Lessons learned (so far)

Getting Rucio and DIRAC deployed using K8s

Volodymyr Savchenko, CTAO DPPS Integration, EPFL
Volodymyr.savchenko@epfl.ch

Goals

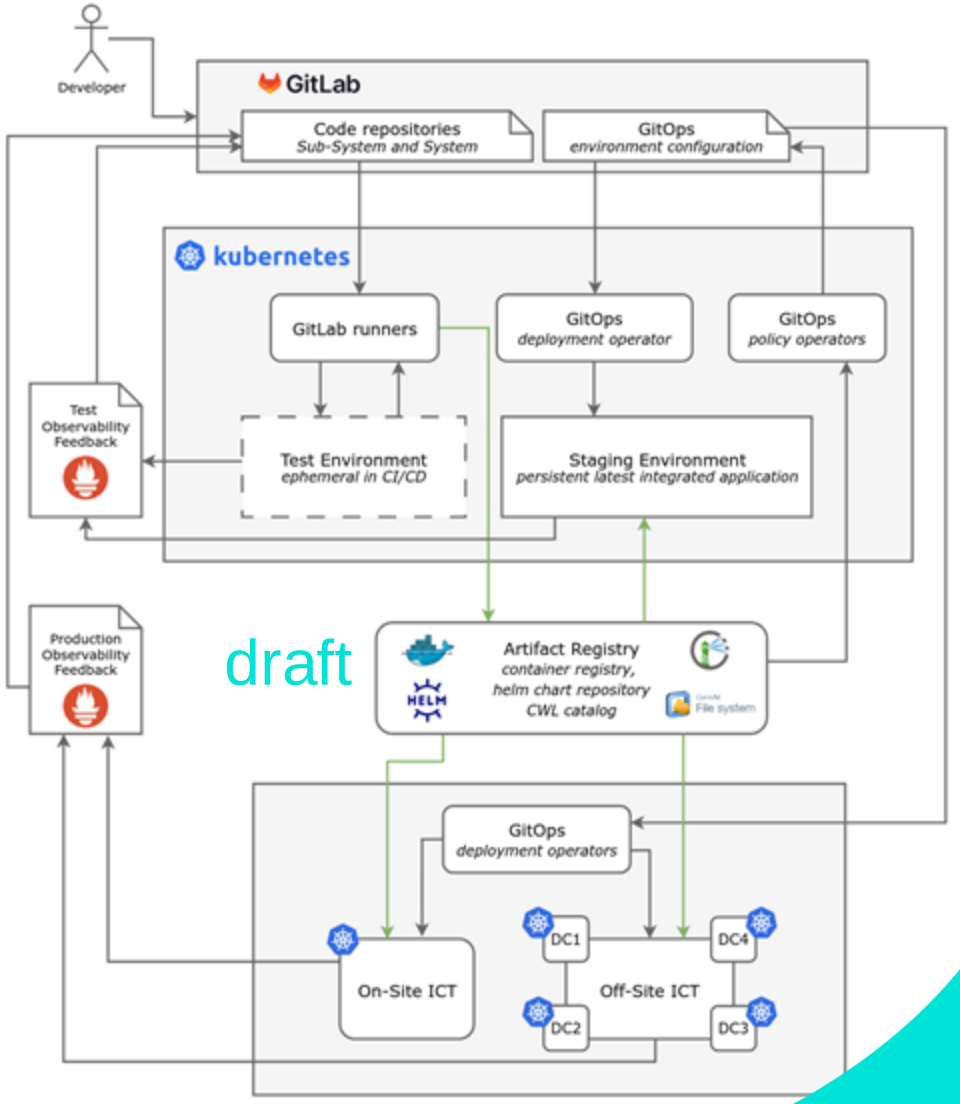
Building completely **new CTAO DPPS deployment, first releases.** Legacy CTAOC deployment will be gradually phased-out.

Need to deploy DPPS (see previous talk of Max) on 6 DCs (4 Off-Site and 2 On-Site)

- Reliably perform upgrades of a multi-component DPPS system
- Centralize observability (monitoring, alerts, tracing)
- Support failover between DCs

Need **reproducible deployment** in different environments:

- **local developer** hosts and private dev clusters: direct full control of the developer: laptop, personal cluster
- **Gitlab CI**: refence environment, created with **kind engine** from scratch, performs **requirement verification**
- **"site" deployments** with **GitOps**:
  - staging: on test infrastructure
  - pre-prod: on DCs
  - prod: on DCs



DPPS Helm Chart architecture

We decide to setup all software as one helm chart with many subcharts:

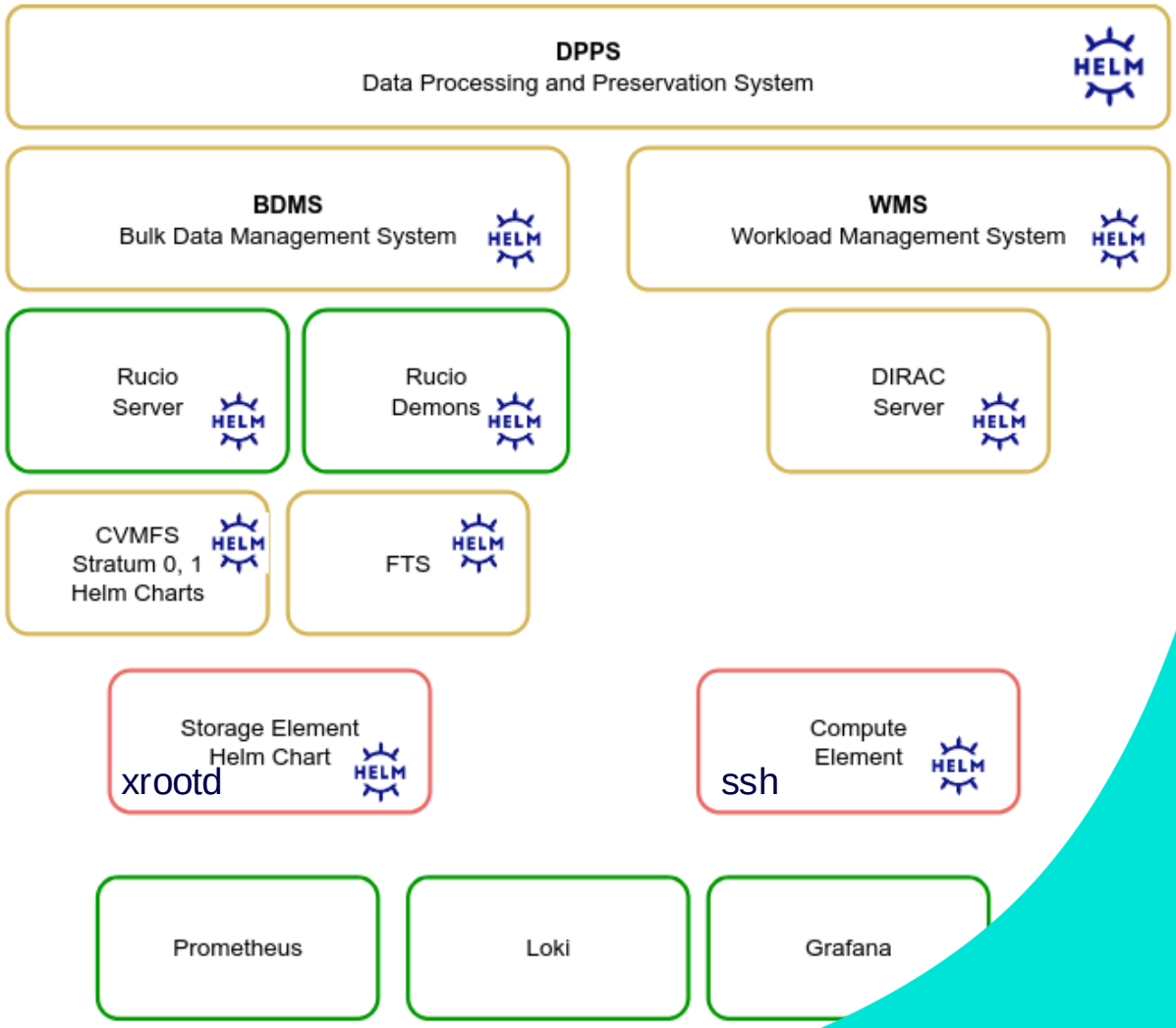
- **Bootstrap** as **jobs**, **init containers** (maybe pre-install hooks later)
- **Helm Tests**, to be used in CI and production liveliness.
- **Test fixtures** included for Storage (SE), Compute (CE), SE, CE, DBs but **configurable interface** to external dependencies.

What we have in current pre-DPPS CTAO prod:

- **DIRAC 8** and **CTADIRAC** – the verified version, and we need it in the DPPS Rel0.0

Available **off-the-shelf** charts:

- Rucio charts
- DiracX charts (not fully tested)
- Supporting: Observability, MQ



concerns

Stateful components

Bootstap, Upgrade, Backup, Restore

Reproducible deployment needs to respect **stateful components** of DPPS.

- DIRAC mysql
- Rucio postgres
- FTS mysql
- Content of SEs

They can be all handled with volumes with *Read(Write/Only)Once volumes*.

The goal is to get **"helm upgrade --install"** do all that's needed (partially achieved for upgrade):

- Deployment on an empty cluster:
  - Bootstrap and configure, either "default" configuration or stored backup state
- DB Migrations
  - Available in Rucio
  - Missing in DIRAC, FTS
- Store and restore (sub)system state Backups, replications, failover
- SEs (and possibly stateful CEs)

concerns

Internal and External Interfaces

Internal

- All internal **interfaces** in **component helm charts as values**
- Currently, credentials (e.g. DB) in Rucio, DIRAC, controlled by values, which means that in DPPS chart some values are repeated in each subchart.
- Seems to be better to rely on secrets, and vault – needs modification of Rucio charts (reference to secrets still defined in **values**).

External

- very likely **SE, CE, IAM**, likely some **DBs**, and possibly **FTS, CVMFS**
- All **external outbound interfaces** specified in **DPPS helm chart values, specially annotated**
- In test setup all external components have fully **functional lightweight versions** in DPPS chart.

```
wms:
  cert-generator-grid:
    enabled: false

  # This configuration of interface between WMS and BDMS.
  # The ConfigMap is created by BDMS chart. The name should match the name of the ConfigMap in BDMS chart given the release name.
  rucio:
    enabled: true
    rucioConfig: "dpps-bdms-rucio-config"
```

Identity management, A&A, Ingresses, LoadBalancers

DPPS ingresses and their auth are **"inbound" interfaces**.

Internal communication is managed through k8s **Services.**
Auth with **private CA, host and grid user certificates.**

Used two alternative certificate generation mechanisms:

- Generate on installation as configured in DPPS values, with **pre-install hook**
- Use "**cert-manager**" model similar to that supporting ACME for **LetsEncrypt**

Right now, all external services are LoadBalancers, and applications own their certificates.

TODO:

- Need to make kinds more configurable, to allow public addresses in needed cases.
- Transition to OAuth2 tokens is needed for all components.

concerns

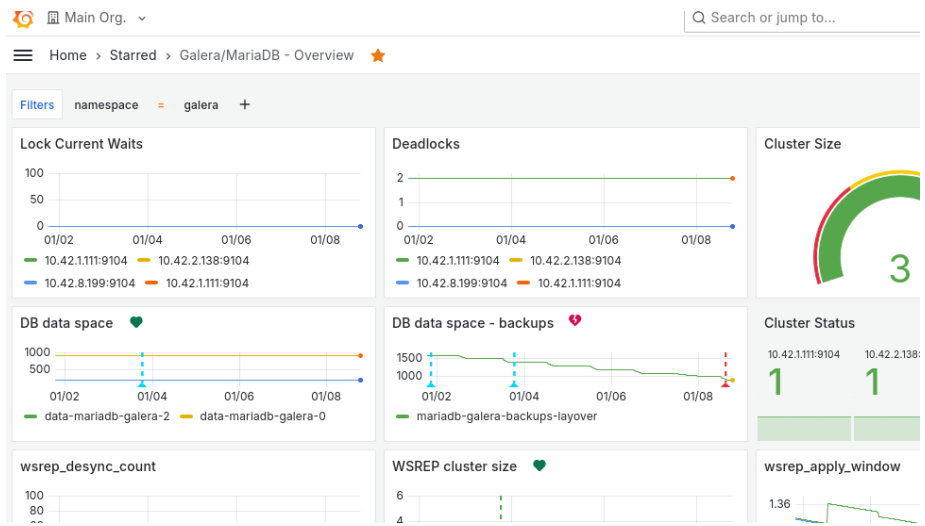
Observability

Monitoring, alerting, tracing

We aim to get monitoring verified in CI, collecting and verifying observability from every test (UC) execution.

Applications will have to provide this information in suitable form.

- Pushed container Logs (ingested to loki)
  - Available from all components, tested in CI
- Scraped metrics (by prometheus)
  - Available in **Rucio charts** only
- Alerts
  - In Grafana? Ingest configuration from systems?
- Tracing: Sentry? Grafana tempo?



Build upon experience of legacy DIRAC8 operations

concerns

Security

Critical in current threat environment

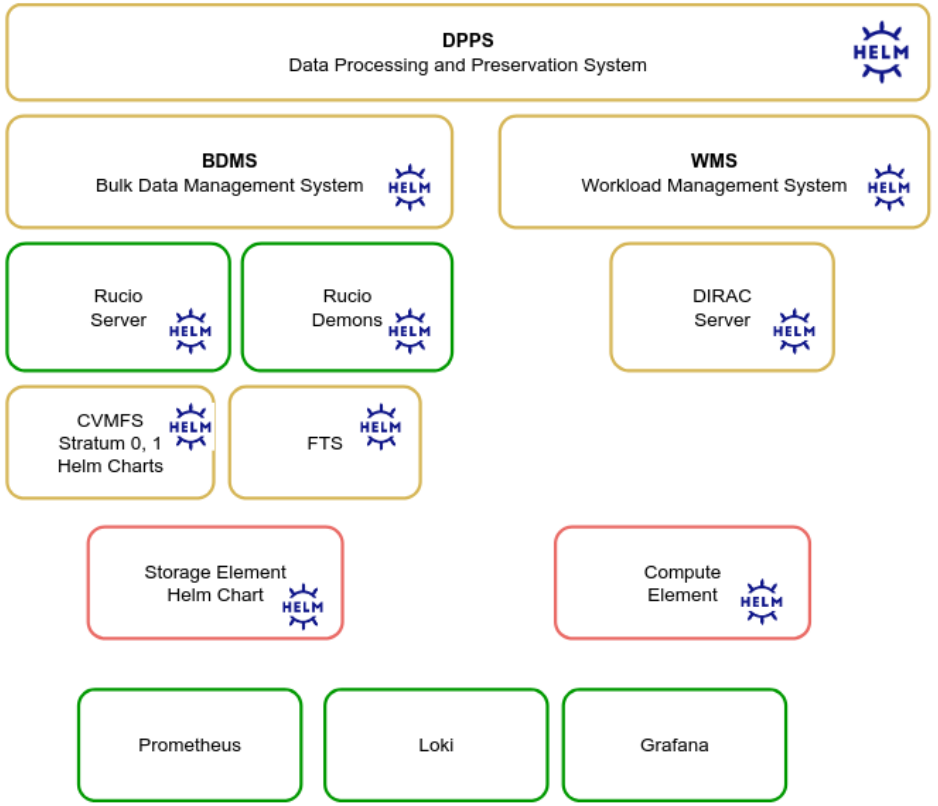
- **Observability** – detect issues well and fast
- **DevOps**: fix and deploy bugfixes fast
- **Deployment Provenance**: determine affected systems and software
- **Reproducible deployment**: quicky recreate from append-only backups

Summary and Next Steps

First **"test" DPPS helm chart,** ready deployment in dev, CI, staging , **release DPPS v0.0 Feb 27:** Rucio 35, DIRAC 8, x509 (fake), sets up **requirement verification.**

Next step will be **DiracX + DIRAC9 + Rucio + etc, all in k8s**

- Rucio:
  - use more and verify applicability of metrics.
  - "Internal" interfaces to DIRAC, shared secrets, **contribute?**
  - more work on upgrades in hooks
- DIRAC:
  - Customize somewhat more DIRAC8/9 deployment
  - Move to DiracX **contribute**?
  - more work on upgrades in hooks
- Multi-cluster deployments
  - Try failover/restore
  - Try service mesh
- Plan to move to OAuth2 tokens, host x509 LetsEncrypt
  - Rucio server can work with Ingress and OIDC
  - DIRAC 8 supports CE, SE interfaces but not other access
  - Will use network policies for DevSecOps but HTTPS termination services – no certificates for every pod
  - Tentatively planned for later this year Rel 0.1




```
NAME                                                READY   STATUS
dpps-bdms-bootstrap-rucio-bqn97                     0/1     Completed
dpps-bdms-configure-test-rucio-7q87r                0/1     Completed
dpps-cert-generator-grid-generate-certificates-dsxhg 0/1    Completed
dpps-conveyor-finisher-5cdb9dd784-l287k             1/1     Running
dpps-conveyor-poller-846797bb79-mqlh8               1/1     Running
dpps-conveyor-receiver-6647677f55-hgvmx             1/1     Running
dpps-conveyor-submitter-74957f8786-lwqfz            1/1     Running
dpps-dirac-server-7f7996fc6f-6xj2r                  2/2     Running
dpps-fts-84cbb59b46-lgsbs                           3/3     Running
dpps-judge-evaluator-68dbbb694d-qqfml               1/1     Running
dpps-postgresql-0                                   1/1     Running
dpps-rucio-server-55d6cfb84b-hnhln                  1/1     Running
dpps-wms-dirac-ce-568c5cdf7-hbwp9                   1/1     Running
dpps-wms-dirac-client-778d6bc6d8-pgnpq              1/1     Running
dpps-wms-mysql-d757795b4-q8674                      1/1     Running
rucio-storage-1-8567bb8744-zmlg9                    1/1     Running
rucio-storage-2-59dcd958d8-s5vjc                    1/1     Running
rucio-storage-3-f8985bfbd-zqzn9                     1/1     Running
testkit-7547474fd8-hfvvp                            1/1     Running
```

Artifact registry: Harbor

- CVMFS is used as last stage of the artifact distribution chain. It is itself part of DPPS software.