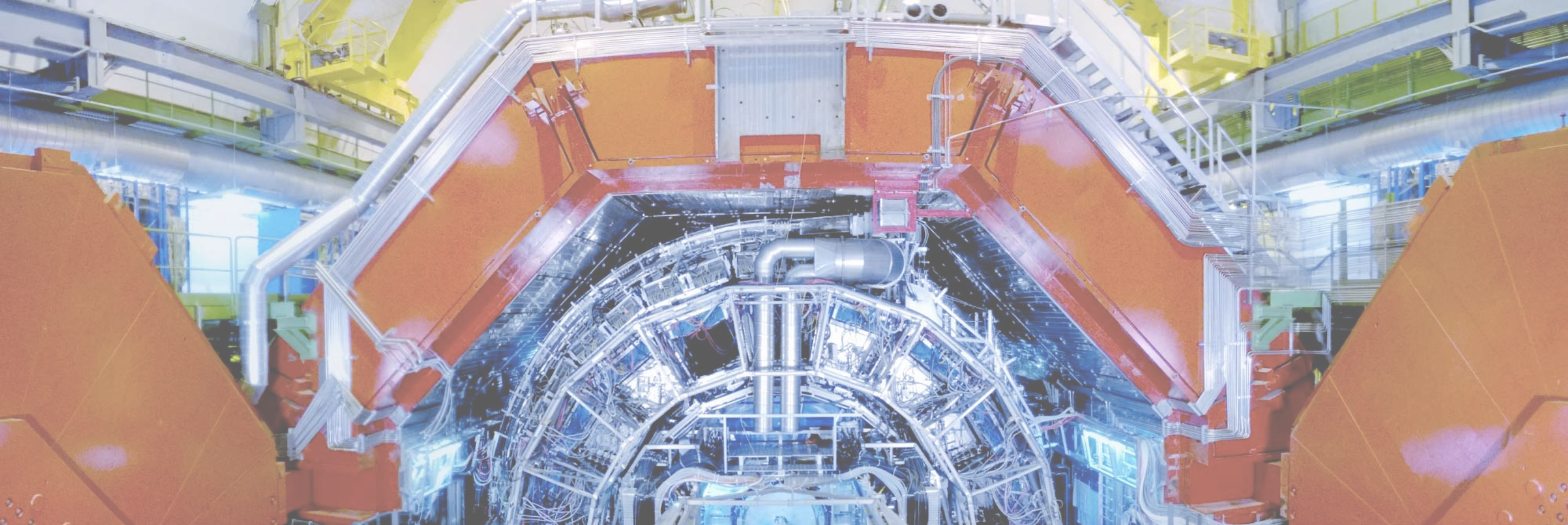


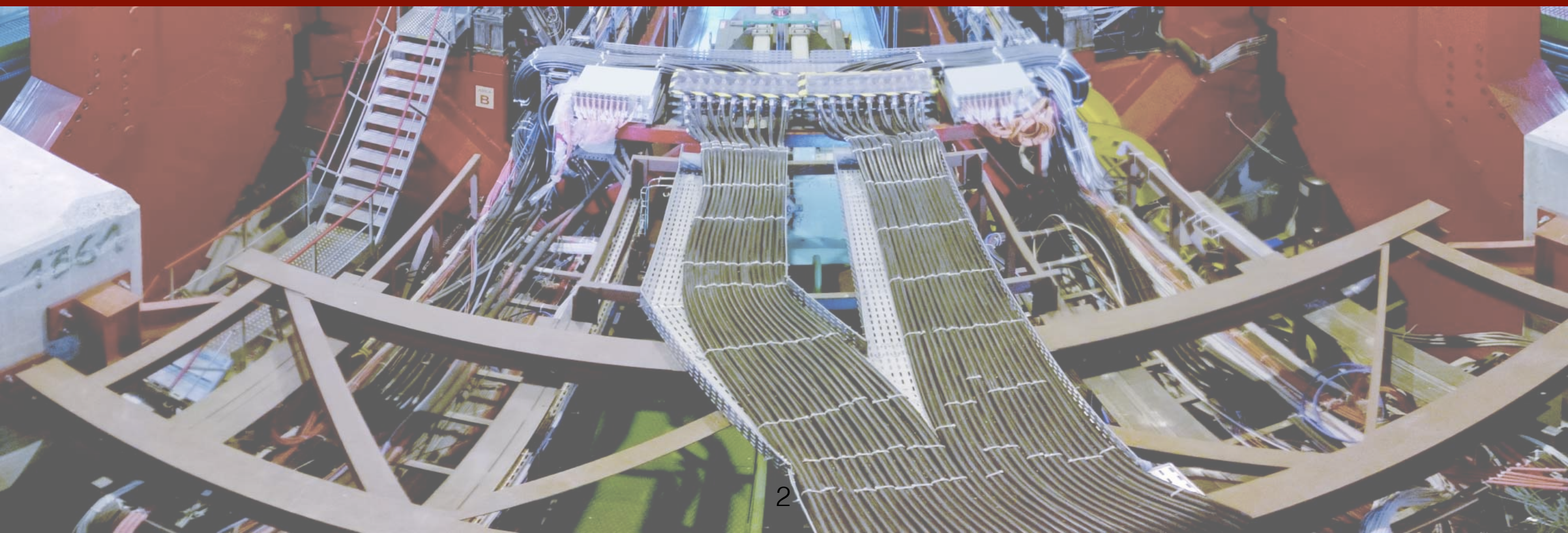
# Lightweight scheduling of elastic analysis containers in a competitive cloud environment: a Docked Analysis Facility for ALICE

**Dario Berzano**  
ALICE Offline - CERN





# Background





# The virtual analysis facility



- A self-contained and self-scalable batch cluster of CernVM VMs
- Resizes on demand with elastiq: [github.com/dberzano/elastiq](https://github.com/dberzano/elastiq)
- CHEP 2013: [indico.cern.ch/event/214784/session/4/contribution/308](https://indico.cern.ch/event/214784/session/4/contribution/308)

# Current applications of the virtual analysis facility



- Opportunistic cloud on top of the High Level Trigger

When unused, Grid worker nodes virtual machines are deployed automatically on selected HLT nodes: 7000 opportunistic slots
- On demand Release Validation cluster (see contrib #460)

HTCondor cluster on CERN OpenStack to run AliRoot validation jobs and certify it for a specific CernVM snapshot
- PROOF-based analysis clusters

Run PROOF via PoD on an elastic virtual cluster: virtualization is invisible to the end user



# Issues of cloud deployments

- Why do we use VMs?

Isolation + consistency + elasticity

- **Elasticity issue #1:** elastic applications vs. inelastic walls

Competitive clouds: cannot scale promptly with little resources

- **Elasticity issue #2:** preemption and rolling updates

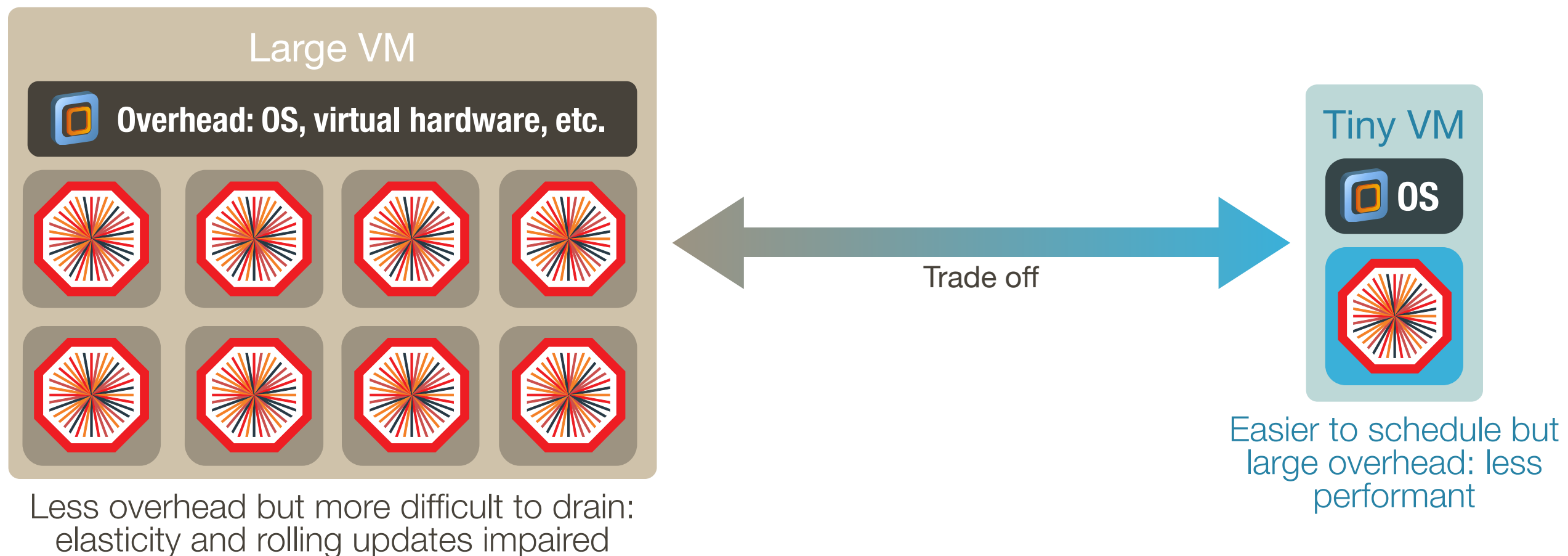
Drain VMs before delete: can be clumsy and wastes resources

- **Bottom line:** no large scale true\* cloud deployment exists

\*We don't submit VMs: they are mere wrappers for our batch jobs



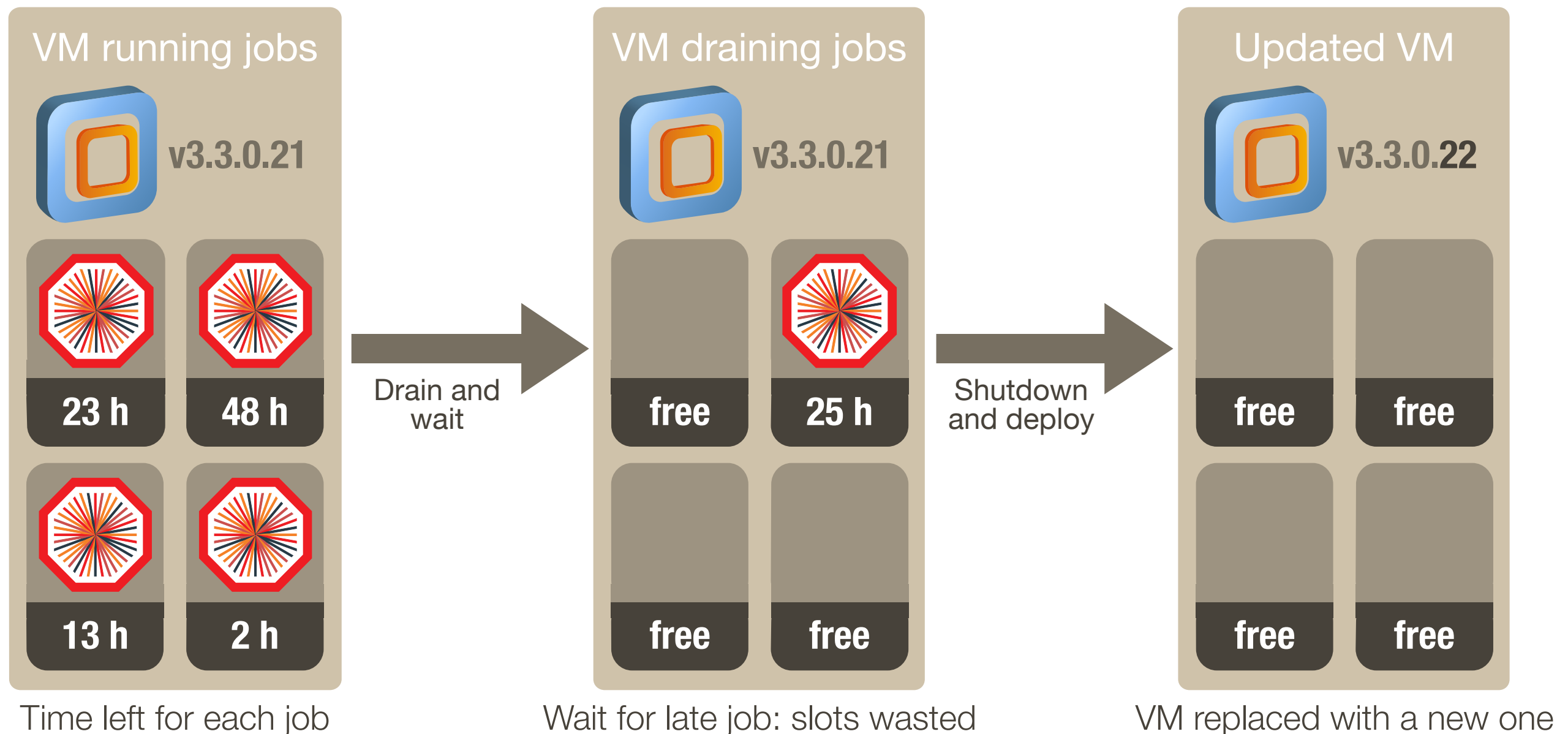
# How many jobs per virtual machine?



- Trade off: *efficiency vs. elasticity*
- Rolling updates inevitably slower and waste resources
- Claiming resources is slower too (unless we kill instead of draining)



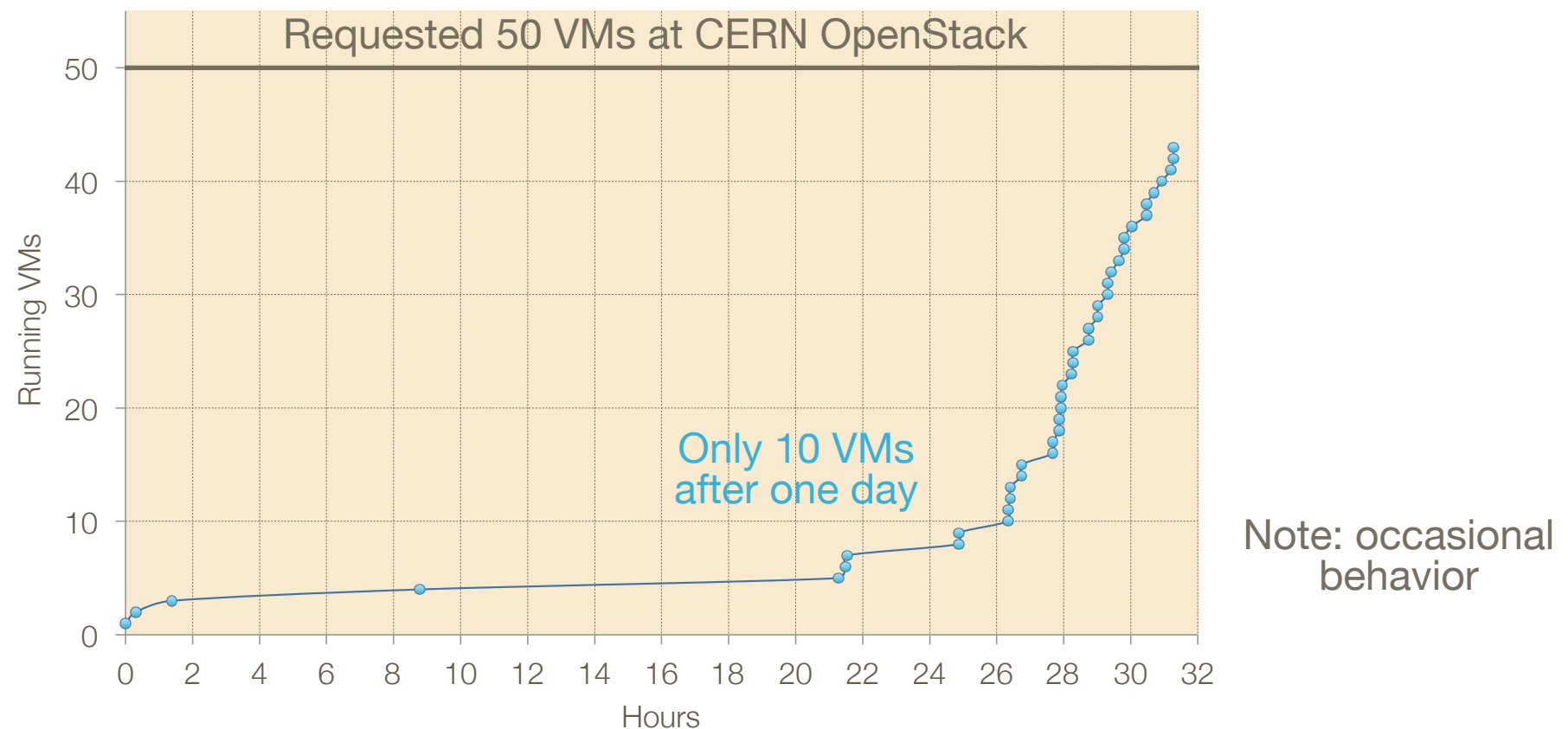
# Elasticity and rolling updates with VMs



- Replace running VMs with updated or different ones
- Drain and wait for late jobs: resources wasted for a long time
- Backfilling requires convoluted interaction between batch and cloud



# Elastic applications vs. inelastic walls



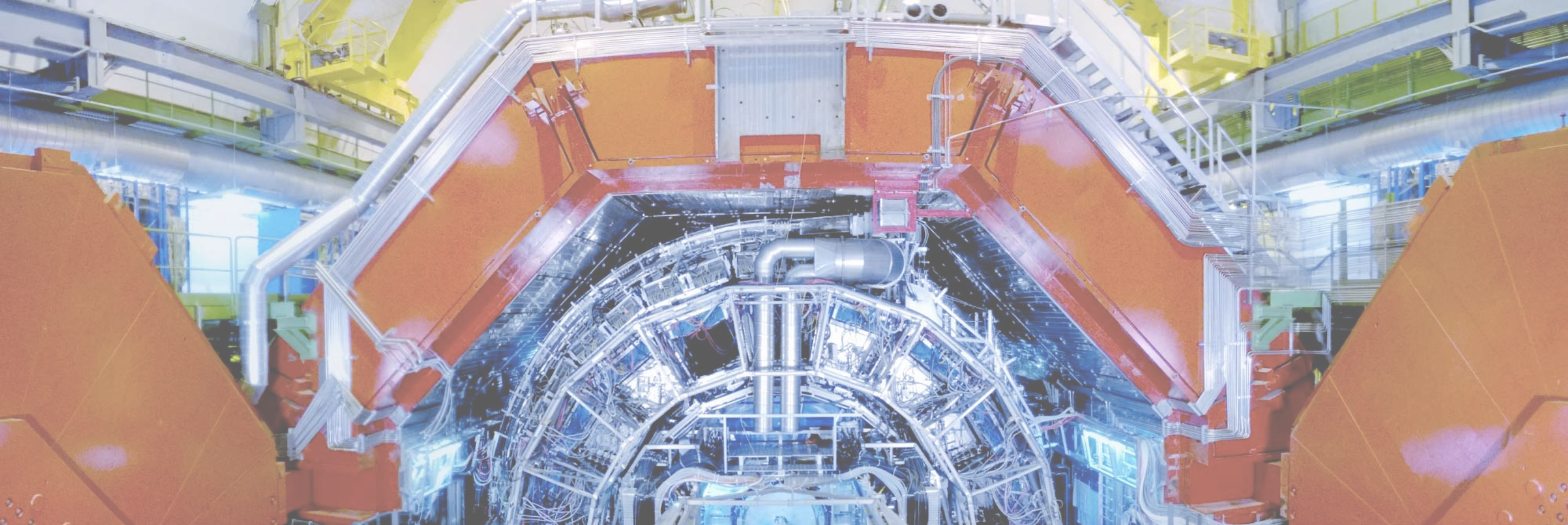
- ALICE is a happy user of VMs on CERN OpenStack

But quota is not guaranteed: sometimes not enough resources

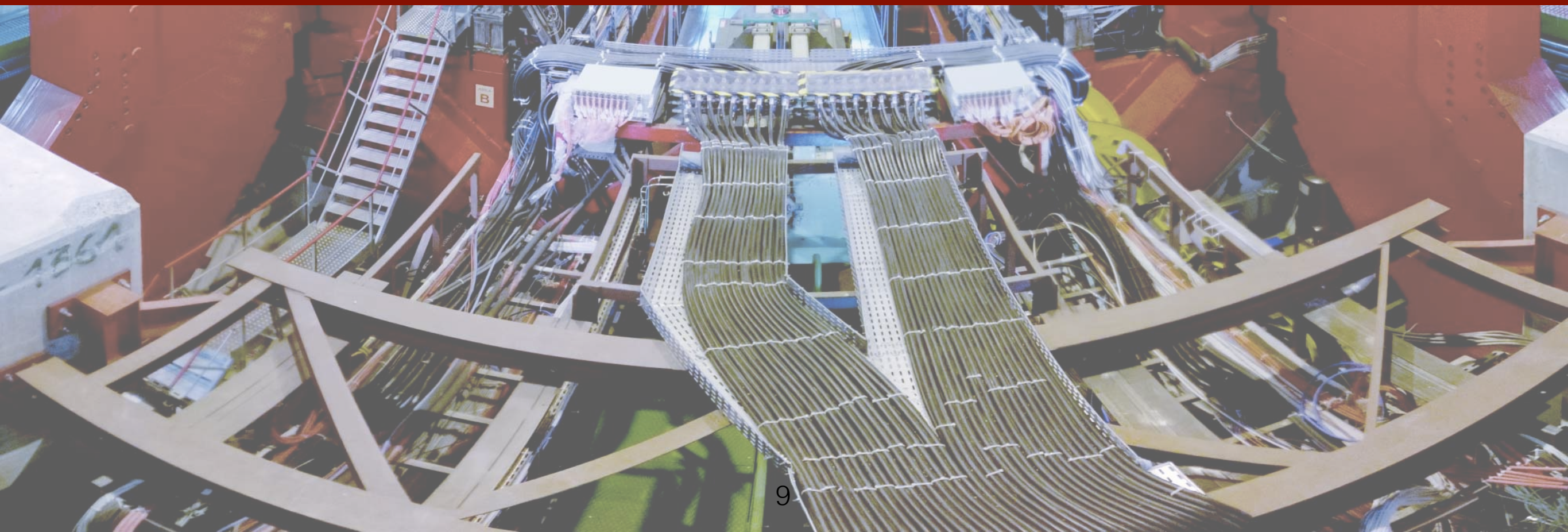
- Mostly usable for static VMs and clusters

Elasticity works in conjunction with an accounting & billing: if users have to pay per use, they want to turn off unused VMs (Amazon)



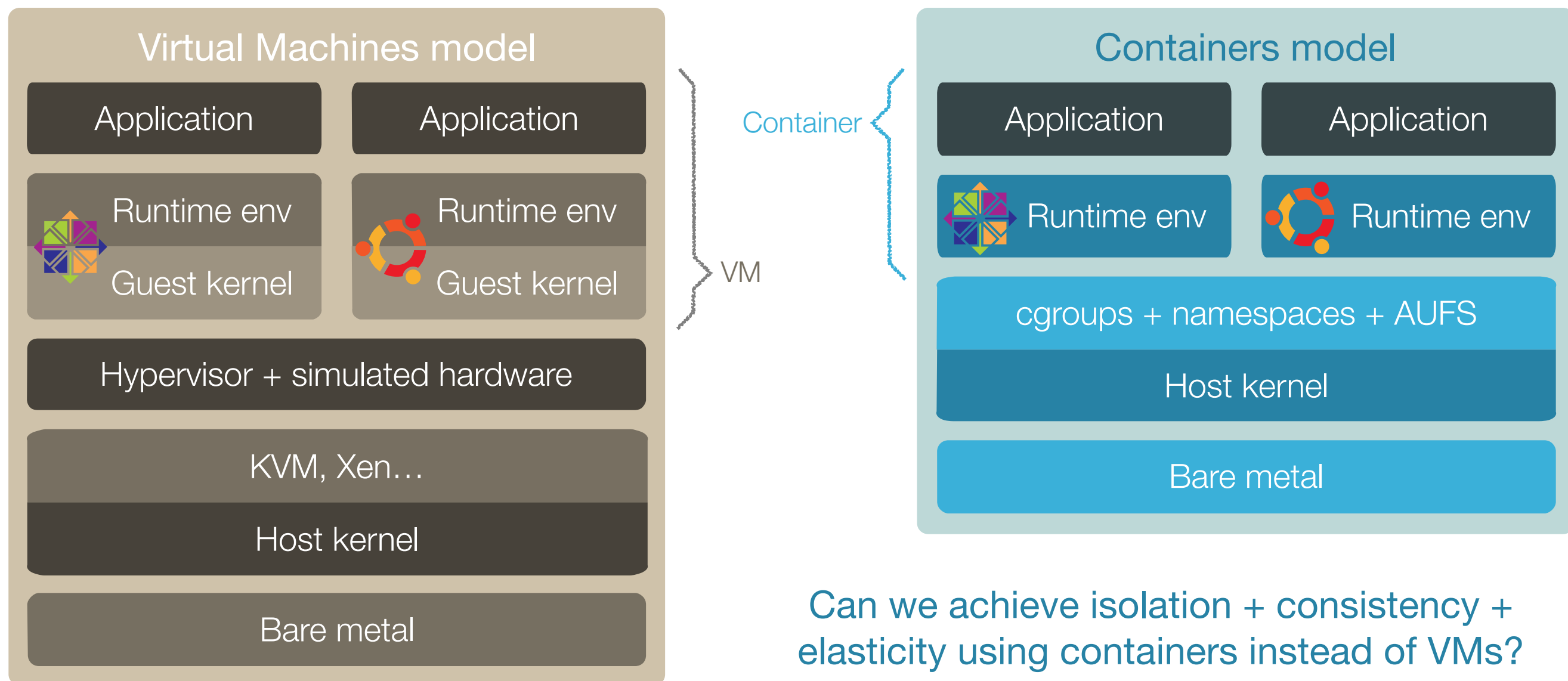


# Containers





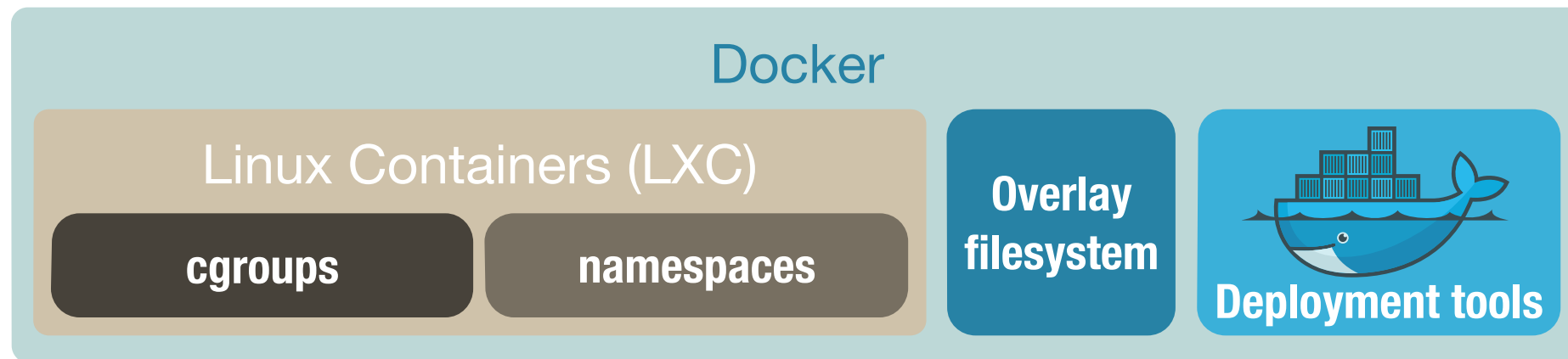
# Virtual machines and containers



- Containers are not lightweight VMs: they are **chroot on steroids**
- Less features than VMs: no custom kernel or virtual hardware
- Applications run on the bare metal: same kernel with isolation



# Docker makes containers trivial



- Docker is built on top of Linux Containers: kernel-level sandboxing  
Makes them usable with VM concepts: “instantiate” a “base image”
- Deployment of a base image takes a fraction of second  
Base image not cloned: overlaid with a read-write filesystem
- Layers are a key concept: versioning, branching and tagging  
Using filesystem layers like Git commits: save only the diff layer



# Make Docker containers useful for us

- CernVM-FS

Our software is there: make it usable from within a container

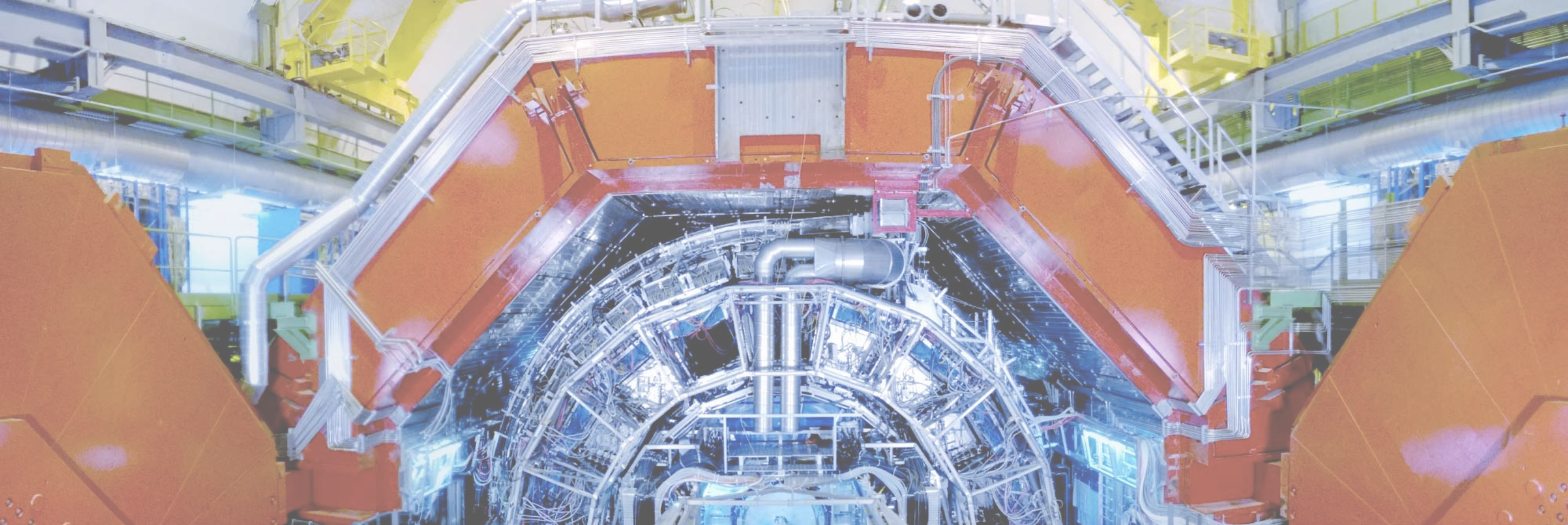
- Running the full CernVM environment as a Docker container

Without cloning the image: take the root directory from CernVM-FS

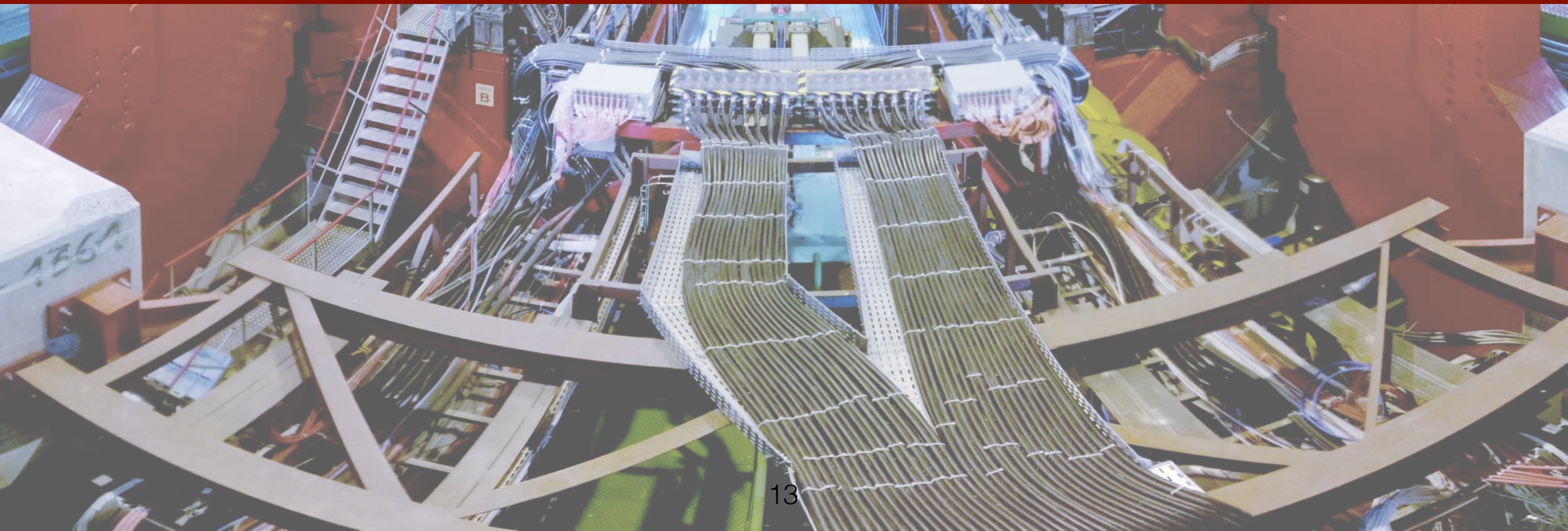
- Deploy containers on the large scale

Simple and scalable system relying on existing technologies





# Running CernVM as a container





# Overlay in CernVM and Docker

## Overlay in CernVM



Writable overlay: running VM

Read only root filesystem from CVMFS

CernVM and Docker both use overlays to work: use Docker overlay mechanism for CernVM containers

## Overlay in Docker



Writable overlay: running container

Read only diff layer #n

Read only diff layer #2

Read only diff layer #1

Base image of runtime environment

- Mount CernVM root directory from CVMFS on the host node
- Docker's AUFS diff layers are directories: we can trick Docker
- CernVM mount point: diff directory of a dummy Docker image



# Running CernVM inside a Docker container

```
# Only once: register the dummy CernVM image (empty)
docker-cernvm --tag alice/cernvm register
```

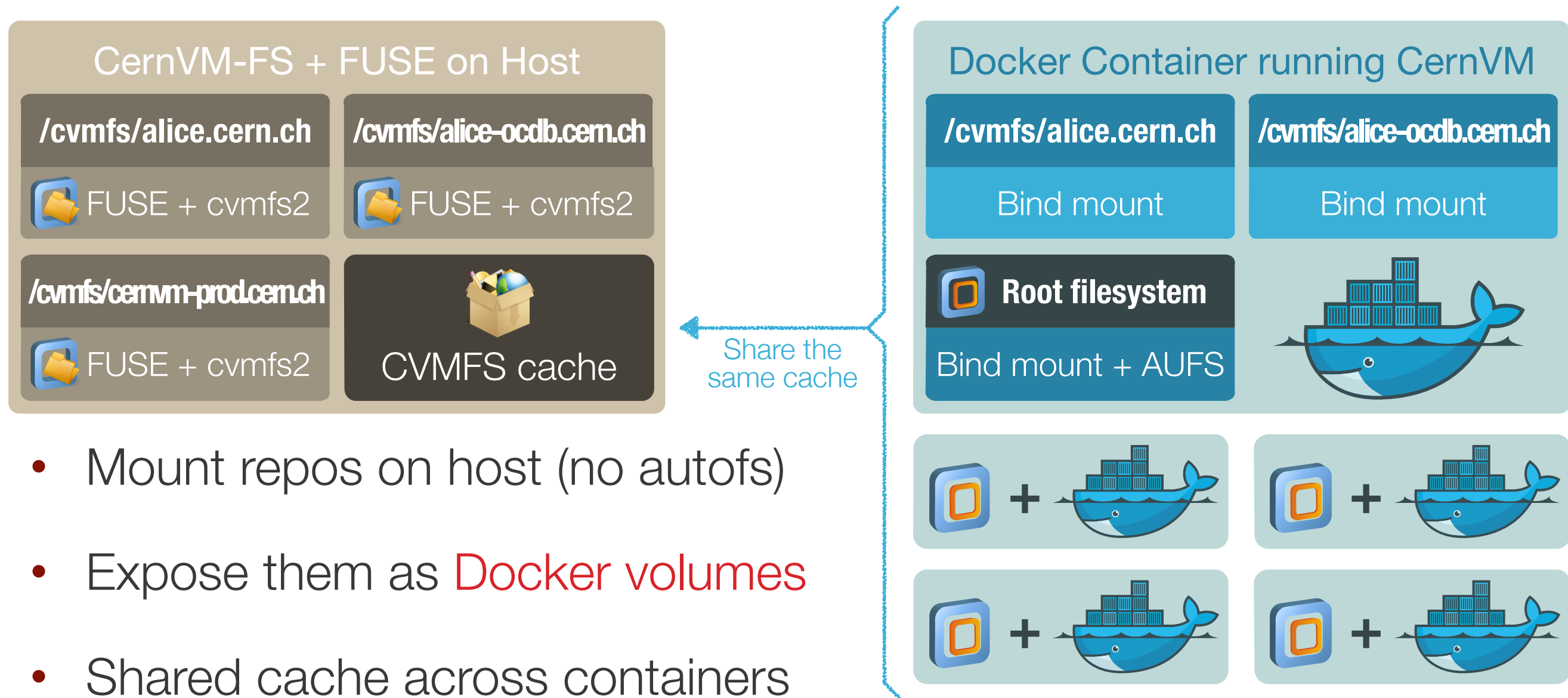
```
# Mount CernVM root from CVMFS (needs root privileges)
sudo docker-cernvm --tag alice/cernvm mount
```

```
# Run it like any other container
docker run -i -t --rm alice/cernvm
```

```
[root@3840c95d737c /]# cat /.installed_cernvm-system-*
CERN Virtual Machine 3.3.0.22
```

- Complete doc and tool: [github.com/dberzano/cernvm-alice-docker](https://github.com/dberzano/cernvm-alice-docker)
- **Note:** saving diffs may not make sense if changing CernVM base image

# Using CVMFS inside containers



Persistent: it will not be thrown away with the container

- No need to configure FUSE and CVMFS inside the container

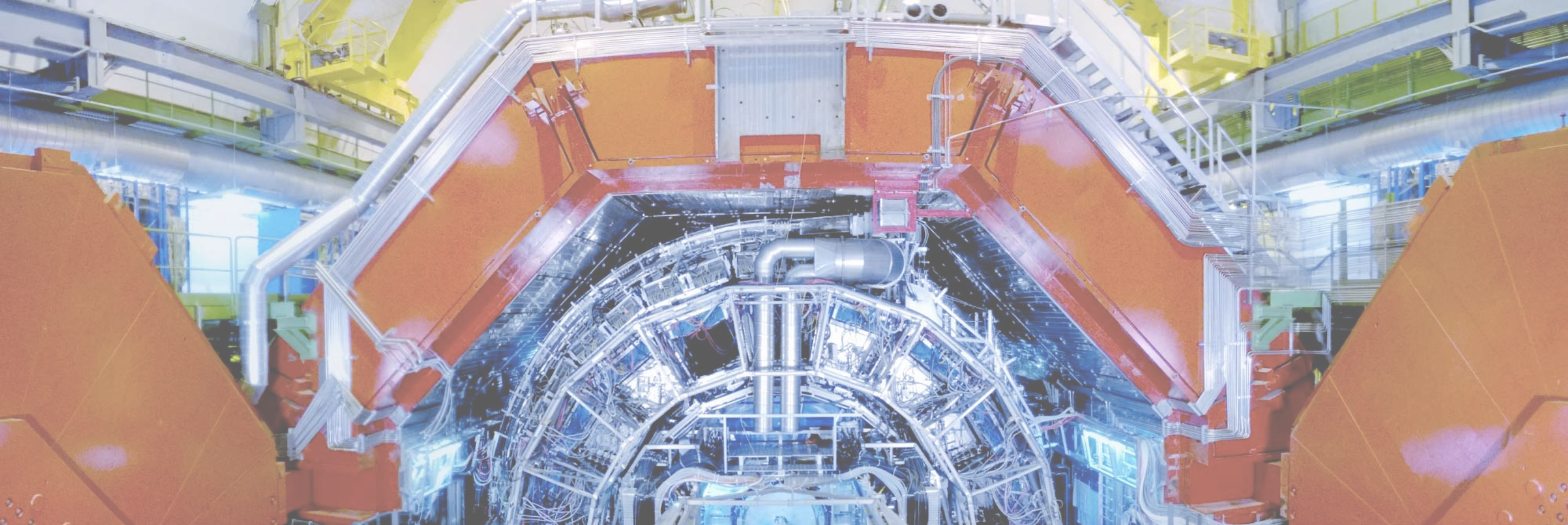
Possible with privileged containers, but we lose security



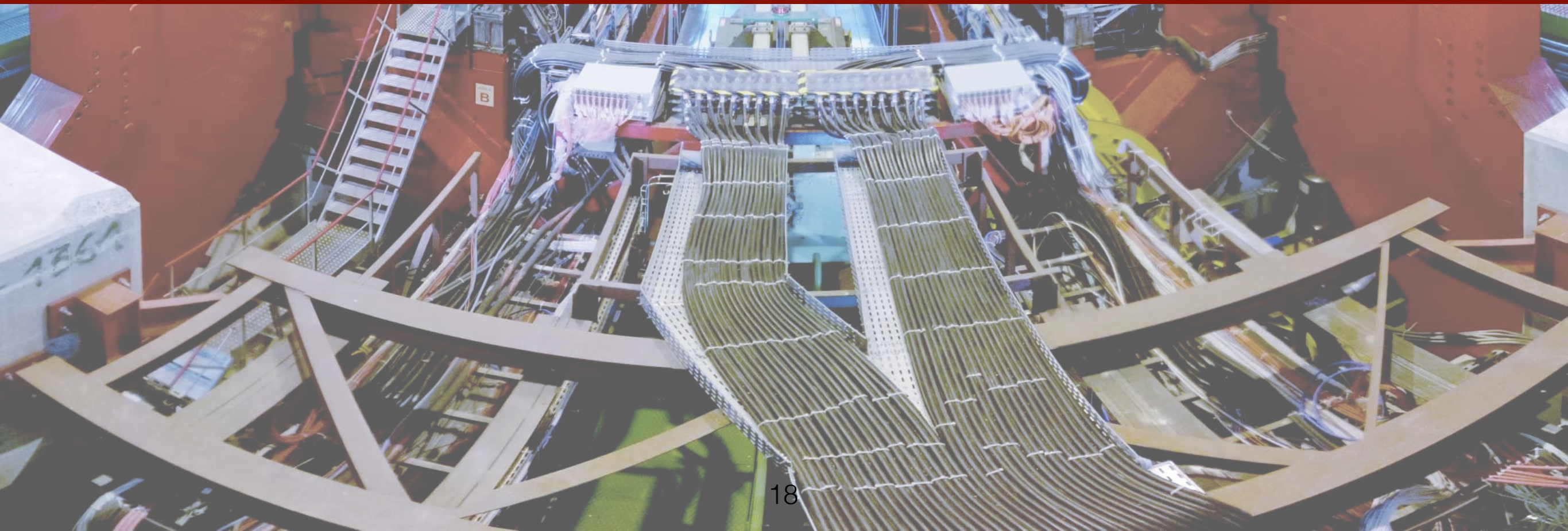
# What we are working on

- `docker-cernvm` shows that it is **natural** to run CernVM as a container
- CernVM releases and updates are snapshots in CVMFS
  - It is currently not possible in CVMFS to mount different snapshots of CernVM at the same time: **we are working on it**
- Objective: show CernVM versions as Docker images automatically
  - Select version of CernVM with **docker run** and not manual mount: enables saving diff layers and **deploying** them everywhere
- Natural way to deploy our container appliances on the large scale
  - CVMFS is consolidated, CernVM is full-fledged yet **lightweight**





# Pilot containers



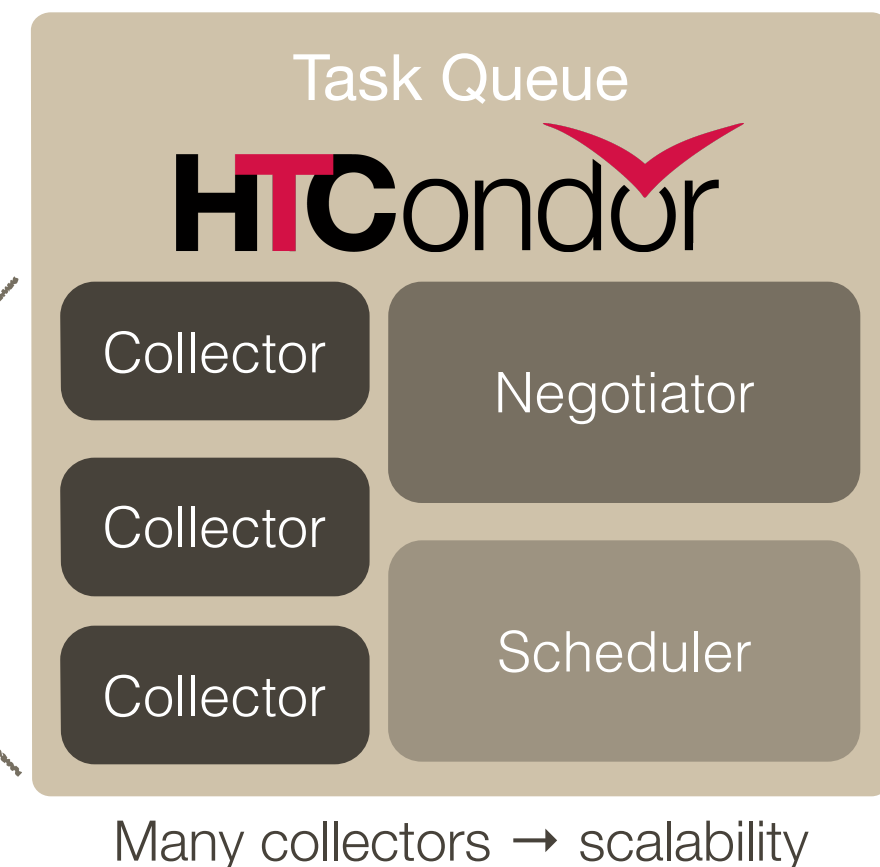
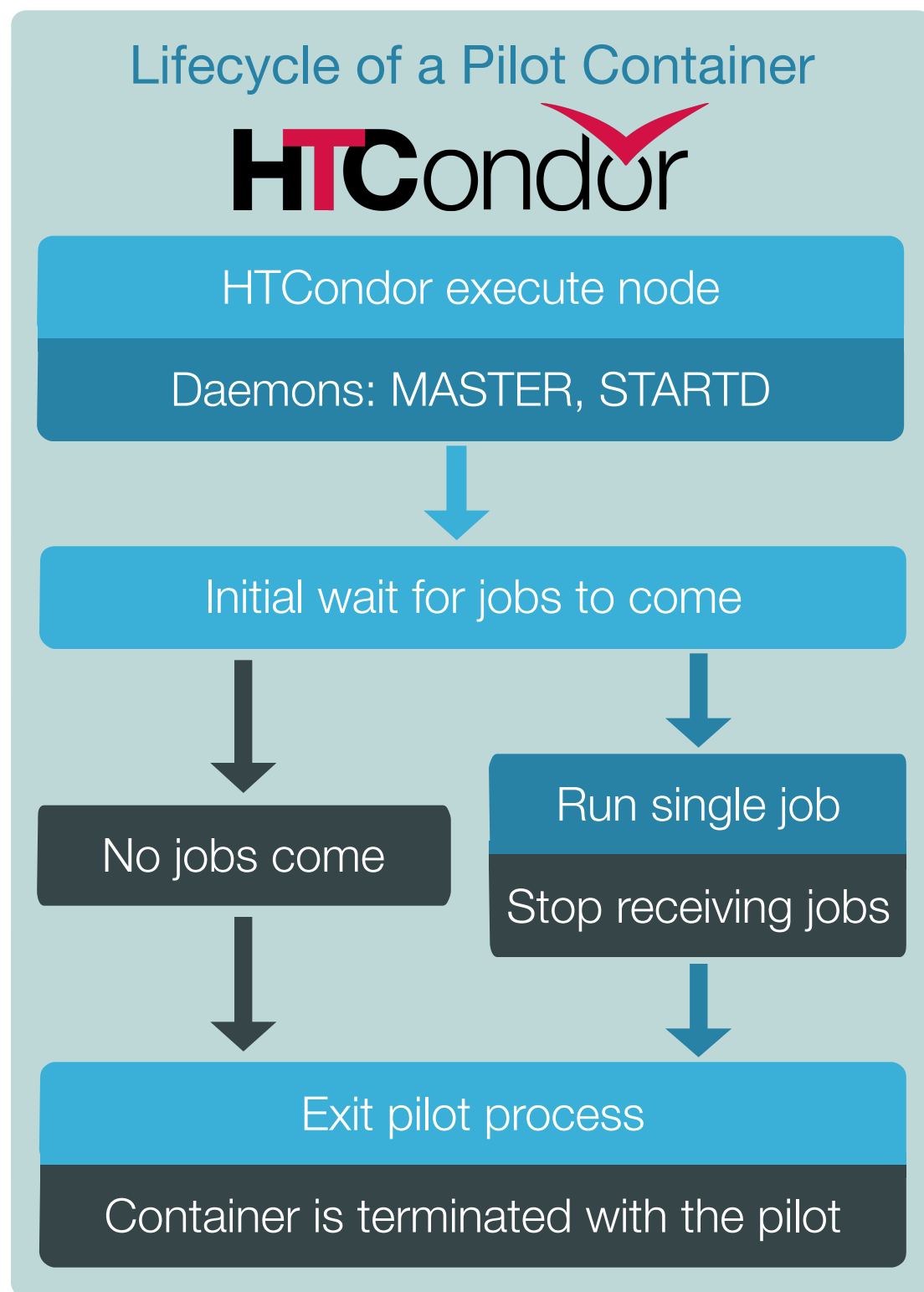


# A “docked” analysis facility

- Our virtual analysis facilities run HTCondor jobs
  - Use containers instead of VMs for running HTCondor jobs: we will be able to run transparently our current VAF use cases
- Container deployment time and overhead: ~zero
  - We can afford one container per job: rolling updates is faster and elasticity is as effective as the job scheduler
- Scalable and opportunistic: pull instead of push scheduling
  - We don't want to schedule containers: focus on scheduling jobs



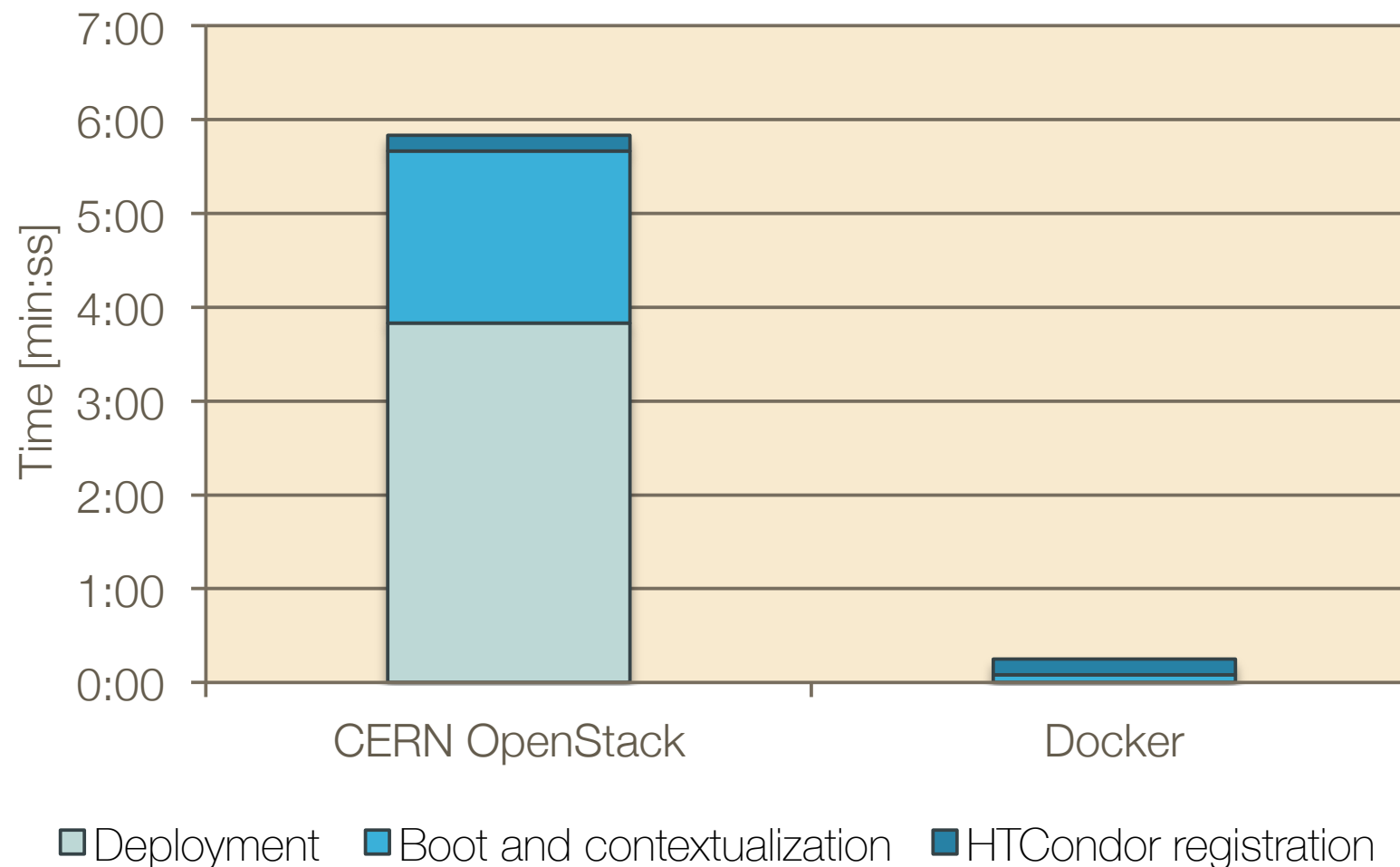
# The pilot container



- Pilot model: efficient scheduling
- Self-contained: only Docker needed
- One configuration for dedicated, opportunistic, volunteer computing

- Prototype and doc: [github.com/dberzano/cernvm-alice-docker](https://github.com/dberzano/cernvm-alice-docker)

# Deployment times compared

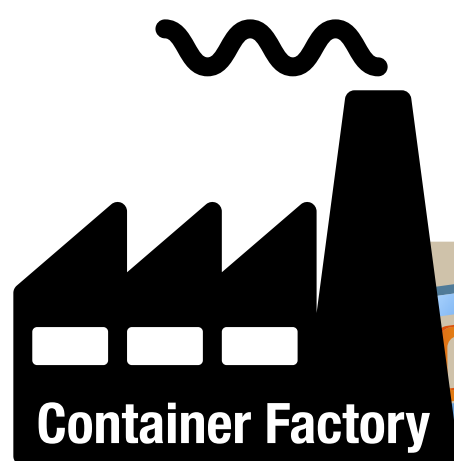
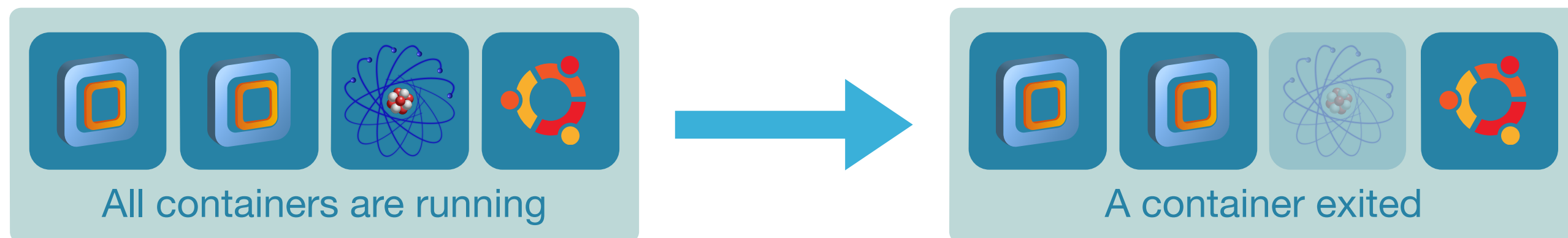


Requested 48 cores: 12 VMs vs. 48 containers

- Deployment negligible: only overlay
- “Boot” negligible: config and start HTCondor, no other daemons
- Within 15 seconds the container is ready to receive jobs



# The container factory

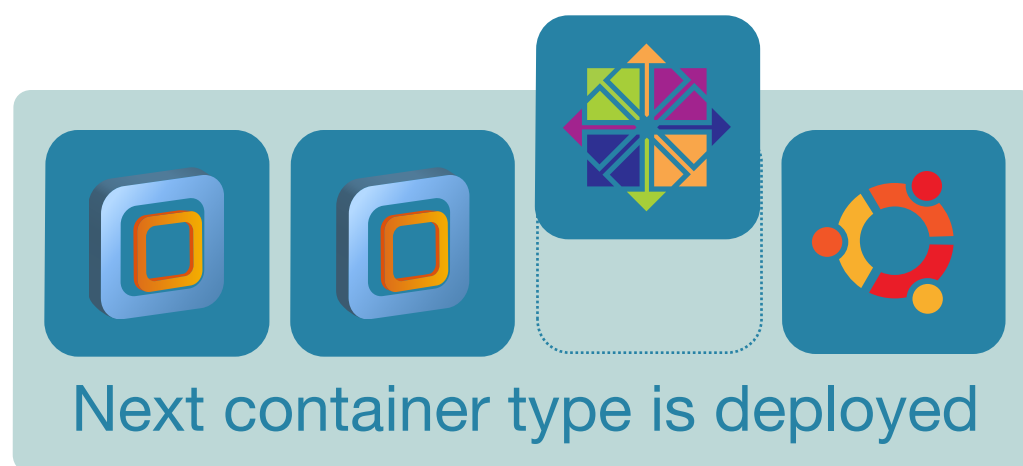


One per host:  
no central control

Produce a new container when requested: round-robin over types



Deploy



- Run pilot containers
- May change num of containers at runtime
- “VAC for containers”

[www.gridpp.ac.uk/vac](http://www.gridpp.ac.uk/vac)

# Rolling updates with Docker containers

```
# Current containers are using image v0.1: update it
docker run -d alice/slc6:v0.1 yum upgrade -y
```

```
# When done, get ID of this container
docker ps -a
```

```
# Save container with a new tag, and "latest" as well
docker commit 3fcc69cc1822 alice/slc6:v0.2
docker tag -f alice/slc6:v0.2 alice/slc6:latest
```

- New containers will pick latest version immediately
- Can be easily automatized
- One job = one container: no drain, no resources wasted





# What we are working on

- Develop and test the scalability of a lightweight pilot factory

Very much suitable for opportunistic use cases

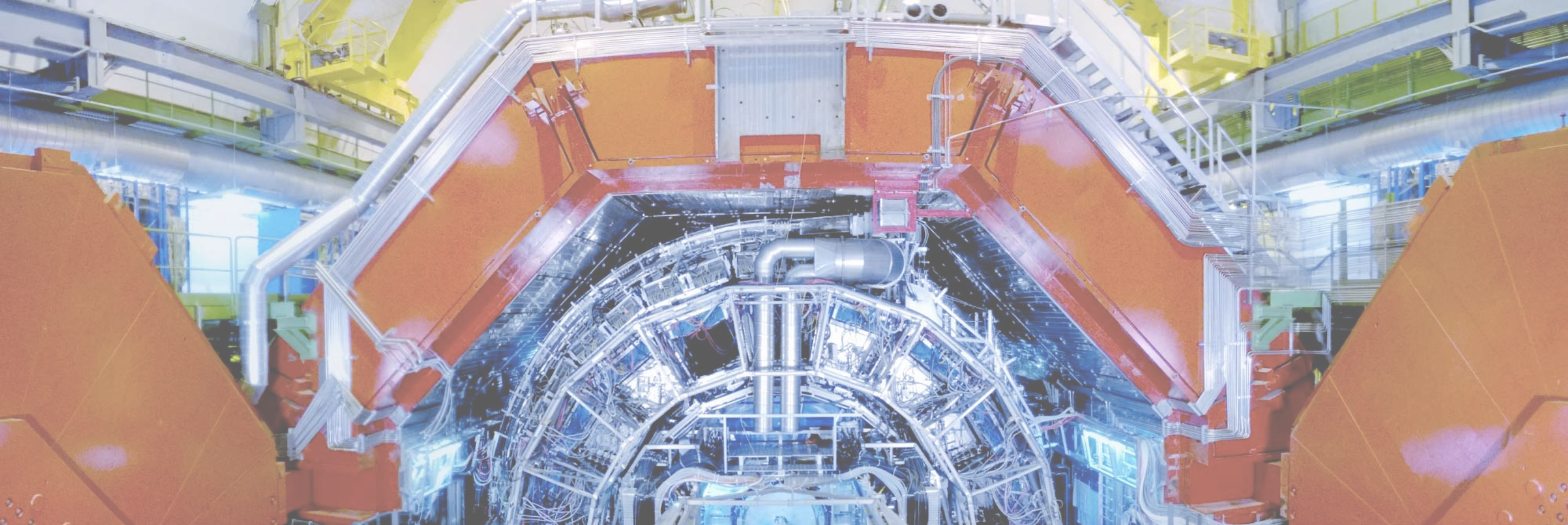
- Evaluate running pilot containers with Apache Mesos

Mesos covers a broader use case: pilot containers fit within the idea of “double scheduling” and cover our current batch cases

- Port our current virtual analysis facility applications with containers

Since they all use HTCondor as interface they should run transparently in the new system





Thanks!

