CSCS’ journey towards complete platform automation in a multi-tenant environment

CUG25

Alejandro Dabin, Ivano Bonesana, Miguel Gila (CSCS)

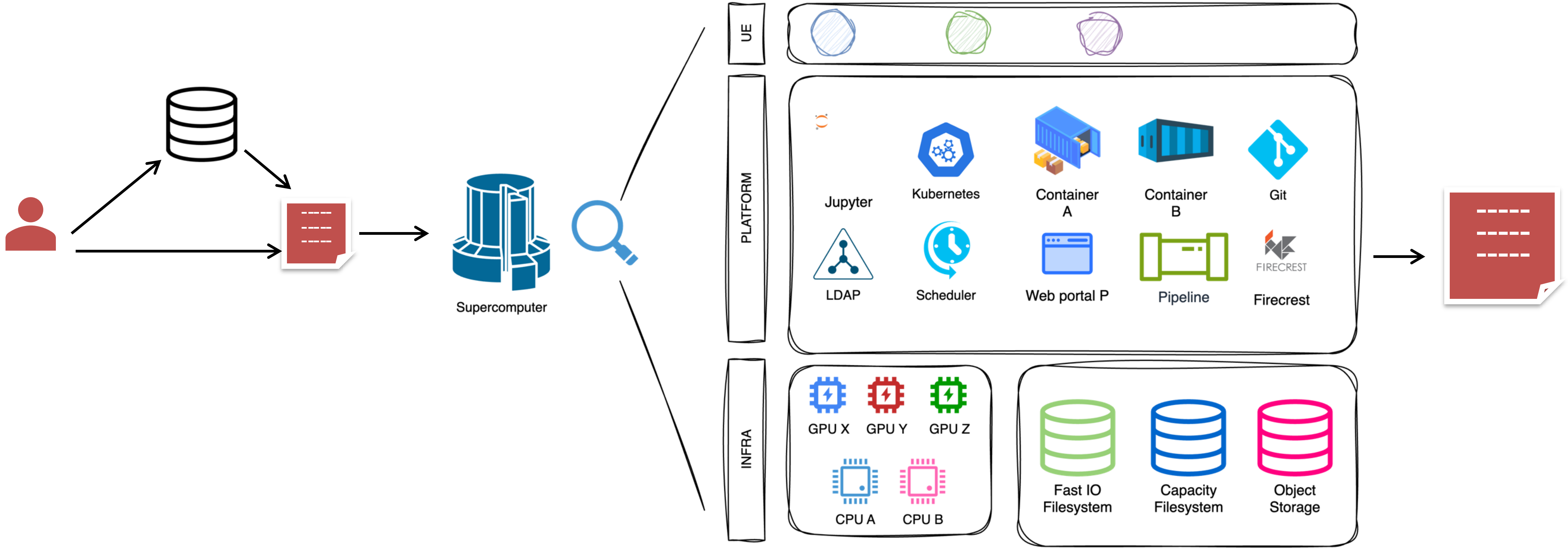
May 06, 2025

Agenda

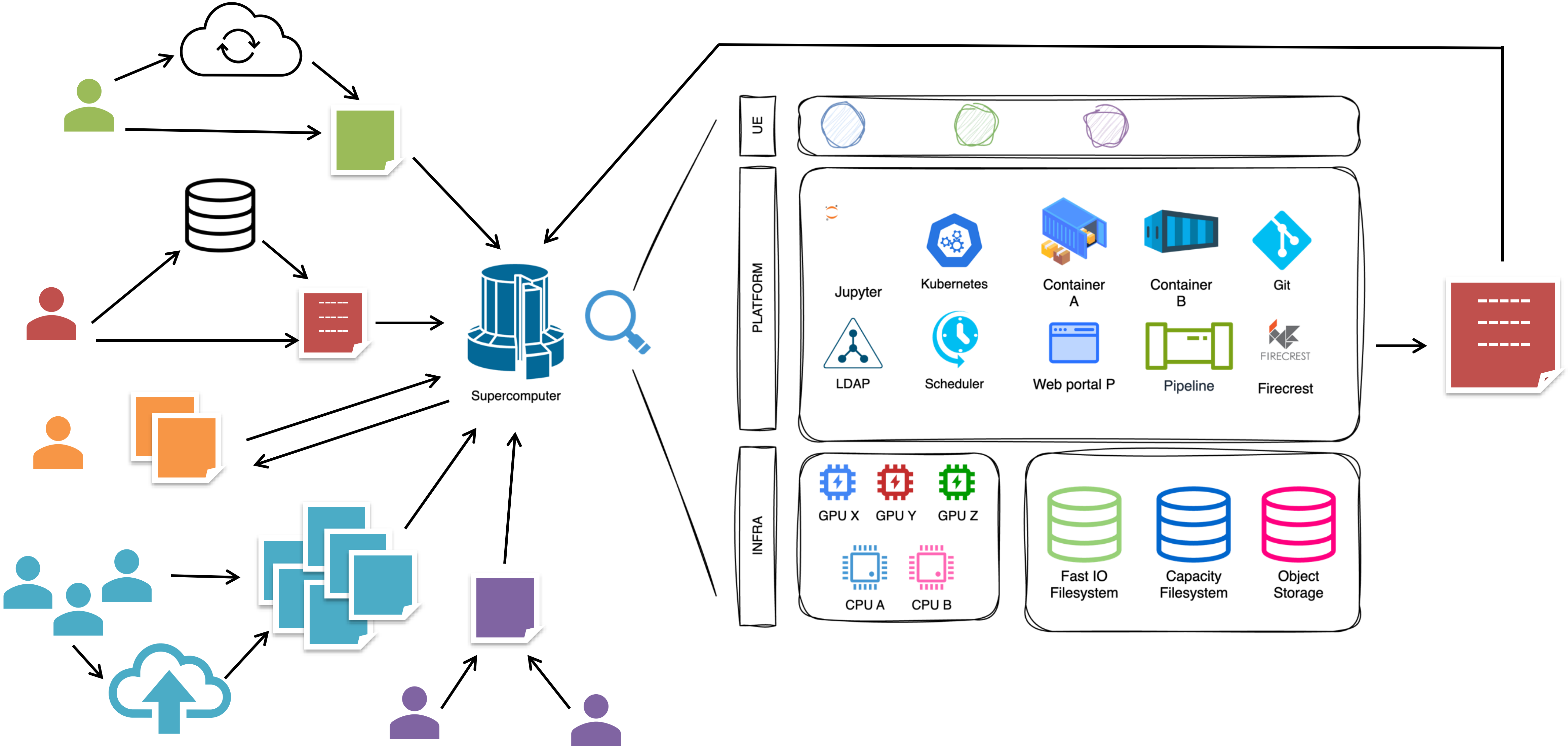
1. Introduction
2. Automation and testing: vCluster technology
3. Examples
4. Conclusions

Introduction

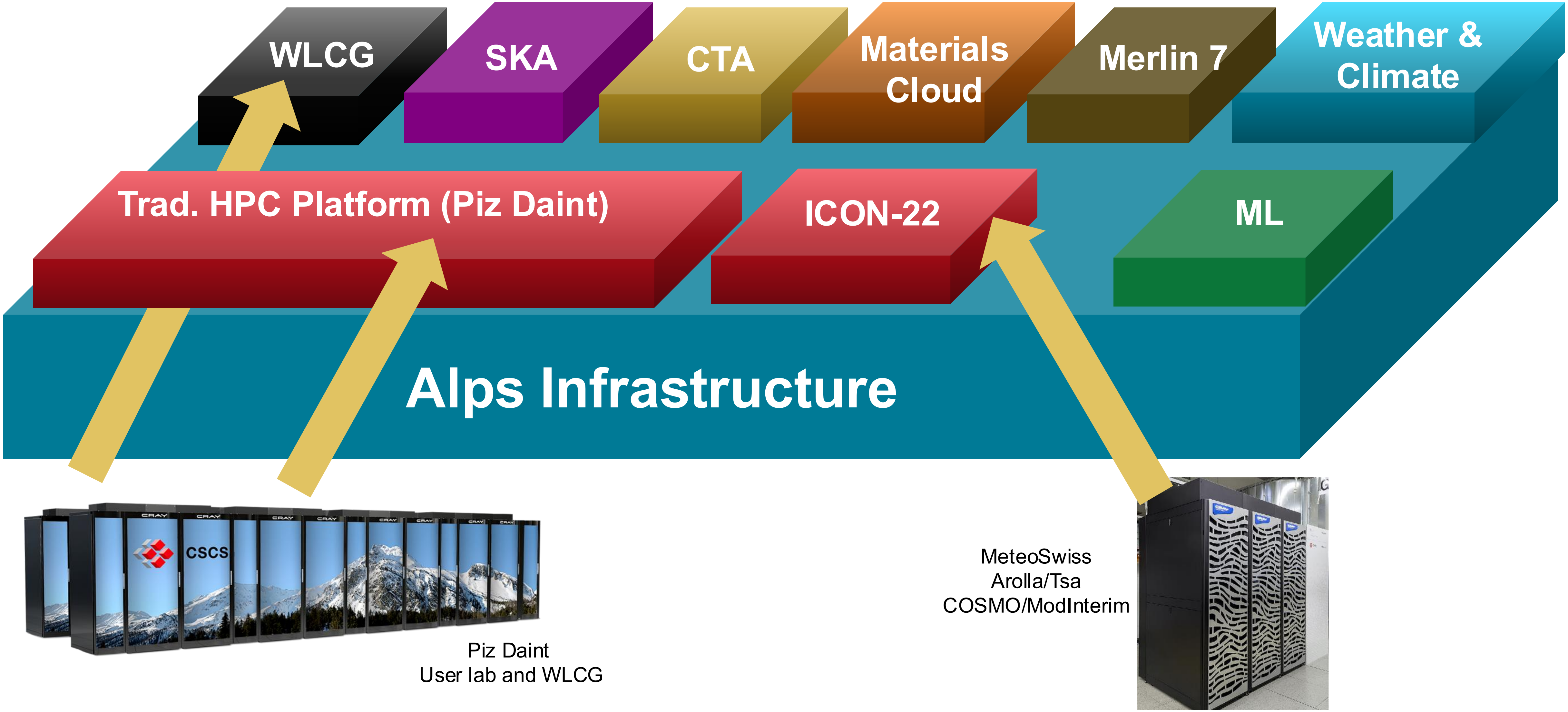
HPC Infrastructure



A Multi-tenant HPC Infrastructure



Consolidation of platforms



WLCG

SKA

CTA

Materials Cloud

Merlin 7

Weather & Climate

Trad. HPC Platform (Piz Daint)

ICON-22

ML

Alps Infrastructure

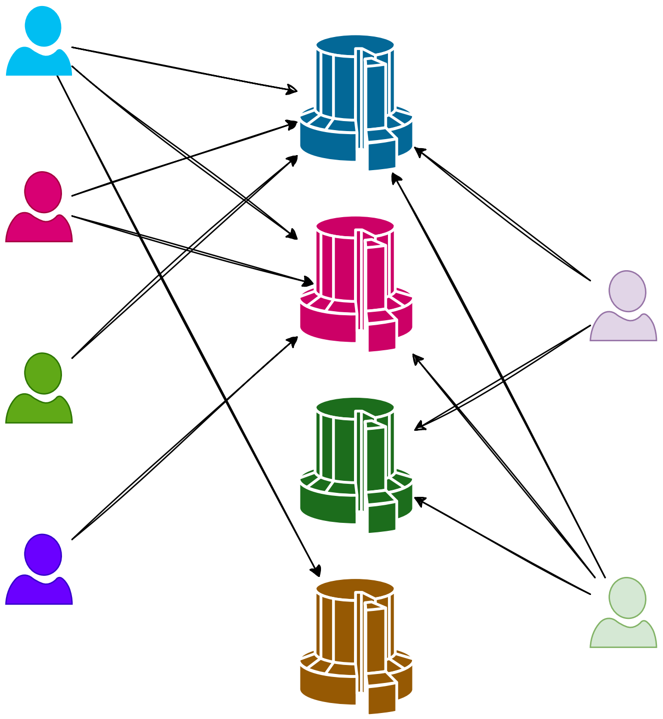
Piz Daint
User lab and WLCG

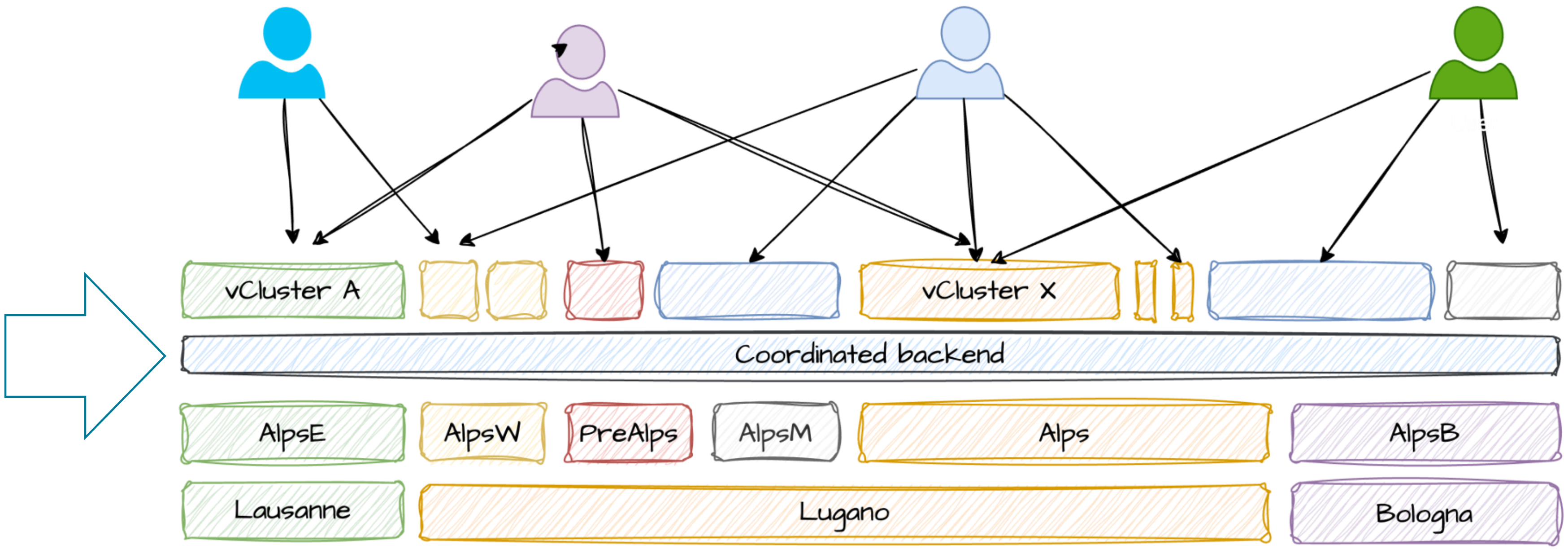
MeteoSwiss
Arolla/Tsa
COSMO/ModInterim

Multi-site, multi-tenant context

Multiple HPC systems

Distributed multi-tenant environment with multiple infrastructures





3 key aspects to manage this complexity

- Break vertically-stacked environments ✓
  - Potentially rely on manual processes
  - Only by a small set of people know how to operate them
- Automate your processes ⚙
  - Introduce the concept of software products
  - DevOps: Use (and abuse) git and pipelines
  - Rule: If you have to do it twice, write a script for it
- Test, test, and test again ⚙
  - Everything you can think of
  - And then some more

Automation and testing: vCluster technology

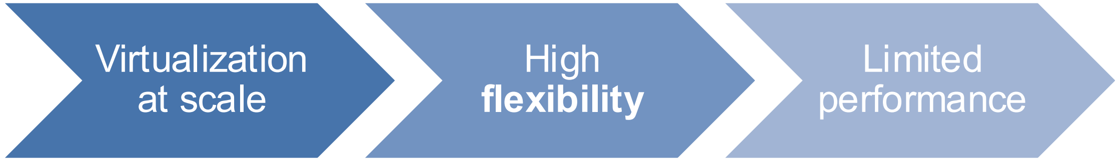
HPC and Cloud convergence

Cloud

- High flexibility for business needs
- Invested heavily in abstractions from the hardware (IaC, APIs, virtualization, etc.)
- Cloud design principle – towards enterprise
- Economy of scale – oversubscription of resources

HPC

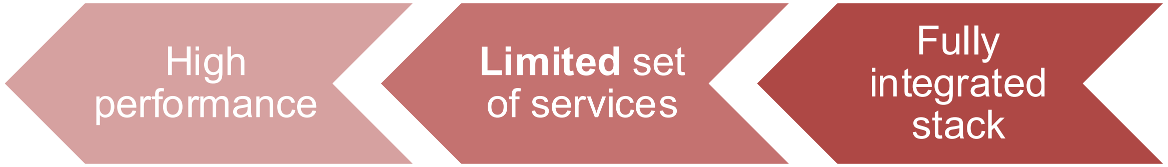
- High-performance compute and data access
- Invested heavily in vertically integrated environments
- HPC design principle – towards science
- Improves time to solution



Virtualization at scale

High **flexibility**

Limited performance



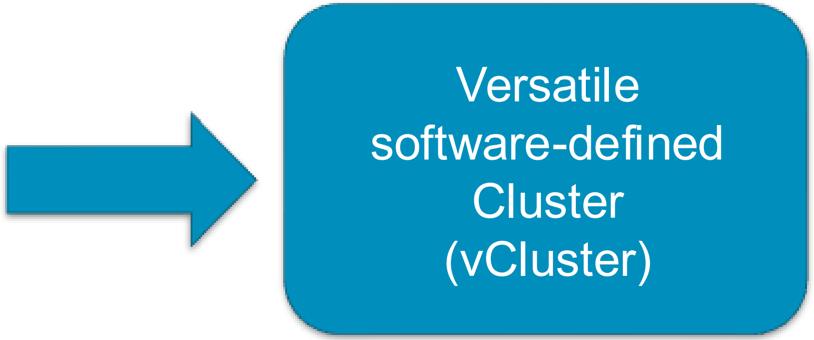
High performance

Limited set of services

Fully integrated stack

How to achieve HPC and Cloud convergence?

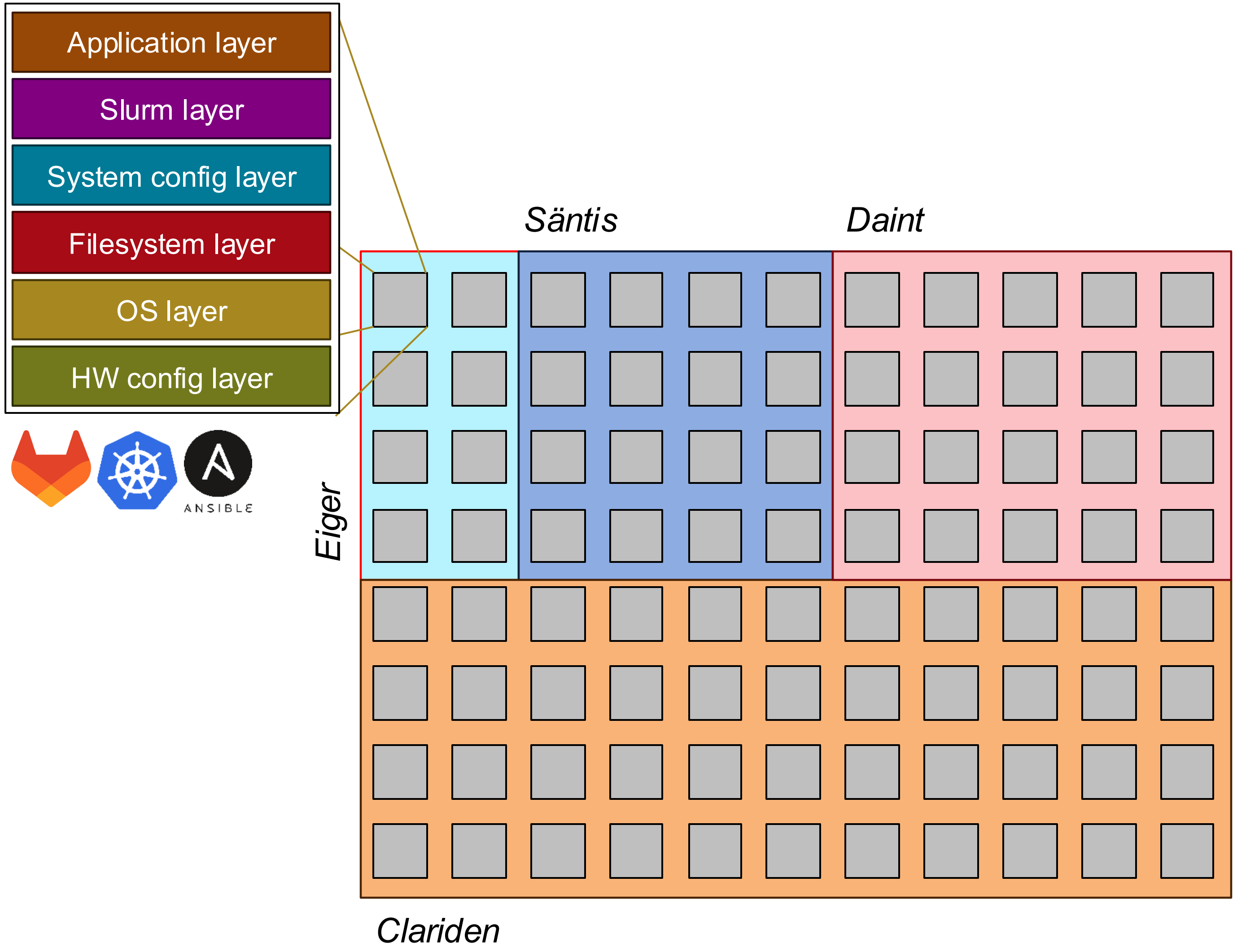
- Separation of concerns with layers
  - Platforms
    - Provisioning of services with orchestrators
    - Container as an abstraction layer for compute nodes
  - Infrastructure as a code
    - APIs and configuration management
    - Multi-tenancy: exclusive compute, network and storage segregation
- Performance and flexibility
  - Use container as an abstraction layer
    - Keep OS near bare metal
    - Bring low-level libraries in the container with OCI hooks
  - Bring your own User Environment technology
    - Decouple HPC programming environments from underlying layers
    - UE as an artifact mounted in the containers
- HPC Business logic
  - Web-facing API to access HPC resources (submit jobs, move data)
  - Web gateway



Versatile software-defined Cluster (vCluster)

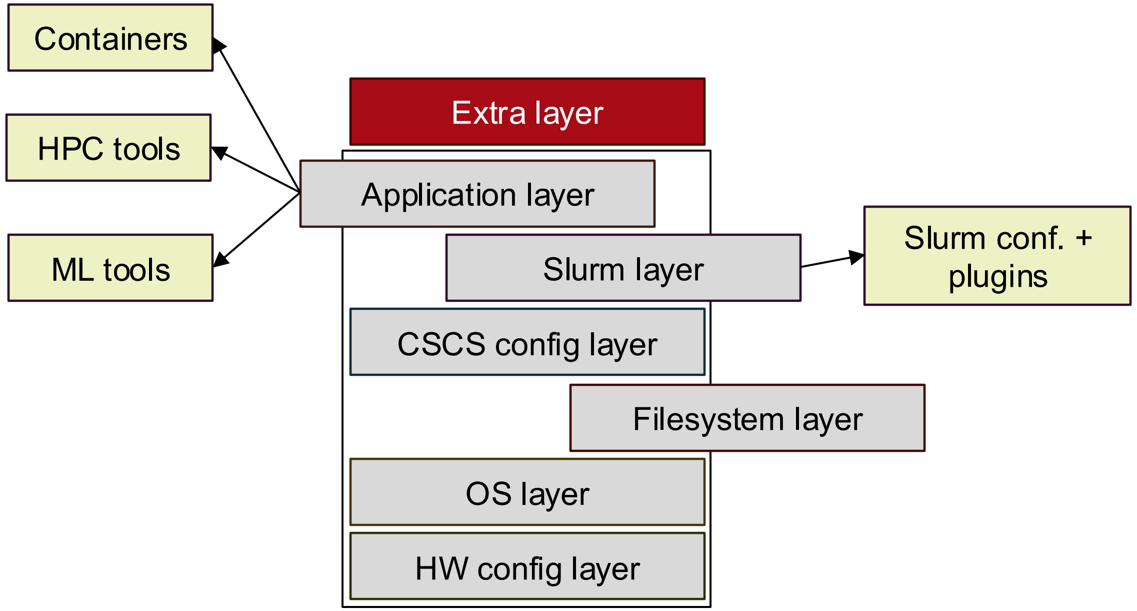
Partitioning an HPE Cray-EX into vClusters

- Image-based infrastructure
- Each node can run a different image
- Nodes not tied to specific images or configuration
- Layers are maintained in Git repositories
- Workflow advantages
  - Tests or builds on Kubernetes
  - Run any image on any node of the system
  - Keep full control on the deployed software
- Nodes can be configured with specific images matching specific requirements
- Nodes can be partitioned into vClusters



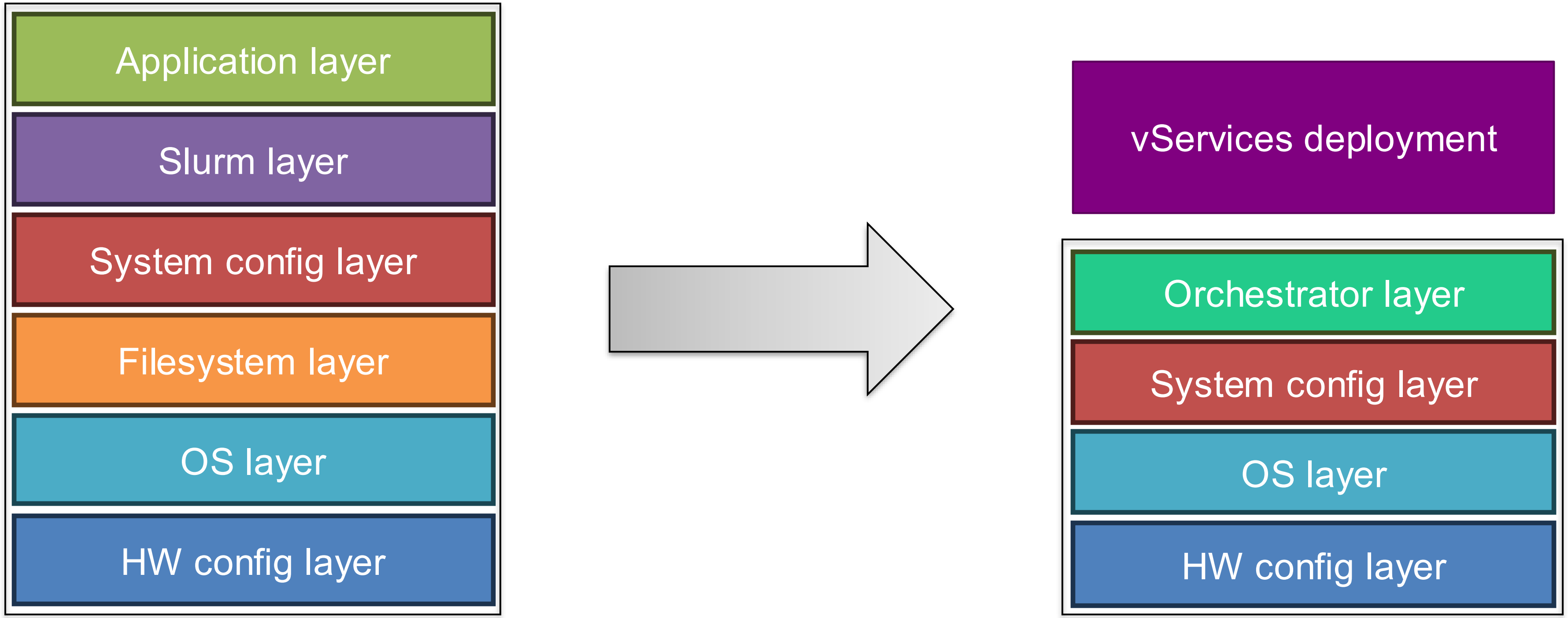
Layered images practical limits

- More flexibility is required
  - Nodes reconfiguration is time consuming
  - Requests from users to adapt and upgrade the application environment are increasing
- The image cannot do all
  - File system mounts may depend on the tenant requirements
  - Extra layers are already required to be applied after the image boots on the single nodes
- We want to more developers
  - wide support for different fields of applications (e.g., ML vs HPC)
  - Maintainers of the images cannot be expert on all the deployed applications: support from other developers is needed



Introducing vServices

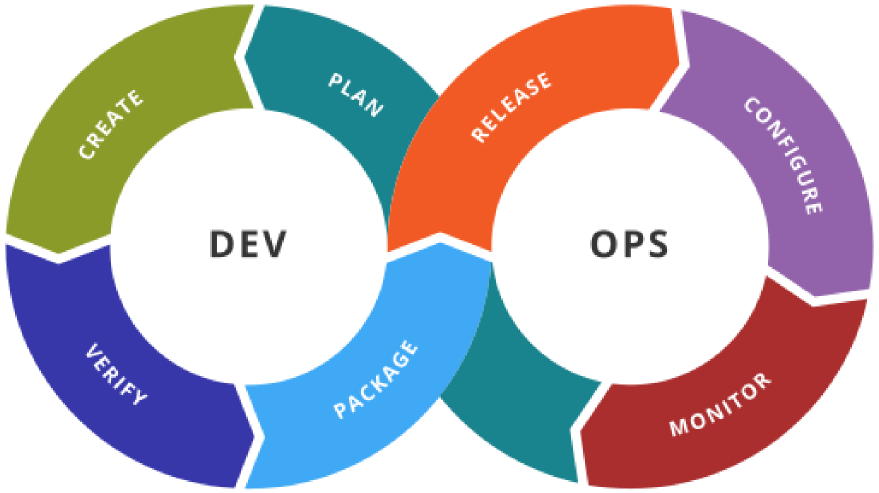
- The image implements essential layers
- User / Platform specific configuration is deployed by **vServices** running on top of it



vServic definition

- A standard software-oriented approach to deploy applications and services on nodes, running on the top of a minimal image
  - Higher flexibility to match evolving user requirements
  - Speedup updates, upgrades and bugfixes
  - Increase number of contributors that can develop and deploy software on a vCluster
  - Dynamic features: rolling updates, staging sub-vClusters and nodes "live" migration
- Terraform modules originally conceived to deploy Nomad *Jobspec* resources (supporting more providers)
- Introduce the DevOps workflow in nodes deployment process

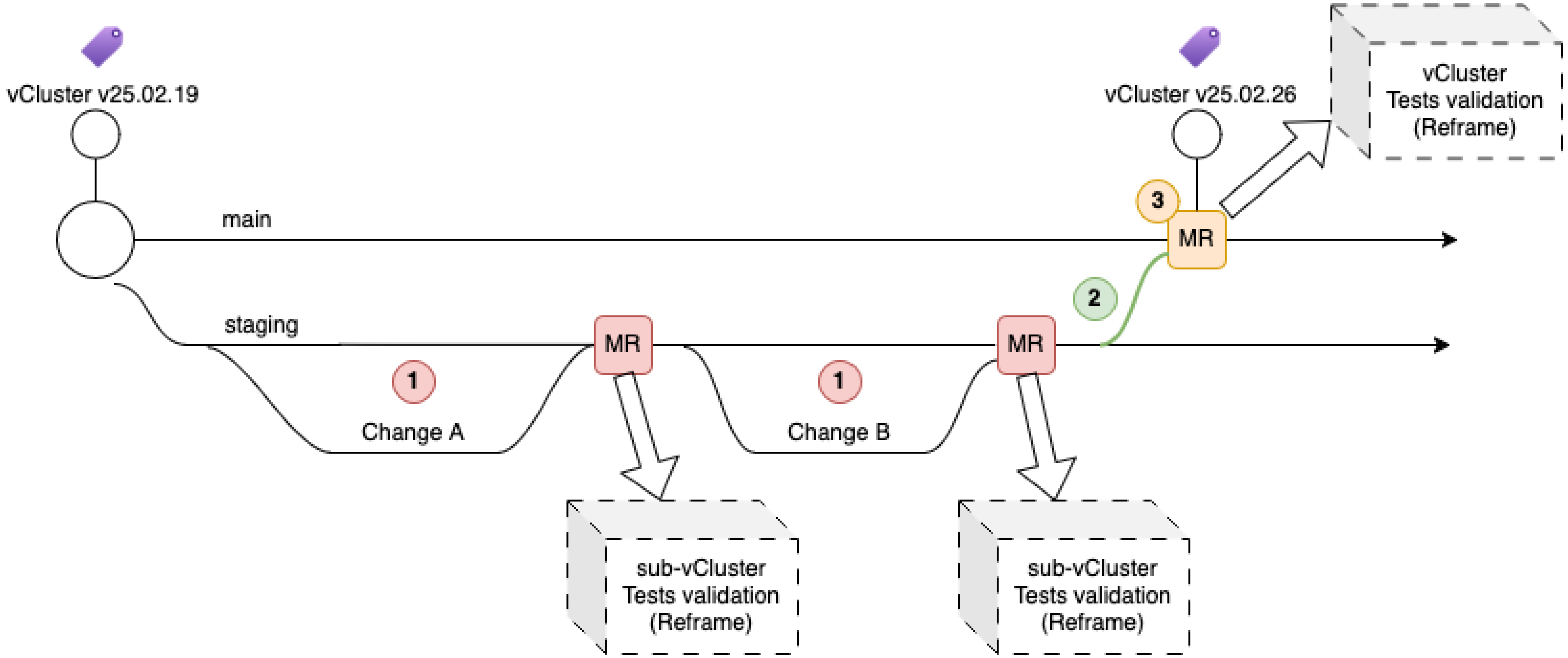
The DevOps approach in Platform Automation



- DevOps is a well know paradigm in software development
- Platform Automation can use the same concept
  - Single source of truth: the **git** repository
  - Automated deployment process for **CI/CD**
  - **Feature branches** for modification
  - **Merge requests** acceptance policy
  - Automated **tests**

The DevOps approach

- GitFlow
- Using Git and pipelines for any change
- Automatic tests for each MR
- Service changes are orchestrated in a rolling fashion
- Minimal human intervention
- Pipeline automation



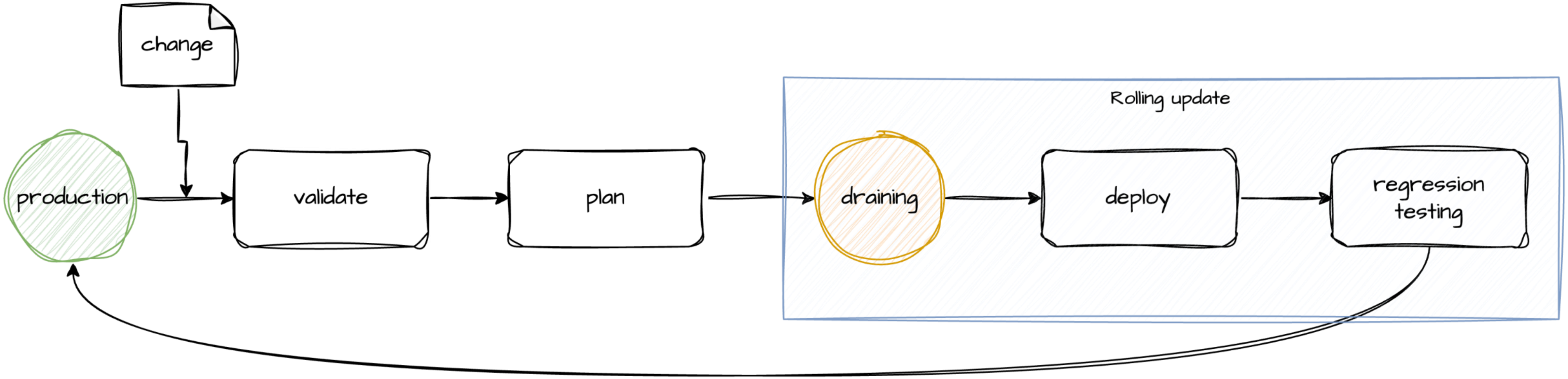
1 Changes happen during the week

2 Once a week, staging branch and all MRs introduced are discussed. Generates of a new MR to main.

3 The roster applies the MR to main during the maintenance window

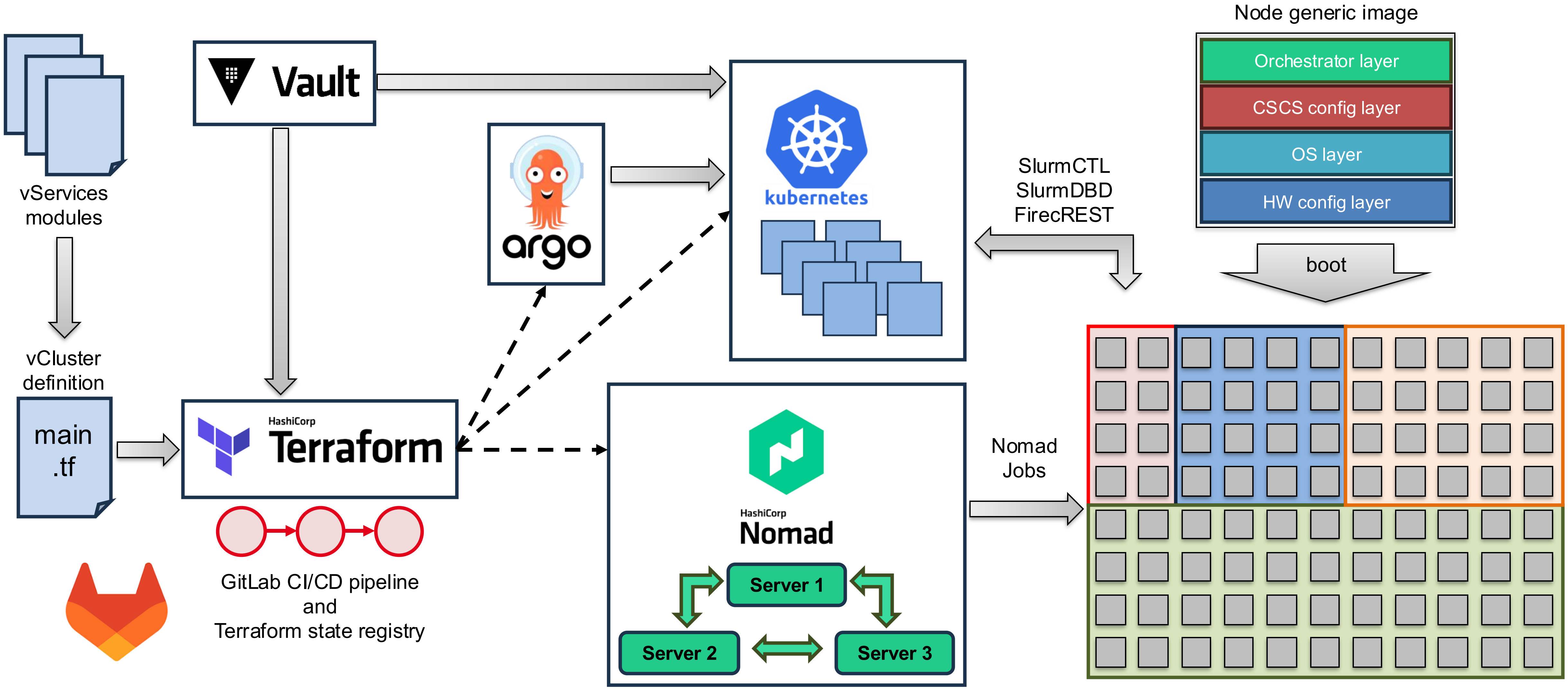
Testing MRs

- vServices and vClusters are software products that follow common development, testing and validation processes
- This enables automated integration testing and rolling updates(*) with minimal human intervention in sub-vClusters, or complete vClusters



(*) work in progress

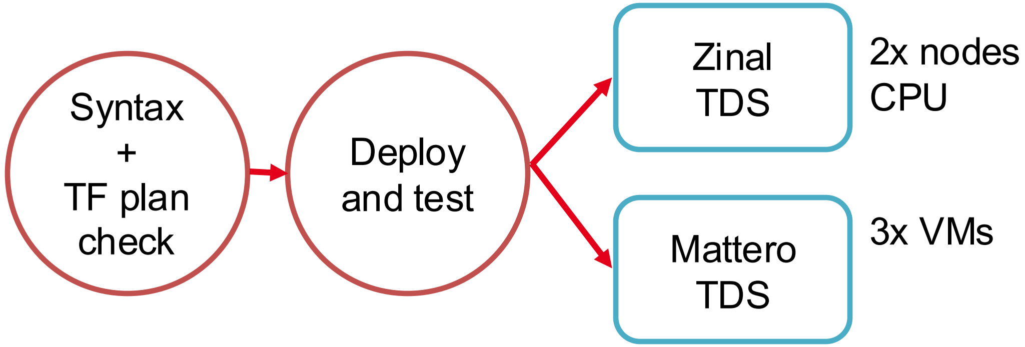
The deployment architecture



vService – Developers

- Anyone can write a vService
- The **vService development CICD pipeline** is used to test the code during the development
  - Syntax check
  - Terraform plan check
  - Deployment of vServices in a controlled TDS environment
  - Automatic tests execution and automatic cleanup
- CSCS provides TDS vClusters to deploy the vServices for integration tests
- Cloud and local VMs environments (depending on the HW requirements of the vService)

vServices development pipeline



Syntax + TF plan check

Deploy and test

Zinal TDS — 2x nodes CPU

Mattero TDS — 3x VMs

vCluster TDS integration

Wildhorn

2x login nodes
2x compute nodes
Cutting edge

Zinal

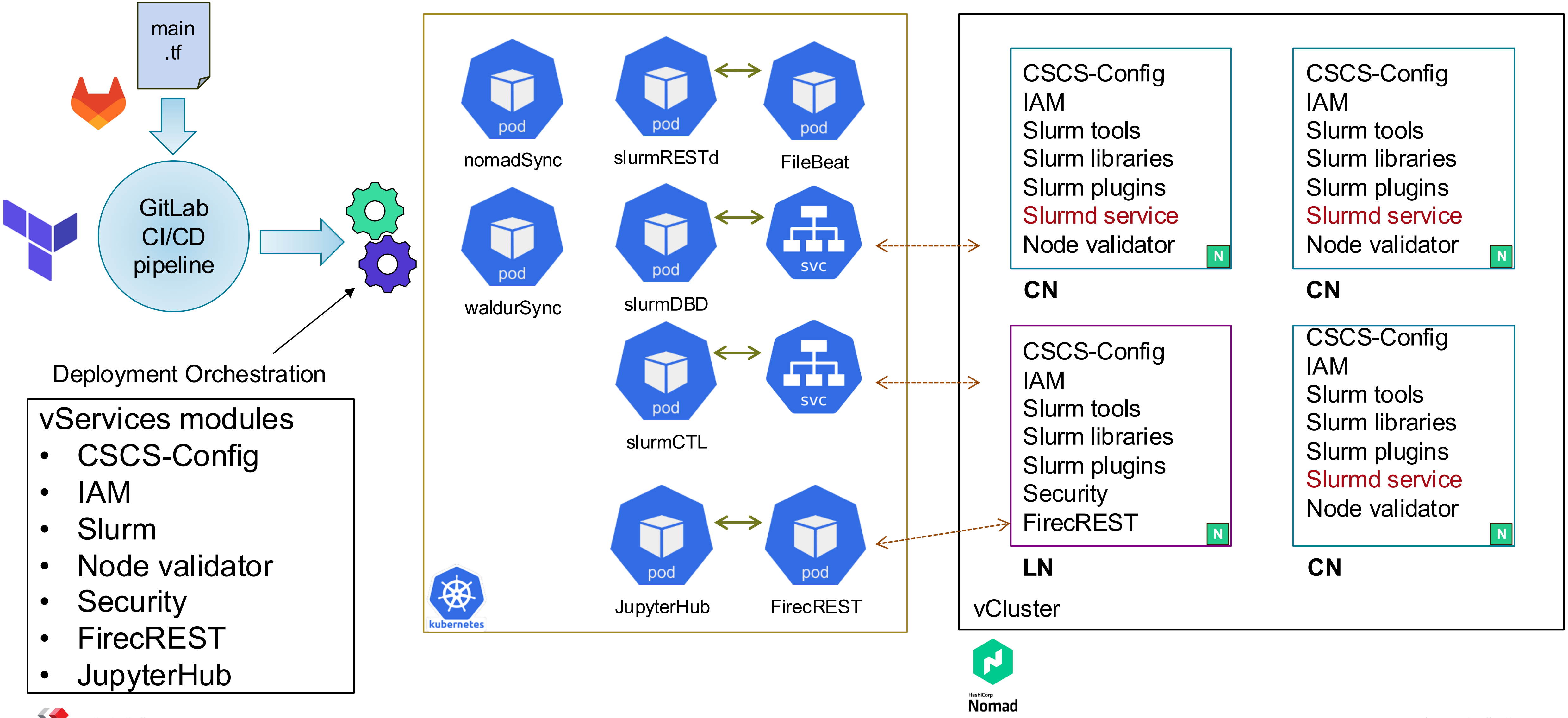
1x login node
4x compute nodes CPU
2x compute nodes GPU
Sub-vClusters
Stable deployment

Cloud & VMs

Development system and deployment are portable

Examples

Example – Deploying a Slurm vCluster



CSCS-Config
IAM
Slurm tools
Slurm libraries
Slurm plugins
Slurmd service
Node validator

CSCS-Config
IAM
Slurm tools
Slurm libraries
Slurm plugins
Slurmd service
Node validator

CSCS-Config
IAM
Slurm tools
Slurm libraries
Slurm plugins
Security
FirecREST

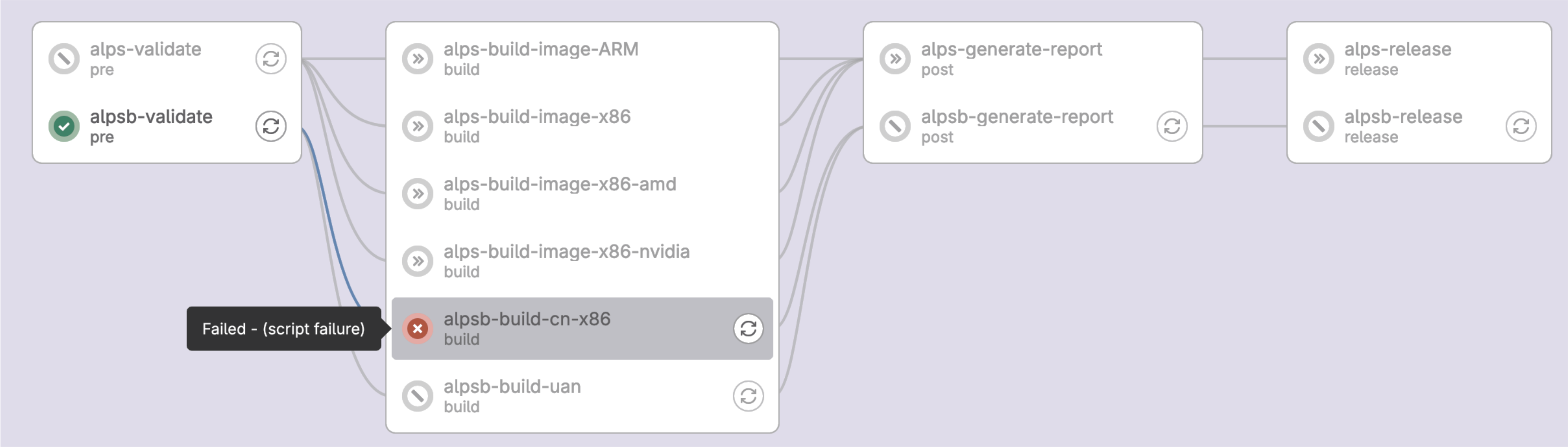
CSCS-Config
IAM
Slurm tools
Slurm libraries
Slurm plugins
Slurmd service
Node validator

vServices modules

- CSCS-Config
- IAM
- Slurm
- Node validator
- Security
- FirecREST
- JupyterHub

Example – Building a vCluster image

- vCluster images are generic: one per infra and hardware type
- Built automatically by a pipeline using our CSM/OpenCHAMI CLI **manta**
- Images will be tested and validated for security issues



Conclusions

My personal takeaway

- We have dramatically increased the visibility of Platform configuration within CSCS, and significantly reduced toil and manual work

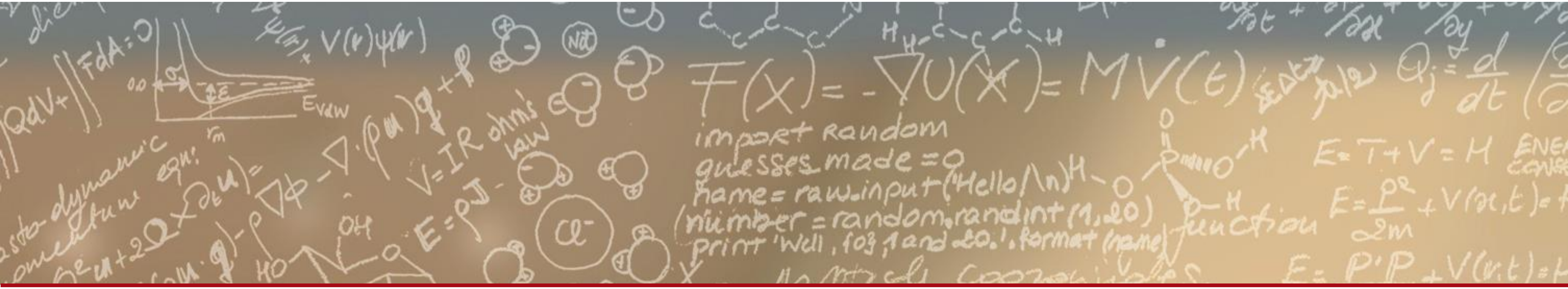
- Platform Admins and Service Managers can now track what changed when, why, and by whom. They can even do it themselves. Everything is in git

- More people is directly contributing to the different platforms
  - If you have a problem, you have two ways of solving it:
    - Open a ticket and wait
    - Open a MR and get it ready by next week's maintenance cycle

Where are we?

- The path to the complete automation is still long, curvy and steep
- The Numbers
  - 16 vClusters in operation + 8 in 2025 Q2
  - over 20 active vServices
  - 900 MRs in the last two years
- Thanks to our colleagues at EPFL, vClusters can now live also on the cloud
- Improved serviceability due to the added consistency, efficiency and versioning that this brings
- None of this is exclusive of our multi-tenant environment. Anyone will be able to contribute, and use it
- Future work will focus on integrating these efforts into an automation framework

Maloja Pass, source: Wikimedia



Thank you for your attention.