FTS in a K8s cluster with S3-only backends and OIDC

Michel Jouvin – michel.jouvin@ijclab.in2p3.fr

XRootD and FTS workshop

14-18 September 2026, Lyon

MesoNET project

- A 7-year French project to build a national federation of regional HPC/HTC resources
  - Called « mesocentre » in French jargon
  - Access to computing resources and a global storage federation
  - Global authentication build around INDIGO IAM (OIDC/OAuth)
- Global storage federation
  - Mainstream service based on iRods (as there is some expertise/experience in France)
  - PoC service based on Rucio: better scalability, better OIDC support, S3 backends, wider adoption
- PoC objectives
  - Get experience with Rucio
  - Identify potential gaps for supporting mesocentre typical storage systems (NFS/LUSTRE…)

Rucio requirement: FTS

- Rucio manages the central catalog and the access to the storage backends
  - Import, export, permissions, namespace management, replication rules…
  - Grid protocols preferred, S3 supported…
- Rucio delegates to FTS the management of inter-site transfers
  - Source of some operational complexity: FTS user is the Rucio server, not the end user
  - Backends receive requests through 2 different paths with different tokens
- No production challenges as it is a PoC
  - Focus is on functional tests, in particular integration of S3 backends in a OIDC world
  - MesoNET will never face a data volume/size challenge similar to big physics experiments

Deployment choices

- Several (OpenStack) clouds in MesoNET: use them to deploy Rucio…
  - Rucio central services + FTS
  - Avoid the need of dedicated hardware
- … for easier maintainability and greater scalability, use Kubernetes!
  - Allows flexible DevOps approach, e.g. automatic deployment of versioned configuration changes with tools like ArgoCD or FluxCD
  - Declarative configuration (based on YAML files)
  - Scalability is a core feature of Kubernetes: ability to scale out individual services, spread them on several workers…
- Requirement: availability of container images for the services
  - Rucio: existing and documented, even if generally not used in production (we are a PoC!)
  - FTS: built for every release, not very much documented… but Mihai's help compensates!

FTS container images

- 3 images
  - fts3: the core FTS services: fts_server, fts_token, ftstokenhousekeeperd, fts-optimizer, fts_qos, fts_activemq
  - fts-monitoring: the operation monitoring service (provides a Web interface with statistics on the operations queued/done)
  - fts-rest-flask: FTS front-end (used by *fts* commands)
- Customized version of these images used
  - Minor customizations (upstreamed) of the gitlab-registry.cern.ch/fts images, e.g. addition of a few tools (e.g. ps) for debugging
  - fts-rest-flask : a few patches for issues with OIDC support before they are released upstream
- Main change: use of supervisord in all images, even for running httpd
  - Ensure that a container starts successfully (even if httpd fails to start): required to log into the container to troubleshoot problems

Kubernetes deployment

- Currently using a K8S 1.30 cluster
  - Deployed in the (OpenStack) cloud using Magnum service
- 3 pods (K8s Deployment)
  - ftsdb: deployed with *helm*, using image *bitnami/mariadb*
  - fts-server: 2 containers, fts-server and fts-monitoring, deployed with a K8s manifest
  - fts-api: container fts-rest-flask, deployed with a K8s manifest
- All services use persistent volumes for their non-volatile data
  - Actual storage in Ceph through OpenStack Cinder service
  - fts-server logs shared between *fts-server* and *fts-monitoring*
- Service configuration (e.g. *fts3config* or certs) passed as K8s secrets
- **All public endpoints exposed on port 443 through a K8S ingress**
  - Port 443: K8s ingress constraint, may change if using the new K8s gateway service

OIDC support

- Basic support present in FTS (and Rucio)
- Main problems experiences related to our INDIGO IAM configuration
  - In particular, lack of trailing / in IAM_ISSUER variable: in fact an INDIGO IAM documentation issue, most INDIGO IAM instances put it…
  - A v3.14.2 fix upstreamed to sanitize the token issuer URL when the trailing / is missing as FTS is expecting it to be present when doing the matching against the valid issuers
- OIDC client need to be created for *fts-rest-flask* and *rucio-server*
  - Scopes (*fts-rest-flask*): fts, wlcg, wlcg.groups, storage.create, storage.read
  - Grant types: authorization_code, refresh_token, token-exchange
  - Ruico server token must contain a specific audience, defined in FTS config and Rucio config
- Need to configure an entry in FTS database for each VO
  - Specific S3 parameters like issuer…
  - No interface to do it: SQL insert…

S3 support

- Still in progress: configured in FTS, not yet fully configured in Rucio
  - Lack of time and pressure!
  - Mostly a Rucio configuration issue: configuration of a grid storage (EOS) as a transfer node for multi-hop transfers as FTS cannot do S3 to S3 (lack of 3d party copy support)
- The most complex part remains the Rucio part…
  - Not Rucio experts (yet!)
  - Troubleshoot proper configuration of the token used by Rucio server to access FTS service
  - Configure properly Rucio distances and rules

Documentation entry points

- Token support:
  - Rucio: https://rucio.github.io/documentation/operator/installing_server/#enable-oidc
  - FTS3: https://fts3-docs.web.cern.ch/fts3-docs/docs/token_support.html
- S3
  - Rucio : https://rucio.github.io/documentation/operator/s3_rse_config/
  - FTS3 : https://fts3-docs.web.cern.ch/fts3-docs/docs/s3_support.html

Conclusion

- FTS successfully deployed in a Kubernetes cluster
  - No stress tests done to identify the impact on performances
- OIDC support is there and working
- S3 endpoints support is present but configuration can be a little bit tedious
- Happy to share work done and K8s manifests
  - Also interested to get/use Helm charts made by others!
- Future work planned: integration into ArgoCD
- Help from Mihai is fantastic!