Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box

発行日：2024 年 7 月

執筆協力：

NUTANIX

Cisco Validated Design プログラムについて

Cisco Validated Design（CVD）プログラムは、お客様による信頼性の高い、確実かつ速やかな展開を容易にするために、デザイン、テスト、および文書化されたシステムおよびソリューションで構成されています。詳細については、http://www.cisco.com/go/designzone を参照してください。

エグゼクティブサマリー

Cisco Validated Design（CVD）は、お客様への展開を円滑化することを念頭に置いて設計、テスト、および文書化されたシステムとソリューションで構成されています。これらの設計では、お客様のビジネス ニーズに対応するために開発されたソリューションのポートフォリオに幅広いテクノロジーと製品を組み込んでいます。

生成型人工知能（生成人工知能）は、あらゆる業界で変革をもたらす力として存在し、さまざまなユースケースでイノベーションを推進しています。機会があるにもかかわらず、企業の環境に生成型人工知能を統合することには固有の課題があります。オンプレミスの人工知能ソリューションでは、内部データを効果的に活用することが重要です。適切なコンピューティング リソースを備えた適切なインフラストラクチャを構築することが重要です。スタック全体の可視性とモニタリングは、運用の観点から重要です。

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box ソリューションは、Nutanix GPT-in-a-Box の基盤となるインフラストラクチャを展開するための規範的な手順を提供することで、生成人工知能 の導入の複雑さを解消します。このソリューションは、Cisco®サーバーとSaaS運用を Nutanixソフトウェアと組み合わせ、最も一般的な大規模言語モデル（LLM）を利用して、データセンターからエッジまでの人工知能イニシアチブを簡素化してすぐに開始できる、完全に検証された人工知能対応プラットフォームを生成します。

このドキュメントでは、オンプレミスの生成人工知能アプリケーションを展開するの、Cisco Compute Hyperconverged with Nutanix 上の GPT-in-a-Box の Cisco 検証済みデザインおよび展開について説明します。このソリューションでは、生成人工知能向けの革新的で回数変更可能かつ人工知能な生成型事前トレーニング済みトランスフォーマ（GPT）ソリューションの導入をサポートする設計の概要を示し言語。

このソリューションでは、検索拡張生成（RAG）に特に焦点を当てた、組織の内部または独自のナレッジベースを活用して、オンプレミスの生成人工知能 の一般的なユースケースについて説明します。

ソリューションの概要

この章の内容は、次のとおりです。

はじめに

生成人工知能は、ダイナミック マーケティング格納ファイルからインタラクティブなリモート対応アシスタントやチャットボットまで、業界を再形成しています。ただし、企業内でその可能性を引き出すには課題があります。これらのアプリケーションでは、組織の内部データを効果的に使用することが重要です。堅牢なインフラストラクチャ、スタック全体のオブザーバビリティ、最適化されたモデルの展開とサービス、高可用性、およびスケーリングはほとんどありません。

このソリューションは、企業が生成人工知能用の Generative Pretrained Transformer（GPT）ソリューションを設計および展開するして、組織が選択した人工知能大型言語モデル（LLM）とそれを活用するアプリケーションをプライベートに実行および管理する方法に焦点を当てています。このソリューションでは、参照ユースケースとして、拡張生成の取得に焦点を当てています。

ハードウェアとソフトウェアのコンポーネントが統合されているため、お客様はソリューションを迅速かつ経済的に展開するできると同時に、同様のソリューションをゼロから調査、設計、構築、および展開することに伴う多くのリスクを排除できます。

このドキュメントの目的

このドキュメントでは、Cisco Compute Hyperconverged with Nutanix での GPT-in-a-Boxソリューションの Cisco 検証済みデザインと展開の詳細について説明します。このドキュメントで説明するソリューションは、組織の内部ドキュメントとオンプレミスのデータによって強化され、運用の簡素化を確保することで、会話型のアドレスのようなエクスペリエンスを目的とした、予測可能でスケーラブルで安全な高性能のクラウドネイティブソリューションの設計、参照アーキテクチャ、および導入に対応します。簡単に操作できます。

この検証済みの設計は、サポートされている GPT 構成の一例にすぎません。GPT ソリューションはさまざまな方法で設計および構築できます。また、CVD のベストプラクティスに従って、この特定の構成から逸脱することができます。

対象読者

このドキュメントの対象対象者には、CTO や CIO などの IT 意思決定者、IT アーキテクト、および生成人工知能システムおよびアプリケーションの設計、展開、およびライフ周期管理に取り組んでいる、またはそれらに関心があるお客様が含まれます。

ソリューションの概要

GPT-in-a-Box は、人工知能対応インフラストラクチャを構築するために必要なすべてを含む新しいターンキー ソリューションです。AI アプリケーションは、GPT-in-a-Box 上に簡単に展開できます。

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box ソリューションは、以下を組み合わせた基準アーキテクチャです。

- Nutanix GPT-in-a-Box ソフトウェア定義型ソリューション

- Cisco コンピューティング ハイパーコンバージド C シリーズ サーバ
- UCS 管理モードまたは Intersight スタンドアロン モードの 2x NVIDIA L40S GPU を搭載した HCIAF240C M7 All-NVMeサーバ
- ソリューション ソフトウェア コンポーネントを展開および管理するための Nutanix Prism Central
- 最も一般的な大規模言語モデルの範囲

GPT-in-a-Box ソリューション ソフトウェア コンポーネントには、次のものが含まれます。

- Kubernetes プラットフォームおよび Nutanix Unified Storage（NUS）として NKE を使用
- NAI-LLM 推論エンドポイントを活用
- Nutanix オブジェクト イベント通知との統合

ソリューションの一部のハイライトは次のとおりです。

- オンプレミスの人工知能ソリューションを迅速に設計、サイジング、展開するできる人工知能対応プラットフォーム
- 市販の人工知能アプリケーションを迅速に展開するか、開発者が独自のアプリケーションを構築できるようにします。
- Nutanix Cloud Platform ハイパーコンバージド インフラストラクチャとグラフィック プロセッシング ユニット（GPU）を使用したシングル クラウド オペレーティング モデル
- 総合的なデータ管理、セキュリティ、プライバシー、復元力を実現する Nutanix Unified Storage（NUS）
- 一般的な人工知能/ML フレームワークのサポート
- LLM 選択の自由

拡張生成の取得：概念とコンポーネント

この章の内容は、次のとおりです。

この章では、モデル開発ワークフロー、推論の課題、ユース ケースなど、生成人工知能のさまざまな概念について説明します。

生成人工知能とは

生成人工知能は人工知能の強力なブランチであり、企業が直面するさまざまな課題に対処する大きな可能性を秘めています。生成人工知能を使用すると、ユーザーとアプリケーションはさまざまな入力に基づいて新しい格納ファイルを迅速に生成できます。これらのモデルへの入出力には、テキスト、画像、音声、アニメーション、3D モデル、またはその他のタイプのデータを含めることができます。生成人工知能モデルの汎用性により、それらを活用するアプリケーションは、利用可能なデータと入力に基づいて複数のタスクを実行でき、単なるテキストとイメージの生成やチャットベースの Q&A を超えた機能を強化できます。

生成人工知能と従来の人工知能の比較

生成人工知能は、新しい格納ファイル、チャット応答、設計、合成データなどを作成できます。一方、従来の人工知能は、パターンの検出、意思決定、分析の強化、データの分類、および不正行為の検出に重点を置いています。

人工知能を使用して新しい格納ファイルを作成する価値を認識する組織が増えるにつれて、大規模言語モデル（LLM）やその他の生成モデルを検討しています。基盤モデルと呼ばれる事前トレーニングされた LLM が利用できるため、生成人工知能を採用する場合、従来の人工知能モデルと比較して必要な事前トレーニングが少なくなります。これにより、生産環境で人工知能アプリケーションを開発、実行、および保守する際のコストと時間を大幅に節約できます。

2023 年は、ChatGPT や Stable Diffusion などのモデルが導入された生成人工知能の年でしたが、このテクノロジーはしばらく前から開発が進んでいます。NVIDIA やその他の企業は、この分野で何年にもわたって研究と革新を続けてきましたので、現在に至るまでの助けとなっています。例としては、人のリアルなイメージを作成する StyleGAN（2018）や、リアルな風景になるフィンガーペイントスタイルのイメージを作成できる GauGAN（2019）などがあります。NVIDIA は、この調査に基づいた「Canvas」と呼ばれるアプリケーションをリリースしており、これらのテクノロジーはエコシステムパートナーによって広く使用されています。

生成人工知能推論とは

生成人工知能推論とは、トレーニングされた生成人工知能モデル（大規模な言語モデルと非大規模な言語モデル）を使用して、入力またはコンテキストのキューに基づいて新しいデータまたは格納ファイルを生成するプロセスを指します。推論中、モデルは学習した知識を適用して、トレーニングデータの直接的な繰り返しではなく、モデルによって生成される新しい作成物である出力を生成します。

推論プロセスは、実際のアプリケーションでモデルの生成機能を活用するために重要です。これにより、ユーザーは、特定の要件または制約に基づいてモデルの動作を入力またはガイドすることで、新しい出力を得ることができます。生成された格納ファイルは、さまざまなクリエイティブな目的、プロトタイピング、またはさまざまなドメインでの調査用のツールとして使用できます。

生成人工知能のコンテキストでの「推論」という用語は、次のような格納ファイルの生成に関連しています。

- テキストの生成
  - ストーリーテリング：生成モデルは、架空のストーリー、ナラティブ、または本の章全体を作成できます。
  - 詩と散文： 人工知能モデルは、詩的な文章、散文、またはクリエイティブな文章を生成できます。
  - ダイアログ：生成モデルを利用した会話型エージェントは、人間のようなダイアログを生成できます。
- イメージの生成
  - アーティスティックなクリエーション：Generative Adversarial Networks（GAN）は、視覚的に魅力的なアーティスティックなイメージを生成できます。
  - スタイルの転送：モデルは、画像をさまざまなアーティスティックなスタイルに変換できます。
  - 顔の合成：GAN は、実在しない人物のリアルな顔を作成できます。
- 作曲
  - メロディー生成： 人工知能モデルは、オリジナルのメロディーや音楽を作成できます。
  - ジャンル特定の音楽：生成モデルは、さまざまなスタイルを模倣して、特定のジャンルの音楽を作成できます。
- コードの生成
  - ソース コード： 人工知能モデルは、特定のタスクまたは説明に基づいてコード スニペットまたはプログラム全体を生成できます。
- 多言語対応
  - 多言語テキスト：OpenAI の GPT などのモデルは、複数の言語でテキストを生成できます。
  - 翻訳： 人工知能モデルは、コンテキストを維持しながら、ある言語から別の言語にテキストを翻訳できます。
- コンテンツの要約
  - テキスト サマリー：生成モデルは、大きなテキストブロックを簡潔で一貫したサマリーに要約できます。
- コンテンツの完了
  - 文の補完： 人工知能モデルは、コンテキストに適した方法で文または段落を完成させることができます。
  - テキストの拡張：生成モデルは、与えられたアイデアや概念を拡張できます。
- 製品の説明
  - eコマースの説明： 人工知能モデルは、 e-コマースの製品説明を生成できます。
- 科学文書
  - 研究論文の要約：モデルは、科学研究論文の要約または要約を生成できます。
- 会話型エージェント
  - チャットボットの応答： 人工知能を搭載したチャットボットは、会話中に自然言語で応答を生成できます。

大規模言語モデル

生成人工知能は、新しい独自の格納ファイルを生成するように設計されたモデルを含む広範なカテゴリです。この格納ファイルには、画像、テキスト、音声、ビデオなど、さまざまな形式があります。大規模言語モデルは、人間の言語を理解して生成するように設計された生成人工知能の特定のサブセットです。主に自然言語処理タスクに焦点を当てています。

大規模言語モデル（LLM）は、ディープ ラーニングの方法論を使用して人間の言語を理解して生成する自然言語処理モデルのクラスです。これらのモデルは、言語のパターン、構造、およびニュアンスを学習するために、大量のテキストデータでトレーニングされます。

LLM の注目すべき例の 1 つは、OpenAI によって開発された GPT（Generative Pre-trained Transformer）シリーズです。

大規模な言語モデルの主な機能は次のとおりです。

- **スケール：** LLM は、多くの場合、数千万から数十億に及ぶパラメータの数が多いことを特徴としています。これらのモデルの規模により、複雑な言語パターンをキャプチャし、多様でコンテキストに関連するテキストを生成できます。
- **事前トレーニング：** LLM は通常、特定のタスクに合わせて微調整される前に、大量のテキスト データで事前トレーニングされます。事前トレーニング中に、モデルは文の次の単語を予測したり、欠落している単語を塗りつぶしたりすることを学習します。これは、言語の幅広い理解を得るのに役立ちます。
- **トランスフォーマー アーキテクチャ**：GPT を含む LLM はトランスフォーマー アーキテクチャ上に構築され、連続的なデータの効率的な処理を可能にします。トランスフォーマーは、自己注意メカニズムを使用して文章内の単語間の関係をキャプチャし、コンテキストの理解を促進します。
- **伝達学習**：LLM は伝達学習を活用し、一般的な言語理解タスクの事前トレーニング中に得た知識を、最小限の追加トレーニングで特定のタスクに移行します。このアプローチにより、これらのモデルはさまざまな自然言語処理（NLP）アプリケーションで優れています。
- **微調整**：事前トレーニング後、テキスト分類、言語変換、自動要約などの特定のタスクに合わせて LLM を微調整できます。この微調整プロセスにより、ターゲット アプリケーションの微妙な違いにモデルが適応します。
- **多様なアプリケーション**：大規模言語モデルは、自然言語理解、テキスト生成、感情分析、マシン翻訳、質問応答、チャットボット開発など、範囲タスクにアプリケーションを提供します。

大規模言語モデルの開発により、自然言語処理の分野が大幅に進歩し、さまざまなドメインで人間のようなテキストを理解して生成できる高度な人工知能システムの作成が可能になりました。ただし、倫理的な考慮事項、トレーニング データのバイアス、および誤用の可能性は、これらのモデルの展開に関連する重要な考慮事項です。

モデル パラメータ

モデル パラメータは、トレーニング プロセス中にモデルが学習する内部変数または重みです。重みは、ニューラル ネットワークの入力特徴をスケーリングする係数です。LLM のコンテキストでは、これらの重みは、異なる層のニューロン間の接続の強度を決定します。たとえば、トランスフォーマー モデルでは、重みはアテンションメカニズムと入力シーケンスに適用される変換に関連付けられます。

LLM は、多くの場合、それぞれが重みとバイアスのセットを持つ複数のレイヤーで構成されます。トランスフォーマー アーキテクチャでは、これらの層に自己注意メカニズムとフィードフォワード ニュートラル ネットワークが含まれる場合があります。各レイヤのパラメータは、入力データのさまざまな側面をキャプチャします。

LLM のパラメータの総数は、複雑な言語パターンやニュアンスをキャプチャ キャパシティの重要な要素です。

生成人工知能ワークフロー

一般的な生成人工知能ワークフローは、すべての段階で簡潔で正確な技術的焦点を維持しながら、ビジネス目標に合わせることから始まります。

ビジネス戦略とユース ケースの定義：ビジネス目標に沿った生成人工知能の目標を定義します。

- 主なタスク
  - ユース ケースを特定します。
  - イメージ生成、テキスト生成、スタイル転送など、生成タスクを明確に定義します。
  - 目標と成功指標を確立します。

データの準備とキュレーション：高品質で適切に管理されたデータセットの可用性を確保します。

- 主なタスク
  - 生成モデルをトレーニングするための多様で代表的なデータセットを収集します。
  - データのクレンジングとラベル付け。
  - データの集約と前処理。
  - ローテーション、スケーリング、反転などの手法を使用して、トレーニングデータの多様性を高めます。
  - 必要に応じて、匿名化または合成データの生成を行います。
  - 効率的なデータ管理のための MLOps プラットフォームを活用します。

モデル トレーニング：効率的なトレーニングのために高速インフラストラクチャを利用します。

- 主なタスク
  - ゼロからトレーニングするか、事前トレーニング済みモデルを選択します。
  - 大量の計算リソースの割り当て。
  - 検証済みの高性能インフラストラクチャによるパフォーマンスの最適化。

モデルのカスタマイズ：微調整、プロンプト学習（プロンプト調整と P 調整を含む）、転送学習、強化学習。

- 主なタスク
  - 事前トレーニング済みのモデルを特定のビジネスニーズに適応させる。
  - 要件に基づいてカスタマイズ方法を導入する。

推論：継続的な生成のためにトレーニングされたモデルを展開して運用する。

- 主なタスク
  - 必要に応じてコンピューティングリソースをスケールアップまたはスケールアウトする。
  - 新しいデータとカスタマイズの機会に基づいて推論を反復する。
  - 推論パフォーマンスの継続的なモニタリング。
  - さらなるカスタマイズと微調整の機会の特定と最適化。

このワークフローでは、技術的な側面を強調し、インフラストラクチャの効率性、モデルのカスタマイズ技術、および推論フェーズでの継続的な最適化の重要性を強調しています。

拡張生成の取得（RAG）

この設計は、主に、企業における生成人工知能のアプリケーションである拡張生成の取得に焦点を当てています。RAG は、外部データを使用して LLM のコンテキストを拡張する LLM アプリケーションのクラスです。

大規模な言語モデルの制限

大規模な言語モデルの使用が急増すると、システムの顕著な弱点のいくつかが、企業スペースで使用する際のボトルネックになりました。これらの LLM は膨大な量のデータでトレーニングされますが、応答の生成に使用される情報は、トレーニング中に使用されるデータのみに制限されます。企業の人工知能チャットボットを例にとると、組織の内部データやサービスに関する特定の情報がないため、LLM を直接使用することはできません。ドメイン固有のデータまたは組織固有のデータがないため、企業アプリケーションでの使用がブロックされます。

これには次のような制約があります。

- 主なものは、幻覚です。LLM は、さまざまなトピックやドメインに関する流暢なテキストを生成できますが、事実を捏造する傾向があります。幻覚は、事実から逸脱した LLM の出力です。これは、軽微な不一致から完全に捏造または矛盾するステートメントまでに及びます。
- ナレッジ カットオフ日とその日以降に発生したイベントがあり、その情報はモデルでは使用できず、LLM は正確な回答を提供できません。
- モデルの多くは、言語タスクの汎用モデルです。ドメイン固有のデータがありません。

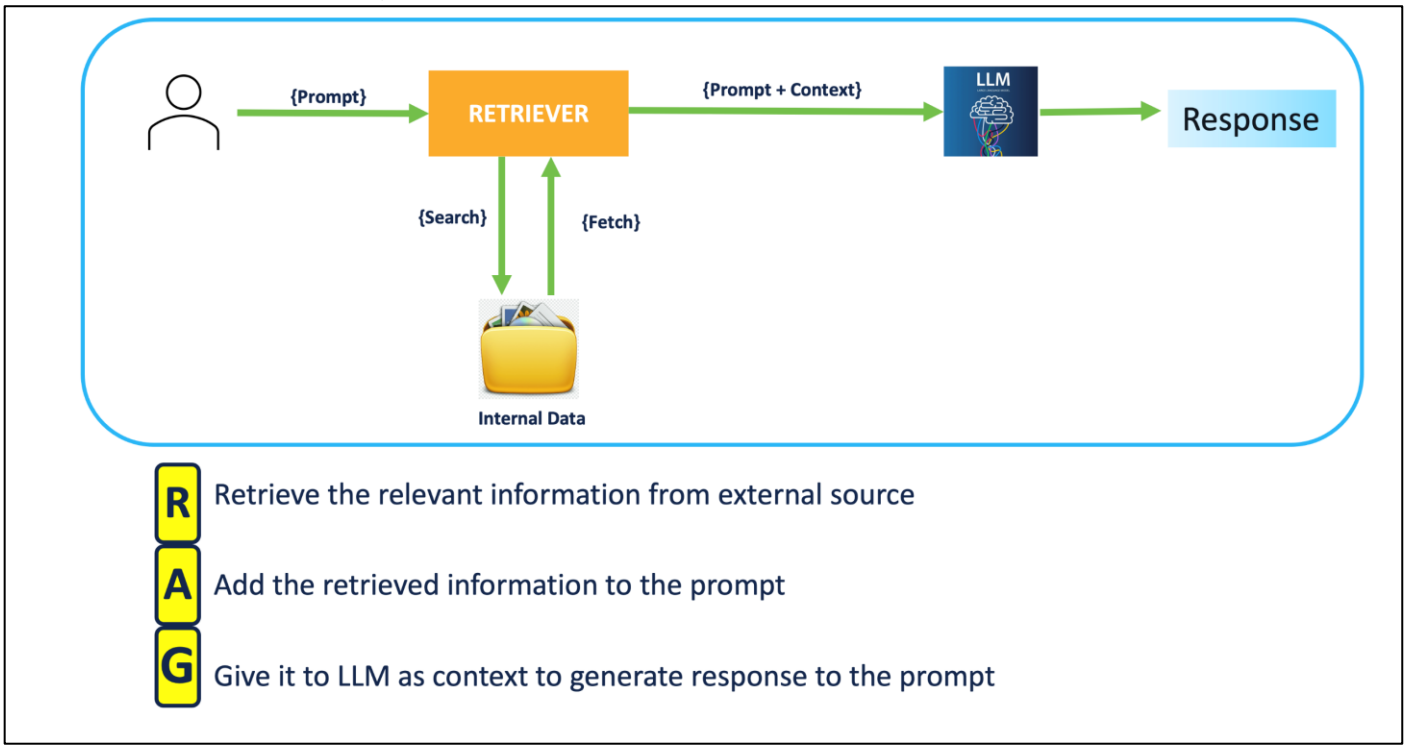
RAG はどのように役立つのか？

RAG は、LLM を企業データに接続することで、最新のドメイン固有の回答を生成します。これは、外部ソースから取得したダイナミック ドメイン固有のデータを使用して LLM の出力を最適化するために使用されるアーキテクチャです。

RAG パイプライン

図 1 に、RAG パイプラインの概要を示します。

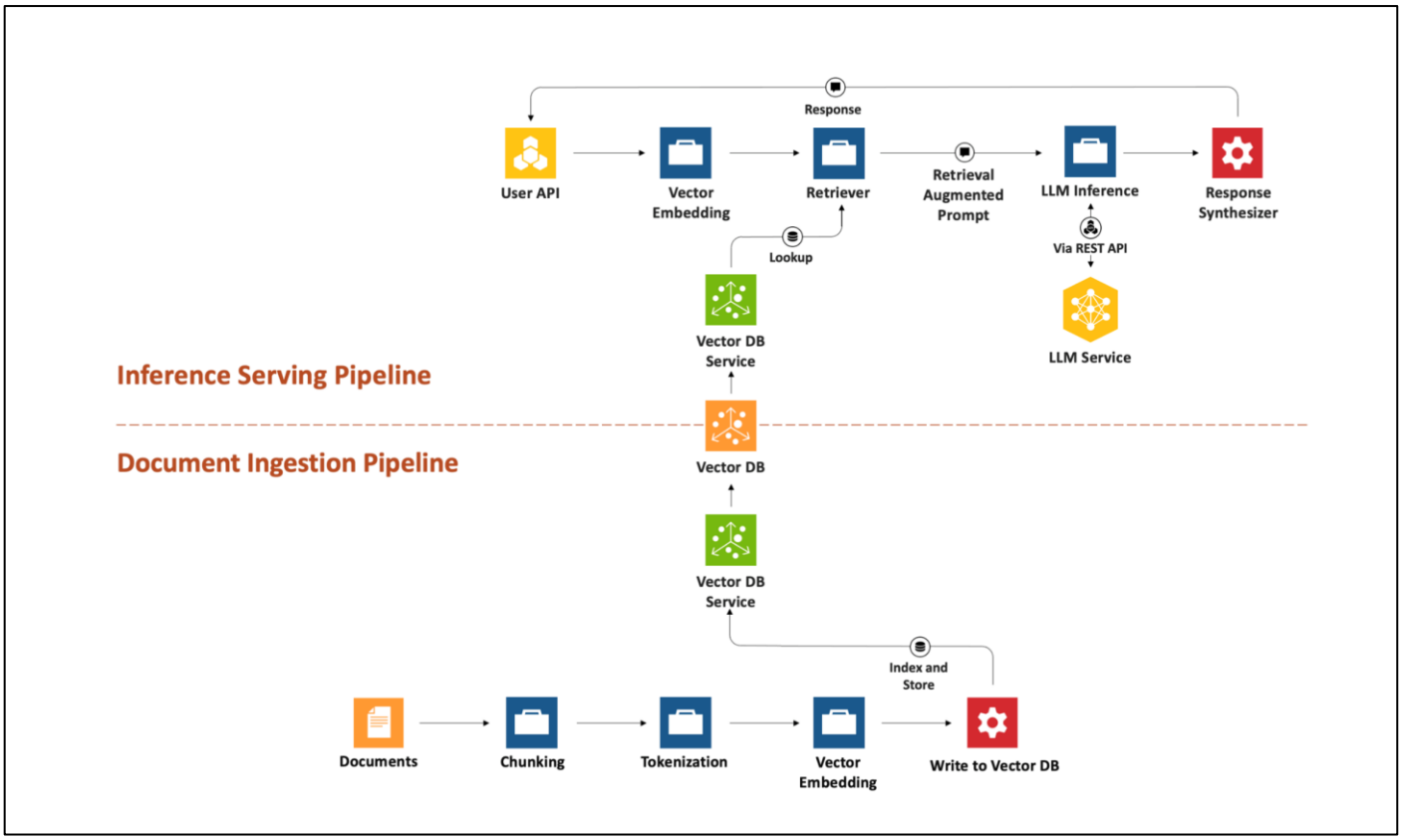
図 1.　　RAG パイプラインの概要



このパイプラインでは、プロンプト/クエリを入力すると、プロンプトに関連するドキュメント チャンクが検索され、システムに取得されます。取得された関連情報は、コンテキストとしてプロンプトに追加されます。LLM は、コンテキストでプロンプトへの応答を生成するように求められ、ユーザーは応答を受け取ります。

汎用 RAG アーキテクチャ

RAG は、情報取得コンポーネントと応答ジェネレータを組み合わせたエンドツーエンドのアーキテクチャです。

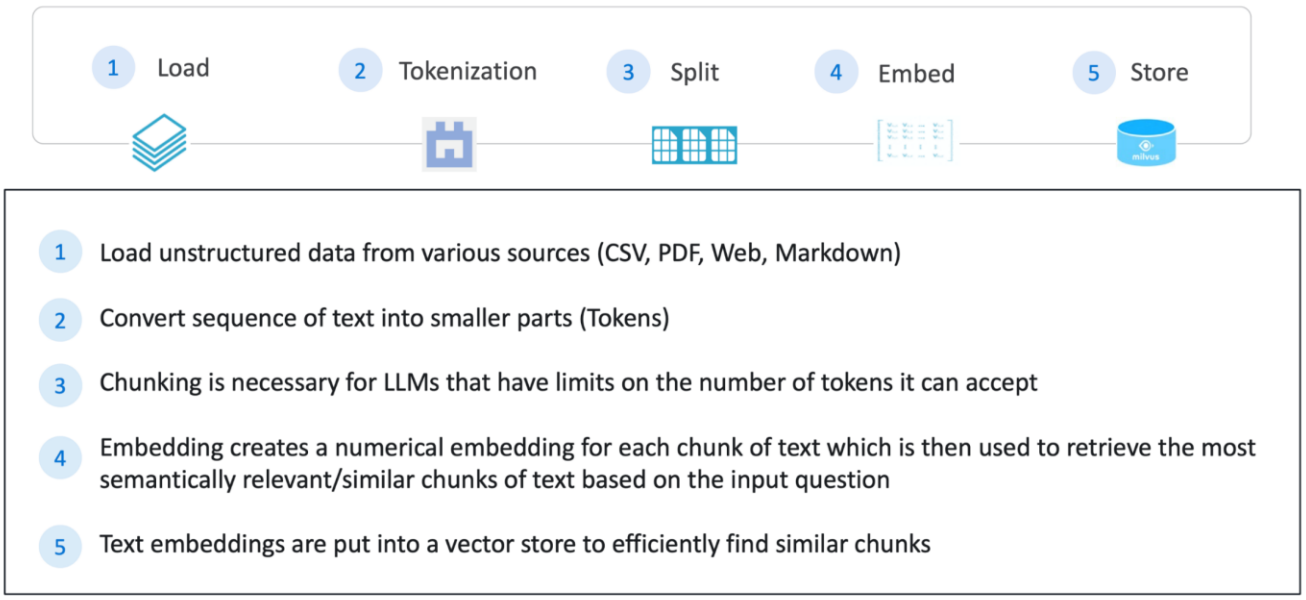


RAG は、ドキュメント取り込みパイプラインと推論提供パイプラインの 2 つのパイプラインに分割できます。

ドキュメント取り込みパイプライン

図 2 は、ドキュメントの取り込みパイプラインを示しています。

図 2.　　ドキュメント取り込みパイプラインの概要



ドキュメント取り込みパイプラインのプロセスは次のとおりです。

1. ドキュメント取り込みパイプラインの最初のステップは、データの読み込みです。さまざまなソースからの raw データが RAG システムに取り込まれます。これらのデータには、PDF、Word ドキュメント、 HTML、YouTube のトランスクリプトなどがあります。

2. 次はトークン化で、これらのテキストはトークンと呼ばれる小さな単位に分割されます。トークンは、LLM で実行される内容を測定する最小単位です。モデルはこれらのトークン間のセマンティック関係を理解することを学習するため、データを数値表現に変換する必要があります。

3. ドキュメントが読み込まれると、次のステップはチャンクです。これは、長いテキストを小さなセグメントに分割します。LLM にはコンテキスト ウィンドウに制限があるため、これが必要です。また、チャンクを小さくすると、インデックス作成と検索の高速化に役立ちます。

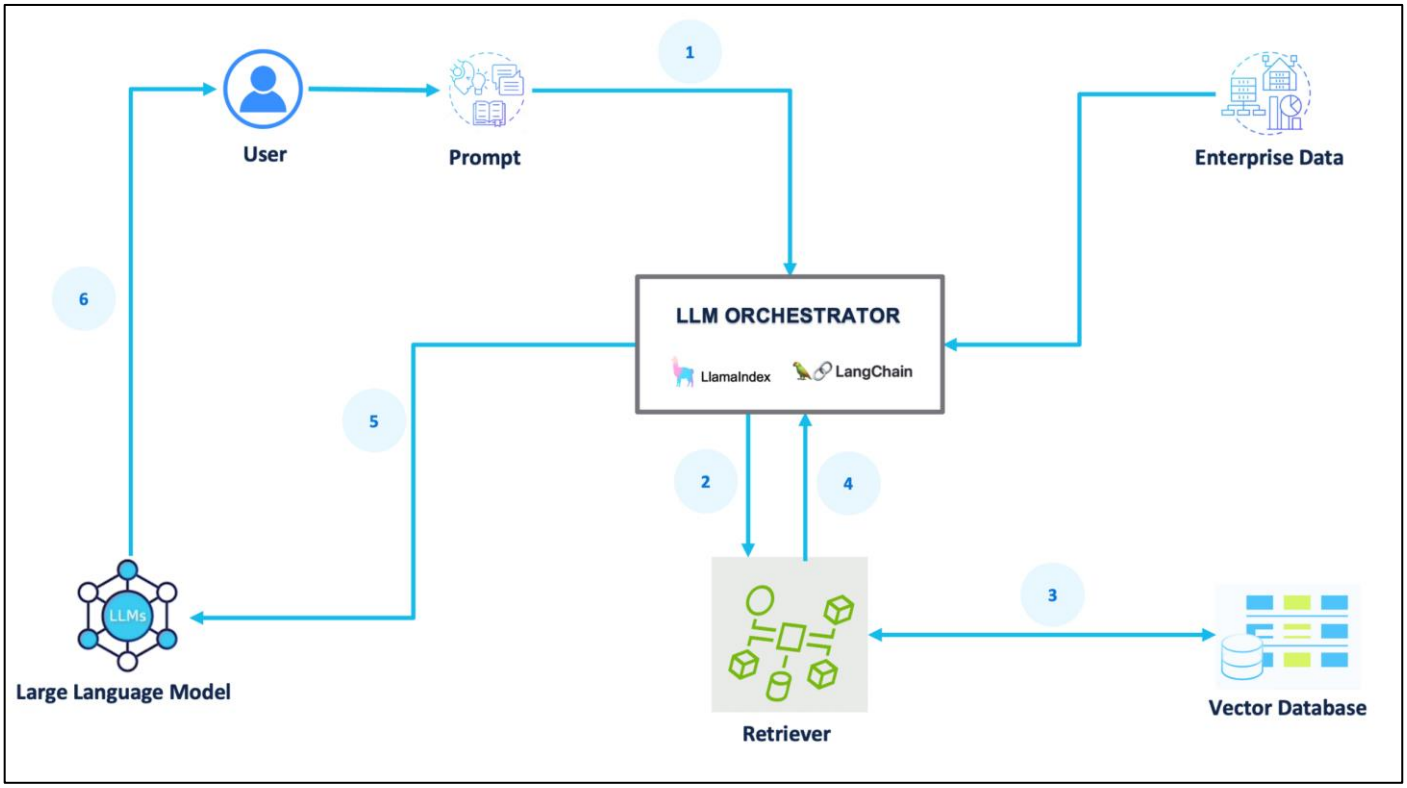
4. 次はベクトル埋め込みです。これは、自然言語処理で広く使用されている手法で、単語やフレーズを埋め込み次元と呼ばれる数値のベクトルとして表現します。ベクトルは単語間のセマンティック関係を捉えるように設計されています。つまり、意味が似ている単語は埋め込み空間内で互いに近いベクトルによって表されます。

5. 作成された埋め込みは、ベクトル データベースと呼ばれる特別なデータベースに保存されます。これらのベクトル データベースにより、高速検索と類似検索が可能になります。これにより、LLM は最も関連性が高く、コンテキストに適した情報にアクセスすることができるようになります。

推論サービス提供パイプライン

図 3 は、推論サービス提供パイプラインを示しています。

図 3.　　推論サービス提供パイプラインの概要



推論サービス提供パイプラインのプロセスは次のとおりです。

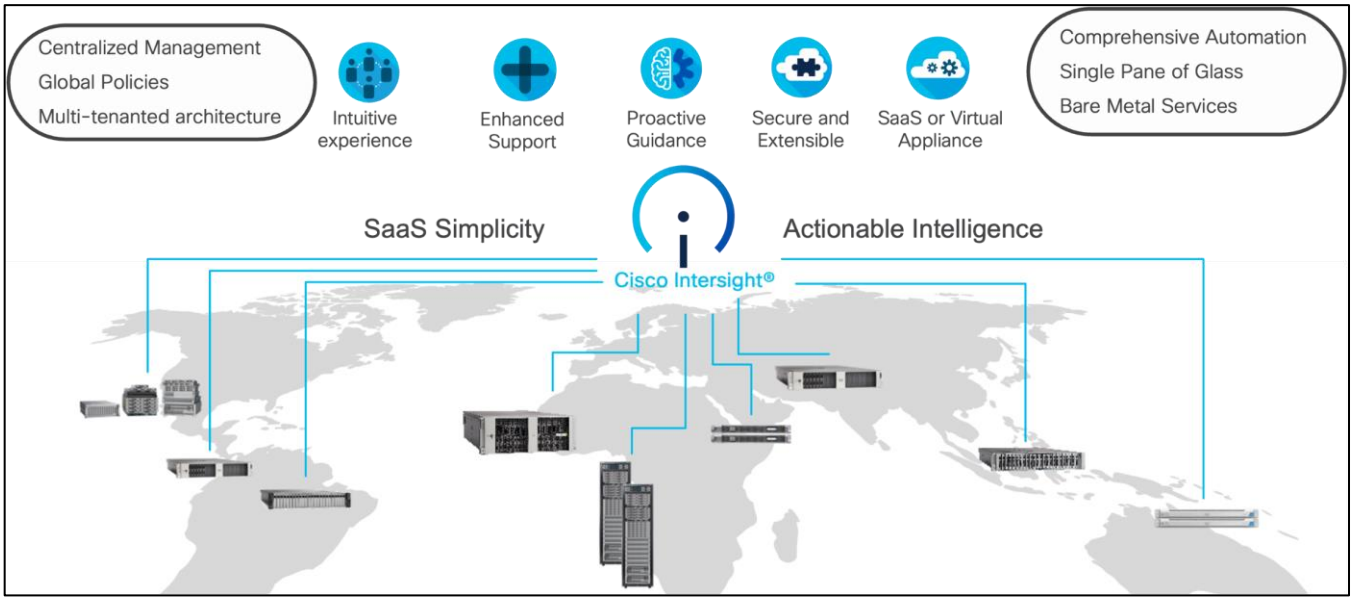
1. プロンプトが LLM オーケストレータに渡されます。
2. オーケストレータが取得者に検索クエリを送信します。
3. リトリーバは、ナレッジベースから関連情報を取得します。
4. リトリーバは、取得した情報をオーケストレータに送り返します。
5. オーケストレータは、コンテキストでプロンプトを拡張し、LLM に送信します。
6. LLM は、オーケストレータを使用してユーザーに表示される生成されたテキストで応答します。

テクノロジーの概要

この章の内容は、次のとおりです。

Cisco Intersight プラットフォーム

アプリケーションとデータがコア データ センターやエッジ ロケーションからパブリック クラウドへと分散するにつれて、中央管理プラットフォームが不可欠になっています。IT の俊敏性は、インフラストラクチャ リソースの統合ビューと一元化された運用なしでは困難です。Cisco Intersightは、ハイパーコンバージド向け Cisco Compute、Cisco UCS、および世界中に展開されているその他のサポート対象のサードパーティインフラストラクチャ向けに、クラウドホスト型の管理および分析プラットフォームを提供します。データセンター、ROBO、エッジ、およびコロケーション環境でインフラストラクチャを効率的に展開、管理、およびアップグレードする方法を提供します。



Cisco Intersight は次の機能を備えています。

- 影響なしの移行：組み込みコネクタ（Cisco HyperFlex、Cisco UCS）により、分岐リフト アップグレードなしで利点の利用を開始できます。
- SaaS/サブスクリプション モデル： SaaS モデルは、プラットフォームを管理するための管理オーバーヘッドなしで、世界中の何百ものサイトで一元化されたクラウドスケールの管理と運用を提供します。
- 強化されたサポート エクスペリエンス：ホスト型プラットフォームにより、Cisco はプラットフォーム全体の問題に対応するとともに、TAC でサポートされているプラットフォームにエクスペリエンスを拡大できます。
- 統合型管理：すべてのシステムとソリューションを管理するための単一の管理画面、一貫性のある運用モデル、エクスペリエンスを提供します。
- プログラマビリティ：ネイティブAPI、SDK、および一般的なDevOpsツールセットを使用したエンドツーエンドのプログラマビリティにより、インフラストラクチャを迅速かつ簡単に展開するおよび管理できます。

- 自動化の単一ポイント：Ansible、Terraform、およびその他のツールを使用した自動化は、管理するすべてのシステムに対して Intersight を介して実行できます。
- 推奨エンジン：マシン インテリジェンスと分析を活用した可視性、インサイト、アクションのアプローチにより、俊敏性と拡張性を備えたリアルタイムの推奨事項を提供します。推奨プラットフォームと Cisco のインストール ベース全体から取得したインサイトを利用して、顧客ごとにカスタマイズされた情報を提供します。

詳細については、 cisco.com の Cisco Intersight 製品ページを参照してください。

Cisco Intersight 仮想アプライアンスおよびプライベート仮想アプライアンス

Intersight.com で実行される SaaS 展開モデルに加え、オンプレミス オプションを別途購入できます。Cisco Intersight 仮想アプライアンスと Cisco Intersight プライベート仮想アプライアンスは、システム管理に関してデータの局所性やセキュリティの追加要件がある組織向けに用意されています。Cisco Intersight 仮想アプライアンスは、簡単に導入できる VMware Open Virtualization Appliance (OVA) または Microsoft Hyper-V Server 仮想マシンで Cisco Intersight プラットフォームの管理機能を提供し、社内から外部に持ち出されるシステムの詳細を制御できるようにします。Cisco Intersight プライベート仮想アプライアンスは、非接続（エア ギャップ）環境で作業するユーザー向けに設計されたフォーム ファクタで提供されます。プライベート仮想アプライアンスでは、パブリック ネットワークへの接続や Cisco への接続は不要です。

ライセンス要件

Cisco Intersight プラットフォームは、複数の階層を持つサブスクリプション ベースのライセンスを使用します。1 年、3 年、または 5 年間のサブスクリプション期間を購入して、選択したサブスクリプション期間に必要な Cisco UCS サーバ ボリューム層を選択します。Cisco Intersight ポータルにアクセスしてデバイスを要求すると、各 Cisco エンドポイントに Cisco Intersight Base ライセンスが追加料金なしで自動的に含まれます。Cisco 注文ツールを使用して、次の上位層の Cisco Intersight ライセンスを購入できます。

- Cisco Intersight Essentials：Essentials には、基本ライセンスのすべての機能に加えて、Cisco UCS Central ソフトウェアと Cisco 統合管理コントローラ （IMC）スーパーバイザ権限付与、サーバ プロファイルを使用したポリシー ベースの設定、ファームウェア管理、および Cisco ハードウェア互換性リスト（HCL）との互換性の評価などの追加機能が含まれています。
- Cisco Intersight Advantage： Advantageは、Base および Essentials ティアのすべての機能を提供します。また、ストレージ ウィジェットと、コンピューティング、ストレージ、およびリモート対応環境（VMware ESXi ）にわたるクロスドメイン インベントリ相関も含まれています。サポートされている Cisco UCS プラットフォームの OS インストールも含まれています。

Cisco Intersight 管理モードのサーバには、少なくとも Essentials ライセンスが必要です。さまざまなライセンス階層で提供される機能の詳細については、
https://www.intersight.com/help/saas/getting_started/licensing_requirements を参照してください。

Cisco Unified Computing System

Cisco Unified Computing System（Cisco UCS）は、コンピューティング、ネットワーキング、ストレージ アクセス、および仮想化のリソースを 1 つのシステムに統合する次世代のデータ センター プラットフォームであり、総所有コスト（TCO）を削減し、ビジネスの俊敏性を高めることを目的として設計されています。このシステムでは、低遅延でロスレスの 10-100 ギガビット イーサネット ユニファイド ネットワーク ファブリックと、エンタープライズクラスの x86 アーキテクチャ サーバを統合しています。このシステムは、統合されたスケーラブルなマルチシャーシ プラットフォームであり、すべてのリソースを管理する統合管理ドメインです。

Cisco Unified Computing System は、次のサブシステムで構成されています。

- コンピューティング：システムのコンピューティング部分には、第 4 世代 Intel Xeon Scalable Processor に基づいたサーバが組み込まれています。サーバは、Cisco UCS Manager で管理されているブレードおよびラック フォーム ファクタで使用可能です。
- ネットワーク：システムの統合型ネットワーク ファブリックは、低遅延、無損失、10/25/40/100 Gbps のイーサネット ファブリックを提供します。LAN、SAN および管理アクセスのネットワークは、ファブリック内に統合されます。統合型ファブリックは、ネットワーク アダプタ、スイッチ、およびケーブルの数を減らすことでよりコストを削減する革新的な単一接続テクノロジーを使用します。次に、システムの電力および冷却のニーズを減らします。
- 仮想化：このシステムは、仮想環境の拡張性、パフォーマンス、および運用管理を強化することで、仮想化の可能性を最大限に引き出します。シスコのセキュリティ、ポリシー適用、および診断機能が仮想化環境にまで拡張され、変化の激しいビジネス要件とIT要件により良く対応できるようになりました。

Cisco UCS の差別化要因

Cisco Unified Computing System は、データセンターでのサーバの管理方法に革命を起こしています。Cisco Unified Computing System と Cisco UCS Manager の独自の差別化要因は次のとおりです。

- 組み込み管理：Cisco UCS のサーバは、ファブリック インターコネクトのファームウェアに組み込まれることで管理され、外部の物理または仮想デバイスの必要性を排除してサーバを管理します。
- ユニファイド ファブリック：Cisco UCS では、ブレード サーバ シャーシまたはラック サーバから FI まで、LAN、 SAN、および管理トラフィックに使用される単一のイーサネット ケーブルがあります。この統合型I/Oにより、ケーブル、SFP、およびアダプタが削減され、ソリューション全体の資本コストと運用コストが削減されます。
- 自動検出：ブレード サーバーをシャーシに挿入するか、ラック サーバーをファブリック インターコネクトに接続するだけで、管理者の介入なしにコンピューティング リソースの検出とインベントリが自動的に実行されます。ユニファイド ファブリックと自動検出の組み合わせにより、Cisco UCS のワイヤワンスアーキテクチャが可能になります。このアーキテクチャでは、 LAN、 SAN 、および管理ネットワークへの既存の外部接続を維持しながら、Cisco UCS のコンピューティング機能を簡単に拡張できます。

Cisco UCS Manager

Cisco UCS Manager（UCSM）は、Cisco UCS のすべてのソフトウェアおよびハードウェア コンポーネントの管理機能が組み込まれています。Cisco SingleConnect テクノロジーを使用して、数千の仮想マシンの複数のシャーシを管理、制御、管理できます。管理者はソフトウェアを使用して、直観的なグラフィカル ユーザー インターフェイス（GUI）、コマンドライン インターフェイス（CLI）、または堅牢なアプリケーション プログラミング インターフェイス（API）を使用した単一論理エンティティとして、Cisco Unified Computing System 全体を管理します。

Cisco Compute Hyperconverged with Nutanix

Cisco と Nutanix は、業界で最もシンプルで包括的な HCI ソリューションを提供するために協力しています。Cisco Compute Hyperconverged with Nutanixソリューションは、 Cisco Unified Computing System（UCS）の革新的なサーバ、ネットワーキング、および SaaS 管理を Nutanix の主要な HCI 基盤と組み合わせ、回数変更可能導入オプションと、 2 つの世界クラスの組織を基礎とする統合された拡張サポート モデルを提供します。

このソリューションには、次の主な利点があります。

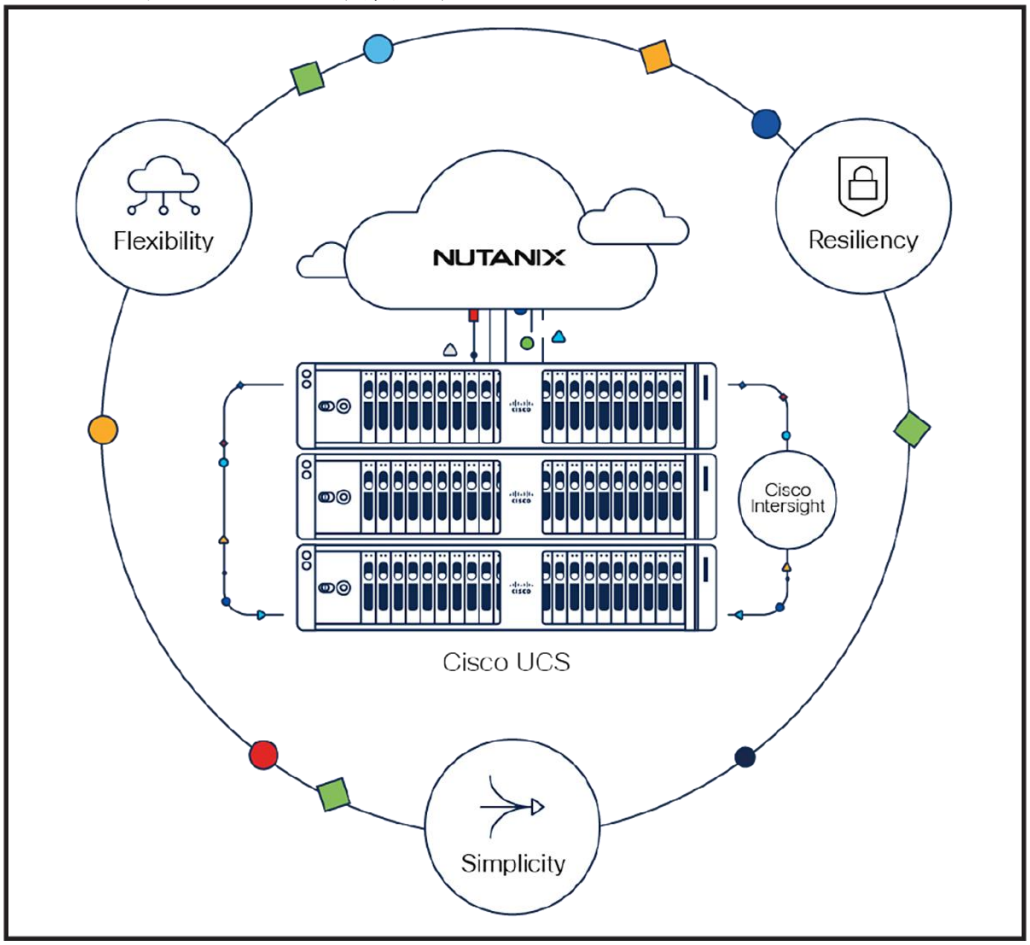
- 完全なシンプルさ：このソリューションは、SaaS とオンプレミスの両方の管理オプションを提供し、Nutanix向けにカスタマイズされたコンピューティング、ストレージ、ネットワークのサービス プロファイルを含む、0 日目から N 日目までの運用を網羅し、クラスターの展開を簡素化および加速し、パフォーマンスと回復力を向上させます。これには、より迅速かつ簡単に開始できるように、ハイパーバイザを選択して設定されたプレ

インストールされたソフトウェアも含まれます。シスコと Nutanix を組み合わせた補完的なクラウド運用モデルは、完全に統合されたクラスタのインストール、拡張、エンドツーエンドのソフトウェアとファームウェアのアップグレードなど、高度に分散された IT 環境全体で制御、可視性、および一貫性を提供します。購入エクスペリエンスを簡素化するために、Cisco に完全なソリューションを注文して提供することができます。

- 完全な柔軟性：Cisco Compute Hyperconverged with Nutanix ソリューションは、最新のアプリケーションとユースケースに対応し、UCS サーバの導入オプション、最新のアクセラレータおよびドライブ テクノロジー、業界をリードする 2 社の SaaS イノベーション（主要なパブリック クラウド プロバイダとの統合を含む）の複数の選択肢を提供します。さらに、このソリューションには、Cisco ACI 統合などの Cisco のクラス最高のネットワーキング テクノロジーが組み込まれており、ハイブリッド クラウド環境でのデータ集約型ワークロードのパフォーマンスとレジリエンシが向上します。
- 完全レジリエンシ：この共同ソリューションは、エンタープライズグレードのコンポーネントのみを使用し、コラボレーション サポート モデルとプロアクティブで自動化された復元力とセキュリティ機能によって強化されたシステム保護を提供します。これには、迅速なトリアージのための統合サポート システムとケース ノートが含まれます。ログファイルがアップロードされるか、ケース メモが生成されるたびに、その情報が共有されるため、サポート チーム間のコラボレーションが強化され、問題を迅速に解決し、顧客体験が向上します。Cisco のポリシーベースのアプローチにより、人的エラーと設定のばらつきが最小限に抑えられ、一貫した信頼性の高いの高いクラスタ展開が実現します。

また、一元化された承認を通じて全体的なセキュリティ態勢を強化し、設定の改ざんを防ぎます。

図 4.　　完全なシンプルさ、柔軟性、レジリエンシ



HCIAF240C M7 All-NVMe/All-Flash Servers

Cisco Compute Hyperconverged HCIAF240C M7 All-NVMe/All-Flash サーバは、第 4 世代Intel®Xeon®スケーラブル プロセッサ（コードネーム Sapphire Rapids）、最大256 GB の DIMM 容量を指す DDR5-4800 DIMM 用 CPU あたり16 DIMM スロットを使用して、Cisco の Compute Hyperconverged ポーフォリオの機能を 2U フォームファクタで拡張します。

All-NVMe/all-Flash サーバは、プロセッサあたり最大 60 コアの第 4 世代 Intel® Xeon® スケーラブル プロセッサ（コード名 Sapphire Rapids）を 2 基サポートします。2 ソケット構成で、32 x 256GB DDR5-4800 DIMM を搭載し、最大 8TB のメモリを搭載します。次の 2 つのサーバから選択できます。

- HCIAF240C-M7SN は最大 24 個の前面 SFF NVMe SSD を搭載します（ドライブは PCIe Gen4 x2 に直接接続）
- 前面 SFF SAS/SATA SSD を最大 24 個搭載した HCIAF240C-M7SX

詳細については、 HCIAF240C M7 All-NVMe/All-Flash Server の仕様シートを参照してください。

図 5. 正面図：**HCIAF240C M7 All-NVMe/All-Flash Servers**

NVIDIA GPU および GPU オペレータ

このソリューションは、NVIDIA L40S GPU を使用する企業の GPT-In-Box の参照アーキテクチャを提供します。

NVIDIA L40S GPU

NVIDIA L40S GPU は、データセンター向けの最も強力なユニバーサル GPU であり、世代人工知能 、LLM 推論、小規模モデルのトレーニング、微調整から 3D グラフィックス 、レンダリング、およびビデオアプリケーションに至るまで、次世代の人工知能対応アプリケーションにエンドツーエンドのアクセラレーションを提供します。

L40S GPU は、24 時間 365 日の企業データセンター運用に最適化されており、最大のパフォーマンス、耐久性、稼働時間を確保するために、NVIDIA によって設計、構築、テスト、およびサポートされています。L40S GPU は、最新のデータセンター標準を満たし、 Network Equipment-Building System（NEBS）レベル 3 に対応しており、ルート オブ トラスト テクノロジーによるセキュア ブートを備えており、データ センターのセキュリティを強化します。

表 1 NVIDIA L40S Tensor コア GPU の仕様

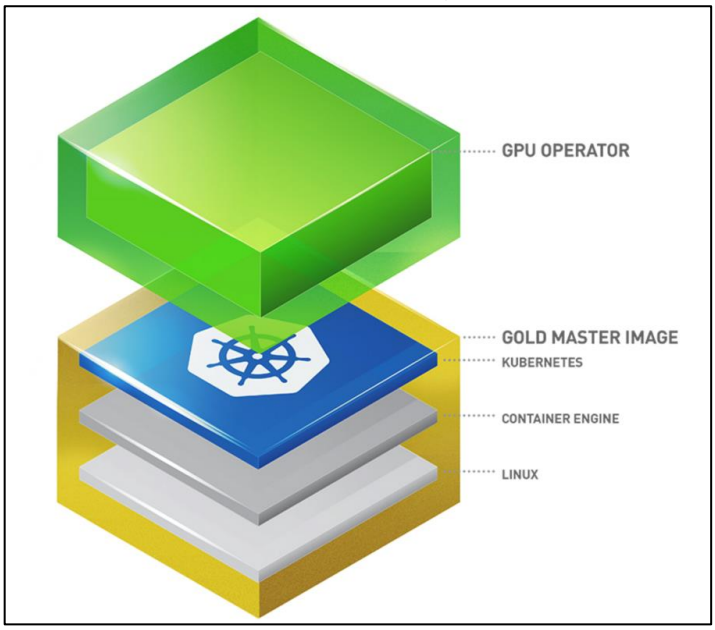
	L40S PCIe GPU
GPU アーキテクチャ	NVIDIA Ada Livelace アーキテクチャ
GPU メモリ	48GB GDDR6（ECC 付き）
GPU メモリ帯域幅	864GB/秒
インターコネクト インターフェイス	PCIe Gen4 x16：64GB/秒 双方向
NVIDIA Ada Livelace アーキテクチャベースの CUDA® コア	18,176

	L40S PCIe GPU
NVIDIA 第 3 世代 RT コア	142
NVIDIA 第 4 世代 Tensor コア	568
RT コア パフォーマンス TFLOPS	209
FP32 TFLOPS	91.6
TF32 Tensor コア TFLOPS	183 | 366
BFLOAT16 Tensor コア TFLOPS	362.05 | 733
FP16 Tensor コア	362.05 | 733
FP8 Tensor コア	733 | 1,466
ピーク INT8 Tensor TOPS	733 | 1,466
ピーク INT4 Tensor TOPS	733 | 1,466
フォーム ファクタ	4.4 インチ（高さ）X 10.5 インチ（長さ）、デュアル スロット
ディスプレイ ポート	ディスプレイ ポート 1.4a X 4
最大電力消費	350 W
電源コネクタ	16 ピン
サーマル	パッシブ
仮想 GPU（vGPU）ソフトウェア サポートあり	はい
NVENC | NVDEC	3x | 3x（AV1 エンコードおよび復号化を含む）
ルート オブ トラストを使用したセキュア ブート	はい
NEBS 対応	レベル 3
MIG サポート	いいえ
NVIDIA® NVLink® サポート	いいえ

NVIDIA GPU オペレータ

図 6 に、NVIDIA GPU オペレータの概要を示します。

図 6.　　NVIDIA GPU オペレータの概要



Kubernetes は、デバイス プラグイン フレームワークを介して、NVIDIA GPU、NIC、Infiniband アダプタ、その他のデバイスなどの特別なハードウェア リソースへのアクセスを提供します。ただし、これらのハードウェア リソースを使用してノードを構成および管理するには、ドライバ、コンテナ ランタイム、またはエラーが発生しやすいその他のライブラリなど、複数のソフトウェア コンポーネントを構成する必要があります。NVIDIA GPU オペレータは、Kubernetes 内のオペレータ フレームワークを使用して、GPU のプロビジョニングに必要なすべての NVIDIA ソフトウェア コンポーネントの管理を自動化するします。これらのコンポーネントには、（CUDA を有効にするための）NVIDIA ドライバ、GPU 用の Kubernetes デバイス プラグイン、NVIDIA Container Toolkit、GFD を使用した自動ノード ラベリング、DCGM ベースのモニタリングなどが含まれます。

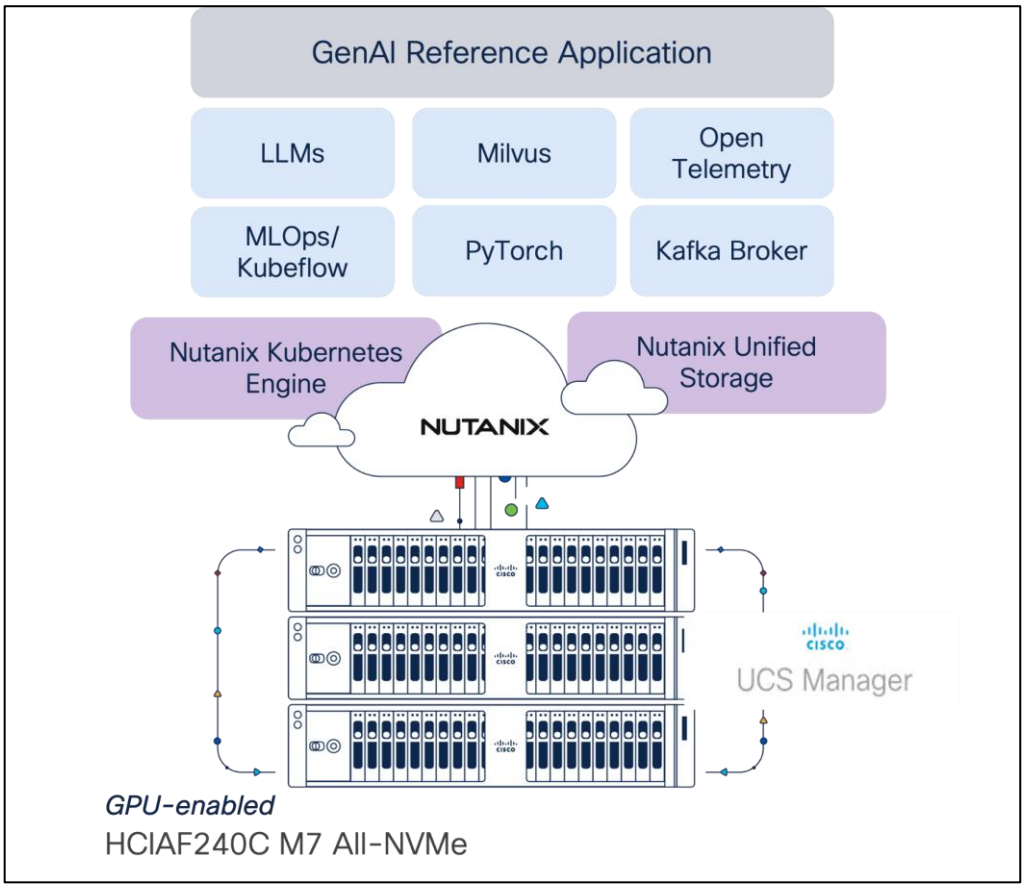
ソリューション設計

この章の内容は、次のとおりです。

ソリューションの概要

このソリューションは、AI 対応プラットフォームの基本的なリファレンス アーキテクチャを提供し、顧客がオンプレミスの AI ソリューションを迅速に設計、サイズ設定、展開できるようにします。このソリューションで設定される主要な設計コンポーネントを 図 7に示します。

図 7.　　GPT-in-a-Box ソリューション設計



この設計には、次のハードウェアおよびソフトウェア コンポーネントが含まれています。

- オンプレミスの Cisco Compute Hyperconverged (CCHC) と Nutanix クラスタ内の単一リージョン内の単一のアベイラビリティ ゾーン (AZ)
- CCHC with Nutanix (AHV) クラスタには、最低 4 ノードの HCIAF240C M7 All-NVMe サーバが含まれ、各ノードには 2 つの NVIDIA L40S: 350W、48GB GPU が搭載されています。
- Nutanixクラスタを搭載した CCHC は、次の主要な Nutanix サービスをホストします
  - Nutanix Unified Storage（NUS）はNFS ストレージと S3 互換ストレージ機能を提供します
  - Nutanix Kubernetes Engine（NKE）クラスタは管理、モニタリング、およびワークロードクラスタで使用される永続アプリケーションに使用されます

注：　Nutanix Prism Central は、別の Nutanix AHV クラスタでホストされていました。Prism Central は、既存のクラスタまたは別の AHV ベースの Nutanix クラスタに展開できます。

インフラストラクチャ 設計

次の図に、Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box の導入アーキテクチャの詳細を示します。第 0 日の展開全体は、Nutanix Foundation VM で定義されたワークフローを通じて管理されます。

GPT-in-a-box 用 Nutanixクラスタを使用した CCHC は、別の Nutanix AHV クラスタに展開された Prism Central を介して管理されます。Prism Central は、同じ Nutanix クラスタでホストされている GPT-in-a-Boxソリューションに展開することもできます。

HCIAF240C M7 All-NVMe ノードは、UCS 管理モードの Cisco UCS 6536 ファブリック インターコネクトのペアに接続されます。

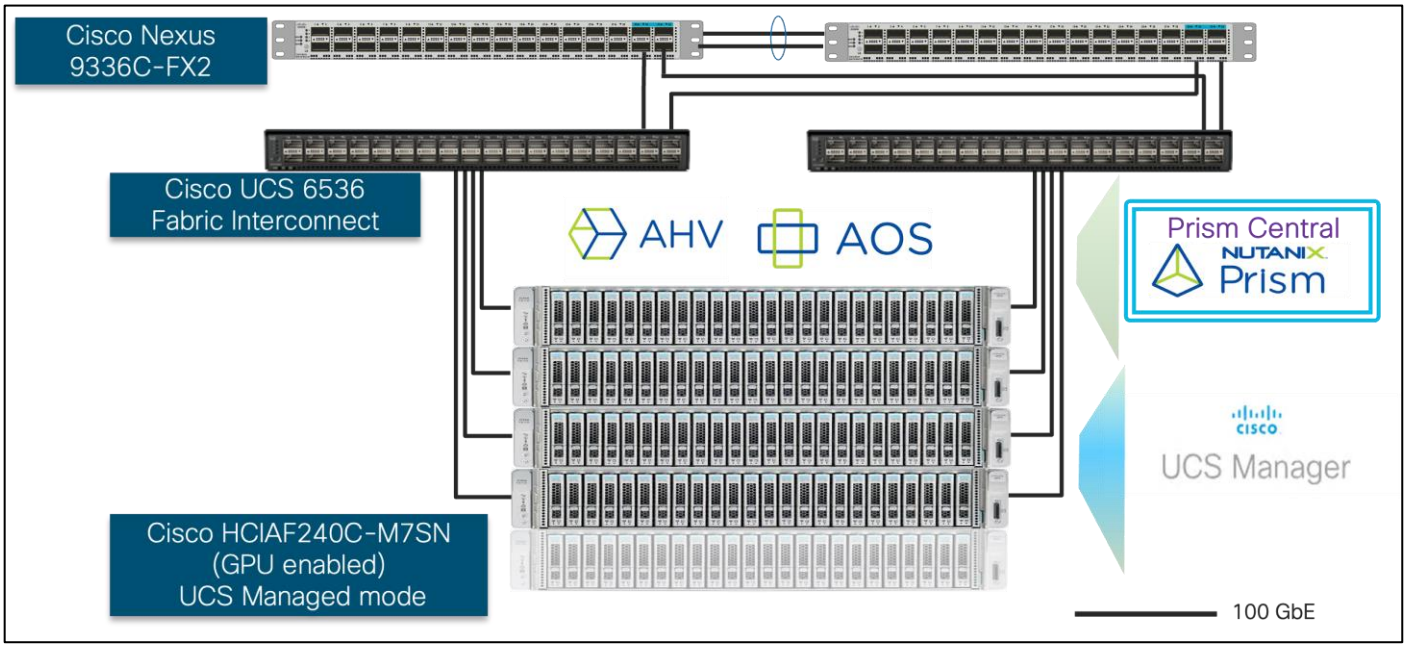
各 HCIAF240C M7 All-NVMe サーバは、次で構成されます。

- Intel I6442Y プロセッサ（2.6 GHz/225W 24C/60MB）X 2
- 1 TB DDR5 メモリ（32 GB DDR5-4800 RDIMM X 32）
- M.2 RAID コントローラを介して管理される 2x 240GB M.2 カード
- 3.8 TB NVMe X 6
- 1 x Cisco VIC 15238 2x 40/100/200G mLOM
- NVIDIA L40S X 2：350W、48GB GPU

このソリューションを検証ために導入されたインフラストラクチャの完全な仕様については、「部品表（BoM）」セクションを参照してください。

図 8 は、 Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box のハードウェア導入アーキテクチャを示しています。

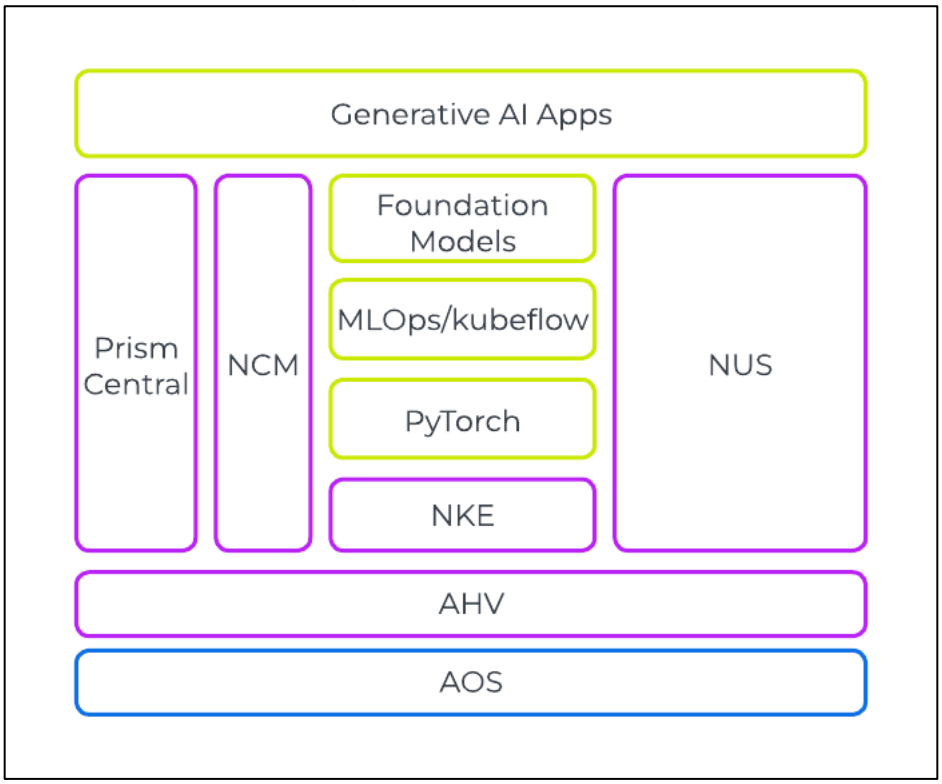
図 8.　　インフラストラクチャ設計



このソリューションは、オンプレミスのデータセンターの単一のリージョンに単一のアベイラビリティ ゾーン（AZ）を備えています。

- 特に次のサービスをホストする単一の Nutanix クラスタ。
  - Prism Central を含む Nutanix 管理コンポーネント
  - NFS ストレージおよび S3 互換ストレージ機能を提供する NUS
  - GPT ワークロードに使用される NKE クラスタ
  - ワークロードクラスタで使用される管理、モニタリング、および永続アプリケーションに使用される NKE クラスタ

図 9.　　GPT-in-a-Box の概念設計



VLAN 設定

表 2 に、既存のソリューションに設定されている VLAN を示します。

表 2 VLAN Usage

VLAN ID	名前	利用	この展開で使用される IP サブネット
2	ネイティブ VLAN	デフォルト VLAN（1）の代わりに、VLAN 2 をネイティブ VLAN として活用します。	
1080	OOB-MGMT-VLAN	さまざまなデバイスの管理ポートを接続するアウトオブバンド管理 VLAN	10.108.0.0/24; GW: 10.108.0.254
1081	NTNX-VLAN	Nutanix クラスタ管理、ファイルとオブジェクト サービス、および Nutanix Kubernetes の展開に使用される VLAN	10.108.1.0/24; GW: 10.108.1.254

表 3 に、ソリューションの IP アドレス割り当てを示します。

表 3 仮想マシン

タイプ	VLAN	IP アドレスの範囲	DNS エントリ	コメント
UCS 管理	1080	10.108.0.110-120		UCS 管理とアウトオブバンド サーバ管理
GPT-in-a-Box Nutanix クラスタ	1081	10.108.1.121-130		Nutanix ノード AHV、CVM、iSCSI データ サービス、およびクラスタ VIP

タイプ	VLAN	IP アドレスの範囲	DNS エントリ	コメント
Prism Central	1081	10.108.1.140	prismcentral0.rtp4.local	別の AHV Nutanix クラスタでホストされている Prism Central
ファイルサーバー	1081	10.108.1.141	fileserver-01.rtp4.local	GPT-in-a-Box クラスタでホスト
オブジェクトおよびファイル サービス用の IPAM	1081	10.108.1.171-180		
Nutanix Kubernetes サービス用 IPAM	1081	10.102.1.180-190		Prism Central で構成済み
Kubernetes 管理クラスタへの ngnix イングレスのワイルドカード サブドメイン	1081	10.108.1.213-216	*.ntnx-k8-mgmt.rtp4.local	ホスト アドレス レコード 10.108.1.213 としてのローカル Domain Name System（DNS）のエントリ
Kubernetes ワークロード クラスタへの ngnix イングレスのワイルド カード サブドメイン	1081	10.108.1.217-220	*.ntnx-k8-workload.rtp4.local	ホスト アドレスレコード 10.108.1.21310.108.1.217 としてのローカル DNS のエントリ
llm エンドポイントのワイルドカード サブドメイン	1081	10.108.1.218	*.llm.ntnx-k8-workload.rtp4.local	ホスト アドレスレコード 10.108.1.218 としてのローカル DNS のエントリ

ソフトウェアのリビジョン

表 4 に、ソリューションのさまざまなコンポーネントのソフトウェアリビジョンを示します。

表 4 ソフトウェアのリビジョン

デバイス	イメージバンドル	コメント
Cisco UCS 6536 ファブリック インターコネクト	4.3(4a)	FI およびサーバ ファームウェアを含むインフラストラクチャ向け Cisco UCS GAリリース
Cisco UCS HCIAF240C M7 All-NVMe サーバ	4.3(4c)	GPT-in-a-Box クラスタ用 HCIAF240C M7 All-NVMe ノード X 4
NVIDIA L40S：350W、48GB GPU	lcm_nvidia_20230302.2008_535.129.03	Nutanix ポータルからのダウンロード
UCS Managed HCIAF240C M7 All-NVMe サーバ上の Nutanix AOS/AHV クラスタ	6.7.1.5	
別の Nutanix AHV クラスタ	2023.4	

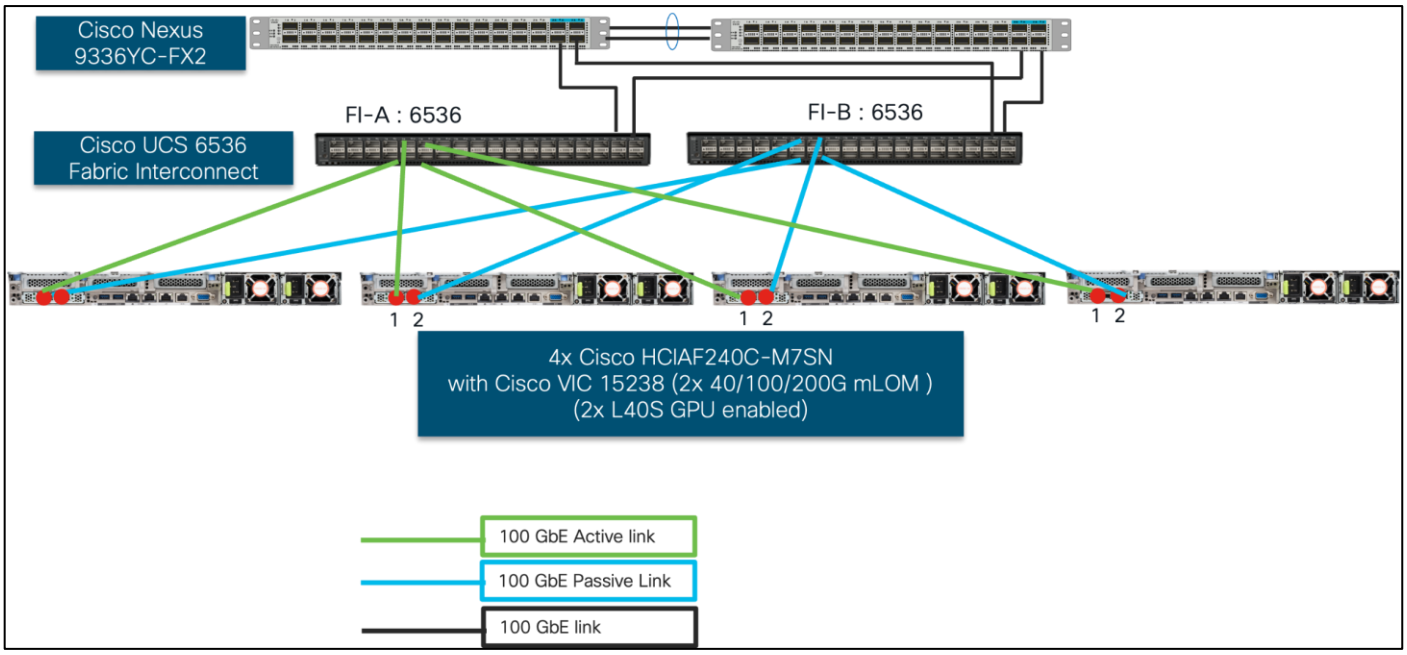
デバイス	イメージバンドル	コメント
でホストされている Prism Central		
Nutanix マーケットプレイスを介して有効になり、GPT-in-a-Box クラスタでホストされる NKE	1.26.8-0 (OS Image - ntnx-1.6.1)	
Nutanix マーケットプレイスを介して有効になり、GPT-in-a-Box クラスタでホストされるファイル	4.4.03	
Nutanix マーケットプレイスを介して有効になり、GPT-in-a-Box クラスタでホストされるオブジェクト	4.3.02	

ネットワーク設計

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box は、100GbE エンドツーエンドネットワークを使用します。

図 10 に、このソリューションで使用されるネットワーク設計を示します。

図 10.　　ネットワーク設計



ネットワーク設計の主な側面は次のとおりです。

- 各 HCIAF240C M7 All-NVMe サーバは、UCS 管理モードで Cisco UCS 6536 ファブリック インターコネクトに接続します。

- 各ノードには、40/100/200 Gbps イーサネットまたは FCoE をサポートするデュアルポート Small Form-Factor Pluggable（QSFP/QSFP28/QSFP56）mLOMカードである Cisco VIC 15238 2 X 40/100/200G mLOM が搭載されています。
- ハイパーバイザのネットワーク ポートは、アクティブ/パッシブ モードで設定されます。これらは、クラスタの自動インストール時に自動的に構成されます。
- UCS 管理モードでの展開は、ボンド モード 4（LACP）をサポートしていません。
- ファブリック インターコネクト ポートは 100G アップリンク ポートとは別に、コア LAN または専用の管理ネットワーク スイッチ経由で接続できます。

クラスタ設計

この設計には、大規模な言語モデル（LLM）へのアクセスを提供する GPT-in-a-Box ワークロード専用の単一の GPU 対応 Nutanix クラスタが組み込まれています。Prism Central やファイルおよびオブジェクト ストレージなどのサポートする管理アプリケーションは、このクラスタでホストされます。

16 個の vCPU と 64 GB のメモリを搭載した Nutanix GPT-in-a-Box クラスタ コントローラ VM（CVM）のサイジング。

この設計では、GPT-in-a-Box クラスタをホストする単一の AZ を持つ 1 つのリージョンを使用します。このソリューションでは、単一のクラスタで GPT ワークロードをホストすることに焦点を当てているため、ワークロードを保護するためにレプリケーション ターゲットを使用しません。

図 11.　GPT-in-a-Box クラスタの概念アーキテクチャ

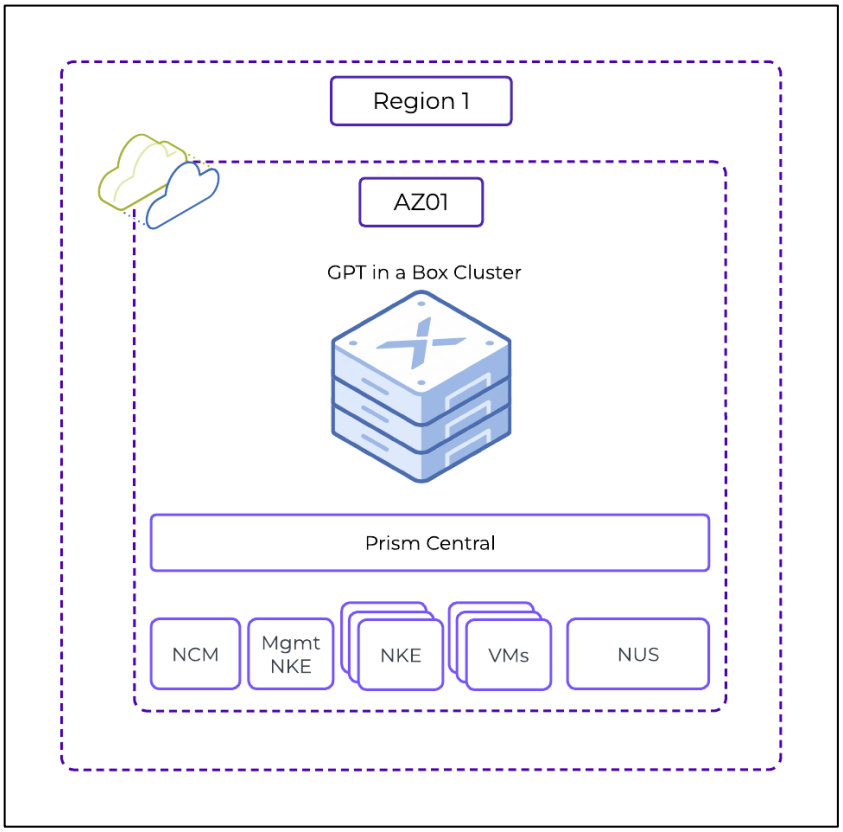


表 5 クラスタ設計の決定事項

設計オプション	検証済みの選択
クラスタ サイズ	データセンター内のすべてのコンポーネントの完全な冗長性を確保します。
CPU	少なくとも 24 コアと高いクロックレートを活用します。
最小クラスタ サイズ	少なくとも 4 つのノードを活用します。
クラスタ拡張	1 ずつ拡張します。
最大クラスタ サイズ	最大 16 ノードを活用します。
Networking	100 GbE ネットワーキングを使用します。
クラスタ レプリケーション係数	ストレージ レプリケーション係数 2 を使用します。
クラスタの高可用性構成	高可用性を保証します。
VM 高可用性	クラスタで高可用性予約を有効にします。

クラスタの復元力

VM の高可用性により、元の AHV ホストで完全な障害が発生したり、ネットワークがパーティション分割されたり、AHV ホスト管理プロセスの障害が発生したりして、ホストが使用できなくなった場合に、AHV クラスタの別の AHV ホストで VM が再起動します。VM が実行されている AHV ホストが使用できなくなると、VM はオフになります。したがって、VM オペレーティング システムの観点からは、VM の高可用性には完全な VM 起動周期が含まれます。

表 6 高可用性構成

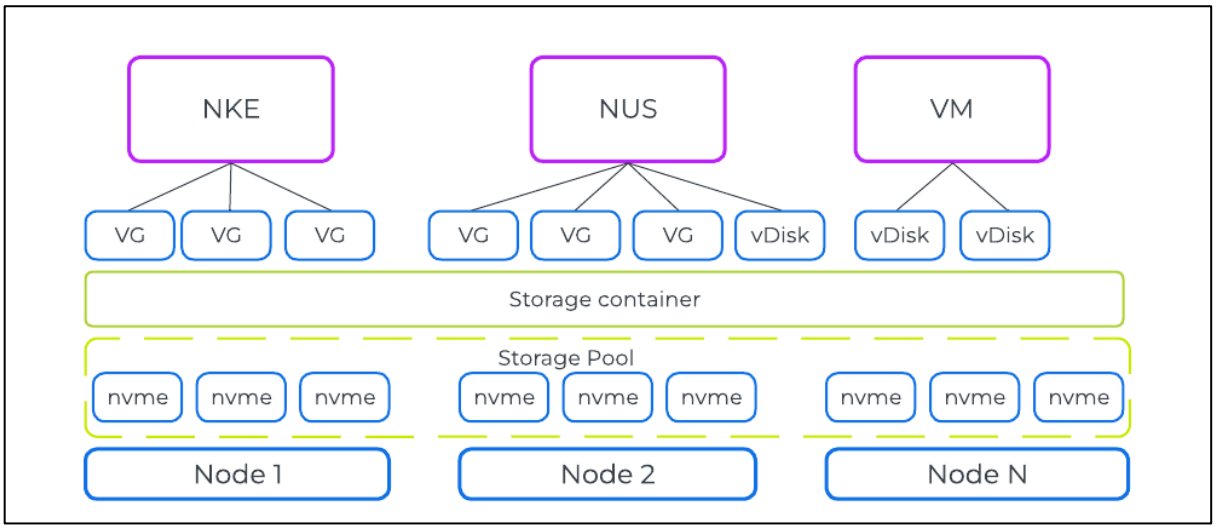
特長	説明	設定
高可用性予約	アプリケーションのクラスタ内のコンピューティング フェールオーバー キャパシティを保証する	有効（Enabled）
キャパシティの予約の再構築	クラスタ内のストレージ再構築キャパシティを保証する	有効（Enabled）

ストレージ設計

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box は、ストレージに分散型シェアードナッシング アーキテクチャを使用します。

図 12 は、このソリューションで使用されるストレージ設計を示しています。

図 12.　　ストレージ設計



Nutanix AHV クラスタを作成すると、次のストレージ コンテナが自動的に作成されます。

- NutanixManagementShare：ファイルやオブジェクトなどの Nutanix 機能やその他の内部ストレージのニーズに使用されます。ワークロードvDiskを保存しません。
- SelfServiceContainer：VM をプロビジョニングするために NCM セルフサービス ポータルで使用されます。
- Default-Container-XXXX：ユーザー VM およびアプリケーションの vDisk を保存するために VM によって使用されます。

自動的に作成されたストレージ コンテナに加えて、Nutanix ファイルとオブジェクトの展開中に、次の追加のストレージ コンテナが作成されます。

- NTNX_<fileserver_name> _ctr：ファイル サーバ インスタンスにストレージを提供し、アプリケーション階層に NFS ストレージを提供します。
- オブジェクト<uniqueidentifier>：Nutanix オブジェクトのデータ コンテナ
- オブジェクト<uniqueidentifier>：Nutanix オブジェクトのメタデータ コンテナ

デフォルト コンテナには、VM とその vDisk が保存されます。Nutanix ファイルとオブジェクトの追加コンテナは、それぞれのコンポーネントの展開プロセス全体で作成されます。

注：　クラスタの実効キャパシティを増やすために、この設計では、すべてのストレージ コンテナでインライン圧縮が有効になっています。重複排除や消去コーディングなどの追加機能は使用しません。レプリケーション係数 2 は、障害やメンテナンスの際に単一のコンポーネントの損失から保護します。

データ削減と復元力のオプション

クラスタの実効キャパシティを増やすために、この設計では、すべてのストレージ コンテナでインライン圧縮が有効になっています。重複排除や消去コーディングなどの追加機能は使用しません。レプリケーション係数 2 は、障害やメンテナンスの際に単一のコンポーネントの損失から保護します。

表 7 データ削減設定

コンテナ	圧縮	重複排除	消去コーディング	レプリケーション係数
Default-Container-XXXX	オン	消灯	消灯	2

コンテナ	圧縮	重複排除	消去コーディング	レプリケーション係数
NutanixManagementShare	オン	消灯	消灯	2
SelfServiceContainer	オン	消灯	消灯	2
NTNX_files_ctr	オン	消灯	消灯	2
objects containers	オン	消灯	消灯	2

表 8 に、この設計で行ったストレージの決定に関する情報を示します。

表 8 ストレージ設計オプション

設計オプション	検証済みの選択
クラスタのサイジング	オールフラッシュ クラスタを活用して、低遅延、高スループットのストレージを提供し、アプリケーションの現用系データセットをサポートします。
ノード タイプのベンダー	同じクラスタに異なるベンダーのノード タイプを混在させないでください。
ノードとディスクのタイプ	同様のディスクを持つ同様のノード タイプを活用します。ハイブリッド SSD または HDD ノードと同じクラスタに NVMe SSD を含むノードを混在させないでください。
ストレージとコンピューティングのノード冗長性のためのサイジング	n + 1 のフェールオーバー キャパシティのすべてのクラスタのサイズを設定します。
耐障害許容度とレプリケーション係数の設定	クラスタを許容度障害性 1 に構成し、コンテナをレプリケーション係数 2 に構成します。
インライン圧縮	インライン圧縮を有効化します。
重複排除	重複排除を有効にしないでください。
消去コーディング	消去コーディングを有効にしないでください。
クラスタの可用性ドメイン	ノード認識を活用します。
クラスタのストレージ コンテナ	クラスタには、NutanixManagementShare、SelfServiceContainer、Default-Container、NTNX_files_ctr、および 2 つのオブジェクト ストレージ コンテナのストレージ コンテナがあります。
再構築キャパシティの予約	再構築キャパシティの予約を有効にします。

Nutanix ファイルとオブジェクトの設計

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box は、Nutanix ファイルを使用して、NFS プロトコルを使用するアプリケーションに高性能な共有ストレージを提供します。Nutanix ファイルは LLM を一時的に保存し、VM と Kubernetes サービス間で使用できるようにします。

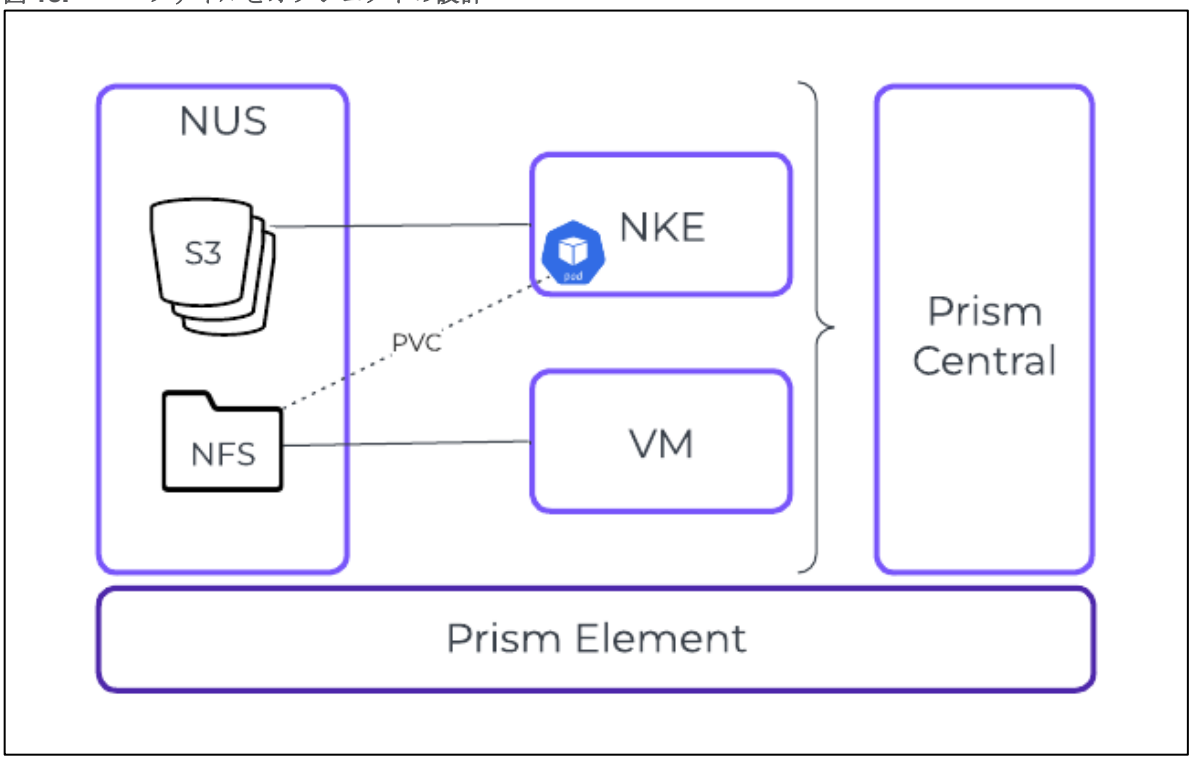
Nutanix オブジェクトは、S3 互換のストレージ機能をアプリケーションに提供し、ユーザーが新しいデータ モデルをアップロードできるようにします。

Nutanix ファイルとオブジェクトは、GPT-in-a-Box ワークロードと同じ Nutanix クラスタで実行されます。Prism Central は、すべてのコンポーネントの展開と管理に使用されます。

注： 環境を拡張する場合は、専用クラスタで Nutanix ファイルまたは Nutanix オブジェクトを実行することもできます。

図 13 に、Nutanix ファイルとオブジェクトの設計を示します。図 13 に表示されている VM は、NFS 共有をマウントして LLM を保存するために使用されるエンド ユーザーまたはジャンプ ホストです。

図 13. ファイルとオブジェクトの設計



Nutanix ファイルとオブジェクトには、次のネットワーク要件があります。

- パフォーマンスを最大化するために、Nutanix ファイルのストレージ ネットワークは CVM と同じサブネットを使用します。
- セキュリティを最大化するために、クライアント ネットワークは別のサブネットに接続します。
- 単一のファイルサーバ インスタンスを提供する GPT-in-a-Boxクラスタには、3 つのファイル サーバ VM（FSVM）用に 3 つのストレージ ネットワーク IP アドレスと 4 つのクライアント ネットワーク IP アドレスが必要です。

Nutanix オブジェクトは、Kubernetes マイクロサービス プラットフォームでコンテナ化されたサービスとして実行され、新機能の速度の向上などの利点を提供します。必要なストレージ IP アドレスのいくつかは、基盤となるマイクロサービス プラットフォームに関連する機能用です。これらのネットワークも管理する必要があります。3 つのワーカーノードを持つ Nutanix オブジェクトには、7 つのストレージ ネットワークIP アドレスと 2 つのクライアント ネットワーク IP アドレスが必要です。

NKE によってプロビジョニングされる各クラスタには、最小 IPアドレス要件があります。3 つのワーカー ノードを持つ実稼働レベルのレイアウトの Kubernetes クラスタには、1 つの静的 IP アドレスと 8 つ以上の IPAM アドレスが必要です。

クライアント ネットワークは、すべての IP アドレスを提供します。追加のワーカー ノードに対して追加の IPAM アドレスが必要になる場合があります。

詳細については 、「Nutanix Files, Nutanix Objects, and Nutanix Kubernetes Engine ネットワーク設計」を参照してください。

表 9 に、この設計で行われた Nutanix ファイルとオブジェクトの決定に関する情報を示します。

表 9 GPT-in-a-Box ファイルの設計上の決定事項

設計オプション	検証済みの選択
FSVM クラスタ サイズ	3 つの FSVMを使用します。
FSVM の vCPU とメモリ	FSVM ごとに 12 個の vCPU と 64 GB のメモリを使用します。
ストレージ ネットワーク	ストレージ ネットワークを CVM および AHV と同じサブネットに保持します。
クライアント ネットワーク	別のサブネットを活用して、ストレージへのクライアント アクセスを提供します。
耐障害許容度とレプリケーション係数の設定	クラスタを許容度障害性 1 に構成し、コンテナをレプリケーション係数 2 に構成します。
ストレージ コンテナ	NUS ファイルをホストするには、別のストレージ コンテナを活用します。
消去コーディング	消去コーディングを有効にしないでください。
圧縮	圧縮を有効にします。
重複排除	重複排除を有効にしないでください。
作成するための共有	1 つの共有を作成：llm-repo
共有のためのプロトコル	共有に NFS を使用します。

詳細については、「Nutanix Files and Nutanix Objects Design Decisions」を参照してください。

表 10 GPT-in-a-Box オブジェクトの設計上の決定事項

設計オプション	検証済みの選択
Nutanix オブジェクトのクラスタ サイズ	3 つのワーカー ノードと 2 つのロード バランサ ノードを使用します。
ワーカー ノード サイズ	各ワーカーノードに 10 個の vCPU と 32 GB のメモリを使用します。
ロード バランサのノード サイズ	ロード バランサ ノードごとに 2 つの vCPU と 4 GB のメモリを使用します。
ストレージ ネットワーク	ストレージ ネットワークを CVM および AHV と同じサブネットに保持します。
クライアント ネットワーク	別のサブネットを活用して、ストレージへのクライアント アクセスを提供します。
耐障害許容度とレプリケーション係数の設定	クラスタを許容度障害性 1 に構成し、コンテナをレプリケーション係数 2 に構成します。
ストレージ コンテナ	NUS オブジェクトをホストするには、別のストレージ コンテナを使用します。

設計オプション	検証済みの選択
消去コーディング	消去コーディングを有効にしないでください。
圧縮	圧縮を有効にします。
重複排除	重複排除を有効にしないでください。
作成するバケット	milvus（ベクトル データベース）、ドキュメント（エンドユーザーアクセス用）、およびバックアップ（バックアップ ターゲット）の 3 つのバケットを作成します。

管理設計

Cisco UCS Manager、Prism Central、Active Directory、 DNS、NTP などの管理コンポーネントは、高可用性を必要とする重要なサービスです。Prism Central は Nutanix のグローバル コントロールプレーンであり、VM 管理、アプリケーション オーケストレーション、マイクロ セグメンテーション、およびその他のモニタリングおよび分析機能を担当します。

このソリューションでは、次の 2 つのキー管理プレーンを使用します。

- Cisco UCS Manager は、Cisco UCS サーバ全体をサポートします。これにより、サーバ、ファブリック、およびストレージのプロビジョニングと、デバイス検出、登録、設定、診断、モニタリング、障害検出、監査、および統計情報の収集が可能になります。
- Prism Central は、別の Nutanix AHVクラスタに単一の VM として展開されました。Prism Central は以下を管理します。
  - Nutanix Kubernetes Engine
  - Nutanix ファイルとオブジェクト
  - VM
  - RBAC
  - コア Nutanix サービスのモニタリング、オブザーバビリティ、および監査

DNS 管理

名前解決と SSL証明書の管理は、Kubernetes クラスタの展開と管理に不可欠です。このソリューションでは、 DNS ルート ドメインは Windows DNS サーバでローカルにホストされます。サブドメインは DNS をマッピングし、トラフィックを Kubernetes クラスタにルートします。

監視

ソリューションのモニタリングは、イベント モニタリングとパフォーマンス モニタリングの 2 つのカテゴリに分類されます。各カテゴリは、さまざまなニーズと問題に対処します。

高可用性環境では、高いサービス レベルを維持するためにイベントをモニタリングする必要があります。障害が発生した場合、管理者ができるだけ早く修復アクションを実行できるように、システムは時間どおりにアラートを生成する必要があります。このソリューションは、障害発生時にアラートを生成する Nutanix プラットフォームの組み込み機能を構成します。

プラットフォームを正常に保つことに加えて、リソース使用率を正常なレベルに維持することも、高パフォーマンスの環境を提供するために不可欠です。パフォーマンス モニタリングは、アプリケーションのパフォーマンスをトラブ

ルシュートする必要がある場合に不可欠なメトリックを継続的にキャプチャして保存します。包括的なモニタリングアプローチでは、次の領域を追跡します。

- アプリケーションとデータベース メトリクス
- オペレーティング システム メトリクス
- ハイパーコンバージド プラットフォーム メトリクス
- ネットワーク環境メトリクス
- 物理環境メトリック

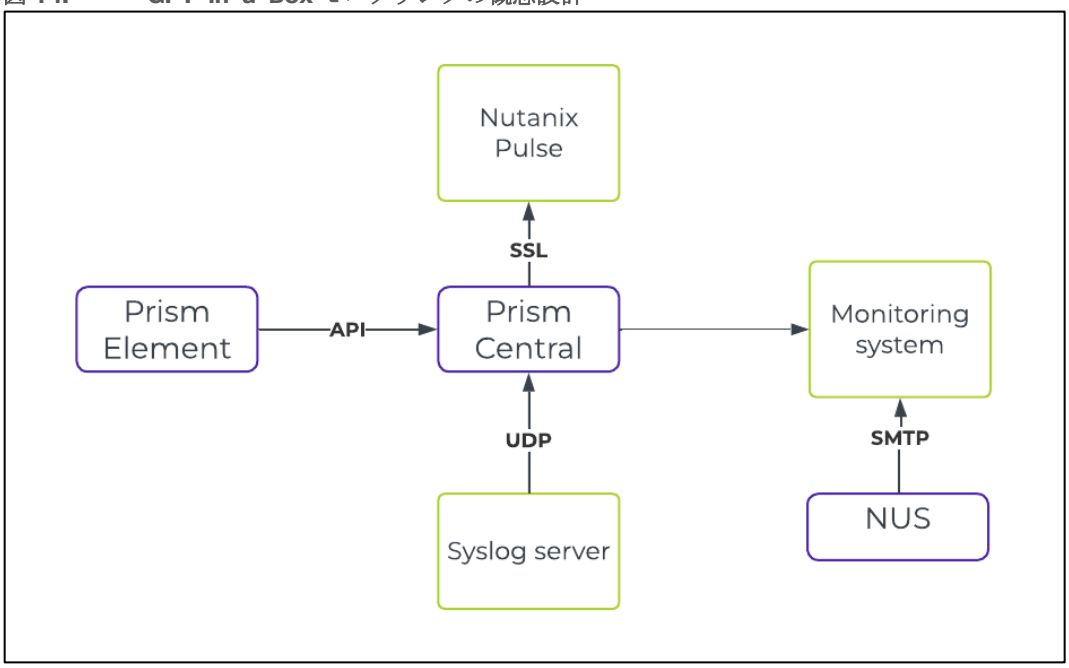
Nutanix プラットフォームは、これらの領域でさまざまなメトリクスを追跡することで、スタック全体のキャパシティ モニタリングも実施できます。ほとんどの企業環境は必然的に成長するため、キャパシティの需要の変化を予測し、リソースの不足によるビジネスへの影響を回避するために、リソースの使用状況と拡張率を理解する必要があります。

概念設計のモニタリング

この設計では、Prism Central は Nutanix コア インフラストラクチャのイベント モニタリングを実行します。この設計では、ログ収集に syslog を使用します。詳細については、「Securityとコンプライアンス」のセクションを参照してください。この設計では、SMTP ベースの電子メール アラートが通知のチャネルとして機能します。Prism Central が使用できない状況に対応するために、この設計の各 Nutanixクラスタは SMTP を使用して通知も送信します。個々の Nutanix クラスタは、Prism Central が使用できない場合にのみモニタリングされる別の受信メール ボックスにアラートを送信します。

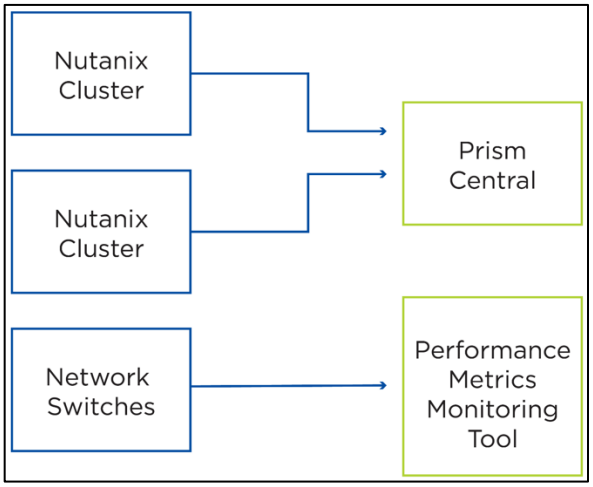
Kubernetesクラスタからのログは、長期的な S3 エクスポータを介して Nutanix オブジェクトに送信され、syslogエクスポータを介して syslog サーバに送信されます。syslog サーバはユーザー データグラム プロトコル（UDP）を使用してデータを Prism Central に送信し、Prism Element は API を使用してデータを Prism Central に送信します。その後、Prism Central はログを Nutanix Pulse（Secure Sockets Layer を使用）とモニタリング システムに送信します。モニタリング システムは SMTP を使用して Nutanix Unified Storage にログを送信します。

図 14.　GPT-in-a-Box モニタリングの概念設計



Prism Central は、CPU、メモリ、ネットワーク、ストレージの使用率などの主要な領域でクラスタのパフォーマンスをモニタリングし、デフォルトでこれらのメトリックをキャプチャします。Prism Central インスタンスがクラスタを管理する場合、Prism Central はすべての Nutanix Pulse データを送信するため、個々のクラスタから発信されることはありません。Nutanix Pulse を有効にすると、クラスタの安定性に影響を与える既知の問題が検出され、サポート ケースが自動的に開きます。

図 15. ハイブリッド クラウド パフォーマンス メトリック システム



クラスタを接続するネットワーク スイッチも、クラスタのパフォーマンスにおいて重要なロールを果たします。展開されたスイッチと互換性のある別のモニタリング ツールは、スイッチの評価メトリクスをキャプチャできます。たとえば、SNMP ベースのツールは、スイッチからカウンタを定期的にポーリングできます。

表 11 に、モニタリング設計の決定事項を示します。

表 11 設計上の決定事項のモニタリング

設計オプション	検証済みの選択
プラットフォーム パフォーマンス モニタリング	Prism Central は、Nutanix プラットフォームのパフォーマンスをモニタリングします。
ネットワーク スイッチ パフォーマンス モニタリング	スイッチへの SNMP 投票を実行する別のツールは、ネットワーク スイッチのパフォーマンスをモニタリングします。
SMTP アラート	SMTP アラートを使用します。 Prism Element および Prism Central のプライマリ SMTP ゲートウェイとして企業 SMTP サービスを使用します。
SMTP アラート送信元の電子メール アドレス	送信元電子メールアドレスを clustername@ に設定します。<yourdomain>.com を使用して、電子メールメッセージの送信元を一意に識別します。Prism Central の場合は、クラスタ名の代わりに Prism Central のホスト名を使用します。
SMTP アラート Prism Central 受信者の電子メール アドレス	Prism Central 受信者の電子メール アドレスを primaryalerts@<yourdomain>.com に設定します。
SMTP アラート Prism Element 受信者の電子メール アドレス	Prism Element の受信者の電子メール アドレスを secondaryalerts@<yourdomain>.com に設定します。
NCC レポート	日次 NCC レポートを現地時間の午前 6 時に実行し、プライマリ アラー

設計オプション	検証済みの選択
	ト メールボックスに電子メールで送信するように設定します。
Nutanix Pulse	Nutanix クラスタをモニタリングし、テレメトリデータを Nutanix に送信するように Nutanix Pulse を構成します。

セキュリティとコンプライアンス

Nutanix は、企業データセンター ソリューション全体でセキュリティを階層化するための多層防御戦略を推奨しています。この設計セクションでは、Nutanix が制御およびデータプレーン レベルで直接監視できる階層の検証に焦点を当てます。

セキュリティ ドメイン

ファイアウォールを使用して、Nutanix クラスタ管理インターフェイスとアウトオブバンド インターフェイスをネットワークの残りの部分から分離し、管理セキュリティ ドメインからの直通アクセスのみを許可します。さらに、Nutanix では、アウトオブバンド管理インターフェイスとクラスタ管理インターフェイスを、アプリケーショントラフィックから離れた専用の VLAN に分離することを推奨しています。

Syslog

コントロール プレーン エンドポイント（Prism Central）ごとに、システムレベルの内部ロギングは、既存のお客様ランドスケープで実行される一元化されたサードパーティの syslog サーバに送信されます。システムは、使用可能なすべてのモジュールが syslog エラー シビラティに達すると、それらのログを送信するように設定されます。

この設計では、集中型 syslog サーバの可用性が高く冗長であることを前提としているため、プライマリ ログ システムが使用できない場合にログ ファイルを検査できます。

証明書

SSL エンドポイントは、すべての Nutanix コントロール プレーン Web ページにサービスを提供します。展開と検証では、自己署名証明書が使用されます。既存のソリューションでは、デフォルトの自己署名証明書を、 Microsoft公開キーインフラストラクチャ（PKI）の内部証明書局によって署名された証明書に置き換えることができます。コントロール プレーンと対話するクライアント エンドポイントには、ブラウザのセキュリティ エラーを防ぐために、信頼された証明機関チェーンをプリロードする必要があります。

注： 証明書の管理は継続的なアクティビティであり、証明書は定期的にローテーションする必要があります。このソリューションは、有効期間が 1 年間のすべての証明書に署名します。

表 12 セキュリティ設計の決定事項

設計オプション	検証済みの選択
静止データの暗号化（DaRE）	DaRE を使用しないでください。
SSL エンドポイント	内部の信頼された証明機関（Microsoft PKI）を使用してコントロール プレーン SSL エンドポイントに署名します。
証明書	毎年の期日を使用して証明書をプロビジョニングし、それに応じてローテーションします。
認証	Active Directory LDAPS 認証を活用します。
コントロール プレーン エンドポイントの管理	すべてのコントロール プレーン エンドポイントに共通の管理 Active

設計オプション	検証済みの選択
	Directory グループを活用します。
クラスタ ロックダウン モード	クラスタ タブのロックダウン モードを有効にしないでください（パスワード指向の SSH を許可します）。
デフォルト以外の強化オプション	AIDE と毎時 SCMA を有効にします。
システム レベルの内部ロギング	使用可能なすべてのモジュールについて、外部 syslog サーバへのエラーレベルのロギングを有効にします。
Syslog 配信	syslog 配信に UDP トランスポートを使用します。

表 13　セキュリティの設定基準

設計オプション	検証済みの選択
Active Directory	AD-admin-group:ntnx-ctrl-admins
Syslog サーバ	infra-az [1..2]-syslog：6514（udp）

Kubernetes クラスタ設計

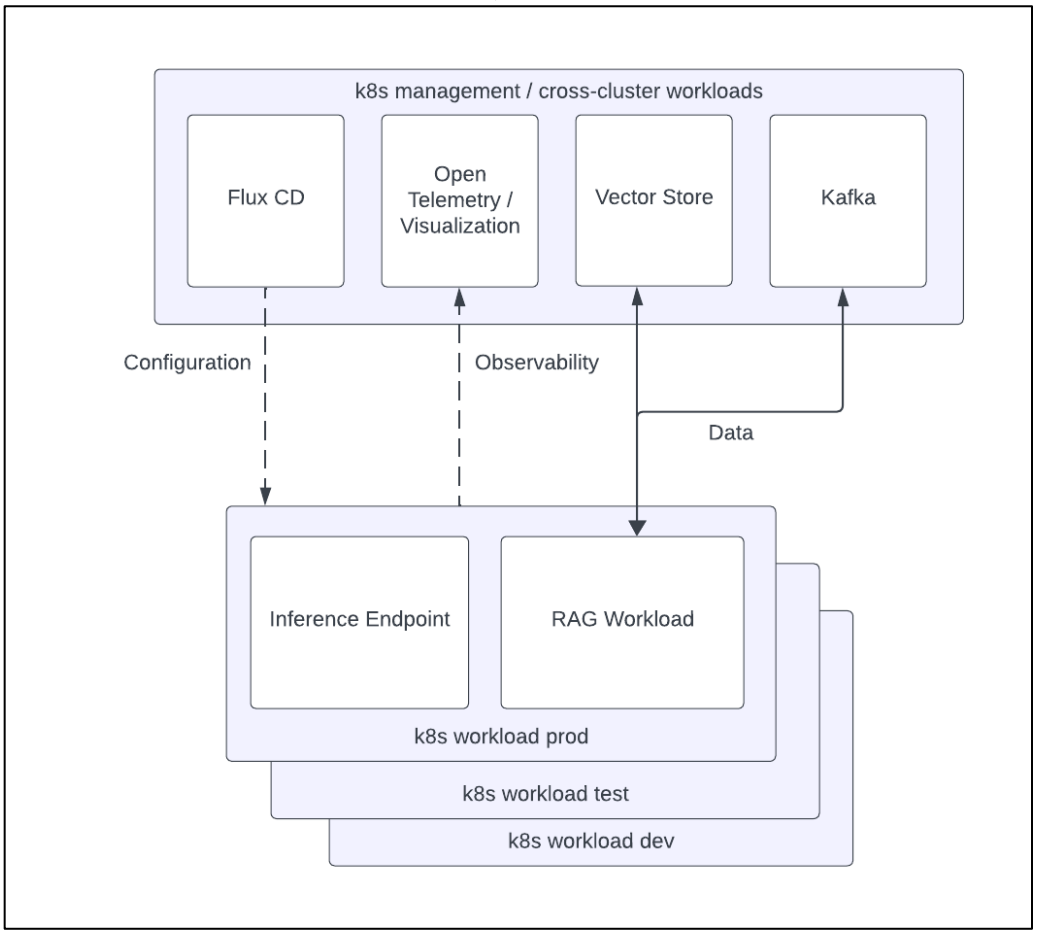
次のセクションでは、Nutanix Kubernetes Engine（NKE）機能を使用し、効率的で安全な運用に不可欠な Kubernetes ツールおよびプラクティスと統合する、Nutanix プラットフォーム上の Kubernetes の設計について説明します。

単一の管理クラスタで、アプリケーション設定、オブザーバビリティ、および Uptrace（ OpenTelemetry ベースのオブザーバビリティ プラットフォーム）、Kafka、Milvus（さまざまなデータ タイプをサポートするクラウドネイティブのベクトル データベース）などのクロスワークロード クラスタ永続アプリケーションのすべてのコンポーネントを実行します。

すべての実稼働 Kubernetes クラスタは、高可用性を確保するために、異なる物理ホストに分散された複数のプライマリ ノードとワーカーノードを使用して、実稼働レベルのクラスタとして設定されます。

Kubernetes 管理クラスタの Flux CD は、Kubernetesワークロードの実稼働クラスタ、テスト クラスタ、および開発クラスタでアプリケーションを構成し、Git リポジトリで指定された構成に合わせた継続的かつ自動化された展開を保証します。ワークロード クラスタは、すべてのクラスタのメトリック、ログ、およびトレースを OpenTelemetry に提供します。また、フロントエンドの使いやすいインターフェイスであるモニタリングは、収集されたデータを可視化し、クエリをデバッグします。Kubernetes ワークロード実稼働クラスタの Retrieval-Augmented Generation ワークロードは、Kubernetes 管理クラスタ上のベクトル ストアと Kafka との間でデータを送受信します。

図 16.　GPT-in-a-Box Kubernetes の概念設計



Kube-VIP はロード バランシングのためのネットワーク接続を処理し、イングレス コントローラは HTTP および HTTPS ルートを使用してサービスへの外部アクセスを管理し、Cert-Manager は TLS 証明書の管理と発行を自動化し、セキュアな通信を強化します。

ワークロード クラスタには、GPU リソース専用のノード プールがあります。このノード プールは、機械学習やデータ処理ワークロードを使用するノードなど、GPU 機能を必要とするポッドをホストします。汚染は、GPU ノードで非 GPU ワークロードがスケジュールされないようにするために GPU ノードで使用されます。対応する許容度を、GPU を必要とするポッドに追加する必要があります。

表 14　NKE スケーラビリティ設計の決定事項を備えた GPT-in-a-Box クラスタ

設計オプション	検証済みの選択
NKE クラスタ タイプ	アクティブ/パッシブ コントロールプレーンで実稼働クラスタを使用します。
コントロール プレーンのサイズ	8 CPU と 16 GB のメモリでコントロール プレーンのサイズを設定します。
初期クラスタ サイズ	12 個の CPU、16 GB のメモリ、および 300 GB のストレージを備えた 3 つのワーカーノードから開始します。
GPU プール サイズ	12 CPU、40 GB のメモリ、および 300 GB のストレージを備えた 2 つのワーカー ノードを使用します。

設計オプション	検証済みの選択
監視	モニタリング コンポーネントを有効にします。

注： NKE クラスタ名には最大 22 文字を使用できます。

この設計では、さまざまなワークロード サイズに対応するために、NKE 管理およびワークロード クラスタのワーカー ノードをスケール アップまたはスケール アウトできます。Nutanix はスケール アウトを推奨しています。GPUノード プール内のワーカーの総数は、物理的にインストールされている GPU の数によって制限されます。

Kubernetes の復元力

NKE クラスタは、コントロール プレーンと etcd 用に複数のノードを実行することで、復元力のあるコントロール プレーンを提供する実稼働レベルのクラスタです。高可用性を確保するために、Kubernetes の展開では複数のレプリカ ポッドを使用し、ポッドの非アフィニティ ルールを実装します。このアプローチにより、ワーカー ノードの更新または障害が発生したイベントでも、サービスの可用性を維持できます。

Kubernetes サービスとイングレス コントローラは、使用可能なすべてのポッドにネットワーク トラフィックを均等に分散することで、ロード バランシングの重要なサービスを実行し、サービスの信頼性とシステム パフォーマンスを向上させます。

Kubernetes ネットワーキング

展開された各クラスタは、ネットワーキングの基本設定を使用します。

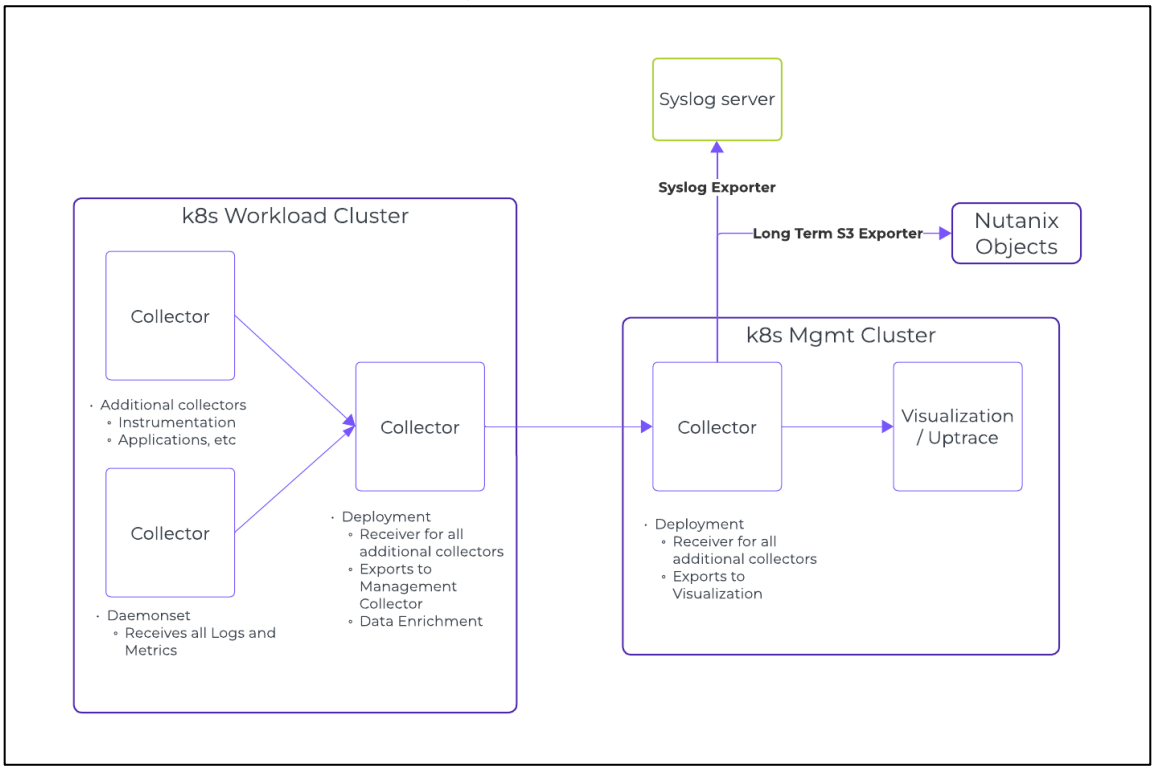
- kube-vip：ロードバランシング サービス
- nginx-ingress：L4 および L7ネットワーク トラフィックのイングレス サービス
- cert-manager： TLS アプリケーション サービスの有効な SSL 証明書を作成するサービス
- 自己署名証明書をもつローカル DNS

Kubernetes モニタリング

Kubernetes およびアプリケーション レベルのモニタリングは、コア インフラストラクチャ モニタリングの概念に基づいています。オブザーバビリティビリティ スタックは、 OpenTelemetryを使用して次のデータを収集し、Kubernetes クラスタ間で移動します。

- Kubelet メトリクス
- ホスト メトリクス
- Kubernetes クラスタ メトリクス
- Kubernetes イベント
- ポッド ログ
- サービス モニターからの Prometheus メトリクス
- インストゥルメンタライゼーションからのアプリケーション固有のデータ

図 17.　　Kubernetes モニタリングの概念設計



Kubernetes 管理クラスタは、オープンソース プロジェクトの Uptrace を使用して、トレース、メトリクス、およびアラートのフロントエンドを提供します。Uptrace を企業製品に置き換えるか、 OpenTelemetry をサポートする別のフロント エンドを使用できます。

Kubernetesワークロード クラスタには、デーモンセット コレクタ (すべてのログとメトリクスを受信) と、展開コレクターにデータを送信するインストルメンテーションおよびアプリケーション用の追加コレクタがあり、管理クラスタ上の展開コレクタでデータの強化とエクスポートを行います。管理クラスタ展開コレクタは、Uptrace または選択した可視化フロントエンドと、Prism Central も使用する外部 syslog サーバにデータをエクスポートします。

Kubernetes バックアップ

展開時に NKE によってインストールされるデフォルトのストレージ クラスは、管理クラスタとワークロード クラスタに永続ストレージを提供します。管理クラスタの永続ボリュームとアプリケーション データは、Velero バックアップ スケジュールによって保護され、単一の S3 バケットに保存されます。アプリケーションの展開は FluxCD を使用した GitOps に基づいているため、アプリケーションをバックアップする必要はありません。

ユーザー バケットのデータから Milvus ベクトル データベースを再構築できます。ユーザー バケット内のデータがエフェメラルの場合は、Milvus Backup ツールを使用して Milvusデータをバックアップおよび復元できます。

大規模言語モデル設計

この基準設計は、オーケストレーション プラットフォームとして NKE を使用して、Kubernetes ベースの Nutanix 人工知能推論エンドポイントで LLM アプリケーションを実行するための堅牢なアーキテクチャを示します。RAG パイプライン アプリケーションを構築するために調整された、包括的でスケーラブルで効率的なフレームワークを提供します。このフレームワークは、最新の LLM テクノロジーとサポート ツールを活用して、開発、展開、およびモニタリングを容易にします。

大規模言語モデルの論理設計

このアーキテクチャの中心となるのは、kserve を介して提供される推論エンドポイントです。この要素は、機械学習モデルの展開と管理、特にリアルタイムの推論のニーズに不可欠です。kserve および Nutanix との統合により、拡張性、信頼性の高いの高いユーザー補助、一貫したパフォーマンスが保証されます。

アーキテクチャはモジュール型であり、各コンポーネントを個別に拡張および更新できます。RAGフレームワークと互換性があり、LLM が Milvus データベースの情報にアクセスできるようになり、生成プロセスが強化されます。

データの取り込みは用途が広く、Kafka を介してバッチベースとイベントベースの両方のアプローチをサポートします。この柔軟性により、システムは定期的な大規模なバッチ アップロードと継続的なリアルタイム データ ストリーミングの両方を管理できます。システムは、Knative 上に構築されたサーバーレス関数を使用してイベントを処理します。イベントは、Kafka を介した Nutanix オブジェクトのイベント通知によってトリガーされます。このセットアップにより、着信データ ストリームの効率的でスケーラブルな処理が保証されます。

取り込まれると、データはラングチェーンを使用してベクトル化され（埋め込みモデルによってベクトル表現に変換され）、Milvus に保存されます。Nutanix Objects は、Milvus の大規模なデータ需要に対応するために必要なスケーラブルなバックエンド ストレージを提供します。kserve でホストされている LLM モデルは、このベクトル化されたデータにアクセスして使用し、言語生成機能を強化します。

ソリューションは、 OpenTelemetry を使用して、システムのオブザーバビリティのためにデータ フロー全体をモニタリングします。さらに、Jupyter Lab は、集中的な計算タスク用の GPU リソースへの直接アクセス、テスト、実験、および開発に役立つ環境を提供します。

大規模言語モデルのリサーチ ワークフロー

LLM は、次のリサーチ ワークフローを使用します。

1. 質問する：エンドユーザーが UI またはチャットボット インターフェイスを介して質問をすると、インタラクションが開始されます。

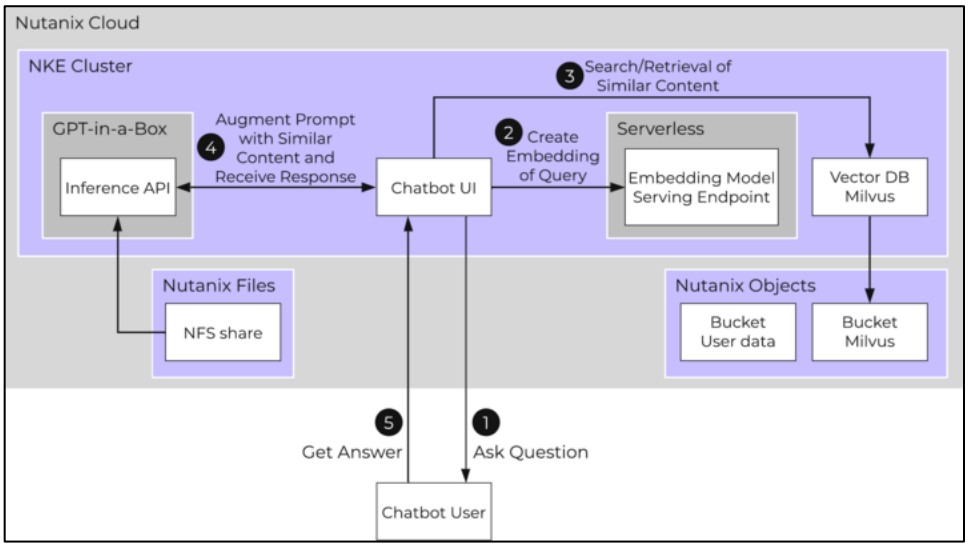
2. クエリ埋め込みの作成：埋め込みモデルは、ユーザーのクエリをベクトル表現に変換します。このプロセスはベクトル化と呼ばれます。

3. 類似コンテキストの検索と取得：類似検索用に特別に設計されたベクトルデータベースには、モデルによって生成されたドキュメントの埋め込みが保存されます。テキストのセマンティックな意味をカプセル化するこれらの埋め込みに基づいて、項目を効率的に検索して取得できます。

4. プロンプトの送信：ワークフローは、データベースから取得した関連コンテキスト情報でユーザのクエリを拡張し、この拡張クエリをプロンプトとして LLM エンドポイントに送信します。LLM は、強化されたクエリを処理し、応答を生成します。

5. 回答の取得：UI またはチャットボット インターフェイスには、ユーザーのクエリに対する回答として LLM の応答が表示されます。

図 18.　　LLM リサーチのワークフロー



ドキュメントの取り込みワークフロー

この設計では、ドキュメントの取り込みに次のワークフローを使用します。

1. ドキュメントのアップロード：新しいドキュメントがバケットに追加されるたびに、Kafka を介してイベント通告が送信されます。

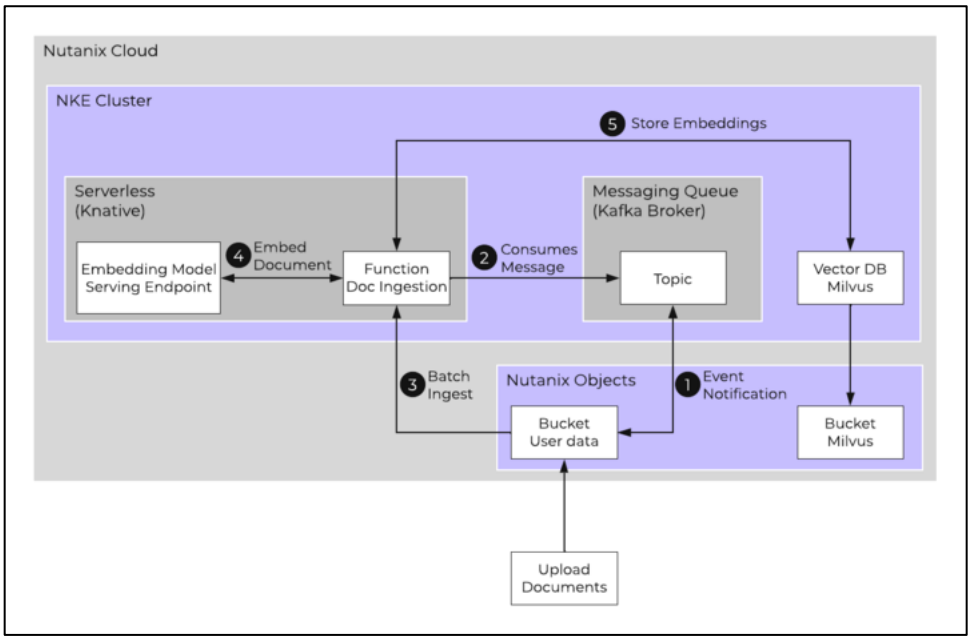
2. Kafkaイベントの処理：Notification（通告）により、ドキュメント取り込みサービスがトリガーされ、その特定のドキュメントの新しい埋め込みが生成されます。

3. バッチの取り込み：ドキュメントの取り込み機能によってドキュメントがダウンロードされます。

4. ドキュメントの埋め込み：埋め込みでは、ドキュメントを小さなセグメントに分割し、埋め込みモデルを使用して各セグメントのベクトル表現を作成します。

5. 埋め込みの保存：生成されたベクトルは、将来の取得と検索のためにベクトル データベースに保存されます。

図 19.　　ドキュメントの取り込みワークフロー



バックアップやディザスタ リカバリ

このソリューションの範囲は単一のスタンドアロン GPT-in-a-Box クラスタであり、Nutanix オブジェクト ストアの S3 バケットのアプリケーション データを外部環境にバックアップする必要があります。Flux CD が展開を処理し、構成データを外部 Git リポジトリに保存するため、NKE 構成データをバックアップする必要はありません。

Nutanix オブジェクトに組み込まれたストリーミング レプリケーション メカニズムを使用して、バケット レベルのデータを GPT-in-a-Boxクラスタ外の別の S3 オブジェクト ストアに複製できます。既存のバックアップ ソリューションを使用して、永続的なアプリケーション データをバックアップし、GPT-in-a-Box クラスタの外部に保存することもできます。

ソリューションの導入

この章の内容は、次のとおりです。

- インフラストラクチャ展開
- GitOps を使用した GPT-in-a-Box の展開

インフラストラクチャ展開

このセクションは、GPT-in-a-Box ソリューションをインストールするための前提条件であり、次の主要なセクションと手順に分かれています。

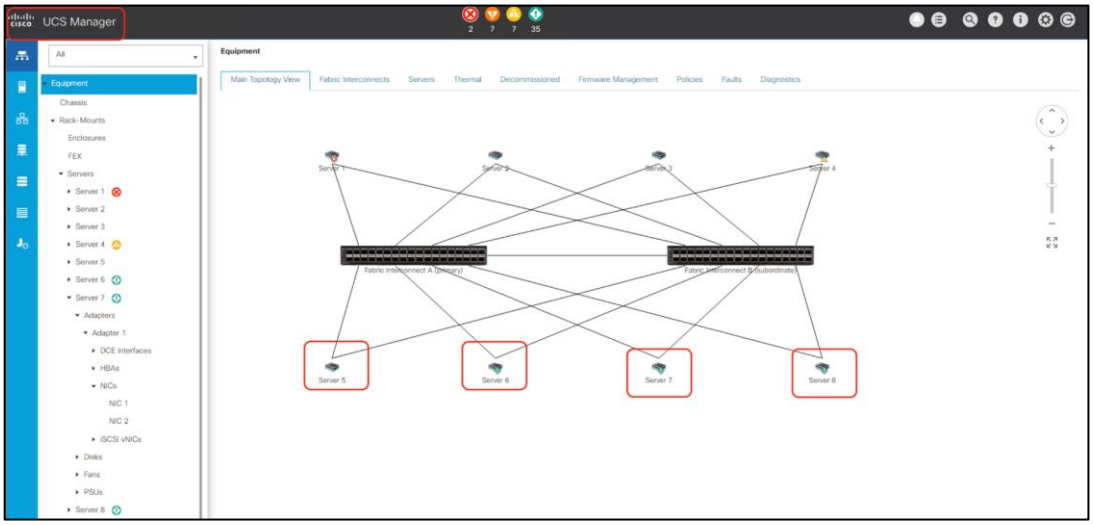
1. Nutanix クラスタを使用した AHV ベースの CCHC のインストール
2. NVIDIA Grid ドライバのインストール
3. Nutanix ファイルのインストールと構成
4. Nutanix オブジェクトのインストールと構成
5. Nutanix Kubernetes クラスタのインストールと構成

手順 1。 Nutanix クラスタを使用した AHV ベースの CCHC のインストール

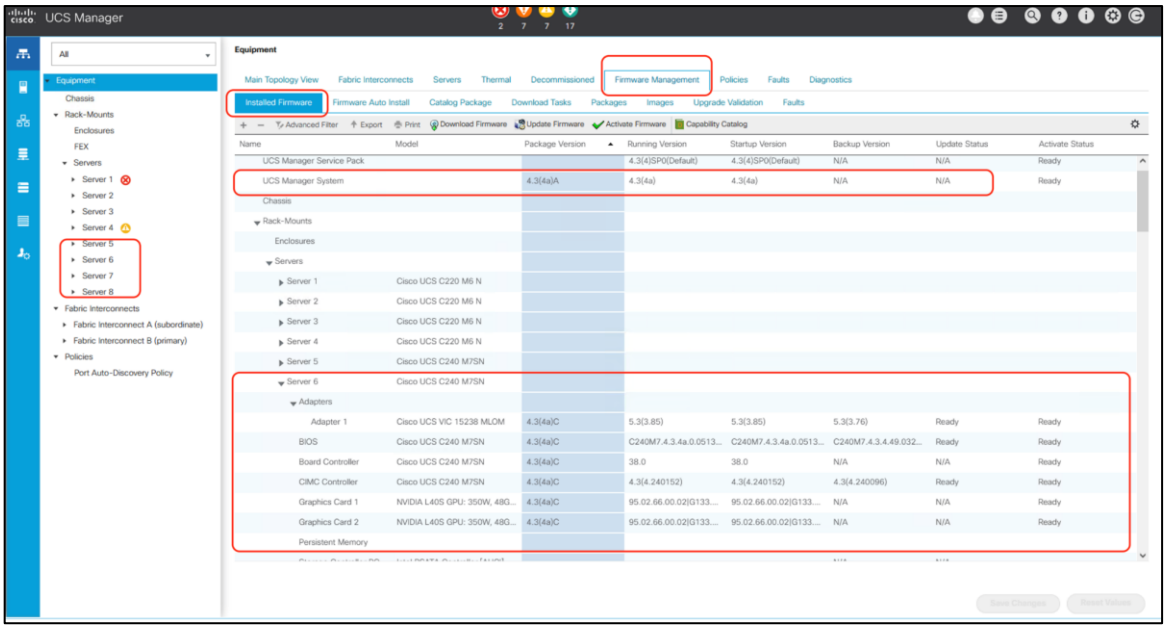
このソリューションには、Nutanix クラスタを備えた CCHC の最小 4 ノードが必要です。各クラスタ ノードは、HCIAF240C M7 All-NVMe サーバを搭載した 2x NVIDIA L40S GPU で有効になっています。サーバ ノードの仕様の詳細については、「ソリューション設計」のセクションを参照してください。

注： Nutanix クラスタを使用した AHV ベースの CCHC の完全なインストールプロセスは、このドキュメントの範囲外です。このセクションでは、主要な検証手順と参考資料について詳しく説明します。

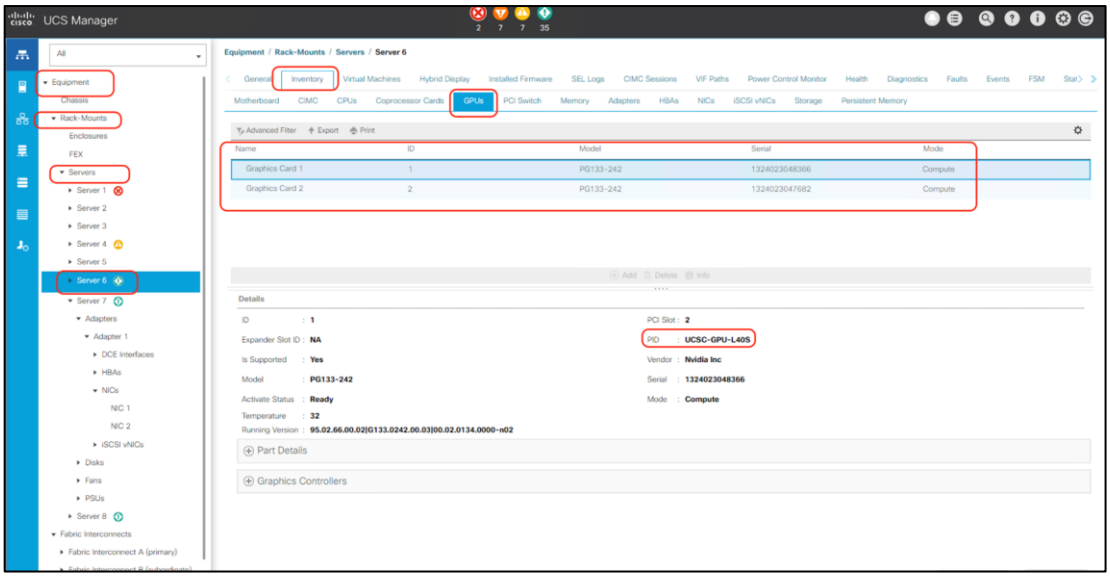
ステップ 1. 4 つの GPU 対応 HCIAF240C M7 All-NVMe ノードが Cisco UCS Manager で検出されていることを確認します。接続の詳細については、「インフラストラクチャとネットワーク設計」セクションを参照してください。



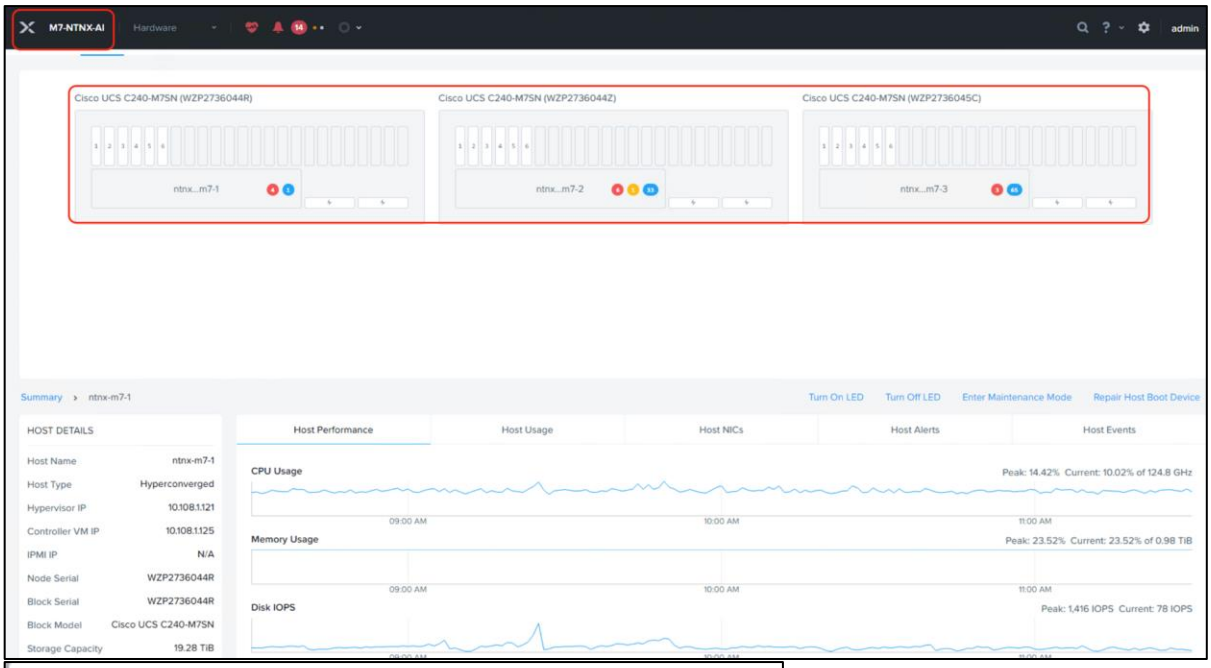
ステップ 2. UCS ファームウェアが 4.3(4a) 以降であることを確認します。

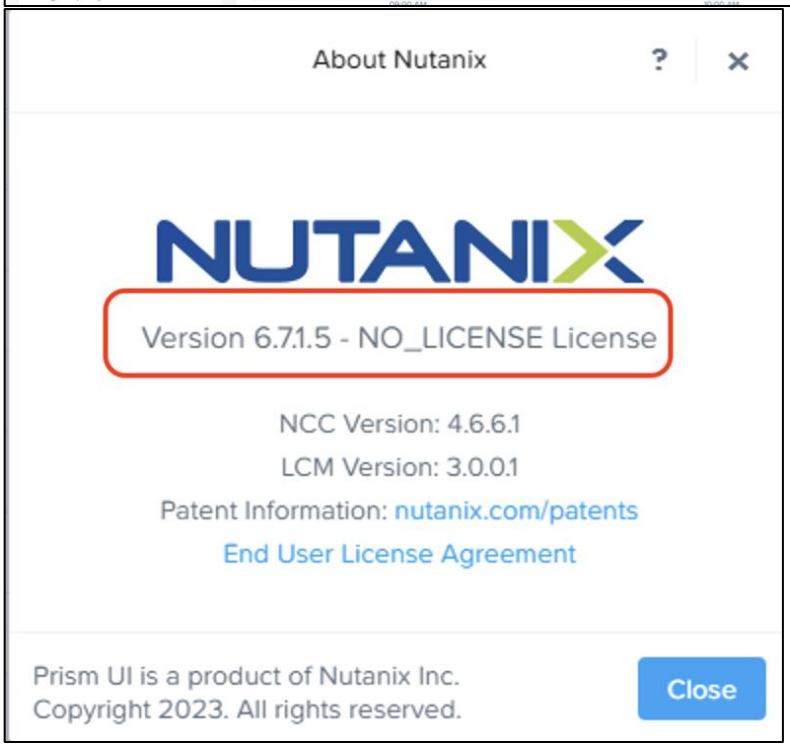


ステップ 3. [機器（Equipment）] > [ラックマウント（Rack-mounts）] > [サーバ（Servers）] > [サーバ(x)（Servers(x)）] に移動します。右側のナビゲーション ウィンドウで、[インベントリ（Inventory）] > [GPU（GPU）] タブの順にクリックします。少なくとも 1 つの Nvidia L40S GPU が表示されていることを確認します。



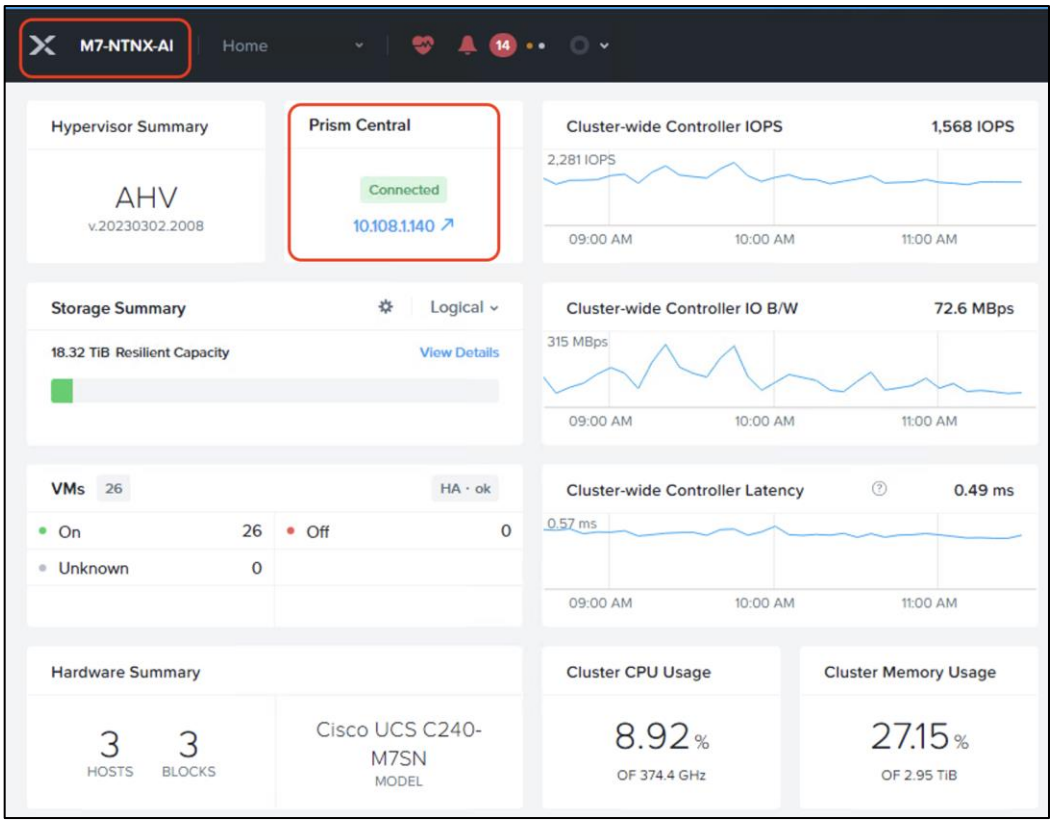
ステップ 4. AOS 6.7.1 を使用して Nutanix クラスタをインストールします。クラスタを正常にインストールには、『フィールド ガイド』を参照してください。





注： 上記のスクリーンショットには、3 つのノードが表示されています。障害に対するレジリエンシを有効にするには、少なくとも 4 つのノードから開始することを強くお勧めします。

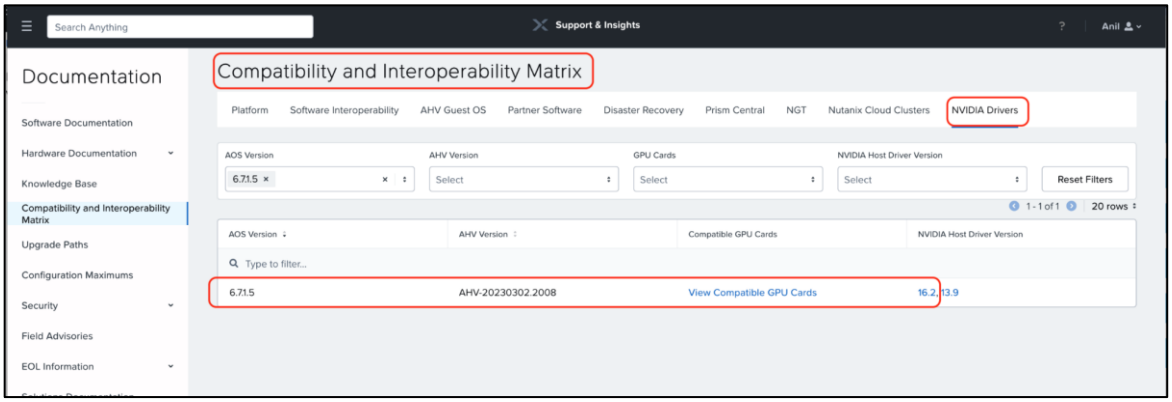
ステップ 5. Nutanix クラスタに展開された Prism Central インスタンスをプロビジョニングし、既存のクラスタを Prism Central に登録します。お客様は、GPT-in-a-Box クラスタに Prism Central を展開するか、既存の Prism Central インスタンスを使用するかを選択できます。



手順 2。 NVIDIA Grid ドライバのインストール

次の手順では、L40S GPU で構成されたクラスタ ノードに NVIDIAドライバを展開するプロセスについて詳しく説明します。

ステップ 1. クラスタにインストールされている AOS ビルドに従って GPU ドライバをダウンロードします。これは、Nutanix ポータルの 互換性および相互運用性マトリックス から取得できます。AOS 6.7.1.5 を搭載した既存のクラスタは、lcm_nvidia_20230302.2008_535.129.03 と互換性があります。



ステップ 2. Nvidia GPU が PCI デバイスとして使用可能であることを確認します。root/nutanix/4u または管理者が設定したパスワードを使用して、AHV（ノード ホスト IP）にログインします。lspci | を実行します。 grep -I nvidia を実行し、GPU がホストにインストールされていることを確認します。

```
[root@ntnx-m7-3 ~]# lspci | grep -i nvidia
3d:00.0 3D controller: NVIDIA Corporation Device 26b9 (rev a1)
ab:00.0 3D controller: NVIDIA Corporation Device 26b9 (rev a1)
[root@ntnx-m7-3 ~]#
```

ステップ 3. NutanixノードCVM（Nutanix/Nutanix/4u）にログインするか、管理者が設定したパスワードを使用して、tarファイルを /home/Nutanix にコピーします。

ステップ 4. ドライバ `install_host_package -r lcm_nvidia_20230302.2008_535.129.03.tar.gz` をインストールします。このドライバは、ローリング再起動プロセスによってすべてのノードにインストールされます。

```
nutanix@NTNX-WZP2736045C-A-CVM:10.108.1.127:~$ install
install              install_host_package   install-info          installkernel         install_secure_ntp     install_secure_syslog  install.sh
nutanix@NTNX-WZP2736045C-A-CVM:10.108.1.127:~$ install_host_package -r lcm_nvidia_20230302.2008_535.129.03.tar.gz

VMs using a GPU must be powered off if their parent host is affected by this
install. If left running, these VMs will be automatically powered off when
installation begins on their parent host, and powered back on after the driver
install is completed.

VMs using vGPU will be migrated if their parent host is affected by this
install. However, some vGPU VMs might be automatically powered off due to lack
of resource when installation begins on their parent host, and powered back on
after the driver install is completed.

If the change in the host driver can make the guest driver incompatible,
it is recommended to first uninstall the GPU drivers in the affect guest VMs
before you proceed to modify the host driver. Please refer to NVIDIA documents
for compatibility between guest GPU drivers and host GPU drivers.

Install now (yes/no) yes
2024-04-17 01:27:33,013Z INFO Dummy-1 ahv_host_agent.py:766 Event listener thread started
2024-04-17 01:27:34,156Z ERROR MainThread command_executor.py:677 Killed by signal 9
2024-04-17 01:27:35,653Z INFO MainThread kvm_upgrade_helper.py:566 Found release marker: el8.nutanix.20230302.2008
2024-04-17 01:27:35,653Z INFO MainThread kvm_upgrade_helper.py:244 Current hypervisor version: el8.nutanix.20230302.2008
2024-04-17 01:27:36,974Z INFO MainThread install_host_package:373 No supported GPU hardware found for host 10.108.1.121
Continue installing on this node anyway?
Install now (yes/no) yes
2024-04-17 01:31:04,444Z INFO MainThread install_host_package:373 No supported GPU hardware found for host 10.108.1.122
Continue installing on this node anyway?
Install now (yes/no) yes
2024-04-17 01:31:12,708Z INFO MainThread install_host_package:435 Starting rolling restart for hypervisors in the cluster

2024-04-17 01:31:12,744Z INFO MainThread install_host_package:449 Rolling restart initiated, Check status by running cmd: u'ecli task.get 10d22151-7c76-49f1-76c5-da3068aa500b'
nutanix@NTNX-WZP2736045C-A-CVM:10.108.1.127:~$
```

ステップ 5. `nvidia-smi` を実行し、ドライバのバージョンと検出された GPU リソースを含むテーブルの出力を調べます。

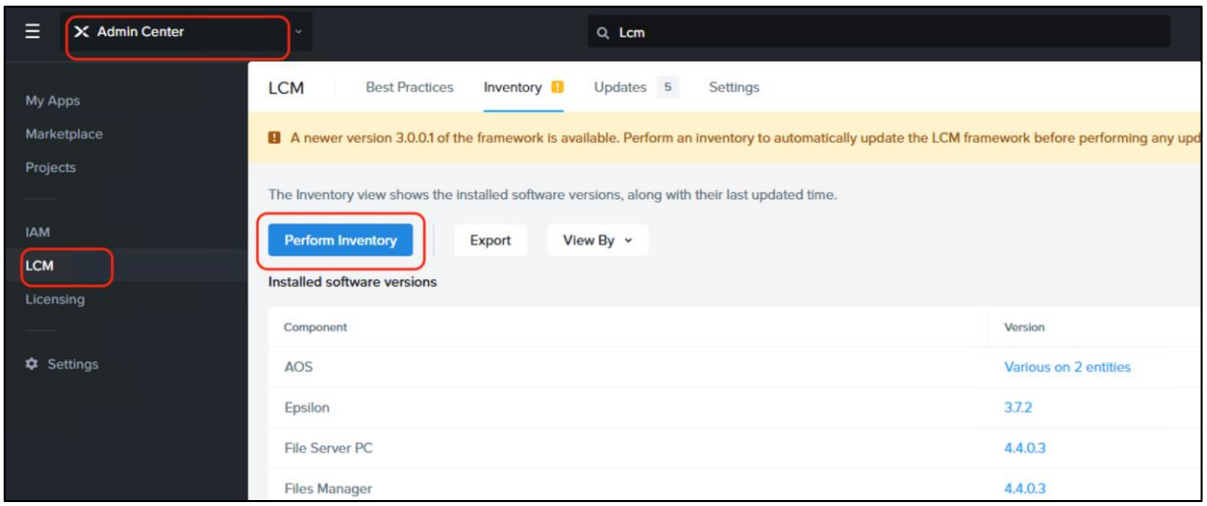
```
[root@ntnx-m7-3 ~]# nvidia-smi
Wed Apr 17 02:18:06 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version: N/A      |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA L40S                    On  | 00000000:3D:00.0 Off |                    0 |
| N/A   39C    P8              41W / 350W |      0MiB / 46068MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA L40S                    On  | 00000000:AB:00.0 Off |                    0 |
| N/A   38C    P8              41W / 350W |      0MiB / 46068MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
[root@ntnx-m7-3 ~]#
```

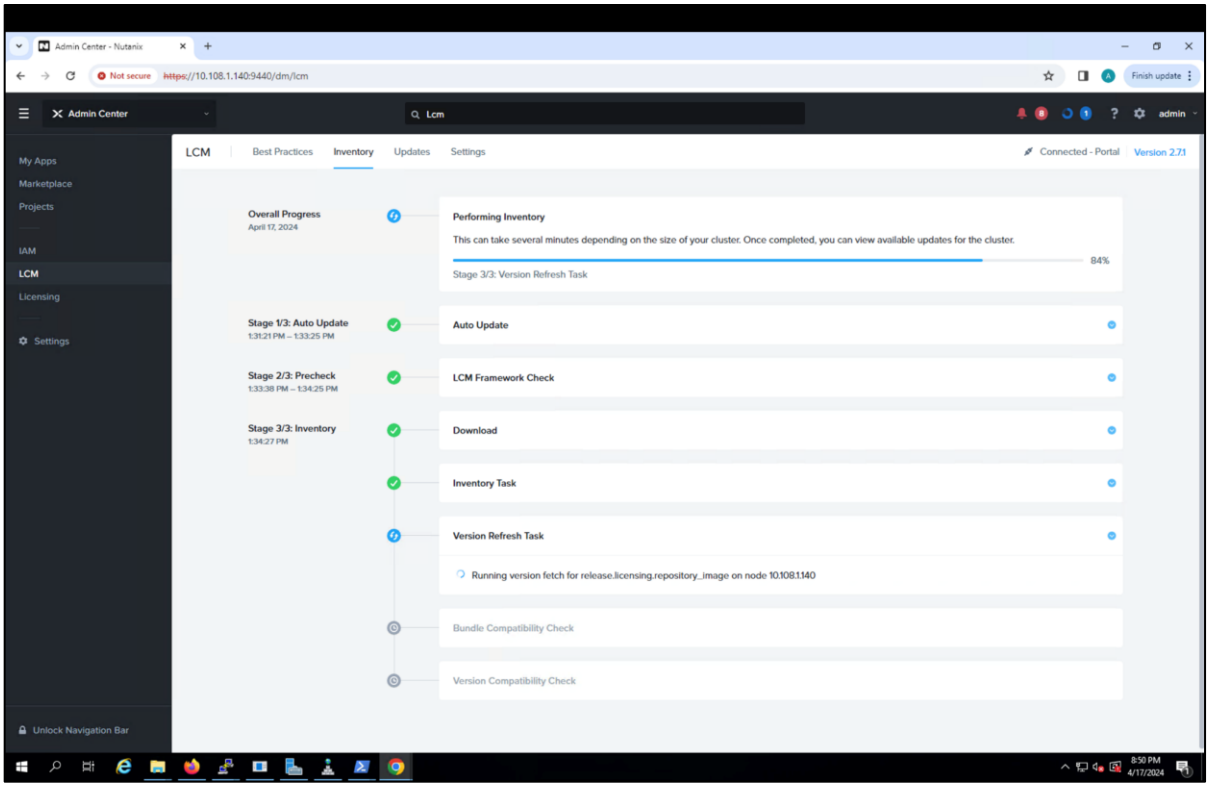
手順 3。　Nutanix ファイルのインストールと構成

この手順では、Prism Central から Nutanix ファイルを有効にして構成する手順の概要を示します。Nutanix ファイルは、GPT-in-a-Box クラスタと同じクラスタで実行されます。Nutanix ファイル アーキテクチャの詳細については、Nutanix ポータルの「ファイル」を参照してください。

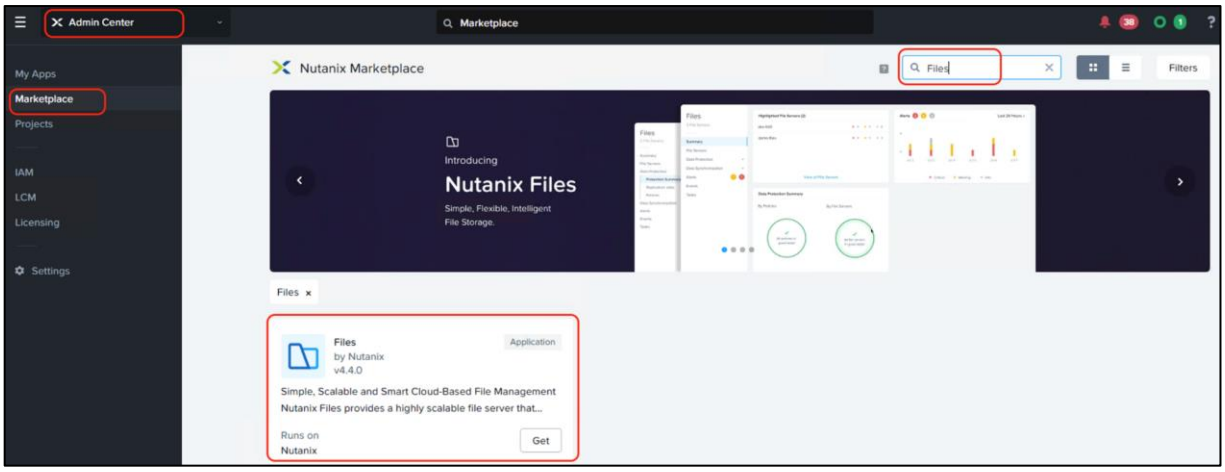
ステップ 1.　Prism Central にログインし、ネーム サーバーと NTP サーバーが更新されていることを確認します。

ステップ 2.　Prism Central に移動し、[Admin Central] を選択し、[LCM] を選択して、[インベントリの実行（Perform Inventory）] をクリックします。





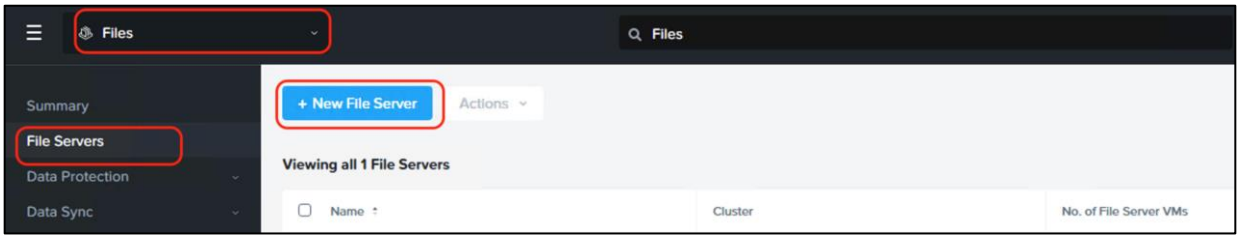
ステップ 3.　[マーケットプレイス（Marketplace）] を選択し、[ファイル（Files）] を検索します。



ステップ 4. [ファイル（Files）] から、[取得（Get）] をクリックします。これにより、ファイル サービスが有効になります。

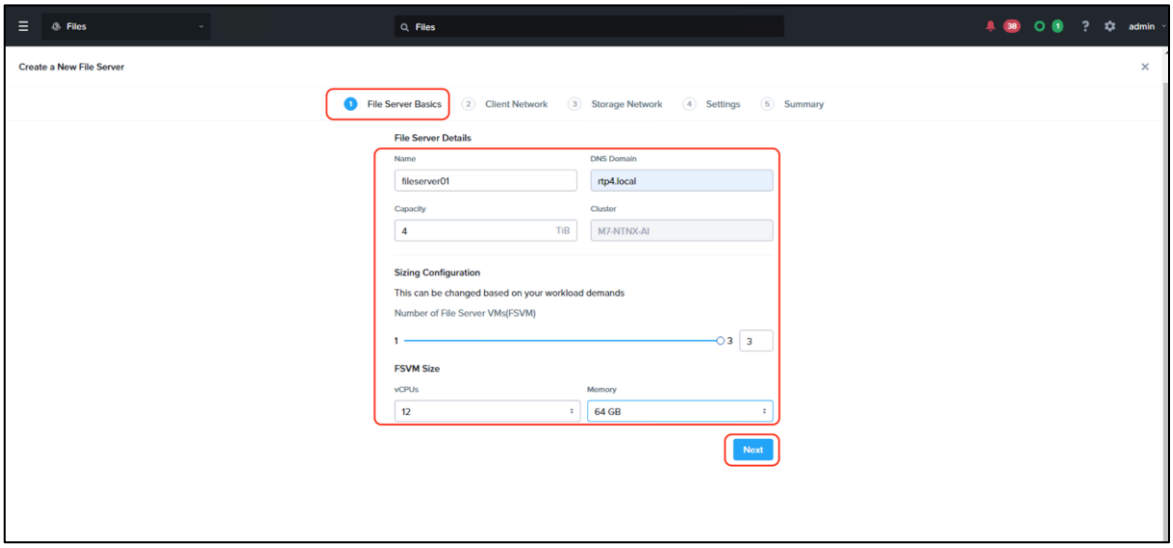
ステップ 5. [ファイル（Files）] を選択し、ファイル サービスを有効にします。

ステップ 6. [ファイル（Files）] から [ファイル サーバ（File Servers）] に移動し、[+ 新しいファイルサーバ（+ New File Server）] を選択します。

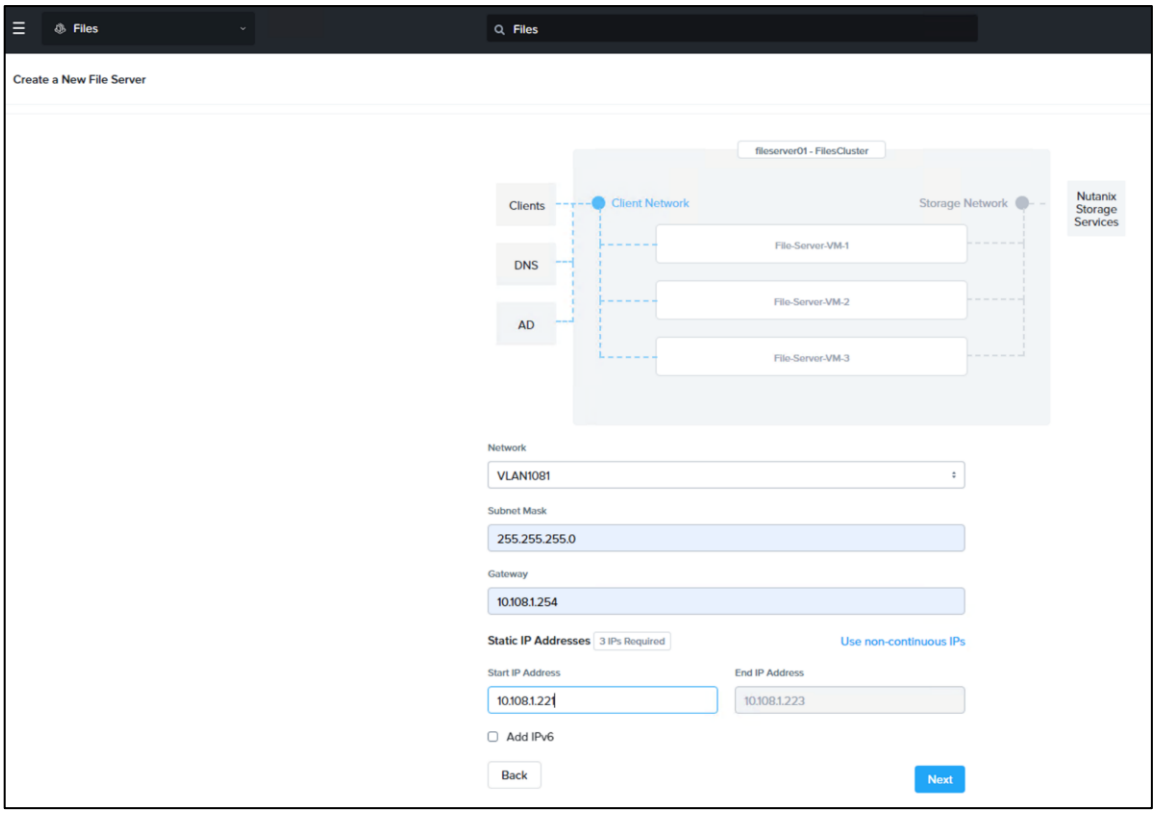


ステップ 7. [ファイル サーバの作成（Create a File Server）] ウィンドウで、[GPT-in-a-Box] クラスタを選択し、[File Server 4.4.0.3] をオンにして、[続行（Proceed）] をクリックします。

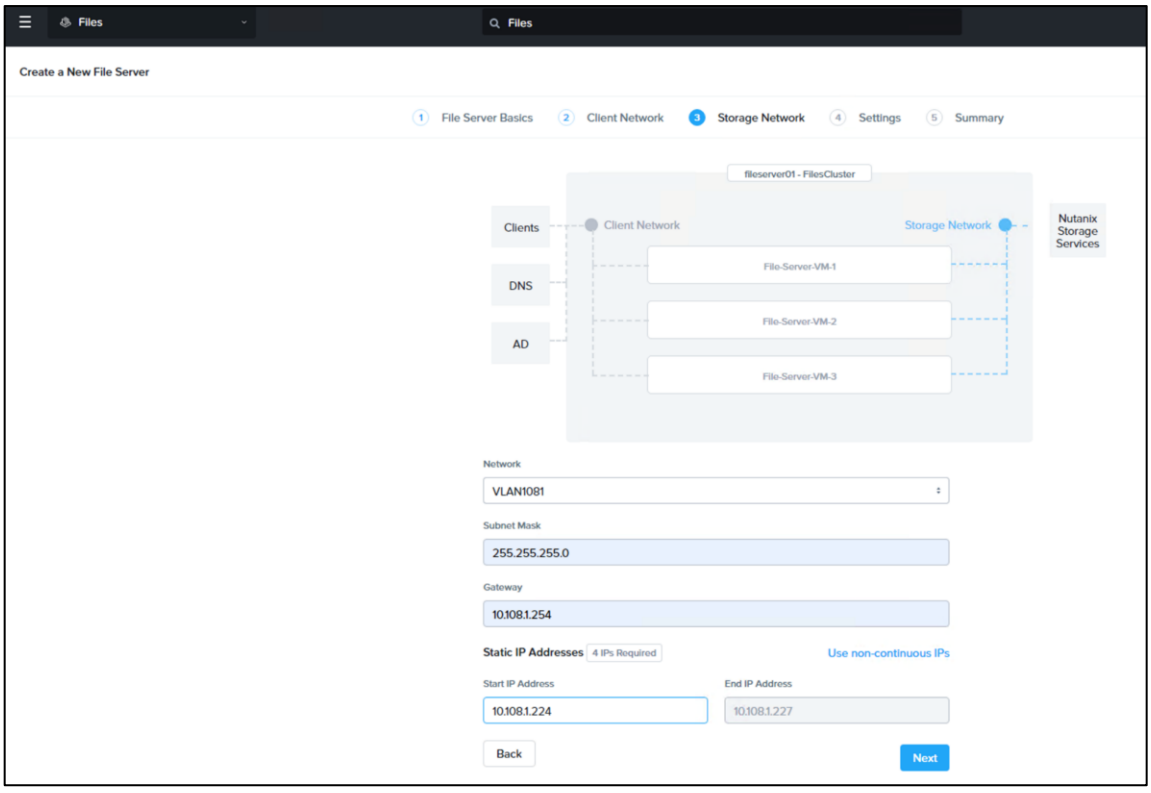
ステップ 8. ファイル サーバ名、 Domain Name System (DNS)ドメイン、キャパシティ（4TiB）、FSMV（12vCPU、64GBメモリ）を 3 として入力し、[次へ（Next）] をクリックします。これらの仕様については、「Nutanix Files and Objects Design Decision」のセクションを参照してください。



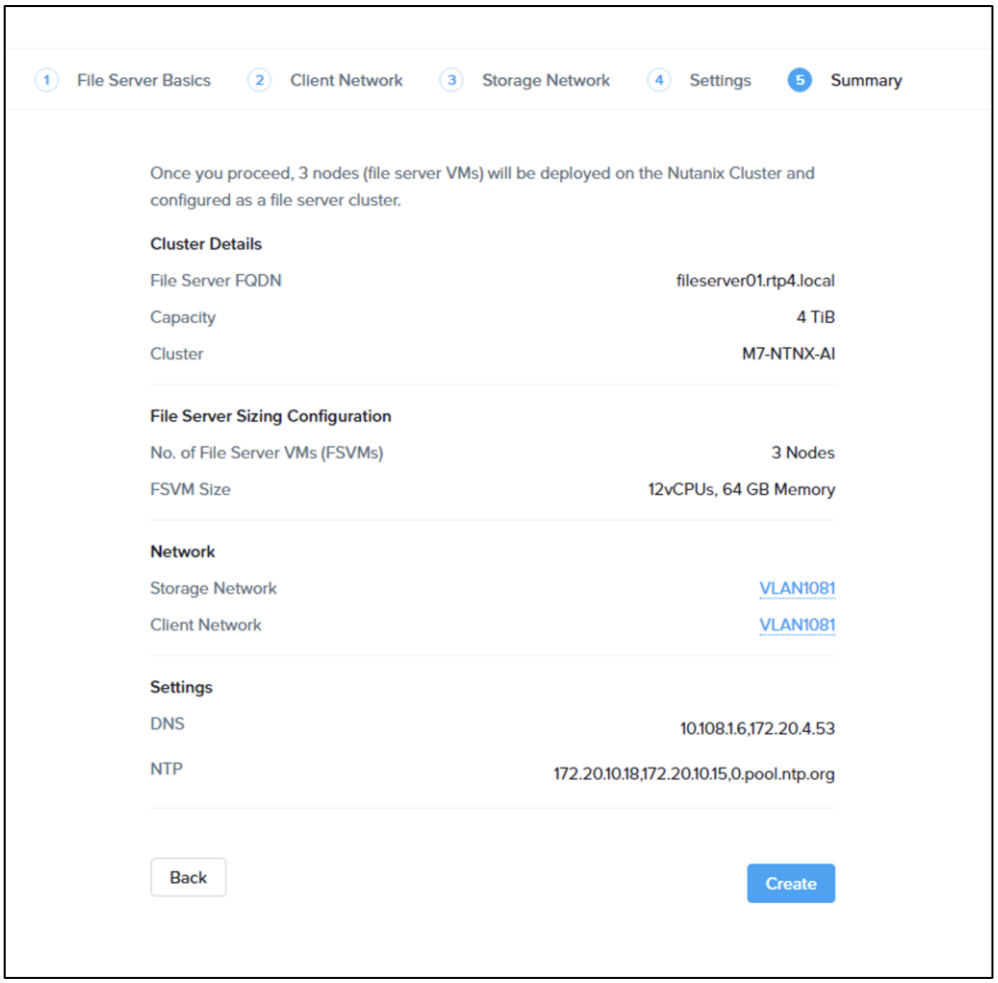
ステップ 9. クライアント ネットワーク、サブネット マスク、ゲートウェイ、およびクライアントネットワークの IP を入力します。



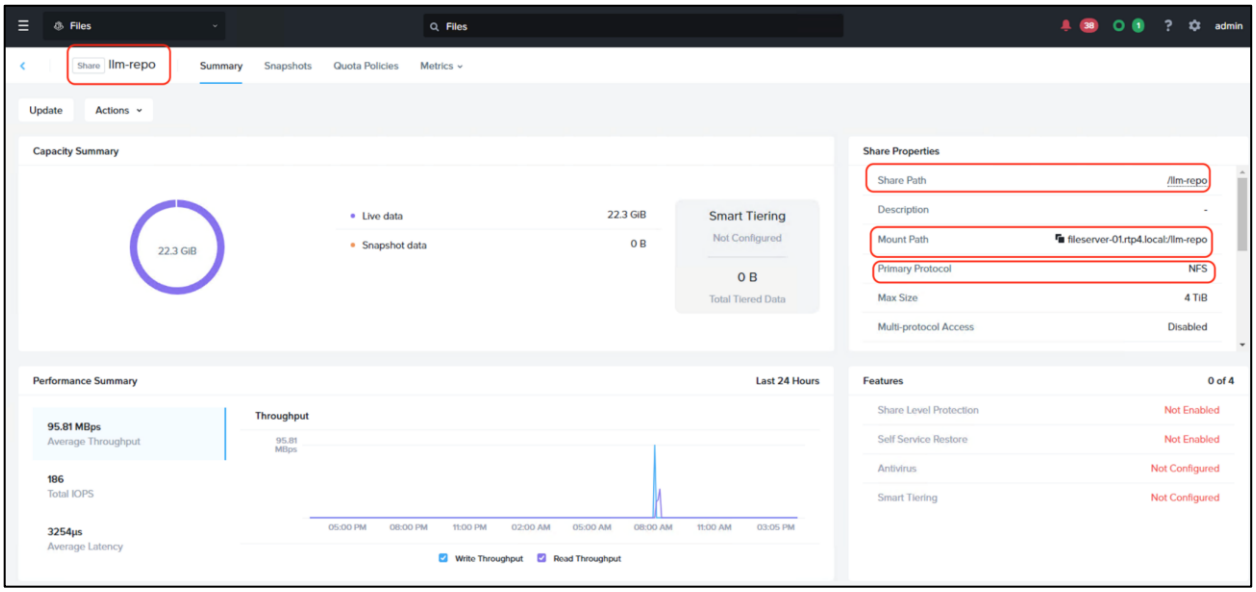
ステップ 10. CVM ネットワークと同じ [ストレージ ネットワーク] を選択し、[次へ（Next）] をクリックします。



ステップ 11. DNS サーバーと NTP サーバがクラスタの設定に従って事前に入力されている場合は、[次へ（Next）] をクリックします。サマリーを確認して、[作成（Create）] をクリックします。



ステップ 12. ファイル サーバが作成されたら、次のようにデフォルト設定で NFS 共有「llm-repo」を作成します。



手順 4。 Nutanix オブジェクトのインストールと構成

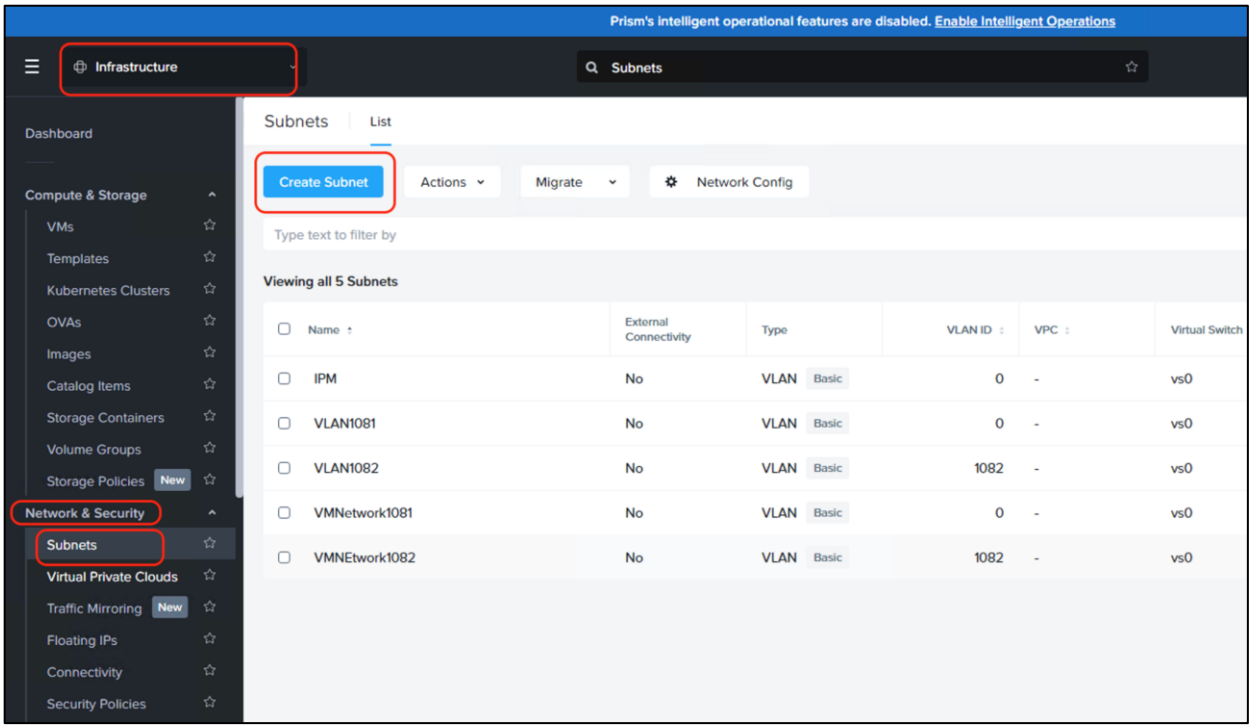
この手順では、Prism Central から Nutanix オブジェクトを有効にして設定する手順の概要を示します。Nutanix ファイルは、GPT-in-a-Box クラスタと同じクラスタで実行されます。Nutanix ファイル アーキテクチャの詳細については、Nutanix ポータルの 『Nutanix Objects User Guide』 を参照してください。

また、Nutanix University で「Nutanix オブジェクトストアの作成に関するビデオ」を表示することもできます。

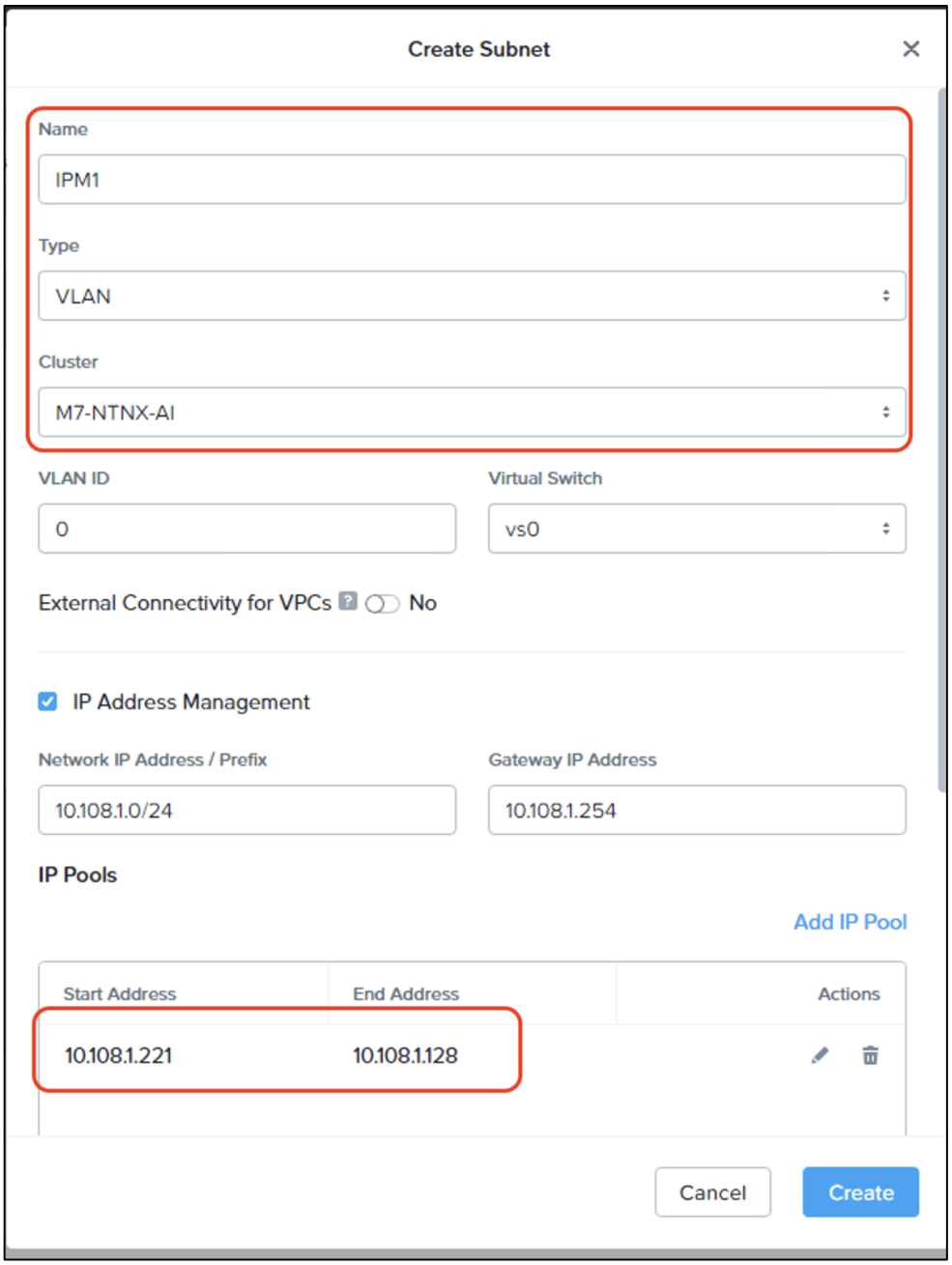
ステップ 1. Prism Central にログインし、ネーム サーバーと NTP サーバーが更新されていることを確認します。

ステップ 2. オブジェクト ストアを作成する前に、IP アドレス管理（IPAM）ネットワークを作成します

ステップ 3. Prism Central から [インフラストラクチャ（Infrastructure）] に移動し、[ネットワークとセキュリティ（Network & Security ）] > [サブネット（Subnet）] の順に選択します。



ステップ 4. 名前、タイプ = VLAN と入力し、GPT-in-a-Box クラスタを選択し、仮想スイッチと VLAN ID = 0 を選択し、IP アドレス管理を有効にして、使用可能な Ips の範囲を指定します。[Create] をクリックします。

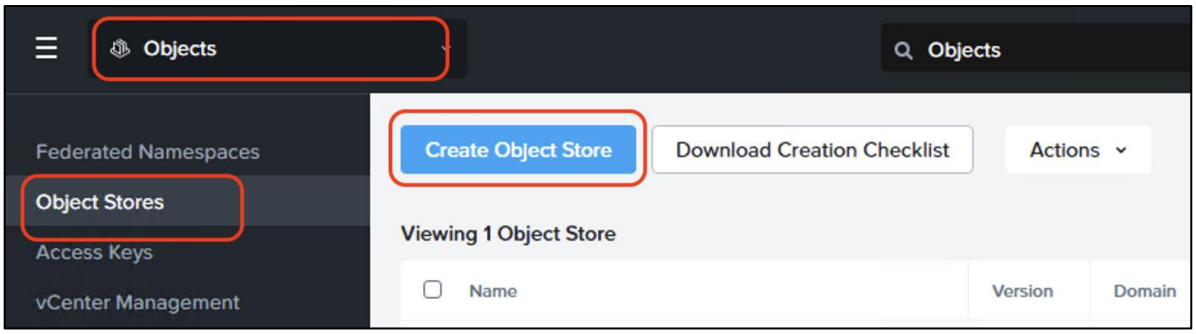


注： IPAM は、後続のセクションで使用されます。IP 範囲は編集できないため、単一の大きな IP 範囲ではなく、複数の小さな IP 範囲を設定することをお勧めします。

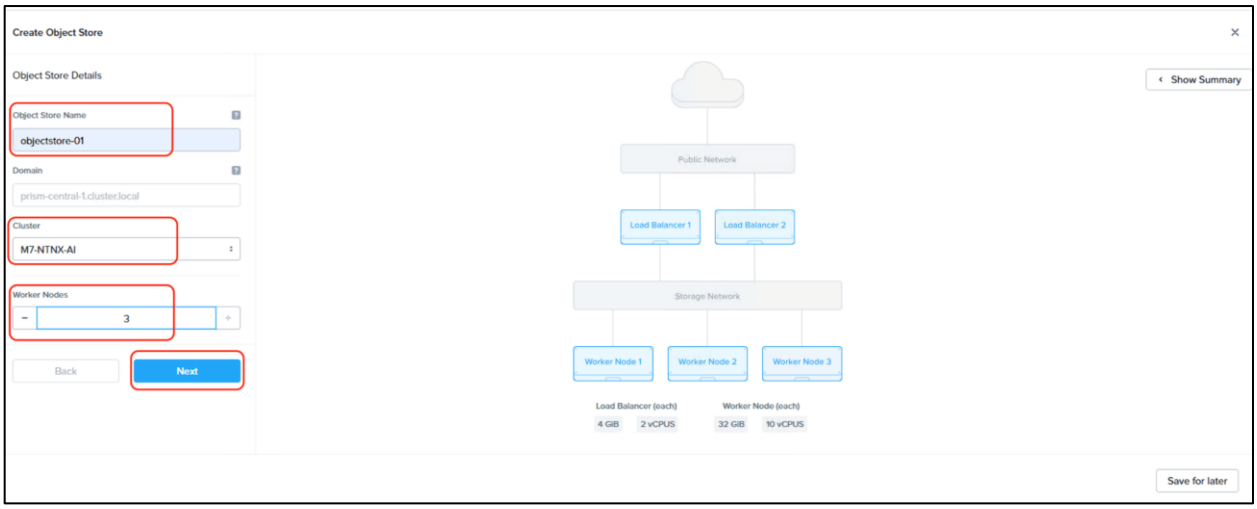
ステップ 5. [マーケットプレイス（Marketplace）] を選択し、オブジェクトを検索します。

ステップ 6. [オブジェクト（Objects）] に移動し、オブジェクト サービスを有効にします。

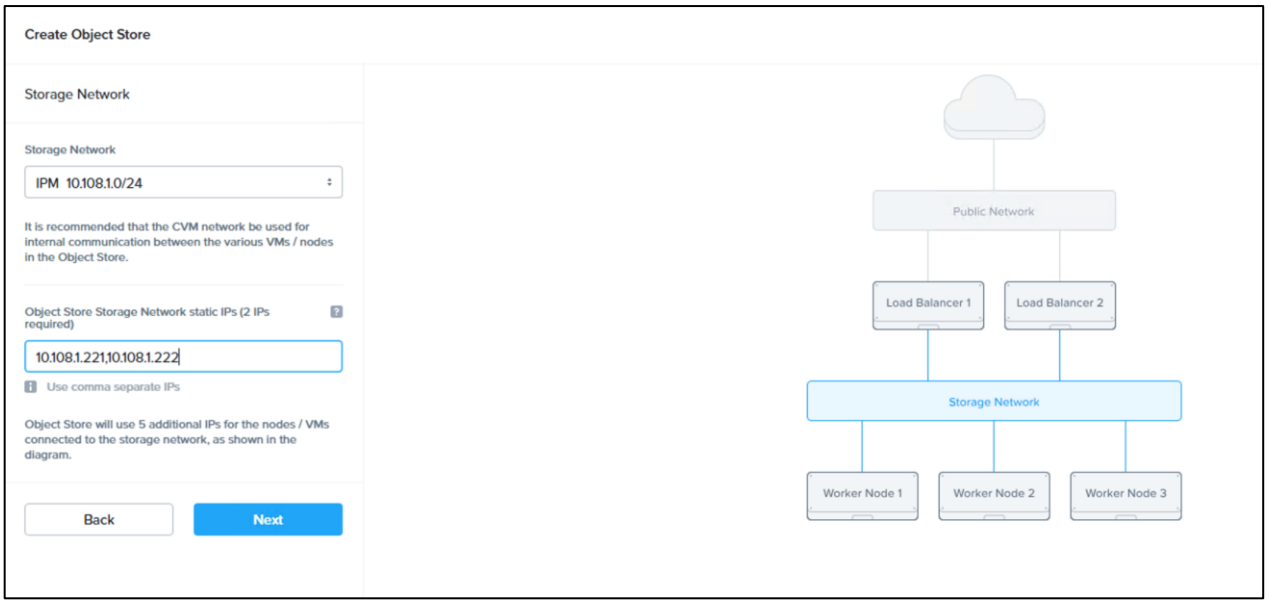
ステップ 7. [オブジェクト ストア（Object Stores）] を選択し、[オブジェクト ストアの作成（Create Object Store）] をクリックします。



ステップ 8. 前提条件ウィンドウで [続行（Continue）] をクリックします。[オブジェクト ストアの作成（create Object store）] ウィンドウで、オブジェクト ストアの名前を入力し、[GPT-in-a-Box クラスタ（GPT-in-a-Box クラスタ）] を選択して、作業ノードのサイズを 3 に編集します。[次へ（Next）] をクリックします。



ステップ 9. [ストレージ ネットワーク] を選択します。このソリューションでは、CVM ネットワークが使用されます。[次へ（Next）] をクリックします。



注： ストレージ ネットワーク内の IP は、IPAM DHCP 構成の範囲外である必要があります。

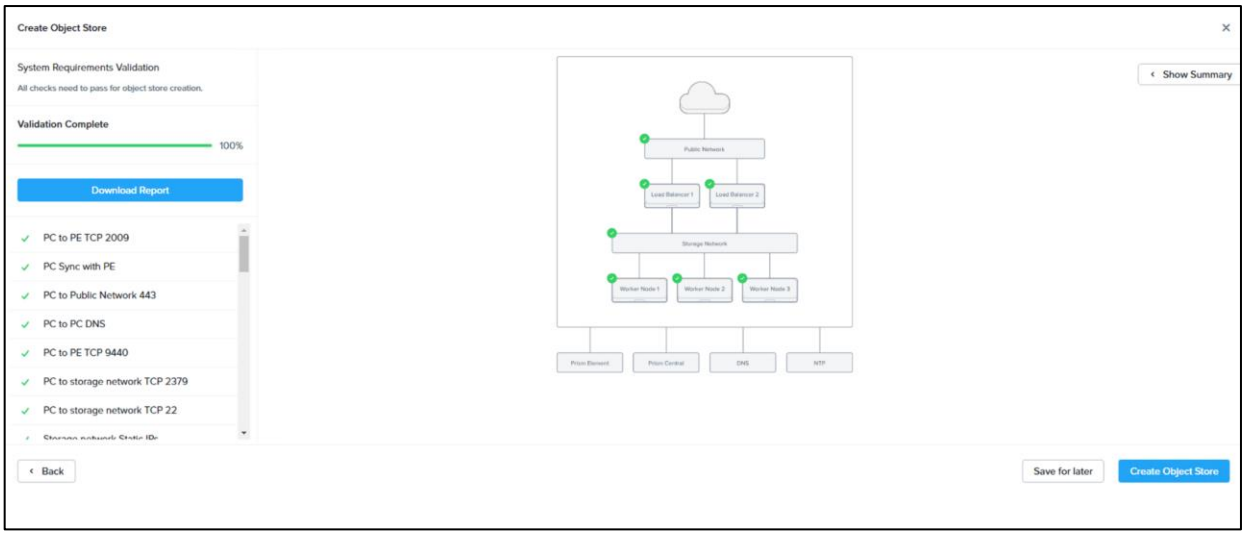
注： この展開では、ストレージ ネットワークとクライアント ネットワークが同じ VLAN 上にあるため、IPAM DHCP 構成は 1 つです。

ステップ 10. [クライアント ネットワーク（クライアント ネットワーク ）] を選択し、クライアント ネットワークで使用可能な IP を入力します。

注： クライアント ネットワークの IP は、IPAM DHCP 構成の範囲外である必要があります。

注： この展開では、ストレージ ネットワークとクライアント ネットワークが同じ VLAN 上にあるため、IPAM DHCP 構成は 1 つです。

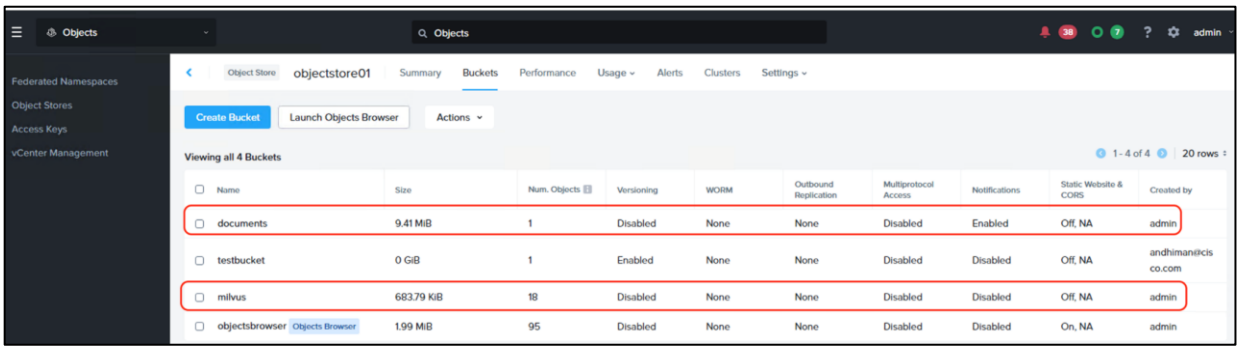
ステップ 11. [保存して続行（Save and Continue）] をクリックします。検証に合格することを許可し、[オブジェクト ストアの作成（Create Object Store）] をクリックします。



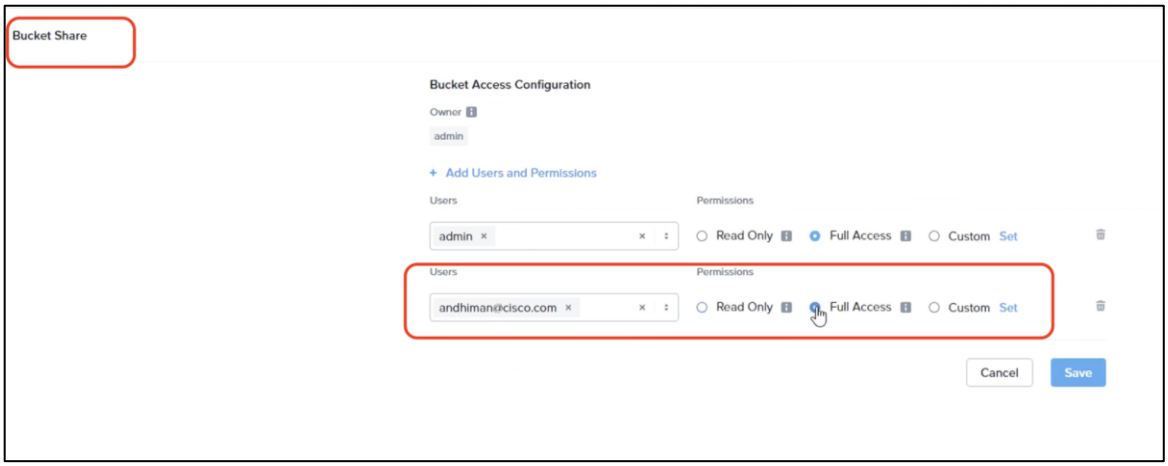
ステップ 12. ソリューションに必要なアクセス キーと秘密キーを保護します。

ステップ 13. 作成したオブジェクト ストアに移動し、[バケットの作成（Create Bucket）] をクリックします。

ステップ 14. 2 つのバケット「milvus」と「ドキュメント」を作成します。milvus バケットは、milvus ベクトル db が埋め込みを保存するために使用され、ドキュメント バケットは、RAG 参照アプリケーションを介してアップロードされたドキュメントとナレッジベースを保存します。



ステップ 15. ドキュメントバケットを選択し、オブジェクトストアの作成時に構成されたユーザーにフル権限を追加します。同様に、milvus バケットについても同じことを行います。



手順 5。 Nutanix Kubernetes クラスタのインストールと構成

この手順では、Nutanix Kubernetes Engine（NKE）を使用して Kubernetes クラスタをインストール手順の概要を示します。GPT-in-a-Box リファレンス デザインには、次の 2 つのクラスタが必要です。

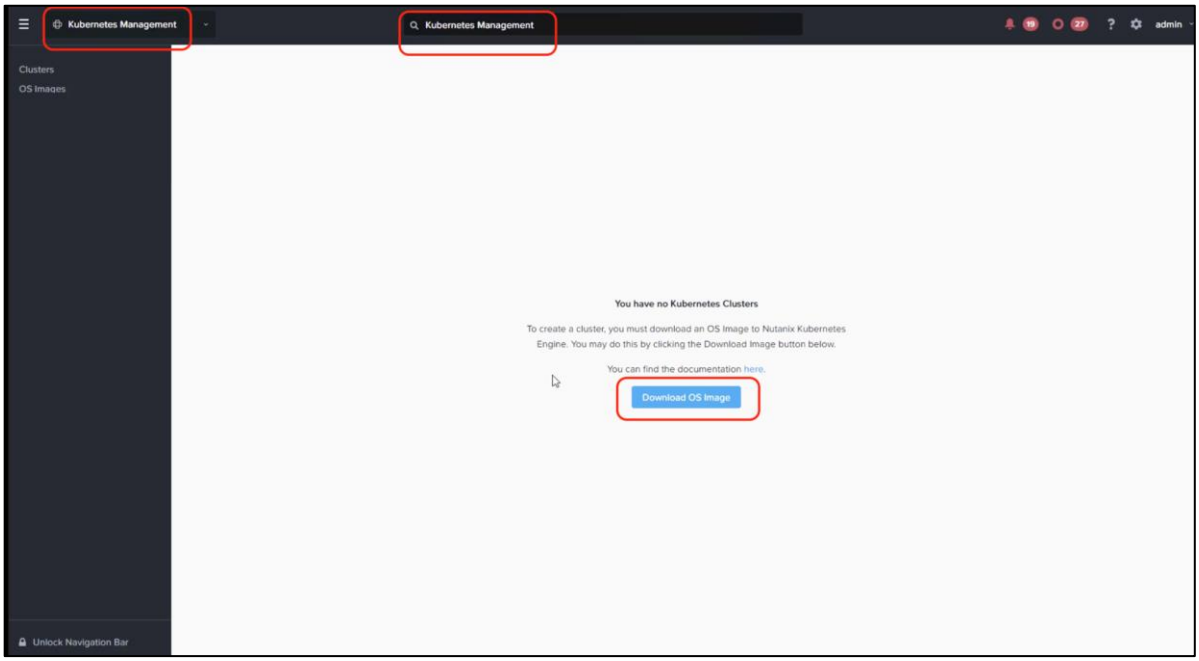
Nutanix 管理クラスタ：単一の管理クラスタで、アプリケーション設定、オブザーバビリティ、および Uptrace（OpenTelemetry ベースのオブザーバビリティ プラットフォーム）、Kafka、Milvus（さまざまなデータ型をサポートするクラウドネイティブのベクトル データベース）などのクロスワークロード クラスタ永続アプリケーションのすべてのコンポーネントを実行します。

Nutanix ワークロード クラスタ： ワークロード クラスタには、GPU リソース専用のノード プールがあります。このノードプールは、機械学習やデータ処理ワークロードを持つノードなど、GPU 機能を必要とするポッドをホストします。

注： 管理およびワークロード クラスタを構成する前に、2x 10 個の IP プールのセットが IPAM（DHCP を使用した IP アドレス管理）で構成されていることを確認します。

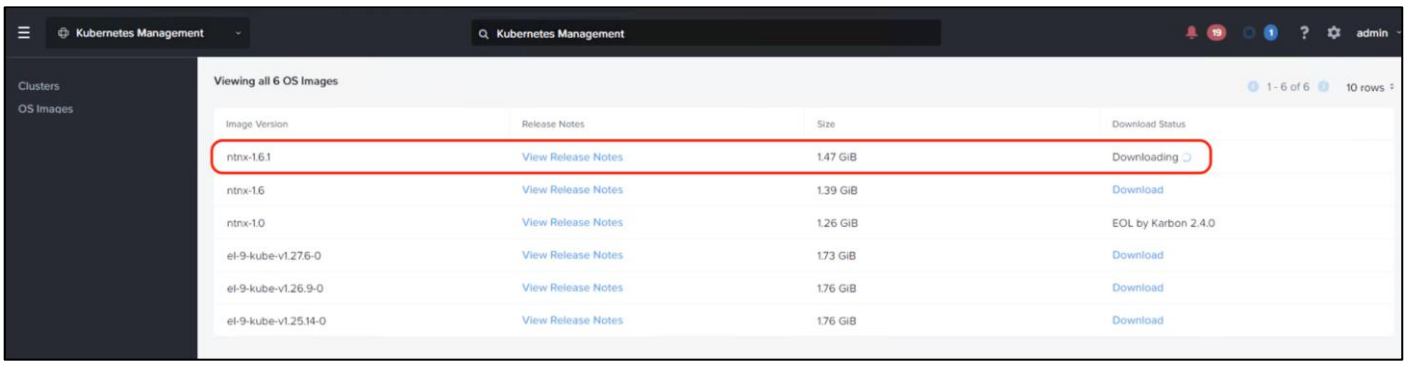
ステップ 1. Prism Central にログインし、マーケットプレイスで NKE を検索します。

ステップ 2. NKE 2.9 を有効にします。Kubernetes 管理を選択します。[OS イメージのダウンロード（Download OS Image）] をクリックします。

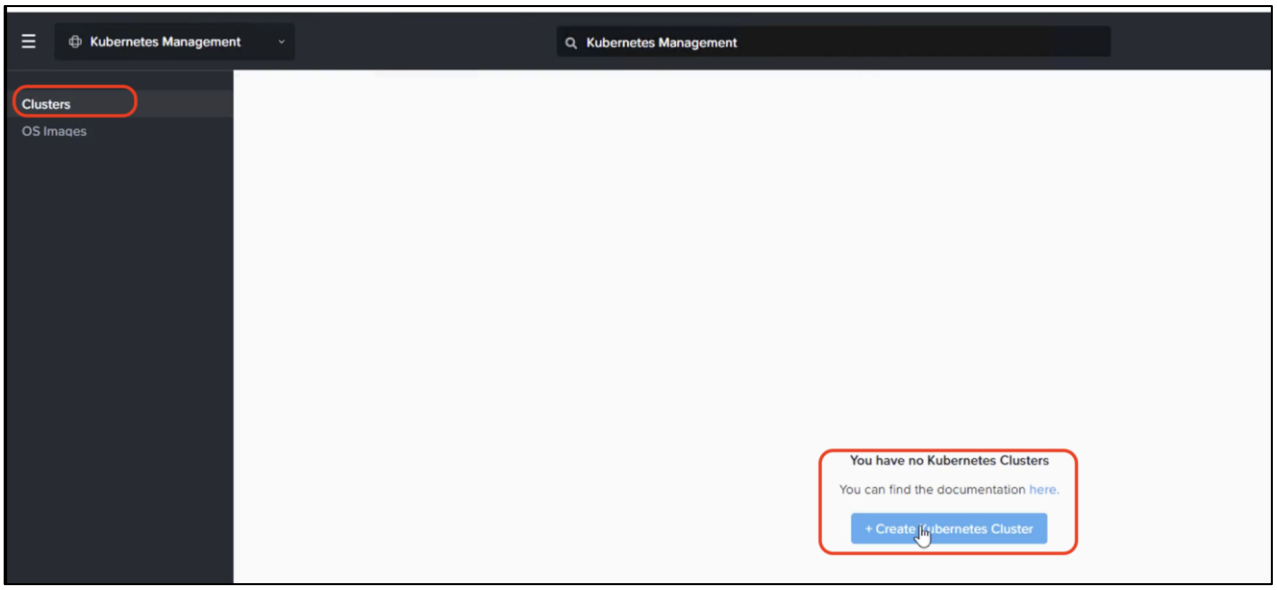


注： マーケットプレイスで NKE を有効にできないイベントは、Prism Central にログインし、クラスタを再起動します（Genesis の再起動）。

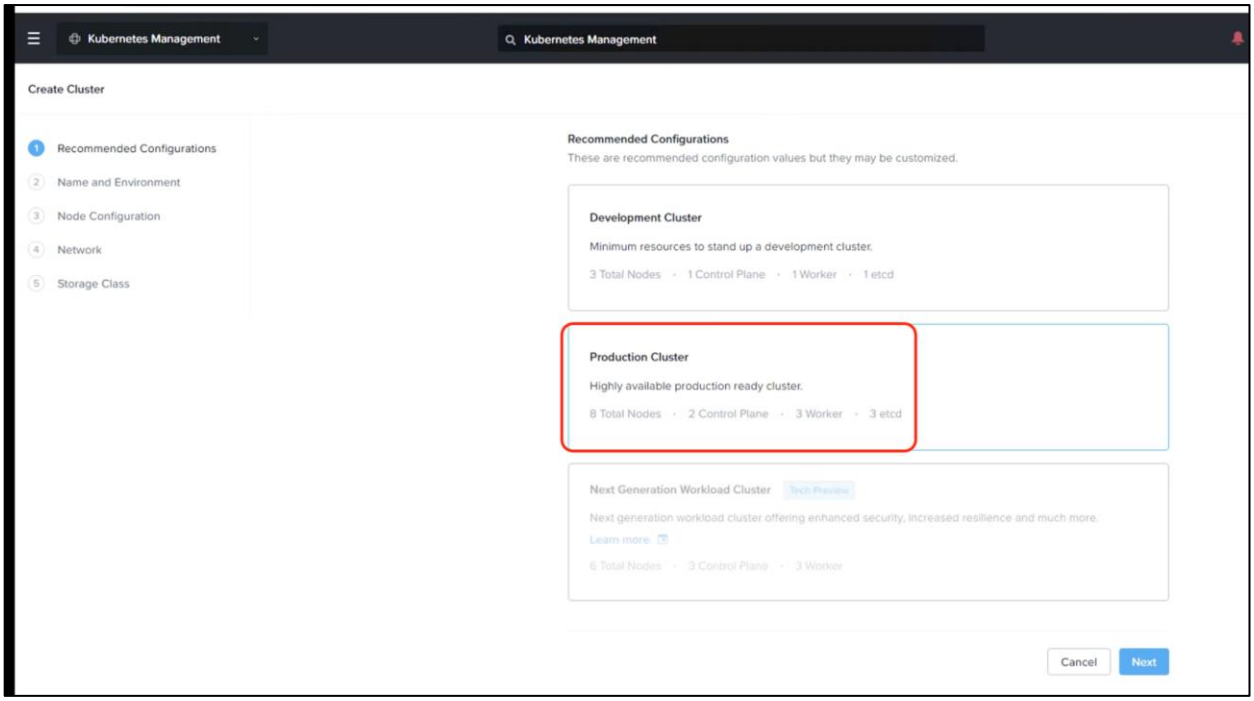
ステップ 3. Kubernetes クラスタのホスト OS イメージの「ntnx-1.6.1」を選択してダウンロードします。



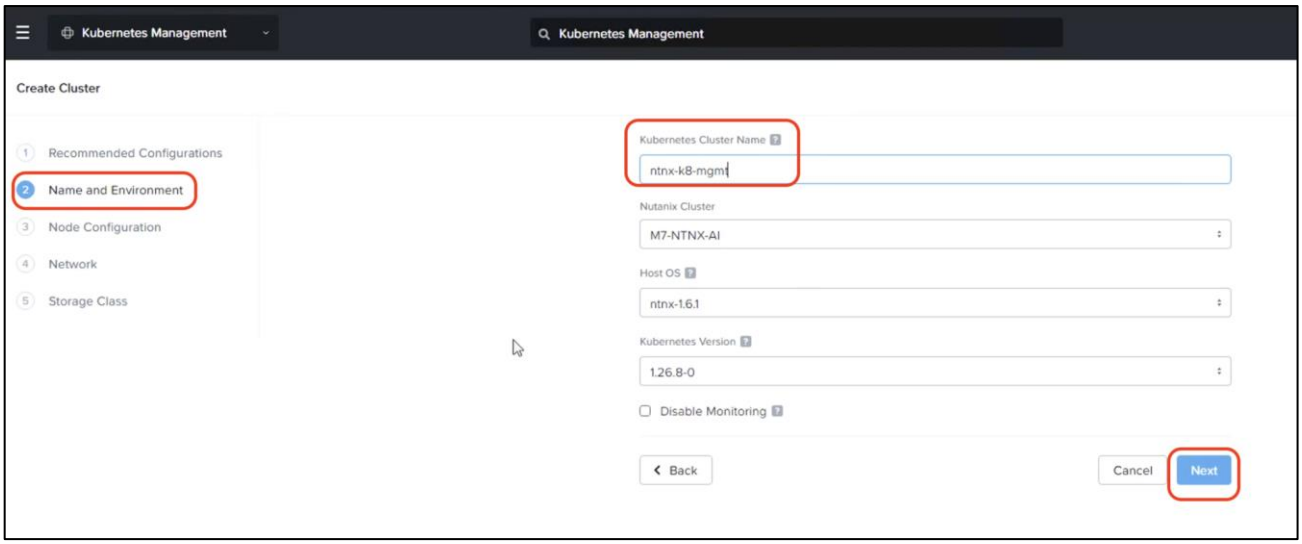
ステップ 4. イメージがダウンロードされたら、左側のナビゲーション バーからクラスタを選択し、[Kubernetes クラスタの作成（Create Kubernetes Cluster）] をクリックします。



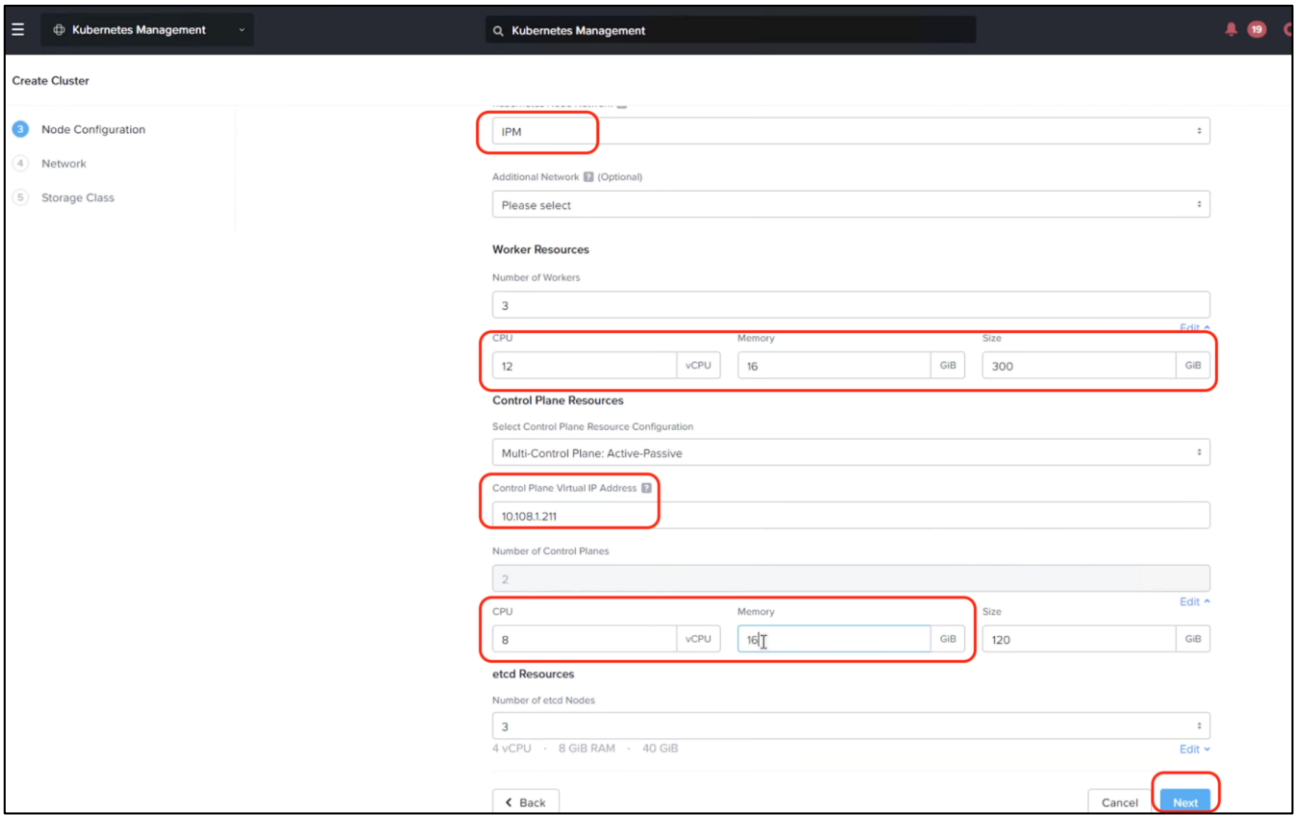
ステップ 5. 実稼働クラスタを選択し、[次へ（Next）] をクリックします。



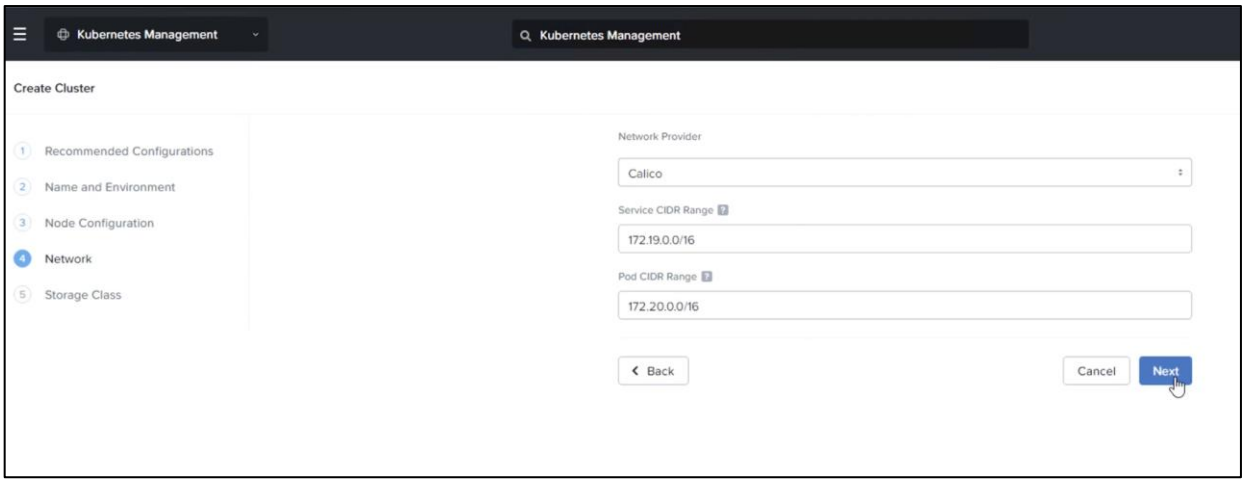
ステップ 6. 展開の名称に従ってクラスタに名前を付けます。このソリューションでは、クラスタの名前は「mtnx-k8-mgmt」と「ntnx-k8-workload」です。[次へ（Next）] をクリックします。



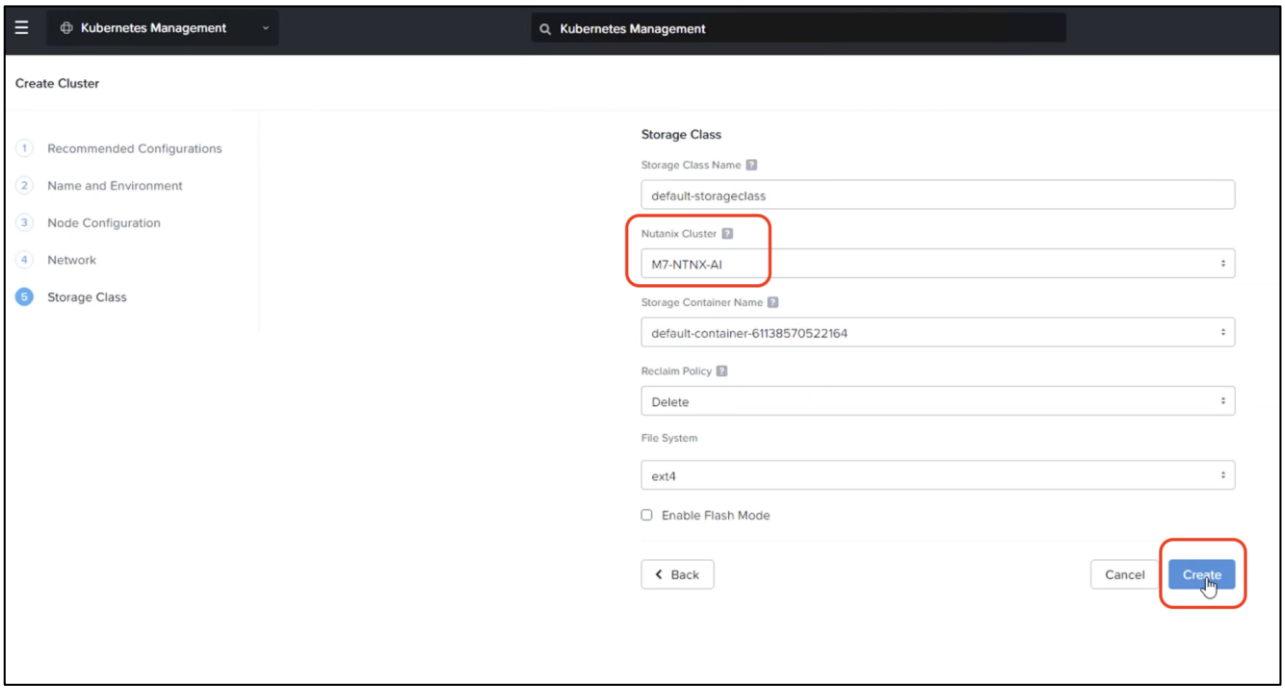
ステップ 7. 次の画面で、IPM の Kubernetes ノード ネットワーク（IPAM ネットワーク）を選択し、使用可能なコントロールプレーン VIP を入力し、ワーカー リソースを 12 vCPU、16 GBメモリ、300 GB ストレージ サイズに編集し、コントロールプレーン リソースを 8 vCPU と 16 GBメモリに編集します。[次へ（Next）] をクリックします。



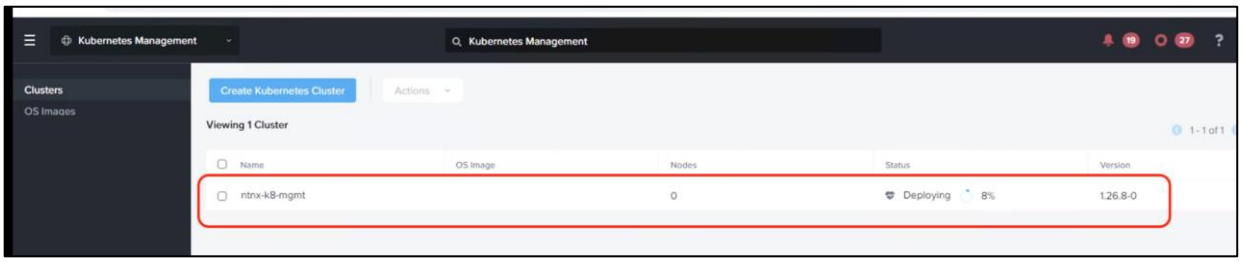
ステップ 8. [ネットワーク] で、デフォルトを選択します。



ステップ 9. [ストレージ クラス（Storage Class）] からデフォルトを選択し、Nutanix クラスタが GPT-in-a-Boxクラスタとして選択されていることを確認します。[Create] をクリックします。

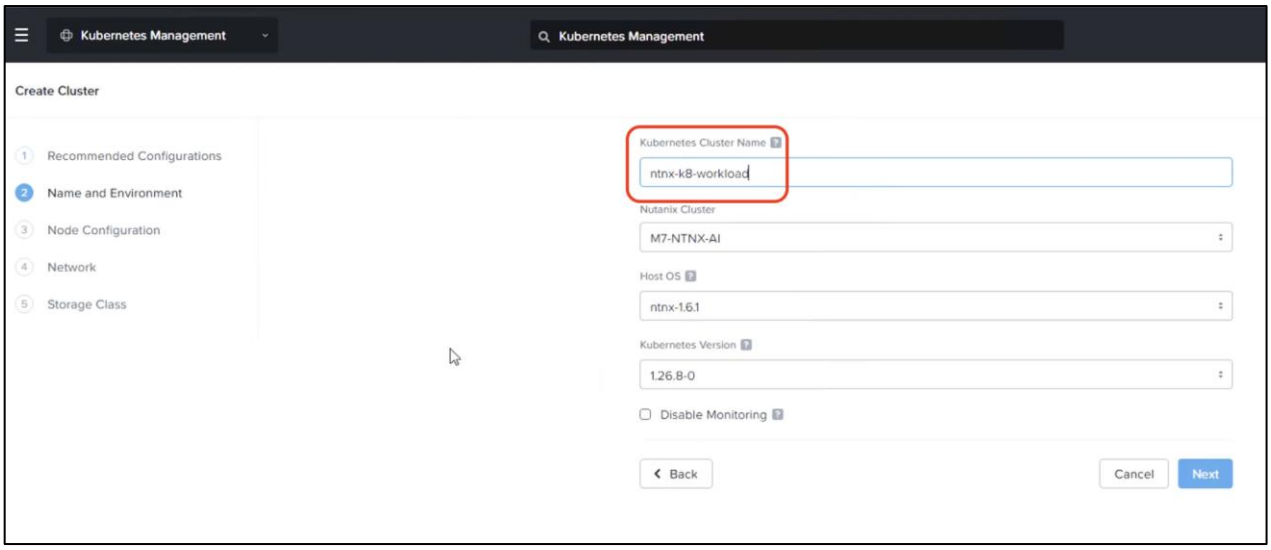


ステップ 10. ntnx-k8-mgmt クラスタの進行状況をモニタリングします。

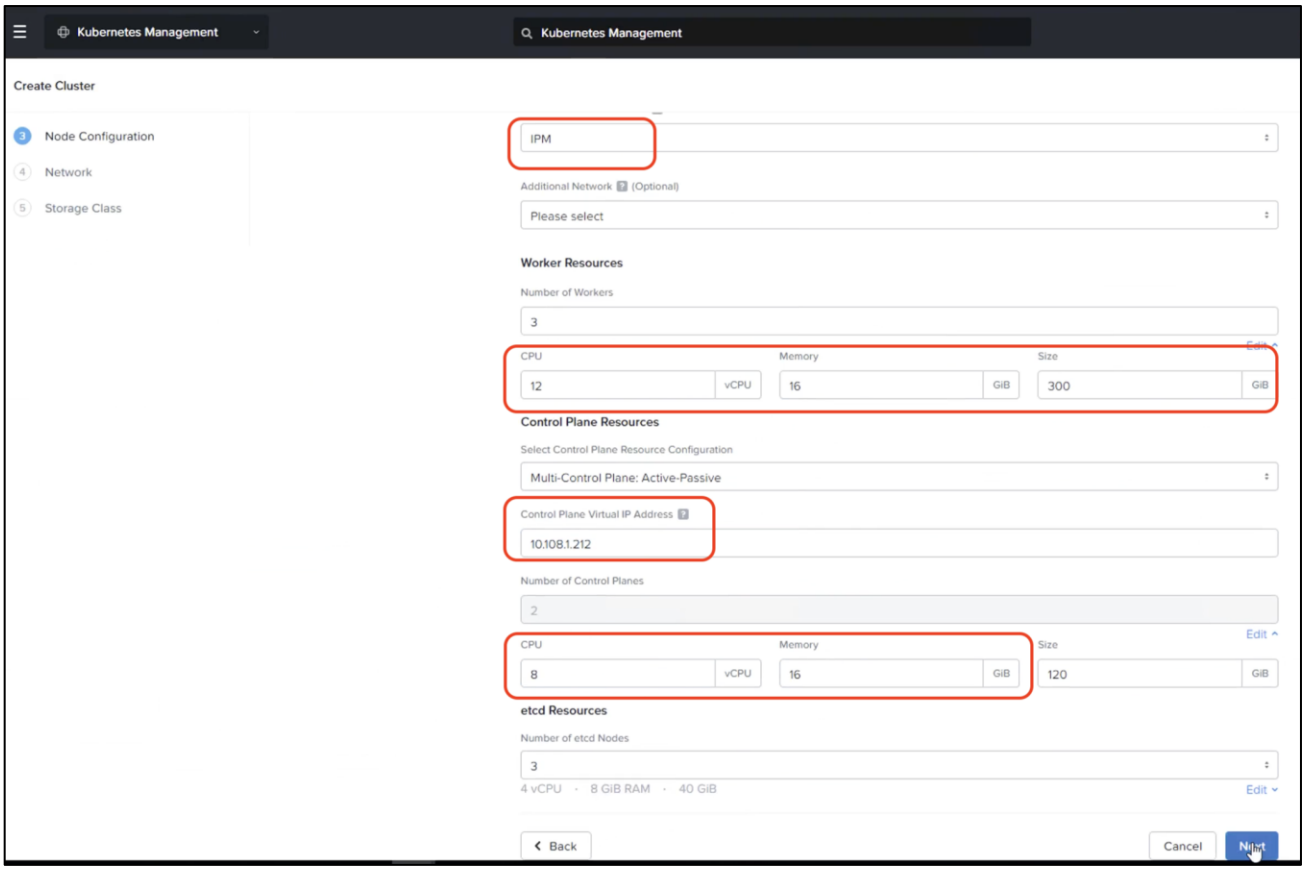


ステップ 11. 手順 1 ～ 10 を繰り返して、ワークロード クラスタをインストールします。

ステップ 12. 製品 Kubernetes クラスタを作成し、ntnx-k8-workload という名前を付けます。

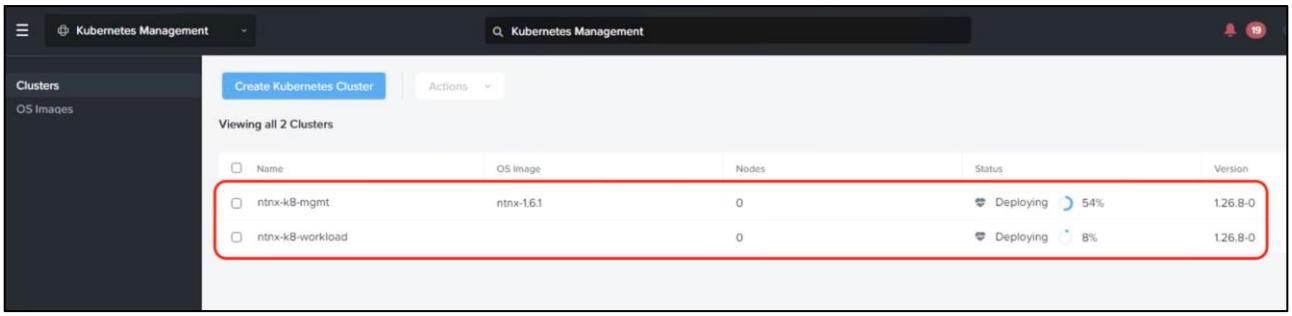


ステップ 13. 管理クラスタと同様に、ネットワーク IPM を選択し、使用可能なコントロールプレーン VIP を入力し、ワークロード クラスタのリソースを編集し、ワーカー リソースを 12 vCPU、16 GB メモリ、300 GB ストレージ サイズに、コントロール プレーン リソースを 8 vCPU と 16 GB メモリに設定します。[次へ（Next）] をクリックします。

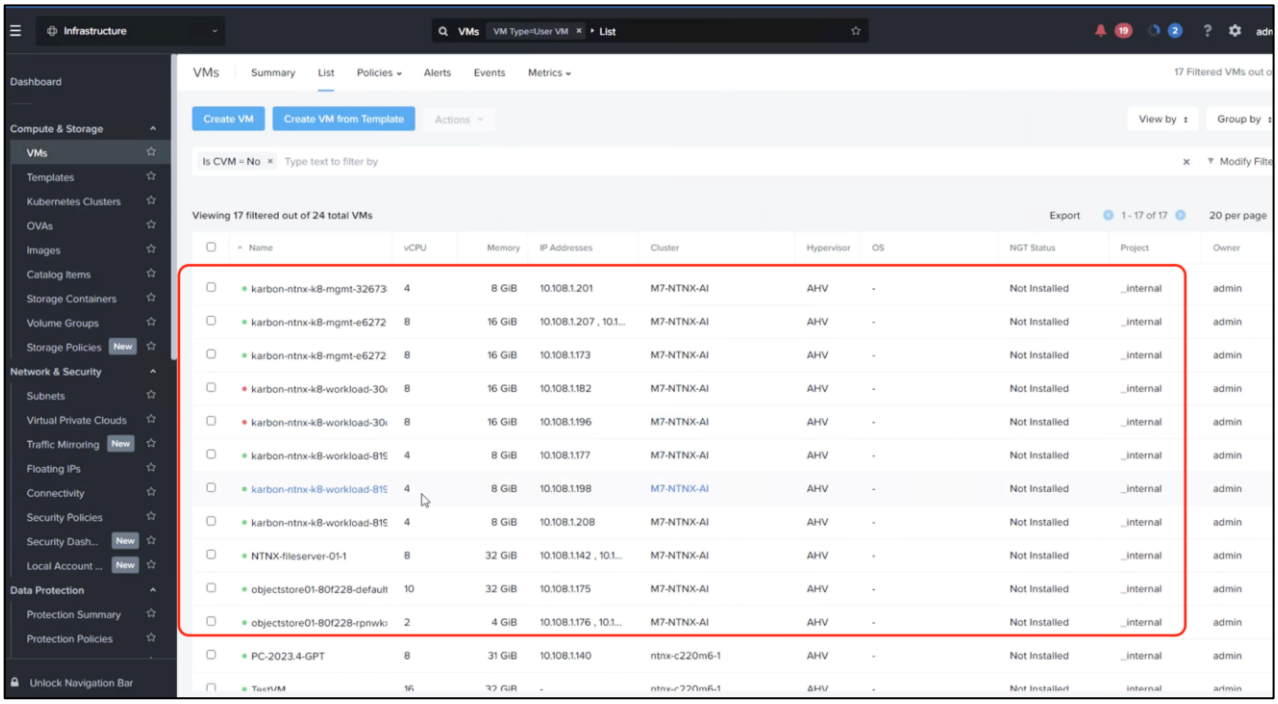


ステップ 14. [ネットワークおよびストレージ クラス（Network and Storage class）] 画面でデフォルトを選択します。[Create] をクリックします。

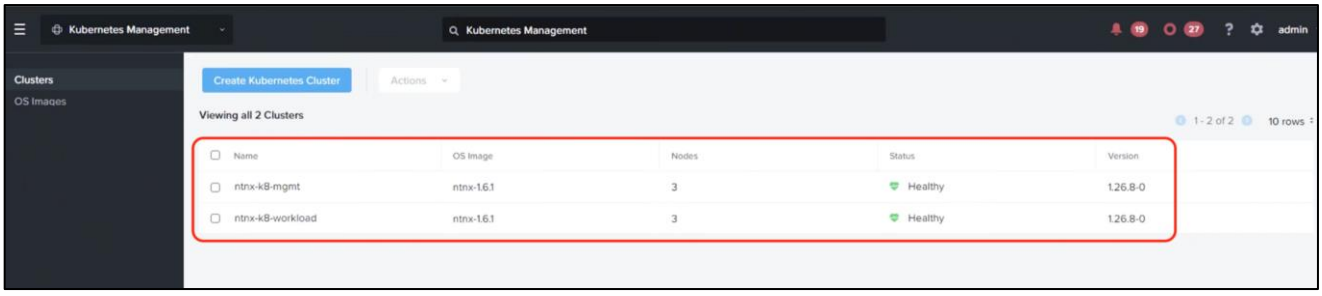
ステップ 15. ntnx-k8-mgmt と ntnx-k8-workload の両方の Kubernetes クラスタの進行状況をモニタリングします。



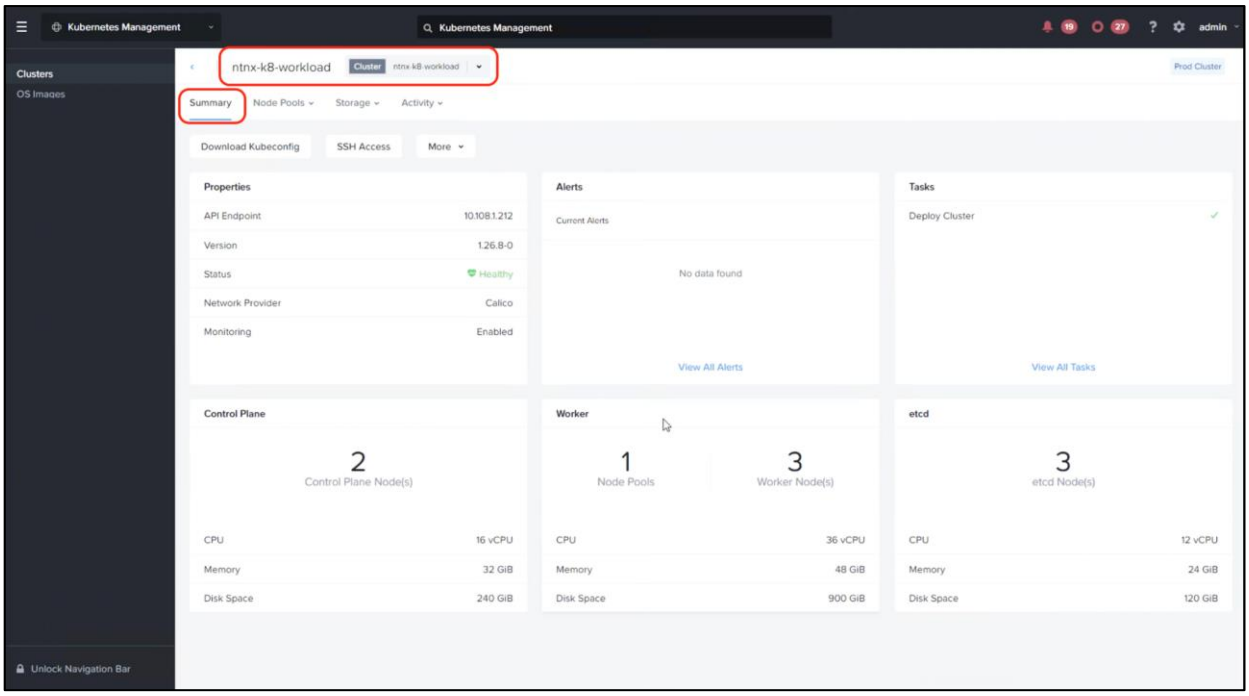
Kubernetes クラスタのすべてのノードは、次に示すように、GPT-in-a-Box Nutanix クラスタ上の VM として作成されます。



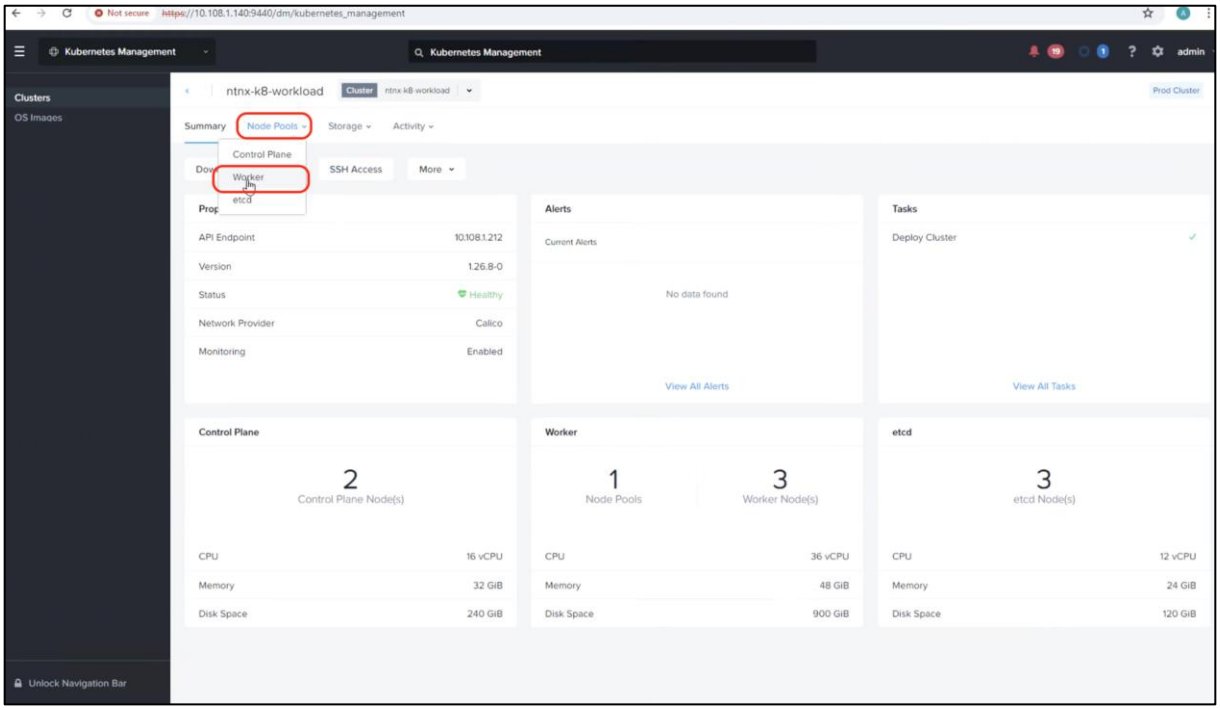
ステップ 16. 両方の Kubernetes クラスタが正常に作成され、正常性ステータスが正常であることを確認します。



ステップ 17. 次のいくつかの手順では、GPU ノードプールが Kubernetes ワークロード クラスタ（ntnx-k8-workload）に追加されます。[ntnx-k8- ワークロード クラスタ] をクリックします。

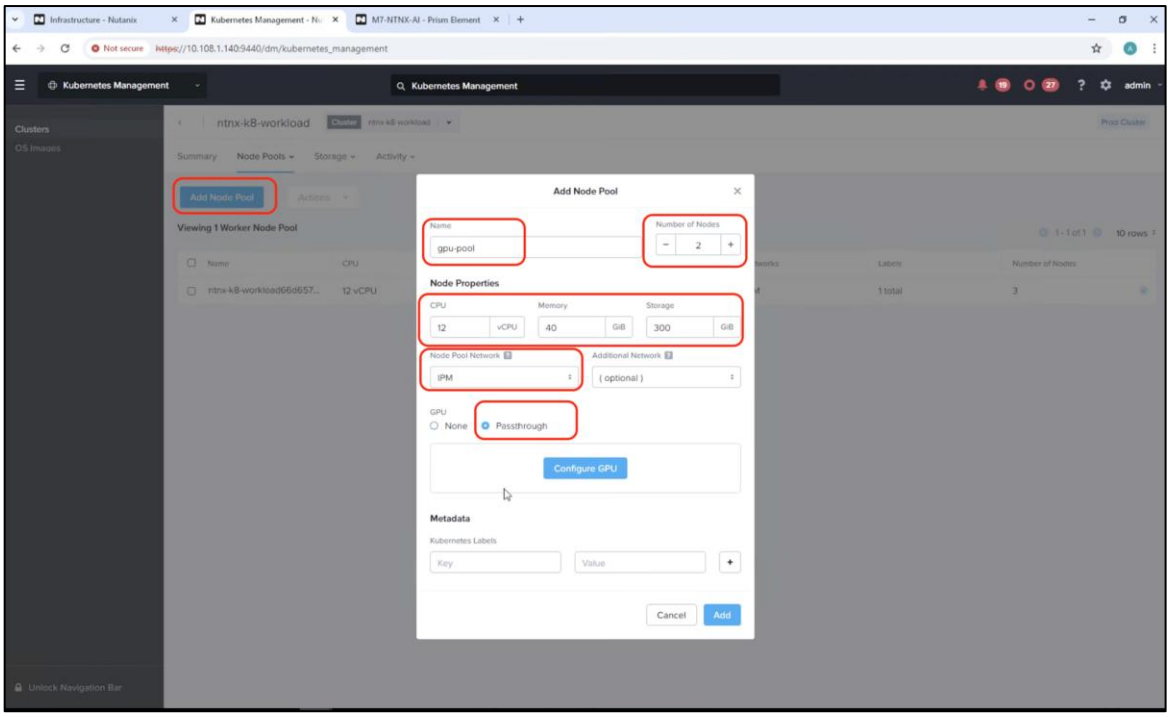


ステップ 18. [ノードプール（Node Pools）] > [ワーカー（Worker）] をクリックして、GPU ワーカーノードを追加します。

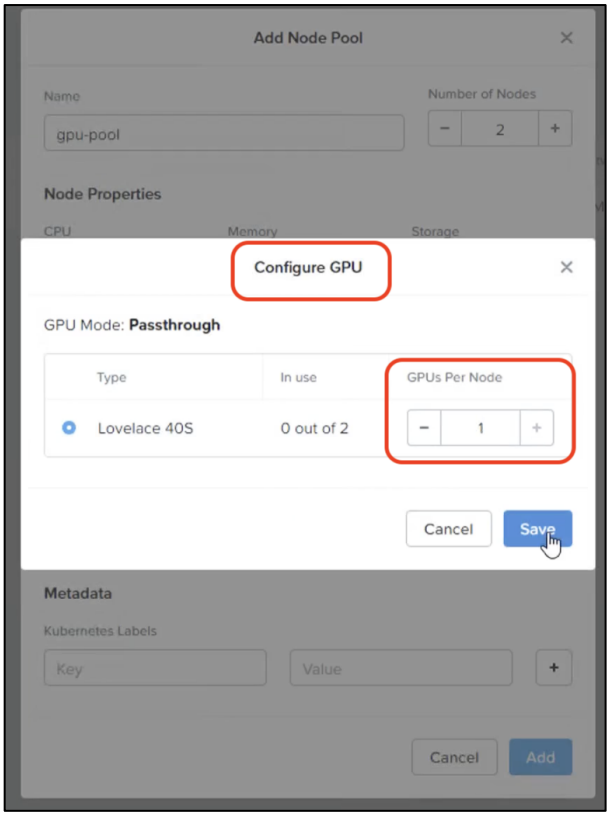


ステップ 19. ワーカーノード プールで、[ノード プールの追加（Add Node Pool）] をクリックし、ノード プール（gpu-pool）に名前を付け、ノード数（クラスタ全体の GPU の数に等しい）を選択します。vCPU = 12、メモリ = 40GB、ストレージ = 300GB、ノード プール ネットワーク：IPM（前のセクションで作成した IPAMアドレス）および GPU=パススルー（GPU=Passthrough）を選択します。

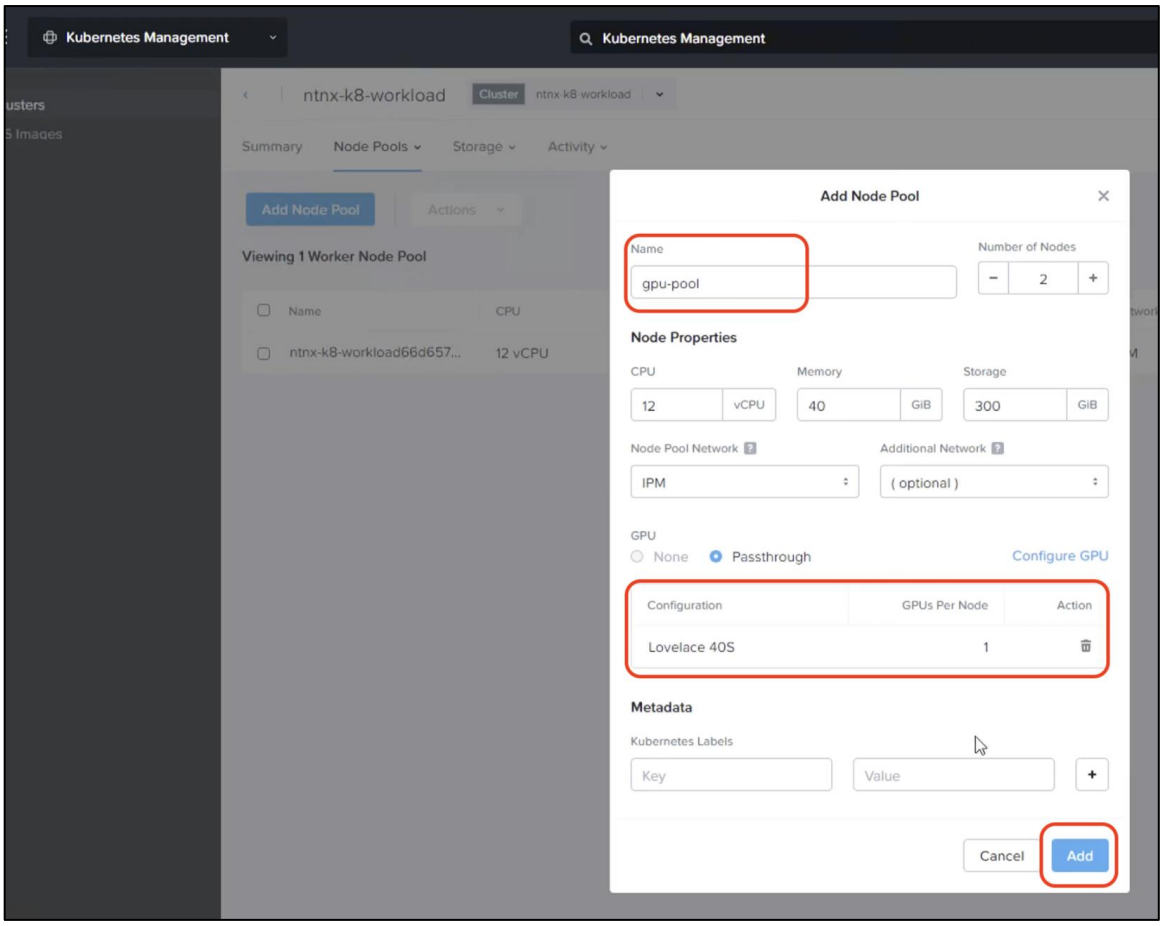
注： 既存の構成では、Nutanix クラスタを使用した 3 ノードCCHC の 1 つのノードにのみ 2 つの GPU があります。



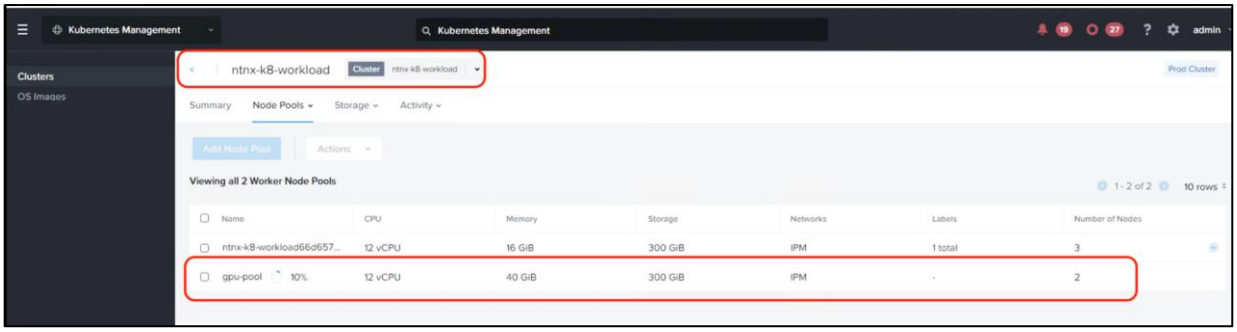
ステップ 20. [GPU の構成（Configure GPU）] をクリックし、ノードごとに 1GPU を構成します。[保存]をクリックします。



ステップ 21. [追加（Add）]をクリックします



ステップ 22. Kubernetes ワークロード クラスタへの GPU ノードプールの追加が完了するまで、進行状況をモニタリングします。



GitOps を使用した GPT-in-a-Box の展開

このセクションでは、基準アプリケーションとともに GPT-in-a-Box の設定に焦点を当てます。

リポジトリ構造

GPT-In-Box 参照アーキテクチャの上にアプリケーションを展開するには、 https://github.com/ucs-compute-solutions/nai-llm-fleet-infra のリポジトリを開始点として利用できます。

リポジトリは、GPT-In-Box の基準アーキテクチャを提供するように設計されています。複数のコンポーネントのカタログがあり、ユーザー アプリケーションを上に展開する柔軟性があり、ユーザーはアプリケーションのカスタム プロファイルを簡単に組み立てることができます。また、サンプル アプリケーションの実際の例も含まれています。

apps

appsディレクトリには、その下に nai-helm と gptnvd-reference-app の 2 つのサブディレクトリがあります。

アプリケーションが分割される方法は、Nai- **helm** が実際の LLM エンドポイントを展開し、 **gptnvd-reference-app** が基準アプリケーションを展開することです。（チャットボットとドキュメント読み込み機能）。

gptnvd-reference-app は、開発チームによって構築されたアプリケーションです。 nai-helm は、 人工知能運用チームによって微調整され、提供されます。

clusters/_profiles
プロファイルは、何をインストールするかを定義します。clusters/_profiles には複数のプロファイルがあります。

_base
すべてのクラスタプロファイルのベース（すべてのバリアントにインストールされているもの）です。

```
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$ tree -C clusters/_profiles/_base/
clusters/_profiles/_base/
├── cert-manager-repo.yaml
├── ingress-nginx-repo.yaml
├── kube-vip-repo.yaml
├── kubernetes-dashboard-repo.yaml
├── kustomization.yaml
├── kyverno-repo.yaml
└── weave-gitops-repo.yaml
```

llm-management
管理クラスタ プロファイルを定義します。管理固有のアプリケーションとプラットフォームのバリアントを定義します。

```
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$ tree -C clusters/_profiles/llm-management/
clusters/_profiles/llm-management/
├── _base
│   ├── elasticsearch-repo.yaml
│   ├── kafka-repo.yaml
│   ├── kustomization.yaml
│   ├── milvus-operator-repo.yaml
│   ├── milvus-repo.yaml
│   ├── otel-collector-daemon-repo.yaml
│   ├── otel-collector-deploy-repo.yaml
│   ├── otel-operator-repo.yaml
│   └── uptrace-repo.yaml
├── non-prod
│   ├── cert-manager-addons.yaml
│   ├── elasticsearch-addons.yaml
│   ├── ingress-nginx-addons.yaml
│   ├── kafka-addons.yaml
│   ├── kubernetes-dashboard-addons.yaml
│   ├── kustomization.yaml
│   ├── lets-encrypt-staging-issuer-patch.yaml
│   ├── milvus-addons.yaml
│   ├── milvus-operator-addons.yaml
│   ├── uptrace-addons.yaml
│   └── weave-gitops-addons.yaml
└── prod
    ├── cert-manager-addons.yaml
    ├── elasticsearch-addons.yaml
    ├── ingress-nginx-addons.yaml
    ├── kafka-addons.yaml
    ├── kubernetes-dashboard-addons.yaml
    ├── kustomization.yaml
    ├── milvus-addons.yaml
    ├── milvus-operator-addons.yaml
    ├── uptrace-addons.yaml
    └── weave-gitops-addons.yaml
```

llm-workloads
ワークロード クラスタ プロファイルを定義します。ワークロード固有のアプリケーションとプラットフォームのバリアントを定義します。

```
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$ tree -C clusters/_profiles/llm-workloads/
clusters/_profiles/llm-workloads/
├── _base
│   ├── gptnvd-reference-app-repo.yaml
│   ├── gpu-operator-repo.yaml
│   ├── knative-eventing-repo.yaml
│   ├── knative-istio-repo.yaml
│   ├── knative-serving-repo.yaml
│   ├── kserve-repo.yaml
│   ├── kuberay-cluster-repo.yaml
│   ├── kuberay-operator-repo.yaml
│   ├── kustomization.yaml
│   ├── nai-helm-repo.yaml
│   ├── otel-collector-daemon-repo.yaml
│   ├── otel-collector-deploy-repo.yaml
│   └── redis-repo.yaml
├── non-prod
│   ├── cert-manager-addons.yaml
│   ├── gpu-operator-addons.yaml
│   ├── ingress-nginx-addons.yaml
│   ├── jupyterhub-addons.yaml
│   ├── jupyterhub-repo.yaml
│   ├── kserve-addons.yaml
│   ├── kuberay-cluster-addons.yaml
│   ├── kubernetes-dashboard-addons.yaml
│   ├── kustomization.yaml
│   ├── nai-helm-addons.yaml
│   └── weave-gitops-addons.yaml
└── prod
    ├── cert-manager-addons.yaml
    ├── ingress-nginx-addons.yaml
    ├── kserve-addons.yaml
    ├── kuberay-cluster-addons.yaml
    ├── kubernetes-dashboard-addons.yaml
    ├── kustomization.yaml
    ├── nai-helm-addons.yaml
    └── weave-gitops-addons.yaml
```

カスタム プロファイルは、アプリケーションのニーズに基づいて作成できます。

.taskfiles

開発環境には Go Task バイナリが統合されています。Task バイナリは、GNU Make よりもシンプルで使いやすいタスク ランナー/ビルド ツールです。

`taskfile.yaml` には、`.taskfiles` ディレクトリに定義されているタスクのさまざまなカテゴリを含めるためのロジックがあります。

Flux ブートストラップ、NKEクラスタ管理、障害対応など、さまざまなヘルパー タスクがタスク ファイルで定義されます。

アプリケーションとプラットフォームのニーズに基づいて、追加のタスクを作成できます。

タスクの詳細については、 https://taskfile.dev/ を参照してください。

configs

configs は、ローカル スクリプトで使用される環境設定用のディレクトリです。

configs/_common/.env には、グローバル環境設定があります。

スクリプト

Scripts ディレクトリは、すべてのローカル スクリプトで構成されます。

platform

すべてのプラットフォーム サービスのカタログが含まれています。

環境プロファイルのカスタマイズ

このドキュメントでは、お客様が環境に展開する開始点として使用できる一般的な設計ブループリントを提供します。ソリューションは、要件とユースケースに基づいて、管理クラスタとワークロード クラスタの両方のプロファイルに必要な変更を簡単に導入できる柔軟性を提供します。

リポジトリには、ディレクトリ **clusters/_profiles**が含まれます。プロファイルには、クラスタにインスkiトールされるものの定義が含まれています。

この基準設計には、管理クラスタとワークロード クラスタ用の 2 つの環境プロファイル prod と non-prod が含まれています。**_base** は、すべてのクラスタプロファイルのベースです。

non-prod には開発、QA、ステージングなどの環境の定義があり、 **prod** には実稼働展開の定義があります。

この一般的な定義は、環境に合わせて必要な変更を導入することで、特定の環境やユースケースに合わせてカスタマイズできます。要件に基づいて、まったく新しいカスタム環境を構成することもできます。

カスタマイズの例を次に示します。

例 1

```
llm-workloads/
├── _base
│   ├── gptnvd-reference-app-repo.yaml
│   ├── gpu-operator-repo.yaml
│   ├── knative-eventing-repo.yaml
│   ├── knative-istio-repo.yaml
│   ├── knative-serving-repo.yaml
│   ├── kserve-repo.yaml
│   ├── kustomization.yaml
│   ├── nai-helm-repo.yaml
│   ├── otel-collector-daemon-repo.yaml
│   └── otel-collector-deploy-repo.yaml
├── non-prod
│   ├── cert-manager-addons.yaml
│   ├── gpu-operator-addons.yaml
│   ├── ingress-nginx-addons.yaml
│   ├── jupyterhub-addons.yaml
│   ├── jupyterhub-repo.yaml
│   ├── kserve-addons.yaml
│   ├── kustomization.yaml
│   ├── nai-helm-addons.yaml
│   └── weave-gitops-addons.yaml
└── prod
    ├── cert-manager-addons.yaml
    ├── gpu-operator-addons.yaml
    ├── ingress-nginx-addons.yaml
    ├── kserve-addons.yaml
    ├── kustomization.yaml
    ├── nai-helm-addons.yaml
    └── weave-gitops-addons.yaml
```

例 1 では、Jupyter は主に開発中に使用されるため、JupyterHub が「非実稼働」環境タイプで有効になっていることがわかります。要件に基づいて、jupyterhub-repo.yaml と jupyterhub-addons.yaml をコピーし、prod/kustomization.yaml を更新して Jupyter アドオンおよび JupyterHub リポジトリを含めるだけで、「prod」プロファイルで使用できるように構成することもできます。

```
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat non-prod/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1alpha1
kind: Component

components:
## load _profiles/_base first
- ../../_base
## then _profiles/llm-workloads/_base next
- ../_base

resources:
- jupyterhub-repo.yaml

patches:
- path: gpu-operator-addons.yaml
- path: cert-manager-addons.yaml
- path: ingress-nginx-addons.yaml
- path: weave-gitops-addons.yaml
- path: kserve-addons.yaml
- path: jupyterhub-addons.yaml
- path: nai-helm-addons.yaml
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat prod/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1alpha1
kind: Component

components:
## load _profiles/_base first
- ../../_base
## then _profiles/llm-workloads/_base next
- ../_base

patches:
- path: gpu-operator-addons.yaml
- path: cert-manager-addons.yaml
- path: ingress-nginx-addons.yaml
- path: weave-gitops-addons.yaml
- path: kserve-addons.yaml
- path: nai-helm-addons.yaml
```

例 2

例 2 では、この基準設計の実稼働環境と非実稼働環境の主な違いは、実稼働環境が Let's Encrypt と Amazon Route 53 を使用していることです。これをプライベート DNS および証明書管理と統合する場合は、このコンポーネントを無効化にして、カスタマイズしたコンポーネントを追加します。

```
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat non-prod/cert-manager-addons.yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: cert-manager-resource-configs
  namespace: flux-system
spec:
  components:
  - ../enable-prometheus
  # - ../enable-letsencrypt-aws-issuer
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat prod/cert-manager-addons.yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: cert-manager-resource-configs
  namespace: flux-system
spec:
  components:
  - ../enable-prometheus
  - ../enable-letsencrypt-aws-issuer

(base) PKOPPA-M-C2KH:llm-workloads pkoppa$
```

例 3

非実稼働環境では、GPU オペレータのタイム スライシングを有効にすると、複数のアプリケーションが GPU を共有できます。実稼働には推奨されないため、実稼働環境プロファイルには含まれません。

```
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat non-prod/gpu-operator-addons.yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: gpu-operator
  namespace: flux-system
spec:
  components:
  - ../repo-config
  - ../enable-time-slicing
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$ cat prod/gpu-operator-addons.yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: gpu-operator
  namespace: flux-system
spec:
  components:
  - ../repo-config
(base) PKOPPA-M-C2KH:llm-workloads pkoppa$
```

要件に基づいて、環境プロファイルでタイムスライスとスライス数を有効にできます。

リポジトリを分岐する

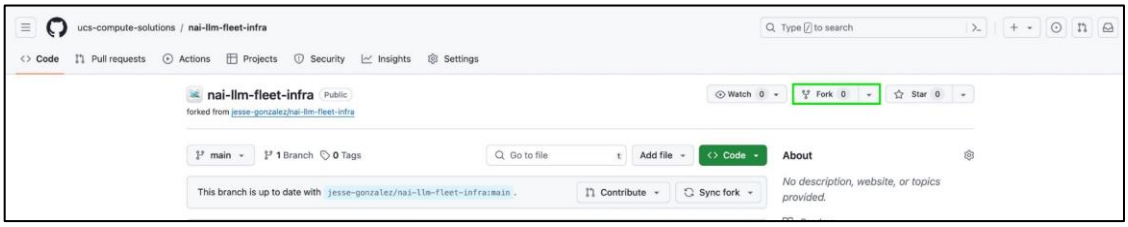
元のリポジトリから分岐する必要があります。分岐は、元の「アップストリーム」リポジトリとコードと可視性の設定を共有する新しいリポジトリです。これは、開発マシンの特定の環境に合わせてカスタマイズされます。

以下でGitHub の nai-llm-fleet-infra リポジトリを分岐します。 https://github.com/ucs-compute-solutions/nai-llm-fleet-infra

手順 1。 リポジトリを分岐する

ステップ 1. GitHub.com で、ucs-compute-solutions/nai-llm-fleet-infra リポジトリに移動します。

ステップ 2. このページの右上隅にある [分岐（Fork）] をクリックします



ステップ 3. [所有者（Owner）] のドロップダウンリストから、分岐したリポジトリの所有者を選択します。

ステップ 4. デフォルトでは、分岐にはアップストリーム リポジトリと同じ名前が付けられます。必要に応じて、分岐をさらに区別するために、[リポジトリ名（Repository name）] フィールドに名前を入力します。

ステップ 5. 任意に、[説明（Description）] フィールドに、分岐の説明を入力します。

ステップ 6. [分岐の作成（Create Fork）] をクリックします。

分岐したリポジトリのクローンを作成する

これで、リポジトリの分岐が作成されましたが、開発マシンのローカルのリポジトリにファイルがありません。

手順 1。 分岐したリポジトリのクローンを作成する

この手順では、関連する変更が行われて GitHub にプッシュされるように、リポジトリをクローンする方法について説明します。

ステップ 1. 開発マシン（Red Hat Enterprise Linux 8 ホストを推奨）で、 ギットがインストールされていることを確認します。

ステップ 2. Git から GitHub.com を使用して Git と認証を設定します。gh はコマンドラインのGitHubです。gh を使用して、GutHub で認証します。次のコマンドを実行し、手順に従います。

```
gh auth login
```

詳細については、 https://docs.github.com/en/get-started/getting-started-with-git/set-up-git を参照してください。

ステップ 3. GitHub.com から、リポジトリの分岐に移動します。

ステップ 4. <> Code をクリックします。

ステップ 5. リンクをコピーします。

ステップ 6. 現在の作業ディレクトリを、ディレクトリを複製する場所に変更します。

ステップ 7. ギット クローンと入力し、URL を貼り付けて、このリポジトリを分岐するされたコピーをクローンします。

```
ギット クローン git@github.com:your-username/ nai-llm-fleet-infra.git
```

ステップ 8. ディレクトリを複製されたリポジトリに変更します。

```
cd nai-llm-fleet-infra
```

開発環境の設定

アプリケーションとともに GPT-In-Box を展開する、管理、およびトラブルシュートするには、複数のツールとライブラリが必要です。これらの各ツールを手動でインストールし、開発ホストに特定のライブラリ セットをインストールするのは面倒です。また、展開されたアプリケーションに基づいて、追加のバイナリが必要になる場合があります。

したがって、最初に、 展開、管理、およびトラブルシュートに必要なすべてのツールを備えた開発環境が必要です。これには Devbox を活用します

Devbox は、開発用の分離シェルを簡単に作成できるコマンドラインツールです。（Pythonの venv と同様）。開発環境に必要なパッケージのリストを定義することから始めます。devbox はその定義を使用して、特定のツールを使用してアプリケーションのニーズに合わせて分離された環境を作成します。

devbox は、パッケージ管理とシステム構成に独自のアプローチを取るツールである Nix を内部的に使用しています。Nix 言語を学習しなくても、devbox を使用できます。

必要なすべてのパッケージは、サブシステムのように実行される Nix パッケージです。devbox は、管理するパッケージがオペレーティング システム レベルであることを除いて、yarn のようなパッケージ マネージャと同様に機能します。devbox では、 https://www.nixhub.io/ で入手可能な Nix Package Registry からパッケージバージョンをインストールできます。

手順 1。 開発環境の設定

ステップ 1. ルート ユーザーとして次のインストール スクリプトを実行して、Devbox の最新バージョンをインストールします。次のとおり、デフォルトを受け入れます。

```
curl -fsSL https://get.jetify.com/devbox | bash
```

```
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$ curl -fsSL https://get.jetpack.io/devbox | bash
Devbox 📦 by Jetify
  Instant, easy and predictable development environments.

  This script downloads and installs the latest devbox binary.

Confirm Installation Details
  Location:     /usr/local/bin/devbox
  Download URL: https://releases.jetify.com/devbox

? Install devbox to /usr/local/bin (requires sudo)? [Y/n] Y

Downloading and Installing
✓ Downloading devbox binary... [DONE]
→ Installing in /usr/local/bin/devbox (requires sudo)...
✓ Installing in /usr/local/bin/devbox... [DONE]
✓ Successfully installed devbox 🚀

Next Steps
  1. Learn how to use devbox
     Run devbox help or read the docs at https://github.com/jetify-com/devbox
  2. Get help and give feedback
     Join our community at https://discord.gg/jetify
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$
```

ステップ 2. パッケージにアクセスする devbox shell を起動します。

```
devbox shell
```

```
(base) PKOPPA-M-C2KH:nai-llm-fleet-infra pkoppa$ devbox shell
✓ Downloading version 0.12.0... [DONE]
✓ Verifying checksum... [DONE]
✓ Unpacking binary... [DONE]


Nix is not installed. Devbox will attempt to install it.

Press enter to continue or ctrl-c to exit.
```

ステップ 3. 初めて実行するときは、Nix は使用できません。インストールように求められます。また、devbox.json ファイルにリストされているすべてのツール/パッケージをインストールして構成します。

ステップ 4. 環境の追加パッケージについては、devbox shell を開始する前に devbox.json を更新します。

アプリケーションの展開

これまでに、ソリューション設計要件ごとに 2 つの Kubernetes クラスタが展開されています。

- 管理クラスタ：flux、kafka などの管理ワークロードをホストします。
- ワークロード クラスタ：開発 LLM および ChatBotアプリケーションをホストします。これは、Kubernetes ワーカーノードに渡される GPU を使用します。

手順 1。 管理クラスタのカスタム環境ファイルの作成

フレームワークは、クラスタの名前にキー入力されます。したがって、初期設定は、管理クラスタと NKE ワークロード クラスタに対して個別に実行されます。管理クラスタと各ワークロード クラスタに個別のシェル端末セッションを保持することをお勧めします。

ステップ 1. シェル端末で、K8S_CLUSTER_NAME 環境変数を NKE 管理クラスタの名前に設定します。

```
export K8S_CLUSTER_NAME = ntnx-k8-mgmt
```

ステップ 2. copy ./.env.sample.yaml to ./.env.${K8S_CLUSTER_NAME}.yaml:

```
cp ./.env.sample.yaml ./.env.${K8S_CLUSTER_NAME}.yaml
```

手順 2。 管理クラスタの環境ファイルのカスタマイズ

このセクションでは、管理クラスタの環境ファイル（この例では .env.ntnx-k8-mgmt.yaml）をカスタマイズする方法について説明します。

ステップ 1. クラスタ名を指定します。

```
## kubernetes cluster name
name: ntnx-k8-mgmt
```

ステップ 2. プロファイル名を設定します。このパラメータには、適切なプロファイルのディレクトリ名を設定します。管理の場合は、llm-management です。

```
## cluster_profile_type - anything under clusters/_profiles (e.g., llm-management, llm-workloads, etc.)
profile: llm-management
```

ステップ 3. 環境タイプを定義します。プロファイル内には、複数の環境タイプがあります。適切な環境タイプを指定します。ll-management には、実稼働環境と非実稼働環境があります。

```
## environment name - based on profile selected under clusters/_profiles/<profile>/<environment> (e.g., prod,
non-prod, etc.)
  environment: prod
```

ステップ 4. docker ハブのレジストリ アクセスの詳細を入力します。

```
## docker hub registry config
 registry:
    docker_hub :
      user: <<user_name>>
      password: <<access token>>
```

ステップ 5. フレーム ワークコンポーネントとアプリケーションは、GITOPS を経由して展開されます。そのため、 同期するには GIT Hub にアクセスする必要があります。作成された分岐にアクセスし、変更をプッシュするための GIT Hubアクセスの詳細を入力します。

```
flux:
  ## flux specific configs for github repo
  github:
    repo_url: <<repo url>>
    repo_user: <<user name>>
    repo_api_token: <<API Token>>
```

ステップ 6. Nutanix Prism Creds や Nutanix Objects Store Configs などの Nutanix 構成を提供します。

```
infra:
  ## Global nutanix configs
  nutanix:
    ## Nutanix Prism Creds, required to download NKE creds
    prism_central:
      enabled : true
      endpoint: <<Endpoint IP>>
      user: <<Configured User>>
      password: <<Password>>

    ## Nutanix Objects Store Configs
    objects:
      enabled : true
      host: <<Host_IP>>
      port: 80
      region: us-east-1
      use_ssl: false
      access_key: <<Access_Key>>
      secret_key: <<Secret_Key>>
```

ステップ 7. GPT in-a-Box は、RAG パイプラインとサーバーレス関数を活用します。ドキュメントのすべてのベクトル埋め込みを保存するには、ベクトルデータベースが必要です。この例では、Milvus はベクトル ストアとして構成されています。

ステップ 8. 埋め込みを保存するために作成されたバケット名を指定します。

```
## Milvus is vector database
  milvus:
    enabled : true
    version: 4.1.13
    milvus_bucket_name: milvus
```

ステップ 9. 次の手順では、サービスを構成します。 kube-vip は、nginx や istio などのロード バランサ IP アドレスを必要とするサービスに明示的に使用されます。 nginx は、ほとんどのフロントエンド ポータル/コンソールや、kafka、opentelemetry などの grpc バックエンドなど、Kubernetes イングレス オブジェクトを活用するすべてのものに明示的に使用されます。範囲内に少なくとも 2 つの IP を指定する必要があります。

```
kube_vip:
    enabled: true
    ## Used to configure default global IPAM pool. 範囲内に少なくとも 2 つの IP を指定する必要があります
    ## For Example: ipam_range: 172.20.0.22-172.20.0.23
    ipam_range: 10.108.1.214-10.108.1.216
```

ステップ 10. nginx_ingress の仮想 IP を指定します。

```
  ## required for all platform services that are leveraging nginx-ingress
  nginx_ingress:
    enabled : true
    version: 4.8.3
    ## Virtual IP Address (VIP) dedicated for nginx-ingress controller.
    ## This will be used to configure kube-vip IPAM pool to provide Services of Type: LoadBalancer
    ## Example: vip: 172.20.0.20
```

```
    vip: 10.108.1.213
```

ステップ 11. クラスタ内で作成されたすべてのデフォルト入力オブジェクトに使用される NGINX ワイルドカード入力サブドメイン。適切なクラスタ名でサブドメインを作成します。サブドメインで、* を使用してホスト レコードを作成します。wildcard_ingress_subdomain の値を指定します。

たとえば、DNS が *.ntnx-k8-mgmt.rtp4.local の場合、値は ntnx-k8-mgmt.rtp4.local です。

```
wildcard_ingress_subdomain: ntnx-k8-mgmt.rtp4.local
```

management_cluster_ingress_subdomain は、管理クラスタのワイルドカード入力サブドメインにマッピングされます。

```
management_cluster_ingress_subdomain: ntnx-k8-mgmt.rtp4.local
```

残りのコンポーネントは、Workload クラスタに必要なため、無効になります。

手順 3。 構成の生成と検証

ステップ 1. Taskfile Workstation コマンドを使用して、必要なワークステーション パッケージをインストールします。

```
task workstation:install-packages
```

ステップ 2. krew コマンドの経路を PATH にエクスポートします。

```
export PATH="${KREW_ROOT:-$HOME/.krew}/bin:$PATH"
```

ステップ 3. Taskfile BootStrap コマンドを実行して、クラスタ構成を検証および生成します。

```
task bootstrap:generate_cluster_configs
```

このタスクを実行すると、次のようになります。

- プライマリ プロジェクト ディレクトリ上の .env.<K8S_CLUSTER_NAME>.yaml ファイルから .local/<K8S_CLUSTER_NAME>/.env をステージングします
- tmpl/cluster/*.tmpl テンプレートからクラスター/<K8S_CLUSTER_NAME>/**.yaml 構成をステージングします

ステップ 4. 生成されたクラスタ構成を確認します。

```
cat .local/${K8S_CLUSTER_NAME}/.env
cat clusters/${K8S_CLUSTER_NAME}/platform/cluster-configs.yaml
```

手順 4。 変更を gitHub にプッシュします

ステップ 1. 次の git 操作を実行して、変更をプッシュします。

```
git add
git commit
git push
```

手順 5。 クラスタを選択します

ステップ 1. 次のコマンドを実行して NKEクラスタに接続します。

```
eval $(task nke:switch-shell-env) && \
task nke:download-creds
```

ステップ 2. 最初のタスクでは、ローカル シェルの既存のクラスタ インスタンスのプロンプトを求められます。

```
(devbox) [root@localhost nai-llm-fleet-infra]# eval $(task nke:switch-shell-env) && \
> task nke:download-creds && \
> kubectl get nodes
Select existing cluster instance to load from .local/ directory.
> ntnx-k8-mgmt
```

ステップ 3. このタスクは、選択したクラスタの NKE kubeconfig をダウンロードします。

```
(devbox) [root@localhost nai-llm-fleet-infra]# eval $(task nke:switch-shell-env) && task nke:download-creds
Verifying required tools are available: gum,krew,kubectl
Verifying required tools are available: gum,krew,kubectl

Logged successfully into ntnx-k8-mgmt cluster
export KUBECONFIG=/root/.kube/ntnx-k8-mgmt.cfg
Switched to context "ntnx-k8-mgmt-context".
```

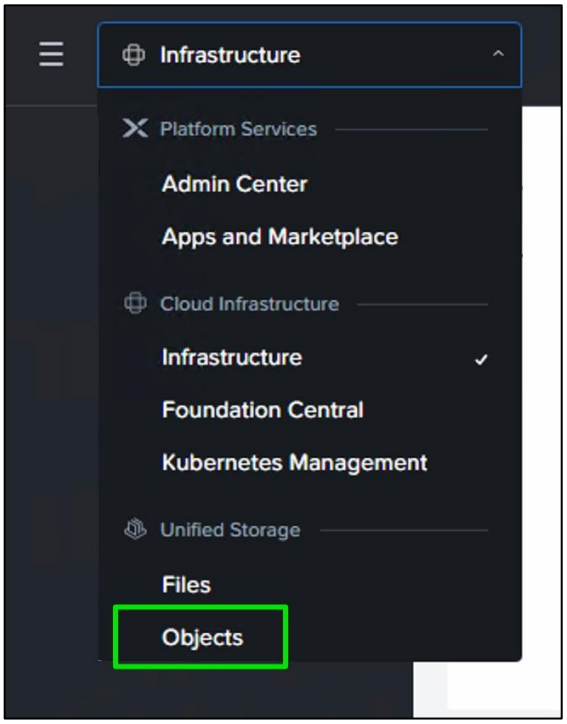
Milvus の Nutanix オブジェクト バケットの構成

バケットは、ベクトル埋め込みを保存するために Milvus が利用する論理的なコンテナです。バケットにユーザー アクセスに適切な権限が設定されていることを確認する必要があります。また、オブジェクト ストアは、Kafka 通告エンドポイントで有効にする必要があります。

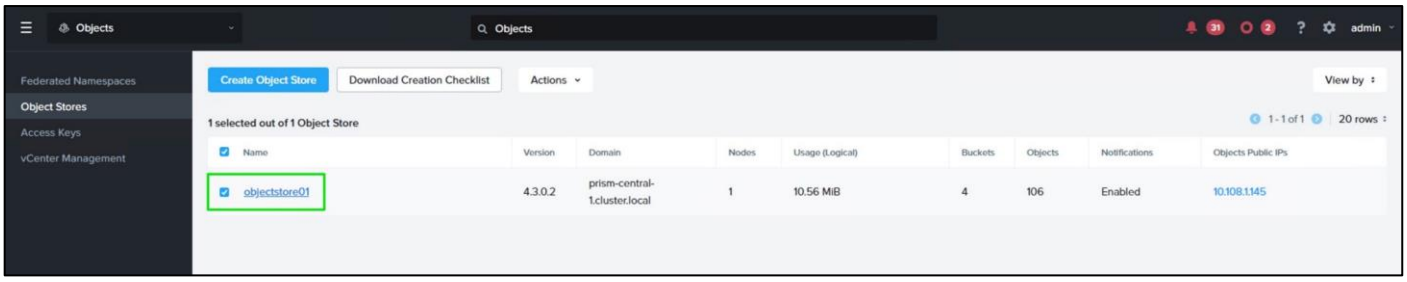
手順 1。 Milvus の Nutanix オブジェクト バケットの構成

ステップ 1. Prism Central Web コンソールにログインします。

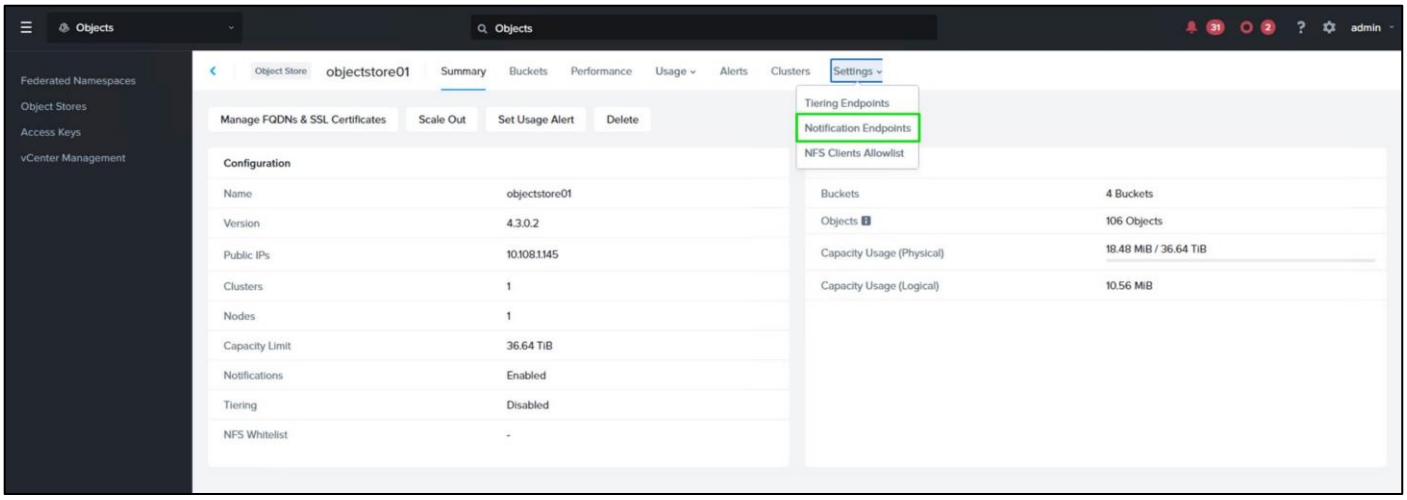
ステップ 2. アプリケーション スイッチャで、[オブジェクト（Objects）] をクリックします。



ステップ 3. [オブジェクト ストア（Object Stores）] テーブルで、ドキュメントと Milvus を保存するためにバケットが作成されるオブジェクト ストアの名前をクリックします。



ステップ 4. [設定（設定）] をクリックし、[通告エンドポイント（Notification Endpoints）] を選択します。

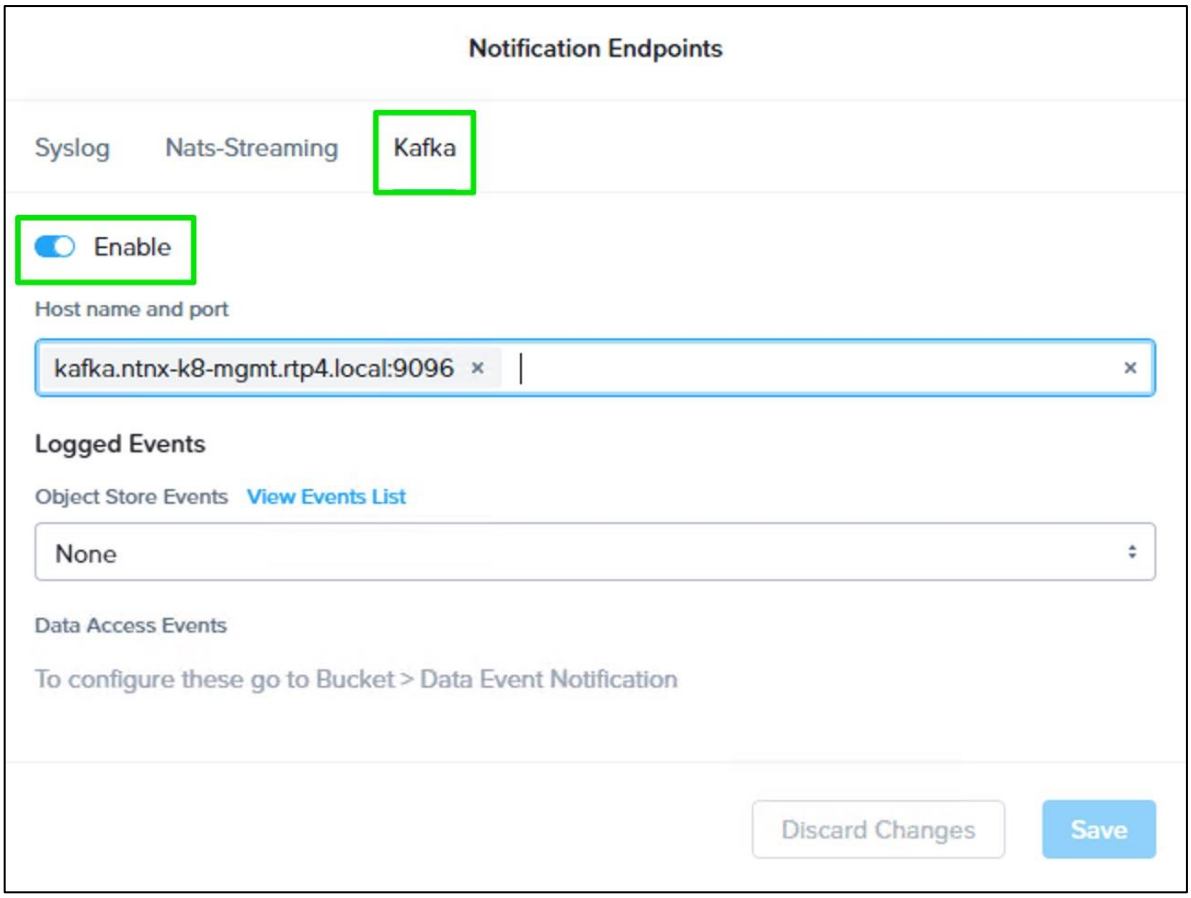


ステップ 5. [Kafka] を選択します。通告を有効にします。[オブジェクト ストア イベント（Object Store Events）] で、[なし（None）] を選択します。

注： Kafka は管理クラスタにあります。そのため、ホスト名には kafka と入力しました。および管理クラスタのワイルドカード入力サブドメインを入力します。

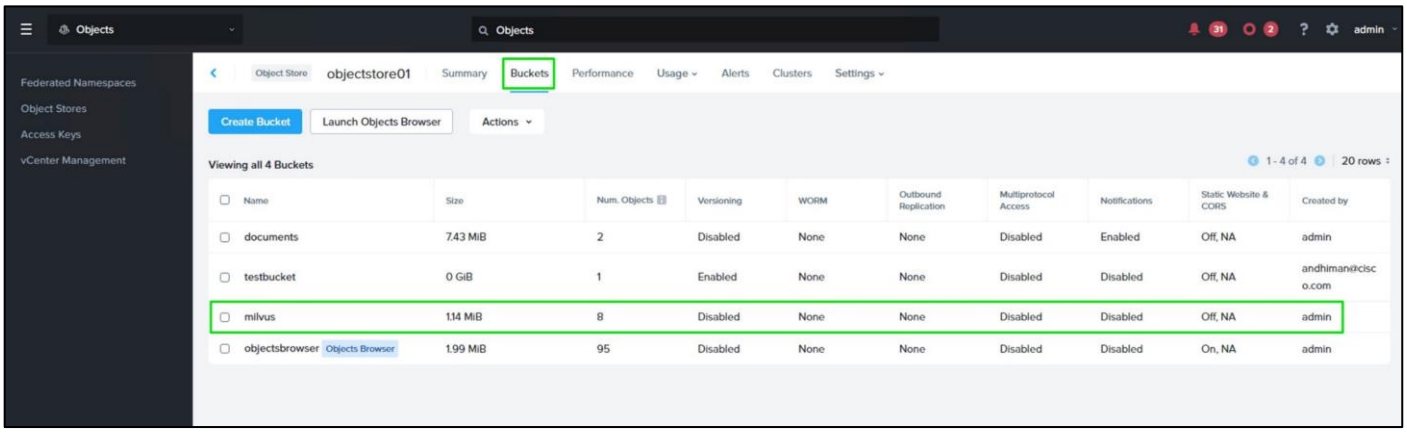
注： ポートに 9096 が指定されました。これは、入力の観点から公開する GRPC ポートです。

注： この例では、管理クラスタをサブドメインとして指定し、Kafka エンドポイントは kafka.ntnx-k8-mgmt.rtp4.local:9096 です。

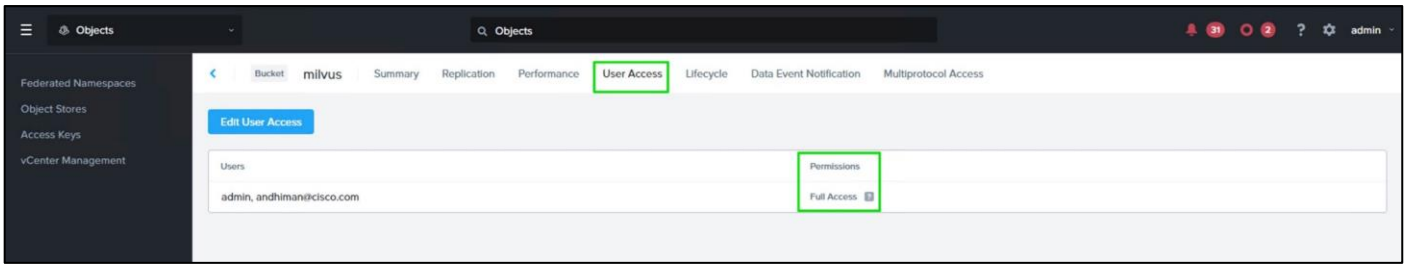


ステップ 6. [保存]をクリックします。

ステップ 7. [オブジェクト ストア（Object Stores）] テーブルで、オブジェクト ストアの名前をクリックし、Milvus 用に設定されたバケットをクリックします。

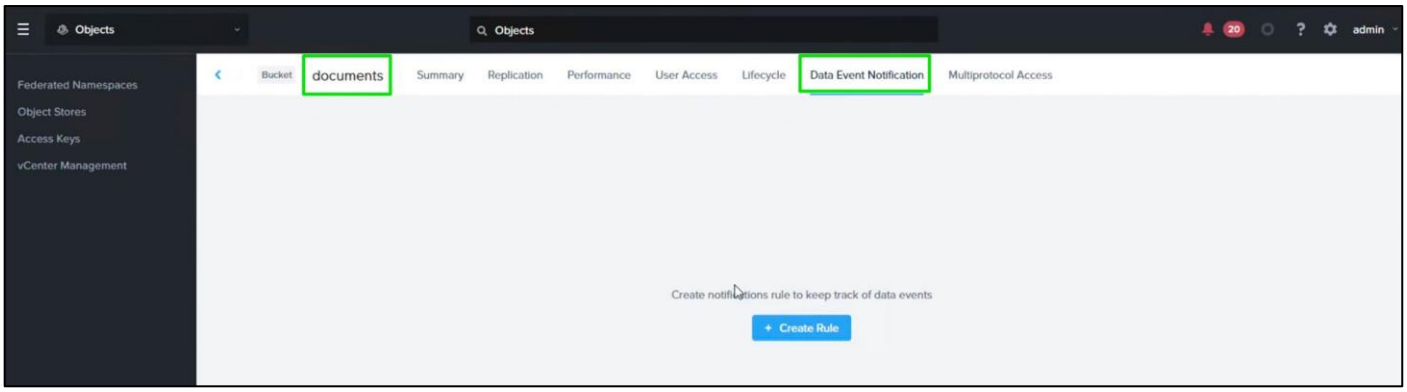


ステップ 8. [ユーザーアクセス（User Access）] に移動し、ユーザーに対してフルアクセスが構成されていることを確認します。



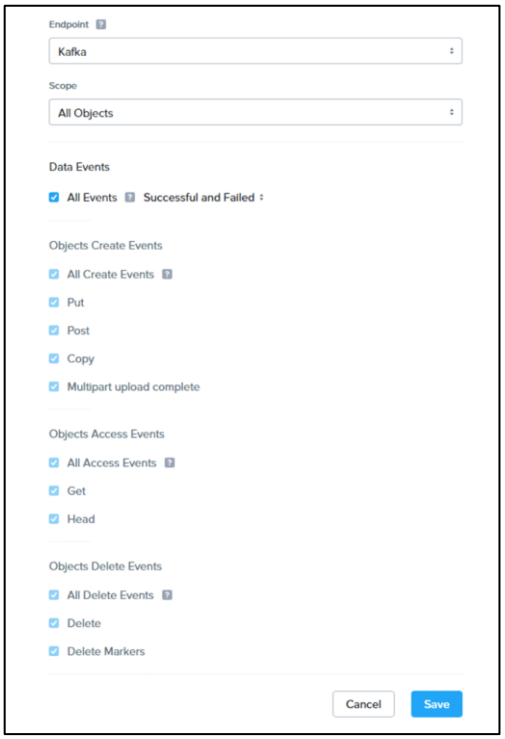
ステップ 9. RAG パイプラインの一部として取り込まれたドキュメントを保存するために作成されたバケットに対して、これを繰り返します。

ステップ 10. ドキュメントを保存するために作成されたバケットに移動します。

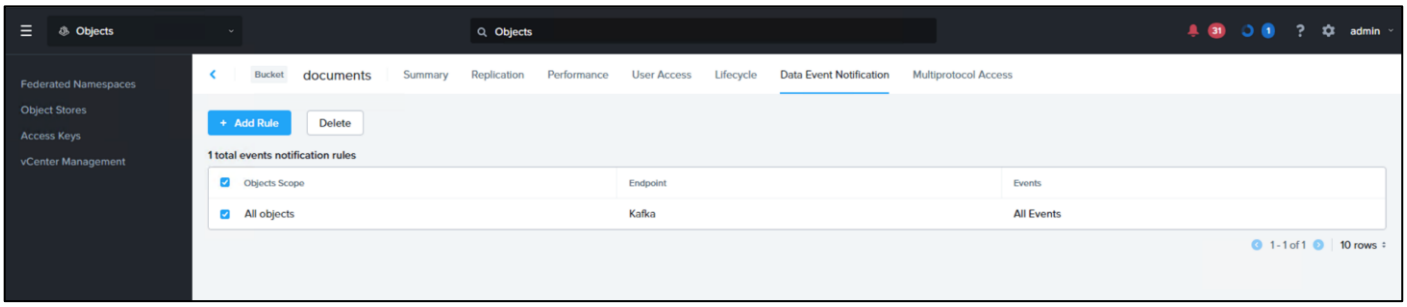


ステップ 11. [ルールの作成（Create Rule）] をクリックします。

ステップ 12. [エンドポイント（Endpoint）] で、[Kafka] を選択します。[範囲（Scope）] で、[すべてのオブジェクト（All Objects）] を選択し、[データ イベント（Data Events ）] の [すべてのイベント（All Events ）] をクリックします。



これにより、knative 内で実行され、ベクトル化のために Milvus をトリガーする取り込みサーバーレス機能を文書化するNotification（通告）イベントが送信されます。



手順 2。　管理クラスタのブートストラップ

クラスタ構成に問題がないことを確認したら、Flux をブートストラップします。

ステップ 1.　次のコマンドを実行して Flux をブートストラップします。

```
task bootstrap:silent
```

注：　問題がある場合は、**task:flux-collect**を使用してトラブルシューティングします。タスク bootstrap:silent は、必要な回数だけ再実行できます。

ステップ 2.　次のコマンドを実行して、新しい端末でのインストールのステータスをモニタリングします。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE 管理クラスタを選択)
task flux:watch
```

ステップ 3.　すべての helm チャートと Kustomization リソースが準備完了であることを確認します。

```
Every 1.0s: kubectl get gitrepo,helmrepo,hr,ks -n flux-system && echo && flux stats

NAME                                              URL                                                AGE   READY   STATUS
gitrepository.source.toolkit.fluxcd.io/flux-system   https://github.com/pkoppa/nai-llm-fleet-infra.git   68d   True    stored artifact for revision 'main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7'

NAME                                                URL                                                    AGE   READY   STATUS
helmrepository.source.toolkit.fluxcd.io/bitnami         https://charts.bitnami.com/bitnami                     68d   True    stored artifact: revision 'sha256:aca4869e15a8abbb527ebecd3e2fa1f54e47c5a497bc2980634ba7e57d05d538'
helmrepository.source.toolkit.fluxcd.io/cert-manager    https://charts.jetstack.io                             68d   True    stored artifact: revision 'sha256:47e64a16a5836278581fe25c168b26cd1dee5e2da6ae03d575f003b55c6cfeeb'
helmrepository.source.toolkit.fluxcd.io/ingress-nginx   https://kubernetes.github.io/ingress-nginx             68d   True    stored artifact: revision 'sha256:ff73d1ebca9f738c8d0688c7755fb50c7a201b631f6024ae2634bccd03afda70'
helmrepository.source.toolkit.fluxcd.io/kyverno         https://kyverno.github.io/kyverno/                     68d   True    stored artifact: revision 'sha256:a777d3ad986453bf6dde96e66288ac4945df8cd33e16a1ef828d0b1290855781'
helmrepository.source.toolkit.fluxcd.io/milvus          https://zilliztech.github.io/milvus-helm/              68d   True    stored artifact: revision 'sha256:7c3dd95be3b7fb1de47ac6bb0e2f56068e535dc4513a57e5185a648838ca8031'
helmrepository.source.toolkit.fluxcd.io/opentelemetry   https://open-telemetry.github.io/opentelemetry-helm-charts   68d   True    stored artifact: revision 'sha256:a2e2e993cbd77e5d9c381453a051b869e1ae50ccdd2171dd74999ebcd3b321f9'
helmrepository.source.toolkit.fluxcd.io/uptrace         https://charts.uptrace.dev                             68d   True    stored artifact: revision 'sha256:8bcab55257181403380a9bb7b3617a3cdf1712d29701b84f8fa9dff91bfa7fd1'
helmrepository.source.toolkit.fluxcd.io/weave-gitops    oci://ghcr.io/weaveworks/charts                        68d

NAME                                                    AGE   READY   STATUS
helmrelease.helm.toolkit.fluxcd.io/cert-manager         68d   True    Helm install succeeded for release cert-manager/cert-manager-cert-manager.v1 with chart cert-manager@v1.9.1
helmrelease.helm.toolkit.fluxcd.io/ingress-nginx        68d   True    Helm install succeeded for release ingress-nginx/ingress-nginx-ingress-nginx.v1 with chart ingress-nginx@4.8.3
helmrelease.helm.toolkit.fluxcd.io/kafka                68d   True    Helm install succeeded for release kafka/kafka-kafka.v1 with chart kafka@26.8.5
helmrelease.helm.toolkit.fluxcd.io/kyverno              68d   True    Helm install succeeded for release kyverno/kyverno-kyverno.v1 with chart kyverno@3.1.4
helmrelease.helm.toolkit.fluxcd.io/milvus-vectordb      68d   True    Helm install succeeded for release milvus/milvus-milvus-vectordb.v1 with chart milvus@4.1.13
helmrelease.helm.toolkit.fluxcd.io/opentelemetry-collector-daemon   68d   True    Helm install succeeded for release opentelemetry/opentelemetry-opentelemetry-collector-daemon.v1 with chart opentelemetry-collector@0.80.1
helmrelease.helm.toolkit.fluxcd.io/opentelemetry-collector-deployment   68d   True    Helm install succeeded for release opentelemetry/opentelemetry-opentelemetry-collector-deployment.v1 with chart opentelemetry-collector@0.80.1
helmrelease.helm.toolkit.fluxcd.io/opentelemetry-operator   68d   True    Helm install succeeded for release opentelemetry/opentelemetry-opentelemetry-operator.v1 with chart opentelemetry-operator@0.47.0
helmrelease.helm.toolkit.fluxcd.io/uptrace              68d   True    Helm upgrade succeeded for release uptrace/uptrace-uptrace.v2 with chart uptrace@1.5.7
helmrelease.helm.toolkit.fluxcd.io/weave-gitops         68d   True    Helm upgrade succeeded for release weave-gitops/weave-gitops-weave-gitops.v2444 with chart weave-gitops@4.0.36

NAME                                                          AGE   READY   STATUS
kustomization.kustomize.toolkit.fluxcd.io/cert-manager        68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/cert-manager-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/flux-system         68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/ingress-nginx       68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kafka               68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kube-vip            68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kube-vip-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kyverno             68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kyverno-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/milvus              68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry       68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-daemon   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-deployment   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/uptrace             68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/weave-gitops        68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/weave-gitops-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7

RECONCILERS             RUNNING FAILING SUSPENDED       STORAGE
GitRepository           1       0       0               1022.5 KiB
OCIRepository           0       0       0               -
HelmRepository          8       0       0               22.6 MiB
HelmChart               10      0       0               924.5 KiB
Bucket                  0       0       0               -
Kustomization           17      0       0               -
HelmRelease             10      0       0               -
Alert                   0       0       0               -
Provider                0       0       0               -
Receiver                0       0       0               -
ImageUpdateAutomation   0       0       0               -
ImagePolicy             0       0       0               -
ImageRepository         0       0       0               -
```

ステップ 4. 問題がある場合は、ローカルの git リポジトリを更新し、変更をプッシュして、次を実行します。

```
task flux:reconcile
```

手順 3。 ワークロード クラスタのカスタム環境ファイルの作成

ステップ 1. 新しい端末を開きます。ディレクトリを複製されたリポジトリに変更します。

```
cd nai-llm-fleet-infra
```

ステップ 2. devbox shell を起動します。

```
devbox shell
```

ステップ 3. シェル端末で、K8S_CLUSTER_NAME 環境変数を NKE ワークロードクラスタの名前に設定します。

```
export K8S_CLUSTER_NAME=ntnx-k8-workload
```

ステップ 4. Copy ./<Environment_file_of_NKE_Management_Cluster.yaml to ./.env.${K8S_CLUSTER_NAME}.yaml

```
cp ./.env.ntnx-k8-mgmt.yaml ./.env.${K8S_CLUSTER_NAME}.yaml
```

ステップ 5. 別のシェル端末で、すべての NKE Workload クラスタに対してステップ 1 ～ 4 を繰り返します。

手順 4。 ワークロード クラスタの環境ファイルのカスタマイズ

ステップ 1. クラスタ名を変更します。

```
## kubernetes cluster name
name: ntnx-k8-workload
```

ステップ 2. プロファイル名を設定します。プロファイルは、何をインストールするかを定義します。clusters/_profiles には複数のプロファイルがあります。

_base は、すべてのクラスタ プロファイルのベースです（すべてのバリアントにインストールされているもの）

llm-management には、管理クラスタ プロファイルの定義があります（管理固有のアプリケーションとプラットフォームのバリアントを定義します）。

llm-workloads に定義されたワークロード クラスタ プロファイル（ワークロード固有のアプリケーションとプラットフォームのバリアントを定義）

ステップ 3. このパラメータには、適切なプロファイルのディレクトリ名を設定します。ワークロードの場合は、llm-workloads です。

```
## cluster_profile_type - anything under clusters/_profiles (e.g., llm-management, llm-workloads, etc.)
profile: llm-workloads
```

ステップ 4. 環境タイプを定義します。プロファイル内には、複数の環境タイプがあります。適切な環境タイプを指定します。llm-workloads には、実稼働環境と非実稼働環境があります。

```
## environment name - based on profile selected under clusters/_profiles/<profile>/<environment> (e.g., prod,
non-prod, etc.)
  environment: non-prod
```

ステップ 5. docker ハブのレジストリ アクセスの詳細を入力します。

```
## docker hub registry config
 registry:
    docker_hub:
      user: <<user_name>>
      password: <<access token>>
```

ステップ 6. GPU は、ワークロード クラスタでのみ必要です。したがって、GPU オペレータを有効にします。非実稼働環境でのみタイム スライシングを有効にします。

```
## nvidia gpu specific configs
  gpu_operator:
    enabled: true
    version: v23.9.0
    cuda_toolkit_version: v1.14.3-centos7
    ## time slicing typically only configured on dev scenarios.
    ## ideal for jupyter notebooks
    time_slicing:
      enabled: false
      replica_count: 4
```

ステップ 7. フレーム ワークコンポーネントとアプリケーションは、GITOPS を経由して展開されます。同期するには GIT Hub にアクセスする必要があります。作成された分岐にアクセスし、変更をプッシュするための GIT Hubアクセスの詳細を入力します。

```
flux:
  ## flux specific configs for github repo
  github:
    repo_url: <<repo url>>
    repo_user: <<user name>>
    repo_api_token: <<API Token>>
```

ステップ 8. Nutanix Prism Creds や Nutanix Objects Store Configs などの Nutanix 構成を提供します。

```
infra:
  ## Global nutanix configs
  nutanix:
    ## Nutanix Prism Creds, required to download NKE creds
    prism_central:
      enabled : true
      endpoint: <<Endpoint IP>>
      user: <<Configured User>>
      password: <<Password>>

    ## Nutanix Objects Store Configs
    objects:
      enabled : true
      host: <<Host_IP>>
      port: 80
      region: us-east-1
      use_ssl: false
      access_key: <<Access_Key>>
      secret_key: <<Secret_Key>>
```

ステップ 9. GPT in a Box は、サーバーレス機能で RAG パイプラインを活用します。ドキュメントのすべてのベクトル埋め込みを保存するには、ベクトルデータベースが必要です。この例では、Milvus はベクトル ストアとして構成されています。

ステップ 10. 埋め込みを保存するために作成されたバケット名を指定します。

```
## Milvus is vector database
  milvus:
    enabled: true
    version: 4.1.13
    milvus_bucket_name: milvus
```

ステップ 11. サービスを構成します。

kube-vip は、nginx や istio などのロード バランサ IP アドレスを必要とするサービスに明示的に使用されます。nginx は、ほとんどのフロントエンド ポータル/コンソールや、kafka、opentelemetry などの grpc バックエンドなど、Kubernetes イングレス オブジェクトを活用するすべてのものに明示的に使用されます。範囲内に少なくとも2つのIPを指定する必要があります。

これは新しいクラスタであるため、別の IP アドレスのセットを指定する必要があります。

```
 サービスと共に使用できます。
  ##################################################
  ## Required variables for kube-vip and depedent services
  ## kube-vip specific configs required for any services needing to be configured with LoadBalancer Virtual
IP Addresses
```

```
  kube_vip:
    enabled: true
    ## Used to configure default global IPAM pool. 範囲内に少なくとも 2 つの IP を指定する必要があります
    ## For Example: ipam_range: 172.20.0.22-172.20.0.23
    ipam_range: 10.108.1.219-10.108.1.220
```

ステップ 12. nginx_ingress の仮想 IP を指定します。

```
  ## required for all platform services that are leveraging nginx-ingress
  nginx_ingress:
    enabled: true
    version: 4.8.3
    ## Virtual IP Address (VIP) dedicated for nginx-ingress controller.
    ## This will be used to configure kube-vip IPAM pool to provide Services of Type: LoadBalancer
    ## Example: vip: 172.20.0.20
    vip: 10.108.1.217
```

ステップ 13. NGINX ワイルドカード入力サブドメインは、クラスタ内で作成されるすべてのデフォルト入力オブジェクトに使用されます。サブドメインを作成し、クラスタ名を指定します。サブドメインで、* を含むホストレコードを作成します。wildcard_ingress_subdomain の値を指定します。

たとえば、DNS が *.ntnx-k8-workload.rtp4.local の場合、値は ntnx-k8-mgmt.rtp4.local になります。

```
wildcard_ingress_subdomain: ntnx-k8-workload.rtp4.local
```

ステップ 14. management_cluster_ingress_subdomain は、管理クラスタのワイルドカード入力サブドメインにマッピングされます。

```
management_cluster_ingress_subdomain: ntnx-k8-mgmt.rtp4.local
```

ステップ 15. istio イングレス ゲートウェイ専用の仮想 IP アドレス（VIP）を指定します。Istio は、LLM 推論エンドポイントで使用される knative-serving 専用です。

ステップ 16. Ingress Gateway の変更：Wildcard Subdomain を Workload NGINX Wildcard Ingress Subdomain の値に変更します。

```
istio:
    enabled：true
    version: 1.17.2
    ## Virtual IP Address (VIP) dedicated for istio ingress gateway.
    ## This will be used to configure kube-vip IPAM pool to provide Services of Type: LoadBalancer
    ## This address should be mapped to wildcard_ingress_subdomain defined below. 例：vip: 172.20.0.21
    vip: 10.108.1.218

    ## Istio Ingress Gateway - Wildcard Subdomain used for all knative/kserve llm inference endpoints.
    ## EXISTING A Host DNS Records are pre-requisites. 例： DNS が *.llm.example.com の場合、値は
llm.example.com です。
    ## If leveraging AWS Route 53 DNS with Let's Encrypt (below), make sure to enable/configure AWS
credentials needed to
    ## support CertificateSigningRequests using ACME DNS Challenges.
    ## For DEMO purposes, you can leverage the NIP.IO with the nginx_ingress vip and self-signed
certificates.
    ## For Example: llm.flux-kind-local.172.20.0.21.nip.io
    wildcard_ingress_subdomain: ntnx-k8-workload.rtp4.local
```

ステップ 17. kserve、knative_serving、および knative_istio を有効にします。

```
  kserve:
    enabled: true
    version: v0.11.2

  knative_serving:
    enabled: true
    version: knative-v1.10.1

  knative_istio:
    enabled: true
    version: knative-v1.10.0
```

ステップ 18. Knative Eventing は、Nutanix Objects ドキュメント バケットからイベント通告を受信するために使用されます。

```
knative_eventing:
    enabled: true
    version: knative-v1.10.1
```

ステップ 19. アプリケーションが分割される方法は、nai-helm が実際の LLM エンドポイントを展開するものであり、gptnvd_reference_app がチャットボットとドキュメント取り込み機能を展開するものです。

ステップ 20. 基準アプリケーションをインストールには、gptnvd_reference_app.enabled を設定し、gptnvd_reference_app.documents_bucket_name でドキュメント用に作成されたバケットを指定します。

```
apps:
  ## Required GPT NVD Reference Application Helm Chart Configs
  gptnvd_reference_app:
    enabled: true
    version: 0.2.7
    documents_bucket_name: documents
```

ステップ 21. nai_helm セクションで LLM エンドポイントを構成します。

```
## Required NAI LLM Helm Chart Configs
  nai_helm:
    enabled : true
    version: 0.1.1
    model: llama2_7b_chat
    revision: 94b07a6e30c3292b8265ed32ffdeccfdadf434a8
    maxTokens: 4000
    repPenalty: 1.2
    temperature: 0.2
    topP: 0.9
    useExistingNFS: false
    nfs_export: /llm-repo
    nfs_server: 10.108.1.141
    huggingFaceToken: <<Hugging Face User Access Token>>
```

モデル ストアのオプション

使用する大規模な言語モデルは、モデルを負荷するために TorchServe が使用するモデル アーカイブ ファイル フォーマットである必要があります。モデル アーカイブ ファイルを LLM に提供するには、2 つのオプションがあります。

1 つ目は、HuggingFace トークンを指定します。ポッドの初期化が行われている間、init コンテナは HuggingFace からモデルを直接ダウンロードし、モデル アーカイブ ファイルに変換します。

もう 1 つのオプションは、モデル アーカイブ ファイルを生成し、設定された NFS を保持することです。

この例では、LLM は HuggingFace からダウンロードされます。

Nutanix ファイルで NFS を利用してモデルを保存する場合は、useExistingNFS を true に設定し、huggingFaceToken を無効化にします。

モデル アーカイブ ファイルの生成の詳細については、https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/generating_mar/ を参照してください。

場合によっては、カスタム モデル（カスタム微調整モデルなど）を使用することもできます。この設計では、カスタムモデルを使用して MAR ファイルを生成し、推論サーバを起動する機能を提供します。

詳細については、https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/custom_model/ を参照してください。

構成の生成と検証

次のステップでは、Workload クラスタのクラスタ構成を検証して生成します。

手順 1。 クラスタ構成の生成と検証

ステップ 1. 次のコマンドを実行します。

```
task bootstrap:generate_cluster_configs
```

ステップ 2. 生成されたクラスタ構成を確認します。

```
cat .local/${K8S_CLUSTER_NAME}/.env
cat clusters/${K8S_CLUSTER_NAME}/platform/cluster-configs.yaml
```

手順 2。 変更を gitHub にプッシュします

ステップ 1. 次の git 操作を実行して、変更をプッシュします。

```
git add .
git commit
git push
```

手順 3。 クラスタを選択します

ステップ 1. NKE クラスタに接続するには、次のコマンドを使用します。

```
eval $(task nke:switch-shell-env) && \
task nke:download-creds
```

ステップ 2. 最初のタスクでは、ローカル シェルの既存のクラスタ インスタンスのプロンプトを求められます。

```
(devbox) [root@localhost nai-llm-fleet-infra]# eval $(task nke:switch-shell-env) && \
> task nke:download-creds
Select existing cluster instance to load from .local/ directory.
  ntnx-k8-mgmt
> ntnx-k8-workload
```

ステップ 3. 次のタスクでは、選択したクラスタの NKE kubeconfig がダウンロードされます。

```
(devbox) [root@localhost nai-llm-fleet-infra]# eval $(task nke:switch-shell-env) && \
> task nke:download-creds
Verifying required tools are available: gum,krew,kubectl
Verifying required tools are available: gum,krew,kubectl
Logged successfully into ntnx-k8-workload cluster
export KUBECONFIG=/root/.kube/ntnx-k8-workload.cfg
Switched to context "ntnx-k8-workload-context".
(devbox) [root@localhost nai-llm-fleet-infra]#
```

ワークロード NKE クラスタのブートストラップ

クラスタ構成に問題がないことを確認したら、Flux をブートストラップできます。

手順 1。 Bootstrap Flux

ステップ 1. 次のコマンドを実行して Flux をブートストラップします。

```
task bootstrap:silent
```

ステップ 2. 問題がある場合は、task:flux-collect を使用してトラブルシューティングします。タスク bootstrap:silent は、必要な回数だけ再実行できます。

ステップ 3. 次のコマンドを実行して、新しい端末でインストールのステータスをモニタリングします。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE ワークロード クラスタを選択)
task flux:watch
```

ステップ 4. すべての helm チャートと Kustomization リソースが READY 状態であることを確認します。

```
Every 1.0s: kubectl get gitrepo,helmrepo,hr,ks -n flux-system && echo && flux stats

NAME                                             URL                                                AGE   READY   STATUS
gitrepository.source.toolkit.fluxcd.io/flux-system   https://github.com/pkoppa/nai-llm-fleet-infra.git   68d   True    stored artifact for revision 'main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7'

NAME                                              URL                                                          AGE   READY   STATUS
helmrepository.source.toolkit.fluxcd.io/cert-manager    https://charts.jetstack.io                             68d   True    stored artifact: revision 'sha256:47e64a16a5836278581fe25c168b26cd1dee5e2da6ae03d575f003b55c6cfeeb'
helmrepository.source.toolkit.fluxcd.io/gpu-operator    https://nvidia.github.io/gpu-operator                  68d   True    stored artifact: revision 'sha256:93ac1bc9b8356ea61e7a85a20fd1e38fbb51f7746cd435db98da798628d1c679'
helmrepository.source.toolkit.fluxcd.io/ingress-nginx   https://kubernetes.github.io/ingress-nginx             68d   True    stored artifact: revision 'sha256:ff73d1ebca9f738c8d0688c7755fb50c7a201b631f6024ae2634bccd03afda70'
helmrepository.source.toolkit.fluxcd.io/kyverno         https://kyverno.github.io/kyverno/                     68d   True    stored artifact: revision 'sha256:a777d3ad986453bf6dde96e66288ac4945df8cd33e16a1ef828d0b1290855781'
helmrepository.source.toolkit.fluxcd.io/nai-llm-helm    https://jesse-gonzalez.github.io/nai-llm-helm/         50d   True    stored artifact: revision 'sha256:1467e89b34762a91290e332da8d5cb22593742658ea6b0848a20fded25f951cc'
helmrepository.source.toolkit.fluxcd.io/opentelemetry   https://open-telemetry.github.io/opentelemetry-helm-charts   68d   True    stored artifact: revision 'sha256:a2e2e993cbd77e5d9c381453a051b869e1ae50ccdd2171dd74999ebcd3b321f9'
helmrepository.source.toolkit.fluxcd.io/weave-gitops    oci://ghcr.io/weaveworks/charts                        68d

NAME                                                    AGE   READY   STATUS
helmrelease.helm.toolkit.fluxcd.io/cert-manager         68d   True    Helm install succeeded for release cert-manager/cert-manager-cert-manager.v1 with chart cert-manager@v1.9.1
helmrelease.helm.toolkit.fluxcd.io/gptnvd-ref-app       50d   True    Helm install succeeded for release gptnvd-reference-app/gptnvd-reference-app-gptnvd-ref-app.v1 with chart gptnvd-referenceapp@0.2.7
helmrelease.helm.toolkit.fluxcd.io/gpu-operator         68d   True    Helm upgrade succeeded for release gpu-operator/gpu-operator-gpu-operator.v2 with chart gpu-operator@v23.9.0
helmrelease.helm.toolkit.fluxcd.io/ingress-nginx        68d   True    Helm install succeeded for release ingress-nginx/ingress-nginx-ingress-nginx.v1 with chart ingress-nginx@4.8.3
helmrelease.helm.toolkit.fluxcd.io/kyverno              68d   True    Helm install succeeded for release kyverno/kyverno-kyverno.v1 with chart kyverno@3.1.4
helmrelease.helm.toolkit.fluxcd.io/llm                  68d   True    Helm upgrade succeeded for release llm/llm-llm.v4 with chart nai-llm@0.1.3
helmrelease.helm.toolkit.fluxcd.io/opentelemetry-collector-daemon   68d   True    Helm install succeeded for release opentelemetry/opentelemetry-opentelemetry-collector-daemon.v1 with chart opentelemetry-collector@0.80.1
helmrelease.helm.toolkit.fluxcd.io/opentelemetry-collector-deployment   68d   True    Helm install succeeded for release opentelemetry/opentelemetry-opentelemetry-collector-deployment.v1 with chart opentelemetry-collector@0.80.1
helmrelease.helm.toolkit.fluxcd.io/weave-gitops         68d   True    Helm upgrade succeeded for release weave-gitops/weave-gitops-weave-gitops.v1684 with chart weave-gitops@4.0.36

NAME                                                        AGE   READY   STATUS
kustomization.kustomize.toolkit.fluxcd.io/cert-manager      68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/cert-manager-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/flux-system       68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/gptnvd-reference-app   50d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/gpu-operator      68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/ingress-nginx     68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-deploy   50d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-post-deploy   50d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-pre-deploy   50d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-istio-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-istio-pre-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-post-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-pre-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kserve-deploy     68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kserve-post-deploy   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kube-vip          68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kube-vip-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kyverno           68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/kyverno-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/nai-helm          68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-daemon   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-deployment   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/weave-gitops      68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7
kustomization.kustomize.toolkit.fluxcd.io/weave-gitops-resource-configs   68d   True    Applied revision: main@sha1:d75197e219ce85a43d6a3cc282d815e0822c40a7

RECONCILERS             RUNNING FAILING SUSPENDED       STORAGE
GitRepository           1       0       0               1022.5 KiB
OCIRepository           0       0       0               -
HelmRepository          7       0       0               1.6 MiB
HelmChart               9       0       0               521.4 KiB
Bucket                  0       0       0               -
Kustomization           0       0       25              -
HelmRelease             0       0       9               -
Alert                   0       0       0               -
Provider                0       0       0               -
Receiver                0       0       0               -
ImageUpdateAutomation   0       0       0               -
ImagePolicy             0       0       0               -
ImageRepository         0       0       0               -
```

ステップ 5. 大規模な言語モデルのロードが完了するまで 15 ー 20 分待ちます。

ステップ 6. 問題がある場合は、ローカル git リポジトリを更新し、変更をGitHubにプッシュして、次のコマンドを実行します。

```
task flux:reconcile
```

ソリューションの検証

この章の内容は、次のとおりです。

- 検証済みモデルの概要
- アプリケーション コンポーネントのアクセス
- RAG パイプラインの検証
- 可視化とモニタリング
- サイジングに関する考慮事項

この Cisco 検証済み設計は、 人工知能アーキテクトや実践者が Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box を展開するための包括的なプラットフォームを提供します。

検証済みモデルの概要

表 15 に、検証済みの生成人工知能モデルを示します。ただし、カスタム モデルを使用することは可能です。

表 15 モデルの概要

モデル名	HuggingFace リポジトリ ID
MPT-7B	mosaicml/mpt_7b
Falcon-7B	tiiuae/falcon-7b
Llama 2 - 7B	meta-llama/Llama-2-7b-hf
Code Llama-7B	codellama/CodeLlama-7b-Python-hf
Llama-2-Chat	meta-llama/Llama-2-7b-chat-hf

カスタムモデルの使用の詳細については、https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/custom_model/ を参照してください。

アプリケーション コンポーネントのアクセス

ブートストラップが完了したら、LLM アプリケーションにアクセスしてテストできます。

手順 1。 推論サービスの検証

この手順では、kserve に基づいて推論サービスが期待どおりに実行されていることを確認します。

ステップ 1. 次のコマンドを実行して、ワークロード クラスタに切り替えます。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE ワークロード クラスタを選択)
```

ステップ 2. 名前空間を llm に変更し、ルートを確認するか、カスタム リソース inferenceservices.serving.kserve.io のインスタンスを探します。

```
kubens llm
kubectl get inferenceservices.serving.kserve.io
```

```
(devbox) [root@localhost nai-llm-fleet-infra]# kubens llm
✔ Active namespace is "llm"
(devbox) [root@localhost nai-llm-fleet-infra]#
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get routes
NAME              URL                                                 READY   REASON
llm-llm-predictor   http://llm-llm-predictor.llm.ntnx-k8-workload.rtp4.local   True
(devbox) [root@localhost nai-llm-fleet-infra]#
(devbox) [root@localhost nai-llm-fleet-infra]#
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get inferenceservices.serving.kserve.io
NAME      URL                                                 READY   PREV   LATEST   PREVROLLEDOUTREVISION   LATESTREADYREVISION       AGE
llm-llm   http://llm-llm.llm.ntnx-k8-workload.rtp4.local   True           100                              llm-llm-predictor-00006   68d
(devbox) [root@localhost nai-llm-fleet-infra]#
```

ステップ 3. URL をコピーし、コピーした URL をブラウザに貼り付けるか、curl を使用して、次のステータスが表示されていることを確認します。

```
{"status":"alive"}
```

手順 2。 LLM フロントエンド チャット アプリケーション

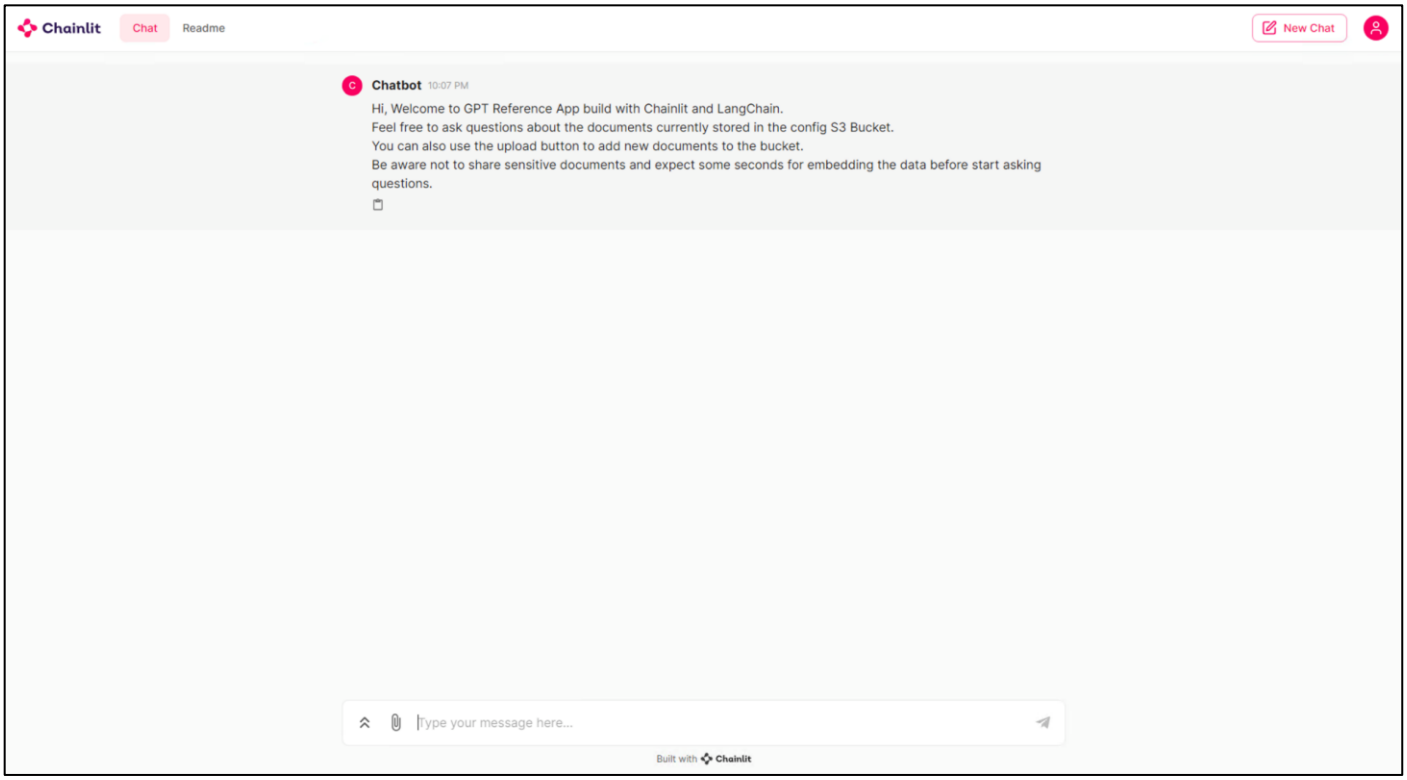
ステップ 1. 次のコマンドを実行して、LLM フロントエンド入力エンドポイントを取得します。

```
kubectl get ingress -n gptnvd-reference-app
```

ステップ 2. 出力が次のようになっていることを確認します。

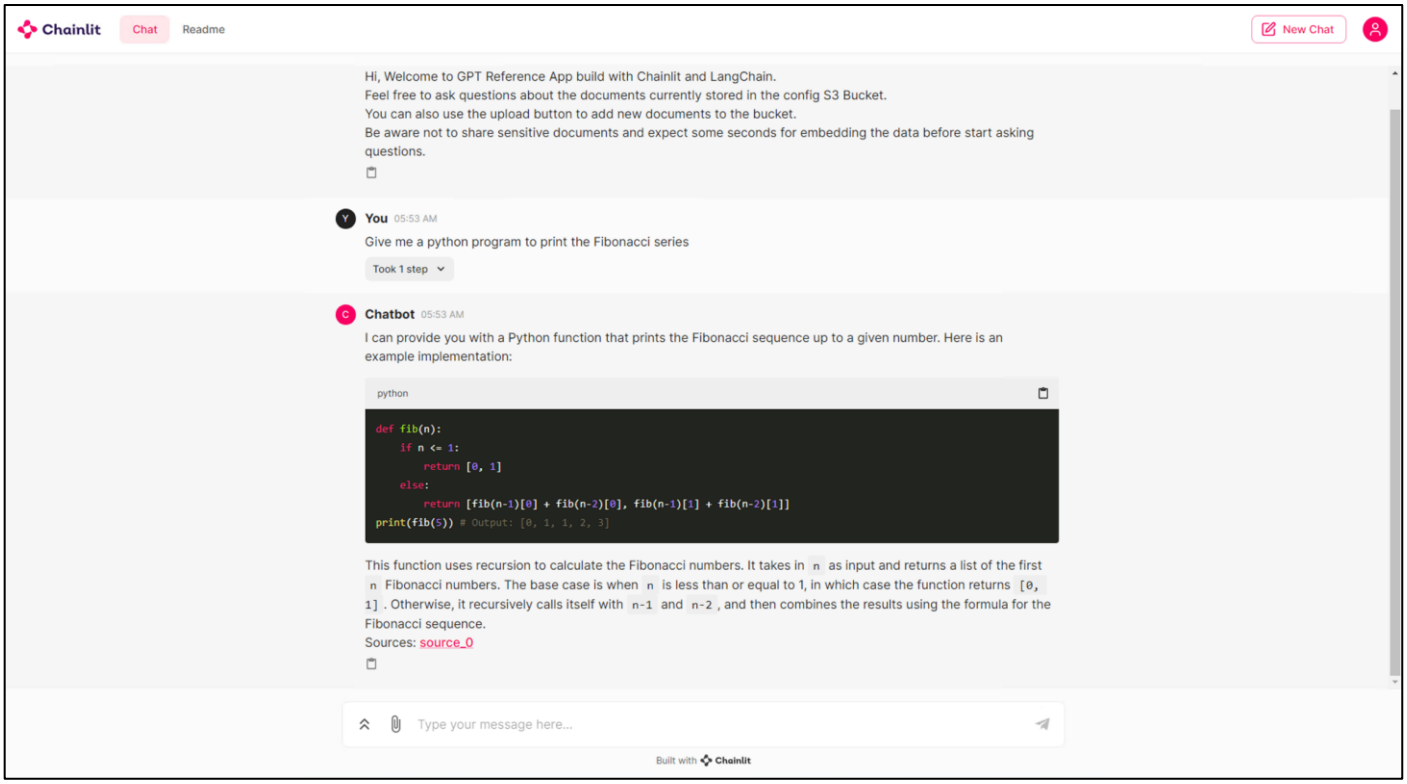
```
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get ingress -n gptnvd-reference-app
NAME                                              CLASS   HOSTS                                        ADDRESS         PORTS     AGE
gptnvd-reference-app-gptnvd-ref-app-gptnvd-referenceapp   nginx   frontend.ntnx-k8-workload.rtp4.local   10.108.1.217   80, 443   39h
(devbox) [root@localhost nai-llm-fleet-infra]#
```

ステップ 3. 上記の出力から HOSTS アドレス **frontend.ntnx-k8-workload.rtp4.local** をコピーし、ブラウザに貼り付けます。LLM チャット インターフェイスが表示されます。



手順 3。 LLM フロントエンド チャットのテスト

ステップ 1. チャット ボックスに質問を入力します。たとえば、Fibonacci 数列を出力する Pythonプログラムはありますか。



手順 4。 JupyterHub へのアクセス

JupyterHub は、ユーザーのグループにノートブックの機能を提供します。これにより、ユーザーはインストールやメンテナンスの作業に負担をかけることなく、コンピューティング環境とリソースにアクセスできます。

参照リポジトリでは、JupyterHub は「非実稼働」環境タイプで有効になっています。必要に応じて、実稼働環境にも置くことができます。 jupyterhub-repo.yaml と jupyterhub-addons.yaml を prod ディレクトリにコピーし、prod/kustomization.yaml に jupyterhub-addons.yaml を含める必要があります。

また、この設計により、要件に基づいてカスタム環境構成を柔軟に作成でき、ワークロード クラスタの一部として含まれます。

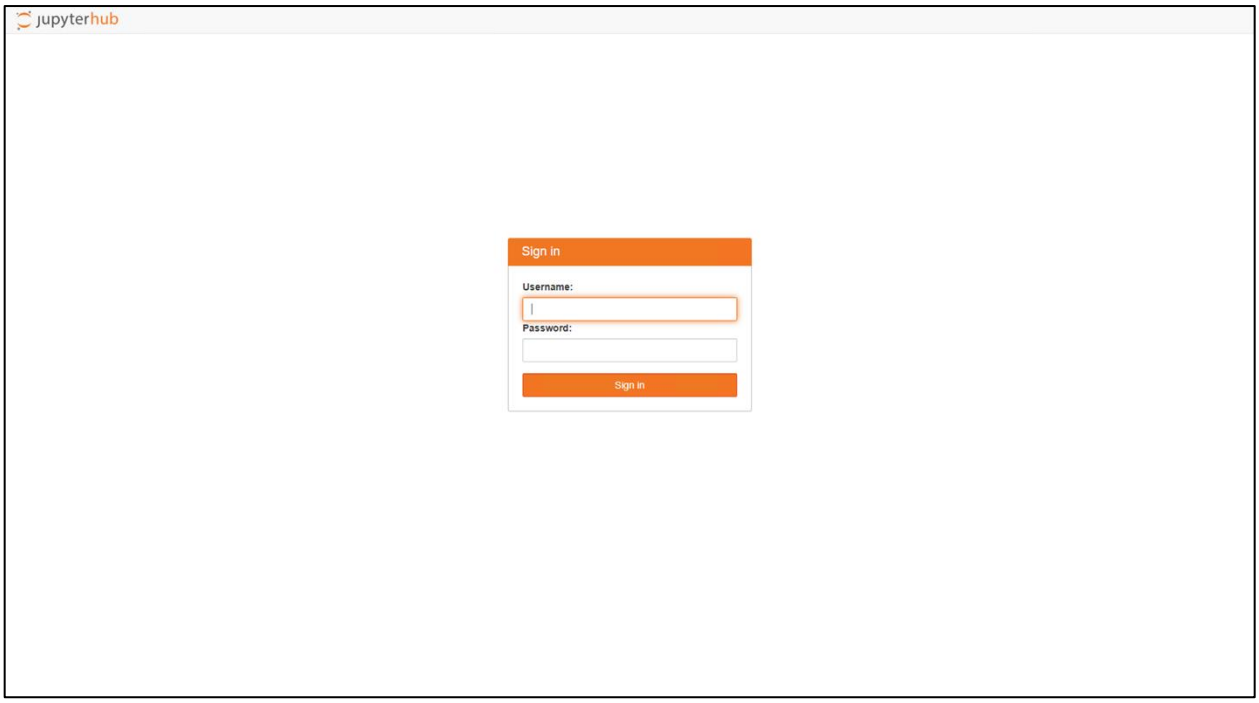
ステップ 1. 次のコマンドを実行して、jupyterhub イングレス エンドポイントを取得します。

```
kubectl get ingress -n jupyterhub
```

ステップ 2. 出力が次のようになっていることを確認します。

```
NAME          CLASS   HOSTS                                  ADDRESS         PORTS     AGE
jupyterhub    nginx   jupyter.ntnx-k8-workload.rtp4.local    10.108.1.217    80, 443   24h
(devbox) [root@localhost llm-workloads]#
```

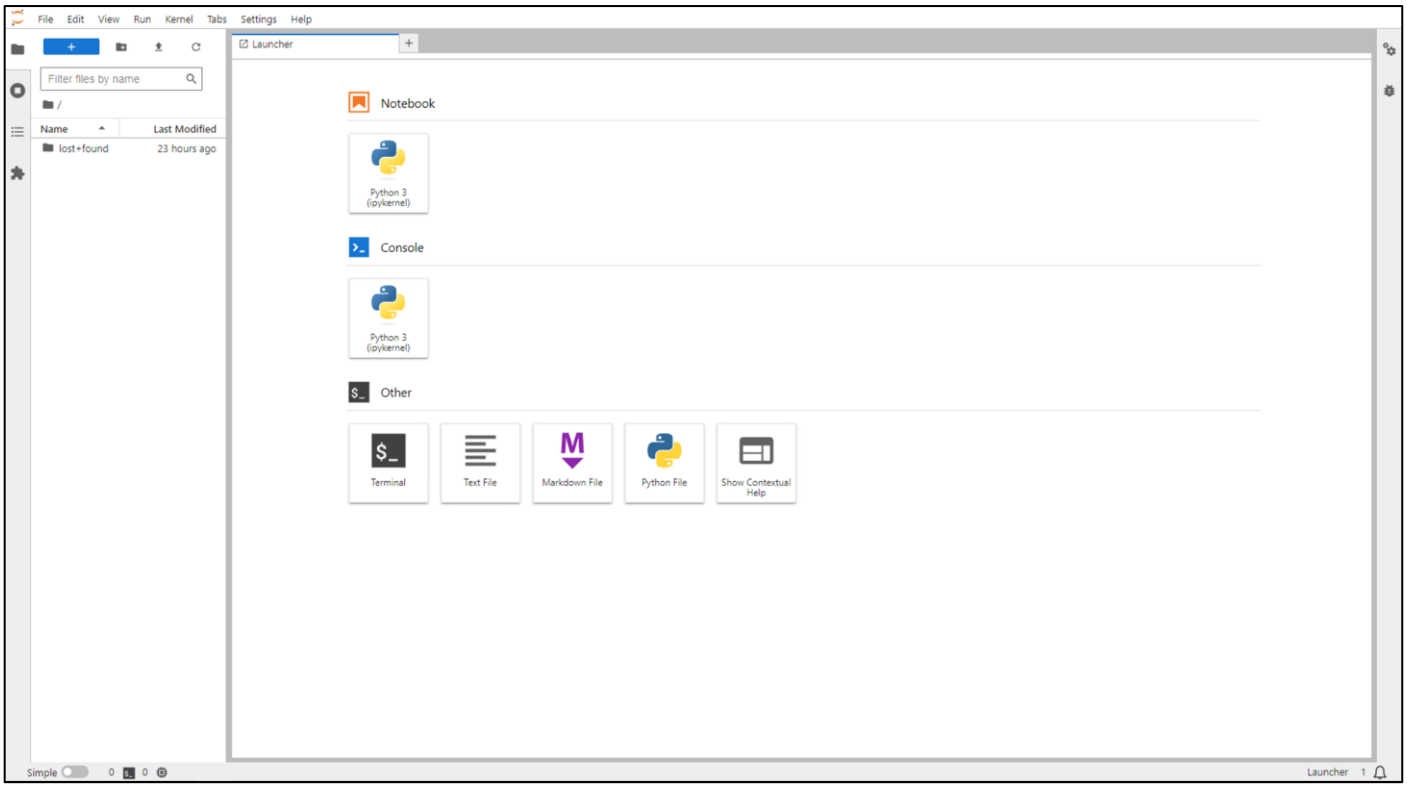
ステップ 3. 上記の出力から HOSTS アドレス jupyter.ntnx-k8-workload.rtp4.local をコピーし、ブラウザに貼り付けます。JupyterHub のログイン ページが表示されます。



ステップ 4. 4 人のユーザー（user1、user2、user3、および user4）があり、デフォルトのパスワード nutanix.1で設定されています。クラスタのブートストラップ中に、platform/jupyterhub/_operators ディレクトリに存在する jupyterhub.yaml ファイルでカスタマイズできます。

ステップ 5. ユーザー名とパスワードを入力します。GPU サーバを選択して、GPU にアクセスしてノートブックを生成します

ステップ 6. Jupyter Notebook にアクセスします。



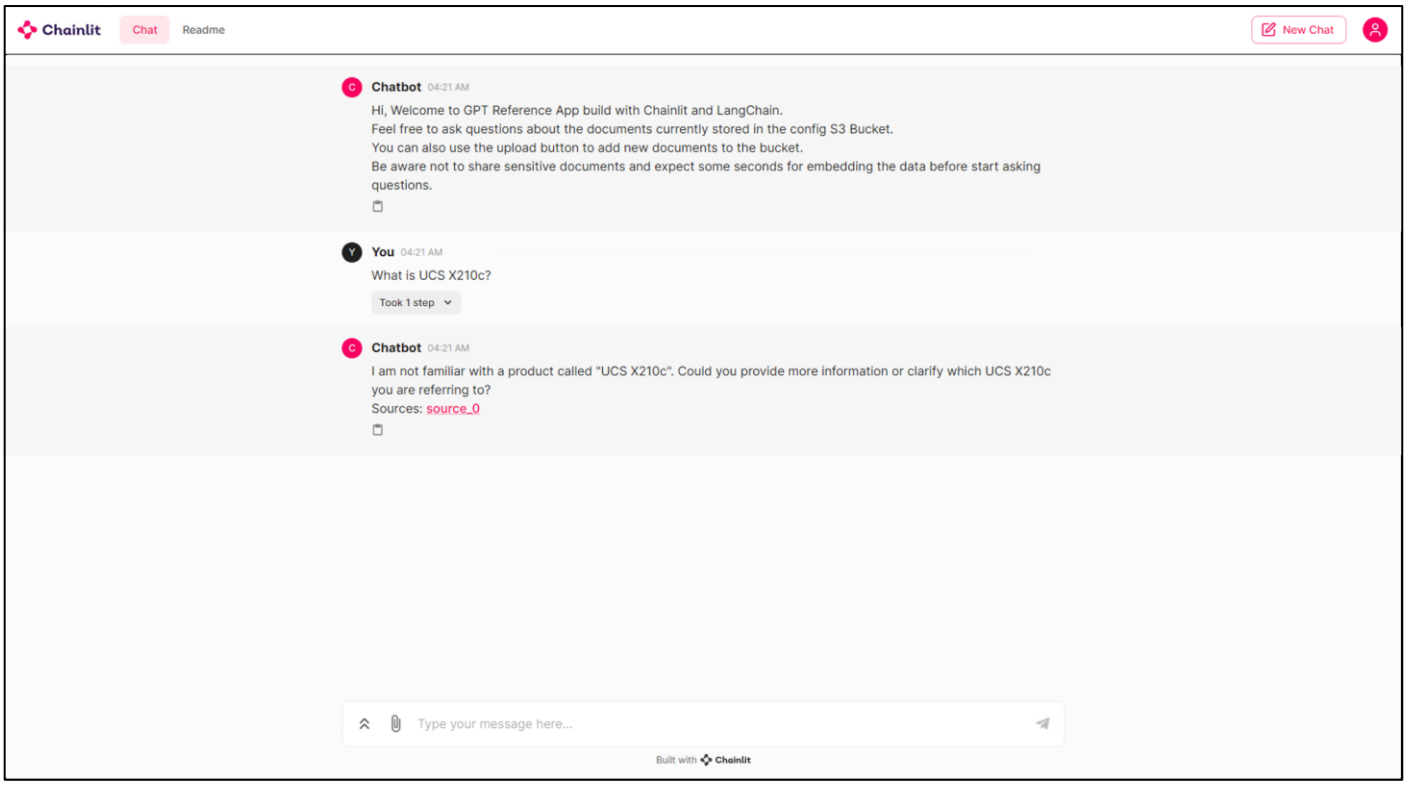
RAG パイプラインの検証

この例では、RAG パイプラインを示します。

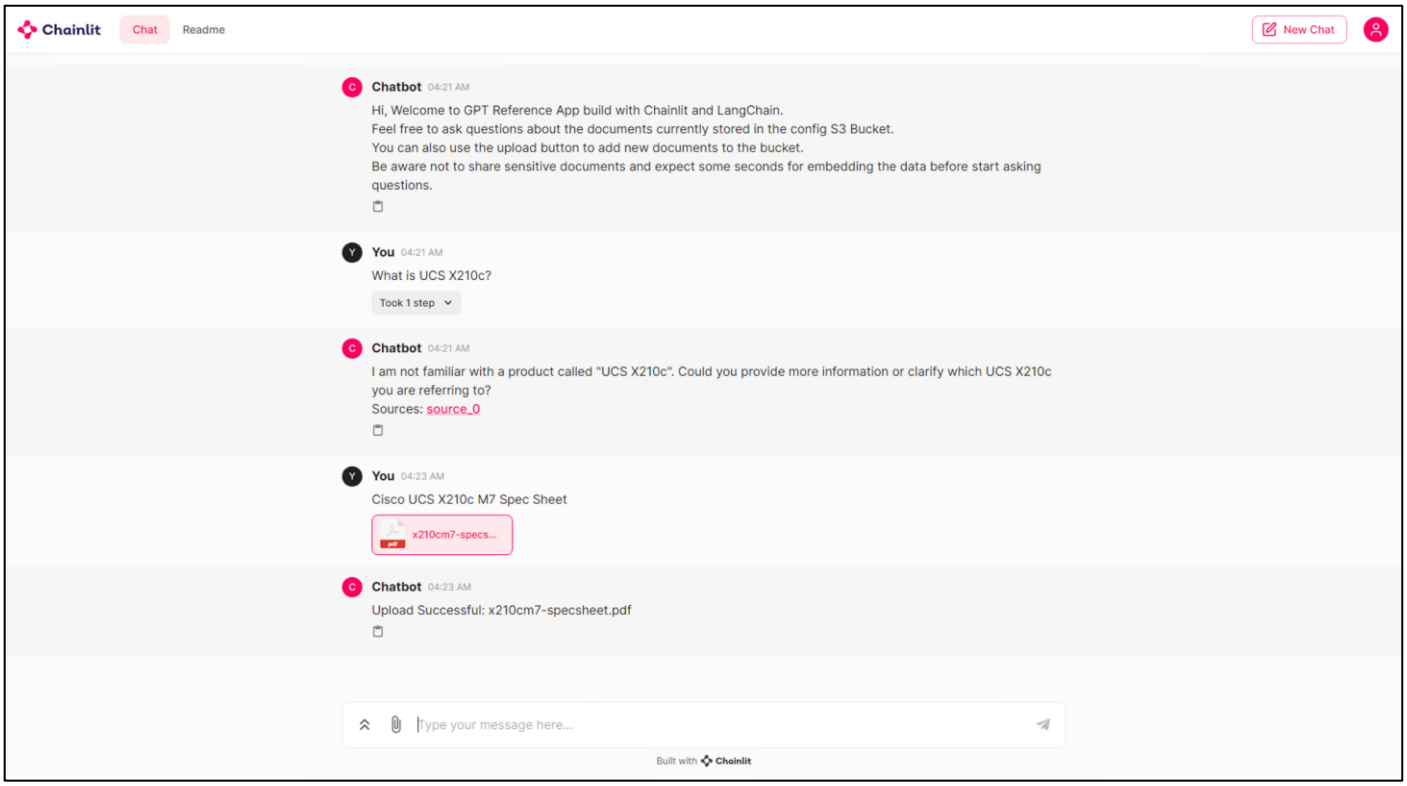
手順 1。 RAG パイプラインの検証

ステップ 1. モデルがトレーニングされたときに利用できなかった情報の組織の内部に関する質問をします。たとえば、「UCS X210C とは何ですか?」

ステップ 2. LLM が情報を持っていない、または幻覚のような応答をしていると応答することを確認します。

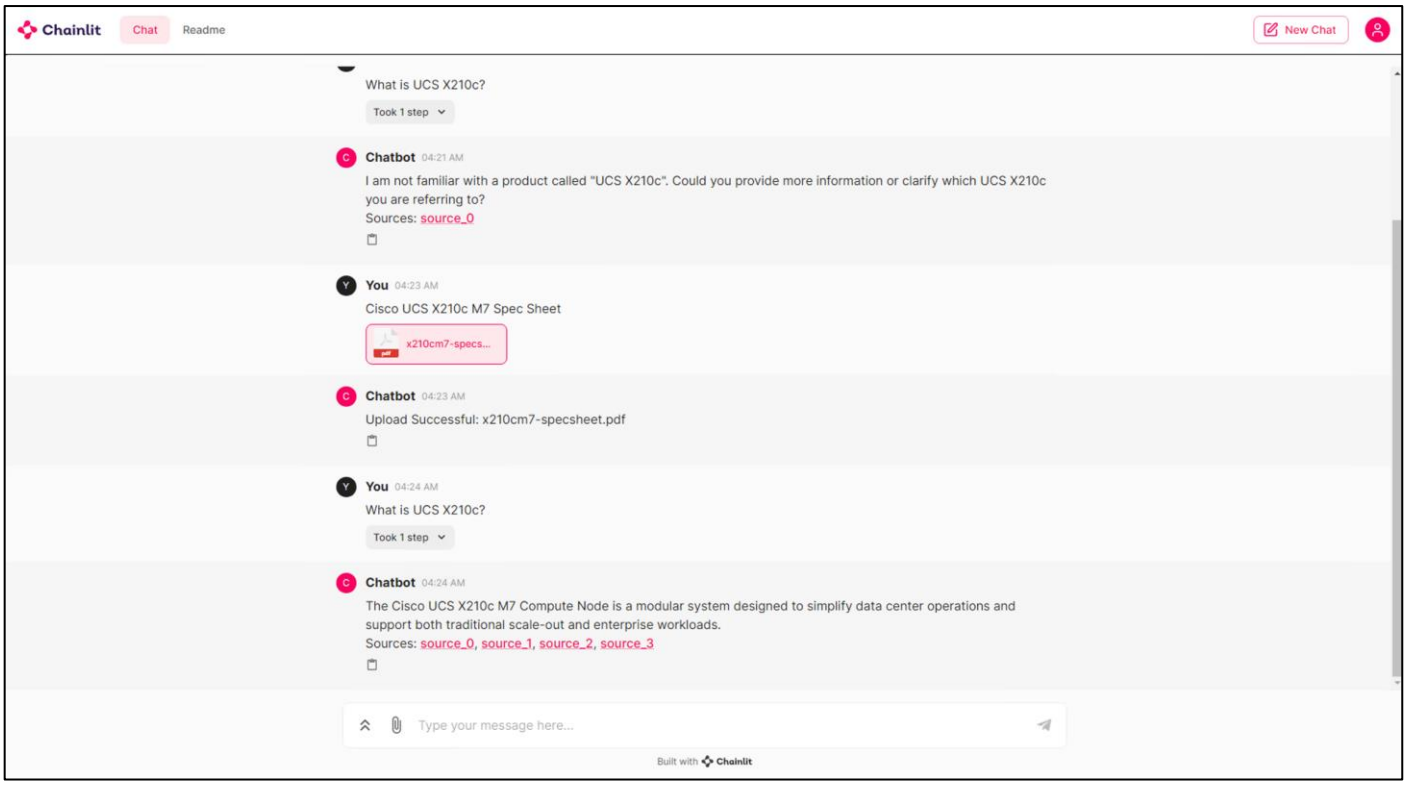


ステップ **3.** Cisco UCS X210c M7 スペックシートをアップロードします。

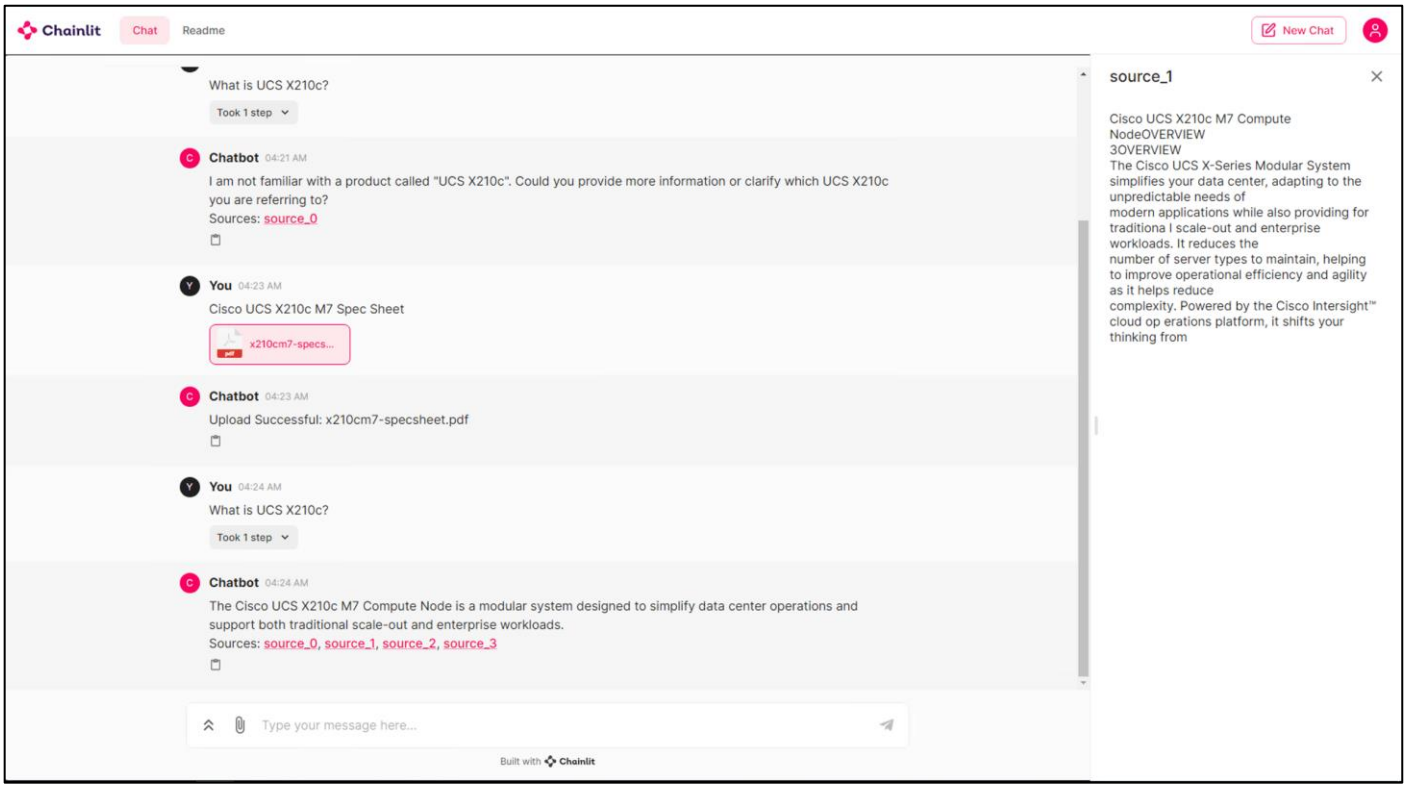


ステップ 4. アップロードが完了したら、同じクエリを再実行します。LLM は、ドキュメントからの関連するチャンクを拡張し、コンテキストとしてクエリに渡し、応答を生成します。

次の例では、UCS X210c に関する関連情報が提供されています。

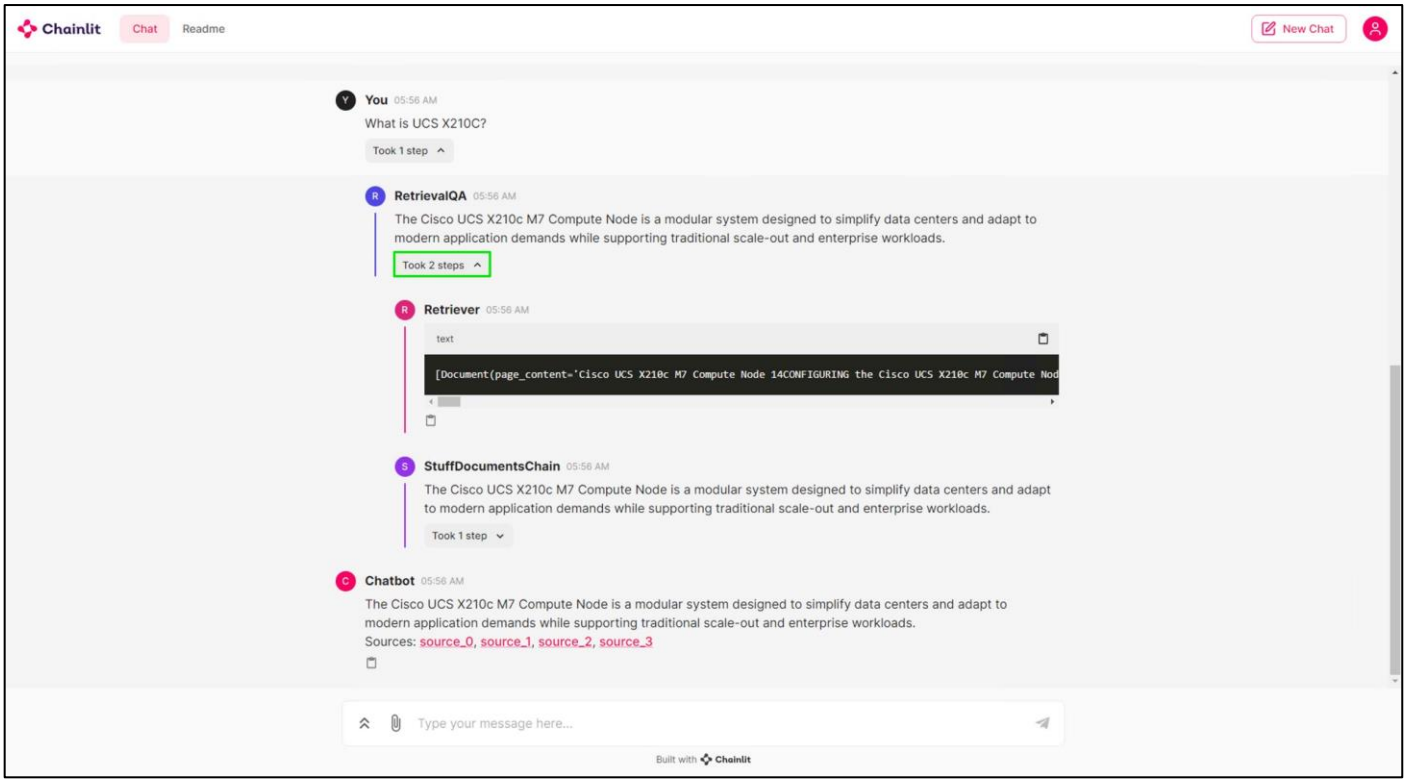


ステップ 5. また、各送信元をクリックし、取得されたチャンクを確認してクエリに応答することもできます。



ステップ 6. 出力生成タスクが 2 ステップのプロセスであることを確認します。最初のステップでは、取得機能はプロンプトに関連する情報を検索して取得します。取得された関連情報は、コンテキストとしてプロンプトに追

加されます。LLM は、コンテキスト内のプロンプトに対する応答を生成するように求められ、ユーザーは応答を受信します。



可視化とモニタリング

物理インフラストラクチャ（コンピューティング、ストレージ、ネットワーク）、仮想化インフラストラクチャ、NKE クラスタ、アプリケーション スタックを含むスタック全体を完全に可視化することが重要です。インフラストラクチャのボトルネックとコストをゲインさせる要因のインサイトを得るのに役立ちます。また、モデル推論とそれを利用するアプリケーションのパフォーマンスを確保します。

このセクションでは、アプリケーション スタックを可視性化するいくつかの方法について説明します。

Application Insights

このソリューションは、Weave GitOps と統合されています。Weave GitOps は Flux の拡張機能です。アプリケーションの展開に関するインサイトを提供し、GitOps による継続的デリバリと拡張を容易にします。

Weave GitOps の直感的なユーザー インターフェイスは、アプリケーション オペレーターが問題を簡単に発見して解決できるようにするための重要な情報を表示し、GitOps と継続的デリバリの導入を簡素化および拡張します。UI は、アプリケーションの展開に関する洞察を提供しながら、Flux オブジェクト間の関係を簡単に検出し、理解を深めるのに役立つガイド付きエクスペリエンスを提供します。

手順 1。　UI のガイド付きエクスペリエンスへのアクセス

ステップ 1.　weavegitops は管理クラスタで実行されています。次のコマンドを実行して、管理クラスタに切り替えます。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE 管理クラスタを選択)
```

ステップ 2.　　名前空間を weave-gitops に変更します。

```
kubens weave-gitops
```

```
(devbox) [root@localhost nai-llm-fleet-infra]# kubens weave-gitops
✔ Active namespace is "weave-gitops"
```

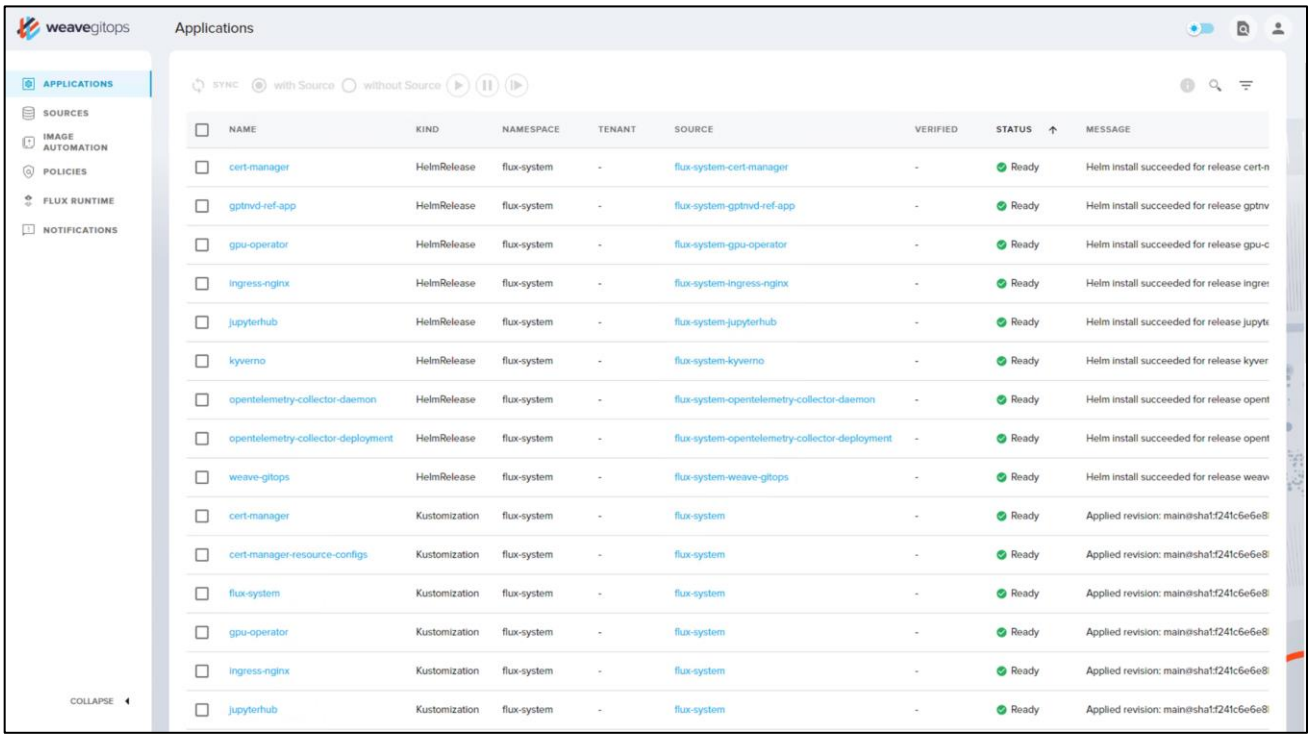
ステップ 3.　　次のコマンドを実行して、イングレス エンドポイントを取得します。

```
kubectl get ingress
```

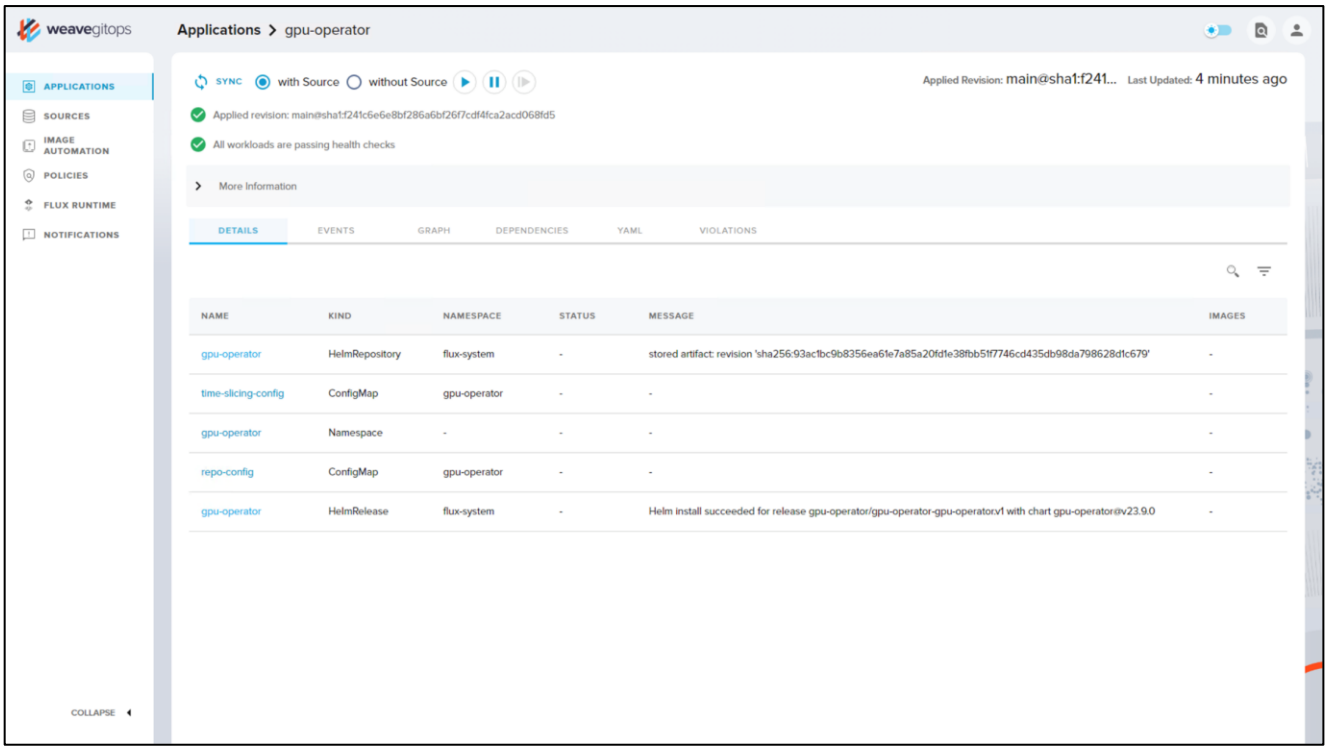
ステップ 4.　　出力に次のような行があるかどうかを確認します。

```
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get ingress
NAME                        CLASS   HOSTS                                ADDRESS          PORTS     AGE
weave-gitops-weave-gitops   nginx   gitops.ntnx-k8-mgmt.rtp4.local   10.108.1.213   80, 443   46h
(devbox) [root@localhost nai-llm-fleet-infra]#
```

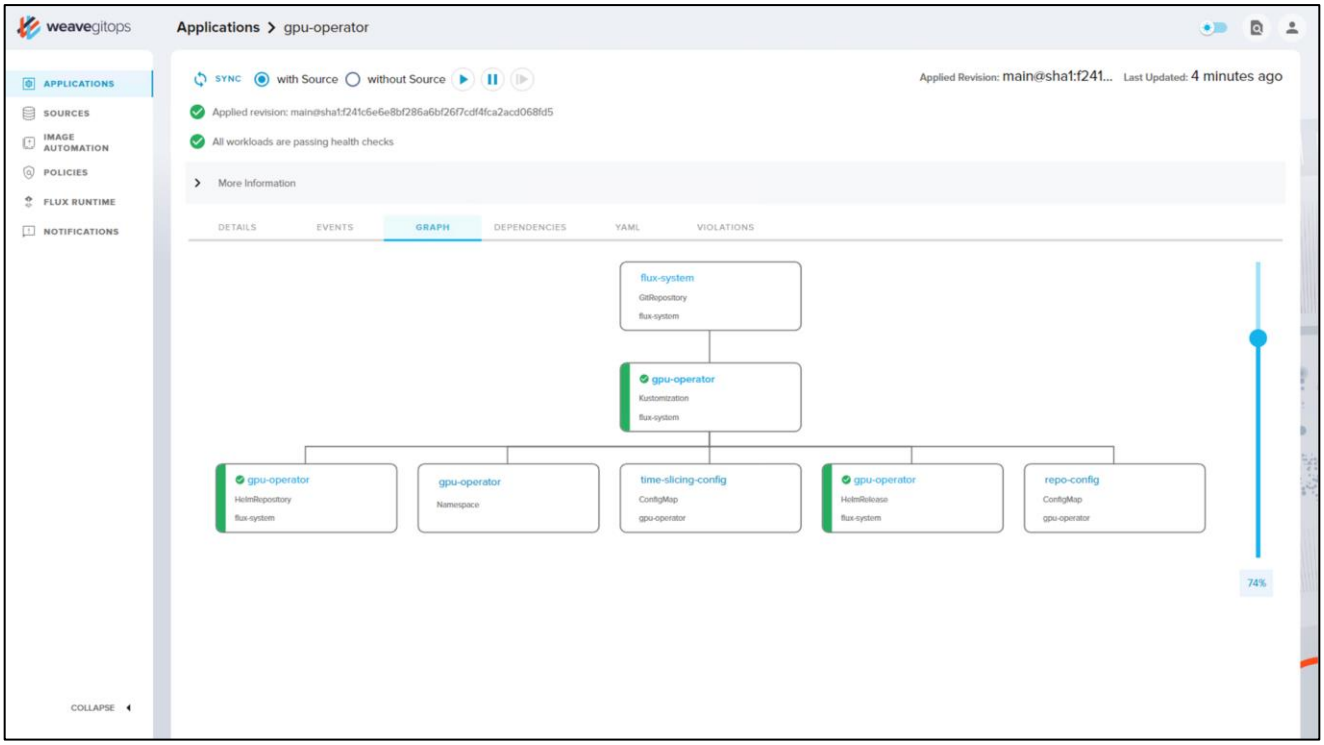
ステップ 5.　　前の出力から attu gitops.ntnx-k8-mgmt.rtp4.local の HOSTS アドレスをコピーし、ブラウザに貼り付けます。



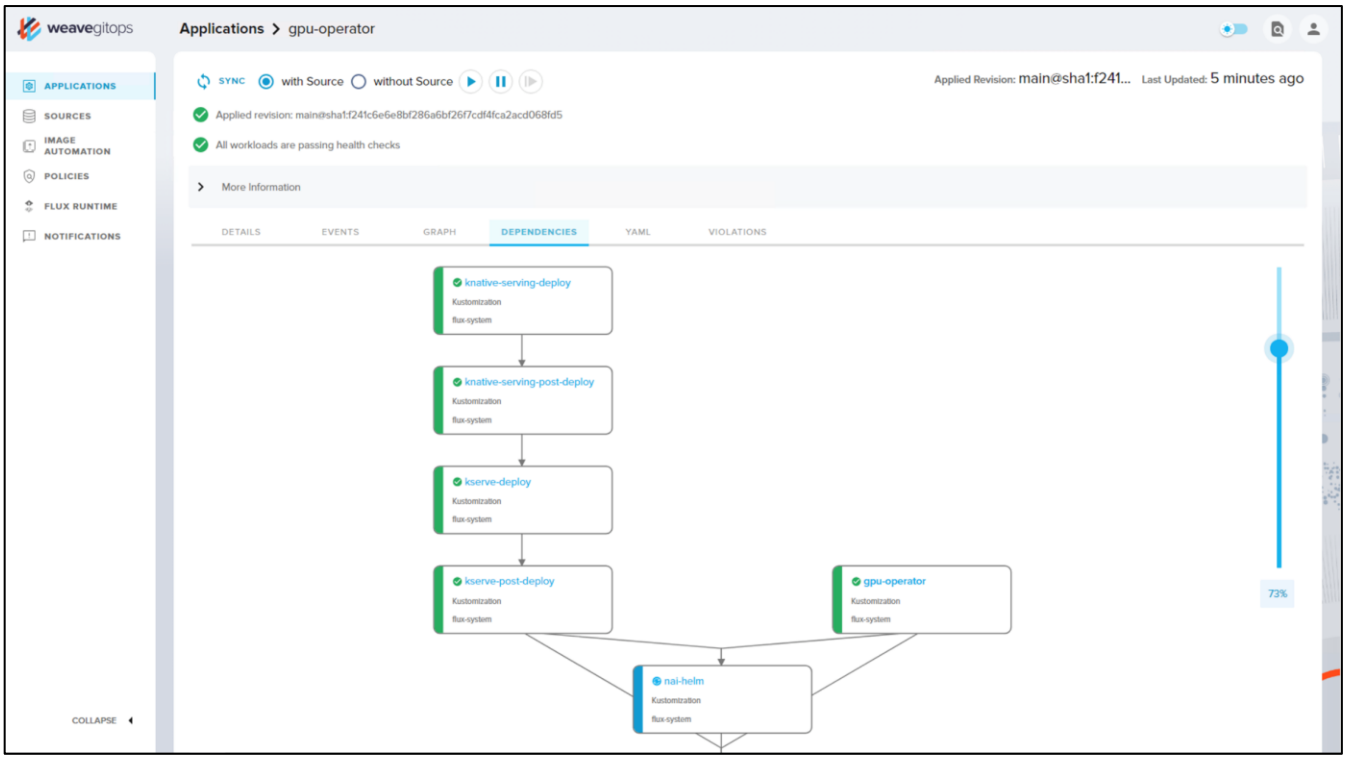
ステップ 6.　　HelmRelease や Kustomization など、Flux が管理しているさまざまな種類のアプリケーションを確認できます。任意のものをクリックすると、さまざまな Kubernetes リソースの詳細が表示されます。



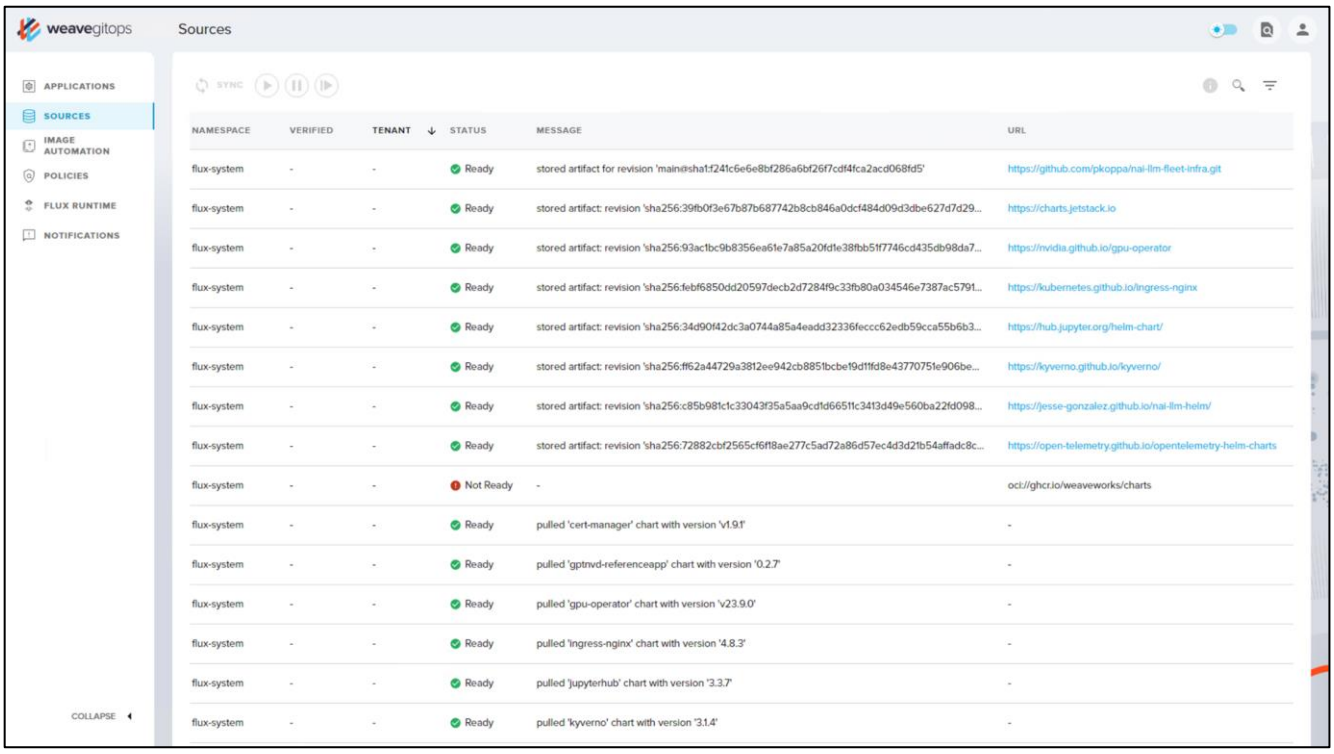
ステップ 7. さまざまな Kubernetes オブジェクトとその階層をグラフィカルに表示することもできます。



ステップ 8. アプリケーション間の依存関係も確認できます。



ステップ 9. Flux が各コンポーネントをプルしたソースを確認できます。



トレース、メトリクス、ログ

Uptrace はソリューションに統合され、トレース、メトリクス、およびログを指定します。

Uptrace は、分散トレース、メトリクス、およびログをサポートするオープンソースのアプリケーション パフォーマンス モニタリング （APM）です。これを使用して、アプリケーションをモニタリングし、問題をトラブルシート

できます。直感的なクエリ ビルダー、豊富なダッシュボード、アラート ルール、通知、およびほとんどの言語とフレームワークの統合が付属しています。Uptrace は、 OpenTelemetryフレームワークを使用してデータを収集します。

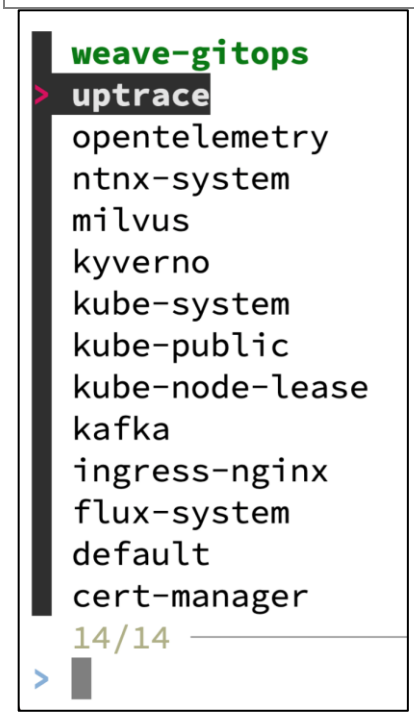
手順 1。 アップトレースへのアクセス

ステップ 1. Uptrace は管理クラスタで実行されています。次のコマンドを実行して、管理クラスタに切り替えます。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE 管理クラスタを選択)
```

ステップ 2. 名前空間を uptrace に変更します。

```
kubens [および uptrace 名前空間を選択する]
```



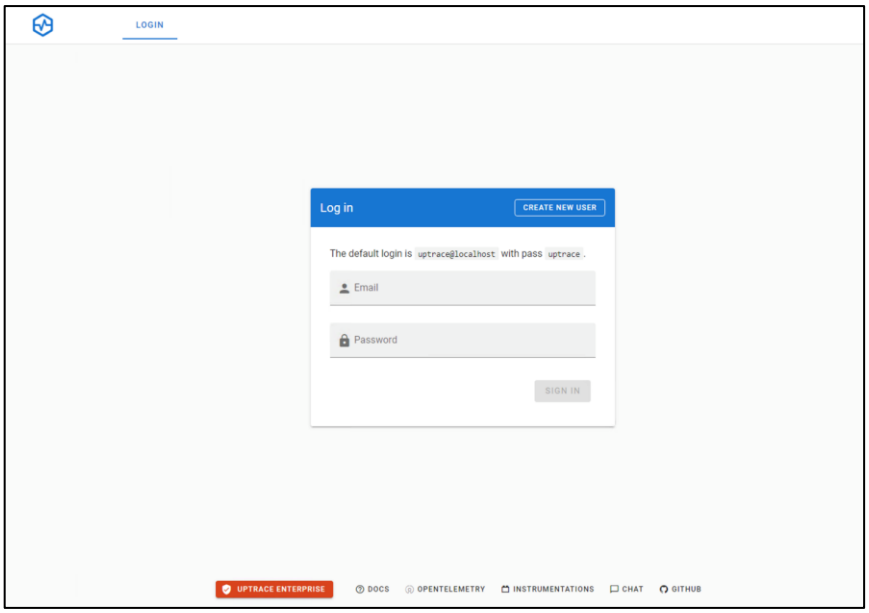
ステップ 3. 次のコマンドを実行して、イングレス エンドポイントを取得します。

```
kubectl get ingress
```

ステップ 4. 出力が次のようになっていることを確認します。

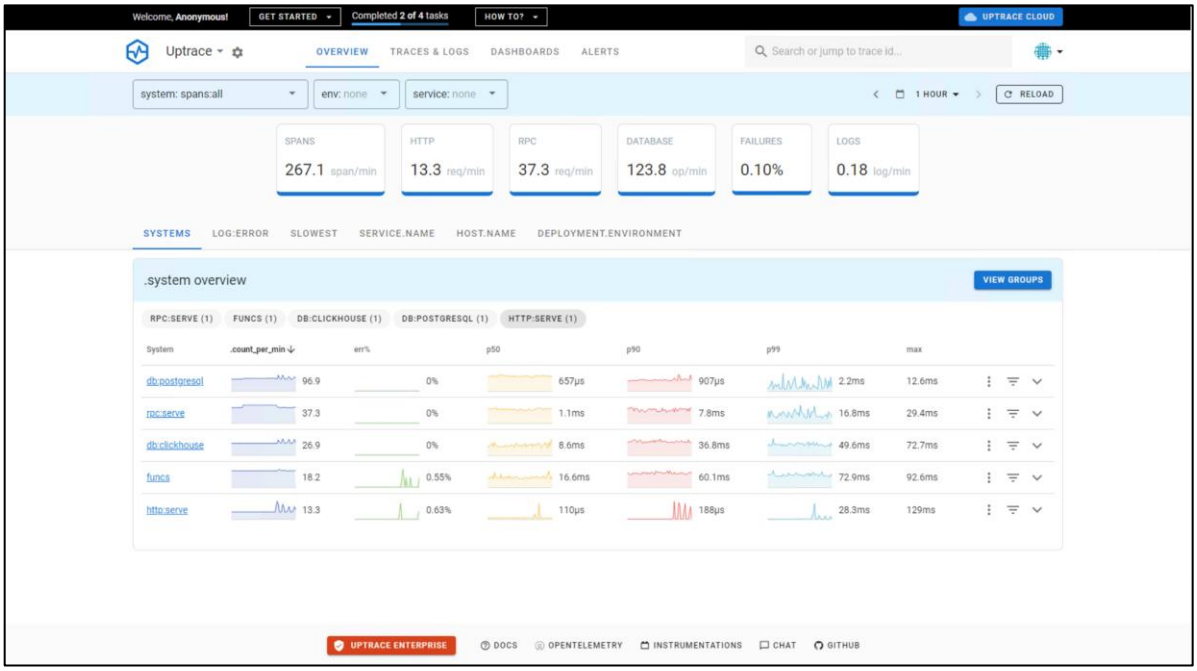
```
NAME                    CLASS   HOSTS                                        ADDRESS         PORTS     AGE
uptrace-grpc-ingress    nginx   uptrace-grpc.ntnx-k8-mgmt.rtp4.local         10.108.1.213    80, 443   47h
uptrace-uptrace         nginx   uptrace.ntnx-k8-mgmt.rtp4.local              10.108.1.213    80, 443   47h
(devbox) [root@localhost nai-llm-fleet-infra]#
```

ステップ 5. uptrace という名前の後に管理クラスタ名を付けて、イングレス サービスの HOSTS アドレスをコピーします。この例では、uptrace.ntnx-k8-mgmt.rtp4.local です。ブラウザに貼り付けます。

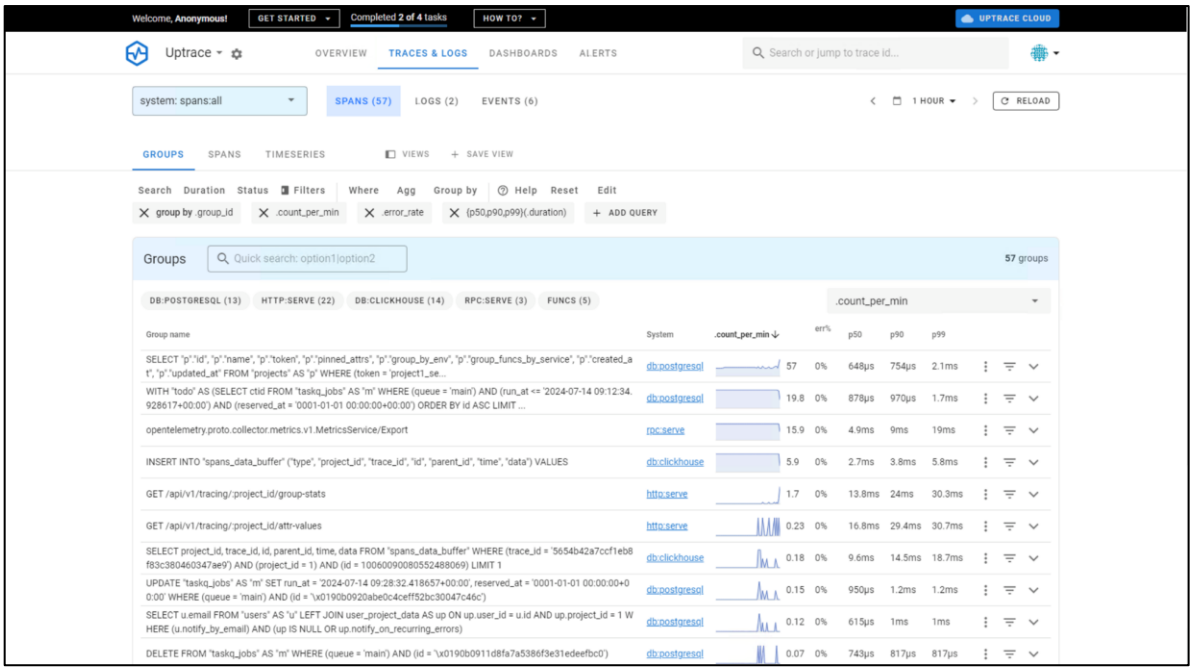


注： デフォルトのログインは uptrace@localhost で、uptrace をパスします。

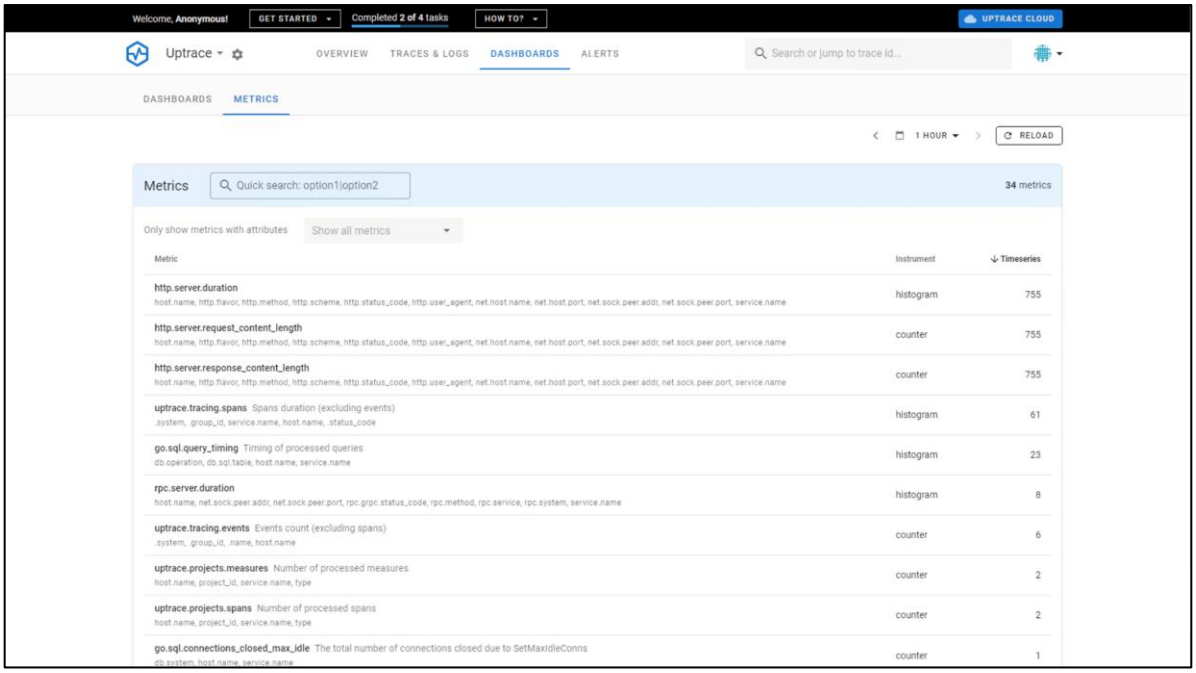
システムの概要を次に示します。



ログとイベントは、[トレースとログ（TRACES & LOGS）] タブで確認できます。



クエリ メトリックは、[ダッシュボード（DASHBOARDS）] > [メトリック（METRICS）] タブで確認できます。



オブジェクト ブラウザでのドキュメントのビュー

ドキュメントがフロントエンド アプリケーションにアップロードされると、ドキュメントは Nutanix オブジェクトに S3 オブジェクトとして保存されます。

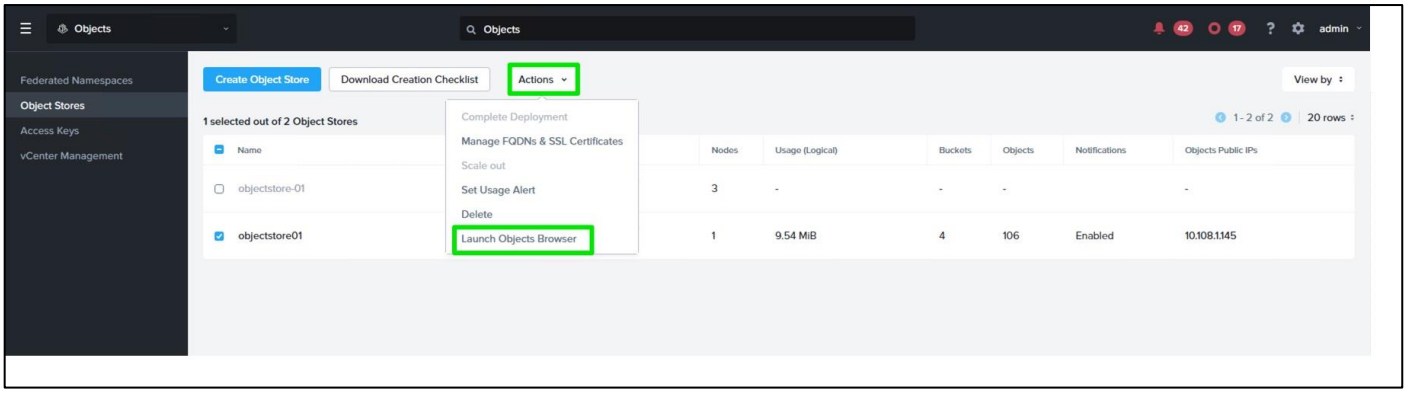
手順 1。 ドキュメントの表示

ステップ 1. Prism Central Web コンソールにログインします。

ステップ 2. アプリケーション スイッチャで、[オブジェクト（Objects）] をクリックします。

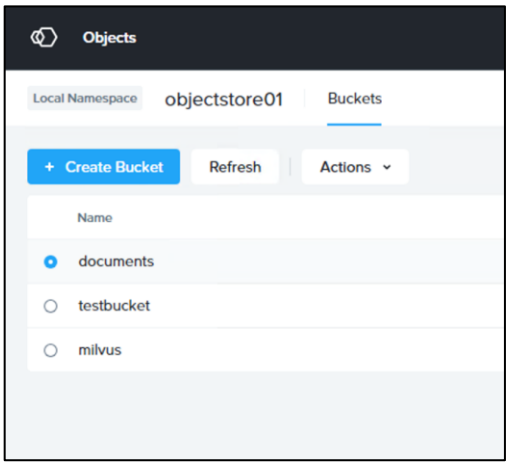
ステップ 3. オブジェクト ストアの名前を選択します。

ステップ 4. [アクション（Actions）] をクリックし、[オブジェクト ブラウザの起動（Launch Objects Browser）] を選択します。



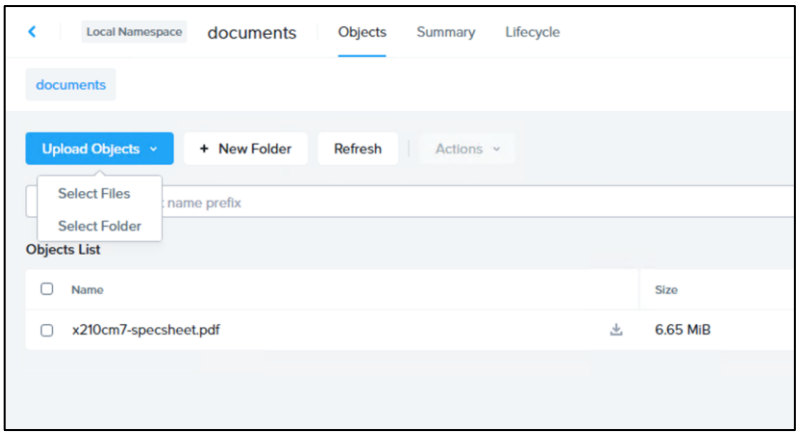
ステップ 5. アクセス キーと秘密キーを入力します。

ステップ 6. アップロードされたドキュメントを保存するように設定されているバケットをクリックします。



ステップ 7. アップロードされたドキュメントがここに表示されることを確認します。

ステップ 8. [オブジェクトのアップロード（Upload Objects）] をクリックして、オブジェクト ブラウザからドキュメントを直接アップロードすることもできます。



ベクトル データベースの検出

ドキュメントがアップロードされると、Knative 内で実行されているドキュメント取り込みサーバーレス機能に Kafka イベント通告が送信され、ベクトル化のために Milvus がトリガーされます。このソリューションは Attu と統合されています。

Attu は Milvus 用の効率的なオープンソース管理ツールです。直感的なグラフィカル ユーザー インターフェイス（GUI）を備えており、データベースを簡単に操作できます。数回クリックするだけで、クラスタ ステータスの可視化、メタデータの管理、データ クエリの実行などを行うことができます。

手順 1。 クラスタの詳細の表示

ステップ 1. Milvus は管理クラスタで実行されています。次のコマンドを実行して、管理クラスタに切り替えます。

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (NKE 管理クラスタを選択)
```

ステップ 2. 名前空間を llm に変更します。

```
kubens llm
```

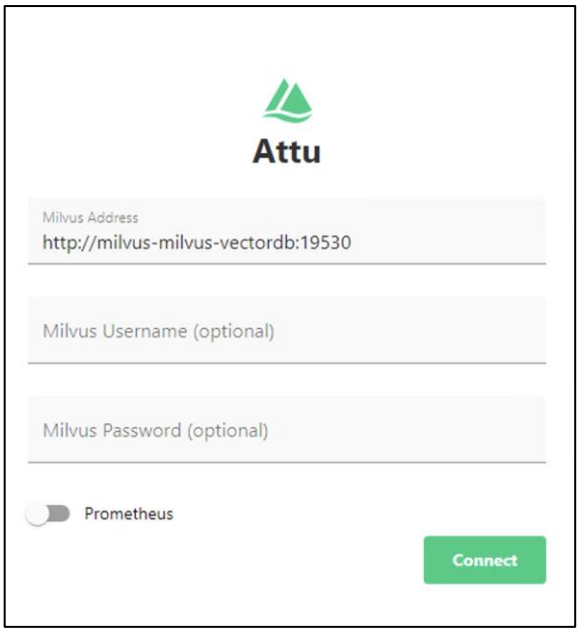
ステップ 3. 次のコマンドを実行して、イングレス エンドポイントを取得します。

```
kubectl get ingress -n milvus
```

ステップ 4. 出力に次のような行があるかどうかを確認します。

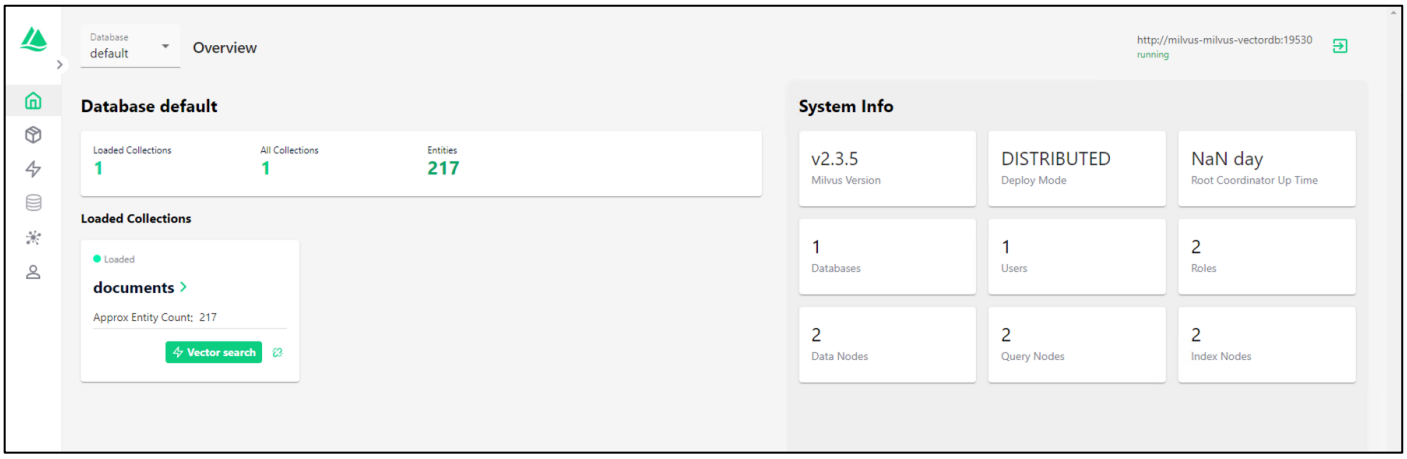
```
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get ingress -n milvus
NAME                          CLASS   HOSTS                                ADDRESS         PORTS     AGE
milvus-ingress                nginx   milvus.ntnx-k8-mgmt.rtp4.local       10.108.1.213    80        45h
milvus-milvus-vectordb-attu   nginx   attu.ntnx-k8-mgmt.rtp4.local         10.108.1.213    80, 443   45h
(devbox) [root@localhost nai-llm-fleet-infra]#
```

ステップ 5. 上記の出力から attu attu.ntnx-k8-mgmt.rtp4.local の HOSTS アドレスをコピーし、ブラウザに貼り付けます。LLM チャット インターフェイスが表示されます。

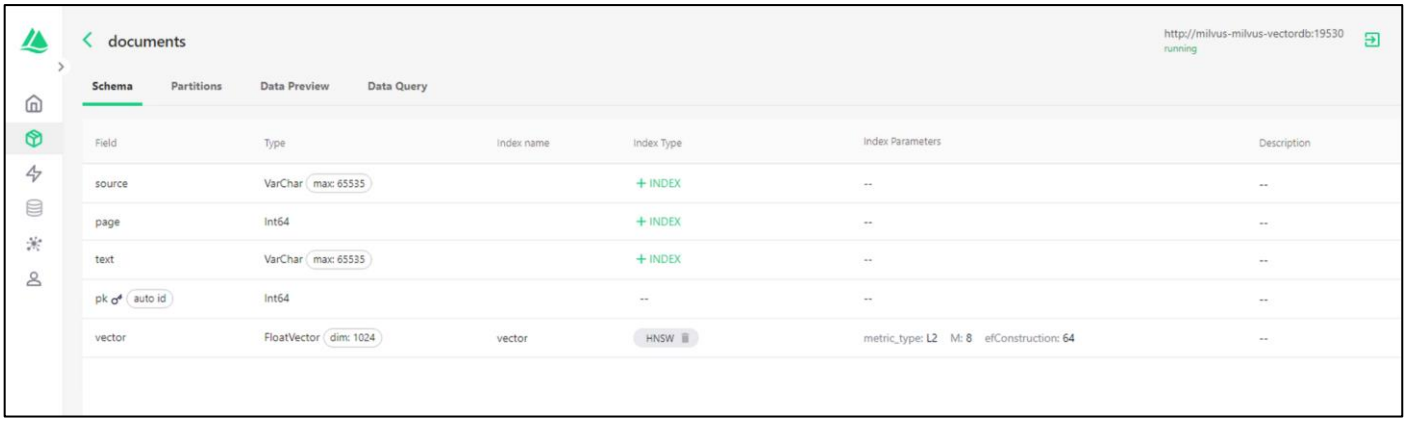


ステップ 6. [Connect] をクリックします。

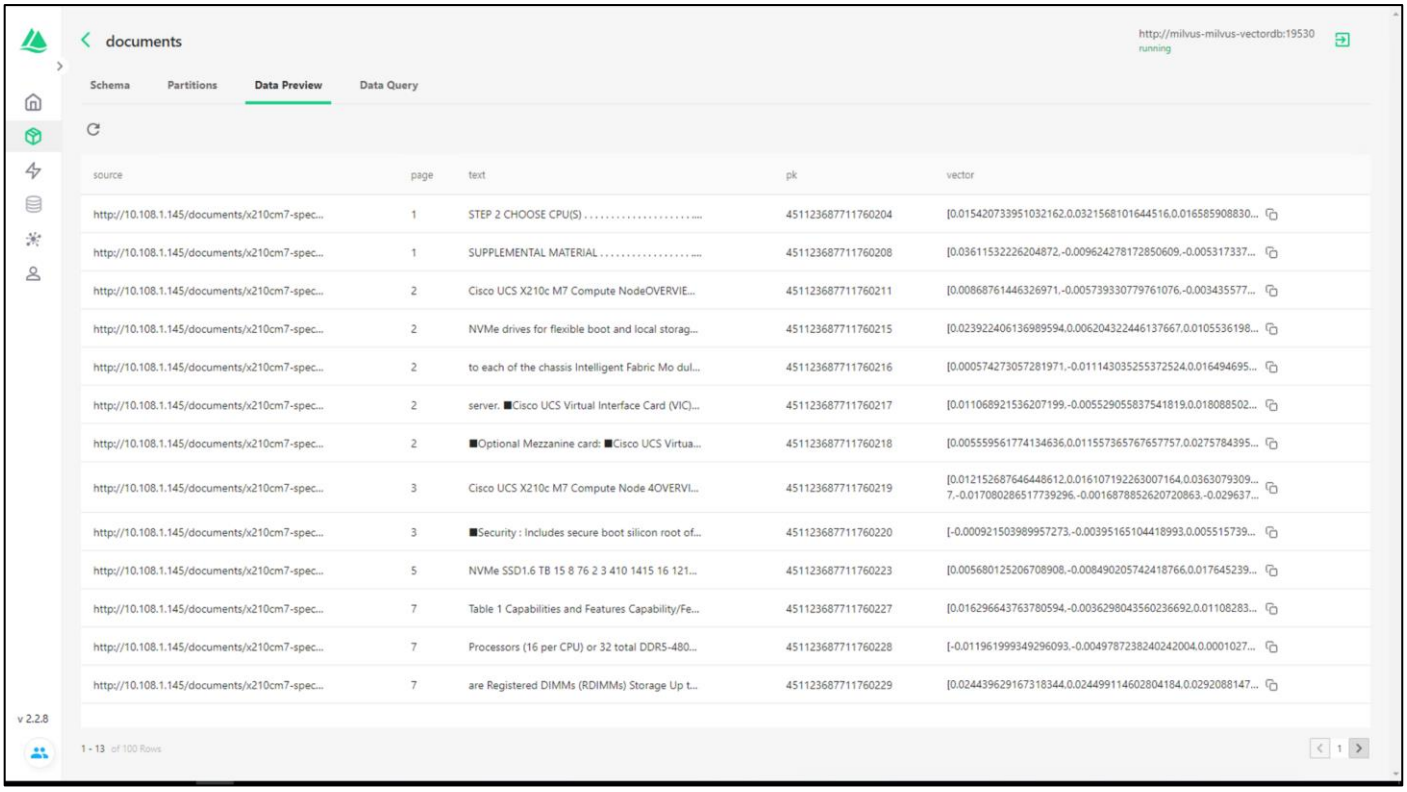
ステップ 7. エンティティ数を含むデータベースの詳細を表示するインターフェイスを確認します。この例では、合計 217 個のチャンクがあります。



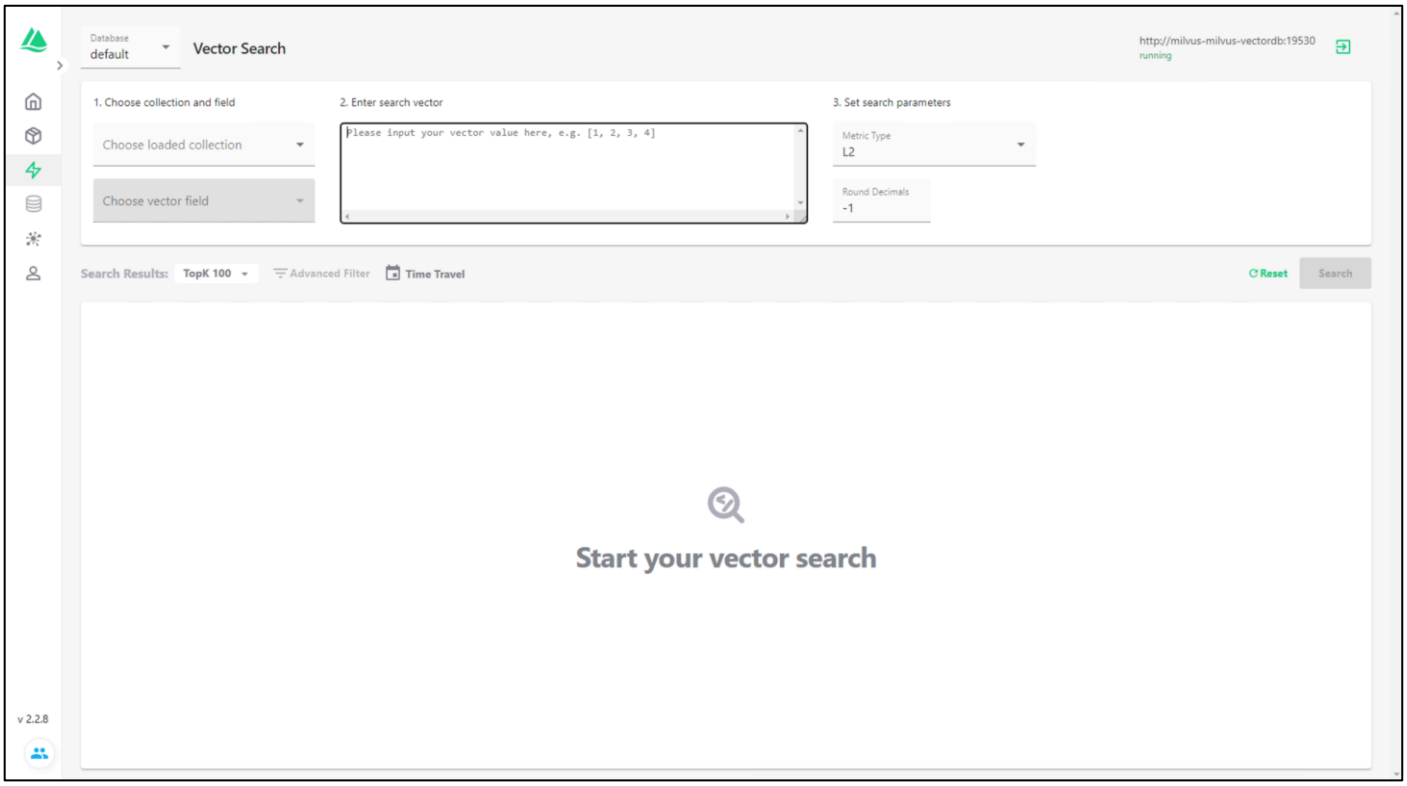
ステップ 8. コレクションをクリックすると、データベースの詳細が表示されます。



ステップ 9. [データプレビュー（Data Preview）] をクリックして、対応するチャンクの格納ファイル、そのドキュメント送信元、ページ番号、および対応するベクトル埋め込みを表示します。



インターフェイスは、ベクトル検索のさまざまなオプションを証明しました。



サイジングに関する考慮事項

主な用語

LLM 推論に関する主な用語は次のとおりです。

- バッチ サイズ：バッチ サイズは、1 回の推論実行で一緒に処理されるサンプルの数を指します。バッチ サイズは、推論プロセスの遅延とスループットに大きな影響を与える可能性があります。たとえば、バッチ サイズを増やすとスループットは向上しますが、遅延が増加します。
- [プレシジョン（Precision）]：モデルの各パラメータに使用されるサイズ（バイト単位）。
- [コンテキスト サイズ（Context Size）]：モデルがプロンプト+ 応答のためにプロセスするトークンの最大数を表します。（入力トークン長 + 出力トークン長）
- キーと値のキャッシュ（KV キャッシュ）：モデルのディメンションとレイヤに基づく単一のトークンによって消費されるメモリの量。
- アクティベーション：トークンがモデル内で処理されている場合、それはアクティベーションと呼ばれます。フルアクティベーション メモリは、KV キャッシュ サイズとコンテキスト サイズから計算されます。

LLM 推論フェーズ

LLM は、次の 2 段階のプロセスでテキストを生成します。

- 事前入力：最初のフェーズでは、モデルはプロンプト トークンを並行して取り込み、キー値（KV）キャッシュに入力します。事前入力は、将来の復号化用のキーと値のキャッシュ（KV キャッシュ）を生成します。
- 復号フェーズ：2 番目のフェーズでは、現在の状態（KVキャッシュに保存されている）を利用して、次のトークンをサンプリングして復号化します。新しいトークンごとにキャッシュを再計算しないように、ストレージに少額の料金を支払います。KV キャッシュがないと、以前に確認されたすべてのトークンをモデルに渡す必要があるため、後続のすべてのトークンのサンプリングに時間がかかります。

ユーザーからの入力シーケンスは、モデルのトレーニング データがトークン化されたのと同じ方法で最初にトークン化されます。次に、トークンがトレーニング済みモデルにフィードされます。

モデル実行の最初のステップとして、入力シーケンスは、トレーニングされたモデルの埋め込み層で埋め込みベクトルに変換されます。このベクトルは、基本的にトークンを、類似するトークンが類似するベクトルを持つ高次元空間に変換します。

トークンの埋め込みを使用して、各トークンのクエリ、キー、および値は、線形プロジェクションと呼ばれるプロセスによって計算されます。ここでは、各トークンの埋め込みベクトルに、トレーニング中に学習された個別の重み行列が乗算され、クエリ、キー、および値ベクトルが生成されます。

クエリ、キー、および値を使用するという概念は、データベースの仕組みに直接影響を受けています。各データベース ストレージには、キーによってインデックスが付けられたデータ値があり、ユーザーはクエリを作成し、キー値がクエリと一致するかどうかを比較することでデータを取得できクエリ。LLM の場合、モデルはクエリ自体を生成します。キー値はクエリと直接比較されませんが、互換性関数を使用してクエリに対する各キーの関連性が計算され、重みベクトルが生成されます。

アテンション出力は、以前に計算された重みベクトルを使用して「値」の重み付き合計としてクエリごとに計算されます。

このアテンション出力は、予測（または復号化）レイヤを通過します。このレイヤは、モデルのボキャブラリ全体の各トークンに確率を割り当て、そのトークンが次のトークンである可能性を示します。

予測された確率流通からサンプリングして、次のトークンを確率的に選択するか、または最も高い確率のトークンを予測された次のトークンとして選択できます。

クラスタ サイズ

この CVD では、4 つ以上のノードを持つ単一の GPT-in-a-Box クラスタの設計について説明します。スケーリング係数の 1 つ（VM、Kubernetes アプリケーション、GPU ワークロードの総数など）がソリューションに指定された最大数を超えている場合は、必要なキャパシティ（CPU、メモリ、ストレージ、GPU）でクラスタを拡張します。最大クラスタ サイズに達したら、さらなる成長の需要に対応するために新しいクラスタを構築することを検討してください。

インフラストラクチャの考慮事項

生成人工知能推論のインフラストラクチャのサイジングの主な要因は次のとおりです。

モデル仕様

- モデル アーキテクチャ：レイヤの数、アテンション ヘッド、パラメータなど、言語モデルのアーキテクチャを理解します。
- トークン埋め込みサイズ：埋め込みサイズが大きいと、メモリ要件に大きく影響する可能性があります。

ハードウェア アクセラレーション

- 混合精度：ハードウェア機能を効率的に活用するための混合精度のトレーニングと推論について説明します。

メモリ要件

- モデル サイズ：大規模な言語モデルでは、大量のメモリ要件が発生する可能性があります。モデルと入力シーケンスの両方に十分な GPUメモリを確保します。
- [シーケンス長（Sequence Length）]：モデルが処理できる最大シーケンス長と、メモリ使用量への影響を考慮します。

バッチ処理と並列化

- バッチ サイズ：バッチ サイズを試します。バッチ サイズを大きくするとスループットが向上しますが、メモリ要件が増加する可能性があります。
- データの並列化：必要に応じて、データのパラレリズム化を導入して推論を複数のデバイスに分散します。

遅延とスループット

- 遅延要件：言語モデルには、特にインタラクティブなアプリケーションでは、多くの場合、リアルタイムの制約があります。ユースケースに基づいて遅延を最小化します。
- [スループット ターゲット（Throughput Targets）]：1 秒あたりの処理済みトークンまたはシーケンスの観点から必要なスループットを決定します。

拡張性

- モデルの並列性：モデルのサイズが使用可能な GPUメモリを超えた場合にモデルの並列性を考慮し、モデルの一部を複数の GPU に分散します。
- インフラストラクチャのスケーリング：需要の増加に対応するための水平方向の拡張性を設計します。

冗長性とハイ アベイラビリティ

- チェックポイント：トレーニングの進行状況を失うことなく障害から回復するために、通常のモデル チェックポイントを実装します。
- レプリケーション：冗長システムを活用して、推論中の高可用性を確保します。

ネットワークに関する考慮事項

- 帯域幅：デバイス間で大きなモデル パラメータを転送するために必要なネットワーク帯域幅を評価します。
- デバイス間通信：デバイス間の通信パターンを最適化して、遅延を最小限に抑えます。

ストレージ要件

- モデル ストレージ：大きなモデル パラメータを効率的にロードできるストレージ ソリューションを選択します。
- データ ストレージ：入力データと中間結果のストレージ ニーズを評価します。

コンテナ化とオーケストレーション

- コンテナ化：コンテナ内に言語モデルを展開して、さまざまな環境での管理と一貫性を容易にします。
- オーケストレーション：Kubernetes などのコンテナ オーケストレーション ツールを活用して、展開を効率的に管理およびスケーリングします。

ミドルウェアとサービス フレームワーク

- [サービス フレームワーク（Serving Framework）]：TensorFlow Serving、Triton Inference Server など、大規模な言語モデルの展開に最適化されたサービス フレームワークワークを選択します。
- [ミドルウェア（Middleware）]：クライアントと展開されたモデル間の通信を処理するためのミドルウェアを導入し、アプリケーションの要件との互換性を確保します。

モニタリングおよび最適化

- リソース モニタリング：モニタリング ツールを使用して、GPU 使用率、メモリ使用率、およびその他の関連メトリックをトラックします。
- 動的最適化：リアルタイムの評価指標に基づいてパラメータを動的に最適化します。

セキュリティ

- データ保護：特に機密情報が含まれる場合は、入出力データを保護するための対策を導入します。
- モデル セキュリティ：大規模な言語モデルを敵対的な攻撃や不正アクセスから保護します。

このソリューションの GPU を選択する場合、考慮すべき重要な要素がいくつかあります。

- 処理能力：LLM 推論に必要な複雑なニューラル ネットワーク計算を処理するのに十分な計算能力を備えた GPU を探します。
- メモリ キャパシティ：LLM モデルには、多くの場合、大量のメモリ要件があります。モデル サイズとバッチ サイズに対応するのに十分な VRAM（ビデオRAM）が GPU にあることを確認します。
- Tensor コア：Tensor コアは、LLM 推論に不可欠なマトリックス演算を高速化します。Tensor コアを備えた GPU（NVIDIA の RTX シリーズなど）は、パフォーマンスを大幅に向上させることができます。
- 互換性：GPU が Cisco UCS および使用する予定のディープ ラーニング フレームワークと互換性があるかどうかを確認します（TensorFlow、PyTorch など）。
- 消費電力：GPU の消費電力を考慮し、システムの電源のキャパシティと一致していることを確認します。

LLM 推論のメモリ計算

必要な合計メモリは、モデル メモリ サイズと KV キャッシュの合計です。

必要な合計メモリの計算は次のとおりです。

モデル メモリ サイズ = モデル パラメータ * 精度
KV キャッシュ サイズ = 2 x バッチ サイズ x コンテキスト サイズ x レイヤ数 x モデル ディメンション x 精度
合計メモリ要件（GB） = モデル メモリ サイズ（GB）+ KV キャッシュ サイズ（GB）

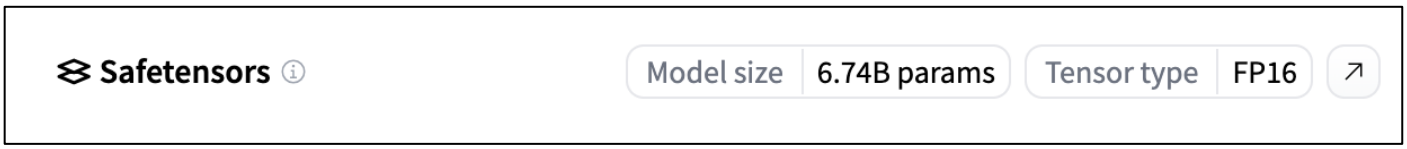
一部のモデルでは、モデル ディメンション データを使用できない場合があります。その場合、モデル ディメンションは次のように計算できます。

モデル ディメンション = アテンション ヘッドのサイズ x アテンション ヘッドの数

モデル パラメータ、精度、レイヤ数、モデル ディメンションはモデルに固有であり、モデルのモデル カードにあります。

コンテキストサイズとバッチ サイズはユーザーからの入力です。

Llama 2 のメモリ計算の例を示します。



Llama 2 モデルの場合：

合計モデル パラメータ：　　6.74B パラメータ。
精度：　　FP16。（2 バイト）
レイヤ数：　　32
モデル ディメンション：　　4096

したがって、モデル メモリは次のように計算されます。

モデル メモリ サイズ = モデル パラメータ * 精度

```
Llama 2 のモデル メモリ サイズ = 6,740,000,000 * 2 バイト/パラメータ
                    = 13,480,000,000 バイト
                          = 13.48 ギガバイト
```

また、最大入力トークン長が 1024、最大出力トークン長が 1024、バッチサイズが 8 の例を考慮した場合、KV キャッシュ サイズの計算は次のとおりです。

```
KV キャッシュ サイズ = 2 x バッチ サイズ x コンテキスト サイズ x レイヤ数 x モデル ディメンション x 精度
KV キャッシュ サイズ = 2 x 8 x (1024+1024) x 32 x 4096 x 2 バイト/パラメータ
               = 8,589,934,592 バイト
               = 8.59 ギガバイト
```

したがって、Llama2 の最大入力トークン長が 1024、最大出力トークン長が 1024、バッチ サイズが 8 の場合、必要な合計メモリは次のとおりです。

```
合計メモリ要件（GB） = モデル メモリ サイズ（GB）+ KV キャッシュ サイズ（GB）
                          = 13.48 + 8.59 ギガバイト
```

```
= 22.07 ギガバイト
```

パフォーマンスの計算

モデルでパフォーマンス ベンチマークを実行可能

パフォーマンス要件、ユーザー数、入出力トークンの数、必要な遅延とスループットに基づいて、適切な大規模言語モデル、推論バックエンド、GPU、およびコンピューティング インフラストラクチャを選択できます。

モデルのパフォーマンスは、事前入力フェーズと復号化フェーズによって異なります。これら 2 つのフェーズは、LLM のパフォーマンスに異なる影響を与えます。事前入力フェーズは、小さなバッチ サイズで GPU コンピューティングを効果的に飽和させますが、復号化フェーズでは、リクエスト、要求など（文脈に応じて）ごとに一度に 1 つのトークンを生成するため、コンピューティング使用率が低くなります。

事前入力フェーズはコンピューティング バウンドであり、復号化フェーズはメモリ バウンドです。そのため、次の要因を考慮して測定する必要があります。

- 事前入力の遅延
- 事前入力スループット
- デコード総遅延
- デコード トークン遅延
- デコード スループット

パフォーマンス ベンチマークは、さまざまなサイズ（1、2、4、8、10、25、250、100 など）で実行できます。また、2 つの異なるモデル間のパフォーマンス比較に焦点を当てた個別のテストを実行できます。

まとめ

この検証済みのソリューションは、実際の企業環境での生成人工知能アプリケーションの導入の複雑さをナビゲートするための貴重なリソースであり、特に Retrieval Augmented Generation に焦点を当てています。

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box 向け Cisco 検証済み設計は、生成人工知能向けの革新的で回数変更可能かつセキュアな生成事前トレーニング済みトランスフォーマ（GPT）ソリューションを導入するための基本的な基準アーキテクチャを提供し、組織が選択した人工知能大規模言語モデル（LLM）と、それが活用するアプリケーションを提供します。

このソリューションは、Cisco UCS と Nutanix を組み合わせることで、オンプレミス ソリューションとして迅速に導入できる人工知能推論のターンキー ソリューションを実現します。このインフラストラクチャは、2x NVIDIA L40S GPU で構成された Cisco UCS マネージド HCIAF240C M7 All-NVMe サーバを使用して設計されており、GPU 対応サーバ ノードをサポートするソフトウェア デファインド Nutanix クラウド インフラストラクチャを活用して、Kubernetes でオーケストレーションされたコンテナと Nutanix ユニファイド ストレージをサポートする仮想化コンピューティング、ストレージ、ネットワークのシームレスなスケーリングを実現します。

付録

この付録の内容は次のとおりです。

- 付録 A：部品表
- 付録 B：このガイドで使用されている参考資料

付録 A：部品表

表 16 に、このドキュメントで説明されているこのソリューション設計、導入、および検証で使用される部品表のリストを示します。

表 16 Bill of Materials（部品表）

Line Number	製品番号	説明	数量
1.0	HCI-M7-MLB	Nutanix MLB を使用したシスコ コンピューティング ハイパーコンバージド M7	1
1.1	HCIAF240C-M7SN	Cisco Compute Hyperconverged HCIAF240cM7 All Flash NVMe Node	4
1.1.0.1	CON-L1NCO-HCIAFM7C	CX LEVEL 1 8X7XNCDOS Cisco Compute Hyperconverged HCIAF240cM	4
1.1.1	HCI-FI-MANAGED	FI によって管理されるサーバーの展開モード	4
1.1.2	HCI-GPUAD-C240M7	C240M7 用 GPU 空気ダクト	4
1.1.3	HCI-NVME4-3840	3.8TB 2.5in U.2 15mm P5520 Hg Perf Med End NVMe	24
1.1.4	HCI-M2-240G	240GB M.2 SATA Micron G2 SSD	8
1.1.5	HCI-M2-HWRAID	Cisco ブート最適化 M.2 Raid コントローラ	4
1.1.6	HCI-RAIL-M7	C220 および C240 M7 ラック サーバ用ボール ベアリング レール キット	4
1.1.7	HCI-TPM-OPT-OUT	OPT OUT、TPM 2.0、TCG、FIPS140-2、CC EAL4 + 認定	4
1.1.8	HCI-AOSAHV-67-SWK9	HCI AOS AHV 6.7 SW	4
1.1.9	UCSC-HSLP-C220M7	UCS C220 M7 ヒートシンク（C240 GPU ヒートシンク用）	8
1.1.10	UCSC-BBLKD-M7	UCS C シリーズ M7 SFF ドライブブランクパネル	72
1.1.11	UCSC-M2EXT-240-D	C240M7 2U M.2 エクステンダ ボード	4
1.1.12	UCSC-RISAB-24XM7	UCS C シリーズ M7 2U エアー ブロッカー GPU のみ	4

1.1.13	CBL-G5GPU-C240M7	C240M7 PCIe CEM 準拠 12VHPWR 電源ケーブル（最大 450 W）	8
1.1.14	HCI-CPU-I6442Y	Intel I6442Y 2.6GHz/225W 24C/60MB DDR5 4800MT/s	8
1.1.15	HCI-MRX32G1RE1	32GB DDR5-4800 RDIMM 1Rx4 (16Gb)	128
1.1.16	HCI-RIS1C-24XM7	UCS C シリーズ M7 2U ライザー 1C PCIe 第 5 世代（2x16）	4
1.1.17	HCI-RIS2C-24XM7	UCS C シリーズ M7 2U ライザー 2C PCIe 第 5 世代（2x16）（CPU2）	4
1.1.18	HCI-RIS3C-24XM7	C240 M7 ライザー 3C	4
1.1.19	HCI-MLOM	Cisco VIC 接続	4
1.1.20	HCI-M-V5D200G	Cisco VIC 15238 2x 40/100/200G mLOM C シリーズ	4
1.1.21	HCI-GPU-L40S	NVIDIA L40S：350W、48GB、2 スロット FHFL GPU	4
1.1.22	HCI-NV-GRID-OPTOUT	NVIDIA GRID SW のオプトアウト	4
1.1.23	HCI-GPU-L40S	NVIDIA L40S：350W、48GB、2 スロット FHFL GPU	4
1.1.24	HCI-NV-GRID-OPTOUT	NVIDIA GRID SW のオプトアウト	4
1.1.25	HCI-PSU1-2300W	ラック サーバー チタン 用 Cisco UCS 2300W AC 電源	8
1.1.26	CAB-C19-CBN	キャビネット ジャンパ電源コード、250 VAC 16 A、C20-C19 コネクタ	8
1.2	HCI-FI-6536	Cisco Compute Hyperconverged ファブリック インターコネクト 6536	2
1.2.0.1	CON-L1NCO-HCIFI6BU	CX LEVEL 1 8X7XNCDOS Cisco Compute Hyperconverged Fabric Int	2
1.2.1	HCI-UCSM-MODE	FI の UCSM 展開モード	2
1.2.2	N10-MGT018	UCS Manager v4.2 および Intersight 管理モード v4.2	2
1.2.3	HCI-FI-6500-SW	6500 シリーズ ファブリック インターコネクトの永久ソフトウェア ライセンス。	2
1.2.4	HCI-PSU-6536-AC	UCS 6536 電源/ AC 1100W PSU：ポート側排気	4
1.2.5	CAB-9K12A-NA	電源コード、125 VAC、13 A、NEMA 5-15 プラグ（北米）	4

1.2.6	UCS-ACC-6536	UCS 6536 シャーシ アクセサリ キット	2
1.2.7	UCS-FAN-6536	UCS 6536 ファンモジュール	12

付録 B：このガイドで使用されている参考資料

Cisco Compute Hyperconverged with Nutanix の展開に関する展開およびフィールドガイド

https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/UCS_CVDs/CCHC_Nutanix_ISM.html

https://community.cisco.com/t5/unified-computing-system-knowledge-base/cisco-compute-hyperconverged-with-nutanix-field-guide/ta-p/4982563?attachment-id=228884

リポジトリ（Repository）

ソリューションの GitHub リポジトリ：https://github.com/ucs-compute-solutions/nai-llm-fleet-infra

GPT-in-a-Box 基盤

https://opendocs.nutanix.com/gpt-in-a-box/overview/

Cisco Compute Hyperconverged with Nutanix

https://www.cisco.com/c/en/us/products/hyperconverged-infrastructure/compute-hyperconverged/index.html

Cisco Intersight

https://www.cisco.com/c/en/us/products/servers-unified-computing/intersight/index.html

HCIAF240C M7 All-NVMe/All-Flash Server

https://www.cisco.com/c/dam/en/us/products/collateral/hyperconverged-infrastructure/hc-240m7-specsheet.pdf

Nutanix リファレンス ドキュメント

https://portal.nutanix.com/

著者について

Cisco Systems, Inc.、テクニカル マーケティング エンジニア、Anil Dhiman

Anil Dhiman は、Cisco UCS サーバー上のデータセンター ソリューションと、大規模企業アプリケーションのパフォーマンス エンジニアリングを専門とする 20 年以上の経験があります。Anil は、過去 14 年にわたって、Cisco のデータセンター テクノロジーに関する企業ソリューション向けの Cisco Validated Design をいくつか作成してきました。現在、Cisco のハイパーコンバージド インフラストラクチャおよびデータ保護ソリューションのポートフォリオに焦点を当てています。

Cisco Systems, Inc.、テクニカル マーケティング エンジニア、Paniraja Koppa

Paniraja Koppa は、 Cisco Unified Computing System （Cisco UCS）ソリューション チームのメンバーです。データセンターでのソリューションの設計、実装、および運用において 15 年以上の経験があります。現在のロールでは、コンピューティングおよびハイブリッドクラウドソリューションの設計と開発、ベスト プラクティス、最適化、自動化、および技術格納ファイルの作成に取り組んでいます。また、データセンター仮想化の分野でテクニカル コンサルティング エンジニアを務めていました。Paniraja は、コンピュータ サイエンスの修士号を取得しています。国際会議でいくつかの論文を発表し、 Cisco Live US and Europe、Open Infrastructure サミット、その他のパートナーイベントなどのイベントでスピーカーを務めています。Paniraja は現在、生成人工知能ソリューションに焦点を当てています。

Nutanix, Inc.、クラウド ネイティブおよび人工知能シニア スタッフ ソリューション アーキテクト、Wolfgang Huse

Wolfgang Huse は Nutanix のソリューションおよびパフォーマンス設計・導入チームのクラウド ネイティブのテクニカル リーダーであり、ドイツを拠点にしています。このロールでは、主にクラウド ネイティブ ソリューションの開発と、Nutanix Cloud Platform 内でクラウド ネイティブ アーキテクチャを採用する利点の普及を担当しています。Wolfgang と彼のチームは、RedHat OpenShift、Rancher、最新の Nutanix GPT in-a-Box などのソリューションをサポートするために、さまざまな Nutanix テクニカル ノート、ベストプラクティスガイド、検証済みの設計など、多くの主要なソリューション アーティファクトの構築に貢献してきました。このロールの前は、Nutanix でセールス エンジニアリングの役割を担い、複雑なソリューションの設計と実装をグローバルに直接支援していました。

Nutanix, Inc. クラウド ネイティブおよび人工知能スタッフ ソリューション アーキテクト、Jesse Gonzalez

Jesse Gonzalez は、Nutanix のソリューションおよびパフォーマンス設計・導入チームのクラウド ネイティブ ソリューション アーキテクトです。IT 分野で 20 年以上の経験を持つ Jesse は、あらゆる規模の多くの組織と緊密に連携して、 権限プラットフォームでクラウドネイティブ（および最近では生成人工知能）ソリューションを実現するという課題を克服してきました。このロールの前は、Nutanix でサービスとセールス エンジニアリングの両方の役割を担当していました。

謝辞

この Cisco 検証済みデザインの設計、検証、作成に関する支援および貢献に対し、下記の諸兄諸姉に感謝の念を申し上げます。

- Cisco Systems, Inc.、シニア ダイレクタ、Chris O’Brien
- Cisco Systems, Inc. プロダクト マネージャ John McAbel

フィードバック

このガイドおよび関連ガイドに関するコメントおよび提案については、https://cs.co/en-cvds のシスココミュニティのディスカッションに参加してください。

CVD プログラム

このドキュメントに記載されているデザイン、仕様、表現、情報、及び推奨事項（総称して「デザイン」）は、障害も含めて本マニュアル作成時点のものです。シスコ及びそのサプライヤは、商品性の保証、特定目的への準拠の保証、及び権利を侵害しないことに関する保証、あるいは取引過程、使用、取引慣行によって発生する保証をはじめとする、一切の保証責任を負わないものとします。いかなる場合においても、シスコ及びそのサプライヤは、このデザインを使用すること、または使用できないことによって発生する利益の損失やデータの損傷をはじめとする、間接的、派生的、偶発的、あるいは特殊な損害について、あらゆる可能性がシスコまたはそのサプライヤに知らされていたとしても、それらに対する責任を一切負わないものとします。

デザインは予告なしに変更されることがあります。このマニュアルに記載されているデザインの使用は、すべてユーザー側の責任になります。これらのデザインは、シスコ、シスコのサプライヤ、またはシスコのパートナーからの技術的な助言や他の専門的な助言に相当するものではありません。ユーザーは、デザインを実装する前に技術アドバイザに相談してください。シスコによるテストの対象外となった要因によって、結果が異なることがあります。

米国本社
Cisco Systems, Inc.
カリフォルニア州サンノゼ

アジア太平洋本社
Cisco Systems (USA), Pte. Ltd.
シンガポール

ヨーロッパ本社
Cisco Systems International BV
Amsterdam, The Netherlands

2023 年 11 月発行

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。