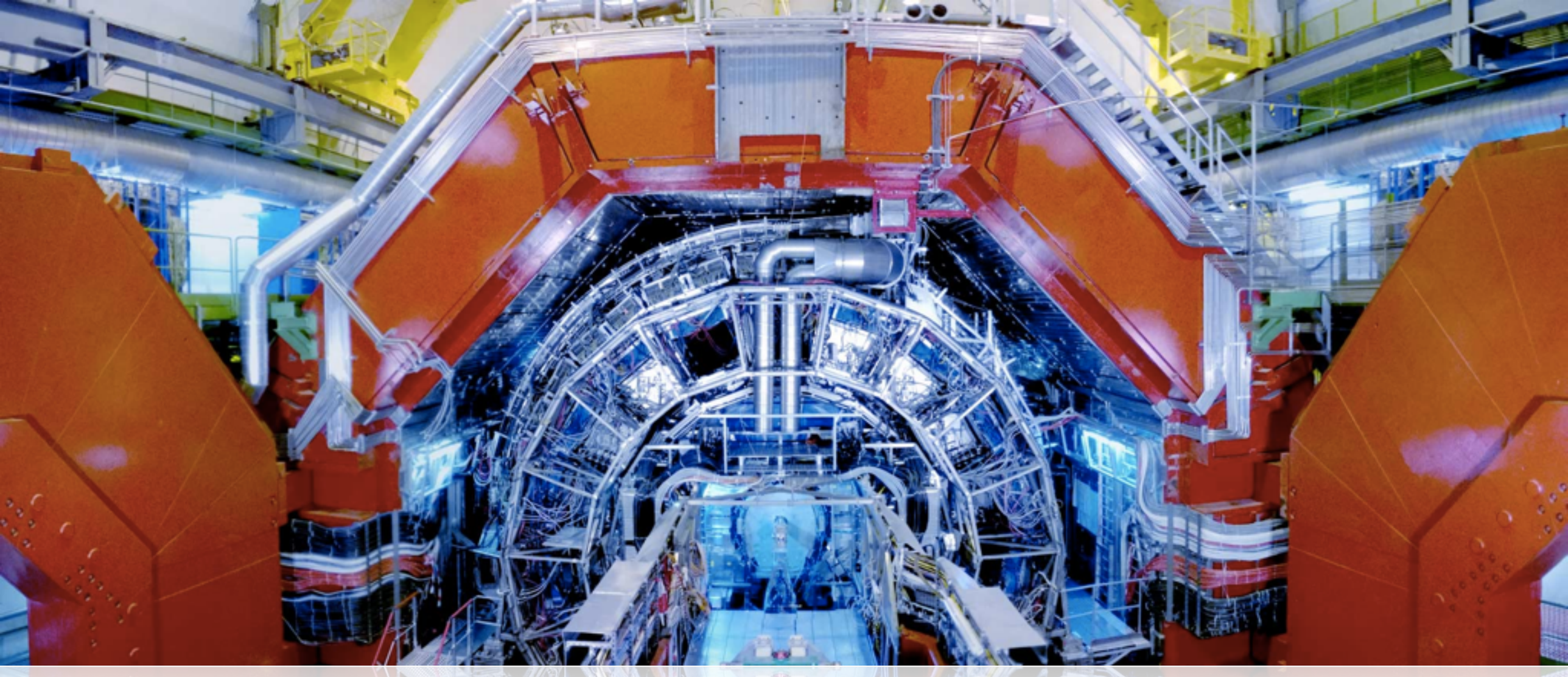


From CAF to VAF and beyond

Dario Berzano

ALICE Offline

ALICE Offline Week - Nov 24-27, 2015

Background: from CAF to VAF

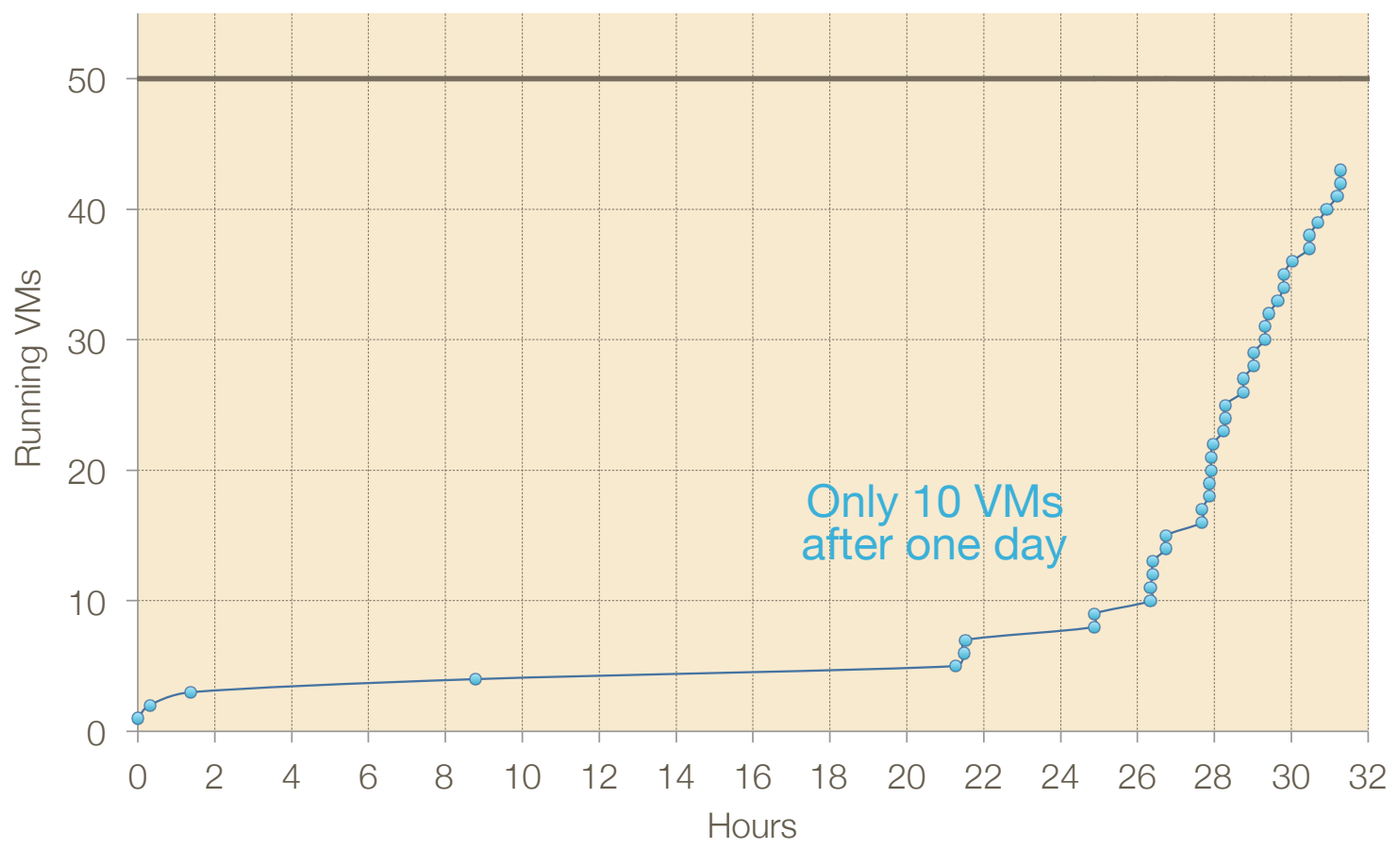
- May 2015: old CAF has been migrated to VAF
  - Last Offline Week: indico.cern.ch/event/400521/
- Infrastructure: Virtual Machines running on CERN OpenStack
  - Offload management of hardware resources
    - Scale by cloning VM instances at will
    - Easier support and maintenance: better use of manpower
- Based on PROOF on Demand
  - Users have personal PROOF daemons they can restart in case of problems *(self servicing)*

CAF status and news

- Quota of 50 VMs, fully used:
  - 1 master node
  - 3 login nodes (alivaf-00[1,2,3])
  - 46 workers (4 CPUs each, 2 GB/core)
- Added monitoring with MonALISA
  - dberzano.github.io/alice/vcaf/#current_status
- aaf.cern.ch now points to the new documentation
- Low usage: only 9 unique users in the last month
  - Some of them are power users: CAF support explicitly requested, *e.g.* the Muon community

Dynamically scaling CAF/1

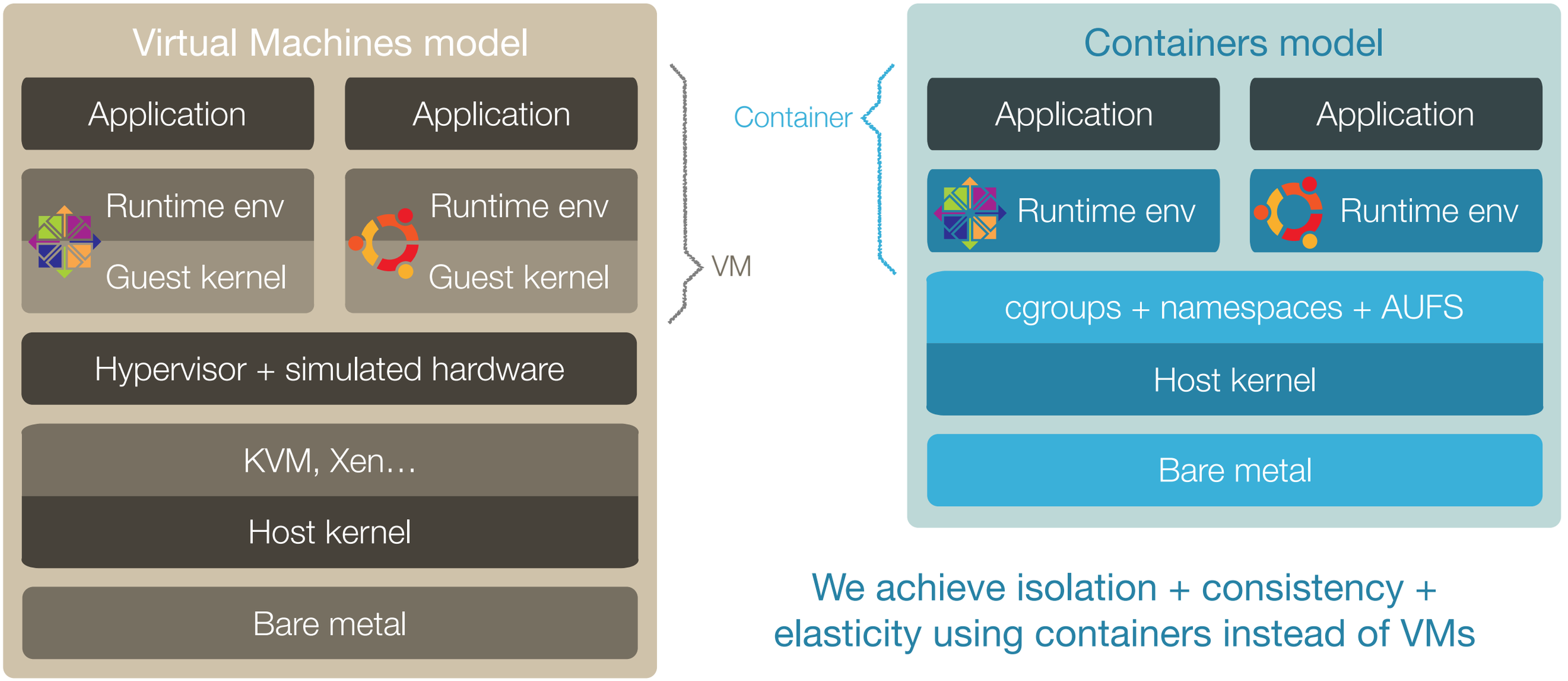
- CAF is underutilized
  - Used in bursts, by a bunch of users, mostly for special cases
- Original VAF: automatically scale by changing number of VMs
- CERN VAF “dynamic” deployment has issues:
  - A VM starts slowly (15 to 30 minutes) on CERN OpenStack
  - Impossible to get from zero to ~50 in time
  - First deployment: almost 36 hours…



Dynamically scaling CAF/2

- CAF is essentially a semi-static deployment
  - Still based on elastiq - github.com/dberzano/elastiq
  - Configured to keep 50 nodes up and never shutdown
  - Needed due to the difficulty to obtain resources
    - elastiq still facilitates the replacement of non-responding nodes but not used for dynamically scaling the cluster
- We face a problem here…
  - 200 CPUs statically allocated are idle most of the time
  - We are wasting resources we can use for something else

Containers to the rescue/1



We achieve isolation + consistency + elasticity using containers instead of VMs

- Containers provide an isolated sandbox to the application: an entire OS root directory seen only by the application
- Processes are isolated, as well as CPU, RAM, swap
- No virtual hardware (runs on bare metal): deployment much faster

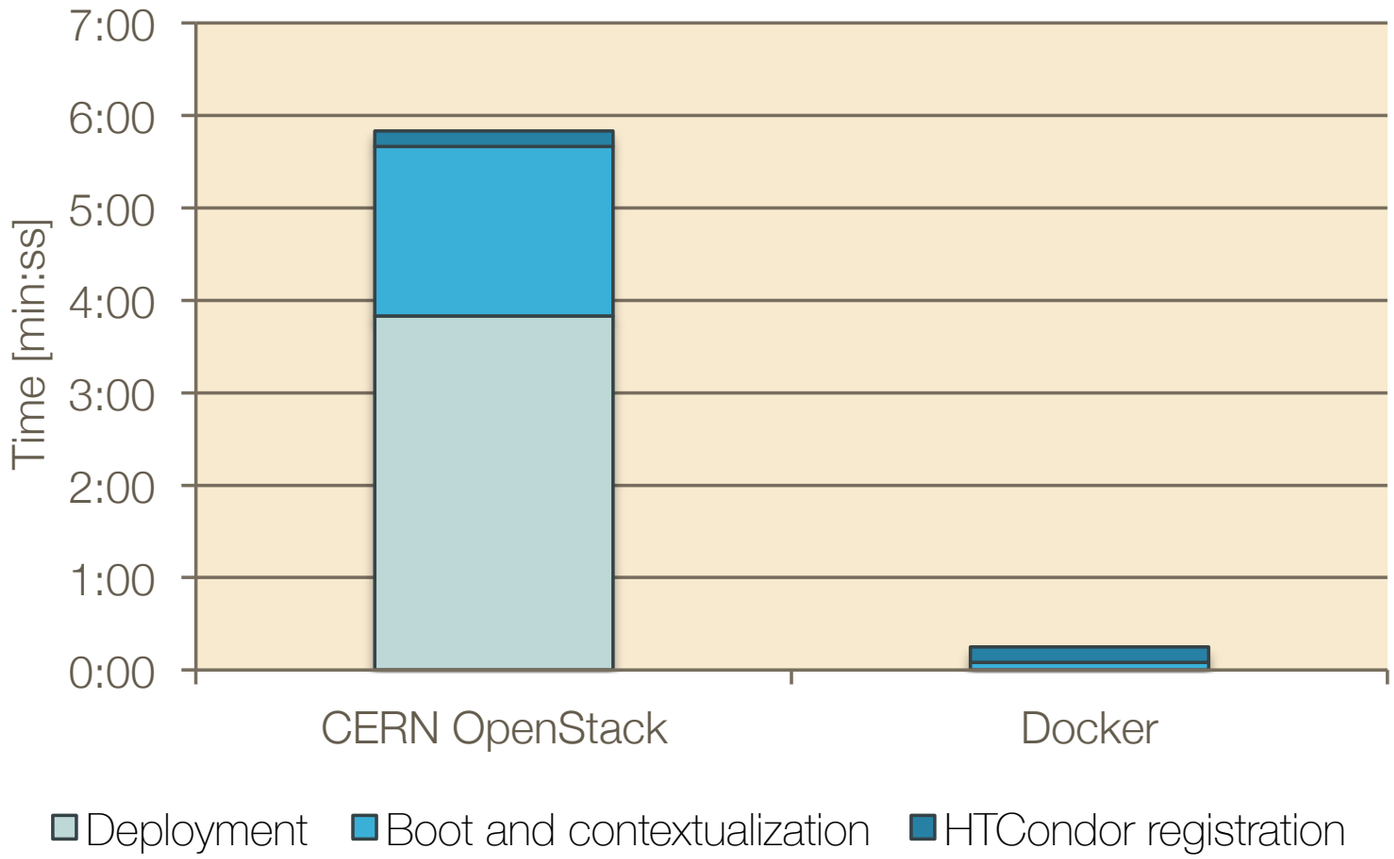
Containers to the rescue/2

- Containers are easy to use. Let's get practical.
- Instant gratification (with Docker):

```
# Install Docker (works on nearly any Linux)
curl -L get.docker.com | bash
# Drop into a CentOS 6 shell. Starts immediately.
docker run -it --rm centos:centos6 /bin/bash
# List processes, play around...
ps auxww
```

- When you exit the shell, you terminate the container
- You can preserve data by attaching external directories
- Docker has a public "registry" of common OS images

Containers are FAST to deploy



Requested 48 cores: 12 VMs vs. 48 containers

- VMs vs. containers running HTCondor. This is what backs CAF.
- No boot process in containers: just start HTCondor
- Starts in < 1 s, ready to receive jobs in < 15 s
- See also j.mp/kvm-vs-lxc for some interesting benchmarks

Intermission: containers and Analysis Train tests

- Analysis Trains are validated on a local machine before the Grid
  - Uses Ubuntu as OS with a special AliPhysics build for Ubuntu
- New build system in place
  - We wanted to use AliPhysics directly from CVMFS and test in a Grid-like environment (SLC6, not Ubuntu)
  - Wrap test inside SLC6 a container. As easy as prepending:

```
docker run alisw/slc6-builder /my/train/script.sh
```

  - Completely transparent to the end users
  - We test exactly what we deploy on the Grid: more reliable
- More tech details tomorrow…

Deploying containers on the scale

- Containerized application: my application can run anywhere as it is wrapped with its runtime environment
- Containers can be deployed on the scale: a variety of solutions exist (Mesos, Kubernetes, Swarm…)
- Mesos is used by important companies (*e.g.* Twitter, Apple…)
- We have our own Apache Mesos cluster:
  - Built essentially for supporting the new build system
  - We can use it for many other things, including the CAF
- Diverse resources combined: virtual/physical, with(out) containers:
  - ~20 CERN OpenStack VMs, ~5 physical nodes, one Mac Mini

Apache Mesos

MESOS

mesos.apache.org

- Apache Mesos helps orchestrating applications on distributed resources (with an eye on scaling and high availability)
- Mesos brings the knowledge of the amount of resources to your application
- Your application registers to Mesos: it becomes a “framework”
- Your framework receives resource offers from Mesos: yon decide which ones to take and tell Mesos to deploy there

Mesos frameworks

- Mesos works like an “operating system” for the cloud
  - It does nothing alone: it brings the knowledge of resources to your application
  - See indico.cern.ch/event/456663/
- We have some frameworks already sharing our Mesos cluster
  - Jenkins: our build system
  - Marathon: long running tasks with load balancing and recovery
  - Chronos: a crontab for periodic distributed tasks

CAF on Mesos

- A Mesos framework for HTCondor would solve CAF issues:
  - We receive offers from Mesos
  - Jobs waiting? We take as many as we need
  - Jobs terminate? Immediately relinquish
- What we would gain:
  - Fine grained: single core containers *(no multicore VMs)*
  - Truly dynamic: quick deployment *(deploying VMs is slow)*
  - Always promptly available: fair share managed by Mesos
- Mesos framework for HTCondor/CAF is our medium term plan

Writing a Mesos framework

- Integrating an application with Mesos requires development
  - mesos.apache.org/documentation/latest/app-framework-development-guide
- As an exercise we are developing a Work Queue framework
  - Work Queue: ccl.cse.nd.edu/software/workqueue/
  - The framework: github.com/alisw/mesos-workqueue
- Giulio has written the base framework in < 1 day…

ALICE Release Validation on Mesos

- Our Release Validation can run on Work Queue via Makeflow
  - The Work Queue framework is not just an exercise
  - Our plan is to have it on Mesos too
- Making the Release Validation work on Mesos. Plan:
  - Containers: **done**
  - Mesos framework: **testing**
  - Release Validation script revamp: **todo**
- ETA: February 2016

One ring to rule them all

- Mesos is proving to be a productive solution to make different use cases share a diverse set of resources
  - Perfect when we need to scale out
- Current major stakeholder: our build machinery
  - Jenkins: Mesos plugin already existing
- Easy to add new use cases. Notably:
  - The CAF (HTCondor)
  - The Release Validation (HTCondor or Work Queue)