NetApp Verified Architecture – Design Guide

NetApp AIPod™ with Lenovo for NVIDIA OVX™

Bobby Oommen, Abhinav Singh, Roney Daniel, NetApp
August 2024 | NVA-1172-DESIGN

In partnership

Abstract

This reference architecture pairs NVIDIA-Certified OVX Lenovo ThinkSystem servers, powered by NVIDIA L40S GPUs, with NVIDIA Spectrum networking to provide an optimal infrastructure solution to optimize and deploy LLMs (Large Language models). The intent of this document is to provide guidance relating to the storage for an OVX configuration. This platform is suitable for various Generative AI workloads, including RAG (Retrieval Augmented Generation), fine-tuning, and lightweight model training.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Executive summary

This document covers a reference architecture optimized for Generative AI workloads comprised of Lenovo ThinkSystem servers certified for NVIDIA OVX, NetApp AFF storage, NVIDIA networking and NVIDIA AI Enterprise software stack. The report covers all aspects of setup, configuration and best practices for NetApp AIPod with Lenovo for NVIDIA OVX systems to perform RAG inferencing on top of a LLM foundation. It provides orderable PNs and gives guidance to customers and partners to make the correct choice of NetApp storage for the solution. As organizations look to implement AI as a production application, they face challenges with workload scalability and data availability. This solution demonstrates how to address these challenges with an architecture that can scale to address computational and data management needs.

Program summary

NVIDIA OVX Certified Lenovo servers combined with NVIDIA networking and NetApp storage provide key benefits to customers such as:

- Flexible reference architecture for Generative AI workloads.
- Optimized infrastructure for high-performance and scalability of compute and storage
- Solution guidance in terms of Compute, Network and Storage.
- Reliability and security– Data protection, built-in ransomware protection, etc.
- Enhanced automated system management from XClarity for simple, faster deployment and configuration as well as updated security features to detect and protect from un-authorized access

NVIDIA Storage Partner Validation

NVIDIA OVX systems are optimized for small-model training, fine-tuning, and inferencing workloads. These workloads require storage which delivers the performance to maximize the GPU utilization. To help customers select the right storage for their AI workloads, NVIDIA created the storage partner validation program for OVX solutions, which defines a process for storage partners to validate their storage appliances with certified NVIDIA OVX systems for enterprise AI clusters. The solution details how the storage validation is done successfully with NetApp ONTAP to provide a scalable solution to meet the Generative AI needs using certified NVIDIA OVX systems. NVIDIA and NetApp have a deep partnership relationship to ensure partners and customers are getting an optimal AI solution to meet dynamic business needs.

Advantages of Running AI Workloads on NetApp

The solution objective is to help customers be successful in their AI journey, with an architecture that provides a powerful data management plane which provides the following benefits:

- Velocity – Handling large datasets at high velocity for versioning
- Disaggregation – Ability to scale performance and capacity independently
- Reliability – Data protection, built-in ransomware protection, and dynamic provisioning of storage
- Cloud Adjacency – Connect with resources across multiple data centers and clouds to load GenAI knowledgebase repositories
- Efficiency – Enterprise storage features such as compression and deduplication on data sets

- Secure Multi-Tenancy – Adaptive QoS to isolate multiple AI workloads along with encryption for data at-rest and in-flight and using multiple SVMs

Advantages of Running AI Workloads on Lenovo ThinkSystem Servers

Lenovo ThinkSystem Servers are designed to deliver optimal performance and energy efficiency for Artificial Intelligence (AI), High Performance Computing (HPC) and graphical workloads across an array of industries.

- Award winning performance and reliability - Lenovo x86 servers had the best uptime among all x86 platforms for the 10th straight year (ITIC Global Server Hardware, Server OS Reliability Report).
- Unique thermal and power efficiency - Lenovo Neptune solutions allow customers to optimize performance, density, and energy consumption for challenging extensive data set modeling and simulation or AI workloads.
- Simple Management - XClarity Controller2 (XCC2) provides advanced service-processor control, monitoring, and alerting functions. The XCC2 consolidates the service processor functionality, super I/O, video controller, and remote presence capabilities into a single chip on the server system board.
- Built-in security - The ThinkSystem SR675 V3 includes Platform Firmware Resiliency (PFR) hardware Root of Trust (RoT) which enables the system to be NIST SP800-193 compliant, further enhancing key platform subsystem protections against unauthorized firmware updates and corruption, to restore firmware to an integral state, and to closely monitor firmware for possible compromise from cyber-attacks.
- Lenovo AI-POD is built upon a comprehensive support model. Through solution-level interoperability testing Lenovo and NetApp can give customers confidence in a supported environment based on proven best practices while still tailoring it exactly to the customer's needs. That means that the infrastructure is not just supported on a component break and fix or "box"-level, but with a holistic perspective including software, firmware and even firmware-settings.
- Lenovo Services and NVIDIA Services provide the necessary experience to bring your Generative AI environment to life.

Solution overview

This engineered solution offers the following benefits:

- Inferencing implementation with AI models on a certified OVX platform with NetApp storage and Lenovo servers
- Predictable facilitation and effective deployment and management of full-stack OVX solutions for AI by testing and documenting end-to-end infrastructure
- Highly available and scalable platform to create repeatable building blocks

Target audience

The document is intended for Data scientists, Engineers, IT operations and field consultants who want to take advantage of an infrastructure built to deliver AI workload needs. Having prior knowledge on AI, container micro-services, networking and its components will help during the implementation phase.

Technology requirements

This section covers the hardware and software which are needed for this solution.

Hardware requirements

Table 1 lists all the hardware used for this solution and a detailed bill of materials is available in the last section of this document.

Table 1) Hardware requirements.

Hardware	Quantity	Comment
Lenovo ThinkSystem SR675 V3 – OVX Certified	2	App Nodes
Lenovo ThinkSystem SR635 V3	3	Control Nodes
NVIDIA SN5600 800Gbe switch	1	East/West Traffic – Cumulus Linux 5.9.1
NVIDIA SN4600 200Gbe switch	1	North/South Traffic – Cumulus Linux 5.9.1
NetApp AFF C800 with 18 * 15.3TB drives	1	HA Pair

Software requirements

Table 2 lists all the software stack which was used for this solution.

Table 2) Software requirements

Software	Version	Comment
Ubuntu	22.04.4 LTS	OS on the Kubernetes (K8s) nodes (control and app nodes)
Kubernetes (upstream)	v1.28.11	Environment to run AI and Framework
Containerd Runtime	1.6.33	Runtime environment to run containers
NVIDIA GPU Operator	v24.3.0	Manages GPU resources and drivers in a K8s cluster
NVIDIA Driver Version	550.54.15	NVIDIA GPU driver
NVIDIA CUDA Toolkit	12.4	Software for GPU-accelerated computing applications
NVIDIA Bluefield-3 Firmware	32.41.10	Firmware for BlueField-3 DPU
NVIDIA ConnectX-7 Firmware	28.41.1000	Firmware for ConnectX-7
NVIDIA DOCA	v2.7.0	Software framework for NVIDIA BlueField and ConnectX-7 networking devices
NetApp ONTAP	9.15.1	Storage operating system on AFF C800
NetApp DataOps toolkit	2.5.0	Python library for development/training workspace and inference server management.
NetApp Astra Trident CSI Plugin	24.06.0	Fully supported open-source storage orchestrator for containers and K8s distributions, including Red Hat OpenShift.
NVAIE	5.0	Software optimized for the development & deployment of AI
Milvus	2.4.4	Open-source vector database
NVIDIA NIM for LLM	1.0.0	Framework for hosting LLM for managed environments
NVIDIA NeMo Retriever	24.06	Collection of generative AI microservices to connect custom models
Multi Turn RAG pipeline	24.06	Multi Turn conversational chat bot

Concepts and Components

NVIDIA OVX

NVIDIA OVX™ is part of the NVIDIA-Certified Systems program which delivers industry-leading performance to accelerate the next generation of AI-enabled workloads in the data center, bringing together NVIDIA GPUs, with high-speed secure networking for a diverse range of workloads. OVX is a validated architecture powered by NVIDIA L40S GPUs for breakthrough performance and NVIDIA AI Enterprise software for training, fine-tuning, and deploying Generative AI. ConnectX®-7 network adapters and NVIDIA Bluefield-3™ DPUs provide ultra-fast, high-bandwidth communication and the ability to efficiently scale across 100s of OVX servers with low-latency ethernet or InfiniBand configurations.

Lenovo ThinkSystem SR675 V3 Server

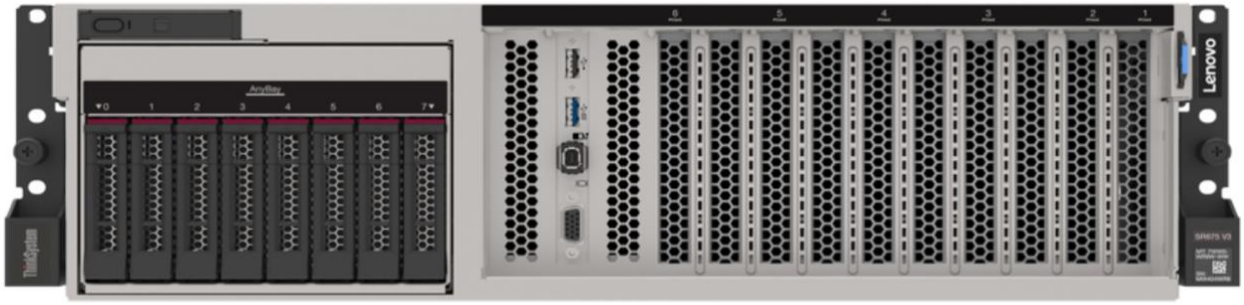

Figure 1) Lenovo ThinkSystem SR675 V3 Server

The Lenovo ThinkSystem SR675 V3 is a versatile GPU-rich 3U rack server, designed for the ultimate flexibility. The server can support up to eight double-wide or single-wide GPUs or the NVIDIA HGX 4-GPU offerings with NVLink and Lenovo Neptune hybrid liquid-to-air cooling. The GPU choice includes the vast NVIDIA Hopper, Lovelace, and Ampere datacenter portfolio and provides best-in-class cooling for the accelerators positioned in the front and allows both for front and rear IO connectivity for maximum graphic performance and IO throughput. The SR675 V3 is built on one or two 4th Generation AMD EPYC™ Processors, up to 24 TruDDR5 DIMMs and a choice of high-speed NVMe storage and networking. The OVX node configuration is built on Lenovo's Think System SR675 V3 which leverages PCIe gen 5 and the latest AMD EPYC processors.

Lenovo's SR675 V3 server OVX configuration (Figure 1) with L40S NVIDIA GPU option plays a pivotal role in enabling businesses to enhance foundational generative AI models. This server configuration provides enterprises with the capability to seamlessly tailor and deploy generative AI applications, incorporating cutting-edge features such as intuitive chatbots, advanced search systems, and efficient summarization tools. The NVIDIA BlueField-3 DPU card which comes with the configuration empowers Gen AI workloads with the integration into Metaverse applications doing North/South traffic with the NetApp storage.

A Lenovo SR675 V3 server comes with following OVX certified configuration:

- 2x AMD EPYC 9634 processors, 84C, 2.25GHz or 2x AMD EPYC 9634 processors, 32C, 3.25GHz
- 4x 128GB TruDDR5 3DS RDIMMs
- 1x 1GbE 4-port OCP Adapter – Management Connector
- 4x NVIDIA L40S GPUs

- 2x NVIDIA ConnectX-7 200 GbE dual port, 1x NVIDIA B3220 BlueField-3 200 GbE dual port DPU (North to South connectivity - Storage integration)

The Lenovo ThinkSystem SR635 V3 is a 1-socket 1U server that features the AMD EPYC 9004 "Genoa" family of processors. With up to 128 processor cores and support for the new PCIe 5.0 standard for I/O, the SR635 V3 offers the ultimate in one-socket server performance in a 1U form factor.

Combining performance and flexibility, the SR635 V3 server is a great choice for enterprises of all sizes. The server offers a broad selection of drive and slot configurations and offers high performance features that industries such as finance, healthcare and telco need. Outstanding reliability, availability, and serviceability (RAS) and high-efficiency design can improve your business environment and can help save operational costs.

Lenovo XClarity Systems Management

Lenovo XClarity provides fast, flexible, and scalable delivery of Lenovo infrastructure. XClarity integrates easily into Lenovo servers to automate provisioning and operations management, and into Lenovo switches and storage to automate operations management.

By seamlessly integrating with a wide range of external IT applications, you can effectively manage Lenovo infrastructure within your existing software tools' console, ensuring a cohesive and efficient workflow for IT operations.

The **NVIDIA ConnectX-7** family of Remote Direct Memory Access (RDMA) network adapters supports InfiniBand and Ethernet protocols and a range of speeds up to 400Gb/s enabling a wide range of smart, scalable, and feature-rich networking solutions that address traditional enterprise needs as well as some of the most-demanding AI, scientific computing, and hyperscale cloud data center workloads

The **NVIDIA BlueField-3** DPU is a 400 gigabits per second (Gb/s) data processing unit designed for data center infrastructure workloads combining computing, software defined networking, storage and security functions, offering low latency, high bandwidth and computing efficiency for HPC, AI and cloud applications to address the most demanding workloads.

The **NVIDIA L40S GPU** is based on the Ada Lovelace architecture, delivers multi-workload acceleration for large language model (LLM) inference and training, graphics, and video applications. As the premier platform for multi-modal generative AI, the L40S GPU provides end-to-end acceleration for inference, training, graphics, and video workflows to power the next generation of AI-enabled audio, speech, 2D, video, and 3D applications.

Networking connectivity

Ethernet was used here with NFS over RDMA to do the North to South data traffic between SR675 servers and NetApp storage which gives a cost-effective, simple and easy to scale solution.

NVIDIA Spectrum-4, with Cumulus Linux enables the extreme networking performance and robust security needed for data center infrastructure at scale. NVIDIA spectrum configuration consists of the NVIDIA Spectrum-4 switch family, NVIDIA ConnectX®-7 SmartNIC, NVIDIA BlueField®-3 DPU and the DOCA™ data center infrastructure software to supercharge cloud-native applications at scale. Spectrum-4 switches (Figure 2) allow nanosecond timing precision — which is an improvement of five to six orders of magnitude compared to typical, millisecond-based data centers.

For this solution a Spectrum-3 (Figure 3) switch was also used to connect to NetApp AFF C800 storage. With the newer generation of NetApp AFF systems customers can avoid this switch and use the same SN5600 switch for both East/West and North/South traffic.

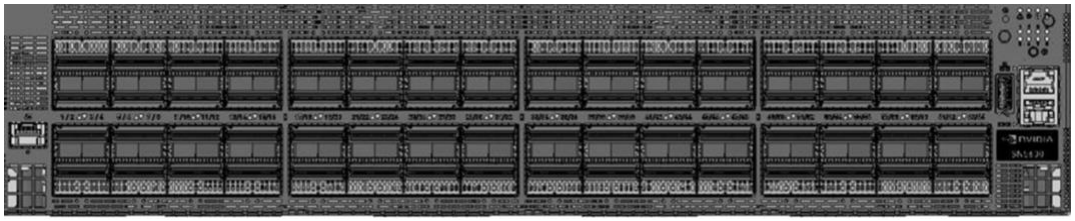

Figure 2) SN5600 - 64 ports of 800Gbe

Figure 3) SN4600 64 ports of 200Gbe

Management Ethernet switches

The NVIDIA® Spectrum™ SN2000 series switches are the 2nd generation of NVIDIA Ethernet switches, purpose-built for leaf, spine, and super-spine datacenter applications. The SN2201 is ideal as an out-of-band (OOB) management switch, or as a top of rack (ToR) switch connecting to 48 x 1G Base-T host ports with four non-blocking 100 GbE spine uplinks. These switches are used for monitoring activities through Lenovo XClarity software and virtual operations if choosing a virtual environment. The choice of a management switch could be any compliant switch including existing switching network in the data center.

For this setup, existing switches in the data center were used as Management Infrastructure.

NetApp ONTAP 9

NetApp ONTAP 9 is the latest generation of storage management software from NetApp that enables businesses to modernize infrastructure and to transition to a cloud-ready data center. With industry-leading data management capabilities, ONTAP enables you to manage and protect your data with a single set of tools regardless of where that data resides. You can also move data freely to wherever you need it: the edge, the core, or the cloud. ONTAP 9 includes numerous features that simplify data management, accelerate and protect your critical data, and future-proof your infrastructure across hybrid cloud architectures.

Simplify Data Management

Data management is crucial for your enterprise IT operations so that you can use appropriate resources for your applications and datasets. ONTAP includes the following features to streamline and simplify your operations and reduce your total cost of operation:

- **Inline data compaction and expanded deduplication.** Data compaction reduces wasted space inside storage blocks, and deduplication significantly increases effective capacity.
- **Minimum, maximum, and adaptive quality of service (QoS).** Granular QoS controls help maintain performance levels for critical applications in highly shared environments.

- **ONTAP FabricPool.** This feature provides automatic tiering of cold data to public and private cloud storage options, including Amazon Web Services (AWS), Azure, and NetApp StorageGRID® object-based storage.

Accelerate and Protect Data

ONTAP delivers superior levels of performance and data protection and extends these capabilities with the following features:

- **High performance and low latency.** ONTAP offers the highest possible throughput at the lowest possible latency.
- **NetApp ONTAP FlexGroup technology.** A FlexGroup volume is a high-performance data container that can scale linearly to up to 20PB and 400 billion files, providing a single namespace that simplifies data management.
- **Data protection.** ONTAP provides built-in data protection capabilities with common management across all platforms.
- **NetApp Volume Encryption.** ONTAP offers native volume-level encryption with both onboard and external key management support.

Future-Proof Infrastructure

ONTAP 9 helps meet your demanding and constantly changing business needs:

- **Seamless scaling and nondisruptive operations.** ONTAP supports the nondisruptive addition of capacity to existing controllers and to scale-out clusters. You can upgrade to the latest technologies, such as NVMe and end-to-end 100Gbe, without costly data migrations or outages.
- **Cloud connection.** ONTAP is one of the most cloud-connected storage management software, with options for software-defined storage (ONTAP Select) and cloud-native instances (NetApp Cloud Volumes Service) in all public clouds.
- **Integration with emerging applications.** By using the same infrastructure that supports existing enterprise apps, ONTAP offers enterprise-grade data services for next-generation platforms and applications such as OpenStack, Hadoop, and MongoDB.

NetApp Astra Trident

Trident is an open-source storage orchestrator (Figure 4) developed and maintained by NetApp that greatly simplifies the creation, management, and consumption of persistent storage for Kubernetes workloads. Trident, itself a Kubernetes-native application, runs directly within a Kubernetes cluster. With Trident, Kubernetes users (developers, data scientists, Kubernetes administrators, and so on) can create, manage, and interact with persistent storage volumes in the standard Kubernetes format that they are already familiar with. At the same time, they can take advantage of NetApp advanced data management capabilities and a data fabric that is powered by NetApp technology. Trident abstracts away the complexities of persistent storage and makes it simple to consume. For more information, visit the Trident website.

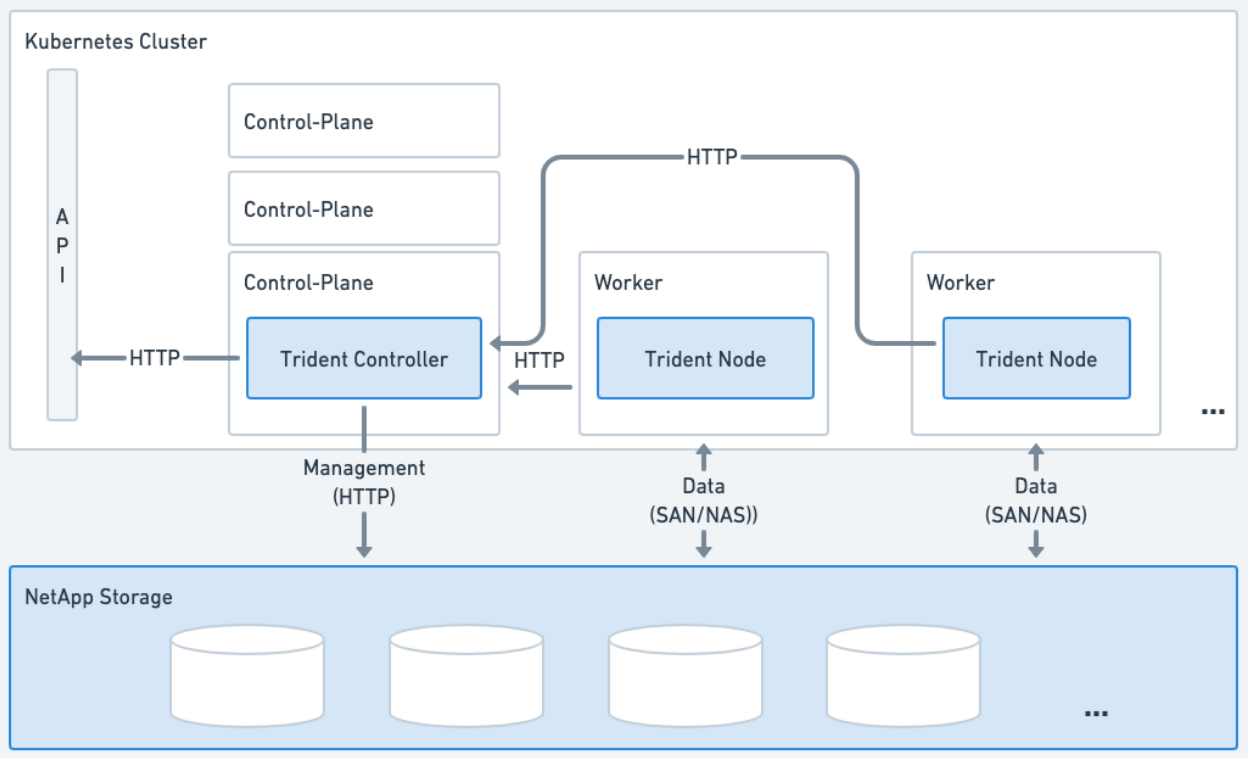

Figure 4) NetApp Astra Trident deployed on the Kubernetes Cluster

NetApp DataOps Toolkit

The NetApp DataOps Toolkit (Figure 5) is a Python library that makes it easy for developers, data scientists, and data engineers to perform numerous data management tasks. These tasks include provisioning a new data volume or development workspace, cloning a data volume or development workspace almost instantaneously, and creating a NetApp Snapshot™ copy of a data volume or development workspace for traceability and baselining. This Python library can function as either a command-line utility or a library of functions that can be imported into any Python program or Jupyter Notebook.

The DataOps Toolkit supports Linux and macOS hosts. The toolkit must be used in conjunction with a NetApp data storage system or service. It simplifies various data management tasks that are executed by the data storage system or service. To facilitate this simplification, the toolkit communicates with the data storage system or service through an API.

The NetApp DataOps Toolkit for Kubernetes abstracts storage resources and Kubernetes workloads up to the data-science workspace level. These capabilities are packaged in a simple, easy-to-use interface that is designed for data scientists and data engineers. Using the familiar form of a Python program, the Toolkit enables data scientists and engineers to provision and destroy JupyterLab workspaces in just seconds. These workspaces can contain terabytes, or even petabytes, of storage capacity, enabling data scientists to store all their training datasets directly in their project workspaces from a central location.

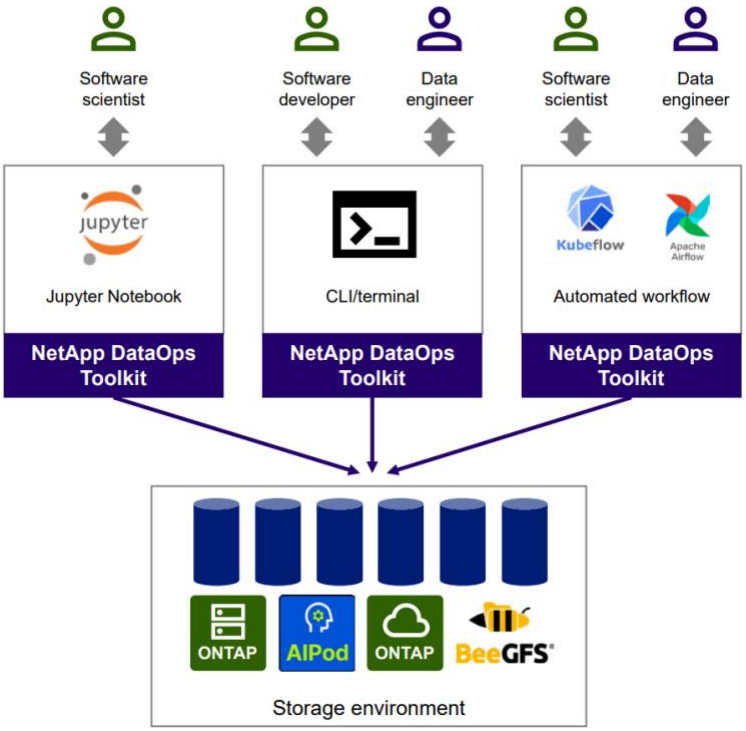

Figure 5) NetApp DataOps Tool Kit

NVIDIA AI Enterprise

Lenovo offers the NVIDIA AI Enterprise (NVAIE) cloud-native enterprise software. NVIDIA AI Enterprise is an end-to-end, cloud-native suite of AI and data analytics software, optimized, certified, and supported by NVIDIA to run on VMware vSphere and bare-metal with NVIDIA-Certified Systems™. It includes key enabling technologies from NVIDIA for rapid deployment, management, and scaling of AI workloads in the modern hybrid cloud.

NVIDIA AI Enterprise is licensed on a per-GPU basis. NVIDIA AI Enterprise products can be purchased as either a perpetual license with support services, or as an annual or multi-year subscription.

- The perpetual license provides the right to use the NVIDIA AI Enterprise software indefinitely, with no expiration. NVIDIA AI Enterprise with perpetual licenses must be purchased in conjunction with one-year, three-year, or five-year support services. A one-year support service is also available for renewals.
- The subscription offerings are an affordable option to allow IT departments to better manage the flexibility of license volumes. NVIDIA AI Enterprise software products with subscription includes support services for the duration of the software's subscription license

The features of NVIDIA AI Enterprise Software are listed in table 3.

Table 3) Features of NVIDIA AI Enterprise Software (NVAIE)

Features	Supported in NVIDIA AI Enterprise
Per GPU Licensing	Yes
Compute Virtualization	Supported
Windows Guest OS Support	No support
Linux Guest OS Support	Supported
Maximum Displays	1
Maximum Resolution	4096 x 2160 (4K)
OpenGL and Vulkan	In-situ Graphics only
CUDA and OpenCL Support	Supported
ECC and Page Retirement	Supported
MIG GPU Support	Supported
Multi-vGPU	Supported
NVIDIA GPUDirect	Supported
Peer-to-Peer over NVLink	Supported
GPU Pass Through Support	Supported
Bare metal Support	Supported
AI and Data Science applications and Frameworks	Supported
Cloud Native ready	Supported

Note: Maximum 10 concurrent VMs per product license

Table4 lists the ordering part numbers and feature codes.

Table 4) Part numbers for NVIDIA AI Enterprise software

Part number	Feature code	Description
AI Enterprise Perpetual License		
7S02001BWW	S6YY	NVIDIA AI Enterprise Perpetual License and Support per GPU, 5 Years
7S02001EWW	S6Z1	NVIDIA AI Enterprise Perpetual License and Support per GPU, EDU, 5 Years
AI Enterprise Subscription License		
7S02001FWW	S6Z2	NVIDIA AI Enterprise Subscription License and Support per GPU, 1 Year
7S02001GWW	S6Z3	NVIDIA AI Enterprise Subscription License and Support per GPU, 3 Years
7S02001HWW	S6Z4	NVIDIA AI Enterprise Subscription License and Support per GPU, 5 Years
7S02001JWW	S6Z5	NVIDIA AI Enterprise Subscription License and Support per GPU, EDU, 1 Year
7S02001KWW	S6Z6	NVIDIA AI Enterprise Subscription License and Support per GPU, EDU, 3 Years
7S02001LWW	S6Z7	NVIDIA AI Enterprise Subscription License and Support per GPU, EDU, 5 Years

Find more information in the NVIDIA AI Enterprise Sizing Guide.

Solution Deployment

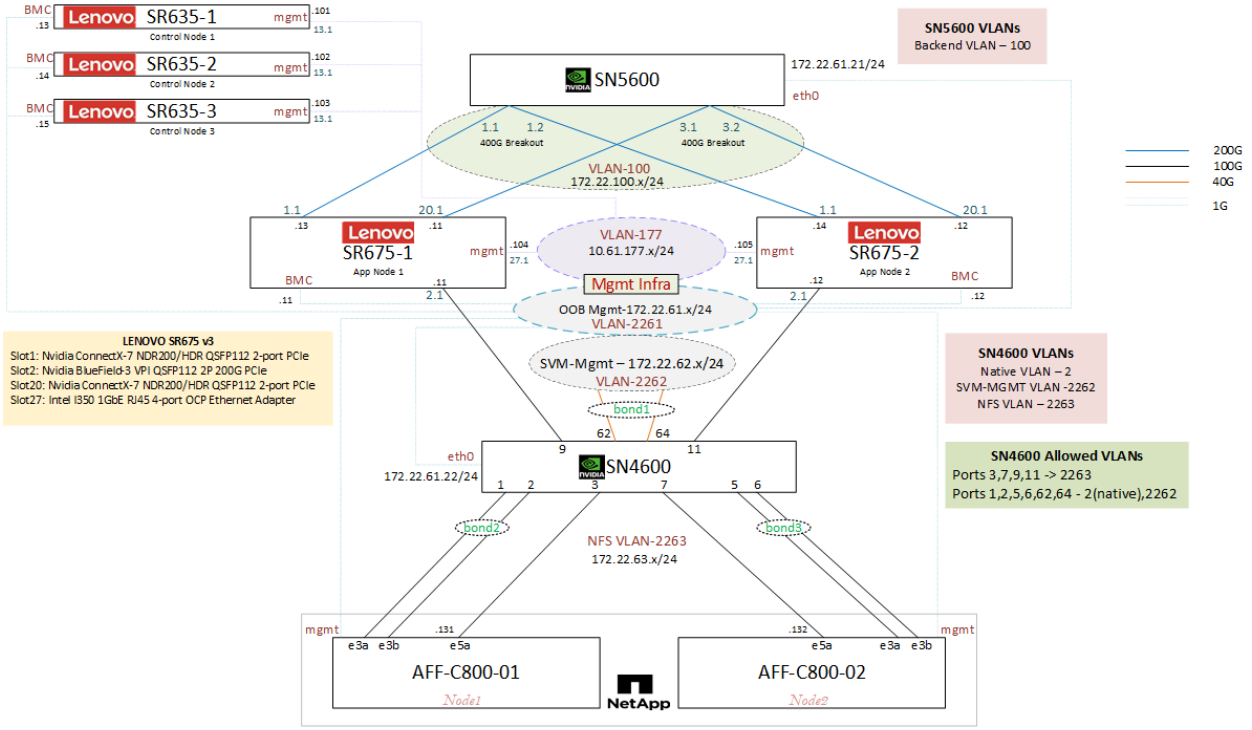

Figure 6) Solution Topology

The solution topology (Figure 6) consists of two Lenovo ThinkSystem SR675 V3 and three Lenovo ThinkSystem SR635 V3 servers, one NVIDIA SN5600 switch, one NVIDIA SN4600 switch and a NetApp AFF C800 storage. For redundancy and high availability, network switches are deployed in pairs. However, the validated topology has just one NVIDIA SN4600 switch for North-South traffic and a SN5600 switch for East-West traffic. The SR675 V3 servers have two ConnectX-7 adapters for East-West traffic and one B3320 BlueField-3 adapter for North-South traffic to the NetApp storage. The first port on each adapter is connected to one of the switches and the second port is reserved for connecting to a redundant switch. All servers have the option of 1G/10G/25G/100G 4-port OCP adapter and the first port connects to the management infrastructure. For this configuration, a 1G OCP adapter was utilized. However, customers can choose and use an adapter that meets their specific needs. Note that only one OCP card can be installed on the server. For additional details on supported OCP adapters, please refer here.

The SN4600 switch has a 100G link connecting to the B3320 BlueField-3 adapter on each SR675 server. This link carries the NFS over RDMA traffic from the SR675 OVX servers. The SN4600 switch has a bond interface with two 40G links connecting to the management infrastructure and it carries the In-Band management VLANs. It also has bond interfaces with 100G ports connecting to each NetApp AFF C800 controllers carrying the SVM management VLAN and any other VLAN of interest. The bond interfaces are configured as trunks with native VLAN (2).

The SN5600 has two 400G breakout ethernet cables, each providing two 200G links. These links connect to ConnectX-7 adapters on SR675 servers and carry the East-West GPU traffic.

Each NetApp controller has a 100G port (e5a) carrying the NFS over RDMA traffic. The second 100G port (e5b) is reserved to connect to a redundant switch, when available. Each controller has two 100G ports

(e3a, e3b) in the interface group (`ifgroup`) forming a bond interface and carrying the SVM management VLAN and standard NFS traffic

Deployment of the NVIDIA AI Enterprise software layer requires a Kubernetes platform running on Linux to host the containerized applications. Customers will have the choice of deploying NetApp AIPod from the following

- Open source kubernetes running on Ubuntu Linux on bare metal servers
- Red Hat Open Shift running on Red Hat Enterprise Linux (RHEL) on bare metal servers

For this solution Ubuntu on Bare metal servers was used to do the solution validation and high-level guidance is provided in **Alternate Deployment** section for RedHat OpenShift on bare metal.

This section describes the tasks that you must complete to deploy a upstream Kubernetes cluster to implement the solution that includes Astra Trident™ and NetApp DataOps Toolkit. For a list of Kubernetes versions that are supported by Trident, see the Trident documentation.

Deployment options on Key components

Note: SN5600 was used in the solution as part of the future proofing for Spectrum-X

Customers and Partners should follow the guidelines below to deviate away from the topology in figure 6 to tailor to any existing data center infrastructure.

- Minimum of two Lenovo NVIDIA-Certified OVX ThinkSystem SR675 V3 is required
- The SN5600 is optional and could be replaced with a SN4600 or SN3700 for the E/W traffic. The SN4600 should be used for larger deployments that don't want to leverage NVIDIAs Spectrum-x network architecture, and the SN3700 (32 port 200gbE switch) should be used for deployments that don't plan to expand after their initial small deployment.
- SN4600 is NOT a requirement if a newer NetApp storage system [AFF A70, AFF A90, AFF A1K] is used and the storage traffic (N/S) will be shared with the E/W switch.

Lenovo ThinkSystem Server configuration –App nodes

This solution is built with five Lenovo ThinkSystem servers, three ThinkSystem SR635 V3 servers used as control nodes and two ThinkSystem SR675 V3 servers used as application nodes. The table below shows the options available in a Lenovo OVX App node configuration however there is also a more strictly configured BOM at the end of the document.

Choosing resources for Control nodes

- For true HA, you need to have three control nodes on three separate physical hosts.
- Customers will have the choice of running the control plane on the worker nodes provide there are at least three physical servers.
- Alternatively, if the customer already has existing infrastructure either in the form of virtualized (VMs) or physical servers, those could be used as control nodes provided the configurations (CPU/RAM) are identical.

Prerequisites

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You have installed NVIDIA-OVX Certified Lenovo servers – ThinkSystem SR675 V3.

2. You have installed at least one SN4600, SN5600 NVIDIA switches (For redundancy, switches are installed as pairs in typical deployments)
3. You have installed a NetApp AFF A-series or C-series storage systems
4. You have installed a supported operating system on all K8s control and app nodes and a ha-proxy load balancer.
5. Having a NVIDIA NGC account to access all the frameworks.
6. You have the NGC CLI available on your client machine. You can download the CLI from https://ngc.nvidia.com/setup/installers/cli.

Network configuration

In this section, we will discuss the configuration of NVIDIA Spectrum switches. In the validated topology, there is one switch for North-South (SN4600) and another switch (SN5600) for East-West traffic. Note that standard installation would require at least a pair of switches for link and switch redundancy.

In the reference topology, the NVIDIA Cumulus Linux is used to configure the spectrum switches. These switches are configured as Layer 2 devices, so the Layer 1 & Layer 2 configuration commands are discussed here.

The following section provides a list of commands used to configure the switches. After each step, you may apply the configuration using `nv config apply` command.

1. Configure the system hostname.

```
nv set system hostname <hostname>
```

2. Configure the management interface.

```
nv set interface eth0 ip address <ip address/mask>
```

3. Set link speed and link state.

```
nv set interface <interface_name> link speed <speed>
nv set interface <interface_name> link state up
```

4. Configure a bridge domain and set various parameters such as VLAN, untagged VLAN, STP priority (optional).

```
nv set bridge domain <bridge_domain> vlan <VLAN>
nv set bridge domain <bridge_domain> untagged <VLAN>
nv set bridge domain <bridge_domain> stp priority <priority>
```

5. Configure a bond interface, add bond members and add the interface as a trunk to the bridge domain

```
nv set interface <bond_interface> bond member <member ports>
nv set interface <bond_interface> bond mode lacp
nv set interface <bond_interface> bridge domain <bridge_domain> vlan <VLAN>
```

6. Configure an access port and add it to the bridge domain

```
nv set interface <interface_name> bridge domain <bridge_domain> access <VLAN>
```

7. Configure switchport breakout.

```
nv set interface <interface_name> link breakout <breakout_option>
```

8. Enable RoCE protocol.

```
nv set qos roce
```

9. Save the configuration across reboots.

```
nv config save
```

10. Display the configuration

```
nv config show
```

NVIDIA SN4600

In the validated topology, the SN4600 switch is configured as a VLAN-aware bridge (bridge1) that carries SVM management VLAN (2262) and NFS VLAN (2263). Switch ports swp1-2, swp5-6 and swp62, swp64 form bond interfaces bond1, bond2 and bond3 respectively and carry tagged VLAN 2262 and untagged (native) VLAN 2. Switch ports swp3, swp7, swp9 and swp11 are configured as access ports and carry NFS VLAN 2263 between worker nodes and storage controllers. Eth0 is configured as switch management interface, and it connects to the OOB management network in vlan 2261.

For more information on configuring VLAN-aware bridge, refer to the following URL.

VLAN-aware Bridge Mode | Cumulus Linux 5.8 (nvidia.com)

The SN4600 switch configuration is shown below.

```
root@aipod-4600:mgmt:/var/home/cumulus# nv config show
- header:
model: MSN4600C
nvue-api-version: nvue_v1
rev-id: 1.0
version: Cumulus Linux 5.9.1
- set:
bridge:
      domain:
        bridge1:
          stp:
            priority: 8192
          untagged: 2
          vlan:
            2262-2263: {}
interface:
      bond1:
        bond:
          member:
            swp62: {}
            swp64: {}
          mode: lacp
      bond1-3:
        bridge:
          domain:
            bridge1:
              vlan:
                '2262': {}
        type: bond
      bond2:
        bond:
          member:
            swp1: {}
            swp2: {}
      bond3:
        bond:
          member:
            swp5: {}
            swp6: {}
      eth0:
        ip:
          address:
            172.22.61.22/24: {}
          gateway:
            172.22.61.1: {}
```

```
        type: eth
      swp1-8:
        link:
          speed: 100G
      swp1-12:
        link:
          state:
            up: {}
      swp1-12,62,64:
        type: swp
      swp3,7,9,11:
        bridge:
          domain:
            bridge1:
              access: 2263
      swp9-12:
        link:
          fast-linkup: on
          speed: auto
      swp62,64:
        link:
          speed: 40G
qos:
      roce:
        enable: on
system:
      hostname: aipod-4600!
```

The CLI command (Figure 7) displays the port-vlan configuration on this switch.

```
root@aipod-4600:mgmt:/var/home/cumulus# nv show bridge port-vlan
domain        port            vlan   tag-state
-------    ---------     ---------   ---------
bridge1      bond1              2    untagged
                             2262      tagged
             bond2              2    untagged
                             2262      tagged
             bond3              2    untagged
                             2262      tagged
              swp3           2263    untagged
              swp7           2263    untagged
              swp9           2263    untagged
             swp11           2263    untagged
root@aipod-4600:mgmt:/var/home/cumulus#
```

Figure 7) SN4600 Switch Port VLAN configuration

The CLI command (Figure 8) displays the global status of RoCE protocol on SN4600 switch.

```
root@aipod-4600:mgmt:/# nv show qos roce
                    operational  applied   pending
------------------  -----------  --------  --------
enable                           on        on
mode                lossless     lossless  lossless
pfc
  pfc-priority      3
  rx-enabled        enabled
  tx-enabled        enabled
  cable-length      100
congestion-control
  congestion-mode   ECN
  enabled-tc        0,3
  min-threshold     146.48 KB
  max-threshold     1.43 MB
  probability       100
trust
  trust-mode        pcp,dscp
lldp-app-tlv
  priority          3
  protocol-id       4791
  selector          UDP
```

Figure 8) SN4600 Global status of RoCE protocol

The CLI command (Figure 9) displays the interface level status of RoCE protocol on swp9.

```
root@aipod-4600:mgmt:/# nv show int swp9 qos roce status
                    operational
------------------  -------------
pfc
  pfc-priority      3
  rx-enabled        yes
  tx-enabled        yes
trust
  trust-mode        pcp,dscp
congestion-control
  congestion-mode   ecn, absolute
  enabled-tc        0,3
  min-threshold     153.00 KB
  max-threshold     1.43 MB
mode                lossless



RoCE PCP/DSCP->SP mapping configurations
========================================
          pcp  dscp  switch-prio
    ----  ---  ----  -----------
```

Figure 9) SN4600 Interface level RoCE protocol status

NVIDIA SN5600

The NVIDIA SN5600 switch is configured as a VLAN-aware bridge (bridge2) for East-West traffic between worker nodes. Depending on the type of cable used, the switch ports can support up to 800Gbps ethernet or InfiniBand traffic. In the validated topology, a 400G to 2x200Gbps Ethernet breakout cable is used on swp1 and swp3 and the 200G breakout interfaces are connected to the first port on CX7 cards installed in the worker nodes. The second port on each adapter can be used for additional link redundancy when you have a redundant switch. The switch ports are configured as access ports in VLAN 100.

The SN5600 switch configuration is shown below.

```
cumulus@aipod-sn5600-1:mgmt:~$ nv config show
- header:
    model: SN5600
    nvue-api-version: nvue_v1
    rev-id: 1.0
    version: Cumulus Linux 5.9.1
- set:
    bridge:
      domain:
        bridge2:
          vlan:
            '100': {}
    interface:
      eth0:
        ip:
          address:
            172.22.61.21/24: {}
          gateway:
            172.22.61.1: {}
        type: eth
      swp1,3:
```

```
        link:
          breakout:
            2x: {}
          state:
```

```
            up: {}
      swp1,3,swp1s0-1,swp3s0-1:
        type: swp
      swp1s0-1,swp3s0-1:
        bridge:
          domain:
            bridge2:
              access: 100
```

The `nv show interface` command (Figure 10) displays the switch port Layer 1 configuration such as link speed, MTU and operation link status. In this example, the switch port information connecting to the CX7 ports on the server is displayed.

```
cumulus@aipod-sn5600-1:mgmt:~$ nv show interface --filter "Speed=200G&MTU=9216"
Interface  Admin Status  Oper Status  Speed  MTU   Type  Remote Host  Remote Port
---------  ------------  -----------  -----  ----  ----  -----------  -----------
swp1s0     up            up           200G   9216  swp
swp1s1     up            up           200G   9216  swp
swp3s0     up            up           200G   9216  swp
swp3s1     up            up           200G   9216  swp
cumulus@aipod-sn5600-1:mgmt:~$
```

Figure 10) SN5600 Switch Layer 1 configuration

The `nv show bridge port-vlan` command (Figure 11) displays the port-vlan configuration on the SN5600 switch.

```
cumulus@aipod-sn5600-1:mgmt:~$ nv show bridge port-vlan
domain        port            vlan   tag-state
-------    ---------     ---------   ---------
bridge2      swp1s0            100    untagged
             swp1s1            100    untagged
             swp3s0            100    untagged
             swp3s1            100    untagged
cumulus@aipod-sn5600-1:mgmt:~$
```

Figure 11) SN5600 Switch Port VLAN configuration

The `bridge sdb show` command (Figure 12) displays the MAC addresses learnt on the switch. In this example, the MAC address of the CX7 adapter ports connected to the SN5600 switch in VLAN 100 as well as the switch port MAC addresses are displayed.

Figure 12) SN5600 Switch MAC address table

Lenovo ThinkSystem server configuration

This section talks about the Lenovo ThinkSystem servers initial setup, before an operating system can be installed. In this section, we will talk about the BMC, Local Boot, Management and BlueField-3 adapter settings. The ThinkSystem SR635 V3 servers do not have the BlueField-3 adapters, hence the BF-3 settings are not applicable on those servers. The major steps are listed below.

- Configure BMC using Lenovo XClarity Provisioning Manager
- Configure RAID for local boot from M.2 NVMe drives
- Configure BlueField-3 adapters DPU and CPU settings
- BlueField-3 NIC configuration
- Mount virtual media and restart the server.

1. Configure BMC using Lenovo XClarity Provisioning Manager

   Before you can access the Lenovo XClarity Controller over your network, you need to specify how Lenovo XClarity Controller will connect to the network. Depending on how the network connection is implemented, you might need to specify a static IP address as well.

   If you are not using DHCP, the following methods are available to set the network connection for the Lenovo XClarity Controller.

   - If a monitor is attached to the server, you can use Lenovo XClarity Provisioning Manager to set the network connection.
   - If no monitor is attached to the server, you can set the network connection through the Lenovo XClarity Controller interface. Connect an Ethernet cable from your laptop to XCC system management port on your server. The laptop should be in the same subnet as the server's default IP settings.
   - If you are using the Lenovo XClarity Administrator Mobile app from a mobile device, you can connect to the Lenovo XClarity Controller through the Lenovo XClarity Controller USB connector on the server.

   The Lenovo XClarity Provisioning Manager was used in the lab to connect the Lenovo XClarity Controller to the management network. Refer to the following steps more details.

   Step 1. Start the server.

   Step 2. Press the key specified in the on-screen instructions to display the Lenovo XClarity Provisioning Manager interface.

Step 3. Go to **LXPM → UEFI Setup → BMC Settings** to specify how the Lenovo XClarity Controller will connect to the network.

– If you choose a static IP connection, make sure that you specify an IPv4 or IPv6 address that is available on the network.

– If you choose a DHCP connection, make sure that the MAC address for the server has been configured in the DHCP server.

Step 4. Click OK to apply the setting and wait for two to three minutes.

Step 5. Use an IPv4 or IPv6 address to connect Lenovo XClarity Controller.

For more information, refer to https://pubs.lenovo.com/lxpm-overview/.

Note: The Lenovo XClarity Controller is set initially with a username of USERID and password of PASSW0RD (with a zero, not the letter O). This default user setting has Supervisor access. It is required to change this username and password during your initial configuration for enhanced security.

Note: Once you have configured BMC, you can access the XClarity Controller interface using the web interface.

Check (Figure 13) for the Lenovo XClarity Provisioning Manager screen.

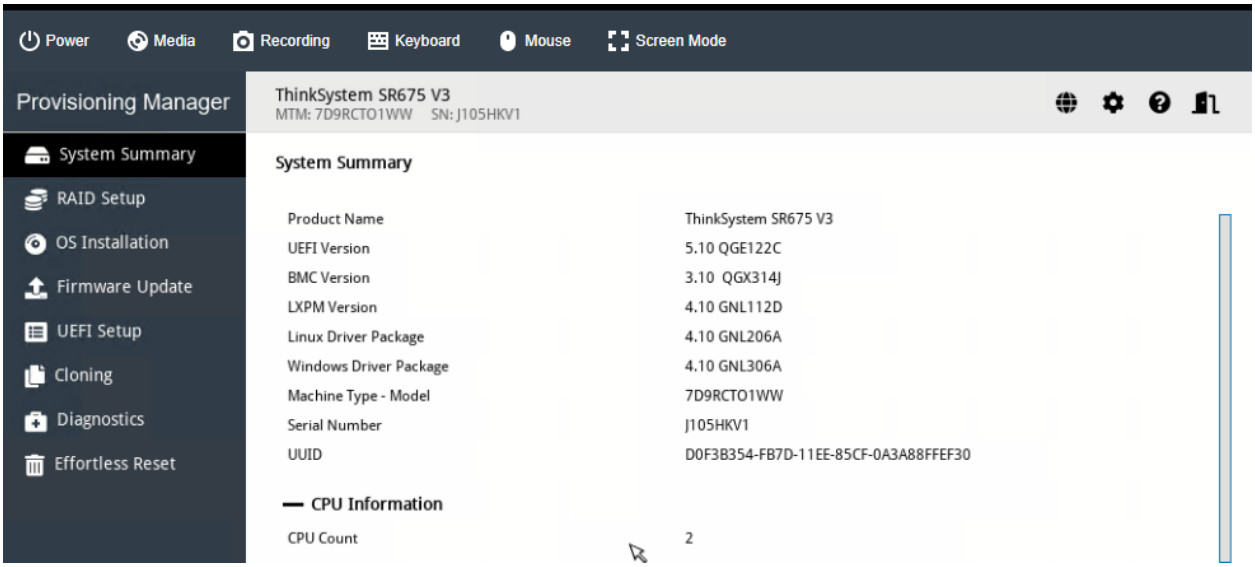

Figure 13) Lenovo XClarity Provisioning Manager

2. Configure RAID for local boot from M.2 NVMe drives

   In the Provisioning Manager, click on “RAID Setup” to select the M.2 NVMe RAID adapter and configure the RAID settings. In the lab, the M.2 drives were configured in RAID 1 as shown in Figure14.

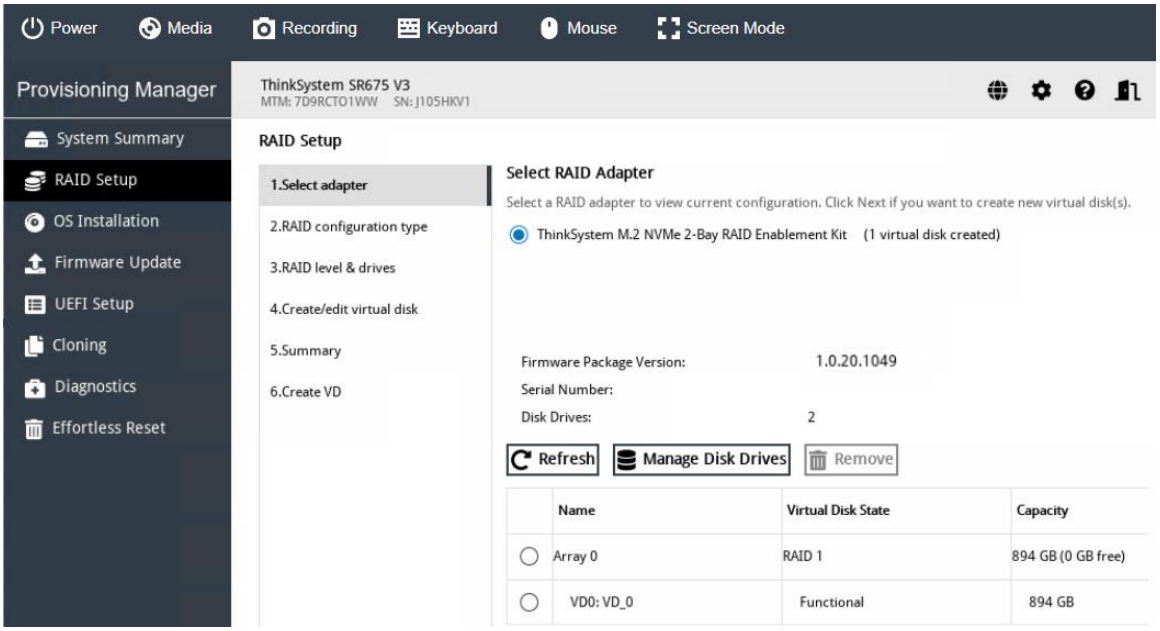

Figure 14) RAID configuration for M.2 NVMe drives

3. Configure BlueField-3 adapters DPU and CPU settings

   In this design, the BlueField-3 adapters are used in NIC mode. For this, you need to set the DPU and CPU settings. In the provisioning manager, click on “**UEFI Setup**” from the left menu bar (Figure 15) and select the BlueField port. Select “**System Settings**” from the left sub-menu and choose the value "Disabled" for DPU setting and “Enable all ports” for Ports Enabled.

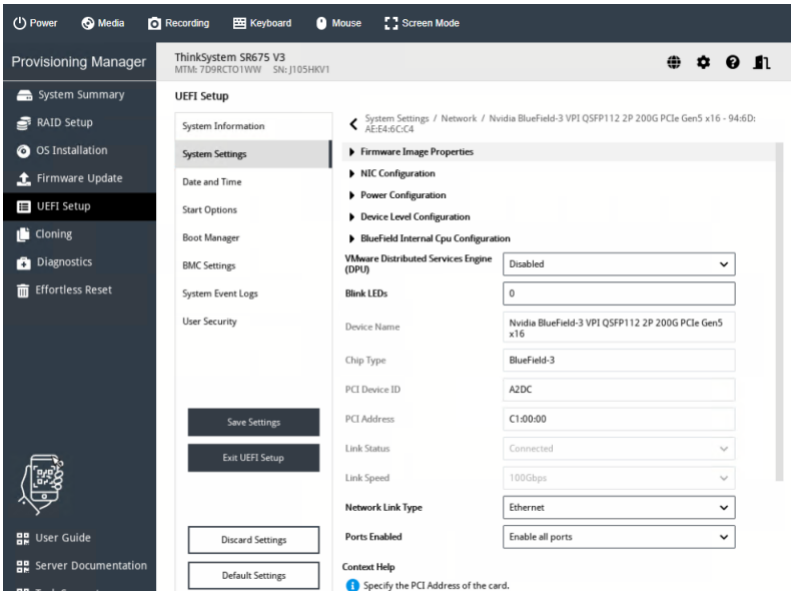

Figure 15) BlueField-3 adapters DPU and CPU settings

Click on “BlueField Internal CPU Configuration” and select the values as shown in Figure 16.

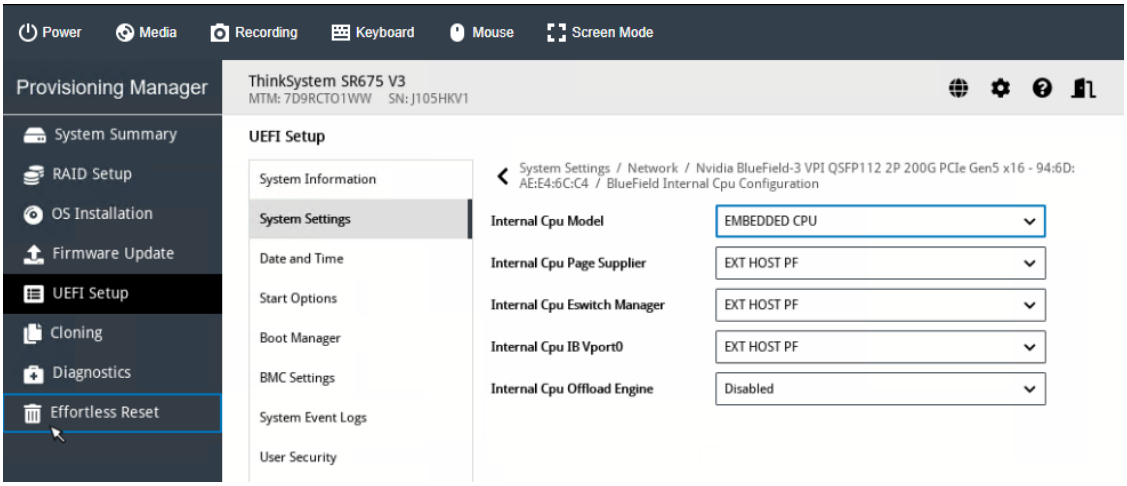

Figure 16) BlueField Internal CPU Configuration

Click on “Save Settings” button to save the configuration

Repeat the procedure on all BlueField-3 ports in use.

4. BlueField-3 and Management NIC configuration

In the validated setup, the BlueField-3 port connects to SN4600 switch port configured in access mode, hence VLAN tagging is not required. Make sure to disable VLAN mode as shown in Figure 17.

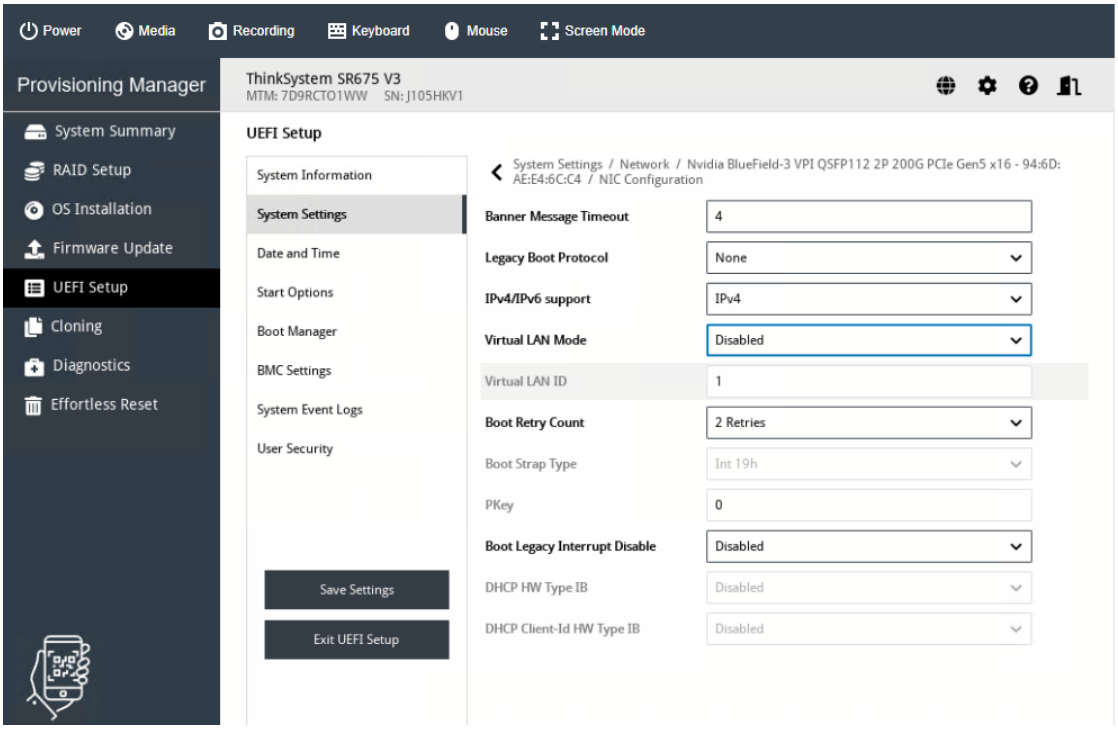

Figure 17) NIC VLAN mode configuration

Click on “Save Settings” button to save the configuration

Repeat the procedure on all BlueField-3 and Management ports in use.

5. Mount virtual media and restart the server.

   From the Provisioning Manager top menu bar, choose “Media” as shown in Figure 18.

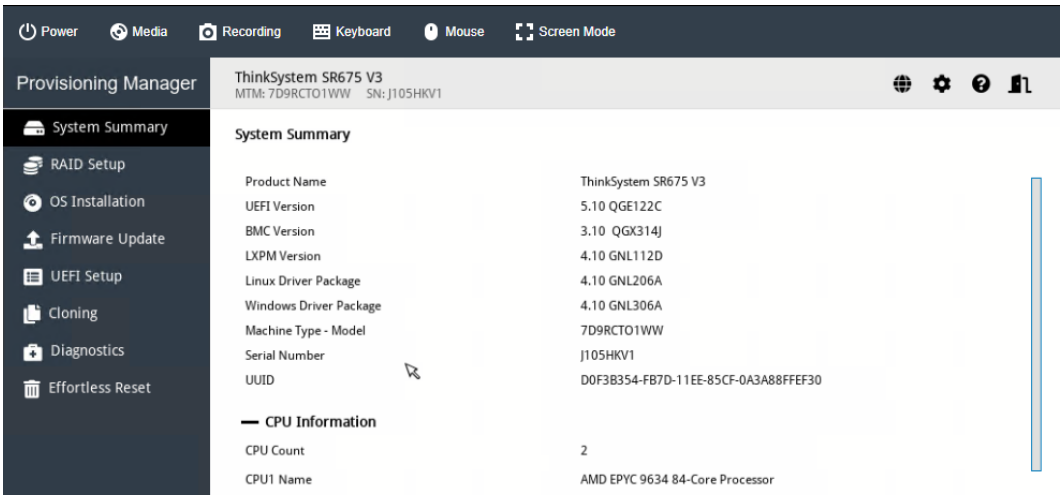

Figure 18) Mounting virtual media in Provisioning Manager

To activate virtual media, click on “Activate” button as shown in Figure 19.

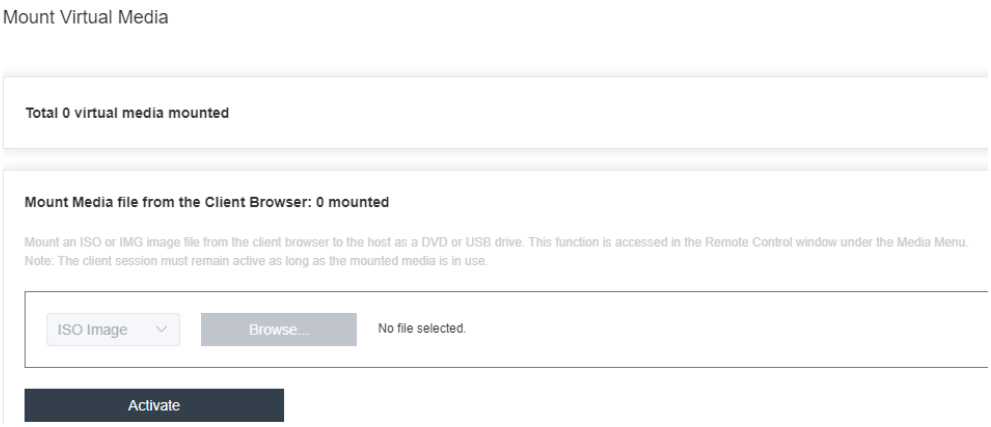

Figure 19) Activate Virtual media

Click the “Browse” button to choose the ISO file, as shown in Figure 20.

Figure 20) Choose media file for activation

Mount the media file by clicking on “Mount all Local media” button, as shown in Figure 21.

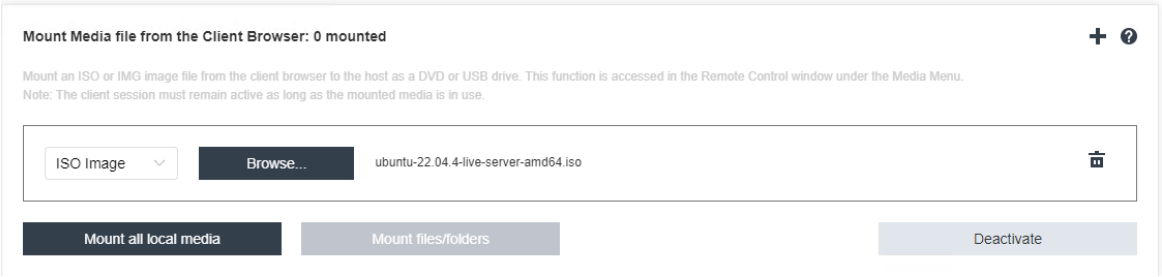

Figure 21) Choose media file for activation

Apply the settings as shown in Figure 22 and restart the server.

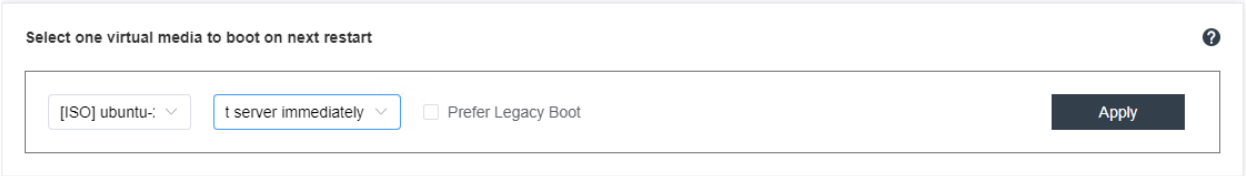

Figure 22) Apply the virtual media settings

The server will boot from the virtual media file. Now you can proceed with the OS installation.

NetApp AFF C800 configuration

See the following section (NetApp Hardware Universe) for planning the physical location of the storage systems:

- Site Preparation
- System Connectivity Requirements
- Circuit Breaker, Power Outlet Balancing, System Cabinet Power Cord Plugs, and Console Pinout Requirements
- AFF Series Systems

NetApp Hardware Universe

The NetApp Hardware Universe (HWU) application provides supported hardware and software components for any specific ONTAP version. It also provides configuration information for all the NetApp storage appliances currently supported by ONTAP software and a table of component compatibilities.

To confirm that the hardware and software components that you would like to use are supported with the version of ONTAP that you plan to install, follow these steps at the NetApp Support site:

1. Access the HWU application to view the System Configuration guides.

2. Click the Products tab to select Platforms menu to view the compatibility between different versions of the ONTAP software and the NetApp storage appliances with your desired specifications.

3. Alternatively, to compare components by storage appliance, click Utilities and select compare Storage Systems.

Controllers

Follow the physical installation procedures for the controllers found here: https://docs.netapp.com/us-en/ontap-systems/index.html.

Disk Shelves

NetApp storage systems support a wide variety of disk shelves and disk drives. The complete list of disk shelves that is supported by the AFF C800 is available at the HWU site.

When using NVMe drive shelves with NetApp storage controllers, refer to: https://docs.netapp.com/us-en/ontap-systems/ns224/hot-add-shelf.html for installation and servicing guidelines.

To setup a new ONTAP cluster, follow the instruction here:

1. Once the cluster is setup, log into the Cluster and verify Storage Failover

```
aipod::> storage failover show
                              Takeover
Node           Partner        Possible State Description
-------------- -------------- -------- -------------------------------------
aipod-01       aipod-02       true     Connected to aipod-02
aipod-02       aipod-01       true     Connected to aipod-01
2 entries were displayed.
```

2 entries were displayed.

Note: Both <st-node01> and <st-node02> must be capable of performing a takeover. Continue with Step 2 if the nodes can perform a takeover.

2. Enable failover on one of the two nodes if it was not completed during the installation:

```
storage failover modify -node <st-node01> -enabled true
```

Note: Enabling failover on one node enables it for both nodes.

3. Verify the HA status for a two-node cluster.

Note: This step is not applicable for clusters with more than two nodes.

```
aipod::> cluster ha show
High-Availability Configured: true
```

4. If HA is not configured use the below commands. Only enable HA mode for two-node clusters.

Note: Do not run this command for clusters with more than two nodes because it causes problems with failover.

```
cluster ha modify -configured true
Do you want to continue? {y|n}: y
```

5. Verify that hardware assist is correctly configured:

```
aipod::> storage failover hwassist show Node
```

6. Set Auto-Revert on Cluster Management Interface

To set the auto-revert parameter on the cluster management interface, follow this step:

```
network interface modify -vserver <clustername> -lif cluster_mgmt_lif_1 -auto-revert true
```

Note: A storage virtual machine (SVM) is referred to as a Vserver or vserver in the GUI and CLI.

7. To zero all spare disks in the cluster, run the following command:

```
disk zerospares
```

Note: Advanced Data Partitioning creates a root partition and two data partitions on each SSD drive in an AFF configuration. Disk auto assign should have assigned one data partition to each node in an HA pair. If a different disk assignment is re-quired, disk auto assignment must be disabled on both nodes in the HA pair by running the disk option modify command. Spare partitions can then be moved from one node to another by running the **`disk removeowner and disk assign`** commands.

8. Set Up Service Processor Network Interface

```
system service-processor network modify –node <st-node01> -address-family IPv4 –enable true –dhcp
none –ip-address <node01-sp-ip> -netmask <node01-sp-mask> -gateway <node01-sp-gateway>

system service-processor network modify –node <st-node02> -address-family IPv4 –enable true –dhcp
none –ip-address <node02-sp-ip> -netmask <node02-sp-mask> -gateway <node02-sp-gateway>
```

Note: The Service Processor IP addresses should be in the same subnet as the node management IP addresses.

An aggregate containing the root volume is created during the ONTAP setup process. To manually create additional aggregates, determine the aggregate name, the node on which to create it, and the number of disks it should contain.

9. To create new aggregates, run the following commands:

```
storage aggregate create -aggregate <aggr1_node01> -node <st-node01> -diskcount <num-disks> -
diskclass solid-state

storage aggregate create -aggregate <aggr1_node02> -node <st-node02> -diskcount <num-disks> -
diskclass solid-state
```

Note: You should have the minimum number of hot spare disks for the recommended hot spare disk partitions for their aggregate.

Note: For all-flash aggregates, you should have a minimum of one hot spare disk or disk partition. For non-flash homogenous aggregates, you should have a minimum of two hot spare disks or disk partitions. For Flash Pool aggregates, you should have a minimum of two hot spare disks or disk partitions for each disk type.

Note: In an AFF configuration with a small number of SSDs, you might want to create an aggregate with all, but one remaining disk (spare) assigned to the controller.

Note: The aggregate cannot be created until disk zeroing completes. Run the **`storage aggregate show`** command to display the aggregate creation status. Do not proceed until both aggr1_node01 and aggr1_node02 are online.

10. Remove Default Broadcast Domains

By default, all network ports are included in separate default broadcast domain. Network ports used for data services (for example e0e, e0f, and so on) should be removed from their default broadcast domain and that broadcast domain should be deleted.

```
network port broadcast-domain delete -broadcast-domain <Default-N> -ipspace Default

network port broadcast-domain show
```

Note: Delete the Default broadcast domains with Network ports.

11. Disable Flow Control on 25/100GbE Data Ports and verify.

```
network port modify -node <st-node01> -port e3a,e3b,e5a,e5b-flowcontrol-admin none
network port modify -node <st-node02> -port e3a,e3b,e5a,e5b -flowcontrol-admin none

network port show -node * -port e3a,e3b,e5a,e5b -fields speed-admin,duplex-admin,flowcontrol-
admin
```

12. Enable Link-layer Discovery Protocol (LLDP).

```
node run * options lldp.enable on
```

13. Enable FIPS Mode on the NetApp ONTAP Cluster (Optional)

NetApp ONTAP is compliant in the Federal Information Processing Standards (FIPS) 140-2 for all SSL connections. When SSL FIPS mode is enabled, SSL communication from NetApp ONTAP to external client or server components outside of NetApp ONTAP will use FIPS compliant crypto for SSL.

```
set -privilege advanced
security config modify -interface SSL -is-fips-enabled true
```

14. Configure Time zone

```
timezone -timezone <timezone>
```

15. Configure Simple Network Management Protocol

Note: If you have enabled FIPS then please look at the following points while configuring SNMP.

- The SNMP users or SNMP traphosts that are non-compliant with FIPS will be deleted automatically. “Configure SNMP traphosts” configuration will be non-compliant with FIPS.
- The SNMPv1 user, SNMPv2c user (After configuring SNMP community) or SNMPv3 user (with none or MD5 as authentication protocol or none or DES as encryption protocol or both) is non-compliant with FIPS.

1. Configure basic SNMP information, such as the location and contact. When polled, this information is visible as the sysLocation and sysContact variables in SNMP.

```
snmp contact <snmp-contact>
snmp location “<snmp-location>”
snmp init 1
options snmp.enable on
```

2. Configure SNMP traps to send to remote hosts, such as an Active IQ Unified Manager server or another fault management system.

Note: This step works when FIPS is disabled. An SNMPv1 traphost or SNMPv3 traphost (configured with an SNMPv3 user non-compliant to FIPS) is non-compliant to FIPS.

```
snmp traphost add <oncommand-um-server-fqdn>
```

3. Configure SNMP community.

Note: This step works when FIPS is disabled. SNMPv1 and SNMPv2c are not supported when cluster FIPS mode is enabled.

```
system snmp community add -type ro -community-name <snmp-community> -vserver <clustername>
```

4. Configure SNMPv3 Access

SNMPv3 offers advanced security by using encryption and passphrases. The SNMPv3 user can run SNMP utilities from the traphost using the authentication and privacy settings that you specify.

Note: When FIPS is enabled, the following are the supported/compliant options for authentication and privacy protocol:

- Authentication Protocol: sha, sha2-256
- Privacy protocol: aes128

```
security login create -user-or-group-name <<snmp-v3-usr>> -application snmp -authentication-
method usm

Enter the authoritative entity's EngineID [local EngineID]:
Which authentication protocol do you want to choose (none, md5, sha, sha2-256) [none]: <<snmp-v3-
auth-proto>>
```

```
Enter the authentication protocol password (minimum 8 characters long):
Enter the authentication protocol password again:
Which privacy protocol do you want to choose (none, des, aes128) [none]: <<snmpv3-priv-proto>>
Enter privacy protocol password (minimum 8 characters long):
Enter privacy protocol password again:
```

Note: Refer to the SNMP Configuration Express Guide for additional information when configuring SNMPv3 security users.

16. Create Management Broadcast Domain

Note: If the management interfaces are required to be on a separate VLAN, create a new broadcast domain for those interfaces by running the following command:

```
network port broadcast-domain create -broadcast-domain IB-MGMT -mtu 1500
```

17. Create NFS Broadcast Domain

- Create standard NFS Broadcast Domain

Note: To create an NFS data broadcast domain with a maximum transmission unit (MTU) of 9000, run the following commands:

```
network port broadcast-domain create -broadcast-domain Infra-NFS -mtu 9000
```

- Create Broadcast Domain for NFS over RDMA

```
network port broadcast-domain create -broadcast-domain nfs_rdma -mtu 9000
```

18. Create 2 Default Broadcast Domains and Add ifgroups a0a interface on each node to Default Broadcast Domain (applicable for 2-node cluster only)

```
broadcast-domain create -broadcast-domain Default-1 -mtu 9000 -ipspace Default
broadcast-domain create -broadcast-domain Default-2 -mtu 9000 -ipspace Default

broadcast-domain add-ports -broadcast-domain Default-1 -ports aipod-01:a0a
  (network port broadcast-domain add-ports)

broadcast-domain add-ports -broadcast-domain Default-2 -ports aipod-02:a0a
  (network port broadcast-domain add-ports)
```

19. Create Interface Groups for the data interfaces

```
network port ifgrp create -node <st-node01> -ifgrp a0a -distr-func port -mode multimode_lacp
network port ifgrp add-port -node <st-node01> -ifgrp a0a -port e3a
network port ifgrp add-port -node <st-node01> -ifgrp a0a -port e3b

network port ifgrp create -node <st-node02> -ifgrp a0a -distr-func port -mode multimode_lacp
network port ifgrp add-port -node <st-node02> -ifgrp a0a -port e3a
network port ifgrp add-port -node <st-node02> -ifgrp a0a -port e3b
```

To Verify:

```
aipod::> network port ifgrp show
         Port       Distribution                 Active
Node     IfGrp      Function     MAC Address     Ports   Ports
-------- ---------- ------------ ----------------- ------- -------------------
aipod-01
         a0a        port         d2:39:ea:b1:ed:7a full    e3a, e3b
aipod-02
         a0a        port         d2:39:ea:b1:f0:72 full    e3a, e3b
2 entries were displayed.
Create the management VLAN ports and add them to the management broadcast domain:
network port vlan create –node <st-node01> -vlan-name a0a-<ib-mgmt-vlan-id>
network port vlan create –node <st-node02> -vlan-name a0a-<ib-mgmt-vlan-id>
```

```
network port broadcast-domain add-ports -broadcast-domain IB-MGMT -ports <st-node01>:a0a-<ib-
mgmt-vlan-id>,<st-node02>:a0a-<ib-mgmt-vlan-id>

network port vlan show
```

20. Create the NFS VLAN ports and add them to the Infra-NFS broadcast domain:

```
network port vlan create –node <st-node01> -vlan-name a0a-<infra-nfs-vlan-id>
network port vlan create –node <st-node02> -vlan-name a0a-<infra-nfs-vlan-id>

network port broadcast-domain add-ports -broadcast-domain Infra-NFS -ports <st-node01>:a0a-
<infra-nfs-vlan-id>,<st-node02>:a0a-<infra-nfs-vlan-id>
```

21. Create SVM (Storage Virtual Machine)

Note: The SVM is used to configure NFS services for storage access by NetApp Astra Trident.

```
vserver create –vserver aipod-svm
```

22. Add the required data protocols to the SVM:

```
vserver add-protocols -protocols nfs –vserver aipod-svm
```

23. Remove the unused data protocols from the SVM:

```
vserver remove-protocols –vserver aipod-svm -protocols cifs,fcp,iscsi
```

24. Add the two data aggregates to the aipod-svm vserver.

```
vserver modify -vserver aipod-svm -aggr-list aggr1_aipod_01,aggr1_aipod_02
```

25. Enable and run the NFS protocol

```
vserver nfs create -vserver aipod-svm -udp disabled -v3 enabled -v4.1 enabled -vstorage enabled
```

Note: If the NFS license was not installed during the cluster configuration, make sure to install the license before starting the NFS service.

26. Vserver Protocol Verification

```
vserver show-protocols -vserver aipod-svm
```

27. Create a load-sharing mirror volume of the “aipod-svm” SVM root volume on the node that does not have the Root Volume:

```
volume show -vserver aipod-svm  # Note down the aggregate and node for the root volume

volume create –volume aipod_svm_root_lsm01 –aggregate aggr1_aipod_02 –size 1GB –type DP
```

28. Create a job schedule to update the root volume mirror relationships every 15 minutes:

```
job schedule interval create -name lsm-15min -minutes 15
```

29. Create mirroring relationships:

```
snapmirror create -source-path aipod-svm:aipod_svm_root -destination-path aipod-
svm:aipod_svm_root_lsm01 -type LS -vserver aipod-svm -schedule lsm-15min
```

30. Initialize the mirroring relationship:

```
snapmirror initialize-ls-set -source-path aipod-svm:aipod_svm_root
[Job 1781] Job is queued: "snapmirror initialize-ls-set" for source "aipod://aipod-
svm/aipod_svm_root".
```

To verify:

```
snapmirror show -vserver aipod-svm
                                                                    Progress
Source            Destination Mirror  Relationship   Total             Last
Path        Type  Path        State   Status         Progress  Healthy Updated
----------- ---- ------------ ------- -------------- --------- ------- --------
aipod://aipod-svm/aipod_svm_root
            LS   aipod://aipod-svm/aipod_svm_root_lsm01
                              Snapmirrored
                                      Idle           -         true    -
```

31. Configure HTTPS Access

    1. To configure secure access to the storage controller, follow these steps:

```
set -privilege diag
Do you want to continue? {y|n}: y
```

    2. A self-signed certificate is already in place. Verify the certificate and obtain parameters (for example, the <serial-number>) by running the following command:

```
security certificate show
```

    3. For each SVM shown, the certificate common name should match the DNS fully qualified domain name (FQDN) of the SVM. Delete the two default certificates and replace them with either self-signed certificates or certificates from a certificate authority (CA). To delete the default certificates, run the following commands:

```
security certificate delete -vserver aipod-svm -common-name aipod-svm -ca aipod-svm -type server
-serial <serial-number>
```

Note: Deleting expired certificates before creating new certificates is best practice. Run the **security certificate delete** command to delete the expired certificates. In the following command, use TAB completion to select and delete each default certificate.

    4. To generate and install self-signed certificates, run the following commands as one-time commands. Generate a server certificate for the AIPOD-SVM and the cluster SVM. Use TAB completion to aid in the completion of these commands.

```
security certificate create -common-name <cert-common-name> -type  server -size 2048 -country
<cert-country> -state <cert-state> -locality <cert-locality> -organization <cert-org> -unit
<cert-unit> -email-addr <cert-email> -expire-days <cert-days> -protocol SSL -hash-function SHA256
-vserver aipod-svm
```

    5. Obtain the values for the parameters required in step 5 (<cert-ca> and <cert-serial>)

```
security certificate show
```

    6. Enable each certificate that was just created by using the –server-enabled true and –client-enabled false parameters. Use TAB completion to aid in the completion of these commands.

```
security ssl modify -vserver <clustername> -server-enabled true -client-enabled false -ca <cert-
ca> -serial <cert-serial> -common-name <cert-common-name>
```

    7. Disable HTTP cluster management access.

```
network interface service-policy remove-service -vserver <clustername> -policy default-management
-service management-http
```

Note: It is normal for some of these commands to return an error message stating that the entry does not exist.

8. Change back to the normal admin privilege level and verify that the system logs are available in a web browser.

```
set -privilege admin

https://<node01-mgmt-ip>/spi
https://<node02-mgmt-ip>/spi
```

32. Set password for SVM vsadmin user and unlock the user

```
security login password -username vsadmin -vserver aipod-svm
Enter a new password:  <password>
Enter it again:  <password>

security login unlock -username vsadmin -vserver aipod-svm
```

33. Configure login banner for the SVM

```
security login banner modify -vserver aipod-svm -message "This AIPOD-SVM is reserved for
authorized users only!"
```

Note: If the login banner for the SVM is not configured, users will observe a warning in AIQUM stating "Login Banner Disabled."

34. Configure Export Policy Rule

Note: This step is crucial when using NFS storage driver in Astra Trident since this SVM is added as a Trident Backend.

To configure NFS on the SVM, follow these steps:

1. Create a new rule for the infrastructure NFS subnet in the default export policy:

```
vserver export-policy rule create -vserver aipod-svm -policyname default -ruleindex 1 -protocol
nfs -clientmatch <infra-nfs-subnet-cidr> -rorule any -rwrule any -superuser sys -allow-suid true
-anon 65534
```

Note: For more information on configuring NFS Export Policy for Trident, go to: https://docs.netapp.com/us-en/trident/trident-use/ontap-nas-prep.html#requirements.

2. Assign the export policy to the SVM (aipod-svm) root volume:

```
volume modify -vserver aipod-sm -volume aipod_svm_root -policy default
```

35. Create FlexVol® Volumes

The following information is required to create a NetApp FlexVol® volume:

- The volume name
- The volume size
- The aggregate on which the volume exists

Run the following command to create a volume for storing SVM audit log configuration:

```
volume create -vserver aipod-svm -volume audit_log -aggregate <aggr1_node01> -size 50GB -state
online -
policy default -junction-path /audit_log -space-guarantee none -percent-snapshot-space 0

snapmirror update-ls-set -source-path aipod-svm:aipod_svm_root  # Update set of load-sharing
mirrors
```

36. Create standard NFS LIFs.

```
network interface create -vserver aipod-svm -lif nfs-lif-01 -service-policy default-data-files -
home-node <st-node01> -home-port a0a-< nfs-vlan-id> -address <node01-nfs-lif-01-ip> -netmask
```

```
<node01-nfs-lif-01-mask> -status-admin up -failover-policy broadcast-domain-wide -auto-revert
true

network interface create -vserver aipod-svm -lif nfs-lif-02 -service-policy default-data-files -
home-node <st-node02> -home-port a0a-< nfs-vlan-id> -address <node02-nfs-lif-02-ip> -netmask
<node02-nfs-lif-02-mask>> -status-admin up -failover-policy broadcast-domain-wide -auto-revert
true
```

37. Create NFS over RDMA LIFs.

Note: Interface groups are not supported with NFS over RDMA. You may use ONTAP system manager to check the RDMA protocol status before configuring NFS over RDMA LIFs.

Figure 23 shows the RDMA protocol status of the ports used for validation.

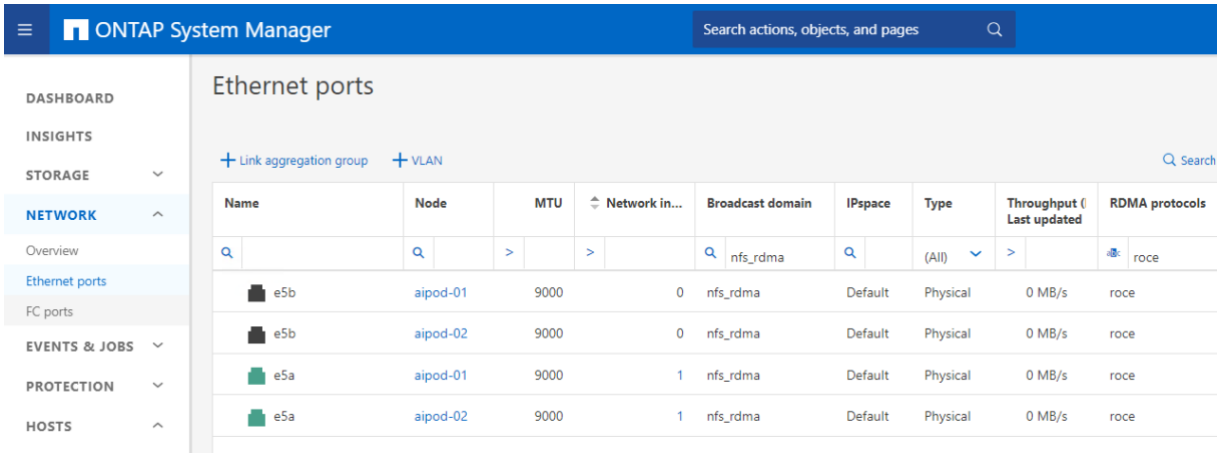

Figure 23) RDMA protocol Status

```
network interface create -vserver aipod-svm -lif nfs-rdma-n1-01 -service-policy default-data-
files home-node <st-node01> -home-port e5a -address <node01-nfs-lif-01-ip> -netmask <node01-nfs-
lif-01-mask> -status-admin up -auto-revert true -rdma-protocols roce


network interface create -vserver aipod-svm -lif nfs-rdma-n2-01 -service-policy default-data-
files home-node <st-node02> -home-port e5a -address <node01-nfs-lif-02-ip> -netmask <node01-nfs-
lif-02-mask> -status-admin up -auto-revert true -rdma-protocols roce
```

Note: The *-rdma-protocols* parameter accepts a list, which is by default empty. When roce is added as a value, the LIF can only be configured on ports supporting RoCE offload, affecting both LIF migration and failover.

To verify, run the following commands:

```
network interface show -vserver aipod-svm -service-policy default-data-files
            Logical    Status     Network            Current       Current Is
Vserver     Interface  Admin/Oper Address/Mask       Node          Port    Home
----------- ---------- ---------- ------------------ ------------- ------- ----
AIPOD-SVM
            nfs_rdma_n1_01 up/up  172.22.63.131/24   aipod-01      e5a     true
            nfs_rdma_n2_01 up/up  172.22.63.132/24   aipod-02      e5a     true
```

2 entries were displayed.

38. Create SVM Management LIF (Add Infrastructure SVM Administrator)

1. To add the SVM administrator and SVM administration LIF in the in-band management network, follow these steps:

```
network interface create –vserver aipod-svm –lif svm-mgmt –home-node <st-node02> -home-port a0a-
<ib-mgmt-vlan-id> -address <svm-mgmt-ip> -netmask <svm-mgmt-mask> -status-admin up -auto-revert
true -service-policy default-management –failover-policy broadcast-domain-wide
```

2. Create a default route that enables the SVM management interface to reach the outside world.

```
network route create –vserver aipod-svm -destination 0.0.0.0/0 –gateway <svm-mgmt-gateway>
```

3. To verify, run the following commands:

```
network route show -vserver aipod-svm
Vserver             Destination     Gateway         Metric
------------------- --------------- --------------- ------
aipod-svm
                    0.0.0.0/0       172.22.62.1     20
```

4. Add route to reach the corresponding control/app network:

```
net route show -vserver aipod-svm
  (network route show)
Vserver             Destination     Gateway         Metric
------------------- --------------- --------------- ------
aipod-svm
                    0.0.0.0/0       172.22.62.1     20
```

Note: A cluster serves data through at least one and possibly several SVMs. With these steps, you've created a single data SVM. You can create additional SVMs depending on your requirement.

39. To configure AutoSupport™, run the following command:

```
system node autosupport modify -state enable –mail-hosts <mailhost> -from <from-address> -
transport https -support enable -to <storage-admin-email>
```

40. To configure DNS for the aipod-svm, run the following command:

```
dns create -vserver <vserver-name> -domains <dns-domain> -nameserver <dns-servers>
```

Example:

```
dns create -vserver aipod-svm -domains fpmc.sa -name-servers 10.61.176.251,10.61.176.252
```

41. To create auditing configuration for the SVM, run the following command:

```
vserver audit create -vserver aipod-svm -destination /audit_log
```

42. Run the following command to enable audit logging for the SVM:

```
vserver audit enable -vserver aipod-svm
```

Note: It is recommended that you enable audit logging so you can capture and manage important support and availability information. Before you can enable auditing on the SVM, the SVM's auditing configuration must already exist.

43. To test the Auto Support configuration by sending a message from all nodes of the cluster, run the following command:

```
autosupport invoke -node * -type all -message "ONTAP storage configuration for AIPOD is
completed"
```

Now the setup of the NetApp storage is completed. Since Lenovo servers are booted with local disks, storage configuration for boot is not required. The configuration required for NetApp Astra Trident will be done later.

Bare metal Ubuntu configuration

For this deployment Ubuntu 22.04 was chosen and customers could use a later version of the operating system ensuring the compatibility between various software, drivers and hardware are being considered for the deployment at that time.

1. Install Ubuntu 22.04 on all the control and app nodes
2. Install DOCA host drivers on all the worker nodes for Ubuntu 22.04. For this solution the following steps were followed.

```
wget https://www.mellanox.com/downloads/DOCA/DOCA_v2.7.0/host/doca-host_2.7.0-204000-24.04-ubuntu2204_amd64.deb
dpkg -i doca-host_2.7.0-204000-24.04-ubuntu2204_amd64.deb
apt-get update
apt-get -y install doca-ofed
```

3. Install the DOCA firmware bundle for the BlueField-3 adapters and following steps were followed for this solution

```
Make sure to install host drivers - DOCA-Host drivers
Run the following command:
bfb-install --bfb bf-fwbundle-2.7.0-31_24.04-prod.bfb --rshim rshim0
```

4. Update the firmware on the BlueField-3 adapter to 32.41.10 and is available for the Lenovo OPN of BlueField-3 and can be installed as follows.

Download the file to all the worker nodes and unzip it and run as an executable

```
./mlxfwmanager_LES_24A_ES_OFED-24.04-0_build1
reboot
```

Interface configuration on the worker node to provide connectivity to management, east-west (GPU) and NFS over RDMA networks.

```
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens1f0np0:       # For East-West Traffic-Port 1
      addresses:
      - 172.22.100.13/24
      mtu: 9000
      nameservers:
        addresses: []
        search: []
---
    ens20f0np0:       # For East-West Traffic-Port 2
      addresses:
      - 172.22.100.11/24
      mtu: 9000
      nameservers:
        addresses: []
        search: []
---
    ens27f0:          # For Management Traffic
      addresses:
      - 10.61.177.104/24
      nameservers:
        addresses:
        - 10.61.176.251
        search:
        - fpmc.sa
      routes:
      - to: default
        via: 10.61.177.1
```

```
---
    ens2f0np0:       # For NFSoRDMA Traffic
      addresses:
      - 172.22.63.11/24
      mtu: 9000
      nameservers:
        addresses: []
```

Upstream K8s deployment – Highly Available Cluster

The steps below will cover how to deploy the upstream K8s cluster and configure to run the AI framework. Optionally K8s deployment can be automated using kubespray. As kubespray will need ansible, a control resource (server or a virtual machine) will be a requirement.

1. Ensure that the system is up to date.

```
sudo apt update
sudo apt upgrade
```

2. For the kubelet to work properly swap must be disabled for all the control and app nodes

```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

3. Load the two kernel modules in the current running environment and configure them to load on boot.

```
sudo tee /etc/modules-load.d/containerd.conf <<EOF
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
```

4. Configure the kernel parameters into sysctl to persist across system reboots

```
sudo tee /etc/sysctl.d/kubernetes.conf <<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

5. Re-load sysctl for the new changes

```
sudo sysctl –system
```

6. Install containerd runtime for all the nodes including control and app.

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
```

7. Enable the Docker repository

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o
/etc/apt/trusted.gpg.d/docker.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release
-cs) stable"
```

8. Install containerd after updating the package

```
sudo apt update
sudo apt install -y containerd.io
```

9. Configure containerd to start using systemd as cgroup

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
```

10. Restart and enable the containerd service

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

11. Add Apt Repository for K8s for all nodes including control and app

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o
/etc/apt/trusted.gpg.d/kubernetes-xenial.gpg
sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

12. Install Kubectl, Kubeadm, and Kubelet for all nodes including control and app

```
sudo apt update
sudo apt install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

13. SSH to the node which will function as the load balancer and execute the following commands to install HAProxy:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install haproxy
```

1. Edit haproxy.cfg to connect it to the control nodes by setting the correct values for <kube-balancer-ip> and <kube-master-ip> and add an extra entry for each additional control node. Change the ip address which fits your installation.

```
sudo nano /etc/haproxy/haproxy.cfgglobal
       ....
defaults
       ....
frontend kubernetes
    bind 10.61.177.106:6443
    option tcplog
    mode tcp
    default_backend kubernetes-master-nodes
frontend http_front
    mode http
    bind 10.61.177.106:80
    default_backend http_back
frontend https_front
    mode http
    bind 10.61.177.106:443
    default_backend https_back
backend kubernetes-master-nodes
    mode tcp
    balance roundrobin
    option tcp-check
    server aipod-master-01 10.61.177.101:6443 check fall 3 rise 2
    server aipod-master-02 10.61.177.102:6443 check fall 3 rise 2
    server aipod-master-03 10.61.177.103:6443 check fall 3 rise 2
backend http_back
    mode http
    server aipod-master-01 10.61.177.101:6443 check fall 3 rise 2
    server aipod-master-02 10.61.177.102:6443 check fall 3 rise 2
    server aipod-master-03 10.61.177.103:6443 check fall 3 rise 2
backend https_back
    mode http
    server aipod-master-01 10.61.177.101:6443 check fall 3 rise 2
    server aipod-master-02 10.61.177.102:6443 check fall 3 rise 2
    server aipod-master-03 10.61.177.103:6443 check fall 3 rise 2
```

2. Restart HA proxy

```
sudo systemctl restart haproxy
```

3. Update the apt package index and install packages needed to use the Kubernetes apt repository.

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
```

4. Download the public signing key for the Kubernetes package repositories. The same signing key is used for all repositories so you can disregard the version in the URL.

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

5. Add the appropriate Kubernetes apt repository.

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

6. Update the apt package index, install kubelet, kubeadm and kubectl, and pin their version.

```
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

7. Initialize the first control plane

```
sudo kubeadm init --control-plane-endpoint "10.61.177.106:6443"

Your Kubernetes control-plane has initialized successfully!
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: stable
apiServer:
certSANs:
- "10.61.177.106"
controlPlaneEndpoint: "10.61.177.106:6443"

sudo kubeadm init --control-plane-endpoint "10.61.177.106:6443"
***********
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of control-plane nodes by copying certificate authorities
and service account keys on each node and then running the following as root:

  kubeadm join 10.61.177.106:6443 --token inivnq.6x79z9oh39xih6o2 \
        --discovery-token-ca-cert-hash
sha256:25d1803e1e293cecff51a2810d6f09a0c81d29d1302d545a3ced7575a904e182 \
        --control-plane

Then you can join any number of worker nodes by running the following on each as root:
```

```
kubeadm join 10.61.177.106:6443 --token inivnq.6x79z9oh39xih6o2 \
        --discovery-token-ca-cert-hash
sha256:25d1803e1e293cecff51a2810d6f09a0c81d29d1302d545a3ced7575a904e182
```

8. Execute the below command to start using the cluster.

```
 mkdir -p $HOME/.kube
 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
 sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

9. For the networking we are using calico as the pod-network.

```
kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml
```

10. Manually distribute the certificate by running the below script on the first control plane node.

```
USER=aipod # customizable
# Set the control_plane_ips to all other master node ips or hostnames
CONTROL_PLANE_IPS="10.61.177.102 10.61.177.103"
for host in ${CONTROL_PLANE_IPS}; do
    scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:ca.crt
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:ca.key
    scp /etc/kubernetes/admin.conf "${USER}"@$host:
done
```

Note: Copy only the certificates in the above list. kubeadm will take care of generating the rest of the certificates with the required SANs for the joining control-plane instances.

11. Then on each joining control plane node run the below script to move the certificates to /etc/kubernetes/pki directory.

```
USER=aipod # customizable
mkdir -p /etc/kubernetes/pki/etcd
mv /home/${USER}/ca.crt /etc/kubernetes/pki/
mv /home/${USER}/ca.key /etc/kubernetes/pki/
mv /home/${USER}/sa.pub /etc/kubernetes/pki/
mv /home/${USER}/sa.key /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.crt /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.key /etc/kubernetes/pki/
mv /home/${USER}/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
# Skip the next line if you are using external etcd
mv /home/${USER}/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
```

12. Join the remaining control nodes. Change the IP's/Keys based on your installation

```
kubeadm join 10.61.177.106:6443 --token inivnq.6x79z9oh39xih6o2 --discovery-token-ca-cert-hash
sha256:25d1803e1e293cecff51a2810d6f09a0c81d29d1302d545a3ced7575a904e182 --control-plane
```

13. Once all the control-plane are added, add any number of app nodes, in this case two by running the following on each control node as root user

```
kubeadm join 10.61.177.106:6443 --token inivnq.6x79z9oh39xih6o2 --discovery-token-ca-cert-hash
sha256:25d1803e1e293cecff51a2810d6f09a0c81d29d1302d545a3ced7575a904e182
```

Installing NetApp Astra Trident

Astra Trident is an open-source and fully supported storage orchestrator for containers and K8s distributions, including Red Hat OpenShift. Trident works with the entire NetApp storage portfolio,

including the NetApp ONTAP and Element storage systems, and it also supports NFS, iSCSI and NVMe-TCP connections. Trident accelerates the DevOps workflow by allowing end users to provision and manage storage from their NetApp storage systems without requiring intervention from a storage administrator and can installed as follows.

1. Download Astra Trident software from GitHub and untar the trident-installer-24.06.0.tar.gz file to obtain the trident-installer folder.

```
wget https://github.com/NetApp/trident/releases/download/v24.06.0/trident-installer-24.06.0.tar.gz
Saving to: ‘trident-installer-24.06.0.tar.gz’

tar -xvzf trident-installer-24.06.0.tar.gz
cd trident-installer
```

2. Create a Trident namespace.

```
kubectl create ns trident
```

3. Start the installation using helm

```
cd helm

helm install trident trident-operator-100.2406.0.tgz-n trident

#######
NAME: trident
---
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident' namespace.
Please note that there must be only one instance of Trident (and trident-operator) in a
Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.trident.netapp.io created
```

4. Verify the operator and associated elements have been deployed correctly.

```
kubectl get all -n trident
NAME                                  READY   STATUS    RESTARTS        AGE
pod/trident-controller-5fd6fdc7f9-v8bsw   6/6     Running   0               3m28s
pod/trident-node-linux-2gdl7              2/2     Running   0               3m28s
pod/trident-node-linux-4rrvx              2/2     Running   0               3m28s
pod/trident-node-linux-fcz8x              2/2     Running   0               3m28s
pod/trident-node-linux-gf7cz              2/2     Running   0               3m28s
pod/trident-node-linux-tsppt              2/2     Running   0               3m28s
pod/trident-operator-545869857c-z9sp5     1/1     Running   0               4m16s

NAME                  TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)              AGE
service/trident-csi   ClusterIP   10.97.243.79   <none>        34571/TCP,9220/TCP   3m34s

NAME                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
```

```
daemonset.apps/trident-node-linux   5         5         5       5            5           <none>
3m28s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-controller   1/1     1            1           3m28s
deployment.apps/trident-operator     1/1     1            1           4m16s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-controller-5fd6fdc7f9   1         1         1       3m28s
replicaset.apps/trident-operator-545869857c     1         1         1       4m16s
```

Note: If the Astra Trident deployment fails and does not bring up the pods to Running state, use the **`tridentctl logs -l all -n trident`** command for debugging.

5. Verify the Trident server and client version.

```
tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.06.0        | 24.06.0        |
+----------------+----------------+
```

Note: Before configuring the backend that Trident needs to use for user apps, go to: https://docs.netapp.com/us-en/trident/trident-reference/objects.html#kubernetes-customresourcedefinition-objects to understand the storage environment parameters and its usage in Trident.

6. Now configure Trident backend for NFS using the json file.

```
# cat << EOF > ontapnas-backend.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "aipod-nfs",
    "managementLIF": "172.22.61.25 ",
    "dataLIF": "172.22.63.31",
    "svm": " aipod-svm ",
    "username": "admin",
    "password": "#########"
}
EOF
```

7. Run the following command to create

```
tridentctl create backend -f ontapnas-backend.json -n trident
+-----------+----------------+--------------------------------------+--------+------------+------
---+
|   NAME    | STORAGE DRIVER |                 UUID                 | STATE  | USER-STATE |
VOLUMES |
+-----------+----------------+--------------------------------------+--------+------------+------
---+
| aipod-nfs | ontap-nas      | 949b84ae-108c-4174-9ade-6e6ba754dfbb | online | normal     |
0 |
+-----------+----------------+--------------------------------------+--------+------------+------
```

8. Backend definition for ONTAP FlexGroup driver.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "backendName": " aipod-flexgrp-backend",
    "managementLIF": "172.22.61.25",
    "dataLIF": " 172.22.63.32",
    "svm": " aipod-svm ",
    "username": "admin",
    "password": "##########!",
    "defaults": {
```

```
        "spaceReserve": "volume",
        "exportPolicy": "default",
        "snapshotPolicy": "default",
        "snapshotReserve": "10"
    }
}
```

9. Create the FlexGroup backend

```
tridentctl create backend -f aipod-flexgrp-backend.json -n trident
+------------------------+--------------------+--------------------------------------+---------+-
----------+---------+
|          NAME          |   STORAGE DRIVER   |                 UUID                 | STATE   |
USER-STATE | VOLUMES |
+------------------------+--------------------+--------------------------------------+---------+-
----------+---------+
| aipod-flexgrp-backend  | ontap-nas-flexgroup | 154f4712-5254-4789-9a49-382abe765b12 | online |
normal     |       0 |
+------------------------+--------------------+--------------------------------------+---------+-
----------+---------+
```

10. Create storage class for FlexGroup enabled Trident Backend. This storage class will utilize NFS over RDMA.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-flexgroups-rdma
provisioner: csi.trident.netapp.io
mountOptions: ["vers=4.1", "proto=rdma", "max_connect=16", "rsize=262144", "wsize=262144"]
parameters:
  backendType: "ontap-nas-flexgroup"
  storagePools: "aipod-flexgroups"
reclaimPolicy: Retain
```

Note: The following example Storage class uses a maximum transfer size of 262144. To use this maximum transfer size, you must configure the maximum transfer size on your ONTAP system accordingly. Refer to the ONTAP documentation for details.

11. Create storage class for NFS to the FlexVol enabled Trident backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: aipod-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true
```

NVIDIA GPU Operator

Prerequisites-

- Helm is required for installing GPU operator
- All worker nodes or node groups to run GPU workloads in the Kubernetes cluster must run the same operating system version using the NVIDIA GPU Driver container. Alternatively, if you pre-install the NVIDIA GPU Driver on the nodes, then you can run different operating systems.
- Nodes must be configured with a container engine such CRI-O or containerd.
- If the cluster uses Pod Security Admission (PSA) to restrict the behavior of pods, label the namespace for the Operator to set the enforcement policy to privileged:

```
kubectl create ns gpu-operator
kubectl label --overwrite ns gpu-operator pod-security.kubernetes.io/enforce=privileged
```

- Node Feature Discovery (NFD) is a dependency for the Operator on each node. By default, NFD control and app are automatically deployed by the Operator. If NFD is already running in the cluster, then you must disable deploying NFD when you install the Operator.

```
kubectl get nodes -o json | jq '.items[].metadata.labels | keys |
any(startswith("feature.node.kubernetes.io"))'
```

Note: The output should be **false**.

Procedure

1. Add the NVIDIA Helm repository and update the repo.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia \
    && helm repo update

"nvidia" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "nvidia" chart repository
Update Complete. ⎈Happy Helming!⎈
```

2. Install the Operator with the configuration below.

```
helm install gpu-operator -n gpu-operator --create-namespace \
  nvidia/gpu-operator $HELM_OPTIONS \
    --set toolkit.env[0].name=CONTAINERD_CONFIG \
    --set toolkit.env[0].value=/etc/containerd/config.toml \
    --set toolkit.env[1].name=CONTAINERD_SOCKET \
    --set toolkit.env[1].value=/run/containerd/containerd.sock \
    --set toolkit.env[2].name=CONTAINERD_RUNTIME_CLASS \
    --set toolkit.env[2].value=nvidia \
    --set toolkit.env[3].name=CONTAINERD_SET_AS_DEFAULT \
    --set-string toolkit.env[3].value=true \
    --set driver.rdma.enabled=true \
    --set driver.rdma.useHostMofed=true \
```

Note: MOFED drivers were installed directly on hosts. To know more about specifying configuration option for containerd, refer here.

3. Verify the NVIDIA GPU Operator, NVIDIA GPU Driver, and associated elements have been deployed correctly.

```
kubectl get all -n gpu-operator
NAME                                                          READY   STATUS      RESTARTS
AGE
pod/gpu-feature-discovery-24zrt                               1/1     Running     0
2m51s
pod/gpu-feature-discovery-6bg8b                               1/1     Running     0
2m51s
pod/gpu-operator-7bbf8bb6b7-zrqn6                             1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-gc-79d6d968bb-tjgnh   1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-master-6d9f8d497c-9ggkl   1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-worker-5hlh7          1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-worker-9529v          1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-worker-g2wvw          1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-worker-jz99z          1/1     Running     0
2m57s
pod/gpu-operator-node-feature-discovery-worker-mhsnf          1/1     Running     0
2m57s
```

```
pod/nvidia-container-toolkit-daemonset-bgpbj                      1/1     Running     0
2m51s
pod/nvidia-container-toolkit-daemonset-jsksg                      1/1     Running     0
2m51s
pod/nvidia-cuda-validator-5hzjn                                   0/1     Completed   0
59s
pod/nvidia-cuda-validator-9g8bw                                   0/1     Completed   0
54s
pod/nvidia-dcgm-exporter-cgfqj                                    1/1     Running     0
2m51s
pod/nvidia-dcgm-exporter-sbwdn                                    1/1     Running     0
2m51s
pod/nvidia-device-plugin-daemonset-dnwtn                          1/1     Running     0
2m51s
pod/nvidia-device-plugin-daemonset-wrfgc                          1/1     Running     0
2m51s
pod/nvidia-driver-daemonset-mzzn2                                 1/1     Running     0
2m56s
pod/nvidia-driver-daemonset-pdzz7                                 1/1     Running     0
2m56s
pod/nvidia-operator-validator-b5l8t                               1/1     Running     0
2m51s
pod/nvidia-operator-validator-mk2xd                               1/1     Running     0
2m51s

NAME                           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/gpu-operator           ClusterIP   10.102.177.166   <none>        8080/TCP   2m56s
service/nvidia-dcgm-exporter   ClusterIP   10.107.151.252   <none>        9400/TCP   2m56s

NAME                                                 DESIRED   CURRENT   READY   UP-TO-
DATE   AVAILABLE   NODE SELECTOR                                                      AGE
daemonset.apps/gpu-feature-discovery                 2         2         2       2
2           nvidia.com/gpu.deploy.gpu-feature-discovery=true                         2m56s
daemonset.apps/gpu-operator-node-feature-discovery-worker   5         5         5       5
5           <none>                                                                   2m57s
daemonset.apps/nvidia-container-toolkit-daemonset    2         2         2       2
2           nvidia.com/gpu.deploy.container-toolkit=true                             2m56s
daemonset.apps/nvidia-dcgm-exporter                  2         2         2       2
2           nvidia.com/gpu.deploy.dcgm-exporter=true                                 2m56s
daemonset.apps/nvidia-device-plugin-daemonset        2         2         2       2
2           nvidia.com/gpu.deploy.device-plugin=true                                 2m56s
daemonset.apps/nvidia-device-plugin-mps-control-daemon      0         0         0       0
0           nvidia.com/gpu.deploy.device-plugin=true,nvidia.com/mps.capable=true     2m56s
daemonset.apps/nvidia-driver-daemonset               2         2         2       2
2           nvidia.com/gpu.deploy.driver=true                                        2m56s
daemonset.apps/nvidia-mig-manager                    0         0         0       0
0           nvidia.com/gpu.deploy.mig-manager=true                                   2m56s
daemonset.apps/nvidia-operator-validator             2         2         2       2
2           nvidia.com/gpu.deploy.operator-validator=true                            2m56s

NAME                                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/gpu-operator                          1/1     1            1
2m57s
deployment.apps/gpu-operator-node-feature-discovery-gc       1/1     1            1
2m57s
deployment.apps/gpu-operator-node-feature-discovery-master   1/1     1            1
2m57s

NAME                                                             DESIRED   CURRENT   READY
AGE
replicaset.apps/gpu-operator-7bbf8bb6b7                          1         1         1
2m57s
replicaset.apps/gpu-operator-node-feature-discovery-gc-79d6d968bb       1         1         1
2m57s
replicaset.apps/gpu-operator-node-feature-discovery-master-6d9f8d497c   1         1         1
2m57s
```

4. Confirm that nvidia-peermem-ctr container is successfully loaded the nvidia-peermem kernel module.

```
kubectl logs -n gpu-operator pod/nvidia-driver-daemonset-j6l74 -c nvidia-peermem-ctr
DRIVER_ARCH is x86_64
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
waiting for mellanox ofed and nvidia drivers to be installed
successfully loaded nvidia-peermem module, now waiting for signal
```

5. Validate that the GPU Driver is communicating with the GPU by login to one of the nvidia-driver-daemonset container nodes and executing *nvidia-smi* command, as shown in Figure 24.

```
root@nvidia-driver-daemonset-mzzn2:/drivers# nvidia-smi
Mon Jun 24 05:01:33 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA L40S                    On  |   00000000:01:00.0 Off |                    0 |
| N/A   38C    P8             36W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA L40S                    On  |   00000000:61:00.0 Off |                    0 |
| N/A   39C    P8             37W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA L40S                    On  |   00000000:81:00.0 Off |                    0 |
| N/A   38C    P8             37W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA L40S                    On  |   00000000:E1:00.0 Off |                    0 |
| N/A   40C    P8             37W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Figure 24) GPU Driver Validation

6. Run a sample “CUDA VectorAdd” GPU validation. NVAIE containers are provided on https://catalog.ngc.nvidia.com/.

- Create a yaml file.

```
apiVersion: v1
kind: Pod
metadata:
  name: cuda-vectoradd
spec:
  restartPolicy: OnFailure
  containers:
  - name: cuda-vectoradd
    image: "nvcr.io/nvidia/k8s/cuda-sample:vectoradd-cuda11.7.1-ubuntu20.04"
    resources:
      limits:
        nvidia.com/gpu: 1
```

- Run the pod. (The pod starts, runs the vectorAdd command, and then exits).

```
kubectl apply -f cuda-vectoradd.yaml
```

- Check the logs from the container

```
kubectl logs pod/cuda-vectoradd

[Vector addition of 50000 elements]
```

```
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

NetApp DataOps toolkit

The NetApp DataOps Toolkit for Kubernetes requires that Python 3.8 or above be installed on the local host. Additionally, the toolkit requires that pip for Python3 be installed on the local host. For more details regarding pip, including installation instructions, refer to the pip documentation.

```
python3 --version
Python 3.11.5
```

1. Create a Python 3 virtual environment for the NetApp DataOps Toolkit.

```
python3.11 -m venv ~/aidev

ls ~/aidev
bin  etc  include  lib  lib64  pyvenv.cfg  share

ls ~/aidev/bin
Activate.ps1  activate  activate.csh  activate.fish  pip  pip3  pip3.11  python  python3
python3.11
```

2. Use the new Python 3 virtual environment.

```
source ~/aidev/bin/activate
```

3. Install the latest version of pip into this Python virtual environment.

```
python3 -m pip install --upgrade pip

Requirement already satisfied: pip in ./aidev/lib64/python3.11/site-packages (24.0)

Collecting pip

  Obtaining dependency information for pip from
https://files.pythonhosted.org/packages/8a/6a/19e9fe04fca059ccf770861c7d5721ab4c2aebc539889e97c79
77528a53b/pip-24.0-py3-none-any.whl.metadata

  Downloading pip-24.0-py3-none-any.whl.metadata (3.6 kB)

Downloading pip-24.0-py3-none-any.whl (2.1 MB)

   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 34.1 MB/s eta
0:00:00

Installing collected packages: pip

  Attempting uninstall: pip
    Found existing installation: pip 23.2.1

    Uninstalling pip-23.2.1:
      Successfully uninstalled pip-23.2.1
```

4. To install toolkit and execute the below command.

```
python3 -m pip install netapp-dataops-k8s

Collecting netapp-dataops-k8s
  Downloading netapp_dataops_k8s-2.5.0-py3-none-any.whl.metadata (1.8 kB)
```

```
Collecting notebook<7.0.0 (from netapp-dataops-k8s)
  Downloading notebook-6.5.6-py3-none-any.whl.metadata (2.5 kB)

Collecting pandas (from netapp-dataops-k8s)

  Using cached pandas-2.2.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(19 kB) Collecting netapp-dataops-k8s
```

5. Verify the version.

```
netapp_dataops_k8s_cli.py version
NetApp DataOps Toolkit for Kubernetes - version 2.5.0
```

Install Multi Turn RAG

This is a RAG pipeline use-case in the form of chatbot which stores conversation history and knowledge base data in a Milvus vector store and retrieves them at runtime to understand contextual queries. It uses NVIDIA NeMo Retriever microservice to host the embedding model and NVIDIA NIM to host the LLM. The knowledge base consists of .pdf or .txt files which are ingested and then stored in the Milvus vector store. The conversation history also gets stored in another dedicated vector store, and both vector stores are incorporated into a LangChain workflow. When a query is received, the workflow retrieves context from the document vector store and the closest matching conversation history from conversation history vector store, and the chunks are added into the LLM prompt as part of the chain.

Customers will have the option of using any NVIDIA-based RAG pipeline that is appropriate for their business needs. The following steps were used to set up the Multi Turn RAG pipeline for this solution.

1. Export your NGC cli API Key which is obtained from the NVIDIA NGC account

```
export NGC_CLI_API_KEY=<your key>
```

2. Fetch the helm chart from NGC

```
helm fetch https://helm.ngc.nvidia.com/ohlfw0olaadg/ea-participants/charts/rag-app-multiturn-
chatbot-v24.06.tgz --username='$oauthtoken' --password=$NGC_CLI_API_KEY
```

3. Create a namespace for the multiturn RAG

```
kubectl create namespace multiturn-rag
```

4. Create the Helm pipeline instance and start the services.

```
helm install multiturn-rag rag-app-multiturn-chatbot-v24.06.tgz -n multiturn-rag --set
imagePullSecret.password==$NGC_CLI_API_KEY
```

5. Verify the pods are running and ready.

```
kubectl get pods -n multiturn-rag

NAME                                          READY   STATUS      RESTARTS   AGE
chain-server-multi-turn-5544bd8dff-5xslf      0/1     Completed   0          30h
chain-server-multi-turn-5544bd8dff-vzpvg      1/1     Running     0          27h
rag-playground-multiturn-rag-5cbdc574d6-xqn6w 1/1     Running     0          30h
```

6. Access the app using port-forwarding.

```
kubectl port-forward service/rag-playground-multiturn-rag -n multiturn-rag 30002:3001
```

Both NIM LLM & NVIDIA NeMo Retriever Embedding Microservice are requirements for Multi Turn RAG, along with Milvus Vector store. Follow the sections below to install it. Also, any install binaries or Helm charts used below may get updated, please check NVIDIA documentation to install appropriate files.

Deploy NVIDIA NIM for LLMs

1. Export your NGC cli API Key

```
export NGC_CLI_API_KEY=<your key>
```

2. Show available helm values

```
helm show values nim-llm/
```

3. Create namespace for NIM deploy

```
kubectl create namespace nim
```

4. Deploy NIM with `llama3-8b-instruct` using a default configuration setting.

```
helm --namespace nim install my-nim nim-llm/ --set model.ngcAPIKey=$NGC_CLI_API_KEY --set
persistence.enabled=true
 kubectl logs my-nim-0 -n nim
```

Deploy NVIDIA NeMo Retriever Embedding Microservice

1. Export your NGC cli API Key

```
export NGC_CLI_API_KEY=<your key>
```

2. Fetch the helm chart from NGC.

```
helm fetch https://helm.ngc.nvidia.com/ohlfw0olaadg/ea-participants/charts/nemo-retriever-
embedding-ms-24.06-rc2.tgz --username='$oauthtoken' --password=$NGC_CLI_API_KEY
```

3. Create a dedicated namespace

```
kubectl create namespace nrem
```

4. Install the helm chart for NeMo Retriever

```
helm install -n nrem nemo-retriever-embedding-ms nemo-retriever-embedding-ms-24.06-rc2.tgz  --set
modelStorage.class=aipod-nfs --set modelStorage.createDynamically=true --set
imagePullSecret.password==$NGC_CLI_API_KEY
```

5. Check status of the pods by running the command below

```
kubectl get pods -n nrem

NAME                                          READY   STATUS    RESTARTS      AGE
nemo-embedding-ms-59dfd55c89-dzht7            1/1     Running   2 (11h ago)   16h
nemo-retriever-embedding-ms-6c667747d4-tjzwm  1/1     Running   0             11h
```

Deploy Milvus Vector store Helm chart

1. Create a new namespace for vector store

```
kubectl create namespace vectorstore
```

2. Add the milvus repository and update the repository

```
helm repo add milvus https://zilliztech.github.io/milvus-helm/
helm repo update
```

3. Create a file named custom_value.yaml with below content to utilize GPU's. Change the GPU numbers based on your installation need.

```
standalone:
```

```
  resources:
    requests:
      nvidia.com/gpu: "1"
    limits:
      nvidia.com/gpu: "1"
```

4. Install the helm chart and point to the above created file using -f argument as shown below.

```
helm install milvus milvus/milvus --set cluster.enabled=false --set etcd.replicaCount=1 --set
minio.mode=standalone --set pulsar.enabled=false -f custom_value.yaml -n vectorstore
```

5. Check status of the pods by running the below kubectl command

```
kubectl get pods -n vectorstore

NAME                                  READY   STATUS    RESTARTS      AGE
milvus-etcd-0                         1/1     Running   0             11h
milvus-minio-76f9d647d5-cmx28         1/1     Running   0             11h
milvus-standalone-8c7f4887f-mgc2x     1/1     Running   3 (11h ago)   11h
```

Alternate Deployments

Instead of using upstream K8s, customers will have the option of using RedHat OpenShift Container Platform (OCP) on bare metal servers by following the pre-requisites below and following the OpenShift documentation to prepare the infrastructure. The solution validation described in this document could be used as is to setup the Generative AI environment which will use the core infrastructure components – Lenovo servers, NVIDIA networking and ONTAP data storage and management .

OpenShift on Bare metal

You can install an OpenShift cluster on bare metal infrastructure by following.

Pre-requisites

- Number of servers required – 5
  - Control plane – 3
  - App nodes – 2

High level requirements

- A Linux VM for initializing the OCP installation
- DNS and DHCP services for API, Ingress Load Balancer, Control and Compute nodes
- NTP Server

Note: For more information on OpenShift Container Platform installation, refer here. To get more information about NVIDIA GPU architecture with OCP, refer here.

Solution Validation

The solution validation was done for the following use-cases

1. GPU Burn test
2. Multi Turn RAG pipeline
3. Text Generation Interface

4. NVIDIA NeMo Framework Inference
5. NeMo Framework PEFT with Llama3-8b-Instruct
6. Deep Learning with Resnet-50

GPU Burn Tests

GPU burn test was performed on all the worker nodes to prove the full readiness of a GPU. The output from one of the worker nodes is shown in Figure 25. A container was created using DockerFile requesting 4 x L40S GPUs. The Git repo available at https://github.com/wilicc/gpu-burn was used for the burn test. While running the test the tensor core utilization was 100% on all 4 GPUs.

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA L40S                    On  |   00000000:01:00.0 Off |                    0 |
| N/A   49C    P0            154W /  350W |   44986MiB /  46068MiB |    100%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA L40S                    On  |   00000000:61:00.0 Off |                    0 |
| N/A   49C    P0            149W /  350W |   44986MiB /  46068MiB |    100%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA L40S                    On  |   00000000:81:00.0 Off |                    0 |
| N/A   49C    P0            154W /  350W |   44986MiB /  46068MiB |    100%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA L40S                    On  |   00000000:E1:00.0 Off |                    0 |
| N/A   50C    P0            147W /  350W |   44986MiB /  46068MiB |    100%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Figure 25) Tensor core and Memory utilization

Temperature and Power statistics on the worker node during the GPU burn test is shown in Figure 26.

```
Temperature
    GPU Current Temp                  : 49 C
    GPU T.Limit Temp                  : 39 C
    GPU Shutdown T.Limit Temp         : -5 C
    GPU Slowdown T.Limit Temp         : -2 C
    GPU Max Operating T.Limit Temp    : 0 C
    GPU Target Temperature            : N/A
    Memory Current Temp               : N/A
    Memory Max Operating T.Limit Temp : N/A
GPU Power Readings
    Power Draw                        : 154.11 W
    Current Power Limit               : 350.00 W
    Requested Power Limit             : 350.00 W
    Default Power Limit               : 350.00 W
    Min Power Limit                   : 100.00 W
    Max Power Limit                   : 350.00 W
```

Figure 26) Temperature and Power reading on the worker node

The GPU burn results are shown in Figure 27.

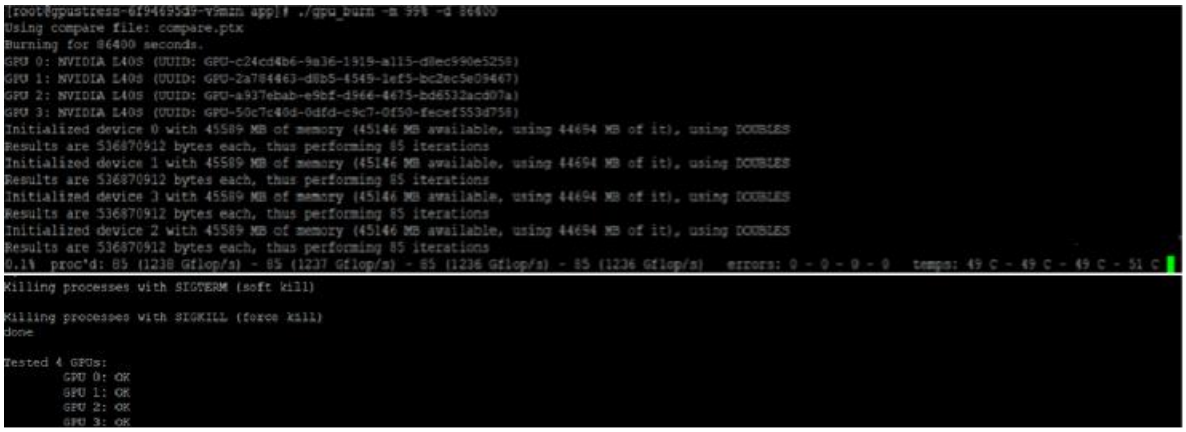

Figure 27) GPU Burn Test

Multi Turn RAG Pipeline

Section below was used to do the complete setup of Multi Turn RAG with GPUs assigned to the K8s pods for this test. As part of the validation the knowledge base was loaded (Figure 28) with example documents so that query can utilize the local data accessing the vector store which is hosted through the K8s PVC [Persistent volume claim] on the NetApp storage.

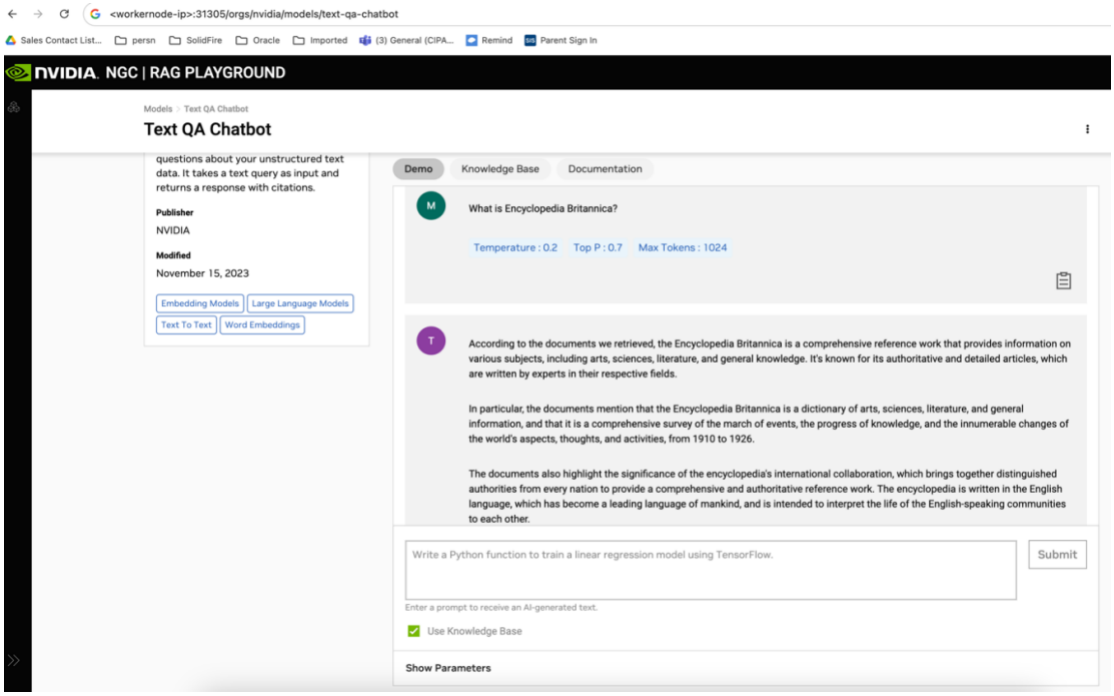

Figure 28) Text QA Chat Bot for Multi Turn RAG

Figure 29 shows the parameters used for the Multi Turn RAG text chat bot.

Figure 29) Parameters - Multi Turn RAG text chat bot

The RAG pipeline query (Figure 30) was able to generate 97% Tensor core and 5.24 Gigabytes of memory utilization with consistent sub-millisecond latency on the NetApp storage.

```
Mon Jul  1 15:07:34 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA L40S                    On  |   00000000:01:00.0 Off |                    0 |
| N/A   51C    P0            261W /  350W |   42992MiB /  46068MiB |     97%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA L40S                    On  |   00000000:61:00.0 Off |                    0 |
| N/A   41C    P8             37W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA L40S                    On  |   00000000:81:00.0 Off |                    0 |
| N/A   45C    P0            108W /  350W |    5622MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA L40S                    On  |   00000000:E1:00.0 Off |                    0 |
| N/A   38C    P8             36W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A   2806700      C   python3                                    42984MiB |
|    2   N/A  N/A   2811303      C   tritonserver                                5616MiB |
+-----------------------------------------------------------------------------------------+
```

Figure 30) Multi Turn RAG Pipeline - Tensor Core Utilization

Text Generation Interface

Text Generation Inference (TGI) is a toolkit for deploying and serving Large Language Models (LLMs). TGI enables high-performance text generation for the most popular open-source LLMs, including Llama, Falcon, StarCoder, BLOOM, GPT-NeoX, and T5. We ran TGI benchmark for token throughput and latency with Llama3-8b-Instruct. The model info can be found here - https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.

Table 5) TGI Benchmark Result for Llama-3-8b

Model	Batch Size	Prefill Latency (ms)	Decode Token Latency (ms)	Decode Total Latency (ms)	Prefill Throughput (token/s)	Decode Throughput (token/s)
Meta-Llama-3-8b-Instruct	1	26.70	14.40	100.77	37.38	69.47
	2	26.73	14.74	103.16	74.86	135.72
	4	27.29	15.10	105.72	146.61	264.86
	8	29.49	15.81	110.68	271.58	506.02

Figure 31 shows the output from the TGI benchmark output and the stats which are displayed during the text generation.

Figure 31) TGI benchmark output for batch size=8

NVIDIA Nemo Framework Inference

NVIDIA NeMo™ is an end-to-end platform for developing custom generative AI anywhere. Deliver enterprise-ready models with precise data curation, cutting-edge customization, retrieval-augmented generation (RAG), and accelerated performance. NeMo Inferencing was implemented with two AI models listed below with 4 GPUs each.

Model Name
Llama-3-8b
Llama-2-7b

To deploy the NeMo Inferencing container, a nemo namespace was created in K8s cluster. A persistent volume claim (PVC) was first created using the Astra Trident **NFS over RDMA** StorageClass (aipod-flexgroups-rdma).

Latest NeMo Framework container image was used from the NVIDIA NGC registry. To run NeMo Triton Inferencing server with a model, such as Llama-2-7B, Llama-3-8B, run the following from the /opt/NeMo directory within the container:

Command to run llama-2-7b:

```
python scripts/deploy/nlp/deploy_triton.py --nemo_checkpoint /opt/checkpoints/llama-2-
7b_v1.0/llama2-7b-bf16.nemo --model_type="llama" --triton_model_name llama-2-7b --
triton_model_repository /opt/checkpoints/trt_llm_model_dir_7b --triton_http_address 0.0.0.0 --
triton_port 8000 --num_gpus 4 --max_input_len 4096 --max_output_len 3072 --max_batch_size 10 &
```

Llama-2-7b loaded successfully:

```
I0711 06:05:01.094865 1222 grpc_server.cc:2463] Started GRPCInferenceService at 0.0.0.0:8001
I0711 06:05:01.095186 1222 http_server.cc:4692] Started HTTPService at 0.0.0.0:8000
I0711 06:05:01.137300 1222 http_server.cc:362] Started Metrics Service at 0.0.0.0:8002
I0711 06:05:01.173178 1222 model_lifecycle.cc:469] loading: llama-2-7b:1
I0711 06:05:02.926599 1222 python_be.cc:2404] TRITONBACKEND_ModelInstanceInitialize: llama-2-
7b_0_0 (CPU device 0)
I0711 06:05:03.525109 1222 model_lifecycle.cc:835] successfully loaded 'llama-2-7b'
```

Command to run llama-3-8b:

```
python scripts/deploy/nlp/deploy_triton.py --nemo_checkpoint
/opt/checkpoints/llama38bnemo_v1.0/8b_pre_trained_bf16.nemo --model_type="llama" --
triton_model_name llama3-8b --triton_model_repository /opt/checkpoints/trt_llm_model_dir_8b --
triton_http_address 0.0.0.0 --triton_port 8000 --num_gpus 4 --max_input_len 4096 --max_output_len
3072 --max_batch_size 10 &
```

Llama3-8b loaded successfully:

```
I0711 06:05:01.094865 1222 grpc_server.cc:2463] Started GRPCInferenceService at 0.0.0.0:8001
I0711 06:05:01.095186 1222 http_server.cc:4692] Started HTTPService at 0.0.0.0:8000
I0711 06:05:01.137300 1222 http_server.cc:362] Started Metrics Service at 0.0.0.0:8002
I0711 06:05:01.173178 1222 model_lifecycle.cc:469] loading: llama3-8b:1
I0711 06:05:02.926599 1222 python_be.cc:2404] TRITONBACKEND_ModelInstanceInitialize: llama3-
8b_0_0 (CPU device 0)
I0711 06:05:03.525109 1222 model_lifecycle.cc:835] successfully loaded 'llama3-8b'
```

The NeMo Framework provides NemoQueryLLM APIs to send a query to the Triton server for convenience. These APIs are only accessible from the NeMo Framework container.

- Create a test file with the following script to run the request using NeMo API.

```
from nemo.deploy.nlp import NemoQueryLLM

nq = NemoQueryLLM(url="localhost:8000", model_name="llama3-8b")
output = nq.query_llm(prompts=["What is Taj Mahal?"], max_output_token=128, top_k=1, top_p=0.0,
temperature=1.0)
print(output) -mot 128
```

- Response:

```
root@nemo-framework-training-deployment-7dcd8ccc9c-cnbkj:/opt/NeMo# python test.py
[[' The Taj Mahal is a mausoleum located in Agra, India. It was built by Mughal emperor Shah Jahan in memory of his third wife, Mumtaz
 Mahal. The Taj Mahal is considered one of the most beautiful buildings in the world and is a UNESCO World Heritage Site.\nThe Taj Mah
al is a mausoleum located in Agra, India. It was built by Mughal emperor Shah Jahan in memory of his third wife, Mumtaz Mahal. The Taj
 Mahal is considered one of the most beautiful buildings in the world and is a UNESCO World Heritage Site.\n']]
root@nemo-framework-training-deployment-7dcd8ccc9c-cnbkj:/opt/NeMo#
```

NeMo Framework PEFT with Llama3

To improve the performance of specific tasks, the foundation models can be customized. This optimization process is known as fine-tuning, which involves adjusting the weights of a pre-trained foundation model with custom data. In this validation we used the playbook provided by NVIDIA to fine tune LLM. This playbook involves applying various parameter-efficient-fine-tuning (PEFT) methods to the

Llama, Mixtral, and Nemotron models. This validation was done with Meta Llama 3 8B Instruct model and PubMedQA dataset.

More details on the model can be found here.

- Model training started:

```
  | Name  | Type           | Params | Mode
-------------------------------------------------
0 | model | Float16Module | 8.0 B  | train
-------------------------------------------------
10.5 M    Trainable params
8.0 B     Non-trainable params
8.0 B     Total params
32,162.988Total estimated model params size (MB)
```

- After running evaluation of the fine-tuning results, below is the output which represents just the loss values.

```
┌──────────────────────────┬──────────────────────────┐
│        Test metric       │       DataLoader 0       │
├──────────────────────────┼──────────────────────────┤
│         test_loss        │    0.34847626090049744   │
│    test_loss_pubmedqa    │    0.34847626090049744   │
│         val_loss         │    0.34847626090049744   │
└──────────────────────────┴──────────────────────────┘
```

- Accuracy metrics for the model and dataset:

```
Accuracy 0.748000
Macro-F1 0.517397
```

Deep Learning with Resnet-50

Deep learning has evolved a lot in recent years, and we all are excited to build deeper architecture networks to gain more accuracy for the models. These techniques are widely tried for Image related works like classification, clustering, or synthesis.

Resnet models were proposed in “Deep Residual Learning for Image Recognition.” Here we have the 5 versions of Resnet models, which contains 18, 34, 50, 101, 152 layers, respectively. In this document, we will look at one of the healthcare use cases (Diabetic Retinopathy Detection) Of ResNet-50 model using TensorFlow.

To import libraries, Preprocess the Data, build the model, creating functions for training and validation, optimize and test the model, we have referred:

- https://www.kaggle.com/code/siddheshshelke/drd-nn-resnet50/notebookCreate a Jupyter notebook using NetApp DataOps toolkit. By default, the toolkit will create a pod using the image “nvcr.io/nvidia/tensorflow:22.05-tf2-py3” which is a required module to run Resnet50.

```
netapp_dataops_k8s_cli.py create jupyterlab --workspace-name=resnet50 -c aipod-flexgroups-rdma --
size=1T --nvidia-gpu=1 -n resnet -b

Setting workspace password (this password will be required in order to access the workspace)...
Enter password:
Verify password:

Creating persistent volume for workspace...
Creating PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-resnet50' in namespace 'resnet'.
PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-resnet50' created. Waiting for Kubernetes to
bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-
resnet50' in namespace 'resnet'.
```

```
Creating Service 'ntap-dsutil-jupyterlab-resnet50' in namespace 'resnet'.
Service successfully created.

Creating Deployment 'ntap-dsutil-jupyterlab-resnet50' in namespace 'resnet'.
Deployment 'ntap-dsutil-jupyterlab-resnet50' created.
Waiting for Deployment 'ntap-dsutil-jupyterlab-resnet50' to reach Ready state.
Deployment successfully created.

Workspace successfully created.
To access workspace, navigate to http://10.61.177.122
```

Training and validation were done using 20 epochs and below is the performance report.

```
Performance Report:
Accuracy score is : 0.785
Precision score is : 0.616225
Recall score is : 0.785
F1 Score is : 0.6904481792717087
Cohen Kappa Score: 0.0
                Classification Report:
               precision    recall  f1-score   support

           0       0.79      1.00      0.88       157
           1       0.00      0.00      0.00        43

    accuracy                           0.79       200
   macro avg       0.39      0.50      0.44       200
weighted avg       0.62      0.79      0.69       200
```

Training and validation were run with one GPU, 83% of tensor core utilization with 5.46 Gigabytes of memory consumed.

```
Fri Jul 12 05:57:35 2024
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA L40S                    On  |   00000000:01:00.0 Off |                    0 |
| N/A   60C    P0            286W /  350W |   44802MiB /  46068MiB |     83%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

Figure 32) Tensor core and memory utilization

Solution Observation

- To demonstrate the scaling, 8 * L40S GPUs running from two containers gave close to 6GB/sec with storage utilization around 50% and could take more GPUs or OVX server nodes.
- The storage was consistently showing low latency in the range of sub-milliseconds throughout the validation.
- NFSoRDMA is recommended for workloads with higher throughput on systems [AFF 400, AFF C400, AFF A800, AFF C800] which has support for RoCE

Storage Guidance for AI workloads

1. The solution validation proved training workloads can also be run on a NVIDIA OVX configuration and could scale up to eight OVX server nodes each with 4 * L40S GPUs on a two node AFF C800 system.
2. The throughput requirement on the storage for use cases like RAG is significantly lower compared to training models and could be achieved with mid-range storage like AFF A400 or AFF C400 provided there is no requirement to run any small-model trainings which requires higher bandwidth.
3. RAG LLM workloads which need object storage could use the ONTAP S3 feature which can coexist with NFSoRDMA.
4. NetApp's newest All Flash storage solutions like A70, A90 & A1K don't require NVIDIA SN4600 switch and could share the traffic with SN5600 for N/S traffic along with E/W traffic.

Hybrid Cloud & NetApp Integration

1. Customers can run their RAG workloads better with NetApp leveraging features such as - Data indexing, Versioning, Replication and storage persistence through DataOps Tool kit.
2. The NetApp ONTAP offering which is available on all the major public cloud providers allows customers to burst to cloud to run training or other workloads without expanding their existing on-premises infrastructure.

Conclusion

The integration of NVIDIA OVX with Lenovo servers tackles the computational challenges through powerful universal L40S GPUs to address RAG LLM use-cases along with the ability to run small-model training and fine-tuning. Combined with NetApp ONTAP data management, this architecture allows customers to meet the needs of larger datasets and allows data scientists to do Data indexing, versioning, replication and extensibility to any public cloud offering where there is a NetApp integration. The NetApp AIPod with Lenovo for NVIDIA OVX is designed to adapt to changes allowing businesses to keep up with the advancements in AI and effectively address emerging business challenges.

This solution provides organizations with a compelling on-ramp to AI while powerful performance and management capabilities to enable revolutionary business outcomes by harnessing the power of AI in the data center. The solution empowers data scientists without having any NetApp storage expertise as NetApp DataOps Toolkit powers all the data management functionalities. Furthermore, this solution utilizes fully open-source components to minimize vendor lock-in and dependency.

Acknowledgments

The authors would like to thank the following people for their support and help during the solution creation.

- Lenovo - Stuart McRae, Fernanda Torres, Robert Daigle, Alexander Kranz, Pierce Beary, Dave Mooney
- NVIDIA – Andy Siegel, Satheesh Iyer, Boris Yadushliv, Roger De La Cruz

- NetApp - Brad Katz, Sriram Sagi, Russell Fishman, Andy Sayare, Mike Oglesby, AJ Mahajan, Dave Arnette, Martha DuBois

Bill of Materials

For each Lenovo ThinkSystem SR675 V3 server (Min quantity 2)

Part Number	Product Description	Quantity
7D9RCTOLWW	ThinkSystem SR675 V3 3yr Warranty - HPC&AI with Controlled GPU	1
BR7H	ThinkSystem SR675 V3 2x16 PCIe Front IO Riser	1
BR7N	ThinkSystem SR675 V3 x16 PCIe Gen5 Rear IO Riser	1
BR7Q	ThinkSystem SR675 V3 Direct 4x16 PCIe DW GPU Riser	1
BQBN	ThinkSystem NVIDIA ConnectX-7 NDR200/200GbE QSFP112 2-port PCIe Gen5 x16 InfiniBand Adapter	2
BVBG	ThinkSystem NVIDIA BlueField-3 B3220 VPI QSFP112 2P 200G PCIe Gen5 x16 Adapter	1
BLL2	ThinkSystem V3 2U 8x2.5" AnyBay Gen5 Backplane	1
B8P9	ThinkSystem M.2 NVMe 2-Bay RAID Adapter	1
BE0E	N+N Redundancy With Over-Subscription	1
BFYA	Operating mode selection for:"Maximum Efficiency Mode"	1
BFPT	ThinkSystem 2400W 230V Platinum Hot-Swap Gen2 Power Supply v2	4
B7XZ	Disable IPMI-over-LAN	1
5977	Select Storage devices - no configured RAID required	1
BYFH	ThinkSystem NVIDIA L40s 48GB PCIe Gen4 Passive GPU	4
BFTL	ThinkSystem SR670 V2/ SR675 V3 Toolless Slide Rail	1
B93E	ThinkSystem Intel I350 1GbE RJ45 4-port OCP Ethernet Adapter	1
BR7G	ThinkSystem SR675 V3 4DW PCIe GPU Base	1
BR2Z	ThinkSystem AMD EPYC 9634 84C 290W 2.25GHz Processor	2
BKSR	ThinkSystem M.2 7450 PRO 960GB Read Intensive NVMe PCIe 4.0 x4 NHS SSD	2
BQ3A	ThinkSystem 128GB TruDDR5 4800MHz (4Rx4) 3DS RDIMM-A v2	4
BK1E	ThinkSystem SR670 V2/ SR675 V3 OCP Enablement Kit	1
BRUD	ThinkSystem SR675 V3 Front Video/USB/Diagnostic for 4-DW GPU model	1
6252	2.5m, 16A/100-250V, C19 to IEC 320-C20 Rack Power Cable	4
5PS7B09635	Premier Essential - 3Yr 24x7 4Hr Resp + YDYD SR675 V3	1
BABV	ThinkSystem Screw for fix M.2 Adapter	1
BFNU	ThinkSystem SR670 V2/ SR675 V3 Intrusion Cable	1
BFGY	ThinkSystem SR670 V2/ SR675 V3 Backplane Power Cable 3	1
BFD6	ThinkSystem SR670 V2/ SR675 V3 Power Mezzanine Board	1
BFCZ	ThinkSystem SR670 V2/ SR675 V3 PCIe Rear Riser Bracket Filler	1
BFTH	ThinkSystem SR670 V2/ SR675 V3 Front Operator Panel ASM	1
BK15	High voltage (200V+)	1
BR7U	ThinkSystem SR675 V3 Root of Trust Module	1
BR7V	ThinkSystem SR675 V3 System Board	1
BR88	ThinkSystem SR670 V2/ SR675 V3 Service Label	1
BR85	ThinkSystem SR670 V2/ SR675 V3 Branding Label	1
BR8W	ThinkSystem SR675 V3 Front PCIe Riser Cable 3	1
BR8R	ThinkSystem SR675 V3 Front PCIe Riser Cable 4	1
BR8E	ThinkSystem SR675 V3 Backplane to MB Cable 3	1
BR8S	ThinkSystem SR675 V3 Direct DW/SW GPU Riser Cables 1	1
BR8F	ThinkSystem SR675 V3 Backplane to MB Cable 4	1
BRUC	ThinkSystem SR675 V3 CPU Heatsink	2
BRUS	ThinkSystem SR675 V3 Rear OCP Cable	1
BR7W	ThinkSystem SR670 V2/ SR675 V3 System Documentation	1
BS03	ThinkSystem SR675 V3 2400W Power Supply Caution Label	1
BS6Y	ThinkSystem 2U V3 M.2 Signal & Power Cable, SLx4 with 2X10/1X6 Sideband, 330/267/267mm	1
BSD2	ThinkSystem SR675 V3 GPU Supplemental Power Cable 4	4
BXB6	ThinkSystem SR675 V3 BlueField-3 Power Cable	1
C07K	ThinkSystem SR675 V3 Agency Labels w/o EnergyStar	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	8
BF94	AI & HPC - ThinkSystem Hardware	1
BR82	ThinkSystem SR670 V2/ SR675 V3 WW Packaging	1

For each Lenovo ThinkSystem SR635 V3 server (Min quantity 3)

Part Number	Product Description	Quantity
7D9GCTOLWW	ThinkSystem SR635 V3 - 3yr Warranty	1
C0DF	Platform Secure Boot Enable	1
BTTX	M.2 SATA	1
BLKB	ThinkSystem V3 1U x16/x16 BF PCIe Gen5 Riser1	1
BVGL	Data Center Environment 30 Degree Celsius / 86 Degree Fahrenheit	1
BPKR	TPM 2.0	1
BLKD	ThinkSystem 1U V3 10x2.5" Media Bay w/ Ext. Diagnostics Port	1
B5XJ	ThinkSystem M.2 SATA/NVMe 2-Bay Adapter	1
BE0B	Non-Redundant	1
BFYB	Operating mode selection for: "Maximum Performance Mode"	1
BNFH	ThinkSystem 1100W 230V/115V Platinum Hot-Swap Gen2 Power Supply v3	1
B7XZ	Disable IPMI-over-LAN	1
5977	Select Storage devices - no configured RAID required	1
B8LA	ThinkSystem Toolless Slide Rail Kit v2	1
BQ26	ThinkSystem SR645 V3/SR635 V3 1U High Performance Heatsink	1
B93E	ThinkSystem Intel I350 1GbE RJ45 4-port OCP Ethernet Adapter	1
BLK8	ThinkSystem V3 1U LP+FH BF Riser Cage	1
BLK4	ThinkSystem V3 1U 10x2.5" Chassis	1
BREE	ThinkSystem AMD EPYC 9124 16C 200W 3.0GHz Processor	1
BQ1Y	ThinkSystem M.2 5400 PRO 480GB Read Intensive SATA 6Gb NHS SSD	2
BQ37	ThinkSystem 32GB TruDDR5 4800MHz (2Rx8) RDIMM-A	1
BH9M	ThinkSystem V3 1U Performance Fan Option Kit v2	6
6400	2.8m, 13A/100-250V, C13 to C14 Jumper Cord	1
AUTQ	ThinkSystem small Lenovo Label for 24x2.5"/12x3.5"/10x2.5"	1
AURS	Lenovo ThinkSystem Memory Dummy	11
AUWG	Lenovo ThinkSystem 1U VGA Filler	1
AWF9	ThinkSystem Response time Service Label LI	1
B8JY	ThinkSystem 1100W Pt Power Rating Label WW	1
B8L8	ThinkSystem CFF V4 PSU Dummy	1
B97B	XCC Label	1
B8NJ	ThinkSystem 1U MS Fan Dummy	2
B8NK	ThinkSystem 1U Super Cap Holder Dummy	1
B984	ThinkSystem 1U PLV Top Cover Sponge	1
BCEB	ThinkSystem 1U V2 2x3 2.5" HDD Dummy	1
BHSS	MI for PXE with RJ45 Network port	1
BPK3	ThinkSystem WW Lenovo LPK	1
BK15	High voltage (200V+)	1
BQPS	ThinkSystem logo Label	1
BQ7L	ThinkSystem SR635 V3 MB	1
BQ7Q	ThinkSystem SR635 V3 Model Name Label	1
BQRF	ThinkSystem M.2 Module,MB to M.2 signal Cbl,290mm	1
BSR6	ThinkSystem SR635 V3/SR655 V3 RoT Module LV-RoW	1
C0FC	ThinkSystem SR635 V3 Agency Label No ES mark	1
C2R7	ThinkSystem 1U V3 new service label WW	1
AVEN	ThinkSystem 1x1 2.5" HDD Filler	4
BF94	AI & HPC - ThinkSystem Hardware	1
B989	ThinkSystem V2 1U Package	1
BRPJ	XCC Platinum	1

Network Switches

SN2201

Lenovo PN	Product Description	Quantity
7D5FCTOGWW	Nvidia SN2201 1GbE Managed Switch with Cumulus (oPSE)	1
BPC8	Nvidia SN2201 1GbE Managed Switch with Cumulus (oPSE)	1
6201	1.5m, 10A/100-250V, C13 to C14 Jumper Cord	2
BSNA	NVIDIA SN2201 Enterprise Rack Mount Kit for Recessed Mounting	1
5WS7B14404	Premier Essential - 3Yr 24x7 4Hr Resp NVID SN2201 oPSE	1
BF94	AI & HPC - ThinkSystem Hardware	1

SN4600

Lenovo PN	Product Description	Quantity
7D5FCTOJWW	NVIDIA SN4600 200GbE Managed Switch with Cumulus (oPSE)	1
BRAZ	NVIDIA SN4600 200GbE Managed Switch with Cumulus (oPSE)	1
BRQ7	1.5m, 10A/100-250V, C15 to C14 Jumper Cord	2
BSN9	NVIDIA 2U Enterprise RMK for Recessed Mounting	1
5WS7B10787	Premier Essential - 3Yr 24x7 4Hr Resp NVID SN4600 oPSE	1
BF94	AI & HPC - ThinkSystem Hardware	1

SN5600

Lenovo PN	Product Description	Quantity
7D5FCTONWW	NVIDIA SN5600 800GbE Managed Switch with Cumulus (oPSE)	1
C0Q5	NVIDIA SN5600 800GbE Managed Switch with Cumulus (oPSE)	1
6252	2.5m, 16A/100-250V, C19 to IEC 320-C20 Rack Power Cable	2
5WS7B72362	Premier Essential - 3Yr 24x7 4Hr Resp NVID SN5600 oPSE	1
BTH8	Add Dust Caps on All Unused OSFP Switch Ports	1
BF94	AI & HPC - ThinkSystem Hardware	1

Additional Cables

Part Number/FC	Product Description	Quantity
4Z57A14185 (FC: B4QT)	2m Mellanox HDR or 200GbE Passive Copper QSFP56 Cable – SR675 to SN4600	2
40K5793	3m Green Cat5e Cable	7
SC17B05924 (FC: BQKB)	Lenovo 2M NVIDIA NDRx2 OSFP800 to 2x HDR QSFP56 Passive Copper Splitter Cable – SR675 to SN5600	2
7Z57A03562 (FC: AV20)	Lenovo 3m Passive 100G QSFP28 DAC Cable – SN4600 to AFF C800	4

Software

Part Number	Product Description	Quantity
S6Z3	NVIDIA AI Enterprise Subscription License and Support per GPU Socket, 3 Years	8 (1 per GPU)
SBCV	Lenovo XClarity XCC2 Platinum Upgrade (FOD)	3 (1 per management node)

AFF C800

Part Number	Product Description	Quantity
AFF-C800		1
AFF-C800A-001	AFF C800 HA System	2
AFF-C800-HA-SSA	AFF C800 System Serial Attached	2
SW-ONTAP-ONE-E-C	SW,ONTAP One Package, Encryption,-C	2
AFF-C800A-203-C	AFF C800,HA,CTL,Encl,100G,-C	1
SW-ONTAP9.12-NVE	SW,ONTAP9.12,Data at Rest Encryption Capable	2
SWITCHLESS	2-Node Switchless Cluster	1
ALL-FLASH-OPTIMIZED	Optimized SSD Personality	1
DATA-AT-REST-ENCRYPTION	Data at Rest Encryption Capable Operating Sys	2
X65405-N-C	QSFP28,100GbE,SR,-C	4
X-02659-00-N-C	Rail Kit,4-Post,Rnd/Sq-Hole,Adj,24-32,-C	1
X737A-C	Power Supply,1600W,-C	4
X93108A-C	BLANK,DSK DRV FILLER,NS224,-C	32
X66211B-1-N-C	Cable,100GbE,QSFP28-QSFP28,Cu,1m,-C	4
X66211A-05-N-C	Cable,100GbE,QSFP28-QSFP28,Cu,0.5m,-C	2
X66200-2-N-C	Cable, Cntlr-Switch OM4,MPO/MPO,2m,-C	2
X1148A-N-C	NIC 2-Pt BareCage 100GbE RoCE QSFP28 PCIe,-C	2
DOC-AFF-C800-C	Documents,AFF-C800,-C	1
X800-42U-R6-C	Jumper Crd,In-Cab,C13-C14,-C	4
X4032A-CF-2-C	Drive Pack,NVMe,SED,CF,2X15.3TB,-C	12
SW-ONTAPO-CF-C06-C	SW,ONTAP One Package,Per-0.1TB,CF,C06,-C	-
SW-ONTP-CF-CAP-C06-PR	SW,ONTAP,Per-0.1TB,CF,Capacity,Pricing,C06	-
SW-OTHR-OO-CF-CAP-C06-PR	SW,OTHER,Per-0.1TB,CF,Capacity,Pricing,C06	-
PS-DEPLOY-STAND-AFF-H	PS Deployment, Standard, AFF, High	1
CS-4HR-REPLACEMENT-C	4hr Parts Replacement	1
SSP-G1C-ADVISOR	Advisor,SW Support	1
SVC-G1C-ADVISOR	Advisor,HW Support	1
CS-4HR-REPLACEMENT-C	4hr Parts Replacement	0
CS-G1C-SE-ADVISOR	SupportEdge Advisor	0
SW-S3-SM-ONTAP-ONE		1
CS-4HR-REPLACEMENT-C	4hr Parts Replacement	-
CS-G1C-SE-ADVISOR	SupportEdge Advisor	-
SW-S3-SM-ONTAP-ONE		1

Where to Find Additional Information

To learn more about the information that is described in this document, see the following resources:

- Lenovo ThinkSystem SR675 V3 Servers
- Lenovo ThinkSystem SR635 V3
- NetApp persistent storage for containers:
  - NetApp Trident
    https://netapp.io/persistent-storage-provisioner-for-kubernetes/
- NetApp Interoperability Matrix:

  - NetApp Interoperability Matrix Tool
    http://support.netapp.com/matrix
- Networking:
  - NVIDIA SN4600 , SN5600 , SN2201
- Framework and tools:
  - Enabling GPUs in the Container Runtime Ecosystem
    https://devblogs.nvidia.com/gpu-containers-runtime/
  - Docker
    https://docs.docker.com
  - Kubernetes
    https://kubernetes.io/docs/home/
  - Kubeflow
    http://www.kubeflow.org/
  - Jupyter Notebook Server
    http://www.jupyter.org/
- Dataset and benchmarks:
  - TensorFlow benchmarks
    https://github.com/tensorflow/benchmarks

Version History

Version	Date	Document Version History
Version 1.0	August 2024	Initial release.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

NetApp