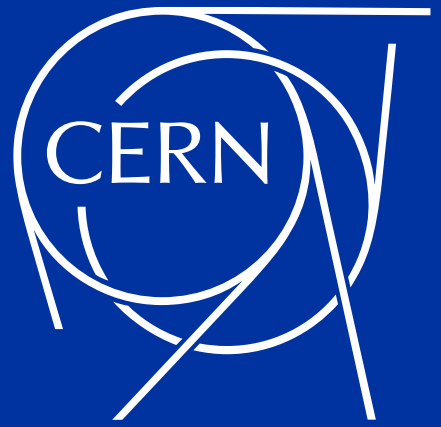


Rucio deployment in ATLAS and CMS

Radu Carpa

07 June 2022

What is Rucio?

- https://rucio.cern.ch/
- **Open-source data-management software**

- Upload files to storage servers
- Group them into datasets
- Enforce replication rules
  - (ex: maintain 2 copies on 2 different continents)
- Recover from replica lost
- Etc.

Rucio components

Written in python:

- **Server instances exposing a REST API**
- **Clients for that API**
- **Web UI**
- **20+ different daemons (many optional)**
  - rule evaluation
  - transfers
  - deletion
  - …

Rucio Deployment

The Rucio core development team officially supports 2 deployment options:

- **Directly via `pip install`:**
  - configurable via the `rucio.cfg` file
- **Containerized environments:**
  - Bare containers
    - configurable via mounting `rucio.cfg`, or via ENV variables
    - possibility to hotfix Rucio code before running by mounting '*.patch' files into the container
  - Kubernetes via the official helm charts
    - configurable via helm values

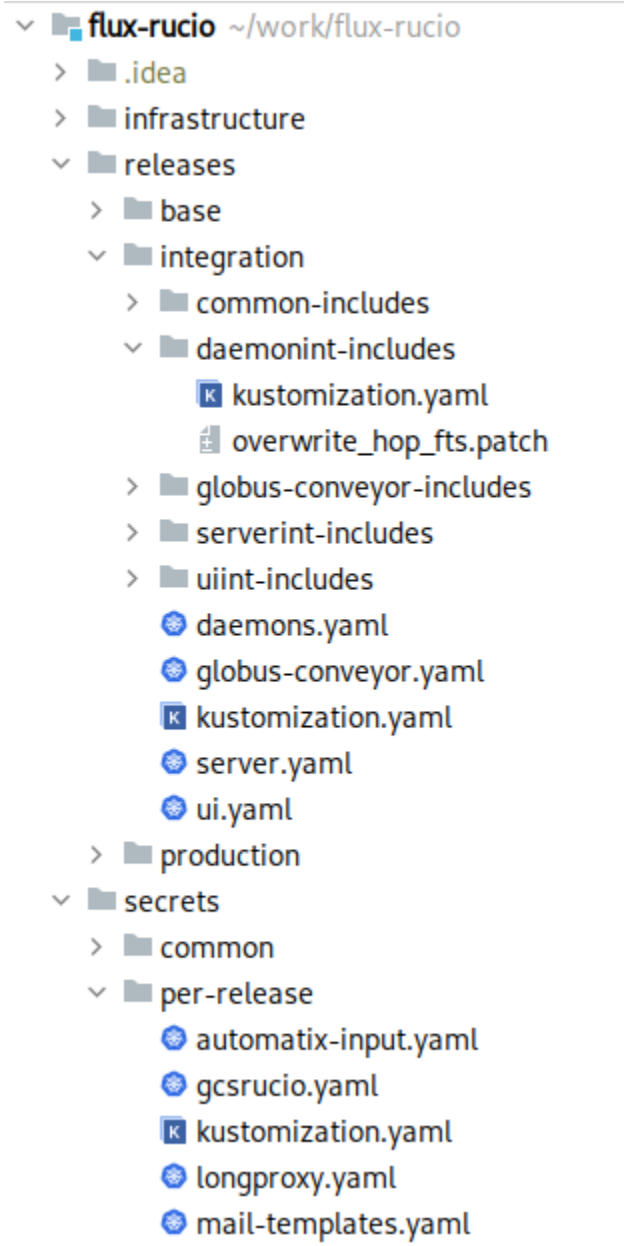
Rucio on Kubernetes

- **Both ATLAS and CMS use official helm charts in Openstack-managed k8s clusters:**
  - CMS was the first user of Rucio on Kubernetes in Production. Running fully on k8s now.
  - ATLAS migration to Kubernetes still on-going
    - most workloads migrated, but authentication, web UI and part of monitoring still on puppet-managed VMs

- **ATLAS: 1 integration (5 nodes) + 3 production clusters (3 * 15 = 45 nodes)**
  - Required capacity: ~ ½ of that. The rest is for spare capacity and comfortable cluster re-installs
  - Behind self-managed load-balancer (haproxy 2.5.6) on puppet-managed VMs
    - In charge of x509-based authentication
    - Performs path-based traffic splitting for performance reasons
- **CMS: 1 integration (4 nodes) + 1 production cluster (6 nodes)**

ATLAS Rucio k8s deployment

- **Uses vanilla mainstream containers and helm charts**
- **Managed from a Git(lab) repository using Flux2:**
  - HelmRelease custom resources to manage Rucio
  - Mozilla SOPS (to store encrypted secrets in git)
  - Kustomize (required to use SOPS in flux2, but also used for rucio hot-patching)



GitOps using Flux

releases/production/daemons.yaml +2 -2 View file @e2014a49

```
...  ...  @@ -36,7 +36,7 @@ spec:
36   36         valuesKey: values.yaml
37   37       values:
38   38         automatixCount: 1
39        -     conveyorTransferSubmitterCount: 2
     39   +     conveyorTransferSubmitterCount: 5
40   40         conveyorPollerCount: 1
41   41         conveyorFinisherCount: 2
42   42         conveyorReceiverCount: 1
...  ...  @@ -111,7 +111,7 @@ spec:
111  111        conveyorTransferSubmitter:
112  112          activities: "'Analysis Input' 'Analysis Output' 'Data Brokering' 'Data Challenge' 'Data
                  Consolidation' 'Data rebalancing' 'Debug' 'Express' 'Functional Test' 'Functional Test WebDAV' 'Group
                  Subscriptions' 'Production Input' 'Production Output' 'Recovery' 'Staging' 'T0 Export' 'T0 Export' 'T0
                  Tape' 'User Subscriptions' 'default'"
113  113          sleepTime: 60
114       -       threads: 30
     114  +       threads: 10
115  115          sourceStrategy: "throughput"
116  116          bulk: 1000
117  117          groupBulk: 200
...  ...
```

GitOps using Flux

overwrite_hop_fts.patch 892 bytes

```
diff --git a/lib/rucio/transfertool/fts3.py b/lib/rucio/transfertool/fts3.py
index a4970b4c5..0b35112e5 100644
--- a/lib/rucio/transfertool/fts3.py
+++ b/lib/rucio/transfertool/fts3.py
@@ -278,7 +278,8 @@ def bulk_group_transfers(transfer_paths, policy='rule', group_bulk=200, s
                 # We don't allow multihop via a tape, so bring_online should not be set on a
                 if transfer is transfer_path[0] and hop_params['bring_online']:
                     job_params['bring_online'] = hop_params['bring_online']
-
+            if not job_params['overwrite']:
+                job_params['overwrite_hop'] = True
             group_key = 'multihop_%s' % transfer_path[-1].rws.request_id
             grouped_transfers[group_key] = {'transfers': transfer_path[0:group_bulk], 'job_p
         elif len(transfer_path[0].legacy_sources) > 1:
```

CMS Rucio k8s deployment

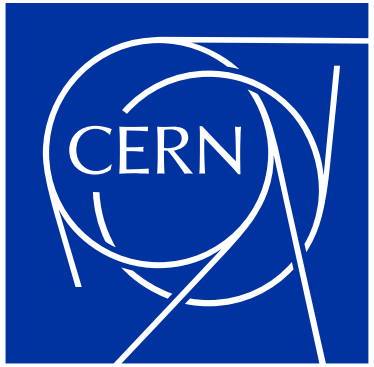
- **Custom containers based on official ones, but with minor customization**
  - CMS-specific configuration of Rucio behavior via policies
  - patches to hotfix Rucio
  - templates for mail messages
  - CA bundles for TLS communication
- **Rucio helm release managed by Flux from Git(hub) repository**
  - secrets managed manually in Kubernetes
- **Monitoring probes run as pod with jobber (cron replacement)**
- **Proxy renewal, syncing user accounts, site definitions all run as kubernetes cronjobs**

Logging, Monitoring and Alerting

- **Logging: filebeat and logstash. Send metrics to CERN monit**
- **Application metrics:**
  - Rucio mostly relies on statstd
  - On-going migration to native Prometheus metrics (everything except timers should work already)
  - Prometheus push-gateway for probes and other cron-like jobs
- **ATLAS: both stastd and Prometheus (alerting from Prometheus using Alertmanager)**
  - custom kube-prometheus-stack in clusters both for application and infrastructure metrics
- **CMS: Fully on Prometheus. Statsd-exporter to get statsd metrics into Prometheus**
  - also, kube-eagle for infrastructure-level metrics

Conclusion

- **The path to modernizing Rucio deployments was long, but had satisfying results**



home.cern