Spectro Cloud

Orchestrating AI Factory fabric multitenancy

Integration reference for Spectro Cloud PaletteAI, Aviz ONES, and NVIDIA Spectrum-X Ethernet

Rev 1.0.0 | March 2026

Contents

About this document

This document describes how to integrate Spectro Cloud PaletteAI with the Aviz ONES and NVIDIA Spectrum-X Ethernet (or NVIDIA Air for validation) so that orchestration and fabric control work together.

It provides a repeatable approach that can be adapted to specific environments and customer requirements, enabling a multi-tenant AI factory where compute pools and network fabric tenants stay aligned without manual reconciliation.

Change history

Version	Release date	Change summary
V1.0.0	March 2026	Initial version

Introduction

Using the integration described in this document, you can connect PaletteAI (apps and compute), Aviz ONES API (fabric and network control), and NVIDIA Spectrum-X Ethernet / Air (networking and infrastructure) into a single stack for enterprise AI factories.

It enables you to run AI factory workloads with self-service GPU pools, programmatic fabric control, and network-level isolation, from orchestration to Spectrum-X Ethernet in one integrated stack.

Platform teams keep governance while offering self-service GPU environments; data science teams deploy AI/ML workloads without touching Kubernetes or the network. Compute pools managed by PaletteAI are provisioned and wired through the Aviz ONES API into logical node groups (e.g., Red/Blue) of NVIDIA HGX™ servers on Spectrum-X Ethernet / Air.

You get a programmatic tenant lifecycle on the Spectrum-X fabric (create tenants, allocate and deallocate GPUs by server, delete tenants) with isolation across east-west GPU traffic, north-south user traffic, and storage. Orchestration (e.g., PaletteAI) drives pool and capacity changes through one API; the fabric stays consistent and auditable.

Scope

This document:

- Describes the integration of PaletteAI (compute pools, Profile Bundles, tenant policies) with Aviz ONES (fabric and node-level control) on NVIDIA Spectrum-X / Air.
- Covers the flow from project and Aviz tenant configuration through pool creation, fabric provisioning, and verification.
- Includes an example test (Red/Blue network isolation) with screenshots to illustrate connectivity before and after isolation.
- Provides design and deployment considerations, troubleshooting and scaling guidance, and improvement opportunities.

It does not define hardware sizing, storage solutions, or application-level AI frameworks beyond what is needed to understand the integration.

Who this document is for

This document is intended for practitioners: IT architects, solution architects, consultants, and platform or network administrators involved in the planning, design, or deployment of Kubernetes-based AI solutions using Spectro Cloud PaletteAI with Aviz ONES on NVIDIA Spectrum-X Ethernet.

It is assumed that the reader is familiar with Kubernetes concepts and with Spectro Cloud PaletteAI; familiarity with network fabric concepts and with Aviz or SONiC-based networking is helpful but not required.

Integration overview

1. Problem statement

Enterprise AI factories often run multiple tenants for teams, projects, or workloads on shared GPU and network fabric.

To achieve safe multitenancy requires isolation between tenants at both the application layer (who can use which pools and quotas) and the network layer (which nodes and fabric segments belong to which tenant, with separate VLANs, VNIs, and traffic boundaries).

When these orchestration and fabric control functions are siloed, platform teams must manually keep the two in sync: adding a pool or a tenant in the orchestration tool does not automatically create or update the corresponding fabric tenant or GPU allocation on the network.

That gap slows rollout, increases the risk of cross-tenant leakage or misconfiguration, and makes it hard to offer self-service GPU environments with consistent isolation and governance. Organizations need orchestration and fabric control to work together on Spectrum-X so that the multitenant pool and fabric tenant lifecycle stay aligned and auditable.

2. Architecture overview

The diagram illustrates the connections between PaletteAI and its compute pools, the Aviz ONES API, Red/Blue node groups, and NVIDIA Spectrum-X Ethernet / NVIDIA Air.

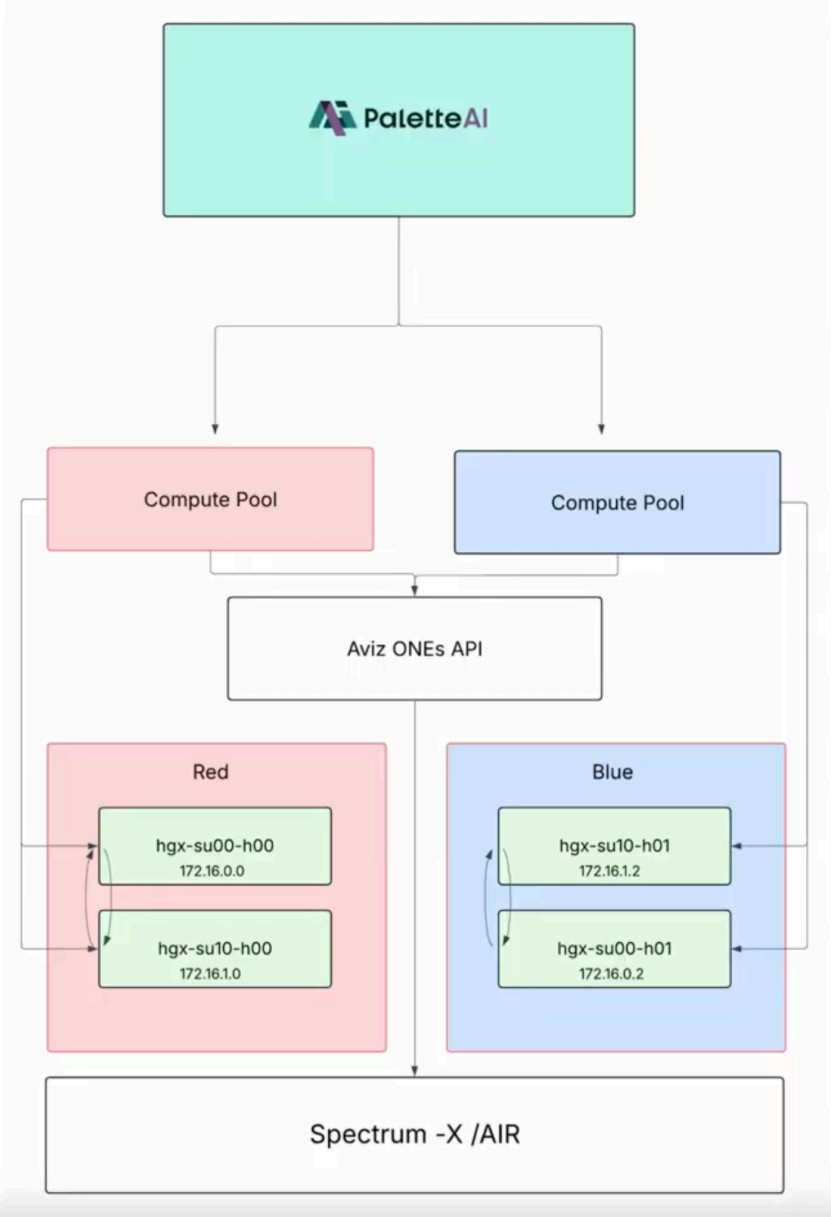


Figure 1: Integration flow

The solution follows a clear hierarchy from platform to fabric to hardware:

Layer	Component	Role
Management and orchestration	PaletteAI	Design, deploy, and manage AI workloads at scale; GPU-as-a-Service and self-service for data science and platform teams.
Resource abstraction	Compute Pools	Shared or dedicated Kubernetes clusters for AI/ML apps; provisioned via Palette, exposed by PaletteAI.
Control plane / integration	Aviz ONES API	Control plane for SONiC and AI fabric; turns orchestration intent into network and node configuration.
Logical node groups	Red / Blue (or similar)	Isolated GPU node groups (e.g., HGX) for tenants, projects, or tiers.
Hardware and networking	NVIDIA HGX + Spectrum-X / Air	GPU servers and AI-optimized Ethernet fabric; Air for simulation, validation, and AI-ready infrastructure.

Data flow (top-down):

1. PaletteAI defines and manages Compute Pools (e.g., two pools in the example diagram).
2. Compute Pools interface with the Aviz ONES API for network and node-level provisioning and configuration.
3. Aviz ONES API drives SONiC-based (and related) fabric and configures logical node groups, each containing specific HGX nodes with defined IPs and connectivity.
4. Those nodes run on NVIDIA Spectrum-X Ethernet / Air for high-performance, deterministic AI networking.

Application and tenant semantics live in PaletteAI; Aviz ONES owns fabric and node placement and wires the right physical resources to each pool. That division is how the integration works.

3. Component summary

Each part of the stack is described below, with its role in the integration.

3.1 Spectro Cloud PaletteAI

PaletteAI is a Kubernetes-native platform for full AI/ML workload lifecycle orchestration at scale. It uses a hub-spoke model: one control plane hub orchestrates many Compute Pool 'spokes', where apps run. Workload data stays on the spokes.

PaletteAI is built on Palette and needs a Palette integration (in Settings) to discover machines and provision clusters.

Profile Bundles combine Palette Cluster Profiles (infra) and workload profiles (apps/models) into governed, reusable stacks. Platform teams define a stack once and data scientists deploy it repeatedly. Tenants and Projects give isolation, GPU quotas, and RBAC.

PaletteAI does not configure the network or fabric; that is delegated to the Aviz ONES API. Aviz ONES handles SONiC, the NVIDIA Spectrum-X fabric and tenant allocation.

3.2 Compute Pools (PaletteAI ↔ Aviz ONES)

A Compute Pool is a group of machines (discovered via Palette and the Compute resource) used to create Kubernetes clusters for AI/ML. Each pool is a spoke cluster.

Pools can be dedicated or shared. Dedicated pools give one cluster per App Deployment, which is ideal for isolation or compliance. Shared pools host multiple deployments, delivering better efficiency and ideal for dev/staging.

In this integration, pools feed into the Aviz ONES API: Aviz ONES decides which nodes back each pool and keeps fabric configuration (VLANs, QoS, etc.) in line with each pool's needs.

Using the same HGX hostnames in both Palette (for cluster provisioning) and Aviz ONES (for fabric tenant allocation) means one node registry and consistent mapping between Kubernetes clusters and Spectrum-X tenants. PaletteAI stays infra-agnostic; Aviz ONES handles SONiC and fabric details.

3.3 Aviz ONES API

Aviz ONES and its API are the control platform for SONiC and AI fabric. It delivers orchestration, observability, and automation across vendors and ASICs.

For Northbound connectivity, ONES provides a REST API so PaletteAI or automation can drive tenant and GPU allocation without touching switches.

For Southbound connectivity, ONES uses GNMI and SONiC push config to Spectrum-X.

At the node level, Aviz ONES maps Compute Pools to node groups (Red/Blue) and to HGX hostnames (the same names used when allocating or deallocating servers to tenants via the Aviz ONES APIs). Aviz provides 24×7 SONiC support; the open APIs plug into existing NetOps and GitOps so you can script or automate fabric changes.

3.4 Logical node groups and HGX nodes

“Red” and “Blue” in our example diagram are logical groups of NVIDIA HGX servers.

Aviz ONES assigns nodes to groups, attaches them to the fabric, and maps them to Compute Pools. Each group gets its own fabric tenant and node set. This results in multitenancy, quota isolation, and security boundaries between pools.

At scale, the same pattern extends to many tenants and many nodes. You can have more logical groups (e.g. by team, project, or tier), each mapped to a Compute Pool and an Aviz ONES tenant, with more HGX hosts spread across the Spectrum-X fabric.

Allocation is still by hostname via the Aviz ONES APIs, so automation and tooling stay the same whether you have two groups or many. For very large deployments, groups are often aligned to physical or logical boundaries (e.g. rack, row, or availability zone) so that isolation and failure domains are predictable.

3.5 NVIDIA Spectrum-X and NVIDIA Air

Spectrum-X is NVIDIA’s Ethernet platform for AI factories. It offers higher bandwidth and lower latency than off-the-shelf Ethernet at scale. It uses Spectrum-X Ethernet switches and SuperNICs, runs SONiC and Cumulus (aligned with Aviz ONES), and fits east-west GPU scale-out, and north-south traffic.

NVIDIA Air is a cloud-hosted simulation platform that builds 1:1 simulations of AI data center infrastructure. Unlike a staging environment or a proof-of-concept rack, Air runs entirely in the cloud — no hardware required. It supports simulation of NVIDIA Spectrum Ethernet switches running Cumulus Linux or SONiC, NVIDIA BlueField DPUs and SuperNICs, and x86 servers with full OS and application stacks. Topologies can scale to thousands of switches and servers, and simulations can be shared across teams for collaborative testing and training.

In this stack, Air lets you validate Spectrum-X topology and Aviz ONES-driven tenant config in a lab before production so you can catch issues before going live.

4. Aviz ONES Spectrum-X API

Aviz ONES exposes REST APIs for multitenant Spectrum-X fabric operations. In public materials, Aviz describes endpoints covering tenant lifecycle management, GPU-aware resource allocation, and fabric-wide telemetry for AI workloads.

Orchestration platforms such as PaletteAI can use these APIs to drive pool and capacity changes, create tenants, allocate or deallocate GPUs by server, and decommission tenants, so that the "Compute Pool → Aviz ONES API → Red/Blue node groups" flow in the diagram is implemented programmatically.

Compute Pools in PaletteAI map conceptually to Aviz ONES tenants; HGX nodes are identified by hostname so that the same machines can be used for cluster provisioning (Palette) and fabric tenant isolation (Aviz ONES).

Integration flow

1. When creating a project, select and configure Aviz network isolation. This creates the Aviz tenants that will map to compute pools.
2. PaletteAI exposes one or more Compute Pools and attaches Profile Bundles and tenant/project policies. Whenever a compute pool is created, the Aviz tenant must be added as a label so the pool is bound to the correct fabric tenant. Configure or discover nodes before creating the pool so infrastructure resources and guardrails are in place.
3. When a pool is created or needs capacity or reconfiguration, PaletteAI talks to the Aviz ONES API to request or update node membership (which HGX nodes back the pool) and to keep the SONiC/AI fabric (Spectrum-X) configured for that pool's traffic and isolation.
4. Aviz ONES API maps pools to logical node groups, pushes configuration to SONiC on Spectrum-X switches (and, where used, to node-level or out-of-band management), and exposes topology and status for visibility and automation.
5. HGX nodes in each group run workloads (Kubernetes nodes), interconnected and connected to the rest of the AI factory via Spectrum-X / Air.

5. Example deployment: Red/Blue network isolation

Baseline: full connectivity before pools and isolation

Before any compute pools are created and before Aviz network isolation is applied, all HGX nodes on the fabric can reach each other. This establishes the starting state for the test.

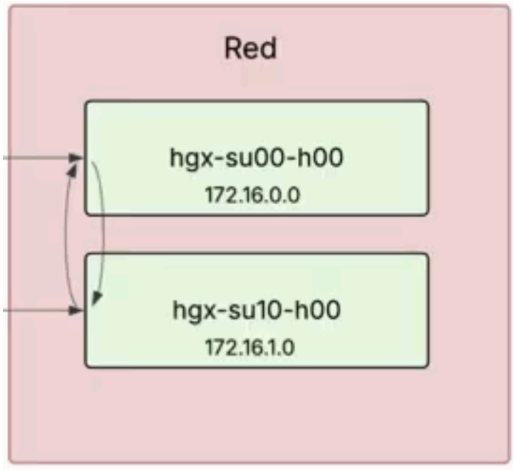
From a node such as **ubuntu@hgx-su00-h00**, run ping to the other nodes' IPs (e.g. **172.16.1.0**, **172.16.1.2**, **172.16.0.2**). With no isolation in place, all pings succeed. That confirms full connectivity across the fabric before Red/Blue pools and tenant isolation are configured.

```
ubuntu@hgx-su00-h00:~$ ping 172.16.1.0
PING 172.16.1.0 (172.16.1.0) 56(84) bytes of data.
64 bytes from 172.16.1.0: icmp_seq=1 ttl=61 time=4.41 ms
^C
--- 172.16.1.0 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.413/4.413/4.413/0.000 ms
ubuntu@hgx-su00-h00:~$ ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
64 bytes from 172.16.1.2: icmp_seq=1 ttl=61 time=2.95 ms
^C
--- 172.16.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 2.952/2.952/2.952/0.000 ms
ubuntu@hgx-su00-h00:~$ ping 172.16.0.2
PING 172.16.0.2 (172.16.0.2) 56(84) bytes of data.
64 bytes from 172.16.0.2: icmp_seq=1 ttl=63 time=1.66 ms
64 bytes from 172.16.0.2: icmp_seq=2 ttl=63 time=1.67 ms
^C
--- 172.16.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.663/1.668/1.673/0.005 ms
```

Applying isolation and verifying Red/Blue pools

The following steps describe the test walkthrough. The Red and Blue pools are not run at the same time: create the Red pool, run connectivity tests to verify isolation, then delete the Red pool; next, create the Blue pool and test again. In each phase, the single active pool is Running and Dedicated, and tests confirm that network isolation works automatically for that tenant.

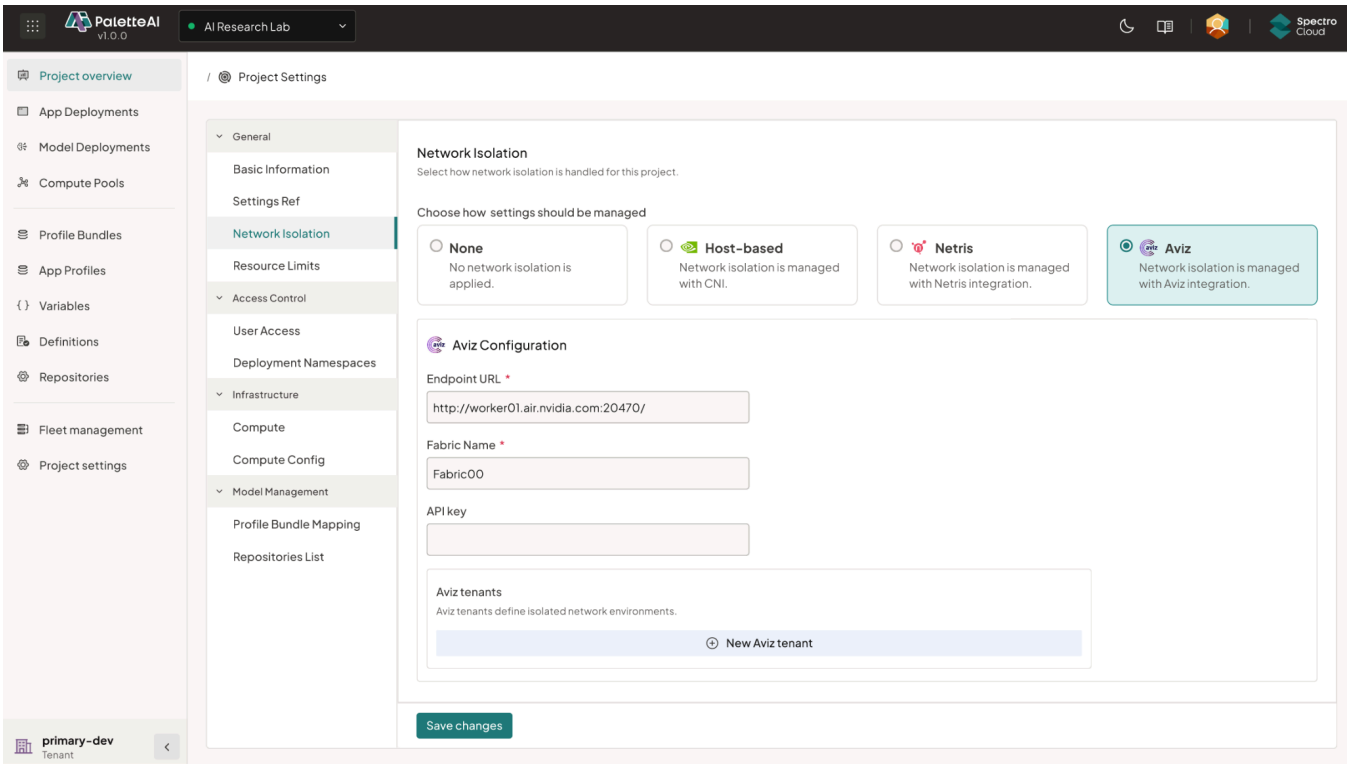
First test: For this phase, the two nodes **hgx-su00-h00** (172.16.0.0) and **hgx-su10-h00** (172.16.1.0) are both in the same tenant (Red). Connectivity between them is expected and confirms that the Red tenant is wired correctly before you run isolation checks.



Step 1: In Project settings → General → Network Isolation, the options are shown: None, Host-based, Netris, and Aviz. Initially None ("No network isolation is applied") may be selected.

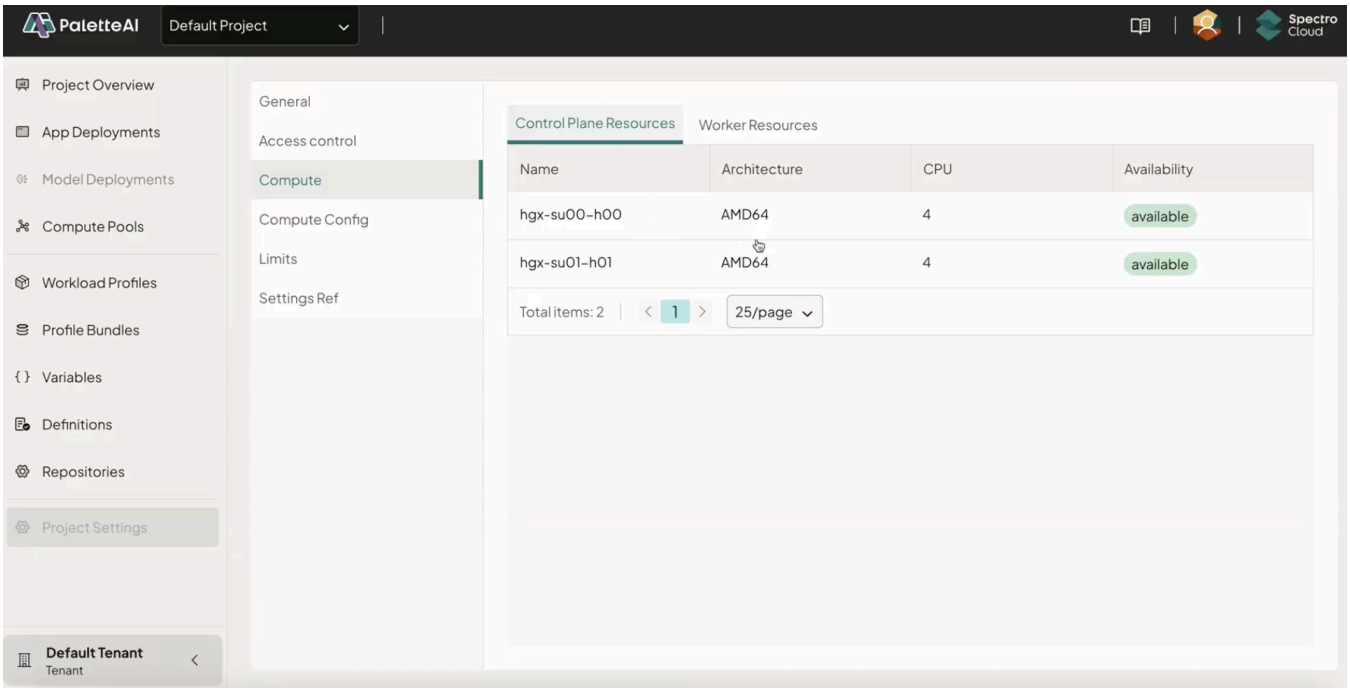
Step 2: Aviz is selected for "Network isolation is managed with Aviz integration." The Aviz Configuration section appears: Endpoint URL (e.g. http://worker01.air.nvidia.com:20470/), Fabric Name (e.g. Fabric00), and API key.

Under Aviz tenants, use "+ New Aviz tenant" to define isolated network environments (e.g. Red, Blue). Save changes.

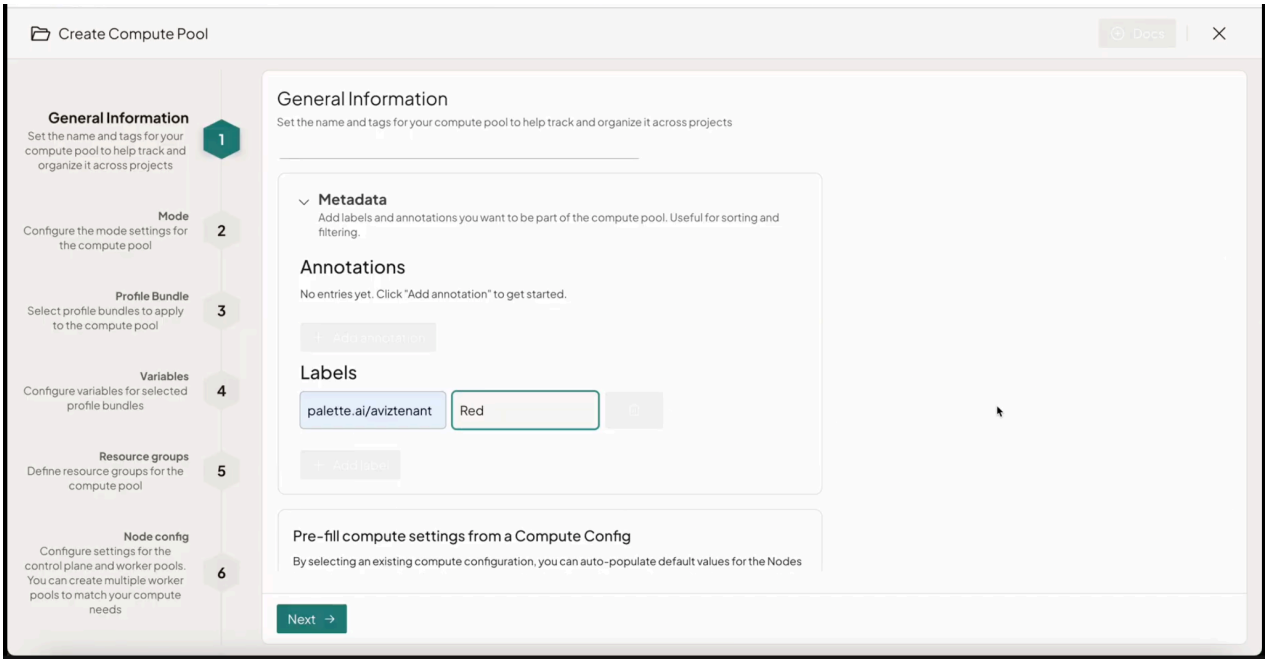


Spectro Cloud

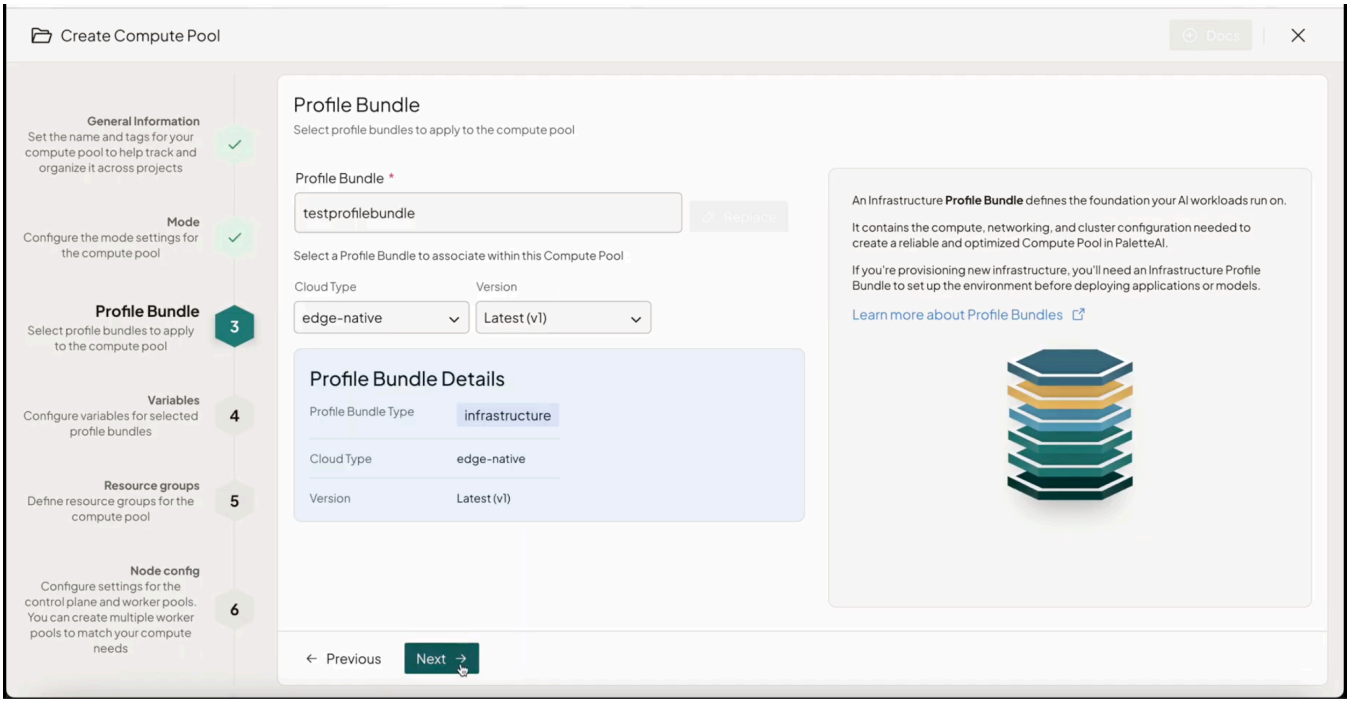
Step 3: In Compute Pools (or project Compute), the Control Plane Resources table shows available HGX hosts (e.g. hgx-su00-h00, hgx-su01-h01) with architecture, CPU, and Availability. This is the pool of machines that will be assigned to Red or Blue when creating compute pools.



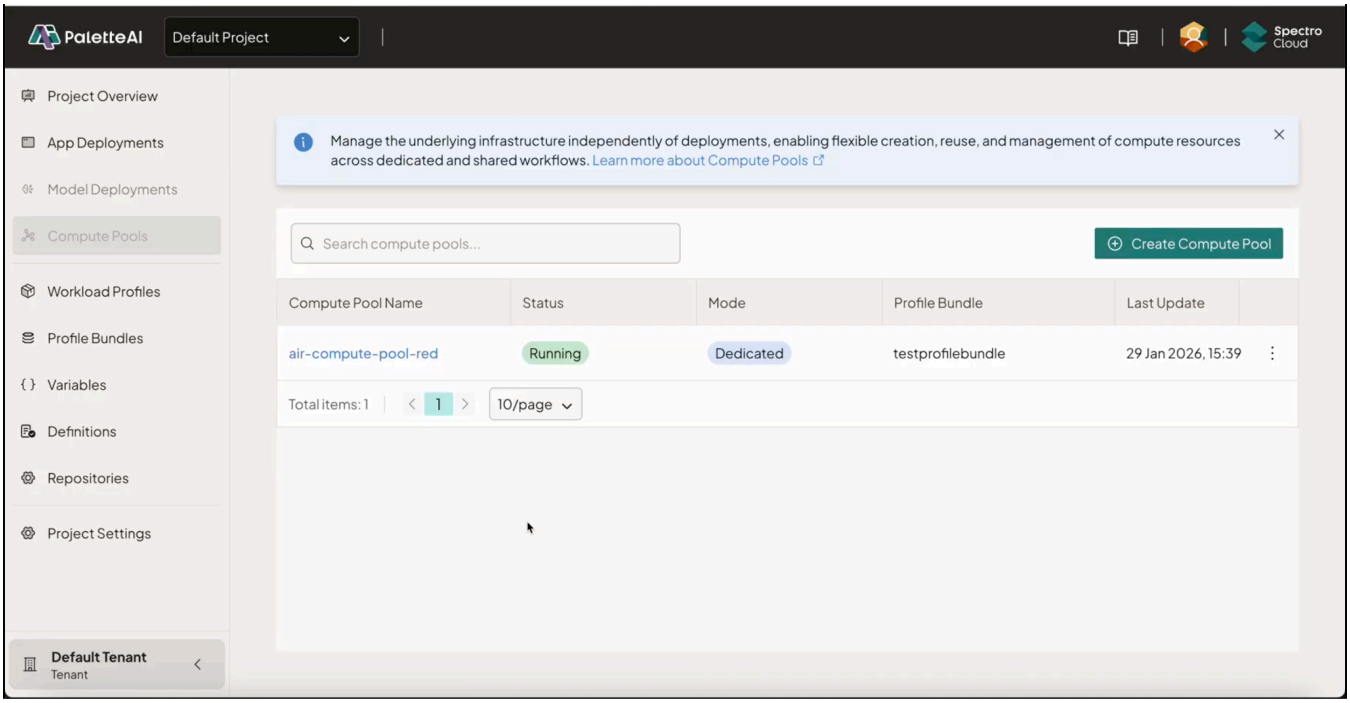
Step 4: Create Compute Pool wizard, step 1 – General Information. Set Compute pool name (e.g. air-compute-pool-red) and optional description. In Metadata → Labels, add the Aviz tenant label: palette.ai/aviztenant = Red (for the red pool; use Blue when creating the blue pool). This label ties the pool to the Aviz tenant and drives fabric isolation.



Step 5: Wizard step 3 – Profile Bundle. Select an Infrastructure Profile Bundle (e.g. testprofilebundle) and Cloud Type (e.g. edge-native), Version (e.g. Latest v1). The Infrastructure Profile Bundle defines the foundation (compute, networking, cluster configuration) for the pool. Complete the remaining wizard steps (Variables, Resource groups, Node config, Deployment, Summary) and Submit.



Step 6: After creating the red pool, the Compute Pools table shows *air-compute-pool-red* with Status: *Running*, Mode: *Dedicated*.

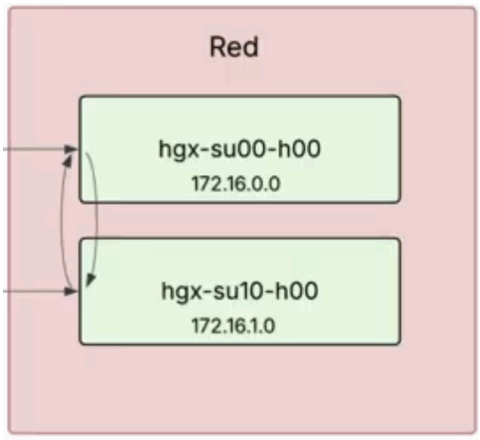


Now let's validate the network configuration:

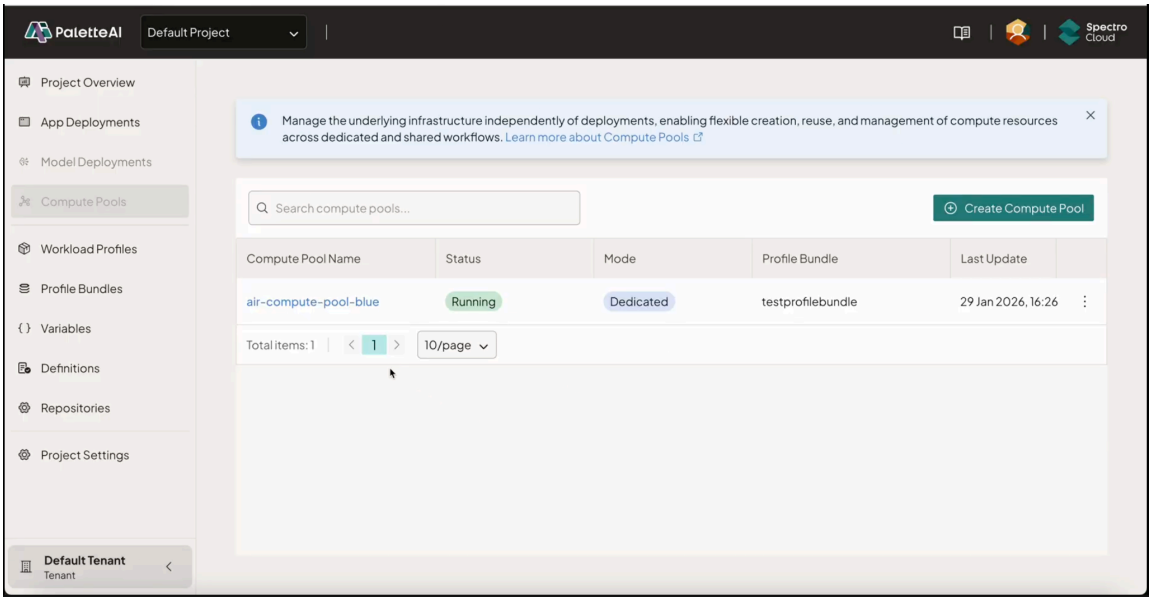
- From a node in the Red tenant (e.g. **hgx-su00-h00**), run ping to the other addresses. Same tenant (e.g. **172.16.1.0**): 0% packet loss. Illustration that in-tenant connectivity works.
- From a node that's not in tenant (e.g. **172.16.0.2**, **172.16.1.2**): 100% packet loss, meaning that they are unreachable. So only nodes in the same tenant can communicate; the rest cannot. Now delete the Red pool and go to the next step.

```
ubuntu@hgx-su00-h00:~$ ping 172.16.0.2
PING 172.16.0.2 (172.16.0.2) 56(84) bytes of data.
^C
--- 172.16.0.2 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6121ms

ubuntu@hgx-su00-h00:~$ ping 172.16.1.0
PING 172.16.1.0 (172.16.1.0) 56(84) bytes of data.
64 bytes from 172.16.1.0: icmp_seq=1 ttl=62 time=3.02 ms
64 bytes from 172.16.1.0: icmp_seq=2 ttl=62 time=3.37 ms
64 bytes from 172.16.1.0: icmp_seq=3 ttl=62 time=3.27 ms
^C
--- 172.16.1.0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.019/3.219/3.374/0.148 ms
ubuntu@hgx-su00-h00:~$ ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
^C
--- 172.16.1.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2042ms
```



Step 7: Create the blue pool (name *air-compute-pool-blue*, label *palette.ai/aviztenant = Blue*). The table shows *air-compute-pool-blue* with Status: *Running*, Mode: *Dedicated*.



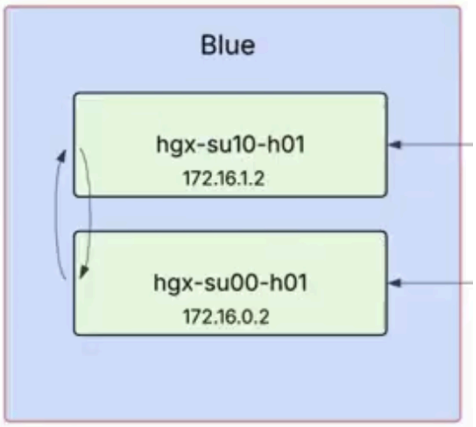
Now let's validate the network configuration for the Blue tenant:

- From a node in the Blue tenant (e.g. **hgx-su00-h01** or **hgx-su10-h01**), run ping to the other addresses. Same tenant (e.g. **172.16.0.2**, **172.16.1.2**): 0% packet loss, in-tenant connectivity works.

- From a node that's not in the same tenant (e.g. **172.16.1.0**): 100% packet loss, unreachable. So only nodes in the same tenant can communicate; the rest cannot. This completes the test; both Red and Blue phases confirm that network isolation is enforced automatically.

```
ubuntu@hgx-su00-h00:~$ ping 172.16.0.2
PING 172.16.0.2 (172.16.0.2) 56(84) bytes of data.
^C
--- 172.16.0.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1011ms

ubuntu@hgx-su00-h00:~$ ping 172.16.1.0
PING 172.16.1.0 (172.16.1.0) 56(84) bytes of data.
64 bytes from 172.16.1.0: icmp_seq=1 ttl=62 time=3.39 ms
64 bytes from 172.16.1.0: icmp_seq=2 ttl=62 time=3.04 ms
^C
--- 172.16.1.0 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 3.038/3.212/3.387/0.174 ms
ubuntu@hgx-su00-h00:~$ ping 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data.
^C
--- 172.16.1.2 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1003ms
```



6. Design and deployment considerations

Architecture

- **Mapping pools to node groups:** Define rules (e.g., by tenant, project, or tier) for which Compute Pool maps to which Aviz ONES tenant and node group. Document for operations and security so resize and decommissioning stay predictable.
- **IP and naming:** The example above uses a structured IP scheme (e.g., 172.16.x.x) and hostnames (e.g., hgx-su00–h00). Server hostnames should match what Aviz ONES expects from Day-0; that keeps allocation/deallocation and troubleshooting simple.
- **Lifecycle coordination:** PaletteAI for app and cluster lifecycle; Aviz ONES for fabric and tenant/GPU lifecycle (day-1 and day-2). On pool resize or teardown, deallocate GPUs via Aviz ONES before or in sync with PaletteAI so tenant deletion can complete (APIs typically require no GPUs allocated before a tenant can be removed).
- **Validation:** Use NVIDIA Air to validate Spectrum-X topology and Aviz ONES-driven tenant configs in a lab before production, to lower risk when adding tenants or nodes. Align logical node groups with physical or logical boundaries so isolation and failure domains are predictable. Use GPU quotas and resource groups in PaletteAI to control usage across pools.
- **High availability:** For production, configure high availability for the PaletteAI, Palette, and Aviz ONES control planes to keep pool and fabric operations consistent during failures.

Troubleshooting

- **Isolation and connectivity:** If nodes in the same tenant cannot reach each other, confirm the compute pool has the correct Aviz tenant label, that Aviz ONES has the right node membership for that tenant, and that fabric (SONiC) config for the tenant is applied. If cross-tenant traffic is allowed when it should not be, verify tenant-to-node-group mapping in Aviz ONES and that no node is assigned to multiple tenants; check fabric isolation (e.g. VLANs/VNIs) for the tenant.
- **Pool or nodes not provisioning:** Ensure nodes are discoverable in Palette and that the Aviz tenant is defined and matches the label on the pool. Confirm Aviz ONES API calls succeed (tenant created, nodes allocated by hostname). Check PaletteAI and Aviz ONES logs or status for errors.
- **Trust the separation of concerns:** Look to PaletteAI for pool, app, and quota issues; Aviz ONES for fabric config and node-to-tenant assignment; Spectrum-X and NVIDIA stack for physical network and hardware. Aviz ONES exposes topology and status for visibility; use that to verify node groups and fabric state.

Scaling

- **Adding tenants:** Define new Aviz tenants in the project's network isolation settings, then create compute pools with the corresponding tenant label. Aviz ONES and the fabric support additional logical groups; keep pool-to-tenant mapping documented.
- **Adding nodes:** Register or discover new nodes in Palette so they are available for pools. Allocate them to the intended tenant via the Aviz ONES API (by hostname). Ensure the Spectrum-X fabric has capacity and that node groups align with your failure-domain strategy (e.g. rack, row, or availability zone).

7. Conclusion: a foundation for orchestrating multitenancy

This integration tackles a real problem in enterprise AI factories: that orchestration and fabric control are usually disconnected. Once implemented, PaletteAI gives platform and data science teams one place to manage GPU pools, Profile Bundles, and tenant policies; Aviz ONES turns that intent into Spectrum-X Ethernet fabric configuration through open APIs.

When you create a pool and attach an Aviz tenant label, the fabric is configured for that tenant via the Aviz ONES API. Because the same pool-to-tenant binding drives both PaletteAI's pool (and its quotas and policies) and the fabric's logical node group, you no longer need manual runbooks to keep pools and fabric tenants aligned.

The integration is most valuable where you need multitenant GPU environments with clear boundaries: different teams, projects, or compliance zones sharing the same physical fabric but with strict traffic isolation. It also helps when you scale out tenants and nodes without a proportional increase in manual work, because pool and fabric changes are driven programmatically through the Aviz ONES API and PaletteAI's model, including GitOps-friendly resources where that fits your workflow.

The stack described in this document is a solid baseline. However, there is immediate potential to extend it in two ways.

- Adding load balancers or ingress controllers for north-south traffic to model serving, APIs, or dashboards can improve availability and scalability; PaletteAI's Compute Config and deployment settings work alongside external or in-stack load balancing. Aviz ONES already provides fabric-level observability, topology, and alerts, and PaletteAI provides app and pool-level visibility; correlating those two can assist with incident response and capacity planning.
- Also note that this reference integration focuses on compute and network isolation only. Integrating shared or tenant-scoped storage for training data, checkpoints, or model artifacts under the same tenant identity would complete the multitenant data story.

NVIDIA Air remains the right tool to validate Spectrum-X Ethernet topology and Aviz ONES-driven tenant config before production changes, and testing can be extended to load balancer configs, ingress paths, and failover behavior where that matters. None of these extensions replace the core integration; they build on it so that self-service GPU environments stay governable, observable, and resilient as usage grows.

8. References and further reading

To learn more about the components of this integration, check out these resources:

- PaletteAI: spectrocloud.com and docs.palette-ai.com
- Aviz ONES: aviznetworks.com/products/ones, REST API Explorer.
- NVIDIA Spectrum-X: nvidia.com/networking/spectrumx
- NVIDIA AI Factory: nvidia.com/en-gb/solutions/ai-factories/validated-design/