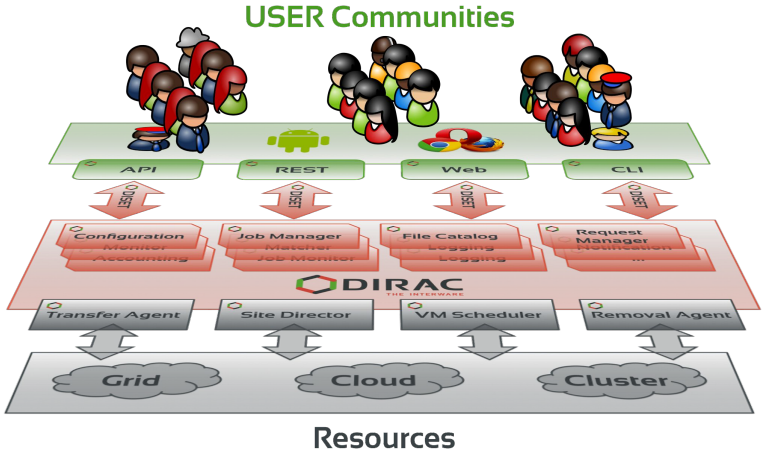
Introduction + recent and ongoing developments

DIRAC User Group meeting, EGI 2022, September 20th 2022

Federico Stagni
DIRAC technical coordinator
federico.stagni@cern.ch

What's DIRAC?

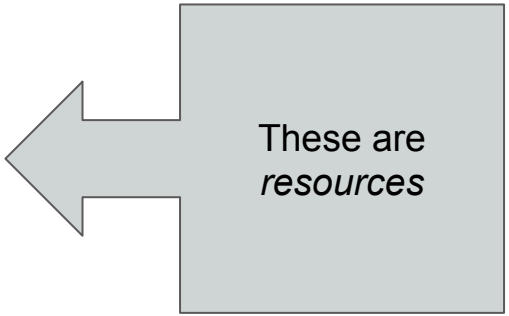
- A software framework for distributed computing
- A **complete** solution to one (or more) user community
- Builds a layer between users and resources



- Started as an LHCb project, experiment-agnostic in 2009
- Developed by communities, for communities
  - Open source (GPL3+), GitHub hosted
  - Python 3.9 (python 2.7 kept for legacy releases)
  - Publicly documented, active assistance forum, yearly users workshops, open developers meetings and hackathons
- The DIRAC consortium as representing body

… a few examples of what DIRAC can be used for

- sending jobs to “the Grid”
  - the obvious one...
- interfacing with different *sites*
  - with different *computing elements*
    - and *batch systems*
  - with different *storage elements*
- interfacing with different *information systems*
- interfacing with different *catalogs*
- interfacing with different *MQs, DBs*
- authenticate through different *identity providers* *(in preparation)*



- managing “productions” (e.g. reconstruction, simulation…)
- managing dataset transfers and removals
  - or delegate the task
- interacting with FTS
- providing a failover system
  - your jobs won’t fail because a certain SE is down, nor because of central service are down
- (if possible at all) transfer data from the experiment (the “online”) to a Grid SE (the “offline”)
- monitor your resources with a policy-based system
- … and more

Installations and communities

(that I know)

A *framework* shared by multiple experiments/projects, both inside HEP, astronomy, and life science

Experiment agnostic

Extensible

Flexible

Features

- **DIRAC as-a-service** (1 installation, several VOs)
  - especially good for medium-small communities
  - Apart from EGI, national initiatives exist to provide DIRAC as-a-service (e.g. GridPP, DutchGrid) – very few DIRAC administrators for possibly dozens of communities
- Feature-rich, all-in-one (WMS, DMS, but also Productions and Dataset management, and monitoring)
  - again, good for limited-manpower communities
- Tightly-integrated DIRAC WebApp
- Actively developed and maintained

"Stable"

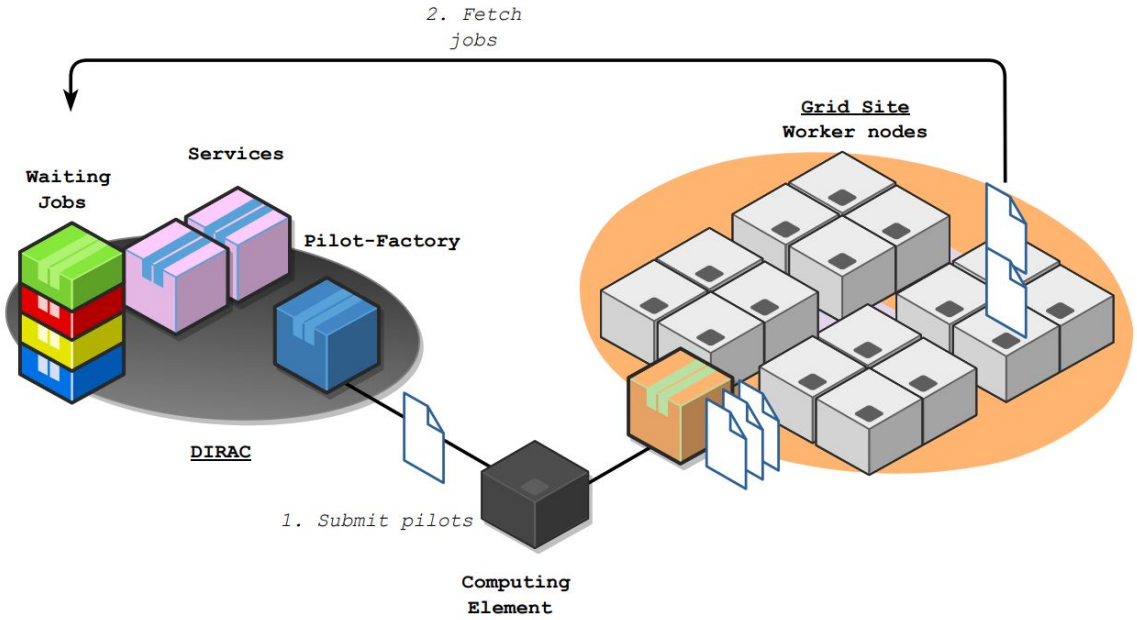
from workshops

"Communication is the killer feature of DIRAC"

[WMS] basics

Pilots-based WMS, with late binding

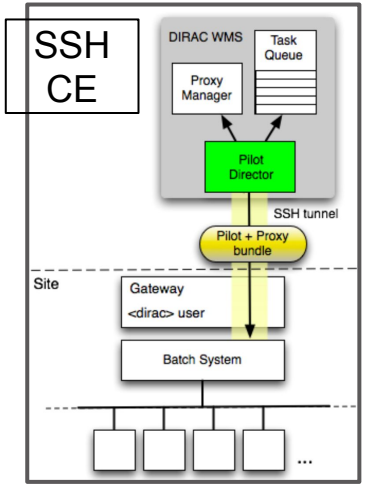
- **Users** define and submit **jobs**. Jobs have **requirements**. Job descriptions are stored in DIRAC's Job DB.
- Independently, **Pilots** are started (1) on the **sites' worker nodes (WN)**
- Pilots will try to **match** (2) the worker nodes' capabilities to the jobs requirements.
- Jobs are started on WNs. DIRAC monitors their progress.

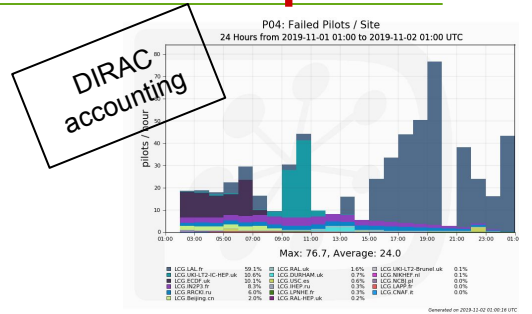


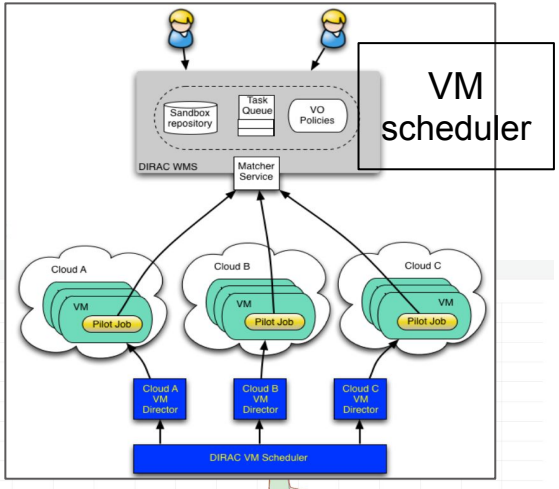
[WMS] Computing resources

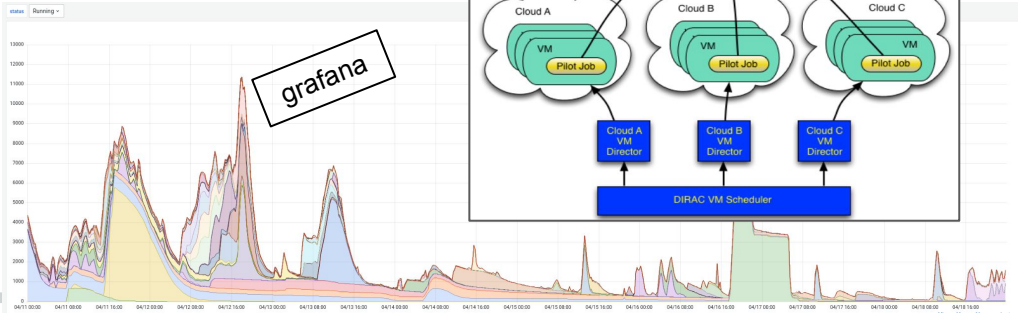
where to run the pilots

- **Grids** (nowadays: HTCondor, ARC)
- **Clusters** behind a BS
  - access through SSH/GSISSH tunnel
    - a really thin layer that we call “SSH CE”
  - and then talks with batch system
- **VMs scheduler**:
  - Based on apache libcloud
  - Contextualization from standard images
    - with, at least, the DIRAC pilot
- **Vacuum**:
  - VAC/vcycle resources
  - BOINC Volunteer resources
  - HLT farm (LHCb)
- **HPC sites**
  - it often means at least SSH+Slurm
    - more later on





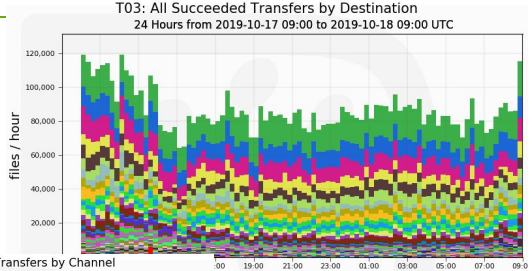


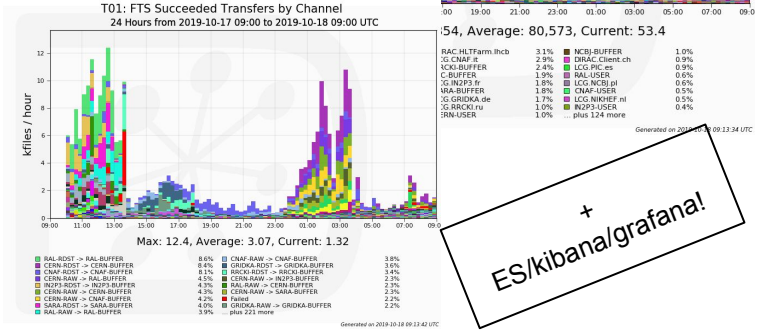


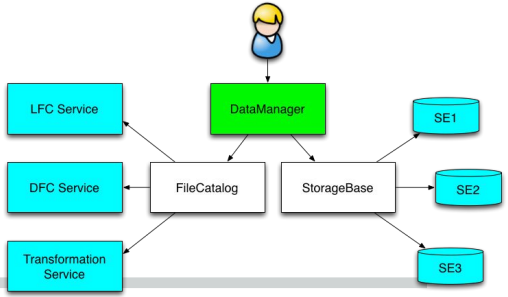
[DMS] Data Management

Basics of **DIRAC** DMS:

- **LFN**s: unique identifier within DIRAC of a file
  Logical File Name
  (described as paths)
- LFNs are registered in **catalog**(s).
  and there are implementations like the DFC
- LFNs *may* have **PFN**s, stored in **SE**s.
  Physical File Name on **Storage Elements**
  (and SEs are monitored, within the DIRAC Resource Status System)
  You can access those PFNs with several protocols.
- DIRAC **catalogs** for implementing namespace functionalities
  - Several catalogs can live in parallel
    → **DFC**: full replica and metadata catalog
    → *plugins* for LFC, Rucio, DIRAC TS, LHCb Bookkeeping







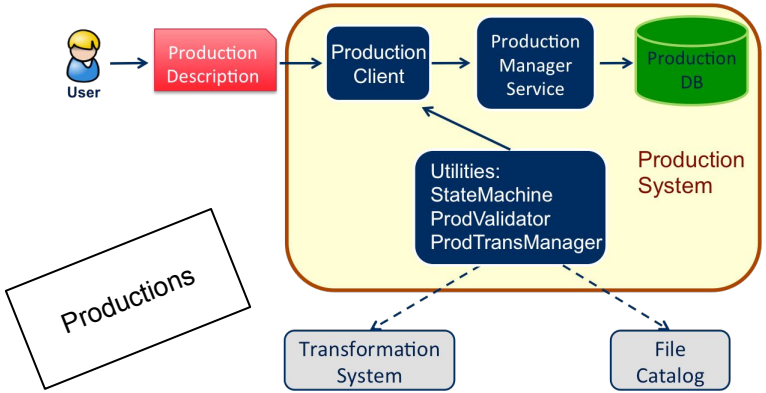
[DMS] DIRAC+Rucio

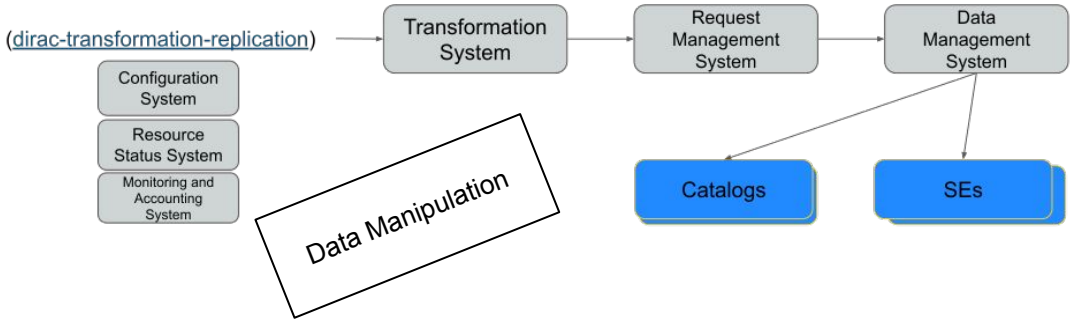
- Some VOs using DIRAC would like to use Rucio as DMS
  - and maybe some VOs using Rucio would like to use DIRAC
- Discussions started at the 8th DIRAC workshop (May 2018)
- Since January 2021 Belle2 uses DIRAC and Rucio
  - from LCG file catalog to Rucio FC
- Few developments done on both sides
  - available: integration of (multi-VO) DIRAC with (multi-VO) Rucio, working without a rucio.cfg

Job Productions and datasets management with the “Transformation System”

The Transformation System (TS) is used to automate common tasks related to production activities

- A *“production”* is a transformation managed by the TS that is a “Data Processing” transformation (e.g. Simulation, Merge, DataReconstruction…). A Production ends up creating jobs in the WMS.
- A “Data Manipulation” transformation replicates, or remove, data from storage elements. A “Data Manipulation” transformation ends up creating requests in the RMS (Request Management System), which feeds the DMS.

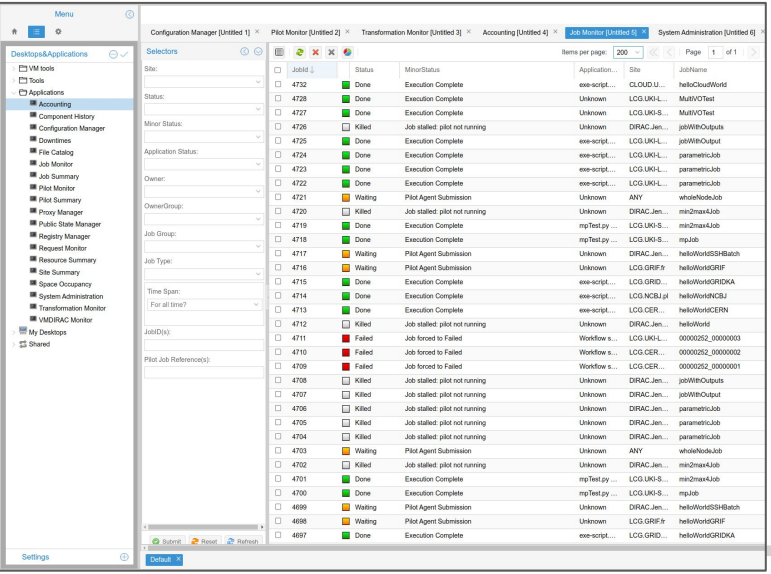


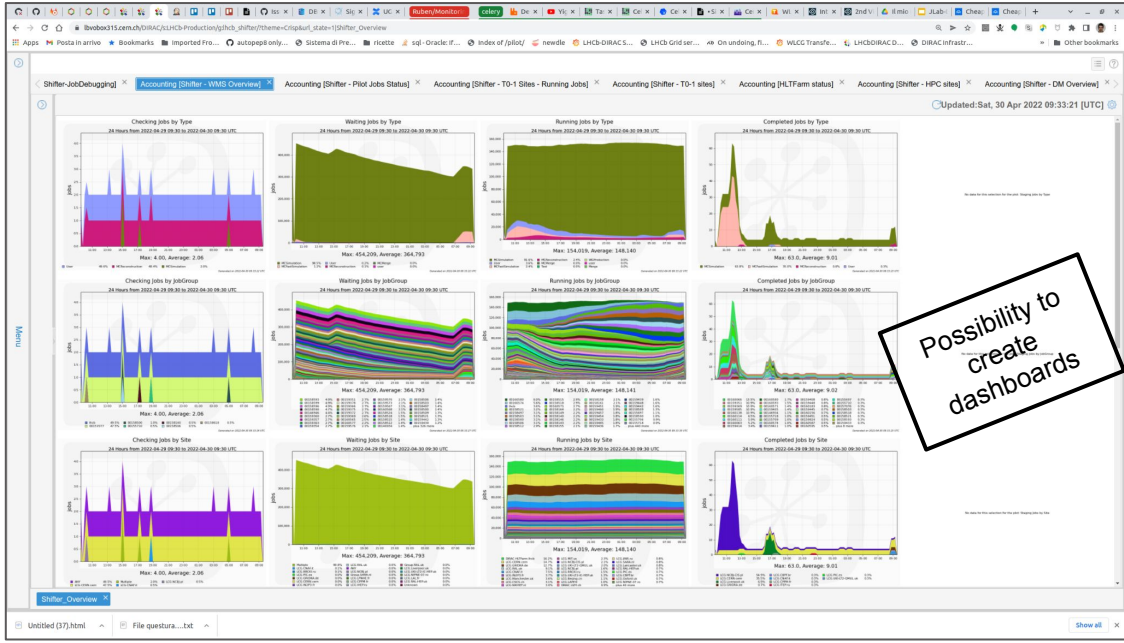


The TransformationSystem is finely tuned and can manage millions of jobs and files daily

DIRAC WebApp

- Web apps available for monitoring jobs, files, productions, etc.
- Configurable
- Extendable



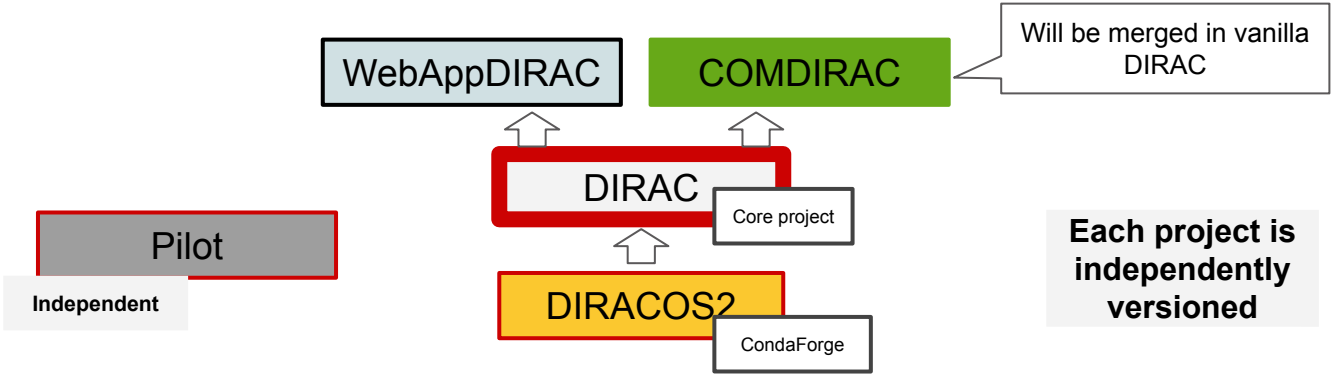


DIRAC extensions

"Horizontal" extensibility

-

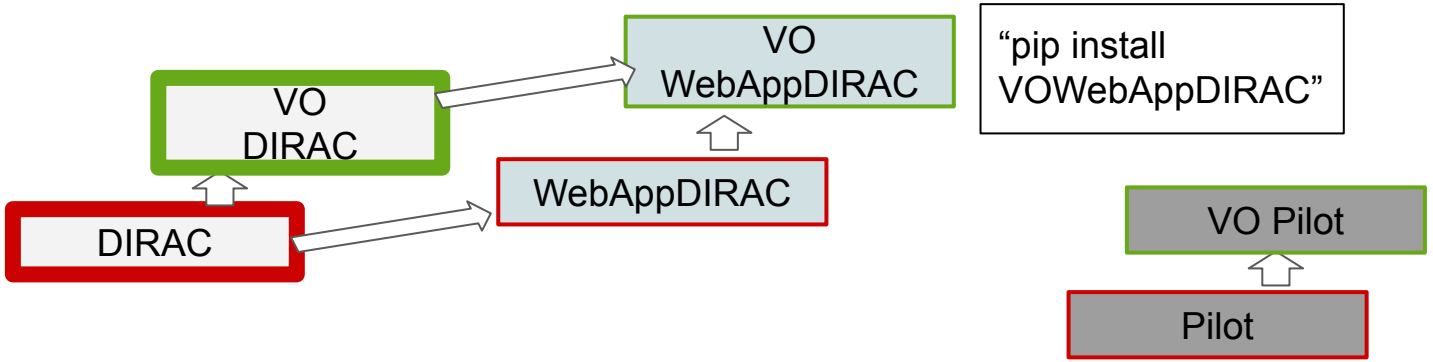
For specific requirements



"Vertical" extensibility

-

Community driven

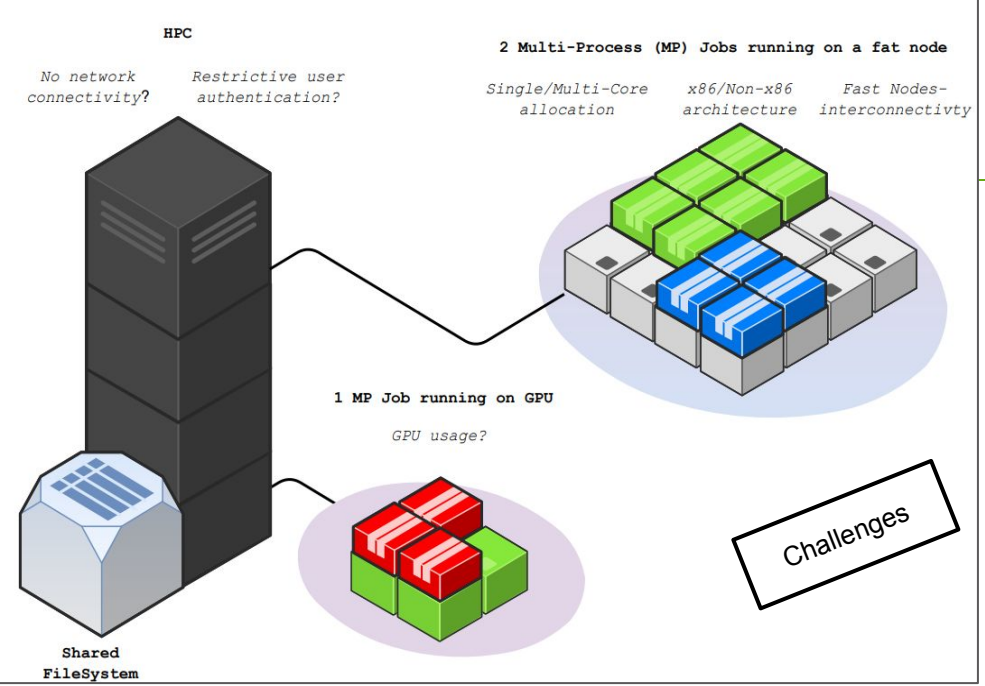


DIRAC issues

- complex, with high entrance bar
  - getting better dropping python2 compatibility
- somewhat cumbersome deployment
  - getting better dropping python2 compatibility
- often a bit late on “standards”
  - http services
  - tokens
  - monitoring
- “old”-ish design

Some DIRAC developments

- always valid: Integrating DIRAC workflows in HPCs
- always valid: DMS advancements
- Done: Python 3
  - py3 clients supported since version 7.2 (pip installable)
  - py3 server supported since version 7.3
  - py2 support ended with 8.0 (released last week)
    - with some obvious exceptions of part of pilots code
- Ongoing (basically Done): ES/kibana/grafana dashboards
- Ongoing/advanced: `dips://` → `https://`
  - `dips`: DIRAC proprietary protocol for RPC calls
  - `http`: based on tornado
  - several DIRAC services already available using HTTP, and adding more
    - `http` will be the default for all the DIRAC services from version 8.1
- Ongoing: token support, and IdP (IaM, Check-in)
- Ongoing: running on kubernetes (goal: define a *helm* chart)
- Started: using celery and RabbitMQ (retiring part of DIRAC framework)



Running on HPCs

Different solutions must be adopted for different HPCs. DIRAC can only take care of the distributed computing issues, such as:

- For MP WNs: logical partitioning using DIRAC “inner” PoolCE
- Interfacing with the batch system (mostly slurm)
- Interfacing with WNs with limited network connectivity using the *PushJobAgent*

VO actions often needed.

On software challenges: the good news is that py3 versions of DIRAC clients (so, DIRAC pilots) can be installed on *ppc64le* and *aarch64*

→ but, you need to be able to start the pilots

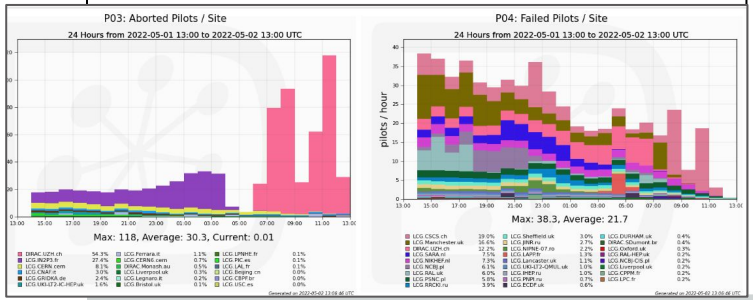
Tokens and OAuth2

- Current production release depends on VOMS
- DIRAC v8 rationalizes many aspects related to AuthN, AuthZ, Tokens and OAuth2 support, and adds *experimental* support to new Identity Providers (IaM and CheckIn)
- Longer term goals include:
  - externalize (to IdPs) users' management
  - use tokens (and/or proxies) for interfacing with computing and storage resources (v8.1)

Accounting and Monitoring

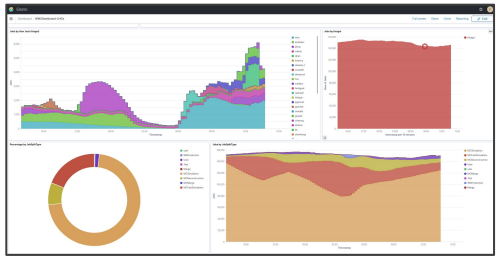
Accounting:

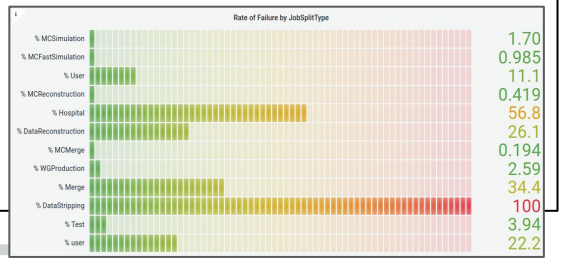
- For historic data
  - Jobs
  - Pilots
  - Data Operations
  - Storage
- MySQL backend
- DIRAC Web App dashboard



Monitoring:

- Real Time monitoring and not only
- ElasticSearch (OpenSearch) backend
- Visualize in kibana, grafana, and (partially) DIRAC WebApp
- largely improved within DIRAC v8



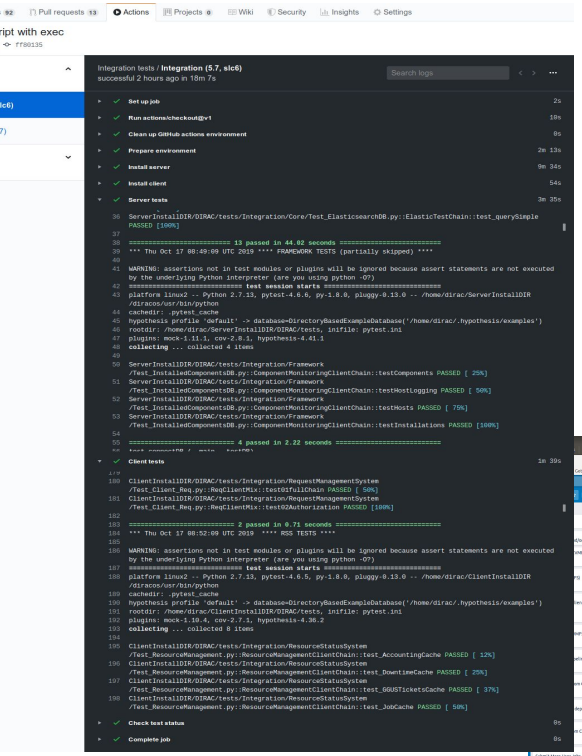


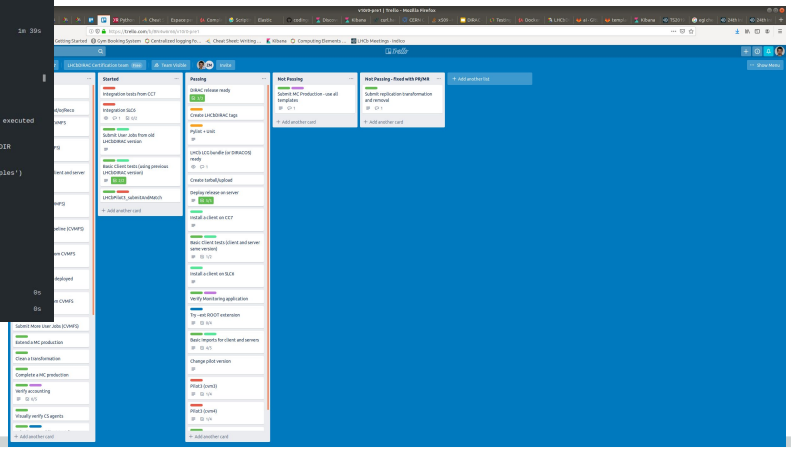
Development and testing

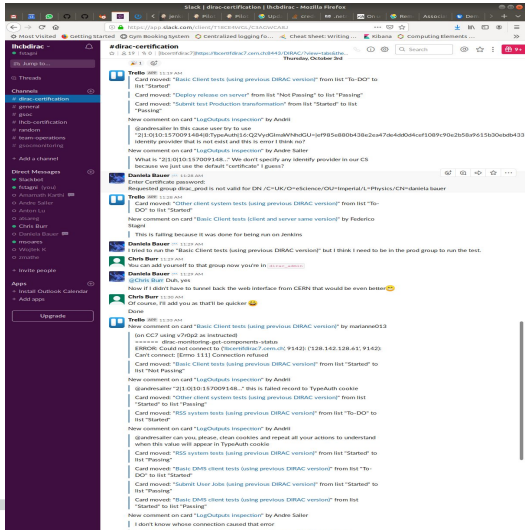
~5 FTE as core developers, a dozen contributing developers

Tests, certification, integration process is a daily work.

- We use (lots of) GitHub Actions, and Jenkins for some bits
- We run certification hackathons every 2nd week







Questions/comments?

- dirac.readthedocs.io
  - including code documentation
- Ops and general questions: Google forum – but we prefer github discussions
- Dev and DevOps issues: on github
- Bi-weekly developers meetings (and/or hackathons): BILD

backup

[DMS] Storage Elements

- **Storage Elements**: abstraction of the storage endpoints
  - fully described in Configuration Service (CS)
  - several abstractions of the same physical endpoint are possible
  - Multi-protocol
    - DIP: DIRAC custom protocol
    - File: offers an abstraction of the local access as an SE.
    - RFIO (deprecated): for the rfio protocol.
    - Proxy: to be used with the StorageElementProxy.
    - S3: for S3 (e.g. AWS, CEPH)
    - GFAL2_SRM2: for srm
    - GFAL2_XROOT: for xroot
    - GFAL2_HTTPS: for https
    - GFAL2_GSIFTP: for gsiftp
  - SpaceOccupancy plugins:
    - BDIIOccupancy, WLCGAccountingJson, WLCGAccountingHTTPJson
- SEs definitions are sync-ed from DIRAC CS to Rucio RSE via a DIRAC agent

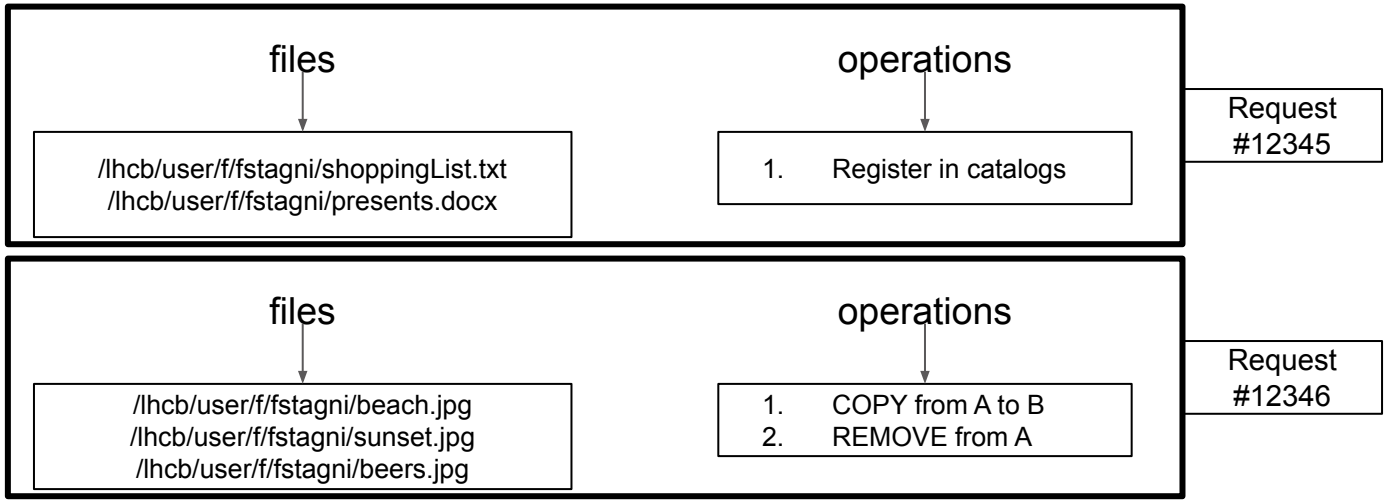
```
CERN-BUFFER
{
  BackendType = Eos
  AccessProtocols = root, gsiftp, https
  WriteProtocols = root, gsiftp, https
  SEType = T0D1
  SpaceReservation = LHCb-EOS
  OccupancyLFN = /eos/lhcb/proc/accounting
  OccupancyPlugin = WLCGAccountingJson
  GFAL2_XROOT
  {
        Host = eoslhcb.cern.ch
        Protocol = root
        Path = /eos/lhcb/grid/prod
        Access = remote
        Path = /eos/lhcb/grid/prod/lhcb/buffer
  }
  GFAL2_GSIFTP
  {
        Host = eoslhcbftp.cern.ch
        Protocol = gsiftp
        Path = /eos/lhcb/grid/prod
        Access = remote
        Path = /eos/lhcb/grid/prod/lhcb/buffer
  }
  GFAL2_HTTPS
  {
        Host = eoslhcb.cern.ch
        Protocol = https
        Path = /eos/lhcb/grid/prod
        Access = remote
        Path = /eos/lhcb/grid/prod/lhcb/buffer
  }
}
```

[RSS] Resource Status System

- Stores info on the status of Resources (e.g. SEs)
- An autonomic computing tool evaluates a few policies to determine the status of the resources. E.g.:
  - space left < threshold → ban for writing
  - endpoint in downtime in GocDB → ban r/w
  - …
- DIRAC SEs states are sync-ed from DIRAC RSS to Rucio via a DIRAC agent

DIRAC RMS

A generic system, which can be used for queueing (also) DMS *operations*



and the Request Executing Agent will… execute the requests.

Operation types:

- ReplicateAndRegister (e.g. using FTS)
- RemoveFile/RemoveReplica
- …others (not useful for this pres)
- ...add your own (e.g. ReplicateUsingAnotherExternalSystem)

DIRAC TS

A generic system for queueing similar *operation types* on certain *datasets* and forward them to the appropriate *systems*

An *operation type* can be, e.g.: • a simulation workflow • a reconstruction workflow • a replication • a removal • ...	A *dataset* is split into groups, based on criterias defined by *plugins*, e.g.: • split by size • by destination • by metadata • … [code it]	A *system* is either (today) the DIRAC WMS (for productions) or the DIRAC RMS (for dataset management operation types)

E.g. `Take all my holidays pictures from 2018 with tag='sunset', make sure that there is one copy on tape and one on disk, distributed on all the sites according to free space, and group the operations by group of at most 100 files.`