Status and Future of DIRAC and DiracX for CEPC

Xiaomei Zhang

2026.10.24

CEPC workshop, Shanghai

Content

01 Introduction of DIRAC and DiracX

02 Status of IHEPDIRAC for CEPC

03 DIRAC status and DiracX Time line

04 The architecture today

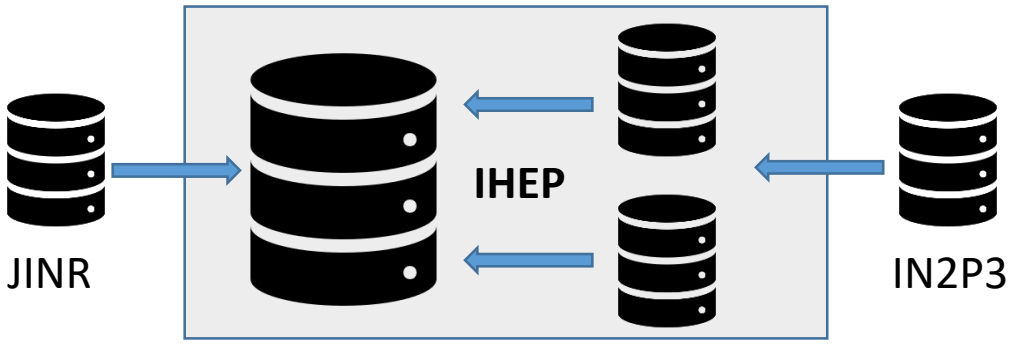
05 DiracX development status

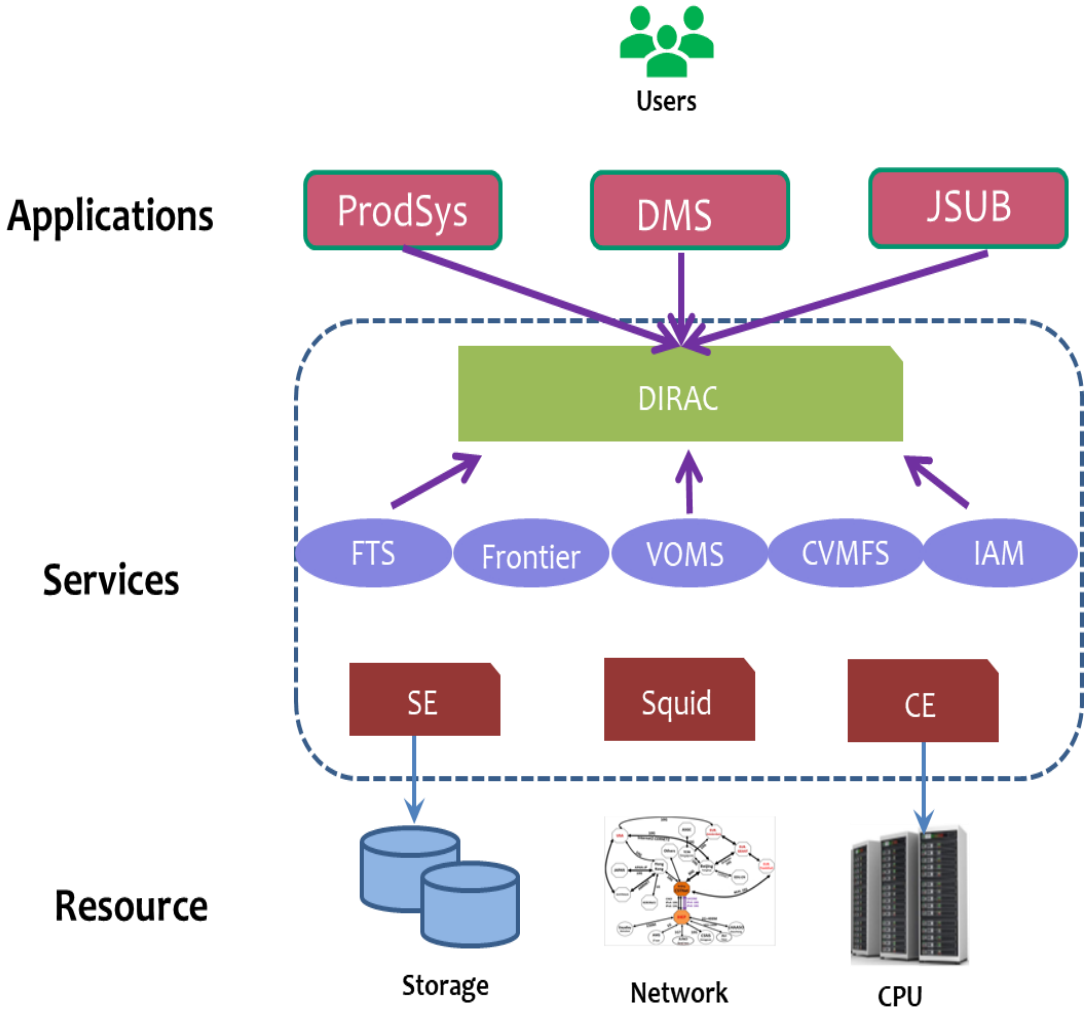
DIRAC: a framework for distributed computing

- Runs **large-scale workflows** on heterogeneous, distributed compute and storage.
- Four pillars: **WMS** (pilots and jobs), **DMS** (catalogues, transfers), **Transformation System** (productions), **RSS** (resources).
- Multi-VO since 2008: **one installation serves several communities**.
  - IHEPDIRAC set-up in 2012, now supporting: BESIII, JUNO, **CEPC,** Herd
- Extensible: LHCb, Belle II, CTAO, JUNO and others add their own layers.

IHEPDIRAC in production

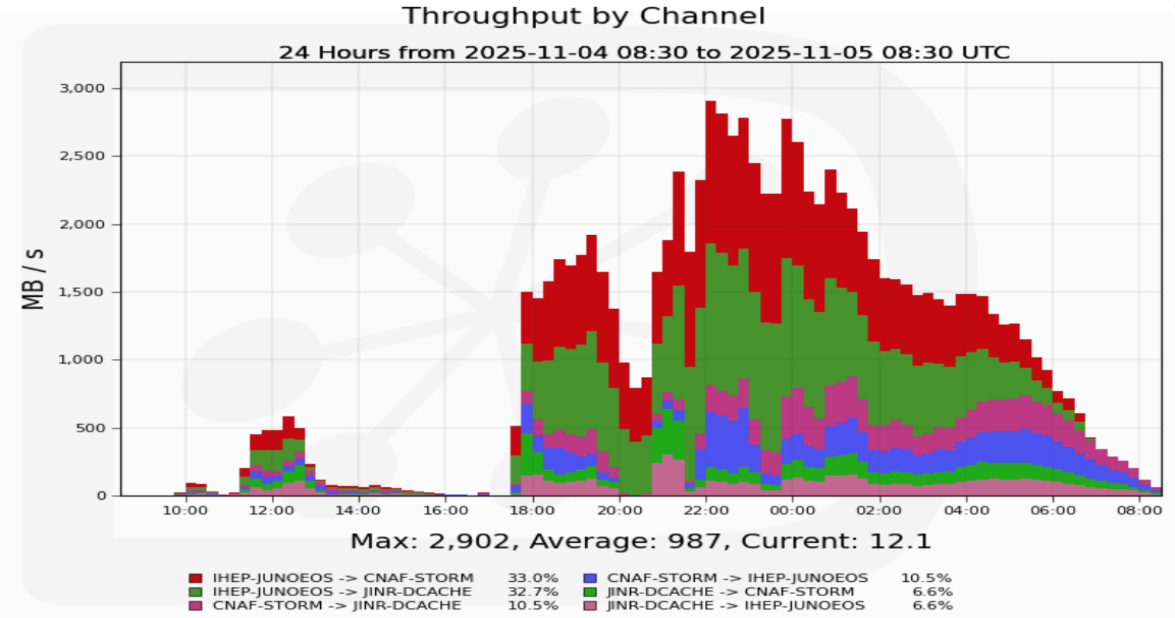
- Current deployment
  - 1 physical-hosted master server at IHEP
  - Four slave servers (IHEP, IN2P3, JINR) for load balance and failover
- The distributed computing system build on IHEPDIRAC has been verified through JUNO in its first year production

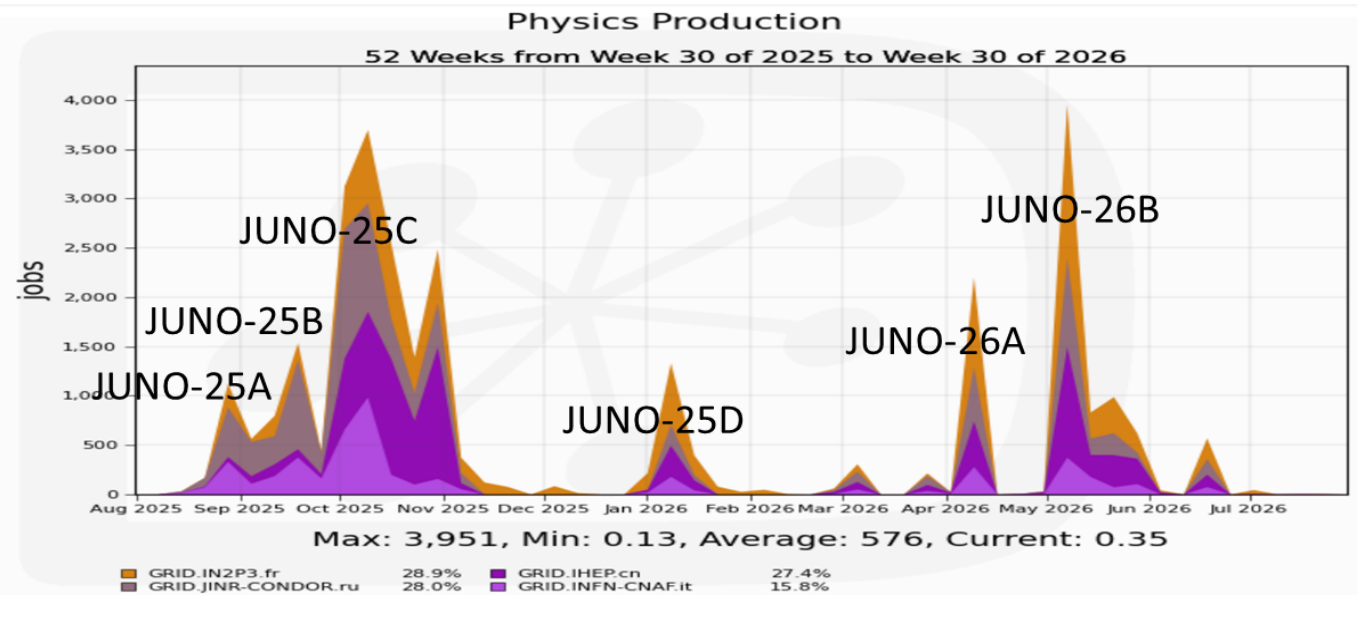




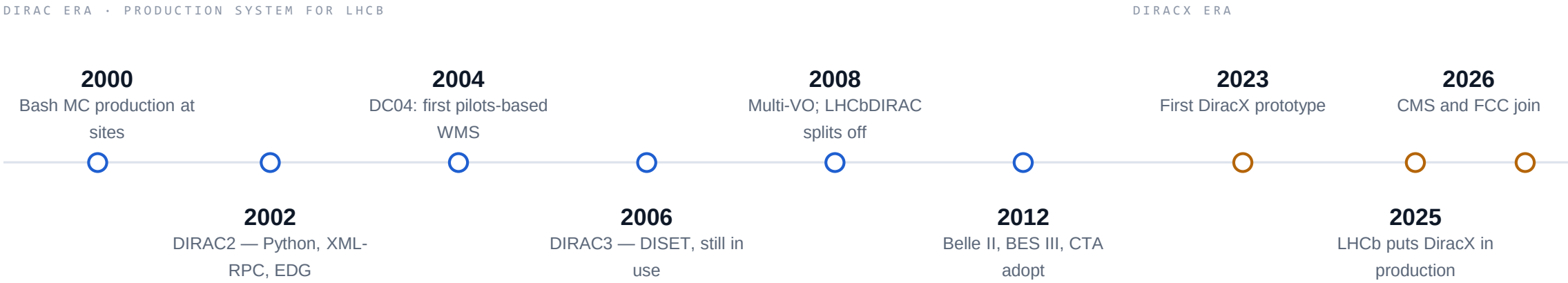
JUNO production in first year with IHEPDIRAC

- 6 large-scale physics production campaigns in first year
  - Peak concurrency: ~22,000 jobs across four sites
- Real data ~7PB, > 6 Million files, Combined real + MC dataset: ~34 million files in total
- Data transfer ~ 19PB/year, peak throughput up to 30 Gb/s





DiracX: the same problem, a new spine

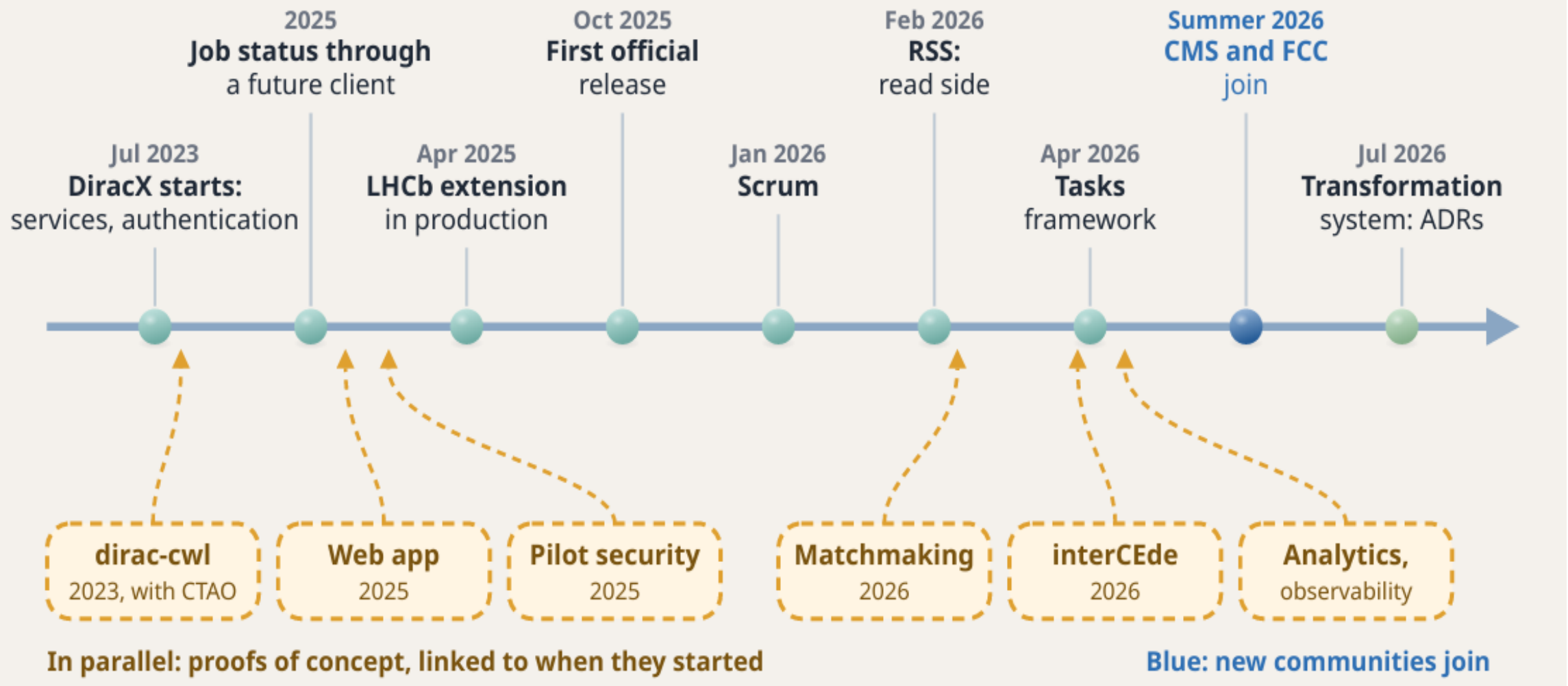


- DIRAC has been used for twenty years and it is time to redesign with modern technologies and meet future experiments' requirements
- DiracX is the first full change of spine since 2006-07 rewrite
- **A reimplementation, not a port**
- Status: v0.6.1, 23 releases since alpha

DIRAC: keep in safe and stable

- **Three lines are maintained**: v8.0.85, v9.0.27 and v9.1.21
  - Only security fixes, plus performance fixes when needed.
  - New features go to DiracX; DIRAC is only patched so that DiracX can progress.
- There is **no end-of-life cliff announced for v9** — but v8 users were already told a year ago to target v9.0.
- **IHEP now is in v8.0.58, and plan to migrate to v9.0 this year.**

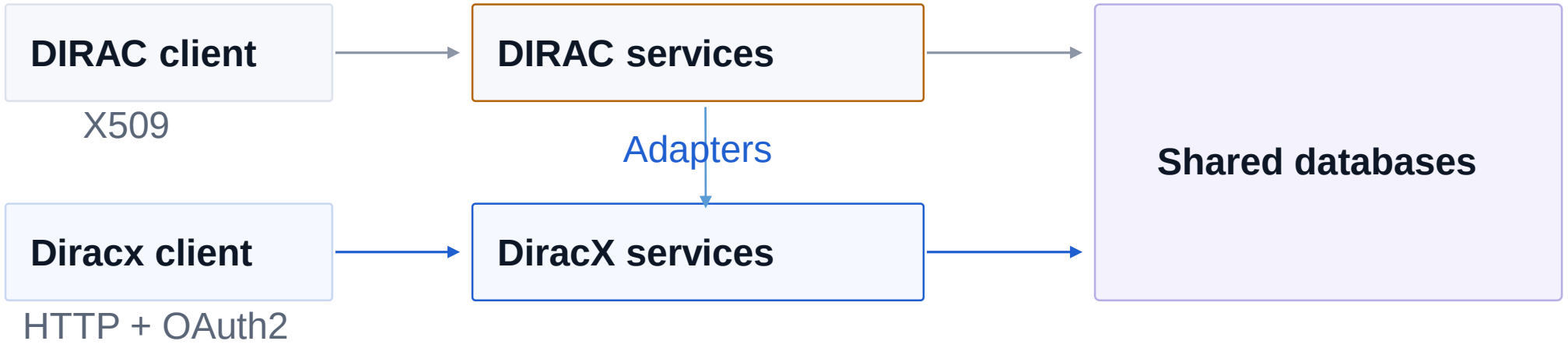
DiracX Timeline



Timeline adjusted with CMS and FCC joined

- Summer 2026: **CMS and FCC join**. CMS has **no DIRAC installation** and runs its own workload management.
- They need a **new model, decoupled from DIRAC** — their requirements do not fit the data model shared by both stacks.
- Last year DiracX planned to be ready in one year
- **Now** become a reimplementation with a new data model, only priority and no date was promised for the remaining components

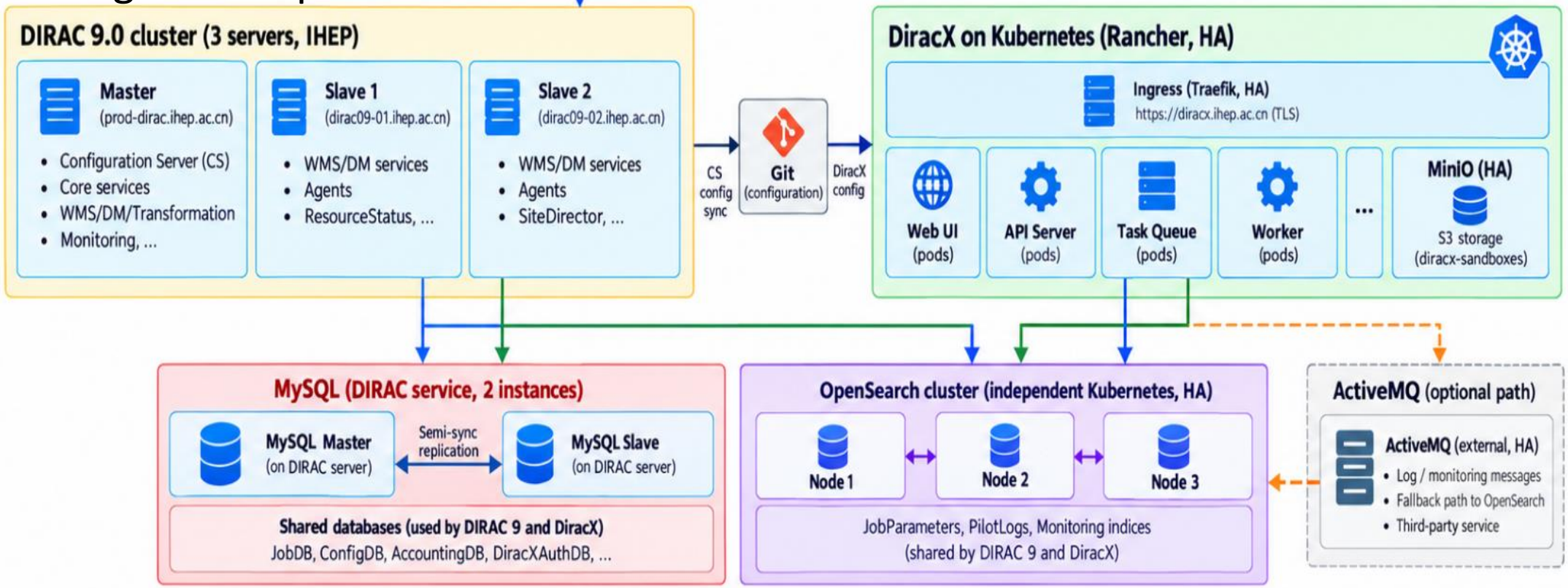
Today's architecture: two stack, one state



- To ensure a smooth, transparent migration for users
  - DIRAC stack and the new DiracX stack run **in parallel until DiracX is completed**
  - They **share the same underlying database and data state**
- When new components available in DiracX, experiments decide when to test and switch
  - DIRAC client can call either stack with Adapters
  - No data migration outside shared databases

IHEP DIRAC 9.0 preproduction deployment

- DIRAC 9.0.26 is in similar set-up as DIRAC 8.0
- DiracX 0.6 deployed on K8s managed by Rancher+Havester
  - Shares MySQL, OpenSearch and CS
- Targeted for production roll-out in November



Changes known in DiracX so far

01 · CLIENTS

Future clients

DiracX has its own client

DIRAC: x509

DiracX: token

02 · DATA MODEL

The spine

Workgraph, transformations, parcels with new tables and new logic.

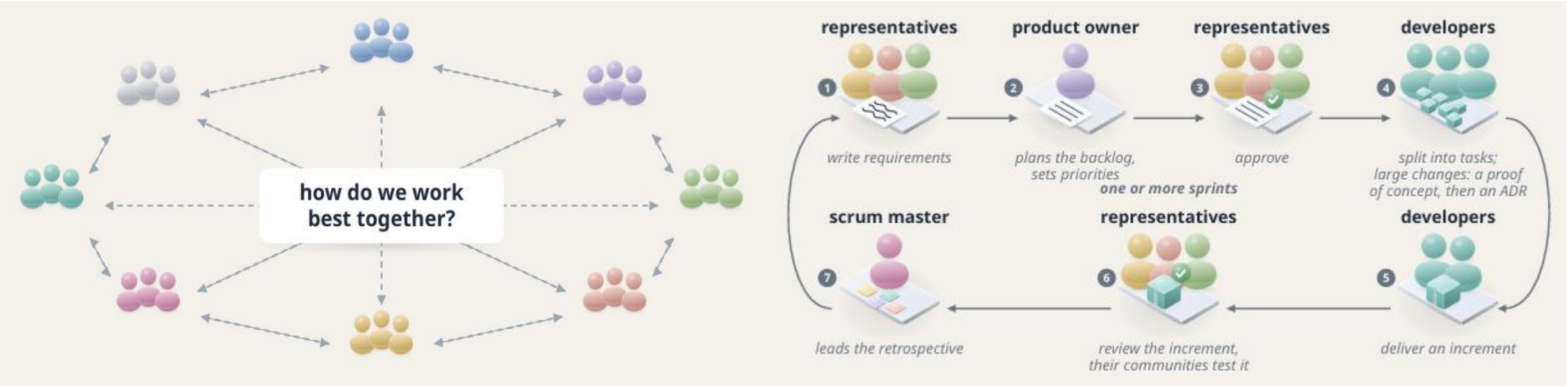
03 · PROTOCOLS

Standards

Systems internal to DIRAC give way to OIDC, S3 and OpenTelemetry.

DIRAC CONCEPT	DIRACX CONCEPT
Production	**Workgraph** — a DAG of transformations, created in one go
WMS (Pilot, CE….)	**Compute backend** — native pilots, HPC, HTCondor…..
Monitoring system	**OpenTelemetry** — metrics, traces, logs

DiracX development: use scrum for organization



- 36 people contributed since the last workshop, 10 to 18 in any given month. It is important to work together efficiently
- Scrum since 21 January 2026 is a milestone: two-week sprints; planning and refinement in one weekly meeting; a monthly meeting with the representatives (DOps); hackathons.
- Large changes start with a proof of concept, then an ADR.

AI in developments: An amplifier, not a shortcut

- **Code got cheap** → spikes and proofs of concept are easier to try
- **Deciding what to build, and checking that it was built, did not get cheaper.**
- **Specification and review become bottlenecks**
- **Prepare for AI to involve** → A clear ADR with a scenario test



The first ADR: Transformation

001	**Tasks**	The execution layer for actions and hooks.
002	**Overview**	The workgraph model
003	**Compute backends**	Where a parcel actually runs.
004	**Schema**	The tables the new model needs.
005	**State machines**	States and the transitions that are legal.
006	**Extensions**	Where a community plugs in its own pieces.
007	**CWL**	The document format for all of it.
008	**Data management**	Requests: copy, remove, stage.
009	**Counters**	Exact accounting for every state.

WHERE IT STANDS

- **Design, not implementation.**
- The ADR in review, with an explanation and an interactive playground.

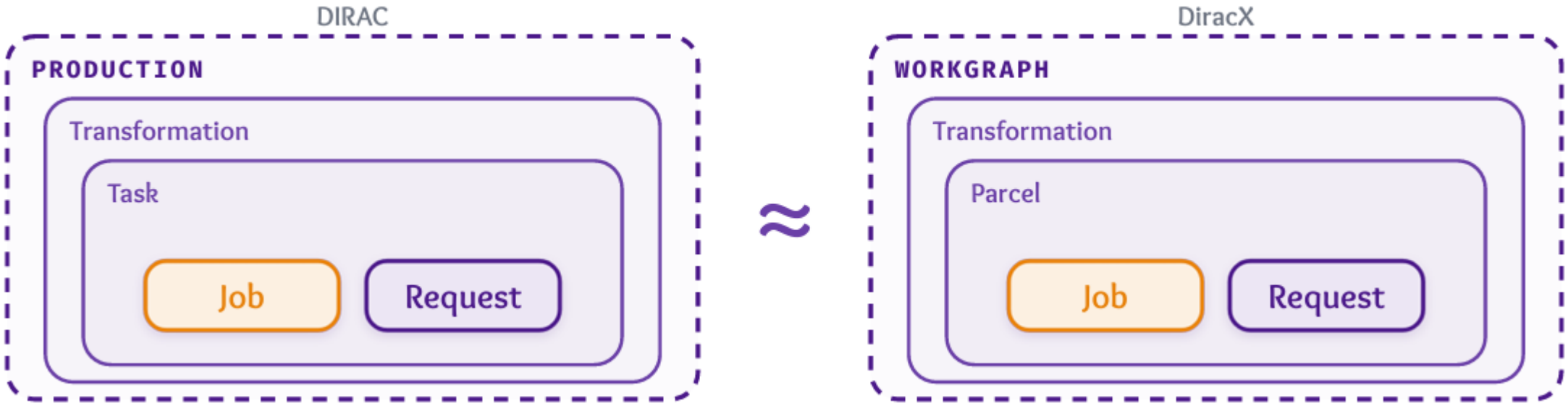
WHAT COMES NEXT

- Operations and debugging in **CWL** with experiment real use cases.
  - To see if missing cases
- Deploying software — still to be designed.

Burr & Haen, “The Transformation System ADRs”; the DX-ADR series (001–009); Boyer, roadmap slide 23.

Design Transformation in a common way

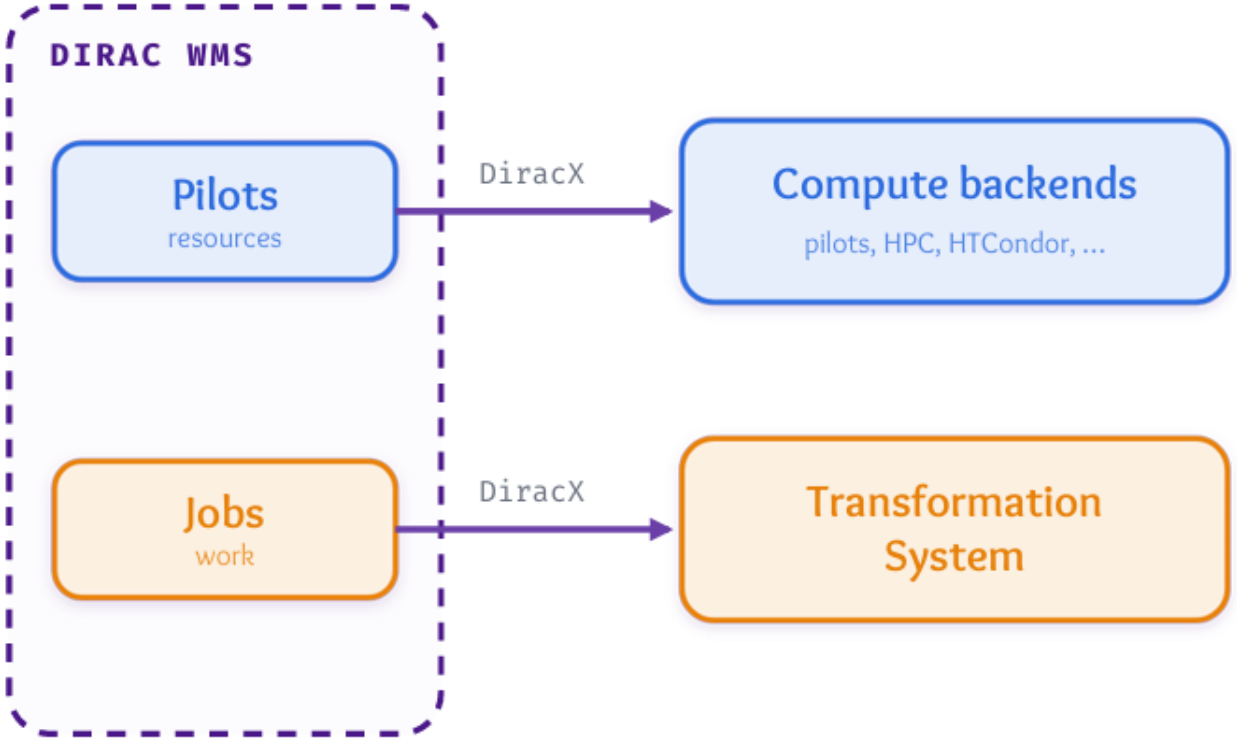
- **Workgraph** describes a DAG (directed acyclic graph) of workflows **in CWL**
- **Feeder** tops up the input pool from the experiment's metadata catalogue
- **Packer** groups pooled inputs into **parcels**
- **Dispatcher** hands each parcel to a compute or data **backend**, which executes it as a job or a request.



Workload Management System (WMS)

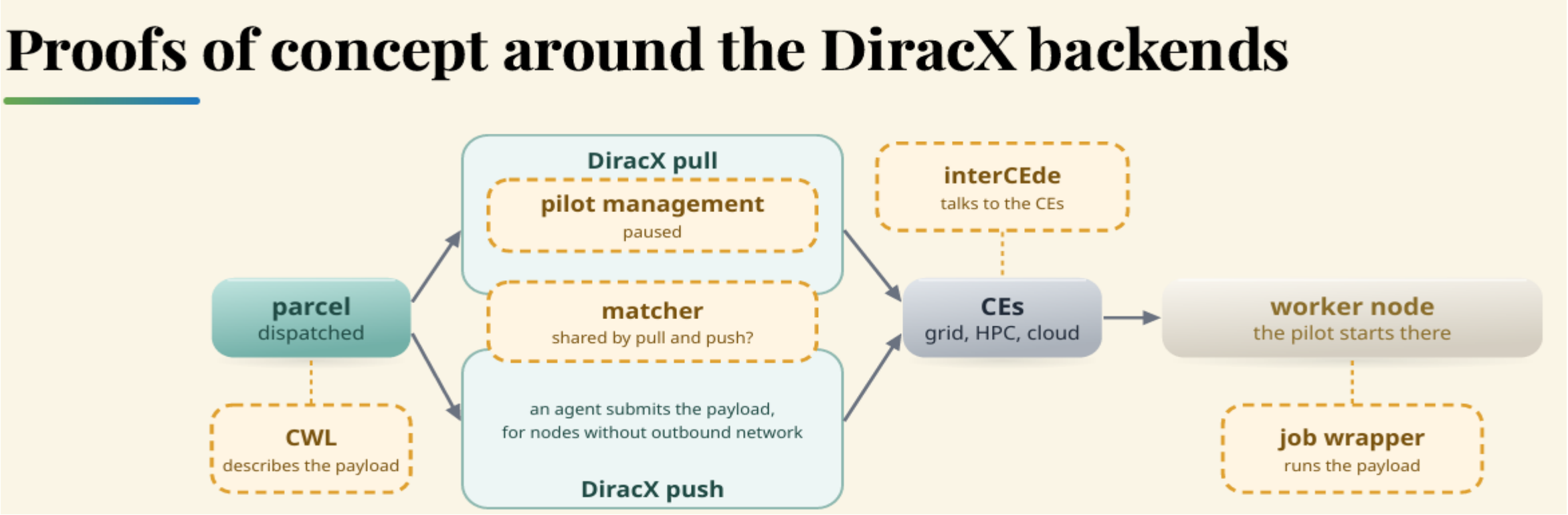
> jobs, transformation and productions are the same
>
> — The future of DIRAC

- Dirac
  - Pilots: getting hold of resources
  - Jobs: describing the work to run on them
- DiracX pulls them apart:
  - Pilots → Compute Backends
  - Jobs → Transformation System



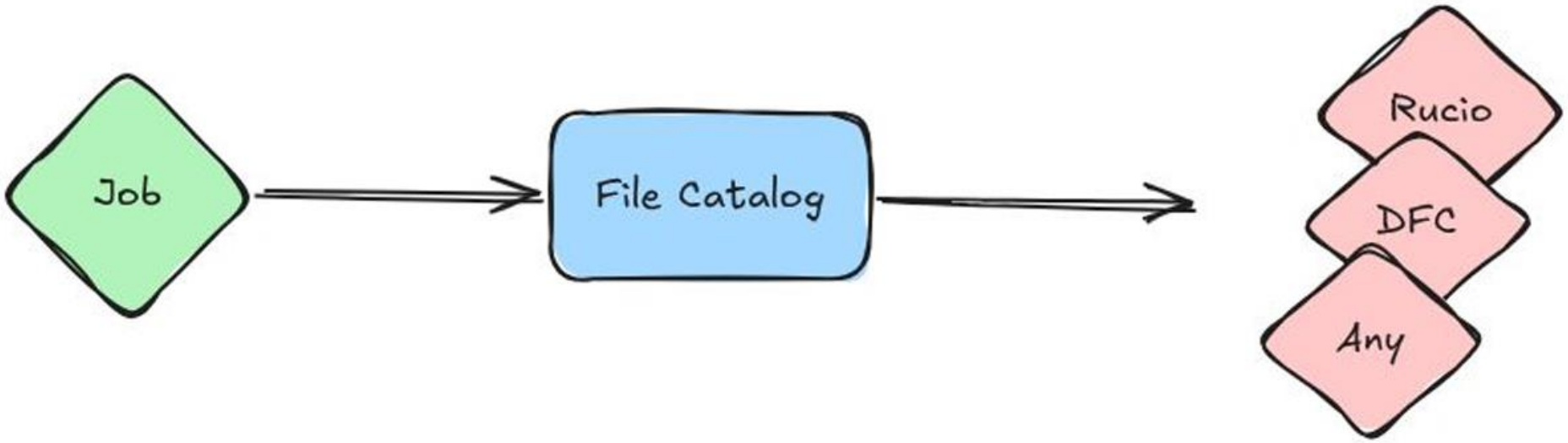
WMS in Proofs of Concept

- CWL is used to describe the payload
- Pilot, matcher, job wrapper in design
- InterCEde is introduced as a common layer to talk to CEs



DMS in collecting use cases

- Rucio will be included into File Catalog
- DFC is still there, an important part of File Catalogue



New Analytics system in proposal

- DIRAC monitoring and Accounting will be taken place by OLAP + OLTP
  - OLAP (Online Transaction Processing) focus on “Analytics, observability”
  - OLTP (Online Transaction Processing) handles "Data Collection" and real-time transaction recording
- **Proposed Stack: Parquet + S3 + DuckDB + DuckLake**, with Grafana for dashboards.
  - **Raw Parquet is the source of truth** and is kept forever
  - **Two access modes:** power users query the lakehouse directly with DuckDB; everyone else goes through the analytics API behind Grafana.
- More flexible and powerful than DIRAC, also need more efforts from experiments

Summary

- IHEPDIRAC, which supports CEPC and is validated by JUNO, is built on DIRAC.
- DIRAC is staying stable as DiracX evolves.
- This dual-architecture ensures a smooth DIRAC-to-DiracX migration.
- As DiracX redesigns its components in a more common way, IHEPDIRAC will need parallel rewrites to keep pace.

Backup

How the system evolve? One component at a time

Both run

DIRAC and DiracX operate in parallel on the same databases.

Pick a component

You choose when to switch it, usually during a shutdown.

Test with CWL

Your own workflows, in CWL, against the increment.

Keep or roll back

The extension still works on DIRAC while you move.

- **No flag day**: DIRAC keeps running with security and performance fixes for as long as you need it.
- Increments are shown at **Data Operation Meeting** whenever a sprint delivers one — installations test them with their own workflows.
- The CWL document is the same for local tests, user jobs and productions: that is what makes the move testable.

Design principle: standard protocols

- DIRAC: everything with one set-up
- DiracX comes up with standard protocols, More flexibility, but more responsibility and complexity
- One project, many contexts: a small installation does not need a cutting-edge stack; a large one plugs into its own.
- Databases are the exception to the shipped default — running MySQL or OpenSearch from the charts is not supported in production.
- Kubernetes is the default path; non-Kubernetes deployments are an open topic in this workshop.
- The same instinct as DIRAC's minimal footprint: depend on standards, not on one product.

FOR	PROTOCOL	IN THE CHARTS	OR YOURS
identity	**OIDC**	Dex; INDIGO IAM optional	your SSO or IAM
sandboxes	**S3**	SeaweedFS, optional	your site's S3 service
metrics, traces	**OpenTelemetry**	collector, Jaeger, Grafana, optional	your monitoring
state, job parameters	**SQL, OpenSearch**	MySQL, OpenSearch: **tests and demos only**	your IT's databases: **required in production**

DiracX development: Procedures

- **Large changes:** a proof of concept, then an ADR that the other developers approve. **Public:** ADRs and roadmap. **Private:** the board. **Principle:** individuals and interactions over processes and tools.

1

Requirements

Community representatives write them down.

2

Backlog

The product owner sets priorities and plans.

3

Tasks

Work is split; large changes get a proof of concept.

4

Sprints

One or more two-week sprints; refinement weekly.

5

Increment

Something an installation can actually test.

6

Review

Representatives test it with their communities.

7

Retrospective

The scrum master leads; the process adapts.

CADENCE

Two weeks

Sprints of two weeks; planning and refinement in one weekly meeting.