
# LHCb DIRAC Containers

Andrew McNab, on behalf of the LHCb Collaboration

LHCb, GridPP,
University of Manchester

# Outline

- DIRAC is the workload + data management system used by LHCb
- The aim of the DIRAC containers are to provide black boxes in which DIRAC jobs can run
  - Similar idea to the DIRAC VMs
  - Some experience with LHCb containers for Yandex Skygrid
- We have now created Docker-based containers that adhere to the Vacuum Containers (VC) interface now provided by Vac
  - VC interface defines how hosts provide things like Machine/ Job Features and /cvmfs to containers in a generic way
- It will be possible to create similar containers for other workload management frameworks than DIRAC (eg HTCondor)
- We use CernVM-FS to provide the operating system files as well as the experiment code

# Container components

- To create a container following this pattern you need:
  - An image
  - A contextualisation script (“user_data”)
  - Access to CernVM-FS
  - Extra parameters like min/max lifetime, number of processors, accounting VO name/fqan
- All of this is described in the lhbc.pipe “Vacuum Pipe” JSON file on the LHCb DIRAC webserver, fetched and parsed by the container factory at the site
- Vacuum Pipes can contain definitions of multiple container (and/or VM) versions / flavours
  - Container factories can use them programmatically, reducing the site configuration for a VO to a couple of lines

# Docker images

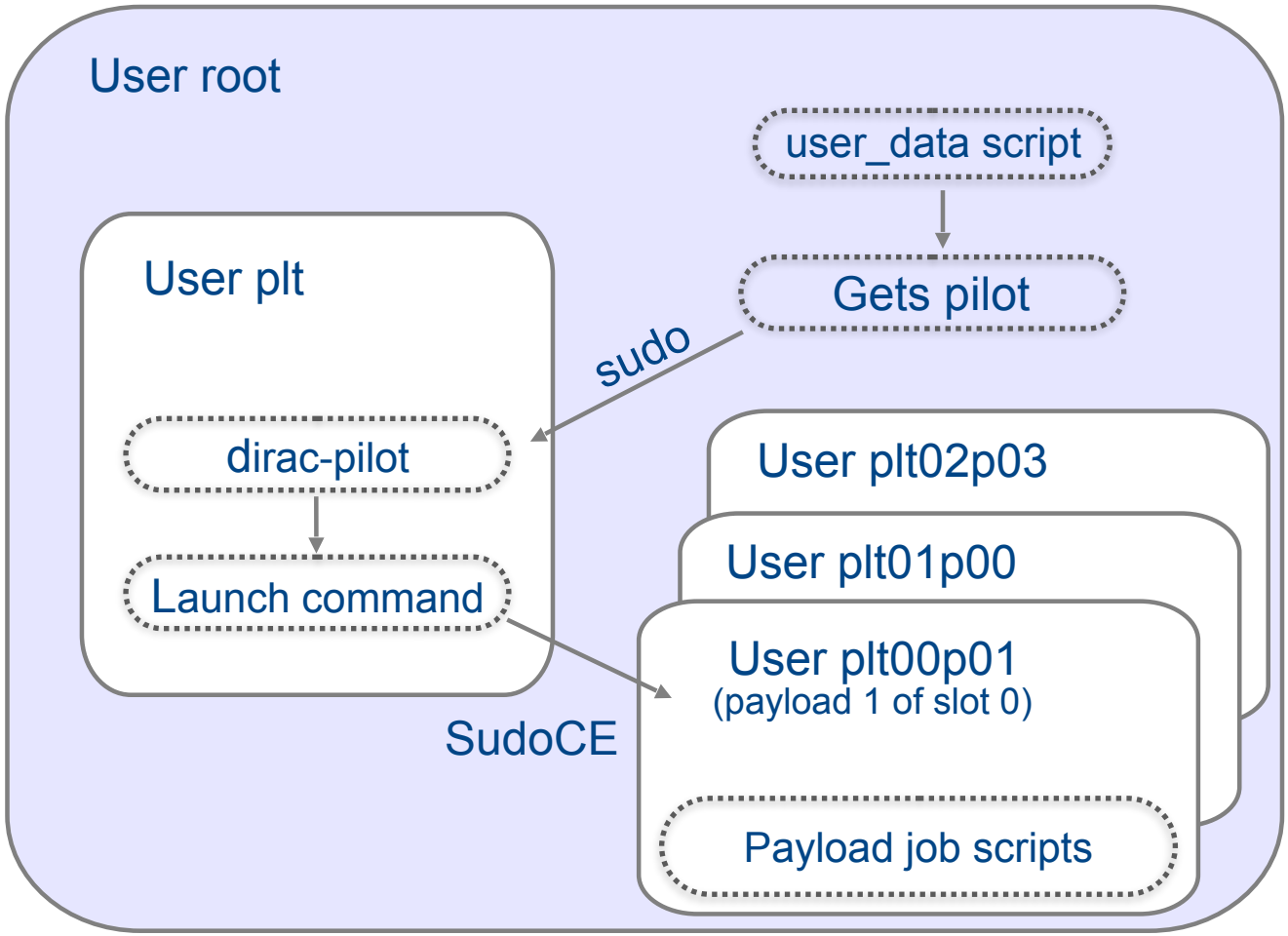
- Generic Docker images we use just bootstrap the containers rather than contain all the code
  - They are a simplified version of Docker images produced by the CernVM project
  - Published in Docker Hub (vacproject/vcbusybox)
- The image consists of:
  - busybox, a self-contained shell+commands binary
  - a script ("/init") which sets up the root filesystem and then runs /user_data
- The root filesystem is mostly populated with symbolic links to /cvmfs/cernvm-prod.cern.ch/cvm3
  - Some files are copied, so they can be modified (eg /etc) or be set setuid to root (eg sudo)

# LHCb DC user_data file

- The container factories fetch user_data_dc_prod template from the LHCb DIRAC webserver
- Some patterns in the file are substituted by the factory
  - eg ##user_data_space## is the space (“CE”) name at the site
- The preprocessed user_data file given to the container with a bind mount
- It consists of a script which converts the generic SL6-like container into an LHCb DIRAC execution node
- Its main tasks are:
  - Set up the X.509 credentials needed to talk to DIRAC services
  - Create the unix accounts within the container to isolate users
  - Download and run the LHCb DIRAC pilot code

# Container structure and user isolation

- Uses unix accounts and sudo to isolate root vs pilot vs payloads
- Requires account creation per payload
- If multiple consecutive payloads per slot, they each get an account
- This is the similar to the LHCb DIRAC VMs



# Volumes bind-mounted in the container

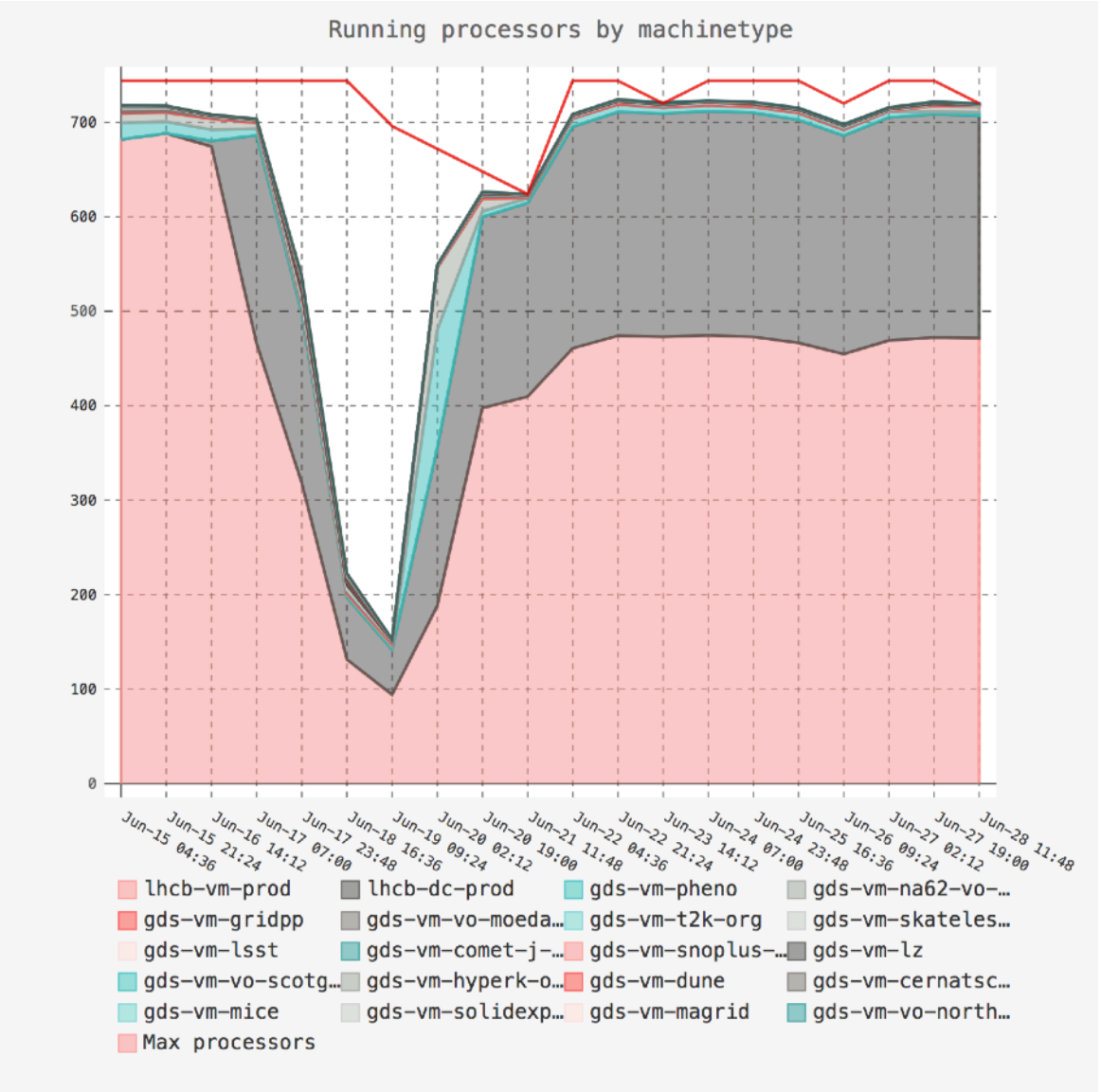
- /cvmfs (ro)
- /user_data (ro)
- /scratch (rw)
  - The host may provide a large, fast volume so the container can avoid using the copy-on-write filesystem as work space
- /etc/machinefeatures (ro)
- /etc/jobfeatures (ro)
  - Machine/Job Features are used by the DIRAC pilot to discover the maximum container lifetime, local UUID, and to receive early stop instructions via shutdowntime
- /etc/joboutputs (rw)
  - Log files and shutdown_message explaining why it finished

# Singularity

- We also did some proof of concept with Singularity containers
  - Same idea but no isolation between payloads and pilot
  - Validated with Monte Carlo production jobs
  - It would be possible to run LHCb production jobs this way if a site only provided the ability to run “black box” Singularity containers
- Singularity is also an attractive alternative to Sudo within the containers
  - This needs tested and supported “SingularityCE” in DIRAC
  - Will also be used for DIRAC jobs running on grid/batch
  - Containers will replace Sudo once this is ready for production use

# LHCb containers running alongside VMs

- The chart shows LHCb Docker containers (lhcb-dc-prod) in dark grey before and after a software update
- LHCb VMs (lhcb-vm-prod) are shown in pink and are running on the same Vac VM/Container factories



# Summary and next steps

- Now added LHCb DIRAC Docker containers in addition to the VMs we've used for several years
- Again use Sudo to achieve excellent isolation between user jobs and pilot code
- Use CernVM-FS to provide the root filesystem as with VMs
  - So again we immediately benefit from operating system security updates etc from CernVM group
- Will provide CentOS7-based containers (cvm4) when needed
- Intend to migrate from Sudo to Singularity for isolation when tested and supported in DIRAC
- This pattern can be used by other workload frameworks
- And could be run as black boxes by other Docker container factories (eg Kubernetes)