
6.1.2 モデルの管理とサービス . . . 322
6.1.3 ベクトルエンジン . . . 322
6.2 はじめる . . . 322
6.2.1 Enabling AI Factory . . . 322
6.2.2 Gen AI in Hybrid Manager . . . 327
6.2.3 Learn AI Factory in Hybrid Manager . . . 332
6.2.4 Model capabilities in Hybrid Manager . . . 343

第 7 章 Analytics in Hybrid Manager 349
7.1 ハイブリッドマネージャーでアナリティクスを使用する理由 . . . 349
7.2 コアコンセプトと機能 . . . 349
7.2.1 Apache Iceberg . . . 349
7.2.2 デルタレイク . . . 350
7.2.3 PGD を使用した階層テーブル . . . 350
7.2.4 レイクハウスクラスター . . . 350
7.3 これらの機能がどのように連携するか . . . 351
7.4 はじめに: クイックスタートチェックリスト . . . 351
7.5 さらに探索する . . . 352
7.5.1 Delta Lake in Hybrid Manager . . . 352
7.5.2 Apache Iceberg in Hybrid Manager . . . 354
7.5.3 EDB Postgres Lakehouse Clusters on Hybrid Manager . . . 357
7.5.4 Learning Analytics in Hybrid Manager . . . 359
7.5.5 Tiered Tables in Hybrid Manager . . . 396

第 8 章 Sovereign AI and Data Factory 401
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? . . . 401
8.2 中身は何ですか . . . 401
8.3 サポートされているワークロード . . . 402
8.4 ハードウェア構成 . . . 402
8.5 次に探索する . . . 402
8.5.1 Concepts and architecture . . . 402
8.5.2 Frequently asked questions . . . 405
8.5.3 Glossary and key terms . . . 406
8.5.4 System specifications . . . 408
8.5.5 Common workloads and use cases . . . 411
8.5.6 What to Expect . . . 413

EDB Postgres AI Hybrid Manager 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。

これは、次の Kubernetes ディストリビューション **Amazon EKS** 、**Google GKE** 、および **Red Hat OpenShift RHOS** で動作します。

第1章　EDB Postgres AI Hybrid Manager

1.1 Hybrid Managerの主な機能

Area	Capabilities
Database Management	Kubernetes 環境全体で PostgreSQL および分散 Postgres クラスターをプロビジョニング、アップグレード、および操作します。
AI Integration	NVIDIA NIM を使用してモデルを提供し、ナレッジベースを構築し、AI パイプラインを接続します。すべてが管理され観察可能です。
Analytics Tiering	コールドデータをオブジェクトストレージにプッシュし、Iceberg または Delta Lake を介して照会し、トランザクションと分析のユースケースを組み合わせます。
Kubernetes-Native Monitoring	ビルトインダッシュボード Prometheus、Grafana、Loki を使用して、リソースの使用状況、クエリ診断、アラート、およびクラスターの状態を追跡します。
Security and Access	ID プロバイダーに接続し、最小権限ロールを適用し、チーム全体でプロジェクトレベルのアクセスを管理します。

第2章　Hybrid Manager Release Notes

2.1 Hybrid Manager 1.2 release notes

発売日 2025 年 6 月 17 日

EDB Postgres AI Hybrid Manager v1.2 の最初の一般公開リリースを発表できることを嬉しく思います。

EDB Postgres AI Hybrid Manager は、ソブリン AI、分析、および Postgres 管理ソリューションである EDB Postgres AI プラットフォームのコントロールプレーンです。このリリースでは、高度な分析、生成 AI、および運用管理を単一の凝集プラットフォームに統合することにより、データの操作方法に革命をもたらします。

2.1.1 リリースハイライト

- **ソブリン AI ＆分析プラットフォーム**ハイブリッドマネージャーは、最新のデータと AI アプリケーションを構築するための完全なプラットフォームになりました。Postgres ネイティブのレイクハウスと、統合されたガバナンスを備えたソブリン GenAI ワークロードをすべて独自のインフラストラクチャ内で展開します。
- **新しい展開オプションソブリン AI と Data Factory** 外部を使用せずに独自のデータセンターにプロダクショングレードの Postgres および AI ワークロードを展開するための、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アプライアンスを紹介します。依存関係。
- **Postgres-Native Lakehouse** 画期的な分析スタックがビルトインされました。**レイクハウスクラスター**を展開して、オブジェクトストレージ内の **Apache Iceberg** や **Delta Lake フォーマット**のようなオープンデータフォーマットに対して直接高速のベクトル化クエリを実行します。
- **GenAI ワークロード** の AI ファクトリー ローコード **GenAI Builder** 、GPU アクセラレーションを使用して提供する統合モデル、データベース内**セマンティック検索**、および包括的な Python を使用して、強力な AI アプリケーションを構築して提供します。SDK
- **統合管理とプロアクティブな診断**完全に再設計されたモニタリングインターフェイス、新しいクラスターレベルのスコアカード、**プロアクティブなクエリの推奨事項**、および詳細なクエリレイテンシー統計を体験してください。

2.1.2 展開オプション

EDB Postgres AI Hybrid Manager v1.2 は 2 つの方法で利用でき、運用ニーズに最大限の柔軟性を提供します。

ソブリン AI と Data Factory ハードウェアアプライアンス

最大限の制御とゼロの外部依存関係を必要とする組織にとって、**ソブリン AI と Data Factory** は、完全に統合された事前に組み立てら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。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、データ主権、セキュリティ、ライフサイクルの信頼性が最優先される環境に理想的な選択です。

このエンジニアドシステムはスケール向けに最適化されており、インストールした瞬間に高度な Postgres、vector、AI パイプラインを実行する準備が整います。これには、検証された Supermicro サーバー、ライフサイクル管理される Kubernetes、および AI 推論用のオプションの GPU ノードを備えた完全なハイブリッドマネージャーソフトウェアスイートが含まれています。

ソフトウェアのみの展開

既存のインフラストラクチャを持つ組織の場合、EDB Postgres AI Hybrid Manager は、優先する Kubernetes ディストリビューションにソフトウェアのみのソリューションとして展開できます。**Amazon EKS** 、**Google GKE** 、および **Red Hat OpenShift RHOS** に対して認定されています。

2.1.3 新機能と機能強化

ソブリン AI ＆分析プラットフォーム

Hybrid Manager v1.2 は、Postgres プラットフォームで直接次世代の分析と AI ワークロードを構築および管理するための強力な統合ツールスイートを紹介します。

Postgres ネイティブアナリティクス

Postgres の能力とオブジェクトストレージのスケールを組み合わせた、最新の費用対効果の高いデータアーキテクチャを構築します。Hybrid Manager は、分析スタック全体を管理するツールを提供します。

- **レイクハウスクラスター**オブジェクトストレージでスケーラブルなベクトル化されたクエリ実行を提供する専用のコンピューティングクラスターをプロビジョニングおよび管理します。これらは、Postgres ネイティブのレイクハウスのクエリエンジンとして機能します。
- **Apache Iceberg のサポート** Iceberg 形式で大規模な分析テーブルをネイティブに管理します。 EDB Postgres 分散 PGD で **階層テーブル**を使用して、コールドデータパーティションを Iceberg に自動的にオフロードし、ストレージコストを削減しながら、レイクハウスクラスターを介してクエリ可能にします。
- **デルタレイクフォーマットの統合**既存のデータレイクとシームレスに統合します。Lakehouse クラスターは、Delta Lake オープンフォーマットを使用してテーブルを直接照会できるため、Spark のような他のビッグデータパイプラインによって生成されたデータの高速分析が可能になります。
- **PGD を使用した階層テーブル**テーブルを自動的にパーティション化し、コールドパーティションを Iceberg 形式のオブジェクトストレージにオフロードすることにより、大規模な時系列データセットを効率的に管理します。1 つのクエリで、Postgres のホットトランザクションデータとデータレイクのコールド履歴データの両方にまたがることができます。

AI Factory を使用した GenAI ワークロード

EDB Postgres AI Hybrid Manager v1.2 では、エンタープライズグレードの展開用に設計された包括的な GenAI 機能スタックである **AI Factory** が導入されています。

- **GenAI Builder** 単純なワンショットプロンプトから複雑な RAG パイプラインに至るまで、AI アプリケーションを迅速にプロトタイプ、反復、および展開するための Web ベースのローコードインターフェイス。
- **モデルサービス**可用性と自動スケーリングのビルトインサポートを備えた、GPU アクセラレーションインフラストラクチャでの AI 推論用の **NVIDIA NIM** コンテナを展開および管理します。
- **セマンティック検索** Postgres に組み込まれているベクトル検索機能を使用して、GenAI アプリケーションの取得を強化します。テキスト、HTML、PDF、または画像を取り込んで埋め込み、類似性検索を介して照会します。
- **AI 開発用の Python SDK** EDB AI Builder Framework SDK を使用すると、開発者は Python を使用して AI ワークフローを組み立てることができます。より少ないコード行で、取得、再ランキング、埋め込み、および生成応答を統合します。

2.1.4 可観測性、モニタリング、インサイト

v1.2 は、拡張されたメトリック、持続可能性の洞察、パフォーマンス診断によりプラットフォームの可観測性を強化します。

- **再デザインされたモニタリングダッシュボード**改良された UI、改善されたレイアウト、およびメトリックとしきい値のより詳細な制御により、クラスターの状態に関するより深い洞察が得られます。
- **クエリレイテンシーメトリック** `pg_stat_monitor` 拡張機能を介して収集されたクエリレイテンシー統計を [**Query Diagnostics**] タブで直接視覚化します。
- **プロアクティブなクエリの推奨事項**実際のワークロードパターンに基づいて、新しいインデックスの提案や古い統計アラートを含む最適化のヒントを受信します。
- **クラススコアカード**各クラスターの診断、推奨事項、およびリスク指標の統合された概要を 1 か所に表示します。
- **自己管理の Postgres サポート**外部 Postgres インスタンスをハイブリッドマネージャーに接続して、クエリ診断、推奨事項、およびモニタリングメトリックを収集します。

2.1.5 サステナビリティ ESG レポートプレビュー

EDB Hybrid Manager には、 **ESG ダッシュボードの実験サポート**が含まれ、組織がデータインフラストラクチャのエネルギー使用量と炭素排出量を追跡するのに役立ちます。

- **カーボンおよびエネルギーメトリック**構成可能な排出係数を使用して、CPU ベースのエネルギー使用量と推定 CO_2 排出量を視覚化します。
- **リージョンアウェアなレポート**展開地域に基づいて、ローカルエネルギーコスト$/kWh と排出率 kg CO_2/kWh を使用してダッシュボードをカスタマイズします。
- **統合された Grafana ダッシュボード** 新しい**電力使用量とカーボンフットプリント**ダッシュボードに、ハイブリッドマネージャー UI から直接アクセスします。

- **プレビューフェーズの目標** ESG オブザーバビリティの需要を検証し、関心のあるユーザーからフィードバックを収集します。

2.1.6 プラットフォームの改善

Area	Highlights
Deployment options	New Sovereign AI and Data Factory hardware appliance.
Analytics	Lakehouse clusters with Iceberg and Delta Lake query support.
AI application tooling	GenAI Builder, model serving, semantic search, and SDK.
Monitoring	Refreshed UI, scorecard view, extended query diagnostics.
Postgres coverage	Supports external/self-managed Postgres instances.
Sustainability	Preview of ESG metrics and dashboards for power and carbon tracking.

2.1.7 リソース

- *Installing Hybrid Manager on AWS EKS*
- *Using Hybrid Manager*
- *AI Factory Overview*
- Analytics Lakehouse Architecture
- *Power & ESG Reporting*
- Release Notes Archive

リリース日 2025 年 6 月 17 日

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 主要な機能

Area	Description
Database provisioning	UI、テンプレート、または宣言的マニフェストを使用して、可用性の高い 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 を使用した統合ダッシュボードは、クエリの状態、リソースの使用状況、システムアラートに対する深い可視性を提供します。
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。

3.2 統合ソリューション

ハイブリッドマネージャーは、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。

- **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する
- **Analytics** – 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する
- **Migration** – プラットフォーム内の移行ツールで Postgres の採用を加速する
- **Monitoring** – Kubernetes とデータベースの洞察を備えたビルトインオブザーバビリティスタック
- **Cluster Lifecycle** – テンプレート、アップグレード、HA 設定、および復元ワークフロー

3.3 次のステップ

- *Hybrid Manager* アーキテクチャにどのように適合するか
- *Supported Postgres Distributions*
- *Supported Database Types*
- *Install Hybrid Manager on AWS EKS*
- Use Hybrid Manager

3.3.1 Hybrid Manager Architecture

図の最も外側のコンテナは、EKS on AWS や GKE on GCP などの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、またはオンプレミスインフラストラクチャで実行されている Kubernetes クラスターを示しています。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。図の内部に向かって作業すると、注意すべき 3 つの主な論理グループがあります。

1. **Kubernetes コントロールプレーン** Kubernetes コントロールプレーンは、`kube-apiserver` 、`etcd` 、`kube-scheduler` 、`kube-controller-manager` 、クラウドセットアップの場合は `cloud-controller-manager` などのコア Kubernetes コンポーネントで構成され、2 つまたは 3 つの Kubernetes コントロールによって実装されますノード。

2. **コントロールコンピューティンググループ** HM のコントロールコンピューティンググループは、Grafana、Loki、Thanos、Prometheus などの HM のコア管理コンポーネントを実装する 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。可観測性のための Cert Manager、証明書管理のための Cert Manager、ネットワーキングのための 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。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。これは、図に示されているほど単純ではありません。必要に応じて、より多くの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あり、`kube-scheduler` が配置する少なくとも 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。写真ほど単純ではありません。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は 3 から増やすことができますが、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の構成が必要になる場合があります。

Control Compute Grouping のワーカーノードと、 および Data Compute Grouping のワーカーノードは両方とも、スナップショット SAN を使用するオンプレミス、または EKS で EBS を使用するか、Barman for the cloud を使用して S3 互換オブジェクトストアにバックアップされます。

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 の管理対象サービス 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

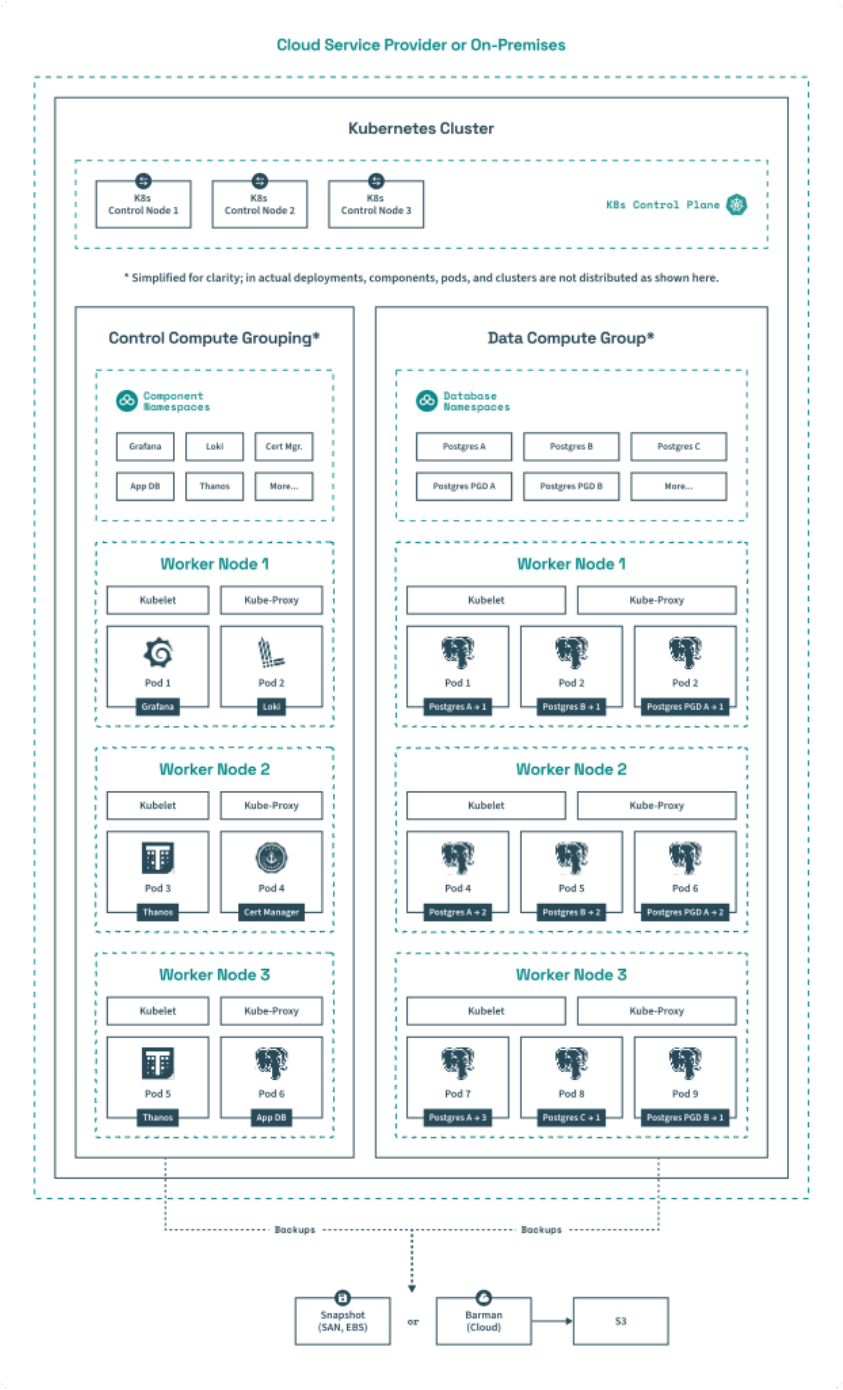


図 1: High-level diagram of Hybrid Manager’s architecture

に、オンプレミス展開用の AWS の Auto Scaling グループのようなすぐに使える自動スケーリング機能がないため、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することが重要です。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のモニタリングは、*included Grafana dashboards* または using Kubernetes `kubectl commands <Tracing clusters back to their Kubernetes resources>` を介して実現できます。

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復旧に最適ですが、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復旧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 Cloud バックアップを、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。

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 SAN ストレージエリアネットワークなどの高度なストレージ構成を使用するお客様の場合、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。

3.3.2 Supported Database Types

ハイブリッドマネージャーは、4 つのクラスタータイプをサポートしています。

- *Single node*
- **高可用性と復元力**
- *Advanced high availability*
- *Distributed high availability*

Create Cluster ページで、必要なクラスターのタイプを選択します。

Single-node cluster

高可用性が主要な懸念事項ではない非運用ユースケースの場合、高可用性が有効になっ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。



図 2: Single node cluster

High availability

プライマリ/スタンバイの高可用性オプションは、障害が発生した場合のダウンタイムを最小限に抑えるために提供されます。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。

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。

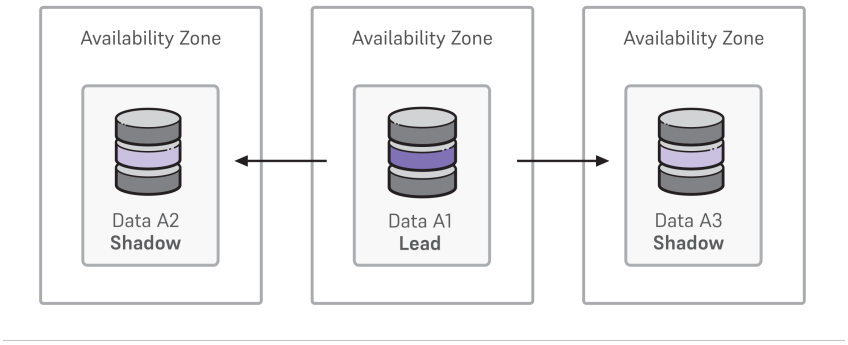


図 3: Primary/standby high availability

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。

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。

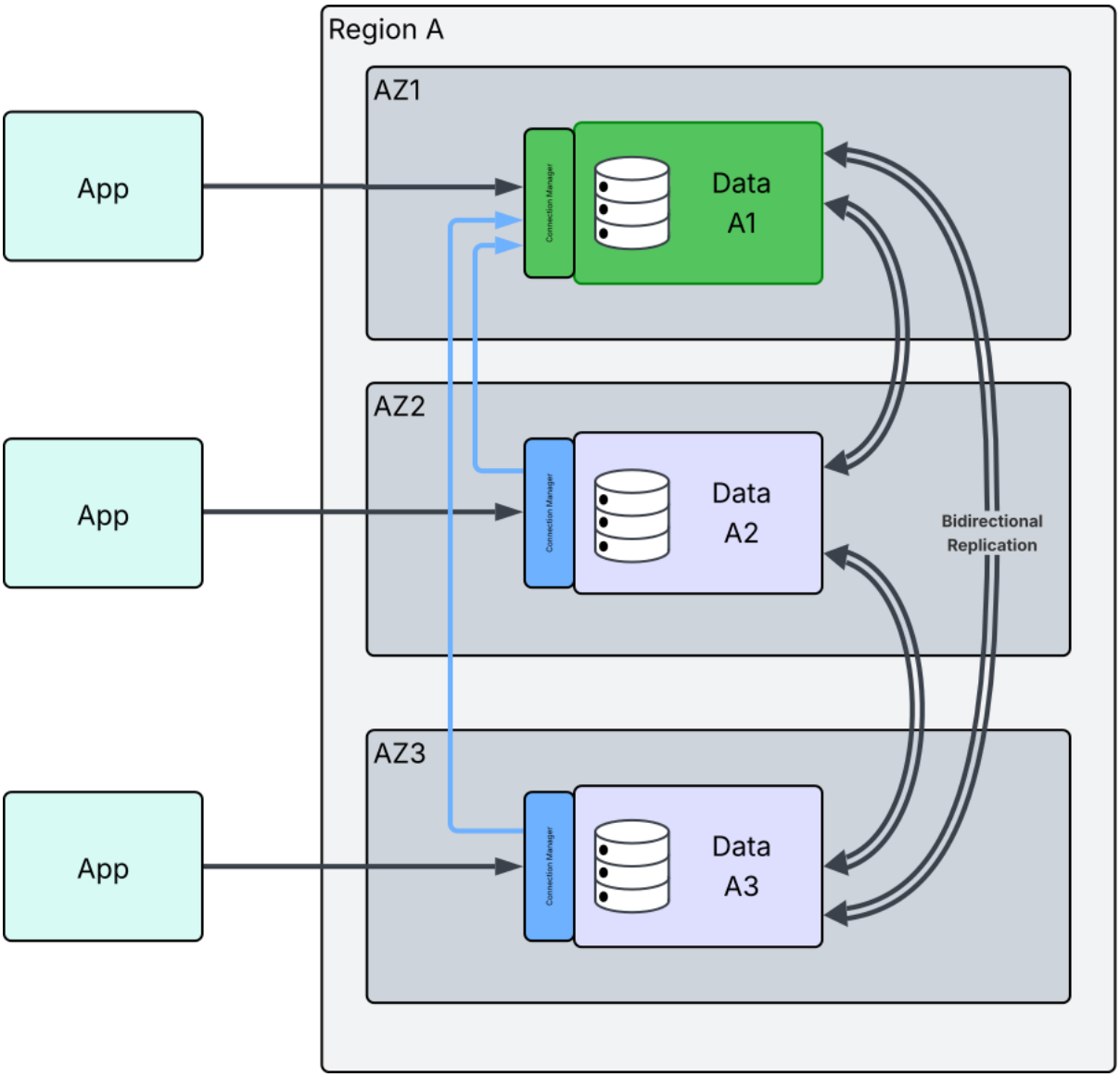
Advanced high availability

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。*Distributed high availability* 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、そのリージョン内で高可用性を提供するように設計されています。

これらは、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おり、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。

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。



width

70% :alt: Advanced high-availability cluster architecture

Advanced high-availability cluster architecture

この図では、接続マネージャーポートを介して PGD クラスターに接続しているアプリケーションが表示されます。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。

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と比較して、より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大きな懸念事項であるユースケースの場合、分散高可用性を有効にしたクラスター展開は、3 番目のリージョンに監視グループを持つ 2 つのデータ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、分散データの競合を削減します。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。

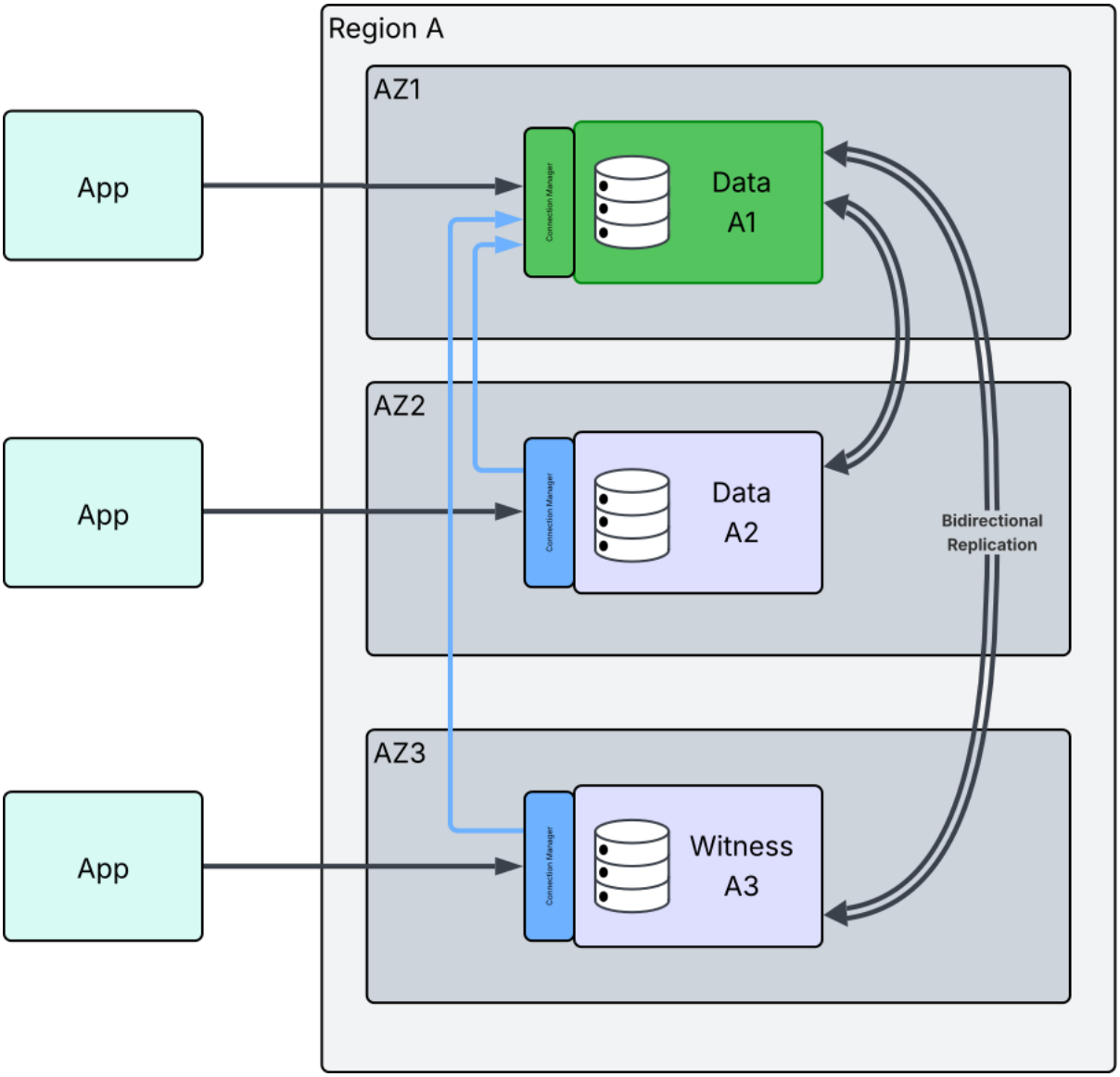
単一のデータの場所

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。

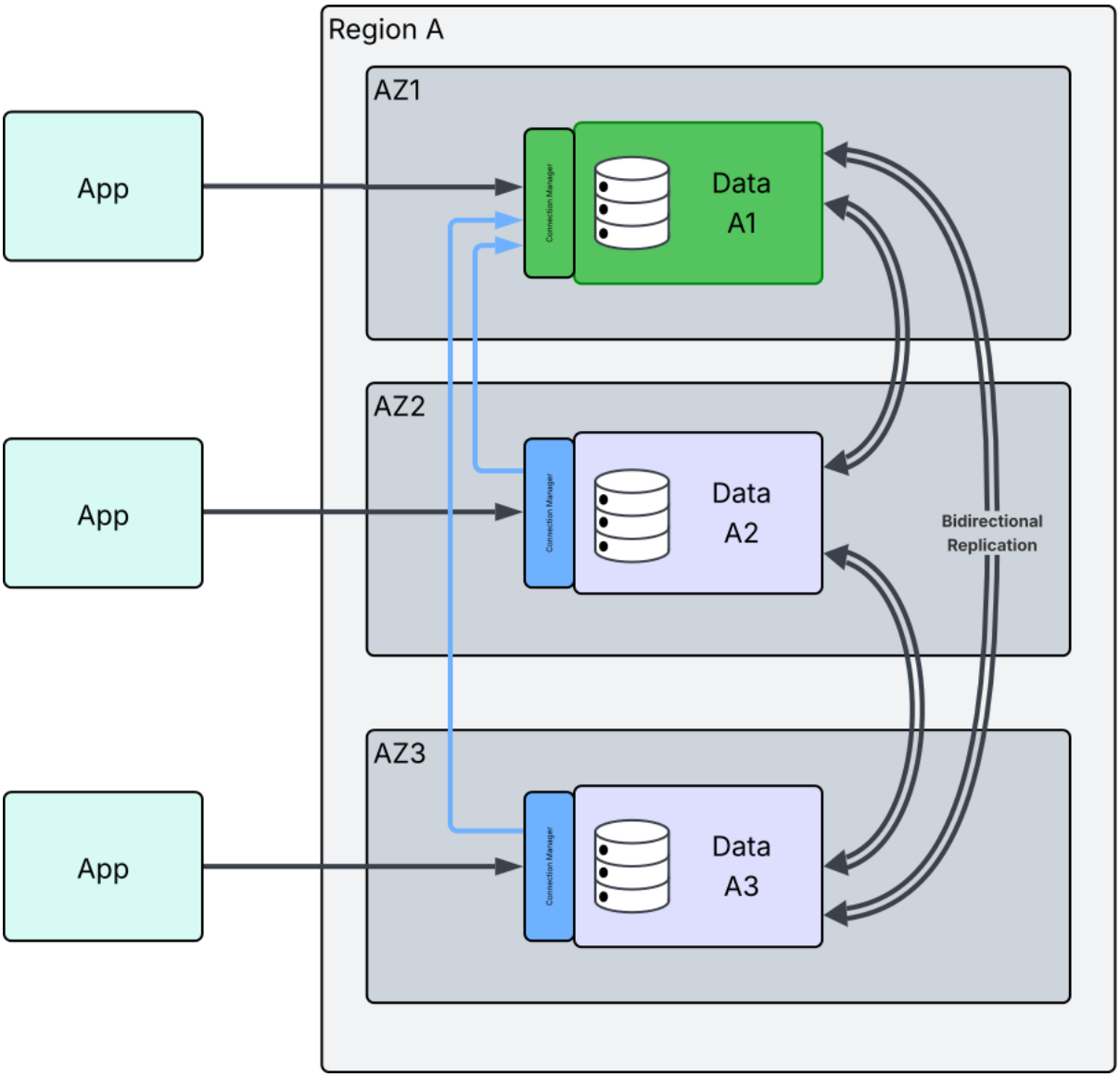
- 1 つのリード、1 つのシャドウ、および監視ノードを備えた 2 つのデータノード、それぞれが別のアベイラビリティーゾーンにある

  width
  : 70% :alt: region(2 data + 1 witness)

    region(2 data + 1 witness)


Region A
AZ1
App
Connection Manager
Data
A1
AZ2
App
Connection Manager
Data
A2
Bidirectional
Replication
AZ3
App
Connection Manager
Witness
A3

- 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



width

70% :alt: region(3 data)

region(3 data)

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、次のいずれかを含む 2 つのデータグループがあります。

- 各グループ内の 3 つのデータノード *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。

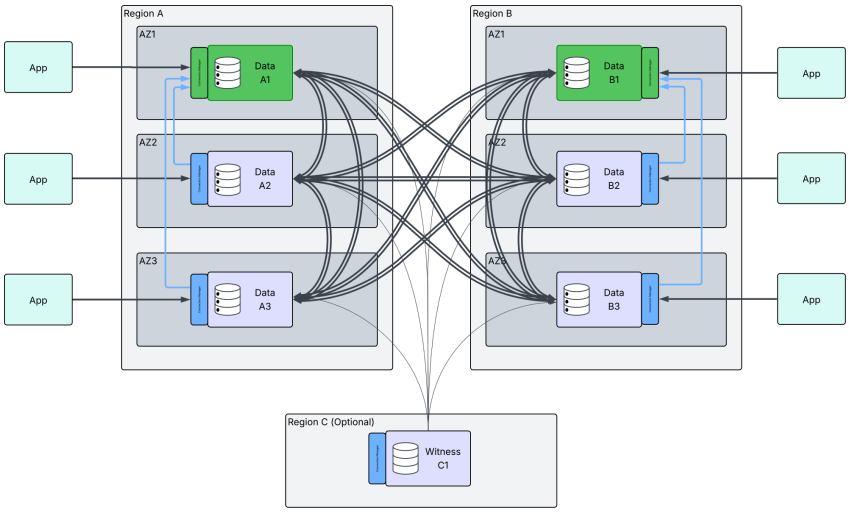


図 4: groups(2 x 3 data + 1 witness)

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、リード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。

3.3.3 Supported Postgres Distributions

PostgreSQL

PostgreSQL は、 *PostgreSQL documentation* に収集されたすべての機能を提供します。

EDB Postgres Extended Server

EDB Postgres Extended Server は、 PostgreSQL に多くの機能を追加して、transparent data encryption をサポートします。

EDB Postgres Extended Server は、EDB の強力な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 EDB Postgres Distributed と組み合わせて、完全に利用することもできます。

EDB Postgres Advanced Server

EDB Postgres Advanced Server は、 EDB Postgres Extended Server の機能をベースに構築され、 advanced security 、advanced SQL 、オプションで Oracle compatibility などの機能をサポートしています。

第4章　Installing the PG AI Hybrid Manager

4.1 PG AI Hybrid Manager のインストール

PG AI Hybrid Manager HM は、クラウドおよびオンプレミスの展開全体で Postgres、分析、および AI ワークロードのオーケストレーション、オブザーバビリティ、およびライフサイクルオペレーションを提供します。

このセクションでは、サポートされている Kubernetes プラットフォームにわたって、独自のインフラストラクチャにハイブリッドマネージャーを展開するユーザーを対象にインストールガイダンスを提供します。ソブリン *AI* と *Data Factory* ハードウェアアプライアンス のようなターンキーハードウェアシステムを使用している場合、Hybrid Manager はプリインストールされており、セットアップは必要ありません。

4.2 計画

4.2.1 Kubernetes 汎用

このガイドを使用して、ハイブリッドマネージャー用のカスタム Kubernetes クラスターを準備します。

- Planning your Kubernetes cluster for HM installation

これには、サイジングの推奨事項、ストレージクラスの予想、およびノードグループのセットアップが含まれます。

4.3 プリインストールされたハードウェアソリューションを検討していますか?

Hybrid Manager とそのサポートインフラストラクチャのフルマネージド展開を探している場合は、EDB が提供するターンキーハードウェアソリューションである ソブリン *AI* と *Data Factory* ハードウェアアプライアンス を検討してください。

このシステムには以下が含まれます。

- ハイブリッドマネージャーと Redhat Openshift のプリインストールされたエンドツーエンドのサポート
- ハードウェアとソフトウェアの完全なライフサイクル管理
- Postgres、ベクターワークロード、および GPU 駆動の推論のビルトインサポート

詳細については、 Engineered Systems overview をご覧ください。 —

4.3.1 Kubernetes planning and minimum recommended requirements

このドキュメントでは、ハイブリッドマネージャーをインストールするために Kubernetes クラスターを計画するための一般的なガイダンスと、クラスターの最小推奨仕様の両方を提供します。

- *General guidance for planning your Kubernetes cluster*
- *Minimum recommended specifications for Kubernetes clusters*

Planning Your Kubernetes Cluster for Hybrid Manager Installation

ハイブリッドマネージャー HM を展開する準備をするときは、最初にデザインの次の予備的な側面を検討してください。

1. ユースケース

- どのようなワークロードを管理する必要がありますか?
- 実稼働環境、テスト環境、またはハイブリッド環境で展開していますか？

2. ベースインフラストラクチャ

- クラスターはクラウドサービスプロバイダー CSP またはオンプレミスによってホストされますか? - オンプレミスの場合、ベアメタル、VM、またはプライベートクラウドオプションを検討します。

3. ネットワーキング

- アベイラビリティーゾーンとサブネットパブリック/プライベート
- NAT ゲートウェイ/下り
- ネットワークセキュリティルール
- リソースのタグ付けとレイテンシーの考慮事項
- **ネットワークセキュリティ**
  - クラスターネットワークポリシー
    - Kubernetes ネットワークポリシーを実装して、ポッド、名前空間、および外部リソース間のトラフィックフローを制御します。
  - 暗号化された通信
    - HM コンポーネントと外部クライアント間のすべての通信に TLS を使用します。
    - クラスター内サービス通信を保証しますは暗号化されています b_tran_7 ファイアウォールとセキュリティグループ
    - オンプレミス展開の場合、Kubernetes API サーバーとコントロールプレーンノードへのアクセスを制限するようにファイアウォールを構成します。
    - クラウドベースの展開の場合、セキュリティグループを使用してクラスターへのインバウンドおよびアウトバウンドトラフィックを制限します

4.Kubernetes クラスター設定

- Kubernetes バージョン。
- ID 管理 LDAP、SAML、OpenID Connect など。
- リソースタグ、シークレット/構成マップ。
- ノードグループのテイントと許容範囲。

5. ノードグループ構成

- インスタンスタイプとコンピューティンググループ。

6. アクセス許可ポリシー

- PGAIHM には、Kubernetes クラスター全体への権限が必要です
- ノードロールの権限と RBAC 設定。
- クォータと制限範囲。
- ネットワークポリシー。

7. ストレージ統合

- CSI ドライバーとオブジェクトストレージ S3 互換など。
- ボリュームタイプ、永続ボリューム要求、およびスナップショットのサポート。

8. ロードバランサー/イングレスの統合

- クラウドプロバイダーまたはオンプレミスの詳細。
- ロードバランサー統合のための権限。

9. データセキュリティ

```
- Encryption at rest
    - Ensure persistent volumes used by HM have encryption enabled
    - Use a storage backend that supports encryption (e.g. AWS S3 for backups in EKS,␣
↪NVMe with encryption for local storage on-premises)

- Encryption in transit
    - Enable encryption for all data transfer between HM components and the database␣
↪clusters they manage.

- Secrets management
    - Store all sensitive information (e.g. database passwords, API keys) in␣
↪Kubernetes Secrets.
    - Use a secure external secrets manager (e.g. HashiCorp Vault, AWS Secrets␣
↪Manager) when possible.
```

10. プラットフォーム固有のカスタム構成

- Red Hat OpenShift、AWS EKS、およびその他のディストリビューションの違い。

これらの考慮事項に対処することで、Kubernetes クラスターがあらゆる展開シナリオで HM をサポートするように適切に構成されていることを確認できます。

Minimum recommended specifications for the Hybrid Manager

一般的な推奨リソース要件

PGAIHM を実行するように Kubernetes クラスターを構成する場合、最小推奨される一般リソース要件は次のとおりです。

新しい Kubernetes クラスター

既存の Kubernetes クラスターでの PGAIHM の実行は、現時点ではサポートされていません。したがって、PGAIHM をホストするために、新しい Kubernetes クラスターを作成する必要があります。

コンピューティング

ハイブリッドマネージャー HM のコンピューティング要件を検討する場合、PGAIHM 自分自身はすぐに比較的軽量であり、スケーリングが必要になる可能性があるさまざまなユースケースのワークロードのみであることを理解することが重要です。

PGAIHM の最小要件は、1 つのマシン上の 16GB の RAM、4 つの vCPU、および 50GB のディスクです。これらは、インストール時に PGAIHM が消費するリソースです。これは、PGAIHM のテストのみに推奨され、実稼働使用には推奨されません。

ワークロードの処理

一般的な運用環境での使用の場合、1〜20 の Postgres クラスターワークロードのサイズに応じて、次のコンピューティングの推奨事項が提案されます。

- **6 Kubernetes ワーカーノード**、HM 制御コンポーネント用に 3 つ、HM データベースノード用に 3 つ、それぞれに次のものが含まれます。
  - 8+コア
  - 32 GB のメモリ
  - 64 GB のストレージ

大規模なデータワークロードの場合、より多くのワーカーノードが必要になる可能性があります。

AI

AI ワークロードを実行する予定がある場合は、少なくとも以下を追加します。

- 1 GPU ノード、16〜32 コア、30 Gi ストレージ領域

ただし、AI ワークロードが非常に大きい場合、または多数がある場合、より多くの GPU リソースが必要になる可能性があります。特定のモデルのニーズに必要な正確な仕様を確認するには、NVIDIA documentation を参照してください。

オブジェクトストア

- **S3 互換オブジェクトストア**
  - Ceph 、 MinIO 、
  - EKS 展開の場合 AWS S3

ストレージレイヤー

- 永続ボリュームの動的プロビジョニング用の**コンテナストレージインターフェイス** CSI ドライバー compatible with Kubernetes 永続ボリュームの動的プロビジョニング用
  - オンプレミス展開の OpenShift Container StorageOCS または Rook-Cephs
  - EKS 用の AWS EBS CSI ドライバー
- **ストレージクラス**
  - 動的ボリュームプロビジョニングで構成された少なくとも 1 つのデフォルトのストレージクラス
  - オンプレミス展開の場合、最適なパフォーマンスを得るには NVMe をお勧めします。たとえば、Supermicro SYS-221H-TNR のようなシステムは、高速ストレージを必要とする展開に適しています。
  - SAN
    - HM の集中化された高性能ニーズにはストレージエリアネットワーク SAN を使用する必要があります。
    - SAN が保存時の暗号化をサポートし、経由で Kubernetes と統合していることを確認します CSI ドライバー.
    - SAN トラフィック用の専用で安全なネットワークを提供します。
  - スナップショットをサポート b_tran_9 HM でスナップショットバックアップを使用する場合、CSI ドライバーがボリュームスナップショットをサポートしていることを確認します

構成設定

Kubernetes で HM を実行するには、次の構成設定をお勧めします。

- Kubernetes バージョン 1.20 以降。

ネットワーク設定

- **ポッドネットワーク**
  - ポッド IP の最小 CIDR レンジは、クラウドサービスプロバイダーまたはオンプレミスのニーズによって異なります。
    - 例 AWS EKS の場合は/22 以上をお勧めします
- **サービスネットワーク** - Kubernetes サービスの最小 CIDR レンジは、CSP またはオンプレミスのニーズによって異なります。

- **外部アクセス** - インターネットを使用して HM にアクセスする場合、これらのイングレス/エグレス要件が適用されます
- **DNS** - クラスター全体の DNS サービスが動作している必要があります。

セキュリティ設定

- **アクセス制御**

```
- Ensure that Kubernetes RBAC is configured with the principle of least privelege.␣
↪Users, services, and pods should only have access to the resources they need.
```

- **認証**

```
- Use secure authentication mechanisms such as integration with an identity provider␣
↪(IDP)
    - LDAP, SAML, or OpenID Connect
- Multi-factor Authentication (MFA)
    - Enforce MFA for all administrator accounts accessing the HM.
```

- **ネットワークセキュリティ**
  - クラスターネットワークポリシー
    - Kubernetes ネットワークポリシーを実装して、ポッド、名前空間、および外部リソース間のトラフィックフローを制御します。
  - 暗号化された通信
    - HM コンポーネントと外部クライアント間のすべての通信に TLS を使用します。
    - クラスター内サービス通信を保証しますは暗号化されています b_tran_7 ファイアウォールとセキュリティグループ
    - オンプレミス展開の場合、Kubernetes API サーバーとコントロールプレーンノードへのアクセスを制限するようにファイアウォールを構成します。
    - クラウドベースの展開の場合、セキュリティグループを使用してクラスターへのインバウンドおよびアウトバウンドトラフィックを制限します
- **データセキュリティ**
  - rest
    - HM が使用する永続ボリュームの暗号化が有効になっていることを確認します
    - 暗号化をサポートするストレージバックエンドを使用します例、EKS のバックアップ用の AWS S3、オンプレミスのローカルストレージの暗号化を備えた NVMe
  - transit
    - HM コンポーネントと、それらが管理するデータベースクラスター間のすべてのデータ転送の暗号化を有効にします。
  - シークレット管理

    - すべての機密情報たとえば、データベースパスワード、API キーを Kubernetes に保存します。
    - 安全な外部シークレットマネージャーを使用しますたとえば、HashiCorp Vault、AWS Secrets Manager) 可能な場合。

- **バックアップセキュリティ**
    - Barman Cloud またはスナップショットに保存されているバックアップデータが暗号化されていることを確認します。
    - バックアップストレージに RBAC ポリシーを使用して、権限のある担当者とサービスへのアクセスを制限します。
    - バックアップ復元プロセスを定期的にテストして、安全で効果的であることを確認します。
- **モニタリングと監査**
    - 監査ログ s
        - 管理アクションと異常なアクティビティの Kubernetes 監査ログを有効にして監視します
    - ログ集計
        - 統合された Grafan および Loki スタックを使用して、セキュリティイベントのログを一元化および分析します。
    - アラート
        - 次のような重要なイベントのアラートを構成します認証の失敗、不正なアクセス試行、またはリソースポリシー違反。
- **コンテナセキュリティ**
    - 信頼できるソースからのコンテナイメージを使用し、脆弱性について定期的にスキャンします Trivy や Clair などのツールを使用して特権エスカレーションを無効にします。
        - ポッドセキュリティコンテキストで `runAsNonRoot` を設定します
- **Kubernetes クラスターセキュリティ**
    - Kubernetes API サーバーへの匿名アクセスを無効にします。
    - Kubernetes とすべての関連コンポーネントを定期的にパッチおよび更新します。
- **物理セキュリティ** オンプレミス展開の場合
    - SAN Security
        - SAN ハードウェアへの安全な物理的アクセス。制限されたアクセス制御されたデータセンターにあることを確認します。
        - 重要な SAN コンポーネントの冗長化とフェイルオーバー構成を実装して、高可用性.
        - SAN ネットワークトラフィックを分離して、不正なアクセスを防止し、傍受のリスクを削減します。
- **コンプライアンス**

- HM 展開が組織または業界固有のセキュリティ標準 SOC 2、HIPAA、GDPR などに準拠していることを確認します。
- 定期的にコンプライアンスを確認し、標準の進化に応じて構成を更新します。

HM 機能またはユースケースに基づいたリソースに関するガイダンスについては、キャパシティ計画ドキュメントを参照してください。

2. Kubernetes プラットフォームとバージョンの要件

HM は、さまざまな適合した Kubernetes ディストリビューションで実行されるように設計されています。サポートされているプラットフォームには次のものが含まれます。

- Amazon Elastic Kubernetes Service EKS
- Google Kubernetes EngineGKE
- Red Hat OpenShift RHOS

注特定のバージョンの互換性は、HM リリースノートと比較して確認する必要があります。

3. Kubernetes ノードとプロセッサ要件

3.1. ノードの特性

- **オペレーティングシステム** ノードは Linux 互換である必要があります。
- **プロセッサアーキテクチャ** HM コンポーネントは、主に AMD64 / x86-64 アーキテクチャをターゲットとしています。PostgreSQL 自分自身は ARM64 で実行できますが、HM 展開の混合アーキテクチャ Kubernetes クラスターは標準ではなく、サポートが複雑になる場合があります。

3.2. 最小環境レイアウト

一般的な最小の実稼働可能環境は次で構成されます。

- **場所** 単一の都市。
- **インフラストラクチャ**単一の物理データセンターオンプレミスの場合、またはクラウドサービスプロバイダー CSP の単一リージョン。
- **Kubernetes クラスター** 1 つの Kubernetes クラスター。
- **HM のノードの割り当て** - HM コントロールプレーンコンポーネントをホストし、高可用性を保証するには、少なくとも 3 つの Kubernetes ノードが必要です。これらは、適切にラベル付けされた Kubernetes コントロールプレーンノードまたはワーカーノードです。
- **PostgreSQL のノードの割り当てオプショナルただし推奨** - 厳密な HM 要件ではありませんが、リソースとラベルが許可する場合に PostgreSQL はコロケーションできます。次の場合、PostgreSQL インスタンス用に少なくとも 3 つの追加の Kubernetes ワーカーノードを使用することをお勧めします。運用ワークロードを調整して、リソースの分離とパフォーマンスを保証します。

4. 外部サービスの要件

Kubernetes 内で実行される HM は、Kubernetes クラスターにプロビジョニングしてアクセスする必要があるいくつかの外部サービスに依存しています。

- **ネットワーキング**
  - ローカルネットワークインフラストラクチャオンプレミスまたは CSP の仮想プライベートクラウド VPC。
  - **ロードバランサー** HM および PostgreSQL サービスを公開するために必要です。これは通常、CSP の管理対象サービスです。オンプレミスの RHOS の場合、MetalLB またはエンタープライズロードバランサーのようなソリューションを統合する必要があります。
- **DNS:** クラスター内および外部のサービス検出のための信頼できる DNS 解決。サービス公開のための DNS レコードの制御が必要です。
- **ID プロバイダー IDP** オプション、 *integrating with existing authentication systems* の場合。
- **オブジェクトストレージ** S3 互換のオブジェクトストレージソリューションは、バックアップおよびその他のプラットフォーム機能に必須です。これは CSP の管理対象サービスです。オンプレミスの場合、互換性のあるソリューションをプロビジョニングし、RHOS にアクセスする必要があります。
- **キー管理システム KMS** 暗号化キーを管理するためのオプション。
- **ブロックストレージ** PostgreSQL データボリューム、および HM.
  - の他のステートフルコンポーネントには、永続的な高性能ブロックストレージが必要です。
  - CSP では、これは通常、AWS EBS、GCP Persistent Disk、または Azure Disk Storage のようなサービスです.
  - オンプレミス展開の場合、ストレージエリアネットワーク SAN または同等の高性能ストレージソリューションが必要です。CSI ドライバーを介して Kubernetes と統合されています。この状況では、SAN は CSP ブロックストレージサービスと機能的に同等であると考えられます。

5. 依存する外部サービス接続

HM コンポーネントと EDB Postgres インスタンスを実行している Kubernetes クラスターノードには、次の EDB サービスへのアウトバウンドインターネットアクセスが必要です。

- **EDB Container Registry** コンテナイメージをプルするため。 - エンドポイント docker.enterprisedb.com
- **EDB Entitlements Service** ライセンス検証とサービス資格。 - エンドポイント biganimal.com

4.3.2 Install Hybrid Manager on Red Hat OpenShift

Red Hat OpenShift RHOS に Hybrid ManagerHM をインストールできます。

- 前提条件 を実行します。
- 2 つのインストール方法の違いを理解する *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator*
- EDB Postgres AI Operator を使用してインストールを実行します。

- *Installing Hybrid Manager on RHOS using EDB Postgres AI Operator*
- *Update your DNS zone records*
- *Connecting to Hybrid Manager* によるインストールの検証

Red Hat Open Shift installation prerequisites

Red Hat Open SHIft には、EDB Postgres AI Hybrid ManagerHM をインストールする前に満たす前提条件があります。インストールプロセスを開始する前に、前提条件を満たす必要があることを確認します。

前提条件

- *A system from which you can run the installation process*
- *A RHOS cluster configured for use with HM*
- *Necessary environment variables set*

次のステップ

これらの要件を満たしたら、 *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator* に進むことができます。

Install system

インストールプロセスを実行できるシステムが必要です。このシステムには、次のソフトウェアをインストールする必要があります。

- OpenShift CLI (oc)
- Kubernetes クラスターを管理するための kubectl
- yq Hybrid Manager インストールファイルの準備

リンクされた手順、または以下にあるオペレーティングシステムの手順を使用して、これらのツールをインストールできます。

macOS

```
brew install kubectl yq
```

Linux

```
sudo apt install kubectl
sudo snap install yq
```

```
sudo yum install kubectl
sudo wget https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64 -O /
↪usr/bin/yq && sudo chmod +x /usr/bin/yq
```

```
sudo apt-get install kubectl
sudo wget https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64 -O /
↪usr/bin/yq && sudo chmod +x /usr/bin/yq
```

Windows

```
winget install kubectl --source winget
winget install yq --source winget
```

1. kubectl をインストールします。
2. yq をインストールします。

次のステップ

必要なソフトウェアをインストールしたら、 *setting up aa RHOS cluster* に進むことができます。

Creating an Red Hat OpenShift cluster for use with Hybrid Manager

installed your system が完了したら、Hybrid ManagerHM で使用する Red Hat OpenShiftRHOS クラスターを作成する準備が整います。

HM インストールのための OpenShift クラスターの準備

Red Hat OpenShiftRHOS オンプレミスクラスターに HM を正常にインストールするには、特定の前提条件と構成を満たしている必要があります。

事前構成のクラスター要件

- HM は現在以下をサポートしています
- OpenShift 4.17 with Kubernetes 1.30

または

- OpenShift 4.18 with Kubernetes 1.31

ステップ 1 ネットワークを構成する

- ネットワークポリシーを定義します。Kubernetes コントロールプレーン、ワーカーノード、外部システム間のトラフィックを管理するために、適切なネットワークポリシーが設定されていることを確認します。
- イングレス/エグレスルールを定義します。
- クラスターコンポーネントと HM スタック間の通信が安全であることを確認します。- 入力/出力ネットワークポリシーを検証して、不要な外部アクセスを制限しながら、クラスターコンポーネントが通信できるようにします。
- ロードバランサーのセットアップ外部ロードバランサーを使用する場合、OpenShift API トラフィックとワーカーノード通信を処理するように構成します。

- DNS の構成 HM の内部サービス検出を含む、クラスターに DNS レコードが設定されていることを確認します。
- クラスターにワーカーノード用の適切なプライベートネットワーク分離があることを確認します。OpenShift は通常、ソフトウェア定義のネットワーク SDN 構成を介してこれを処理します。

ステップ 2 ノードサイズとリソースを構成する

HM とその関連コンポーネントをサポートするには、OpenShift クラスター内のノードがリソース要件を満たしている必要があります。

ノードのサイジング

- **マスターノード** OpenShift コントロールプレーンノードは、リソース割り当ての標準要件 CPU、メモリ、ディスクサイズなどを満たしている必要があります。
    - 一般的な推奨事項
        - 3 つのマスターノードそれぞれに次のものが含まれます。
            - CPU
                - 最小値 4 つの vCPU 最大 10 のワーカーノードの場合
                - 推奨 中規模のクラスター 10〜50 のワーカーノード用の 8 vCPU
                - 50 ワーカーノードの場合 16+ vCP Us
            - メモリ
                - 最小 32 GB RAM 最大 10 の場合ワーカーノード)
                - 推奨 大規模なワークロード 10〜50 ワーカーノードの場合は 64 GB RAM 以上
                - 50 ワーカーノードの場合 64+ GB RAM
            - Disk
                - 最小 100 GB SSD
                - 推奨 大規模なワークロードの場合は 200 GB SSDs
                - 50 ワーカーの場合ノード >200 GB SSD
- **ワーカーノード** HM 制御コンポーネントと Postgres クラスターを処理するために十分なリソースをワーカーノードに割り当てます。小規模なワークロードには、HM 制御コンポーネント用に 3 つのワーカーノードと Postgres ワークロード用の 3 つのワーカーノードをお勧めします。5〜20 の Postgres クラスター)。ここから Postgres ワークロードをスケールアップすると、Postgres ワークロード用により多くのワーカーノードが必要になり、最終的にはさらに多くの制御ワーカーノードが必要になる場合があります。増加する Postgres ワークロードワーカーノード。
    - 一般的な推奨事項

6 **つのワーカーノードそれぞれが以下を備えています。** CPU: **最小** *4 vCPU* **これは、ワークロードの種類によっては高くなる場合があります** メモリ: **最小** *32 GB* **の** *RAM /* **ノードごとに** ディスク * 最小 100 GB の永

続ストレージノードごとにデータベースとロギングの要件に基づいて調整します。最適なパフォーマンスを得るには、高速ディスク SSD を使用します。

HM 制御ノードと Postgres ワークロードをホストする HM Postgres データノードに異なるタイプのノードをオプションで使用するには、RHOS に追加のマシンセットが必要です。

- 制御ノードマシンセット

```
spec:
    replicas: 3
    template:
    spec:
        metadata:
        labels:
            edbaiplatform.io/control-plane: "true"
    taints:
    - key: edbplatform.io/control-plane
        value: "true
        effect: NoSchedule
```

- Postgres ワークロードマシンセットをホスティングするデータノード

```
spec:
    replicas: 3
    template:
    spec:
        metadata:
        labels:
            edbaiplatform.io/control-plane: “true”
        taints:
        - key: edbplatform.io/control-plane
            value: "true"
            effect: NoSchedule
```

リソース予約

- HM ワークロードとの競合を防ぐには、システムコンポーネント用に CPU とメモリを予約します。
- HM モニタリングおよび可観測性ツールたとえば、Grafana、Loki、および Prometheus に追加リソースを割り当てます。
- ノードプール構成
    - 必要に応じて、テイントと寛容を使用して、特定のノードで HM ワークロードを分離します。
    - 特定のタスクたとえば、「モニタリング」、「データベース」のノードにラベルを付けて、リソース割り当てを最適化します。
- スケーリング

- 自動スケーリングを設定して、クラスターが需要の増加を動的に処理できるようにします。OpenShift のホリゾンタルポッドオートスケーラー HPA を使用して、必要に応じてポッドをスケーリングできます。

ネットワーキング帯域幅

各ノードに、HM 通信と外部データ転送 S3 バックアップなどを処理するための十分なネットワーク容量があることを確認します。

ベースラインの推奨事項

Component	Recommended bandwidth	Justification
K8s control plane nodes	1 Gbps+ per node	Handles internal Kubernetes traffic, API server requests, logs/metrics, and orchestration tasks.
Worker nodes (HM)	1 Gbps+ per node	For HM control components, PostgreSQL replication, internode communication, and metrics/logs.
External bandwidth	1-20 Gbps (aggregated)	S3 backups and inter-cluster replication may require high throughput.

ステップ 3 必要な CSI ドライバーをインストールする

HM は、CSI ドライバーに依存するストレージおよびスナップショット機能を使用します。次のドライバーが OpenShift クラスターにインストールされ、適切に構成されていることを確認します。

- 永続的ストレージドライバー ストレージバックエンドたとえば、OpenShift Container Storage、Ceph、またはその他のオンプレミスソリューションなどと互換性のあるストレージクラスドライバーをインストールします。
- スナップショットコントローラボリュームスナップショット管理用に CSI スナップショットコントローラを構成します。

ステップ 4 ストレージクラスを構成する

デフォルトのストレージクラス HM ワークロードに最適化されたデフォルトのストレージクラスを設定します。

カスタムストレージクラス特定のワークロードでカスタム構成が必要な場合、追加のストレージクラスを作成します。

ステップ 5 名前空間とシークレットを準備する

HM の名前空間 HM コンポーネントの専用の名前空間を作成して、分離と管理性を保証します。

```
``` bash oc create ns edbpgai-bootstrap
```

``````

**シークレット**オブジェクトストレージ資格情報、データベースアクセストークン、またはその他の機密情報など、必要な資格情報の Kubernetes シークレットを作成します。以下は、HM 固有のシークレットと構

成です。最初の ImagePullSecret は、HM が適切に機能するために必要です。カタログのグリップテープと DeltaLake オブジェクトストレージとシークレットは、GenAI Builder またはカタログのどちら用にクラスターを設定するかによって異なるため、オプショナル。

- **ImagePullSecret と名前空間の構成** HM に必要

1. 環境変数を設定します。

OC Registry にもっと簡単にログインできるように、設定するコンテナレジストリ変数がいくつかあります。

```
export CONTAINER_REGISTRY_URI=<local-registry-uri>
export CONTAINER_REGISTRY_USERNAME=<container-registry-username>
export CONTAINER_REGISTRY_PASSWORD=<container-registry-password>
```

2. プルシークレットファイルを準備します。

イメージプルシークレットのディレクトリを準備します。

```
tmpdir=$(mktemp -d /tmp/pull-secret-<your-pull-secret-name)
chmod 0700 "$(tmpdir)"
```

3. レジストリにログインし、シークレットを更新します。

OC レジストリにログインします。

```
oc registry login \
--registry=$${CONTANER_REGISTRY_URI}" \
--auth-basic="${CONTAINER_REGISTRY_USERNAME}:${CONTAINER_REGISTRY_PASSWORD}"
    --to="$tmpdir/pull-secret"
```

次に、シークレットを更新します。

```
oc set data secret/pull-secret -n openshift-config \
    --from-file=.dockerconfigjson=" $tmpdir/pull-secret"
```

4. クリーンアップ。

シークレット用に作成した一時ディレクトリをクリーンアップします。

```
rm -rf $tmpdir
```

- ** *Enabling GenAI Builder on Hybrid Manager* GenAI Builder のオプショナル**
- ** *Enabling Data Catalogs on Hybrid Manager* カタログを使用する場合のオプショナル**
- ** *Customizing Migration Portal secrets for secure internal communication* 移行ポータルとの内部通信を保護するオプション**

### ステップ 6 認証キーとセキュリティキーを構成する

HM とその基になるコンポーネントには、コンポーネント間の適切な通信を保証し、機密データを保護するために、安全な認証メカニズムが必要です。

1. **AES-256 暗号化キーを生成します。**

HM は、AES-256 暗号化を使用して、通信中または保存中の機密データたとえば、データベース資格情報、トークンなどを保護します。ランダムな AES-256 暗号化キーを生成するには

```
export AES_256_KEY=$(openssl rand -base64 32)
```

2. **キーを Kubernetes に保存します。**

HM および関連サービスがキーにアクセスできるようにするには、適切な名前空間に Kubernetes シークレットを作成します。

1. 次のコマンドを実行してシークレットを作成します。

```
kubectl create secret generic hm-auth-key \
    --namespace <hm-namespace> \
    --from-literal=aes-256-key=$AES_256_KEY
```

2. シークレットを確認します。

```
kubectl get secret hm-auth-key --namespace <hm-namespace>
```

### ステップ 7 オブジェクトストレージを構成する

オブジェクトストレージバックエンド HM バックアップをサポートするには、クラスターを専用のオブジェクトストレージソリューションたとえば、MinIO、Ceph Object Gateway、または AWS S3 互換ストレージに接続します。

特に、ハイブリッドマネージャーはこのバケットを使用して以下を保存します。

- 管理対象ストレージの場所
- Postgres WAL + バックアップ
- ログ Postgres と内部サービスの両方
- メトリック Postgres と内部サービスの両方

#### バケットを作成する

MinIO または Ceph bucket を作成します。

#### バケットユーザーを作成する

プロバイダーでハイブリッドマネージャー用の専用バケットを作成した後、バケットへの完全なアクセスを持つ ID ユーザーと、ID ユーザーの AWS 静的資格情報を作成して、バケットへのハイブリッドマネージャーにアクセスを付与します。

MinIO Client をダウンロードします。

1. MinIO 展開名を取得し、ユーザー名とパスワードを作成します。

```
export MINIO_DEPLOYMENT_NAME=<minio_deployment_name> # deployment name to minio
export BUCKET_NAME=<minio_bucket_name> # bucket name
export AWS_ACCESS_KEY_ID=<minio_user_name> # user name
export AWS_SECRET_ACCESS_KEY=<minio_user_password> # password to the user
export MINIO_POLICY_NAME=<minio_policy_name> # policy to minio
```

1. 新しいユーザーを作成します。

```
mc admin user add ${MINIO_DEPLOYMENT_NAME} ${AWS_ACCESS_KEY_ID} ${AWS_SECRET_ACCESS_
↪KEY}
```

1. MinIO ポリシーを 1 つ作成します。

```
cat << EOF > policy.json
{
    "Version" : "2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Action" : [
                "s3:*"
            ],
            "Resource" : [
                "arn:aws:s3:::${BUCKET_NAME}",
                "arn:aws:s3:::${BUCKET_NAME}/*"
            ]
        }
    ]
}
EOF
```

1. ポリシーを適用します。

```
mc admin policy create ${MINIO_DEPLOYMENT_NAME} ${MINIO_POLICY_NAME} ./policy.json
```

1. ユーザーにポリシーをアタッチします。

```
mc admin policy attach ${MINIO_DEPLOYMENT_NAME} ${MINIO_POLICY_NAME} --user ${AWS_
↪ACCESS_KEY_ID}
```

Ceph のコマンドラインインターフェイス Radosgw-admin utility をダウンロードします。

1. バケットといくつかの環境変数を取得します。

```
export BUCKET_NAME=<bucket_name> # bucket name
export USER_NAME=<user_name> # user name
```

1. ユーザーを作成します。

```
radosgw-admin user create --uid ${USER_NAME} --display-name= ${USER_NAME} > user.json

export AWS_ACCESS_KEY_ID=$(cat user.json | jq -r .keys[0].access_key)
export AWS_SECRET_ACCESS_KEY=$(cat user.json | jq -r .keys[0].secret_key)
```

1. ユーザーにバケットへのアクセスを付与します。

```
radosgw-admin bucket link --bucket ${BUCKET_NAME} --uid ${USER_NAME}
```

### バケットアクセスのシークレットを作成する

バケットへの HM アクセスのための専用ユーザーを準備した後、作成したユーザーをハイブリッドマネージャーに関連付けるシークレットを作成して、オブジェクトストレージへのアクセスを提供します。

```
apiVersion: v1
kind: Secret
metadata:
  name: edb-object-storage # name cannot be changed
  namespace: default # namespace cannot be changed
stringData:
    auth_type: credentials

    # Optional: Used only when the object storage servers certificate is not issued␣
↪by a well-known CA
    #
    # Base64 string of the CA bundle for the certificate used by the object storage␣
↪server
    aws_ca_bundle_base64: xxx

    # Required: Endpoint URL to the object storage
    aws_endpoint_url_s3: xxx

    # Required: AWS Static Credentials - AWS_ACCESS_KEY_ID
    aws_access_key_id: xxx

    # Required: AWS Static Credentials - AWS_SECRET_ACCESS_KEY
    aws_secret_access_key: xxx

    # Required: Bucket name
    bucket_name: xxx
```

(次のページに続く)

(前のページからの続き)

```
    # Required: Region of the bucket
    aws_region: xxx

    # Optional: true or false
    #
    # When server-side encryption is disabled, set this to true. By default, its␣
↪value is false, indicating that server-side encryption is enabled.
    server_side_encryption_disabled: bool
```

#### ステップ 8 ネットワークセキュリティを検証する

- ファイアウォールルール OpenShift クラスター、外部オブジェクトストレージ、および HM の外部依存関係の間で必要な通信がファイアウォールルールで許可されていることを確認します。
- 上り/下りルール必要なエンドポイントと IP 範囲のみにアクセスを制限して、露出を最小限に抑えます。

#### ステップ 9 クラスター構成を調整する

一部のデフォルトの OpenShift 構成は調整が必要になる場合があります。

- ノードプールの調整リソースの割り当て、テイント/トレランスを含む、HM ワークロードのノード構成を最適化します。
- ストレージクラスの調整必要に応じて、デフォルトのストレージクラスを置き換えまたは変更します。
- ネットワークポリシーネットワーク構成を調整して、HM 要件との互換性を保証します。

#### ステップ 10 必要なアドオンをインストールする

- 必要な OpenShift オペレーターまたはアドオンのインストールと構成
  - サービスメッシュ HM は istio サービスに依存しています Mesh.
  - モニタリングとロギング OpenShift モニタリングおよびロギングソリューションを設定して、HM 可観測性コンポーネントと統合します。

#### 必要な環境変数の設定に進みます。

クラスターが構成されたら、プレインストールの次のフェーズ *setting the necessary environmental variables for installation* に進みます。

### Set the necessary environmental variables for install

HM クラスターは、インストール中に特定のユーザードメインへのアクセスが必要です。

### ドメイン変数を設定する

```
export DEFAULT_INGRESS_DOMAIN=<rhos default ingress domain name> # rhos default␣
↪ingress domain
```

**重要**

*DEFAULT_INGRESS_DOMAIN* は、クラスターのデフォルトのイングレスドメイン名です。次のコマンドを使用して照会できます。

```
oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

ingresses.config/cluster オブジェクトは、インストール中に作成されます。

詳細については、 the OpenShift documentation を参照してください。

```
export OPENSHIFT_CONSOLE_DOMAIN_NAME=<openshift console domain name> # Domain name to␣
↪OpenShift console
export PORTAL_DOMAIN_NAME=<portal domain name> # Customer domain name for the EDB␣
↪Postgres AI Platform Portal
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME=<transporter rw service domain name> #␣
↪Customer domain name for the EDB Postgres AI Platform Transporter RW Service
```

**重要**

*PORTAL_DOMAIN_NAME* および `TRANSPORTER_RW_SERVICE_DOMAIN_NAME` は次のとおりである必要があります。

- 互いに異なる - 両方に同じ名前を使用しないでください - OPENSHIFT_CONSOLE_DOMAIN_NAME とは異なる - <app-name>.<cluster-domain> としてフォーマット

```
export BEACON_SERVICE_DOMAIN_NAME=<beacon service domain name> # Customer domain name␣
↪for the EDB Postgres AI Beacon Service
export CERT_MANAGER_NAMESPACE=<cert-manager namespace> # Customer namespace in which␣
↪cert-manager is installed as pre-requirements
```

### ストレージクラス変数を設定します

HM クラスターには、事前定義されたストレージクラスが必要です。

```
export STORAGE_CLASS=<storage class> # Customer storage class on OpenShift Container␣
↪Platform
```

#### 認証変数を設定する

HM のインストールにはいくつかの認証値が必要です。

```
export AUTHENTICATION_EMAIL=<email> # The email address of the user.
export AUTHENTICATION_HASH=<password hash>  # The hashed password of the user␣
↪obtained for example as: echo "mypassword" | htpasswd -BinC 10 admin | cut -d: -f
export AUTHENTICATION_USERNAME=<Owner MyCompany> # The username of the user
export AUTHENTICATION_USER_ID="c5998173-a605-449a-a9a5-4a9c33e26df7" # A unique␣
↪identifier for the user must not be changed since the first time installation.
```

#### オプション FIPs モードを設定します Transporter Data Operator

オプションで、Transporter Data Operator の FIPS モードを設定できます。

```
export TRANSPORTER_FIPS_ENABLED=true # Enable FIPS mode for Transporter Data Operator
```

### Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator

#### オペレーターの依存関係をインストールします

EDB Postgres AI Operator には、すべての名前空間を管理する Red Hat OpenShiftRHOS の明示的な依存関係 cert-manager オペレーターがあります。 EDB Postgres AI Platform Operator の前に自動的にインストールされます。

Red Hat のオペレーターには、次のような暗黙の依存関係もあります。

- すべての名前空間を管理する RHOS Service Mesh 2
- 名前空間を管理するノード機能検出オペレーター `openshift-nfd`
- 名前空間 `nvidia-gpu-operator` を管理する NVIDIA GPU Operator

これらのオペレーターを既にインストールしている場合、再インストールする必要はありません。

#### RHOS のオペレーターをインストールする

1. RHOS に EDB Postgres AI Operator をインストールするには、最初に EDB Cloudsmith リポジトリにアクセスするためのイメージプルシークレットを準備していることを確認します。また、リポジトリ `docker.enterprisedb.com/pgai-platform/edb-hcp-operator/manager` からイメージをプルするために必要な権限があることを確認します。

```
export DOCKER_SERVER="docker.enterprisedb.com"
export DOCKER_USERNAME="<your-docker-user-name>"
export DOCKER_PASSWORD="<your-docker-password>"
```

1. EDB EDB Postgres AI Operator の新しい OC プロジェクトを作成します。

```
oc new-project edb-hcp-operator-system
```

1. EDB Cloudsmith リポジトリにアクセスするようにイメージプルシークレットを設定します。

```
oc create secret docker-registry edb-hcp-operator-pull-secret \
  --docker-server="$DOCKER_SERVER" \
  --docker-username="$DOCKER_USERNAME" \
  --docker-password="$DOCKER_PASSWORD" \
  --namespace-edb-hcp-operator-system
```

1. OpenShift Container PlatformOCP を使用して、Red Hat OpenShift の EDB Postgres AI Operator をインストールします。
2. OCP Web コンソールにログインします。
3. **Operators > OperatorHub** を選択します。
4. フィルタボックスに `EDB Postgres AI` と入力します。
5. **EDB Postgres AI Operator** を選択し、 **Install** を選択します。
6. [Install Operator] ページで、次のことを行います。オペレーターには、前に作成した `edb-hcp-operator-system` 名前を使用して **インストールされた名前空間** を選択します。 b. **Install** を選択します。

> **Tip**
>
> RHOS では、cert-manager はハイブリッドマネージャーの必須依存関係です。 cert-manager をインストールするときは、正常に動作するように、**インストールモード**が**クラスター上のすべての名前空間**として構成されていることを確認します。

### RHOS にハイブリッドマネージャーをインストールする

EDB Postgres AI Operator を RHOS クラスターにインストールした後、Operator を使用して Hybrid Manager をインストールする必要があります。次の YAML ファイルは、ハイブリッドマネージャーのインストールを開始します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  source:
    useLocalKustomizations: true
  flavour: rhos
  version: v1.2.0+appl
  imageRegistry: ${CONTAINER_REGISTRY_URI}
  imagePullSecrets:
    - name: edb-cred
      namespace: edbpgai-bootstrap
  globalParameters:
    internal_backup_folder: 4befab698c8d
```

(次のページに続く)

(前のページからの続き)

```
  portal_domain_name: ${PORTAL_DOMAIN_NAME}
  default_ingress_domain: ${DEFAULT_INGRESS_DOMAIN}
  storage_class: ${STORAGE_CLASS}
componentsParameters:
  edb-migration-portal:
    enable_pdb: "false"
  transporter-data-operator:
    fips_enabled: "${TRANSPORTER_FIPS_ENABLED}"
  transporter-dp-agent:
    rw_service_url: https://${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}/transporter
  transporter-rw-service:
    domain_name: ${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}
  upm-api-pnxt-metric:
    log_level: info
    tsdb_host: thanos-query.monitoring.svc.cluster.local
    tsdb_port: "9090"
  upm-api-pnxt-pd:
    log_level: info
  upm-beaco-ff-base:
    usage_generator_interval: 24h
  upm-beaco-usages:
    usage_generator_interval: 24h
  upm-beacon:
    server_host: ${BEACON_SERVICE_DOMAIN_NAME}
    beacon_server_replicas: "1"
    external_metrics_replicas: "1"
    beacon_db_hibernation: "\"off\""
  upm-istio-gateway:
    openshift_console_domain_name: ${OPENSHIFT_CONSOLE_DOMAIN_NAME}
    cookie_aeskey: ${AES_256_KEY}
    ingress_http_node_port: "32542"
    ingress_https_node_port: "30288"
    ingress_grpc_tls_node_port: "30290"
  upm-prometheus:
    memory: 4Gi
  upm-trust-manager:
    cert_manager_namespace: ${CERT_MANAGER_NAMESPACE}
pgai:
  portal:
    authentication:
      clientSecret: edb-postgres-ai-secret
      idpConnectors: []
      staticPasswords:
        - email: ${AUTHENTICATION_EMAIL}
          hash: ${AUTHENTICATION_HASH}
```

(次のページに続く)

(前のページからの続き)

```
            username: ${AUTHENTICATION_USERNAME}
            userID: ${AUTHENTICATION_USER_ID}
  beaconAgent:
    applianceProviderEnabled: true
    plaintext: true
    beaconServerAddress: "upm-beacon-server-ingestion.upm-beacon.svc.cluster.
↪local:8080"
    tlsInsecure: false
    location: "devspatcher"
    logLevel: "debug"
    beaconDBConnected: true
    inCluster: true
    provisioning:
      loadBalancersEnabled: true
      nodePortDomain: "nodeportdomain.com"
      imageDiscovery: true
      imagesetDiscoveryAuthenticationType: "basic"
      imagesetDiscoveryContainerRegistryURL: "us-east1-docker.pkg.dev"
    providers:
      - appliance
```

YAML が適用された後、次のコマンドを使用してインストールステータスを監視できます。

```
oc get hybridcontrolplane edbpgai -o yaml
```

インストールが現在いるフェーズに応じて、3 つの異なるステータスを返すことができます。

```
- deployed # the installation was successful

- deploying # the installation is still in progress

- failed: # the install was unsuccessful
```

インストールが成功すると、 *updating DNS zone records and verify the install* ** .

### Operator を介してインストールされた Hybrid Manager のアンインストール

> **警告**
>
> 次の手順は、Hybrid Manager をアンインストールします。 Hybrid Manager をアンインストールする場合にのみ、この手順を使用します。

ハイブリッドマネージャーを削除する必要がある場合があります。ハイブリッドマネージャーはデフォルトで削除から保護されているため、最初にその保護を無効にする必要があります。

```
kubectl annotate hybridcontrolplane edbpgai biganimal.enterprisedb.io/deletion-
↪protect=disabled --overwrite
```

次に、アンインストールできます。

```
kubectl delete hybridcontrolplane edbpgai --wait=false
```

プロセスをモニタするには、次を実行します。

```
kubectl get hybridcontrolplane edbpgai -o yaml
```

### Connecting to Hybrid Manager

EDB Postgres AI Hybrid Manager プラットフォームのインストールが成功したことを確認するには、ブラウザーを使用して Hybrid Manager コンソールに接続します。

```
https://portal.foo.network/
```

この構成では、ホスト名が広域インターネットに存在しないため、https は検証されません。証明書の警告を受け入れて、ポータルに進みます。証明書の警告を受け入れて、ポータルに進みます。

ログイン画面で、ブートストラップインストールで生成されたユーザー名とパスワードを入力できます。詳細については、*Managing users in Hybrid Manager* を参照してください。

ログインしたら、 *Create a database cluster template* ができます。

### Updating DNS zone records

Helm または Operator のインストールが完了したら、DNS ゾーンレコードを更新し、ハイブリッドマネージャーが予想通りに動作していることを確認します。

#### DNS ゾーンレコードの更新

1. ポータルサービスのホスト名を取得します。このホスト名は、ハイブリッドマネージャーの入力エンドポイントを表します。

```
kubectl get svc -n istio-system istio-ingressgateway -o yaml | yq .status.
↪loadBalancer.ingress[0].hostname
```

1. 次のドメイン名の DNS ゾーンに CNAME レコードを作成し、取得したホスト名をポイントします。

   - `PORTAL_DOMAIN_NAME`
   - `TRANSPORTER_RW_SERVICE_DOMAIN_NAME`
   - `BEACON_SERVICE_DOMAIN_NAME`

各 CNAME レコードは、ユーザー向けドメイン名を入力ホスト名にマッピングします。

これで、 *connect to the portal* を実行して、インストールを確認できます。

### 4.3.3 Install Hybrid Manager on AWS EKS

ブートストラップする Helm チャート、または Hybrid Manager オペレーターを使用して、 Amazon EKS に Hybrid ManagerHM をインストールできます。

- 前提条件
- *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator*
- *Hybrid Manager Operator* を使用したアンインストール
- *Installing using the Helm chart*
- *Connecting to the portal*
- ハイブリッドマネージャーのアンインストール

#### EKS installation prerequisites

AWS EKS には、EDB Postgres AI ソフトウェア展開をインストールする前に満たす必要がある前提条件があります。インストールプロセスを開始する前に、前提条件を満たす必要があることを確認します。

#### 前提条件

- *A system from which you can run the installation process*
- *An AWS account with the necessary permissions*
- *An EKS cluster configured to host HM*

#### 次のステップ

前提条件を完了したら、 *installing the EDB Postgres AI Software Deployment on AWS EKS* に進むことができます。

#### Install system

インストールプロセスを実行できるシステムが必要です。このシステムには、次のソフトウェアをインストールする必要があります。

- AWS CLI —AWS のサービスを管理するための AWS コマンドラインインターフェイス CLI。
- kubectl Kubernetes クラスターを管理します。
- yq HM インストールファイルを準備します。
- Helm Kubernetes マニフェストをレンダリングおよび適用します。

リンクされた手順、または以下にある特定のオペレーティングシステムの手順を使用して、これらのツールをインストールできます。

### macOS

AWS CLI、kubectl、yq、および Helm をインストールします。

```
brew install awscli kubectl yq helm
```

### Linux

> **注釈**
>
> これらの手順は、デフォルトでスナップサポートが利用できることを前提としているため、Ubuntu 16.04 以降を対象としています。

```
sudo snap install aws-cli --classic
sudo snap install helm --classic
sudo snap install kubectl --classic
sudo snap install yq
```

AWS CLI をインストールします。

```
sudo yum install awscli
```

kubectl をインストールします。

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.
↪txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
```

Helm をインストールします

```
curl -LO https://get.helm.sh/helm-v3.12.3-linux-amd64.tar.gz
tar -xvf helm-v3.12.3-linux-amd64.tar.gz
sudo mv linux-amd64/helm /usr/local/bin/helm
helm version
```

yq をインストールします。

```
sudo curl -L "https://github.com/mikefarah/yq/releases/latest/download/yq_linux_amd64
↪" -o /usr/bin/yq
sudo chmod +x /usr/bin/yq
yq --version
```

```
sudo apt update
sudo apt install snapd
```

(次のページに続く)

(前のページからの続き)

```
sudo snap install snapd
sudo snap install aws-cli --classic
sudo snap install kubectl --classic
sudo snap install yq
```

### Windows

```
winget install Amazon.awscli --source winget
winget install kubectl --source winget
winget search helm
winget install helm
winget install yq --source winget
```

1. AWS CLI をインストールします。
2. kubectl をインストールします。
3. yq をインストールします。
4. *Helm* **チャートを適用する** をインストールします。

### Helm のインストール

1. Windows の Helm バイナリをダウンロードします。 Helm GitHub Releases page で、最新リリースを見つけて、help-v3.xx-windows-amd64.zip ファイルをダウンロードします。
2. Zip ファイルを解凍します。7-Zip、WinRAR、またはビルトインの Windows 抽出ユーティリティなどのツールを使用して、ダウンロードしたファイルを解凍します。
3. `helm.exe` を、抽出したフォルダーからパス内のディレクトリにコピーします。一般的なディレクトリには、`C:\Program Files\` 、`C:\Program Files (x86)\` が含まれます。または、 `C:\Helm\` などのディレクトリを作成し、パスに追加できます。
4. インストールを確認します。
    1. 新しいコマンドプロンプトまたは PowerShell ウィンドウを開きます。
    2. 次のコマンドを実行して、Helm が正しくインストールされていることを確認します。

````` bash Helm バージョン

```

``````
3. 出力には、Helm バージョン情報が表示されます。
``````

次のステップ

必要なソフトウェアをインストールしたら、*setting up an AWS account with the necessary permissions* に進むことができます。

Setting up an AWS account

Hybrid Manager クラスターをインストールするには、必要なリソースを作成する権限を持つ AWS アカウントが必要です。

必要な権限

EKS クラスターを作成するには、次の権限を持つ AWS アカウントが必要です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*",
                "sts:AssumeRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:PutRolePolicy",
                "iam:PassRole",
                "iam:ListPolicies",
                "iam:ListRolePolicies",
                "iam:ListAttachedRolePolicies",
                "iam:ListRoleTags",
                "iam:TagRole",
                "iam:GenerateServiceLastAccessedDetails",
                "iam:ListInstanceProfilesForRole",
                "iam:ListOpenIDConnectProviders",
                "iam:GetOpenIDConnectProvider",
                "iam:CreateOpenIDConnectProvider",
                "iam:TagOpenIDConnectProvider",
                "iam:GetServiceLastAccessedDetails",
                "iam:AttachRolePolicy",
                "ec2:TerminateInstances",
                "iam:deleteOpenIDConnectProvider",
                "iam:detachRolePolicy",
                "iam:DeleteRolePolicy"
            ],
            "Resource": "*"
```

(次のページに続く)

(前のページからの続き)

```
        }
    ]
}
```

上記の権限でロールを作成し、ユーザーにアタッチして既存の権限に追加することをお勧めします。また、セキュリティチームとこれらの権限を確認して、ユースケースに適切であることを確認することをお勧めします。

AWS アカウントを作成する

AWS アカウントをまだお持ちでない場合は、 AWS website で作成できます。**Create an AWS Account** を選択し、指示に従います。

IAM ユーザーを作成する

1. AWS Management Console にログインします。
2. 検索バーで **IAM** を検索して、IAM サービスに移動します。
3. IAM ダッシュボードの左側のメニューで、 **Users** を選択し、次に **Add user** を選択します。
4. 新しいユーザーのユーザー名を入力し、 **Programmatic access** チェックボックスを選択します。
5. **Next: Permissions** を選択します。
6. ポリシーを添付します。 [**アクセス許可の設定**] ステップで、[**既存のポリシーを直接アタッチします**] を選択し、AdministratorAccess ポリシーを検索します。ポリシーを選択し、**Next: Tags** を選択します。
7. 必要に応じて、ユーザーにタグを追加し、 **Next: Review** を選択します。
8. ユーザーの詳細を確認します。 IAM ユーザーを作成するには、 **Create user** を選択します。

> **注釈**
>
> アクセスキー ID とシークレットアクセスキーを記録します。 AWS CLI を構成するために必要です。

AWS CLI の設定

AWS CLI を構成するには、最初にローカルマシンに AWS CLI をインストールする必要があります。*Install system* 前提条件の一部としてこの手順を実行します。

CLI を構成するには、次のコマンドを実行します。

```
aws configure
```

次のステップ

AWS CLI を構成すると、 *create an EKS cluster* ができます。

Creating an EKS cluster

installed your system および *configured your AWS account* を取得したら、AWS マネジメントコンソールを使用して EKS クラスターを作成できます。

ステップ 1AWS マネジメントコンソールにログインする

https://aws.amazon.com/console/ にアカウント資格情報を使用してログインして、管理コンソールにアクセスします。

検索バーに `EKS` と入力し、検索結果から **Elastic Kubernetes Service** を選択します。

ステップ 2 ブートストラップキットを入手する

ブートストラップキットを使用して Hybrid ManagerHM をインストールします。このキットは、EDB Software Deployment プラットフォームのインストールに使用されるスクリプトと Helm チャートのコレクションです。

次のファイルが必要です。

- *eks-fix-auto.sh*

ファイルをローカルマシン上のディレクトリにコピーし、cd をそのディレクトリにコピーします。

次に、AWS マネジメントコンソールを使用して次のことを行います。

1. クラスターを作成する AWS リージョンを選択します。
2. 選択したリージョンをメモします。これは、後のプロセスで必要になります。
3. EKS ページに移動します。検索バーに `EKS` と入力し、検索結果から **Elastic Kubernetes Service** を選択します。
4. EKS ページで、 **Create Cluster** を選択します。

ステップ 3 クラスターを構成する

構成オプションで

- **Quick configuration (with EKS Auto Mode) -*new** *を選択します。
- **Cluster name** フィールドに、クラスターの名前を入力し、メモします。
- **Kubernetes version** フィールドで、使用するバージョンを選択します。(EKS 自動モードのデフォルトは 1.31 です。)
- **Cluster IAM Role** フィールドでは、値 `AmazonEKSAutoClusterRole` のままにします。
- **Node IAM Role** フィールドでは、値 `AmazonEKSAutoNodeRole` のままにします。

クラスターの VPC を選択または作成します

クラスターに VPC を提供する必要があります。VPC を作成するか、既存の VPC を使用できます。

1. **Create VPC** を選択します。
2. [**Create VPC**] ページの **Resources to create** の下で、 **VPC and more** を選択します。
3. 次の情報を入力します。
    - **Name tag auto-generate** の下に、VPC の名前を入力します。
    - **Number of Availability Zones** の下で、3.
    - **パブリックサブネットの数** の下で 3.
    - 、3 が選択されていることを確認します。
    - **プライベートサブネットの数**の下で、3 が選択されていることを確認します。
    - NAT ゲートウェイで、 **In 1 AZ** を選択します
4. **Create VPC** を選択します。 [**Create VPC workflow**] ページが表示され、VPC の作成手順が実行されます。完了したら、[**View VPC**] を選択します。

VPC のビューで、リソースマップのパブリックサブネットごとに

1. サブネットにカーソルを合わせ、表示される矢印アイコンのある正方形のボックスを選択します。サブネットを編集できるページが開きます。
2. **Actions > Edit subnet settings** を選択します。
3. **Auto-assign public IPv4 address > Enable auto-assign public IPv4 address** を選択します。
4. **Save** を選択します。

各パブリックサブネットのタグを追加します。

1. **Tags** タブを選択します。
2. **Manage Tags** を選択します
3. **Add Tag** を選択します
4. **Key** フィールドに `kubernetes.io/role/elb` を入力し、 **Value** フィールドに 1 を入力します。
5. **Save** を選択します
6. タブ/ウィンドウを閉じ、VPC ビューに戻ります。

リソースマップ内の各プライベートサブネットのタグを追加します。

1. サブネットにカーソルを合わせ、表示される矢印アイコンのある正方形のボックスを選択します。
2. **Tags** タブを選択します。
3. **Manage Tags** を選択します
4. **Add Tag** を選択します

5. **Key** フィールドに kubernetes.io/role/internal-elb を入力し、**Value** フィールドに 1 を入力します。
6. **Save** を選択します
7. タブ/ウィンドウを閉じ、VPC ビューに戻ります。

使用する既存の VPC がある場合は、ドロップダウンから選択します。

VPC には、異なるアベイラビリティーゾーンに少なくとも 2 つのパブリックサブネットと 2 つのプライベートサブネットが必要で、次のタグが必要です。

パブリックサブネット

- kubernetes.io/role/elb 1

プライベートサブネット

- kubernetes.io/role/internal-elb 1

EKS クラスター作成ページに戻る

新しい VPC を作成した場合、EKS クラスター作成ページを更新する必要があります。ドロップダウンを更新して、作成した VPC を選択できるようにするには、**Create VPC** の横にある円形の矢印アイコンを選択します。

1. クラスターのサブネットを選択します。異なるアベイラビリティーゾーンで少なくとも 2 つのプライベートサブネットを選択する必要があります。

> **注釈**
>
> これらのサブネットは、Kubernetes コントロールプレーンと Kubernetes ワーカーノード間の通信を促進することであるため、パブリックではなくプライベートである必要があります。パブリックサブネットは、クラスターの内部を不必要なセキュリティの脅威にさらしたままにする可能性があります。サブネットのタグ付けは、正しいサブネットにイングレスロードバランサーを作成するように EKS に通知します。

2. **Create** を選択します。

クラスターを作成しています。これには数分かかる場合があります。ページの上部にあるパンくずリストで**クラスター**を選択することにより、EKS コンソールで作成をモニタできます。

ステップ 4kubectl を構成する

クラスターを作成した後、クラスターと対話するように kubectl を構成する必要があります。

ターミナルを開き、実行します。

```
aws eks --region <REGION> update-kubeconfig --name <CLUSTER-NAME>
```

そこで

- <REGION> は、クラスターを作成したリージョンです。

- <CLUSTER-NAME> は、クラスターに付けた名前です。

> **注釈**
>
> *AWS_PROFILE* 環境変数を設定していない場合は、コマンドに `--profile <profile-name>` オプションを含めます。

EKS クラスターと対話できるようになりました。テストするには、次を実行します。

```
kubectl get pods -A
```

ステップ 5S3 バケットを作成し、クラスターに接続する

HM には、バックアップやその他のデータを保存するために S3 バケットが必要です。S3 バケットを作成し、クラスターに接続します。

S3 バケットを作成する

1. AWS マネジメントコンソールで S3 サービスを開きます。検索バーに S3 と入力し、検索結果から **S3** を選択します。
2. **バケットの作成** を選択します。
3. バケット名フィールドの名前を入力します。これはグローバルに一意である必要があります。
4. **Create Bucket** を選択します。

アクセス用のロールを作成する

1. EKS ダッシュボードに戻ります。検索バーに EKS と入力し、検索結果から **Elastic Kubernetes Service** を選択します。
2. 作成したクラスターを選択します。
3. 概要の **Details** セクションで、OpenID Connect プロバイダーの URL をコピーします。
4. AWS マネジメントコンソールで IAM サービスを開きます。検索バーに IAM と入力し、検索結果から **IAM** を選択します。
5. 左側のメニューで、 **Identity Providers** を選択します。
6. **Add Provider** を選択します。
7. **Configure Provider** ページで、**OpenID Connect** を選択し、コピーした URL を **Provider URL** フィールドに貼り付けます。
8. オーディエンスフィールドに sts.amazonaws.com と入力します。
9. **Add Provider** を選択します。

緑色のバナーにメッセージ Provider created successfully が表示され、[**役割の割り当て**] ボタンが表示されます。

10. **役割の割り当て** を選択します。

11. **ロールオプション** の下で、 **Create a New Role** を選択します。**Next** を選択します。

信頼できるエンティティを選択します

1. [**信頼できるエンティティの選択**] ページで、 **Web ID** を選択します。

Web attempts セクションの **Identity Provider** フィールドには、作成したばかりのプロバイダーが表示されます。

2. **Audience** フィールドで `sts.amazonaws.com` を選択し、 **Next** を選択します。

新しいロールに権限を付与します。

1. **Add Permissions** ページで、検索フィールドに `s3full` と入力します。 `AmazonS3FullAccess` ポリシーを選択し、 **Next** を選択します。

2. `<clustername>-s3access` などのロール名を入力し、 **Create Role** を選択します。

緑色のバナーがメッセージ `Role created successfully` とともに表示され、[**View Role**] ボタンが表示されます。

3. **役割の表示** を選択します。

4. ロールのビューで、ロールの **ARN** 値をコピーします。

シークレットを作成する

正常に機能するには、デフォルトの名前空間`-n default` にシークレットを作成する必要があります。

ターミナルで、次を実行します。

```
kubectl create secret generic -n default edb-object-storage \
    --from-literal auth_type=workloadIdentity \
    --from-literal aws_region=<REGION> \
    --from-literal aws_role_arn=<ROLE-ARN> \
    --from-literal bucket_name=<BUCKET-NAME>
```

そこで

- `<REGION>` は、クラスターを作成するリージョンです。
- `<ROLE-ARN>` は、作成したロールの ARN です。
- `<BUCKET-NAME>` は、作成したバケットの名前です。

ステップ 6EBS CSI ドライバーと CSI スナップショットコントローラを追加する

PGAIHM でスナップショットを正しく管理するには、EBS CSI ドライバーと CSI スナップショットコントローラの 2 つの AWS CSI 関連ドライバーが必要です。

AWS EBS CSI ドライバー

EBS CSI ダイバーを使用すると、Kubernetes クラスターは、Elastic Block StoreEBS や Elastic File SystemEFS などの AWS サービスを使用して、ストレージリソースを動的にプロビジョニングおよび管理できます。この統合により、Kubernetes ワークロードは AWS ストレージサービスを永続ストレージとしてシームレスに使用できます。

EBS CSI ドライバーをインストールするには

1. PGAIHM をインストールする EKS クラスターの EKS cluster ページで、**Add-ons** タブを選択します。
2. **Get More Addons** を選択します。
3. [**アドオン**] ページの EBS CSI Driver のタイルで、チェックボックスを選択し、[**Next**] を選択します。
4. [**Configure**] ページで、[**Next**] を選択します。
5. **Review** ページで、 **Create** を選択します。

EBS CSI アドオンドライバーがクラスターに追加されます。

AWS CSI スナップショットコントローラ

CSI スナップショットコントローラは、ボリュームスナップショットを管理する Kubernetes コントローラです。

CSI スナップショットコントローラをインストールするには

1. PGAIHM をインストールする EKS クラスターの EKS cluster ページで、**Add-ons** タブを選択します。
2. **Get More Addons** を選択します。
3. [**アドオン**] ページの CSI スナップショットコントローラのタイルで、チェックボックスを選択し、[**Next**] を選択します。
4. [**Configure**] ページで、[**Next**] を選択します
5. **Review** ページで、 **Create** を選択します。

CSI スナップショットコントローラアドオンがクラスターに追加されます。

ステップ 7 互換性のためにクラスターを更新する

AWS マネジメントコンソールによって作成された EKS クラスターは、HM と完全な互換性がないため、クラスターにいくつかの変更を加える必要があります。これらの変更を行うためのスクリプトを利用できます。

Helm eks-fix-auto.sh スクリプトまたは *Operator eks-fix-auto.sh* スクリプトをダウンロードし、ターミナルで実行します。

```
$SHELL eks-fix-auto.sh
```

スクリプトは、デフォルトのノードプールとストレージクラスをワークロードにより適切なものに置き換え、その他のいくつかの設定を構成します。

ステップ 8 ハイブリッドマネージャーをインストールする

次に、 *Install the Hybrid Manager on EKS* 。

Install Hybrid Manager on EKS

ブートストラップする Helm チャート、または Hybrid Manager オペレーターを使用して、 Amazon EKS に Hybrid ManagerHM をインストールできます。

EKS にハイブリッドマネージャーをインストールする前に、 **前提条件** を満たす必要があります。

次に、次のいずれかの方法を使用してハイブリッドマネージャーをインストールできます。

- *Installing Hybrid Manager on AWS EKS using the Hybrid Manager Operator*
- *Installing Hybrid Manager on AWS EKS*

Hybrid Manager をインストールした後、 *Connecting to the Hybrid Manager Console* ガイドを使用して EDB Postgres AI ハイブリッドマネージャーポータルに接続できます。

Installing Hybrid Manager on AWS EKS using the Hybrid Manager Operator

Hybrid Manager Operator を使用して、 Amazon EKS に Hybrid Manager をインストールできます。これは、EKS クラスターに Hybrid Manager プラットフォームを展開する Helm チャートです。オペレーターは、プラットフォームのインストール、アップグレード、アンインストールを含む、Hybrid Manager プラットフォームのライフサイクルを管理する責任を負います。

> **重要**
>
> インストールプロセスのこの部分を開始する前に、 **前提条件** を完了します。 EKS クラスターの作成、必要なコンポーネントのインストール、および Hybrid Manager Operator の準備ができた環境の構成をステップ実行します。

インストールするバージョンを設定します

EDB Hybrid Manager プラットフォームのバージョンは、環境変数 EDB_PLATFORM_VERSION によって設定されます。これは、インストールするプラットフォームのバージョンに設定する必要があります。たとえば、バージョン v1.2.0 をインストールするには、次のように環境変数を設定します。

```
export EDB_PLATFORM_VERSION="v1.2.0"
```

トークンを設定します

このインストールプロセスでは、EDB サブスクリプショントークンを環境変数として保存する必要があります。EDB Account Profile に移動して、取得できます。プロンプトが表示されたら、ログインします。そこに、Repos 2.0 トークンのエントリがあります。

この値を取得して、環境変数として設定します。

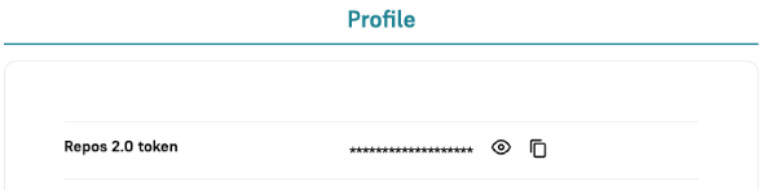


図 1: Repo Token

```
export EDB_SUBSCRIPTION_TOKEN="your-token"
```

ブートストラップキットを入手する

Hybrid Manager は、ブートストラップキットを使用してインストールされます。このキットは、EDB Hybrid Manager プラットフォームのインストールに使用されるスクリプトとチャートのコレクションです。

- eks-install-operator.sh - このスクリプトは、EKS クラスターに Hybrid Manager Operator をインストールします。

```
registry="docker.enterprisedb.com"
username="pgai-platform"
password=""

echo "Enter the password for ${username}@${registry}"
read -s password

if [ -z "$password" ]; then
    echo 1>&2 "Error: password is required" >&2
    echo 1>&2
    usage
    exit 1
fi

kubectl create namespace edb-hcp-operator-system

helm repo add enterprisedb-edbpgai "https://downloads.enterprisedb.com/${password}/$
↪{RELEASE}pgai-platform/helm/charts/"
helm repo update

kubectl create secret docker-registry edb-hcp-operator-pull-secret \
    --docker-server="${registry}" \
    --docker-username="${RELEASE}${username}" \
    --docker-password="${password}" \
    --namespace edb-hcp-operator-system

helm upgrade \
    -n edb-hcp-operator-system \
```

(次のページに続く)

(前のページからの続き)

```
    --install \
    --set "imagePullSecrets[0].name=edb-hcp-operator-pull-secret" \
    --set controllerManager.manager.image.tag=$OPERATOR_VERSION \
    --set controllerManager.manager.image.repository=docker.enterprisedb.com/$
↪{RELEASE}pgai-platform/edb-hcp-operator/manager \
    --version $OPERATOR_VERSION \
    edb-hcp-operator enterprisedb-edbpgai/edb-hcp-operator
```

- eks-install-secrets.sh - このスクリプトは、インストールプロセスに必要なシークレットをインストールします。

```
registry="docker.enterprisedb.com"
username="pgai-platform"
password=""

echo "Enter the password for ${username}@${registry}"
read -s password

if [ -z "$password" ]; then
    echo 1>&2 "Error: password is required" >&2
    echo 1>&2
    usage
    exit 1
fi

kubectl create ns edbpgai-bootstrap
kubectl create ns upm-replicator

kubectl create secret docker-registry edb-cred \
    -n edbpgai-bootstrap \
    --docker-server="${registry}" \
    --docker-username="${username}" \
    --docker-password="${password}"

kubectl create secret docker-registry edb-cred \
    -n upm-replicator \
    --docker-server="${registry}" \
    --docker-username="${username}" \
    --docker-password="${password}"

kubectl annotate secret -n upm-replicator edb-cred replicator.v1.mittwald.de/
↪replicate-to="*" --overwrite

kubectl create namespace upm-griptape
```

(次のページに続く)

(前のページからの続き)

```
FERNET_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)

kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
    name: fernet-secret
    namespace: upm-griptape
stringData:
    FERNET_KEY: ${FERNET_KEY}
EOF

kubectl create namespace upm-lakekeeper

PG_CONFOUNDING_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)

kubectl apply -f - <<EOF
  apiVersion: v1
  kind: Secret
  metadata:
    name: pg-confounding-key
    namespace: upm-lakekeeper
  stringData:
    PG_CONFOUNDING_KEY: ${PG_CONFOUNDING_KEY}
EOF

echo "PG_CONFOUNDING_KEY is ${PG_CONFOUNDING_KEY} - store safely"
```

- prepare-op.sh - このスクリプトは、インストールプロセスの値ファイルを準備します。

```
# !/bin/bash
echo "HM Operator values.yaml preparation\n"

#  Import the eks-default-env file

source ./default-env.sh

#  If EDB_PLATFORM_VERSION is not set, error and exit

if [ -z "$EDB_PLATFORM_VERSION" ]; then
  echo "EDB_PLATFORM_VERSION is not set, please set it to the version of EDB Platform␣
↪you are deploying"
  exit
fi
```

(次のページに続く)

(前のページからの続き)

```
#  Test for the presence of yq

if ! command -v yq &> /dev/null
then
    echo "yq could not be found, please install it"
    exit
fi

cp hm-default.yaml hm-values.yaml

#  Prepare values.yaml for EKS

export AES_256_KEY=$(openssl rand -base64 32)

#  if AUTHENTICATION_PASSWORD_HASH is not set, generate it

if [ -z "$AUTHENTICATION_PASSWORD_HASH" ]; then
  echo "Generating password hash - please ensure you remove the password from your␣
↪history"
  export AUTHENTICATION_PASSWORD_HASH=$(echo -n $AUTHENTICATION_PASSWORD | htpasswd -
↪BinC 10 admin | cut -d: -f2)
else
  echo "Using prehashed password"
fi

if [[ "$EDB_PLATFORM_VERSION" == v1.2.* ]]; then
    echo "EDB Platform version is v1.2.*"
    yq -i .spec.imageRegistry = env(CONTAINER_REGISTRY_URI) |
    .spec.componentsParameters.upm-istio-gateway.cookie_aeskey = env(AES_256_KEY) |
    .spec.globalParameters.portal_domain_name = env(PORTAL_DOMAIN_NAME) |
    .spec.globalParameters.storage_class = env(STORAGE_CLASS) |
    .spec.componentsParameters.transporter-data-operator.fips_enabled =␣
↪(env(TRANSPORTER_FIPS_ENABLED)|tostring) |
    .spec.componentsParameters.transporter-rw-service.domain_name = env(TRANSPORTER_
↪RW_SERVICE_DOMAIN_NAME) |
    .spec.componentsParameters.transporter-dp-agent.rw_service_url= "https://
↪"+env(TRANSPORTER_RW_SERVICE_DOMAIN_NAME)+"/transporter" |
    .spec.componentsParameters.upm-beacon.server_host= env(BEACON_SERVICE_DOMAIN_
↪NAME) |
    .spec.pgai.portal.authentication.staticPasswords[0].email=env(AUTHENTICATION_
↪EMAIL) |
    .spec.pgai.portal.authentication.staticPasswords[0].hash=env(AUTHENTICATION_
↪PASSWORD_HASH) |
    .spec.pgai.portal.authentication.staticPasswords[0].username=env(AUTHENTICATION_
↪USER) |
```

(次のページに続く)

(前のページからの続き)

```
    .spec.pgai.portal.authentication.staticPasswords[0].userID="c5998173-a605-449a-
↪a9a5-4a9c33e26df" |
    .spec.beaconAgent.provisioning.imagesetDiscoveryContainerRegistryURL=␣
↪env(IMAGESET_REGISTRY_URI) |
    .spec.beaconAgent.provisioning.imagesetDiscoveryAuthenticationType= env(IMAGESET_
↪AUTHTYPE) |
    .spec.beaconAgent.location=env(LOCATION_NAME) hm-values.yaml
fi

echo "Your hm-values.yaml file is ready"
```

- default-env.sh - このファイルには、インストールプロセスのデフォルトの環境変数が含まれています。

```
#  exports for EKS

export EDB_PLATFORM_VERSION="v1.2.0"
export CONTAINER_REGISTRY_URI="docker.enterprisedb.com/pgai-platform"
export IMAGESET_REGISTRY_URI=$CONTAINER_REGISTRY_URI
export IMAGESET_AUTHTYPE="token"
export PORTAL_DOMAIN_NAME="portal.foo.network"
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME="transporter.foo.network"
export BEACON_SERVICE_DOMAIN_NAME="beacon.foo.network"
export AUTHENTICATION_EMAIL="owner@mycompany.com"
export AUTHENTICATION_USER="owner@mycompany.com"
export LOCATION_NAME="default-location"
export STORAGE_CLASS="gp2"
export TRANSPORTER_FIPS_ENABLED=false

#  Set a password hash for the user or pass a password to have it hashed for you.

#  If you pass a password, you will need to remove it from your history.

#  You can hash a password using the following command:

#  echo -n "password" | htpasswd -BinC 10 admin | cut -d: -f2

#

#  export AUTHENTICATION_PASSWORD="password"

#  If setting AUTHENTICATION_PASSWORD_HASH, ensure it is single quoted to prevent␣
↪variable expansion (e.g. $2y$10$...).

export AUTHENTICATION_PASSWORD_HASH=$2y$10$vKOAXfLHbeV1OQxMpxlLdOIwnX.JAN.ZrD9ZU//
↪ocrNQwhQIMtXhy
```

- hm-default.yaml - このファイルには、構成のデフォルト値が含まれています。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  source:
    useLocalKustomizations: true
  flavour: eks
  version: v1.2.0+appl
  imageRegistry: ${CONTAINER_REGISTRY_URI}
  imagePullSecrets:
    - name: edb-cred
      namespace: edbpgai-bootstrap
  globalParameters:
    internal_backup_folder: 4befab698c8d
    portal_domain_name: ${PORTAL_DOMAIN_NAME}
    storage_class: ${STORAGE_CLASS}
  componentsParameters:
    edb-migration-portal:
      enable_pdb: "false"
    transporter-data-operator:
      fips_enabled: "${TRANSPORTER_FIPS_ENABLED}"
    transporter-dp-agent:
      rw_service_url: https://${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}/transporter
    transporter-rw-service:
      domain_name: ${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}
    upm-api-pnxt-metric:
      log_level: info
      tsdb_host: thanos-query.monitoring.svc.cluster.local
      tsdb_port: "9090"
    upm-api-pnxt-pd:
      log_level: info
    upm-beaco-ff-base:
      usage_generator_interval: 24h
    upm-beaco-usages:
      usage_generator_interval: 24h
    upm-beacon:
      server_host: ${BEACON_SERVICE_DOMAIN_NAME}
      beacon_server_replicas: "1"
      external_metrics_replicas: "1"
      beacon_db_hibernation: "\"off\""
    upm-istio-gateway:
      cookie_aeskey: ${AES_256_KEY}
    upm-prometheus:
```

(次のページに続く)

(前のページからの続き)

```
    memory: 4Gi
  pgai:
    portal:
      authentication:
        clientSecret: edb-postgres-ai-secret
        idpConnectors: []
        staticPasswords:
          - email: ${AUTHENTICATION_EMAIL}
            hash: ${AUTHENTICATION_HASH}
            username: ${AUTHENTICATION_USERNAME}
            userID: ${AUTHENTICATION_USER_ID}
  beaconAgent:
    applianceProviderEnabled: true
    plaintext: true
    beaconServerAddress: "upm-beacon-server-ingestion.upm-beacon.svc.cluster.
↪local:8080"
    tlsInsecure: false
    location: "devspatcher"
    logLevel: "debug"
    beaconDBConnected: true
    inCluster: true
    provisioning:
      loadBalancersEnabled: true
      nodePortDomain: "nodeportdomain.com"
      imageDiscovery: true
      imagesetDiscoveryAuthenticationType: "eks_managed_identity"
      imagesetDiscoveryContainerRegistryURL: 123456079902.dkr.ecr.us-east-1.amazonaws.
↪com
    providers:
      - appliance
```

すべてのファイルをローカルマシンのディレクトリにコピーし、cd をそのディレクトリにコピーします。

Hybrid Manager Operator のインストール

Hybrid Manager Operator をインストールするには、eks-install-operator.sh スクリプトを実行します。すでにトークンを環境変数として設定しているため、スクリプトにパイプできます。また、環境変数 OPERATOR_VERSION をインストールするオペレーターのバージョン、たとえば v1.2.0 に設定する必要があります。これは通常、インストールする EDB Hybrid Manager プラットフォームと同じバージョンであるため、次のように設定できます。

```
export OPERATOR_VERSION=$EDB_PLATFORM_VERSION
```

次に、スクリプトを実行してオペレーターをインストールできます。

```
echo $EDB_SUBSCRIPTION_TOKEN | $SHELL eks-install-operator.sh
```

シークレットのインストール

インストールプロセスでは、多数のシークレットを Kubernetes クラスターにインストールする必要があります。これらのシークレットは、EDB ダウンロードリポジトリで認証し、EDB ソフトウェア展開プラットフォームの資格情報を保存するために使用されます。

ブートストラッププロセスに必要なシークレットをインストールするには、install-secrets.sh スクリプトを実行します。トークンは環境変数として既に設定しているため、スクリプトにパイプできます。

```
echo $EDB_SUBSCRIPTION_TOKEN | $SHELL eks-install-secrets.sh
```

```
Enter the password for pgai-platform@docker.enterprisedb.com
namespace/edbpgai-bootstrap created
namespace/upm-replicator created
secret/edb-cred created
secret/edb-cred created
secret/edb-cred annotated
namespace/upm-griptape created
secret/fernet-secret created
namespace/upm-lakekeeper created
secret/pg-confounding-key created
PG_CONFOUNDING_KEY is 01234567890123456789ABCDEFG - store safely
```

このプロセスでは、ハイブリッドマネージャーを実行するために必要なシークレットを作成します。これらのシークレットの詳細については、次の場所で確認できます。

- ** *Enabling GenAI Builder on Hybrid Manager* GenAI Builder のオプショナル**
- ** *Enabling Data Catalogs on Hybrid Manager* カタログを使用する場合のオプショナル**
- ** *Customizing Migration Portal secrets for secure internal communication* 移行ポータルとの内部通信を保護するオプション** も参照してください。

値ファイルの準備

ブートストラッププロセスでは、Helm チャートで多くの値を設定する必要があります。これらの値は、values.yaml というファイルに保存されます。このファイルは、 default-env.sh ファイルの変数を使用する prepare.sh スクリプトを実行して作成できます。 default-env.sh ファイルを編集して、必要な値を設定します。

これは、デフォルトの default-env.sh ファイルです。

```
#  exports for EKS

export EDB_PLATFORM_VERSION="v1.2.0"
export CONTAINER_REGISTRY_URI="docker.enterprisedb.com/pgai-platform"
```

(次のページに続く)

(前のページからの続き)

```
export IMAGESET_REGISTRY_URI=$CONTAINER_REGISTRY_URI
export IMAGESET_AUTHTYPE="token"
export PORTAL_DOMAIN_NAME="portal.foo.network"
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME="transporter.foo.network"
export BEACON_SERVICE_DOMAIN_NAME="beacon.foo.network"
export AUTHENTICATION_EMAIL="owner@mycompany.com"
export AUTHENTICATION_USER="owner@mycompany.com"
export LOCATION_NAME="default-location"
export STORAGE_CLASS="gp2"
export TRANSPORTER_FIPS_ENABLED=false

#  Set a password hash for the user or pass a password to have it hashed for you.

#  If you pass a password, you will need to remove it from your history.

#  You can hash a password using the following command:

#  echo -n "password" | htpasswd -BinC 10 admin | cut -d: -f2

#

#  export AUTHENTICATION_PASSWORD="password"

#  If setting AUTHENTICATION_PASSWORD_HASH, ensure it is single quoted to prevent␣
↪variable expansion (e.g. $2y$10$...).

export AUTHENTICATION_PASSWORD_HASH=$2y$10$vKOAXfLHbeV1OQxMpxlLdOIwnX.JAN.ZrD9ZU//
↪ocrNQwhQIMtXhy
```

次に、 `default-env.sh` ファイルの変数の簡単なガイドを示します。

Variable	Description
CONTAINER_REGISTRY_UR	使用するコンテナレジストリの URI。これは EDB ダウンロードリポジトリに設定されます。
IMAGESET_REGISTRY_URI	イメージセットレジストリの URI。これは EDB ダウンロードリポジトリに設定されます。
IMAGESET_AUTHTYPE	イメージセットレジストリの認証タイプ。 EDB ダウンロードリポジトリでの認証にトークンを使用するため、トークンに設定します。
TRANSPORTER_RW_SERVICE_	Transporter サービスのドメイン名。
PORTAL_DOMAIN_NAME	ポータルのドメイン名。
BEACON_SERVICE_DOMAI	ビーコンサービスのドメイン名。

次のページに続く

表 2 – 前のページからの続き

Variable	Description
AUTHENTICATION_EMAIL	作成するユーザーのメールアドレス。
AUTHENTICATION_USER	作成するユーザーのユーザー名。
AUTHENTICATION_PASSWORD	作成するユーザーのパスワード。これは、プレーンテキストとして、または AUTHENTICATION_PASSWORD_HASH を使用したハッシュとして設定できます。
AUTHENTICATION_PASSWORD_HAS	作成するパスワードのハッシュ。
LOCATION_NAME	ビーコンエージェントのロケーション名。エージェントとしても知られます。
STORAGE_CLASS	インストールに使用するストレージクラス。これは、AWS の EBS ボリュームの gp2 に設定されます。
TRANSPORTER_FIPS_ENABLE	Transporter サービスの FIPS モードを有効にするかどうか。

prepare.sh スクリプトを実行して、 values.yaml ファイルを作成します。

```
$SHELL prepare.sh
```

これにより、default-env.sh ファイルで設定された値を使用して values.yaml ファイルが作成されます。

`hcp-で始める必要があります。ブートストラッププロセス用に 2 つの値を作成する必要があります。これらは AES_256_KEY と AUTHENTICATION_PASSWORD_HASH です。次のコマンドを使用してこれらの値を作成できます。

```
export AES_256_KEY=$(openssl rand -base64 32)
```

ハッシュするパスワードがある場合は、次のコマンドで行うことができます。

```
export AUTHENTICATION_PASSWORD_HASH=$(echo -n $AUTHENTICATION_PASSWORD | htpasswd -
↪BinC 10 admin | cut -d: -f2)
```

次に、values.yaml ファイルを編集し、インストールに必要な値を設定する必要があります。values.yaml ファイルで適切なキーを見つけ、次のように値を設定します。$<name> エントリを、同じ名前の環境変数を設定した値に置き換えます。

1. spec.componentsParameters.upm-istio-gateway.cookie_aeskey を$AES_256_KEY に設定します。
2. spec.imageRegistry を$CONTAINER_REGISTRY_URI に設定します。
3. spec.globalParameters.portal_domain_name を$PORTAL_DOMAIN_NAME に設定します。
4. spec.componentsParameters.transporter-rw-service.domain_name を$TRANSPORTER_RW_SERVICE_DOMAIN_NAME に設定します。
5. spec.componentsParameters.transporter-dp-agent.rw_service_url を https:// 、続いて

$TRANSPORTER_RW_SERVICE_DOMAIN_NAME、/transporter に設定します。たとえば、上記の値を指定すると、これは https://transporter.foo.network/transporter になります。

6. spec.componentsParameters.upm-beacon.server_host を$BEACON_SERVICE_DOMAIN_NAME に設定します。
7. spec.pgai.portal.authentication.staticPasswords[0].email を$AUTHENTICATION_EMAIL に設定します。
8. spec.pgai.portal.authentication.staticPasswords[0].hash を$AUTHENTICATION_PASSWORD_HASH に設定します。
9. spec.pgai.portal.authentication.staticPasswords[0].username を$AUTHENTICATION_USER に設定します。
10. spec.pgai.portal.authentication.staticPasswords[0].userID “c5998173-a605-449a-a9a5-4a9c33e26df”を設定します。
11. spec.beaconAgent.location を$LOCATION_NAME に設定します
12. spec.globalParameters.storage_class を$STORAGE_CLASS に設定します。
13. spec.componentsParameters.transporter-data-operator.fips_enabled を$TRANSPORTER_FIPS_ENABLED に設定します。ブール値は文字列として処理されるため、引用符で囲まれていることを確認します。

hm-values.yaml ファイルには、インストールプロセスに必要な値が含まれています。

Hybrid Manager プラットフォームのインストール

次の手順は、前の手順で作成した値ファイルをハイブリッドマネージャーオペレーターに適用することです。これにより、EKS クラスターへの Hybrid Manager プラットフォームのインストールがトリガーされます。

```
kubectl apply -f ./hm-values.yaml
```

次を実行して、インストールのステータスを確認できます。

```
kubectl get hybridcontrolplanes.edbpgai.edb.com
```

完了すると、出力はこれに似ているはずです。

```
NAME      PHASE
edbpgai   deployed
```

デプロイしない場合は、次を実行します。

```
kubectl get hybridcontrolplanes.edbpgai.edb.com -o yaml
```

インストールのステータスの詳細については、Hybrid Manager Operator のログを確認できます。

/etc/hosts で IP アドレスを構成します

/etc/hosts ファイルで構成するイングレスゲートウェイの IP アドレスを取得します。実行します

```
kubectl -n istio-system get service istio-ingressgateway -o jsonpath={.status.
↪loadBalancer.ingress[0].hostname} | strings | nslookup
```

このコマンドは、次のような出力を結果ます。

```
Server:         192.168.111.1
Address:        192.168.111.1#53

Non-authoritative answer:
Name:   k8s-istiosys-istioing-b3dc9b4f3c-81dc5c45b1d5f6d2.elb.us-east-1.amazonaws.com
Address: 52.70.180.102
```

この時点で、ポータルの IP アドレスを取得します。この IP アドレスを DNS サービスに追加し、default-env.sh ファイルこの例では portal.foo.network に設定したドメイン名にマップできます。

DNS サービスを構成していない場合は、IP アドレスをローカルの/etc/hosts ファイルに追加することを検討します。これにより、default-env.sh ファイルで設定したドメイン名を使用してハイブリッドマネージャーポータルにアクセスできます。

次に、次のように IP アドレスを/etc/hosts ファイルに追加できます。

```
echo "52.70.180.102 portal.foo.network" | sudo tee -a /etc/hosts
```

IP アドレスを、前に取得した値に置き換えます。

次のステップ

connecting to the HM Console によるインストールを検証できるようになりました。

Installing Hybrid Manager on AWS EKS

Hybrid ManagerHM は Amazon EKS にインストールできます。

> **重要**
>
> インストールプロセスのこの部分を開始する前に、 **前提条件** を完了します。

インストールするバージョンを設定します

EDB Software Deployment プラットフォームのバージョンは、環境変数 EDB_PLATFORM_VERSION によって設定されます。これは、インストールするプラットフォームのバージョンに設定する必要があります。たとえば、バージョン v1.2.0 をインストールするには、次のように環境変数を設定します。

```
export EDB_PLATFORM_VERSION="v1.2.0"
```

トークンを設定します

このインストールプロセスでは、EDB サブスクリプショントークンを環境変数として保存する必要があります。EDB Account Profile に移動して、取得できます。プロンプトが表示されたら、ログインします。そこに、Repos 2.0 トークンのエントリがあります。

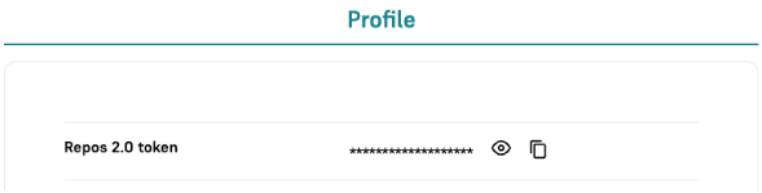


図 2: Repo Token

この値を取得して、環境変数として設定します。

```
export EDB_SUBSCRIPTION_TOKEN="your-token"
```

ブートストラップキットを入手する

Hybrid Manager は、ブートストラップキットを使用してインストールされます。このキットは、EDB Software Deployment プラットフォームのインストールに使用されるスクリプトと Helm チャートのコレクションです。

- eks-install-secrets.sh

```
registry="docker.enterprisedb.com"
username="pgai-platform"
password=""

echo "Enter the password for ${username}@${registry}"
read -s password

if [ -z "$password" ]; then
    echo 1>&2 "Error: password is required" >&2
    echo 1>&2
    usage
    exit 1
fi

kubectl create ns edbpgai-bootstrap
kubectl create ns upm-replicator

kubectl create secret docker-registry edb-cred \
    -n edbpgai-bootstrap \
    --docker-server="${registry}" \
    --docker-username="${username}" \
    --docker-password="${password}"
```

(次のページに続く)

(前のページからの続き)

```
kubectl create secret docker-registry edb-cred \
    -n upm-replicator \
    --docker-server="${registry}" \
    --docker-username="${username}" \
    --docker-password="${password}"

kubectl annotate secret -n upm-replicator edb-cred replicator.v1.mittwald.de/
↪replicate-to="*" --overwrite

kubectl create namespace upm-griptape

FERNET_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)

kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
    name: fernet-secret
    namespace: upm-griptape
stringData:
    FERNET_KEY: ${FERNET_KEY}
EOF

kubectl create namespace upm-lakekeeper

PG_CONFOUNDING_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)

kubectl apply -f - <<EOF
  apiVersion: v1
  kind: Secret
  metadata:
    name: pg-confounding-key
    namespace: upm-lakekeeper
  stringData:
    PG_CONFOUNDING_KEY: ${PG_CONFOUNDING_KEY}
EOF

echo "PG_CONFOUNDING_KEY is ${PG_CONFOUNDING_KEY} - store safely"
```

- prepare.sh

```
# !/bin/bash
echo "HM Helm values.yaml preparation\n"
```

(次のページに続く)

(前のページからの続き)

```
##  If EDB_PLATFORM_VERSION is not set, error and exit

if [ -z "$EDB_PLATFORM_VERSION" ]; then
  echo "EDB_PLATFORM_VERSION is not set, please set it to the version of EDB Platform␣
↪you are deploying"
  exit
fi

##  Test for the presence of yq

if ! command -v yq &> /dev/null
then
  echo "yq could not be found, please install it"
  exit
fi

##  Import the eks-default-env file

source ./default-env.sh

##  Test for presence of helm repo

value=`helm repo list -o yaml | yq eval contains([{ "name":"edbpgai"}]) -`
if [[ "$value" == "false" ]]; then
  echo "EDB Platform Helm repository not found, adding in now"
  helm repo add edbpgai "https://downloads.enterprisedb.com/${EDB_SUBSCRIPTION_TOKEN}/
↪pgai-platform/helm/charts/"
fi

##  Always update the helm repo

helm repo update
helm show values edbpgai/edbpgai-bootstrap --version $EDB_PLATFORM_VERSION > values.
↪yaml
echo "\n\n"

##  Prepare version dependent values

export AES_256_KEY=$(openssl rand -base64 32)

if [[ "$EDB_PLATFORM_VERSION" == v1.2.* ]]; then
    echo "EDB Platform version is v1.2.*"
    yq -i .bootstrapImageName = env(CONTAINER_REGISTRY_URI)+"/edbpgai-bootstrap/
↪bootstrap-"+env(EDB_TARGET_PLATFORM) |
    .bootstrapImageTag=env(EDB_PLATFORM_VERSION) |
```

(次のページに続く)

(前のページからの続き)

```
    del(.bootstrapImage) |
    .parameters.upm-istio-gateway.cookie_aeskey = env(AES_256_KEY) |
    del(.parameters.upm-beacon-ff-base) |
    .beaconAgent.provisioning.imagesetDiscoveryContainerRegistryURL= env(IMAGESET_
↪REGISTRY_URI) |
    .beaconAgent.provisioning.imagesetDiscoveryContainerRegistryAuthType=␣
↪env(IMAGESET_AUTHTYPE) |
    .containerRegistryURL = env(CONTAINER_REGISTRY_URI) values.yaml
fi

##  Prepare values.yaml for EKS

##  if AUTHENTICATION_PASSWORD_HASH is not set, generate it

if [ -z "$AUTHENTICATION_PASSWORD_HASH" ]; then
  echo "Generating password hash - please ensure you remove the password from your␣
↪history"
  export AUTHENTICATION_PASSWORD_HASH=$(echo -n $AUTHENTICATION_PASSWORD | htpasswd -
↪BinC 10 admin | cut -d: -f2)
else
  echo "Using prehashed password"
fi

yq -i .system = env(EDB_TARGET_PLATFORM) |
.parameters.global.portal_domain_name = env(PORTAL_DOMAIN_NAME) |
.parameters.transporter-rw-service.domain_name = env(TRANSPORTER_RW_SERVICE_DOMAIN_
↪NAME) |
.parameters.transporter-dp-agent.rw_service_url= "https://"+env(TRANSPORTER_RW_
↪SERVICE_DOMAIN_NAME)+"/transporter" |
.parameters.upm-beacon.server_host= env(BEACON_SERVICE_DOMAIN_NAME) |
.pgai.portal.authentication.staticPasswords[0].email=env(AUTHENTICATION_EMAIL) |
.pgai.portal.authentication.staticPasswords[0].hash=env(AUTHENTICATION_PASSWORD_HASH)␣
↪|
.pgai.portal.authentication.staticPasswords[0].username=env(AUTHENTICATION_USER) |
.pgai.portal.authentication.staticPasswords[0].userID="c5998173-a605-449a-a9a5-
↪4a9c33e26df" |
.beaconAgent.location=env(LOCATION_NAME) values.yaml

echo "Your values.yaml file is ready"
```

- default-env.sh

```
##  exports for EKS

export EDB_PLATFORM_VERSION="v1.2.0"
```

(次のページに続く)

(前のページからの続き)

```
export EDB_TARGET_PLATFORM="eks"
export CONTAINER_REGISTRY_URI="docker.enterprisedb.com/pgai-platform"
export IMAGESET_REGISTRY_URI=$CONTAINER_REGISTRY_URI
export IMAGESET_AUTHTYPE="token"
export PORTAL_DOMAIN_NAME="portal.foo.network"
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME="transporter.foo.network"
export BEACON_SERVICE_DOMAIN_NAME="beacon.foo.network"
export AUTHENTICATION_EMAIL="owner@mycompany.com"
export AUTHENTICATION_USER="owner@mycompany.com"
export LOCATION_NAME="default-location"

#  Set a password hash for the user or pass a password to have it hashed for you.

#  If you pass a password, you will need to remove it from your history.

#  You can hash a password using the following command:

#  echo -n "password" | htpasswd -BinC 10 admin | cut -d: -f2

#

#  export AUTHENTICATION_PASSWORD="password"

#  If setting AUTHENTICATION_PASSWORD_HASH, ensure it is single quoted to prevent␣
↪variable expansion (e.g. $2y$10$...).

export AUTHENTICATION_PASSWORD_HASH=$2y$10$vKOAXfLHbeV1OQxMpxlLdOIwnX.JAN.ZrD9ZU//
↪ocrNQwhQIMtXhy
```

すべてのファイルをローカルマシンのディレクトリにコピーし、cd をそのディレクトリにコピーします。

シークレットのインストール

ブートストラッププロセスでは、Kubernetes クラスターに多数のシークレットをインストールする必要があります。これらのシークレットは、EDB ダウンロードリポジトリで認証し、EDB ソフトウェア展開プラットフォームの資格情報を保存するために使用されます。

ブートストラッププロセスに必要なシークレットをインストールするには、install-secrets.sh スクリプトを実行します。トークンは環境変数として既に設定しているため、スクリプトにパイプできます。

```
echo $EDB_SUBSCRIPTION_TOKEN | $SHELL eks-install-secrets.sh
```

```
Enter the password for pgai-platform@docker.enterprisedb.com
Creating secret edb-cred
namespace/upm-replicator configured
secret/edb-cred configured
```

(次のページに続く)

(前のページからの続き)

```
namespace/edbpgai-bootstrap configured
secret/edb-cred configured
secret/edb-cred annotated
PG_CONFOUNDING_KEY is 01234567890123456789ABCDEFG - store safely
Installation completed
```

このプロセスでは、ハイブリッドマネージャーを実行するために必要なシークレットを作成します。これらのシークレットの詳細については、次の場所で確認できます。

- ** *Enabling GenAI Builder on Hybrid Manager* GenAI Builder のオプショナル**
- ** *Enabling Data Catalogs on Hybrid Manager* カタログを使用する場合のオプショナル**
- ** *Customizing Migration Portal secrets for secure internal communication* 移行ポータルとの内部通信を保護するオプション** も参照してください。

値ファイルの準備

ブートストラッププロセスでは、Helm チャートで多くの値を設定する必要があります。これらの値は、values.yaml というファイルに保存されます。このファイルは、 default-env.sh ファイルの変数を使用する prepare.sh スクリプトを実行して作成できます。 default-env.sh ファイルを編集して、必要な値を設定します。

これは、デフォルトの default-env.sh ファイルです。

```
#  exports for EKS

export EDB_PLATFORM_VERSION="v1.2.0"
export EDB_TARGET_PLATFORM="eks"
export CONTAINER_REGISTRY_URI="docker.enterprisedb.com/pgai-platform"
export IMAGESET_REGISTRY_URI=$CONTAINER_REGISTRY_URI
export IMAGESET_AUTHTYPE="token"
export PORTAL_DOMAIN_NAME="portal.foo.network"
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME="transporter.foo.network"
export BEACON_SERVICE_DOMAIN_NAME="beacon.foo.network"
export AUTHENTICATION_EMAIL="owner@mycompany.com"
export AUTHENTICATION_USER="owner@mycompany.com"
export LOCATION_NAME="default-location"

#  Set a password hash for the user or pass a password to have it hashed for you.

#  If you pass a password, you will need to remove it from your history.

#  You can hash a password using the following command:

#  echo -n "password" | htpasswd -BinC 10 admin | cut -d: -f2

```

(次のページに続く)

(前のページからの続き)

```
#

# export AUTHENTICATION_PASSWORD="password"

export AUTHENTICATION_PASSWORD_HASH="$2y$10$vKOAXfLHbeV1OQxMpxlLdOIwnX.JAN.ZrD9ZU//
↪ocrNQwhQIMtXhy"
```

次に、 default-env.sh ファイルの変数の簡単なガイドを示します。

Variable	Description
EDB_TARGET_PLATFOI	インストールのターゲットプラットフォーム。 Amazon EKS にインストールするため、eks に設定します。
CONTAINER_REGISTRY_UR	使用するコンテナレジストリの URI。これは EDB ダウンロードリポジトリに設定されます。
IMAGESET_REGISTRY_URI	イメージセットレジストリの URI。これは EDB ダウンロードリポジトリに設定されます。
IMAGESET_AUTHTYPE	イメージセットレジストリの認証タイプ。 EDB ダウンロードリポジトリでの認証にトークンを使用するため、トークンに設定します。
TRANSPORTER_RW_SERVICE_	Transporter サービスのドメイン名。
PORTAL_DOMAIN_NAME	ポータルのドメイン名。
BEACON_SERVICE_DOMAI	ビーコンサービスのドメイン名。
AUTHENTICATION_EMAIL	作成するユーザーのメールアドレス。
AUTHENTICATION_USER	作成するユーザーのユーザー名。
AUTHENTICATION_PASSWORD	作成するユーザーのパスワード。これは、プレーンテキストとして、または AUTHENTICATION_PASSWORD_HASH を使用したハッシュとして設定できます。
AUTHENTICATION_PASSWORD_HAS	作成するパスワードのハッシュ。
LOCATION_NAME	ビーコンエージェントのロケーション名。エージェントとしても知られます。

prepare.sh スクリプトを実行して、 values.yaml ファイルを作成します。

```
$SHELL prepare.sh
```

これにより、default-env.sh ファイルで設定された値を使用して values.yaml ファイルが作成されます。

次のコマンドを使用して Helm チャートリポジトリをインストールします。

```
helm repo add edbpgai "https://downloads.enterprisedb.com/${EDB_SUBSCRIPTION_TOKEN}/
↪pgai-platform/helm/charts"
```

チャートリポジトリが追加されたら、最新の状態であることを確認します。

```
helm repo update
```

これで、次のコマンドを使用して helm リポジトリから values.yaml ファイルを作成できます。

```
helm show values edbpgai/edbpgai-bootstrap > values.yaml
```

これで、ブートストラッププロセスのチャートのデフォルト値を含む values.yaml ファイルが完成しました。

ブートストラッププロセス用に 2 つの値を作成する必要があります。これらは AES_256_KEY と AUTHENTICATION_PASSWORD_HASH です。次のコマンドを使用してこれらの値を作成できます。

```
export AES_256_KEY=$(openssl rand -base64 32)
```

ハッシュするパスワードがある場合は、次のコマンドで行うことができます。

```
export AUTHENTICATION_PASSWORD_HASH=$(echo -n $AUTHENTICATION_PASSWORD | htpasswd -
↪BinC 10 admin | cut -d: -f2)
```

次に、values.yaml ファイルを編集し、インストールに必要な値を設定する必要があります。values.yaml ファイルで適切なキーを見つけ、次のように値を設定します。$<name> エントリを、同じ名前の環境変数を設定した値に置き換えます。

1. bootstrapImageName を $CONTAINER_REGISTRY_URI$、続いて「$/edbpgai-bootstrap/bootstrap-$」、次にEDB_TARGET_PLATFORM を設定します。たとえば、上記の値を指定すると、これは docker.enterprisedb.com/pgai-platform/edbpgai-bootstrap/bootstrap-eks になります。
2. bootstrapImageTag を$EDB_PLATFORM_VERSION に設定します。
3. parameters.upm-istio.cookie_aeskey を$AES_256_KEY に設定します。
4. containerRegistryURL を$CONTAINER_REGISTRY_URI に設定します。
5. system を$EDB_TARGET_PLATFORM に設定します。
6. parameters.global.portal_domain_name を$PORTAL_DOMAIN_NAME に設定します。
7. parameters.transporter-rw-service.domain_name を$TRANSPORTER_RW_SERVICE_DOMAIN_NAME に設定します。
8. parameters.transporter-dp-agent.rw_service_url を https:// 、続いて$TRANSPORTER_RW_SERVICE_DOMAIN_NAME、/transporter に設定します。たとえば、上記の値を指定すると、これは https://transporter.foo.network/transporter になります。
9. parameters.upm-beacon.server_host を $BEACON_SERVICE_DOMAIN_NAME$ に設定します。 10.‘$pgai.portal.authentication.staticPasswords[0].email$ を `AUTHENTICATION_EMAIL に設定します。11. pgai.portal.authentication.staticPasswords[0].hash を $AUTHENTICATION_PASSWORD_HASH$ に設定します。12.‘$pgai.portal.authentication.staticPasswords[0].username$ を `AUTHENTICATION_USER に

設定します。13. pgai.portal.authentication.staticPasswords[0].userID “c5998173-a605-449a-a9a5-4a9c33e26df”を設定します。 14. beaconAgent.location を$LOCATION_NAME に設定します

Helm チャートを適用する

values.yaml ファイルを準備したら、Helm チャートを適用してブートストラッププロセスを開始できます。これにより、Kubernetes クラスターに必要なリソースが作成され、ブートストラッププロセスを開始します。

```
helm upgrade -n edbpgai-bootstrap \
   --install -f ./values.yaml \
   --version "${EDB_PLATFORM_VERSION/-appl/+appl}" \
   edbpgai-bootstrap edbpgai/edbpgai-bootstrap
```

このコマンドは、Kubernetes クラスターにブートストラップコンテナをインストールします。ブートストラップコンテナは、EDB Software Deployment プラットフォームの残りの部分をインストールします。

ブートストラッププロセスに従います

ブートストラッププロセスが完了するまでに時間がかかります。ブートストラップコンテナのログを確認することにより、進行状況を追跡できます。

```
kubectl logs -f -l app=edbpgai-bootstrap -n edbpgai-bootstrap
```

/etc/hosts で IP アドレスを構成します

/etc/hosts ファイルで構成するイングレスゲートウェイの IP アドレスを取得します。実行します

```
kubectl -n istio-system get service istio-ingressgateway -o jsonpath={.status.
↪loadBalancer.ingress[0].hostname} | strings | nslookup
```

このコマンドは、次のような出力を結果ます。

```
Server:         192.168.111.1
Address:        192.168.111.1#53

Non-authoritative answer:
Name:   k8s-istiosys-istioing-b3dc9b4f3c-81dc5c45b1d5f6d2.elb.us-east-1.amazonaws.com
Address: 52.70.180.102
```

この時点で、ポータルの IP アドレスを取得します。この IP アドレスを DNS サービスに追加し、default-env.sh ファイルこの例では portal.foo.network に設定したドメイン名にマップできます。

DNS サービスを構成していない場合は、IP アドレスをローカルの/etc/hosts ファイルに追加することを検討します。これにより、default-env.sh ファイルで設定したドメイン名を使用してハイブリッドマネージャーポータルにアクセスできます。

次に、次のように IP アドレスを/etc/hosts ファイルに追加できます。

```
echo "52.70.180.102 portal.foo.network" | sudo tee -a /etc/hosts
```

IP アドレスを、前に取得した値に置き換えます。

次のステップ

connecting to the HM Console によるインストールを検証できるようになりました。

Connecting to the Hybrid Manager Console

EDB ソフトウェア展開プラットフォームの **シークレットをインストールに適用する** が成功したことを確認するには、ブラウザーを使用して HM コンソールに接続します。

ブラウザーに URL を入力して、ポータルに接続します。

```
https://portal.foo.network/
```

この構成では、ホスト名がより広範なインターネットに存在しないため、https は検証されません。証明書の警告を受け入れて、ポータルに進みます。

ログイン画面では、*Helm values file* または *Operator values file* を介してブートストラップインストールで生成されたユーザー名とパスワードを入力できます。詳細については、*Managing users in Hybrid Manager* を参照してください。

ログインしたら、 *Create a database cluster template* ができます。

Uninstall Hybrid Manager from EKS

EKS クラスターからのハイブリッドマネージャーのアンインストールは、インストール方法によって異なります。

ブートストラップでインストールされたハイブリッドマネージャーを使用したアンインストール

ブートストラップインストール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た場合、次のコマンドを実行してアンインストールできます。

```
helm uninstall -n edbpgai--n edbpgai-bootstrap edbpgai-bootstrap edbpgai/edbpgai-
↪bootstrap
```

このコマンドは、ブートストラップ方法を使用してインストールされたハイブリッドマネージャーコンポーネントを削除します。

Hybrid Manager Operator を使用したアンインストール

Hybrid Manager Operator をアンインストールするには、次のコマンドを実行できます。

```
##  by default, EDB Postgres AI is protected from the deletion, disable the␣
↪protection first
```

(次のページに続く)

(前のページからの続き)

```
kubectl annotate hybridcontrolplane edbpgai biganimal.enterprisedb.io/deletion-
↪protect=disabled --overwrite

##  uninstall the EDB Postgres AI

kubectl delete hybridcontrolplane edbpgai --wait=false

##  monitor the uninstallation process

kubectl get hybridcontrolplane edbpgai -o yaml
```

これにより、Hybrid Manager Operator とすべての関連リソースが EKS クラスターから削除されます。

アンインストール後

Hybrid Manager をアンインストールした後、EKS クラスターには、インストールの副作用として作成されたリソースがまだ保持されていることに注意してください。これらのリソースは、必要がなくなった場合は手動で削除したり、将来の使用のために保存したり、EKS クラスター自分自身を削除したりできます。これにより、それに関連付けられているすべてのリソースが削除されます。

4.3.4 Install Hybrid Manager on GCP GKE

Hybrid ManagerHM は Google Cloud GKE にインストールできます。

- 前提条件
- *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator*
- *Connecting to the portal*

Hybrid Manager installation methods for Google Cloud GKE

GCP GKE にハイブリッドマネージャーをインストールするために使用できる方法は 2 つあります。ユースケースに合った方法を選択してください

- *Helm* チャートを適用する を使用する 。 Hybrid Manager でのテストと実験には Helm インストールを使用します。Helm インストールは、非実稼働のユースケースに適しています。
- *Install Hybrid Manager on Google Cloud GKE using EDB Postgres AI Operator* を使用する。運用でハイブリッドマネージャーを使用する準備ができている場合は、 EDB Postgres AI Operator インストールを使用します。

Install Hybrid Manager on GCP GKE using Helm

Helm インストールの環境変数を設定する

1. インストールの準備をするには、コンテナレジストリの URI を指定します。

```
export CONTAINER_REGISTRY_URI=<local-registry-URI>
```

2. edb-bootstrap イメージバージョンを設定します。

```
export EDBPGAI_BOOTSTRAP_IMAGE_VERSION="<edb-bootstrap-image-version"
```

<edb-bootstrap-image-version> の値を、このインストールのターゲットとしているイメージバージョンに変更してください。

3. edb-bootstrap Helm チャートバージョンを設定します。

```
export EDBPGAI_BOOTSTRAP_HELM_CHART_VERSION="<edbpgai-bootstrap-helm-version>"
```

<edbpgai-boostrap-helm-version> の値を、このインストールのターゲットとしている edb-pgai Helm チャートバージョンに変更してください。

4. EDB Cloudsmith トークンを設定します。

```
export EDB_CLOUDSMITH_TOKEN=<edb-cloudsmith-token>
```

5. edb-cloudsmith-token を EDB Cloudsmith トークンに変更します。

注釈

Hybrid Manager バージョン v1.1.0 以降の場合、変数 `upm-istio.cooke_aeskey` および `upm-istio.openshift_console_domain_name` を使用します。 HM バージョンの正しい変数を使用していることを確認し、[Helm upgrade command](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ます) の他のパラメーターを削除します。

環境変数が設定されたら、インストールの準備をできます。

インストールの準備

1. EDB Cloudsmith から Helm チャートリポジトリを追加します。

```
helm repo add enterprisedb-edbpgai "https://downloads.enterprisedb.com/${EDB_
↪CLOUDSMITH_TOKEN}/pgai-platform/helm/charts/"
```

2. 追加したリポジトリを更新します。

```
helm repo update
```

3. デフォルトの edbpgai-bootstrap Helm values.yaml ファイルを取得します。

```
helm show values enterprisedb-edbpgai/edbpgai-bootstrap > values.yaml
```

4. values.yaml ファイルを正しい値に編集し、必要に応じて前に設定した環境変数を置き換えます。

```
system: "gke"
containerRegistryURL: "${CONTAINER_REGISTRY_URI}"
bootstrapImageName: "${CONTAINER_REGISTRY_URI}/edbpgai-bootstrap/bootstrap-gke"
bootstrapImageTag: "${EDBPGAI_BOOTSTRAP_IMAGE_VERSION}"
  global:
    default_ingress_domain: "${DEFAULT_INGRESS_DOMAIN}"
    portal_domain_name: "${PORTAL_DOMAIN_NAME}"
  transporter-rw-service:
    domain_name: "${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}"
  transporter-dp-agent:
    rw_service_url: "https://${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}/transporter"
  transporter-data-operator:
    fips_enabled: ${TRANSPORTER_FIPS_ENABLED}
  upm-beacon:
    server_host: "${BEACON_SERVICE_DOMAIN_NAME}"
  upm-istio-gateway:
    cookie_aeskey: "${AES_256_KEY}"
pgai:
  portal:
    authentication:
      staticPasswords:
        - email: "${AUTHENTICATION_EMAIL}"
          hash: "${AUTHENTICATION_HASH}"
          username: "${AUTHENTICATION_USERNAME}"
          userID: "${AUTHENTICATION_USER_ID}"
```

動的画像検出を有効にする

レジストリからの動的イメージの検出を有効にするには、必要なオプションで構成する必要があります。これを行う方法の詳細な手順については、Image discovery configuration を参照してください。

必要なイメージ検出構成を特定した後、適切な設定で `values.yaml` を更新します。

たとえば、次は、 `"basic"` 認証を使用して `us-east1` リージョンで Google Artifact Registry を使用するサンプル構成です。

```
beaconAgent:
  location: "devspatcher"
  beaconDBConnected: true
  plaintext: true
  logLevel: "debug"
  inCluster: true
  provisioning:
    provider: "gcp"
    imageDiscovery: true
    loadBalancersEnabled: true
```

(次のページに続く)

(前のページからの続き)

```
    imagesetDiscoveryAuthenticationType: "basic"
    imagesetDiscoveryContainerRegistryURL: "us-east1-docker.pkg.dev"
  providers:
    - appliance
beaconServer:
  logLevel: "debug"
```

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する

すべての準備ができたら、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できます。

```
helm upgrade \
    -n edbpgai-bootstrap \
    --install \
    -f ./values.yaml \
    --version "${EDBPGAI_BOOTSTRAP_HELM_CHART_VERSION}" \
    edbpgai-bootstrap enterprisedb-edbpgai/edbpgai-bootstrap
```

次のコマンドを使用して、インストールプロセスを監視できます。20〜30 分ほどかかります。

```
kubectl logs -f -l app=edbpgai-bootstrap -n edbpgai-bootstrap
```

インストールが正常に完了すると、 *update DNS zone records and verify the install* .

Install Hybrid Manager on Google Cloud GKE using EDB Postgres AI Operator

インストールの準備

1. オペレーターを取得するための認証には、次の環境変数が必要です。

```
export EDB_CLOUDSMITH_TOKEN=<edb-cloudsmith-token>
```

1. オペレーターの新しい名前空間を作成します。

```
kubectl create namespace edb-hcp-operator-system
```

1. オペレーターの Helm チャートリポジトリを追加し、更新します。

```
helm repo add enterprisedb-edbpgai "https://downloads.enterprisedb.com/${EDB_
↪CLOUDSMITH_TOKEN}/pgai-platform/helm/charts/"
helm repo update
```

1. 適切なバージョン、この場合 1.2 を使用してインストールを続行します。

```
export OPERATOR_VERSION=1.2.0
```

GKE にオペレーターをインストールする

ネットワークのユースケースの指示を使用して、オペレーターをインストールします。

1. GKE に EDB Postgres AI Operator をインストールするには、最初に EDB Cloudsmith リポジトリにアクセスするためのイメージプルシークレットを準備していることを確認します。また、リポジトリ docker.enterprisedb.com/pgai-platform/edb-hcp-operator/manager からイメージをプルするために必要な権限があることを確認します。

```
export DOCKER_SERVER="docker.enterprisedb.com"
export DOCKER_USERNAME="<your-docker-user-name>"
export DOCKER_PASSWORD="<your-docker-password>"
```

1. オペレーターのイメージプルシークレットを作成します。

```
kubectl create secret docker-registry edb-hcp-operator-pull-secret \
    --docker-server="${DOCKER_SERVER}" \
    --docker-username="${DOCKER_USERNAME}" \
    --docker-password="${DOCKER_PASSWORD}" \
    --namespace edb-hcp-operator-system
```

1. オペレーターをインストールします。

```
helm upgrade \
    -n edb-hcp-operator-system \
    --install \
    --set "imagePullSecrets[0].name=edb-hcp-operator-pull-secret" \
    --set controllerManager.manager.image.tag=$OPERATOR_VERSION \
    --set controllerManager.manager.image.repository=docker.enterprisedb.com/pgai-
↪platform/edb-hcp-operator/manager \
    --version $OPERATOR_VERSION \
    edb-hcp-operator enterprisedb-edbpgai/edb-hcp-operator
```

オペレーターを使用してハイブリッドマネージャーをインストールする

YAML を適用してインストールする

EDB Postgres AI Operator を GKE クラスターにインストールした後、Operator を使用して Hybrid Manager をインストールする必要があります。次の YAML ファイルは、ハイブリッドマネージャーのインストールを開始します。

```
apiVersion: edbpgai.edb.com/v1alpha1
kind: HybridControlPlane
metadata:
  name: edbpgai
spec:
  source:
    useLocalKustomizations: true
```

(次のページに続く)

(前のページからの続き)

```
  flavour: gke
  version: v1.2.0+appl
  imageRegistry: ${CONTAINER_REGISTRY_URI}
  imagePullSecrets:
    - name: edb-cred
      namespace: edbpgai-bootstrap
  globalParameters:
    internal_backup_folder: 4befab698c8d
    portal_domain_name: ${PORTAL_DOMAIN_NAME}
    storage_class: ${STORAGE_CLASS}
  componentsParameters:
    edb-migration-portal:
      enable_pdb: "false"
    transporter-data-operator:
      fips_enabled: "${TRANSPORTER_FIPS_ENABLED}"
    transporter-dp-agent:
      rw_service_url: https://${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}/transporter
    transporter-rw-service:
      domain_name: ${TRANSPORTER_RW_SERVICE_DOMAIN_NAME}
    upm-api-pnxt-metric:
      log_level: info
      tsdb_host: thanos-query.monitoring.svc.cluster.local
      tsdb_port: "9090"
    upm-api-pnxt-pd:
      log_level: info
    upm-beaco-ff-base:
      usage_generator_interval: 24h
    upm-beaco-usages:
      usage_generator_interval: 24h
    upm-beacon:
      server_host: ${BEACON_SERVICE_DOMAIN_NAME}
      beacon_server_replicas: "1"
      external_metrics_replicas: "1"
      beacon_db_hibernation: "\"off\""
    upm-istio-gateway:
      cookie_aeskey: ${AES_256_KEY}
    upm-prometheus:
      memory: 4Gi
  pgai:
    portal:
      authentication:
        clientSecret: edb-postgres-ai-secret
        idpConnectors: []
        staticPasswords:
          - email: ${AUTHENTICATION_EMAIL}
```

(次のページに続く)

(前のページからの続き)

```
            hash: ${AUTHENTICATION_HASH}
            username: ${AUTHENTICATION_USERNAME}
            userID: ${AUTHENTICATION_USER_ID}
  beaconAgent:
    applianceProviderEnabled: true
    plaintext: true
    beaconServerAddress: "upm-beacon-server-ingestion.upm-beacon.svc.cluster.
↪local:8080"
    tlsInsecure: false
    location: "devspatcher"
    logLevel: "debug"
    beaconDBConnected: true
    inCluster: true
    provisioning:
      loadBalancersEnabled: true
      nodePortDomain: "nodeportdomain.com"
      imageDiscovery: true
      imagesetDiscoveryAuthenticationType: "basic"
      imagesetDiscoveryContainerRegistryURL: "us-east1-docker.pkg.dev"
    providers:
      - appliance
```

YAML が適用された後、次のコマンドを使用してインストールステータスを監視できます。

```
kubectl get hybridcontrolplane edbpgai -o yaml
```

インストールが現在いるフェーズに応じて、3 つの異なるステータスを返すことができます。

```
- deployed # the installation was successful

- deploying # the installation is still in progress

- failed: # the install was unsuccessful
```

インストールが成功すると、 *updating DNS zone records and verify the install* ** .

ハイブリッドマネージャーのアンインストール

EDB Postgres AI Operator 対 *Helm* チャートを適用する を使用して Hybrid Manager をインストールした場合、アンインストールできます。

> **警告**
>
> 次の手順は、Hybrid Manager をアンインストールします。 Hybrid Manager をアンインストールする場合にのみ、この手順を使用します。

ハイブリッドマネージャーを削除する必要がある場合があります。ハイブリッドマネージャーはデフォルトで削除から保護されているため、最初にその保護を無効にする必要があります。

```
kubectl annotate hybridcontrolplane edbpgai biganimal.enterprisedb.io/deletion-
↪protect=disabled --overwrite
```

次に、アンインストールできます。

```
kubectl delete hybridcontrolplane edbpgai --wait=false
```

プロセスをモニタするには、次を実行します。

```
kubectl get hybridcontrolplane edbpgai -o yaml
```

Updating DNS zone records

Helm または Operator のインストールが完了したら、DNS ゾーンレコードを更新し、ハイブリッドマネージャーが予想通りに動作していることを確認します。

DNS ゾーンレコードの更新

1. ポータルサービスのホスト名を取得します。このホスト名は、ハイブリッドマネージャーの入力エンドポイントを表します。

```
kubectl get svc -n istio-system istio-ingressgateway -o yaml | yq .status.
↪loadBalancer.ingress[0].hostname
```

1. 次のドメイン名の DNS ゾーンに CNAME レコードを作成し、取得したホスト名をポイントします。
    - PORTAL_DOMAIN_NAME
    - TRANSPORTER_RW_SERVICE_DOMAIN_NAME
    - BEACON_SERVICE_DOMAIN_NAME

各 CNAME レコードは、ユーザー向けドメイン名を入力ホスト名にマッピングします。

これで、 *connect to the portal* を実行して、インストールを確認できます。

Prerequisites

GCP GKE には、EDB Postgres AI ソフトウェア展開をインストールする前に満たす前提条件があります。インストールプロセスを開始する前に、前提条件を満たす必要があることを確認します。

前提条件

- *A system from which you can run the installation process*
- 必要な権限を持つ GCP アカウント
- *A GCP cluster configured to host HM*

次のステップ

前提条件を完了したら、 *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator* に進むことができます。

Install system

インストールプロセスを実行できるシステムが必要です。このシステムには、次のソフトウェアをインストールする必要があります。

- kubectl Kubernetes クラスターを管理します。
- Helm Kubernetes マニフェストをレンダリングおよび適用します。

リンクされた手順、または以下にある特定のオペレーティングシステムの手順を使用して、これらのツールをインストールできます。

macOS

kubectl と Helm をインストールします。

```
brew install kubectl helm
```

Linux

=== 「Ubuntu」

> **注釈**
>
> これらの手順は、デフォルトでスナップサポートが利用できることを前提としているため、Ubuntu 16.04 以降を対象としています。

```
sudo snap install helm --classic
sudo snap install kubectl --classic
```

=== 「Red Hat」

kubectl をインストールします。

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.
↪txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
kubectl version --client
```

Helm をインストールします

```
curl -LO https://get.helm.sh/helm-v3.12.3-linux-amd64.tar.gz
tar -xvf helm-v3.12.3-linux-amd64.tar.gz
```

(次のページに続く)

(前のページからの続き)

```
sudo mv linux-amd64/helm /usr/local/bin/helm
helm version
```

=== “Debian”

```
sudo apt update
sudo apt install snapd
sudo snap install snapd
sudo snap install kubectl --classic
```

Windows

=== “Winget”

```
winget install kubectl --source winget
winget search helm
winget install helm
```

=== “手動”

1. kubectl をインストールします。
2. *Helm* **のインストール** をインストールします。

Helm のインストール

1. Windows の Helm バイナリをダウンロードします。 Helm GitHub Releases page で、最新リリースを見つけて、help-v3.xx-windows-amd64.zip ファイルをダウンロードします。
2. Zip ファイルを解凍します。7-Zip、WinRAR、またはビルトインの Windows 抽出ユーティリティなどのツールを使用して、ダウンロードしたファイルを解凍します。
3. `helm.exe` を、抽出したフォルダーからパス内のディレクトリにコピーします。一般的なディレクトリには、`C:\Program Files\` 、`C:\Program Files (x86)\` が含まれます。または、 `C:\Helm\` などのディレクトリを作成し、パスに追加できます。
4. インストールを確認します。
    1. 新しいコマンドプロンプトまたは PowerShell ウィンドウを開きます。
    2. 次のコマンドを実行して、Helm が正しくインストールされていることを確認します。

```
helm version
```

3. 出力には、Helm バージョン情報が表示されます。

次のステップ

必要なソフトウェアをインストールしたら、必要な権限を持つ GCP アカウントの設定に進むことができます。

Creating or configuring a GKE cluster for Hybrid Manager

GKE クラスター

Hybrid ManagerHM をインストールするには GKE クラスターが必要です。必要に応じて、GKE ドキュメントの Creating a regional cluster に関するドキュメントに従って、新しいものをセットアップします。

アクセス許可

`roles/container.admin` に `Kubernetes Engine Admin` 権限があることを確認します。

さまざまなノードタイプ

HM は、GKE 上のさまざまなノードのセットでの管理サービス HM コントロールプレーンと PostgreSQL ワークロードをサポートしていますが、追加のノードプールが必要です。

さまざまなノードセットを実装するには、次の手順を実行します。

HM コントロールプレーンノード

HM コントロールプレーンノードを、独自のノードタイプで実装できる論理グループを作成するには

1. ラベルで、次を設定します。

```
edbaiplatform.io/control-plane: "true"
```

1. テイントで、次を設定します。

```
key: edbaiplatform.io/control-plane
value: "true"
effect: NoSchedule
```

PostgreSQL ワークロードノード

Postgres ワークロードノードを、独自のノードタイプで実装できる論理グループにするには

1. ラベルで、次を設定します。

```
edbaiplatform.io/postgres: "true"
```

1. テイントで、次を設定します。

```
key: edbaiplatform.io/postgres
value: "true"
effect: NoSchedule
```

GKE 管理対象ノードグループのラベルとテイントを追加する方法の詳細については、ドキュメント here を参照してください。

オブジェクトストレージの構成

オブジェクトストレージバックエンド HM バックアップをサポートするには、クラスターを専用のオブジェクトストレージソリューション STORAGE_CLASS 、例、GCP バケットに接続します。

特に、ハイブリッドマネージャーはこのバケットを使用して以下を保存します。

- 管理対象ストレージの場所
- Postgres WAL + バックアップ
- ログ Postgres と内部サービスの両方
- メトリック Postgres と内部サービスの両方

GCP バケットを作成する

gcloud CLI をダウンロードしてインストールします。

1. GCP バケットを作成するための環境変数を設定します。

```
export GCP_PROJECT_ID="appliance-dev-444915" # change me to your own gcp project id
export GCP_BUCKET_NAME="tq-upm45906" # change me to your preferred bucket name
export GCP_BUCKET_REGION="us" # change me to your preferred location id
```

1. HM に使用する GCP バケットをホストするプロジェクトを設定します。

```
gcloud config set project $GCP_PROJECT_ID
```

1. バケットを作成します。

```
gcloud storage buckets create gs://$GCP_BUCKET_NAME --location=$GCP_BUCKET_REGION
```

バケットの階層名前空間が無効になっていることを確認します。

バケットユーザーを作成する

専用バケットを作成した後、バケットにアクセスするために必要な権限 roles/storage.admin を持つハイブリッドマネージャーのサービスアカウントを作成します。

1. 必要な環境変数を設定します。

```
export SERVICE_ACCOUNT_NAME="tq-upm45906" # change this to your preferred service␣
↪account name
export SERVICE_ACCOUNT_EMAIL=$SERVICE_ACCOUNT_NAME@$GCP_PROJECT_ID.iam.
↪gserviceaccount.com
```

1. サービスアカウントを作成します。

```
gcloud iam service-accounts create $SERVICE_ACCOUNT_NAME
```

1. バケットのサービスアカウントにストレージ管理ロールを割り当てます。

```
gcloud storage buckets add-iam-policy-binding gs://$GCP_BUCKET_NAME \
  --member="serviceAccount:$SERVICE_ACCOUNT_EMAIL" \
  --role="roles/storage.admin"
```

1. 次のユーザー権限で json キーを作成します。

```
gcloud iam service-accounts keys create key-file.json \
  --key-file-type=json \
  --iam-account=$SERVICE_ACCOUNT_EMAIL
```

1. base64 エンコードされた資格情報を取得します。

```
export GCP_CREDENTIAL_BASE64=$(base64 -w0 < key-file.json)
```

```
export GCP_CREDENTIAL_BASE64=$(base64 < key-file.json)
```

バケットアクセスのシークレットを作成する

バケットへの HM アクセスのための専用ユーザーを準備した後、作成したユーザーをハイブリッドマネージャーに関連付けるシークレットを作成して、オブジェクトストレージへのアクセスを提供します。

```
apiVersion: v1
kind: Secret
metadata:
  # The name must be edb-object-storage
  name: edb-object-storage
  # The namespace must be default
  namespace: default
stringData:
  # Specifies the cloud provider. In this case, it is set to gcp.
  provider: gcp
  # The region where the GCP bucket is located.
  location_id: ${GCP_BUCKET_REGION}
  # The ID of the GCP project.
  project_id: ${GCP_PROJECT_ID}
  # The name of the GCP bucket.
  bucket_name: ${GCP_BUCKET_NAME}
  # The base64 encoded JSON credentials for accessing the GCP bucket.
  credentials_json_base64: ${GCP_CREDENTIAL_BASE64}
```

AES-256 キーを作成する

認証用の AES-256 キーを生成します。

```
export AES_256_KEY=$(openssl rand -base64 32)
```

HM コンテナイメージをプライベートレジストリに同期する

HM を使用する要件は、独自の安全で承認された内部レジストリをホストする必要があることです。また、インストールする HM のバージョンを知る必要があります。同期プロセスは、基本的に Cloudsmith からすべてのアーティファクトを取得し、関連する Helm チャートで HM をインストールまたはアップグレードする前に、内部レジストリと内部同期することです。

同期プロセスは、コンテナイメージ SHA256 を保持して、さまざまな環境全体でイメージのセキュリティと不変性を保証します。

プロセスを実行するには

1. このツールを使用して同期を行うため、skopeo をインストールします。
2. 同期する HM リリースを構成します。

```
export EDBPGAI_RELEASE=<your-release-version>
```

1. EDB Cloudsmith アクセストークンを構成します。

```
export CS_EDB_TOKEN=<your-cloudsmith_edb_token>
```

1. イメージリストアーティファクトをダウンロードします。

```
curl -sLO "https://downloads.enterprisedb.com/${CS_EDB_TOKEN}/pgai-platform/raw/names/
↪${EDBPGAI_RELEASE}-images.txt/versions/${EDBPGAI_RELEASE}/images.txt"
```

1. EDB Cloudsmith レジストリソースを構成します。

```
export LOCAL_REGISTRY_URI=<your-local-registry-address>
```

1. skopeo を使用してソースレジストリにログインし、要求に応じて必要な資格情報を提供します。

```
skopeo login docker.enterprisedb.com
```

1. skopeo を使用して宛先レジストリにログインし、要求に応じて必要な資格情報を提供します。

```
skopeo login <your-local-registry-address>
```

1. 画像リストを解析し、各画像を同期します。

```
while read -r image; do skopeo --override-os linux copy --multi-arch all docker://
↪$EDB_SOURCE_REGISTRY/${image/: *@/@} docker://$LOCAL_REGISTRY_URI/${image/:* @/@} --
↪retry-times 3; done < images.txt
```

HM に必要な名前空間と必要なシークレットを作成する

プリインストールのこのフェーズでは、HM に必要な名前空間と、それに必要なシークレットを作成します。名前空間とシークレットの両方がインストールプロセスで使用されます。

1. 必要なコンテナレジストリ環境変数を設定します。

```
export CONTAINER_REGISTRY_URI=<local-registry-uri>
export CONTAINER_REGISTRY_USERNAME=<container-registry-username>
export CONTAINER_REGISTRY_PASSWORD=<container-registry-password>
```

1. LakeKeeper の Postgres 交絡キーと Griptape の fernet キーを生成および設定します。

```
PG_CONFOUNDING_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)
FERNET_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)
```

1. 必要な名前空間を作成します。

```
kubectl create ns edbpgai-bootstrap
kubectl create ns upm-replicator
kubectl create ns upm-lakekeeper
kubectl create namespace upm-griptape
```

1. HM、UPM-レプリケーター、LakeKeepr、および Griptape のシークレットを作成します。

```
kubectl create secret docker-registry edb-cred \
    -n edbpgai-bootstrap \
    --docker-server="${CONTAINER_REGISTRY_URI}" \
    --docker-username="${CONTAINER_REGISTRY_USERNAME}" \
    --docker-password="${CONTAINER_REGISTRY_PASSWORD}"
```

```
kubectl create secret docker-registry edb-cred \
    -n upm-replicator \
    --docker-server="${CONTAINER_REGISTRY_URI}" \
    --docker-username="${CONTAINER_REGISTRY_USERNAME}" \
    --docker-password="${CONTAINER_REGISTRY_PASSWORD}"
```

```
kubectl create secret generic pg-confounding-key \
--from-literal=PG_CONFOUNDING_KEY=${PG_CONFOUNDING_KEY} \
--namespace=upm-lakekeeper
```

```
kubectl create secret generic fernet-secret \
--from-literal=FERNET_KEY=${FERNET_KEY} \
--namespace=upm-griptape
```

1. シークレットに注釈を付けます。

```
kubectl annotate secret -n upm-replicator edb-cred replicator.v1.mittwald.de/
↪replicate-to="*" --overwrite
```

グリップテープのシークレット

グリップテープシークレットの設定については、following documentation を参照してください。

カタログシークレット

カタログシークレットの設定については、 following documentation を参照してください。

シークレットがすべて設定されたので、インストールに必要な *setting up the environmental variables* に進むことができます。

Set the necessary environmental variables for install

ドメイン変数を設定する

Hybrid ManagerHM クラスターは、インストール中に特定のユーザードメインへのアクセスが必要です。

1. HM のポータルドメイン名の変数を設定します。

```
export PORTAL_DOMAIN_NAME=<your-hm-portal-domain-name>
```

1. Transporter RW サービスドメイン名の変数を設定します。

```
export TRANSPORTER_RW_SERVICE_DOMAIN_NAME=<transporter rw service domain name> #␣
↪Customer domain name for the EDB Postgres AI Platform Transporter RW Service
```

1. EDB Postgres AI エージェントドメイン名の変数を設定します。

```
export BEACON_SERVICE_DOMAIN_NAME=<beacon service domain name> # Customer domain name␣
↪for the EDB Postgres AI Beacon Service
```

ストレージクラス変数を設定します

HM クラスターには、事前定義されたストレージクラスが必要です。 :

```
export STORAGE_CLASS=<storage class>
```

1 つのオプションは、デフォルトの GKE ストレージクラス `standard-rwo` を使用することです。

認証変数を設定する

HM のインストールにはいくつかの認証値が必要です。

1. メールアドレスの変数を設定します。

```
export AUTHENTICATION_EMAIL=<your-email>
```

1. パスワードハッシュの変数を設定します-取得したハッシュ化されたパスワード。例 `echo "mypassword" | htpasswd -BinC 10 admin | cut -d:  -f`

```
export AUTHENTICATION_HASH=<your-password-hash>
```

1. ユーザー名の変数を設定します。 Owner MyCompany はデフォルトのユーザー名です。

```
export AUTHENTICATION_USERNAME=<Owner MyCompany>
```

1. ユーザー ID を設定します。

```
export AUTHENTICATION_USER_ID="your-unique-user-identifier"
```

オプション FIPs モードを設定します Transporter Data Operator

オプションで、Transporter Data Operator の FIPS モードを設定できます。

```
export TRANSPORTER_FIPS_ENABLED=true # Enable FIPS mode for Transporter Data Operator
```

Connecting to Hybrid Manager

EDB Postgres AI Hybrid Manager プラットフォームのインストールが成功したことを確認するには、ブラウザーを使用して Hybrid Manager コンソールに接続します。

```
https://portal.foo.network/
```

この構成では、ホスト名が広域インターネットに存在しないため、https は検証されません。証明書の警告を受け入れて、ポータルに進みます。証明書の警告を受け入れて、ポータルに進みます。

ログイン画面で、ブートストラップインストールで生成されたデフォルトのユーザー名とパスワード `owner@mycompany.com` および `password` を入力できます。

ログインしたら、 *Create a database cluster template* ができます。

4.3.5 Customizations

インストールをカスタマイズすることにより、ハイブリッドマネージャーの追加機能を有効にできます。

- *Enabling GenAI Builder on Hybrid Manager* を有効にして AI エージェントをビルドします。
- *Enabling Data Catalogs on Hybrid Manager* を有効にして、分析ワークロードの単一のテーブルでデータを管理します。
- 内部コンポーネント間の安全な通信のために *Customizing Migration Portal secrets for secure internal communication* をカスタマイズします。

Enabling Data Catalogs on Hybrid Manager

ハイブリッドマネージャーのデータカタログでは、Lakehouse データストレージを管理するために追加の構成が必要です。これには、Lakekeeper サービスが暗号化されたデータを保存するために使用できる交絡キーを使用して Kubernetes シークレットを作成することが含まれます。

キーを使用してシークレットを作成する

eks-install-secrets.sh スクリプトを使用した EKS インストールの場合、キーとシークレットの作成をスキップし、交絡キーのバックアップに直接進むことができます。他の EKS セットアップとオペレーティングシステムでは、交絡キーを使用した手動シークレットの作成が必要です。

1. 交絡キーを作成し、変数に保存します。

```
PG_CONFOUNDING_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)
```

!!!注-交絡キーは、少なくとも 32 バイトの長さ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、新しい交絡キーを作成します。 .. ::

1. サービスの名前空間を作成します。

```
kubectl create namespace upm-lakekeeper
```

1. 作成した交絡キーを参照し、専用の名前空間に保存されるシークレットを作成します。

```
kubectl apply -f - <<EOF
  apiVersion: v1
  kind: Secret
  metadata:
    name: pg-confounding-key
    namespace: upm-lakekeeper
  stringData:
    PG_CONFOUNDING_KEY: ${PG_CONFOUNDING_KEY}
EOF
```

シークレットの構成が完了したら、*Hybrid Manager installation methods for Google Cloud GKE* に進みます。

交絡キーのバックアップ

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。災害シナリオでこのキーを失うと、ハイブリッドマネージャーデータカタログによって管理されるレイクハウスデータにアクセスするメカニズムが存在しない状況が発生します。これには、管理者が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。このような手順はリスクがいっぱいであり、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。

キーを取得して、安全に保存できます。

```
kubectl get secrets -n upm-lakekeeper pg-confounding-key -o yaml
```

Enabling GenAI Builder on Hybrid Manager

GenAI Builder は、 AI エージェントをビルドできるツール Griptape の EDB の統合です。

ハイブリッドマネージャーインスタンスで GenAI Builder ランチパッドアプリケーションを有効にするには、アプリケーションのシークレットを作成し、専用のオブジェクトストレージを構成する必要があります。

GenAI Builder のシークレットの作成

`eks-install-secrets.sh` スクリプトを使用した EKS インストールの場合、キーとシークレットの作成をスキップし、*GenAI ビルダーの DataLake オブジェクトストレージの構成*、*CORS の設定* 、および *Fernet キーのバックアップ* に直接進むことができます。他の EKS セットアップとオペレーティングシステムでは、Fernet キーを使用した手動シークレットの作成が必要です。

1. Fernet キーを作成し、変数に保存します。

```
FERNET_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)
```

!!!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. 統合の名前空間を作成します。

```
kubectl create namespace upm-griptape
```

1. 作成した fernet キーを参照し、専用の名前空間に保存されるシークレットを作成します。

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
   name: fernet-secret
   namespace: upm-griptape
stringData:
   FERNET_KEY: ${FERNET_KEY}
EOF
```

GenAI ビルダーの DataLake オブジェクトストレージの構成

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-griptape-datalake-bucket-name> -region     <your-region>
```

```
gsutil mb -l <your-region> gs://<your-griptape-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、Griptape DataLake バケットの一意の名前を持つ新しいバケットを作成します。

1. バケットに関する次の情報に注意してください。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。UI は、DataLake バケット構成の入力を求めます。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

CORS の設定

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 を利用できます。

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 create a service account を更新します。
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。

```
cat cors-config.json
[
  {
```

(次のページに続く)

(前のページからの続き)

```
    "origin": ["https://${PORTAL_DOMAIN_NAME}"],
    "method": ["GET", "PUT", "POST", "DELETE", "HEAD"],
    "responseHeader": ["*"],
    "maxAgeSeconds": 3600
  }
]
```

`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。

1. CORS 構成を、前に作成したバケットに適用します。

```
gsutil cors set cors-config.json gs://<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、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。

シークレットを構成し、専用バケットを作成したら、 *Hybrid Manager installation methods for Google Cloud GKE* に進みます。

Fernet キーのバックアップ

Hybrid Manager 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。

キーを取得して、安全に保存できます。

```
kubectl get secret -n upm-griptape fernet-secret -o yaml
```

Customizing Migration Portal secrets for secure internal communication

移行ポータルは、内部コンポーネント通信にいくつかのシークレットを使用します。これらのシークレットは、すばやく機能するインストールのためにハイブリッドマネージャーリリースでハードコーディングされていますが、運用環境のカスタムシークレットでオーバーライドすることを強くお勧めします。

これにより、シークレットが簡単に発見されず、侵害のリスクが削減されます。

Migration Portal のシークレットを理解する

各シークレットの **説明** は、それが必要な理由を説明し、**helm チャート** は、構成ファイルのどこを参照する必要があるかを説明し、**parameters** セクションはデフォルト以外の設定方法に関するガイダンスを提供します値。

このセクションでは、シークレットとその機能の概要を提供するのに対し、**デフォルトのシークレットのオーバーライド** は、カスタムシークレットを構成し、デフォルトのシークレットをオーバーライドする方法に関する例を提供します。

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。**Helm 値** parameters.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 helm 値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。このユーザーは、parameters.edb-migration-portal.db_owner シークレットをインストールに適用するに入力する必要があります。
- 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 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。**Helm 値** parameters.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 helm 値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。

移行ポータルからの移行コパイロット 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。**Helm 値** parameters.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 helm 値で参照します。
- stringData.username コパイロットで認証するときに移行ポータルが使用する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。初期化時に、edb-migration-copilot コンポーネントは、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確認します。**Helm 値** parameters.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へのアクセス時にコパイロットが使用する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。**Helm 値** `parameters.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別の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。

デフォルトのシークレットのオーバーライド

最初に、内部コンポーネント通信用のカスタムシークレットを作成します。次に、使用しているインストール方法に応じて、`values.yaml` 構成ファイルまたは `HybridControlPlane` カスタムリソースのカスタムシークレットを参照して、値をオーバーライドします。

この例では、カスタムシークレットを作成し、ターミナルで直接作業してそれらを使用するようにハイブリッドマネージャーを構成する方法を示しています。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)
```

(次のページに続く)

(前のページからの続き)

```
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. *Migration Portal* **データベースアカウント** 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. *Migration Portal* **データベースのスーパーユーザーアカウント** 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 \
  --from-literal=username=${HTTP_USERNAME} \
  --from-literal=password=${HTTP_PASSWORD} \
  --type=kubernetes.io/basic-auth
```

1. **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** 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. **モニタリングと移行コパイロット** 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

Hybrid Manager のインストールに付属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、values.yaml または HybridControlPlane カスタムリソース使用したインストール方法に応じて、新しいシークレットを参照します。

```
parameters:
  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
```

インストールプロセスを続行します。 helm upgrade コマンドを呼び出して values.yaml 構成ファイルを適用するか、HybridControlPlane を適用することによりブートストラッププロセスを実行すると、インストールはシークレットオーバーライドを使用する場合に行われます。

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、必ず再度ブートストラップして変更を有効にしてください。

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。

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 タスク

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。

- *Assessing schemas for external projects* —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。
- API Access — HM API を使用してプラットフォーム機能を自動化します。
- *Upgrading on Hybrid Manager* —Postgres および PGD のメジャーおよびマイナーアップグレードの手順に従います。
- *Customizing Migration Portal secrets for secure internal communication* —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 ナレッジベース資産を管理します。
- *Enabling the Migration Portal AI Copilot* —Copilot を使用してガイド付き移行ワークフローのロックを解除します。

5.1.1 Hybrid Manager Terminology

これらの用語を確認して、PG AI Hybrid Manager 機能を理解および活用します。

PG AI ハイブリッドマネージャー

PG AI 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、モニタリング、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。

プライベート Database-as-a-Service DBaaS

PG AI Hybrid Manager 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。

エンジニアードシステム

Engineered System は、ボックス内の PG AI Hybrid Manager 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。

PG AI Console

PG AI Console は、PG AI Hybrid Manager 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。

プロジェクト

PG AI Hybrid Manager 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、 *Assessing schemas for external projects* を参照してください。

エステート

PG AI Hybrid Manager では、 **Estate** 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。PG AI Hybrid Manager 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、エステートの一部になります。

詳細については、 *Postgres* **エステート統計** を参照してください。

Launchpad

Launchpad は、PG AI Hybrid Manager 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。

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。

Kubernetes クラスター

PG AI Hybrid Manager のスコープ内で、Kubernetes クラスターは、 PG AI Hybrid Manager 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。PG AI 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。

Kubernetes ストレッチクラスター

これは、PG AI Hybrid Manager 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、アドバンスト高可用性 AHA、または分散高可用性 DHA が有効になっている PG AI Hybrid Manager 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。

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。

データベースクラスター

PG AI Hybrid Manager 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、PG AI Hybrid Manager 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 PG AI Hybrid Manager に接続します。 PG AI Hybrid Manager 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、*PG AI Hybrid Manager* **のデータベースクラスタータイプ** 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。

PG AI Hybrid Manager のデータベースクラスタータイプ

PG AI 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 PG AI Hybrid Manager 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、PG AI Hybrid Manager に接続してモニタリングと移行機能を提供します。

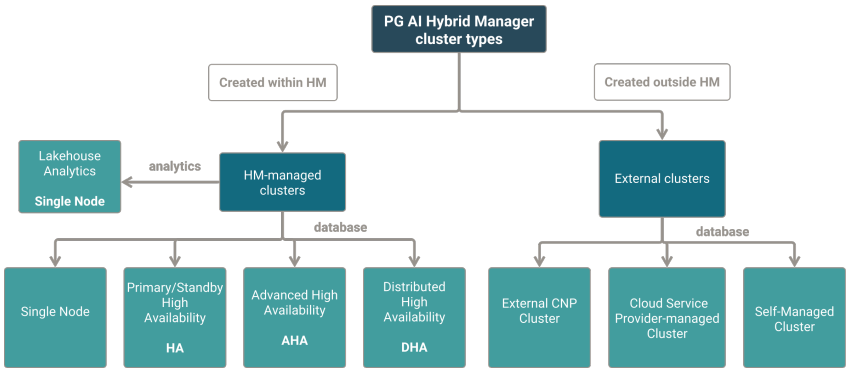


図 1: Cluster Types in PG AI Hybrid Manager

現在、次のことができます。

- *HM* **管理対象クラスター** の作成

または、

- **外部クラスターの要件** を接続します。

HM 管理対象クラスター

HM 管理対象クラスターは、 PG AI Hybrid Manager 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 PG AI Hybrid Manager 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。

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。詳細については、*Distributed high availability cluster* を参照してください。

レイクハウス分析-シングルノード PG AI ハイブリッドマネージャーを使用して *Lakehouse Analytics* クラスターを作成することもできます。PG AI Lakehouse Analytics クラスターは、オブジェクトストレージ内のオープンレイクハウスフォーマット Delta Lake などで保存されているデータを照会および分析するための環境を提供します。PG AI Hybrid Manager の統合コントロールプレーンを活用して、ユーザーは高速分析クエリ用に最適化されたこれらの特殊なクラスターを簡単に展開および管理できます。

PG AI Hybrid Manager は、これらの分析エンジンの展開と操作を簡素化し、データ駆動の意思決定を加速します。

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。

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 PG AI Hybrid Manager の外部で作成されます。これらは、PG AI Hybrid Manager 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、PG AI Hybrid Manager 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。

PG AI 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 PG AI Hybrid Manager に接続し、Data Migration Service 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

PG AI Hybrid Manager 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。

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 PG AI Hybrid Manager 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 the PG AI Hybrid Manager for observability 、 Data Migration Service を使用して移行を実行できます。

例 AWS RDS の Postgres。

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 PG AI Hybrid Manager 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。

その他の用語

データノード

「データノード」という用語は、PG AI Hybrid Manager 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。

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。

5.1.2 Hybrid Manager Console

ハイブリッドマネージャー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-コンソール-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。

Hybrid Manager コンソールでできることは何ですか?

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
- Lakehouse クラスターの作成、表示、および管理
- 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

Hybrid Manager Console の組織

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*** 、プロジェクトを作成および管理し、クラスターを作成および管理し、さまざまなタスクを実行してクラスターと各プロジェクトを管理します。
- ***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。
- *The Asset Library* 、コンテナイメージの表示、リポジトリの管理、およびナレッジベースの表示ができます。

サードパーティサービスのローンチ

HM *Launchpad* を使用して、HM ポータルを離れることなく、関連するサードパーティサービスを起動できます。

Exploring 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、ハイブリッドマネージャー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、アカウントの概要ページと、 EDB Postgres AI Hybrid Manager サービスの **単一ウィンドウ**の概要が開きます。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、*Assessing schemas for external projects* および **エステート内のリソースのモニタリング** 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、 *Assessing schemas for external projects* 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

ハイブリッドマネージャーコンソールでは、 *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*
- ***Lakehouses***
- *Configure an Iceberg REST catalog connection*
- **モデルクラスター**

エステートの機能の探索

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。
- テーブルでデータベースクラスターを選択して、ハイブリッドマネージャーで作成した Postgres クラスターにアクセスして、詳細ビューを開きます。

- ハイブリッドマネージャーに接続して移行の準備をした *assessment of the databases* を表示します。
- 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。
- *Create a destination clusters for a migration and perform a database migration* 。

エステートの **レイクハウス** タブでは、次のことができます。

- *analytics clusters* 、分析ワークロードに特化したハイブリッドマネージャークラスターをモニタします。

エステートの **カタログ** タブでは、次のことができます。

- ハイブリッドマネージャーに接続されたまたは作成されたすべての *Configure an Iceberg REST catalog connection* テーブルを表示します。

エステートの **モデルクラスター** タブでは、次のことができます。

- AI モデルを提供するクラスターを表示およびアクセスします。

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の Create an EDB Postgres AI cluster 。
- **新しいレイクハウスクラスターを追加** —*Add a new Lakehouse cluster* を既存のプロジェクトに追加します。
- **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。

- *Using Grafana dashboards*
- プロメテウス
- サノス - クエリ
- Thanos - ストアゲートウェイ
- Thanos - Compactor

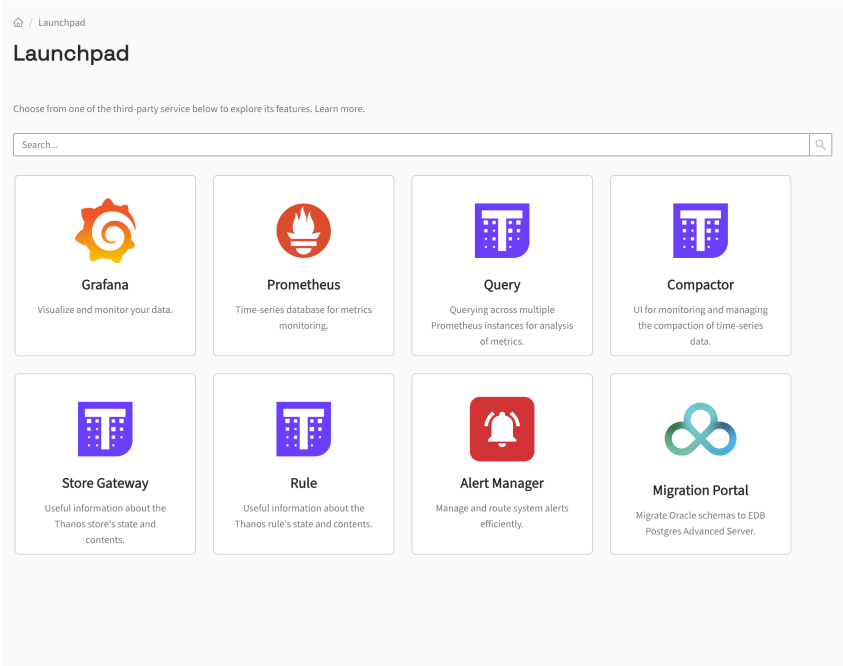


図 2: Launchpad

- 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。
- **プロジェクト** — Projects overview を開きます。
- **タグ** — **Tags** ページが開き、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。
- **ユーザー管理** —*User Management* ページが開き、ユーザーを追加し、ロールを割り当てます。
- **使用レポート**—ディスク IO、vCPU、メモリ、ディスクと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

- **私のアカウント**—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、プロジェクトの通知の管理ができます。
- **ログアウト**

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。
- **トップ 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 つのクラスターまで。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。

- プロジェクト名 —クラスターが属するプロジェクト。
- クラスター —クラスター名。
- 管理 —クラスターの管理方法アプライアンスまたは自己管理。
- クラスタータイプ —クラスターのタイプ PGD、シングルノード、または HA。
- ステータス —クラスターの現在のステータス OK、Error、 注意、または Working 。
- エンジン—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。
- バージョン—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。
- アラート —クラスターの高、中、および低の重大度のアラートの数。

- 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。
- メモリ—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。
- ディスク—クラスターによって現在使用され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。
- タグ —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。
- **:raw-latex:`\コア数`**—クラスターが使用するコアの合計数。
- Conn. 使用法 —パーセンテージでの接続使用量。
- IO 待機% —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。
- 最新のバックアップ —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。
- プロバイダー — CSP プロバイダーの ID。
- リージョン —アプライアンスクラスターのロケーション ID。
- ラグ議員 —ミリ秒単位のレプリケーション遅延。
- TDS —ギビバイト単位のデータベースの合計サイズ。

> **注釈**
>
> 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

2 つの手法を使用して、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できます。
- **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できます。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
- ヘルススコア統計
- 削除されたクラスターを含む b_tran_14 ロケーション
- Postgres バージョン
- プロジェクト名
- リソースタイプ、以下を含むアプライアンス&mash_3; ハイブリッドマネージャーによって管理されるクラスター
  - 自己管理 Postgres —自己管理されるクラスター
  - 自己管理の Postgres データベース—自己管理のデータベースを含むクラスター
- 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。

Lakehouse clusters

Estate ページから **Lakehouse** を選択して、Lakehouse クラスターを表示します。Lakehouse クラスターを表示するときに、フィルターを適用して結果を絞り込むことができます。アクセスできる Lakehouse クラスターの数、クラスターのメタデータとメトリックを確認できます。

Lakehouse クラスターの追加

Lakehouse クラスターをエステートに追加するには、 **Quick Actions** メニューから **Add New Lakehouse Cluster** にカーソルを合わせ、新しいクラスターに関連付けるプロジェクトを選択します。**Create Analytics Cluster** ページが開きます。ここでは、クラスターを追加する 2 つの方法を提供します。テンプレートからビルドするか、カスタムビルドを作成します。

- テンプレートからビルドするには、使用可能なテンプレートがない場合、**Create New Template** を選択して作成します。既存のテンプレートを使用するには、 **Use Template** を選択します。
- カスタムビルドを作成するには、 **Start from Scratch** を選択します。

分析クラスターのテンプレート作成ワークフローとカスタムビルドワークフローは、通常のクラスター作成ワークフローと同じです。Creating clusters を参照してください。

ハイブリッドマネージャーのアナリティクス/レイクハウス機能の使用方法の詳細については、Analytics/Lakehouse を参照してください。

Lakehouse 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すべてのレイクハウス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。

- プロジェクト名 —クラスターが属するプロジェクト。
- クラスター —クラスター名。
- 管理 —クラスターの管理方法アプライアンスまたは自己管理。
- クラスタータイプ —クラスターのタイプ PGD、シングルノード、または HA。
- ステータス —クラスターの現在のステータス OK、Error、 注意、または Working 。
- エンジン—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。
- バージョン—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。
- アラート —クラスターの高、中、および低の重大度のアラートの数。
- ヘルススコア
- 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。
- メモリ—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。
- ディスク—クラスターによって現在使用され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。
- タグ —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、上部に検索フィールドを備えたメニューを開きます。検索フィールドを使用して列をフィルタリングし、テキスト一致を使用して列名前を表示します。列名のリストから、テーブルに表示するものを選択します。

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。
- **:raw-latex:`\コア数`**—クラスターが使用するコアの合計数。
- Conn. 使用法 —パーセンテージでの接続使用量。
- IO 待機% —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。
- 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。
- プロバイダー — CSP プロバイダーの ID。
- リージョン —アプライアンスクラスターのロケーション ID。
- TDS —ギビバイト単位のデータベースの合計サイズ。

> **注釈**
>
> 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

2 つの手法を使用して、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できます。
- **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できます。各プロパティには、その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サブメニューがあります。
  - Engine
  - 削除されたクラスターを含める s
  - Location
  - Management
  - 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

テーブルのデータは 30 秒ごとに更新されます。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。

5.1.3 For Users

Projects

ハイブリッドマネージャーでは、プロジェクトを使用して、組織関連の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。

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。

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などのメトリックが含まれています。

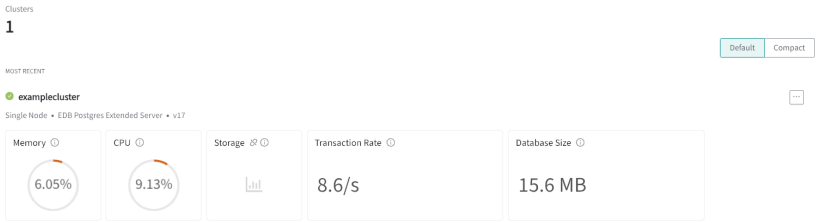


図 3: Project Overview

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 。

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。

- **Edit Cluster** から *edit the cluster* を選択します。
- **資格情報を選択して、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、*Credentials* を参照してください。

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。

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。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. プロジェクト内にクラスターがある場合、プロジェクトを削除する前にそれら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` を入力して削除を確認する必要があります。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. プロジェクトにクラスターがない場合、ダイアログボックスが表示され、チェックボックス付きの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. **Delete Project** を選択して、プロジェクトを完全に削除します。

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 — **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インスタンス内で一意である必要があります。
- *Notifications* — **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 — **Security** を選択して、透過的データ暗号化 TDE の外部管理キーをこの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 Key Management Systems for TDE* を参照してください。
- *Alerts* — **Alerts** を選択して、プロジェクトのアラート設定を表示し、いつ送信される方法を制御します。

Notifications

EDB Postgres AI 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。

- **ユーザー通知チャンネル**—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、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。
- **システム通知チャンネル**—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 回トリガーされるように設計されています。

ハイブリッドマネージャー HM でのユーザーのロールは、通知設定へのアクセスの範囲を決定します。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のみ
- グローバル —ユーザーとシステムの両方の通知チャネル
- プロジェクト —ユーザーとシステムの両方の通知チャネル

[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

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。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。

これらの手順は、組織レベルでシステム通知チャネルを構成する方法を示しています。

グローバルレベルで通知を設定する

グローバルレベルの通知は、組織の所有者のみが設定できます。

Webhook:

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。
3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Webhook を選択し、ポップアップウィンドウに詳細を入力します。

i. Webhook のホスト URL

ii. Webhook の方法-POST または PUT

iii. 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
    1. ユーザー名
    2. パスワード
- ベアラートークンの場合

1. ベアラートークン

iv. [保存] ボタンを選択します。

ポケベルデューティ

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。
3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Pager Duty を選択し、ポップアップウィンドウに詳細を入力します。
    1. ルーティングキー統合キーを設定します。
    2. Save ボタンを選択します。 !!!note Pager Duty 通知チャンネルを構成する前に
    - 昇格した特権で Pager Duty アカウントにアクセスする必要があります。
    - Pager Duty サービスは Events API v2 統合で作成および構成されました。
    - Pager Duty アカウントで、に移動します目的のサービスを選択し、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、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 が提供する統合キールーティングキーをコピーします。

メールアドレス

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。
3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Email を選択し、ポップアップウィンドウに詳細を入力します。
    1. Email To - 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
    2. Email From - 電子メールを送信している電子メールの場合
    3. Host - 電子メールサービスの URL
    4. ポート- 電子メールサービスのポート
    5. ユーザー名 - 電子メールサービスのユーザー名
    6. パスワード - 電子メールサービスの
    7. [Save] ボタンを選択します。

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。

EDB Postgres AI コンソールにログインします

1. 上部の [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 [設定] 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のグローバル通知設定に記載されている手順に従います。

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、 EDB Postgres AI で割り当てられた複数のロールで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。

EDB Postgres AI コンソールの上部にあるベルアイコンを選択して、アプリ内通知を表示します。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

通知設定を管理するには

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[My Account] を選択します。
3. [通知] タブを選択します。通知は、アカウント、グローバル、および使用可能なプロジェクトごとにグループ化されています。
4. 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。
5. 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。
6. 各通知に対して、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、特定のイベントの通知の受信をオプトアウトまたはオプトインできます。- 電子メールとアプリ内用のチェックボックスを使用して、イベントの電子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、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ハイブリッドマネージャー HM オブザーバビリティ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] タブの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。

1. HM クラスターの場合

    - コンソールに移動し、上部で Projects
    - リストされたプロジェクトを選択します b_tran_5 左側のナビゲーションで、 Clusters
    - に移動します 準備ができているクラスターを選択します b_tran_5 クラスターの詳細ページで、[モニタリング] タブを選択します

2. 外部 HM クラスターの場合

    ```
    -  コンソールに移動し、上部で Estate を選択します
    ```

    - すべてのクラスターをリストするテーブルに移動します s
    - 管理が自己管理としての準備ができているクラスターを選択します
    - クラスターの詳細ページで、[モニタリング] タブを選択します

[Monitoring] タブには、 Active Alerts リストが表で表示されます。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
- Sort ユーティリティ、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ことができます
- Refresh ボタン、アラートリストを更新します
- Column Selector ドロップダウン、設定に応じてテーブルに表示する列を選択できます

- Download ボタン、アラートリストを CSV 形式でダウンロードします

Refresh ボタンを使用して、アラートを手動で更新できます。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、[モニタリング] タブのフィルタが自動的に適用され、関連するグラフが表示されます。

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

アラートはプロジェクトレベルで設定できます。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. EDB Postgres AI Console にサインインします。
2. コンソールの上部で、[プロジェクト] を選択します
3. リストからプロジェクトを選択します
4. 左側のナビゲーションで、[設定] に移動します
5. 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するには
- 後に通知—アラートがトリガーされるまでの時間を設定するには
- 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

データベースレベルで設定できるアラート

- 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

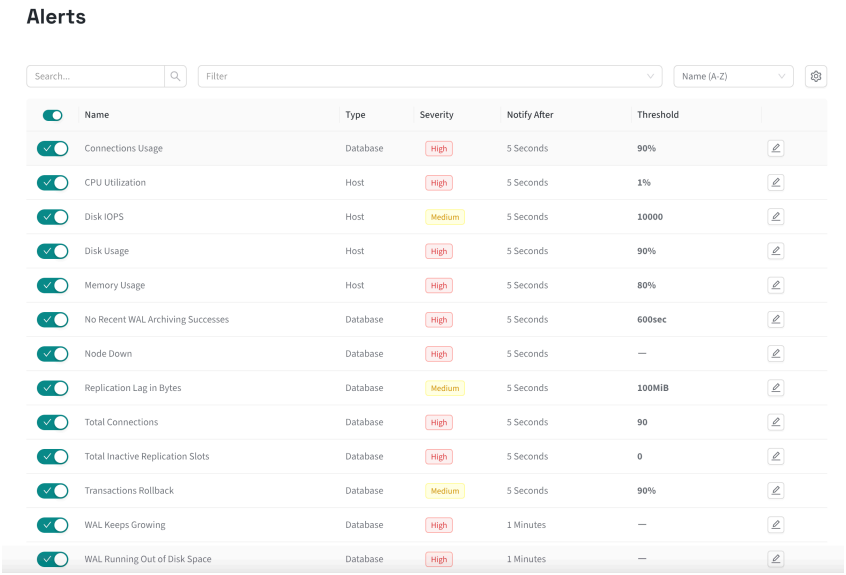


図 4: Alerts

- 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

ホストレベルで設定できるアラート

- CPU 使用率
- ディスク IOPS
- ディスクの使用量
- メモリ使用量

!!!note-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。 - 外部 HM クラスターを利用できるようにするための技術的な制限のため、すべてのアラートは外部 HM クラスターでは利用できません。

Managed Storage locations

管理対象ストレージ場所は、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 Manager UI 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場所を作成できますが、現在、管理対象ストレージロケーション機能を使用している Hybrid Manager サービスはありません。

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.	•
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

Working with clusters

HM を使用して、データベースクラスターを作成できます。データベースは、次の Postgres タイプを使用できます。EDB EDB Postgres Advanced Server、 EDB Postgres Extended Server、または PostgreSQL。

クラスターの作成については、 *Creating a database cluster* を参照してください。

クラスターの作成の一部として、またはクラスターの作成後に、いくつかの関連タスクを実行する必要がある場合があります。たとえば、クラスターの編集、復元、および削除を管理したり、クラスターのパフォーマンスを理解したりする必要がある場合があります。また、今後作成するクラスターのベースとして使用するテンプレートを作成することもできます。

これらのクラスター管理タスクの詳細については、*Managing clusters* を参照してください。

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、新しい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** ステップ *2* 自分の道を選択してください 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、 **Create Cluster** ステップ *2* 自分の道を選択してください 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
- ***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、クラスターのプロビジョニングが開始され、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、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クラスターの場合。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。**高度な高可用性** を選択すると、タブ **DB 構成** と **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、タブ **DB Configuration** と**追加設定** は表示されなくなります。代わりに [**Data Groups**] タブが表示されます。

詳細については、 Supported cluster types 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このタブに表示されます。他のクラスタータイプの場合は、 *Creating 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 —クラスターを展開する場所を選択します。デフォルトの場所は、EDB Postgres AI サービスをホストする Kubernetes クラスター内です。マルチプル HM 環境で HM が構成されている場合、クラスターを展開する場所を選択できます。

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。*Migrations* を表示します。 EDB Postgres Advanced Server は、3 つのクラスタータイプすべて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、*Asset Library in Hybrid Manager* 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。

先行書き込みログには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先行ログ 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書き込み先行ログ 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復元力が得られます。選択を解除すると、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は、データベースストレージクラス gp2 または gp3 と GiB 単位のデータベースストレージボリュームのサイズデフォルト 1 です。

書き込み先行ログストレージ – 書き込み先行ログストレージの下で、WAL ストレージクラス gp2 または gp3 と GiB 単位の WAL ストレージボリュームのサイズデフォルトは 1 を選択できます。

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。

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と 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。 PSR および PGD クラスター展開のデフォルトの `wal_segment_size` は、それぞれ 64MB と 128MB に設定されています。

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** リストから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。

> **危険**
>
> 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。

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

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は、パスフレーズを使用したこの secures the data encryption key created by the Hybrid Manager 。
- Hybrid Manager インスタンスの管理者が以前に *enabled a key from a key management system* 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 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 Group**] 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。設定できるのは、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のみです。

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* リファレンスを参照してください。

展開の場所 —クラスターを展開する場所を選択します。デフォルトの場所は、EDB Postgres AI サービスをホストする Kubernetes クラスター内です。マルチ HM 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、クラスターを展開する場所を選択できます。

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、3 つのクラスタータイプすべて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、*Asset Library in Hybrid Manager* 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。

先行書き込みログには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先行ログ 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**の下では、データベースストレージクラス gp2 または gp3 と GiB 単位のデータベースストレージボリュームのサイズデフォルト 1 です。

書き込み先行ログストレージ – **書き込み先行ログストレージ**の下で、WAL ストレージクラス gp2 または gp3 と GiB 単位の WAL ストレージボリュームのサイズデフォルトは 1 を選択できます。

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。

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** リストから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。

危険

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。
- **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、事前定義された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 tempaltes* 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 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、選択可能な使用可能なテンプレートのリストを表示します。

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。

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。

- *Edit the cluster* 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。

[クラスターテンプレート] ページには、Lakehouse Analytics テンプレートを含む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** アイコンボタンを選択します。

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 の [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。

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。詳細はこちら *Cluster creation settings*
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

ハイブリッドマネージャー HM 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。

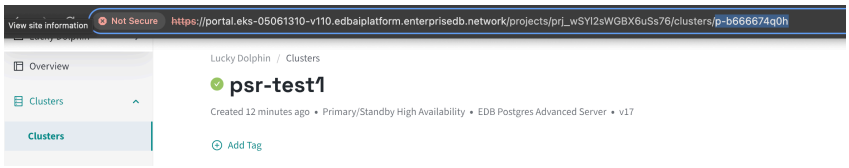


図 5: Browse cluster

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で、 **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。

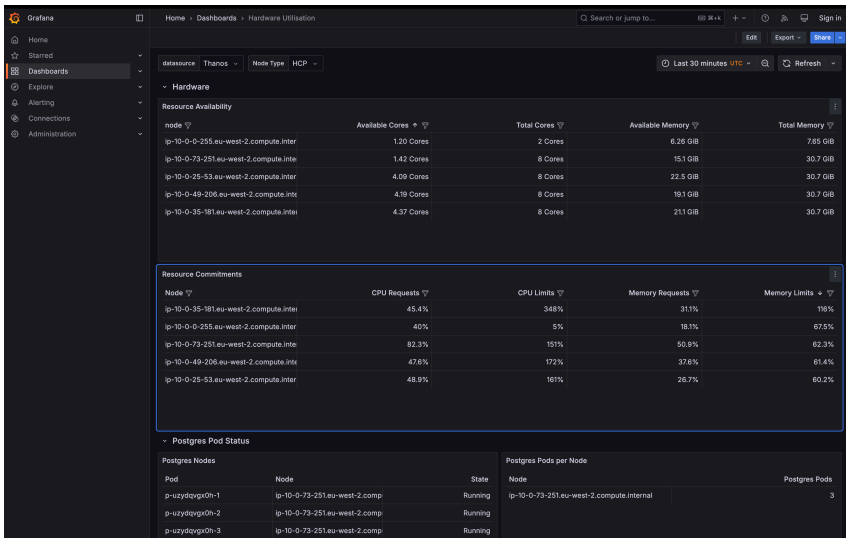


図 6: Hardware Utilization dashboard

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。
- **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

要求、CPU とメモリの両方は、その上で実行されているポッドによるノードの要求の満たされる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 とメモリの両方の制限は、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。UI 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 **ハードウェア使用率**ダッシュボードを *Grafana* **ハードウェア使用率ダッシュボード** 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 と、それを使用して、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確認し、ノードに残っているコア数、メモリ量などを確認できます。で。

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、名前空間ごとにクラスターが表示されます。クラスターが関連付けられているプロジェクト ID、その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、およびクラスターの状態と年齢を含みます。

```
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 Provisioning 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。繰り返しになりますが、最初に kubectl get clusterwrappers を使用して、HM の PG クラスターの名前とその名前空間を照合します。

```
kubectl get clusterwrappers
__OUTPUT__
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 Pending であることはすでにわかります。HM は、まだプロビジョニング中であることを示していたため、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、ポッドがまだ Pending である場合、リソースが不足している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、出力の最後に、何が問題になっているのかをより詳細に示したものが表示されるはずです。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され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を構成する場合は、*Configuring LDAP* 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。
- LDAP を介した認証の前に、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ます。これにより、LDAP 認証が構成されます。

次の手順を使用して、LDAP を使用するようにクラスターを作成または変更します。

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,
```

(次のページに続く)

(前のページからの続き)

```
    "resourceRequest": {
      "requests": {
        "memory": "2Gi",
        "cpu": "1"
      }
    },
    "storageConfiguration": {
      "primaryStorage": {
        "size": "4"
      }
    },
    "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=",
```

(次のページに続く)

(前のページからの続き)

```
    "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"
```

バインドおよび検索モードの構成

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

例

```
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

例

```
postgresql:
  ldap:
    server: "openldap.default.svc.cluster.local"
    bindAuth:
      prefix: "uid="
      suffix: ",dc=example,dc=org"
```

LDAP over TLS

TLS は、PostgreSQL サーバーと LDAP サーバー間のトラフィックを暗号化するために使用できます。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。

リファレンス

https://www.postgresql.org/docs/current/auth-ldap.html

https://cloudnative-pg.io/documentation/preview/postgresql_conf/#ldap-configuration

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。

- **重要な用語とアーキテクチャの概要** — クラスターの簡単な概要。
- **バケットを *GenAI Builder* に接続する** — クラスターの接続情報。
- **モニタリングと移行コパイロット** — クラスターのモニタリング情報。
- **診断 *[Query Diagnostics]* および *[Recommendations]* タブ** — クラスターの診断情報を照会します。
- ***PGAIHM* バックアップと *DR*** — クラスターのバックアップ情報。
- *Enabling Data Catalogs on Hybrid Manager* — クラスターのログ。
- *Cluster Properties* — クラスターのプロパティ。

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 *Quick Connect* 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 *Creating data groups* 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。

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 Manager 1.1 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、およびそれらの間の相互接続を示しています。

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、 ***Monitoring*** 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、 ***Logs*** タブを選択してください。

Cluster connections

Cluster Connect タブを使用して、クラスターの接続情報と、クラスター内の `edb_admin` データベースに接続する 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。

.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。

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。

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。

- **メモリ** — 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。
- **ストレージ** — Postgres プライマリノードに使用される合計ストレージ。
- **Disk IOPS** —期間中の 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 ノードの期間にわたって 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

- **接続**—タイプ別のクライアントアプリケーションと Postgres データベース間の現在の接続数。
- **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 ノードの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

- **タプルイン** —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。
- **タプルアウト** —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

- **クエリレート** — Postgres ノードの 1 秒ごとのクエリーの合計数。
- **クエリレイテンシー**—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 ノードの合計データベースサイズ。
- **ディスク使用量** —プライマリ Postgres ノードのディスク使用率。
- **WAL サイズ**—プライマリ Postgres ノード全体の WAL ディレクトリサイズの合計。
- **WAL の使用状況** —プライマリ Postgres ノードの WAL 使用率。
- **ライブ/デッドタプル** — 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以降の時間。
- **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 ノードの最後のチェックポイントからの秒単位の時間。
- **チェックポイント統計** —Postgres ノードの 1 秒ごとのチェックポイントの合計数。
- **最後に成功した 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 アーカイブ統計** —Postgres ノードの 1 秒ごとの 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 `0` 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、クラスターのスコアカードと推奨事項のリストが表示され、Postgres クラスターのパフォーマンスの向上に役立ちます。

スコアカードと推奨事項は直接関係しています。1 つ以上の推奨事項が使用できる場合、クラスターは Excellent よりも低いスコアのみを示します。これらの推奨事項が適用または却下されると、スコアは Excellent に戻ります。

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。

注現時点では、最初の 2 つインデックスと統計のみが利用できます。

通常、グレードは A〜F スケールで割り当てられます。

インデックスの場合

- 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。

統計の場合

- 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、
- Good B、少なくとも 1 つの推奨事項が生成された場合。

次のことに注意してください。

- “Fail”F は、即時のアクションが必要な異常なクラスター状態のために予約されているため、当面は使用されません。
- 「Unknown」U は、クラスターを評価するための情報が利用できない場合に使用されます。

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、パフォーマンスが最も悪い PG クラスター内のデータベースに関連しています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。

特定のカテゴリを選択すると、スコアが計算された方法に関する詳細なコンテキストを提供するだけでなく、これらの推奨事項の関連性をユーザーが理解するのに役立つテキストが表示されます。

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。

- 「Status」、推奨事項のステータス New、Ignored、Applied、
- “データベース名”,
- “ノード ID”、
- “Size Increase”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で得ら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、実行時間とこのクエリを実行するためのプラン時間を加えたもの、
- 「コスト削減」インデックスの場合、(abs_benefit/total_cost)*100 として計算される推定コスト削減割合。これは、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、そのような統計を適用することの潜在的な利点の大まかな推定、
- 「Recommend Time」このレコメンデーションの作成時間、
- 「Query」 インデックスまたは統計を作成するクエリのテキスト。

歯車アイコンを使用すると、列の表示/非表示が可能です。

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。

!!!注推奨事項は Web インターフェイスから適用できません。ユーザーはそれらを手動でコピーして、ターミナルまたは管理 GUI で実行する必要があります。

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、
- ノード、
- ステータス。

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。

推奨事項を理解する

[推奨事項を適用/無視としてマークする] ボタンは何をしますか?

’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に伝える必要があります。おすすめを適用済みとしてマークすると、リストから消え、次にページがリロードされると、スコアがそれに応じて増加します。同じ推奨事項を再度受信すると、変更が実際に適用されないか、元に戻されたことを示唆します、推奨事項が再度表示され、スコアが低下します。

’ignored’オプションを使用して、推奨される変更を確認し、適用することが適切でないと判断したことをハイブリッドマネージャーに伝える必要があります。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、次にページがリロードされたときに、スコアがそれに応じて増加します。同じ推奨事項を再度受け取った場合、それは無視され、再現されません。

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るため、この拡張機能は監視対象データベースで有効にする必要があります。

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推奨事項が生成されます。

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。

Upgrading on Hybrid Manager

アップグレードは、データベースのメジャーアップグレードでもマイナーアップグレードでも、データベース管理の重要な部分です。

- *Postgres major version upgrades*
- *Postgres minor version upgrades on HM*

Postgres major version upgrades

- *Postgres major version upgrades on PGD with HM*

Postgres major version upgrades on Hybrid Manager

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、ユーザーは自分の裁量でメジャーバージョンのアップグレードを管理できます。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**
>
> PG 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。

メジャーアップグレードは大きな利点を提供しますが、注意深い準備が必要です。

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。

前提条件

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。次に、クラスターとデータベースメトリックを確認し、すべてが動作していること、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 PG メジャーバージョンをアップグレードする

1. 最初に、クラスターのメインページから **Quick Action>Upgrade Cluster** を選択し、 **Major Upgrade** を選択します。
2. **アップグレードクラスター— Major** ダイアログウィンドウが開きます。
3. 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。
4. アップグレードパスを確認し、[**Continue**] ボタンを選択して確認します。
5. ダイアログウィンドウには、アップグレードで予想される特定のパッケージの変更が表示されます。これらを確認し、すべてが正しい場合は**アップグレードクラスター** を選択します。
6. アップグレードは自動的に適用されます。 次に、 **Health Status** 、**Monitoring** 、および **Logs** タブを使用して、プロセスをモニタしたり、予期しない動作のトラブルシューティングを行ったりできます。

> **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。 問題の製品の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使用可能な自動ロールバックはなく、ダウングレードの実行はお勧めできません。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し、アプリケーションワークロードを新しいクラスターにスイッチ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ます。次に、PG 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。

Postgres minor version upgrades on HM

マイナーバージョンのアップグレードは、クラスタータイプ単一ノード、プライマリ/スタンバイ、分散および Postgres データベースタイプ PostgreSQL コミュニティ、EDB Postgres Extended Server、 EDB Postgres Advanced Server に関係なく、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。

PG 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。PG 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。

ユーザーは、ハイブリッドコントロールプレーン HM を使用して、新しいバージョンを利用でき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**
>
> PG 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的ですが、推奨される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リカバリプラン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。次に、クラスターとデータベースメトリックを確認し、すべてが動作していること、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 PG マイナーバージョンをアップグレードする

1. 最初に、クラスターのメインページから **Quick Action>Upgrade Cluster** を選択し、 **Minor Upgrade** を選択します。
2. **アップグレードクラスター—マイナー**ダイアログウィンドウが開きます。
3. 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。
4. アップグレードパスを確認し、[**Continue**] ボタンを選択して確認します。
5. ダイアログウィンドウには、アップグレードで予想される特定のパッケージの変更が表示されます。これらを確認し、すべてが正しい場合は**アップグレードクラスター** を選択します。

6. アップグレードは自動的に適用されます。 次に、 **Health Status** 、**Monitoring** 、および **Logs** タブを使用して、プロセスをモニタしたり、予期しない動作のトラブルシューティングを行ったりできます。

> **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。 問題の製品の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使用可能な自動ロールバックはなく、ダウングレードの実行はお勧めできません。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し、アプリケーションワークロードを新しいクラスターにスイッチ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ます。次に、PG 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。

Migration with Hybrid Manager

データベースクラスターを移行する方法。

- Oracle から移行する場合、ハイブリッドマネージャーに組み込まれている移行ポータルインスタンスを使用して *manage and convert schemas for HM migrations* 。 Postgres から移行する場合、外部ツールである Migration Toolkit を使用して *schema migration* を管理します。
- Data Migration Service を使用して、自己管理データベースのスナップショットまたはストリーミングレプリケーションを実行し、HM 管理データベースに移行します。

Oracle、自己管理 Postgres、AWS RDS などからのエンドツーエンドの移行の例については、Migration Guides を参照してください。

Migration Portal

移行ポータルは、Oracle データベーススキーマの評価、分析、変換、および EDB Postgres Advanced Server への移行に役立つサービスです。Web ベースのサービスとして、またはハイブリッドマネージャーのアプリケーションとして利用できます。このセクションでは、ハイブリッドマネージャーの一部としての使用に焦点を当てます。

機能

移行ポータルのハイブリッドマネージャー展開内の一部の機能は一意であるか、Web ベースのバージョンと比較して機能が異なります。

AI コパイロット

!!!note

ハイブリッドマネージャー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が、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、AI 駆動のチャットボットツールであり、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を支援するように設計されています。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。

移行ポータルとは異なり、AI コパイロットはハイブリッドマネージャーでデフォルトでは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、ハイブリッドマネージャー管理者に *enable and configure it with the AI/LLM of your organization* を要求します。
- 詳細については、 AI Copilot を参照してください。

SPL チェック

!!!note

この機能は、Hybrid Manager Migration Portal に含まれています。Web ベースの移行ポータルには、将来のリリースにも含まれる予定です。

SPL Check は、Oracle プロシージャーコード内の潜在的な実行時エラーを特定するコード分析ツールです。これは、EDB Postgres Advanced Server に移行する際の評価プロセスの重要な部分です。

移行ポータルのハイブリッドマネージャーバージョンで評価されるすべてのスキーマは、SPL Check によって分析されます。このファンクションは、EDB SPL Check 拡張機能に基づいており、`FUNCTION`、`PROCEDURE`、`PACKAGE BODY`、`TYPE BODY`、および `TRIGGER` オブジェクトタイプ内の実行時エラーを検出します。

スキーマが評価された後、チェックに合格しないオブジェクトは、他の互換性問題とともに、移行ポータルプロジェクトで**失敗**としてマークされます。これにより、オブジェクトを宛先データベースにインポートする前にこれらの問題を解決でき、後で実行時エラーを防ぐことができます。

SPL Check エラーは、`[spl_check]` という接頭辞が付いているため、他の非互換性と区別できます。影響を受けるオブジェクトを変更して問題を修正した後、**Reassess** を選択してチェックを再実行し、編集によって問題が解決されたことを確認します。

プロジェクトコラボレーション

!!!note

この機能は、Hybrid Manager Migration Portal に含まれています。Web ベースの移行ポータルには、将来のリリースにも含まれる予定です。

移行ポータルは、共同 DDL 編集をサポートし、同じハイブリッドマネージャー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。別のユーザーユーザー B が、あなたも編集中の DDL を変更すると、あなたの変更ユーザー A が競合解決ビューをトリガーします。このビューには、ユーザー B の更新が **Update** ビューに表示され、独自の編集が **Target** ビューに表示されます。次に、**更新されたものを使用**してユーザー B の修正を採用するか、 **再評価**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。これにより、効率的な同時スキーマ互換性調整が保証されます。

Accessing Migration Portal

ハイブリッドマネージャーの移行ポータルアプリケーションは、使用する方法に応じて異なる方法でアクセスされる 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

ハイブリッドマネージャーを展開すると、移行ポータルのビルトインインスタンスが自動的に含まれます。ただし、このインスタンスには、使用する方法に応じて異なる 2 種類のワークスペースがあります。アクセスも異なります。

- 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、 *Agent Studio in Hybrid Manager* によってロードされたスキーマのハイブリッドマネージャーコンソールで作成された移行ポータル MP 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、MP プロジェクトをハイブリッドマネージャー HM プロジェクトと直接調整します。ここには、自分自身と、同じ HM プロジェクトの一部である他のユーザーの両方が作成したすべての MP プロジェクトが表示されます。ただし、他の HM プロジェクトで実行された移行の MP プロジェクトは表示されません。

- 移行ポータルの **ユーザーのワークスペース**は、分析、変換、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なく、自分のユーザーのみが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と、より柔軟な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 2 種類のワークスペースがあります。これらにアクセスする方法は、移行ポータルを使用する方法によって異なります。

- エージェントによってハイブリッドマネージャー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、MP へのプロジェクトスコープのアクセスを使用します。
- 外部提供の 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 MP 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。ハイブリッドマネージャーは、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、データベースの移行の実行中に *Agent Studio in Hybrid Manager* 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簡単に追加できます。

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。

外部提供の 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

- ハイブリッドマネージャーを使用して *registered an Oracle database* します。
- **オプション**ハイブリッドマネージャーの管理者は、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ために *enabled and configured the AI Copilot with the AI/LLM of your organization* を持っています。

Migration Portal へのアクセス

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. 移行するデータベースの **データベース名**を選択して、データベースの詳細ビューを開きます。

!!!注

Oracle データベースでのみ使用できます。

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、コンソールの **MP Project** 列にプロジェクトの名前のハイパーリンクが表示されます。これにより、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が開きます。

スキーマ移行のため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**+ New MP Project** ボタンまたはスキーマスコープの **+** 記号を選択します。これにより、 *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

ハイブリッドマネージャーから移行ポータルプロジェクトを作成できます。

1. **Estate** ページで、 **Migrations** タブを選択します。
2. 移行するデータベースの **データベース名**を選択して、データベースの詳細ビューを開きます。
3. データベースの詳細ビューで、 **Schemas** タブを選択します。
4. 移行するスキーマの 1 つの横にある **+ New MP Project** またはプラス記号を選択します。
5. 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、**Continue** を選択します。
6. 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。エージェントが正常に取り込んだスキーマ、およびまだ別の MP プロジェクトの一部ではないスキーマのみを選択できます。

注既存のプロジェクトに後で新しいスキーマを追加することはできませんが、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することはできます。

7. **Create New Project** を選択します。

8. バックグラウンドで、これにより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。

ハイブリッドマネージャー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を完了すると、プロジェクトの名前を含むハイパーリンクが追加されます。ハイパーリンクを選択して、移行ポータルを開きます。

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**を無効にします。 **Next** を選択します。

!!!note

Schema integrity and performance considerations を確認して、コンストレインを無効にする必要がある理由と、その他の考慮事項を確認します。

4. [**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。

!!!note

HM 管理のデータベースクラスターに移行する場合読み取り/書き込みホストおよびその他の接続データは、**クラスター** > クラスターを選択> **Connect** で見つけることができます。

5. **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。
6. 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。これで、Migration Portal を閉じることができます。

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。

このためには、 **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ず、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。

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提供された DDL ファイルを評価、変換、および移行します。

エージェント *access the Migration Portal from the Oracle databases' detailed view* を活用するハイブリッドマネージャー移行のスコープで、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**ハイブリッドマネージャーの管理者は、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ために *enabled and configured the AI Copilot with the AI/LLM of your organization* を持っています。

Migration Portal へのアクセス

Launchpad アプリを使用して Migration Portal にアクセスします

1. *Access the Launchpad* 。
2. **Migration Portal** タイルにカーソルを合わせて **View Service** を表示し、表示されたボタンを選択します。これにより、新しいタブが開き、移行ポータルに移動します。

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 。

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、Migration Portal documentation を参照してください。

Migration Walkthroughs

このセクションでは、ハイブリッドコントロールプレーンを使用して移行を実行するのに役立つステップバイステップガイドを集めています。

- *Migrating a self-managed Oracle database to an HM-managed database*
- *Migrating a self-managed Postgres database to an HM-managed database*
- *Migrating a cloud service provider-managed AWS RDS database to an HM-managed database*
- *Using the PG AI Agent for Oracle Migration Readiness*

Migrate your Oracle database to the Hybrid Manager

自己管理 Oracle データベースの HM 管理クラスターへの移行には、いくつかの手順が含まれます。このガイドでは、すべての手順を 1 か所にまとめており、各手順の手順へのリンクがあります。

Oracle データベースを登録します

Oracle データベースをエージェントに登録すると、移行評価と、移行ポータルで使用される DDL スキーマ抽出が提供されます。

Using the Agent for Oracle Migration Readiness を参照して、Oracle データベースを登録します。

Oracle ソースデータベースの準備

エージェントがモニタリングメトリック、移行統計、およびスキーマを取得できるように Oracle データベースを準備しました。次に、DMS Reader がデータベースデータを取得し、移行を実行できるように準備する必要があります。詳細については、Preparing Oracle source databases を参照してください。

HM 管理の宛先を作成する

データベースを HM 管理のデータベースクラスターに移行する場合、優れた Oracle 互換性があるため EDB Postgres Advanced Server を選択することをお勧めします。スキーマの互換性の問題が最小限に抑えられている場合は、PostgreSQL または EDB Postgres 拡張サーバーを選択することもできます。このクラスターは移行先として機能するため、後でスキーマの移行を実行するときにこのデータベースの宛先を選択します。

Creating a database cluster は、HM 管理クラスターを作成する手順を説明します。

ソーススキーマの準備とインポート

ソーススキーマを準備し、宛先データベースにインポートします。この手順は前の手順に依存しないため、移行を作成する前にいつでも実行できます。*create a data migration* を行う前に、スキーマが変換されて宛先データベースに移行されていることを確認する必要があります。

スキーマを準備するために EDB が提供するツールは数多くありますが、Oracle からの移行の場合、ハイブリッドマネージャーには移行ポータルの統合が含まれています。統合を使用してスキーマをインポートするには、*Assessing schemas for HM projects* を参照してください。

EDB DMS リーダーの展開

ソースデータベースサーバーにアクセスするマシンに EDB DMS Reader を展開すると、Reader は Oracle データベースとハイブリッドマネージャー間の接続を実行できるため、移行のソースとして認識されます。

詳細については、 *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator* を参照してください。

EDB DMS リーダーの構成

ハイブリッドマネージャーがソースデータベースデータにアクセスできるように EDB DMS リーダーを構成するには、移行のためにソースデータベースと宛先データベースの両方に接続できるようにリーダースクリプトを構成する必要があります。

ハイブリッドマネージャー **コンソール** から資格情報を取得し、Configuring and running Reader の説明に従ってリーダーを構成します。

EDB DMS リーダーを実行する

EDB DMS リーダーを実行する をクリックして、ソースデータベースの読み取りを開始します。これは移行をまだ実行せず、DMS がアクセスしてソースデータベースを読み取ることができることを保証するだけです。

移行を作成する

ハイブリッドマネージャーへの接続を確立するように EDB DMS Reader を構成した後、コンソールで新しい移行を作成します。移行は、EDB DMS の助けを借りて、ソースデータベースと宛先データベース間の同期を確立します。

手順については、 移行の作成 を参照してください。

移行を完了する

Snapshot オプションを選択した場合、DMS は宛先へのスナップショット全体のコピーが終了した後、移行を自動的に完了としてマークするため、この手順をスキップできます。

移行の作成中に **スナップショット+ストリーミング**オプションを選択した場合、EDB DMS Reader は、移行が完了とマークするまで、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。

手順については、 Stop the streaming process を参照してください。

コンストレインを適用する

EDB Data Migration Service でデータを移行する前に、制約を除外しながらソースデータベースの prepared and imported the schema を実行します。

次に、スキーマとデータの移行から除外された宛先データベースと re-apply the constraints に接続します。

移行の検証

ソースデータベースと宛先データベースを比較して、すべてのデータが移行されたことを確認します。

LiveCompare を使用できます。

Migrate your self-managed Postgres database to the PG AI Hybrid Manager

自己管理の Postgres データベースを PG AI Hybrid Manager で作成したデータベースクラスターに移行するには

1. *EDB DMS* リーダーの展開
2. *PG AI Hybrid Manager* によって管理される宛先データベースクラスターを作成する
3. *EDB DMS Reader* がデータにアクセスできるように、*Postgres* ソースデータベースを準備します
4. *EDB DMS* リーダーの構成
5. *EDB DMS* リーダーを実行する
6. ソーススキーマを準備し、ターゲットデータベースにインポートする

7. 移行を作成する

EDB DMS リーダーの展開

データベースサーバーにアクセスするマシンに EDB DMS Reader を展開すると、Reader は Postgres データベースと PG AI Hybrid Manager 間の接続を実行できるため、移行ソースとして認識されます。

詳細については、 *Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator* を参照してください。

PG AI Hybrid Manager によって管理される宛先データベースクラスターを作成する

データベースを PG AI Hybrid Manager に移行するときは、 EDB Postgres Advanced Server データベースクラスターを作成する必要があります。このクラスターは移行先として機能するため、後で *schema migration* を実行するときにこのデータベースをターゲットとします。

詳細については、 *Creating a database cluster* を参照してください。

EDB DMS Reader がデータにアクセスできるように、Postgres ソースデータベースを準備します

詳細については、 *Preparing Postgres source databases* を参照してください。

EDB DMS リーダーの構成

PG AI Hybrid Manager がソースデータベースデータにアクセスできるように EDB DMS Reader を構成するには、移行のソースデータベースとターゲットデータベースの両方へのアクセスを付与するリーダースクリプトにいくつかの値を入力する必要があります。

1. Getting credentials の説明に従って、 **PG AI Hybrid Manager Console** から資格情報を取得します。
2. Configuring the reader セクションの手順を実行しますが、リーダースクリプトで設定された次の変数を設定していることを確認します。

```
export CLOUD_PROVIDER=appliance
export RW_SERVICE_HOST=<transporter-rw-service-host>
```

!!!note

`RW_SERVICE_HOST` を、PG AI Hybrid Manager イングレスに関連付けられたドメイン名またはホストに設定します。

インストール中に `values.yaml` ファイルを介してインスタンスに割り当てた `TRANSPORTER_RW_SERVICE_DOMAIN_NAME` から `RW_SERVICE_HOST` を導き出し、 `/transporter` を追加します。

または、PG AI コンソールへのアクセスに使用する URL から `RW_SERVICE_HOST` を取得します。たとえば、URL が `https://portal-transporter.foo.network` の場合、ドメイン名を取得し、最後に`/transporter` を追加します。

この例では、次を設定する必要があります。

`export RW_SERVICE_HOST=transporter.foo.network/transporter`

EDB DMS リーダーを実行する

Execute the EDB DMS Reader して、ソースデータベースの読み取りを開始します。これは移行をまだ実行せず、DMS がアクセスしてソースデータベースを読み取ることができることを保証するだけです。

ソーススキーマを準備し、ターゲットデータベースにインポートする

この手順は前の手順に依存しないため、移行を作成する前にいつでも実行できます。*create a data migration* の前に、スキーマが変換され、ターゲットデータベースに移行されていることを確認する必要があります。

スキーマを準備するために EDB が提供するツールは多数あります。自己管理の Postgres インスタンスからの移行の場合、EDB Migration Toolkit を使用することをお勧めします。MTK's offline migration 機能は、データベースのスキーマを抽出し、制約を分離する簡単な方法を提供します。

!!!note

スキーマをターゲットデータベースにインポートする前に、DDL から外部キー、チェック、および除外制約を除外していることを確認します。Schema integrity and performance considerations を確認して、コンストレインを無効にする必要がある理由と、その他の考慮事項を確認します。

移行を作成する

PG AI コンソールで

1. 移行を作成する 。
2. 移行が成功したことを確認します。psql でテーブル、行、スキーマなどのスポットチェックを実行したり、LiveCompare で両方のデータベースを比較したりできます。
3. **スナップショット+レプリケーション** 移行を選択した場合 mark the migration as completed 。それ以外の場合、この手順をスキップします。
4. Re-apply any excluded constraints 。

Migrate your AWS RDS Postgres database to the Hybrid Control Plane

AWS RDS データベースクラスターは、CSP 管理のデータベースです。

AWS RDS Postgres データベースから HM 管理データベースにデータを移行するには

1. ビーコンエージェントを *AWS RDS* データベースに展開する
2. *AWS RDS* ソースデータベースの準備
3. ハイブリッドコントロールプレーンによって管理される宛先データベースクラスターを作成する
4. スキーマを宛先データベースにインポートする
5. *EDB DMS Reader* を補助マシンに展開する
6. *Configure the EDB DMS Reader, so it can access source data*
7. 移行を作成する
8. 移行を完了する

9. コンストレインを適用する
10. *Verify the migration*

ビーコンエージェントを AWS RDS データベースに展開する

データベースサーバーにビーコンエージェントを展開すると、ハイブリッドコントロールプレーンは可観測性データと監視データを生成できます。この手順は、データベースの移行に厳密に必要なわけではありませんが、ハイブリッドコントロールプレーンの外部を含むすべてのデータベースインスタンスの単一の統合ビューを提供するという大きな利点があります。

構成手順については、 *Monitoring AWS RDS deployments on AWS* を参照してください。 **Estate** にアクセスして、RDS インスタンスの概要を表示します。

ハイブリッドコントロールプレーンに移行するデータベースを決定します。

AWS RDS ソースデータベースの準備

EDB Data Migration Service を使用して AWS RDS データベースからデータを移行する前に、ソースデータベースを準備して、EDB DMS Reader がソースに接続し、レプリケーションを実行できるようにする必要があります。

論理レプリケーションを有効にし、移行のユーザーを作成する方法の詳細については、*Preparing AWS RDS source databases* を参照してください。

ハイブリッドコントロールプレーンによって管理される宛先データベースクラスターを作成する

既存の HM 管理対象クラスターを使用するか、新しいクラスターを作成して、移行の宛先として使用できます。

詳細については、 *Creating a database cluster* を参照してください。

スキーマを宛先データベースにインポートする

RDS データベースからの移行の場合、スキーマのエクスポートとコンストレインの処理にオフラインモードで Migration Toolkit を使用することをお勧めします。

ソーススキーマのエクスポート、制約の処理、および宛先データベースへのスキーマのインポートの方法については、Preparing schema を参照してください。

EDB DMS Reader を補助マシンに展開する

RDS データベースにアクセスするマシンに EDB DMS Reader を展開します。これを補助マシンと呼びます。

!!!note

補助マシンが RDS インスタンスにアクセスできるようにする方法の 1 つは、RDS インスタンスにインバウンドルールを追加することです。RDS インスタンスに割り当てられた VPC セキュリティグループで、 **タイプ** PostgreSQL と補助マシンの IP アドレスを使用して **インバウンドルール**を追加します。

詳細は、 Installing EDB Data Migration Service Reader を参照してください。

EDB DMS Reader の構成と起動

補助マシンで EDB DMS Reader を構成して実行するには、HM 移行資格情報を取得し、EDB DMS Reader にこの資格情報およびその他の設定へのアクセスを提供する必要があります。

これにより、EDB DMS Reader はソースデータベースにアクセスし、HM 移行サービスに接続できます。

詳細については、 Configuring and running the EDB DMS Reader を参照してください。

!!!note

RDS データベースの場合、 `DBZ_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。

移行を作成する

HM への接続を確立するように EDB DMS Reader を構成した後、HM ポータルで新しい移行を作成します。移行は、EDB DMS の助けを借りて、ソースデータベースと宛先データベース間の同期を確立します。

手順については、 移行資格情報の作成 を参照してください。

移行を完了する

Snapshot オプションを選択した場合、DMS は宛先へのスナップショット全体のコピーが終了した後、移行を自動的に完了としてマークするため、この手順をスキップできます。

移行の作成中に **スナップショット+ストリーミング**オプションを選択した場合、EDB DMS Reader は、移行が完了とマークするまで、ソースデータベースで実行されたデータ更新をターゲットデータベースにストリーミングし続けます。

手順については、 Stop the streaming process を参照してください。

コンストレインを適用する

EDB Data Migration Service EDB DMS を使用したデータ移行の開始時に、ソースデータベースの prepared and imported the schema を行います。次に、ターゲットデータベースに接続し、スキーマとデータの移行から除外された制約を再適用します。

移行の検証

ソースデータベースとターゲットデータベースを比較して、すべてのデータが移行されたことを確認します。

LiveCompare を使用できます。

Using the Agent for Oracle Migration Readiness

Oracle データベースソースから移行を開始する前に、エージェントを使用してハイブリッドマネージャーに接続します。Oracle データベースをエージェントに登録すると、次のリソースにアクセスできます。

- 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。**Estate** ページの *Migrations* から入手できます。

- 移行データベース *Detailed view* は、 **スキーマ**の拡張分析、グローバルでより詳細な **アセスメント**、およびコンソールの他のツールへの直接接続を提供する **移行**ハブを提供します。移行の実行に役立ちます。これらのリソースは、**Estate** ページ> **Migrations** タブから特定の Oracle データベースを選択すると、利用できます。
- ビルトインの移行ポータルでスキーマの移行を実行するために活用できる自動スキーマ DDL 抽出。

Oracle データベースの準備

エージェントには、Oracle データベースを評価し、スキーマおよび移行に必要なその他の情報を抽出するために、ソースデータベースへのアクセスと必要な権限を持つ Oracle ユーザーが必要です。

手順については、 *Oracle* **インスタンスの準備** を参照してください。

マシンユーザーを作成する

エージェントは、マシンユーザーがハイブリッドマネージャーにデータを取り込む必要があります。

手順については、 *Creating a machine user* を参照してください。

エージェントのインストール

Oracle データベースにアクセスするマシン上の *Agent* **パッケージのインストール** 。

エージェントの構成

エージェントの構成 、Oracle データベースがハイブリッドマネージャーに接続し、ソースデータベースから移行データとスキーマ DDL を取得できるようにします。

エージェントを実行する

エージェントのインストールと構成が完了すると、*Running the Agent* および *running it as a service* に進むことができます。

ハイブリッドマネージャーへの中断のない接続を維持するには、エージェントをサービスとして実行する必要があります。手動実行では、ホストマシンが動作している限り接続を維持できますが、エージェントをサービスとして実行すると、システムの自動起動が保証され、予期しないシャットダウン後に再起動されます。

エージェントをサービスとして設定しない場合、手動で実行したときにのみ評価を実行し、スキーマデータをストリーミングします。さらに、デフォルトでは、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 時間にハイブリッドマネージャーの**エステート** から消えます。

!!!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。

次のステップ

エージェントが実行されたら、次の手順については *Migrate your Oracle database to the Hybrid Manager* を参照してください。

Migrating databases

Migrate オプションを使用して、次のことを行います。

- 移行データベース **主要なリンクと学習リソース** を追加および管理します。
- 移行データベース *Destinations* を管理します。
- データベース *Migrations* を実行します。
- データベース移行用にプロジェクトスコープの *Credentials* 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、*Install Hybrid Manager on Red Hat OpenShift using EDB Postgres AI Operator* 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、`cdcreader` ID、現在の *InferenceServices* **のステータスの検証** 、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 Reader 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 Reader 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 Reader 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。 `cdcreader` ログを調査して、特定の原因を理解します。

!!!note

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を、別の移行に使用することはできません。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、*EDB DMS Writer* 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、`cdcwriter` ID、現在の *InferenceServices* のステータスの検証 、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 Writer 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 Writer 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 Writer 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 `cdcwriter`ログを調査して、特定の原因を理解します。

!!!note

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を、別の移行に使用することはできません。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。
- **Status** —移行の現在のステータス。

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 リーダー/ライターがソースおよび/またはターゲット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。

!!!note

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を、別の移行に使用することはできません。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。

移行の作成

移行を作成するには、**Create Migration** を選択します。詳細は、移行資格情報の作成 を参照してください。

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。このページには、移行のステータス、移行のタイプ、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。

移行の作成 のときに **スナップショット+ストリーミング**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。

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 リーダーまたはライターが 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

移行資格情報を作成およびダウンロードする方法については、Getting credentials 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`

Assessing Migrations

ハイブリッドマネージャーの主な機能の 1 つは、データベースの移行を促進することです。

Estate ページからアクセスできる **Migrations** タブには、*Agent Studio in Hybrid Manager* に登録した外部データベースのリストが表示されます。

これは、外部データベースをハイブリッドマネージャーに移行する際の最初の目的地です。

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、ステップ *7* 互換性のためにクラスターを更新する 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、エージェントの パラメーターリファレンス 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、ステップ *7* 互換性のためにクラスターを更新する を参照してください。

of Schemas: DSN ユーザーがアクセスし、DSN ユーザーが `SELECT` 特権を持つ少なくとも 1 つのテーブルを含む評価対象データベース内のスキーマの数。DSN ユーザーは、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。

!!!note s

- 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、通常除外されますが、既知の問題により、抽出中に DDL エクストラクターが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。
- 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。

or Cores: Oracle データベースに割り当てられたコアの数。

Complexity: データベースをハイブリッドマネージャーに移行するための複雑さのレベル。指定できる値は、Very Easy、Easy、Medium、Hard、および Very Hard 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 スキーマを変換し、ハイブリッドマネージャーにデータを移行するために必要な 8 時間営業日の労力レベル。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。

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分自身管理自己管理、clud サービスプロバイダー、またはハイブリッドマネージャー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、エージェントの パラメーターリファレンス 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、エージェントの パラメーターリファレンス 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** > **Migration**] タブには、エージェントに登録した後にハイブリッドマネージャー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** および**プロジェクト名** ごとに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。
- 稲妻アイコンでテーブルを更新する頻度を選択します。無効になっている場合、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、30 秒ごとにテーブルが更新されます。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

- ステップ *7* 互換性のためにクラスターを更新する 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。
- *Data security and sovereignity* を参照して、データがエージェントに保存される方法を理解します。

Detailed view

Estate ページ> **Migrations** 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。この詳細ビューには、外部データベースからの移行を実行するために活用できる機能が含まれています。

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。ここには、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あります。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。さらに、**Schemas** タブでは、抽出したスキーマから直接 *create new Migration Portal projects* を行うことができます。

!!!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。

概要カード

- 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。
- 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非常に簡単、簡単、中程度、難しい、非常に難しい。
- 8 時間労働日で定義された日範囲 0-5、6-15、16-25、26-40、40+日で色分けされた努力のレベル。

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。可能な値は、未開始、進行中、Complete、および Error です。
- **最終評価**最後にスキーマが評価された時刻に対応するタイムスタンプ。新しいスキーマが利用可能なことをエージェントが検出した場合、または既存のスキーマが更新された場合、評価は自動的に再生成または更新されます。
- **# of Tables:** 評価されたスキーマのテーブルの数。
- **# of Objects:** 評価された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、特定のスキーマを 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、特定の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。
- **MP プロジェクト**統合された移行ポータルを使用して、Oracle スキーマを EDB Postgres Advanced Server に変換します。プラス記号を選択してウィンドウが開き、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**で新しいスキーマ移行プロジェクトを開始するための詳細を提供できます。詳細を入力すると、ハイブリッドマネージャーはバックグラウンドで移行ポー

タルプロジェクトを作成します。*Customizing Migration Portal secrets for secure internal communication* タブからこのプロジェクトに後でアクセスできます。

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 は、主要な分析結果と評価メタデータを要約したものです。

互換性の概要は、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労力のレベルに寄与するその他の要素をまとめています。複雑さは、複数の要因を分析するスコアリングシステムに応じて割り当てられます。要素には、データベースサイズ、ラージオブジェクトの使用量、使用されるパーティションタイプ、Oracle データベース機能、プロシージャルデータベースオブジェクト、および EPAS 互換かどうかが含まれます。分析された各オブジェクトには、1〜5 のスコア非常に簡単から非常に難しいまでが与えられます。最大複雑度の値がここに表示されます。

使用される Oracle PL/ SQL 機能は、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。 Oracle PL/ SQL 機能の移行の影響を理解するには、オブジェクト数、出現数、EPAS 互換かどうか、および移行の複雑さ評価の結果を参照します。

特定の機能のすべての呼び出しに、同等の EDB Postgres Advanced Server 実装がある場合、その機能は **EPAS 互換**としてマークされます。同等のものが検出されない場合、機能は非互換としてマークされます。

使用されている Oracle データベース機能は、データベース内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。

オブジェクト内の互換性のない PL/ SQL 機能は、Oracle および非 Oracle 以外の互換性のない PL/ SQL 機能をすべてリストし、そのスキーマ、オブジェクトタイププロシージャ、ファンクションなど、オブジェクト名、互換性のない PL/SQL 機能を表示します機能名、機能が表示される頻度、および移行の複雑さ。

移行

移行関連のタスクの実行に役立ちます。

宛先の作成では、移行の宛先として機能する新しいクラスターを作成できます。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 では、ビルトイン移行ポータル **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**でプロジェクトを作成し、Oracle スキーマを EDB Postgres Advanced Server と互換性のあるものに変換するために使用できます。非互換性を解決すると、移行ポータルサービスを使用して宛先クラスターに接続し、変換されたスキーマをインポートすることもできます。

データの移行は、ハイブリッドマネージャーの Data Migration Service を活用して、CDC テクノロジーでデータの移行を実行します。この機能で移行を実行する前に、データベースの準備、スキーマのエクスポート、EDB DMS リーダーおよびライターのインストールと構成が完了していることを確認してください。Oracle データベースから移行するために実行する必要がある手順の時系列の概要については、*Migrate your Oracle database to the PG AI Hybrid Manager* を参照してください。

Compatibility

現在、ハイブリッドマネージャーは、次の **Oracle** データベースの評価をサポートしています。

- 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、これらのバージョンはまだ認定されていません。

Postgres データベースは **Estate > Migrations** ビューにも表示されますが、評価はまだサポートされていません。

Asset Management

Hybrid Manager 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。

ハイブリッドマネージャー Postgres とモデルアセットの管理

- *Asset Library in Hybrid Manager*

Hybrid Manager 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。

- *HM* コンテナイメージをプライベートレジストリに同期する

イメージの詳細の検索、フィルタリング、表示を含む、アセットライブラリーのコンテナイメージの探索と管理に関する詳細なガイド。

- リポジトリの管理

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ー内のリポジトリと画像の可視性を制御する方法。

- ナレッジベース

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ーでの AI モデルの管理に関するガイド。

The Asset Library

アセットライブラリーは、ハイブリッドマネージャークラスターで使用できるアセット、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ーへのアクセス

アセットライブラリーにアクセスするには、ハイブリッドマネージャーコンソールの上部にある **Asset ライブラリー** タブをクリックします。

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。

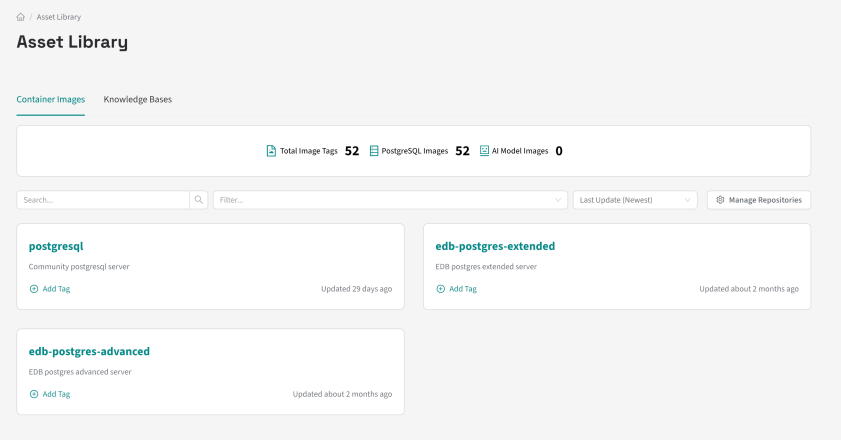


図 7: Asset Library Page

アセットライブラリーは、 **コンテナイメージ** と **ナレッジベース**の 2 つのメインセクションに分かれています。各セクションでは、ハイブリッドマネージャークラスターで使用可能な資産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ーの **Container Images** セクションでは、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、*HM* **コンテナイメージをプライベートレジストリに同期する** ガイドを読んでください。

ナレッジベース

ナレッジベースのモデルは、ハイブリッドマネージャーで使用できる AI モデルです。これらのモデルは、特定のデータセットでトレーニングでき、自然言語処理、画像認識などのさまざまなタスクを実行するために使用できます。このタブの使用方法の詳細については、**ナレッジベース** ガイドを読んでください。

Container Images in the Asset Library

アセットライブラリーの **Container Images** セクションでは、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あり、各イメージは名前とイメージタグで識別されます。

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、公式のハイブリッドマネージャーイメージには、イメージ名が `postgresql` があり、イメージタグが `17.4-2503191025` がバージョン 17.4 のコミュニティ Postgresql イメージを表します。`2503191025` は、イメージのビルド日時 2025/03/19 10:25 です。さらに後続の `-full` は、このイメージに Hybrid Manager 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ています。

このタグ形式は公式のハイブリッドマネージャーイメージに固有であり、他のイメージとは異なる場合があることに注意してください。イメージ名は、Postgres の特定のフレーバーなど、画像のタイプを識別するために使用されますが、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。

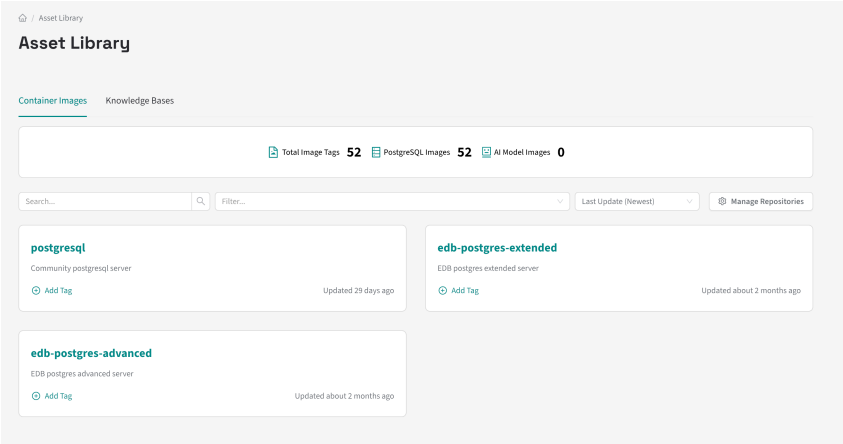


図 8: Container Images Page

画像統計

[Container Images] ページの最初の部分は、ライブラリで使用可能なイメージの数の概要をタイプ別に表示します。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。

次に、ページは Container Images タブのメインコンテンツに移動します。

コンテナイメージ 検索およびフィルタバー

このビューでは、 **Search** フィールドに、イメージリポジトリの名前または説明に含まれる検索用語を入力できます。**Filter** フィールドを使用すると、リポジトリのタイプ Postgres または AI モデルでリポジトリをフィルタリングできます。

Sort フィールドでは、表示リポジトリをリポジトリの名前昇順または降順、説明別昇順または降順、リポジトリタイプ昇順または降順、またはリポジトリが更新された日付でソートできます。昇順または降順。

最後に、検索とフィルターバーの **Manage Repositories** ボタンを使用して、ライブラリで使用可能なリポジトリを管理できます。これにより、リポジトリを追加、編集、または削除できるダイアログが開きます。これにより、**Manage Repositories** ページが表示され、アセットライブラリー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、リポジトリの管理 ガイド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**このイメージリポジトリに関連付けられたハイブリッドマネージャータグ。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、このフィールドには**タグの追加** 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すると、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。また、 **Add Tag** ボタンもあり、ハイブリッドマネージャータグをイメージ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、タグをクリックして、そのタグの詳細を表示できます。

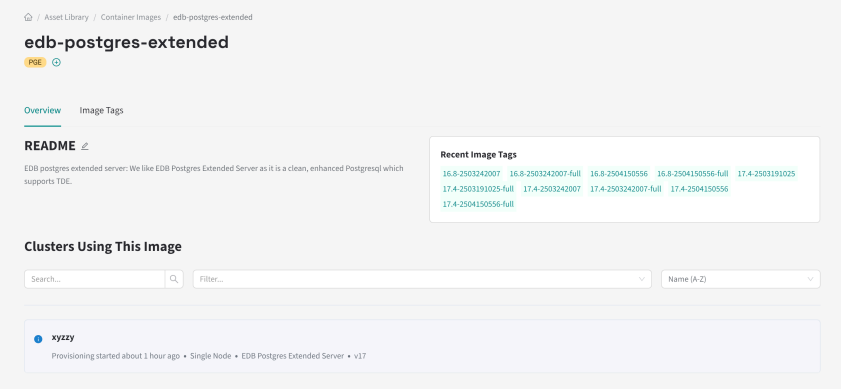


図 9: Image Details Page - Overview Tab

[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。
- **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。各タグは選択可能で、そのタグの**イメージタグの詳細**ページに移動します。

README と最近のイメージタグの下に、**このイメージを使用したクラスター**のセクションがあります。これは、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をリストします。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。

クラスターの検索を支援するために、検索およびフィルターバーが提供されます。**Search** フィールドでは、クラスターの名前または説明に含まれる検索用語を入力できます。**Filter** フィールドを使用すると、クラスターを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、Postgres バージョンでフィルタリングできます。**Sort** フィールドを使用すると、クラスターの表示を名前昇順または降順、説明昇順または降順、またはクラスターが作成された日付昇順または降順でソートできます。

クラスター名を選択すると、そのクラスターの **クラスターの詳細**概要ページに移動します。

画像タグタブ

[**Image Tags**] タブには、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が表示されます。

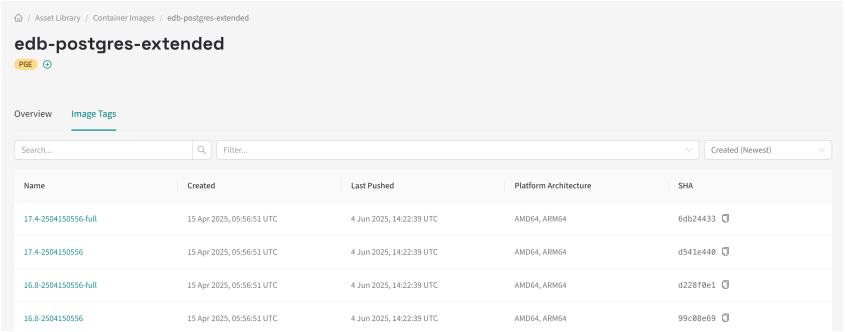


図 10: Image Details Page - Image Tags Tab

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** X86、AMD64、ARM64、PPC64LE、S390X など、イメージタグが利用可能なプラットフォームアーキテクチャ。
- **SHA** 画像の SHA256 ハッシュ。画像の一意の識別子。他のリポジトリと比較してイメージを検証するために使用できます。ハッシュの最後の 8 桁が表示され、これらの数字にカーソルを合わせると完全なハッシュを利用できます。表示されているハッシュの横にあるコピーアイコンをクリックして、完全なハッシュをクリップボードにコピーすることもできます。

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** フィールドでは、画像タグの名前に含まれる検索用語を入力できます。**Filter** フィールドを使用すると、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できます。**Sort** フィールドを使用すると、イメージタグの表示をバージョン別昇順または降順、作成日別昇順または降順、最後にプッシュした日昇順または降順でソートできます。

イメージタグの名前を選択すると、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 ページは、特定の画像タグに関する最も詳細な情報を提供します。

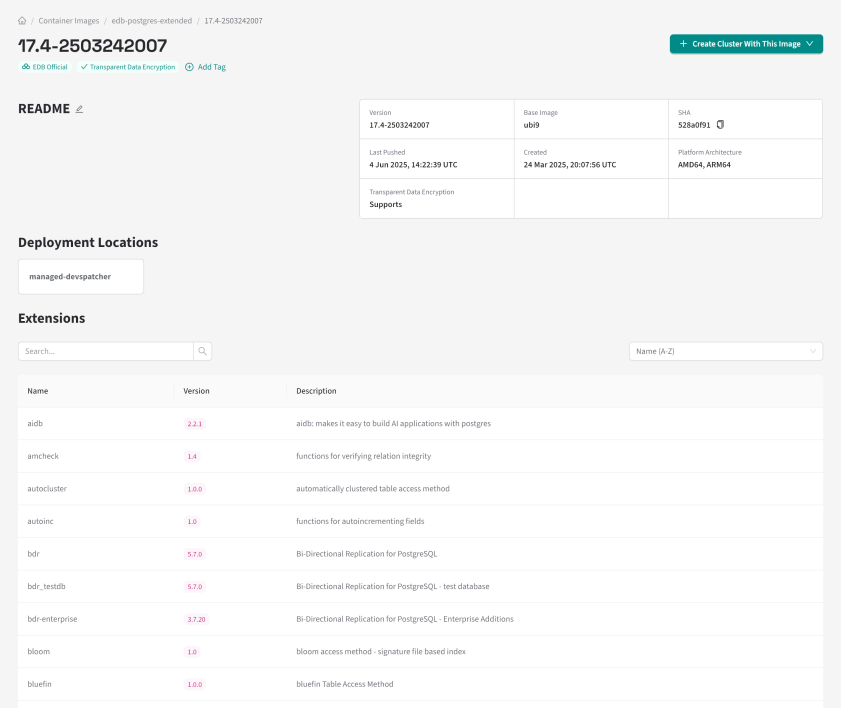


図 11: Image Tag Details Page

ページの上部に、イメージタグの名前が表示されます。その下には、このイメージタグに関連付けられているハイブリッドマネージャータグが存在する場合があります。これには、プラットフォームによって生成される EDB 公式タグと透過的データ暗号化タグ、およびユーザーが作成したタグが含まれます。これらのタグの横には **Add Tag** があり、ユーザーが作成したタグを画像タグに追加または削除し、必要に応じて新しいタグを作成できます。

以下には、次のセクションが含まれます。

- **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 **プラットフォームアーキテクチャ** X86、AMD64、ARM64、PPC64LE または S390X など、イメージタグが利用可能なプラットフォームアーキテクチャ。
  - **最終プッシュ**日付とイメージタグがリポジトリにプッシュされた時間。
  - **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、AMD64、ARM64、PPC64LE または S390X.

- **透過的データ暗号化**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。

この情報の下には、 **展開場所**というラベルのフィールドがあり、イメージタグが利用可能な場所をリストしています。

その下は、この画像タグに含まれる **拡張機能**のテーブルです。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。テーブルの上には、名前または説明で特定の拡張機能を検索できる検索フィールドがあります。テーブルは名前で昇順または降順でソートすることもできます。

最後に、 **このイメージタグを使用したクラスター**のセクションがあります。これは、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をリストします。このイメージタグ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。

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ーで使用可能なリポジトリを管理できます。

これは、各パターンがルールとして定義されている、特定のパターンに一致するリポジトリとタグ付け画像をシステムに含めることができることにより実現されます。これにより、どのリポジトリをアセットライブラリーに含めるか、およびハイブリッドマネージャークラスターで使用できるイメージを制御できます。

デフォルトでは、アセットライブラリには、すべての公式ハイブリッドマネージャーリポジトリと、こ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ー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ために、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、必要に応じて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は引き続き動作し、アセットライブラリーで使用可能なものを制御します。

リポジトリの管理

Manage Repositories ページでは、アセットライブラリーで使用可能なリポジトリ登録を追加、編集、または削除できます。このページには、アセットライブラリーの **コンテナイメージ** タブにある **Manage Repositories** ボタンをクリックしてアクセスします。

画面には、アセットライブラリーに現在登録されているリポジトリのテーブルが表示されます。次の列。

- **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。

各行にデータが入力されると、リポジトリ登録を編集できる **Edit** ボタンと、リポジトリ登録を削除できる **Delete** ボタンもあります。

特定のリポジトリ登録を名前で検索できる検索フィールドと、リポジトリ名または作成日でテーブルをソートできるソートコントロールもあります。

画面の右上隅に、 **新しいリポジトリの登録**というボタンがあります。このボタンをクリックすると、新しいリポジトリの登録を追加できるダイアログが開きます。

新しいリポジトリの登録

Register New Repository ボタンをクリックすると、ダイアログが開き、新しいリポジトリ登録の詳細を入力できます。

ダイアログには次のフィールドが含まれています。

- **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ー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ー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、`pgaihm-` 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`^pgaihm-.*$` 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が、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ー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、`latest` タグを持つすべての画像を含める場合は、正規表現として`^latest$` 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が、正規表現を使用して、タグが目的のパターンと完全に一致することを確認することをお勧めします。

[**Add Rule**] ボタンをクリックして、複数のルールを追加できます。ルールのリストに新しい行が追加されます。ルールの横にある **Delete** ボタンをクリックして、ルールを削除することもできます。

新しいリポジトリ登録の詳細を入力したら、**Register Repository** ボタンをクリックして登録を保存します。新しいリポジトリ登録は登録リポジトリのリストに追加され、アセットライブラリー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。

[**Cancel**] ボタンをクリックして登録をキャンセルすることもできます。変更を保存せずにダイアログを閉じます。

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。

Asset Library in Hybrid Manager に戻って、新しいリポジトリールールの効果と、アセットライブラリーで使用できるよう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ーのナレッジベースを使用すると、ハイブリッドマネージャー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、検索およびフィルターバー、ソートコントロール、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。

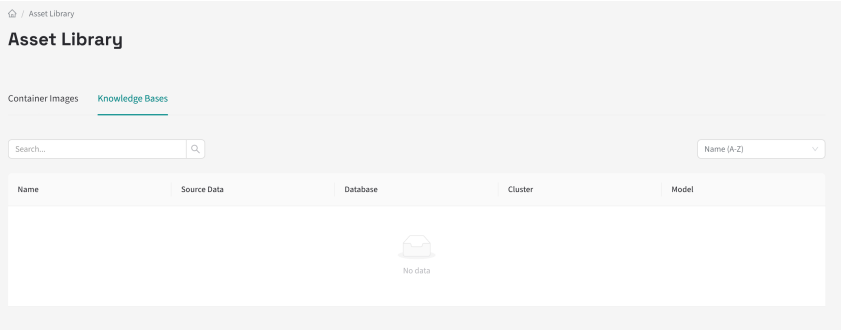


図 12: Knowledge Bases Page

- **モデル** ナレッジベースの構築に使用される AI モデル。

About Backups and Restores

基本

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。

- *PGAIHM バックアップと DR*
- *About Backups and Restores*

Backups

EDB PG AI Hybrid ManagerPGAIHM は、さまざまなテクノロジーを使用したさまざまな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。

PGAIHM は、2 つの異なる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。
- **PGAIHM バックアップ** PGAIHM 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 データベース内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。次のオプションが使用できます。

- **ベースバックアップ**ベースバックアップは、すべての PGAIHM 展開で利用できます。
- **スナップショットバックアップ**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、書き込み先行ロギング 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。これにより、非常に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。

PGAIHM バックアップコントロールプレーン

PGAIHM 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 PGAIHM システム自分自身を保護します。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。PGAIHM バックアップはファイルシステムベースであり、メタデータの時間ごとのバックアップと、デフォルトで PGAIHM のシステム Postgres データベースのバックアップが行われます。

Postgres バックアップとは異なり、PGAIHM 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。これは、復旧が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ることを意味します。

主な違いとリカバリーの影響

これらの違いのため、目標復旧ポイント RPO と目標復旧時間 RTO は、ユーザーデータ Postgres と PGAIHM システムで異なります。

- **Postgres データ)** 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へのリカバリーが可能になる可能性があります。
- **PGAIHM システム)** RPO は、ユーザー変更可能なデフォルトで時間ごとのバックアップスケジュールに制限されます。

Component	RPO	RTO	Method	Location
PGAIHM	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、ハードウェアのパフォーマンスによっては最大 1 TB が有効なサイズ制限であると予想されます。1TB を超えるデータを含む Postgres クラスターは、サービスの品質に悪影響を与えるバックアップ期間とパフォーマンスの負荷が発生する場合があります。

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

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予想され、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。これにより、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。

スナップショットをサポートする**エンタープライズストレージと CSI ドライバー**で PGAIHM が構成されている場合、PGAIHM 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 詳細

- デフォルトでは、AWS ボリュームスナップショットは **トランスポート不可**です。
- 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 詳細

- TopoLVM スナップショットは **トランスポート不可**であり、別のホストに復元することはできません。

構成

次のセクションを使用して、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します。

- 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
- Postgres DistributedPGD 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。

- PGAIHM を使用して展開されたシングルノードクラスター。
- 保持期間とバックアップ時間は変更するか、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、その後毎日発生します。
- 手動バックアップを開始できます。

バックアップ プライマリ/スタンバイクラスタープライマリ/スタンバイクラスターのバックアップには、次の特徴を持つことができます。

- PGAIHM を使用して展開されたプライマリ/スタンバイクラスター。
- 保持期間とバックアップ時間は変更するか、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、その後毎日発生します。
- 手動バックアップを開始できます。
- 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、次の特徴があります。

- PGAIHM を使用して展開された PGD クラスター。
- 保持期間とバックアップ時間は変更するか、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、その後毎日発生します。
- 手動バックアップを開始できます。

- 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。

PGAIHM バックアップ

PGAIHM バックアップには、PGAIHM 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。PGAIHM バックアップでは、Velero と呼ばれるツール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、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 PGAIHM 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指すものではありません。

バックアップの取得

次の手順を使用して、Postgres または PGAIHM バックアップを取ります。

Postgres のバックアップの取得

PGAIHM Web 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、PGAIHM Web コンソール内の API ドキュメントリンクを参照してください。

次の手順を使用して、PGAIHM Web コンソールで Postgres バックアップを取得します。

Backups タブを選択して、既存のバックアップを表示します。

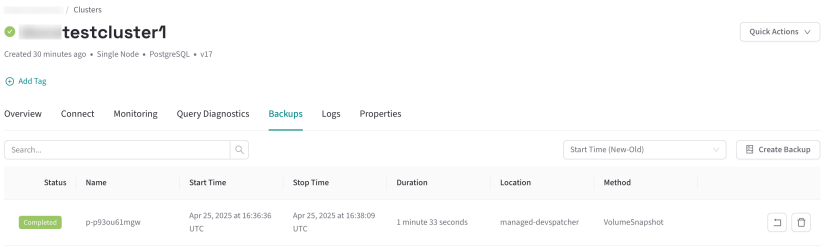


図 13: Viewing Backups

次のフィールドが使用できます。

Name	Description
Status	バックアップの現在のステータス。
Name	バックアップの名前。
Start Time	バックアップが開始された時刻。
Stop Time	バックアップが停止した時間。
Duration	バックアップが完了するまでにかかった時間。
Location	バックアップの場所。
Method	バックアップが取られた方法。

Create Backup タブを選択して、新しいバックアップを作成します。

次のフィールドが使用できます。

- ロケーション
- バックアップ名
- バックアップ方法

これらのフィールドにデータを入力した後、 **OK** を選択します。

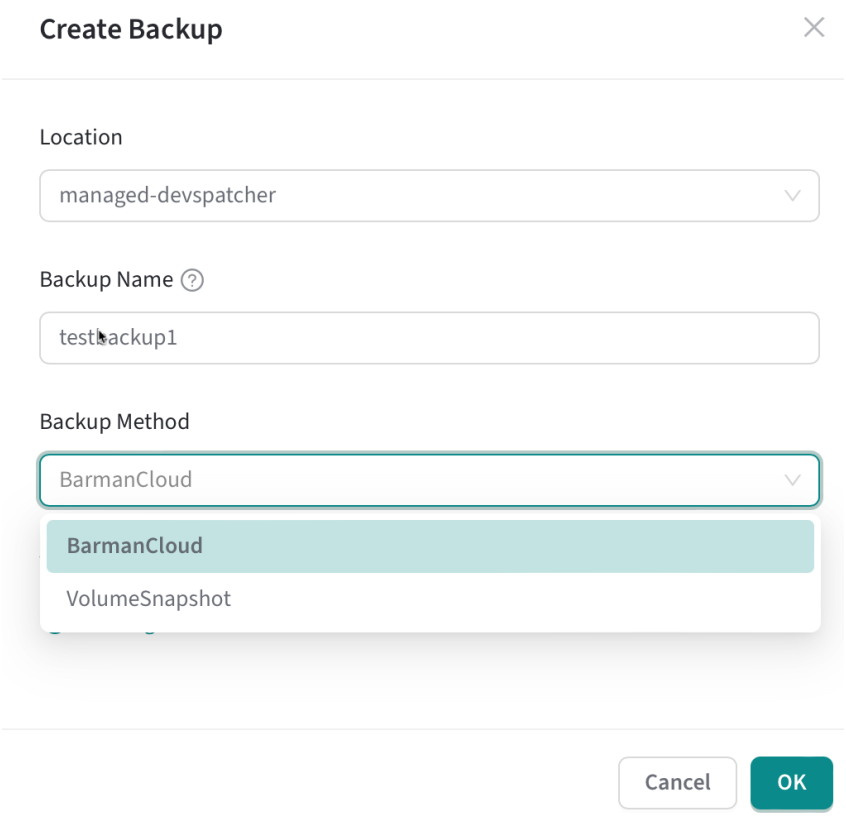


図 14: Create Backup

PGAIHM バックアップの取得

PGAIHM のバックアップを構成する方法については、サイトの HA/DR セクションを参照してください。

用語

次の用語は、Postgres および PGAIHM 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

バックアップは、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です。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と展開タイプたとえば CSP、エンジニアドシステム、または RHOS によって異なります。

バックアップ保持期間

バックアップ保持期間は、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期間です。組織が復旧のためにバックアップを使用できる期間を定義します。バックアップを保持する期間です。

Barman

Barman は、主に EDB によってメンテナンスおよびサポートされているオープンソースの Postgres バックアップツールです。単独で使用される場合、Barman という用語は、以下で説明する Barman クラウドユーティリティスクリプトとは対照的に、「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 B\&R に必要な Postgres のバックアップおよび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 がブロックレベルのインクリメンタルバックアップを提供する前に、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、多くの革新的な機能を提供します。

Barman Cloud ユーティリティスクリプト

Barman Cloud Utility Scripts 、または略して「Barman Cloud」は、Barman サーバーとは別に使用できるスクリプトのコレクションであり、その目的は、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。Barman Cloud 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します。

ベースバックアップ

「ベースバックアップ」という用語は、 Postgres バックアップユーティリティ *pg_basebackup* を指します。このユーティリティは、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

クラウドは、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 を指します。

クラスター

クラスターという用語は、単一の Postgres インスタンス別名 Postgres クラスター内のデータベースのコレクションを参照するために使用されます。PGAIHM では、展開タイプつまりシングルノード、PSR、PGD クラスターを参照するためにも使用されます。

クラウドネイティブ Postgres

Cloud Native Postgres CNP は、EDB によって開発および保守されているオープンソースの Postgres オペレーターを指します。PGAIHM の Kubernetes での Postgres の基盤を提供します。

ダンプ

「dump」という用語は、 Postgres ユーティリティ *pg_dump* を指します。以下の pg_dump を参照してください。

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

これは、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です。データベースの一部または全体をエクスポートするための多くの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。そうは言っても、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。PGAIHM ユーザーは、pg_dump を使用して Postgres データにアクセスできることを想定しています。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

リカバリーは、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です。

目標リカバリポイント

目標復旧ポイント RPO は、データベースタイムライン内の復旧に使用できるポイントまたは時間たとえば 5 分を指します。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、時間で測定されたデータ損失の最大許容量であり、中断後に組織が耐えることができます。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。

目標リカバリ時間

目標復旧時間 RTO は、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です。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 分を超えないことを意味します。

リストア

リストアとは、ファイルをバックアップ場所から元の場所または代替の場所にコピーして、復旧の準備をするプロセスを指します。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

ここで「スナップショット」という用語は、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を指すために使用されています。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、PGAIHM 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

トランスポータブルボリュームスナップショットは、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です。これにより、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。

トランスポートブル以外のボリュームのスナップショット

非トランスポートブルボリュームのスナップショットは、特定のストレージシステムまたは作成された場所に制限されているデータのポイントインタイムコピーです。これは、主にそのシステム内のローカルデータの復旧に使用されます。

Restores

Postgres バックアップの復元

次のオプションのいずれかを使用して、復元を実行できます。

- PGAIHM Web UI
- API
- CLI

次の手順を使用して、Postgres クラスターを復元します。

PGAIHM Web UI クラスター復元機能からの復元

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。次のフィールドが使用できます。

Name	Description
Status	バックアップの現在のステータス。
Name	バックアップの名前。
Start Time	バックアップが開始された時刻。
Stop Time	バックアップが停止した時間。
Duration	バックアップが完了するまでにかかった時間。
Location	バックアップの場所。
Method	バックアップが取られた方法。

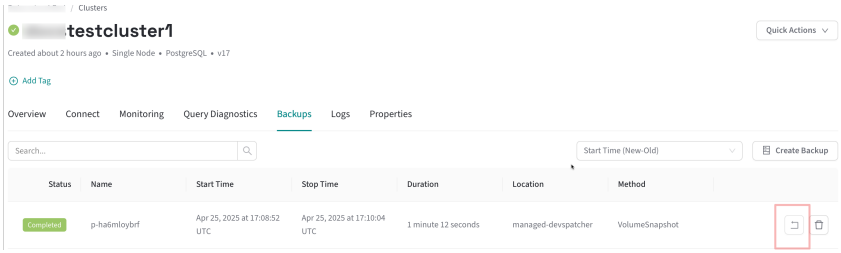


図 15: Viewing Backups

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] 構成を変更せずに完全な復元を行うには、 **Restore Cluster** を選択します。
5. **クラスター設定** で、復元されたクラスターの一意の名前を指定します。
6. **復元タイプ** では、**完全な復元** または**ポイントインタイムの復元** 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、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。
7. **DB Configuration** 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。
8. バックアップからクラスターを復元する準備ができたら、 **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。

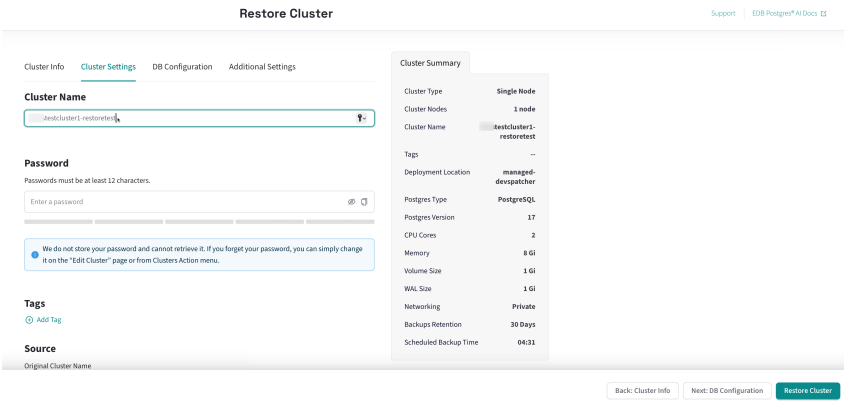


図 16: Restore Cluster

API からの復元

API を使用して復元するには、以下が必要です。

- 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の名前は `pitr_` です。

API を使用してポイント インタイムに復元するには、次の手順を使用します。

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

2. 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

```
$ pitr_cluster.sh
1
2  HCP_API_ACCESS_KEY=baak_                                    ekaR
3  HCP_PROJECT_ID=prj_6E                                       shWH
4
5  curl \
6    --insecure \
7    --header "Content-Type: Application/JSON" \
8    --header "x-access-key: ${HCP_API_ACCESS_KEY}" \
9    --data-binary "@pitr_cluster.json" \
10   --request POST \
11   https://                    .enterprisedb.network/api/v1/projects/prj_6E                    hWH/clusters \
12   | jq
```

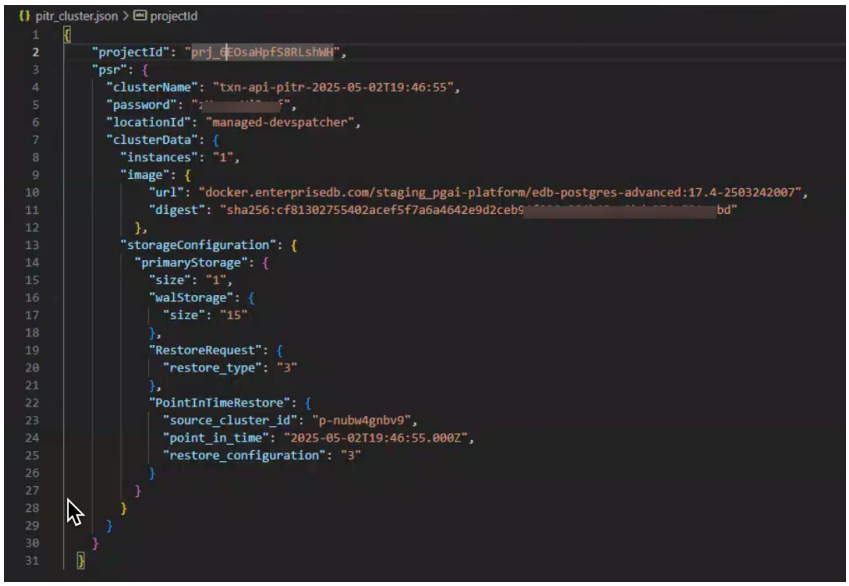


図 17: PGAIHM Restore via API JSON file

3. 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

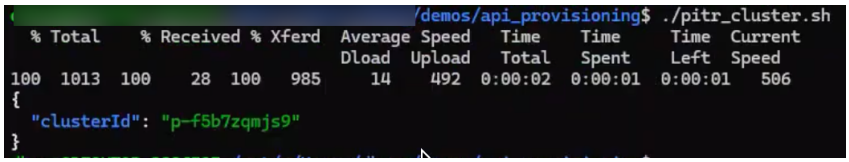


図 18: PGAIHM Restore via API shell file conf

CLI からの復元

CLI を使用してポイント インタイムに復元するには、次の手順を使用します。

CLI で復元するには、以下が必要です。

- 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 ファイルを呼び出します。

1. YAML ファイルで、次のパラメーターを提供します。

> **注釈**
>
> resume コマンドを使用するときに予想されるパラメーターの詳細については、CLI で `edbctl cluster restore -h` を使用してヘルプファイルを表示してください。

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

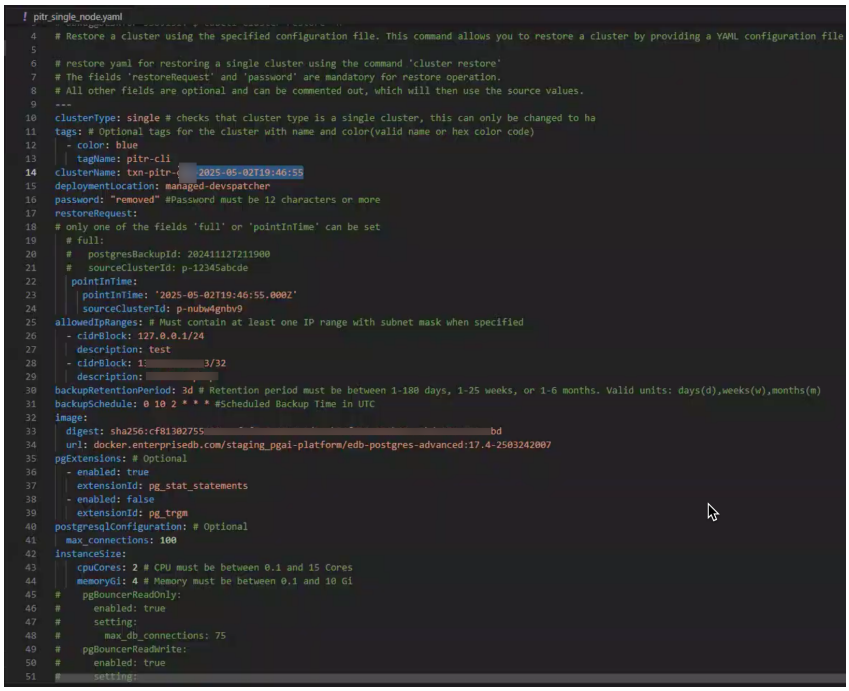


図 19: PGAIHM Restore via CLI YAML file

```
~$ edbctl cluster restore -F $DEMOS/cli_pitr/pitr_single_node.yaml
```

図 20: PGAIHM Restore via CLI YAML file

Using the API

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、クラスターのプロビジョニング、プロビジョニング解除、スケーリング、管理操作の自動化などのアクティビティが含まれます。

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。

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。
- 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。

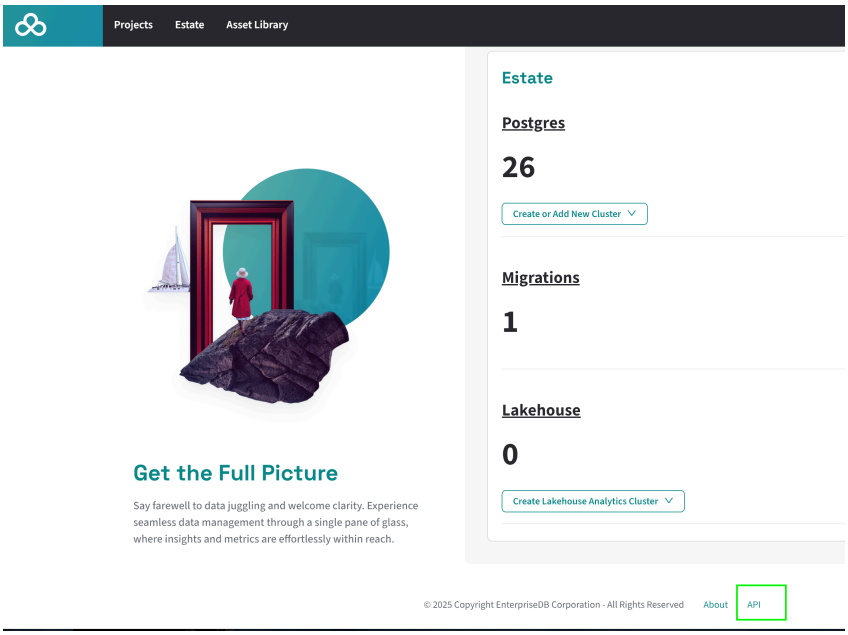


図 21: API Docs

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 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。組織所有者はマシンユーザーのアクセスキーを管理でき、一般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は、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 the *Creating a machine user* . Optionally, can be individually added from **Access keys** tab on machine user's details page using **Create New Key** option.
Normal user	1	One access key, ステップ *3PGFS* ストレージの場所が作成されたことを検証する and managed by user from their home page.

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. ポータルの右上にある組織名の横にあるドロップダウンメニューから、 **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 タブから、 **Create New Key** を選択します。
5. **Access Key Name** を入力します。
6. **有効期限** を 1〜365 の範囲で指定します。
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

ユーザーホームページから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. ポータルの右上にある組織名の横にあるドロップダウンメニューから、**Users Management** を選択します。
2. リスト内のユーザーの **Edit** ボタンを選択します。
3. ユーザーのホームページにある **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。
    1. **Name** — アクセスキーの名前。
    2. **Access Key** — 半分暗号化された形式のアクセスキー。
    3. **Expiry at** — アクセスキーの有効期限。
    4. **作成日** — アクセスキーの作成日。

特定の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** タブの右上隅にある **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。
- ユーザー権限

次に、アクセスを承認します。詳細については、以下を参照してください。

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、HM UI からコピーされ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":
[]
}
]
}
```

5.1.4 For Kubernetes Administrators

Managing users in Hybrid Manager

ハイブリッドマネージャーは、コンソールでのユーザー認証に Dex を使用し、ハイブリッドマネージャーとネイティブユーザーまたは LDAP や SAML のような外部ユーザー管理システム ID プロバイダーまたは IdP 間のプロキシとして機能します。

ハイブリッドマネージャーの初期ブートストラップ中に、管理者は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。

本番展開の場合、LDAP、SAML、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。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する必要があります。

- 詳細については、 Configuring an IdP を参照してください。

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 an external IdP* 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

環境への不正アクセスを回避するために、展開中または展開の直後にデフォルトのネイティブユーザーのパスワードを変更します。同じ方法で手動で追加したネイティブユーザーのパスワードを変更できます。

1. 新しい bcrypt ハッシュ化されたパスワードを作成します。この値を生成するには、次を使用します。

```
echo ${password} | htpasswd -BinC 10 admin | cut -d: -f2
```

新しいパスワードが${password} 変数の後ろに保存される場所。

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$2y$10$YDHMEnc8VZKnhvze538ePOLKol7dFH.S7IgpqC3.X3R/3P2Q9zRcG

1. 既存のユーザーこの例では owner@mycompany.com のハッシュを置き換えます。使用するインストール方法に応じて、values.yaml ファイルまたは HybridControlPlane カスタムリソースのこの値を置き換える必要があります。

```
portal:
  authentication:
    staticPasswords:
      - email: "owner@mycompany.com"
        hash: "$2y$10$YDHMEnc8VZKnhvze538ePOLKol7dFH.S7IgpqC3.X3R/    3P2Q9zRcG"
        username: "owner@mycompany.com"
        userID: "c5998173-a605-449a-a9a5-4a9c33e26df7"
```

1. Hybrid Manager をインストールする場合、インストールの実行を続行します。既に展開されているインストールのパラメーターを変更する場合は、*Applying modifications to the user configuration* を使用して変更を入力します。

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 an external IdP* を強くお勧めします。

現在、コンソールで人間のユーザーを作成することはできませんが、ハイブリッドマネージャー構成ファイルを調整することによってのみです。

ネイティブユーザーの追加

インストール中または後に、新しいユーザーを `portal.authentication.staticPasswords` セクションに追加します。使用するインストール方法に応じて、`values.yaml` ファイルまたは `HybridControlPlane` カスタムリソースのこの値を置き換える必要があります。

```
portal:
  authentication:
    staticPasswords:
      - email: "owner@mycompany.com"
        hash: "$2y$10$STTzuGJa3PydqHvlzi2z6OgDU1G/JLTqiuGblH.RemIutWxkztN5m"
        username: "owner@mycompany.com"
        userID: "c5998173-a605-449a-a9a5-4a9c33e26df7"
      - email: "<userA@example.com>"
        hash: "<example_hashed_password>"
        username: "<userA@example.com>"
        userID: "<unique_user_id>"
```

Applying modifications to the user configuration を使用して変更を入力します。

パラメーター

`portal.authentication.staticPasswords` エントリーのユーザーの設定。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。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 ${password} I htpasswd -BinC 10 userA I cut -d: -f2`を使用します。ここで、実際のパスワードは`${password}`変数の後ろに保存されます。 `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。構成内の他の場所で使用されていないため、任意のテキストにすることができます。結果のハッシュのみが利用されます。
userID	静的パスワードで構成された各新しいユーザーには、ユーザーの一意の識別子 uuid が必要です。 UUID ジェネレータツールでこの値を生成するか、文字のランダムなシーケンスを手動で割り当てることを選択できます。

次のページに続く

表 12 – 前のページからの続き

Parameter	Description
username	ユーザーの一意のユーザー名。これは、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**ロールを持つ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を割り当てます

1. 少なくとも 1 回は組織の所有者になる IdP ユーザーとして Hybrid Manager コンソールにログインします。これにより、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。その後、ログアウトします。
2. ハイブリッドマネージャーの現在のネイティブ所有者として `owner@mycompany.com` など `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を無効にする

1. ネイティブユーザーログインを無効にするには、`portal.authentication.staticPasswords` を `[]` に設定して `staticPasswords` 構成の設定を解除します。

```
portal:
  authentication:
    staticPasswords: []
```

1. *Applying modifications to the user configuration* を再度実行して、変更を入力します。

Configuring your own identity provider

本番展開の場合、LDAP、SAML、または別の supported Dex 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。次にいくつかの例を示します。

- *Configuring LDAP*
- *Configuring SAML*

テストまたはデモ環境の場合、 *Creating native users for testing* による操作を簡素化できます。

Configuring LDAP

Dex 認証と連携するように LDAP を構成して、エンドユーザーが LDAP サーバーの資格情報を使用してコンソールにログインできるようにします。この例では、LDAP 構成に Okta を使用します。

LDAP 統合のための Okta ユーザーの作成

Okta にログインし、読み取り専用の管理者ロールで新しいユーザーを作成します。Dex はこのユーザーを使用して、Okta LDAP インターフェイスと通信します。

Okta は、クライアント統合のための LDAP インターフェイスを提供します。詳細については、*Okta LDAP Interface documentation* を参照してください。

LDAP コネクタを介して Okta に接続するように Dex を構成する

Okta LDAP 構成のエントリーを追加し、作成した読み取り専用の管理者ユーザーのユーザー名とパスワード BindDN および BindPW を指定します。使用するインストール方法に応じて、`values.yaml` ファイルまたは `HybridControlPlane` カスタムリソースにエントリーを追加します。

```
portal:
  authentication:
    clientSecret: "edb-postgres-ai-secret"
    idpConnectors:
      - type: ldap
        name: okta
        id: oktaLdap
        config:
          host: <ldap_server>:<ldap_server_port>
          insecureNoSSL: false
          insecureSkipVerify: false
          bindDN: uid=<okta_admin@example.com>, dc=<org_subdomain>, dc=<domain>,␣
↪dc=com
          bindPW: <okta_admin_password>
          usernamePrompt: Email Address
          userSearch:
            baseDN: ou=users, dc=<org_subdomain>, dc=<domain>, dc=com
            filter: "(objectclass=inetOrgPerson)"
            username: mail
            idAttr: DN
            emailAttr: mail
            nameAttr: cn
          groupSearch:
            baseDN: ou=groups, dc=<org_subdomain>, dc=<domain>, dc=com
            filter: "(objectclass=groupOfUniqueNames)"
            userMatchers:
              - userAttr: memberOf
                groupAttr: uniqueMember
            nameAttr: cn
```

(次のページに続く)

(前のページからの続き)

```
  staticPasswords:
    - [...]
```

パラメーター

このセクションでは、`portal.authentication` 構成内で必要な ID プロバイダー設定について詳しく説明します。環境に基づいてカスタマイズが必要なパラメーターにのみ焦点を当てています。ここで明示的に記載されていない他のパラメーターは、提供された例に示すように構成できます。

Parameter	Description
clientSecret	ハイブリッドマネージャー認証のクライアントシークレットは、デフォルトの `edb-postgres-ai-secret` を使用します。
idpConnectors	認証に使用する IdP コネクタごとにデータの配列を入力します。詳細なガイダンスについては、公式 *Enabling the Migration Portal AI Copilot* を参照してください。
idpConnectors.config	LDAP サーバーへの接続の構成。 `<ldap_server>:<ldap_server_port>` 形式を使用します。
idpConnectors.config.bindDN	Okta で作成した読み取り専用の管理ユーザーの識別名。Dex が LDAP サーバーとの通信に使用します。また、Dex が LDAP サーバー内のこのユーザーを検索する場所も指定します。
idpConnectors.config.bindPW	Okta で作成した読み取り専用の管理ユーザーのパスワード。Dex が LDAP サーバーとの通信に使用します。
idpConnectors.config.usernameProm	ログイン画面のユーザー名/電子メール入力フィールドの上に表示されるテキスト。
idpConnectors.config.userSearch.bas	Dex が LDAP サーバー内のユーザーを検索する場所を指定します。 Okta LDAP インスタンスの管理者は、ユーザーの baseDN を知っています。
idpConnectors.config.groupSearch.ba	Dex が LDAP サーバー内のグループを検索する場所を指定します。 Okta LDAP インスタンスの管理者は、ユーザーの baseDN を知っています。
idpConnectors.config.groupSearch.ba	ユーザーがグループのメンバーであるかどうかを Dex が判断する方法を定義します。ユーザーエントリとグループエントリの間で属性を一致させます。

Dex 構成の適用

認証設定を変更した後、Dex 構成を調整し、Dex サービスを再起動して変更を有効にする必要があります。手順については、*Applying modifications to the user configuration* を参照してください。

新しい所有者を割り当て、ネイティブユーザーを無効にします

Hybrid Manager コンソールで新しい組織の所有者を割り当てた後、ネイティブユーザーを無効にします。これについては、*Disabling native user login* を参照してください。

Configuring SAML

Dex 認証と連携するように SAML を構成して、エンドユーザーが SAML サーバー資格情報を使用してコンソールにログインできるようにします。この例では、SAML 構成に Okta を使用します。

SAML アプリケーションの作成

1. Okta にログインし、新しいアプリ統合を作成します。
2. **SAML 2.0** を選択します。
3. アプリ名とその他の **一般設定** を割り当て、 **Next** を選択します。
4. **シングルサインオン URL** (ACS URL) と **Audience URI** フィールドの両方に Dex コールバック URL を入力します。フォーマットは `https://<hybrid_manager_URL>/auth/callback` です。
5. ユーザー名と電子メールの **属性ステートメント**をマッピングします。例

Name	Name format	Value
name	Unspecified	“user.firstName+” “+user.lastName”
email	Unspecified	user. email

1. アプリケーションが作成されたら、ハイブリッドマネージャーにアクセスするユーザーまたはグループを、作成したアプリ統合に割り当てます。これは、[**Assignments**] タブで行うことができます。

SAML コネクタを介して Okta に接続するように Dex を構成する

通信の Okta および Dex エンドポイントを指定する Okta SAML 構成のエントリーを追加します。使用するインストール方法に応じて、`values.yaml` ファイルまたは `HybridControlPlane` カスタムリソースにエントリーを追加します。

```
portal:
  authentication:
    clientSecret: "edb-postgres-ai-secret"
    idpConnectors:
      - type: saml
        name: Okta
        id: okta
        config:
          ssoURL: <https://saml.example.com/sso>
          redirectURI: <https://<hybrid_manager_URL>/auth/callback>
          usernameAttr: name
          emailAttr: email
```

(次のページに続く)

(前のページからの続き)

```
        groupsAttr: groups
        caData: <dNekV4TURSYUZ3MHpOREV3T01Rc3dDUVlEVlFRR1CRU=>
  staticPasswords:
    - [...]
```

パラメーター

このセクションでは、`portal.authentication` 構成内の重要な ID プロバイダー設定について説明します。環境に合わせてカスタマイズが必要なパラメーターのみに焦点を当てています。ここで詳しく説明されていない他のパラメーターは、提供された例に示すように構成できます。

Parameter	Description
clientSecret	ハイブリッドマネージャー認証のクライアントシークレットは、デフォルトの `edb-postgres-ai-secret` を使用します。
idpConnectors	認証に使用する IdP コネクタごとにデータの配列を入力します。詳細なガイダンスについては、公式 *Enabling the Migration Portal AI Copilot* を参照してください。
idpConnectors.config.ssoURL	シングルサインオン URL は、Dex が認証要求を送信するエンドポイントです。 Okta サインオン構成から取得できます。
idpConnectors.config.redirectURI	ハイブリッドマネージャーの ACS URL。これは、Okta SAML インスタンスが応答を送信するエンドポイントです。 SAML アプリケーションの作成時に指定した Dex コールバック URL と同じです。
idpConnectors.config.usernameAttr	返信のどの属性にユーザーのユーザー名が含まれるかを Dex に指示します。値は、SAML アプリケーションの作成時に指定した属性名と一致する必要があります。
idpConnectors.config.email	返信のどの属性にユーザーのメールアドレスが含まれるかを Dex に指示します。値は、SAML アプリケーションの作成時に指定した属性名と一致する必要があります。
idpConnectors.config.groupsAttr	応答のどの属性に、ユーザーが属しているグループに関する情報が含まれているかを Dex に指示します。
idpConnectors.config.caData	Okta アプリケーションの SAML 構成からの証明書。これにより、ハイブリッドマネージャーは SAML サーバーからの応答を認証できます。 Okta サインオン構成、SAML セットアップオプションから取得できます。

Dex 構成の適用

認証設定を変更した後、Dex 構成を調整し、Dex サービスを再起動して変更を有効にする必要があります。手順については、*Applying modifications to the user configuration* を参照してください。

新しい所有者の割り当てとネイティブユーザーの無効化

Hybrid Manager コンソールで新しい組織の所有者を割り当てた後、ネイティブユーザーを無効にします。これについては、*Disabling native user login* を参照してください。

Applying modifications to the user configuration

ユーザー認証設定を変更した後、Dex 構成を調整し、サービスを再起動して変更を有効にする必要があります。これらの更新を調整する方法は、ハイブリッドマネージャーをインストールした方法によって異なります。Helm チャートを介してまたは EDB Operator を使用して。

Helm チャートのインストール

Helm チャート方法を使用してインストールされたハイブリッドマネージャーインスタンスでは、次のように Dex 変更を適用します。

1. 更新された `values.yaml` で helm チャートを再適用します。GCP インストールでこれを行う方法の例については、*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する** を参照してください。
2. Dex を再起動してリロードし、新しいユーザーを追加します。

```
kubectl rollout restart deploy upm-dex -n upm-dex
```

オペレーターのインストール

オペレーター方法を使用してインストールされたハイブリッドマネージャーインスタンスでは、次のように DEX 変更を適用します。

1. `HybridControlPlane` カスタムリソース CR を再適用します。RHOS インストールでこれを行う方法の例については、*Install Hybrid Manager using Operator* を参照してください。
2. Dex を再起動してリロードし、新しいユーザーを追加します。

```
kubectl rollout restart deploy upm-dex -n upm-dex
```

Monitoring with Hybrid Manager

ハイブリッドマネージャーを使用すると、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、ロギングデータなどを収集できます。収集されたデータをコンソールまたは 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などを行うことができます。

監視アーキテクチャの理解

ハイブリッドマネージャーには、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ハイブリッドマネージャー内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。
- **Console** 、たとえば、 [Estate] タブまたは *Cluster view* 。

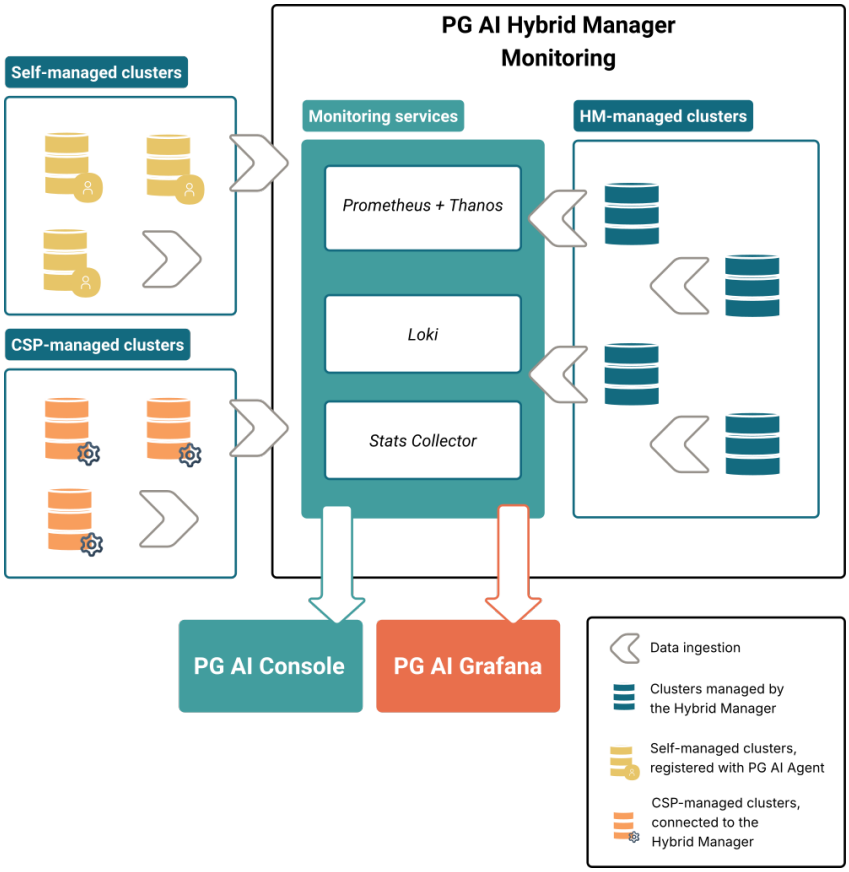


図 22: Monitoring infrastructure in the Hybrid Manager

データベースモニタリングの有効化

ハイブリッドマネージャー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、統計コレクターを有効にするためのいくつかの拡張機能が必要です。

- **HM 管理** クラスター **ハイブリッドマネージャー**で作成の場合、デフォルトのセットアップで監視スタックへのアクセスが既に提供されています。
- 外部 **自己管理**クラスターの場合、各データベースを registered with the **Agent** にしてデータ収集を有効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、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、統計コレクター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 **CSP 管理**データベースクラウドサービスプロバイダー管理の場合、ハイブリッドマネージャーは 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 を使用したデータの照会は、現時点では CSP 管理データベースでは利用できま

せん。

データの取り込みとスコープの監視

ハイブリッドマネージャーはさまざまなクラスタータイプに対するアクセスレベルが異なるため、データの取り込み方法、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 **ハイブリッドマネージャー**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、推奨事項に必要なデータ、および対応するログに自動的にプッシュしますプロメテウス、タノス、ロキ、および統計コレクターストア。
- 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、Stats Collector ストアにプッシュします。
- **AWS RDS** などの外部クラウドサービスプロバイダー管理データベース CSP 管理データベースの場合、ハイブリッドマネージャーはマネージャーからデータベースへの接続を活用してホストをプルし、Postgres メトリックを中央ストアにプルします。Stats Collector を使用したデータの照会は、現時点では CSP 管理データベースでは利用できません。

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

ハイブリッドマネージャー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。

データを外部ツールに接続する

現在、EDB は、ハイブリッドマネージャー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。

関連トピック

- *Enabling monitoring* ☒エージェントをインストールするか、データベースをハイブリッドマネージャーに接続して、外部データベースでのモニタリングを有効にします。
- *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 console ☒ コンソールを使用してデータベースをモニタします。
- *Notifications* ☒ 通知を構成します。

Enabling monitoring

HM 管理のデータベースクラスター

HM 管理データベースクラスターはハイブリッドマネージャーと同じ環境で作成されるため、そ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。

外部データベースの場合

外部データベースは、クラウドサービスプロバイダーまたは自分自身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ハイブリッドマネージャー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 2 種類の 外部データベースをモニタできますか? 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。

例ベアメタル上にある自己管理の Postgres データベース、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データベース 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 、データベースをハイブリッドマネージャーに接続します。

例 Amazon RDS。

!!!note

Oracle データベースをエージェントに登録して *get your Oracle database ready for migration* に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 **PG AI Hybrid Manager** から自己管理の外部 Postgres インスタンスをモニタするには、最初に PG 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 PG AI Hybrid Manager に提供します。

このガイドを完了すると、自己管理の Postgres インスタンスのモニタリング、クエリ診断、および推奨事項が得られます。

1. *Create an OS user to run the Agent*
2. *Prepare your database*
3. マシンユーザーを作成する
4. *Install the PG AI Agent*
5. *Configure the PG AI Agent*
6. *Run the PG AI Agent*

7. *Configure the PG AI Agent to run as a service*

エージェントの構成に必要な管理タスクに関する詳細なヘルプについては、*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* 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 sovereignity* 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、その機能に必要のないエージェントへの権限の付与を回避するため、セキュリティのベストプラクティスです。

新しい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、このガイドの後半で Postgres ログファイルへのエージェントアクセスを許可することが簡単になります。

```
usermod -aG postgres <agent_os_user>
```

Preparing your database for monitoring

Postgres インスタンスの準備

監視ユーザー

PG AI エージェントを構成するときに、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指定する必要があります。

これは、pg_monitor ロールが割り当てられた Postgres ユーザーです。この例では、新しいユーザー agent_user を作成します。

```
CREATE USER agent_user WITH PASSWORD <password>;
GRANT pg_monitor TO agent_user;
```

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自動的に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

PG AI Hybrid Manager の完全なモニタリングと最適化機能を有効にするには、edb_internals と呼ばれるスキーマを作成し、そこに次の拡張機能をインストールする必要があります。これらの拡張機能は、インストルメンテーションを Postgres に追加します。これがないと、**クエリ診断** および **推奨事項**タブ、および一部のメトリックは使用できません。

- *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。

拡張機能をインストールするには

1. ご使用のプラットフォームと 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、上にリンクされているインストール手順を参照してください。

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ないことに注意してください。

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
```

2. 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";
```

3. 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;
```

4. 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;
```

5. 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、 *dedicated OS user* 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。

Oracle インスタンスの準備

モニタリング機能を提供する代わりに、Oracle データベースを PG AI エージェントに登録すると、移行評価およびスキーマ抽出機能が提供されます。

Oracle データベースを評価するには、PG AI エージェントにデータベースへのアクセスと、評価データを取得するために必要な権限を持つ Oracle ユーザーが必要です。

これらの権限は、自動スキーマ抽出とデータベース移行の評価に必要です。これらのロールが割り当てられていない場合でもエージェントは機能しますが、エージェントは削減された機能容量で機能します。

1. 評価の実行に使用する Oracle ユーザーがこれらのリソースにアクセスできることを確認します

```
V$VERSION
V$INSTANCE
V$LICENSE
sys.all_tables
sys.all_tab_privs
```

2. また、次のいずれかの権限が必要です。

```
SELECT_CATALOG_ROLE
SELECT ANY DICTIONARY
```

後で、このユーザーを PG AI エージェント構成ファイルの `provider.onprem.databases.dsn` セクションで指定します。

Creating a machine user

EDB Postgres ® AI ハイブリッドコントロールプレーンコンソールで、ビーコン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の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. オプションユーザーに関連付ける電子メールアドレスを入力します。これは必要ありません。
5. **Create Access Key** チェックボックスをオンにします。フォームが展開され、**アクセスキー名** と **有効期限** 時間を要求します。
6. **Add User** を選択します。
7. アクセスキーが安全であることを確認します。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** ポップアップウィンドウで、 **estate ingester** をチェックします。
7. **Submit** を選択します。

新しいマシンユーザーは、ビーコンエージェントからデータを取り込む準備が整います。

Installing the Agent

ハイブリッドマネージャーの **エステート**ページでデータベースを監視するようにエージェントをインストールして構成します。

前提条件

- `estate ingester` ロールが割り当てられたマシンユーザーとアクセスキーを作成しました。*Creating a machine user* を参照してください。
- データベースをモニタするプロジェクトのプロジェクト ID が必要です。ID を取得するには、プロジェクトに移動し、ブラウザーに表示される URL から ID を取得します。

システムでパッケージをダウンロードできるようにします

エージェントを実行する予定のシステムを有効にして、 EDB Postgres AI の EDB リポジトリからパッケージをダウンロードします。

EnterpriseDB repos download page で EDB サブスクリプショントークンを見つけます。

取得したサブスクリプショントークンを使用して、環境変数 `EDB_SUBSCRIPTION_TOKEN` をトークンの値に設定します。また、EDB サブスクリプションタイプ `EDB_SUBSCRIPTION_TYPE` の環境変数を、サインアップしているプランに応じて `standard` または `enterprise` のいずれかに設定します。

```
export EDB_SUBSCRIPTION_TOKEN=<your-repos-token>
export EDB_SUBSCRIPTION_TYPE=<your-subscription-type>
```

オペレーティングシステムによっては、次のコマンドを実行して、EDB リポジトリからパッケージをダウンロードする容量をシステムに提供します。

RHEL ライクまたは SLES の場合

```
curl -1sSLf "https://downloads.enterprisedb.com/$EDB_SUBSCRIPTION_TOKEN/$EDB_
↪SUBSCRIPTION_TYPE/setup.rpm.sh" | sudo -E bash
```

Debian または Ubuntu の場合

```
curl -1sSLf "https://downloads.enterprisedb.com/$EDB_SUBSCRIPTION_TOKEN/$EDB_
↪SUBSCRIPTION_TYPE/setup.deb.sh" | sudo -E bash
```

Agent パッケージのインストール

EDB のリポジトリから `beacon-agent` パッケージをインストールします。

RHEL のような場合

```
dnf install beacon-agent
```

または、dnf が利用できない場合

```
yum install beacon-agent
```

SLES の場合

```
zypper install beacon-agent
```

Debian または Ubuntu の場合

```
apt install beacon-agent
```

Configuring the Agent

エージェント beacon-agent として展開を構成して、ハイブリッドマネージャーに接続し、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。

エージェントを使用して移行に *get your Oracle database ready for migration* を使用したり、変換のためにスキーマを抽出したりすることもできます。

OS user you previously created に変更して、エージェント構成を実行します。

環境変数を設定する

次の環境変数が環境で使用できることを確認します。これらの環境変数は、エージェントが起動するたびに必要になるため、永続的に利用できる必要があります。

1. BEACON_AGENT_ACCESS_KEY を、*created a machine user* のときにコンソールから取得したアクセスキーに設定します。
2. $DSN を、接続するデータソース名 DSN に設定します。たとえば、パスワード password で認証されたユーザー postgres として、ポート 5432 でデータベース hr に接続するには、次のように指定します。

```
postgresql://postgres:password@localhost:5432/hr
```

このエージェントを使用して複数のデータベースに接続する予定がある場合は、このような変数を複数指定できますたとえば、$DSN1$、DSN2。これらの変数は構成ファイルで名前で参照されるため、任意の名前を使用できます。

構成ファイルの準備

1. ホームディレクトリに Beacon 構成ディレクトリを作成します。

```
mkdir ${HOME}/.beacon
```

!!!注 上記で指定した場所が便利でない場合、次のいずれかも使用します。

- ***/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。 .. ::**

2. 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
```

3. 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するように設計されています。

```
---
agent:
  access_key: $BEACON_AGENT_ACCESS_KEY
  beacon_server: <beacon-server>:9443
  project_id: <project-id>
  providers:
    - onprem
  root_ca_path: ""
general:
  logging:
    level: "info"
  metrics:
    push:
      enabled: true
      include_logs: true
      push_endpoint: <beacon-server>:9443
      root_ca_path: ""
    usage:
      company_code: <company-code>
      output:
        file:
          enabled: true
          path: beacon_usage.json
        http:
          enabled: false
          url: https://pg-usage.enterprisedb.com
provider:
  onprem:
    runner:
      enabled: true
    databases:
      - resource_id: <resource_id>
        disable_host_association: false
        dsn: $DSN
        logs:
          disabled: false
        metrics:
          disabled: false
          stats:
            enabled: true
            wait_states:
              enabled: true
              timeout: 10m
              buckets_count: 5
              bucket_duration: 1m
```

(次のページに続く)

(前のページからの続き)

```
            bucket_offset: 1s
          recommendations:
            enabled: true
            timeout: 10m
            interval: 1m
          query_texts:
            enabled: true
            scraping_timeout: 10m
            scraping_interval: 2m
            scraping_offset: 1s
            eviction_interval: 30m
            cache_items_max: 200
          query_stats:
            enabled: true
            buckets_count: 2
            bucket_duration: 1m0s
    host:
      metrics:
        disabled: false
      resource_id: <host-id>
dispatcher:
  enabled: true
  mode: standalone
  location_id: <location-id>
```

<> で示されるプレースホルダーは、適切な値に置き換える必要があります。各パラメーターの説明および適切な値の取得方法については、パラメーターリファレンス を参照してください。

構成ファイルを検証する

次のコマンドを実行します。これにより、ハイブリッドマネージャーサーバーへのテスト接続を作成し、構成の概要を印刷することにより、前の手順で指定された資格情報を検証します。

```
beacon-agent config
```

接続が成功すると、接続の詳細の概要が印刷されます。出力にエラーが表示された場合、続行する前に解決する必要があります。

注釈

Connectivityチェックが `failed to verify certificate x509...` のようなメッセージで失敗した場合、これはおそらく、エージェントを実行しているシステムがサーバー証明書の署名に使用される証明書を信頼していないことを意味します。これは、ルート証明書の配布に関する問題の症状である可能性が高いため、インフラストラクチャを管理するチームに報告する必要があります。ルート証明書のコピーがある場合は、*agent.root_ca_path* および `general.metrics.push.root_ca_path` を設定することにより、ルート証明書へのパスを提供できます。これらのパラメーターを設定した後、*beacon-agent config* を再実行し

て、問題が解決されたことを確認します。

Connectivity チェックが `connection reset by peer` のようなメッセージで失敗した場合、これは一般に URL が正しくないか、ファイアウォールが接続を防止しているため、エージェントが指定された<beacon-server> に到達できないことを意味します。

エージェントの構成が完了しました。 :ref:`Create an OS user to run the Agent <Create an OS user to run the Agent>` または :ref:`configure the agent to run as a service <Running the Agent as a service>` に進むことができます。エージェントが実行されたら、 *configuring usage reporting* を検討します。

使用状況レポート

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。有効にするまで EDB にデータは送信されません。

エージェントが実行されると、`general.metrics.usage.output.file.enabled` 設定上記のテンプレートで `true` に設定されているため、エージェントは使用状況データのローカルコピーを `beacon_usage.json` ファイルに保存できます。

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。次に、`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。

パラメーターリファレンス

このセクションには、エージェントの構成ファイルを理解および編集するのに役立つ追加情報が含まれています。

YAML agent settings	Placeholder	Guidelines
agent.access_key	N/A	Access key you obtained from the console when you *Enabling the Migration Portal AI Copilot* .
agent.root_ca_path	N/A	If required, 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.
general.metrics.push.root_c	N/A	Set to the same value as *agent.root_ca_path*.
agent.beacon_server	*<beacon-server>:9443*	Obtain the host URL of your beacon server from an administrator, the port is fixed as 9443. See *Administrative tasks* for more information.
agent.beacon_dispatche	*<beacon-server>:9443*	The same as *agent.beacon_server*.
*general.metrics.push.push_*	*<beacon-server>:9443*	The same as *agent.beacon_server*.

次のページに続く

表 16 – 前のページからの続き

YAML agent settings	Placeholder	Guidelines
agent.project_id	<*project-id*>	To obtain your project’s ID, go to the console, select **Projects** , and select your project. The ID is the identifier starting with “prj_” in the URL of that page.
general.metrics.usage.comp	<*company-code*>	Your company code can be found on the EDB Support Portal .
dispatcher.location_id	<*location_id*>	A unique identifier used by the Hybrid Manager for this agent/location. Consider setting this to the same value as *provider.onprem.host.resource_id* for convenience.

データベース構成に関連する設定

YAML database settings	Placeholder	Guidelines
databases.resource_id	<*resource-id*>	Assign an identifier to your database. This will link your database to an identifier. Encase in double quotes. Must be unique within the project.
databases.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.
databases.poll_interval	N/A	Specifies how often the Hybrid Manager checks polls the database for basic metadata. Provide the value in minutes (m) or seconds (s).
databases.tags	N/A	Assign tags to label the migration of that specific database. Encase in double quotes and introduce each tag with a dash.
*databases.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.
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.
host.metrics.disabled	N/A	Set to *true* to disable the collection of host metrics from the host on which the agent is located.

複数のデータベースの監視

リストに項目を追加することにより、複数のインスタンスを指定できます。

```
databases:
      - resource_id: selfmanaged_postgres_1
        disable_host_association: false
        dsn: "$DSN1"
        ...
      - resource_id: selfmanaged_postgres_2
        disable_host_association: true
        dsn: "$DSN2"
        ...
```

注釈

現在の監視データは、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。

監視データ収集の微調整

ここで提供されるテンプレートは、すべての監視機能を有効にします。有効にする機能を微調整できます。

時系列メトリック [Monitoring] タブ

構成ファイルの `general.metrics.push` セクションは、ハイブリッドマネージャーの Prometheus インスタンスへの時系列メトリック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。 `general.metrics.push.enabled` を `false` に設定することにより、これを無効にできます。

Postgres ログ [ログ] タブ

!!!警告 既知の問題 ハイブリッドマネージャーでのログの表示エージェントを使用して収集されたログは、現在 [ログ] タブに正しく表示されません。ただし、それらは正しく保存され、Grafana を介してアクセスするか、[ログ] タブからダウンロードできます。

この問題の部分的な回避策は、`postgresql.conf` で log_destination = ‘stderr’ `を構成することです。これにより、ログコンテンツが UI に表示されますが、改行を含むログエントリは UI に複数のログエントリとして表示されます。

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、次のパターンを使用できます `/var/log/postgres-lc/*.log`

```
databases:
  - resource_id: selfmanaged_postgres_1
  ...
  logs:
    disabled: false
    mode: file
    file:
      paths: <logs_directory>/*log
```

診断 [Query Diagnostics] および [Recommendations] タブ

provider.onprem.databases 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、個々の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。

上記の両方のオプションが無効になっている場合、[**Query Diagnostics**] タブは無効になります。

- 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**] タブが空白になります。
- metrics.stats.query_text は、 EDB 待機状態と EDB Stat Monitor の両方からの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。

Running the Agent

次のコマンドを使用してエージェントを実行します。

```
beacon-agent
```

エージェントは EDB Postgres AI Hybrid Manager 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、ハイブリッドコントロールプレーンを使用した EDB Postgres AI プラットフォームにエージェントからのデータが表示されます。 **Estate** ページに移動して、エージェントからのデータを確認します。

!!!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。

実稼働環境で使用するには、 *configure the agent to run as a service* を使用する必要があります。

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。これにより、システムの起動時に自動的に起動し、エラーの後に再起動できます。

エージェントをサービスとして設定しない場合、手動で実行したとき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 時間に、ハイブリッドマネージャーの**エステート** 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が完了していることを確認します。

!!!note

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、新しい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

2. 次のテンプレートを作成した `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/>
```

(次のページに続く)

(前のページからの続き)

```
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 which 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.

3. 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
```

4. エージェントがサービスとして実行されます。次のコマンドを実行して、ログを確認します。

```
journalctl -u beacon-agent.service
```

Troubleshooting

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、このページはトラブルシューティングに役立ちます。隔離された方法でエージェントの機能をテストして、問題の原因を絞り込みます。

エージェントから HM への接続のテスト

エージェントとハイブリッドマネージャーの間の接続があることを確認するには、エージェントを実行します。

```
beacon-agent setup
```

出力にエラーが表示されない場合、ハイブリッドマネージャーへの接続は正常に機能しており、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、ハイブリッドマネージャーへの接続が適切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、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。

Administrative tasks

エージェントを構成するには、ユーザーは自分自身では取得できないいくつかのリソースへのアクセスが必要です。ただし、管理者は、ハイブリッドマネージャーのインストール中または後に証明書とホスト関連の設定を構成する権利と権限があるため、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、upm-beacon.server_host の下の values.yaml ファイルで構成された URL です。ポートは 9443 に固定されています。

ルート証明書 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、ビーコン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。

何らかの理由でこれに当てはまらない場合、ユーザーは beacon_agent.yaml ファイルに root_ca_path 値を設定できます。これは、ユーザーがルート証明書、 EDB Postgres AI Platform サーバー証明書を検証できる証明書を保存するディレクトリです。ただし、管理者以外のユーザーは証明書にアクセスできないため、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 edbpgai-bootstrap beaconator-ca-bundle -o json
```

ルート証明書のみを印刷するより読みやすい出力を得るには、 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 edbpgai-bootstrap 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必要な人に送信します。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 sovereignity

ハイブリッドマネージャーで外部データベースをモニタするように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

- *Connecting to the Hybrid Manager Console* および *Using Grafana dashboards* 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、**エステート > 移行** 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、移行ポータルが最終的に取得して使用するスキーマ DDL。

!!!note

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。後続のリリースでは、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。ハイブリッドマネージャーは、データの主権を尊重します。たとえば、ハイブリッドマネージャーを自社の敷地内に展開している場合、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。Hybrid Manager 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内に安全に存在します。

Monitoring Cloud Service Provider managed Postgres deployments

クラウドサービスプロバイダー管理の CSP 管理環境でハイブリッドマネージャーの外部で実行されている PostgreSQL データベースは、完全な不動産表現サポートを使用して簡単に監視できます。

現在、次の外部 CSP のモニタリングをサポートしています。

- *AWS RDS Setup*

Monitoring AWS RDS deployments on AWS

EDB AI Hybrid ManagerHM 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。

これは、必要な特権が少ない 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 beacon-csp-credentials に保存されているユーザー資格情報に、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"
            ],
            "Resource": [
                "arn:aws:rds: *:* :db:*"
            ]
        },
        {
            "Sid": "AllowMaintenanceDescriptions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribePendingMaintenanceActions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowGettingCloudWatchMetrics",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData"
            ],
            "Resource": "*"
```

(次のページに続く)

(前のページからの続き)

```
        },
        {
            "Sid": "AllowRDSUsageDescriptions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeAccountAttributes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowQuotaDescriptions",
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowInstanceTypeDescriptions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowInstanceFilterByTags",
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*"
        }
    ]
}
```

ハイブリッドマネージャーで AWS RDS 展開をモニタするためのエージェントの設定

ハイブリッドマネージャー 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するプロセスには、次の手順が含まれます。

> **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、複数のリソースがエステート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どうかです。

1. beacon-csp-credentials シークレットを追加します。

HM の Kubernetes クラスターの upm-beacon 名前空間に beacon-csp-credentials シークレットを追加します。

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
name: beacon-csp-credentials
namespace: upm-beacon
stringData:
AWS_ACCESS_KEY_ID: replace_with_your_access_key_id
AWS_SECRET_ACCESS_KEY: replace_with_your_secret_access_key
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、合計 5 つのメトリックが表示されます。

- 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

ハイブリッドマネージャーでの AWS RDS モニタリングの無効化

次の手順を使用して、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 rollout restart deployment upm-beacon-agent-k8s
```

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
```

エステートに 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 with the PG AI Console

PG AI コンソールを介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が、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の詳細については、こちらをご覧ください。
- *Metrics latency* —メトリックが更新される頻度の詳細については、こちらをご覧ください。

Support scope

PG AI Hybrid Manager 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。

PG AI コンソールで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。このページでは、サポートスコープの概要を提供し、PG AI コンソールでメトリックの各グループを見つけられる場所を説明します。

概要の概要

次に、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の概要を示します。

	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	*Enabling the Migration Portal AI Copilot*	*Enabling Data Catalogs on Hybrid Manager*
HM-managed cluster	yes	yes	yes	yes	yes	yes
External: **Self-managed cluster**	yes	partial support	no	yes	yes	yes
External: **CNPG database cluster**	no	partial support	partial support	yes	yes	yes
External: **AWS RDS database**	yes	partial support	no	no	no	no

> **注釈**
>
> 自己管理のマルチノードインスタンスは、現時点ではグループとして表示されません。各データノードインスタンスとそのメトリックは個別に表示されます。マルチノードインスタンスのクラスター化

ビューは、次のリリースに追加されます。現在、ハイブリッドマネージャーは、EDB CloudNativePG クラスターのモニタリングをサポートしています。オープンソースの CloudNativePG クラスターのモニタリングは、次のリリースでサポートされる予定です。

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。これらは **Estate** ビュー、およびクラスタービュー☒ ***Monitoring*** タブ☒ **Host** セクションから表示されます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Memory	yes	yes	no	yes
CPU	yes	yes	no	yes
Storage	yes	no	no	yes
Disk IOPS	yes	yes	no	yes
Disk throughput	yes	yes	no	yes
Network activity	yes	yes	no	yes

!!!important - 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、- *Using Grafana dashboards* 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自分自身に固有のパフォーマンスと操作統計を示します。これらは **Estate** ビュー、およびクラスタービュー ☒***Monitoring*** タブから表示されます。

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	no	yes	no
Nr. of blocked backends	yes	yes	yes	no
Transactions				
Tuples in	yes	yes	yes	yes
Tuples out	yes	yes	yes	yes

次のページに続く

表 22 – 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**
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
Disk usage	yes	no	no	no
WAL size	yes	no	no	yes
WAL usage	yes	no	no	no
Live/Dead tuples	yes	no	no	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。これらは **Estate** ビュー、およびクラスタービュー☒ ***Monitoring*** タブ☒ **Topology** および **Replication** セクションを介して表示されます。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	no	yes	no
Clock skew	yes	no	yes	no
Node status	yes	no	yes	no
Transaction rate	yes	no	yes	no
Connections (total)	yes	no	yes	no
Connections (per node)	yes	no	yes	no

次のページに続く

表 23 – 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**
Memory (per node)	yes	no	yes	no
Storage (per node)	yes	no	yes	no
WAL size (per node)	yes	no	no	no
Replication status	yes	no	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。これらは、クラスタービュー☒ ***Query diagnostics*** タブから表示されます。

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。これらは、クラスタービュー☒ **推奨事項** タブから表示されます。

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	no	no	no	no
Security	no	no	no	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。これらは、クラスタービュー☒ **Logs** タブから表示されます。

Metric	HM-managed cluster	External: **Self-managed cluster**	External: **CNPG database cluster**	External: **AWS RDS database**
Logs	yes	yes	yes	no

Metrics latency

PG AI Hybrid Manager は、幅広い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

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、ハイブリッドマネージャー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 管理か外部データベースかにかかわらず、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、ハイブリッドマネージャーの中央データストアに保存されます。

PG AI Console の更新 これは 2 番目のステップであり、PG AI Console は中央のデータストアを呼び出して、最近収集されたデータを取得して表示します。

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ハイブリッドマネージャーの内部サービスは、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。

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、*PG 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。

PG AI Console の更新 PG AI Console 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

!!!note

(*) 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。

Grafana

Grafana を使用すると、PG AI Hybrid Manager 展開、HM 管理データベースクラスター、およびハイブリッドマネージャーに接続し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 での説明に従って、最初に PG 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 —すぐに使えるフルサポート。
- 自己管理クラスター —クラスターが PG 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 PG 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

ハイブリッドマネージャー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 名前空間に同等またはポッド名で 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 *Loki* **ポッドのクラッシュ** を認識する方法を理解します。

関連トピック

- *Using Grafana dashboards*
- Grafana's Explore

Using Grafana dashboards

PG AI Hybrid Manager に同梱されている Grafana サービスには、さまざまな Hybrid Manager リソースをモニタするように設計された多数の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

PG AI Hybrid Manager 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多数の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*

!!!note

ダッシュボードの説明では、Hybrid Manager の Grafana に含まれるデフォルトのダッシュボードの一般的および例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ること、および/または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、ハイブリッドマネージャーのアップグレード、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。

ダッシュボードを作成、使用、および管理する

ダッシュボードの作成、使用、管理の方法の詳細については、公式 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、PG AI Hybrid Manager を駆動する基になる 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

!!!note

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、このダッシュボードには、オンプレミスの PG AI Hybrid Manager 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 / Compute Resources / *

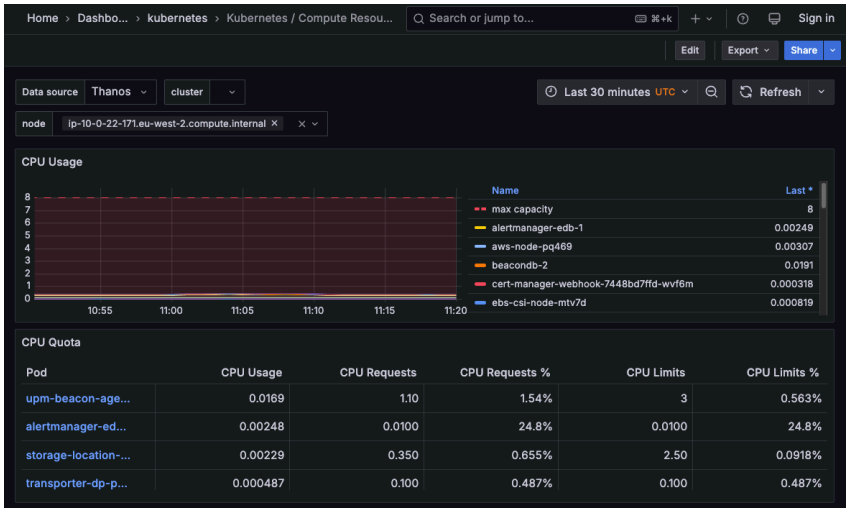


図 23: Compute Resources by Node and Pods

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、さまざまな Kubernetes レベルに焦点を当てて、異なる方法でデータを表示し、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を使用して、すべてのワークロードのリソース消費を理解し、リソースを大量に使用するアプリケーションまたはチームを特定し、アプリケーション内の特定のワークロード、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し、特定のアプリケーション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。インスタンス、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します。

Kubernetes / Kubelet

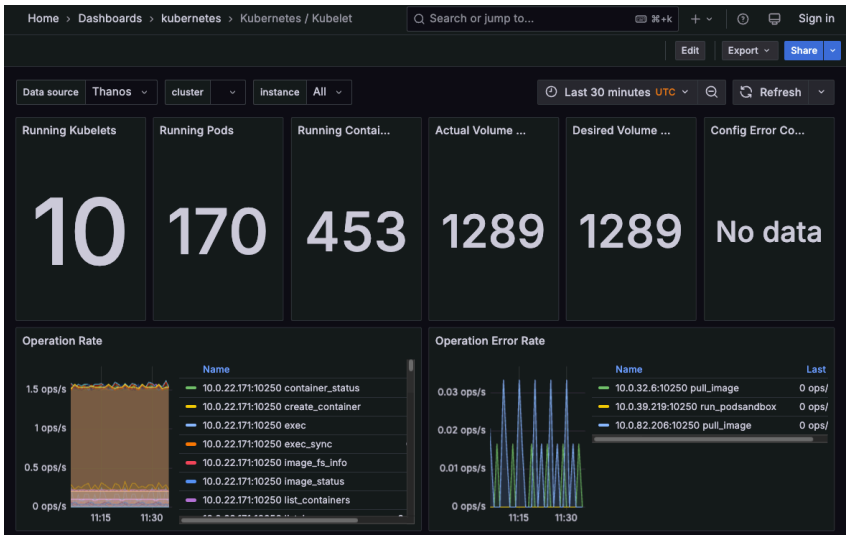


図 24: Kubelet dashboard

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、カウンタナー数、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、レイテンシーを特定し、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 / *

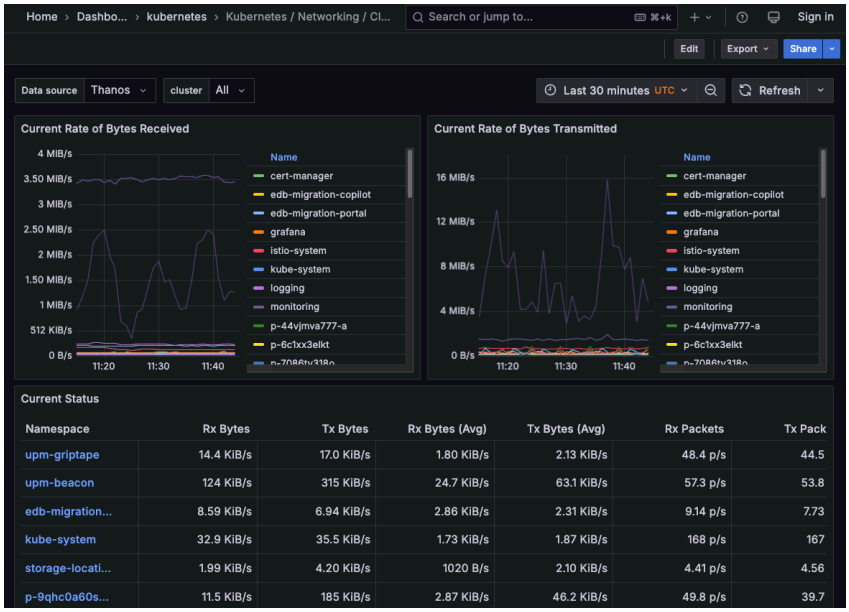


図 25: Networking overall cluster dashboard

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、その名前空間内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、さまざまな Kubernetes レベルに焦点を当てて、異なる方法でデータを表示し、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を使用して、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し、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

し、アプリケーション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し、アプリケーション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を行います。アプリケーションを監視したり、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したりできます。

Kubernetes / 永続ボリューム

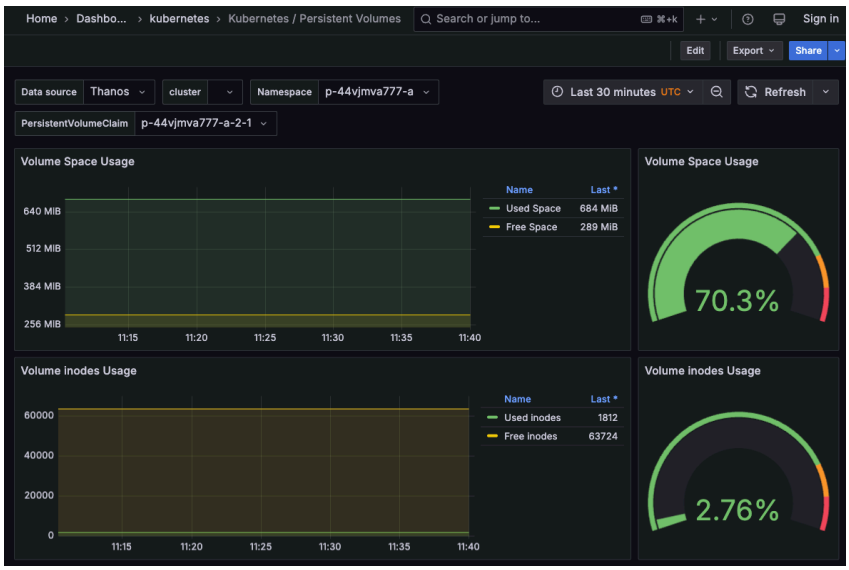


図 26: Persistent Volumes dashboard

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、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、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たかどうかを追跡します。

Kubernetes CoreDNS

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 クラスター内の 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、クエリタイプ別、DNS ゾーン別、サイズ別、応答コード別、応答期間別、応答サイズ別などが含まれます。**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。

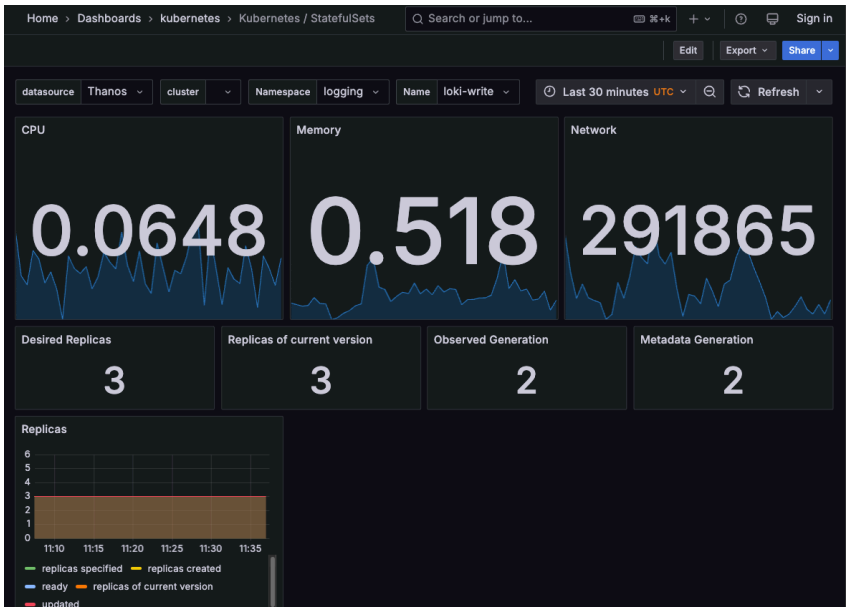


図 27: StatefulSets dashboard



図 28: CoreDNS dashboard

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。`logging` 名前空間は 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 29: Loki Chunks dashboard

説明 Loki 内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 日に設定されています。

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、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、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 30: Loki Operational dashboard

説明 Loki システム自分自身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 内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、およびポッドごとのレイテンシー。

使用法 ログクエリに対する Loki の応答性を評価できます。

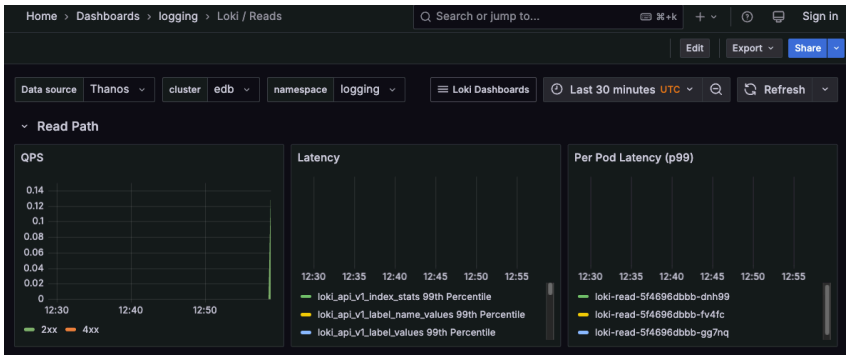


図 31: Loki Reads dashboard

Loki / Reads リソース

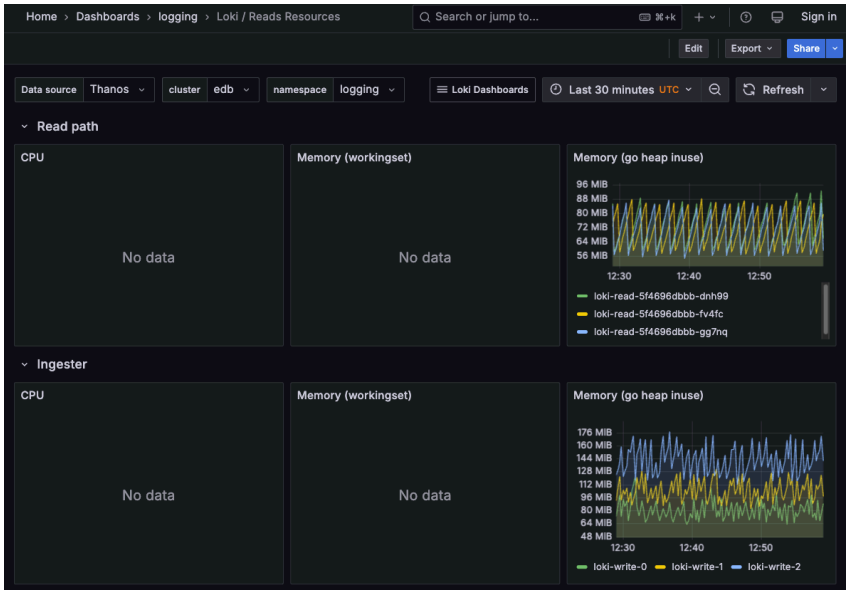


図 32: Loki Reads Resources dashboard

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 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によって使用され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 内のストレージ消費への影響を理解できます。

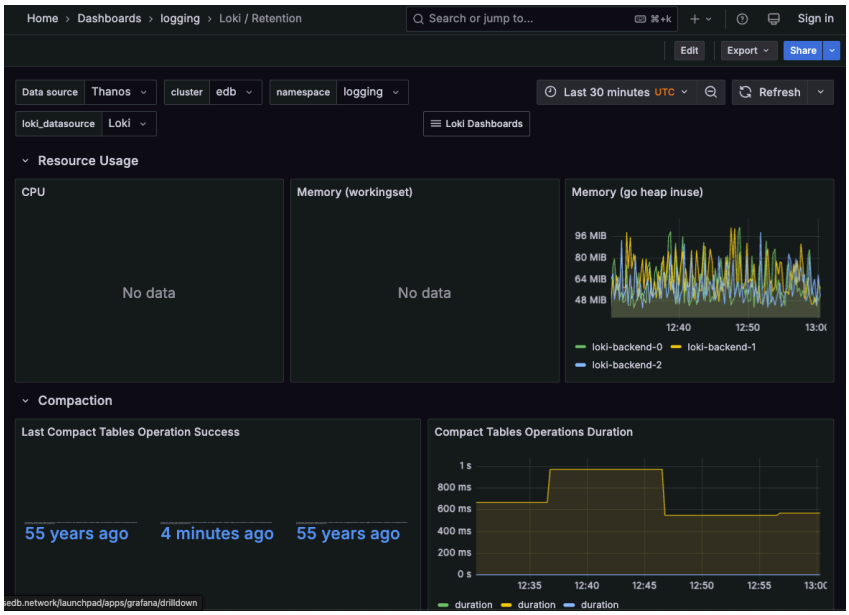


図 33: Loki Retention dashboard

Loki / Writes



図 34: 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、書き込み操作ごと、およびポッドごとのレイテンシ。

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

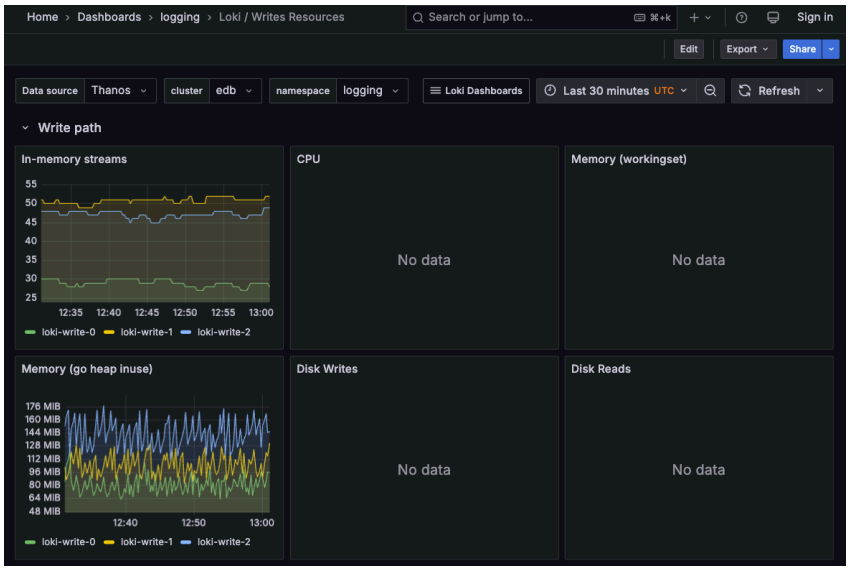


図 35: Loki Writes Resources dashboard

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 36: 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、PG AI Hybrid Manager 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

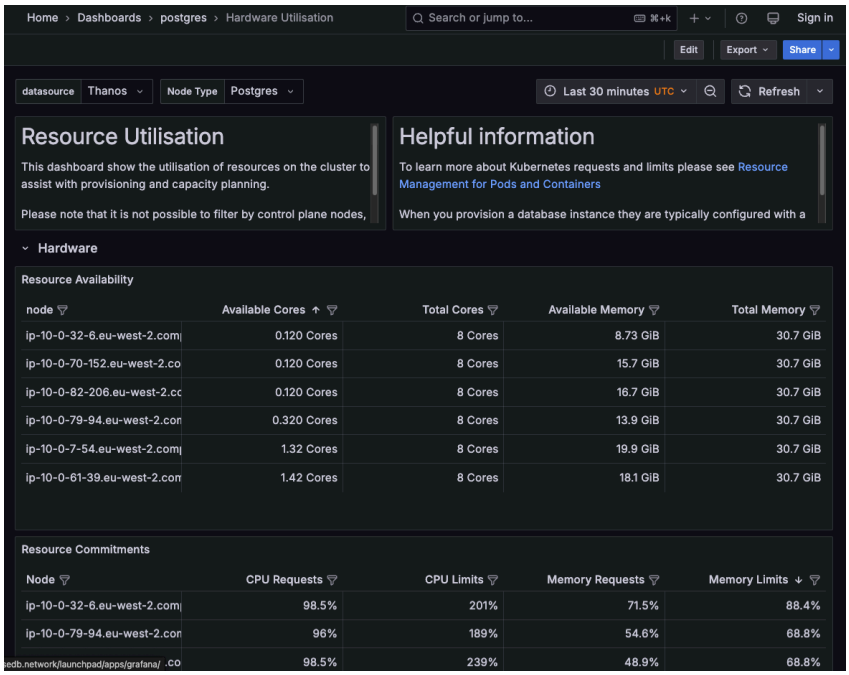


図 37: Postgres Hardware Utilisation dashboard

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ます。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るため、たとえば、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場合があります。

メトリック YAML 形式のイベントのタイプポッドの作成、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

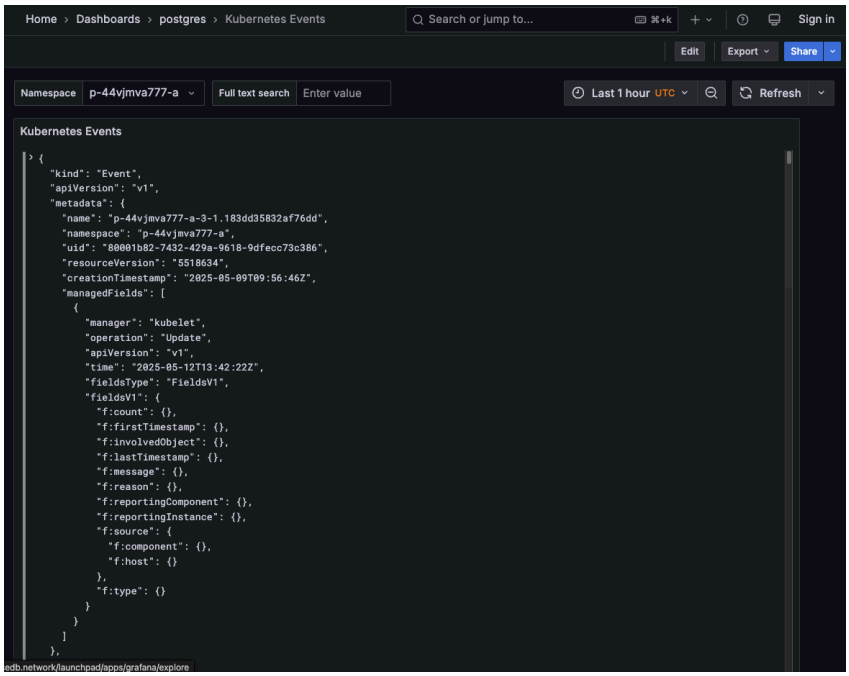


図 38: Postgres Kubernetes Events dashboard

の問題の診断に役立ちます。

PGD の概要

分散 Hidh 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、経過時間、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、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。

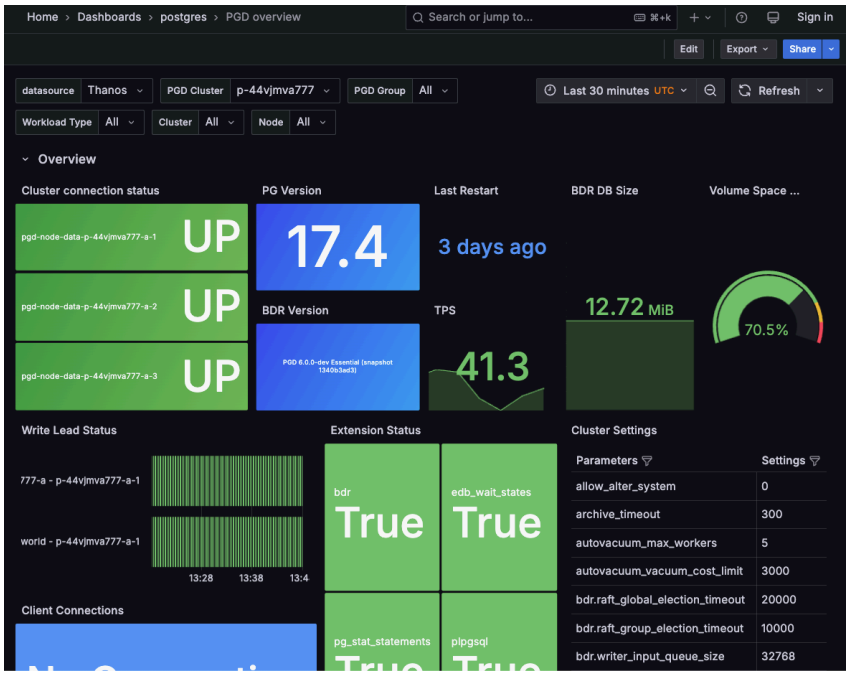


図 39: Postgres PGD Overview dashboard

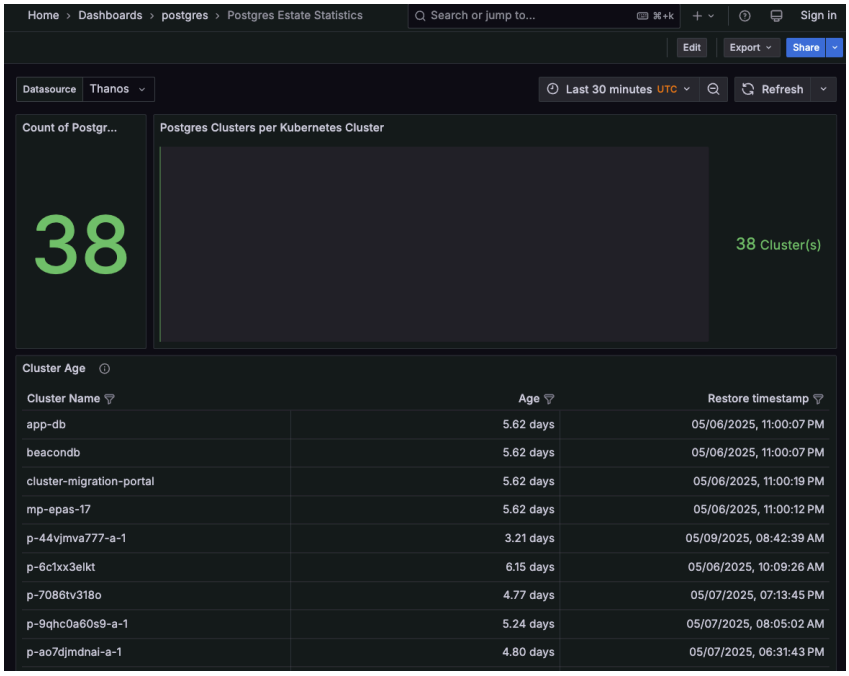


図 40: Postgres Estate Statistics dashboard

Postgres ステータス

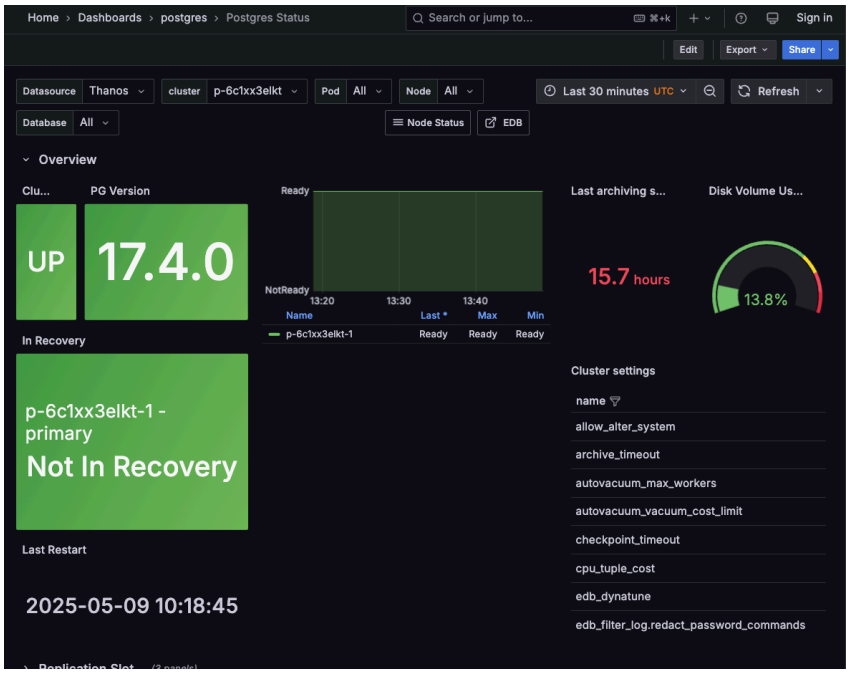


図 41: Postgres Status dashboard

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、WALfiling 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、PG AI Hybrid Manager プラットフォーム自分自身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 FluentBit インスタンス自分自身に関するメトリックを提供します。ノードからログを収集し、Loki に送信するために使用されます。Loki は、PG AI 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 PG AI Hybrid Manager コンポーネントからログの収集と送信中に、FluentBit がパフォーマンスボトルネック、リソースの枯渇、または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 Grafana インスタンス自分自身に関するメトリックを提供します。さまざまなソースからメトリックを収集し、単一の場所に表示するプラットフォーム。

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を表示するために使用している 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 自分自身がオーバーロードされているか、問題が発生しているかどうかを特定します。

Memcached

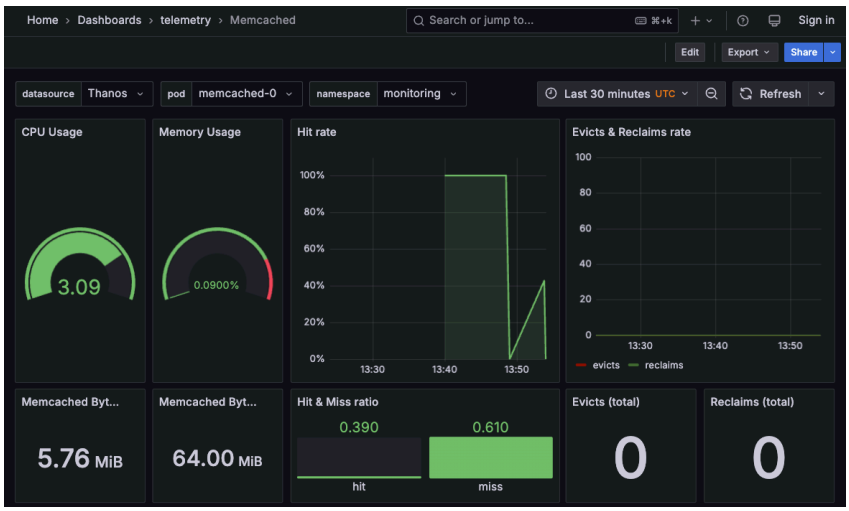


図 42: Telemetry Memcached dashboard

説明 PG AI Hybrid Manager インフラストラクチャ内の 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 とメモリの使用率、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 メソッド/クラスター** 、および **USE メソッド/ノード**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。

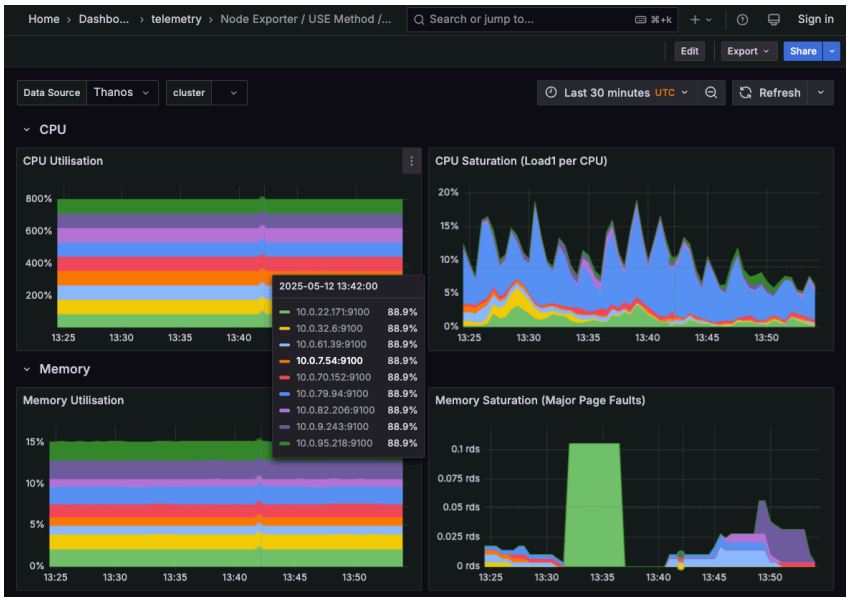


図 43: Telemetry Node Exporter dashboards

- **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。

説明 PG AI Hybrid Manager 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 PG AI Hybrid Manager をホスト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

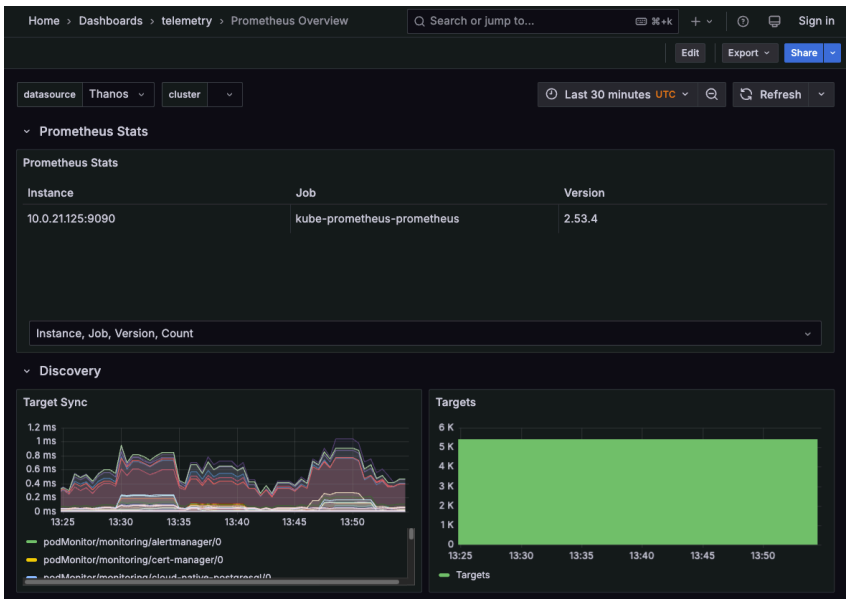


図 44: Telemetry Prometheus dashboard

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 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。これは、PG AI Hybrid Manager 内でメトリックを収集および出荷するために使用されます。

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、クエリ率など。

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 / *

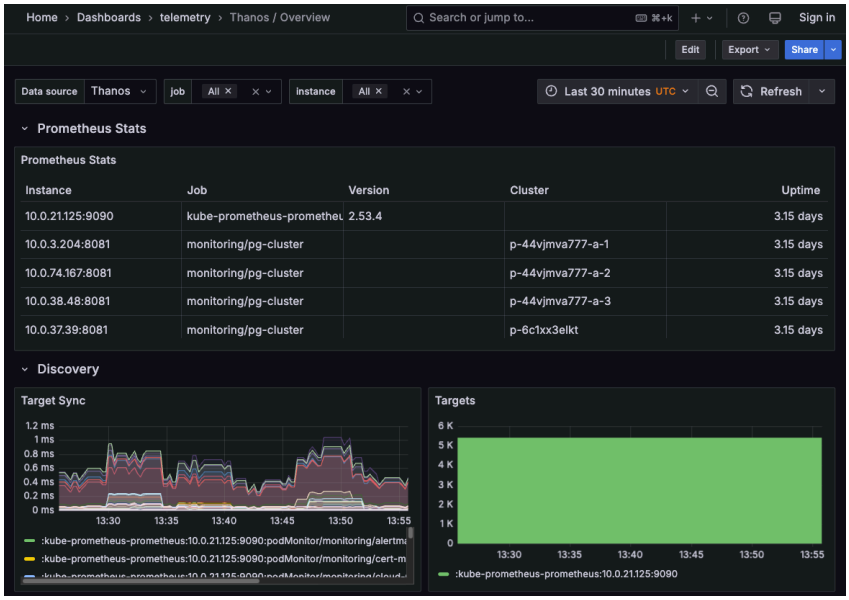


図 45: Telemetry Thanos dashboards

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、PG AI ハイブリッドマネージャー内の 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、コンパクションに関するメトリック、クエリー、着信要求とストアプロシージャ、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。

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

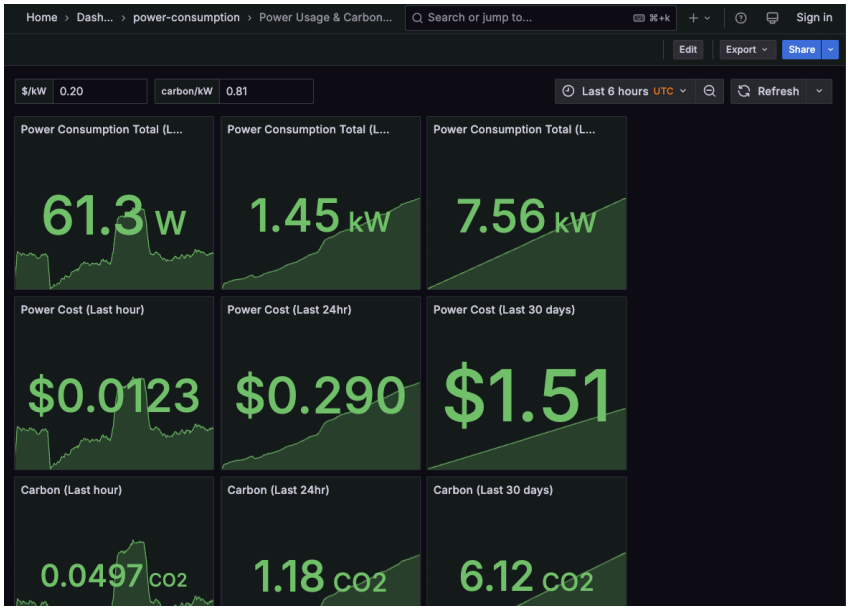


図 46: Power usage and carbon footprint dashboard

説明 PG AI Hybrid Manager インフラストラクチャ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、PG AI Hybrid Manager 内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 PG AI Hybrid Manager 内の TLS 証明書が適切に管理され、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があることを確認するのに役立ちます。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。

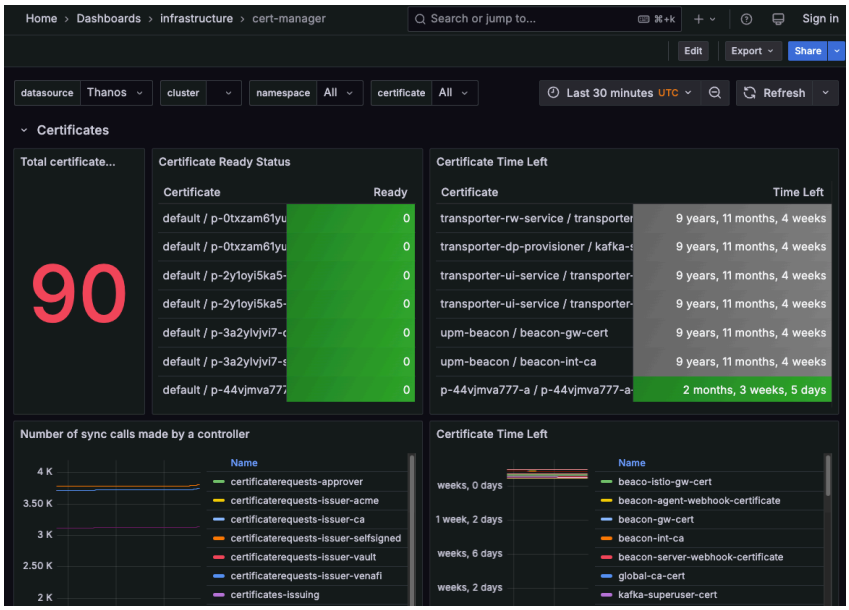


図 47: Cert-manager dashboard

インフラストラクチャ Velero の統計

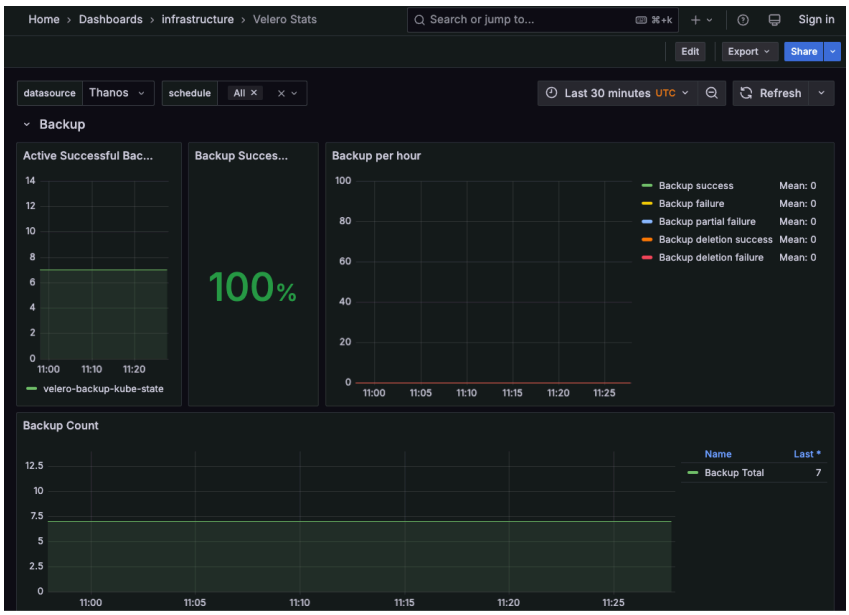


図 48: Velero dashboard

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 内で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 PG AI Hybrid Manager 内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

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、 `cnp_instance_id` を選択します
2. **Label filters** ドロップダウンで、 **cnp_instance_id** を選択します。
3. **Select value** ドロップダウンで、ポッド 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を作成します。

Notifications

EDB Postgres AI 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 チャンネル—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、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 チャンネル—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 回トリガーされるように設計されています。

ハイブリッドマネージャー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のみ
- グローバル —ユーザーとシステムの両方の通知チャネル
- プロジェクト —ユーザーとシステムの両方の通知チャネル

[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

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。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。

これらの手順は、組織レベルでシステム通知チャネルを構成する方法を示しています。

グローバルレベルで通知を設定する

グローバルレベルの通知は、組織の所有者のみが設定できます。

Webhook:

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。
3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Webhook を選択し、ポップアップウィンドウに詳細を入力します。

i. Webhook のホスト URL

ii. Webhook の方法 — POST または PUT

iii. 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
    1. ユーザー名
    2. パスワード
- ベアラートークンの場合
    1. ベアラートークン

iv. [保存] ボタンを選択します。

ポケベルデューティ

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。

3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Pager Duty を選択し、ポップアップウィンドウに詳細を入力します。
    1. ルーティングキー統合キーを設定します。
    2. Save ボタンを選択します。

!!!note

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 アカウントにアクセスする必要があります。 - Pager Duty サービスは、 Events API v2 統合を使用して作成および構成されました。 - 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。- 新しい「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。

メールアドレス

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[設定] を選択します。
3. [通知] タブを選択します。使用可能なオプションは、Webhook、電子メール、およびポケットベルデューティです。
4. Email を選択し、ポップアップウィンドウに詳細を入力します。
    1. Email To —オプションで、システムメール通知メールグループ
    2. Email From —メールを送信しているメールの場合
    3. Host —電子メールサービスの URL
    4. ポート —電子メールサービスのポート
    5. ユーザー名 —メールサービスのユーザー名
    6. パスワード —電子メールサービスの
    7. [Save] ボタンを選択します。

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。

EDB Postgres AI コンソールにログインします

2. 上部の [Projects] タブを選択します
3. 任意のプロジェクトを選択します
4. [プロジェクトの概要] ページに移動します
5. 左側のメニューで [設定] を選択します

6. [通知] を選択します

7. 各システム通知チャネルのグローバル通知設定に記載されている手順に従います。

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、 EDB Postgres AI で割り当てられた複数のロールで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。

EDB Postgres AI コンソールの上部にあるベルアイコンを選択して、アプリ内通知を表示します。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

通知設定を管理するには

1. EDB Postgres AI コンソールにログインします。
2. 右上パネルの自分の名前の下のメニューから、[My Account] を選択します。
3. [通知] タブを選択します。通知は、アカウント、グローバル、および使用可能なプロジェクトごとにグループ化されています。
4. 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。
5. 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。
6. 各通知では、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、特定のイベントの通知の受信をオプトアウトまたはオプトインできます。- 電子メールとアプリ内用のチェックボックスを使用して、イベントの電子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、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ハイブリッドマネージャー HM オブザーバビリティ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] タブの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。

1. HM クラスターの場合
    - コンソールに移動し、上部で Projects
    - リストされたプロジェクトを選択します b_tran_5 左側のナビゲーションで、 Clusters
    - に移動します 準備ができているクラスターを選択します b_tran_5 クラスターの詳細ページで、[モニタリング] タブを選択します
2. 外部 HM クラスターの場合
    - コンソールに移動し、上部で Estate を選択します
    - すべてのクラスターをリストするテーブルに移動します s
    - 管理が自己管理としての準備ができているクラスターを選択します
    - クラスターの詳細ページで、[モニタリング] タブを選択します

[Monitoring] タブには、 Active Alerts リストが表で表示されます。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
- Sort ユーティリティ、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ことができます
- Refresh ボタン、アラートリストを更新します
- Column Selector ドロップダウン、設定に応じてテーブルに表示する列を選択できます
- Download ボタン、アラートリストを CSV 形式でダウンロードします

Refresh ボタンを使用して、アラートを手動で更新できます。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、[モニタリング] タブのフィルタが自動的に適用され、関連するグラフが表示されます。

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

アラートはプロジェクトレベルで設定できます。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. EDB Postgres AI Console にサインインします。
2. コンソールの上部で、[プロジェクト] を選択します
3. リストからプロジェクトを選択します
4. 左側のナビゲーションで、[設定] に移動します
5. 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。

- Enable/Disable —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

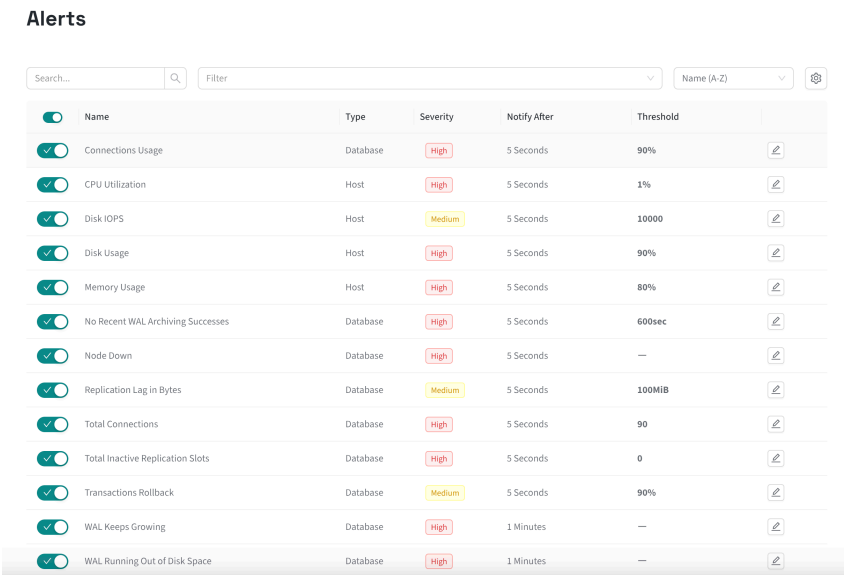


図 49: Alerts

データベースレベルで設定できるアラート

- 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

ホストレベルで設定できるアラート

- CPU 使用率
- ディスク IOPS
- ディスクの使用量
- メモリ使用量

!!!note

- 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。 - 外部 HM クラスターを利用できるようにするための技術的な制限のため、すべてのアラートは外部 HM クラスターでは利用できません。

PGAIHM High Availability & Disaster Recovery (HA/DR)

PGAIHM クラスターの HA / DR、ベストプラクティスなどについて学びます。

- *HA/DR planning and best practices*
- *PGAIHM Backup and Recovery*

HA/DR planning and best practices

概要

PostgreSQL 分散 PGD 展開を使用してハイブリッドマネージャー PGAIHM を使用してマルチデータセンタートポロジを構成し、高可用性およびディザスターリカバリー HA/DR 要件を満たすことができます。このトポロジには、プライマリ PGAIHM と、複数の Kubernetes クラスターにわたる 1 つ以上のデータ専用 PGAIHM が含まれます。

この HA 環境では、プライマリ PGAIHM は標準の PGAIHM 展開です。データのみの PGAIHM は、本質的に同じ PGAIHM 展開ですが、プライマリ PGAIHM に接続し、誤ってローカルプロビジョニングを防ぐようにバックエンドを調整します。この接続を促進するキーコンポーネントは、PGAIHM 間の通信と証明書の受け渡しを処理するビーコンエージェントとビーコンサーバーです。

1 つのクラスターがダウンした場合、アプリケーションは同期されたデータを含む別のクラスターを指すことができ、継続的なオペレーションを保証します。マルチデータセンター設定は、データセンターが停止

した場合の冗長性を提供します。中央の PGAIHM は、複数のリモート Kubernetes クラスター全体でデータベースを管理でき、データベース管理を簡素化します。

トポロジ

マルチ PGAIHM セットアップは、次のトポロジで展開できます。

- **プライマリおよびデータのみの PGAIHM** これは、1 つのプライマリ PGAIHM と少なくとも 1 つのデータのみの PGAIHM を使用した基本的なセットアップです。これにより、2 つ以上の Kubernetes クラスターにまたがる PGD 展開が可能になります。
- **複数のデータ専用 PGAIHM** システムは複数のデータ専用 PGAIHM をサポートできるため、単一のプライマリ PGAIHM で多くの Kubernetes クラスターにわたってデータベースを管理できます。これは、それぞれが独自のクラスターを持つ多数のアプリケーションチームを持つ組織に役立ちます。
- **PGD 用の 3 つの PGAIHM** 最適な PGD 展開、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-PGAIHM トポロジをお勧めします。これにより、適切な分散コンセンサスのための奇数クォーラムが保証されます。

マルチデータセンター展開には、メイン PGAIHM インスタンスを含むプライマリデータセンターと、データ専用の PGAIHM または他の Kubernetes クラスターを含む可能性のあるセカンダリデータセンター、およびマルチプルのデータセンターからアクセスできるグローバルオブジェクトストアが含まれます。

> **注釈**
>
> PGAIHM バージョン 1.2 は、単一の PGAIHM のみをサポートします。 将来のバージョンでは、マルチプルの PGAIHM を使用した高度なトポロジが計画されています。

> **注釈**
>
> クロスデータセンター HA データセンターまたはリージョンを超えた高可用性は、現在サポートされていません。バックアップとリストアを介したディザスターリカバリーが唯一のオプションです。

要件

HA トポロジを構成するときは、次の要件が適用されます。

- 2 つ以上の既存の Kubernetes クラスターたとえば、Amazon EKS または RedHat OpenShift を使用します。
- Kubernetes クラスターのいずれかに完全に展開されたプライマリ PGAIHM。
- 提供された構成用スクリプトを使用して、他の Kubernetes クラスターにインストールされたデータのみの PGAIHM。
- データベースを複数のクラスターにまたがる場合、PGD を使用する必要があります。
- 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。
- クラウドおよびマルチリージョン展開の高性能ネットワーキングレイヤー。EKS は、AWS エコシステムとの統合により、これを簡素化します。OpenShift では、より複雑なネットワーク構成が必要に

なる場合があります。複雑なネットワーク構成、特に OpenShift では、クラスター間の接続の問題が発生する可能性があります。適切なネットワーク設定を確認して、PGD ノード間の通信を許可します。raft 設定が適切に調整されていない場合、ネットワーク遅延が高いと PGD クラスターが不安定になる可能性があります。症状には、障害の検出と復旧の遅延が含まれる場合があります。raft 関連の設定を調整して、ネットワーク遅延の増加に対応します。

注釈

単一のスタンドアロンまたはレプリカクラスター非 PGD の場合、クラスターは 1 つの場所内で完全にのみデプロイできます。場所をまたがるレプリカのスパンはサポートされていません。

ネットワーク遅延

PGD がコンセンサス問題にどれだけ早く反応できるかと、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、PGAIHM ユーザーインターフェイス UI 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。高遅延の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、構成設定、特に UI で公開されていない 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。

Kubernetes とフェイルオーバー

Kubernetes は、PGAIHM 内でフェイルオーバーと冗長性を処理します。次の構成がサポートされています。

- **単一クラスター、単一ゾーン**複数のノードとポッドを含む 1 つのクラスターと 1 つのゾーンを含む基本的な Kubernetes セットアップ。
- **単一クラスター、複数ゾーン**複数のゾーンに分割された Kubernetes クラスター。ゾーンは別個のラックまたはデータセンターを表すことができます。これを「ストレッチクラスター」と呼びます。
- **マルチクラスター、マルチロケーション**複数の Kubernetes クラスターは、異なるデータセンターにあり、地理的に離れている可能性があります。

これらの構成は、冗長性とフェイルオーバーメカニズムを介してクラスター内またはゾーン全体で継続的なサービスの可用性を保証することにより、HA ポスチャを実現します。DR は、データセンターの停止などの大規模な障害が発生した場合に、さまざまな場所にあるバックアップとデータレプリケーションを使用してサービスとデータを復旧することにより実現されます。

推奨事項

Kubernetes を使用する場合、次の推奨事項が適用されます。

- PGAIHM は HA / DR を簡素化することを目指していますが、Kubernetes の概念の基本的な理解は有益です。HA / DR 計画には、ノード、ポッド、ゾーン、クラスターなどの Kubernetes の概念を理解することが重要です。

- ノードに障害が発生した場合にポッドを再スケジュールするには、Kubernetes クラスター内の適切なリソースが必要です。
- PGAIHM および PostgreSQL ワークロードを別のワーカーノードに分離します。
- ストレージの種類ローカル、共有、オブジェクト、その場所、冗長性、バックアップ/復元手順を含むストレージについては、注意深く計画することが重要です。オブジェクトストレージは、ディザスターリカバリーシナリオ、特にデータセンターの障害の場合に強くお勧めします。
- ストレッチクラスターの場合、ネットワーク遅延は許容範囲内であり、監視ノードの配置を考慮する必要があります。たとえば、OpenShift は 5 ミリ秒未満を推奨しています。

考慮事項

Kubernetes を使用するときは、次の考慮事項に注意してください。

- ワーカーノードがローカルストレージを使用する場合、ノードに障害が発生してもデータは自動的にフェイルオーバーされません。リカバリーにはバックアップからの復元が必要です。
- データの永続性と復旧には、共有ストレージ Ceph、SAN やオブジェクトストレージなどの適切なストレージソリューションを利用することが重要です。
- PGAIHM サービス自分自身のデータが特定のゾーンに関連付けられている場合、マルチゾーンシナリオで自動的にフェイルオーバーしない場合があります。
- フェールオーバーシナリオでのデータの可用性は、ストレージの場所とレプリケーションによって異なります。
- Kubernetes は、ノードに障害が発生した場合にポッドの再スケジュールを処理できますが、データの復旧はストレージ構成に依存します。
- 別のゾーンまたはデータセンターへのフェールオーバーには、データレプリケーションとアプリケーションのリダイレクトについて注意深く計画を立てる必要があります。
- 共有または複製されたストレージが不足していると、リカバリが妨げられる可能性があります。
- PGAIHM は、 HA の Kubernetes 機能を活用しますが、PGAIHM 固有の構成を検討することが重要です。

PGAIHM バックアップと DR

PGAIHM マルチデータセンター環境は、以下を使用して、堅牢なデータ保護戦略を実現します。

- **ボリュームスナップショット**ディスクベースのバックアップ、通常はデータセンターにローカル。
- **Barman:** オブジェクトストレージベースのバックアップ。グローバルオブジェクトストアに保存できます。

推奨事項

PGAIHM バックアップに関する次の推奨事項に注意してください。

- Barman バックアップを保存するには、グローバルオブジェクトストレージソリューションをお勧めします。これにより、データセンター全体でバックアップの可用性を保証できます。

- ボリュームスナップショットと Barman バックアップの両方を使用して、3-2-1 バックアップ戦略 3 つのデータのコピー、2 つの異なるメディア、1 つのオフサイトコピーに従います。
- Kubernetes クラスター全体でバックアップメタデータをレプリケートして、ディザスターリカバリーシナリオでのリストアを簡素化することを検討します。
- Barman とボリュームスナップショットの両方のバックアップを自動化する機能を実装します。
- PostgreSQL データベースに加えて、PGAIHM 自分自身のディザスターリカバリー計画を開発します。
- オンプレミス展開で作業する場合、複数の Red Hat OpenShift インスタンスを接続するためのネットワーク構成を検討します。

考慮事項

PGAIHM バックアップに関する次の考慮事項に注意してください。

- セカンダリデータセンターで PGAIHM が利用できない場合のバックアップの復元は、簡単ではありません。
- バックアップのメタデータは、バックアップが開始される Kubernetes クラスター内に保存されます。そのクラスターが失われると、メタデータへのアクセスも失われ、復元が複雑になります。
- プライマリ PGAIHM を使用せずにセカンダリデータセンターでバックアップを復元するには、手動の手順が必要になる場合があります。
- バックアップとリストアを管理するには、プライマリ PGAIHM インスタンスが必要です。
- 自動バックアップは 1 つの場所にのみ送信できます。システムは、場所を超えてバックアップ定義を自動的にレプリケートしません。
- 自動バックアップ用に Barman またはボリュームスナップショットを設定できますが、UI を介して両方を同時に設定することはできません。
- プライマリ PGAIHM が利用できない場合、バックアップの復元には CLI の相互作用と手動構成が必要になる場合があります。
- セカンダリデータセンターでデータを復旧するには、プライマリ PGAIHM がなくても、バックアップデータにアクセスし、復元を開始する機能が必要です。
- バックアップのメタデータは各 Kubernetes クラスターに対してローカルであるため、あるクラスターで取得されたバックアップは別のクラスターで自動的に可視されません。

使用する

PGAIHM ユーザーインターフェイスは、バックアップとリストアを構成および管理するためのツールを提供します。API を使用して、特定の場所へのバックアップを開始することもできます。`kubectl` を使用して、Kubernetes クラスター内のバックアップを表示できます。

PGAIHM クラスターのバックアップとリストアを構成する方法については、*PGAIHM Backup and Recovery* を参照してください。

PGAIHM HA/DR Recovery

EKS ベースのアプライアンスの CSP リージョンの使用不可、またはハードウェアアプライアンスを使用不可にするデータセンターの停止など、アプライアンスに影響を与えて使用不可にする災害が発生した場合、ディザスターリカバリー DR オプションにより使用可能なバックアップからポイントインタイムにデータベースを復元できます。

PGAIHM バックアップは Velero で処理されます。

PGAIHM を回復するには、2 つの考えられるシナリオがあります。

- **PGAIHM を元の場所に復元する** 2 つのデータセンター DC1、DC2 があり、PGAIHM は DC1 で実行されています。PGAIHM 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。
- **PGAIHM を代替の場所に復元する** 2 つのデータセンター DC1、DC2 があり、PGAIHM は DC1 で実行されています。PGAIHM 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。

DR スコープ

DR 手順は次のことに対応します。

- アプライアンスで作成した Postgres クラスター
- 関連する s3 互換ストレージ領域のアプライアンスで内部定義されたカスタム管理ストレージの場所。

> **注釈**
>
> DR プロシージャは移行コンポーネントをカバーしていませんが、これらを使用して、元のアプライアンスの `transporter-db` および `migration-db` データベースを復元することはできます。

RTO と RPO

復元を実行する機能、および関連する目標復旧時間 RTO、および目標復旧ポイント RPO は、バックアップの頻度とサイズによって異なります。

これらの要素は、環境に割り当てられたクリティカル度とデータの性質によって大きく異なるため、RTO および RPO 値は事前に不明です。環境を適切に準備し、定期的なディザスターリカバリー演習を実行して、RTO および RPO 要件を満たせるようにすることをお勧めします。

バックアップの準備状況

各アプライアンスには、以下を保存するリンクされた s3 互換ストレージがあります。

- **内部バックアップ** PGAIHM アプライアンスデータ
- *Postgres* **バックアップ** Postgres データベースバックアップ

プラットフォームで使用する同じバケットにカスタム **ストレージ場所**を定義することもできます。

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、アプライアンスが存在する 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**
>
> AWS S3 バケットを使用する場合、レプリケーションは Cross-Region Replication を使用して実現できます

Postgres データベースは、デフォルトで継続的バックアップを使用するため、任意のポイントインタイムに復元でき、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。

Postgres クラスターの定義などの重要なアプライアンスデータは、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、デフォルトのスケジュール `velero-backup-kube-state` で定義されているように、毎日 23 時に発生します。

RPO でより頻繁なバックアップが必要な場合、新しいバックアップスケジュールを定義できます。

> **危険**
>
> **アプライアンス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。**

次の例は、必要なリソースを 1 時間ごとにバックアップするカスタムスケジュールを示しています。

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: custom-velero-backup-kube-state
  namespace: velero
spec:
  schedule: 0 \ *\*  \ *\*
  skipImmediately: false
  template:
    includedNamespaces:
    \- \*
    includedResources:
    \- storagelocations.biganimal.enterprisedb.com
    \- clusterwrappers.beacon.enterprisedb.com
    \- backupwrappers.beacon.enterprisedb.com
    snapshotVolumes: false
    ttl: 168h
```

DR プロシージャ

DR 手順は、新しいアプライアンスの展開から、通常の復元手順を使用して Postgres クラスターを復元できるまで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**

この手順は、アプライアンスの 1.0 リリースに基づいており、機能セットの変更に応じて継続的に変更される可能性があります。有効な状態を維持するには、常にテストと更新を行う必要があります。

1. バックアップの可用性を確認する

最初の手順では、使用できないアプライアンスのバックアップ別名「古いバックアップ」が、新しいアプライアンスから到達可能であることを確認します。

これは複数の方法で実現できます。

- 新しいアプライアンスの s3 互換リンクバケットとしてレプリケートバケットを使用するため、古いバックアップを新しいアプライアンスで直接利用できます
- 破損したアプライアンスのバックアップを、新しいアプライアンスのリンクストレージにコピーする。次の項目をコピーする必要があります。
  - フォーマットの内部 EDB バックアップフォルダー `edb-internal-backups/\<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 **カスタム保存場所** に対応するフォルダー

> **注釈**
>
> 新しいアプライアンスに定義された内部バックアップフォルダーは、異なる ` <random-string>` を持つため、古いものとは異なります。

2. 準備手順

Velero のリカバリバックアップ保存場所を定義する

利用可能なバックアップがある場合は、Velero の新しい保存場所を定義して、破損したアプライアンスのバックアップからリソースを復元できるようにします。これは、これらのバックアップの上書きまたは削除を防ぐ読み取り専用の場所になります。

新しい保存場所を定義するには、次の Kubernetes マニフェストを使用します。

```
apiVersion: velero.io/v1
kind: BackupStorageLocation
metadata:
  annotations:
    appliance.enterprisedb.com/s3-prefixes: edb-internal-backups/\<old-backups-ramdom-
↪string\>/velero
  labels:
    appliance.enterprisedb.com/s3-credentials: bound
  name: recovery
  namespace: velero
spec:
```

(次のページに続く)

(前のページからの続き)

```
  accessMode: ReadOnly
  config:
    insecureSkipTLSVerify: "false"
    region: \<region-of-attached-bucket\>
    s3ForcePathStyle: "true"
  default: false
  objectStorage:
    bucket: \<linked-bucket-name\>
    prefix: edb-internal-backups/\<old-backups-random-string\>/velero
  provider: aws
```

`velero get backup-locations` コマンドを使用して確認します。`Available` として表示される必要があります。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。

リカバリーのための Velero バックアップの選択

古い内部 Velero バックアップが復旧保存場所で利用できるようになると、次のコマンドを使用してリストできます。

```
velero get backups \--selector velero.io/storage-location=recovery
```

通常、最新の利用可能な完全なバックアップがリカバリーに選択されます。復元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。

例

```
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、文書の残りの部分では**リカバリ日**として参照されます

追加要件

リカバリ手順には、次の要件が適用されます。

- 新しいアプライアンスは、古いものと同じバージョンの Postgres AI ソフトウェア展開を実行します。
- 古いアプライアンスで使用されていたのと同じ場所 `locations.beacon.enterprisedb.com` カスタムリソースは、新しいアプライアンスで使用できます。 `Locations` は、現在インストール中に作成される内部リソースであり、UI では使用できません。`managed-devspatcher` がデフォルト値です。
- 古いアプライアンスでクラスターを構築するために使用されるコンテナイメージは、新しいアプライアンスで利用できます。

3. 復旧手順

EDB 内部データベース app-db および beacon-db の復元

古いバックアップが利用できるようになると、EDB 内部データベースを復元できます。内部データベースごとに、次の手順を使用します。

1. クラスターマニフェストを yaml ファイル kubectl get \<cluster-name\> \-o yaml \>\ <cluster-name\>.yaml に保存します。
2. yaml ファイルのクラスター仕様を編集して、バックアップからクラスターが作成されるようにします。

- ブートストラップの **init** セクションを **recovery** セクションに置き換えます。

```
recovery:
  database: \<database name as in the init section\>
  owner: \<owner name as in the init section\>
  source: \<pg-cluster-name\>
  secret:
    name: \<secret name as in the init section\>
  recoveryTarget:
    targetTime: "\<recovery date in YYYY-MM-DD HH:MM:SS+00 format\>"
```

- 次のセクションを追加します。

```
externalClusters:
\- barmanObjectStore:
    destinationPath:  S3://\<linked-bucket-name\>/edb-internal-backups/\<old-backups-
↪random-string\>/databases
    s3Credentials:
      inheritFromIAMRole: true
    wal:
      maxParallel: 8
  name: \<pg-cluster-name\>
```

- 次の接頭辞を inheritedMetdata セクションの appliance.enterprisedb.com/s3-prefixes アノテーションに追加しますリストはコンマ区切りです。

```
edb-internal-backups/\<old-backups-random-string\>/databases/\<db-name\>
```

- クラスターを削除します。

```
kubectl delete cluster \<cluster-name\>)
```

- クラスターのバックアップ領域をクリーンアップします。

```
aws s3 rm s3://\<linked-bucket-name\>/edb-internal-backups/\<new-backups-random-
↪string\>/databases/\<pg-cluster-name\>  \--recursive
```

- 再作成するクラスターの yaml ファイルを適用します kubectl apply \-f \<cluster-name\>.yaml

- クラスターが正常に復元され、正常な状態になったら、名前空間 upm-beaco-ff-base の accm-server を再起動します。

この時点で、新しいクラスターのポータルは再び利用できるようになります。

Velero プラグインの構成

プラグインは、Kubernetes リソースを正しい状態に復元するのに役立つため、カスタム管理対象ストレージの場所のみが復元され、Postgres クラスターリソースが削除されたものとして復元されるため、必要に応じてデータを後で復元できます。

プラグイン構成は ConfigMap を介して作成されるため、次のマニフェストを適用する必要があります。

```
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: "\<recovery date in YYYY-MM-DDTHH:MM:SSZ format\>"
  \# old and new buckets for internal custom storage locations
  oldBucket: \<old-appliance-bucket-name\>
  newBucket: \<new-appliance-bucket-name\>
```

カスタム管理ストレージの場所を復元する

次の Velero 復元リソースマニフェストを構成および適用します。

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-1-storagelocations
  namespace: velero
spec:
  \# Change the backup name to a custom backup name as required
  backupName: \<velero-backup-name\>
  includedResources:
  \- storagelocations.biganimal.enterprisedb.com
  includeClusterResources: true
  labelSelector:
    matchLabels:
      biganimal.enterprisedb.io/reserved-by-biganimal: "false"
```

クラスターラッパーを復元する

次の Velero 復元リソースマニフェストを構成および適用します。

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-2-clusterwrappers
  namespace: velero
spec:
  \# Change the backup name to a custom backup name as required
  backupName: \<velero-backup-name\>
  includedResources:
  \- clusterwrappers.beacon.enterprisedb.com
  restoreStatus:
    includedResources:
    \- clusterwrappers.beacon.enterprisedb.com
```

バックアップラッパーを復元する

次の Velero 復元リソースマニフェストを構成および適用します。

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: restore-3-backupwrappers
  namespace: velero
spec:
  \# Change the backup name to a custom backup name as required
  backupName: \<velero-backup-name\>
  includedResources:
  \- backupwrappers.beacon.enterprisedb.com
  restoreStatus:
    includedResources:
    \- backupwrappers.beacon.enterprisedb.com
```

Enabling the Migration Portal AI Copilot

Hybrid ManagerHM を使用して EDB Postgres AI プラットフォームを展開した後、外部データベースの移行に役立つ 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。

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とする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。さらに、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`
      - 推奨バージョン:3
      -

AI コパイロットの有効化

HM をインストールする前または後、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、ハイブリッドマネージャー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。

OpenAI の場合

```
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
```

Azure OpenAI の場合

```
export AZURE_OPENAI_ENDPOINT=<azure_openai_endpoint> # endpoint for the azure openai␣
↪instance
```

(次のページに続く)

(前のページからの続き)

```
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。

OpenAI の場合

```
kubectl create secret generic ai-vendor-secrets \
  --namespace=edb-migration-copilot \
  --type=opaque \
  --from-literal=AI_VENDOR=OPENAI \
  --from-literal=OPENAI_ENDPOINT="${OPENAI_ORG_ID}" \
  --from-literal=OPENAI_API_KEY="${OPENAI_API_KEY}"
```

Azure OpenAI の場合

```
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 値を使用して migration-portal-values.yaml という新しい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。

OpenAI の場合

```
parameters:
  edb-migration-copilot:
    ai_vendor_secrets: ai-vendor-secrets
```

Azure OpenAI の場合

```
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、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. シークレットをインストールに適用する 中に helm upgrade コマンドを実行したときに、作成された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、シークレットはベンダーのデフォルトのシークレッ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 Key Management Systems 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。

最初に、選択したキー管理システムプロバイダーが `values.yaml` で有効になっていることを確認します。

- *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*
- *Google Cloud KMS* **キーの作成**

Adding KMS support

ハイブリッドマネージャーを展開すると、ブートストラップキット構成 `values.yaml` 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, 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、変更した `values.yaml` で helm チャートを再適用して、システムに新しい構成を設定する必要があります。Helm チャート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, gcp_kms, hashicorp_vault.
#     Default values: passphrase, hashicorp_vault.
transparentDataEncryptionMethods:
  - passphrase
  - hashicorp_vault
  - aws_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, 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 Google Cloud Key Management Service as the KMS for TDE*

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: |
   nodeAffinity:
     preferredDuringSchedulingIgnoredDuringExecution:
     - preference:
         matchExpressions:
         - key: edbaiplatform.io/control-plane
           operator: Exists
       weight: 1
 ha:
     enabled: true
     raft:
        enabled: true
```

(次のページに続く)

(前のページからの続き)

```
        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. オーバーライドを使用して Vault をインストールします。

```
helm install -n vault vault hashicorp/vault --values helm-vault-raft-values.yml  --
↪create-namespace
```

4. 続行する前に、すべてのポッドが「正常」ステータスに到達するまで待ちます。

```
kubectl -n vault get pods
```

次に、base64 エンコードされた公開 PGP キーまたは Keybase ユーザー名を使用して Vault インスタンスを初期化し、マスターキーで封印を解除する必要があります。Vault のドキュメント *Initialize and unseal Vault* を参照してください。

Vault の構成とキーの作成

Vault サービスとハイブリッドマネージャー間の通信を有効にします。

1. Vault がインストールされているポッドにアクセスします。

```
kubectl -n vault exec --stdin=true --tty=true vault-0 -- /bin/sh
```

2. ボルトインスタンスの Kubernetes 通信を有効にします。

```
vault auth enable kubernetes
```

3. Kubernetes クラスターで認証するように Vault を更新し、ハイブリッドマネージャーが Vault を使用できるようにします。

```
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. *Transit* シークレットエンジンを有効にして、キー管理機能をアクティブにします。

```
vault secrets enable transit
```

5. キーの Vault ポリシー定義を含む新しいファイルを作成します。

この例では、ポリシーを含むファイルの名前は `tde.txt` です。

```
cat > /tmp/tde.txt << EOL
path "transit/*" {
  capabilities = ["create", "read", "update", "patch", "delete", "list"]
}
EOL
```

6. 作成したファイルを使用して、HashCorp Vault のセキュリティポリシーを更新します。

```
vault policy write tde /tmp/tde.txt
```

7. Transit エンジン周辺キーローテーション、暗号化方法などと *adjust according to your requirements* のデフォルトパラメーター値を確認します。または、次の手順でキーの作成中に *vault write* コマンドを使用して、特定のキーのデフォルト値をオーバーライドできます。

8. トランジットエンジンを使用する新しいキーを作成します。

<my-tde-key> を選択したキー名に置き換えます。

```
vault write -f transit/keys/<my-tde-key>
```

9. Vault ポリシーを Kubernetes サービスアカウントにマッピングする `tde` というロールを作成します。

```
vault write auth/kubernetes/role/tde \
  bound_service_account_names="p-*" \
  bound_service_account_namespaces="p-*" \
  policies=tde \
  ttl=72h
```

!!!note

別の `ttl` 値を構成して、アプリケーションが再認証が必要になる頻度を指定できます。詳細については、*token metadata* に関する Vault のドキュメントを参照してください。

10. ポッドを終了します。

```
exit
```

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

PG AI コンソールを使用して、作成したキーを PG AI Hybrid Manager 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 **HashCorp Vault** を選択します。
5. **Vault address** に、HashCorp 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。

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 `values.yaml` 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"
    ],
    "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。

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 `values.yaml` 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。

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。

Troubleshooting PGAIHM

このガイドでは、ハイブリッドマネージャー PGAIHM のトラブルシューティングの手順を提供します。

ログファイル

PGAIHM のトラブルシューティングを行うときに次のログファイルを使用できます。

Log file name	Category	Location	Explanation

次のページに続く

表 28 – 前のページからの続き

Log file name	Category	Location	Explanation

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 boot-strap: the LoadBalancer service is pending.	Fix the loadbalancer IAM role first, then delete the LB pod to trigger recreation.
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

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 ロールを修正して、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/
```

(次のページに続く)

(前のページからの続き)

```
↪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:
```

(次のページに続く)

(前のページからの続き)

```
  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 ではないことが原因で、Loki < https://github.com/grafana/loki/issues/8634 > 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 にする必要があります。

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="${PORTAL_DOMAIN_NAME}" \
    --set parameters.transporter-rw-service.domain_name="${TRANSPORTER_RW_SERVICE_
↪DOMAIN_NAME}" \
```

(次のページに続く)

(前のページからの続き)

```
    --set parameters.transporter-dp-agent.rw_service_url="https://${TRANSPORTER_RW_
↪SERVICE_DOMAIN_NAME}/transporter" \
    --set parameters.upm-beacon.server_host="${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。 ``staticPasswords``

セクションが適切に設定されていない場合。適切な値を使用して 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、aes_key が正しく設定されていないことが原因で発生します。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、ノードなしの 16vCPU と 64GB RAM です。クラスターの作成時に、UI 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しますが、動作せず、ノードの作成後も UI 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

UI でディスクをスケールすると、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",
```

(次のページに続く)

(前のページからの続き)

```
    "namespace": "p-2hgee2782y"
  },
  "namespace": "p-2hgee2782y",
  "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 の確認と編集

```
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、新しい変更を適用します。

```
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
```

Using query recommendations in Hybrid Manager

ハイブリッドマネージャーでのクエリ推奨事項の使用

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。

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。

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** 。

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。

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。

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。

クエリの推奨事項を調整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。

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。

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 **ビーコンエージェント**をインストールして実行する必要があります。

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。

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。

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。

関連情報

- *Cluster Query Diagnostics*
- *Cluster overview*

第6章　AI Factory in Hybrid Manager

Hybrid Manager の **AI Factory ワークロード**は、スケーラブルな AI、機械学習、および Gen AI 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 アシスタント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で管理されてい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 AI 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 Gen AI アシスタントとエージェントを展開します**
- KServe を活用した推論と GPU アクセラレーションを統合した**大規模な AI モデルの提供**
- **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**

詳細なアーキテクチャとコア概念については、AI Factory Hub を参照してください。

6.1 ラーニングパス

構造化された学習パスに従って、ハイブリッドマネージャーの AI ファクトリーをマスターします。

- *AI Factory 101* **ハイブリッドマネージャー** — 初歩の概念と使用法
- *AI Factory 201* **ハイブリッドマネージャー** — Gen AI および AI Factory のワークロードの構築と管理
- *AI Factory 301* **ハイブリッドマネージャー** — 高度な統合、スケーリング、ガバナンス、および最適化

ハイブリッドマネージャーワークロードの## AI Factory

ハイブリッドマネージャーは、AI ファクトリー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 Gen AI ワークロード

- Gen AI Overview in Hybrid Manager — ハイブリッドマネージャーの Gen AI 機能
- *Agent Studio in Hybrid Manager* —アシスタント、ツール、構造、およびルールセットを作成および管理する
- *Gen AI Builder in Hybrid Manager* — ナレッジベースとデータレイクの管理

6.1.2 モデルの管理とサービス

- *Image and Model Library Explained* — モデルアセットを管理および統治する
- *GPUs in Model Serving* — 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

6.1.3 ベクトルエンジン

- ベクトルエンジン — Postgres を使用した統合ベクトル検索および類似性機能

6.2 はじめる

ハイブリッドマネージャーで AI ファクトリーを使用してビルドを開始するには

- *AI Factory Learning Paths (Hybrid Manager)* に従ってください
- 一般的な使用例 および industry solutions の探索
- ハイブリッドマネージャー固有の *AI Factory How-To Guides (Hybrid Manager)* の読み取り
- *Agent Studio in Hybrid Manager* または *GPUs in Model Serving* を使用して、最初の Gen AI アシスタントまたはモデルを展開する

Hybrid Manager の AI Factory は、ハイブリッドデータ資産全体で AI を運用化するためのスケーラブルで安全なプラットフォームを提供します。今日から構築を始めましょう。

6.2.1 Enabling AI Factory

AI Factory の Gen AI Builder 機能を使用する前に、いくつかの重要な構成手順で有効にする必要があります。

Fernet キーの生成オプショナル

Fernet キーをまだ生成していない場合は、これを行って、Gen AI Builder が内部で使用する暗号ライブラリを有効にする必要があります。このキーは、AI Factory の機密データを暗号化するために使用されます。

EDB Postgres AI Hybrid Manager `-install-secrets.sh` スクリプトを使用して環境をセットアップしている場合、既に Fernet キーが生成されている場合があります。そうでない場合は、ターミナルで次のコマンドを使用して生成できます。EDB Postgres AI Hybrid Manager クラスターをポイントするように `kubectl` が構成されていることを確認します。

```
FERNET_KEY=$(dd if=/dev/urandom bs=32 count=1 2>/dev/null | base64)

kubectl create namespace upm-griptape

kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
```

(次のページに続く)

(前のページからの続き)

```
    name: fernet-secret
    namespace: upm-griptape
stringData:
    FERNET_KEY: ${FERNET_KEY}
EOF
```

このコマンドは、ランダムな Fernet キーを生成し、upm-griptape 名前空間に Kubernetes シークレットを作成します。キーは AI Factory が機密データを暗号化するために使用されます。

S3 バケットを作成する

GenAI Builder では、データを保存するために S3 バケットが必要です。このバケットはデータレイクと呼ばれます。このデータレイクは、GenAI Builder 構造、ツール、インデックス付きデータをホストします。理想的には、S3 バケットが AI Factory 展開と同じリージョンで作成されていることを確認します。AWS を使用する場合、バケット名はグローバルに一意である必要があることに注意してください。S3 バケットを作成するときは、次の情報をメモする必要があります。- S3 バケット Datalake に使用するバケットの名前。- S3 アクセスキー Datalake への接続に使用されるアクセスキー。 - S3 Secret Access Key Datalake への接続に使用されるシークレットアクセスキー。 - S3 URL Datalake への接続に使用されるエンドポイント URL。- S3 リージョン us-west-2 や eu-central-1 などの S3 バケットが存在するリージョン。

アクセスキーとシークレットアクセスキーは、AI Factory の S3 バケットへのアクセスを認証するために使用されます。これらは AWS アカウントから取得されるか、S3 バケット用に作成された特定のキーである場合があります。

S3 バケットの CORS 設定

Web ベースの GenAI Builder UI を使用している場合、S3 バケットの CORS クロスオリジンリソース共有設定を構成する必要がある場合があります。これにより、GenAI Builder UI は、さまざまなオリジンから S3 バケット内のリソースにアクセスできます。AWS の CORS ルールは、たとえば、 **アクセス許可**の下の S3 バケット設定で設定できます。 CORS 構成は次のようになります。

```
[
{
    "AllowedHeaders": [
        "*"
    ],
    "AllowedMethods": [
        "PUT",
        "POST",
        "DELETE",
        "GET",
        "HEAD"
    ],
    "AllowedOrigins": [
        "https://${PORTAL_DOMAIN_NAME}"
    ],
```

(次のページに続く)

(前のページからの続き)

```
    "ExposeHeaders": []
}
]
```

${PORTAL_DOMAIN_NAME} を GenAI Builder UI のドメイン名に置き換えます。この構成により、許可されたオリジンから指定された方法が S3 バケットにアクセスできます。

```
[
 {
"origin": ["https://<portal url>"],
"method": ["GET", "PUT", "POST", "DELETE", "HEAD"],
"responseHeader": ["*"],
"maxAgeSeconds": 3600
 }
]
```

次に、 gsutil を使用してそれを設定します。

```
gsutil cors set cors.json gs://<bucket name>
```

これにより、GenAI Builder UI は、S3 バケットの場合と同様に、さまざまなオリジンから GCS バケット内のリソースにアクセスできます。

バケットを GenAI Builder に接続する

S3 バケットを作成し、必要な資格情報を入手したら、 EDB Postgres AI Hybrid Manager の GenAI Builder に接続できます。このプロセスには、GenAI UI へのログインと AI Factory 設定の構成が含まれます。

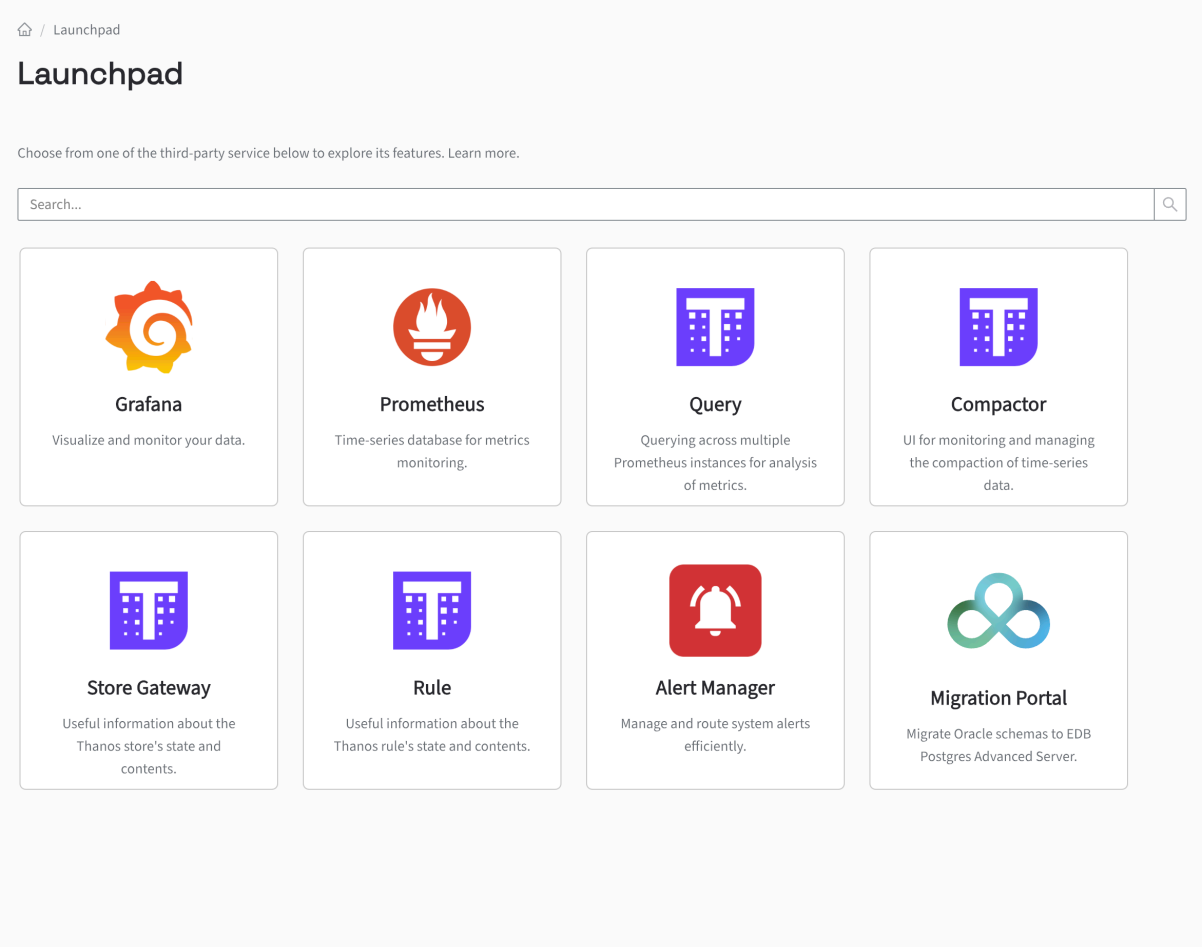
GenAI UI にログインするには、 EDB Postgres AI Hybrid Manager をセットアップして実行する必要があります。それを入手したら、次の手順を実行します。

1. **EDB Postgres AI Hybrid Manager にアクセスします** Web ブラウザーを開き、EDB Postgres AI Hybrid Manager インスタンスの URL に移動します。
2. **Launchpad に移動します** EDB Postgres AI Hybrid Manager で、“Launchpad” セクションに移動します。

   width
   70% :alt: Launchpad Menu

   Launchpad Menu

1. **GenAI Builder を選択します** Launchpad で「GenAI Builder」を見つけて選択して、GenAI UI にアクセスします。
2. **ログイン** 電子メールアドレスを使用して、GenAI UI にログインします。EDB Postgres AI ハイブリッドマネージャーのセットアップ中に作成したアカウントを使用している場合、owner@mycompany.com はそのメールアドレスを使用します。
3. **パスワードを入力します**プロンプトが表示されたら、アカウントのパスワードを入力します。セットアップ中に作成されたデフォルトのアカウントを使用している場合、パスワードは pgai_admin です。


Launchpad
Launchpad
Choose from one of the third-party service below to explore its features. Learn more.
Search...
Grafana
Visualize and monitor your data.
Prometheus
Time-series database for metrics monitoring.
Query
Querying across multiple Prometheus instances for analysis of metrics.
Compactor
UI for monitoring and managing the compaction of time-series data.
Store Gateway
Useful information about the Thanos store's state and contents.
Rule
Useful information about the Thanos rule's state and contents.
Alert Manager
Manage and route system alerts efficiently.
Migration Portal
Migrate Oracle schemas to EDB Postgres Advanced Server.

4. **GenAI UI にアクセスします** ログインすると、GenAI UI が表示され、AI Factory の構成を求めるダイアログが表示されます。

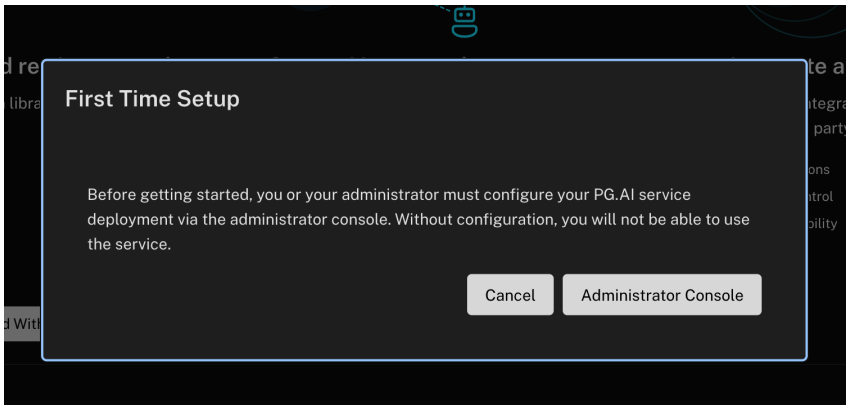


図 1: GenAI UI Configuration

1. **管理者コンソールの選択** [Administrator Console] オプションを選択して、AI Factory の構成を続行します。 AI Factory の構成フォームが表示されます。

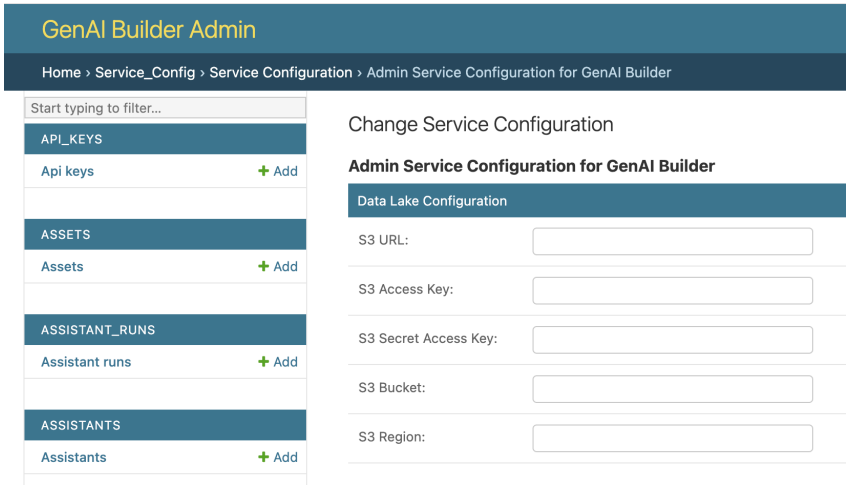


図 2: GenAI Builder first time form

1. **サービス構成変更フォームに入力します**構成フォームには、次の情報を提供する必要があります。

- **S3 URL** S3 バケットの URL を入力します。これは通常、`https://s3.<region-name>.amazonaws.com/` のフォーマットです。`<region-name>` .
- の代わりに s3 バケットが作成されたリージョンです。 **S3 Access Key** S3 バケットのアクセスキーを入力します。これは、AWS アクセスキーまたは S3 バケット用に作成された特定のアクセスキーです。
- **S3 Secret Access Key** S3 バケットのシークレットアクセスキーを入力します。これは、AWS シークレットアクセスキーまたは S3 バケット用に作成された特定のシークレットアクセスキーです。
- **S3 バケット** 前に作成した S3 バケットの名前を入力します。
- **S3 リージョン** `us-west-2` や `eu-central-1` など、S3 バケットが存在するリージョンを指定します。

1. **構成を送信する** 必須フィールドに入力したら、[Submit] ボタンをクリックして構成を保存します。

これにより、AI Factory が有効になり、GenAI Builder 機能の使用を開始できるようになります。

6.2.2 Gen AI in Hybrid Manager

ハイブリッドマネージャー は、AI ファクトリーワークロードを介して **Gen AI 機能** の運用グレードの展開を可能にし、 **Postgres データ** 、**AI パイプラインと深く統合する ソブリン AI ソリューションを提供します、および**エンタープライズアーキテクチャ**。

Hybrid Manager の **Kubernetes ネイティブ** 基盤と **AI Factory** を組み合わせることで、信頼できるデータで動作し、管理されたインフラストラクチャ内で実行され、Postgres にシームレスに接続する高度な**エージェント AI アプリケーション** を構築できます。

このページでは、Gen AI ワークロードがハイブリッドマネージャーでどのように動作し、より広範な **EDB Postgres AI PG AI)** エコシステムにどのように適合するかについて説明します。

アーキテクチャの詳細と完全な機能のリファレンスについては、 *Gen AI Hub* から始めます。

コアコンポーネント

Agent Studio

- AI アシスタントを構築および管理するための**ローコード/ノーコードインターフェイス** 。
- ビジネスチーム、コンテンツ所有者、および開発者がコラボレーションできるようにします。
- 以下の構成をサポート
- LLM モデル Model Serving を介して提供されます
- ナレッジベース
- ルールセットとガバナンスポリシー
- システム統合用ツール
- コンテキスト永続化のためのメモリ設定

Agent Studio in Hybrid Manager を参照してください。

Gen AI Builder

- 高度な AI 開発者に最適な **コード駆動エージェント開発環境** 。
- 以下の作成をサポート
- 複雑なマルチエージェントワークフロー
- ツールチェーンと API オーケストレーション
- 基本的なチャットを超えたカスタムエージェントロジック
- ビルダーは以下を活用できます。
- モデル提供エンドポイント KServe InferenceService
- 検索拡張生成 RAG のナレッジベース
- プロジェクト内で定義されたツールと構造
- AIDB および Vector Engine を介した Postgres ベースのデータソース

Gen AI Builder in Hybrid Manager を参照してください。

ハイブリッドマネージャー固有の考慮事項

- すべての Gen AI ワークロードは、ハイブリッドマネージャープロジェクトの **Kubernetes クラスター** 内で実行されます。
- **Model Serving** を介して提供されるモデルは、プロジェクトレベルの制御で保護されます。
- ナレッジベースとパイプラインは、プロジェクトのストレージとデータガバナンス境界内で動作します。
- すべての Gen AI アクティビティの可観測性は、ハイブリッドマネージャーモニタリングと統合されています。
- 推論メトリック
- アシスタントの使用
- パイプラインの状態
- GPU リソース使用率
- ハイブリッドマネージャーは **ソブリン AI** を有効にします
- データはプロジェクトに残ります。
- モデルはインフラストラクチャで実行されます。
- すべての AI パイプラインと出力は監査可能および観察可能です。

Sovereign AI explained を参照してください。

Agent Studio in Hybrid Manager

ハイブリッドマネージャー の **エージェントスタジオ** は、**ソブリン AI** と完全に連携して、完全にハイブリッドマネージャープロジェクト環境内で実行される **AI アシスタント**を作成、管理、および制御する統合方法を提供しますアーキテクチャ。

Agent Studio を使用して、次のアシスタントをビルドできます。

- モデルサービス **を介して展開された** モデルと対話します。
- **ナレッジベース** からコンテンツを取得します。
- ガバナンスとコンプライアンスに **ルールセット** を適用します。
- アクションを実行するか、外部システムと統合する **ツール**を実行します。
- ステートフルで監査可能な会話のために **スレッド** を維持します。

  より深い背景については、AI Factory ハブの *Agent Studio concepts page* を参照してください。

ハイブリッドマネージャーでの Agent Studio の動作

Agent Studio コンポーネントは、**ハイブリッドマネージャープロジェクトの Kubernetes クラスター**で完全に実行され、完全な可観測性とガバナンスを備えています。

- **アシスタント** は Agent Studio で定義され、プロジェクトインフラストラクチャ内で実行されます。
- アシスタントは、 **モデル提供**を介して展開されたモデルを呼び出すことができます。*Model Serving in Hybrid Manager* を参照してください。
- アシスタントは、同じプロジェクト内で構築された **ナレッジベース**からコンテンツを取得します。*Knowledge Bases explained* および *AI Accelerator Knowledge Base page* を参照してください。
- アシスタントは Agent Studio 構成で定義された **ルールセット**を適用します。 *Rulesets explained* を参照してください。
- アシスタントは、プロジェクト内に実装された **ツール**を呼び出すことができます。 *Tools explained* を参照してください。
- **スレッド**は、監査とレビューをサポートするために自動的にキャプチャされます。*Threads explained* を参照してください。

これらのコンポーネントはすべてあなたの制御の下で動作します。

- データはインフラストラクチャに残ります
- モデルは Kubernetes クラスターの GPU で実行されます
- ナレッジベースと埋め込みは Postgres とオブジェクトストレージに残ります
- アシスタントの動作は完全に観察可能で、ルールセットによって管理されます

これは、 **ソブリン AI** 原則をサポートします —*Sovereign AI explained* を参照してください。

ハイブリッドマネージャーの一般的な Agent Studio フロー

1. **ナレッジベース** — *How-To: Create a Knowledge Base* を定義します。
2. **ルールセット** — *How-To: Create a Ruleset* を作成して適用します。
3. ナレッジベース検索— *How-To: Create a Retriever* を制御する **Retriever** を定義します。
4. API 呼び出しまたはビジネスロジック用の **ツール** *How-To: Create a Tool* を展開します。
5. モデル提供 — *How-To: Deploy a NIM container* を介して、サポートする **モデル**を展開します。
6. Agent Studio — *How-To: Create an Assistant* で **アシスタント** を作成します。
7. **Threads** — *How-To: View Threads* を使用してテストおよび検証します。

主要なリンクと学習リソース

- *Agent Studio in Hybrid Manager*
- *Create an Assistant — How-To Guide*
- *Assistants explained*

- *Rulesets explained*
- *AI Accelerator Knowledge Bases*
- *Tools explained*
- *Threads explained*
- *Model Serving in Hybrid Manager*
- *Sovereign AI explained*

次のステップ 最初のアシスタントを構築した後、AI Factory **ラーニングパス** の高度なトピックを探索します。

Gen AI Builder in Hybrid Manager

ハイブリッドマネージャー の **Gen AI Builder** を使用すると、データ、モデル、アプリケーションの動作を完全に制御しながら、**ソブリン AI** アーキテクチャと完全に調整された、高度な*エージェント AI アプリケーション** を構築できます。 。

ビルダーアプリケーションは、プロジェクトの Kubernetes クラスター内のコンテナとして実行され、以下と対話します。

- **KServe-powered Model Serving** エンドポイント
- ハイブリッドマネージャープロジェクトに組み込まれた **ナレッジベース**
- ビジネスロジックを実行し、外部システムにアクセスするための **ツール**および **構造体**

  より深い背景については、AI Factory ハブの *Gen AI Builder concepts page* と *Gen AI Builder in Hybrid Manager* を参照してください。

ハイブリッドマネージャーでの Gen AI Builder の仕組み

- Gen AI Builder アプリケーションは、プロジェクトの **Kubernetes クラスター**で実行されます。外部コンピューティングの依存関係は必要ありません。
- ビルダーは、 **モデル提供**を介して展開されたモデルを呼び出すことができます。*Model Serving in Hybrid Manager* を参照してください。
- ビルダーは、ハイブリッドマネージャープロジェクトにビルドされた**ナレッジベース** からコンテンツを取得します。*Knowledge Bases explained* を参照してください。
- ビルダーは、プロジェクト内に実装された **ツール** を呼び出します。*Tools explained* を参照してください。
- ビルダーは **スレッド**を管理して、ステートフルで監査可能な会話を有効にします。*Threads explained* を参照してください。
- ビルダーは **ルールセット**を活用して、ガバナンスと動作の一貫性を適用できます。*Rulesets explained* を参照してください。

ハイブリッドマネージャーの主要な考慮事項

- ビルダーアプリケーションは、高パフォーマンスの推論のために必要な場合、**GPU 対応インフラストラクチャ** で実行されます *Update GPU Resources* 。
- ビルダーは、ハイブリッドマネージャープロジェクト *Deploy AI Models* 内に展開された **KServe ベースの Model Serving** エンドポイントにアクセスします。
- ビルダーは、プロジェクト内で作成された **ナレッジベース**にアクセスできます。 *AI Accelerator Knowledge Base page* を参照してください。
- ビルダーは完全にハイブリッドマネージャー環境内で動作します—**ソブリン AI** 原則をサポート
- お客様のデータ
- あなたのモデル
- AI パイプライン
- 完全な可観測性と制御

Sovereign AI explained を参照してください。

ハイブリッドマネージャーの一般的な Gen AI Builder フロー

1. **ツール** — *How-To: Create a Tool* をビルドまたは展開します。
2. **ナレッジベース** — *How-To: Create a Knowledge Base* を作成および構成します。
3. モデル提供 — *How-To: Deploy a NIM container* を介して、サポートする **モデル**を展開します。
4. 必要に応じて **ルールセット** を構成します — *How-To: Create a Ruleset* 。
5. Gen AI ビルダー *Gen AI Builder in Hybrid Manager* を使用して、**アシスタント**および/または **エージェントワークフロー** をビルドします。
6. **スレッド** をモニタし、動作を調整します — *How-To: View Threads* 。

詳細はこちら

- *AI Factory Concepts*
- *Sovereign AI explained*
- *Gen AI Builder in Hybrid Manager*
- *Create an AI Tool — How-To Guide*
- *Model Serving in Hybrid Manager*
- *AI Accelerator Knowledge Bases*

6.2.3 Learn AI Factory in Hybrid Manager

AI Factory で学ぶ

このセクションは、 EDB Postgres AI プラットフォームの AI ファクトリーを使用するための基礎知識を構築し、実践的なスキルを開発するのに役立ちます。GPU サポートを有効にするか、モデルを展開するか、AI 搭載アプリケーションをデザインするかにかかわらず、ここにあるリソースは、あらゆる段階であなたをサポートするように構成されています。

コンテンツは、 Diátaxis framework を使用して、インテントごとに編成されます。

AI Factory How-To Guides (Hybrid Manager)

AI Factory でタスクを実行するための詳細な手順

- *Setup GPU resources in Hybrid Manager* —GPU リソースをプロビジョニングし、クラスターを準備します。
- *Verify InferenceServices and GPU Usage in Hybrid Manager* —展開後のモデルのステータスと GPU 使用率を確認します。

ラーニングパス

複雑さによって調整された、AI Factory トピックを介した厳選されたパスをたどります。

- *AI Factory Learning Paths (Hybrid Manager)* —プラットフォームを有効にすることから始めて、モデルの展開と高度な使用に進みます。

AI Factory in Hybrid Manager

このセクションでは、主要な **AI Factory** 機能が**ハイブリッドマネージャー**のコンテキストでどのように動作するかについて説明します。

Hybrid Manager は、Hybrid ManagerHCP プロジェクト内の管理対象 **AI ワークロード** として AI Factory を提供します。これにより、**エージェント AI アプリケーション** を実行し、 **AI モデル**を展開および提供し、**ベクターおよび知識ベースの検索**を統合できます。すべて Kubernetes ネイティブの制御とセキュリティを使用して。

トピック

- *Gen AI in Hybrid Manager*
- *Agent Studio in Hybrid Manager*
- *Gen AI Builder in Hybrid Manager*
- *Model Serving in Hybrid Manager*
- *Asset Library in Hybrid Manager*
- *AI Accelerator Pipelines Knowledge Bases*

AI Factory と **Hybrid Manager の Kubernetes ネイティブ制御および GPU リソース**を組み合わせることで、高度に統合された実稼働対応の AI アプリケーションを提供できます。

Container Registry Integration

Container Registry の統合

ハイブリッドコントロールプレーン HCP のイメージおよびモデルライブラリーは、コンテナレジストリを利用しています。レジストリは、次の実際の画像コンテンツを提供します。

- **データベースイメージ** PostgreSQL、EPAS、PGE
- **AI モデルイメージ** NVIDIA NIM、将来のカスタムモデル

コンテナレジストリの統合がどのように機能するかを理解することは、イメージライブラリーを管理および拡張するためのキーとなります。

それはどのように機能しますか？

HCP は、 **Beacon Agent** を利用して、接続されたレジストリからイメージメタデータをプルします。

ビーコンエージェントは **リポジトリルール**を使用して、次のことを知ります。

- どのレジストリに接続するか
- どのリポジトリをスキャンするか
- どの画像タグをインポートするか

→ HCP のイメージライブラリー UI は、エージェントがプルしたものを反映しています。

主要なコンセプト

Concept	Meaning
Registry	A container registry (ex: Azure ACR, AWS ECR, GCP GCR, private registry)
Repository	A collection of images for a given software or model (ex: *postgresql*, *meta/llama-3.3-49b*)
Tag	A version of an image (ex: *17.4-2503310734*, *1.8.5*, *latest*)
Repository Rule	A rule that tells Beacon Agent which repositories/tags to discover
HCP Tags	Organizational tags within HCP → these are separate from registry tags

デフォルトレジストリ

エフェメラル HCP 環境の場合

- デフォルトのレジストリは `upmdev.azurecr.io` Azure ACR です。
- リポジトリ
- `postgresql` → PostgreSQL コミュニティ画像。
- `edb-postgres-advanced` → EPAS 画像。
- `edb-postgres-extended` → PGE 画像。

顧客環境の場合

- 顧客は **プライベートレジストリ** を提供します。
- イメージライブラリーは、そのレジストリからプルするように構成されています。

リポジトリルール

リポジトリルールは、ビーコンエージェントが検出するものを駆動します。

- Image ライブラリー UI → “Manage Repositories”から**カスタムリポジトリ** を追加できます。
- 高度な制御用にビーコンエージェント構成を編集できます。

エージェント構成セクションの例

```
dispatcher:
  capabilities:
    provisioning:
      images:
        discovery:
          registry:
            postgres_repositories:
              - name: postgresql
                description: Community PostgreSQL
              - name: edb-postgres-advanced
                description: EPAS
              - name: edb-postgres-extended
                description: PGE
              - name: postgresql-torsten
                description: Custom PG Build with AIDB
```

これにより、どのリポジトリをプルして表示するかエージェントに指示されます。

→ この構成を編集して、リポジトリをさらに追加できます上級。

プライベートレジストリの統合フローイメージライブラリー UI を介してリポジトリを追加します → ほとんどのユーザーにとって最も簡単なフロー。

エージェントはイメージリストを定期的にプルします。

画像タグとメタデータは画像ライブラリーに表示されます。

これらの画像は次の目的で選択できます。

データベースプロビジョニング

AI モデルの提供 AI Factory のモデルライブラリーを介して

AI Factory モデルライブラリーとの関係 AI Factory のモデルライブラリーは、Image ライブラリーによって提供されています。

すべての AI モデルイメージは、この同じレジストリ → エージェントプル → イメージライブラリー → モデル提供に利用可能を介してフローします。

これにより、すべてのモデルイメージの一貫したガバナンスが保証されます。

→ 参照 AI Factory Model Serving の説明概要

- レジストリは、画像の信頼できるソースを提供します。
- リポジトリルールは、スキャンするリポジトリ/タグを HCP に指示します。
- ビーコンエージェントはイメージをプルし、イメージライブラリー UI を更新します。
- プライベートレジストリを完全に統合および管理できます。
- AI Factory Model ライブラリーは、これと同じシステムを活用します。

関連するハウツー プライベートレジストリの統合

リポジトリルールの定義

関連する説明ページ イメージとモデルライブラリーの説明

AI ファクトリーモデルのサービングの説明

Image and Model Library Explained

イメージとモデルライブラリーの説明

イメージとモデルライブラリーは、ハイブリッドコントロールプレーン HCP のコア機能です。HCP 環境全体でコンテナイメージを管理するための、統合された管理対象リポジトリを提供します。

これには以下が含まれます

- **データベースイメージ** → PostgreSQL クラスタープロビジョニング用。
- **AI モデル画像** → モデルサービング用 AI Factory Model ライブラリー経由。

なぜ単一のライブラリなのでしょうか？

統合されたイメージとモデルライブラリーを使用すると、次のことが保証されます。

- **ガバナンス** → すべての画像 DB または AI モデルは同じ承認フローを通過します。
- **一貫性** → 同じエージェント駆動の同期がコンテナレジストリからイメージをプルします。
- **セキュリティ** → プライベートレジストリの統合は、すべてのイメージタイプに等しく適用されます。
- **柔軟性** → データベースチームと AI チームの両方が共通のフレームワーク内で画像を管理できます。

イメージライブラリーとモデルライブラリーの違いは何ですか？

Capability	Image and Model Library (Hub)	Model Library (AI Factory)
Scope	All container images used in HCP	Subset: AI model images used for Model Serving
Primary Users	DB admins, platform teams, AI teams	AI Factory users, AI engineers
Backed by Container Registry?	Yes	Yes (via Image Library)

次のページに続く

表 2 – 前のページからの続き

Capability	Image and Model Library (Hub)	Model Library (AI Factory)
Supports Private Registries?	Yes	Yes (via Image Library)
Source of Truth	Container Registry → Image Library	Filtered view of Image Library
Usage	Postgres cluster provisioning, AI model serving	AI model serving only
UI Location	Image Library (HCP Portal)	Model Library (AI Factory UI)

概要 → モデルライブラリーはイメージライブラリーを利用しています。 → すべての画像は最初にイメージライブラリーを通過します → モデルライブラリーは、AI モデル画像の使いやすいビューを表示します。

アーキテクチャフロー

```
Container Registry → Image and Model Library →
→ Postgres Provisioning / AI Model Serving
→ Model Library UI → Model Serving
```

- Postgres プロビジョニングは、イメージライブラリーからデータベースイメージを消費します。
- AI Factory Model ライブラリーは、モデル画像を消費します → 画像ライブラリーからキュレーション。

主要な概念 コンセプト 意味 リポジトリ 画像のコレクション例 postgresql、meta/llama-3.3-49b タグイメージのバージョン例 17.4-2503310734、1.8.5、latest HCP タグのイメージリポジトリに適用される組織タグ HCP リポジトリルール接続されたレジストリから同期するリポジトリ/タグを定義プライベートレジストリ HCP ビーコンエージェントに統合された顧客提供のコンテナレジストリイメージメタデータをイメージライブラリーにプルするエージェント

ユースケース データベースイメージ厳選されたデータベースイメージを使用して、新しい Postgres クラスターをプロビジョニングします。

以下のサポート

PostgreSQL コミュニティ。

EDB Postgres Advanced Server 。

EDB Postgres 拡張サーバー PGE。

OCI アノテーションを介して拡張機能を管理します。

AI モデルの画像 AI Factory → Model Serving KServe を使用して AI モデルを展開します。

以下のサポート

NVIDIA NIM モデルのイメージ。

将来 顧客提供のモデルイメージ。

Model ライブラリーは、 Image ライブラリーから AI モデル画像の厳選されたビューを表示します。

ガバナンスフローコンテナレジストリ → プライベートまたはパブリックを接続します。

リポジトリルール → 同期するリポジトリ/タグを定義します。

ビーコンエージェントが画像をプル → 画像ライブラリーが設定されます。

次の目的で使用可能な画像ライブラリー

Postgres プロビジョニングフロー。

モデルライブラリー AI モデル提供用。

モデルライブラリーフィルタ 画像ライブラリー → AI モデルのみが表示されます。

特典

- すべてのイメージに対する単一のガバナンスモデル。
- セキュリティまたはコンプライアンスポリシーのバイパスはありません。
- プライベートレジストリの統合 → データベースと AI モデルイメージに等しく動作します。
- チームは自分自身のリポジトリを管理できますが、プラットフォーム全体の可視性の範囲で動作します。

関連する説明ページ コンテナレジストリ統合の説明

AI ファクトリーモデルの提供の説明 モデルライブラリー

関連するハウツー

- プライベートレジストリの統合
- リポジトリルールの定義
- 画像を見つけて使用する
- AI モデルの展開
- カスタムイメージを使用した Postgres クラスターのプロビジョニング

AI Factory How-To Guides (Hybrid Manager)

これらの **ハウツーガイド** を使用して、 **ハイブリッドマネージャー HCP** 環境内に **AI Factory** 機能を展開および操作します。

Hybrid Manager は、HCP Kubernetes プロジェクト内の統合 AI ワークロードとして AI Factory を実行します。多くのコア概念が *AI Factory Hub* と共有されていますが、次の場合に従う必要がある重要な**ハイブリッドマネージャー固有のパターンと手順** があります。

- モデルサービングインフラストラクチャのセットアップ
- AI モデルの展開と検証 InferenceServices
- モデルサービングの GPU リソースの管理
- Gen AI アプリケーションとデータベース拡張機能での AI モデルの使用
- AI Factory のプロジェクトおよびユーザースコープの権限の管理

詳細はこちら

概念的な背景については、以下を参照してください。

- *AI Factory in Hybrid Manager*
- *AI Factory Hub — Full Concepts*

Setup GPU resources in Hybrid Manager

このガイドを使用して、ハイブリッドマネージャー HCP Kubernetes クラスターの GPU リソースを準備して、KServe でのモデルサービングをサポートします。

GPU 対応ノードをプロビジョニングし、KServe 用に構成し、NVIDIA NIM モデルを展開するために必要なシークレットを保存します。

ゴール

KServe を使用して GPU ベースのモデルサービングワークロードを実行するように HCP クラスターを準備します。

推定時間

20〜40 分クラウドプロバイダーによっては、プロビジョニングにもっと時間がかかる場合があります。

あなたが達成すること

- HCP クラスターに GPU ノードグループまたはノードプールをプロビジョニングします。
- GPU ノードを正しくラベル付けとテイントします。
- NVIDIA デバイスプラグイン DaemonSet を展開します。
- NVIDIA API キーを Kubernetes シークレットとして保存します。
- クラスターが KServe で NIM モデルコンテナを実行できるようにします。

これによりアンロックされるもの

この手順を完了すると、モデル提供を介してサポートされている GPU アクセラレーションモデルを展開できます。

- AIDB ナレッジベースの場合。
- GenAI Builder アシスタントの場合。
- カスタムモデルベースのアプリケーションの場合。

前提条件

- 適切な権限による HCP Kubernetes クラスターへのアクセス。
- プロビジョニングノードグループ AWS EKS / GCP GKE / RHOS への管理アクセス。
- NIM モデルにアクセスするための NVIDIA API キー。

- 基本的な kubectl の使用法に精通していること。

GPU ノードのプロビジョニング

HCP クラスターに GPU ノードグループ EKS またはノードプール GKE / RHOS をプロビジョニングします。

- L40S または A100 GPU たとえば、AWS の g6e.12xlarge または GCP の a2-highgpu-4g を使用したインスタンスを使用します。
- 推奨 Llama 70B などの大規模なモデルをサポートするには、4 つの GPU を備えた少なくとも 1 つのノードをプロビジョニングします。

GPU ノードのラベル付けとテイント

次の Kubernetes ラベルとテイントを GPU ノードに適用します。

レーベル

```
kubectl label node <gpu-node-name> nvidia.com/gpu=true
```` ```


テイント


```shell
kubectl taint nodes <gpu-node-name> nvidia.com/gpu=true:NoSchedule
```

これにより、KServe モデルポッドが正しくスケジュールされ、Postgres クラスターが GPU ノードに着陸しないことが保証されます。

NVIDIA デバイスプラグインを展開する

NVIDIA デバイスプラグイン DaemonSet を展開します。

```
kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.14.1/
↪nvidia-device-plugin.yml
```

プラグインが実行されていることを確認します。

```
kubectl get ds -n kube-system nvidia-device-plugin-daemonset
```

プラグインは、GPU リソースを Kubernetes と KServe に公開します。

NVIDIA API キーを Kubernetes シークレットとして保存する

NGC カタログポータルから NVIDIA API キーを生成します。

Kubernetes シークレットを作成します。

```
kubectl create secret generic nvidia-nim-secrets --from-literal=NGC_API_KEY=<your_
↪NVIDIA_API_KEY>
```

このシークレットは、NIM モデルの `ClusterServingRuntime` リソースを展開するときに必要です。

関連概念

- *GPUs in Model Serving*
- *KServe in AI Factory concepts*
- *AI Factory Learning Paths (Hybrid Manager)*

次のステップ

- *Deploy NIM containers*
- *Verify InferenceServices and GPU Usage in Hybrid Manager*

Verify InferenceServices and GPU Usage in Hybrid Manager

ハイブリッドマネージャーでの推論サービスと GPU の使用状況を検証する

このガイドを使用して、ハイブリッドマネージャー HCP Kubernetes クラスター内の InferenceServices の正しい展開と操作ステータス、および GPU リソースの使用状況を確認します。

ゴール

デプロイされた InferenceServices が GPU リソースを正しく利用していることを確認します。

推定時間

15〜20 分。

あなたが達成すること

- デプロイされた InferenceServices のステータスを確認します。
- GPU リソースの割り当てと使用率を確認します。
- 一般的な展開と GPU 関連の問題のトラブルシューティング。

前提条件

- ハイブリッドマネージャーでの GPU リソースと NIM コンテナのセットアップと展開が完了しました。
- 適切な権限による HCP Kubernetes クラスターへのアクセス。
- 基本的な `kubectl` の使用法に精通していること。

InferenceServices のステータスの検証

デプロイした InferenceServices のステータスを確認して、動作していることを確認します。

```
kubectl get inferenceservice -n <namespace>
```` ``

`READY` ステータスを探して、サービスが正常に実行されていることを確認します。

GPU リソースの使用状況を確認する

ノードでの GPU リソースの割り当てと使用状況を確認します。

```shell
kubectl describe nodes | grep nvidia.com/gpu
```

出力を確認して、GPU の可用性と割り当てを確認します。

GPU 対応ポッド内から nvidia-smi を使用して、リアルタイムの GPU 使用率を確認します。

```
kubectl exec -n <namespace> -it <pod-name> -- nvidia-smi
```

一般的な問題のトラブルシューティング

InferenceService の準備ができていない場合、または GPU リソースが適切に割り当てられていない場合

- NVIDIA デバイスプラグイン DaemonSet が実行されていることを確認します。

```
kubectl get ds -n kube-system nvidia-device-plugin-daemonset
```

- 次を使用して、リソース制約またはスケジュールの問題がないかを確認します。

```
kubectl describe pods -n <namespace>
```

これらのコマンドによって報告されたエラーまたは問題に対処します。

関連概念

- *Setup GPU resources in Hybrid Manager*
- *Deploy NIM containers*
- *KServe in AI Factory concepts*

次のステップ

- GPU リソースの使用率とスケーリングを最適化します。
- モデルのパフォーマンスと状態を監視します。

AI Factory Learning Paths (Hybrid Manager)

これらの **ラーニングパス** に従って、 **ハイブリッドマネージャー**プロジェクト内で **AI Factory** 機能を実行するための実践的なスキルを習得します。

- **

はじめに

AI Factory 101 ハイブリッドマネージャー

- HCP の AI Factory アーキテクチャを理解する
- AI Factory 機能が Kubernetes に展開される方法を理解する
- ハイブリッドマネージャーでのモデルライブラリーとモデル提供を探索する
- ハイブリッドマネージャーで Gen AI Builder を探索する

以下から始めます

- AI Factory in Hybrid Manager Explained
- *Model Library How-To Guides*
- *Model Serving How-To Guides*
- **

中級者

AI Factory 201 ハイブリッドマネージャー

- モデルライブラリーを介してカスタム AI モデルを展開する
- AI モデルサービングの GPU リソースを管理する
- 高度な構成での InferenceServices の展開
- モデル提供の監視とトラブルシューティング
- Gen AI アプリケーションをサービス対象モデルに接続する

推奨されるガイド

- *Integrate Private Container Registry*
- *Deploy NVIDIA NIM Containers*
- *Monitor InferenceServices*
- **

Advanced

AI Factory 301 ハイブリッドマネージャー

- AI ワークロード全体で GPU 使用率を最適化する
- マルチモデル提供パターンを実装する
- モデル説明可能ツールを使用する将来
- モデルエンドポイントを Hybrid Manager Gen AI パイプラインと統合する
- AI Factory 機能のプロジェクトおよびユーザースコープを管理する

次に進みます。

- *Update GPU Resources*
- *Verify InferenceServices and GPU Usage in Hybrid Manager*
- **

詳細はこちら

完全な概念的な背景については

- *AI Factory in Hybrid Manager*
- *AI Factory Hub — Concepts*

トレーニング オプションの場合

- インストラクター主導でマイペースの AI Factory コース *(EDB* **トレーニングカタログを参照してください***)*。

6.2.4 Model capabilities in Hybrid Manager

ハイブリッドマネージャーは、AI Factory ワークロードを介して提供される**モデルサービング** と **モデルライブラリー* の完全なサポートを提供します。

モデルは、Hybrid Manager の Kubernetes インフラストラクチャで KServe を使用して、スケーラブルな**推論サービス** として展開されます。

モデルライブラリーは、サポートされているモデルとコンテナイメージの検出と管理を提供します。

詳細はこちら

- *Model Serving in Hybrid Manager*
- *Asset Library in Hybrid Manager*
- 完全な機能の詳細 *Model Serving in Hybrid Manager* 、 *Asset Library in Hybrid Manager*

GPUs in Model Serving

GPU アクセラレーションは、最新のディープラーニングモデルを実稼働環境で実行するために不可欠です。多くの大規模言語モデル LLM、埋め込みモデル、およびビジョンモデルでは、許容可能な推論パフォーマンスを提供するために GPU が必要です。

AI Factory のモデルサービングは、これらのモデルを実行する KServe InferenceServices をホストするために GPU 対応の Kubernetes ノードに依存しています。

モデルサービスにとって GPU が重要な理由

- 多くの **NVIDIA NIM コンテナ**は、最適化された推論サービスを使用して GPU で実行されるように設計されています。
- AI Factory の Model Serving は、Kubernetes を介して **GPU スケジューリングとリソース制御** をサポートしています。
- GPU ノードは、CPU で実行するには遅すぎるか高価すぎるサービスモデルを有効にします。
- GPU ベースのサービングは、 **AIDB Knowledge Bases** および **GenAI Builder アシスタント** を大規模にサポートしています。

ハイブリッドマネージャーでの GPU 使用率

Hybrid ManagerHCP は、Model Serving が実行される Kubernetes インフラストラクチャを管理します。このコンテキストでは

- GPU 対応ノードグループ AWS EKS またはノードプール GCP GKE、RHOS をプロビジョニングする必要があります。
- **KServe モデルポッド**が適切にスケジュールできるように、これらのノードにラベルを付けて汚染する必要があります。
- GPU リソースを Kubernetes に公開するには、NVIDIA Kubernetes デバイスプラグインをインストールする必要があります。
- Kubernetes シークレットを作成して、NIM モデルに必要な **NVIDIA API キー**を保存する必要があります。

あなたが実行できるアクション

GPU ベースのモデルサービスを有効にするには

1. HCP Kubernetes クラスターに GPU ノードグループをプロビジョニングします。
2. GPU ノードを正しくラベル付けし、テイントします。
3. NVIDIA デバイスプラグイン DaemonSet を展開します。
4. NVIDIA API キーを使用して Kubernetes シークレットを作成します。
5. GPU ノードをターゲットとする ClusterServingRuntime および InferenceService マニフェストを展開します。

関連概念

- Model Serving overview
- *KServe in AI Factory concepts*
- *AI Factory Learning Paths (Hybrid Manager)*
- *Model Serving How-To Guides*

次のステップ

- *Setup GPU resources in Hybrid Manager* に従います。

Asset Library in Hybrid Manager

ハイブリッドマネージャーの **アセットライブラリー**は、プロジェクト内の AI 関連のアセットを管理するための中心的な場所を提供します。これは、以前の**モデルライブラリー** を置き換え、機能を拡張して以下を含みます。

- Model Serving で使用される AI モデル画像。
- 追加の AI アーティファクトレトリバー、埋め込みインデックス、データセットの将来のサポート。

アセットライブラリーを使用して、次のことを行います。

- NVIDIA NIM モデルを検出および管理します。
- プライベートレジストリを統合および管理します。
- モデル画像のメタデータとガバナンスを管理します。
- プロジェクトスコープのモデルの展開とバージョン管理をサポートします。

Hybrid Manager アーキテクチャにどのように適合するか

- アセットライブラリーは、ハイブリッドマネージャーが提供する**イメージライブラリー* をオーバーレイします。
- アセットライブラリーのすべてのモデル画像は、プロジェクトのイメージリポジトリとガバナンスフローによってサポートされています。
- イメージは、同じプロジェクト内の **KServe ベースの Model Serving** エンドポイントに展開できます。
- アセットライブラリーは、以下で使用されるモデルの中心的なガバナンスポイントです。
- Gen AI Builder アシスタント。
- AIDB ナレッジベース埋め込みモデル。
- カスタム AI パイプラインと API。

Asset ライブラリーは、AI Factory のワークロードで使用されるすべてのモデルが制御、監査可能、およびプロジェクトの Kubernetes インフラストラクチャ内で安全に展開されることを保証します。

ハイブリッドマネージャーの考慮事項

- リポジトリ構成は各プロジェクトにスコープされます。プロジェクトごとに異なるプライベートレジストリを構成できます。
- ガバナンスモデルは、イメージ署名、出所チェック、メタデータの追跡を含むハイブリッドマネージャー制御に従います。

一般的な使用例

- 事前統合された NVIDIA NIM モデル LLM、エンベディング、OCR、リランカーを展開します。
- プライベートレジストリから組織固有のモデルを追加および管理します。
- さまざまな AI Factory ワークロードで使用可能なモデルバージョンを制御します。
- メタデータとガバナンスルールを適用して、モデルの使用状況とコンプライアンスを追跡します。

詳細はこちら

- *How to deploy AI models in Hybrid Manager*
- *Manage repository metadata*
- 基になる概念に関する完全なリファレンス *Asset Library in Hybrid Manager*
- *AI Factory Concepts*

Model Serving in Hybrid Manager

ハイブリッドマネージャーの **Model Serving** は、AI モデルをプロダクショングレードの推論サービスとして提供するスケーラブルな Kubernetes ネイティブの方法を提供します。

これは、 **KServe** を使用して実装され、ハイブリッドマネージャープロジェクトの Kubernetes クラスターの GPU 対応ノードで実行されます。Model Serving により、Gen AI アプリケーション、ナレッジベース、カスタムパイプラインは、制御の下で高性能モデルを使用できます。

Model Serving がハイブリッドマネージャーアーキテクチャにどのように適合するか

モデルサービングは、ハイブリッドマネージャーの **AI Factory** ワークロードのコア機能です。

- モデルは、プロジェクトの Kubernetes クラスター内の KServe **InferenceServices** として展開されます。
- Model Serving は、プロビジョニングおよび管理する GPU 対応インフラストラクチャによって駆動されます。
- モデル画像は、ハイブリッドマネージャーのイメージガバナンスに支えられている**アセットライブラリー** 旧モデルライブラリーから取得しています。
- モデルエンドポイント HTTP / gRPC は、次の目的で使用できます。
- Gen AI Builder アシスタント。
- AIDB ナレッジベース。
- 外部アプリケーションと API。

ハイブリッドマネージャーのモデルサービングは、すべてのモデルサービングが管理され、監査可能で、インフラストラクチャ内で安全に実行されることを保証し、**ソブリン AI** パターンを有効にします。

ハイブリッドマネージャーでの仕組み

- KServe は、プロジェクトの Kubernetes クラスター内のハイブリッドマネージャーによってインストールおよび管理されます。
- 高パフォーマンスのモデルサービングをサポートするには、GPU ノードグループまたはノードプールをプロビジョニングする必要があります。
- GPU ノードは正しくラベル付けされ、KServe ワークロードをサポートするように構成されている必要があります。
- モデルは、 ClusterServingRuntime および InferenceService 定義を介して**アセットライブラリー** から展開されます。
- アプリケーションと AI Factory のワークロードは、REST または gRPC を介してモデルエンドポイントを呼び出すことができます。

ハイブリッドマネージャーの主要な考慮事項

- GPU インフラストラクチャは、LLM、エンベディング、ビジョンモデルなどの最も高度なモデルに必要です。
- ハイブリッドマネージャーは、Prometheus メトリックや Kubernetes ネイティブのモニタリングを含むモデルサービングの完全な可観測性を有効にします。
- モデルサービングエンドポイントは、ハイブリッドマネージャープロジェクトスコープ内で保護および管理されます。
- モデルイメージと展開のガバナンスは、ハイブリッドマネージャーの統合された**アセットライブラリー** とイメージコントロールから実現します。

一般的な使用例

- LLM または埋め込みモデルを使用した Power Gen AI Builder アシスタント。
- GPU アクセラレーションの埋め込みパイプラインを使用して AIDB ナレッジベースを有効にします。
- マルチモーダル検索システムの一部として画像モデル OCR、ビジョンを提供します。
- エンタープライズグレードのモデル API をダウンストリームアプリケーションに公開します。

詳細へのリンク

- *Deploying NVIDIA NIM models in Hybrid Manager*
- *Verify InferenceServices and GPU Usage in Hybrid Manager*
- *Model Serving Concepts Hub*
- *Frequently asked questions*
- *AI Factory Concepts*

- *Setup GPU resources in Hybrid Manager*

第7章　Analytics in Hybrid Manager

EDB Hybrid ManagerHM に埋め込まれている分析機能を理解して活用するための中心的なリソースへようこそ。

ハイブリッドマネージャーは、次の統合サポートを提供します。

- Apache Iceberg とカタログサービス
- デルタレイクのクエリー
- EDB Postgres 分散 PGD を使用した階層テーブル
- オブジェクトストレージの高速分析のためのレイクハウスクラスター
- 分析スタックの統合管理と可観測性

これらの機能を使用することにより、運用オーバーヘッドを削減して、Postgres で最新の費用対効果の高い分析アーキテクチャを構築できます。ハイブリッドマネージャーで完全に管理できます。

7.1 ハイブリッドマネージャーでアナリティクスを使用する理由

EDB Hybrid Manager は、分析を統合して次のことを支援します。

- **統合された可視性の獲得** HM が管理する EDB Postgres フリート全体の主要なメトリックと運用傾向を表示します。
- **パフォーマンスの最適化**パフォーマンスのボトルネックを特定し、HM 管理データベースとレイクハウスクラスター内のクエリ動作を理解します。
- **リソースを効率的に管理する**分析ワークロードを含む、HM 管理インスタンスのリソース消費を監視および調整します。
- **オペレーションの状態の強化**分析コンポーネントと基になるデータベースインフラストラクチャの可用性とステータスを追跡します。
- **データアーキテクチャの簡素化** EDB Postgres Lakehouse クラスターのプロビジョニングと管理、PGD での階層テーブルの実装、およびオブジェクトストレージとの統合。すべて HM からです。

7.2 コアコンセプトと機能

7.2.1 Apache Iceberg

ハイブリッドマネージャーでは、Iceberg は次の目的で使用されます。

- Lakehouse クラスターによって照会される、Iceberg 形式で大規模な分析テーブルを管理します。

- Iceberg 形式のオブジェクトストレージへのコールドパーティションの PGD オフロードをサポートします。
- カタログ管理テーブルを介して外部ツール Spark、Trino などとの相互運用性を有効にします。
- HM 管理サービス Lakekeeper または外部カタログを介したカタログ管理を促進します。

Apache Iceberg in Hybrid Manager

次のトピック *Configure an Iceberg REST catalog connection*

7.2.2 デルタレイク

ハイブリッドマネージャーは、Delta Lake テーブルの照会をサポートしています。

- Lakehouse クラスターは、オブジェクトストレージに保存されている既存の Delta Lake テーブルを照会できます。
- HM 管理のレイクハウスをデルタベースのパイプライン Spark、Databricks などと統合します。
- レイクハウスクラスターを介して既存のデルタデータレイクの高速分析を有効にします。

Delta Lake in Hybrid Manager

次のトピック *Query Delta Lake tables*

7.2.3 PGD を使用した階層テーブル

階層テーブルにより、大規模な時系列データセットの効率的な管理が可能になります。

- PGD の AutoPartition は、時間ベースのパーティショニングを自動化します。
- コールドパーティションは、オブジェクトストレージで Iceberg 形式にオフロードされます。
- Lakehouse クラスターは、ホットデータとコールドデータにわたるクエリアクセスを提供します。
- ハイブリッドマネージャーは、階層化テーブルをエンドツーエンドで構成および監視するのに役立ちます。

ハイブリッドマネージャーの階層化テーブル

次のトピック *Configure BDR AutoPartition with analytics offload*

7.2.4 レイクハウスクラスター

Lakehouse クラスターは、HM の管理対象コンピューティングノードです。

- オブジェクトストレージでのスケーラブルなベクトル化されたクエリ実行を提供します。
- Iceberg と Delta Lake の両方のフォーマットをサポートします。
- カタログ HM 管理または外部との統合。
- より広範な HM エコシステム PGD オフロード、階層テーブル、BI 統合との統合。

EDB Postgres Lakehouse Clusters on Hybrid Manager

次のトピック *Create a Lakehouse cluster*

7.3 これらの機能がどのように連携するか

- **データのオフロード** PGD クラスターは、コールドデータを Iceberg 形式のオブジェクトストレージにオフロードします。
- **カタログサービス** HM はカタログと統合して、テーブルのメタデータを管理します。Lakehouse と PGD は、これらのカタログを介して照会します。
- **レイクハウスクラスター**管理対象コンピューティングノードは、オフロードされたデータでスケーラブルな分析クエリを実行します。
- **Delta Lake 統合** Lakehouse クラスターは、インプレースにある既存の Delta Lake テーブルも照会します。
- **統合管理** Hybrid Manager は、分析アーキテクチャをプロビジョニング、構成、監視、および最適化するための単一のコントロールプレーンを提供します。

7.4 はじめに: クイックスタートチェックリスト

このチェックリストに従って、ハイブリッドマネージャー分析機能の適用を開始します。

- *Analytics Concepts in Hybrid Manager* —主要なパターンと機能を理解します。
- *Create a Lakehouse cluster* —Lakehouse クラスターを展開して、オブジェクトストレージでのスケーラブルなクエリーを有効にします。
- *Configure an Iceberg REST catalog connection* —HM 管理または外部カタログを接続して、Iceberg テーブルを管理します。
- *Query existing Apache Iceberg tables* — Lakehouse クラスターから Iceberg テーブルを照会します。
- *Query Delta Lake tables* — レイクハウスクラスターを既存の Delta Lake テーブルに接続します。
- *Configure BDR AutoPartition with analytics offload* — PGD AutoPartition を有効にし、コールドパーティションを Iceberg にオフロードします。
- *Monitor Tiered Tables status and storage savings* — 階層テーブル設定の状態とパフォーマンスを追跡します。
- *Leveraging Analytics in Hybrid Manager* — DBA、DevOps / SRE、データサイエンティスト、および開発者向けに調整された推奨事項。
- *Solving Business Problems with Analytics in Hybrid Manager* —一般的な分析のユースケースとハイブリッドマネージャーがそれらを解決する方法。
- Full How-To Index — 高度な構成と操作のためのステップバイステップガイド。

7.5 さらに探索する

- *Analytics Concepts in Hybrid Manager*
- *Apache Iceberg in Hybrid Manager*
- *Delta Lake in Hybrid Manager*
- **ハイブリッドマネージャーの階層化テーブル**
- *EDB Postgres Lakehouse Clusters on Hybrid Manager*
- *Leveraging Analytics in Hybrid Manager*
- *Solving Business Problems with Analytics in Hybrid Manager*
- *AI Factory How-To Guides (Hybrid Manager)*
- Analytics Tutorials

これらの機能を活用することにより、ハイブリッドマネージャーは、 EDB Postgres 展開全体で最新の柔軟でパフォーマンスの高い分析アーキテクチャを構築できます。すべてが統合された運用効率の高いエクスペリエンスを通じて。

7.5.1 Delta Lake in Hybrid Manager

Delta Lake は、データレイクに ACID トランザクションと信頼性をもたらすオープンソースのテーブル形式です。

Hybrid ManagerHM は、Delta Lake 機能を EDB Postgres 展開に統合し、Lakehouse クラスターがオブジェクトストレージに保存されている Delta Lake テーブルを照会できるようにします。

Delta Lake の一般的な概要については、 *Understanding Delta Lake with EDB Solutions* を参照してください。

ハイブリッドマネージャーで Delta Lake を使用する理由

- **既存のデルタレイクを照会する** 高速な Postgres SQL を使用して、Delta Lake フォーマットで既に構築されているデータレイクを活用します。
- **分析パイプラインの簡素化**不要な ETL を回避します-Postgres から適切な場所に Delta Lake テーブルを照会します。
- **広範なエコシステムの統合** HM 管理の Postgres を、Spark、Trino、Flink、およびその他のツールによって生成された Delta Lake データに接続します。
- **費用対効果の高いレイクハウスアーキテクチャ**大規模なデータセットをオブジェクトストレージに保存し、レイクハウスクラスターを介してクエリします。

重要な用語とアーキテクチャの概要

ハイブリッドマネージャー PGFS、PGAA、レイクハウスクラスター、アナリティクスオフロードなどで使用されるコア分析用語の定義については、*Analytics Concepts in Hybrid Manager* を参照してください。

ハイブリッドマネージャーで Delta Lake を使用するのはどのような場合ですか?

次の場合にハイブリッドマネージャーで Delta Lake を使用します。

- デルタレイクフォーマットで構築された **既存のデータレイクを照会**、ETL またはデータの重複なし。
- **Postgres とデータレイクエコシステムの統合** - Postgres SQL クライアントから Delta Lake テーブルを照会します。
- 運用中の Postgres データとデータレイクデータにわたって**統合分析を有効にします** 。
- 使い慣れた Postgres ツールを使用して、Delta Lake コンテンツの**サポート BI ツールとアドホッククエリー** 。
- 大規模なデルタデータセットでのスケーラブルで高速な SQL のために、**レイクハウスクラスターを活用します** 。

重要: PGAA は現在、Delta Lake テーブルで **読み取り専用クエリー**をサポートしています。PGAA を介したデルタテーブルの書き込みまたは更新はサポートされていません。

ハイブリッドマネージャーの Delta Lake の主要機能

既存の Delta Lake テーブルの照会

内容 オブジェクトストレージに保存されている Delta Lake テーブルで SQL クエリーを実行します。

理由 BI ツールと Postgres ユーザーが既存の Delta Lake データを複製または移動せずに照会できるようにします。

方法 Lakehouse クラスターで PGFS ストレージの場所と PGAA 外部テーブルを定義します。

ここで Delta Lake フォーマット `_delta_log` + Parquet ファイルを使用した S3 互換オブジェクトストレージ。

Query Delta Lake tables

Postgres + Delta Lake の統合の簡素化

内容 HM Lakehouse クラスターを、Spark またはその他のツールで作成した Delta Lake テーブルに接続します。

理由運用システムとデータレイクシステム全体で統合されたレポートと分析を構築します。

方法 Delta Lake パスを指す PGFS ストレージ場所と PGAA リーダーテーブルを作成します。

ここで Delta Lake パイプラインが使用する共有オブジェクトストレージの場所。

Configure PGFS for Delta Lake

統合 SQL ベースの分析のサポート

内容 運用データと Delta Lake データの両方で Postgres SQL クエリーを有効にします。

理由アプリケーション開発者、データサイエンティスト、および BI ユーザーが、複雑なツールを使用せずにデータレイクコンテンツを照会できるようにします。

方法 Lakehouse クラスターで PGAA リーダーテーブルを使用します。オプションで Postgres データと結合します。

ここで オブジェクトストレージの Delta Lake テーブル+ Lakehouse クラスターまたは FDW / dblink 経由の Postgres テーブル。

Query Delta Lake tables

ハイブリッドマネージャーでの Delta Lake のスタートガイド

ハイブリッドマネージャーで Delta Lake の使用を開始するには

1. *Create a Lakehouse cluster* 。
2. Delta Lake オブジェクトストレージを指す *Configure PGFS for Delta Lake* 。
3. Lakehouse クラスターで pgaa 拡張機能を有効にします。
4. Delta Lake パスの PGAA リーダーテーブルを作成します。
5. 標準の Postgres クライアントを使用して Delta Lake テーブルを照会します。

関連するハウツー

- *Query Delta Lake tables*
- *Configure PGFS for Delta Lake*
- *Performance tuning for Delta Lake queries*

次のトピック

ハイブリッドマネージャーの階層化テーブル

7.5.2 Apache Iceberg in Hybrid Manager

Apache Iceberg は、オブジェクトストレージに保存される大規模な分析データセット用のオープンテーブル形式です。

Hybrid ManagerHM は、Iceberg 機能を EDB Postgres 展開に統合し、次のことを可能にします。

- Lakehouse クラスターを介したオブジェクトストレージの効率的なクエリー
- PGD クラスターから Iceberg 形式への構造化データのオフロード
- Iceberg テーブルの集中カタログ管理
- 外部分析エンジン Spark、Trino、Flink などとの相互運用性

Iceberg の一般的な概要については、 *Apache Iceberg with EDB Solutions* を参照してください。

ハイブリッドマネージャーで Iceberg を使用する理由

Apache Iceberg は、大規模なデータセットの分析のためのオープンで信頼性の高いパフォーマンスの基盤を提供します。ハイブリッドマネージャーを介して使用すると、次のことができます。

- ベクトル化された実行を使用して、Lakehouse クラスターから Iceberg テーブルを高いパフォーマンスで照会します。
- トランザクション PGD データを Iceberg 形式にオフロードして、費用効率の高い階層ストレージと長期分析を実現します。
- HM を介して Iceberg カタログを一元管理するか、外部カタログに接続します。
- Iceberg 形式を使用して、Postgres と他のツール Spark、Trino、Flink 間でデータをシームレスに共有します。

重要な用語とアーキテクチャの概要

ハイブリッドマネージャー PGFS、PGAA、レイクハウスクラスター、カタログ、AutoPartition、Analytics Offload などで使用されるコア分析用語の定義については、*Analytics Concepts in Hybrid Manager* を参照してください。

ハイブリッドマネージャーで Iceberg を使用する必要があるのはどのような場合ですか?

次の場合に Iceberg をハイブリッドマネージャーで使用します。

- **費用対効果の高い方法で PGD データをアーカイブし、Postgres またはその他のツールでクエリします。
- **データレイクハウスアーキテクチャ**を HM 管理または外部の Iceberg カタログと統合します。
- ETL を使用しないラージオブジェクトストレージデータセットで**アドホック分析を有効にします** 。
- **階層テーブルを実装**して、PGD で大規模な時系列データセットとストレージライフサイクルを管理します。
- Iceberg 形式を使用して **Postgres と外部データ処理**ツールを統合します。

ハイブリッドマネージャーの Iceberg の主要機能

既存の Iceberg テーブルのクエリー

内容オブジェクトストレージに既に保存されている Iceberg テーブルで SQL クエリーを実行します。

理由 ETL または重複なしで他のツール Spark、Trino、Flink、PGD オフロードで作成されたデータを再利用できます。

方法 Lakehouse クラスターを使用して、Iceberg データを指す PGAA 外部テーブルを定義します。

ここでファイルベースまたはカタログ管理の S3 互換オブジェクトストレージの Iceberg テーブル。

Query existing Apache Iceberg tables

Iceberg カタログの統合

内容 Lakehouse および PGD ノードを Iceberg カタログ HM 管理または外部に接続します。

理由 Iceberg テーブルのメタデータを一元管理し、一貫したセマンティクスでクエリを実行します。

方法 PGAA ファンクション `pgaa.add_catalog` 、`pgaa.attach_catalog` を使用して、カタログを接続およびインポートします。

ここで HM 管理の Lakekeeper カタログ、またはサードパーティのカタログ Glue、Nessie、Polaris。

How-To: Configure an Iceberg REST catalog connection

注 階層テーブル または **PGD オフロード**を使用する場合、相互運用性とライフサイクル管理のためにカタログ管理の Iceberg テーブルを使用することを強くお勧めします。

PGD データの Iceberg へのオフロード

内容 PGD トランザクションデータを分析ストレージ用に Iceberg 形式にオフロードします。

理由データのクエリー可能を維持しながら、ストレージコストを最適化し、PGD 操作負荷を削減します。

方法 PGFS の場所および/または Iceberg カタログを構成し、PGD テーブルで分析レプリケーションを有効にします。

ここでオブジェクトストレージにオフロードされた Iceberg テーブル、Lakehouse または他のツールでクエリ可能。

Offload PGD data to Apache Iceberg

ハイブリッドマネージャーでの Iceberg のユースケース

- クエリアクセスを維持しながら PGD データを Iceberg 形式でアーカイブする
- HM 管理カタログを使用した一元化データレイクハウスアーキテクチャ
- Lakehouse クラスターを介した大規模なデータのアドホック分析
- PGD データセットの階層化ストレージとライフサイクル管理の実装
- Iceberg 形式を使用して PGD オフロードデータを Spark、Trino、Flink パイプラインに公開する

How-To: Iceberg use cases in Hybrid Manager

ハイブリッドマネージャーでの Iceberg のスタートガイド

ハイブリッドマネージャーで Iceberg の使用を開始するには

1. *Create a Lakehouse cluster* 。
2. *Configure an Iceberg REST catalog connection* HM 管理または外部。
3. PGD データをオフロードする場合、 *Offload PGD data to Apache Iceberg* 。
4. 既存の Iceberg テーブルを照会する場合、 *Query existing Apache Iceberg tables* 。
5. 標準の Postgres クライアントを使用して分析クエリを実行します。

関連するハウツー

- *Configure an Iceberg REST catalog connection*
- *Query existing Iceberg tables*
- *Offload PGD data to Apache Iceberg*

次のトピック

Delta Lake in Hybrid Manager

7.5.3 EDB Postgres Lakehouse Clusters on Hybrid Manager

EDB Postgres Lakehouse クラスターは、オブジェクトストレージでオープンレイクハウステーブルフォーマット Apache Iceberg、Delta Lake を照会するための高速分析コンピューティングを提供します。

Hybrid ManagerHM を使用すると、UI と API を介してレイクハウスクラスターをプロビジョニングおよび管理し、既存の PGD クラスターおよびより広範な分析エコシステムと統合できます。

EDB Postgres Lakehouse アーキテクチャの基本的な概念については、*EDB Postgres Lakehouse: An Overview* を参照してください。

ハイブリッドマネージャーで Lakehouse クラスターを使用する理由

- **マネージド分析コンピューティング** HM でレイクハウスクラスターをプロビジョニングして、オブジェクトストレージで高速クエリーを実行します。
- **コンピューティングとストレージの分離**データをオブジェクトストレージに保持し、必要に応じて分析コンピューティングをスケーリングします。
- **PGD との統合** PGD 階層テーブルおよびオフロードデータでレイクハウスクラスターを使用します。
- **相互運用性** Lakehouse クラスターは、他の分析エンジンと共有している Iceberg および Delta Lake テーブルを照会できます。
- **使いやすい Postgres インターフェイス** EDB Postgres Lakehouse クラスターで標準の SQL クライアントとツールを使用します。

重要な用語とアーキテクチャの概要

ハイブリッドマネージャー PGFS、PGAA、レイクハウスクラスター、アナリティクスオフロードなどで使用されるコア分析用語の定義については、*Analytics Concepts in Hybrid Manager* を参照してください。

ハイブリッドマネージャーでレイクハウスクラスターを使用する必要がある場合

次の場合にハイブリッドマネージャーでレイクハウスクラスターを使用します。

- オブジェクトストレージに保存されている大規模なデータセットで**高速でスケーラブルな分析** を実行します。
- **PGD 階層テーブル**またはオフロードされた PGD トランザクションデータからのクエリデータ。

- **Postgres を外部データレイクと統合します** Iceberg または Delta Lake に構築されています。
- データレイクコンテンツへの Postgres SQL アクセスで、**サポート BI ツールとアドホックユーザー** 。
- **分析コンピューティングをトランザクションデータベースとは別に管理** - 必要な場合にのみスケーリング。

ハイブリッドマネージャーのレイクハウスクラスターの主要な機能

オブジェクトストレージでの高速クエリの実行

内容 オブジェクトストレージに保存されている Apache Iceberg および Delta Lake テーブルで分析クエリを実行します。

理由大規模なデータセットを Postgres ストレージに移動せずに、高速でスケーラブルな分析を有効にします。

方法 HM でレイクハウスクラスターをプロビジョニングし、ベクトル化されたクエリ実行で PGAA を使用してデータをクエリします。

ここで PGFS を介して接続された S3 互換オブジェクトストレージバケット。

Query existing Apache Iceberg tables

Query Delta Lake tables

PGD 階層化テーブルとオフロードのサポート

内容 PGD オフロードおよび階層化データのクエリターゲットとしてレイクハウスクラスターを使用します。

理由履歴データの照会可能な状態を維持しながら、PGD 運用ストレージを最適化します。

方法 Iceberg 形式のオブジェクトストレージへの分析レプリケーション用に PGD ノードグループを構成します。

ここでレイクハウスクラスターを介して照会された、Iceberg のオフロード PGD データ。

Offload PGD data to Apache Iceberg

Query Tiered Tables from PGD and Lakehouse

外部データレイクのクエリ

内容 他のツール Spark、Trino、Flink で作成された Iceberg または Delta Lake テーブルを照会します。

理由 Postgres から外部データレイクを照会することにより、データの重複と ETL を回避します。

方法 Lakehouse クラスターを外部の Iceberg カタログに、または Delta Lake ストレージに直接接続します。

ここでオプションのカタログ統合を備えた S3 互換オブジェクトストレージ。

Configure an Iceberg REST catalog connection

Query existing Apache Iceberg tables

Query Delta Lake tables

分析コンピューティングの集中管理

内容 PGD クラスターと並行して HM で分析クラスターをプロビジョニング、監視、および管理します。

理由分析コンピューティングとトランザクションコンピューティングを 1 つのコントロールプレーンで管理します。

方法 HM UI または API を使用して、Lakehouse クラスターを作成および構成します。

ここで Lakehouse クラスターは、HM が管理するクラウド環境で実行されます。

Create a Lakehouse cluster

ハイブリッドマネージャーでのレイクハウスクラスターの入門

ハイブリッドマネージャーで Lakehouse クラスターの使用を開始するには

1. *Create a Lakehouse cluster* 。
2. オブジェクトストレージを指す *Configure PGFS for Delta Lake* 。
3. オプショナル *Configure an Iceberg REST catalog connection* 。
4. 階層化テーブルを使用するオプションの *Offload PGD data to Apache Iceberg* 。
5. Iceberg/Delta データの Lakehouse テーブルまたは外部リーダーを作成します。
6. Postgres クライアントを使用してデータを照会します。

関連するハウツー

- *Create a Lakehouse cluster*
- *Configure an Iceberg REST catalog connection*
- *Query existing Apache Iceberg tables*
- *Query Delta Lake tables*
- *Offload PGD data to Apache Iceberg*
- *Query Tiered Tables from PGD and Lakehouse*

次のトピック

Apache Iceberg in Hybrid Manager

7.5.4 Learning Analytics in Hybrid Manager

このセクションを使用して、EDB Hybrid ManagerHM 分析機能を実際のワークロードに適用する方法を学習します。

ハイブリッドマネージャーを使用すると、Postgres 展開全体で高度な分析機能を管理および操作できます。

- オブジェクトストレージでの高速分析クエリのための**レイクハウスクラスター**

- 最新のデータレイクアーキテクチャのための **Apache Iceberg および Delta Lake** の統合
- 大規模な時系列データを費用対効果の高い管理のための **階層テーブル**
- 単一のコントロールプレーンを介した運用データと分析データの両方の統合管理

ビジネス上の問題の解決

一般的な分析のユースケースと、ハイブリッドマネージャー機能を使用してそれらを解決する方法を参照してください。

- *Solving Analytics Problems in Hybrid Manager*

次のトピック *Persona-Based Guide to Using Analytics in Hybrid Manager*

ハウツーガイド

Hybrid Manager 分析機能を構成および使用するための詳細な手順。

- *How-To Guides for Analytics in Hybrid Manager*

次のトピック Analytics Tutorials

次のステップ

- ** *Analytics Concepts in Hybrid Manager* ** を確認して、コアアーキテクチャとパターンを理解することから始めます。
- ** *Solving Problems* ** を探索して、ビジネスニーズに機能を一致させます。
- ** *AI Factory How-To Guides (Hybrid Manager)* ** を使用して、システムを構成および操作します。
- より深い実践学習のために、 ** Analytics Tutorials ** に進みます。

さらに探索する

- *Analytics in Hybrid Manager*
- *Working with Apache Iceberg in Hybrid Manager*
- *Working with Delta Lake in Hybrid Manager*
- *Implementing Tiered Tables in Hybrid Manager*
- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**

Analytics Concepts in Hybrid Manager

Hybrid Manager を使用すると、 EDB Postgres の最新のデータレイクハウスと分析ソリューションを構築できます。

このページでは、ハイブリッドマネージャーで適用される主要な概念を紹介します。背景に Analytics Hub の詳細な説明へのリンクがあります。

アーキテクチャの概要

ハイブリッドマネージャーは、次のレイヤーを統合します。

- スケーラブルな分析コンピューティングのための**レイクハウスクラスター**
- コスト効率の高い分析データストレージのための **Object Storage**
- トランザクションワークロードと自動階層テーブルオフロード用の **PGD クラスター**
- メタデータと相互運用性のための **カタログサービス** HM 管理または外部
- **PGAA および PGFS** が Postgres をデータレイクにブリッジする

このアーキテクチャにより、Postgres ファーストとマルチエンジンのデータレイクハウスデザインの両方が可能になります。

一般的な概要については、 *What is a Postgres Lakehouse* を参照してください。

ハイブリッドマネージャーの##レイクハウスクラスター

Lakehouse クラスターは、ハイブリッドマネージャーでスケーラブルな分析コンピューティングを提供します。

- HM を介してプロビジョニングおよび管理
- ベクトル化されたクエリ実行のための PGAA 拡張機能を装備
- オブジェクトストレージ Iceberg、Delta Lake のデータを照会するように設計
- 外部ツール Spark、Trino との相互運用性

レイクハウスクラスターは、ハイブリッドマネージャーの分析の中心です。

詳細はこちら EDB Postgres Lakehouse: An Overview

ハイブリッドマネージャーの## Apache Iceberg

Hybrid Manager は、オフロードデータと分析データのプライマリテーブル形式として Iceberg を統合します。

- PGD クラスターは、階層化テーブルの Iceberg 形式にデータをオフロードします
- Lakehouse クラスターは Iceberg テーブルを効率的に照会します
- カタログサービス Lakekeeper または外部は、Iceberg テーブルのメタデータを管理します

Iceberg は、HM のレイクハウスアーキテクチャのためのオープンで信頼性の高い基盤を提供します。

概念の概要 *Apache Iceberg with EDB Solutions*

ハイブリッドマネージャーの## Delta Lake

Delta Lake は、さらなる柔軟性を提供します。

- Lakehouse クラスターは、オブジェクトストレージの既存の Delta Lake テーブルを照会できます
- Spark / Databricks パイプラインを使用する環境で一般的な
- HM の Delta Lake のサポートは現在読み取り専用です

ハイブリッドマネージャーは、ETL なしで Delta Lake データの Postgres ベースの分析を有効にします。

概念の概要 *Understanding Delta Lake with EDB Solutions*

ハイブリッドマネージャーの階層化テーブル

階層テーブルは、大規模な時系列または履歴データセットの自動ライフサイクル管理を提供します。

- PGD クラスターは BDR AutoPartition を使用してパーティションを管理します
- 古いパーティションは、Iceberg 形式のオブジェクトストレージに自動的にオフロードされます
- Lakehouse クラスターと PGD 親テーブルは、ホットデータとコールドデータの両方を照会できます

階層テーブルは、ハイブリッドマネージャーでトランザクションワークロードと分析ワークロードを結合するための重要な最適化パターンです。

概念の概要 *Understanding Tiered Tables with EDB Postgres*

ハイブリッドマネージャーの PGAA および PGFS

PGAA Postgres 分析アプライアンス

PGAA 拡張機能は以下を提供します。

- ベクトル化されたクエリーの実行
- PGFS を介したオブジェクトストレージとの統合
- Iceberg および Delta Lake テーブルの照会のサポート
- カタログの統合

PGAA は、レイクハウスクラスターと PGD オフロードパイプラインの両方を強化します。

PGFS Postgres ファイルシステム

PGFS はオブジェクト格納場所を定義します。

- PGAA がオブジェクトストレージのデータの読み取り/書き込みに使用します
- S3 互換ストレージ、GCS などをサポート
- PGD クラスターと Lakehouse クラスターの両方で構成

PGFS は、データレイク統合のためのシンプルですが重要なビルディングブロックです。

これらの概念がハイブリッドマネージャーでどのように適合するか

Layer	Role
PGD Clusters	Transactional layer, source for Tiered Tables
Lakehouse Clusters	Analytical compute, queries across Iceberg/Delta and Postgres
Object Storage	Cost-efficient analytical data storage
Catalog Services	Centralized metadata for Iceberg tables

次のページに続く

表 1 – 前のページからの続き

Layer	Role
PGAA + PGFS	Bridge between Postgres and object storage

次のトピック

Solving Business Problems with Analytics in Hybrid Manager

How-To Guides for Analytics in Hybrid Manager

このセクションでは、ハイブリッドマネージャー HM 環境内でコア分析機能を構成および使用するためのステップバイステップガイドを提供します。

- 高度な分析クエリのための **レイクハウスクラスター** の展開と管理
- **Apache Iceberg** テーブルの構成とクエリー
- **Delta Lake** テーブルの構成とクエリー
- EDB Postgres DistributedPGD を使用した **階層テーブル** の実装と管理

概念的な背景については、 *Analytics Concepts in Hybrid Manager* を参照してください。

Apache Iceberg での作業

- *Configure an Iceberg REST catalog connection*
- *Query existing Apache Iceberg tables*
- *Offload PGD data to Apache Iceberg*

次へ Working with Delta Lake

階層化テーブルの実装

階層化テーブルの構成

- *Configure PGFS storage for Tiered Tables*
- *Configure PGD node group for analytics offload*
- *Configure BDR AutoPartition with analytics offload*

階層化テーブルの監視と操作

- *Monitor Tiered Tables status and storage savings*
- *Query Tiered Tables from PGD and Lakehouse*

次へ *Solving Business Problems with Analytics in Hybrid Manager*

これらのハウツーガイドを使用して、ハイブリッドマネージャーの高度な分析機能を構成および操作します。機能をビジネスニーズに合わせるには、*Solving Business Problems with Analytics in Hybrid Manager* を

参照してください。ロール固有のガイダンスについては、*Leveraging Analytics in Hybrid Manager* にアクセスしてください。

Configure PGFS for Delta Lake

このガイドでは、ハイブリッドマネージャー HM レイクハウスクラスターで PGFS EDB Postgres ファイルシステムストレージの場所を構成し、オブジェクトストレージに保存されている Delta Lake テーブルの照会を有効にする方法について説明します。

Delta Lake テーブルをポイントする PGAA 外部テーブルを作成する前に、PGFS ロケーションを構成する必要があります。

HM での Delta Lake 統合の背景については、次を参照してください。*Delta Lake in Hybrid Manager*

前提条件

デルタレイクの PGFS を構成する前に

- ハイブリッドマネージャーで Lakehouse クラスターをプロビジョニングしている。まだ完了していない場合は、次を参照してください。 *Create a Lakehouse cluster*
- S3 互換のオブジェクトストレージバケットに保存されている既存の Delta Lake テーブルがある。バケットには次の両方を含める必要があります。
- 寄木細工のデータファイル
- Delta Lake トランザクションログが含まれる`_delta_log` ディレクトリ
- バケットのアクセス資格情報があるプライベートの場合。
- SQL クライアントが Lakehouse クラスターに接続されています。
- psql
- DBeaver
- pgAdmin
- Postgres 互換クライアント

ステップ

ステップ 1PGFS ストレージの場所を作成する

`pgfs.create_storage_location()` を実行して、オブジェクトの保存場所を登録します。

必要な数の PGFS ロケーションを、バケットごとにまたはデルタテーブルの論理グループごとに 1 つ作成できます。

例 パブリックバケット

SELECT pgfs.create_storage_location(name => ‘my_public_delta_lake_store’, url => ‘s3://my-public-delta-data/’, options => ‘{“aws_skip_signature”: “true”}’, credentials => ‘{ }’);

例 プライベートバケット AWS S3

SELECT pgfs.create_storage_location(name => ‘my_private_delta_lake_assets’、 url => ‘s3://my-private-delta-data/’、options => ‘{}’、 credentials => ‘{“access_key_id”: “YOUR_AWS_ACCESS_KEY_ID”, “secret_access_key”: “YOUR_AWS_SECRET_ACCESS_KEY”}’);

ステップ 2PGFS の場所を検証する

次を実行して、すべての PGFS ストレージ場所をリストします。

SELECT * FROM pgfs.list_storage_locations();

新しい場所 `my_private_delta_lake_assets` などリストに表示されることを確認します。

注

- PGFS 格納場所は、PGAA 外部テーブル定義で使用できます。
- 複数の PGFS の場所を定義できます。Delta Lake テーブルがバケット全体に分散されている場合に役立ちます。
- ストレージ場所は、S3 互換オブジェクトストアをポイントできます。
- AWS S3
- Google Cloud Storage
- Azure ストレージ
- MinIO
- その他の互換性のあるシステム
- 階層テーブルの PGFS を構成する場合、次を参照してください。 *Configure PGFS storage for Tiered Tables*

次のステップ

これで、PGFS ストレージの場所の構成が完了しました。

- PGAA リーダーテーブルを使用して、Lakehouse クラスターから **既存の Delta Lake テーブルを照会**できます。 *Query Delta Lake tables*
- 大規模な Delta Lake テーブルで **クエリパフォーマンスを調整**して、ベクトル化されたクエリ効率を最大化できます。 *Performance tuning for Delta Lake queries*
- **ハイブリッドマネージャーで Delta Lake の概念とアーキテクチャを確認して、これが分析スタックにどのように適合するかを理解できます。*Delta Lake in Hybrid Manager*

Configure an Iceberg REST catalog connection

このガイドでは、ハイブリッドマネージャー HM で Iceberg REST カタログ接続を構成する方法について説明します。

Lakehouse クラスターと PGD ノードを Iceberg カタログに接続して、Iceberg テーブルを管理および照会できます。

ゴール

このガイドを完了すると、次のことができるようになります。

- Lakehouse クラスターまたは PGD ノードを Iceberg REST カタログに接続する
- クエリーまたは PGD オフロードにカタログ管理の Iceberg テーブルを使用する
- カタログに新しい Iceberg テーブルを作成する
- カタログメタデータを Postgres に更新

始める前に

最初にこれらのトピックを確認してください。

- *Apache Iceberg in Hybrid Manager*
- *EDB Postgres Lakehouse Clusters on Hybrid Manager*
- *Offload PGD data to Apache Iceberg*

前提条件

- ハイブリッドマネージャーでプロビジョニングされた Lakehouse クラスターまたは PGD ノード
- Iceberg REST カタログが利用できます。
- HM 管理の Lakekeeper カタログ、または
- 外部カタログ AWS Glue、Project Nessie、Snowflake Polaris など
- カタログの資格情報 REST エンドポイント URL、必要に応じてトークン、ウェアハウス ID
- ターゲットノード Lakehouse クラスターまたは PGD ノードで有効になっている PGAA 拡張機能

ステップ

1. カタログ接続を追加します

pgaa.add_catalog() を実行して、カタログを登録します。

例 総合外部カタログ

```
SELECT pgaa.add_catalog(
your_catalog_alias,
iceberg-rest,
{
"url": "https://your_catalog_rest_endpoint",
"token": "your_catalog_access_token",
"warehouse": "your_catalog_warehouse_id",
"danger_accept_invalid_certs": "true"
}
);
```

例 HM 管理の Lakekeeper カタログ

```
SELECT pgaa.add_catalog(
hm_lakekeeper_main,
iceberg-rest,
{
"url": "https://hm.example.com/catalog/v1",
"token": "your_hm_api_key",
"warehouse": "lakehouse_warehouse_1"
}
);
```

2. カタログを添付します

カタログを追加した後、 pgaa.attach_catalog() を実行します。

```
SELECT pgaa.attach_catalog(your_catalog_alias);
```

これにより、Postgres でカタログテーブルが表示および照会可能になります。

3. インポートカタログオプショナル

テーブル定義を更新するには

```
SELECT pgaa.import_catalog(your_catalog_alias);
```

新しいテーブルが外部で作成された後、またはカタログコンテンツが変更された場合、これを実行します。

4. カタログ管理の Iceberg テーブルの作成

カタログが追加してアタッチされると、カタログによって管理される新しい Iceberg テーブルを作成できます。

例

```
CREATE TABLE public.catalog_managed_sensor_data (
    device_id TEXT,
    event_time TIMESTAMP WITH TIME ZONE,
    temperature FLOAT,
    humidity FLOAT
)
USING PGAA
WITH (
    pgaa.format = iceberg,
    pgaa.managed_by = your_catalog_alias,
    pgaa.catalog_namespace = iot_data,
    pgaa.catalog_table = hourly_sensor_readings
);
```

(次のページに続く)

(前のページからの続き)

```
pgaa.catalog_namespace = iot_data,
pgaa.catalog_table = hourly_sensor_readings
);
```

- pgaa.manched_by pgaa.add_catalog() で使用されるカタログエイリアス。
- pgaa.catalog_namespace カタログ内のターゲット名前空間。
- pgaa.catalog_table カタログ内のターゲットテーブル名。

注

- 接続したら、 SELECT * FROM information_schema.tables を実行して、カタログ管理テーブルをリストできます。
- Iceberg への PGD オフロードを使用している場合、最初にカタログを接続する必要があります。*Offload PGD data to Apache Iceberg* を参照してください。
- pgaa.add_catalog() を介して追加されたカタログは、一貫したメタデータアクセスのために Lakehouse クラスターと PGD ノード間で共有できます。

これによりできること

Iceberg カタログ接続を構成した後

- Lakehouse クラスターまたは PGD ノードからカタログによって管理されている Iceberg テーブルを照会できます
- PGD データをカタログ管理の Iceberg テーブルにオフロードできます
- Postgres から新しいカタログ管理の Iceberg テーブルを作成できます
- カタログメタデータを更新して Postgres 環境に同期できます

次のステップ

これで、Iceberg カタログ接続の構成が完了しました。

- **既存の Iceberg テーブルを照会**して、Postgres で外部データにアクセスできるようにできます。*Query existing Apache Iceberg tables*
- **PGD データをこのカタログにオフロード**して、運用データのコスト効率の高いストレージと高速分析を有効にすることができます。*Offload PGD data to Apache Iceberg*
- **新しいカタログ管理の Iceberg テーブル**を作成して、Postgres と外部エンジンの両方の分析データを構造化できます。*4.* **カタログ管理の** *Iceberg* **テーブルの作成** を参照
- Iceberg とカタログがハイブリッドマネージャーアーキテクチャにどのように適合するかに関する** コアコンセプトを確認できます。*Apache Iceberg in Hybrid Manager*

Configure BDR AutoPartition with analytics offload

階層テーブルを実装するには、PGD テーブルで **BDR AutoPartition** を構成する必要があります。

これにより、次のことが可能になります。

- テーブルの自動時間ベースのパーティショニング
- 古いパーティションの Apache Iceberg 形式のオブジェクトストレージへの自動オフロード

制御できます

- 新しいパーティションが作成される頻度
- パーティションをオフロードするまでの経過日数 analytics_offload_period

階層テーブルの仕組みの背景については、次を参照してください。**ハイブリッドマネージャーの階層化テーブル**

前提条件

AutoPartition を構成する前に

- 分析オフロード用に PGD ノードグループを構成しました。 *Configure PGD node group for analytics offload*
- PGFS ストレージの場所を作成している、および/または Iceberg カタログを構成している。*Configure PGFS storage for Tiered Tables*
- ターゲットテーブルを作成し、時間ベースの列たとえば `event_timestamp` によってパーティション化されています。

ステップ

1. パーティション分割テーブルを作成しますまだ作成していない場合。

例

```
CREATE TABLE public.application_logs (
log_id BIGSERIAL,
event_timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
application_name TEXT,
log_level VARCHAR(10),
message TEXT,
PRIMARY KEY (log_id, event_timestamp)
) PARTITION BY RANGE (event_timestamp);
```

2. テーブルの AutoPartition を構成します。

例

```
SELECT bdr.autopartition (
relation := public.application_logs,
```

(次のページに続く)

(前のページからの続き)

```
partition_increment := 1 month,
partition_initial_lowerbound := 2023-01-01 00:00:00,
managed_locally := TRUE,
analytics_offload_period := 6 months
);
```

- partition_increment — 新しいパーティションを作成する頻度。
- partition_initial_lowerbound — パーティション作成の開始日。
- analytics_offload_period —パーティションをオフロードするための年齢しきい値たとえば 6 か月。

3. AutoPartition 構成を確認します。

```
SELECT * FROM bdr.partition_config WHERE relid = public.application_logs::regclass;
```

パーティショニングポリシーと analytics_offload_period が表示されます。

4. パーティションが作成されていることを確認します。

```
SELECT relname, relkind FROM pg_class WHERE relname LIKE application_logs_part_%;
```

5. AutoPartition タスクを監視します。

```
wqs.workid, wis.ap_wi_started_at, wis.ap_wi_finished_at,
wis.ap_wi_status, "partition", sql
FROM bdr.taskmgr_local_workitem_status wis
RIGHT JOIN bdr.taskmgr_local_work_queue_status wqs
ON wis.ap_wi_workid = wqs.workid
WHERE relname = public.application_logs AND sql NOT LIKE %bdr.autopartition_create_
↪partition%
ORDER BY ap_wi_finished_at DESC;
```

注

- AutoPartition は、新しいパーティションを自動的に管理し、パーティションが対象になるとオフロードをトリガーします。
- 新しい値で bdr.autopartition() を再度実行することにより、いつでも**analytics_offload_period を調整できます。
- 必要に応じて、テーブルで AutoPartition を **一時停止/再開** できます。

次のステップ

これで AutoPartition の構成が完了しました。

- **階層テーブルのオフロードを監視し、ストレージ節約を検証できます** *Monitor Tiered Tables status and storage savings*

- **階層テーブルを照会できます** 。PGD 親テーブルには、ホットローカルおよびコールドオフロードパーティションが透過的に含まれます *Query Tiered Tables from PGD and Lakehouse*
- 必要に応じて、**PGD オフロードポリシー** ができます。 *Configure PGD node group for analytics offload*
- 必要に応じて、**Iceberg のストレージ場所** を最適化できます。*Configure PGFS storage for Tiered Tables*

関連概念

アーキテクチャの概要と階層テーブルの背後にある概念については、次を参照してください。**ハイブリッドマネージャーの階層化テーブル**

Configure PGD node group for analytics offload

PGFS ストレージの場所を構成した後、階層テーブルの分析データをオフロードするように **PGD ノードグループ** を構成する必要があります。

次のことを選択できます。

- **カタログなしのオフロード** 簡単なセットアップ - ファイルシステムベースの Iceberg テーブル、または
- **Iceberg カタログへのオフロード** 推奨 - 外部ツールとの相互運用性を有効にします

オフロード用に PGD ノードグループを構成する理由

これを構成すると

- BDR AutoPartition によって作成された**コールドパーティションの自動オフロード** を有効にしている。
- オフロードデータの保存場所を制御できます生のファイルシステムパスまたはカタログ管理の Iceberg テーブルとして。
- この手順は、完全な **階層テーブル** 動作を実装するために必要です。

背景については、次を参照してください。 **ハイブリッドマネージャーの階層化テーブル**

前提条件

オフロード用に PGD ノードグループを構成する前に

- ハイブリッドマネージャーでプロビジョニングされた **PGD クラスター**があります。
- PGFS ストレージの場所 *Configure PGFS storage for Tiered Tables* を作成しました。
- (オプション) Iceberg カタログ接続推奨を構成しました。 *Configure an Iceberg REST catalog connection*

カタログなしのオフロード

PGD を生の Iceberg ファイルシステムパスに直接オフロードする場合

ステップ 1PGD ノードグループを特定する

SELECT node_group_name FROM bdr.node_group;

結果の例 edb_pgdx_node_group_1

ステップ 2 ノードグループオプション analytics_storage_location を設定する

```
SELECT bdr.alter_node_group_option(
edb_pgdx_node_group_1,
analytics_storage_location,
hm_tiered_analytics_store
);
```

- 'hm_tiered_analytics_store' を PGFS 保存場所の名前に置き換えます。

結果

- PGD は、対象となるパーティションをこの場所にあるファイルシステムベースの Iceberg テーブルにオフロードします。

Iceberg カタログを使用したオフロード推奨

PGD を **Iceberg カタログ管理テーブル** にオフロードする場合

ステップ 1 カタログが構成および接続されていることを確認します

```
SELECT pgaa.add_catalog(
hm_main_lakekeeper,
iceberg-rest,
{
"url": "https://hm.example.com/catalog/v1",
"token": "your_hm_api_key",
"warehouse": "lakehouse_warehouse_1"
}
);

SELECT pgaa.attach_catalog(hm_main_lakekeeper);
```

ステップ 2 ノードグループオプション analytics_write_catalog を設定する

```
SELECT bdr.alter_node_group_option(
edb_pgdx_node_group_1,
analytics_write_catalog,
hm_main_lakekeeper
);
```

- 'hm_main_lakekeeper' を、接続したカタログの名前に置き換えます。

結果

- PGD は、対象となるパーティションを**このカタログによって管理される Iceberg テーブル**としてオフロードします。

構成を検証する

これを実行して確認します。

```
SELECT * FROM bdr.node_group_option WHERE node_group_name = edb_pgdx_node_group_1;
```

いずれか が表示されます。

- `analytics_storage_location` セットカタログなし、または
- `analytics_write_catalog` セットカタログを使用します。

注

- `analytics_storage_location` **または** `analytics_write_catalog` の **いずれか** 両方ではなく設定する必要があります。
- オフロードデータを次の方法で照会可能にしたい場合は、カタログを使用することを強くお勧めします。
- **レイクハウスクラスター**
- **外部分析エンジン** Spark、Trino など
- カタログを使用しない場合、オフロードされたテーブルは PGFS の場所のファイルシステムパスに書き込まれます。内部使用に適していますが、相互運用が困難です。

次のステップ

これで、PGD ノードグループオフロードの構成が完了しました。

- **オートパーティションを構成して、オフロードのパーティショニングとトリガーを開始できます** *Configure BDR AutoPartition with analytics offload*
- **階層テーブルのオフロードとストレージ削減をモニタできます** *Monitor Tiered Tables status and storage savings*
- **PGD 親テーブルまたは Lakehouse から階層テーブルを照会できます** *Query Tiered Tables from PGD and Lakehouse*
- 後で保存場所またはカタログを調整する必要がある場合は、`bdr.alter_node_group_option()` を再度実行するだけです。

関連概念

これが階層テーブルフローにどのように適合するかのアーキテクチャビューについては、次を参照してください。ハイブリッドマネージャーの階層化テーブル

Configure PGFS storage for Tiered Tables

階層テーブルを有効にするには、PGD クラスターで **PGFS ストレージの場所**を構成する必要があります。これは、PGD が「コールド」データ Iceberg テーブルとしてオブジェクトストレージにオフロードする場所を定義します。

S3 互換オブジェクトストア AWS S3、Google Cloud Storage、または互換性のあるプライベートストレージを使用できます。

階層化テーブルの PGFS を構成する理由

これを構成すると

- 階層テーブルによって管理される **パーティションの分析オフロード**の宛先を PGD に指定します。
- **レイクハウスクラスター**または PGD 親テーブルを使用して、コールドデータを透過的に照会する機能のロックを解除します。
- これは、オフロード用に PGD ノードグループを構成し、AutoPartition を有効にする前に最初に必要な手順です。

背景については、次を参照してください。 ハイブリッドマネージャーの階層化テーブル

ゴール

このハウツーを完了すると、次のことができるようになります。

- オブジェクトストレージバケットを指す有効な PGFS ストレージの場所を作成します。
- PGD クラスターがオフロードにこの場所を使用できることを確認します。
- PGD ノードグループのオフロード設定の構成に進みます。

前提条件

始める前に

- ハイブリッドマネージャーでプロビジョニングされた **PGD クラスター**があります。
- **オブジェクトストレージバケット** が作成されました。
- バケットの **アクセス資格情報** があることプライベートの場合。
- `psql` または SQL クライアントを使用して PGD ノードに接続している。

ステップ

ステップ 1PGD ノードに接続する

次を使用して接続します。

- psql
- DBeaver
- pgAdmin

- PGD ノードをサポートする SQL クライアント。

ステップ 2 pgfs.create_storage_location() を実行する

パブリック S3 バケットの例

```
SELECT pgfs.create_storage_location(
name => hm_tiered_analytics_store,
url => s3://your-public-bucket/path/to/tiered-data,
options => {"aws_skip_signature": "true"},
credentials => {}
);
```

プライベート S3 バケットの例

```
SELECT pgfs.create_storage_location(
name => hm_tiered_analytics_store,
url => s3://your-private-bucket/path/to/tiered-data,
options => {"region": "us-west-2"},
credentials => {"access_key_id": "YOUR_AWS_ACCESS_KEY_ID", "secret_access_key": "YOUR_
↪AWS_SECRET_ACCESS_KEY"}
);
```

ガイダンス

- `url` をターゲットバケットパスに置き換えます。
- 必要に応じて、クラウドプロバイダーに合わせて `options` および `credentials` を調整します。

ステップ 3PGFS ストレージの場所が作成されたことを検証する

実行します

SELECT * FROM pgfs.storage_location;

新しく作成した PGFS の場所がリストされるはずです。

注

- **複数の PGFS ストレージ場所**を作成し、別の PGD ノードグループに割り当てることができます。
- 複数のノードグループが同じバケットにオフロードする場合、データのオーバーラップを回避するために、ノードグループごとに**個別のパスプレフィックス** を使用します。

次のステップ

これで、PGFS ストレージの場所の構成が完了しました。

- この場所をターゲットとするように **PGD ノードグループオフロード設定**を構成できます *Configure PGD node group for analytics offload*

- パーティショニングを管理し、オフロードをトリガーするように **BDR AutoPartition** を構成できます *Configure BDR AutoPartition with analytics offload*
- 後で、 **オフロードステータスとストレージ節約** をモニタできます。*Monitor Tiered Tables status and storage savings*
- PGD または Lakehouse から階層テーブルデータを照会できます *Query Tiered Tables from PGD and Lakehouse*

関連概念

PGFS が階層テーブルにどのように適合するかのアーキテクチャビューについては、次を参照してください。**ハイブリッドマネージャーの階層化テーブル**

Create a Lakehouse cluster

このガイドでは、ハイブリッドマネージャー HM で **レイクハウスクラスター**を作成する方法について説明します。Lakehouse クラスターは、オブジェクトストレージに保存されている **Apache Iceberg** または **Delta Lake** テーブルに対する高速クエリーに**スケーラブルな分析コンピューティング** を提供します。

アーキテクチャの背景については、 **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由** を参照してください。

Lakehouse クラスターを作成する理由

Lakehouse クラスターを作成することにより

- オブジェクトストレージ内の **大規模なデータセットで高速 Postgres SQL クエリを実行** できる機能のロックを解除します。
- PGD **または** 外部 Iceberg/Delta Lake テーブル **からオフロードされた**階層テーブルを照会できます。
- レイクハウスデータの **BI ツールとアドホック分析**のコンピューティングを提供します。

これは、ハイブリッドマネージャーで分析アーキテクチャを構築するための基礎的な手順です。

ゴール

このハウツーを完了すると、次のことができるようになります。

- Lakehouse クラスターを展開します。
- 標準の Postgres クライアントを使用して接続します。
- Iceberg および Delta Lake テーブルを照会するためのストレージとカタログを構成します。
- クラスターを使用して、既存のまたは新しく定義されたレイクハウステーブルを照会します。

前提条件

始める前に

- Lakehouse クラスターを作成する権限を持つ Hybrid Manager 環境にアクセスしていること。
- 利用可能なオブジェクトストレージがあること S3、GCS、MinIO など。

- 使用可能な Iceberg または Delta Lake テーブルがあるか、クラスターの作成後にそれらを定義する予定があります。
- Iceberg テーブルを使用する場合、Iceberg カタログを準備しています。
- *Configure an Iceberg REST catalog connection*

ステップ

ステップ 1 クラスターの作成を開始する

- HM ダッシュボードで、 **Create New** をクリックします。
- **Lakehouse Analytics** または **分析クラスター** を選択します。
- オープンレイクハウスフォーマットを照会するための分析クラスターを作成していることを確認します。

ステップ 2 自分の道を選択してください

以下を選択するように求められます。

- **テンプレート** —事前構成されたテンプレートを使用します利用可能な場合。
- **カスタムビルド** — 完全な構成制御には **Start from Scratch** をクリックしますほとんどの場合にお勧めします。

ステップ 3 クラスター設定を構成する

クラスター設定

- **クラスター名** — 一意の名前たとえば `sales_analytics_lakehouse` を入力します。
- **Password** — 強力なスーパーユーザーパスワードを入力します。
- **タグ** — オプションでタグを追加します `environment:production` 、`project:q3-reporting` など。
- **展開場所** — クラウドプロバイダーとリージョンを選択します。
- **データベースタイプ** — EDB Postgres Extended Server または Advanced Server を選択します。
- **Postgres バージョン** —Lakehouse 拡張機能を含むサポートされているバージョンを選択します。
- **インスタンスサイズ** —ワークロードに適切なコンピューティングサイズを選択します。

追加設定

- **ネットワーク** —VPC、サブネット、セキュリティグループ、IP ホワイトリストを構成します。
- **バックアップ** — 必要に応じてバックアップ設定を構成します。

ステップ 4 確認と作成

- **クラスターの概要** を確認します。
- すべての設定を確認します。
- **Create Cluster** をクリックします。
- HM ダッシュボードで進行状況を監視します。

次にできること

Lakehouse クラスターがプロビジョニングされたため、次のことができます。

- **クラスターに接続して** SQL クエリーを実行します。HM 提供の接続の詳細で標準の Postgres クライアント psql、DBeaver、pgAdmin を使用します。
- **データにアクセスするようにストレージとカタログを構成します**
- *Configure PGFS for Delta Lake* — Delta Lake またはファイルシステムベースの Iceberg を照会する場合。
- *Configure an Iceberg REST catalog connection* —カタログ管理の Iceberg テーブルを照会する場合、または PGD 階層化テーブルを使用する場合。
- **レイクハウステーブルの定義と照会**
- *Query existing Apache Iceberg tables*
- *Query Delta Lake tables*
- **PGD からオフロードされたクエリ階層テーブル**
- *Query Tiered Tables from PGD and Lakehouse*

関連トピック

- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**
- *Analytics in Hybrid Manager*

Monitor Tiered Tables status and storage savings

階層テーブルを構成した後、以下を監視することが重要です。

- どのパーティションがオフロードされたか
- オフロードが予想通りに進行しているかどうか
- PGD とオブジェクトストレージでのストレージ節約
- PGD クラスターでのオフロード関連アクティビティ

階層化テーブルは次の両方からモニタできます。

- **PGD クラスター** — SQL 可観測性クエリを使用する
- **ハイブリッドマネージャー** —モニタリングダッシュボードを介して環境で利用可能な場合

階層テーブルを監視する理由

階層テーブルを監視すると、次のことができます。

- オフロードが構成どおりに機能していることを検証します。
- ホット/コールドパーティションが予想通りに動作していることを確認します。
- ストレージの節約とパーティションのライフサイクルを追跡します。
- オフロードまたはパーティション管理に関する問題のトラブルシューティング。

ゴール

このハウツーを完了すると、次のことができるようになります。

- オフロード用に構成されているテーブルとその現在のステータスを特定します。
- AutoPartition オフロードアクティビティとジョブの成功を監視します。
- PGD プライマリストレージのストレージ節約を測定します。
- 可観測性クエリを使用して、パーティションの状態をモニタします。

前提条件

始める前に

- 階層化テーブルを構成する必要があります。参照
- *Configure PGFS storage for Tiered Tables*
- *Configure PGD node group for analytics offload*
- *Configure BDR AutoPartition with analytics offload*

分析テーブルのステータスをモニタする

分析オフロード用にマークされている PGD テーブルを表示するには

```
SELECT * FROM bdr.analytics_table;
```

役立つコラム

- `relation` — テーブル名
- `state` —現在のオフロード状態
- `enabled` — 分析オフロードが有効かどうか
- `disabled` — オフロードが無効かどうか
- `in_progress` — オフロードが現在実行されているかどうか

AutoPartition オフロードアクティビティの監視

オフロード関連タスクの AutoPartition ワークキューを確認します。

```
SELECT
wqs.workid, wis.ap_wi_started_at, wis.ap_wi_finished_at,
wis.ap_wi_status, "partition", sql
FROM bdr.taskmgr_local_workitem_status wis
RIGHT JOIN bdr.taskmgr_local_work_queue_status wqs
ON wis.ap_wi_workid = wqs.workid
WHERE sql NOT LIKE %bdr.autopartition_create_partition%
ORDER BY ap_wi_finished_at DESC;
```

これは、以下を追跡するのに役立ちます。

- オフロードジョブが実行されたとき
- ステータス成功/失敗
- どのパーティションが影響を受けたのか

PGD のストレージ節約をモニタする

親テーブルが使用するトランザクションストレージの削減を観察するには

```
SELECT pg_size_pretty(pg_total_relation_size(public.application_logs));
```

パーティションがオフロードおよび切り捨てられるため、このサイズは減少するはずです。

高度な可観測性 — ヒープとオフロードされたサイズ

ローカルヒープとパーティションのオフロードサイズを比較するには EDB の推奨ランブックから高度なオブザーバビリティクエリを実行できます。

- *EDB PGAA-Tiered-Tables-Observability-Queries* EDB パブリック runbook リポジトリから可観測性クエリーを参照してください。

これにより、次のことができます。

- パーティションごとのストレージをヒープとアイスバーグ/デルタに分割
- 残りのヒープサイズとオフロードされたサイズ

次にできること

階層化テーブルのステータスとストレージ節約を監視できるようになったので、次のことができます。

- **パーティショニングとオフロードポリシーを調整します** —観察されたホット/コールドアクセスパターンに基づいて、必要に応じて `analytics_offload_period` を調整します。
- **分析および BI ワークロードのクエリ階層テーブル**
- *Query Tiered Tables from PGD and Lakehouse*

- **レイクハウスクラスターを適切にスケーリング**して、オフロードされたデータクエリを処理します。
- **コンプライアンスと保持の目標を検証する** —ポリシーごとに古いデータがオフロードおよび保持されることを確認します。

関連するハウツー

- *Configure PGFS storage for Tiered Tables*
- *Configure PGD node group for analytics offload*
- *Configure BDR AutoPartition with analytics offload*
- *Query Tiered Tables from PGD and Lakehouse*

Offload PGD data to Apache Iceberg

このガイドでは、ハイブリッドマネージャー HM で **PGD データを Apache Iceberg** にオフロードする方法とタイミング、およびこれにより分析アーキテクチャに何が解除されるかについて説明します。

PGD データを Iceberg にオフロードする理由

- **コスト効率の高い階層化ストレージ**古い PGD データ「コールドデータ」をオブジェクトストレージに移動して、プライマリデータベースのストレージコストを削減します。
- **PGD パフォーマンスの高速化**トランザクションワークロードに合わせて、PGD 操作テーブルをより小さく、より高速に保ちます。
- **レイクハウス分析を有効にする**スケーラブルなレイクハウスクラスターベクトル化クエリを使用して、アイスバーグ形式でオフロードデータを照会します。
- **広範なデータ統合**オープンな Iceberg フォーマットを介して他のツール Spark、Trino、Flink で PGD データを利用できるようにします。

これを使用する場合

- **階層テーブル** を実装する場合主な使用ケース
- 統合された PGD +外部分析を使用してエンタープライズ**レイクハウスアーキテクチャ** を構築する場合
- 継続的なクエリアクセスニーズがある履歴 PGD データをアーカイブする場合

基本的な概念については、 **ハイブリッドマネージャーの階層化テーブル** を参照してください。

ハイブリッドマネージャーでの PGD オフロードの仕組み

PGD データの Iceberg へのオフロードには、3 つのレイヤーが含まれます。

Layer	Action
Storage	PGFS ストレージの場所および/または Iceberg カタログオフロードされたデータの移動先を定義します。
Node group	このストレージターゲットを使用するように PGD ノードグループを構成します。

次のページに続く

表 2 – 前のページからの続き

Layer	Action
Table level	オフロードする各 PGD テーブルの分析レプリケーションを有効にします。

前提条件

- HM 管理の PGD クラスター。
- PGD で定義された *Configure PGFS storage for Tiered Tables* S3 互換オブジェクトストア。
- オプション、推奨 *Configure an Iceberg REST catalog connection* が構成および接続されました。
- 分析オフロード用に構成された PGD ノードグループ
- *Configure PGD node group for analytics offload*

PGD データをオフロードする手順

ステップ 1 ノードグループがオフロード用に構成されていることを確認する 必須

- PGFS のみを使用する場合 `analytics_storage_location` を設定します。
- カタログを使用する場合 `analytics_write_catalog` を設定します。

参考 *Configure PGD node group for analytics offload*

ステップ 2 テーブルで分析レプリケーションを有効にする

特定の PGD テーブルのオフロードを開始するには

```
ALTER TABLE public.operational_logs SET (pgd.replicate_to_analytics = TRUE);
```

結果

- PGD は、このテーブルのデータを Iceberg に自動的にレプリケートします。
- フルテーブルオフロードを使用する場合 **即時** 。
- 階層化テーブルおよび BDR AutoPartition と組み合わせた場合、**パーティション** 。

オフロードされたデータへのアクセス

Iceberg カタログを使用する場合

- Lakehouse クラスターにカタログをアタッチします。

```
SELECT pgaa.attach_catalog(your_catalog_alias);
```

- オフロードされたテーブルをカタログ名で直接照会します。

PGFS のみを使用する場合ファイルシステムオフロード

- オブジェクトストレージの Iceberg パスを指す PGAA リーダーテーブルを定義します。

```
CREATE TABLE public.operational_logs_offloaded (...) USING PGAA WITH (pgaa.path = s3:/
↪/your/path, pgaa.format = iceberg);
```

注

- このメカニズムは **階層テーブル** を強化します。 *Configure BDR AutoPartition with analytics offload* を使用する場合、古いパーティションは自動的にオフロードされます。
- テーブルごとにオフロードを動的に有効/無効にできます。
- オフロードされたデータは、Iceberg で無期限に照会可能なままです。

これによりアンロックされるもの

☒ 大規模な PGD テーブルのためのシームレスな階層ストレージ ☒履歴 PGD データの高速レイクハウス分析 ☒オープンフォーマット Iceberg の相互運用性 ☒ PGD の運用コストとフットプリントの削減

次のステップ

PGD データが Iceberg にオフロードされるため、次のことができます。

- *Monitor Tiered Tables status and storage savings*
- *Query Tiered Tables from PGD and Lakehouse*
- *Query existing Apache Iceberg tables*

より広範なアーキテクチャビューについては、*Analytics in Hybrid Manager* を参照してください。

Performance tuning for Delta Lake queries

このガイドでは、ハイブリッドマネージャー HM レイクハウスクラスターから実行される Delta Lake クエリのパフォーマンスを調整および最適化するためのヒントを提供します。

Lakehouse クラスターは、ベクトル化クエリエンジン PGAA を使用して、オブジェクトストレージに保存されている Delta Lake テーブルを効率的に照会します。

Delta Lake 統合の背景については、 *Delta Lake in Hybrid Manager* を参照してください。

ゴール

このガイドを完了すると、次のことを行います。

Check レイクハウスクラスターからの Delta Lake クエリの一般的なチューニングパターンを理解する チェックパーティションプルーニングとデータスキップを利用するクエリを作成できるチェッククエリパフォーマンスを監視し、必要に応じてレイクハウスセットアップを調整する方法を知っていますチェック より速く、より多くのコストを解除- Postgres ツールからの Delta Lake データの効率的な分析

これを使用する場合

次の場合にこれらのプラクティスを適用する必要があります。

- HM Lakehouse クラスターから既存の Delta Lake テーブルを照会している
- クエリ速度とリソース効率を向上させたい
- BI ツール統合 Tableau、Power BI などのための Lakehouse クラスターを準備している場合
- 長時間実行される Delta Lake クエリーのトラブルシューティングを行っている場合

一般的なベストプラクティス

パーティショニングを効果的に使用する

- Delta Lake はパーティショニングをサポート —一般的なクエリフィルタと一致するパーティション列を使用して Delta Lake テーブルをデザインします。
- パーティション列でフィルタリングするクエリーは、スキャンを高速化するために不要な Parquet ファイルをプルーニングできます。

例

```
SELECT order_id, sale_amount
FROM public.sales_delta_table
WHERE sale_year = 2024 AND sale_month = 5;
```

データスキップを活用する

- Delta Lake は、Delta トランザクションログに保存されている統計のデータスキップをサポートしています。
- Delta Lake テーブルが Spark または統計を収集する別のツールによって書き込まれている場合、PGAA はこれを活用して、より効率的なクエリプランニングを行う場合があります。

LIMIT とターゲットを絞った予測を使用する

- 探索クエリーの場合、I/O を最小限に抑えるために、常に LIMIT を使用し、必要な列のみを SELECT します。

例

```
SELECT order_id, sale_amount
FROM public.sales_delta_table
LIMIT 100;
```

早期かつ具体的にフィルタリングする

フィルタを WHERE 句にプッシュして、最大限のプルーニングを有効にします。

良い

```
SELECT * FROM public.sales_delta_table
WHERE sale_date >= 2023-01-01 AND product_category = Electronics;
```

避けてください

```
SELECT * FROM public.sales_delta_table;
```

(フルテーブルスキャン)

PGFS ロケーションの近接性を検証する

- データ転送のレイテンシーを最小限に抑えるために、Lakehouse クラスターがオブジェクトストレージに近いクラウドリージョンでプロビジョニングされていることを確認します。

クエリパフォーマンスの監視

- HM モニタリングダッシュボードを使用して、Lakehouse クラスターでのクエリ時間とリソースの使用状況を観察します。
- 長いランタイムまたは大規模な I/O を伴うクエリーを特定および調整します。

既知の制限

- PGAA は現在、Delta Lake テーブルへの読み取り専用アクセスをサポートしています。
- スキーマの進化がサポートされています —PGAA は、デルタトランザクションログから現在のスキーマを読み取ります。
- 複雑な ACID 機能 MERGE、DELETE は、PGAA を介した書き込みではサポートされていません- デルタテーブルの変更には Spark またはサポートされている ETL ツールを使用します。

次のステップ

Delta Lake クエリの最適化が完了しました。

- より良いパフォーマンスで *Query Delta Lake tables* できます。
- *Configure PGFS for Delta Lake* を使用して、高速で信頼性の高いストレージアクセスを保証できます。
- *Integrate Lakehouse queries with BI tools* して、新しい分析ユースケースのロックを解除できます。

アーキテクチャの概要については、 *Analytics in Hybrid Manager* を参照してください。

Query Delta Lake tables

このガイドでは、ハイブリッドマネージャー HM レイクハウスクラスターから既存の Delta Lake テーブルを照会する方法について説明します。

Lakehouse クラスターをオブジェクトストレージに保存されている Delta Lake データに接続し、PGAA を使用して SQL クエリーを実行できます。

Delta Lake 統合の背景については、 *Delta Lake in Hybrid Manager* を参照してください。##ゴール

このガイドを完了すると、次のことを行います。

チェック デルタレイクデータを指す PGFS ストレージ場所を定義できることチェック デルタレイクデータを Postgres SQL に公開する PGAA リーダーテーブルを作成できること チェック 既存の Delta Lake テーブルへの SQL ベースの分析と BI アクセスのロックを解除します チェック Postgres を有効にする移動または ETL なしでデルタデータを照会するためのツールとユーザー##これを使用する場合

次の場合にこのガイドに従う必要があります。

- オブジェクトストレージの場所に Delta Lake テーブルが既にあります
- ハイブリッドマネージャーの Lakehouse クラスターを使用してこのデータを照会したい
- BI ツールまたは Postgres クライアントを Delta Lake データに接続したい
- デルタ##前提条件とのレイクハウスの統合をテストまたは検証しています
- ハイブリッドマネージャーを介してプロビジョニングされたアクティブな Lakehouse クラスター
- S3 互換のオブジェクト保存場所の Delta Lake テーブル
- _delta_log ディレクトリと Parquet ファイルは Delta Lake パスに存在する必要があります
- オブジェクトストレージにアクセスするための資格情報必要な場合 ## ステップ

1.PGFS ストレージの場所を定義する

Lakehouse クラスターに接続し、PGFS ストレージの場所を定義します。

パブリックバケットの例

```
SELECT pgfs.create_storage_location(
name => my_public_delta_lake_store,
url => s3://my-public-delta-data/,
options => {"aws_skip_signature": "true"},
credentials => {}
);
```

プライベートバケットの例

edb_notranlate_1 ### 2.PGAA 拡張機能を有効にする

edb_notranlate_2 ### 3. Delta Lake の PGAA 外部テーブルを作成します

これにより、Delta Lake テーブルが Postgres SQL に公開されます。

edb_notranlate_3 ### 4. Delta Lake データの照会

例 レコード数を取得します。

```
SELECT COUNT(*) FROM public.sales_delta_table;
```

例フィルターとプロジェクト列。

sql SELECT order_id, customer_name, sale_amount FROM public.sales_delta_table WHERE sale_date >= 2023-01-01 AND product_category = Electronics LIMIT 100; ## 注

- デルタレイクスキーマは_delta_log から推測されます
- Lakehouse クラスターは、高速なクエリのためにベクトル化されたクエリエンジンを使用します
- カタログ統合は必要ありません — Delta Lake はファイルシステムベースです## 次のステップ

Delta Lake 外部テーブルを作成し、クエリを実行しました。

- 必要に応じて、 *Configure PGFS for Delta Lake* して場所を追加できます
- *Performance tuning for Delta Lake queries* 大規模なクエリーを最適化できます
- *Integrate Lakehouse queries with BI tools* を使用して、デルタデータへのビジネスインテリジェンスへのアクセスを有効にすることができます。アーキテクチャの概要については、*Analytics in Hybrid Manager* を参照してください。

Query existing Apache Iceberg tables

このガイドでは、ハイブリッドマネージャー HM レイクハウスクラスターから既存の Apache Iceberg テーブルを照会する方法について説明します。

Lakehouse クラスターをオブジェクトストレージに保存されている Iceberg テーブルに接続し、PGAA を使用して SQL クエリーを実行できます。

Iceberg 統合の背景については、 *Apache Iceberg in Hybrid Manager* を参照してください。

ゴール

このガイドを完了すると、次のことを行います。

チェック Iceberg データを指す PGFS ストレージの場所を定義できることチェック Iceberg テーブルを Postgres SQL に公開する PGAA リーダーテーブルを作成できることチェック SQL ベースの分析と既存の Iceberg テーブルへの BI アクセスのロックを解除しますチェック Postgres ツールとユーザーを有効にする他のシステムによって作成された Iceberg データを照会する

これを使用する場合

次の場合にこのガイドに従う必要があります。

- Spark、Trino、Flink、PGD オフロード、またはその他のシステムによって書き込まれた Iceberg テーブルが既にあります
- ハイブリッドマネージャーの Lakehouse クラスターを使用してこのデータを照会したい
- BI ツールまたは Postgres クライアントを Iceberg データに接続したい
- パフォーマンスをテストしているか、既存の Iceberg データセットとの Lakehouse 統合を検証しています。

前提条件

- HM プロビジョニングされた EDB Postgres Lakehouse クラスター
- S3 互換オブジェクトストレージの既存の Apache Iceberg テーブル
- 適切なストレージ権限必要な場合

- version-hint.text または vX.metadata.json のいずれかを含む Iceberg テーブルルートへのパス

ステップ

1.PGFS ストレージの場所を定義する

Lakehouse クラスターに接続し、PGFS ストレージの場所を定義します。

例

```
SELECT pgfs.create_storage_location(
name => my_s3_iceberg_data,
url => s3://your-bucket/path/to/iceberg,
options => {},
credentials => {"access_key_id": "...", "secret_access_key": "..."}
);
```

2. Iceberg の PGAA 外部テーブルを作成します

```
CREATE TABLE public.my_sales_iceberg_data (
    sales_id INT,
    sales_date DATE,
    sales_amount NUMERIC
)
USING PGAA
WITH (
    pgaa.storage_location = my_s3_iceberg_data,
    pgaa.path = sales_records/iceberg_table_root,
    pgaa.format = iceberg
);
```

これにより、Iceberg テーブルが Postgres SQL に公開されます。

```
CREATE TABLE public.my_sales_iceberg_data ()
USING PGAA
WITH (
pgaa.storage_location = my_s3_iceberg_data,
pgaa.path = sales_records/iceberg_table_root,
pgaa.format = iceberg
);
```

注カタログで管理されていない Iceberg テーブルの場合、pgaa.path は通常、version-hint.text ファイルが存在するディレクトリを指すか、vX.metadata.json ファイルを直接指します。

3. Iceberg テーブルを照会します

標準の SQL を使用してクエリーを実行します。

例

```
SELECT * FROM public.my_sales_iceberg_data
WHERE sales_region = North America
LIMIT 100;
```

注

- Lakehouse クラスターは、高速な Iceberg クエリーのためのベクトル化クエリエンジンを使用します
- ファイルベースの Iceberg テーブルを照会するためにカタログ接続は必要ありません
- Iceberg テーブルがカタログ管理されている場合、代替フローについては *Configure an Iceberg REST catalog connection* を参照してください

次のステップ

Iceberg 外部テーブルを作成し、クエリを実行しました。

- *Configure an Iceberg REST catalog connection* でカタログ管理テーブルを照会できます
- *Offload PGD data to Apache Iceberg* を使用して、アイスバーグレイクハウスにデータを追加できます
- *Query Tiered Tables from PGD and Lakehouse* を使用して運用データと統合できます

アーキテクチャの概要については、 *Analytics in Hybrid Manager* を参照してください。

Query Tiered Tables from PGD and Lakehouse

階層テーブルが実装されると、 **ホットデータ**ローカル PGD パーティションと **コールドデータ**オフロードされた Iceberg パーティションの両方を透過的に照会できます。

このガイドでは、次のことを説明します。

- PGD から階層テーブルを照会する方法
- Lakehouse クラスターから階層テーブルを照会する方法
- クエリプランナーがクエリをルーティングする方法
- オフロード専用データを照会する方法

バックグラウンドについては、 **ハイブリッドマネージャーの階層化テーブル** を参照してください。

ゴール

このガイドを完了すると、次のことを行います。

- PGD および Lakehouse から階層化テーブルを透過的に照会できるようになります
- クエリプランナーがホットおよびコールドデータにアクセスする方法を理解する

- **オフロード専用** パーティションを直接照会する方法を理解する
- レポートと分析にオフロードされたビューを使用する場合を理解する

これを使用する場合

次の場合にこのガイドに従う必要があります。

- ハイブリッドマネージャーに階層化テーブルを実装している
- PGD クエリーでホット+コールドデータにアクセスできることを検証したい
- PGD または Lakehouse からオフロードされたデータのみを照会したい
- 履歴コールドデータに対してレポートを最適化したい

前提条件

- `analytics_offload_period` が構成された BDR AutoPartition *Configure BDR AutoPartition with analytics offload* を参照
- PGD ノードグループオフロード構成済み *Configure PGD node group for analytics offload* を参照してください
- Iceberg カタログに接続されたオプションの Lakehouse クラスター *Query existing Apache Iceberg tables* を参照してください

PGD での親テーブルの照会

親パーティションテーブル `public.application_logs` などは、ホット+コールドデータの統一インターフェイスです。

クエリーの例

```
SELECT log_level, COUNT(*)
FROM public.application_logs
WHERE event_timestamp >= 2023-06-01 AND event_timestamp < 2024-06-01
GROUP BY log_level;
```

ボンネットの下で何が起こっているのか

- プランナーは、ホットデータの **ローカル PGD パーティション**にアクセスします
- プランナーは、コールドデータの PGAA を介して **Iceberg オフロードパーティション** にアクセスします
- クエリーに対して透過的 — 特別な構文は必要ありません

ヒント EXPLAIN ANALYZE を実行して、プランナーの動作を観察します。

オフロードデータのみの照会 PGD

PGAA は、 **オフロードビュー** を自動的に作成します。

```
SELECT * FROM public.application_logs_offloaded;
```

このビューは、 **オフロードされたパーティションのみ** を照会します。

例

```
SELECT log_level, COUNT(*)
FROM public.application_logs_offloaded;
```

オフロードビューを使用する場合

- **コールド/履歴** データのみをターゲットとしたレポート
- ホットパーティションのスキャンを回避するクエリー
- オフロードが予想通りに機能していることのテスト

Lakehouse からオフロードされたデータの照会

PGD オフロードがカタログを使用する場合推奨、Lakehouse クラスターに同じカタログを接続します。

```
SELECT pgaa.add_catalog(...);

SELECT pgaa.attach_catalog(...);
```

次に、Iceberg テーブルを直接照会します。

```
SELECT * FROM iot_data.application_logs_part_2023_jan;
```

または、 `application_logs_offloaded` ビューロジックを PGD から Lakehouse に複製できます。

- すべてのカタログ管理のオフロードパーティションを選択するビューを作成する
- PGD エクスペリエンスと一致する BI ツールに役立ちます

プランナーの動作に関する注意事項

- **親テーブル** は常にデフォルトの推奨インターフェイスです
- プランナーは、PGD と Iceberg パーティションへのアクセスを自動的に最適化します
- **広いタイムウィンドウ**を使用するクエリーは、`application_logs_offloaded` をターゲットとすることによりメリットがある場合があります
- オフロードビューは、AutoPartition が `analytics_offload_period` で構成された場合にのみ作成されます

次のステップ

階層化テーブルの照会が完了しました。

- *Monitor Tiered Tables status and storage savings* でパーティションの移動と領域の節約を追跡できます
- *Offload PGD data to Apache Iceberg* を使用して、より広範な履歴をカバーできます
- *Query existing Apache Iceberg tables* を外部ツールおよび BI プラットフォームと統合できます

より広範なアーキテクチャビューについては、*Analytics in Hybrid Manager* を参照してください。

Leveraging Analytics in Hybrid Manager

このページでは、ハイブリッドマネージャー HM 内の分析機能を活用する方法に関するロールベースのガイダンスを提供します。

データベース管理者、DevOps エンジニア、データサイエンティスト、またはアプリケーション開発者であっても、Hybrid Manager は EDB Postgres 展開の管理、分析、および最適化に役立つツールを提供します。

主要な概念については、 *Analytics Concepts in Hybrid Manager* を参照してください。

データベース管理者 DBA

主な焦点 HM 管理の Postgres 環境のパフォーマンス、信頼性、およびコスト効率。

主要な目的と HM 機能

- HM ダッシュボードを使用して PGD クラスターと Lakehouse ノードをモニタします。
- PGD に階層テーブルを実装して、コールドデータをオブジェクトストレージにオフロードします。
- 傾向を分析し、必要に応じて Lakehouse クラスターをスケーリングします。
- PGD および Lakehouse メタデータカタログを含むのバックアップとリカバリーを管理します。
- HM および Postgres のセキュリティ機能を介してレイクハウスクラスターへのアクセスを制御します。

キーリンク

- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**
- *Implementing Tiered Tables in Hybrid Manager*
- *Working with Apache Iceberg in Hybrid Manager*
- *Leveraging Analytics in Hybrid Manager*

DevOps エンジニア/SRE

主な焦点 分析インフラストラクチャの展開と可観測性の自動化。

主要な目的と HM 機能

- HM UI、API、または CLI を使用して、PGD クラスターと Lakehouse クラスターを展開します。
- 必要に応じて、HM を介してレイクハウスクラスターサイズを調整します。
- HM ダッシュボードでレイクハウスと PGD のパフォーマンスを監視します。
- HM API / CLI を介してプロビジョニングと構成を自動化します。
- 個別のコンピューティング/ストレージアーキテクチャを使用してコストを最適化します。

キーリンク

- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**
- *Implementing Tiered Tables in Hybrid Manager*
- *Working with Apache Iceberg in Hybrid Manager*
- *Leveraging Analytics in Hybrid Manager*

データサイエンティスト / データアナリスト

主な焦点ハイブリッドマネージャーによって管理されるデータの効率的なアクセスと分析。

主要な目的と HM 機能

- BI ツールまたは分析ツール Jupyter、R、SQL クライアントを Lakehouse ノードに接続します。
- Lakehouse を介して Iceberg および Delta Lake テーブルを照会します。
- 大規模なデータセットの高速分析には、ベクトル化された Lakehouse ノードを使用します。
- Lakehouse または PGD で Postgres SQL を使用してデータを準備およびクリーンアップします。

キーリンク

- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**
- *Working with Apache Iceberg in Hybrid Manager*
- *Working with Delta Lake in Hybrid Manager*
- *Implementing Tiered Tables in Hybrid Manager*

アプリケーション開発者

主な焦点ハイブリッドマネージャーによって管理されるデータと対話するアプリケーションを構築します。

主要な目的と HM 機能

- 埋め込みレポートのために、アプリケーションを Lakehouse ノードまたは PGD リードレプリカに接続します。
- 複雑な分析クエリを Lakehouse ノードにルーティングします。

- 階層テーブルを使用して、ホット PGD およびコールドレイクハウス階層全体で時系列データを管理します。
- オプション HM で Gen AI Builder を使用する場合、グリップテープ構造/ツールをアプリケーションと統合します。

キーリンク

- **ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由**
- *Implementing Tiered Tables in Hybrid Manager*
- *Leveraging Analytics in Hybrid Manager*

コア分析概念の概要については、 *Analytics Concepts in Hybrid Manager* を参照してください。

Solving Business Problems with Analytics in Hybrid Manager

このページでは、一般的な分析のユースケースを検討し、ハイブリッドマネージャー HM 機能がそれらの問題への対処にどのように役立つかについて説明します。

EDB Postgres Lakehouse クラスター、PGD を使用した階層テーブル、およびオープンテーブルフォーマット Apache Iceberg、Delta Lake との統合を使用することにより、Postgres 環境内で柔軟な分析ソリューションを構築できます。

主要な概念については、 *Analytics Concepts in Hybrid Manager* を参照してください。

ペルソナベースのガイダンスについては、 *Persona-Based Guide to Using Analytics in Hybrid Manager* を参照してください。

一般的なユースケースと HM ソリューション

大規模なデータセットに関するアドホックビジネスインテリジェンスとレポート

課題ビジネスユーザーとデータアナリストは、運用中の PGD クラスターに影響を与えることなく、大規模な履歴データセットで複雑なクエリーを実行する必要があります。

HM ソリューション

- 大規模なデータセットを Iceberg または Delta Lake テーブルとしてオブジェクトストレージに保存します。
- 高速なベクトル化クエリ実行のためにレイクハウスクラスターをプロビジョニングします。
- 標準の Postgres コネクタを使用して、BI ツール Tableau、Power BI、Looker をレイクハウスクラスターに接続します。
- テーブルの検出と管理には、Iceberg カタログ HM 管理または外部を使用します。

詳細はこちら ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由

Working with Apache Iceberg in Hybrid Manager

時系列データの費用対効果の高いアーカイブと分析

課題履歴分析へのアクセスを維持しながら、増加する時系列データを費用対効果の高い方法で管理します。

HM ソリューション

- Iceberg へのコールドパーティションの自動オフロードのために、 PGD AutoPartition と `analytics_offload_period` を使用して階層テーブルを構成します。
- Lakehouse クラスターを使用して、オブジェクトストレージのコールドデータを照会します。
- PGD 親テーブルを介したホットデータとコールドデータの両方への透過的アクセスをサポートします。

詳細はこちら *Implementing Tiered Tables in Hybrid Manager*

業務データとエンタープライズデータレイクの統合

課題統合分析のために、PGD クラスターからのデータを他のエンタープライズデータソースと組み合わせます。

HM ソリューション

- PGD データを Iceberg または Delta 形式のオブジェクトストレージにオフロードします。
- 共有 Iceberg カタログ HM 管理または外部を使用して、相互運用性をサポートします。
- Lakehouse クラスターまたは外部エンジン Spark、Trino を使用して、共有 Iceberg または Delta テーブルを照会します。

詳細はこちら *Working with Apache Iceberg in Hybrid Manager*

ハイブリッドマネージャーで *Lakehouse* クラスターを使用する理由

アプリケーションの分析コンポーネントの高速化

課題アプリケーションには、プライマリ PGD クラスターのパフォーマンスに影響を与える可能性のある分析クエリが必要です。

HM ソリューション

- 読み取り負荷の高い分析クエリを専用の Lakehouse クラスターにルーティングします。
- Lakehouse テーブルまたはマテリアライズドビューの集計を事前計算して、アプリケーションクエリを高速化します。
- 階層テーブルを使用して、ホットストレージとコールドストレージ全体で時系列データを効率的に管理します。

利点

- PGD パフォーマンスを保護します。
- アプリケーション分析のためのスケーラブルなコンピューティングを提供します。
- 費用対効果の高いストレージを活用します。

業界の例とチュートリアル

金融サービス

- Lakehouse とオフロードされた PGD データを使用した日次リスク報告システムの構築
- 長期的な取引データのアーカイブと分析のための階層テーブルの実装

小売と e コマース

- Lakehouse と BI ツールの統合を使用した顧客の購入履歴の分析

電気通信

- 階層テーブルとレイクハウスクラスターを使用した通話詳細レコード CDR 分析

次のトピック

Leveraging Analytics in Hybrid Manager

7.5.5 Tiered Tables in Hybrid Manager

Hybrid ManagerHM の階層テーブルは、古いデータを PGD クラスターから Apache Iceberg 形式の費用対効果の高いオブジェクトストレージに移動することにより、大規模な時系列または履歴データセットを管理するための自動ソリューションを提供します。

ハイブリッドマネージャーは、この機能を次の全体にわたって統合します。

- BDR AutoPartition を使用した PGD クラスター
- オフロードのための PGAA 拡張機能
- クエリーのためのレイクハウスクラスター
- 一元化カタログサービス Lakekeeper または外部 Iceberg カタログ

階層テーブルの概念的な概要については、 *Understanding Tiered Tables with EDB Postgres* を参照してください。

ハイブリッドマネージャーで階層化テーブルを使用する理由

- **ストレージコストの削減**「コールド」データをオブジェクトストレージ Iceberg にオフロードし、プライマリ PGD トランザクションテーブルを縮小します。
- **トランザクションパフォーマンスの高速化**効率的な PGD 操作のために、「ホット」データパーティションを小さく保ちます。
- **自動ライフサイクル管理**経過期間に基づいて、階層間でデータを自動的に移動します。
- **透過的分析** PGD 親テーブルまたはレイクハウスクラスターを介してホットデータとコールドデータの両方を照会します。
- **統合管理**ハイブリッドマネージャーを介してすべてのコンポーネントを構成および監視します。

重要な用語とアーキテクチャの概要

ハイブリッドマネージャーで使用されるコア分析用語 PGFS、PGAA、アナリティクスオフロード、レイクハウスクラスターなどの定義については、*Analytics Concepts in Hybrid Manager* を参照してください。

ハイブリッドマネージャーで階層化テーブルを使用する必要がある場合

次の場合にハイブリッドマネージャーの階層テーブルを使用します。

- **大規模な時系列データセットを費用効率の高い方法で管理** 。
- **パフォーマンスを向上させるために、PGD 操作テーブルをリーンに保ちます**。
- 古いデータを利用可能な状態にしておくことにより、コンプライアンスニーズに対応しますが、PGD ストレージの外部にあります。
- **BI ユーザーが運用データベースに影響を与えることなく履歴トレンドクエリを実行できるようにします**。
- 最小限の手動介入で **データライフサイクル** を自動化します。

階層化テーブルのユースケース

- **時系列データ**ロギング、IoT センサーの読み取り値、アプリケーションテレメトリ。
- **アーカイブ** コンプライアンスのためのコールドデータの長期保持。
- **ヒストリカルトレンド分析** PGD のパフォーマンスに影響を与えることなく、長年のデータを照会する BI ツール。
- **大規模な追加ほとんどのテーブル**完全な分析アクセスを維持しながら、トランザクションのフットプリントを小さく保ちます。

HM アーキテクチャでの階層テーブルの仕組み

- **PGD クラスター**パーティショニングと古いパーティションの Iceberg への自動オフロードを管理します。
- **PGFS ストレージの場所**オフロードのオブジェクトストレージターゲットを定義します。
- **Iceberg カタログ**オプションでカタログ Lakekeeper または外部でオフロードされたテーブルを管理します。
- **レイクハウスクラスター**オフロードされた Iceberg データを照会するためのスケーラブルな分析コンピューティングを提供します。
- **モニタリング** HM モニタリングツールと可観測性クエリを使用して、オフロードステータスとストレージの節約を追跡します。

EDB Hybrid Manager 内の前提条件

HM に階層テーブルを実装する前に

- **アクティブなハイブリッドマネージャーインスタンス**
- **プロビジョニングされた PGD クラスター** PGAA および PGFS 拡張機能が有効になったバージョン 6.0+
- **レイクハウスクラスター推奨** オフロードデータの照会用
- **カタログサービス** オプショナルただし推奨 —HM 管理の Lakekeeper または外部 REST 互換カタログ
- **カタログのマシンユーザーカタログを使用する場合**適切なカタログデータのライタ/読み取り権限がある場合
- **オブジェクトストレージ** S3 互換、プライベートの場合資格情報あり
- **ユーザー権限**データベースユーザーは BDR および PGAA ファンクションの作成/変更/実行権限を持っている必要があります

メイン機能

- **自動パーティショニング** BDR AutoPartition ストラテジーと `analytics_offload_period` を定義します。
- **ストレージ階層化**オフロードデータには PGFS または Iceberg カタログターゲットを使用します。
- **クエリの透過的** PGD 親テーブルクエリは、ローカル層とアイスバーグ層の両方にヒットする可能性があります。Lakehouse クラスターは、Iceberg テーブルを直接照会できます。
- **モニタステータス**オフロードの進行状況を追跡し、Iceberg コンテンツを検証し、領域の節約を観察します。

ハイブリッドマネージャーでの階層化テーブルのスタートガイド

ハイブリッドマネージャーで階層テーブルの使用を開始するには

1. *Configure PGFS storage for Tiered Tables* 。
2. *Configure PGD node group for analytics offload* 。
3. *Configure BDR AutoPartition with analytics offload* 。
4. オプショナル *Query Tiered Tables from PGD and Lakehouse* 。
5. *Monitor Tiered Tables status and storage savings* 。

関連するハウツー

- *Configure PGFS storage for Tiered Tables*
- *Configure PGD node group for analytics offload*
- *Configure BDR AutoPartition with analytics offload*

- *Query Tiered Tables from PGD and Lakehouse*
- *Monitor Tiered Tables status and storage savings*

可観測性のヒント

- PGD クラスターの状態とオフロードの進行状況には HM ダッシュボードを使用します。
- `bdr.analytics_table` およびパーティションビューで分析クエリを実行します。
- `pg_total_relation_size()` を使用して、PGD ノードで再利用された領域を観察します。
- クラウドストレージコンソールまたは分析を使用して、Iceberg オブジェクトサイズの増加を追跡します。

次のトピック

EDB Postgres Lakehouse Clusters on Hybrid Manager

第8章　Sovereign AI and Data Factory

Sovereign AI と Data 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、インストールした瞬間に高度な Postgres、vector、および 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。主権 AI と Data Factory 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**は一貫性がない、オーバーサブスクライブ、または制限されています
- **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です

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、 *Solving Business Problems with Analytics in Hybrid Manager* を参照してください。

8.4 ハードウェア構成

EDB は、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に 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 次に探索する

- *Concepts and architecture*
- *Glossary and key terms*
- *What to Expect*
- *Solving Business Problems with Analytics in Hybrid Manager*
- *Frequently asked questions*

ライフサイクル操作の詳細については、*Agent Studio in Hybrid Manager* にアクセスしてください。

8.5.1 Concepts and architecture

Sovereign AI と Data Factory は、物理的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。

ノードタイプとロール

前提条件として、*System specifications* の構成の詳細を参照してください。システムには、複数のノードのロールが含まれており、それぞれに明確な責任と構成があります。

管理ノード

管理ノードは、ソブリン AI と Data Factory の追加ソフトウェアパッケージ、ライセンス、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

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。

コンピューティングノード

計算ノードは、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

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。

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。顧客は Kubernetes に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。

- Postgres クラスター高可用性の展開
- 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

「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
- 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 AI と Data Factory 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

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 | OK | OK 埋め込み|内部データをベクター形式に変換する | NO| OK 取得拡張世代 RAG | Postgres + LLM を組み合わせる | NO| OK 推論 | GPU を使用してモデルをローカルに提供する | NO| OK 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 | NO| OK

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

よくある質問

クラウドサービスの代わりにこのシステムを使用する理由は何ですか?

主権 AI と Data 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

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、Hybrid Manager、および AI ツールのサポートと同様に、契約の下でカバーされています。

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
- AI 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

これ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。

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、標準のコンテナ化モデルをサポートしています。これには以下が含まれます

- `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。

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 AI および Data Factory 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。

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。

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

この用語集は、Sovereign AI および Data Factory のドキュメント全体で使用される重要な用語とアーキテクチャコンポーネントを定義します。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

Postgres DistributedPGD

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。

RAG 検索拡張生成

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。

ソブリン AI

すべてのデータ、モデル、実行環境が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。必要に応じて完全にエアギャップ。

Vector Store

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

システム仕様

Sovereign AI および Data 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。

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

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

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
Networking	AOC-AG-I2M-O	AIOM デュアルポート 1GbE RJ45	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-S25G-I2S-O	標準ロープロファイル 25G デュアルポート SFP28	1

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
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

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

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
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-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
GPU	NVIDIA HGX B200	8x B200 180GB GPU を搭載したベースボード	1

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
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 MCX75310AAS-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

一般的なワークロードとユースケース

Sovereign AI と Data 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。

このセクションでは、顧客がクラウドベースのプラットフォームではなくシステムを選択する理由と、どのようなユースケースに最適化されているかについて概要を説明します。

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。ソブリン AI と Data Factory は、次のニーズを満たすために存在します。

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。

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。

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。

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。

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。

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 AI および Data 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。

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した。

ハイブリッド DBaaS

- 1 つのインターフェイスから、複数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。

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。

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。

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。

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

このページでは、最初の 90 日間のソブリン AI および Data Factory システムの配信からアクティブな使用とサポートまでの受け取り、セットアップ、および操作のエクスペリエンスを概要します。

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

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。

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、ハイブリッドマネージャーインターフェイスを介して実行されます。

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 コンテナ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。すべては UI を介して管理されます。

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。

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。
