Autonomous Control Planes, Direct L3 Routing, and Hardware Storage Locality

IBERGRID 2026 | Santiago de Compostela

Noctilucent: Optimized Bare-Metal Kubernetes Architecture

Speaker: **Samuel Bernardo**

LIP

Our path in Kubernetes

2019 – 2026: From Cloud IaaS Back to Bare Metal

- We started **our first** Kubernetes deployment using **kubeadm** on decommissioned servers
- **About 2 years later**, we moved to the cloud and started using **Kubespray** (Terraform + Ansible) to deploy **Kubernetes over OpenStack**
- **From 2023 to 2025**, with the project "Open Clouds for Research Environments" from EOSC, we migrated some services to Google **Kubernetes Engine (GKE)** on the Google Cloud Platform (GCP)
- **In 2026**, we deploy **Project Noctilucent bare-metal architecture** eliminating cloud abstractions and hypervisor overhead



Where the friction appears

State Latency:
Raft quorum churn under I/O jitter
NodeLocal DNS & conntrack bypass

Traffic Delivery:
Overlay encapsulation overhead
Direct L3 routing via frr-k8s & BGP

Storage Locality:
Mayastor NVMe-over-TCP IOPS
OpenZFS snapshotter & SeaweedFS

Node Lifecycle:
Deterministic host state machine
Serialized Kured reboot guards

Continuous Delivery:
FluxCD declarative reconciliation
Dual-sink containerd logging

HPC Integration:
Kubernetes to Slurm offloading
Slinky bridge & Keycloak OIDC

Bare-metal is not forgiving

We are in the real world, not in the clouds!
Routing, disks, bootstrapping, and failure domains remain visible all the way up to the platform.

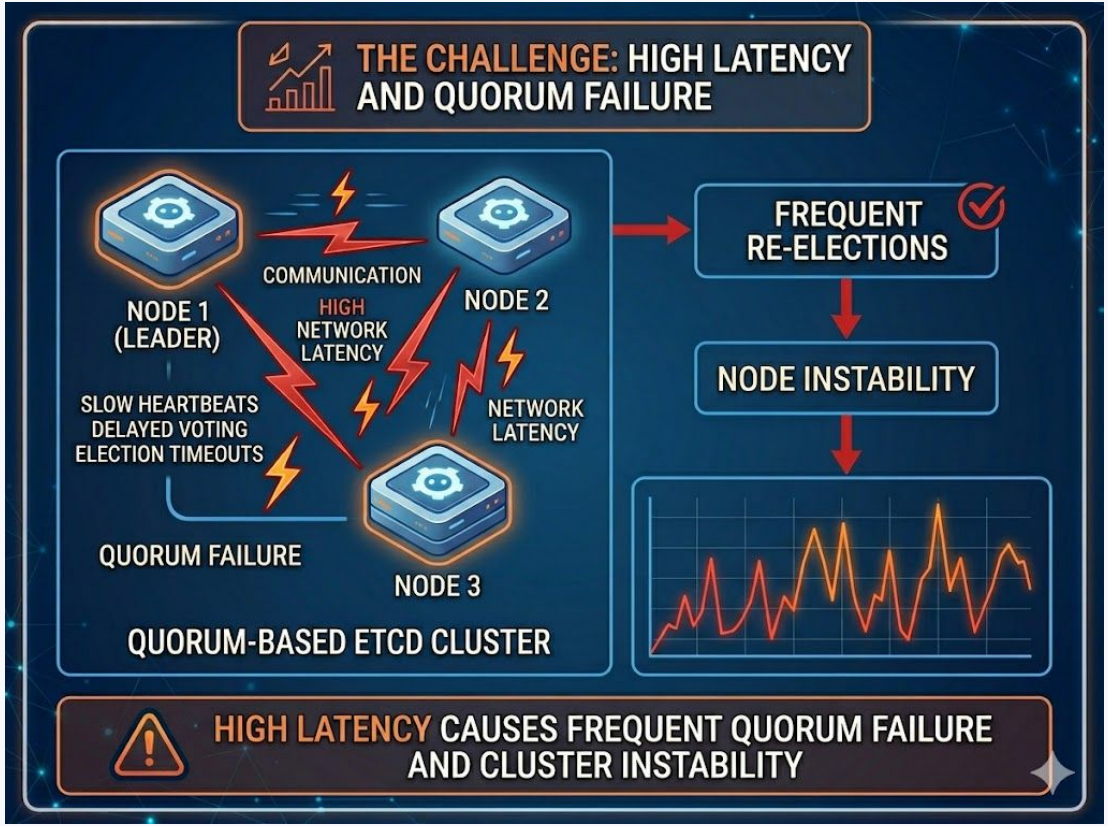
- **Zero cloud abstraction:** Hardware, raw disks, and failure domains are directly exposed to the operator.
- **Latency hurts first:** Jitter triggers cascading Raft elections and network transport bottlenecks.
- **Every hop and layer matters:** In-host datapath, BGP peering, and NVMe storage locality are essential.

Raft consensus under physical jitter

The Raft Quorum Problem

Disk I/O & Network Jitter: Bursty bare-metal workloads cause transient latency spikes on physical disks and links.

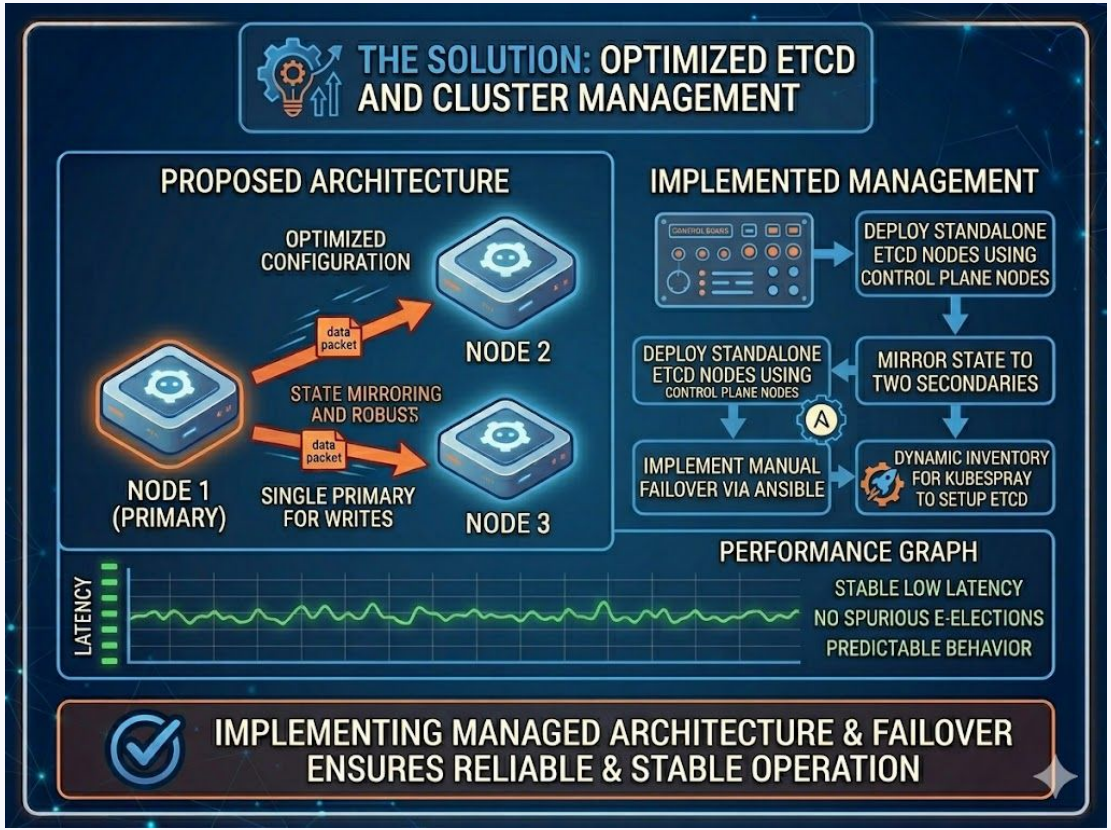
Cascaded Leader Elections: Spurious re-elections lock the Kubernetes control plane, inducing exponential API client retries and cluster freeze.



Primary-Secondary etcd Architecture

The Noctilucent Solution

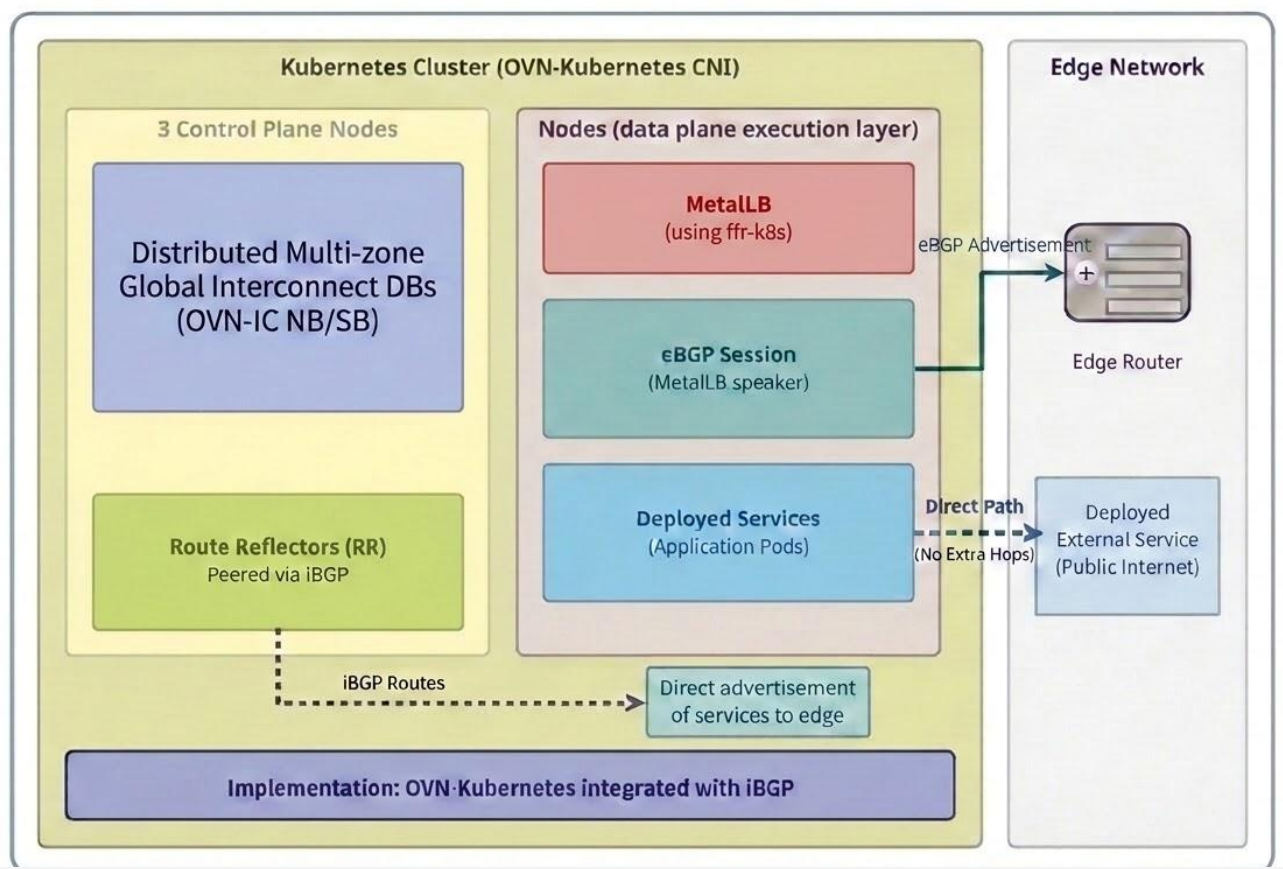
- **Single Primary Engine:** All writes target one node; zero quorum deadlocks.
- **Dual Mirrored Secondaries:** etcd-mirror-watcher streams state over mTLS.
- **systemd + Podman:** Managed cgroups and automated process restart.
- **Kubespray & Ansible:** Dynamic inventory & controlled administrative failover.



No-Overlay Transport: OVN MZA & BGP Routing

Standalone OVN Zones:
Per-node NBDB/SBDB eliminates centralized network database bottleneck.

Zero-Overlay Direct L3:
No Geneve/VXLAN encapsulation overhead. Control plane runs **frr-k8s iBGP Route Reflectors**; **MetalLB** advertises VIPs outwards to ToR switches via eBGP.



Host Datapath & Kernel Tuning

NodeLocal DNSCache (169.254.20.10)

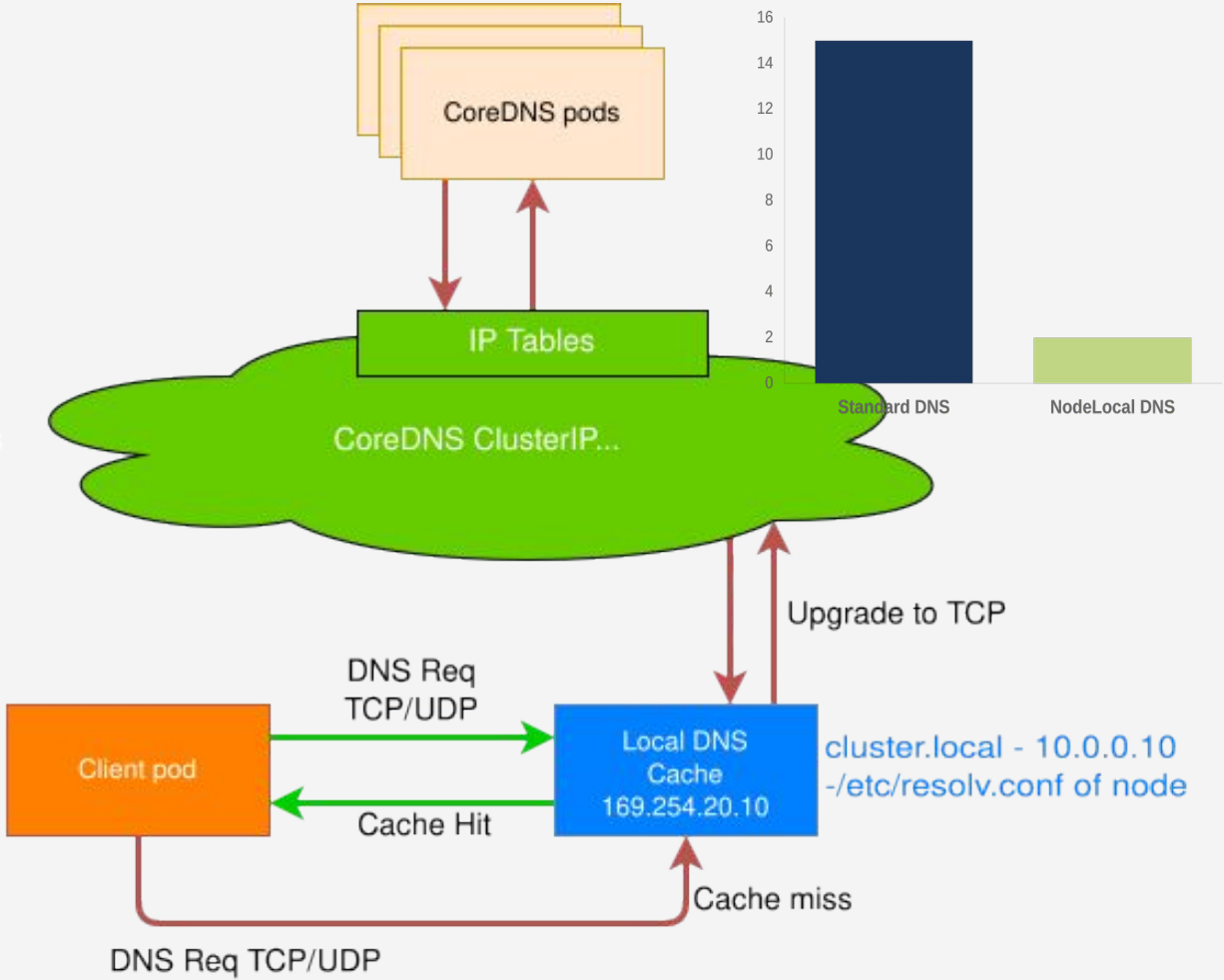
Bypasses iptables DNAT; prevents UDP DNS conntrack table exhaustion under heavy spikes.

MTU 9000 Jumbo Frames

Enforced across physical NICs, br-int, and pod veths to eliminate fragmentation.

rp_filter = 0 (Multi-VRF)

Disables reverse path filtering to prevent dropping asymmetric packets across complex bare-metal routes.



Storage locality is the critical bottleneck

Physical NVMe drives exist on every host — yet generic virtualized SAN/Ceph layers destroy bare-metal IOPS.

Noctilucent enforces strict hardware-adjacent locality via NVMe-oF and OpenZFS primitives.

Block Storage: OpenEBS Mayastor & NVMe-over-TCP

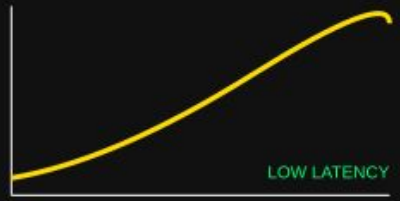
THE BARE-METAL PROBLEM



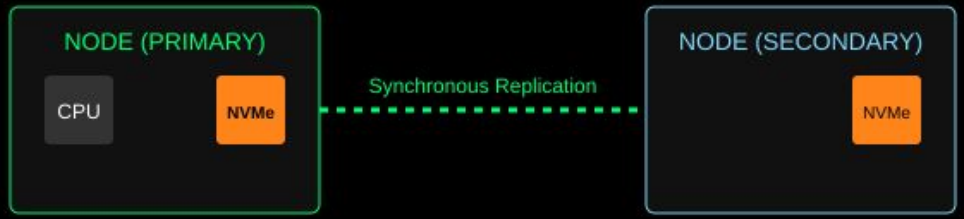
Local NVMe

Problem: Generic distributed storage adds latency

NVMe-oF PERFORMANCE



CHOSEN STACK: OPENEBS WITH MAYASTOR



1. Locality First
Data path near the node

2. Maintenance
Deliberate Pod/Volume moves

3. Explicit Placement
Strict workload-locality sync

Move the pod to the location where there is a volume replica

We leverage OpenEBS Mayastor to optimize NVMe-oF performance

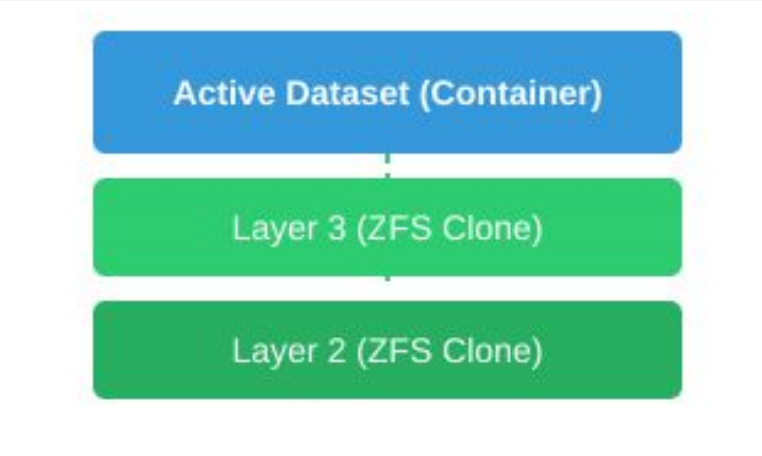
OpenZFS Snapshotter & SeaweedFS Shared Primitives

Copy-on-write

Instant snapshots and clones, with compression reducing footprint at the same time.

ARC cache

Single Copy Cache improves read efficiency without adding another storage layer.



ZFS changes the unit of container storage from a shared overlay to a managed dataset with native filesystem behavior.

Isolation and quotas

Dataset-per-image or container makes limits and separation explicit.

Checksumming

Data and metadata are verified, strengthening integrity from image to running container.

OpenZFS snapshotter: Instant CoW snapshots, LZ4 compression, ARC single-copy memory caching, replaces overlayfs.
SeaweedFS: Provides distributed ReadWriteMany (RWX) & high-throughput S3 object storage for AI/HPC datasets.

CONTAINER LOGGING & OBSERVABILITY

Operational Observability: Dual-Sink Container Logging

Zero-Overhead Log Multiplexing

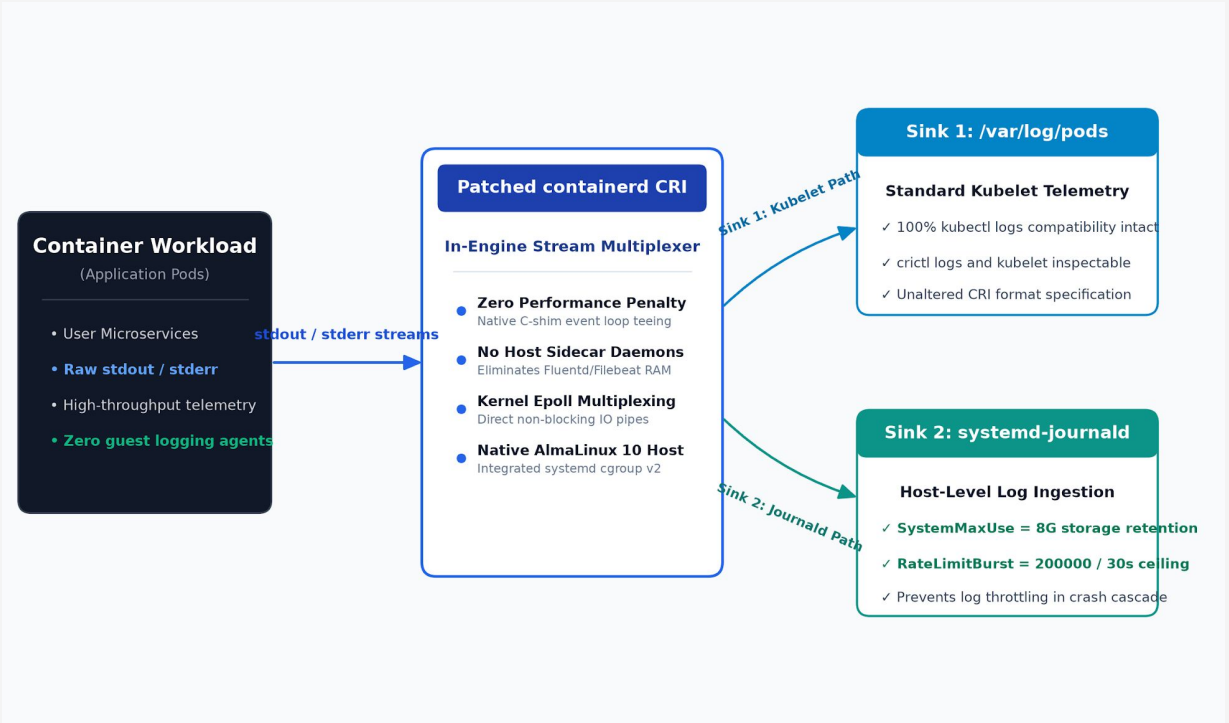
Standard containerd CRI isolates stdout/stderr into flat files, bypassing systemd-journald. Noctilucent deploys a **custom-patched CRI driver** to tee streams simultaneously without heavyweight sidecar daemons.

100% Kubelet Compliance

Preserves standard **/var/log/pods** layout ensuring **kubectl logs** and crictl remain completely intact while unifying host logs.

Crash-Loop Drop Protection

Host journal limits configured to **SystemMaxUse=8G** and **RateLimitBurst=200000/30s** to eliminate log throttling during crash cascades.



Dual-Sink Ingestion Architecture: Eliminates Fluentd/Filebeat daemonset RAM overhead while streaming 100% compliant pod telemetry to local journald harvesters.

Host Lifecycle Automation: Explicit Node State Machine

A node changes role only when the operator has completed the storage and cluster steps that make the next state safe.



PXE, TFTP, Kickstart deploy AlmaLinux 10 with xfs for root and ZFS datasets for containers.

Before touching the node move Mayastor volumes and evacuate pods somewhere else.

Only then join or return the node to Kubernetes for normal scheduling.

Deterministic Lifecycle: Kickstart AlmaLinux 10 + ZFS → Health Gate → Production → Replica Evacuation → Serialized Reboot via Kured.

Continuous Platform Delivery: GitOps with FluxCD



The question is not which tool looks better, but which one keeps **deployment flow, repository structure, and operational ownership** aligned.

CURRENT PRACTICE

Projects are deployed via an ArgoCD app using GitLab.

ALREADY IN USE

GitLab itself uses FluxCD, so both models co-exist on the platform.

OPERATIONAL OBSERVATION

ArgoCD leverages robust Web UI and multi-cluster support. But can mean **less commit churn**, but GitLab support favors Flux.

POSSIBLE DIRECTION

Move toward a **Flux-based GitOps path** if it simplifies onboarding and maintenance.

Unified GitOps Architecture: FluxCD continuously reconciles cert-manager, Kyverno, External Secrets, Crossplane, Backstage, and Keycloak.

WORK IN PROGRESS

Heterogeneous HPC Integration: K8s to Slurm

Why this matters

Kubernetes stays the service front end while heavier processing can leave the cluster when it makes more sense to run on Slurm.



Slurm bridge
used to offload processing needs from services deployed on Kubernetes

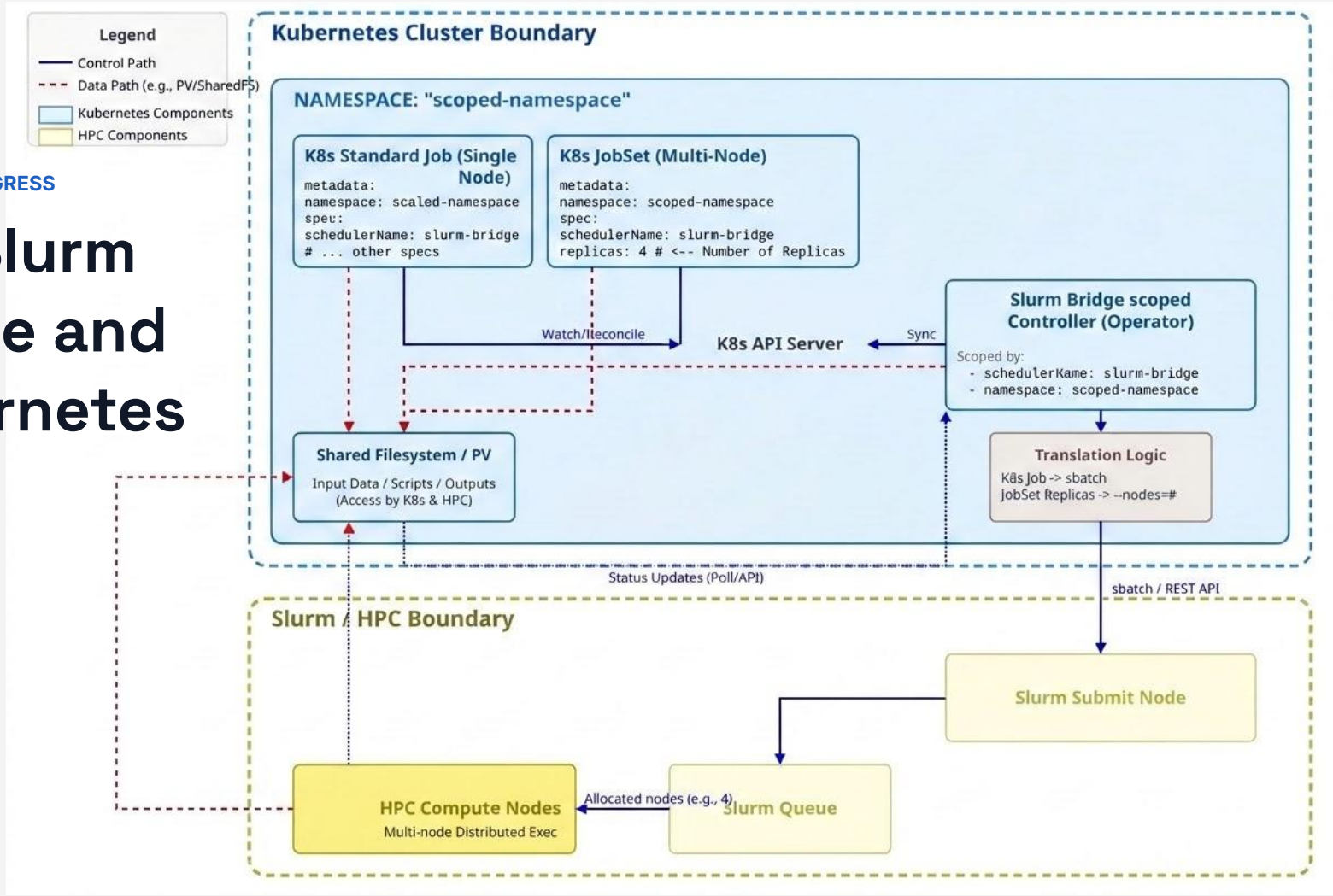
Keycloak AIM
provides the identity and access management layer

Internal IAP
a Node/Express implementation retrieves the token for the Slurm API

Architectural Boundary: Kubernetes frontends capture research workflows; Slinky operator bridges compute-intensive jobs to Slurm HPC clusters.

WORK IN PROGRESS

The Slurm Bridge and Kubernetes



Summary & Key Takeaways

PROJECT NOCTILUCENT ARCHITECTURE



Performance First

Address latency at every layer—from etcd quorum to NodeLocal DNS and NVMe storage.



Explicit Operations

Use state machines for node lifecycle and clear transitions to ensure stability.



Workload Awareness

Optimize data paths with storage locality and specialized offloading like Slurm.



Continuous Evolution

Align GitOps workflows and infrastructure primitives to match real-world lifecycles.

Thank You

Noctilucent: Optimized Bare-Metal Kubernetes Architecture

Engineering & Operations Team (LIP & CNCA)



Zacarias Benta
LIP - Minho



João Machado
CNCA



Samuel Bernardo [Speaker]
LIP



João Paulo Martins
LIP



César Ferreira
CNCA - Minho



Pedro Pinheiro
LIP



Jorge Gomes
LIP



João Pina
LIP

Conference Contribution
Scan for slides & details

Questions, Collaboration & Operational Inquiries
Contact Email: **helpdesk@lip.pt** • Conference: *IBERGRID 2026*

LIP - Laboratório de Instrumentação e Física Experimental de Partículas
Partículas e Tecnologia