DIRAC Interware
Project Status

DIRAC
THE INTERWARE

A.Tsaregorodtsev,
Aix Marseille Univ, CNRS/IN2P3, CPPM,
EGI Conference, 21 October 2021

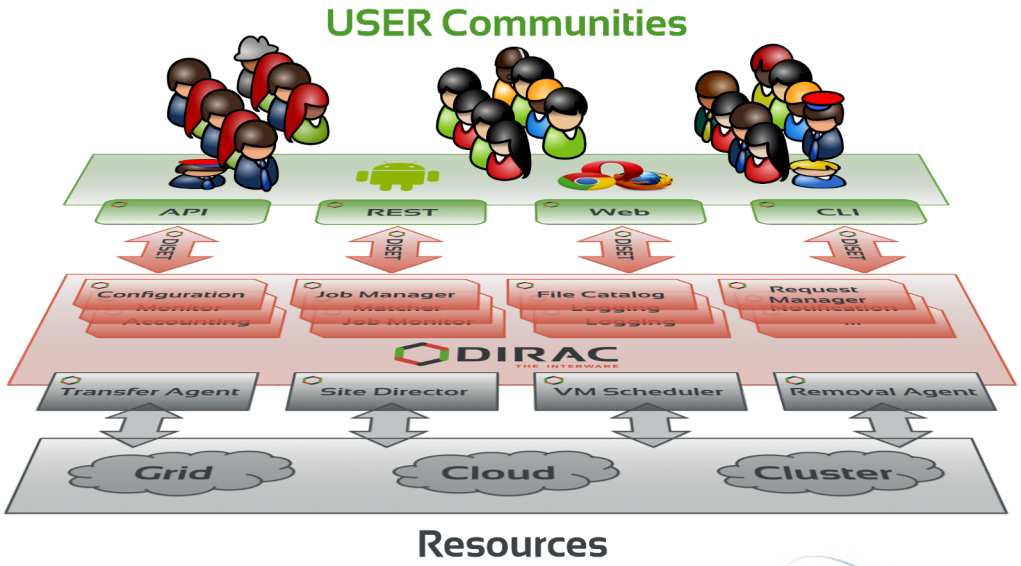
DIRAC
THE INTERWARE

Outline

- DIRAC overview
- Managing jobs and workflows
- Managing data
- Development focus
- Conclusions

Interware

- A software framework for distributed computing
- A **complete** solution to one (or more) user community
- Builds a layer between users and resources
- A *framework* shared by multiple experiments, both inside HEP, astronomy, and life sciences

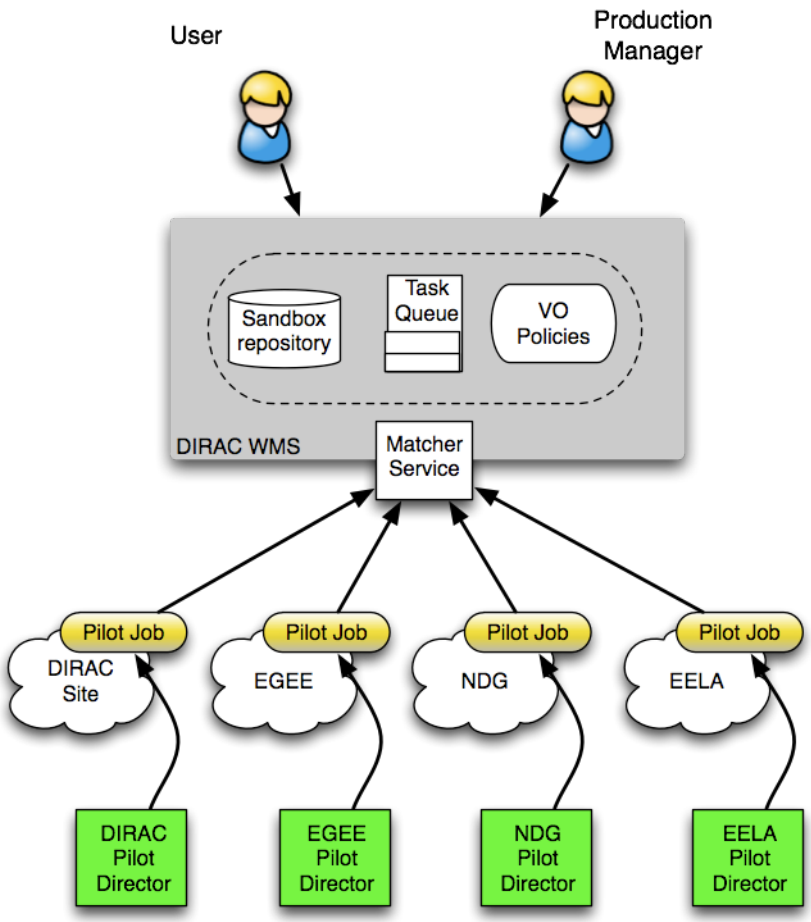


DIRAC
THE INTERWARE

Managing user jobs

WMS: Pilots are federators

- Pilot jobs are submitted to computing resources by specialized Pilot Directors
- Pilots retrieve user jobs from the central Task Queue and steer their execution on the worker nodes including final data uploading
- Pilot based WMS advantages:
  - increases efficiency of the user job execution
  - allows to apply efficiently community policies at the Task Queue level
  - allows to integrate heterogeneous computing resources



Pilot3 Package

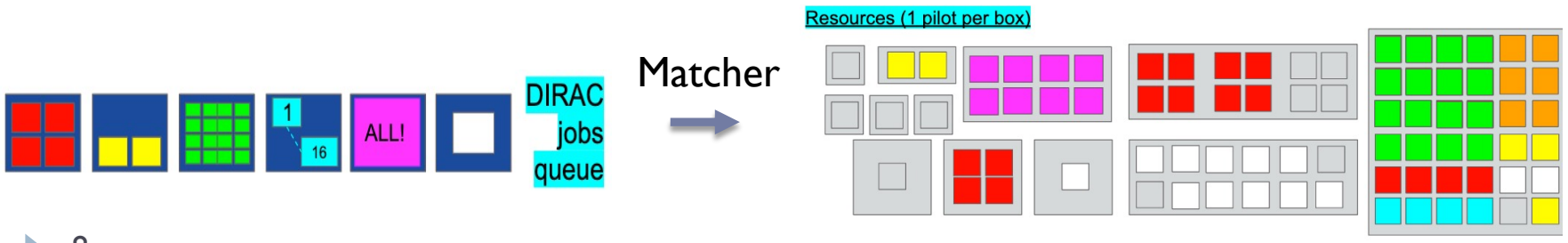
- Pilot3 is a separate DIRAC subproject to encapsulate code used in pilot jobs
  - Pilots are running in a DIRAC-free environment
  - Steer user jobs execution on a worker node
- Pilot3 software is bundled and stored on a web server by running a special agent.
  - No special configuration for the web server is required for file uploads
- Pilot3 commands can be modified/added in the community specific DIRAC extensions
  - Adapt to community specific workflows

User jobs steered by pilots

- Pilots are launching user jobs on WN's to "inner" Computing Elements
- **InProcess** CE – execution in the same process as JobAgent
- **Sudo** CE – execution in a spawned process with a different user ID
  - Used on VMs to isolate pilot environment from the user job
- **Singularity** CE
  - The user job is executed inside a Singularity container
  - Isolation of the pilot environment
  - Possibility to update the environment for user job execution, e.g. reinstall DIRAC client with different options.
- **Pool** CE

Fat multi-core nodes

- Pilots can exploit multi-core nodes use **PoolCE** "inner" Computing Element
  - On-WN batch system, e.g. on VMs or on HPC nodes
  - Flexible strategy with prioritized job requests to the Matcher, e.g.:
    - First, ask for jobs requiring WholeNode tag
    - If none, ask for jobs requesting as many cores as available
    - If none, ask for jobs with MultiProcessor requirement
    - If none, ask for single-core jobs
  - The goal is to fill the nodes with payloads fully exploiting there multi-core capacity
    - Typically for cloud VM or HPC nodes



User jobs

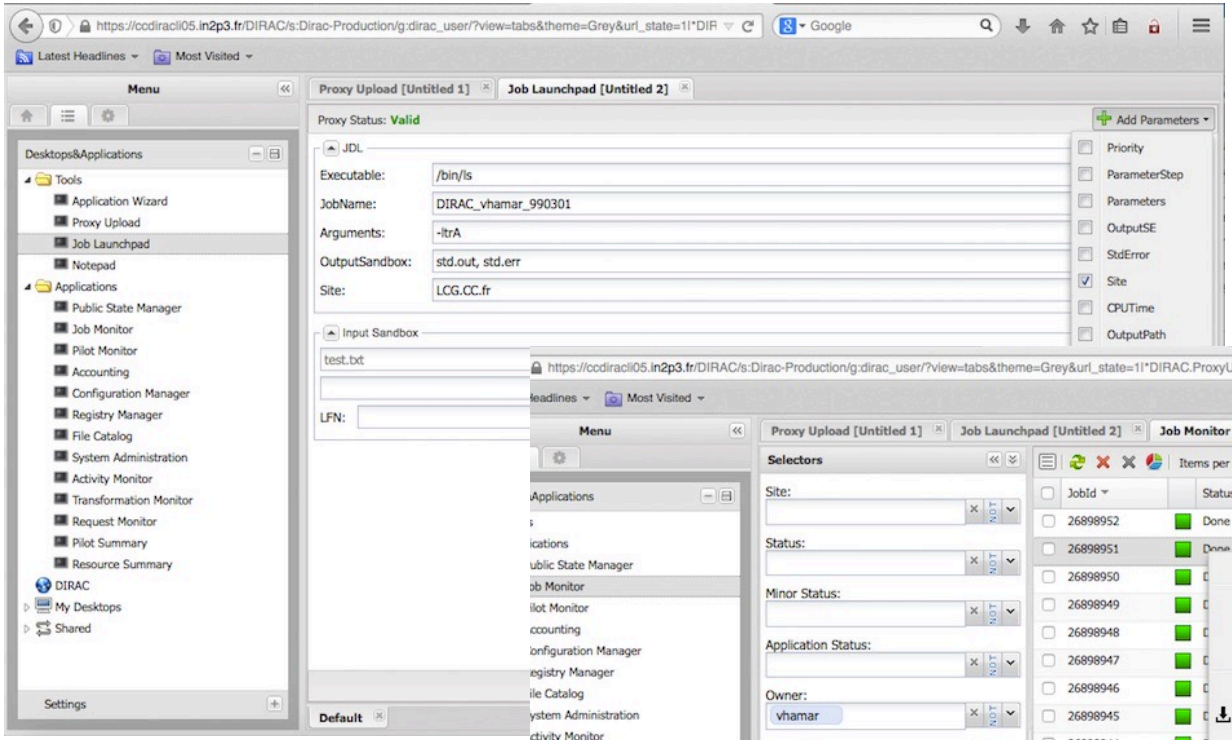
- Users are managing jobs using various tools
  - Command line (batch system like interface):

```
bash-4.2# dsub /bin/echo "Hello world"
53917277
bash-4.2# dstat
JobID     Owner    JobName  OwnerGroup  JobGroup  Site           Status   MinorStatus  SubmissionTime
=============================================================================================================
53917277  atsareg  Unknown  wenmr_user  NoGroup   EGI.NIKHEF.nl  Running  Application  2020-10-22 19:06:24

bash-4.2# doutput 53917277
bash-4.2# ls -l 53917277
total 4
-rw-r--r-- 1 71139 2062 12 Oct 22 19:06 std.out
```

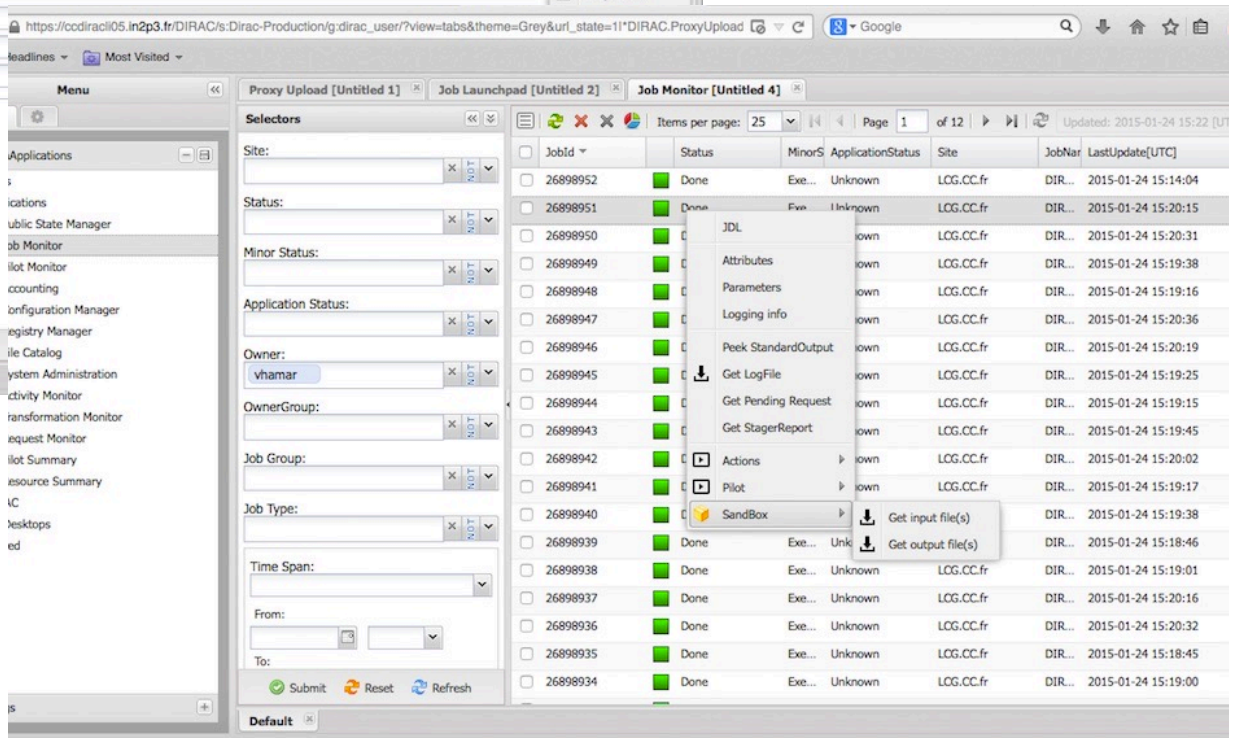
  - Python API
    - Starting from v7r2 Pyhton3 client API is supported

```
from DIRAC.Interfaces.API.Job import Job
from DIRAC.Interfaces.API.Dirac import Dirac

dirac = Dirac()
j = Job()

j.setCPUTime(500)
j.setExecutable('/bin/echo hello')
j.setExecutable('/bin/hostname')
j.setExecutable('/bin/echo hello again')
j.setName('API')

dirac.submitJob(j)
```

Web Interface



Job Launchpad

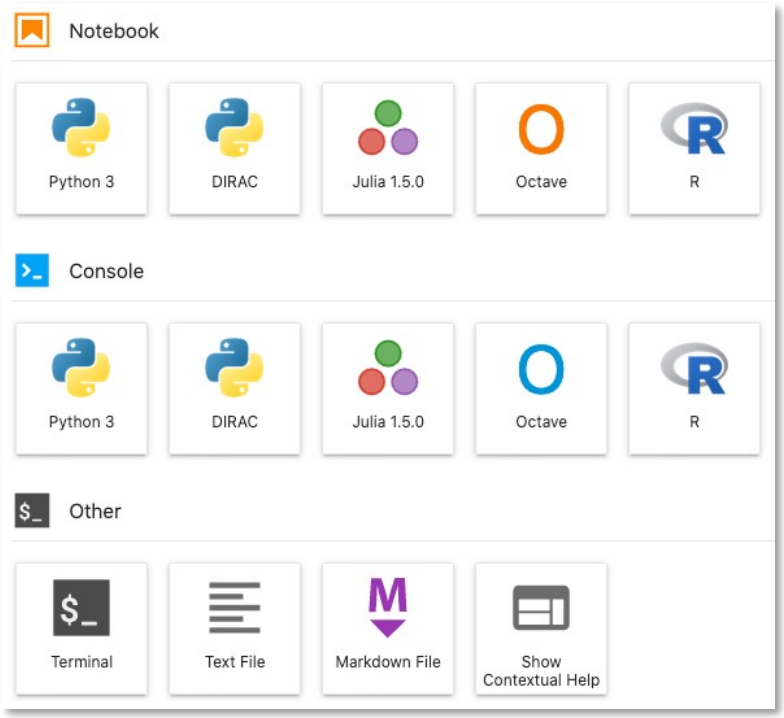


Job Monitoring

Other WMS interfaces

- REST API
  - A language neutral interface for job manipulation
  - To be refurbished together with the introduction of the HTTPS based client/server protocol (*see below*)

- Jupyter Notebook interface
  - In a prototype phase
  - DIRAC API enabled iPython shell
  - Terminal with DIRAC command line interface
  - Managing user credentials is being sorted out
    - Functional for users having grid certificates and registered in the Check-In SSO service

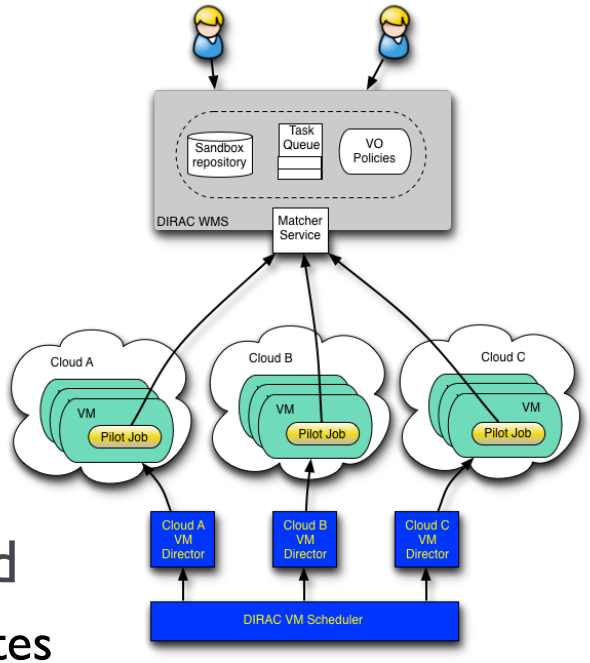


DIRAC
THE INTERWARE

Managing user computing resources

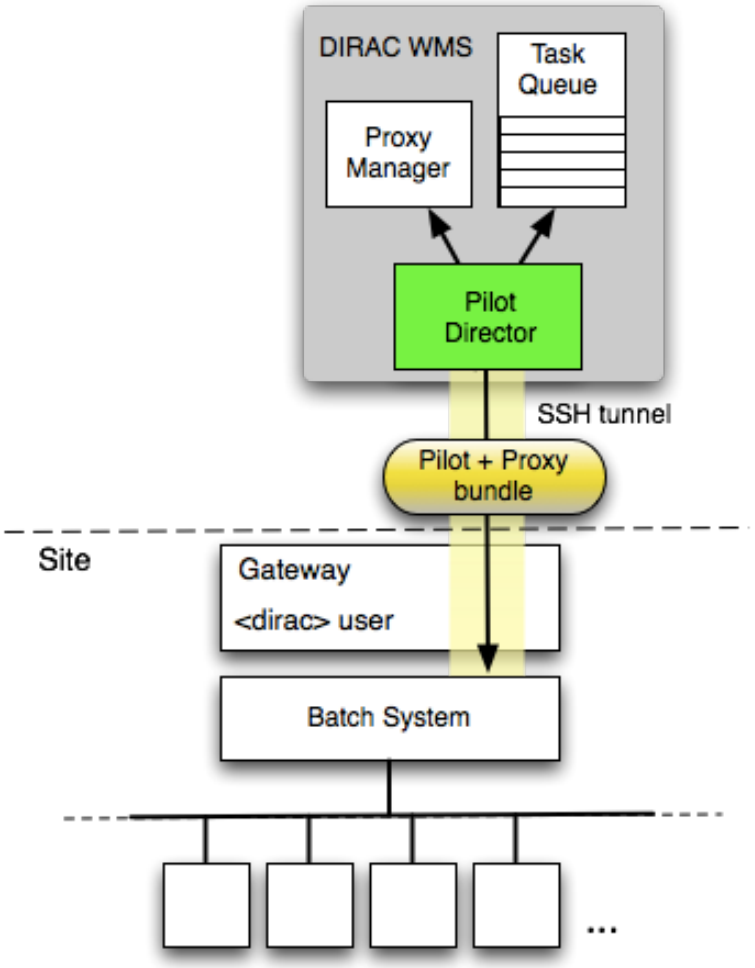
Computing Grids and Clouds

- DIRAC was initially developed with the focus on accessing conventional Grid computing resources
  - WLCG grid resources for the LHCb Collaboration
- Grid infrastructures
  - E.g. EGI, WLCG, OSG
  - CREAM, HTCondorCE, ARC
- Cloud infrastructures
  - EGI Federated Cloud, France-Grilles cloud
    - Access with login/password or x509 certificates
    - Access with tokens – work in progress



Standalone computing clusters

- Users can connect their own computing resources
  - Not making part of any grid infrastructure
    - E.g. LHCb Online farm
- The user site can be:
  - a single computer or several computers without any batch system
  - a computing cluster with a batch system
    - LSF, BQS, SGE, PBS/Torque, Condor
      - Commodity computer farms
    - SLURM+PoolCE
      - HPC centers

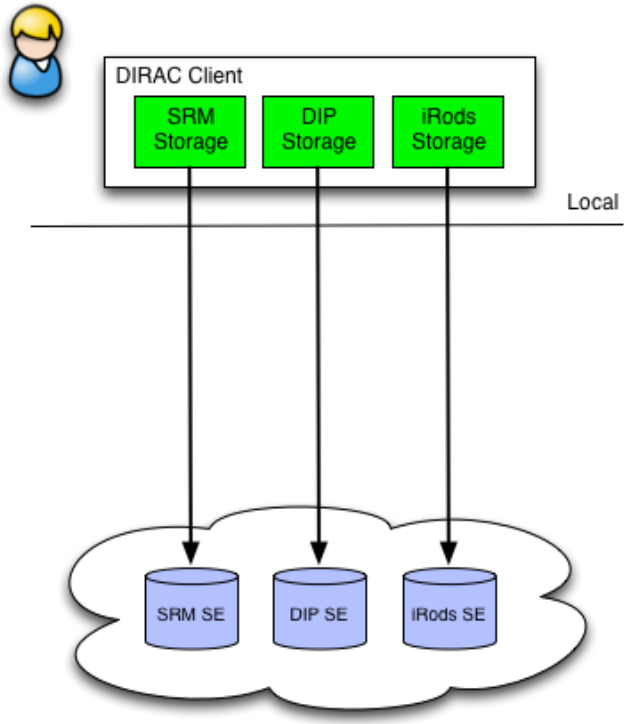


DIRAC
THE INTERWARE

Managing user data

Storage plugins

- Storage element abstraction with a client implementation for each access protocol
  - DIPS – DIRAC data transfer protocol
  - FTP, HTTP, WebDAV
  - SRM, XROOTD, RFIO, DCAP, etc
    - Using gfal2 library developed at CERN
  - S3, Swift: cloud specific data access protocols
- Each SE is seen by users as a logical entity
  - With some specific operational properties
    - Archive, limited access, etc
  - SE's can be configured with multiple protocols
- New data access technologies require creating new specific plug-ins

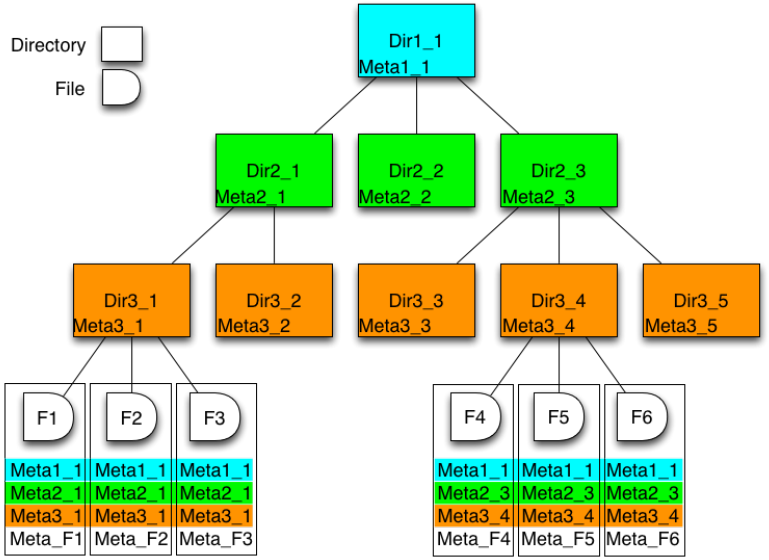


File Catalog Service

- File Catalog is a service to keep track of all the physical file replicas in all the SE's
  - Stores also file properties:
    - Size, creation/modification time stamps, ownership, checksums
    - User ACLs
- DIRAC relies on a *central* File Catalog
  - Defines a single logical name space for all the managed data
  - Organizes files hierarchically like in common file systems
- VO's can ask for dedicated File Catalog services
  - No interference with other users, catalog is chosen based on the user identity
  - Customized behavior
    - Example: Eiscat 3D File Catalog in the EGI DIRAC Service
      - 139M files
      - Custom access policies

File Catalog: Metadata

- DFC is Replica and Metadata Catalog
  - User defined metadata
  - The same hierarchy for metadata as for the logical name space
    - Metadata associated with files and directories
    - Allow for efficient searches



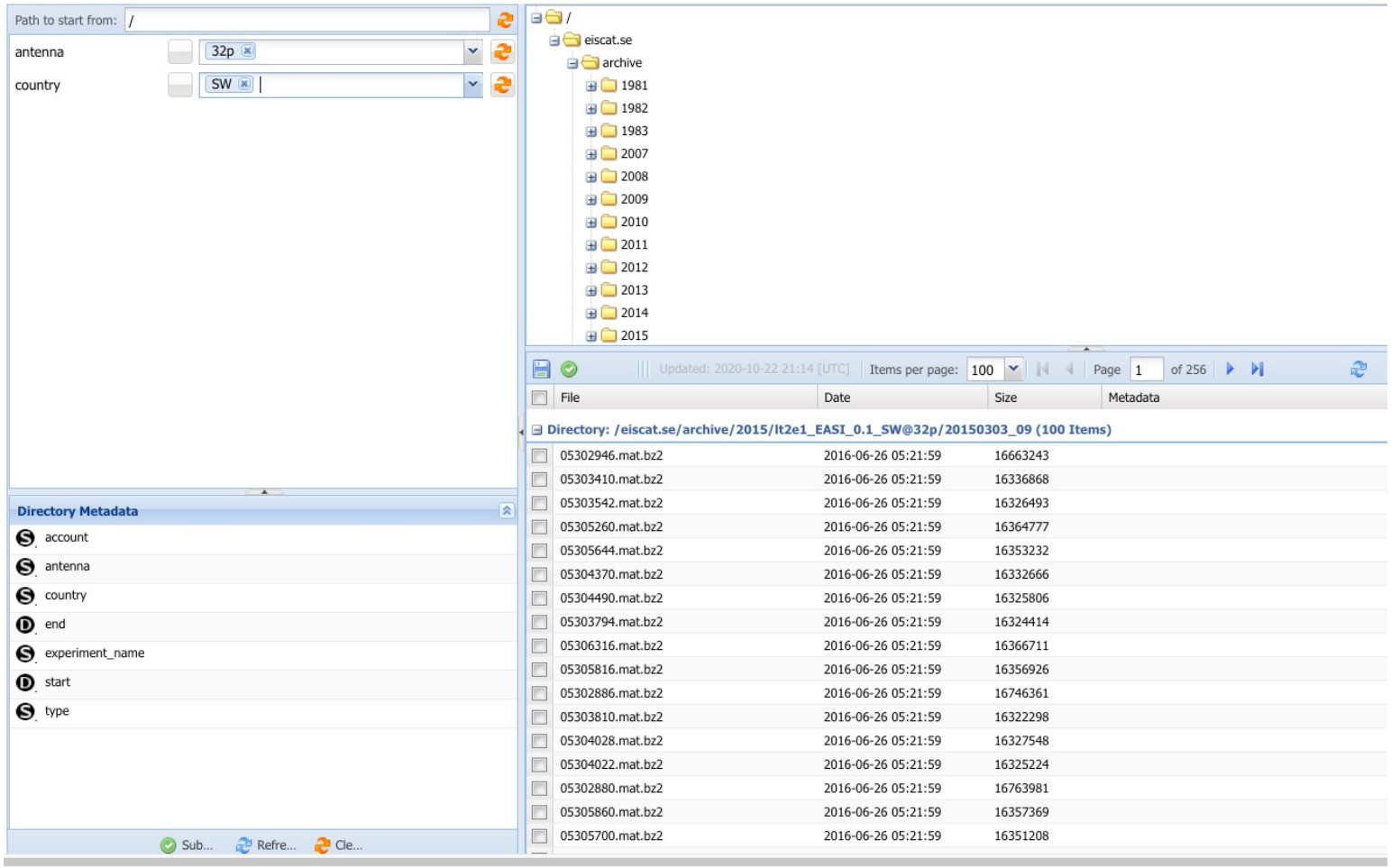
- Efficient Storage Usage reports
  - Suitable for user quota management
- Stored ancestor/successor file relations
  - Simple provenance catalog

DM user interfaces

- COMDIRAC
  - Representing the logical DIRAC file namespace as a parallel shell
    - **dls, dcd, dpwd, dfind, ddu,** etc commands
  - Commands for file upload/download/replication
    - **dput, dget, drepl**

```
bash-4.2# dput test.jdl /enmr.eu/user/a/atsareg/test/test.jdl
bash-4.2# dls -L /enmr.eu/user/a/atsareg/test/test.jdl
-rwxrwxr-x 1 atsareg wenmr_user 256 2020-10-22 22:33:12 test.jdl
   CYFRONET-USER   dips://dirac-dms.egi.eu:9148/DataManagement/StorageElement/enmr.eu/user/a/atsareg/test/test.jdl
bash-4.2# rm test.jdl
bash-4.2# dget /enmr.eu/user/a/atsareg/test/test.jdl
bash-4.2# ls test.jdl
test.jdl
bash-4.2# drm /enmr.eu/user/a/atsareg/test/test.jdl

1 object(s) removed in total
```

File Catalog Web interface



DIRAC-Rucio bridge

- Multiple communities are looking for a combined solution with Rucio for the DMS tasks and DIRAC for the WMS tasks
  - Example, ESCAPE Collaborations
- The RucioFileCatalog encapsulates Rucio service access from within the DIRAC Framework
  - The work done by C.Serfon (Belle II)
- Belle II used LFC and DIRAC data operation tools
  - Decided to move to Rucio for the DM tasks
- Initially developed as part of the BelleDIRAC extension
  - Available in the core DIRAC starting from v7r2

DIRAC-Rucio bridge(2)

- Several limitations examples
  - On the contrary to LFC or DFC, Rucio is not inherently a hierarchical namespace
  - To mimic LFC hierarchy, following mapping is used (LFC → Rucio) :
    - File → File
    - Directory containing files → Dataset
    - Directory containing directories → Container
  - Containers in Rucio cannot contain mixtures of files and directories
  - No correspondence of Rucio's scope concept in DIRAC
- The limitations impose constraints on structuring the namespace for a given community
  - Not critical, reasonable compromises are possible

DIRAC-Rucio bridge status

- The RFC prototype has been tested extensively in Belle II
- Not all the catalog methods are implemented (yet)
  - E.g. managing metadata
- Work is ongoing
  - Implementing the lacking functionality
  - Handling of Rucio configuration via the DIRAC Configuration Service
  - Adaptation for multi-VO support

DIRAC
THE INTERWARE

Ongoing development focus

Python3 migration

- Python3 client is available starting from v7r2 (current production)
  - Technology preview level
- v7r3 in production:
  - Python3 client side becomes the default
  - Python3 server side becomes available
- v8r0 (fall 2021): drop python2 support

HTTPS based client/server protocol

- DIRAC used a custom client/server protocol and service framework (DISET) for many years
  - Much more efficient than e.g. XMLRPC at the time of introduction
- HTTP(S) protocol
  - Standard
  - Many interoperability libraries (OAuth, REST, etc)
  - Better suited for Python3
- How:
  - Using Tornado framework
  - The DIRAC base client is instrumented to understand both DISET and HTTP protocols depending on the service URL
    - No code changes needed, seamless migration
  - Service handlers need minor modifications

HTTPS based client/server protocol

DISET:

```
In [27]: from DIRAC.Resources.Catalog.FileCatalogClient import FileCatalogClient

In [28]: FileCatalogClient().whoami()
Out[28]:
{u'OK': True,
 u'Value': {u'DN': u'/C=ch/O=DIRAC/OU=DIRAC CI/CN=ciuser/emailAddress=lhcb-dirac-ci@cern.ch',
  u'group': u'dirac_user',
  u'identity': u'/C=ch/O=DIRAC/OU=DIRAC CI/CN=ciuser/emailAddress=lhcb-dirac-ci@cern.ch',
  u'isLimitedProxy': False,
  u'isProxy': True,
```

HTTPS:

```
In [31]: import requests

In [32]: url='https://server:8443/DataManagement/TornadoFileCatalog'
    ...: cert='/tmp/x509up_u1000'
    ...: kwargs={'method':'whoami'}
    ...: caPath='/home/dirac/ClientInstallDIR/etc/grid-security/certificates/'
    ...: with requests.post(url,data=kwargs,cert=cert, verify=caPath) as r:
    ...:     print r.json()
    ...:
{u'OK': True, u'Value': {u'DN': u'/C=ch/O=DIRAC/OU=DIRAC CI/CN=ciuser/emailAddr
Proxy': True, u'validGroup': False, u'validDN': False, u'issuer': u'/C=ch/O=DIR
lUser'], u'identity': u'/C=ch/O=DIRAC/OU=DIRAC CI/CN=ciuser/emailAddress=lhcb-c
38813'}}
```

HTTPS based client/server protocol

- HTTPS services are available in v7r2 already now as technology preview
  - Configuration, FileCatalog
  - All services will be HTTP-aware starting from 8.0
- This is an RPC protocol
  - "json over http"
- The REST interface will be also provided in the new Tornado based framework
  - The existing REST interface is still maintained in several installations, e.g. EGI
- The HTTP based protocol will be instrumented with the OIDC tokens authentication (*see below*)

DIRAC running environment

- DIRAC philoshopy – all the dependencies are included in he distribution
- DIRACOS – current production
  - Based on Fedora mock
  - Linux flavor independent
  - Python code installs in $PYTHONPATH
  - Will stay for Python 2 installations
- DIRACOS2 – available since v7r2 (current production) for Python3 only installations
  - Based on Conda, conda-forge packages
  - Linux, MacOS
  - Deployed in user space
  - “pip install” DIRAC python code inside the DIRACOS2 environment

DIRAC client installation summary

- Several methods to install the DIRAC client software on user workstations/laptops (Linux flavors)
  - **dirac-install** installer tool (old style still available)
    - DIRACOS
    - Rather tedious (see tutorials)
    - Suitable for various flavors of Linux
  - **Docker** container (Linux, MacOS)
    - docker run -it -v $HOME:$HOME -e HOME=$HOME diracgrid/client:egi
  - **CVMFS** client installation (Linux)
    - source /cvmfs/dirac.egi.eu/dirac/bashrc_egi
  - **DIRACOS2** for Python 3 installations
    - Using standard Python packaging and installation tools
      - pip install

DIRAC
THE INTERWARE

AAI integration

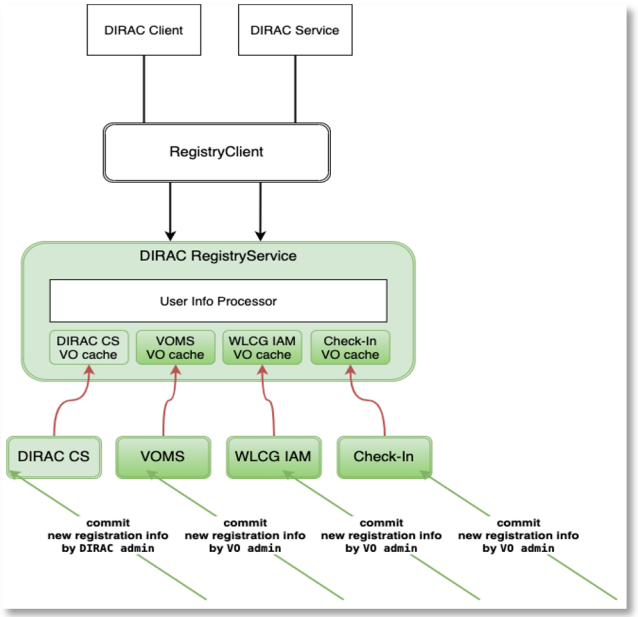
SSO solutions

- There are multiple examples of SSO solutions

- AAI service are based on OAuth2/OIDC technologies
  - Handling user identities
  - Managing user communities
  - Managing delegations of users rights in a distributed environment

DIRAC AAI solution

- Single solution for various Identity Providers (IdP)
  - EGI Check-In, (WLCG) Indigo AIM, etc
- Community management is fully done on the side of Identity Provider
  - Similar to VOMS
  - Defining user groups/roles/scopes
- DIRAC keeps a cache of user data dynamically updated from the IdP information
  - Including mapping of user roles onto the DIRAC groups
- Access control to DIRAC services is defined by the user DIRAC group
- TokenManager service to provide tokens to asynchronous operations
- The solution is now (partially) in certification for the next DIRAC release 8.0

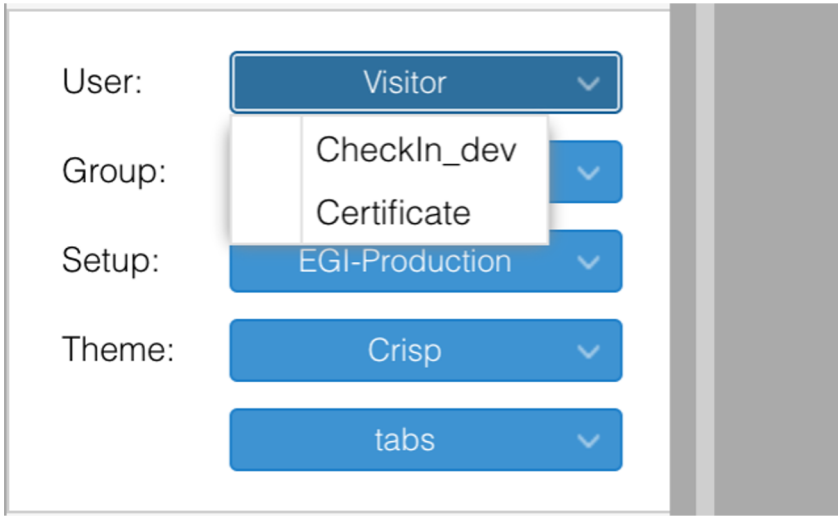



```
training_pilot
    Users = alitov, atsareg, vmendez
    Properties = GenericPilot, LimitedDelegation,
    VO = training.egi.eu
    VOMSRole = /training.egi.eu
training_user
    Users = alitov, atsareg
    Properties = NormalUser
    JobShare = None
    AutoUploadProxy = True
    VO = training.egi.eu
    VOMSRole = /training.egi.eu
```

User view

LOG IN

WebApp portal users will be able to choose authorization methods by selecting a certificate or identity provider:



Example of executing a command from DIRAC CLI (device code flow):

```
[[alitov@mardirac DIRAC]$ dirac-login -g checkin-integration_user -P
Use next link to continue, your user code is "PXXJ-ZNGF"
https://marosvn32.in2p3.fr/DIRAC/auth/device
Authorization pending.. (use CNTL + C to stop)
```

Conclusions

- Large scientific communities have to employ various geographically distributed computing and storage resources
- DIRAC provides a framework for building distributed computing systems aggregating multiple types resources
- DIRAC is constantly evolving to cope with the new emerging computing technologies and provide users with a most efficient and comfortable environment

http://diracgrid.org

Acknowledgements

- This work is co-funded by the EOSC-hub project (Horizon 2020) under Grant number 777536
- EGI-ACE receives funding from the European Union's Horizon 2020 research and Innovation programme under grant agreement no. 101017567
- France-Grilles community, FG-DIRAC service admins, FG animation team

http://diracgrid.org

DIRAC
THE INTERWARE

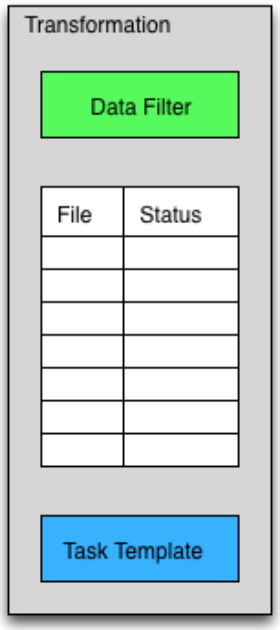
Backup slides

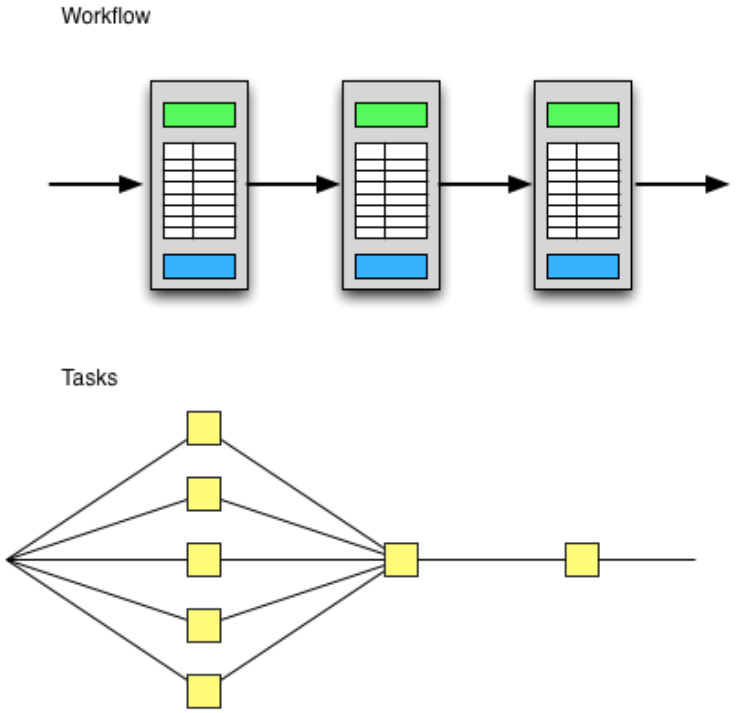
DIRAC
THE INTERWARE

Managing workflows

Transformation System for data driven workflows

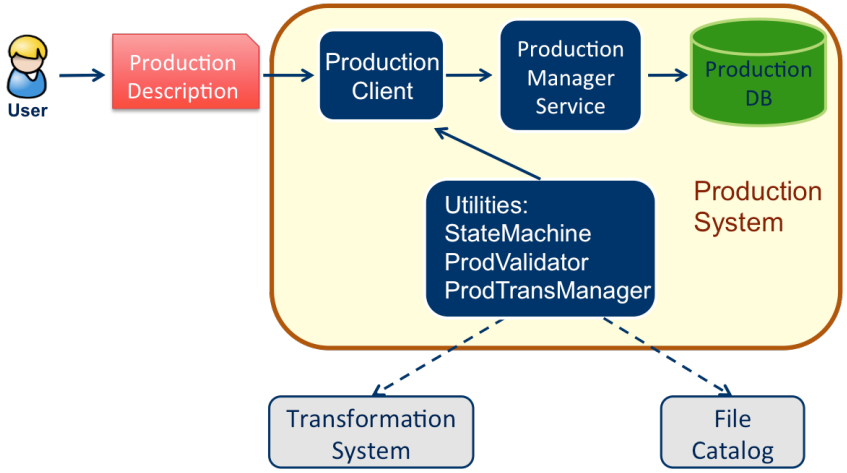
- Data driven workflows as chains of data transformations
  - Transformation: input data filter + recipe to create tasks
  - Tasks are created as soon as data with required properties is registered into the system
  - Tasks:
    - Jobs submission
    - Data replication, removal
    - *etc*
- Transformations can be used for automatic data driven bulk data operations
  - Scheduling RMS tasks
  - Often as part of a more general workflow





Production System

- TS automatizes a single step of workflow execution
  - Need to monitor dozens of transformations at once
  - Manual transformation definition is error prone
- LHCb, ILC, Belle II developed specific Production Systems on top of the TS
  - Found many commonalities



- Production System to help transformations management
  - Automatic transformation instantiation based on the production definition
  - Fully data-driven
  - Transformation are connected via "links" – metadata queries
- Part of the DIRAC distribution
  - In use by the CTA Collaboration