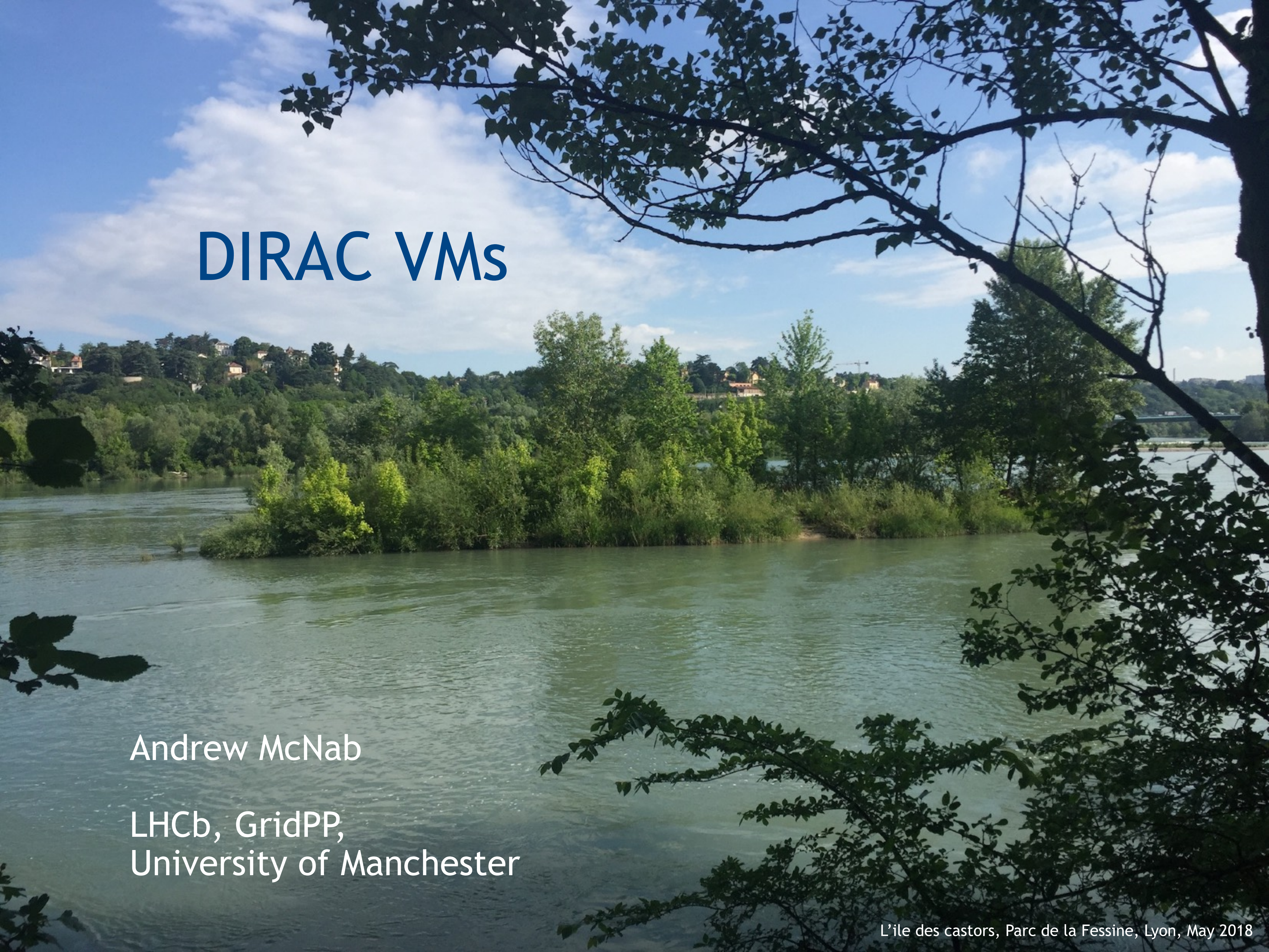


DIRAC VMs

Andrew McNab

LHCb, GridPP,
University of Manchester

L'ile des castors, Parc de la Fessine, Lyon, May 2018

DIRAC VMs

- Cover the Vacuum VMs defined in the DIRAC Pilot repo
  - These are NOT the VMDIRAC VMs
- Aims
- VM components
- Bootstrapping process
- Payload vs pilot Isolation
- Universality of VMs: eg Google Cloud
- Customization hooks
- Next steps

VM aims

- Someone needs to provide VMs to be able to run on OpenStack, Google Cloud, Amazon EC2 etc
  - Could be the site: lxbatch at CERN is in VMs on OpenStack provided by the site and connected to HTCondor CE.
- But our VM aims go beyond this
- Provide generic “black box” VMs that sites can run
  - By themselves or mixed in with VMs from other experiments
  - This “vacuum model” is supported by Vac and Vcycle on OpenStack etc
- Provide internally uniform VMs across sites and platforms
  - eg same VMs run on OpenStack, Google Cloud, and Vac
  - No more chasing up one missing RPM at site XYZ!

VM components

- To create a VM you need:
  - A boot image (CernVM in this case)
  - A contextualisation user_data file
  - Extra parameters like min/max lifetime, number of processors, accounting VO name/fqan
- These can be published on web page, or now in a Vacuum Pipe JSON file
  - Vacuum Pipes can contain definitions of multiple VM (or Container) versions / flavours: pro, dev, squid-cache, … ?
  - Vac and Vcycle use them programmatically which reduces the site configuration for a VO to a couple of lines

VM universality across clouds

- CernVM supported across OpenStack (= Vac), Google Cloud, Amazon EC2, Microsoft Azure, …
- Cloud Init framework used by CernVM to fetch and interpret user_data file across the above platforms
- The user_data file in the Pilot repo is
  - used in production on OpenStack at multiple sites
  - (and on Vac which has the same API)
  - and has been demonstrated with a production run on Google Cloud **without modification**
- Should also work on Amazon EC2 as the user_data file mechanism is almost identical to OpenStack
- Should be extendable to Azure too

“wget” bootstrapping

- Pilot 3.0 does bootstrapping by

```
wget –recursive https://some.wh.ere/some/directory/
./dirac-pilot.py …
```

- You need an HTTPS web server where you can put the pilot files
  - The cert should be from an IGTF CA (a grid cert)
- The files are in https://github.com/DIRACGrid/Pilot/tree/master/Pilot
  - This is used by LHCb in production and still contains some LHCb specific code which should be ignored or fail silently for other VOs
- The key to the whole thing is the file user_data_vm which does the wget command and much else

user_data_vm file

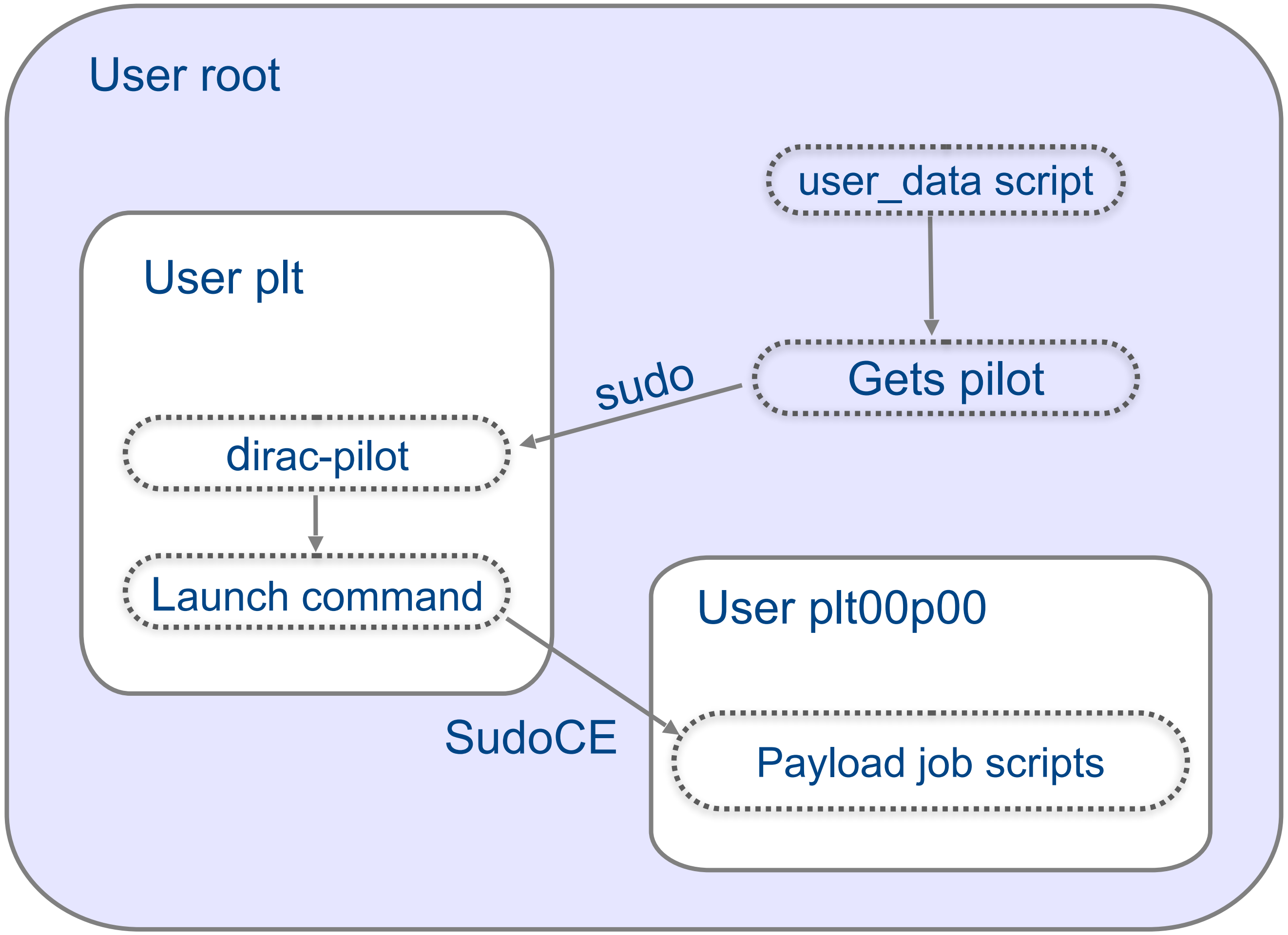
- This can be preprocessed before being passed to the VMs
- Patterns like ##user_data_***## are replaced with defined values or removed
- If you're using Vac or Vcycle to start the VMs, you don't need to do anything with the file
- If you were starting VMs by hand, then you would replace
  - ##user_data_space## with the virtual CE name of your site
  - ##user_data_uuid## with a unique string
  - ##user_data_file_hostkey## and ##user_data_file_hostcert# with the PEM-encoded cert and key
  - And remove all other ##user_data_***## patterns

Bootstrapping configuration

- Also have a configuration bootstrapping issue in creating DIRAC VMs
  - Generic pilot code needs to know the CS URL etc of this XyzDIRAC instance, DIRAC version, …
  - Needs to know the site it's running at (eg may need to run different pilot commands at different types of site)
- For this reason, Pilot 3.0 includes a file pilot.json in the directory wget fetches
- This is minimal dump of the CS with enough info to configure the pilot
  - DIRAC version, commands to run, Setup, and all the CE to Site mappings

Pilot 3.0 VMs structure and isolation

- Uses unix accounts and sudo to isolate root vs pilot vs payloads
- SudoComputingElement does the sudo for the payloads
- Requires account creation per payload
  - Easy within VMs but doesn't generalise to batch cases



Job isolation and Singularity

- WLCG has working group on Traceability and Isolation
- Has settled on Singularity lightweight container framework
  - Which has a lot of traction in HPC and growing support from CMS and ATLAS
- Singularity does not require a daemon (cf Docker)
- In sufficiently late Linux kernels, you do not need any setuid wrapper
- Now we have support in DIRAC (the new SingularityCE) we will be able to have a single isolation framework for pilots in VMs and in batch
  - ie replace sudo with singularity inside the VMs

Per DIRAC instance customisation

- By design, the pilot directory fetched by wget can have a mixture of standard and custom files
  - In particular you can add your own files containing pilot commands for your XyzDIRAC flavour
  - And enable them in the pilot.json, perhaps only for VMs
  - These run as the pilot user inside the VMs
- In the pilot directory, can also have include_vm_*.sh scripts
  - These are sourced by the user_data script in alphanumeric order
  - This feature is currently only in the GridPP DIRAC Service VMs, but will be merged into the main Pilot repo
  - These run as root inside the VMs

Advert: Lightweight Sites WG

- The WLCG GDB has created a Lightweight Sites Working Group
  - **Vacuum VMs of the form described here are a key component of the WG**
- Aimed at sites and deployment
  - Will identify areas of development needed and liaise will appropriate working groups, task forces, and developers
- A point of contact **for new sites wanting simple ways of providing resources to experiments**
- https://twiki.cern.ch/twiki/bin/view/LCG/LightweightSites

Summary and next steps

- Running in production by LHCb and GridPP DIRAC Service for several years
  - Vac and Vcycle/OpenStack
- Successful LHCb production run on Vcycle/GoogleCloud
- Excellent pilot vs payload isolation provided by SudoCE
  - But SudoCE is only used in VMs

- Replace SudoCE with more general SingularityCE inside VMs
- Create Docker containers with pilot vs payload isolation
  - Non-isolating Docker containers for DIRAC already exist