Hydra Lab Infrastructure Proposal

Storage & GPU Cluster Modernization

Computer Science Department
SUNY New Paltz

January 9, 2026

Abstract

This proposal outlines the infrastructure modernization plan for the Hydra computing cluster, including ZFS RAID-10 storage deployment, GPU node optimization, multi-node inference setup, and Kubernetes orchestration. The plan addresses current storage limitations, optimizes GPU utilization across heterogeneous hardware, and establishes a scalable foundation for student container workloads.

Contents

1 Executive Summary 3
- 1.1 Current State . . . 3
- 1.2 Proposed Changes . . . 3
- 1.3 Expected Outcomes . . . 3

2 Hardware Inventory 3
- 2.1 Cluster Overview . . . 3
- 2.2 Hydra Storage Drives . . . 3
- 2.3 Chimera Storage Drives . . . 4
- 2.4 Cerberus Storage Drives . . . 4

3 Architecture Overview 4
- 3.1 Target Architecture . . . 4
- 3.2 Storage Architecture . . . 5

4 Phase 0: Pre-Migration Backup 5
- 4.1 Overview . . . 5
- 4.2 Backup Actions . . . 5

5 Phase 1: ZFS Storage Setup on Hydra 6
- 5.1 Overview . . . 6
- 5.2 Storage Actions . . . 6
- 5.3 Phase 1 Veri cation . . . 8

6 Phase 2: GPU Node Storage Con guration 8
- 6.1 Overview . . . 8
- 6.2 GPU Node Storage Layout . . . 8
- 6.3 Phase 2A: Chimera Con guration . . . 8
- 6.4 Phase 2B: Cerberus Con guration . . . 9
- 6.5 Phase 2 Veri cation . . . 10

7 Phase 2C: Open WebUI Multi-Node Setup 10
- 7.1 Overview . . . 10
- 7.2 Multi-Node Setup Actions . . . 10
- 7.3 Model Distribution Strategy . . . 11

8 Phase 3: RKE2 Kubernetes Deployment 11
- 8.1 Overview . . . 11
- 8.2 Why RKE2 over Docker Swarm . . . 12
- 8.3 Cluster Topology . . . 12
- 8.4 RKE2 Deployment Actions . . . 12
- 8.5 Phase 3 Veri cation . . . 13

9 GPU Job Scheduling 13
- 9.1 Scheduling Strategy . . . 13
- 9.2 Container Migration Flow . . . 14
- 9.3 Migration Actions . . . 14

10 Implementation Timeline 15
- 10.1 Phase 0: Pre-Migration Backup (Day 0) . . . 15
- 10.2 Phase 1: Hydra Storage (Day 1) . . . 15

10.3 Phase 2: GPU Node Storage (Day 2) 15
10.4 Phase 2C: Open WebUI Multi-Node (Day 2) 16
10.5 Phase 3: RKE2 Deployment (Day 3) 16

11 Risk Assessment 16

12 Future Enhancements 16

13 Appendix: Technology References 17

1 Executive Summary

1.1 Current State

The Hydra Lab currently operates three servers with underutilized storage capacity and sub-optimal GPU resource allocation. Student containers run on limited local storage, and GPU access requires manual intervention.

1.2 Proposed Changes

Deploy 21TB ZFS RAID-10 storage pool on Hydra for centralized container storage

Optimize GPU node storage to leverage fast NVMe drives for model inference

Implement multi-node Ollama deployment for distributed inference (136GB VRAM)

Deploy RKE2 Kubernetes cluster for automated GPU scheduling

Establish centralized logging and monitoring infrastructure

1.3 Expected Outcomes

21TB redundant storage with automatic compression and snapshots

5× GPU utilization improvement through intelligent scheduling

Reduced manual intervention for student GPU access requests

Foundation for future expansion and self-service portal

2 Hardware Inventory

2.1 Cluster Overview

Table 1: Cluster Hardware Con guration

Node	Role	RAM	GPUs	Storage
Hydra	Control + Storage	251GB	None	6×7TB RAID-10, 1.7TB OS, 1.1TB backup
Chimera	GPU Inference	251GB	3×RTX 3090 (72GB)	3.5TB NVMe (fast), 3.5TB SATA (slow)
Cerberus	GPU Training	64GB	2×RTX 5090 (64GB)	3.6TB NVMe, 1.7TB NVMe

2.2 Hydra Storage Drives

Device	Size	Model	Speed	Purpose
sda-sdf	7TB each	PA33N7T6 EMC7680	550-750 MB/s	RAID-10 pool (tank)
sdg	1.7TB	Micron MTFDDAK1T9TDS	548 MB/s	OS boot drive
sdh	1.1TB	Seagate ST1200MM0099	255 MB/s	Daily OS backups

Note: The 1.1TB Seagate drive (sdh) is labeled as "SAS 10k" but benchmarks at only 255 MB/s likely a mislabeled SATA drive. Suitable for cold backups only, not as ZFS cache.

2.3 Chimera Storage Drives

Device	Size	Model	Speed	Purpose
nvme0n1	3.5TB	Samsung MZ1L23T8HBLA	2.2 GB/s	OS + Ollama models
sda	3.5TB	Micron 5210	19 MB/s	Cold storage only

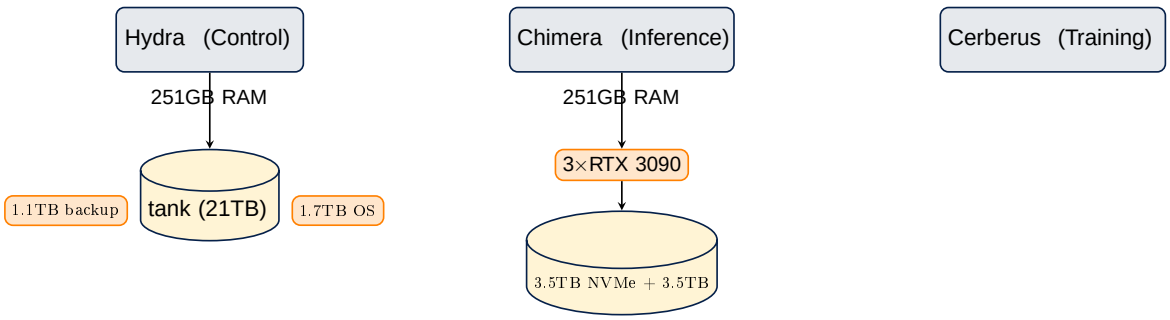
Critical: The Micron 5210 (sda) benchmarks at only 19 MB/s extremely slow. All active workloads must use the NVMe drive. The SATA drive is suitable only for archives and cold backups.

2.4 Cerberus Storage Drives

Device	Size	Model	Speed	Purpose
nvme0n1	3.6TB	MSI M480 PRO	2.5 GB/s	OS + containers
nvme1n1	1.7TB	Samsung MZQL21T9HCJR	2.5 GB/s	Model storage

3 Architecture Overview

3.1 Target Architecture



3.2 Storage Architecture

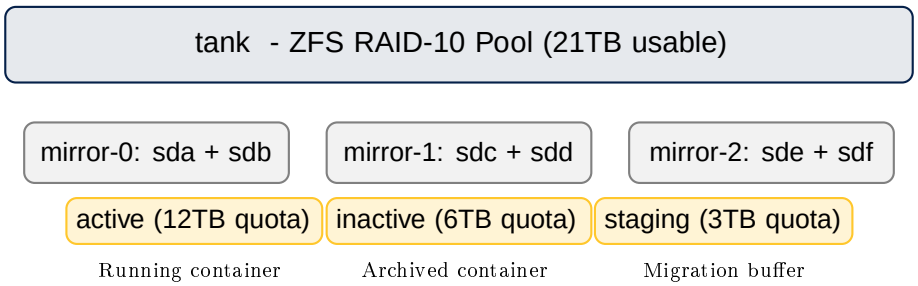


Figure 2: ZFS Storage Pool Architecture

4 Phase 0: Pre-Migration Backup

4.1 Overview

Before making any changes to the cluster, all critical data will be backed up to Chimera's slow /data drive (3.5TB Micron). This drive is only 19 MB/s but adequate for one-time backups.

Critical: All backups must be completed and veri ed before proceeding to Phase 1.

4.2 Backup Actions

0.1 Prepare Backup Location

Verify Chimera /data has su cient space (∼3.5TB available)
Create backup directory structure at /data/backups/
Create subdirectories for hydra, chimera, cerberus

0.2 Backup Hydra

Backup /home directories (user data) via rsync over network
Backup /etc con guration les
Backup Docker volumes from /var/lib/docker/volumes/
Dump MySQL databases to SQL le
Dump PostgreSQL databases to SQL le

0.3 Backup Chimera

Backup /home directories locally
Backup /etc con guration les
Backup Ollama models from /var/lib/ollama/
Backup Docker volumes
Dump MySQL databases if present

0.4 Backup Cerberus

Backup /home directories via rsync over network
Backup /etc con guration les

0.5 Verify Backups

Check backup sizes with du -sh /data/backups/*/
Verify critical les exist in each backup directory
Test database dumps are valid SQL
Con rm total backup size ts on /data

5 Phase 1: ZFS Storage Setup on Hydra

5.1 Overview

Deploy a ZFS RAID-10 pool using 6×7TB enterprise drives in 3 mirror pairs, providing 21TB usable storage with redundancy (can lose one drive per mirror pair).

5.2 Storage Actions

1.1 Verify Hardware

Con rm 6 ×7TB drives (sda-sdf) are detected
Verify drives are PA33N7T6 EMC7680 model
Check drive health with smartctl

1.2 Install ZFS

Install zfsutils-linux package
Load ZFS kernel modules
Verify installation with zfs version

1.3 Create RAID-10 Pool

Create pool "tank" with 3 mirror vdevs
Mirror pair 1: sda + sdb
Mirror pair 2: sdc + sdd
Mirror pair 3: sde + sdf
Set ashift=12 for 4K sector alignment
Enable lz4 compression
Disable atime for performance

1.4 Create Datasets

Create tank/active with 12TB quota (running containers)
Create tank/inactive with 6TB quota (archived containers)

Create tank/staging with 3TB quota (migration bu er)

1.5 Con gure Backup Drive

Format sdh (1.1TB Seagate) as ext4
Mount at /backups
Add to /etc/fstab for persistence
Create backup subdirectories for each node

1.6 Con gure NFS Exports

Install nfs-kernel-server
Export /tank/active to 192.168.1.0/24
Export /tank/inactive to 192.168.1.0/24
Export /tank/staging to 192.168.1.0/24
Set rw,async,no_root_squash options

1.7 Setup Automatic Snapshots

Con gure hourly snapshots of tank/active (retain 24)
Con gure daily snapshots of all datasets (retain 7)
Schedule weekly scrub on Sundays at 1 AM

1.8 Setup Daily OS Backups

Create backup script at /usr/local/bin/backup-os.sh
Backup Hydra /etc and /home to /backups/hydra/
Backup Chimera /etc via rsync to /backups/chimera/
Backup Cerberus /etc via rsync to /backups/cerberus/
Schedule via cron at 3 AM daily

1.9 Setup Cluster Aliases

Create /etc/pro le.d/hydra-aliases.sh on all nodes
Add SSH shortcuts: hydra, chimera, cerberus
Add status aliases: gpu, gpuw, zstat, zlist, dps
Add kubectl aliases: kgn, kgp
Add log aliases: logs-backup, logs-docker, logs-ollama

1.10 Setup Centralized Logging

Con gure rsyslog on Chimera/Cerberus to forward to Hydra
Enable UDP reception on Hydra port 514
Create log directories at /var/log/hydra-cluster/
Logs will be used for future dashboard integration

5.3 Phase 1 Veri cation

Success Criteria:

Pool status shows ONLINE with 3 mirror vdevs
All datasets created with correct quotas
NFS exports accessible from Chimera and Cerberus
Backup drive mounted at /backups
Aliases working on all nodes

6 Phase 2: GPU Node Storage Con guration

6.1 Overview

Optimize storage on both GPU nodes to ensure Ollama models run on fast NVMe storage, not slow SATA drives.

6.2 GPU Node Storage Layout

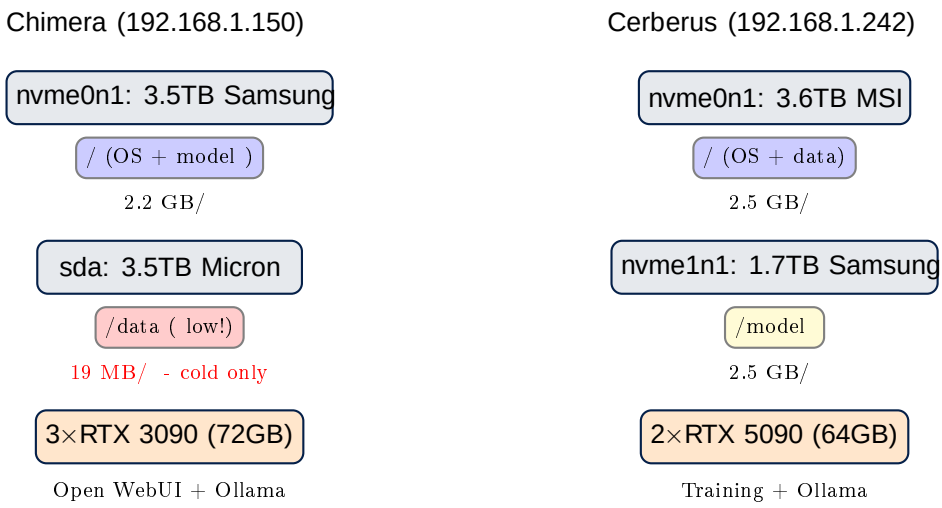


Figure 3: GPU Node Storage Layout

6.3 Phase 2A: Chimera Con guration

2A.1 Verify Chimera Drives

Con rm nvme0n1 (3.5TB Samsung) is mounted at /
Con rm sda (3.5TB Micron) is mounted at /data
Test drive speeds with hdparm
Verify NVMe is ∼2.2 GB/s, SATA is only ∼19 MB/s

Important: Do NOT store Ollama models on /data the Micron 5210 is only 19 MB/s. All models must be stored on the fast NVMe root partition.

2A.2 Con gure Ollama on Fast NVMe

Create model directory at /var/lib/ollama/models
Set ownership to ollama:ollama
Con gure systemd override for OLLAMA_MODELS path
Con gure OLLAMA_HOST=0.0.0.0 for network access
Restart ollama service

2A.3 Con gure Chimera Directories

Create /var/lib/containers/{staging,active} on NVMe
Set ownership to root:docker
Use /data only for cold storage (archives, old backups)

2A.4 Verify NVIDIA Drivers

Check nvidia-smi shows 3× RTX 3090
Install nvidia-driver-550 if not present
Enable nvidia-persistenced service

6.4 Phase 2B: Cerberus Con guration

2B.1 Verify Cerberus Drives

Con rm nvme0n1 (3.6TB MSI) is mounted at /
Con rm nvme1n1 (1.7TB Samsung) is available for /models

2B.2 Con gure Model Storage

Format nvme1n1 as ext4 with label "models"
Create mount point at /models
Add to /etc/fstab for persistence
Create subdirectories: ollama, huggingface, checkpoints

2B.3 Con gure Container Directories

Create /var/lib/containers/{staging,active,training}
Set ownership to root:docker

2B.4 Verify NVIDIA Drivers

Check nvidia-smi shows 2× RTX 5090
Install nvidia-driver-550 (or 560+ for 5090 support)
Enable nvidia-persistenced service

2B.5 Con gure Ollama

Con gure systemd override for OLLAMA_MODELS=/models/ollama
Con gure OLLAMA_HOST=0.0.0.0 for network access
Restart ollama service

6.5 Phase 2 Veri cation

Chimera Success Criteria:

nvme0n1 mounted at / with ∼2.2 GB/s
sda mounted at /data (cold storage only)
3×RTX 3090 detected by nvidia-smi
Ollama using /var/lib/ollama/models on fast NVMe

Cerberus Success Criteria:

nvme0n1 mounted at /
nvme1n1 mounted at /models
2×RTX 5090 detected by nvidia-smi
Ollama using /models/ollama

7 Phase 2C: Open WebUI Multi-Node Setup

7.1 Overview

Deploy Open WebUI on Chimera with connections to Ollama instances on both GPU nodes, providing 136GB total VRAM (72GB + 64GB) for inference workloads.

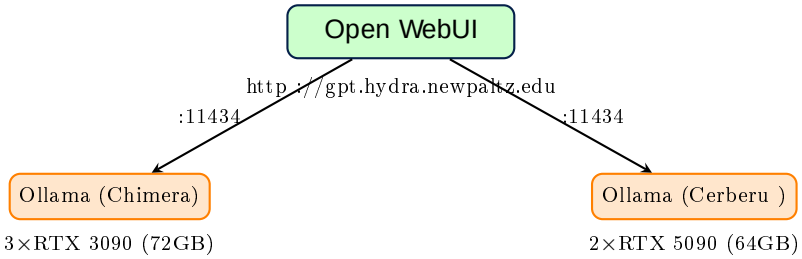


Figure 4: Open WebUI Multi-Node Architecture (136GB VRAM)

7.2 Multi-Node Setup Actions

2C.1 Install Ollama on Chimera

Install Ollama via o cial install script
Con gure OLLAMA_MODELS to /var/lib/ollama/models (fast NVMe)
Con gure OLLAMA_HOST=0.0.0.0 for network access
Enable and start ollama service

2C.2 Install Ollama on Cerberus

Install Ollama via o cial install script
Con gure OLLAMA_MODELS to /models/ollama
Con gure OLLAMA_HOST=0.0.0.0 for network access
Enable and start ollama service

2C.3 Pull Models

On Chimera (inference): llama3.1:70b, codellama:34b, mistral
On Cerberus (training): llama3.1:70b, deepseek-coder:33b

2C.4 Con gure Open WebUI

Update docker-compose.yml on Chimera
Set OLLAMA_BASE_URLS to include both localhost:11434 and cerberus:11434
Enable WEBUI_AUTH for authentication
Restart Open WebUI container

7.3 Model Distribution Strategy

Table 2: Recommended Model Placement

Node	Models	Rationale
Chimera (72GB)	llama3.1:70b, codellama:34b, mistral	Primary inference, lower latency
Cerberus (64GB)	llama3.1:70b, deepseek-coder, ne-tunes	Training workloads, large batches

Load Balancing: Open WebUI automatically distributes requests across available Ollama instances. If Cerberus is busy with training, requests route to Chimera.

8 Phase 3: RKE2 Kubernetes Deployment

8.1 Overview

Deploy RKE2 (Rancher Kubernetes Engine 2) across all three nodes with Hydra as the control plane and Chimera/Cerberus as GPU workers.

8.2 Why RKE2 over Docker Swarm

Table 3: RKE2 vs Docker Swarm Comparison

Feature	RKE2	Docker Swarm
GPU Scheduling	Native NVIDIA plugin	Manual placement
Resource Limits	Fine-grained (millicores)	Basic
Heterogeneous GPUs	Node selectors/taints	Di cult
Container Isolation	Pod security policies	Limited
SAML/OIDC Auth	Native support	External tools
Production Grade	CNCF certi ed	Community only

8.3 Cluster Topology

Table 4: RKE2 Cluster Con guration

Node	Role	Labels	Taints
Hydra	Server (control plane)	role=control	
Chimera	Agent (GPU worker)	gpu-type=rtx3090, gpu-count=3	

3.4 Con gure kubectl Access

Copy kubecon g from /etc/rancher/rke2/rke2.yaml
Set KUBECONFIG environment variable
Verify with kubectl get nodes

3.5 Install NVIDIA Device Plugin

Deploy NVIDIA k8s-device-plugin DaemonSet
Verify GPUs detected with kubectl describe nodes

8.5 Phase 3 Veri cation

Success Criteria:

All 3 nodes show Ready status
Chimera shows nvidia.com/gpu: 3
Cerberus shows nvidia.com/gpu: 2
System pods running in kube-system namespace

9 GPU Job Scheduling

9.1 Scheduling Strategy

Table 5: GPU Request Routing

Request Type	Target Node	Policy
Training jobs	Cerberus	Never interrupt running jobs
Inference requests	Chimera	Route if <50% utilization
Default containers	Hydra	CPU-only, no GPU

9.2 Container Migration Flow

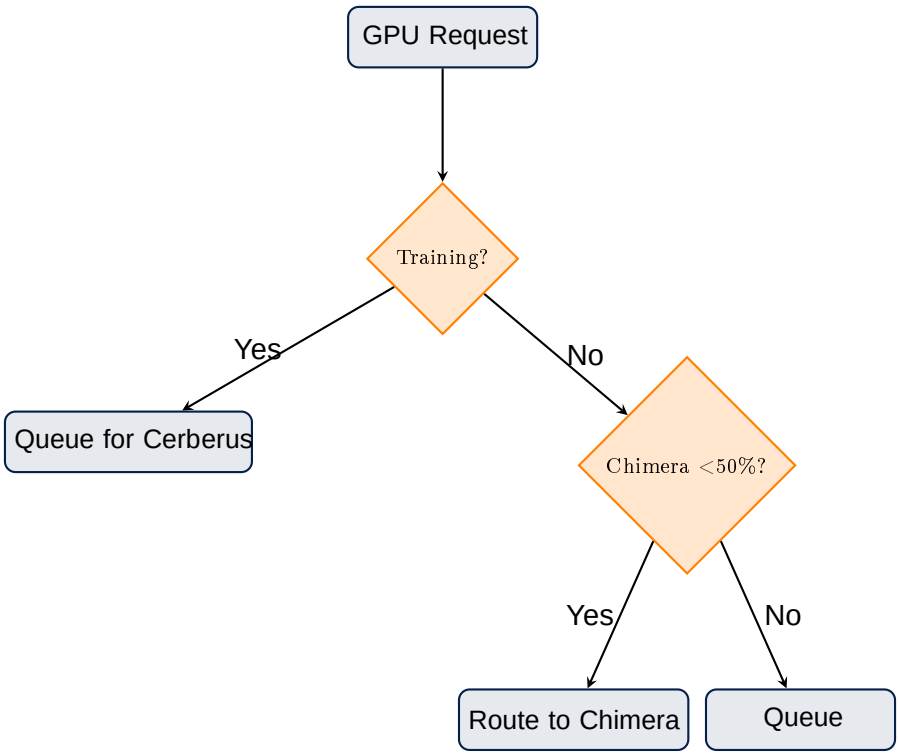


Figure 5: GPU Request Routing Flow

9.3 Migration Actions

Q.1 Container Migration Script

Export container from source node (docker export)
Transfer to /tank/staging via NFS
Import on target GPU node (docker import)
Start with GPU runtime ags

Q.2 Simple Queue Manager

Track pending GPU requests in SQLite database
Monitor GPU utilization on both nodes
Process queue when capacity available
Notify users via email when job starts

Q.3 Dashboard GPU Request Form

Add GPU request form to student dashboard
Fields: container name, GPU type preference, duration estimate
Submit to queue manager
Display queue position and estimated wait time

10 Implementation Timeline

10.1 Phase 0: Pre-Migration Backup (Day 0)

- ☐ Verify Chimera /data has space (3.5TB)
- ☐ Create backup directories at /data/backups/
- ☐ Backup Hydra: /home, databases, Docker volumes
- ☐ Backup Chimera: /home, Ollama models, Docker volumes
- ☐ Backup Cerberus: /home, /etc
- ☐ Verify all backups completed successfully

10.2 Phase 1: Hydra Storage (Day 1)

- ☐ Verify 6×7TB drives present
- ☐ Install ZFS
- ☐ Create RAID-10 pool (3 mirror pairs)
- ☐ Format sdh (1.1TB) for daily OS backups
- ☐ Create datasets: active, inactive, staging
- ☐ Con gure NFS exports
- ☐ Setup daily backup cron job
- ☐ Setup cluster aliases
- ☐ Setup centralized logging
- ☐ Verify Phase 1 complete

10.3 Phase 2: GPU Node Storage (Day 2)

- ☐ Chimera: Verify drive layout
- ☐ Chimera: Test drive speeds
- ☐ Chimera: Con gure Ollama on fast NVMe
- ☐ Chimera: Verify NVIDIA drivers
- ☐ Chimera: Verify Phase 2A complete
- ☐ Cerberus: Verify drive layout
- ☐ Cerberus: Format nvme1n1 for /models
- ☐ Cerberus: Con gure Ollama
- ☐ Cerberus: Verify NVIDIA drivers
- ☐ Cerberus: Verify Phase 2B complete

10.4 Phase 2C: Open WebUI Multi-Node (Day 2)

- ☐ Install Ollama on Chimera (models on fast NVMe)
- ☐ Install Ollama on Cerberus (models at /models)
- ☐ Pull models on both nodes
- ☐ Con gure Open WebUI with both Ollama endpoints
- ☐ Verify both nodes accessible
- ☐ Test inference from Open WebUI

10.5 Phase 3: RKE2 Deployment (Day 3)

- ☐ Install RKE2 server on Hydra
- ☐ Con gure and start RKE2 server
- ☐ Retrieve node token
- ☐ Install RKE2 agent on Chimera
- ☐ Install RKE2 agent on Cerberus
- ☐ Con gure kubectl access
- ☐ Install NVIDIA device plugin
- ☐ Verify all nodes Ready with GPUs

11 Risk Assessment

Table 6: Risk Matrix

Risk	Severity	Mitigation
Data loss during migration	High	Complete backups before any changes
Drive failure in RAID-10	Medium	Can lose 1 drive per mirror; maintain hot spare
Slow /data performance	Medium	Never use for active workloads; NVMe only
GPU driver incompatibility	Low	Test on one node before cluster-wide deploy
NFS performance issues	Low	10GbE network; async exports; local staging

12 Future Enhancements

Student Self-Service Portal: Web interface for GPU access requests

Monitoring Dashboard: Grafana dashboards for GPU/storage metrics

Automated Scaling: Scale containers based on GPU utilization

In niBand Integration: RDMA for distributed training workloads

Model Registry: Centralized model versioning and deployment

13 Appendix: Technology References

Table 7: Technology Documentation

Technology	Documentation URL
ZFS on Linux	https://openzfs.github.io/openzfs-docs/
RKE2	https://docs.rke2.io/
NVIDIA Device Plugin	https://github.com/NVIDIA/k8s-device-plugin
Ollama	https://ollama.com/docs
Open WebUI	https://docs.openwebui.com/