

Hydra Infrastructure

Complete Reference Manual

RKE2 Kubernetes Cluster
Student Container Platform
GPU Computing Infrastructure

SUNY New Paltz — Computer Science Department

Infrastructure Team

February 9, 2026

Contents

I	System Overview	5
1	Introduction	5
1.1	Key Features	5
1.2	Access URLs	5
2	Cluster Architecture	6
2.1	Node Inventory	6
2.2	Architecture Diagram	6
2.3	Network Architecture	6
3	Storage	7
3.1	Hydra Storage Layout	7
3.2	Kubernetes Storage Classes	7
3.3	NFS Configuration	7
II	Kubernetes Services	8
4	Namespace Layout	8
5	Core Services (hydra-system)	8
5.1	Traefik (Reverse Proxy)	8
5.2	Hydra Auth (SAML Gateway)	8
5.3	CS Lab Website	9
5.4	IngressRoute Summary	9
6	Infrastructure Services (hydra-infra)	9
6.1	Ollama (LLM Inference)	9
6.2	OpenWebUI	9
6.3	n8n (Workflow Automation)	10

6.4	Ray Cluster (Distributed Computing)	10
6.5	Other Services	10
7	GPU Infrastructure	10
7.1	NVIDIA GPU Operator	10
7.2	GPU Allocation	11
III	Authentication System	12
8	SAML 2.0 SSO Flow	12
9	Session and JWT Management	12
10	Cross-Service Authentication	12
10.1	OpenWebUI Account Provisioning	12
10.2	n8n Account Provisioning	12
10.3	CS Lab JWT Verification	13
IV	Student Containers	14
11	Container Features	14
11.1	Service Approval Gating	14
12	Container Presets	14
13	Resource Presets	15
14	SSH Access via SSHPiper	15
15	Container Labels and Routing	15
V	Networking	16
16	Firewall Configuration (UFW)	16
16.1	Hydra (Control Plane)	16
16.2	Chimera (GPU Worker)	16
16.3	Cerberus (GPU Worker)	16
17	Router Port Forwarding	16
18	DNS	16
VI	Deployment and Operations	18
19	Ansible Playbooks	18
19.1	Playbook Execution Order	18
19.2	What 05-deploy-hydra.yml Deploys	18
20	CS Lab Website Deployment	18
21	Image Management	19

22 Backup System	19
22.1 Daily Cluster Backups	19
22.2 etcd Snapshots	19
22.3 Backup Exclusions	19
23 Common Operations	19
23.1 Kubectl Shortcuts	19
23.2 Service Management	20
 VII OpenWebUI API Integration	 21
24 Getting Started	21
25 API Configuration	21
26 cURL Example	21
27 Python Example	21
28 JavaScript Example	21
 VIII Security	 23
29 Security Architecture Layers	23
30 Known Vulnerabilities	23
31 Security Best Practices for Students	23
 IX RDMA and GPUDirect	 24
32 Overview	24
33 Installation Order (Critical)	24
34 SoftRoCE Setup	24
35 GPUDirect RDMA Verification	24
 X Troubleshooting	 26
36 Authentication Issues	26
37 Container Issues	26
38 GPU Issues	26
39 Networking Issues	27
40 Traefik Deployment Issues	27
41 CS Lab Website Catch-All Route	27

XI Repository Structure	28
42 hydra-saml-auth	28
43 Other Repositories	29
XII Environment Configuration	30
44 Required Variables (hydra-saml-auth)	30
45 Ansible Inventory Variables	30
Appendices	31
A Cleanup History (February 2026)	31
B Migration History	31
C February 9, 2026 — Infrastructure Overhaul	31
D February 9, 2026 — Jenkins Service + Jupyter Gating + Repo Cleanup	32
E References	32

Part I

System Overview

1 Introduction

Hydra is a containerized development platform providing persistent development environments for Computer Science students and faculty at SUNY New Paltz. The system runs on a 3-node RKE2 Kubernetes cluster with GPU acceleration for AI/ML workloads.

1.1 Key Features

- **SSO Authentication:** Azure AD SAML 2.0 with automatic user provisioning
- **Persistent Containers:** Per-student development environments with SSH, VS Code, and Jupyter
- **GPU Computing:** 5 GPUs across 2 nodes (3x RTX 3090 + 2x RTX 5090)
- **AI Chat:** OpenWebUI + Ollama LLM inference (gpt.hydra.newpaltz.edu)
- **Ray Cluster:** Distributed computing framework for ML training
- **Dynamic Routing:** Traefik reverse proxy with ForwardAuth
- **Workflow Automation:** n8n with integrated user management
- **21 TB Storage:** RAID-10 ZFS array with NFS exports

1.2 Access URLs

Service	URL	Description
Dashboard	https://hydra.newpaltz.edu/dashboard	Main user interface
OpenWebUI	https://gpt.hydra.newpaltz.edu/	AI chat (Ollama)
CS Lab Site	https://hydra.newpaltz.edu/	Department homepage
Student Forms	https://hydra.newpaltz.edu/student-forms	Form hub
Hackathons	https://hydra.newpaltz.edu/hackathons	Hackathon voting
VS Code	https://hydra.newpaltz.edu/students/{user}/vscode	Browser IDE
Jupyter	https://hydra.newpaltz.edu/students/{user}/jupyter	Notebooks
n8n	https://n8n.hydra.newpaltz.edu/	Workflow automation
Servers	https://hydra.newpaltz.edu/servers	Cluster status

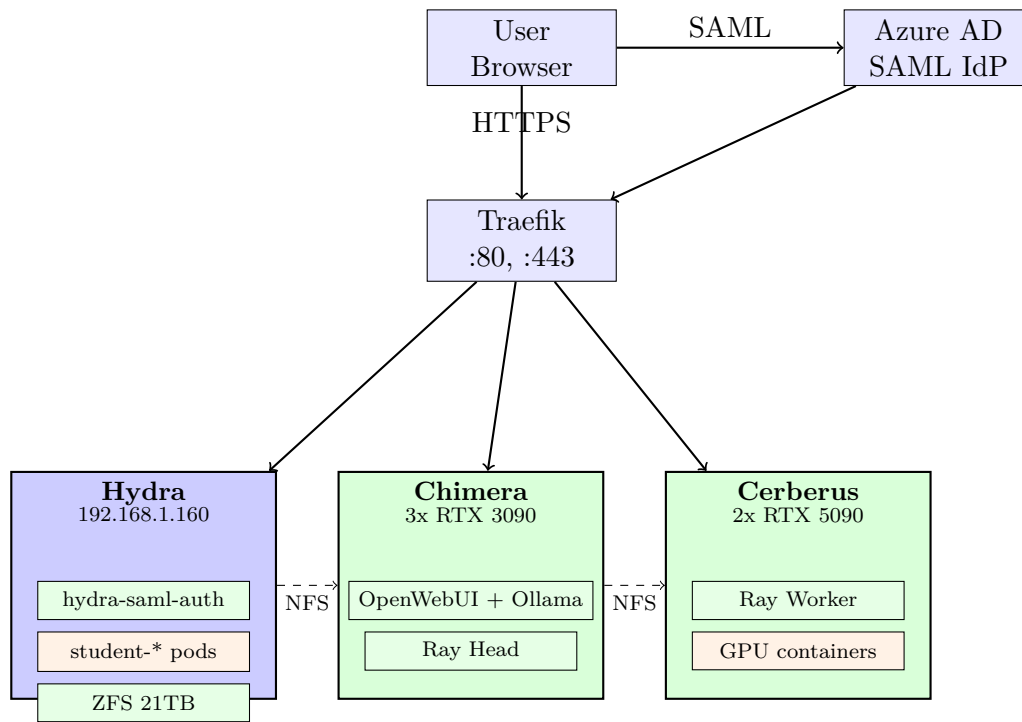
2 Cluster Architecture

2.1 Node Inventory

Node	IP	Role	OS	Hardware
Hydra	192.168.1.160	Control plane, etcd	Ubuntu 22.04.5	64 cores, 256 GB RAM, 21 TB RAID-10
Chimera	192.168.1.150	GPU inference	Ubuntu 24.04.2	48 cores, 256 GB RAM, 3x RTX 3090 (72 GB VRAM)
Cerberus	192.168.1.242	GPU training	Ubuntu 24.04.3	48 cores, 64 GB RAM, 2x RTX 5090 (64 GB VRAM)

Table 1: All nodes run RKE2 v1.28.4+rke2r1 with containerd 1.7.7.

2.2 Architecture Diagram



2.3 Network Architecture

- All nodes on 192.168.1.0/24 LAN (gateway 192.168.1.1)
- Direct ethernet bridge between Chimera and Cerberus (reserved for RDMA/RoCE)
- WireGuard VPN: Chimera wg0 = 10.8.0.2, Cerberus wg0 = 10.8.0.3
- Flannel VXLAN (port 8472/udp) for K8s pod networking, restricted to LAN
- UFW firewall on all nodes — workers expose only SSH publicly

3 Storage

3.1 Hydra Storage Layout

Device	Mount	Size	Purpose
/dev/mapper/ubuntu-vg-*	/	1 TB	OS, applications
/dev/md0 (RAID-10, 6 SSDs)	/data	21 TB	Student volumes, K8s PVCs
/dev/sdh4	/mnt/sdh4	1.1 TB	Daily backups

```
# RAID-10 details
/dev/md0: 6 active devices (sda-sdf), Chunk 512K, Layout near=2
State: clean, ext4, 4096-byte blocks
```

3.2 Kubernetes Storage Classes

Name	Provisioner	Usage
hydra-local	rancher.io/local-path	Student PVCs (default)
hydra-nfs	nfs.csi.k8s.io	Cross-node shared storage

3.3 NFS Configuration

Hydra exports /data/containers to the cluster LAN:

```
# /etc/exports on Hydra
/data/containers 192.168.1.0/24(rw, sync, no_root_squash)
```

CSI-NFS runs as a DaemonSet on all 3 nodes for dynamic PV provisioning.

Part II

Kubernetes Services

4 Namespace Layout

Namespace	Contents
hydra-system	Core platform: traefik, hydra-auth, cs-lab-backend, cs-lab-db
hydra-infra	Infrastructure services: ollama, open-webui, n8n, hackathons, java-executor, git-learning, sshpipec, ray-head, ray-worker
hydra-students	Student container pods (25+ active)
gpu-operator	NVIDIA GPU operator, device plugin, DCGM exporter
kube-system	RKE2 system: etcd, coredns, canal, metrics-server, CSI-NFS
local-path-storage	Local-path provisioner

5 Core Services (hydra-system)

5.1 Traefik (Reverse Proxy)

Traefik v2.11 serves as the cluster ingress controller. It runs on Hydra with `hostPort` binding on ports 80, 443, and 6969. The deployment uses `strategy: Recreate` to avoid `hostPort` conflicts during rolling updates.

Port	Name	Purpose
80	web	HTTP (redirects to HTTPS)
443	websecure	HTTPS with Let's Encrypt
6969	hydra-auth	Direct auth service access

Manifests: `k8s/components/traefik/`

5.2 Hydra Auth (SAML Gateway)

The main authentication and container management service. Handles:

- SAML 2.0 SSO via Azure AD
- JWT cookie issuance and JWKS endpoint
- Student container lifecycle (create, start, stop, delete)
- Dashboard UI, admin panel
- OpenWebUI and n8n account provisioning
- WebSocket terminal bridge

Manifests: `k8s/components/hydra-auth/`

5.3 CS Lab Website

React frontend + Express backend + SQLite database (single pod). Serves the department homepage at `hydra.newpaltz.edu`. MariaDB was removed on Feb 9, 2026 — the codebase uses `better-sqlite3` exclusively.

Component	Port	Image
cs-lab	5001	newpaltz-cs-lab-website-backend:latest

Database: SQLite at `/app/server/data/cslab.db`. 15 tables including Admins, Events, Faculty, Courses, StudentHighlightBlog, TechBlog, etc. Persisted via PVC `cs-lab-data`.

Manifests: `k8s/components/cs-lab/`

5.4 IngressRoute Summary

Name	Namespace	Match	Backend
hydra-main	hydra-system	hydra.newpaltz.edu catch-all	hydra-auth:6969
cs-lab-website	hydra-system	/api/ prefix	cs-lab-backend:5001
hydra-default	hydra-system	HTTP redirect	HTTPS redirect
hackathons	hydra-infra	/hackathons/ prefix	hackathons:45821
java-executor	hydra-infra	/java/ prefix	java-executor:55392
git-learning	hydra-infra	/git/ prefix	git-learning:8080
n8n	hydra-infra	n8n.hydra.newpaltz.edu	n8n:5678
openwebui	hydra-infra	gpt.hydra.newpaltz.edu	openwebui-chimera:3000

6 Infrastructure Services (hydra-infra)

6.1 Ollama (LLM Inference)

Runs on Chimera with all 3 RTX 3090 GPUs. Serves LLM models (gemma3:12b, etc.) via the Ollama API on port 11434.

Manifests: `k8s/components/ollama/`

Ollama requests all 3 GPUs on Chimera. Other GPU workloads on Chimera (like Ray head) must **not** request GPU resources, or they will conflict.

6.2 OpenWebUI

AI chat frontend at `gpt.hydra.newpaltz.edu`. Connects to Ollama for inference. Includes a **middleman sidecar** container for user account management.

Middleman API (port 7070):

- POST `/openwebui/api/check-user` — Check if user exists
 - POST `/openwebui/api/create-account` — Create new user
 - POST `/openwebui/api/change-password` — Update password
- Authentication via `x-api-key` header with timing-safe comparison.

Source: `k8s/components/openwebui/middleman/index.js`

Manifests: `k8s/components/openwebui/`

6.3 n8n (Workflow Automation)

Workflow automation platform at `n8n.hydra.newpal.tz.edu`. Uses PostgreSQL for data storage.

Components:

- n8n application (port 5678)
- PostgreSQL 16 (StatefulSet with PVC)
- n8n User Manager API (port 3000)

n8n User Manager API:

- GET /health — Health check (no auth)
- GET /api/users — List all users (auth required)
- GET /api/users/:email — Get user by email
- POST /api/users/change-password — Change password

Authentication via `x-api-key` header.

Source: `k8s/components/n8n/user-manager/`

Manifests: `k8s/components/n8n/`

6.4 Ray Cluster (Distributed Computing)

Ray provides distributed computing for ML training and inference.

Component	Node	GPU	Purpose
ray-head	Chimera	None (coordinator)	Scheduling, dashboard
ray-worker	Cerberus	2x RTX 5090	Training compute

Manifests: `k8s/components/ray/`

6.5 Other Services

Service	Port	Description
hackathons	45821	Hackathon voting/judging app (Vue.js + Express)
java-executor	55392	Remote Java code execution service
git-learning	8080	Interactive Git learning environment
sshpiper	2222	SSH proxy routing to student containers

Manifests: `k8s/components/{hackathons,java-executor,git-learning,sshpiper}/`

7 GPU Infrastructure

7.1 NVIDIA GPU Operator

The GPU Operator runs in the `gpu-operator` namespace and manages:

- Device plugin (exposes GPUs to K8s scheduler)
- Container toolkit (`nvidia-container-runtime`)
- GPU Feature Discovery (node labels)
- DCGM Exporter (GPU metrics)
- CUDA validator (verifies GPU access)

Hydra Exclusion: Hydra (control plane) has no GPUs. All `nvidia.com/gpu.deploy.*` labels are set to false on Hydra to prevent GPU operator pods from scheduling there.

7.2 GPU Allocation

Node	GPUs	Model	VRAM	Primary Consumer
Chimera	3	RTX 3090	72 GB total	Ollama (all 3)
Cerberus	2	RTX 5090	64 GB total	Ray Worker / Student GPU containers

Part III

Authentication System

8 SAML 2.0 SSO Flow

1. User visits <https://hydra.newpaltz.edu/login>

2. Hydra redirects to Azure AD with SAML AuthnRequest

3. User authenticates with SUNY New Paltz credentials

4. Azure AD returns signed SAML assertion

5. Hydra validates signature, extracts: email, groups, displayName

6. Session created, JWT cookie (np_access) issued

7. User redirected to /dashboard

9 Session and JWT Management

- **Express Session:** Server-side storage in SQLite
- **JWT Cookie (np_access):** Site-wide SSO cookie
- **JWKS Endpoint:** /.well-known/jwks.json for public key distribution
- **Algorithm:** RS256
- **TTL:** Configurable via JWT_TTL_SECONDS (default: 86400)
- **Cookie Domain:** .newpaltz.edu

10 Cross-Service Authentication

10.1 OpenWebUI Account Provisioning

When a user logs in via SAML, Hydra automatically provisions an OpenWebUI account via the middleman API. The password is derived and set transparently.

10.2 n8n Account Provisioning

Similarly, n8n accounts are provisioned via the n8n User Manager API on first login.

10.3 CS Lab JWT Verification

The CS Lab backend verifies JWT tokens using the public key mounted via ConfigMap `cs-lab-jwt-key` at `/app/server/keys/jwt-public.pem`.

Part IV

Student Containers

11 Container Features

Each student receives a persistent container with:

Feature	Details
Node.js	Latest LTS via nvm
Python	3.11+ with pip, venv, Jupyter
Java	OpenJDK 21
Docker	Docker CLI (Docker-in-Docker via rootless)
VS Code	code-server browser IDE (port 8443, always on)
Jupyter	JupyterLab (port 8888, approval required)
Jenkins	CI/CD server (port 8080, approval required)
SSH	Direct access via SSHPiper (port 2222)
Tools	Git, curl, wget, build-essential, gdb, cmake

11.1 Service Approval Gating

Each student container runs three managed services via `supervisord`. Code-server starts automatically; Jupyter and Jenkins require admin approval before they can be started.

Service	Port	Path	Autostart	Approval
code-server	8443	/students/{user}/vscode/	Yes	None required
Jupyter Lab	8888	/students/{user}/jupyter/	No	jupyter_execution_approved
Jenkins	8080	/students/{user}/jenkins/	No	jenkins_execution_approved

Approval flow:

1. Student clicks “Request Access” on the dashboard
2. Request stored in `resource_requests` table (type: `jupyter_execution` or `jenkins_execution`)
3. Admin approves via admin panel, setting the flag in `user_quotas`
4. On next container init, the `JUPYTER_APPROVED` / `JENKINS_APPROVED` env var is injected
5. `entrypoint.sh` creates `/var/run/jupyter-approved` or `/var/run/jenkins-approved` marker
6. Supervisor starts the approved service

CLI gating: Jupyter also has a CLI wrapper (`jupyter-gate.sh`) that blocks direct `jupyter` command usage. The real binary is at `/usr/local/bin/jupyter.real`. Students see an error directing them to request access via the dashboard.

12 Container Presets

- **Jupyter:** `jupyter/minimal-notebook`, port 8888, `ForwardAuth`
- **Static:** `nginx:alpine`, port 80, no auth
- **Repo:** Cloned from GitHub, runtime varies (Node/Python/nginx)
- **VS Code:** `codercom/code-server`, mounts any project volume

13 Resource Presets

Preset	RAM	CPU	Storage	GPU	Approval
Minimal	256 MB	0.5	5 GB	0	Auto
Conservative	512 MB	1	10 GB	0	Auto
Standard	1 GB	1	20 GB	0	Auto
Enhanced	2 GB	2	40 GB	0	Required
GPU Inference	32 GB	8	100 GB	1	Required
GPU Training	48 GB	16	200 GB	2	Required

14 SSH Access via SSHPiper

Students access containers via SSH through the SSHPiper proxy:

```
# Connect to your container
ssh -p 2222 student@hydra.newpaltz.edu

# Port 2222 is forwarded through the router to the sshpiper K8s pod
```

- SSHPiper routes connections based on username to the correct student pod
- Passwords displayed in dashboard after container creation
- Key-based auth supported (~/.ssh/authorized_keys)

15 Container Labels and Routing

Common labels on student containers:

- `hydra.managed_by=hydra-saml-auth`
- `hydra.owner=<username>`
- `hydra.project=<project>`
- `hydra.basePath=/students/<user>/<project>`

Traefik routes requests at `/students/<user>/<project>` to the corresponding container using StripPrefix middleware (except Jupyter, which uses `base_url`).

Part V

Networking

16 Firewall Configuration (UFW)

16.1 Hydra (Control Plane)

22/tcp	ALLOW	Anywhere	# SSH
80/tcp	ALLOW	Anywhere	# HTTP
443	ALLOW	Anywhere	# HTTPS
6969	ALLOW	172.17.0.0/16, 172.24.0.0/16	# Auth (Docker)
51820/udp	ALLOW	Anywhere	# WireGuard
6443/tcp	ALLOW	192.168.1.0/24	# K8s API
9345/tcp	ALLOW	192.168.1.0/24	# RKE2 supervisor
10250/tcp	ALLOW	192.168.1.0/24	# Kubelet
2379:2380/tcp	ALLOW	192.168.1.0/24	# etcd
2222/tcp	ALLOW	Anywhere	# SSHPiper
2049/tcp	ALLOW	192.168.1.0/24	# NFS
111/tcp, udp	ALLOW	192.168.1.0/24	# portmapper
8472/udp	ALLOW	192.168.1.0/24	# Flannel VXLAN

16.2 Chimera (GPU Worker)

22/tcp	ALLOW	Anywhere	# SSH
7070/tcp	ALLOW	192.168.1.148	# OpenWebUI middleman
9100	ALLOW	192.168.1.0/24	# Metrics
8472/udp	ALLOW	192.168.1.0/24	# Flannel VXLAN
10250/tcp	ALLOW	192.168.1.0/24	# Kubelet

16.3 Cerberus (GPU Worker)

22/tcp	ALLOW	Anywhere	# SSH
9100	ALLOW	192.168.1.160	# Metrics from Hydra
2376	ALLOW	192.168.1.160	# Docker from Hydra
8472/udp	ALLOW	192.168.1.0/24	# Flannel VXLAN
10250/tcp	ALLOW	192.168.1.0/24	# Kubelet

17 Router Port Forwarding

External Port	Internal IP	Internal Port	Service
22	192.168.1.160	22	Admin SSH
80	192.168.1.160	80	HTTP
443	192.168.1.160	443	HTTPS
2222	192.168.1.160	2222	Student SSH (SSHPiper)

18 DNS

- hydra.newpal.tz.edu — Main domain, points to campus public IP

- gpt.hydra.newpal.tz.edu — OpenWebUI subdomain
 - n8n.hydra.newpal.tz.edu — n8n subdomain
- TLS certificates managed by Let's Encrypt via Traefik ACME.

Part VI

Deployment and Operations

19 Ansible Playbooks

The cluster can be deployed from scratch using Ansible playbooks in `ansible/`:

```
cd /home/infra/hydra-saml-auth/ansible
ansible-playbook -i inventory.yml playbooks/site.yml
```

19.1 Playbook Execution Order

1. `00-preflight-backup.yml` — Create backups before changes
2. `01-prepare-nodes.yml` — Install packages, configure kernel
3. `02-rke2-server.yml` — Setup RKE2 control plane on Hydra
4. `03-rke2-agents.yml` — Join Chimera and Cerberus to cluster
5. `04-gpu-setup.yml` — Configure NVIDIA drivers and GPU Operator
6. `05-deploy-hydra.yml` — Deploy all K8s manifests

19.2 What `05-deploy-hydra.yml` Deploys

In order:

1. Namespaces and RBAC
2. Storage classes
3. Traefik CRDs and deployment
4. Hydra Auth deployment
5. CS Lab website (backend + DB)
6. Hackathons, Java Executor, Git Learning
7. SSHPiper
8. n8n (app + Postgres + user manager)
9. Ollama
10. OpenWebUI (with middleman sidecar)
11. Ray cluster (head + worker)

20 CS Lab Website Deployment

```
# 1. Build the image
cd /home/infra/NewPal tz-CS-Lab-Website
docker build --no-cache -t newpal tz-cs-lab-website-backend:latest .

# 2. Export to tarball
docker save newpal tz-cs-lab-website-backend:latest \
  -o /data/containers/images/newpal tz-cs-lab-website-backend-latest.tar

# 3. Import into RKE2's containerd
sudo ctr --address /run/k3s/containerd/containerd.sock \
  -n k8s.io images import \
  /data/containers/images/newpal tz-cs-lab-website-backend-latest.tar
```

```
# 4. Restart the pod
kubectl delete pod -l app.kubernetes.io/component=backend -n hydra-system
```

Docker vs RKE2 Containerd: Docker and RKE2 use separate containerd instances with separate image stores. Docker builds go to Docker's containerd. You must explicitly import images into RKE2's containerd at `/run/k3s/containerd/containerd.sock`.

21 Image Management

- Image tarballs stored at `/data/containers/images/`
- Use `imagePullPolicy`: Never for locally-imported images
- Use unique tags (e.g., `v20260206144528`) to force pod recreation

22 Backup System

22.1 Daily Cluster Backups

Setting	Value
Backup Location	<code>/mnt/sdh4/backups/</code>
Schedule	Daily at 1:00 AM (crontab)
Method	rsync with compression
Script	<code>/home/infra/backup-cluster.sh</code>
Log File	<code>/var/log/cluster-backup.log</code>

22.2 etcd Snapshots

Automatic every 12 hours via RKE2. Stored in `/var/lib/rancher/rke2/server/db/snapshots/`.

22.3 Backup Exclusions

`/dev/*, /proc/*, /sys/*, /run/*, /tmp/*, /var/tmp/*, /var/cache/*, /mnt/*, /var/lib/docker/*, /var/lib/rancher/*`

23 Common Operations

23.1 Kubectl Shortcuts

Sourced from `~/hydra-aliases`:

```
k          # kubectl
kgp        # kubectl get pods -A
kgs        # kubectl get svc -A
students   # list student pods
hydra-health # quick cluster health check
gpu-check  # GPU availability per node
```

23.2 Service Management

```
# View all pods
kubectl get pods -A

# Restart a deployment
kubectl rollout restart deployment/<name> -n <namespace>

# View logs
kubectl logs -f deployment/<name> -n <namespace>

# Execute shell in pod
kubectl exec -it <pod-name> -n <namespace> -- /bin/bash
```

Part VII

OpenWebUI API Integration

24 Getting Started

1. Log in at <https://hydra.newpaltz.edu/dashboard>
2. Visit <https://gpt.hydra.newpaltz.edu>
3. Go to Settings → Account → Generate New API Key
4. Copy the key (format: sk-...) — shown only once

25 API Configuration

```
ENDPOINT=https://gpt.hydra.newpaltz.edu/api/chat/completions
MODEL=gemma3:12b
API_KEY=sk-your-api-key-here
```

26 cURL Example

```
curl https://gpt.hydra.newpaltz.edu/api/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer sk-your-api-key-here" \
-d '{
  "model": "gemma3:12b",
  "messages": [{"role": "user", "content": "Hello!"}]
}'
```

27 Python Example

```
import openai, os
openai.api_base = "https://gpt.hydra.newpaltz.edu/api"
openai.api_key = os.getenv("HYDRA_API_KEY")

response = openai.ChatCompletion.create(
    model="gemma3:12b",
    messages=[{"role": "user", "content": "Hello!"}]
)
print(response.choices[0].message.content)
```

28 JavaScript Example

```
const response = await fetch(
  'https://gpt.hydra.newpaltz.edu/api/chat/completions',
  {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
```

```
    'Authorization': 'Bearer ' + API_KEY
  },
  body: JSON.stringify({
    model: 'gemma3:12b',
    messages: [{role: 'user', content: 'Hello!'}]
  })
}
);
const data = await response.json();
console.log(data.choices[0].message.content);
```

Additional language examples (PHP, Java, C#, Ruby, Go, Rust) are available in the full API access guide at [docs/access.md](#).

Part VIII

Security

29 Security Architecture Layers

1. **Network:** UFW firewall, TLS encryption, CORS policy
2. **Authentication:** SAML 2.0, JWT tokens, API keys
3. **Authorization:** Role-based access, container ownership labels
4. **Runtime:** Container isolation, resource limits, seccomp profiles

30 Known Vulnerabilities

Critical:

- **Privileged containers** in Docker mode grant full host access
- **Docker socket mount** in student containers is equivalent to root on host

High:

- Passwordless sudo for student user in container images
- Supervisor web interface (port 9001) without authentication
- Mining detection without automatic enforcement
- K8s pod security context missing `runAsNonRoot`, `allowPrivilegeEscalation`: `false`

Medium:

- No NetworkPolicy isolation between student pods
- No PID limits (fork bomb vulnerability)
- Jupyter/VS Code without application-level auth (relies on ForwardAuth)

See docs/SECURITY_VULNERABILITIES.md for full details and remediation steps.

31 Security Best Practices for Students

- Never share API keys publicly or commit to version control
- Use environment variables for sensitive configuration
- Rotate API keys regularly
- Use HTTPS only for all API communications
- Validate and sanitize user inputs before sending to API

Part IX

RDMA and GPUDirect

32 Overview

The cluster supports RDMA networking for high-performance GPU-to-GPU communication:

Node	NIC	GPUs	RDMA
Hydra	Onboard	None	SoftRoCE (testing)
Chimera	ConnectX	3x RTX 3090	Hardware RoCE
Cerberus	ConnectX	2x RTX 5090	Hardware RoCE + GPUDirect

33 Installation Order (Critical)

1. MLNX_OFED / DOCA (network drivers)
2. NVIDIA GPU Drivers (includes nvidia-peermem)
3. CUDA Toolkit
4. Load nvidia-peermem module

If the NVIDIA GPU driver is installed before MLNX_OFED, the driver must be reinstalled to compile nvidia-peermem with RDMA APIs.

34 SoftRoCE Setup

```
# Install prerequisites
sudo apt install rdma-core ibverbs-utils perftest

# Create SoftRoCE device
sudo rdma link add rxe0 type rxe netdev eth0

# Verify
rdma link && ibv_devices
```

35 GPUDirect RDMA Verification

```
# Load nvidia-peermem
sudo modprobe nvidia-peermem

# Make persistent
echo "nvidia-peermem" | sudo tee /etc/modules-load.d/nvidia-peermem.conf

# Test bandwidth (two nodes)
# Server: ib_write_bw -d mlx5_0 --use_cuda=0
# Client: ib_write_bw -d mlx5_0 --use_cuda=0 <server_ip>
```


See docs/rdma-gpudirect-setup.md for complete SR-IOV, DOCA, and KVM passthrough configuration.

Part X

Troubleshooting

36 Authentication Issues

Symptom	Solution
SAML assertion invalid	Verify <code>METADATA_URL</code> and <code>SAML_SP_ENTITY_ID</code> match Azure config
Cookie not set	Check <code>COOKIE_DOMAIN</code> , ensure HTTPS
JWT verification fails	Check JWKS endpoint accessible, verify key rotation

37 Container Issues

Symptom	Solution
Container won't start	Verify student container image exists in RKE2 containerd
Container 404	Check Traefik is running, container has correct labels
VS Code not loading	Check code-server process, ForwardAuth middleware
Jupyter issues	Verify <code>base_url</code> setting matches path
SSH not working	Check SSHPiper pod, verify port 2222 routing
Files not persisting	Only <code>/home/student/</code> is persisted via PVC

38 GPU Issues

Symptom	Solution
GPU not detected	Run <code>nvidia-smi</code> on host, check NVIDIA drivers
GPU pod pending	Check GPU operator pods in <code>gpu-operator</code> namespace
Ollama can't use GPU	Verify all 3 GPUs allocated to Ollama deployment
Ray worker can't use GPU	Check NVIDIA device plugin on Cerberus

39 Networking Issues

Symptom	Solution
Service unreachable	Check pod is Running, service exists, Ingress-Route matches
502 Bad Gateway	Backend pod crashed or port mismatch
TLS certificate error	Check Traefik ACME, run <code>certbot renew --dry-run</code>
NFS mount failed	Verify NFS server running on Hydra, firewall allows 2049
Cross-node pod issue	Check Flannel VXLAN (8472/udp) allowed between nodes

40 Traefik Deployment Issues

Stuck Rolling Update: Traefik uses `hostPort` which means only one pod can bind ports 80/443 at a time. The deployment MUST use strategy: `Recreate` (not `RollingUpdate`). If stuck:

```
kubectl rollout undo deployment/traefik -n hydra-system
```

41 CS Lab Website Catch-All Route

The Express server has a catch-all that serves `index.html` for SPA routes. Backend API paths are excluded:

```
# Paths excluded from SPA catch-all (served by backend):
/api/*, /faq, /faculty, /uploads, /scripts, /tech-blog,
/student-resources, /student-highlights, /admins, /auth,
/school-calendar, /sd-forms

# Paths explicitly allowed through for SPA routing:
/student-forms, /submit-*
```

If adding new frontend routes starting with `/student`, update the catch-all in `server.js`.

Part XI

Repository Structure

42 hydra-saml-auth

```

hydra-saml-auth/
|-- index.js                # SAML auth, JWT, routes, WebSocket
|-- routes/
|   |-- containers.js       # Container lifecycle
|   |-- resource-requests.js # Resource allocations
|   |-- webui-api.js        # OpenWebUI proxy
|   |-- n8n-api.js          # n8n account management
|   |-- servers-api.js      # Cluster status
|   |-- admin.js            # Admin panel
|-- services/
|   |-- db-init.js          # Database init
|   |-- resource-expiry.js   # Resource expiry checker
|   |-- security-monitor.js  # Process monitoring
|-- config/
|   |-- resources.js         # Presets and node config
|   |-- runtime.js          # Docker/K8s switcher
|-- k8s/
|   |-- base/               # Namespace, RBAC, storage
|   |-- components/
|       |-- traefik/        # Reverse proxy
|       |-- hydra-auth/     # Auth service
|       |-- cs-lab/         # CS Lab website
|       |-- ollama/         # LLM inference
|       |-- openwebui/      # AI chat + middleman
|       |-- n8n/            # Workflows + user manager
|       |-- ray/            # Distributed computing
|       |-- hackathons/     # Hackathon app
|       |-- java-executor/  # Code execution
|       |-- git-learning/   # Git learning
|       |-- sshpipec/       # SSH proxy
|       |-- student-pods/   # Pod templates
|   |-- gpu/               # GPU operator config
|-- ansible/
|   |-- inventory.yml        # Node definitions
|   |-- playbooks/          # Deployment scripts
|-- student-container/
|   |-- Dockerfile          # Student image
|   |-- supervisord.conf    # Process manager
|-- docs/                   # This document + sources
|-- docker-compose.yaml     # Legacy Docker deployment

```

43 Other Repositories

Repo	Path	Description
NewPaltz-CS-Lab-Website	~/NewPaltz-CS-Lab-Website/	React + Express CS Lab homepage
Hackaton-Voting	~/Hackaton-Voting/	Vue.js hackathon app

Part XII

Environment Configuration

44 Required Variables (hydra-saml-auth)

Variable	Description
BASE_URL	https://hydra.newpaltz.edu
METADATA_URL	Azure AD federation metadata URL
SAML_SP_ENTITY_ID	SP Entity ID (must match Azure)
COOKIE_DOMAIN	.newpaltz.edu
PORT	Service port (default: 6969)
DB_PATH	SQLite path (/app/data/hydra.db)
JWT_TTL_SECONDS	Token lifetime (default: 86400)

45 Ansible Inventory Variables

```
rke2_version: "v1.28.4+rke2r1"
cluster_domain: hydra.newpaltz.edu
nfs_server: "192.168.1.160"
nfs_path: "/srv/hydra-nfs"
```

Appendices

A Cleanup History (February 2026)

A comprehensive infrastructure cleanup was performed February 4–7, 2026:

Node	Action	Reclaimed
Hydra	Docker system prune	114.8 GB
Hydra	Remove stale files (/opt/local -path-provisioner.bak, temp files)	20+ GB
Hydra	Truncate backup log	389 MB
Chimera	Remove Docker Ollama duplicate + prune	41.2 GB
Cerberus	Docker system prune	51.3 GB
Total		~227 GB

Key cleanup actions:

- Migrated all services from Docker containers to K8s pods
- Archived legacy/ directory to legacy-archive git branch
- Relocated middleman sources to k8s/components/ directories
- Removed stale Apache configs, scripts, temp files across all nodes
- Cleaned orphaned Docker networks, volumes, and images
- Fixed Traefik stuck rolling update (added strategy: Recreate)
- Fixed Ray cluster (removed GPU request from head, deployed properly)
- Verified all middleman APIs operational
- Cloned hydra-saml-auth repo to all 3 nodes

B Migration History

The infrastructure evolved through several phases:

1. **Bare metal** — Apache web server, manual user management
2. **Docker Compose** — Containerized services, Nginx reverse proxy
3. **K3s** — Initial Kubernetes, migrated from Docker Compose
4. **RKE2** — Current production cluster (January 2026), Traefik ingress
5. **Infrastructure Overhaul** — February 9, 2026 (see below)

C February 9, 2026 — Infrastructure Overhaul

Following a 5-hour OOM death spiral that made the server unresponsive, the following changes were applied:

1. **Phase 1: Docker cleanup** — Pruned 11.86 GB orphaned Docker volumes, stopped and disabled Docker daemon on all nodes. Build tool changed to buildah (daemonless).
2. **Phase 2: Networking** — Removed nginx (conflicted with Traefik on port 80), fixed SUNYCAT.png route, fixed OpenWebUI cross-namespace reference, corrected OpenWebUI API fallback URL.
3. **Phase 3: CS Lab consolidation** — Removed MariaDB pod (app uses SQLite). Went from 2 pods (backend + MariaDB) to 1 pod (cs-lab). Removed mariadb npm dependency.
4. **Phase 4: RKE2 data migration** — Moved /var/lib/rancher/rke2 (40 GB) to RAID at /data/rke2. Symlinked old path for backward compatibility. Config updated: data-dir: /data/rke2.

5. **Phase 5: OOM prevention** — Added 32 GB swap on RAID (vm. swappiness=10), kubelet eviction thresholds (memory. available<2Gi hard, 4Gi soft), system-reserved=4Gi, kube-reserved=2Gi. Applied ResourceQuota and LimitRange to hydra-students namespace (default 2Gi/1CPU per container, max 48Gi/16CPU). Enabled auto-reboot after kernel updates at 04:00.

Build tools after overhaul:

- buildah — Daemonless OCI image builder (primary)
- nerdctl — Docker-compatible CLI for containerd (/usr/local/bin/nerdctl)
- Docker daemon is **disabled** (systemctl disable docker)

D February 9, 2026 — Jenkins Service + Jupyter Gating + Repo Cleanup

1. **Jupyter execution gating** — Supervisor autostart=false, CLI gate wrapper (jupyter-gate.sh), API approval endpoints, JUPYTER_APPROVED env var marker. Students cannot run Jupyter until admin approves.
2. **Jenkins CI/CD service** — Added as 3rd managed service inside student containers. Supervisor config (port 8080, autostart=false), K8s pod/service/IngressRoute/strip-prefix routing, DB schema (jenkins_execution_approved), admin approval flow, dashboard UI card with Start/Stop/Open buttons.
3. **Repo cleanup** — Removed 12 dead Docker-era files (deploy scripts, old Python metrics agent, Apache config, SSHPiper Docker Compose, student-mvp, Ray reference compose files). Scrubbed plaintext secret files from git history. Added .example templates for K8s secrets.
4. **K8s template updates** — Added Jenkins port 8080 to pod template, Jenkins route/middleware/service port to IngressRoute template, flapjs deployment.
5. **Security** — Removed cs-lab/secret.yaml and n8n/secret.yaml from git tracking (contained plaintext credentials). Added to .gitignore. Credentials should be rotated.

E References

- RKE2 Documentation: <https://docs.rke2.io/>
- Traefik Documentation: <https://doc.traefik.io/traefik/>
- SAML 2.0 Spec: <https://docs.oasis-open.org/security/saml/v2.0/>
- Azure AD SAML: <https://learn.microsoft.com/en-us/entra/identity/>
- NVIDIA GPU Operator: <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/>
- OpenWebUI: <https://docs.openwebui.com>
- Ray: <https://docs.ray.io/>
- n8n: <https://docs.n8n.io/>