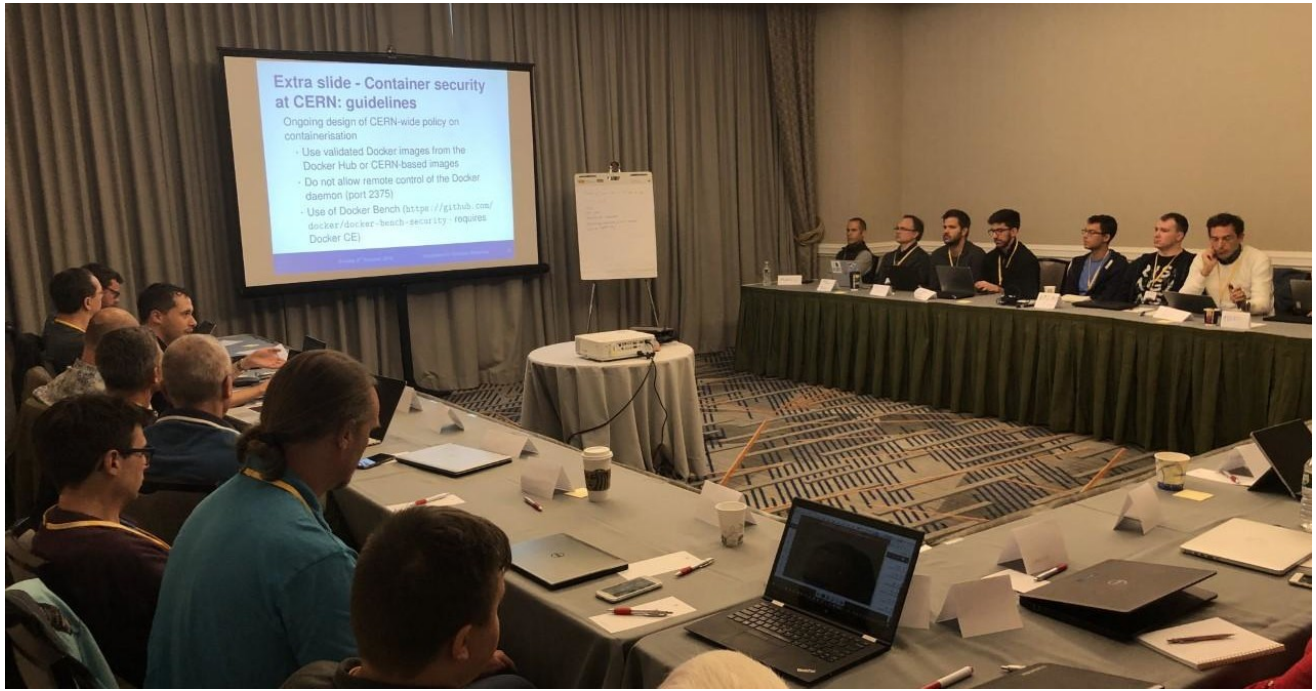
The State of Containerization in CERN Accelerator Controls

Rémi Voirin

21/10/2021 – ICALEPCS 2021

Containers in Controls workshop in Brooklyn



ICALEPCS 2019

1 day

30 attendees

6 presentations

2 group work sessions

3 hands-on exercices

Outline

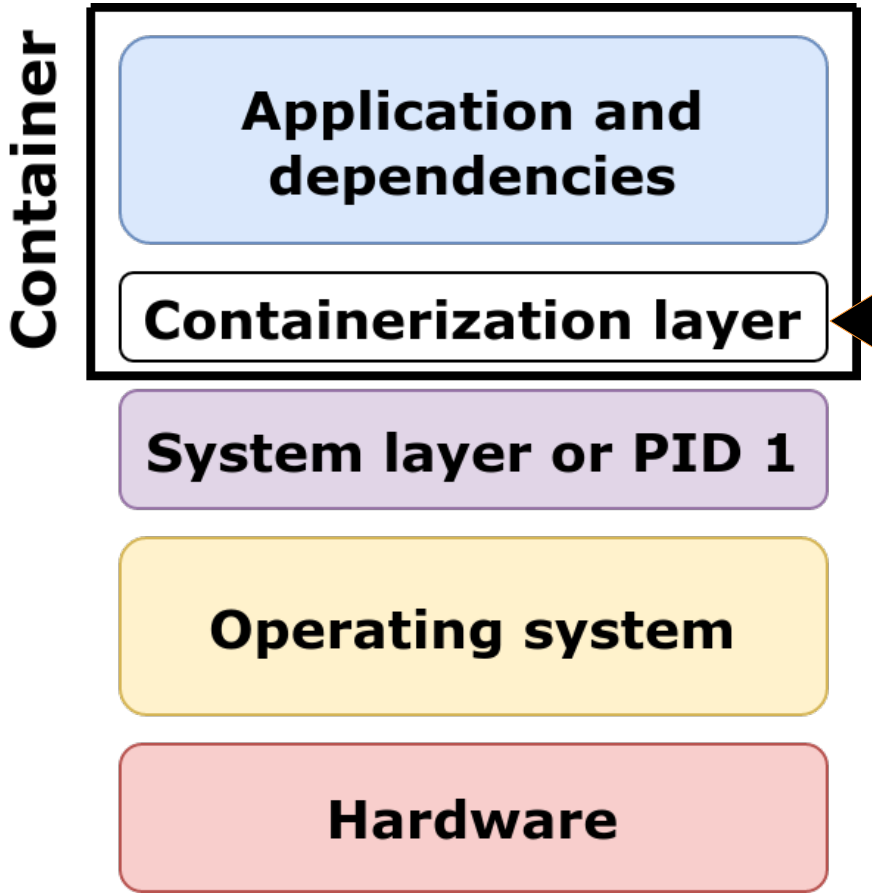
Technical background

Container availability

Future plans

Table of Contents

Simplified view of a container



Isolation with namespaces:

- Filesystem
- Process ID
- Network
- ...

OPEN CONTAINER INITIATIVE

Industry use cases

Containers are becoming the norm in the industry for:

- Idempotent execution between development and operational environments
- Streamlining the use of DevOps tools
- Managing application dependencies
- Encapsulating legacy solutions

Table of Contents

Container project: overview

Purpose: offer the ability to run containers on servers and technical consoles

What we need:

- **Base images** that our developers can rely on
- **An image registry** to store container images
- **A container engine** to run containers on hosts

Container project: base images

```
dnf -y --installroot=${WORKDIR}/rootfs
--nodocs install ${PACKAGES}
```

Install base CentOS packages in subdirectory

Apply custom configuration files

Create an archive of the work directory

```
tar -C ${WORKDIR}/rootfs
-cf ${WORKDIR}/rootfs.tar .
```

Convert to OCI image

```
podman import -
acc_cc7:latest < rootfs.tar
```

Container project: base images

acc-co › docker › acc-base-images › **Pipelines**

All 57 | Finished | Branches | Tags | Run Pipeline | Clear Runner Caches | CI Lint

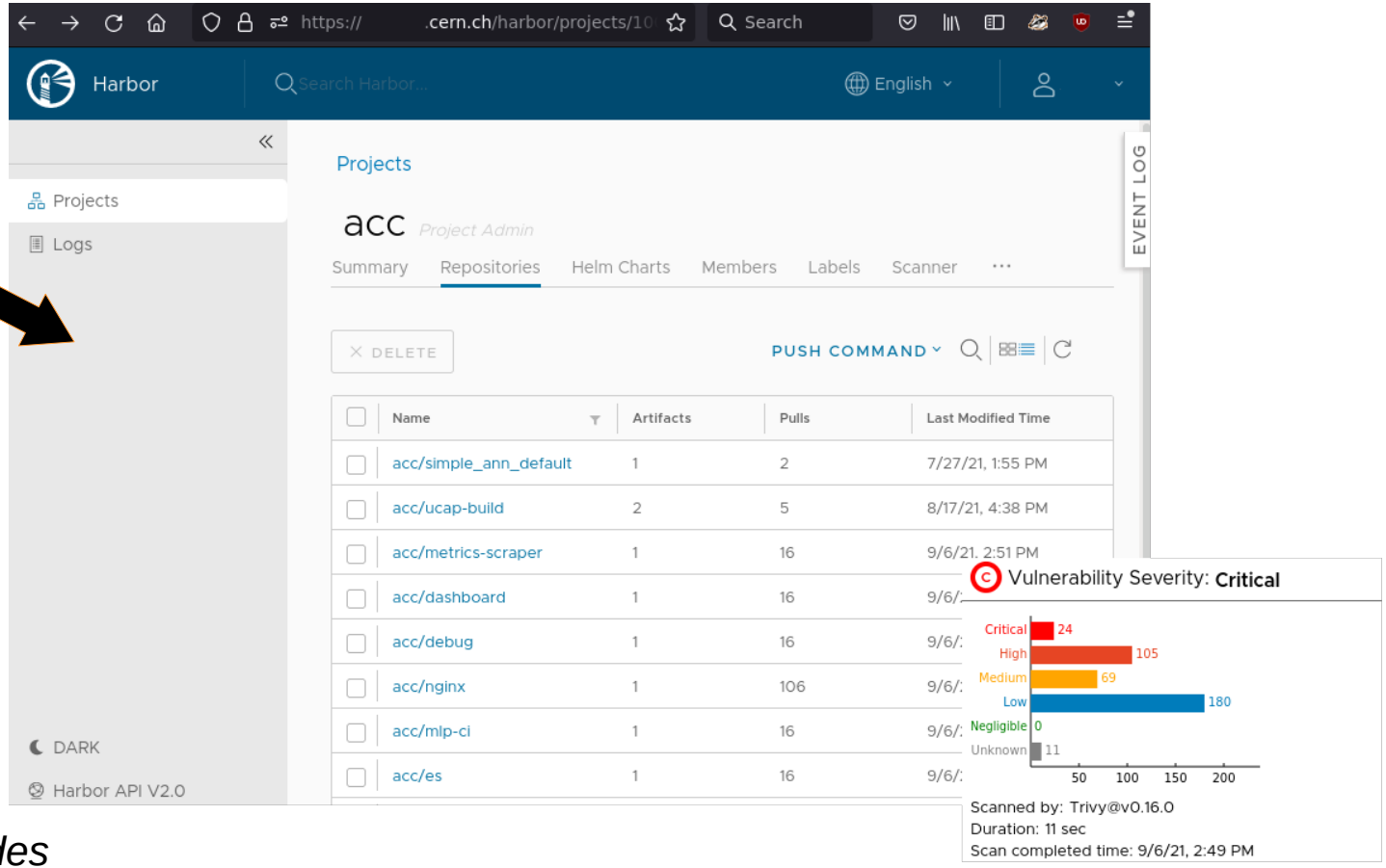
Filter pipelines

Status	Pipeline	Triggerer	Commit	Stages	
passed	#2098034 Scheduled latest		master 284d619f .gitlab-ci.yml: Add prune		00:21:24 2 hours ago
failed	#2095193 Scheduled latest		master 284d619f .gitlab-ci.yml: Add prune		00:17:49 1 day ago

Container project: registry

Approval mechanism*

List of allowed images on the Controls registry



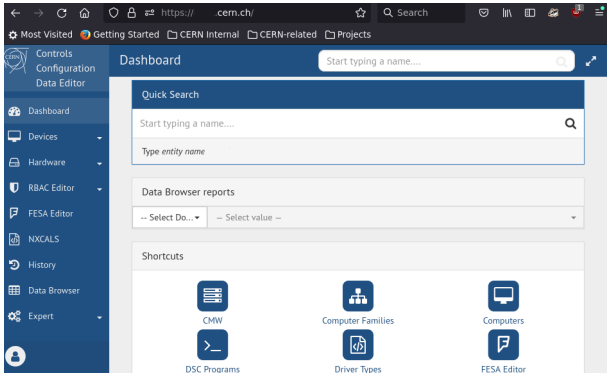
** Details in extra slides*

Container project: engine

podman

Podman is our container engine of choice:

- Containers run as plain systemd services



1. Declare in the Controls Configuration Database

Controls server

2. Deploy and run

Container project: engine

podman

Podman is our container engine of choice:

- Containers run as plain systemd services
- Rootless for security and practicality
- Daemonless architecture
- Community-driven project

Current CERN use cases

- Standard way to deploy software (e.g. SourceGraph, Nexus)
- Decoupling software upgrades from operating system upgrades (e.g. WinCC OA 3.16 on EL8 consoles)
- Unified deployment and operational environments (e.g. LHC Injector Chain Timing Sequence Manager)
- Replicate production in local environments (e.g. LHC Orbit Feedback, Controls Middleware Directory Service)

Table of Contents

Making sense of containers on a larger scale

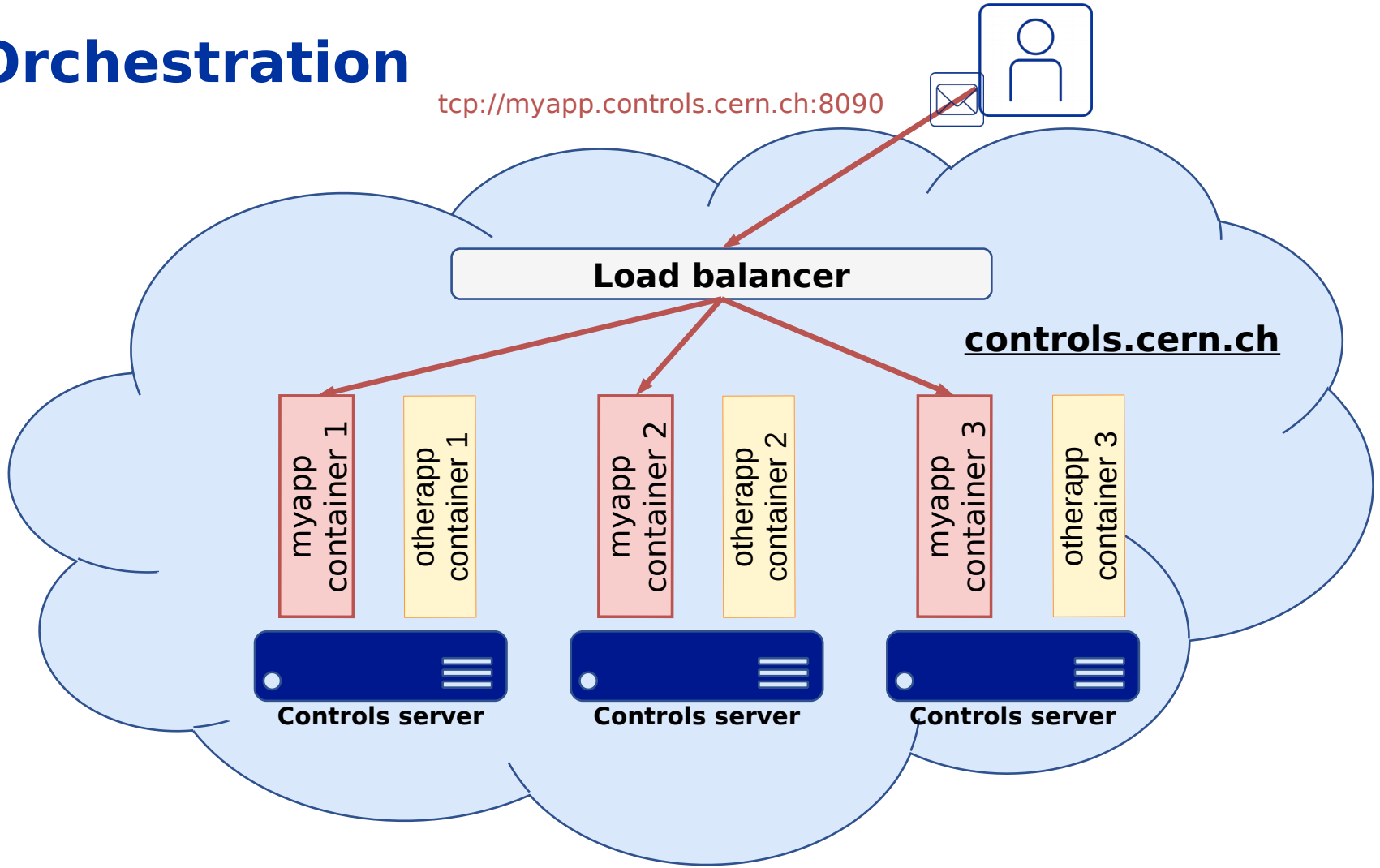
Quick delivery of components was necessary to:

- Address demands for specific use cases
- Channel its use / provide tooling around this emerging technology
- Enforce security rules for container deployment

Questions arising are now:

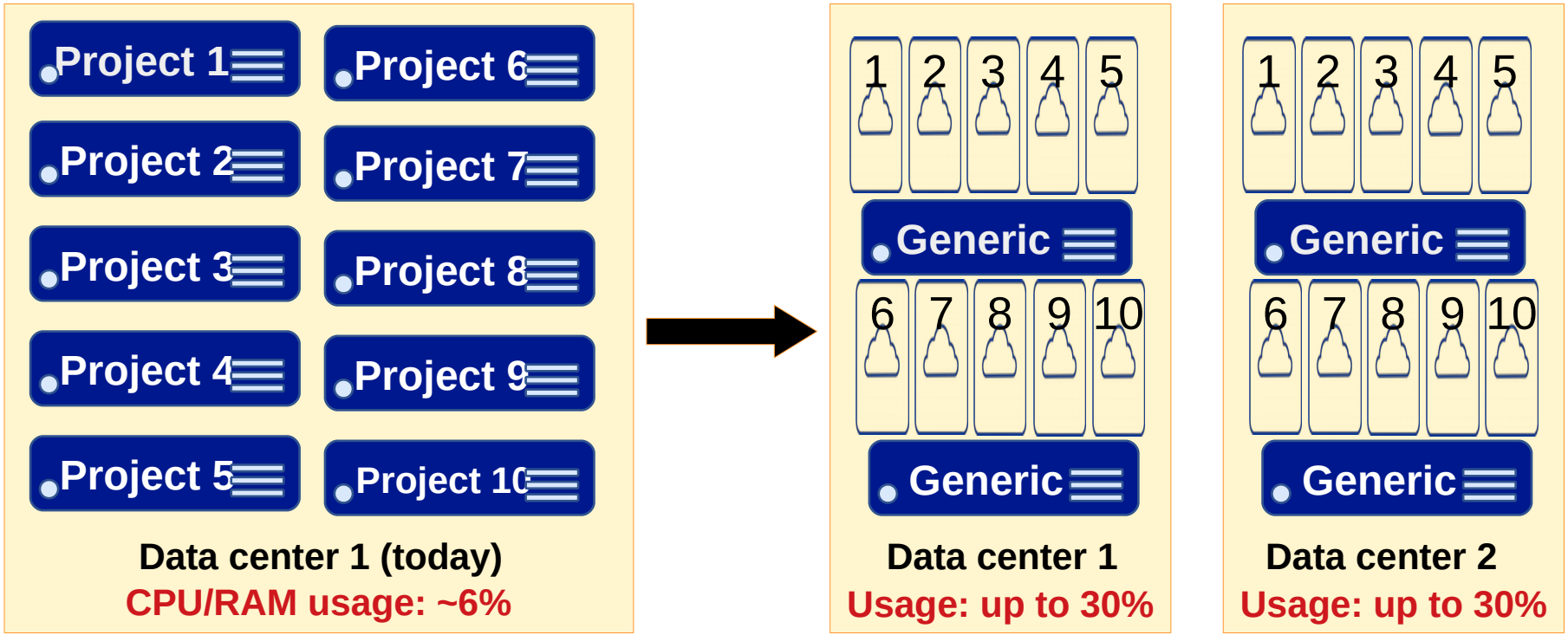
- Should we push for global use of containers? What is the added value for existing projects?
- Does the use of containerization pave the way to container orchestration?

Orchestration



Infrastructure challenges addressed by orchestration

Mitigation of data center failure + improved usage of the bare metal infrastructure

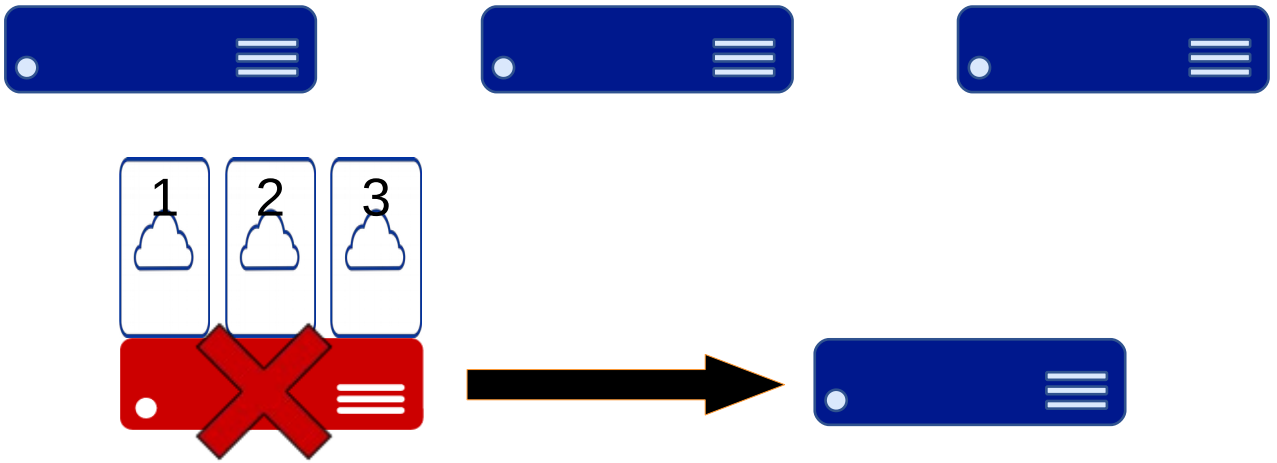


Infrastructure challenges addressed by orchestration

Generic, orchestrated servers

=

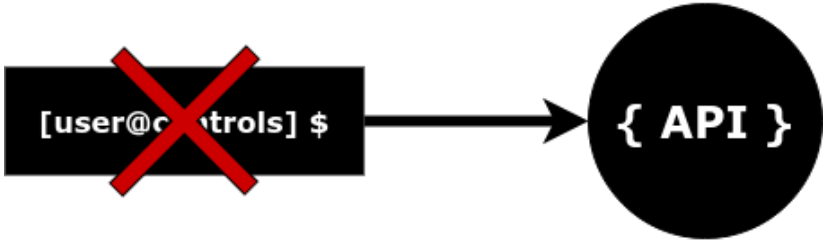
easier lifecycle management and shorter maintenance windows



Infrastructure challenges addressed by orchestration

Moving away from the “Linux shell” interface

- Give access to APIs instead
- More security by design
- Easier for developers
- Some use cases are ready
  - MOBL03 Machine Learning Platform: Deploying and Managing Models in the CERN Control System
  - TUBL01 Distributed Caching at Cloud Scale With Apache Ignite for the C2MON Framework



Is container orchestration a silver bullet?

Concerns and questions:

- Orchestration vs virtualization for monolithic applications
- Added complexity
  - More risks, more human resources
  - For this reason, self-managed Kubernetes was dismissed in 2019
- Orchestrator maintenance

Our next step: try Nomad

To conclude…

Containers offer advantages in Controls environments

- Quick and easy way to provide software, similar development and operational environments, management of software dependencies, …

Plain containerization is available

- Three bricks: base images, registry, container engine

Should the future be containerized or even orchestrated?

- Would “standard” containerization on a larger scale make sense?
- Is orchestration the right technical solution for providing more flexibility to Controls infrastructures?

Please share your experience!

Thanks for your attention!