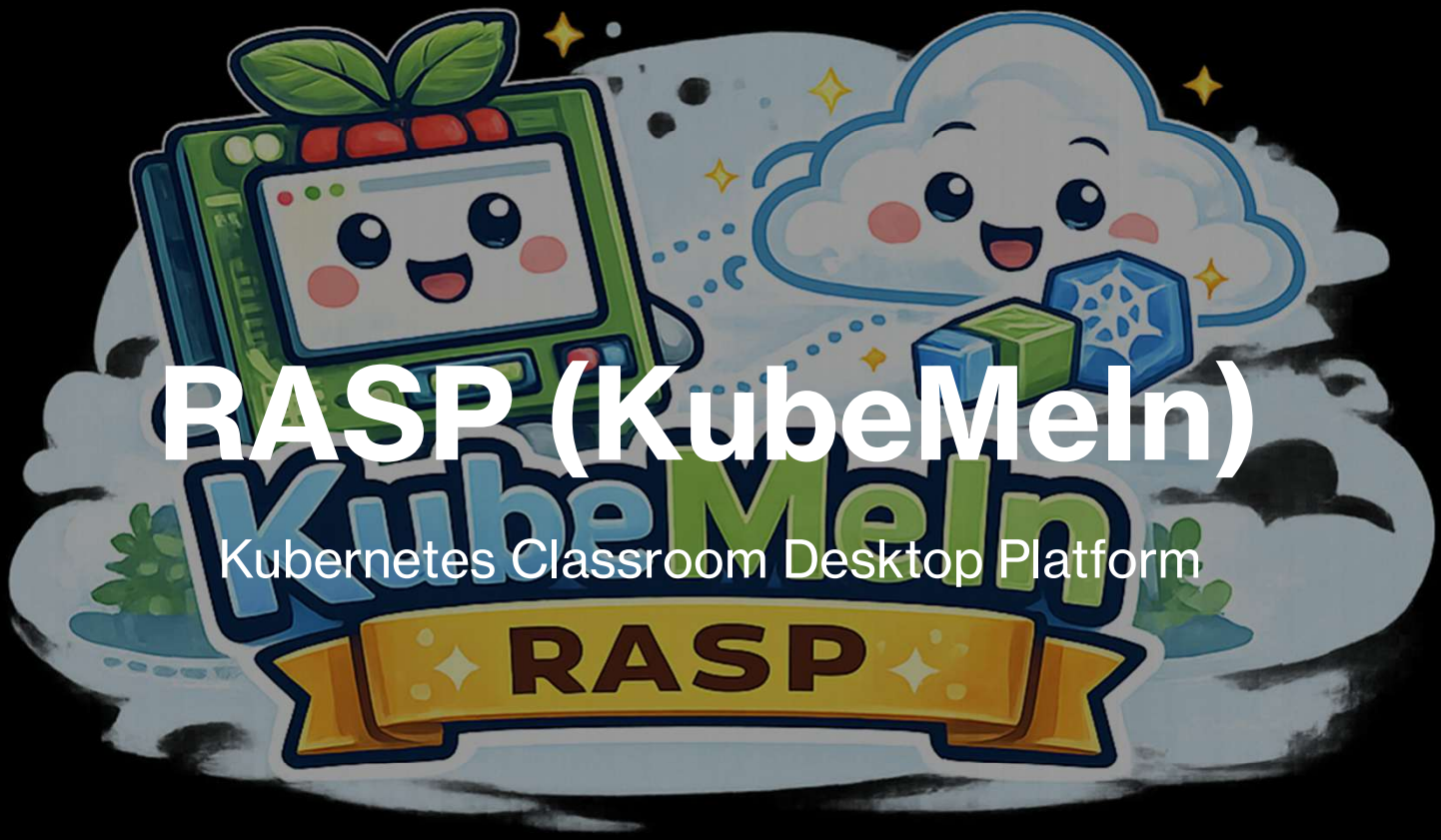


RASP (KubeMeIn)

Kubernetes Classroom Desktop Platform

The real problem

HIGH-LEVEL ARCHITECTURE

System Overview

- Raspberry Pi → Student access device
- Kubernetes (k3s) → Control plane
- Proxmox VE → Virtual machines
- NFS Server → Persistent storage

COMPUTE LAYER (Proxmox)

Virtual Machine Infrastructure

Proxmox VE hypervisor

Ubuntu 24.04 + XFCE desktops

Preconfigured “golden template” (Template 102)

EMPLATE 102 (GOLDEN TEMPLATE)

Includes:

- Ubuntu 24.04 + XFCE
- Student & teacher accounts
- NFS mount configuration
- QEMU Guest Agent
- Network via DHCP

Why it matters

- All new VMs are cloned from this template
- Fix once → all future VMs fixed
- Break once → all future VMs broken

ACCESS LAYER (Raspberry Pi)

Student Access Layer

- Raspberry Pi in kiosk mode
- Chromium browser only
- No local storage
- No VNC client required

CONTROL LAYER (KUBERNETES)

Why Kubernetes (k3s)?

- Runs system services reliably
- Automatically restarts failed services
- Handles scaling during peak usage

Why k3s

- Lightweight Kubernetes
- Low memory usage
- Perfect for lab environments

K3S CLUSTER SETUP

Cluster Topology

- Single-node k3s cluster
- Control plane and worker on same server
- IP: 10.0.96.99

Runs all backend services:

- Session Manager
- Pool Manager
- VM Controller
- Cleanup Service
- Teachers Portal
- PostgreSQL
- Guacamole

CORE SERVICE ROLES

Session Management

Decides which student gets which VM and ensures one active session per student.

VM Pool Management

Keeps a pool of pre-created desktops ready for instant access.

VM Control

Communicates with Proxmox to start, stop, clone, and delete virtual machines.

Cleanup & Monitoring

Removes expired sessions and recovers stuck or unused VMs.

Teacher Interface

Provides a web dashboard to view and control active sessions.

SUPPORTING SERVICES

Guacamole

Converts VM desktop connections (VNC) into HTML5 so desktops appear in a browser.

PostgreSQL

Stores session data, VM pool state, and Guacamole connection information.

NFS Server (External)

Stores student home directories so files persist across different VMs and sessions.

KUBERNETES SELF-HEALING

Self-Healing Mechanism

- Kubernetes checks /health
- Failed service is restarted automatically
- No manual intervention required

Result

- High availability
- Automatic recovery

Service	Liveness Probe	Interval	Restart Trigger
session-broker	`GET /healthz:8081`	10s	✅ Auto-restart
vm-controller	❌ None	-	⚠️ No auto-restart
pool-manager	`GET /healthz:8083`	30s	✅ Auto-restart
cleanup-service	`GET /healthz:8084`	30s	✅ Auto-restart
teachers-portal	`GET /health:8082`	10s	✅ Auto-restart
guacamole	`GET /guacamole/:8080`	20s	✅ Auto-restart
guacd	`TCP :4822`	10s	✅ Auto-restart
guac-postgres	`exec pg_isready`	10s	✅ Auto-restart
postgres	❌ None	-	⚠️ No auto-restart

KUBERNETES AUTO-SCALING

Service	Min	Max	Scale Triggers
session-broker	1	5	CPU 70%, Memory 80%
vm-controller	1	3	CPU 70%

Horizontal Pod Autoscaler (HPA)

- Monitors CPU usage
- Scales Session Manager during login spikes
- Scales down when load decreases

Why only Session Manager

- Login storms happen there
- Other services are backend-bound

FAILURE HANDLING

What Happens When Things Fail

- Service crash → Pod restarts
- Node reboot → Services recreated
- VM crash → Cleanup replaces it
- Student disconnects → Resources reclaimed

No single point of failure

If probe fails → Kubernetes removes pod from Service endpoints (no traffic)

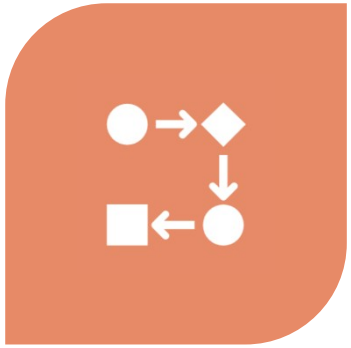
Service	Readiness Probe	Initial Delay	Check Interval	Traffic Control
session-broker	`GET /healthz:8081`	2s	5s	✅ Protected
vm-controller	`GET /healthz:8080`	3s	10s	✅ Protected
pool-manager	`GET /healthz:8083`	5s	10s	✅ Protected
cleanup-service	`GET /readyz:8084`	5s	10s	✅ Protected
teachers-portal	`GET /health:8082`	10s	5s	✅ Protected
guacamole	`GET /guacamole/:8080`	10s	10s	✅ Protected
guacd	`TCP :4822`	3s	5s	✅ Protected
guac-postgres	`exec pg_isready`	5s	5s	✅ Protected
postgres	❌ None	-	-	⚠️ Unprotected


Demo time
make a gif.com

PROMA

GOAL: STABLE END TO END PLATFORM + FULL EVIDENCE

WORKFLOW: 2 WEEK SPRINTS



CADENCE: PLAN → STAND-UP → REVIEW → RETRO (ALL DOCUMENTED)

Conclusion (technical + PM) and what we have learned

Technical outcomes
End-to-end remote desktop platform on Kubernetes
Automated VM lifecycle with Proxmox
Monitoring + health checks + autoscaling
Admin UI for session control
PM outcomes
Delivered within sprint timeline
Clear responsibilities and documentation
Demo aligned with rubric requirements
Next steps
Stronger auth (SSO), quotas, richer dashboards, HA database, improved security hardening

Q & A

THANK YOU!