VMware Private AI Foundation with NVIDIA 5.1

Table of Contents

Release Notes

Release highlights, such as what's new, supported software components, known issues, and other, for VMware Private AI Foundation with NVIDIA provide information about each release.

VMware Deep Learning VM Image Release Notes

This document contains the following sections

Introduction

The VMware Deep Learning VM images are delivered as part of VMware Private AI Foundation with NVIDIA. They are preconfigured with popular DL workloads, and are optimized and validated by NVIDIA and VMware for GPU acceleration in a VMware Cloud Foundation environment.

VMware Deep Learning VM 1.2 | 09 OCT 2024 Check for additions and updates to these release notes.

Content Library

Deep learning VM images are delivered as vSphere VM templates, hosted and published by VMware in a content library. You can use these images to deploy Deep learning VM by using the vSphere Client or VMware Aria Automation.

The content library with deep learning VM images for VMware Private AI Foundation with NVIDIA is available at the https://packages.vmware.com/dl-vm/lib.json URL. In a connected environment, you create a subscribed content library connected to this URL, and in a disconnected environment - a local content library where you upload images from the central content library.

Compatibility and Upgrade

Use the latest release of VMware Deep Learning VM if supported by your environment.

Updating a running deep learning VM to a later image is not supported. You must deploy a new deep learning virtual machine by using a later deep learning VM image release.

Installation

You deploy a deep learning VM image from a content library on the vCenter Server instance for the AI-ready VI workload domain. You can deploy a deep learning VM on the following systems:

- As a data scientists, MLOps engineer, or DevOps engineer
  - On a Supervisor in vSphere IaaS control plane by using the VMware Aria Automation.
- As a cloud administrator
  - Directly on a vSphere cluster
- As a DevOps engineer
  - On a Supervisor in vSphere IaaS control plane by using the kubectl.

See Deploying a Deep Learning VM in VMware Private AI Foundation with NVIDIA.

VMware Deep Learning VM 1.2

Image Snapshot

VMware Deep Learning VM 1.2 is available for use with VMware Cloud Foundation 5.2.1.

Snapshot	Release Date	Compatible VMware Cloud Foundation Version
common-container-nv-vgpu-ubuntu-2204-v20240814	09 OCT 2024	VMware Cloud Foundation 5.2.1

What's New

- The deep learning VM image includes the Broadcom EULA and VMware Private AI Foundation with NVIDIA SPD (specific program documentation).
- The embedded Miniconda 24.3.0 component is updated to Miniforge3 24.3.0.
- In addition to `pytorch2.3.0_py3.12` , by using the `Conda Environment Install` OVF parameter, you can also have the `pytorch1.13.1_py3.10` , `tf2.16.1_py3.12` , and `tf1.15.5_py3.7` Conda environments installed during VM deployment.
- Private AI Services (`pais`) CLI version 1.0.0 for storing ML models in a central Harbor registry is now available.
- In a connected environment, downloading the vGPU guest driver now requires only NVIDIA AI Enterprise entitlement.
- In a connected environment, error messages that appear while downloading the vGPU guest driver are improved.

Supported NVIDIA GPU Devices

VMware Deep Learning VM 1.2 supports the following GPUs on your ESXi hosts:

NVIDIA Component	Supported Option
NVIDIA GPUs	• NVIDIA A100 • NVIDIA L40S • NVIDIA H100
GPU sharing mode	• Time slicing • Multi-Instance GPU

Components of VMware Deep Learning VM 1.2

This version of the deep learning virtual machine image contains the following software:

Software Component Category	Software Component		Version
Embedded	Canonical Ubuntu		22.04
	NVIDIA Container Toolkit		1.15.0
	Docker Community Engine		26.0.2
	Miniforge		24.3.0-0 (Python 3.10)
	VMware Private AI Services (pais) CLI		1.0.0
Can be pre-installed automatically	NVIDIA vGPU guest driver		According to the version of the NVIDIA vGPU host driver
	PyTorch Conda Environment		2.3.0 (Python 3.12), 1.13.1 (Python 3.10)
	TensorFlow Conda Environment		2.16.1 (Python 3.12), 1.15.5 (Python 3.7)
	Deep learning (DL) workload from NVIDIA NGC	CUDA Sample	-
		PyTorch	-
		TensorFlow	-
		DCGM Exporter	-
		Triton Inference Server	-
		NVIDIA RAG	-

Resolved Issues

- Containers deployed by using `cloud-init` are running as `root` .
- When the deep learning VM is restarted, only log information from the most recent boot is visible in `/var/log/dl.log` . DL workload log information from earlier boots is overwritten.
- Installation of Conda environments fails if the password OVF parameter is set.

VMware Deep Learning VM 1.1

Image Snapshot

VMware Deep Learning VM 1.1 is available for use with VMware Cloud Foundation 5.2.

Snapshot	Release Date	Compatible VMware Cloud Foundation Version
common-container-nv-vgpu-ubuntu-2204-v20240613	23 JUL 2024	VMware Cloud Foundation 5.2

What's New

- The deep learning VM image now contains a built-in Miniconda installation.
- The deep learning VM image now contains a verified PyTorch Conda environment manifest.
- You can use the `Conda Environment Install` OVF parameter to specify a comma-separated list of Conda environments to automatically install during VM deployment. Currently, you can install a `pytorch2.3_py3.12` environment.

- More detailed logs on the initialization script are available in `/var/log/vgpu-install.log`.

Supported NVIDIA GPU Devices

VMware Deep Learning VM 1.1 supports the following GPUs on your ESXi hosts:

NVIDIA Component	Supported Option
NVIDIA GPUs	• NVIDIA A100 • NVIDIA L40S • NVIDIA H100
GPU sharing mode	• Time slicing • Multi-Instance GPU

Components of VMware Deep Learning VM 1.1

This version of the deep learning virtual machine image contains the following software:

Software Component Category	Software Component		Version
Embedded	Canonical Ubuntu		22.04
	NVIDIA Container Toolkit		1.15.0
	Docker Community Engine		26.0.2
	Miniconda		24.3.0-0 (Python 3.12)
Can be pre-installed automatically	NVIDIA vGPU guest driver		According to the version of the NVIDIA vGPU host driver
	PyTorch Conda Environment		2.3.0 (Python 3.12)
	Deep learning (DL) workload from NVIDIA NGC	CUDA Sample	-
		PyTorch	-
		TensorFlow	-
		DCGM Exporter	-
		Triton Inference Server	-
		NVIDIA RAG	-

Resolved Issues

- Earlier versions of the NVIDIA vGPU driver are not downloaded from the NVIDIA License Portal.
- The GuestBootstrap status is shown incorrectly in certain cases.
- NVIDIA vGPU driver download might fail because of network issues.
- The `authorized_keys` SSH file, used during the image build process, is available in the `~/.ssh/` directory.

VMware Deep Learning VM 1.0.1

Image Snapshot

VMware Deep Learning VM 1.0.1 is available for use with VMware Cloud Foundation 5.1.1.

Snapshot	Release Date	Compatible VMware Cloud Foundation Version
common-container-nv-vgpu-ubuntu-2204-v20240419	06 MAY 2024	VMware Cloud Foundation 5.1.1

What's New

- The versions of the NVIDIA Container Toolkit and Docker Community Engine are updated.
- The description of the OVF properties shown when deploying a deep learning VM by using the OVF deployment wizard is improved.
- The download URL format for vGPU guest drivers for disconnected environments now supports directory index listings, as generated by Web servers, such as NGINX or Apache HTTP Server.
- A link to documentation of the VMware deep learning VM appears as a "message of the day" in the Ubuntu operating system.

Supported NVIDIA GPU Devices

VMware Deep Learning VM 1.0.1 supports the following GPUs on your ESXi hosts:

NVIDIA Component	Supported Option
NVIDIA GPUs	• NVIDIA A100 • NVIDIA L40S • NVIDIA H100
GPU sharing mode	• Time slicing • Multi-Instance GPU

Components of VMware Deep Learning VM 1.0.1

This version of the deep learning virtual machine image contains the following software:

Software Component Category	Software Component		Version
Embedded	Canonical Ubuntu		22.04
	NVIDIA Container Toolkit		1.15.0
	Docker Community Engine		26.0.2
Can be pre-installed automatically	NVIDIA vGPU guest driver		According to the version of the NVIDIA vGPU host driver
	Deep learning (DL) workload from NVIDIA NGC	CUDA Sample	-
		PyTorch	-
		TensorFlow	-
		DCGM Exporter	-
		Triton Inference Server	-
		NVIDIA RAG	-

Resolved Issues

- Unable to log in to a Docker private container registry if the registry password set in the OVF properties of the deep learning VM contains special characters, such as `& < > " '`.
- The OVF properties for a secondary container registry are not processed.
- Running `apt update` fails because of errors and security warnings.
- The execution status of the `get-vgpu-driver.sh` script, run at VM startup, is not reflected in the `guestinfo.vmservice.bootstrap.condition` setting of VM Tools.

VMware Deep Learning VM 1.0

Image Snapshot

VMware Deep Learning VM 1.0 is available for use with VMware Cloud Foundation 5.1.1.

Snapshot	Release Date	Compatible VMware Cloud Foundation Version
common-container-nv-vgpu-ubuntu-2204-v20240217	26 MAR 2024	VMware Cloud Foundation 5.1.1

Supported NVIDIA GPU Devices

VMware Deep Learning VM 1.0 supports the following GPUs on your ESXi hosts:

NVIDIA Component	Supported Option
NVIDIA GPUs	• NVIDIA A100 • NVIDIA L40S • NVIDIA H100
GPU sharing mode	• Time slicing • Multi-Instance GPU

Components of VMware Deep Learning VM 1.0

The initial version of the deep learning virtual machine image contains the following software:

Software Component Category	Software Component		Version
Embedded	Canonical Ubuntu		22.04
	NVIDIA Container Toolkit		1.13.5
	Docker Community Engine		25.03
Can be pre-installed automatically	NVIDIA vGPU guest driver		According to the version of the NVIDIA vGPU host driver
	Deep learning (DL) workload from NVIDIA NGC	CUDA Sample	-
		PyTorch	-
		TensorFlow	-
		DCGM Exporter	-
		Triton Inference Server	-
		NVIDIA RAG	-

License Information

VMware Deep Learning VM releases are available under a VMware Private AI Foundation with NVIDIA license. See VMware Private AI Foundation with NVIDIA Guide.

Documentation

Examine the VMware Private AI Foundation with NVIDIA Guide for an overview and how-to instructions for running deep learning VMs in a VMware Cloud Foundation environment.

VMware Private AI Foundation with NVIDIA Guide

The *VMware Private AI Foundation with NVIDIA Guide* provides an overview of the components of VMware Private AI Foundation with NVIDIA and high-level workflows for development and production use cases.

Intended Audience

The information in *VMware Private AI Foundation with NVIDIA Guide* is intended for data center cloud administrators, data scientists, and DevOps engineers who are familiar with:

- Cloud administrators
  - Concepts of virtualization and software-defined data centers (SDDCs)
  - Hardware components such as top-of-rack (ToR) switches, inter-rack switches, servers with direct attached storage, cables, and power supplies
  - Methods for setting up NVIDIA GPUs on servers in a data center
  - Using VMware vSphere® to work with virtual machines.
  - Using vSphere with Tanzu to configure and assign vSphere resources to vSphere namespaces on a Supervisor.
- Data scientists
  - Provisioning virtual machines in vSphere with Tanzu by using the VM Service and the Kubernetes API.
  - Containers, including Docker, Helm charts and Harbor Registry
- DevOps engineers
  - Provisioning virtual machines in vSphere using the Kubernetes API.
  - Containers, including Docker, Helm charts and Harbor Registry
  - Working with vSphere with Tanzu for provisioning VMs and Tanzu Kubernetes Grid (TKG) clusters.

VMware Software Components

The functionality of the VMware Private AI Foundation with NVIDIA solution is available across several software components.

Software Category	Supported Software Versions
Management components	See VMware Components in VMware Private AI Foundation with NVIDIA.
Deep learning VM components	See VMware Deep Learning VM Image Release Notes .
TK releases (TKr)	See VMware Tanzu Kubernetes releases Release Notes.

Related VMware Documentation

The VMware Private AI Foundation with NVIDIA solution includes a stack of VMware software products and components. The documentation for those software products is as follows:

- VMware Cloud Foundation 5.1 Documentation
- VMware vSphere Documentation
- VMware vSAN Documentation
- VMware vSphere with Tanzu Documentation
- VMware Aria Automation Documentation
- VMware Aria Operations Documentation
- VMware Aria Suite Lifeycle Documentation

- VMware Data Services Manager Documentation

VMware Cloud Foundation Glossary

The VMware Cloud Foundation Glossary defines terms specific to VMware Cloud Foundation.

Update History

This *VMware Private AI Foundation with NVIDIA Guide* is updated when necessary.

Revision	Description
14 MAY 2024	You can now use VMware Aria Automation 8.17 for provisioning AI workloads in VMware Private AI Foundation with NVIDIA. See Set Up VMware Aria Automation for VMware Private AI Foundation with NVIDIA.
06 MAY 2024	• You can now use VMware Aria Operations 8.17.1 for monitoring workload domains running VMware Private AI Foundation with NVIDIA. See Monitoring VMware Private AI Foundation with NVIDIA. • The documentation is updated for the latest versions of the deep learning VM images. See Deploying a Deep Learning VM in VMware Private AI Foundation with NVIDIA and Deploying RAG Workloads in VMware Private AI Foundation with NVIDIA. • Learn more about deploying a deep learning VM with a static IP address. See Assign a Static IP Address to a Deep Learning VM in VMware Private AI Foundation with NVIDIA. • For a disconnected environment, to easily keep the NVIDIA NGC container images in the local Harbor registry up-to-date, use a setup with two registry instances. See Setting Up a Private Harbor Registry in VMware Private AI Foundation with NVIDIA. • For efficient metrics collection, the documentation is updated with details about running a deep learning (DL) workload and DCGM Exporter in the same deep learning VM. See DCGM Exporter. • Learn more about deploying a database in VMware Data Services Manager by using a self-service catalog item in VMware Aria Automation. See Deploying a Vector Database by Using a Self-Service Catalog Item in VMware Aria Automation.
26 MAR 2024	Initial version.

What is VMware Private AI Foundation with NVIDIA?

As a multi-component solution, you can use VMware Private AI Foundation with NVIDIA to run generative AI workloads by using accelerated computing from NVIDIA, and virtual infrastructure management and cloud management from VMware Cloud Foundation.

VMware Private AI Foundation with NVIDIA provides a platform for provisioning AI workloads on ESXi hosts with NVIDIA GPUs. In addition, running AI workloads based on NVIDIA GPU Cloud (NGC) containers is specifically validated by VMware.

VMware Private AI Foundation with NVIDIA supports two use cases:

Development use case
: Cloud administrators and DevOps engineers can provision AI workloads, including Retrieval-Augmented Generation (RAG), in the form of deep learning virtual machines. Data scientists can use these deep learning virtual machines for AI development.

Production use case
: Cloud administrators can provide DevOps engineers with a VMware Private AI Foundation with NVIDIA environment for provisioning production-ready AI workloads on Tanzu Kubernetes Grid (TKG) clusters on vSphere with Tanzu.

Licensing

You need the VMware Private AI Foundation with NVIDIA add-on license to access the following functionality:

- Private AI setup in VMware Aria Automation for catalog items for easy provisioning of GPU-accelerated deep learning virtual machines and TKG clusters.
- Provisioning of PostgreSQL databases with the pgvector extension with enterprise support.
- Deploying and using the deep learning virtual machine image delivered by VMware by Broadcom.

You can deploy AI workloads with and without Supervisor enabled and use the GPU metrics in vCenter Server and VMware Aria Operations under the VMware Cloud Foundation license.

The NVIDIA software components are available for use under an NVIDIA AI Enterprise license.

What are the VMware Private AI Foundation with NVIDIA components?

Figure 1: Example Architecture for VMware Private AI Foundation with NVIDIA

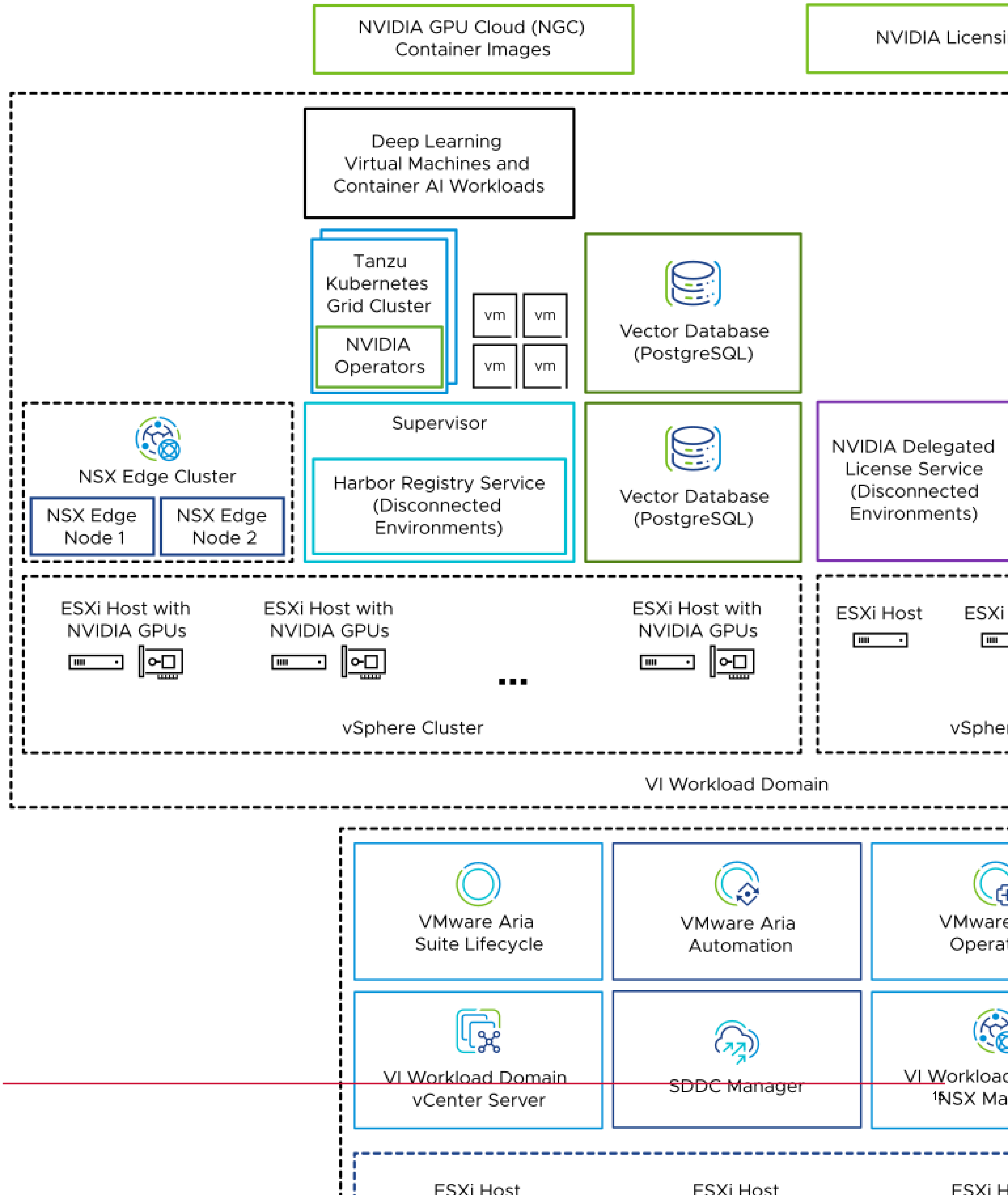
NVIDIA GPU Cloud (NGC) Container Images
NVIDIA Licensi
Deep Learning Virtual Machines and Container AI Workloads
Tanzu Kubernetes Grid Cluster
NVIDIA Operators
vm
vm
vm
vm
Vector Database (PostgreSQL)
NSX Edge Cluster
NSX Edge Node 1
NSX Edge Node 2
Supervisor
Harbor Registry Service (Disconnected Environments)
Vector Database (PostgreSQL)
NVIDIA Delegated License Service (Disconnected Environments)
ESXi Host with NVIDIA GPUs
ESXi Host with NVIDIA GPUs
ESXi Host with NVIDIA GPUs
...
vSphere Cluster
ESXi Host
ESXi
vSphe
VI Workload Domain
VMware Aria Suite Lifecycle
VMware Aria Automation
VMware Operat
VI Workload Domain vCenter Server
SDDC Manager
VI Workload NSX Ma
ESXi Host
ESXi Host
ESXi H

Table 1: Components for Running AI Workloads in VMware Private AI Foundation with NVIDIA

Component	Description
GPU-enabled ESXi hosts	ESXi hosts that configured in the following way: • Have an NVIDIA GPU that is supported for VMware Private AI Foundation with NVIDIA. The GPU is shared between workloads by using the time slicing or Multi-Instance GPU (MIG) mechanism. • Have the NVIDIA vGPU host manager driver installed so that you can use vGPU profiles based on MIG or time slicing.
Supervisor	One or more vSphere clusters enabled for vSphere with Tanzu so that you can run virtual machines and containers on vSphere by using the Kubernetes API. A Supervisor is a Kubernetes cluster itself, serving as the control plane to manage workload clusters and virtual machines.
Harbor registry	A local image registry in a disconnected environment where you host the container images downloaded from the NVIDIA NGC catalog.
NSX Edge cluster	A cluster of NSX Edge nodes that provides 2-tier north-south routing for the Supervisor and the workloads it runs. The Tier-0 gateway on the NSX Edge cluster is in active-active mode.
NVIDIA Operators	• NVIDIA GPU Operator. Automates the management of all NVIDIA software components needed to provision GPU to containers in a Kubernetes cluster. NVIDIA GPU Operator is deployed on a TKG cluster. • NVIDIA Network Operator. NVIDIA Network Operator also helps configuring the right mellanox drivers for containers using virtual functions for high speed networking, RDMA and GPUDirect. Network Operator works together with the GPU Operator to enable GPUDirect RDMA on compatible systems. NVIDIA Network Operator is deployed on a TKG cluster.
Vector database	A PostgreSQL database that has the pgvector extension enabled so that you can use it in Retrieval Augmented Generation (RAG) AI workloads.
• NVIDIA Licensing Portal • NVIDIA Delegated License Service (DLS)	You use the NVIDIA Licensing Portal to generate a client configuration token to assign a license to the guest vGPU driver in the deep learning virtual machine and the GPU Operators on TKG clusters. In a disconnected environment or to have your workloads getting license information without using an Internet connection, you host the NVIDIA licenses locally on a Delegated License Service (DLS) appliance.

Component	Description
Content library	Content libraries store the images for the deep learning virtual machines and for the Tanzu Kubernetes releases. You use these images for AI workload deployment within the VMware Private AI Foundation with NVIDIA environment. In a connected environment, content libraries pull their content from VMware managed public content libraries. In a disconnected environment, you must upload the required images manually or pull them from an internal content library mirror server.
NVIDIA GPU Cloud (NGC) catalog	A portal for GPU-optimized containers for AI, and machine learning that are tested and ready to run on supported NVIDIA GPUs on premises on top of VMware Private AI Foundation with NVIDIA.

As a cloud administrator, you use the management components in VMware Cloud Foundation

Table 2: Management Components in VMware Private AI Foundation with NVIDIA

Management Component	Description
SDDC Manager	You use SDDC Manager for the following tasks: • Deploy a GPU-enabled VI workload domain that is based vSphere Lifecycle Manager images and add clusters to it. • Deploy an NSX Edge cluster in VI workload domains for use by Supervisor instances and in the management domain for the VMware Aria Suite components of VMware Private AI Foundation with NVIDIA. • Deploy a VMware Aria Suite Lifecycle instance which is integrated with the SDDC Manager repository.
VI Workload Domain vCenter Server	You use this vCenter Server instance to enable and configure a Supervisor.
VI Workload Domain NSX Manager	SDDC Manager uses this NSX Manager to deploy and update NSX Edge clusters.
VMware Aria Suite Lifecycle	You use VMware Aria Suite Lifecycle to deploy and update VMware Aria Automation and VMware Aria Operations.
VMware Aria Automation	You use VMware Aria Automation to add self-service catalog items for deploying AI workloads for DevOps engineers and data scientists.
VMware Aria Operations	You use VMware Aria Operations for monitoring the GPU consumption in the GPU-enabled workload domains.
VMware Data Services Manager	You use VMware Data Services Manager to create vector databases, such as a PostgreSQL database with pgvector extension.

Deploying VMware Private AI Foundation with NVIDIA

As a cloud administrator, you must deploy specific software and configure the target VI workload domains so that data scientists and DevOps engineers can deploy AI workloads on top of VMware Private AI Foundation with NVIDIA.

VMware Components in VMware Private AI Foundation with NVIDIA

The functionality of the VMware Private AI Foundation with NVIDIA solution is available across several software components.

- VMware Cloud Foundation 5.1.1
- VMware Aria Automation 8.16.2 and VMware Aria Automation 8.17
- VMware Aria Operations 8.16 and VMware Aria Operations 8.17.1
- VMware Data Services Manager 2.0.x

For information about the VMware Private AI Foundation with NVIDIA architecture and components, see What is VMware Private AI Foundation with NVIDIA?.

Deployment Workflows for VMware Private AI Foundation with NVIDIA

In a disconnected environment, you must take additional steps to set up and deploy appliances and provide resources locally, so that your workloads can access them.

Connected Environment

Task	Related AI Workload Deployment Options	Steps
Review the requirements for deploying VMware Private AI Foundation with NVIDIA.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	Requirements for Deploying VMware Private AI Foundation with NVIDIA
Configure a License Service instance on the NVIDIA Licensing Portal and generate a client configuration token.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	NVIDIA License System User Guide.
Generate an API key for access to the NVIDIA NGC catalog.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	Pulling and Running NVIDIA AI Enterprise Containers
If you plan to deploy deep learning VMs or TKG cluster directly on a Supervisor in vSphere with Tanzu, set up a machine that has access to the Supervisor instance, and has Docker, Helm, and Kubernetes CLI Tools for vSphere.	• Deploy a deep learning VM directly by using `kubectl` • Deploy AI workloads on a GPU-accelerated TKG cluster that is provisioned by using `kubectl` • Deploy a RAG workload – Deploy a deep learning VM with a RAG workload by using `kubectl` – Deploy a RAG Workload on a TKG cluster	Install the Kubernetes CLI Tools for vSphere
Enable vSphere with Tanzu.	• Deploy a deep learning VM directly by using `kubectl` • Deploy AI workloads on a GPU-accelerated TKG cluster that is provisioned by using `kubectl` • Deploy a RAG workload – Deploy a deep learning VM with a RAG workload by using `kubectl` – Deploy a RAG Workload on a TKG cluster	Configure vSphere with Tanzu for VMware Private AI Foundation with NVIDIA
Deploy VMware Aria Automation.	• Deploy a deep learning VM directly by using a self-service catalog item • Deploy AI workloads on a GPU-accelerated TKG cluster that is provisioned by using a self-service catalog item • Deploy a RAG workload – Deploy a deep learning VM with a RAG workload by using a self-service catalog item – Deploy a RAG Workload on a TKG cluster that is provisioned by using a self-service catalog item	Set Up VMware Aria Automation for VMware Private AI Foundation with NVIDIA

Task	Related AI Workload Deployment Options	Steps
Deploy VMware Aria Operations.	Monitor GPU metrics at the cluster, host system and host properties with the option to add these metrics to custom dashboards.	For VMware Aria Operations 8.16, follow Intelligent Operations Management for VMware Cloud Foundation. If you want to use the extended GPU monitoring features in VMware Aria Operations 8.17.1, perform the following steps: 1. Apply the product support packs for VMware Aria Operations 8.17.1 to VMware Aria Suite Lifecycle 8.16. See VMware Aria Suite Lifecycle 8.16 Product Support Pack Release Notes. 2. Deploy VMware Aria Operations according to Intelligent Operations Management for VMware Cloud Foundation
Deploy VMware Data Services Manager	• Deploy a RAG workload	Installing and Configuring VMware Data Services Manager You deploy a VMware Data Services Manager instance in the VI workload domain with the AI workloads. To be able to provision a PostgreSQL database with pgvector extension by using a self-service catalog item in VMware Aria Automation, deploy VMware Data Services Manager 2.0.2.

Disconnected Environment

Task	Related AI Workload Deployment Options	Steps
Review the requirements for deploying VMware Private AI Foundation with NVIDIA.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	Requirements for Deploying VMware Private AI Foundation with NVIDIA
Deploy an NVIDIA Delegated License Service Instance.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	Installing and Configuring the DLS Virtual Appliance You can deploy the virtual appliance in the same workload domain as the AI workloads or in the management domain.
1. Register an NVIDIA DLS instance on the NVIDIA Licensing Portal, and bind and install a license server on it. 2. Generate a client configuration token.	• Deploy a deep learning VM • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload	• Configuring a Service Instance • Managing Licenses on a License Server.
Enable vSphere with Tanzu	• Deploy a deep learning VM directly by using `kubectl` • Deploy AI workloads on a GPU-accelerated TKG cluster that is provisioned by using `kubectl` • Deploy a RAG workload — Deploy a deep learning VM with a RAG workload by using `kubectl` — Deploy a RAG Workload on a TKG cluster	Configure vSphere with Tanzu for VMware Private AI Foundation with NVIDIA
Set up a Harbor registry service in the Supervisor.	• Deploy a deep learning VM — Deploy a deep learning VM directly by using `kubectl` — Deploy a deep learning VM directly by using a self-service catalog item • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload — Deploy a deep learning VM with a RAG workload by using `kubectl` — Deploy a deep learning VM directly by using a self-service catalog item — Deploy a RAG Workload on a TKG cluster	Setting Up a Private Harbor Registry in VMware Private AI Foundation with NVIDIA

Task	Related AI Workload Deployment Options	Steps
Provide a location to download the vGPU guest drivers from.	Deploy a deep learning VM	Upload to a local Web server the required vGPU guest driver versions and an index in one of the following formats: • An index file with a list of the `.run` files of the vGPU guest drivers. `host-driver-version-1guest-driver-download-URL-1host-driver-version-2guest-driver-download-URL-2host-driver-version-3guest-driver-download-URL-3` • A directory index in the format generated by Web servers, such as NGINX and Apache HTTP Server.
Upload the NVIDIA NGC container images to a private container registry, such as the Harbor Registry service of the Supervisor.	• Deploy a deep learning VM – Deploy a deep learning VM directly by using `kubectl` – Deploy a deep learning VM directly by using a self-service catalog item • Deploy AI workloads on a GPU-accelerated TKG cluster • Deploy a RAG workload – Deploy a deep learning VM with a RAG workload by using `kubectl` – Deploy a deep learning VM directly by using a self-service catalog item – Deploy a RAG Workload on a TKG cluster	Upload AI Container Images to a Private Harbor Registry in VMware Private AI Foundation with NVIDIA
Deploy VMware Aria Automation	• Deploy a deep learning VM directly by using a self-service catalog item • Deploy AI workloads on a GPU-accelerated TKG cluster that is provisioned by using a self-service catalog item • Deploy a RAG workload – Deploy a deep learning VM directly by using a self-service catalog item – Deploy a RAG Workload on a TKG cluster that is provisioned by using a self-service catalog item	Set Up VMware Aria Automation for VMware Private AI Foundation with NVIDIA

Task	Related AI Workload Deployment Options	Steps
Deploy VMware Aria Operations	Monitor GPU metrics at the cluster, host system and host properties with the option to add these metrics to custom dashboards.	For VMware Aria Operations 8.16, follow Intelligent Operations Management for VMware Cloud Foundation. If you want to use the extended GPU monitoring features in VMware Aria Operations 8.17.1, perform the following steps: 1. Apply the product support packs for VMware Aria Operations 8.17.1 to VMware Aria Suite Lifecycle 8.16. See VMware Aria Suite Lifecycle 8.16 Product Support Pack Release Notes. 2. Deploy VMware Aria Operations according to Intelligent Operations Management for VMware Cloud Foundation.
Deploy VMware Data Services Manager	• Deploy a RAG workload	Installing and Configuring VMware Data Services Manager You deploy a VMware Data Services Manager instance in the VI workload domain with the AI workloads. To be able to provision a PostgreSQL database with pgvector extension by using a self-service catalog item in VMware Aria Automation, deploy VMware Data Services Manager 2.0.2.
• Set up a machine that has access to the Internet and has Docker and Helm installed. • Set up a machine that has access to vCenter Server for the VI workload domain, the Supervisor instance, and the local container registry. The machine must have Docker, Helm, and Kubernetes CLI Tools for vSphere.	• Deploy a deep learning VM • Deploy a GPU-accelerated TKG cluster • Deploy a RAG workload	• Install the Kubernetes CLI Tools for vSphere • Installing VMware vSphere with VMware Tanzu (Air-gapped)

Requirements for Deploying VMware Private AI Foundation with NVIDIA

You deploy components of VMware Private AI Foundation with NVIDIA in your VMware Cloud Foundation environment in a VI workload domain where you must have certain NVIDIA components installed.

Required VMware Software Versions

See VMware Components in VMware Private AI Foundation with NVIDIA.

Supported NVIDIA GPU Devices

Before you start using VMware Private AI Foundation with NVIDIA, make sure that the GPUs on your ESXi hosts are supported by VMware by Broadcom:

Table 3: Supported NVIDIA Components for VMware Private AI Foundation with NVIDIA

NVIDIA Component	Supported Options
NVIDIA GPUs	• NVIDIA A100 • NVIDIA L40S • NVIDIA H100
GPU sharing mode	• Time slicing • Multi-Instance GPU (MIG)

Required NVIDIA Software

The GPU device must support the latest NVIDIA AI Enterprise (NVAIE) vGPU profiles. See the NVIDIA Virtual GPU Software Supported GPUs documentation for guidance.

- NVIDIA vGPU host driver (including the VIB for ESXi hosts), that is compatible with your VMware Cloud Foundation version. See Virtual GPU Software for VMware vSphere Release Notes.
- NVIDIA GPU Operator that is compatible with the Kubernetes version of the deployed TKG clusters. See NVIDIA GPU Operator Release Notes and VMware Tanzu Kubernetes releases Release Notes.

Required VMware Cloud Foundation Setup

Before you deploy VMware Private AI Foundation with NVIDIA, a specific configuration must be available in VMware Cloud Foundation.

- VMware Cloud Foundation
  VMware Cloud Foundation on Dell VxRail is not supported.
- A VMware Cloud Foundation license.
- A VMware Private AI Foundation with NVIDIA add-on license.
- Licensed NVIDIA vGPU product including the host driver VIB file for ESXi hosts and the guest OS drivers. See the NVIDIA Virtual GPU Software Supported GPUs documentation for guidance.
- The VIB file of the NVIDIA vGPU host driver downloaded from https://nvid.nvidia.com/
- A vSphere Lifecycle Manager image with the VIB file of the vGPU host manager driver available in SDDC Manager. See Managing vSphere Lifecycle Manager Images in VMware Cloud Foundation.
- A VI workload domain with at least 3 ESXi GPU-enabled hosts that is based on the vSphere Lifecycle Manager image containing the host manager driver VIB file. See Deploy a VI Workload Domain Using the SDDC Manager UI and Managing vSphere Lifecycle Manager Images in VMware Cloud Foundation.
- NVIDIA vGPU host driver installed and vGPU configured on each ESXi host in the cluster for AI workloads.
  a. On each ESXi host, enable SR-IOV in the BIOS and Shared Direct on the graphics devices for AI operations.
     For information about configuring SR-IOV, see the documentation from your hardware vendor. For information about configuring Shared Direct on graphics devices, see Configure Virtual Graphics on vSphere.
  b. Install the NVIDIA vGPU host manager driver on each ESXi host in one of the following ways:
     - Install the driver on each host and add the VIB file of the driver to the vSphere Lifecycle image for the cluster. See NVIDIA Virtual GPU Software Quick Start Guide.
     - Add the VIB file of the driver to the vSphere Lifecycle image for the cluster and remediate the hosts.
  c. If you want to use the Multi-Instance GPU (MIG) sharing, enable it on each ESXi host in the cluster.
     See NVIDIA MIG User Guide.
  d. On the vCenter Server instance for VI workload domain, set the `vgpu.hotmigrate.enabled` advanced setting to `true` so that virtual machines with vGPU can be migrated by using vSphere vMotion.
     See Configure Advanced Settings.

Configure vSphere with Tanzu for VMware Private AI Foundation with NVIDIA

To provide DevOps engineers and data scientists with the ability to deploy deep learning virtual machines or TKG clusters with AI container workloads, you must deploy a Supervisor on a GPU-enabled cluster in a VI workload domain and create vGPU-enabled VM classes.

See Requirements for Deploying VMware Private AI Foundation with NVIDIA.

1. Deploy an NSX Edge Cluster in the VI workload domain by using SDDC Manager.

   SDDC Manager also deploys a Tier-0 gateway that you specify at Supervisor deployment. The Tier-0 gateway is in active-active high availability mode.

2. Configure a storage policy for the Supervisor.

   See Create Storage Policies for vSphere with Tanzu.

3. Deploy a Supervisor on a cluster of GPU-enabled ESXi hosts in the VI workload domain.

   You use static IP address assignment for the management network. Assign the supervisor VM management network on the vSphere Distributed Switch for the cluster.

   Configure the workload network in the following way:

   - Use the vSphere Distributed Switch for the cluster or create one specifically for AI workloads.
   - Configure the Supervisor with the NSX Edge cluster and Tier-0 gateway that you deployed by using SDDC Manager.
   - Set the rest of the values according to your design.

   Use the storage policy you created.

   For more information on deploying a supervisor on a single cluster, see Deploy a One-Zone Supervisor with NSX Networking.

4. Configure vGPU-based VM classes for AI workloads.

   In these VM classes, you set the compute requirements and a vGPU profile for an NVIDIA GRID vGPU device according to the vGPU devices configured on the ESXi hosts in the Supervisor cluster.

   - For information about setting up vGPU-based VM classes for virtual machines, see Create a Custom VM Class Using the vSphere Client and Add PCI Devices to a VM Class in vSphere with Tanzu.
   - For information about setting up vGPU-based VM classes for TKG worker nodes, see Create a Custom VM Class with a vGPU Profile in vSphere 8 Update 2b and later and Configuring vSphere Namespaces for TKG Clusters on Supervisor .

   For the VM class for deploying deep learning VMs with NVIDIA RAG workloads, set the following additional settings in the VM class dialog box:

   - Select the full-sized vGPU profile for time-slicing mode or a MIG profile. For example, for NVIDIA A100 40GB card in vGPU time-slicing mode, select **nvidia_a100-40c**.
   - On the **Virtual Hardware** tab, allocate more than 16 virtual CPU cores and 64 GB of virtual memory.
   - On the **Advanced Parameters** tab, set the `pciPassthru<vgpu-id>.cfg.enable_uvm` parameter to `1`.
     where `<vgpu-id>` identifies the vGPU assigned to the virtual machine. For example, if two vGPUs are assigned to the virtual machine, you set `pciPassthru0.cfg.parameter=1` and `pciPassthru1.cfg.parameter = 1` .

5. If you plan to use the `kubectl` command line tool to deploy a deep learning VM or an GPU-accelerated TKG cluster on a Supervisor, create and configure a vSphere namespace, adding resource limits, storage policy, permissions for DevOps engineers, and associating the vGPU-based VM classes with it.
   - For information about setting up vSphere namespaces for virtual machines, see Create and Configure a vSphere Namespace on the Supervisor.
   - For information about setting up vSphere namespaces for TKG clusters, see Configuring vSphere Namespaces for TKG Clusters on Supervisor.

Setting Up a Private Harbor Registry in VMware Private AI Foundation with NVIDIA

You can use Harbor as a local registry for container images from the NVIDIA NGC catalog.

> **NOTE**
> The installation of the Harbor service in the Supervisor requires an Internet connection.

If you want to use the Harbor registry integration with Supervisor, you can follow these setup approaches:

- Use a Harbor registry only in the Supervisor in the GPU-enabled workload domain. Perform the following tasks:
  a. Enable Harbor as a Supervisor Service.
  b. Upload AI Container Images to a Private Harbor Registry in VMware Private AI Foundation with NVIDIA

  You can disconnect your environment from the Internet and start using the Harbor service as a local container registry after you install the service or after you install it and download the initial set of required container images.

  In this approach, you must manually download container images from the NVIDIA NGC catalog to a machine in the environment and then upload them to the registry.
- Use a Harbor registry that is as a replica of an Internet-connected Harbor registry.

  One Harbor registry, running outside the VMware Private AI Foundation with NVIDIA environment, is always connected to the Internet. The Harbor registry in the Supervisor for the GPU-enabled workload domain receives container images from the connected one using a proxy mechanism. In this way, the main components of the VMware Cloud Foundation instance remain isolated.

  In this approach, additional resources are required for the connected registry.

  > **NOTE**
  > Allocate enough storage space for hosting the NVIDIA NGC containers you plan to deploy on a deep learning VM or on a TKG cluster. Accommodate at least three versions of each container in the storage space.

If connecting to the Internet while installing the Harbor service or setting up a connected Harbor registry is not an option for your organization, use a container registry by another vendor.

Upload AI Container Images to a Private Harbor Registry in VMware Private AI Foundation with NVIDIA

In a disconnected environment where you use a Harbor registry only on the AI-ready Supervisor, you must manually upload the AI container images that you intend to deploy on a deep learning VM or a TKG cluster from the NVIDIA NGC catalog to Harbor.

1. On the machines for access to NVIDIA NGC and to the disconnected VMware Cloud Foundation instance, configure the Docker client with the certificate of the Harbor Registry.

   See Configure a Docker Client with a Registry Certificate.

2. Log in to NVIDA NGC.

   Use the reserved user name of $oauthtoken and paste the API key in the password field.

   ```
   docker login nvcr.io
   ```

3. Pull the required container images to the machine with access to NVIDIA NGC catalog and save them to an archive.

   For example, to download the CUDA Sample container, run the following commands.

   ```
   docker pull nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8
   docker save > cuda-sample.tar nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8
   ```

4. Copy the archive to the machine with access to the local container registry.
5. On the machine with access to the local container registry, load the container image.

   ```
   docker load < cuda-sample.tar
   ```

6. Push the container images to the Harbor Registry.

Create a Harbor Registry in VMware Private AI Foundation with NVIDIA as a Replica of a Connected Registry

To be able to update easily to the latest images in the NVIDIA NGC catalog, you can use a Harbor registry in a Supervisor that is in another VI workload domain or VMware Cloud Foundation instance and can be connected to Internet. You then replicate this connected registry on the Supervisor where you plan to run AI workloads.

Enable Harbor as a Supervisor Service in the Supervisor in the GPU-enabled workload domain.

You pull the latest container images from NVIDIA NGC to the connected Harbor registry and transfer them to the disconnected one by using a proxy-cached connection. In this way, you do not have to download container images and then upload them manually on a frequent basis.

NOTE
You can also use a connected container registry by another vendor.

You set up the network between the two registries in the following way:

- The connected registry is routable to the replica registry.
- The connected registry is placed in a DMZ where only `docker push` and `docker pull` communication is allowed between the two registries.

1. Log in to the connected Harbor Registry UI as a Harbor system administrator.
2. Go to the **Administration** > **Registries** page and create an endpoint for the NVIDIA NGC catalog `nvcr.io/nvaie` selecting the **Docker Registry** provider and with your NVIDIA NGC API key.
3. Go to the **Administration** > **Projects** page and create a proxy-cache project, connected to the endpoint for `nvcr.io/nvaie`.
4. Back on the **Registries** page, create a replication endpoint for the disconnected registry, selecting the **Harbor** provider.
5. Go to the **Administration** > **Replications** page and create a replication rule.
   - Use push-based replication mode.
   - In the **Destination registry** property, enter the URL of the disconnected registry on the AI-ready Supervisor.
   - Set filters, target namespace and trigger mode according to the requirements of your organization.

1. Pull the container images that are required by your organization from NVIDIA NGC to the connected registry by running `docker pull` on the Docker client machine.
2. If the replication rule has manual trigger mode, run manually replications as needed.

Set Up VMware Aria Automation for VMware Private AI Foundation with NVIDIA

VMware Aria Automation provides support for self-service catalog items that DevOps engineers and data scientists can use to provision AI workloads in VMware Private AI Foundation with NVIDIA in a user-friendly and customizable way.

Deploy VMware Aria Automation for VMware Private AI Foundation with NVIDIA

The VMware Private AI Foundation with NVIDIA features are available in VMware Aria Automation 8.16.2 and VMware Aria Automation 8.17. You use VMware Aria Suite Lifecycle in VMware Cloud Foundation mode to deploy it.

By using SDDC Manager, create the following components in VMware Cloud Foundation:

- Deploy an NSX Edge cluster in the management domain. See Deploy an NSX Edge Cluster in VMware Cloud Foundation in the *VMware Cloud Foundation Administration Guide*.
- Create Application Virtual Networks. See Deploying Application Virtual Networks in VMware Cloud Foundation in the *VMware Cloud Foundation Administration Guide*.
- Deploy VMware Aria Suite Lifecycle in VMware Cloud Foundation mode. See VMware Aria Suite Lifecycle Implementation in the *VMware Cloud Foundation Administration Guide*.

Because the features for VMware Private AI Foundation with NVIDIA are available in a VMware Aria Automation version that is later than the version supported by VMware Aria Suite Lifecycle delivered with VMware Cloud Foundation 5.1.1, you must apply a patch to VMware Aria Suite Lifecycle before you deploy VMware Aria Automation.

1. Apply the product support packs for VMware Aria Automation 8.16.2 or VMware Aria Automation 8.17 to VMware Aria Suite Lifecycle 8.16.

   See VMware Aria Suite Lifecycle 8.16 Product Support Pack Release Notes.

2. Deploy VMware Aria Automation 8.16.2 or VMware Aria Automation 8.17 by using VMware Aria Suite Lifecycle.

   See Private Cloud Automation for VMware Cloud Foundation.

Upgrade VMware Aria Automation for VMware Private AI Foundation with NVIDIA

After you upgrade VMware Aria Automation to the version in VMware Cloud Foundation 5.1.1, you must update VMware Aria Automation once again to a version supported for VMware Private AI Foundation with NVIDIA.

- Upgrade VMware Aria Suite Lifecycle and VMware Aria Automation to their versions in VMware Cloud Foundation 5.1.1. See VMware Cloud Foundation Lifecycle Management.

Because the features for VMware Private AI Foundation with NVIDIA are available in VMware Aria Automation versions that are later than the version supported by VMware Aria Suite Lifecycle delivered with VMware Cloud Foundation 5.1.1, after you complete the upgrade of the VMware Cloud Foundation 5.1.1 components, you must update it explicitly by using VMware Aria Suite Lifecycle.

1. Apply the product support packs for VMware Aria Automation 8.16.2 or VMware Aria Automation 8.17 to VMware Aria Suite Lifecycle 8.16.

   See VMware Aria Suite Lifecycle 8.16 Product Support Pack Release Notes.

2. Update to VMware Aria Automation 8.16.2 or VMware Aria Automation 8.17.

   See Upgrade VMware Aria Automation by using VMware Aria Suite Lifecycle.

Connect VMware Aria Automation to a Workload Domain for VMware Private AI Foundation with NVIDIA

You use the Quickstart wizard for VMware Cloud Foundation to create the required constructs for provisioning workloads on a GPU-enabled workload domain in a VMware Cloud Foundation instance.

You must have VMware Aria Automation 8.16.2 running in the management domain. See Deploy VMware Aria Automation for VMware Private AI Foundation with NVIDIA and Upgrade VMware Aria Automation for VMware Private AI Foundation with NVIDIA.

In the VMware Aria Automation UI, run the VMware Cloud Foundation Quickstart wizard.

See How do you get started using the VMware Cloud Foundation Quickstart in the *VMware Aria Automation Product Documentation*.

Deploying a Deep Learning VM in VMware Private AI Foundation with NVIDIA

VMware Private AI Foundation with NVIDIA supports provisioning deep learning VMs that data scientists can use for AI development.

About Deep Learning VM Images in VMware Private AI Foundation with NVIDIA

The deep learning virtual machine images delivered as part of VMware Private AI Foundation with NVIDIA are preconfigured with popular ML libraries, frameworks, and toolkits, and are optimized and validated by NVIDIA and VMware for GPU acceleration in a VMware Cloud Foundation environment.

Data scientists can use the deep learning virtual machines provisioned from these images for AI prototyping, fine tuning, validation, and inference.

The software stack for running AI applications on top of NVIDIA GPUs is validated in advance. As a result, you directly start AI developing, without spending time installing and validating the compatibility of operating systems, software libraries, ML frameworks, toolkits, and GPU drivers.

What Does a Deep Learning VM Image Contain?

The initial deep learning virtual machine image contains the following software. For information on the component versions in each deep learning VM image release, see VMware Deep Learning VM Image Release Notes.

Software Component Category	Software Component
Embedded	• Canonical Ubuntu • NVIDIA Container Toolkit • Docker Community Engine
Can be pre-installed automatically	• vGPU guest driver according to the version of the vGPU host driver • Deep learning (DL) workloads, such as PyTorch, TensorFlow, and RAG. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA.

Content Library for Deep Learning VM Images

Deep learning VM images are delivered as vSphere VM templates, hosted and published by VMware in a content library. You can use these images to deploy Deep learning VM by using the vSphere Client or VMware Aria Automation.

The content library with deep learning VM images for VMware Private AI Foundation with NVIDIA is available at the https://packages.vmware.com/dl-vm/lib.json URL. In a connected environment, you create a subscribed content library connected to this URL, and in a disconnected environment - a local content library where you upload images downloaded from the central URL.

OVF Properties of Deep Learning VMs

When you deploy a deep learning VM, you must fill in custom VM properties to automate the configuration of the Linux operating system, the deployment of the vGPU guest driver, and the deployment and configuration of NGC containers for the DL workloads.

Category	Parameter	Description
Base OS Properties	instance-id	Required. A unique instance ID for the VM instance. An instance ID uniquely identifies an instance. When an instance ID changes, cloud-init treats the instance as a new instance and runs the cloud-init process to again.
	hostname	Required. The host name of the appliance.
	public-keys	If provided, the instance populates the default user's SSH `authorized_keys` with this value.
	user-data	A set of scripts or other metadata that is inserted into the VM at provisioning time. This property is the actual the `cloud-init` script. This value must be base64 encoded. • You can use this property to specify the DL workload container you want to deploy, such as PyTorch or TensorFlow. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA. • You use this property to set a static IP address to a virtual machine that is deployed directly on a vSphere cluster. See Assign a Static IP Address to a Deep Learning VM in VMware Private AI Foundation with NVIDIA.
	password	Required. The password for the default **vmware** user account.
vGPU Driver Installation	vgpu-license	Required. The NVIDIA vGPU client configuration token. The token is saved in the `/etc/nvidia/ClientConfigToken/client_configuration_token.tok` file.
	nvidia-portal-api-key	Required in a connected environment. The API key you downloaded from the NVIDIA Licensing Portal. The key is required for vGPU guest driver installation.
	vgpu-fallback-version	The version of the vGPU guest driver to fall back to if the version of the vGPU guest driver cannot be determined by using the entered license API key.

Category	Parameter	Description
	vgpu-url	Required in a disconnected environment. The URL to download the vGPU guest driver from.
DL Workload Automation	registry-uri	Required in a disconnected environment or if you plan to use a private container registry to avoid downloading images from the Internet. The URI of a private container registry with the deep learning workload container images. Required if you are referring to a private registry in `user-data` or `image-oneliner`.
	registry-user	Required if you are using a private container registry that requires basic authentication.
	registry-passwd	Required if you are using a private container registry that requires basic authentication.
	registry-2-uri	Required if you are using a second private container registry that is based on Docker and required basic authentication.
	registry-2-user	Required if you are using a second private container registry.
	registry-2-passwd	Required if you are using a second private container registry.
	`image-oneliner`	A one-line bash command that is run at VM provisioning. This value must be base64 encoded. You can use this property to specify the DL workload container you want to deploy, such as PyTorch or TensorFlow. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA. **NOTE** If both `user-data` and `image-oneliner` are provided, the value of `user-data` is used.
	`docker-compose-uri`	URI of the Docker compose file. Required if you need a Docker compose file to start the DL workload container. This value must be base64 encoded.
	`config-json`	Configuration file for multiple container registry login operations when using a Docker compose file. This value must be base64 encoded.

Assign a Static IP Address to a Deep Learning VM in VMware Private AI Foundation with NVIDIA

By default, the deep learning VM images are configured with DHCP address assignment. If you want to deploy a deep learning VM with a static IP address directly on a vSphere cluster, you must add additional code to the cloud-init section.

On vSphere with Tanzu, IP address assignment is determined by the network configuration for the Supervisor in NSX.

1. Create a cloud-init script in plain-text format for the DL workload you plan to use.

   See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA.

2. Add the following additional code to the cloud-init script.

```
#cloud-config
<instructions_for_your_DL_workload>

manage_etc_hosts: true

write_files:
  - path: /etc/netplan/50-cloud-init.yaml
    permissions: '0600'
    content: |
      network:
        version: 2
        renderer: networkd
        ethernets:
          ens33:
            dhcp4: false # disable DHCP4
            addresses: [x.x.x.x/x]  # Set the static IP address and mask
            routes:
                - to: default
                  via: x.x.x.x # Configure gateway
            nameservers:
              addresses: [x.x.x.x, x.x.x.x] # Provide the DNS server address. Separate mulitple DNS server
 addresses with commas.

runcmd:
  - netplan apply
```

3. Encode the resulting cloud-init script in base64 format.

4. Set the resulting cloud-init script in base64 format as a value to the `user-data` OVF parameter of the deep learning VM image.

   Assigning a Static IP Address to a CUDA Sample Workload

   For an example deep learning VM with a CUDA Sample DL workload:

Deep Learning VM Element	Example Value
DL workload image	nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8
IP address	10.199.118.245
Subnet prefix	/25

Deep Learning VM Element	Example Value
Gateway	10.199.118.253
DNS servers	• 10.142.7.1 • 10.132.7.1

you provide the following cloud-init code:

```
I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybWlzc2lvbn-
M6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1kIG52Y3IuaW8vbnZpZGlh-
L2s4cy9jdWRhLXNhbXBsZTp2ZWN0b3JhZGQtY3VkYTExLjcuMS11Ymk4CgptYW5hZ2VfZXRjX2hvc3RzOiB0c-
nVlCiAKd3JpdGVfZmlsZXM6CiAgLSBwYXRoOiAvZXRjL25ldHBsYW4vNTAtY2xvdWQtaW5pdC55YW1sCiAgICB-
wZXJtaXNzaW9uczogJzA2MDAnCiAgICBjb250ZW50OiB8CiAgICAgIG5ldHdvcms6CiAgICAgICAgdmVyc2lvb-
jogMgogICAgICAgIHJlbmRlcmVyOiBuZXR3b3JrZAogICAgICAgIGV0aGVybmV0czoKICAgICAgICAgIGVuczMzO-
gogICAgICAgICAgICBkaGNwNDogZmFsc2UgIyBkaXNhYmxlIERIQ1A0CiAgICAgICAgICAgIGFkZHJlc3NlczogWzEwL-
jE5OS4xMTguMjQ1LzI1XSAgIyBTZXQgdGhlIHN0YXRpYyBJUCBhZGRyZXNzIGFuZCBtYXNrCiAgICAgICAgICAgIHJvdXRl-
czoKICAgICAgICAgICAgICAgIC0gdG86IGRlZmF1bHQKICAgICAgICAgICAgICAgICAgdmlhOiAxMC4xOTkuMTE4LjI1MyA-
jIENvbmZpZ3VyZSBnYXRld2F5CiAgICAgICAgICAgIG5hbWVzZXJ2ZXJzOgogICAgICAgICAgICAgIGFkZHJlc3NlczogWzEwL-
jE0Mi43LjEsIDEwLjEzMi43LjFdICMgUHJvdmlkZSB0aGUgRE5TIHNlcnZlciBhZGRyZXNzLiBTZXBhcmF0ZSBtdWxpdHBsZS-
BETlMgc2VydmVyIGFkZHJlc3NlcyB3aXRoIGNvbW1hcy4KIApydW5jbWQ6CiAgLSBuZXRwbGFuIGFwcGx5
```

which corresponds to the following script in plain-text format:

```
#cloud-config
write_files:
- path: /opt/dlvm/dl_app.sh
  permissions: '0755'
  content: |
    #!/bin/bash
    docker run -d nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8

manage_etc_hosts: true

write_files:
  - path: /etc/netplan/50-cloud-init.yaml
    permissions: '0600'
    content: |
      network:
        version: 2
        renderer: networkd
        ethernets:
          ens33:
            dhcp4: false # disable DHCP4
            addresses: [10.199.118.245/25]  # Set the static IP address and mask
            routes:
                - to: default
                  via: 10.199.118.253 # Configure gateway
            nameservers:
              addresses: [10.142.7.1, 10.132.7.1] # Provide the DNS server address. Separate mulit-
ple DNS server addresses with commas.

runcmd:
  - netplan apply
```

Deep Learning Workloads in VMware Private AI Foundation with NVIDIA

You can provision a deep learning virtual machine with a supported deep learning (DL) workload in addition to its embedded components. The DL workloads are downloaded from the NVIDIA NGC catalog and are GPU-optimized and validated by NVIDIA and VMware by Broadcom.

For an overview of the deep learning VM images, see About Deep Learning VM Images in VMware Private AI Foundation with NVIDIA.

CUDA Sample

You can use a deep learning VM with running CUDA samples to explore vector addition, gravitational n-body simulation, or other examples on a VM. See the CUDA Samples page.

After the deep learning VM is launched, it runs a CUDA sample workload to test the vGPU guest driver. You can examine the test output in the `/var/log/dl.log` file.

Table 4: CUDA Sample Container Image

Component	Description
Container image	`nvcr.io/nvidia/k8s/cuda-sample:`*`ngc_image_tag`*
	For example:
	`nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8`
	For information on the CUDA Sample container images that are supported for deep learning VMs, see VMware Deep Learning VM Image Release Notes.
Required inputs	To deploy a CUDA Sample workload, you must set the OVF properties for the deep learning virtual machine in the following way:
	• Use one of the following properties that are specific for the CUDA Sample image.
	– Cloud-init script. Encode it in base64 format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d nvcr.io/nvidia/k8s/cuda-sample:`*`ngc_image_tag`*
	For example, for vectoradd-cuda11.7.1-ubi8, provide the following script in base64 format:
	`I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybW`
	`lzc2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1k`
	`IG52Y3IuaW8vbnZpZGlhL2s4cy9jdWRhLXNhbXBsZTp2ZWN0b3JhZGQtY3VkYTExLjcuMS11Ymk4`
	which corresponds to the following script in plain-text format:
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8`
	– Image one-liner. Encode it in base64 format
	`docker run -d nvcr.io/nvidia/k8s/cuda-sample:`*`ngc_image_tag`*
	For example, for vectoradd-cuda11.7.1-ubi8, provide the following script in base64 format:
	`ZG9ja2VyIHJ1biAtZCBudmNyLmlvL252aWRpYS9rOHMvY3VkYS1zYW1wbGU6dmVjdG9yYWRkLWN1ZGExMS`
	`43LjEtdWJpOA==`
	which corresponds to the following script in plain-text format:
	`docker run -d nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubi8`
	• Enter the vGPU guest driver installation properties.
	• Provide values for the properties required for a disconnected environment as needed.
	See About Deep Learning VM Images in VMware Private AI Foundation with NVIDIA.
Output	• Installation logs for the vGPU guest driver in `/var/log/vgpu-install.log`.
	To verify that the vGPU guest driver is installed, and the license is allocated, run the following command:
	`nvidia-smi -q |grep -i license`
	• Cloud-init script logs in `/var/log/dl.log`.

PyTorch

You can use a deep learning VM with a PyTorch library to explore conversational AI, NLP, and other types of AI models, on a VM. See the PyTorch page.

After the deep learning VM is launched, it starts a JupyterLab instance with PyTorch packages installed and configured.

Table 5: PyTorch Container Image

Component	Description
Container image	`nvcr.io/nvidia/pytorch:`*`ngc_image_tag`*
	For example:
	`nvcr.io/nvidia/pytorch:23.10-py3`
	For information on the PyTorch container images that are supported for deep learning VMs, see [VMware Deep Learning VM Image Release Notes]().
Required inputs	To deploy a PyTorch workload, you must set the OVF properties for the deep learning virtual machine in the following way:
	• Use one of the following properties that are specific for the PyTorch image.
	– Cloud-init script. Encode it in base64 format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d -p 8888:8888 nvcr.io/nvidia/pytorch:`*`ngc_image_tag`*`/usr/local/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	For example, for pytorch:23.10-py3, provide the following script in base 64 format:
	`I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybWlzc2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1kIC1wIDg4ODg6ODg4OCBudmNyLmlvL252aWRpYS9weXRvcmNoOjIzLjEwLXB5MyAvdXNyL2xvY2FsL2Jpbi9qdXB5dGVyIGxhYiAtLWFsbG93LXJvb3QgLS1pcD0qIC0tcG9ydD04ODg4IC0tbm8tYnJvd3NlciAtLU5vdGVib29rQXBwLnRva2VuPScnIC0tTm90ZWJvb2tBcHAuYWxsb3dfb3JpZ2luPScqJyAtLW5vdGVib29rLWRpcj0vd29ya3NwYWNl`
	which corresponds to the following script in plain-text format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d -p 8888:8888 nvcr.io/nvidia/pytorch:23.10-py3 /usr/local/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	– Image one-liner. Encode it in base64 format.
	`docker run -d -p 8888:8888 nvcr.io/nvidia/pytorch:`*`ngc_image_tag`*`/usr/local/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	For example, for pytorch:23.10-py3, provide the following script in base 64 format:
	`ZG9ja2VyIHJ1biAtZCAtcCA4ODg4Ojg4ODggbnZjci5pby9udmlkaWEvcHl0b3JjaDoyMy4xMC1weTMgL3Vzci9sb2NhbC9iaW4vanVweXRlciBsYWIgLS1hbGxvdy1yb290IC0taXA9KiAtLXBvcnQ9ODg4OCAtLW5vLWJyb3dzZXIgLS1Ob3RlYm9va0FwcC50b2tlbj0nJyAtLU5vdGVib29rQXBwLmFsbG93X29yaWdpbj0nKicgLS1ub3RlYm9vay1kaXI9L3dvcmtzcGFjZQ==`
	which corresponds to the following script in plain-text format:
	`docker run -d -p 8888:8888 nvcr.io/nvidia/pytorch:23.10-py3 /usr/local/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --Note-bookApp.allow_origin='*' --notebook-dir=/workspace`
	• Enter the vGPU guest driver installation properties.
	• Provide values for the properties required for a disconnected environment as needed.
	See [#unique_24]().
Output	• Installation logs for the vGPU guest driver in `/var/log/vgpu-install.log`.

TensorFlow

You can use a deep learning VM with a TensorFlow library to explore conversational AI, NLP, and other types of AI models, on a VM. See the TensorFlow page.

After the deep learning VM is launched, it starts a JupyterLab instance with TensorFlow packages installed and configured.

Table 6: TensorFlow Container Image

Component	Description

Container image	`nvcr.io/nvidia/tensorflow:`*`ngc_image_tag`*
	For example:
	`nvcr.io/nvidia/tensorflow:23.10-tf2-py3`
	For information on the TensorFlow container images that are supported for deep learning VMs, see VMware Deep Learning VM Image Release Notes.

Required inputs	To deploy a TensorFlow workload, you must set the OVF properties for the deep learning virtual machine in the following way:
	• Use one of the following properties that are specific for the TensorFlow image.
	– Cloud-init script. Encode it in base64 format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d -p 8888:8888 nvcr.io/nvidia/tensorflow:`*`ngc_image_tag`*`/usr/lo-`
	`cal/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.to-`
	`ken='' --NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	For example, for tensorflow:23.10-tf2-py3, provide the following script in base64 format:
	`I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybW`
	`lzc2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1k`
	`IC1wIDg4ODg6ODg4OCBudmNyLmlvL252aWRpYS90ZW5zb3JmbG93OjIzLjEwLXRmMi1weTMgL3Vzci9sb2`
	`NhbC9iaW4vanVweXRlciBsYWIgLS1hbGxvdy1yb290IC0taXA9KiAtLXBvcnQ9ODg4OCAtLW5vLWJyb3dz`
	`ZXIgLS1Ob3RlYm9va0FwcC50b2tlbj0nJyAtLU5vdGVib29rQXBwLmFsbG93X29yaWdpbj0nKicgLS1ub3`
	`RlYm9vay1kaXI9L3dvcmtzcGFjZQ==`
	which corresponds to the following script in plain-text format:
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d -p 8888:8888 nvcr.io/nvidia/tensorflow:23.10-tf2-py3 /usr/lo-`
	`cal/bin/jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.to-`
	`ken='' --NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	– Image one-liner. Encode it in base64 format.
	`docker run -d -p 8888:8888 nvcr.io/nvidia/tensorflow:`*`ngc_image_tag`*`/usr/local/bin/`
	`jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --`
	`NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	For example, for tensorflow:23.10-tf2-py3, provide the following script in base64 format:
	`ZG9ja2VyIHJ1biAtZCAtcCA4ODg4Ojg4ODggbnZjci5pby9udmlkaWEvdGVuc29yZmxvdzoyMy4xMC10Zj`
	`ItcHkzIC91c3IvbG9jYWwvYmluL2p1cHl0ZXIgbGFiIC0tYWxsb3ctcm9vdCAtLWlwPSogLS1wb3J0PTg4`
	`ODggLS1uby1icm93c2VyIC0tTm90ZWJvb2tBcHAudG9rZW49JycgLS1Ob3RlYm9va0FwcC5hbGxvd19vcm`
	`lnaW49JyonIC0tbm90ZWJvb2stZGlyPS93b3Jrc3BhY2U=`
	which corresponds to the following script in plain-text format:
	`docker run -d -p 8888:8888 nvcr.io/nvidia/tensorflow:23.10-tf2-py3 /usr/local/bin/`
	`jupyter lab --allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --`
	`NotebookApp.allow_origin='*' --notebook-dir=/workspace`
	• Enter the vGPU guest driver installation properties.
	• Provide values for the properties required for a disconnected environment as needed.
	See #unique_24.

Output	• Installation logs for the vGPU guest driver in `/var/log/vgpu-install.log`. To verify that the vGPU guest driver is installed, log in to the VM over SSH and run the `nvidia-smi` command.

DCGM Exporter

You can use a deep learning VM with a Data Center GPU Manager (DCGM) exporter to monitor the health of and get metrics from GPUs used by a DL workload, using NVIDIA DCGM, Prometheus, and Grafana.

See the DCGM Exporter page.

In a deep learning VM, you run the DCGM Exporter container together with a DL workload that performs AI operations. After the deep learning VM is started, DCGM Exporter is ready to collect vGPU metrics and export the data to another application for further monitoring and visualization. You can run the monitored DL workload as a part of the cloud-init process or from the command line after the virtual machine is started.

Table 7: DCGM Exporter Container Image

Component	Description

Container image	`nvcr.io/nvidia/k8s/dcgm-exporter:`*`ngc_image_tag`*
	For example:
	`nvcr.io/nvidia/k8s/dcgm-exporter:3.2.5-3.1.8-ubuntu22.04`
	For information on the DCGM Exporter container images that are supported for deep learning VMs, see VMware Deep Learning VM Image Release Notes.

Required inputs	To deploy a DCGM Exporter workload, you must set the OVF properties for the deep learning virtual machine in the following way:
	• Use one of the following properties that are specific for the DCGM Exporter image.
	– Cloud-init script. Encode it in base64 format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d --gpus all --cap-add SYS_ADMIN --rm -p 9400:9400 nvcr.io/nvidia/`
	`k8s/dcgm-exporter:`*`ngc_image_tag`*`-ubuntu22.04`
	For example, for a deep learning VM with a pre-installed a dcgm-exporter:3.2.5-3.1.8-ubuntu22.04 DCGM Exporter instance, provide the following script in base64 format
	`I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybW`
	`lzc2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1k`
	`IC0tZ3B1cyBhbGwgLS1jYXAtYWRkIFNZU19BRE1JTiAtLXJtIC1wIDk0MDA6OTQwMCBudmNyLmlvL252aW`
	`RpYS9rOHMvZGNnbS1leHBvcnRlcjozLjIuNS0zLjEuOC11YnVudHUyMi4wNA==`
	which corresponds to the following script in plain-text format:
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`  permissions: '0755'`
	`  content: |`
	`    #!/bin/bash`
	`    docker run -d --gpus all --cap-add SYS_ADMIN --rm -p 9400:9400 nvcr.io/nvidia/`
	`k8s/dcgm-exporter:3.2.5-3.1.8-ubuntu22.04`
	NOTE
	You can also add the instructions for running the DL workload whose GPU performance you want to measure with DCGM Exporter to the cloud-init script.
	– Image one-liner. Encode it in base64 format.
	`docker run -d --gpus all --cap-add SYS_ADMIN --rm -p 9400:9400 nvcr.io/nvidia/k8s/`
	`dcgm-exporter:`*`ngc_image_tag`*`-ubuntu22.04`
	For example, for dcgm-exporter:3.2.5-3.1.8-ubuntu22.04, provide the following script in base64 format:
	`ZG9ja2VyIHJ1biAtZCAtLWdwdXMgYWxsIC0tY2FwLWFkZCBTWVNfQURNSU4gLS1ybSAtcCA5NDAwOjk0MD`
	`AgbnZjci5pby9udmlkaWEvazhzL2RjZ20tZXhwb3J0ZXI6My4yLjUtMy4xLjgtdWJ1bnR1MjIuMDQ=`
	which corresponds to the following script in plain-text format:
	`docker run -d --gpus all --cap-add SYS_ADMIN --rm -p 9400:9400 nvcr.io/nvidia/k8s/`
	`dcgm-exporter:3.2.5-3.1.8-ubuntu22.04`
	• Enter the vGPU guest driver installation properties.
	• Provide values for the properties required for a disconnected environment as needed.
	See OVF Properties of Deep Learning VMs.

Output	• Installation logs for the vGPU guest driver in `/var/log/vgpu-install.log`. To verify that the vGPU guest driver is installed, log in to the VM over SSH and run the `nvidia-smi` command. • Cloud-init script logs in `/var/log/dl.log`. • DCGM Exporter that you can access at `http://`*`dl_vm_ip`*`:9400`.

Run a DL Workload on the Deep Leaning VM

Run the DL workload you want to collect vGPU metrics for and export the data to another application for further monitoring and visualization.

1. Log in to the deep learning VM as**vmware** over SSH.
2. Add the**vmware** user account to the**docker** group by running the following command.
   ```
   sudo usermod -aG docker ${USER}
   ```
3. Run the container for the DL workload, pulling it from the NVIDIA NGC catalog or from a local container registry.
   For example, to run the following command to run the tensorflow:23.10-tf2-py3 image from NVIDIA NGC:
   ```
   docker run -d -p 8888:8888 nvcr.io/nvidia/tensorflow:23.10-tf2-py3 /usr/local/bin/jupyter lab --allow-root
    --ip=* --port=8888 --no-browser --NotebookApp.token='' --NotebookApp.allow_origin='*' --notebook-dir=/
   workspace
   ```
4. Start using the DL workload for AI development.

Install Prometheus and Grafana

You can visualize and monitor the vGPU metrics from the DCGM Exporter virtual machine on a virtual machine running Prometheus and Grafana.

1. Create a visualization VM with Docker Community Engine installed.
2. Connect to the VM over SSH and create a YAML file for Prometheus.
   ```
   $ cat > prometheus.yml << EOF
   global:
    scrape_interval: 15s
    external_labels:
      monitor: 'codelab-monitor'
   scrape_configs:
    - job_name: 'dcgm'
      scrape_interval: 5s
      metrics_path: /metrics
      static_configs:
        - targets: [dl_vm_with_dcgm_exporter_ip:9400']
   EOF
   ```
3. Create a data path.
   ```
   $ mkdir grafana_data prometheus_data && chmod 777 grafana_data prometheus_data
   ```
4. Create a Docker compose file to install Prometheus and Grafana.
   ```
   $ cat > compose.yaml << EOF
   services:
    prometheus:
      image: prom/prometheus:v2.47.2
      container_name: "prometheus0"
      restart: always
      ports:
        - "9090:9090"
      volumes:
        - "./prometheus.yml:/etc/prometheus/prometheus.yml"
        - "./prometheus_data:/prometheus"
    grafana:
      image: grafana/grafana:10.2.0-ubuntu
      container_name: "grafana0"
   ```

```
        ports:
          - "3000:3000"
        restart: always
        volumes:
          - "./grafana_data:/var/lib/grafana"
    EOF
```

5. Start the Prometheus and Grafana containers.

```
$ sudo docker compose up -d
```

View vGPU Metrics in Prometheus

You can access Prometheus at `http://`*`visualization-vm-ip`*`:9090`. You can view the following vGPU information in the Prometheus UI:

Information	UI Section
Raw vGPU metrics from the deep learning VM	**Status** > **Target** To view the raw vGPU metrics from the deep learning VM, click the endpoint entry.
Graph expressions	1. On the main navigation bar, click the **Graph** tab. 2. Enter an expression and click **Execute**

For more information on using Prometheus, see the Prometheus documentation.

Visualize Metrics in Grafana

Set Prometheus as a data source for Grafana and visualize the vGPU metrics from the deep learning VM in a dashboard.

1. Access Grafana at `http://`*`visualization-vm-ip`*`:3000` by using the default user name**admin** and password `admin` .
2. Add Prometheus as the first data source, connecting to *`visualization-vm-ip`* on port 9090.
3. Create a dashboard with the vGPU metrics.

For more information on configuring a dashboard using a Prometheus data source, see the Grafana documentation.

Triton Inference Server

You can use a deep learning VM with a Triton Inference Server for loading a model repository and receive inference requests.

See the Triton Inference Server page.

Table 8: Triton Inference Server Container Image

Component	Description

Container image	`nvcr.io/nvidia/tritonserver:`*`ngc_image_tag`*
	For example:
	`nvcr.io/nvidia/tritonserver:23.10-py3`
	For information on the Triton Inference Server container images that are supported for deep learning VMs, see VMware Deep Learning VM Release Notes.

Required inputs	To deploy a Triton Inference Server workload, you must set the OVF properties for the deep learning virtual machine in the following way:
	• Use one of the following properties that are specific for the Triton Inference Server image.
	– Cloud-init script. Encode it in base64 format.
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`permissions: '0755'`
	`content: |`
	` #!/bin/bash`
	` docker run -d --gpus all --rm -p8000:8000 -p8001:8001 -p8002:8002 -v /home/`
	`vmware/model_repository:/models nvcr.io/nvidia/tritonserver:`*`ngc_image_tag`*`triton-`
	`server --model-repository=/models --model-control-mode=poll`
	
	For example, for tritonserver:23.10-py3, provide the following script in base64 format:
	`I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybW`
	`lzc2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBkb2NrZXIgcnVuIC1k`
	`IC0tZ3B1cyBhbGwgLS1ybSAtcDgwMDA6ODAwMCAtcDgwMDE6ODAwMSAtcDgwMDI6ODAwMiAtdiAvaG9tZS`
	`92bXdhcmUvbW9kZWxfcmVwb3NpdG9yeTovbW9kZWxzIG52Y3IuaW8vbnZpZGlhL3RyaXRvbnNlcnZlcjpu`
	`Z2NfaW1hZ2VfdGFnIHRyaXRvbnNlcnZlciAtLW1vZGVsLXJlcG9zaXRvcnk9L21vZGVscyAtLW1vZGVsLW`
	`NvbnRyb2wtbW9kZT1wb2xs`
	which corresponds to the following script in plain-text format:
	`#cloud-config`
	`write_files:`
	`- path: /opt/dlvm/dl_app.sh`
	`permissions: '0755'`
	`content: |`
	` #!/bin/bash`
	` docker run -d --gpus all --rm -p8000:8000 -p8001:8001 -p8002:8002 -v /home/`
	`vmware/model_repository:/models nvcr.io/nvidia/tritonserver:23.10-py3 tritonserver`
	` --model-repository=/models --model-control-mode=poll`
	
	– Image one-liner encoded in base64 format
	`docker run -d --gpus all --rm -p8000:8000 -p8001:8001 -p8002:8002 -v /home/vmware/`
	`model_repository:/models nvcr.io/nvidia/tritonserver:`*`ngc_image_tag`*`tritonserver --`
	`model-repository=/models --model-control-mode=poll`
	For example, for tritonserver:23.10-py3, provide the following script in base 64 format:
	`ZG9ja2VyIHJ1biAtZCAtLWdwdXMgYWxsIC0tcm0gLXA4MDAwOjgwMDAgLXA4MDAxOjgwMDEgLXA4MDAyOj`
	`gwMDIgLXYgL2hvbWUvdm13YXJlL21vZGVsX3JlcG9zaXRvcnk6L21vZGVscyBudmNyLmlvL252aWRpYS90`
	`cml0b25zZXJ2ZXI6MjMuMTAtcHkzIHRyaXRvbnNlcnZlciAtLW1vZGVsLXJlcG9zaXRvcnk9L21vZGVscy`
	`AtLW1vZGVsLWNvbnRyb2wtbW9kZT1wb2xs`
	which corresponds to the following script in plain-text format:
	`docker run -d --gpus all --rm -p8000:8000 -p8001:8001 -p8002:8002 -v /home/vmware/`
	`model_repository:/models nvcr.io/nvidia/tritonserver:23.10-py3 tritonserver --mod-`
	`el-repository=/models --model-control-mode=poll`
	• Enter the vGPU guest driver installation properties.
	• Provide values for the properties required for a disconnected environment as needed.
	See OVF Properties of Deep Learning VMs.

Output	• Installation logs for the vGPU guest driver in `/var/log/vgpu-install.log`.

Create a Model Repository

To load your model for model inference, perform these steps:

1. Create the model repository for your model.
   See the NVIDIA Triton Inference Server Model Repository documentation .
2. Copy the model repository to `/home/vmware/model_repository` so that the Triton Inference Server can load it.
   ```
   sudo cp -rpath_to_your_created_model_repository/* /home/vmware/model_repository/
   ```

Send Model Inference Requests

1. Verify that the Triton Inference Server is healthy and models are ready by running this command in the deep learning VM console.
   ```
   curl -v localhost:8000/v2/simple_sequence
   ```
2. Send a request to the model by running this command on the deep learning VM.
   ```
   curl -v localhost:8000/v2/models/simple_sequence
   ```

For more information on using the Triton Inference Server, see NVIDIA Triton Inference Server Model Repository documentation.

NVIDIA RAG

You can use a deep learning VM to build Retrieval Augmented Generation (RAG) solutions with an Llama2 model.

See the AI Chatbot with Retrieval Augmented Generation documentation.

Table 9: NVIDIA RAG Container Image

Component	Description
Container images and models	`rag-app-text-chatbot.yaml` in the NVIDIA sample RAG pipeline. For information on the NVIDIA RAG container applications that are supported for deep learning VMs, see VMware Deep Learning VM Image Release Notes.
Required inputs	To deploy an NVIDIA RAG workload, you must set the OVF properties for the deep learning virtual machine in the following way: • Enter a cloud-init script. Encode it in base64 format. For example, for version 24.03 of NVIDIA RAG, provide the following script:

```
I2Nsb3VkLWNvbmZpZwp3cml0ZV9maWxlczoKLSBwYXRoOiAvb3B0L2Rsdm0vZGxfYXBwLnNoCiAgcGVybWlz
c2lvbnM6ICcwNzU1JwogIGNvbnRlbnQ6IHwKICAgICMhL2Jpbi9iYXNoCiAgICBlcnJvcl9leGl0KCkgewog
ICAgICBlY2hvICJFcnJvcjogJDEiID4mMgogICAgICBleGl0IDEKICAgIH0KCiAgICBjYXQgPDxFT0YgPiAv
b3B0L2Rsdm0vY29uZmlnLmpzb24KICAgIHsKICAgICAgIl9jb21tZW50IjogIlRoaXMgcHJvdmlkZXMgZGVm
YXVsdCBzdXBwb3J0IGZvciBSQUc6IFRlbnNvclJUIGluZmVyZW5jZSwgbGxhbWEyLTEzYiBtb2RlbCwgYW5k
IEgxMDB4MiBHUFUiLAogICAgICAicmFnIjogewogICAgICAgICJvcmdfbmFtZSI6ICJjb2Nmd2dhOGpxMmMi
LAogICAgICAgICJvcmdfdGVhbV9uYW1lIjogIm5vLXRlYW0iLAogICAgICAgICJyYWdfcmVwb19uYW1lIjog
Im52aWRpYS9wYWlmIiwKICAgICAgICAibGxtX3JlcG9fbmFtZSI6ICJudmlkaWEvbmltIiwKICAgICAgICAi
ZW1iZWRfcmVwb19uYW1lIjogIm52aWRpYS9uZW1vLXJldHJpZXZlciIsCiAgICAgICAgInJhZ19uYW1lIjog
InJhZy1kb2NrZXItY29tcG9zZSIsCiAgICAgICAgInJhZ192ZXJzaW9uIjogIjI0LjAzIiwKICAgICAgICAi
ZW1iZWRfbmFtZSI6ICJudi1lbWJlZC1xYSIsCiAgICAgICAgImVtYmVkX3R5cGUiOiAiTlYtRW1iZWQtUUEi
LAogICAgICAgICJlbWJlZF92ZXJzaW9uIjogIjQiLAogICAgICAgICJpbmZlcmVuY2VfdHlwZSI6ICJ0cnQi
LAogICAgICAgICJsbG1fbmFtZSI6ICJsbGFtYTItMTNiLWNoYXQiLAogICAgICAgICJsbG1fdmVyc2lvbiI6
ICJoMTAweDJfZnAxNl8yNC4wMiIsCiAgICAgICAgIm51bV9ncHUiOiAiMiIsCiAgICAgICAgImhmX3Rva2Vu
IjogImh1Z2dpbmdmYWNlIHRva2VuIHRvIHB1bGwgbGxtIG1vZGVsLCB1cGRhdGUgd2hlbiB1c2luZyB2bGxt
IGluZmVyZW5jZSIsCiAgICAgICAgImhmX3JlcG8iOiAiaHVnZ2luZ2ZhY2UgbGxtIG1vZGVsIHJlcG9zaXRvc
nksIHVwZGF0ZSB3aGVuIHVzaW5nIHZsbG0gaW5mZXJlbmNlIgogICAgICB9CiAgICB9CiAgICBFT0YKICAgIE
NPTkZJR19KU09OPSQoY2F0ICIvb3B0L2Rsdm0vY29uZmlnLmpzb24iKQogICAgSU5GRVJFTkNFX1RZUEU9JCh
lY2hvICIke0NPTkZJR19KU09OfSIgfCBqcSAtciAnLnJhZy5pbmZlcmVuY2VfdHlwZScpCiAgICBpZiBbICIk
e0lORkVSRU5DRV9UWVBFfSIgPSAidHJ0IiBdOyB0aGVuCiAgICAgIHJlcXVpcmVkX3ZhcnM9KCJPUkdfTkFNR
SIgIk9SR19URUFNX05BTUUiICJSQUdfUkVQT19OQU1FIiAiTExNX1JFUE9fTkFNRSIgIkVNQkVEX1JFUE9fTk
FNRSIgIlJBR19OQU1FIiAiUkFHX1ZFUlNJT04iICJFTUJFRF9OQU1FIiAiRU1CRURfVFlQRSIgIkVNQkVEX1Z
FUlNJT04iICJMTE1fTkFNRSIgIkxMTV9WRVJTSU9OIiAiTlVNX0dQVSIpCiAgICBlbGlmIFsgIiR7SU5GRVJF
TkNFX1RZUEV9IiA9ICJ2bGxtIiBdOyB0aGVuCiAgICAgIHJlcXVpcmVkX3ZhcnM9KCJPUkdfTkFNRSIgIk9SR
19URUFNX05BTUUiICJSQUdfUkVQT19OQU1FIiAiTExNX1JFUE9fTkFNRSIgIkVNQkVEX1JFUE9fTkFNRSIgIl
JBR19OQU1FIiAiUkFHX1ZFUlNJT04iICJFTUJFRF9OQU1FIiAiRU1CRURfVFlQRSIgIkVNQkVEX1ZFUlNJT04
iICJMTE1fTkFNRSIgIk5VTV9HUFUiICJIRl9UT0tFTiIgIkhGX1JFUE8iKQogICAgZWxzZQogICAgICBlcnJv
cl9leGl0ICJJbmZlcmVuY2UgdHlwZSAnJHtJTkZFUkVOQ0VfVFlQRX0nIGlzIG5vdCByZWNvZ25pemVkLiBOb
yBhY3Rpb24gd2lsbCBiZSB0YWtlbi4iCiAgICBmaQogICAgZm9yIGluZGV4IGluICIkeyFyZXF1aXJlZF92YX
JzW0BdfSI7IGRvCiAgICAgIGtleT0iJHtyZXF1aXJlZF92YXJzWyRpbmRleF19IgogICAgICBqcV9xdWVyeT0
iLnJhZy4ke2tleSwsfSB8IHNlbGVjdCAoLiE9bnVsbCkiCiAgICAgIHZhbHVlPSQoZWNobyAiJHtDT05GSUdf
SlNPTn0iIHwganEgLXIgIiR7anFfcXVlcnl9IikKICAgICAgaWYgW1sgLXogIiR7dmFsdWV9IiBdXTsgdGhlb
iAKICAgICAgICBlcnJvcl9leGl0ICIke2tleX0gaXMgcmVxdWlyZWQgYnV0IG5vdCBzZXQuIgogICAgICBlbH
NlCiAgICAgICAgZXZhbCAke2tleX09XCIiJHt2YWx1ZX0iXCIKICAgICAgZmkKICAgIGRvbmUKCiAgICBSQUd
fVVJJPSIke1JBR19SRVBPX05BTUV9LyR7UkFHX05BTUV9OiR7UkFHX1ZFUlNJT059IgogICAgTExNX01PREVM
X1VSST0iJHtMTE1fUkVQT19OQU1FfS8ke0xMTV9OQU1FfToke0xMTV9WRVJTSU9OfSIKICAgIEVNQkVEX01PR
EVMX1VSST0iJHtFTUJFRF9SRVBPX05BTUV9LyR7RU1CRURfTkFNRX06JHtFTUJFRF9WRVJTSU9OfSIKCiAgIC
BOR0NfQ0xJX1ZFUlNJT049IjMuNDEuMiIKICAgIE5HQ19DTElfVVJMPSJodHRwczovL2FwaS5uZ2MubnZpZGl
hLmNvbS92Mi9yZXNvdXJjZXMvbnZpZGlhL25nYy1hcHBzL25nY19jbGkvdmVyc2lvbnMvJHtOR0NfQ0xJX1ZF
UlNJT059L2ZpbGVzL25nY2NsaV9saW51eC56aXAiCgogICAgbWtkaXIgLXAgL29wdC9kYXRhCiAgICBjZCAvb
3B0L2RhdGEKCiAgICBpZiBbICEgLWYgLmZpbGVfZG93bmxvYWRlZCBdOyB0aGVuCiAgICAgICMgY2xlYW4gdX
AKICAgICAgcm0gLXJmIGNvbXBvc2UuZW52ICR7UkFHX05BTUV9KiAke0xMTV9OQU1FfSogbmdjKiAke0VNQkV
EX05BTUV9KiAgLmpzb24gLmZpbGVfZG93bmxvYWRlZAoKICAgICAgIyBpbnN0YWxsIG5nYy1jbGkKICAgICAg
d2dldCAtLWNvbnRlbnQtZGlzcG9zaXRpb24gJHtOR0NfQ0xJX1VSTH0gLU8gbmdjY2xpX2xpbnV4LnppcCAmJ
iB1bnppcCBuZ2NjbGlfbGludXguemlwCiAgICAgIGV4cG9ydCBQQVRIPWBwd2RgL25nYy1jbGk6JHtQQVRIfQ
oKICAgICAgQVBJS0VZPSIiCiAgICAgIFJFR19VUkk9Im52Y3IuaW8iCgogICAgICBpZiBbWyAiJChncmVwIHJ
lZ2lzdHJ5LXVyaSAvb3B0L2Rsdm0vb3ZmLWVudi54bWwgfCBzZWQgLW4gJ3MvLipvZTp2YWx1ZT0iXChbXiJd
KlwpLiovXDEvcCcpIiA9PSAgIiR7UkVHX1VSSX0iKiBdXTsgdGhlbgogICAgICAgIEFQSUtFWT0kKGdyZXAgc
```

Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA

Deep learning VM images in VMware Private AI Foundation with NVIDIA are distributed in a shared content library published by VMware. As a cloud administrator, you use a content library to pull specific VM images in your VI workload domain during VM deployment.

Prerequisites

As a cloud administrator, verify that VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA.

Procedure

As a cloud administrator, create a content library with deep learning VM images. If you plan to depoy deep learning VMs by using a YAML file and the `kubectl` command, you add the content library to the namespace you created for AI workloads.

1. Log in to the vCenter Server instance for the VI workload domain at `https://<vcenter_server_fqdn>/ui`.
2. Select **Menu** > **Content Libraries** and click **Create**.
3. Create a content library for the deep learning VM images.
   - For a connected environment, create a subscribed content library that is connected to `https://packages.vmware.com/dl-vm/lib.json`. Authentication is not required.
   - For a disconnected environment, download the deep learning VM images from `https://packages.vmware.com/dl-vm/` and import them in to a local content library.

     For each image, download the relevant `.ovf`, `.vmdk`, `.mf`, and `.cert` files.

   See Create and Edit a Content Library and How to Populate Libraries with Content.
4. If you plan to enable deployments of deep learning VMs on a Supervisor by directly calling `kubectl`, add the content library to the vSphere namespace for AI workloads.

   VMware Aria Automation creates a namespace every time a deep learning VM is provisioned, automatically adding the content library to it.

   a. Select **Menu** > **Workload Management**.
   b. Navigate to the namespace for AI workloads.
   c. On the **VM Service** card, click **Manage content libraries**.
   d. Select the content library with the deep learning VM images and click **OK**.

Deploy a Deep Learning VM Directly on a vSphere Cluster in VMware Private AI Foundation with NVIDIA

To quickly test the deep learning VM templates in VMware Private AI Foundation with NVIDIA, you can deploy a deep learning VM directly on a vSphere cluster by using the vSphere Client.

Verify that the following prerequisites are in place for the AI-ready infrastructure.

- VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA.
- A content library with deep learning VMs is available. See Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA.

1. Log in to the vCenter Server instance for the VI workload domain.
2. From the vSphere Client home menu, select **Content Libraries**.
3. Deploy a deep learning VM from the content library.
   a) Navigate to the deep learning VM image in the content library.
   b) Right-click an OVF template and select **New VM from This Template**.
   c) Follow the wizard to enter a name and select a VM folder, and select a GPU-enabled cluster in the VI workload domain.
   d) Select a datastore and a network on the distributed switch for the cluster.
   e) On the **Customize template** page, enter the custom VM properties that are required for setting up the AI functionality.

      See OVF Properties of Deep Learning VMs.
   f) Click **Finish**
4. After the deep learning VM is created, in the virtual machine settings, assign it an NVIDIA vGPU device.

   See Add an NVIDIA GRID vGPU to a Virtual Machine.

   For a deep learning VM that is running an NVIDIA RAG, select the full-sized vGPU profile for time-slicing mode or a MIG profile. For example, for NVIDIA A100 40GB in vGPU time-slicing mode, select **nvidia_a100-40c**.
5. For a deep learning VM that is running an NVIDIA RAG, in the **Advanced Parameters** tab of the virtual machine settings, set the `pciPassthru<vgpu-id>.cfg.enable_uvm` parameter to `1`.

   where `<vgpu-id>` identifies the vGPU assigned to the virtual machine. For example, if two vGPUs are assigned to the virtual machine, you set `pciPassthru0.cfg.parameter=1` and `pciPassthru1.cfg.parameter = 1 .`
6. Power on the deep learning VM.

The vGPU guest driver and the specified deep learning workload is installed the first time you start the deep learning VM.

You can examine the logs or open the JupyterLab instance that comes with some of the images. You can share access details with data scientists in your organization. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA.

- Connect to the deep learning VM over SSH and verify that all components are installed and running as expected.
- Send access details to your data scientists.

Deploy a Deep Learning VM by Using the `kubectl` Command in VMware Private AI Foundation with NVIDIA

The VM service in the Supervisor in vSphere with Tanzu enables data scientists and DevOps engineers to deploy and run deep learning VMs by using the Kubernetes API.

As a data scientist or DevOps engineer, you use `kubectl` to deploy a deep learning VM on the namespace configured by the cloud administrator.

Prerequisites

Verify with the cloud administrator that the following prerequisites are in place for the AI-ready infrastructure.

- VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA.
- A content library with deep learning VMs is added to the namespace for AI workloads. See Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA.

Procedure

1. Log in to the Supervisor control plane.

   ```
   kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN--vsphere-usernameUSERNAME
   ```
2. Examine that all required VM resources, such as VM classes and VM images, are in place on the namespace.
   See View VM Resources Available on a Namespace in vSphere with Tanzu.
3. Prepare the YAML file for the deep learning VM.
   Use the vm-operator-api, setting the OVF properties as a ConfigMap object.
   For example, you can create a YAML specification `example-dl-vm.yaml` for an example deep learning VM running PyTorch.

   ```
   apiVersion: vmoperator.vmware.com/v1alpha1
   kind: VirtualMachine
   metadata:
   name: example-dl-vm
   namespace: vpaif-ns
   labels:
     app: example-dl-app
   spec:
   className: gpu-a30
   imageName: vmi-xxxxxxxxxxxxx
   powerState: poweredOn
   storageClass: tanzu-storage-policy
   vmMetadata:
     configMapName: example-dl-vm-config
     transport: OvfEnv
   ```

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
   name: example-dl-vm-config
   namespace: vpaif-ns
   data:
   user-data: I2Nsb3VkLWNvbmZpZwogICAgd3JpdGVfZmlsZXM6CiAgICAtIHBhdG-
   g6IC9vcHQvZGx2bS9kbF9hcHAuc2gKICAgICAgcGVybWlzc2lvbnM6ICcwNzU1JwogICAgICBjb250ZW50OiB8CiAgICAgICAgIyEvYm-
   luL2Jhc2gKICAgICAgICBkb2NrZXIgcnVuIC1kIC1wIDg4ODg6ODg4OCBudmNyLmlvL252aWRpYS9weXRvcmNoOjIzLjEw-
   LXB5MyAvdXNyL2xvY2FsL2Jpbi9qdXB5dGVyIGxhYiAtLWFsbG93LXJvb3QgLS1pcD0qIC0tcG9ydD04ODg4IC0tbm8tYnJvd3Nlci-
   AtLU5vdGVib29rQXBwLnRva2VuPScnIC0tTm90ZWJvb2tBcHAuYWxsb3dfb3JpZ2luPScqJyAtLW5vdGVib29rLWRpcj0vd29ya3NwYWNl
   vgpu-license:NVIDIA-client-configuration-token
   nvidia-portal-api-key:API-key-from-NVIDIA-licensing-portal
   password:password-for-vmware-user
   ```

   NOTE
   `user-data` is the base64 encoded value for the following cloud-init code:

   ```
   #cloud-config
     write_files:
     - path: /opt/dlvm/dl_app.sh
   ```

```
                permissions: '0755'
                content: |
                  #!/bin/bash
                  docker run -d -p 8888:8888 nvcr.io/nvidia/pytorch:23.10-py3 /usr/local/bin/jupyter lab --
            allow-root --ip=* --port=8888 --no-browser --NotebookApp.token='' --NotebookApp.allow_origin='*'
             --notebook-dir=/workspace
    apiVersion: vmoperator.vmware.com/v1alpha1
    kind: VirtualMachineService
    metadata:
    name: example-dl-vm
    namespace: vpaif-ns
    spec:
    ports:
    - name: ssh
      port: 22
      protocol: TCP
      targetPort: 22
    - name: junyperlab
      port: 8888
      protocol: TCP
      targetPort: 8888
    selector:
      app: example-dl-app
      type: LoadBalancer
```

4. Switch to the context of the vSphere namespace created by the cloud administrator.
   For example, for a namespace called `example-dl-vm-namespace`.
   ```
   kubectl config use-context example-dl-vm-namespace
   ```
5. Deploy the deep learning VM.
   ```
   kubectl apply -f example-dl-vm.yaml
   ```
6. Verify that the VM has been created by running these commands.
   ```
   kubectl get vm -n example-dl-vm.yaml
   kubectl describe virtualmachine example-dl-vm
   pingIP_address_returned_by_kubectl_describe
   ```
7. Ping the IP address of the virtual machine assigned by the requested networking service.
   To get the public address and the ports for access to the deep learning VM, get the details about the load balancer service that has been created.
   ```
   kubectl get services
   NAME    TYPE           CLUSTER-IP             EXTERNAL-IP          PORT(S)                         AGE
   example-dl-vm   LoadBalancer  <internal-ip-address><public-IPaddress> 22:30473/TCP,8888:32180/TCP   9m40s
   ```

The vGPU guest driver and the specified DL workload is installed the first time you start the deep learning VM.

You can examine the logs or open the JupyterLab notebook that comes with some of the images. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA.

Deploy a Deep Learning VM by Using a Self-Service Catalog in VMware Private AI Foundation with NVIDIA

Data scientists, DevOps engineers and developers can use VMware Aria Automation to provision deep learning virtual machines on the Supervisor instance in a VI workload domain.

The workflow for deploying a deep learning VM has two parts:

- As a cloud administrator, you add self-service catalog items for private AI to Automation Service Broker.
- As a data scientist or DevOps engineer, you use an AI workstation catalog item to deploy a deep learning VM on a new namespace on the Supervisor.

Create AI Self-Service Catalog Items in VMware Aria Automation

As a cloud administrator, you can use the catalog setup wizard for private AI in VMware Aria Automation to quickly add catalog items for deploying deep learning virtual machines or GPU-accelerated TKG clusters in a VI workload domain.

- Verify that VMware Private AI Foundation with NVIDIA is available for the VI workload domain.
- Verify that the prerequisites for deploying deep learning VMs are in place.
- Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA.

Data scientists can use deep learning catalog items for deployment of deep learning VMs. DevOps engineers can use the catalog items for provisioning AI-ready TKG clusters. Every time you run it, the catalog setup wizard for private AI adds two catalog items to the Service Broker catalog - one for a deep learning virtual machine and one for a TKG cluster.

Every time you run it, the catalog setup wizard for private AI adds two catalog items to the Service Broker catalog - one for a deep learning virtual machine and one for a TKG cluster. You can run the wizard every time you need the following:

- Enable provisioning of AI workloads on another supervisor.
- Accommodate a change in your NVIDIA AI Enterprise license, including the client configuration `.tok` file and license server, or the download URL for the vGPU guest drivers for a disconnected environment.
- Accommodate a deep learning VM image change.
- Use other vGPU or non-GPU VM classes, storage policy, or container registry.
- Create catalog items in a new project.

1. Navigate to the VMware Aria Automation home page and click **Quickstart**.
2. Run the Private AI Automation Services catalog setup wizard for Private AI Automation.

   See Add Private AI items to the Automation Service Broker catalog in the *VMware Aria Automation Product Documentation*.

Deploy a Deep Learning VM by Using a Self-Service Catalog in VMware Aria Automation

In VMware Private AI Foundation with NVIDIA, as a data scientist or DevOps engineer, you can deploy a deep learning VM from VMware Aria Automation by using an AI workstation self-service catalog items in Automation Service Broker.

> **NOTE**
> VMware Aria Automation creates a namespace every time you provision a deep learning VM.

In Automation Service Broker, deploy an AI workstation catalog item on the Supervisor instance in the VI workload domain.
See Deploy a deep learning virtual machine to a VI workload domain.

If you plan to use DCGM Exporter with a DL workload that uses GPU capacity, you can have the DL workload installed at virtual machine startup as a part of the cloud-init process or from the command line after the virtual machine is started. To include the DL workload in the cloud-init process, in the request form of the AI workstation catalog item, add the following configuration in addition to the other details for provisioning the deep learning VM:

1. From the **Software Bundle** drop-down menu, select **DCGM Exporter**.
2. Select the **Custom cloud-init** check box and enter the instructions for running the DL workload.

The vGPU guest driver and the specified deep learning workload is installed the first time you start the deep learning VM.

You can examine the logs or open the JupyterLab instance that comes with some of the images. See Deep Learning Workloads in VMware Private AI Foundation with NVIDIA.

For details on how to access the virtual machine and the JupyterLab instance on it, in Automation Service Broker, navigate to **Consume** > **Deployments** > **Deployments**.

Deploying AI Workloads on TKG Clusters in VMware Private AI Foundation with NVIDIA

You can deploy container AI workloads on Tanzu Kubernetes Grid (TKG) clusters whose worker nodes are accelerated with NVIDIA GPUs.

For information about the support of AI workloads on TKG clusters, see About Deploying AI/ML Workloads on TKGS Clusters.

Configure a Content Library with Ubuntu TKr for a Disconnected VMware Private AI Foundation with NVIDIA Environment

As a cloud administrator, if your environment has no Internet connectivity, you provide a local content library where you manually upload Tanzu Kubernetes releases (TKr) and associate it with the Supervisor.

Deploying NVIDIA-aware AI workloads on TKG clusters requires the use of the Ubuntu edition of Tanzu Kubernetes releases.

CAUTION
The TKr content library is used across all vSphere namespaces in the Supervisor when you provision new TKG clusters.

Prerequisites

As a cloud administrator, verify that VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA

Procedure

1. Download the Ubuntu-based TKr images with the required Kubernetes versions from `https://wp-content.vmware.com/v2/latest/`.
2. Log in to the vCenter Server instance for the VI workload domain at `http://<vcenter_server_fqdn>/ui`.
3. Select **Menu** > **Content Libraries** and click **Create**.
4. Create a local content library and import the TKr images there.
   See Create a Local Content Library (for Air-Gapped Cluster Provisioning).

5. Add the content library to the Supervisor.
   a. Select **Menu** > **Workload Management**.
   b. Navigate to the Supervisor for AI workloads.
   c. On the **Configure** tab, select **General**.
   d. Next to the **Tanzu Kubernetes Grid Service** property, click **Edit**.
   e. On the **General** page that appears, expand **Tanzu Kubernetes Grid Service**, and next to **Content Library**, click **Edit**.
   f. Select the content library with the TKr images and click **OK**.

Provision a GPU-Accelerated TKG Cluster by Using the `kubectl` Command in a Connected VMware Private AI Foundation with NVIDIA Environment

In VMware Private AI Foundation with NVIDIA, as a DevOps engineer, by using the Kubernetes API, you provision a TKG cluster that uses NVIDIA GPUs. Then, you can deploy container AI workloads from the NVIDIA NGC catalog.

Verify with the cloud administrator that the following prerequisites are in place for the AI-ready infrastructure.

- VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA.

You use `kubectl` to deploy the TKG cluster on the namespace configured by the cloud administrator.

1. Log in to the Supervisor control plane.

   ```
   kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN --vsphere-username USERNAME
   ```

2. Provision a TKG cluster and install the NVIDIA GPU Operator and NVIDIA Network Operator on it.

   See Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters.

Deploy an AI container image from the NVIDIA NGC catalog.

Provision a GPU-Accelerated TKG Cluster by Using the `kubectl` Command in a Disconnected VMware Private AI Foundation with NVIDIA Environment

In VMware Private AI Foundation with NVIDIA, as a DevOps engineer, by using the Kubernetes API, you provision a TKG cluster that uses NVIDIA GPUs. In a disconnected environment, you must additionally set up a local Ubuntu package repository and use the Harbor Registry for the Supervisor.

Prerequisites

Verify with the cloud administrator that the following prerequisites are in place for the AI-ready infrastructure.

- VMware Private AI Foundation with NVIDIA is deployed and configured. See Deploying VMware Private AI Foundation with NVIDIA.
- A content library with Ubuntu TKr images is added to the namespace for AI workloads. See Configure a Content Library with Ubuntu TKr for a Disconnected VMware Private AI Foundation with NVIDIA Environment.
- A machine that has access to the Supervisor endpoint.

Procedure

1. Provision a TKG cluster on the vSphere namespace configured by the cloud administrator.
   See Provision a TKGS Cluster for NVIDIA vGPU.

2. Complete the TKG cluster setup.

   See Installing VMware vSphere with VMware Tanzu (Air-gapped).

   a. Provide a local Ubuntu package repository and upload the container images in the NVIDIA GPU Operator package to the Harbor Registry for the Supervisor.
   b. Update the Helm chart definitions of the NVIDIA GPU Operator to use the local Ubuntu package repository and private Harbor Registry.
   c. Provide NVIDIA license information.
   d. Install the NVIDIA GPU Operator.

What to do next

Deploy an AI container image from the Harbor Registry for the Supervisor.

Provision a GPU-Accelerated TKG Cluster by Using a Self-Service Catalog in VMware Private AI Foundation with NVIDIA

DevOps engineers and developers can use VMware Aria Automation to provision GPU-accelerated TKG clusters for hosting container AI workloads on the Supervisor instance in a VI workload domain.

The workflow for deploying a GPU-accelerated TKG cluster has two parts:

- As a cloud administrator, you add self-service catalog items for private AI for a new namespace on the Supervisor to Automation Service Broker.
- As a data scientist or DevOps engineer, you use an AI Kubernetes cluster catalog item to deploy a GPU-accelerated TKG cluster on a new namespace on the Supervisor.

Create AI Self-Service Catalog Items in VMware Aria Automation

As a cloud administrator, you can use the catalog setup wizard for private AI in VMware Aria Automation to quickly add catalog items for deploying deep learning virtual machines or GPU-accelerated TKG clusters in a VI workload domain.

- Verify that VMware Private AI Foundation with NVIDIA is available for the VI workload domain.
- Verify that the prerequisites for deploying deep learning VMs are in place.
- Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA.

Data scientists can use deep learning catalog items for deployment of deep learning VMs. DevOps engineers can use the catalog items for provisioning AI-ready TKG clusters. Every time you run it, the catalog setup wizard for private AI adds two catalog items to the Service Broker catalog - one for a deep learning virtual machine and one for a TKG cluster.

Every time you run it, the catalog setup wizard for private AI adds two catalog items to the Service Broker catalog - one for a deep learning virtual machine and one for a TKG cluster. You can run the wizard every time you need the following:

- Enable provisioning of AI workloads on another supervisor.
- Accommodate a change in your NVIDIA AI Enterprise license, including the client configuration `.tok` file and license server, or the download URL for the vGPU guest drivers for a disconnected environment.
- Accommodate a deep learning VM image change.
- Use other vGPU or non-GPU VM classes, storage policy, or container registry.
- Create catalog items in a new project.

1. Navigate to the VMware Aria Automation home page and click **Quickstart**.
2. Run the Private AI Automation Services catalog setup wizard for Private AI Automation.

   See Add Private AI items to the Automation Service Broker catalog in the *VMware Aria Automation Product Documentation*.

Provision a GPU-Accelerated TKG Cluster by Using a Self-Service Catalog in VMware Aria Automation

In VMware Private AI Foundation with NVIDIA, as a DevOps engineer, you can provision a TKG cluster accelerated with NVIDIA GPUs from VMware Aria Automation by using an AI Kubernetes Cluster self-service catalog items in Automation Service Broker. Then, you can deploy AI container images from NVIDIA NGC on the cluster.

> **NOTE**
> VMware Aria Automation creates a namespace every time you provision a GPU-accelerated TKG cluster.

1. In a connected environment, in Automation Service Broker, deploy an AI Kubernetes Cluster catalog item on the Supervisor instance configured by the cloud administrator.

   See Deploy an AI-enabled Tanzu Kubernetes Grid cluster.

2. In a disconnected environment, upload the components of the NVIDIA GPU Operator on internal locations and modify the AI Kubernetes Cluster catalog item for the Supervisor instance configured by the cloud administrator.
   a) Provide a local Ubuntu package repository and upload the container images in the NVIDIA GPU Operator package to the Harbor Registry for the Supervisor.
   b) Provide a local Helm chart repository with NVIDIA GPU Operator chart definitions.
   c) Update the Helm chart definitions of the NVIDIA GPU Operator to use the local Ubuntu package repository and private Harbor Registry.
   d) On the **Design** > **Cloud Templates** page of Automation Assembler, modify directly the **AI Kubernetes Cluster** cloud template, or clone the cloud template and modify the clone.
      1. Add a ConfigMap to for using the local Ubuntu repository in the NVIDIA GPU Operator.
      2. Update the Helm chart repository URL.
      3. Deploy the cloud template.
   e) Deploy the modified or cloned **AI Kubernetes Cluster** catalog item on the Supervisor instance.

1. For details on how to access the TKG cluster by using `kubectl`, in Automation Service Broker, navigate to **Consume** > **Deployments** > **Deployments**.
2. Deploy an AI container image from the NVIDIA NGC catalog.

   In a disconnected environment, you must upload the AI container images to a private container registry. See Setting Up a Private Harbor Registry in VMware Private AI Foundation with NVIDIA.

Deploying RAG Workloads in VMware Private AI Foundation with NVIDIA

A Retrieval-Augmented Generation (RAG) workload consists of an LLM and external knowledge base with latest data, stored in a vector database. In VMware Private AI Foundation with NVIDIA, you can configure a RAG workload to use embeddings from a vector database managed by VMware Data Services Manager.

Deploy a Vector Database in VMware Private AI Foundation with NVIDIA

If you plan to use Retrieval-Augmented Generation (RAG) with VMware Private AI Foundation with NVIDIA, set up a PostgreSQL database with pgvector by using VMware Data Services Manager.

- Verify that VMware Private AI Foundation with NVIDIA is available for the VI workload domain. See Deploying VMware Private AI Foundation with NVIDIA.
- Install the `psql` command line utility from the PostgreSQL Web site.

You can create the database manually or create a self-service catalog in VMware Aria Automation that can be used by DevOps engineers and developers.

1. Deploy a PostgreSQL database in the VI workload domain and get the connection string for the database.

Deployment Workflow	Description
Deploy and get the connection string of a PostgreSQL database from the VMware Data Services Manager Console.	See Creating Databases and Connecting to a Database.
Deploy and get the connection string of a PostgreSQL database by using the `kubectl` command	See Enabling Self-Service Consumption of VMware Data Services Manager.
Deploy and get the connection string of a PostgreSQL database from VMware Aria Automation	See Deploying a Vector Database by Using a Self-Service Catalog Item in VMware Aria Automation.

   The connection string of the deployed database has the following format.

```
postgres://pgvector_db_admin:encoded_pgvector_db_admin_password@pgvector_db_ip_address:5432/pgvector_d-
b_name
```

2. Activate the pgvector extension on the database by using the `psql` command line utility.
   a) Connect to the database.

   ```
   psql -hpgvector_db_ip_address-p 5432 -dpgvector_db_name-Upgvector_db_admin-W
   ```

   b) Activate the pgvector extension.

   ```
   pgvector_db_name=# CREATE EXTENSION vector;
   ```

Deploying a Vector Database by Using a Self-Service Catalog Item in VMware Aria Automation

Data scientists and DevOps engineers can use VMware Aria Automation to provision a PostgreSQL database with pgvector extension in the VI workload domain dedicated for your AI workloads.

Create a Vector Database Catalog Item in VMware Aria Automation

As a cloud administrator, add a catalog item for provisioning databases in VMware Data Services Manager to Service Broker in VMware Aria Automation.

- Verify that you have VMware Data Services Manager 2.0.2 deployed.
- Provide a machine that has Python 3.10 installed and has access to the VMware Data Services Manager and VMware Aria Automation instances.

1. On the machine running Python, download the `AriaAutomation_DataServicesManager` bundle for VMware Data Services Manager 2.0.2 from VMware Tanzu Network and extract its content.

2. Update the `config.json` file in the folder where you extracted the bundle with the URLs and user credentials for VMware Data Services Manager and VMware Aria Automation.

   Optionally, you can also set the name of the catalog item, Automaton Assembler project, and other parameters.

3. To create the catalog items in VMware Aria Automation, run the `aria.py` Python script in the following way.

   ```
   python3 aria.py enable-blueprint-version-2
   ```

The Python script creates items in VMware Aria Automation that are required for using VMware Data Services Manager for database provisioning. See the `readme.md` file in the `AriaAutomation_DataServicesManager` bundle

Deploy a Vector Database by Using a Self-Service Catalog Item in VMware Aria Automation

In VMware Private AI Foundation with NVIDIA, as data scientist or a DevOps engineer, you can deploy a vector database from VMware Aria Automation by using a self-service catalog item in Automation Service Broker.

1. Log in to VMware Aria Automation and, in Automation Service Broker, locate the catalog item for database deployment according to the information from your cloud administrator.

   By default, the catalog item is called **DSM DBaaS**.
2. In the catalog item card, click **Request** and enter the details for the new PostgreSQL database.

   For more information on the settings for the database, see Creating Databases.
3. Get the connection string of the deployed database.
   a) In Automation Service Broker, click **Deployments** > **Deployments** .
   b) Select the deployment entry for the database.
   c) On the **Topology** tab, select the cloud template for the database deployment and from the **Actions** menu for the template, select **Get Connection String**.

For more information on provisioning and performing operations on databases in VMware Data Services Manager from VMware Aria Automation, see the `readme.md` file in the `AriaAutomation_DataServicesManager` bundle.

Deploy a Deep Learning VM with a RAG Workload

You can deploy a deep learning VM with an NVIDIA RAG workload using a pgvector PostgreSQL database managed by VMware Data Services Manager.

- Verify that VMware Private AI Foundation with NVIDIA is available for the VI workload domain. See Deploying VMware Private AI Foundation with NVIDIA.
- Create a Content Library with Deep Learning VM Images for VMware Private AI Foundation with NVIDIA
- Deploy a Vector Database in VMware Private AI Foundation with NVIDIA.

1. If you are deploying the deep learning VM directly on the vSphere cluster or by using the kubectl command, create a cloud-init script and deploy the deep learning VM.
   a) Create a cloud-init script for NVIDIA RAG and the pgvector PostgreSQL database you have created.

      You can modify the initial version of the cloud-init script for NVIDIA RAG. For example, for NVIDIA RAG 24.03 and a pgvector PostgreSQL database with connection details `postgres://`*`pgvector_db_admin`*`:`*`encoded_pgvector_db_admin_password`*`@`*`pgvector_db_ip_address`*`:5432`
      .

      ```
      #cloud-config
      write_files:
      - path: /opt/dlvm/dl_app.sh
        permissions: '0755'
        content: |
          #!/bin/bash
          error_exit() {
            echo "Error: $1" >&2
            exit 1
          }
      ```

```
   cat <<EOF > /opt/dlvm/config.json
   {
     "_comment": "This provides default support for RAG: TensorRT inference, llama2-13b model, and
H100x2 GPU",
     "rag": {
       "org_name": "cocfwga8jq2c",
       "org_team_name": "no-team",
       "rag_repo_name": "nvidia/paif",
       "llm_repo_name": "nvidia/nim",
       "embed_repo_name": "nvidia/nemo-retriever",
       "rag_name": "rag-docker-compose",
       "rag_version": "24.03",
       "embed_name": "nv-embed-qa",
       "embed_type": "NV-Embed-QA",
       "embed_version": "4",
       "inference_type": "trt",
       "llm_name": "llama2-13b-chat",
       "llm_version": "h100x2_fp16_24.02",
       "num_gpu": "2",
       "hf_token": "huggingface token to pull llm model, update when using vllm inference",
       "hf_repo": "huggingface llm model repository, update when using vllm inference"
     }
   }
   EOF
   CONFIG_JSON=$(cat "/opt/dlvm/config.json")
   INFERENCE_TYPE=$(echo "${CONFIG_JSON}" | jq -r '.rag.inference_type')
   if [ "${INFERENCE_TYPE}" = "trt" ]; then
     required_vars=("ORG_NAME" "ORG_TEAM_NAME" "RAG_REPO_NAME" "LLM_REPO_NAME" "EMBED_REPO_NAME"
"RAG_NAME" "RAG_VERSION" "EMBED_NAME" "EMBED_TYPE" "EMBED_VERSION" "LLM_NAME" "LLM_VERSION" "NUM_GPU")
   elif [ "${INFERENCE_TYPE}" = "vllm" ]; then
     required_vars=("ORG_NAME" "ORG_TEAM_NAME" "RAG_REPO_NAME" "LLM_REPO_NAME" "EMBED_REPO_NAME"
"RAG_NAME" "RAG_VERSION" "EMBED_NAME" "EMBED_TYPE" "EMBED_VERSION" "LLM_NAME" "NUM_GPU" "HF_TOKEN"
"HF_REPO")
   else
     error_exit "Inference type '${INFERENCE_TYPE}' is not recognized. No action will be taken."
   fi
   for index in "${!required_vars[@]}"; do
     key="${required_vars[$index]}"
     jq_query=".rag.${key,,} | select (.!=null)"
     value=$(echo "${CONFIG_JSON}" | jq -r "${jq_query}")
     if [[ -z "${value}" ]]; then
       error_exit "${key} is required but not set."
     else
       eval ${key}=\""${value}"\"
     fi
   done

   RAG_URI="${RAG_REPO_NAME}/${RAG_NAME}:${RAG_VERSION}"
   LLM_MODEL_URI="${LLM_REPO_NAME}/${LLM_NAME}:${LLM_VERSION}"
   EMBED_MODEL_URI="${EMBED_REPO_NAME}/${EMBED_NAME}:${EMBED_VERSION}"

   NGC_CLI_VERSION="3.41.2"
```

```
    NGC_CLI_URL="https://api.ngc.nvidia.com/v2/resources/nvidia/ngc-apps/ngc_cli/ver-
sions/${NGC_CLI_VERSION}/files/ngccli_linux.zip"

    mkdir -p /opt/data
    cd /opt/data

    if [ ! -f .file_downloaded ]; then
      # clean up
      rm -rf compose.env ${RAG_NAME}* ${LLM_NAME}* ngc* ${EMBED_NAME}* *.json .file_downloaded

      # install ngc-cli
      wget --content-disposition ${NGC_CLI_URL} -O ngccli_linux.zip && unzip ngccli_linux.zip
      export PATH=`pwd`/ngc-cli:${PATH}

      APIKEY=""
      REG_URI="nvcr.io"

      if [[ "$(grep registry-uri /opt/dlvm/ovf-env.xml | sed -n 's/.*oe:value="\([^"]*\).*/\1/p')" ==
 *"${REG_URI}"* ]]; then
        APIKEY=$(grep registry-passwd /opt/dlvm/ovf-env.xml | sed -n 's/.*oe:value="\([^"]*\).*/\1/p')
      fi

      if [ -z "${APIKEY}" ]; then
          error_exit "No APIKEY found"
      fi

      # config ngc-cli
      mkdir -p ~/.ngc

      cat << EOF > ~/.ngc/config
      [CURRENT]
      apikey = ${APIKEY}
      format_type = ascii
      org = ${ORG_NAME}
      team = ${ORG_TEAM_NAME}
      ace = no-ace
    EOF

      # ngc docker login
      docker login nvcr.io -u \$oauthtoken -p ${APIKEY}

      # dockerhub login for general components, e.g. minio
      DOCKERHUB_URI=$(grep registry-2-uri /opt/dlvm/ovf-env.xml | sed -n 's/.*oe:value="\([^"]*\).*/\1/
p')
      DOCKERHUB_USERNAME=$(grep registry-2-user /opt/dlvm/ovf-env.xml | sed -n 's/.*oe:val-
ue="\([^"]*\).*/\1/p')
      DOCKERHUB_PASSWORD=$(grep registry-2-passwd /opt/dlvm/ovf-env.xml | sed -n 's/.*oe:val-
ue="\([^"]*\).*/\1/p')

      if [[ -n "${DOCKERHUB_USERNAME}" && -n "${DOCKERHUB_PASSWORD}" ]]; then
        docker login -u ${DOCKERHUB_USERNAME} -p ${DOCKERHUB_PASSWORD}
      else
        echo "Warning: DockerHub not login"
      fi

      # get RAG files
      ngc registry resource download-version ${RAG_URI}

      # get llm model
      if [ "${INFERENCE_TYPE}" = "trt" ]; then
        ngc registry model download-version ${LLM_MODEL_URI}
        chmod -R o+rX ${LLM_NAME}_v${LLM_VERSION}
        LLM_MODEL_FOLDER="/opt/data/${LLM_NAME}_v${LLM_VERSION}"
      elif [ "${INFERENCE_TYPE}" = "vllm" ]; then
        pip install huggingface_hub
        huggingface-cli login --token ${HF_TOKEN}
        huggingface-cli download --resume-download ${HF_REPO}/${LLM_NAME} --local-dir ${LLM_NAME} --lo-
cal-dir-use-symlinks False
        LLM_MODEL_FOLDER="/opt/data/${LLM_NAME}"
        cat << EOF > ${LLM_MODEL_FOLDER}/model_config.yaml
        engine:
          model: /model-store
          enforce_eager: false
          max_context_len_to_capture: 8192
          max_num_seqs: 256
          dtype: float16
          tensor_parallel_size: ${NUM_GPU}
          gpu_memory_utilization: 0.8
    EOF
        chmod -R o+rX ${LLM_MODEL_FOLDER}
        python3 -c "import yaml, json, sys; print(json.dumps(yaml.safe_load(sys.stdin.read())))" <
 "${RAG_NAME}_v${RAG_VERSION}/rag-app-text-chatbot.yaml"> rag-app-text-chatbot.json
        jq '.services."nemollm-inference".image = "nvcr.io/nvidia/nim/nim_llm:24.02-day0" |
            .services."nemollm-inference".command = "nim_vllm --model_name ${MODEL_NAME} --model_con-
fig /model-store/model_config.yaml" |
            .services."nemollm-inference".ports += ["8000:8000"] |
            .services."nemollm-inference".expose += ["8000"]' rag-app-text-chatbot.json > temp.json &&
 mv temp.json rag-app-text-chatbot.json
        python3 -c "import yaml, json, sys; print(yaml.safe_dump(json.load(sys.stdin), de-
fault_flow_style=False, sort_keys=False))" < rag-app-text-chatbot.json > "${RAG_NAME}_v${RAG_VERSION}/
rag-app-text-chatbot.yaml"
      fi

      # get embedding models
      ngc registry model download-version ${EMBED_MODEL_URI}
      chmod -R o+rX ${EMBED_NAME}_v${EMBED_VERSION}

      # config compose.env
      cat << EOF > compose.env
      export MODEL_DIRECTORY="${LLM_MODEL_FOLDER}"
      export MODEL_NAME=${LLM_NAME}
      export NUM_GPU=${NUM_GPU}
      export APP_CONFIG_FILE=/dev/null
      export EMBEDDING_MODEL_DIRECTORY="/opt/data/${EMBED_NAME}_v${EMBED_VERSION}"
      export EMBEDDING_MODEL_NAME=${EMBED_TYPE}
      export EMBEDDING_MODEL_CKPT_NAME="${EMBED_TYPE}-${EMBED_VERSION}.nemo"
```

```
  export POSTGRES_HOST_IP=pgvector_db_ip_address
  export POSTGRES_PORT_NUMBER=5432
  export POSTGRES_DB=pgvector_db_name
  export POSTGRES_USER=pgvector_db_admin
  export POSTGRES_PASSWORD=encoded_pgvector_db_admin_password
EOF

  touch .file_downloaded
fi

# start NGC RAG
docker compose -f ${RAG_NAME}_v${RAG_VERSION}/docker-compose-vectordb.yaml up -d pgvector
source compose.env; docker compose -f ${RAG_NAME}_v${RAG_VERSION}/rag-app-text-chatbot.yaml up -d
```

b) Encode the cloud-init script to base64 format.

   You use a base 64 encoding tool, such as https://decode64base.com/ to generate the encoded versio of your cloud-init script.

c) Deploy the deep learning VM, passing the base64 value of the cloud-init script to the `user-data` input parameter.

   See Deploy a Deep Learning VM Directly on a vSphere Cluster in VMware Private AI Foundation with NVIDIA or Deploy a Deep Learning VM by Using the kubectl Command in VMware Private AI Foundation with NVIDIA.

2. If you are deploying the deep learning VM by using a catalog item in VMware Aria Automation, you provide the details of the pgvector PostgreSQL database after you deploy the virtual machine.

   a) Deploy the deep learning VM from Automation Service Broker.

      See Deploy a Deep Learning VM by Using a Self-Service Catalog in VMware Private AI Foundation with NVIDIA.

      Wait until the deployment is complete.

   b) Navigate to **Consume** > **Deployments** > **Deployments** and locate the deep learning VM deployment.

   c) In the **Workstation VM** section, save the details for SSH login to the virtual machine.

   d) Log in to the deep learning VM over SSH by using the credentials available in Automation Service Broker.

   e) Add the following pgvector variables to the `/opt/data/compose.env` file:

```
POSTGRES_HOST_IP=pgvector_db_ip_address
POSTGRES_PORT_NUMBER=5432
POSTGRES_DB=pgvector_db_name
POSTGRES_USER=pgvector_db_admin
POSTGRES_PASSWORD=encoded_pgvector_db_admin_password
```

   f) Restart the NVIDIA RAG multi-container application by running the following commands.

      For example, for NVIDIA RAG 24.03:

```
cd /opt/data

docker compose -f rag-docker-compose_v24.03/rag-app-text-chatbot.yaml down

docker compose -f rag-docker-compose_v24.03/docker-compose-vectordb.yaml down

docker compose -f rag-docker-compose_v24.03/docker-compose-vectordb.yaml up -d
```

Deploy a RAG Workload on a TKG Cluster

On a TKG cluster in a Supervisor, you can deploy a RAG workload based on the RAG Sample Pipeline from NVIDIA that uses a pgvector PostgreSQL database managed by VMware Data Services Manager.

- Verify that VMware Private AI Foundation with NVIDIA is available for the VI workload domain. See Deploying VMware Private AI Foundation with NVIDIA.
- Deploy a Vector Database in VMware Private AI Foundation with NVIDIA.

1. Provision a GPU-accelerated TKG cluster.

   See Deploying AI Workloads on TKG Clusters in VMware Private AI Foundation with NVIDIA.
2. Install the RAG LLM Operator.

   See Install the RAG LLM Operator.
3. Download the manifests for the NVIDIA sample RAG pipeline.

   See Sample RAG Pipeline.
4. Configure the sample RAG pipeline with the pgvector PostgreSQL database.
   a) Edit the sample pipeline YAML file.

      See Step 4 in Sample RAG Pipeline.
   b) In the YAML file, configure the sample pipeline with the pgvector PostgreSQL database by using the database's connection string.

      See Vector Database for RAG Sample Pipeline .
5. To provide an external IP for the sample chat application, in the YAML file, set `frontend.service.type` to `loadBalancer`.
6. Start the sample RAG pipeline.

   See Sample RAG Pipeline.
7. To access the sample chat application, run the following command to get the application's external IP address.

   ```
   kubectl -n rag-sample get service rag-playground
   ```
8. In a Web browser, open the sample chat application at `http://`*`application_external_ip`*`:3001/orgs/nvidia/models/text-qa-chatbot`.

Monitoring VMware Private AI Foundation with NVIDIA

You can monitor GPU metrics at the cluster and host level in the vSphere Client and VMware Aria Operations.

In VMware Aria Operations, you can monitor GPU metrics at the cluster, host system and host properties levels. For more information, see Private AI (GPU) Dashboards and Properties for vCenter Server Components in VMware Aria Operations.

In the vSphere Client, you can monitor GPU metrics in the following way:

- At the host level. See Hosts Performance Charts in vSphere.
- At the cluster level in custom charts. See Working with Advanced and Custom Charts in vSphere.

Documentation Legal Notice

vmware®
by Broadcom