GridPP48 DIRAC Update

Daniela Bauer & Simon Fayer

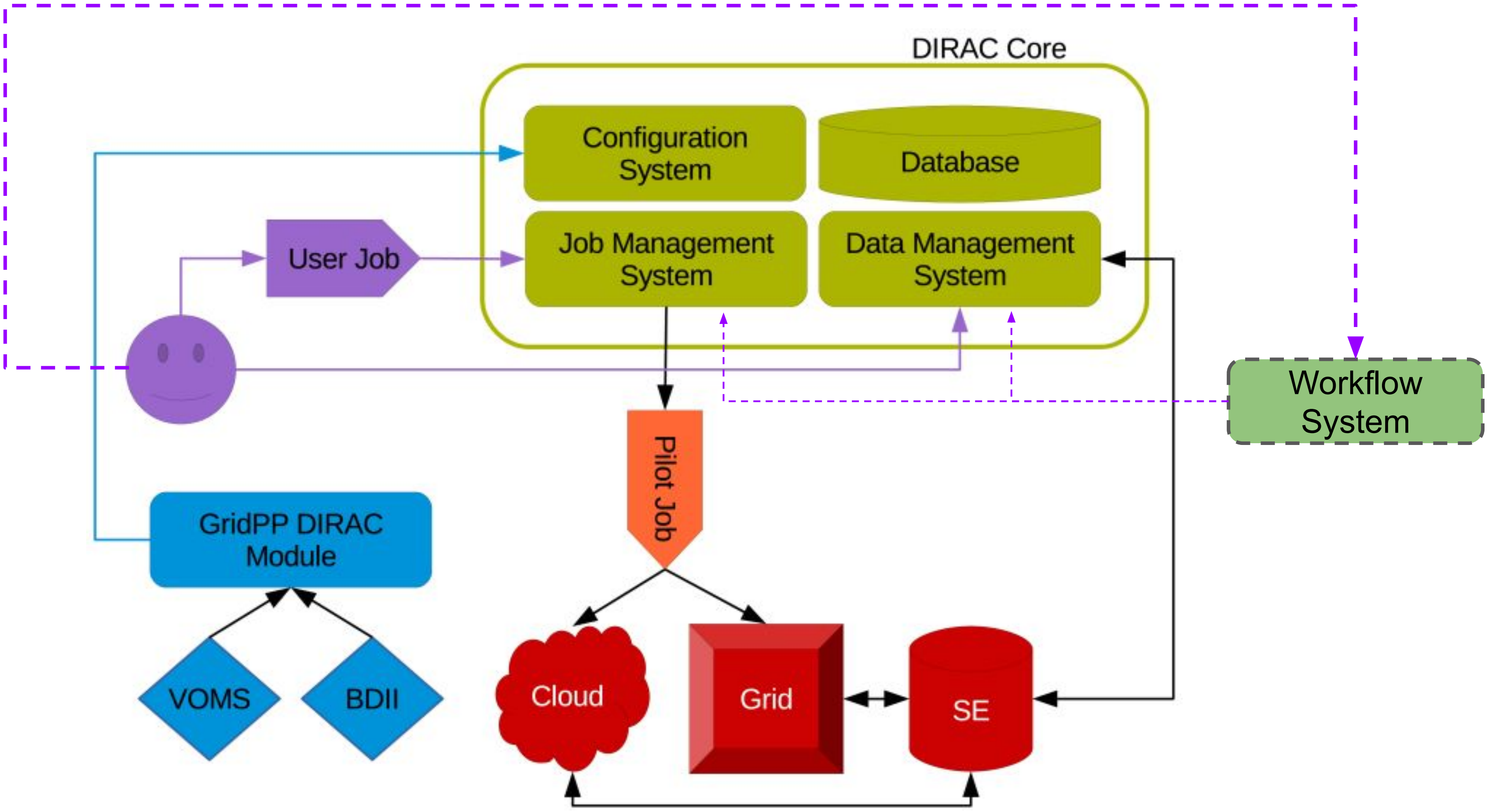
Imperial College London

GridPP UK Computing for Particle Physics

What is DIRAC ?

- DIRAC is software that comprises of:
  - Workload Management System (“Global batch system”)
  - File Catalog/Data Management
  - Workflow Management System
  - Documentation: https://dirac.readthedocs.io/en/latest/
- Provides a standardized user interface to multiple compute and storage resources
- Written in Python (for Linux)
  - OpenSource: https://github.com/DIRACGrid/DIRAC
- Originally developed by LHCb the project is administered by the DIRAC consortium (UK member: Imperial College)
- Used by a number of communities to manage the various aspects of their data processing:
  - Experiment specific: e.g. LHCb, Belle2, ILC, Cherenkov Telescope Array, NICA (JINR), BES (Beijing), biomed
  - **Multi-Community:** e.g GridPP, France-Grilles, EGI

The GridPP DIRAC instance

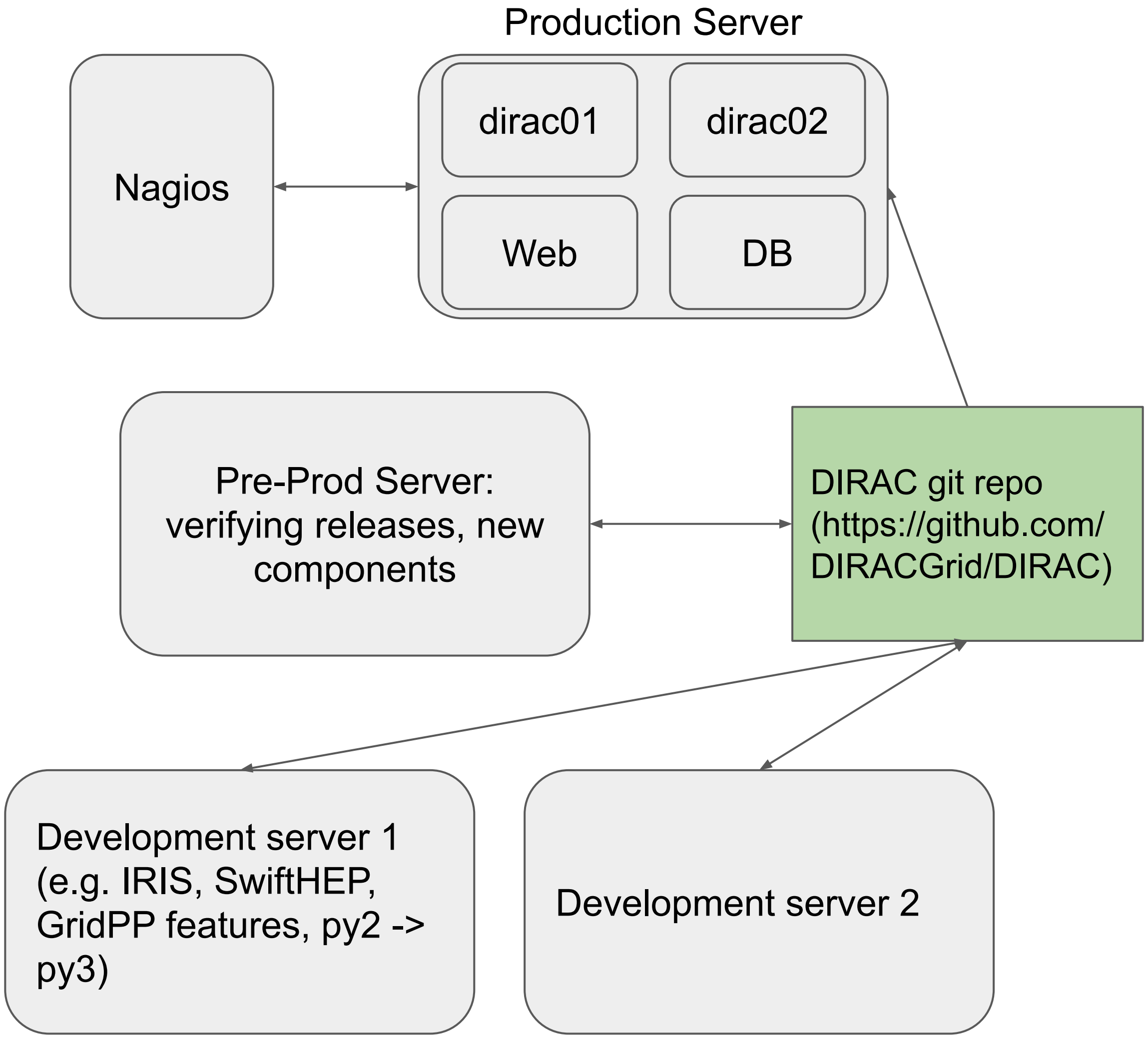


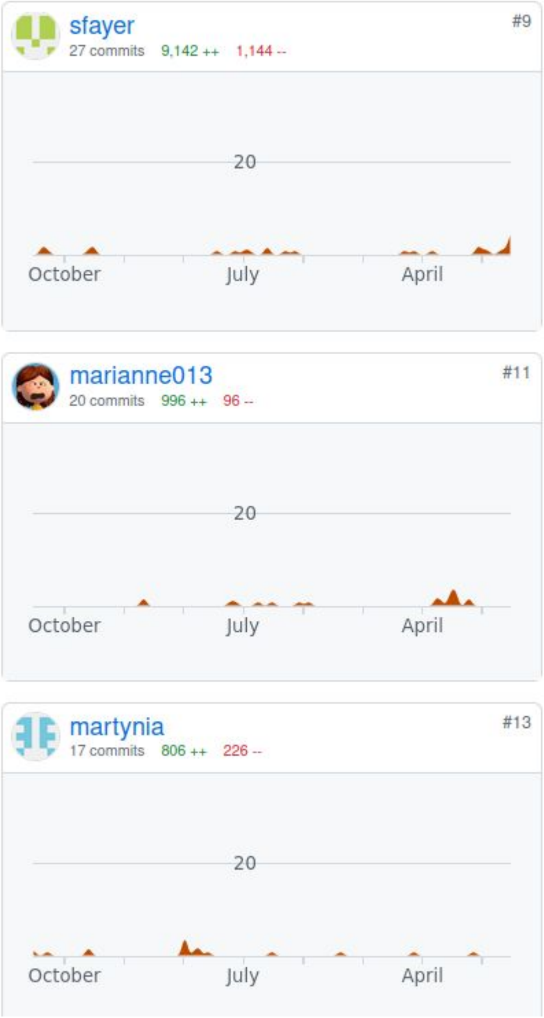
Imperial College London

DIRAC THE INTERWARE

GridPP UK Computing for Particle Physics

The GridPP DIRAC instance: How to (mostly) not break stuff


Production Server
Nagios
dirac01
dirac02
Web
DB
Pre-Prod Server: verifying releases, new components
DIRAC git repo (https://github.com/DIRACGrid/DIRAC)
Development server 1 (e.g. IRIS, SwiftHEP, GridPP features, py2 -> py3)
Development server 2

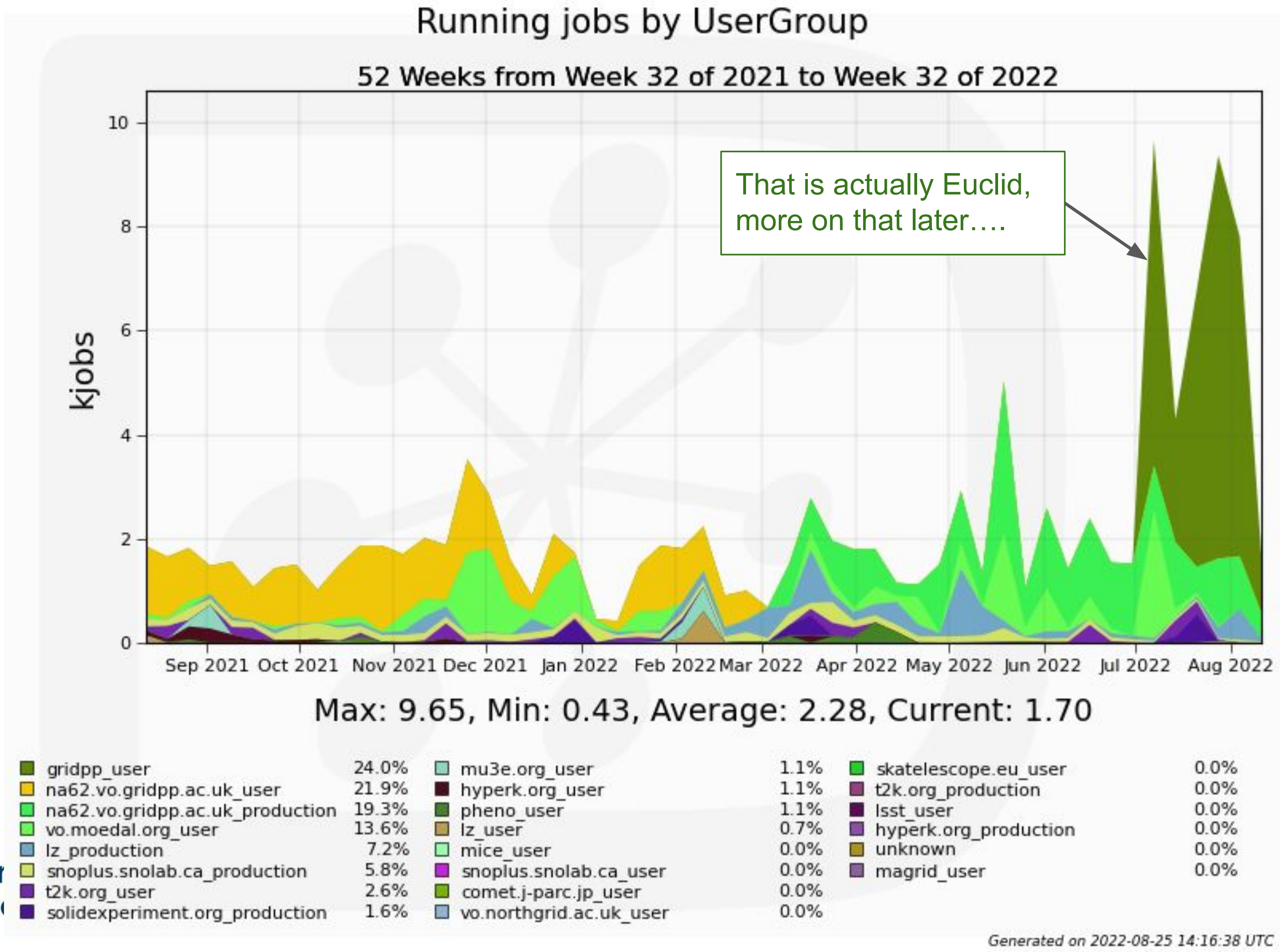


sfayer #9
27 commits 9,142 ++ 1,144 --

marianne013 #11
20 commits 996 ++ 96 --

martynia #13
17 commits 806 ++ 226 --

DIRAC Usage



That is actually Euclid, more on that later….

Case studies: LZ

- DIRAC forms a large part of their processing work chain in the UK:
    - Used for grid and cloud computing
    - 12.5 mio files registered in the DIRAC File Catalog
    - LZ built a experiment specific front end using the DIRAC python API for bulk processing (main developer: A. Richards)
- A lot of DIRAC Cloud developments were driven by this project.

Please see their talk on Friday:
https://indico.cern.ch/event/1169160/contributions/4990486/

Case studies: T2K/Hyper-K

- DIRAC forms the backend to the T2K/Hyper-K Data Management System:
  - T2K: 7.5M files
  - Hyper-K 4.0M files
- Uses the DIRAC Request Management System with an FTS3 backend for data movement/replication
- This requires operational support from DIRAC, especially when new storage elements are commissioned

Imperial College London

Hyper-K

T2K

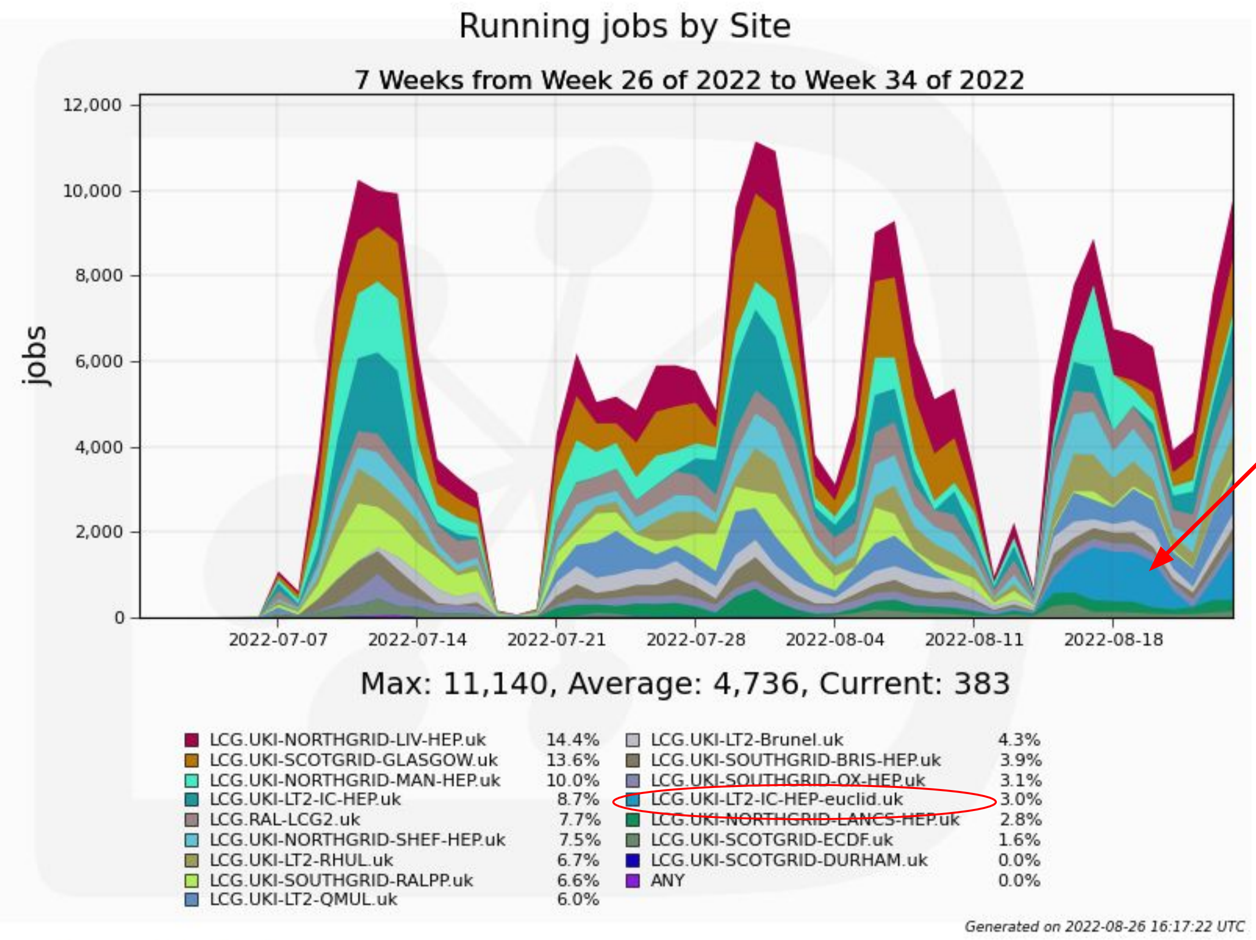
GridPP
UK Computing for Particle Physics

Case Study: Euclid

- Euclid UK computing is primarily supported by IRIS
- Recently a Euclid user had a production task to be run at scale that was more suited to grid-type resources
- Used the gridpp VO as an incubator VO
- Worked with the user to prepare their project for grid deployment:
  - Code deployment: replicated code bundles & conda
  - Operational assistance: Debugging failures, mostly fixed after discovering an unexpected high memory usage
  - Grid storage used as scratch space for results

Euclid consortium

Case Study: Euclid



- Quick ramp up
- All sites contribute resources
- On Euclid's request we made their IRIS cloud allocation at Imperial available through DIRAC
- Migrating to new VO after successful startup: eucliduk.net

Engagement with non-GridPP projects

- IRIS:
  - User Support:
    - Current: LZ, Euclid
  - Digital Assets (DIRAC enhancements):
    - Standardizing Cloud Interfaces using libcloud
      - builds on initial asset “direct cloud submission in multi-VO DIRAC”
    - Basic multi-VO Rucio Integration
    - Multi-VO File catalogue metadata
    - Multi-VO Resource Status system: Automatic disabling of ‘bad’ sites

As these features make their way into production, there is mutual profit for both GridPP and IRIS.

Engagement with non-GridPP projects

- SwiftHEP Development - Pilot log system (J.Martyniak):
  - Summary: If you want a reliable service, you need to be able to debug problems.
    - Pilot jobs:
      - Check the worker-node environment
      - Can stage required input/output files
      - Start and supervise the user job (record memory usage, efficiency, etc.)
    - Pilot job logs are stored in an technology dependent way at the execution resources
    - Retention policies vary by technology and site:
      - Some logs only kept while job running!
      - Others kept 3 days - 1 month depending on configuration.
      - Transient (cloud) resources may not have space suitable for archiving these logs.
    - Log can be completely lost in cases where job crashes (i.e. exceeding batch limits).
    - Retaining pilot job logs in a reliable, resource independent manner was identified as a high priority item by all DIRAC supported communities
    - We are hoping to release a first version of this within a month

Engagement with non-GridPP projects

- SwiftHEP - Harnessing DIRAC's inbuilt functionality
  - DIRAC has an inbuilt workflow component that can be used to chain multiple steps in an analysis workflow together
  - Analysis WorkPackage would like to use DIRAC to:
    - Store analysis output and attach metadata to it for further processing
    - Distinguish temporary and long lived files
    - Automate certain steps in a processing cycle
    - Delete temporary files at the end of a processing cycle automatically
  - Similar workflows are used by e.g. ILC and CTA on DIRAC
  - The challenge is to implement this in a multi-VO environment
  - System currently deployed on pre-prod server for testing

Summary

- The GridPP DIRAC instance is currently the entry point to grid and cloud resources for ~ 10 non-LHC communities
- The project is in active development
- The GridPP DIRAC instance is regularly updated/extended in a well defined process
- GridPP also provides the necessary user support that enables non-LHC communities to use their facilities
- The GridPP DIRAC team engages with other UK projects (IRIS, SwiftHEP) to ensure DIRAC continues to meet their requirements which in turn benefits GridPP communities