Full-stack HPC Mobility

Portability Across the Holistic HPC Stack

A Path to Sovereign, Multi-Cloud HPC/AI Infrastructure

Dr. Gilles Fourestey (SCITAS/EPFL)
On Behalf of the SCITAS and CSCS Teams

The Problem: HPC systems cannot move

Today's HPC systems:

- cannot move
- cannot replicate
- cannot fail over

Why?

- They are **tightly coupled stacks**:
  - network, storage, compute
  - OS, identity, scheduler…
- Redeployment across platforms is:
  - time-consuming
  - fragile

The issue is not "just" code portability: it's environment portability

Why it Matters for HPC

Resilience requirements (existing HPC)

- Critical workloads (e.g., weather forecasting) require **georedundancy and failover**
- Current systems are difficult to replicate across sites

Reproducibility across infrastructures

- Scientific workflows must run consistently across centers
- Environment differences create friction

Elasticity requirements (AI & new workloads)

- Demand is variable and often exceeds local capacity
- Requires hybrid and multi-cloud execution

We already need HPC mobility for resilience and disaster control, AI just makes the lack of it impossible to ignore.

Why AI Workloads Require Mobility

AI workloads don’t run in one place

- **Demand is elastic**
  Training and experimentation create unpredictable GPU spikes
- **Data and compute are fragmented**
  Data is local, GPUs are elsewhere, collaborators are distributed
- **Workflows are multi-stage**
  Training, fine-tuning and inference use different optimal environments
- **Constraints vary**
  Cost, latency, sovereignty, availability

AI must run where data, compute, and constraints allow

SwissTwins: A Mobile HPC Ecosystem

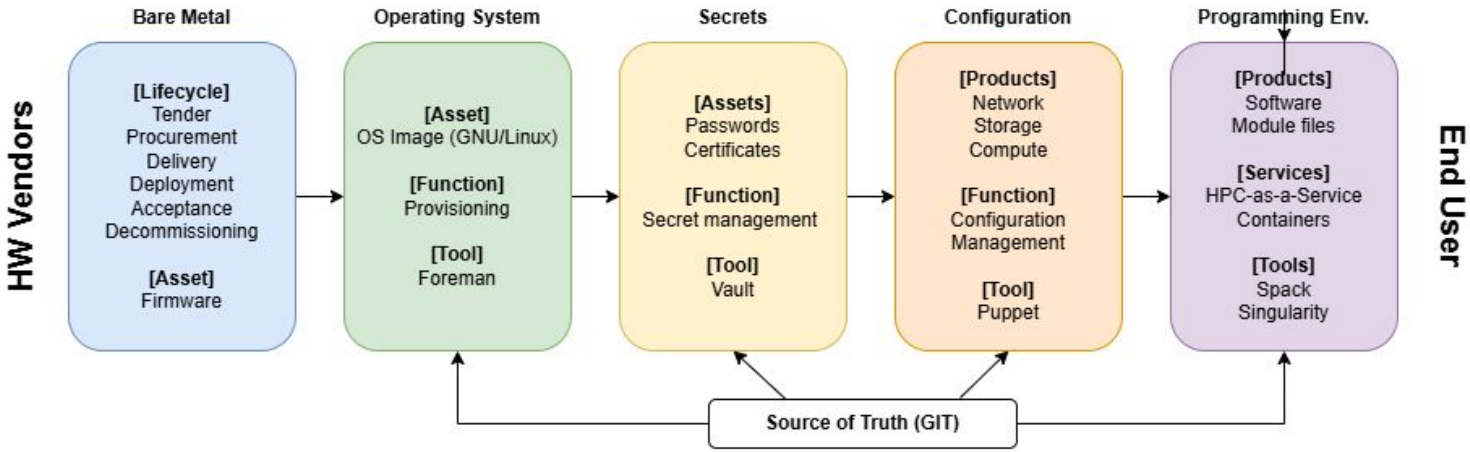
Funded by SERI (https://www.cscs.ch/about/collaborations/swisstwins)

- Enable Swiss research workloads to run seamlessly
  - On-Premises (Alps)
  - Public Cloud (GCP, AWS, Azure, Exoscale)
  - Institutional clusters (EPFL, ...)
- Goal:
  - **Sovereignty**: data and compute control
  - **Resilience**: instant fail-over
  - **Elasticity**: scale with demand

Our Approach:

- **Solution:** make CSCS' **vCluster** technology portable through a **Resource Portability Interface (RPI)**.
- **Strategy:** decouple any platform-specific dependencies from the vCluster environment using a IaC approach.

HPC End-to-End Infrastructure: Powerful But Rigid



Pros:

- Vertically integrated
- Tightly coupled
- Full control
- Maximum performance

Cons:

- Not portable
- Vendor lock-in
- Project life time can be very long (6+ years)
- Monolithic

Performance comes at the cost of flexibility

vCluster: A Software-defined HPC System

What is vCluster?

- Versatile Software-Defined Cluster technology from CSCS
- Originally designed for Alps supercomputer
- Cloud-native architecture without hypervisor performance penalty

Key properties:

- Fully described as code (IaC)
- Reproducible and version-controlled
- Independent from underlying infrastructure

Key distinction:

- Not just a container or VM
- Not just a Kubernetes cluster
- Represents the full operational HPC system

vCluster Technology Foundation

Three-Layer Architecture:

	Layer	Function
1	**Infrastructure**	- Compute, network, storage
2	**Services**	- identity, orchestration, CI/CD, secrets
3	**User Environments**	- User environment customization - Stack provisioning via Spack or containers - Application deployment

Key Innovation: decouples system definition from underlying hardware

Only the infrastructure layer changes across platforms

- Layers 2 & 3 are portable
- Layer 1 is adapted per platform

We move the system, not just the workload

Resource Portability Interface (RPI): The Key Abstraction

Problem: portability is broken

- Infrastructure differs across platforms:
  - cloud vs on-prem
  - networking models
  - compute provisioning

Solution: RPI defines a **common interface for infrastructure resources**

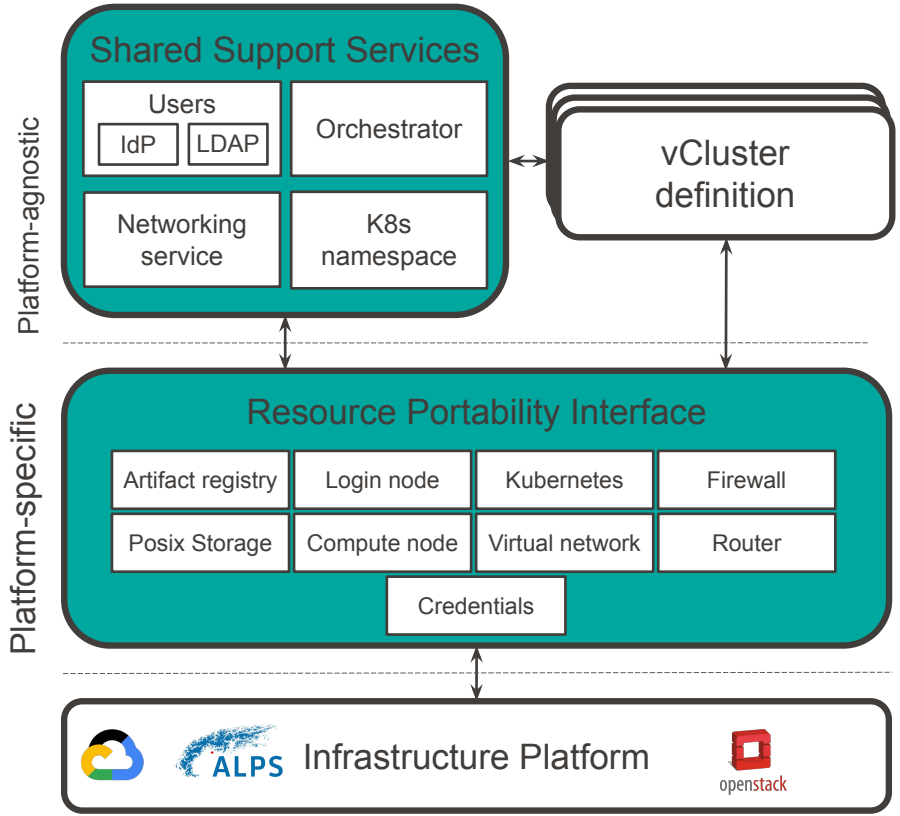
- Abstract resources:
  - compute nodes, networks, srotage
  - access control, ...
- Each platform provides its own implementation

How it works:

- vCluster defines **what resources it needs**
- RPI maps them to **platform-specific modules**

Key property:

- Same vCluster definition
- Runs unchanged across platforms



What is the “landing zone”?

The landing zone is the **minimal infrastructure foundation** required to host vClusters

It provides:

- Networking (VPC, routing)
- Core services (identity, orchestration, secrets)
- Shared resources (artifact registries, control plane)

Key points:

- **Created once** per platform
- **Reused** by multiple vClusters

It prepares the environment so clusters can be deployed consistently

Service	Role
Identity Management	User Authentication
Secret Management	Securely stores keys/tokens
Artifact Repositories	Trusted source of images (OS, containers)
Control Nodes	Hosts management
Orchestration	Manages lifecycle of control daemons
Source of Truth	Version-controlled configuration
CI/CD Pipelines	Automate provisioning and updates

From One Model To Many HPC Systems

Same abstraction, different systems:

- Minimal cluster → prototyping, development
- GPU cluster → AI workloads
- Slurm cluster → traditional HPC scheduling
- Application-specific cluster → tuned for workloads (e.g., ICON)

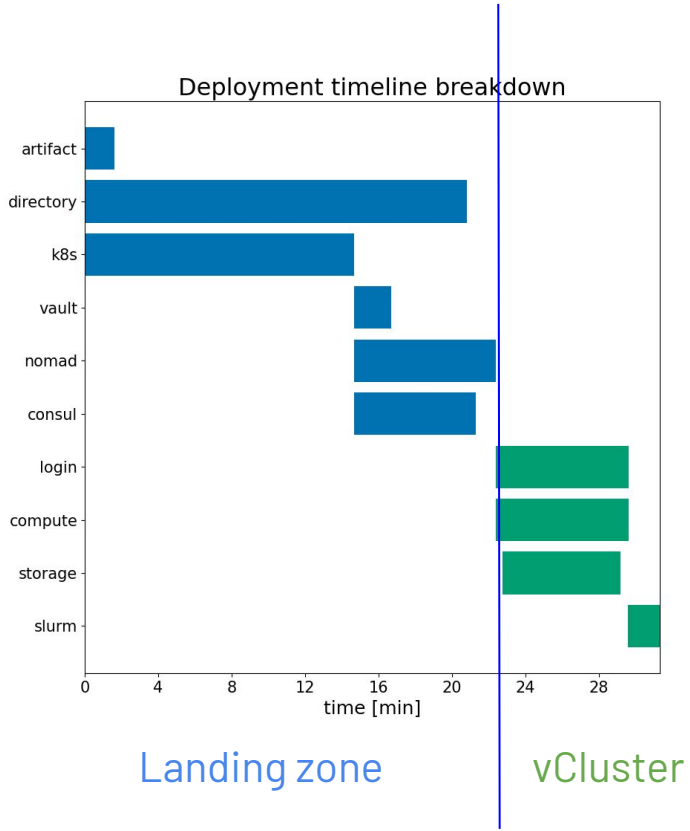
All defined using the same vCluster model

We don’t build clusters, we compose them

Example: GCP Implementation

Four-Step Deployment Process:

1. **Landing Zone Creation** (~23 min)
   - Virtual private network
   - GKE Kubernetes cluster
   - Artifact registries (RPM, DEB, Docker)
   - HashiCorp stack (Consul, Nomad, Vault)
   - OpenLDAP + Keycloak
2. **Artifact Generation**
   - RPM/DEB packages via Docker Bake
   - Custom Docker images for control plane services
3. **vCluster Instantiation** (~7 min)
   - Terraform modules
   - Provider-specific modules
4. **Software Stack Availability**
   - Spack deployment for user builds
   - Cloud-based build caches



Running ICON outside CSCS ALPS-E

Target Workload:

- ICON numerical weather forecasting model (**mch_icon-ch1**)
- Used by Swiss national weather service
- 33-hour simulation

Original Production Setup (Alps-E):

- 2 compute nodes × 64 AMD EPYC cores
- 4× NVIDIA A100 GPUs per node (8 total)
- HPE Slingshot interconnect

Kuma@EPFL:

- 4xH100 94GB
- Infiniband

GCP Configurations Tested:

Config	CPU	GPU	RAM	Storage
GCP Double	24 cores/node	4× A100	680 GB	Shared filestore
GCP Single	48 cores	8× A100	1360 GB	Local SSD

Platform	Time (s)
EPFL Kuma single node (H100)	**2281.58**
CSCS ALPS-E single node (A100)	**2664.16**
GCP Single-node (A100)	**3707.86**
GCP Two-nodes (A100)	**6822.58**

Cloud enables portability, but network topology limits multi-node performance.

Limitations & Considerations

Platform coverage

- Validated on Alps and GCP
- Expanding to additional providers (e.g., Exoscale)

Performance variability

- Workloads require tuning to underlying hardware
- Interconnect and topology impact multi-node performance

Operational complexity

- Multiple service domains (identity, networking, orchestration)
- Cross-site networking and latency remain challenging

Key Takeaways: HPC must become mobile

Static infrastructures limit:

- resilience
- reproducibility
- flexibility

Full-stack portability is **feasible today using vClusters and our RPI infrastructure**

It enables:

- sovereign and georedundant infrastructure
- multi-cloud execution
- AI and emerging workloads

We can now move HPC systems, not just workloads

Thank you for listening!

Questions?