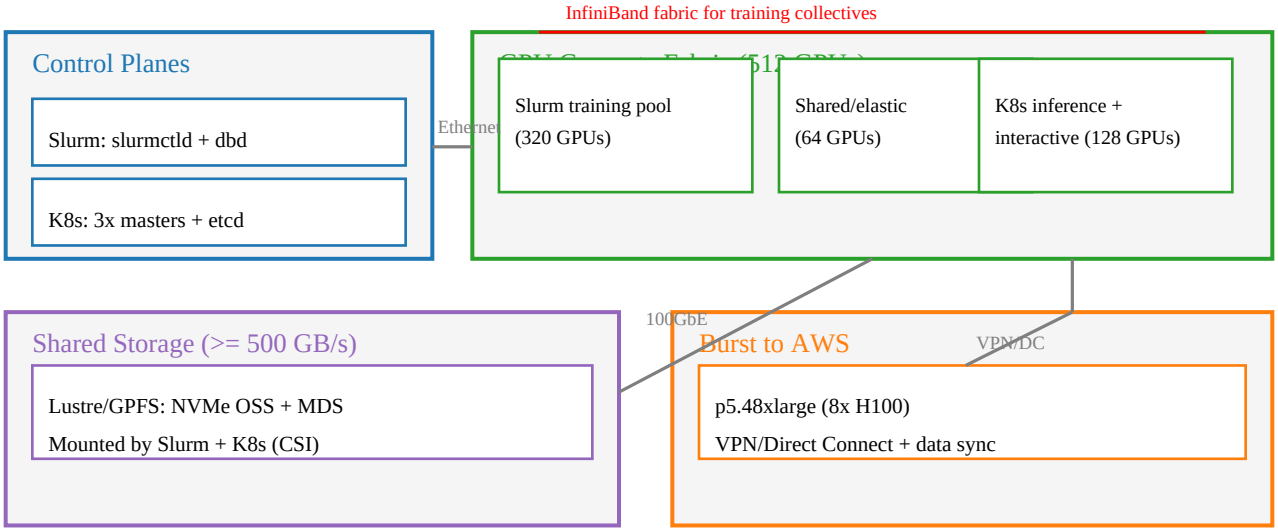
Hybrid GPU Cluster Architecture (512 GPUs)

On-prem hybrid AI platform for interactive development, Slurm training, Kubernetes inference, and cloud bursting.

Target users	200+ data scientists (Jupyter/VSCode), multiple teams
Workloads	Interactive notebooks, distributed training up to 512 GPUs, low-latency inference
Schedulers	Slurm for training; Kubernetes for inference + interactive; controlled shared pool
Networking	InfiniBand for training collectives; 25/100GbE for services, storage, mgmt
Storage	Shared POSIX (Lustre/GPFS) >= 500 GB/s, CSI mount for K8s
Bursting	Overflow training to AWS p5.48xlarge via VPN/Direct Connect and staged data

1. System Architecture

A single physical GPU fleet is split into three logical pools: **Slurm training**, **Kubernetes inference/interactive**, and a small **shared elastic** pool. This keeps inference isolated for latency while still enabling high utilization.



Compute sizing: 32 GPU nodes x 16 GPUs/node = 512 GPUs total. Dual-socket CPUs; 1-2 TB RAM/node; local NVMe for caching.

Pool split (configurable): 20 nodes Slurm-only (320 GPUs), 8 nodes K8s-only (128 GPUs), 4 nodes shared (64 GPUs).

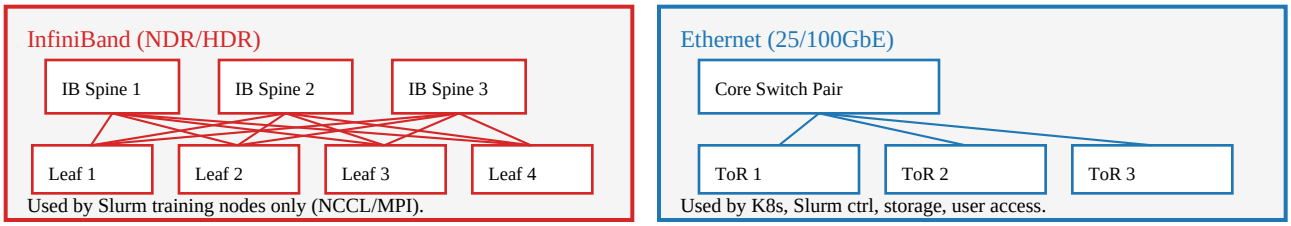
Interactive platform (200+ data scientists)

Interactive sessions run on Kubernetes using JupyterHub and VSCode-server pods. GPU sharing uses **NVIDIA MIG** (preferred) or time-slicing for light notebooks; heavier interactive jobs request full GPUs. Ingress terminates TLS, integrates with SSO, and routes users into per-team namespaces.

2. Network and Storage

Two physically separate fabrics avoid cross-traffic contention: **InfiniBand** for distributed training collectives (NCCL/MPI) and **Ethernet** for management, Kubernetes services, inference ingress, and storage access.

Network Topology (separate fabrics)



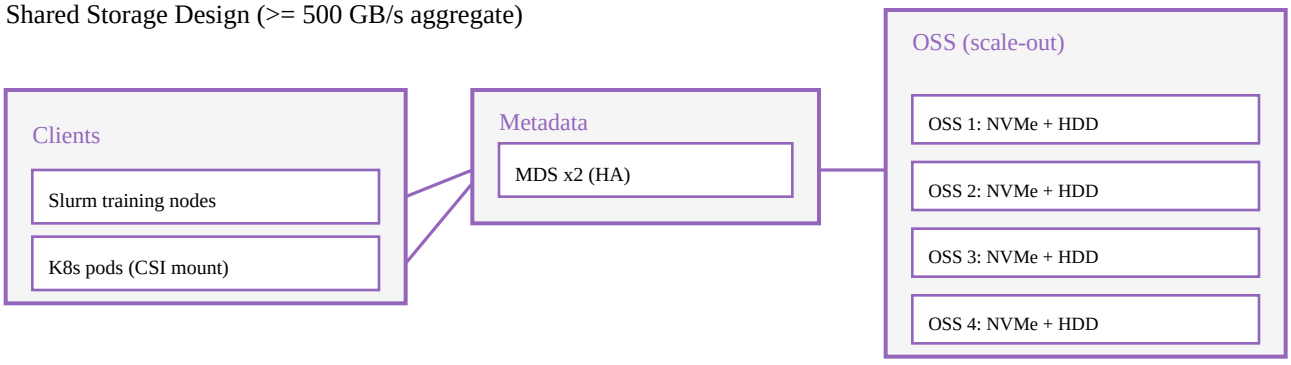
Segmentation

- Mgmt VLAN (BMC/iDRAC)
- User access VLAN (Jupyter/VSCode)
- Storage VLAN (RDMA over Converged Ethernet optional)
- Service mesh / ingress VLAN for inference
- Firewall between tenants and control planes
- No east-west between namespaces by default (K8s NetworkPolicy)

Shared Storage (>= 500 GB/s)

Use a shared POSIX filesystem (Lustre or GPFS) with scale-out object storage servers (OSS) and NVMe tiers. Expose the filesystem to Slurm nodes via native clients and to Kubernetes via CSI.

Shared Storage Design (>= 500 GB/s aggregate)



Notes

- Data plane over 100GbE (or IB for storage clients if desired).
- Add OSS servers to reach 500+ GB/s.

- Baseline sizing: 12-16 OSS servers (scale to meet throughput); separate metadata servers in HA.
- Optional: S3-compatible on-prem object tier for long-term checkpoints and replication to AWS.

3. Scheduling, Multi-tenancy, and Security

Schedulers coexist via **node-level ownership** plus a small shared pool. This avoids dual-scheduler contention and provides a hard isolation boundary for inference SLOs.

Workload	Placement	Why
Training (32-512 GPUs)	Slurm (training pool + shared pool)	Gang scheduling; topology-aware placement; mature fair-share
Inference (low latency)	Kubernetes (inference pool)	Autoscaling; services; canaries; traffic routing
Interactive (Jupyter/VSCode)	Kubernetes (interactive pool)	Multi-user isolation; fast start/stop; GPU sharing (MIG/time-slice)

Multi-tenancy, quotas, and fair-share

• **Slurm**: accounts per team; partitions mapped to pools; QOS tiers (normal/high); fair-share via multifactor priority (historical usage, job size, age).
• **Kubernetes**: namespace per team; ResourceQuota + LimitRange for GPU/CPU/memory; PriorityClasses reserve capacity for inference.
• **Policy**: admission checks require labels (team/user/job_type) and reject requests above team quota before submission.

Security isolation

• AuthN: SSO (OIDC) for K8s + JupyterHub; Slurm via LDAP; short-lived tokens for APIs.
• AuthZ: K8s RBAC; Slurm account controls; separate admin break-glass path with audit logging.
• Supply chain: private registry, image signing, vulnerability scanning; disallow privileged pods by default; runtime sec policies.
• Data: POSIX ACLs per project; encryption in transit; optional at-rest encryption for sensitive tiers.

4. AWS Bursting and Cost

Bursting is used only for **training** jobs when local capacity cannot meet an SLO (e.g., expected start time > 6h) or when the requested GPU count cannot be satisfied locally. Jobs run on AWS **p5.48xlarge** (8x H100) using the same container images.

• Connectivity: Direct Connect preferred; VPN acceptable for smaller bursts. Centralized logging and metrics back to on-prem.
• Data: stage datasets to S3 / FSx for Lustre; sync checkpoints back to on-prem; avoid running interactive over WAN.
• Guardrails: per-team cloud budget caps; only burst high-priority or large jobs; automated cleanup of idle clusters.

Cost estimates (order-of-magnitude)

Component	Assumption	Rough cost (USD)
GPU servers	32 nodes x (16 GPUs + CPUs + RAM + NICs)	$18M - $28M
InfiniBand fabric	Spines/leaves, cables, optics	$1.5M - $3.5M
Ethernet fabric	Leaf-spine 100GbE, optics	$0.6M - $1.5M
Shared storage	Lustre/GPFS: OSS + MDS + media	$3M - $8M
Control + login nodes	K8s masters, Slurm ctrl/db, bastion	$0.2M - $0.6M
Support + rack/power overhead	3-year support, racks, PDUs (excl. DC buildout)	$2M - $6M
Total (ballpark)	CapEx	**$25M - $48M**

Note: Excludes data center buildout, staffing, and software licensing (if applicable).