

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

Published: July 2024



In partnership with:



About the Cisco Validated Design Program

The Cisco Validated Design (CVD) program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information, go to: <http://www.cisco.com/go/designzone>.

Executive Summary

Cisco Validated Designs (CVDs) consist of systems and solutions that are designed, tested, and documented to facilitate and improve customer deployments. These designs incorporate a wide range of technologies and products into a portfolio of solutions that have been developed to address the business needs of our customers.

Generative Artificial Intelligence (Generative AI) stands as a transformative force across every industry, driving innovation in multiple use cases. Despite opportunities, integrating generative AI into enterprise settings poses unique challenges. Leveraging internal data effectively is critical for On-prem AI solutions. Building the right infrastructure with appropriate computational resources is critical. Visibility and monitoring of the entire stack is important from the operations point of view.

The Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box solution takes the complexity out of adopting generative AI by providing prescriptive steps for deploying the underlying infrastructure for Nutanix GPT-in-a-Box. This solution combines Cisco® servers and SaaS operations with Nutanix software, utilizing the most popular Large Language Models (LLMs) to produce a fully validated AI-ready platform that can simplify and jumpstart your AI initiatives from the data center to the edge.

This document explains the Cisco Validated Design and Deployment for GPT-in-a-Box on Cisco Compute Hyperconverged with Nutanix to deploy on-premises Generative AI applications. This solution outlines the design that supports the deployment of an innovative, flexible, and secure generative pretrained transformer (GPT) solution for Generative AI to privately run and manage organization's choice of AI large language models (LLMs) and applications leveraging it.

The solution explains a spectrum of common use cases for Generative AI on-premises leveraging organization's internal or proprietary knowledge base with special focus on Retrieval Augmented Generation (RAG).

Solution Overview

This chapter contains the following:

- [Introduction](#)
- [Purpose of this Document](#)
- [Audience](#)
- [Solution Summary](#)

Introduction

Generative AI is reshaping industries, from dynamic marketing content to interactive virtual assistants and chatbots. However, unlocking its potential within enterprises poses challenges. It is critical to effectively use the internal data of the organization in these applications. A robust infrastructure, observability across the stack, optimized model deployment and serving, high availability, and scaling are few.

The solution highlights how enterprises can design and deploy Generative Pretrained Transformer (GPT) solution for Generative AI to privately run and manage organization's choice of AI large language models (LLMs) and applications leveraging it. The solution focusses on Retrieval Augmented Generation as the reference use case.

The hardware and software components are integrated so that customers can deploy the solution quickly and economically while eliminating many of the risks associated with researching, designing, building, and deploying similar solutions from the ground up.

Purpose of this Document

This document explains the Cisco Validated Design and Deployment details for GPT-in-a-Box solution on Cisco Compute Hyperconverged with Nutanix. The solution presented in this document will address design, reference architecture, and deployment for a predictable, scalable, secure, high-performance and cloud native solution aimed at conversational ChatGPT-like experience empowered by organization's internal documents and data on-premises ensuring operational simplicity and ease.

This validated design is just one example of a supported GPT configuration. You can design and build a GPT solution in many ways, and you can deviate from this specific configuration while still following CVD best practices.

Audience

The intended audience of this document includes IT decision makers like CTOs and CIOs, IT architects and customers who are working on or interested in design, deployment, and life cycle management of generative AI systems and applications.

Solution Summary

GPT-in-a-Box on is a new turnkey solution that includes everything needed to build AI-ready infrastructure. AI applications can be easily deployed on top of GPT-in-a-Box.

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box solution is a reference architecture that combines:

- Nutanix GPT-in-a-Box software-defined solution
- Cisco Compute Hyperconverged C-Series Servers

-
- HClAF240C M7 All-NVMe server with 2x NVIDIA L40S GPU either in UCS Managed Mode or Intersight Standalone mode
 - Nutanix Prism Central for deploying and managing solution software components
 - A range of the most popular large language models

GPT-in-a-Box solution software component includes:

- Uses NKE as Kubernetes Platform and Nutanix Unified Storage (NUS)
- Leverages NAI-LLM inference endpoint
- Integrates with Nutanix Objects Event Notification

Some highlights of the solution include the following:

- AI-ready platform to enable customers to quickly design, size, and deploy an on-prem AI solution
- Quickly deploy off-the-shelf AI applications or empower developers to build their own
- A single-cloud operating model using Nutanix Cloud Platform hyperconverged infrastructure with graphic processing units (GPUs)
- Nutanix Unified Storage (NUS) for total data management, security, privacy, and resilience
- Support for popular AI/ML frameworks
- LLM freedom of choice

Retrieval Augmented Generation: Concepts and Components

This chapter contains the following:

- [What is Generative AI?](#)
- [What is Generative AI Inferencing?](#)
- [Large Language Models](#)
- [Generative AI Workflow](#)
- [Retrieval Augmented Generation](#)

This chapter explains various concepts of Generative AI, including model development workflow, inferencing challenges and use cases.

What is Generative AI?

Generative AI is a powerful branch of artificial intelligence that holds immense potential for addressing various challenges faced by enterprises. With generative AI, users and applications can quickly generate new content based on a variety of inputs; inputs and outputs to these models can include text, images, sounds, animation, 3D models, or other types of data. Due to the versatility of generative AI models, applications leveraging them can perform multiple tasks based on available data and inputs, increasing functionality beyond just text and image generation or chat-based Q&A.

How Does Generative AI Compare to Traditional AI?

Generative AI can create new content, chat responses, designs, synthetic data, and more. Traditional AI, on the other hand, is focused on detecting patterns, making decisions, honing analytics, classifying data, and detecting fraud.

As more organizations recognize the value of using AI to create new content, they're now exploring large language models (LLMs) and other generator models. Since there are pretrained LLMs available, known as foundation models, adopting generative AI requires less upfront training compared with traditional AI models. This results in significant cost and time savings when developing, running, and maintaining AI applications in production.

While 2023 has been the year of Generative AI with the introduction of ChatGPT and models like Stable Diffusion, the technology has been in development for some time. NVIDIA and other companies have been researching and innovating in this space for years, which has helped lead us to where we are today. Examples include StyleGAN (2018), which creates realistic images of people, and GauGAN (2019), which allows you to create fingerprint-style images that instantly become realistic landscapes. NVIDIA has released an app based on this research called Canvas, and these technologies have been used broadly by ecosystem partners.

What is Generative AI Inferencing?

Generative AI inferencing refers to the process of using a trained generative AI model (large language models and non-large language models) to generate new data or content based on input or contextual cues. During inferencing, the model applies its learned knowledge to produce outputs that are not direct repetitions of the training data but are rather novel creations generated by the model.

The inferencing process is crucial for leveraging the generative capabilities of the models in practical applications. It allows users to obtain novel outputs by providing inputs or guiding the model's behavior based

on specific requirements or constraints. The generated content can be used for various creative purposes, prototyping, or as a tool for exploration in different domains.

The term "inference" in the context of generative AI is associated with generating content like:

- Text Generation
 - Storytelling: Generative models can create fictional stories, narratives, or even entire chapters of books.
 - Poetry and Prose: AI models can generate poetic verses, prose, or creative writing.
 - Dialogues: Conversational agents powered by generative models can produce human-like dialogues.
- Image Generation
 - Artistic Creations: Generative Adversarial Networks (GANs) can generate visually appealing and artistic images.
 - Style Transfer: Models can transform images into different artistic styles.
 - Face Synthesis: GANs can create realistic faces of non-existent individuals.
- Music Composition
 - Melody Generation: AI models can compose original melodies and music.
 - Genre-specific Music: Generative models can create music in specific genres, mimicking different styles.
- Code Generation
 - Source Code: AI models can generate code snippets or even entire programs based on a given task or description.
- Language Translation
 - Multilingual Text: Models like OpenAI's GPT can generate text in multiple languages.
 - Translation: AI models can translate text from one language to another while preserving context.
- Content Summarization
 - Text Summaries: Generative models can summarize large blocks of text into concise and coherent summaries.
- Content Completion
 - Sentence Completion: AI models can complete sentences or paragraphs in a way that fits the context.
 - Text Expansion: Generative models can expand on given ideas or concepts.
- Product Descriptions
 - E-commerce Descriptions: AI models can generate product descriptions for e-commerce websites.
- Scientific Writing
 - Research Abstracts: Models can generate abstracts or summaries of scientific research papers.
- Conversational Agents

- **Chatbot Responses:** AI-powered chatbots can generate responses in natural language during conversations.

Large Language Models

Generative AI is a broad category that includes models designed to generate new and original content. This content can be in various forms, such as images, text, audio, or even video. Large language models are a specific subset of generative AI designed to understand and generate human language. They are primarily focused on natural language processing tasks.

Large language models (LLMs) are a class of natural language processing models which uses deep learning methodologies to comprehend and generate human language. These models are trained in vast amounts of textual data to learn the patterns, structures, and nuances of language.

One of the notable examples of LLMs is the GPT (Generative Pre-trained Transformer) series developed by OpenAI.

Key features of large language models include:

- **Scale:** LLMs are characterized by their large number of parameters, often ranging from tens of millions to billions. The scale of these models allows them to capture complex linguistic patterns and generate diverse and contextually relevant text.
- **Pre-training:** LLMs are typically pre-trained on a massive corpus of text data before being fine-tuned for specific tasks. During pre-training, the model learns to predict the next word in a sentence or fill in missing words, which helps it acquire a broad understanding of language.
- **Transformer Architecture:** LLMs, including GPT, are built on the Transformer architecture, which enables efficient processing of sequential data. Transformers use self-attention mechanisms to capture relationships between words in a sentence, facilitating better context understanding.
- **Transfer Learning:** LLMs leverage transfer learning, where the knowledge gained during pre-training on a general language understanding task is transferred to specific tasks with minimal additional training. This approach allows these models to excel in a variety of natural language processing (NLP) applications.
- **Fine-tuning:** After pre-training, LLMs can be fine-tuned for specific tasks, such as text classification, language translation, summarization, and more. This fine-tuning process adapts the model to the nuances of the target application.
- **Diverse Applications:** Large Language Models find applications in a wide range of tasks, including but not limited to natural language understanding, text generation, sentiment analysis, machine translation, question answering, and chatbot development.

The development of Large Language Models has significantly advanced the field of natural language processing, enabling the creation of sophisticated AI systems capable of understanding, and generating human-like text across various domains. However, ethical considerations, biases in training data, and potential misuse are important considerations associated with the deployment of these models.

Model Parameters

Model parameters are the internal variables or weights that the model learns during the training process. Weights are the coefficients that scale the input features in a neural network. In the context of LLMs, these weights determine the strength of connections between neurons in different layers. For example, in a transformer model, weights are associated with the attention mechanisms and transformations applied to input sequences.

LLMs often consist of multiple layers, each with its set of weights and biases. In transformer architectures, these layers may include self-attention mechanisms and feedforward neural networks. The parameters of each layer capture different aspects of the input data.

The total number of parameters in an LLM is a critical factor in its capacity to capture complex language patterns and nuances.

Generative AI Workflow

Typical Generative AI workflow starts with aligning to the business objectives while maintaining a concise and accurate technical focus in every stage.

Business Strategy and Use Case Definition: Define generative AI objectives aligning with business goals.

- Key Tasks
 - Identify use cases.
 - Clearly define the generative task, whether it's image generation, text generation, style transfer, etc.
 - Establish goals and success metrics.

Data Preparation and Curation: Ensure high-quality, well-managed dataset availability.

- Key Tasks
 - Gather a diverse and representative dataset for training the generative model.
 - Data cleansing and labeling.
 - Data aggregation and preprocessing.
 - Increase the diversity of the training data through techniques like rotation, scaling, or flipping.
 - Anonymization or synthetic data generation if required.
 - Leveraging MLOps platforms for efficient data management.

Model Training: Utilize accelerated infrastructure for efficient training.

- Key Tasks
 - Training from scratch or selecting pretrained models.
 - Allocating heavy computational resources.
 - Optimizing performance with validated, high-performance infrastructure.

Model Customization: Fine-tuning, prompt learning (including prompt tuning and P-tuning), transfer learning, reinforcement learning.

- Key Tasks
 - Adapt pretrained models to specific business needs.
 - Implement customization methods based on requirements.

Inferencing: Deploy and operate trained models for ongoing generation.

- Key Tasks
 - Scale computing resources (scaling up or out) based on demand.
 - Iterate on inferencing based on new data and customization opportunities.

- Continuous monitoring of inferencing performance.
- Identification and optimization of opportunities for further customization and fine-tuning.

This workflow emphasizes technical aspects, highlighting the importance of infrastructure efficiency, model customization techniques, and ongoing optimization in the inferencing phase.

Retrieval Augmented Generation (RAG)

This design is mainly focused on Retrieval Augmented Generation which is an application of Generative AI in the enterprises. RAG is a class of LLM applications that use external data to augment the LLM's context.

Limitation of Large Language Models

As the usage of Large Language Models exploded, some of the prominent weakness of the system became bottleneck for utilizing it in the enterprise space. Even though these LLMs are trained on vast amount of data, the information used to generate the response is limited only to the data used during training. If you take corporate AI chatbot as example, we can't use LLMs directly because it will not have specific information of organization's internal data or services. This lack of domain specific data or organization specific data blocks it from using it for enterprise applications.

Some limitations are:

- Main one is the hallucination. While LLM can generate fluent text on various topics and domains, they are also prone to just make the stuff up. Hallucinations are outputs of LLMs that deviate from facts. It can range from minor inconsistencies or completely fabricated or contradictory statements.
- There is a knowledge cutoff date and any event that happened post that date, that information is not available in the model and LLM will not be able to provide an accurate answer for them.
- Many of the models are generic models for language tasks. They lack domain specific data.

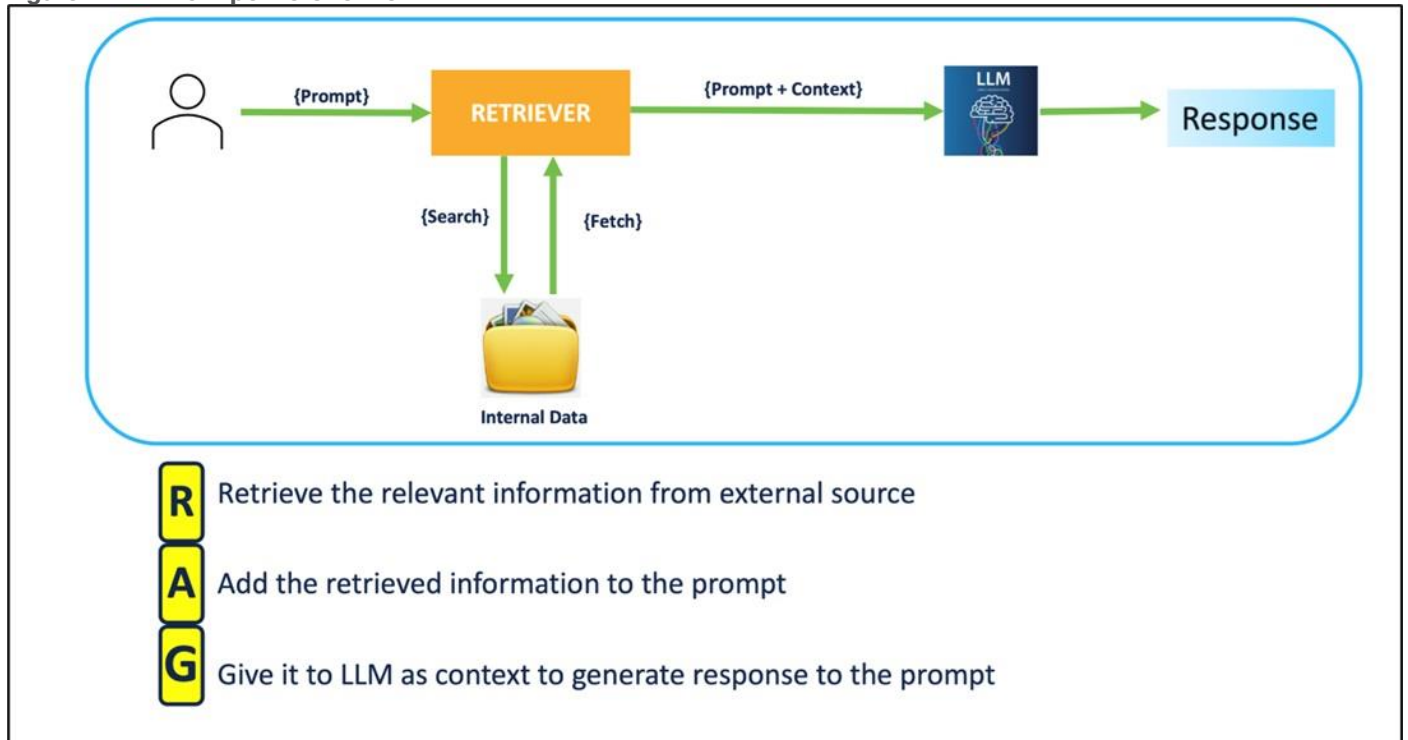
How can RAG help?

RAG generates up-to-date and domain-specific answers by connecting a LLM to enterprise data. It is architecture used to optimize the output of LLMs with dynamic domain specific data fetched from external sources.

RAG Pipeline

[Figure 1](#) illustrates the RAG Pipeline overview.

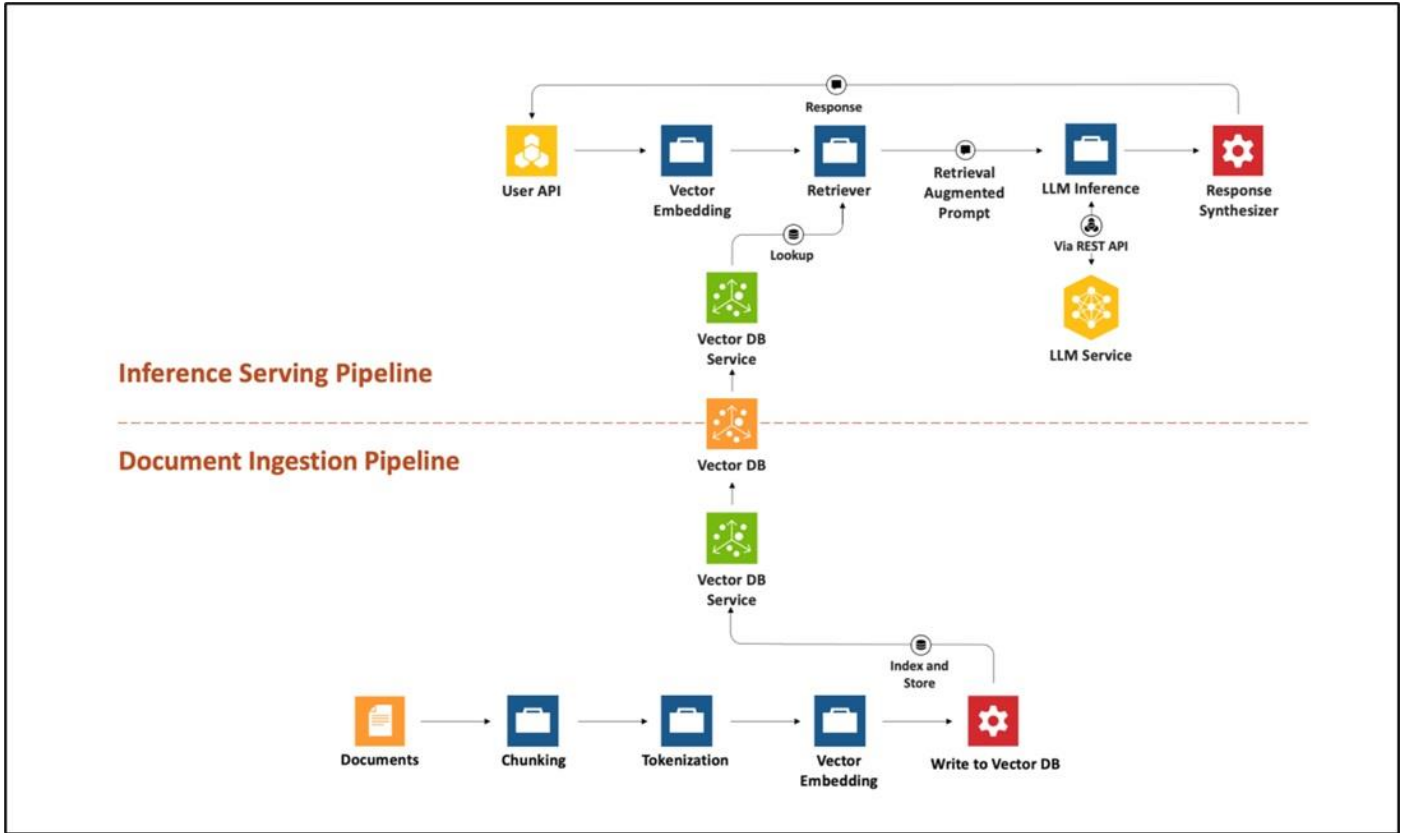
Figure 1. RAG Pipeline Overview



In this pipeline, when you enter a prompt/query, document chunks relevant to the prompt are searched and fetched to the system. The retrieved relevant information is augmented to the prompt as context. LLM is asked to generate a response to the prompt in the context and the user receives the response.

Generic RAG Architecture

RAG is an end-to-end architecture that combines information retrieval component with response generator.

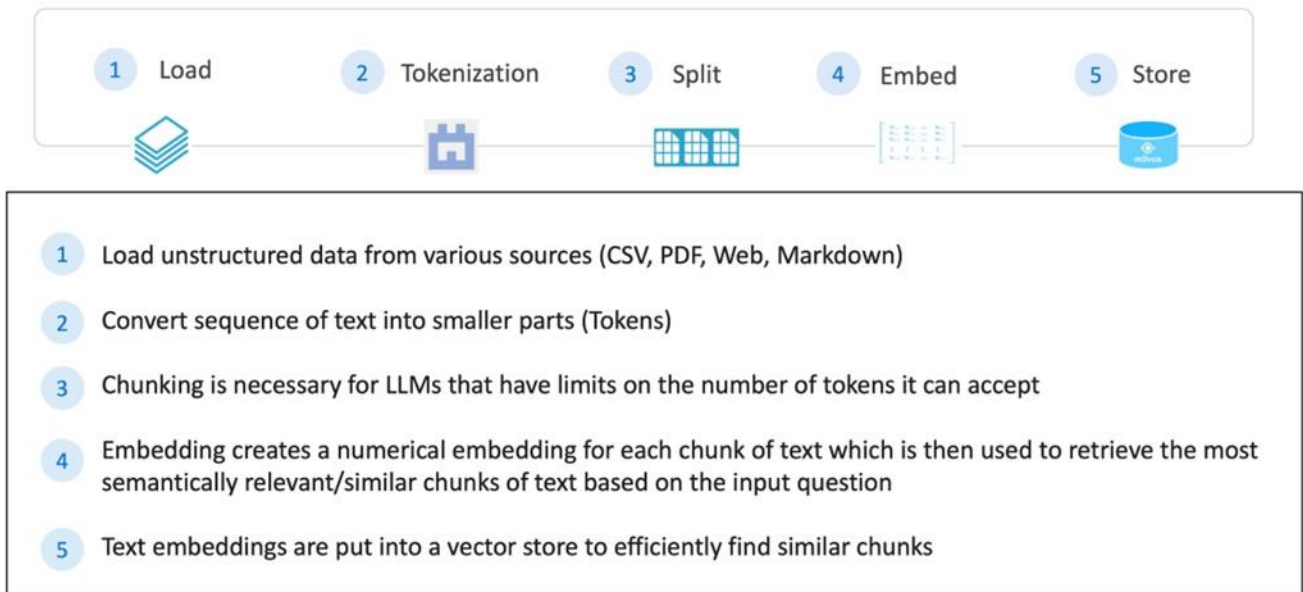


RAG can be broken into two pipelines, document ingestion pipeline and Inference serving pipeline.

Document Ingestion Pipeline

[Figure 2](#) illustrates the document ingestion pipeline.

Figure 2. Document Ingestion Pipeline Overview



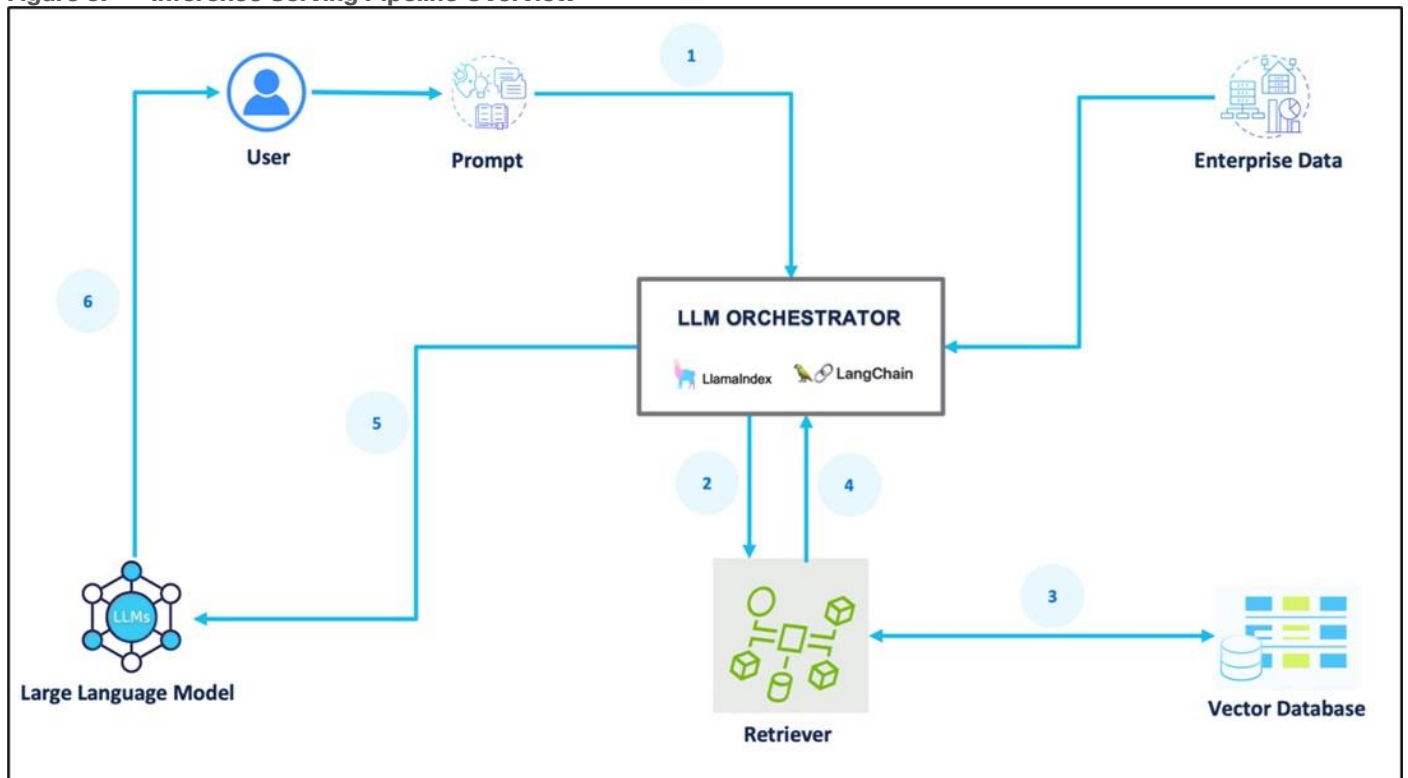
The process for the document ingestion pipeline is as follows:

1. The first step in the document ingestion pipeline is loading data. Raw data from various sources are ingested into RAG system. These data could be PDFs, word docs, HTML, even YouTube transcripts and much more.
2. Next is Tokenization, where these pieces of text are broken down into smaller units, called tokens. Tokens are the minimum unit to measure what is done with LLMs. Data needs to be transformed into a numerical representation because models learn to understand the semantic relationships between these tokens.
3. When the document is loaded, the next step is chunking, which breaks down long text into smaller segments. This is necessary since LLMs have a limit on the context window. Also, smaller chunks assist with better indexing and faster search.
4. Next is Vector embedding, which is a technique used extensively in natural language processing to represent words or phrases as vectors of numbers called embedding dimension. Vectors are designed to capture semantic relationships between words which means Words with similar meanings are represented by vectors that are close together in the embedding space.
5. Created embeddings will be stored in a special database called as vector databases. These vector database enables fast retrieval and similarity search. These ensures that the LLMs have access to the most relevant and contextually appropriate information.

Inference Serving Pipeline

[Figure 3](#) illustrates the inference serving pipeline.

Figure 3. Inference Serving Pipeline Overview



The process for the inference serving pipeline is as follows:

-
1. Prompt is passed to an LLM orchestrator.
 2. Orchestrator sends a search query to the retriever.
 3. Retriever fetches relevant information from the knowledge base.
 4. Retriever sends back the retrieved information to the orchestrator.
 5. Orchestrator augments the prompt with the context and sends it to the LLM.
 6. LLM responds with generated text, displayed to the user using the orchestrator.

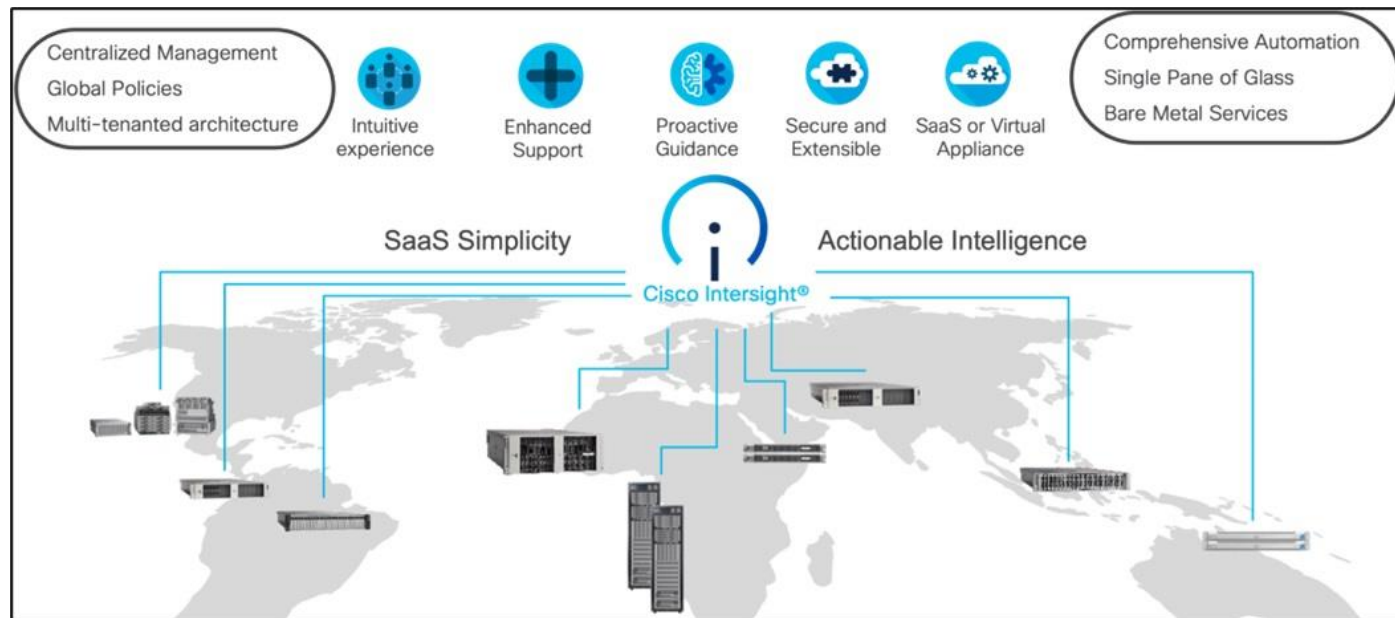
Technology Overview

This chapter contains the following:

- [Cisco Intersight Platform](#)
- [Cisco Unified Computing System](#)
- [Cisco Compute Hyperconverged with Nutanix](#)
- [NVIDIA GPUs and GPU Operator](#)

Cisco Intersight Platform

As applications and data become more distributed from core data center and edge locations to public clouds, a centralized management platform is essential. IT agility will be a struggle without a consolidated view of the infrastructure resources and centralized operations. Cisco Intersight provides a cloud-hosted, management and analytics platform for all Cisco Compute for Hyperconverged, Cisco UCS, and other supported third-party infrastructure deployed across the globe. It provides an efficient way of deploying, managing, and upgrading infrastructure in the data center, ROBO, edge, and co-location environments.



Cisco Intersight provides:

- No Impact Transition: Embedded connector (Cisco HyperFlex, Cisco UCS) will allow you to start consuming benefits without forklift upgrade.
- SaaS/Subscription Model: SaaS model provides for centralized, cloud-scale management and operations across hundreds of sites around the globe without the administrative overhead of managing the platform.
- Enhanced Support Experience: A hosted platform allows Cisco to address issues platform-wide with the experience extending into TAC supported platforms.
- Unified Management: Single pane of glass, consistent operations model, and experience for managing all systems and solutions.
- Programmability: End to end programmability with native API, SDK's and popular DevOps toolsets will enable you to deploy and manage the infrastructure quickly and easily.

- Single point of automation: Automation using Ansible, Terraform, and other tools can be done through Intersight for all systems it manages.
- Recommendation Engine: Our approach of visibility, insight and action powered by machine intelligence and analytics provide real-time recommendations with agility and scale. Embedded recommendation platform with insights sourced from across Cisco install base and tailored to each customer.

For more information, go to the Cisco Intersight product page on cisco.com.

Cisco Intersight Virtual Appliance and Private Virtual Appliance

In addition to the SaaS deployment model running on Intersight.com, you can purchase on-premises options separately. The Cisco Intersight virtual appliance and Cisco Intersight private virtual appliance are available for organizations that have additional data locality or security requirements for managing systems. The Cisco Intersight virtual appliance delivers the management features of the Cisco Intersight platform in an easy-to-deploy VMware Open Virtualization Appliance (OVA) or Microsoft Hyper-V Server virtual machine that allows you to control the system details that leave your premises. The Cisco Intersight private virtual appliance is provided in a form factor designed specifically for users who operate in disconnected (air gap) environments. The private virtual appliance requires no connection to public networks or to Cisco network.

Licensing Requirements

The Cisco Intersight platform uses a subscription-based license with multiple tiers. You can purchase a subscription duration of 1, 3, or 5 years and choose the required Cisco UCS server volume tier for the selected subscription duration. Each Cisco endpoint automatically includes a Cisco Intersight Base license at no additional cost when you access the Cisco Intersight portal and claim a device. You can purchase any of the following higher-tier Cisco Intersight licenses using the Cisco ordering tool:

- Cisco Intersight Essentials: Essentials includes all the functions of the Base license plus additional features, including Cisco UCS Central software and Cisco Integrated Management Controller (IMC) supervisor entitlement, policy-based configuration with server profiles, firmware management, and evaluation of compatibility with the Cisco Hardware Compatibility List (HCL).
- Cisco Intersight Advantage: Advantage offers all the features and functions of the Base and Essentials tiers. It also includes storage widgets and cross-domain inventory correlation across compute, storage, and virtual environments (VMware ESXi). OS installation for supported Cisco UCS platforms is also included.

Servers in the Cisco Intersight managed mode require at least the Essentials license. For more information about the features provided in the various licensing tiers, go to:

https://www.intersight.com/help/saas/getting_started/licensing_requirements

Cisco Unified Computing System

Cisco Unified Computing System (Cisco UCS) is a next-generation datacenter platform that integrates computing, networking, storage access, and virtualization resources into a cohesive system designed to reduce total cost of ownership and increase business agility. The system integrates a low-latency, lossless 10-100 Gigabit Ethernet unified network fabric with enterprise-class, x86-architecture servers. The system is an integrated, scalable, multi-chassis platform with a unified management domain for managing all resources.

Cisco Unified Computing System consists of the following subsystems:

- Compute—The compute piece of the system incorporates servers based on the Fourth Generation Intel Xeon Scalable processors. Servers are available in blade and rack form factor, managed by Cisco UCS Manager.

-
- Network—The integrated network fabric in the system provides a low-latency, lossless, 10/25/40/100 Gbps Ethernet fabric. Networks for LAN, SAN and management access are consolidated within the fabric. The unified fabric uses the innovative Single Connect technology to lower costs by reducing the number of network adapters, switches, and cables. This in turn lowers the power and cooling needs of the system.
 - Virtualization—The system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtual environments to support evolving business needs.

Cisco UCS Differentiators

Cisco Unified Computing System is revolutionizing the way servers are managed in the datacenter. The following are the unique differentiators of Cisco Unified Computing System and Cisco UCS Manager:

- Embedded Management—In Cisco UCS, the servers are managed by the embedded firmware in the Fabric Inter-connects, eliminating the need for any external physical or virtual devices to manage the servers.
- Unified Fabric—In Cisco UCS, from blade server chassis or rack servers to FI, there is a single Ethernet cable used for LAN, SAN, and management traffic. This converged I/O results in reduced cables, SFPs and adapters - reducing capital and operational expenses of the overall solution.
- Auto Discovery—By simply inserting the blade server in the chassis or connecting the rack server to the fabric interconnect, discovery and inventory of compute resources occurs automatically without any management intervention. The combination of unified fabric and auto-discovery enables the wire-once architecture of Cisco UCS, where compute capability of Cisco UCS can be extended easily while keeping the existing external connectivity to LAN, SAN, and management networks.

Cisco UCS Manager

Cisco UCS Manager (UCSM) provides unified, integrated management for all software and hardware components in Cisco UCS. Using Cisco Single Connect technology, it manages, controls, and administers multiple chassis for thousands of virtual machines. Administrators use the software to manage the entire Cisco Unified Computing System as a single logical entity through an intuitive graphical user interface (GUI), a command-line interface (CLI), or through a robust application programming interface (API).

Cisco Compute Hyperconverged with Nutanix

Cisco and Nutanix have come together to offer the industry's most simple, comprehensive HCI solution. The Cisco Compute Hyperconverged with Nutanix solution combines the Cisco Unified Computing Systems (UCS) innovative server, networking, and SaaS management with Nutanix's leading HCI foundation, offering a fully integrated and validated system with flexible deployment options and a unified, enhanced support model backed by two world-class organizations.

The solution offers the following key benefits:

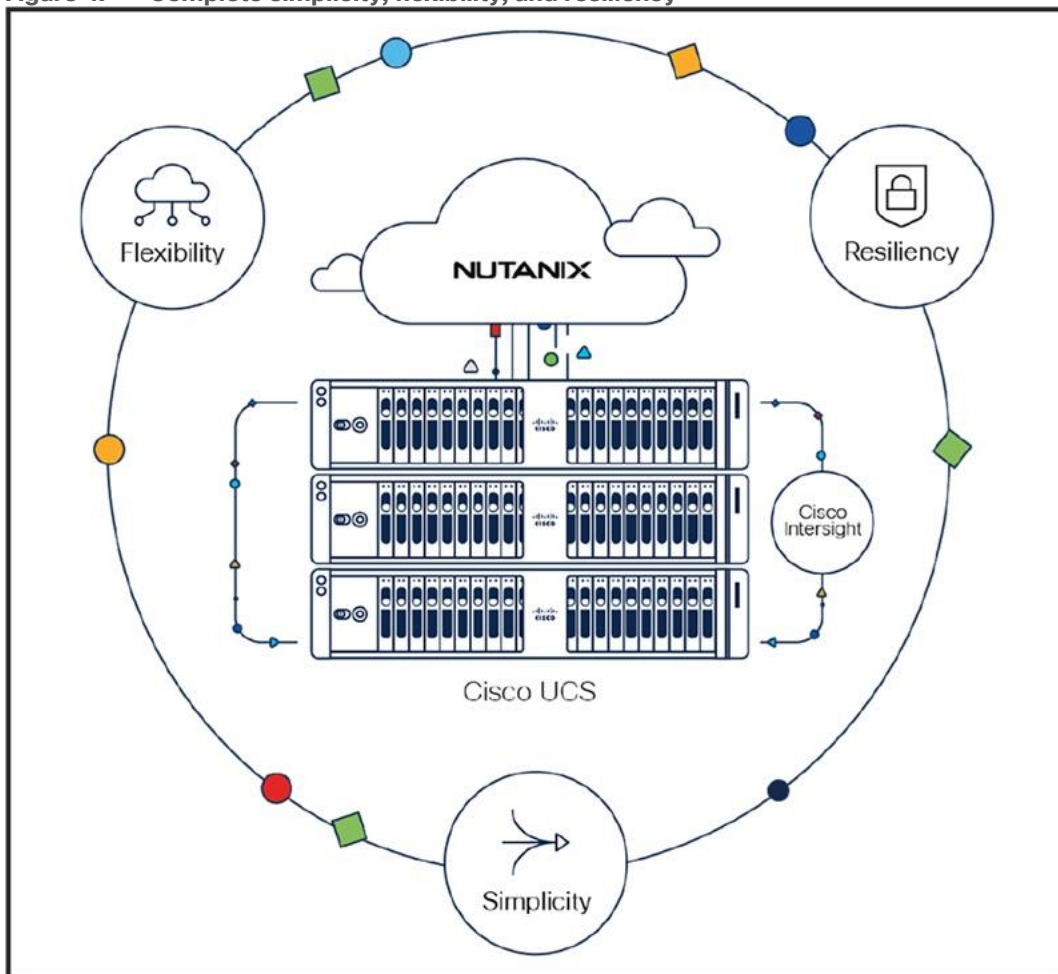
- Complete simplicity: The solution offers both SaaS and on-premises management options and includes day-0 through day-N operations, including service profiles for compute, storage, and networking customized for Nutanix to help simplify and accelerate cluster deployment and deliver better performance and resiliency. This also includes preinstalled software configured with a choice of hypervisor for a faster, easier start. Cisco and Nutanix combined and complementary cloud operating models provide control, visibility, and consistency across highly distributed IT environments, including fully integrated cluster

installation, expansion, and end-to-end software and firmware upgrades. To simplify the buying experience, the complete solution can be ordered and delivered from Cisco.

- Complete flexibility: The Cisco Compute Hyperconverged with Nutanix solution addresses modern applications and use cases, offering multiple choices in UCS server deployment options, the latest accelerator and drive technologies, and SaaS innovations from two industry powerhouses, including integrations with the leading public cloud providers. Additionally, the solution incorporates Cisco's best-in-class networking technology, including Cisco ACI integrations, to enhance performance and resiliency for data-intensive workloads in hybrid cloud environments.
- Complete resiliency: The joint solution utilizes only enterprise-grade components and offers augmented system protection with a collaborative support model and proactive, automated resilience and security capabilities. This includes integrated support systems and case notes for faster triage. Any time log files are uploaded, or case notes are generated, that information is shared, enabling enhanced collaboration among support teams to resolve issues faster and provide an improved customer experience. Our policy-based approach minimizes human error and configuration drift, resulting in consistent, reliable cluster deployments.

It also enforces overall security posture through centralized authorizations, preventing tampering of configurations.

Figure 4. Complete simplicity, flexibility, and resiliency



HCIAF240C M7 All-NVMe/All-Flash Servers

The Cisco Compute Hyperconverged HCIAF240C M7 All-NVMe/All-Flash Servers extends the capabilities of Cisco’s Compute Hyperconverged portfolio in a 2U form factor with the addition of the 4th Gen Intel® Xeon® Scalable Processors (codenamed Sapphire Rapids), 16 DIMM slots per CPU for DDR5-4800 DIMMs with DIMM capacity points up to 256GB.

The All-NVMe/all-Flash Server supports 2x 4th Gen Intel® Xeon® Scalable Processors (codenamed Sapphire Rapids) with up to 60 cores per processor. With memory up to 8TB with 32 x 256GB DDR5-4800 DIMMs, in a 2-socket configuration. There are two servers to choose from:

- HCIAF240C-M7SN with up to 24 front facing SFF NVMe SSDs (drives are direct-attach to PCIe Gen4 x2)
- HCIAF240C-M7SX with up to 24 front facing SFF SAS/SATA SSDs

For more details, go to: [HCIAF240C M7 All-NVMe/All-Flash Server specification sheet](#)

Figure 5. Front View: HCIAF240C M7 All-NVMe/All-Flash Servers



NVIDIA GPUs and GPU Operator

This solution provides a reference architecture for GPT-In-Box in the enterprises using NVIDIA L40S GPUs.

NVIDIA L40S GPU

The NVIDIA L40S GPU is the most powerful universal GPU for the data center, delivering end-to-end acceleration for the next generation of AI-enabled applications—from gen AI, LLM inference, small-model training and fine-tuning to 3D graphics, rendering, and video applications.

The L40S GPU is optimized for 24/7 enterprise data center operations and designed, built, tested, and supported by NVIDIA to ensure maximum performance, durability, and uptime. The L40S GPU meets the latest data center standards, is Network Equipment-Building System (NEBS) Level 3 ready, and features secure boot with root of trust technology, providing an additional layer of security for data centers.

Table 1. NVIDIA L40S Tensor Core GPU Specifications

	L40S PCIe GPU
GPU Architecture	NVIDIA Ada Lovelace Architecture
GPU Memory	48GB GDDR6 with ECC
GPU Memory Bandwidth	864GB/s
Interconnect Interface	PCIe Gen4 x16: 64GB/s bidirectional
NVIDIA Ada Lovelace Architecture-Based CUDA® Cores	18,176
NVIDIA Third-Generation RT Cores	142

	L40S PCIe GPU
NVIDIA Fourth-Generation Tensor Cores	568
RT Core Performance TFLOPS	209
FP32 TFLOPS	91.6
TF32 Tensor Core TFLOPS	183 366
BFLOAT16 Tensor Core TFLOPS	362.05 733
FP16 Tensor Core	362.05 733
FP8 Tensor Core	733 1,466
Peak INT8 Tensor TOPS	733 1,466
Peak INT4 Tensor TOPS	733 1,466
Form Factor	4.4" (H) x 10.5" (L), dual slot
Display Ports	4x DisplayPort 1.4a
Max Power Consumption	350W
Power Connector	16-pin
Thermal	Passive
Virtual GPU (vGPU) Software Support Yes	Yes
NVENC NVDEC	3x 3x (includes AV1 encode and decode)
Secure Boot With Root of Trust	Yes
NEBS Ready	Level 3
MIG Support	No
NVIDIA® NVLink® Support	No

NVIDIA GPU Operator

[Figure 6](#) illustrates the NVIDIA GPU Operator overview.

Figure 6. NVIDIA GPU Operator Overview



Kubernetes provides access to special hardware resources such as NVIDIA GPUs, NICs, Infiniband adapters and other devices through the device plugin framework. However, configuring and managing nodes with these hardware resources requires configuration of multiple software components such as drivers, container runtimes or other libraries which are difficult and prone to errors. The NVIDIA GPU Operator uses the operator framework within Kubernetes to automate the management of all NVIDIA software components needed to provision GPU. These components include the NVIDIA drivers (to enable CUDA), Kubernetes device plugin for GPUs, the NVIDIA Container Toolkit, automatic node labelling using GFD, DCGM based monitoring and others.

Solution Design

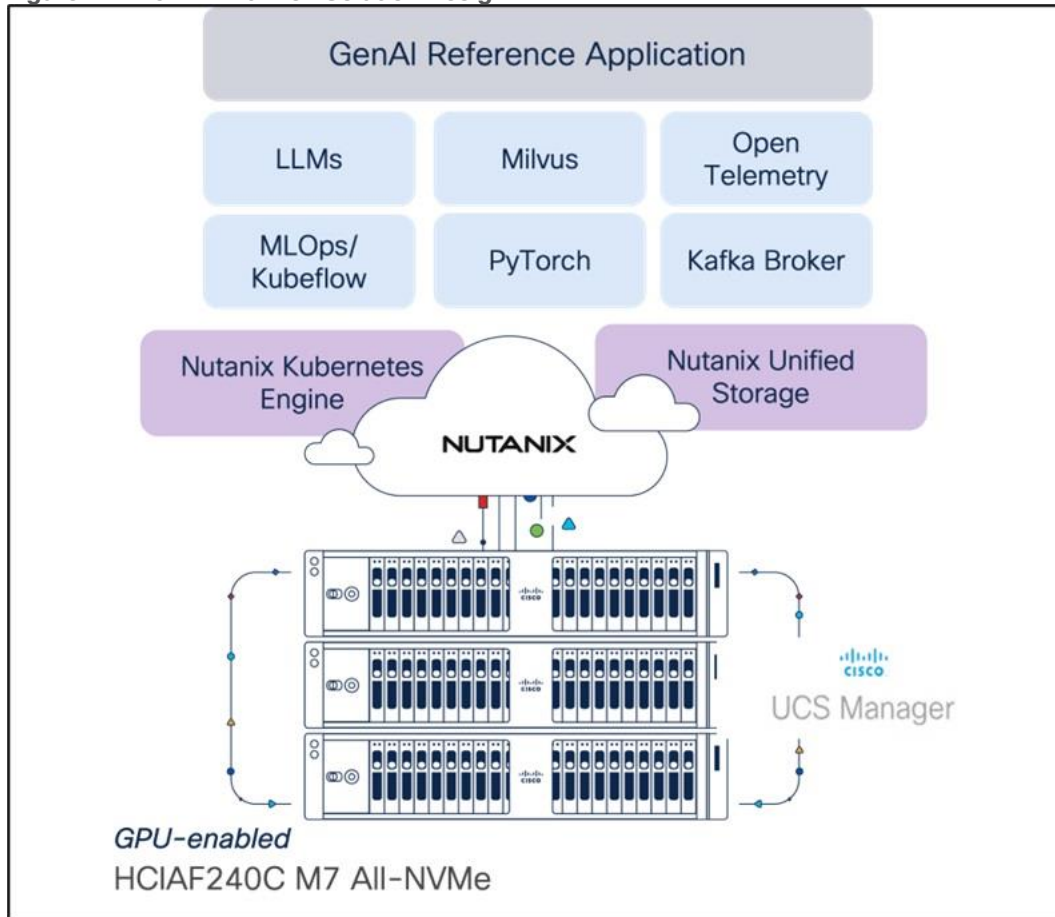
This chapter contains the following:

- [Solution Overview](#)
- [Infrastructure Design](#)
- [Network Design](#)
- [Cluster Design](#)
- [Storage Design](#)
- [Nutanix Files and Objects Design](#)
- [Management Design](#)
- [Security and Compliance](#)
- [Kubernetes Cluster Design](#)
- [Large Language Model Design](#)
- [Backup and Disaster Recovery](#)

Solution Overview

This solution provides a foundational reference architecture for AI-ready platform to enable customers to quickly design, size, and deploy an on-prem AI solution. The key design components configured in this solution are elaborated in [Figure 7](#).

Figure 7. GPT-in-a-Box Solution Design



This design includes the following hardware and software components:

- Single availability zones (AZs) in a single region in an on-premises Cisco Compute Hyperconverged (CCHC) with Nutanix cluster
- The CCHC with Nutanix (AHV) cluster with a minimum of four (4) nodes of HClAF240C M7 All-NVMe servers with each node enabled with 2x NVIDIA L40S: 350W, 48GB GPUs.
- The CCHC with Nutanix cluster host the following key Nutanix services
 - Nutanix Unified Storage (NUS) to provide NFS storage and S3-compatible storage capabilities
 - Nutanix Kubernetes Engine (NKE) cluster used for management, monitoring, and the persistent applications used by workload clusters

Note: Nutanix Prism Central was hosted in a separate Nutanix AHV Cluster. Prism Central can be deployed either on the existing cluster or on a separate AHV based Nutanix cluster.

Infrastructure Design

The deployment architecture for Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box is detailed in the figure below. The entire Day 0 deployment is managed through workflow defined in the Nutanix Foundation VM.

CCHC with Nutanix cluster for GPT-in-a-box is managed through Prism Central deployed on a separate Nutanix AHV cluster. Prism Central can also be deployed on the same Nutanix Cluster hosted GPT-in-a-Box solution.

The HCIAF240C M7 All-NVMe nodes are connected to a pair of Cisco UCS 6536 Fabric Interconnect in UCS Managed mode.

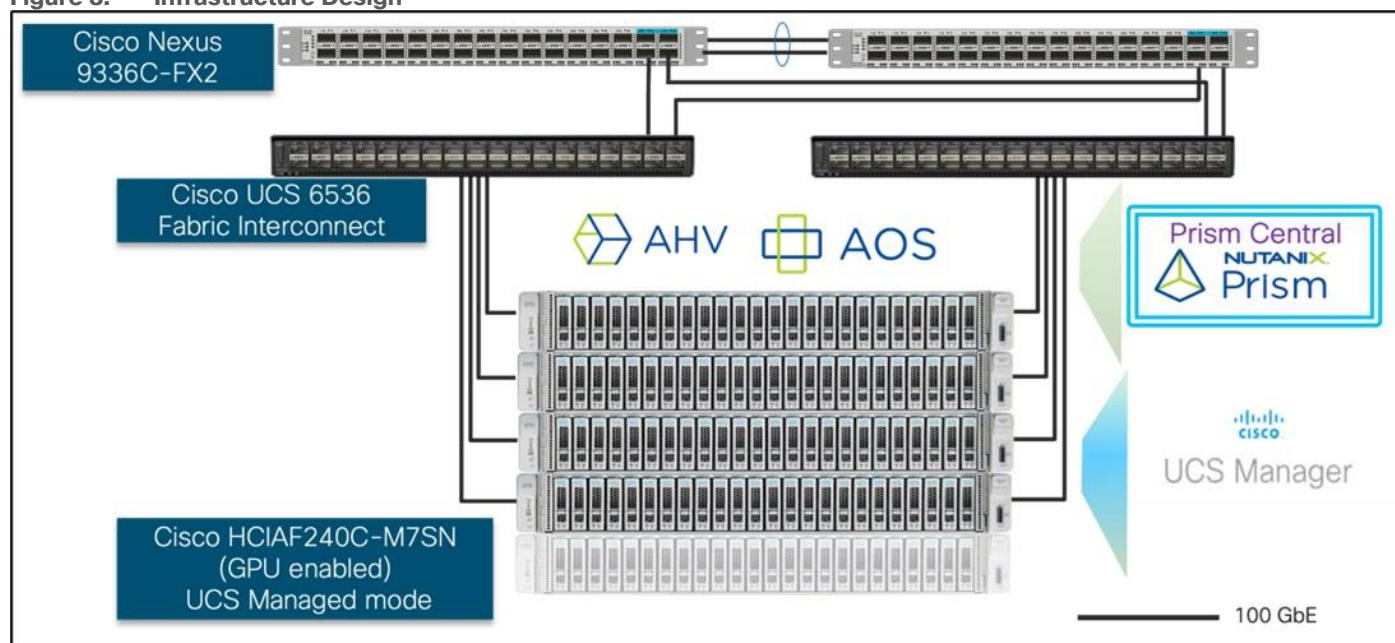
Each HCIAF240C M7 All-NVMe server is configured with:

- 2x Intel I6442Y processor (2.6GHz/225W 24C/60MB)
- 1 TB DDR5 memory (32x 32GB DDR5-4800 RDIMM)
- 2x 240GB M.2 card managed through M.2 RAID controller
- 6x 3.8 TB NVMe
- 1x Cisco VIC 15238 2x 40/100/200G mLOM
- 2x NVIDIA L40S: 350W, 48GB GPUs

Go to the [Bill of Materials \(BoM\)](#) section for the complete specifications of the infrastructure deployed to validate this solution.

[Figure 8](#) illustrates the hardware deployment architecture for Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box.

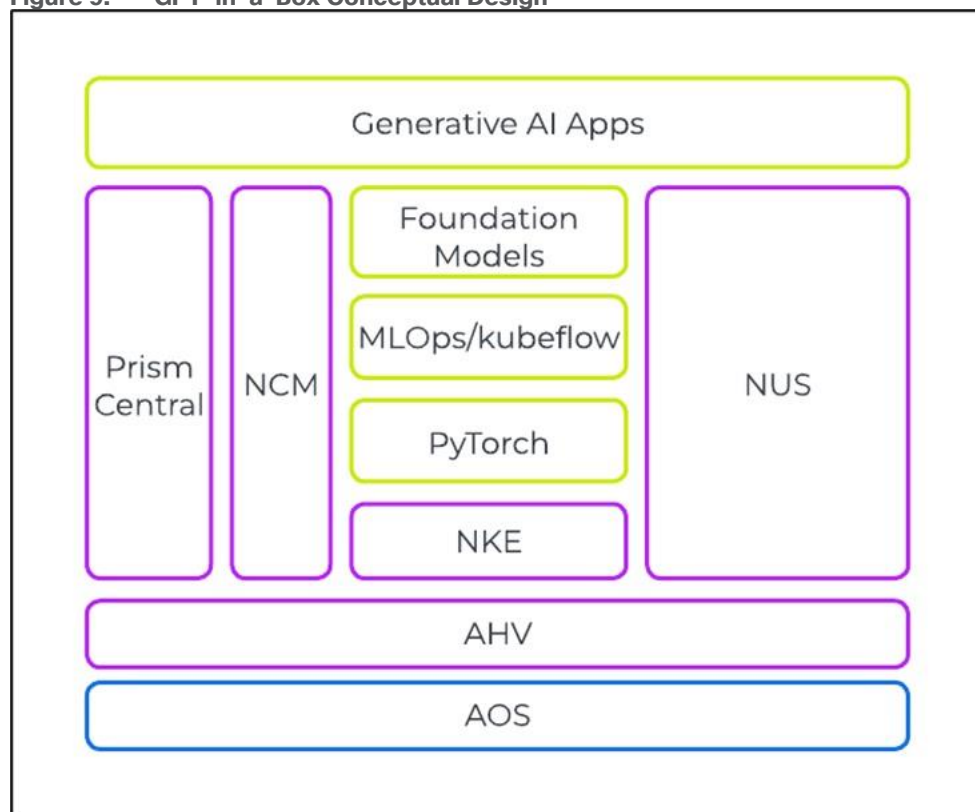
Figure 8. Infrastructure Design



The solution features a single availability zones (AZs) in a single region in an on-premises data center:

- A single Nutanix cluster that hosts the following services, among others:
 - Nutanix management components, including Prism Central
 - NUS to provide NFS storage and S3-compatible storage capabilities
 - NKE clusters used for the GPT workloads
 - NKE cluster used for management, monitoring, and the persistent applications used by workload clusters

Figure 9. GPT-in-a-Box Conceptual Design



VLAN Configuration

Table 2 lists VLANs configured for the existing solution.

Table 2. VLAN Usage

VLAN ID	Name	Usage	IP Subnet used in this deployment
2	Native-VLAN	Use VLAN 2 as native VLAN instead of default VLAN (1).	
1080	OOB-MGMT-VLAN	Out-of-band management VLAN to connect management ports for various devices	10.108.0.0/24; GW: 10.108.0.254
1081	NTNX-VLAN	VLAN utilized for Nutanix Cluster Management, Files & Objects Services, and Nutanix Kubernetes deployment	10.108.1.0/24; GW: 10.108.1.254

Table 3 lists the IP Address Assignment for the solution.

Table 3. Virtual Machines

Type	VLAN	IP Address Range	DNS Entries	Comments
UCS Management	1080	10.108.0.110-120		UCS Management and Out of Band Server Management
GPT-in-a-Box	1081	10.108.1.121-130		Nutanix Nodes AHV, CVM,

Type	VLAN	IP Address Range	DNS Entries	Comments
Nutanix Cluster				iSCSI Data Service and Cluster VIP
Prism Central	1081	10.108.1.140	prismcentral0.rtp4.local	Prism Central hosted on a separate AHV Nutanix Cluster
File Server	1081	10.108.1.141	fileserver-01.rtp4.local	Hosted on GPT-in-a-Box cluster
IPAM for object and File services	1081	10.108.1.171-180		
IPAM for Nutanix Kubernetes Services	1081	10.102.1.180-190		Configured on Prism Central
Wild card subdomain for nginx ingress to Kubernetes management cluster	1081	10.108.1.213-216	*.ntnx-k8-mgmt.rtp4.local	Entry in local DNS as Host Address record 10.108.1.213
Wild card subdomain for nginx ingress to Kubernetes workload cluster	1081	10.108.1.217-220	*.ntnx-k8-workload.rtp4.local	Entry in local DNS as Host Address record 10.108.1.21310.108.1.217
Wildcard subdomain for ILM end point	1081	10.108.1.218	*.ilm.ntnx-k8-workload.rtp4.local	Entry in local DNS as Host Address record 10.108.1.218

Software Revisions

[Table 4](#) lists the software revisions for various components of the solution.

Table 4. Software Revisions

Device	Image Bundle	Comments
Cisco UCS 6536 Fabric Interconnect	4.3(4a)	Cisco UCS GA release for infrastructure including FIs and Server Firmware
Cisco UCS HCIAF240C M7 All-NVMe server	4.3(4c)	4x HCIAF240C M7 All-NVMe nodes for GPT-in-a-Box cluster
NVIDIA L40S: 350W, 48GB GPUs	lcm_nvidia_20230302.2008_535.129.03	Download from Nutanix Portal
Nutanix AOS/AHV Cluster on UCS Managed HCIAF240C M7 All-NVMe server	6.7.1.5	
Prism Central hosted on separate Nutanix AHV	2023.4	

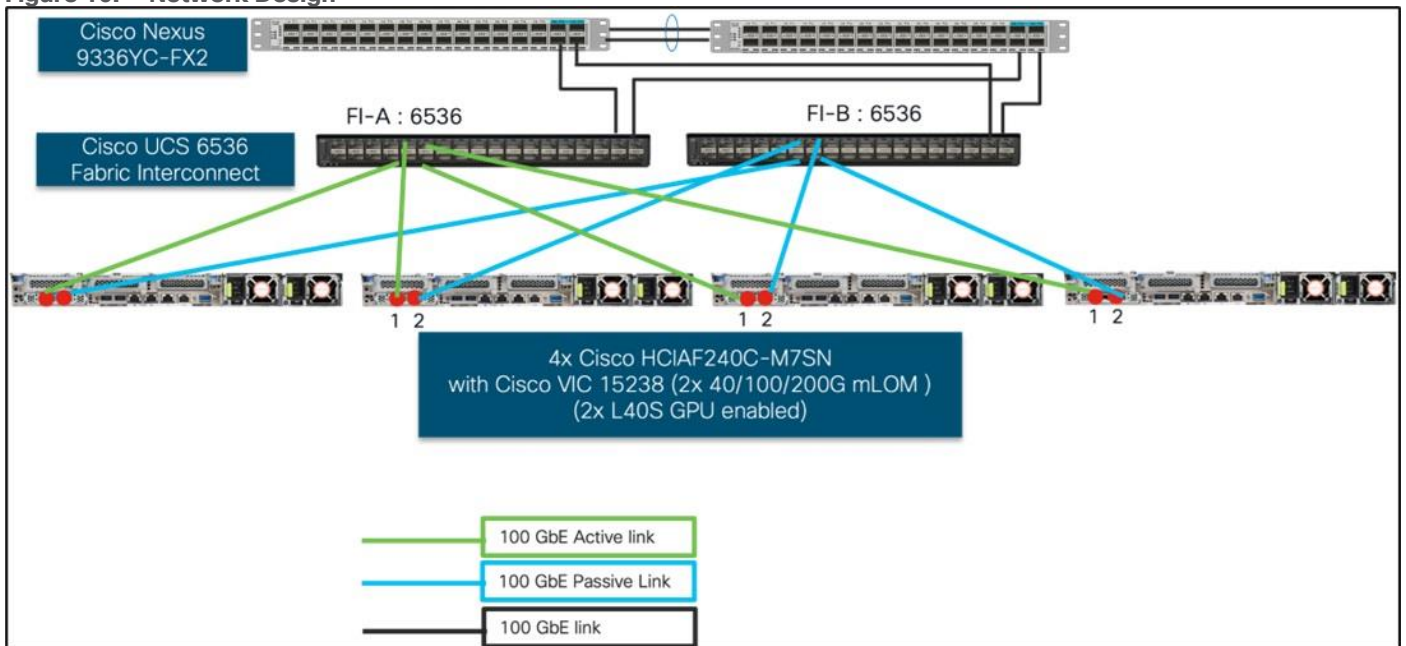
Device	Image Bundle	Comments
cluster		
NKE enabled through Nutanix marketplace and hosted on GPT-in-a-Box cluster	1.26.8-0 (OS Image - ntnx-1.6.1)	
Files enabled through Nutanix marketplace and hosted on GPT-in-a-Box cluster	4.4.03	
Objects enabled through Nutanix marketplace and hosted on GPT-in-a-Box cluster	4.3.02	

Network Design

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box utilizes 100GbE end-to-end network.

[Figure 10](#) illustrates the network design used in this solution.

Figure 10. Network Design



The key aspects of network design are as follows:

- Each of the HClAF240C M7 All-NVMe server is connect to Cisco UCS 6536 Fabric Interconnect in UCS Managed Mode.
- Each node is equipped with 1x Cisco VIC 15238 2x 40/100/200G mLOM, a dual-port small-form-factor pluggable (QSFP/QSFP28/QSFP56) mLOM card which supports 40/100/200-Gbps Ethernet or FCoE.

- The network ports in the hypervisor are configured in Active-Passive Mode. These are configured automatically during automated cluster installation.
- The deployment in UCS Managed mode does not support bond mode 4 (LACP).
- Fabric Interconnect ports are separate from 100G Uplink ports and can be connected via the core LAN or through dedicated management network switch.

Cluster Design

The design incorporates a single GPU-enabled Nutanix cluster dedicated to GPT-in-a-Box workloads that offers access to large language models (LLMs). Supporting management applications like Prism Central and file and object storage are hosted on this cluster.

Size the Nutanix GPT-in-a-Box cluster Controller VM (CVM) with 16 vCPU and 64 GB of memory.

This design uses one region with a single AZ that hosts the GPT-in-a-Box cluster. This solution doesn't use any replication targets to protect the workloads because the focus is to host the GPT workloads in a single cluster.

Figure 11. GPT-in-a-Box Cluster Conceptual Architecture

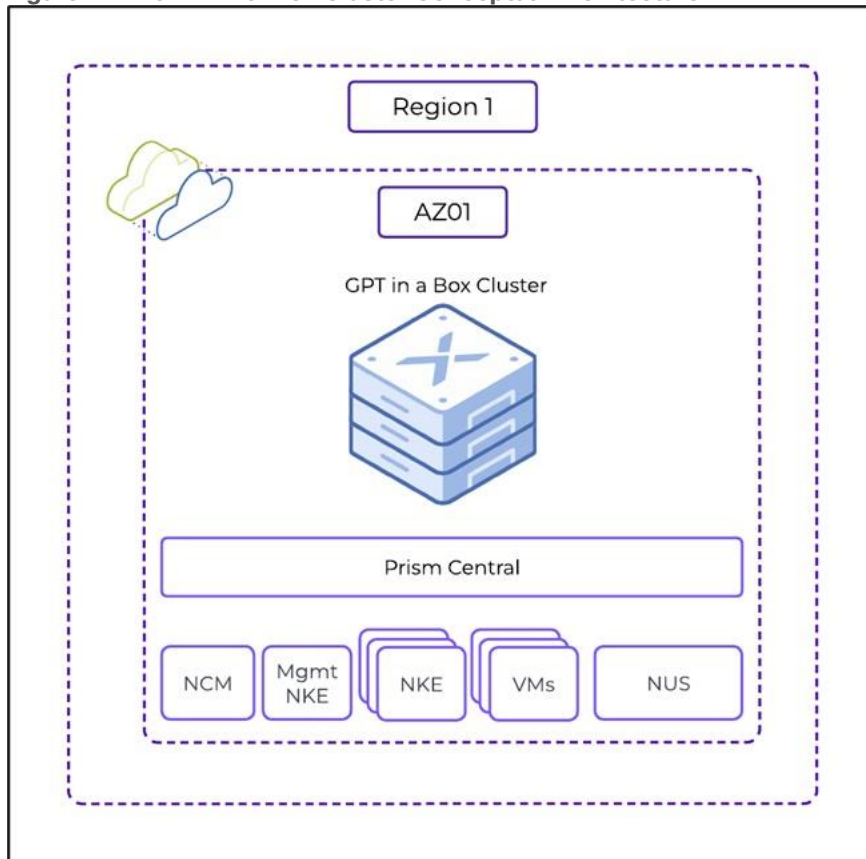


Table 5. Cluster Design Decisions

Design Option	Validated Selection
Cluster size	Ensure the full redundancy of all components in the datacenter.
CPU	Use at least 24 cores and a high clock rate.

Design Option	Validated Selection
Minimum cluster size	Use at least 4 nodes.
Cluster expansion	Expand in increments of 1.
Maximum cluster size	Use at most 16 nodes.
Networking	Use 100 GbE networking.
Cluster replication factor	Use storage replication factor 2.
Cluster high availability configuration	Guarantee high availability.
VM high availability	Enable high availability reservation on the clusters.

Cluster Resilience

VM high availability ensures that VMs restart on another AHV host in the AHV cluster when a host becomes unavailable, either because the original AHV host has a complete failure or becomes network partitioned or because of an AHV host management process failure. When the AHV host where a VM is running becomes unavailable, the VM turns off; therefore, from the perspective of the VM operating system, VM high availability involves a full VM start cycle.

Table 6. High Availability Configuration

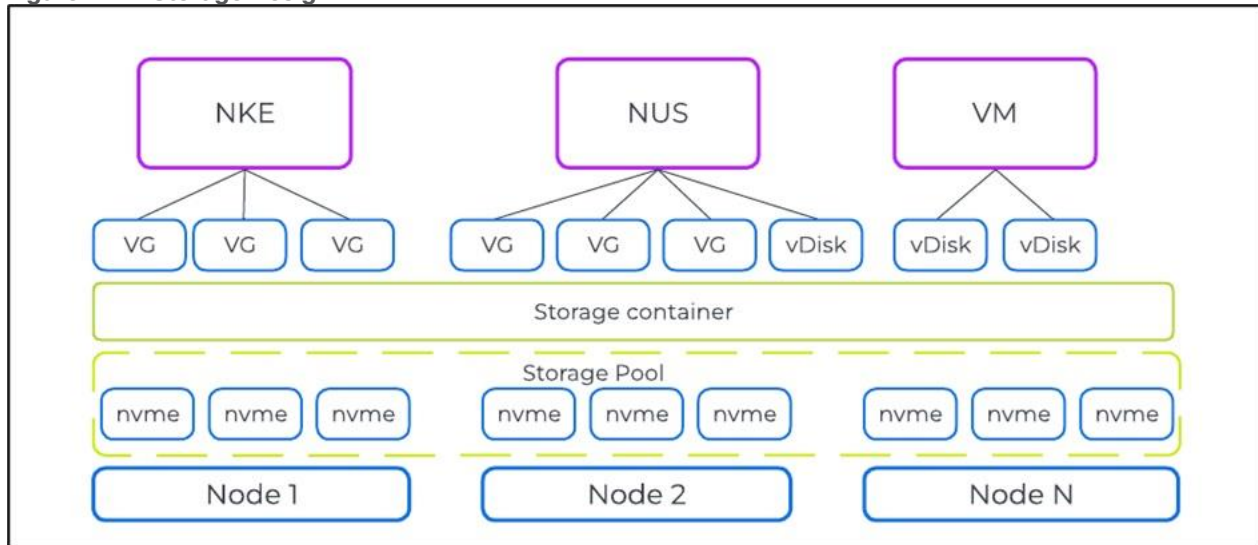
Feature	Description	Setting
High availability reservation	Guarantee compute failover capacity within cluster for application	Enabled
Rebuild capacity reservation	Guarantee storage rebuild capacity within cluster	Enabled

Storage Design

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box utilizes a distributed, shared-nothing architecture for storage.

[Figure 12](#) illustrates the storage design used in this solution.

Figure 12. Storage Design



When creating the Nutanix AHV cluster, the following storage containers are automatically created:

- NutanixManagementShare: Used for Nutanix features like Files and Objects and other internal storage needs; doesn't store workload vDisks
- SelfServiceContainer: Used by the NCM Self-Service Portal to provision VMs
- Default-Container-XXXX: Used by VMs to store vDisks for user VMs and applications

In addition to the automatically created storage containers, the following additional storage containers are created during the Nutanix Files and Objects deployment:

- NTN_<fileserver_name>_ctr: Provides storage for the file server instances, which provide NFS storage for the application tier
- objectsd<uniqueidentifier>: Data container for Nutanix Objects
- objectsm<uniqueidentifier>: Metadata container for Nutanix Objects

The Default-Container stores VMs and their vDisks. The additional containers for Nutanix Files and Objects are created throughout the deployment process of the respective components.

Note: To increase the effective capacity of the cluster, the design enables inline compression on all storage containers. It doesn't use additional functionalities such as deduplication or erasure coding. Replication factor 2 protects against the loss of a single component in case of failure or maintenance.

Data Reduction and Resilience Options

To increase the effective capacity of the cluster, the design enables inline compression on all storage containers. It doesn't use additional functionalities such as deduplication or erasure coding. Replication factor 2 protects against the loss of a single component in case of failure or maintenance.

Table 7. Data Reduction Settings

Container	Compression	Deduplication	Erasure Coding	Replication Factor
Default-Container-XXXX	On	Off	Off	2
NutanixManagementShare	On	Off	Off	2

Container	Compression	Deduplication	Erasure Coding	Replication Factor
SelfServiceContainer	On	Off	Off	2
NTNX_files_ctr	On	Off	Off	2
objects containers	On	Off	Off	2

[Table 8](#) provides information about the storage decisions made for this design.

Table 8. Storage Design Decisions

Design Option	Validated Selection
Sizing a cluster	Use an all-flash cluster to provide low-latency, high-throughput storage to support the application's active data set.
Node type vendors	Don't mix node types from different vendors in the same cluster.
Node and disk types	Use similar node types that have similar disks. Don't mix nodes that contain NVMe SSDs in the same cluster with hybrid SSD or HDD nodes.
Sizing for node redundancy for storage and compute	Size all clusters for $n + 1$ failover capacity.
Fault tolerance and replication factor settings	Configure the cluster for fault tolerance 1 and configure the container for replication factor 2.
Inline compression	Enable inline compression.
Deduplication	Don't enable deduplication.
Erasure coding	Don't enable erasure coding.
Availability domain for cluster	Use node awareness.
Storage containers in cluster	The cluster has the following storage containers: NutanixManagementShare, SelfServiceContainer, Default-Container, NTNX_files_ctr, and the two objects storage containers.
Reserve rebuild capacity	Enable reserve rebuild capacity.

Nutanix Files and Objects Design

Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box utilizes Nutanix Files providing high-performance, shared storage to applications using the NFS protocol. Nutanix Files temporarily stores the LLMs and makes them available across VMs and Kubernetes services.

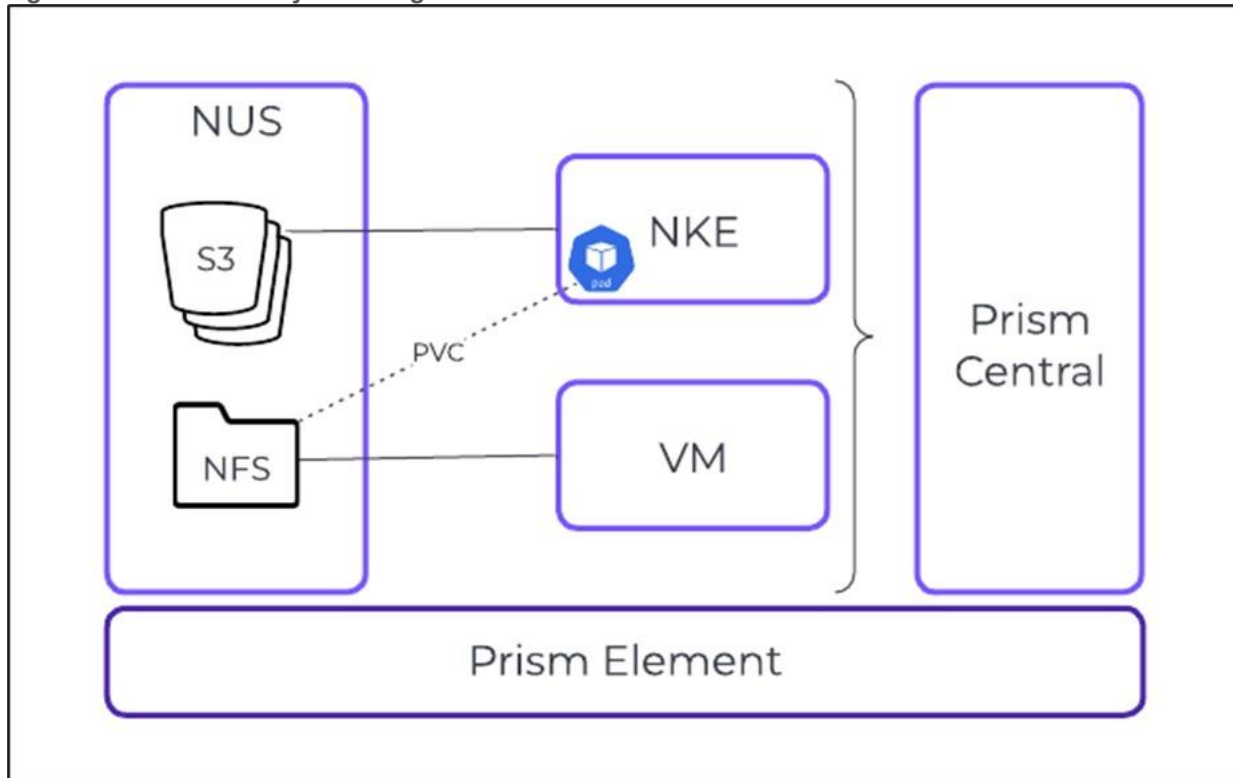
Nutanix Objects provides S3-compatible storage capabilities to the application and enables users to upload new data models.

Nutanix Files and Objects run on the same Nutanix cluster as the GPT-in-a-Box workloads. Prism Central is utilized to deploy and manage all the components

Note: If you plan to expand the environment, you can also run Nutanix Files or Nutanix Objects in a dedicated cluster.

[Figure 13](#) lists the Nutanix Files and Objects design. The VM displayed in [Figure 13](#) is an end user or jump host used to mount the NFS share and store the LLMs.

Figure 13. Files and Objects Design



Nutanix Files and Objects have the following network requirements:

- For maximum performance, the storage network for Nutanix Files uses the same subnet as the CVMs.
- To maximize security, the client network connects to a separate subnet.
- The GPT-in-a-Box cluster that provides a single file server instance requires three storage network IP addresses and four client network IP addresses for its three file server VMs (FSVMs).

Nutanix Objects runs as a containerized service on a Kubernetes microservices platform, which provides benefits such as increased velocity of new features. Several of the required storage IP addresses are for functions related to the underlying microservices platform. You must also manage these networks. Nutanix Objects with three worker nodes requires seven storage network IP addresses and two client network IP addresses.

Each cluster provisioned by NKE has minimum IP address requirements. A Kubernetes cluster in a production-level layout with three worker nodes requires one static IP address and eight or more IPAM addresses.

The client network provides all IP addresses. You might need additional IPAM addresses for additional worker nodes.

For more information, go to: [Nutanix Files, Nutanix Objects, and Nutanix Kubernetes Engine Network Design](#)

[Table 9](#) lists information about the Nutanix Files and Objects decisions made for this design.

Table 9. GPT-in-a-Box Files Design Decisions

Design Option	Validated Selection
FSVM cluster size	Use 3 FSVMs.
vCPU and memory for FSVM	Use 12 vCPU and 64 GB of memory for each FSVM.
Storage networking	Keep the storage network in the same subnet as the CVMs and AHV.
Client networking	Use a separate subnet to provide client access to storage.
Fault tolerance and replication factor settings	Configure the cluster for fault tolerance 1 and configure the container for replication factor 2.
Storage containers	Use a separate storage container to host NUS files.
Erasure coding	Don't enable erasure coding.
Compression	Enable compression.
Deduplication	Don't enable deduplication.
Shares to create	Create 1 share: ilm-repo
Protocols for shares	Use NFS for shares.

For more information, go to: [Nutanix Files and Nutanix Objects Design Decisions](#)

Table 10. GPT-in-a-Box Objects Design Decisions

Design Option	Validated Selection
Nutanix Objects cluster size	Use 3 worker nodes and 2 load balancer nodes.
Worker node size	Use 10 vCPU and 32 GB of memory for each worker node.
Load balancer node size	Use 2 vCPU and 4 GB of memory for each load balancer node.
Storage networking	Keep the storage network in the same subnet as the CVMs and AHV.
Client networking	Use a separate subnet to provide client access to storage.
Fault tolerance and replication factor settings	Configure the cluster for fault tolerance 1 and configure the container for replication factor 2.
Storage containers	Use a separate storage container to host NUS Objects.
Erasure coding	Don't enable erasure coding.
Compression	Enable compression.
Deduplication	Don't enable deduplication.
Buckets to create	Create 3 buckets: milvus (vector database), documents (for end-user access), and backup (backup target).

Management Design

Management components such as Cisco UCS Manager, Prism Central, Active Directory, DNS, and NTP are critical services that must be highly available. Prism Central is the global control plane for Nutanix, responsible for VM management, application orchestration, micro segmentation, and other monitoring and analytics functions.

This solution utilizes two key management plane:

- Cisco UCS Manager supports the entire Cisco UCS server. It enables server, fabric, and storage provisioning as well as, device discovery, inventory, configuration, diagnostics, monitoring, fault detection, auditing, and statistics collection.
- Prism Central was deployed as single VM on a separate Nutanix AHV cluster. Prism Central manages:
 - Nutanix Kubernetes Engine
 - Nutanix Files and Objects
 - VMs
 - RBAC
 - Monitoring, observability, and auditing for the core Nutanix Services

DNS Management

Name resolution and SSL certificate management are critical to deploying and managing Kubernetes clusters. In this solution, DNS root domain is hosted locally on a windows DNS server. Subdomains map the DNS and route the traffic to the Kubernetes clusters.

Monitoring

Monitoring in the solution falls into two categories: event monitoring and performance monitoring. Each category addresses different needs and issues.

In a highly available environment, you must monitor events to maintain high service levels. When faults occur, the system must raise alerts on time so that administrators can take remediation actions as soon as possible. This solution configures the Nutanix platform's built-in ability to generate alerts in case of failure.

In addition to keeping the platform healthy, maintaining a healthy level of resource usage is also essential to the delivery of a high-performing environment. Performance monitoring continuously captures and stores metrics that are essential when you need to troubleshoot application performance. A comprehensive monitoring approach tracks the following areas:

- Application and database metrics
- Operating system metrics
- Hyperconverged platform metrics
- Network environment metrics
- Physical environment metrics

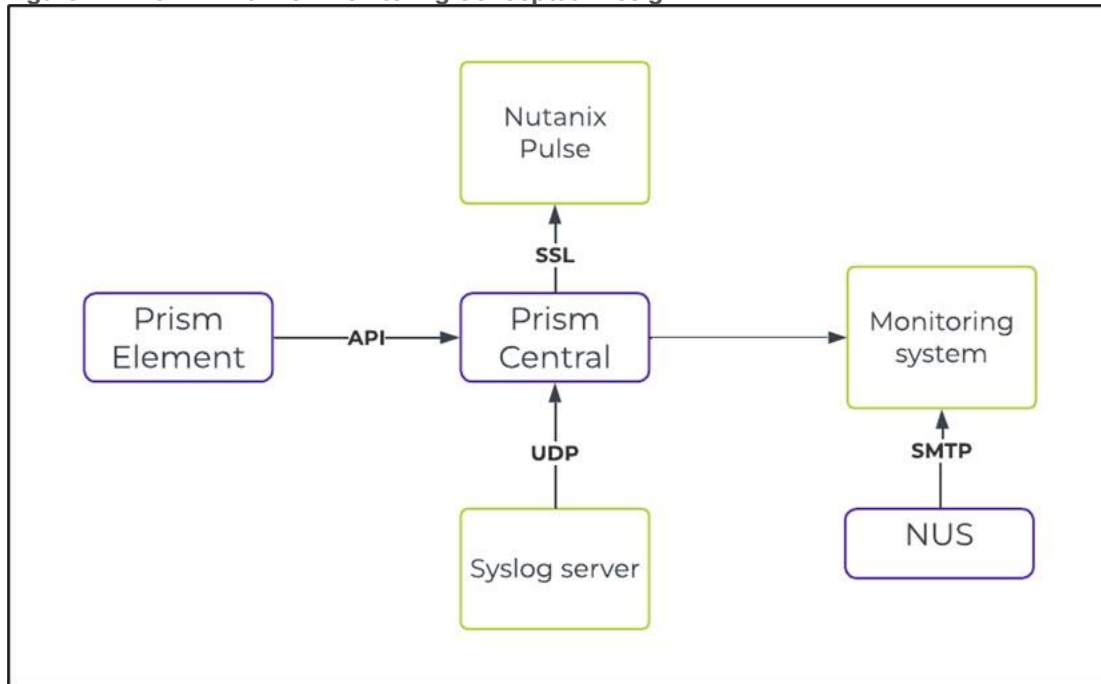
By tracking a variety of metrics in these areas, the Nutanix platform can also provide capacity monitoring across the stack. Most enterprise environments inevitably grow, so you need to understand resource usage and the rate of expansion to anticipate changing capacity demands and avoid any business impact caused by a lack of resources.

Monitoring Conceptual Design

In this design, Prism Central performs event monitoring for the Nutanix core infrastructure. This design uses syslog for log collection; for more information, see the Security and Compliance section. SMTP-based email alerts serve as the channel for notifications in this design. To cover situations where Prism Central might be unavailable, each Nutanix cluster in this design sends out notifications using SMTP as well. The individual Nutanix clusters send alerts to a different receiving mailbox that's only monitored when Prism Central isn't available.

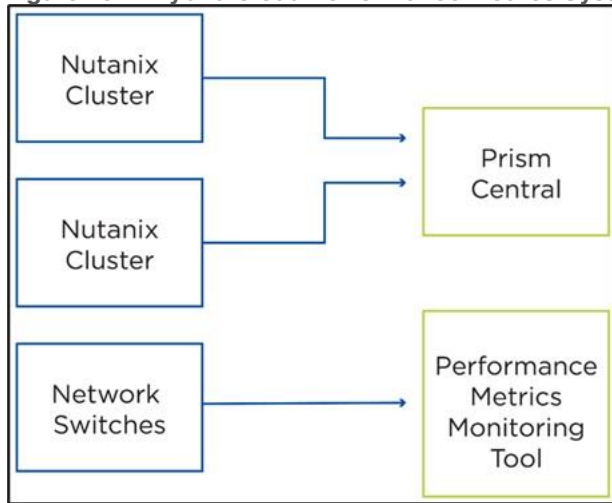
Logs from the Kubernetes cluster go to Nutanix Objects through a long-term S3 exporter and to the syslog server through a syslog exporter. The syslog server transmits the data to Prism Central using the User Datagram Protocol (UDP), and Prism Element transmits the data to Prism Central using an API. Prism Central then transmits the log to Nutanix Pulse (using the Secure Sockets Layer) and to the monitoring system, which sends the log to Nutanix Unified Storage using SMTP.

Figure 14. GPT-in-a-Box Monitoring Conceptual Design



Prism Central monitors cluster performance in key areas such as CPU, memory, network, and storage usage, and captures these metrics by default. When a Prism Central instance manages a cluster, Prism Central transmits all Nutanix Pulse data, so it doesn't originate from individual clusters. When you enable Nutanix Pulse, it detects known issues affecting cluster stability and automatically opens support cases.

Figure 15. Hybrid Cloud Performance Metrics Systems



The network switches that connect the cluster also play an important role in cluster performance. A separate monitoring tool that's compatible with the deployed switches can capture switch performance metrics. For example, an SNMP-based tool can regularly poll counters from the switches.

[Table 11](#) provides descriptions of the monitoring design decisions.

Table 11. Monitoring Design Decisions

Design Option	Validated Selection
Platform performance monitoring	Prism Central monitors Nutanix platform performance.
Network switch performance monitoring	A separate tool that performs SNMP polling to the switches monitors network switch performance.
SMTP alerting	Use SMTP alerting; use an enterprise SMTP service as the primary SMTP gateway for Prism Element and Prism Central.
SMTP alerting source email address	Configure the source email address to be <code>clustername@<yourdomain>.com</code> to uniquely identify the source of email messages. For Prism Central, use the Prism Central host name in place of cluster name.
SMTP alerting Prism Central recipient email address	Configure the Prism Central recipient email address to be <code>primaryalerts@<yourdomain>.com</code> .
SMTP alerting Prism Element recipient email address	Configure the Prism Element recipient email address to be <code>secondaryalerts@<yourdomain>.com</code> .
NCC reports	Configure daily NCC reports to run at 6:00 AM local time and send them by email to the primary alerting mailbox.
Nutanix Pulse	Configure Nutanix Pulse to monitor the Nutanix cluster and send telemetry data to Nutanix.

Security and Compliance

Nutanix recommends a defense-in-depth strategy for layering security throughout any enterprise datacenter solution. This design section focuses on validating the layers that Nutanix can directly oversee at the control and data-plane levels.

Security Domains

Isolate Nutanix cluster management and out-of-band interfaces from the rest of the network using firewalls, and only allow direct access to them from the management security domain. In addition, Nutanix recommends separating out-of-band management and cluster management interfaces onto a dedicated VLAN away from the application traffic.

Syslog

For each control plane endpoint (Prism Central), system-level internal logging goes to a centralized third-party syslog server that runs in the existing customer landscape. The system is configured to send logs for all available modules when they reach the syslog Error severity level.

This design assumes that the centralized syslog servers are highly available and redundant, so you can inspect the log files in case the primary log system is unavailable.

Certificates

SSL endpoints serve all Nutanix control plane web pages. In the deployment and validation, self-signed certificates are utilized. The existing solution can replace the default self-signed certificates with certificates signed by an internal certificate authority from a Microsoft public key infrastructure (PKI). Any client endpoints that interact with the control plane should have the trusted certificate authority chain preloaded to prevent browser security errors.

Note: Certificate management is an ongoing activity, and certificates need to be rotated periodically. This solution signs all certificates for one year of validity.

Table 12. Security Design Decisions

Design Option	Validated Selection
Data-at-rest encryption (DaRE)	Don't use DaRE.
SSL endpoints	Sign control plane SSL endpoints with an internal trusted certificate authority (Microsoft PKI).
Certificates	Provision certificates with a yearly expiration date and rotate accordingly.
Authentication	Use Active Directory LDAPS authentication.
Control plane endpoint administration	Use a common administrative Active Directory group for all control plane endpoints.
Cluster lockdown mode	Don't enable cluster lockdown mode (allow password-driven SSH).
Non-default hardening options	Enable AIDE and hourly SCMA.
System-level internal logging	Enable error-level logging to an external syslog server for all available modules.

Design Option	Validated Selection
Syslog delivery	Use UDP transport for syslog delivery.

Table 13. Security Configuration References

Design Option	Validated Selection
Active Directory	AD-admin-group:ntnx-ctrl-admins
Syslog Server	infra-az[1..2]-syslog:6514 (udp)

Kubernetes Cluster Design

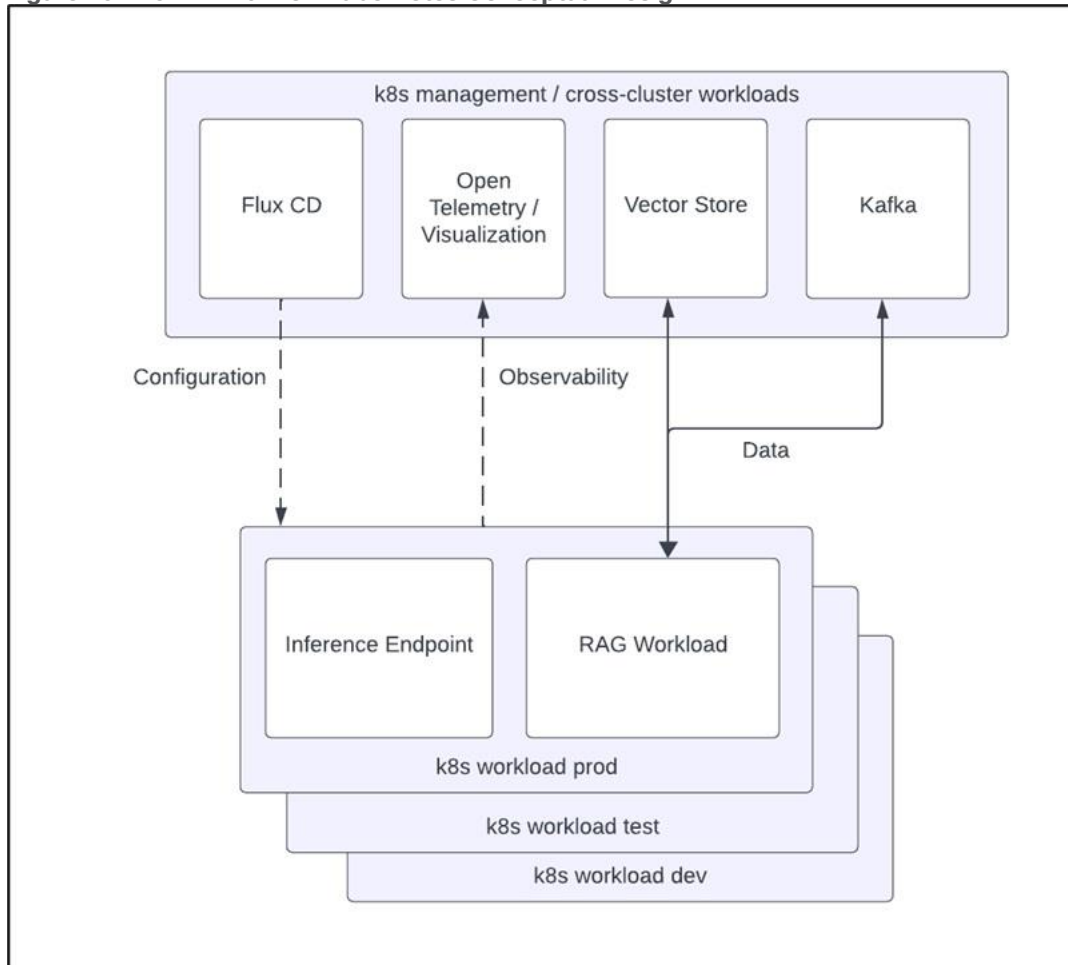
The following sections describe the designs for the Kubernetes on the Nutanix platform, using Nutanix Kubernetes Engine (NKE) features and integrating with essential Kubernetes tools and practices for efficient and secure operations.

A single management cluster runs all components for application configuration, observability, and cross-workload cluster-persistent applications like Uptrace (an OpenTelemetry-based observability platform), Kafka, and Milvus (a cloud-native vector database that supports various data types).

All production Kubernetes clusters are set up as production-level clusters using multiple primary and worker nodes distributed across different physical hosts to ensure high availability.

Flux CD on the Kubernetes management cluster configures applications on the Kubernetes workload prod, test, and dev clusters, ensuring continuous and automated deployment that aligns with the configurations specified in the Git repositories. The workload clusters provide metrics, logs, and traces across all clusters to OpenTelemetry, and Uptrace, the front-end, user-friendly interface, visualizes and queries the collected data for thorough monitoring and debugging. The Retrieval-Augmented Generation workload on the Kubernetes workload production cluster sends data to and receives data from the vector store and Kafka on the Kubernetes management cluster.

Figure 16. GPT-in-a-Box Kubernetes Conceptual Design



Kube-VIP handles network connectivity for load balancing, Ingress controllers manage external access to services with HTTP and HTTPS routes, and Cert-Manager automates the management and issuance of TLS certificates, enhancing secure communications.

Workload clusters have a dedicated node pool specifically for GPU resources. This node pool hosts pods that require GPU capabilities, such as nodes with machine learning or data processing workloads. Taints are used on GPU nodes to prevent non-GPU workloads from being scheduled on them. Corresponding tolerations need to be added to pods that require GPUs.

Table 14. GPT-in-a-Box Cluster with NKE Scalability Design Decisions

Design Option	Validated Selection
NKE cluster type	Use a production cluster with an active-passive control plane.
Control plane size	Size the control plane with 8 CPU and 16 GB of memory.
Initial cluster size	Start with 3 worker nodes with 12 CPU, 16 GB of memory, and 300 GB of storage.
GPU pool size	Use 2 worker nodes with 12 CPU, 40 GB of memory, and 300 GB of storage.

Design Option	Validated Selection
Monitoring	Enable monitoring components.

Note: NKE cluster names use a maximum of 22 characters.

In this design, you can scale the worker nodes for the NKE management and workload clusters up or out to accommodate different workload sizes. Nutanix recommends scaling out. The total number of workers in the GPU node pools is constrained by the number of physically installed GPUs.

Kubernetes Resilience

NKE clusters are production-level clusters that provide a resilient control plane by running multiple nodes for the control plane and etcd. To ensure high availability, Kubernetes deployments use multiple replica pods and implement pod anti-affinity rules. This approach helps maintain service availability, even in the event of a worker node update or failure.

Kubernetes services and ingress controllers perform the essential service of load balancing by evenly distributing network traffic across all available pods, enhancing service reliability and system performance.

Kubernetes Networking

Each deployed cluster uses a base configuration for networking:

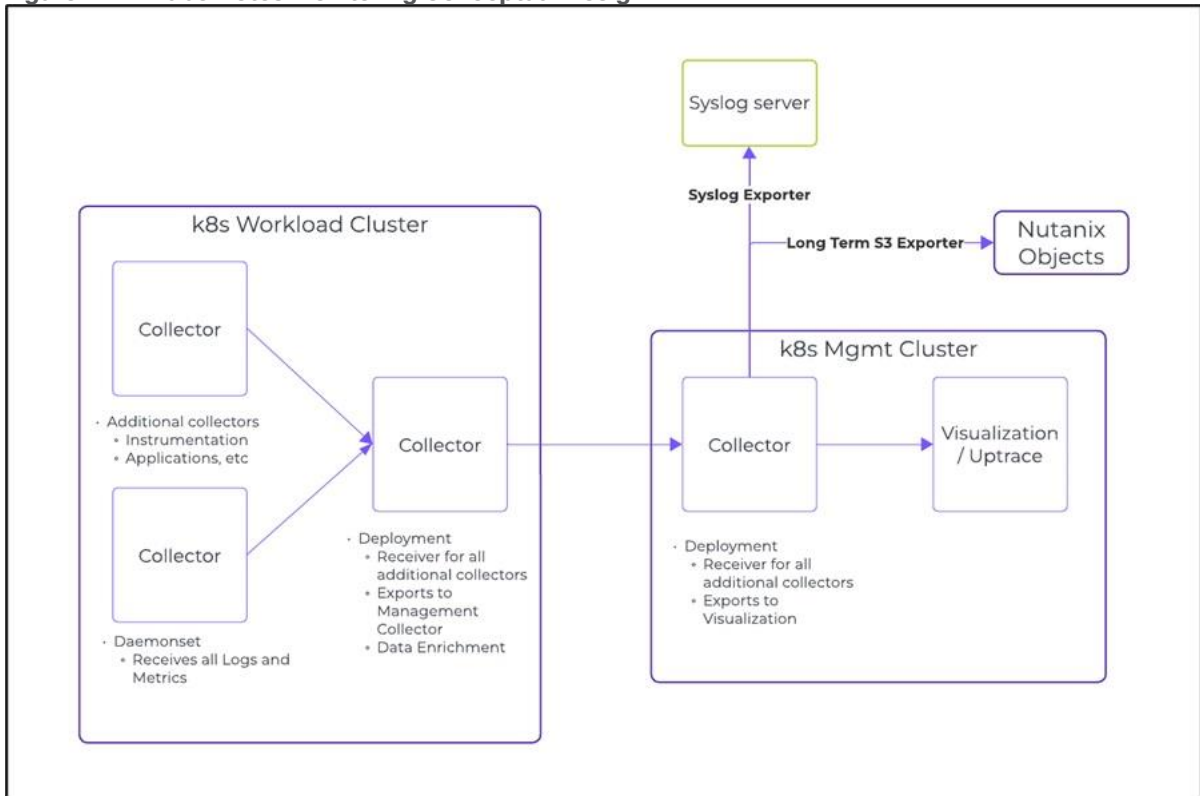
- kube-vip: Load-balancing service
- nginx-ingress: Ingress service for L4 and L7 network traffic
- cert-manager: Service that creates valid SSL certificates for TLS application services
- Local DNS with self-signed certificate

Kubernetes Monitoring

Kubernetes and application-level monitoring is based on the core infrastructure monitoring concept. The observability stack uses OpenTelemetry to collect and move the following data between the Kubernetes clusters:

- Kubelet metrics
- Host metrics
- Kubernetes cluster metrics
- Kubernetes events
- Pod logs
- Prometheus metrics from service monitors
- Application-specific data from instrumentalization

Figure 17. Kubernetes Monitoring Conceptual Design



The Kubernetes management cluster uses the open-source project Upttrace to provide a front end for traces, metrics, and alerts. You can substitute Upttrace with its enterprise offering or use another front end that supports OpenTelemetry.

The Kubernetes workload cluster has the Daemonset collector (receives all logs and metrics) and additional collectors for instrumentation and applications that send data to the deployment collector, which provides data enrichment and exports to the deployment collector on the management cluster. The management cluster deployment collector exports data to Upttrace or the chosen visualization front end and to the external syslog server, which Prism Central also uses.

Kubernetes Backup

The default storage class installed by NKE during deployment provides persistent storage for the management and workload clusters. The persistent volumes and application data in the management cluster are protected by a Velero backup schedule and stored in a single S3 bucket. Because application deployment is based on GitOps using FluxCD, you don't need to back up the applications.

You can rebuild the Milvus vector database from data in the user buckets. If the data in the user buckets is ephemeral, you can use the Milvus Backup tool to back up and restore Milvus data.

Large Language Model Design

This reference design presents a robust architecture for running LLM applications around the Kubernetes-based Nutanix AI inference endpoint, using NKE as the orchestration platform. It provides a comprehensive, scalable, and efficient framework tailored for building RAG pipeline applications. This framework capitalizes on the latest LLM technologies and supporting tools, ensuring ease of development, deployment, and monitoring.

Large Language Model Logical Design

At the heart of this architecture is the inference endpoint, delivered through kserve. This element is vital for the deployment and management of machine learning models, especially for real-time inference needs. The integration with kserve and Nutanix ensures scalability, reliable accessibility, and consistent performance.

The architecture is modular, allowing you to independently scale and update each component. It is compatible with the RAG framework, enabling the LLM to access information from the Milvus database, which enhances the generation process.

Data ingestion is versatile, supporting both batch and event-based approaches through Kafka. This flexibility enables the system to manage both periodic large-scale batch uploads and continuous real-time data streaming. The system uses serverless functions built on Knative for processing events, which are triggered by Nutanix Objects event notifications through Kafka. This setup ensures efficient and scalable processing of incoming data streams.

Once ingested, the data is vectorized (transformed by the embedding model into a vector representation) using Langchain and stored in Milvus. Nutanix Objects provides the scalable back-end storage required to accommodate the large-scale data demands of Milvus. The LLM models, hosted on kserve, access and use this vectorized data to enhance language generation capabilities.

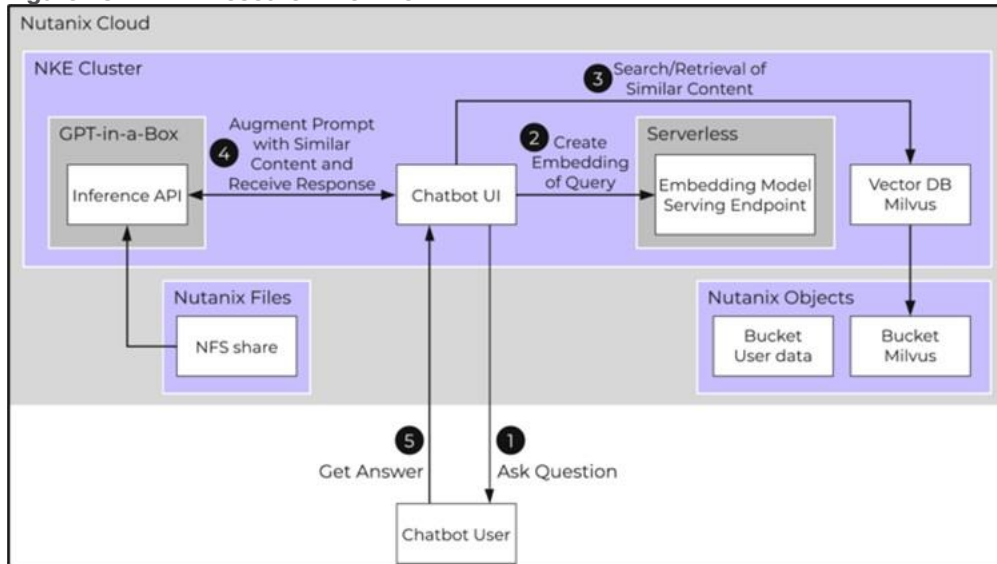
Solution uses OpenTelemetry to monitor the entire data flow for system observability. Additionally, Jupyter Lab offers a conducive environment for testing, experimentation, and development, with direct access to GPU resources for intensive computational tasks.

Large Language Model Research Workflow

The LLM uses the following research workflow:

1. **Ask a question:** The interaction begins when the end user poses a question through the UI or chatbot interface.
2. **Create a query embedding:** The embedding model transforms the user's query into a vector representation. This process is known as vectorization.
3. **Search and retrieve similar context:** The vector database, which is specifically designed for similarity searches, stores the document embeddings generated by the model. It can efficiently search for and retrieve items based on these embeddings, which encapsulate the semantic meaning of the texts.
4. **Send the prompt:** The workflow augments the user's query with relevant contextual information retrieved from the database, then sends this enriched query as a prompt to the LLM endpoint. The LLM processes the enriched query and generates a response.
5. **Get an answer:** The UI or chatbot interface presents the LLM's response as the answer to the user's query.

Figure 18. LLM Research Workflow

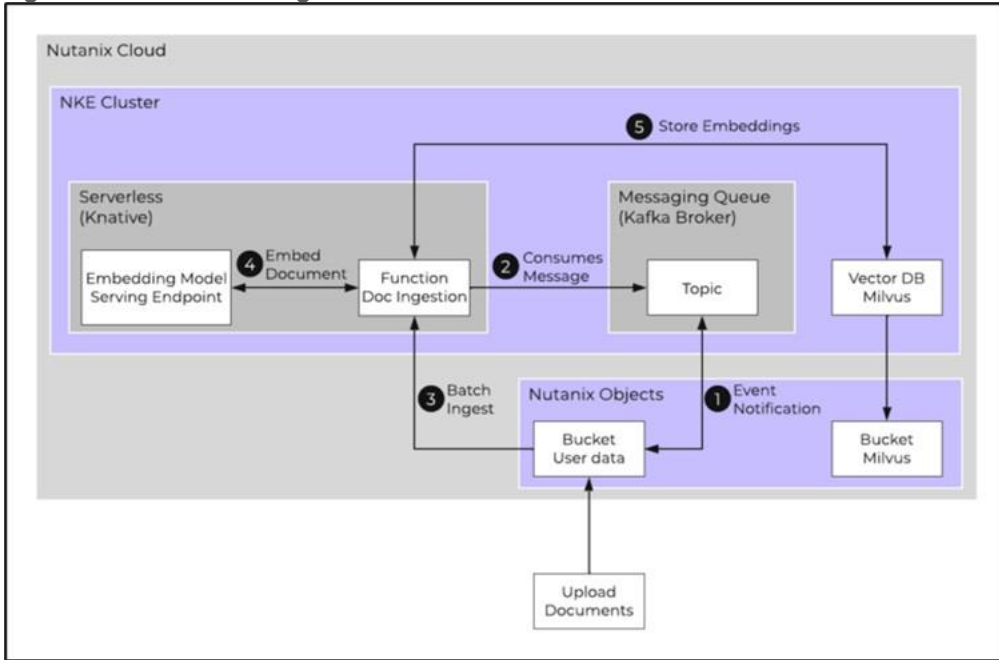


Document Ingestion Workflow

The design uses the following workflow for document ingestion:

1. Uploading the document: Each time a new document is added to the bucket, an event notification is sent through Kafka.
2. Processing the Kafka event: The notification triggers the document ingestion service to generate a new embedding for that specific document.
3. Ingesting the batch: The document ingest function downloads the document.
4. Embedding the document: Embedding involves splitting the document into smaller segments and using an embedding model to create a vector representation of each segment.
5. Storing the embeddings: The generated vectors are stored in a vector database for future retrieval and searches.

Figure 19. Document Ingestion Workflow



Backup and Disaster Recovery

The scope of this solution is a single standalone GPT-in-a-Box cluster, and you must back up the application data on the S3 buckets in the Nutanix Objects store to an external environment. You don't need to back up the NKE configuration data because Flux CD handles the deployment and stores the configuration data in an external Git repository.

You can use the streaming replication mechanism built into Nutanix Objects to replicate the data at the bucket level to a different S3 object store outside the GPT-in-a-Box cluster. You can also use the existing backup solution to back up the persistent application data and store it outside the GPT-in-a-Box cluster.

Solution Deployment

This chapter contains the following:

- [Infrastructure Deployment](#)
- [Deploy GPT-in-a-Box using GitOps](#)

Infrastructure Deployment

This section is a prerequisite for installing the GPT-in-a-Box solution and divided into the following key sections and procedures:

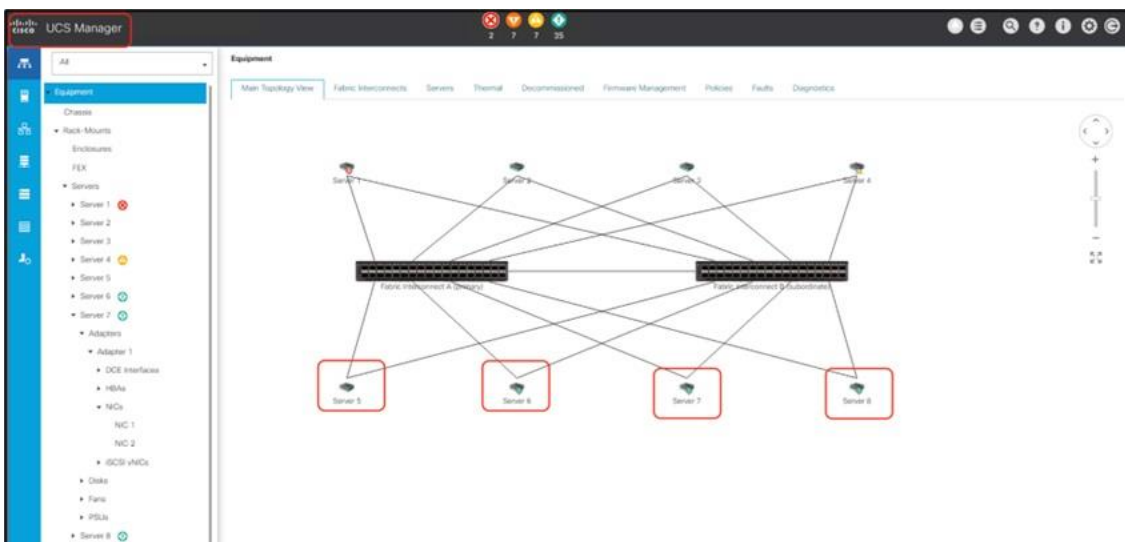
1. [Install AHV-based CCHC with Nutanix Cluster](#)
2. [Install NVIDIA Grid Driver](#)
3. [Install and Configure Nutanix Files](#)
4. [Install and Configure Nutanix Objects](#)
5. [Install and Configure Nutanix Kubernetes Clusters](#)

Procedure 1. Install AHV-based CCHC with Nutanix Cluster

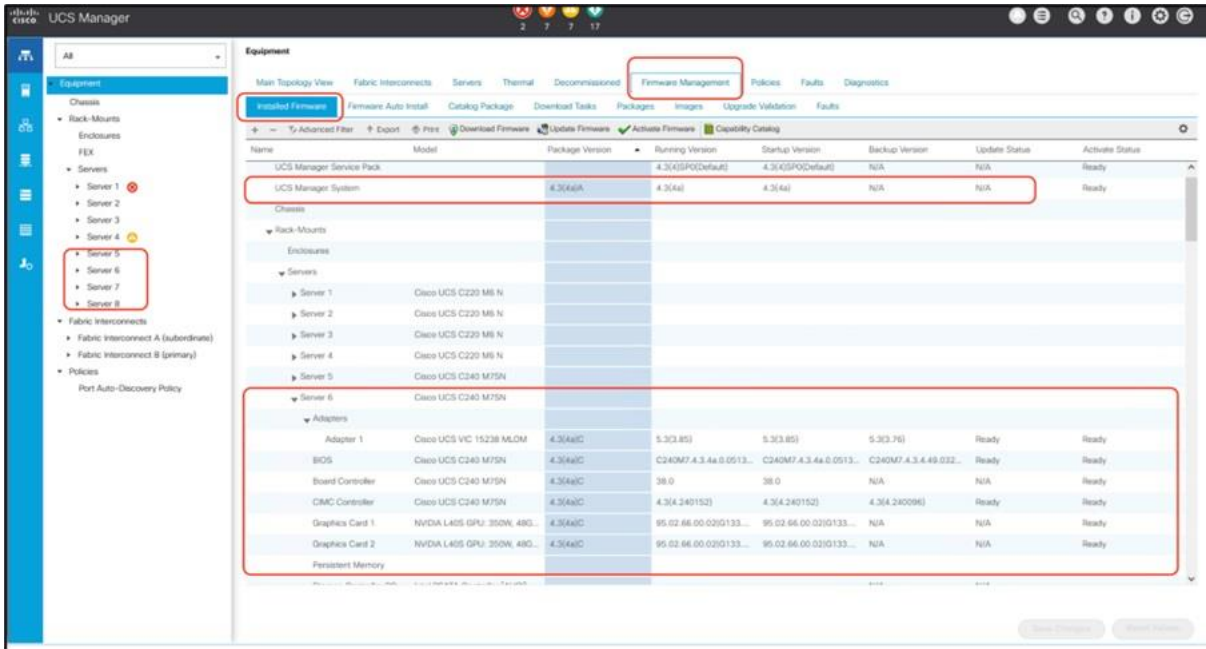
This solution requires a minimum of four (4) node of CCHC with Nutanix Cluster. Each of the cluster node is enabled with 2x NVIDIA L40S GPUs with HClAF240C M7 All-NVMe server. For detailed specifications on the specification of server nodes, go to section [Solution Design](#).

Note: A complete install process of AHV based CCHC with Nutanix cluster is outside the scope of this document. The key validation steps and references are detailed in this section.

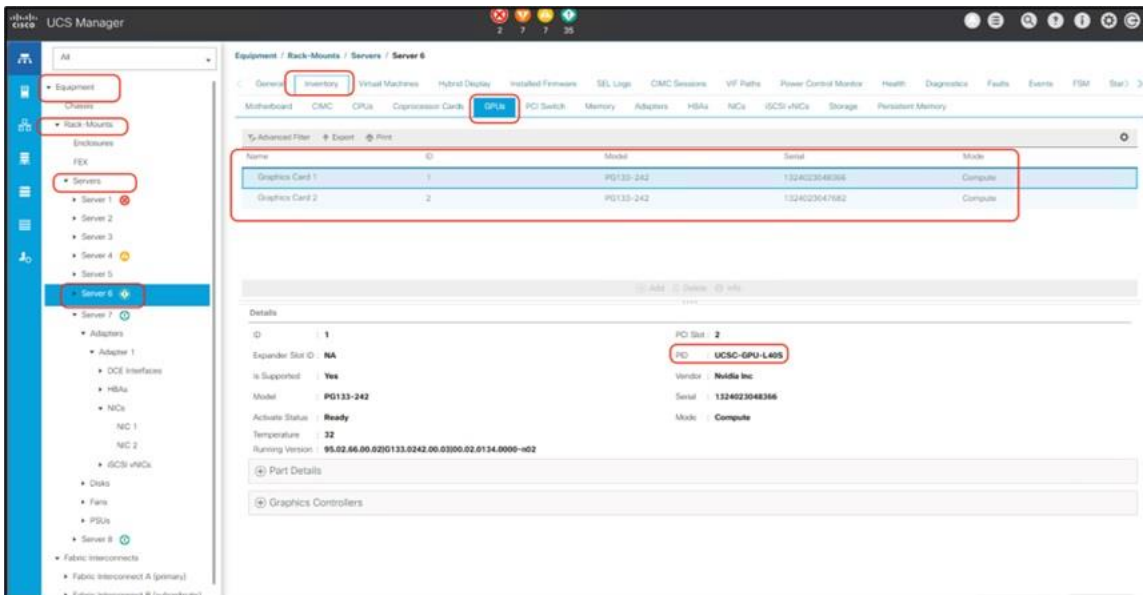
Step 1. Verify four (4) GPU enabled HClAF240C M7 All-NVMe nodes are discovered in Cisco UCS Manager. For connectivity details please refer to Infrastructure and Network Design section.



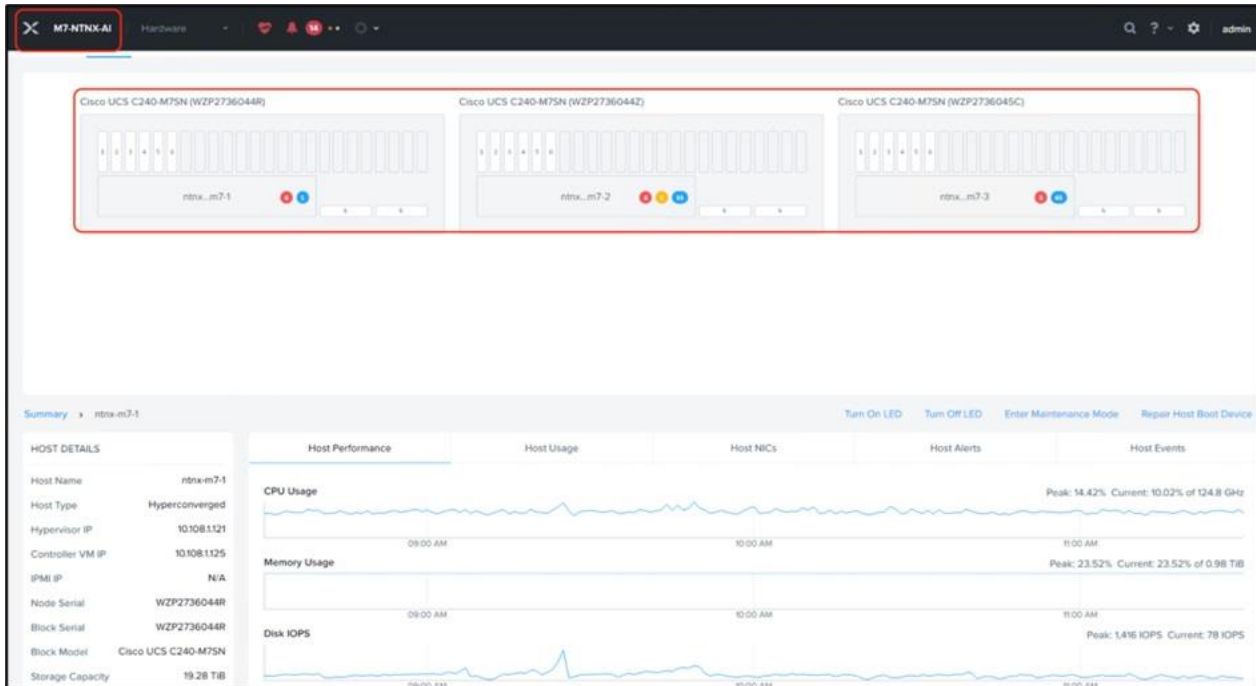
Step 2. Ensure UCS Firmware is 4.3(4a) or above.




Step 3. Navigate to Equipment > Rack-mounts > Servers > Server(x). On the right navigation window, click Inventory > GPUs tab. Ensure a minimum of 1 x Nvidia L40S GPUs are displayed.



Step 4. Install Nutanix cluster with AOS 6.7.1., see the [Field Guide](#) to successfully install the cluster.



About Nutanix ? X



Version 6.7.1.5 - NO_LICENSE License

NCC Version: 4.6.6.1
LCM Version: 3.0.0.1

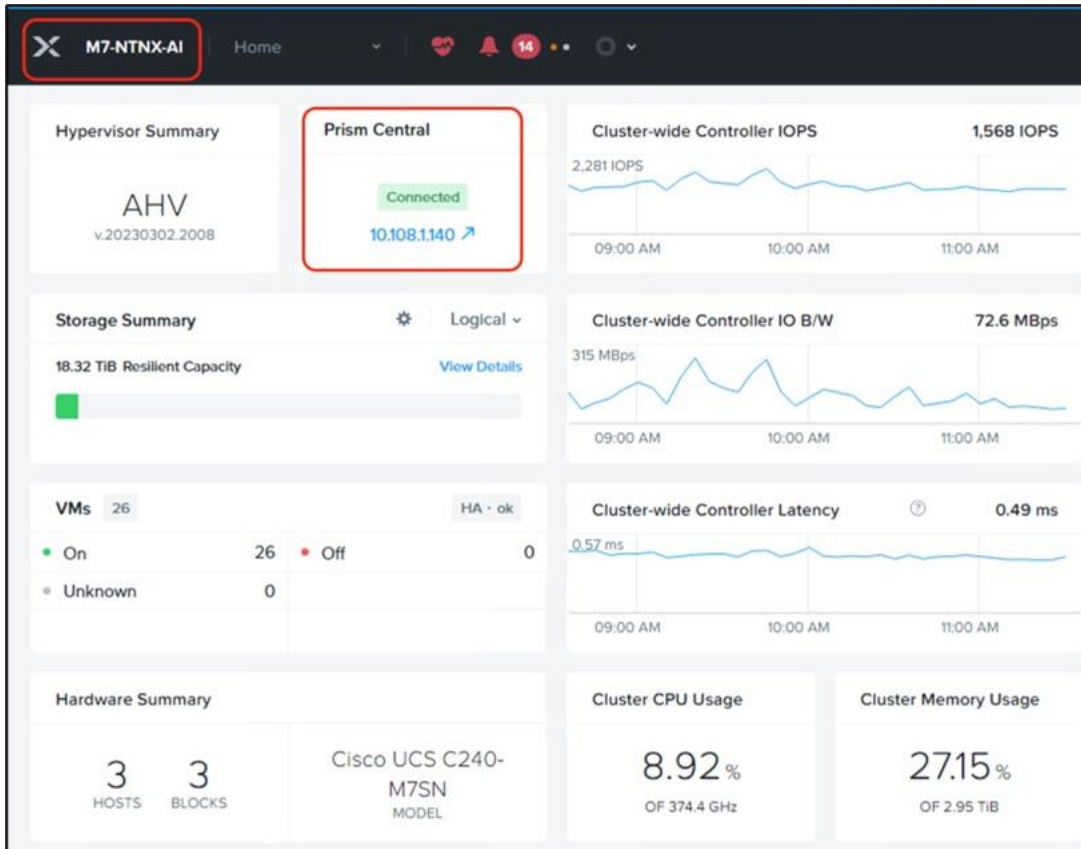
Patent Information: nutanix.com/patents
[End User License Agreement](#)

Prism UI is a product of Nutanix Inc.
Copyright 2023. All rights reserved.

Close

Note: The screenshot shown above displays three (3) nodes. To enable resiliency to failures, it is highly recommended to start with at least four (4) nodes.

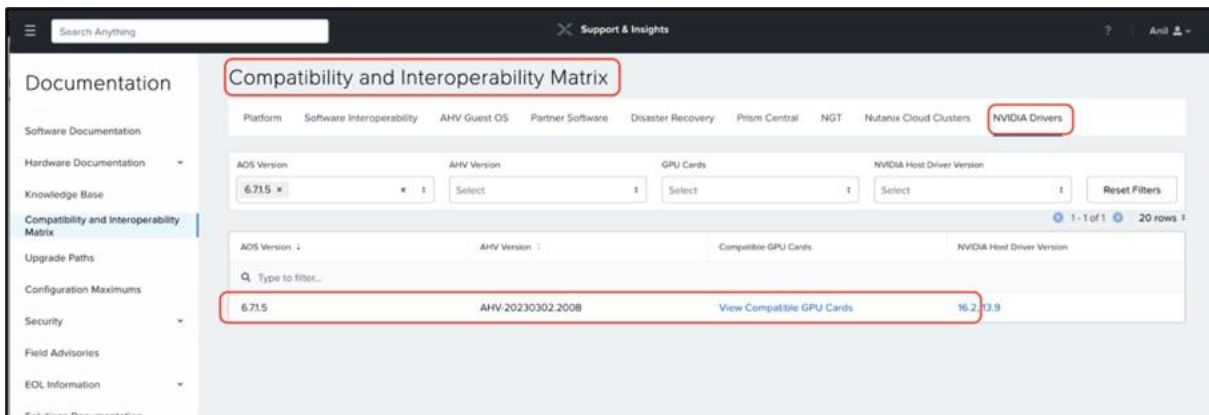
Step 5. Provision a Prism Central Instance deployed on Nutanix cluster and register the existing cluster to Prism Central. Customers have a choice to either deploy Prism Central on the GPT-in-a-Box cluster or utilize a pre-existing Prism Central Instance.



Procedure 2. Install NVIDIA Grid Driver

The following steps details the process to deploy the NVIDIA driver on cluster nodes configured with L40S GPUs.

Step 1. Download the GPU driver as per the AOS build installed on the cluster. This can be retrieved from [Compatibility and Interoperability matrix](#) on Nutanix portal. The existing cluster with AOS 6.7.1.5 is compatible with lcm_nvidia_20230302.2008_535.129.03.



30XXX

Step 2. Ensure the Nvidia GPU is available as a pci device. Log into AHV (node Host IP) with root / nutanix/4u or password set by the administrator. Execute `lspci | grep -l nvidia` and ensure GPU is installed on the host.

```

EU:00.5 RAID bus controller: Intel Corporation volume management L
[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 ~]# █

```

Step 3. Log into Nutanix node CVM (Nutanix/Nutanix/4u) or with the password set by the administrator and copy the tar file to /home/Nutanix

Step 4. Install the driver `install_host_package -r lcm_nvidia_20230302.2008_535.129.03.tar.gz`. This driver is installed across all nodes through the rolling restart process.

```

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
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:764 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: e18.nutanix.20230302.2008
2024-04-17 01:27:35,653Z INFO MainThread kvm_upgrade_helper.py:244 Current hypervisor version: e18.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'eccli task.get 10d22151-7c76-49f1-76c5-da3068aa500b'
nutanix@NTNX-WZP2736045C-A-CVM:10.108.1.127:~$ █

```

Step 5. Run `nvidia-smi` and inspect the output for a table containing the driver version and detected GPU resources.

```

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
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:764 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: e18.nutanix.20230302.2008
2024-04-17 01:27:35,653Z INFO MainThread kvm_upgrade_helper.py:244 Current hypervisor version: e18.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'eccli task.get 10d22151-7c76-49f1-76c5-da3068aa500b'
nutanix@NTNX-WZP2736045C-A-CVM:10.108.1.127:~$ █

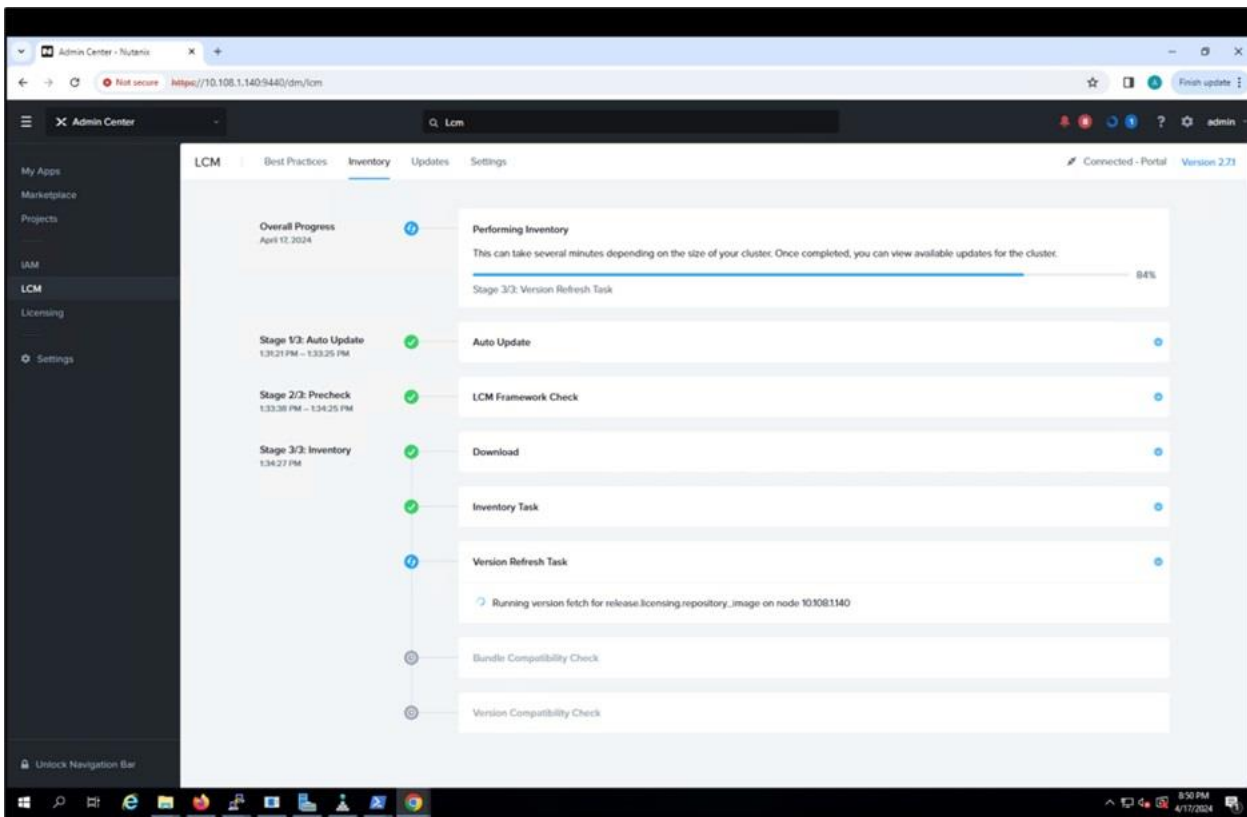
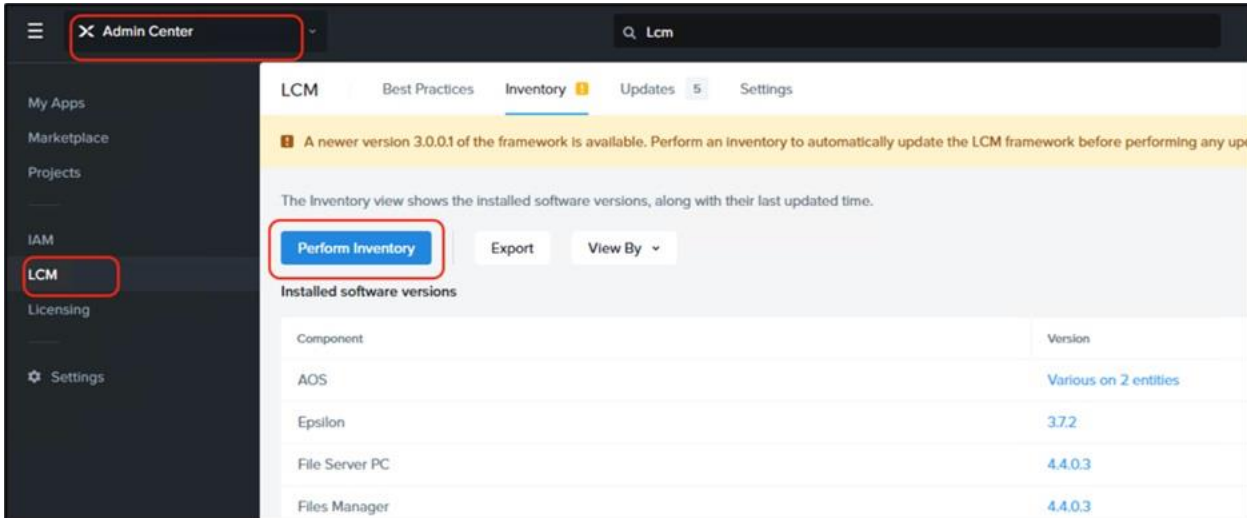
```

Procedure 3. Install and Configure Nutanix Files

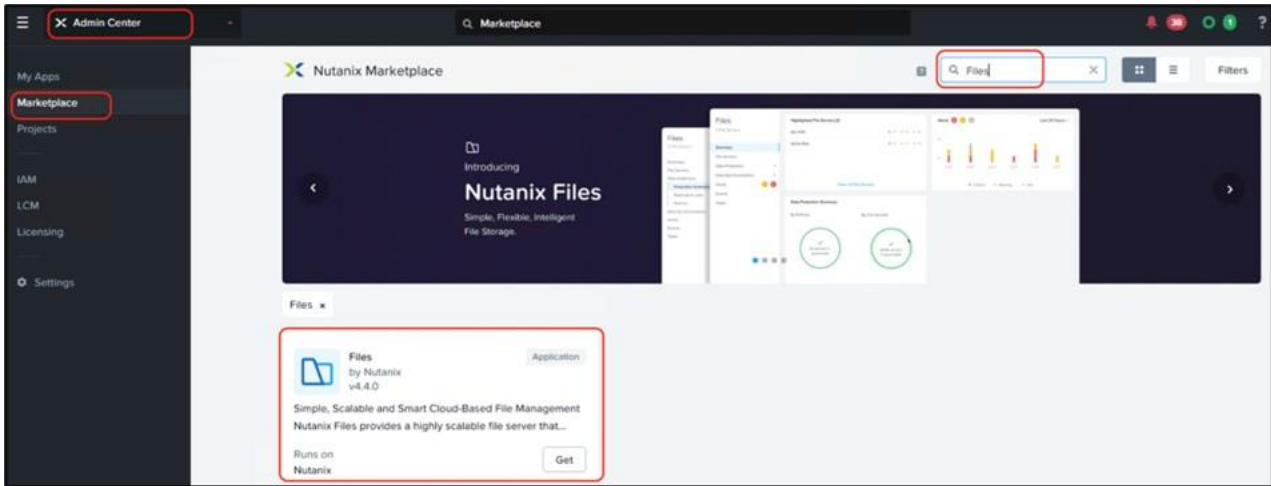
This procedure provides the high-level steps to enable and configure Nutanix Files from Prism Central. Nutanix Files runs on the same cluster as GPT-in-a-Box cluster. For Nutanix Files architecture details, go to [Files](#) on the Nutanix Portal.

Step 1. Log into Prism Central and ensure Name Server and NTP Server are updated.

Step 2. Go to Prism Central, select Admin Central, select LCM, and click Perform Inventory.



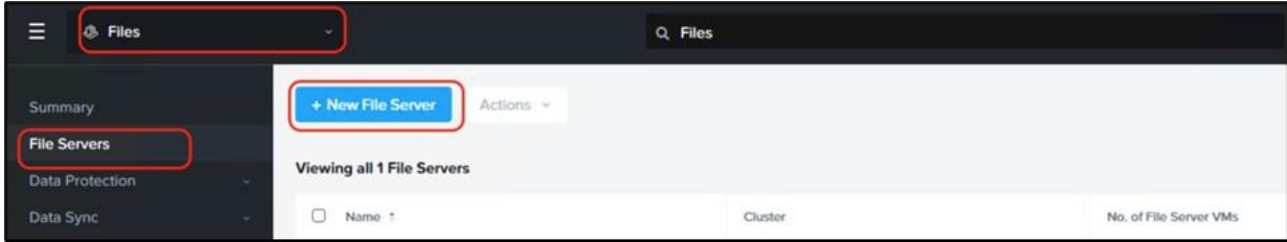
Step 3. Select Marketplace and search for Files.



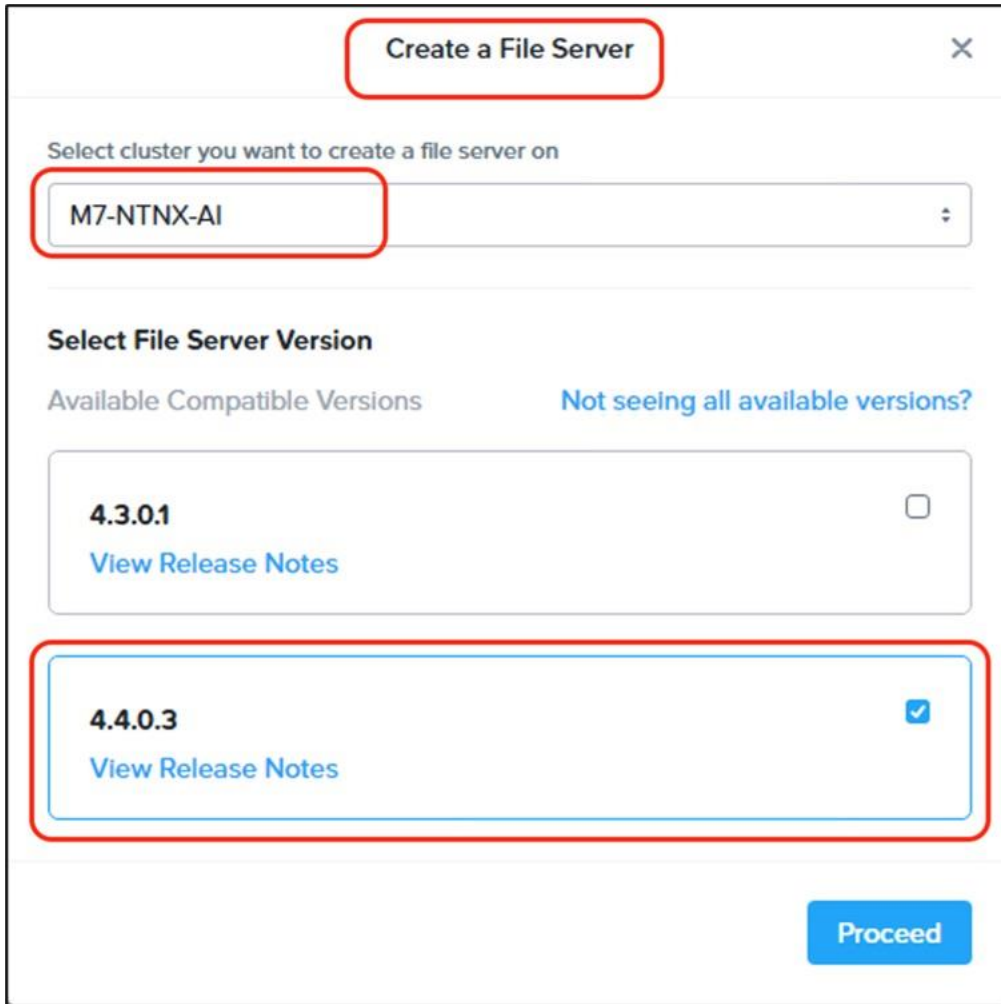
Step 4. From Files click Get. This will enable File Services.

Step 5. Select Files and enable File Services.

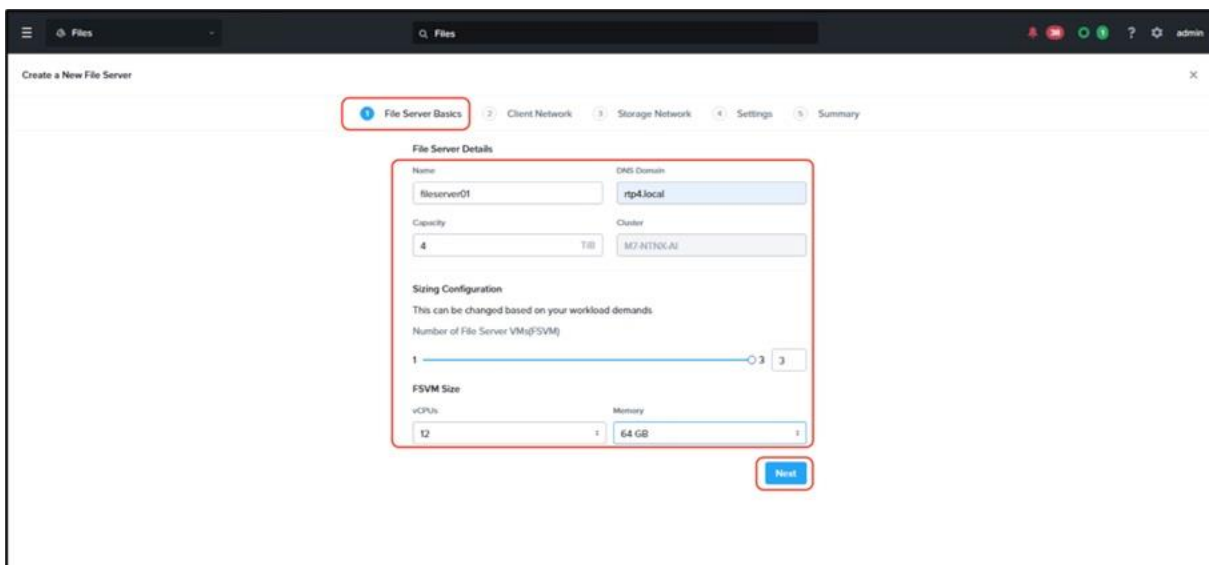
Step 6. From Files navigate to File Servers and select + New File Server.



Step 7. From the Create a File Server window, select the 'GPT-in-a-Box' cluster and check File Server 4.4.0.3 and click Proceed.



Step 8. Enter file server name, DNS domain, capacity (4TiB) , FSMV as 3 with 12vCPU and 64GB memory and click Next. These specifications are detailed in section [Nutanix Files and Objects Design Decision](#).



Step 9. Enter the client network, subnet Mask, Gateway, and IPs for the client network.

Files

Create a New File Server

Network

VLAN1081

Subnet Mask
255.255.255.0

Gateway
10.108.1.254

Static IP Addresses 3 IPs Required [Use non-continuous IPs](#)

Start IP Address: 10.108.1.224

End IP Address: 10.108.1.223

Add IPv6

Back Next

Step 10. Select Storage Network, which is the same as the CVM network, and click Next.

Files

Create a New File Server

1 File Server Basics 2 Client Network 3 Storage Network 4 Settings 5 Summary

Network

VLAN1081

Subnet Mask
255.255.255.0

Gateway
10.108.1.254

Static IP Addresses 4 IPs Required [Use non-continuous IPs](#)

Start IP Address: 10.108.1.224

End IP Address: 10.108.1.227

Back Next

Step 11. DNS Server and NTP server are pre-populated as per the settings on the cluster, click Next. Verify the summary and click Create.

The screenshot shows the 'Summary' step of a configuration wizard. At the top, there are five numbered steps: 1 File Server Basics, 2 Client Network, 3 Storage Network, 4 Settings, and 5 Summary (highlighted). Below the steps, a message states: "Once you proceed, 3 nodes (file server VMs) will be deployed on the Nutanix Cluster and configured as a file server cluster." The configuration is organized into sections:

- Cluster Details:**
 - File Server FQDN: fileserver01.rtp4.local
 - Capacity: 4 TiB
 - Cluster: M7-NTNX-AI
- File Server Sizing Configuration:**
 - No. of File Server VMs (FSVMs): 3 Nodes
 - FSVM Size: 12vCPUs, 64 GB Memory
- Network:**
 - Storage Network: [VLAN1081](#)
 - Client Network: [VLAN1081](#)
- Settings:**
 - DNS: 10.108.16,172.20.4.53
 - NTP: 172.20.10.18,172.20.10.15,0.pool.ntp.org

At the bottom, there are two buttons: "Back" and "Create".

Step 12. When File Server is created, create a NFS share 'llm-repo' with the default settings as shown below:

The screenshot shows the configuration page for an NFS share named 'llm-repo'. The share is highlighted with a red box. The configuration is as follows:

- Capacity Summary:**
 - Live data: 22.3 GB
 - Snapshot data: 0 B
 - Smart Tiering: Not Configured
 - Total Tiered Data: 0 B
- Performance Summary (Last 24 Hours):**
 - Average Throughput: 95.81 MBps
 - Total IOPS: 186
 - Average Latency: 3254µs
- Share Properties:**
 - Share Path: [/llm-repo](#)
 - Description: -
 - Mount Path: [fileserver-01.rtp4.local/llm-repo](#)
 - Primary Protocol: NFS
 - Max Size: 4 TiB
 - Multi-protocol Access: Disabled
- Features (0 of 4):**
 - Share Level Protection: Not Enabled
 - Self Service Restore: Not Enabled
 - Antivirus: Not Configured
 - Smart Tiering: Not Configured

Procedure 4. Install and Configure Nutanix Objects

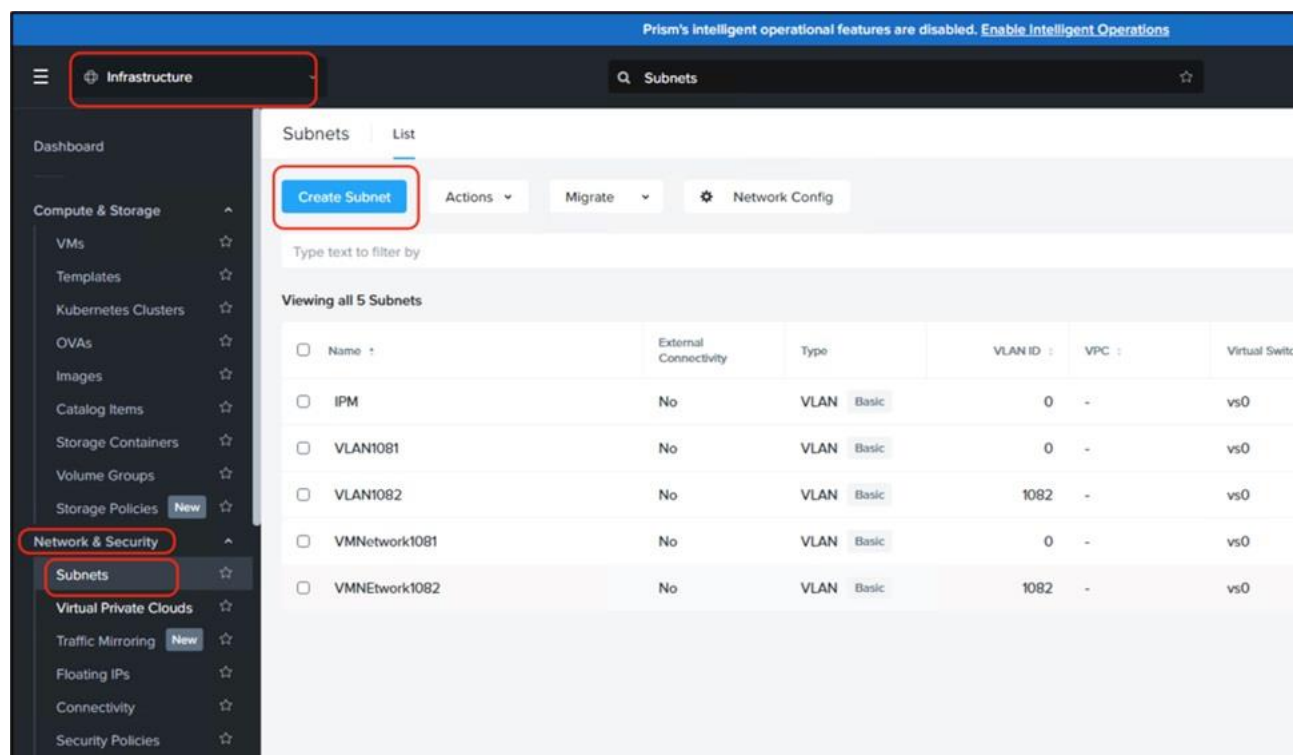
This procedure provides the high-level steps to enable and configure Nutanix Objects from Prism Central. Nutanix Files runs on the same cluster as GPT-in-a-Box cluster. For Nutanix Files architecture details go to the [Nutanix Objects User Guide](#) on Nutanix Portal.

You can also view the [Nutanix object store creation video](#) on Nutanix University.

Step 1. Log into Prism Central and ensure Name Server and NTP Server are updated.

Step 2. Prior to creating objects store, create IP Address Management (IPAM) network

Step 3. From Prism Central, navigate to Infrastructure, select Network & Security > Subnet.



Step 4. Enter Name, Type=VLAN, select GPT-in-a-Box cluster, select the virtual switch and VLAN ID = 0 , enable IP Address Management and give a range of available Ips. Click Create.

Create Subnet ✕

Name
IPM1

Type
VLAN

Cluster
M7-NTNX-AI

VLAN ID: 0 Virtual Switch: vs0

External Connectivity for VPCs No

IP Address Management

Network IP Address / Prefix: 10.108.1.0/24 Gateway IP Address: 10.108.1.254

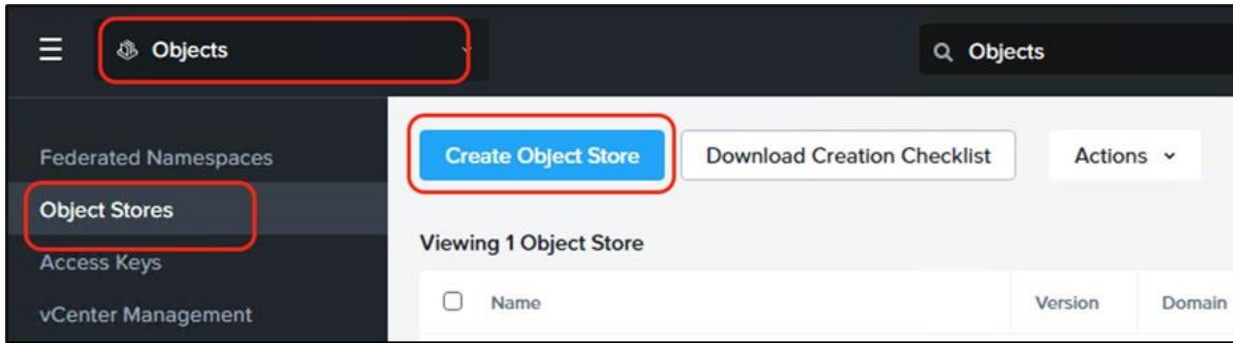
IP Pools Add IP Pool

Start Address	End Address	Actions
10.108.1.221	10.108.1.128	✎ 🗑

Cancel
Create

Note: IPAM would be utilized in subsequent section. As IP range cannot be edited, It is recommended to have several small range of IPs rather than a single large IP range.

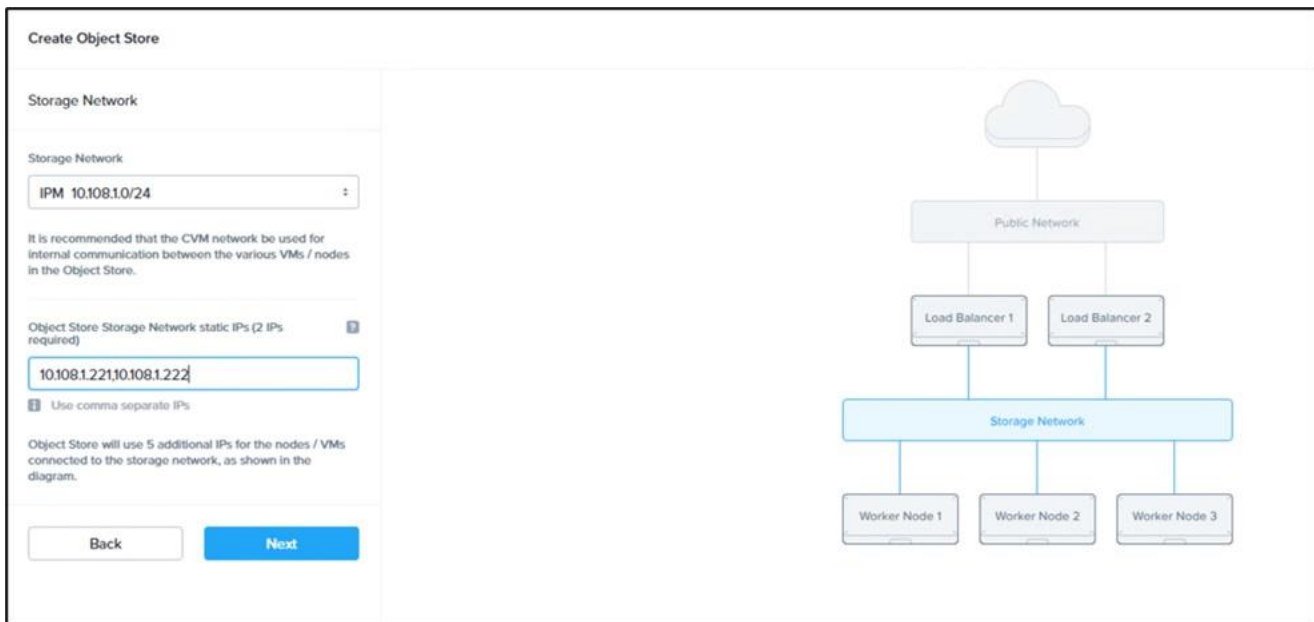
- Step 5.** Select Marketplace and search for Objects.
- Step 6.** Navigate to Objects and enable object services.
- Step 7.** Select Object Stores and click Create Object Store.



Step 8. Click Continue on the prerequisite window. From the create Object store window, enter the name of object store, select GPT-in-a-Box cluster, and edit work node size to 3. Click Next.



Step 9. Select the Storage Network. For this solution, the CVM network is used. Click Next.



Note: The IPs in the storage network should be outside the scope of IPAM DHCP configuration.

Note: In this deployment, the storage network and the client network are on the same VLAN, therefore there is a single IPAM DHCP configuration.

Step 10. Select Client Network and enter the available IPs for the client network.

Note: The IPs in the client network should be outside the scope of IPAM DHCP configuration.

Note: In this deployment, the storage network and the client network are on the same VLAN, therefore there is a single IPAM DHCP configuration.

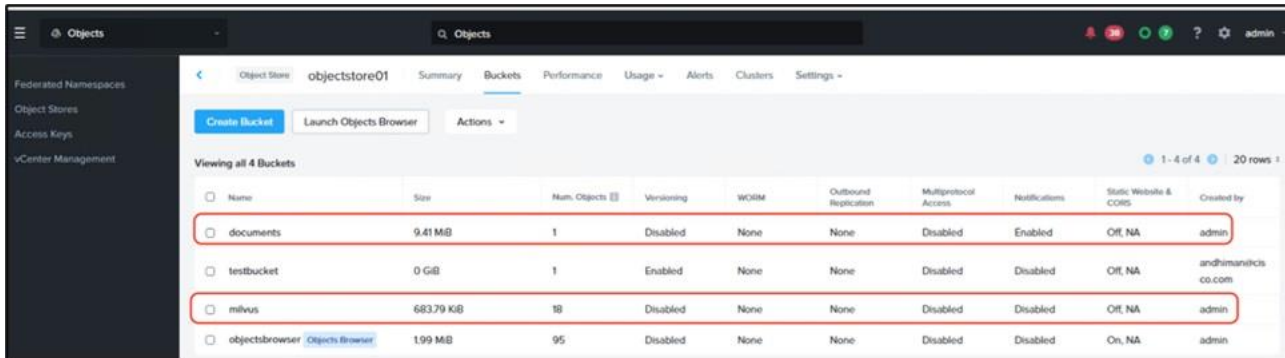
Step 11. Click Save & Continue. Allow the validation to pass, and then click Create Object Store.

Step 12. Secure the Access Key and Secret Key which is required in the solution.

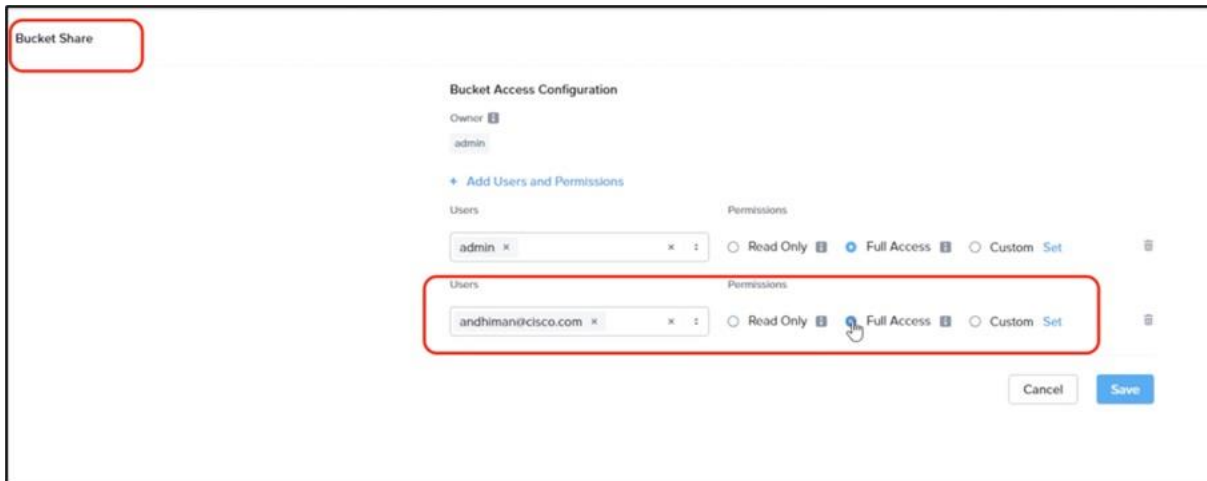


Step 13. Navigate to the created object store and click Create Bucket.

Step 14. Create two buckets 'milvus' and 'documents.' The milvus bucket is used for the milvus vector db to store the embedding and the documents bucket will store the documents and a knowledge base uploaded through the RAG reference application.



Step 15. Select the documents bucket and add Full permission to the user configured during object store creation. Similarly do the same for the milvus bucket.



Procedure 5. Install and Configure Nutanix Kubernetes Clusters

This procedure provides the high-level steps to install Kubernetes clusters through Nutanix Kubernetes Engine (NKE). GPT-in-a-Box reference design requires two clusters:

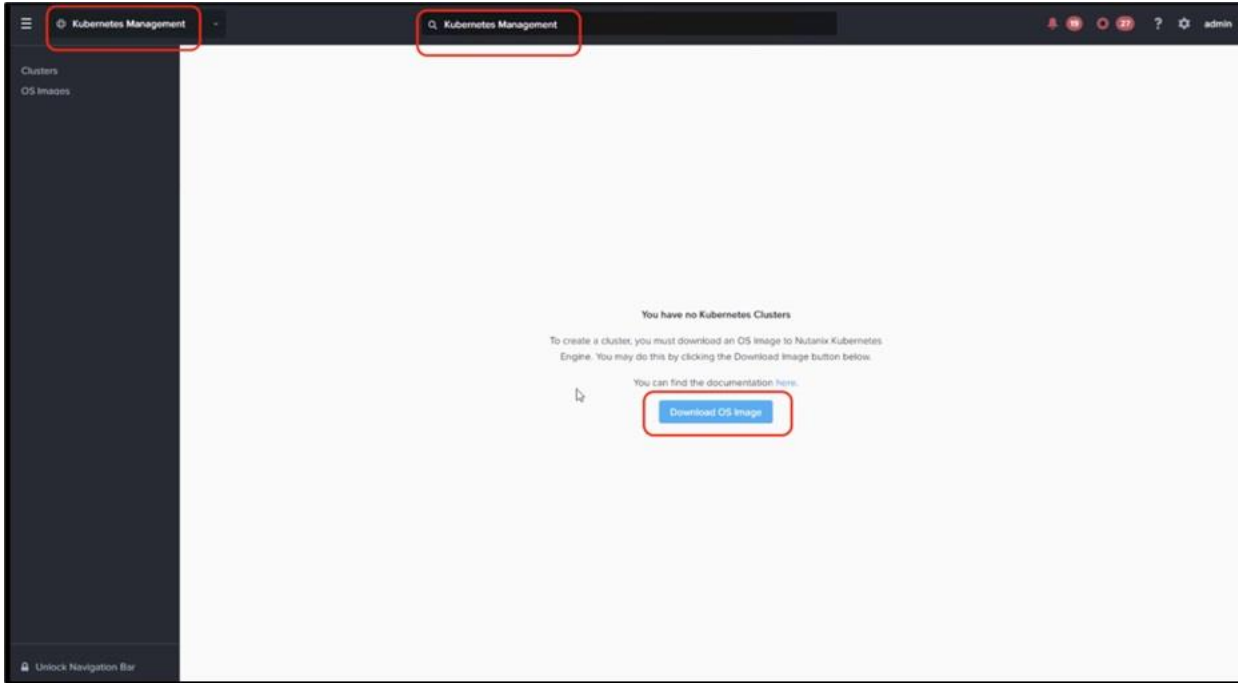
Nutanix management cluster: A single management cluster runs all components for application configuration, observability, and cross-workload cluster-persistent applications like Upttrace (an OpenTelemetry-based observability platform), Kafka, and Milvus (a cloud-native vector database that supports various data types).

Nutanix workload cluster: Workload clusters have a dedicated node pool specifically for GPU resources. This node pool hosts pods that require GPU capabilities, such as nodes with machine learning or data processing workload

Note: Prior to configuring management and workload cluster, ensure set of 2x 10 IP Pools are configured in the IPAM (IP Address Management with DHCP)

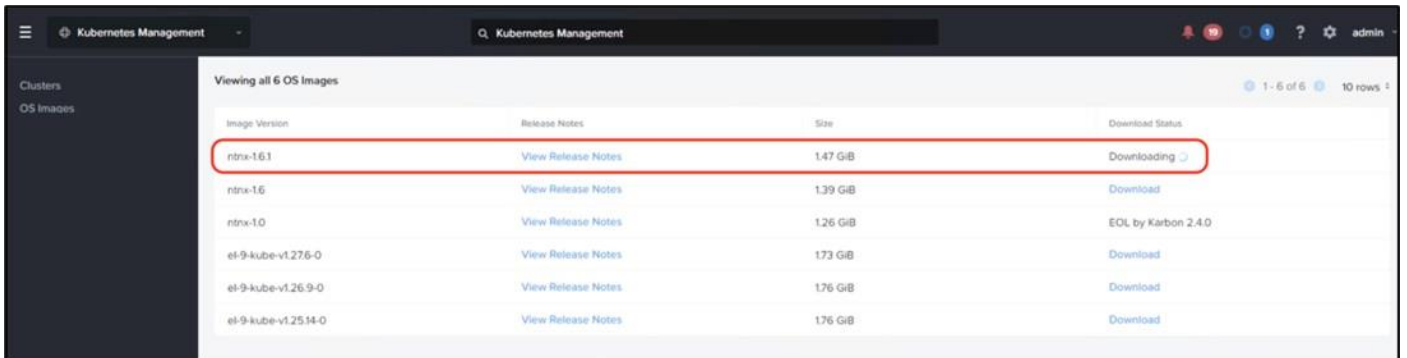
Step 1. Log into Prism Central and search for NKE in marketplace.

Step 2. Enable NKE 2.9. Select Kubernetes Management. Click Download OS Image.

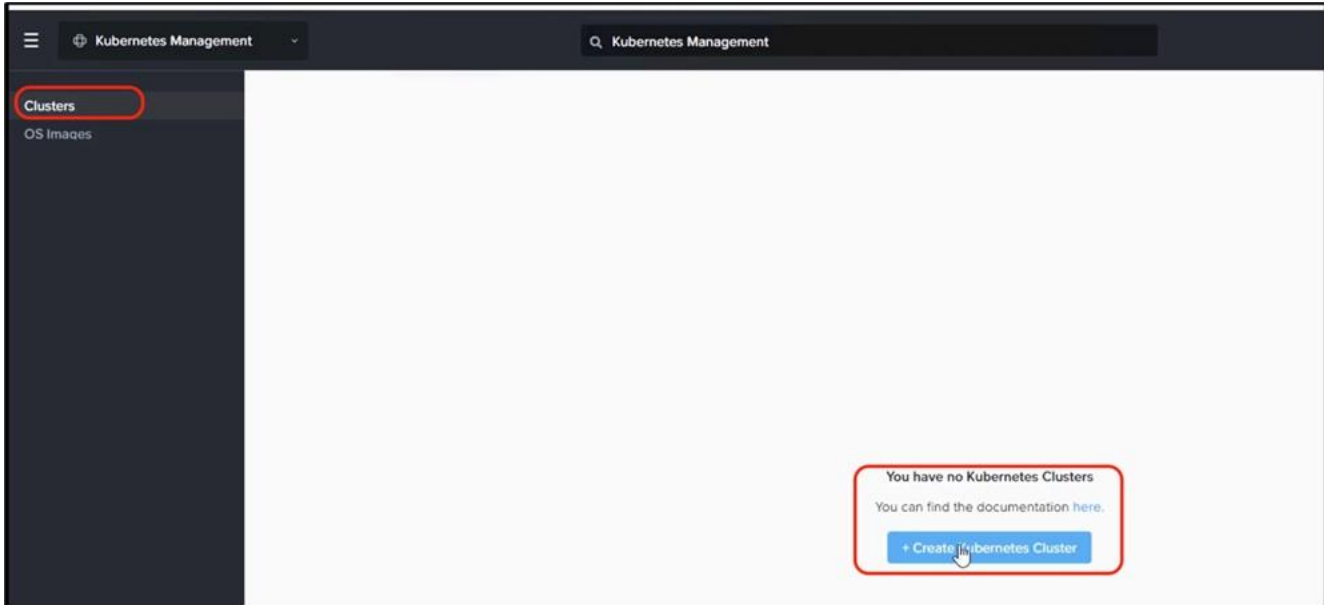


Note: In the event NKE fails to enable through marketplace, log into Prism Central and restart the cluster (genesis restart).

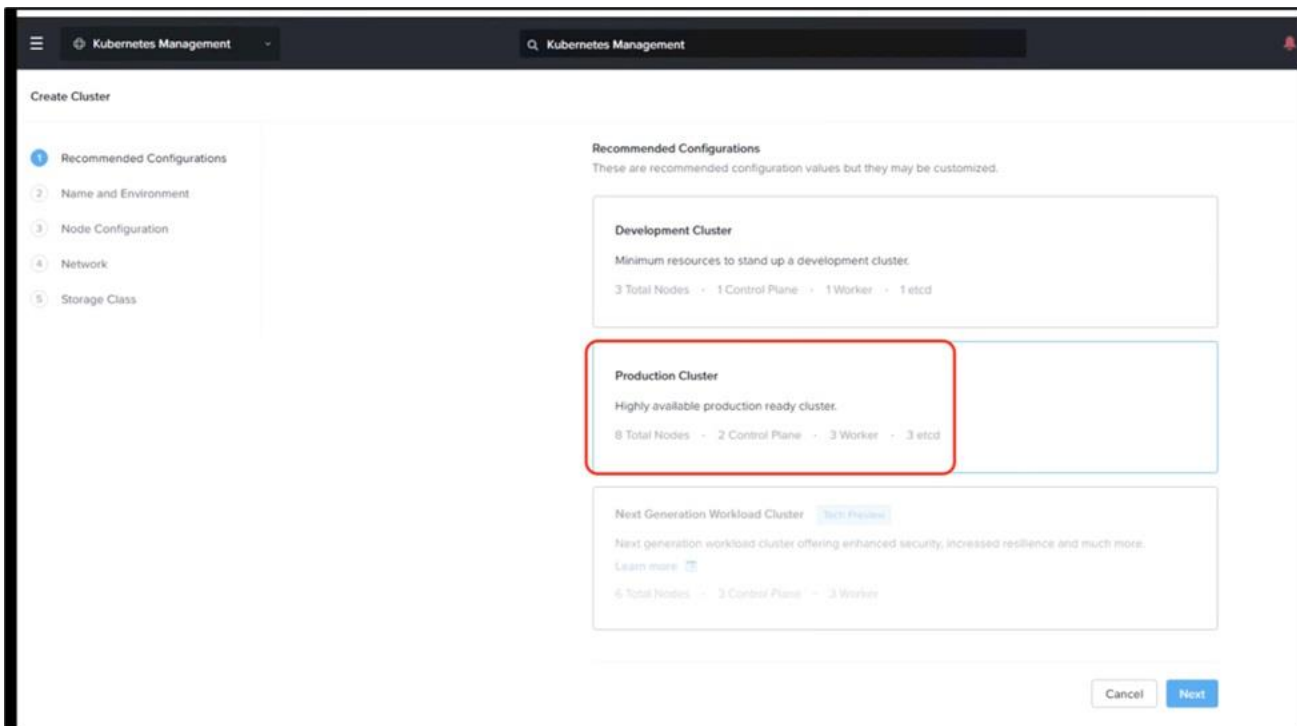
Step 3. Select and download 'ntnx-1.6.1' for the host OS image for Kubernetes cluster.



Step 4. When the image is downloaded, select the cluster from left navigation bar and click Create Kubernetes Cluster.



Step 5. Select production cluster and click Next.



Step 6. Name the cluster as per the deployment nomenclature; in this solution the clusters are name 'mtnx-k8-mgmt' and 'ntnx-k8-workload'. Click Next.

Create Cluster

1 Recommended Configurations

2 **Name and Environment**

3 Node Configuration

4 Network

5 Storage Class

Kubernetes Cluster Name: ntnx-k8-mgm

Nutanix Cluster: M7-NTNX-AI

Host OS: ntnx-16.1

Kubernetes Version: 1.26.8-0

Disable Monitoring

< Back Cancel **Next**

Step 7. On the next screen, select Kubernetes node network for the IPM (IPAM network), enter an available control plane VIP and edit the worker resource to 12 vCPU, 16 GB memory and 300GB storage size and control plane resources to 8 vCPU and 16GB memory. Click Next.

Create Cluster

3 **Node Configuration**

4 Network

5 Storage Class

IPM

Additional Network (Optional): Please select

Worker Resources

Number of Workers: 3

CPU	Memory	Size
12 vCPU	16 GB	300 GB

Control Plane Resources

Select Control Plane Resource Configuration: Multi-Control Plane: Active-Passive

Control Plane Virtual IP Address: 10.108.1.211

Number of Control Planes: 2

CPU	Memory	Size
8 vCPU	16 GB	120 GB

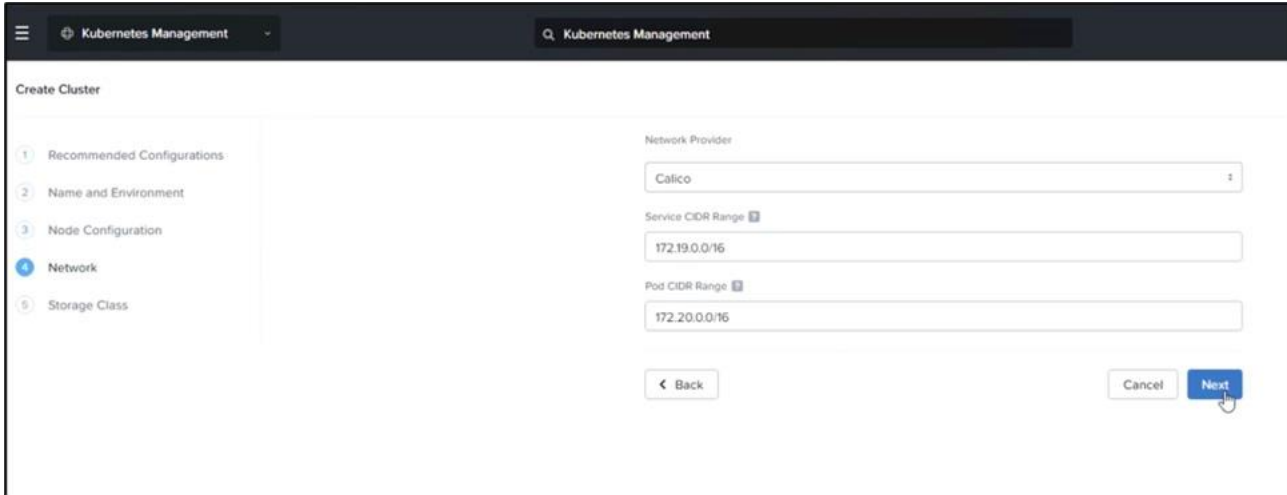
etcd Resources

Number of etcd Nodes: 3

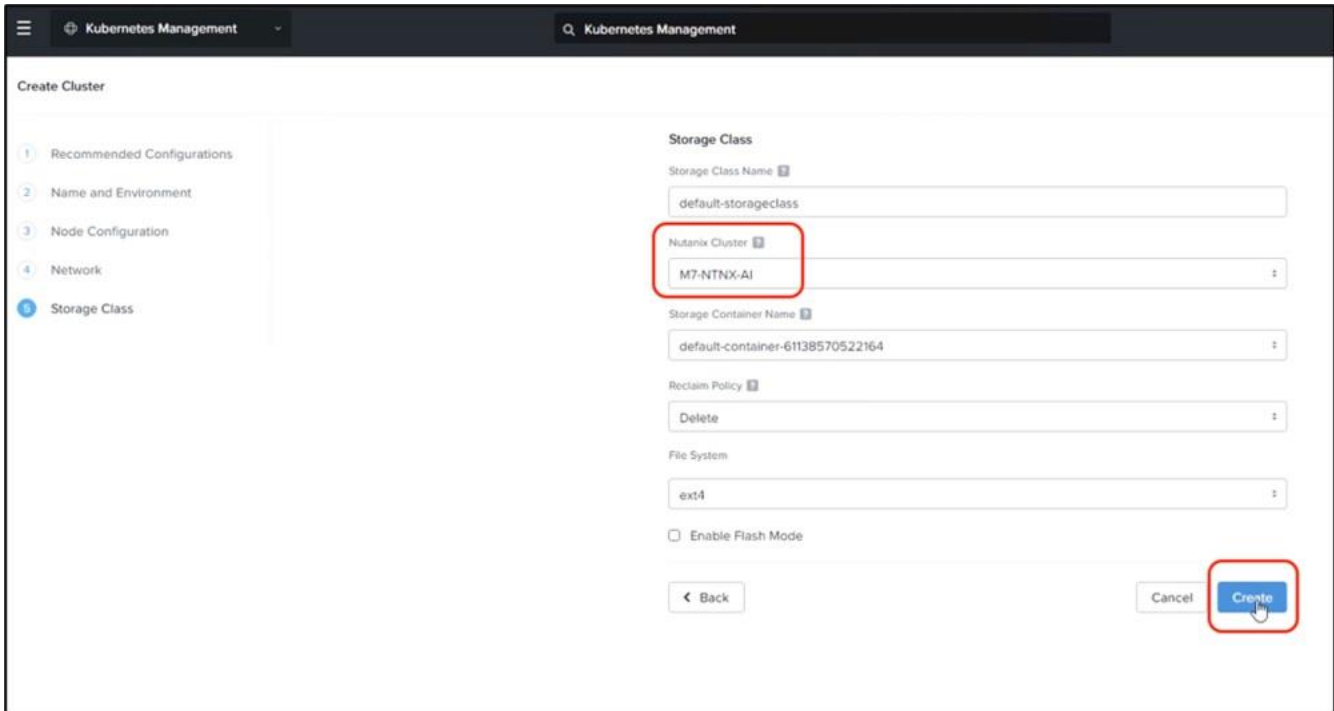
4 vCPU · 8 GB RAM · 40 GB

< Back Cancel **Next**

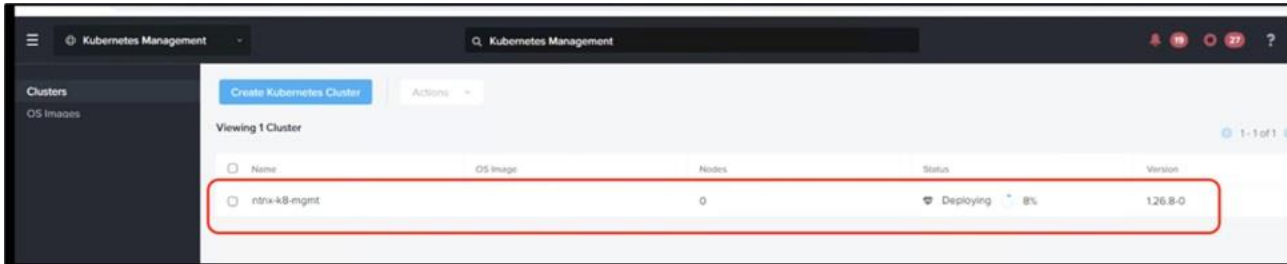
Step 8. From Network, select the defaults.



Step 9. From Storage Class, select the defaults and ensure the Nutanix Cluster is selected as the GPT-in-a-Box cluster. Click Create.



Step 10. Monitor the progress of ntnx-k8-mgmt cluster.



Step 11. Install the workload cluster by repeating steps 1 – 10.

Step 12. Create a product Kubernetes cluster and name it ntnx-k8-workload.

Create Cluster

1 Recommended Configurations

2 **Name and Environment**

3 Node Configuration

4 Network

5 Storage Class

Kubernetes Cluster Name: ntnx-k8-workload

Nutanix Cluster: M7-NTNX-AI

Host OS: ntnx-15.1

Kubernetes Version: 1.26.8-0

Disable Monitoring

[Back](#) [Cancel](#) [Next](#)

Step 13. Similar to the management cluster, select the network IPM, enter an available control plane VIP, edit the resources of workload cluster, worker resource to 12 vCPU, 16 GB memory and 300GB storage size and control plane resources to 8 vCPU and 16GB memory. Click Next.

Create Cluster

1 Recommended Configurations

2 Name and Environment

3 **Node Configuration**

4 Network

5 Storage Class

IPM

Additional Network (Optional): Please select

Worker Resources

Number of Workers: 3

CPU	Memory	Size
12 vCPU	16 GB	300 GB

Control Plane Resources

Select Control Plane Resource Configuration: Multi-Control Plane: Active-Passive

Control Plane Virtual IP Address: 10.108.1.212

Number of Control Planes: 2

CPU	Memory	Size
8 vCPU	16 GB	120 GB

etcd Resources

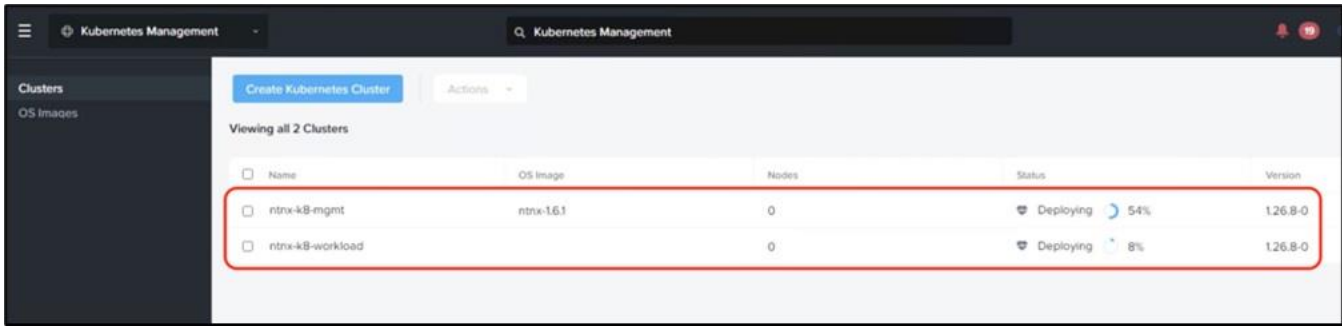
Number of etcd Nodes: 3

4 vCPU · 8 GIB RAM · 40 GIB

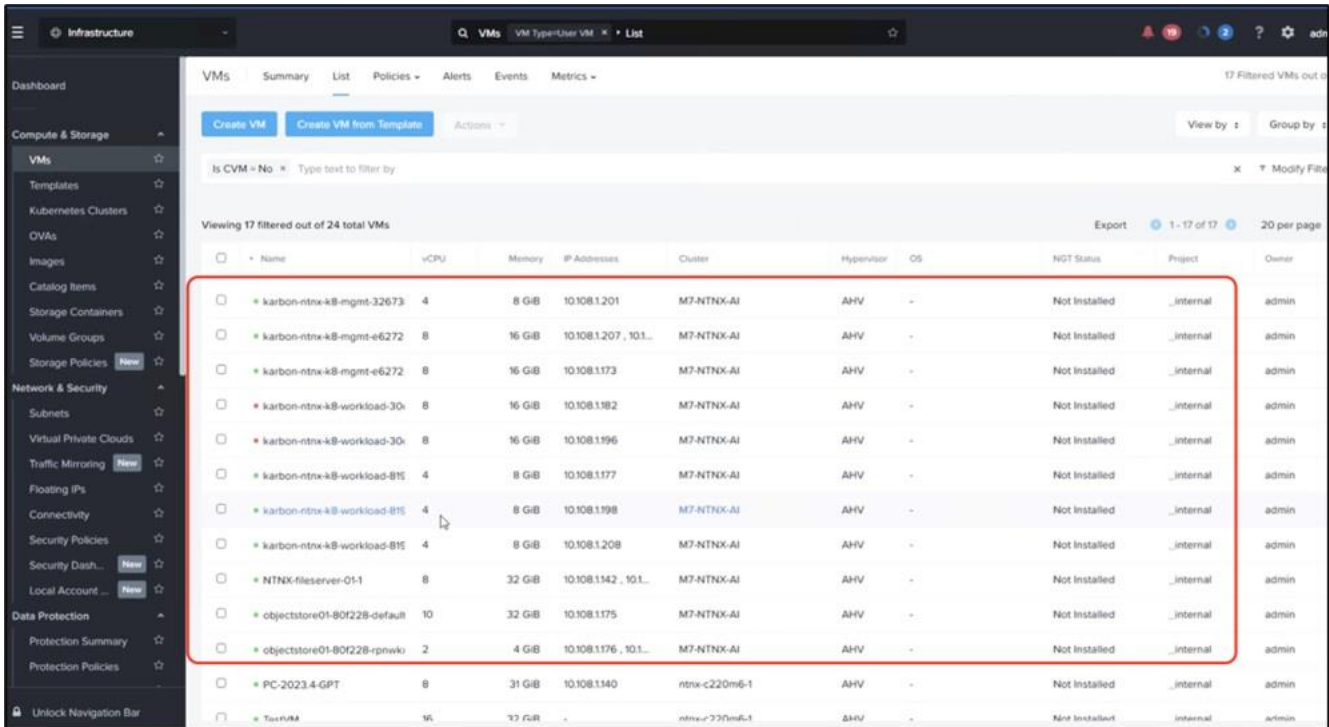
[Back](#) [Cancel](#) [Next](#)

Step 14. Select the defaults in the Network and Storage class screen. Click Create.

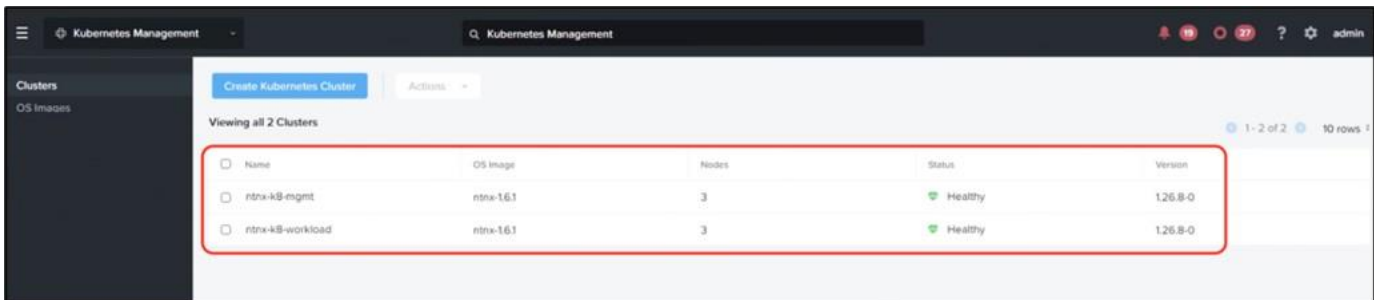
Step 15. Monitor the progress of both ntnx-k8-mgmt and ntnx-k8-workload Kubernetes clusters.



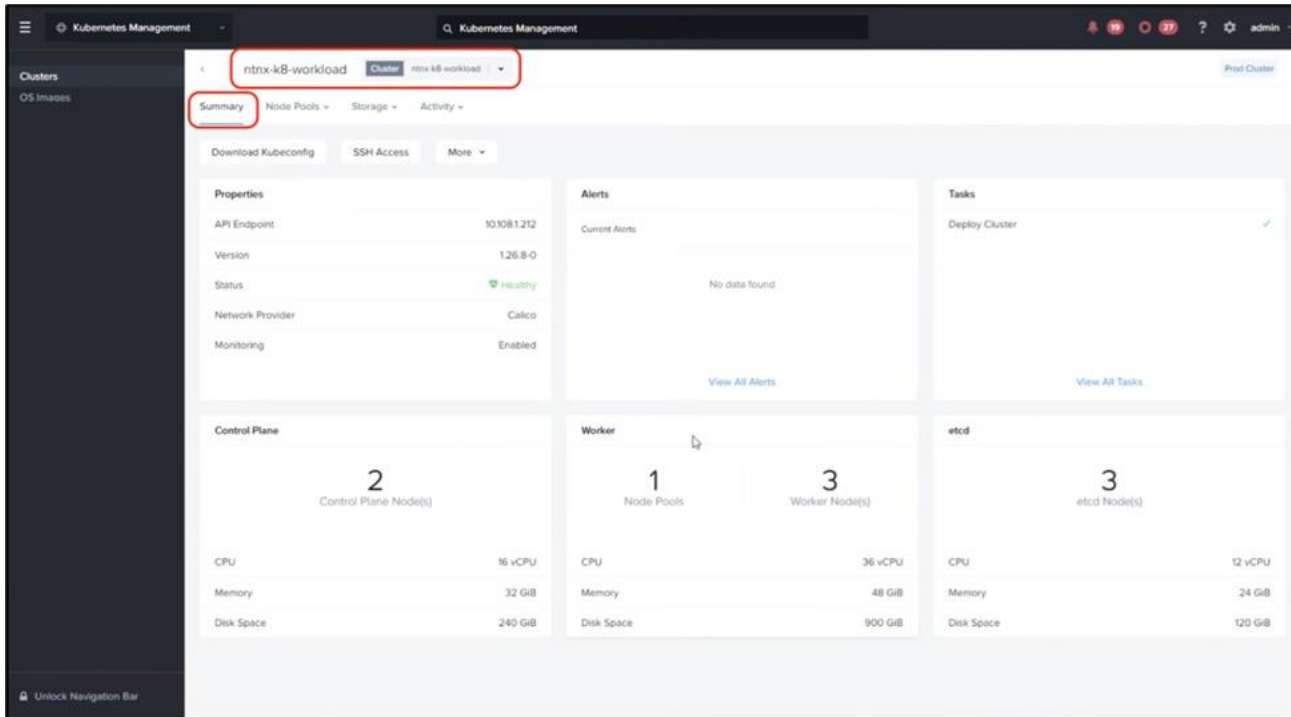
All the nodes in the Kubernetes cluster are created as VMs on the GPT-in-a-Box Nutanix cluster as shown below:



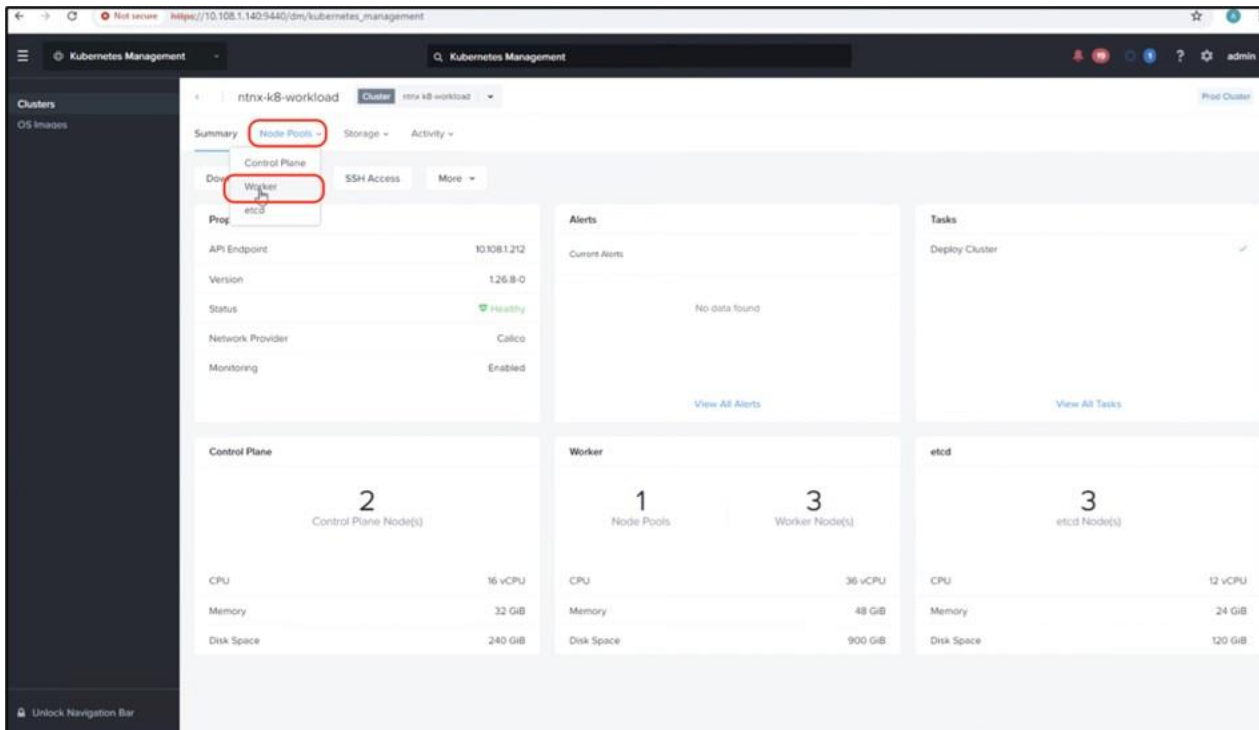
Step 16. Ensure both the Kubernetes clusters are created successfully and the health status is healthy.



Step 17. In the next few steps the gpu node pool is added to the Kubernetes workload cluster (ntnx-k8-workload). Click the ntnx-k8-workload cluster.

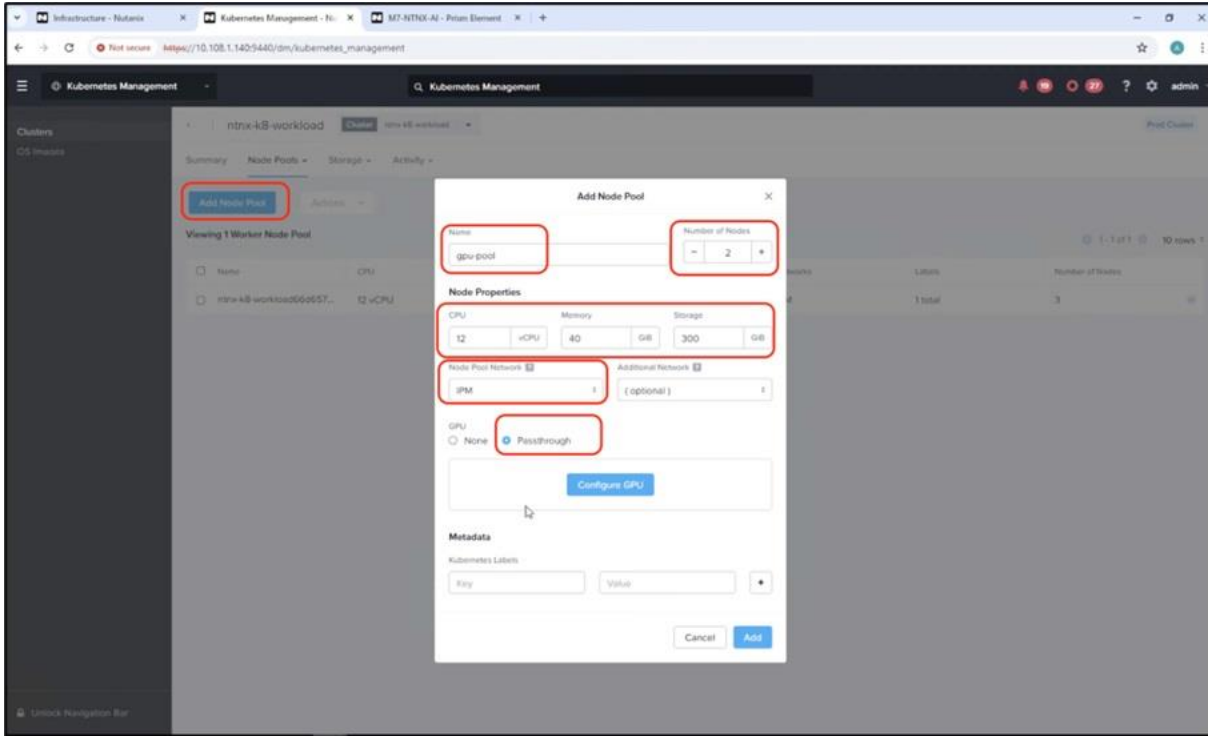


Step 18. Click Node Pools > Worker to add GPU worker nodes.

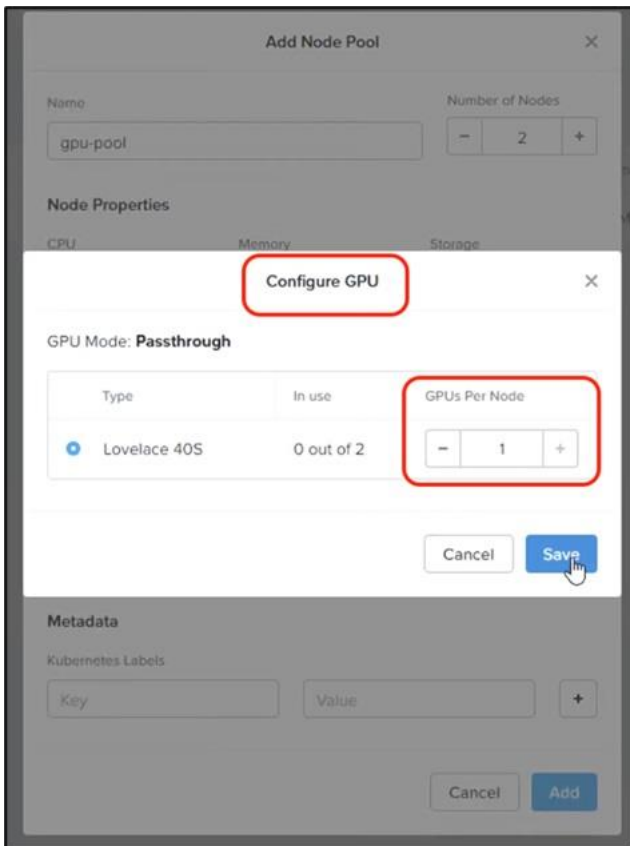


Step 19. On the worker node pool, click Add Node Pool, name the node pool (gpu-pool), select number of nodes (equal to the number of GPUs across cluster), vCPU=12, Memory=40GB, Storage=300GB, Node Pool Network - IPM (IPAM address created in the previous section), and select GPU=Passthrough.

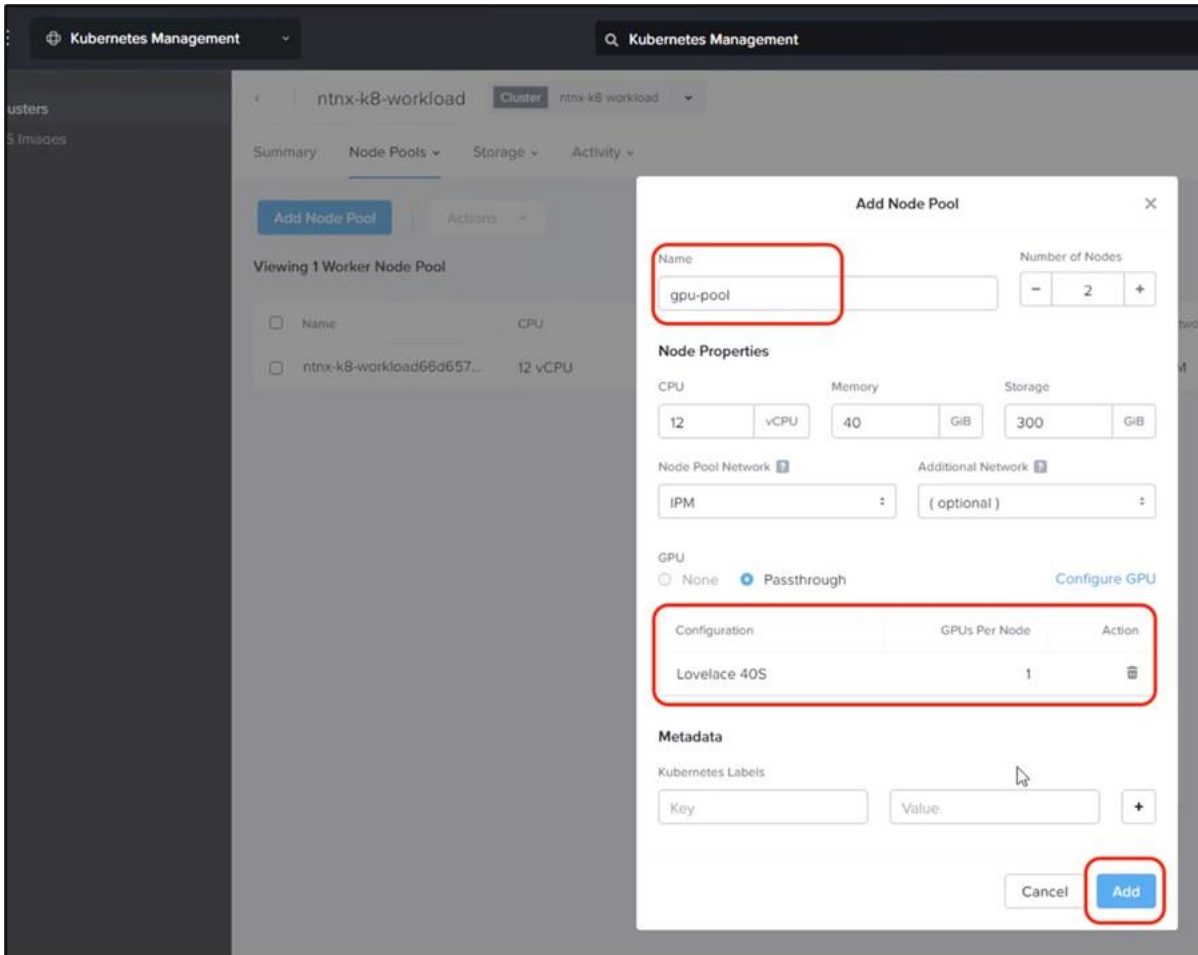
Note: In the existing configuration, there are 2 GPUs on only one of the node in 3 node CCHC with Nutanix Cluster.



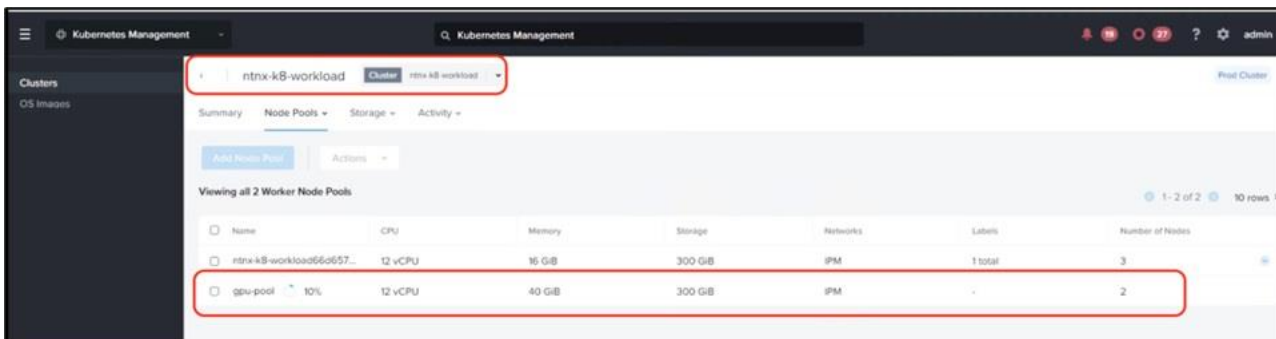
Step 20. Click Configure GPU and configure the 1GPU per node. Click Save.



Step 21. Click Add.



Step 22. Monitor the progress until completion of the GPU node pool addition to the Kubernetes workload cluster.



Deploy GPT-in-a-Box using GitOps

This section focusses on configuring GPT-in-a-Box along with the reference application.

Repository Structure

To deploy applications on top of GPT-In-Box reference architecture, the repository at <https://github.com/ucs-compute-solutions/nai-llm-fleet-infra> can be leveraged as the starting point.

The repository is designed to provide reference architecture for GPT-In-Box. It has a catalogue of multiple components and give flexibility to deploy user applications on top and enables user to easily assemble custom profile for their application. It also has a working example for a sample application.

apps

apps directory has two sub-directories under it, `nai-helm` and `gptnvd-reference-app`.

The way application is broken up is that **nai-helm** is what deploys the actual LLM endpoint and **gptnvd-reference-app** is what deploys the reference application. (Chatbot and the document ingester).

`gptnvd-reference-app` is the application built by development teams. `nai-helm` is what is fine-tuned and served by the AI Operations team.

clusters/_profiles

Profile defines what gets installed. There are multiple profiles available at `clusters/_profiles`.

_base

The base for all cluster profiles (things installed in all variants).

```
(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

Defines the Management cluster profile. It defines management specific applications and platform variants.

```
(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

Defines the Workloads cluster profile. It defines workload specific applications and platform variants.

```
(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
```

Custom profiles can be created based on the application need.

.taskfiles

Development environment has Go Task binary integrated. Task binary is a task runner / build tool which is simpler and easier to use than GNU Make.

taskfile.yaml has the logic to include different categories of tasks defines in the .taskfiles directory.

Many different helper tasks are defined in .taskfiles like flux bootstrapping, NKE cluster management, troubleshooting and so on.

Additional tasks can be created based on application and platform needs.

More information on tasks can be found here: <https://taskfile.dev/>

configs

configs is the directory for environment configuration used by local scripts.

configs/_common/.env has the global environment configuration.

scripts

Scripts directory consists of all the local scripts.

platform

Contains catalogue of all the platform services.

Customizing Environment Profiles

This document provides generic design blueprint that customers can use as a starting point to deploy in their environment. Solution gives flexibility to easily introduce the necessary changes to the profiles of both management and workload clusters based on the requirement and use case.

The repository includes directory **clusters/_profiles**. The profiles have the definitions for what gets installed in the cluster.

This reference design includes two environment profiles prod and non-prod for management and workload clusters. **_base** is the base for all the cluster profiles.

non-prod has definitions for environments like Development, QA, Staging and so on, and **prod** has definition for production deployment.

This generic definition can be customized for specific environment and use case by introducing the necessary changes to fit into your environment. You can also have totally new custom environments configured based on the requirement.

A few example customizations are provided below:

Example-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
```

In Example-1, you can see JupyterHub is enabled in the "non-prod" environment type because Jupyter will be used mostly during development. Based on the requirement it can also be configured to be available in the "prod" profile by simply copying jupyterhub-repo.yaml and jupyterhub-addons.yaml and update prod/kustomization.yaml to include Jupyter addon. and JupyterHub repository.

```

(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

```

Example-2

In Example-2, you can see that the main difference between prod and non-prod in this reference design is that prod environment is using Let's Encrypt and Amazon Route 53. If you want to integrate this with your private DNS and Certificate management, disable this component and add your customized component.

```

(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$ █

```

Example -3

In the non-prod, while configuring the GPU Operator time slicing can be enabled so that multiple applications share the GPU. Since it is not recommended for production, it is not included in the prod environment profile.

```

(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$ █

```

Based on the requirement, time slicing, and number of slices can be enabled in your environment profile.

Fork the Repository

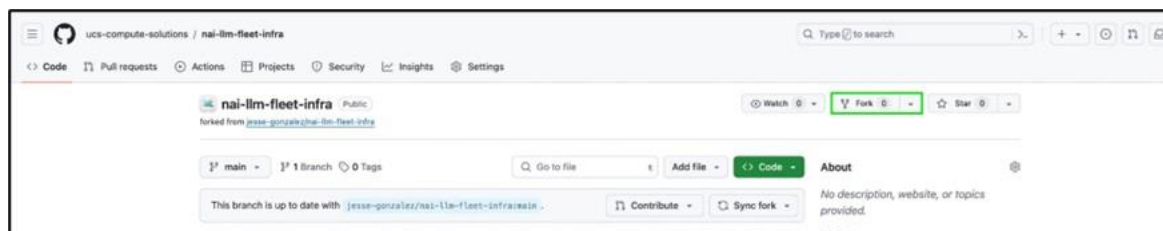
You will need to fork from original repository. A fork is a new repository that shares code and visibility settings with the original “upstream” repository. This will be customized for specific environments in the development machine.

Fork the nai-llm-fleet-infra repository on GitHub, here: <https://github.com/ucs-compute-solutions/nai-llm-fleet-infra>

Procedure 1. Fork the repository

Step 1. On GitHub.com, navigate to the ucs-compute-solutions/nai-llm-fleet-infra repository

Step 2. In the top-right corner of the page, click Fork.



Step 3. Under Owner, from the drop-down list, select an owner for the forked repository.

Step 4. By default, forks are named the same as their upstream repositories. Optionally, to further distinguish your fork, in the "Repository name" field, type a name.

Step 5. Optionally, in the Description field, type a description of your fork.

Step 6. Click Create Fork.

Clone the Forked Repository

Now you have a fork of the repository, but you do not have the files in that repository locally on your Development machine.

Procedure 1. Clone the forked repository

This procedure provides how to clone the repository so that relevant changes are done and pushed back to GitHub.

Step 1. On the development machine (Red Hat Enterprise Linux 8 host preferred), make sure git is installed.

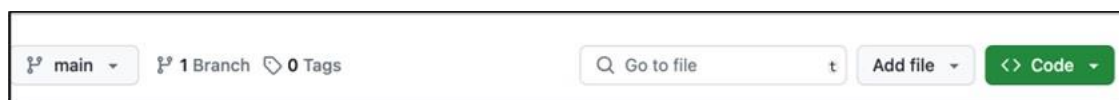
Step 2. Set up Git and authentication with GitHub.com from Git. gh is GitHub on the command line. Use gh to authenticate with GutHub. Run the following command and follow the steps:

```
gh auth login
```

For more information, go to: <https://docs.github.com/en/get-started/getting-started-with-git/set-up-git>

Step 3. From GitHub.com, navigate to your fork of the repository.

Step 4. Click <> Code.



Step 5. Copy the link.

Step 6. Change the current working directory to the location where you want the cloned directory.

Step 7. Type git clone, and then paste the URL to clone the forked copy of this repository.

```
git clone git@github.com:your-username/ nai-llm-fleet-infra.git
```

Step 8. Change the directory to cloned repository.

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

Set up Development Environment

To deploy, manage and troubleshoot GPT-In-Box along with application, we need multiple tools and libraries. It is tedious to manually install each of these tools and install specific set of libraries in the development host. Also based on the applications deployed, there might be additional binaries required.

Hence, a development environment with all the required tools for deploy, manage, and troubleshoot are required to start with. We leverage Devbox for this.

Devbox is a command-line tool that enables to easily create isolated shells for development. (Similar to venv for Python). You start by defining the list of packages required by your development environment, and devbox uses that definition to create an isolated environment for your application needs with specific tools.

Devbox is internally powered by nix which is a tool that takes a unique approach to package management and system configuration. You can use Devbox without needing to learn the Nix Language.

All the packages you need are nix packages that run like a sub-system. Devbox works similar to a package manager like yarn - except the packages it manages are at the operating-system level. With Devbox, you can install package versions from the Nix Package Registry available here: <https://www.nixhub.io/>

Procedure 1. Set up the development environment

Step 1. Run the following install script as a non-root user to install the latest version of Devbox. Accept all the defaults:

```
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$ █
```

Step 2. Start the devbox shell with access to your packages:

```
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.
█
```

Step 3. When it is run for the first time, nix will not be available. You will be prompted to install. It will also install and configures all the tools/packages listed in the devbox.json file.

Step 4. For any additional packages for your environment, update the devbox.json before starting the devbox shell.

Deploy the Application

So far two Kubernetes clusters have been deployed per the solution design requirements:

- Management Cluster: It will host the management workloads like flux, kafka, and so on.
- Workload Cluster: It hosts the dev LLM and ChatBot application - this will use GPU passed through to the kubernetes worker nodes.

Procedure 1. Create Custom Environment Files for the Management Cluster

Framework is keyed to the name of the cluster. Hence the initial configurations will be done separately for the Management cluster and the NKE Workload clusters. It is advisable to keep separate shell terminal sessions for the Management and each of the Workload cluster.

Step 1. In the shell terminal, set K8S_CLUSTER_NAME environment variable with the name of the NKE Management Cluster:

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

Step 2. copy ./env.sample.yaml to ./env.\${K8S_CLUSTER_NAME}.yaml:

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

Procedure 2. Customize the Environment File for the Management Cluster

This section explains how to customize the environment file for Management Cluster (in this example, it is .env.ntnx-k8-mgmt.yaml).

Step 1. Provide the cluster name:

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

Step 2. Set the profile name. Set this parameter with the directory name of the appropriate profile. For management it is, llm-management.

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

Step 3. Define the environment type. Within a profile, there are multiple environment types. Specify the appropriate environment type. Under ll-management, there are pod and non-prod.

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

```
environment: prod
```

Step 4. Provide the docker hub registry access details:

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

Step 5. The framework components and applications are deployed via GITOPS. So it requires GIT Hub access to synchronize. Provide the GIT Hub access details to access the created fork and push the changes:

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

Step 6. Provide the Nutanix configurations like Nutanix Prism Creds and Nutanix Objects Store Configs:

```
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>>
```

Step 7. GPT in-a-Box leverages a RAG Pipeline with Serverless Functions. You need to Vector database to store all the vector embeddings of the documents. In this example, Milvus is configured as vector store.

Step 8. Provide the bucket name which is created to store the embeddings:

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

Step 9. Next step is to configure the services. kube-vip is explicitly used for service that require Load Balancer IP addresses such as nginx and istio. nginx is explicitly used for anything that leverages Kubernetes Ingress objects such as most front-end portals/consoles and grpc backends like kafka, opentelemetry, and so on. A minimum of 2 ips should be provide in a range.

```
kube_vip:
  enabled: true
  ## Used to configure default global IPAM pool. A minimum of 2 ips should be provide in a range
  ## For Example: ipam_range: 172.20.0.22-172.20.0.23
  ipam_range: 10.108.1.214-10.108.1.216
```

Step 10. Provide the virtual IP for nginx_ingress:

```
## 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
```

Step 11. NGINX Wildcard Ingress Subdomain used for all default ingress objects created within cluster. Create a subdomain with preferable cluster name. In the sub-domain, create a host record with *. Provide the value for wildcard_ingress_subdomain.

For example, If DNS is equal to *.ntnx-k8-mgmt.rtp4.local, then value is ntnx-k8-mgmt.rtp4.local.

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

management_cluster_ingress_subdomain maps to Wildcard Ingress Subdomain for management cluster.

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

The remainder of the components will be disabled since they are required for Workload clusters.

Procedure 3. Generate and Validate the Configuration

Step 1. Install required workstation packages using the Taskfile Workstation command.

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

Step 2. Export the krew command path to the PATH:

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

Step 3. Run the Taskfile Bootstrap Command to validate and generate the cluster configuration:

```
task bootstrap:generate_cluster_configs
```

When we run this task, it:

- Stages .local/<K8S_CLUSTER_NAME>/ .env from .env.<K8S_CLUSTER_NAME>.yaml file on primary project directory
- Stages clusters/<K8S_CLUSTER_NAME>/** .yaml configs from tmp/cluster/*.tmpl templates

Step 4. Verify the generated cluster configs:

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

Procedure 4. Push the changes to gitHub

Step 1. Run the following git Operations to push the changes:

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

Procedure 5. Select the Cluster

Step 1. Run the following command to connect to the NKE cluster:

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

Step 2. The first task will prompt for existing cluster instance in local shell:


```
(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
```

Step 3. This task will download the NKE kubeconfig for selected cluster:

```
(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".
```

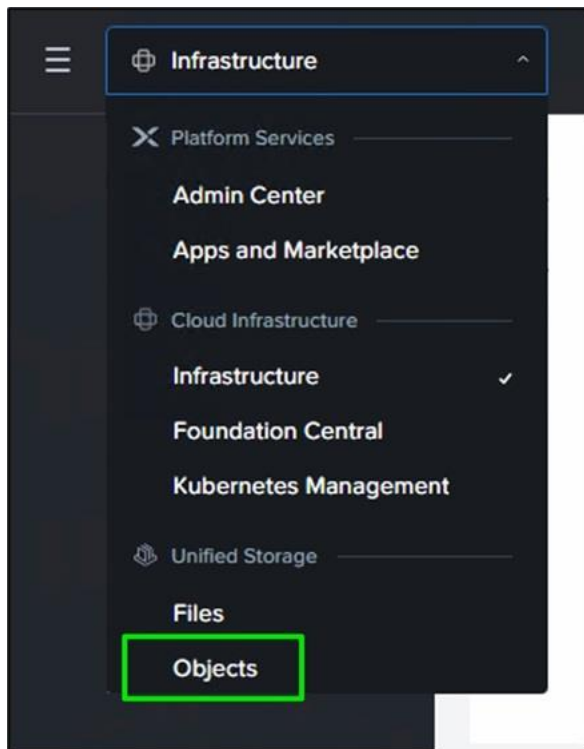
Nutanix Objects Bucket Configurations for Milvus

Buckets are logical containers which Milvus leverages to store the Vector Embeddings. We need to make sure that the bucket has appropriate permissions are set for user access. Also, Object Store must be enabled with Kafka notification endpoint.

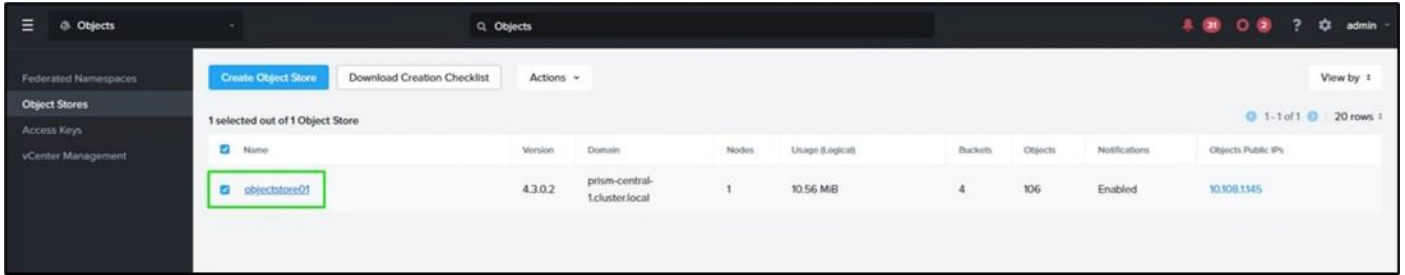
Procedure 1. Configure the Nutanix Objects Bucket for Milvus

Step 1. Log into the Prism Central web console.

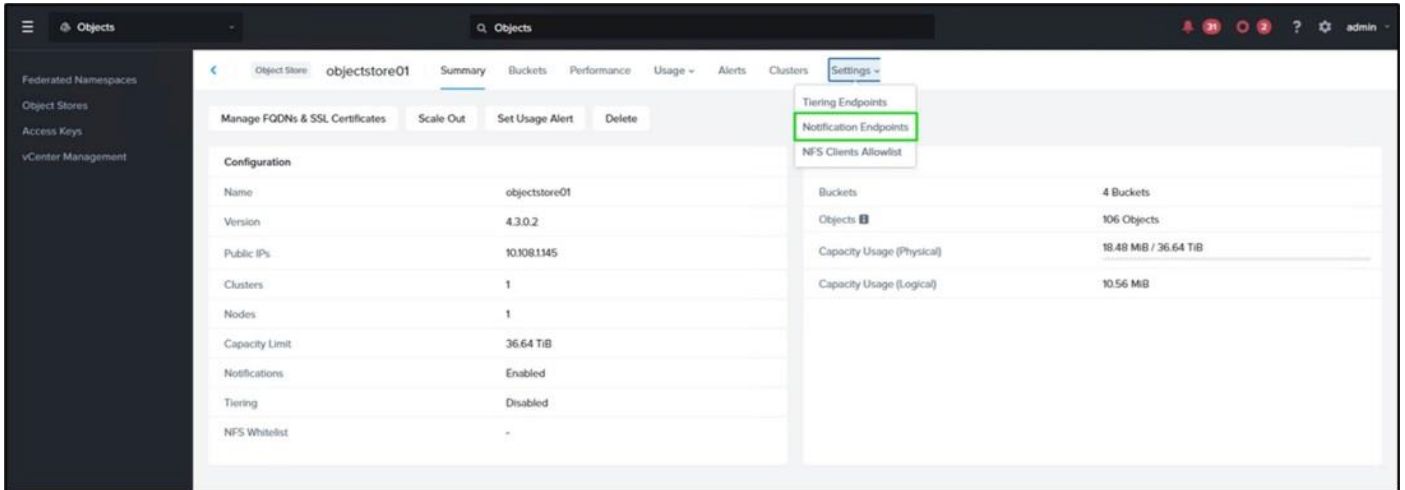
Step 2. In the Application Switcher, click Objects.



Step 3. In the Object Stores table, click the name of the object store where buckets are created for storing documents and Milvus.



Step 4. Click Settings and select Notification Endpoints.



Step 5. Select Kafka. Enable the Notification. For Object Store Events, select None.

Note: Kafka is at management cluster. So for the Host name we entered kafka. and the Wildcard Ingress Subdomain for management cluster.

Note: For port 9096 was provided. This is the GRPC port that we expose from ingress perspective.

Note: In our example, we provided the management cluster as a sub domain and the Kafka endpoint is - kafka.ntnx-k8-mgmt.rtp4.local:9096.

Notification Endpoints

Syslog Nats-Streaming **Kafka**

Enable

Host name and port

kafka.ntnx-k8-mgmt.rtp4.local:9096 × | ×

Logged Events

Object Store Events [View Events List](#)

None

Data Access Events

To configure these go to [Bucket > Data Event Notification](#)

Step 6. Click Save.

Step 7. In the Object Stores table, click the name of the object store and click the bucket configured for Milvus.

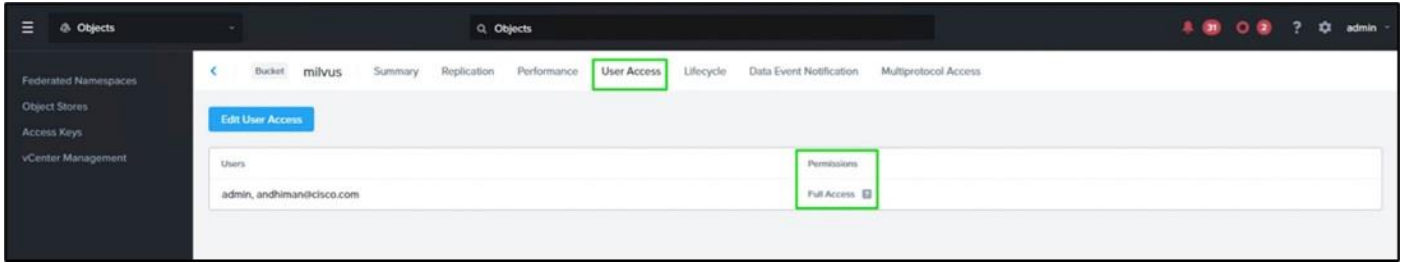
Object Stores: objectstore01 Summary **Buckets** Performance Usage Alerts Clusters Settings

Create Bucket Launch Objects Browser Actions

Viewing all 4 Buckets 1-4 of 4 20 rows

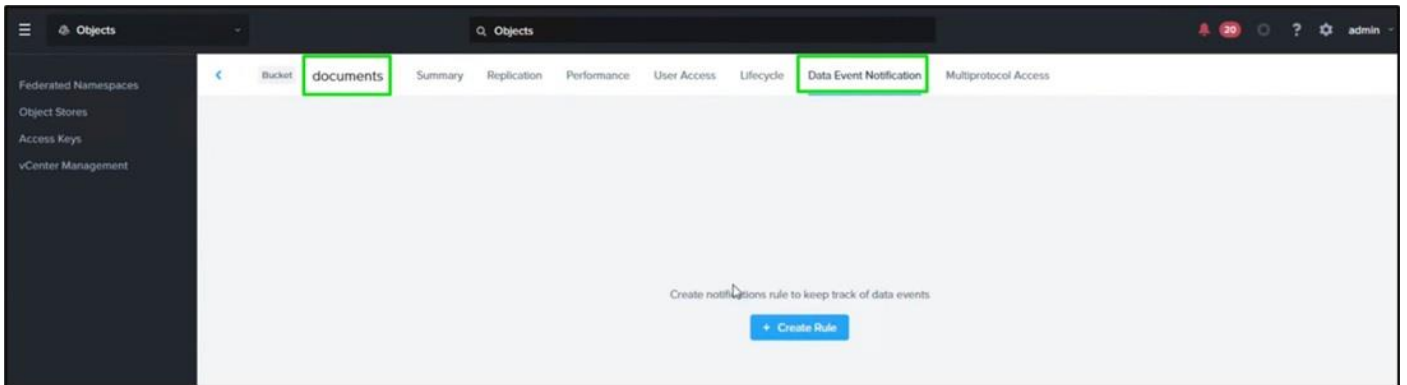
Name	Size	Num. Objects	Versioning	WORM	Outbound Replication	Multiprotocol Access	Notifications	Static Website & CORS	Created by
documents	743 MB	2	Disabled	None	None	Disabled	Enabled	Off, NA	admin
testbucket	0 GiB	1	Enabled	None	None	Disabled	Disabled	Off, NA	andhian@cisco.com
milvus	114 MB	8	Disabled	None	None	Disabled	Disabled	Off, NA	admin
objectsbrowser	199 MB	95	Disabled	None	None	Disabled	Disabled	On, NA	admin

Step 8. Navigate to User Access and make sure Full Access is configured for the user.



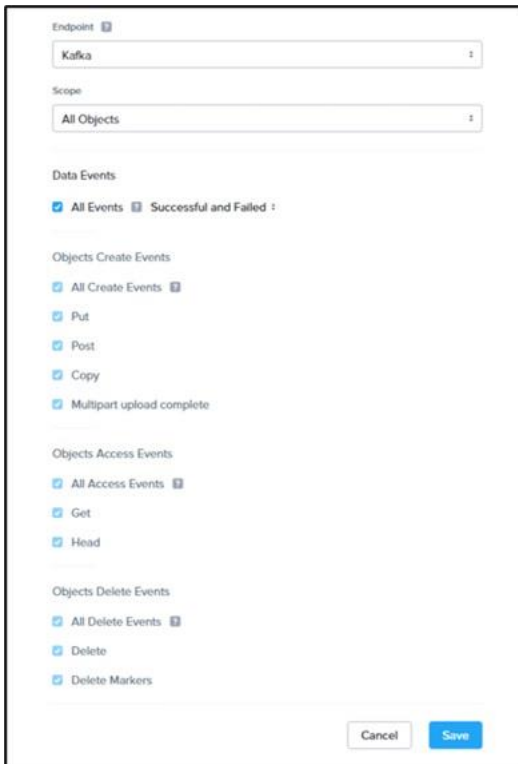
Step 9. Repeat this for the bucket that is created for storing the documents ingested as part of RAG Pipeline.

Step 10. Navigate to the bucket that is created to store the documents.



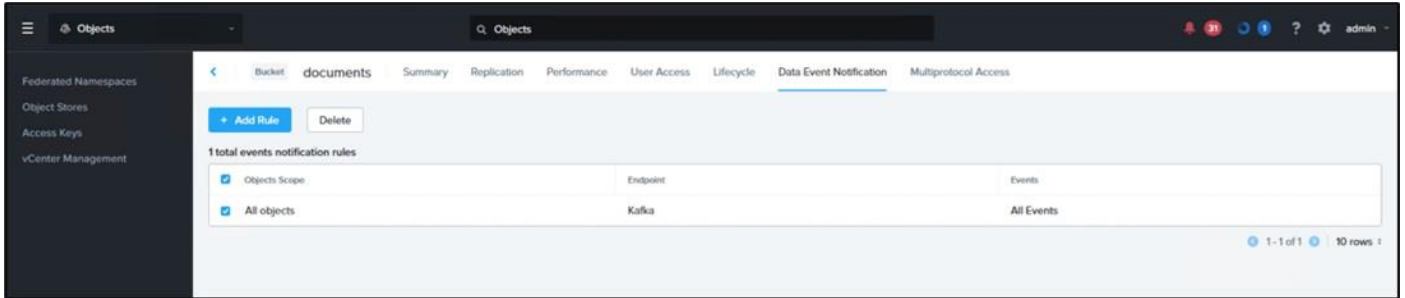
Step 11. Click Create Rule.

Step 12. In the Endpoint, select Kafka. For Scope select All Objects and click All Events for Data Events.



94xxxx

This will send a notification event to document the ingestion serverless function that is running inside knative and that will trigger Milvus for vectorization.



Procedure 2. Bootstrapping Management Cluster

After making sure cluster configuration is fine, bootstrap the Flux.

Step 1. Run the following command to bootstrap Flux:

```
task bootstrap:silent
```

Note: If there are any issues, troubleshoot using `task:flux-collect`. You can re-run the `task bootstrap:silent` as many times needed.

Step 2. Monitor the status of the installation in a new terminal by running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Management Cluster)
task flux:watch
```

Step 3. Make sure all helm charts and Kustomization resources are READY:

```
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:aca4809e15a8abb527ebcc3e2fa1f54e47c5a497bc2980634ba7e57d05d538'
helmrepository.source.toolkit.fluxcd.io/cert-manager	https://charts.jetstack.io	68d	True	stored artifact: revision 'sha256:47e4a16a8386278581fe25c168b26c1dee5e2d4ae83d57f003b55cfcfeeb'
helmrepository.source.toolkit.fluxcd.io/ingress-nginx	https://kubernetes.github.io/ingress-nginx	68d	True	stored artifact: revision 'sha256:ff73d1ebca9f738c8d0668c7755fb59c7a201b531f6024ae2634bccd83afda70'
helmrepository.source.toolkit.fluxcd.io/kyverno	https://kyverno.github.io/kyverno/	68d	True	stored artifact: revision 'sha256:a777d3ad986453bf6dde9666288ac4945df8cd33e16a1ef828d0b1290855781'
helmrepository.source.toolkit.fluxcd.io/mlvus	https://zilliztech.github.io/mlvus-helm/	68d	True	stored artifact: revision 'sha256:7c3d695be3b7fb1de47ac60b0e2f5668e535dc4513a57e5185a648839ca0931'
helmrepository.source.toolkit.fluxcd.io/opentelemetry	https://open-telemetry.github.io/opentelemetry-helm-charts	68d	True	stored artifact: revision 'sha256:a2e2993ca8d7f5e9c381453a813869e1ae50cdd2171d674999ebcd38321f9'
helmrepository.source.toolkit.fluxcd.io/uptrace	https://charts.uptrace.dev	68d	True	stored artifact: revision 'sha256:8cab55257181463380a9b7b3617a3cdf1712d29701b84f8a9df9f1b7a7fd1'
helmrepository.source.toolkit.fluxcd.io/weave-gitops	oci://ghcr.io/weaveworks/charts	68d		
NAME	AGE	READY	STATUS	
helorelease.helm.toolkit.fluxcd.io/cert-manager	68d	True	Helm install succeeded for release cert-manager/cert-manager-cert-manager.v1 with chart cert-manager@1.9.1	
helorelease.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	
helorelease.helm.toolkit.fluxcd.io/kafka	68d	True	Helm install succeeded for release kafka/kafka-kafka.v1 with chart kafka@26.8.5	
helorelease.helm.toolkit.fluxcd.io/kyverno	68d	True	Helm install succeeded for release kyverno/kyverno-kyverno.v1 with chart kyverno@3.1.4	
helorelease.helm.toolkit.fluxcd.io/mlvus-vectordb	68d	True	Helm install succeeded for release mlvus/mlvus-milvus-vectordb.v1 with chart mlvus@4.1.13	
helorelease.helm.toolkit.fluxcd.io/opentelemetry-collector-daemon	68d	True	Helm install succeeded for release opentelemetry/opentelemetry-collector-daemon.v1 with chart opentelemetry-collector@0.88.1	
helorelease.helm.toolkit.fluxcd.io/opentelemetry-operator	68d	True	Helm install succeeded for release opentelemetry/opentelemetry-operator.v1 with chart opentelemetry-operator@0.47.0	
helorelease.helm.toolkit.fluxcd.io/uptrace	68d	True	Helm upgrade succeeded for release uptrace/uptrace-uptrace.v2 with chart uptrace@1.5.7	
helorelease.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/mlvus	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	
RECONCILIERS	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	-

Step 4. If there are any issues, update local the git repository, push up changes and run the following:

```
task flux:reconcile
```

Procedure 3. Create Custom Environment Files for the Workload Clusters

Step 1. Open a new terminal. Change the directory to cloned repository:

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

Step 2. Start the devbox shell:

```
devbox shell
```

Step 3. In the shell terminal, set `K8S_CLUSTER_NAME` environment variable with the name of the NKE Workload Cluster:

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

Step 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
```

Step 5. Repeat steps 1 - 4 for all NKE Workload clusters in a separate shell terminal.

Procedure 4. Customize the Environment File for Workload Clusters

Step 1. Change the cluster name:

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

Step 2. Set the profile name. Profile defines what gets installed. There are multiple profiles available at `clusters/_profiles`.

`_base` is the base for all cluster profiles (Things installed in all variants)

`llm-management` has definition for Management cluster profile (defines management specific applications and platform variants).

`llm-workloads` has definition Workloads cluster profile (defines workload specific applications and platform variants)

Step 3. Set this parameter with the directory name of the appropriate profile. For workload it is, `llm-workloads`.

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

Step 4. Define the environment type. Within a profile, there are multiple environment types. Specify the appropriate environment type. Under `llm-workloads`, there are `pod` and `non-prod`.

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

Step 5. Provide the docker hub registry access details:

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

Step 6. GPUs are required only in the Workload cluster. Hence enable the GPU Operator. Enable time slicing only for non-production environments.

```
## 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
```

Step 7. The framework components and applications are deployed via GITOPS. This requires GIT Hub access to synchronize. Provide the GIT Hub access details to access the created fork and push the changes:

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

Step 8. Provide the Nutanix configurations like Nutanix Prism Creds and Nutanix Objects Store Configs:

```
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>>
```

Step 9. GPT in a Box leverages a RAG Pipeline with Serverless Functions. We need to Vector database to store all the vector embeddings of the documents. In this example, Milvus is configured as vector store.

Step 10. Provide the bucket name that is created to store the embeddings:

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

Step 11. Configure the services.

kube-vip is explicitly used for service that require Load Balancer IP addresses such as nginx and istio. nginx is explicitly used for anything that leverages Kubernetes Ingress objects such as most front-end portals/consols and grpc backends like kafka, opentelemetry, and so on. A minimum of 2 IPs should be provided in a range.

Since this is a new cluster, you need to provide a different set of IP addresses:

```
services:
#####
## Required variables for kube-vip and dependant 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. A minimum of 2 ips should be provide in a range
  ## For Example: ipam_range: 172.20.0.22-172.20.0.23
  ipam_range: 10.108.1.219-10.108.1.220
```

Step 12. Provide the virtual IP for nginx_ingress:

```
## 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
```

Step 13. The NGINX Wildcard Ingress Subdomain is used for all default ingress objects created within The cluster. Create a subdomain and provide a cluster name. In the sub-domain, create a host record containing *. Provide the value for wildcard_ingress_subdomain.

For example, If DNS is equal to *.ntnx-k8-workload.rtp4.local, then value is ntnx-k8-mgmt.rtp4.local

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

Step 14. The management_cluster_ingress_subdomain maps to the Wildcard Ingress Subdomain for management cluster.

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

Step 15. Provide the Virtual IP Address (VIP) dedicated for istio ingress gateway. Istio is exclusive to knative-serving which is used by LLM inferencing endpoint.

Step 16. Change the Istio Ingress Gateway - Wildcard Subdomain to the value of the Workload NGINX Wildcard Ingress Subdomain:

```
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. For Example: 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. Example: If DNS is equal to *.llm.example.com, then
value is llm.example.com
  ## 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
```

Step 17. Enable kserve, knative_serving and knative_istio:

```
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
```

Step 18. Knative Eventing is used to receive Event notifications from Nutanix Objects Document Bucket:

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

Step 19. The way application is broken up is that nai-helm is what deploys the actual LLM endpoint and gptnvd_reference_app is what deploy the chatbot and the document ingester.

Step 20. To install the reference application, set gptnvd_reference_app.enabled and specify the bucket created for documents in gptnvd_reference_app.documents_bucket_name:

```
apps:
  ## Required GPT NVD Reference Application Helm Chart Configs
  gptnvd_reference_app:
```



```
enabled: true
version: 0.2.7
documents_bucket_name: documents
```

Step 21. Configure the LLM endpoint in `nai_helm` section:

```
## 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>>
```

Model Store Options

The large language model we use must be in the Model Archive file format which will be used by TorchServe to load the model. We have two options to provide the Model Archive file to LLM.

One, provide the HuggingFace token. While pod initialization happens, initi containers will download the model directly from HuggingFace and converts to Model Archive file.

Another option is to generate the Model Archive file and keep the configured NFS.

In the example, LLM is downloaded from HuggingFace.

If you are leveraging NFS with Nutanix files to store the model, then set `useExistingNFS` to true and disable `huggingFaceToken`.

For more information about Generating Model Archive File, go to: https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/generating_mar/

In some cases, you may want to use a custom model, for example a custom fine-tuned model. The design provides the capability to generate a MAR file with custom models and start an inference server.

More information is available here: https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/custom_model/

Generate and Validate the Configuration

The next step is to validate and generate the cluster configuration for Workload clusters.

Procedure 1. Validate and generate the cluster configuration

Step 1. Run the following command:

```
task bootstrap:generate_cluster_configs
```

Step 2. Verify the generated cluster configs:

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

Procedure 2. Push the changes to gitHub

Step 1. Run the following git Operations to push the changes:

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

Procedure 3. Select the cluster

Step 1. The following command can be used connect to the NKE cluster:

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

Step 2. The first task will prompt for the existing cluster instance in local shell:

```
(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
```

Step 3. The next task will download the NKE kubeconfig for selected cluster:

```
(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]# █
```

Bootstrap the Workload NKE Clusters

After making sure cluster configuration if fine, you can bootstrap Flux.

Procedure 1. Bootstrap Flux

Step 1. Run the following command to bootstrap Flux:

```
task bootstrap:silent
```

Step 2. If there are any issues, troubleshoot using task:flux-collect. You can re-run task bootstrap:silent as many times needed.

Step 3. Monitor the status of the installation in a new terminal running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Workload Cluster)
task flux:watch
```

Step 4. Make sure all helm charts and Kustomization resources are in READY state:

```
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/naï-llm-fleet-infra.git	68d	True	stored artifact for revision 'main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7'
NAME	URL	AGE	READY	STATUS
helmrepository.source.toolkit.fluxcd.io/cert-manager	https://charts.jetstack.io	68d	True	stored artifact: revision 'sha256:47e64a16a5836278581fe25c168b26cd1dee52da6ae83d575f003b55c6feb'
helmrepository.source.toolkit.fluxcd.io/gpu-operator	https://nvidia.github.io/gpu-operator	68d	True	stored artifact: revision 'sha256:93ac1bc9b8356ae61e7a85a20fd1e38fb517746cd435db98da79628d1c679'
helmrepository.source.toolkit.fluxcd.io/ingress-nginx	https://kubernetes.github.io/ingress-nginx	68d	True	stored artifact: revision 'sha256:f73d1ebca9f738c88698c7755f059c7a201b631f6924a6234bcc033afda70'
helmrepository.source.toolkit.fluxcd.io/kyverno	https://kyverno.github.io/kyverno/	68d	True	stored artifact: revision 'sha256:a777d3ad986453bf6dde96628ac4945df8c33161e7828d9b1290855781'
helmrepository.source.toolkit.fluxcd.io/naï-llm-helm	https://jesse-gonzalez.github.io/naï-llm-helm/	58d	True	stored artifact: revision 'sha256:1467e89b34762a91290632da85dc22593742658ae80848a26fde25f951cc'
helmrepository.source.toolkit.fluxcd.io/opentelemetry	https://open-telemetry.github.io/opentelemetry-helm-charts	68d	True	stored artifact: revision 'sha256:a2e2e993cbd77e5d9c381453a051b8691ae58ccdd217dd74999ebcc33b321f9'
helmrepository.source.toolkit.fluxcd.io/weave-gtrops	oci://ghcr.io/weaveworks/charts	68d	True	
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@1.9.1	
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.v4 with chart naï-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-gtrops	68d	True	Helm upgrade succeeded for release weave-gtrops/weave-gtrops-weave-gtrops.v164 with chart weave-gtrops@4.0.36	
NAME	AGE	READY	STATUS	
kustomization.kustomize.toolkit.fluxcd.io/cert-manager	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/cert-manager-resource-configs	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/flux-system	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/gptnvd-reference-app	50d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/gpu-operator	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/ingress-nginx	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-deploy	50d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-post-deploy	50d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-eventing-pre-deploy	50d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-istio-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-istio-pre-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-post-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/knative-serving-pre-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kserve-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kserve-post-deploy	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kube-vip	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kube-vip-resource-configs	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kyverno	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/kyverno-resource-configs	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/naï-helm	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-daemon	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/opentelemetry-collector-deployment	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/weave-gtrops	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
kustomization.kustomize.toolkit.fluxcd.io/weave-gtrops-resource-configs	68d	True	Applied revision: main@sha1:d75197e219ce85a436a3cc282d815e0822c40a7	
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	921.4 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	-

Step 5. Wait for 15-20 minutes for the large language model loading to complete.

Step 6. If there are any issues, update the local git repository, push the changes to GitHub and run the following command:

```
task flux:reconcile
```

Solution Validation

This chapter contains the following:

- [Summary of Validated Models](#)
- [Access Application Components](#)
- [RAG Pipeline Validation](#)
- [Visibility and Monitoring](#)
- [Sizing Considerations](#)

This Cisco Validated Design offers a comprehensive platform for AI architects and practitioners to deploy Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box.

Summary of Validated Models

[Table 15](#) lists the Generative AI models that were validated. However, it is possible to use a custom model.

Table 15. Summary of Models

Model Name	HuggingFace Repository 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

For more information on using custom model, go to: https://opendocs.nutanix.com/gpt-in-a-box/kubernetes/v0.2/custom_model/

Access Application Components

When the bootstrapping is completed, you can access and test the LLM application.

Procedure 1. Inferencing Service validation

This procedure checks the inference service based on kserve is running as expected.

Step 1. Switch to the workload cluster running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Workload Cluster)
```

Step 2. Change the namespace to llm and check the route or look for instance of custom resource - inferencservices.serving.kserve.io:

```
kubens llm
kubectl get inferencservices.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 inferencservices.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]#
```

Step 3. Copy the URL and paste the copied URL in browser or use curl and make sure it shows the following status:

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

Procedure 2. LLM Frontend Chat Application

Step 1. Get the LLM Frontend ingress endpoint by running the following command:

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

Step 2. Ensure the output is similar to this:

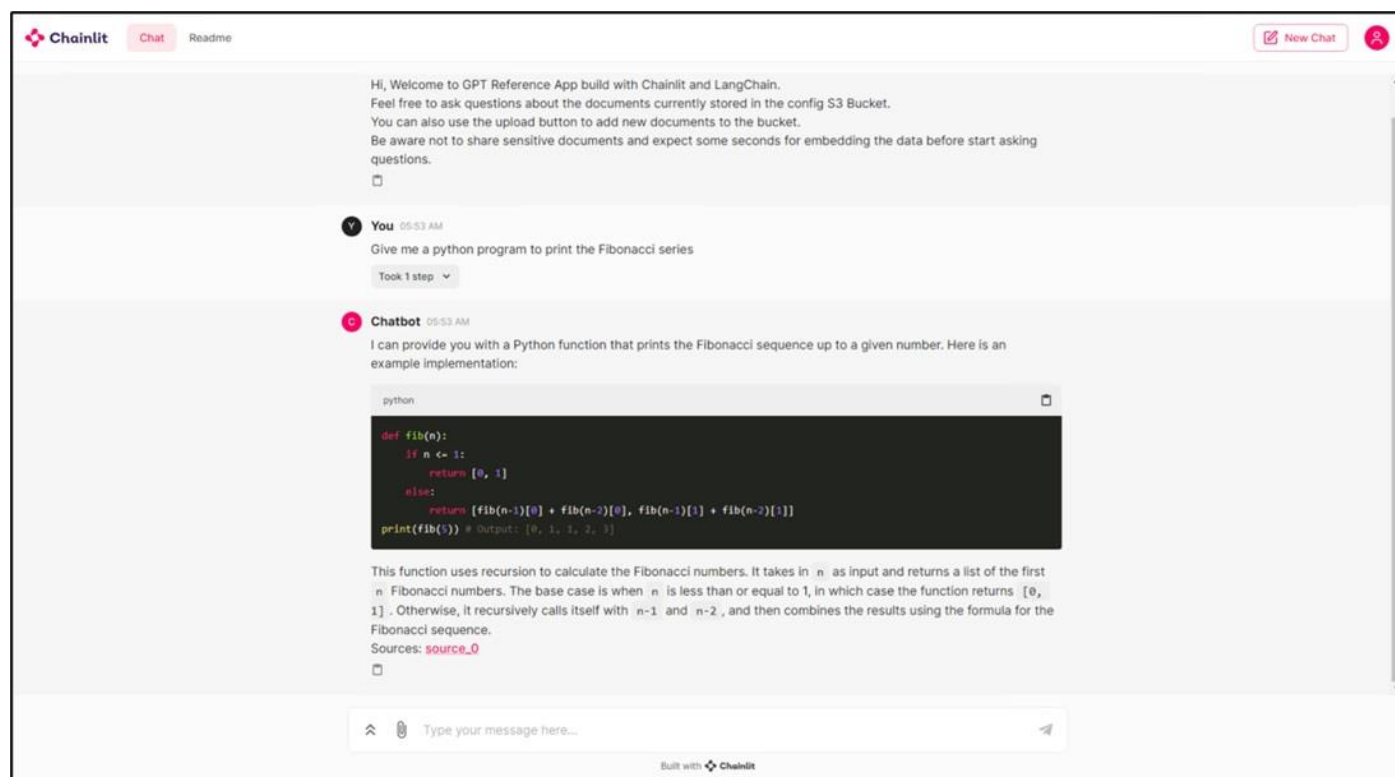
```
(devbox) [root@localhost nai-llm-fleet-infra]# kubectl get ingress -n gptnvd-reference-app
NAME CLASS HOSTS ADDRESS PORTS AGE
gptnvd-reference-app-gptnvd-ref-app-gptnvd-referenceapp nginx frontend.ntnx-k8-workload.rtp4.local 10.108.1.217 80, 443 39h
(devbox) [root@localhost nai-llm-fleet-infra]#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

Step 3. Copy the HOSTS address `frontend.ntnx-k8-workload.rtp4.local` from the above output and paste it in your browser. You should be able to see the LLM chat interface.

100XXXX

Procedure 3. Test LLM Frontend Chat

Step 1. Type any question in the chat box. For example, give me a python program to print the Fibonacci series?



Procedure 4. Access JupyterHub

JupyterHub brings the power of notebooks to groups of users. It gives users access to computational environments and resources without burdening the users with installation and maintenance tasks.

In the reference repository, JupyterHub is enabled in the "non-prod" environment type. It can be in the prod environment as well if required. jupyterhub-repo.yaml and jupyterhub-addons.yaml need to be copied to prod directory and include jupyterhub-addons.yaml in the prod/kustomization.yaml.

The design also gives flexibility to create custom environment configurations based on the requirement and included as part of your workload cluster.

Step 1. Get the jupyterhub ingress endpoint by running the following command:

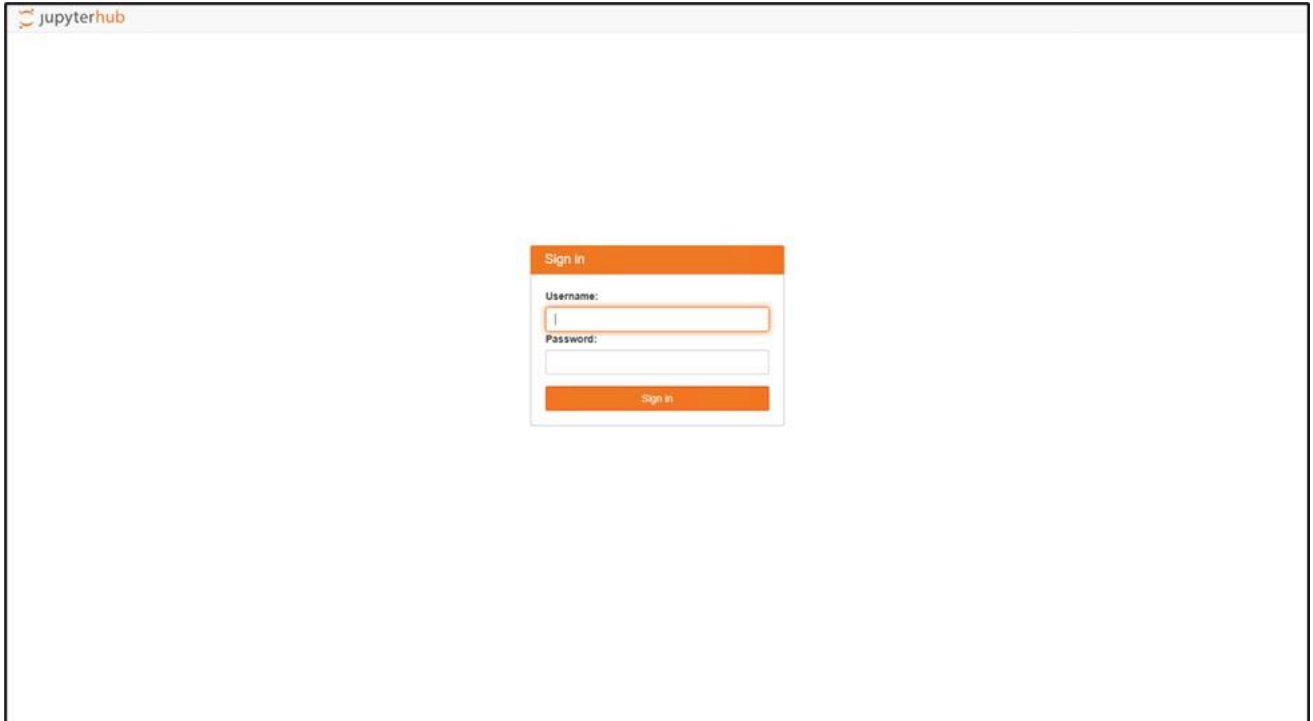
```
kubectl get ingress -n jupyterhub
```

Step 2. Ensure the output is similar to this:

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
jupyterhub	nginx	jupyter.ntnx-k8-workload.rtp4.local	10.108.1.217	80, 443	24h

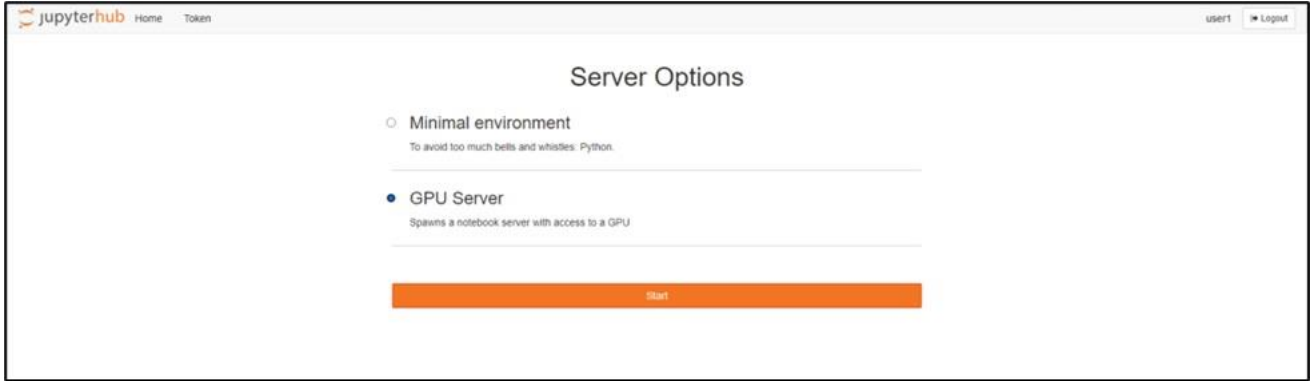
(devbox) [root@localhost llm-workloads]#

Step 3. Copy the HOSTS address jupyter.ntnx-k8-workload.rtp4.local from the above output and paste it in your browser. You should be able to see JupyterHub login page.

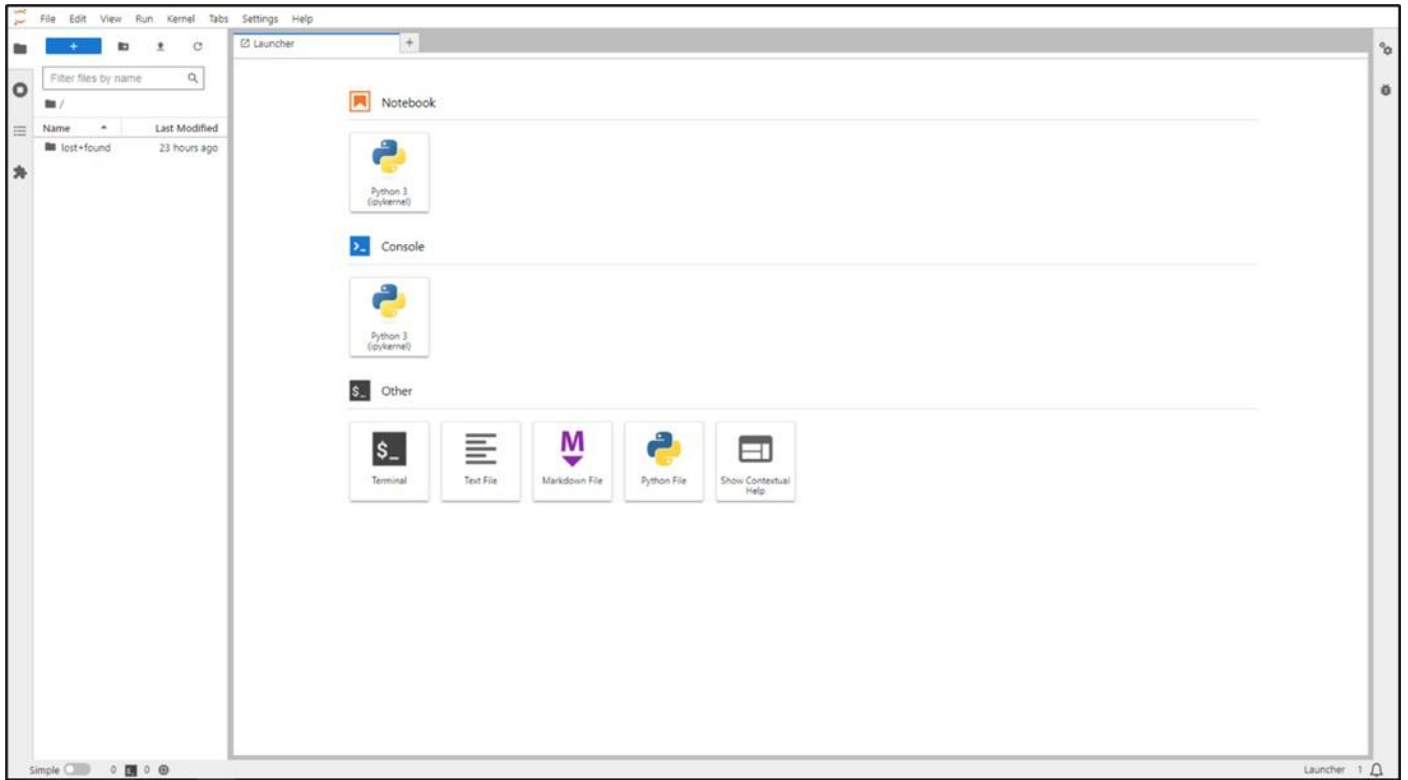


Step 4. Four users (user1,user2,user3 and user4) and configured with default password nutanix.1. It can be customized in jupyterhub.yaml file present in platform/jupyterhub/_operators directory while bootstrapping cluster.

Step 5. Input the username and password. Select GPU server to spawn a notebook with access to GPU



Step 6. Access the Jupyter Notebook.



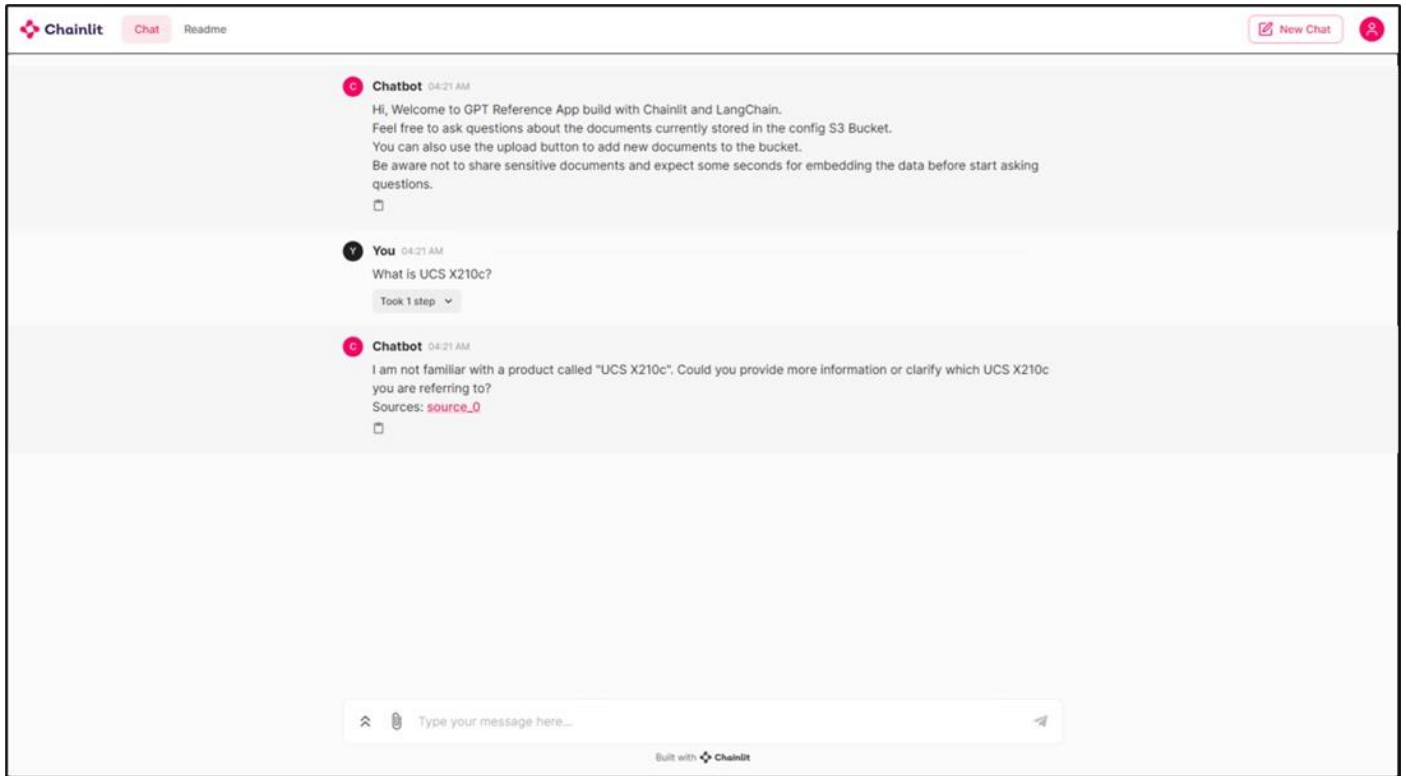
RAG Pipeline Validation

This example showcases RAG pipeline.

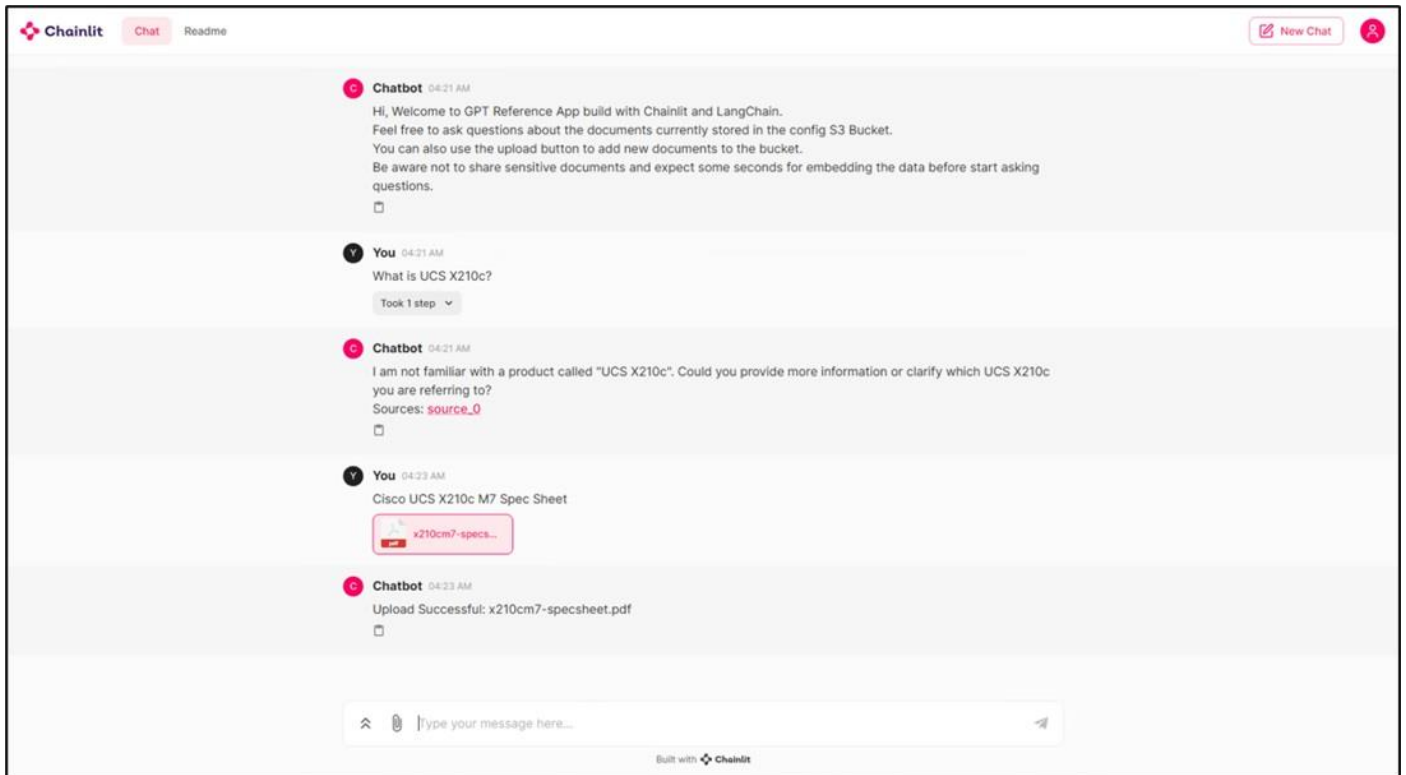
Procedure 1. Validate the RAG pipeline

Step 1. Ask a question which is internal to the organization of the information that was not available when the model was trained on. For example, What is UCS X210C?

Step 2. Observe that LLM responds saying it doesn't have the information or is providing the hallucinated response.

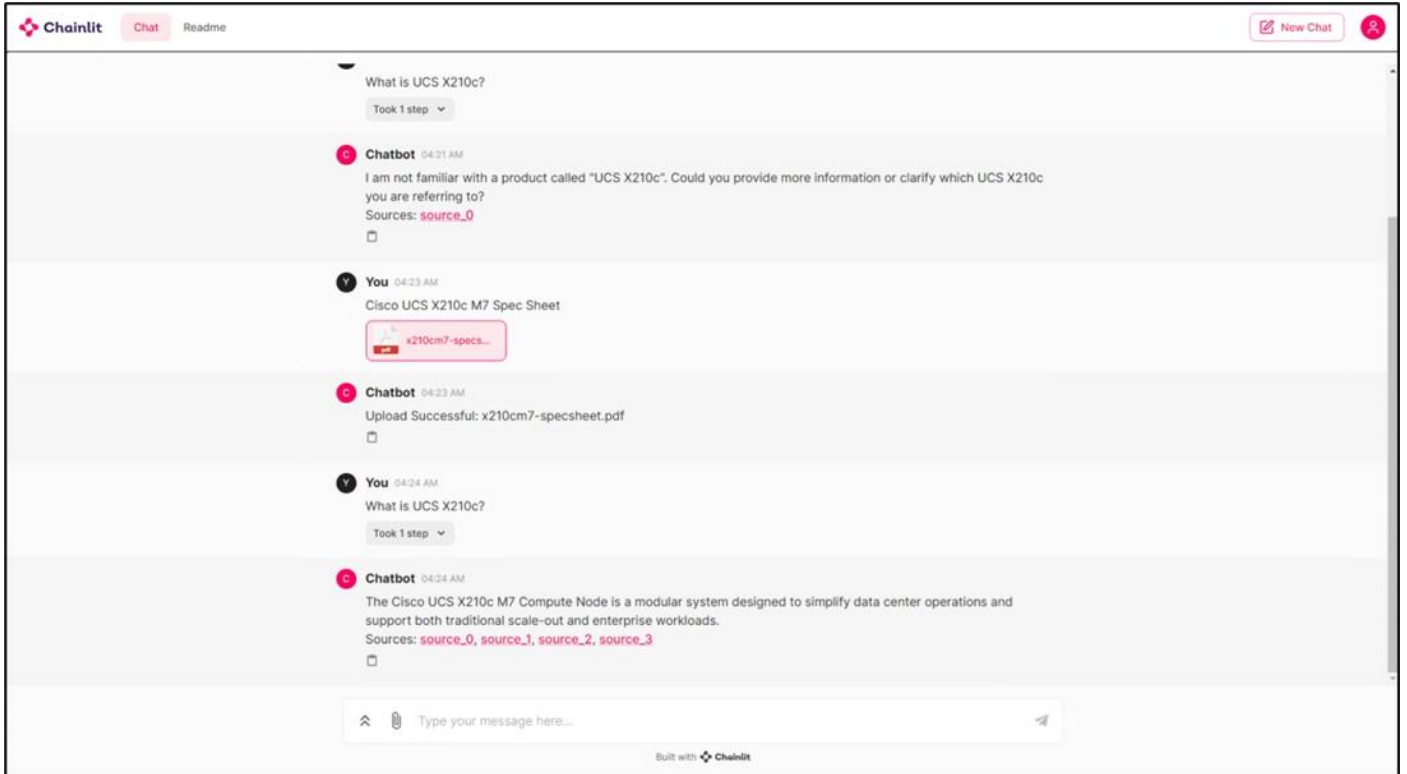


Step 3. Upload the Cisco UCS X210c M7 Spec Sheet:

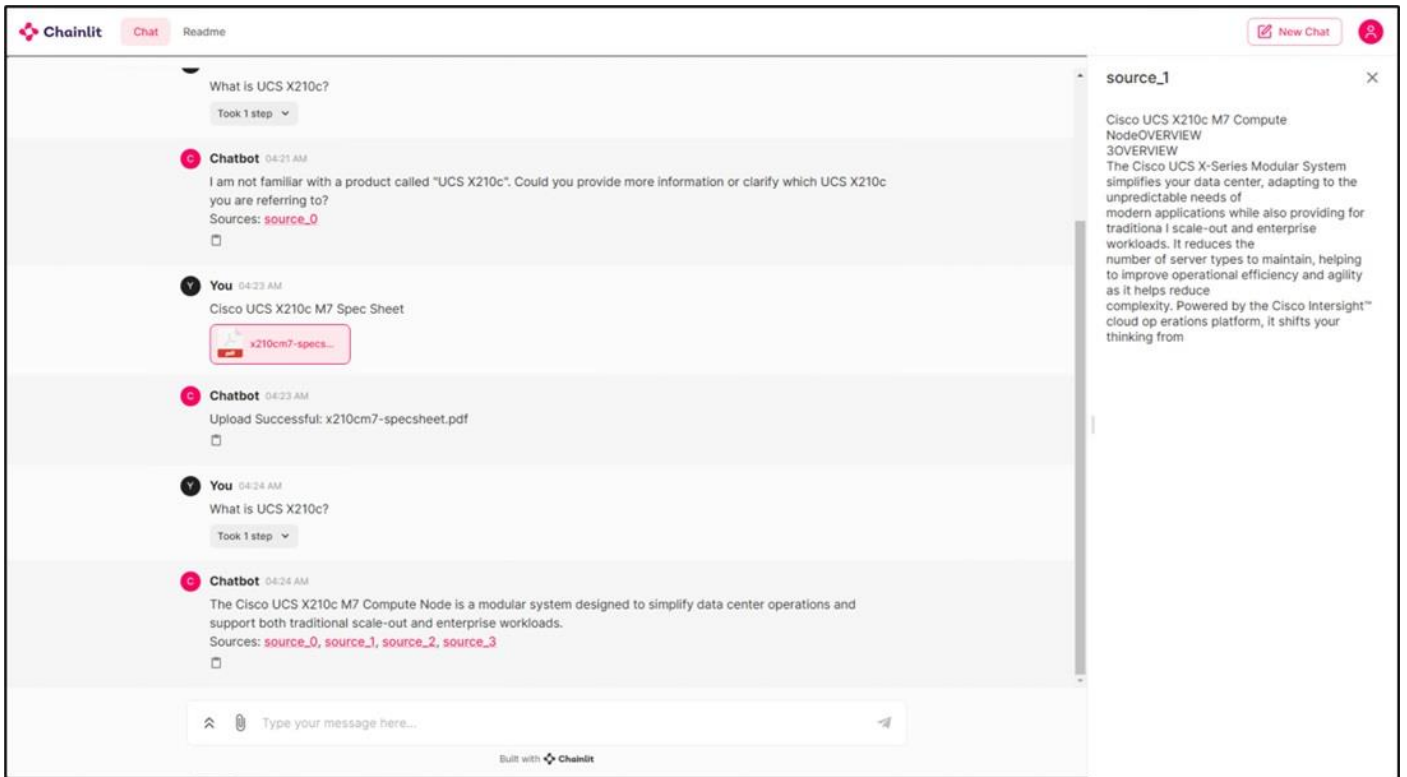


Step 4. When the upload is complete, we re-execute the same query. LLM augments the relevant chunk of from the documents and passes as context to the query and produces the response.

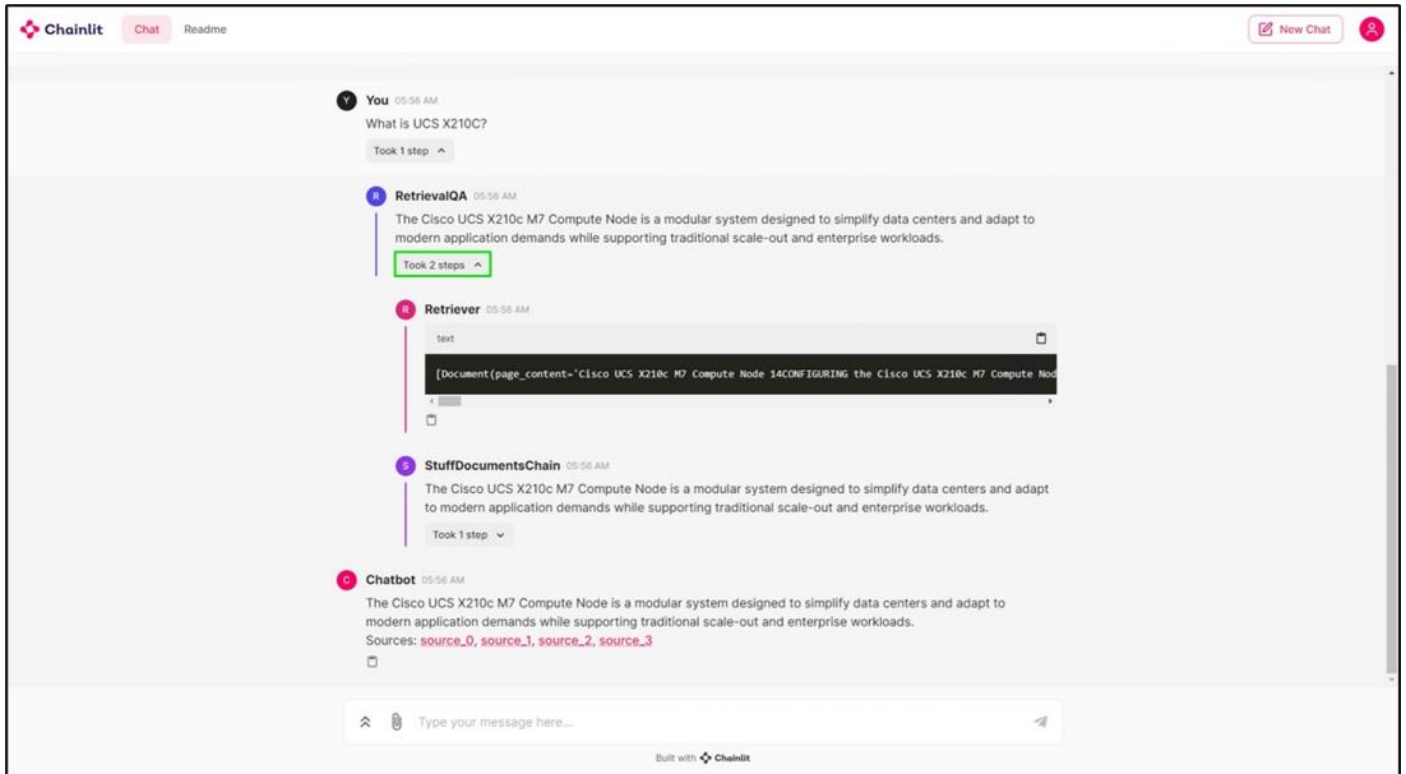
In the example below, the relevant information about UCS X210c is provided:



Step 5. You can also click each source and observe the chunks retrieved to answer the query:



Step 6. Observe that the output generation task is a 2 step process. In the first step, retriever searches and fetches information relevant to the prompt. The retrieved relevant information is augmented to the prompt as context. LLM is asked to generate response to the prompt in the context and User receives the response.



110XXX

Visibility and Monitoring

It is critical to gain complete visibility into the entire stack, including Physical infrastructure [Compute, Storage and Network], Virtualized infrastructure, NKE clusters and the Application stack. It helps to gain insight into infrastructure bottlenecks and factors that increase costs, and ensure the performance for model inferencing and applications making use of it.

This section provides some ways of gaining visibility of the application stack.

Application Insights

The solution is integrated with Weave GitOps. Weave GitOps is an extension to Flux. It provides insights into your application deployments and makes continuous delivery with GitOps easier to adopt and scale.

Weave GitOps' intuitive user interface surfaces key information to help application operators easily discover and resolve issues—simplifying and scaling adoption of GitOps and continuous delivery. The UI provides a guided experience that helps you to easily discover the relationships between Flux objects and build understanding while providing insights into application deployments.

Procedure 1. Access the UI guided experience

Step 1. weavegitops is running on Management cluster. Switch to the management cluster by running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Management Cluster)
```

Step 2. Change the namespace to weave-gitops:

```
kubens weave-gitops
```

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

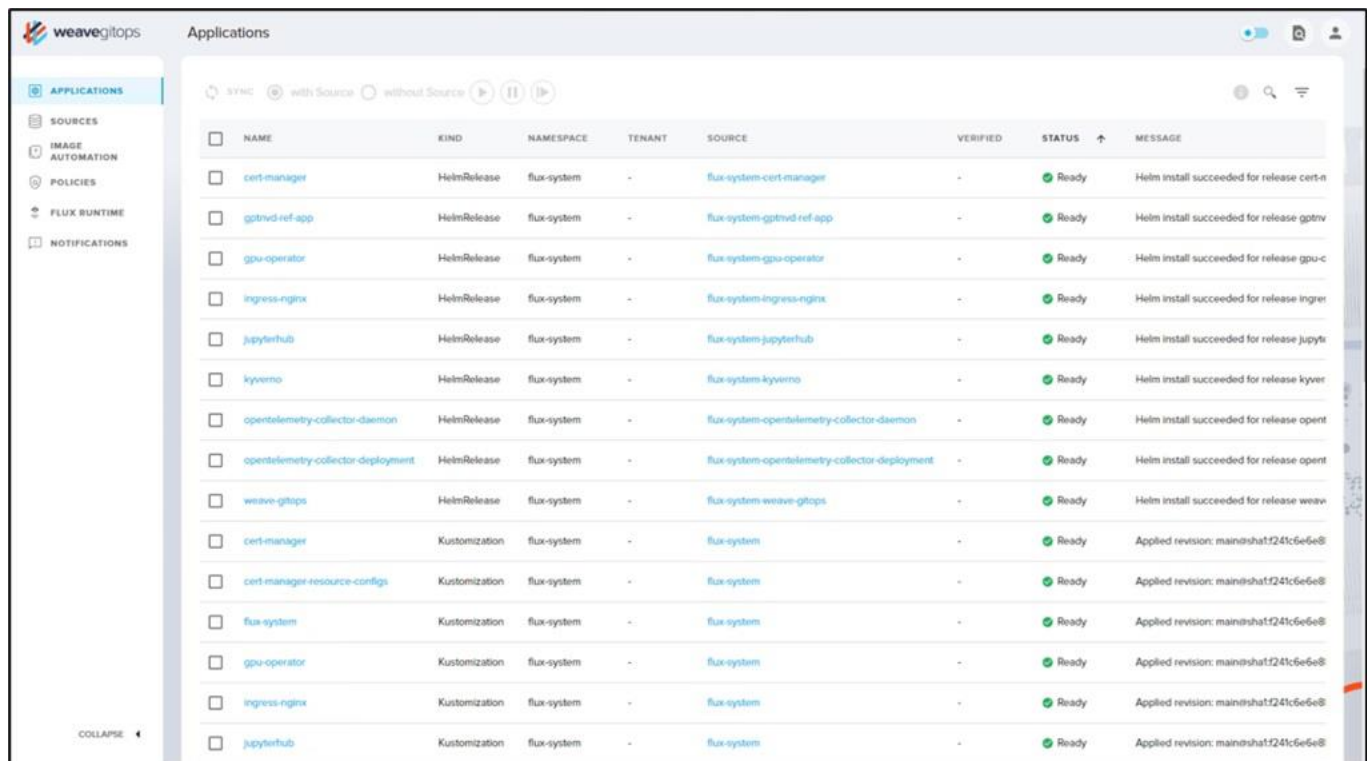
Step 3. Get the ingress endpoints by running the following command:

```
kubectl get ingress
```

Step 4. Check that the output similar to this:

```
(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]#
```

Step 5. Copy the HOSTS address for `gitops.ntnx-k8-mgmt.rtp4.local` from the previous output and paste it in your browser:

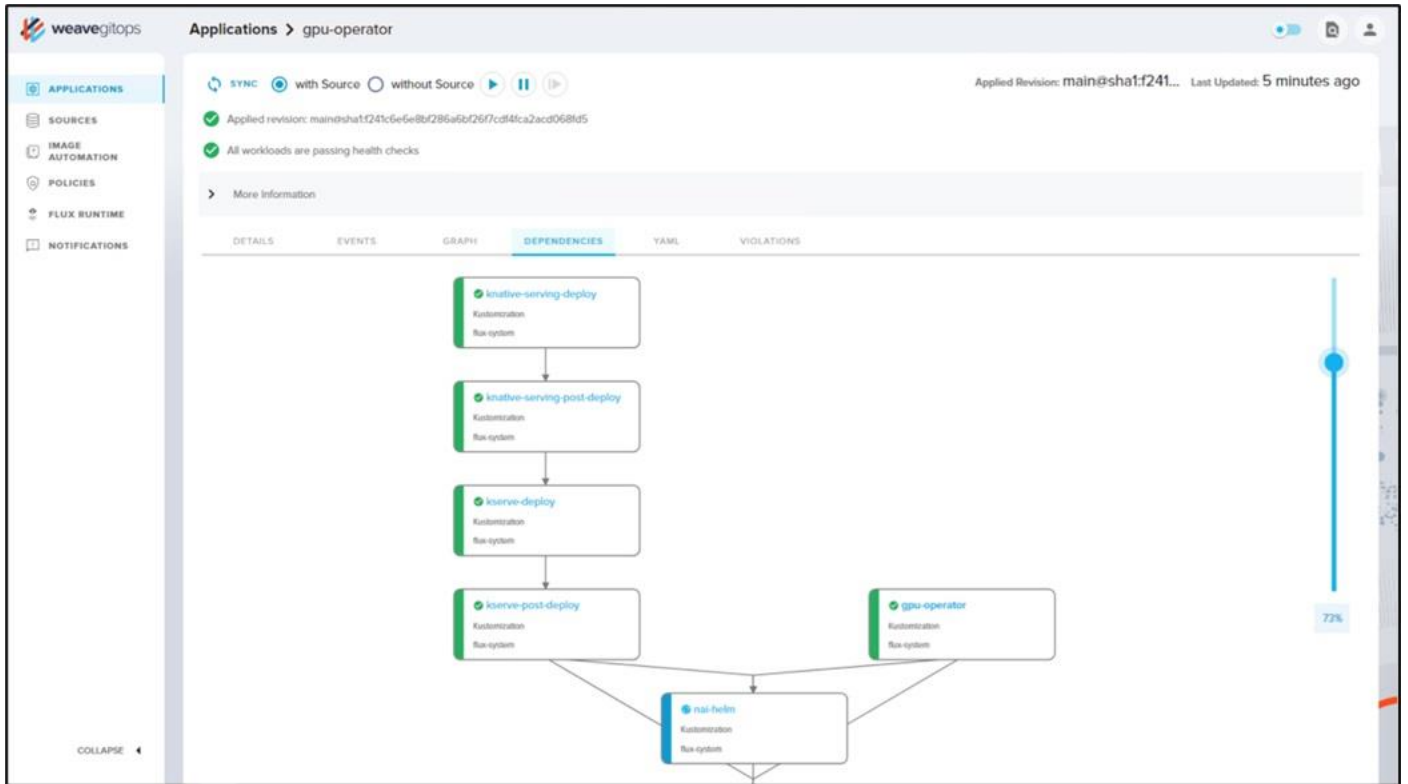


Step 6. You can see different kinds of application Flux is managing like HelmRelease and Kustomization. Click any to see the details of different Kubernetes resources.

NAME	KIND	NAMESPACE	STATUS	MESSAGE	IMAGES
gpu-operator	HelmRepository	flux-system	-	stored artifact: revision 'sha256-93ac1bc9b8356ea61e7a85a20dfe388bb517746cd435db08da798628dc679'	-
time-slicing-config	ConfigMap	gpu-operator	-	-	-
gpu-operator	Namespace	-	-	-	-
repo-config	ConfigMap	gpu-operator	-	-	-
gpu-operator	HelmRelease	flux-system	-	Helm install succeeded for release gpu-operator/gpu-operator-gpu-operatorv1 with chart gpu-operator/v23.9.0	-

Step 7. You can also get a graphical view of different Kubernetes objects and its hierarchy:

Step 8. Dependency between application can also be observed:



Step 9. Sources from where the Flux has pulled each component can be observed:

The screenshot shows the 'Sources' page in the Weave GitOps interface. The page displays a table of sources with the following columns: Namespace, Verified, Tenant, Status, Message, and URL. The table contains 15 rows of source information.

Namespace	Verified	Tenant	Status	Message	URL
flux-system	-	-	Ready	stored artifact for revision 'main:sha1:f241c5e6e8bf286afbf267cd4fca2acd068fd5'	https://github.com/pkoppa/nsi-ilm-flux-entra-gt
flux-system	-	-	Ready	stored artifact revision 'sha256:39bf0f3e67b87b687742b8cb846a0dc484d99d3dbe627d7d29...	https://charts.jetstack.io
flux-system	-	-	Ready	stored artifact revision 'sha256:93acfb98356ea6e7a85a20fd1e38bb597746cd435db98da7...	https://nvda.github.io/gpu-operator
flux-system	-	-	Ready	stored artifact revision 'sha256:fbf850dd20597dec2d72849c33b80a034546e7387ac5791L...	https://kubernetes.github.io/ingress-nginx
flux-system	-	-	Ready	stored artifact revision 'sha256:34d9042dc3a0744a85a4ead32336fccc62eb59cca55b6b3...	https://hub.jupyter.org/helm-charts/
flux-system	-	-	Ready	stored artifact revision 'sha256:862a44729a3812ee942cb885fcbef9d1f88e43770751e906be...	https://kyverno.github.io/kyverno/
flux-system	-	-	Ready	stored artifact revision 'sha256:c85b981c1c33043f35a5aa9cd9d6651c3413d49e560ba22d098...	https://jesse-gonzalez.github.io/nsi-ilm-helm/
flux-system	-	-	Ready	stored artifact revision 'sha256:72882cbf2565c9f8bae277c5ad72a86d57ec4d3d2fb54ffad3c...	https://open-telemetry.github.io/opentelemetry-helm-charts
flux-system	-	-	Not Ready	-	oci://ghcr.io/weaveworks/charts
flux-system	-	-	Ready	pulled 'cert-manager' chart with version 'v1.9.1'	-
flux-system	-	-	Ready	pulled 'gptwd-referenceapp' chart with version '0.2.7'	-
flux-system	-	-	Ready	pulled 'gpu-operator' chart with version 'v23.9.0'	-
flux-system	-	-	Ready	pulled 'ingress-nginx' chart with version '4.8.3'	-
flux-system	-	-	Ready	pulled 'jupyterhub' chart with version '3.3.7'	-
flux-system	-	-	Ready	pulled 'kyverno' chart with version '3.1.4'	-

Traces, Metrics, and Logs

Upttrace is integrated into the solution to provide traces, metrics, and logs.

Uptrace is an open source APM that supports distributed tracing, metrics, and logs. You can use it to monitor applications and troubleshoot issues. It comes with an intuitive query builder, rich dashboards, alerting rules, notifications, and integrations for most languages and frameworks. Uptrace uses OpenTelemetry framework to collect data.

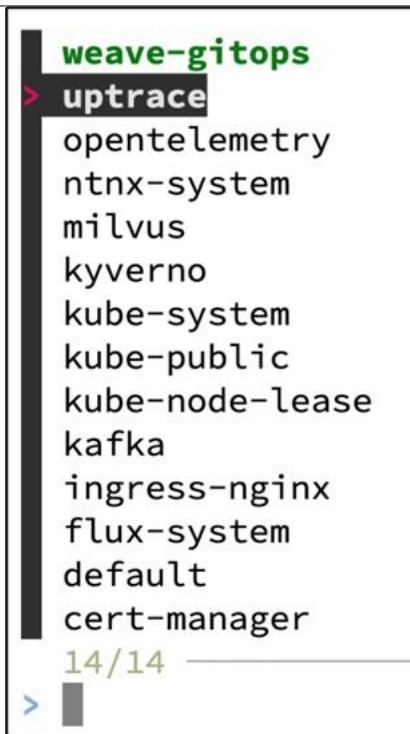
Procedure 1. Access Uptrace

Step 1. Uptrace is running on Management cluster. Switch to the management cluster by running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Management Cluster)
```

Step 2. Change the namespace to uptrace:

```
kubens [And select uptrace namespace]
```



Step 3. Get the ingress endpoints by running the following command:

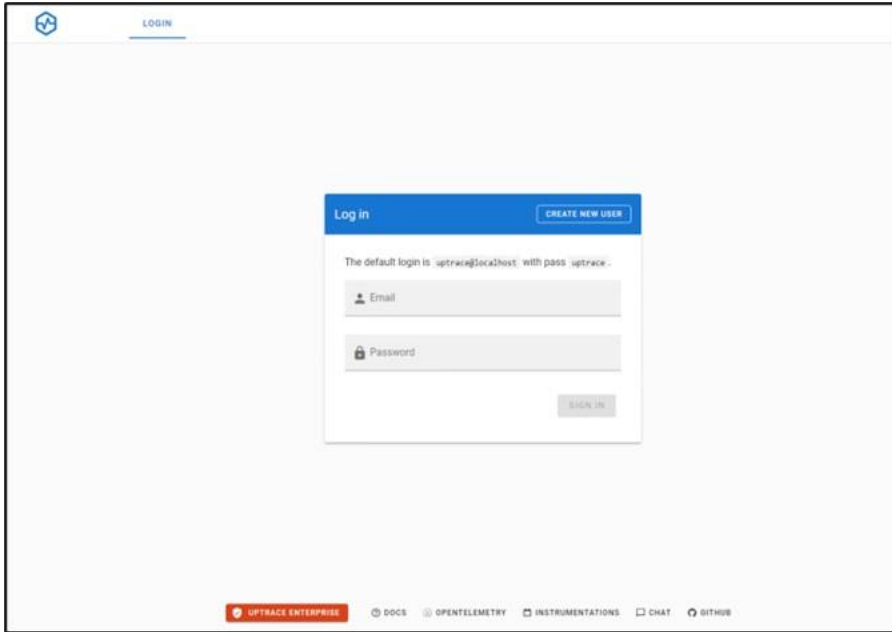
```
kubect1 get ingress
```

Step 4. Check that the output is similar to this:

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

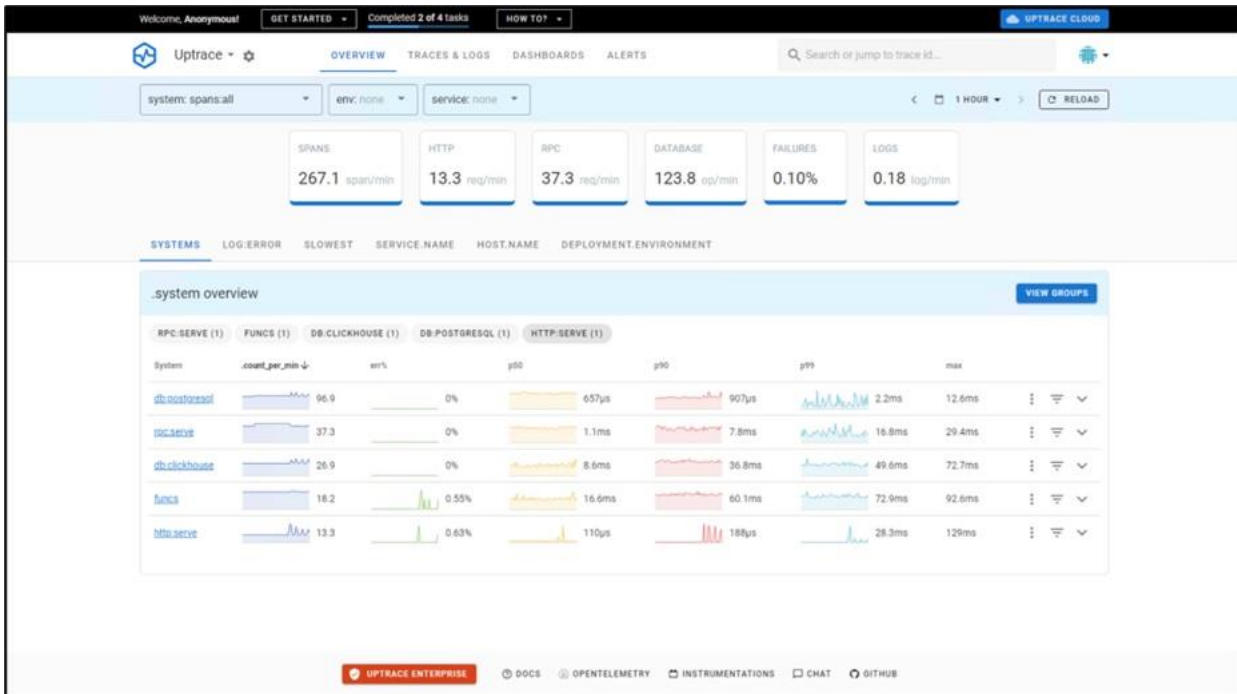
```
(devbox) [root@localhost nai-llm-fleet-infra]#
```

Step 5. Copy the HOSTS address for ingress service with name uptrace followed by management cluster name. In this example it is uptrace.ntnx-k8-mgmt.rtp4.local. Paste it in your browser.

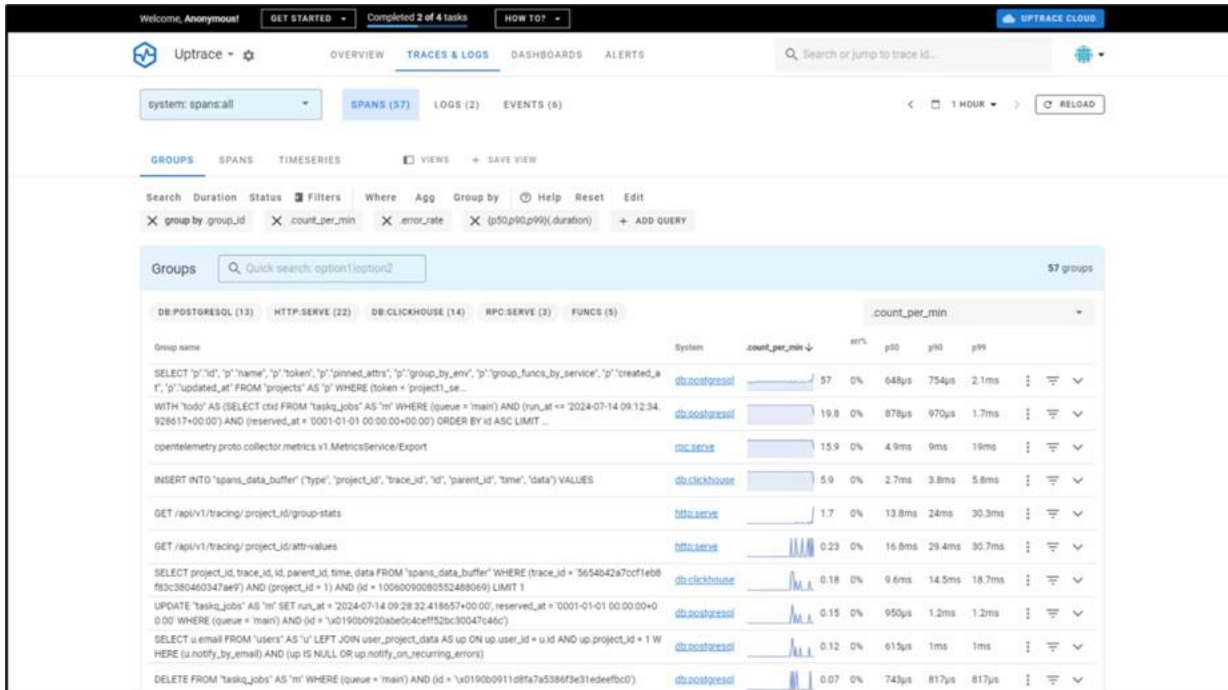


Note: The default login is uptrace@localhost with pass uptrace.

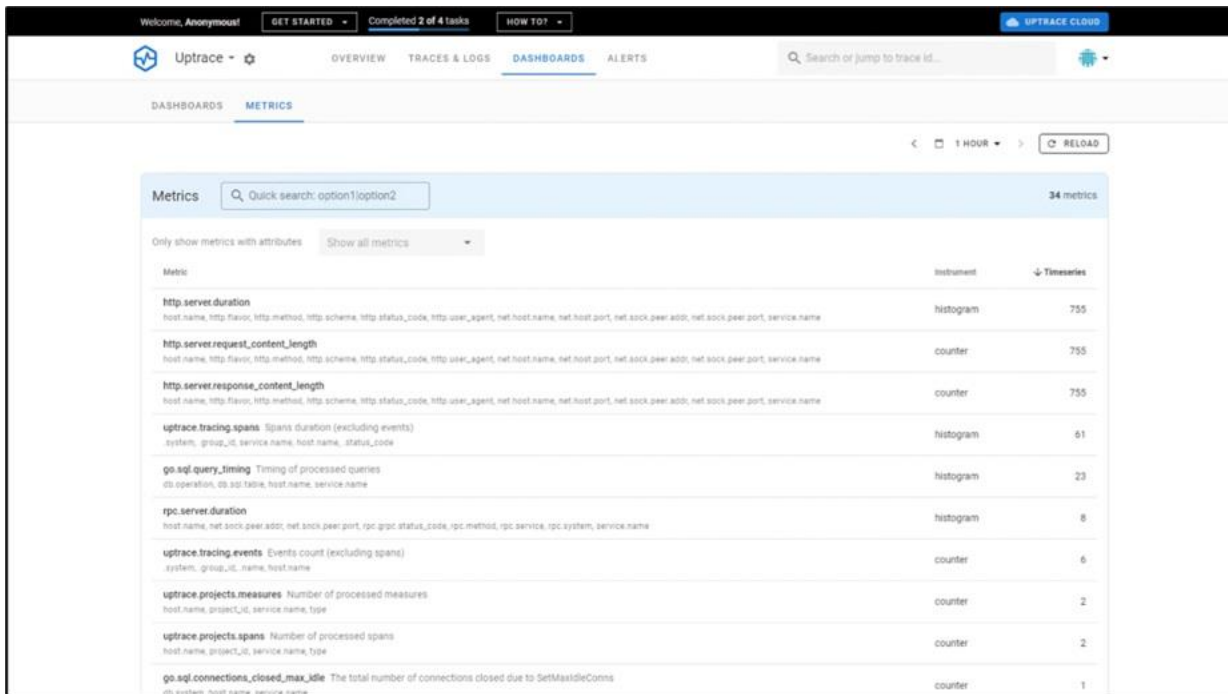
The System Overview is shown below:



Logs and events can be seen in TRACES & LOGS tab:



Query metrics can be seen in DASHBOARDS > METRICS tab:



View Documents in Object Browser

When the document is uploaded in the frontend application, the document gets stored as S3 objects in Nutanix Objects.

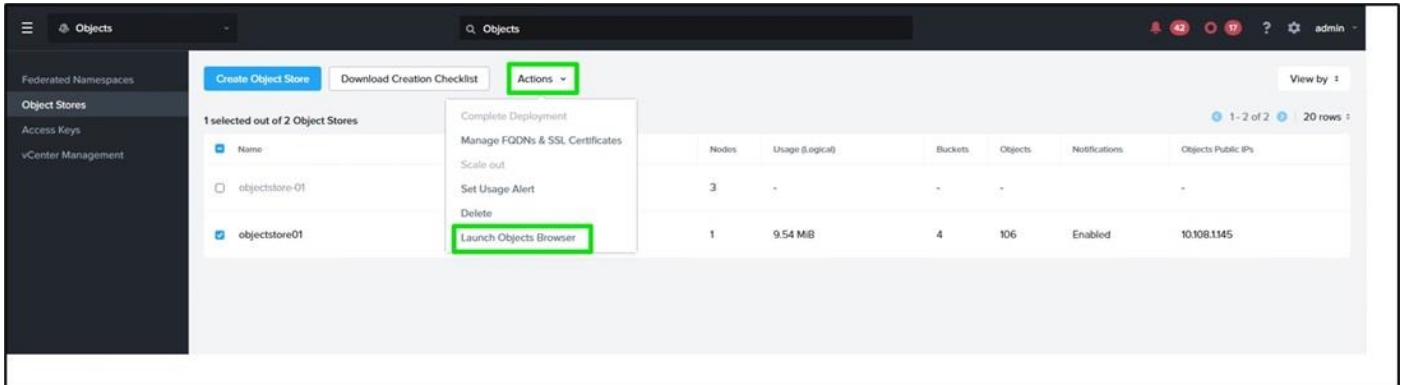
Procedure 1. View documents

Step 1. Log into the Prism Central web console.

Step 2. In the Application Switcher, click Objects.

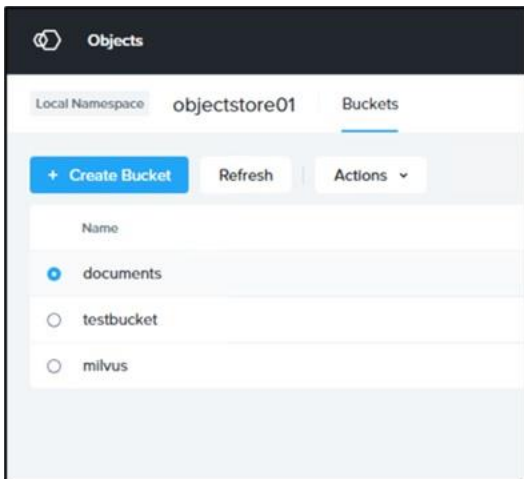
Step 3. Select the name of the object store.

Step 4. Click Actions and select Launch Objects Browser.

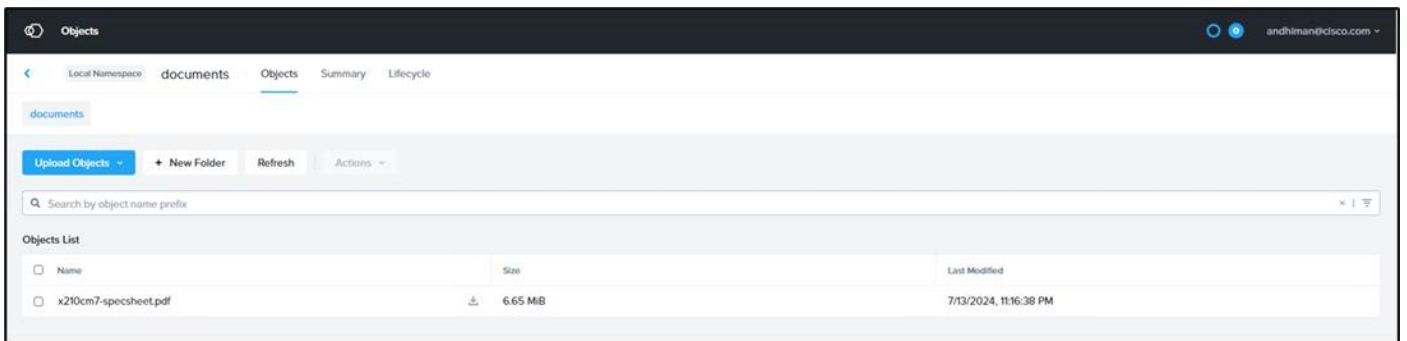


Step 5. Provide the access key and Secret Key.

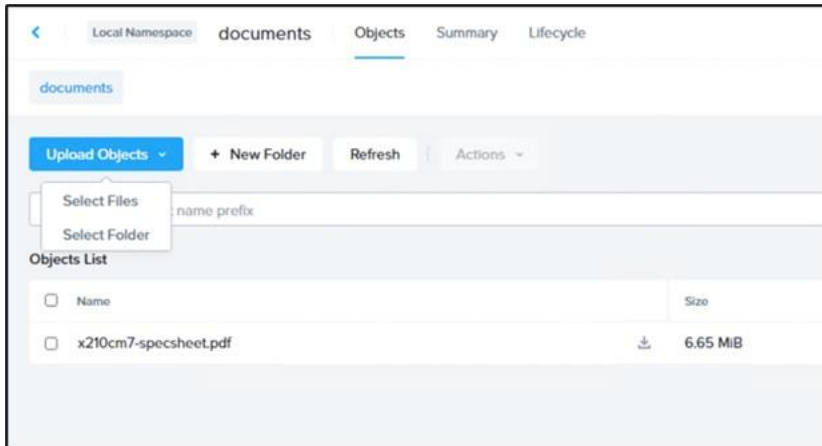
Step 6. Click the bucket which is configured to store the uploaded documents.



Step 7. Observe that the uploaded document can be seen here.



Step 8. Documents can also be uploaded directly from Objects Browser by clicking Upload Objects.



Explore Vector Database

When the document is uploaded, Kafka event notification is sent to document ingestion serverless function that is running inside knative and that will trigger Milvus for vectorization. The solution is integrated with Attu.

Attu is an efficient open-source management tool for Milvus. It features an intuitive graphical user interface (GUI), allowing you to easily interact with your databases. With just a few clicks, you can visualize your cluster status, manage metadata, perform data queries, and much more.

Procedure 1. View cluster details

Step 1. Milvus is running on Management cluster. Switch to the management cluster by running the following commands:

```
cd nai-llm-fleet-infra
devbox shell
eval $(task nke:switch-shell-env) (Choose NKE Management Cluster)
```

Step 2. Change the namespace to llm:

```
kubens llm
```


Step 3. Get the ingress endpoints by running the following command:

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

Step 4. Check that the output similar to this:

```
(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]# █
```

Step 5. Copy the HOSTS address for attu `attu.ntnx-k8-mgmt.rtp4.local` from the above output and paste it in your browser. You should be able to see the LLM chat interface.



Milvus Address
http://milvus-milvus-vectordb:19530

Milvus Username (optional)

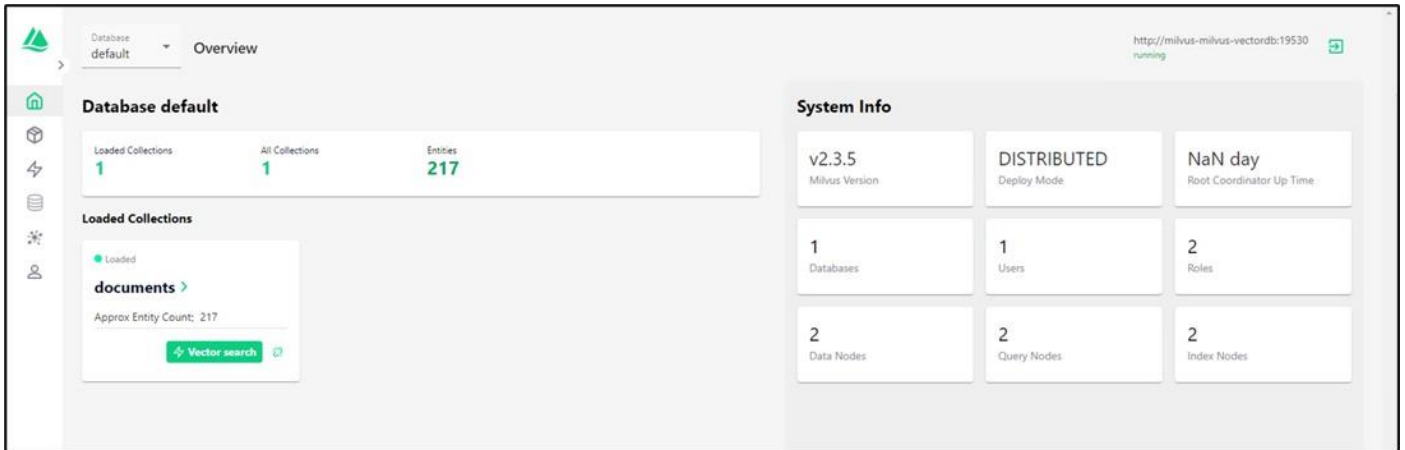
Milvus Password (optional)

Prometheus

Connect

Step 6. Click Connect.

Step 7. Observe that interface showing details of the database with the entity count. In this example, there are total of 217 chunks.



Database default Overview http://milvus-milvus-vectordb:19530 running

Database default

Loaded Collections	All Collections	Entities
1	1	217

Loaded Collections

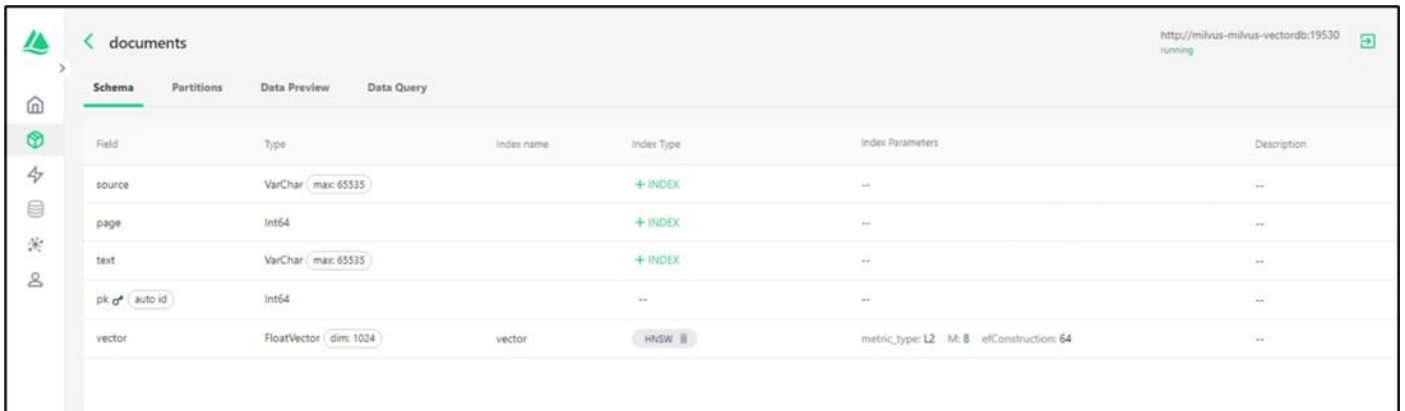
- Loaded
- documents** >

Approx Entity Count: 217

System Info

v2.3.5 Milvus Version	DISTRIBUTED Deploy Mode	NaN day Root Coordinator Up Time
1 Databases	1 Users	2 Roles
2 Data Nodes	2 Query Nodes	2 Index Nodes

Step 8. Click your collection to see details of the database.



< documents http://milvus-milvus-vectordb:19530 running

Schema Partitions Data Preview Data Query

Field	Type	Index name	Index Type	Index Parameters	Description
source	VarChar (max: 65535)		+ INDEX	--	--
page	Int64		+ INDEX	--	--
text	VarChar (max: 65535)		+ INDEX	--	--
pk (auto id)	Int64		--	--	--
vector	FloatVector (dim: 1024)	vector	HNSW	metric_type: L2 M: 8 efConstruction: 64	--

Step 9. Click Data Preview to see the content of the corresponding chunk, its document source, page number, and the corresponding vector embedding.

source	page	text	pk	vector
http://10.108.1.145/documents/x210cm7-spec...	1	STEP 2 CHOOSE CPU(S)	451123687711760204	[0.015420733951032162.0.0321568101644516.0.01658590830...
http://10.108.1.145/documents/x210cm7-spec...	1	SUPPLEMENTAL MATERIAL	451123687711760208	[0.0361153226204872.-0.009624278172850609.-0.005317337...
http://10.108.1.145/documents/x210cm7-spec...	2	Cisco UCS X210c M7 Compute NodeOVERIE...	451123687711760211	[0.00868761446326971.-0.005739330779761076.-0.003435577...
http://10.108.1.145/documents/x210cm7-spec...	2	NVMe drives for flexible boot and local storag...	451123687711760215	[0.023922406136989594.0.006204322446137667.0.0105536198...
http://10.108.1.145/documents/x210cm7-spec...	2	to each of the chassis Intelligent Fabric Mo dul...	451123687711760216	[0.000574273057281971.-0.011143035255372524.0.016494695...
http://10.108.1.145/documents/x210cm7-spec...	2	server: ■Cisco UCS Virtual Interface Card (VIC)...	451123687711760217	[0.011068921536207199.-0.005529055837541819.0.018088502...
http://10.108.1.145/documents/x210cm7-spec...	2	■Optional Mezzanine card: ■Cisco UCS Virtua...	451123687711760218	[0.005559561774134636.0.011557365767657757.0.0275784395...
http://10.108.1.145/documents/x210cm7-spec...	3	Cisco UCS X210c M7 Compute Node 4OVERVL...	451123687711760219	[0.012152687646448612.0.016107192263007164.0.0363079309...
http://10.108.1.145/documents/x210cm7-spec...	3	■Security : Includes secure boot silicon root of...	451123687711760220	[-0.000921503989957273.-0.00395165104418993.0.005515739...
http://10.108.1.145/documents/x210cm7-spec...	5	NVMe SSD1.6 TB 15 B 76 2 3 410 1415 16 121...	451123687711760223	[0.005680125206708908.-0.008490205742418766.0.017845239...
http://10.108.1.145/documents/x210cm7-spec...	7	Table 1 Capabilities and Features Capability/Fe...	451123687711760227	[0.016296643763780594.-0.0036298043560236692.0.01108283...
http://10.108.1.145/documents/x210cm7-spec...	7	Processors (16 per CPU) or 32 total DDR5-480...	451123687711760228	[-0.011961999349296093.-0.004978738240242004.0.0001027...
http://10.108.1.145/documents/x210cm7-spec...	7	are Registered DIMMs (RDIMMs) Storage Up L...	451123687711760229	[0.024439629167318344.0.024499114602804184.0.0292088147...

The interface proved various options for vector search:

Vector Search

1. Choose collection and field

Choose loaded collection: [Dropdown]

Choose vector field: [Dropdown]

2. Enter search vector

Please input your vector value here, e.g. [1, 2, 3, 4]

3. Set search parameters

Metric Type: L2 [Dropdown]

Round Decimals: -1 [Dropdown]

Search Results: TopK 100 [Dropdown] Advanced Filter [Icon] Time Travel [Icon]

[Reset] [Search]

Start your vector search

Sizing Considerations

Key Terms

Some of the key terms with respect to LLM inferencing are:

- **Batch Size:** Batch size refers to the number of samples that are processed together in a single inference run. The batch size can significantly impact the latency and throughput of the inference process. For example, increasing the batch size can increase throughput but at the cost of increased latency.
- **Precision:** The size in bytes used for each parameter in the model.
- **Context Size:** Represents the maximum number of tokens the model processes for the prompt + response. (Input Tokens Length + Output Tokens Length)
- **Key and Value Cache (KV Cache):** The amount of memory consumed by a single token based on the model dimensions and layers.
- **Activations:** When a token is being processed within the model, it is called an Activation. Full activation memory is calculated from the KV Cache size and context size.

LLM Inferencing Phases

LLMs generate text in a two-step process:

- **Prefill:** In the first phase, the model ingests your prompt tokens in parallel, populating the key-value (KV) cache. Prefill generates the key and value cache (KV Cache) for future decoding.
- **Decoding Phase:** In the second phase, we leverage our current state (stored in the KV cache) to sample and decode the next token. We pay a small price in storage to not recalculate the cache for every single new token. Without the KV cache, every successive token would take longer to sample because we would have to pass all previously seen tokens through the model.

The input sequence from the user is first tokenized in the same way the model's training data was tokenized. Then the tokens are fed to the trained model.

As the first step of model execution, the input sequence is converted into an embedding vector in the embedding layer of the trained model. This vector essentially translates the token into a high-dimensional space where similar tokens have similar vectors.

Using the token embeddings, queries, keys, and values for each token are computed through a process known as linear projection. Here, each token's embedding vector is multiplied by separate weight matrices learned during the training to produce the query, key, and value vectors.

The concept of using queries, keys, and values is directly inspired by how databases work. Each database storage has its data values indexed by keys, and users can retrieve the data by making a query and comparing if the key value matches the query. In the LLM case, the model generates the queries itself. The key values are not directly compared to the query, but the relevance of each key to the query is computed using a compatibility function to generate a weight vector.

An attention output is computed for each query as the weighted sum of the "values" using the weight vector previously computed.

This attention output goes through a prediction (or decode) layer, which assigns probabilities to each token in the entire vocabulary of the model, indicating the likelihood of that token being the next one.

One can sample from the predicted probability distribution to choose the next token probabilistically or select the token with the highest probability as the predicted next token.

Cluster Sizing

This CVD describes the design for a single GPT-in-a-Box cluster with four or more nodes. If one of the scaling factors (such as the total number of VMs, Kubernetes applications, or GPU workloads) exceeds the maximum specified for the solution, extend the cluster with the required capacity (CPU, memory, storage, GPU). After reaching the maximum cluster size, consider building a new cluster to support the demand for further growth.

Infrastructure Considerations

The following are the key factors for sizing infrastructure for Generative AI inferencing:

Model Specifications

- **Model Architecture:** Understand the architecture of the language model, including the number of layers, attention heads, and parameters.
- **Token Embedding Size:** Larger embedding sizes can significantly impact memory requirements.

Hardware Acceleration

- **Mixed Precision:** Explore mixed-precision training and inference to leverage hardware capabilities efficiently.

Memory Requirements

- **Model Size:** Large language models can have substantial memory requirements. Ensure sufficient GPU memory for both the model and input sequences.
- **Sequence Length:** Consider the maximum sequence length the model can handle and its impact on memory usage.

Batching and Parallelization

- **Batch Size:** Experiment with batch sizes. Larger batch sizes can improve throughput but may increase memory requirements.
- **Data Parallelism:** Implement data parallelism to distribute inference across multiple devices, if necessary.

Latency and Throughput

- **Latency Requirements:** Language models often have real-time constraints, especially in interactive applications. Minimize latency based on use case.
- **Throughput Targets:** Determine the required throughput in terms of processed tokens or sequences per second.

Scalability

- **Model Parallelism:** Consider model parallelism if the model size exceeds available GPU memory, distributing parts of the model across multiple GPUs.
- **Infrastructure Scaling:** Design for horizontal scalability to handle increased demand.

Redundancy and High Availability

- **Checkpointing:** Implement regular model checkpointing to recover from failures without losing training progress.
- **Replication:** Use redundant systems to ensure high availability during inference.

Network Considerations

-
- **Bandwidth:** Assess the network bandwidth required for transferring large model parameters between devices.
 - **Inter-Device Communication:** Optimize communication patterns between devices to minimize latency.

Storage Requirements

- **Model Storage:** Choose storage solutions capable of efficiently loading large model parameters.
- **Data Storage:** Assess storage needs for input data and any intermediate results.

Containerization and Orchestration

- **Containerization:** Deploy the language model within containers for easier management and consistency across different environments.
- **Orchestration:** Use container orchestration tools like Kubernetes for managing and scaling the deployment efficiently.

Middleware and Serving Frameworks

- **Serving Framework:** Choose a serving framework optimized for deploying large language models, such as TensorFlow Serving, Triton Inference Server, or others.
- **Middleware:** Implement middleware for handling communication between clients and the deployed model, ensuring compatibility with your application's requirements.

Monitoring and Optimization

- **Resource Monitoring:** Employ monitoring tools to track GPU utilization, memory usage, and other relevant metrics.
- **Dynamic Optimization:** Optimize parameters dynamically based on real-time performance metrics.

Security

- **Data Protection:** Implement measures to secure input and output data, especially if it involves sensitive information.
- **Model Security:** Protect large language models from adversarial attacks and unauthorized access.

When selecting the GPU for this solution, there are several crucial factors to consider:

- **Processing Power:** Look for GPUs with sufficient computational power to handle the complex neural network computations required for LLM inference.
- **Memory Capacity:** LLM models often have large memory requirements. Ensure that the GPU has enough VRAM (Video RAM) to accommodate the model size and batch sizes.
- **Tensor Cores:** Tensor cores accelerate matrix operations, which are essential for LLM inference. GPUs with tensor cores (such as NVIDIA's RTX series) can significantly improve performance.
- **Compatibility:** Check if the GPU is compatible with the Cisco UCS and the deep learning framework you plan to use and if (e.g., TensorFlow, PyTorch).
- **Power Consumption:** Consider the GPU's power draw and ensure it aligns with your system's power supply capacity.

Memory Calculations for LLM Inferencing

Total Memory required is the sum of model memory size and the KV cache.

Calculations for the required total memory is provided below:

$\text{Model Memory Size} = \text{Model Parameters} \times \text{Precision}$
 $\text{KV Cache Size} = 2 \times \text{Batch Size} \times \text{Context Size} \times \text{Number of Layers} \times \text{Model Dimensions} \times \text{Precision}$
 $\text{Total Memory Requirements (GB)} = \text{Model Memory Size (GB)} + \text{KV Cache Size (GB)}$

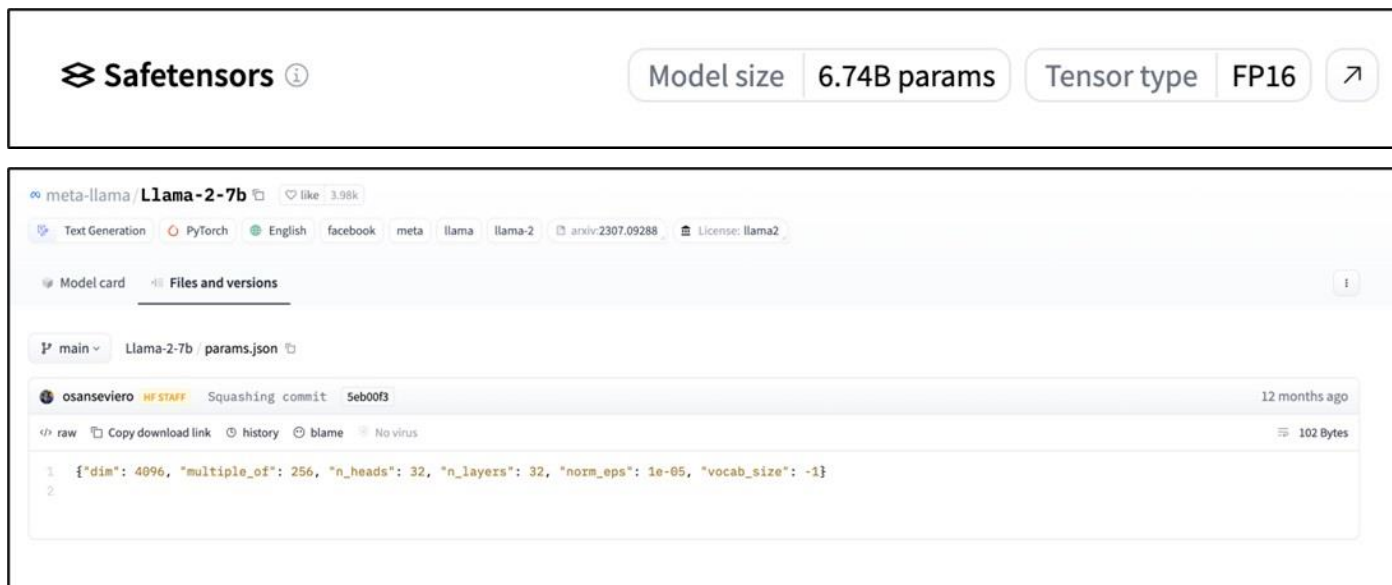
For some models, model dimension data might not be available. In that case, model dimension can be calculated as:

$\text{Model Dimensions} = \text{Attention Head Size} \times \text{Number of Attention Heads}$

Model Parameters, Precision, Number of layers, Model Dimension are specific to models, and it can be found in the Model card for the model.

Context Size and batch size are input from users.

We will provide an example memory calculation for Llama 2.



For the Llama 2 model:

Total model parameters: 6.74B Parameters.
Precision: FP16. (2 Bytes)
Number of layers: 32
Model Dimension: 4096

Therefore the model memory is calculated as shown below:

$\text{Model Memory Size} = \text{Model Parameters} \times \text{Precision}$

$\text{Model Memory Size for Llama 2} = 6,740,000,000 \times 2 \text{ Bytes/Parameter}$
 $= 13,480,000,000 \text{ Bytes}$
 $= 13.48 \text{ Giga Bytes}$

Also, considering an example of maximum Input Tokens Length of 1024, Maximum Output Tokens Length of 1024, and the Batch size of 8, below are the calculations for KV Cache Size:

$\text{KV Cache Size} = 2 \times \text{Batch Size} \times \text{Context Size} \times \text{Number of Layers} \times \text{Model Dimensions} \times \text{Precision}$
 $\text{KV Cache Size} = 2 \times 8 \times (1024+1024) \times 32 \times 4096 \times 2 \text{ Bytes/Parameter}$
 $= 8,589,934,592 \text{ Bytes}$
 $= 8.59 \text{ Giga Bytes}$

Therefore, Llama2 with maximum Input Tokens Length of 1024, Maximum Output Tokens Length of 1024, and the Batch size of 8, the total memory required is as shown below:

$$\begin{aligned}\text{Total Memory Requirements (GB)} &= \text{Model Memory Size (GB)} + \text{KV Cache Size (GB)} \\ &= 13.48 + 8.59 \text{ Giga Bytes} \\ &= 22.07 \text{ Giga Bytes}\end{aligned}$$

Performance Calculations

Performance benchmark can be run on model

Based on the performance requirement, number of users, number of input and output tokens, latency and throughput required, you can choose the appropriate Large Language Model, Inferencing backend, GPUs, and compute infrastructure.

Performance of the model depends on the prefill and decode phases. These two phases have different impacts on the performance of the LLM. While the prefill phase effectively saturates GPU compute at small batch sizes, the decode phase results in low compute utilization as it generates one token at a time per request.

The prefill phase is compute-bound, while the decode phase is memory-bound. So, the following factors need to be considered and measured:

- Prefill Latency
- Prefill Throughput
- Decode Total Latency
- Decode Token Latency
- Decode Throughput

The performance benchmark can be run with different sizes (1,2,4,8,10,25,250, 100 and so on). Also, separate tests can be run focused on performance comparison between 2 different models.

Conclusion

This validated solution stands as a valuable resource for navigating the complexities of Generative AI application deployment in real-world enterprise environments with special focus on Retrieval Augmented Generation.

This Cisco Validated Design for Cisco Compute Hyperconverged with Nutanix GPT-in-a-Box provides a foundational reference architecture for deployment of an innovative, flexible, and secure generative pretrained transformer (GPT) solution for Generative AI to privately run and manage organization's choice of AI large language models (LLMs) and applications in which it leverages.

In combination of Cisco UCS and Nutanix, this solution intends to enable an AI inferencing turnkey solution which can be quickly deployed as an on-premises solution. The infrastructure is designed using the Cisco UCS Managed HClAF240C M7 All-NVMe servers configured with 2x NVIDIA L40S GPUs which leverages Software-defined Nutanix Cloud Infrastructure supporting GPU-enabled server nodes for seamless scaling of virtualized compute, storage, networking supporting Kubernetes-orchestrated containers and Nutanix Unified Storage.

Appendix

This appendix contains the following:

- [Appendix A – Bill of Materials](#)
- [Appendix B – References used in this guide](#)

Appendix A - Bill of Materials

[Table 16](#) provides list of the Bill of Materials used in this solution design, deployment and validation described in this document.

Table 16. Bill of Materials

Line Number	Part Number	Description	Quantity
1.0	HCI-M7-MLB	Cisco Compute Hyperconverged M7 with Nutanix MLB	1
1.1	HCI AF240C-M7SN	Cisco Compute Hyperconverged HCI AF240cM7 All Flash NVMe Node	4
1.1.0.1	CON-L1NCO-HCIAFM7C	CX LEVEL 1 8X7XNCDOS Cisco Compute Hyperconverged HCI AF240cM	4
1.1.1	HCI-FI-MANAGED	Deployment mode for Server Managed by FI	4
1.1.2	HCI-GPUAD-C240M7	GPU AIR DUCT FOR C240M7	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 Boot optimized M.2 Raid controller	4
1.1.6	HCI-RAIL-M7	Ball Bearing Rail Kit for C220 & C240 M7 rack servers	4
1.1.7	HCI-TPM-OPT-OUT	OPT OUT, TPM 2.0, TCG, FIPS140-2, CC EAL4+ Certified	4
1.1.8	HCI-AOSAHV-67-SWK9	HCI AOS AHV 6.7 SW	4
1.1.9	UCSC-HSLP-C220M7	UCS C220 M7 Heatsink for & C240 GPU Heatsink	8
1.1.10	UCSC-BBLKD-M7	UCS C-Series M7 SFF drive blanking panel	72
1.1.11	UCSC-M2EXT-240-D	C240M7 2U M.2 Extender board	4
1.1.12	UCSC-RISAB-24XM7	UCS C-Series M7 2U Air Blocker GPU only	4
1.1.13	CBL-G5GPU-C240M7	C240M7 PCIe CEM compliant 12VHPWR power cable(up to 450W)	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-Series M7 2U Riser 1C PCIe Gen5 (2x16)	4
1.1.17	HCI-RIS2C-24XM7	UCS C-Series M7 2U Riser 2C PCIe Gen5 (2x16) (CPU2)	4
1.1.18	HCI-RIS3C-24XM7	C240 M7 Riser 3C	4
1.1.19	HCI-MLOM	Cisco VIC Connectivity	4
1.1.20	HCI-M-V5D200G	Cisco VIC 15238 2x 40/100/200G mLOM C-Series	4
1.1.21	HCI-GPU-L40S	NVIDIA L40S: 350W, 48GB, 2-slot FHFL GPU	4
1.1.22	HCI-NV-GRID-OPTOUT	NVIDIA GRID SW OPTOUT	4
1.1.23	HCI-GPU-L40S	NVIDIA L40S: 350W, 48GB, 2-slot FHFL GPU	4
1.1.24	HCI-NV-GRID-OPTOUT	NVIDIA GRID SW OPTOUT	4
1.1.25	HCI-PSU1-2300W	Cisco UCS 2300W AC Power Supply for Rack Servers Titanium	8
1.1.26	CAB-C19-CBN	Cabinet Jumper Power Cord, 250 VAC 16A, C20-C19 Connectors	8
1.2	HCI-FI-6536	Cisco Compute Hyperconverged Fabric Interconnect 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	UCSM Deployment mode for FI	2
1.2.2	N10-MGT018	UCS Manager v4.2 and Intersight Managed Mode v4.2	2
1.2.3	HCI-FI-6500-SW	Perpetual SW License for the 6500 series Fabric Interconnect	2
1.2.4	HCI-PSU-6536-AC	UCS 6536 Power Supply/AC 1100W PSU - Port Side Exhaust	4
1.2.5	CAB-9K12A-NA	Power Cord, 125VAC 13A NEMA 5-15 Plug, North America	4
1.2.6	UCS-ACC-6536	UCS 6536 Chassis Accessory Kit	2
1.2.7	UCS-FAN-6536	UCS 6536 Fan Module	12

Appendix B – References use in this guide

Deployment & Field Guide to deploy Cisco Compute Hyperconverged with Nutanix

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>

Repository

GitHub repository for solution: <https://github.com/ucs-compute-solutions/nai-llm-fleet-infra>

GPT-in-a-Box Foundation

<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 Reference Documentation

<https://portal.nutanix.com/>

About the Authors

Anil Dhiman, Technical Marketing Engineer, Cisco Systems, Inc.

Anil Dhiman has over 20 years of experience specializing in data center solutions on Cisco UCS servers, and performance engineering of large-scale enterprise applications. Over the past 14 years, Anil has authored several Cisco Validated Designs for enterprise solutions on Cisco data center technologies. Currently, Anil's focus is on Cisco's portfolio of hyperconverged infrastructure and data protection solutions.

Paniraja Koppa, Technical Marketing Engineer, Cisco Systems, Inc.

Paniraja Koppa is a member of the Cisco Unified Computing System (Cisco UCS) solutions team. He has over 15 years of experience designing, implementing, and operating solutions in the data center. In his current role, he works on design and development, best practices, optimization, automation and technical content creation of compute and hybrid cloud solutions. He also worked as technical consulting engineer in the data center virtualization space. Paniraja holds a master's degree in computer science. He has presented several papers at international conferences and speaker at events like Cisco Live US and Europe, Open Infrastructure Summit, and other partner events. Paniraja's current focus is on Generative AI solutions.

Wolfgang Huse, Sr. Staff Solution Architect - Cloud Native & AI, Nutanix, Inc.

Wolfgang Huse is the technical lead for Cloud Native in Solutions and Performance Engineering team at Nutanix and is based out of Germany. In this role, he and his team are primarily responsible for development of cloud native solutions and evangelizing the benefits of adopting cloud native architectures within Nutanix Cloud Platform. Wolfgang and his team have been instrumental in building many key solution artifacts such as the various Nutanix tech notes, best practice guides and validated designs to support solutions such as RedHat OpenShift, Rancher and most recently Nutanix GPT in-a-Box. Prior to this role, he worked in sales engineering roles at Nutanix, directly providing assistance on designing and implementing complex solutions, globally.

Jesse Gonzalez, Staff Solution Architect - Cloud Native & AI, Nutanix, Inc.

Jesse Gonzalez is a Cloud Native Solutions Architect on the Solutions and Performance Engineering team at Nutanix. With over 20+ years of experience in IT, Jesse has had the privilege of working closely with many organizations of all sizes to overcome their challenges in enabling cloud-native (and more recently Generative AI) solutions on the Nutanix platform. Prior to this role, Jesse has worked in roles within both services and sales engineering at Nutanix.

Acknowledgements

For their support and contribution to the design, validation, and creation of this Cisco Validated Design, the authors would like to thank:

- Chris O'Brien, Senior Director, Cisco Systems, Inc.
- John McAbel, Product Manager, Cisco Systems, Inc.

Feedback

For comments and suggestions about this guide and related guides, join the discussion on [Cisco Community](https://cs.co/en-cvds) at <https://cs.co/en-cvds>.

CVD Program

ALL DESIGNS, SPECIFICATIONS, STATEMENTS, INFORMATION, AND RECOMMENDATIONS (COLLECTIVELY, "DESIGNS") IN THIS MANUAL ARE PRESENTED "AS IS," WITH ALL FAULTS. CISCO AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE. IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THE DESIGNS, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THE DESIGNS ARE SUBJECT TO CHANGE WITHOUT NOTICE. USERS ARE SOLELY RESPONSIBLE FOR THEIR APPLICATION OF THE DESIGNS. THE DESIGNS DO NOT CONSTITUTE THE TECHNICAL OR OTHER PROFESSIONAL ADVICE OF CISCO, ITS SUPPLIERS OR PARTNERS. USERS SHOULD CONSULT THEIR OWN TECHNICAL ADVISORS BEFORE IMPLEMENTING THE DESIGNS. RESULTS MAY VARY DEPENDING ON FACTORS NOT TESTED BY CISCO.

CCDE, CCENT, Cisco Eos, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, the Cisco logo, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unified Computing System (Cisco UCS), Cisco UCS B-Series Blade Servers, Cisco UCS C-Series Rack Servers, Cisco UCS S-Series Storage Servers, Cisco UCS X-Series, Cisco UCS Manager, Cisco UCS Management Software, Cisco Unified Fabric, Cisco Application Centric Infrastructure, Cisco Nexus 9000 Series, Cisco Nexus 7000 Series, Cisco Prime Data Center Network Manager, Cisco NX-OS Software, Cisco MDS Series, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trade-marks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. (LDW_P3)

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0809R)

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at <https://www.cisco.com/go/offices>.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)