Towards the Future: DiracX

26/03/2023 ISGC

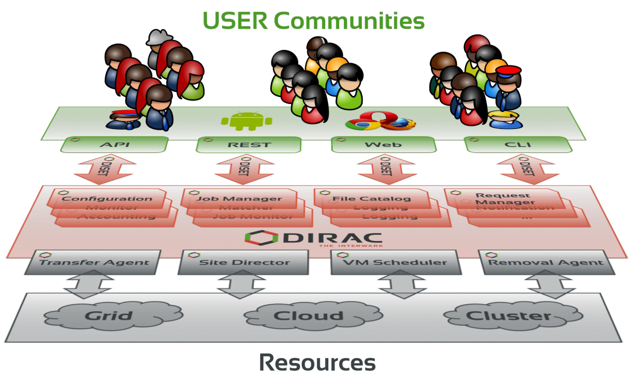
Alexandre Boyer, Christopher BURR, Christophe Haen, Federico Stagni, Andrei tsaregorodtsev

A Modern Incarnation of the DIRAC Framework

What is DIRAC

Slide that has been presented for years, with minimal variations

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



- Developed by communities, for communities
  - Open source (GPL3+), GitHub hosted
  - Python 3
  - Publicly documented, yearly users workshops, open developers meetings and hackathons
  - Deployed mostly via Puppet on VMs (really, not bound to any specific technologies)
- The DIRAC consortium as representing body

In summary: DIRAC is an open source project and the governing body includes institutes behind other experiments.

LHCb is the driving force behind the developments.

Installations and communities

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

<u>Experiment agnostic</u>

<u>Extensible</u>

<u>Flexible</u>

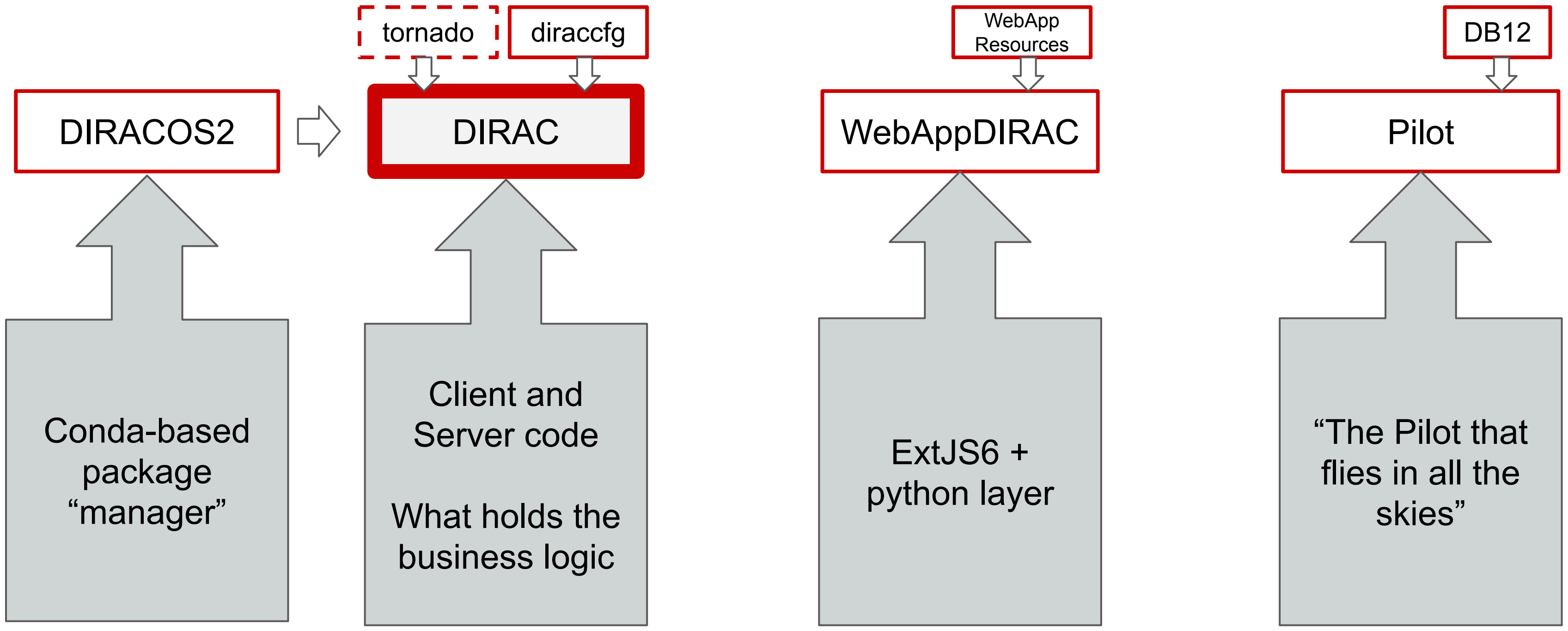
Brief history

- 2000: Started in LHCb as an MC production system
- 2002: “DIRAC2”, python, xml-rpc, Grid, DataChallenge 04 (Pilot jobs)
- 2006: “DIRAC3” 3 year long full refurbishment (DISET, Configuration SYstem, Accounting, etc)
- 2008: Multi-VO, split into “vanilla” and extensions, DFC

Successful project

- Project evolving from an experiment specific to a general-purpose one
- Pilot based architecture adopted by all the LHC experiments and multiple grid infrastructures
- Rare example of an efficient complex solution
  - Both WMS and DMS at a scale
- Contributions from more that a hundred developers during 20 years of the project life
  - Plus specific extensions

Today's DIRAC (py3) stack



DIRAC issues

- Complex, with high entrance bar
- Somewhat cumbersome deployment
- Late on “standards”
- Oldish design
- Not very developer friendly
- multi-VO is an afterthought
- No clear interface to a running DIRAC service
- Custom WebApp

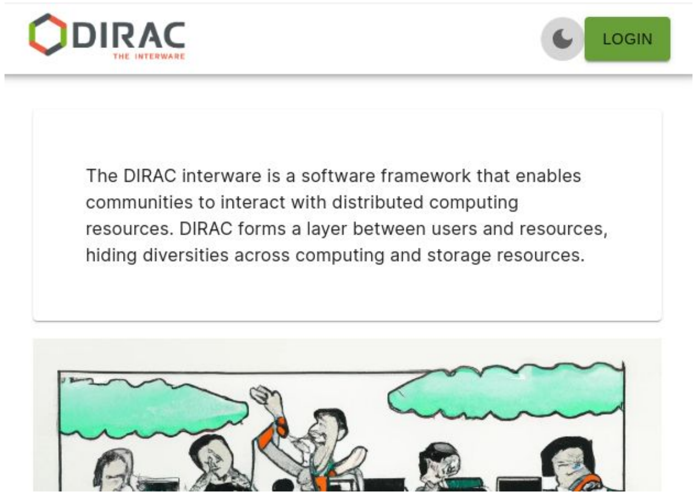
DiracX

- DIRAC is old and is filled with technical debt
  - Attempts to make major change now systematically fail (OAuth2, HTTPS)
- DIRAC was very well thought out with a solid foundation
  - Wasn’t clear what a “Grid” even was when it started
- DiracX is a new approach, learning from the past 20 years
  - Learn from 20+ years of DIRAC
  - Tens of years of developer experience

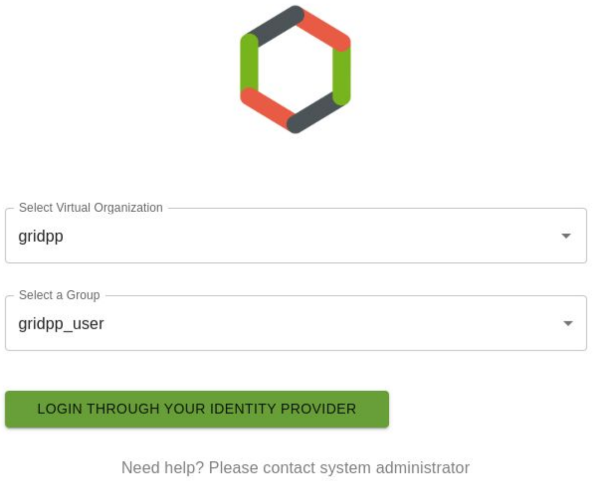
DiracX: requirements list

- Make authentication transparent to users (no certificate errors)
- Simpler interfaces and clearer errors
- First class Multi VO support
- More flexibility (e.g. access via HTTPS without a DIRAC client)
- More stable releases
- Simpler installation and configuration
- Easier to maintain extensions (especially for the webapp)
- More accessible to new developers

Standard logging methods



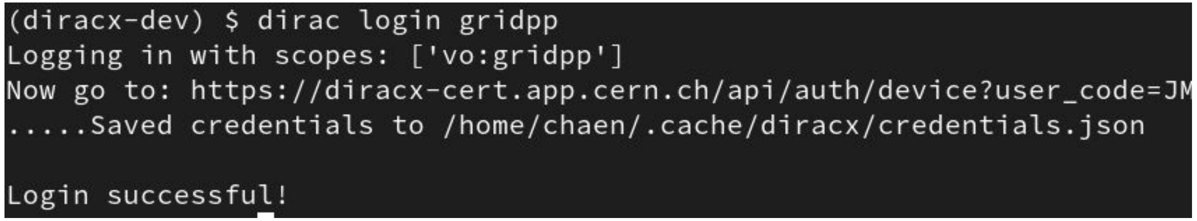
The DIRAC interware is a software framework that enables communities to interact with distributed computing resources. DIRAC forms a layer between users and resources, hiding diversities across computing and storage resources.



Select Virtual Organization: gridpp

Select a Group: gridpp_user

LOGIN THROUGH YOUR IDENTITY PROVIDER
Need help? Please contact system administrator

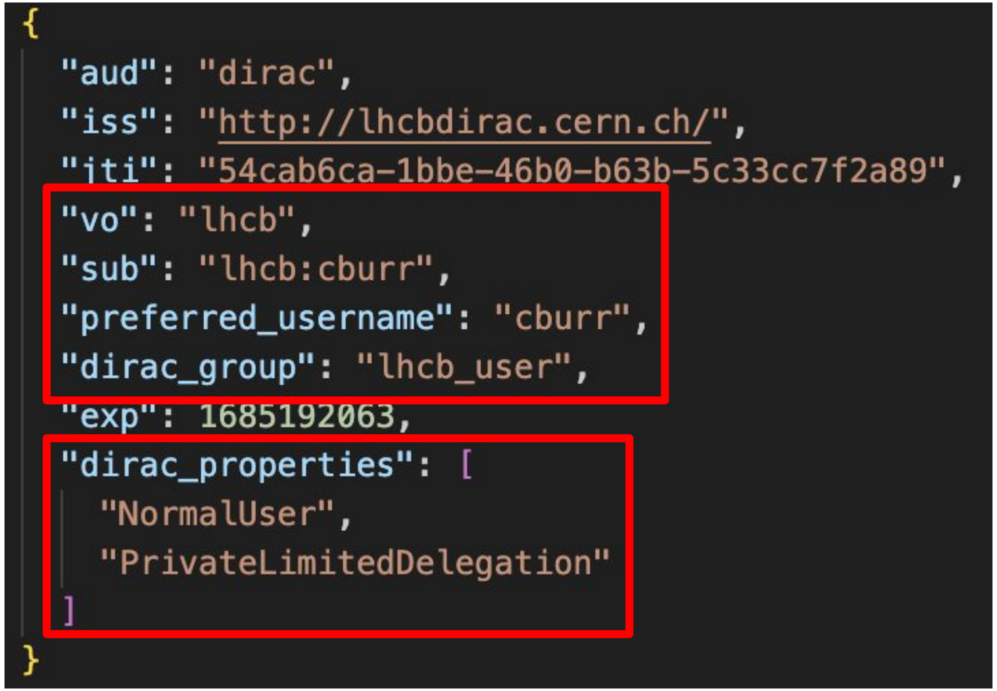


```
(diracx-dev) $ dirac login gridpp
Logging in with scopes: ['vo:gridpp']
Now go to: https://diracx-cert.app.cern.ch/api/auth/device?user_code=JM
.....Saved credentials to /home/chaen/.cache/diracx/credentials.json

Login successful!
```

Change in Internal Auth/Autz model

- DIRAC:
  - X509 proxies
  - Identity based (DN + group)
  - Config lookup to assess permissions
- DiracX:
  - Tokens
  - Permissions embedded in the token
  - Oauth flows
- The change should be:
  - *Transparent* to users
  - More flexible for experts
- Security model



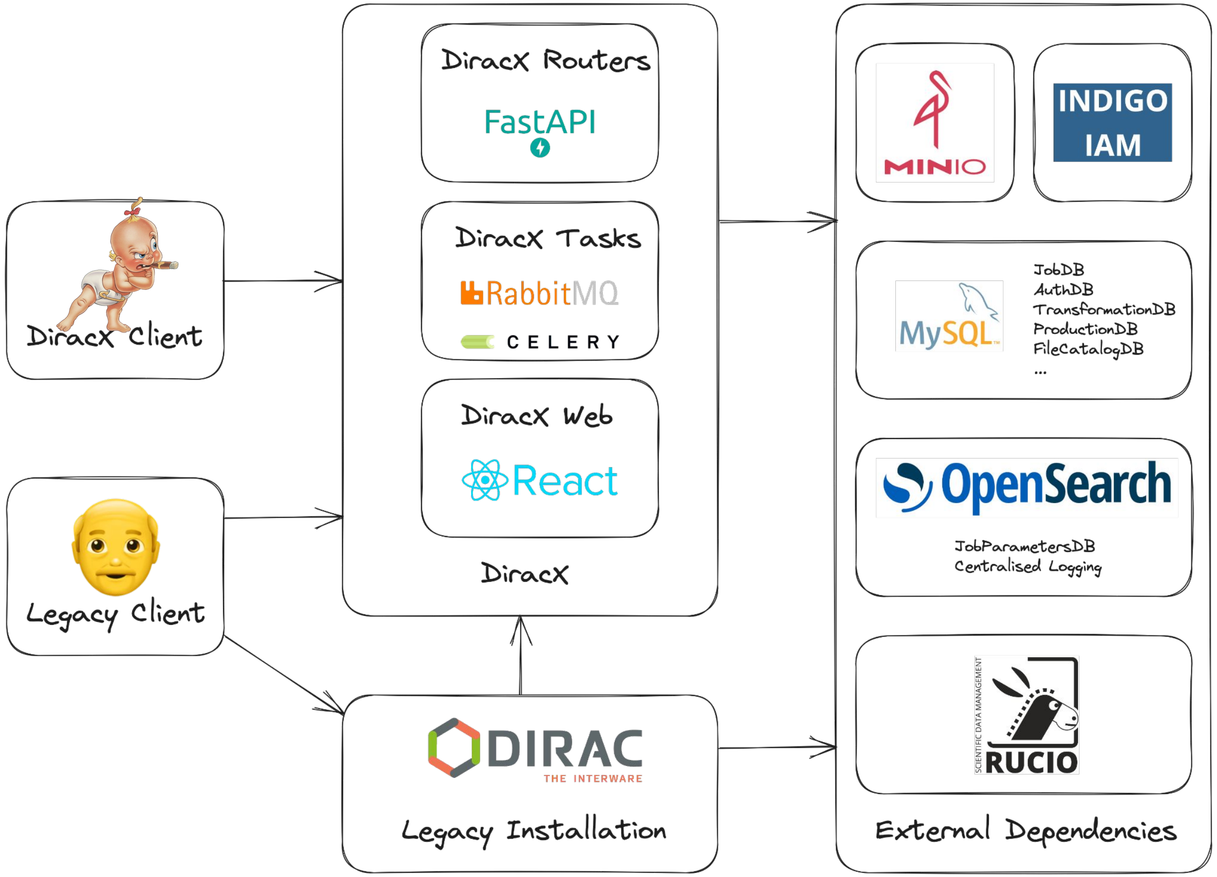
```
{
  "aud": "dirac",
  "iss": "http://lhcbdirac.cern.ch/",
  "jti": "54cab6ca-1bbe-46b0-b63b-5c33cc7f2a89",
  "vo": "lhcb",
  "sub": "lhcb:cburr",
  "preferred_username": "cburr",
  "dirac_group": "lhcb_user",
  "exp": 1685192063,
  "dirac_properties": [
    "NormalUser",
    "PrivateLimitedDelegation"
  ]
}
```

Architecture: DIRAC

- ***DB*** classes connects to the databases
- Services expose the DB classes to ***Clients***
- ***Agents*** are cron-like job executing periodic tasks
- ***Clients*** are called by ***Agents***, scripts, API, etc
- WebApp calls ***Services*** directly or uses ***Clients***

Reminder: pretty much everything is custom (protocol, serialization, plotting, etc)

Architecture: DiracX



Services -> FastAPI

FastAPI

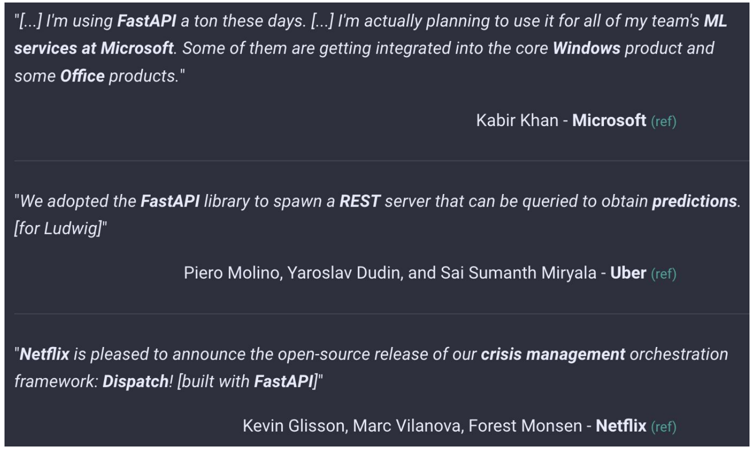
https://fastapi.tiangolo.com/

High performance framework and widely used at scale

Designed for easy prototyping and development

Removes a lot of low level code and boilerplate

Standards based



*"[...] I'm using **FastAPI** a ton these days. [...] I'm actually planning to use it for all of my team's **ML services at Microsoft**. Some of them are getting integrated into the core **Windows** product and some **Office** products."*

Kabir Khan - **Microsoft** (ref)

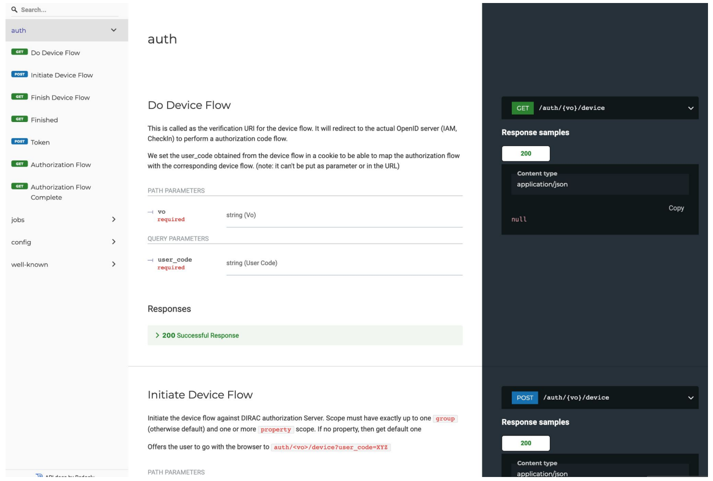
*"We adopted the **FastAPI** library to spawn a **REST** server that can be queried to obtain **predictions**. [for Ludwig]"*

Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala - **Uber** (ref)

*"**Netflix** is pleased to announce the open-source release of our **crisis management** orchestration framework: **Dispatch**! [built with **FastAPI**]"*

Kevin Glisson, Marc Vilanova, Forest Monsen - **Netflix** (ref)

Swagger/ReDoc



auth

Do Device Flow

This is called as the verification URI for the device flow. It will redirect to the actual OpenID server (IAM, CheckIn) to perform a authorization code flow.

We set the user_code obtained from the device flow in a cookie to be able to map the authorization flow with the corresponding device flow. (note: it can't be put as parameter or in the URL)

PATH PARAMETERS

vo required	string (Vo)

QUERY PARAMETERS

user_code required	string (User Code)

Responses

> 200 Successful Response

GET /auth/{vo}/device

Response samples

200

Content type
application/json

Copy

```
null
```

Initiate Device Flow

Initiate the device flow against DIRAC authorization Server. Scope must have exactly up to one `group` (otherwise default) and one or more `property` scope. If no property, then get default one

Offers the user to go with the browser to `auth/<vo>/device?user_code=XYZ`

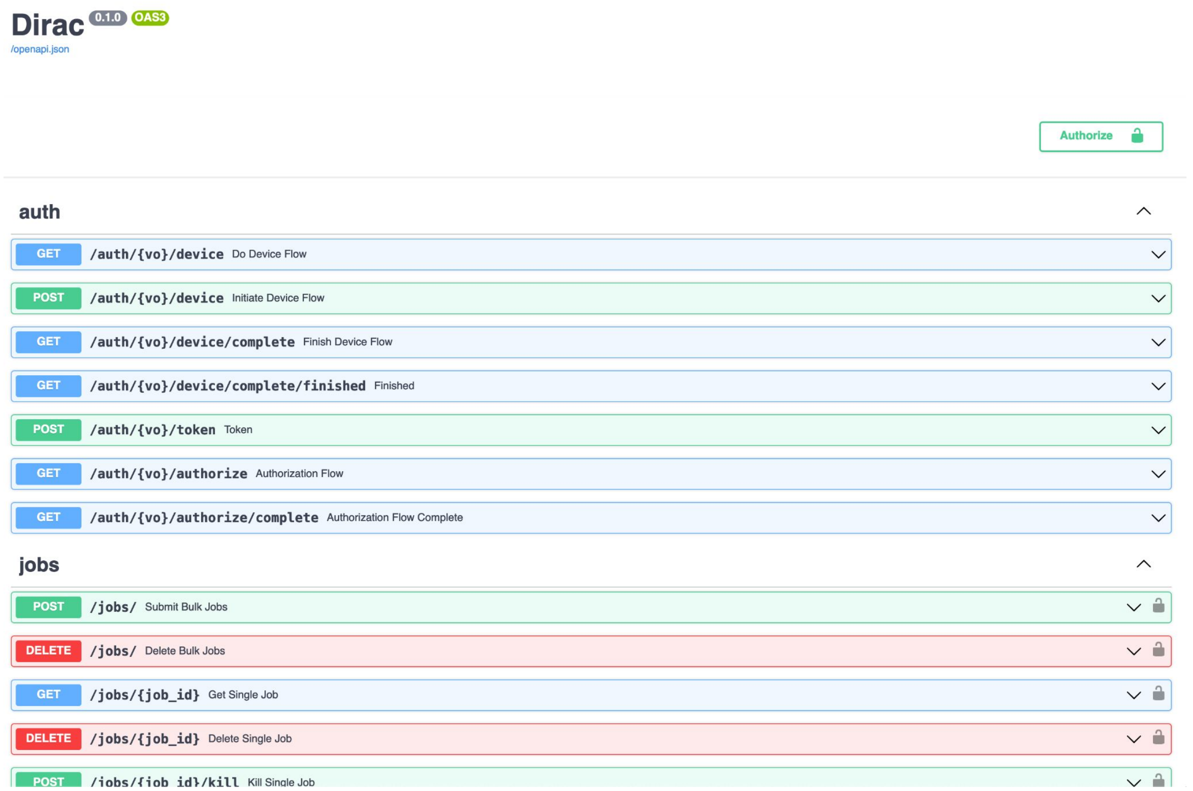
PATH PARAMETERS

POST /auth/{vo}/device

Response samples

200

Content type
application/json



auth

GET	/auth/{vo}/device	Do Device Flow
POST	/auth/{vo}/device	Initiate Device Flow
GET	/auth/{vo}/device/complete	Finish Device Flow
GET	/auth/{vo}/device/complete/finished	Finished
POST	/auth/{vo}/token	Token
GET	/auth/{vo}/authorize	Authorization Flow
GET	/auth/{vo}/authorize/complete	Authorization Flow Complete

jobs

POST	/jobs/	Submit Bulk Jobs
DELETE	/jobs/	Delete Bulk Jobs
GET	/jobs/{job_id}	Get Single Job
DELETE	/jobs/{job_id}	Delete Single Job
POST	/jobs/{job_id}/kill	Kill Single Job

Swagger/redoc generate interactive documentation from the JSON

Included in FastAPI by default

Agents -> Celery

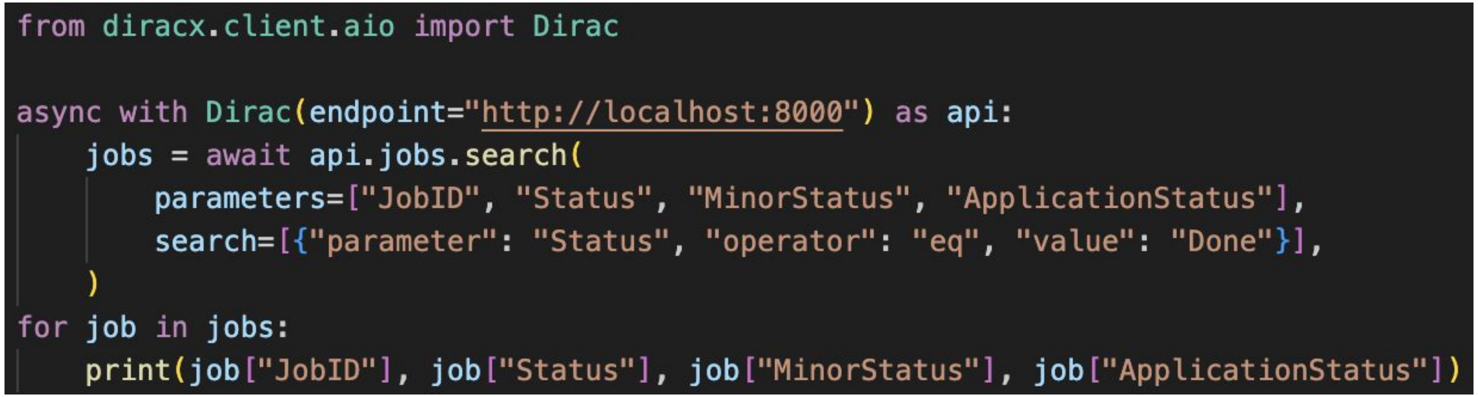
CELERY

- We need more than just API calls
- Long running “things” (seconds -> hours)
- Covers “Agents”, “Requests” and “Executors” in DIRAC

- Will be turned into asynchronous tasks
- Celery works well for this and is widely used

Clients

- Auto generated from the OpenAPI json generated by FastAPI
- Using Autorest
  - Developed by Microsoft for Azure and used by DigitalOcean
  - Supports many languages including Python



```python
from diracx.client.aio import Dirac

async with Dirac(endpoint="http://localhost:8000") as api:
    jobs = await api.jobs.search(
        parameters=["JobID", "Status", "MinorStatus", "ApplicationStatus"],
        search=[{"parameter": "Status", "operator": "eq", "value": "Done"}],
    )
for job in jobs:
    print(job["JobID"], job["Status"], job["MinorStatus"], job["ApplicationStatus"])
```

WebApp Evolution

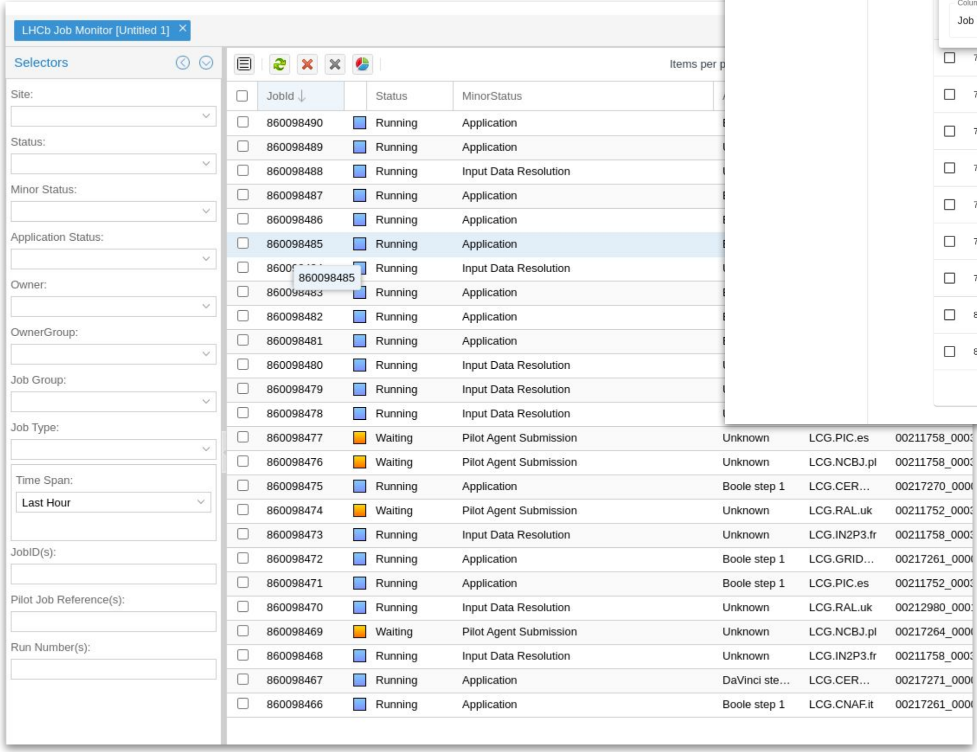
DIRAC

- Highly custom: not based on a framework (not easy to modify, lack of support)
- Based on vendor lock-in libraries: components rely on ExtJS, which requires a custom compiler to work
- Tightly coupled with DIRAC itself

DiracX

- Similar requirements as for DiracX itself
- Typescript, NextJS, React, Material UI

The new WebApp



LHCb Job Monitor [Untitled 1]

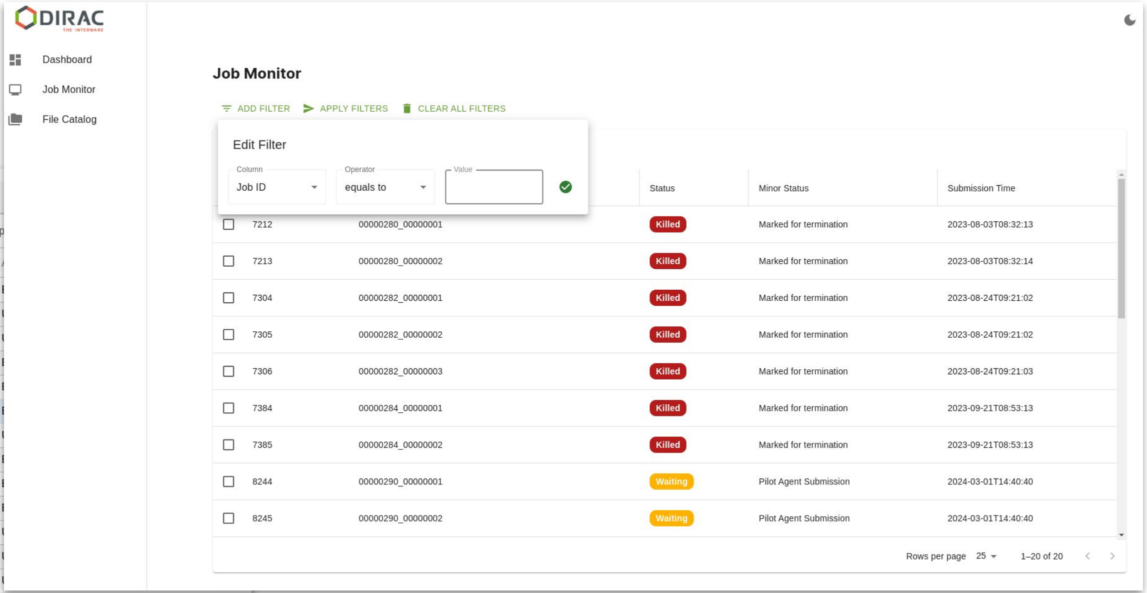
Selectors

Site:	
Status:	
Minor Status:	
Application Status:	
Owner:	
OwnerGroup:	
Job Group:	
Job Type:	
Time Span:	Last Hour
JobID(s):	
Pilot Job Reference(s):	
Run Number(s):	

Items per p

JobId		Status	MinorStatus			
860098490	🟦	Running	Application	E		
860098489	🟦	Running	Application	U		
860098488	🟦	Running	Input Data Resolution	U		
860098487	🟦	Running	Application	E		
860098486	🟦	Running	Application	E		
860098485	🟦	Running	Application	E		
8600	🟦	Running	Input Data Resolution	U		
860098483	🟦	Running	Application	E		
860098482	🟦	Running	Application	E		
860098481	🟦	Running	Application	E		
860098480	🟦	Running	Input Data Resolution	U		
860098479	🟦	Running	Input Data Resolution	U		
860098478	🟦	Running	Input Data Resolution	U		
860098477	🟧	Waiting	Pilot Agent Submission	Unknown	LCG.PIC.es	00211758_0003
860098476	🟧	Waiting	Pilot Agent Submission	Unknown	LCG.NCBJ.pl	00211758_0003
860098475	🟦	Running	Application	Boole step 1	LCG.CER…	00217270_000
860098474	🟧	Waiting	Pilot Agent Submission	Unknown	LCG.RAL.uk	00211752_0003
860098473	🟦	Running	Input Data Resolution	Unknown	LCG.IN2P3.fr	00211758_0003
860098472	🟦	Running	Application	Boole step 1	LCG.GRID…	00217261_000
860098471	🟦	Running	Application	Boole step 1	LCG.PIC.es	00211752_0003
860098470	🟦	Running	Input Data Resolution	Unknown	LCG.RAL.uk	00212980_000
860098469	🟧	Waiting	Pilot Agent Submission	Unknown	LCG.NCBJ.pl	00217264_000
860098468	🟦	Running	Input Data Resolution	Unknown	LCG.IN2P3.fr	00211758_0003
860098467	🟦	Running	Application	DaVinci ste…	LCG.CER…	00217271_000
860098466	🟦	Running	Application	Boole step 1	LCG.CNAF.it	00217261_000

860098485



Job Monitor

Edit Filter

Column	Operator	Value
Job ID	equals to	

		Status	Minor Status	Submission Time
7212	00000280_00000001	Killed	Marked for termination	2023-08-03T08:32:13
7213	00000280_00000002	Killed	Marked for termination	2023-08-03T08:32:14
7304	00000282_00000001	Killed	Marked for termination	2023-08-24T09:21:02
7305	00000282_00000002	Killed	Marked for termination	2023-08-24T09:21:02
7306	00000282_00000003	Killed	Marked for termination	2023-08-24T09:21:03
7384	00000284_00000001	Killed	Marked for termination	2023-09-21T08:53:13
7385	00000284_00000002	Killed	Marked for termination	2023-09-21T08:53:13
8244	00000290_00000001	Waiting	Pilot Agent Submission	2024-03-01T14:40:40
8245	00000290_00000002	Waiting	Pilot Agent Submission	2024-03-01T14:40:40

Rows per page 25 1–20 of 20

Deployment

- DIRAC: custom scripts, manual work, based on runit (build our own RPM as no longer maintained)
- DiracX:
  - Kubernetes - Standard to define a distributed system
  - Separate infrastructure from applications
    - “Please IT department(/cloud provider) run this for me”
  - Helm gives the ability:
    - to parameterise
    - distribute a kubernetes config

DiracX Helm chart

- https://github.com/DIRACGrid/diracx-charts
- How can you use it?
    - If your institution provides a kubernetes service: use it
    - If you work with public clouds: use their container services
    - If you’re a smaller install: use a lightweight option (k3s/k0s/rke2)
- This is used for:
    - DiracX testing (GitHub actions)
    - Local development instance
    - Running a demo instance
    - Running various DIRAC test instances
    - Soon: running production instances

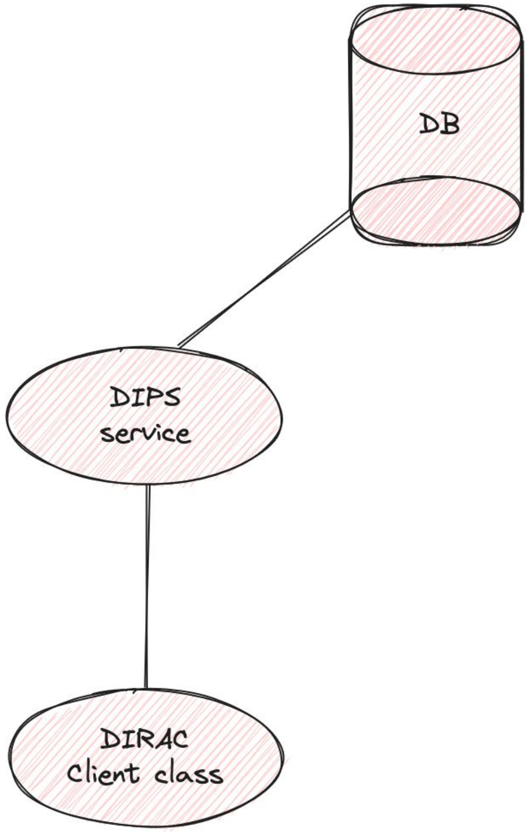
Migration

- Minimise operational work to migrate
- Avoid disruptive changes
- Don’t need hard things (downtimes, schema changes)
- Make the transition as simple as possible

Service migration

Current situation has:

- MySQL database
- DIPS service using a DB class
- DIRAC Client class

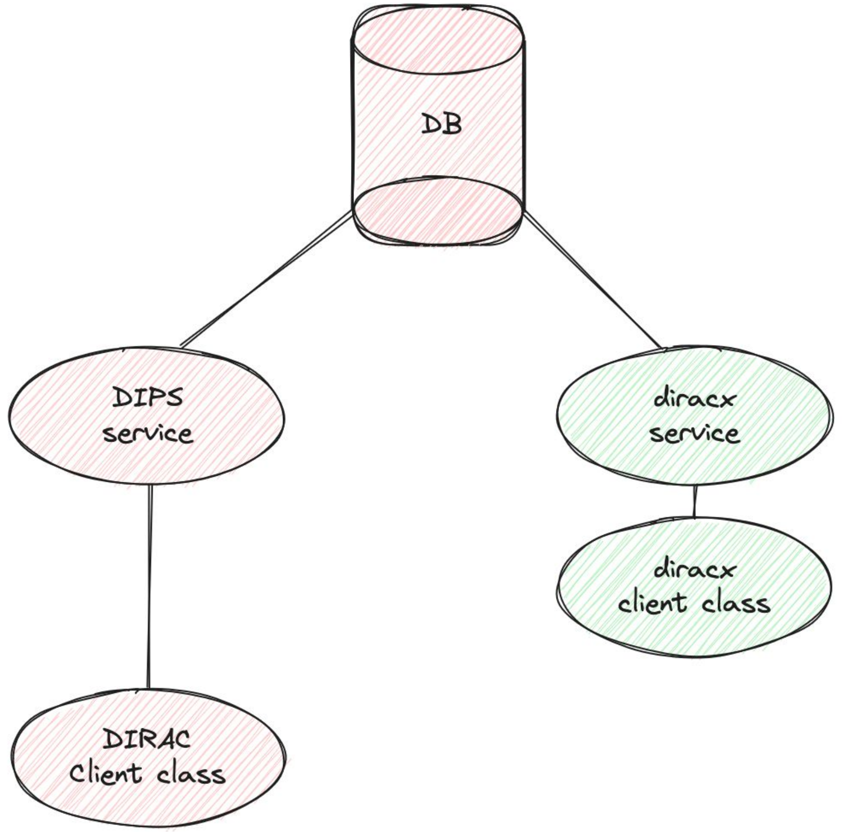


Service migration

The MySQL DB stays the same.

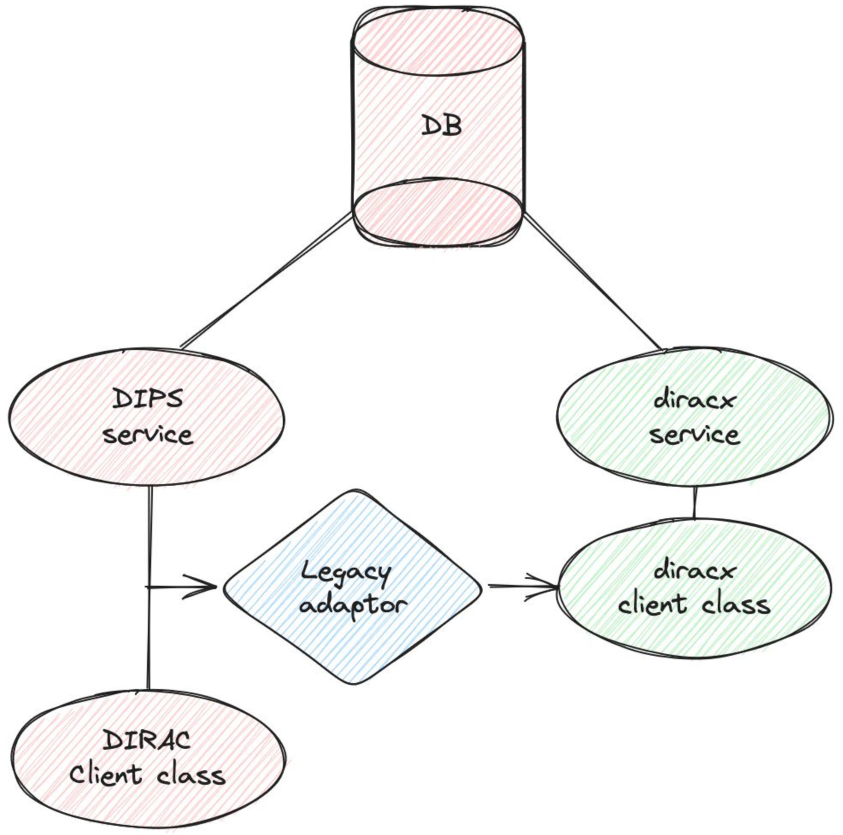
Develop in parallel:

- FastAPI router
- Async SQLAlchemy DB class
- Modern API + CLI + tests

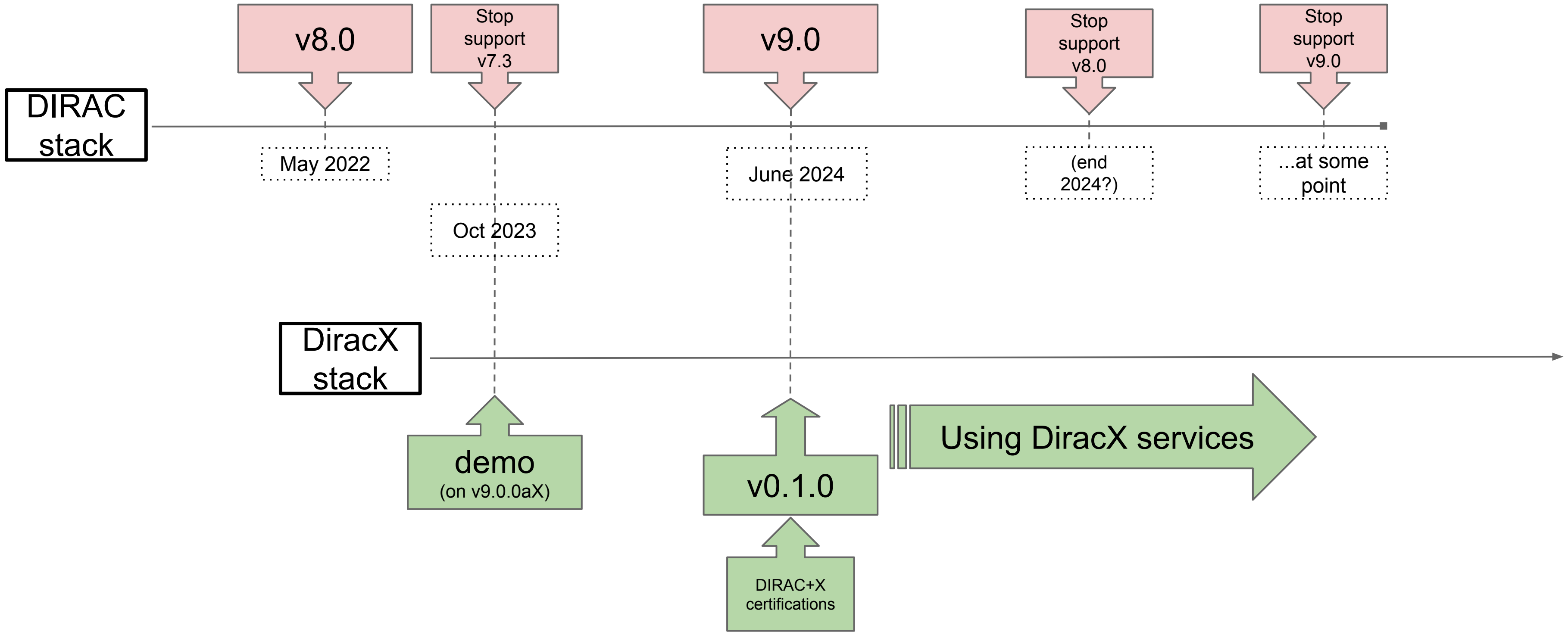


Service migration

Once diracx service is ready, add a “legacy adaptor”



DiracX Status

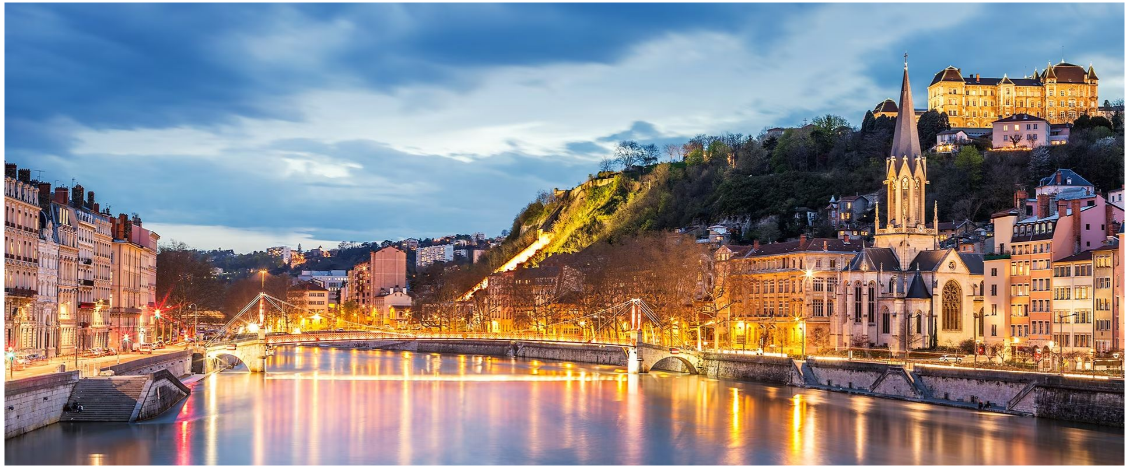


DiracX Status

- We still have a lot to finish
  - “Groundwork”
  - Interoperability with legacy DIRAC
  - Deployment
  - Telemetry and monitoring
  - Documentation
  - Extensions
- DiracX will need to be installed alongside DIRAC v9.0
- DiracX won’t do much at this point
  - But all of the groundwork for a smooth transition will be ready
- Functionality will then be slowly moved to DiracX
  - Lot’s of interest from the community

Hackaton & Workshop

- Very exciting times ahead
- Good opportunity to join
- Next hackathon @ CERN: 9-10 April 2024
- DIRAC workshop in Lyon, France: 19-21 June 2024



Questions?