The neXt Dirac incarnation

Federico Stagni
federico.stagni@cern.ch

October 23rd 2024 __ CHEP 2024

This is the story of why and how we decided to take a successful project and rewrite its code from scratch

DIRAC

THE INTERWARE

DIRAC^{x}

THE INTERWARE

What is DIRAC?

DIRAC

/ Documenta on

View page source

DIRAC

THE INTERWARE

The DIRAC interware is a complete Grid solu on for one, or more than one community of users that need to exploit distributed heterogeneous resources.

DIRAC forms a layer between a community and various compute resources to allow op mized, transparent and reliable usage. The types of resources that DIRAC can handle include:

Compu ng Resources, including Grids, Cloud

latest

Batch systems

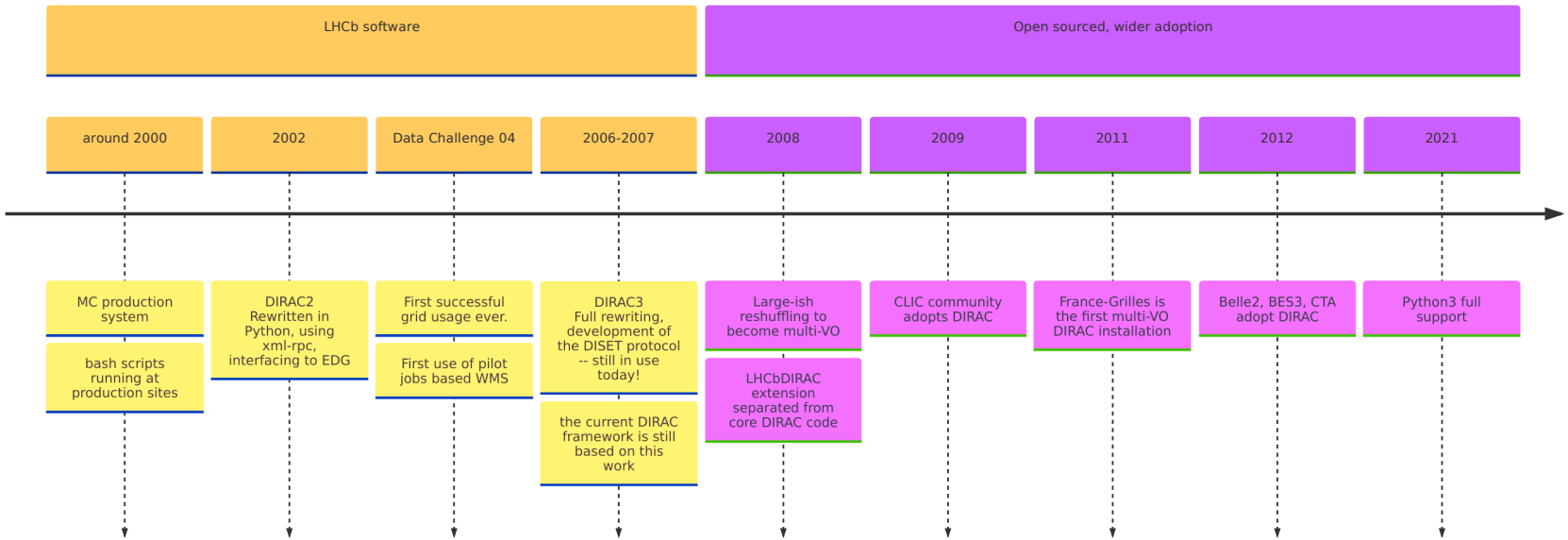
Action! (and extensions)

Few real life examples, also reported in this conference:

- LHCb stores the metadata and provenance of every produced file in a LHCb-specific database (with an Oracle backend)
  - see talk in Track 3 on Monday and poster #461 on `LbMCSubmit`
- Belle2 is a HEP experiment. Uses Rucio as a data management solution.
  - see talk in Track 1 this afternoon
- CTAO has radically different requirements (compared to HEP experiments) on how to process its data.
  - see talk in Track 4 tomorrow
- HERD is an astronomy and particle astrophysics experiment using dHTC for data management.
  - see talk in Track 4 tomorrow
- EGI uses DIRAC as WMS, and EGI CheckIn as an identity provider. Hosts (among others) WeNMR (structural biology and life science)

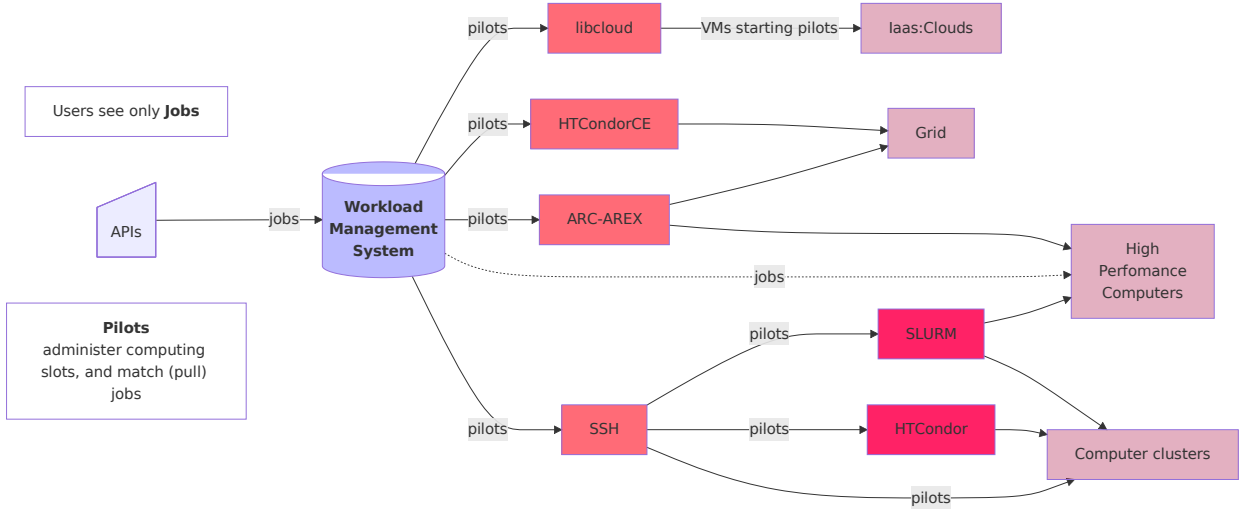
DIRAC is coded for being flexible and extendable

DIRAC timeline



Workload Management System

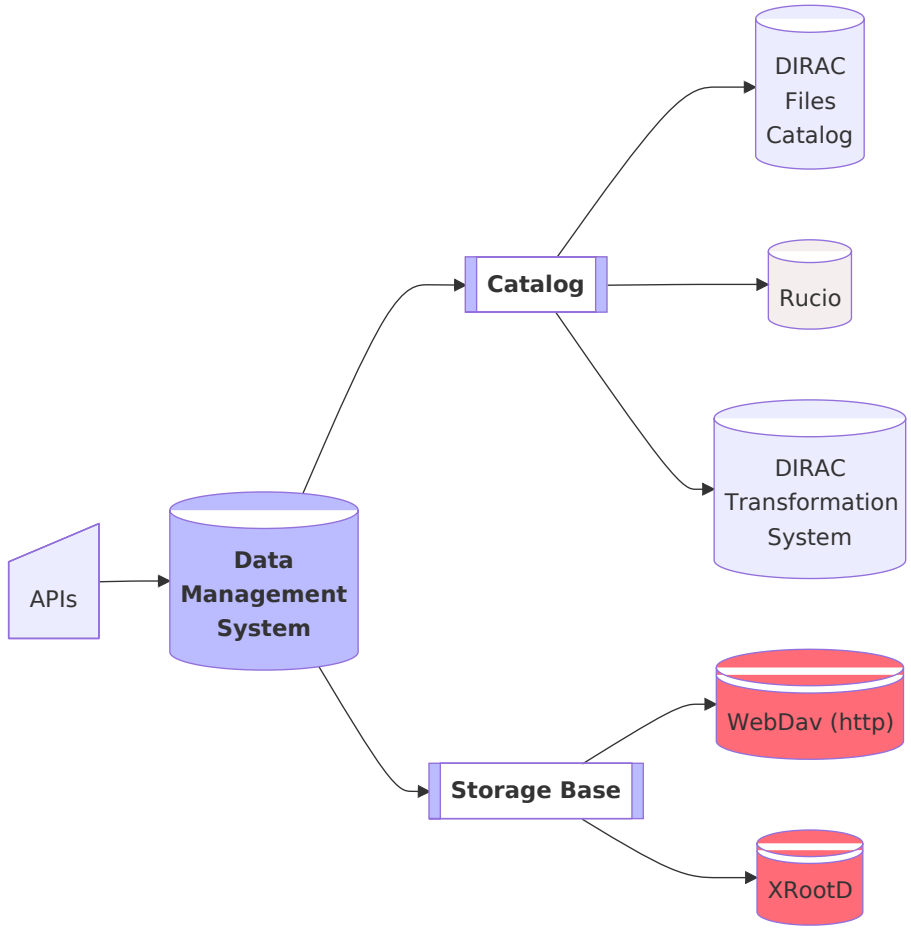
- Pull model based on Pilot jobs
- Also "Push" solution for HPCs that do not support pilots (because of limited internet access).
- Will integrate CWL (Common Workflow Language) as a way of defining jobs



Data Management System

It's about **files**: placing, replicating, removing files

- there are **LFNs** (logical file names)
- **LFNs** are registered in catalog(s)
  - where are the LFNs? (in the DIRAC File Catalog DFC , or in Rucio)
  - where are their metadata? (in the DFC, or in the LHCb Bookkeeping, or in AMGA
- LFNs mayhave **PFNs** (physical file names), stored in **SEs** Storage Elements), that can be accessed with several protocols.

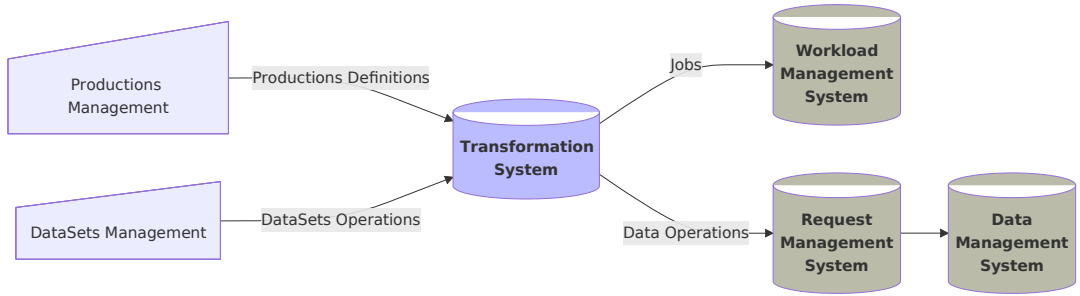


Transformation System

For productions and Dataset management

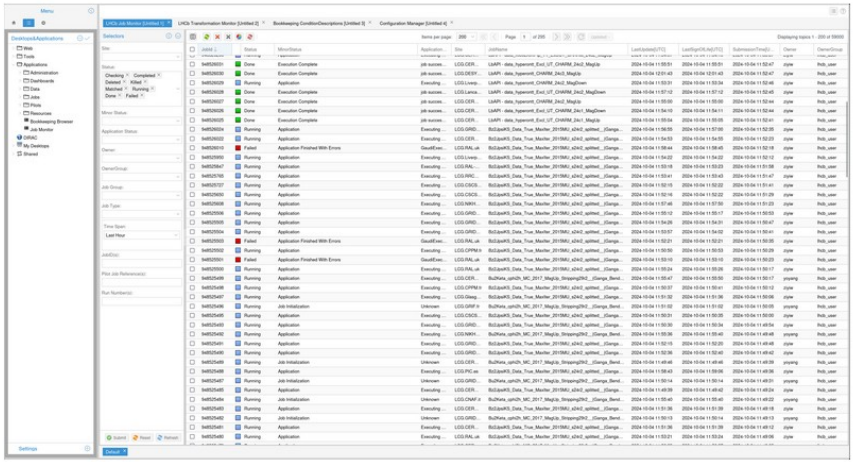
- A Data Processing **transformation** (e.g. Simulation, Merge, DataReconstruction...) creates jobs in the WMS (and re-submits them if needed, eventually destroys them).
- A Data Manipulation **transformation** replicates, or removes, data from storage elements.

The Transformation System is used to automate common tasks related to production activities. It can handle thousands of productions, millions of files and jobs.



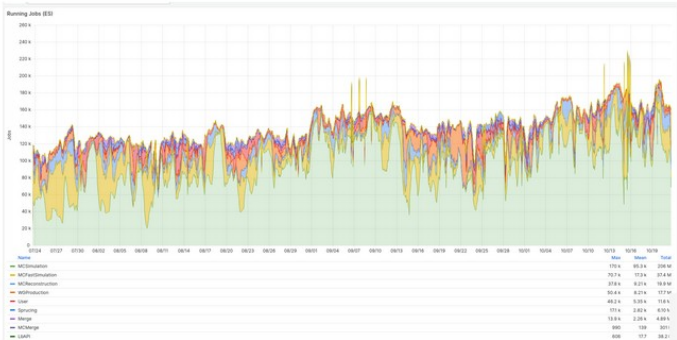
Visualizations

DIRAC also provides a WebApp:



Dashboards can be created within the DIRAC Web App:

and/or in Grafana:

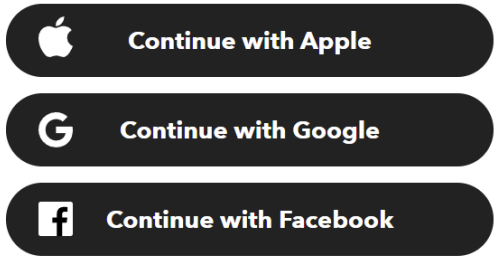


Technicalities

- DIRAC is written in python 3 (the Pilot still supports Python 2.7
- Services are exposed at urls like `dips://box.some.where:9132/WorloadManagement/`
  - `dips` stands for "DIRAC protocol"
- The DIRAC framework provides also "Agents" (~ cron jobs) and "Executors" (~ tasks execution) to animate the system
- As backends, MySQL and OpenSearch are supported (for different purposes)
- The Web App is implemented using `ExtJS`, and fully custom Python "bindings"
- For its internal AuthN/Z, DIRAC understands certificates and proxies
  - VOMS Virtual Organization Membership Service) is effectively a hard DIRAC dependency

Technology trends

You authenticate with an external "Identity provider":



For authorization purposes you are using tokens everywhere:

Nicely documented) REST APIs are a de-facto standard:

```
# "get a tag" from github

curl -L \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer <YOUR-TOKEN>" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  https://api.github.com/repos/OWNER/REPO/git/tags/TAG_SHA
```

Dirac should follow these trends

What is the best way to keep up with these trends? Can we do it within the current framework?

Recommendations from WLCG, EGI, etc.

- VOMS Virtual Organization Membership Service) has been, for many years, a de-facto standard for **community management**
  - it issues VOMS proxies ("short" certificates)
  - Outside of WLCG and EGI, proxies are not a thing
- There are new Identity Providers delivering tokens instead of proxies

In this conference:

- WLCG transition from X.509 to Tokens: Progress and Outlook
- CMS Token Transition
- Fermilab's Transition to Token Authentication

Dirac needs to follow these recommendations

What is the best way to implement these recommendations? Can we do it within the current framework?

Minimum Requirements

Communities/Users requirements

Ease of use, including ease of access

Fast and responsive interfaces

Scalable and flexible

Administrator requirements

Ease of installation and update

Up-to-date documentation

Clear confguration

Ready-to-use dashboards

Developers and maintainers requirements

Easy to test (will make it easier to code), but also modern, fun, and accessible to new developers

Paramount requirement

We need to ensure business continuity

DIRAC challenges

- complex, with high entrance bar
- somewhat cumbersome deployment
- late on “standards”
  - No http services
  - No tokens
  - Old monitoring
- "old"-ish design (RPC, "cron" agents...)
- not very developer-friendly: rather un-appealing/confusing, especially for new (and young) developers
- multi-VO, but was not designed to do so since the beginning
- a custom interface is needed to interact with a running DIRAC instance
  - meaning that you need to install a DIRAC client for interacting with DIRAC

We decided that the best way of satisfying the requirements was to code a new Dirac

DiracX, the neXt DIRAC incarnation

What is DiracX?

- A cloud native app
- Multi-VO from the get-go
- Standards-based

Important
Still Dirac, in terms of functionalities.

DiracX Web API

Caution
What is on the right is the certification Web API, loaded live. Use with caution!

DIRAC Web APIs with FastAPI

- Nicely documented by Swagger
  - this is what you see on the right
- Follows the OPENAPI INITIATIVE specification, with the

  (python) client generated by AutoREST.

Dirac 0.1.0 OAS 3.0

/api/openapi.json

Authorize

.well-known

GET /.well-known/openid-configuration
Openid Configuration

GET /.well-known/dirac-metadata
Installation Metadata

auth

POST /api/auth/device
Initiate Device Flow

CLI Interactions

Logging in (using the `diracx cli`):

```
❯ dirac login gridpp
Logging in with scopes: ['vo:gridpp']
Now go to: https://diracx-cert.app.cern.ch/api/auth/device?user_code=XYZXYZXYZ
...Saved credentials to /home/fstagni/.cache/diracx/credentials.json
Login successful!
```

Submitting a job (using Python `requests`):

```
import requests

requests.post('https://diracx-cert.app.cern.ch/api/jobs/', headers={'accept': 'application/json', 'Authorization': 'Bea
```

Getting its status (using `curl`):

```
curl -X 'GET' \
  'https://diracx-cert.app.cern.ch/api/jobs/status?job_ids=8971' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbG...'  | jq
```

Select Virtual Organization

DiracX web

We are also rewriting the Web App from scratch.

Software stack:

- NextJS
- Material UI
- TypeScript

Caution
What is on the left is the certification WebApp, loaded live. Use with caution!

Deployments

Kubernetes - Standard to define a distributed system

- Separates infrastructure from applications
  - "Please IT department(/cloud provider) run this for me"

Helm gives the ability:

- to parameterise
- to distribute a kubernetes config

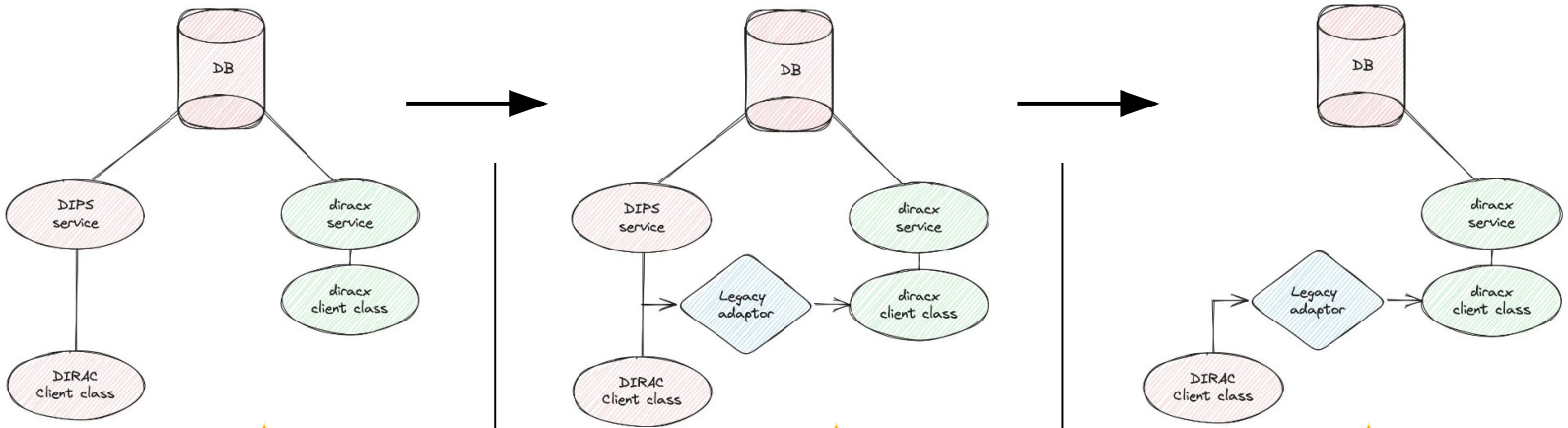
- DiracX Helm chart
  - If your institution provides a kubernetes service: use it
  - If you work with public clouds: use their container services
  - Alternatively, follow these k3s instructions
- Used for:
  - DiracX testing GitHub actions)
  - Local development instance
  - Running a demo instance
  - Running the test instance you saw in the previous slides
  - Soon: running production instances

"OK, but there are several communities using DIRAC right now. How do they migrate?"

- Some of you out there

Business continuity for DIRAC communities is our top priority

Services of DIRAC v9 and DiracX will need to live together for some time



1 DIRAC and DiracX share the databases

2 A legacy adaptor moves traffic from DIRAC to DiracX services

3 DIRAC services can be removed

Future action! (and extensions)

By now, we know that it is sometimes necessary to extend all Dirac(X) components

`DiracX aims to provide an easy way to do so.`

```
# entrypoints in pyproject.toml

[project.entry-points."diracx.db.sql"]
AuthDB = "diracx.db.sql:AuthDB"
JobDB = "<extension>.db.sql:ExtendedJobDB"
```

For DiracX and DiracX Web we already provide reference extensions

"You have shown tokens-based authorizations for DiracX. But the Grid still uses proxies. VOMS is alive!"

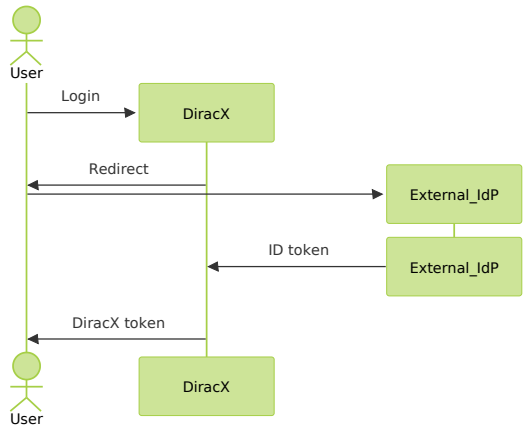
- Again, some of you out there

What are proxies and/or tokens needed for?

- **Identity (community membership)**: "in transition"
- **Submitting pilots**: The computing elements right now prefer the tokens
- **Data access**: at least in WLCG, proxies. One day, will be token
- **Verifying a user's identity** (internally to Dirac):
  - **DiracX** uses only tokens (link to security model)
  - **DIRAC** uses only X509 proxies and certificates to verify identities

More on proxies and tokens

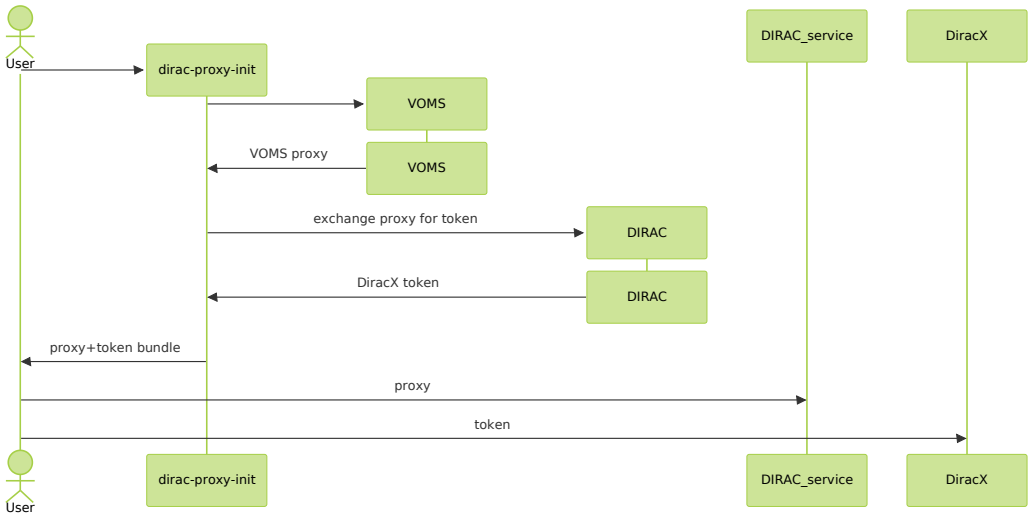
DiracX Authorization with "standard" Authorization Code Flow redirecting to IdP



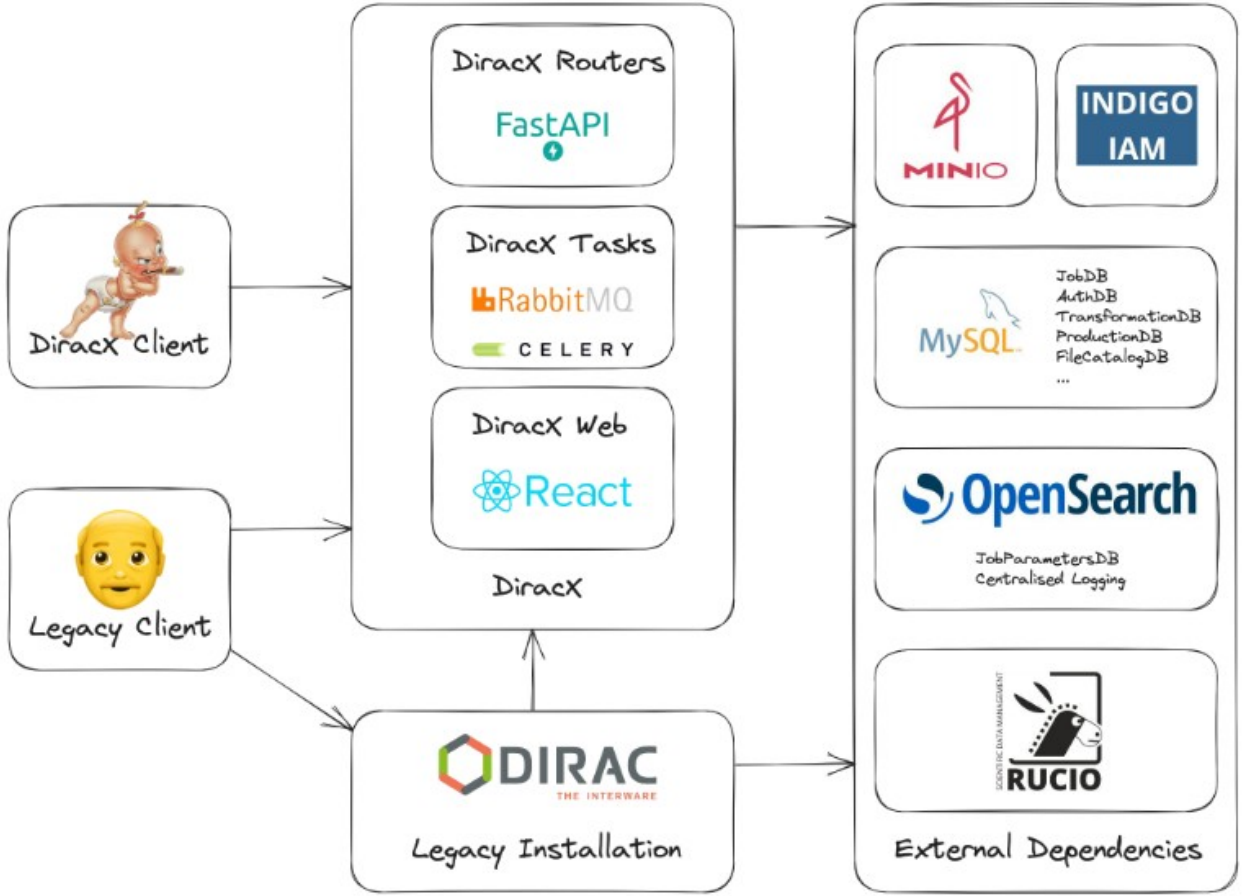
Note

DiracX delivers its own tokens, they are not the same tokens used for the Grid endpoints

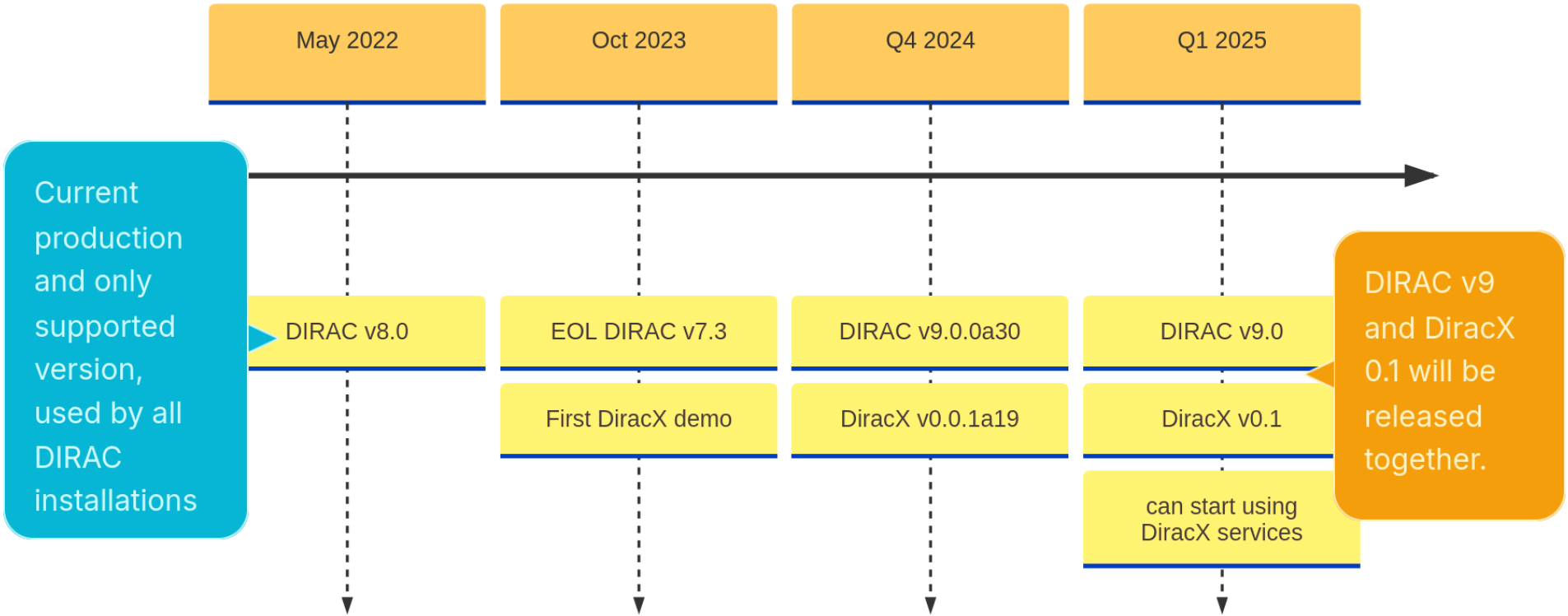
DIRAC DiracX: working with proxies and tokens



Architecture diagram



Versions


May 2022
Oct 2023
Q4 2024
Q1 2025
Current production and only supported version, used by all DIRAC installations
DIRAC v8.0
EOL DIRAC v7.3
First DiracX demo
DIRAC v9.0.0a30
DiracX v0.0.1a19
DIRAC v9.0
DiracX v0.1
can start using DiracX services
DIRAC v9 and DiracX 0.1 will be released together.

"I want to contribute"

The obvious ways:

- code (github.com/DIRACGrid)
- tests: (as you could see we have a somewhat open test deployment infrastructure). Try something out, and let us know!

Run the demo (on your laptop):

```
git clone https://github.com/DIRACGrid/diracx-charts
diracx-charts/run_demo.sh # this is run for each and every commit in Github Actions
```

Discuss:

- **mattermost**: https://mattermost.web.cern.ch/diracx/
- **meetings**: (almost) every week on Thursday morning CET
- **hackathons**: we have been doing 2-days DiracX hackathons every quarter, at CERN
  - Next one in January
  - Followed by a Dirac&Rucio mini-workshop and hackathon
- **workshops**: once per year, more or less

Summary

- DiracX is "the neXt Dirac incarnation", ensuring the future of the widely used Dirac
  - We are rewriting the code, but it is still Dirac that you love!
- DiracX will ease the interoperability with Rucio and/or dask and/or any other tool out there
  - DiracX will still have the Data Management part, but its Workload Management functionalities will come first
- The first DiracX release will soon be here
  - It will live together with DIRAC v9 for a while, until it will replace it completely

People

DiracX is an idea of

Chris Burr CERN, LHCb
Christophe Haen CERN, LHCb

Current Developers, maintainers, supporters

Alexandre Boyer CERN, LHCb
Natthan Piggoux LUPM FR , CTA
Cedric Serfon Brookhaven National Laboratory US , Belle2
Ryunosuke O'Neil CERN, LHCb
Jorge Lisa Laborda Univ. of Valencia and CSIC ES , LHCb
Daniela Bauer Imperial college UK , GridPP
Simon Fayer Imperial college UK , GridPP
Janusz Martyniak Imperial college UK , GridPP
Bertrand Rigaud IN2P3 FR
Luisa Arrabito LUPM FR , CTA
Xiaomei Zhang Beijing, Inst. High Energy Phys. CN , Juno
André Sailer CERN

QR codes for your fun →

or just click here (for DiracX web) and here (for the Web API docs)

WebApp:

WebAPI

Backup

Q/A

- I am using Rucio|dask|another_tool}. I could use DiracX as WMS but do not want to fiddle with DIRAC

  It will probably be possible, but we do not know when.

- What is in a DiracX token (is it "special" ?

  It carries the `dirac_properties` (which are the same as in current DIRAC

- What did you use to make these slides?

  slidev with neversink theme. Diagrams with mermaid

Testing

- we use Github Actions "massively"
- our Integration tests create a "grid-in-a-box":
  - run DIRAC and DiracX servers, including databases
  - run ancillary services (e.g. IdP, CA
  - authenticate, submit pilots, match and run jobs, upload files, etc