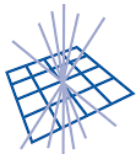
Science & Technology Facilities Council

Virtualisation & containers – a RAL perspective

Andrew Lahiff

STFC Rutherford Appleton Laboratory

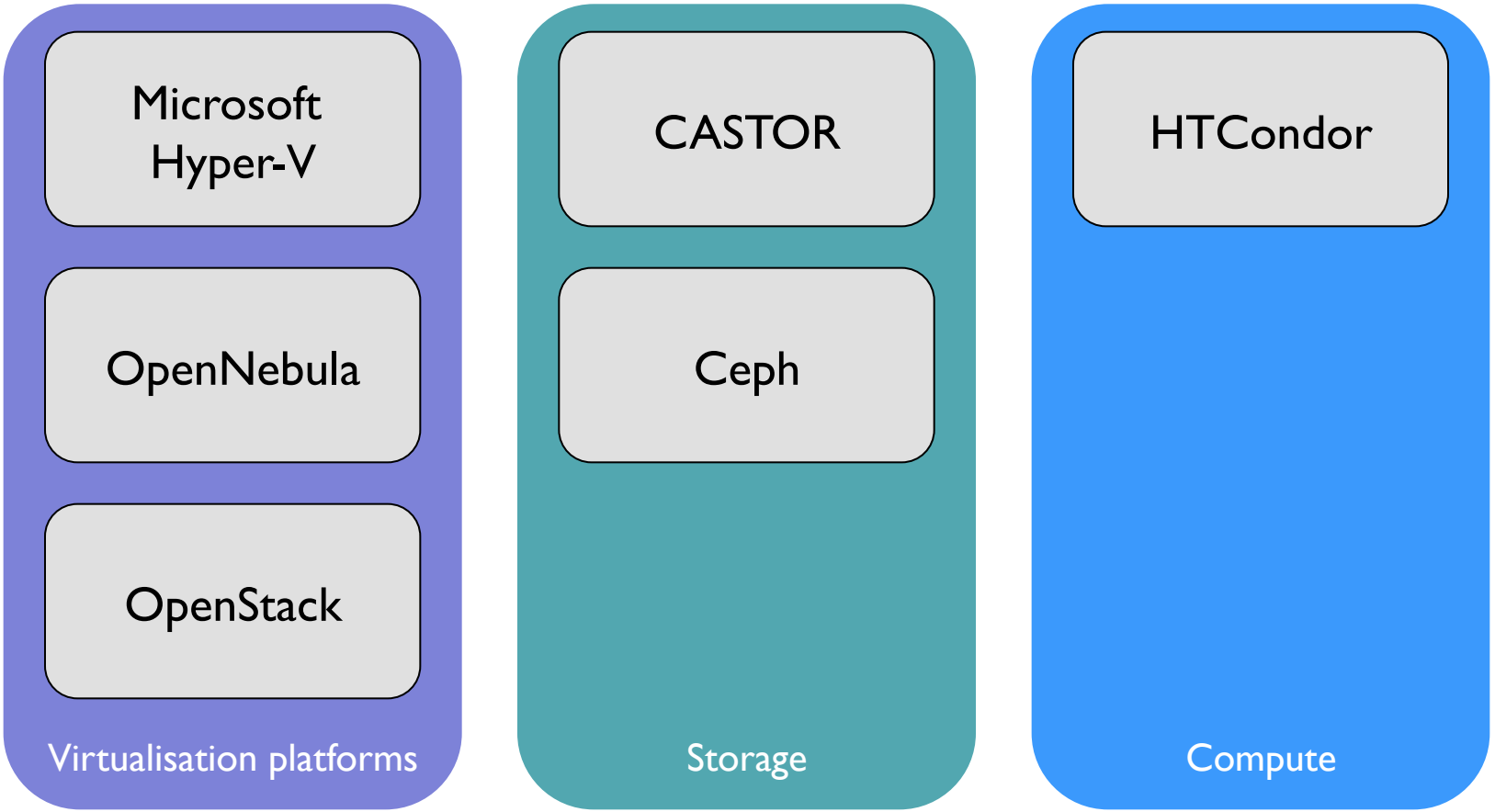
29th June, NorduGrid 2017



GridPP

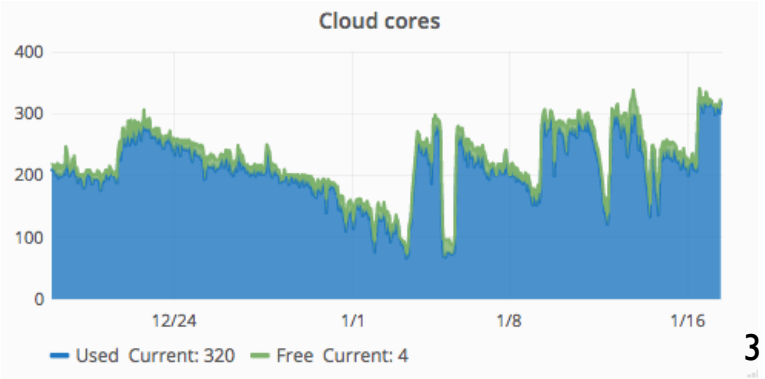
UK Computing for Particle Physics

Infrastructure in the RAL Tier-1



Virtualisation at RAL

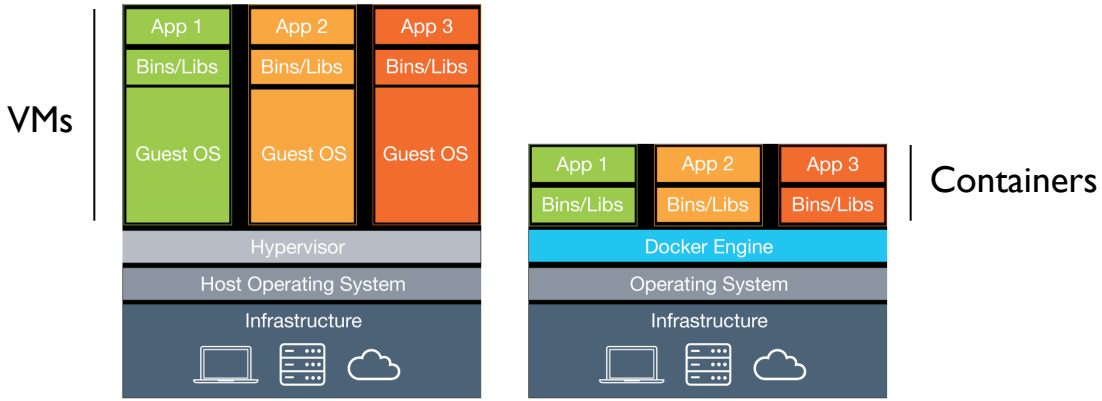
- Enterprise virtualisation platforms for services
  - Microsoft Hyper-V
    - cluster of hypervisors with shared storage
  - Starting to migrate to VMware
- Private clouds, mostly for development work only (so far)
  - OpenNebula
    - ~892 cores, Ceph storage backend
    - for almost 2 years the RAL HTCondor pool has made opportunistic use of otherwise unused cloud resources
    - will be decommissioned later this year
  - OpenStack



Typical cloud usage by HTCondor, generally up to ~300 cores are used

Containers

- What is a container?
  - a container consists of an application and all its dependencies which can be run in an isolated way
  - make uses of kernel features (cgroups, namespaces, ...)
- Benefits include
  - more lightweight than VMs
  - independence from host OS & libraries
  - can be run anywhere, regardless of kernel version or host Linux distribution



Batch systems

- Use of containers for running jobs on worker nodes can be very useful
  - benefits for sites
    - much more flexibility
    - jobs are decoupled from the OS & libraries on the hosts
    - site can upgrade worker node OS without affecting VOs
    - different VOs can use different (Linux) OSs if necessary
  - benefits for VOs
    - jobs can have exactly the same environment no matter what site they're running on

Batch systems

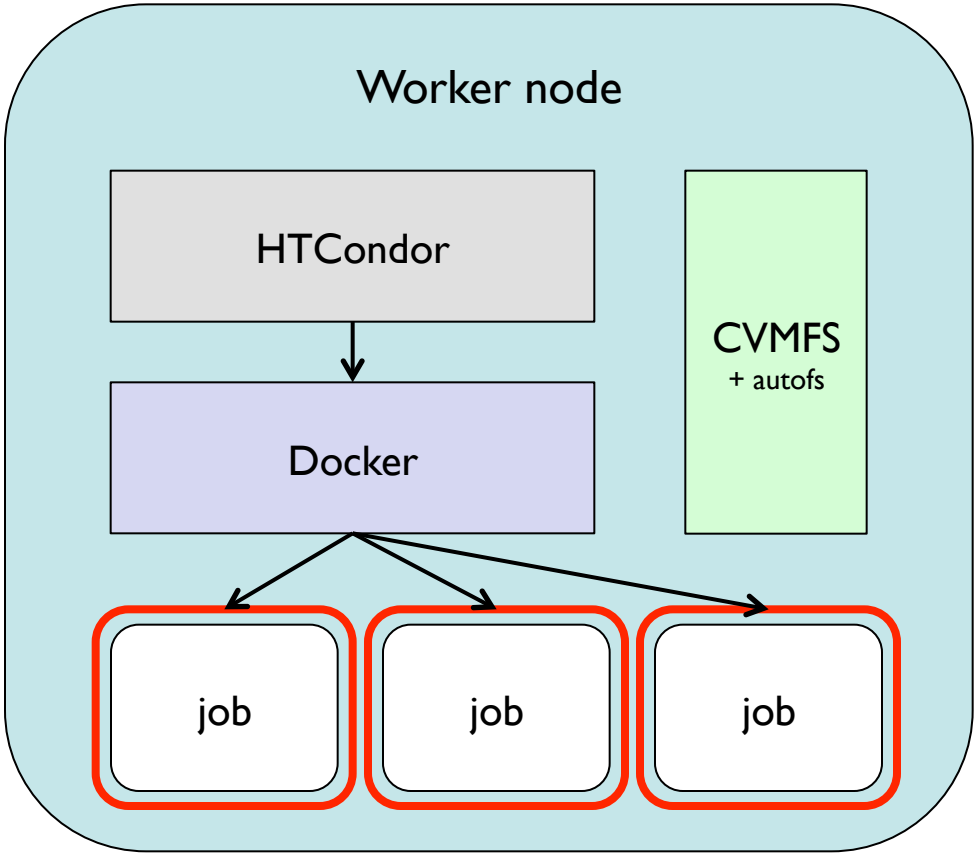
- We migrated from Torque+Maui to HTCondor in 2013
  - gave us the ability to use Linux kernel functionality to isolate jobs
  - cgroups (CPU, memory, ...)
    - resource limits & monitoring
    - ensuring processes can't escape the batch system
  - PID namespaces
    - processes in a job can't see any other processes on the host
  - mount namespaces
    - /tmp, /var/tmp inside each job is unique
- Limitation: all jobs share the same root filesystem
  - e.g. need to run SL6 worker nodes to run jobs in SL6 environments

HTCondor Docker universe

- Docker universe
  - introduced in HTCondor 8.3.6 in June 2015
  - HTCondor runs each job in a Docker container
    - Docker makes it easy to create & manage images
  - successfully ran LHC jobs at RAL in 2015
    - jobs in SL6 containers on SL7 worker nodes
- (Some) features
  - can bind-mount directories/files from the host
    - useful for CVMFS, configuration files
  - all Linux capabilities dropped by default
    - needs to be disabled for jobs requiring glexec

HTCondor Docker universe

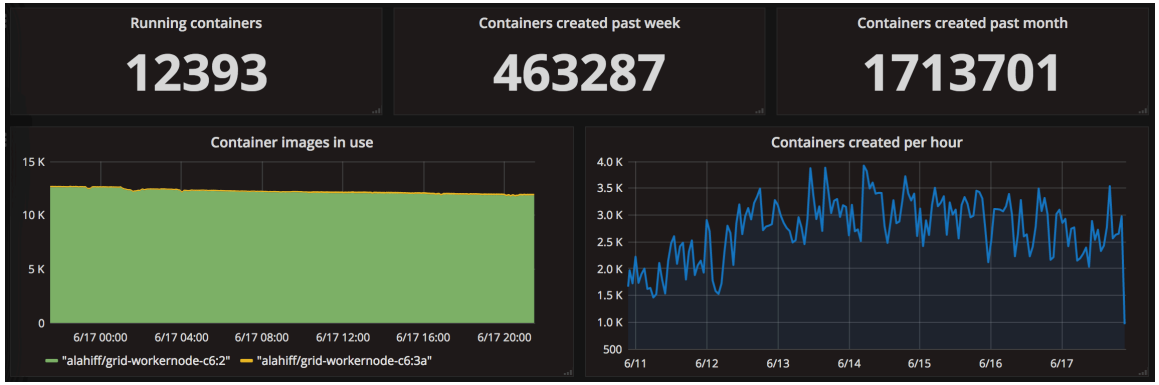
- A RAL SL7 worker node



- containers run as unprivileged pool account users
- users don't have access to the Docker daemon at all
- no way for users to specify arbitrary images via the Grid
- CVMFS available in containers using bind mounts (shared mount propagation)

HTCondor Docker universe

- Earlier this year we migrated fully to the Docker universe
  - all jobs run in containers on bare metal
  - migrated slowly over a period of a few months
  - all existing functionality preserved, e.g. glexec, machine job features, CPU accounting, ...
- Some statistics
  - ~400K containers per week
  - 1.7M past month

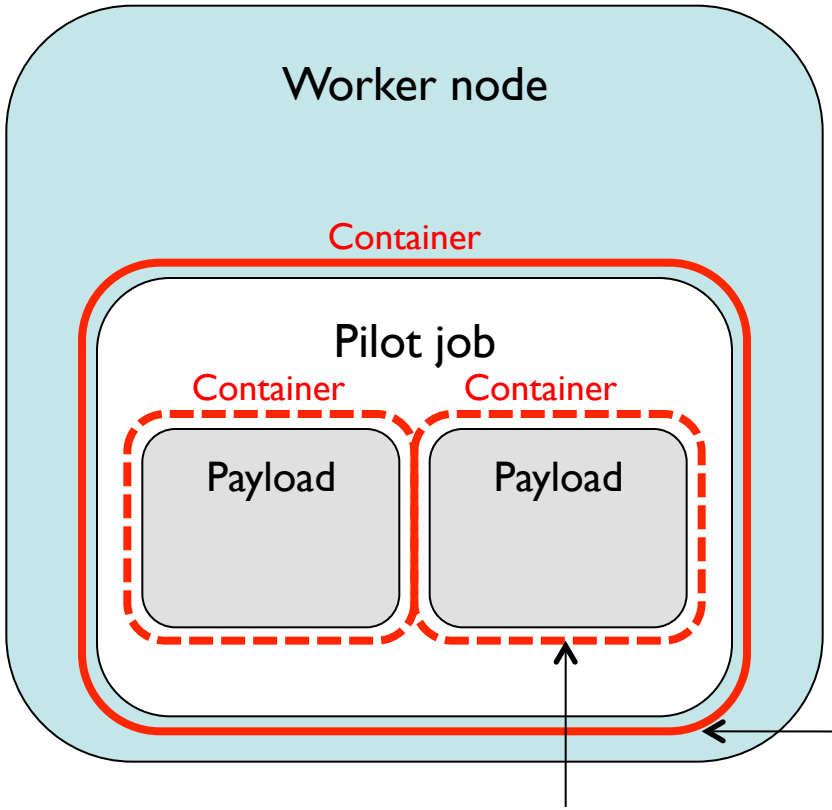


Containers & unprivileged users

- Docker engine
  - daemon runs as root
  - to create to containers you essentially have root access
- What if users want to run their own containers?
  - e.g. run each payload job in containers inside the pilot
  - need to be able to run containers as unprivileged users
- Singularity
  - allows a user to run a process as the same user in a specified environment
  - provides
    - file isolation
    - process isolation

Singularity

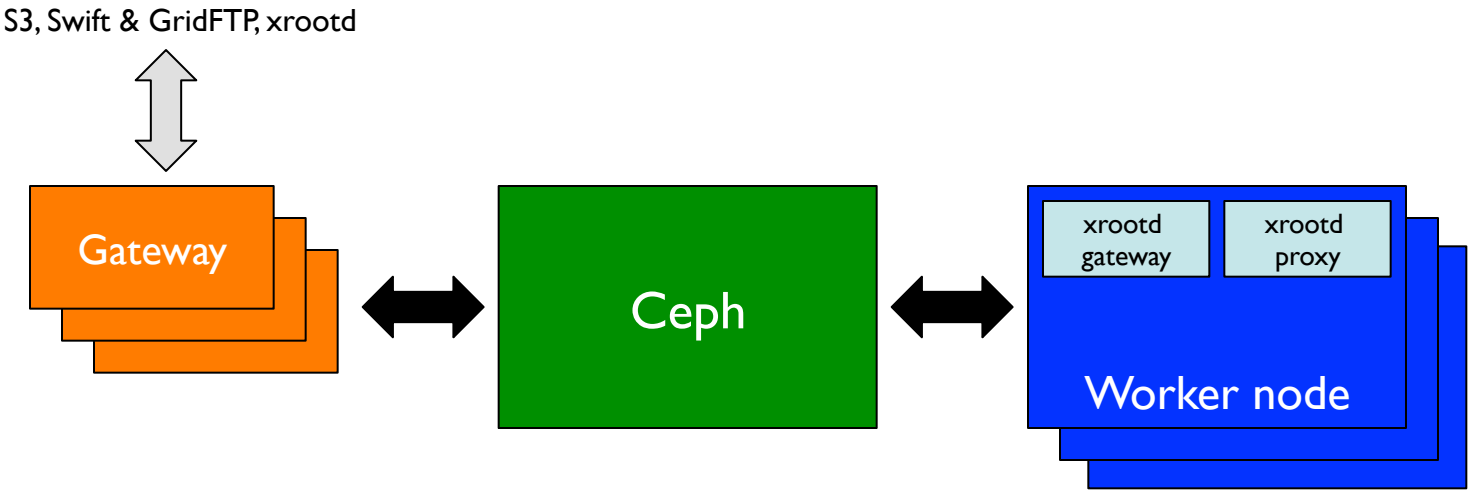
- Will provide Singularity in Centos 7 containers
  - allow VOs such as CMS to run payload jobs in containers



- payload jobs cannot see other processes on the host or even processes from the pilot
- payload jobs cannot see any files from the pilot
- but no limits on resources used by payload jobs

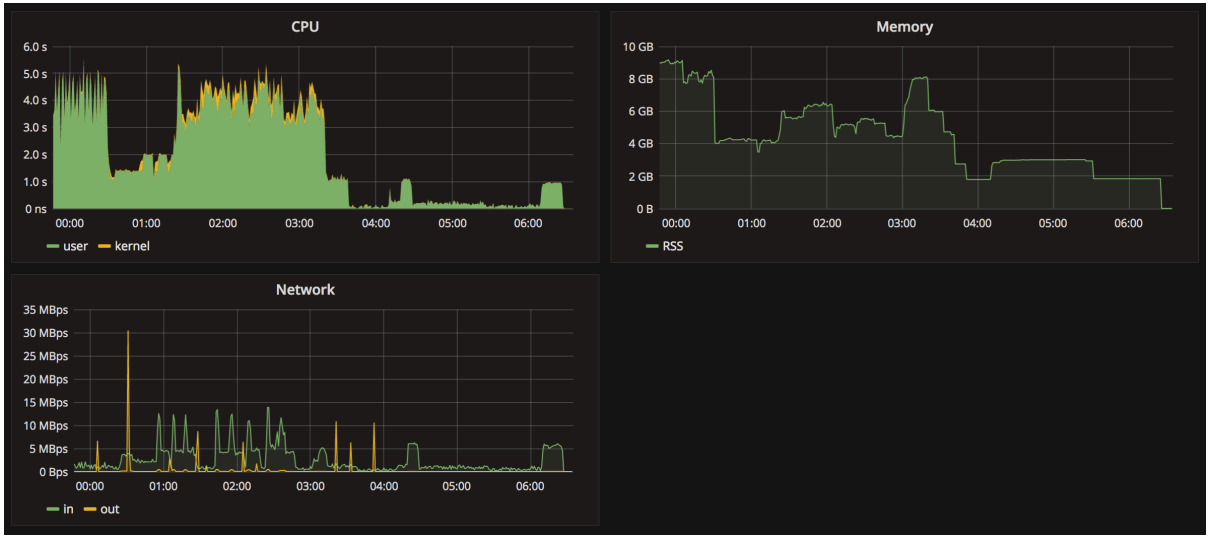
Worker nodes & storage

- We have more than just jobs running in containers on WNs
  - started rolling out xrootd Ceph gateways & proxies onto WNs
- Migrating from CASTOR to Ceph for disk-only storage
  - an important driver for migrating to SL7 worker nodes
    - jobs access data via the local gateways
    - highly scalable xrootd access to Ceph



Monitoring & traceability

- Containers give greater visibility into what each job is doing



Time series resource usage metrics per job, including network

```
170614 11:01:49 144 XrootdXeq: tatls011.5668:88@htcjob5609679_0_slot1_11_pid5408.ralworker pvt IP
v4 login as atlasprod
170614 11:01:49 144 acc_Audit: tatls011.5668:88@htcjob5609679_0_slot1_11_pid5408.ralworker grant
gsi atlasprod@htcjob5609679_0_slot1_11_pid5408.ralworker read atlas:scratchdisk/rucio/panda/ce/f1
/panda.0614095933.558317.lib._11488760.9783001038.lib.tgz
170614 11:01:59 144 XrootdXeq: tatls011.5668:88@htcjob5609679_0_slot1_11_pid5408.ralworker disc 0
:00:10
```

Local xrootd gateway access to storage by user & by job

Network connections per job

Towards the future

- Since on worker nodes we're
  - running jobs in containers
  - running xrootd servers in containers
- Why not just run everything in containers?
  - just doing this on its own wouldn't give many benefits
- However, if the containers were managed by a scheduler
  - instead of having just a dedicated HTCondor batch farm, the same nodes could be used for
    - Big Data, HPC, cloud hypervisors, ...
  - gain lots of more flexibility, help support a wider range of activities
    - 'new' communities becoming more and more important

Container orchestration

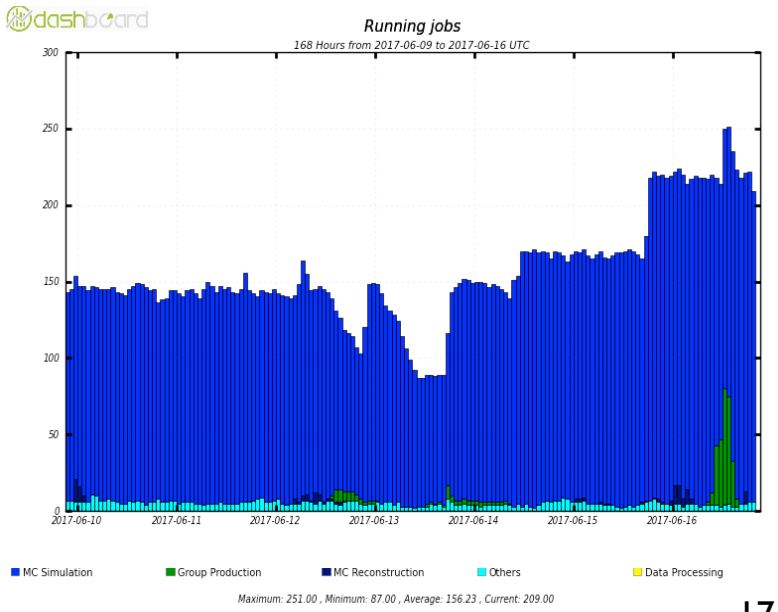
- Simplify managing long-running services by making two fundamental changes
  - Run applications in containers
    - removes the dependency between applications & hosts
    - allows for isolation between different applications
  - Manage applications using a scheduler
    - if an application dies, it will be restarted
    - if a machine dies, the applications running on it will be restarted elsewhere
    - automated staged-rollouts
    - auto-scaling
- Don't think about machines at all, just run your application

Mesos

- Mesos is a cluster manager which
  - enables a large group of machines to appear as a single pool of resources
  - allows you to have multiple schedulers sharing the same resources
- Have had a Mesos cluster running for around 2 years
  - varied in size from 256 to over 7000 cores (currently 352)
- What has it being used for?
  - originally concentrated on investigating the benefits of container orchestration for long-runnng services
  - more recently looking at providing flexible computing infrastructure

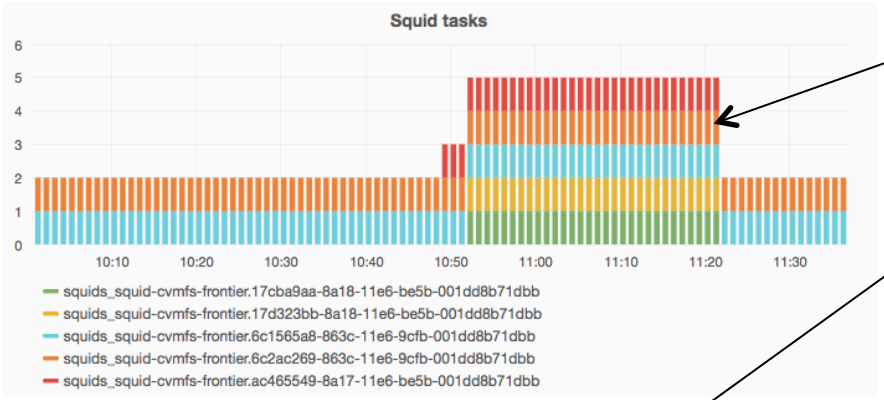
Mesos

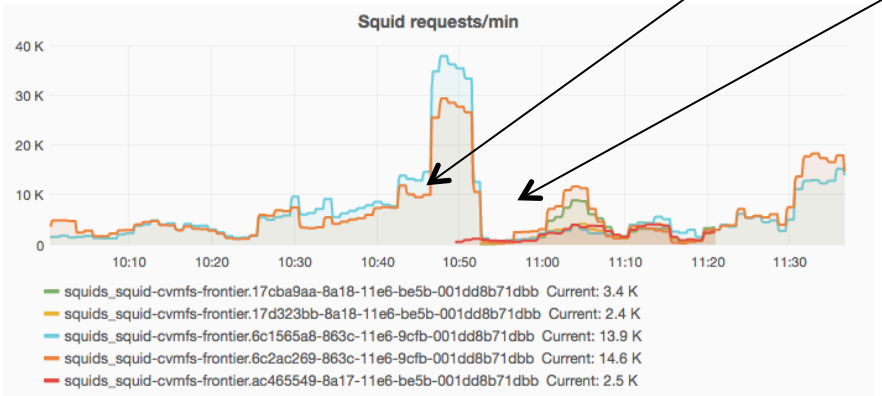
- Last year did tests running > 5000 cores of jobs from all LHC experiments
  - startds + CVMFS running in containers on Mesos joining our production HTCondor pool
- Currently an improved version is running real ATLAS jobs
  - CVMFS provided by (privileged) containers
  - startds in unprivileged containers join a CERN HTCondor pool



Mesos

- Example: number of squid instances changing based on load (request rate)





Each colour corresponds to a unique squid instance in a container

Spike in request rate triggers creation of additional squid instances

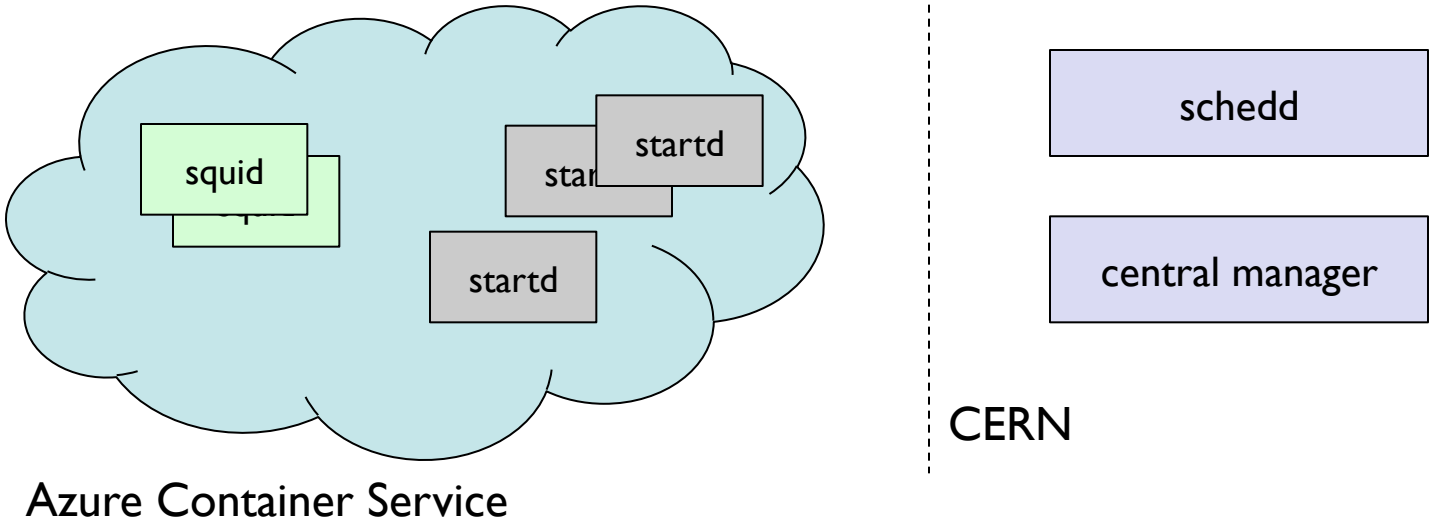
Drop in request rate therefore number of squid instances will be reduced

Kubernetes as an abstraction layer

- Kubernetes is an open-source container cluster manager which can be run anywhere
  - on-premises
  - "as a service" on public clouds (natively or via 3rd parties)
- Using it as an abstraction to enable portability between
  - on-premises resources
  - multiple public clouds
- Benefits compared to traditional ways of using public clouds
  - no vendor lock-in
  - don't need to worry about handling different cloud APIs
  - run workloads on public clouds in the same way that they're run locally

Kubernetes as an abstraction layer

- Did initial testing with CMS CRAB3 analysis jobs
  - RAL, Google (GKE), Azure (ACS), AWS (via StackPointCloud)
- Now running ATLAS production jobs on Azure
  - using "vacuum" model for independently creating startds which join a HTCondor pool at CERN



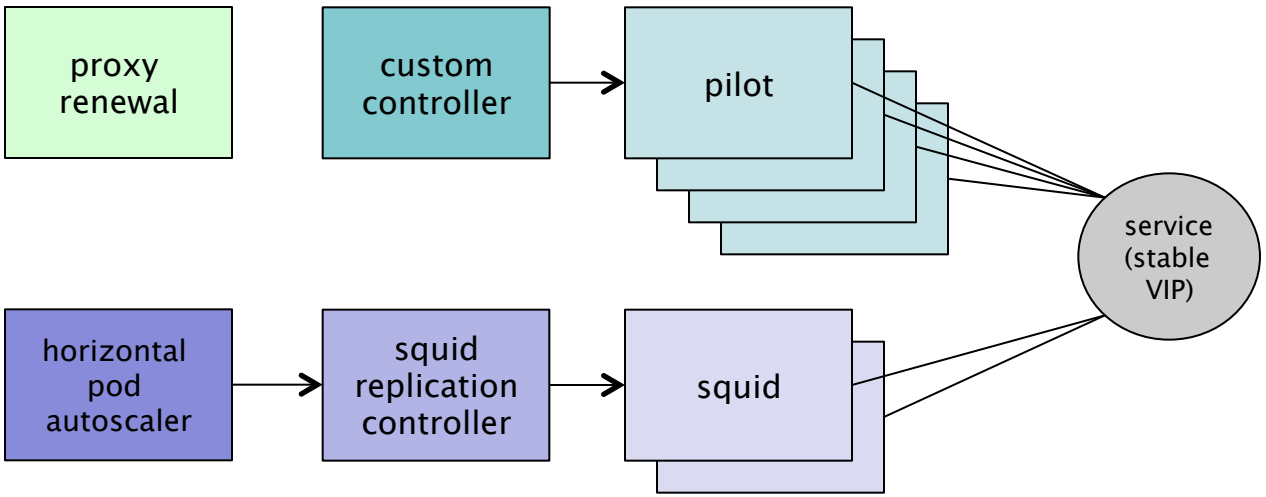
Thanks to Microsoft for an Azure Research Award and to Google & AWS for credits

Kubernetes as an abstraction layer

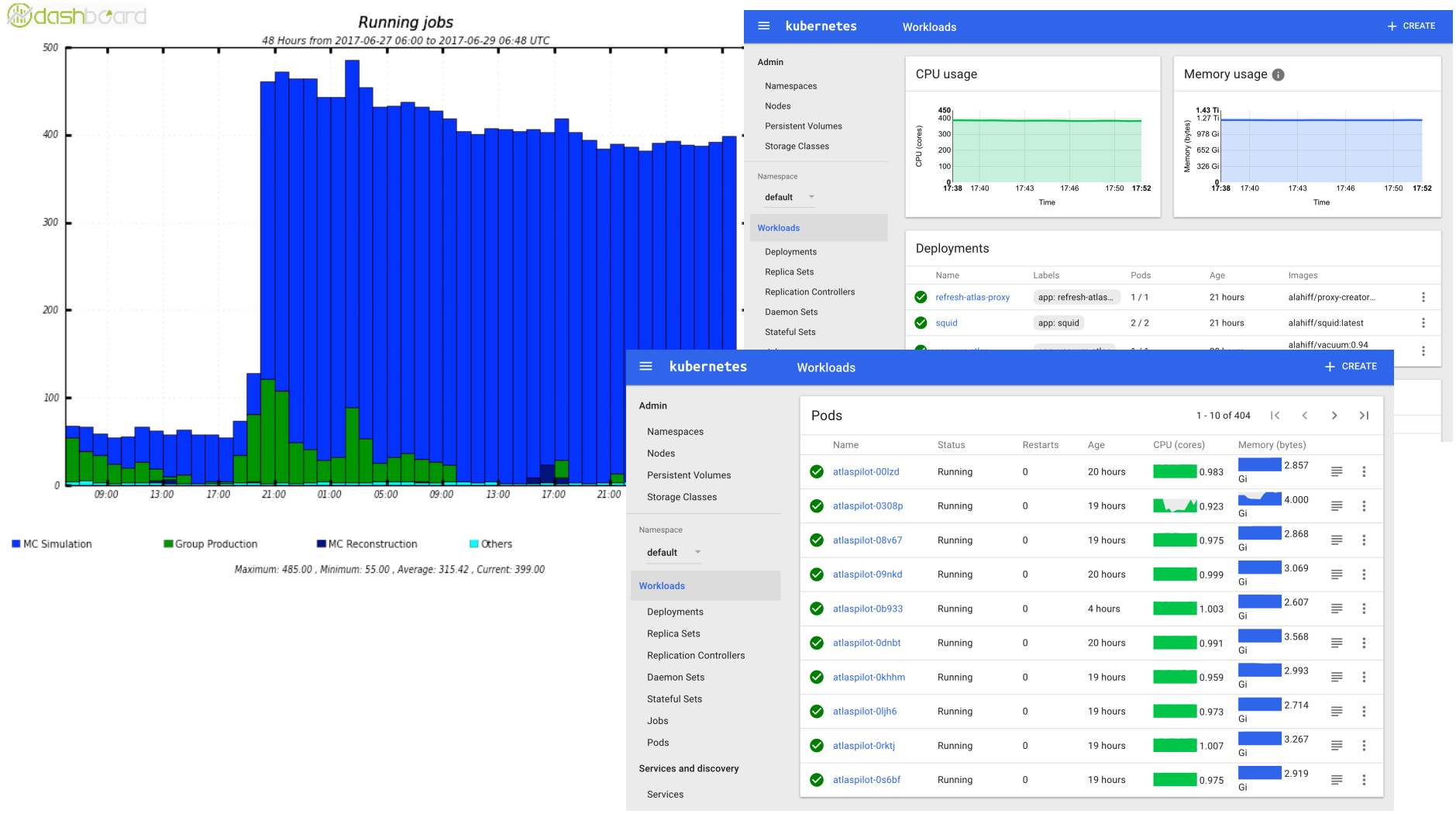
- Created by a single command

```
kubectl create –f atlas.yaml
```

- This creates an elastic, self-healing site for running LHC jobs
  - on a single Kubernetes cluster
  - on multiple clusters around the world (via Kubernetes federations)



Kubernetes as an abstraction layer



Summary

- Containers are being used a lot at RAL in production
  - migrated our HTCondor batch system to run all jobs in Docker containers
  - have started rolling out xrootd gateways to Ceph in containers on worker nodes
- Other efforts at RAL involving containers
  - providing more flexible computing infrastructure
  - making it easier to run services
  - making it easier to use public clouds

Questions?