



AI Factory

リリース 1.3

EnterpriseDB Corporation

2025 年 10 月 29 日

目次

第 1 章	Where AI Factory runs	2
1.1	前提条件	3
1.2	なぜソブリン AI なのか？	3
1.3	あなたが構築できるもの	4
1.4	道を選択してください	4
1.5	対象者、および開始方法	4
1.6	コア機能	5
1.6.1	Gen AI ワークロード	5
1.6.2	アセットライブラリーとモデル提供	5
1.6.3	パイプラインとベクターエンジン	5
1.7	EDB PG AI とアナリティクス	5
1.8	学んで構築する	6
1.9	EDB PG AI ソリューションを構築する	6
第 2 章	AI Factory	7
2.1	AI Factory Hub Map	7
2.1.1	ここから始めます	7
2.1.2	AI Factory でビルドする	7
2.1.3	ハイブリッドマネージャー	8
2.1.4	実行される場所と前提条件	8
2.1.5	リファレンス	8
2.2	AI Factory Learning Center	8
2.2.1	クイックスタート	9
2.2.2	概念的な説明	9
2.2.3	ハウツーガイド	9
2.2.4	ラーニングパス	9
2.2.5	ソリューション	10
2.2.6	ユースケース	10
	Getting Started - AI Factory 101	10
	ラーニングパスモジュール	11
	学習の旅を続けます	13
	Building Production-Ready AI Features - AI Factory 201	14
	次のステップ	17

	関連する学習リソース	17
	Advanced AI Factory Usage and Extensibility - AI Factory 301	17
	次のステップ	19
	関連する学習リソース	20
	AI Factory Explained	20
	AI Factory How-To Guides	21
	Learning Paths	22
	ラーニングパス	23
	おすすめトレーニングコース	24
	次はどこに行きますか？	25
	Industry Solutions with AI Factory	25
	AI Factory Use Cases	28
2.3	Concepts	33
2.4	Concepts	33
2.4.1	Postgres とハイブリッドマネージャーを使用した AI に関する EDB のビジョン	33
2.5	Concepts	34
2.6	Concepts	34
2.7	Concepts	34
2.8	Concepts	34
2.8.1	概要	34
	AI Factory vs. “Generic AI”	35
	Fundamental AI Concepts	38
	AI Factory terminology	44
	Sovereign AI Explained	49
	Security and governance in AI Factory	52
第 3 章	Usage	55
3.1	AI Factory Models	55
3.2	AI ファクトリーモデル	55
3.2.1	システムの概要	55
	コアコンポーネント	55
3.2.2	戦略的メリット	56
	ソブリン AI の実装	56
	オペレーションの一貫性	56
	スケーラブルなリソース管理	56
3.2.3	アーキテクチャの統合	56
	ハイブリッドマネージャーの依存関係	56
3.2.4	運用ワークフロー	56
	モデルのライフサイクル管理	56
	ガバナンスフレームワーク	57
3.2.5	学習パスウェイ	57
	基礎の理解	57
	実際の実装	57
	参考資料	58
3.2.6	ユースケースアプリケーション	58
	Enterprise RAG Systems	58
	プライベート LLM 展開	58

	マルチモーダル AI アプリケーション	58
	ハイブリッドモデルアーキテクチャ	59
3.2.7	実装の考慮事項	59
	インフラストラクチャ要件	59
	ネットワークアーキテクチャ	59
	セキュリティフレームワーク	59
	運用の複雑さ	59
3.3	Gen AI Builder	153
3.4	Gen AI Builder	153
3.4.1	プラットフォームアーキテクチャ	153
	コアインフラストラクチャの統合	153
3.4.2	アーキテクチャコンポーネント	154
	ナレッジベースフレームワーク	154
	検索システム	154
	アシスタントフレームワーク	154
3.4.3	実装パターン	155
	エンタープライズナレッジシステム	155
	検索拡張生成アプリケーション	155
	会話型 AI プラットフォーム	156
3.4.4	開発フレームワーク	156
	ナレッジベース開発	156
	アシスタント設定	157
	API 開発	157
3.4.5	統合アーキテクチャ	158
	EDB Postgres AI エコシステム	158
	外部システム統合	158
3.4.6	運用上の考慮事項	158
	パフォーマンス特性	158
	セキュリティフレームワーク	158
	ガバナンス統合	159
3.4.7	はじめに	159
	前提条件	159
	初期実装	159
	学習リソース	160
3.5	AI Factory Pipelines	299
3.5.1	なぜパイプラインなのか?	300
3.5.2	あなたにできること	300
3.5.3	その仕組み	300
	準備者	300
	ナレッジベース	301
	自動処理	301
3.5.4	誰がパイプラインを使用する必要がありますか?	301
3.5.5	パイプラインを使用する場合	302
3.5.6	構成値の説明	302
3.5.7	詳細はこちら	302
	Start here	302
	Get started	306

	Core concepts	315
	Storage and access	422
	Pipelines reference	431
	EDB Postgres AI - AI Accelerator release notes	462
	Legal	475
3.6	Vector Engine (PGvector)	493
3.6.1	Vector Engine	493
3.6.2	Postgres で Vector Search を使用する理由	493
3.6.3	Vector Engine を使用するのは誰ですか?	493
3.6.4	それは何を提供しますか?	494
3.6.5	Vector エンジンの仕組み	494
3.6.6	Vector Engine を使用する必要がある場合	495
3.6.7	ユースケースの例	495
3.6.8	AI Accelerator パイプラインおよび Gen AI Builder との統合	495
3.6.9	次のステップ	496
3.7	AI Factory Release Notes	496
3.7.1	AI Factory 1.3 release notes	496
	リリースハイライト	496
	展開オプション	497
	新機能と機能強化	497
	可観測性、モニタリング、インサイト	498
	サステナビリティ ESG レポートプレビュー	499
	プラットフォームの改善	499
	リソース	499

EDB Postgres® AI は、AI 搭載アプリケーションとインテリジェントデータ製品を構築および運用するためのスケーラブルなモジュール式プラットフォームを提供します。**あなたのデータ** から始まり、**データベース** で、**あなたのデータで実行されますセンターまたはクラウド環境**。EDB PG AI は、最初に **Hybrid Manager™** 用に設計されています。ほとんどの機能は **Hybrid Manager's AI Factory** 内で配信および管理され、そこで統合されたコントロールプレーン、ガバナンス、およびオブザーバビリティを取得します。ライセンスでサポートされている場合、ハイブリッドマネージャーの外部で選択したコンポーネント、特に Postgres 拡張機能と AI DB AIDB を実行することもできます。

その中核では、EDB Postgres® AI EDB PG AI は、**ソブリン AI** を有効にするように設計されており、企業境界内で管理される、信頼できる Postgres データに基づいてインテリジェンスを構築できます。モデル、パイプライン、ベクターインデックス-すべてはあなたによって管理および制御されます。

EDB PG AI を使用すると、次のことができます。

- ****Gen AI agents**** と、エンタープライズ知識と ****Hybrid Manager**** の主権 Postgres データを活用した会話エクスペリエンスを構築して実行します
 - 構造化および非構造化ソースで ****vector search**** および **取得拡張生成 RAG** を実行する
 - 多様なソースからデータを取り込み、処理、および分類する ****document intelligence pipelines**** を作成します
 - GPU アクセラレーションとエンタープライズオブザーバビリティを使用した、スケーラブルな *Model Serving Reference Manual* の展開
 - AI 駆動のクエリパターンと ****vector-native capabilities**** を使用して Postgres を拡張する
 - Postgres、EDB Lakehouse、およびその他の EDB 統合プラットフォーム全体で AI と分析をシームレスに組み合わせる
-

Where AI Factory runs

EDB Postgres® AI は、AI 搭載アプリケーションとインテリジェントデータ製品を構築および運用するためのスケーラブルなモジュール式プラットフォームを提供します。あなたのデータ から始まり、データ ベース で、あなたのデータで実行されますセンターまたはクラウド環境。EDB PG AI は、最初に Hybrid Manager™用に設計されています。ほとんどの機能は Hybrid Manager's AI Factory 内で配信および管理され、そこで統合されたコントロールプレーン、ガバナンス、およびオブザーバビリティを取得します。ライセンスでサポートされている場合、ハイブリッドマネージャーの外部で選択したコンポーネント、特に Postgres 拡張機能と AI DB AIDB を実行することもできます。

その中核では、EDB Postgres® AI EDB PG AI は、ソブリン AI を有効にするように設計されており、企業境界内で管理される、信頼できる Postgres データに基づいてインテリジェンスを構築できます。モデル、パイプライン、ベクターインデックス-すべてはあなたによって管理および制御されます。

EDB PG AI を使用すると、次のことができます。

- **Gen AI agents** と、エンタープライズ知識と **Hybrid Manager** の主権 Postgres データを活用した会話エクスペリエンスを構築して実行します
- 構造化および非構造化ソースで **vector search** および **取得拡張生成 RAG** を実行する
- 多様なソースからデータを取り込み、処理、および分類する **document intelligence pipelines** を作成します
- GPU アクセラレーションとエンタープライズオブザーバビリティを使用した、スケーラブルな *Model Serving Reference Manual* の展開
- AI 駆動のクエリパターンと **vector-native capabilities** を使用して Postgres を拡張する
- Postgres、EDB Lakehouse、およびその他の EDB 統合プラットフォーム全体で AI と分析をシームレスに組み合わせる

チームと環境に適切なオペレーティングモデルを選択します。

- ハイブリッドマネージャー推奨ハブ [AI Factory in Hybrid Manager](#) を使用して、Postgres、パイプライン、ベクトル検索、モデル提供、および Gen AI 全体で AI ワークロードをプロビジョニング、管理、および観察します。
- 外部ハイブリッドマネージャー選択したコンポーネント [Vector Engine \(PGvector\)](#)、Pipelines and data preparation、または AI DB AIDB 使用可能な場合は、互換性のあるプラットフォームで Postgres に焦点を当てた機能を実行します。

ハイブリッドマネージャーの外部で動作している場合、クラスターのライフサイクル、セキュリティ、およびオペラビリティを所有します。展開する前に、ターゲットプラットフォームの互換性、パフォーマンス、およびライセンスを検証します。

1.1 前提条件

必要なものは、実行する場所によって異なります。

- ハイブリッドマネージャーの使用 AI Factory が有効になっており、Gen AI、モデル提供、ベクター検索に必要な資格を備えたハイブリッドマネージャーインスタンスへのアクセス。[AI Factory in Hybrid Manager](#) を参照してください。
- ハイブリッドマネージャーの外部で実行されているコンポーネント拡張機能と AIDB 用にサポートされている Postgres 環境、および該当する場合に機能するパイプラインとモデルに必要なインフラストラクチャ。

正確なコンポーネントのサポートハイブリッドマネージャーの外部、最小バージョン、およびライセンス境界については、EDB 代表およびプラットフォーム所有者と調整してください。

1.2 なぜソブリン AI なのか？

EDB PG AI を使用すると、データの所有権、セキュリティ、ガバナンス要件を尊重するソブリン AI ソリューションを作成できます。

- Postgres データベースと EDB 分析レイクで **あなたのデータ** を操作します
- あなたの制御の下で **モデル** をトレーニングして提供します - 不透明なサードパーティ API は必要ありません
- **必要な場所** に展開します - ハイブリッドマネージャー、オンプレミスデータセンター、またはクラウドテナント
- AI、分析、および運用データを統合された管理対象アーキテクチャに統合する
- AI ワークロードのライフサイクル全体を監査、監視、および制御します取り込みから推論まで

EDB PG AI は、Postgres、最新の AI パイプライン、およびフルガバメントアーキテクチャで提供されるスケラブルモデルを組み合わせることにより、**信頼またはコンプライアンスを損なうことなく** AI イノベーションのロックを解除するのに役立ちます。

1.3 あなたが構築できるもの

EDB PG AI は、Postgres を中心に使用して、**信頼できるデータの上にインテリジェントなアプリケーション**とシステムを作成できるように設計されています。

以下をビルドできます。

- Gen AI および Postgres ネイティブを搭載した**エンタープライズナレッジアシスタント** ****vector search****
- 構造化、半構造化、非構造化データを統合する **AI 搭載の検索ポータル**
- ****Pipelines****、OCR、要約、分類、および抽出を使用した**ドキュメント処理パイプライン**
- *Knowledge Bases Reference Manual* で構築された、内部ユーザーと外部顧客の両方のための**会話アプリケーション**
- 検索、レコメンデーション、NLP、ビジョンのための *Model Serving Reference Manual* を公開する **AI 搭載データアプリ**
- ビジネスシステムとワークフローに埋め込まれた**インテリジェントオートメーション** - データによって駆動されます
- Postgres、Lakehouse、Gen AI を組み合わせて信頼できる回答を提供する **RAG アーキテクチャ**

詳細な例と解決策パターンについては、*AI Factory Use Cases* および *Industry Solutions with AI Factory* を参照してください。

1.4 道を選択してください

目標に基づいてこれらのエントリポイントを使用します Diataxis

- コンセプトとアーキテクチャ *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- ハウツーガイド *Gen AI How-To Guides* および *How-To Deploy AI Models from Model Library*
- チュートリアルと学習 *Learning Paths* — 101、201、および 301
- リファレンス Gen AI reference、*Model Serving Reference Manual*、*Vector Engine (PGvector)*、および Pipelines and data preparation

1.5 対象者、および開始方法

EDB PG AI は、**コア Postgres と分析資産**の上に AI アプリケーションとシステムを構築したい **データドリブン組織**にサービスを提供します。

- Gen AI アプリとインテリジェントなデータエクスペリエンスを構築する **開発者**
- **データエンジニア** ****Pipelines****、****vector search****、およびハイブリッドデータの準備を管理する
- 管理された Postgres 統合環境内で *Model Serving Reference Manual* を使用してモデルを運用化する **AI エンジニア**

- 大規模な AI +分析ソリューションをデザインする データアーキテクト
- エンタープライズグレードの制御と監査可能性を求める IT リーダーとセキュリティチーム

始めるのは簡単です。

- *Learning Paths* を探索して、基礎的なスキルを構築
- *Knowledge Bases Reference Manual* および *Assistants Reference Manual* を使用して ナレッジベース とアシスタント の構築を開始する
- 最初の *Model Serving Reference Manual* を展開し、Postgres クエリーと統合する
- ****pipelines**** および ****vector search**** をビルドして、信頼できるデータに基づいてインテリジェントアプリケーションを強化する

1.6 コア機能

EDB PG AI の各機能領域を探索します。

1.6.1 Gen AI ワークロード

- Gen AI Workloads — 機能の概要
- *Assistants Reference Manual* — ビルドアシスタント、ツール、構造、およびルールセット
- *Knowledge Bases Reference Manual* — ナレッジベースとデータレイクの管理

1.6.2 アセットライブラリーとモデル提供

- *Model Library Reference Manual* — モデルと AI アセットの管理とガバナンス
- *Model Serving Reference Manual* — モデル推論サービスの展開とスケーリング

1.6.3 パイプラインとベクターエンジン

- Pipeline Capabilities — ドキュメントインテリジェンスとベクトル化のためのパイプラインを構築する
- *Vector Engine (PGvector)* ベクトル検索と AI 駆動のクエリパターンを Postgres に統合する

1.7 EDB PG AI とアナリティクス

EDB PG AI は、****EDB Analytics ecosystem**** と深く統合されています。

- AI を活用した検索とインサイトのために構造化データと非構造化データを組み合わせます
- Gen AI エージェントにフィードするレイクハウス支援の RAG パイプラインを構築する
- Postgres および Lakehouse データとともにベクトル埋め込みをネイティブに保存および照会

- Postgres 拡張機能、分析エンジン、AI サービスを活用するハイブリッド AI +分析ソリューションを作成します。すべて EDB PG AI 内で管理されます

これにより、真の **データ駆動型 AI** が可能になります。ここでは、最高の操作データ、トランザクションデータ、分析データが AI システムを強化します。

1.8 学んで構築する

次のリソースを使用して、さらに詳しく説明します。

- *AI Factory Explained* — EDB PG AI のアーキテクチャとコンポーネントを理解する
- *Gen AI How-To Guides* — EDB PG AI ソリューションを構築するための実践ガイド

1.9 EDB PG AI ソリューションを構築する

次の 1 つ以上の手順から今日を始めてください。

- Agent Studio を使用して最初の *Assistants Reference Manual* をビルドする
- Gen AI Builder で *Knowledge Bases Reference Manual* を展開し、RAG と統合
- 最初の *Model Serving Reference Manual* を起動します
- ドキュメントインテリジェンス用に ****Data Pipelines**** を作成します
- Postgres に ****Vector Search**** を追加

エンドツーエンドのパターンとアーキテクチャガイダンスについては、*AI Factory Use Cases* および *Industry Solutions with AI Factory* を参照してください。

EDB PG AI は、自分自身が管理するデータに、データセンターまたはクラウドで AI 搭載ソリューションを構築するための強力なモジュール基盤を提供します。

今日から構築を開始します。

第 2

AI Factory

2.1 AI Factory Hub Map

このページを使用して、必要なものに直接ジャンプします。各タイルは、ハブまたはハイブリッドマネージャースポークの適切な場所にリンクしています。前提条件とコンポーネントが実行される場所については、AI Factory ランディングページを参照してください。

2.1.1 ここから始めます

- 概念 アーキテクチャとコンポーネントを理解する
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory terminology*
- 学習 ガイド付きパスとユースケースに従ってください
- *Learning Paths*
- *AI Factory Use Cases* および *Industry Solutions with AI Factory*

2.1.2 AI Factory でビルドする

- Gen AI アシスタント、ルールセット、ツール、スレッド
- Gen AI hub
- *Customer Service Assistant Implementation Guide*
- パイプライン RAG、ナレッジベース、自動処理
- Pipelines and data preparation
- Knowledge Bases

- Vector Engine Postgres でのベクター検索 - *Vector Engine (PGvector)*
- モデルサービング+ライブラリ
- *Model Serving Reference Manual*
- *Model Library Reference Manual*

2.1.3 ハイブリッドマネージャー

- ハイブリッドマネージャーの AI ファクトリーガバナンド操作、オブザーバビリティ、GPUs
- Spoke root
- Gen AI in HM
- *Pipelines models reference*
- GPU management in HM

2.1.4 実行される場所と前提条件

- AI Factory が実行される場所最初にハイブリッドマネージャー。HM 外部の選択されたコンポーネント See landing page: Where it runs
- 前提条件エンタイトルメント、GPU 該当する場合 - See landing page: Prerequisites

2.1.5 リファレンス

- Gen AI リファレンス- SDK, tools, drivers
- モデルリファレンス - *Model Serving Reference Manual*
- パイプラインリファレンス - Capabilities, PGFS, models

2.2 AI Factory Learning Center

AI Factory Learning Center へようこそ。ここは、**EDB Postgres® AI** EDB PG AI を使用して構築するための中心的な開始点です。

EDB PG AI は、**ソブリン AI** を実現します。データベースとデータセンターのデータを使用して、インテリジェントなアプリケーションとデータ製品を強化します。このプラットフォームは、統合されたベクター検索、パイプライン、Gen AI ビルダー、ナレッジベース、モデル提供を備えた Postgres に直接構築されています。

このラーニングセンターを使用して、次のことを行います。

- 主要な概念とアーキテクチャを理解する
- ステップバイステップのハウツーガイドに従ってください
- 構造化された学習パスを介した進歩
- 完全な実践的なチュートリアル

- ソリューションパターンと実際のユースケースを探索する

2.2.1 クイックスタート

ロールベースのガイドで生産性をすばやく高めます。

- *Create an Assistant in Gen AI Builder* — カスタマーサービスアシスタントをビルドする
- *Data Engineer quickstart* — Pipelines + Vector Engine を使用して RAG パイプラインを立ち上げる
- *How-To Setup GPU resources* — KServe と GPU を使用してプライベートモデルを提供する

2.2.2 概念的な説明

アーキテクチャ、コアコンポーネント、および EDB PG AI の動作を理解するには、ここから始めます。

- *AI Factory Explained*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Knowledge Bases Reference Manual*
- *Model Serving Concepts*
- *Sovereign AI Explained* — ソブリン AI が重要な理由と、EDB PG AI がデータとインフラストラクチャで信頼できるガバナンスされた AI をどのように有効にするか。

2.2.3 ハウツーガイド

EDB PG AI でビルドするためのステップバイステップガイド

- *Gen AI How-To Guides*
- *Create a Knowledge Base in Gen AI Builder*
- *Configure the Data Lake in Gen AI Builder*
- *How-To Create an InferenceService for NVIDIA NIM Container*

2.2.4 ラーニングパス

構造化された学習パス — 初めての方、またはガイド付きの旅をご希望の場合は、ここから始めてください。

- *Learning Paths*
- *Getting Started - AI Factory 101*
- *Building Production-Ready AI Features - AI Factory 201*
- *Advanced AI Factory Usage and Extensibility - AI Factory 301*

2.2.5 ソリューション

エンタープライズニーズに対応したリファレンスアーキテクチャとソリューションパターン

- *Industry Solutions with AI Factory*
- ソリューションパターンの例
- Gen AI を活用した検索ポータル
- 検索拡張生成 RAG を使用したナレッジアシスタント
- インテリジェントな文書処理パイプライン

2.2.6 ユースケース

EDB PG AI の実世界のアプリケーションを探索します。

- *AI Factory Use Cases*
- エンタープライズナレッジアシスタント
- ワークフローのインテリジェントな自動化
- AI を活用したレコメンデーションと検索

ヒント EDB PG AI を初めて使用する場合は、*Getting Started - AI Factory 101* から始めてください。

コア機能をより深く掘り下げることができます。

- Pipelines and data preparation
- *Gen AI Builder*
- *Model Serving Reference Manual*
- *Vector Engine (PGvector)*

このラーニングセンターは継続的に進化します。新しいガイド、チュートリアル、および学習リソースを頻繁にチェックしてください。

EDB Postgres® AI を使用して、インテリジェントなアプリケーションとデータ製品の構築を今すぐ始めます。

Getting Started - AI Factory 101

AI Factory のスキルを次のレベルに引き上げます - 複雑なアシスタント、高度なデータパイプライン、エンタープライズグレードのモデル提供をマスターします。

この道をとるべき人

この中級学習パスは、次のことを目的としています。

- AI 機能を実稼働システムに統合する **アプリケーション開発者**
- **データエンジニア** AI 駆動のナレッジベースの堅牢なパイプラインを構築する
- モデルの提供と可観測性を実装する **AI & プラットフォームエンジニア**
- **Enterprise Teams** ソブリン AI ワークロードを大規模に展開する

前提条件 *Getting Started - AI Factory 101* の完了または基本的な AI Factory の概念に関する同等の経験。

学習成果

このパスを完了すると、以下をマスターします。

OK 高度な AI オーケストレーション — 複雑なペルソナ、永続メモリ、ツール統合を備えたマルチステップアシスタントを構築します

OK エンタープライズデータパイプライン — マルチソース RAG、メタデータフィルタリング、ハイブリッド検索を使用してハイブリッドナレッジベースをデザインする

OK Production Model Serving — パフォーマンスチューニングを施した KServe を使用して、GPU 搭載モデルを展開および最適化します

OK オペレーショナルエクセレンス — AI 駆動型アプリケーションの包括的な可観測性、モニタリング、およびガバナンスを実装します。

推定合計時間 2〜3 時間 | 難易度 中級

ラーニングパスモジュール

1. 高度なアシスタントと構造デザイン

洗練された AI オーケストレーションパターンをマスター

学習する内容

- 微妙なペルソナと状況記憶を備えたデザインアシスタント
- エラー処理を使用して信頼性の高いマルチステップ構造を構築する
- 外部ツールとデータソースを安全に統合する

時間投資 30〜45 分

モジュール

1. **** Assistants Reference Manual **** — アシスタントアーキテクチャ、ペルソナデザイン、およびメモリ管理の詳細
 2. **** Structures Reference Manual **** — 分岐ロジックと検証パターンを使用したマスターワークフローオーケストレーション
 3. **** Create an Assistant in Gen AI Builder **** — ハンズオン統合ツールとナレッジベースを使用して制作アシスタントを構築する
 4. **** Create a Structure in Gen AI Builder **** — ハンズオン条件付きロジックとエラーリカバリーを使用して複雑なワークフローを実装する
 5. **** Create a Tool in Gen AI Builder **** — ハンズオン内部 API を管理された安全な AI ツールとしてラップする
-

2. データエンジニアリングとハイブリッドナレッジベース

AI 用のエンタープライズグレードのデータパイプラインを構築

学習する内容

- 最適なパフォーマンスとコストを実現するハイブリッドナレッジベースをアーキテクト
- マルチプルデータソースを使用して高度な RAG パターンを実装する
- メタデータのフィルタリングとランキングを使用したインテリジェントな検索を構成します

時間投資 30～45 分

モジュール

1. **** Best practices for Hybrid Knowledge Bases **** — データの鮮度、カバレッジ、運用コストのバランスを保つための実証済みのパターン
 2. **** Manage Knowledge Bases in Gen AI Builder **** — 運用ワークフロー再埋め込み戦略、ソースライフサイクル管理、およびアクセス制御
 3. **** Create a Retriever in Gen AI Builder **** — ハンズオン検索動作の微調整、高度なフィルタリングの実装、およびランキングアルゴリズムの最適化
-

3. KServe を使用したモデルサービング

GPU 搭載 AI モデルの展開とスケーリング

学習する内容

- GPU インフラストラクチャに高性能モデルを展開する
 - リソースの割り当てとランタイム構成を最適化する
 - 完全なモデル展開のライフサイクルを管理する
-

時間投資 45～60 分

モジュール

1. ***Model Serving Concepts*** ** — KServe アーキテクチャと展開パターンを理解する
2. ***How-To Configure a ClusterServingRuntime*** ** — 共有構成を使用してモデルファミリーの最適化されたランタイムを定義する
3. ***How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe*** ** — ハンズオン NVIDIA NIM を活用して推論パフォーマンスを加速する
4. ***How-To Update GPU Resources for an InferenceService*** ** — ハンズオン最適なコストパフォーマンスを実現する適切なサイズの GPU と CPU リソース
5. ***How-To Verify InferenceServices and GPU usage*** ** — エンドポイントを検証し、ハイブリッドマネージャーを介して GPU 使用率を監視する

4. 可観測性とプロダクションモニタリング

信頼性の高い監視された AI オペレーションを保証

学習する内容

- AI パイプラインの包括的な可観測性を実装する
- モデルのパフォーマンス、リソース使用量、およびシステムの状態を監視します
- 実稼働の準備状況と SLA コンプライアンスを確立する

時間投資 20～30 分

モジュール

1. ***Observability for Model Serving*** ** — モデルとデータパイプラインのメトリック、ロギング、およびアラートを構成する
2. ***How-To Monitor deployed models with KServe*** ** — 実稼働環境でのエンドポイントの状態、レイテンシー、およびスループットを追跡する
3. ***Observability for Model Serving*** ** — プラットフォームのダッシュボードと自動警告システムを活用する

学習の旅を続けます

次は何ですか？

高度なパターンの準備はできましたか? ***Advanced AI Factory Usage and Extensibility - AI Factory 301*** **
に進んで、以下をマスターしてください

- マルチエージェントオーケストレーションと複雑なワークフロー

- 高度な埋め込みパイプラインとベクトル最適化
- エンタープライズガバナンスとコンプライアンスフレームワーク
- 大容量 AI アプリケーションのスケーリングパターン

追加リソース

理解が深まる

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
 - Hybrid Manager: Gen AI Builder Guide
 - *Sovereign AI Implementation Guide*
 - *Model Serving Concepts*
 - *Structures Reference Manual*
-

エンタープライズ AI を構築する準備はできていますか？

この 201 パスを完了して、**EDB Postgres AI** を使用してモデル、データパイプライン、および運用インフラストラクチャを完全に制御しながら**プロダクショングレードのソブリン AI アプリケーション**を展開するために必要な専門知識を取得します。

最初のモジュールを開始します *Assistants Reference Manual*

Building Production-Ready AI Features - AI Factory 201

これは誰に向けていますか

- Gen AI 機能を運用アプリケーションに統合する開発者
- AI 駆動型のナレッジベースのパイプラインを構築するデータチーム
- モデルの提供と可観測性を構成する AI チームとプラットフォームチーム
- 大規模なソブリン AI ワークロードを有効にするハイブリッドマネージャーユーザー

一般的な目標

このパスを完了すると、次のことができます。

- マルチステップ AI アシスタント と高度な構造を構築する
- ハイブリッドナレッジベースおよびマルチソース RAG パイプラインのデザイン
- GPU 搭載のモデルサービング の展開と管理
- AI 駆動型機能の 可観測性とモニタリング を実装する
- 実稼働グレードのガバナンスとパフォーマンスチューニングのパターンを学習する

フォーカスエリア別のモジュール数

1. 高度なアシスタントと構造デザイン

目標

- 複雑なペルソナとメモリを備えたアシスタントを実装する
- 高度なマルチステップ構造を作成する
- ツールと外部データフローを統合する

推定時間 ～30～45 分

モジュール

1. *Assistants Reference Manual* —ペルソナ、メモリ、ツールの使用パターンを備えたデザインアシスタント。
2. *Structures Reference Manual* —取得とアクションを信頼できるフローにチェーンする方法を学びます。
3. *Create an Assistant in Gen AI Builder* —ツールと KB を使用して実稼働対応アシスタントをビルドします。
4. *Create a Structure in Gen AI Builder* —検証と分岐を使用したマルチステップワークフローを実装します。
5. *Create a Tool in Gen AI Builder* —内部サービスと API を安全で管理されたツールとしてラップします。

2. データエンジニアリングとハイブリッドナレッジベース

目標

- ハイブリッドナレッジベースのデザインと管理
- マルチソース RAG パイプラインを調整する
- メタデータフィルタリングとハイブリッド検索を実装する

推定時間 ～30～45 分

モジュール

1. *Best practices for Hybrid Knowledge Bases* —鮮度、カバレッジ、コストのバランスを保つパターン。
2. *Manage Knowledge Bases in Gen AI Builder* — KB の操作: 再埋め込み、ソースライフサイクル、およびアクセス制御。
3. *Create a Retriever in Gen AI Builder* —検索動作、フィルター、およびランキングを大規模に調整します。

3. KServe を使用したモデルサービング

目標

- GPU 搭載モデルの展開
- ランタイムとリソース設定を調整する
- Model Serving のライフサイクルを理解する

推定時間 約 45～60 分

モジュール

1. *Model Serving Concepts*
2. *How-To Configure a ClusterServingRuntime* — 共有設定を使用してモデルのホストファミリーにランタイムを定義します。
3. *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe* — 高性能推論のために NVIDIA NIM イメージを使用します。
4. *How-To Update GPU Resources for an InferenceService* — パフォーマンスとコストの適切なサイズの GPU/CPU。
5. *How-To Verify InferenceServices and GPU usage* — ハイブリッドマネージャーでのエンドポイントと GPU の使用率を検証します。

4. 可観測性とモニタリング

目標

- AI パイプラインとモデルサービスの可観測性を実装する
- パフォーマンスとリソースの使用状況を監視する
- 実稼働準備状況チェックを有効にする

推定時間 約 20～30 分

モジュール

1. *Observability for Model Serving* — モデルとパイプラインのメトリックとログ。
2. *How-To Monitor deployed models with KServe* — サービス提供エンドポイントの状態とパフォーマンスを追跡します。
3. *Observability for Model Serving* — プラットフォームレベルのダッシュボードとアラート。

次のステップ

この 201 パスを完了した後

- *Advanced AI Factory Usage and Extensibility - AI Factory 301* に進む — AI アプリのスケールリング、マルチエージェントオーケストレーション、埋め込みパイプライン、および高度なガバナンスのための高度なパターン。

関連する学習リソース

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Hybrid Manager: Using Gen AI Builder*
- *Sovereign AI Explained*
- *Model Serving Concepts*
- *Structures Reference Manual*

この 201 パスをマスターすることにより、**EDB PG AI** を使用して、モデル、パイプライン、およびプロダクションオブザーバビリティを完全に制御しながら、**ソブリン AI** アプリケーションを展開およびスケールリングする準備が整います。

Advanced AI Factory Usage and Extensibility - AI Factory 301

これは誰に向けていますか

- **AI ファーストアプリケーション** とエージェントを構築する上級開発者
- **大規模なモデルサービング** を管理する AI OPs チーム
- AI Factory を使用して **拡張機能と統合** を構築するプラットフォームエンジニア
- **エンタープライズグレードのソブリン AI** ソリューションを構築するハイブリッドマネージャーユーザー

一般的な目標

このパスを完了すると、次のことができます。

- ツール呼び出しと高度な RAG パイプラインを使用して、複雑なエージェントアシスタント を構築する
- カスタムランタイム、トランスフォーマー、説明者を使用して**モデルサービス** を拡張する
- API を使用して **AI Factory** パイプラインを自動化およびオーケストレーションする

- プロダクション AI の 高度な可観測性とガバナンス をデザインする
- AI Factory を エンタープライズシステムおよび既存の AI パイプラインと統合する

フォーカスエリア別のモジュール数

1. エージェントアシスタントと高度なツール

目標

- マルチステップのツールの使用を推論し、動作するビルドアシスタント
- 外部 API 統合のための高度なツールを実装する
- チェーンツールと管理コンテキストフロー

推定時間 約 45～60 分

モジュール

1. *Assistants Reference Manual* —ツール全体で推論、取得、および動作を行うアーキテクトアシスタント。
2. *Tools Reference Manual* —内部または外部システムを呼び出す管理対象アクションをデザインします。
3. *Create a Tool in Gen AI Builder* —エージェントワークフローのための堅牢で再利用可能なツール統合を実装します。
4. *Structures Reference Manual* —マルチステップのパイプラインとエージェントの動作を構築します。

2. モデル提供の拡張

目標

- 特殊なモデルのカスタム **ServingRuntime** 定義を作成する
- **Transformers** および **Explainers** をモデル提供パイプラインに追加します
- KServe ベースのサービスの高度な展開パターンを理解する

推定時間 約 45～60 分

モジュール

1. *Model Serving Concepts*
2. *How-To Configure a ClusterServingRuntime* —特殊なモデルと設定のカスタムランタイムを作成します。
3. *How-To Deploy AI Models from Model Library* (ブレースホルダー —カスタムの *Transformers/Explainers* パターンで展開します)

3. 大規模な可観測性とガバナンス

目標

- AI パイプラインとモデルサービスのためのエンタープライズグレードの**可観測性**を実装する
- ツールの使用状況 とパイプラインパフォーマンスをモニタする
- ハイブリッド AI ワークロードのカスタムダッシュボードを構築する

推定時間 ～30～45 分

モジュール

1. *Observability for Model Serving* — サービス提供とパイプラインのメトリック、ログ、およびトレース。
2. *How-To Monitor deployed models with KServe* — サービス提供エンドポイントの状態とパフォーマンス。
3. *Observability for Model Serving*

4. 自動化と API 統合

目標

- ハイブリッドマネージャーと AI ファクトリー API を使用してワークフローを自動化する
- AI Factory パイプラインを CI / CD および AI システムに統合する
- API 駆動制御を介して大規模な AI ファクトリー展開を管理

推定時間 約 20～30 分

モジュール

1. *Hybrid Manager API Overview*
2. *Using the Hybrid Manager API*

次のステップ

この 301 パスを完了した後

- エンタープライズ規模で **ソブリン AI アプリケーション**を設計する能力が備わります。
- AI Factory、Hybrid Manager、および既存の AI パイプラインとの高度な統合パターンを理解します。
- 高度なワークフローと再利用可能なコンポーネントツール、構造、トランスフォーマーを AI Factory エコシステムに貢献する準備が整います。

関連する学習リソース

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Sovereign AI Explained*
- *Hybrid Manager: Using Gen AI Builder*
- *Structures Reference Manual*
- *Model Serving Concepts*

この 301 パスをマスターすることにより、**EDB PG AI** と **Hybrid Manager** を使用して、完全に制御の下で、組織全体で**高度な AI イノベーション**を推進する準備が整います。

AI Factory Explained

AI Factory Explained へようこそ — **EDB Postgres® AI** EDB PG AI のアーキテクチャ、コンセプト、およびコンポーネントを詳しく説明します。

EDB PG AI は、**ソブリン AI** を実現します。基盤として Postgres を使用して、データベースとデータセンターに、データの上にインテリジェントなアプリケーションとデータ製品を構築します。このプラットフォームは、ベクター検索、パイプライン、Gen AI ビルダー、ナレッジベース、およびモデルサービングを統合し、AI をシステムに埋め込む方法を完全に制御します。

このセクションでは、すべてがどのように動作するかについて説明します。

- EDB PG AI の根底にある重要なコンセプトは何ですか
- アーキテクチャがどのように適合するか
- Pipelines、Gen AI Builder、Model Serving などの個々の機能の仕組み
- ベクター検索と AI モデルが Postgres 資産内に収まる場所

コアコンセプト

プラットフォームを理解するには、ここから開始します。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory terminology*
- *Fundamental AI Concepts*

Gen AI とナレッジベース

データに基づいてインテリジェントな知識がどのように構築されるかを理解します。

- *Knowledge Bases Reference Manual*
- *Retrievers Reference Manual*

- [Rulesets Manual](#)
- [Structures Reference Manual](#)
- [Threads Reference Manual](#)
- [Tools Reference Manual](#)

データパイプラインと AI アクセラレーター

パイプラインが AI データのワークフローをどのように変換するかを理解します。

- [AI Accelerator Pipelines overview](#)
- [Data Lake Reference Manual](#)

Vector Search と PGvector

ベクトル検索を Postgres に直接埋め込む方法

- [PGvector and Vector Engine concepts](#)

モデルの提供

AI モデルがどのように展開、管理、および使用されるか。

- [Model Serving Concepts](#)
- [Model Library Explained In Depth](#)

次に行く場所

ビルドする準備ができている場合

- [Learning Paths](#) で始める
- [Gen AI How-To Guides](#) をフォローする

または、[Industry Solutions with AI Factory](#) および現実世界の [AI Factory Use Cases](#) で、より高度なソリューションパターンを探索します。

このセクションは、EDB PG AI の進化に応じて成長します。新機能が追加されたら、最新の説明をチェックしてください。

AI Factory How-To Guides

これらのガイドを使用して、AI Factory 機能を展開および操作します。ハイブリッドマネージャー推奨で実行する場合、各ガイドの HM 固有の注意事項に従ってください。

トピック

Gen AI

- [Gen AI Builder でデータソースの構成 — ハブの Gen AI リファレンスを参照してください](#)
- [Gen AI Builder でデータレイクを構成する — ハブの Gen AI リファレンスを参照してください](#)

モデルサービング

- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe](#)
- [How-To Monitor deployed models with KServe](#)
- [How-To Update GPU Resources for an InferenceService](#)
- [How-To Configure a ClusterServingRuntime](#)
- [Accessing KServe endpoints \(internal and external\)](#)

モデルライブラリー

- [How-To Integrate Private Registry with Model Library](#)
- [How-To Define Repository Rules for Model Library](#)
- [How-To Manage Repository and Image Tag Metadata](#)
- [How-To Deploy AI Models from Model Library](#)

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。

- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- [Model Serving Reference Manual](#)

Learning Paths

AI Factory Learning Paths は、AI Factory を使用して **AI 搭載機能** と **インテリジェントアプリケーション** を完全な制御とガバナンスで構築するのに役立つ構造化されたガイダンスを提供します。

実稼働環境で **ソブリン AI** ソリューションを始めたばかりか、スケーリングしているかにかかわらず、これらのパスは、主要な概念、ツール、および実際の実装をガイドし、進行に応じてエクスペリエンスを構築します。

ラーニングパスの使用方法

各パスは **セルフペース** でモジュール式です。

次のことができます。

- 完全なパス **シークエンシャル** 101 → 201 → 301、または

- **ロール と プロジェクトのニーズ**に基づいてトピックを掘り下げます。

パスごとに

- **前提条件**を確認して、準備ができていることを確認します。
- 次の学習フローに従います。
- **コンセプト → ハウツーガイド → チュートリアル → 実践**
- 既存の AI Factory 環境またはサンドボックスプロジェクトを使用して、ハンズオンを練習します。

ラーニングパス

Getting Started - AI Factory 101

対象者 新しいユーザー、開発者、データエンジニア、アーキテクト。**推定時間** 1～2 時間。**前提条件** 基本的な AI 概念と Web アプリケーションの使用に精通していること。

次のことを学びます

- AI Factory のコア概念を理解する
- AI ファクトリーとハイブリッドマネージャーをナビゲート
- 最初の AI アシスタントを作成します
- ナレッジベースとレトリバーを接続する
- インタラクションスレッドの実行とレビュー
- AI Factory の主要な用語を理解する

Getting Started - AI Factory 101

Building Production-Ready AI Features - AI Factory 201

対象者 プロダクション AI 機能を構築する開発者、データエンジニア、ソリューションアーキテクト。**推定時間** 2～4 時間。**前提条件** 101 パスの完了、Kubernetes の概念の基本的な知識。

次のことを学びます

- 高度な検索のための **ハイブリッドナレッジベース** をアーキテクト
- **ルールセット** とガバナンスパターンを使用および管理する
- 高度な**アシスタント** とマルチソースレトリバーを構築する
- KServe を使用して **GPU 搭載モデルサービング** を実装する
- モニタリングと **可観測性のベストプラクティス** を適用する

- 実稼働の準備状況とスケーリング戦略の実装

Building Production-Ready AI Features - AI Factory 201

Advanced AI Factory Usage and Extensibility - AI Factory 301

対象者 AI プラットフォームの所有者、高度な開発者、AI エンジニア、アーキテクト。**推定時間** 4～6 時間
マイペースで高度なトピック。**前提条件** 201 パスの完了、Kubernetes およびコンテナベースの AI ワーク
ロードの経験。

次のことを学びます

- デザイン エージェントアシスタント と高度な構造
- カスタムツールの開発と展開
- カスタムサービングランタイム を使用してモデルサービスを拡張する
- モデルの説明可能性 と責任ある AI パターンを実装する
- API 駆動のワークフロー を介して AI Factory パイプラインを自動化する
- 高度な可観測性 とパフォーマンスチューニングを適用する

Advanced AI Factory Usage and Extensibility - AI Factory 301

おすすめトレーニングコース

これらの自分のペースでの学習パスを補完するために、以下も提供しています。

インストラクター主導のトレーニング

- 高度な AI ファクトリーアーキテクチャ
- AI ファクトリーの管理とオペレーション
- AI Factory を使用したカスタムモデルの開発と展開

自分のペースで行うトレーニング

- AI ファクトリーの紹介
- AI Factory を使用して AI アシスタントを構築する
- AI モデルとハイブリッドナレッジベースの管理
- AI Factory を使用した AI ワークロードのスケーリング

次はどこに行きますか？

- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- [Gen AI How-To Guides](#)
- [Model Serving Quickstart](#)

Industry Solutions with AI Factory

AI Factory は、エンタープライズデータ、AI パイプライン、モデルサービング、会話エージェントを組み合わせて、業界全体で信頼できるスケーラブルな **AI 搭載アプリケーション** を構築するための柔軟なプラットフォームを提供します。

このページでは、**一般的なソリューションパターン** と **業界別の例** を強調表示し、それらを構築するのに役立つ関連する [Gen AI How-To Guides](#) および [Learning Paths](#) へのリンクを示します。

より技術的なパターンについては、[AI Factory Use Cases](#) を参照してください。

金融サービス

AI Factory は、金融サービス組織がカスタマーエクスペリエンスと内部業務に**安全でコンプライアンスの AI** を提供するのを支援します。

目標の例

- 独自の知識に基づいた **自動エージェント** でカスタマーサービスを向上させます。
- アドバイザーとスタッフが安全で説明可能な **内部ナレッジ検索** を有効にします。
- 文書量の多いワークフロー KYC、オンボーディングを自動化します。

ソリューションの例

- クライアントサポートチャットボット [Create an Assistant in Gen AI Builder](#) によって提供され、[Create a Knowledge Base in Gen AI Builder](#) によって支援され、[Create a Ruleset in Gen AI Builder](#) によって管理されます。
 - [Create a Retriever in Gen AI Builder](#) と [Create a Knowledge Base in Gen AI Builder](#) を組み合わせた **内部ナレッジ検索**、Pipelines and data preparation を介してオプションで強化されます。
 - Pipelines and data preparation による **自動ドキュメント処理**、[Preparers perform OCR operation examples](#)、[Preparers summarize text operation examples](#)、および AI 駆動の抽出を使用。
-

ヘルスケアとライフサイエンス

AI Factory は、臨床および運用上のユースケースのために**規制されたコンプライアンス AI**をサポートします。

目標の例

- **文献対応エージェント**を使用して臨床意思決定のサポートを向上させます。
- 複雑な医療文書の **要約** を自動化します。
- パーソナライズされた **患者対応アシスタント** を提供します。

ソリューションの例

- *Create a Knowledge Base in Gen AI Builder*、最適化された *Create a Retriever in Gen AI Builder*、および会話型 *Create an Assistant in Gen AI Builder* を使用した**医学文献検索**。
 - Pipelines and data preparation を介した文書準備と *Model Serving Reference Manual* を使用した高パフォーマンスの推論を使用した**臨床文書要約**。
 - *Create an Assistant in Gen AI Builder*、準拠の *Create a Ruleset in Gen AI Builder*、および *Create a Tool in Gen AI Builder* を介した外部統合で構築された**患者サポートアシスタント**。
-

小売と電子商取引

AI Factory は、小売および e コマースアプリケーションで**パーソナライズされた AI**を活用した**エクスペリエンス**を可能にします。

目標の例

- **セマンティック商品検索** エクスペリエンスを提供します。
- チャンネル全体で **パーソナライズされたレコメンデーション**を強化します。
- **カスタマーサポートワークフロー** を自動化します。

ソリューションの例

- *Model Serving Reference Manual* を介して提供される埋め込みを使用した**セマンティック製品検索**、調整された *Create a Retriever in Gen AI Builder* を使用して照会されます。
 - *Create an Assistant in Gen AI Builder*、*Create a Retriever in Gen AI Builder*、および *Create a Tool in Gen AI Builder* との**リアルタイム API 統合** を組み合わせた**パーソナライズされたレコメンデーション**。
 - 会話型 *Create an Assistant in Gen AI Builder*、*Create a Knowledge Base in Gen AI Builder*、およびブランドセーフ *Create a Ruleset in Gen AI Builder* を使用した **カスタマーサービスオートメーション**。
-

製造業と産業

AI Factory は、**AI 駆動の知識と自動化**を製造現場と現場業務にもたらしめます。

目標の例

- エンジニアリングの知識を表面化する **メンテナンスアシスタント**を構築します。
- **技術マニュアルと SOP の検索** を有効にします。
- **ドキュメントの理解** とレポートを自動化します。

ソリューションの例

- **知識主導型のメンテナンスアシスタント** *Create an Assistant in Gen AI Builder* で構築され、*Create a Knowledge Base in Gen AI Builder* に接続され、API ベースの *Create a Tool in Gen AI Builder* によって駆動されます。
 - 文書処理に Pipelines and data preparation を使用した**マニュアルと SOP にわたるドキュメント検索**文書処理に Pipelines and data preparation を使用し、調整された *Create a Retriever in Gen AI Builder* を使用した RAG。
-

公共部門と政府

AI Factory は、政府および公共部門の組織が機密性の高いコンテキストで**透明で監査可能な AI**を提供するのを支援します。

目標の例

- **内部ポリシー検索とコンプライアンスチェック** をサポートします。
- **パワー 市民向けサービスポータル**。
- ガバナンスを備えた **知識ベースの AI エクスペリエンス** を有効にします。

ソリューションの例

- 信頼できる *Create a Knowledge Base in Gen AI Builder*、*Create a Retriever in Gen AI Builder*、および監査可能な *Create an Assistant in Gen AI Builder* を使用した**内部ポリシー検索とコンプライアンス QA**。
 - 会話型 *Create an Assistant in Gen AI Builder*、*Create a Tool in Gen AI Builder* を介した外部システム統合、および管理された *Create a Ruleset in Gen AI Builder* で構築された**シチズンサポートポータル**。
-

ソリューションを構築する

AI Factory は、以下を組み合わせる **モジュールアーキテクチャ**を提供します。

- 会話型 AI 用の **** *Create an Assistant in Gen AI Builder* **** および **Agent Studio**。

- RAG およびセマンティック検索用の **** Create a Knowledge Base in Gen AI Builder **** および **** Create a Retriever in Gen AI Builder ****。
 - **** Create a Ruleset in Gen AI Builder **** ガバナンスとコンプライアンス用。
 - **** Create a Tool in Gen AI Builder **** 外部システムとビジネスロジックを接続するため。
 - **** Model Serving Reference Manual **** スケーラブルで実稼働対応の推論用。
 - **** Pipelines and data preparation **** AI 駆動アプリケーションのデータの準備用。
-

詳細はこちら

- [Gen AI How-To Guides](#) — ステップバイステップのビルディングブロック
- [Learning Paths](#) — 構造化されたスキル構築の過程
- [AI Factory Use Cases](#) — ソリューションに適応するための一般的な技術パターン

→ 始める準備はできていますか？ [Learning Paths](#) を選択するか、関連する [Gen AI How-To Guides](#) を探索します。

AI Factory は、信頼できるスケーラブルな **AI** を安全にガバナンスの下で業界に導入するための基盤を提供します。

AI Factory Use Cases

AI Factory は、ソブリン AI ソリューションの構築を支援するように設計されています。ここでは、データ、モデル、および AI 駆動アプリケーションが、ガバナンスの下で、インフラストラクチャで、信頼できるコンテンツ。

これらのユースケースは、AI Factory コンポーネントがどのように連携して、業界やソリューションパターンを超えて現実の運用対応ソブリン AI アプリケーションをサポートする方法を強調しています。

業界固有のソリューションの例については、[Industry Solutions with AI Factory](#) を参照してください。

取得拡張生成 RAG

RAG は、ソブリン AI の基本的なパターンです。LLMs とエンタープライズデータを組み合わせることで、正確で根拠のある応答を提供できます。

何

- 不透明なベースモデルのみに依存するのではなく、ナレッジベースから信頼できるコンテンツを取得する AI モデルを使用します。
- 応答を **auditable** および **explainable** にする — ソブリン AI のキー。

なぜ

- 幻覚を減らします。
- コンプライアンスとトレーサビリティを維持します。
- AI の回答が最新のビジネス知識を反映していることを確認します。

どのように

- データソースから *Create a Knowledge Base in Gen AI Builder* をビルドします。
- セマンティック検索用に調整された *Create a Retriever in Gen AI Builder* を実装します。
- Power *Create an Assistant in Gen AI Builder* および RAG パイプラインを使用するアプリ。

詳細はこちら

- *Sovereign AI Explained*
- *Best practices for Hybrid Knowledge Bases*

Document Intelligence パイプライン

ドキュメントの **ガバナンス境界内** にある **構造化された照会可能なナレッジ** への変換を自動化します。

何

- **OCR**、**解析**、**要約**、および **メタデータ抽出**を適用します。
- 結果を *Create a Knowledge Base in Gen AI Builder* または **Vector Engine** に保存します。

なぜ

- 機密文書をインフラストラクチャに保存します。サードパーティの API 呼び出しは必要ありません。
- コンプライアンスが重視されるドメインの説明可能なパイプラインを構築します。

どのように

- Pipelines and data preparation を使用します。
- ビルトイン *Pipelines reference - Preparers* OCR、PDF の解析、HTML の解析を活用します。
- 検索または RAG のインデックス結果。

詳細はこちら

- *Sovereign AI Concepts*
- Pipeline Getting Started
- *Document Processing Examples*

会話アシスタントとチャットボット

完全な可視性と制御を備えた **インフラストラクチャ** 内で実行され、**モデル** を使用し、**ナレッジベース** に依存するビルドアシスタント。

何

- **Model Serving** と **RAG パイプライン** を活用した AI アシスタント。
- *Create a Ruleset in Gen AI Builder* で定義および管理されます。

なぜ

- LLM 応答のパブリック API サービスに依存しないようにします。
- コントロールトーン、スタイル、コンプライアンス - ソブリン AI に不可欠。

どのように

- Agent Studio で *Create an Assistant in Gen AI Builder* をビルドします。
- Rulesets (optional) による動作の管理。
- *Create a Tool in Gen AI Builder* を追加して、内部システムを安全に統合します。

詳細はこちら

- ハイブリッドマネージャーの Agent Studio
- *Sovereign AI Explained*
- *Getting Started - AI Factory 101*

エンタープライズコンテンツ全体のセマンティック検索

コンテンツで **自然言語検索** を有効にしますが、埋め込みと検索データは**コントロール内** のままです。

何

- **Knowledge Bases** または **Vector Engine** でのベクターベースの検索。
- テキスト、ハイブリッド構造化/非構造化データ、画像をサポートします。

なぜ

- エンベディングは、サードパーティのベクター DB ではなく、データベースまたはオブジェクトストレージに保持します。
- 監査可能性とセキュリティを維持します。

どのように

- *Create a Knowledge Base in Gen AI Builder* をビルドします。
- *Create a Retriever in Gen AI Builder* を使用して、検索をアシスタントまたはアプリに公開します。
- データベース内検索には *Vector Engine (PGvector)* を使用します。

詳細はこちら

- Vector Engine Hub
- *Knowledge Bases Reference Manual*
- *Building Production-Ready AI Features - AI Factory 201*

リアルタイムモデル推論 API

不透明なサードパーティ API を介してではなく、ガバナンスの下で **Kubernetes クラスター内** にモデルを展開して提供します。

何

- **KServe ベースのモデルサービング**を介して LLM、エンベディング、ビジョンモデルを提供します。
- *Model Library Reference Manual* から取得されたモデル、*How-To Create an InferenceService for NVIDIA NIM Container* を介してデプロイされました。

なぜ

- モデルスタックとモデルバージョンの完全な制御。
- 必要に応じて、**GPU アクセラレーションのエアギャップ環境**で推論を実行します。

どのように

- *Model Library Reference Manual* でモデルを管理します。
- *Model Serving Reference Manual* で展開します。
- ハイブリッドマネージャー制御を使用してエンドポイントを保護します。

詳細はこちら

- *Model Serving Concepts*
- *Sovereign AI Explained*
- *Advanced AI Factory Usage and Extensibility - AI Factory 301*

自動化された AI 搭載 ETL パイプライン

AI データパイプラインを制御された環境内に維持します—データ漏洩はありません。

何

- *Create a Structure in Gen AI Builder* とパイプラインを使用して実行します。
- AI の分類
- ドキュメントの概要
- メタデータの抽出
- データクレンジング

なぜ

- 機密データをクラウド API に送信しないでください。
- 再利用可能で説明可能な ETL コンポーネントをビルドします。

どのように

- *Create a Structure in Gen AI Builder* をビルドします。
- *Create an Assistant in Gen AI Builder* またはスタンドアロンのパイプラインでツールとして実行します。
- ハイブリッドマネージャーを介してパイプラインの実行を監視および管理します。

詳細はこちら

- *Structures Reference Manual*
- *Advanced AI Factory Usage and Extensibility - AI Factory 301*

概要

ソブリン AI のユースケースは、データ、モデル、AI アプリケーションをガバナンスの下で維持することに関するものです。

- エンタープライズデータに基づいた **RAG パイプライン** を構築する
- ドキュメントインテリジェンスワークフローを自動化する
- インフラストラクチャに会話型アシスタントを展開する
- GPU によるモデル提供を介して AI モデルを提供
- 環境からデータを残さずに AI を活用した ETL パイプラインを実行します
- 埋め込みの完全な制御でエンタープライズセマンティック検索を強化

次のステップ

- *Industry Solutions with AI Factory* を探索する
- *Learning Paths* をフォローする
- *Gen AI How-To Guides* をはじめる

AI Factory は、信頼でき、管理され、説明可能なソブリン AI をあなたの条件に応じて構築するための基盤を提供します。

→ *Getting Started - AI Factory 101* で今日を始めるか、最初の *Create an Assistant in Gen AI Builder* を展開します。

2.3 Concepts

EDB Postgres® AI EDB PG AI は、AI ワークロードを Postgres プラットフォームとインフラストラクチャにもたらしめます。ハイブリッドおよびクラウドネイティブ環境でソブリン AI を強化するように設計されたアーキテクチャを使用しています。

AI Factory は Postgres を拡張して以下をサポートします

- ベクトル検索とセマンティック検索
- エンベディング生成のためのデータパイプライン
- Gen AI アプリケーション開発
- スケーラブルなモデルの提供と推論
- AI 駆動のガバナンスと可観測性

AI Factory は、データ、モデル、アプリケーションを使用して、信頼できるエンタープライズグレードの AI システムを構築および運用するのに役立ちます。

2.4 Concepts

2.4.1 Postgres とハイブリッドマネージャーを使用した AI に関する EDB のビジョン

EDB の AI ファクトリー戦略により、Postgres は AI ワークロードと AI 駆動アプリケーションのためのファーストクラスのプラットフォームになりました。

このアーキテクチャは以下を組み合わせます。

- Postgres の AI ベクトル検索、埋め込みパイプライン、およびデータベース内の AI 機能
- Gen AI アプリ Gen AI Builder を使用した会話型およびタスク駆動型 AI アプリケーション
- スケーラブルな推論 Kubernetes と KServe を使用した GPU アクセラレーションモデルサービング
- 統合コントロールプレーンハイブリッドマネージャーでのモデルのライフサイクル管理と可観測性

これにより、**ソブリン AI** が有効になります。完全なガバナンスを備えたデータ、インフラストラクチャ上の AI。

2.5 Concepts

RAG パイプラインは、以下を組み合わせることにより LLM 応答を強化します。

- *Knowledge Bases Reference Manual* からのドメイン固有の埋め込み
- *Retrievers Reference Manual* を使用した設定可能な検索動作
- *Model Serving Reference Manual* を介したスケーラブルモデルの推論
- *Gen AI Builder* を介したアプリケーションオーケストレーション

これにより、より正確で根拠のある、コンプライアンスの AI アプリケーションが可能になります。

2.6 Concepts

AI Factory は、以下を提供する堅牢なモデルを提供します。

- LLM、埋め込みモデル、およびビジョンモデルのスケーラブルな推論サービス
- Kubernetes + KServe を介した GPU 高速化サービング
- *Model Serving Reference Manual* フレームワークを介した柔軟な展開オプション
- *Model Library Explained In Depth* によるライフサイクル管理と検出可能性

これらのサービスは、**Gen AI Builder エージェント** と **ナレッジベース** の両方を強化します。

2.7 Concepts

AI Factory Pipelines and data preparation は自動化します

- エンベデッド生成
- コンテンツのチャンク、解析、要約
- *Pipeline auto-processing* で埋め込みを最新の状態に保つ

再利用可能なデータ準備コンポーネントについては、*Pipelines reference - Preparers* を参照してください。

2.8 Concepts

2.8.1 概要

EDB PG AI により、次のことができます。

- AI 機能を Postgres に直接埋め込む
- スケーラブルな実稼働対応 AI アプリケーションを構築する
- 統合されたガバナンスを使用してインフラストラクチャでソブリン AI を動作させる

- AI スタック全体でオープンソースツールと標準を統合する

次のステップ

- [AI Factory terminology](#)
 - [Fundamental AI Concepts](#)
 - [AI Factory Learning Center](#)
-

AI Factory vs. “Generic AI”

AI Factory は、Postgres エステート、主に EDB Hybrid Manager 内で実行される Gen AI とデータインフラストラクチャに焦点を当てています。「汎用 AI」とは、既に使用している汎用 AI / ML スタックとサービスたとえば、外部 LLM API、ML プラットフォーム、データサイエンスツールなどを指します。

このページでは、各アプローチが誰を対象としているか、それぞれが最適な機能、どこで実行されるか、どちらかを選択する場合、およびそれらが連携する方法について説明します。また、EDB Postgres® AI を採用するチームから聞こえる一般的な質問にも答えます。

それぞれの意味

- AI ファクトリー Postgres データの Gen AI とインテリジェントデータアプリのモジュール型機能セット。これには、[Gen AI Builder](#)、[Pipelines and data preparation](#)、[Vector Engine \(PGvector\)](#)、および [Model Serving Reference Manual](#) が含まれます。これは、最初にハイブリッドマネージャー用に設計されており、[AI Factory in Hybrid Manager](#) を介してガバナンスと可観測性のための統合コントロールプレーンを取得します。
- 汎用 AI AI Factory の外部の汎用 AI および ML ツール—実験やモデル開発に使用されるクラウド LLM API またはスタンドアロン ML プラットフォームなど。これらを AI Factory と組み合わせて、エンドツーエンドのソリューションを構築できます。

概念とアーキテクチャの概要については、[Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#) および [AI Factory Explained](#) を参照してください。

対象者

- AI ファクトリー ガバナンスを備えた信頼できる Postgres データに Gen AI アシスタント、ドキュメントインテリジェンス、および AI 搭載機能を構築したいアプリケーション開発者、データエンジニア、およびプラットフォームチーム。ハンズオンについては、[Learning Paths](#) を参照してください。
 - 汎用 AI 専用の ML ツールでモデルの探索、微調整、またはドメインの R&D を行うデータサイエンスと研究チーム。これらの取り組みにより、モデルとコンポーネントを AI Factory にフィードして、運用化できます。
-

実行される場所

- ハイブリッドマネージャーハブ AI Factory の機能のほとんどは、[Hybrid Manager' s AI Factory](#) 内で配信および管理されます。これにより、Gen AI、パイプライン、ベクター検索、モデル提供全体でのライフサイクル制御、ガバナンス、およびオブザーバビリティが提供されます。
 - 外部ハイブリッドマネージャー選択したコンポーネント [Vector Engine \(PGvector\)](#)、Pipelines and data preparation、または AIDB 機能などの Postgres 焦点を当てた機能がサポートされている場合を実行できます。そうすると、クラスターのライフサイクル、セキュリティ、およびオブザーバビリティを所有します。詳細については、[Where AI Factory runs](#) を参照してください。
-

それぞれを選択する場合

次の場合に AI Factory を選択します。

- Postgres のエンタープライズデータに依存するアシスタント、検索、またはワークフローの自動化をビルドします。
- Pipelines and data preparation および [Vector Engine \(PGvector\)](#) を使用したニーズ管理型検索拡張生成 RAG。
- 統合された可観測性とライフサイクル管理が必要ですハイブリッドマネージャー。
- [Model Serving Reference Manual](#) を介して安定したエンドポイントの背後でモデルを提供する計画。

次の場合に汎用 AI を選択します。

- 研究設定でのモデルアーキテクチャ、トレーニングデータ、または評価ワークフローを実験します。
- Postgres ガバナンスとの統合を必要としないプロトタイピングまたは一時的なワークロードには、外部 API を使用します。
- 運用中の Postgres またはハイブリッドマネージャー環境の一部ではないバッチモデルトレーニングを実行します。

多くの組織は、探索のための汎用 AI と、Postgres でデータ中心の AI を運用化するための AI ファクトリーの両方を行っています。

連携する方法

- 独自のモデルの持ち込み 精査されたモデルをパッケージ化し、[Model Serving Reference Manual](#) を使用した推論サービスとして展開します。[Model Library Reference Manual](#) パターンを使用して、モデルアセットを整理します。
 - 外部プロバイダーを使用する [Gen AI Builder](#) で、ドライバーを介して外部サービスを呼び出すことができるアシスタントとツールをビルドします。Drivers overview を参照してください。
 - データフローの運用化 Pipelines and data preparation を使用して、ベクター検索と RAG 用にコンテンツを準備、変換、および最新の状態に保ちます。
 - Postgres のベクターを使用したクエリーリレーショナルデータとともに [Vector Engine \(PGvector\)](#) を使用してエンベディングを保存および検索して、操作を簡単にしてセキュリティを高めます。
-

ユースケース

- エンタープライズナレッジアシスタントナレッジベースと Postgres から信頼できる回答を取得するドメインアシスタントを構築します。[Gen AI Builder](#) および [Vector Engine \(PGvector\)](#) から始めます。
 - ドキュメントインテリジェンス Gen AI アシスタントと統合された Pipelines and data preparation を使用して、ドキュメントを分類、抽出、および要約します。
 - アプリ用の AI 搭載 API [Model Serving Reference Manual](#) の背後にある埋め込み、レコメンデーション、または言語タスクを提供します。
 - モデルの探索と評価優先する「汎用 AI」ツールを使用して、候補モデルを評価します。提供のために AI ファクトリーに昇格する前に。
-

前提条件

- ハイブリッドマネージャーの使用 AI Factory が有効になり、Gen AI、パイプライン、ベクター検索、モデル提供の資格を備えたハイブリッドマネージャーインスタンスへのアクセス。[AI Factory in Hybrid Manager](#) を参照してください。
- ハイブリッドマネージャーの外部で実行されているコンポーネント拡張機能と AIDB 用にサポートされている Postgres 環境、およびパイプラインとサービスに必要なインフラストラクチャ該当する場合。プラットフォームの互換性、セキュリティ、およびライセンスを検証します。

HM 外部での使用の正確なサポートマトリックスが必要な場合は、EDB 担当者と調整してください。

よくある質問

Q: AI Factory はモデルをトレーニングしますか?

A: AI Factory は、Gen AI とデータインフラストラクチャに焦点を当てています。データを使用してモデルを取得、強化、および提供します。モデルのトレーニングと実験に汎用スタックを使用し、[Model Serving Reference Manual](#) を介して精査されたモデルを提供します。

Q: ハイブリッドマネージャーなしで AI Factory を実行できますか?

A: 多くの機能はハイブリッドマネージャーで実行することを目的としています。[Vector Engine \(PGvector\)](#)、[Pipelines and data preparation](#)、[AIDB](#) 機能などの選択されたコンポーネントは、サポートされている場合にハイブリッドマネージャーの外部で実行できます。[Where AI Factory runs](#) を参照してください。

Q: 独自の LLM または埋め込みモデルを持ち込むにはどうすればよいですか?

A: [Model Serving Reference Manual](#) を使用してパッケージ化および展開します。[Model Library Reference Manual](#) の下のライブラリパターンを使用して、アーティファクトとメタデータを管理します。

Q: アシスタントはどのようにしてエンタープライズデータに安全にアクセスしますか?

A ガバナンスされたナレッジベースで *Gen AI Builder* を使用し、RAG では Pipelines and data preparation、および Postgres ネイティブ *Vector Engine (PGvector)* を使用します。ハイブリッドマネージャーでは、これらのワークロードは、統合されたガバナンスと可観測性の下で実行されます。

Q どこから始めればよいですか？

A: *AI Factory Concepts* を確認し、*Learning Paths* を選択します。ハイブリッドマネージャーで操作する場合、AI Factory in Hybrid Manager から開始します。

関連トピック

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory Explained*
- *Gen AI Builder*
- Pipelines and data preparation
- *Vector Engine (PGvector)*
- *Model Serving Reference Manual*
- AI Factory in Hybrid Manager

Fundamental AI Concepts

AI Factory は、最新の AI 概念、アーキテクチャ、およびテクノロジーに基づいて構築され、高度なデータ駆動型のインテリジェントアプリケーションを可能にします。

このページでは、AI Factory のデザインと機能の基礎を形成する重要な **業界コンセプト** について説明します。

- AI の概念
- ベクトルおよびセマンティック検索
- 取得拡張生成 RAG
- データベース用 AI
- インフラ用 AI
- ハイブリッドおよびソブリン AI アーキテクチャ

これらの概念が AI Factory と Hybrid Manager でどのように実装されているかを確認するには、以下から始めます。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory terminology*

始める前に

次の場合、このセクションを最大限に活用できます。

- AI ワークフロー の基本的な理解
- ベクターデータベース および セマンティック検索 に関する知識
- AI パイプライン および LLM アーキテクチャ の認識
- 役立つ AI Factory Concepts と *Getting Started - AI Factory 101* の経験

コア AI コンセプト

学習システムとモデルのトレーニング

学習システムにより、モデルはデータから学習し、明示的なプログラミングをせずに特定のタスクのパフォーマンスを向上させることができます。

一般的な学習タイプ

- 教師あり学習 — ラベル付けされたデータに基づいて予測します例売上予測
- 教師なし学習 — ラベルのないデータからパターンを見つけます例顧客セグメンテーション
- 強化学習 インタラクションと報酬フィードバックを介して学習します例ゲームプレイエージェント

インフラストラクチャのニーズ

- トレーニングデータ用のスケーラブルなストレージ
- モデルトレーニングのためのコンピューティング CPU、GPU、TPU
- 推論用の最適化されたサービスインフラストラクチャ

ディープラーニング DL とニューラルネットワーク

DL は、マルチプルのレイヤーを備えたディープニューラルネットワークを使用して、複雑なパターンを抽出および学習します。

長所

- 非構造化データの処理画像、テキスト、オーディオ、ビデオ
- 画像認識、音声認識、テキスト生成などの高度なアプリケーションを強化

インフラストラクチャのニーズ

- ハイパフォーマンスコンピューティング GPU または TPU
- 最適化されたデータパイプラインとモデル提供インフラストラクチャ

自然言語処理 NLP

NLP は、AI を適用して人間の言語を理解および生成します。

一般的な使用例

- テキスト分類
- 感情分析
- 機械翻訳
- 会話型 AI
- インテリジェントな文書処理
- コンテンツの概要

インフラストラクチャのニーズ

- 大規模モデルアーキテクチャトランスフォーマー
- リアルタイムインタラクションのための低遅延のサービスインフラストラクチャ
- モデルをフィードおよび微調整するためのデータパイプライン

大規模言語モデル LLM

LLM は、トランスフォーマーなどの深層学習アーキテクチャに基づいた高度な NLP モデルです。

主な特徴

- 大規模なテキストコーパスでトレーニング
- 一貫した状況依存のテキストを生成
- チャットボット、コンテンツ生成、コードアシスタント、高度な Q&A をサポート

LLM は、最新の Gen AI アプリケーションの基盤です。

インフラストラクチャのニーズ

- トレーニング用の巨大なコンピューティングとストレージ
- リアルタイム推論のための GPU に最適化されたサービング
- マルチユーザーワークロードのためのスケーラブルなインフラストラクチャ

エンベディングとベクター表現

エンベディングはデータの密なベクトル表現であり、数値形式で意味をキャプチャします。

アプリケーション

- セマンティック検索

- パーソナライズとレコメンデーション
- 異常検出
- RAG 検索拡張生成

エンベディングの使用方法

- テキスト、画像、オーディオ、ビデオ、または構造化データをベクターに変換する
- pgvector を使用してベクターデータベースまたは Postgres に保存
- 近似最近傍 ANN アルゴリズムを使用して類似性検索を実行します

ベクターデータベースとセマンティック検索

エンベディングを保存および検索するように最適化されたデータベース。

主な機能

- 高速ベクトル類似性検索
- 高次元ベクトルストレージ
- 数十億のベクトルにスケーリング
- ハイブリッド検索のサポートベクター+キーワード/フィルター

一般的な使用例

- セマンティック検索エンジン
- パーソナライズされたおすすめ
- グラウンディングを備えたナレッジアシスタント
- 不正検出と異常検出
- 強化された Gen AI パイプライン RAG

取得拡張生成 RAG

RAG はベクトル検索と LLM を組み合わせて、根拠のある応答を生成します。

仕組み

1. ユーザークエリー → エンベディングに変換
2. ベクトル検索は、関連するドキュメント/コンテンツを取得します
3. 取得されたコンテキストは LLM プロンプトに注入されます
4. LLM は、モデル知識と取得したコンテキストの両方を使用して応答を生成します

利点

- より正確で根拠のある AI 応答
- リアルタイムの動的な知識の統合
- ドメイン固有でコンプライアンスを意識した AI アプリケーションのサポート

RAG は、AI Factory がサポートするコアアーキテクチャパターンです。

データベース用の AI

インテリジェントなデータベース管理

AI を使用してデータベース操作を最適化する

- 自動パフォーマンスチューニング
- インテリジェントなクエリ最適化
- プロアクティブなリソース管理
- 予知メンテナンス
- 自己修復機能

目標 自律的で高度に最適化されたデータベースインフラストラクチャ。

インデータベース AI

モデルのトレーニングと推論を データベース内で実行し、データの移動を削減します。

利点

- 迅速なインサイトと応答時間
- 簡素化されたアーキテクチャ
- データベース内アプリケーションのレイテンシーの削減
- リアルタイム予測機能
- 合理化されたモデル操作ワークフロー

例 Postgres で SQL からモデルを呼び出す。

Postgres のベクターデータベース

AI Factory は、Postgres の **pgvector** 拡張機能を活用します。

- ネイティブベクター型と演算
- AI パイプラインおよび埋め込みモデルとの統合
- ハイブリッドマネージャーおよびモデルサービングとの完全な統合

- **Postgres データ** でセマンティック検索と RAG パイプラインを有効にする

AI 機能をコアデータに近づけます。

インフラストラクチャのための AI

AI 高速化ハードウェア GPU、TPU

AI ワークロードは、専用のハードウェアに大きく依存します。

- **GPUs** — トレーニングと推論のための汎用アクセラレーション
- **TPU** — ディープラーニング用に最適化された ASIC
- **その他のアクセラレータ** — FPGA、推論チップ

利点

- モデルトレーニングの大幅な高速化
- プロダクション推論の高スループット
- クラウドネイティブ AI クラスターでの効率的なリソース利用

AI Factory は、**KServe** を介して最適化された GPU サービングを統合します。

AI ワークロードのクラウドコンピューティングの活用

最新の AI ワークロードは、スケーリングのためにクラウドコンピューティングに依存しています。

- オンデマンドの GPU / TPU インスタンス
- トレーニングとサービスのためのマネージド Kubernetes サービス例ハイブリッドマネージャー
- 最適化された AI スタック CUDA、PyTorch、TensorFlow、JAX
- 高度なネットワーキングを使用した分散モデルトレーニング
- ハイブリッドおよびマルチクラウドアーキテクチャ

目標

- 柔軟性とパフォーマンスを最大化
- コストを最小限に抑える
- クラウドプロバイダーと環境全体でのポータビリティを有効にする

AI Factory は、**ハイブリッドクラウド** および **ソブリン AI** パターンをネイティブにサポートしています。

ハイブリッドおよびソブリン AI

現代の企業では、**ソブリン AI** の必要性がますます高まっています。

- データはインフラストラクチャとコントロールに残ります
- AI パイプラインは、信頼できる知識とコンテンツを活用します
- モデルサービングはインフラストラクチャで実行されます例ハイブリッドマネージャー+KServe
- ガバナンス、可観測性、およびコンプライアンスがビルトインされています

AI Factory は次のことを可能にします。

- Postgres の AI
- あなたのデータを使用したパイプラインとナレッジベース
- Kubernetes クラスターでのモデルサービング
- ハイブリッドマネージャーダッシュボードを介した完全な可視性と制御

ソブリン AI は、EDB AI Factory のファーストクラスのデザイン目標です。

関連概念

- *Fundamental AI Concepts*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*

次のステップ

これらのコンセプトが **EDB AI Factory** をどのように強化するかを確認するには、次のことに進みます。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory terminology*
- *Getting Started - AI Factory 101*

AI Factory terminology

AI Factory の用語は、**EDB Postgres® AI** EDB PG AI、**Hybrid Manager AI Factory**、および関連コンポーネントで使用する主要な概念とテクノロジーを定義します。

このページは、これらの概念的な説明を補完します。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Fundamental AI Concepts*

これらの用語を理解することは、EDB PG AI を使用して、信頼でき、ガバナンスがあり、スケーラブルな AI ソリューションを構築するのに役立ちます。

コア AI コンセプト

機械学習 ML

機械学習は、データ露出を通じて改善するアルゴリズムを使用して、明示的なプログラミングなしでタスクを実行します。[Model Serving Reference Manual](#)、パイプライン、およびアシスタントを介して、AI Factory での予測分析、自動化、および意思決定を強化します。

[Learn more](#)

- これは、現在ハイブリッドマネージャーによってサポートされている意図的なユースケースではないことに注意してください*

ディープラーニング DL

ディープラーニングは、多層ニューラルネットワークを活用して、複雑なデータパターンをモデル化します。AI Factory 内の言語理解や画像認識などの高度なアプリケーションをサポートし、GPU アクセラレーションを備えた [Model Serving Reference Manual](#) を介して有効になります。

[Learn more](#)

自然言語処理 NLP

自然言語処理 NLP により、コンピューターは人間の言語を理解および生成できます。[Knowledge Bases Reference Manual](#)、[Retrievers Reference Manual](#)、および [Assistants Reference Manual](#) を介して、AI Factory のセマンティック検索と会話型 AI を基盤とします。

[Learn more](#)

大規模言語モデル LLM

LLM は、大規模なテキストコーパスでトレーニングされた大規模な深層学習モデルです。AI Factory では、[Assistants Reference Manual](#)、取得拡張生成 RAG、およびさまざまなモデルパイプラインを駆動し、[Model Serving Reference Manual](#) を使用して展開されます。

[Learn more](#)

エンベディング

エンベディングは、意味的な意味をキャプチャするデータのベクトル表現です。AI Factory Pipelines は、[Knowledge Bases Reference Manual](#) で使用されるエンベディングを作成し、[Vector Engine \(PGvector\)](#) を介して提供されて、セマンティック検索と RAG を有効にします。

[Learn more](#)

ベクターデータベース

ベクターデータベースはエンベディングを保存し、高速な類似性検索を有効にします。AI Factory は、オープンソースの `pgvector` 拡張機能に構築され、Postgres と直接統合された *Vector Engine (PGvector)* を介してこれを提供します。

取得拡張生成 RAG

RAG は、ベクトル検索と LLM 生成を組み合わせ、関連ドキュメントの地上モデル応答を実現します。AI Factory では、*Knowledge Bases Reference Manual*、*Retrievers Reference Manual*、および *Model Serving Reference Manual* を介して実装されます。

Intro to RAG

データベース用の AI

インテリジェントなデータベース管理

インテリジェントなデータベース管理は AI を適用して、Postgres のパフォーマンスと操作を最適化します。AI Factory は、*Vector Engine (PGvector)* とパイプラインを使用したインテリジェントな取得と検索でこれを拡張します。

Postgres のベクトル検索

ベクター検索を使用すると、Postgres 内で直接エンベディングを照会できます。AI Factory は、`pgvector` を使用して、*Vector Engine (PGvector)* を介してこの機能を強化し、ナレッジベースと RAG をサポートします。

AIDB

AIDB AI-in-Database は、ベクトル検索、埋め込みパイプライン、および将来の機能を HM 管理の Postgres クラスターにもたらしめます。これは、AI Factory Pipelines and data preparation およびナレッジベースの基盤です。===== インテリジェントなデータベース管理は AI を適用して、Postgres のパフォーマンスと操作を最適化します。AI Factory は、*Vector Engine* とパイプラインを使用したインテリジェントな取得と検索でこれを拡張します。

データベース内機械学習 In-DB ML

In-DB ML は、Postgres 内でベクトル検索と ML パイプラインを実行できるようにし、データの移動とレイテンシーを削減します。AI Factory は、*Vector Engine* および *Pipelines* を介してこれを実装します。

Postgres のベクトル検索

ベクター検索を使用すると、Postgres 内で直接エンベディングを照会できます。AI Factory は、`pgvector` を使用して、*Vector Engine (PGvector)* を介してこの機能を強化し、ナレッジベースと RAG をサポートします。

AIDB

AIDB は、ベクター検索、埋め込みパイプライン、およびその他の機能を HM 管理の Postgres クラスターにもたらしめます。これは、AI Factory [Pipelines](#) およびナレッジベースの基盤です。

データベースへの自然言語インターフェイス

自然言語インターフェイスにより、ユーザーは SQL ではなく自然言語を使用して Postgres を照会できます。AI Factory Assistant は、[Gen AI Builder](#) を介してこのパターンを活用し、LLM と構造化および非構造化取得を組み合わせます。

AI インフラ

AI 高速化ハードウェア

AI Factory は、GPU 高速化された Kubernetes クラスターを使用して、ディープラーニングモデルと高スループットの推論を提供します。[Model Serving](#) のモデルワークロードは、GPU 対応ノードで実行されます。

[Learn more](#)

KServe

KServe は、AI Factory がモデルの展開と管理に使用するオープンソースの Kubernetes ネイティブフレームワークです。AI Factory [Model Serving Reference Manual](#) の InferenceServices、自動スケーリング、およびオペラビリティを提供します。

[KServe documentation](#)

モデルサービング

Model Serving は、内部で KServe を使用して、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展開します。LLM、埋め込みモデル、ビジョンモデル、およびカスタム AI ワークロードをサポートしています。

[Model Serving Reference Manual](#)、[Model Serving Explained](#) を参照してください。

Gen AI Builder

Gen AI Builder

Gen AI Builder は、AI 搭載アプリとエージェントを構築するための AI Factory のアプリケーションレイヤーです。それは統合します

- 会話型 AI の場合 [Assistants Reference Manual](#)
- RAG およびセマンティック検索の場合 [Knowledge Bases Reference Manual](#)
- [Retrievers Reference Manual](#) 高度な検索パターンの場合
- [Tools Reference Manual](#) 外部アクションを実行する

- [Rulesets Manual](#) アシスタントの動作を制御する
- [Threads Reference Manual](#) 会話履歴を永続化する
- AI ワークフローとパイプラインを定義する [Structures Reference Manual](#)

Hybrid Manager AI Factory に展開され、フルスタックのソブリン AI が有効になります。

ハイブリッドマネージャー AI Factory

ハイブリッドマネージャー

Hybrid Manager は、AI Factory ワークロード、GPU リソース、パイプライン、モデルサービング、ナレッジベースの管理に Kubernetes ベースのコントロールプレーンを提供します。エンドツーエンドのガバナンス、セキュリティ、およびオブザーバビリティを提供します。

Hybrid Manager AI Factory を参照してください。

パイプライン

パイプラインは、コントロール内のデータソースからの埋め込みとナレッジベースの作成を自動化し、透明性と監査可能性を保証します。

[AI Accelerator Pipelines overview](#) を参照してください。

ナレッジベース

セマンティック検索と RAG のナレッジベースインデックスコンテンツ。マルチソース、マルチモーダル検索をサポートし、AI Factory Assistant をサポートします。

[Knowledge Bases Reference Manual](#) を参照してください。

モデルサービング

Model Serving は、Kubernetes ネイティブの KServe を使用してモデルを展開し、[Model Library Reference Manual](#) と統合します。アシスタント、ナレッジベース、カスタム AI アプリケーションを強化します。

[Model Serving Reference Manual](#) を参照してください。

追加の概念

画像とモデルライブラリー

ハイブリッドマネージャーのイメージとモデルライブラリーは、Postgres と AI モデル展開の両方のコンテンツを管理します。[Model Library Reference Manual](#) は、AI に焦点を当てたビューを提供し、モデル提供と管理されたイメージワークフローをサポートします。

[Model Library Reference Manual](#) を参照してください。

NVIDIA NIM

NVIDIA NIM マイクロサービスは、高性能の推論のために NVIDIA から最適化された AI モデルイメージです。AI Factory は、LLM、エンベディング、およびビジョンモデルに NIM コンテナを使用し、Model Serving を介して提供されます。

How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe を参照してください。

概要

これらの用語を理解すると、次のことをビルドするのに役立ちます。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *AI Factory Explained*
- Hybrid Manager AI Factory

これらの機能を組み合わせることで、**信頼でき、ガバナンスされたソブリン AI** を、自分のデータ、データベース、自分の制御の下で提供できます。

Sovereign AI Explained

ソブリン AI は、EDB Postgres® AI EDB PG AI の中心原則です。データ、モデル、および AI スタックを完全に制御できます。— **監査性**、**セキュリティ**、および **信頼** がビルトインされています。

ソブリン AI は、不透明なクラウド API と外部データフローに依存するのではなく、知的財産、ビジネスデータ、および AI 駆動型アプリケーションがガバナンスの下にあり、あなたが制御するインフラストラクチャで実行されるようにします。

それが重要な理由

AI は、組織がデータと対話し、新しいエクスペリエンスを提供する方法を変革しつつあります。しかし、AI の導入には実際のリスクが伴います。

- 多くのパブリック AI API は **ブラックボックス** です。データの使用方法を制御できなくなります。
- AI の埋め込みとベクターは、サードパーティシステムで **孤児** になることがよくあります。
- 機密データはコンプライアンスの境界を逸脱する可能性があり、**規制リスク** が発生します。
- AI の結果は **非監査可能** である可能性があり、規制対象業界での懸念が高まっています。
- アプリケーション **トレーサビリティ** と **説明可能性** が欠けていることがよくあります。

ソブリン AI は、これらの課題に答えます。

- **自分のデータと埋め込みを所有** — Postgres とオブジェクトストレージに保存されます。

- モデルと AI スタックを所有 —パイプラインとモデルサービングを介して提供されます。
- 自分のナレッジベースを所有 —完全に管理可能、バージョン管理、監査可能。
- 独自の AI ワークロードを所有 —あなたが制御する Kubernetes 環境で実行されます。
- アシスタントの動作を所有する — Agent Studio で定義、管理、および観察可能。

ソブリン AI とは何ですか？

ソブリン AI は、次のようなアーキテクチャとオペレーティングモデルです。

- データは信頼境界内に残ります。
- AI パイプラインは完全に透明で監査可能。
- モデルの提供は、管理するインフラストラクチャで発生します。
- 埋め込みは、制御する場所に保存されます Postgres、オブジェクトストレージ。
- アシスタントの動作は管理され、バージョン化されています。
- ユーザーに公開される API は安全で監視可能です。
- 外部 API が関係するかどうかとどのように選択できます。隠れた依存関係はありません。

ソブリン AI はスペクトルです - 全か無かの選択ではありません。

- 一部の組織は、**完全な自己管理、ゼロ外部 API** アーキテクチャを必要とします。
- その他は、注意深く管理されたハイブリッドモデル**プライベートナレッジベース+パブリック LLM**を使用します。
- EDB PG AI はこのフルスペクトラムをサポートし、外部スタックにロックされることはありません。

EDB PG AI がソブリン AI を有効にする方法

EDB PG AI は、ソブリン AI 用にゼロから設計されています。

- **データファーストアーキテクチャ** —ナレッジはデータベースとオブジェクトストレージから始まります。
- **Vector Engine** —埋め込みは Postgres または制御されたストレージにあります。
- **パイプライン** —データから埋め込み、クエリーまでのパイプラインのすべてのステップを制御します。
- **モデルサービス** —モデルは、ガバナンスの下で Kubernetes インフラストラクチャで実行されます。
- **Agent Studio** —ルールセット、ツール、ストラクチャー、およびアシスタントはすべて管理対象アセットです。
- **可観測性** —モデルとアシスタントアクティビティの完全な監査証跡とモニタリング。

主要コンポーネント

- *Vector Engine (PGvector)* — Postgres クラスターでのベクター検索。
- Pipelines and data preparation — ナレッジベースと埋め込みの作成の制御。
- *Model Serving Reference Manual* — Kubernetes ネイティブ、GPU アクセラレーションモデル推論。
- Gen AI in Hybrid Manager — ガバナンドアシスタントとツール開発。

外部 LLM API を選択的に統合することもできますが、完全な可視性と選択が必要です。

なぜハイブリッドマネージャーを使うのか？

本日、EDB PG AI の **最強のソブリン AI モデル** は、**Hybrid Manager™** EDB で提供されます。

See: Hybrid Manager Gen AI Workloads

Hybrid Manager は、次のアーキテクチャを提供します。

- EDB に最適化された **Kubernetes** にパイプラインとモデルサービングを展開します。
- 完全なライフサイクル制御 でナレッジベースを管理します。
- すべての AI アクティビティをデータベース、パイプライン、およびアシスタントレベルで監視およびトレースします。
- モデル展開、イメージの来歴、およびレトリバー構成を管理します。
- アシスタントとセマンティック検索への **安全な API アクセス** を有効にします。
- 詳細な **データの所在地と信頼境界** を実装します。

フルスタック制御 Postgres データベース → 埋め込み → パイプライン → モデル推論 → アシスタントと API からすべて観察可能および監査可能。

概要

- **ソブリン AI** は、信頼、透明性、および制御に関するものです。
- EDB PG AI は、ロックインなしでソブリン AI を実装するための完全なスタックを提供します。
- AI アーキテクチャを **オープンまたはクローズ** する方法を選択しますが、**データの制御** を失うことはありません。
- Hybrid Manager は、企業に **信頼され管理された AI** を展開するための運用実績のあるプラットフォームを提供します。

詳細はこちら

- [AI Factory Overview](#)
- [Vector Engine \(PGvector\)](#)
- [Pipelines and data preparation](#)
- [Model Serving Reference Manual](#)
- [Gen AI in Hybrid Manager](#)

Security and governance in AI Factory

AI Factory は、明確なガードレールを備えたインフラストラクチャで Gen AI およびデータ駆動型アプリケーションを構築するのに役立ちます。このページでは、誰が何を担当するか、ガバナンスが適用される場所、およびコンポーネント全体に最小特権パターンを適用する方法について説明します。

セキュリティとガバナンスが重要な理由

Gen AI システムは、機密データに触れ、強力なモデルを呼び出し、アクションを実行する場合があります。適切なコントロールは、チームの生産性を維持しながらリスクを削減します。

- 保存および転送中の機密データを保護する
- モデルの展開、取得ルールの変更、またはアシスタントの公開をできる人を制限する
- コンプライアンスのための監査可能性と可観測性を提供する
- アシスタントがツールまたは外部サービスを呼び出すときに最小特権を適用する

[Sovereign AI Explained](#) も参照してください。

執行が行われる場所

- ハイブリッドマネージャー推奨 AI Factory は、ガバナンスと可観測性のためにハイブリッドマネージャーコントロールプレーンに統合されています。日常の操作にスポークを使用します。
- [AI Factory in Hybrid Manager](#)
- [Gen AI in Hybrid Manager](#)
- [Pipelines models reference](#)
- ハイブリッドマネージャーの外部選択したコンポーネント [Pipelines and data preparation](#) または [Vector Engine \(PGvector\)](#) などのコンポーネントをハイブリッドマネージャーの外部で実行すると、クラスターのライフサイクル、アクセス制御、およびオペレービリティを所有します。

責任者は誰ですか

- プラットフォームチームクラスターアクセス、シークレット、およびネットワークポリシーを制御します。 *Model Serving Reference Manual* の GPU をプロビジョニングします。
- データエンジニア ソース、 Knowledge Bases 、および埋め込みの鮮度を管理します。
- Application/Gen AI ビルダー *Rulesets Manual* 、 *Retrievers Reference Manual* 、 *Tools Reference Manual* 、および最小の権限を持つアシスタントを定義します。

AI Factory にガバナンスを適用する方法

- 管理されたストアにデータを維持する
- プラットフォームによって管理される Postgres とオブジェクトストレージを使用します。標準のポリシーでアクセスを制御します。
- *Vector Engine (PGvector)* を使用して、エンベディングをデータの近くに保ち、既存のデータベースコントロールを再利用します。
- 取得と動作のガバナンス
- レトリバーとナレッジベースを使用して取得スコープと制限を表現します。
- ルールセットで許可可能な動作を強制します。外部呼び出しは、ツールを介して明示的に行う必要があります。
- セキュアモデルエンドポイント
- 制御されたアクセスで *Model Serving Reference Manual* を介してプライベートエンドポイントを公開します。
- 適切なサイズの GPU リソース。 *How-To Update GPU Resources for an InferenceService* を参照してください。
- 観察と監査 b_tran_3 ハイブリッドマネージャーでは、統合された可観測性を使用して、モデル提供、Gen AI 実行、およびパイプラインをモニタします。
- ポリシーの必要性に応じて、スレッド、ログ、およびメトリックを監査アーティファクトとして扱います。

前提条件

- ハイブリッドマネージャー AI Factory が有効になり、Gen AI、パイプライン、ベクトル検索、およびモデルサービングの資格を備えたプロジェクト。AI Factory spoke を参照してください。
- 外部ハイブリッドマネージャーエンタープライズシークレットストア、ネットワークポリシー、および監査ログが、実行するコンポーネントに設定されていることを確認します。

ユースケース

- 規制されたナレッジアシスタント管理されたナレッジベース、コンストレインされたレトリバー、およびルールセットを使用します。プライベートエンドポイントを介してモデルを実行します。

- PII のドキュメントインテリジェンスコンテンツを Postgres/オブジェクトストレージに保持し、パイプラインを介して処理し、ツールを非破壊操作に制限します。
- 内部オートメーション明示的なスコープを持つツールの背後にあるゲート書き込みアクション。スレッドとログを定期的に確認します。

関連トピック

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Sovereign AI Explained*
- *Gen AI Builder*
- Pipelines and data preparation
- *Vector Engine (PGvector)*
- *Model Serving Reference Manual*

第 3

Usage

3.1 AI Factory Models

3.2 AI ファクトリーモデル

AI Factory Models は、EDB Postgres AI 環境内の AI モデルの包括的なライフサイクル管理を提供し、ハイブリッドマネージャーインフラストラクチャとの緊密な統合を介してエンタープライズグレードのガバナンスとスケーラブルな推論機能を提供します。

3.2.1 システムの概要

AI Factory Models は、モデルガバナンスと運用展開を橋渡しする統合プラットフォームとして動作し、AI 資産の組織的な制御を保証しながら、EDB PG AI 機能全体のシームレスな統合を可能にします。このシステムは、取得からリタイアまでの完全なモデルのライフサイクルを管理し、一貫した操作手順とセキュリティ制御を提供します。

コアコンポーネント

****モデルライブラリー*** モデルイメージリポジトリ、承認ワークフロー、メタデータ追跡を管理する集中ガバナンスシステム。このライブラリは、コンプライアンス要件の包括的な監査証跡を維持しながら、検証されたモデルのみが運用環境に到達することを保証します。

モデルサービングインフラストラクチャ 承認されたモデルを、ハイブリッドマネージャーによって管理される Kubernetes クラスター内のスケーラブルな推論エンドポイントに変換する KServe ベースの展開エンジン。サービングレイヤーは、自動スケーリング、状態モニタリング、リソースの最適化を含むエンタープライズグレードの操作特性を提供します。

統合フレームワーク モデル推論機能を AI アクセラレーターパイプライン、ナレッジベース、および Gen AI Builder アプリケーションに接続し、すべての EDB PG AI ワークロードで一貫したモデルアクセスパターンを保証する標準化されたインターフェイス。

3.2.2 戦略的メリット

ソブリン AI の実装

組織は、外部の API サービスに依存するのではなく、独自の Kubernetes 環境内にモデルを展開することにより、AI インフラストラクチャの完全な制御を実現します。このアプローチにより、エンタープライズグレードの運用機能を維持しながら、機密データと独自のモデルが組織の境界内にとどまることが保証されます。

オペレーションの一貫性

標準化された展開パターンとガバナンスフレームワークにより、多様なモデルタイプとフレームワークの管理の複雑さが解消されます。チームは、基になるモデルアーキテクチャまたは意図されたユースケースに関係なく、統一された操作手順の恩恵を受けます。

スケーラブルなリソース管理

複数のモデルにわたる効率的な GPU とコンピューティングリソースの割り当てにより、パフォーマンス保証を維持しながらインフラストラクチャコストが最適化されます。システムは、実際のワークロード要求に基づいた動的な割り当て戦略により、リソースの十分な使用率を防ぎます。

3.2.3 アーキテクチャの統合

AI Factory Models は、ハイブリッドマネージャーと深く統合し、既存の Kubernetes インフラストラクチャ内で包括的なモデルライフサイクル管理を提供します。このシステムは、ハイブリッドマネージャーのクラスターオーケストレーション機能を活用しながら、AI 固有のガバナンスと展開機能で拡張しています。

ハイブリッドマネージャーの依存関係

アセットライブラリーの統合モデルイメージ、メタデータ、およびバージョン情報はハイブリッドマネージャーのアセットライブラリーを介して管理され、従来のコンテナイメージ管理を超えて拡張される一元的なストレージ機能とガバナンス機能を提供します。

Kubernetes オーケストレーションモデルサービングワークロードは、ハイブリッドマネージャーのクラスター管理機能を介してネイティブ Kubernetes リソースとして展開され、展開されたすべてのモデルにわたって一貫した操作動作とリソースの割り当てが保証されます。

セキュリティフレームワークアクセス制御、ネットワークポリシー、および監査ログは、ハイブリッドマネージャーのセキュリティインフラストラクチャを活用しながら、モデル固有のガバナンス制御とコンプライアンス機能を追加します。

3.2.4 運用ワークフロー

モデルのライフサイクル管理

取得と検証モデルは、プライベートレジストリまたは手動登録プロセスからの自動同期を介してシステムに入力されます。各モデルは、運用使用が承認される前に、セキュリティスキャン、パフォーマンス検証、およびコンプライアンス検証を受けます。

展開とスケーリング承認されたモデルは、需要パターンに基づいて自動的にスケーリングする Inference-Services として展開されます。システムは、分離とパフォーマンス保証を維持しながら、複数のモデルにわたってリソース割り当てを最適化します。

モニタリングとメンテナンス包括的な可観測性により、モデルのパフォーマンス、リソースの使用率、および運用の状態に対する可視性が提供されます。自動化されたアラートと復旧手順により、詳細な監査証跡を維持しながら、サービスの中断を最小限に抑えます。

ガバナンスフレームワーク

承認ワークフロー構成可能な承認プロセスにより、運用展開前にモデルが組織基準を満たしていることを保証します。このシステムは、複雑なマルチステークホルダーのワークフローをサポートしながら、事前定義された基準を満たすモデルの日常的な承認を自動化します。

ポリシーの適用セキュリティポリシー、リソース制約、およびコンプライアンス要件は、すべてのモデル展開にわたって自動的に適用されます。これにより、手動介入なしで組織標準の一貫した適用が保証されます。

監査とコンプライアンス詳細なログとレポート機能は、法的コンプライアンス要件をサポートしながら、モデルの使用パターンと運用アクティビティの包括的な可視性を提供します。

3.2.5 学習パスウェイ

基礎の理解

実際の実装に進む前に、コアコンセプトから始めてアーキテクチャの理解を確立します。

概念的なフレームワーク

1. *Model Serving Concepts* - 推論展開パターンの理解
2. *Gen AI Architecture (Entities and Flows)* - システムデザインとコンポーネントの関係
3. *Model Library Explained In Depth* - ガバナンスとライフサイクル管理の原則

システムデザイン

1. *Deployment Overview* - 本番展開のパターンと考慮事項
2. *Observability for Model Serving* - モニタリングと操作の可視化

実際の実装

概念的な理解が確立された後のハンズオン実装による進行状況。

初期セットアップ

1. *Quickstart — Use an Inference Endpoint with Python* - 基本的なクライアント統合
2. *How-To Setup GPU resources* - モデルワークロードのインフラストラクチャの準備
3. *How-To Configure a ClusterServingRuntime* - フレームワーク固有の展開設定

モデルの展開

1. *How-To Create an InferenceService for NVIDIA NIM Container* - 基本的なモデル展開手順
2. *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe* - 特殊な展開パターン
3. *Accessing KServe endpoints (internal and external)* - 内部および外部接続

高度な操作

1. *How-To Integrate Private Registry with Model Library* - カスタムモデルソース構成
2. *How-To Define Repository Rules for Model Library* - 自動モデル検出
3. *How-To Verify InferenceServices and GPU usage* - 展開の検証とトラブルシューティング

参考資料

テクニカルリファレンス

- *Model Serving Reference Manual* - 包括的な展開と構成のリファレンス
- *Model Library Reference Manual* - 完全なガバナンスフレームワークドキュメント
- *Gen AI Builder FAQ* - 一般的な問題と実装の考慮事項

統合ガイド

- *AI Accelerator Pipelines* - パイプラインモデルの統合
- *Knowledge Bases Reference Manual* - サードモデルを使用したアプリケーション開発
- *Vector Engine (PGvector)* - 埋め込みモデルの統合

3.2.6 ユースケースアプリケーション

Enterprise RAG Systems

開発環境と実稼働環境全体で一貫したモデルバージョンを必要とするセマンティック検索および取得拡張生成ワークフロー用の埋め込みモデルを展開します。

プライベート LLM 展開

エンタープライズレベルの運用機能を維持しながら、組織の境界内で機密データを処理する内部言語モデルサービスを実装します。

マルチモーダル AI アプリケーション

ビジョン、言語、および埋め込みモデルを、さまざまなモデルタイプにわたって調整された推論機能を必要とする統合アプリケーションに統合します。

ハイブリッドモデルアーキテクチャ

機密操作の独自モデルと一般的な機能用の外部サービスの両方を必要とするアプリケーション内でプライベートモデルサービスと外部モデルサービスを組み合わせます。

3.2.7 実装の考慮事項

インフラストラクチャ要件

モデルサービングには、大量のコンピューティングリソース、特に大規模な言語モデルとビジョンワークロード用の GPU インフラストラクチャが必要です。組織は、予想されるモデルサイズとスループット要件に基づいてキャパシティを計画する必要があります。

ネットワークアーキテクチャ

ハイブリッドマネージャークラスターの接続は、モデルの展開とランタイムのアクセスパターンの両方に影響を与えます。モデルサービングアーキテクチャを設計するときは、ネットワーク遅延、帯域幅要件、およびセキュリティ境界を考慮します。

セキュリティフレームワーク

モデルガバナンスは、従来のコンテナセキュリティを超えて、モデルの来歴、データプライバシー、推論監査証跡などの AI 固有の懸念を含みます。組織の AI ガバナンスポリシーに合わせて包括的なセキュリティ制御を実装します。

運用の複雑さ

異なるフレームワークでマルチプルモデルを管理するには、かなりの運用の専門知識が必要です。運用ワークロードにスケーリングする前に、展開、モニタリング、およびトラブルシューティングの明確な手順を確立します。

AI Factory Models は、制御されたインフラストラクチャ環境内での包括的なモデルライフサイクル管理とスケラブルな推論展開を介して、エンタープライズ AI オペレーションの基盤を提供します。

Getting Started

Model Serving Concepts

Model Serving は、**EDB Postgres® AI** EDB PG AI のコア機能であり、Kubernetes での AI モデルのスケラブル、柔軟、およびパフォーマンスの提供を可能にします。

それは次のことを可能にします。

- Gen AI アプリケーション
- インテリジェントな検索システム
- 高度なデータパイプライン

モデルサービングは、標準化されたモデル推論用のオープンソースの Kubernetes ネイティブエンジンである **KServe** を使用して実装されます。AI Factory は、KServe をハイブリッドマネージャーと統合して、エンタープライズグレードのライフサイクル管理、セキュリティ、およびオブザーバビリティを提供します。

ソブリン AI のキー モデルは、**Kubernetes クラスター** で実行され、制御下で、完全な可観測性とガバナンスを備えています。

始める前に

モデルサービスを理解するための前提条件

- **Kubernetes の基本** ポッド、サービス、展開に関する知識
- Kubernetes CRD としての **InferenceService** の理解
- **モデルライブラリー** とモデルの登録および展開方法の認識
- EDB PG AI の **ソブリン AI 原則** の理解

推奨されるパス

- *Fundamental AI Concepts*
- *Model Library Explained In Depth*

それが重要な理由

モデルサービングにより、**AI モデル** と **Postgres データ** は、**ソブリン AI 原則** の下でシームレスに安全かつスケーラブルに連携できます。

- オープンソースまたは商用モデルを Kubernetes クラスターに展開します。
- アシスタント、ナレッジベース、および **RAG** パイプラインの Serve Gen AI モデル。
- マルチモーダル取得テキスト、画像、ハイブリッド検索をサポートします。
- **GPU アクセラレーション** とサーバーサイドのバッチ処理でパフォーマンスを最適化します。
- モデルの使用よりも完全な **可観測性**、**監査**、および **ガバナンス** を維持します。

実稼働環境でのハイブリッドマネージャーモデルサービングの統合も参照してください。

コアコンセプト

InferenceServiceKServe 経由

Model Serving の中核は、**InferenceService** です。これは、展開されたモデルを表す Kubernetes ネイティブリソースです。

エンドツーエンドのサービングパイプラインを定義します。

- **Predictor** — モデルサーバーを実行し、推論を処理します。
- **Transformer** オプション — 前処理または後処理を適用します。
- **Explainer** オプション — モデルの説明可能性出力を提供します。

プレディクター

予測子は次を定義します。

- **モデルフォーマット** — PyTorch、TensorFlow、ONNX、Triton など
- **モデルの場所** — S3 互換ストレージ、OCI レジストリ、PVC。
- **リソース** — CPU、メモリ、GPU。
- **自動スケーリング** — スケールトゥゼロを含む弾性スケーリングのポリシー。

ServingRuntime / ClusterServingRuntime

提供するための再利用可能なランタイム定義

- **ServingRuntime** — 名前空間スコープ。
- **ClusterServingRuntime** — クラスター全体の再利用可能なランタイム。

利点

- ランタイム設定をモデルタイプとハードウェアに合わせて調整します。
- チームとプロジェクト全体でランタイム構成を標準化します。

その仕組み

ライフサイクルフロー

1. モデル画像を **モデルライブラリー** に登録します。
2. **Model Serving** コンソールまたは CLI を介してモデルを展開 → InferenceService を作成します。
3. AI Factory + KServe は、Kubernetes リソースポッド、サービスをプロビジョニングします。
4. モデルがランタイムコンテナにロードされます。
5. Kubernetes サービスエンドポイントが公開されます。
6. クライアントは HTTP / gRPC 推論要求を送信します。
7. 要求はトランスフォーマーと説明者を介して渡される場合があります。
8. 推論応答が返されます。

主な機能

- マルチフレームワークのサポート PyTorch、TensorFlow、ONNX、XGBoost、Triton など。
- GPU アクセラレーション — ネイティブ NVIDIA GPU サポート。
- オートスケーリング — Knative を介したスケールトゥゼロを含みます。
- オブザーバビリティ Prometheus メトリック、Kubernetes ロギング、および Hybrid Manager ダッシュボード。
- バッチ処理 — スループットを向上させるためのサーバーサイドのバッチ処理。
- Explainability — モデルの説明可能性ツールのサポート。
- セキュリティとソブリン AI — モデルは、制御の下で Kubernetes クラスターで実行されます。

使用パターン

Gen AI Builder

- アシスタント および エージェントを強化する LLM とマルチモーダルモデルを提供します。

ナレッジベース

- 次の場所で使用されるサービス埋め込みおよび取得モデル
- ナレッジベースのインデックス作成
- RAG パイプライン

カスタムアプリケーション

- InferenceService エンドポイントを次の場所に公開します。
- ビジネスアプリケーション
- マイクロサービス
- ETL / ELT パイプライン

ハイブリッド+ソブリン AI の連携

- すべてのモデルは、ハイブリッドマネージャー KServe レイヤーを介してインフラストラクチャで実行されます。
- あなたは以下をコントロールします
- どのモデルが展開されているか
- リソースの割り当て
- デプロイメントトポロジ
- 可観測性と監査

ベストプラクティス

- ガバナンスを保証には、常に **モデルライブラリー** → **モデル提供フロー**を介してモデルを展開します。
- リソース消費量、特に GPU 使用率を監視します。
- 運用環境で使用する前に、**スケールトゥゼロ**ポリシーを注意深くテストします。
- 一貫性を保つために **ServingRuntime** および **ClusterServingRuntime** テンプレートを使用します。
- モデルライブラリーに **量産モデル** を明確にタグ付けして文書化します。
- InferenceService の展開を定期的に監査します。ソブリン AI にとって重要です。

AI Factory で

Model Serving は、EDB PG AI の複数のコンポーネントを強化します。

Component	How it uses Model Serving
Gen AI Builder	Runs LLMs and specialized models for Assistants
Knowledge Bases	Serves embedding and retrieval models for RAG
Custom AI apps	Exposes InferenceService endpoints for business use cases

AI Factory は、以下を介してモデルサービングを管理します。

- 画像管理のための統合 ****モデルライブラリー***。
- **GPU リソース管理** とスケジューリング。
- 一元化された **可観測性** とロギング。
- ライフサイクル制御のためのシームレスなハイブリッドマネージャーの統合。

関連トピック

- *Model Library Explained In Depth*
- *How-To Deploy AI Models from Model Library*
- *How-To Create an InferenceService for NVIDIA NIM Container*
- *How-To Verify InferenceServices and GPU usage*
- [KServe Official Docs](#)
- [KServe GitHub Repository](#)

次のステップ

- モデルライブラリー で使用可能なモデルを探索します。
- **Model Serving** を使用して、最初のモデルを展開します。
- ハイブリッドマネージャーの可観測性ダッシュボードを介して展開されたモデルを監視します。
- 高度なモデルサービス機能を探索します。
- マルチモーダルパイプライン
- トランスフォーマーとインストーラー
- スケールトゥゼロポリシー

Model Serving は、**EDB PG AI** を使用した **ソブリン AI** 戦略の一部として、インテリジェントで管理された AI アプリケーションを安全かつスケーラブルに構築するための強力な基盤を提供します。

Gen AI Factory LLM Architecture (Library and Serving Overview)

EDB AI Factory のモデルライブラリーと LLM サービスコンポーネントの理解

このドキュメントでは、モデルライブラリーとモデルサービングコンポーネントが AI Factory 内でどのように連携して、エンタープライズグレードの大規模言語モデル LLM 推論機能を提供する方法を理解するための概念的フレームワークを提供します。AI Factory を実装する組織は、運用要件とコンプライアンス義務に合わせた LLM 展開戦略、リソース割り当て、およびガバナンスフレームワークに関する情報に基づいた決定を下すために、このアーキテクチャの理解が必要です。

目的と範囲

AI Factory の LLM サービングアーキテクチャにより、組織はガバナンス、セキュリティ、運用制御を維持しながら、大規模な言語モデルを大規模に展開および管理できます。このシステムは、実験的な LLM 開発と実稼働対応の推論サービス間の重要なギャップに対処します。そこでは、従来のアプローチでは、信頼性、セキュリティ、コンプライアンスに関するエンタープライズ要件を満たさないことがよくあります。組織は、運用の複雑さの削減、GPU リソース使用率の向上、および LLM ライフサイクル全体にわたる一貫したポリシーの適用から利益を得ることができます。このアーキテクチャは、キューレーションのためのモデルライブラリーとランタイム操作のためのモデルサービングインフラストラクチャという 2 つの主要コンポーネントを介して、LLM 開発と運用展開間のギャップを埋めます。

アーキテクチャ原則

関心の分離

このアーキテクチャは、LLM ストレージ/キューレーションモデルライブラリーと LLM 実行モデルサービスを区別し、各懸念の独立したスケーリングと管理を可能にします。この分離により、プラットフォームチームは、実行中の推論サービスに影響を与えることなくストレージ戦略を最適化できる一方、AI チームは、基になるインフラストラクチャの複雑さを心配せずに LLM パフォーマンスに集中できます。1 つのコ

コンポーネントの障害が他のコンポーネントにカスケードされず、チームは特定の運用サイクルに基づいて独立してキューレーションポリシーまたはサービング構成の変更を実装できるため、システムの安定性が向上します。

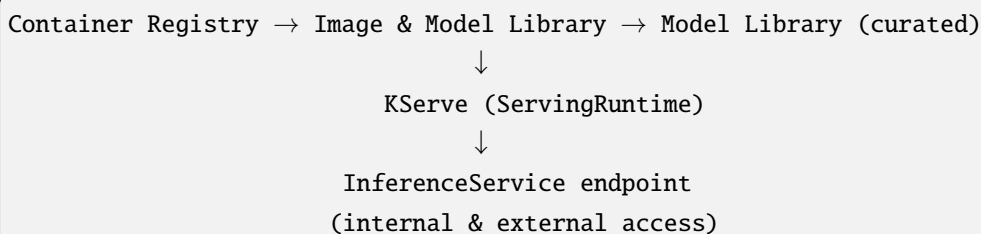
ガバナンス統合

セキュリティとコンプライアンス制御は、後付けとして追加されるのではなく、LLM パイプライン全体に埋め込まれ、画像の取り込みからプロダクション推論まで一貫したポリシーの適用を保証します。このアプローチは、セキュリティ対策が既存のシステムに遡って適用されるときに発生するガバナンスギャップという一般的な問題を解消します。プラットフォーム管理者は、すべての LLM に自動的に適用される一元的なポリシー管理のメリットを得る一方、コンプライアンスチームは、手動介入なしで規制要件を満たす包括的な監査証跡と適用メカニズムを取得します。

リソースの最適化

GPU リソースとコンピューティングインフラストラクチャは、個々の LLM 展開から抽象化され、ワークロードの需要に基づいて効率的なリソース共有と動的な割り当てが可能になります。この抽象化により、高価な GPU リソースが特定の LLM に静的に割り当てられるために十分に活用されない一般的なシナリオが防止されます。組織は、パフォーマンス保証を維持しながらリソース使用率の向上により大幅なコスト削減を達成し、開発チームはインフラストラクチャの詳細な知識を必要としない簡素化された展開プロセスの恩恵を受けます。

制御とデータフロー



フローコンポーネント

1. **LLM イメージの同期** LLM コンテナイメージは、取り込むリポジトリ、タグ、および LLM タイプを定義するリポジトリルールに基づいて、レジストリからイメージ&モデルライブラリーに同期されます。
2. **キューレーションプロセス** モデルライブラリーは、管理された選択プロセスを介して提供するためにこれらの LLM 画像の厳選されたサブセットを公開します
3. **展開構成** オペレーターは、LLM フレームワークに最適化された ServingRuntime を選択し、選択した LLM イメージを使用して InferenceService を展開します
4. **アクセス管理** アプリケーションは、ネットワークとセキュリティの要件に応じて、クラスター通信用の内部 DNS またはアクセスキーを使用して外部ポータルエンドポイントを介して LLM を呼び出します。

コアコンポーネント

画像&モデルライブラリー

外部ソースから LLM コンテナイメージを同期し、使用可能な言語モデルの集中カタログを維持する基本的なレジストリシステム。このコンポーネントは、組織内のすべての LLM イメージの信頼できるソースとして機能し、チームが別個の切断されたモデルリポジトリを維持することから通常発生する混乱を排除します。プラットフォームチームは、組織全体の LLM インベントリへの完全な可視性を取得しますが、AI チームは、展開ごとに複雑な承認プロセスをナビゲートせずに、承認されたモデルへの合理化されたアクセスの恩恵を受けます。

イメージ&モデルライブラリーは、構成された外部レジストリを継続的に監視し、定義された基準に基づいて新しい LLM バージョンを自動的に取り込み、チームが手動介入なしで常に最新の承認された言語モデルにアクセスできるようにします。この自動化により、プラットフォームチームの運用負担が削減され、取り込まれたすべてのコンテンツに組織ポリシーの一貫した適用が保証されます。

キーファンクション

- 構成済みのコンテナレジストリからの自動同期により、手動オーバーヘッドが削減され、LLM バージョンとソース間の一貫性が保証されます
- 選択的 LLM 取り込みのリポジトリルールの適用により、未承認または非準拠のモデルが組織の展開パイプラインに入るのを防ぎます
- メタデータ管理とバージョン追跡は、運用のトラブルシューティングとコンプライアンス報告要件の両方をサポートする完全なリネージ情報を提供します
- セキュリティスキャンと脆弱性評価は、LLM が運用環境に到達する前に潜在的なリスクを自動的に特定します

統合ポイント

- *How-To Define Repository Rules for Model Library*
- *How-To Integrate Private Registry with Model Library*
- *How-To Manage Repository and Image Tag Metadata*

モデルライブラリーキュレーションビュー

運用展開用に承認された LLM のみを提示し、組織ポリシーとコンプライアンス要件を実装するイメージ&モデルライブラリーの管理対象サブセット。この厳選されたビューは、包括的な LLM カタログを、セキュリティ、パフォーマンス、コンプライアンスに関する特定の組織基準を満たす言語モデルの焦点を当てたコレクションに変換します。開発チームは、組織のすべての要件を満たしていることに自信を持って厳選されたライブラリ内のモデルから選択できるため、簡素化された LLM 選択のメリットがあります。

キュレーションプロセスは、承認されたモデルを使用して作業するチームの開発速度を維持しながら、問題のある LLM が運用環境に到達するのを防ぐ品質ゲートとして機能します。プラットフォーム管理者は、複数の基準に対して LLM を自動的に評価する高度な承認ワークフローを実装でき、厳密な品質基準を維持しながら手動レビューの負担を軽減できます。

ガバナンス制御

- LLM プロモーションの承認ワークフローにより、組織の標準を満たすモデルのみが運用環境に到達し、開発の俊敏性を維持しながらリスクを削減します。
- セキュリティポリシーの適用は、セキュリティのベストプラクティスに関する個々のチームの専門知識を必要とせず、すべての LLM に一貫したセキュリティ標準を自動的に適用します
- パフォーマンスベンチマーク要件は、運用 LLM が展開前に運用要件を満たしていることを保証する最小パフォーマンスしきい値を確立します
- コンプライアンス検証チェックは、LLM が業界固有の規制要件を満たしていることを自動的に検証し、コンプライアンスリスクと監査準備のオーバーヘッドを削減します

関連ドキュメント

- [Model Library Reference Manual](#)
- [Model Library Explained In Depth](#)

LLM サービスインフラストラクチャ

承認された LLM イメージを、エンタープライズグレードの操作特性を備えたスケーラブルな推論エンドポイントに変換する KServe ベースの展開システム。このインフラストラクチャは、ヘルスモニタリング、自動スケーリング、GPU リソース管理を含む、大規模な言語モデルを大規模に展開するために必要な複雑なオーケストレーションを処理します。開発チームは、インフラストラクチャの複雑さを抽象化する簡素化された展開プロセスの恩恵を受ける一方、運用チームは、展開されたすべての LLM の包括的な監視および管理機能を取得します。

サービングインフラストラクチャは、さまざまな LLM タイプとフレームワークにわたって一貫した操作特性を提供し、チームが基になるモデルアーキテクチャに関係なく標準の操作手順を適用できます。この一貫性により、運用の複雑性が削減され、標準化されたモニタリング、ロギング、および復旧手順を通じて信頼性が向上します。

コアリソース

- **ServingRuntime** LLM 固有の実行環境 vLLM、NIM、TensorRT-LLM は、さまざまな言語モデルタイプに最適化されたランタイム特性を提供し、展開チームからフレームワーク固有の構成の複雑さを抽象化します
- **InferenceService** 自動スケーリング機能を備えた運用 LLM エンドポイントは、需要パターンに基づいてリソース割り当てを自動的に調整し、コストを最小限に抑えながら最適なパフォーマンスを保証します
- **リソース管理** GPU 割り当てとコンピューティング最適化システムにより、マルチプルの LLM とワークロードにわたって高価なハードウェアリソースの効率的な利用が保証されます

実装ガイド

- [Model Serving Concepts](#)
- [How-To Configure a ClusterServingRuntime](#)
- [How-To Create an InferenceService for NVIDIA NIM Container](#)

アクセス管理

セキュリティ境界を維持しながら、内部クラスター通信と外部 API アクセスの両方を提供するマルチレイヤーアクセス制御システム。このシステムは、モデルが異なるセキュリティプロファイルで内部マイクロサービスと外部アプリケーションの両方にサービスを提供する必要がある実稼働 LLM システムの複雑なセキュリティ要件に対応します。プラットフォーム管理者は、すべてのアクセスパターンにセキュリティポリシーを一貫して適用する一元的なアクセス制御のメリットを享受できますが、開発チームは、多様な統合要件をサポートする柔軟なアクセスオプションを取得します。

アクセス管理システムは、誰がどのような状況でどの LLM にアクセスできるかを詳細に制御し、運用の簡素化を維持しながら複雑な組織の要件をサポートします。このアプローチは、さまざまなアクセス方法にわたってアクセス制御が一貫して適用されない場合に発生する一般的なセキュリティギャップを排除します。

アクセスパターン

- **内部アクセス** サービス間通信用のクラスターローカル DNS は、ネットワークレベルのセキュリティの分離を維持しながら、同じインフラストラクチャ内で実行されているアプリケーションに高性能で低遅延のアクセスを提供します
- **外部アクセス** 認証と承認を備えたポータルベースのエンドポイントにより、包括的な監査証跡とアクセス制御を維持しながら、外部アプリケーションおよびサードパーティシステムとの安全な統合が可能になります

構成の詳細

- *Accessing KServe endpoints (internal and external)*
-

システムフロー

LLM インジェストパイプライン

LLM コンテナイメージは、リポジトリルールによって制御される自動同期プロセスを介して、外部レジストリからイメージ&モデルライブラリーにフローします。これらのルールは、どのリポジトリ、名前空間、およびタグを監視および取り込むかを決定し、組織が日常的な同期タスクを自動化しながら、LLM パイプラインを正確に制御できるようにします。プラットフォームチームは、LLM 更新を管理する際の手動オーバーヘッドの削減の恩恵を受けながら、環境に入るモデルの厳密な制御を維持します。自動化されたパイプラインは、事前に定義された基準を満たすモデルの手動承認のボトルネックを排除するため、AI チームは承認された言語モデルにすばやくアクセスできます。

インジェストパイプラインは、包括的なロギングとモニタリングを提供し、プラットフォームチームが LLM 来歴を追跡し、運用システムに影響を与える前に潜在的な問題を特定できます。この可視性は、運用のトラブルシューティングとコンプライアンス報告要件の両方をサポートします。

キュレーションプロセス

モデルライブラリーは、組織ガバナンスポリシーを適用して完全な LLM カタログをフィルタリングし、展開が承認されたモデルのみを表示します。このキュレーションプロセスには、セキュリティスキャン、パフォーマンス検証、およびコンプライアンス検証が含まれ、モデルバージョンごとに手動でレビューを必要とすることなく、組織の標準に対して LLM を自動的に評価します。このプロセスは、問題のあるモデ

ルが実稼働環境に到達するのを防ぐ厳密な品質ゲートを維持しながら、準拠 LLM の実稼働までの時間を短縮します。

組織は、ソースまたは開発チームに関係なく、すべての LLM にわたってガバナンスポリシーの一貫した適用から恩恵を受けます。この一貫性により、コンプライアンスリスクが削減され、チームは組織標準を満たすモデルを迅速に移行できます。

導入ワークフロー

承認された LLM は、モデルサービングインフラストラクチャを介して展開されます。ここで、**ServingRuntime** 構成は実行環境を定義し、**InferenceService** 仕様はスケーリング動作とリソース要件を決定します。このワークフローは、運用展開の複雑さを抽象化しながら、操作特性の包括的な制御を提供します。開発チームは、インフラストラクチャの深い専門知識を必要としない簡素化された展開プロセスの恩恵を受ける一方、プラットフォームチームはリソースの割り当てと運用ポリシーの制御を維持します。

展開ワークフローは、自動検証とロールバック機能を提供し、開発速度を維持しながら展開リスクを削減します。システムは、運用トラフィックを新しい展開に転送する前に運用の準備状況を自動的に検証するため、Teams は安心して LLM を展開できます。

ランタイム操作

実稼働 LLM 推論エンドポイントは、自動スケーリング、状態チェック、パフォーマンスの最適化を備えた継続的な監視の下で動作します。このシステムは、操作メトリックを維持し、モデルのパフォーマンスとリソース使用率の可観測性を提供するため、実稼働システムのプロアクティブな管理を可能にします。運用チームは、システムの状態とパフォーマンスの包括的な可視化からメリットを得る一方、開発チームは、運用条件の下での LLM 動作に関する洞察を取得します。

ランタイム操作フレームワークは、サービスの中断を最小限に抑えながら、トラブルシューティングとコンプライアンスのための詳細な監査証跡を維持する自動復旧機能を提供します。この自動化により、運用のベストプラクティスの一貫した適用を通じてシステムの信頼性が向上しながら、運用の負担が削減されます。

リソース管理

LLM 用の GPU インフラストラクチャ

このアーキテクチャは、個々の LLM 展開から GPU リソースを抽象化し、需要パターンに基づいて効率的なリソース共有と動的な割り当てを可能にします。この抽象化は、実際の需要に関係なく、リソースが特定の LLM に静的に割り当てられるときに発生する GPU の使用率の低下という一般的な課題に対処します。組織は、運用ワークロードのパフォーマンス保証を維持しながら、リソース使用率の向上により大幅なコスト削減を達成します。

プラットフォームチームは、展開されているすべての LLM 全体の使用率を最適化する集中リソース管理の恩恵を得る一方、開発チームは、複雑な割り当て戦略を理解する必要がなく、GPU リソースにアクセスできます。システムは、ワークロードの特性と組織の優先順位に基づいて、リソースのスケジュールと割り当てを自動的に処理します。

セットアップ要件

- [How-To Setup GPU resources](#)

- *How-To Update GPU Resources for an InferenceService*

LLM スケーリング戦略

自動スケーリング機能は、トラフィックパターンとパフォーマンス要件に基づいてコンピューティングリソースを調整し、サービスレベルを維持しながら費用対効果の高い操作を保証します。スケーリングシステムは履歴パターンから学習し、需要に合わせてリソースをプロアクティブに調整し、エンドユーザーのコストとレイテンシーの両方を削減します。組織は、パフォーマンスを損なうことなく最適化されたリソースコストのメリットを享受できますが、開発チームは運用スケーリングの懸念を心配することなく LLM 開発に集中できます。

スケーリング戦略は、予測可能なワークロードパターンと突然のトラフィックスパイクの両方をサポートし、多様な使用シナリオで一貫したパフォーマンスを保証します。この柔軟性により、組織は需要の変動に関係なく LLM に効率的にサービスを提供できます。

LLM のパフォーマンスの最適化

このシステムには、バッチ処理、量子化、テンソル並列処理を含む LLM 固有の最適化が組み込まれ、スループットを最大化し、レイテンシーを最小限に抑えます。これらの最適化は、モデルの特性とワークロードパターンに基づいて自動的に適用され、開発チームの専門知識を必要とせずに最適なパフォーマンスを保証します。組織は、リソースの効率的な使用を介して LLM パフォーマンスの向上とインフラストラクチャコストの削減の恩恵を受けます。

パフォーマンスの最適化は、実際のワークロード特性に基づいて継続的に評価および調整され、要件とモデルの特性の変化に応じて最適化戦略が進化します。この適応的なアプローチは、時間の経過とともにワークロードと LLM が進化しても、最適なパフォーマンスを維持します。

セキュリティとガバナンス

アクセス制御フレームワーク

レジストリアクセスからプロダクション推論エンドポイントまで、複数のレベルで認証と承認を強制する多層セキュリティモデル。このフレームワークは、さまざまなユーザーとアプリケーションがさまざまなモデルと機能へのさまざまなレベルのアクセスを必要とする運用 LLM システムの複雑なセキュリティ要件に対応します。プラットフォーム管理者は、多様なアクセス要件をサポートする柔軟性を維持しながら、すべてのシステムコンポーネントにセキュリティ制御を一貫して適用する一元的なポリシー管理のメリットを享受できます。

アクセス制御フレームワークは、コンプライアンスレポートとセキュリティモニタリングをサポートする包括的な監査証跡を提供し、組織が LLM インフラストラクチャ全体にわたってセキュリティの可視性を維持できるようにします。この可視性は、運用セキュリティモニタリングとコンプライアンス報告要件の両方をサポートします。

コンプライアンス統合

ビルトインのコンプライアンス制御により、LLM 展開がライフサイクル全体を通じて組織および規制の要件を満たします。これらの制御は、インジェストから運用展開まで、モデルパイプラインのすべての段階に統合されており、開発速度に影響を与えることなく一貫したコンプライアンスを保証します。組織は、

包括的なコンプライアンスカバレッジを維持しながら、手動監査オーバーヘッドを削減する自動コンプライアンス検証の恩恵を受けます。

コンプライアンス統合は、規制報告要件をサポートする詳細な文書と監査証跡を提供し、組織が監査人や規制当局にコンプライアンスを実証できるようにします。この文書は、コンプライアンスの負担を軽減しながら、制御の有効性の包括的な証拠を提供します。

監査とモニタリング

包括的なロギングおよびモニタリング機能は、システム操作、LLM パフォーマンス、およびセキュリティイベントの可視性を提供します。この監視フレームワークにより、実稼働システムに影響を与える前に問題を積極的に特定でき、同時に、トラブルシューティングとコンプライアンス調査のための詳細なフォレンジック機能を提供します。運用チームは、日常の業務とインシデント対応の両方をサポートする包括的なシステムの可視性の恩恵を受ける一方、コンプライアンスチームは、規制要件をサポートする詳細な監査証跡を取得します。

監視システムは、差し迫った問題に関するリアルタイムのアラートと、キャパシティプランニングとパフォーマンスの最適化をサポートする履歴分析機能の両方を提供します。この二重のアプローチにより、運用の応答性と戦略的計画機能の両方が保証されます。

モニタリング設定

- *Observability for Model Serving*
 - *How-To Monitor deployed models with KServe*
-

統合パターン

開発ワークフロー

このシステムは、SDK サポートと API 互換性を介して既存の開発プロセスと統合し、アプリケーション開発ライフサイクルとのシームレスな統合を可能にします。この統合により、開発チームの摩擦が削減され、一元的な LLM 管理とガバナンスの利点が維持されます。開発チームは、使い慣れたツールとパターンを使用して LLM 機能をアプリケーションに組み込むことができ、プラットフォームが提供するエンタープライズグレードの運用機能のメリットを活用できます。

開発ワークフローとの統合は、アプリケーションコードまたは展開プロセスに大幅な変更を必要とせずにプロトタイプから実稼働までスケールする一貫したインターフェイスを介して、実験開発と実稼働展開の両方をサポートします。

はじめに

- *Quickstart — Use an Inference Endpoint with Python*
- *How-To Deploy AI Models from Model Library*

API の互換性

標準の API インターフェイスは、既存の LLM ツールチェーンとの互換性を保証しながら、運用ワークロードにエンタープライズグレードの機能を提供します。この互換性、組織はツールとプロセスへの既存の投資を活用しながら、一元的な LLM 管理とガバナンスの利点を得ることができます。開発チームは、使い

慣れた API とツールを使用しながら、プラットフォームが提供する信頼性、セキュリティ、および運用機能の向上の恩恵を得ることができます。

API の互換性は LLM トレーニングワークフローとアプリケーション統合パターンの両方に拡張され、チームは既存のプロセスやツールを大幅に変更せずにプラットフォームを採用できます。

高度な機能

マルチ LLM フレームワークのサポート

このアーキテクチャは、さまざまな LLM フレームワークとランタイム環境に対応し、組織が統一インフラストラクチャ内で多様な言語モデルタイプを展開できるようにします。この柔軟性により、ベンダーロックインを防止しながら、チームは運用プロセスを断片化することなく特定のユースケースに最適なフレームワークを選択できます。組織は、成熟に応じて新しいフレームワークとテクノロジーを採用する柔軟性を維持しながら、さまざまな LLM タイプ間で一貫した運用手順の恩恵を受けます。

マルチフレームワークのサポートには、基になるフレームワークテクノロジーに関係なく最適なパフォーマンス特性を保証する、さまざまな LLM タイプの最適化されたランタイム構成が含まれます。この最適化により、開発チームに必要な専門知識が削減され、同時に最適なリソースの使用率とパフォーマンスが保証されます。

特殊な展開

- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe*

カスタムランタイム構成

組織は、カスタムの `ServingRuntime` 構成を定義して、特殊な LLM 要件または最適化戦略をサポートできます。このカスタマイズ機能により、標準化された運用プロセスの利点を維持しながら、固有の組織要件のサポートが可能になります。プラットフォームチームは、すべての展開にわたって一貫した操作動作を保証しながら、特定の LLM 特性またはビジネス要件に最適化する特殊なランタイム構成を作成できます。

カスタムランタイム構成は、標準の最適化パターンと、デフォルト構成では対応できない組織固有の要件の両方をサポートします。この柔軟性により、プラットフォームは運用の一貫性を維持しながら、多様な組織のニーズに適応できます。

次のステップ

このアーキテクチャの概要を理解することは、組織内に LLM サービス機能を実装するための基礎を提供します。参照される実装ガイドは、これらのアーキテクチャ概念を作業用システムに変換する特定の構成タスクと操作手順の詳細な手順を提供します。組織は、基本的なコンポーネントから始めて、特定の運用の成熟度とビジネス要件に基づいて、高度な機能を進めていく必要があります。

システムを実際に体験するには、クイックスタートガイドから始めて、特定の展開要件に基づいて詳細な構成ドキュメントを読んでいきます。この進歩により、実装の取り組みがアーキテクチャの原則と一致することが保証され、システム機能と操作手順に関する実践的な専門知識を構築できます。

Deployment Overview

AI Factory を使用すると、AI モデルをスケーラブルな運用対応の推論サービスとして簡単に展開できます。このプラットフォームは、モデルサービングエンジンとして **KServe** を使用し、Hybrid ManagerHM Kubernetes インフラストラクチャ内で動作します。

このページでは、モデル展開の一般的なフローを説明し、実践的な手順に関する主要なハウツーガイドへのリンクを提供します。

導入フローの概要

- HM プロジェクトで **KServe InferenceServices** を作成することにより、モデルを展開します。
 - AI Factory は、これらのサービスを実行するための **GPU 対応の Kubernetes インフラストラクチャ** を提供します。
 - サポートされている **NVIDIA NIM マイクロサービス** または他の互換性のあるモデルを展開できます。
 - **モデルライブラリー** は、モデルイメージの検出と管理に役立ちます。
 - アプリケーションは、**HTTP** または **gRPC API** を介してモデルエンドポイントにアクセスします。
-

展開コンポーネント

KServe InferenceService

各モデルは、Kubernetes ネイティブの **InferenceService** オブジェクトを介して展開されます。

- モデルサーバーポッドのライフサイクルを管理します。
- スケーリング、ヘルスチェック、およびルーティングを処理します。
- モデル使用のためのネットワークエンドポイントを公開します。

ClusterServingRuntime

上級ユーザーは、**ClusterServingRuntime** リソースを構成して、モデルのランタイム環境をカスタマイズすることもできます。

どこから始めるべきか

モデルを展開する準備ができている場合は、次のガイドに従ってください。

- *[How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com/container with KServe](#)*
 - *[How-To Create an InferenceService for NVIDIA NIM Container](#)*
 - *[How-To Configure a ClusterServingRuntime](#)*
 - *[How-To Monitor deployed models with KServe](#)*
-

- [How-To Update GPU Resources for an InferenceService](#)
-

ハイブリッドマネージャーの統合

Model Serving は、**Hybrid Manager**HM Kubernetes クラスターで実行されます。ハイブリッドマネージャーと GPU セットアップの詳細については、次のとおりです。

- [How-To Setup GPU resources](#)
 - [How-To Verify InferenceServices and GPU usage](#)
-

ベストプラクティス

- モデルライブラリーを使用して、サポートされているモデルを選択します。
 - クラスターに十分な GPU リソースがあることを確認します。
 - 導入されたモデルを監視して、パフォーマンスと可用性を保証します。
 - 高度なカスタマイズが必要な場合は、**ClusterServingRuntime** を使用します。
-

次のステップ

- [How-To Deploy AI Models from Model Library](#) を探索する
 - [Supported Models Index](#) でサポートされているモデルを確認します
 - [Observability for Model Serving](#) について学ぶ
-

この展開フローに従うことにより、完全な可観測性とスケールで AI モデルを実行できます。より広範な AI Factory および Hybrid Manager エコシステムと直接統合されています。

AI Factory Model Use Cases and Personas

このガイドでは、さまざまな組織のペルソナとユースケースにわたる AI ファクトリーモデルの実践的な実装シナリオの概要を説明します。各シナリオは、ロールベースの要件に合わせた特定の実装パスを提供します。

ペルソナベースの実装シナリオ

プラットフォームエンジニア

プラットフォームエンジニアは、インフラストラクチャのプロビジョニング、システムの信頼性、および AI モデル展開全体のオペレーショナルエクセレンスに焦点を当てています。

インフラストラクチャ管理シナリオ

目的 マルチプルの開発チームと運用ワークロードをサポートする、信頼性の高いスケーラブルなモデル提供インフラストラクチャを確立します。

重要な要件

- さまざまなチームやプロジェクトのマルチテナントリソースの分離
- 推論需要パターンに基づいた自動スケーリング
- プロアクティブな問題解決のための包括的なモニタリングとアラート
- 組織ポリシーに合わせたセキュリティ制御

実装アプローチ

1. インフラストラクチャ基盤

- 適切なティントとラベルを使用して GPU 対応の Kubernetes ノードを構成する
- 安全なマルチテナントの分離のためのネットワークポリシーを実装します
- 組織のキャパシティプランニングに合わせてリソースのクォータと制限を確立する

2. 運用フレームワーク

- 推論サービスの可観測性のための監視スタック統合を展開する
- モデルのメタデータと構成の自動バックアップ手順を構成する
- クリティカルモデルを提供するワークロードのディザスターリカバリー手順を実装する

3. ガバナンス統合

- 実稼働モデル展開の承認ワークフローを確立する
- モデルイメージ検証用のセキュリティスキャンパイプラインを構成する
- コンプライアンスとオペレーションフォレンジックのための監査ログを実装する

リファレンスドキュメント

- [*How-To Setup GPU resources*](#)
 - [*Observability for Model Serving*](#)
 - [*How-To Verify InferenceServices and GPU usage*](#)
-

AI エンジニア

AI エンジニアは、インフラストラクチャの複雑性を抽象化しながら、モデルのパフォーマンス特性の包括的な制御を提供する、合理化されたモデル展開ワークフローを必要としています。

モデル展開最適化シナリオ

目的最小限の操作オーバーヘッドで、プロダクション推論のための大規模な言語モデルを展開および最適化します。

重要な要件

- 多様なモデルタイプをサポートするフレームワークに依存しない展開手順
- 量子化やバッチ処理を含むパフォーマンス最適化機能
- モデルバリエーションを比較するための A/B テストフレームワーク
- 既存の AI パイプラインおよびワークフローとの統合

実装戦略

1. モデルの準備

- 包括的なメタデータを使用してモデル画像をアセットライブラリーに登録する
- 特定のモデルフレームワークの `ServingRuntime` 最適化を構成する
- 体系的なベンチマークを通じてパフォーマンスのベースラインを確立する

2. 展開構成

- モデル要件に合わせたリソース割り当てで `InferenceServices` を作成する
- 推論需要パターンに基づいて自動スケーリングポリシーを実装する
- 安全な運用更新のためのカナリア展開戦略を構成する

3. パフォーマンスモニタリング

- レイテンシー、スループット、および精度追跡のためのモデル固有のメトリックを確立する
- 自動パフォーマンス回帰検出を実装する
- モデルのパフォーマンスの低下のアラートを構成する

リファレンスドキュメント

- [*How-To Configure a ClusterServingRuntime*](#)
 - [*How-To Create an InferenceService for NVIDIA NIM Container*](#)
 - [*How-To Monitor deployed models with KServe*](#)
-

アプリケーション開発者

アプリケーション開発者は、開発速度と操作の簡素化を維持しながら、モデル推論機能をビジネスアプリケーションに統合します。

RAG アプリケーション開発シナリオ

目的組織インフラ内でプライベート埋め込みモデルと言語モデルを活用する検索拡張生成アプリケーションを構築します。

重要な要件

- 既存のアプリケーションフレームワークと互換性のある標準化された API インターフェイス
- 開発環境と実稼働環境全体で一貫したモデルバージョン
- インタラクティブなユーザーエクスペリエンスのための低遅延推論
- ベクトルデータベースおよびナレッジ管理システムとの統合

開発ワークフロー

1. モデルの統合

- モデルライブラリーを介して承認された埋め込みモデルと言語モデルを特定します
- 標準化された OpenAI 互換エンドポイントを使用してアプリケーションクライアントを構成する
- モデルの可用性のためのエラー処理とフォールバック戦略を実装する

2. パフォーマンスの最適化

- 頻繁に要求される埋め込みのキャッシュ戦略を実装する
- 大量のドキュメント埋め込み操作のバッチ処理を構成する
- 要求パターンを最適化して推論レイテンシーを最小限に抑える

3. 運用展開

- 環境固有のモデルエンドポイント構成を確立する
- アプリケーション固有のモデルの使用パターンのモニタリングを実装する
- モデルインスタンス間の自動フェイルオーバーの構成

リファレンスドキュメント

- [*Quickstart — Use an Inference Endpoint with Python*](#)
- [*Accessing KServe endpoints \(internal and external\)*](#)
- Gen AI Builder Integration

DevOps エンジニア

DevOps エンジニアは、モデル提供ワークロードの CI / CD 統合、展開の自動化、および運用の信頼性に焦点を当てています。

自動モデル展開パイプラインシナリオ

目的モデルレジストリの変更に基づいてモデルサービスインフラストラクチャを自動的に更新する継続的展開パイプラインを実装します。

重要な要件

- 自動化されたモデルの検証と展開のワークフロー
- 既存の CI / CD インフラストラクチャおよび承認プロセスとの統合
- 失敗した展開またはパフォーマンスの低下のロールバック機能
- 展開アクティビティの包括的なロギングと監査証跡

パイプライン実装

1. 自動フレームワーク

- モデルレジストリ更新の Webhook トリガーを構成する
- モデルの互換性テストのための自動検証パイプラインを実装する
- パフォーマンスとセキュリティ基準に基づいて展開ゲートを確立する

2. 展開オーケストレーション

- InferenceService 展開用のインフラストラクチャ・アズ・コードテンプレートを作成する
- ゼロダウンタイム更新のためのブルーグリーン展開戦略を実装します
- ヘルスチェックの失敗に基づいて自動ロールバック手順を構成する

3. モニタリングの統合

- 展開の成功/失敗メトリックとアラートを確立する
- 展開前/事後分析のためのパフォーマンス比較ダッシュボードを実装する
- 展開アクティビティと承認ワークフローの監査ログを構成する

リファレンスドキュメント

- [*How-To Deploy AI Models from Model Library*](#)
 - [*How-To Integrate Private Registry with Model Library*](#)
 - [*How-To Define Repository Rules for Model Library*](#)
-

セキュリティエンジニア

セキュリティエンジニアは、組織および規制要件へのコンプライアンスを保証しながら、モデルサービングインフラストラクチャの包括的なセキュリティ制御を実装します。

セキュアモデルサービングアーキテクチャシナリオ

目的モデル提供のライフサイクル全体にわたってモデル資産と推論データを保護するセキュリティ制御を確立します。

重要な要件

- モデル推論トラフィックのゼロトラストネットワークアーキテクチャ
- モデルのアクセスと使用パターンの包括的な監査ログ
- モデルイメージとサービスインフラストラクチャの脆弱性管理
- 推論要求と応答のデータプライバシー制御

セキュリティの実装

1. アクセス制御フレームワーク

- モデルの登録と展開操作にロールベースのアクセス制御を実装する
- 安全なマルチテナントモデル分離のためのネットワークポリシーを構成する
- 推論エンドポイントの API 認証と承認を確立する

2. 脆弱性管理

- 登録時にモデルイメージの自動セキュリティスキャンを構成します
- 特定されたセキュリティ問題の脆弱性修復ワークフローを実装する
- 実稼働モデル展開のセキュリティ承認ゲートを確立する

3. 監査とコンプライアンス

- すべてのモデル操作の包括的な監査ログを構成する
- 規制要件に合わせたデータ保持ポリシーを実装する
- セキュリティ評価のコンプライアンス報告機能を確立する

リファレンスドキュメント

- *Model Library Reference Manual*
- *Accessing KServe endpoints (internal and external)*
- *Observability for Model Serving*

クロスファンクショナルのユースケース

エンタープライズソブリン AI の実装

シナリオ複数のビジネスユニットにわたって業務効率を維持しながら、包括的な AI ガバナンスを実装する大企業。

ステークホルダープラットフォームエンジニア、セキュリティエンジニア、コンプライアンスチーム

実装要件

- 多様なビジネス要件をサポートするインフラストラクチャを提供するマルチテナントモデル
- 規制コンプライアンスを保証する包括的なガバナンスフレームワーク
- 既存のエンタープライズ ID およびアクセス管理システムとの統合
- さまざまなモデルタイプと使用パターンにわたるパフォーマンスの最適化

ソリューションアーキテクチャ

1. インフラストラクチャレイヤー

- 地理的分散のために複数の Kubernetes クラスターにハイブリッドマネージャーを展開する
- 同期されたガバナンスポリシーを使用してフェデレーションモデルレジストリを実装する
- モデル共有とロードバランシングのためにクロスクラスターネットワークを構成する

2. ガバナンス統合

- ビジネスユニットの委任と全社的な承認ワークフローを確立する
- 規制要件の自動コンプライアンスレポートを実装する
- エンタープライズリスク管理フレームワークに合わせてセキュリティポリシーを構成する

3. オペレーショナルエクセレンス

- すべてのモデルサービスインフラストラクチャにわたって包括的なモニタリングを展開する
- 履歴使用パターンに基づいて、自動化されたキャパシティプランニングを実装する
- モデルサービスの中断に対するインシデント対応手順を確立する

ハイブリッドクラウドモデルの展開

シナリオ一貫した運用手順を維持しながら、オンプレミスとクラウドインフラストラクチャ全体にモデルを展開する組織。

ステークホルダープラットフォームエンジニア、DevOps エンジニア、ネットワークエンジニア

実装に関する考慮事項

- 展開環境間のネットワーク接続と遅延の最適化
- さまざまなインフラプロバイダーにわたって一貫したセキュリティポリシー
- モデルの推論とトレーニングワークロードのデータの局所性要件
- 複数のインフラストラクチャ層にまたがるディザスターリカバリー機能

アーキテクチャコンポーネント

- クラウドとオンプレミスのインフラストラクチャにわたる Federated Hybrid Manager の展開
- 環境固有の展開構成を使用した統合モデルレジストリ
- クロス環境のモニタリングとアラートの統合
- 一貫したバックアップとディザスターリカバリーの手順

実装の考慮事項

リソースプランニング

モデルサービングインフラストラクチャは、特に大規模な言語モデルと高スループットの推論ワークロードの場合、大量の計算リソースを必要とします。組織は、予想されるモデルサイズと使用パターンに基づいて、キャパシティプランニング手順を確立する必要があります。

リソース割り当てのガイドライン

- GPU メモリ要件はモデルパラメーター数に直接スケーリング
- CPU 割り当ては前処理および後処理のパフォーマンスに影響を与える
- ネットワーク帯域幅は、高スループットの推論シナリオに影響を与える
- ストレージパフォーマンスはモデルのロードとチェックポイント操作に影響を与える

運用の複雑さ

さまざまなフレームワークで多様なモデルタイプを管理するには、かなりの運用の専門知識が必要です。運用ワークロードにスケーリングする前に、包括的なトレーニングプログラムと操作手順を確立します。

運用のベストプラクティス

- さまざまなモデルタイプにわたって標準化された展開手順を実装する
- 運用上の問題に対する明確なエスカレーション手順を確立する
- トラブルシューティングとメンテナンスのための包括的なドキュメントを維持する
- 改善の機会を特定するための定期的な業務レビュー

パフォーマンスの最適化

モデルの推論パフォーマンスは、ハードウェア構成、モデルの特性、アプリケーションの使用パターンを含む複数の要素に依存します。体系的なパフォーマンステストと最適化手順を実装します。

最適化戦略

- 導入されたすべてのモデルのパフォーマンスベースラインを確立する
- 自動パフォーマンス回帰検出を実装する
- 実際の使用パターンに基づいてリソース割り当てを構成する
- 定期的なパフォーマンスレビューと最適化サイクル

これらのユースケースは、さまざまな組織のロールの具体的な実装経路を提供するとともに、多様な展開シナリオにわたる AI ファクトリーモデルの包括的な機能を強調します。

Model Serving FAQ

ここでは、AI Factory での Model Serving の使用に関するよくある質問への回答を示します。

現在サポートされているモデルは何ですか？

AI Factory Model Serving バージョン 1.2 は、**NVIDIA NIM** マイクロサービスとしてパッケージされたモデルの展開をサポート

追加のモデルタイプとフォーマットの展開のサポートは、将来のリリースで拡張されます。

サービス提供モデルにはどのような **Kubernetes** リソースが使用されますか？

モデルは、モデルを提供するポッドのライフサイクルを管理する **KServe InferenceServices** を使用して提供されます。

ServingRuntimes や関連する GPU 構成の Kubernetes ノードなどのサポート KServe リソースも使用されます。

どのようなハードウェアが必要ですか？

NIM ベースのモデルサービングには、**GPU 対応の Kubernetes ノード**が必要です。

クラウドプロバイダーによってサポートされている構成とノードの推奨事項については、[How-To Update GPU Resources for an InferenceService](#) を参照してください。

カスタムモデルを展開できますか？

現時点では、AI Factory Model Serving は **NIM ベースのコンテナ用**に最適化されています。KServe を介した任意のモデルの展開のサポートが計画されていますが、現在の AI Factory の合理化されたフローの外部で手動構成が必要です。

スケーリングはどのように処理されますか？

KServe InferenceServices は、**Kubernetes ネイティブのスケーリング**を使用し、互換性のあるランタイムで自動スケーリングがサポートされています。

スケーリング動作は、InferenceService 仕様レプリカ、リソースで構成できます。

実行中のモデルをモニタするにはどうすればよいですか？

次を使用して展開されたモデルをモニタできます。

- **KServe InferenceService ステータス**
- Kubernetes モニタリングツール Prometheus、Grafana など
- [How-To Monitor deployed models with KServe](#)

モデルサービングアーキテクチャの詳細はどこで学べますか？

参照

- [Model Serving Concepts](#)
- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)

もっと質問がありますか？ AI Factory の管理者に連絡するか、AI Factory Support Resources に相談してください。

Manuals

Model Serving Reference Manual

Model Serving は、検証された AI モデルを、EDB Postgres AI 機能とシームレスに統合する運用対応の推論エンドポイントに変換します。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 Kubernetes 環境内のスケーラブルサービスとして、コンテナ化されたモデルを展開します。

前提条件 Model Serving では、モデルイメージ管理に Hybrid Manager installation および [Asset Library configuration](#) が必要です。すべての推論サービスは、制御された Kubernetes 環境内で動作します。

システムアーキテクチャ

KServe Foundation

Model Serving は、基になるオーケストレーションエンジンとして KServe を利用し、モデルの展開、スケーリング、ライフサイクル管理のためのエンタープライズグレードの機能を提供します。KServe は、推論操作の包括的な制御を維持しながら、コンテナ化されたモデル展開の複雑さを抽象化します。

コアコンポーネント

- **ServingRuntime** さまざまなモデルタイプに最適化されたフレームワーク固有の実行環境
- **InferenceService** 自動スケーリングと状態管理を備えた運用エンドポイント
- **リソース管理** 導入されたモデル全体の GPU 割り当てとコンピューティングの最適化

統合フレームワーク

サービスインフラストラクチャは、モデルライブラリの管理対象モデルカタログと EDB PG AI の操作機能間のブリッジとして動作します。ガバナンスワークフローを通じて承認されたモデルは、さまざまな AI ワークロードをサポートする推論サービスとして展開できるようになります。

サポートされているモデルタイプ

- テキスト生成と補完のためのラージ言語モデル LLM
- ベクトル検索および取得拡張生成のためのモデルの埋め込み
- CLIP、OCR、画像埋め込み機能を含むビジョンモデル
- 検索結果最適化のためのモデルの再ランキング

動作特性

導入パターン

InferenceServices は、組織の要件とモデルの特性に基づいて、複数の展開戦略をサポートします。このシステムは、適切なリソース割り当てとスケーリング動作を備えた高可用性の運用展開と開発環境の両方に対応します。

展開構成

- 専用の推論ワークロードの **単一モデル展開**
- 互換性のあるモデルでのリソース効率のための **マルチモデルサービング**
- 段階的なトラフィックシフトを使用した安全な運用更新のための **カナリア展開**
- モデルパフォーマンス比較のための **A/B テスト構成**

スケーリングメカニズム

自動スケーリングは、推論需要に基づいてコンピューティングリソースを調整し、サービスレベル目標を維持しながらコストを最適化します。システムは、ワークロード特性に基づいて、ホリゾンタルポッドスケーリングと垂直リソース調整の両方をサポートしています。

スケーリング係数

- 要求の同時実行性とキューの深さ
- モデル固有のレイテンシー要件
- GPU メモリ使用パターン
- 特殊なワークロードのカスタムメトリック

リソース管理

GPU リソースは、マルチプルのモデル全体で使用率を最適化する集中的な割り当て戦略を介して管理されます。システムは、さまざまな GPU 構成をサポートし、リソースの競合を防ぐための分離メカニズムを提供します。

リソースに関する考慮事項

- モデルのロードと推論のための GPU メモリ要件
- 前処理および後処理操作のための CPU 割り当て
- 高スループットの推論ワークロードのネットワーク帯域幅
- モデルアーティファクトと一時データのストレージ要件

アクセスと統合

エンドポイント構成

InferenceServices は、一般的な AI フレームワークおよびアプリケーションとの互換性を提供する標準化された API エンドポイントを公開します。システムは、安全なゲートウェイ構成を介して内部クラスター通

信と外部アクセスの両方をサポートします。

内部アクセスパターン

- サービス間通信のクラスターローカル DNS 解決
- セキュリティ分離のためのネットワークポリシーの適用
- 高度なトラフィック管理のためのサービスメッシュ統合

外部アクセス構成

- 認証と承認を備えたポータルベースのエンドポイント
- 安全な外部統合のための API キー管理
- レート制限とクォータ適用機能

API の互換性

標準エンドポイントパスは、OpenAI 互換クライアントおよびカスタムアプリケーションとの互換性を提供し、既存の AI ワークフローおよびツールチェーンとのシームレスな統合を可能にします。

エンドポイント仕様

Chat completions (LLMs):	<base>/v1/chat/completions
Embeddings:	<base>/v1/embeddings
Reranking:	<base>/v1/ranking

EDB PG AI 統合

Model Serving は、運用制御を維持しながら展開の複雑さを抽象化する一貫したインターフェイスを介して、すべての EDB PG AI コンポーネントの推論機能を提供します。

統合ポイント

- **AI Accelerator Pipelines** は、データ変換ワークフローの埋め込みモデルと前処理モデルを活用
- **ナレッジベース**は、ベクトル検索およびセマンティック検索操作に埋め込みモデルを利用します
- **Gen AI Builder** アプリケーションは、会話型 AI およびテキスト生成機能の LLM にアクセスします

構成フレームワーク

ServingRuntime Management

ServingRuntime 構成は、さまざまなモデルフレームワークの実行環境を定義し、最適化されたパフォーマンス特性を提供しながら、展開操作からフレームワーク固有の複雑さを抽象化します。

実行時最適化機能

- モデルの読み込みと推論のためのフレームワーク固有の最適化
- GPU カーネルの最適化によるスループットの向上
- 大規模なモデル展開のメモリ管理戦略

- リソース使用率を向上させるバッチ処理機能

How-To Configure a ClusterServingRuntime

リソースの割り当て

適切なリソース構成により、最適なパフォーマンスが保証され、リソース競合やメモリ不足状態が回避されます。組織は、リソース仕様を定義するときに、パフォーマンス要件とインフラストラクチャコストのバランスを保つ必要があります。

重要な構成領域

- コンテナスケジューリングの CPU およびメモリ要求/制限
- GPU の割り当てと共有戦略
- ネットワークとストレージのパフォーマンス要件
- ヘルスプローブとタイムアウト構成

How-To Update GPU Resources for an InferenceService

可観測性構成

包括的なモニタリングにより、推論サービスのプロアクティブな管理が可能になると同時に、パフォーマンス特性とリソース使用パターンの可視性を提供します。

モニタリング機能

- 要求レイテンシーとスループットメトリック
- CPU、メモリ、GPU 全体のリソース使用率の追跡
- 詳細な障害分析を含むエラー率モニタリング
- アプリケーション固有の要件のカスタムメトリック

Observability for Model Serving

運用上の考慮事項

パフォーマンスの最適化

モデルサービスのパフォーマンスは、モデルの特性、ハードウェア構成、トラフィックパターンを含む複数の要素に依存します。組織は、ベースラインのパフォーマンスメトリックを確立し、本番使用パターンに基づいて継続的に最適化する必要があります。

最適化戦略

- 最適なスループットとレイテンシーのトレードオフのためのバッチサイズの調整
- 大幅な精度の損失なしでメモリ効率を高める量子化技術
- 頻繁にアクセスされるモデルと予測のキャッシュ戦略
- マルチモデル展開の要求ルーティングの最適化

セキュリティフレームワーク

実稼働モデルの提供には、業務効率を維持しながらモデル資産と推論データの両方を保護する包括的なセキュリティ制御が必要です。

セキュリティ制御

- Kubernetes ネットワークポリシーを介したネットワークの分離
- API アクセスの認証・承認
- モデルイメージのセキュリティスキャンと脆弱性管理
- コンプライアンスとフォレンジック分析のための監査ログ

スケーラビリティの制限

システムの制限を理解することにより、大規模な展開の適切なキャパシティプランニングとアーキテクチャの決定が可能になります。

スケーリングコンストレイン

- GPU メモリはモデルサイズとバッチ処理機能を制限します
- ネットワーク帯域幅は高スループットの推論ワークロードに影響を与える
- ストレージ I/O パフォーマンスはモデルのロードとチェックポイント操作に影響を与える
- Kubernetes クラスター制限は最大同時展開に影響を与える

トラブルシューティングフレームワーク

よくある問題

モデルサービス展開では、体系的なトラブルシューティングのアプローチを必要とするさまざまな運用上の問題が発生する場合があります。

一般的な問題

- **HTTP 404 エラー**は、多くの場合、不正なエンドポイントパスまたはサービスの構成の誤りを示します
- **リソース割り当ての失敗**は、通常、不十分な GPU またはメモリの可用性が原因で発生します
- **モデルのロードタイムアウト**には、ヘルスプローブ構成の調整が必要になる場合があります
- **パフォーマンスの低下**は、多くの場合、リソース競合または準最適なバッチサイズを示します

診断手順

体系的な診断アプローチにより、効率的な問題の解決が可能になり、トラブルシューティング操作中のサービスの中断を最小限に抑えます。

診断手順

1. InferenceService のステータスとイベントログの確認

2. リソースの割り当てと使用率メトリックの確認
3. ネットワーク接続とエンドポイントのアクセシビリティを検証する
4. モデルのロードログとコンテナステータスを確認します

How-To Verify InferenceServices and GPU usage

実装パターン

開発環境

開発およびテスト環境は、実稼働グレードの信頼性とパフォーマンス特性よりも迅速な反復を優先する簡素化された構成の恩恵を受けます。

開発構成

- コスト効率のためのリソース割り当ての削減
- 外部アクセス要件のない簡素化されたネットワーキング
- 緩和されたヘルスチェックとタイムアウト構成
- 複数の開発モデルにわたる共有 GPU リソース

本番展開

実稼働環境には、運用要件をサポートしながら、信頼性、パフォーマンス、セキュリティを保証する包括的な構成が必要です。

制作要件

- 適切な冗長性を備えた高可用性構成
- 包括的なモニタリングとアラートの統合
- 組織ポリシーに合わせたセキュリティ制御
- 予想されるワークロード特性に基づいたキャパシティプランニング

ハイブリッドアーキテクチャ

組織は、一貫した運用手順を維持しながら、パフォーマンス、セキュリティ、またはコンプライアンス要件に基づいて、複数のインフラストラクチャ層にモデルを展開する場合があります。

ハイブリッドの考慮事項

- さまざまな展開環境間での一貫したモデルバージョン
- 環境間のネットワーク接続と遅延の最適化
- クロス環境通信のセキュリティ境界管理
- ハイブリッド展開全体での統合された監視と管理

はじめに

初期展開

Model Serving を始める組織は、単純な構成から始めて、運用経験と要件に基づいて徐々に複雑さを追加する必要があります。

セットアップシーケンス

1. ハイブリッドマネージャーとアセットライブラリーのインストールの確認
2. ターゲットモデルフレームワークの基本的な `ServingRuntime` の構成
3. 保守的なリソース割り当てを使用して初期推論サービスを展開する
4. エンドポイントのテストとモニタリングを介して機能を検証する

構成リソース

- *How-To Create an InferenceService for NVIDIA NIM Container*
- *How-To Setup GPU resources*
- *Accessing KServe endpoints (internal and external)*

ベストプラクティス

- カスタマイズの前に、実績のある `ServingRuntime` 構成から開始する
- 運用展開前に包括的なモニタリングを実装する
- モデルの特性に基づいて、リソース割り当てガイドラインを確立する
- 構成の決定事項と操作手順を文書化する

リファレンスドキュメント

コアアーキテクチャ

- *Gen AI Architecture (Entities and Flows)*
- *Model Library Reference Manual*

実装ガイド

- *How-To Configure a ClusterServingRuntime*
- *How-To Create an InferenceService for NVIDIA NIM Container*
- *Observability for Model Serving*

統合ドキュメント

- *AI Accelerator Pipelines*
- *Gen AI Builder*
- *Hybrid Manager Asset Library*

Model Serving は、ガバナンス AI モデルを、パフォーマンス、セキュリティ、操作特性の包括的な制御を維持しながら、組織の要件に合わせてスケールする本番対応の推論機能に変換します。

Model Library Reference Manual

****モデルライブラリー*** は、EDB Postgres AI 内の AI モデルイメージのセントラルガバナンスシステムとして機能し、モデルのライフサイクルの組織的な制御を提供しながら、パイプライン、モデルサービング、ナレッジベース、Gen AI Builder 全体の統合を可能にします。

前提条件モデルライブラリーには、*Hybrid Manager Asset Library* のインストールが必要です。
すべてのモデルのメタデータ、バージョンニング、およびライフサイクル操作は、アセットライブラリーインフラストラクチャを介して管理されます。

目的と機能

AI を大規模に実装する組織は、モデル管理に関する基本的な課題に直面しています。断片化したりポジトリ、バージョンの不一致、コンプライアンスギャップ、運用の可視性の制限。モデルライブラリーは、開発速度を犠牲にすることなく制御を維持する集中ガバナンスを介してこれらの課題に対処します。

このシステムは、外部モデルソースと内部 AI 機能間の厳選されたゲートウェイとして動作し、検証されたモデルのみが運用環境に到達することを保証しながら、コンプライアンス要件の包括的な監査証跡を提供します。

コアアーキテクチャ

レジストリ統合レイヤー

モデルライブラリーは、構成可能な同期ルールを介して外部コンテナレジストリに接続します。このレイヤーは、セキュリティ境界を維持しながら、組織の基準に基づいてモデルイメージを自動的に検出、検証、および取り込みます。

サポートされているレジストリタイプ

- Harbor プライベートコンテナレジストリ
- Amazon Elastic Container Registry ECR
- Azure Container Registry ACR
- Google Container Registry GCR
- カスタムレジストリ実装

ガバナンスフレームワーク

ビルトインの承認ワークフローは、運用が可能になる前に組織のポリシーに対してモデルを検証します。このフレームワークは、標準コンプライアンス要件の自動検証を維持しながら、複雑なマルチステークホルダーの承認プロセスをサポートします。

ポリシー適用ポイント

- セキュリティスキャンと脆弱性評価

- 組織基準に対するパフォーマンスベンチマーク
- 規制要件のコンプライアンス検証
- モデルの来歴のデジタル署名検証

ディストリビューションハブ

承認されたモデルは、標準化されたインターフェイスを介してすべての EDB PG AI 機能で利用できるようになります。これにより、環境固有の構成をサポートしながら、開発、ステージング、および実稼働環境全体で一貫したモデルバージョンが保証されます。

運用ワークフロー

モデル登録プロセス

自動検出リポジトリルールは、新しいモデルバージョンを監視する外部ソースを定義します。システムは、構成されたレジストリを継続的にスキャンし、定義された基準と一致するモデルを自動的に取り込み、ポリシーコンプライアンスを保証しながら手動オーバーヘッドを削減します。

検証パイプライン新しく取り込まれたモデルは、セキュリティスキャン、メタデータ抽出、パフォーマンス検証を含む自動検証を受けます。検証チェックに失敗したモデルは、問題が解決されるか、オーバーライド手順を介して明示的に承認されるまで、隔離されます。

承認ルーティング検証されたモデルは、分類とリスク評価に基づいて承認ワークフローに入ります。システムは、モデルを適切な関係者にルーティングしながら、事前定義された基準を満たすモデルの自動承認を提供します。

ライフサイクル管理

バージョン制御システムは、登録されたすべてのモデルの不変のバージョン履歴を維持し、セマンティックバージョンスキームと下位互換性検証をサポートします。バージョン追跡により、複雑なモデルエコシステム全体でのロールバック機能と依存関係管理が可能になります。

プロモーションパイプラインモデルは、検証チェックポイントと承認ゲートに基づいて、環境層開発 → ステージング → 運用を介して進行します。自動プロモーションにより、品質管理を維持しながら展開の手間が削減されます。

非推奨の処理構成可能なリテンションポリシーは、モデルのライフサイクルの移行を管理し、非推奨になる前に事前通知を提供し、アクティブな展開を中断せずに安全なリタイアを保証します。

統合ポイント

モデルサービスプラットフォーム

モデルライブラリーは、*Model Serving Reference Manual* 展開用の厳選されたカタログを提供します。ガバナンスフレームワークを介して承認されたモデルのみがプロダクション推論エンドポイントに展開でき、一貫したセキュリティとコンプライアンス標準を保証します。

統合のメリット

- 検証チェックポイントを使用した自動モデルの更新

- 問題のある展開のロールバック機能
- リソース割り当てのパフォーマンスメタデータの最適化
- シームレスな KServe インフラストラクチャの統合

AI アクセラレーターパイプライン

パイプラインの実行は、ライブラリの検証済みモデルを活用し、*AI Accelerator Pipelines* ステージと環境全体での一貫性を保証します。システムは、パイプラインリソースの割り当てと実行戦略を最適化するパフォーマンス特性を提供します。

Gen AI Builder

Gen AI Builder アプリケーションは、ライブラリインターフェイスを介して管理モデルコレクションにアクセスします。これにより、ナレッジベース実装は検証された埋め込みモデルを使用し、アシスタント作成は承認された言語モデルを参照します。

セキュリティとアクセス制御

マルチテナントアーキテクチャ

ロールベースのアクセス制御は、組織階層と連携しながら、さまざまなチームやプロジェクトに名前空間の分離を提供します。詳細な権限は、ユーザーロールと組織ポリシーに基づいて、モデルの登録、承認、および展開アクティビティを制御します。

セキュリティ検証

自動脆弱性スキャンは、登録されているすべてのモデルイメージを調べて、既知のセキュリティ問題がないかを調べます。システムは、登録時にモデルの署名と来歴情報を検証し、リスクのある展開を防ぐポリシーベースの制御を実装します。

セキュリティ機能

- モデルの信頼性のデジタル署名検証
- 脅威検出のための脆弱性データベースの統合
- 自動セキュリティ制御適用のためのポリシーエンジン
- セキュリティモニタリングのための包括的な監査ログ

実装の考慮事項

レジストリ構成

レジストリ接続を確立するには、ネットワークプロトコル、認証メカニズム、および同期頻度を注意深く検討する必要があります。組織は、リポジトリルールを構成するときに、自動化のメリットとセキュリティ要件のバランスを保つ必要があります。

重要な要素

- ネットワーク接続とファイアウォールの要件

- 認証および承認メカニズム
- 同期頻度とリソースへの影響
- エラー処理と再試行戦略

How-To Integrate Private Registry with Model Library

承認ワークフローのデザイン

承認プロセスでは、ガバナンス要件と開発速度のバランスを保つ必要があります。組織は、高リスクモデルの適切な監督を保証しながら、日常的な承認を自動化するワークフローをデザインする必要があります。

デザイン原則

- 効率的なステークホルダーの参加のためのリスクベースのルーティング
- 標準基準を満たすモデルの自動承認
- 複雑な承認シナリオの明確なエスカレーションパス
- コンプライアンス要件の包括的な監査証跡

How-To Define Repository Rules for Model Library

メタデータ管理

包括的なメタデータのキャプチャは、検出、ガバナンス、および運用要件をサポートします。組織は、ビジネス要件に合わせた一貫したタグ付けスキームとドキュメント標準を確立する必要があります。

How-To Manage Repository and Image Tag Metadata

操作パターン

エンタープライズガバナンス

大規模な組織は、Model Library の洗練された承認ワークフローと包括的な監査機能のメリットを活用します。このシステムは、インテリジェントな自動化を通じて開発速度を維持しながら、複雑なマルチステークホルダーのプロセスをサポートします。

一般的な実装

1. 開発チームはステージング環境にモデルを登録します
2. 自動検証はセキュリティとコンプライアンス要件を検証します
3. リスクベースのルーティングは、モデルを適切な承認関係者に転送します
4. 承認されたモデルは、すべての EDB PG AI 機能に伝播されます
5. 運用展開は、検証されたバージョンを自動的に参照します

ソブリン AI の展開

完全なモデルの主権を必要とする組織は、プライベートレジストリの統合とエアギャップ展開機能を活用します。このパターンにより、エンタープライズ運用機能を維持しながら、機密モデルが組織の境界を離れることはありません。

主な特徴

- オフライン配布による完全なエアギャップのサポート
- メタデータを保持するプライベートレジストリの統合
- 規制要件の自動コンプライアンスレポート
- 機密モデルの処理手順と制御

開発の一貫性

チームは、開発ライフサイクル全体で一貫したモデルの可用性の恩恵を受けます。モデルライブラリーは、環境間のバージョンのドリフトを排除しながら、環境固有の構成と展開戦略をサポートします。

制限と考慮事項

パフォーマンスへの影響

レジストリの同期とモデル検証のプロセスは、計算リソースを消費します。組織は、同期頻度と検証の深さを構成するときに、インフラストラクチャの容量を考慮する必要があります。

ネットワークの依存関係

システムでは、同期操作のために外部レジストリへの信頼できるネットワーク接続が必要です。ネットワーク障害または接続の問題は、モデルの可用性と同期スケジュールに影響を与える可能性があります。

ストレージ要件

モデルイメージのストレージは、組織モデルインベントリに合わせて拡張されます。広範なモデルカタログを持つ大規模な組織は、重要なストレージ要件と関連するバックアップ戦略を計画する必要があります。

はじめに

初期セットアップ

1. ハイブリッドマネージャーのインストールと構成の確認
2. 必要な外部レジストリへのネットワーク接続を確立する
3. モデル検出の初期リポジトリルールの構成
4. 組織の要件に合わせた基本的な承認ワークフローを実装する

構成リソース

- *How-To Integrate Private Registry with Model Library*
- *How-To Define Repository Rules for Model Library*
- *How-To Manage Repository and Image Tag Metadata*

ベストプラクティス

- すべてのモデル操作に最小特権のアクセス制御を実装する
- 登録済みモデルの定期的なセキュリティスキャンスケジュールを確立する
- 包括的なバックアップとディザスタリカバリー手順を維持する
- 承認ワークフローとエスカレーション手順を明確に文書化する

リファレンスドキュメント

コアコンポーネント

- *Hybrid Manager Asset Library*
- *Gen AI Architecture (Entities and Flows)*

統合機能

- *Model Serving Reference Manual*
- *AI Accelerator Pipelines*
- *Gen AI Builder*

実装ガイド

- *Model Library Reference Manual*
 - *How-To Define Repository Rules for Model Library*
-

モデルライブラリーは、断片的なモデル管理を、組織の AI イニシアチブに合わせて拡張する体系的なガバナンスに変換し、すべての EDB Postgres AI 機能にわたってセキュリティ、コンプライアンス、およびオペレーショナルエクセレンスを維持します。

Observability for Model Serving

可観測性は、展開した AI モデルが AI Factory 内で効率的かつ確実に実行されていることを確認するのに役立ちます。

AI Factory でのモデル提供は、Kubernetes で KServe を使用してモデルを提供します。これにより、以下を監視するビルトインオプションが提供されます。

- モデル提供ステータスと可用性
- リソース使用量 CPU、メモリ、GPU

- 推論パフォーマンスとスループット

主要なモニタリング機能

KServe InferenceService ステータス

Kubernetes を介してモデルの提供ステータスを直接検査できます。

```
kubectl get inferencesservice -n <namespace>
```

一般的なステータスフィールドには次のものが含まれます。

- Ready / NotReady
- URL エンドポイント
- 現在のレプリカ
- 割り当てられたリソース GPU、CPU、メモリ

詳細検査の場合

```
kubectl describe inferencesservice <name> -n <namespace>
```

GPU 使用率モニタリング

モデルが GPU ノードに展開されている場合、GPU の使用状況をモニタしてリソース割り当てを最適化します。

例

```
kubectl top node
```

より詳細な GPU 固有のメトリックサポートされている場合

```
nvidia-smi
```

Prometheus と Grafana の統合

Prometheus が構成されている場合、AI Factory モデルサービングは、KServe を介してメトリックを公開します。

スクレイピングを有効にする Prometheus アノテーション

```
serving.kserve.io/enable-prometheus-scraping: "true"  
prometheus.kserve.io/port: "8000"  
prometheus.kserve.io/path: "/v1/metrics"
```

Grafana ダッシュボードを構築して、以下を監視できます。

- 1 秒あたりの推論要求
- レイテンシーとエラー率
- GPU 使用率の傾向
- ポッドの再起動とヘルス状態

ログとデバッグ

モデル提供ポッドから詳細なログにアクセスできます。

```
kubectll logs -f <pod-name> -n <namespace>
```

ログを使用して次のことを行います。

- モデルのロードまたは初期化の問題をデバッグします
- 推論要求の動作を確認します
- モニタ応答処理とエラー条件

Hybrid Manager の可観測性の統合

ハイブリッドマネージャー HM と組み合わせて AI Factory を実行する場合、モデルサービング可観測性をより広範な HM 可観測性スタックに統合できます。

Model Serving で HM 可観測ツールを使用するための追加ガイダンスについては、[How-To Verify Inference-Services and GPU usage](#) を参照してください。

ベストプラクティス

- 本番環境の InferenceServices で Prometheus スクレイピングを有効にします。
- GPU の使用率を定期的に監視して、容量を最適化します。
- InferenceService 仕様で適切なリソース要求/制限を設定します。
- Grafana ダッシュボードを活用して、リアルタイムの可観測性を実現します。
- ログ検査を使用して、デバッグとチューニングをサポートします。

関連リンク

- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe](#)
- [Model Serving Concepts](#)
- [How-To Verify InferenceServices and GPU usage](#)

これらの可観測性プラクティスを適用することにより、AI Factory Model Serving ワークロードをより自信を持って操作し、ユーザーに影響を与える前に問題をプロアクティブに特定して対処できます。

Explainers

Model Serving Explained

AI Factory の **Model Serving** を使用すると、Kubernetes インフラストラクチャで実行されるスケーラブルな運用グレードの推論サービスとして AI モデルを展開できます。

KServe に基づいた **Kubernetes ネイティブアーキテクチャ**を提供し、モデルにネットワークアクセス可能な API を介して予測と埋め込みを提供できる機能を提供します。

AI Factory Model Serving は、エンタープライズクラスの AI ワークロードをサポートするように最適化されています。

- GPU 高速化インフラストラクチャ
- 柔軟なスケーリング
- 統合された可観測性
- ソブリン AI アライメント — ガバナンスの下で実行されるモデル
- Gen AI Builder、ナレッジベース、およびその他の AI Factory パイプラインとの シームレスな統合

始める前に

モデルサービスを理解するための前提条件

- **Kubernetes** の基本に関する知識
- **KServe** および **InferenceService** の理解
- **モデルライブラリー** の認識 → AI Factory のモデル提供ワークフロー
- **ソブリン AI** 原則の理解 — ガバナンスの下で実行されるモデル

推奨される読書

- [Model Serving Concepts](#)
- [Model Library Explained In Depth](#)
- [Fundamental AI Concepts](#)

モデルサービスの仕組み

コアスタック

Layer	Purpose
AI Factory	Provides infrastructure and Model Serving APIs
Hybrid Manager Kubernetes Cluster	Hosts model-serving workloads
KServe	Manages model serving lifecycle and APIs
InferenceService	Deployed model resource
Model Library	Manages model image versions
GPU Nodes	Run high-performance model serving pods
User Applications	Call model endpoints via REST/gRPC

主要コンポーネント

- **InferenceService** — デプロイされたモデルを表す Kubernetes CRD。
- **ServingRuntime / ClusterServingRuntime** — 再利用可能なランタイム構成を定義します。
- **モデルコンテナ** - 現在、AI Factory 1.2 の **NVIDIA NIM** コンテナに焦点を当てています。
- **可観測性** — 統合された Prometheus 互換のメトリック、Kubernetes ロギング。

サポートされているモデル

AI Factory Model Serving は、現在 **NVIDIA NIM** コンテナをサポートしています。

Model Type	Example Usage
Text Completion	LLM agents, Assistants
Text Embeddings	Knowledge Bases, RAG
Text Reranking	RAG pipelines
Image Embeddings	Multi-modal search
Image OCR	Document extraction

Supported models を参照してください。

デプロイメントアーキテクチャ

Applications → Model Endpoints (REST/gRPC) → KServe → GPU-enabled Kubernetes → Model Containers

- 各モデルは、独自の **InferenceService** に分離されます。
- KServe は以下を管理します

- モデルのライフサイクル 開始、停止、更新
- スケーリング スケールトゥゼロを含む
- エンドポイントルーティング REST / gRPC
- GPU リソースは、ハイブリッドマネージャー統合を介してプロビジョニングおよびスケジュールされます。

使用パターン

Gen AI Builder

- LLM エンドポイントは、**アシスタント** および **エージェント**を強化します。
- 埋め込みモデルは、ハイブリッド RAG パイプラインをサポートします。

ナレッジベース

- 埋め込みモデルは、**ベクトル化** のニーズに対応します。
- 取得と再ランキングモデルは、**セマンティック検索**パイプラインを強化します。

カスタムアプリケーション

- ビジネスアプリケーションは、次の **InferenceService** エンドポイントを消費できます。
- リアルタイム予測
- 画像分析
- テキスト処理

ベストプラクティス

- **モデルライブラリー** → **モデル提供**フローを介してモデルを展開して、ガバナンスを保証します。
- 再利用可能なランタイム構成には **ClusterServingRuntime** を使用します。
- **GPU 使用率** とモデルのレイテンシーを注意深く監視します。
- 運用環境での準備のために **スケールトゥゼロ** 構成をテストします。
- **モデルライブラリータグ**がバージョン管理され文書化されていることを確認します。
- ソブリン AI ガバナンスの一部として、展開された **InferenceServices** を定期的に監査します。

概要

AI Factory の Model Serving は、**運用 AI モデル**を提供するための堅牢でスケーラブルなアーキテクチャを提供します。

- KServe を使用した Kubernetes ネイティブサービング
- GPU アクセラレーションと最適化されたサービングランタイム
- 統合されたオブザーバビリティとガバナンス
- AI Factory コンポーネントとの緊密な統合 Gen AI Builder、ナレッジベース、カスタム AI パイプライン

Model Serving は、**ソブリン AI** をモデル、インフラストラクチャ、制御下で実装するのに役立ちます。

次のステップ

- *How-To Create an InferenceService for NVIDIA NIM Container*
- *How-To Verify InferenceServices and GPU usage*
- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe*
- *Model Library Explained In Depth*

Model Serving は、**EDB Postgres® AI** の一部として、インテリジェントなアプリケーションとデータ製品を安全に、スケーラブルに、ガバナンスの下で構築するための強力な基盤を提供します。

Model Library Explained In Depth

AI Factory の **モデルライブラリー** は、EDB PG AI 内の AI モデルを検出、管理、および展開するための中心的なインターフェイスです。

これは、ハイブリッドマネージャー HM のコア**イメージとモデルライブラリー* を利用しています。モデルライブラリーは、実稼働使用のために検証されたモデル画像の厳選された AI 中心のビューを提供します。

モデルライブラリーを介して展開されたモデルイメージは、次のような AI Factory 機能を強化します。

- **Gen AI Builder** アシスタントとパイプライン
- **ナレッジベースパイプライン** 埋め込み、RAG
- モデルサービスを使用する **その他の AI Factory アプリケーション**

始める前に

モデルライブラリーを理解するための前提条件

- **Model Serving** および **KServe** の概念に関する基本的な知識
- **InferenceService** とランタイムモデルの理解
- ハイブリッドマネージャーの **イメージとモデルライブラリー**アーキテクチャの認識
- 役立つ **ソブリン AI 原則** の認識

推奨されるパス

- *Fundamental AI Concepts*
- *Model Serving Concepts*

モデルライブラリーとは何ですか？

モデルライブラリー

- より広範な画像とモデルライブラリーの **厳選された AI ビュー**を提示します
- ユーザーは検証されたモデルイメージを **モデルサービング KServe** レイヤーに展開できます
- AI Factory エクスペリエンスを強化するモデルのライフサイクル管理を提供します

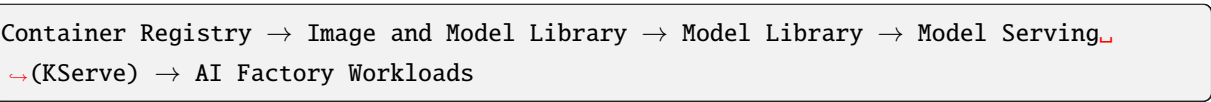
キーポイントモデルライブラリーは別のレジストリではありません。これは、**管理されたイメージとモデルライブラリー** 上の制御された **ビュー**です。

すべてのモデル画像

- コアアセットライブラリーパスを介したフロー
- プラットフォーム全体の **ガバナンス** および **セキュリティポリシー**の対象となります
- AI Factory での使用が検証された場合にのみモデルライブラリーで利用可能になります

それはどのように機能しますか？

アーキテクチャフロー



Layer	Role
Container Registry	Stores model container images (e.g., NVIDIA NIM, open models)
Image and Model Library	Single source of truth for all container images
Model Library	AI-focused console for validated model images

次のページに続く

表 4 – 前のページからの続き

Layer	Role
Model Serving (KServe)	Runs deployed model instances as InferenceServices
AI Factory Workloads	Gen AI Builder, Knowledge Base pipelines, Assistants, AI applications

ワークフロー

1. 画像は **アセットライブラリー** プライベートまたはパブリックレジストリに取り込まれます。
2. AI モデル提供について検証された画像は、 **モデルライブラリー** に表示されます。
3. ユーザーは、KServe を介してモデルを参照、選択、および **Model Serving** に展開します。
4. 展開モデルのパワー
 - ナレッジベースの取り込みと取得
 - **Gen AI Builder** アシスタントとパイプライン
 - その他の **AI Factory** 機能

モデルライブラリーを使用する理由

- **統合ガバナンス** – すべての画像は同じガバナンスプロセスを介してフローします。
- **セキュリティと監査可能** – レジストリ統合、タグ選択、およびイメージ監査が統合されています。
- **ソブリン AI** – どのモデルを使用するか、展開方法、および実行場所を制御します。
- **一貫性** – データベースと AI モデルは、同じコンテナイメージ管理システムを共有します。
- **柔軟性** – アセットライブラリーパスを介して独自のモデル画像を追加し、AI Factory での可用性を制御できます。

モデルライブラリーは何を提供しますか？

ユーザーは次のことを行うことができます。

- 利用可能な **AI モデル画像** を参照します
- サポートされている **タグとバージョン** を表示します
- **Model Serving** インフラストラクチャにモデルを展開する
- モデル展開のライフサイクルを管理する
- 現在の **AI Factory** ワークロードを強化している **モデル** を理解します。
- ナレッジベース AIDB の取り込みと取得経由
- **Gen AI Builder** アシスタントとパイプライン用

サポートされているモデルタイプ

Model Type	Example Image
Text Completion	llama-3.3-nemotron-super-49b
Text Embedding	arctic-embed-l
Image Embedding	nvclip
OCR	paddleocr
Text Reranker	llama-3.2-nv-rerankqa-1b-v2

- 追加のモデルタイプのサポートが計画されていますビジョン、マルチモーダル、RAG 最適化。
- カスタムモデルは、プライベートレジストリ統合を介して追加できます。

使用パターン

ナレッジベース

- ナレッジベースのインデックス作成 および **RAG** パイプラインで使用される埋め込みモデルと取得モデル。
- 例 **arctic-embed-l** または **nvclip** モデルを使用します。

Gen AI Builder アシスタント

- モデルライブラリーを介して展開されたモデルを使用した Power Assistant
- テキスト生成用の LLMs 例 llama-3
- OCR または Vision モデルたとえば、paddleocr
- Model Serving エンドポイントを使用したカスタムツール

ハイブリッド+ソブリン AI の連携

- すべてのモデルは、ハイブリッドマネージャー KServe レイヤーを介してインフラストラクチャで実行されます。
- あなたは以下をコントロールします
- どのモデルが展開されているか
- リソース割り当て CPU / GPU
- デプロイメントトポロジ
- 可観測性と監査

カスタムモデルの統合

- **Integrate Private Registry** フローを使用して、独自のレジストリからモデルを追加します。
- **リポジトリルール** を定義して、イメージの可用性を制御します。
- モデルライブラリーに表示される前に、画像がガバナンスチェックに合格することを確認します。

ベストプラクティス

- モデルイメージに対する **明確なガバナンス** を維持します。
- **リポジトリルール** を使用して、可視性とコンプライアンスを管理します。
- ハイブリッドマネージャーダッシュボードで **モデルの使用率**とリソース消費を監視します。
- 実稼働で使用されるバージョンとドキュメントモデル。
- AI Factory ワークロードを有効にする前に、カスタムモデルを徹底的にテストします。

モデルサービングの統合

モデルライブラリーから展開されたモデルは、 **Model Serving KServe** レイヤーで実行されます。

- Model ライブラリーコンソールで提供されるガイド付きワークフロー
 1. モデル画像とタグを選択します。
 2. ランタイムレプリカ、リソース、GPUなどを構成します。
 3. Deploy → HM Kubernetes 環境に **InferenceService** を作成します。
- 導入モデルのパワー
- **AIDB ナレッジベースの取り込み**
- **Gen AI Builder** アシスタントとパイプライン
- その他の AI Factory サービス

関連トピック

- *Model Library Reference Manual*
- *How-To Deploy AI Models from Model Library*
- *How-To Integrate Private Registry with Model Library*
- *How-To Define Repository Rules for Model Library*
- *Model Serving Concepts*
- *Pipelines models reference*

次のステップ

- モデルライブラリー で使用可能なモデルを参照することから始めます。
- ナレッジベース インジェストまたは **Gen AI Builder Assistants** 用の最初のモデルを展開します。
- モデルサービングハウツーガイド を参照して、展開を最適化します。
- モデルイメージフローのガバナンスとコンプライアンスポリシーを定義します。

How to set up serving

How-To Integrate Private Registry with Model Library

プライベートレジストリとモデルライブラリーを統合する

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、組織の **プライベートコンテナレジストリ** を AI Factory **モデルライブラリー** と統合する方法について説明します。

統合が完了すると、カスタム構築または内部承認のモデルイメージがモデルライブラリーコンソールに表示され、Model Serving に展開する準備が整います。

このガイドは誰が使用する必要がありますか？

- コンテナレジストリガバナンスを担当する AI プラットフォーム管理者。
- AI イメージのプライベートレジストリを管理する DevOps エンジニア。
- AI Factory を介してプライベート AI モデルを展開する必要がある開発者。

これによりできること

- Hybrid ManagerHM にプライベートレジストリを登録できます。
- HM は、レジストリで AI モデルイメージを検出できます。
- これらのイメージは、Model Serving に展開するために**モデルライブラリー** で利用できます。
- AI Factory のワークロード全体でプライベートモデルイメージの可視性と使用を制御できます。

完了までの推定時間

レジストリ構成ごとに 10～15 分。

前提条件

始める前に

- プライベートコンテナレジストリ AWS ECR、GCP GCR、Azure ACR、Harbor、または同様のものの管理アクセスが必要です。
- HM への管理アクセスが必要ですレジストリ統合を構成するには。
- レジストリに必要な資格情報ユーザー名/パスワードまたはトークンを知っている必要があります。
- レジストリは、互換性のある Docker Registry API v2 エンドポイントを公開する必要があります。

サポートされているレジストリタイプ

- AWS Elastic Container Registry ECR
- Google Container Registry GCR / Artifact Registry
- Azure Container Registry ACR
- ハーバー
- Docker Registry API v2 をサポートする汎用レジストリ

統合の手順

1. レジストリ資格情報の準備

- パブリックレジストリの場合 → 認証は必要ありません。
- プライベートレジストリの場合 → 次のいずれかを準備します
 - ユーザー名とパスワードのペア
 - 個人アクセストークン
 - ロボットアカウントの資格情報 Harbor

これらの資格情報を使用して、モデルイメージの `docker pull` をローカルで実行できることを確認します。

2.HM にレジストリを登録する

- コンソールで、次の場所に移動します。

AI Factory > Model Library > Manage Repositories > Add Registry

次のことを求めるダイアログが表示されます。

Field	Description
Registry URL	完全なレジストリホスト名たとえば <i>myregistry.company.com</i>
Registry Type	サポートされているレジストリタイプから選択してください
Username	(オプション) 認証用のユーザー名
Password/Token	(オプション) 認証用のパスワードまたはトークン

次のページに続く

表 6 – 前のページからの続き

Field	Description
Registry Name (Label)	モデルライブラリーコンソールに表示されるフレンドリ名

- 必須フィールドに入力します。
- **Add Registry** をクリックします。

3. レジストリ統合の検証

追加した後

- HM はレジストリに接続し、資格情報を検証しようとします。
- 成功すると、レジストリが **リポジトリの管理** リストに表示されます。
- HM は、このレジストリからリポジトリタグの定期的な同期を開始します。

4. リポジトリルールを定義する

検出されるリポジトリ/タグを制御するには

- レジストリを追加した後、**リポジトリルール** を構成します。
- *How-To Define Repository Rules for Model Library* を参照してください

5. イメージの可用性を検証する

同期が完了したら

- **AI Factory > Model ライブラリー** に移動します。
- レジストリスコープまたはリポジトリフィルタを選択します。
- プライベートモデル画像が正しいタグで表示されることを確認します。

通常の **Deploy to Model Serving** フローを介してこれらのイメージを展開できるようになりました。

ヒントとベストプラクティス

- 個人情報情報の公開を回避するには、可能な場合は **ロボットアカウント** または **トークンベースの認証** を使用します。
- **リポジトリルール** を介して検出範囲を制限する – レジストリ全体の同期を回避します。
- モデルライブラリーでの選択を支援するために、バージョン情報で画像を明確にタグ付けします。
- マルチテナント環境の場合、レジストリの可視性を注意深くセグメント化します。

障害対応

レジストリ接続に失敗しました

- レジストリ URL を確認します **https://** は含めず、ホスト名のみを含めてください。

- `docker login` を手動でテストすることにより、資格情報を検証します。
- Registry API v2 が有効になっていることを確認します。

画像が表示されない

- リポジトリルールに関連するリポジトリが許可されていることを確認します。
- 画像に有効なタグがあることを確認します。
- 定期的な同期が完了したことを確認します。

認証エラー

- AWS ECR の場合 → IAM 権限で `ecr:GetAuthorizationToken` が許可されていることを確認します。
- Azure ACR の場合 → トークンに `acrPull` ロールがあることを確認します。
- GCR の場合 → Artifact Registry アクセスを持つサービスアカウントを使用します。

概要

- プライベートレジストリをモデルライブラリーと統合できます。
- プライベートモデルイメージは、モデル提供を介して展開できるようになります。
- リポジトリルールを介して可視性を管理できます。
- Model ライブラリーは、AI ワークロードのパブリックとプライベートの両方の画像ソースを統合します。

関連リンク

- [*Model Library Explained In Depth*](#)
- [*How-To Define Repository Rules for Model Library*](#)
- [*Model Library Explained In Depth*](#)
- [*How-To Deploy AI Models from Model Library*](#)

How-To Define Repository Rules for Model Library

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、**リポジトリルール**を定義して、接続されたレジストリのどのモデルイメージリポジトリとタグを AI Factory **モデルライブラリー**に表示するかを制御する方法について説明します。

リポジトリルールは以下の管理に役立ちます。

- 画像検出の **スコープ**
- 画像が公開される **ガバナンス**
- 可視するタグを制御することによる **バージョン管理**

このガイドは誰が使用する必要がありますか？

- プライベートレジストリを統合する AI プラットフォーム管理者
- コンテナレジストリの統合を管理する DevOps エンジニア
- AI モデル画像の使用を管理するセキュリティ/コンプライアンスチーム

これによりできること

- HM が同期するイメージリポジトリを正確に制御できます。
- 各リポジトリ内のどのタグを公開するかをフィルタリングできます。
- モデルライブラリーで展開可能なイメージを制御できます。

完了までの推定時間

リポジトリルールごとに 5～10 分。

前提条件

- 既に *How-To Integrate Private Registry with Model Library* が必要です。
- レジストリは **AI Factory > Model ライブラリー > Manage Repositories** に表示される必要があります。

リポジトリルールを理解する

リポジトリルールは、構成済みの各レジストリにアタッチされます。

これらにより、以下を定義できます。

- **どのリポジトリ** を検出するか
- これらのリポジトリ内の **どのタグ** を公開するか

これにより、テストイメージ、未検証のモデルビルド、またはレジストリコンテンツ全体が誤って公開されるのを防ぎます。

リポジトリルールは **定期的に評価**されます。ルールに一致する新しくプッシュされた画像は、次回の同期時にモデルライブラリーに表示されます。

リポジトリルールの仕組み

通常、各ルールには次のものが含まれます。

Field	Description
Repository Name	含めるリポジトリの名前完全一致またはパターン
Tag Filter	(オプション) タグ名のフィルタ完全一致、プレフィックス、正規表現
Rule Enabled	このルールが現在アクティブかどうか

例

Repository Name	Tag Filter
<i>my-company/llama3-models</i>	<i>^release-.*</i>

→ my-company/llama3-models の release- で始まるタグのみが表示されます。

リポジトリルールの定義

1.[リポジトリルール]に移動します

- 次の場所に移動します

AI Factory > Model Library > Manage Repositories

- 目的のレジストリの **Manage Rules** をクリックします。

2. リポジトリルールを追加する

- **Add Rule** をクリックします。
- Fill in
 - リポジトリ名 — 必須
 - タグフィルタ — オプショナル
 - 有効/無効トグル
- **Save** をクリックします。

ルールがルールリストに表示されます。

3. 次の同期を待機するか、手動同期をトリガーします

HM は、次のレジストリ同期サイクルにルールを適用します。

UI が手動同期をサポートしている場合

- ルールを追加/更新した後、 **Sync Now** をクリックします。

4. モデルライブラリーのコンテンツの検証

- **AI Factory > Model ライブラリー** に移動します。
- フィルタを使用して、レジストリからイメージを参照します。
- 目的のリポジトリ/タグのみが表示されることを確認します。

ユースケースの例

基本: 単一のリポジトリに制限する

Repository Name	Tag Filter
<i>my-org/nim-models</i>	(blank — all tags)

→ *my-org/nim-models* のすべてのタグが表示されます。

Advanced 実稼働タグにフィルタリングする

Repository Name	Tag Filter
<i>models/gpt4</i>	<i>^prod-.*</i>

→ *prod-* で始まるタグのみが表示されます。

マルチリポジトリのガバナンス

- 必要に応じて、レジストリごとに複数のリポジトリルールを追加します。
- タグフィルターを使用して、モデルのプロモーションパイプライン開発、ステージング、製品版と調整します。

ヒントとベストプラクティス

- 明示的に — 公開するリポジトリのみのリポジトリルールを追加します。
- タグフィルタを使用して、バージョンガバナンスと CI/CD プロモーションステージを適用します。
- 定期的に リポジトリルールを監査して、内部イメージ標準へのコンプライアンスを確認します。
- レジストリで構造化タグ *prod-*、*test-* などを使用している場合、タグフィルターを活用して実験モデルと運用モデルを分離します。

障害対応

画像が表示されない

- 目的のリポジトリにリポジトリルールが存在することを確認します。
- タグフィルタを使用する場合 — タグがフィルタと正確に一致することを確認します。
- 最後に同期が実行されたときを確認します。必要に応じて、手動同期をトリガーします。

表示される画像が多すぎます

- リポジトリルールを調整します。
- タグフィルタを狭い範囲に追加します。

リポジトリルールは効果がありません

- リポジトリ名が正確であることを確認します大文字と小文字が区別されます。
- ルールが有効になっていることを確認します。
- 問題が解決しない場合は、手動同期をトリガーし、HM ログを確認します。

概要

- リポジトリルールは、モデルライブラリーに表示されるイメージリポジトリとタグを制御します。
- リポジトリ名とタグフィルタによって公開の範囲を指定できます。
- これは、ガバナンス、バージョン管理、およびセキュリティのベストプラクティスをサポートします。
- 詳細な制御により、精査されたモデルイメージのみが展開されます。

関連リンク

- [Model Library Explained In Depth](#)
- [How-To Integrate Private Registry with Model Library](#)
- [How-To Deploy AI Models from Model Library](#)
- [Model Library Explained In Depth](#)

How-To Manage Repository and Image Tag Metadata

リポジトリとイメージタグのメタデータを管理する

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、リポジトリのメタデータと、イメージおよびモデルライブラリーでイメージタグを管理する方法について説明します。適切なメタデータは、AI モデル提供と Postgres クラスタープロビジョニングに使用されるイメージを追跡、整理、および管理するのに役立ちます。

このガイドは誰が使用する必要がありますか？

- プラットフォーム管理者は、承認された画像セットをキュレーションします。
- DevOps チームは、特定の環境 Dev、Test、Prod 用のイメージをタグ付けします。
- AI ファクトリーでモデルのライフサイクルを管理する AI チーム。
- 所有権メタデータを適用するセキュリティとコンプライアンスの所有者。

これによりできること

- 説明的なタグをリポジトリとイメージタグに適用します。
- チームガイダンスのためのリポジトリの README を編集します。
- イメージの所有権とプロモーション状態を追跡します。

- Image ライブラリーコンソールでエンドユーザーに明確なコンテキストを提供します。

完了までの推定時間

リポジトリ/イメージごとに 5～10 分。

前提条件

- ハイブリッドマネージャーまたは AI ファクトリーのイメージおよびモデルライブラリーにアクセスできます。
- リポジトリ統合は既に構成され、イメージの同期が完了しました。
- メタデータを編集するための適切な権限があること。

サポートされているメタデータ機能

Metadata Type	Applies To
HM Tags	Repository, Image Tag
README	Repository

リポジトリのメタデータの管理

HM タグをリポジトリに適用する

1. Image and Model ライブラリーに移動します。
2. 目的のリポジトリ例 `postgresql`、`meta/llama-3.3`、`your-private-model-repo`などをクリックします。
3. **Manage Tags** または **Edit Tags** をクリックします。
4. HM タグを追加します。
 - 例 `approved-production`、`beta`、`team-ai`、`archived`
5. 変更を保存します。

リポジトリの README を編集する

1. [リポジトリ] ページで、[**Edit README**] をクリックします。
2. README コンテンツを入力または更新します Markdown をサポート
 - セクションの例
 - リポジトリの目的。
 - 含まれるモデルまたは DB イメージタイプ。
 - サポート連絡先。
 - セキュリティに関する注意事項。

3. README を保存します。リポジトリの詳細ページに表示されます。

イメージタグのメタデータの管理

HM タグを画像タグに適用する

1. リポジトリに移動します。
2. **Image Tags** タブをクリックします。
3. 目的のイメージタグで、**Manage Tags** または **Edit Tags** をクリックします。
4. HM タグを追加します。
 - 例 approved-production、testing、deprecated
5. 変更を保存します。

一般的なタグの使用パターン

Tag Example	Usage
<i>approved-production</i>	プロダクション使用が承認された画像。
<i>beta</i>	評価中の画像。
<i>team-ai</i>	AI チームが使用するモデル。
<i>archived</i>	画像は現在使用されていません。
<i>test-only</i>	テスト/PoC 展開に使用されるイメージ。

ベストプラクティス

- 常に完全な検証の後にのみ **approved-production** を適用します。
- 非推奨またはアーカイブされた画像を明確にタグ付けします。
- チームベースのタグ **team-ml**、**team-dba** を使用して、所有権を示します。
- すべてのカスタムリポジトリに README を追加します - 透明性が向上します。
- 正確なバージョン追跡にはイメージタグ+HM タグを使用します例 **approved-production** でタグ付けされた **postgresql:17.4-xxxx**。

制限と注意事項

- タグ/README の適用は、イメージプルまたはランタイム動作に影響を与えません。これらは UI とガバナンスのためです。
- タグは、組織で確立された命名規則に従う必要があります。
- 現在、HM タグはプライベートレジストリに伝播しません。それらは HM メタデータに保存されます。

ワークフローの例

シナリオ AI チームは、新しいリランカーモデルを実稼働環境に移行する準備をしています。

1. チームはモデルイメージ `nv-rerankqa-1b-v2:1.8.2` を `private-model-repo` に同期します。
2. テスト検証後
 - この画像タグに `approved-production` タグを適用します。
 - リポジトリとイメージタグに `team-ai` を適用します。
 - リポジトリの README を編集して、モデルバージョンの履歴と所有権を含めます。
3. AI Factory のユーザーは `approved-production Reranker` モデルを簡単に識別できるようになりました。

概要

- リポジトリとイメージタグの HM タグと README を管理できます。
- これにより、ガバナンス、追跡、および透明性が向上します。
- プロダクションが承認した画像には、常に明確にタグ付けします。
- チームのガイダンスとして README を最新の状態に保ちます。

関連リンク

- [How-To Deploy AI Models from Model Library](#)
- [How-To Integrate Private Registry with Model Library](#)
- [How-To Define Repository Rules for Model Library](#)
- [Model Library Explained In Depth](#)
- [Model Library Explained In Depth](#)

How-To Setup GPU resources

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドを使用して、Kubernetes クラスターハイブリッドマネージャーまたは互換性の 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。

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クラスターを準備します。

推定時間

20～40 分プロビジョニングはクラウドプロバイダーによって異なります。

あなたが達成すること

- クラスター内の GPU ノードグループ/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。

前提条件

- 適切な権限による 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。
- `kubectl` に関する知識。

GPU ノードのプロビジョニング

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 た例えば、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 大規模なモデルの場合、4 つの GPU を備えた少なくとも 1 つのノード。

GPU ノードのラベル付けとテイント

```
kubectl label node <gpu-node-name> nvidia.com/gpu=true
kubectl taint nodes <gpu-node-name> nvidia.com/gpu=true:NoSchedule
```

NVIDIA デバイスプラグインを展開する

```
kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.14.1/
↪nvidia-device-plugin.yml
kubectl get ds -n kube-system nvidia-device-plugin-daemonset
```

NVIDIA API キーを Kubernetes シークレットとして保存する

```
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 によって使用されます。

関連概念

- *Model Serving Reference Manual*

次のステップ

- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com container with KServe*
- *How-To Verify InferenceServices and GPU usage*

How-To Configure a ClusterServingRuntime

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、KServe で **ClusterServingRuntime** を構成する方法について説明します。ClusterServingRuntime は、AI モデルを提供するために使用される環境を定義します。コンテナイメージ、リソース設定、環境変数、およびサポートされているモデル形式を指定します。

Hybrid Manager ユーザーにとって、ランタイムの構成は、モデル提供を有効にするための中心的手順です。 *Pipelines models reference* を参照してください。

ゴール

ClusterServingRuntime を構成して、InferenceServices がモデルを展開するために使用できるようにします。

推定時間

5～10 分。

あなたが達成すること

- ClusterServingRuntime YAML マニフェストを定義します。
- Kubernetes クラスターに適用します。
- 1 つ以上のモデルの再利用可能なサービス構成を有効にします。

これによりアンロックされるもの

- 標準のランタイム定義を使用して、モデルの一貫した展開をサポートします。
- 提供する画像とリソースプロファイルを集中制御できます。
- KServe を使用して NVIDIA NIM コンテナを展開するために必要な手順。

前提条件

- KServe がインストールされた Kubernetes クラスター。
- 目的のモデルサーバーイメージを使用してコンテナイメージレジストリにアクセスします。
- NVIDIA GPU ノードプール構成済み GPU ベースのモデルを使用する場合。
- API キー build.nvidia.com などに構成された Kubernetes シークレット。

背景の概念については、以下を参照してください。

- *Model Serving Concepts*
- *AI Factory terminology*

ステップ

1. クラスターサービングランタイム YAML を作成する

ClusterServingRuntime.yaml という名前のファイルを作成します。

例

```
apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-llama-3.1-8b-instruct-1.3.3
  namespace: default
spec:
  containers:
    - env:
      - name: NIM_CACHE_PATH
        value: /tmp
      - name: NGC_API_KEY
        valueFrom:
          secretKeyRef:
            name: nvidia-nim-secrets
            key: NGC_API_KEY
      image: upmdev.azurecr.io/nim/meta/llama-3.1-8b-instruct:1.3.3
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
          cpu: "12"
          memory: 64Gi
        requests:
          cpu: "12"
          memory: 64Gi
        volumeMounts:
          - mountPath: /dev/shm
            name: dshm
      imagePullSecrets:
        - name: edb-cred
      protocolVersions:
        - v2
```

(次のページに続く)

(前のページからの続き)

```

- grpc-v2
supportedModelFormats:
- autoSelect: true
  name: nvidia-nim-llama-3.1-8b-instruct
  priority: 1
  version: "1.3.3"
volumes:
- emptyDir:
  medium: Memory
  sizeLimit: 16Gi
  name: dshm

```

キーフィールドの説明

- `containers.image` モデルサーバーコンテナたとえば、NVIDIA NIM イメージ。
- `resources CPU`、メモリ、および GPU の要件。
- `NGC_API_KEY` NVIDIA モデルのシークレットリファレンス。
- `supportedModelFormats` このランタイムを参照するために `InferenceService` が使用する論理名。

2.ClusterServingRuntime を適用する

実行します

```
kubectl apply -f ClusterServingRuntime.yaml
```

3. 展開された ClusterServingRuntime の確認

実行します

```

kubectl get ClusterServingRuntime
__OUTPUT__
NAME                                     AGE
nvidia-nim-llama-3.1-8b-instruct-1.3.3 1m

```

次の方法で完全な詳細を検査できます。

```
kubectl get ClusterServingRuntime <name> -o yaml
```

4.InferenceService のリファレンスランタイム

`InferenceService` を作成するときに、このランタイムを参照します。

```

runtime: nvidia-nim-llama-3.1-8b-instruct-1.3.3
modelFormat:
name: nvidia-nim-llama-3.1-8b-instruct

```

How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe を参照してください。

注

- ランタイムは再利用可能です。同じ ClusterServingRuntime を参照する複数のモデルを展開できます。
- トレーサビリティのために、`supportedModelFormats` の意味のある名前とバージョンフィールドを使用します。
- YAML を編集および再適用することにより、ランタイムを更新できます。

次のステップ

- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe*
- *How-To Update GPU Resources for an InferenceService*
- *How-To Monitor deployed models with KServe*

関連読書

- *Model Serving Concepts*
- *AI Factory terminology*

How to deploy model

How-To Deploy AI Models from Model Library

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、AI Factory **モデルライブラリー* からハイブリッドマネージャー HM 環境の Model Serving KServe を利用して AI モデルを展開する方法について説明します。

これらのモデルは、展開されると、主要な AI Factory 機能を強化します。

- ナレッジベース AIDB パイプライン経由
- Gen AI Builder アシスタントとパイプライン
- その他の AI ファクトリーとアプリケーションの統合

このガイドは誰が使用する必要がありますか？

- 検証されたモデルイメージを展開する AI プラットフォーム管理者
- ナレッジベースの AI モデルを構成するデータエンジニア
- アシスタント用のモデルを構成する AI アプリケーション開発者

これによりできること

導入が完了すると

- AI モデルは Model Serving で利用できます。
- ナレッジベースまたは Gen AI Builder パイプラインにリンクできます。
- HM Model Serving UI または Kubernetes を介して展開されたモデルを監視および管理できます。

完了までの推定時間

モデルのサイズとクラスターリソースによって異なります。モデルごとに 10～20 分。

前提条件

始める前に

- GPU ワーカーノード が構成されたアクティブな HM 環境。
- build.nvidia.com Kubernetes に `nvidia-nim-secrets` として保存される API キー。
- デプロイするモデルイメージは、 **モデルライブラリー** に表示される必要があります。
- KServe はクラスターで構成され、準備ができていない必要があります。

→ 完全なセットアップガイドについては、次を参照してください。 [How-To Setup GPU resources](#)

AI モデルを展開する手順

1. モデルライブラリーでモデルを参照して選択します

- **AI Factory > Model ライブラリー** に移動します。
- 利用可能なモデル画像を参照します。
- バージョンとタグを確認します。
- 導入するバージョンを選択します。

2. モデルの構成と展開

- **Deploy** または **Deploy to Model Serving** をクリックします。
- デプロイメントパラメーターの構成
 - レプリカの数
 - リソース要求/制限 GPU、CPU、メモリ
 - モデルランタイム設定必要な場合
- 確認して展開します。

これにより、次の作成がトリガーされます。

- モデルの `ClusterServingRuntime` まだ定義されていない場合。

- 特定の展開の InferenceService。

3. 展開されたモデルの検証

次を使用して、展開したモデルを検証できます。

HM のモデル提供 UI

- AI Factory > Model Serving または Hybrid Manager > AI Factory > Model Serving に移動します。
- モデルのステータスが **Ready** と表示されることを確認します。

または、kubectl を使用します。

```
kubectl get InferenceService -n default
__OUTPUT__
NAME STATUS URL GPUs
meta-nim-llama-3-3-nemotron-super-49b Ready http://meta-nim-llama-3-3-nemotron-super-
↪49b-predictor... 4
nim-snowflake-arctic-embed-l Ready http://nim-snowflake-arctic-embed-l-predictor... 1
...
```

4. モデルを AI Factory ワークロードに接続します

モデルの準備ができれば

- 次の場所で選択できます
 - ナレッジベースパイプライン埋め込みまたはリランキング用
 - Gen AI Builder パイプライン
 - アシスタント構成

→ UI には、**タイプ**エンベディング、補完、リランキングなどに基づいて、各ユースケースで使用可能なモデルが表示されます。

サポートされているモデルタイプ

Model Type	Example Image
Text Completion	llama-3.3-nemotron-super-49b
Text Embedding	arctic-embed-l
Image Embedding	nvclip
OCR	paddleocr
Text Reranker	llama-3.2-nv-rerankqa-1b-v2

ヒントとベストプラクティス

- **GPU 配置** モデルが GPU 能力と一致することを確認します。llama-3.3-49b のような大規模なモデルには、単一ノードにマルチプルの GPU が必要です。
- **クォータ管理** 同時に展開される大規模なモデルの数を制限して、GPU ノードの過負荷を回避します。
- **バージョンテスト** 運用パイプラインまたはアシスタントに昇格する前に、分離された展開で新しいモデルバージョンをテストします。

障害対応

モデルが保留中になった

- GPU ノードのテイント/ラベルを確認します。
- InferenceService の許容範囲と nodeSelector が一致することを確認します。

モデルがモデルライブラリーに表示されない

- 画像が正しくタグ付けされ、画像とモデルライブラリーを介して同期されていることを確認します。
- プライベートレジストリを使用する場合、リポジトリルールを確認します。

デプロイ時の Kubernetes エラー

- 詳細なエラーログについては、`kubectl describe InferenceService <model>` を確認してください。

概要

- AI Factory Model ライブラリーから AI モデルを展開できます。
- 展開されたモデルは、KServe Model Serving を介して実行されます。
- 導入されたモデルは、ナレッジベースと Gen AI Builder Assistant を強化します。
- 展開フローにより、一貫したガバナンスと可視性が保証されます。

関連リンク

- [Model Library Explained In Depth](#)
- [Model Library Explained In Depth](#)
- [How-To Setup GPU resources](#)
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com/container with KServe](#)

How-To Create an InferenceService for NVIDIA NIM Container

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このハウツーでは、Kubernetes で KServe を使用して NVIDIA NIM コンテナを推論サービスとして展開する方法について説明します。

既存の ClusterServingRuntime を参照し、GPU アクセラレーションモデルインスタンスを実行する InferenceService.yaml マニフェストを定義して適用します。

これは、ネットワークにアクセス可能なエンドポイントを介して推論にモデルを利用できるようにするための中心的な手順です。

ゴール

KServe を使用して、NVIDIA NIM コンテナの InferenceService を展開します。

推定時間

10～15 分。

あなたが達成すること

- InferenceService マニフェストを定義します。
- InferenceService を Kubernetes クラスターに展開します。
- ネットワークエンドポイントを介してモデルを利用できるようにします。

これによりアンロックされるもの

展開されると、モデルは標準の推論 API を介してアプリケーション Gen AI Builder エージェントなどで使用できます。また、KServe モニタリングにも表示され、動的にスケーリングまたは更新できます。

前提条件

- KServe がインストールおよび構成された Kubernetes クラスター。
- NIM コンテナ用に構成された ClusterServingRuntime。 [How-To Configure a ClusterServingRuntime](#) を参照してください。
- GPU 対応ノードプールが適切にプロビジョニングされ、ラベル付けされます。
- build.nvidia.com 必要に応じて、Kubernetes シークレットとして保存される API キー。
- レジストリで使用可能な目的の NIM コンテナイメージ。

ステップ

1. InferenceService マニフェストを作成する

次の内容を使用して InferenceService.yaml という名前のファイルを作成します。

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: llama-3-1-8b-instruct-1xgpu-g5
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
  model:
    modelFormat:
      name: nvidia-nim-llama-3.1-8b-instruct
    resources:
      limits:
        nvidia.com/gpu: "1"
      requests:
        nvidia.com/gpu: "1"
    runtime: nvidia-nim-llama-3.1-8b-instruct-1.3.3
```

キーポイント

- **name** → InferenceService に一意の名前を指定します。
- **tolerations** および **nodeSelector** → GPU ノードプール構成と一致する必要があります。
- **runtime** および **modelFormat** → ClusterServingRuntime で使用される値と一致する必要があります。
- **GPU リソース** → 割り当てる GPU の数を指定します。

2.InferenceService を適用する

実行します

```
kubectl apply -f InferenceService.yaml
```

3. 展開の検証

InferenceService のステータスを確認します。

```
kubectl get InferenceService -o custom-columns=NAME:.metadata.name,MODEL:.spec.  
↪predictor.model.modelFormat.name,URL:.status.address.url,RUNTIME:.spec.predictor.  
↪model.runtime,GPU:.spec.predictor.model.resources.limits.nvidia\.com/gpu --  
↪namespace=default
```

次のものが表示されます。

- InferenceService の名前。
- 提供されるモデルフォーマット。
- エンドポイント URL。
- 使用される ClusterServingRuntime。
- 割り当てられた GPU の数。

次に来ること

モデルが展開されました。

- アプリケーションまたは Gen AI Builder エージェントは、指定された URL に推論要求を送信できます。
- モデルのパフォーマンスとリソースの使用状況を監視できます。
- 必要に応じて、GPU 割り当てまたはその他のランタイムパラメーターを更新できます。

関連トピック

- [How-To Configure a ClusterServingRuntime](#)
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com/container with KServe](#)
- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- [AI Factory terminology](#)

次のステップ

- [How-To Update GPU Resources for an InferenceService](#)
- [How-To Monitor deployed models with KServe](#)
- [How-To Create an InferenceService for NVIDIA NIM Container](#)

AI Factory learning guide で詳細を確認します。

How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com container with KServe

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、Kubernetes クラスターで KServe を使用して、build.nvidia.com コンテナで利用可能な NVIDIA NIM マイクロサービスを展開する方法について説明します。

ここでの手順は一般的な Kubernetes 環境に適用されますが、Hybrid Manager AI Factory では、ライフサイクル管理、可観測性、簡素化された統合などの追加価値を提供します。詳細については、*Model Serving in Hybrid Manager* セクションをご覧ください。

ゴール

KServe を使用して NVIDIA NIM コンテナを展開して、アプリケーションが使用できるネットワークアクセス可能な推論サービスを作成します。

推定時間

クラスターのセットアップによっては、15～30 分。

あなたが達成すること

- NVIDIA NIM コンテナの ClusterServingRuntime を定義および展開します。
- このランタイムを使用する InferenceService を展開します。
- 展開を検証し、モデルエンドポイントを取得します。

これによりアンロックされるもの

- 標準の推論プロトコル HTTP / gRPC を介して NVIDIA NIM を提供する機能。
- これらのモデルをグリップテープ Gen AI Builder または AIDB ナレッジベースなどのアプリケーションまたはツールと統合する準備をします。
- Hybrid Manager AI Factory のモデル提供機能を使用するための基礎。

前提条件

- KServe がインストールされた Kubernetes クラスター。
- GPU ノードプール構成済み NVIDIA デバイスプラグインを使用。
- プライベートレジストリまたは NGC で使用可能な NVIDIA NIM コンテナイメージ。
- NGC API キーを含む Kubernetes シークレット。
- クラスター用に構成された kubectl。

背景の概念については、以下を参照してください。

- *Model Serving Concepts*
- *AI Factory terminology*

ステップ

1. クラスター **ServingRuntime** を作成する

ClusterServingRuntime.yaml を定義します。

```
apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-llama-3.1-8b-instruct-1.3.3
  namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:
              name: nvidia-nim-secrets
              key: NGC_API_KEY
      image: your-registry/nim/meta/llama-3.1-8b-instruct:1.3.3
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
          cpu: "12"
          memory: 64Gi
        requests:
          cpu: "12"
          memory: 64Gi
        volumeMounts:
          - mountPath: /dev/shm
            name: dshm
  imagePullSecrets:
    - name: edb-cred
  protocolVersions:
    - v2
    - grpc-v2
  supportedModelFormats:
    - autoSelect: true
      name: nvidia-nim-llama-3.1-8b-instruct
      priority: 1
      version: "1.3.3"
  volumes:
```

(次のページに続く)

(前のページからの続き)

```
- emptyDir:
  medium: Memory
  sizeLimit: 16Gi
  name: dshm
```

ランタイムを適用します。

```
kubectl apply -f ClusterServingRuntime.yaml
```

2. InferenceService を作成する

InferenceService.yaml を定義します。

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: llama-3-1-8b-instruct-1xgpu
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
    model:
      modelFormat:
        name: nvidia-nim-llama-3.1-8b-instruct
      resources:
        limits:
          nvidia.com/gpu: "1"
        requests:
          nvidia.com/gpu: "1"
      runtime: nvidia-nim-llama-3.1-8b-instruct-1.3.3
```

InferenceService を展開します。

```
kubectl apply -f InferenceService.yaml
```

3. 展開されたモデルの検証

アクティブな推論サービスをリストします。

```
kubectl get InferenceService \
- o custom-columns=NAME:.metadata.name,MODEL:.spec.predictor.model.modelFormat.name,
↪URL:.status.address.url,RUNTIME:.spec.predictor.model.runtime,GPU:.spec.predictor.
↪model.resources.limits.nvidia\.com/gpu \
- -namespace=default
__OUTPUT__
```

NAME	MODEL	URL
		RUNTIME
	GPUs	
llama-3-1-8b-instruct-1xgpu	nvidia-nim-llama-3.1-8b-instruct	http://llama-3-1-8b-
↪instruct-1xgpu.default.svc.cluster.local	nvidia-nim-llama-3.1-8b-	
↪instruct-1.3.3	1	

4. ランタイムの詳細を取得する

ポートとリソースを確認します。

```
kubectl get ClusterServingRuntimes \
- o custom-columns=NAME:.metadata.name,IMAGE:.spec.containers[0].image,PORT:.spec.
↪containers[0].ports[0].containerPort,CPU:.spec.containers[0].resources.limits.cpu,
↪MEMORY:.spec.containers[0].resources.limits.memory \
- -namespace=all-namespaces
```

次のステップ

- [How-To Monitor deployed models with KServe](#)
- [How-To Update GPU Resources for an InferenceService](#)

関連読書

- [Model Serving Concepts](#)
- [AI Factory terminology](#)
- [Model Serving in Hybrid Manager](#)

How to integrate with your app

Quickstart — Use an Inference Endpoint with Python

このクイックスタートを使用して、プライベート推論サービス KServe の背後に展開されたモデルとチャットします。エンドポイント URL とアクセスキーに加えて、モデル名を渡します。この例では、実稼働対応の単一ファイルの Python スクリプトを使用します。

完了までの時間 5～10 分

ゴール

- HTTP 経由でプライベートの管理対象モデルエンドポイント OpenAI 互換を呼び出す
- チャット完了に必要なヘッダーとペイロードの形状を理解する
- アプリおよび CI / CD で再利用できる最小限のデモスクリプトを実行します

前提条件

- InferenceService が展開され、準備が整います。参照
- *Model Serving Reference Manual*
- *How-To Create an InferenceService for NVIDIA NIM Container*
- 外部アクセスパスとアクセスキー、または内部クラスターローカルパス。 - *Accessing KServe endpoints (internal and external)* を参照してください。
- Python 3.9+および httpx インストール済み `pip install httpx`

環境

スクリプトを実行する前に、これらの環境変数を設定します。

```
export EDB_API_URL="https://<portal>/inferenceservices/<inferenceservice-id>"
export EDB_API_TOKEN="<hm-user-access-key>"
export MODEL_NAME="meta/llama-3.1-8b-instruct" # or the model you serve
```

注

- 内部呼び出し者の場合、EDB_BASE_URL をクラスターローカルパスプロキシまたはダイレクト KServe URL に設定します。
- チャットのエンドポイントパスは、EDB_BASE_URL に `/v1/chat/completions` を追加します。

デモスクリプトを実行する

実稼働対応の単一ファイルをダウンロードして実行します。

- スクリプト `hm_kserve_quickstart.py`

例カスタムプロンプト

```
python hm_kserve_quickstart.py chat --prompt "Write a haiku about Postgres and GPUs."
```

オプションハイブリッドマネージャーを使用してクラスターをリストし、要約します。

```
pip install httpx typer
export EDB_API_URL="https://<hm-host>"; export EDB_API_TOKEN="<hm-access-token>"; ↵
↵ export PROJECT_ID="<uuid>"
python hm_kserve_quickstart.py summarize-clusters --project-id "$PROJECT_ID"
```

要求とヘッダーリファレンス

- エンドポイント POST `${EDB_BASE_URL}/v1/chat/completions`
- ヘッダー `Authorization: Bearer ${EDB_API_KEY}` 、 `Accept: application/json` 、 `Content-Type: application/json`
- 本文簡素化

```
{
  "model": "${MODEL_NAME}",
  "messages": [
    {"role": "user", "content": "Hello"}
  ],
  "max_tokens": 256
}
```

エンベディングとリランクモデルは別のパスを使用します。

- 埋め込み `${EDB_BASE_URL}/v1/embeddings`
- リランキング `${EDB_BASE_URL}/v1/ranking`

HTTP 404 Not Found と表示された場合は、操作固有のパスを確認します。

ベストプラクティス

- 推論のソブリンを保つ呼び出し元がクラスター内で実行される場合、内部パスを優先します。
- アクセスキーを定期的にローテーションします。決してソース管理にコミットしないでください。
- TLS を適用し、クライアントからの下りを制限します。
- レイテンシーとエラー率を監視します。必要に応じてリソースをスケールしたり、同時実行性を調整したりできます。

障害対応

- 401/403 `EDB_API_KEY` と `InferenceService` へのユーザー権限を確認します。
- 404 `InferenceService` ID を確認し、サービスの準備ができていることを確認します。
- `Timeouts/5xx KServe` ポッドのステータス、正常性プローブ、およびログを確認します。エンドポイントパスを検証します。

次のステップ

- 同じヘッダーとペイロードを使用してアプリをビルドします。 [Quickstart — Use an Inference Endpoint with Python](#) を参照してください。
- RAG の Gen AI アシスタントおよびナレッジベースとの統合。 [Gen AI Builder](#) および [Pipelines and data preparation](#) を参照してください。

- 可観測性と SLO を追加します。 *Observability for Model Serving* および *How-To Update GPU Resources for an InferenceService* を参照してください。

1.3 の既知の問題

- リストに表示される一部の内部モデル URL は、オペレーションサフィックスを省略する場合がありますたとえば、埋め込みには `/v1/embeddings` が必要です。呼び出しが 404 を返した場合、適切なサフィックスを追加します。これは将来のリリースで対応される予定です。
- 外部アクセスには、適切なロールたとえば、Gen AI Builder ユーザーなどの有効なハイブリッドマネージャーユーザーアクセスキーが必要です。不正なキーまたは不十分な権限は、HTTP 401 を返します。

hm_kserve_quickstart.py

```
import os, sys, json, httpx

BASE=os.getenv("EDB_API_URL","").rstrip("/")
TOKEN=os.getenv("EDB_API_TOKEN","")
ISVC=os.getenv("INFERENCESERVICE_ID","")
MODEL=os.getenv("MODEL_NAME","nvidia/llama-3.3-nemotron-super-49b-v1")
OPENAPI_FILE=os.getenv("OPENAPI_FILE","")

HEADERS={"Authorization":f"Bearer {TOKEN}","Accept":"application/json","Content-Type":
↪"application/json"}

def load_docs():
    if not OPENAPI_FILE:
        print("warning: no OPENAPI_FILE provided; LLM has no API docs context")
        return "No API docs available."
    try:
        with open(OPENAPI_FILE,"r",encoding="utf-8") as f:
            spec=json.load(f)
            trimmed={"paths":spec.get("paths",{}),"tags":spec.get("tags",[])}
            return json.dumps(trimmed,ensure_ascii=False)[:6000]
    except Exception as e:
        print(f"warning: failed to read OPENAPI_FILE: {e}")
        return "No API docs available."

def chat(messages):
    body={"model":MODEL,"messages":messages,"max_tokens":4096}
    with httpx.Client(base_url=BASE,headers=HEADERS,timeout=60) as c:
        r=c.post(f"/inferenceservices/{ISVC}/v1/chat/completions",json=body)
        r.raise_for_status()
        return r.json()["choices"][0]["message"]["content"].strip()

def main():
    docs=load_docs()
```

(次のページに続く)

(前のページからの続き)

```

messages=[
    {"role": "system", "content": "You are an API helper for engineers."},
    {"role": "system", "content": "Here are the API docs:\n"+docs},
]
print("API helper chat. Commands: /reset, /exit")
try:
    while True:
        user=input("> ").strip()
        if not user: continue
        if user=="/exit": break
        if user=="/reset":
            messages=messages[:2]
            print("(reset to API docs)")
            continue
        messages.append({"role": "user", "content": user})
        try:
            reply=chat(messages)
        except Exception as e:
            print(f"[error] {e}")
            messages.pop()
            continue
        messages.append({"role": "assistant", "content": reply})
        print(reply)
    except KeyboardInterrupt:
        print()

if __name__=="__main__":
    main()

```

Accessing KServe endpoints (internal and external)

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これらの手順を使用して、KServe で展開されたモデルエンドポイントを呼び出します。

これを使用する理由と場合

KServe エンドポイントにアクセスすると、アプリケーションは、パブリック AI API に依存せずにプライベートの管理モデル推論を使用できます。セキュリティ体制とネットワークポロジに一致するアクセスパスを選択します。

- 内部アクセス 呼び出し元が同じ Kubernetes 環境たとえば、Gen AI Builder Assistant、Pipelines ジョブ内で実行される場合に使用します。このパスは、外部の露出を回避し、ゲートウェイホップをスキップするため、遅延が削減されます。
- 外部アクセス呼び出し元がクラスターの外部にある場合に使用します顧客向けアプリ、パートナー

サービス、または共有エンタープライズ API など。アクセスキーを備えたポータルは、推論の主権を維持しながら、制御された境界を提供します。

KServe アクセスを使用する一般的な理由

- データ保護、コンプライアンス、コスト制御のために、環境にモデル推論を維持します。
- OpenAI 互換エンドポイントを提供して、ガバナンスを維持しながらクライアント統合を簡素化します。
- パブリック AI サービスが許可されていないエアギャップまたは制限されたネットワーク環境をサポートします。

前提条件

- デプロイされた InferenceService。 *How-To Create an InferenceService for NVIDIA NIM Container* を参照してください。
- 外部アクセスの場合ポータルエンドポイントとハイブリッドマネージャー環境からのユーザーアクセスキー。
- 内部アクセスの場合 Kubernetes クラスター内のサービスへのネットワークアクセス。

内部アクセスクラスター内

KServe のフロントにあるクラスター内アドレスクラスターローカル DNS を呼び出します。プレースホルダーを自分の値に置き換えます。

例プロキシサービスパススタイル

```
curl -X POST \
  http://upm-kservice-model-proxy.upm-ai-model-server.svc.cluster.local:80/
→inferenceservices/<inferenceservice-id>/<endpoint-path> \
-H accept: application/json \
-H Content-Type: application/json \
-d { ... }
```

例 KServe サービスの直接 URL、リソースステータスで使用可能な場合

```
status:
  address:
    url: http://<inferenceservice-name>-predictor.<namespace>.svc.cluster.local
```

HTTP クライアントでその URL を使用して、モデルを呼び出します。

外部アクセスクラスターの外部

ポータルエンドポイントを呼び出し、ハイブリッドマネージャーのユーザーアクセスキーをベアラートークンとして含めます。プレースホルダーを自分の値に置き換えます。

```
curl -X POST \
  https://<portal_domain>:<portal_port>/inferenceservices/<inferenceservice-id>/
  ↳<endpoint-path> \
  -H accept: application/json \
  -H Content-Type: application/json \
  -H "Authorization: Bearer a" \
  -d { ... }
```

例 OpenAI 互換のチャット補完

```
curl -X POST \
  https://portal-HM-pm.edbHM.com/inferenceservices/m-po7d0fta/v1/chat/completions \
  -H accept: application/json \
  -H Content-Type: application/json \
  -H "Authorization: Bearer a" \
  -d {
    "model": "meta/llama-3.1-8b-instruct",
    "messages": [{"role": "user", "content": "Write a limerick about the wonders of GPU
    ↳computing."}],
    "max_tokens": 64
  }
```

内部 URL と外部 URL — どこで見つけますか

クラスター内の呼び出し元には内部 URL を使用し、Kubernetes 外部の呼び出し元には外部ポータル URL を使用します。

- 内部 URL クラスターローカル
- SQL AIDB から `SELECT * FROM aidb.list_HM_models();` を実行して、データベースコンテキスト内の既知モデルの内部サービス URL を表示します。
- Kubernetes から名前空間の InferenceService リソースの `status.address.url` を確認しますたとえば `http://<name>-predictor.<ns>.svc.cluster.local`)。
- 注一部のタスクには、オペレーションサフィックスが必要でしたたとえば、埋め込みには `/v1/embeddings` が必要です。リストされた URL が 404 を返した場合、適切なサフィックスを追加します。
- 外部 URL ポータル - ハイブリッドマネージャー UI で AI Factory→Model Serving を開きます →InferenceService を選択します → サービスのポータルエンドポイントベースをコピーします。外部呼び出しは、HM ユーザーアクセスキーで `https://<portal>/inferenceservices/<id>/...` を使用します。

オペレーション別のエンドポイントパス

KServe は、操作ごとに異なるパスを公開します。モデル/タスクに一致するパスを使用します。

- チャット完了 LLM `<base>/v1/chat/completions`
- 埋め込み NIM 埋め込みモデル `<base>/v1/embeddings`

- Rerank モデルのランキング <base>/v1/ranking

モデルを呼び出すときに HTTP 404 Not Found が表示された場合は、オペレーションに正しいパスを使用していることを確認します。

オペレーション別のペイロード

タスクが異なれば、使用されるペイロード形状も異なります。以下の例は、一般的な OpenAI 互換スキーマを示しています。

- チャット完了 s
- エンドポイント <base>/v1/chat/completions
- ペイロード

```
{
  "model": "<model-name>",
  "messages": [
    {"role": "user", "content": "Hello"}
  ],
  "max_tokens": 256
}
```

- 埋め込み s
- エンドポイント <base>/v1/embeddings
- ペイロード

```
{
  "model": "<model-name>",
  "input": "I am the Yonk, I work in product"
}
```

- Rerank
- エンドポイント <base>/v1/ranking
- ペイロード 形状例。モデルのスキーマを確認してください

```
{
  "model": "<model-name>",
  "query": "search phrase",
  "documents": ["doc one", "doc two"],
  "top_n": 5
}
```

注

- AIDB SQL ヘルパーたとえば、`aidb.encode_text()` の場合、データベースは HTTP 呼び出しを構成します。内部 URL を使用してモデルを手動で構成する場合は、404 を回避するように正しい操作

パスたとえば/v1/embeddings を設定します。

- 一部のモデルは追加の操作パスを公開します。モデルリファレンスまたは InferenceService のドキュメントを参照してください。

セキュリティに関する考慮事項

- 外部に公開する必要がないサービスには、内部アクセスを優先します。
- アクセスキーを定期的にローテーションし、ユーザーごとにスコープを設定します。クライアントコードリポジトリにキーを埋め込まないでください。
- 外部呼び出しに TLS を適用します。プラットフォームチームが管理する証明書を使用します。
- 可能な場合、クライアントからの下りを制限します。オブザーバビリティを介して使用状況を監視します。

障害対応

- 404 - 正しいエンドポイントを使用していることを確認してください。(例、ID がモデル名に一致する内部/外部)
- 外部呼び出しの 401/403 Authorization ヘッダーと、ユーザーが InferenceService へのアクセスがあることを確認します。
- タイムアウトまたは 503 InferenceService ステータス、正常性プローブ、およびバックエンドログを確認します。エンドポイントパスが正しいことを確認します。
- 高レイテンシーリソースのサイズを適切なサイズにするか、同時実行性を調整します。[How-To Update GPU Resources for an InferenceService](#) を参照してください。

1.3 の既知の問題

- 外部アクセスには、十分なロール権限を持つ有効なハイブリッドマネージャーユーザーアクセスキーが必要ですたとえば、Gen AI Builder ユーザー。無効または不正なキーは HTTP 401 Unauthorized を結果ます。

関連

- [Model Serving Reference Manual](#)
- [How-To Create an InferenceService for NVIDIA NIM Container](#)
- [Observability for Model Serving](#)

How to operate and observe

How-To Verify InferenceServices and GPU usage

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドを使用して、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

5～10 分。

ステップ

InferenceService のステータスを確認する

```
kubectl get inferencservice -n <namespace>
```

READY を探して、サービスが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の確認

```
kubectl describe nodes | grep nvidia.com/gpu  
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

```
kubectl get ds -n kube-system nvidia-device-plugin-daemonset  
kubectl describe pods -n <namespace>
```

関連概念

- [How-To Setup GPU resources](#)
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe](#)
- [Model Serving Reference Manual](#)

How-To Monitor deployed models with KServe

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、Kubernetes で KServe を使用して展開された AI モデルを監視する方法について説明します。

モデルをモニタリングすることは、一般的な Kubernetes で作業しているか、ハイブリッドマネージャー AI ファクトリーを使用しているかにかかわらず、リソースの最適な使用を保証するのに役立ちます。

AI Factory ユーザーに、ハイブリッドマネージャーは追加の付加価値モニタリングおよびオブザーバビリティ機能を提供します。[Model Serving in Hybrid Manager](#) を参照してください。

ゴール

デプロイされたモデルを監視し、モデルのステータスとサービス提供エンドポイントを確認し、リソース使用状況情報を取得します。

推定時間

5～10 分。

あなたが達成すること

- デプロイされた推論サービスモデルをリストします。
- モデルエンドポイントとランタイムの詳細を取得します。
- モデルのパフォーマンスを観察する方法を理解します。
- モデルメトリックを可観測性パイプラインに統合する準備をします。

これによりアンロックされるもの

- モデルが正しく展開され、提供されているという自信。
- モデルの展開のトラブルシューティングまたはスケーリングの機能。
- モデル提供にハイブリッドマネージャー AI ファクトリーの可観測性を使用するための基礎。

前提条件

- KServe に InferenceService を展開しました。
- ClusterServingRuntime が定義されました。
- Kubernetes クラスター用に構成された kubectl。

背景の概念については、以下を参照してください。

- [Model Serving Concepts](#)
- [AI Factory terminology](#)

ステップ

1. デプロイされた推論サービスのリスト

デプロイされたモデルをリストし、重要な詳細を表示するには

```
kubectl get InferenceService \
- o custom-columns=NAME:.metadata.name,MODEL:.spec.predictor.model.modelFormat.name,
↪URL:.status.address.url,RUNTIME:.spec.predictor.model.runtime,GPU:.spec.predictor.
↪model.resources.limits.nvidia.com/gpu \
- -namespace=default
```

キー列

- **NAME** InferenceService の名前。
- **MODEL** モデルフォーマット名 ClusterServingRuntime から。
- **URL** 推論要求のサービスエンドポイント。
- **RUNTIME** 使用される ClusterServingRuntime。
- **GPUs** 割り当てられた GPU の数。

2. ランタイムの詳細を取得します

サービス提供ポートやリソースの割り当てを含むクラスターサービングランタイムの詳細を表示するには

```
kubectl get ClusterServingRuntimes \
- o custom-columns=NAME:.metadata.name,IMAGE:.spec.containers[0].image,PORT:.spec.
↪containers[0].ports[0].containerPort,CPU:.spec.containers[0].resources.limits.cpu,
↪MEMORY:.spec.containers[0].resources.limits.memory \
- --namespace=all-namespaces
```

キー列

- **NAME** ランタイムの名前。
- **IMAGE** 使用されるモデルサーバーイメージ。
- **PORT** 推論ポート通常 NIM の場合 8000。
- **CPU**s 割り当てられた CPU リソース。
- **MEMORY** 割り当てられたメモリ。

3. モデルメトリックを観察する

InferenceService アノテーションを介した Prometheus スクレイピングを有効にした場合

- Serving.kserve.io/enable-prometheus-scraping: “true”

次に、Prometheus は次の場所でメトリックをスクレイピングできます。

モデルサービスのポート 8000 の/v1/metrics。

メトリックには通常次のものが含まれます。

- 要求レイテンシー
- スループット 1 秒ごとの要求
- エラー率
- GPU 使用率 GPU を使用する場合

これらのメトリックは、**Grafana** などのツールで視覚化できます。

4. ポッドのステータスを確認する

デバッグの場合、モデルポッドを直接表示することもできます。

```
kubectl get pods --namespace=default
```

名前が一致するポッドを探します。

```
<inference-service-name>-predictor-*
```

必要に応じて、ポッドのステータスとログを確認します。

```
kubectl logs <pod-name> --namespace=default
```

次のステップ

- [How-To Update GPU Resources for an InferenceService](#) 近日公開予定
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe](#)
- [Model Serving in Hybrid Manager](#) 近日公開予定

関連読書

- [Model Serving Concepts](#)
- [AI Factory terminology](#)
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe](#)

How-To Update GPU Resources for an InferenceService

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このハウツーでは、KServe で展開された既存の推論サービスに割り当てられた GPU の数を調整する方法について説明します。これにより、スクラッチから再展開せずにモデルの展開を動的にスケーリングできます。

ゴール

デプロイされた InferenceService の GPU リソース割り当て GPU 数を変更します。

推定時間

5 分。

あなたが達成すること

- 既存の InferenceService リソースを編集します。
- 更新された GPU リソース制限と要求を適用します。
- KServe をトリガーして、新しい GPU 設定でモデルコンテナを再展開します。

これによりアンロックされるもの

GPU リソースの動的なスケーリングを有効にして、パフォーマンスとコストを最適化します。

- GPU を増やして、応答時間とスループットを向上させます。
- 需要が低い場合は GPU を減らしてリソースを節約します。

前提条件

- KServe を使用して既に展開されている InferenceService。 *How-To Create an InferenceService for NVIDIA NIM Container* を参照してください。
- Kubernetes クラスターで使用可能な GPU 対応ノードプール。

ステップ

1.InferenceService を編集する

実行します

```
kubectl edit InferenceService <your-inferenceservice-name>
```

例

```
kubectl edit InferenceService llama-3-1-8b-instruct-1xgpu-g5
```

2.GPU リソース設定の更新

このセクションを見つけてます。

```
spec:
  predictor:
    model:
      resources:
    limits:
      nvidia.com/gpu: "1"
    requests:
      nvidia.com/gpu: "1"
```

limits と **requests** の両方を目的の GPU の数に変更します。4 つの GPU にスケーリングする例

```
spec:
  predictor:
    model:
      resources:
        limits:
          nvidia.com/gpu: "4"
      requests:
        nvidia.com/gpu: "4"
```

保存してエディターを閉じます。

3. 更新された GPU 割り当ての確認

実行します

```
kubectl get InferenceService -o custom-columns=NAME:.metadata.name,MODEL:.spec.
↪predictor.model.modelFormat.name,URL:.status.address.url,RUNTIME:.spec.predictor.
↪model.runtime,GPU:.spec.predictor.model.resources.limits.nvidia.com/gpu --
↪namespace=default
```

GPUs 列が更新した設定を反映していることを確認します。

KServe は、新しい構成で InferenceService を自動的に再展開します。

関連トピック

- [How-To Create an InferenceService for NVIDIA NIM Container](#)
- [How-To Configure a ClusterServingRuntime](#)
- [How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe \(ブレースホルダー\)](#)
- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- [AI Factory terminology](#)

次のステップ

- [How-To Monitor deployed models with KServe](#)
- [How-To Update GPU Resources for an InferenceService](#)
- [How-To Update GPU Resources for an InferenceService](#)

Analytics & AI Factory learning guide で詳細を確認します。

How-To Use NVIDIA NIM Model Cache in Air - Gapped Clusters in Hybrid Manager

概要

モデルキャッシュを使用して、インターネットアクセスのない環境で NVIDIA NIM モデルを実行します。キャッシュには、NIM コンテナが実行時に NVIDIA NGC からダウンロードするモデルのプロファイルファイルが含まれています。

このガイドでは、すべてのキャッシュタイプの単一のフローについて説明します。バックエンドまたはオブジェクトストアごとに個別のページは必要ありません。

- フェーズ 1 モデルキャッシュをビルドする 接続環境
- フェーズ 2 キャッシュをオブジェクトストレージ AWS または GCP にアップロードする
- フェーズ 3 エアギャップクラスターでキャッシュを使用する UI のみ

前提条件

- GPU ノードを備えた HM クラスター。ノードに `nvidia.com/gpu=true` のラベルを付け、`nvidia.com/gpu` でテイントします。
- NVIDIA NGC API キー。
- エアギャップクラスターからアクセスできるプライベートレジストリ。

フェーズ 1—モデルキャッシュをビルドする 接続されている

接続された環境では、必要な NIM イメージをプライベートレジストリにコピーし、互換性のあるプロファイルを検出して、ローカルキャッシュディレクトリに事前ダウンロードします。

「モデルキャッシュ」とは何ですか？

- モデルキャッシュは、実行時に NVIDIA NIM イメージが必要とする「プロファイル」ファイルのセットです。NIM は、ハードウェアとバックエンドに基づいてプロファイルを選択し、通常は NGC からダウンロードします。
- NIM モデルのみがこの方法を使用できます。他の KServe サポートモデルは、NIM プロファイルキャッシュを利用できません。

1) NIM イメージをプライベートレジストリにコピーします

skopeo を使用して、NVIDIA NGC からレジストリにイメージをコピーします。

```
NGC_API_KEY=<your NGC API key>
REGISTRY=<your registry, e.g. registry.example.com>
USER=<your registry username>
PASSWORD=<your registry password>

skopeo login -u $oauthtoken -p "${NGC_API_KEY}" nvcr.io
skopeo login -u "${USER}" -p "${PASSWORD}" "${REGISTRY}"
```

(次のページに続く)

(前のページからの続き)

```
# Example: text embeddings model

skopeo copy --override-os linux --multi-arch all \
  docker://nvcr.io/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest \
  "docker://${REGISTRY}/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest"
```

使用する予定の NIM イメージごとに繰り返します。HM の一般的なデフォルトには次のものが含まれます。

```
nvcr.io/nim/nvidia/llama-3.3-nemotron-super-49b-v1:1.8.5
nvcr.io/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest
nvcr.io/nim/nvidia/llama-3.2-nv-rerankqa-1b-v2:latest
nvcr.io/nim/nvidia/nvclip:latest
nvcr.io/nim/baidu/paddleocr:latest
```

2)(オプション)HM のデフォルトのモデル画像 URL を更新する

HM が接続されている場合は、デフォルトのモデルイメージの URL を更新してレジストリをポイントし、将来のクラスターがそこからプルするようにします。

```
access_key=<your HM API key>

# List models; note each model ID

curl -k -H "x-access-key:${access_key}" \
  -X GET "https://<your HM Portal URL>/api/v1/ai-models" | jq .

# Patch a models image URL

curl -k -H "x-access-key:${access_key}" \
  -X PATCH "https://<your HM Portal URL>/api/v1/ai-models/<model ID>" \
  -d {"imageUrl":"<your-registry>/nim/nvidia/llama-3.2-nemoretriever-300m-embed-
  ↪v1:latest"}
```

3) 互換性のあるプロファイルを検出する

プロファイルは、モデルプラスハードウェアバリエーションです。NIM コンテナを使用して、ターゲット環境のプロファイルをリストします。GPU ホスト上の Docker を使用してこれを直接行うことができます。

```
export NGC_API_KEY=<your NGC API key>
docker run -it --rm \
  --runtime=nvidia \
  --gpus all \
  --shm-size=16GB \
  -e NGC_API_KEY \
  <your-registry>/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest \
  list-model-profiles
```


GPU の「互換性とシステム」セクションのプロファイル ID を記録します。

代替 Kubernetes からプロファイルをリストするジョブ GPU ノードに SSH で接続できない場合は、印刷互換プロファイル ID への GPU アクセスを使用してクラスターで短期ジョブを実行します。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: nim-list-job
  namespace: default
spec:
  template:
    metadata:
      name: nim-list-pod
    spec:
      containers:
        - name: nim-list
          image: <your-registry>/nim/nvidia/llama-3.2-nemoretriever-300m-embed-
↪v1:latest
          args: ["list-model-profiles"]
          env:
            - name: NGC_API_KEY
              valueFrom:
                secretKeyRef:
                  name: nvidia-nim-secrets
                  key: NGC_API_KEY
          resources:
            limits:
              nvidia.com/gpu: "1"
            requests:
              nvidia.com/gpu: "1"
      tolerations:
        - key: nvidia.com/gpu
          operator: Exists
          effect: NoSchedule
      imagePullSecrets:
        - name: <your-pull-secret>
      restartPolicy: Never
      securityContext:
        fsGroup: 26
        runAsGroup: 26
        runAsNonRoot: true
        runAsUser: 26
        seccompProfile:
          type: RuntimeDefault
```

出力例 ID はモデルとハードウェアによって異なります。

MODEL PROFILES

- Compatible **with** system **and** runnable:
 - 4f904d571fe60ff24695b5ee2aa42da58cb460787a968f1e8a09f5a7e862728d (vllm-bf16-tp1-
→pp1)
- With LoRA support:
 - f749ba07aade1d9e1c36ca1b4d0b67949122bd825e8aa6a52909115888a34b95 (vllm-bf16-tp1-
→pp1-lora)
- Compilable to TRT-LLM using just-**in**-time compilation:
 - ac34857f8dcdbd174ad524974248f2faf271bd2a0355643b2cf1490d0fe7787c2 (tensorrt_llm-
→trtllm_buildable-bf16-tp1-pp1)
- With LoRA support:
 - 7b8458eb682edb0d2a48b4019b098ba0bfbcb4377aadeeaa11b346c63c7adf724 (tensorrt_llm-
→trtllm_buildable-bf16-tp1-pp1-lora)
- Incompatible **with** system:
 - 6c3f01dd2b2a56e3e83f70522e4195d3f2add70b28680082204bbb9d6150eb04 (tensorrt_llm-
→h100-fp8-tp2-pp1-latency)

注 HM は現在 UI での LoRA サポートを公開していませんが、手動展開の LoRA 対応プロファイルをキャッシュすることはできます。

4) ローカルにキャッシュをビルドする

互換性のあるプロファイルのみをダウンロードするか、ユニバーサルキャッシュを使用する場合はすべてのプロファイルをダウンロードします。

選択したプロファイルをダウンロードします。

```
export NGC_API_KEY=<your NGC API key>
export LOCAL_NIM_CACHE=./model-cache
mkdir -p "$LOCAL_NIM_CACHE" && chmod -R a+w "$LOCAL_NIM_CACHE"

docker run -v $LOCAL_NIM_CACHE:/opt/nim/.cache \
  -u $(id -u) \
  -e NGC_API_KEY \
  --rm \
  <your-registry>/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest \
  download-to-cache \
  --profiles \
    <profile_id_1> \
    <profile_id_2>
```

すべてのプロファイルをダウンロードします。

```
export NGC_API_KEY=<your NGC API key>
export LOCAL_NIM_CACHE=./model-cache
mkdir -p "$LOCAL_NIM_CACHE" && chmod -R a+w "$LOCAL_NIM_CACHE"

docker run -v $LOCAL_NIM_CACHE:/opt/nim/.cache \
  -u $(id -u) \
  -e NGC_API_KEY \
  --rm \
  <your-registry>/nim/nvidia/llama-3.2-nemoretriever-300m-embed-v1:latest \
  download-to-cache --all
```

この手順の後、`./model-cache` には、NIM コンテナがオフラインのときに使用するプロファイルが含まれています。フォルダーのレイアウトは次のとおりです。

```
model-cache/
├── nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1
│   ├── local_cache
│   └── ngc
│       └── hub
│           ├── models--nim--nvidia--llama-3.2-nemoretriever-300m-embed-v1
│           │   ├── blobs
│           │   │   ├── <hash-1>
│           │   │   ├── <hash-2>
│           │   │   └── ...
│           │   ├── refs
│           │   │   └── <ref-name>
│           │   └── snapshots
│           │       └── <snapshot-name>
│           └── tmp
```

フェーズ 2-キャッシュをオブジェクトストレージにアップロードする

ローカルキャッシュディレクトリをオブジェクトストアにアップロードします。`model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1` など、モデルごとに個別のプレフィックスを使用します。

AWS S3 の例

```
aws s3 cp -r ./model-cache \
  s3://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1
```

オプション AWS CLI プロファイルを指定します

```
aws s3 cp -r ./model-cache \
  s3://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1 \
  --profile <aws-profile>
```

GCP バケットの例

```
gcloud storage cp -r ./model-cache \
gs://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1
```

サポートされているオブジェクトストア URI 形式

ハイブリッドマネージャーは、モデルプロファイルパスの次の URI 形式をサポートしています。

```
- s3://<bucket>/<path>
- gs://<bucket>/<path>
- hdfs://<namenode-host>:<port>/<path>
- webhdfs://<namenode-host>:<port>/<path>
- https://<account>.blob.core.windows.net/<container>/<path> (Azure Blob Storage)
- https://<hostname>/<path> (generic HTTPS object stores)
```

フェーズ 3 — エアギャップクラスターでキャッシュを使用する

HM UI からモデルを作成し、オブジェクトストアのキャッシュパスをポイントします。Kubernetes YAML は必要ありません。これは KServe とは異なります。

HM UI のステップ

1. AI Factory → Models → Create Model に移動します
2. デフォルトモデル例 llama-3.2-nemoretriever-300m-embed-v1 を選択します
3. “Model Profiles Path on Object Storage” をフェーズ 2 でアップロードしたパスに設定します。例
 - s3://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1
 - gs://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1
4. コンピューティング GPU とサイズを選択します。次に、作成します

スクリーンショット

- スクリーンショットプレースホルダー モデルの作成 — モデル選択
- スクリーンショットプレースホルダー モデルの作成 — モデルプロファイルパス
- スクリーンショットプレースホルダー モデルの詳細 — 使用中のキャッシュ

注

- クラスターがプライベートレジストリからイメージをプルできることを確認します必要に応じて、HM またはクラスターでイメージプルのシークレットを構成します。
- 使用する予定のモデルごとにキャッシュをビルドしてアップロードします。
- HM でキャッシュを使用するために、シークレットを作成したり、YAML を適用したりする必要はありません。

キャッシュと展開を検証する

次の手順に従って、イメージコピー、キャッシュのアップロード、およびモデルがエアギャップ環境でキャッシュを使用していることを確認します。

1. プライベートレジストリのイメージを検証する

```
skopeo inspect docker://${REGISTRY}/nim/nvidia/llama-3.2-nemoretriever-300m-embed-
↪v1:latest | jq .Name, .Digest
```

2. アップロードする前にローカルキャッシュを検証する

```
du -sh ./model-cache
find ./model-cache -maxdepth 3 -type d | head -n 20
```

3. オブジェクトストアの検証 upload

- S3 `aws s3 ls s3://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1/`
- GCS `gcloud storage ls gs://<bucket-name>/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1/`
- HDFS `hdfs dfs -ls /model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1`
- WebHDFS `curl -sS "http://<namenode-host>:9870/webhdfs/v1/model-cache/nim-nvidia-llama-3.2-nemoretriever-300m-embed-v1?op=LISTSTATUS" | jq`

4. HM で検証する UI

- モデルカードはステータスを示します Ready/Healthy
 - 「オブジェクトストレージのモデルプロファイルパス」は S3/GS/HDFS/WebHDFS/HTTPS パスと一致します
5. モデルエンドポイントに対してスモークテストを実行するモデルの詳細ページから、エンドポイント URL と API キーをコピーします。次に、最小限のエンベディング要求を送信しますエンベディングモデルの例の場合。

```
ENDPOINT=<your model base URL>
API_KEY=<your model api key>

curl -sS -X POST "$ENDPOINT/v1/embeddings" \
```

(次のページに続く)

(前のページからの続き)

```
-H "Authorization: Bearer $API_KEY" \
-H "Content-Type: application/json" \
-d {"input":["hello from EDB"]} | jq .data[0].embedding | length
```

返された数値ベクトルの長さが表示されるはずです。

6. (オプション) 実行中のポッドのキャッシュ使用量を検査する

```
kubectl -n <model-namespace> get pods -l app.kubernetes.io/name=<model-name>
kubectl -n <model-namespace> logs <pod-name> --tail=200
kubectl -n <model-namespace> exec -it <pod-name> -- sh -lc ls -al /opt/nim/.cache &&
↪ du -sh /opt/nim/.cache
```

キャッシュが存在する場合、NGC からのダウンロードの試行はログに表示されません。

3.3 Gen AI Builder

3.4 Gen AI Builder

Gen AI Builder は、*retrieval - augmented generation* パターンを介して組織の知識を活用するインテリジェントアプリケーションを開発するための包括的なツールを提供します。プラットフォームは、制御されたインフラストラクチャ環境内で動作し、データ、モデル、推論操作に対する完全な組織の主権を保証します。

ここは新しいですか？ *Gen AI Builder Hub* で始めて、*Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)* および *Gen AI Architecture (Entities and Flows)* を確認してから、*Gen AI How-To Guides* を使用するか、*Customer Service Assistant Implementation Guide* にジャンプします。

3.4.1 プラットフォームアーキテクチャ

Gen AI Builder は、複数のコンポーネントを統合して、EDB Postgres AI エコシステム内でエンドツーエンドの AI アプリケーション開発機能を提供します。このプラットフォームは、エンタープライズレベルのガバナンスと監査可能性を提供しながら、組織の境界内ですべての操作を維持することにより、データ主権を強調します。*Assistants Reference Manual*、*Knowledge Bases Reference Manual*、*Retrievers Reference Manual*、*Rulesets Manual*、*Structures Reference Manual*、*Tools Reference Manual*、*Threads Reference Manual*、*Configure Data Sources in Gen AI Builder*、*Data Lake Reference Manual*、および *edb_16lk_asis_16lk_asis_13*、*Data Lake Reference Manual*、および *edb_16lk_asis_13* を参照します。

コアインフラストラクチャの統合

Hybrid Manager Foundation Gen AI Builder は、EDB's vision for AI with Postgres and Hybrid Manager の Kubernetes オーケストレーションレイヤー内で動作し、一元的なガバナンスを維持しながら分散コンピューティング機能を活用します。このアーキテクチャにより、開発環境と実稼働環境全体で一貫した操作手順が保証されます。

データエステートの統合 *Knowledge Bases Reference Manual* は、組織インフラストラクチャ内の Postgres データベースとオブジェクトストレージシステム *Data Lake Reference Manual* を参照してください。多様な *Configure Data Sources in Gen AI Builder* タイプをサポートしながら、外部データの依存関係を排除します。

モデルサービング統合 アプリケーションは、*AI Factory Models* インフラストラクチャを介して言語モデルと埋め込みモデルにアクセスし、すべての AI 機能にわたって一貫したガバナンスポリシーを保証します。

3.4.2 アーキテクチャコンポーネント

ナレッジベースフレームワーク

Knowledge Bases Reference Manual は、構造化データベースコンテンツと非構造化ドキュメントリポジトリを組み合わせることにより、検索拡張生成アプリケーションの基盤として機能します。フレームワークは、完全なデータリネージを維持しながら、ベクター埋め込みを介してセマンティック検索機能を提供します *Vector Engine (PGvector)* を参照してください。

データソースの統合

- セマンティックインデックスを使用した Postgres データベーステーブル
- オブジェクトストレージシステムからのドキュメントリポジトリ
- API 統合を介したリアルタイムデータフィード
- 制御されたスクレイピング操作を介した Web コンテンツ

エンベディング管理

- 機密コンテンツ処理のためのプライベート埋め込みモデル
- Postgres インフラストラクチャ内のベクターストレージの最適化
- 取得精度のための意味的類似性アルゴリズム
- ベクトルとキーワードのアプローチを組み合わせたハイブリッド検索

検索システム

Retrievers Reference Manual コンポーネントは、多様なデータソースにわたって検索操作をオーケストレーションし、ソースの帰属とアクセス制御を維持しながら、ナレッジ発見のための統一インターフェイスを提供します。

取得戦略

- セマンティックコンテンツマッチングのベクトル類似性検索
- キーワードベースの検索による正確な用語の特定
- 複数の検索方法を組み合わせたハイブリッドアプローチ
- ユーザー権限とデータ分類に基づいたコンテキスト依存フィルタリング

アシスタントフレームワーク

インテリジェントな *Assistants Reference Manual* は、ナレッジの取得と言語モデル機能を組み合わせて、組織のナレッジに会話型インターフェイスを提供します。このフレームワークは、すべてのインタラクションの監査証跡を維持しながら、複雑な推論ワークフローをサポートします *Threads Reference Manual* を参照してください。

アシスタント機能

- コンテキスト保存を備えたマルチターン会話インターフェイス
- 外部システム相互作用のための *Tools Reference Manual* 統合
- 複雑なクエリ解決のための推論ワークフロー
- コンプライアンスと運用の透明性のための監査ログ

3.4.3 実装パターン

エンタープライズナレッジシステム

組織は、セマンティック検索機能を介して構造化および非構造化情報ソースを統合する包括的なナレッジマネージメントシステムを実装します。

一般的なアーキテクチャ

1. **データ統合レイヤー** データベース、ドキュメントリポジトリ、ナレッジマネージメントシステムを含む複数の組織の *Configure Data Sources in Gen AI Builder* を接続します
2. **処理パイプライン** 適切なメタデータとアクセス制御注釈を使用してコンテンツを検索可能な形式に変換します
3. **取得インターフェイス** 関連性ランキングとソース帰属を使用して、すべての統合コンテンツの統合検索を提供します
4. **アプリケーションレイヤー** 会話型インターフェイスと API 統合を介してナレッジアクセスを提供します

実装に関する考慮事項

- コンテンツ分類とアクセス制御のためのデータガバナンスポリシー
- 大規模なドキュメントコレクションのパフォーマンスの最適化
- 既存の認証および承認システムとの統合
- 監査証跡とデータリネージのコンプライアンス要件

検索拡張生成アプリケーション

アプリケーションは、取得機能と言語モデルの推論を組み合わせて、組織の知識に基づいて状況認識的な応答を提供します。

開発ワークフロー

1. **ナレッジベースの構築** セマンティック埋め込みを使用した包括的なドキュメントインデックスを確立します *Create a Knowledge Base in Gen AI Builder* および *Manage Knowledge Bases in Gen AI Builder* を参照してください。
2. **取得構成** アプリケーション要件に最適化された検索戦略を実装します。
3. **生成統合** 取得したコンテキストに基づいて、応答生成のための言語モデルを接続します

- レスポンスオーケストレーション適切なキャッシュと最適化を使用して取得と生成ワークフローを調整します

技術要件

- インタラクティブアプリケーションの低レイテンシーの取得
- 大規模なドキュメント処理のためのコンテキストウィンドウ管理
- 応答品質評価と改善のワークフロー
- ナレッジベースの更新全体での統合テスト

会話型 AI プラットフォーム

インタラクティブアプリケーションは、複雑な組織ワークフローと情報検出のための自然言語インターフェイスを提供します。

プラットフォームコンポーネント

- テキスト、音声、ドキュメントのインタラクションのマルチモーダル入力処理
- 長時間の会話セッションのコンテキスト管理
- 外部システム操作のためのツール統合
- ユーザーのロールと設定に基づいた応答のパーソナライゼーション

操作機能

- 会話状態の永続性を使用したセッション管理
- 応答品質と遅延のパフォーマンスモニタリング
- アプリケーションの最適化とキャパシティプランニングのための使用分析
- 機密情報の取り扱いのセキュリティ制御

3.4.4 開発フレームワーク

ナレッジベース開発

ナレッジベースの作成には、データソースの統合、コンテンツ処理、および効率的な取得操作をサポートするセマンティックインデックス作成が含まれます。

開発プロセス

- ソース評価コンテンツの品質、更新頻度、およびアクセス要件について *Configure Data Sources in Gen AI Builder* を評価します
- 処理パイプラインデザインクリーニング、チャンク、メタデータ抽出を含むコンテンツ変換ワークフローを構成します *Configure the Data Lake in Gen AI Builder* を参照してください
- インデックス戦略エンベディング生成とベクタストレージの最適化を実装する
- 取得テスト 体系的な評価手順を介して検索品質を検証する

品質保証

- 体系的なサンプリングによるコンテンツの精度検証
- 標準メトリックを使用した検索の関連性評価
- 予想される負荷条件でのパフォーマンスベンチマーク
- アクセス制御とデータ分離のためのセキュリティテスト

アシスタント設定

アシスタントの開発では、取得、推論、および応答生成機能を注意深くオーケストレーションして、信頼性の高いユーザーエクスペリエンスを提供する必要があります [Create an Assistant in Gen AI Builder](#) を参照してください。

構成領域

- **動作定義** アシスタントの性格、機能、および操作境界を確立します
- **知識統合** 取得戦略と知識ソースの優先順位付けを構成します [Retrievers Reference Manual](#) を参照してください。
- **ツール統合** 外部システムと API を接続して機能を拡張します [Tools Reference Manual](#) を参照してください
- **応答の最適化** 適切な応答特性に合わせて生成パラメーターを調整します

テスト戦略

- 体系的なダイアログテストを介した会話フローの検証
- 多様なクエリタイプにわたる知識の精度検証
- 外部ツール相互作用の統合テスト
- 同時ユーザーシナリオの負荷テスト

API 開発

アプリケーションは、一貫したセキュリティと操作特性を維持しながら、多様なクライアントアプリケーションをサポートする標準化された API を介して機能を公開します [Where to find SDK references and examples?](#) を参照してください。

API デザイン原則

- 明確なリソースモデリングを使用した RESTful インターフェイス
- 組織システムとの認証・承認の統合
- リソース保護のためのレート制限とクォータ管理
- 包括的なドキュメントと SDK のサポート

3.4.5 統合アーキテクチャ

EDB Postgres AI エコシステム

Gen AI Builder は、他の EDB Postgres AI コンポーネントとシームレスに統合し、包括的な AI アプリケーション開発機能を提供します。

コンポーネントの相互作用

- **** AI Factory Models **** 言語モデルと埋め込みモデルへの管理されたアクセス
- **** AI Accelerator Pipelines **** 自動コンテンツ処理とナレッジベースの更新
- **** Vector Engine (PGvector) **** Postgres インフラストラクチャ内の高性能セマンティック検索
- **** EDB' s vision for AI with Postgres and Hybrid Manager **** オーケストレーションとリソース管理

外部システム統合

アプリケーションは、セキュリティ境界と監査機能を維持しながら、標準化されたインターフェイスを介して既存の組織システムに接続します。

統合パターン

- リアルタイムデータアクセスのための API ベースの接続
- 大規模なデータソース同期のバッチ処理
- 動的コンテンツ管理のイベント駆動型更新
- ユーザーアクセス制御のための安全な認証委任

3.4.6 運用上の考慮事項

パフォーマンス特性

Gen AI Builder アプリケーションでは、計算リソースを効率的に管理しながら、応答性の高いユーザーエクスペリエンスを提供するために、注意深いパフォーマンスの最適化が必要です。

パフォーマンス要素

- ナレッジベースのサイズは取得レイテンシーとリソース要件に影響を与える
- モデルの複雑さは、応答生成時間と GPU 使用率に影響を与える
- 同時ユーザーの負荷は、システムのスケーラビリティとリソース割り当てに影響を与えます
- ネットワーク接続は分散コンポーネント通信に影響を与える

セキュリティフレームワーク

包括的なセキュリティ制御は、組織の機密情報を保護しながら、許可されたユーザーとアプリケーションの適切なアクセスを有効にします。

セキュリティコンポーネント

- さまざまな組織単位のマルチテナントの分離
- 既存の承認システムと連携したロールベースのアクセス制御
- ナレッジベースコンテンツと通信チャネルのデータの暗号化
- すべてのナレッジアクセスとアシスタントのインタラクションの監査ログ

ガバナンス統合

ビルトインのガバナンス機能により、アプリケーションは運用ライフサイクル全体にわたって組織のポリシーと規制要件に準拠します。

ガバナンス機能

- ナレッジベース更新のコンテンツ承認ワークフロー
- モデルの使用状況の追跡とコンプライアンスレポート
- 監査とトラブルシューティングのためのデータリネージのメンテナンス
- サービスレベル目標に合わせたパフォーマンスモニタリング

3.4.7 はじめに

前提条件

Gen AI Builder には、ハイブリッドマネージャーのインストールと AI ワークロードの適切なコンピューティングリソースを含む基本的なインフラストラクチャコンポーネントが必要です。

インフラ要件

- 十分なコンピューティング容量とストレージ容量を備えた Kubernetes クラスター
- モデル提供とエンベディング生成のための GPU リソース
- 組織データソースへのネットワーク接続
- ID およびアクセス管理システムとの統合

初期実装

組織は、より広範な展開のための運用手順を確立しながら、価値を実証する焦点を絞ったユースケースから始める必要があります。

実装シーケンス

1. **パイロットナレッジベース** 明確に定義されたコンテンツスコープで初期ナレッジベースを作成します
2. **基本的な取得** パフォーマンス検証を備えた単純な検索機能を実装する
3. **アシスタントプロトタイプ** パイロットナレッジドメインの会話インターフェイスを開発する
4. **運用展開** 包括的な運用制御を使用して、成功したプロトタイプをスケールする

学習リソース

概念的な基礎

- [Gen AI Builder Hub](#) - プラットフォームの概要と初期セットアップ手順
- [Gen AI How-To Guides](#) - 一般的なシナリオの実践的な実装ガイダンス

コンポーネントドキュメント

- [Libraries Ω Manual](#) - 包括的なナレッジベース開発ガイダンス
- [Assistants Reference Manual](#) - インテリジェントアシスタントの作成と構成
- [API Reference](#) - 開発チームのための詳細な技術文書

高度なトピック

- [Data Lake Reference Manual](#) - 大規模なデータの処理と管理
- [SDK Documentation](#) - カスタム開発用のプログラマティックインターフェイス

Gen AI Builder を使用すると、組織は、既存のインフラストラクチャ環境内のデータ、モデル、推論操作の完全な制御を維持しながら、洗練された AI アプリケーションを開発できます。

Getting Started

Gen AI Builder Hub

Gen AI Builder は、**AI Factory** の開発者向け環境であり、信頼できるデータと管理されたインフラストラクチャで高度なエージェント AI アプリケーションを構築するのを助けるように設計されています。

Builder は、ナレッジベース、レトリバー、ルールセット、ストラクチャ、および ツールをきめ細かく制御できます。深く統合された AI 搭載アプリケーションを構成できます。あなたのビジネスコンテキストに合わせて。

Gen AI Builder は、**AI Factory** でソブリン AI エージェントとデータ駆動 AI ワークフローを構築するための基盤です。

コア機能

- ナレッジベース を管理し、RAG ベースのエクスペリエンスを強化する
- 最適化されたセマンティックおよび構造化検索のための **レトリバー**を作成および管理します
- **ルールセット**を定義して、アシスタントのトーン、動作、コンプライアンスを制御します
- ツールをビルドして展開して、API 呼び出し、計算などでエージェント機能を拡張します
- **Structures** のパッケージ化と実行 — 複雑な Griptape を利用したワークフローとデータパイプライン
- **スレッド** の監視と分析 — 会話履歴とシステム動作

ラーニングパス

Builder をステップバイステップで学習するには、ここから始めます。

- *Getting Started - AI Factory 101*
- *Building Production-Ready AI Features - AI Factory 201*
- *Advanced AI Factory Usage and Extensibility - AI Factory 301*

概念と説明

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Assistants Reference Manual*
- *Knowledge Bases Reference Manual*
- *Retrievers Reference Manual*
- *Rulesets Manual*
- *Tools Reference Manual*
- *Structures Reference Manual*
- *Threads Reference Manual*

ハウツーガイド

- *Create a Knowledge Base in Gen AI Builder*
- *Create a Retriever in Gen AI Builder*
- *Create a Ruleset in Gen AI Builder*
- *Create a Tool in Gen AI Builder*
- *Create a Structure in Gen AI Builder*
- *How-To View and Manage Threads in Gen AI Builder in Hybrid Manager*

リファレンス

- Gen AI Builder SDK reference

関連するハブページ

- Gen AI in Hybrid Manager
- *Model Serving Reference Manual*
- Vector Engine Hub
- *AI Factory Hub Map*

Gen AI Builder を使用する場合

次の場合に Gen AI Builder を使用します。

- アシスタントが取得するものとどのように動作するかを **きめ細かく制御**する必要があります。
- **構造化データと非構造化データ**をエージェントに統合したいと考えています。
- 単純なチャットボットデモではなく、**信頼できる、準拠した、実稼働グレードのエージェント**を構築する必要があります。
- 実世界のシステムに接続する **ツール**でエージェントを拡張したい。
- あなたは、Structures を使用して**マルチステップ AI パイプラインとワークフロー**を構築しています。

AI Factory Gen AI Builder は、完全にガバナンスの下で、完全な可視性を備えた、高度で拡張可能なエンタープライズ対応 AI アプリケーションを構築するツールを提供します。

Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)

このページでは、AI Factory で Gen AI アプリケーションをビルドするために使用するコアエンティティとそれらの関係を紹介します。ハウツーまたは SDK の詳細に入る前に説明します。

アシスタントオーケストレーター

アシスタントは、アプリケーションロジックのオーケストレーターです。

- ディレクティブゴールを実行し、座標の取得とツールの使用を実行します。
- ナレッジ RAG の場合、ツールを使用しますアクション、API の場合。
- プラットフォームによって構成されたモデルエンドポイントで実行されます *Model Serving Reference Manual* を参照してください。

詳細はこちら *Assistants Reference Manual*

ルールセットポリシーとコンストレイン

ルールセットは動作を制約し、出力をガイドします。

- キャプチャポリシー トーン、スタイル、安全ガイドライン、ガードレール。
- アシスタントに適用して、実行全体に境界を適用します。

詳細はこちら [Rulesets Manual](#)

ナレッジ 検索

ナレッジスタックは、エンタープライズデータに基づいてモデルを根拠とします。

- ナレッジベースフィルタのメタデータを含むテーブル/ファイルからの埋め込みコンテンツ。
- レトリバーハイブリッド検索の検索パラメーター top-K、しきい値、フィルターを定義します。
- データレイクナレッジの取り込みをフィードするアセットとファイルのリポジトリ。

詳細はこちら

- [Knowledge Bases Reference Manual](#)
- [Data Lake Reference Manual](#)

ツール アクションと統合

アシスタントはツールを使用して外部システムと内部 API を呼び出すことができます。

- REST エンドポイントまたは内部サービスを型指定されたインターフェイスでラップします。
- 認証と入出力マッピングを提供します。
- ルールセットと構成を介してアシスタントが呼び出すことができるツールを制御します。

詳細はこちら [Tools Reference Manual](#)

構造体ワークフローとビルディングブロック

構造は、以下を構成できる高レベルのビルディングブロックタスク、パイプライン、ワークフローです。

- 再利用可能なロジックパターンをカプセル化します。
- アシスタントと統合するか、内部でツールと取得を呼び出します。

詳細はこちら [Structures Reference Manual](#)

スレッド状態と可観測性

スレッドは会話状態を持続し、監査可能性を提供します。

- プロンプト、ツール呼び出し、取得したコンテキスト、および応答を記録します。
- レビュー、デバッグ、およびコンプライアンスを有効にします。

詳細はこちら [Threads Reference Manual](#)

モデルとエンドポイント世代

アシスタントは生成にモデルエンドポイントを使用します。

- プライベート推論には *Model Serving Reference Manual* を使用します。
- エンドポイントを内部クラスターローカルまたは外部ポータル+アクセスキーで呼び出します。
- オペレーション固有のパスを使用しますチャット /v1/chat/completions、埋め込み /v1/embeddings、リランク/v1/ranking。

詳細はこちら *Accessing KServe endpoints (internal and external)*

それをまとめる

高レベルでは

- 1) ナレッジハイブリッド検索を備えたナレッジベースとツールを準備します。2) ディレクティブでアシスタントを構成し、ナレッジとツールを接続し、モデルエンドポイントを選択します。3) スレッドを使用して、動作を観察、トレースをデバッグ、および相互作用を管理します。

ビルドガイド

- *Customer Service Assistant Implementation Guide*
- *Gen AI Builder Hub*

Gen AI Architecture (Entities and Flows)

このページでは、Gen AI エンティティが実行時にどのように連携するか、および構成とガバナンスが存在する場所について説明します。コンセプト、ハウツー、クイックスタートのコンパニオンとして使用します。

エンティティと関係

- アシスタント → 指示を満たすために取得とツールの使用を調整します。
- ルールセット → アシスタントの動作ポリシー、トーン、安全性を制限します。
- ナレッジ → ナレッジベース埋め込みコンテンツ+レトリバーハイブリッド検索+データレイク入力。
- ツール → 認証された型付けインターフェイスを介したアクションと統合を有効にします。
- 構造 → アシスタントによって呼び出される再利用可能なビルディングブロックタスク、パイプライン、ワークフロー。
- スレッド → キャプチャ状態と実行トレースプロンプト、取得、ツール呼び出し、出力。

エンティティドキュメント

- *Assistants Reference Manual*
- *Rulesets Manual*
- *Knowledge Bases Reference Manual*
- *Tools Reference Manual*

- *Structures Reference Manual*
- *Threads Reference Manual*

制御フローとデータフロー

User/App $\rightarrow$ Assistant $\rightarrow$ (Retriever $\rightarrow$ Knowledge Base) $\rightarrow$ Model endpoint
 $\quad \quad \quad \hookrightarrow$ (Tool call $\rightarrow$ API/service)
 $\quad \quad \quad \boxtimes$ Thread (state, trace)

フロー

1. ユーザーまたはアプリは、指示子/入力を使用してアシスタントを呼び出します。
2. アシスタントは、設定された取得装置とナレッジベースを介してコンテキストを取得します。
3. アシスタントはオプションでツールを呼び出して、データを収集するか、アクションを実行します。
4. アシスタントは、取得したコンテキストに基づいて、生成のためにモデルエンドポイントを呼び出します。
5. スレッドは、プロンプト、取得したチャンク、ツール I/O、および監査とデバッグのための応答をキャプチャします。

エンドポイントとモデル

- Sovereign AI のプライベート *Model Serving Reference Manual* エンドポイントを使用します。
- 内部呼び出しはクラスターローカル DNS を使用します。外部呼び出しはポータル+アクセスキーを使用します
- 操作パス チャット `/v1/chat/completions`、埋め込み `/v1/embeddings`、リランク `/v1/ranking`。
- クライアントパターン OpenAI 互換または生の HTTP。 *Accessing KServe endpoints (internal and external)* および *Quickstart — Use an Inference Endpoint with Python* を参照してください。

ガバナンスとオブザーバビリティ

- 入力管理ナレッジキュレーション、取得フィルタ、ツールスコープ、ルールセット。
- 可観測性スレッドとプラットフォームメトリック/ログを使用して、デバッグと監視をします。
- 監査可能性 スレッドは、決定とデータの使用の永続的な記録を提供します。

どこから始めるべきか

- *Customer Service Assistant Implementation Guide*
- *Gen AI Builder Hub*
- *Best practices for Hybrid Knowledge Bases*
- *Create a Tool in Gen AI Builder*
- *How-To View and Manage Threads in Gen AI Builder in Hybrid Manager*

Gen AI Builder FAQ

この FAQ では、ハイブリッドマネージャーの AI ファクトリー内の Gen AI Builder に焦点を当てています。モデル、ナレッジベース、アシスタント、ツール、SDK の使用、セキュリティ、パフォーマンス、操作に関する一般的な質問をカバーしています。

目次

- Typical use cases and patterns
- Which models/drivers are supported?
- When to use RAG vs. fine - tuning?
- How do I create and update Knowledge Bases?
- What are Retrievers and how do I tune them?
- How do Assistants call Tools and Structures?
- Where does data live and how is access governed?
- How do I improve latency and control cost?
- How do I test, observe, and troubleshoot?
- Where to find SDK references and examples?

どのモデル/ドライバーがサポートされていますか？

Gen AI Builder はドライバーベースです。プロンプトドライバー、埋め込みドライバー、および Vector ストアドライバーを介してプライベート/エンドポイントまたはホストされた API を使用できます。

- プロンプトドライバー OpenAI、Anthropic、Google、など [reference/sdk/drivers/index.mdx](#) およびプロバイダーごとのページを参照してください。
- 埋め込みドライバー OpenAI、Google、NVIDIA NIM、Hugging Face、など [reference/sdk/drivers/embedding-drivers.mdx](#)
- Vector Store ドライバー pgvector Postgres、local、Pinecone、Qdrant、Redis、その他 [reference/sdk/drivers/vector-store-drivers.mdx](#)
- NVIDIA NIM Model serving and inference を介してプライベートエンドポイントとして展開

AI Factory Models および *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com/container with KServe* も参照してください。

RAG と微調整を使用する場合

生きた知識と管理された回答には RAG を優先します。トーン/フォーマット/狭いタスクの微調整を使用します。多くのプロダクションアシスタントは、[Knowledge Bases Reference Manual](#) からコンテキストを取得し、[Rulesets Manual](#) でガイド応答をガイドします。モジュールパイプラインの場合、RAG Engines を探索します。

ナレッジベースを作成および更新するにはどうすればよいですか？

- *Create a Knowledge Base in Gen AI Builder* を作成し、*Configure Data Sources in Gen AI Builder* の構成 Confluence、Google ドライブ、S3、Web ページ、データレイク、カスタム
- アップデート *Manage Knowledge Bases in Gen AI Builder* ; ソースライブラリが変更されたときに再同期
- ストレージ 埋め込みは Postgres pgvector または構成済みのベクターストアにあります
- チューニング取得に一致するチャンキングとメタデータを選択します。黄金の質問で検証する

SDK 参照 Data Loaders 、 Embedding Drivers 、 Vector Store Drivers 。

レトリバーとは何ですか?どのように調整すればよいですか？

レトリバーは、アシスタントがコンテキストを取得する方法を制御しますターゲット KB、最大トークン、フィルタ、リランキング。

- 作成します *Create a Retriever in Gen AI Builder*
- 調整 類似性しきい値、top-K、メタデータフィルタ、リランカー
- Advanced 取得/リランク/レスポンスステージを備えたモジュール式 RAG Engines

SDK 参照 RAG Engines 、 Rerank Drivers 。

アシスタントはどのようにツールとストラクチャーを呼び出すのでしょうか？

アシスタントは、取得+生成+アクションをオーケストレーションします。外部システムのツールを使用するか、構造パイプライン/ワークフロー/エージェントを呼び出し可能なツールとしてプロモートします。

- ビルドアシスタント *Create an Assistant in Gen AI Builder* *Create a Ruleset in Gen AI Builder* 、 *Create a Retriever in Gen AI Builder* 、 および *Create a Tool in Gen AI Builder*
- 構造 ロジックを ZIP としてパッケージし、展開 *Create a Structure in Gen AI Builder* 。 ツールとして公開するか、スタンドアロンで実行
- メモリとスレッド *Threads Reference Manual* および会話メモリドライバーを参照してください

SDK リファレンス *Structures Reference Manual* 、 *Tools Reference Manual* 、 Assistant Drivers 、 Conversation Memory Drivers 。

データはどこにありますか、アクセスはどのように制御されますか？

- ドキュメント *Configure Data Sources in Gen AI Builder* 経由で取り込みます。管理されたデータレイクまたは Postgres に保存する
- 埋め込み/ベクター Postgres pgvector 推奨または別の構成済みベクターストア
- ガバナンスソースシステムに権限を適用します。プロジェクトごとにツールの使用を制限します。スレッド/ログを使用した監査取得とツール呼び出し

Configure the Data Lake in Gen AI Builder 、 *Vector Engine (PGvector)* を参照してください。

レイテンシーと制御コストを改善するにはどうすればよいですか？

- 取得トップ K の削減、チャンキング/メタデータの改善、再ランクの選択的な使用
- 生成ルートごとに適切なモデルを選択します。ストリーム応答。安全な場所でバッチ
- キャッシング可能であれば、埋め込み、ホット取得、ツール出力をメモ化します
- インフラ モデルと KB をコロケーションします。Model Serving の自動スケーリングを使用したスケーリング

SDK Engines およびドライバーを参照してください。モデル Model serving and inference を参照してください。

テスト、観察、およびトラブルシューティングを行うにはどうすればよいですか？

- テスト ゴールデンセット、会話プレイブック、SDK 単体テスト
- 可観測性スレッドログ、アシスタント/ストラクチャー実行イベント。オプションでオブザーバビリティスタックにエクスポート
- デバッグ最初取得セットを確認します。ルールセットの変更を検査します。保存された入力を使用して同じストラクチャ/アシスタントを再実行します

ドキュメント *Threads Reference Manual*、*SDK Observability for Model Serving*。

一般的なユースケースとパターン

- エンタープライズナレッジアシスタント KB + レトリバー + アシスタント + ツール チケット/CRM
- カスタマーサポートコパイロット ポリシー/よくある質問 KB + ルーティング + ルールセットを介したガードレール
- ワークフローボット承認、データエンリッチメント、およびレポートのための構造 + ツール
- 分析用 RAG pgvector + パイプライン + ガイド付き探索のためのアシスタント

Explore *Best practices for Hybrid Knowledge Bases*、*Customer Service Assistant Implementation Guide*。

SDK のリファレンスと例はどこにありますか？

- SDK の概要 Gen AI Builder SDK
- データ Artifacts、Loaders、Chunkers
- ドライバー *Prompt/Assistant*、Embeddings and vector representations、Vector Stores
- 構造体 *Tasks*、Pipelines and data preparation、Structures (workflows and building blocks)
- ツール *Tools Reference Manual*

製品ガイドも参照してください。*Assistants Reference Manual*、*Knowledge Bases Reference Manual*、*Retrievers Reference Manual*、*Rulesets Manual*、*Structures Reference Manual*、*Tools Reference Manual*、*Threads Reference Manual*。

Manual

Assistants Reference Manual

アシスタント Gen AI Builder、Griptape を採用は、インテリジェントアプリケーションのオーケストレーションレイヤーとして機能し、知識の取得、動作ガバナンス、ツールの実行を会話型インターフェイスに組み合わせます。これらは、静的な *Knowledge Bases Reference Manual* および *Tools Reference Manual* を、EDB' s vision for AI with Postgres and Hybrid Manager 内で実行されるインタラクティブな管理対象 AI エクスペリエンスに変換します。

目次

- *Getting Started - AI Factory 101*
- Practical Scenarios
- Configuration Checklist
- Architectural Role
- Core Architecture
- Operational Characteristics
- Integration Ecosystem
- Use cases
- SDK Quick Links
- FAQ

はじめに

- アシスタントの作成 *Create an Assistant in Gen AI Builder*
- データの準備 *Create a Knowledge Base in Gen AI Builder*、*Manage Knowledge Bases in Gen AI Builder*
- コントロールの取得 *Create a Retriever in Gen AI Builder*
- 動作の定義 *Create a Ruleset in Gen AI Builder*
- アクション *Create a Tool in Gen AI Builder* または *Create a Structure in Gen AI Builder* を追加し、ツールとして公開する

一般的なフロー ナレッジベース → レトリバー → ルールセット → ツール/構造 → アシスタント → *Threads Reference Manual*

実践的なシナリオ

- エンタープライズナレッジアシスタント引用付きの信頼できるドキュメントから従業員の質問に回答します。
- ビルドステップ *Create a Knowledge Base in Gen AI Builder* → *Create a Retriever in Gen AI Builder* → *Create a Ruleset in Gen AI Builder* → *Create an Assistant in Gen AI Builder*

- オプション チケット作成またはドキュメント検索のために *Tools Reference Manual* を追加します。
Threads Reference Manual のレビュー。
- カスタマーサポートコパイロットポリシー/よくある質問グラウンディングと CRM アクションでエージェントをガイドします。
- ビルドステップ KB のポリシー/よくある質問 → *Retrievers Reference Manual* 最新性のために調整されました → *Rulesets Manual* ガードルール → *Create an Assistant in Gen AI Builder*
- アクション *Create a Tool in Gen AI Builder* CRM /チケット用。ルールセットを介してスコープを適用します。
- ワークフローボット管理されたアクションで承認とデータエンリッチメントを自動化します。
- ビルドステップビジネスロジックを *Create a Structure in Gen AI Builder* としてパッケージし、ツールとして公開します → *Create an Assistant in Gen AI Builder*
- ロギング *Threads Reference Manual* で実行と結果を確認します。
- 分析用 RAG pgvector - backed retrieval.
- を使用してデータを探索するのに役立ちます ビルドステップ Pipelines and data preparation または *Configure the Data Lake in Gen AI Builder* + KB → *Retrievers Reference Manual* → *Create an Assistant in Gen AI Builder*
- で埋め込みを準備します
- 調整: チャンク/メタデータ、類似度しきい値、およびリランクモジュール。

構成チェックリスト

- タスクに適切なモデル/ドライバーの選択 SDK Which models/drivers are supported? を参照してください。
- ナレッジ 1 つ以上の *Knowledge Bases Reference Manual* および調整された *Retrievers Reference Manual*
- 動作 明確で競合のないガイダンスを備えた *Rulesets Manual*
- アクションスコープ指定された資格情報を使用した *Tools Reference Manual* および/または *Structures Reference Manual*
- メモリー/スレッド 監査、可観測性、およびレビューフローを有効にする *Threads Reference Manual*

アーキテクチャロール

アシスタントは、ユーザーと Gen AI Builder エコシステム間のプライマリインターフェイスとして機能し、複数のコンポーネントを調整して一貫した AI を利用したインタラクションを提供します。これらは、推論機能用の Model serving and inference、データ処理用の *AI Accelerator Pipelines*、セマンティック検索操作用の *Vector Engine (PGvector)* を含む他の AI Factory コンポーネントと統合しています。

システムインテグレーション

Gen AI Builder コンポーネント

- ナレッジベース 検索拡張生成パターンを介して事実の根拠を提供します
- レトリバー 組織のナレッジ全体でセマンティック検索操作を実行します
- ルールセット 動作パラメーターと操作境界を定義します
- ツール 外部システムの統合とアクションの実行を有効にします
- スレッド 会話状態とインタラクション履歴を維持します

AI ファクトリーインフラストラクチャ

- モデルサービング管理対象エンドポイントを介して言語モデル推論機能を提供します
- ハイブリッドマネージャー基になるオーケストレーションとリソース管理を提供します
- **Vector Engine** Postgres インフラストラクチャ内で高性能セマンティック検索を提供します

コアアーキテクチャ

会話処理パイプライン

アシスタントオペレーションは、一貫した動作と包括的な知識の利用を保証する構造化された処理シーケンスに従います。

1. **入力分析** ユーザークエリーは解析とインテント分類を受けます
2. **知識検索** 接続された知識ベースからの関連情報の抽出
3. **動作アプリケーション** 応答生成パラメーターに適用されるルールセットガイドライン
4. **ツール実行** 適切かつ許可された場合に実行される外部アクション
5. **応答生成** 取得したコンテキストと動作制約を使用した言語モデルの推論
6. **出力フォーマット** 引用とソースの帰属を含む応答パッケージ

コンポーネントオーケストレーション

ナレッジ統合アシスタントは、コンテンツタイプと組織の要件に基づいて検索戦略を最適化する *Retrievers Reference Manual* 構成を介して複数のナレッジベースを調整します。このアーキテクチャは、ベクトルの類似性とキーワード一致を組み合わせたハイブリッド検索アプローチをサポートしています。

行動ガバナンスルールセットは、アシスタントの応答が組織ポリシー、コンプライアンス要件、およびブランド標準と一致することを保証する操作境界を提供します。これらのガイドラインは、会話の柔軟性を維持しながら、すべてのインタラクションに一貫して適用されます。

ツール統合外部システム接続により、アシスタントはデータ更新、ワークフローの開始、サードパーティサービスの統合を含む情報取得を超えたアクションを実行できます。ツールの実行により、監査証跡とアクセス制御の適用が維持されます。 *Tools Reference Manual* および *Structures Reference Manual* を参照してください。

動作特性

会話管理

メモリシステムアシスタントは、マルチターンのインタラクションにわたって会話状態を維持し、コンテキストを保存しながら、言語モデルのコンストレインによって課されるメモリ制限を管理します。メモリシステムは、短期の会話の継続と長期の設定学習の両方をサポートします。

コンテキストウィンドウの最適化効率的なコンテキスト管理により、会話の一貫性を維持しながら、言語モデルのコンテキストウィンドウの最適な利用が保証されます。これには、会話履歴のインテリジェントな要約と動的なコンテキストの優先順位付けが含まれます。

パフォーマンスの最適化

応答生成言語モデルの統合は、ユーザーエクスペリエンスを向上させるためのストリーミング応答、高スループットシナリオのバッチ処理を含むさまざまな推論戦略をサポートします。応答生成には、ソースの帰属を維持しながら、取得したナレッジが組み込まれます。

取得効率ナレッジベースクエリーは、構成可能な取得戦略を介して精度とレイテンシーの両方を最適化します。これらには、類似度のしきい値、結果ランキングアルゴリズム、および頻繁にアクセスされる情報のキャッシュメカニズムが含まれます。

スケーラビリティフレームワーク

コンカレントオペレーションアシスタントは、個々の会話状態の分離を維持しながら、同時ユーザーのインタラクションをサポートします。リソースの割り当ては、需要パターンと組織のキャパシティ制約に基づいて動的にスケーリングします。

リソース管理ハイブリッドマネージャーインフラストラクチャとの統合により、モデル推論のための GPU 割り当てや会話データのストレージ最適化を含む、アシスタントインスタンス全体でのリソースの効率的な利用が可能になります。

構成フレームワーク

アシスタントの定義

コア構成アシスタントの作成には、言語モデルエンドポイント、知識ベースの関連付け、および動作パラメーターの仕様がが必要です。構成は、テンプレートベースの初期化とカスタムパラメーター指定の両方をサポートしています。

コンポーネントバインディングナレッジベース、取得者、ルールセット、およびツールは、運用の柔軟性を維持しながら、展開全体で一貫した動作を保証する宣言的構成を介してアシスタントに接続します。

動作構成

ルールセットの適用動作ガイドラインは、トーン、形式、ドメインの専門知識、操作境界を含む応答特性を定義します。ルールセットは、会話コンテキストに基づいて、静的構成と動的な適応の両方をサポートします。

応答パラメーター温度、トークン制限、サンプリング戦略を含む言語モデルパラメーターは、応答生成特性を構成します。これらのパラメーターは、アプリケーション要件に基づいて、創造性と一貫性のバランスをとります。

統合構成

ナレッジベースの接続複数のナレッジベースは、検索戦略、コンテンツフィルタリング、結果ランキングのアプローチを指定する取得構成を介して個々のアシスタントに接続します。このアーキテクチャは、専門的な知識ドメインとアクセス制御要件をサポートします。

ツール承認外部システム統合には、許可される操作、データアクセス境界、および監査ログ要件を定義する明示的な承認構成が必要です。ツール構成は、セキュリティの分離を維持しながら、必要な機能を有効にします。

実装パターン

エンタープライズナレッジシステム

部門の専門家アシスタントは、特定の組織ドメインの主題専門家インターフェイスとして機能し、深い知識ベースとさまざまなユーザーコミュニティの適切な行動特性を組み合わせます。

クロスファンクションサポートマルチドメインアシスタントは、ユーザーの権限とクエリ特性に基づいて、適切なアクセス制御と情報ルーティングを維持しながら、さまざまな組織ソースからの知識を統合します。

顧客向けアプリケーション

サポートオートメーションカスタマーサービスアシスタントは、製品ドキュメント、ポリシー情報、およびトラブルシューティングの知識を組み合わせ、複雑な問題のエスカレーション経路を維持しながら、一貫したサポートエクスペリエンスを提供します。

セルフサービスポータル会話インターフェイスを介したインタラクティブなナレッジアクセスは、サポートの負担を削減しながら、顧客のセルフサービスシナリオに包括的な情報検出機能を提供します。

社内生産性向上ツール

リサーチアシスタンスナレッジワーカーは、多様な内部ソースにわたって情報を集約するアシスタントの恩恵を受け、適切な帰属とソース検証を維持しながら、組織のナレッジへの迅速なアクセスを提供します。

意思決定サポート複雑な意思決定シナリオでは、定量的データ分析と定性的な知識の合成を組み合わせたアシスタントを活用して、情報に基づいた組織の選択をサポートします。

品質保証

応答の検証

精度検証体系的な評価手順により、アシスタントの回答が事実の正しさを維持しながらソース知識と一致することが保証されます。これには、重要なアプリケーションの自動検証と人間のレビュープロセスの両方が含まれます。

引用の完全性ソース帰属の精度は、組織の透明性をサポートし、ユーザーが独立して情報を検証できるようにします。引用システムは、応答生成までオリジナルソースからの詳細な出所追跡を維持します。

パフォーマンスモニタリング

会話分析包括的なメトリックは、ユーザー満足度、クエリ解決率、エスカレーションパターンなどのアシスタントの有効性を追跡します。分析は、継続的な改善と最適化戦略をサポートします。

システムパフォーマンス技術的なパフォーマンスモニタリングには、応答遅延、リソース使用率、およびシステム可用性メトリックが含まれ、さまざまな負荷条件にわたって一貫したユーザーエクスペリエンスを保証します。

運用上の考慮事項

導入戦略

開発ワークフローアシスタント開発は、ナレッジベースの検証、動作テスト、運用展開前の統合検証を含む体系的な手順に従います。開発環境は、反復の調整とテストをサポートします。 *Create an Assistant in Gen AI Builder* および SDK リファレンスを参照してください。

プロダクション管理ライブアシスタント管理には、パフォーマンスモニタリング、コンテンツの更新、および操作上のフィードバックと組織の要件の変化に基づいた動作調整が含まれます。

メンテナンス手順

ナレッジベースの更新コンテンツの同期により、アシスタントは現在の組織情報にアクセスし、移行期間中の応答の一貫性を維持できます。更新プロシージャは、インクリメンタルな変更と包括的なコンテンツの更新の両方をサポートしています。

動作の洗練ルールセットを変更すると、ユーザーのフィードバック、組織ポリシーの変更、および操作経験に基づいて、アシスタントの動作を適応できます。行動の変化は、組織の進化をサポートしながら一貫性を維持します。

セキュリティフレームワーク

アクセス制御

ユーザー認証アシスタントのアクセスは、組織の ID 管理システムと統合し、展開要件とセキュリティポリシーに基づいて、認証されたユーザーと匿名の相互作用の両方をサポートします。

情報ガバナンスナレッジベースのアクセスは、組織のデータ分類とユーザー権限フレームワークを尊重し、包括的な知識のアクセシビリティを維持しながら、適切な情報の公開を保証します。

監査機能

インタラクションログ詳細な会話ログは、コンプライアンス要件、運用分析、およびセキュリティモニタリングをサポートし、プライバシー考慮事項とデータ保持ポリシーを尊重します。

決定トレーサビリティ応答生成プロセスは、説明責任と改善を目的とした情報ソース、推論経路、および決定要素の検証を可能にする包括的な監査証跡を維持します。

統合エコシステム

AI ファクトリーコンポーネント

アシスタントは、より広範な AI Factory エコシステム内で動作し、次の機能を活用します。

- **** Model serving and inference **** 管理された言語モデルの推論を提供します
- **** AI Accelerator Pipelines **** ナレッジベースのコンテンツ処理を有効にします
- **** Vector Engine (PGvector) **** セマンティック検索インフラストラクチャを提供

開発リソース

実装ガイド

- *Create an Assistant in Gen AI Builder* ステップバイステップのアシスタント構成
- *Knowledge Bases Reference Manual* 包括的なナレッジマネジメント
- *Tools Reference Manual* 外部システム統合パターン

ユースケース

- エンタープライズナレッジアシスタント KB + レトリバー + アシスタント + ツール 例 チケット発行/CRM
- カスタマーサポートコパイロット ポリシー/よくある質問 KB + ルーティング + ルールセットを介したガードレール
- ワークフローボット 承認、エンリッチメント、レポートのための構造 + ツール
- 分析用 RAG pgvector + パイプライン + ガイド付き探索のためのアシスタント

SDK クイックリンク

- SDK の概要 Gen AI Builder SDK
- ドライバー *Assistant/Prompt*、Conversation Memory
- 構造体 *Agents*、*Tasks*、Structures (workflows and building blocks)
- オブザーバビリティ *Structures/observability*

よくある質問

- アシスタントを複数のナレッジベースに接続するにはどうすればよいですか？ - 複数の KB をターゲットとする *Create a Retriever in Gen AI Builder* を作成します。 *Create an Assistant in Gen AI Builder* で割り当てます。ソースごとにメタデータフィルターを使用します。
- アシスタントに内部 API を呼び出させるにはどうすればよいですか？ - スコープされた資格情報を使用して *Create a Tool in Gen AI Builder* をパッケージ化するか、 *Create a Structure in Gen AI Builder* をプロモートします。アシスタントに接続します。 *Create a Ruleset in Gen AI Builder* を介して使用を強制します。

- テストとデバッグはどうすればよいですか？ - テストチャットと *Threads Reference Manual* を使用します。取得セットを検証し、ルールセットリビジョンを確認し、保存された入力で再実行します。必要に応じてトレースをエクスポートします。
 - モデル/ドライバーはどのように選択すればよいですか？ - 一般的なモデルから始めます。タスク固有のプロンプトを安価/高速のドライバーにルーティングします。プライベートエンドポイントの場合、Model serving and inference を介して統合します。
-

アシスタントは、Gen AI Builder 内のインテリジェントアプリケーションの基本的なインターフェイスを提供し、知識、動作、機能を組織のガバナンスフレームワーク内で動作する一貫した会話エクスペリエンスにオーケストレーションします。

Knowledge Bases Reference Manual

ナレッジベース リファレンスマニュアル

ナレッジベースは、組織のコンテンツを照会可能なナレッジリポジトリに変換するセマンティック検索インフラストラクチャを提供し、AI アシスタントが *retrieval - augmented generation* ワークフローを介して特定の組織のナレッジにアクセスして利用できるようにします。これらは、処理されたコンテンツと、コンテキスト情報を必要とするインテリジェントアプリケーション間の操作インターフェイスとして機能します。

目次

- *Getting Started - AI Factory 101*
- Practical Scenarios
- Architectural Purpose
- Technical Architecture
- Content Organization Framework
- Search Operation Types
- Implementation Patterns
- Operational Considerations
- Development Resources

はじめに

- KB を作成します *Create a Knowledge Base in Gen AI Builder*
- ソースの準備 *Configure Data Sources in Gen AI Builder* Confluence、Google Drive、S3、Web ページ、Data Lake、Custom
- ストレージの構成 *Configure the Data Lake in Gen AI Builder* オブジェクトストレージ取り込みを使用する場合
- 管理と更新 *Manage Knowledge Bases in Gen AI Builder*
- アプリに接続する *Retrievers Reference Manual* および *Assistants Reference Manual* を使用

実践的なシナリオ

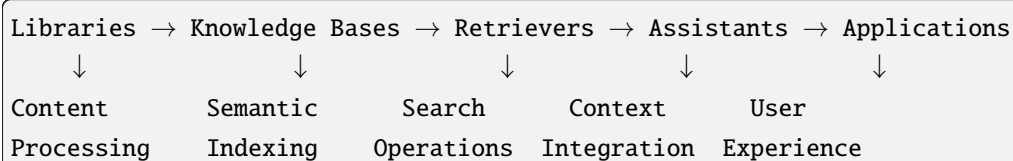
- ポリシーと手順 KB 企業内部 強力な引用を使用して HR / IT / 法務ドキュメントを一元化します。 - ビルド手順 *Configure Data Sources in Gen AI Builder* → *Create a Knowledge Base in Gen AI Builder* を介してソースを接続します → ゴールデンの質問で取得を検証する → *Create a Retriever in Gen AI Builder* を介して接続します。
- 製品ドキュメント KB サポートマニュアルと FAQ に関する地上エージェントの回答。 - ビルド手順ドキュメントを取り込み S3 / Drive / Confluence → *Create a Knowledge Base in Gen AI Builder* → チューンチャンキング/メタデータ → サポートへのリンク *Create an Assistant in Gen AI Builder* 。
- ハイブリッドカタログ KB 構造化+非構造化製品属性構造化と説明/レビュー非構造化を組み合わせます。 - ビルド手順 構造化列と非構造化テキストを準備する → *Create a Knowledge Base in Gen AI Builder* でハイブリッド KB を作成する → *Create a Retriever in Gen AI Builder* でフィルターを公開する。
- Analytics RAG KB レポートと Wiki ナレッジのセマンティック探索を有効にします。 - ビルドステップ Pipelines and data preparation またはデータレイク+ローダーを介して埋め込みを準備します → *Create a Knowledge Base in Gen AI Builder* → 関連性とレイテンシーをテストします。 *Create an Assistant in Gen AI Builder* と統合します。

アーキテクチャの目的

ナレッジベースは、静的コンテンツコレクションを動的なクエリ可能なリソースに変換するセマンティック検索レイヤーとして機能し、状況理解機能を備えた AI アプリケーションをサポートします。これらは、メタデータフィルタリングと組み合わせてベクトルベースの類似性検索を実装し、言語モデルのコンテキスト強化に関連する情報を提供します。

システムインテグレーションフレームワーク

Gen AI Builder パイプラインのポジション ナレッジベースは、AI Factory エコシステム内のコンテンツ処理とアプリケーション使用の間の重要なポジションを占めています。



インフラストラクチャの統合

- **Vector Engine** PostgreSQL ベースのベクトルストレージを活用して、高性能セマンティック検索操作を実現
- **ライブラリ** 埋め込みベクトルと強化されたメタデータを含む処理されたコンテンツを使用します
- **AI Accelerator Pipelines** セマンティックインデックスとコンテンツ変換に埋め込みモデルを利用します
- **モデルサービス** クエリの理解と応答生成のための言語モデルにアクセスします

テクニカルアーキテクチャ

Vector ストレージフレームワーク

セマンティックインデックスナレッジベースは、組織のコンテンツの特性に最適化された特殊な言語モデルによって生成された高次元の埋め込みを介して意味をキャプチャする高度なベクターストレージを実装します。

ストレージの最適化 Vector ストレージは、Vector エンジンを通じて PostgreSQL のネイティブベクター機能を利用し、包括的なクエリ機能と運用管理機能を維持しながら、エンタープライズグレードのパフォーマンス、スケーラビリティ、および信頼性を提供します。

検索機能

類似度アルゴリズムナレッジベースは、コサイン類似度、ユークリッド距離、意味的理解と組織の優先順位要素を組み合わせたハイブリッドスコアリングメカニズムを含む複数の類似度測定アプローチを実装しています。

クエリ処理パイプライン検索操作は、計算効率を維持しながら結果の関連性を最適化する体系的な処理に従います。

1. **クエリベクトル化**組織的な埋め込みモデルを使用してセマンティック埋め込みに変換されたユーザークエリ
2. **類似性の計算**インデックス付けされたコンテンツコレクションにわたるベクトルの類似性計算
3. **メタデータフィルタリング**アクセス制御、コンテンツ分類、および組織ポリシーに基づいた結果の絞り込み
4. **関連性ランキング**意味的な類似性、コンテンツの最新性、およびソースの信頼性を組み込んだ多要素スコアリング
5. **コンテキストアセンブリ**言語モデルコンテキスト統合用に最適化された結果の選択とフォーマット

コンテンツ編成フレームワーク

インデックス戦略

コンテンツのセグメンテーションナレッジベースは、意味の一貫性を維持しながら、取得の粒度と言語モデルのコンテキストウィンドウに最適化するインテリジェントなコンテンツチャンクを実装します。

階層組織コンテンツ構造の保持により、ドキュメントの関係、セクション階層、および相互参照接続が維持され、包括的な情報検出と正確なソースの帰属が可能になります。

メタデータの統合

分類システム包括的なメタデータフレームワークは、コンテンツタイプ、アクセスレベル、件名分類、時間的関連性指標を含む組織の分類スキームをサポートします。

アクセス制御の統合メタデータベースのアクセス制御は、組織のセキュリティポリシーおよびユーザー権限フレームワークに合わせて適切な情報露出を保証します。

検索操作タイプ

セマンティック検索

ベクトルの類似性 純粋なセマンティック検索操作は、キーワード一致に関係なく概念的な類似性に基づいてコンテンツを識別し、リテラルのテキスト一致ではなく意味を介して関連情報の発見を可能にします。

コンテキストの理解 高度なセマンティック検索には、会話コンテキスト、ユーザーの意図、および組織ドメインの知識を組み込み、情報検出の関連性と精度を強化します。

ハイブリッド検索オペレーション

複合アプローチ ハイブリッド検索戦略は、ベクトルの類似性を従来のキーワードマッチング、メタデータフィルタリング、および組織の優先度要素と統合して、意味的な理解と正確な情報取得の最適なバランスを実現します。

パフォーマンスの最適化 インテリジェントなクエリルーティングは、クエリ特性、コンテンツタイプ、およびパフォーマンス要件に基づいて最適な検索戦略を選択し、結果の関連性を最大化しながら一貫した応答時間を保証します。

フィルタリングされた検索

メタデータベースのフィルタリング 高度なフィルタリング機能により、組織の分類、アクセス権限、コンテンツの最新性、ソース信頼性指標に基づいて、正確なコンテンツを選択できます。

動的フィルタリング コンテキスト認識のフィルタリングは、ユーザーロール、会話コンテキスト、および組織ポリシーに基づいて検索範囲を調整し、包括的なカバレッジを維持しながら適切な情報露出を保証します。

パフォーマンス特性

スケーラビリティフレームワーク

ホリゾンタル スケーリング ナレッジベースアーキテクチャは、インテリジェントな負荷分散を通じて検索の一貫性とパフォーマンス特性を維持しながら、複数の PostgreSQL インスタンスにわたるホリゾンタルスケーリングをサポートします。

クエリの最適化 高度なクエリプランニングは、インテリジェントなインデックス戦略とキャッシュメカニズムを介して、増加するコンテンツボリューム全体で応答時間の一貫性を保つように検索操作を最適化します。

リソース管理

メモリの最適化 Vector ストレージの最適化は、インテリジェントなインデックス戦略とストレージ圧縮技術を介して、包括的なコンテンツカバレッジとメモリ効率のバランスをとります。

計算効率 検索操作は、インテリジェントなクエリプランニング、結果キャッシュ、および並列処理機能を介して計算リソースの使用率を最適化します。

品質保証

コンテンツ品質管理

取り込み検証ナレッジベースの作成中の体系的なコンテンツ検証により、情報の精度、完全性、および最適な検索効果のための適切なフォーマットが保証されます。

進行中の品質評価検索効果とコンテンツ品質の継続的なモニタリングは、パフォーマンス分析とユーザーフィードバックの統合を介した体系的な改善をサポートします。

検索の有効性

関連性モニタリング検索結果の関連性の包括的な評価は、ユーザーのインタラクション、フィードバックパターン、タスク完了率の体系的な分析を介して継続的な最適化をサポートします。

パフォーマンス追跡詳細なパフォーマンスモニタリングは、検索応答時間、結果の精度、およびリソース使用パターンを追跡し、キャパシティプランニングと最適化の決定をサポートします。

実装パターン

エンタープライズナレッジシステム

組織は、適切なアクセス制御と組織ガバナンス要件を維持しながら、多様なコンテンツソースを統合する包括的なナレッジベースを実装します。

ユニファイドアクセスアーキテクチャ

- 一元化されたナレッジベースインフラストラクチャを介した部門を超えたコンテンツの統合
- 適切な情報露出を保証するロールベースのアクセス制御
- コンプライアンスと運用監視をサポートする包括的な監査証跡
- 同時アクセスと組織のスケーラビリティをサポートするパフォーマンスの最適化

特殊なドメインアプリケーション

ドメイン固有の実装は、特殊なコンテンツ構成と検索特性を必要とする特定の業界またはユースケースのナレッジベースを最適化します。

ドメイン最適化機能

- ドメイン用語と概念に最適化された業界固有の埋め込みモデル
- ドメイン固有の分類と編成をサポートする特殊なメタデータスキーマ
- 業界固有の要件の遵守を保証する規制コンプライアンスの統合
- ドメイン固有のクエリパターンとコンテンツ特性のパフォーマンスの最適化

マルチモーダルな知識の統合

高度な実装は、統一された検索エクスペリエンスを維持しながら、ドキュメント、構造化データ、マルチメディアリソースを含む多様なコンテンツタイプにわたって知識を調整します。

クロスモーダル機能

- コンテンツタイプの最適化により、多様なメディア形式にわたって適切な処理を保証
- さまざまな情報種類にわたって一貫したアクセスを提供する統合検索インターフェイス
- 多様なコンテンツ処理と取得要件をサポートするパフォーマンスの最適化
- コンテンツモダリティ全体で一貫した結果を保証する品質保証手順

運用上の考慮事項

展開管理

構成管理 ナレッジベースの展開には、組織のガバナンス要件をサポートしながら、開発、ステージング、および運用環境全体で一貫した検索動作を保証する体系的な構成管理が必要です。

更新プロシージャ コンテンツの更新には、コンテンツの更新操作中のサービスの中断を最小限に抑えながら、検索効率を最適な状態に保つ、調整されたインデックスのメンテナンスが必要です。

メンテナンスフレームワーク

インデックスの最適化 定期的なメンテナンス手順は、インデックスの再構築、ベクトル最適化、およびシステム効率とユーザーエクスペリエンスをサポートするストレージ統合を介して最適な検索パフォーマンスを保証します。

コンテンツの同期 体系的な手順は、ナレッジベースの更新操作とライブラリからのコンテンツの更新を調整し、検索パフォーマンス特性を維持しながら情報の最新性を保証します。

開発リソース

構成プロシージャ

ナレッジベースの設定

- *Create a Knowledge Base in Gen AI Builder* 包括的なナレッジベースの作成と構成手順
- *Manage Knowledge Bases in Gen AI Builder* 進行中の管理と最適化手法
- *Best practices for Hybrid Knowledge Bases* 最適な検索効率のための高度な構成

統合ドキュメント

システムの接続性

- *Libraries Ω Manual* コンテンツソースの構成と処理の最適化
- *Retrievers Reference Manual* 検索インターフェイスの構成と最適化
- *Assistants Reference Manual* 会話システム内でのナレッジベースの利用

テクニカルリファレンス

- Vector Store Drivers 技術的実装の詳細
 - Embedding Integration セマンティックインデックス作成構成
 - RAG Engine Reference 高度な取得拡張生成の実装
-

ナレッジベースは、静的な組織コンテンツを動的なインテリジェントなリソースに変換します。これにより、AI アプリケーションは、適切なアクセス制御とパフォーマンス特性を維持しながら、特定の組織の知識に基づいて、状況に関連した正確な応答を提供できます。

Configure Data Sources in Gen AI Builder

これは誰に向けたものですか

Gen AI Builder を使用してナレッジベースと AI アプリケーションを構築するプラットフォームユーザー。これには、開発者、AI エンジニア、コンテンツマネージャー、システムインテグレーターが含まれます。

ドキュメント、Web ページ、クラウドストレージ、または独自のシステムから外部または内部コンテンツを AI ワークフローに取り込む責任がある場合、このガイドセットはその方法を示します。

あなたが達成すること

Gen AI Builder でさまざまな種類のデータソースを構成および管理する方法を学習します。これらのソースは、AI アプリケーションのライブラリとナレッジベースを設定する生のコンテンツを提供します。

データソースを使用する理由

- AI アプリケーションの品質と関連性は、取り込むコンテンツに大きく依存します。
- Gen AI Builder を使用すると、さまざまなコンテンツソースに接続し、取り込みパイプラインを安全かつ効率的に管理できます。
- データソースを使用すると、ナレッジベースを新鮮で正確に保ち、組織のニーズと一致することが保証されます。

Gen AI Builder のデータソースについて

データソースは、コンテンツをシステムに取り込むための最初のステップです。データソースからのコンテンツはライブラリ に流れ、次に ナレッジベースに流れ、そこでインデックスが作成され、AI ワークロードで利用できるようになります。

一般的なデータソースのタイプには次のものが含まれます。

- Web ページ
- データレイク
- Amazon S3 バケット
- Google Drive フォルダー
- Atlassian Confluence スペース

- PG.AI 構造を介したカスタムシステム

主な機能

- データへの安全な接続
- コンテンツを最新の状態に保つためのスケジュールされた更新
- PG.AI 構造を使用したオプションの変換
- 下流のライブラリおよびナレッジベースとの統合

基本的な概念については、以下を参照してください。

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Structures Reference Manual*
- *Embeddings explained*
- *Retrieval Augmented Generation (RAG)*

AI アプリケーションでこのコンテンツを使用するためのガイダンスについては、次のとおりです。

- Knowledge Bases overview
- *Create a Knowledge Base in Gen AI Builder*

ハイブリッドマネージャー展開内でデータソースがどのように使用されるかを確認するには、以下にアクセスしてください。

- Hybrid Manager: Using Gen AI Builder

ハウツーガイド

次のガイドを使用して、各種類のデータソースを構成します。

- *Configure a Web Page data source*
- *Configure a Data Lake data source*
- *Configure an Amazon S3 data source*
- *Configure a Google Drive data source*
- *Configure an Atlassian Confluence data source*
- *Configure a Custom data source*

各ガイドは以下を提供します。

- 誰に向けて
- あなたが達成すること
- このソースを使用する理由
- 詳細な構成手順
- 重要な考慮事項とベストプラクティス

- 障害対応のヒント
- サンプルシナリオ

次のステップ

データソースの構成が完了したら、次の手順に進みます。

- ライブラリを構成してコンテンツを整理します。
- ナレッジベースを構築して、セマンティック検索と RAG アプリケーションを有効にします。
- Gen AI Builder と Structures を使用して、ナレッジベースを AI エージェントとワークフローに接続します。

より高度な統合については、以下を参照してください。

- *Create a Knowledge Base in Gen AI Builder*
- *Structures Reference Manual*

SDK リファレンス

- Data Loaders
- File Manager Drivers
- Web Scraper Drivers

データソースを注意深く構成することにより、組織内に高品質の状況認識 AI アプリケーションの強力な基盤を構築します。

Retrievers Reference Manual

レトリバーリファレンスマニュアル

レトリバーは、AI アシスタントが正確な構成可能な検索操作を介して組織の知識にアクセスして利用できるようにする特殊な情報取得機能を提供します。これらは、ユーザークエリとナレッジベースコンテンツ間の重要なインターフェイスとして機能し、取得拡張生成ワークフローの取得コンポーネントを実装します。

アーキテクチャファンクション

レトリバーは、組織的な制約と最適化戦略を適用しながら、ナレッジベース全体でセマンティック検索操作をオーケストレーションする特殊なグリップテープ構造として動作します。これらはユーザークエリを構造化した検索操作に変換し、言語モデルコンテキスト統合に関連するコンテンツを識別します。

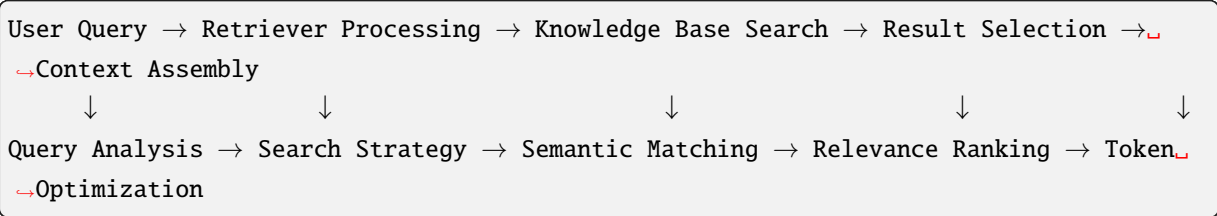
システムインテグレーションフレームワーク

Gen AI Builder の統合 レトリバーは、コア Gen AI Builder コンポーネントとシームレスに統合して、包括的な知識駆動型アプリケーションを有効にします。

- ナレッジベースインデックス付けされたコンテンツコレクション全体でセマンティック検索操作を実行する

- アシスタントインフォームドレスポンスを生成するためのコンテキスト情報を提供します
- ライブラリ最適化された取得インターフェイスを介して処理されたコンテンツにアクセスします
- **Vector Engine PostgreSQL** インフラストラクチャ内の高性能セマンティック検索機能を活用

取得パイプラインアーキテクチャ



このパイプラインは、計算リソースを管理し、応答品質基準を維持しながら、最適なコンテンツ検出を保証します。

テクニカルアーキテクチャ

取得戦略フレームワーク

検索オペレーションタイプレトリバーは、さまざまなコンテンツタイプと組織の要件に最適化された複数の検索戦略を実装します。

Strategy Type	Primary Function	Optimization Focus
Semantic Search	Vector similarity matching	Content relevance and contextual understanding
Hybrid Search	Combined vector and keyword approaches	Precision balance between semantic and exact matching
Filtered Search	Metadata-based content selection	Access control and content classification adherence
Multi-Source Search	Cross-knowledge base retrieval	Comprehensive information coverage

クエリ処理パイプラインレトリバー操作は、パフォーマンス要件を維持しながら検索効率を最適化する体系的なクエリ処理に従います。

1. **クエリ分析** ユーザー入力の解析と_intentの分類
2. **検索戦略の選択** クエリ特性に基づいた最適なアプローチの決定
3. **ナレッジベースのターゲティング** 適切なコンテンツソースの識別と優先順位付け
4. **セマンティックマッチング** ベクトルの類似性計算と関連性スコアリング
5. **結果アセンブリ** コンテンツ選択とコンテキストウィンドウの最適化

コンテンツ選択フレームワーク

関連性の最適化レトリバーは、意味的な類似性と、コンテンツの最新性、ソース権限、ユーザーのアクセス権限を含む組織の優先順位要素を組み合わせた高度な関連性スコアリングを実装します。

コンテキスト管理トークン制限管理は、計算制約内で情報の完全性を維持するインテリジェントなコンテンツ選択、要約、および優先順位付け戦略を介して、最適な言語モデルコンテキストの利用を保証します。

構成アーキテクチャ

ナレッジベースのターゲティング

ソース選択レトリバー構成は、検索操作のための特定のナレッジベースを定義し、情報スコープの正確な制御を可能にし、組織の要件に基づいて適切なコンテンツアクセスを保証します。

マルチソース調整アドバンスドレトリバーは、複数のナレッジベースにわたって検索を調整し、さまざまなコンテンツソースにわたって結果の統合、重複排除、および関連性ランキングを管理します。

パフォーマンスパラメーター

トークン制限最大トークン構成は、取得したコンテンツの量を制御し、言語モデルのコンテキストウィンドウの制限および処理効率要件と包括的な情報提供のバランスをとります。

検索の深さ結果量パラメーターは、取得操作中にコンテンツを検討する幅を定義し、包括的なカバレッジと計算リソースの使用率の間で最適化します。

メタデータの統合

コンテンツ分類メタデータの利用により、組織の分類スキーム、アクセス制御要件、およびコンテンツ品質指標に基づいて、高度なコンテンツフィルタリングが可能になります。

コンテキストの強化高度なメタデータの統合は、クエリの強化、結果ランキングの最適化、および組織の透明性と監査機能のためのコンテンツ帰属要件をサポートします。

動作特性

検索パフォーマンス

応答の最適化レトリバーは、インテリジェントなキャッシュ、インデックス最適化、およびクエリプランニング戦略を通じて結果の品質を維持しながら、一貫した応答時間で検索操作を最適化します。

スケーラビリティフレームワーク検索操作は、分散検索機能とインテリジェントなリソース割り当て戦略を介して、増大するナレッジベースコレクション全体にわたって効率的にスケーリングします。

品質保証

結果の検証体系的な結果品質評価は、自動検証と継続的モニタリング手順を介して、取得されたコンテンツが関連性のしきい値と組織の品質基準を満たしていることを保証します。

関連性モニタリング取得有効性の継続的な評価は、パフォーマンス分析、ユーザーフィードバックの統合、および体系的な改善手順を介した継続的な最適化をサポートします。

実装パターン

エンタープライズナレッジアクセス

組織は、適切なアクセス制御と組織ガバナンス要件を維持しながら、多様な知識ソースへの統一アクセスを提供する包括的な検索システムを実装しています。

統合検索アーキテクチャ

- 一元的な検索機能を介した部門を超えた知識の統合
- 適切な情報露出を保証するロールベースのコンテンツアクセス
- コンプライアンスと運用監視をサポートする包括的な監査証跡
- 同時ユーザーアクセスとシステムのスケーラビリティをサポートするパフォーマンスの最適化

特殊なドメインアプリケーション

ドメイン固有の実装は、特殊な知識組織とアクセスパターンを必要とする特定の業界またはユースケースの取得操作を最適化します。

ドメイン最適化機能

- 業界固有の関連性スコアリングアルゴリズム
- ドメイン固有のコンテンツ構成に特化したメタデータの利用
- 業界固有の要件をサポートする規制コンプライアンスの統合
- ドメイン固有のクエリパターンとコンテンツ特性のパフォーマンスの最適化

マルチモーダル情報システム

高度な実装は、統一された検索エクスペリエンスを維持しながら、ドキュメント、構造化データ、マルチメディアリソースを含む多様なコンテンツタイプにわたる取得を調整します。

クロスモーダル統合

- コンテンツタイプの最適化により、多様なメディア形式にわたって適切な処理を保証
- さまざまな情報種類と構造にわたる統合された関連性スコア
- 多様なコンテンツ処理要件をサポートするパフォーマンスの最適化
- コンテンツモダリティ全体で一貫した結果を保証する品質保証手順

品質フレームワーク

取得効果

関連性評価体系的な評価手順は、精度、再現率、ユーザー満足度の測定などの標準化されたメトリックを介して、取得の精度と有効性を検証します。

パフォーマンスモニタリング 包括的なモニタリングは、応答時間、結果の品質、最適化とキャパシティプランニングの決定をサポートするリソース使用パターンを含む取得パフォーマンス特性を追跡します。

継続的な改善

最適化手順取得パラメーターの定期的な評価と最適化は、パフォーマンス特性の体系的な分析とユーザーフィードバックの統合を介して継続的な改善をサポートします。

フィードバック統合ユーザーインタラクション分析とフィードバックメカニズムは、検索の最適化を情報提供し、検索効果とユーザーエクスペリエンス品質の体系的な強化を可能にします。

運用上の考慮事項

展開管理

構成管理 Retriever の展開には、組織のガバナンス要件をサポートしながら、開発、ステージング、および実稼働環境全体で一貫した動作を保証する体系的な構成管理が必要です。

バージョン管理体系的なバージョン管理手順は、運用上の問題が発生した場合、または最適化要件が変更された場合に、信頼性の高い展開管理とロールバック機能をサポートします。

メンテナンス手順

パフォーマンスの最適化定期的なメンテナンス手順は、インデックス最適化、キャッシュ管理、およびシステム効率とユーザーエクスペリエンスをサポートするクエリパターン分析を介して最適な取得パフォーマンスを保証します。

コンテンツの同期ナレッジベースの更新には、コンテンツが進化し、組織の要件が変化しても検索効率を最適な状態に保つための、調整された取得者の最適化が必要です。

統合リソース

構成プロシージャ

レトリバーのセットアップ

- *Create a Retriever in Gen AI Builder* 包括的なレトリバーの構成と展開の手順
- *Retrievers Reference Manual* 最適な検索効率のための高度な構成テクニック

システムインテグレーション

コンポーネントの接続

- *Assistants Reference Manual* 会話システム内でのレトリバーの使用
- *Libraries Ω Manual* コンテンツソースの構成と最適化
- *Structures Reference Manual* 高度な取得ワークフロー開発

インフラストラクチャの統合

- *Vector Engine Integration* 高性能セマンティック検索機能
- *Hybrid Manager Deployment* インフラ管理とリソースの最適化

SDK リファレンス

- RAG Engines
- Embedding Drivers
- Rerank Drivers
- Vector Store Drivers

レトリバーは、静的知識リポジトリを動的でクエリ可能なリソースに変換する重要な情報アクセス機能を提供し、包括的な組織知識の統合とセマンティックな理解でインテリジェントアプリケーションをサポートします。

Rulesets Manual

ルールセットは、応答パターン、通信標準、操作上の制約を定義する自然言語指示を介して、AI アシスタントの動作ガバナンスを提供します。これらは、知識検索システムを補完する明示的な動作制御を確立して、組織の AI アプリケーション全体で一貫したコンプライアンスの相互作用を保証します。

アーキテクチャファンクション

ルールセットは、Gen AI Builder 内の動作制御レイヤーとして機能し、組織の標準とコンプライアンス要件を維持しながら、アシスタントがユーザーのインタラクションを解釈して応答する方法を定義します。これらはプロンプトレベルで動作し、応答生成が発生する前に言語モデルの動作に影響を与えます。

システム統合コンテキスト

Gen AI Builder コンポーネント

- **アシスタント** 会話処理中にルールセットを適用して、動作の一貫性を保証します
- **ナレッジベース** 検索システムと連携して、事実の精度と適切な動作を組み合わせます
- **構造複雑なワークフロー** 操作内にルールセットガイダンスを組み込みます
- **ツール** 外部システムの相互作用中に動作コンストレインを適用します

動作制御レイヤー ルールセットは、定義された組織のガイドラインに従ってユーザー入力とアシスタントの応答を処理するミドルウェアコンポーネントとして機能します。この制御レイヤーは、会話の複雑さやトピックの変化に関係なく、動作標準の一貫した適用を保証します。

コアアーキテクチャ

ルールセットコンポーネント

構造要素

- **名前** 組織参照と管理のための一意の識別子
- **説明** ルールセットの目的と意図されたアプリケーションのドキュメント
- **ルールコレクション** 自然言語で書かれた個々の行動指示

- **エイリアスプログラム**によるアクセスとバージョン管理のためのオプションの識別子
- **メタデータ** 高度な動作制御のための追加構成パラメーター

ルール定義フレームワークルールセット内の個々のルールは次で構成されています。

- **ルール名** 特定の動作指示の説明的な識別子
- **ルールテキスト** 予想される動作を定義する自然言語指示
- **優先レベル** 競合が発生した場合のルール適用の暗黙的な順序付け
- **コンテキストスコープ** ルール適用のためのアプリケーション境界

行動アプリケーションプロセス

ランタイム統合ルールセットは、会話フロー中に体系的なアプリケーションを介してアシスタント処理に統合されます。

1. **入力処理** ルールセットコンストレインに対して評価されるユーザーメッセージ
2. **コンテキストの確立** 会話コンテキストに基づいて有効化される関連ルール
3. **応答生成** アクティブなルールセット指示によってガイドされる言語モデル
4. **出力検証** 動作要件に対して検証された生成された応答
5. **一貫性の適用** 会話全体を通じた進行中の動作モニタリング

ルールの優先順位と競合の解決 複数のルールが同様の状況に適用される場合、システムはルールの特異性、組織の優先度レベル、およびコンテキストの関連性に基づいて優先度を適用します。このアプローチでは、複雑なシナリオの柔軟性を維持しながら、一貫した動作を保証します。

行動カテゴリ

通信標準

プロフェッショナルコミュニケーション さまざまな組織コンテキストやユーザー関係に応じて、適切な口調、言語の形式性、および相互作用パターンを定義するルール。

ブランドのアライメント アシスタントの応答が、すべてのインタラクションにわたって組織の声、価値観、およびコミュニケーション標準を反映することを保証する行動ガイドライン。

文化的敏感度 文化の違いと組織のインクルージョンポリシーを尊重しながら、多様なユーザー集団間での適切なコミュニケーションのための指示。

コンプライアンスとガバナンス

規制コンプライアンス データプライバシー、金融サービス規制、医療コンプライアンス基準を含む業界固有の規制要件を適用するルール。

情報ガバナンス 適切な情報共有、ソース帰属要件、および機密保護対策を定義する動作制約。

エスカレーションプロトコル 複雑さ、機密性、または規制要件に基づいて、アシスタントが人間のオペレーターに会話をエスカレーションする時期と方法を定義する指示。

オペレーションの境界

機能の制限 アシスタントの機能と制限の明確な定義、適切な期待設定とサービス境界通信を保証します。

エラー処理 ユーザーの信頼とサービス品質を維持しながら、システム障害、情報ギャップ、および予期せぬ状況を管理するための行動ガイドライン。

セキュリティプロトコル セキュリティ関連の問い合わせ、認証手順、および機密情報の保護対策の適切な処理を定義するルール。

実装パターン

カスタマーサービスアプリケーション

顧客対応アシスタントは、組織のブランド標準と規制コンプライアンス要件を維持しながら、一貫したサービスの提供を保証する包括的な行動フレームワークを実装します。

サービスエクセレンスルール

- すべてのインタラクションタイプにわたるプロフェッショナルなトーンのメンテナンス
- プロアクティブな支援の提供と問題解決のアプローチ
- 機能と制限の明確なコミュニケーション
- 適切なエスカレーショントリガーの認識と実行

内部ナレッジシステム

Enterprise Knowledge Assistant は、生産性とユーザーエクスペリエンス標準を維持しながら、情報アクセシビリティと組織のセキュリティ要件のバランスを保つ動作ルールセットを利用します。

ナレッジガバナンスルール

- すべての事実に基づく主張の出典帰属要件
- 適切な情報の分類とアクセス制御の尊重
- 情報の信頼性が低い場合の明確な不確実性コミュニケーション
- 監査およびコンプライアンスを目的とした推論プロセスの文書

特殊なドメインアプリケーション

業界固有のアシスタントは、技術的な精度と、適切な専門的な通信標準および規制コンプライアンス要件を組み合わせたドメインの専門知識の動作パターンを実装します。

ドメイン専門知識ルール

- ユーザーの専門知識レベルに応じた専門用語の使用
- 複雑な技術または規制情報の精度の免責事項
- 適切な制限ステートメントを介した専門的責任保護
- ガイダンスと推奨事項における業界のベストプラクティスの統合

構成フレームワーク

ルールセットの開発

ルール作成プロセス効果的なルールセットには、明確性と強制可能性を維持しながら、組織の要件、ユーザーの期待、および操作上の制約を考慮した体系的な開発が必要です。

開発に関する考慮事項

- 明確で具体的な指示言語、あいまいまたは矛盾するガイダンスを回避
- 適切なスコープ定義により、広すぎるまたは指定されていない動作制約が防止されます
- 組織の連携により、ルールが実際のビジネス要件と値を反映するように保証
- ルールの有効性とコンプライアンスの検証を可能にするテスト容易性の考慮事項

ルール管理

バージョン管理ルールセットには、制御された更新、ロールバック機能、および動作変更の履歴追跡を有効にする体系的なバージョン管理が必要です。

変更管理ルールの変更は、テスト、承認ワークフロー、展開調整を含む確立された変更制御手順に従って、運用の中断を最小限に抑えます。

品質保証

動作検証

一貫性テスト体系的な評価手順は、多様な会話シナリオにわたってルールセットの有効性を検証し、さまざまな条件の下で一貫した動作の適用を保証します。

コンプライアンス検証組織のコンプライアンス要件と規制基準を満たす際のルールの遵守と有効性の定期的な評価。

パフォーマンス評価

ユーザーエクスペリエンスへのインパクトユーザーの満足度、タスク完了率、および全体的なサービスの有効性に対する動作ルールのインパクトをモニタリングして、最適化の決定を通知します。

操作効率システムパフォーマンス、応答時間、およびリソース使用率に与えるルールセットの影響を評価して、動作制御と操作要件のバランスをとります。

運用上の考慮事項

ルールのメンテナンス

進行中のレビュールールの有効性、関連性、および組織の調整を定期的に評価して、継続的な適用性と価値を保証します。

パフォーマンスモニタリングルール適用の有効性、ユーザーフィードバック、およびコンプライアンス結果を体系的に追跡して、継続的な改善の取り組みをサポートします。

組織統合

ステークホルダーの調整法務、コンプライアンス、ビジネスのチームと調整して、ルールセットと組織のポリシーおよび規制要件との調整を保証します。

トレーニング統合アシスタントの行動基準をヒューマンエージェントのトレーニングおよび組織のサービス標準と調整して、サービスチャネル全体で一貫性を維持します。

開発リソース

作成ガイドライン

実装手順

- [Create a Ruleset in Gen AI Builder](#) 包括的なルールセットの開発と展開の手順
- [How-To View and Manage Threads in Gen AI Builder in Hybrid Manager](#) ルールセットの有効性評価の検証方法

統合ドキュメント

コンポーネントの統合

- [Assistants Reference Manual](#) ルールセットの割り当てと動作統合プロシージャ
- [Structures Reference Manual](#) 複雑なワークフロー内の動作ガバナンス

AI ファクトリーインフラストラクチャ

- [Gen AI Builder Overview](#) プラットフォームアーキテクチャとコンポーネントの関係
- [Hybrid Manager Integration](#) インフラストラクチャの管理と展開の手順

SDK リファレンス

- [Rulesets in Structures](#)
- [Ruleset Drivers](#)

ルールセットは、AI システムを技術的に有能なツールから、すべてのユーザーインタラクションにわたって制度の価値、コンプライアンス要件、およびサービス標準を一貫して反映する組織的に調整されたアプリケーションに変換する重要な動作ガバナンスを提供します。

Structures Reference Manual

Structures は、マルチステップ AI 処理パイプライン、外部システム統合、および Gen AI Builder 内でのカスタムビジネスロジックの実行を可能にする高度なワークフローオーケストレーション機能を提供します。これらは、単純な会話型 AI を、複雑な推論、データ変換、システムの自動化が可能な洗練されたエンタープライズアプリケーションに変換します。

アーキテクチャの目的

ストラクチャーは、複数の処理ステップ、外部サービス統合、カスタムロジック実装を必要とする複雑な AI ワークフローのオーケストレーションエンジンとして機能します。これらは、基本的な取得拡張生成を超えて、高度な推論機能と運用統合要件を備えたエンタープライズグレードのアプリケーションを有効にします。

システムインテグレーションフレームワーク

Gen AI Builder コンポーネント

- アシスタント複雑なタスクを実行するためにツールインターフェイスを介して構造を呼び出します
- データソースコンテンツ変換とエンリッチメントパイプラインの構造を利用します
- ナレッジベース カスタム取得ロジックとコンテンツ処理の構造を活用
- ツールマルチステップ機能を備えた洗練されたツールとして構造を実装します。

外部システムの接続

- API 統合 マルチプルのエンドポイントにわたる RESTful および GraphQL サービスオーケストレーション
- データベースオペレーション 複数のデータソースにまたがる複雑なデータ処理ワークフロー
- ビジネスシステム エンタープライズリソースプランニング、顧客関係管理、および金融システム統合
- ワークフローの自動化 多様な組織システムにわたるプロセスオーケストレーション

テクニカルアーキテクチャ

構造実装フレームワーク

ストラクチャーは、Griptape フレームワークを使用して洗練されたワークフローパターンを実装し、Gen AI Builder オーケストレーション要件との互換性を維持しながら、複雑なマルチステップ操作のための標準化されたインターフェイスを提供します。

実装基盤 ストラクチャーは、Gen AI Builder 展開用に最適化された統合パターンを使用して Griptape Python フレームワークを使用して開発されています。このアーキテクチャは、シークエンシャルパイプライン、並列処理、ビジネスロジック要件に基づいた条件付き分岐などのさまざまなワークフロータイプをサポートしています。

コア構造カテゴリ

Structure Type	Primary Function	Use Cases
Agent Structures	Autonomous decision-making and task execution	Complex problem-solving, multi-step reasoning
Pipeline Structures	Sequential data transformation workflows	Content processing, data enrichment

次のページに続く

表 15 – 前のページからの続き

Structure Type	Primary Function	Use Cases
Workflow Structures	Conditional branching and parallel processing	Business process automation, approval workflows

実行アーキテクチャ

構造体呼び出しパターン

1. **直接実行** コンソールインターフェイスを介した手動構造の呼び出し
2. **API 統合** RESTful API エンドポイントを介したプログラムによる実行
3. **アシスタント統合** アシスタントツールインターフェイスを介した構造呼び出し
4. **パイプライン統合** データソース処理ワークフロー内の自動実行
5. **イベント駆動実行** システムイベントまたはスケジュールに基づいてトリガーされた実行

処理調整 構造体は、言語モデル、外部 API、データソース、ビジネスロジックプロセッサを含む複数のコンポーネントの体系的なオーケストレーションを介して、複雑な操作を調整します。この調整により、状態の一貫性が維持され、洗練されたワークフローパターンが可能になります。

構造カテゴリ

エージェントベースの構造

自律推論エージェント 複数の情報ソースと高度な推論機能を組み合わせて、状況の理解と戦略的思考を必要とする多面的な問題を解決する複雑な意思決定構造。

機能

- マルチソース情報の統合と分析
- 問題の特性に基づいた動的なツールの選択
- フィードバックを取り入れた反復問題解決
- 拡張されたインタラクションにわたるコンテキストを意識した意思決定

データ処理パイプライン

コンテンツ変換構造 生のコンテンツをナレッジベースの統合とセマンティック検索操作に適した豊富な構造化情報に変換する体系的なデータ処理ワークフロー。

処理ステージ

- **コンテンツ抽出** フォーマット固有の解析とコンテンツ識別
- **機能強化** メタデータの追加、分類、および状況依存のアノテーション
- **Transformation** フォーマット変換、構造の最適化、および品質の向上
- **検証** コンテンツの精度検証と完全性評価

統合ワークフロー

外部システムオーケストレーションデータの一貫性、エラー処理、トランザクションの完全性を維持しながら、複数の外部システムにわたる操作を調整する複雑な統合構造。

統合パターン

- **API Orchestration** 依存関係管理を使用したマルチサービスの調整
- **データの同期** システム間のデータの一貫性の維持
- **プロセスオートメーション** エンドツーエンドのビジネスプロセスの実装
- **Error Recovery** 洗練されたエラー処理と再試行メカニズム

実装パターン

エンタープライズデータ処理

組織は、データの品質とガバナンス標準を維持しながら、多様な情報ソースを統合された検索可能なナレッジリポジトリに変換する包括的なデータ処理構造を実装します。

アーキテクチャコンポーネント

- **ソースアダプタ** システム固有のデータの抽出と正規化
- **処理エンジン** コンテンツの変換とエンリッチメントのワークフロー
- **品質保証** 検証と検証手順
- **出力の最適化** ダウンストリーム使用のためのフォーマットと構造の最適化

ビジネスプロセスオートメーション

構造により、適切な監視と制御メカニズムを維持しながら、人間の意思決定とシステム自動化を組み合わせた高度なビジネスプロセスの自動化が可能になります。

自動化機能

- **承認ワークフロー** ルーティングロジックを使用したマルチステークホルダーの意思決定プロセス
- **コンプライアンス検証** 自動化されたポリシー遵守のチェックと検証
- **例外処理** インテリジェントな例外の識別と解決のワークフロー
- **監査証拠の生成** 包括的なアクティビティログとコンプライアンス文書

カスタム検索システム

高度な検索構造は、ユーザーの特性と組織の要件に基づいて、意味的な類似性とビジネスロジック、状況依存フィルタリング、および動的な結果の最適化を組み合わせた高度な検索戦略を実装します。

取得強化機能

- **ハイブリッド検索** ベクトルの類似性とルールベースのフィルタリングを組み合わせ
- **コンテキスト適応** ユーザーのコンテキストと権限に基づいた検索戦略の調整

- **結果の最適化**関連性の基準に基づいた動的なランキングとフィルタリング
- **パフォーマンスの最適化**応答時間を向上させるためのキャッシュとインデックス作成戦略

構成フレームワーク

構造定義

構造には、セキュリティ境界とパフォーマンスの最適化を維持しながら、処理ロジック、外部システム接続、および操作パラメーターを定義する包括的な構成が必要です。

構成コンポーネント

展開構成

構造の展開には、適切なセキュリティ制御を維持しながら、開発、ステージング、および実稼働環境全体で一貫した動作を保証する体系的な構成管理が必要です。

展開パラメーター

- **リソースの割り当て** CPU、メモリ、および処理時間の制限
- **セキュリティ構成** 認証、承認、およびデータアクセス制御
- **パフォーマンスの最適化** キャッシュ戦略、並列処理、およびリソース使用率
- **モニタリング統合** ロギング、メトリック収集、およびパフォーマンス追跡

品質保証

パフォーマンスモニタリング

操作メトリックストラクチャは、さまざまな実行シナリオにわたってパフォーマンス特性、リソース使用率、および操作の有効性を追跡する包括的なモニタリングを実装します。

重要なパフォーマンス指標

- **実行レイテンシー** さまざまなワークフローの複雑さレベルでの処理時間の測定
- **成功率** さまざまな構造タイプと構成の完了率
- **リソース使用率** CPU、メモリ、およびネットワークリソースの消費パターン
- **エラーパターン** 障害モードと復旧効果の系統的な分析

信頼性フレームワーク

エラー処理とリカバリ洗練されたエラー処理メカニズムにより、データの整合性と操作の継続性を維持しながら、さまざまな障害シナリオからの正常な低下と復旧が保証されます。

信頼性機能

- **再試行メカニズム** 指数関数的バックオフとサーキットブレーカーパターンを使用したインテリジェントな再試行戦略
- **フォールバックプロシージャ** 重要な操作の代替処理パス

- **状態管理** トランザクションの一貫性とロールバック機能
- **モニタリング統合** リアルタイムエラー検出および警告システム

運用上の考慮事項

展開管理

構造のライフサイクル

- **バージョン管理** 下位互換性を考慮した体系的なバージョン管理
- **展開の自動化** 環境全体で一貫した展開手順
- **ロールバック機能** 問題のある展開の自動ロールバックメカニズム
- **ドキュメントメンテナンス** 構造の進化に合わせた包括的なドキュメント

メンテナンス手順

進行中のオペレーション

- **パフォーマンスの最適化** 使用パターンとフィードバックに基づいた継続的なパフォーマンスチューニング
- **セキュリティの更新** 定期的なセキュリティ評価と脆弱性の修正
- **Dependency Management** 外部システムの接続性の監視とメンテナンス
- **キャパシティプランニング** 成長予測に基づいたリソース割り当ての最適化

開発フレームワーク

実装ガイドライン

構造開発は、多様な組織要件にわたって一貫性、保守性、および運用の信頼性を保証する体系的な手順に従います。

開発のベストプラクティス

- **モジュールデザイン** コンポーネントの分離により、独立したテストとメンテナンスが可能になります
- **エラー処理** 適切なユーザーフィードバックを備えた包括的なエラー管理
- **セキュリティ統合** 組織のポリシーに合わせたビルトインのセキュリティ制御
- **パフォーマンスの最適化** リソースの効率的な使用とスケーラビリティの考慮事項

テスト戦略

検証フレームワーク

- **単体テスト** 個々のコンポーネントの機能の検証
- **統合テスト** システム境界を超えたエンドツーエンドのワークフロー検証
- **パフォーマンステスト** 予想される動作条件での負荷およびストレステスト

- [セキュリティテスト](#) 脆弱性評価と侵入テストの手順

実装リソース

開発ドキュメント

作成プロシージャ

- [Create a Structure in Gen AI Builder](#) 包括的な構造の開発および展開の手順
- [Structure Configuration](#) 詳細な構成オプションとベストプラクティス
- [Structure Execution](#) 展開された構造のテストと検証手順

統合リソース

Gen AI Builder コンポーネント

- [Assistants Reference Manual](#) 会話インターフェイス内の構造の使用
- [Tools Reference Manual](#) 洗練されたツールとしての構造実装
- [Libraries Ω Manual](#) コンテンツ処理ワークフローでの構造の使用

AI ファクトリーインフラストラクチャ

- [Model Serving integration](#) 構造体操作の言語モデルの接続
- [Hybrid Manager Deployment](#) インフラストラクチャの統合とリソース管理

SDK リファレンス

- [Structures Reference Manual](#)
- [Tasks](#) 、 [Pipelines and data preparation](#) 、 [Structures \(workflows and building blocks\)](#)
- [Tools Reference Manual](#)

ストラクチャーを使用すると、組織は、エンタープライズ運用要件をサポートする管理されたスケーラブルなインフラストラクチャ環境内で推論、統合、および自動化機能を組み合わせた高度な AI 利用ワークフローを実装できます。

Tools Reference Manual

ツールリファレンスマニュアル

ツールは、パッシブ AI アシスタントを、外部サービスとの対話、計算、カスタムビジネスロジックの実行ができるアクティブシステムに変換する拡張可能なアクション機能を提供します。これらは、会話型 AI インターフェイスと組織インフラストラクチャ内のオペレーションシステム間のブリッジとして機能します。

アーキテクチャの目的

ツールは、外部システム統合のための構造化インターフェイスを提供することにより、テキスト生成を超えて言語モデル機能を拡張します。これらにより、アシスタントは、セキュリティ境界と運用ガバナンスを維持しながら、データの取得、システム操作、ワークフローの自動化を含む具体的なアクションを実行できます。

システムインテグレーションフレームワーク

Gen AI Builder コンポーネント

- **アシスタント** 言語モデルの意思決定プロセスを介してツールを呼び出します
- **構造** パイプライン操作内のワークフローコンポーネントとしてツールを組み込みます
- **ルールセット** ツール使用ポリシーと操作上の制約を定義します
- **メモリシステム** マルチステップツール実行シーケンスのコンテキストを維持します

外部システムの接続

- **API** データ交換のための RESTful および GraphQL サービスの統合
- **データベース** 運用データにアクセスするための直接データベース接続
- **サービス** 認証接続を介した内部システム統合
- **ワークフロー** マルチツールオーケストレーションを介したビジネスプロセスの自動化

テクニカルアーキテクチャ

ツール実装フレームワーク

ツールは、さまざまなアシスタントタイプと操作コンテキストにわたって一貫した統合を可能にする標準化されたインターフェイスを実装します。このフレームワークは、単純な単一目的ツールと複雑なマルチステップのワークフローの自動化の両方をサポートしています。

実装リファレンス ツールは、[EnterpriseDB/pgai-ai-builder](#) で入手可能な pgai-ai-builder SDK フレームワークを使用して開発されています。この SDK は、ツール開発のための基本的なクラスとユーティリティを提供します。

実行アーキテクチャ

ツール呼び出しフロー

1. **決定フェーズ** 言語モデルは、会話のコンテキストに基づいてツールの必要性を判断します
2. **パラメーター抽出** ユーザー入力と会話履歴からの関連情報の抽出
3. **セキュリティ検証** 権限検証とレート制限の適用
4. **実行フェーズ** エラー処理とタイムアウト管理を使用したツール固有のロジック実行
5. **結果の統合** 出力フォーマットと会話フローへの組み込み
6. **コンテキストの更新** ツールの実行結果を反映するメモリシステムの更新

ツールカテゴリ

システム統合ツール

API コネクタ REST、GraphQL、サービス固有のプロトコルを含むさまざまな API タイプの標準化されたインターフェイスを提供する外部サービス統合ツール。

一般的な実装

- **CRM 統合** 顧客関係管理システムの接続
- **ERP システム** エンタープライズリソースプランニングのデータアクセス
- **Payment Processor** 金融トランザクション処理機能
- **Communication Services** 電子メール、SMS、および通知システムの統合

データベースアクセスツール セキュリティ境界とパフォーマンスの最適化を維持しながら、構造化データのクエリを有効にする直接データベース接続ツール。

サポートされているデータベースタイプ

- **PostgreSQL Vector Engine** 機能とのネイティブ統合
- **クラウドデータベース** AWS RDS、Google Cloud SQL、Azure Database
- **データウェアハウス** Snowflake、BigQuery、Redshift
- **NoSQL システム** MongoDB、Redis、Elasticsearch

コンピューティングツール

数学的操作 統計分析、財務モデリング、科学計算など、言語モデルの機能を超えた計算を実行する専用の計算ツール。

ビジネスロジックプロセッサ 価格計算、資格判断、コンプライアンス検証を含む組織ロジックをエンコードするカスタムビジネスルール実装ツール。

データ変換ツール データ形式の操作、集計、および業務に必要な複雑な変換を実行するコンテンツ処理ツール。

ワークフロー自動化ツール

プロセスオーケストレーション 承認ワークフロー、データ同期、システム統合シーケンスを含む複雑なビジネスプロセスを管理するマルチステップのワークフロー調整ツール。

通知システム 複数のチャネルとシステムにわたってアラートの配信、ステータスの更新、および関係者コミュニケーションを管理する通信調整ツール。

実装パターン

エンタープライズシステムインテグレーション

組織は、セキュリティ境界とパフォーマンス要件を維持しながら、AI アシスタントを既存の業務システムに接続する包括的なツールスイートを実装しています。

セキュリティフレームワーク

- **認証統合** 組織 ID システムの接続
- **権限境界** ツール操作のロールベースのアクセス制御
- **監査ログ** コンプライアンス要件の包括的な操作追跡
- **レート制限** 構成可能な使用制限を介したリソース保護

カスタマーサービスオートメーション

顧客対応アプリケーションは、顧客情報、注文ステータス、サポートシステム統合へのリアルタイムアクセスを提供する専用ツールを活用します。

一般的なツールスイート

- **Account Lookup** CRM システムからの顧客情報の取得
- **注文管理** 注文ステータスの照会と変更機能
- **支払い処理** 安全な金融取引の処理
- **チケット管理** サポートケースの作成とステータスの更新

社内の生産性の向上

ナレッジワーカーアプリケーションは、ドキュメント管理、プロジェクト追跡、通信プラットフォームを含む生産性システムと統合するツールを利用します。

生産性統合ツール

- **ドキュメント取得** エンタープライズコンテンツ管理システムへのアクセス
- **カレンダーの統合** スケジュール管理と会議の調整
- **プロジェクト管理** プロジェクトシステム全体のタスク作成とステータス追跡
- **Analytics Access** ビジネスインテリジェンスとレポートシステムの統合

構成管理

ツール構成スキーマ

ツールには、操作パラメーター、セキュリティ境界、および統合要件を定義する体系的な構成が必要です。

環境マネジメント

ツールの展開には、運用の柔軟性とセキュリティ要件のバランスをとった体系的な構成管理が必要です。

品質保証

パフォーマンスモニタリング

オペレーションメトリック

- **実行レイテンシー** ツールオペレーションの完了時間の測定

- **成功率** さまざまなツールタイプの操作の成功率
- **エラーパターン** 故障モードとエラー条件の系統的な分析
- **リソース使用率** CPU、メモリ、およびネットワークリソースの消費追跡

セキュリティ検証

セキュリティ評価フレームワーク

- **アクセス制御検証** 権限境界のテストと検証
- **認証テスト** セキュリティ資格情報の検証とローテーション手順
- **データ保護** 機密情報の取り扱いの検証と監査手順
- **Compliance Validation** 規制要件の遵守検証

開発フレームワーク

SDK 統合

ツール開発では、以下を提供する pgai-ai-builder SDK を利用します。

- **基本クラス** 標準化されたツールインターフェイスと結果構造
- **ユーティリティファンクション** API 統合とデータ処理の一般的な操作
- **構成管理** 環境変数の処理とセキュリティ検証
- **テストフレームワーク** ツール検証のための単体テストユーティリティ

SDK ドキュメント 完全な実装ガイダンスは [pgai-ai-builder repository](#) で入手できます。

導入プロセス

ツールパッケージ展開可能なコンポーネントとしてのツールパッケージ。以下を含みます。

- pgai-ai-builder SDK を使用した実装コード
- パラメーターと要件を定義する構成スキーマ
- 使用例やセキュリティ上の考慮事項を含むドキュメント
- 機能とエラー処理を検証するテストケース

運用上の考慮事項

展開管理

ツールのライフサイクル

- **バージョン管理** 下位互換性管理を使用した体系的なツールのバージョン管理
- **展開の自動化** 環境全体で一貫したツール展開手順
- **ロールバックプロシージャ** 問題のある展開の自動ロールバック機能

- ドキュメントメンテナンスツールの進化に合わせた包括的なドキュメントの更新

メンテナンス手順

進行中のオペレーション

- **Dependency Management** 外部システムの接続性の監視とメンテナンス
- **パフォーマンスの最適化** 使用パターンに基づいた継続的なパフォーマンスチューニング
- **セキュリティの更新** 定期的なセキュリティ評価と脆弱性の修正
- **キャパシティプランニング** 使用量の増加に基づいたリソース割り当ての最適化

実装リソース

開発ガイド

特定の実装手順は、専用のハウツーガイドでカバーされています。

- *Create a Tool in Gen AI Builder* pgai-ai-builder SDK を使用したステップバイステップツール開発
- Tool Configuration File 構成スキーマと展開プロシージャ
- *Run Tool* 実行テストと検証プロシージャ

参考資料

SDK ドキュメント

- pgai-ai-builder SDK 完全な SDK ドキュメントと例
- Tool API Reference ツールインターフェイスの技術仕様

統合ドキュメント

- *Assistants Reference Manual* 会話インターフェイス内のツールの使用率
- *Structures Reference Manual* 自動プロセスへのツールの組み込み

ツールは、Gen AI Builder エコシステム内での外部システム統合とカスタムロジック実行のための安全でスケーラブルなインターフェイスを提供することにより、パッシブ AI システムを組織ワークフローのアクティブな参加者に変えます。

Threads Reference Manual

Threads リファレンスマニュアル

スレッドは、Gen AI Builder 内の包括的な会話管理機能を提供し、永続的な対話履歴、コンテキストの保存、および AI アシスタント展開の操作監視を維持します。これらは、実稼働 AI アプリケーション全体の会話の継続性、品質保証、およびコンプライアンスモニタリングの基盤として機能します。

アーキテクチャファンクション

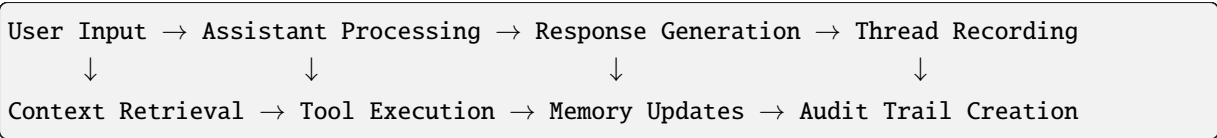
スレッドは、関連するメタデータ、システムコンテキスト、および操作の詳細を維持しながら、ユーザーとアシスタント間の完全な対話シーケンスをキャプチャする会話永続レイヤーとして動作します。この永続性により、エンタープライズ AI 展開の継続的な会話エクスペリエンス、体系的な品質評価、および包括的な監査機能が可能になります。

システムインテグレーションフレームワーク

Gen AI Builder の統合 スレッドは、Gen AI Builder エコシステム内のすべての会話コンポーネントとシームレスに統合します。

- **アシスタントマルチターンのインタラクション全体で会話の継続性を維持します**
- **メモリシステム会話コンテキストと学習した設定のための永続ストレージを提供します**
- **ツール ツールの実行の詳細と統合結果を記録します**
- **ルールセットインタラクション全体で動作のコンプライアンスとポリシーの遵守を追跡します**

会話処理パイプライン



この統合により、システムパフォーマンスとユーザーエクスペリエンスの品質を維持しながら、すべてのインタラクション要素の包括的なキャプチャが保証されます。

コアアーキテクチャ

スレッド構造

会話要素各スレッドは、インタラクションコンポーネントの包括的なレコードを維持します。

Component	Description	Purpose
Message Sequence	完全なユーザーアシスタントの交換履歴	Conversation continuity and context preservation
Metadata Records	タイムスタンプ、識別子、およびシステム情報	Operational tracking and performance analysis
Tool Execution Logs	外部システムの相互作用の詳細	Integration monitoring and troubleshooting
Memory State Tracking	コンテキストの進化と設定の学習	Personalization and conversation enhancement

時間的組織スレッドは、正確なタイムスタンプ情報で時系列のメッセージ組織を維持し、品質保証とコンプライアンスを目的とした正確な会話の再構成と時間的分析を可能にします。

状態管理

コンテキスト保持スレッドは、複数のインタラクションセッションにわたって会話コンテキストを維持し、ユーザーとアシスタント間の長期的な関係をサポートしながら、一貫したサービス品質とパーソナライズされたエクスペリエンスを保証します。

メモリ統合高度なメモリシステムは、スレッドデータを利用してユーザー設定、会話パターン、および操作コンテキストを学習し、適切なプライバシー境界を維持しながら将来のインタラクションを強化します。

運用機能

品質保証フレームワーク

会話分析スレッドは、完全な会話履歴への体系的なアクセスを提供し、多次元にわたる包括的な品質評価を可能にします。

- **応答精度** 事実の正しさの検証とソースの帰属
- **行動コンプライアンス** ルールセットの遵守とポリシーコンプライアンスの評価
- **ユーザーエクスペリエンス** 会話フロー、満足度、およびタスク完了の評価
- **システムパフォーマンス** 応答時間、ツール実行、および業務効率の分析

パフォーマンスモニタリング体系的なスレッド分析は、アシスタントの展開全体でのパターン識別、一般的な問題の検出、および最適化の機会の発見を介して継続的な改善をサポートします。

コンプライアンスと監査のサポート

規制コンプライアンススレッドは、金融サービス、ヘルスケア、データ保護規制を含む業界全体の規制要件をサポートする包括的な監査証跡を提供します。

文書標準完全なインタラクションレコードは、品質システム、コンプライアンスフレームワーク、および組織のポリシーで義務付けられている運用監視手順の文書要件をサポートします。

デバッグとトラブルシューティング

問題の調査スレッド分析により、次のような運用上の問題の体系的な調査が可能になります。

- **動作の異常** 予期しないアシスタントの応答またはポリシー違反の特定
- **統合障害** ツール実行問題と外部システムの接続問題の分析
- **パフォーマンスの低下** 応答遅延または品質低下パターンの調査
- **ユーザーエクスペリエンスの問題** 会話の破綻または満足度の問題の検査

根本原因分析詳細なスレッド情報は、包括的な根本原因分析をサポートし、操作上の問題の体系的な解決と再発の問題の防止を可能にします。

管理操作

スレッド構成

検索と検出 包括的な検索機能により、複数の基準を介して効率的なスレッドの特定が可能になります。

- **時間フィルタリング** 日付範囲とアクティビティ期間
- **コンテンツ検索** キーワードとトピックの識別
- **メタデータフィルタリング** アシスタントタイプ、ユーザーカテゴリ、およびインタラクションパターン
- **パフォーマンス基準** 応答品質、完了率、および満足度メトリック

分類システム 体系的なスレッド組織は、運用要件および分析目的に合わせた分類スキームを介して効率的な管理をサポートします。

ライフサイクル管理

リテンションポリシー スレッド管理は、操作要件とプライバシー義務およびストレージ最適化の考慮事項のバランスをとった構成可能なリテンションポリシーを実装します。

アーカイブ手順 体系的なアーカイブプロセスは、コンプライアンス要件を満たすためにデータの長期保存を保証しながら、適切なデータライフサイクル管理を通じてアクティブなシステムのパフォーマンスを最適化します。

アクセス制御フレームワーク

アクセス許可管理 包括的なアクセス制御により、組織の役割、プライバシー要件、および運用責任に合わせて適切なスレッドの可視性が保証されます。

監査証跡の完全性 スレッド変更追跡は、コンプライアンス要件と操作の完全性検証手順をサポートする完全な監査証跡を維持します。

実装パターン

エンタープライズ品質保証

組織は、業務効率とコンプライアンス要件を維持しながら、アシスタントの展開全体での包括的な品質管理をサポートする体系的なスレッド分析手順を実装します。

品質評価ワークフロー

- 会話品質評価のための定期的なサンプリング手順
- 規制要件に合わせた体系的なコンプライアンスチェック
- 継続的な改善の取り組みをサポートするパフォーマンス傾向分析
- ユーザーフィードバックの統合により、サービスの品質と満足度が向上

コンプライアンスモニタリング

規制対象産業は、スレッド機能を利用して、監査要件と規制監視義務をサポートする包括的なコンプライアンスモニタリングを行います。

コンプライアンスフレームワークの統合

- 自動ポリシー遵守チェックと違反検出
- 規制報告要件をサポートする包括的な文書
- 潜在的なコンプライアンス問題を特定するリスク評価手順
- 体系的な問題の解決を保証する修正追跡

ユーザーエクスペリエンスの最適化

製品チームは、スレッド分析を活用して、パターン識別と最適化の機会発見を介して体系的なユーザーエクスペリエンスを向上させます。

エクスペリエンス強化プロシージャ

- 会話フロー分析、改善の機会を特定する
- システムパフォーマンス特性とユーザー満足度の相関
- 開発の優先順位を通知する機能使用状況の追跡
- パーソナライゼーション効果測定支援システム強化

技術的な考慮事項

パフォーマンス特性

ストレージ効率 スレッドストレージの最適化は、インテリジェントな圧縮およびアーカイブ戦略を介して、包括的なデータキャプチャとシステムパフォーマンス要件のバランスをとります。

クエリパフォーマンス 検索と分析操作は、最適化されたインデックスとクエリ実行戦略を介して、大規模なスレッドコレクション全体で応答性の高いパフォーマンスを維持します。

統合要件

システム互換性 Thread システムは、ID 管理、監査システム、コンプライアンスプラットフォームを含む既存の組織インフラストラクチャとシームレスに統合します。

データエクスポート機能 包括的なエクスポート機能は、外部分析ツール、コンプライアンスシステム、および品質管理プラットフォームとの統合をサポートします。

運用のベストプラクティス

スレッド分析プロシージャ

系統的なレビュー方法

- 定期的なサンプリングスケジュールにより、包括的な品質を保証

- 一貫した評価をサポートする標準化された評価基準
- 包括的な分析記録を維持する文書手順
- 重要な問題またはポリシー違反に対処するエスカレーションプロトコル

継続的な改善の統合

- 分析結果とシステム強化手順との統合
- プロアクティブな問題防止をサポートするパターン識別
- 開発の優先順位を通知する最適化の機会ドキュメント
- 改善の有効性を示す成功メトリックの追跡

プライバシーとセキュリティ

データ保護スレッド管理は、組織のポリシーや規制要件に合わせて、データの最小化、アクセス制御、保持制限を含む包括的なプライバシー保護を実装します。

セキュリティフレームワーク包括的なセキュリティ対策は、スレッドデータの完全性と機密性を保護しながら、操作アクセス要件とコンプライアンス義務をサポートします。

実装リソース

管理手順

スレッド管理

- *How-To View and Manage Threads in Gen AI Builder in Hybrid Manager* 包括的なスレッドアクセスと分析プロシージャ
- *Threads Reference Manual* 体系的な評価手法と品質評価アプローチ

統合ドキュメント

システムの接続性

- *Assistants Reference Manual* 会話システムとのスレッド統合
- *Assistants Reference Manual* コンテキストの保存と学習機能
- *Rulesets Manual* 動作モニタリングとポリシーの適用

SDK リファレンス

- *Conversation Memory*
- Conversation Memory Drivers

スレッドは、継続的な改善とオペレーショナルエクセレンスをサポートしながら、すべての Gen AI Builder アプリケーションにわたって永続的なインタラクション、包括的な品質保証、および体系的なコンプライアンスモニタリングを可能にする重要な会話管理インフラストラクチャを提供します。

Data Lake Reference Manual

データレイクリファレンスマニュアル

データレイクは、Gen AI Builder 操作の基礎的なオブジェクトストレージインフラストラクチャを提供し、コンテンツの取り込み、ワークフローアーティファクト、システムコンポーネントの永続ストレージバックエンドとして機能します。このストレージレイヤーは、初期データソースの取り込みからプロダクションアシスタントの展開に至るまで、完全な AI Factory コンテンツパイプラインを有効にします。

Architectural Foundation

データレイクは、永続的なデータ管理を必要とするすべての Gen AI Builder 操作をサポートする中央ストレージリポジトリとして機能します。コンテンツ処理ワークフロー、コンポーネントの展開、および AI Factory エコシステム全体のオペレーションアーティファクト管理に必要なストレージインフラストラクチャを提供します。

システムインテグレーションロール

Gen AI Builder パイプライン統合データレイクは、完全なコンテンツ処理パイプライン内の重要なインフラストラクチャコンポーネントとして動作します。

Data Sources → Data Lake → Libraries → Knowledge Bases → Retrievers → Assistants

この統合により、データの永続性とアクセシビリティの要件を維持しながら、初期取り込みから運用展開までの体系的なコンテンツフローが可能になります。

コンポーネントストレージファンクション

- コンテンツステージング処理および変換ワークフロー中のコンテンツの一時ストレージ
- アーティファクト管理展開パッケージ、構成、およびシステムコンポーネント用の永続ストレージ
- ワークフローサポート 多段階処理操作のための中間ストレージ
- コンポーネントリポジトリ構造、ツール、およびカスタム実装パッケージ用のストレージ

ストレージアーキテクチャ

インフラストラクチャ要件

オブジェクトストレージの互換性データレイクには、標準のオブジェクト操作、アクセス制御メカニズム、およびコンソール統合のためのクロスオリジンリソース共有 CORS 構成をサポートする S3 互換オブジェクトストレージインフラストラクチャが必要です。

サポートされているストレージバックエンド

- 適切なバケットポリシーと IAM 権限を備えた Amazon S3
- 適切なアクセス制御とサービスアカウント構成を備えた Google Cloud Storage
- コンテナレベルの権限と認証セットアップを備えた Azure Blob Storage
- 標準の API 操作をサポートする S3 互換ストレージシステム

アクセス制御フレームワーク

権限要件 データレイク操作には、さまざまな組織のロールとシステムコンポーネントにわたってコンテンツのアップロード、取得、および管理操作を有効にする包括的な権限が必要です。

セキュリティの分離 専用のデータレイク構成は、AI Factory 操作とその他の組織のストレージ要件を適切に分離し、セキュリティの分離と業務効率の両方をサポートします。

オペレーションファンクション

コンテンツ管理

データソースの統合 データレイクは、データレイクタイプのデータソースのストレージバックエンドとして機能し、Gen AI Builder インターフェイスを介したコンテンツの直接のアップロードと管理を可能にします。

処理ワークフローのサポート コンテンツ変換ワークフローは、中間処理段階で Data Lake ストレージを利用し、データの永続性と復旧可能性を維持しながら、複雑なマルチステップの操作を可能にします。

コンポーネントの展開

Structure Storage データレイク内に保存される展開アーティファクトとしての Griptape Structures パッケージ、複雑なワークフロー実装の体系的なコンポーネント管理とバージョン管理が可能になります。

ツールリポジトリ カスタムツールと拡張機能は、運用機能に必要な展開パッケージ、構成ファイル、およびランタイムの依存関係に Data Lake ストレージを利用します。

ワークフローオーケストレーション

一時アーティファクト管理 処理ワークフローは、多段階の操作および複雑な変換プロセス中に永続的ストレージを必要とする中間アーティファクトと一時データを生成します。

状態の永続性 長時間実行される操作と複雑なワークフローは、状態管理とチェックポイント操作に Data Lake ストレージを使用して、信頼性の高い処理とリカバリ機能をサポートします。

構成要件

インフラストラクチャのセットアップ

ストレージプロビジョニング データレイク構成には、AI ワークロード要件に適した適切な容量計画、アクセス制御構成、およびパフォーマンス特性を備えた専用のオブジェクトストレージの割り当てが必要です。

ネットワーク構成 CORS 構成により、プログラムによるアクセスと自動処理ワークフローの適切なセキュリティ境界を維持しながら、ブラウザベースのコンソール操作が可能になります。

アクセス管理

資格情報要件 データレイクへのアクセスには、サポートされているすべてのユースケースでのコンテンツ管理、コンポーネントの展開、およびワークフロー操作のための十分な権限を備えた適切な認証資格情報が必要です。

アクセス制御構成 アクセス制御構成は、最小特権の原則を実装しながら、コンテンツ管理、コンポーネント展開、およびシステム機能に必要な操作を有効にする必要があります。

パフォーマンスの考慮事項

ストレージの最適化

アクセスパターンの最適化 データレイク構成では、コンテンツのアップロード頻度、処理ワークフロー要件、コンポーネントの展開パターンを含む予想されるアクセスパターンを考慮して、パフォーマンス特性を最適化する必要があります。

容量プランニング ストレージ容量要件は、コンテンツ量、コンポーネントの複雑さ、およびワークフローアーティファクト生成に応じて拡大するため、体系的な容量計画とモニタリング手順が必要です。

業務効率

コンテンツの組織 データレイクコンテンツの体系的な組織は、効率的な操作をサポートし、コンテンツ、コンポーネント、ワークフローアーティファクトの管理手順を簡素化します。

クリーンアップ手順 定期的なメンテナンス手順では、一時的なアーティファクト、古いコンポーネント、および不要なコンテンツの蓄積の体系的なクリーンアップを介して、ストレージの最適な使用率を保証します。

統合パターン

データソースの接続

直接アップロードの統合 データレイクタイプのデータソースは、直接コンテンツアップロード機能を提供し、中間処理または変換要件なしの体系的なコンテンツの取り込みを可能にします。

ワークフロー統合 複雑なデータソースのワークフローは、コンテンツのステージング、変換アーティファクト、および多段階の取り込み操作中の処理状態管理に Data Lake ストレージを利用します。

コンポーネント管理

開発ワークフローの統合 ストラクチャーとツールの開発ワークフローは、展開パッケージに Data Lake ストレージを利用し、体系的なコンポーネント管理とバージョン管理手順を有効にします。

運用展開運用 コンポーネントの展開では、ランタイムアーティファクト、構成ファイル、および信頼性の高いシステム操作をサポートする依存関係管理に Data Lake ストレージを利用します。

運用のベストプラクティス

ストレージ管理

分離戦略 専用のデータレイク構成は、AI Factory 操作とその他の組織のストレージ要件を適切に分離し、セキュリティと業務効率の両方をサポートします。

バージョン管理 保存されたコンポーネントとコンテンツの体系的なバージョン管理手順により、運用上の問題が発生した場合の信頼できる展開管理とロールバック機能が有効になります。

セキュリティフレームワーク

アクセス制御の実装 包括的なアクセス制御構成は、セキュリティ境界と操作要件を維持しながら、さまざまなユーザータイプとシステムコンポーネントに適切な権限を保証します。

監査機能 ストレージアクセスのログとモニタリング機能は、データレイク操作の包括的な可視性を介して、運用の監視とコンプライアンス要件をサポートします。

実装の依存関係

前提条件

データレイク構成は、Gen AI Builder 機能の基本的な要件を表します。ライブラリ、ナレッジベース、およびアシスタントオペレーションは、コンテンツ管理とシステム操作のために適切に構成された Data Lake インフラストラクチャに依存しています。

構成リソース

セットアップ手順

- [Configure the Data Lake in Gen AI Builder](#) オブジェクトストレージ統合の包括的なセットアップ手順
- [Configure Data Sources in Gen AI Builder](#) コンテンツ取り込みワークフローとのデータレイクの統合

統合ドキュメント

- [Libraries Ω Manual](#) コンテンツ管理ワークフローの統合
- [Structures Reference Manual](#) 構造とツールの展開プロシージャ

データレイクは、Gen AI Builder 内のすべてのコンテンツ管理、コンポーネント展開、およびワークフローオーケストレーション機能を有効にする重要なインフラストラクチャを提供し、完全な AI Factory エコシステムの基本的なストレージレイヤーとして機能します。

Libraries Ω Manual

ライブラリは、Gen AI Builder 内の知識駆動型 AI アプリケーションの基礎的なデータ管理インフラストラクチャを提供します。これらは、多様なデータソースを体系的に整理、処理、最適化して、インテリジェントアシスタントと自動システムを強化する検索拡張生成ワークフローをサポートします。

アーキテクチャファンクション

ライブラリは、生の情報ソースとインテリジェントアプリケーション間のデータオーケストレーションレイヤーとして機能します。これらは、さまざまな起源の非構造化コンテンツを、アシスタントやその他の AI コンポーネントが正確な情報取得のために確実にアクセスできる、検索可能な意味論的にインデックスされた知識リソースに変換します。

データ処理パイプライン

図書館システムは、構造化データ変換ワークフローを実装します。

Data Sources → Content Ingestion → Processing & Indexing → Knowledge Bases → Retrieval Interface

このアーキテクチャでは、多様なコンテンツタイプと組織の要件にわたって業務効率を維持しながら、一貫したデータ品質を保証します。

システムインテグレーション

Gen AI Builder コンポーネント

- ナレッジベースセマンティックインデックスとメタデータエンリッチメントを使用して処理されたコンテンツを受信します
- レトリバーライブラリ管理コンテンツ全体にわたって最適化された検索操作を実行します
- アシスタント標準化された検索インターフェイスを介して厳選された知識にアクセスします
- データレイク AI アクセラレーターパイプラインを介して大規模なデータ処理ワークフローと統合

AI ファクトリーインフラストラクチャ

- **Vector Engine** Postgres インフラストラクチャ内で高性能のセマンティック検索機能を提供します
- **AI Accelerator Pipelines** 自動コンテンツ処理とナレッジベースのメンテナンスを可能にします
- **モデル提供**セマンティックインデックス操作のための埋め込みモデルを提供します

コアコンポーネント

データソースの統合

ライブラリは、包括的なコンテンツアクセス機能を維持しながら、ソース固有の複雑さを抽象化する標準化されたインターフェイスを介して多様なデータリポジトリに接続します。

サポートされているソースタイプ

Source Type	Integration Method	Content Processing
Web Pages	URL crawling and scraping	HTML parsing with content extraction
Document Repositories	API integration or file system access	Format-specific processing (PDF, DOCX, etc.)
Database Systems	Direct database connectivity	Query-based content extraction
Cloud Storage	Object storage APIs (S3, GCS, Azure)	Batch processing with meta-data preservation
Collaborative Platforms	API-based integration (Confluence, SharePoint)	Structured content extraction with hierarchy

次のページに続く

表 17 – 前のページからの続き

Source Type	Integration Method	Content Processing
-------------	--------------------	--------------------

コンテンツ処理フレームワーク

データの取り込みは、体系的なコンテンツ変換手順を実装します。

1. **フォーマット検出** コンテンツタイプと構造の自動識別
2. **コンテンツ抽出** 意味的な意味を保持するフォーマット固有の解析
3. **メタデータのエンリッチメント** ソース属性、タイムスタンプ、分類タグを含むコンテキスト情報の追加
4. **品質検証** コンテンツの精度検証と完全性チェック
5. **セマンティックインデックス類似度ベースの検索操作のベクトル埋め込み生成**

ナレッジベースアーキテクチャ

ナレッジベースは、処理されたライブラリコンテンツと検索システム間の操作インターフェイスを提供します。これらは、検索パフォーマンスとリソース効率のバランスを保つ、最適化されたストレージとインデックス作成戦略を実装します。

ストレージの最適化

- Vector Engine 機能を使用して Postgres インフラストラクチャ内に保存されたベクター埋め込み
- 取得精度とモデルコンテキストウィンドウのために最適化されたコンテンツチャンク戦略
- 文書構造と関係を保持する階層インデックス
- フィルタリングされた検索操作とアクセス制御適用のためのメタデータのインデックス作成

検索機能

- ベクトル埋め込みを使用した意味的類似性検索
- ベクトルの類似性とキーワード一致を組み合わせたハイブリッド検索
- メタデータベースのコンテンツ選択をサポートするフィルタリングされた検索
- 特定のクエリタイプの結果の関連性を最適化する再ランキングアルゴリズム

取得インターフェイス

レトリバーは、ナレッジアクセスの操作パラメーターを定義し、さまざまなアプリケーション要件とコンテンツの特性に最適化する構成可能な検索戦略を提供します。

構成パラメーター

Parameter	Purpose	Typical Range
Similarity Threshold	Controls result relevance filtering	0.6 - 0.9

次のページに続く

表 18 – 前のページからの続き

Parameter	Purpose	Typical Range
Result Limit (Top-K)	Defines maximum results returned	3 - 20
Context Window	Manages content length for language models	1000 - 8000 tokens
Reranking Strategy	Optimizes result ordering for specific use cases	None, semantic, hybrid

高度な取得戦略

- 包括的な情報カバレッジのためのマルチクエリーの拡張
- ユーザー権限とデータ分類に基づいたコンテキスト依存フィルタリング
- クエリ特性に基づいた動的なしきい値調整
- ソース帰属の引用と出自の追跡

実装パターン

エンタープライズナレッジマネジメント

組織は、一元化された図書館インフラストラクチャを介して部門やビジネス機能を超えて情報を統合する包括的なナレッジマネジメントシステムを実装します。

アーキテクチャの特徴

- 多様な組織システムにわたるマルチソースの統合
- 組織階層に合わせたロールベースのアクセス制御
- ソースシステムの変更と同期した自動コンテンツ更新
- 規制要件の遵守を保証するコンプライアンスフレームワーク

一般的な実装

```
# Library Configuration Example

library_config:
  name: "enterprise_knowledge"
  data_sources:
    - type: "confluence"
      connection: "corporate_confluence"
      spaces: ["hr_policies", "it_procedures", "finance_guidelines"]
    - type: "sharepoint"
      connection: "corporate_sharepoint"
      sites: ["project_documentation", "training_materials"]
  processing:
    chunk_size: 1000
    overlap: 100
    embedding_model: "organizational_embeddings"
  knowledge_base:
```

(次のページに続く)

(前のページからの続き)

```
name: "enterprise_kb"
search_strategy: "hybrid"
access_control: "rbac_enabled"
```

カスタマーサポートシステム

顧客対応アプリケーションはライブラリを活用して、製品およびポリシー情報の包括的なカバレッジを維持しながら、会話型インターフェイスを介して一貫した正確な情報配信を提供します。

デザインの考慮事項

- 顧客向け情報の内容精度検証手順
- 内部ドキュメントと外部ドキュメント間の一貫性を保証する更新の同期
- 高同時性の顧客インタラクションのパフォーマンスの最適化
- 既存のサポートシステムおよびエスカレーションワークフローとの統合

研究開発支援

技術チームはライブラリを利用して研究資料、技術文書、プロジェクト情報を集約し、開発イニシアチブ全体での生産性と知識の共有を向上させます。

特殊な機能

- コード例と図に最適化された技術文書処理
- ドキュメントの進化を追跡するためのバージョン管理の統合
- 関連素材間の関係を維持する相互参照リンク
- 開発ツールおよびワークフローと統合するためのエクスポート機能

運用管理

コンテンツのライフサイクル

取り込みプロシージャライブラリは、業務効率を維持しながらデータ品質を保証する体系的なコンテンツの取り込みを実装します。

1. **ソースモニタリング** コンテンツの変更と追加の自動検出
2. **処理キュー** さまざまなコンテンツ量を処理する管理処理ワークフロー
3. **品質保証** コンテンツの精度と完全性を保証する検証手順
4. **インデックスの更新** コンテンツの変更中に検索パフォーマンスを維持するインクリメンタルな更新

メンテナンスオペレーション

- 定期的なコンテンツ検証により、継続的な精度と関連性を保証
- インデックスの再構築やストレージの最適化を含むパフォーマンスの最適化

- アクセス制御の精度とコンプライアンス要件を維持するセキュリティ更新
- コンテンツと構成情報の両方を保存するバックアップ手順

パフォーマンスの最適化

検索パフォーマンスライブラリは、大規模なコンテンツコレクション全体での応答的な検索操作を保証する複数の最適化戦略を実装しています。

- インデックスの最適化 検索パフォーマンスを最適化する定期的なインデックスのメンテナンス
- キャッシュ戦略 頻繁にアクセスされるコンテンツと検索結果のインテリジェントなキャッシュ
- リソースの割り当て 使用パターンとパフォーマンス要件に基づいた動的なリソースのスケーリング
- クエリ最適化 検索の精度と効率を向上させる自動クエリ強化

スケーラビリティ管理

- 成長するコンテンツ量とユーザーベースをサポートするホリゾンタルスケーリング機能
- 処理リソース間のロードバランシングにより、一貫したパフォーマンスを保証
- パフォーマンスとコスト効率のバランスをとったストレージ最適化戦略
- システムパフォーマンスと容量使用率の可視性を提供する監視フレームワーク

構成フレームワーク

データソース構成

データソース接続には、包括的なコンテンツアクセスとセキュリティ要件および操作効率のバランスをとった注意深い構成が必要です。

認証と承認

```
# Example Data Source Configuration

data_source:
  type: "confluence"
  connection:
    base_url: "https://company.atlassian.net"
    authentication:
      method: "api_token"
      credentials: "service_account"
    access_scope:
      spaces: ["public_docs", "team_docs"]
      permissions: "read_only"
```

コンテンツの選択

- コンテンツ含める基準を定義するフィルター構成
- 鮮度とリソース消費のバランスをとった更新頻度設定

- 特定のコンテンツタイプと組織の要件に合わせて最適化された処理パラメーター

ナレッジベースの構成

ナレッジベースのセットアップには、コンテンツの品質とアクセシビリティを維持しながら、検索パフォーマンスを最適化する体系的な構成が必要です。

処理パラメーター

- コンテキストの保存と検索の粒度のバランスをとったチャンク戦略
- 組織のコンテンツの特性に合わせた埋め込みモデルの選択
- 関連するコンテキスト情報をキャプチャするメタデータ抽出ルール
- コンテンツの精度と完全性を保証する品質検証しきい値

取得構成

レトリバーのセットアップは、ナレッジアクセスの操作特性を定義し、検索精度とパフォーマンス要件のバランスをとった注意的な調整が必要です。

検索戦略の構成

```
# Retriever Configuration Example

retriever_config:
  name: "enterprise_retriever"
  search_strategy: "hybrid"
  parameters:
    similarity_threshold: 0.75
    max_results: 10
    reranking: "enabled"
    context_optimization: "automatic"
  access_control:
    user_filtering: "enabled"
    content_classification: "enforced"
```

品質保証

コンテンツ検証

ライブラリは、データ処理パイプライン全体でコンテンツの精度と完全性を保証する包括的な検証手順を実装します。

検証フレームワーク

- 自動および手動レビュープロセスを介したソースコンテンツの精度検証
- 変換中の意味の保持を保証する処理品質評価
- 検索結果の精度と関連性を確認する検索結果の検証

- 継続的な品質向上を可能にするユーザーフィードバックの統合

パフォーマンスモニタリング

オペレーションメトリック

- 検索の応答時間と精度の測定
- コンテンツ処理のスループットとエラー率
- システムリソース使用率とキャパシティプランニングのメトリック
- ユーザー満足度とシステム効果の測定

継続的な改善

- 最適化の機会を特定するパフォーマンス傾向分析
- コンテンツ品質評価により、的を絞った改善が可能
- 検索戦略の調整を通知するユーザー行動分析
- ピークパフォーマンスを維持するシステム最適化手順

セキュリティとコンプライアンス

アクセス制御

図書館は、業務効率とユーザーエクスペリエンス品質を維持しながら、適切な情報公開を保証する包括的なアクセス制御フレームワークを実装しています。

セキュリティフレームワーク

- 組織 ID システムとのユーザー認証の統合
- 組織データの分類に関するコンテンツレベルのアクセス制御
- 情報アクセスパターンの包括的な可視性を提供する監査ログ
- 機密情報の適切な取り扱いを保証するプライバシー制御

コンプライアンス統合

規制のコンプライアンス

- 組織および規制上の要件に合わせたデータ保持ポリシー
- 個人情報の適切な取り扱いを保証するプライバシー保護の仕組み
- コンプライアンス検証と報告をサポートする監査証跡のメンテナンス
- 情報処理の包括的な可視性を提供するデータリネージ追跡

はじめに

前提条件

ライブラリーの実装には、Gen AI Builder のインストール、コンテンツ処理のための適切なコンピューティングリソース、および組織データソースへのネットワーク接続を含む基礎インフラストラクチャが必要です。

初期構成

1. **データソース評価** 利用可能なコンテンツソースとアクセス要件を評価します
2. **処理構成** 組織のニーズに基づいてコンテンツ処理パラメーターを定義します
3. **ナレッジベースの設定** 検索戦略とパフォーマンスの最適化を構成します
4. **取得テスト** 検索精度とパフォーマンス特性を検証する

実装リソース

構成ガイド

- *Configure Data Sources in Gen AI Builder* 包括的なソース統合手順
- *Knowledge Bases Reference Manual* 検索の最適化とパフォーマンスチューニング
- *Retrievers Reference Manual* 検索戦略定義とパラメーターの最適化

統合ドキュメント

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)* エコシステムの接続パターン
- *Hybrid Manager Deployment* インフラストラクチャ統合手順

SDK リファレンス

- Embedding Drivers
- Vector Store Drivers
- Data Loaders

ライブラリは、Gen AI Builder エコシステム内の体系的なコンテンツ編成、処理最適化、およびインテリジェントな取得機能を介して知識駆動型 AI アプリケーションを有効にする基礎的なデータ管理インフラストラクチャを提供します。

Explainers

Best practices for Hybrid Knowledge Bases

これは誰に向けたものですか

ハイブリッドナレッジベースを作成して、構造化データと非構造化データを組み合わせた高度な AI アプリケーションを強化するプラットフォームユーザー。一般的なユーザーには、製品カタログ、顧客プロファイル、または混合データセットを操作する開発者、AI アーキテクト、データエンジニア、コンテンツマネージャーが含まれます。

あなたが達成すること

Gen AI Builder でハイブリッドナレッジベースをデザインするためのベストプラクティスを学びます。

- 構造化列と非構造化列を選択する方法
- 強力な AI クエリーをサポートするスキーマをデザインする方法
- ハイブリッドナレッジベースデザインでの一般的な落とし穴を回避する方法

ハイブリッドナレッジベースを注意深くデザインする理由

- ハイブリッドナレッジベースを使用すると、**構造化フィルタ**の完全一致、レンジ、ファセットと **セマンティック検索ベクトルベース**の検索を組み合わせることができます。
- 適切に設計されたハイブリッドナレッジベースにより、より正確で柔軟、説明可能な AI 応答が可能になります。
- スキーマ設計が不十分であると、非効率なクエリ、不完全な検索結果、または無駄な処理が発生する可能性があります。

背景については、以下を参照してください。

- *Knowledge Bases Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Retrieval Augmented Generation (RAG)*
- *Embeddings explained*

ハイブリッドマネージャー展開でハイブリッドナレッジベースがどのように使用されるかを確認するには、次の Web サイトにアクセスしてください。

- [Hybrid Manager: Using Gen AI Builder](#)

複雑さと完了までの時間

- **複雑さ** 中度 - スキーマデザインには、データとユースケースを理解する必要があります。
- **推定時間** ハイブリッドナレッジベースごとに 15～30 分。

ハイブリッドナレッジベースのデザイン

ユースケースを理解する

AI アプリケーションでサポートするクエリの種類を理解することから始めます。

例

- 「カテゴリ’アウトドア’にある 500 ドル未満の商品を表示し、検索用語との関連性によってランク付けます。」
- “データ保持に言及したリージョン’EU’のポリシーを検索します。”

構造化列の選択

構造化列 のサポート

- 完全一致フィルタ例 `region = 'EU'`
- レンジクエリー 例 `price BETWEEN 100 AND 500`
- ファセット検索たとえば、すべての `categories` をリストします

ベストプラクティス

- ライブラリースキーマに正確に一致する **短くて明確な列名**を使用します。
- データ型を注意深く選択します。
- 識別子とカテゴリ SKU、リージョン、カテゴリの場合は `Text`
- `Number` 数値範囲価格、評価の場合
- 日付ベースのフィルタリングの場合 `Date` 発売日、有効期限
- `Boolean` 単純な yes/no フィルタ `isActive`、`inStock`
- 自由形式のテキストフィールドを構造化列として含めないでください。非構造化列に含める必要があります。

非構造化列の選択

非構造化列 のサポート

- 全文セマンティック検索
- 検索拡張生成 RAG クエリー

ベストプラクティス

- 埋め込みからメリットを得るリッチテキストを含む列を選択します。
- 説明
- レビュー
- ポリシー
- よくある質問の回答
- 詳細な注意事項
- 冗長または簡単なテキストを含む列を避けます。これにより、検索品質が低下する可能性があります。
- **非構造化列の数** を意図的に定義します。
- 希薄な検索結果が多すぎます。
- 少なすぎると、コンテキストの豊富さが制限されます。

スキーマデザインの例

製品カタログ Hybrid KB

構造化列

- SKU テキスト
- Category テキスト
- Price 数値
- AvailableSince 日付
- InStock ブール値

非構造化列

- ProductName
- Features
- DetailedDescription

よくある落とし穴

列名の不一致

- 定義する列名は、ソースライブラリー CSV ヘッダー、Google シート列の列名と**正確に一致する必要があります**。
- 名前のスペルが間違っている場合、または大文字と小文字の違いにより、取り込みエラーが発生します。

タイプの選択が不十分

- 数値フィールドをテキストとしてマークしないでください。その逆も同様です。
- データ型を正しく使用して、適切な日付クエリをサポートします。

構造化列のオーバーロード

- 長い自由形式のテキスト列を構造化として追加しないでください。これにより、フィルタリングのパフォーマンスが低下します。

非構造化列の使用が不十分

- 非構造化テキストが複数の列にまたがっている場合 FAQ の質問、FAQ の回答など、取り込む前に構造と組み合わせることを検討するか、両方を非構造化として明確に選択します。

シナリオ例

あなたは、製品カタログのハイブリッドナレッジベースをデザインしています。

構成例

- 構造化列
- SKU テキスト
- Category テキスト
- Price 数値
- AvailableSince 日付
- 非構造化列
- ProductName
- DetailedDescription

結果のクエリーがサポートされています

- Category および Price 範囲でフィルターします。
- AvailableSince で並べ替えます。
- 製品名と説明にわたってセマンティック検索を実行します。

障害対応

不完全な検索結果

- 非構造化列が選択され、正しくインデックス付けされたことを確認します。
- クエリーの構造化フィルターが制限しすぎないことを確認します。

フィルターが機能しない

- 列が構造化されたものとしてマークされ、正しいデータ型であることを確認します。
- ライブラリースキーマと列名が正確に一致することを確認します。

インジェストエラー

- すべての列名がソースライブラリーと正確に一致することを確認します。
- 列のデータ型が目的のスキーマと一致することを確認します。

関連トピック

- *Knowledge Bases Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Retrieval Augmented Generation (RAG)*

- *Embeddings explained*
- *Structures Reference Manual*
- *Create a Knowledge Base in Gen AI Builder*
- *Manage Knowledge Bases in Gen AI Builder*
- *Hybrid Manager: Using Gen AI Builder*

How to prepare data and kb

Gen AI How-To Guides

これらのガイドを使用して、AI Factory の Gen AI 機能で AI 搭載アプリケーションを構築および管理します。

これらのハウツーガイドでは、AI エージェントアシスタントとストラクチャーの構成、外部データソースの統合、ルールセットの管理などの日常タスクをカバーしています。

どこから始めるべきか

- *Configure the Data Lake in Gen AI Builder*
- *Create a Knowledge Base in Gen AI Builder*
- *Create a Retriever in Gen AI Builder*
- *Create a Ruleset in Gen AI Builder*
- *Create a Structure in Gen AI Builder*
- *Create a Tool in Gen AI Builder*

データソースの操作

- *Configure an Atlassian Confluence data source*
- *Configure a Custom data source*
- *Configure a Data Lake data source*
- *Configure a Google Drive data source*
- *Configure an Amazon S3 data source*
- *Configure a Web Page data source*

ベストプラクティスと管理

- *Best practices for Hybrid Knowledge Bases*
- *Manage Knowledge Bases in Gen AI Builder*
- *How-To View and Manage Threads in Gen AI Builder in Hybrid Manager*

関連概念

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Structures Reference Manual*
- *Tools Reference Manual*
- *Knowledge Bases Reference Manual*
- *Retrievers Reference Manual*
- *Rulesets Manual*
- *Threads Reference Manual*

SDK リファレンス

- SDK Overview
- *Structures Reference Manual*
- *Tools Reference Manual*
- Vector databases and semantic search

Create a Knowledge Base in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ハイブリッドマネージャー AI ファクトリー の **Gen AI Builder** でナレッジベースを構成するプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、実稼働 AI アプリケーションのナレッジベースを管理するビジネスオーナーが含まれます。

あなたが達成すること

コンテンツソースとして 1 つ以上のライブラリを選択して、**ナレッジベース**を作成します。

以下を構成できます。

- フルマネージドベクターストア

- 自己管理の Postgres ベクターストア
- 構造化データと非構造化データを組み合わせたハイブリッドナレッジベース

次の目的で使用するナレッジベースを準備します。

- AI アシスタント
- RAG パイプライン
- セマンティック検索アプリケーション

ナレッジベースを使用する理由

ナレッジベースは、Gen AI アプリケーションの **照会可能な基盤**を提供します。

- パワー セマンティック検索 と 検索拡張生成 RAG。
- LLM ベースのエージェントがデータに基づいてコンテンツを取得できるようにします。
- 高度なユースケースのために、構造化メタデータをリッチな非構造化テキストと組み合わせます。
- 更新可能なパイプラインのサポート —ソースライブラリが更新されると、ナレッジベースを再同期できます。

背景については、以下を参照してください。

- *Knowledge Bases Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Retrieval Augmented Generation (RAG)*
- *Embeddings explained*
- *Structures Reference Manual*

ハイブリッドマネージャー展開でナレッジベースがどのように使用されているかを確認するには、以下にアクセスしてください。

- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 低～中程度選択したナレッジベースのタイプによって異なります。
- 推定時間 5～15 分

ナレッジベースを作成する場合

- アシスタント で使用するコンテンツを準備する場合。
- RAG パイプライン を有効にします。
- セマンティック検索エクスペリエンスを提供するため。
- 構造化フィルターと AI 駆動の検索ハイブリッド KB を組み合わせます。
- ソースコンテンツで ライブラリ を作成および検証した後。

ナレッジベースを作成する方法

1. ナレッジベースに移動します

- Gen AI Builder コンソールで、**Knowledge Bases** に移動します。

2. 新しいナレッジベースを作成する

- **Create Knowledge Base** をクリックします。

3. 共通フィールドの構成

お名前必須

- 明確で一意的な名前を指定します。
- 例 Company Policy KB

説明オプション

- ナレッジベースの目的または使用目的について説明します。

4. ナレッジベースのタイプを選択します

Gen AI Builder は、**3つのナレッジベースタイプ**をサポートしています

PG.AI-フルマネージド Vector ストア

- フルマネージドのバックエンドを使用してベクターベースのナレッジベースを作成します。
- 非構造化テキストおよび汎用セマンティック検索用に最適化されています。

設定します。

- **Libraries** 1つ以上のライブラリを選択します。
- 例 HR Policy Documents Library、Employee Handbook Library

例

- 名前 Company Policy KB
- 選択したライブラリ HR Policy Documents Library、Employee Handbook Library

PG.AI データベース-自己管理

- AI / Vector 拡張機能を使用して、独自の Postgres 互換データベースに接続します。
- ベクターストアを完全に制御したい場合に役立ちます例 HM 管理の Postgres + AIDB。

設定します。

- **接続文字列** データベースへの完全な接続文字列。
- 例 `postgresql://user:password@host:port/dbname`
- **Password** データベースパスワード。
- **Libraries** 1 つ以上のライブラリを選択します。

例

- 名前 `Tech Docs Self-Managed KB`
- 接続文字列 `postgresql://user:password@host:5432/techdocs`
- パスワード `**** **** **`
- 選択したライブラリー `API Documentation Library`

PG.AI ハイブリッドデータベース

- **構造化フィルタ** と **セマンティック検索** の両方を使用してハイブリッドナレッジベースを構築します。
- カatalog、顧客プロフィール、および多属性の検索エクスペリエンスに最適です。

設定します。

- **ライブラリ** 1 つ以上のライブラリ通常は CSV、Google シートなどの構造化データソースを選択します。
- **構造化列**
 - フィルタリングに使用する列を追加します。
 - 名前、オプションの説明、データ型テキスト、数値、日付、ブール値を定義します。
- **非構造化列**
 - セマンティック検索の埋め込みに使用される列を追加します。
 - 名前とオプションの説明を定義します。

例

- 名前 `Product Catalog Hybrid KB`
- 選択したライブラリー `Product CSV Library`
- 構造化列
 - `SKU` テキスト
 - `Price` 数値
- 非構造化列

- ProductName
- Features

5. 作成を完了する

- **Create** をクリックします。
- システムは、選択したライブラリからナレッジベースを構築および設定します。
- 進行状況は **ナレッジベース** リストビューに表示されます。

シナリオ例

あなたは、AI アシスタントでの製品検索とフィルタリングを強化するハイブリッドナレッジベースを構築したいと考えています。

構成例

- 名前 **Product Catalog Hybrid KB**
- 選択したライブラリー **Product CSV Library**
- 構造化列
- SKU テキスト
- Price 数値
- 非構造化列
- ProductName
- Features

障害対応

不足しているライブラリ

- ソースライブラリが作成され、データが設定されていることを確認します。
- ハイブリッドナレッジベースには、定義されたスキーマを備えたライブラリが必要です。

接続エラー自己管理

- データベースの接続文字列と資格情報を検証します。
- 必要な拡張機能 pgvector、AIDB がインストールされていることを確認します。
- データベースが Gen AI Builder から到達可能であることを確認します。

不完全なインデックス作成 ハイブリッド KB

- 構造化および非構造化インデックスに正しい列が選択されたことを確認します。
- 列名はライブラリースキーマと一致する必要があります。

SDK リファレンス

- Embedding Drivers — ベクトル埋め込みを生成
- Vector Store Drivers — ストア/クエリーエンベディング
- RAG Engines — アセンブル取得および応答ステージ

関連トピック

- *Knowledge Bases Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Retrieval Augmented Generation (RAG)*
- *Embeddings explained*
- *Structures Reference Manual*
- *Manage Knowledge Bases in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Manage Knowledge Bases in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

Gen AI Builder で AI アプリケーションのナレッジベースを管理するプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、データエンジニア、コンテンツマネージャーが含まれます。

あなたが達成すること

Gen AI Builder で既存のナレッジベースを管理する方法を学習します。一般的な管理タスクを実行します。

- 既存のナレッジベースの表示
- ナレッジベース設定の編集
- ナレッジベースの更新または再同期
- ナレッジベースの削除
- ナレッジベースの詳細の表示

ナレッジベースを管理する理由

- ナレッジベースを最新の状態に保つことで、AI アプリケーションが最新の正確な応答を提供できます。
- ナレッジベースの管理は、未使用または古いナレッジベースをクリーンアップすることにより、ストレージとコストの制御に役立ちます。
- ナレッジベースのステータスとコンテンツの監視により、実稼働 AI ワークフローの信頼性とパフォーマンスが向上します。

背景については、以下を参照してください。

- *Knowledge Bases Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Retrieval Augmented Generation (RAG)*

ハイブリッドマネージャー展開でナレッジベースがどのように使用されているかを確認するには、以下にアクセスしてください。

- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 低。
- 推定時間 ナレッジベースごとに 2～5 分。

既存のナレッジベースの表示

1. Gen AI Builder コンソールで **Knowledge Bases** に移動します。
2. Knowledge Bases リストビューには以下が表示されます。
 - **Name** ユーザー定義の名前例 docs-temp-hcp。
 - **Created** 作成日時。
 - **更新** 最終更新日時。
 - **Status** 動作中のステータス例 Succeeded。
 - **Type** バックエンドタイプ例 GenAI Builder Vector または vector。
3. ページネーションコントロールを使用して、大きなリスト例 Rows per page: 5 1-5 of 8 < > をナビゲートします。

ナレッジベースの詳細の表示

1. ナレッジベース名または **View Details** アイコンをクリックして、その詳細ページを開きます。
2. 詳細ビューには通常次のものが表示されます。
 - **名前** と **説明**
 - **Created** および **Updated** タイムスタンプ

- ステータス とバックエンド タイプ
- リンクされたライブラリ または データソース
- インデックス作成または同期ステータス
- 最近の活動または仕事履歴

ナレッジベースの編集

注タイプによっては、ナレッジベースの作成後の編集機能が制限される場合があります。

1. Knowledge Bases リストビューで、**Edit** アクション通常はアクションメニューまたは編集アイコンから選択します。
2. 次を編集できます。
 - 説明
 - 場合によっては、ライブラリまたはデータソースのセットタイプによって異なります
 - ハイブリッド KB の構造化/非構造化列の構成サポートされている場合
3. 変更を保存します。

ナレッジベースの更新または再同期

1. Knowledge Bases リストビューで、**Refresh** または **Re-sync** アクションを選択します。
2. システムは次のことを行います。
 - リンクされたライブラリから最新のデータを取得します。
 - 必要に応じて、ナレッジベースのインデックスを再作成します。

ソースライブラリが更新され、変更をナレッジベースに伝播する場合にこのアクションを使用します。

ナレッジベースの削除

警告ナレッジベースの削除は元に戻せません。これにより、ナレッジベースとそのコンテンツが AI アプリケーションから削除されます。

1. Knowledge Bases リストビューで、**Delete** アクション通常はアクションメニューまたは削除アイコンから選択します。
2. プロンプトが表示されたら、削除を確認します。

これにより、ソースライブラリは削除されません。ナレッジベースのみが削除されます。

障害対応

ナレッジベースのステータスに「Error」または「Failed」が表示される

- **View Details** をクリックし、アクティビティログまたはジョブ履歴を確認します。

- ナレッジベースが外部データベース自己管理 PG.AI データベースにリンクされている場合、データベース接続が正常であることを確認します。

更新操作では予想されるコンテンツが更新されません

- リンクされたライブラリが更新され、最新のデータが含まれていることを確認します。
- Refresh / Re-sync アクションを再実行します。
- ライブラリが予想されるデータソースと変換で正しく構成されていることを確認します。

特定のフィールドは編集できません

- 一部のナレッジベースタイプ特にフルマネージドベクタストアと自己管理 DB は、作成後の特定の構成フィールドの編集を制限します。
- これらのフィールドを変更するには、新しいナレッジベースを作成する必要がある場合があります。

シナリオ例

実稼働で使用するナレッジベースのステータスを更新して確認したい

1. **Knowledge Bases** に移動します。
2. **Company Policy KB** を見つけます。
3. **View Details** をクリックして、最近のライブラリが更新されたことを確認します。
4. **Refresh** をクリックして、ナレッジベースを再同期します。
5. **Succeeded** までステータスを監視します。

SDK リファレンス

- [Embedding Drivers](#)
- [Vector Store Drivers](#)
- [RAG Engines](#)

関連トピック

- [Knowledge Bases Reference Manual](#)
- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- [Retrieval Augmented Generation \(RAG\)](#)
- [Embeddings explained](#)
- [Structures Reference Manual](#)
- [Create a Knowledge Base in Gen AI Builder](#)
- [Hybrid Manager: Using Gen AI Builder](#)

Configure an Atlassian Confluence data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ナレッジベースおよび検索拡張生成 RAG ワークフローで使用するために、Atlassian Confluence から Gen AI Builder にコンテンツを取り込みたいプラットフォームユーザー。一般的なユーザーには、Confluence で組織のナレッジを維持する開発者、テクニカルライター、ナレッジマネージャーが含まれます。

あなたが達成すること

1 つ以上の Confluence スペースまたは URL からページを取り込むように、Gen AI Builder で Atlassian Confluence データソースを構成します。取り込まれたコンテンツは、ライブラリとナレッジベースへの変換とインデックス作成に利用できます。

Confluence データソースを使用する理由

- Confluence は、内部の技術文書、ポリシー、エンジニアリングノート、およびプロジェクトの知識のための共通プラットフォームです。
- Confluence コンテンツを Gen AI Builder に取り込むことにより、この貴重な制度的知識を検索可能、AI ワークフローで使用できるようになります。
- 選択したスペースまたはページのターゲットを絞った取り込みにより、きめの細かい制御が提供されます。

このコンテンツがダウンストリームの AI ユースケースをどのように強化するかに関する背景については、以下を参照してください。

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ中程度。アトラシアン API トークンを取得し、Confluence スペース構造を理解する必要があります。
- 推定時間 API トークンが既に生成されている場合は 10~15 分。

重要な考慮事項

データの関連性と品質

- 高品質のドキュメントを使用して、信頼できるスペースまたは特定のページを取り込みます。
- コンテンツが適切に構造化され、セマンティックページレイアウトを使用して AI 処理を向上させることを確認します。

インストレーションの範囲

- 可能であればスペース固有またはページ固有の URL を使用して、広範で無関係なコンテンツの取り込みを回避します。
- 複数のスペースを取り込む場合は、複数のデータソースエントリを構成します。

資格情報のセキュリティ

- 可能な場合は、読み取り専用アクセスの API トークンを使用します。
- 必要な場合を除き、管理者レベルのトークンを使用しないでください。

Atlassian Confluence データソースを構成する方法

1. データソースを管理するインターフェイスで、 **+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **Confluence** を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 **Engineering Wiki Confluence**
 - **Description** オptional 将来の参照のために説明的なコンテキストを追加します。
4. Confluence に接続します。
 - **Confluence Site URL** Confluence サイトの URL を入力します。
 - 例 `https://mycompany.atlassian.net` または `https://mycompany.atlassian.net/wiki/spaces/DEVKB`
 - **API Token** 有効な Atlassian API トークンを入力します。
 - **Atlassian Email** API トークンに関連付けられたメールアドレスを入力します。
5. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - cron 式を提供します。例 `0 2 ** * 毎日午前 2 時`
 - **データの変換** PG.AI 構造を適用して、取り込み中にデータを変換するには、このオプションを有効にします。
 - リストから既存の構造を選択します。
6. **Create** を選択して、Confluence データソースを追加します。

サポートされているコンテンツ

- Confluence ページのテキストデータ。
- サポートされているマクロのコンテンツテキストとしてレンダリングされる場合。
- ページ階層のコンテンツ。

コンテンツのインデックス付けと取得の方法の詳細については、*Embeddings explained* を参照してください。

データソースの管理と更新

Confluence データソースは、作成すると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。
- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- Data Loaders
- Web Scraper Drivers

障害対応

認證失敗

- Confluence サイト URL、API トークン、およびメールが正しいことを確認します。
- API トークンに十分な読み取り権限があることを確認します。
- データジョブ履歴で特定のエラーを確認します。

スペースまたはページが見つかりません

- **Confluence URL** をダブルチェックし、アカウントにターゲットスペースまたはページにアクセスできることを確認します。
- ページ固有の **URL** を使用する場合、ページが制限されていないことを確認します。

シナリオ例

会社の Confluence インスタンスの「Developer Knowledge Base」スペースからコンテンツのインデックスを作成したいと考えています。

構成例

- 名前 Developer KB Confluence Space
- Confluence サイト URL <https://mycompany.atlassian.net/wiki/spaces/DEVKB>
- API トークン xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- Atlassian Email developer@example.com
- スケジュールされた更新 @ 2 ** * 毎日午前 2 時

- データの変換構造を適用してヘッダーとフッターをストリップし、コアコンテンツを抽出します。

関連トピック

- Knowledge Bases overview
- *Embeddings explained*
- *Create a Knowledge Base in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Configure a Google Drive data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ナレッジベースおよび検索拡張生成 RAG ワークフローで使用するために、Google ドライブに保存されているファイルを Gen AI Builder に取り込みたいプラットフォームユーザー。一般的なユーザーには、Google Workspace 内でプロジェクトドキュメントを共同作業する開発者、データエンジニア、ビジネス関係者が含まれます。

あなたが達成すること

選択した Google ドライブフォルダーからファイルを取り込むように、Gen AI Builder で Google ドライブデータソースを構成します。コンテンツは、ライブラリとナレッジベースへの変換とインデックス作成に利用できます。

Google Drive データソースを使用する理由

- 多くのプロジェクトドキュメントは、共同的な Google ドライブフォルダーにあります。
- Google ドキュメント、スライド、スプレッドシートは、他では利用できない動的なチームコンテンツによく使用されます。
- Google ドライブに直接接続すると、手動でエクスポートする手順を行わずにこの貴重なナレッジをキャプチャできます。

このコンテンツがダウンストリームの AI ユースケースをどのように強化するかに関する背景については、以下を参照してください。

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- **複雑さ** 中程度。OAuth フローを完了し、Google ドライブのフォルダー構造を理解する必要があります。
- **推定時間** 5～10 分。

重要な考慮事項

データの関連性と品質

- 有用で信頼できるドキュメントを含む厳選されたフォルダーを取り込みます。
- Google ドキュメントのよく書かれたコンテンツと、適切にフォーマットされた Google スライドまたはスプレッドシートは、より効果的な AI 応答に貢献します。

インストレーションの範囲

- OAuth 承認フロー中にフォルダーを注意深く選択します。
- 意図的な場合を除き、広範なフォルダーマイドライブまたは共有ドライブのルート全体などを取り込まないようにします。

権限スコープ

- OAuth プロセス中に要求された権限を確認します。
- Google Workspace を使用している場合、必要に応じて組織がサードパーティのアプリへのアクセスを許可していることを確認します。

Google Drive データソースを構成する方法

1. データソースを管理するインターフェイスで、**+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **Google Drive** を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 **Project Alpha GDrive Docs**
 - **Description** オプショナル将来の参照のために説明的なコンテキストを追加します。
4. Google ドライブに接続します。
 - **Connect With Google** を選択して、OAuth 2.0 承認フローを開始します。
 - Google アカウントにサインインし、要求された権限を付与します。
 - 画面のプロンプトに従って、取り込むファイルまたはフォルダーを選択します。
5. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - **cron 式** を提供します。例 **0 2 ** * 毎日午前 2 時**

- **データの変換** PG.AI 構造を適用して、取り込み中にデータを変換するには、このオプションを有効にします。
 - リストから既存の構造を選択します。
6. **Create** を選択して、Google Drive データソースを追加します。

サポートされているファイルタイプ

- Google Workspace ファイルドキュメント、スプレッドシート、スライドテキストに変換。
- PDF
- CSV
- マークダウン
- ほとんどのテキストベースのファイルタイプ。

コンテンツのインデックスが作成され、これらのファイルから取得される方法の詳細については、[Embeddings explained](#) を参照してください。

データソースの管理と更新

Google Drive データソースは、作成が完了すると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。
- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- Data Loaders
- File Manager Drivers

障害対応

認証に失敗しました

- Google アカウントを確認し、OAuth フローを再試行します。
- 組織がサードパーティのアプリへのアクセスを許可していることを確認します。
- OAuth フローが完了しない場合は、ブラウザ設定とポップアップブロッカーを確認します。

ファイルまたはフォルダーが表示されない

- Google ドライブのターゲットファイルまたはフォルダーにアクセスできることを確認します。
- 共有ドライブを使用する場合、選択したファイルがアカウントに表示され、アクセスできることを確認します。

シナリオ例

共有の Google ドライブフォルダーに保存されているプロジェクト計画ドキュメントのインデックスを作成したいと考えています。

構成例

- 名前 Project Plans & Presentations GDrive
- OAuth フロー 正常に完了しました。
- 選択したフォルダー Active Projects Q2
- スケジュールされた更新 0 2 ** * 毎日午前 2 時
- データを変換構造を適用して Google ドキュメントコンテンツを要約し、主要なセクションを抽出します。

関連トピック

- Knowledge Bases overview
- *Embeddings explained*
- *Create a Knowledge Base in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Configure an Amazon S3 data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ナレッジベースおよび検索拡張生成 RAG ワークフローで使用するために、Amazon S3 バケットに保存されているファイルを Gen AI Builder に取り込みたいプラットフォームユーザー。一般的なユーザーには、エンタープライズデータパイプラインを管理する開発者、データエンジニア、クラウドアーキテクトが含まれます。

あなたが達成すること

1 つ以上の S3 URI からファイルを取り込むように Gen AI Builder で Amazon S3 データソースを構成します。取り込まれたコンテンツは、ライブラリとナレッジベースへの変換とインデックス作成に利用できます。

Amazon S3 データソースを使用する理由

- S3 は、PDF、CSV、ドキュメントなどのエンタープライズコンテンツのための一般的なストレージプラットフォームです。
- S3 に直接接続することにより、データを複製せずに既存のクラウドストレージを活用できます。
- フォルダーまたはファイルの正確なターゲティングにより、取り込まれたコンテンツの高効率と制御が保証されます。

このコンテンツがダウンストリームの AI ユースケースをどのように強化するかに関する背景については、以下を参照してください。

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- **複雑さ** 中程度。AWS 資格情報を安全に構成し、S3 URI 構造を理解する必要があります。
- **推定時間** IAM 権限が既に構成されている場合、10～15 分。

重要な考慮事項

データの関連性と品質

- AI アプリケーションに有意義に貢献する、信頼できる厳選されたファイルを取り込みます。
- よく構造化され、明確に書かれたドキュメントは、下流の AI の結果の向上につながります。

インストレーションの範囲

- ターゲットを絞った S3 URI を使用して、必要なコンテンツのみを取り込みます。
- 不必要または機密コンテンツを取り込む可能性のあるルートレベルのプレフィックスを指定しないでください。

資格情報のセキュリティ

- **最小の特権** で IAM 資格情報を使用する - Gen AI Builder が必要とする特定の S3 リソースへの読み取り専用アクセスのみを付与します。
- 過剰に許可されたルートまたは管理キーを使用しないでください。

コスト意識

- 特に大規模な取り込みジョブまたは頻繁な更新スケジュールの場合、S3 データの転送と取得のコストに注意してください。

Amazon S3 データソースを構成する方法

1. データソースを管理するインターフェイスで、 **+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **Amazon S3** を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 **Marketing Assets S3**
 - **Description** オプション将来の参照のために説明的なコンテキストを追加します。
4. S3 に接続します。
 - **AWS Access Key** AWS アクセスキー ID を入力します。
 - **AWS Secret Key** AWS シークレットアクセスキーを入力します。
 - **URI** オブジェクトまたはプレフィックスの 1 つ以上の S3 URI を提供します。
 - 例 `s3://bucket-name/key_name`
 - **Add URI** を選択して、URI を追加します。
5. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - **cron 式** を提供します。例 `0 2 ** * 毎日午前 2 時`
 - **データの変換** PG.AI 構造を適用して、取り込み中にデータを変換するには、このオプションを有効にします。
 - リストから既存の構造を選択します。
6. **Create** を選択して、Amazon S3 データソースを追加します。

サポートされているファイルタイプ

- PDF
- CSV
- マークダウン
- ほとんどのテキストベースのファイルタイプ

コンテンツのインデックスが作成され、これらのファイルから取得される方法の詳細については、[Embeddings explained](#) を参照してください。

データソースの管理と更新

Amazon S3 データソースは、作成されると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。

- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- File Manager Drivers
- Data Loaders

障害対応

アクセスが拒否されました

- IAM ユーザーまたはロールに、必要な `s3:GetObject` および該当する場合は `s3:ListBucket` 権限があることを確認します。
- バケットポリシーと S3 URI が正しいことを確認します。
- データジョブ履歴で、特定の権限関連のエラーを確認します。

バケットまたはキーが見つかりません

- S3 URI が正しいことをダブルチェックします。
- 指定されたキーまたはプレフィックスが存在し、IAM ユーザーまたはロールがアクセスできることを確認します。

シナリオ例

S3 フォルダーに保存されている四半期財務レポートのインデックスを作成します。

構成例

- **名前** Financial Reports S3 Archive
- **AWS アクセスキー** AKIAxxxxxxxxxxxxxxxxxx
- **AWS シークレットキー** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- **URI** `s3://my-company-financials/quarterly-reports/pdf/`
- **スケジュールされた更新** 0 2 ** * 毎日午前 2 時
- **データを変換構造を適用して PDF を要約し、主要な財務数値を抽出します。**

関連トピック

- Knowledge Bases overview
- *Embeddings explained*
- *Create a Knowledge Base in Gen AI Builder*

- Hybrid Manager: Using Gen AI Builder

Configure a Web Page data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ナレッジベースおよび検索拡張生成 RAG ワークフローで使用するために Web コンテンツを Gen AI Builder に取り込みたいプラットフォームユーザー。一般的なユーザーには、コンテキスト依存 AI アプリケーションに取り組んでいる開発者、AI アーキテクト、およびビジネスオーナーが含まれます。

あなたが達成すること

選択した Web ページを取り込むように Gen AI Builder で Web ページデータソースを構成します。コンテンツは、下流のライブラリとナレッジベースへの変換とインデックス作成に利用できます。

Web ページデータソースを使用する理由

- Web ページには、内部データソースを補完する貴重な厳選された公開情報が含まれることがよくあります。
- 関連する Web コンテンツを追加すると、ナレッジベースが拡張され、AI 回答、RAG パフォーマンス、およびセマンティック検索が強化されます。
- 特定の Web ページの的を絞った取り込みにより、システムは高品質で関連性の高い外部コンテンツのみを使用します。

ナレッジベースと RAG アプリケーションがこのコンテンツを使用する方法に関するガイダンスについては、以下を参照してください。

- Knowledge Bases overview
- Hybrid Manager Gen AI Builder overview

重要な考慮事項

データの関連性と品質

- 役立つ、高品質、関連性の高いコンテンツを含む Web ページを取り込みます。
- 構造化された明確に書かれた HTML を使用したページが最も効果的です。
- 主要なコンテンツに JavaScript を必要とする動的なページは避けてください。

インストレーションの範囲

- 特定のページをターゲットとします。
- 必要な場合を除き、ウェブサイト全体またはルートレベル URL の追加は避けてください。大量の無関係なデータと処理コストの増加につながる可能性があります。

コンプライアンスと利用規約

- ウェブサイトの robots.txt ファイルと利用規約を常に尊重してください。
- 目的の使用のためにコンテンツを摂取する権限が付与されていることを確認します。

Web ページデータソースを構成する方法

1. データソースを管理するインターフェイスで、**+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **Web Page** を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 EDB Main Site FAQs
 - **Description** オプション将来の参照のために説明的なコンテキストを追加します。
 - **URLs** 取り込む Web ページの完全な URL を 1 つ以上入力します。
 - 例 `https://www.enterprisedb.com/`
 - **Add URL** を選択して、追加の URL を提供します。
4. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - cron 式を提供します。例 `0 2 ** * 毎日午前 2 時`
 - **データの変換** 取り込み中にデータを変換する PG.AI 構造を適用するには、このオプションを有効にします。
 - リストから既存の構造を選択します。
5. **Create** を選択して、Web ページデータソースを追加します。

サポートされているコンテンツ

- HTML ページのテキストコンテンツ。
- クライアントサイドでレンダリングされたコンテンツまたは非常に動的なページのサポートは限定的です。

データソースの管理と更新

Web ページデータソースは、作成されると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。
- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- Web Scraper Drivers
- Data Loaders

障害対応

URL にアクセスできません

- URL にタイプミスがないか確認します。
- ページがパブリックであり、ログインまたはアクセス制御によって保護されていないことを確認します。

コンテンツがないか不完全なコンテンツ

- ページは、主要なコンテンツで JavaScript に依存している場合がありますが、完全にはサポートされていません。
- スクレイピング防止保護は、摂取をブロックする場合があります。
- データジョブ履歴を確認します。0 バイトが取り込まれた場合、動的レンダリングが原因である可能性があります。

シナリオ例

公開 EDB ウェブサイトの関連記事でナレッジベースを充実させたいと考えています。

構成例

- 名前 EDB Blog - Latest GenAI Articles
- URL <https://www.enterprisedb.com/blog/category/genai>
- スケジュールされた更新 0 2 ** * 毎日午前 2 時
- データを変換する構造を使用して、作者の経歴などの不要な定型プレートを削除します。

関連トピック

- Knowledge Bases overview
- *Create a Knowledge Base in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Configure a Custom data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

独自のシステム、内部データベース、またはビルトインデータソースタイプでサポートされていない API からデータを取り込む必要があるプラットフォームユーザー。一般的なユーザーには、カスタムパイプラインと構造を構築する開発者、システムインテグレーター、高度なデータエンジニアが含まれます。

あなたが達成すること

データプロバイダーとして機能する事前構成された PG.AI 構造を選択することにより、Gen AI Builder でカスタムデータソースを構成します。この構造の出力は取り込まれ、ライブラリとナレッジベースのインデックス作成に利用できます。

カスタムデータソースを使用する理由

- 内部データは、多くの場合、独自のデータベース、API、または専用のデータストアに存在します。
- カスタムデータソースを使用すると、データの取得を実行する PG.AI 構造を記述することにより、これらのシステムを接続できます。
- これにより、最大限の柔軟性を提供しながら、結果のコンテンツが Gen AI Builder の取り込みおよび変換パイプラインを介してフローすることを保証します。

このコンテンツがダウンストリームの AI ユースケースをどのように強化するかに関する背景については、以下を参照してください。

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- Hybrid Manager: Using Gen AI Builder

構造物に関連する概念については、以下を参照してください。

- *Structures Reference Manual*

複雑さと完了までの時間

- **複雑さ** 高。データ取得ロジックを正しく実装する PG.AI 構造を書き込むか、それにアクセスする必要があります。
- **推定時間** 15~60 分 構造の複雑さとターゲットシステムの統合によって異なります。

重要な考慮事項

構造設計

- 選択した PG.AI 構造は、データソースデータプロバイダー構造として機能するように実装およびテストする必要があります。

- 構造体は、ダウンストリーム処理に適した形式通常はテキストでデータを返す必要があります。
- 構造体は、再試行、エラー条件、およびページングを処理する必要があります該当する場合。

変換ロジック

- オプションで別の「データの変換」構造を適用して、インデックスを作成する前にコンテンツをさらに洗練、クリーン、または強化できます。

セキュリティとコンプライアンス

- ストラクチャが機密システムに接続する場合、適切なアクセス制御、認証、およびロギングが実装されていることを確認します。
- プライバシー規制とデータ処理要件に注意してください。

カスタムデータソースの構成方法

1. データソースを管理するインターフェイスで、 **+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **カスタムデータソース**を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 **Legacy CRM Connector**
 - **Description** オptional将来の参照のために説明的なコンテキストを追加します。
4. カスタムデータソースに接続します。
 - **PG.AI 構造** データ取得ロジックを実装する事前構成された PG.AI 構造を選択します。
 - 例 **InternalProductAPIReader**
5. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - **cron 式** を提供します。例 **0 2 ** * 毎日午前 2 時**
 - **データの変換** 追加の PG.AI 構造を適用して、取り込み中にデータを変換するには、このオプションを有効にします。
 - リストから既存の構造を選択します。
6. **Create** を選択して、カスタムデータソースを追加します。

サポートされているコンテンツ

- 選択した PG.AI 構造の出力によって決定されます。
- 通常、AI 処理に適したテキストデータ。

データソースの管理と更新

カスタムデータソースは、作成すると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。
- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- *Structures Reference Manual*
- *Tools Reference Manual*

障害対応

リストされていない構造

- PG.AI 構造がデータソースデータプロバイダー構造として正しく登録され、マークされていることを確認します。
- 現在のプロジェクトまたは環境で構造にアクセスできることを確認します。

エラーまたは不完全なデータの取得

- 構造の実装にエラー、認証の問題、または予期しない API の変更がないかを確認します。
- データジョブ履歴で詳細なエラーメッセージを確認します。

シナリオ例

内部 REST API から Gen AI Builder に製品カタログデータを取り込みたいと考えています。

構成例

- 名前 Product Catalog via Custom API
- PG.AI 構造 InternalProductAPIReader
- スケジュールされた更新 0 2 ** * 毎日午前 2 時
- データの変換製品の説明を抽出およびフォーマットする構造を適用します。

関連トピック

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- *Structures Reference Manual*

- *Embeddings explained*
- *Create a Knowledge Base in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Configure a Data Lake data source

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ナレッジベースおよび検索拡張生成 RAG ワークフローで使用するために、プラットフォームの管理対象データレイクから Gen AI Builder にファイルベースのコンテンツを取り込みたいプラットフォームユーザー。これは通常、内部構造化または非構造化ドキュメントを必要とする AI アプリケーションを構築する開発者、データエンジニア、およびビジネスオーナーが使用します。

あなたが達成すること

選択した Data Lake バケットとオプションで特定のアセットパスからファイルを取り込むように、Gen AI Builder で Data Lake データソースを構成します。コンテンツは、ライブラリとナレッジベースのインデックス作成に利用できます。

Data Lake データソースを使用する理由

- 多くのエンタープライズドキュメント PDF、CSV、Markdown、テキストファイルは、既に内部 Data Lake ストレージに存在します。
- Data Lake ソースは、このコンテンツを AI ワークフローに取り込むための信頼性の高い費用対効果の高い方法を提供します。
- パブリック Web コンテンツとは異なり、Data Lake ファイルは通常、信頼性の高い内部ドキュメントです。
- Data Lake ソースにより、正確なターゲティング特定のフォルダーまたはファイルが可能になり、効率的でスケーラブルになります。

このコンテンツがダウンストリームの AI ユースケースをどのように強化するかに関する背景については、以下を参照してください。

- Knowledge Bases overview
- *Retrieval Augmented Generation (RAG)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- **複雑度** 低～中程度。データレイク構造バケット名とパスに精通する必要があります。
- **推定時間** パスが既知で、バケットアクセスが既に構成されている場合は 5～10 分。

重要な考慮事項

データの関連性と品質

- AI アプリケーションに役立つ、信頼できる高品質のファイルのみを取り込みます。
- 適切に構造化されたファイル一貫したフォーマット、よく書かれたテキストは、AI の結果の向上につながります。

インストレーションの範囲

- 取り込むファイルまたはフォルダーを具体的にします。可能であれば、バケット全体ではなくターゲットパスを使用します。
- スcopeが広すぎる生データバケット全体などを取り込むと、大量の処理コストと無関係なコンテンツが発生する可能性があります。

権限とアクセス

- 選択したデータレイクバケットとパスに対する適切な権限が必要です。
- 特定のアセットパスが提供されない場合、選択したバケット内のすべてのファイルが取り込まれます。

Data Lake データソースの構成方法

1. データソースを管理するインターフェイスで、 **+ Add New Data Source** を選択します。
2. 使用可能なデータソースタイプから **Data Lake** を選択します。
3. 次のフィールドを設定します。
 - **名前** 明確で一意の名前を指定します。例 **Q1 Product Specs - Data Lake**
 - **Description** オptional 将来の参照のために説明的なコンテキストを追加します。
4. データレイクに接続します。
 - **バケット** ドロップダウンリストからデータレイクバケットを選択します。
 - **アセットパス**
 - ファイルまたはフォルダーへの 1 つ以上の特定のパスを入力します。例 `manuals/pdf/current-version/`
 - **Add Asset Path** を選択して、パスを追加します。
 - パスが提供されない場合、システムはバケット内のすべてのファイルを取り込みます。
5. (オプション) 詳細オプションを構成します。
 - **スケジュールされた更新** スケジュールしてデータを自動的に更新するには、このオプションを有効にします。
 - **cron 式** を提供します。例 `0 2 ** * 毎日午前 2 時`
 - **データの変換** PG.AI 構造を適用して、取り込み中にデータを変換するには、このオプションを有効にします。

- リストから既存の構造を選択します。
6. **Create** を選択して、Data Lake データソースを追加します。

サポートされているファイルタイプ

- PDF
- CSV
- マークダウン
- ほとんどのテキストベースのファイルタイプ

コンテンツのインデックスが作成され、これらのファイルから取得される方法の詳細については、[Embeddings explained](#) を参照してください。

データソースの管理と更新

データレイクデータソースは、作成されると、データソースインターフェイスを介して表示および管理できます。

使用可能なアクション

- **Edit** データソース構成を変更します。
- **更新** データ取り込みジョブを手動でトリガーします。
- **Delete** データソースを削除します。

データジョブ履歴を確認して、取り込みパフォーマンスを監視し、問題のトラブルシューティングを行うこともできます。

SDK リファレンス

- File Manager Drivers
- Data Loaders

障害対応

ファイルが見つかりません

- バケットとアセットのパスが正しいことを確認します。
- データジョブ履歴にエラーがないか確認します。

アクセスの問題

- ユーザーまたはシステムが、選択したバケットとパスにアクセスするために必要な権限があることを確認します。

シナリオ例

データレイクに保存されている一連の PDF 製品マニュアルのインデックスを作成します。

構成例

- 名前 Product Manuals Archive
- バケット technical-documentation-prd
- アセットパス manuals/pdf/current-version/
- スケジュールされた更新 0 2 ** * 毎日午前 2 時
- データを変換 構造を適用して PDF から特定のセクションを抽出します。

関連トピック

- Knowledge Bases overview
- *Embeddings explained*
- *Create a Knowledge Base in Gen AI Builder*
- Hybrid Manager: Using Gen AI Builder

Configure the Data Lake in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ハイブリッドマネージャー AI ファクトリーで **Gen AI Builder** をセットアップするプラットフォームユーザー。これには、プラットフォーム管理者、DevOps エンジニア、AI Factory パイプラインのバックエンドストレージを構成する AI ビルダーが含まれます。

あなたが達成すること

データレイク -AI Factory サービスが機能するために必要なオブジェクトストレージバックエンドを構成します。

あなたは次のことを行います。

- ストレージバケットを作成する
- 必要な CORS ポリシーを設定します
- アクセス資格情報を生成および提供する

データレイクを構成する理由

データレイクは、以下を保存して電力を供給するために必要です。

- データソースからアップロードされたファイル

- パイプラインとナレッジベースで使用するインデックス付きデータとエンベディング
- アシスタントに必要な構造とツール
- AI パイプラインによって使用される一時アーティファクト

構成済みのデータレイクがない場合

- ライブラリ、ナレッジベース、およびアシスタントは機能しません。
- RAG パイプラインは動作しません。
- データソースを取り込むことはできません。

データレイクデータソースを追加またはナレッジベースを作成する前に構成する必要があります。

背景の場合

- *Data Lake Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 中程度オブジェクトストレージセットアップ+構成
- 推定時間 10～20 分

前提条件

- S3 互換オブジェクトストレージプロバイダー へのアクセス
- AWS S3
- S3 相互運用性が有効になった Google Cloud StorageGCS
- その他の互換サービス MinIO、Ceph など
- バケットを作成し、CORS を管理する権限。
- オブジェクトストレージの資格情報を取得する機能。

データレイクの構成方法

1. 専用のバケットを作成する

Gen AI Builder とパイプライン専用の新しいバケットをプロビジョニングします。

ベストプラクティス専用のバケットを使用します。汎用または混合コンテンツのバケットを再利用しないでください。

AWS S3 または互換性

- 新しいバケット例 `genai-data-lake-hcp` を作成します。

Google Cloud StorageGCS

- バケットを作成します。
- **S3 相互運用性** を有効にします。
- 必要なロールの割り当て
- ストレージ管理者
- サービスアカウントトークン作成者
- HMAC キーを生成しますアクセスキー ID およびシークレットアクセスキーとして使用されます。

2. CORS ポリシーの構成

Cross-Origin Resource SharingCORS を構成して、Gen AI Builder コンソールと API がデータレイクと対話できるようにする必要があります。

S3 互換ストレージの CORS の例

```
[
  {
    "AllowedHeaders": ["*"],
    "AllowedMethods": ["PUT", "POST", "DELETE", "GET", "HEAD"],
    "AllowedOrigins": ["https://<PORTAL_DOMAIN_NAME>"],
    "ExposeHeaders": []
  }
]
```

GCS の CORS の例

1. `cors-config.json` を作成します。

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

2. CORS ポリシーを適用します。

```
gsutil cors set cors-config.json gs://<your-gcs-bucket-name>
```

3. ストレージ資格情報の取得

以下が必要です。

- バケット名
- S3 エンドポイント URL
- アクセスキー ID
- シークレットアクセスキー
- リージョン該当する場合

GCS の場合アクセスキー ID およびシークレットアクセスキーとして HMAC キーを使用します。

4. 展開に資格情報を提供する

Kubernetes 展開またはバックエンドを構成します。

- 環境変数または構成ファイルを使用します。
- Helm チャート、シークレットマネージャー、または Kubernetes Secret の一部として適用します。

環境変数の例

```
DATA_LAKE_BUCKET_NAME=genai-data-lake-hcp  
DATA_LAKE_ENDPOINT_URL=https://<your-endpoint>  
DATA_LAKE_ACCESS_KEY_ID=<access-key-id>  
DATA_LAKE_SECRET_ACCESS_KEY=<secret-access-key>  
DATA_LAKE_REGION=<region> # If applicable
```

5. 接続性の検証

1. Gen AI Builder → **Data Lake** セクションにログインします。
2. 設定したバケットが表示されることを確認します。
3. フォルダーを作成するか、テストファイルをアップロードして、権限を確認します。

例 genai-data-lake-hcp → Create Folder → Upload File

障害対応

バケットは **Gen AI Builder** に表示されません

- 資格情報が正しいことを確認します。
- CORS ポリシーで必要な HTTP メソッドが許可されていることを確認します。
- バケットが Gen AI Builder / AI Factory 専用であることを確認します。

アップロードまたはコンソールのインタラクションが失敗する

- CORS 設定を確認します。
- API キーまたは資格情報に正しい権限があることを確認します。

アクセス拒否エラー

- アクセスキー ID とシークレットアクセスキーに必要な権限があることを確認します。
- S3 またはストレージオブジェクト管理者
- バケットの読み取り/書き込み/削除権限

SDK リファレンス

- Data Artifacts — コンポーネント間でデータを渡す
- File Manager Drivers — オブジェクトストレージのアクセスパターン

関連トピック

- *Data Lake Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Configure Data Sources in Gen AI Builder*
- *Knowledge Bases Reference Manual*
- Hybrid Manager: Using Gen AI Builder

Create a Retriever in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ハイブリッドマネージャー AI ファクトリー の **Gen AI Builder** でレトリバーを構成するプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、会話型デザイナーが含まれます。

あなたが達成すること

レトリバー を作成して、アシスタントが **どのナレッジベース**をクエリし、 **どのように** コンテンツを取得するかを制御します。

以下を構成します。

- ターゲットナレッジベース
- 最大トークン制限

- オプションのメタデータ

レトリバーは、アシスタントの **グラウンディング**、**コンテキスト認識** AI 応答を有効にします。

レトリバーを作成する理由

レトリバーは、**検索拡張生成 RAG** パイプラインの重要なコンポーネントです。

- どの **ナレッジベース** アシスタントがクエリを定義します。
- LLM に渡される **セマンティック検索** と **コンテキストサイズ**を制御します。
- 監査、バージョン管理、またはカスタムロジックのメタデータインジェクションを有効にします。
- レトリバーがないと、アシスタントはコンテンツを取得して応答を基にすることができません。

背景については、以下を参照してください。

- *Retrievers Reference Manual*
- *Knowledge Bases Reference Manual*
- *Retrieval Augmented Generation (RAG)*

ハイブリッドマネージャー展開でレトリバーがどのように使用されるかを確認するには、以下にアクセスしてください。

- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 低～中程度
- 推定時間 5～10 分

レトリバーを作成する場合

- **RAG 機能** をアシスタントに割り当てる前。
- アシスタントがナレッジを取得および優先順位を付ける方法を調整する場合。
- **複数のナレッジベース** のコンテンツを組み合わせる場合。
- さまざまなペルソナまたはユースケースに合わせて取得動作をカスタマイズする場合。

レトリバーの作成方法

1.[Retrievers] に移動します

- Gen AI Builder コンソールで、**Retrievers** に移動します。

2. 新しいレトリバーを作成します

- **Create Retriever** をクリックします。

3. Retriever フィールドの構成

お名前必須

- 明確で一意の名前を指定します。
- 例 Product Docs Retriever

説明オプショナル

- レトリバーの目的または使用目的について説明します。
- 例 Retriever for product documentation and customer FAQs.

ナレッジベース 必須

- ターゲットする 1 つ以上の既存の ナレッジベース を選択します。
- 例 PG Financial Product FAQs KB、PG Financial Account Services KB

最大トークンオプショナル

- LLM に渡されるコンテキストの **最大トークン** 制限を設定します。
- 例 2000

ガイダンス

- トークン制限が高いほど → コンテキストは多くなりますが、LLM プロンプトが大きくなります。
- トークン制限が低い → 応答は速くなりますが、必要なコンテンツを見逃す場合があります。
- 一般的な範囲 **1000-3000**。

メタデータオプショナル

- オプションのメタデータを有効な JSON オブジェクトとして提供します。
- 例 {"version": "1.0", "team": "AI Engineering"}

使用法

- Retriever 構成をタグ付けまたはバージョン付けします。
- カスタムグリップテープ構造またはアシスタントテンプレートで高度なロジックを駆動します。

4. 作成を完了する

- **Create** をクリックします。
- 新しいレトリバーは **レトリバー** リストに表示されます。
- 1 人以上のアシスタントに割り当てられるようになりました。

レトリバーとアシスタントの関連付け

- **アシスタント** を構成するときは、**Advanced Options** に移動します。
- 関連する 1 つ以上のレトリバーを選択します。
- これは、実行時にアシスタントがクエリする **ナレッジベース** を制御します。

ヒント さまざまなペルソナまたはアシスタントのユースケースに**異なるレトリバー** を使用します。きめの細かい RAG 制御が有効になります。

シナリオ例

顧客の製品の質問に答えるアシスタントのレトリバーを作成します。

- 名前 `Product Support Retriever`
- 説明 `Fetches product specs and pricing from KBs.`
- ナレッジベース
- `Product Catalog Hybrid KB`
- `Pricing Policies KB`
- 最大トークン `2000`
- メタデータ `{"team": "Support AI"}`

SDK リファレンス

- RAG Engines — 取得モジュールと応答アセンブリ
- Embedding Drivers
- Rerank Drivers
- Vector Store Drivers

障害対応

レトリバーが予想した結果を返さない

- ターゲット **ナレッジベース** に関連するコンテンツが含まれていることを確認します。
- **最大トークン** 設定を確認します。
- 正しいレトリバーがアシスタントに割り当てられていることを確認します。

アシスタントの構成時にレトリバーが表示されない

- レトリバーが正常に作成されたことを確認します。
- 選択したナレッジベースがアクセス可能で、インデックス付けされていることを確認します。

トークンのインパクト

- 高すぎる設定は、LLM コンテキストウィンドウ → リスク切り捨てを超える可能性があります。
- 設定が低すぎると、不完全または役に立たない応答が発生する場合があります。

ヒント 2000 トークンから開始し、アシスタントのテストに基づいて調整します。

関連トピック

- *Retrievers Reference Manual*
- *Knowledge Bases Reference Manual*
- *Retrieval Augmented Generation (RAG)*
- *Structures Reference Manual*
- Hybrid Manager: Using Gen AI Builder

How to build assistants

Create a Ruleset in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

ハイブリッドマネージャー AI ファクトリー の **Gen AI Builder** でアシスタント または ストラクチャーをビルドするプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、コンテンツデザイナー、コンプライアンス所有者が含まれます。

あなたが達成すること

ルールセット-アシスタントまたはストラクチャの動作、トーン、およびコンプライアンスをガイドするために自然言語で書かれた**ルール** のコレクションを作成します。

あなたは次のことを行います。

- 1 つ以上のルールを定義します。
- ルールセットにパッケージ化します。
- アシスタントでルールセットを使用できるようにします。

ルールセットを作成する理由

- ルールセットは、AI 駆動のインタラクションの **行動ガイドライン** と **ガバナンス** を定義します。
- 応答のトーン、スタイル、およびコンプライアンスを制御します。
- さまざまな **アシスタントペルソナ** を実装するのに役立ちます。

- アシスタントとアプリケーション全体で一貫した動作を有効にします。
- これらは、AI パイプラインのナレッジベースとレトリバーを補完します。

背景については、以下を参照してください。

- *Rulesets Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 低
- 推定時間 5～10 分

ルールセットを作成する場合

- アシスタントの **トーン**、**音声**、または**ポリシーコンストレイン** を定義する場合。
- さまざまなペルソナまたはユーザー対象者にアシスタントを実装する場合。
- **ブランド基準** または **規制ガイドライン** へのコンプライアンスを保証する場合。
- アシスタントを運用環境に割り当てる前。

前提条件

- **アシスタント機能** を有効にして展開された Gen AI Builder。

ルールセットの作成方法

1.[ルールセット] に移動します

- Gen AI Builder コンソールで、**Rulesets** に移動します。

2. 新しいルールセットを作成します

- **Create Ruleset** をクリックします。

3. ルールセットフィールドの構成

お名前必須

- 明確で一意的な名前を指定します。
- 例 Customer Service Tone Rules

説明オプション

- ルールセットの目的または範囲について説明します。
- 例 Guidelines for polite, professional responses in customer interactions.

ルール必須

- 1 つ以上の既存の **ルール** を選択して、ルールセットを設定します。
- 必要に応じて、この手順中に新しいルールを作成します。

新しいルールの作成

- **Rules** フィールドで、**Create new rules** をクリックします。
- モーダル内
- 名前 Maintain Polite Tone
- ルールテキスト Always address the user respectfully. Use polite greetings and sign-offs.
- **Create** をクリックして、ルールを保存します。
- 繰り返して、さらにルールを追加します。

エイリアスオプション

- オプションの **alias** - 使いやすい参照名を提供します。
- 例 polite-support

メタデータオプション

- オプションのメタデータを有効な JSON オブジェクトとして提供します。
- 例 { "version": "1.0", "author": "support_team" }

4. ルールセットの作成を完了する

- **Create** をクリックして、ルールセットを保存します。
- 新しいルールセットは **ルールセット** リストに表示されます。
- アシスタントに割り当てられるようになりました。

ルールセットとアシスタントの関連付け

- **アシスタント** を構成するときは、**動作とルール** または **詳細オプション** に移動します。
- 割り当てる 1 つ以上の **ルールセット** を選択します。
- ルールセットは、アシスタントのインタラクション中に **行動ガイダンス** として使用されます。

ヒント モジュール式ルールセットを使用する — 多層動作制御のためにグローバル、ペルソナ固有、およびコンプライアンスルールセットを組み合わせます。

シナリオ例

あなたは、**カスタマーサポートアシスタント**に丁寧でプロフェッショナルな口調を適用するルールセットを作成したいと考えています。

構成例

- 名前 Polite Support Tone
- 説明 Ensures the Assistant maintains a polite and professional tone during customer interactions.
- エイリアス polite-support
- メタデータ { "version": "1.0", "author": "support_team" }

ルール

1. 正式な挨拶を使用します Always begin responses with a polite greeting, such as "Hello," or "Good day,". Address the user respectfully if their name is known.
2. 明確に支援を提供する Clearly state how you can assist the user. For example, "How may I help you today?" or "I can assist you with..."

作成後

- ルールセットを **カスタマーサポートアシスタント** に割り当てます。

障害対応

アシスタントの構成時にルールセットが表示されない

- ルールセットに少なくとも 1 つのルールがあることを確認します。
- ルールセットが正常に作成されたことを確認します。

競合するまたはあいまいなルール

- ルールテキストを注意深く確認します。重複 または 矛盾する指示を避けます。
- ルールセットを適用してアシスタントをテストし、必要に応じて調整します。

ルールセットの更新が反映されない

- **アシスタント構成**が最新のルールセットバージョンを参照していることを確認します。
- 必要に応じて、アシスタントを再割り当てまたは更新します。

ベストプラクティス

- **モジュール式** ルールセットを使用します — 個別のコンプライアンス、トーン、およびペルソナのガイドライン。
- **ルールを明確で明確な** にしてください — 競合する指示を避けます。
- ルールセットメタデータを介して **バージョン管理** を実装します。
- アシスタントの応答を定期的に確認して、割り当てられたルールセットに対する動作を検証します。

関連トピック

- [Rulesets Manual](#)
- [Structures Reference Manual](#)
- [Assistants Reference Manual](#)
- [Gen AI Builder Concepts \(Assistants, Knowledge, Tools, Threads\)](#)
- Hybrid Manager: Using Gen AI Builder

Create a Structure in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

これは誰に向けたものですか

Hybrid Manager AI Factory の **Gen AI Builder** で高度な AI ワークフローを構築するプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、データエンジニアが含まれます。

あなたが達成すること

Zip ファイルとしてパッケージ化されたエージェント、パイプライン、またはワークフローを展開することにより、Gen AI Builder で **構造** を作成します。AI Factory で実行するように構造を構成します。

構造体を作成する理由

- 基本的な会話を超えた **高度な AI ワークフロー** を構造化します。
- これらは次のことを可能にします。
- データソースの **データ変換**。
- アシスタントのツールを介した **タスク実行**。
- カスタム取得パイプライン。
- 外部 **API** およびサービスとの統合。
- AI Factory アプリケーション全体で再利用可能なビジネスロジックコンポーネント。

背景については、以下を参照してください。

- *Structures Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- Hybrid Manager: Using Gen AI Builder

複雑さと完了までの時間

- 複雑さ 中程度
- 推定時間 10～20 分

構造体を作成する場合

- アシスタントから呼び出し可能な **カスタムツール** をビルドする場合。
- データソースパイプラインのデータを変換またはエンリッチする場合。
- マルチステップのロジックを必要とする複雑な **AI パイプライン** を実装する場合。
- 外部 **API**、**システム**、または **ビジネスワークフロー** を統合する場合。

前提条件

- **Zip ファイル** としてパッケージされた構造
- 構造体コードと依存関係が含まれます。
- 構造構成通常は **YAML** が含まれます。
- 構成済みの **データレイク** にアップロードされます。

構造体の作成方法

1. Structures に移動します

- Gen AI Builder コンソールで、**Structures** に移動します。

2. 新しい構造を作成します

- **Create Structure** をクリックします。

3. 作成方法を選択します

- **Create Structure from Zip file** を選択します。

4. 構造体フィールドの構成

お名前必須

- 明確で一意の名前を指定します。
- 例 PGFT_TransactionCategorizer

説明オプション

- ストラクチャーが何をするのかを説明します。
- 例 `Categorizes financial transactions using PGFT business logic.`

5. データレイクパラメーターを構成する

バケット ID 必須

- Structure Zip ファイルが保存されている データレイクバケットを選択します。

アセットパス必須

- 選択したバケット内の Zip ファイルへのフルパスを提供します。
- 例 `griptape-structures/transaction_categorizer.zip`

6. ストラクチャー構成パラメーターを構成する

構造構成ファイル推奨

- Zip 内のストラクチャ構成ファイルへの相対パスを入力します。
- 例 `structure_config.yaml`

7. 環境変数の構成オプション

ストラクチャで利用できる環境変数を定義できます。

個々の変数を追加する

- **タイプ** 変数またはシークレット。
- **名前** 例 `API_KEY`
- **値** 例 `your_actual_api_key_here`

.env ファイルのインポート

- オプションで、.env ファイルの内容を一括インポート変数に貼り付けます。

8. 作成を完了する

- **Create** をクリックします。

システムは次のことを行います。

- ファイルを解凍します。
- ランタイム環境をセットアップします。
- AI Factory でストラクチャを利用できるようにします。

9. 展開の検証

- ストラクチャは **ストラクチャ** リストに表示されます。
- **展開ステータス** を監視します。
- 次のことができるようになりました。
- コンソールまたは API を介してストラクチャを手動で実行します。
- アシスタントの **ツール** として構造を使用します。
- **データソースパイプライン** で構造を使用します。

サンプルシナリオ トランザクションカテゴライザー構造の展開

1. パッケージのグリップテープ構造 `TransactionCategorizer` 。
2. データレイクにアップロードする
 - パッケージ `pgai-structures`
 - パス `transaction_categorizer/transaction_categorizer.zip`
3. 構造を作成します。
 - 名前 `PGFT_TransactionCategorizer`
 - 説明 `Categorizes financial transactions.`
 - パッケージ ID `pgai-structures`
 - アセットパス `transaction_categorizer/transaction_categorizer.zip`
 - 構造構成ファイル `structure_config.yaml`
4. 環境変数を追加します。
 - 名前 `MODEL_API_KEY`
 - 値 `your_key_here`
5. **Create** をクリックします。

次のステップ

- 構造を実行して動作を確認します。
- 構造を **ツール** としてアシスタントに割り当てます。
- 必要に応じて、データソースパイプラインの構造を使用します。

SDK リファレンス

- *Structures Reference Manual*
- *Structures Reference Manual*
- *Tools Reference Manual*

- RAG Engines

障害対応

ストラクチャの展開に失敗する

- Zip ファイルの内容を確認します。
- 構造体コードは有効です。
- Structure Config ファイルが存在し、正しく参照されています。
- データレイクパス が正しく、アクセス可能であることを確認します。

環境変数が機能しない

- 変数が正しく入力されていることを確認します。
- シークレットを使用する場合、シークレットが適切に構成されていることを確認します。

ツールとして表示されない構造

- 構造が **ツール互換インターフェイス** を実装していることを確認します。
- 構造が公開され、ツールの割り当てに使用できることを確認します。

実行時の構造エラー

- Gen AI Builder コンソールの構造ログを確認します。
- 外部 API エンドポイントと資格情報を確認します。
- Structure Config 設定を確認します。

ベストプラクティス

- モジュール式の再利用可能な構造デザインを使用します。
- 環境変数を介して構成を外部化します。
- **堅牢なエラー処理** を実装します。
- 実行パフォーマンスのモニター レイテンシーとスループットを検証します。
- Data Lake の構造アーティファクトをバージョン付けします。

関連トピック

- *Structures Reference Manual*
- *Structures Reference Manual*
- *Create an Assistant in Gen AI Builder*
- *Rulesets Manual*
- *Knowledge Bases Reference Manual*

- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- Hybrid Manager: Using Gen AI Builder

Create a Tool in Gen AI Builder

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

Gen AI Builder でツールを作成する

このガイドでは、AI アシスタントが外部システムと対話し、計算を実行し、Gen AI Builder 内でビジネスロジックを実行できるカスタムツールの展開を示します。

前提条件

必須コンポーネント

- 実装コードと依存関係を含む ZIP ファイルとしてパッケージ化されたツール
- ツール構成ファイル YAML または Python 形式
- 構成済みの Data Lake バケットにアップロードされた ZIP ファイル
- 適切な権限による Gen AI Builder のアクセス

開発の依存関係

- pgai-ai-builder SDK を使用したツール実装
- 外部 API 資格情報またはデータベースアクセス該当する場合
- ツール検証のためのテスト環境

実装手順

1. アクセスツール管理

Gen AI Builder コンソール内の **Tools** セクションに移動します。

2. ツール作成の初期化

1. **Create Tool** をクリックして、展開プロセスを開始します
2. 作成方法として **データレイクのグリップテープツール** を選択します

3. ツール ID の構成

ツール名 必須

- 説明的な一意の識別子を提供します
- 明確な命名規則を使用します **Organization_Category_Function**

- 例 PGAI_Financial_StockTicker

説明 オプション

- ドキュメントツールの機能と使用例
- 統合の詳細と予想される動作を含めます
- 例 Retrieves real-time stock prices from external financial API

4. データレイク統合の構成

バケット選択 必須

- ツールパッケージを含む Data Lake バケットを選択します
- バケットのアクセシビリティと権限を確認する

アセットパス 必須

- 選択したバケット内の ZIP ファイルへの完全なパスを指定します
- パス形式 category/tool_name/package.zip
- 例 financial_tools/stock_ticker/stock_ticker_v1.2.zip

5. ツールパラメーターを構成する

ツール構成ファイル 推奨

- 構成ファイルへの ZIP 内の相対パスを指定します
- 標準構成ファイル tool_config.yaml または config.py
- 構成は、ツールパラメーター、検証ルール、および実行設定を定義します

構成構造の例

```
# tool_config.yaml

name: "stock_ticker_tool"
version: "1.2.0"
description: "Retrieves stock market data"
parameters:
  - name: "symbol"
    type: "string"
    required: true
    description: "Stock ticker symbol"
  - name: "period"
    type: "string"
    default: "1d"
    options: ["1d", "5d", "1mo"]
```

6. 環境構成

環境変数 ツール実行用のランタイム環境を構成します。

個々の変数構成

- **タイプ** 変数プレーンテキストまたはシークレット暗号化されたストレージを選択します
- **Name** 環境変数識別子
- **Value** 構成値または資格情報

一般的な変数の例

```
API_KEY = your_api_key_here
BASE_URL = https://api.example.com
TIMEOUT_SECONDS = 30
RETRY_ATTEMPTS = 3
```

大量インポートオプション

- .env ファイル形式を使用して複数の変数をインポート
- バッチ構成用の完全な.env ファイルの内容を貼り付けます

7. 展開ツール

Create をクリックして、展開プロセスを開始します。

展開プロセス

1. システムは ZIP ファイルの構造とコンテンツを検証します
2. ツール構成の解析と検証
3. 実行環境の準備
4. Gen AI Builder カタログへのツール登録

8. 展開の検証

展開後の検証

- ツールはアクティブなステータスでツールカタログに表示されます
- 構成パラメーターが正しく表示される
- 適切に構成された環境変数
- アシスタントの割り当てに使用可能なツール

実践例金融為替レートツール

ツールパッケージの構造

```
fx_rate_tool.zip
├── fx_rate_tool.py      # Tool implementation
├── requirements.txt     # Python dependencies
├── tool_config.yaml    # Tool configuration
└── README.md           # Documentation
```

展開構成

基本情報

- 名前 PGAI_Financial_FXRate
- 説明 Retrieves current foreign exchange rates from external API

データレイク構成

- バケット pgai-financial-tools
- アセットパス fx_tools/fx_rate_tool_v2.1.zip
- 構成ファイル tool_config.yaml

環境変数

```
FX_API_KEY = abcd1234-your-api-key-here
FX_API_URL = https://api.exchangerate-api.com/v4/latest
CACHE_DURATION = 300
MAX_RETRIES = 3
```

ツール構成ファイル

```
name: "fx_rate_tool"
version: "2.1.0"
description: "Foreign exchange rate retrieval tool"
parameters:
  - name: "base_currency"
    type: "string"
    required: true
    description: "Base currency code (USD, EUR, etc.)"
  - name: "target_currency"
    type: "string"
    required: true
    description: "Target currency for conversion"
  - name: "amount"
    type: "number"
```

(次のページに続く)


```
default: 1.0
description: "Amount to convert"
```

障害対応

導入の問題

ZIP ファイル検証エラー

- ZIP ファイルに必要な実装ファイルが含まれていることを確認します
- ツール構成ファイルのパスの精度の確認
- repaired.txt の Python 依存関係仕様を検証する

データレイクアクセスの問題

- バケットの選択と権限を確認する
- ファイル拡張子を含むアセットパスの精度を確認します
- データレイクの接続と認証を確認する

構成の問題

環境変数の問題

- 変数名がツール実装の予想と一致することを確認します
- 機密資格情報のシークレットタイプ変数を確認する
- 変数値とフォーマットを検証する

ツール構成エラー

- 構成ファイルの構文 YAML フォーマットを確認します
- パラメーター定義がツール実装と一致することを確認します
- 必要なパラメーター仕様の確認

実行時の問題

ツールの実行の失敗

- Gen AI Builder コンソールのレビューツールログ
- 外部 API エンドポイントと資格情報を検証する
- 外部統合のネットワーク接続を確認する
- チェックツール実装のエラー処理

アシスタントの統合の問題

- ツールのステータスがカタログでアクティブと表示されていることを確認します

- ツールの権限と可用性を確認する
- ツールの割り当てのアシスタント構成を確認します

ベストプラクティス

ツール開発

実装ガイドライン

- 単一目的のモジュールコンポーネントとしてツールを開発する
- 包括的なエラー処理とタイムアウト管理を実装する
- 一貫したインターフェイスとユーティリティには pgai-ai-builder SDK を使用します
- デバッグとモニタリングのための詳細なログを含めます

構成管理

- 環境変数を介してすべての資格情報と API キーを外部化
- 説明的なパラメーター名と包括的なドキュメントを使用します
- パラメーターの検証とデフォルト値の処理を実装する
- 展開追跡のための体系的なバージョンツールパッケージ

セキュリティに関する考慮事項

資格管理

- Secret タイプの環境変数を使用して機密情報を保存する
- API キーと資格情報を定期的にローテーションする
- 外部システム統合のための最小特権アクセスを実装する
- 通常とは異なるアクティビティパターンのツールの使用状況を監視する

アクセス制御

- ツールの可用性を適切なユーザーグループに制限する
- ツール実行追跡のための監査ログを実装する
- 入力パラメーターを検証して、インジェクション攻撃を防止
- 外部 API 呼び出しに安全な通信プロトコルを使用する

オペレーショナルエクセレンス

テスト手順

- アシスタントを統合する前にツールを独立してテストする
- 予想される入力範囲全体で機能を検証する
- さまざまな障害条件の下でのエラー処理を検証する

- パフォーマンス特性とリソース使用率を監視する

モニタリングとメンテナンス

- モニタツールの実行ログとエラー率
- パフォーマンスメトリックと使用パターンを追跡する
- 依存関係管理の定期的な更新を計画する
- Data Lake でツールアーティファクトのバージョン管理を維持する

SDK リファレンス

- *Tools Reference Manual*
- *Structures Reference Manual*

次のステップ

ツールの展開が成功した後

1. テストツールの機能

- Gen AI Builder 内のツールテスト機能を使用する
- パラメーター処理と応答のフォーマットを検証する
- さまざまな条件の下でのエラー処理を検証する

2. アシスタントとの統合

- ユースケースに基づいて、適切なアシスタントにツールを割り当てます
- ルールセットを介してツール使用ポリシーを構成する
- ツール機能とのテストアシスタントの相互作用

3. モニタパフォーマンス

- ツールの実行メトリックと成功率を追跡
- リソースの使用率と応答時間を監視する
- ツールの障害またはパフォーマンスの低下に関するアラートを確立します

関連ドキュメント

実装リソース

- [pgai-ai-builder SDK](#) 完全な開発フレームワーク
- *Tool Configuration Reference* 詳細な構成オプション
- *Tool Execution Testing* テスト手順と検証方法

統合ガイド

- *Tools Reference Manual* 包括的なツールアーキテクチャと機能

- *Create an Assistant in Gen AI Builder* ツールの割り当てプロシージャ
- *Structures Reference Manual* 自動プロセスでのツールの統合

Create an Assistant in Gen AI Builder

これは誰に向けたものですか

ハイブリッドマネージャー AI ファクトリー の **Gen AI Builder** で AI 駆動アシスタントを構築するプラットフォームユーザー。一般的なユーザーには、開発者、AI アーキテクト、会話デザイナー、コンテンツ所有者が含まれます。

あなたが達成すること

アシスタント - これは、次のことを行うことができる AI 搭載エージェントです。

- 自然言語を介してユーザーと対話する
- ナレッジベースからナレッジを取得
- ルールセットを使用して行動ガイドラインに従う
- オプションでツールを介してアクションを実行します

アシスタントのコアコンポーネントを構成し、実稼働またはテストで使用する準備をします。

アシスタントを作成する理由

アシスタントは、AI Factory のコンテンツと機能の**インタラクティブレイヤー**を提供し、組織のナレッジとサービスへの信頼できる会話型アクセスを可能にします。

彼らは以下を統合します

- 大規模言語モデル LLM
- ナレッジベース
- ルールセット
- ツール
- メモリ状況依存のインタラクション用

アシスタントを使用すると、エンドユーザーは**ガイド、管理、および便利な方法**で AI と対話でき、内部コパイロット、サポートエージェント、ナレッジアシスタントなどのアプリケーションを強化します。

背景については、以下を参照してください。

- *Assistants Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- *Hybrid Manager: Using Gen AI Builder*

複雑さと完了までの時間

- 複雑さ 中程度
- 推定時間 10～15 分

アシスタントを作成する場合

- 会話型インターフェイスを介してナレッジベースを公開したい場合。
- ルールセットを介して AI の動作とトーンを制御したい場合。
- AI がツールを介して外部タスクを実行する必要がある場合。
- AI を活用したエクスペリエンスコパイロット、チャットボット、API エージェントなどを構築するための最初のステップとして。

前提条件

- 少なくとも 1 つのナレッジベースオプションですが強くお勧めします。
- 1 つ以上のルールセットオプションですが、強くお勧めします。
- オプション設定され使用可能なツール。

アシスタントを作成する方法

1.[Assistants] に移動します

- Gen AI Builder コンソールで、 **Assistants** に移動します。
- これは通常、 **APPS** または **Agent Studio** の下にあります。

2. 新しいアシスタントを作成します

- **Create Assistant** をクリックします。

3. アシスタントフィールドの構成

お名前必須

- 一意の説明的な名前を指定します。
- 例 PG Financial Client Support AI

説明オプション

- アシスタントの目的と対象者について説明します。
- 例 Virtual support agent for PG Financial clients.

LLM モデル選択必須

- アシスタントを強化するために大規模言語モデル LLM を選択します。
- オプションの例
- GPT-4
- クロード 3 ソネット
- Gemini Pro
- *Model Serving Reference Manual* を介して展開されたカスタムモデル

システムプロンプト/指示オプション

- アシスタントのペルソナまたは高レベルの動作を定義します。
- 例 You are a helpful and friendly customer support agent for PG Financial.

ナレッジベースオプション

- 1 つ以上の既存のナレッジベースを選択して、**検索拡張生成 RAG** を有効にします。
- 例 PG Financial Product FAQs KB、PG Financial Account Services KB

ルールセットオプション

- 1 つ以上の既存のルールセットを選択して、動作ガイドラインを定義します。
- 例 Polite Support Tone、Financial Advice Disclaimer

ツールオプション

- アシスタントが呼び出すことができるツールを選択しますツールが構成されている場合。
- ツール例
- 電卓
- カレンダーのルックアップ
- カスタム API コネクタ

SDK リファレンス

- *Structures Reference Manual*
- Assistant Drivers
- Conversation Memory Drivers
- *Tools Reference Manual*

メモリ構成オプション、詳細

- アシスタントが会話のコンテキストを記憶する方法を構成します。
- 一般的なオプション
- 覚えるターン数
- 要約または要約+切り捨て戦略

その他の詳細設定オプション

- 温度 応答の創造性を制御します
- Top_p 核サンプリングを制御します
- ツールの使用設定
- RAG 動作設定高度なチューニング

4. 作成を完了する

- **Create** をクリックして、アシスタントを保存します。
- 新しいアシスタントは **アシスタント** リストに表示されます。

例サポートアシスタントを作成する

1. **Assistants** に移動します。
2. **Create Assistant** をクリックします。
3. フィールドを構成します。
 - 名前 PG Financial Client Support AI
 - 説明 Virtual agent providing customer support for PG Financial.
 - LLM モデル GPT-4
 - システムプロンプト You are a polite and helpful customer support agent for PG Financial.
 - ナレッジベース
 - PG Financial Product FAQs KB
 - PG Financial Account Services KB
 - ルールセット
 - Polite Support Tone
 - Financial Advice Disclaimer
 - ツール
 - Account Balance Lookup API 例

4. **Create** をクリックします。
5. 次の手順
 - ビルトインテストチャット UI を使用してアシスタントをテストします。
 - 結果に基づいて、ルールセット、ツール、およびメモリ設定を調整します。

障害対応

アシスタントが予想通りに応答しない

- 次のことを確認します。
- ナレッジベースには、関連するコンテンツが含まれています。
- 適切なルールセットが割り当てられます。
- システムプロンプトは目的の動作と一致します。

RAG の動作が一貫していない

- Retriever 構成とナレッジベースのインデックス作成を確認します。
- ツールを使用する場合、正しく構成されていることを確認します。

メモリが予想通りに動作しない

- メモリ設定を確認します。
- 必要に応じて、要約パラメーターを調整します。

ツールが機能しない

- ツール構成と API の可用性を確認します。
- アシスタントにツールが割り当てられていることを確認します。

関連トピック

- *Assistants Reference Manual*
- *Rulesets Manual*
- *Knowledge Bases Reference Manual*
- *Retrievers Reference Manual*
- *Structures Reference Manual*
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- Hybrid Manager: Using Gen AI Builder

How to integrate with your apps

Customer Service Assistant Implementation Guide

この実装ガイドでは、ハイブリッドマネージャーの統合 AI 機能を使用して、ガバナンスされたカスタマーサービスアシスタントを構築する方法を示します。アシスタントは、組織のナレッジベースから正確な情報を取得し、制御された環境内に展開されたモデルを使用して応答を生成します。

実装結果 包括的なナレッジの取得、管理された応答生成、および完全な運用の可視性を備えた Agent Studio の本番対応アシスタント。

SDK リファレンス

- *Structures Reference Manual*
- *Tools Reference Manual*
- RAG Engines

前提条件とアーキテクチャ要件

インフラストラクチャの依存関係

- AI Factory 機能が有効になったハイブリッドマネージャークラスター
- モデル提供とエンベディング生成のための十分なコンピューティングリソース
- 組織データソースへのネットワーク接続
- Gen AI Builder および Agent Studio 操作のための適切な権限

アクセス制御要件

- ナレッジベースの作成と管理のための Gen AI Builder の権限
- アシスタントの構成とテストのための Agent Studio アクセス
- 推論エンドポイントを展開またはアクセスするためのモデル提供権限
- 組織のセキュリティポリシーに合わせたデータソースアクセス

データの準備

ドキュメント、FAQ データベース、ポリシー文書、手順ガイドなどのカスタマーサービスコンテンツを準備します。コンテンツは、明確なソースの帰属を使用して、最適な取得パフォーマンスを得るように構成する必要があります。

推奨されるコンテンツタイプ

- 明確なセクション階層を含むカスタマーサポートドキュメント
- 質問と回答のペアを含む FAQ データベース
- 構造化された手順を含むポリシー文書
- ステップバイステップの指示を含むトラブルシューティングガイド

実装ワークフロー

フェーズ 1 ナレッジベースの構成

ナレッジベースは、組織のコンテンツ全体のセマンティック検索を介して、アシスタントの応答のための事実の基礎を提供します。

データソースの統合

1. コンテンツ評価

- 既存のカスタマーサービス文書の完全性と正確性を評価する
- 現在のコンテンツカバレッジのギャップを特定する
- 知識の鮮度を維持するためのコンテンツ更新手順を確立します

2. 処理構成

- ドキュメント構造に基づいて、適切なチャンク戦略を構成する
- ソース属性のメタデータ抽出を実装する
- 組織モデルを使用してエンベディング世代を確立する

3. 品質検証

- 代表的な顧客クエリを使用したテスト取得精度
- ソースの帰属と引用の生成を検証する
- 取得レイテンシーのパフォーマンスベースラインを確立する

実装リソース

- *Create a Knowledge Base in Gen AI Builder*
- Pipeline Knowledge Base Configuration

自動処理の実装

自動コンテンツの更新を構成して、手動介入なしでナレッジベースの精度を維持します。

```
# Example auto-processing configuration

processing_schedule: "0 2 ** *" # Daily at 2 AM
content_sources:
  - type: "object_storage"
    path: "s3://customer-docs/support/"
    include_patterns: ["*.pdf", "*.docx", "*.md"]
update_strategy: "incremental"
embedding_refresh: "on_change"
```

重要な考慮事項

- 更新頻度と計算リソース消費のバランス
- 変更検出を実装して、不要な処理を回避する
- コンテンツの更新による埋め込みドリフトの経時的な監視

フェーズ 2 アシスタントの動作構成

ルールセットの開発

ルールセットはアシスタントの動作を制限して、応答が組織の標準およびコンプライアンス要件と一致するように保証します。

行動ガイドライン

- 顧客対応に適したプロフェッショナルな口調
- 適切な免責事項を含む正確な情報配信
- アシスタント機能を超えた複雑なクエリのエスカレーション手順
- 機密の顧客情報のデータプライバシー制御

ルールセット構造の例

```
# Customer Service Assistant Guidelines
```

```
## Response Standards
```

- Provide accurate, helpful information based solely on knowledge base content
- Maintain professional, empathetic tone in all interactions
- Include source citations for all factual claims
- Acknowledge limitations when information is unavailable

```
## Escalation Criteria
```

- Complex technical issues requiring human expertise
- Account-specific information requiring authentication
- Complaints or sensitive customer concerns
- Requests for policy exceptions or special handling

(次のページに続く)

(前のページからの続き)

Compliance Requirements

- Never request or process personally identifiable information
- Follow data retention policies for conversation logs
- Maintain audit trails for all customer interactions

取得ストラテジーの構成

取得パラメーターを構成して、カスタマーサービスクエリの精度と関連性を最適化します。

キー構成領域**1. 類似性のしきい値**

- 最小の関連性スコアを設定して、低品質の一致を防止
- リコール関連情報の検索と精度ノイズの回避のバランス
- さまざまなコンテンツタイプに異なるしきい値を設定する

2. 結果の制限

- 応答生成要件に基づいて top-K 値を構成する
- 大規模な結果セットの計算オーバーヘッドを検討する
- クエリの複雑度に基づいて動的な調整を実装する

3. コンテンツフィルタリング

- コンテンツの最新性のためにメタデータベースのフィルターを適用する
- ユーザー権限に基づいてアクセス制御フィルタを実装する
- さまざまなクエリカテゴリのコンテンツタイプ設定を構成する

取得構成例

```
{
  "similarity_threshold": 0.75,
  "max_results": 8,
  "content_filters": {
    "recency_days": 365,
    "content_types": ["faq", "documentation", "policy"],
    "access_level": "customer_facing"
  },
  "reranking": {
    "enabled": true,
    "model": "organizational_rerank_model"
  }
}
```

(次のページに続く)

(前のページからの続き)

```
}  
}
```

フェーズ 3 モデルの統合

モデルの選択と展開

応答品質、遅延、運用コストなどの顧客サービス要件に基づいて、適切な言語モデルを選択します。

カスタマーサービスのモデル特性

- 顧客の問い合わせに対する適切な応答の長さ
- プロのトーン生成機能
- 取得したコンテキストからの正確な情報合成
- さまざまな負荷条件下での一貫したパフォーマンス

展開の考慮事項

- 予想される同時会話のリソース割り当て
- ピークサポート期間の自動スケーリング構成
- モデルの可用性とパフォーマンスのヘルスマニタリング
- モデルが使用不可になった場合のフォールバック手順

外部プロバイダーの統合

組織ポリシーで外部モデルの使用が許可されている場合、適切なアクセス制御とモニタリングを構成します。

セキュリティ要件

- 組織のセキュリティポリシーに合わせた API キー管理
- 監査とトラブルシューティングのための要求/応答ログ
- 外部プロバイダーに送信される情報のデータ取り扱いポリシー
- コストの監視と使用制御

フェーズ 4 アシスタントの組み立てとテスト

コンポーネントの統合

ナレッジベース、ルールセット、レトリバー、およびモデルを Agent Studio 内の機能的なアシスタント構成に組み立てます。

統合チェックリスト

- ☐ ナレッジベースの接続と取得の検証
- ☐ ルールセットの適用と動作の検証

- [] モデルエンドポイントのアクセシビリティと応答生成
- ☒ 外部システム接続のためのツール統合該当する場合
- ☒ 引用文献と出典の帰属の精度

包括的なテストプロトコル

体系的なテストにより、運用展開前にアシスタントの信頼性が保証されます。

テストカテゴリ

1. 機能の検証

- 多様な顧客サービスシナリオにわたるクエリ処理
- グラウンドトゥルースドキュメントと比較した応答精度
- すべての事実に基づく主張の引用検証
- ナレッジベースのスコープ外のクエリのエラー処理

2. パフォーマンステスト

- 通常およびピーク負荷条件での応答レイテンシー
- 同時ユーザー処理機能
- 運用中のリソース使用率監視
- スケーリング動作の検証

3. 行動のコンプライアンス

- 会話シナリオ全体でのルールセットの遵守
- 適切なエスカレーショントリガー
- 一貫した口調とプロ意識
- データプライバシーポリシーのコンプライアンス

テストの実装

```
# Example testing framework structure

class AssistantTestSuite:
    def test_response_accuracy(self):
        """Validate responses against known correct answers."""
        test_queries = [
            "What is the return policy for electronics?",
            "How do I reset my account password?",
            "What are the shipping options available?"
        ]

        for query in test_queries:
```

(次のページに続く)

(前のページからの続き)

```
response = self.assistant.query(query)
assert self.validate_accuracy(response)
assert self.validate_citations(response)

def test_performance_characteristics(self):
    """Measure response times and resource usage."""
    # Implementation details for load testing

def test_behavioral_compliance(self):
    """Verify adherence to organizational guidelines."""
    # Implementation details for compliance testing
```

構成の最適化

パフォーマンスチューニングパラメーター

取得構成

- top_k 5～10 の結果から開始し、応答品質に基づいて調整します
- similarity_threshold 0.7 から始まり、増加してノイズを削減するか、減少してリコールを向上させます
- context_window 包括的なコンテキストと生成速度のバランスをとります

モデルパラメーター

- temperature 一貫した事実に基づいた回答には 0.1～0.3 を使用します
- max_tokens 一般的な応答長要件に基づいて構成
- timeout カスタマーエクスペリエンスの期待に応じて適切な値を設定します

自動処理設定

- コンテンツの変更パターンに合わせた更新頻度
- 処理操作のためのリソースの割り当て
- 処理障害の監視しきい値

運用モニタリング

制作アシスタント業務の包括的なモニタリングを実装します。

主要なメトリック

- ユーザーフィードバックを介した応答精度率
- エスカレーションなしのクエリ解決率
- 平均応答レイテンシーと 95 パーセンタイル測定

- ナレッジベースのヒット率と検索効果
- モデルのパフォーマンスと可用性の統計

アラート構成

- サービスレベル目標を超える応答遅延
- ナレッジベースの取得の失敗またはパフォーマンスの低下
- モデルエンドポイントの非可用性またはエラー率
- 潜在的な問題を示す異常なクエリパターン

トラブルシューティングフレームワーク

一般的な問題と解決策

不正確な応答

- **症状** アシスタントが不正確または古い情報を提供します
- **調査** ナレッジベースのコンテンツの精度と最新性を検証します。
- **解決策** ソースドキュメントの更新、類似性のしきい値の調整、コンテンツのチャンク化の改善

ソースの引用がありません

- **症状** 回答に適切なソースの帰属がありません。
- **調査** 取得構成と引用生成設定の確認
- **解決策** ナレッジベースのメタデータを検証し、取得パラメーターを調整します

遅い応答パフォーマンス

- **症状** 応答遅延が許容可能なしきい値を超えます
- **調査** モデルの推論時間、取得レイテンシー、およびリソース使用率を監視します
- **解像度** 取得パラメーターの最適化、モデルリソースのスケーリング、応答キャッシュの実装

不適切なエスカレーション動作

- **症状** アシスタントが処理する必要があるクエリをエスカレートするか、複雑な問題のエスカレーションに失敗する
- **調査** ルールセット構成とエスカレーショントリガーロジックを確認します
- **解決策** 行動ガイドラインを調整し、信頼性しきい値を調整します。

診断手順

応答品質評価

1. 回答を文書化された正解と比較する
2. ソース帰属の精度を検証する

3. 応答の一貫性とプロフェッショナルなトーンを確認します
4. 組織のガイドラインへのコンプライアンスを確認する

パフォーマンス分析

1. エンドツーエンドの応答レイテンシーを測定する
 2. コンポーネント固有のパフォーマンスへの寄与を分析する
 3. リソース使用パターンの監視
 4. 負荷の下でのスケーリング動作を評価する
-

運用展開

導入準備状況チェックリスト

技術的検証

- ☐ サポートされているすべてのクエリタイプの包括的なテスト
- ☒ 予想される負荷条件でのパフォーマンス検証
- ☐ 組織システムとの統合テスト
- ☐ セキュリティレビューとコンプライアンス検証

運用の準備

- ☐ モニタリングとアラート構成
- ☐ サポート手順とエスカレーションワークフロー
- ☐ メンテナンスと障害対応のためのドキュメント
- ☒ ユーザートレーニングと導入計画

継続的改善フレームワーク

パフォーマンスモニタリング

- ユーザーのインタラクションと満足度メトリックの定期的な分析
- 検索分析を介したナレッジベースの有効性
- モデルのパフォーマンス傾向と最適化の機会

コンテンツ管理

- ナレッジベースコンテンツの体系的な確認と更新
- 未解決の顧客クエリに基づいたギャップ分析
- 新しいドキュメントとポリシーの更新の統合

システムの進化

- モデルのアップグレードの評価とテストの手順
 - ユーザーのフィードバックに基づいた機能強化
 - 追加の組織システムとの統合の機会
-

次のステップと高度な機能

機能の拡張

ツール統合 *Tools Reference Manual* を使用して、アカウント検索、チケット作成、およびワークフローの自動化のために外部システムを接続します。

マルチモーダルサポート ドキュメントのアップロード、画像クエリ、および音声インタラクションを処理する機能を拡張します。

高度な分析 顧客のインサイトとサービスの最適化のために会話分析を実装します。

学習リソース

基本的なコンセプト

- AI ファクトリーを包括的に理解するための *Learning Paths*
- 高度なコンテンツ処理のための *Pipelines reference - Preparers*

オペレーショナルエクセレンス

- モデル提供の最適化とスケーリング戦略
 - 複雑な知識領域の高度な検索技術
 - 既存のカスタマーサービスインフラストラクチャとの統合パターン
-

この実装ガイドは、データ、モデル、操作手順の組織的な制御を維持しながら、本番対応のカスタマーサービスアシスタントを展開するための包括的な基盤を提供します。

How to operate and observe

How-To View and Manage Threads in Gen AI Builder in Hybrid Manager

前提条件 AI Factory を有効にしてハイブリッドマネージャー UI にアクセスします。/edb-postgres-ai/1.3/hybrid-manager/ai-factory/を参照してください。

このガイドでは、Gen AI Builder 内の会話スレッドにアクセスして分析し、品質保証、コンプライアンスモニタリング、およびアシスタントの最適化アクティビティをサポートするための体系的な手順を提供します。

前提条件

システム要件

- 適切なスレッド表示権限を備えた Gen AI Builder アクセス
- スレッドデータを生成する少なくとも 1 つの完了したアシスタント会話
- アシスタントの構成と予想される動作パターンの理解

ユーザーロール この手順は以下に適用されます。

- アシスタントのパフォーマンスを評価する品質保証チーム
- 会話監査を実施するコンプライアンス責任者
- アシスタントの動作の問題をデバッグする開発者
- インタラクションパターンを分析する AI アーキテクト

アクセススレッド管理

1.[スレッドインターフェイス]に移動します

Gen AI Builder コンソール内の **Threads** セクションにアクセスします。スレッド管理は、通常、メインナビゲーション構造内またはアプリケーション管理カテゴリの下に表示されます。

2. レビュースレッドカタログ

スレッドカタログには、包括的な会話のメタデータが表示されます。

スレッド識別

- **Name** ユーザー割り当てまたはシステム生成のスレッド識別子
- **エイリアス** プログラムによるアクセスのための一意のシステム識別子
- **Created** 初期スレッド作成タイムスタンプ
- **更新** 最新のメッセージまたは変更のタイムスタンプ

ナビゲーションコントロール

- **検索機能** キーワード、エイリアス、またはメタデータフィルタを使用してスレッドを見つけます。
- **ページネーション** 大規模なスレッドコレクションを効率的にナビゲート
- **並べ替えオプション** 作成日、アクティビティ、またはその他の基準でスレッドを並べ替えます

スレッド分析プロシージャ

3. ターゲットスレッドを見つけます

検索戦略

- **キーワード検索** 会話コンテンツまたはトピックキーワードを使用します
- **時間フィルタリング** 作成または更新の日付範囲でフィルタリングします

- アシスタントフィルタリング結果を特定のアシスタントインスタンスに絞り込む
- ユーザー識別 ユーザー識別子またはセッションデータによる検索

検索例

Search Terms:

- "account opening inquiry"
- Assistant: "customer_service_bot"
- Date Range: "Last 7 days"
- Thread Alias: "thread_abc123"

4. スレッドの詳細にアクセスします

カタログからスレッドを選択して、以下を含む詳細な会話ビューを開きます。

スレッド概要パネル

- 完全なスレッドのメタデータと識別情報
- スレッドのライフサイクルタイムスタンプと変更履歴
- 関連するアシスタントの構成とバージョン情報

アクションコントロール

- スレッド管理操作編集、削除、エクスポート
- 共有およびコラボレーション機能
- アーカイブと保持の制御

5. 会話フローを分析する

メッセージシーケンス分析体系的な検査を通じて会話の進行を確認します。

1. メッセージナビゲーションシーケンスコントロールを使用して、個々のメッセージをステップ実行します
2. コンテンツインスペクション 完全なメッセージのコンテンツとフォーマットを検査します
3. メタデータのレビュー メッセージのタイムスタンプ、処理時間、およびシステム情報を分析します
4. ツール実行追跡 ツールの呼び出しと結果の統合を特定します

メッセージの詳細検査 会話ターンごとに、以下を評価します。

ユーザー入力分析

- 意図の明確性と具体性
- 会話ターン全体でのコンテキストの保持

- 情報の完全性と正確性

アシスタントの応答評価

- 応答の関連性と精度
- 出典の帰属と引用の質
- 設定されたルールセットとのトーンの一貫性
- ナレッジベースの活用効果

ツール統合評価

- ユーザー要求に対するツール選択の適切性
- 会話コンテキストからのパラメーター抽出精度
- 結果の統合とプレゼンテーションの品質
- ツールの実行失敗時のエラー処理

6. パフォーマンス評価

品質メトリックの評価

Assessment Area	Evaluation Criteria
Intent Recognition	Accuracy of user request interpretation
Knowledge Retrieval	Relevance and accuracy of information sources
Response Quality	Coherence, completeness, and helpfulness
Behavioral Compliance	Adherence to configured rulesets and policies
Tool Utilization	Appropriate tool selection and execution

行動分析

- **ルールセットコンプライアンス** 設定された動作ガイドラインの遵守を確認します。
- **トーンの一貫性** 組織の標準とのコミュニケーションスタイルの調整を評価します
- **エスカレーション処理** 適切なエスカレーショントリガー認識を確認します。
- **プライバシー保護** 機密情報の適切な取り扱いを確認します

スレッド管理操作

7. 管理アクションを実行する

スレッドの変更

- **名前の更新** 組織を改善するためにスレッド識別子を変更します
- **メタデータの強化** 分類タグまたは組織ラベルを追加します
- **アクセス制御** コンテンツの機密性に基づいて表示権限を調整します

データマネジメント

- エクスポート操作外部分析またはアーカイブのためのスレッドエクスポートを生成します
- リテンション管理 組織のデータ保持ポリシーを適用します
- 削除手順確立されたデータガバナンスプロトコルに従って、スレッドを削除します。

品質保証ワークフロー

1. パフォーマンス評価の結果を文書化する
2. アシスタント構成の改善の機会を特定する
3. コンプライアンス違反またはポリシー遵守の問題を報告する
4. ナレッジベースの更新またはルールセットの変更を推奨します

実践例 金融サービスの照会分析

シナリオコンテキスト 新しい普通預金口座の開設に関する顧客の問い合わせは、金融サービスアシスタントによって処理されました。

分析手順

スレッドの場所

1. Threads カタログに移動します
2. 検索パラメーター `savings account opening`
3. アシスタントによるフィルター `financial_services_bot`
4. 結果から関連するスレッドを選択します

会話の評価

1. 初期要求評価
 - ユーザーの目的 口座開設情報要求
 - コンテキスト 特定の製品に興味がある新規顧客の問い合わせ
2. アシスタントの応答分析
 - 製品情報の精度検証
 - 金融サービスの通信要件のコンプライアンス
 - 現在の製品の詳細に関するナレッジベースの利用
3. マルチターンインタラクションレビュー
 - フォローアップの質問処理の有効性
 - 会話ターン全体でのコンテキストの保持
 - 口座開設手続きのための適切な情報収集

品質評価ドキュメント

- **長所** 正確な製品情報の配信、準拠した通信トーン
- **改善の余地** 資格要件の説明の強化
- **推奨事項** 現在のプロモーションオファーを含むナレッジベースの更新

一般的な問題のトラブルシューティング

スレッドアクセスの問題

スレッドが見つかりません

- 検索パラメーターの精度と範囲を検証する
- スレッド作成完了の確認処理が遅延する可能性があります。
- スレッドの可視性に関するユーザー権限を確認します

不完全なメッセージデータ

- 会話完了ステータスの検証
- ツール実行ログ構成の検証
- 最近の会話のシステム処理ステータスを確認します

分析の課題

パフォーマンスの問題

- **応答精度の問題** ナレッジベースのコンテンツカバレッジと取得構成を確認します
- **動作違反** ルールセットの構成と適用メカニズムを調べます
- **ツールの実行障害** ツール構成と外部システムの接続を分析します

解決戦略

- 会話のコンテキストに関する特定の問題を文書化する
- トラブルシューティングのための相互参照アシスタント構成
- システム的な問題を開発チームにエスカレーションする
- 根本原因に対処しながら、一時的な回避策を実装します

運用のベストプラクティス

系統的な分析アプローチ

定期的なレビュー手順

- 一貫したスレッドレビュースケジュールを確立する
- 標準化された評価基準の導入
- 傾向分析のために調査結果を体系的に文書化する

- コンプライアンスを目的とした評価記録の維持

品質向上の統合

- スレッド分析の結果をアシスタントの最適化に使用する
- ナレッジベースの更新が必要なパターンを特定する
- 動作分析に基づいてルールセットの変更を推奨します
- 体系的な改善のために開発チームと調整する

データ管理の考慮事項

プライバシーとセキュリティ

- 機密の会話に対する適切なアクセス制御を実装する
- 組織のデータ保持ポリシーに従う
- 監査および分析を目的とした安全な輸出手順
- レビュープロセス中の機密保持

コンプライアンス要件

- 規制遵守のための文書レビュー手順
- スレッドアクセスと分析アクティビティの監査証跡を維持する
- コンプライアンス違反報告への体系的なアプローチの実施
- ポリシーの遵守のための法務およびコンプライアンスチームとの調整

関連ドキュメント

スレッド分析リソース

- *Threads Reference Manual* 包括的なスレッドシステムの概要
- *Assistants Reference Manual* アシスタント最適化プロシージャ
- *Rulesets Manual* 行動コンプライアンス評価方法

システムインテグレーション

- Gen AI Builder Overview プラットフォームアーキテクチャとコンポーネントの関係
- Hybrid Manager integration インフラストラクチャ管理プロシージャ

3.5 AI Factory Pipelines

AI Factory Pipelines は、EDB Postgres® AI EDB PG AI のコア機能であり、取得拡張生成 RAG を含む Postgres クラスタで直接インテリジェントに自動化された AI データワークフローを有効にします、ナレッジベース**、およびベクターベースのアプリケーション。

パイプラインは、Postgres の AI データの準備、管理、および提供の複雑さを抽象化します。**Vector Engine** と統合し、データがデータベースまたは信頼できるオブジェクトストレージ内にある**ソブリン AI** パターンを有効にします。

3.5.1 なぜパイプラインなのか？

最新の AI ワークロード、特に RAG と Gen AI には、以下が必要です。

- クリーンな前処理データ
- 一貫したエンベディング
- 効率的なベクトルインデックス
- ソースデータが変更された場合のリアルタイム更新

これらの手順を手動で管理するのは複雑で、エラーが発生しやすくなります。Pipelines はこのライフサイクルを自動化します。

- データの準備とクリーンアップ
- エンベディング生成を実行します
- ベクトルインデックスを維持します
- 自動処理とナレッジベースをサポート

Pipelines は、古いエンベディングから「幻覚」を排除し、信頼性の高い Gen AI の構築を簡単にします。

3.5.2 あなたにできること

パイプラインを使用すると、次のことができます。

- テキストおよび画像データの **ベクター埋め込みパイプライン**を自動化する
- リアルタイムまたはバッチ更新で **ナレッジベース** を維持する
- *Vector Engine (PGvector)* を使用して **セマンティックおよび類似性検索** を実行する
- **Gen AI Builder** およびその他の AI Factory コンポーネントとの統合
- データに基づいて、完全に監査可能、制御された **ソブリン AI** システムを構築する

3.5.3 その仕組み

準備者

プリペアラールは、ソースデータをクリーンアップ、チャンク分け、および埋め込み用に準備する方法を定義します。次のプリペアラールを作成できます。

- Postgres テーブル `aidb.create_table_preparer()`
- オブジェクトストレージ S3 互換 `aidb.create_volume_preparer()`

作成したら、次を実行できます。

- 大量処理 `aidb.bulk_data_preparation()`
- 自動処理 `aidb.set_auto_preparer()` 埋め込みを自動的に最新の状態に保ちます。

詳細については、[Pipelines reference - Preparers](#) を参照してください。

ナレッジベース

ナレッジベースは、RAG の単純な抽象化を提供します。

- ナレッジベースの定義 `aidb.create_table_knowledge_base()` 、`aidb.create_volume_knowledge_base()`
- エンベディングとベクターインデックスを自動的に管理
- セマンティック検索に `aidb.retrieve_key()` および `aidb.retrieve_text()` を使用する

サポート

- テキスト、画像データ
- マルチモーダルモデル CLIP など
- Gen AI Builder の取得フローとの統合

Knowledge Base Concepts を参照してください。

自動処理

パイプラインは、柔軟な自動処理モードをサポートしています

- 同期または非同期
- テーブルまたはボリュームソース
- 設定可能なバッチサイズとトリガー

詳細は、[Pipeline auto-processing](#) を参照してください。

3.5.4 誰がパイプラインを使用する必要がありますか？

パイプラインは次の場合に最適です。

- エンタープライズナレッジベースを作成する Gen AI Builders
- セマンティック検索パイプラインを構築するデータエンジニア
- 自動化された RAG パイプラインを必要とする AI チーム
- 規制された環境で ソブリン AI を実装するアーキテクト

3.5.5 パイプラインを使用する場合

次のことが必要な場合にパイプラインを使用します。

- データの変更に応じて埋め込みを最新の状態に保つ
- 古いまたは不完全な埋め込みによって発生する RAG エラーを削減します
- テキストと画像をまたいだ強力なマルチモーダル Gen AI 検索
- 埋め込みプロセスの完全な監査可能性とガバナンスを維持する

3.5.6 構成値の説明

- チャンクサイズとオーバーラップチャンクが小さいほど、グラウンディングが向上します。オーバーラップは、チャンク全体でコンテキストを保持するのに役立ちます。文書構造に基づいて調整します。
- 埋め込みモデルの選択インジェストとクエリーに一貫したモデルを使用します。ドメインと言語に合わせて選択します。
- 自動処理レイテンシーデータ変更頻度と SLA に合わせて調整します。同期モードと非同期モードを検討します。
- バッチサイズと並列処理スループットは向上しますが、ストレージ、ネットワーク、およびメモリを監視します。
- 類似性しきい値と top-K しきい値が高いほどノイズが削減されます。リコール対レイテンシーの top - K を調整します。

3.5.7 詳細はこちら

- Knowledge Base Concepts
- *Pipelines reference - Preparers*
- *Pipeline auto-processing*
- *Supported Embedding Models*
- *Vector Engine (PGvector)*

AI Accelerator Pipelines は、Postgres データに基づいて、信頼できるスケーラブルな **ソブリン AI** ソリューションを構築するのに役立ちます。AI パイプラインをクリーンで最新の運用準備ができた状態に保ちます。

Start here

AI Accelerator Pipelines overview

EDB Postgres AI - AI Accelerator の一部として、Pipelines は AI データの操作の複雑さを抽象化します。Postgres を AI データ管理のための強力なプラットフォームに変換し、PGvector からのベクトル検索と複雑な AI ワークフローの自動化を組み合わせます。

パイプライン、自動処理、およびインテリジェントなナレッジベース

Pipelines は、埋め込み生成、ストレージ、インデックス作成を介してデータ準備のライフサイクル全体を処理します。サポートされているモデルのリストから目的のモデルを選択するか、OpenAI API 互換の外部モデルに接続できます。次に、データの取り込みから効率的な類似性検索まで、その他すべてを Pipelines に管理させることができます。

データの準備により、パイプラインは Postgres から直接 AI アクセラレーター内の一般的な前処理手順を実行できます。パイプラインは、この前処理の準備を自動的に実行し、処理されたデータを最新の状態に保つこともできます。

自動処理により、Pipelines はエンベディングの作成と更新を自動化し、手動のエンベディング管理なしでベクターストアを最新の状態に保ちます。このアプローチにより、GenAI エラーや幻覚を引き起こす古いデータのリスクが削減されます。

インテリジェントなナレッジベース機能を使用すると、単一のナレッジベースファンクション呼び出しでテキストと画像データの類似性検索とセマンティック検索を実行できます。これらの検索は、データが Postgres またはオブジェクトストレージに保存されているかどうかにかかわらず可能です。

パイプラインの仕組み

Pipelines は、Postgres サーバーに埋め込まれている Pipelines promoteb 拡張機能を介して機能を提供します。この拡張機能は、Postgres データベース内の AI データを操作できるようにする一連の機能とビューを提供します。

準備者

Pipelines fencedb 拡張機能は、ソース AI データの特定の種類と場所に対して作成できる **プリペアラ** の概念を導入します。このデータは、Postgres テーブルの通常の列または S3 互換のオブジェクトストレージバケットのいずれかに存在できます。

プリペアラは、エンベディングの生成に使用するソースデータに一般的な前処理手順を実行する機能をカプセル化します。アプリケーションは、Postgres テーブルソースの場合は `aidb.create_table_preparer()` ファンクションを介して、S3 またはローカルファイルシステムに外部に保存されているソースデータの場合は `aidb.create_volume_preparer()` を介してプリペアラを作成するだけです。

デフォルトでは、自動処理がないため、アプリケーションは `aidb.bulk_data_preparation()` を使用して、ソースの場所にあるすべての既存のデータの大量データの準備処理をトリガーできます。自動処理は、Postgres テーブルのソースデータに `aidb.set_auto_preparer()` を使用して構成することもできます。プリペアラからの出力は、ナレッジベースの入力にフィードできます。

ナレッジベース

Pipelines agentb 拡張機能は、AI データの特定の種類と場所に対して作成できる **ナレッジベース** の概念を導入します。このデータは、Postgres テーブルの通常の列または S3 互換のオブジェクトストレージバケットのいずれかに存在できます。現在、Pipelines は、非構造化プレーンテキストドキュメントと一連の画像形式をサポートしています。

ナレッジベースは、提供されたソースの場所の AI データを類似性を介して検索および取得可能にするために必要なすべての処理をカプセル化します。ナレッジベースを作成するには、アプリケーションは Postgres テーブルの場合は `aidb.create_table_knowledge_base()` ファンクションを、S3 またはローカルファ

イルシステムの外部に保存されているデータの場合は `aidb.create_volume_knowledge_base` を使用します。

自動処理

自動処理には、同期および非同期、バッチ処理を含む幅広い構成性があります。これらの機能の可用性は、パイプラインのタイプナレッジベースまたはプリペアラ、およびデータソースと宛先のタイプテーブルまたはボリュームによって異なります。

詳細については、[Pipeline auto-processing](#) を参照してください。

インテリジェントなナレッジベース

ナレッジベースが作成され、すべての埋め込みが最新の状態になると、アプリケーションは `aidb.retrieve_key()` を使用して、クエリ入力を提供することにより類似性検索と取得を実行できます。一致したアイテムの ID を結果ます。ナレッジベースがテキストデータ用に作成されている場合、クエリ入力もテキスト用語です。画像ナレッジベースの場合、クエリ入力は画像です。CLIP などのナレッジベースでマルチモーダルモデルを使用している場合、テキストまたは画像をソースおよびクエリ入力として使用できます。このアプローチでは、テキストを使用して画像を検索できます。その逆も可能です。

aidb ナレッジベースは、埋め込みに使用したものと同一エンコーダー LLM をクエリ入力に使用し、類似性検索を実行し、ソースの場所から類似データのランク付けリストを結果ます。Postgres テーブルの場合、`aidb.retrieve_text()` を使用してソーステーブルから実際のデータを取得することもできます。

Pipelines は、現在、HuggingFace のオープンエンコーダー LLM の広範なリストと OpenAI エンコーダーをサポートしています。HuggingFace LLM は Postgres ノードでローカルに実行されますが、OpenAI エンコーダーは OpenAI クラウドサービスへの呼び出しが関係します。`aidb.model_providers` テーブルは、現在サポートされているモデルプロバイダーのリストを示します。

Compatibility

helpb

サポートされているプラットフォーム

- X86/64 上の Ubuntu 22.04LTS および 24.04LTS。
- X86/64 および ARM64 での Debian 12 Bookworm。
- X86/64 上の Redhat / RHEL 9/8。
- ARM64 での Redhat/RHEL 9。

現在サポートされていません

- SLES。
- 現在のバージョン 12 より前の Debian。
- 非 Linux プラットフォーム。

サポートされている PostgreSQL バージョン

- EDB Postgres Advanced Server バージョン 14、15、16、および 17。
- EDB Postgres 拡張バージョン 14、15、16、および 17。
- PostgreSQL 14、15、16、および 17。

pgfs

サポートされているプラットフォーム

- X86/64 上の Ubuntu 22.04LTS および 24.04LTS。
- X86/64 および ARM64 での Debian 12 Bookworm。
- X86/64 上の Redhat / RHEL 9/8。
- ARM64 での Redhat/RHEL 9。

現在サポートされていません

- SLES。
- 現在のバージョン 12 より前の Debian。
- 非 Linux プラットフォーム。

サポートされている PostgreSQL バージョン

- EDB Postgres Advanced Server バージョン 14、15、16、および 17。
- EDB Postgres 拡張バージョン 14、15、16、および 17。
- PostgreSQL 14、15、16、および 17。

Limitations

ブロック操作

Preparer パイプラインは現在非同期操作をサポートしていません。すべての操作はブロッキングであり、フォアグラウンドで実行されます。大量のデータの準備を実行すると、現在データベースまたはストレージにあるすべての項目の埋め込みが完了するまで、システムはブロックされます。

可観測性

Preparer パイプラインの可観測性は現在制限されています。Postgres サーバーログとコマンドの出力を参照してください。システム全体を監視するための単一のガラスはありません。

大規模なドキュメント

Pipelines は現在、大きなドキュメントのチャンク化を処理していません。

データフィルタリング

パイプラインは、行のコンテンツに基づいて SQL フィルターとビューを使用して埋め込みドキュメントを制限できますが、現在 S3 ストレージ内のデータのフィルタリングはサポートされていません。サブパスとプレフィックスフィルタリングの使用に制限されています。

モデルのロードバランシング

現在、モデルアクセスのためのロードバランシングメカニズムはありません。

データ形式

ナレッジベースパイプラインは、現在テキストと画像形式のみをサポートしています。構造化データ、ビデオ、オーディオを含む他の形式は、現在サポートされていません。

自動処理

- ライブ自動処理はテキストデータのみでサポートされています。画像データの埋め込みは、手動で計算するか、バックグラウンドの自動処理を使用して計算できます。
- バックグラウンド自動処理は、ナレッジベースパイプラインでのみサポートされており、プリペアラーパーパイプラインではありません。

アップグレード

aidb および pgfs 拡張機能をアップグレードする場合、現在 Postgres 拡張機能のアップグレードはサポートされていません。したがって、拡張機能の新しいバージョンにアップグレードする場合、拡張機能を削除して再作成する必要があります。

詳細およびコマンドについては、[Upgrading AI Accelerator Pipelines](#) を参照してください。

Get started

Getting started with Pipelines

どこから始めるべきか

[AI Accelerator Pipelines overview](#) から始めて、パイプラインとは何か、それがどのように動作するかを理解するのが最適です。

インストール

Pipelines は、EDB Postgres AI - AI Accelerator ツールスイートに含まれています。Pipelines をインストールするには、AI Accelerator Installation Guide の指示に従ってください。

パイプラインを使用する

Pipelines をインストールしたら、それを使用してデータを操作できます。

Postgres サーバーにログインし、Pipelines 拡張機能がインストールされていることを確認します。

```
CREATE EXTENSION aidb CASCADE;
```

この例は Postgres テーブルデータでのみ動作するため、pgfs 拡張機能をインストールする必要はありません。
例では、作業するテーブルを作成します。

```
CREATE TABLE products (
  id SERIAL PRIMARY KEY,
  product_name TEXT NOT NULL,
  description TEXT,
  last_updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
__OUTPUT__
CREATE TABLE
```

次に、それにいくつかのデータを挿入します。

```
INSERT INTO products (product_name, description) VALUES
  (Hamburger, A delicious combination of bread and meat),
  (Cheesburger, Improving on a classic, the cheese brings favorite flavors),
  (Fish n Chips, The fish is a little greasy and the chips do not help),
  (Fries, Never sure about these on their own, needs seasoning),
  (Burrito, Always ready for this parcel of edible wonder),
  (Pizza, It is very much a staple, but the rolled dough with toppings does not
↳inspire),
  (Sandwich, The blandest of offerings, the sandwich is predominantly boring
↳bread),
  (Veggie Burger, The ultra-processed vegetable product in this is neither
↳healthy nor delicious),
  (Kebab, Maybe one of the great edible treats, sliced lamb, salad and crisp
↳pitta);
__OUTPUT__
INSERT 0 9
```

結果は、いくつかのデータを含むテーブルです。食品とそれに関する非常に個人的な意見。

!!!注

この例では、単純な処理されたデータを使用します。パイプラインは、*Pipelines reference - Preparers* を介して複数の前処理ステップを処理することもできます。

Pipelines のコアは **ナレッジベース**です。ナレッジベースは、テーブル内のデータにアクセスし、AI ワークフローで使用する方法です。

Getting started with Pipelines

関連項目

[Pipelines models reference](#) では、OpenAI API 互換サービスからの外部モデルを含む、パイプラインを使用してより多くのモデルを作成する方法を学習できます。

[Pipelines reference - Preparers](#) では、エンベディングが計算される前にデータの前処理ステップを実行するプリペアラを作成する方法の詳細を学習できます。

Knowledge Bases では、外部データソース、ローカルファイル、または S3 ストレージでナレッジベースを使用する方法、およびナレッジベース機能を使用して必要なデータを取得する方法について詳しく学ぶことができます。

Installing AI Accelerator Pipelines

Pipelines は、拡張機能のセットとして配信されます。パイプラインを展開する方法に応じて、これらの拡張機能は展開プラットフォームプロバイダーがインストールするか、既にインストールされています。詳細については、展開プラットフォームプロバイダーに問い合わせてください。独自の Postgres サーバーを展開する場合、拡張機能を手動でインストールする必要があります。

- [AI Accelerator Pipelines](#)

パッケージがインストールされたら、Postgres の拡張機能をアクティブにすることにより [Completing and verifying the extension installation](#) できます。

既存のインストールがあり、新しいバージョンにアップグレードする場合は、[Upgrading AI Accelerator Pipelines](#) を参照してください。

AI Accelerator Pipelines

拡張機能パッケージのインストール

AI Accelerator Pipelines を使用するには、3 つの拡張機能パッケージが必要です。

- `helpb` — AI データベース拡張機能。
- `pgfs` — Postgres File System PGFS 拡張子。
- `ベクター` — PGvector 拡張子。

前提条件

パッケージのインストールを開始する前に、PostgreSQL サーバーを実行する必要があります。AI Accelerator Pipelines は、コミュニティ PostgreSQL サーバー、EDB Postgres Advanced Server サーバー、または EDB Postgres Extended サーバーにインストールできます。サーバーのセットアップにヘルプが必要な場合は、これらの製品のインストール手順を参照してください。AI Accelerator Pipelines パッケージをインストールするために必要になるため、サーバーをインストールするときに使用するパッケージ名をメモします。

- コミュニティ PostgreSQL の場合、パッケージ名は `postgresql-<version>` です。
- EDB Postgres Advanced Server の場合、パッケージ名は `edb-as<version>` の後に `-server` が続きます。

- EDB Postgres Extended の場合、パッケージ名は `edb-postgresextended-<version>` の後に `-server` が続きます。

パッケージ名が決定したら、オペレーティングシステムのパッケージマネージャーを使用してサーバーをインストールできます。

パッケージのインストール

サーバーが実行されたら、いくつかのパッケージをインストールする必要があります。パッケージは EDB リポジトリで入手でき、オペレーティングシステムの適切なパッケージマネージャーを使用してインストールできます。使用する

- `.deb` パッケージを使用した Debian/Ubuntu ベースのシステムの場合 `apt`
- `.rpm` パッケージを使用した Red Hat ベースのシステムの場合 `dnf`

パッケージ名は、オペレーティングシステムによって異なります。

コミュニティ PostgreSQL の場合

パッケージは次のとおりです。

- `edb-pg<version>-aidb` — AI データベース拡張機能。
- `edb-pg<version>-pgfs` — Postgres ファイルシステム拡張機能。

PGvector 拡張機能の場合、パッケージは次のとおりです。

- rpm ベースのシステム `edb-pg<version>-pgvector0`
- deb ベースのシステム `edb-pg<version>-pgvector-0`

EDB Postgres Advanced Server の場合

パッケージは次のとおりです。

- `edb-as<version>-aidb` — AI データベース拡張機能。
- `edb-as<version>-pgfs` — Postgres ファイルシステム拡張機能。

PGvector 拡張機能の場合、パッケージはパッケージフォーマットによって異なります。

- rpm ベースのシステムの場合 `edb-as<version>-pgvector0`
- deb ベースのシステムの場合 `edb-as<version>-pgvector-0`

EDB Postgres 拡張の場合

パッケージは次のとおりです。

- `edb-pgextended-<version>-aidb` — AI データベース拡張機能。
- `edb-pgextended-<version>-pgfs` — Postgres ファイルシステム拡張機能。

PGvector 拡張機能の場合、パッケージはパッケージフォーマットによって異なります。

- rpm ベースのシステムの場合 `edb-postgresextended<version>-pgvector0`

- deb ベースのシステムの場合 edb-postgresextended<version>-pgvector-0

例

EDB Postgres Advanced Server 18 を使用して Ubuntu 22.04 にパッケージをインストールするコマンドは次のとおりです。

```
sudo apt-get install edb-as17-aidb edb-as17-pgfs edb-as17-pgvector-0
```

パッケージをインストールすると、データベースと *Completing and verifying the extension installation* に接続できます。

Completing and verifying the extension installation

AI データベース拡張機能のインストール

AI Database 拡張機能は、データベース内の AI モデルを実行するための機能セットを提供します。CREATE EXTENSION コマンドを使用して拡張機能をインストールします。

```
edb=# CREATE EXTENSION aidb CASCADE;
__OUTPUT__
NOTICE: installing required extension "vector"
CREATE EXTENSION
```

EDB Postgres DistributedPGD の追加手順

AIDB 拡張カタログテーブルの PGD クラスター内でレプリケーションを設定するには、手動の手順が必要です。

```
bdrdb=# SELECT aidb.bdr_setup();
__OUTPUT__
bdr_setup
-----
(1 row)
```

次の SQL コマンドを使用して、AIDB テーブルが目的のレプリケーションセットの一部であることを確認できます。

```
bdrdb=# select * from bdr.tables where nspname=aidb;
__OUTPUT__
reloid | nspname | relname | set_name | set_ops | rel_columns | row_filter | conflict_detection |
-----+-----+-----+-----+-----+-----+-----+-----
18281 | aidb | pipeline_runtime_state | p-2y7357q195-a | {INSERT, UPDATE, DELETE, TRUNCATE} | | | row_origin
18354 | aidb | preparer_registry | p-2y7357q195-a | {INSERT,
```

(次のページに続く)

(前のページからの続き)

```

↪UPDATE,DELETE,TRUNCATE} | | | row_origin
18363 | aidb | preparer_registry_source_table | p-2y7357q195-a | {INSERT,
↪UPDATE,DELETE,TRUNCATE} | | | row_origin
18375 | aidb | preparer_registry_source_volume | p-2y7357q195-a | {INSERT,
↪UPDATE,DELETE,TRUNCATE} | | | row_origin
18430 | aidb | knowledge_base_registry | p-2y7357q195-a | {INSERT,
↪UPDATE,DELETE,TRUNCATE} | | | row_origin
18440 | aidb | knowledge_base_registry_source_table | p-2y7357q195-a | {INSERT,
↪UPDATE,DELETE,TRUNCATE} | | | row_origin
18452 | aidb | knowledge_base_registry_source_volume | p-2y7357q195-a | {INSERT,
↪UPDATE,DELETE,TRUNCATE} | | | row_origin
(7 rows)

```

バックグラウンドワーカーの設定

AIDB パイプラインでバックグラウンド処理を使用するには、拡張機能を Postgres 構成ファイルに「プリロードライブラリ」として登録する必要があります。

次の設定を `postgresql.conf` ファイルに追加します。構成ファイルの場所は、環境によって異なります。
`SHOW config_file;` で場所を特定できます。

```
shared_preload_libraries = aidb
```

プロキシ設定

HTTP プロキシを使用している場合、Postgres 環境で `HTTP_PROXY` および `HTTPS_PROXY` 環境変数を設定する必要がある場合があります。これは、次の例に示されている行を、Postgres データディレクトリの `main` ディレクトリにある `environment` ファイルに追加することにより、行うことができます。

たとえば、コミュニティ Postgres を使用する Ubuntu では、`environment` ファイルは `/etc/postgresql/16/main/environment` にあります。次のコマンドを実行してプロキシ設定を `environment` ファイルに追加し、Postgres サービスを再起動します。

```

echo "HTTP_PROXY = http://<yourproxysettings>/" | sudo tee -a /etc/postgresql/16/main/
↪environment
echo "HTTPS_PROXY = http://<yourproxysettings>/" | sudo tee -a /etc/postgresql/16/
↪main/environment
sudo systemctl restart postgresql@16-main

```

`<yourproxysettings>` をプロキシ設定に置き換えます。Postgres の別のバージョンを使用している場合は、16 をバージョン番号に置き換えます。`environment` ファイルの場所については、Postgres ディストリビューションのドキュメントを参照してください。

エアギャップ環境は現在サポートされていません。

Postgres ファイルシステム拡張機能のインストール

Postgres File SystemPGFS 拡張機能は、データベースからファイルシステムと対話するための一連の機能を提供します。CREATE EXTENSION コマンドを使用して拡張機能をインストールします。

```
edb=# create extension pgfs;  
CREATE EXTENSION
```

インストールの検証

psql で\dx コマンドを実行することにより、拡張機能がインストールされたかどうかを確認できます。

```
edb=# \dx  
__OUTPUT__  
  
List of installed extensions  


| Name   | Version | Schema | Description                                                |
|--------|---------|--------|------------------------------------------------------------|
| aidb   | 2.1.1   | aidb   | aidb: makes it easy to build AI applications with postgres |
| pgfs   | 1.0.6   | pgfs   | pgfs: enables access to filesystem-like storage locations  |
| vector | 0.8.0   | public | vector data type and ivfflat and hnsw access methods       |


```

出力にリストされる aidb、pgfs、および vector 拡張機能に加えて、通常は他の拡張機能も同様にリストされます。

Upgrading AI Accelerator Pipelines

バージョン 4.3.0 以降、AI Accelerator は拡張機能のインプレース更新をサポートしています。新しいバージョンがインストールされたら、ALTER EXTENSION コマンドを実行して Postgres データベースの拡張機能を更新できます。

```
ALTER EXTENSION aidb UPDATE;
```

- 4. 3.0 より前のバージョンからアップグレードする場合、クリーンなアップグレードを保証には、以下の手順を実行する必要があります。これは、拡張機能がこのバージョンより前にインプレース更新用に設計されていないためです。

パイプライン拡張機能 aidb のアップグレード

!!!注これは、管理対象データを削除するものではありません。AIDB のアンインストール中にダングリング参照たとえばトリガーが残らないようにするだけで、アップグレード後にクリーンセットアップを行うことができます。

1. 設定済みのパイプラインをすべて削除します

構成したすべてのナレッジベースとプリペアラーパーパイプライン `SELECT name FROM aidb.knowledge_bases;` および `SELECT name FROM aidb.preparers;` のリストを取得します。次に、`SELECT aidb.delete_knowledge_base('name');` および `SELECT aidb.delete_preparer('knowledge_base__5887');` を呼び出して、すべてのナレッジベースとプリペアラーパーを削除します。

次の SQL コードを使用して、1 つの操作で両方を行うことができます。

```
WITH names AS (SELECT name FROM aidb.knowledge_bases)
  SELECT aidb.delete_knowledge_base(name) FROM names;

WITH names AS (SELECT name FROM aidb.preparers)
  SELECT aidb.delete_preparer(name) FROM names;
```

2. 拡張機能の古いバージョンをアンインストールします

```
edb=# DROP EXTENSION aidb CASCADE;
DROP EXTENSION
edb=#
```

3. 環境に拡張機能の新しいバージョンをインストールします

古いバージョン *AI Accelerator Pipelines* のインストールに実行したのと同じ手順に従います。

4. Postgres データベースに拡張機能を作成します

```
edb=# CREATE EXTENSION aidb CASCADE;
NOTICE: installing required extension "vector"
CREATE EXTENSION
edb=#
```

5. 使用するパイプラインを再作成します

パイプラインを作成するために最初に実行したのと同じコマンドを実行します。

PGFS のアップグレード

PGFS を使用している場合、アップグレード後に拡張機能を削除して再作成する必要があります。

!!!注これにより、PGFS ストレージ場所への参照が削除されます。実際のデータは削除されません。アップグレード後にストレージの場所を再作成する必要があります。

```
DROP EXTENSION pgfs CASCADE;
CREATE EXTENSION pgfs CASCADE;
```

AI Accelerator Permissions Setup

新しいデータベースロール `aidb_users` は、AIDB 拡張機能のインストールの一部として作成されます。ほとんどの拡張機能にアクセスするには、このロールのメンバーシップが必要です。拡張機能が削除されても、このロールは削除されません。

`aidb_users` は、AIDB 機能のアクセス管理を簡素化するように設計されています。

- **目的** ユーザーに `aidb_users` ロールを付与すると、すべての AIDB 機能へのアクセスが提供され、拡張機能の機能へのアクセスが許可されます。
- **セキュリティ** `aidb_users` ロールは、AIDB の内部テーブルへのアクセスを付与せず、拡張機能のパブリック API のみがユーザーに公開されます。
- **バックグラウンドワーカー** バックグラウンドワーカーが使用される場合、バックグラウンドワーカーは、定義したユーザーとして、またはユーザーが指定したロールとして実行されます。ユーザーがそのロールにアクセスできる場合。

ユーザーが AIDB 機能を使用できるようにするには、`aidb_users` ロールを付与するだけです。

```
GRANT aidb_users TO your_user;
```

次に、最初に特権またはロールメンバーシップのない `alice` という名前のユーザーを作成する例を示します。

```
CREATE ROLE alice
LOGIN
PASSWORD change_me;  -- replace with a real password
```

次に、`alice` をアプリケーショングループロール `aidb_users` に追加します

```
GRANT aidb_users TO alice;
```

`alice` は、`aidb_users` が持っているグローバル特権を継承するようになりましたたとえば `aidb` のルーチンの `EXECUTE`、スキーマなどの `USAGE`

パブリックスキーマで選択したテーブルに対する完全な権限を `alice aidb_users` ではありませんこの例は、3つの任意のテーブルを示しています。これらは通常、パイプラインで使用されるソーステーブルです

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON public.documents,
   public.email_text,
   public.chats
TO alice;
```

`public` でオブジェクトを作成できるように `alice` が必要な場合

```
GRANT CREATE ON SCHEMA public TO alice;
```

アリスは、自分が作成したものを自動的に所有するため、自分のオブジェクトを `ALTER` または `DROP` できます。`public` で他のユーザーのオブジェクトをドロップまたは変更することはできません。

特権を検証できるようになりました ``sql :raw-latex:`z`public.documents, :raw-latex:`z`public.email_text, :raw-latex:`z`public.chats

Core concepts

AI Accelerator Pipelines knowledge bases

ナレッジベースは、セマンティック検索または取得拡張生成 RAG に使用できます。

構成されたナレッジベースは、入力データのエンベディングベクトルを計算できます。これが完了すると、ナレッジベースは、クエリに基づいてデータを返すための多数のインターフェイスを提供します。

Knowledge bases concepts

ナレッジベースは、セマンティック検索または取得拡張生成 RAG に使用できます。構成されたナレッジベースは、入力データのエンベディングベクトルを計算できます。コンピューティングが完了すると、ナレッジベースは、クエリに基づいてデータを返すインターフェイスを提供します。

データソースとデータ型

データソースは、埋め込み/取得されるデータの由来です。aidb 拡張機能は、2 種類のデータソースをサポートしています。

- テーブル — PG データベースのテーブルの列。
- ボリューム — AIDB ボリューム、PGFS 拡張機能を介して S3 オブジェクトストアまたはローカルファイルシステムにアクセスするためのラッパーです。

データ型は次のとおりです。

- テキスト
- 画像

任意のデータソースと任意のデータ型を組み合わせることができます。

- テーブルとテキスト この場合、テキストは、Postgres テーブルの TEXT / VARCHAR タイプの単なる列です。例は、製品リストの説明です。
- テーブルと画像 この場合、画像はテーブルの BYTEA 列に保存され、画像のバイトを保持します。Pipelines は、現在ラージオブジェクト BLOB としても知られるタイプをサポートしていません。
- ボリュームとテキスト 例は、S3 バケット内のテキストファイルです。
- ボリュームとイメージ画像はオブジェクトとして S3 バケットに保存できます。

埋め込み

ナレッジベースには、ソースデータに属するベクター埋め込みを保存するためのベクターテーブルが常に必要です。Pipelines は、エンコーダー機能を備えたモデルを使用して、ソースデータのベクターを生成します。

各モデルは異なる **次元**、つまりベクトルの次元数があります。ベクトル [1, 2, 2] は3つの次元があります。パイプラインは、PGvector *vector* 列タイプを使用してエンベディングを保存します。これらには固定数の次元があります。このため、モデルのエンコーダーインターフェイスは、ベクトルの次元数を結果ます。

Pipelines はベクターテーブルを作成し、既存のテーブルの使用もサポートします。ベクトル列の次元はモデルに適合する必要があります。

ソースデータとの一貫性

正しく一貫したデータを取得するには、エンベディングがソースデータと同期している必要があります。パイプラインは、これを実現するさまざまな *Pipeline auto-processing* モードをサポートしています。

- ライブ自動処理は、ソースデータに Postgres トリガーを設定して、一貫性を保証します
- バックグラウンド自動処理は、バックグラウンドで変更を非同期に処理します
- 自動処理を無効にすると、1つのファンクション呼び出しで必要に応じて処理を手動で実行できます。

Pipelines knowledge bases usage

新しいナレッジベース構成の作成

ナレッジベースには、テーブルとボリュームの2種類があります。データソースのさまざまな性質と、それぞれに必要なオプションを考慮すると、それらを作成するためのさまざまな機能があります。

テーブルデータソースのナレッジベース

`aidb.create_table_knowledge_base()` ファンクションは、テーブルデータソースのナレッジベースを作成するために使用されます。以下に最小限の例を示します。オプションの完全なリストについては、`aidb.create_table_knowledge_base` を参照してください。

例 ナレッジベースの作成

```
SELECT aidb.create_table_knowledge_base(  
    name => test_knowledge_base,  
    model_name => bert,  
    source_table => test_source_table,  
    source_data_column => content,  
    source_data_format => Text  
);
```

Postgres テーブルのみを使用している場合、Creating the embeddings に進んでください。

外部データソースを使用している場合、ボリュームを作成し、そのナレッジベースを作成する必要があります。これについては、次に説明します。

ボリュームデータソースのナレッジベース

新しいボリュームの作成

ボリュームのナレッジベースを作成する前に、ボリュームと PGFS ストレージ場所を作成する必要があります。詳細については、[AIDB volumes for accessing PGFS storage locations](#) を参照してください。

ここでは、パブリック S3 バケットの [Pipelines PGFS reference](#) を作成します。

```
SELECT pgfs.create_storage_location(aidb_text_bucket, s3://aidb-rag-app,
    options => {"region": "eu-central-1", "skip_signature": "true"}
);
```

aidb.create_volume_knowledge_base ファンクションは、ボリュームの作成に使用されます。

例 ボリュームの作成

```
SELECT aidb.create_volume(aidb_text, aidb_text_bucket, /, Text);
```

server_name は、PGFS ファンクションを呼び出して保存場所 [pgfs.create_storage_location](#) を作成することから取得されます。path は、保存場所のデータへのパスです。最後の引数はデータ形式を示します。Text または Image。

ボリュームのナレッジベースの作成

aidb.create_volume_knowledge_base() ファンクションは、ボリュームデータソースのナレッジベースを作成するために使用されます。以下に最小限の例を示します。オプションの完全なリストについては、aidb.create_volume_knowledge_base を参照してください。

例 ボリュームのナレッジベースの作成

```
SELECT aidb.create_volume_knowledge_base(
    name => test_knowledge_base_volume,
    model_name => bert,
    source_volume_name => aidb_text
);
```

埋め込みの作成

大量埋め込みを使用して、埋め込み計算を手動で実行できます。

```
SELECT aidb.bulk_embedding(test_knowledge_base);
```

ボリュームの例

```
SELECT aidb.bulk_embedding(test_knowledge_base_volume);
```

今後の変更に備えてライブ自動処理自動埋め込みを有効にします。

```
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Live);
```

自動処理を無効にすることもできます。新しいナレッジベースを作成するときは、デフォルトで「Disabled」です。

```
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Disabled);
```

取得

基本的な主要ナレッジベース `aidb.retrieve_key` が利用できます。ソースデータは検索しませんが、一致する埋め込みの ID / キーを結果ます。

キーの取得

```
aidb.retrieve_key(<knowledge base name>, <query string>, <optional number of results>
→);
```

例 キーの取得

```
SELECT * FROM aidb.retrieve_key(test_knowledge_base, shoes, 2);
__OUTPUT__
  key | distance
-----+-----
43941 | 0.2938963414490189
19337 | 0.3023805122617119
(2 rows)
```

キーに基づいて自分自身で結合/ルックアップを実行する場合、このコマンドを使用できます。外部ボリュームデータソースを使用するナレッジベースの場合、この機能は特に役立ちます。通常、アプリケーション自分自身は、外部データソースからの取得を行います。または、実際の取得をクライアントアプリケーションにプッシュダウンすることができます。

テキストの取得

`retrieve_text` ファンクションは、エンベディングをソースデータと結合し、結果を直接返します。

```
aidb.retrieve_text(<knowledge base name>, <query string>, <optional number of results>
→);
```

例

```
SELECT * FROM aidb.retrieve_text(test_knowledge_base, jacket, 2);
__OUTPUT__
  key | value | distance
```

(次のページに続く)

(前のページからの続き)

```
19337 | United Colors of Benetton Men Stripes Black Jacket | 0.2994317672742334
55018 | Lakme 3 in 1 Orchid Aqua Shine Lip Color          | 0.3804609668507203
(2 rows)
```

ナレッジベースのリスト

すべての Knowledge_base をリストするビューが利用できます。aidb.knowledge_bases には、ナレッジベース構成の一部も含まれています。

```
SELECT * FROM aidb.knowledge_bases;
```

__OUTPUT__

id	name	vector_table_name	vector_table_key_	
column	vector_table_vector_column	model_name	topk	distance_operator
options	source_table_name	source_table_data_column	source_table_data_column_	
type	source_table_key_column	source_volume_name		
2	test_knowledge_base	test_knowledge_base_vector	id	
	embeddings	simple_model	5	InnerProduct
	test_source_table	content	Text	id
5	test_knowledge_base_cosa	test_knowledge_base_cosa_vector	id	
	embeddings	simple_model	1	L2
	test_source_table	content	Text	id
3	test_knowledge_base_cos	test_knowledge_base_cos_vector	id	
	embeddings	simple_model	5	Cosine
	test_source_table	content	Text	id

(3 rows)

関心のある列のみを選択することをお勧めします。

```
SELECT name, source_table_name FROM aidb.knowledge_bases;
__OUTPUT__
```

name	source_table_name
test_knowledge_base	test_source_table
test_knowledge_base_cos	test_source_table
test_knowledge_base_cosa	test_source_table

(3 rows)

ナレッジベースの削除

このコマンドは、ベクタテーブルなどは削除しません。構成のみを削除します。

```
SELECT aidb.delete_knowledge_base(<name>);
```

エンドツーエンドの例

この *Knowledge bases end-to-end example usage* は、テーブル/テキストナレッジベースのエンドツーエンドの例です。

Knowledge bases end-to-end example usage

これは、EDB Postgres AI - AI Accelerator でナレッジベースを使用する完全なエンドツーエンドの例です。

```
- - Create extension
DROP EXTENSION aidb CASCADE;
CREATE EXTENSION aidb CASCADE;

drop table if exists test_source_table_ajz72eb cascade;
drop table if exists test_knowledge_base_ajz72eb_vector cascade;

- - Create source test table
CREATE TABLE test_source_table_ajz72eb
(
    id            INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content       TEXT NOT NULL,
    unrelated_column TEXT
);
INSERT INTO test_source_table_ajz72eb
VALUES (43941, Catwalk Women Brown Heels),
       (55018, Lakme 3 in 1 Orchid Aqua Shine Lip Color),
       (19337, United Colors of Benetton Men Stripes Black Jacket);

SELECT aidb.create_table_knowledge_base(
    name => test_knowledge_base_ajz72eb,
    model_name => bert, -- this is a pre-defined locally running model
    source_table => test_source_table_ajz72eb,
    source_data_column => content,
    source_data_format => Text
);

SELECT * FROM aidb.kbs; -- This is an alias for the view aidb.knowledge_bases. It
↳ shows the configuration for all knowledge bases

SELECT aidb.bulk_embedding(test_knowledge_base_ajz72eb); -- Since no auto-processing
↳ was configured, we manually run it
```

(次のページに続く)

(前のページからの続き)

```

SELECT * FROM aidb.kbstat; -- This is an alias for aidb.knowledge_base_stats. It
↳ shows statistics about the number of records and vectors

-- Perform retrieval similarity search for the closest `key`
SELECT * FROM aidb.retrieve_key(test_knowledge_base_ajz72eb, orchid);
SELECT * FROM aidb.retrieve_key(test_knowledge_base_ajz72eb, orchid, 2); -- Limit to
↳ top 2 results

SELECT * FROM aidb.retrieve_text(test_knowledge_base_ajz72eb, orchid);
SELECT * FROM aidb.retrieve_text(test_knowledge_base_ajz72eb, orchid, 2); -- Limit
↳ to top 2 results

-- enable live auto-processing
SELECT aidb.set_auto_knowledge_base(test_knowledge_base_ajz72eb, Live);

-- add additional data to test auto-processing
INSERT INTO test_source_table_ajz72eb
VALUES (11211, Bicycle),
      (11311, What is this?),
      (11411, Elephants);

-- check embeddings
SELECT id FROM test_knowledge_base_ajz72eb_vector;

-- delete one of the source rows
DELETE FROM test_source_table_ajz72eb WHERE id = 11211;

-- check embeddings
SELECT id FROM test_knowledge_base_ajz72eb_vector;

```

Pipelines knowledge base performance tuning

背景

パイプラインとモデル設定を変更することにより、パフォーマンスつまり 1 秒あたりの埋め込みのスループットを最適化できます。

ナレッジベースパイプラインは、個々のレコードテーブルの行またはボリューム内のオブジェクトのコレクションを処理します。AIDB は、各レコードを個別にシークエンシャルに処理したり、すべてを同時に処理したりするのではなく、バッチ処理を提供します。すべてのバッチは、次々にシークエンシャルに処理されます。各バッチ内で、レコードは可能な限り同時に処理されます。

- *Pipeline auto-processing* は、各バッチのレコード数を決定します。
- 一部のモデルプロバイダーには、構成可能な内部バッチ/並列処理があります。これらの設定をデフォルト値のままにし、パイプラインのバッチサイズを使用して実行を制御することをお勧めします。

i 注釈

Vector indexing also has an impact on pipeline performance. You can disable the vector by using `index_type => 'disabled'` to exclude it from your measurements.

パフォーマンスのテストとチューニング

この例では、最初にテストデータとナレッジベースパイプラインを設定し、次にバッチサイズを測定および調整します。

1) テーブルを作成し、テストデータを挿入します

実際のデータコンテンツの長さは、モデルのパフォーマンスに一定程度の影響を与えます。長いテキストを使用してテストできます。

```
CREATE TABLE test_data_10k (id INT PRIMARY KEY, msg TEXT NOT NULL);

INSERT INTO test_data_10k (id, msg) SELECT generate_series(1, 10000) AS id, hello_
↪world;
```

2) ナレッジベースパイプラインを作成する

最適なバッチサイズは、モデルによって大きく異なる場合があります。使用するさまざまなモデルごとにバッチサイズを測定および調整します。

```
SELECT aidb.create_table_knowledge_base(
  name => perf_test,
  model_name => dummy, -- use the model you want to optimize for
  source_table => test_data_10k,
  source_data_column => msg,
  source_data_format => Text,
  index_type => disabled, -- optionally disable vector indexing to include/exclude_
↪it from the measurement
  auto_processing => Disabled, -- we want to manually run the pipeline to measure_
↪the runtime
  batch_size => 100 -- this is the paramter we will tune during this test
);
__OUTPUT__
INFO: using vector table: public.perf_test_vector
NOTICE: index "vdx_perf_test_vector" does not exist, skipping
NOTICE: auto-processing is set to "Disabled". Manually run "SELECT aidb.bulk_
↪embedding(perf_test);" to compute embeddings.
create_table_knowledge_base
-----
perf_test
(1 row)
```

3) パイプラインを実行し、パフォーマンスを測定する

このテストでは psql を使用します。 \timing on コマンドは psql の機能です。別のインターフェイスを使用する場合、タイミング情報を表示する方法を確認してください。

```
\timing on
__OUTPUT__
Timing is on.
```

次に、パイプラインを実行します。

```
SELECT aidb.bulk_embedding(perf_test);
__OUTPUT__
INFO: perf_test: (re)setting state table to process all data...
INFO: perf_test: Starting... Batch size 100, unprocessed rows: 10000, count(source_
↪records): 10000, count(embeddings): 0
INFO: perf_test: Batch iteration finished, unprocessed rows: 9900, count(source_
↪records): 10000, count(embeddings): 100
INFO: perf_test: Batch iteration finished, unprocessed rows: 9800, count(source_
↪records): 10000, count(embeddings): 200
...
INFO: perf_test: Batch iteration finished, unprocessed rows: 0, count(source_
↪records): 10000, count(embeddings): 10000
INFO: perf_test: finished, unprocessed rows: 0, count(source records): 10000,
↪count(embeddings): 10000
bulk_embedding
-----

(1 row)

Time: 207161,174 ms (03:27,161)
```

4) バッチサイズを調整する

この呼び出しを使用して、パイプラインのバッチサイズを調整できます。この例では、バッチサイズを 10 倍の 1000 レコードに増加します。

```
SELECT aidb.set_auto_knowledge_base(perf_test, Disabled, batch_size=>1000);
```

パイプラインを再度実行します。

注釈

When using a Postgres table as the source, with auto-processing disabled, AIDB has no means to detect changes in the source data. So each bulk_embedding call has to reprocess everything.

これは、パフォーマンステストに便利です。

ボリュームソースを使用してパフォーマンスを測定する場合は、各テストの間にナレッジベースを削除して再作成します。AIDB は、自動処理が無効でもボリュームの変更を検出できます。

結論

このテストでは、パイプラインはバッチサイズ 1000 で 02:34 分、サイズ 100 で 03:27 分かかりました。パフォーマンスが向上しなくなるか、低下しなくなるまで、より大きなサイズのテストを続行できます。

Pipelines knowledge bases usage auto processing

自動処理モードの選択

背景の詳細については、[Pipeline auto-processing](#) も参照してください。

Live、**Background**、および **Disabled** の自動処理から選択できます。このモードは、`aidb.create_table_knowledge_base` および `aidb.create_volume_knowledge_base` 呼び出しでナレッジベースを作成するときに構成できます。

`aidb.set_auto_knowledge_base` 呼び出しの処理設定バッチサイズや同期間隔などとともに、既存のナレッジベースでモードを変更することもできます。

入力データの準備

次の例はすべて、この共通のソーステーブルを使用します。

```
CREATE TABLE test_source_table
(
  id          INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  content     TEXT NOT NULL
);
INSERT INTO test_source_table
VALUES (1, Catwalk Women Brown Heels),
       (2, Lakme 3 in 1 Orchid Aqua Shine Lip Color),
       (3, United Colors of Benetton Men Stripes Black Jacket);
```

ライブ自動処理でのナレッジベースの使用

この処理モードは Postgres トリガーに依存するため、テーブルソースでのみサポートされています。このモードでは、各レコードが作成/変更されるとすぐに個別に処理されるため、バッチ処理も利用しません。

ナレッジベースを作成し、ライブ自動処理を構成します。

```
SELECT aidb.create_table_knowledge_base(
  name => test_kb_live,
  model_name => bert, -- this is a pre-defined locally running model
  source_table => test_source_table,
  source_data_column => content,
  source_data_format => Text,
  auto_processing => Live
```

(次のページに続く)

(前のページからの続き)

```
);
__OUTPUT__
INFO:  using vector table: public.test_kb_live_vector
NOTICE:  auto-processing is set to "Live". AIDB will process all new/updates rows,
↳going forward. To process existing data, manually run "SELECT aidb.bulk_
↳embedding(test_kb_live);"
create_table_knowledge_base
-----
test_kb_live
(1 row)
```

または、既存のナレッジベースをライブ自動処理に手動で設定します。

```
SELECT aidb.set_auto_knowledge_base(test_kb_live, Live);
```

ライブ自動処理は、今後のソースデータの変更に対して有効になります。しかし、テーブル内の既存のデータはまだ処理されていません。

新しいレコードを追加して、これをテストします。

```
INSERT INTO test_source_table VALUES (4, This is also a Jacket);
__OUTPUT__
INFO:  Running live embedding for knowledge_base test_kb_live. key: "4" content:
↳"This is also a Jacket"
INSERT 0 1
```

クエリーは自動処理された新しいレコードを結果ますが、既存のレコードは結果ません。

```
SELECT * FROM aidb.retrieve_text(test_kb_live, Jackets, topk=>4);
__OUTPUT__
key | value | distance
-----+-----+-----
4 | This is also a Jacket | 0.9696326387986498
(1 row)
```

次に、上記の NOTICE: に示すように bulk_embedding 呼び出しを実行します。

```
SELECT aidb.bulk_embedding(test_kb_live);
__OUTPUT__
INFO:  test_kb_live: (re)setting state table to process all data...
INFO:  test_kb_live: Starting... Batch size 100, unprocessed rows: 4, count(source_
↳records): 4, count(embeddings): 1
INFO:  test_kb_live: Batch iteration finished, unprocessed rows: 0, count(source_
↳records): 4, count(embeddings): 4
INFO:  test_kb_live: finished, unprocessed rows: 0, count(source records): 4,
↳count(embeddings): 4
```

(次のページに続く)

(前のページからの続き)

bulk_embedding

(1 row)

別の取得では、存在するすべての行が表示されます。

```
SELECT * FROM aidb.retrieve_text(test_kb_live, Jackets, topk=>4);
```

__OUTPUT__

key	value	distance
4	This is also a Jacket	1.0353689874817347
1	Catwalk Women Brown Heels	1.0404924577497399
3	United Colors of Benetton Men Stripes Black Jacket	1.324741586641547
2	Lakme 3 in 1 Orchid Aqua Shine Lip Color	1.4018234945521992

(4 rows)

統計ビューを照会して、存在するレコードの数と、まだ処理されたレコードの数を確認できます。

```
SELECT * FROM aidb.kbstat;
```

__OUTPUT__

knowledge base	auto processing	table: unprocessed rows	volume: scans completed
count(source records)	count(embeddings)		

test_kb_live	Live	0	
	4	4	

(1 row)

バックグラウンドの自動処理でナレッジベースを使用する

バックグラウンドの自動処理は、すべてのソースタイプとデータ型でサポートされています。

注釈

Background workers for AIDB must be configured in the Postgres config file; otherwise they will not run. Refer to our installation instructions for more details.

ナレッジベースを作成し、バックグラウンドの自動処理を構成します。バッチサイズと同期間隔の設定はオプション。

- 同期間隔が短いと、埋め込みが再び同期するまでの遅延が短くなります。しかし、短い遅延は、特にストレージシステムへの潜在的にコストの高い要求を必要とするボリュームソースの場合、より多くの処理オーバーヘッドを発生させます。
- バッチサイズが大きいと、特定の点まではスループットが高くなる可能性があります。これは、インフラストラクチャ、データソース、モデルの種類、さらにはデータ自身自身に大きく依存します。さ

まざまなバッチサイズをテストすることをお勧めします。

```
SELECT aidb.create_table_knowledge_base(
  name => test_kb_background,
  model_name => bert, -- this is a pre-defined locally running model
  source_table => test_source_table,
  source_data_column => content,
  source_data_format => Text,
  auto_processing => Background,
  batch_size => 50,
  background_sync_interval => 10 seconds
);
```

__OUTPUT__

```
INFO: using vector table: public.test_kb_background_vector
INFO: test_kb_background: (Re)setting state table to process all data...
NOTICE: auto-processing is set to "Background". Check progress with "SELECT * FROM_
↪aidb.kbstat;"
create_table_knowledge_base
-----
test_kb_background
(1 row)
```

または、既存のナレッジベースをバックグラウンド自動処理に手動で設定します。

```
SELECT aidb.set_auto_knowledge_base(test_kb_background, Background);
```

注釈

When enabling background auto-processing on a pre-existing table knowledge base, only new change events will be processed. If the source table contains unprocessed source data, you can use the `aidb.bulk_embedding()` function to trigger re-processing of all records.

バッチサイズおよび/または同期間隔を変更するには

```
SELECT aidb.set_auto_knowledge_base(test_kb_background, Background, batch_size=>10);
SELECT aidb.set_auto_knowledge_base(test_kb_background, Background, background_sync_
↪interval=>10 minutes);
SELECT aidb.set_auto_knowledge_base(test_kb_background, Background, batch_size=>10, ↪
↪background_sync_interval=>10 minutes);
```

次のような統計を照会できるようになりました。

```
SELECT * FROM aidb.kbstat;
```

__OUTPUT__

```
knowledge base | auto processing | table: unprocessed rows | volume: scans_
↪completed | count(source records) | count(embeddings)
```

(次のページに続く)

(前のページからの続き)

test_kb_background	Background		0
	4	4	
(1 row)			

新しいデータがソーステーブルに挿入されると、AIDB テーブルトリガーが変更イベントを作成します。これは、上記の表に `table: unprocessed rows` として示されています。次の `background_sync_interval` の有効期限が切れると、バックグラウンドワーカーはこれらの変更された行をすべて処理します。

新しいレコードを挿入することにより、これを表示できます。

```
INSERT INTO test_source_table VALUES (5, This is also another Jacket);
```

そして、統計を再度確認します。

```
SELECT * FROM aidb.kbstat;
```

__OUTPUT__

knowledge base	auto processing	table: unprocessed rows	volume: scans
completed	count(source records)	count(embeddings)	

test_kb_background	Background		1
	5	4	
(1 row)			

自動処理を無効にしたナレッジベースの使用

このモードでは、自動処理は発生しません。ユーザーは `aidb.bulk_embedding()` を手動で呼び出すか、`aidb.set_auto_knowledge_base()` 呼び出しで 2 つの自動処理モードのいずれかを有効にできます。

AI Accelerator Pipelines - Preparers

プリペアラーは、テーブルまたはボリュームソースのソースデータに一般的な前処理手順を実行するために使用されます。処理されたデータは宛先テーブルに保存され、ナレッジベースで埋め込み生成に使用できます。

Preparers concepts

プリペアラーは、テーブルまたはボリュームソースのソースデータに一般的な前処理手順を実行するために使用されます。処理されたデータは宛先テーブルに保存され、他の準備者またはナレッジベースで埋め込み生成に使用できます。

コンセプト

データ準備オペレーション

データ準備操作は、プリペアラがソースデータを変換する方法です。サポートされている操作は、`aidb.DataPreparationOperation` 列挙型のバリエーションとしてエンコードされます。

i 注釈

Each operation has its own set of parameters that are used to customize the operation. Learn more in [Pipelines model primitives](#).

データソース

データソースは、データ準備操作の入力データです。aidb 拡張機能は、プリペアラの 2 種類のデータソースをサポートしています。

- テーブル PG データベースのテーブルの列。
- ボリューム PGFS 「ボリューム」。S3 オブジェクトストアまたはローカルファイルシステムにアクセスするためのラッパーです。

実行

プリミティブ関数は、最小限のセットアップで個々の入力での操作とその構成のテストに役立ちます。これは、大量データの準備のための準備者でスケールアップする前の簡単な実験に役立ちます。

大量データの準備は、準備者のすべてのソースデータに対して準備者に関連する操作を実行します。

i 注釈

Bulk data preparation doesn't delete existing destination data unless it conflicts with newly generated data. We recommend configuring separate destination tables for each preparer.

ネストを解除する

一部のプリペアラ [Pipelines model primitives](#) は、指定されたデータの形状を変換します。たとえば、`ChunkText` は 1 つのテキストブロックを受信し、1 つ以上のテキストブロックを生成します。これらのプリミティブは、結果のネストされたコレクションを返すのではなく、追加のディメンションを追跡するための新しい `part_id` 列を使用して、出力を解凍または「展開」します。

これは、[Pipelines model primitives](#) および該当する [Where to find SDK references and examples?](#) のアクションで見ることができます。

ソースデータとの一貫性

正しく一貫したデータを保証するには、準備された宛先データがソースデータと同期している必要があります。テーブルデータソースの場合、プリペアラの自動処理を有効にして、ソースデータの変更についてプリペアラパイプラインに通知できます。

i 注釈

If the source table already contains data when the preparer is created, then you must make an initial manual bulk data preparation call.

Preparer の自動処理

プリペアラの自動処理を有効にして Postgres でトリガーを作成し、ソースデータが追加、更新、または削除された場合に、準備された宛先データを最新の状態に保つことができます。

i 注釈

Preparer auto-processing currently works only with preparers with a table data source, not a volume source.

Pipelines data preparation primitives and options

ほとんどのユースケースでは、aidb *Pipelines knowledge bases usage* を使用して、データの大量前処理の準備手順を実行することをお勧めします。

ただし、データに最適な構成をテストおよび開発するために、次のプリミティブを使用できます。これらの機能を使用すると、最小限のセットアップで個々の入力で操作とその構成をテストできます。これは、大量データの準備のための準備者でスケールアップする前の簡単な実験に役立ちます。

構成

すべてのデータ準備操作は、さまざまなオプションを使用してカスタマイズできます。これらのオプションの API はプリミティブとプリペアラで同一であるため、ChunkText オペレーションを実行するスケールブルプリペアラで使用する aidb.chunk_text() プリミティブでオプションをプロトタイプにできます。

チャンクテキスト

aidb.chunk_text() を呼び出して、長いテキストをより小さい意味的に一貫したチャンクにインテリジェントに分割します。LLM コンテキスト制限内での処理または保存に最適です。

```
SELECT * FROM aidb.chunk_text(
  input => This is a significantly longer text example that might require splitting.
  into smaller chunks. The purpose of this function is to partition text data into
  segments of a specified maximum length, for example, this sentence 145 is
  characters. This enables processing or storage of data in manageable parts.,
  options => {"desired_length": 120, "max_length": 150}
);
```

__OUTPUT__

part_id	chunk
0	This is a significantly longer text example that might require splitting.

(次のページに続く)

(前のページからの続き)

```

→into smaller chunks.
    1 | The purpose of this function is to partition text data into segments of a
→specified maximum length, for example, this sentence 145 is characters.
    2 | This enables processing or storage of data in manageable parts.
(3 rows)

```

オプション

- `desired_length` 必須ターゲットチャンクサイズ、文字単位。これは、セマンティック構造を維持しながら、各チャンクを形成するときにスプリッターが到達しようとするサイズです。`max_length` が提供されない場合、`desired_length` はチャンクサイズのハード上限としても機能します。
- `max_length` オプションナルチャンクサイズの上限文字数。指定すると、ファンクションは `desired_length` に近いチャンクを生成しようとしませんが、より大きな意味単位完全な文や段落などを保持するために `max_length` まで拡張する場合があります。チャンクは、意味のある境界を横切ることを避ける必要がある場合にのみ `desired_length` を超えます。
- `desired_length = max_length` を指定すると、固定サイズのチャンクたとえば、埋め込みのためにコンテキストウィンドウを正確に埋める場合。
- 範囲内に収まりたい場合は、より大きな `max_length` を使用しますソフトリミットですが、RAG、要約、または Q&A アプリケーションで一般的な、より高い意味単位を保持するためある程度の柔軟性を許可します。

アルゴリズムの概要

- テキストは、意味的境界の階層を使用して分割されます文字、書記素、単語、文、およびますます長くなる改行シーケンス段落など。
- スプリッターは、指定されたサイズ範囲内に収まる意味的に有効な最大のチャンクを形成しようとします。
- 次のセマンティックユニットを追加すると `max_length` を超える場合、`desired_length` より短いチャンクが返される場合があります。

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the [unnesting concept](#) for more details.

HTML を解析する

`aidb.parse_html()` を呼び出して、HTML からテキストを抽出します。

```

SELECT * FROM aidb.parse_html(
  html =>
    <h1>Hello, world!</h1>
    <p>This is my first web page.</p>

```

(次のページに続く)

(前のページからの続き)

```

<p>
  It contains some <strong>bold text</strong>, some <em>italic test</em>,
  and a <a href="https://google.com" target="_blank">link</a>.
</p>



<ol>
  <li>List item</li>
  <li>List item</li>
  <li>List item</li>
</ol>,
options => {"method": "StructuredPlaintext"} -- Default
);

```

__OUTPUT__

parse_html

```

-----
Hello, world!
+
+
This is my first web page.
+
+
It contains some bold text, some italic test, and a link.
+
+
Postgres Logo Image
+
List item
+
List item
+
List item
+

```

(1 row)

- method は、HTML を解析する方法を決定します。
- StructuredPlaintext デフォルトプレーンテキストへのアルゴリズムテキスト抽出。
- StructuredMarkdown 一部の構文コンテキストを保持する Markdown のようなテキストへのアルゴリズムテキスト抽出。

PDF を解析する

aidb.parse_pdf() を呼び出して、PDF バイトからテキストを抽出します。

```

SELECT * FROM aidb.parse_pdf(
  bytes =>
  decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6672f506167657320332
  hex),

```

(次のページに続く)

(前のページからの続き)

```

options => {"method": "Structured", "allow_partial_parsing": true} -- Default
);

__OUTPUT__
part_id |      text
-----+-----
      0 | Hello World!+
      |
(1 row)

```

- `method` は、PDF を解析する方法を決定します。
- `Structured` (デフォルト) — アルゴリズムテキスト抽出。PDF 仕様に依存する構造 PDF 解析は、特に複雑なまたは独自のエンコーディングとさまざまなフォントが含まれる場合、仕様に準拠していない PDF に苦戦する可能性があります。
- `allow_partial_parsing` フラグは、パーサーが 1 つでエラーを検出したときに PDF の解析を続行するかどうかを決定します。それ以上のページ。デフォルト `true`
- 出力の `part_id` 列は、テキストが抽出されたページのインデックスを参照します。

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a `part_id` column. See the [unnesting concept](#) for more detail.

テキストの要約

`aidb.summarize_text()` を呼び出してテキストを要約します。

```

- - Create a model for use in summarization
SELECT aidb.create_model(my_t5_model, t5_local);

SELECT * FROM aidb.summarize_text(
    input => There are times when the night sky glows with bands of color. The bands
    ↳ may begin as cloud shapes and then spread into a great arc across the entire sky.
    ↳ They may fall in folds like a curtain drawn across the heavens. The lights usually
    ↳ grow brighter, then suddenly dim. During this time the sky glows with pale yellow,
    ↳ pink, green, violet, blue, and red. These lights are called the Aurora Borealis.
    ↳ Some people call them the Northern Lights. Scientists have been watching them for
    ↳ hundreds of years. They are not quite sure what causes them. In ancient times Long
    ↳ Beach City College WRSC Page 2 of 2 people were afraid of the Lights. They imagined
    ↳ that they saw fiery dragons in the sky. Some even concluded that the heavens were
    ↳ on fire.,
    options => {"model": "my_t5_model"}
);

```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
create_model

-----

my_t5_model
(1 row)

summarize_text

-----

the night sky glows with bands of color . they may begin as cloud shapes and then
spread into a great arc across the entire sky . the lights usually grow brighter,
then suddenly dim .
(1 row)
```

- `model` は、要約に使用する作成したモデルの名前です。モデルは、`decode_text()` および `decode_text_batch()` *Pipelines model primitives* をサポートする必要があります。

OCR を実行する

`aidb.perform_ocr()` を呼び出して、画像バイトからテキストを解析します。

[illegible]

(次のページに続く)

(次のページに続く)

335

(前のページからの続き)

```

↪ABgpgZaVE4RVIHg/
↪1EgCwPtea0+SXzCBjv8AeVaVVHvRvGCerMDsoXeZ6IOAmhqPtexhPnjcq4tc031NUR+mLwIKtGeutmsqg2nm+/
↪uIsIBLPckplqGzXMmc99Q8hpxs/
↪cUksbY5jSmSEXzVEwNxwezFa8u9Wyt0a8h6nmqYZ6plC5lR54Ji7A1SYaejjxkDxe7wDBxJKkngg+1zaWm06jiZ7yPVqVckEH3
↪rqtqWdm+9QPwGYBndhL94Ydr3wSQvZDPNH9sss0g8katWYTFuLdOryHqWaeB84pH94/
↪raaRxDuLUwybkBsUXrJJibmtVdNHSxuIO7k7vTx4v1VbNX3eJwxTQ1RnDPbECmN/
↪SNnSMimkHhIJAS1ytipZQQRU0rqWN9zHKokorMc4iCPfd3i/
↪gVXUyNYQZ6SeTeMmid6scU6WhsdkhbEGy7iLN4uerVqRj+8zFNv4Jg0+5UeA1dNjey+/
↪C5t4IVsz1TI6mtghgl00Npc4sblVqVMTJhjE1UYC4cGhSTzwNI3lNUNGHx+8b1JipquASMP+L0dxC/
↪M2n1KaJJvJtQeBkkKrJ4qM0/Vl7rTt4MKralIKXAXul7xTycpKYI52nBVQfhyj6jm2cen1PpZtVdkK/
↪I1CfhfUGOnBBuID1YE9bXVTgGzxa3BEB5V2Rr/A0exUT7PgmtOCUQuIJbicA//
↪ACKpH1MDKQR1LGeZmFWJ3tsDBGJYzuprmqIxVsbCEIqgY/MfKHBZNHuoIu4sLvU/
↪0P2gf5hW0dzLLfUUtQwt++BdiY5WTVUsFVHupmkCVjghA6mrLt5JE8uxYfAG9UclDTxWnN06naQC7CTgAv8AQFdkK7+exdn560
↪sxktzQoWyiyOY9yx/
↪4s5P3tloystgGR4iMxneDN5jgVi1lW7wYtDEFSimnr00FBIYgCAwb5gATiGVtdLLIPfu7P/
↪siiHebUlmqJ3n2a8xsGUJrXteC1zXC8E09CpKwTtbIwMfJezDIF+bqV+S/
↪3020sktHNVx1kFREMe5mFxuczpMgAa6ekIGZpTGPtt+707nvhe8xqR7KCGQVDnMl3IaYwftJVXU1UMc75SIgZcTj7vK7PTMiQ
↪fouj15JY6RmYqNsUFPEYKJjfAMaLgmAPkpKkHMFc1jTZ0Eh9LzI0PJTbJbaXVhXtP1TA+FlqVtQ5h9dy8vQAA+wAeiiAqrOqY8
↪dkieXsoa26L+ML83aX+op5EFLRh4Z7ueux1YI6WBkZLZWA0cux9ZdUxEMe6RhDH+j1J/
↪RTUrJAzN1HNs49PqfSzaq7IV+RqFFjljiE7GjzExlWXST15eJqV84F7m+BYF2Ss4Aeu7X/
↪AA6pKyYVrIGVceHdiVgTSsDID3gPAbgmCsChqmTRh7JWxgEh3qCFVPjZVzPYaR5xFuEediop7Q3VUaIgpUOE0Ia4kqw6fFE8xT
↪ACFVvmbuZKyYF9zp34gMK70WdTxXC8vMTSAPc16LDRO7RWZFCWC5pEL2RphLKGuFHNwE7fE5F0zv1kySMfET950T317XBVLiaa
↪sbjndtsGSnAr5qE1M8jTGHMcWaldmaXfP/
↪tYRunDiCFakr2vibVRfiwEOICkkjFtUMVTW85ETDhVBFaNbWY33Sm9kVzixWdRUs9ZSvpaaNkTQ8umC/
↪Eouj17UOz8rVdWL+6qP/wBTV/eY/
↪wBDl+90jdTC1ayKd3tHM8sJUzJYZWB0b2G8ODvUFTtdXV08b3RjzMiJUZYa6te9vM2ML83aX+oqAiKamfC9/
↪GNdm7Nkl3EbKhpYC4StFxxhdlrOENMwuwlgbF7MHersPFY/
↪dY2MdUAg4yf1PqObZx6fUgGSeKmJYD4XuaQFSQRUndZo8TJg43yJoc1zS1zT4E09FUhoMpl7sX7p8TuQqrteU7xc8PnAap2VNr
↪Rgg3QH6wP4l6tEzRknBLTS7p/8TSqmsFK/
↪z94nDI10Ki16pobLMBcxjP3GKZlLbEEeASHySNHgx6qa1tMz7g7vUB0StAiEPBkY6XeTTAfbbhVOWSSUHD6WLyTGG64KmiJdVPh
↪zDVXmojB04mil3c4BVVUmmxjGKmoAYFM2ptKrINVVe4Hgxns0KozFX0VRv4d55X/
↪dIc0lVZhsmtGCSohZV3s3QlBftnigtGX1MDzgErX9gcFU2juQLmXTtcq4iLGHYxbzezy+zT7KR1LV0Lg+iP6jcIuwkexCNZD
↪PUTnzSv8AqObZx6ftQf8AX9nJhwie1P2BzboPT4JQwRyG++RsbQ84uLR+wObZx6fJ0bZx6fJ0bZx6fJ0bZx6fJ0bZx6fJ0bZx6
↪p8mGz//xAAUEQEAAAAAAAAAAAAAAAAAACg/9oACAECAQE/ABK//8QAFBEBAAAAAAAAAAAAAAAAAAAAoP/
↪aAAgBAwEBPwASv//Z, base64),
  options => {"model": "my_paddle_ocr_model"}
);

__OUTPUT__
create_model
-----
my_paddle_ocr_model
(1 row)

part_id |      text

```

(次のページに続く)

(前のページからの続き)

```
-----+-----
0 | Tesseract sample
(1 row)
```

- `model` は、OCR に使用する作成したモデルの名前です。モデルは `perform_ocr` オペレーションをサポートする必要があります。

Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the [unnesting concept](#) for more detail.

注釈

Limitations of the model still apply. For example, the [NVIDIA NIM Image OCR API](#) model provider only supports *png* and *jpeg* image inputs.

Pipelines preparers usage

新しいプリペアラの作成

ソースデータプリペアラは、Postgres テーブルまたは PGFS ボリュームから取得できます。データソースのさまざまな性質とそれぞれに必要なオプションを考慮して、さまざまなファンクションを使用してデータソースを作成します。

注釈

You can customize the behavior of the data preparation operation for the preparer with different options. The API for these options is identical between the primitives and the preparer, so you can prototype options with the `aidb.chunk_text()` primitive for use with a scalable preparer that performs the *ChunkText* operation. Learn more in [Pipelines model primitives](#).

テーブルデータソースの準備者

`aidb.create_table_preparer` ファンクションは、テーブルデータソースのプリペアラを作成するために使用されます。これはファンクションシングネチャです。

```
aidb.create_table_preparer(
-----
```

<code>name</code>	<code>TEXT,</code>
<code>operation</code>	<code>aidb.DataPreparationOperation,</code>
<code>source_table</code>	<code>TEXT,</code>
<code>source_data_column</code>	<code>TEXT,</code>
<code>destination_table</code>	<code>TEXT,</code>

(次のページに続く)

(前のページからの続き)

```

destination_data_column    TEXT,
source_key_column          TEXT DEFAULT id,
destination_key_column     TEXT DEFAULT id,
options                    JSONB DEFAULT {}::JSONB
)

```

 Tip

The *source_key_column* must be a unique key for the source data. If the data source is the output of a preparer that *transforms the data shape* with a *part_id* column, make sure to use the new *unique_id* column.

例プリペアラの作成

```

SELECT aidb.create_table_preparer(
  name => test_preparer,
  operation => ChunkText,
  source_table => test_source_table,
  source_data_column => content,
  destination_table => chunked_data_destination_table,
  destination_data_column => chunk,
  source_key_column => id,
  destination_key_column => id,
  options => {"desired_length": 100}::JSONB -- Configuration for the ChunkText_
  ↳ operation
);

```

ボリュームデータソースの準備

`aidb.create_volume_preparer` ファクションは、ボリュームデータソースのプリペアラを作成するために使用されます。これはファクションシングネチャです。

```

aidb.create_volume_preparer(
  -----
  name                TEXT,
  operation            aidb.DataPreparationOperation,
  source_volume_name   TEXT,
  destination_table    TEXT,
  destination_data_column TEXT,
  destination_key_column TEXT DEFAULT id,
  options              JSONB DEFAULT {}::JSONB
)

```

例 ボリュームのプリペアラの作成

```
SELECT aidb.create_volume_preparer(
  name => test_preparer_volume,
  operation => ChunkText,
  source_volume_name => test_volume,
  destination_table => chunked_data_destination_table,
  destination_data_column => chunk,
  destination_key_column => id,
  options => {"desired_length": 100}::JSONB -- Configuration for the ChunkText
  ↳ operation
);
```

データの準備の実行

大量データの準備を使用して、ソースの場所にあるすべての既存のデータの処理操作を実行できます。

```
SELECT aidb.bulk_data_preparation(test_preparer);
```

プリペアラの自動処理を有効にして、ソースデータへの今後の変更を自動的に処理しますテーブルデータソースのみ。

```
SELECT aidb.set_auto_preparer(test_preparer, Live);
```

自動プリペアラ処理を無効にすることもできます。

```
SELECT aidb.set_auto_preparer(test_preparer, Disabled);
```

プリペアラのリスト

すべての準備者をリストするビューが利用できます。aidb.preparers には、プリペアラ構成の一部も含まれています。

```
SELECT * FROM aidb.preparers;
__OUTPUT__
id | name | operation | destination_schema | destination_table |
↳| destination_key_column | destination_data_column | options |
↳source_type | source_schema | source_table | source_data_column | source_key_
↳column | source_volume_name
- - - - -
↳+-----+-----+-----+-----+-----+
↳+-----+-----+-----+-----+-----+
↳+-----+-----+-----+-----+-----+
↳+-----+-----+-----+-----+-----+
1 | test_preparer | ChunkText | public | chunked_data_destination_table |
↳| id | chunk | {"desired_length": 100} | Table
| public | test_source_table | content | id |
(1 row)
```


プリペアラの削除

```
SELECT aidb.delete_preparer(<name>);
```

i 注釈

This command doesn't delete the destination table or any data in it.

エンドツーエンドの例

Where to find SDK references and examples? には、プリペアラのエンドツーエンドの例があります。

AI Accelerator Pipelines - Preparers examples

これらの例は、Data preparation operation によってグループ化されています。

Preparer chaining example

AI Accelerator での *ChunkText* および *SummarizeText* オペレーションを使用した自動処理と複数のプリペアラをチェーンする例。

チャンクテキストの最初のプリペアラを作成する

```
-- Create source test table
CREATE TABLE source_table__1321
(
  id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  content TEXT NOT NULL
);

SELECT aidb.create_table_preparer(
  name => chunking_preparer__1321,
  operation => ChunkText,
  source_table => source_table__1321,
  source_key_column => id,
  source_data_column => content,
  destination_table => chunked_data__1321,
  destination_data_column => chunk,
  destination_key_column => id,
  options => {"desired_length": 1000}:JSONB -- Configuration for the ChunkText_
  ↳operation
);
```

チャンク化されたテキストを要約する 2 番目のプリペアラを作成します

```
-- Create the model. It must support the decode_text and decode_text_batch_
--operations.
SELECT aidb.create_model(model__1321, t5_local);

SELECT aidb.create_table_preparer(
  name => summarizing_preparer__1321,
  operation => SummarizeText,
  source_table => chunked_data__1321,           -- Reference the output from the_
--ChunkText preparer
  source_key_column => unique_id,               -- Reference the unique column from_
--the output of the ChunkText preparer
  source_data_column => chunk,                  -- Reference the output from the_
--ChunkText preparer
  destination_table => summarized_data__1321,
  destination_data_column => summary,
  destination_key_column => chunk_unique_id,
  options => [{"model": "model__1321"}]:JSONB -- Configuration for the_
--SummarizeText operation
);
```

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the [unnesting concept](#) for more detail.

両方のプリペアラをライブ自動処理に設定します

```
SELECT aidb.set_auto_preparer(chunking_preparer__1321, Live);
SELECT aidb.set_auto_preparer(summarizing_preparer__1321, Live);
```

処理するデータの挿入

ソースデータテーブルにデータを挿入すると、処理結果が自動的に流れるのが表示されます。

```
INSERT INTO source_table__1321
VALUES (1, This is a significantly longer text example that might require splitting_
--into smaller chunks. The purpose of this function is to partition text data into_
--segments of a specified maximum length, for example, this sentence 145 is_
--characters. This enables processing or storage of data in manageable parts.);
```

チャンクは自動的に計算されます。

```
SELECT * FROM chunked_data__1321;
```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
id | part_id | unique_id |
      chunk
-----+-----+-----+
1  |      0 | 1.part.0  | This is a significantly longer text example that might
require splitting into smaller chunks.
1  |      1 | 1.part.1  | The purpose of this function is to partition text data
into segments of a specified maximum length, for example, this sentence 145 is
characters.
1  |      2 | 1.part.2  | This enables processing or storage of data in manageable
parts.
(3 rows)
```

自動的に計算されたチャンクの概要

```
SELECT * FROM summarized_data__1321;

__OUTPUT__
chunk_unique_id |
summary
-----+-----
1.part.0        | text example might require splitting into smaller chunks .
1.part.1        | the purpose of this function is to partition text data into
segments of a specified maximum length .
1.part.2        | enables processing or storage of data in manageable parts .
(3 rows)
```

削除の場合にも、同じロジックの自動フローが発生します。

```
DELETE FROM source_table__1321 WHERE id = 1;
```

```
SELECT * FROM chunked_data__1321;

__OUTPUT__
id | part_id | unique_id | chunk
-----+-----+-----+
(0 rows)
```

```
SELECT * FROM summarized_data__1321;

__OUTPUT__
chunk_unique_id | summary
-----+-----
```

(次のページに続く)

(前のページからの続き)

(0 rows)

Preparers auto-processing example

この例では、AI Accelerator の *ChunkText operation* を使用したプリペアラー自動処理を使用します。

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the *unnesting concept* for more detail.

テーブルデータソースを使用したプリペアラー

```
-- Create source test table
CREATE TABLE source_table__1628
(
  id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  content TEXT NOT NULL
);

SELECT aidb.create_table_preparer(
  name => preparer__1628,
  operation => ChunkText,
  source_table => source_table__1628,
  source_data_column => content,
  destination_table => chunked_data__1628,
  destination_data_column => chunk,
  source_key_column => id,
  destination_key_column => id,
  options => {"desired_length": 150}::JSONB -- Configuration for the ChunkText
  ↳ operation
);

SELECT aidb.set_auto_preparer(preparer__1628, Live);

INSERT INTO source_table__1628
VALUES (1, This is a significantly longer text example that might require splitting
  ↳ into smaller chunks. The purpose of this function is to partition text data into
  ↳ segments of a specified maximum length, for example, this sentence 145 is
  ↳ characters. This enables processing or storage of data in manageable parts.);
```

```
SELECT * FROM chunked_data__1628;
```

```
__OUTPUT__
```

(次のページに続く)

(前のページからの続き)

```
id | part_id | unique_id |
    chunk
- - - - -
1 | 0 | 1.part.0 | This is a significantly longer text example that might
require splitting into smaller chunks.
1 | 1 | 1.part.1 | The purpose of this function is to partition text data
into segments of a specified maximum length, for example, this sentence 145 is
characters.
1 | 2 | 1.part.2 | This enables processing or storage of data in manageable
parts.
(3 rows)
```

```
INSERT INTO source_table__1628
VALUES (2, This sentence is short enough for desired_length=150 so it wont be chunked.
↳ But this second sentence will be in another chunk since both together are over 150
↳ characters.);
```

```
SELECT * FROM chunked_data__1628;
```

__OUTPUT__

```
id | part_id | unique_id |
    chunk
- - - - -
1 | 0 | 1.part.0 | This is a significantly longer text example that might
require splitting into smaller chunks.
1 | 1 | 1.part.1 | The purpose of this function is to partition text data
into segments of a specified maximum length, for example, this sentence 145 is
characters.
1 | 2 | 1.part.2 | This enables processing or storage of data in manageable
parts.
2 | 0 | 2.part.0 | This sentence is short enough for desired_length=150 so
it wont be chunked.
2 | 1 | 2.part.1 | But this second sentence will be in another chunk since
both together are over 150 characters.
(5 rows)
```

```
DELETE FROM source_table__1628 WHERE id = 1;
```

```
SELECT * FROM chunked_data__1628:
```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
id | part_id | unique_id | chunk
-----+-----+-----+-----
2 | 0 | 2.part.0 | This sentence is short enough for desired_length=150 so
↳ it wont be chunked.
2 | 1 | 2.part.1 | But this second sentence will be in another chunk since
↳ both together are over 150 characters.
(2 rows)
```

```
SELECT aidb.set_auto_preparer(preparer__1628, Disabled);
```

Preparers chunk text operation examples

これらの例では、AI Accelerator の *ChunkText operation* を使用したプリペアラーを使用しています。

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the *unnesting concept* for more detail.

プリミティブ

```
- - Only specify a desired length
SELECT * FROM aidb.chunk_text(This is a simple test sentence., {"desired_length":
↳ 10});
```

```
__OUTPUT__
part_id | chunk
-----+-----
0 | This is a
1 | simple
2 | test
3 | sentence.
(4 rows)
```

```
- - Specify a desired length and a maximum length
SELECT * FROM aidb.chunk_text(This is a simple test sentence., {"desired_length": 10,
↳ "max_length": 15});
```

```
__OUTPUT__
part_id | chunk
-----+-----
0 | This is a
1 | simple test
```

(次のページに続く)

(前のページからの続き)

```
2 | sentence.
(3 rows)
```

```
- - Named parameters
SELECT * FROM aidb.chunk_text(
    input => This is a significantly longer text example that might require splitting
    ↳into smaller chunks. The purpose of this function is to partition text data into
    ↳segments of a specified maximum length, for example, this sentence 145 is
    ↳characters. This enables processing or storage of data in manageable parts.,
    options => {"desired_length": 40}
);

__OUTPUT__
part_id |          chunk
-----+-----
0 | This is a significantly longer text
1 | example that might require splitting
2 | into smaller chunks.
3 | The purpose of this function is to
4 | partition text data into segments of a
5 | specified maximum length, for example,
6 | this sentence 145 is characters.
7 | This enables processing or storage of
8 | data in manageable parts.
(9 rows)
```

テーブルデータソースを使用したプリペアラ

```
- - Create source test table
CREATE TABLE source_table__1628
(
    id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL
);
INSERT INTO source_table__1628
VALUES (1, This is a significantly longer text example that might require splitting
    ↳into smaller chunks. The purpose of this function is to partition text data into
    ↳segments of a specified maximum length, for example, this sentence 145 is
    ↳characters. This enables processing or storage of data in manageable parts.),
    (2, This is sentence number one. This is sentence number one.);

SELECT aidb.create_table_preparer(
    name => preparer__1628,
    operation => ChunkText,
```

(次のページに続く)

(前のページからの続き)

```

source_table => source_table__1628,
source_data_column => content,
destination_table => chunked_data__1628,
destination_data_column => chunks,
source_key_column => id,
destination_key_column => id,
options => [{"desired_length": 120}]::JSONB -- Configuration for the ChunkText_
operation
);

SELECT aidb.bulk_data_preparation(preparer__1628);

SELECT * FROM chunked_data__1628;

__OUTPUT__
id | part_id | unique_id |
chunks
-----+-----+-----+
1 | 0 | 1.part.0 | This is a significantly longer text example that might
require splitting into smaller chunks.
1 | 1 | 1.part.1 | The purpose of this function is to partition text data
into segments of a specified maximum length, for example, this
1 | 2 | 1.part.2 | sentence 145 is characters. This enables processing or
storage of data in manageable parts.
2 | 0 | 2.part.0 | This is sentence number one. This is sentence number one.
(4 rows)

```

Preparers parse HTML operation examples

これらの例では、AI Accelerator の *ParseHtml operation* を使用したプリペアラールを使用しています。

プリミティブ

```
- - Default method is to structurally parse HTML to plaintext
SELECT * FROM aidb.parse_html(
    <html><body><h1>Hello World Heading</h1><p>Hello World paragraph</p></body></html>
);

__OUTPUT__
    parse_html
-----
Hello World Heading  +
                    +
Hello World paragraph+
```

(次のページに続く)

(前のページからの続き)

(1 row)

```
- - Parse Hello World HTML to plaintext
SELECT * FROM aidb.parse_html(
  html =>
    <h1>Hello, world!</h1>
    <p>This is my first web page.</p>
    <p>
      It contains some <strong>bold text</strong>, some <em>italic test</em>,
      and a <a href="https://google.com" target="_blank">link</a>.
    </p>

    <ol>
      <li>List item</li>
      <li>List item</li>
      <li>List item</li>
    </ol>,
  options => {"method": "StructuredPlaintext"} -- Default
);
```

__OUTPUT__

parse_html	
Hello, world!	+
	+
This is my first web page.	+
	+
It contains some bold text, some italic test, and a link.	+
	+
Postgres Logo Image	+
List item	+
List item	+
List item	+

(1 row)

```
- - Parse Hello World HTML to markdown-esque text that retains some syntactical
context
SELECT * FROM aidb.parse_html(
  html =>
    <h1>Hello, world!</h1>
```

(次のページに続く)

(前のページからの続き)

```

<p>This is my first web page.</p>
<p>
  It contains some <strong>bold text</strong>, some <em>italic test</em>,
  and a <a href="https://google.com" target="_blank">link</a>.
</p>



<ol>
  <li>List item</li>
  <li>List item</li>
  <li>List item</li>
</ol>,
options => {"method": "StructuredMarkdown"}
);

__OUTPUT__

                                parse_html
-----
--
# Hello, world!
+
+
+
This is my first web page.
+
+
+
It contains some **bold text** , some *italic test* , and a [link](https://google.
com).+
+
+
! [Postgres Logo Image](postgres_logo.png)
+
+
1. List item
+
2. List item
+
3. List item
+
(1 row)

```

テーブルデータソースを使用したプリペアラ

```

-- Create source test table
CREATE TABLE source_table__2772
(
    id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL
);
INSERT INTO source_table__2772
VALUES (1, <html><body><h1>Hello World Heading</h1><p>Hello World paragraph</p></body>
↪</html>),
      (2, <p>This is some <strong>bold text</strong>, some <em>italic test</em>, and
↪a <a href="https://google.com" target="_blank">link</a>.);

SELECT aidb.create_table_preparer(
    name => preparer__2772,
    operation => ParseHtml,
    source_table => source_table__2772,
    source_data_column => content,
    destination_table => destination_table__2772,
    destination_data_column => parsed_html,
    source_key_column => id,
    destination_key_column => id,
    options => [{"method": "StructuredPlaintext"}::JSONB -- Configuration for the
↪ParseHtml operation
);

SELECT aidb.bulk_data_preparation(preparer__2772);

SELECT * FROM destination_table__2772;

__OUTPUT__
id |                parsed_html
---+-----
1  | Hello World Heading      +
   |                          +
   | Hello World paragraph    +
   |                          +
2  | This is some bold text, some italic test, and a link.+
   |
(2 rows)

```

Preparers parse PDF operation examples

これらの例では、AI Accelerator の *ParsePdf operation* を使用したプリペアラーを使用しています。

💡 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the *unnesting concept* for more detail.

プリミティブ

```
- - Parse Hello World PDF with default options
- - Bytes can come from a binary column or a manual `pg_read_binary_file()` call
SELECT * FROM aidb.parse_pdf(
    ↳
    ↳ decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6672f506167657320332
    ↳ hex)
);

__OUTPUT__
part_id |      text
-----+-----
      0 | Hello World!+
      |
(1 row)
```

```
- - Manually specify the default options
SELECT * FROM aidb.parse_pdf(
    bytes => ↳
    ↳ decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6672f506167657320332
    ↳ hex),
    options => {"method": "Structured", "allow_partial_parsing": true} -- Default
);

__OUTPUT__
part_id |      text
-----+-----
      0 | Hello World!+
      |
(1 row)
```

テーブルデータソースを使用したプリペアラ

```

-- Create source test table
CREATE TABLE source_table__6124
(
    id      INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL
);
INSERT INTO source_table__6124
VALUES (1,
↳ decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6f672f506167657320332
↳ hex)),
        (2,
↳ decode(255044462d312e340a25b89a929d0a312030206f626a3c3c2f547970652f436174616c6f672f506167657320332
↳ hex));

SELECT aidb.create_table_preparer(
    name => preparer__6124,
    operation => ParsePdf,
    source_table => source_table__6124,
    source_data_column => content,
    destination_table => destination_table__6124,
    destination_data_column => parsed_pdf,
    source_key_column => id,
    destination_key_column => id,
    options => {"method": "Structured"}::JSONB -- Configuration for the ParsePdf
↳ operation
);

SELECT aidb.bulk_data_preparation(preparer__6124);

SELECT * FROM destination_table__6124;

__OUTPUT__
id | part_id | unique_id |  parsed_pdf
- - - - -
1  |         | 1.part.0  | Hello World!+
   |         |           |
2  |         | 2.part.0  | Hello World!+
   |         |           |
(2 rows)

```

Preparers perform OCR operation examples

AI Accelerator での *PerformOcr operation* でプリペアラーを使用する例。

 Tip

This operation transforms the shape of the data, automatically unnesting collections by introducing a *part_id* column. See the *unnesting concept* for more detail.

モデルの作成必須

この手順は、プリミティブの単一実行とプリペアラ－の大量実行に必要です。

```
- - Create the model. It must support the perform_ocr operation.
SELECT aidb.create_model(
    my_paddle_ocr_model,
    nim_paddle_ocr,
    credentials => {"api_key": "__NVIDIA_NIM_API_KEY__"}::JSONB
);
```

プリミティブ

[illegible]

(次のページに続く)

[illegible]

(次のページに続く)

(前のページからの続き)

[illegible]

(次のページに続く)

(前のページからの続き)

```
part_id |      text
-----+-----
      0 | Tesseract sample
(1 row)
```

テーブルデータソースを使用したプリペアラ

[illegible]

(次のページに続く)

(次のページに続く)

(前のページからの続き)

```

↪fouj15JY6RmYqNsUFPEyKJjfAMaLgmAPkpKkHMFc1jTZ0Eh9LzI0PJTBJbaXVhXtP1TA+FlqVtQ5h9dy8vQAA+wAeiiAqrOqY8
↪dkieXsoa26L+ML83aX+op5EFLRh4Z7ueux1YI6WBkZLZWA0cux9ZdUxEMe6RhDH+j1J/
↪RTUrJAZnH1HNS49PqfSzaq7IV+RqFFjljiE7GjzExlWXST15eJqV84F7m+BYF2Ss4Aeu7X/
↪AA6pKyYVrIGVceHdiVgTSsDID3gPAbgmCsChqmTRh7JWxgEh3qCFVPjZVzPYaR5xFuEediop7Q3VUaIgPuOE0Ia4kqw6fFE8xT
↪ACFVvmbuZKyYF9zp34gMK70WdTxXC8vMTSAPc16LDR07RWZFCWC5pEL2RphLKGuFHNwE7fE5F0zv1kySMfET950T317XBVLiaa
↪sbjndtsGSnAr5qE1M8jTGHMcWaldmaXfP/
↪tYRunDiCFakr2vibVRfiwEOICkkjFtUMVTW85ETDhVBFaNBwY33Sm9kVzixWdRUs9ZSvpaaNkTQ8umC/
↪Eouj17U0z8rVdWL+6qP/wBTV/eY/
↪wBDl+90jdTC1ayKd3tHM8sJUzJYZWB0b2G80DvUFTtdXV08b3RjzMiJUZYa6te9vM2ML83aX+oqAiKamfc9/
↪GNdm7Nkl3EbKhpYC4StFxxhdlrOENMwuwlgbF7MHErsPFY/
↪dY2MdUAg4yf1PqObZx6fUgGSekmjYD4XuaQFSQRUndZo8TJg43yJoc1zS1zT4E09FUhoMpl7sX7p8TuQqrteU7xc8PnAap2VNr
↪Rgg3QH6wP4l6tEzRknBLTS7p/8TSqmsFK/
↪z94nDI10Ki16pobLMBcxjP3GKZlLbEEeASHySNHgx6qaltMz7g7vUB0StAiEPBky6XeTTAfbhV0wSSUhd6WLyTG64KmiJdVPh
↪zDVXmojB04mil3c4BVVUmmxjGKmoAYFM2ptKrINVVe4Hgxns0KoZFX0VRv4d55X/
↪dIc0lVZhsmtGCSohZV3s3QlBftnigtGQX1MDzgErX9gcFU2juQLmXTtcq4iLGHYxbzezy+zT7KRlLV0Lg+iP6jcIuwkexCNZD
↪PUTnzSv8AqObZx6ftQf8AX9nJhwie1P2BzbOPT4JQwRyG++RsbQ84uLR+wObZx6fJObZx6fJObZx6fJObZx6fJObZx6fJObZx6f
↪p8mGz//xAAUEQEAAAAAAAAAAAAAAAAAACg/9oACAECAQE/ABK//8QAFBEBAAAAAAAAAAAAAAAAAAAAoP/
↪aAAgBAwEBPwASv//Z, base64));

```

```

SELECT aidb.create_table_preparer(
    name => preparer__1527,
    operation => PerformOcr,
    source_table => source_table__1527,
    source_data_column => content,
    destination_table => ocr_data__1527,
    destination_data_column => parsed__text,
    source_key_column => id,
    destination_key_column => id,
    options => {"model": "model__1527"}::JSONB -- Configuration for the PerformOcr_
↪operation
);

```

```
SELECT aidb.bulk_data_preparation(preparer__1527);
```

```
SELECT * FROM ocr_data__1527;
```

```
__OUTPUT__
```

id	part_id	unique_id	parsed__text
1	0	1.part.0	Trunch Parish Council
1	1	1.part.1	BANK RECONCILIATION AS AT 31STOCTOBER 2019
1	2	1.part.2	Account:
1	3	1.part.3	14,389.43
1	4	1.part.4	BANK STATEMENT BALANCE 30TH SEPTEMBER 2019
1	5	1.part.5	83.60

(次のページに続く)

(27 rows)

AI アクセラレーターデータ準備パイプラインは、OCR アダプターとのモデルの互換性を含む、作成時にオプションを検証します。その場合、次のエラーが返されます。

[illegible]

361

(前のページからの続き)

[illegible]

(次のページに続く)

(前のページからの続き)

```

operation => PerformOcr,
source_table => source_table__1527,
source_data_column => content,
destination_table => ocr_data__1527,
destination_data_column => parsed__text,
source_key_column => id,
destination_key_column => id,
options => {"model": "bert_model"}::JSONB -- Incompatible model
);
__OUTPUT__
ERROR: Failed to create preparer: The requested adapter is not supported by the
↳model provider: bert_local

```

Preparers summarize text operation examples

AI Accelerator での *SummarizeText operation* でプリペアラーを使用する例。

モデルの作成必須

この手順は、プリミティブの単一実行とプリペアラーの大量実行に必要です。

```

- - Create the model. It must support the decode_text and decode_text_batch
↳operations.
SELECT aidb.create_model(model__1952, t5_local);

```

プリミティブ

```

- - Keyword arguments
SELECT * FROM aidb.summarize_text(
    input => I dont want a babysitter. I am eleven years old. My babysitter is only
↳three years older than I am," she loudly yelled to her Mom. Now, she really wished
↳she had somebody with her as she heard the clicking, scratching noises outside of
↳the living room window. "This is silly. Its probably the storm," the girl said. She
↳regretted watching the horror show she had been tuned into for the last half hour.
↳As she searched for the remote to turn off the vampire movie, the front door blew
↳open with a thunderous noise. Carla whirled around to see a dark image.,
    options => {"model": "model__1952"}
);
__OUTPUT__

summarize_text
-----
↳
↳

```

(次のページに続く)

(前のページからの続き)

```

the girl yelled to her Mom as she heard the clicking, scratching noises outside of
↳the living room window . she regretted watching the horror show she had been tuned
↳into for the last half hour . the front door blew open with a thunderous noise .
(1 row)

```

```

-- Positional arguments

```

```

SELECT * FROM aidb.summarize_text(

```

```

    There are times when the night sky glows with bands of color. The bands may begin
↳as cloud shapes and then spread into a great arc across the entire sky. They may
↳fall in folds like a curtain drawn across the heavens. The lights usually grow
↳brighter, then suddenly dim. During this time the sky glows with pale yellow, pink,
↳green, violet, blue, and red. These lights are called the Aurora Borealis. Some
↳people call them the Northern Lights. Scientists have been watching them for
↳hundreds of years. They are not quite sure what causes them. In ancient times Long
↳Beach City College WRSC Page 2 of 2 people were afraid of the Lights. They imagined
↳that they saw fiery dragons in the sky. Some even concluded that the heavens were
↳on fire.,

```

```

    {"model": "model__1952"}

```

```

);

```

```

__OUTPUT__

```

```

↳summarize_text

```

```

-----
↳-----

```

```

the night sky glows with bands of color . they may begin as cloud shapes and then
↳spread into a great arc across the entire sky . the lights usually grow brighter,
↳then suddenly dim .

```

```

(1 row)

```

テーブルデータソースを使用したプリペアラ

```

-- Create source test table

```

```

CREATE TABLE source_table__1952

```

```

(

```

```

    id          INT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    content TEXT NOT NULL

```

```

);

```

```

INSERT INTO source_table__1952

```

```

VALUES (1, I dont want a babysitter. I am eleven years old. My babysitter is only
↳three years older than I am," she loudly yelled to her Mom. Now, she really wished
↳she had somebody with her as she heard the clicking, scratching noises outside of
↳the living room window. "This is silly. Its probably the storm," the girl said. She

```

(次のページに続く)

(前のページからの続き)

```

→regretted watching the horror show she had been tuned into for the last half hour.
→As she searched for the remote to turn off the vampire movie, the front door blew
→open with a thunderous noise. Carla whirled around to see a dark image.),
    (2, There are times when the night sky glows with bands of color. The bands
→may begin as cloud shapes and then spread into a great arc across the entire sky.
→They may fall in folds like a curtain drawn across the heavens. The lights usually
→grow brighter, then suddenly dim. During this time the sky glows with pale yellow,
→pink, green, violet, blue, and red. These lights are called the Aurora Borealis.
→Some people call them the Northern Lights. Scientists have been watching them for
→hundreds of years. They are not quite sure what causes them. In ancient times Long
→Beach City College WRSC Page 2 of 2 people were afraid of the Lights. They imagined
→that they saw fiery dragons in the sky. Some even concluded that the heavens were
→on fire.);

```

```

SELECT aidb.create_table_preparer(
    name => preparer__1952,
    operation => SummarizeText,
    source_table => source_table__1952,
    source_data_column => content,
    destination_table => summarized_data__1952,
    destination_data_column => summary,
    source_key_column => id,
    destination_key_column => id,
    options => {"model": "model__1952"}::JSONB -- Configuration for the
→SummarizeText operation
);

```

```

SELECT aidb.bulk_data_preparation(preparer__1952);

```

```

SELECT * FROM summarized_data__1952;

```

```

__OUTPUT__

```

```

id |
                                     summary
-----+-----
→
→
1  | the girl yelled to her Mom as she heard the clicking, scratching noises outside
→of the living room window . she regretted watching the horror show she had been
→tuned into for the last half hour . the front door blew open with a thunderous
→noise .
2  | the night sky glows with bands of color . they may begin as cloud shapes and
→then spread into a great arc across the entire sky . the lights usually grow
→brighter, then suddenly dim .
(2 rows)

```

モデルの互換性

AI アクセラレーターデータ準備パイプラインは、言語アダプターとのモデルの互換性を含む、作成時にオプションを検証します。その場合、次のエラーが返されます。

```
- - This model does not support the language adapter
SELECT aidb.create_model(bert_model, bert_local);
```

```
- - Single execution fails
SELECT * FROM aidb.summarize_text(
  input => Hello world,
  options => {"model": "bert_model"}
);
__OUTPUT__
ERROR: The requested adapter is not supported by the model provider: bert_local
```

```
- - Preparer creation fails
SELECT aidb.create_table_preparer(
  name => preparer_name,
  operation => SummarizeText,
  source_table => source_table__1952,
  source_data_column => content,
  destination_table => summarized_data__1952,
  destination_data_column => summary,
  options => {"model": "bert_model"}::JSONB -- Incompatible model
);
__OUTPUT__
ERROR: Failed to create preparer: The requested adapter is not supported by the_
↪model provider: bert_local
```

Pipelines data preparation output to volumes

プリペアラーパーパイプラインの出力は、Postgres テーブルまたは AIDB ボリューム PGFS 格納場所を使用して保存できます。ほとんどの例は、テーブルの使用方を示しています。このページでは、出力をボリュームに保存する方法について説明します。

出力フォーマット

各結果は独自のオブジェクトに保存されます。オブジェクトの名前/キーは、ソースレコードが持っているのと同じ名前/キーになります。チャンク、OCR、HTML 解析などの結果のリストを生成する操作は、各「部分」を独自のオブジェクトに保存します。

例、単一のオブジェクト `long_text.txt` はチャンクに分割されます。各チャンクは別個のオブジェクトに保存されます。これらの名前は、`long_text.txt.part.0`、`long_text.txt.part.1` などになります。

デスティネーションボリュームの準備

PGFS ストレージ場所と、目的の宛先の AIDB ボリュームが必要です。ボリュームは書き込み可能である必要があります。S3 バケットでもローカルディレクトリでも。

詳細については、*AIDB volumes for accessing PGFS storage locations* および *Pipelines PGFS reference* ドキュメントも参照してください。

ここでは、宛先としてローカルディレクトリ Postgres サーバー上を使用します。

ローカルディレクトリを準備します。

```
mkdir -p /tmp/pgfs/preparer_output_chunks
```

PGFS をセットアップし、ディレクトリへのアクセスを許可し、保存場所を構成します。

```
CREATE EXTENSION IF NOT EXISTS pgfs;
SET pgfs.allowed_local_fs_paths = /tmp/pgfs;

SELECT pgfs.create_storage_location(local_tmp_pgfs, file:///tmp/pgfs/);
```

次に、この保存場所の AIDB ボリュームを設定します。オプションでサブパスここでは `preparer_output_chunks` を構成します。必ず正しいデータ形式を構成してください。この場合、テキストチャンク準備パイプラインを使用するため `Text`。

```
SELECT aidb.create_volume(chunk_output_volume, local_tmp_pgfs, preparer_output_chunks/
→, Text);
```

ボリュームにアクセスできるかどうかをテストし、空であることを確認します。パイプラインは、空ではない宛先に書き込みますが、既存のデータを上書きするリスクがあります。

```
SELECT * FROM aidb.list_volume_content(chunk_output_volume);
```

ソースデータの準備

ボリュームとテーブルはどちらもソースとして使用できます。宛先タイプはこれに影響しません。ここでは、現実的な例としてボリュームを使用します。

```
SELECT pgfs.create_storage_location(
    text_files_bucket_aidb, s3://aidb-rag-app-txt,
    options => {"region": "eu-central-1", "skip_signature": "true"});
SELECT aidb.create_volume(text_files_volume, text_files_bucket_aidb, /, Text);

SELECT * FROM aidb.list_volume_content(text_files_volume);
```

Prepare パイプラインの作成と実行

`aidb.create_volume_preparer()` 呼び出しを使用して、作成したばかりのソースボリュームとデスティネーションボリュームを渡します。

注 ``destination\_data\_column`` および ``destination\_key\_column`` 引数は、デスティネーションとしてボリュームを使用する場合、省略する必要があります。ボリュームはキー/値ストアのように動作するため、構成する列名はありません。

```
SELECT aidb.create_volume_preparer(
    name => chunking_example_volumes,
    operation => ChunkText,
    source_volume_name => text_files_volume,
    destination_table => chunk_output_volume,
    options => {"desired_length": 1000}::JSONB
);

SELECT aidb.bulk_data_preparation(chunking_example_volumes);
```

結果を検査する

AIDB 内からの結果をリストします。

```
SELECT * FROM aidb.list_volume_content(chunk_output_volume);
```

__OUTPUT__	file_name	size	last_modified
	phil.txt.part.17	4	2025-05-02 15:56:50.161647402 UTC
	bilge.txt.part.24	9	2025-05-02 15:56:49.885069607 UTC
	mark.txt.part.5	4	2025-05-02 15:56:50.025494534 UTC
	phil.txt.part.28	4	2025-05-02 15:56:50.179620037 UTC
	phil.txt.part.10	97	2025-05-02 15:56:50.150750239 UTC
	mark.txt.part.2	3	2025-05-02 15:56:50.020690984 UTC
	TianQiu.txt.part.9	13	2025-05-02 15:56:49.809999366 UTC
	bilge.txt.part.23	5	2025-05-02 15:56:49.883231553 UTC
	phil.txt.part.26	8	2025-05-02 15:56:50.176445170 UTC
	TianQiu.txt.part.0	8	2025-05-02 15:56:49.792321435 UTC
	phil.txt.part.19	8	2025-05-02 15:56:50.165073307 UTC
	bilge.txt.part.15	12	2025-05-02 15:56:49.869647891 UTC
...			
	julien.txt.part.2	6	2025-05-02 15:56:49.909471517 UTC
	julien.txt.part.5	4	2025-05-02 15:56:49.914235651 UTC

(269 rows)

各チャンクは、シークエンシャル ID を持つ part として保存されていることがわかります。

Capabilities

パイプラインのライフサイクル

この高レベルの概要では、AI アクセラレーターパイプラインのパイプラインのライフサイクルについて説明します。

オプションで、保存場所を作成します

この手順はオプションであり、データ外部ストレージにアクセスする場合にのみ必要です。

処理用のデータは、データベースのテーブルまたは外部保存場所に保存できます。外部保存場所を使用してデータにアクセスする場合は、保存場所を作成する必要があります。この保存場所は、S3 バケットまたはローカルファイルシステムです。

AI Accelerator はストレージ場所を使用して、ボリュームを作成できます。このボリュームは、ナレッジベースがそのデータにアクセスするために使用できます。

プリペアラーを作成しますオプション

この手順はオプションであり、ソース AI データを前処理する場合にのみ必要です。

Pipelines システムを使用してプリペアラーを作成します。プリペアラーは、テーブルまたはボリュームからデータを取得し、モデルが使用できる形式で返すファンクションです。

どの構成でも、準備者には以下が必要です。

- 名前
- データ準備オペレーション
- 出力データが保存される宛先テーブルの名前
- 出力データが保存される宛先テーブルの列の名前

プリペアラーがテーブル用の場合、以下も必要です。

- ソーステーブルの名前
- データを含むソーステーブルの列の名前

ただし、プリペアラーがボリューム用の場合、以下が必要です。

- ボリュームの名前

プリペアラーが作成されると、デフォルトでは、ソースと宛先の「id」の列識別子を想定していますが、これはカスタマイズが可能です。

モデルを作成する

AI アクセラレーターパイプラインを使用して *Pipelines models reference* を作成します。このモデルには、タスクに適した任意の AI モデルを指定できます。

ナレッジベースを作成する

AI Accelerator Pipelines を使用してナレッジベースを作成します。ナレッジベースは、テーブルまたはボリュームからデータを取得し、モデルが使用できる形式で返すファンクションです。

デフォルトでは、ナレッジベースには以下のみが必要です。

- 名前
- 使用するモデルの名前

ナレッジベースがテーブル用の場合、以下も必要です。

- ソーステーブルの名前
- データを含むソーステーブルの列の名前
- 列のデータ型

ナレッジベースがボリューム用の場合、以下が必要です。

- ボリュームの名前
- データが含まれるボリューム内の列の名前

ナレッジベースを作成すると、デフォルトで、取得したデータのエンベディングを保存するためのベクトルテーブルが作成されます。このテーブルには、エンベディングを保存する列とデータのキーを保存する列があります。

ナレッジベースを作成するときに、ベクターテーブルの名前と、ベクター列とキー列の名前を指定できます。この機能は、ailedb に移行し、既存のベクトルテーブルを使用する場合に役立ちます。

エンベディングを作成する

埋め込みでは、ソーステーブルまたはボリュームから取得され、ベクトルデータ型にエンコードされるデータを確認します。そのベクトルデータはベクトルテーブルに保存されます。

エンベディング計算を構成および実行できる方法については、[Pipeline auto-processing](#) を参照してください。

クエリデータ

ナレッジベースを使用して、埋め込みデータを照会できます。ナレッジベースは、クエリに応じて、データまたはデータ自分自身にキーを返すことができます。取得されるデータの種類に応じて、テキストクエリまたは画像クエリを使用してデータを照会できます。

次のステップ

自動処理が有効になっている間、埋め込みは常に最新の状態であり、アプリケーションはナレッジベースを使用して、必要に応じてデータを照会できます。

クリーンアップ

埋め込みが必要でなくなった場合は、ナレッジベースを削除し、ベクターテーブルをドロップし、モデルを削除できます。

Pipeline auto-processing

概要

パイプライン自動処理は、ソースデータとパイプライン出力の同期を保つように設計されています。この機能がなければ、処理を手動でトリガーするか、外部スクリプト、スケジューラー、またはトリガーを提供する必要があります。自動処理には次の機能が含まれます。

- **完全な同期挿入/削除/更新**はすべて自動的に処理されます。更新の損失、データの欠落、または古いレコードはありません。
- **変更検出**新しいまたは変更されたレコードのみが処理されます。既知のレコードの不必要な再処理はありません。
- **バッチ処理**レコードは同時に処理されるバッチにグループ化されます。オーバーヘッドを削減し、最適なパフォーマンスを実現します。例、GPU ベースの AI モデル推論タスク。
- **バックグラウンド処理**バックグラウンド処理が有効になっている場合、パイプラインはバックグラウンドワーカプロセスで実行されるため、他の DB 操作をブロックまたは遅延しません。大規模なデータセットの処理に最適です。
- **Postgres テーブルのライブ処理**データソースが Postgres テーブルの場合、ライブトリガーベースの自動処理を有効にして、パイプライン結果が常に最新の状態になるようにできます。
- **素早いターンアラウンド**バッチの処理が完了すると、結果はすぐに利用できます。処理を開始するためにソースデータの完全なリストは必要ありません。これは、完全なリストに時間がかかる可能性がある大規模な外部ボリュームにとって重要です。

ナレッジベースパイプラインの例

製品説明のある製品を含む Postgres テーブルのナレッジベースを作成するとします。ユーザーは、product テーブルの操作をブロックまたは遅延することなく、埋め込みを常に同期させるようにバックグラウンド自動処理を構成します。

パイプラインは、既存の製品レコードをバックグラウンドで処理します。ユーザーは統計テーブルを照会して、進行状況を確認できます。

バックグラウンドプロセスは、新しいデータが挿入されるか、既存のデータが変更または削除されるときに実行されます。

ナレッジベースのクエリつまり取得操作は、わずかなバックグラウンド処理の遅延の後、常に正確な結果を返します。

サポートされているパイプラインとモード

ナレッジベースパイプライン

Source type	Destination type	Live	Background	Disabled (manual)
Table	Table	OK	OK	OK
Volume	Table	NO	OK	OK

ボリュームへの出力は、ナレッジベースパイプラインではサポートされていません。ベクトル機能を備えたデータベースが必要です。

Preparer パイプライン

Source type	Destination type	Live	Background	Disabled (manual)
Table	Table	OK	NO	OK
Table	Volume	NO	NO	OK
Volume	Table	NO	NO	OK
Volume	Volume	NO	NO	OK

_プリペアラパイプラインは、バッチ処理とバックグラウンド自動処理をまだサポートしていません。

自動処理モード

さまざまな要件とユースケースに合わせて、次の自動処理モードが利用できます。

ライブ

AIDB は、ソーステーブルに Postgres トリガーを設定して、変更をすぐに処理します。処理はトリガーファンクションで発生します。これは、結果が最新の状態になるようにデータを変更するのと同じトランザクションで発生することを意味します。

長所と短所

- トランザクション保証/即時結果。パイプラインの結果は常に最新のソースデータです。
- ソースデータの操作をブロック/遅延します。ソースデータの変更トランザクションは、処理が完了するまで遅延されます。

背景

AIDB は、バックグラウンドでの自動処理が構成されている各パイプラインの Postgres バックグラウンドワーカーを起動します。処理は、構成可能な `background_sync_interval` に基づいて非同期で発生します。パイプラインの処理方法の詳細については、[Change detection](#) を参照してください。

i 注釈

Make sure Postgres allows running enough background workers for the number of pipelines where you want to use this processing mode. Use the Postgres setting *max_worker_processes* to control this value.

長所と短所

- 非同期実行は、変更の処理中にソースのクエリーを遅延する必要がないことを意味します。
- 結果は遅延され、バックログになる場合があります。
- 大規模なデータセットに最適です。処理はバックグラウンドで継続的に発生し、ユーザーセッションまたは SQL ファンクション呼び出しに関連付けられません。

障害者

自動処理は無効になっています。 *aidb.bulk_embedding()* を手動で呼び出して、パイプラインを処理できます。

i 注釈

On table knowledge bases, change detection is also disabled, since it requires active triggers on the source table. This means manual processing (via *aidb.bulk_embedding()*) has to process all the records in the source.

可観測性

すべての自動処理モードの詳細なステータスと進行状況の出力を確認できます。

概要を把握するのに最適な場所は、統計テーブルです。ビュー *aidb.knowledge_base_stats* を検索するか、その短いエイリアス *aidb.kbstat* を使用します。ビューには、構成されたすべてのナレッジベースパイプライン、設定されている処理モード、および処理されたレコードに関する統計が表示されます。

```
SELECT * from aidb.kbstat;
```

```
__OUTPUT__
```

knowledge base	auto processing	table: unprocessed rows	volume: scans
completed	count(source records)	count(embeddings)	
kb_table_text_bg	Background	15	0
kb_table_text_manual	Disabled	15	0
kb_table_image_manual	Disabled	3	0
kb_table_text_live	Live	15	0
kb_table_image_bg	Background	3	0

(次のページに続く)

(前のページからの続き)

kb_volume_text_bg	Background		
6	7	7	
kb_volume_text_manual	Disabled		
0	0	0	
kb_volume_image_bg	Background		
4	177	6	
kb_volume_image_manual	Disabled		
1	177	6	
(9 rows)			

Change detection メカニズムは、自動処理の動作の中心です。ボリュームソースとテーブルソースでは異なります。このため、統計テーブルには、これら 2 つのソースタイプに対して異なる列があります。

- **table: unprocessed rows** 変更イベントのバックログにリストされる一意の行の数。* 自動処理が無効になっている場合、新しい変更イベントはキャプチャされません。
- **volume: scans completed** これまでに完了したソースの完全なリストの数。
- **count(source records)** このパイプラインのソースに存在するレコードの数。**テーブルソースの場合、この数値は常に正確です。** ボリュームソースの場合、この数値は、フルスキャンが完了した後にのみ更新されます。
- **count(embeddings)** このパイプラインのベクター宛先テーブルに存在するエンベディングの数。

構成

作成時に自動処理を構成できます。

- `aidb.create_table_knowledge_base` を使用する場合
- `aidb.create_volume_knowledge_base` を使用する場合

Tou は、既存のパイプライン用に構成することもできます。

- `aidb.set_auto_knowledge_base` を使用する場合

バッチ処理

バックグラウンドおよび無効モードでは、自動処理は構成可能なサイズのバッチで発生します。パイプラインは、すべてのソースレコードをバッチで処理します。各バッチのすべてのレコードは、可能な限り並列処理されます。これは、データの取得、エンベディングの計算、エンベディングの保存などのパイプラインステップがパラレル操作として実行されることを意味します。たとえば、テーブルをデータソースとして使用する場合、入力レコードのバッチが単一のクエリーで取得されます。ボリュームソースでは、同時要求を使用してレコードのバッチを取得します。

Pipelines knowledge base performance tuning は、最適なスループットを実現するためにバッチサイズを調整する方法について説明します。

変更検出

AIDB 自動処理は、テーブルおよびボリュームデータソースの変更検出メカニズムを中心に設計されています。変更検出により、必要な場合にのみデータを処理できます。

テーブルソース

バックグラウンド自動処理が有効になっている場合、Postgres トリガーがソーステーブルに設定されて、変更を検出します。これらのトリガーは非常に軽量です。彼らは変更イベントを記録し、「変更イベント」テーブルに挿入するだけです。トリガーファンクションでは実際の処理は発生しません。

バックグラウンドワーカーは、これらのイベントを非同期に処理します。

ボリュームソース

このソースタイプは、ソースレコードごとに `last_modified` タイムスタンプを提供します。システムは、これらのタイムスタンプを「状態」テーブルで追跡します。各パイプライン実行で、システムはボリュームの内容をリストし、タイムスタンプと比較して、レコードが変更または追加されたかどうかを確認します。

完全なリストが完了すると、システムは削除されたオブジェクトを検出します。この場合にのみ、以前に処理されたレコードがソースに存在しないことを確認できます。

注釈

Unfortunately, object stores (and other external storage locations supported by our volumes) have limited query capabilities. This means: Change detection for volumes is based on polling, that is, repeated listing. This might be an expensive operation when using cloud object stores like AWS S3. You can use a long *background_sync_interval* (like one per day) on pipelines with volume sources to control this cost.

GPU-accelerated vector index build

概要

ベクトルインデックスの構築は、非常に大規模なテーブルで数時間または数日実行されるコンピューティング集約的なプロセスです。EDB Postgres AI – AI アクセラレーター `aidb` は、NVIDIA GPU を使用して、VectorChord `vchordq` インデックスタイプのベクターインデックスの作成を高速化できます。

大きなベクター列にインデックスを作成するときに GPU アクセラレーションを使用して、ビルド時間を大幅に削減します。

要件

GPU アクセラレーションを使用する前に、環境が次の要件を満たしていることを確認します。

- サポートされている OS Ubuntu 22.04/24.04、Debian 12、RHEL 9。
- CPU アーキテクチャ x86_64。
- サポートされている NVIDIA GPU アーキテクチャ 8.9 および 9.0 た例えば、L40、L4、H100、H200、GH200 など。
- NVIDIA GPU ドライバーがインストールされ、機能している。

- NVIDIA CUDA ランタイム 12.9 以降がインストールされています。
- システムで使用可能な VectorChord 拡張機能 `vchordq` インデックスタイプの場合。インストールと使用方法の詳細については、[Vector Engine documentation](#) を参照してください。

API Compatibility の一般的なプラットフォームサポートも参照してください。

GPU アクセラレーションを有効にする

GPU アクセラレーションはデフォルトで無効になっています。セッションごと、またはサーバーレベルで有効にします。

- セッションごと

```
SET aidb.gpu_acceleration = enable;
```

- サーバー構成を保持してリロードします。

```
ALTER SYSTEM SET aidb.gpu_acceleration = enable;
SELECT pg_reload_conf();
```

無効にするには、値を `disable` に設定し、構成をリロードします。

クイックスタートの例

次の例では、ベクター列を持つテーブルを作成し、サンプルデータを挿入し、GPU 高速化された VectorChord インデックスを構築します。

- 1) テストテーブルを作成します。

```
CREATE TABLE test_10k_vecs (
  id          bigserial PRIMARY KEY,
  embedding   vector(2000)
);
```

- 2) サンプルベクトルを挿入します。

```
INSERT INTO test_10k_vecs (embedding)
SELECT arr.embedding
FROM generate_series(1, 10000) AS g(i)
CROSS JOIN LATERAL (
  SELECT array_agg(((g.i - 1) * 3 + gs.j)::real)
  FROM generate_series(1, 2000) AS gs(j)
) AS arr(embedding);
```

- 3) GPU 高速化インデックスを構築します。

```
SELECT aidb.create_vector_index_on_gpu(
  table_name    => public.test_10k_vecs,
  column_name   => embedding,
```

(次のページに続く)

(前のページからの続き)

```
cluster_count    => 1000
);
```

ファンクションが完了すると、テーブルに VectorChord vchordq インデックスが使用できます。

ファンクションリファレンス

`aidb.create_vector_index_on_gpu()` ファンクションは、GPU で VectorChord インデックスビルド全体を実行します。VectorChord の CREATE INDEX に似ています。

すべてのパラメーターを使用した署名の例

```
SELECT aidb.create_vector_index_on_gpu(
  table_name      => public.test_10k_vecs,
  column_name     => embedding,
  cluster_count   => 1000,
  sampling_factor  => 256,
  kmeans_iterations => 10,
  kmeans_nredo    => 1,
  distance_operator => ip,
  batch_size      => 1000000
);
```

パラメーター

- `table_name` 完全修飾テーブル名 `schema.table`。
- `column_name` インデックスするベクター列の名前。
- `cluster_count` インデックス VectorChord 「リスト」 に使用するクラスターの数。
- `stable_factor` オプション。k-means のサンプルベクトルの数。デフォルト 256
- `kmeans_itrations` オプション。k-means の反復数。デフォルト 10
- `kmeans_nredo` オプション。k-means リスタートの数。デフォルト 1
- `distance_operator` オプション。距離メトリック。ip 内積、l2 ユークリッド、または cos コサインのいずれか。デフォルト ip
- `batch_size` オプション。処理バッチサイズを制御します。バッチが大きくなると、メモリの使用量が増加しますが、スループットを向上させることができます。

検証

ファンクションが返った後、インデックスが存在し使用可能なことを確認します。

- テーブルのリストインデックス

```
SELECT indexname, indexdef
FROM pg_indexes
WHERE schemaname = public AND tablename = test_10k_vecs;
```

- 目的の距離演算子を使用してクエリ例を実行し、EXPLAIN を使用してインデックスの使用を確認します。

EXPLAIN を使用してインデックスの使用を検証する

:query_vec を、列と同じディメンションのクエリベクトルリテラルまたはパラメーターに置き換えます。インデックスの作成に使用した distance_operator と一致する演算子を選択します。

- L2 距離 l2

```
EXPLAIN
SELECT id
FROM test_10k_vecs
ORDER BY embedding <-> :query_vec
LIMIT 10;
```

- 内積 ip

```
EXPLAIN
SELECT id
FROM test_10k_vecs
ORDER BY embedding <#> :query_vec
LIMIT 10;
```

- コサイン距離 cos

```
EXPLAIN
SELECT id
FROM test_10k_vecs
ORDER BY embedding <=> :query_vec
LIMIT 10;
```

計画には、完全なシークエンシャルスキャンではなく、インデックスアシストパスたとえば、インデックススキャンを示す必要があります。テストに必要な場合は、一時的に SET enable_seqscan = off; を設定して、検証中のインデックスの使用を促進します。

ベクトル距離演算子<-> L2、<#> 内積、および<=> コサインの詳細については、[Vector Engine](#) ドキュメントを参照してください。

チューニングとガイダンス

- クラスター数リストの数に関する VectorChord ガイダンスに従って、データサイズとクエリパターンに適切な値を選択します。

- バッチサイズ十分な GPU メモリが利用可能な場合、ビルドを高速化するために増加します。メモリ不足状態が発生した場合は、削減します。
- 距離演算子 データの照会を計画している方法と一致します `ip`、`l2`、または `cos`。

障害対応

- GPU アクセラレーションが有効になっていない `aidb.gpu_acceleration = 'enable'` を確認し、`ALTER SYSTEM` 経由で設定されている場合は構成をリロードします。
- 互換性のある GPU が検出されませんでした NVIDIA ドライバーと CUDA ランタイムのインストールを確認します。
- VectorChord コンポーネントの欠落 VectorChord 拡張機能とそのインデックスタイプがノードにインストールされていることを確認します。
- 無効な引数ベクトル列が存在し、固定ディメンションがあり、テーブル名が完全修飾されていることを確認します。

注意事項と制限事項

- GPU アクセラレーションは、現在 VectorChord `vchordq` インデックスビルドに適用されています。
- サポートされているハードウェアと OS は、要件にリストされているプラットフォームに制限されます。
- ファンクションは同期的に実行され、インデックスの構築が完了または失敗したときに返されます。

Pipelines models

Pipelines には、モデルの構成済みのインスタンスを管理するモデルレジストリがあります。埋め込みや取得など、モデルを使用するパイプラインファンクションは、作成されたモデルを参照する必要があります。

- パイプラインで [Using models in AI Accelerator Pipelines](#) を行う方法。
- モデルとの対話に使用できる [Pipelines model primitives](#) を見つけます。
- Pipelines に付属する Supported models を参照してください。
- パイプラインでの models with OpenAI API-compatible services and Nvidia NIM の使用。

次のステップ

モデルに慣れたら、Knowledge Bases でこれらのモデルを使用する方法を学ぶことができます。

Using models in AI Accelerator Pipelines

Pipelines には、モデルの構成済みのインスタンスを管理するモデルレジストリがあります。埋め込みや取得など、モデルを使用するパイプラインファンクションは、このレジストリのモデルを参照する必要があります。

プリロードされたモデルを発見する

Pipelines には、すぐに使用できる事前構成されたモデルのセットが付属しています。

それらを見つけるには、次のクエリーを実行します。

```
SELECT * FROM aidb.models;
```

このクエリは、システムで現在作成されているすべてのモデルのリストを結果ます。モデルを作成していない場合は、パイプラインに付属するデフォルトのモデルが表示されます。

name	provider	options
bert	bert_local	{ "config={}" }
clip	clip_local	{ "config={}" }
t5	t5_local	{ "config={}" }
dummy	dummy	{ "config={}" }

bert、clip、および t5 モデルはすべて事前構成されており、すぐに使用できます。dummy モデルは、テストのために使用できるプレースホルダーモデルです。

プロキシを使用する場合、それが *configured on the Postgres server* であることを確認します。

独自のモデルを作成できます。これを行うには、aidb.create_model フังก์ションを使用できます。この例は、モデルを作成する方法を示しています。

```
SELECT aidb.create_model(my_model, bert_local);
```

このコマンドは、デフォルトの bert_local モデルプロバイダーを使用する my_model という名前のモデルを作成します。この例では追加パラメーターを使用しないため、my_model は、プリロードされた bert_local モデルと同じです。モデルの作成時に構成するには、Creating a model with a configuration を参照してください。

モデルプロバイダーの検出

次のクエリを実行して、使用可能なモデルプロバイダーを見つけることができます。

```
SELECT * FROM aidb.model_providers;
```

__OUTPUT__

server_name	server_description
	server_options
t5_local	A simple language model, ideal for translation, summarization, and question answering. Runs locally.
embeddings	For any model that implements the OpenAI embeddings API.
completions	For any model that implements the OpenAI completions API.

(次のページに続く)

(前のページからの続き)

openai_embeddings	For any embeddings model on the OpenAI platform
openai_completions	For any completions model on the OpenAI platform.
nim_completions	For any model that implements the OpenAI completions API on_
↳ NVIDIA NIM.	
nim_embeddings	For any model that implements the OpenAI embeddings API on_
↳ NVIDIA NIM.	
nim_clip	Vision/text model, ideal for encoding text and images, runs on_
↳ NVIDIA NIM.	
nim_reranking	Reranking model, runs on NVIDIA NIM.
bert_local	Simple language model, ideal for encoding text, runs locally.
clip_local	A vision/text model, ideal for encoding text and images, runs_
↳ locally.	
dummy	Provides fake data, ideal for testing.
(12 rows)	

このクエリーは、システムで現在使用可能なすべてのモデルプロバイダーのリストを結果ます。これらのプロバイダーとその機能の詳細については、Supported models でご覧になれます。

構成を使用したモデルの作成

モデルを作成するときにモデルにオプションを渡すことができます。例、モデル構成を指定できます。

```
SELECT aidb.create_model(my_model,
                        bert_local,
                        {"model": "sentence-transformers/paraphrase-multilingual-
↳ MiniLM-L12-v2",
                        "revision": "main"}::JSONB);
```

このコマンドは、bert_local モデルプロバイダーと HuggingFace の sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 モデルを使用する my_model という名前のモデルを作成します。revision オプションは、使用するモデルのバージョンを指定します。オプションは、JSONB にキャストされる一重引用符で囲まれた文字列を含む JSONB オブジェクトとして渡されます。文字列には、単一の JSON オブジェクトでモデル構成を定義するキーと値のペアがあります。

構成と資格情報を使用してモデルを作成する

構成と資格情報の使用は、他の Supported models の出番です。構成でモデル名を指定することにより、別のモデルを作成できます。openai-completions および openai-embeddings モデルは、どちらも OpenAI のコンプリーションおよび埋め込み API を利用するために作成できるモデルです。同様に、nim-completions および nim-embeddings モデルは、NVIDIA のコンプリーションおよび埋め込み API を利用するために作成できるモデルです。

この方法でモデルを作成するときは、`aidb.create_model` ファンクションに詳細情報を提供する必要があります。Completions には、OpenAI で使用するモデルの選択を含む多くのオプションがあります。補完と埋め込みは両方とも API 資格情報が必要です。この例は、OpenAI 完了モデルを作成する方法を示しています。

```
SELECT aidb.create_model(
  my_openai_model,
  openai_completions,
  {"model": "gpt-4o"}::JSONB,
  {"api_key": "sk-abc123xyz456def789ghi012jkl345mn"}::JSONB
);
```

`api_key` 値を独自の OpenAI API キーに置き換えます。資格情報の処理の詳細については、Supported models を参照してください。次に、パイプラインファンクションで `my_openai_model` モデルを使用でき、この例では、OpenAI の GPT-4o モデルを活用できます。

OpenAI エンベディングモデルの作成は、コンプリートモデルの作成に似ています。

```
SELECT aidb.create_model(
  my_openai_embeddings,
  openai_embeddings,
  {"model": "text-embedding-3-small"}::JSONB,
  {"api_key": "sk-abc123xyz456def789ghi012jkl345mn"}::JSONB
);
```

このコマンドは、`my_openai_embeddings` という名前でも `text-embedding-3-small` モデルを作成します。このモデルをパイプラインファンクションで使用して、テキストデータのエンベディングを生成できます。

モデルが資格情報を使用する場合、モデルプロバイダーの `create_model` の最初の使用で設定された資格情報は、そのモデルの後続のすべての使用で使用されます。同じプロバイダーを使用する異なるモデルに異なる資格情報を使用することはできません。

資格情報を変更する必要がある場合は、新しいモデルを作成するときに `replace_credentials` オプションを使用できます。このオプションは、モデルプロバイダーの資格情報を更新します。

```
SELECT aidb.create_model(
  my_new_openai_model,
  openai_completions,
  {"model": "gpt-4o"}::JSONB,
  {"api_key": "sk-ngl234meh789ski1111bid011i"}::JSONB,
  replace_credentials => true
);
```

資格情報のみを変更する必要がある場合は、`SELECT aidb.delete_model('my_new_openai_model')` を使用して新しいモデルを削除できます。

OpenAI 互換 API でモデルを使用する

補完と埋め込みモデルは、OpenAI 互換の API で動作します。この互換性、より多くの機種を接続して使用できます。適切な API エンドポイントを `aidb.create_model` ファンクションのオプションの `url` オプションに渡します。

モデルの詳細については、[Completions](#) および [Embeddings and vector representations](#) を参照してください。

Pipelines model primitives

ほとんどのユースケースでは、`aidb Knowledge Bases` を使用してモデルと対話することをお勧めします。多くのアプリケーションの埋め込みの作成と一致するデータの取得を管理できます。

ただし、モデルと直接対話する必要がある場合は、次のプリミティブを使用できます。`encode` 機能はテキストと画像の埋め込みを生成し、`decode` 機能は埋め込みからテキストを生成します。

テキスト埋め込みを生成する

`aidb.encode_text` を呼び出して、特定のテキスト入力のベクトル表現を生成します。

```
select aidb.encode_text(t5, This is text I need vectorized.);
```

バッチ推論の場合、`aidb.encode_text_batch` を呼び出します。

```
select aidb.encode_text_batch(my_bert_model, ARRAY[
  Dwayne "The Rock" Johnson,
  Hatsune Miku,
  Jim Lehrer,
  Dikembe Mutombo
])
```

画像埋め込みを生成する

`aidb.encode_image` を呼び出して、特定の画像入力のベクトル表現を生成します。

```
-- BYTEA type casting included to demonstrate what data type is used
select aidb.encode_image(clip, image1 AS BYTEA);
```

バッチ推論の場合、`aidb.encode_image_batch` を呼び出します。

```
-- BYTEA type casting included to demonstrate what data type is used
select aidb.encode_image_batch(clip, ARRAY[
  image_of_the_rock_cooking AS BYTEA,
  image_of_miku_singing AS BYTEA,
  image_of_jim_reporting_news AS BYTEA,
  image_mutombo_blocking_shots AS BYTEA
])
```

プロンプト応答を生成する

aidb.decode_text を呼び出して、特定のテキスト入力からプロンプト応答を生成します。

```
SELECT * FROM aidb.decode_text(t5, translate to german: hello world);
__OUTPUT__
decode_text
-----
Hallo Welt
(1 row)
```

バッチ推論の場合、aidb.decode_text_batch を呼び出します。

```
SELECT * FROM aidb.decode_text_batch(t5, ARRAY[
  translate to german: i am an example,
  summarize: The missile knows where it is at all times. It knows this because it
  knows where it isnt. By subtracting where it is from where it isnt, or where it
  isnt from where it is (whichever is greater), it obtains a difference, or deviation.
  The guidance subsystem uses deviations to generate corrective commands to drive
  the missile from a position where it is to a position where it isnt, and arriving
  at a position where it wasnt, it now is.
]);
__OUTPUT__

decode_text_batch
-----
ich bin ein Beispiel
the guidance subsystem uses deviations to generate corrective commands . it is
driving the missile from a position where it is, or where it isnt, and arriving at
a position where it wasnt .
(2 rows)
```

リランクテキスト

aidb.rerank_text を呼び出して、テキストの再ランキングログを取得します。

```
SELECT * from aidb.rerank_text(my_nim_reranker, how can I open a door?, {Ask for help,
  Push the handle, Lie down and wait, Shout at it}::text[]) ORDER BY logit_score
DESC;
__OUTPUT__
```

text	logit_score	id
Push the handle	-3.697265625	1
Ask for help	-6.2578125	0
Shout at it	-7.39453125	3

(次のページに続く)

(前のページからの続き)

Lie down and wait	-11.375	2
(4 rows)		

Pipelines supported models

このセクションでは、EDB Postgres AI - AI Accelerator パイプラインでサポートされているモデルとその機能の詳細を提供します。

- *Support matrix* 組み込みモデルの場合。
- *T5* 。
- Embeddings and vector representations 、 *openai-embeddings* および *nim-embeddings* を含む
- *Completions* 、 *openai-completions* および *nim-completions* を含む
- *Gemini*
- *BERT* 。
- *CLIP* 。
- *NIM_CLIP* 。
- *Llama* 。
- *Reranking (NIM)* 。
- *NIM OCR* 。

Support matrix

モデルサポートマトリックス

さまざまな操作で AI Factory によってサポートされている **埋め込み**モデルの概要。各モデルの詳細については、そのモデルプロバイダーの特定のページを参照してください。

Provider	Model name	Operations	Configuration
<i>bert_local</i>	<i>bert_all-MiniLM-L12-v1</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/all-MiniLM-L12-v1", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_all-MiniLM-L6-v1</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/all-MiniLM-L6-v1", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_all-MiniLM-L6-v2</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/all-MiniLM-L6-v2", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_all-distilroberta-v1</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/all-distilroberta-v1", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_msmarco-bert-base-dot-v5</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/msmarco-bert-base-dot-v5", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_multi-qa-MiniLM-L6-cos-v1</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/multi-qa-MiniLM-L6-cos-v1", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_multi-qa-MiniLM-L6-dot-v1</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/multi-qa-MiniLM-L6-dot-v1", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_paraphrase-TinyBERT-L6-v2</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/paraphrase-TinyBERT-L6-v2", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_paraphrase-multilingual-MiniLM-L12-v2</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2", "revision": "main"}</code>
<i>bert_local</i>	<i>bert_paraphrase-multilingual-mpnet-base-v2</i>	<i>TextEncoder</i>	<code>{"model": "sentence-transformers/paraphrase-multilingual-mpnet-base-v2", "revision": "main"}</code>
<i>clip_local</i>	<i>openai/clip-vit-base-patch32</i>	<i>TextEncoder, ImageEncoder</i>	<code>{}</code>
<i>llama_instruct_local</i>	<i>HuggingFaceTB/SmolLM2-1.7B-Instruct</i>	<i>TextDecoder</i>	<code>{"model": "HuggingFaceTB/SmolLM2-1.7B-Instruct", "revision": "main"}</code>
<i>llama_instruct_local</i>	<i>HuggingFaceTB/SmolLM2-135M-Instruct</i>	<i>TextDecoder</i>	<code>{"model": "HuggingFaceTB/SmolLM2-135M-Instruct", "revision": "main"}</code>
<i>llama_instruct_local</i>	<i>HuggingFaceTB/SmolLM2-360M-Instruct</i>	<i>TextDecoder</i>	<code>{"model": "HuggingFaceTB/SmolLM2-360M-Instruct", "revision": "main"}</code>

次のページに続く

表 22 – 前のページからの続き

Provider	Model name	Operations	Configuration
<i>llama_instruct_local</i>	<i>TinyLlama/TinyLlama-1.1B-Chat-v1.0</i>	<i>TextDecoder</i>	<i>{ "model": "TinyLlama/TinyLlama-1.1B-Chat-v1.0", "revision": "main" }</i>
<i>t5_local</i>	<i>t5-base</i>	<i>TextEncoder, TextDecoder</i>	<i>{ "model": "t5-base", "revision": "main" }</i>
<i>t5_local</i>	<i>t5-large</i>	<i>TextEncoder, TextDecoder</i>	<i>{ "model": "t5-large", "revision": "main" }</i>
<i>t5_local</i>	<i>t5-small</i>	<i>TextEncoder, TextDecoder</i>	<i>{ "model": "t5-small", "revision": "main" }</i>

T5

モデル名 `t5_local`

T5 について

T5 は、入力テキストを出力テキストに変換するテキストからテキストへの変換モデルです。これは、エンコーダーが入力を処理し、デコーダーが出力を生成する Transformer アーキテクチャに基づいています。大規模なテキストデータのコーパスで事前トレーニングされており、チャットボット、翻訳、要約、質問回答などの幅広い自然言語処理タスクを実行できます。

[T5 on Wikipedia](#) の詳細を読んでください。

サポートされている `aidb` 操作

- `encode_text`
- `decode_text`
- `encode_text_batch`
- `decode_text_batch`

サポートされているモデル

- `t5-small` デフォルト

詳細は、[Support matrix](#) を参照してください。

デフォルトモデルの作成

```
SELECT aidb.create_model(my_t5_model, t5_local);
```

特定のモデルの作成

```
SELECT aidb.create_model(  
  another_t5_model,  
  t5_local,  
  {"model": "t5-large", "revision": "main"}::JSONB  
)
```

モデル構成設定

T5 モデルでは、次の構成設定が利用できます。

- `model` —使用する T5 モデル。デフォルトは `t5-small` です。
- `revision` —使用するモデルのリビジョン。デフォルトは `refs/pr/15` です。
- `temperature` —サンプリングに使用する温度。デフォルトは `0.0` です。
- `seed` —サンプリングに使用するランダムシード。デフォルトは `1599222198345926291` です。

- `max_tokens` —生成するトークンの最大数。デフォルトは 16384 です。
- `repeat_penalty` —使用するリピートペナルティ。デフォルトは 1.1 です。
- `repeat_last_n` —繰り返しペナルティを考慮するトークンの数。デフォルトは 64 です。

モデルの資格情報

T5 モデルには資格情報は必要ありません。

BERT

モデル名 `bert_local`

BERT について

BERT 双方向エンコーダー表現トランスフォーマーは、テキスト分類、テキスト生成、テキスト補完などの自然言語処理タスクに使用されるトランスフォーマーベースのモデルです。大規模なテキストデータのコーパスで事前トレーニングされており、幅広い自然言語処理タスクを実行できます。

BERT on Wikipedia) および Hugging Face Web サイトの [Sentence Transformers](#) の詳細をご覧ください。

サポートされている `aidb` 操作

- `encode_text`
- `encode_text_batch`

サポートされているモデル

- `promotes/all-MiniLM-L6-v2` デフォルト

詳細は、[Support matrix](#) を参照してください。

デフォルトモデルの作成

```
SELECT aidb.create_model(my_bert_model, bert_local);
```

特定のモデルの作成

オプション JSONB オブジェクトでモデルとリビジョンを指定できます。この例では、`another_bert_model` という名前で `sentence-transformers/all-distilroberta-v1` モデルを作成します。

```
select aidb.create_model(  
  another_bert_model,  
  bert_local,  
  {"model": "sentence-transformers/all-distilroberta-v1", "revision": "main"}::JSONB  
)
```

モデル構成設定

次の構成設定は BERT モデルで使用できます。

- **model** —使用する BERT モデル。デフォルトは `sentence-transformers/all-MiniLM-L6-v2` です。
- **revision** —使用するモデルのリビジョン。デフォルトは `refs/pr/64` です。このエントリは、Hugging Face リポジトリのモデルリビジョンへの参照です。これは、使用するモデルバージョン、この場合は `this branch` を指定するために使用されます。

モデルの資格情報

BERT モデルはローカルで実行されるため、資格情報は必要ありません。

CLIP

モデル名 `clip_local`

CLIP について

CLIP Contrastive Language-Image Pre-training は、自然言語監視から視覚的概念を学習するモデルです。これは、幅広い視覚と言語タスクに使用できるゼロショット学習モデルです。

CLIP on OpenAI's website の詳細を参照してください。

サポートされている **aidb** 操作

- `encode_text`
- `encode_text_batch`
- `encode_image`
- `encode_image_batch`

サポートされているモデル

- `openai/clip-vit-base-patch32` デフォルト

詳細は、*Support matrix* を参照してください。

デフォルトモデルの作成

```
SELECT aidb.create_model(my_clip_model, clip_local);
```

モデルはデフォルトの `openai/clip-vit-base-patch32` のみであるため、構成でモデルを指定する必要はありません。CLIP モデルには資格情報は必要ありません。

特定のモデルの作成

CLIP モデルで使用可能な他のモデル構成はありません。

モデル構成設定

CLIP モデルでは、次の構成設定が使用できます。

- **model** —使用する CLIP モデル。デフォルトは `openai/clip-vit-base-patch32` で、これが使用可能な唯一のモデルです。
- **revision** —使用するモデルのリビジョン。デフォルトは `refs/pr/15` です。このエントリは、Hugging Face リポジトリのモデルリビジョンへの参照です。これは、使用するモデルバージョン、この場合は *this branch* を指定するために使用されます。
- **image_size** —使用する画像のサイズ。デフォルトは 224 です。

モデルの資格情報

CLIP モデルには資格情報は必要ありません。

Llama

モデル名 `llama_instruct_local`

ラマについて

LLaMA は、テキスト生成タスク用に設計されたデコーダー専用のトランスフォーマーモデルです。エンコーダーなしで標準の Transformer アーキテクチャを使用し、入力トークンを自己回帰的に処理して、シーケンス内の次のトークンを予測します。公開されているテキストの大規模なコーパスで事前トレーニングされた LLaMA は、チャット、コード生成、要約、質問回答などのさまざまな自然言語タスクを処理できます。

[Llama on Wikipedia](#) の詳細を読んでください。

サポートされている **aidb** 操作

- `decode_text`
- `decode_text_batch`

サポートされているモデル

詳細は、*Support matrix* を参照してください。

デフォルトモデルの作成

```
SELECT aidb.create_model(my_llama_model, llama_instruct_local);
```

特定のモデルの作成

```
SELECT aidb.create_model(  
  another_llama_model,  
  llama_instruct_local,  
  {"model": "HuggingFaceTB/SmolLM2-135M-Instruct", "revision": "main"}::JSONB  
)
```

モデルの実行

```
SELECT aidb.decode_text(llama_instruct, Why is the sky blue?);
```

モデル構成設定

次の構成設定は Llama モデルで使用できます。

- **model** —使用する Llama モデル。デフォルトは TinyLlama/TinyLlama-1.1B-Chat-v1.0 です。
- **revision** —使用するモデルのリビジョン。デフォルトは main です。
- **system_prompt** —オプション。一般的な LLM 応答をガイドするための基本的な指示。
- **use_flash_attention** —モデルがフラッシュアテンションを使用するかどうかを示します。デフォルトは false です。
- **seed** —サンプリングに使用するランダムシード。デフォルトは 1599222198345926291 です。
- **temperature** —サンプリングに使用する温度。デフォルトは 0.2 です。
- **sample_len** —生成するトークンの最大数。デフォルトは 64 です。
- **repeat_last_n** —繰り返しペナルティを考慮するトークンの数。デフォルトは 64 です。
- **repeat_penalty** —使用するリピートペナルティ。デフォルトは 1.1 です。
- **top_p** —トークン配布をフィルタリングするための累積確率しきい値。デフォルトは 0.9 です。
- **use_kv_cache** —生成中にアテンションキー/値ペアの再利用を有効にして、より速くデコードします。デフォルトは true です。

モデルの資格情報

ローカル Llama モデルには資格情報は必要ありません。

Gemini

モデル名 gemini

Gemini について

Google Gemini は、Google が開発したマルチモーダル大規模言語モデルのファミリーです。モデルには、Gemini API を介してアクセスできます。

サポートされている aidb 操作

- `decode_text`
- `decode_text_batch`

サポートされているモデル

プロバイダーがサポートするテキスト生成モデル。サポートされている Gemini モデルのリストを参照してください [here](#)。

Gemini モデルの作成

`aidb.create_model` ファンクションを使用して、サポートされている Gemini モデルを作成できます。

この例では、`my_gemini_model` という名前で Gemini 2.5 Flash モデルを作成します。

```
SELECT aidb.create_model(  
  my_gemini_model,  
  gemini,  
  {"model": "gemini-2.5-flash"}::JSONB,  
  {"api_key": "AIzaSyBXnKr4yJG2lMhyvza81q3-WFuP7Tk3r8z"}::JSONB  
);
```

モデルの実行

```
SELECT aidb.decode_text(my_gemini_model, Why is the sky blue?);
```

モデル構成設定

次の構成設定は OpenAI モデルで使用できます。

- `model` —使用するモデル。
- `url` —使用するモデルの URL。この設定はオプションであり、カスタムモデル URL を指定するために使用できます。
- `max_concurrent_requests` —OpenAI モデルに対して行う同時要求の最大数。デフォルトは 25 です。
- `thinking_budget` —応答を生成するために使用される思考トークンの数。この設定はオプション。詳細は [here](#) を参照してください。

モデルの資格情報

これらのモデルを提供するサービスでは、次の資格情報が必要になる場合があります。

- **api_key** —ベアラートークン認証に使用する API キー。api_key は、ヘッダーフィールドで `Authorization: Bearer <api_key>` として送信されます。

Completions

モデル名 `completions`

モデルのエイリアス

- `openai_completions`
- `nim_completions`

コンプリートについて

Completions により、OpenAI API 互換のテキスト生成モデルを使用できます。チャット/テキスト変換、テキスト補完、およびその他のテキスト生成タスクに適しています。

モデルの名前に基づいて、モデルプロバイダーはそれに応じてデフォルトを設定します。

- `completions` または `openai_completions` として呼び出された場合、モデルプロバイダーはデフォルトで OpenAI API を使用します。
- `nim_completions` として呼び出された場合、モデルプロバイダーはデフォルトで NVIDIA NIM API を使用します。

サポートされている `aidb` 操作

- `decode_text`
- `decode_text_batch`

サポートされているモデル

- プロバイダーがサポートするテキスト生成モデル。

サポートされている OpenAI モデル

サポートされている OpenAI モデルのリスト [here](#) を参照してください。

サポートされている NIM モデル

- `ibm/granite-guardian-3.0-8b`
- `ibm/granite-3.0-8b-instruct`
- `ibm/granite-3.0-3b-a800m-instruct`
- `meta/llama-3.3-70b-instruct`
- `meta/llama-3.2-3b-instruct`

- *meta/llama-3.2-1b-instruct*
- *meta/llama-3.1-405b-instruct*
- *meta/llama-3.1-70b-instruct*
- *meta/llama-3.1-8b-instruct*
- *meta/llama3-70b-instruct*
- *meta/llama3-8b-instruct*
- *nvidia/llama-3.1-nemotron-70b-instruct*
- *nvidia/llama-3.1-nemotron-51b-instruct*
- *nvidia/nemotron-mini-4b-instruct*
- *nvidia/nemotron-4-340b-instruct*
- *google/shieldgemma-9b*
- *google/gemma-7b*
- *google/codegemma-7b*

デフォルトモデルの作成

コンプリートにはデフォルトのモデルはありません。 `aidb.create_model` ファンクションを使用して、サポートされているモデルを作成できます。

OpenAI モデルの作成

`aidb.create_model` ファンクションを使用して、サポートされている OpenAI モデルを作成できます。

この例では、`my_openai_model` という名前で GPT-4o モデルを作成します。

```
SELECT aidb.create_model(  
  my_openai_model,  
  openai_completions,  
  {"model": "gpt-4o"}::JSONB,  
  {"api_key": "sk-abc123xyz456def789ghi012jkl345mn"}::JSONB  
);
```

NIM モデルの作成

```
SELECT aidb.create_model(  
  my_nim_completions,  
  nim_completions,  
  {"model": "meta/llama-3.2-1b-instruct"}::JSONB,  
  credentials=>{"api_key": "sk-abc123xyz456def789ghi012jkl345mn"}::JSONB);
```

モデル構成設定

次の構成設定は OpenAI モデルで使用できます。

- `model` —使用するモデル。
- `url` —使用するモデルの URL。この設定はオプションであり、カスタムモデル URL を指定するために使用できます。`openai\_completions` または `completions` が `model` である場合、`url` はデフォルトの `https://api.openai.com/v1/chat/completions` になります。`nim\_completions` が `model` の場合、`url` はデフォルトの `https://integrate.api.nvidia.com/v1/chat/completions` になります。
- `max_concurrent_requests` —OpenAI モデルに対して行う同時要求の最大数。デフォルトは 25 です。
- `max_tokens` —モデルによって生成されたトークンの最大数。デフォルトは NULL です。* NULL に設定されている場合、最大トークンパラメーターは省略され、モデルはデフォルトの内部最大値に設定されます。

最大トークン

`max_tokens` パラメーターは、モデルが生成できるトークンの数を設定します。これは上限として機能し、トークンの予算制約を管理できます。

パラメーター

`size` —モデルによって生成されたトークンの最大数。 `format` —最大トークンパラメーターをモデルに送信する方法を定義します。`default` に設定されている場合、`max\_completion\_tokens` パラメーターは要求ペイロードで提供されます。これは、OpenAI が提供するモデルに最適です。legacy に設定されている場合、`max_tokens` パラメーターは要求ペイロードで提供されます。これは、非 OpenAI モデルでより一般的です。* `both` に設定されている場合、`max_completion_tokens` と `max_tokens` の両方が要求ペイロードで提供されます。

使用例

```
SELECT aidb.create_model(
    my_completions_model,
    completions,
    {"model": "meta/llama-3.2-1b-instruct", "max_tokens": {"format": "default",
    → "size": 1024}})::JSONB
```

モデルの資格情報

これらのモデルを提供するサービスでは、次の資格情報が必要になる場合があります。注 `api_key` および `basic_auth` は排他的です。これら 2 つのオプションのうちの 1 つのみを使用できます。

- `api_key` —ベアラートークン認証に使用する API キー。 `api_key` は、ヘッダーフィールドで `Authorization: Bearer <api_key>` として送信されます。
- `basic_auth` — HTTP Basic 認証の資格情報。ここで提供された資格情報は、`Authorization: Basic <basic_auth>` としてそのまま送信されます。サーバーが HTTP 基本認証でユーザー名/パスワード

を必要とする場合、`username:password` 構文を使用します。トークンのみが必要な場合は、トークンのみを提供します。

Embeddings

モデル名 `embeddings`

モデルのエイリアス

- `openai_embeddings`
- `nim_embeddings`

埋め込みについて

OpenAI embeddings は、OpenAI API 互換のテキスト埋め込みモデルを使用できるテキスト埋め込みモデルです。テキスト分類、クラスタリング、およびその他のテキスト埋め込みタスクに適しています。

モデルの名前に基づいて、モデルプロバイダーはそれに応じてデフォルトを設定します。

- `embeddings` または `openai_embeddings` として呼び出された場合、モデルプロバイダーはデフォルトで OpenAI API を使用します。
- `nim_embeddings` として呼び出された場合、モデルプロバイダーはデフォルトで NVIDIA NIM API を使用します。

サポートされている `aidb` 操作

- `encode_text`
- `encode_text_batch`

サポートされているモデル

- プロバイダーがサポートするテキスト埋め込みモデル。

サポートされている OpenAI モデル

- OpenAI でサポートされているテキスト埋め込みモデル。これには、`text-embedding-3-small` および `text-embedding-ada-002` が含まれます。サポートされている OpenAI モデルのリスト [here](#) を参照してください。
- デフォルトは `text-embedding-3-small` です。

サポートされている NIM モデル

- `nvidia/nv-embedqa-e5-v5` デフォルト

OpenAI モデルでのデフォルトの作成

```
SELECT aidb.create_model(my_openai_embeddings,
                        openai_embeddings,
                        credentials=>{"api_key": "sk-
↪abc123xyz456def789ghi012jkl345mn"}::JSONB);
```

この例では、モデルをデフォルトで `text-embedding-3-small` に設定しているため、構成でモデルを指定する必要はありません。ただし、資格情報で OpenAI API キーを渡す必要があります。そのためには、資格情報を名前付けパラメーターとして渡す必要があります。

特定のモデルの作成

`aidb.create_model` ファンクションを使用して、サポートされている OpenAI 埋め込みモデルを作成できます。この例では、`my_openai_model` という名前で `text-embedding-3-small` モデルを作成します。

```
SELECT aidb.create_model(
  my_openai_model,
  openai_embeddings,
  {"model": "text-embedding-3-small"}::JSONB,
  {"api_key": "sk-abc123xyz456def789ghi012jkl345mn"}::JSONB
);
```

この例では、構成オプションと資格情報を渡しているため、前の例とは異なり、資格情報を名前付けパラメーターとして渡す必要はありません。

モデル構成設定

次の構成設定は OpenAI モデルで使用できます。

- `model` —使用する OpenAI モデル。
- `url` —使用するモデルの URL。この値はオプションであり、カスタムモデル URL を指定するために使用できます。`openai\_completions` または `completions` が `model` である場合、`url` はデフォルトの `https://api.openai.com/v1/chat/completions` になります。`nim\_completions` が `model` の場合、`url` はデフォルトの `https://integrate.api.nvidia.com/v1/chat/completions` になります。
- `max_concurrent_requests` —OpenAI モデルに対して行う同時要求の最大数。デフォルトは 25 です。
- `max_batch_size` —1 つの要求でモデルに送信するレコードの最大数。デフォルトは 50.000 です。

バッチおよび並列処理

`embeddings`、`openai_embeddings`、および `nim_embeddings` のモデルプロバイダーは、バッチ要求とコンカレント要求の送信をサポートしています。設定 `max_concurrent_requests` および `max_batch_size` は、この動作を制御します。モデルプロバイダーがナレッジベースパイプラインなどからレコードセットを受信すると、次のことが発生します。

- ナレッジベースパイプラインがバッチサイズ 10.000 で構成されていると仮定します。モデルプロバイダーは `max_batch_size=1000` および `max_concurrent_requests=5` で構成されています。

- プロバイダーは最大 1000 レコードを収集し、単一の要求でモデルに送信します。
- 入力レコードがなくなるまで、このような大規模な要求を 5 つ同時に送信します。
- この例では、プロバイダーは合計 10 バッチを送受信する必要があります。最初の 5 件を送信した後、応答が返されるのを待機します。応答を受信すると、別の要求を送信できます。*これは、プロバイダーが次の 5 つを送信する前に 5 つすべてが返されるのを待たないことを意味します。代わりに、常に最大 5 つの要求を処理中に保持します。

i 注釈

The settings `max_concurrent_requests` and `max_batch_size` can have a significant impact on model performance. But they highly depend on the hardware and infrastructure.

デフォルトをそのまま使用することをお勧めします [Pipelines knowledge base performance tuning](#)

デフォルトの `max_batch_size` の 50.000 は、パイプラインがバッチの実際のサイズを制御できるように意図的に高くしています。

これらのモデルを提供するサービスでは、次の資格情報が必要になる場合があります。注 `api_key` および `basic_auth` は排他的です。次の 2 つのオプションのいずれかのみを使用できます。

- `api_key` — ベアラートークン認証に使用する API キー。 `api_key` は、ヘッダーフィールドで `Authorization: Bearer <api_key>` として送信されます。
- `basic_auth` — HTTP Basic 認証の資格情報。ここで指定された資格情報は、`Authorization: Basic <basic_auth>` としてそのまま送信されます。サーバーが HTTP 基本認証でユーザー名/パスワードを必要とする場合、`username:password` 構文を使用します。トークンのみが必要な場合は、トークンのみを提供します。

NIM_CLIP

モデル名 `nim_clip`

CLIP について

CLIP Contrastive Language-Image Pre-training は、自然言語監視から視覚的概念を学習するモデルです。これは、幅広い視覚および言語タスクに使用できるゼロショット学習モデルです。

この特定のモデルは、NVIDIA NIM で実行されます。NIM の CLIP の詳細については、[here](#) を参照してください。

サポートされている `aidb` 操作

- `encode_text`
- `encode_text_batch`
- `encode_image`
- `encode_image_batch`

サポートされているモデル

build.nvidia.com

- nvidia/nvclip デフォルト

デフォルトモデルの作成

```
SELECT aidb.create_model(  
  my_nim_clip_model,  
  nim_clip,  
  credentials=>{"api_key": "<API_KEY_HERE>"}::JSONB  
);
```

モデルはデフォルトの nvidia/nvclip のみであるため、構成でモデルを指定する必要はありません。

モデル構成設定

CLIP モデルでは、次の構成設定が使用できます。

- model —使用する NIM モデル。デフォルトは nvidia/nvclip で、これが使用可能な唯一のモデルです。
- url —デフォルトは <https://integrate.api.nvidia.com/v1/embeddings> です。

モデルの資格情報

build.nvidia.com 内で実行する場合、次の資格情報が必要です。

- api_key —認証に使用する build.nvidia.com API キー。

NIM OCR

モデル名 nim_paddle_ocr

NVIDIA NIM イメージ OCR について

NVIDIA NIM for Image OCR API は、NVIDIA NIM 上の画像からテキストを抽出する光学文字認識マイクロサービスおよびインターフェイスです。

サポートされている aidb 操作

- Perform_ocr

サポートされているモデル

画像 OCR 用の NVIDIA NIM

- baidu/paddleocr デフォルト

デフォルトモデルの作成

```
SELECT aidb.create_model(  
    my_paddle_ocr_model,  
    nim_paddle_ocr,  
    credentials => {"api_key": "__NVIDIA_NIM_API_KEY__"}::JSONB  
);
```

現在、NVIDIA NIM for Image OCR API、デフォルトの baidu/paddleocr でサポートされているモデルは 1 つだけであるため、構成でモデルを指定する必要はありません。

モデル構成設定

次の構成設定は PADDLE_OCR モデルで使用できます。

- url デフォルトは <https://ai.api.nvidia.com/v1/cv/baidu/paddleocr> です。

モデルの資格情報

build.nvidia.com 内で実行する場合、次の資格情報が必要です。

- api_key — 認証に使用する build.nvidia.com API キー。

Reranking (NIM)

モデル名 `nim_reranking`

リランキングについて

リランキングは、結果を関連性によって並べ替えてより正確にするテキスト検索の方法です。クロスアテンションメカニズムを使用してドキュメントにスコアを与え、最初の検索結果を向上させます。

サポートされている aidb 操作

- rerank_text

サポートされているモデル

build.nvidia.com

- nvidia/llama-3.2-nv-rerankqa-1b-v2 デフォルト

例

ファンクションは、「リランククエリー」としての文字列と、再ランク付けするテキストの配列を受け入れます。出力の id 列は、入力配列内のテキストのインデックスを参照します。

```
SELECT * from aidb.rerank_text(my_nim_reranker, how can I open a door?, {"Ask for help,  
→ Push the handle, Lie down and wait, Shout at it"}::text[]) ORDER BY logit_score_  
→ DESC;
```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
      text      | logit_score | id
-----+-----+---
Push the handle | -3.697265625 | 1
Ask for help    |  -6.2578125 | 0
Shout at it     | -7.39453125 | 3
Lie down and wait |  -11.375 | 2
(4 rows)
```

デフォルトモデルの作成

```
SELECT aidb.create_model(
  my_nim_reranker,
  nim_reranking,
  {"url": "http://nim-nv-rerankqa-llama-1-1xgpu-g6-predictor.default.svc.cluster.
  ↳local/v1/ranking", "model": "nvidia/llama-3.2-nv-rerankqa-1b-v2"}
);
```

この例では、資格情報を必要としないローカルに展開された NIM モデルを使用します。 *Using models in AI Accelerator Pipelines* の説明に従って、資格情報およびその他の構成を提供できます。

モデル構成設定

次の構成設定は、NIM モデルで使用できます。

- **model** —使用する NIM モデル。デフォルトは `nvidia/llama-3.2-nv-rerankqa-1b-v2` で、これが唯一サポートされているモデルです。
- **url** —使用するモデルの URL。この設定はオプションであり、カスタムモデル URL を指定するために使用できます。デフォルトは `https://ai.api.nvidia.com/v1/retrieval` です。
- **tls_config** —セキュア HTTPS エンドポイントのオプションの TLS 構成。このフィールドは、次のオブジェクトを受け入れます。
- **insecure_skip_verify** ブール値 `true` の場合、証明書の検証を無効にします開発または自己署名証明書に役立ちます。
- **ca_path** 文字列 HTTPS 接続を検証するときに信頼する PEM エンコードされた認証局ファイルへのパス。

次の例は、証明書無効を使用せずにモデルを作成します

```
SELECT aidb.create_model(
  my_nim_reranker_notsecure,
  nim_reranking,
  {
    "model": "nvidia/llama-3.2-nv-rerankqa-1b-v2",
    "url": "https://localhost/v1/ranking",
```

(次のページに続く)

(前のページからの続き)

```

        "tls_config": {
            "insecure_skip_verify": true
        }
    }::jsonb
);

```

次の例は、証明書を使用してモデルを作成する方法を示しています。

```

SELECT aidb.create_model(
    my_nim_reranker,
    nim_reranking,
    {
        "model": "nvidia/llama-3.2-nv-rerankqa-1b-v2",
        "url": "https://my.secure.endpoint/v1/ranking",
        "tls_config": {
            "ca_path": "/etc/aidb/myCA.pem"
        }
    }::jsonb
);

```

モデルの資格情報

build.nvidia.com 内で実行する場合、次の資格情報が必要です。

- `api_key` — 認証に使用する NVIDIA Cloud API キー。

Model Serving Quickstart

モデルサービングのクイックスタート

このページは、AI Factory 内で **Model Serving** の使用を開始する方法と、サポートドキュメントの場所をすばやく理解するのに役立ちます。

AI Factory の Model Serving を使用すると、AI モデル NVIDIA NIM マイクロサービスなどをスケーラブルな運用グレードの推論サービスとして展開できます。これは、Kubernetes ネイティブモデルのサービングエンジンである **KServe** を搭載しています。

どこから始めるべきか

1. 概念を学ぶ

モデルを展開する前に、モデル提供がどのように動作し、AI Factory エコシステムにどのように適合するかを理解することが役立ちます。

- [Model Serving Explained](#)
- [AI Factory terminology](#)

2.AI Factory がモデルサービスを統合する方法を理解する

Model Serving は以下と相互作用します。

- モデルライブラリー展開用のモデルイメージを参照および管理します近日提供予定。
- ナレッジベース AIDB モデル提供によって提供される埋め込みモデルを使用する可能性のあるベクターストア
- **Gen AI Builder** アプリケーションは、推論のためにモデルサービングエンドポイントを呼び出す場合があります

3. ハウツーガイドに従ってください

モデルを展開または管理する準備ができている場合

- *How-To Deploy AI Models from Model Library*

入門チェックリスト

このチェックリストを使用して、経験レベルに応じて進行状況をガイドします。

新しいユーザーの場合 101 レベル

- *Model Serving Explained* を読む
- レビューキー *AI Factory terminology*
- KServe が *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)* でモデルサービングを強化する方法を理解する
- *Model Library Reference Manual* を理解する 近日公開予定

Getting Started - AI Factory 101

Kubernetes 101 レベルに精通している既存のユーザー向け

- HCP プロジェクトでの Kubernetes アクセスを確認する
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)* および *AI Factory terminology* を確認します
- クラスターの前提条件を準備します。
- GPU ノードプール必要な場合
- NVIDIA デバイスプラグイン必要な場合
- モデルイメージのコンテナレジストリへのアクセス
- 基本的な KServe リソースを構成します。
- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.comcontainer with KServe*

- *How-To Configure a ClusterServingRuntime*
- *How-To Create an InferenceService for NVIDIA NIM Container*

Getting Started - AI Factory 101

上級者向け 201 レベル

- デプロイされた InferenceService のリソース使用量を調整します。
- *How-To Update GPU Resources for an InferenceService*
- 導入されたモデルを監視します。
- *How-To Monitor deployed models with KServe*
- *How-To Monitor deployed models with KServe* 近日公開予定
- トラフィックルーティング、カナリアロールアウト、およびスケーリングを理解します。
- *Gen AI Builder Concepts (Assistants, Knowledge, Tools, Threads)*
- 未来 高度なハウツー

Building Production-Ready AI Features - AI Factory 201

エキスパートユーザー向け 301 レベル

- 独自のカスタムモデル画像を管理
- カスタム ServingRuntime 定義をビルドおよび構成する
- KServe でトランスフォーマーとエクスペレイラーを使用する近日公開
- KServe にモデルを展開するための CI / CD パイプラインを構築する
- 高度な可観測性のためのインストルメント推論サービス

Advanced AI Factory Usage and Extensibility - AI Factory 301

次のステップ

- *Model Serving Explained*
- *How-To Deploy AI Models from Model Library*
- *Model Library Reference Manual*

このクイックスタートは、AI Factory 内の Model Serving の起動ポイントとして使用します。

Using models with...

このセクションでは、AI アクセラレータパイプラインでモデルを使用する特定の方法について説明します。これらのテクニックは、ローカルまたはクラウドで実行されている API 互換サービスをパイプラインで使用する方法、および NVIDIA NIM を利用する方法を示しています。

- *Using an OpenAI-compatible API with Pipelines* パイプラインを使用する場合。
- *How-To Deploy an NVIDIA NIM Microservices available on build.nvidia.com/container with KServe* パイプラインを使用する場合。 * *Using NVIDIA NIM Microservices (Hosted)*
- *Using NVIDIA NIM Microservices available on build.nvidia.com in your environment*

Using an OpenAI-compatible API with Pipelines

OpenAI 準拠の API を利用するには、エンベディングまたはコンプリーションモデルプロバイダーを使用できます。ナレッジベースは最初にエンコードする必要があるため、ナレッジベースでのみエンベディングモデルプロバイダーを使用できます。

OpenAI 互換 API を使用する理由

OpenAI 互換 API を使用する理由の例には次のものが含まれます。

- *Ollama* を実行しているローカルシステムがあり、そのローカルシステムでエンベディングを処理する場合。これは、*Ollama as a server* を構成していることを前提としています。
- 別のモデルまたは特別に調整されたモデルを提供するサービスへのアクセスがある場合は、他のモデルの代わりにそれを使用できます。

モデルの作成

このプロセスの開始点は、モデルを作成することです。モデルを作成するときに、`options` および `credentials` を登録に渡すことができます。デフォルトは OpenAI サービスエンドポイントを指します。デフォルトをオーバーライドすることにより、任意のサービスをポイントできます。

この例では、ローカル Ollama サーバーを使用するモデルを作成します。

```
select aidb.create_model(
my_local_ollama,
embeddings,
{"model":"llama3.2", "url":"http://llama.local:11434/v1/embeddings"}::JSONB);
```

モデル名とモデルプロバイダー

モデル名は最初のパラメーターです。この例では、`my_local_ollama` に設定されています。

モデルプロバイダーを `embeddings` として指定します。これは、デフォルトで OpenAI サーバーを使用するプロバイダーです。構成パラメーターを使用して、この値をオーバーライドして、準拠サーバーと通信できます。

構成

次のパラメーターは構成です。これは JSON 文字列です。展開すると、モデルと URL の 2 つのパラメーターがあります。

```
{"model":"llama3.2", "url":"http://llama.local:11434/v1/embeddings"}::JSONB
```

この場合、モデルを “*llama3.2*” 、比較的新しく強力なモデルに設定します。ollama run llama3.2 を実行して、サーバー上でモデルをプルおよび起動することを忘れないでください。

次の JSON 設定は重要な設定であり、ailedb モデルが使用するエンドポイントをオーバーライドします。この例では

- サーバーは llama.local というマシンで実行されています。
- ポート 11434 Ollama のデフォルトポートが HTTP この場合 HTTPS ではなくを介したサービス要求に対して開いています。
- サーバー上のエンドポイントへのパスは/v1/embeddings で、OpenAI と同じです。

これらのコンポーネントを組み合わせると、エンドポイントとして http://llama.local:11434/v1/embeddings を取得します。

これで構成パラメーターの作成は完了です。

エンドポイントが API キーを必要とする場合、それは credentials パラメーターで渡されます。これはローカルモデルであるため、資格情報を渡す必要はありません。

!!!注 注 pgvector でインデックスを使用する場合、*pgvector indexing limitations* を検討してください。Aidb は、現在インデックスを自動的に構成しませんが、手動で追加する場合は、モデルが使用するディメンション数をサポートしていることを確認してください。

前に作成したモデル名を使用して、他のパイプラインモデルと同じようにモデルを使用します。この例は、モデルを使用してエンベディングを取得する方法を示しています。

```
select aidb.encode_text(my_local_ollama,I like it);
```

Pipelines がすべての接続管理を処理するため、ユーザーはデータとモデルの結果に集中できます。

Using NVIDIA NIM Microservices

AI アクセラレータを備えた build.nvidia.com モデルで利用可能な NVIDIA NIM マイクロサービスを使用できます。モデルは、build.nvidia.com または自分の制御の下で環境で実行できます。

両方のシナリオで AI アクセラレータで NVIDIA NIM を使用する方法を学習できます。

- *Using NVIDIA NIM Microservices (Hosted)*
- *Using NVIDIA NIM Microservices available on build.nvidia.com in your environment*

Using NVIDIA NIM Microservices (Hosted)

NVIDIA の `build.nvidia.com` でホストされている NVIDIA NIM を使用するには、最初に使用するモデルを選択する必要があります。このチュートリアルでは、NVIDIA NIM モデル `llama-3.3-70b-instruct` を使用します。

前提条件

- `build.nvidia.com` アカウント。(お持ちでない場合は、[here](#) を作成できます。)

`build.nvidia.com` の構成

1. モデルを選択します

NVIDIA's model library からモデルを選択します。この例では、`llama-3.3-70b-instruct` モデルを使用します。

2.API キーを生成する

モデルのページから API キーを作成できます。

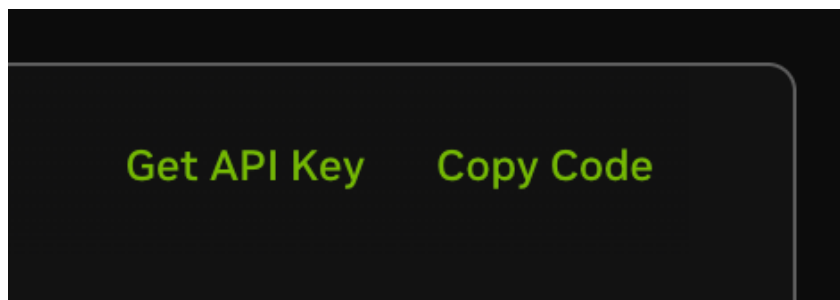


図 1: Get API Key

API キーを生成するには、**Get API Key** を選択します。ログインしていない場合は、ログインするように求められます。サインアップしていない場合は、サインアップするように求められます。モデルと対話するために使用できる API キーを受け取ります。この例では、このキーを<NIM API KEY> として参照します。

モデルと AI Accelerator の統合

EDB Postgres AI で AI アクセラレーターを有効にする

EDB Postgres Advanced Server、EDB Postgres Extended Server、またはコミュニティ Postgres インスタンスが実行されている状態で、データベースに接続し、AI Accelerator 拡張機能を有効にします。

```
CREATE EXTENSION aidb CASCADE;
```

2. モデルを登録する

```
SELECT aidb.create_model(  
    my_nim_llm,  
    nim_completions,
```

(次のページに続く)

(前のページからの続き)

```
{
  "model": "meta/llama-3.3-70b-instruct",
  "api_key": "<NIM API KEY>"
}::JSONB
);
```

3. モデルを実行します

モデルを操作するには、次のクエリを実行します。

```
SELECT aidb.decode_text(my_nim_llm, Tell me a short, one sentence story);
__OUTPUT__

          decode_text
-----
--
As the clock struck midnight, a single tear fell from the porcelain dolls glassy
eye.
```

出力は異なる場合があります。AI Accelerator と統合された NVIDIA NIM を正常に使用できました。

Using NVIDIA NIM Microservices available on build.nvidia.com in your environment

EDB AI Factory は、Kubernetes マニフェストを使用して **KServe** で **NVIDIA NIM microservices** を展開します。このガイドでは、次の NIM タイプのすぐに使用できる YAML テンプレートを提供します。

- チャットコンプリート LLM
- テキストの埋め込み
- テキストリランカー
- 画像の埋め込み
- 画像 OCR

プロセスは次のとおりです。

1. NVIDIA NGC アカウントと API キーを設定します。
2. NGC アクセスとイメージプル用の Kubernetes シークレットを作成します。
3. GPU ノードの作成
4. モデルインスタンスごとに **InferenceService** を展開します。
5. 検証
6. AIDB での使用

サポートされているモデルと GPU 要件

Model Type	NIM Model	NVIDIA Documented Resource Requirements
Text Completion	llama-3.3-70b-instruct	4 × L40S
Text Embeddings	arctic-embed-l	1 × L40S
Image Embeddings	nvclip	1 × L40S
OCR	paddleocr	1 × L40S
Text Reranking	llama-3.2-nv-rerankqa-1b-v2	1 × L40S

1.NVIDIA NGC API キーの設定

1. build.nvidia.com に移動し、NVIDIA アカウントでログインします。
2. 右上のメニューで、**Setup** → **API Key** を選択します。
3. **Generate API Key** をクリックします。
4. キーをコピーして、安全に保存します。Kubernetes シークレットに必要です。

2.Kubernetes シークレットを作成する

NGC API キーシークレットを作成します。

```
kubectrl create secret generic nvidia-nim-secrets \
  --from-literal=NGC_API_KEY="YOUR\_NGC\_API\_KEY"
```

3.GPU ノードを作成する

Kubernetes クラスターに、選択した NIM モデルの最小 GPU 要件を満たすノードがあることを確認します。

ノードラベル

```
nvidia.com/gpu: "true"
```

ノードテイント

```
Key: nvidia.com/gpu
Value: "true"
Effect: NoSchedule
```

これらの設定により、GPU を必要とするワークロードが GPU 対応ノードでのみ正しくスケジュールされます。

4. 次の KServe リソースを展開します

YAML ファイルをダウンロードします

- チャット完了ランタイム *llm-runtime.yaml*
- チャット完了サービス *llm-service.yaml*
- テキスト埋め込みランタイム *image-embed-runtime.yaml*
- テキスト埋め込みサービス *image-embed-service.yaml*
- テキストリランカーランタイム *rerank-runtime.yaml*
- Text Reranker サービス *rerank-service.yaml*
- 画像埋め込みランタイム *image-embed-runtime.yaml*
- 画像埋め込みサービス *image-embed-service.yaml*
- 画像 OCR ランタイム *ocr-runtime.yaml*
- 画像 OCR サービス *ocr-service.yaml*

必要に応じて <your-storage-class>、および環境に一致するようにリソース設定を更新します。

クラスターに適用します

```
kubectl apply -f llm-runtime.yaml
kubectl apply -f llm-service.yaml
```

展開ステータスを確認します

```
kubectl get clusterservingruntime
kubectl get inferencesservice
```

導入する モデルタイプごとに繰り返します。提供された curl コマンドを使用して、エンドポイントを検証します。登録セクションの SQL コマンドを使用して、EDB Postgres AI Accelerator と統合します。

5. 検証

エンドポイントを見つけます。

```
kubectl get inferencesservice llama33-8b-instruct -o jsonpath={.status.url}
```

テストしてください

```
curl -X POST "<ENDPOINT>/v1/chat/completions" \
-H "Content-Type: application/json" \
-d {"model": "meta/llama3-8b-instruct", "messages": [{"role": "user", "content":
↪ "Tell me a story"}]}
```

6. モデルを使用する

1.AIDB を有効にする

```
CREATE EXTENSION aidb CASCADE;
```

2. モデルを登録する

```
SELECT aidb.create_model(
my_nim_llm,
nim_completions,
{"model": "meta/llama-3.3-70b-instruct", "url": "http://<ENDPOINT>:8000/v1/chat/
  ↳ completions"}::JSONB
);
```

3. モデルを実行します

モデルを操作するには、次のクエリを実行します。

```
SELECT aidb.decode_text(my_nim_llm, Tell me a short, one sentence story);
__OUTPUT__

          decode_text
-----
↳ -----
      As the clock struck midnight, a single tear fell from the porcelain dolls glassy
↳ eye.
```

出力は異なる場合があります。EDB AI アクセラレーターを介して NVIDIA NIM マイクロサービスを使用することに成功しました。

ソース

- NVIDIA NIM KServe 展開リソース <https://catalog.ngc.nvidia.com/orgs/nim/resources/llm-nim-kserve>
- NVIDIA NIM API リファレンス <https://docs.nvidia.com/nim/large-language-models/latest/api-reference.html>
- KServe ドキュメント <https://kserve.github.io/website/latest/model-serving/serving-runtimes/>
- EDB AI Factory の展開例 *Deploy NIM containers*
- モデルの例とタグ build.nvidia.com モデルカード

embedding-runtime.yaml

```
apiVersion: serving.kservice.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-snowflake-arctic-embed-1
```

(次のページに続く)

```
namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:
              name: nvidia-nim-secrets
              key: NGC_API_KEY
      image: upmdev.azurecr.io/nim/snowflake/arctic-embed-l:1.0.1
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
          cpu: "12"
          memory: 48Gi
        requests:
          cpu: "12"
          memory: 48Gi
      volumeMounts:
        - mountPath: /dev/shm
          name: dshm
  imagePullSecrets:
    - name: edb-cred
  protocolVersions:
    - v2
    - grpc-v2
  supportedModelFormats:
    - autoSelect: true
      name: nvidia-nim-snowflake-arctic-embed-l
      priority: 1
      version: "1.0.1"
  volumes:
    - emptyDir:
        medium: Memory
        sizeLimit: 40Gi
      name: dshm
```

embedding-service.yaml

```

apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: nim-snowflake-arctic-embed-l
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
    model:
      modelFormat:
        name: nvidia-nim-snowflake-arctic-embed-l
      resources:
        limits:
          nvidia.com/gpu: "1"
        requests:
          nvidia.com/gpu: "1"
      runtime: nvidia-nim-snowflake-arctic-embed-l

```

image-embed-runtime.yaml

```

apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-nvclip
  namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:

```

(次のページに続く)

(前のページからの続き)

```
      name: nvidia-nim-secrets
      key: NGC_API_KEY
    image: upmdev.azurecr.io/nim/nvidia/nvclip:latest
    name: kserve-container
    ports:
      - containerPort: 8000
        protocol: TCP
    resources:
      limits:
        cpu: "8"
        memory: 24Gi
      requests:
        cpu: "8"
        memory: 24Gi
    volumeMounts:
      - mountPath: /dev/shm
        name: dshm
  imagePullSecrets:
    - name: edb-cred
  protocolVersions:
    - v2
    - grpc-v2
  supportedModelFormats:
    - autoSelect: true
      name: nvidia-nim-nvclip
      priority: 1
      version: "latest"
  volumes:
    - emptyDir:
        medium: Memory
        sizeLimit: 20Gi
      name: dshm
```

image-embed-service.yaml

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: nim-nvclip
  namespace: default
```

(次のページに続く)

(前のページからの続き)

```
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
  model:
    modelFormat:
      name: nvidia-nim-nvclip
    resources:
      limits:
        nvidia.com/gpu: "1"
      requests:
        nvidia.com/gpu: "1"
    runtime: nvidia-nim-nvclip
```

llm-runtime.yaml

```
apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: meta-nim-llama-3.3-70b-instruct
  namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:
              name: nvidia-nim-secrets
              key: NGC_API_KEY
      image: upmdev.azurecr.io/nim/meta/llama-3.3-70b-instruct:1.8.4
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
```

(次のページに続く)

(前のページからの続き)

```

    cpu: "24"
    memory: 600Gi
  requests:
    cpu: "24"
    memory: 600Gi
  volumeMounts:
    - mountPath: /dev/shm
      name: dshm
  imagePullSecrets:
    - name: edb-cred
  protocolVersions:
    - v2
    - grpc-v2
  supportedModelFormats:
    - autoSelect: true
      name: meta-nim-llama-3.3-70b-instruct
      priority: 1
      version: "1.8.4"
  volumes:
    - emptyDir:
        medium: Memory
        sizeLimit: 60Gi
      name: dshm

```

llm-service.yaml

```

apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: meta-nim-llama-3-3-70b-instruct
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:

```

(次のページに続く)

(前のページからの続き)

```

- name: edb-cred
model:
  modelFormat:
    name: meta-nim-llama-3.3-70b-instruct
  resources:
    limits:
      nvidia.com/gpu: "4"
    requests:
      nvidia.com/gpu: "4"
  runtime: meta-nim-llama-3.3-70b-instruct

```

ocr-runtime.yaml

```

apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-baidu-paddleocr
  namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:
              name: nvidia-nim-secrets
              key: NGC_API_KEY
      image: upmdev.azurecr.io/nim/baidu/paddleocr:latest
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
          cpu: "8"
          memory: 64Gi
        requests:
          cpu: "8"
          memory: 64Gi
      volumeMounts:
        - mountPath: /dev/shm
          name: dshm
      imagePullSecrets:

```

(次のページに続く)

(前のページからの続き)

```
- name: edb-cred
protocolVersions:
  - v2
  - grpc-v2
supportedModelFormats:
  - autoSelect: true
    name: nvidia-nim-baidu-paddleocr
    priority: 1
    version: "latest"
volumes:
  - emptyDir:
      medium: Memory
      sizeLimit: 12Gi
    name: dshm
```

ocr-service.yaml

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: nim-baidu-paddleocr
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
    model:
      modelFormat:
        name: nvidia-nim-baidu-paddleocr
      resources:
        limits:
          nvidia.com/gpu: "1"
        requests:
          nvidia.com/gpu: "1"
      runtime: nvidia-nim-baidu-paddleocr
```

rerank-runtime.yaml

```

apiVersion: serving.kserve.io/v1alpha1
kind: ClusterServingRuntime
metadata:
  name: nvidia-nim-nv-rerankqa-llama
  namespace: default
spec:
  containers:
    - env:
        - name: NIM_CACHE_PATH
          value: /tmp
        - name: NGC_API_KEY
          valueFrom:
            secretKeyRef:
              name: nvidia-nim-secrets
              key: NGC_API_KEY
      image: upmdev.azurecr.io/nim/nvidia/llama-3.2-nv-rerankqa-1b-v2:latest
      name: kserve-container
      ports:
        - containerPort: 8000
          protocol: TCP
      resources:
        limits:
          cpu: "4"
          memory: 12Gi
        requests:
          cpu: "4"
          memory: 12Gi
      volumeMounts:
        - mountPath: /dev/shm
          name: dshm
  imagePullSecrets:
    - name: edb-cred
  protocolVersions:
    - v2
    - grpc-v2
  supportedModelFormats:
    - autoSelect: true
      name: nvidia-nim-nv-rerankqa-llama
      priority: 1
      version: "latest"
  volumes:
    - emptyDir:
        medium: Memory
        sizeLimit: 20Gi

```

(次のページに続く)

```
name: dshm
```

rerank-service.yaml

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  annotations:
    serving.kserve.io/enable-prometheus-scraping: "true"
    prometheus.kserve.io/port: "8000"
    prometheus.kserve.io/path: "/v1/metrics"
  name: nim-nv-rerankqa-llama-1
  namespace: default
spec:
  predictor:
    tolerations:
      - key: nvidia.com/gpu
        operator: Exists
        effect: NoSchedule
    nodeSelector:
      nvidia.com/gpu: "true"
    imagePullSecrets:
      - name: edb-cred
  model:
    modelFormat:
      name: nvidia-nim-nv-rerankqa-llama
    resources:
      limits:
        nvidia.com/gpu: "1"
      requests:
        nvidia.com/gpu: "1"
    runtime: nvidia-nim-nv-rerankqa-llama
```

Storage and access

Pipelines PGFS

Pipelines には、外部ストレージにアクセスできる Postgres File System PGFS 拡張機能があります。特に、パブリックまたはプライベートの S3 互換バケットまたはローカルファイルシステムにファイルを保存および取得できます。aidb 拡張機能は、PGFS を介してマッピングされた外部ストレージを **ストレージ場所** としてアドレス指定し、基になるストレージの詳細を抽象化します。

PGFS はパイプラインの主要コンポーネントであり、大きすぎるか、リレーショナルデータベースに適した方法で構造化されていないために、データベースに保存されていないデータを操作できます。

PGFS は、外部データラッパー FDW として実装され、拡張機能としてインストールされます。この拡張機

能は、データベースからファイルシステムと対話するための一連の機能を提供します。

PGFS ストレージ場所と AIDB ボリューム

AIDB で PGFS ストレージの場所を使用するには、AIDB に **volume** を作成する必要があります。

- **PGFS ストレージの場所**パスや資格情報などの必要な構成をすべて備えた外部ストレージプロバイダーを表します。
- **AIDB ボリューム**ストレージの場所を AIDB に接続します。同じ保存場所に複数のボリュームが可能です。ボリュームでは、宛先とオプションでサブパスのデータ形式を構成します。

PGFS 拡張機能のインストール

Pipelines インストールの一部として PGFS 拡張機能をインストールする方法については、Installing the AI Database extension を参照してください。

PGFS を使用する

PGFS 拡張機能で使用可能な機能の詳細については、*PGFS management functions* リファレンスを参照してください。これらの機能を使用すると、後で AIDB ボリュームを介して使用するストレージ場所を作成および管理できます。

PGFS functions for Pipelines

保存場所の作成

まずは、保存場所の作成から始めます。保存場所は、外部ファイルシステム内の場所への参照です。pgfs.create_storage_location ファンクションを使用して保存場所を作成できます。

```
select pgfs.create_storage_location(name=>storage_location_name,
                                   url=>protocol://path);
```

保存場所の追加オプションと資格情報を指定することもできます。

```
select pgfs.create_storage_location(name=>storage_location_name,
                                   url=>protocol://path,
                                   options => {},
                                   credentials => {});
```

そこで

- **name** は、保存場所の名前です。
- **url** は、保存場所の URL であり、S3 互換の保存場所またはローカルファイルシステムパスです。
- **options** は、保存場所の追加オプションを備えたオプションの JSON オブジェクトです。
- **credentials** は、必要に応じて、保存場所の資格情報を含むオプションの JSON オブジェクトです。

管理対象ストレージ場所 ID の 3 番目のパラメーターとして **uuid** を使用する pgfs.create_storage_location ファンクションもあることに注意してください。uuid はオプションであり、指定しない場合、null に設定されます。

ストレージプロバイダーのタイプ

サポートされているストレージプロバイダーの詳細な手順は、ここにあります。

- *Pipelines PGFS with S3-compatible storage*
- *Pipelines PGFS with local file storage*

オプションと資格情報を使用してストレージ場所を作成する

`options` および `credentials` パラメーターを使用すると、さまざまな設定を渡すことができます。

`options` パラメーターは、追加オプションを保存場所に渡すために使用できる JSON オブジェクトです。
`credentials` パラメーターは、資格情報を保存場所に渡すために使用できる JSON オブジェクトです。

`options` と `credentials` の違いは、オプションは拡張機能を照会するユーザーに可視されたままですが、資格情報はスーパーユーザーと保存場所を作成するユーザーを除くすべてのユーザーに非表示になることです。

保存場所のテストと AIDB での使用

`aidb` でストレージ場所を使用するには、ストレージ場所からボリュームを作成する必要があります。これを行うには、*AIDB volumes for accessing PGFS storage locations* を参照してください。

保存場所のリスト

`pgfs.list_storage_locations` ファンクションを使用して、すべての保存場所をリストできます。

```
select * from pgfs.list_storage_locations();
```

このコマンドは、現在定義されている保存場所のテーブルを結果ます。資格情報は、ユーザーが必要な権限を持っている場合にのみ表示されます。それ以外の場合、列は NULL です。

保存場所の取得

`pgfs.get_storage_location` ファンクションを使用して、特定の保存場所の詳細を取得できます。

```
select * from pgfs.get_storage_location(my_storage);
```

このコマンドは、`my_storage` という名前の保存場所の詳細を結果ます。

保存場所の更新

`pgfs.update_storage_location` ファンクションを使用して保存場所を更新できます。

```
select pgfs.update_storage_location(my_storage, s3://my_bucket, null, {"region": "eu-  
→west"})
```

保存場所の削除

`pgfs.delete_storage_location` ファンクションを使用して保存場所を削除できます。

```
select pgfs.delete_storage_location(my_storage);
```

このコマンドは、`my_storage` という名前の保存場所をデータベースから削除します。

Pipelines PGFS with Google Cloud Storage

概要 Google Cloud Storage

PGFS は `gs:` プレフィックスを使用して、Google Cloud Storage バケットを示します。

GCS を使用するための一般的な構文は次のとおりです。

```
select pgfs.create_storage_location(
    storage_location_name,
    gs://bucket_name,
    credentials => {}
);
```

資格情報

`credentials` 引数はオプション。JSON 形式の値を取得し、次の設定を提供します。

Option	Description
<code>google_application_credential</code>	アプリケーション資格情報ファイルへのパス
<code>google_service_account_key</code>	サービスアカウントキーファイルへのパス

サービスアカウントキーの管理方法の詳細については、[Google Cloud documentation](#) を参照してください。

同等の環境変数を介してこれらのオプションを設定して、Google Kubernetes Engine などの管理環境での認証を促進することもできます。

例 プライベート GCS バケット

```
SELECT pgfs.create_storage_location(edb_ai_example_images, gs://my-company-ai-images,
    credentials => {"google_service_account_key_file": "/var/run/gcs.json"}
);
```

例 GKE での認証

`GOOGLE_APPLICATION_CREDENTIALS` または `GOOGLE_SERVICE_ACCOUNT_KEY_FILE` 環境変数が PostgreSQL ポッドに設定されていることを確認します。次に、PGFS がそれらをピックアップします。

```
SELECT pgfs.create_storage_location(edb_ai_example_images, gs://my-company-ai-images);
```

Pipelines PGFS with local file storage

概要 ローカルファイルシステム

PGFS は `file:` プレフィックスを使用して、ローカルファイルシステムを示します。

ローカルファイルシステムを使用するための一般的な構文は次のとおりです。

```
select pgfs.create_storage_location(
  local_images,
  file:///var/lib/edb/pipelines/images/
);
```

注釈

Paths must always be absolute; e.g., start at the root `/` . Together with the protocol prefix `file://` , this means your path will have three slashes as in the example above.

注釈

Any local path that you want to access must be allowlisted in the *PGFS settings* .

注釈

Make sure to end your path with a `/` as in the examples above. This is required for path concatenation and validation with AIDB volumes.

Pipelines PGFS with S3-compatible storage

概要 S3 互換ストレージ

PGFS は `s3:` プレフィックスを使用して、S3 互換ストレージシステムを示します。

S3 を使用するための一般的な構文は次のとおりです。

```
select pgfs.create_storage_location(
  storage_location_name,
  s3://bucket_name,
  options => {},
  credentials => {}
);
```

JSON 形式設定の options 引数の設定

Option	Description
<i>bucket</i>	バケット名。URL で渡すことができない場合、バケット名を明示的に提供するために使用されます。
<i>skip_signature</i>	HMAC 認証を無効にします。資格情報で <code>access_key_id/secret_access_key</code> を提供しない場合、これを「true」に設定します。
<i>region</i>	S3 互換ストレージシステムのリージョン。リージョンが指定されない場合、クライアントは自動検出を試行します。
<i>endpoint</i>	S3 互換ストレージシステムのエンドポイント。
<i>allow_http</i>	エンドポイントが HTTPS / TLS ではなくプレーン HTTP を使用するかどうか。エンドポイントが <code>http://</code> で始まる場合、これを「true」に設定します。

JSON 形式の credentials 引数の設定

Option	Description
<i>access_key_id</i>	HMAC 資格情報多くの場合、非 AWS S3 プロバイダーの「ユーザー名」
<i>secret_access_key</i>	HMAC 資格情報多くの場合、非 AWS S3 プロバイダーの「パスワード」
<i>session_token</i>	HMAC 資格情報の代わりに使用できるセッショントークン

例 AWS S3 パブリックバケット

この例では、AWS S3 のパブリックバケットを使用します。パブリックバケットは認証を必要としません。

```
SELECT pgfs.create_storage_location(edb_ai_example_images, s3://public-ai-team,
                                options => {"region": "eu-central-1", "skip_
->signature": "true"}
);
```

例 AWS S3 プライベートバケット

この例では、AWS S3 のプライベートバケットを使用します。プライベートバケットには認証が必要です。この例では、HMAC 資格情報を使用します。

```
SELECT pgfs.create_storage_location(internal_ai_project, s3://my-company-ai-images,
                                options => {"region": "eu-central-1"},
                                credentials => {"access_key_id": "secret",
->"secret_access_key": "secret!"}
);
```


例 HTTPS と非 AWS S3 / S3 互換

この例では、minIO のような S3 互換システムを使用します。この場合、**endpoint** を提供する必要があります。AWS S3 を使用する場合にのみ省略できます。

```
SELECT pgfs.create_storage_location(ai_images_local_minio, s3://my-ai-images,
                                options => {"endpoint": "https://minio-api.apps.
→local"},
                                credentials => {"access_key_id": "my_username",
→"secret_access_key": "my_password"}
);
```

例 非 AWS S3 / S3-HTTP と互換性

この例では、minIO のような S3 互換システムを使用します。この場合、**endpoint** を提供する必要があります。AWS S3 を使用する場合にのみ省略できます。

この場合、サーバーは TLS 暗号化を使用しないため、コードはプレーンな HTTP 接続を構成します。

```
SELECT pgfs.create_storage_location(ai_images_local_minio,
                                s3://my-ai-images,
                                options => {"endpoint": "http://minio-api.apps.
→local", "allow_http": "true"},
                                credentials => {"access_key_id": "my_username",
→"secret_access_key": "my_password"}
);
```

PGFS settings

PGFS 設定について

PGFS 拡張機能には、拡張機能がデータにアクセスする方法を制御する設定があります。これらの設定は GUC グランドユニファイド構成であり、Postgres SET コマンドを使用して設定できます。

PGFS 設定

pgfs.allowed_local_fs_paths

デフォルトでは、PGFS はローカルファイルシステムの `/tmp/pgfs` のファイルにのみアクセスできます。これは、拡張機能がアクセスすべきでないファイルにアクセスしないようにするセキュリティ対策です。

PGFS が他の場所にあるファイルにアクセスできるようにする場合は、これらの場所を GUC 設定 `pgfs.allowed_local_fs_paths` に追加する必要があります。これは、PGFS がアクセスを許可されるパスのクォンタリファイドリストです。

<code>pgfs.allowed_local_fs_paths</code>	Description
value	
<code>'/tmp/pgfs'</code>	PGFS がファイルにアクセスするためのデフォルトの場所。
<code>'/tmp/pgfs:/tmp/other'</code>	PGFS は、 <code>/tmp/pgfs</code> と <code>/tmp/other</code> の両方のファイルにアクセスできます。
<code>'/'</code>	PGFS は、システム上のすべてのファイルにアクセスできます。これはお勧めしません。
<code>''</code>	PGFS はローカルファイルにアクセスできません。

Postgres SET コマンドを使用して設定を変更します。たとえば、PGFS が `/tmp/pgfs` および `/tmp/other` のファイルにアクセスできるようにするには、次のコマンドを実行します。

```
SET pgfs.allowed_local_fs_paths = /tmp/pgfs:/tmp/other;
```

AIDB volumes for accessing PGFS storage locations

概要

AIDB のボリュームを使用すると、postgres テーブルの代わりに外部ストレージをパイプラインのデータソースとして使用できます。

PDF や画像のような非構造化データは、多くの場合 AWS S3 のようなオブジェクトストアに保存されます。AIDB ボリュームを使用すると、このようなデータを AI パイプラインに統合できます。

PGFS ストレージ場所と AIDB ボリューム

AIDB ボリュームにより、PGFS ストレージ場所へのアクセスが許可されます。これは、PGFS の保存場所を最初に作成する必要があることを意味します。 [Pipelines PGFS reference](#) を参照してください。

- **PGFS ストレージの場所** パスや資格情報などの必要な構成をすべて備えた外部ストレージプロバイダーを表します。
- **AIDB ボリューム** 保存場所を AIDB に接続します。同じ保存場所に複数のボリュームが可能です。ボリュームでは、宛先とオプションでサブパスのデータ形式を構成します。

ボリュームの作成

新しいボリュームに名前を割り当て、それがポイントする PGFS ストレージ場所を指定します。

3 番目の引数は、保存場所のオプションのサブパスです。これにより、同じストレージ場所に複数のボリュームを作成し、バケット内の異なるパスをポイントできます。

最後の引数はデータフォーマットであり、サポートされているタイプは `aidb.PipelineDataFormat` です。これは、パイプラインがボリューム内のオブジェクトを処理する方法を決定するために使用されるメタデータです。データ形式はフィルタではありません。ボリュームには、指定されたフォーマットのオブジェクトのみが含まれていることを確認する必要があります。

```
SELECT aidb.create_volume(volume_name, pgfs_storage_location_name, /, Image);
```

例

Pipelines PGFS reference の詳細については、こちらをご覧ください。

テキストドキュメントを使用してパブリック S3 バケットにアクセスする

```
SELECT pgfs.create_storage_location(
    aidb_rag_bucket, s3://aidb-rag-app,
    options => {"region": "eu-central-1", "skip_signature": "true"});

SELECT aidb.create_volume(aidb_rag_bucket_volume, aidb_rag_bucket, /, Text);

SELECT * FROM aidb.list_volume_content(aidb_rag_bucket_volume);
```

画像を含むローカルディレクトリへのアクセス OCR プリペアラーパーパイプラインなど。ボリュームはディレクトリ/tmp/pgfs/ocr_input/ をポイントします。

```
CREATE EXTENSION IF NOT EXISTS pgfs;
SET pgfs.allowed_local_fs_paths = /tmp/pgfs;

SELECT pgfs.create_storage_location(local_tmp_pgfs, file:///tmp/pgfs/);

SELECT aidb.create_volume(ocr_input_volume, local_tmp_pgfs, ocr_input/, Image);

SELECT * FROM aidb.list_volume_content(ocr_input_volume);
```

その他の管理機能

次の管理機能を使用して、ボリュームをリストおよび削除できます。

```
SELECT aidb.list_volumes();
```

```
SELECT aidb.delete_volume(volume_name);
```

注 PGFS ストレージ場所を削除すると、その上に作成されたすべてのボリュームが削除されます。

ボリュームへのアクセス

ボリュームには2つの異なる方法でアクセスできます。

1. パイプライン内でデータソースとして使用すること
2. 直接アクセス SQL ファンクションを使用する場合

直接アクセス SQL ファンクションは、パイプラインで構成する前に、新しく作成したストレージ場所とボリュームをテストするのに役立ちます。これらは、外部ストレージで動作するカスタム SQL クエリを構築するために使用することもできます。

リストコンテンツ

このコマンドは、ボリュームに存在するすべてのオブジェクトをリストしたテーブルを結果ます。

```
SELECT * FROM aidb.list_volume_content(volume_name);
```

オブジェクトの読み取り

このコマンドは、オブジェクトの内容を読み取り、BYTEA タイプとして結果ます。

```
SELECT aidb.read_volume_file(volume_name, object_name.txt);
```

例として、オブジェクトにプレーンテキストが含まれる場合、次のように変換できます。

```
SELECT convert_from(aidb.read_volume_file(my_text_files, hello_world.txt), utf8);
```

Pipelines reference

モデル

テーブル

- aidb.model_providers
- aidb.models

モデルファンクション

- aidb.create_model
- aidb.get_model
- aidb.delete_model
- aidb.list_hcp_models
- aidb.create_hcp_model
- aidb.sync_hcp_models
- aidb.encode_text
- aidb.encode_text_batch
- aidb.decode_text
- aidb.decode_text_batch
- aidb.encode_image
- aidb.rerank_text

準備者

Prepare ビュー

- *aidb.preps*
- *aidb.preparers*

プリペアラースの種類

- *aidb.DataPreparationOperation*

プリペアラースファンクション

- *aidb.create_table_preparer*
- :ref:`aidb.create\_preparer\_for\_table` (deprecated) <*aidb.create_preparer_for_table* (deprecated)>` 非推奨
- *aidb.create_volume_preparer*
- :ref:`aidb.create\_preparer\_for\_volume` (deprecated) <*aidb.create_preparer_for_volume* (deprecated)>` 非推奨
- *aidb.bulk_data_preparation*
- *aidb.set_auto_preparer*
- :ref:`aidb.set\_preparer\_auto\_processing` (deprecated) <*aidb.set_preparer_auto_processing* (deprecated)>` 非推奨
- *aidb.delete_preparer*

ナレッジベース

ナレッジベーステーブル

- *aidb.kbs*
- *aidb.knowledge_bases*

ナレッジベース機能

- *aidb.create_table_knowledge_base*
- *aidb.create_volume_knowledge_base*
- *aidb.set_auto_knowledge_base*
- *aidb.bulk_embedding*
- *aidb.retrieve_key*
- *aidb.retrieve_text*
- :ref:`aidb.delete\_knowledge\_base` <*aidb.delete_knowledge_base*>`
- *aidb.create_volume_knowledge_base*

- `aidb.delete_volume`

PGFS

PGFS ファンクション

- `pgfs.create_foreign_table`
- `pgfs.create_storage_location`
- `pgfs.create_storage_location_with_foreign_table`
- `pgfs.delete_storage_location`
- `pgfs.list_storage_locations`
- `pgfs.get_storage_location`
- `pgfs.update_storage_location`
- `pgfs.set_default_storage_location`
- `pgfs.get_default_storage_location`

レトリバー非推奨

レトリバー機能非推奨

- `:ref:`aidb.create_retriever_for_table` (deprecated) <aidb.create_retriever_for_table (deprecated)>` 非推奨
- `:ref:`aidb.create_retriever_for_volume` (deprecated) <aidb.create_retriever_for_volume (deprecated)>` 非推奨
- `:ref:`aidb.set_retriever_auto_processing` (deprecated) <aidb.set_retriever_auto_processing (deprecated)>` 非推奨
- `:ref:`aidb.delete_retriever` (deprecated) <aidb.delete_retriever (deprecated)>` 非推奨

Pipelines models reference

EDB Postgres AI-AI アクセラレーターパイプラインモデルのこのリファレンスドキュメントには、[Pipelines models reference](#) で作業するための `aidb` 拡張機能で使用可能なファンクションとビューに関する情報が含まれています。

テーブル

`helpb.model_providers`

`aidb.model_providers` テーブルには、使用可能なモデルプロバイダーに関する情報が保存されます。

Column	Type	Description
<code>server_name</code>	name	モデルサーバーの名前
<code>server_description</code>	text	モデルサーバーの説明

次のページに続く

表 28 – 前のページからの続き

Column	Type	Description
<i>server_options</i>	text[]	モデルサーバーのオプション

helpb.models

レジストリ内のすべてのモデルとその構成済みオプションのリストを返します。事前定義されたモデルやユーザー作成モデルを含みます。

Column	Type	Description
<i>name</i>	text	モデルのユーザー定義の名前
<i>provider</i>	text	モデルプロバイダーの名前
<i>options</i>	jsonb	モデルプロバイダーのオプションの構成

例

```
SELECT * FROM aidb.models;
__OUTPUT__
name | provider | options
-----+-----+-----
bert | bert_local | {"config={}"}
```

ファンクション

promoteb.create_model

名前、プロバイダー、およびオプションの構成を保存することにより、システムに新しいモデルを作成します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	text		モデルのユーザー定義の名前。
<i>provider</i>	text		モデルプロバイダーの名前 helpb.model_providers にあります。
<i>config</i>	jsonb	'{}':jsonb	モデルプロバイダーのオプションの構成。 <i>model</i> 、 <i>url</i> 、TLS オプション <code>`tls_config`</code> などのモデル固有のパラメーターが含まれる場合があります。

次のページに続く

表 30 – 前のページからの続き

Parameter	Type	Default	Description
<i>credentials</i>	jsonb	'{}':jsonb	モデルプロバイダーのオプションの資格情報。
<i>replace_credentials</i>	boolean	false	true の場合、モデルプロバイダーの資格情報を置き換えます。false の場合、資格情報は上書きされません。

例

```
SELECT aidb.create_model(
    name => my_t5::text,
    provider => t5_local::character varying,
    config => {"param1": "value1", "param2": "value2"}::jsonb,
    credentials => {"token": "abcd"}::jsonb
);
```

または同等のデフォルト値を使用して

```
SELECT aidb.create_model(my_t5, t5_local);
```

または、既に作成されたモデルのプロバイダーの資格情報を更新する場合。

```
SELECT aidb.create_model(
    name => my_t5::text,
    provider => t5_local::character varying,
    config => {"param1": "value1", "param2": "value2"}::jsonb,
    credentials => {"token": "abcd"}::jsonb,
    replace_credentials => true
);
```

TLS 構成オプション

HTTPS ベースのモデルエンドポイントに安全に接続するために、config オブジェクトに `tls_config` フィールドを含めることができます。

```
"tls_config": {
  "insecure_skip_verify": true, // (optional) disables certificate validation
  "ca_path": "/etc/aidb/myCA.pem" // (optional) path to a trusted CA certificate
}
```


promoteb.get_model

レジストリ内のモデルの構成を結果ます。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		モデルの名前

返品

Column	Type	Description
<i>name</i>	text	モデルのユーザー定義の名前
<i>provider</i>	text	モデルプロバイダーの名前
<i>options</i>	jsonb	モデルプロバイダーのオプションの構成

例

```
SELECT * FROM aidb.get_model(t5);
__OUTPUT__
name | provider | options
-----+-----+-----
t5   | t5_local | {"config={}"}
```

(1 row)

promoteb.delete_model

レジストリからモデルを削除します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		モデルの名前

例

```
SELECT aidb.delete_model(t5);
__OUTPUT__
delete_model
-----
(t5,t5_local,"{"config={}"})
```

(1 row)

返品

Column	Type	Description
<i>delete_model</i>	jsonb	削除されたモデルの名前、プロバイダー、およびオプション

promoteb.list_hcp_models

ハイブリッドマネージャーで実行されているモデルを取得します。

返品

Column	Type	Description
<i>name</i>	text	HCP で実行されているモデルインスタンスの名前
<i>url</i>	text	HCP で実行されているモデルの API URL
<i>model</i>	text	HCP で実行されているモデルの名前

例

```
SELECT * FROM aidb.get_hcp_models();
```

name	url
llama-3-1-8b-instruct-1xgpu-g6	http://llama-3-1-8b-instruct-1xgpu-g6-predictor.default.svc.cluster.local
llama-3-2-nv-embedqa-1b-v2	http://llama-3-2-nv-embedqa-1b-v2-predictor.default.svc.cluster.local
meta-nim-llama3-70b-instruct-8xgpu-g5	http://meta-nim-llama3-70b-instruct-8xgpu-g5-predictor.default.svc.cluster.local

(3 rows)

promoteb.create_hcp_model

HCP で実行中のインスタンスを参照することにより、システムに新しいモデルを作成します

パラメーター

Parameter	Type	Default	Description
<i>name</i>	text		モデルのユーザー定義名

次のページに続く

表 36 – 前のページからの続き

Parameter	Type	Default	Description
<i>hcp_model_name</i>	text		HCP で実行されているモデルインスタンスの名前

promoteb.sync_hcp_models

HCP でモデルを作成し、`is_hcp_model=true` を設定します。HCP で見つからない場合、その設定を持つモデルを削除します。

返品

Column	Type	Description
<i>status</i>	text	同期ステータス。 <i>deleted</i> 、 <i>unchanged</i> 、 <i>created</i> 、または <i>skipped</i> のいずれか。
<i>model</i>	text	同期 HCP モデルの名前。

helpb.encode_text

モデルを使用してテキストをエンコードし、指定されたテキスト入力のベクトル表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		エンコードするモデルの名前
<i>text</i>	text		エンコードするテキスト

helpb.encode_text_batch

モデルを使用してテキストのバッチをエンコードし、指定されたテキスト入力のベクトル表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		エンコードするモデルの名前
<i>text</i>	text[]		エンコードするテキストの配列

promoteb.decode_text

モデルを使用してテキストをデコードし、指定されたテキスト入力のベクトル表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		デコードするモデルの名前
<i>text</i>	text		デコードするテキスト

返品

Column	Type	Description
<i>decode_text</i>	text	デコードされたテキスト

promoteb.decode_text_batch

モデルを使用してテキストのバッチをデコードし、指定されたテキスト入力の表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		デコードするモデルの名前
<i>text</i>	text[]		デコードするテキストの配列

返品

Column	Type	Description
<i>decode_text</i>	text	デコードされたテキスト

promoteb.encode_image

モデルを使用して画像をエンコードし、指定された画像入力のベクトル表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		エンコードするモデルの名前
<i>image</i>	bytea		エンコードする画像

返品

Column	Type	Description
<i>encode_image</i>	bytea	エンコードされた画像

helpb.rerank_text

モデルを使用してテキストを再ランク付けし、指定されたテキスト入力のベクトル表現を生成します。

パラメーター

Parameter	Type	Default	Description
<i>model_name</i>	text		リランキングするモデルの名前
<i>query</i>	text		入力がランク付けされるクエリ
<i>input</i>	text[]	[]	ランク付けの入力

返品

Column	Type	Description
<i>text</i>	text	"input"からのテキスト
<i>logit_score</i>	double precision	このテキストのスコア/ランク
<i>id</i>	int	テキストが入力配列に持っていたインデックス

Pipelines reference - Preparers

パイプライン準備者のためのこのリファレンスドキュメントには、[Pipelines reference - Preparers](#)に関連する `aidb` 拡張機能で使用可能な関数とビューに関する情報が含まれています。詳細については、[Pipelines knowledge bases usage](#) および [Where to find SDK references and examples?](#) を参照してください。

ビュー

helpb.preparers

`aidb.preps` としても参照可能な `aidb.preparers` ビューには、構成されたデータプリペアラに関する情報が含まれています。これには、次の列が含まれます。

Column	Type	Description
<i>id</i>	INTEGER	NA
<i>name</i>	TEXT	準備者の名前。
<i>operation</i>	helpb.DataPreparationOperati	実行する処理ステップの種類。

次のページに続く

表 48 – 前のページからの続き

Column	Type	Description
destination_schema	TEXT	出力データを保存する宛先テーブルのスキーマ。
destination_table	TEXT	出力データを保存する宛先テーブルの名前。
destination_key_column	TEXT	ソースデータのキーを参照する宛先テーブルの列。
destination_data_column	TEXT	出力データを保存する宛先テーブルの列。
options	JSONB	データ準備操作の構成オプション。 <i>**vector-native capabilities**</i> と同じ API を使用します。
source_type	TEXT	準備者が作業しているソースデータのタイプ。 'テーブル' または 'ボリューム' のいずれかです。
source_schema	TEXT	準備者が処理するソースデータを含むテーブルのスキーマ。 'Table' ソースタイプのプリペアラーにのみ適用されます。
source_table	TEXT	プリペアラーが処理するソースデータを含むテーブルの名前。 'Table' ソースタイプのプリペアラーにのみ適用されます。
source_data_column	TEXT	準備者が処理するソースデータを含むソーステーブルの列。 'Table' ソースタイプのプリペアラーにのみ適用されます。
source_key_column	TEXT	出力処理されたデータで参照するソーステーブルのキー列の名前。 'Table' ソースタイプのプリペアラーにのみ適用されます。
source_volume_name	TEXT	データソースとして使用するボリュームの名前。 'Volume' ソースタイプのプリペアラーにのみ適用されます。

タイプ

helpb.DataPreparationOperation

aidb.DataPreparationOperation タイプは、実行可能な前処理ステップのさまざまなタイプを表す列挙型です。

- ChunkText
- テキストの要約
- ParseHtml
- ParsePdf
- PerformOcr

ファンクション

helpb.create_table_preparer

ソースデータテーブルを使用してプリペアラーを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	作成者の名前
operation	helpb.DataPreparatic	Required	データ準備操作の種類
source_table	TEXT	Required	ソースデータテーブルの名前
source_data_column	TEXT	Required	生データを含むソーステーブルの列
destination_table	TEXT	Required	宛先テーブルの名前
destination_data_column	TEXT	Required	処理されたデータの宛先テーブルの列
source_key_column	TEXT	'id'	行を参照するキーとして使用するソーステーブル内の一意の列。
destination_key_column	TEXT	'id'	<code>`source_key_column`</code> を参照する宛先テーブルのキー列
options	JSONB	'{}':JSONB	データ準備操作の構成オプション。 <i>**vector-native capabilities**</i> と同じ API を使用します。

aidb.create_preparer_for_table 非推奨

同じ引数を使用した** aidb.create_table_preparer に置き換えられます。

promoteb.create_volume_preparer

指定された PGFS ボリュームのプリペアラーを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	作成者の名前
operation	helpb.DataPreparatic	Required	データ準備操作の種類
source_volume_name	TEXT	Required	生データが含まれるソースボリュームの名前
destination_table	TEXT	Required	宛先テーブルの名前
destination_data_column	TEXT	Required	処理されたデータの宛先テーブルの列

次のページに続く

表 50 – 前のページからの続き

Parameter	Type	Default	Description
destination_key_column	TEXT	'id'	処理されたデータを一意に識別する宛先テーブルのキー列
options	JSONB	'{}'::JSONB	データ準備操作の構成オプション。 <i>**vector-native capabilities**</i> と同じ API を使用します。
invoker_role	TEXT	NULL	テーブル、パイプライン、およびバックグラウンドジョブ実行を所有するロール。

aidb.create_preparer_for_volume 非推奨

同じ引数を使用した** aidb.create_volume_preparer に置き換えられます。

promoteb.bulk_data_preparation

指定されたプリペアラのソースからすべてのデータに対して、構成されたデータ準備オペレーションを実行します。

パラメーター

Parameter	Type	Description
preparer_name	TEXT	作成者の名前

promoteb.set_auto_preparer

このプリペアラの自動処理モードを設定します。このファンクションは、自動データ準備を有効または無効にするために使用されます。Live モードは Postgres トリガーベースのオートメーションを有効にし、Disabled はすべてのオートメーションを無効にします。

パラメーター

Parameter	Type	Default	Description
preparer_name	TEXT		作成者の名前
mode	<i>helpb.PipelineAutoPr</i> モード		目的の処理モード

例

```
SELECT aidb.set_auto_preparer(test_preparer, Live);
SELECT aidb.set_auto_preparer(test_preparer, Disabled);
```


aidb.set_preparer_auto_processing 非推奨

同じ引数を使用した** aidb.set_auto_preparer に置き換えられます。

promoteb.delete_preparer

プリペアラの構成を削除します。

パラメーター

Parameter	Type	Description
preparer_name	TEXT	削除する準備者の名前

i 注釈

This function doesn't delete the destination table or any data in it.

Pipelines knowledge bases reference

Pipelines ナレッジベースのこのリファレンスドキュメントには、Knowledge Bases に関連する aidb 拡張機能で使用可能なファンクションとビューに関する情報が含まれています。

ビュー

helpb.Knowledge_bases

aidb.kbs としても参照可能な aidb.knowledge_bases ビューは、データベース内のすべてのナレッジベースのリストを提供します。これには、ナレッジベース名、使用されるモデル、およびソースデータ型に関する情報が含まれています。

Column	Type	Description
id	integer	NA
name	text	ナレッジベースの名前。
vector_schema	text	Vector_table のスキーマ。
vector_table	text	エンベディングが保存されるテーブルの名前。存在しない場合は新しく作成されます。aidb が管理しています。
vector_key_column	text	埋め込みを計算するときにソースデータのキーを参照するキーを格納するために使用する列。デフォルトを使用し、aidb にこのテーブルの管理を任せることをお勧めします。
vector_data_column	text	埋め込みを保存する列。デフォルトを使用し、aidb にこのテーブルを管理させることをお勧めします。

次のページに続く

表 54 – 前のページからの続き

Column	Type	Description
model_name	text	埋め込み計算と取得に使用するモデルの名前。
topk	integer	取得中に返される結果のデフォルトの数。SQL の LIMIT に似ています。
distance_operator	<i>helpb.DistanceOperator</i>	取得中に、ベクトルを比較するために使用されるベクトル操作。
options	jsonb	未使用。
auto_processing	<i>helpb.PipelineAutoProcessing</i> モード	自動処理モード。
owner_role	text	このパイプラインを作成した Postgres ロール。バックグラウンド自動処理は、このロールとしてパイプラインを実行します。
batch_size	integer	このパイプラインの実行時に同時に処理するレコードの数。
back-ground_sync_interval	interval	バックグラウンドの自動処理に使用されます。これは、パイプラインの実行間の間隔です。
source_type	text	このパイプラインがデータソースとしてテーブルまたはボリュームを使用するかどうかを示します。
source_schema	text	source_table のスキーマ。
source_table	text	パイプラインの入力として使用されるテーブルの名前。ナレッジベースがソースとしてボリュームを使用する場合、使用されません。
source_data_column	text	Pipelines がエンベディングを計算するソーステーブルの列名。これは、取得操作で返される列でもあります。
source_data_format	<i>helpb.PipelineDataFormat</i>	ナレッジベースが処理しているデータのフォーマット。 <code>:ref:``aidb.PipelineDataFormat`<#aidbpipelinedataformat>`</code> 型を使用します。
source_key_column	text	ベクタテーブルにエンベディングを保存するためのキーとして使用する列。これは、エンベディングからソースデータへの参照を提供します。
source_volume_name	text	データソースとして使用するボリュームの名前。 <code>aidb.create_volume_knowledge_base()</code> で構成されたナレッジベースにのみ適用されます。

helpb.knowledge_base_stats

`aidb.kbstat` としても参照可能な `aidb.knowledge_base_stats` ビューは、ナレッジベースパイプラインの自動処理に関する現在の統計を提供します。

Column	Type	Description
knowledge base	text	ナレッジベースの名前。

次のページに続く

表 55 – 前のページからの続き

Column	Type	Description
auto processing	<i>helpb.PipelineAutoProcessing</i> モード	自動処理モード。
table: unprocessed rows	bigint	テーブルナレッジベースの場合 最後に実行されてから新しいまたは変更された可能性のある未処理の行の数。
volume: scans completed	bigint	ボリュームナレッジベースの場合 これまでに実行されたソースボリュームの完全なリストの数。
count(source records)	bigint	ナレッジベースのソースのレコードの数。ボリュームナレッジベースの場合、これは最後のフルスキャン中に見られたレコードの数です。
count(embeddings)	bigint	宛先テーブルに保存されているエンベディングの数。

タイプ

helpb.DistanceOperator

`aidb.DistanceOperator` 型は、取得中に使用できる距離演算子を表す列挙型です。

Value	Description
L2	ユークリッド距離
Inner	内積
Cosine	コサイン類似度
L1	L1 距離
Hamming	ハミング距離
Jaccard	Jaccard 距離

SQL 定義

```
CREATE TYPE DistanceOperator AS ENUM (
  L2,
  InnerProduct,
  Cosine,
  L1,
  Hamming,
  Jaccard
);
```

helpb.PipelineDataFormat

`aidb.PipelineDataFormat` 型は、ソースデータとして使用できるデータ形式を表す列挙型です。

Value	Description
Text	テキストデータ
Image	画像データ
Pdf	PDF データ

SQL 定義

```
CREATE TYPE PipelineDataFormat AS ENUM (
    Text,
    Image,
    Pdf
);
```

helpb.PipelineAutoProcessing モード

aidb.PipelineAutoProcessingMode タイプは、パイプラインの自動処理がどのように動作するか定義するために使用される列挙型ですナレッジベースや準備者など。

Value	Description
Live	新しいデータは追加中にすぐに処理されます Postgres トリガーを使用
Background	バックグラウンドでの継続処理 Postgres バックグラウンドワーカーを使用
Disabled	自動処理なし

```
CREATE TYPE PipelineAutoProcessingMode AS ENUM (
    Live,
    Background,
    Disabled
);
```

ファンクション

promoteb.create_table_knowledge_base

指定されたテーブルのナレッジベースを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	ナレッジベースの名前。
model_name	TEXT	Required	使用するモデルの名前。
source_table	regclass	Required	ソースとして使用するテーブルの名前。

次のページに続く

表 59 – 前のページからの続き

Parameter	Type	Default	Description
source_data_column	TEXT	Required	使用するソーステーブルの列名。
source_data_format	helpb.PipelineDataFormat	Required	その列のデータの形式"Text"、"Image"、"PDF"。
source_key_column	TEXT	'id'	行を参照するキーとして使用するソーステーブル内の一意の列。
vector_table	TEXT	NULL	NA
vector_data_column	TEXT	'embeddings'	NA
vector_key_column	TEXT	'id'	NA
topk	INTEGER	1	NA
distance_operator	helpb.DistanceOperator	'L2'	NA
options	JSONB	'{}'::JSONB	未使用
auto_processing	helpb.PipelineAutoProcessingMode	'Disabled'	このパイプラインの自動処理を構成します。
index_type	TEXT	'vector'	ベクトルテーブルに使用するインデックスのタイプ。
batch_size	int	100	同時に処理するレコードの数。
background_sync_interval	interval	'30 seconds'	バックグラウンド自動処理が構成されている場合のパイプライン実行間の間隔。
invoker_role	TEXT	NULL	テーブル、パイプライン、およびバックグラウンドジョブ実行を所有するロール。

distance_operator	index_type
L2	<i>vector_l2_ops</i>
InnerProduct	<i>vector_ip_ops</i>
Cosine	<i>vector_cosine_ops</i>
L1	<i>vector_l1_ops</i>

- `index_type` が `ivfflat` に設定されている場合、システムはベクターテーブルに `IVFFlat` インデックスを作成します。
- `index_type` が `disabled` に設定されている場合、インデックスは作成されません。

```
SELECT aidb.create_table_knowledge_base(
    name => test_knowledge_base,
    model_name => bert,
    source_table => test_source_table,
    source_data_column => content,
    source_key_column => doc_id,
    source_data_format => Text
);
```

agentb.create_volume_knowledge_base

指定された PGFS ボリュームのナレッジベースを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	ナレッジベースの名前
model_name	TEXT	Required	モデルの名前
source_volume_name	TEXT	Required	ボリュームの名前
vector_table	TEXT	NULL	ベクタテーブルの名前
vector_data_column	TEXT	'embeddings'	ベクター列の名前
vector_key_column	TEXT	'id'	キーカラムの名前
topk	INTEGER	1	返される結果の数
distance_operator	aidb.distanceoperator	'L2'	距離演算子
options	JSONB	'{}':JSONB	オプション
auto_processing	helpb.PipelineAutoPr モード	'Disabled'	このパイプラインの自動処理を構成します。
index_type	TEXT	'vector'	ベクトルテーブルに使用するインデックスのタイプ。
batch_size	int	100	同時に処理するレコードの数。
back-ground_sync_interval	interval	'30 seconds'	バックグラウンド自動処理が構成されている場合のパイプライン実行間の間隔。
invoker_role	TEXT	NULL	テーブル、パイプライン、およびバックグラウンドジョブ実行を所有するロール。

distance_operator	index_type
L2	<i>vector_l2_ops</i>
InnerProduct	<i>vector_ip_ops</i>
Cosine	<i>vector_cosine_ops</i>
L1	<i>vector_l1_ops</i>

- `index_type` が `ivfflat` に設定されている場合、システムはベクターテーブルに **IVFFlat** インデックスを作成します。
- `index_type` が `disabled` に設定されている場合、インデックスは作成されません。

```
SELECT aidb.create_volume_knowledge_base(
    name => demo_vol_knowledge_base,
    model_name => simple_model,
    source_volume_name => demo_bucket_vol
);
```

promoteb.set_auto_knowledge_base

このナレッジベースの自動処理モードを設定します。

パラメーター

Parameter	Type	Default	Description
knowledge_base_name	TEXT		自動処理を有効にするナレッジベースの名前
mode	<i>helpb.PipelineAutoPr</i> モード		目的の自動処理モード
batch_size	INTEGER DE- FAULT NULL		無効別名手動またはバックグラウンド処理で1つのバッチで処理するレコードの数
back-ground_sync_interval	INTERVAL DE- FAULT NULL		バックグラウンド処理の目的の同期間隔

例

```
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Live);
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Disabled, batch_size => 100);
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Background);
SELECT aidb.set_auto_knowledge_base(test_knowledge_base, Background, batch_size => 100, background_sync_interval => 10 seconds);
```

helpb.bulk_embedding

このパイプラインのソースレコードのエンベディングを計算し、ベクター宛先に保存します。

この関クションの動作は、構成された auto_processing モードによって異なります。

- **ライブおよび無効な自動処理**すべてのソースレコードを直接処理します。
- **バックグラウンド自動処理**処理するすべてのソースレコードをマークしますが、操作は実行しません。バックグラウンドワーカーが実行します。

パラメーター

Parameter	Type	Default	Description
knowledge_base_name	TEXT		エンベディングを生成するナレッジベースの名前。
silent	BOOL	false	印刷ステータスと進行状況ログを無効にします。

例

```

aidb=# SELECT aidb.bulk_embedding(kb_volume_image_manual);
__OUTPUT__
INFO: kb_volume_image_manual: (re)setting state table to process all data...
INFO: kb_volume_image_manual: Starting... Batch size 10, count(known source
↳records): 0, scans completed: 0, count(embeddings): 0
INFO: kb_volume_image_manual: Batch iteration finished, count(known source records):
↳10, scans completed: 0, count(embeddings): 10
[...]
INFO: kb_volume_image_manual: Batch iteration finished, count(known source records):
↳177, scans completed: 0, count(embeddings): 177
INFO: kb_volume_image_manual: finished, count(known source records): 177, scans
↳completed: 1, count(embeddings): 177

```

promoteb.retrieve_key

ソースデータを検索せずに、一致する埋め込みからキーを取得します。

パラメーター

Parameter	Type	Default	Description
knowledge_base_name	TEXT		取得に使用するナレッジベースの名前
query_string	TEXT		取得に使用するクエリ文字列
number_of_results	INTEGER	0	返される結果の数

例

```

SELECT * FROM aidb.retrieve_key(test_knowledge_base, shoes, 2);
__OUTPUT__
key | distance
-----+-----
43941 | 0.2938963414490189
19337 | 0.3023805122617119
(2 rows)

```

helpb.retrieve_text

埋め込みをソーステーブルに結合することにより、一致する埋め込みからソーステキストデータを取得します。

パラメーター

Parameter	Type	Default	Description
knowledge_base_name	TEXT		取得に使用するナレッジベースの名前
query_string	TEXT		取得に使用するクエリ文字列
number_of_results	INTEGER	0	返される結果の数

返品

Column	Type	Description
<i>key</i>	text	取得したデータのキー
<i>value</i>	text	取得したデータの値
<i>distance</i>	double precision	クエリから取得されたデータの距離

例

```
SELECT * FROM aidb.retrieve_text(test_knowledge_base, jacket, 2);
__OUTPUT__
key | value | distance
-----+-----+-----
19337 | United Colors of Benetton Men Stripes Black Jacket | 0.2994317672742334
55018 | Lakme 3 in 1 Orchid Aqua Shine Lip Color | 0.3804609668507203
(2 rows)
```

aidb.delete_knowledge_base

ナレッジベースの構成のみをデータベースから削除します。

パラメーター

Parameter	Type	Default	Description
knowledge_base_name	TEXT		削除するナレッジベースの名前

例

```
select aidb.delete_knowledge_base(test_knowledge_base);
__OUTPUT__
delete_knowledge_base
-----
(1 row)
```

promoteb.create_volume

ナレッジベースのデータソースとして使用するために、PGFS ストレージ場所からボリュームを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT		作成するボリュームの名前
server_name	TEXT		ボリュームに使用するストレージ場所の名前
path	TEXT		保存場所のボリュームへのパス
mime_type	TEXT		ボリューム内のデータの種類テキストまたはイメージ

例

```
select aidb.create_volume(demo_bucket_vol, demo_bucket, demo_bucket/demo_folder,
↪Text);
__OUTPUT__
create_volume
-----
(1 row)
```

promoteb.list_volumes

データベースに作成されたすべてのボリュームをリストします。

例

```
select * from aidb.list_volumes();
```

promoteb.delete_volume

データベースからボリュームを削除します。

パラメーター

Parameter	Type	Default	Description
volume_name	TEXT		削除するボリュームの名前

例

```
select aidb.delete_volume(demo_bucket_vol);
__OUTPUT__
delete_volume
-----
(1 row)
```

Pipelines PGFS reference

EDB Postgres AI - AI アクセラレーターパイプライン PGFS のこのリファレンスドキュメントには、*Pipelines PGFS reference* 拡張機能で使用可能なファンクションとビューに関する情報が含まれています。これらの機能は、AIDB に S3 互換ファイルシステム、Google Cloud Storage バケット、ローカルファイルシステムへのアクセスを提供します。

pgfs

ファンクション

- *pgfs.create_foreign_table*
- *pgfs.create_storage_location*
- *pgfs.create_storage_location_with_foreign_table*
- *pgfs.delete_storage_location*
- *pgfs.list_storage_locations*
- *pgfs.get_storage_location*
- *pgfs.update_storage_location*

ファンクション

pgfs.create_foreign_table

データベースに外部テーブルを作成します。

パラメーター

Parameter	Type	Default	Description
<i>table_name</i>	text		NA
<i>server_name</i>	text		NA

例

```
SELECT pgfs.create_foreign_table(my_table, my_server);
```

pgfs.create_storage_location

データベースに保存場所を作成します。

パラメーター

Parameter	Type	Default	Description
name	text		保存場所の名前
url	text		この保存場所の URL 接頭辞 `s3:`、`gs:`、または `file:`
options	json		保存場所のオプションオプション
credentials	json		保存場所の資格情報オプション

または

Parameter	Type	Default	Description
name	text		保存場所の名前
url	text		この保存場所の URL 接頭辞 `s3:`、`gs:`、または `file:`
msl_id	uuid		未使用
options	json		保存場所のオプションオプション
credentials	json		保存場所の資格情報オプション

例

```
SELECT pgfs.create_storage_location(my_storage,
                                     s3://my_bucket, 123e4567-e89b-12d3-a456-
→426614174000,
                                     {},
                                     {});
```

pgfs.create_storage_location_with_foreign_table

データベースに格納場所を作成し、外部テーブルに関連付けます。

パラメーター

Parameter	Type	Default	Description
<i>storage_location_name</i>	text		保存場所の名前
<i>url</i>	text		この保存場所の URL 接頭辞 `s3:`、 `gs:` または <i>file</i> :
<i>options</i>	json		保存場所のオプションオプションナル
<i>credentials</i>	json		保存場所の資格情報オプションナル

または

Parameter	Type	Default	Description
<i>storage_location_name</i>	text		保存場所の名前
<i>url</i>	text		この保存場所の URL 接頭辞 `s3:`、 `gs:` または <i>file</i> :
<i>msl_id</i>	uuid		未使用
<i>options</i>	json		保存場所のオプションオプションナル
<i>credentials</i>	json		保存場所の資格情報オプションナル

例

```
SELECT pgfs.create_storage_location_with_foreign_table(my_storage,
                                                    s3://my_bucket, 123e4567-e89b-
↪ 12d3-a456-426614174000,
                                                    {},
                                                    {});
```

pgfs.delete_storage_location

データベースから保存場所を削除します。

パラメーター

Parameter	Type	Default	Description
<i>storage_location_name</i>	text		保存場所の名前

例

```
SELECT pgfs.delete_storage_location(my_storage);
```

pgfs.get_default_storage_location

デフォルトの保存場所を返します。

返品

Column	Type	Description
<i>default_storage_location</i>	text	NA

例

```
SELECT * FROM pgfs.get_default_storage_location();
```

pgfs.get_storage_location

保存場所に関する情報を返します。

パラメーター

Parameter	Type	Default	Description
<i>storage_location_name</i>	text		保存場所の名前

返品

Column	Type	Description
<i>name</i>	text	NA
<i>url</i>	text	NA
<i>msl_id</i>	uuid	NA
<i>options</i>	json	NA
<i>credentials</i>	json	NA

例

```
SELECT * FROM pgfs.get_storage_location(my_storage);
```

pgfs.list_storage_locations

データベース内のすべての保存場所をリストします。

返品

Column	Type	Description
<i>name</i>	text	NA
<i>url</i>	text	NA
<i>msl_id</i>	uuid	NA
<i>options</i>	json	NA
<i>credentials</i>	json	NA

例

```
SELECT * FROM pgfs.list_storage_locations();
```

pgfs.set_default_storage_location

デフォルトの保存場所を設定します。

パラメーター

Parameter	Type	Default	Description
<i>storage_location_name</i>	text		NA

例

```
SELECT pgfs.set_default_storage_location(my_storage);
```

pgfs.update_storage_location

データベース内の保存場所を更新します。

パラメーター

Parameter	Type	Default	Description
<i>name</i>	text		NA
<i>url</i>	text		NA
<i>msl_id</i>	uuid		NA
<i>options</i>	json		NA
<i>credentials</i>	json		NA

例

```
SELECT pgfs.update_storage_location(my_storage, s3://my_bucket, 123e4567-e89b-12d3-
↪a456-426614174000, {}, {});
```

ビュー

pgfs スキーマには、ユーザー向けビューがありません。

テーブル

pgfs スキーマにはユーザー向けテーブルがありません。

Pipelines retrievers (deprecated) reference

重要

The use of the term "retrievers" has been deprecated in version 4.0 and later. It is replaced by the term "knowledge base" and users should consult the [Pipelines knowledge bases reference](#) reference content. For backwards compatibility, the retrievers API is still available, but it is recommended to use the knowledge base API for new implementations. This section covers the deprecated functions only. For the new API, see [Pipelines knowledge bases reference](#).

helpb.RetrieverSourceDataFormat

もう利用できません。代わりに aidb.PipelineDataFormat を使用してください。

ファンクション

aidb.create_retriever_for_table 非推奨

置き換えられました aidb.create_table_knowledge_base

指定されたテーブルのリトリーバーを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	レトリバーの名前
model_name	TEXT	Required	使用するモデルの名前
source_table	regclass	Required	ソースとして使用するテーブルの名前
source_data_column	TEXT	Required	使用するソーステーブルの列名
source_data_type	helpb.PipelineDataFormat	Required	その列のデータの種類"Text"."Image","PDF"
source_key_column	TEXT	'id'	行を参照するためのキーとして使用する列
vector_table	TEXT	NULL	NA
vector_data_column	TEXT	'embeddings'	NA
vector_key_column	TEXT	'id'	NA
topk	INTEGER	1	NA
distance_operator	helpb.DistanceOperator	'L2'	NA
options	JSONB	'{}'::JSONB	オプション
index_type	TEXT	'vector'	ベクトルテーブルに使用するインデックスのタイプ。

distance_operator	index_type
L2	<i>vector_l2_ops</i>
InnerProduct	<i>vector_ip_ops</i>
Cosine	<i>vector_cosine_ops</i>
L1	<i>vector_l1_ops</i>

- `index_type` が `ivfflat` に設定されている場合、システムはベクターテーブルに `IVFFlat` インデックスを作成します。
- `index_type` が `disabled` に設定されている場合、インデックスは作成されません。

```
SELECT aidb.create_retriever_for_table(
    name => test_retriever,
    model_name => simple_model,
    source_table => test_source_table,
    source_data_column => content,
    source_data_type => Text,
);
```

aidb.create_retriever_for_volume 非推奨

置き換えられました `aidb.create_volume_knowledge_base`

指定された PGFS ボリュームのリトリーバーを作成します。

パラメーター

Parameter	Type	Default	Description
name	TEXT	Required	レトリーバーの名前
model_name	TEXT	Required	モデルの名前
source_volume_name	TEXT	Required	ボリュームの名前
vector_table	TEXT	NULL	ベクタテーブルの名前
vector_data_column	TEXT	'embeddings'	ベクター列の名前
vector_key_column	TEXT	'id'	キーカラムの名前
topk	INTEGER	1	返される結果の数
distance_operator	aidb.distanceoperator	'L2'	距離演算子
options	JSONB	'{}'::JSONB	オプション
index_type	TEXT	'vector'	ベクトルテーブルに使用するインデックスのタイプ。

distance_operator	index_type
L2	<i>vector_l2_ops</i>
InnerProduct	<i>vector_ip_ops</i>
Cosine	<i>vector_cosine_ops</i>
L1	<i>vector_l1_ops</i>

- `index_type` が `ivfflat` に設定されている場合、システムはベクターテーブルに **IVFFlat** インデックスを作成します。
- `index_type` が `disabled` に設定されている場合、インデックスは作成されません。

```
SELECT aidb.create_retriever_for_volume(
    name => demo_vol_retriever,
    model_name => simple_model,
    source_volume_name => demo_bucket_vol
);
```

aidb.set_retriever_auto_processing 非推奨

置き換えられました `aidb.set_auto_knowledge_base`

このナレッジベースの自動処理モードを設定します。

パラメーター

Parameter	Type	Default	Description
retriever_name	TEXT		自動埋め込みを有効にするレトリバーの名前
mode	aidb.PipelineAutoPro		目的の自動処理モード

例

```
SELECT aidb.set_retriever_auto_processing(test_retriever, Live);
SELECT aidb.set_retriever_auto_processing(test_retriever, Disabled);
```

aidb.delete_retriever 非推奨

置き換えられました :ref:`aidb.delete\_knowledge\_base` <aidb.delete_knowledge_base>

レトリバーの構成のみをデータベースから削除します。

パラメーター

Parameter	Type	Default	Description
retriever_name	TEXT		削除するレトリバーの名前

例

```
select aidb.delete_retriever(test_retriever);
__OUTPUT__
delete_retriever
-----
(1 row)
```

EDB Postgres AI - AI Accelerator release notes

EDB Postgres AI - AI Accelerator では、AI Accelerator の最新バージョンを説明します。リリースノートは、各リリースの新機能に関する情報を提供します。テクニカルプレビューとの違いは、1.0.7 リリースノートで強調表示されています。

AI Accelerator version	Release Date
<i>AI Accelerator - Pipelines 4.4.0 release notes</i>	13 Aug 2025
<i>AI Accelerator - Pipelines 4.3.0 release notes</i>	04 Jul 2025
<i>AI Accelerator - Pipelines 4.2.0 release notes</i>	16 Jun 2025

次のページに続く

表 89 – 前のページからの続き

AI Accelerator version	Release Date
<i>AI Accelerator - Pipelines 4.1.1 release notes</i>	28 May 2025
<i>AI Accelerator - Pipelines 4.1.0 release notes</i>	19 May 2025
<i>AI Accelerator - Pipelines 4.0.1 release notes</i>	09 May 2025
<i>AI Accelerator - Pipelines 4.0.0 release notes</i>	05 May 2025
<i>AI Accelerator - Pipelines 3.0.1 release notes</i>	03 Apr 2025
<i>AI Accelerator - Pipelines 2.2.1 release notes</i>	13 Mar 2025
<i>AI Accelerator - Pipelines 2.1.2 release notes</i>	24 Feb 2025
<i>AI Accelerator - Pipelines 2.1.1 release notes</i>	03 Feb 2025
<i>AI Accelerator - Pipelines 2.0.0 release notes</i>	13 Jan 2025
1.0.7	10 Dec 2024

AI Accelerator - Pipelines 4.4.0 release notes

リリース日 2025 年 8 月 13 日

このリリースでは、スーパーユーザー権限なしで `aidb` を使用する機能を導入しました。

ハイライト

- セキュリティとアクセス制御を強化するために、非スーパーユーザー実行サポートを導入しました。
- 大規模なデータセットでナレッジベースパイプラインを実行する際のパフォーマンスの向上。

機能強化

Description	Addresses
非スーパーユーザーのサポートの追加 --- <code>'aidb'</code> は、セキュリティとアクセス制御を強化するために非スーパーユーザーの実行サポートを持つことができます。	
パフォーマンスの向上 --- 大規模なデータセットでナレッジベースパイプラインを実行する際の <code>'aidb'</code> のパフォーマンスを向上させることができます。	

AI Accelerator - Pipelines 4.3.0 release notes

リリース日 2025 年 7 月 4 日

これは、バグ修正と小さな機能強化を含むマイナーリリースです。

ハイライト

- これは、すべての設定とパイプラインをそのままにして、`ALTER EXTENSION aidb UPDATE;` を介したインプレース更新をサポートする最初のリリースです。
- 2つの新しい LLM がサポートされるようになりました。新しいビルトインモデルと外部 Google Gemini としての「Llama instruct」、新しいモデルプロバイダーを介した外部 Google Gemini。

機能強化

Description	Addresses
Google Gemini のサポートを追加 - `aidb` が Gemini テキストモデルにアクセスできるようになりました。	
ローカルラマのサポートを追加 - `aidb` は、ローカルラマ指示モデルを実行できるようになりました。	
パフォーマンスへの影響を引き起こした過剰なログメッセージを削除しました。 - 処理された各レコードのログに使用されるナレッジベースパイプラインのライブ処理モード。これはもう当てはまりません。これらのログの印刷とストリーミングには、多くの場合、処理自分自身よりも時間がかかります。	
ユーザーが特定の文字列をパイプラインの名前として使用できないバグを修正しました。 - 大文字を含む文字列は、パイプラインの名前としてサポートされていません。これは修正されました。	

AI Accelerator - Pipelines 4.2.0 release notes

リリース日 2025 年 6 月 16 日

これは、バグ修正と小さな機能強化を含むマイナーリリースです。

ハイライト

- ナレッジベースパイプラインのデータソースとしてボリュームを使用する場合、`retrieve_text()` ファクションのサポートを追加します。
- ボリュームナレッジベースでのローカルファイル処理のバグ修正。
- ハイブリッドマネージャー展開でのモデルの同期を簡単にするバグ修正。
- 外部スキーマでのプリペアラーパーパイプラインボリュームの宛先のサポートを追加します。

機能強化

Description	Addresses
ナレッジベースパイプラインのデータソースとしてボリュームを使用する場合、 <code>retrieve_text()</code> ファクションのサポートを追加します。 <code>retrieve_text()</code> ファクションは、データソースとしてボリュームを使用するナレッジベースで使えるようになりました。以前は、 <code>retrieve_key()</code> のみがサポートされており、ユーザーはアプリケーションにオブジェクトのコンテンツを手動でロードする必要がありました。	
準備パイプラインを構成する際の必須値の減少。一部の準備パイプラインオプションのデフォルト値を追加して、必須設定の数を減らします。	

次のページに続く

表 92 – 前のページからの続き

Description	Addresses
外部スキーマでのプリペアラーパーパイプラインボリュームの宛先のサポートを追加します。プリペアラーパーパイプラインの宛先として使用されるボリュームは、postgres データベース内の任意のスキーマに配置できるようになりました。これで、AIDB のスキーマのサポートが完了しました。	

バグ修正

Description	Addresses
ボリュームナレッジベースのローカルファイル処理のバグ修正。ファイル結果の順序に問題があるため、ボリュームナレッジベース内の多数のローカルファイルが処理されない問題が存在しました。これは修正されました。	

AI Accelerator - Pipelines 4.1.1 release notes

リリース日 2025 年 5 月 28 日

これは、バグ修正を含むマイナーリリースです。注このリリースは、新しい PGFS バージョン 2.0.2 に依存しています。AIDB と共に PGFS を使用している場合、PGFS も更新してください。

ハイライト

- バグ修正と UX の機能強化。

機能強化

Description	Addresses
ボリュームナレッジベースの「実行 ID」の追跡を PG シーケンスから PG テーブル列に変更します--- ボリュームソースを使用するナレッジベースは、ソースで行われる各リストにインクリメント ID を割り当てる必要があります。これは、リストが完了したときを検出し、以前に表示されたオブジェクトが検出されなくなったことを認識するために必要です。このリリースでは、以前に使用された Postgres シーケンスが Postgres DistributedPGD クラスタで機能しなかったため、実行 ID を追跡する方法がテーブル列に変更されます。 任意の Postgres スキーマの AIDB ボリュームのサポート --- AIDB のボリュームは、スキーマを create ファンクションに明示的に渡すか、検索パスを介して「現在のスキーマ」を設定することにより、Postgres の任意のスキーマに作成できます。AIDB パイプラインでボリュームを使用する場合、ボリューム参照を保存する必要があります。AIDB は、後でパイプラインを実行するときにスキーマを保存および明示的に使用するようになりました。以前は、検索パスを使用してボリュームを見つけていました。	

次のページに続く

表 94 – 前のページからの続き

Description	Addresses
バックグラウンドワーカーディスパッチャーは永続的ではなくなりました - -- AIDB でバックグラウンドワーカーが有効になっている場合、ルートワーカーは Postgres インスタンス内のデータベースごとにデータベースディスパッチャーを起動します。このデータベースディスパッチャーは、AIDB が DB にインストールされているかどうか、およびバックグラウンドで実行する必要があるパイプラインがあるかどうかを確認します。データベースディスパッチャーは、デフォルトで永続的なプロセスではなくなりました。データベースが AIDB を使用しない場合、データベースディスパッチャーは終了し、再起動されて 2 分ごとに AIDB をチェックします。これにより、たとえば「ドロップテーブル」コマンドをブロックする DB への永続的な接続を保持することが回避されます。AIDB が DB にインストールされている場合、データベースディスパッチャーワーカーは実行され続けます。ユーザーは拡張機能をドロップして、接続を解放できます。	

AI Accelerator - Pipelines 4.1.0 release notes

リリース日 2025 年 5 月 19 日

これは、プリペアラーパーパイプラインとモデル API プロバイダーの機能強化を含むマイナーリリースです。

ハイライト

- データの形状を変換する操作の Preparer 結果の自動ネスト解除。
- 外部モデルを使用した埋め込みのバッチ処理。

機能強化

Description	Addresses
データの形状を変換する操作の Preparer 結果の自動ネスト解除。 --- 追加ディメンションで入力データの形状を変換する操作の準備パイプラインは、結果コレクションのネストを解除します。これにより、プリペアラーパーの出力を他のプリペアラーパーまたはナレッジベースがより簡単に使用できます。ネストされていない結果は、新しいディメンションを追跡するために新しい`part_id`列とともに返されます。ソースキーと part_id の組み合わせを一意に識別する新しい`unique_id`列もあります。 外部モデルを使用した埋め込みのバッチ処理。 --- 外部モデルプロバイダー <i>embeddings</i>、<i>openai_embeddings</i>、および`nim_embeddings`は、複数の同時要求ではなく、単一の要求で入力のバッチを送信できるようになりました。これにより、パフォーマンスとハードウェア使用率が向上します。この機能は完全に構成可能で、無効にすることもできます。	

次のページに続く

表 95 – 前のページからの続き

Description	Addresses
<code>`chunk_text()`</code> プリミティブ関クションの出力列を変更します。 - - - <code>`chunk_text()`</code> プリミティブ関クションによって返される列挙列は、他の Preparer プリミティブ/操作と一致するように、 <code>`chunk_id`</code> ではなく <code>`part_id`</code> になりました。	

AI Accelerator - Pipelines 4.0.1 release notes

リリース日 2025 年 5 月 9 日

これは、いくつかのバグ修正とナレッジベースパイプラインの機能強化を含むマイナーリリースです。

ハイライト

- バグ修正とパフォーマンスの向上。
- HCP のモデル統合の簡素化。

機能強化

Description	Addresses
ナレッジベースとプリペアラーパーパイプラインスキーマ処理のバグ修正。 - - - ナレッジベースとプリペアラーパーパイプラインは、ソースおよび宛先テーブル/ボリュームの任意の Postgres スキーマをサポートするようになりました。バグにより、ユーザーは作成パイプライン呼び出しでソースまたは宛先テーブル/ボリュームを参照するときに、Postgres 修飾識別子 <code>`schema.name`</code> を介して明示的なスキーマを構成できませんでした。このバグは修正されました。	
ナレッジベースの結果の精度のバグ修正。 - - - ナレッジベースパイプラインのバッチ処理コードのバグは、特定の状況下で、取得中に不正確な結果をもたらします。このバグは 4.0.0 リリースで導入され、現在修正されています。	
HCP のモデル統合の簡素化。 - - - HCP モデルサービングインフラストラクチャで実行されている外部モデルをリストし、新しいヘルパー関クション <code>`aidb.get_hcp_models()`</code> および <code>`aidb.create_hcp_model()`</code> を使用して AIDB に統合できるようになりました。	
外部モデルを使用したエンベディング処理のパフォーマンスの強化。 - - - エンベディングに外部モデルを使用する場合 <code>nim_embeddings</code> モデルアダプターなど、AIDB は追加の API 呼び出しを実行して、応答タイプのモデルサービスをプローブします。これらの呼び出しは、ほとんどの状況でスキップされ、埋め込み処理を実行する際のオーバーヘッドを最小限に削減するようになりました。AIDB は、実際の結果を使用して応答タイプを決定します。	

AI Accelerator - Pipelines 4.0.0 release notes

リリース日 2025 年 5 月 5 日

このリリースでは、ナレッジベースパイプラインの完全書き換えられた処理バックエンドを導入します。レトリバーの名前はナレッジベースに変更され、一部のタイプとファンクションの名前も一貫性のために変更されます。プリペアラーパーパイプラインが OCR をサポートし、その結果をボリュームに出力できるようになりました。いつものように、バグ修正や既存の機能のその他の機能強化も含まれています。

ハイライト

- バックグラウンド処理、バッチ処理、テーブルとボリュームの完全な反復同期をサポートするナレッジベースパイプラインの新しい処理バックエンド。
- OCR サポートを準備者パイプラインに追加しました。
- プリペアラーパーパイプライン出力をボリュームに書き込む機能を追加しました。
- レトリバーのナレッジベースへの名前変更。
- 一貫性を維持するためにプリペアラーパーファンクションの名前を変更しました。
- 一貫性を保つために、パイプライン構成の「処理モード」の名前を「自動処理」に変更しました。
- Debian 12 および RHEL 9 での AI アクセラレーターの arm64 サポートを追加しました。
- 構成可能な TLS とカスタム証明書検証を使用して、プリペアラーパー出力をオブジェクトストレージボリュームに書き込むサポートを追加しました。

機能強化

Description	Addresses
レトリバーの名前をナレッジベースに変更し、プリペアラーフランクションの名前を変更しました。----- レトリバーパイプラインの名前がナレッジベースパイプラインに変更されました。ファンクション <code>`aidb.create_retriever_for_table()`</code> および <code>`aidb.create_retriever_for_volume()`</code> の名前は、それぞれ <code>`aidb.create_knowledge_base_for_table()`</code> および <code>`aidb.create_knowledge_base_for_volume()`</code> に変更されました。また、<code>`aidb.delete_retriever`</code> ファンクションの名前が <code>`aidb.delete_knowledge_base()`</code> に変更され、<code>`aidb.RetrieverSourceDataFormat`</code> 列挙型の名前が <code>`aidb.PipelineDataFormat`</code> に名前変更されました。一貫性を保つために、プリペアラーフパイプラインファンクションの名前も変更されました。<code>`aidb.create_preparer_for_table()`</code> ファンクションの名前は <code>`aidb.create_table_preparer()`</code> に変更され、<code>`aidb.create_preparer_for_volume()`</code> の名前は <code>`aidb.create_volume_preparer()`</code> に変更されました。<code>`aidb.retrievers`</code> ビューは <code>`aidb.knowledge_bases`</code> ビューになりました。<code>`aidb.knowledge_bases`</code> および <code>`aidb.preparers`</code> ビューのエイリアスも、それぞれ <code>`aidb.kbs`</code> および <code>`aidb.preps`</code> として追加されました。古い機能とビュー名はまだ利用できますが、非推奨としてマークされています。これらは将来のリリースで削除される予定です。	
一貫性を保つために、パイプライン構成の「処理モード」の名前が「自動処理」に変更されました----- パイプライン作成、構成、および概要機能の「自動処理」の設定の名前が <code>`processing_mode`</code> から <code>`auto_processing`</code> に変更されました。これは、自動処理と呼ばれる機能自分自身の名前と一貫性を保つためにこれを行いました。	
プリペアラーフパイプライン出力をボリュームに書き込む機能を追加しました。----- Preparer パイプラインは、出力先として AIDB ボリュームたとえば PGFS を使用する S3 バケットをサポートするようになりました。以前は、Postgres テーブルのみがサポートされていました。入力ソースとしてのボリュームはテーブルとともに既にサポートされていたため、4つのソース/デスティネーションの組み合わせすべてが可能になりました。	
Debian 12 および RHEL 9 で AI アクセラレーターの arm64 サポートを追加しました。----- AI アクセラレーターは、Debian 12 および RHEL 9 で arm64 アーキテクチャをサポートするようになりました。これには、<code>`aidb`</code> パッケージでの arm64 アーキテクチャのサポートが含まれます。	
<code>aidb.rerank_text()</code> ファンクションは、豊富なテーブルとして結果を返すようになりました----- <code>`aidb.rerank_text()`</code> ファンクションは、ランキングクエリに基づいてテキストのリストを再ランク付けするために使用されますが、テキストごとに1行の結果をテーブルで結果を返すようになりました。テーブルには、生の <code>logit_score</code>、テキスト自分自身、および入力からのインデックスの列が含まれています。これにより、他のデータと結合したり、<code>logit_score</code> で行を並べ替えたりすることにより、大規模な SQL クエリーで結果を使用することが非常に簡単になります。	

次のページに続く

表 97 – 前のページからの続き

Description	Addresses
HTTP API を介してアクセスされる AI モデルのデバッグ機能の向上 ----- AIDB は、API を介して外部モデルを使用するときに発生したエラーの豊富なデバッグ出力を印刷するようになりました。これには、NIM モデルと OpenAI API 互換モデルが含まれます。HTTP ステータスコード、エラー応答コンテンツ、および要求コンテンツはログに記録され、ユーザーが不正なエンドポイント、不正なモデル構成、不正な資格情報などの問題をデバッグするのに役立ちます。	
HTTP API を使用するモデルの TLS 構成のサポートを追加しました。 ----- 外部 HTTP エンドポイントに接続するモデルを、`tls_config` オブジェクトを介して TLS 設定で構成できるようになりました。これは、開発に `insecure_skip_verify` を使用した証明書検証の無効化、または `ca_path` フィールドを使用したカスタム認証局の指定をサポートします。これらの設定により、自己署名またはプライベート CA 証明書を使用してサービスとの安全な統合が可能になります。	
PerformOcr オペレーションを Preparer に追加しました。 ----- Preparer パイプラインは、 NVIDIA NIM Image OCR API で PaddleOCR model を活用する `PerformOcr` オペレーションを利用できるようになりました。	

AI Accelerator - Pipelines 3.0.1 release notes

リリース日 2025 年 4 月 3 日

このリリースでは、新しいプリペアラーパイプラインに自動処理サポートを追加し、2 つのバッチ処理関クションの戻り値の型に重大な変更を加えます。

いつものように、既存の機能の強化も含まれています。

ハイライト

- 新しいデータ準備パイプラインは、データソースがテーブルの場合にライブ自動処理をサポートするようになりました。
- レトリバーパイプラインは、高速なルックアップのためにベクトルインデックスを自動的に作成するようになりました。
- モデルのエンコード/デコードバッチ関クションが行を返すようになり、SQL クエリーにより自然に適合します。
- Ubuntu 24.04LTS ビルドが利用可能になりました

機能強化

Description	Addresses
preparer パイプラインのライブ自動処理 --- preparer パイプラインは、postgres テーブルであるデータソースのライブ自動処理をサポートするようになりました。有効にすると、aidb はソーステーブルにトリガーを作成して、準備タスクを自動的に実行します。 モデルのエンコード/デコードバッチファンクションは、結果の配列ではなく行を返すようになりました。 --- 2 つのファンクション <code>`aidb.decode_text_batch()`</code> および <code>`aidb.encode_text_batch()`</code> は、配列を含む単一の行ではなく、行のセットを結果として返すようになりました。この変更は、お客様のフィードバックに基づいて行われました。 <code>encode</code> ファンクションの場合、戻りの型はベクトル埋め込みのリストです。これらは配列自分自身浮動小数点数の配列です。このベクトルのセットを配列として返すことにより、2 次元配列が得られます。これらの種類の配列は Postgres では十分にサポートされておらず、ファンクションの戻り値の型でフラット化されます。戻り値の型を使用するには、フラット化された配列の複雑な解析が必要でした。これは <code>decode</code> ファンクションにとってはそれほど問題ではありませんが、戻り値の型の一貫性を維持するために両方を変更することにしました。ここで、バッチファンクションは、入力値ごとに 1 行の行セットを結果ます。自動処理自動埋め込みが新しい機能を介して構成されるようになりました。 --- 2 つのファンクション <code>`aidb.enable_auto_embedding_for_table()`</code> および <code>`aidb.disable_auto_embedding_for_table()`</code> は、非推奨としてマークされています。新しいファンクション <code>`aidb.set_retriever_auto_processing()`</code> は、「処理モード」引数を受け入れます。`Live` および `Disabled` モードは、自動埋め込みをオンまたはオフにするために使用されます。プリベアラーパーパイプラインの場合、ファンクション <code>`aidb.set_preparer_auto_processing()`</code> は同じ目的を果たします。 レトリバーパイプラインは、高速なルックアップのためにベクトルインデックスを自動的に作成するようになりました。 --- パイプラインでレトリバーを作成するときに、ベクターインデックスが自動的に作成されるようになりました。これらのインデックスは、オプションの <code>`index_type`</code> 引数を受け入れるようになった <code>`aidb.create_retriever_for_table()`</code> および <code>`aidb.create_retriever_for_volume()`</code> ファンクションで構成できます。これにより、レトリバーに使用するインデックスタイプを指定できます。デフォルト値は <code>`vector`</code> です。 <code>pgfs.allowed_local_paths</code> GUC のサポートを追加しました --- PGFS でローカルファイルアクセスを制御するために、新しい GUC <code>pgfs.allowed_local_paths</code> が追加されました。デフォルトでは、ローカルファイルへのアクセスは <code>/tmp/pgfs/</code> に制限されます。この GUC を使用して、他のディレクトリへのアクセスを許可できます。詳細については、<i>**vector-native capabilities**</i> を参照してください。	

AI Accelerator - Pipelines 2.2.1 release notes

リリース日 2025 年 3 月 13 日

このリリースでは、「プリペアラー」としても知られる 2 番目のパイプラインデータの準備を導入します。最初のパイプライン「Retriever」は、AIDB の初期リリースでローンチされ、自動埋め込み計算、セマンティック検索、および RAG サポートが可能になりました。

これらの事前定義されたパイプラインは AIDB の中核であり、一般的な AI アプリケーションワークフローにエンドツーエンドの自動化を提供します。プリペアラーの追加により、この自動化をさらに拡張し、レトリバーのようなパイプラインで使用するようデータを前処理および最適化できます。

いつものように、既存の機能の強化も含まれています。

ハイライト

- 新しいデータ準備パイプラインにより、データの自動チャンク、解析、要約が可能になり、AI アプリケーションで利用できる準備が整います。

機能

Description	Addresses
導入されたプリペアラー、データ準備のための新しいパイプライン --- 新しいプリペアラーパイプラインは、データの自動チャンク、解析、および要約をサポートし、AI アプリケーションで利用できる準備をします。	

機能強化

Description	Addresses
OpenAI モデルアダプタでの HTTP 基本認証のサポートを追加しました --- OpenAI モデルアダプタたとえば <code>completions`または`embeddings`</code> は、資格情報フィールド <code>`basic_auth`</code> を介して「HTTP Basic 認証」をサポートするようになりました。以前は、 <code>`api_key`</code> を介した「ベアラー認証」のみがサポートされていました。	
構成済みモデルの Postgresql ビューを追加しました --- AIDB で構成されたモデルは、以前のようにファンクション呼び出しが必要な <code>`aidb.list_models()`</code> よりも、ビュー <code>`aidb.models`</code> としてリストできるようになりました。	
API で外部モデルを使用する場合のモデルディメンションの自動検出を追加しました --- 外部モデルのモデルディメンションたとえば <code>`completions`または`embeddings`</code> モデルアダプターを介して、明示的に構成する必要はありません。AIDB は、返されたエンベディング/ベクターの次元数を自動検出するようになりました。	

AI Accelerator - Pipelines 2.1.2 release notes

リリース日 2025 年 2 月 24 日

このメンテナンスリリースでは、2 つのバグを修正し、モデルプロバイダーのリスト出力を改善します。

ハイライト

- モデルプロバイダーの出力には、プロバイダーの説明が含まれるようになりました。

機能強化

Description	Addresses
モデルプロバイダーの説明の追加 --- モデルプロバイダービュー`select * from aidb.model_providers;`には、各モデルプロバイダーの目的と機能を詳細に説明する説明列が含まれます。	

バグ修正

Description	Addresses
ボリュームリストコマンドを修正しました --- `aidb.list_volumes();`呼び出しが機能するようになりました。バグにより、この呼び出しは成功できませんでした。他の機能に影響を与えることなく。	
検索パスの処理を修正しました --- `aidb`を search_path に追加するために使用される一部の内部コマンド。これは、既存の検索パスの内容によっては、特定の場合に失敗する場合があります。この問題を回避するために、search_path を変更しないように実装を変更しました。これは、既存の search_path が AIDB によって変更されないことも意味します。	
資格情報は OpenAI 互換モデルで完全にオプションになりました --- 以前は、`aidb.create_model()`ファンクションでは、空の値を含むキーを API キーに渡す必要がありました。`credentials`パラメーターを省略できるようになりました。	

AI Accelerator - Pipelines 2.1.1 release notes

リリース日 2025 年 2 月 3 日

このリリースでは、Nvida NIM のサポートを追加し、新しいモデル名を導入し、取得パイプラインを改善します。

ハイライト

- Nvidia NIM のサポートが追加されました。
- `embeddings` および `completions` は新しいモデル名です。
- NIM を使用したリランキングが利用できるようになりました。
- レトリバーパイプラインのソーステーブルがスキーマをサポートするようになりました。

機能強化

Description	Addresses
openai_embedding および openai_completion の名前変更 --- モデル名 <code>`openai_embedding`</code> および <code>`openai_completion`</code> は、それぞれ <code>`embeddings`</code> および <code>`completions`</code> に名前が変更されました。古い名前は、下位互換性のために引き続きサポートされています。	
モデルプロバイダーの資格情報管理の向上 --- <code>create_model()</code> 呼び出しでのモデルプロバイダーの資格情報管理を改善します。資格情報が既に存在する場合、新しい資格情報はサイレントに無視されなくなりました。	
モデルレジストリに使用される FDW 外部データラッパーの追加ハンドラー --- ハンドラーには機能がありませんが、FDW ハンドラーインターフェイスを満たす。これにより、FDW にハンドラーがない場合に発生する PG システムコールの特定の問題が回避されます。	
スキーマサポート/完全修飾識別子を取得者パイプラインに追加しました。 --- テーブルソースを使用するレトリバーの場合、ソーステーブルとベクターテーブルを PG データベース内の任意のスキーマにできるようになりました。スキーマは、レトリバーの作成中に完全修飾識別子を渡すか、省略できます。スキーマを省略した場合、該当するスキーマを解決および保存します。	

AI Accelerator - Pipelines 2.0.0 release notes

リリース日 2025 年 1 月 13 日

フィードバックに応じて、EDB Postgres AI - AI Accelerator - Pipelines の GA リリースの更新。

ハイライト

- ファンクションと引数の API 名前付けを改善しました。

機能強化

Description	Addresses
ファンクション/引数の名前の変更 --- 管理機能は、 <code>`register`</code> ではなく <code>`create`</code> を使用し、 <code>`drop`</code> ではなく <code>`delete`</code> を使用するよう移動します。ファンクションの引数名は、一貫性を保つために前述の <code>`p_`</code> を失います。	

Legal

Open Source Licenses for software incorporated in EDB Postgres AI - AI Accelerator - Pipelines

ライセンスの概要

次の表は、EDB Postgres AI - AI Accelerator - Pipelines に組み込まれているソフトウェアのライセンスをリストしています。

- MIT
- Apache 2.0
- Apache 2.0 例外あり
- BSD Two Clause
- BSD 三句
- ICSL
- OpenSSL
- zlib
- Mozilla Public License 2.0
- Unicode DFS 2016
- Unicode 3.0
- イリノイ大学/NCSA

パッケージごとのライセンスの詳細

- attempts-src 0.3.2 <https://github.com/blas-lapack-rs/accelerate-src> : (MIT ライセンスまたは Apache ライセンス 2.0)
- addr2line 0.24.2 MIT ライセンスまたは Apache ライセンス 2.0
- adler2 2.0.0 <https://crates.io/crates/adler2> : (BSD ゼロ句ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0)
- alexcrichon/cfg-if 1.0.0 <https://github.com/alexcrichon/cfg-if> : (MIT ライセンスおよび Apache ライセンス 2.- 0)
- alexcrichon/jobserver-rs 0.1.32 <https://github.com/alexcrichon/jobserver-rs> : Apache License 2.0
- alexcrichon/openssl-probe 0.1.5 <https://github.com/alexcrichon/openssl-probe> : (MIT ライセンスまたは Apache ライセンス 2.0)
- alexcrichon/proc-macro2 1.0.89 <https://github.com/alexcrichon/proc-macro2> : (MIT ライセンスまたは Apache ライセンス 2.0)
- alexcrichon/rustc-demangle 0.1.24 <https://github.com/alexcrichon/rustc-demangle> : (MIT ライセンスまたは Apache ライセンス 2.0)

- alexcrichton/socket2-rs 0.5.7 <https://github.com/alexcrichton/socket2-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- aligned-vec 0.5.0 <https://crates.io/crates/aligned-vec> : MIT License
- allocator-api2 0.2.18 <https://github.com/zakarumych/allocator-api2> MIT ライセンスまたは Apache ライセンス 2.0
- android_system_properties 0.1.5 https://github.com/nical/android_system_properties : (MIT ライセンスまたは Apache ライセンス 2.0)
- android-tzdata 0.1.1 <https://crates.io/crates/android-tzdata> : (MIT ライセンスまたは Apache ライセンス 2.0)
- annotate-snippets 0.9.2 <https://crates.io/crates/annotate-snippets> : (MIT ライセンスまたは Apache ライセンス 2.0)
- anstyle 1.0.10 <https://github.com/rust-cli/anstyle> MIT ライセンスまたは Apache ライセンス 2.0
- とにかく 1.0.93 MIT ライセンスまたは Apache ライセンス 2.0
- 任意の 1.4.1 MIT ライセンスまたは Apache ライセンス 2.0
- arg_enum_proc_macro 0.3.4 https://crates.io/crates/arg_enum_proc_macro MIT License
- asset-json-diff 2.0.2 <https://github.com/davidpdrsn/assert-json-diff> : MIT ライセンス
- async-trait 0.1.83 MIT ライセンスまたは Apache ライセンス 2.0
- atomic-traits 0.3.0 <https://github.com/flier/rust-atomic-traits> : (MIT ライセンスまたは Apache ライセンス 2.0)
- atomic-waker 1.1.2 <https://github.com/stjepang/atomic-waker> MIT ライセンスまたは Apache ライセンス 2.0
- autocfg 1.4.0 Apache License 2.0
- av1-グレイン 0.2.3 <https://github.com/rust-av/av1-grain> BSD 2 句「簡素化」ライセンス
- avif-serialize 0.8.2 <https://lib.rs/avif-serialize> BSD 3 句「新規」または「改訂」ライセンス
- bit_field 0.10.2 https://crates.io/crates/bit_field MIT ライセンスまたは Apache ライセンス 2.0
- bitflags 1.3.2 <https://github.com/rust-lang-nursery/bitflags> MIT ライセンスまたは Apache ライセンス 2.0
- bitflags 2.6.0 <https://github.com/rust-lang-nursery/bitflags> MIT ライセンスまたは Apache ライセンス 2.0
- bitstream-io 2.6.0 <https://github.com/tuffly/bitstream-io> MIT ライセンスまたは Apache ライセンス 2.0
- bitvec 1.0.1 <https://myrrlyn.net/crates/bitvec> MIT ライセンス
- bit-vec 0.6.3 <https://github.com/contain-rs/bit-vec> MIT ライセンスまたは Apache ライセンス 2.0
- block-buffer 0.10.4 MIT License OR Apache License 2.0
- bluss/arrayvec 0.7.6 Apache License 2.0
- bluss/ether 1.13.0 Apache License 2.0
- bluss/indexmap 2.6.0 <https://docs.rs/indexmap/> : (MIT ライセンスおよび Apache ライセンス 2.0)

- bluss/scopeguard 1.2.0 : Apache License 2.0
- briansmith/ring 0.17.8 <https://crates.io/crates/ring> : (MIT ライセンスと ISC ライセンスとオープン SSL ライセンス)
- briansmith/untrusted 0.9.0 : ISC ライセンス
- built 0.7.5 : MIT License
- BurntSushi/byteorder 1.5.0 <https://github.com/BurntSushi/byteorder> : The Unlicense
- BurntSushi/regex-automata 0.4.8 <https://github.com/BurntSushi/regex-automata> : (MIT ライセンスまたは Apache ライセンス 2.0)
- BurntSushi/rust-memchr 2.7.4 <https://github.com/BurntSushi/rust-memchr> : (MIT ライセンスまたはアンライセンス)
- BurntSushi/same-file 1.0.6 <https://github.com/BurntSushi/same-file> : (MIT ライセンスとアンライセンス)
- BurntSushi/ucd-generate 0.1.7 <https://github.com/BurntSushi/ucd-generate> : Apache License 2.0
- BurntSushi/walkdir 2.5.0 <https://github.com/BurntSushi/walkdir> : The Unlicense
- bytemuck 1.19.0 zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- bytemuck_derive 1.8.0 zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- byteorder-lite 0.1.0 <https://github.com/image-rs/byteorder-lite> : (MIT ライセンスまたはアンライセンス)
- camino 1.1.9 <https://crates.io/crates/camino> : (MIT ライセンスまたは Apache ライセンス 2.0)
- キャンドルコア 0.7.2 MIT ライセンスまたは Apache ライセンス 2.0
- Candle-nn 0.7.2 MIT ライセンスまたは Apache ライセンス 2.0
- キャンドルトランスフォーマー 0.7.2 MIT ライセンスまたは Apache ライセンス 2.0
- Cargo_toml 0.19.2 https://lib.rs/cargo_toml : (MIT ライセンスまたは Apache ライセンス 2.0)
- cavif (Rust) 0.11.11 <https://lib.rs/cavif> BSD 3-clause 「New」 または 「Revised」 ライセンス
- cc-rs 1.1.36 <https://docs.rs/cc> MIT ライセンスまたは Apache ライセンス 2.0
- cee-scape 0.2.0 <https://github.com/pnkfelix/cee-scape> MIT ライセンスまたは Apache ライセンス 2.0
- cexpr 0.6.0 <https://crates.io/crates/cexpr> MIT ライセンスまたは Apache ライセンス 2.0
- cfg_aliases 0.2.1 https://github.com/katharostech/cfg_aliases MIT License
- cfg-expr 0.15.8 <https://github.com/EmbarkStudios/cfg-expr> : (MIT ライセンスまたは Apache ライセンス 2.0)
- chronotope/chrono 0.4.38 <https://github.com/chronotope/chrono> : (MIT ライセンスおよび Apache ライセンス 2.0)
- clap-cargo 0.14.1 MIT ライセンスまたは Apache ライセンス 2.0
- clap-rs/clap 0.7.2 <https://clap.rs/> : MIT License
- clap-rs/clap 4.5.18 <https://clap.rs/> : (MIT ライセンスまたは Apache ライセンス 2.0)

- clap-rs/clap 4.5.20 <https://clap.rs/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- colin-kiegel/rust-derive-builder 0.20.2 : Apache License 2.0
- color_quant 1.1.0 https://crates.io/crates/color_quant MIT License
- console-rs/console 0.15.8 <https://github.com/mitsuhiko/console> : MIT License
- contains-rs/bit-set 0.5.3 <https://github.com/contain-rs/bit-set> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Convert_case 0.6.0 https://crates.io/crates/convert_case : MIT License
- cpufeatures 0.2.14 MIT ライセンスまたは Apache ライセンス 2.0
- crossbeam-deque 0.8.5 <https://github.com/crossbeam-rs/crossbeam/tree/master/crossbeam-deque> : (MIT ライセンスまたは Apache ライセンス 2.0)
- crossbeam-epoch 0.9.18 <https://github.com/crossbeam-rs/crossbeam> : (MIT ライセンスまたは Apache ライセンス 2.- 0)
- crossbeam-utils 0.8.20 <https://github.com/crossbeam-rs/crossbeam/tree/master/crossbeam-utils> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Crunchy 0.2.2 <https://crates.io/crates/crunchy> : MIT License
- fence_arbitrary 1.4.1 MIT ライセンスまたは Apache ライセンス 2.0
- dguo/strsim-rs 0.11.1 <https://github.com/dguo/strsim-rs> MIT ライセンス
- ダイジェスト 0.10.7 MIT ライセンスまたは Apache ライセンス 2.0
- ディレクトリ 5.0.1 <https://crates.io/crates/directories> : (MIT ライセンスまたは Apache ライセンス 2.0)
- dirs 5.0.1 <https://crates.io/crates/dirs> MIT ライセンスまたは Apache ライセンス 2.0
- dirs-sys 0.4.1 <https://crates.io/crates/dirs-sys> MIT ライセンスまたは Apache ライセンス 2.0
- displaydoc 0.2.5 <https://github.com/yaahc/displaydoc> MIT ライセンスまたは Apache ライセンス 2.0
- djc/quinn 0.11.5 MIT ライセンスまたは Apache ライセンス 2.0
- djc/quinn 0.11.8 MIT ライセンスまたは Apache ライセンス 2.0
- doc-comment 0.3.3 <https://crates.io/crates/doc-comment> : MIT License
- dtolnay/itoa 1.0.11 Apache License 2.0
- dtolnay/quote 1.0.37 : Apache License 2.0
- dtolnay/ Ryu 1.0.18 Apache License 2.0 OR Boost Software License 1.0
- dyn-stack 0.10.0 MIT ライセンス
- encode_unicode 0.3.6 https://crates.io/crates/encode_unicode : (MIT ライセンスまたは Apache ライセンス 2.0)
- enterprise_rs 0.8.35 https://docs.rs/encoding_rs/ : (BSD 3-clause “New” または” Revised” License AND MIT License OR Apache License 2.0)

- enum-as-inner 0.6.1 MIT ライセンスまたは Apache ライセンス 2.0
- enum-map 2.7.3 <https://crates.io/crates/enum-map> MIT ライセンスまたは Apache ライセンス 2.0
- enum-map-derive 0.17.0 <https://crates.io/crates/enum-map-derive> : (MIT ライセンスまたは Apache ライセンス 2.0)
- 同等 1.0.1 : (MIT ライセンスまたは Apache ライセンス 2.0)
- errno 0.3.9 MIT ライセンスまたは Apache ライセンス 2.0
- esaxx-rs 0.1.10 <https://github.com/Narsil/esaxx-rs> Apache License 2.0
- exr 1.73.0 <https://crates.io/crates/exr> BSD 3 条項の「新規」または「改訂」ライセンス
- eyre 0.6.12 <https://github.com/yaahc/eyre> MIT ライセンスまたは Apache ライセンス 2.0
- fallible-iterator 0.2.0 <https://crates.io/crates/fallible-iterator> : (MIT License OR Apache License 2.0)
- fancy-regex 0.13.0 <https://github.com/fancy-regex/fancy-regex> : MIT License
- ファストランド 2.1.1 <https://github.com/stjepang/fastrand> MIT ライセンスまたは Apache ライセンス 2.0
- fdeflate <https://github.com/image-rs/fdeflate> MIT ライセンスまたは Apache ライセンス 2.0
- filetime 0.2.25 <https://github.com/alexcrichton/filetime> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Fitzgen/bumpalo 3.16.0 Apache License 2.0
- fixedbitset 0.4.2 MIT ライセンスまたは Apache ライセンス 2.0
- flate2 1.0.34 <https://github.com/alexcrichton/flate2-rs> MIT ライセンスまたは Apache ライセンス 2.0
- fence 0.1.3 <https://crates.io/crates/foldhash> zlib License
- 外部タイプ 0.1.1 <https://github.com/sfackler/foreign-types> : (MIT ライセンスまたは Apache ライセンス 2.0)
- 外国型 0.3.2 <https://github.com/sfackler/foreign-types> MIT ライセンスまたは Apache ライセンス 2.0
- funty 2.0.0 <https://ferrilab.github.io/ferrilab> MIT License
- futures-task 0.3.31 <https://rust-lang-nursery.github.io/futures-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- gemm 0.17.1 : MIT ライセンス
- gemm-c32 0.17.1 <https://crates.io/crates/gemm-c32> MIT ライセンス
- gemm-c64 0.17.1 <https://crates.io/crates/gemm-c64> MIT ライセンス
- gemm-common 0.17.1 : MIT ライセンス
- gemm-f16 0.17.1 <https://crates.io/crates/gemm-f16> : MIT License
- gemm-f32 0.17.1 <https://crates.io/crates/gemm-f32> : MIT License
- gemm-f64 0.17.1 <https://crates.io/crates/gemm-f64> MIT License
- generic-array 0.14.7 <https://crates.io/crates/generic-array> MIT License

- `getrandom` 0.2.15 : (MIT ライセンスまたは Apache ライセンス 2.0)
- `getset` 0.1.3 : MIT ライセンス
- `gif` 0.13.1: <https://github.com/image-rs/image-gif> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `gimli-rs` 0.31.1 <https://docs.rs/gimli/> : Apache License 2.0
- `glob` 0.3.1 <https://github.com/rust-lang/glob> MIT ライセンスまたは Apache ライセンス 2.0
- `hash32` 0.3.1 <https://crates.io/crates/hash32> MIT ライセンスまたは Apache ライセンス 2.0
- `heck` 0.4.1 <https://github.com/withoutboats/heck> MIT ライセンスまたは Apache ライセンス 2.0
- `heck` 0.5.0 <https://github.com/withoutboats/heck> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `hermitcore/libhermit-rs` 0.3.9 <https://hermitcore.org> BSD 3 条項の「新規」または「改訂」ライセンス
- `hermitcore/libhermit-rs` 0.4.0 <https://hermitcore.org> BSD 3 条項の「新規」または「改訂」ライセンス
- `hf-hub` 0.3.2 <https://github.com/huggingface/hf-hub> Apache License 2.0
- `home` 0.5.9 MIT ライセンスまたは Apache ライセンス 2.0
- `httpdate` 1.0.3 <https://crates.io/crates/httpdate> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `hyperium/h2` 0.4.6 <https://github.com/crates-patch> : MIT License
- `hyperium/http` 1.1.0 <https://github.com/hyperium/http> : Apache License 2.0
- `hyperium/http-body` 0.1.2 <https://github.com/hyperium/http-body> : MIT License
- `hyperium/http-body` 1.0.1 <https://github.com/hyperium/http-body> : MIT License
- `hyperium/mime` 0.3.17 : MIT ライセンス
- `hyper-rustls` 0.27.3 <https://github.com/ctz/hyper-rustls> MIT ライセンスまたは ISC ライセンスまたは Apache ライセンス 2.0
- `hyper-tls` 0.6.0 <https://crates.io/crates/hyper-tls> MIT ライセンスまたは Apache ライセンス 2.0
- `hyper-util` <https://hyper.rs> MIT License
- `iana-time-zone` 0.1.61 <https://crates.io/crates/iana-time-zone> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `iana-time-zone-haiku` 0.1.2 <https://crates.io/crates/iana-time-zone-haiku> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `icu_locid` <https://icu4x.unicode.org>
- `icu_normalizer` <https://icu4x.unicode.org>
- `icu_provider` <https://icu4x.unicode.org>
- `icu_provider_macros` <https://icu4x.unicode.org>
- `idna_adapter` 1.2.0 https://docs.rs/crate/idna_adapter/latest MIT ライセンスまたは Apache ライセンス 2.0
- `image-rs/image` 0.25.5 <https://github.com/image-rs/image> MIT ライセンスまたは Apache ライセンス 2.0

- image-webp 0.2.0 <https://github.com/image-rs/image-webp> : (MIT ライセンスまたは Apache ライセンス 2.0)
- imgref 1.11.0 <https://crates.io/crates/imgref> クリエイティブコモンズゼロ v1.0 ユニバーサルまたは Apache ライセンス 2.0
- インデンター 0.3.3 <https://github.com/yaahc/indenter> : (MIT License OR Apache License 2.0)
- instant 0.1.13 BSD 3 条項の「新規」または「改訂」ライセンス
- intel-mkl-src 0.8.1 <https://crates.io/crates/intel-mkl-src>
- intel-mkl-tool 0.8.1 <https://crates.io/crates/intel-mkl-tool> MIT ライセンス
- interpolate_name 0.2.4 https://crates.io/crates/interpolate_name MIT License
- ipnet 2.10.1 MIT ライセンスまたは Apache ライセンス 2.0
- is_ci 1.2.0 https://crates.io/crates/is_ci : ISC ライセンス
- is-terminal 0.4.13 : MIT ライセンス
- japaric/heapless 0.8.0 MIT ライセンスまたは Apache ライセンス 2.0
- jedisct1/rust-siphash 0.3.11 <https://docs.rs/siphasher> : Apache License 2.0
- jpeg-decoder 0.3.1 <https://crates.io/crates/jpeg-decoder> : (MIT ライセンスまたは Apache ライセンス 2.0)
- js-sys 0.3.72 <https://rustwasm.github.io/wasm-bindgen/> MIT ライセンスまたは Apache ライセンス 2.0
- Kimundi/rustc-version-rs 0.3.3 MIT ライセンスまたは Apache ライセンス 2.0
- KyleMayes/clang-sys 1.8.1 Apache License 2.0
- stable_static 1.5.0 <https://github.com/rust-lang-nursery/lazy-static.rs> : Apache License 2.0
- lebe 0.5.2 <https://crates.io/crates/lebe> BSD 3 句「新規」または「改訂」ライセンス
- libfuzzer-sys 0.4.7 <https://crates.io/crates/libfuzzer-sys> イリノイ大学/NCSA オープンソースライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- libm 0.2.11 <https://crates.io/crates/libm> MIT ライセンス
- libredox 0.1.3 : MIT ライセンス
- linux-raw-sys 0.4.14 MIT ライセンスまたは Apache ライセンス 2.0 例外または Apache ライセンス 2.0
- ライトマップ <https://crates.io/crates/litemap>
- lock_api 0.4.12 MIT ライセンスまたは Apache ライセンス 2.0
- log 0.4.22 MIT ライセンスまたは Apache ライセンス 2.0
- LOOP9 0.1.5 <https://lib.rs/loop9> MIT License
- lru 0.12.5 <https://github.com/jeffmcfroe/lru-rs> : MIT License
- mackwic/coloured 2.1.0 <https://github.com/mackwic/colored> : Mozilla Public License 2.0
- Macro_rules_attribute 0.2.0 https://crates.io/crates/macro_rules_attribute MIT License

- Macro_rules_attribute-proc_macro 0.2.0 https://crates.io/crates/macro_rules_attribute-proc_macro MIT License
- matklad/once_cell 1.20.2 : Apache License 2.0
- might-rayon 0.1.1 <https://github.com/shssoichiro/maybe-rayon> : MIT License
- mmap2 0.9.5 MIT ライセンスまたは Apache ライセンス 2.0
- Microsoft/windows-rs 0.1.0 <https://github.com/microsoft/windows-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- microsoft/windows-rs 0.1.2 <https://github.com/microsoft/windows-rs> : MIT License
- Microsoft/windows-rs 0.2.0 <https://github.com/microsoft/windows-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Microsoft/windows-rs 0.52.0 <https://github.com/microsoft/windows-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Microsoft/windows-rs 0.52.6 <https://github.com/microsoft/windows-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- Microsoft/windows-rs 0.57.0 <https://github.com/microsoft/windows-rs> : Apache License 2.0
- minimum-lexical 0.2.1 <https://crates.io/crates/minimal-lexical> MIT License OR Apache License 2.0
- miniz_oxide 0.8.0 https://github.com/Frommi/miniz_oxide/tree/master/miniz_oxide zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- 光彦/indicatif 0.17.8 : MIT ライセンス
- mockito 1.5.0 <https://github.com/lipanski/mockito> MIT License
- monostate 0.1.13 <https://crates.io/crates/monostate> MIT ライセンスまたは Apache ライセンス 2.0
- monostate-impl 0.1.13 <https://crates.io/crates/monostate-impl> : (MIT ライセンスまたは Apache ライセンス 2.0)
- mvdnes/spin-rs 0.9.8 : MIT ライセンス
- nagisa/rust_libloading 0.8.5 : ISC ライセンス
- Native-tls 0.2.12 <https://crates.io/crates/native-tls> MIT ライセンスまたは Apache ライセンス 2.0
- new_debug_unreachable 1.0.6 https://crates.io/crates/new_debug_unreachable MIT License
- noop_proc_macro 0.3.0 https://crates.io/crates/noop_proc_macro MIT ライセンス
- nox/serde_urlencoded 0.7.1 MIT ライセンスまたは Apache ライセンス 2.0
- ntapi 0.4.1 <https://crates.io/crates/ntapi> MIT ライセンスまたは Apache ライセンス 2.0
- num 0.4.3 <https://github.com/rust-num/num> : (MIT ライセンスまたは Apache ライセンス 2.0)
- number_prefix 0.4.0 https://crates.io/crates/number_prefix : MIT License
- num-bigint 0.4.6 <https://github.com/rust-num/num-bigint> : (MIT ライセンスまたは Apache ライセンス 2.0)

- num_cpus 1.16.0 https://crates.io/crates/num_cpus : (MIT ライセンスまたは Apache ライセンス 2.0)
- num-derive 0.4.2 <https://github.com/rust-num/num-derive> : (MIT ライセンスまたは Apache ライセンス 2.0)
- num_enum 0.7.3 https://crates.io/crates/num_enum MIT ライセンスまたは Apache ライセンス 2.0 または BSD 3 条項の「新規」または「改訂」ライセンス
- num_enum_derive 0.7.3 https://crates.io/crates/num_enum_derive : (MIT ライセンスまたは Apache ライセンス 2.0 または BSD 3-clause “New” または “Revised” License)
- num-integer 0.1.46 <https://github.com/rust-num/num-integer> : (MIT ライセンスまたは Apache ライセンス 2.0)
- num-iter 0.1.45 <https://github.com/rust-num/num-iter> : (MIT ライセンスまたは Apache ライセンス 2.0)
- num-rational 0.4.2 <https://github.com/rust-num/num-rational> : (MIT ライセンスまたは Apache ライセンス 2.0)
- num-traits 0.2.19 <https://github.com/rust-num/num-traits> : (MIT ライセンスまたは Apache ライセンス 2.0)
- object-rs 0.36.5 <https://crates.io/crates/object> : Apache License 2.0
- object_store 0.10.2 MIT ライセンスまたは Apache ライセンス 2.0
- ocipkg 0.2.9 <https://crates.io/crates/ocipkg> MIT ライセンスまたは Apache ライセンス 2.0
- oci-spec 0.6.7 <https://github.com/containers/oci-spec-rs> Apache License 2.0
- oli-obk/cargo_metadata 0.18.1 : MIT ライセンス
- onig 6.4.0 <https://crates.io/crates/onig> MIT ライセンス
- onig_sys 69.8.1 https://crates.io/crates/onig_sys MIT ライセンス
- openssl-macros 0.1.1 <https://crates.io/crates/openssl-macros> : (MIT ライセンスまたは Apache ライセンス 2.0)
- options-ext 0.2.0 <https://github.com/soc/option-ext> : Mozilla Public License 2.0
- ordian/toml_edit 0.22.22 : Apache License 2.0
- owo-colors 4.1.0 : MIT ライセンス
- paholg/typenum 1.17.0 MIT ライセンスまたは Apache ライセンス 2.0
- parking_lot 0.12.3 : Apache License 2.0
- parking_lot_core 0.9.10 MIT ライセンスまたは Apache ライセンス 2.0
- 貼り付け 1.0.15 : (MIT License OR Apache License 2.0)
- pathsearch 0.2.0 <https://crates.io/crates/pathsearch> : MIT License
- pest-parser 2.7.14 <https://github.com/pest-parser/pest> : Apache License 2.0
- petgraph 0.6.5 MIT ライセンスまたは Apache ライセンス 2.0
- pgrx 0.12.8 <https://github.com/tcdi/pgrx> MIT ライセンス

- pgrx-bindgen 0.12.8 <https://github.com/pgcentralfoundation/pgrx> MIT ライセンス
- pgrx-macros 0.12.8 <https://github.com/tcdi/pgrx> MIT License
- pgrx-pg-config 0.12.8 <https://github.com/tcdi/pgrx> MIT ライセンス
- pgrx-pg-sys 0.12.8 <https://github.com/tcdi/pgrx> MIT ライセンス
- pgrx-sql-entity-graph 0.12.8 <https://github.com/tcdi/pgrx> MIT ライセンス
- pgrx-tests 0.12.8 <https://github.com/tcdi/pgrx> : MIT ライセンス
- pin-project-lite 0.2.15 <https://github.com/taiki-e/pin-project-lite> : (MIT ライセンスまたは Apache ライセンス 2.0)
- pin-utils 0.1.0 <https://crates.io/crates/pin-utils> : (MIT ライセンスまたは Apache ライセンス 2.0)
- pkg-config 0.3.31 MIT ライセンスまたは Apache ライセンス 2.0
- png 0.17.14: <https://crates.io/crates/png> : (MIT ライセンスまたは Apache ライセンス 2.0)
- ポータブルアトミック 1.9.0 MIT ライセンスまたは Apache ライセンス 2.0
- ppv-lite86 0.2.20 MIT ライセンスまたは Apache ライセンス 2.0
- proc-macro-crate 3.2.0 MIT ライセンスまたは Apache ライセンス 2.0
- proc-macro-error2 2.0.1 MIT ライセンスまたは Apache ライセンス 2.0
- proc-macro-error-attr2 2.0.0 <https://crates.io/crates/proc-macro-error-attr2> : (MIT ライセンスまたは Apache ライセンス 2.0)
- プロファイリング 1.0.16 <https://github.com/aclyσμα/profiling> MIT ライセンスまたは Apache ライセンス 2.0
- profiling-procmacros 1.0.16 <https://github.com/aclyσμα/profiling> : (MIT License OR Apache License 2.0)
- proptest 1.5.0 <https://altsysrq.github.io/proptest-book/proptest/index.html> MIT ライセンスまたは Apache ライセンス 2.0
- pulp 0.18.22 : MIT License
- qoi 0.4.1 <https://github.com/aldanor/qoi-rust> MIT ライセンスまたは Apache ライセンス 2.0
- quinn-udp 0.5.7 MIT ライセンスまたは Apache ライセンス 2.0
- ラジウム 0.7.0 <https://github.com/mystor/radium> : MIT License
- rand_chacha 0.3.1 https://crates.io/crates/rand_chacha : MIT License
- rand_core 0.6.4 https://crates.io/crates/rand_core MIT ライセンスまたは Apache ライセンス 2.0
- rand_distr 0.4.3 https://crates.io/crates/rand_distr : MIT License
- rand_xorshift 0.3.0 https://crates.io/crates/rand_xorshift : (MIT ライセンスまたは Apache ライセンス 2.0)
- rav1e 0.3.8 <https://github.com/xiph/rav1e> BSD 2 句「簡素化」ライセンス

- rav1e 0.7.1 <https://github.com/xiph/rav1e> BSD 2 句「簡素化」ライセンス
- raw-cpuid 10.7.0 <https://github.com/gz/rust-cpuid> MIT ライセンス
- rayon 1.10.0 <https://github.com/rayon-rs/rayon> MIT ライセンスまたは Apache ライセンス 2.0
- rayon 1.12.1 <https://github.com/rayon-rs/rayon> : (MIT ライセンスまたは Apache ライセンス 2.0)
- rayon-cond 0.3.0 <https://crates.io/crates/rayon-cond> : (MIT ライセンスまたは Apache ライセンス 2.0)
- 0.5.5 を再借します <https://crates.io/crates/reborrow> : MIT License
- redox-os 0.5.7 <http://www.redox-os.org> MIT ライセンス
- redox_users 0.4.6 MIT ライセンス
- regex-syntax 0.8.5 <https://github.com/rust-lang/regex> : (MIT ライセンスまたは Apache ライセンス 2.0)
- repair-aho-corasick 1.1.3 <https://github.com/BurntSushi/aho-corasick> : The Unlicense
- repair-base64 0.13.1 MIT ライセンスまたは Apache ライセンス 2.0
- repair-base64 0.22.1 MIT ライセンスまたは Apache ライセンス 2.0
- Rustc-hash 1.1.0 <https://crates.io/crates/rustc-hash> : (MIT ライセンスまたは Apache ライセンス 2.0)
- repairc-hash <https://crates.io/crates/rustc-hash> MIT ライセンスまたは Apache ライセンス 2.0
- RustCrypto/hashes 0.10.6 MIT ライセンスまたは Apache ライセンス 2.0
- RustCrypto/hashes 0.10.8 MIT ライセンスまたは Apache ライセンス 2.0
- RustCrypto/MACs 0.12.1 <https://crates.io/crates/hmac> MIT ライセンスまたは Apache ライセンス 2.0
- RustCrypto/traits 0.1.6 <https://github.com/RustCrypto/traits> MIT ライセンスまたは Apache ライセンス 2.0
- RustCrypto/traits 0.2.0 <https://github.com/RustCrypto/traits> : (MIT ライセンスまたは Apache ライセンス 2.0)
- repair-hyper 1.5.0 <https://hyper.rs> : MIT ライセンス
- repair-itertools/itertools 0.11.0 : Apache License 2.0
- repair-itertools/itertools 0.12.1 : Apache License 2.0
- repair-itertools/itertools 0.13.0 MIT ライセンスまたは Apache ライセンス 2.0
- Rustix 0.38.39 MIT ライセンスまたは例外を含む Apache ライセンス 2.0 または Apache ライセンス 2.0
- repair-lang/backtrace-rs 0.3.74 <https://github.com/rust-lang/backtrace-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- stable-lang/cargo 0.1.8 <https://github.com/rust-lang/cargo> : (MIT ライセンスまたは Apache ライセンス 2.0)
- stable-lang/hashbrown 0.15.1 MIT ライセンスまたは Apache ライセンス 2.0
- stable-lang-nursery/futures-rs 0.3.31 <http://rust-lang-nursery.github.io/futures-rs> : (MIT ライセンスおよび Apache ライセンス 2.0)

- `stable-lang/rust-bindgen` 0.70.1 <https://rust-lang.github.io/rust-bindgen/> BSD 3-clause 「New」または「Revised」ライセンス
- `repair-libc` 0.2.161 <https://github.com/rust-lang/libc> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `Rustls` <https://github.com/ctz/rustls> MIT ライセンスまたは ISC ライセンスまたは Apache ライセンス 2.0
- `Rustls-native-certs` 0.8.0 <https://github.com/ctz/rustls-native-certs> : (MIT ライセンスまたは ISC ライセンスまたは Apache ライセンス 2.0)
- `repairs-pemfile` 2.2.0 <https://github.com/rustls/pemfile> MIT ライセンスまたは ISC ライセンスまたは Apache ライセンス 2.0
- `Rustls-pki-types` 1.10.0 <https://github.com/rustls/pki-types> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repairs-webpki` 0.102.8 : ISC ライセンス
- `repair-mio` 1.0.2 <https://github.com/carllerche/mio> : MIT ライセンス
- `repair-nom` 7.1.3 <https://github.com/Geal/nom> : MIT ライセンス
- `stable-num/num-complex` 0.4.6 <https://github.com/rust-num/num-complex> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-openssl` 0.10.68 : Apache License 2.0
- `repair-openssl` 0.9.104 : MIT ライセンス
- `repair-phf` 0.11.2 <https://github.com/sfackler/rust-phf> : MIT ライセンス
- `repair-postgres` 0.19.9 <https://github.com/sfackler/rust-postgres> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-postgres` 0.2.8 <https://github.com/sfackler/rust-postgres> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-postgres` 0.6.7 <https://github.com/sfackler/rust-postgres> MIT ライセンスまたは Apache ライセンス 2.0
- `repair-postgres` 0.7.12 <https://github.com/sfackler/rust-postgres> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `stable-random/rand` 0.8.5 <https://crates.io/crates/rand> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-regex` 1.11.1 <https://docs.rs/regex> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-rgb` 0.8.50 <https://kornelski.github.io/rust-rgb/rgb/> : MIT ライセンス
- `repair-url` 1.2.1 <https://docs.rs/url/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- `repair-url` 2.3.1 <https://docs.rs/url/> : Apache License 2.0
- `repair-url` <https://docs.rs/url/> : (MIT ライセンスおよび Apache ライセンス 2.0)
- `stableversion` 1.0.18 MIT ライセンスまたは Apache ライセンス 2.0
- `repair-xattr` 1.3.1 <https://github.com/Stebalien/xattr> : Apache License 2.0

- 錆びたフォーク 0.3.0 <https://crates.io/crates/rusty-fork> MIT ライセンスまたは Apache ライセンス 2.0
- safetensors 0.4.5 <https://github.com/huggingface/safetensors> : Apache License 2.0
- seahash 4.1.0 <https://crates.io/crates/seahash> MIT License
- seanmonstar/httparse 1.9.5 <https://crates.io/crates/httparse> : (MIT ライセンスまたは Apache ライセンス 2.0)
- seanmonstar/reqwest : (MIT ライセンスまたは Apache ライセンス 2.0)
- seanmonstar/try-lock 0.2.5 <https://github.com/seanmonstar/try-lock> : MIT License
- seanmonstar/want 0.3.1 <https://github.com/seanmonstar/want> : MIT License
- security-framework 2.11.1 MIT License OR Apache License 2.0
- security-framework 2.12.0 MIT License OR Apache License 2.0
- semver 0.11.0 <https://docs.rs/crate/semver/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- semver 1.0.23 <https://docs.rs/crate/semver/> MIT ライセンスまたは Apache ライセンス 2.0
- semver-parser 0.10.2 <https://github.com/steveklabnik/semver-parser> : (MIT ライセンスまたは Apache ライセンス 2.0)
- seq-macro 0.3.5 <https://crates.io/crates/seq-macro> : (MIT ライセンスまたは Apache ライセンス 2.0)
- serde <https://serde.rs/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- serde_cbor 0.11.2 https://crates.io/crates/serde_cbor MIT ライセンスまたは Apache ライセンス 2.0
- serde_json 1.0.132 http://docs.serde.rs/serde_json/ : Apache License 2.0
- serde_plain 1.0.2 https://docs.rs/serde_plain MIT ライセンスまたは Apache ライセンス 2.0
- serde_spanned 0.6.8 <https://github.com/toml-rs/toml> : (MIT ライセンスまたは Apache ライセンス 2.0)
- SergioBenitez/version_check 0.9.5 MIT ライセンスまたは Apache ライセンス 2.0
- servo/core-foundation-rs 0.8.7 <https://github.com/servo/core-foundation-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- servo/core-foundation-rs 0.9.4 <https://github.com/servo/core-foundation-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- servo/rust-fnv 1.0.7 MIT ライセンスまたは Apache ライセンス 2.0
- servo/rust-smallvec 1.13.2 : Apache License 2.0
- servo/unicode-bidi 0.3.17 MIT License OR Apache License 2.0
- shlex 1.3.0 <https://crates.io/crates/shlex> MIT ライセンスまたは Apache ライセンス 2.0
- signal-hook-registry 1.4.2 <https://crates.io/crates/signal-hook-registry> : (MIT ライセンスまたは Apache ライセンス 2.0)
- simd-adler32 0.3.7 <https://crates.io/crates/simd-adler32> MIT ライセンス
- simd_helpers 0.1.0 https://crates.io/crates/simd_helpers : MIT License

- 同様の 2.6.0 Apache License 2.0
- slab 0.4.9 <https://github.com/carllerche/slab> MIT License
- snafu 0.7.5 <https://crates.io/crates/snafu> MIT ライセンスまたは Apache ライセンス 2.0
- snafu-derive 0.7.5 <https://crates.io/crates/snafu-derive> : (MIT ライセンスまたは Apache ライセンス 2.0)
- spm_precompiled 0.1.4 https://github.com/huggingface/spm_precompiled : Apache License 2.0
- sptr 0.3.2 <https://crates.io/crates/sptr> MIT ライセンスまたは Apache ライセンス 2.0
- srijs/rust-crc32fast 1.4.2 : Apache License 2.0
- stable_deref_trait 1.2.0 https://crates.io/crates/stable_deref_trait : (MIT ライセンスまたは Apache ライセンス 2.0)
- starkat99/half-rs 1.8.3 MIT ライセンスまたは Apache ライセンス 2.0
- starkat99/half-rs 2.4.1 MIT ライセンスまたは Apache ライセンス 2.0
- Stebalien/tempfile 3.13.0 <http://stebalien.com/projects/tempfile-rs> : Apache License 2.0
- steffengy/schannel-rs 0.1.26 : MIT ライセンス
- stringprep 0.1.5 <https://crates.io/crates/stringprep> : (MIT ライセンスまたは Apache ライセンス 2.0)
- strum-rs 0.26.3 <https://crates.io/crates/strum> : MIT ライセンス
- strum-rs 0.26.4 <https://crates.io/crates/strum> : MIT ライセンス
- 微妙な 2.6.1 <https://crates.io/crates/subtle> BSD 3 条項の「新規」または「改訂」ライセンス
- attempts-color 2.1.0 <https://crates.io/crates/supports-color> : Apache License 2.0
- attempts-color 3.0.1: <https://crates.io/crates/supports-color> : Apache License 2.0
- syn 1.0.109 <https://github.com/dtolnay/syn> : Apache License 2.0
- syn 2.0.87 <https://github.com/dtolnay/syn> MIT ライセンスまたは Apache ライセンス 2.0
- sync_wrapper 1.0.1 https://docs.rs/sync_wrapper : Apache License 2.0
- syn Structure 0.13.1 <https://crates.io/crates/synstructure> : MIT License
- sysctl 0.5.5 MIT ライセンス
- sysinfo-rs 0.30.13 <https://github.com/GuillaumeGomez/sysinfo> : MIT License
- sysinfo-rs 0.32.0 <https://github.com/GuillaumeGomez/sysinfo> MIT License
- system-configuration 0.6.1 MIT License OR Apache License 2.0
- system-configuration-sys 0.6.0 <https://crates.io/crates/system-configuration-sys> : (MIT ライセンスまたは Apache ライセンス 2.0)
- system-deps 6.2.2 <https://crates.io/crates/system-deps> MIT ライセンスまたは Apache ライセンス 2.0
- tafia/quick-xml 0.36.2 : MIT ライセンス

- tailhook/humantime 2.1.0 <https://github.com/tailhook/humantime> : (MIT ライセンスまたは Apache ライセンス 2.0)
- tailhook/quick-error 1.2.3 <http://github.com/rossmacarthur/quick-error2> : (MIT ライセンスまたは Apache ライセンス 2.0)
- tailhook/quick-error 2.0.1 <http://github.com/rossmacarthur/quick-error2> : (MIT ライセンスまたは Apache ライセンス 2.0)
- タップ 1.0.1 <https://github.com/myrrlyn/tap> : MIT License
- target-lexicon 0.12.16 <https://crates.io/crates/target-lexicon> 例外を含む Apache License 2.0
- tar (Rust) 0.4.43 <https://github.com/alexcrichton/tar-rs> : (MIT ライセンスまたは Apache ライセンス 2.0)
- TedDriggs/darling 0.20.10 : MIT License
- TedDriggs/ident_case 1.0.1 https://github.com/TedDriggs/ident_case.git : (MIT ライセンスまたは Apache ライセンス 2.0)
- thiserror 1.0.68 MIT License OR Apache License 2.0
- thiserror-impl 1.0.68 MIT ライセンスまたは Apache ライセンス 2.0
- tiff 0.9.1 <https://crates.io/crates/tiff> : MIT License
- tinystr <https://crates.io/crates/tinystr> : (MIT ライセンスまたは Apache ライセンス 2.0)
- tinyvec 1.8.0 <https://crates.io/crates/tinyvec> zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- tinyvec_macros 0.1.1 https://crates.io/crates/tinyvec_macros zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- トークナイザー 0.20.3 <https://github.com/epwalsh/tokenizers> Apache License 2.0
- tokio 1.41.0 <https://tokio.rs> : MIT License
- tokio-macros 2.4.0 <https://tokio.rs> : MIT License
- tokio-native-tls 0.3.1 <https://crates.io/crates/tokio-native-tls> : MIT ライセンス
- tokio-rs/bytes 1.8.0 <https://github.com/carllerche/bytes> : MIT ライセンス
- tokio-rustls 0.26.0 <https://github.com/tokio-rs/tls> : (MIT ライセンスまたは Apache ライセンス 2.0)
- tokio-util 0.7.12 <https://tokio.rs> : MIT ライセンス
- toml_datetime 0.6.8 <https://github.com/toml-rs/toml> : (MIT ライセンスまたは Apache ライセンス 2.0)
- toml-rs 0.8.19 <https://crates.io/crates/toml> MIT ライセンスまたは Apache ライセンス 2.0
- tower-rs/tower 0.3.3 <https://github.com/tower-rs/tower> MIT ライセンス
- トレース 0.1.40 <https://tokio.rs> MIT License
- tracing-attributes 0.1.27 <https://tokio.rs> : MIT License
- tracing-core 0.1.32 <https://tokio.rs> : MIT License

- ug 0.0.2 <https://crates.io/crates/ug> MIT ライセンスまたは Apache ライセンス 2.0
- unarray 0.1.4 <https://crates.io/crates/unarray> MIT ライセンスまたは Apache ライセンス 2.0
- unescape 0.1.0 <https://crates.io/crates/unescape> : MIT License
- unicode_ locations 0.1.1 https://crates.io/crates/unicode_categories : MIT License OR Apache License 2.0
- unicode-ident 1.0.13 : (Unicode License Agreement Data Files and Software (2016) AND (MIT License OR Apache License 2.0))
- unicode-normalization-alignments 0.1.12 <https://github.com/n1t0/unicode-normalization> : (MIT License OR Apache License 2.0)
- unicode-org/icu4x 0.1.4 <https://github.com/unicode-org/icu4x#readme> Unicode License Agreement - データファイルとソフトウェア 2016
- unicode-org/icu4x 1.5.1 <https://github.com/unicode-org/icu4x#readme> Unicode License v3
- unicode-org/icu4x <https://github.com/unicode-org/icu4x#readme> Unicode License v3
- unicode-properties 0.1.3 <https://github.com/unicode-rs/unicode-properties> : (MIT License OR Apache License 2.0)
- unicode-rs/unicode-normalization 0.1.24 <https://github.com/unicode-rs/unicode-normalization> : (MIT ライセンスおよび Apache ライセンス 2.0)
- unicode-rs/unicode-segmentation 1.12.0 <https://github.com/unicode-rs/unicode-segmentation> : (MIT License OR Apache License 2.0)
- unicode-rs/unicode-width 0.1.14 <https://github.com/unicode-rs/unicode-width> : (MIT License OR Apache License 2.0)
- ureq 2.10.1 MIT ライセンスまたは Apache ライセンス 2.0
- utf16_iter 1.0.5 https://docs.rs/utf16_iter/ : (MIT ライセンスまたは Apache ライセンス 2.0)
- utf8_iter 1.0.4 https://docs.rs/utf8_iter/ MIT ライセンスまたは Apache ライセンス 2.0
- uuid-rs/uuid 1.11.0 <http://docs.rs/uuid> : (MIT ライセンスまたは Apache ライセンス 2.0)
- vcpkg-rs 0.2.15 <https://docs.rs/vcpkg> MIT ライセンスまたは Apache ライセンス 2.0
- version-compare 0.2.0 <https://github.com/timvissee/version-compare> MIT License
- wait-timeout 0.2.0 <https://github.com/alexcrichon/wait-timeout> MIT ライセンスまたは Apache ライセンス 2.0
- wasi 0.11.0+wasi-snapshot-preview1 MIT ライセンスまたは Apache ライセンス 2.0 例外または Apache ライセンス 2.0
- wasite 0.1.0 <https://github.com/ardaku/wasite/blob/stable/CHANGELOG.md> : (MIT ライセンスまたは Apache ライセンス 2.0 または Boost ソフトウェアライセンス 1.0)
- wasm-bindgen 0.2.95 <https://github.com/rustwasm/wasm-bindgen> : (MIT ライセンスまたは Apache ライセンス 2.0)

- wasm-bindgen 0.3.72 <https://github.com/rustwasm/wasm-bindgen> : (MIT ライセンスまたは Apache ライセンス 2.0)
- wasm-bindgen 0.4.45 <https://github.com/rustwasm/wasm-bindgen> : (MIT ライセンスまたは Apache ライセンス 2.0)
- wasm-bindgen-macro 0.2.95 <https://rustwasm.github.io/wasm-bindgen/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- wasm-bindgen-macro-support 0.2.95 <https://rustwasm.github.io/wasm-bindgen/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- wasm-bindgen-shared 0.2.95 <https://rustwasm.github.io/wasm-bindgen/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- wasm-streams 0.4.2 MIT ライセンスまたは Apache ライセンス 2.0
- webpki-roots 0.26.6 <https://github.com/ctz/webpki-roots> Mozilla Public License 2.0
- weezl 0.1.8 <https://crates.io/crates/weezl> MIT ライセンスまたは Apache ライセンス 2.0
- whoami 1.5.2 MIT ライセンスまたは Apache ライセンス 2.0 または Boost ソフトウェアライセンス 1.0
- winapi-i686-pc-windows-gnu 0.4.0 <https://crates.io/crates/winapi-i686-pc-windows-gnu> : (MIT ライセンスまたは Apache ライセンス 2.0)
- winapi-rs 0.3.9 <https://crates.io/crates/winapi> MIT ライセンスおよび Apache ライセンス 2.0
- winapi-util 0.1.9 <https://github.com/BurntSushi/winapi-util> MIT ライセンスまたは アンライセンス
- winapi-x86_64-pc-windows-gnu 0.4.0 https://crates.io/crates/winapi-x86_64-pc-windows-gnu : (MIT ライセンスまたは Apache ライセンス 2.0)
- windows_aarch64_gnullvm 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_aarch64_gnullvm 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows_aarch64_msvc 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_aarch64_msvc 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- Windows-core 0.52.0 MIT ライセンスまたは Apache ライセンス 2.0
- Windows-core 0.57.0 MIT ライセンスまたは Apache ライセンス 2.0
- windows_i686_gnu 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_i686_gnu 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows_i686_msvc 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_i686_msvc 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows-implement 0.57.0 <https://crates.io/crates/windows-implement> : (MIT ライセンスまたは Apache ライセンス 2.0)
- windows-interface 0.57.0 <https://crates.io/crates/windows-interface> : (MIT ライセンスまたは Apache ライセンス 2.0)

- windows-sys 0.48.0 MIT ライセンスまたは Apache ライセンス 2.0
- windows-sys 0.52.0 MIT ライセンスまたは Apache ライセンス 2.0
- windows-sys 0.59.0 MIT ライセンスまたは Apache ライセンス 2.0
- windows-targets 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows-targets 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_gnu 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_gnu 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_gnullvm 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_gnullvm 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_msvc 0.48.5 MIT ライセンスまたは Apache ライセンス 2.0
- windows_x86_64_msvc 0.52.6 MIT ライセンスまたは Apache ライセンス 2.0
- winnow 0.6.20 MIT ライセンス
- write16 1.0.0 <https://docs.rs/write16/> MIT ライセンスまたは Apache ライセンス 2.0
- 書き込み可能 <https://crates.io/crates/writeable>
- wyz 0.5.1 <https://myrrlyn.net/crates/wyz> MIT ライセンス
- yansi-term 0.1.2: <https://github.com/botika/yansi-term> : MIT License
- yoke 0.7.4 <https://crates.io/crates/yoke> Unicode License v3
- yoke-derive 0.7.4 <https://crates.io/crates/yoke-derive> Unicode License v3
- zerocopy 0.7.35 : (MIT ライセンスまたは BSD 2-clause “Simplified” License OR Apache License 2.0)
- zerocopy-derive 0.7.35 MIT ライセンスまたは BSD 2 条項「簡素化」ライセンスまたは Apache ライセンス 2.0
- zeroize 1.8.1 <https://github.com/iqlusioninc/crates/> : (MIT ライセンスまたは Apache ライセンス 2.0)
- zerovec <https://crates.io/crates/zerovec>
- zerovec-derive 0.10.3 <https://crates.io/crates/zerovec-derive> Unicode License v3
- zip 1.1.4 <https://crates.io/crates/zip> : MIT License
- zune-core 0.4.12 <https://github.com/etemesi254/zune-image/tree/dev/zune-core> zlib ライセンスまたは MITLicense または Apache ライセンス 2.0
- zune-inflate 0.2.54 <https://github.com/etemesi254/zune-image/tree/main/zune-inflate> zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0
- zune-jpeg 0.4.13 <https://crates.io/crates/zune-jpeg> zlib ライセンスまたは MIT ライセンスまたは Apache ライセンス 2.0

3.6 Vector Engine (PGvector)

3.6.1 Vector Engine

Vector Engine は EDB Postgres® AI EDB PG AI の一部であり、Postgres データベースで直接高度な **ベクトル検索** および **ハイブリッド検索** 機能を有効にします。

オープンソースの **PGvector** 拡張機能を活用して、高次元ベクトルデータの効率的なストレージと類似性検索を提供します。これは、**検索拡張生成 RAG**、**セマンティック検索**、およびインテリジェントアプリケーションの基本的な機能です。

PGvector は、オープンソースコミュニティによって維持される独立した拡張機能です。EDB PG AI は、スタックの一部として PGvector を活用します。

- Pipelines and data preparation は、埋め込みモデルを使用してベクター列を自動的に入力できます。
- *Gen AI Builder* およびナレッジベースは、PGvector による取得を使用します。
- PGvector を独立して使用することもできます。これは独自のコンポーネントではありません。

3.6.2 Postgres で Vector Search を使用する理由

従来のデータベースは、構造化データとトランザクションワークロードに優れています。AI 搭載アプリケーションは、テキスト、画像、音声などの非構造化コンテンツの数値表現 **ベクトル埋め込み** への依存を高めています。

Vector Engine では、次のことができます。

- ベクター埋め込みを **Postgres クラスタ** に保存します — 別個のベクターデータベースは必要ありません
- **近似最近傍 ANN** インデックスを使用して、類似度によってコンテンツを検索および取得します
- 構造化データと非構造化データにわたるハイブリッドクエリのために **ベクトル検索 + SQL 検索** を組み合わせます
- **Power Sovereign Gen AI** — Postgres エステートに埋め込み、メタデータ、および結果を維持する
- Postgres データプラットフォームの一部としての AI 検索と取得の監査と管理

3.6.3 Vector Engine を使用するのは誰ですか？

Vector Engine は次の目的に設計されています。

- Postgres をベースに RAG パイプラインとセマンティック検索を構築する **データチーム**
- **Developers** 既存の Postgres データと統合する Gen AI アシスタントを構築
- **分析チーム** ベクトルベースの検索パターンで BI とアナリティクスを強化
- 埋め込みと結果を管理する必要がある **ソブリン AI** ソリューションを設計する **アーキテクト**

アプリケーションが **ベクターエンベディング** を使用し、Postgres にベクターとソースデータの両方を保持したい場合 — Vector Engine は、スケーラブルな実稼働対応のパスを提供します。

3.6.4 それは何を提供しますか？

その中核として、Vector Engine — PGvector を使用して — は以下を追加します。

- **vector** 列タイプ — 高次元ベクトルデータを保存するため
- 距離ファンクション
- **vector <-> vector** — ユークリッド距離 L2
- **vector <#> vector** — 内積
- **vector <=> vector** — コサイン距離
- 近似最近傍 ANN インデックスのサポート
- IVFFlat インデックスタイプ
- 精度と速度のバランスをとるための **lists** および **probes** を調整可能

PGvector 自分自身はオープンソースです — EDB PG AI はそれを使用して以下を強化します。

- ナレッジベースでの取得 *Gen AI Builder* 経由
- パイプラインベースの RAG パイプラインでのベクトル検索
- Postgres のハイブリッド AI + SQL クエリパターン

3.6.5 Vector エンジンの仕組み

1. Postgres データベースに PGvector をインストールして有効にします。

```
CREATE EXTENSION IF NOT EXISTS vector;
```

1. テーブルにベクター列を定義します。

```
ALTER TABLE documents ADD COLUMN embedding vector(768);
```

1. 高速類似性検索のためのインデックスベクトル列

```
CREATE INDEX ON documents USING ivfflat (embedding vector_l2_ops) WITH (lists = 100);
```

Pipelines and data preparation を使用してエンベディングを生成

または、外部埋め込みモデルを使用し、ベクトルを手動で挿入します

1. 類似性検索を使用したクエリー

```
SELECT id, content
FROM documents
ORDER BY embedding <-> [0.12, 0.34, 0.56, ...]::vector
LIMIT 5;
```

3.6.6 Vector Engine を使用する必要がある場合

次の場合に Vector Engine を使用します。

- 独自のデータの基礎を必要とする Gen AI または AI 搭載アプリケーションを構築している
- ドキュメント、Web ページ、ナレッジベース全体でセマンティック検索を実行したい
- ソブリン AI アーキテクチャが必要です — エンベディングとソースデータが完全に Postgres クラスター内にあります
- 統合データプラットフォームでハイブリッド構造化+非構造化検索をサポートしたい
- EDB PG AI および AI Accelerator パイプラインを使用して RAG パイプラインを開発している

3.6.7 ユースケースの例

- Enterprise RAG Assistants — ナレッジベースの取得と LLM を組み合わせます
- 内部セマンティック検索 — 複数のリポジトリにわたって企業コンテンツを検索します
- レコメンドシステム — コンテンツまたは製品の類似性検索
- ビジョン/オーディオの類似性 — 画像またはオーディオの埋め込み検索

3.6.8 AI Accelerator パイプラインおよび Gen AI Builder との統合

AI アクセラレーター [Pipelines](#) を使用すると、次のことが簡単にできます。

AI Factory [Pipelines](#) では、次のことが簡単にできます。

- ドキュメントをベクトル化し、Postgres ベクター列を設定します
- RAG パイプラインを自動化およびオーケストレーションする
- ベクトル化にサポートされている埋め込みモデルを使用する
- [Pipelines](#) は、ナレッジベースを構築するときに PGvector ベースの取得を使用します。

ベクトル化にサポートされている埋め込みモデルを使用する

[Pipelines](#) は、ナレッジベースを構築するときに PGvector ベースの取得を使用します。

ナレッジベースは Postgres ベクター列に埋め込みをストアします

類似性検索に PGvector ファンクションを使用した RAG フロークエリー

3.6.9 次のステップ

- AI アクセラレーターパイプラインについて学ぶ
- ベクトル化パターンのパイプラインプリペアラーの例を探索する
- サポートされている埋め込みモデルの確認
- Gen AI Builder ナレッジベースと RAG パイプラインを探索する
- PGvector の完全なドキュメントを参照してください

EDB PG AI Vector Engine は、データ、埋め込み、コントロールを使用した強力なソブリン AI ソリューションを Postgres エステートに直接構築するのに役立ちます。

3.7 AI Factory Release Notes

3.7.1 AI Factory 1.3 release notes

発売日 2025 年 6 月 17 日

EDB Postgres AI Factory v1.3 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。

リリースハイライト

- **ソブリン AI & 分析プラットフォーム AI Factory** 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データと AI ファクトリー** 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。

展開オプション

EDB Postgres AI Factory v1.3 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。

ソブリンデータと AI ファクトリー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、ソブリンデータおよび AI ファクトリーは、完全に統合され、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 AI Factory 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 Factory 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。

新機能と機能強化

ソブリン AI & 分析プラットフォーム

AI Factory v1.3 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、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。AI Factory 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[®] 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 Factory v1.3 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。

可観測性、モニタリング、インサイト

v1.3 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み最適化のヒントを受信します。
- **クラスタースコアカード** 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 AI Factory に接続して、クエリ診断、推奨事項、モニタリングメトリックを収集します。

サステナビリティ ESG レポートプレビュー

EDB AI Factory 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 CO₂ 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 CO₂/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コンソール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。

プラットフォームの改善

Area	Highlights
Deployment options	New Sovereign Data and AI 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 console, 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.

リソース

- [Installing AI Factory](#)
- [Using AI Factory](#)
- [AI Factory Overview](#)
- [Analytics Lakehouse Architecture](#)
- [Power & ESG Reporting](#)
- [Release Notes Archive](#)

リリース日 2025 年 6 月 30 日