Spectro Cloud

NVIDIA Enterprise AI Factory

Reference architecture for Supermicro SYS-212GB-FNR

Rev. 1.0.1 | January 2026

About this reference architecture

This document represents the reference architecture for the NVIDIA Enterprise AI Factory for Supermicro SYS-212GB-FNR systems. It is a comprehensive explanation of the technology used, each component's purpose and configuration.

The selected systems in this document are intended to showcase a known working solution. That does not imply that other systems are incompatible, far from it. To aid in the selection of alternative systems, this reference architecture provides the necessary requirements for running the NVIDIA Enterprise AI Factory platform.

Change History

Version	Release date	Change summary
1.0.0	November 2025	Initial version
1.0.1	January 2026	Updated DOCA, GPU Operator and Network Operator

Introduction

This section provides the purpose, scope, and the intended audience of this document.

Purpose

This reference architecture provides a standard, repeatable and highly scalable design that can be adapted to specific environments and customer requirements. It aims at developing a scalable Kubernetes infrastructure environment capable of running AI workloads in a performant manner.

Scope

This reference architecture:

- Demonstrates a platform for running AI workloads through NVIDIA Run:ai.
- Provides guidelines for hardware sizing and component selection, depending on the type of AI workload.
- Provides the foundation for follow-up documents that extend the reference architecture to the NVIDIA AI Data Platform (AIDP) by adding storage solutions.

Audience

This reference architecture is intended for customers — IT architects, consultants and administrators — involved in the early phases of planning, design and deployment of Kubernetes-based AI solutions using Spectro Cloud PaletteAI. It is assumed that the reader is familiar with the concepts and operations of Kubernetes technologies and Spectro Cloud products.

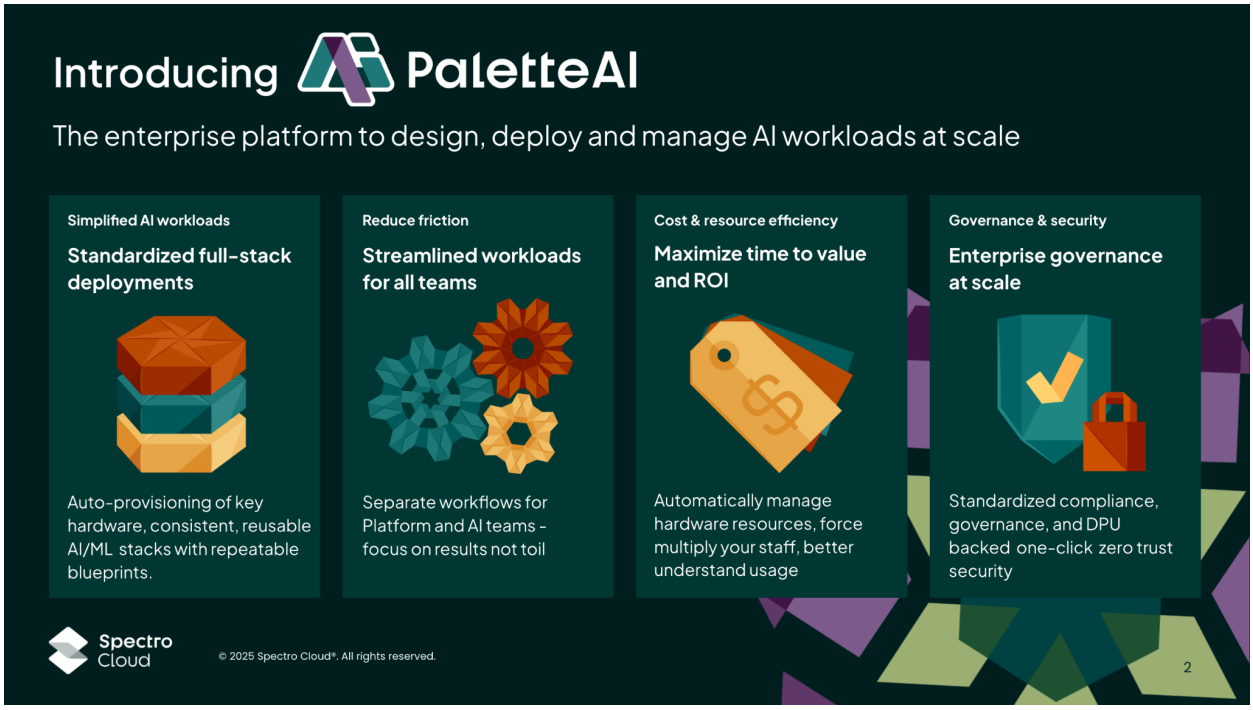
Technology Overview

Overview

This section provides an overview of the technologies that are used in this solution:

- Management systems:
  - Spectro Cloud PaletteAI
  - Canonical MAAS 3.7.0
- Infrastructure hardware:
  - Supermicro SuperServer SYS-212GB-FNR (1S, 2U rackmount) with
    - Intel 6774P 64-core processor, 350W, 136 PCIe 5.0 lanes, max 2TB RAM
    - Up to 4 NVIDIA GPUs of either:
      - NVIDIA RTX PRO™ 6000 Blackwell Server Edition (96GB GDDR7, 512-bit bus)
      - NVIDIA H200 NVL (141GB HBM3e, 6144-bit bus) with NVIDIA NVLink™ bridge
    - Up to 2 Bluefield-3 B3140H SuperNICs for GPU-to-GPU traffic
    - Bluefield-3 B3220 DPU for north-south traffic
    - Up to 4 NVMe E1.S 15mm drives (1-8TB each)
    - Up to 2 NVMe M.2 drives
  - NVIDIA Spectrum-X SN5610 Ethernet switches for the data planes
  - NVIDIA Spectrum-X SN2201 Ethernet switches for the OOB management
- Infrastructure software:
  - Ubuntu 22.04 LTS
  - Kubernetes 1.33.5
  - Cilium 1.18.1
  - Longhorn 1.9.0
  - NVIDIA Network Operator 25.10.0
- AI Middleware:
  - MetalLB 0.15.2
  - KGateway 2.1.1
  - NVIDIA GPU Operator 25.10.1
  - Prometheus Operator 78.3.0
- AI Runtime:
  - Kubeflow Training Operator 1.8.1
  - KNative Operator 1.20.0
  - NVIDIA Run:ai Control Plane 2.23.20
  - NVIDIA Run:ai Workload Cluster 2.23.20

Spectro Cloud PaletteAI

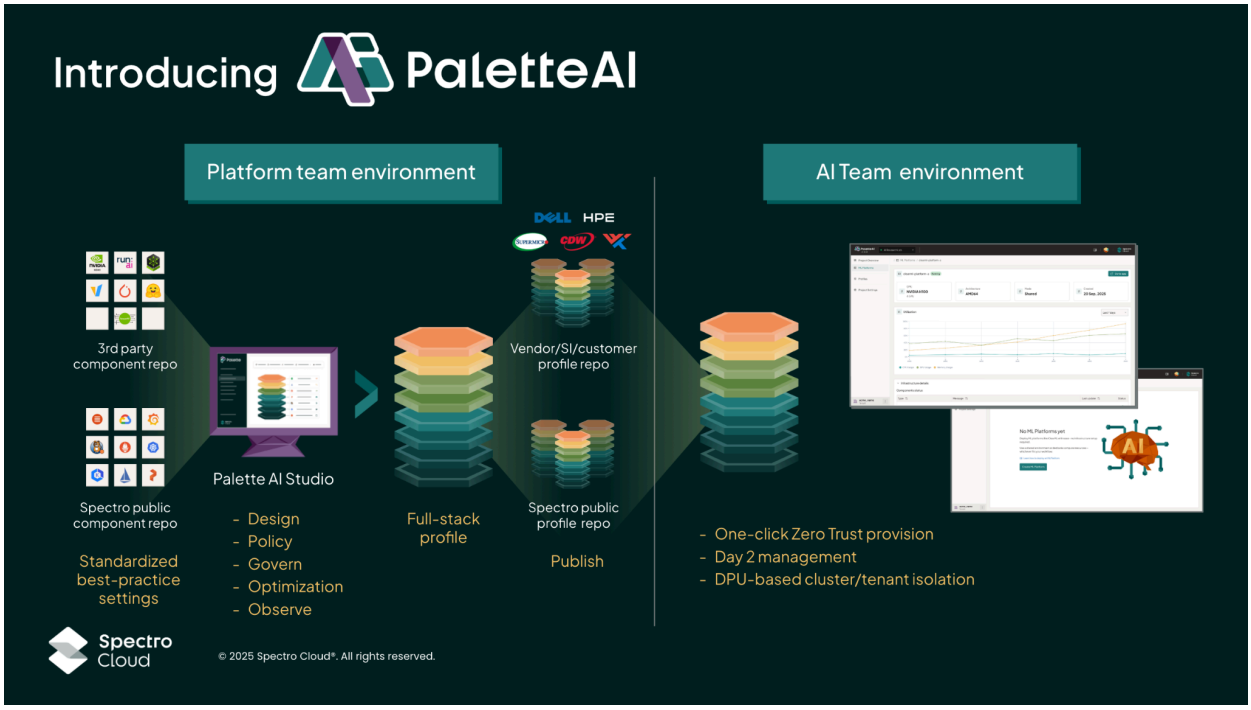


Spectro Cloud PaletteAI is the fastest path to realizing the vision of the AI factory, built from end-to-end accelerated infrastructure in your data centers. It is validated to deploy, configure and manage hardware in accordance with NVIDIA's Enterprise AI Factory validated designs.

That includes NVIDIA Blackwell GPUs, BlueField 3 DPUs, Spectrum-X networking, and the whole ecosystem of software and hardware you need for model training, fine-tuning, inference and more — from the OS and Kubernetes distribution, to NIM microservices, NeMo, and AIDP.

But we know that AI lives beyond the four walls of the data center. With PaletteAI you can also build and deploy clusters for edge inference — one of the fastest-growing categories of AI, and one of the most challenging to achieve at production scale. We can help you make it real.

PaletteAI is brought to you by Spectro Cloud, the team behind Palette, the leading Kubernetes management platform. Deploying full software stacks across cloud, data center and edge — it's what we do best. Today's AI factories run on Kubernetes, so nobody is better placed to build the foundation for your AI future.



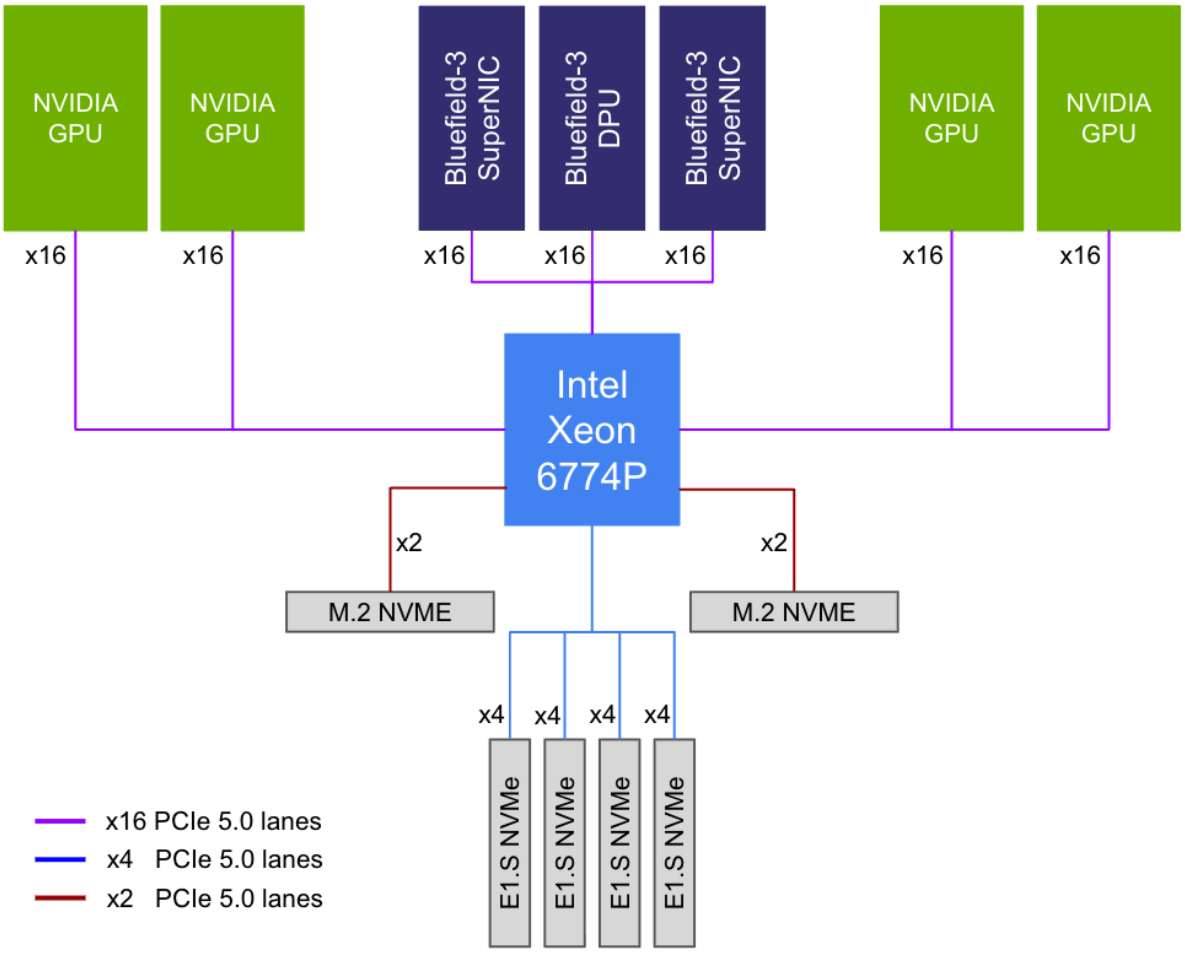
Spectro Cloud provides full prebuilt stack profiles for leading AI solutions such as NVIDIA Run:ai from an online library of content we call PaletteAI Studio. Your platform team can retrieve the desired AI stacks from PaletteAI Studio and import them into PaletteAI. From there the stacks can be published to AI practitioners for simplified self-service deployment.

Canonical MAAS

Canonical MAAS is an open-source bare metal deployment solution that uses PXE network boot to deploy an operating system onto bare metal servers automatically. Spectro Cloud has developed a Cluster API provider for MAAS to enable fully automated deployment of Kubernetes clusters onto bare metal hardware and day-2 operations for cluster upgrades.

Supermicro SuperServer SYS-212GB-FNR

The Supermicro SuperServerSYS-212GB-FNR is a versatile chassis for AI computing. It can support different GPU and NIC configurations, providing flexibility to scale to both smaller and larger deployments.



The main differentiators are:

- Single socket Intel Xeon 6774P CPU system, boasting 64 cores and 136 PCIe 5.0 lanes. Having a single CPU socket prevents GPU-to-GPU communication over the bandwidth-limited CPU interconnect found in multi-CPU systems. With 136 PCIe 5.0 lanes, every peripheral can be connected at full bandwidth on a single NUMA node.

- Up to 4 GPUs of either NVIDIA RTX PRO 6000 Blackwell Server Edition or H200 NVL

    - The NVIDIA RTX PRO 6000 Blackwell Server Edition is the right choice for organizations planning to support larger AI workloads with the latest precision modes, graphics innovation, and memory capacity. It provides a future-ready foundation for simulation, agentic AI, and media processing at scale. It natively supports FP4 precision for LLMs, delivering double the performance and half the memory usage compared to the default method of FP16. With 96GB of GDDR7 VRAM, it provides double the memory compared to its L40S predecessor.

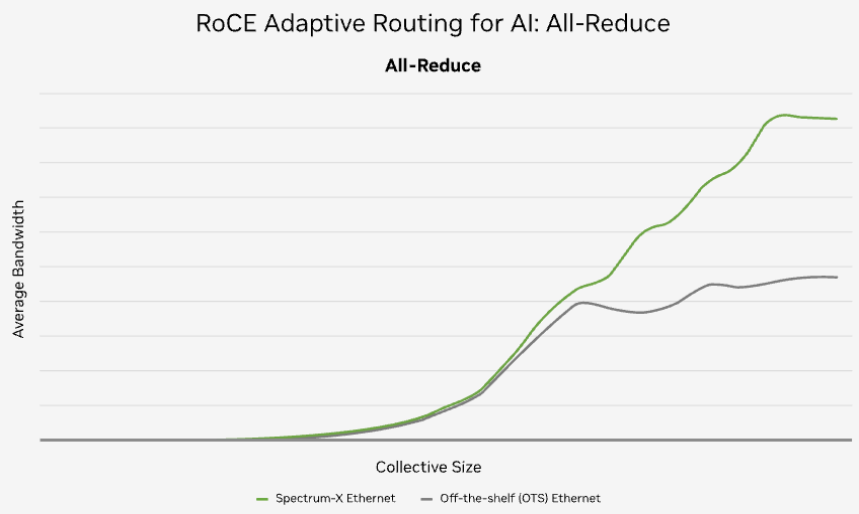
    - The H200 NVL is the premier choice for organizations planning to run large-scale AI, ML, and HPC applications that benefit from large memory capacity and high bandwidth, such as LLM inference and fine-tuning, processing large genomic datasets, real-time autonomous vehicle sensor data, and complex scientific simulations. The 141GB of HBM3e memory sits directly on the GPU die, connected via a 6144–bus for 4× the memory bandwidth of the RTX PRO 6000 Blackwell. Multiple H200s can be directly interconnected with the NVLink bridge to support even larger workloads.

- Up to 2 Bluefield-3 B3140H SuperNICs for GPU-to-GPU communications across chassis. Scaling in a 2 GPU to 1 SuperNIC ratio, the system lets you run 1 SuperNIC in a dual-GPU configuration and 2 SuperNICs in a quad-GPU configuration, supporting a GPU-to-GPU bandwidth of 200Gbps per GPU.

- 1 Bluefield-3 B3220 DPU for north-south communications and PXE boot for OS installation. The DPU enables offloading of certain tasks from the CPU, delivering faster options to ingest data or inspect traffic flows.

NVIDIA Spectrum-X Ethernet

When AI clouds use traditional IP/Ethernet as their compute network, they achieve only a portion of the MLPerf performance levels possible with an optimized network. In multi-tenant environments where multiple AI jobs run simultaneously, traditional Ethernet struggles to provide performance isolation, risking the interference of one tenant's jobs with another's. For those deploying AI clouds with Ethernet, NVIDIA has developed the Spectrum-X Networking Platform which improves performance, while increasing the predictability and power efficiency of Ethernet-based AI clouds.



The NVIDIA® Spectrum™-X Ethernet Networking Platform is the first platform designed specifically to improve the performance and efficiency of Ethernet-based AI clouds. This breakthrough technology, which is based on open standards, delivers markedly better network performance for massive AI workloads such as LLM and inferencing, than traditional Ethernet solutions. It also improves power efficiency and ensures consistent, predictable performance in multi-tenant environments. Spectrum-X Ethernet leverages network innovations achieved through the tight coupling of the Spectrum-X Ethernet switches with the NVIDIA BlueField®-3 SuperNIC. This integration facilitates the delivery of end-to-end network capabilities purpose-built for AI workloads, thereby reducing the run times of massive transformer-based generative AI models. As a result, network engineers, data scientists, and cloud service providers can attain faster results and make informed decisions.

An effective AI compute network is defined by its ability to support and accelerate AI workloads. Optimization is crucial for every aspect of the network, from the SuperNICs to the switches, cables/optics, networking, and acceleration software. To address these needs, NVIDIA created a number of new innovations to achieve the highest effective bandwidth under load and at scale:

1. NVIDIA RDMA over Converged Ethernet (RoCE) Adaptive Routing on Spectrum-X switches
2. NVIDIA Direct Data Placement (DDP) on Spectrum-X SuperNIC
3. NVIDIA RoCE Congestion Control on both Spectrum-X switches and SuperNICs
4. NVIDIA AI Acceleration Software
5. End-to-End AI Network Visibility

Ubuntu

Ubuntu 22.04 is the current OS of choice for our reference architecture. It is natively supported by Canonical MAAS, Spectro Cloud Palette, NVIDIA DOCA and the RTX PRO 6000 Blackwell and H200 NVL GPUs. Due to a compatibility issue between the RTX PRO 6000 Blackwell and the 6.8 Linux kernel version, support for Ubuntu 24.04 is not currently available.

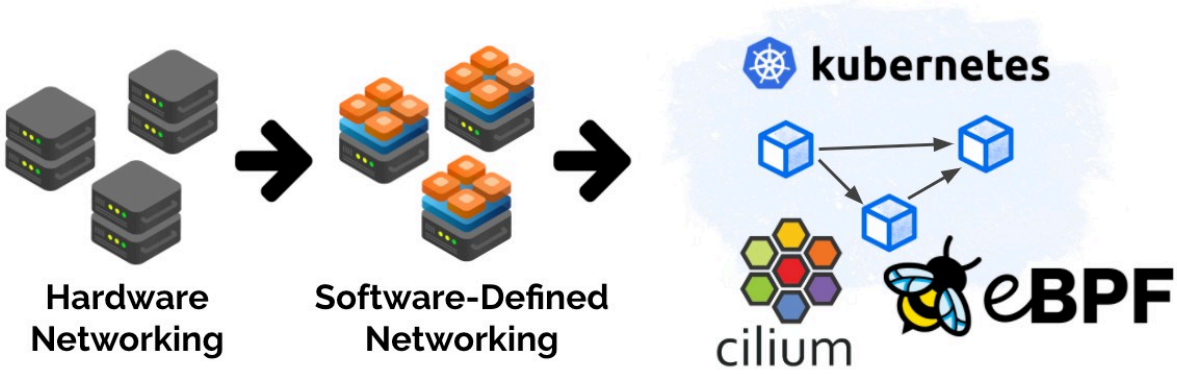
Kubernetes

Kubernetes 1.33 ensures support for all the components in the reference architecture, most importantly Spectro Cloud Palette and NVIDIA Run:ai. Every release of Kubernetes improves the capabilities for running AI and HPC workloads, hence we want to be using the latest version possible.

Spectro Cloud recommends keeping your Kubernetes versions current using Spectro Cloud Palette, where we validate and cross-reference all packs deployed in a Cluster Profile to provide guardrails and protection against incompatible and unsupported configurations. Palette handles the pre-flight validation so you don't have to learn the hard way.

Cilium

Cilium is a modern, open-source networking and security solution for containerized environments. It provides high-level visibility and control over network traffic and offers advanced security features, including encryption, network policy enforcement, and more. Cilium uses eBPF to provide high-performance networking and security, and it is designed to work with Kubernetes.



We recommend the use of Cilium as it provides a robust networking foundation for container networking. We also enable the ingress controller support in Cilium to facilitate applications that still require an ingress resource, instead of the more modern Gateway API.

Longhorn

Longhorn provides persistent storage capabilities for Kubernetes, using local drive storage in nodes. It is used for basic storage needs in this solution, serving storage only from the control plane nodes. Typically a dedicated external storage solution will be used for the actual AI workloads. Longhorn is used to serve the firmware files for the Bluefield network adapters and, if no other CSI is available, the persistent volumes for the RunAI control plane.

NVIDIA Network Operator

The NVIDIA Network Operator manages Networking related components in order to enable Fast networking, RDMA and Magnum IO™ GPUDirect for workloads in a Kubernetes cluster. The Goal of Network Operator is to manage all networking related components to enable execution of RDMA and GPUDirect RDMA workloads in a kubernetes cluster including:

- Mellanox Networking configuration to enable advanced features
- Kubernetes device plugins to provide hardware resources for fast network
- Kubernetes secondary network for Network intensive workloads

MetalLB

MetalLB provides load-balancing services for bare metal clusters. It can advertise load-balanced IP addresses via ARP directly on cluster node interfaces or it can advertise BGP routes to existing BGP routers in the network. MetalLB is used for external access to the Nginx ingress service, as well as being available for other applications in the cluster to use.

KGateway

KGateway provides Kubernetes Gateway API services to the cluster. It (alongside Istio) has proven to be the most scalable implementation of the Gateway API and includes a number of AI-specific capabilities, such as inference routing that is aware of the GPU load.

Prometheus Operator

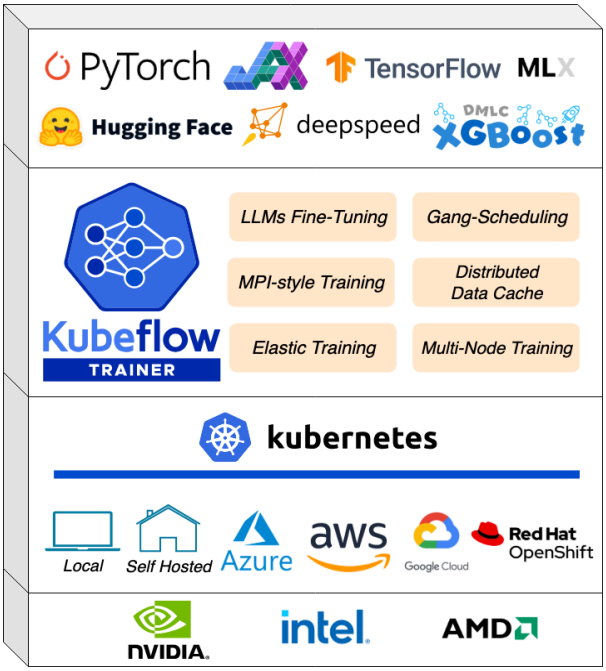
The Prometheus Operator handles deployments of the Prometheus stack for Kubernetes. NVIDIA Run:ai deploys Prometheus into its own namespace through the operator. Prometheus then collects metrics from all components in the Kubernetes cluster, including the AI accelerators. These metrics power the analytics dashboards in NVIDIA Run:ai.

NVIDIA GPU Operator

The NVIDIA GPU Operator automates the management of all NVIDIA software components needed to provision GPUs. These components include the NVIDIA drivers (to enable CUDA), Kubernetes device plugin for GPUs, the NVIDIA Container Runtime, automatic node labelling, DCGM based monitoring and others. The GPU Operator can enable RDMA capabilities for GPU-to-GPU communications, as well as enable MIG mode for slicing a GPU into multiple instances.

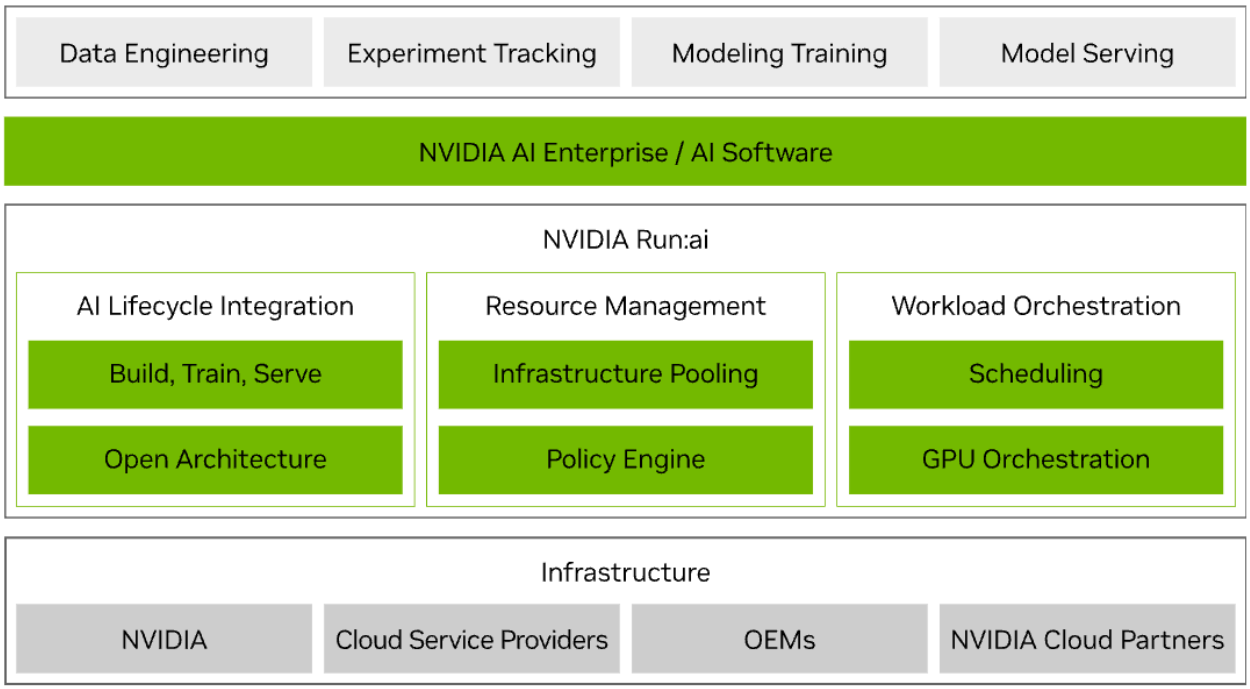
Kubeflow Training Operator

Kubeflow Trainer is a Kubernetes-native project designed for large language models (LLMs) fine-tuning and enabling scalable, distributed training of machine learning (ML) models across various frameworks, including PyTorch, JAX, TensorFlow, and others. You can integrate other ML libraries such as HuggingFace, DeepSpeed, or Megatron-LM with Kubeflow Trainer to run them on Kubernetes. Kubeflow Trainer enables you to effortlessly develop your LLMs with the Kubeflow Python SDK, and build Kubernetes-native Training Runtimes using Kubernetes Custom Resource APIs.



NVIDIA Run:ai

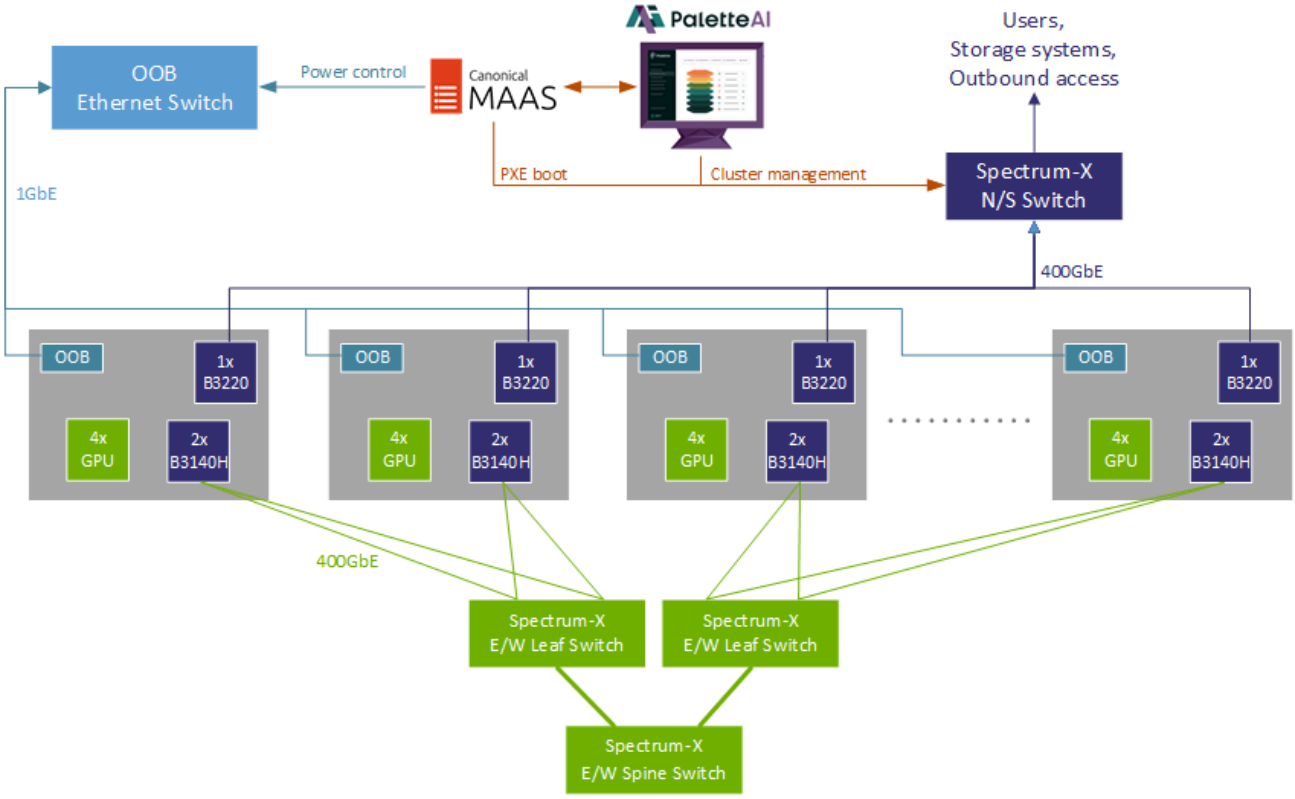
NVIDIA Run:ai accelerates AI and machine learning operations by addressing key infrastructure challenges through dynamic resource allocation, comprehensive AI life-cycle support, and strategic resource management. By pooling resources across environments and utilizing advanced orchestration, NVIDIA Run:ai significantly enhances GPU efficiency and workload capacity. With support for public clouds, private clouds, hybrid environments, or on-premises data centers, NVIDIA Run:ai provides unparalleled flexibility and adaptability.



Solution Configuration

This section describes the overall architecture of the solution, the resources required and the configuration of each component.

Architecture



The diagram shows the Enterprise AI Factory reference architecture with these core technologies:

- **Supermicro ServerServer SYS-212GB-FNR** to support up to 4 GPUs per server
- **NVIDIA RTX PRO 6000 Blackwell or NVIDIA H200 NVL GPUs** for premier AI performance
- **Bluefield-3 B3140H SuperNICs** for 200-400GbE per GPU of GPU-to-GPU RDMA traffic
- **Bluefield-3 B3220 DPU** for 2×200GbE of north-south traffic, and PXE boot for installation
- **Spectrum-X SN5610 Ethernet switches** for unparalleled RDMA-over-Converged-Ethernet (RoCE) performance, up to 1.6× faster than off-the-shelf Ethernet.
- **Spectro Cloud PaletteAI** to automate deployment & management of Kubernetes AI stacks
- **Canonical MAAS** to automate deployment of OS software onto bare metal hardware

Hardware resources

The hardware resources guidance detailed here provides an overview of component options that align with the NVIDIA AI Enterprise Software Reference Architecture for RTX 6000 Pro Blackwell Server Edition. This is not an exhaustive list but an example of a known-good working solution. The reference architecture contains the following components:

Parameter	System Configuration
GPU	NVIDIA RTX PRO 6000 Blackwell Server Edition
GPU configuration	GPUs are balanced across CPU sockets and root ports: • Inferencing: 2×, 4× and 8× GPUs per server • Fine-tuning: 8× GPUs per server • HPC: 4× or 8× GPUs per server • Physical: 4× or 8× GPUs per server
CPU	Intel 4th, 5th and 6th Gen Xeon® Scalable processors; AMD 4th and 5th Gen EPYC™ processors
CPU sockets	At least one CPU socket
CPU speed	At least a 2.0.GHz CPU clock
CPU cores	At least 8 physical CPU cores per GPU: • For MIG configurations: at least 2 CPU cores per MIG instance • For OS kernel or virtualization, 1 additional core per GPU
System memory	Minimum 128 GB of system memory per GPU. The system memory should be evenly distributed across all CPU sockets, and all memory channels should be populated with at least one DPC.
DPU	One NVIDIA® BlueField®-3 DPU per server for the North-South network
PCI express	At least one Gen5 x16 link per GPU
PCIe topology	Balanced PCIe topology with GPUs spread evenly across CPU sockets and PCIe root ports. • NIC and NVMe drives should be under the same PCIe switch or PCIe root complex as the GPUs. • A PCIe switch might not be needed for low-cost inference servers; direct-attach to CPU is best if possible
PCIe switches	Gen5 PCIe switches as needed (where additional link fanout is not required, direct attach is best).
NIC Speed (E-W)	At least 200Gbps per GPU. For maximum performance, 400Gbps per GPU is recommended
Local storage	Inference Servers: Minimum 1 TB NVMe drive per CPU socket. Training / DL Servers: Minimum 2 TB NVMe drive per CPU socket. HPC Servers: Minimum 1 TB NVMe drive per CPU socket
Remote management	SMBPBI over SMBus (OOB) protocol to BMC MCTP over USB (OOB) protocol to BMC PLDM T5–enabled SPDM-enabled
Security	TPM 2.0 module, Secure Boot enabled

Bluefield-3 adapters support both InfiniBand and Ethernet. This reference architecture focuses on the Ethernet option as it typically aligns better with existing enterprise networks. Using NVIDIA Spectrum-X SN5600 switches, a 1.6× RDMA performance increase is seen over off-the-shelf Ethernet switches due to the RoCE accelerations, adaptive routing and congestion control that the Spectrum-X Ethernet provides.

The Supermicro SuperServer SYS-212GB-FNR is a single-CPU socket server that supports a maximum of 4 GPUs. As this server uses the latest Intel Xeon Granite Rapids generation, this CPU boasts 136 PCIe lanes instead of the 80 found in Intel Xeon Emerald Rapids. With 136 PCIe 5.0 lanes, all 4 GPUs, 3 Bluefield-3 NICs and 6 NVMe drives can be directly attached to the CPU, making a second CPU socket unnecessary. This also simplifies the solution as we don't have to deal with NUMA nodes either.

Using a single CPU socket and PCIe GPUs provides a cost-effective entry point into serious AI research and/or production work. Multiple SYS-212GB-FNR servers can be combined to handle larger workloads. While the very largest datasets or the very highest training/deep-learning performance will require stepping up to an 8–GPU/server-based system, such 8–GPU systems represent a different price point and different power & cooling requirements altogether.

With the SYS-212GB-FNR, three different layouts are possible:

- 1–2–2–200 (1 CPU, 2 GPUs, 1 east-west NIC, 1 north-south NIC, 200GbE/GPU)
- 1–2–3–400 (1 CPU, 2 GPUs, 2 east-west NICs, 1 north-south NIC, 400GbE/GPU)
- 1–4–3–200 (1 CPU, 4 GPUs, 2 east-west NICs, 1 north-south NIC, 200GbE/GPU)

The reference solution uses 1 SuperNIC for every 2 GPUs, providing a net 200GbE of bandwidth of GPU-to-GPU communication across nodes. Optionally, it is possible to get double the bandwidth by installing 2 GPUs and 2 SuperNICs. However the next generation of Bluefield-4 DPUs and ConnectX-8 SuperNICs will deliver 800Gbps connections, which also nets to 400GbE/GPU in a 2:1 ratio. Hence we would recommend staying with the 2:1 ratio for PCIe-based GPU solutions. For SXM-based 8–GPU solutions in HGX systems, the reference architecture does use a 1:1 ratio of GPU to SuperNIC, but again that is a different pricepoint.

The full system configuration for the GPU worker nodes in each layout is as follows:

Component	1-2-2-200	1-2-3-400	1-4-3-200
Chassis	SYS-212GB-FNR (2U)	SYS-212GB-FNR (2U)	SYS-212GB-FNR (2U)
CPU	Intel Xeon 6774 (64c, 2.5GHz base, 3.6Ghz turbo, 8-core PCT)	Intel Xeon 6774 (64c, 2.5GHz base, 3.6Ghz turbo, 8-core PCT)	Intel Xeon 6774 (64c, 2.5GHz base, 3.6Ghz turbo, 8-core PCT)
RAM	256 GB minimum	256 GB minimum	512GB minimum
GPU	2x RTX PRO 6000 Blackwell or 2x H200 NVL	2x RTX PRO 6000 Blackwell or 2x H200 NVL	4x RTX PRO 6000 Blackwell or 4x H200 NVL
Network Adapters	1x Bluefield-3 B3140H 1x Bluefield-3 B3220	2x Bluefield-3 B3140H 1x Bluefield-3 B3220	2x Bluefield-3 B3140H 1x Bluefield-3 B3220
NVMe drives (E1.S)	1x 4TB drive	1x 4TB drive	1x 4TB drive
NVMe drives (M.2)	2x 1TB OS drive (RAID1)	2x 1TB OS drive (RAID1	2x 1TB OS drive (RAID1

The reference architecture does not depict Kubernetes control plane nodes, but they are necessary. Typically the GPU worker nodes require all the design work in AI architectures, hence that is what this document focuses on.

For control plane nodes we recommend a scale-up strategy, not a scale-out strategy, as etcd replication complexity increases with every control plane node added. A 3-node control plane of 4-core, 8 GB RAM servers each is sufficient for a minimum spec control plane. The control plane nodes can be virtual machines, as long as the virtualization platforms are supported by MAAS for power management: VMware, Proxmox, OpenStack, LXD and Virsh.

Note that you always need 1 extra server available in MAAS to facilitate rolling control plane repaves for OS/K8S upgrades and reconfiguration. So when using a 3-node control plane, a 4th server with identical specs must be present but unused in the same resource pool in MAAS. This is not the case for worker nodes, where a "contract-first" strategy can be selected that does not require any spare hardware.

Adjust the specs upwards depending on the total number of nodes (control plane + workers) that will be part of the cluster. The following table provides some guidance on sizing control plane nodes:

# of worker nodes	# of namespaces	CPU cores	Memory (GB)
10	100	4	8
25	500	4	16
125	1000	8	32
250	2000	16	64
500	4000	32	128
1000	4000	64	256

The table above assumes using at least 3 control plane nodes for a cluster.

Software resources

The software guidance detailed here provides recommended versions for the used components in this reference architecture:

Component	Software product	Recommended version
Kubernetes Management	Spectro Cloud PaletteAI	1.0.0
	Spectro Cloud Palette	4.7.29
Bare metal deployment	Canonical MAAS	3.7.0
Operating System	Canonical Ubuntu	22.04 LTS with 6.5 kernel
NVIDIA DOCA software	doca-host	3.2.1–044000–25.10
	Congestion Control (spcx-cc)	3.10
Kubernetes	Palette eXtended Kubernetes	1.33.5
Kubernetes networking	Cilium CNI	1.18.1
Kubernetes storage*	Longhorn	1.9.0
Acceleration Operators	NVIDIA GPU Operator	25.10.1
	NVIDIA Network Operator	25.10.0
Ingress	Envoy (via Cilium CNI)	1.18.1
Gateway API	KGateway	2.1.1
Load Balancing	MetalLB	0.15.2
Monitoring	Prometheus Operator	78.3.0
Serverless Functions	Knative	1.20.0
AI Runtime	Kubeflow Training Operator	1.8.1
	NVIDIA Run:ai	2.23.20

* For basic Kubernetes storage, supporting the Network Operator and NVIDIA Run:ai storage needs. For external storage, you can use the CSI of the chosen storage vendor, or you can use NFS.

Solution sizing

This reference architecture is suitable for a GPU cluster with up to 64 worker nodes, for a total of 256 GPUs. Above that node count, it may be better to consider 8-GPU HGX-based nodes instead.

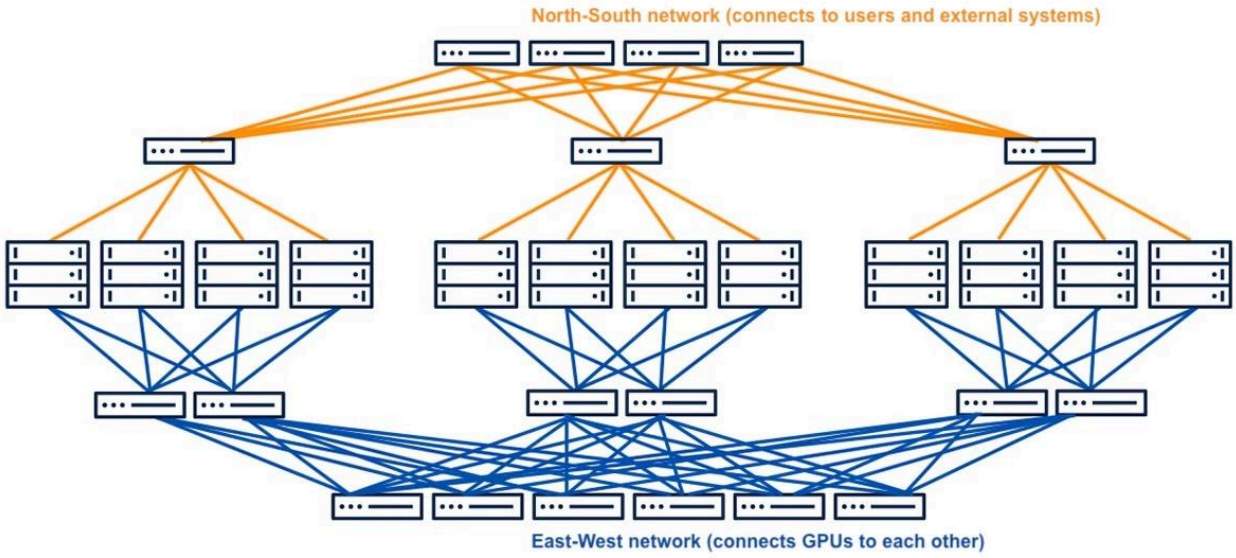
The solution is sized in the following manner:

- 3 control plane nodes, using 6 CPUs and 20GB of RAM per node will suffice for up to 64 workers.
  - A 4th node must be kept in spare to facilitate control plane repaves
- Up to 64 bare metal worker nodes in the cluster
  - Each node can have 2 or 4 GPUs and 1 or 2 SuperNICs
  - With RTX PRO 6000 Blackwell GPUs, that is a maximum combined 24TB of VRAM
  - With H200 NVL GPUs , that is a maximum combined 35TB of VRAM.
- Kubernetes pod CIDR configured to 100.64.0.0/12
  - This allows for a maximum of 4096 sub-CIDRs of /24 each, one for each node.
- Kubernetes service CIDR configured to 100.80.0.0/14
  - This allows for 256K service IP addresses.
- Maximum number of pods configured to 250
  - Allowing 250 * 64 = 16.000 pods to be active on worker nodes
- Overhead pods:
  - 131 pods for controllers, agents and other non-daemonset overhead
  - 25 pods per node for daemonsets
  - Total overhead (64*25)+131=1.731, leaving 14.269 pods for workloads.

Typically AI clusters don't run a high number of pods in the cluster, so the extra pod capacity is not used in higher node counts. But for a smaller cluster of only a few worker nodes, the increased pod capacity is useful to handle the pods consumed by RunAI and other control services.

Network configuration

Networking in AI clusters requires special consideration, as it is not comparable to generic enterprise networks.



AI clusters have distinct networking requirements:

- **A dedicated east-west network**, also called a GPU Fabric, to allow GPUs to directly exchange information via RDMA. This network needs to be non-blocking, high-bandwidth and ideally lossless.
- **A high-bandwidth north-south network**, as typically AI clusters need to consume large amounts of external data. High-performance storage systems are typically external to the cluster and thus are accessed via the north-south network.

East-West Networking

Since most communication happening over the GPU Fabric is RDMA, we need the fastest possible RDMA-capable network adapters, such as the NVIDIA ConnectX-based SuperNIC. For this reference architecture this is the 400Gbps Bluefield-3 B3140H, with the 800Gbps ConnectX-8/Bluefield-4 successor planned for a future version of the architecture.

Connecting these highspeed links together can be done in two ways:

- **InfiniBand**: using the NVIDIA Quantum switches
- **Ethernet**: using the NVIDIA Spectrum-X Ethernet switches

The Bluefield-3 B3140H SuperNICs supports both InfiniBand and Ethernet modes.

GPU-to-GPU communications happens through vast amounts of small 256-byte packets that need to be exchanged with extreme speed and low latency. Traditional 10Gbps Ethernet switches have 100× higher port latency and 4× lower packet forwarding rates than InfiniBand switches, making them unsuitable as GPU Fabrics. However, specialized Ethernet products like NVIDIA Spectrum-X have significantly narrowed that gap:

	InfiniBand (NVIDIA Quantum-2)	**Ethernet** (NVIDIA Spectrum-X)
Forwarding rate	66.5B PPS	33.3B PPS
Total throughput	51.2 Tb/sec	51.2 Tb/sec
In-Network Collectives	NVIDIA SHARP	-
Congestion control	N/A (lossless fabric)	Spectrum-X Ethernet CC

While InfiniBand still reigns supreme, it also comes with higher complexity compared to Ethernet. Since this Reference Architecture does not aim to force InfiniBand upon organizations that don't have the necessary expertise to support it, we will use Spectrum-X Ethernet as it is very close to InfiniBand in terms of raw performance, as well as more cost-effective due to having double the number of 400Gbps ports per switch vs Quantum-2 switches.

If you do want to use InfiniBand, this Reference Architecture fully supports it and only requires minor changes to the NVIDIA Network Operator configuration.

In order to provide a non-blocking network for the GPU Fabric, we need to ensure the same amount of bandwidth is available across the network tiers. This means that for every 2 leaf switches, we need 1 spine switch. 50% of the connectivity of each leaf switch is used to connect to GPUs, the other 50% is used to connect up to the spine switch. For a 64-worker node cluster, that uses all ports of the Spectrum-X SN5610 switches.

The following table shows node count to switchports and switches:

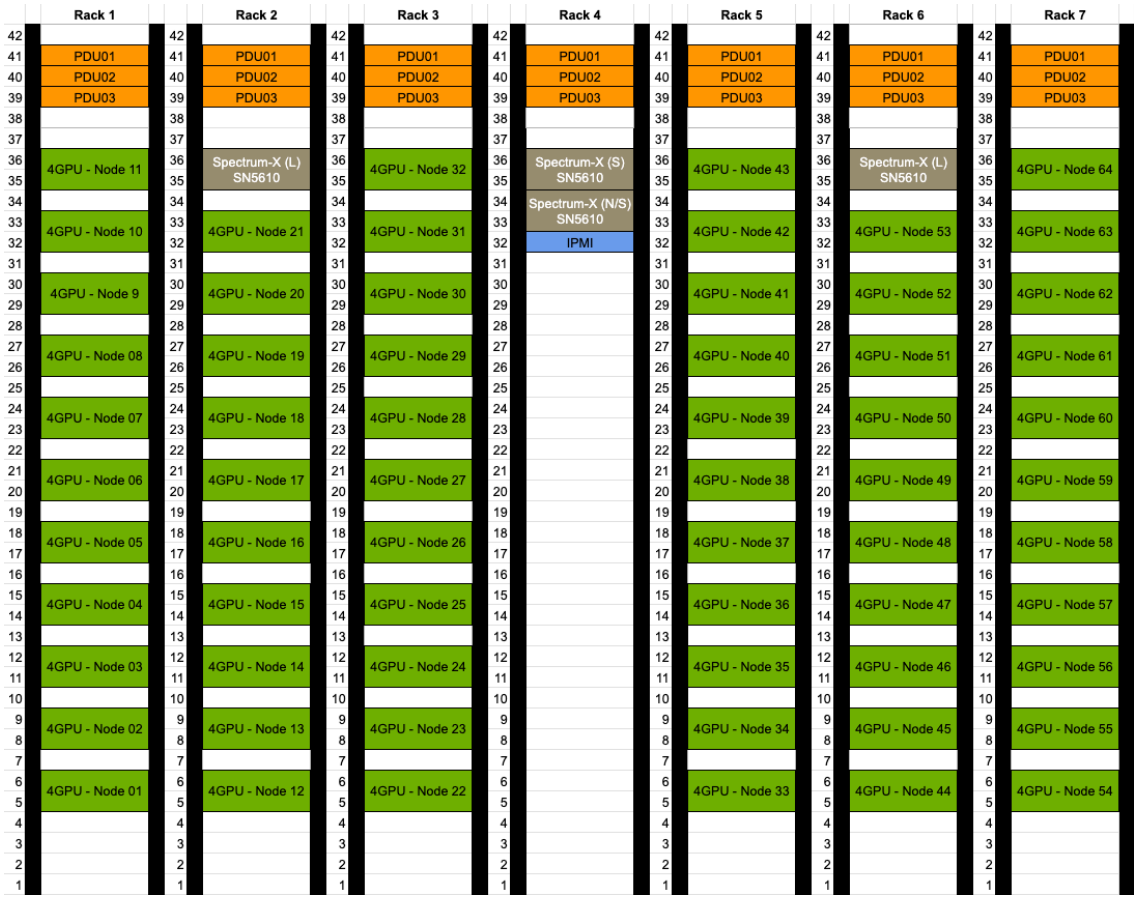
GPU nodes	GPUs/server	B3140H/server	Ports needed (GPU + spine)	# of 128-port leaf switches	# of 128-port spine switches
4	2	1	4 + 0	1	0
8	2	2	16 + 0	1	0
8	4	2	16 + 0	1	0
16	4	2	32 + 0	1	0
32	4	2	64 + 0	1	0
48	4	2	96 + 48	2	1
64	4	2	128 + 64	2	1

Each Spectrum-X SN5610 switch has 64 ports of 800GbE, which support 128 connections of 400GbE (using splitter cables). Ensuring we never use more than 50% of the port capacity for server connections, that means we can use a maximum of 64 connections of 400GbE before needing another leaf switch. For every 2 leaf switches, a spine switch is added which interconnects the other 50% of port capacity. This design ensures you can keep scaling the GPU Fabric, up to a maximum of 4.096 GPU nodes, before needing to move from a 2-tier to a 3-tier network design.

Another aspect of this network is the cost, power usage and heat dissipation of the network interconnects. For comparison, see the table below comparing the cost and power draw of four different options to connect one 800G port on the Spectrum-X switch to the 400G port of two B3140H SuperNICs in a GPU server:

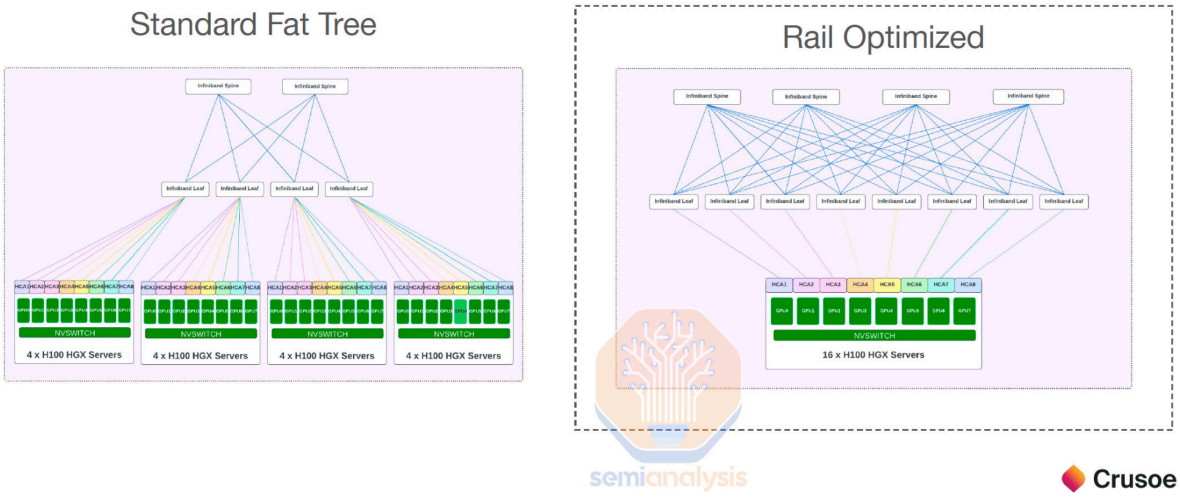
Type	**Part Number**	**Max length**	**Power draw**	**MSRP**
DAC	O2Q112-800G-CU3	10ft / 3m	0.15W	$249
ACC	O2Q112-800G-AC5	16ft / 5m	2.5-3W	$990
AEC	O2O112-800G-AE7	23ft / 7m	25W	$1689
Fiber (MM)	OSFP-800G-2×SR4 + 2× Q112-400G-SR4	328ft / 100m	32W (14 + 2*9)	$1975

This Reference Architecture recommends a 7 rack approach, with every 32 GPU servers connecting to a Spectrum-X leaf switch in the center (2nd, 6th) rack, using all 64 available 400GbE ports. The leaf switch then connects with 32× 800GbE links to the spine switch in the 4th rack, centered between the two 3-rack GPU setups. This allows maximum use of 10ft/3m passive DAC cables, which should be usable for most servers for the east-west network and about half the servers for the north-south network. This can best be visualized with the following physical rack diagram:



This design consumes around 40kW per rack for racks 1–3 and 5–7.The main building block is a set of 3 racks with 32 GPU servers and 1 Spectrum-X leaf switch for the GPU Fabric in the middle. For clusters up to 32 GPU servers, you only need the 1 Spectrum-X switch for east-west, and another for north-south. Up to 64 GPU servers, most connections to the Spectrum-X switches can be made with the more cost-effective copper cables. Beyond 64 GPU servers, it may be better to place the Spectrum-X spine and north-south switches at the end of the rack isle, so you can route connections from multiple 3-rack blocks to it. This will require fiber optics due to the longer cable runs.

This design is a fat-tree design, not a rail-optimized design. In a rail-optimized design, every GPU connects to its own leaf switch, while in a fat-tree design, all GPUs of a server connect to the same leaf switch.



The advantage of a rail-optimized design is that each GPU can talk to a higher amount of distant GPUs through only 1 switch hop, compared to a fat-tree design. However, the rail-optimized design has a number of assumptions that aren't always in place:

- It requires all GPUs in a server to be inter-connected via NVLink, in order to enable GPU A in one server to talk to GPU B in another server over only 1 network hop.
- It requires the use of costly optics, as cable runs from GPUs to switches will be much longer.
- It requires significantly more leaf and spine switches.

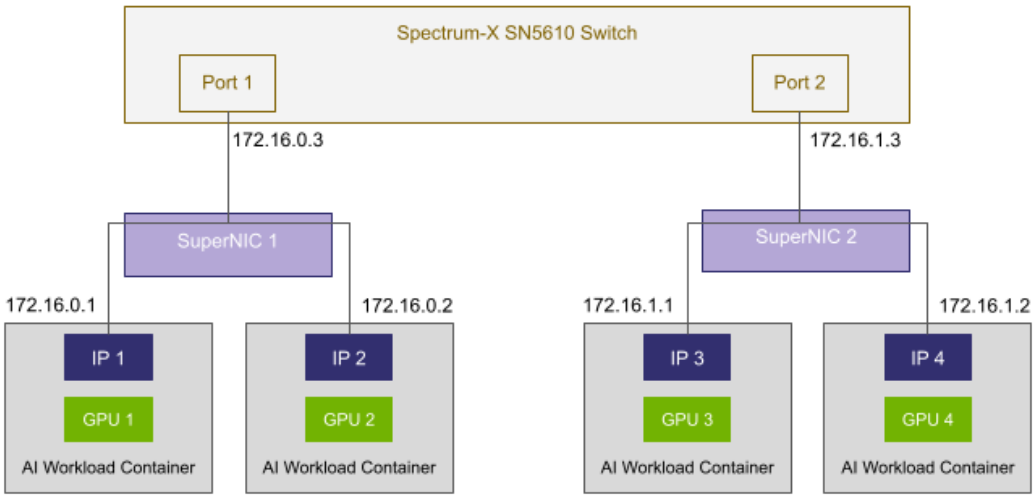
The Supermicro SYS-212GB-FNR does not use the 8-GPU NVLink HGX board, it just provides 4 PCI express slots for regular GPU cards. If you use NVIDIA H200 GPUs, NVLink is available and officially the H200 supports 4-way NVLink, but due to the 4 GPUs being seated on opposite sides of the server, 4-way NVLink is not possible. Only pairs of two H200's can be connected with a 2-way NVLink bridge. The RTX PRO 6000 Blackwell does not support NVLink at all. Also, this server shares a SuperNIC for every 2 GPUs, which means you can't really dedicate a leaf switch to each GPU anyway.

That means the rail-optimized design will not deliver as many benefits in this case, and isn't worth the additional cost of extra switches and more expensive optics. By using a fat-tree design, we can almost exclusively use copper cabling which is both more economical and more reliable, as copper interconnects suffer from fewer flapping issues and failures, which is a major problem for all high-speed interconnects using optics. This means less downtime for the cluster as a whole due to link failure, which is the most common problem encountered in these clusters and a major source of low cluster productivity.
In order to share a SuperNIC between two GPUs, we need to consider the requirements that poses to container networking:

- Spectrum-X typically uses a Layer 3 routed network, in order to avoid the scaling issues of large, flat Layer 2 networks where broadcast messages introduce unwanted traffic.
- To this end, the default Spectrum-X design uses very small /31 subnets (2 IP addresses) on every switchport, with one address assigned to the switchport and the other address assigned to the SuperNIC it's connected to.

While this design is suitable for non-Kubernetes environments, it poses some challenges for Kubernetes clusters. It can still be used in single-tenant environments where every GPU has a dedicated SuperNIC, but the limitation of 1 IP address per SuperNIC does not work for multi-tenant environments or 2:1 ratios of GPUs to SuperNICs.
There are two approaches to sharing an InfiniBand or RoCE device:

1. Using the **RdmaSharedDevice** plugin to allow Kubernetes pods to share a single RDMA device from the host, then using the macvlan plugin (for Ethernet) or IPoIB plugin (for InfiniBand) to define an additional network interface in the pod that leverages the physical SuperNIC on the host so that the container can access the external network as if it was using a host adapter.
2. Using **SR-IOV** to create Virtual Functions (VFs) on the SuperNIC, which can then be directly passed to containers that need it. This ensures isolation at the hardware level and is suitable for multi-tenancy scenarios that favor isolation over absolute performance. We have measured the performance degradation of this approach to be about 20% (compared to the RdmaSharedDevice plugin approach). Also, the SR-IOV approach is not recommended for GPUDirect Storage use cases, which require IOMMU to be disabled.



In both cases, the NVIDIA IPAM plugin is used to assign IP addresses to the additional network interface in a manner that is consistent with the Spectrum-X IP addressing scheme. When not using NVIDIA MIG to split the GPUs into even smaller instances, we recommend generating a Spectrum-X addressing scheme that uses /30 subnet prefixes per switchport. This gives you 4 usable addresses per link, one used by the switchport and 3 available to be used by VFs on the host.

When using NVIDIA MIG to have even more GPU instances available, the Spectrum-X per-port subnet prefix will need to be aligned to the number devices that can simultaneously communicate. For example, when using the "all-2g.48gb" MIG configuration to split each RTX PRO 6000 Blackwell into two 48GB GPU instances, a total of 4 GPU workloads will share the same NIC, requiring a total of 5 IP addresses (4 GPU containers plus 1 IP address for the switchport itself), which requires a /29 subnet.

Storage configuration

The cluster requires some basic storage functionality to serve 6 persistent volume claims:

PVC	Namespace	Size	Access mode
nic-fw-storage-pvc	nvidia-network-operator	10Gi	RWX
data-runai-backend-nats-0	runai-backend	3Gi	RWO
data-runai-backend-nats-1	runai-backend	3Gi	RWO
data-runai-backend-nats-2	runai-backend	3Gi	RWO
data-runai-backend-postgresql-0	runai-backend	8Gi	RWO
data-runai-backend-thanos-receive-0	runai-backend	100Gi	RWO

This design uses Longhorn 1.9.1 to provide this functionality as it can run with a low amount of overhead. The following specific parameters can be used to run Longhorn for basic storage on the control plane nodes, with a reduced CPU guarantee of 3%:

Parameter	Value

`global:`
`  tolerations:`
	`- effect: NoSchedule`
	`  key: node-role.kubernetes.io/control-plane`
	`  operator: "Exists"`

`defaultSettings:` `  taintToleration:`	`"node-role.kubernetes.io/control-plane:NoSchedule"`

`persistence:`
`  defaultNodeSelector:`
`  enable:`
	`true`

`persistence:`
`  defaultNodeSelector:`
`  selector:`
	`"storage"`

`Longhorn Setting` `  → taint-toleration`	`"node-role.kubernetes.io/control-plane:NoSchedule"`

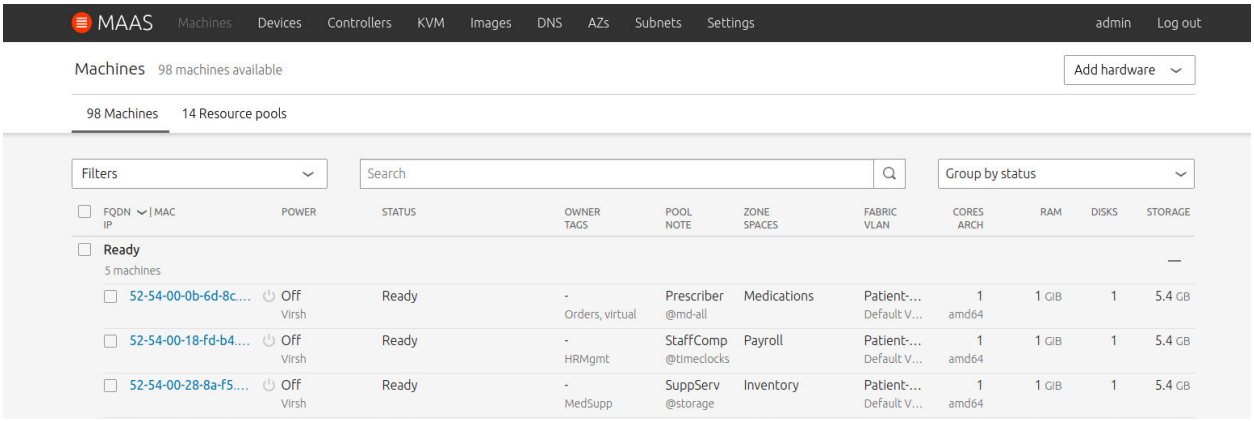
`Longhorn Setting`
`  →`
`guaranteed-instance-manager-cpu`
	`"3"`

Our platform combines this with the following annotation for the control plane nodes:
`node.longhorn.io/default-node-tags: '["storage"]'`

Canonical MAAS configuration

Spectro Cloud natively supports Canonical MAAS for bare metal cluster deployment. The flexibility of MAAS to configure networking and storage for bare metal servers is particularly helpful for the AI use case.

We will use MAAS to ensure consistent configuration across the nodes, allowing enough flexibility to fit any environment.



We will use the following features of MAAS to aid in the deployment of the AI clusters:

Feature	Configuration	Remarks
Resource Pools	control-plane and worker-pool resource pools. Add more as necessary.	Separate pools so hardware specs can be optimized for each function.
AZs	Align Availability Zones to datacenter layout	If your hardware is located in separate racks/aisles/rooms in the datacenter, you can create an AZ in MAAS for each. The cluster will be spread across the selected AZs.
Subnets	PXE subnet Management network North-south network	Align subnets to network layout. The subnets for PXE deployment, management and north-south network should be configured in MAAS.
DNS	metal.[company DNS zone]	Recommended approach for DNS is to create a delegated DNS subzone in the corporate DNS infrastructure, delegating the subzone to MAAS.

Spectro Cloud PaletteAI configuration

This section shows the detailed configuration of each layer in the Cluster Profile for this reference architecture.

This reference architecture can be imported by retrieving the exported cluster profiles from this Github Gist and saving the JSON files locally. They can then be imported one by one into Palette by selecting **Import Cluster Profile**.

The following pages describe changes made to each layer in the profiles.

Ubuntu layer (AI-RA-Infra-MaaS profile)

Pack Type	OS
Registry	Palette Registry (OCI)
Pack Name	Ubuntu
Pack Version	22.04
Values	Modified from default

By default this layer does not contain any files or commands, so all of the configuration in this layer is custom for this reference architecture.

In the `kubeadmconfig.preKubeadmCommands` section:

- Add the node's IP address on the management network to `/etc/default/kubelet`, to ensure this IP becomes the InternalIP for this cluster node
- Run `update-ca-certificates` so that custom CA certificates are trusted. If Palette is used as the OIDC Identity Provider in the Kubernetes layer, the issuing CA for the certificate that Palette is using needs to be installed on all nodes. If you are using your own OIDC Identity Provider and use private certificates, you need to install the issuing CA's certificate on all nodes.
- Restart containerd service to let new configurations take effect

In the `kubeadmconfig.postKubeadmCommands` section:

- Remove the kube-proxy daemonset and configmap from the cluster for Cilium
- For control plane nodes:
  - Remove the Node Preparation taint
  - Annotate the node with Longhorn storage tags
- For worker nodes:
  - Trigger the first run on the Node Preparation script

In the `kubeadmconfig.files` section:

- Blacklist the open source nouveau GPU driver
- Configure containerd to remove the memory and number of processes limits. Also increase the open files limit to the maximum value of 1048576.
- Tune the kubelet configuration to enable CPU Manager, set the pod limit to 250 and register all nodes with a Node Preparation taint. This will result in a `spectrocloud.com/nodeprep:NoSchedule` taint on every node that joins the cluster, to ensure that no user workloads get scheduled on new nodes until node preparation has completed.
- Import a custom CA certificate (optional). The provided certificate in the profile is an example and should be replaced by your custom CA certificate if you have one.
- Place a script on the system that performs Spectrum-X node preparation:
    - Installs version 3.1 of the DOCA package and other required software packages
    - Installs the NVIDIA Congestion Control software for Spectrum-X
    - Updates the firmware of Bluefield-3 SuperNIC and DPU adapters
    - Configures firmware settings of Bluefield-3 SuperNIC and DPU adapters
    - If SR-IOV is enabled by the user:
        - Configures Grub for IOMMU passthrough and RDMA isolation
        - Creates Virtual Functions on each boot
        - Sets unique hardware addresses for Virtual Functions if the type is InfiniBand
    - If SR-IOV is disabled by the user (default):
        - Optionally disables ACS for higher performance with GPUDirect Storage

Kubernetes layer (AI-RA-Infra-MaaS profile)

Pack Type	Kubernetes
Registry	Palette Registry (OCI)
Pack Name	Palette eXtended Kubernetes
Pack Version	1.33.5
Values	Modified from default

The modified aspects of the configuration of this layer are as follows:

- The OIDC Identity Provider is set to Palette. You should use an OIDC provider with AI clusters, so that users can be given access to specific namespaces and authenticate through SSO. You can change this setting to use your own OIDC provider if so desired.
- The Pod CIDR is set to a profile variable for easy configuration. This variable defaults to `100.64.0.0/12` and is read-only. Refer to the Solution Sizing section on changing this.
- The Service CIDR is set to a profile variable for easy configuration. This variable defaults to `100.80.0.0/16` and is read-only. Refer to the Solution Sizing section on changing this.
- The `--config-dir` extra argument for kubelet is added, enabling the use of the `/etc/kubernetes/kubelet.conf.d` directory for drop-in configuration files (which is where the OS layer writes the kubelet tuning file)

Cilium layer (AI-RA-Infra-MaaS profile)

Pack Type	Network
Registry	Palette Registry (OCI)
Pack Name	Cilium
Pack Version	1.18.1-rev1
Values	Set using Pack Presets and further modifications

The following pack presets were selected:

IPAM mode	Kubernetes
Cilium Operator	For Multi-Node Cluster
Kube-proxy replacement	Replace kube-proxy with eBPF
Pod networking	Use VXLAN Overlay
VMO Compatibility	Disable
VMO - Bridge Interface	Autodetect Cilium Interface
Loadbalancer mode	No XDP Acceleration

Cilium is configured to leverage eBPF to replace kube-proxy for better performance in clusters with large amounts of service resources. The pod network still runs over a VXLAN overlay for maximum compatibility.

Additional changes made to the Cilium layer are:

- The `cni.exclusive` option is set to false, since we need compatibility with Multus.
- The `ingressController.enabled` and `ingressController.default` options are set to true, to leverage Cilium Envoy for ingress resources. Nginx Ingress is sunsetting, so we will use the ingress support in Cilium for those applications that still require ingress.
- The `ingressController.loadbalancerMode` option is set `shared`, in order to more closely mimic an Nginx ingress solution using a single load balancer IP address.
- The `operator.tolerations` list has been expanded with the taint for node preparation.

Longhorn layer (AI-RA-Infra-MaaS profile)

Pack Type	Storage
Registry	Palette Registry (OCI)
Pack Name	Longhorn
Pack Version	1.9.0
Values	Modified from default

The modified aspects of the configuration of this layer are as follows:

Adjustments to allow Longhorn to run on control plane nodes:

- A toleration for the `node-role.kubernetes.io/control-plane:NoSchedule` taint is added to the `global.tolerations` parameter. This allows Longhorn pods to be scheduled onto control plane nodes, providing minimal basic storage functionality.
- The `defaultSettings.taintToleration` parameter is set to `"node-role.kubernetes.io/control-plane:NoSchedule"`, in order to allow system-managed Longhorn components to run on the control plane nodes.
- An additional manifest is defined to reconfigure a Longhorn `Setting` parameter that also needs to be adjusted in order to let Longhorn run on control plane nodes:

```
apiVersion: longhorn.io/v1beta2
kind: Setting
metadata:
  name: taint-toleration
  namespace: longhorn-system
value: "node-role.kubernetes.io/control-plane:NoSchedule"
```

Adjustment to the Longhorn persistence settings:

- The `persistence.defaultClass` parameter is set to false, ensuring that Longhorn is not set as the default storage class in the system.

  A third party CSI is expected to be added as the default storage class, in line with the external storage solution the customer will use.

- The `persistence.defaultNodeSelector.enable` parameter is set to true, with the `persistence.defaultNodeSelector.selector` parameter set to `"storage"`. This ensures that persistent volumes requested from Longhorn are only stored on nodes annotated with the `node.longhorn.io/default-node-tags='["storage"]'` annotation. In the Ubuntu layer, the control plane nodes are given this annotation.

An adjustment to the percentage of CPU cores that Longhorn reserves for itself:

- Most GPU servers have a large amount of CPU cores and Longhorn's default setting of reserving 12% of the CPU cores for itself can be wasteful for a non-default CSI that is meant to serve only basic needs in this cluster. Hence we can downtune this parameter to a more reasonable 2% by defining this manifest:

```
apiVersion: longhorn.io/v1beta2
kind: Setting
metadata:
  name: guaranteed-instance-manager-cpu
  namespace: longhorn-system
value: "2
```

nodeprep-controller layer (AI-RA-Infra-MaaS profile)

Pack Type	Manifest
Registry	N/A
Pack Name	N/A
Pack Version	N/A
Install order	0

This manifest layer installs the nodeprep-controller, which monitors nodes for the `spectrocloud.com/nodeprep:complete` label, indicating that node preparation has successfully completed. When this label is applied, the nodeprep-controller removes the `spectrocloud.com/nodeprep:NoSchedule` taint from the node, allowing workloads to get scheduled by Kubernetes onto the node.

Network Operator layer (AI-RA-Infra-MaaS profile)

Pack Type	System App
Registry	Palette Community Registry (OCI)
Pack Name	NVIDIA Network Operator
Pack Version	25.10.0
Install order	0
Values	Modified from default

The modified aspects of the configuration of this layer are as follows:

- `sriovNetworkOperator.enabled` is set to a cluster profile variable for easy configuration.
- `maintenanceOperator.enabled` and `maintenance-operator-chart.operatorConfig.deploy` are set to `true` since we will leverage this operator for Day 2 configuration of NVIDIA network interfaces.
- `operator.useDTK` is set to `false` since this is not an OpenShift cluster.
- `operator.ofedDriver.initContainer.enable` is set to `false` since Palette will install the DOCA drivers on the host OS during node preparation.

network-operator-config layer (AI-RA-Infra-Spectrum-X or AI-RA-Infra-Quantum-X profile)

Pack Type	Manifest
Registry	N/A
Pack Name	N/A
Pack Version	N/A
Install order	10

This manifest layer defines the NVIDIA Network Operator configuration for the system. It defines 6 manifests that control different aspects of the NVIDIA Network Operator functionality:

- The **nic-cluster-policy** manifest defines a `NicClusterPolicy` resource that:
  - Deploys the NIC Configuration Operator, used for Day 2 management of Bluefield-3 and ConnectX network adapters
  - Deploys and configures the RDMA Shared Device Plugin (unless SR-IOV is enabled)
  - Deploys the NVIDIA IPAM plugin, which provides IP address assignment in line with Spectrum-X topologies
  - Deploys the CNI plugins package
  - Deploys the Multus CNI
  - Optionally deploys the IPoIB CNI when used with InfiniBand
- The **nic-fw-source** manifest defines a `NicFirmwareSource` resource that retrieves the desired firmware file (typically a BFB) from a user-defined URL and stores it on a Longhorn persistent volume. We can use the Canonical MaaS TFTP server to host this content in airgapped environments.
- The **nic-fw-template** manifest defines a `NicFirmwareTemplate` resource that can validate (and optionally update) the firmware version of the selected network adapters to the given firmware content. It can be used for Day 2 management as you can selectively upgrade the NIC firmware on specific nodes through this resource.
- The **nic-config-template** manifest defines a `NicConfigurationTemplate` resource that configures the runtime settings of the Bluefield-3 SuperNIC adapters for maximum GPU fabric performance.
- The **nic-ipam** manifest defines `IPPool` or `CIDRPool` resources for IP address assignment to AI container workloads that require RDMA acceleration through either an RDMA shared device or an RDMA-enabled SR-IOV Virtual Function:
  - A `CIDRPool` resource is used for larger scale Spectrum-X GPU fabrics, as this resource type allows the IP address and routing configurations across the cluster to line up with the Spectrum-X switchport configuration.

    - An `IPPool` resource can be used on InfiniBand networks (both east-west and north-south) and on Ethernet north-south networks, where typically a more simple flat single-subnet layer 3 IP network is used.
- The **nic-networks** manifest defines `HostDeviceNetwork`, `MacvlanNetwork`, `IPoIBNetwork`, `SriovNetwork` or `SriovIBNetwork` resources for secondary RDMA-accelerated networks that AI container workloads can leverage:
    - A `HostDeviceNetwork` resource is used when there is a 1:1 ratio of GPUs to SuperNICs. This network type passes the entire SuperNIC through to the container, without requiring SR-IOV to be enabled.
    - A `MacvlanNetwork` (for Ethernet) or `IPoIBNetwork` (for InfiniBand) resource is used when there are more GPUs than SuperNICs and sharing of the SuperNIC becomes mandatory. These approaches also do not require SR-IOV. These resource types can also be leveraged for RDMA-accelerated access to the north-south network, for example to access accelerated storage.
    - An `SriovNetwork` (for Ethernet) or `SriovIBNetwork` (for InfiniBand) resource is used when SR-IOV is enabled and VF-based devices need to be passed through to the container.

Configuration of the **nic-networks** manifest is critical and supports many scenarios. Please refer to the NVIDIA Network Operator deployment examples for more information.

MetalLB layer (AI-RA-Core profile)

Pack Type	Load balancer
Registry	Palette Registry (OCI)
Pack Name	MetalLB (Helm)
Pack Version	0.15.2
Install order	20
Values	Modified from default

The modified aspects of the configuration for the Descheduler layer are as follows:

- `configuration.ipaddresspools.first-pool.spec.addresses` is set to a cluster profile variable for easy configuration.
- `configuration.l2advertisements.default.spec.interfaces` is set to a cluster profile variable for easy configuration.

KGateway layer (AI-RA-Core profile)

Pack Type	Load balancer
Registry	Palette Registry (OCI)
Pack Name	kgateway
Pack Version	2.1.1
Install order	20
Values	Default

This pack is installed without modifications.

NVIDIA GPU Operator layer (AI-RA-Core profile)

Pack Type	AI
Registry	Palette Registry (OCI)
Pack Name	NVIDIA GPU Operator
Pack Version	25.10.1
Install order	20
Values	Modified from default

The modified aspects of the configuration of this layer are as follows:

- `nfd.enabled` is set to `false` as the Node Feature Discovery operator is already installed earlier by the NVIDIA Network Operator.
- `driver.rdma.useHostMofed` is set to `true` as the Node Preparation script installs the DOCA package (which includes the OFED drivers) on the host.

The following options depend on whether GPUDirect Storage will be used:

- `driver.rdma.enabled` is set to `false` by default as the NVIDIA peermem module is considered legacy and the use of the DMA-BUF capability in the Linux kernel is preferred at this point. However you may still need to set this option to `true` when expecting to use GPUDirect Storage, as a number of GPUDirect Storage solutions still depend on the NVIDIA peermem module.
- `gds.enabled` is set to `false` by default but can be changed to `true` if GPUDirect Storage is expected to be used.

Prometheus Operator layer (AI-RA-Core profile)

Pack Type	Monitoring
Registry	Palette Registry (OCI)
Pack Name	Prometheus - Grafana
Pack Version	78.3.0
Install order	20
Values	Modified from default

The modified aspects of the configuration for the Monitoring layer are as follows:

- Since NVIDIA Run:ai wants to use the Prometheus Operator to deploy its own Prometheus instance, the configuration of this pack is adjusted to only install the Prometheus Operator:
  - `alertmanager.enabled` has been changed to `false`
  - `grafana.enabled` has been changed to `false`
  - `nodeExporter.enabled` has been changed to `false`
  - `prometheus.enabled` has been changed to `false`
- The mandatory value for `grafana.adminPassword` has been set to “welcome” (even though Grafana will not be deployed).

runai-backend-prereqs layer (AI-RA-RunAI-Backend profile)

Pack Type	Manifest
Registry	N/A
Pack Name	N/A
Pack Version	N/A
Install order	50

This manifest layer defines the prerequisite Kubernetes resources for deploying the NVIDIA Run:ai Control Plane. It defines 3 manifests that set up the environment:

- The **ns-runai-backend** manifest defines a `Namespace` resource that creates the `runai-backend` namespace. NVIDIA Run:ai requires some resources to be present in the namespace prior to installation, which is why we have to use this approach.
- The **secret-runai-ca** manifest defines 2 `Secret` resources and an `Issuer` resource. This sets up a new Cert-Manager Issuer from a predefined SSL certificate and ensures that this SSL certificate can be passed to the NVIDIA Run:ai installer as the certificate to trust.
- The **secret-pullsecret** manifest defines the `Secret` resource containing the registry credentials to runai.jfrog.io, which is where the NVIDIA Run:ai images are currently served from. NVIDIA is expected to change this at some point to nvcr.io and you will need to use your NGC key instead at that point to access the images.

The provided certificate in the secret-runai-ca manifest is a self-signed certificate. This ensures a functional solution regardless of the chosen FQDN for a self-hosted NVIDIA Run:ai installation. If you prefer to use a trusted SSL certificate, you can either:

- Replace the self-signed certificate in the **runai-predefined-ca** `Secret` with a child-CA certificate issued by your own trusted CA. This ensures that certificates generated by the Issuer are trusted by your clients.
  or;
- Remove the **runai-predefined-ca** `Issuer` and **runai-predefined-ca** `Secret` resources entirely and instead define a **runai-backend-tls** secret that contains the trusted SSL certificate for the FQDN you will publish NVIDIA Run:ai on. If the SSL certificate comes from a private CA, you will also need to populate the CA certificate into the **runai-ca-cert** `Secret` to ensure that NVIDIA Run:ai trusts it.

NVIDIA Run:ai Control Plane layer (AI-RA-RunAI-Backend profile)

Pack Type	Helm chart
Registry	https://runai.jfrog.io/artifactory/cp-charts-prod
Pack Name	control-plane
Pack Version	2.23.20
Install order	60
Values	Modified from default

The modified aspects of the configuration for the NVIDIA Run:ai Control Plane layer are as follows:

- `pack.namespace` is set to `runai-backend`.
- `pack.releaseNameOverride` is defined, overriding the chart's release name from `control-plane` to `runai-backend`.
- `global.domain` is set to a cluster profile variable for easy configuration during deployment.
- `global.ingress.ingressClass` is changed from `nginx` to `cilium` in order to support the use of Cilium as the ingress controller.
- Additional entries are added to `global.ingress.extraAnnotations` to support the Cert-Manager Issuer:
  - `cert-manager.io/issuer: runai-predefined-ca` to define the Issuer to use
  - `cert-manager.io/common-name: "{{.spectro.var.RunAIBackendURL}}"` to ensure, as a best practice, that the CN property of the certificate is populated with the FQDN from the cluster profile variable.
- `global.customCA.enabled` is set to `true`, making NVIDIA Run:ai trust the SSL CA certificate defined in the **runai-ca-cert** `Secret` resource (which was defined in the previous cluster profile layer).

runai-cluster-prereqs layer (AI-RA-RunAI-Cluster profile)

Pack Type	Manifest
Registry	N/A
Pack Name	N/A
Pack Version	N/A
Install order	100

This manifest layer defines the prerequisite Kubernetes resources for deploying the NVIDIA Run:ai Workload cluster. It defines 3 manifests that set up the environment:

- The **ns-runai** manifest defines a `Namespace` resource that creates the `runai` namespace. NVIDIA Run:ai requires some resources to be present in the namespace prior to installation, which is why we have to use this approach.
- The **secret-runai-ca** manifest defines the **runai-ca-cert** `Secret` resource, identical to the resource by the same name in the `runai-backend` namespace. This ensures that the NVIDIA Run:ai workload cluster will trust the SSL certificate of the NVIDIA Run:ai control plane, as this secret is referenced by the installer.
- The **secret-pullsecret** manifest defines the `Secret` resource containing the registry credentials to runai.jfrog.io, which is where the NVIDIA Run:ai images are currently served from. NVIDIA is expected to change this at some point to nvcr.io and you will need to use your NGC key instead at that point to access the images.

Kubeflow Training Operator layer (AI-RA-RunAI-Cluster profile)

Pack Type	AI
Registry	Palette Community Registry (OCI)
Pack Name	Kubeflow Training Operator
Pack Version	1.8.1
Install order	100
Values	Default

This pack is installed without modifications.

Knative Operator layer (AI-RA-RunAI-Cluster profile)

Pack Type	Helm chart
Registry	https://knative.github.io/operator
Pack Name	knative-operator
Pack Version	1.20.0
Install order	100
Values	Default

The helm chart itself is installed without modifications. There are only two metadata changes:

- `pack.namespace` is set to `knative-operator`.
- `pack.releaseNameOverride` is defined, ensuring the chart's release name stays consistent as `knative-operator`.

Two additional manifests are defined alongside the chart:

- A **knative-serving** manifest that defines a `Namespace` resource to create the knative-serving namespace, as well as a `KnativeServing` resource to set up Kourier-based Knative serving.
- A **knative-patch** manifest that defines the resources for a Kubernetes Job that patches the Knative Serving configuration in order to make it compatible with NVIDIA Run:ai.

NVIDIA Run:ai Cluster layer (AI-RA-RunAI-Cluster profile)

Pack Type	Helm chart
Registry	https://runai.jfrog.io/artifactory/api/helm/run-ai-charts
Pack Name	runai-agent
Pack Version	2.23.20
Install order	110
Values	Modified from default

This chart does not come with any YAML values be default, hence all configuration is custom.

The following aspects are configured for this helm chart:

- `pack.namespace` is set to `runai`.
- `pack.releaseNameOverride` is defined, ensuring the chart's release name stays consistent as `runai-cluster`.
- `pack.namespaceLabels` is defined, enabling the PSA privileged policy for the runai namespace, as is currently required per NVIDIA Run:ai documentation.
- `global.customCA` is defined to ensure SSL certificate trust.
- `global.imagePullSecrets` is defined to set the credentials for image pulling.
- The `controlPlane` and `cluster` sections are defined, pointing to 4 cluster profile variables. When a new workload cluster deployment is requested from the NVIDIA Run:ai control plane, the user will be presented with 4 values to use for deploying the workload cluster. Those values can then be entered into the cluster profile deployment wizard in Palette.