



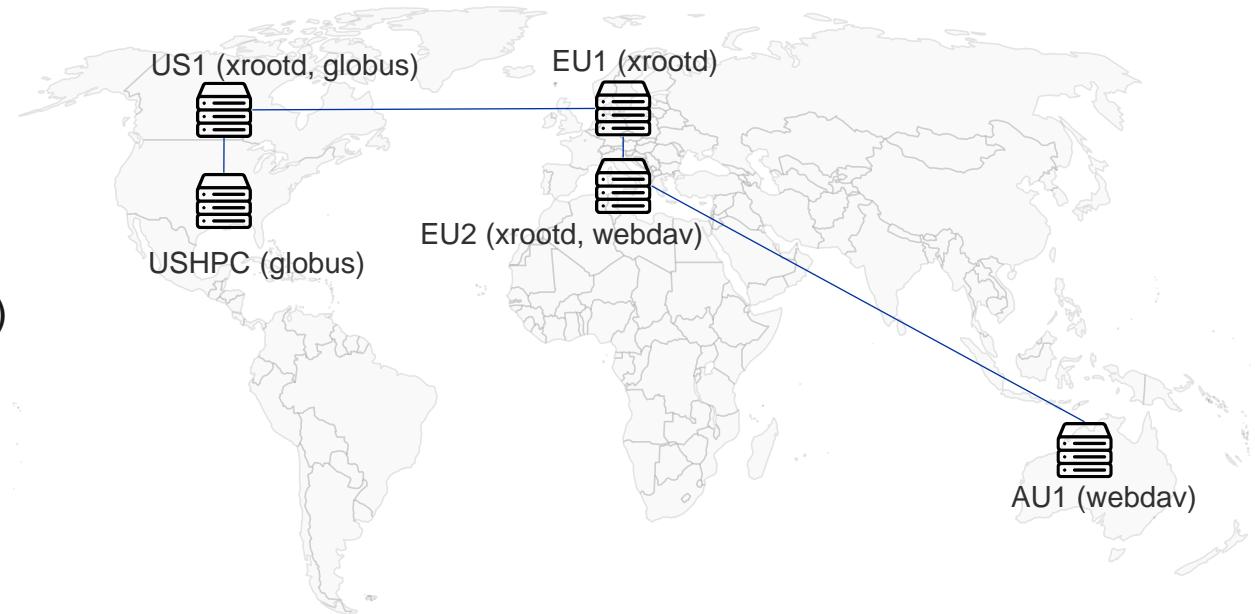
The journey of a file in Rucio

Radu Carpa

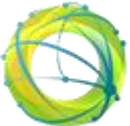

28 Mar. 2023

What is Rucio?

- <https://rucio.cern.ch/>
- Open-source data-management software
- Data catalog to reference and organize files stored on multiple storage servers (RSE: Rucio Storage Element):
 - Upload files to RSEs
 - Group them into datasets
 - Enforce replication rules
 - (ex: maintain 2 copies on 2 different continents)
 - Recover from replica lost
 - Etc.



About Rucio

- **Client – server architecture**
 - Server instances exposing a REST API
 - Client for that API + a separate web UI (a complete re-work of the UI in active development)
 - 20+ different daemons on the backend (many optional)
- **Multiple pluggable drivers for directly accessing the storage (uploading data, deletion)**
- **2 supported backends for third party copy transfers:**
 - 1) FTS  2) Globus 
- **In practice: most production deployments heavily use gfal2 for direct storage access; and, for third-party copy transfers: FTS in 99+% of cases**

Uploading a file with/to rucio

```
rucio upload --rse 'EU1' --scope rcarpa --name analysis_input cat_meme.png
```

- **Takes care of:**

- authentication and authorization to the Rucio catalog
- generating the correct path (root://eu1:1094//rucio/rcarpa/73/f4/analysis_input)
- registering the file into the catalog
- authentication to storage; in the future: very probably authorization too
- actual upload

Replicating data via FTS

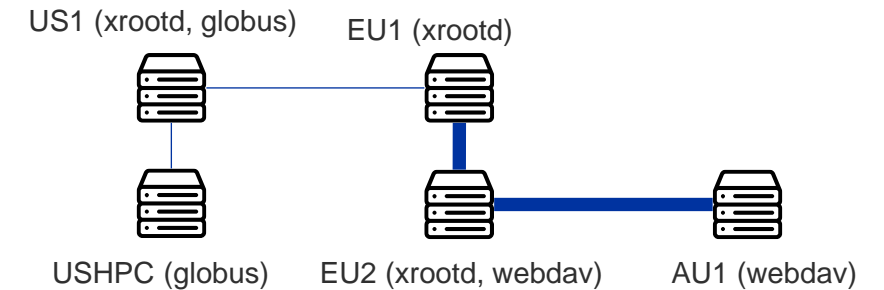
```
rucio add-rule rcarpa:analysis_input 1 'country=AU'
```

- **Rucio will:**

- pick a storage server in AU
- compute paths: maybe multi-hop; take care of protocol compatibility
- throttle: wait and delay submission if too many active transfers
- submit to FTS: generate paths/URLs (sign URLs when needed), set job parameters

```
{files: [{sources: ["root://us1:1094//rucio/rcarpa/73/f4/analysis_input"],
             destinations: ["root://eu1:1094//rucio/rcarpa/73/f4/analysis_input"],
             checksum: ...},
         {sources: ["davs://eu1/rucio/rcarpa/73/f4/analysis_input"],
             destinations: ["davs://au1/rucio/rcarpa/73/f4/analysis_input"],
             checksum: ...},
         ],
  params: {"multihop": True, "overwrite": true, "priority": ...}}
```

- wait for completion: via ActiveMQ and/or polling FTS

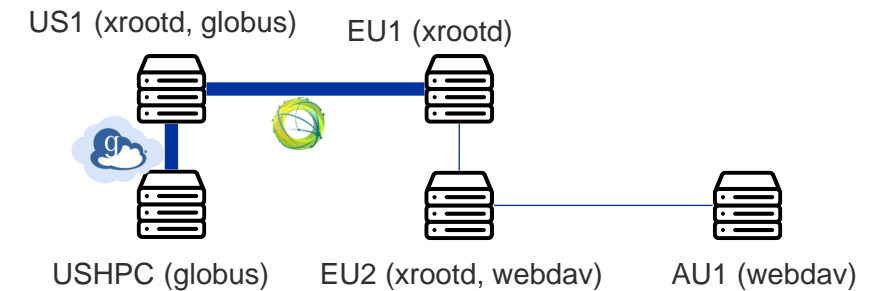


Replicating data across transfertools(FTS + Globus)

```
rucio add-rule rcarpa:analysis_input 1 USHPC
```

- **Rucio will:**

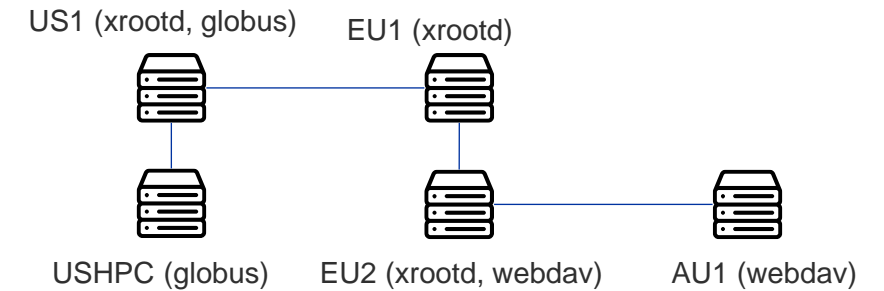
- transfer the file to US1 using FTS as described on previous slide
- once the first hop is done:
 - request transfer of the file between US1 and USHPC using Globus online
 - wait for completion: poll globus regularly until the transfer is done



Deletion

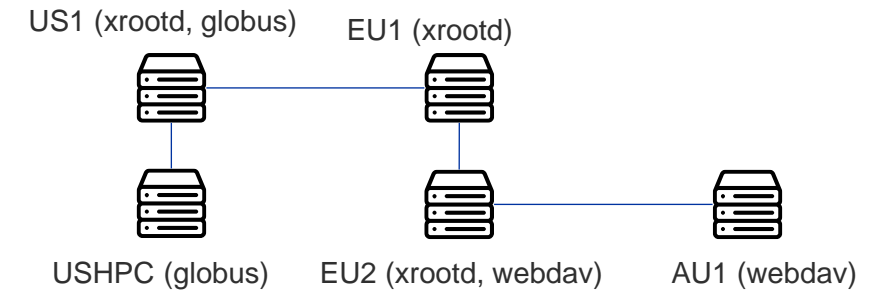
- **There are now 5 copies of the file:**
 - EU1 as result of 'rucio upload'
 - AU1 and USHPC: from the 2 rules we just added
 - US1 and EU2: temporary data for multi-hop transfers

- **Will delete the US1 and EU2 replicas**



Other responsibilities of Rucio

- **Handle transfer failures and retries**
- **Quota management**
- **Cleaning files which are not registered in the catalog**
- **(semi-automatically) recover from loss or corruption of a replica**
- **Grouping files into groups (datasets; containers) to be handed together**



Rucio and FTS grow up together

- **Many advanced features require joint development from both teams. Recent examples:**
 - Automatic detection and overwrite of corrupted file
 - Improved ActiveMQ message handling for faster convergence
 - Cloud storage support
 - Wait for tape archival (mostly implemented, but requires a bit more work)
 - Multi-source and multi-hop support
 - Planned, but not yet done:
 - scitags packet marking
 - full OIDC token support

The Rucio Kubernetes demo

- Available at <https://github.com/rucio/k8s-tutorial>
- Runs Rucio, FTS, xrootd containers in a local Kubernetes env (minikube)
- For some newcomers, it's the first exposure to Rucio, FTS and xrootd
 - Sometimes we end being at the frontline for problems and bugs in any of the 3 components

Friendly punches towards FTS

- **Pop quiz (it's a trap! 😊):**

- ``verify_checksum``: is it a per-file, or a per-job attribute ?
- Enumerate all consequences of setting ``archive_timeout`` on a transfer
- In a submission job, given that both sources and destinations are lists. What happens if we set a list of destinations? What if we set both sources AND destinations to lists ?

```
{files: [{sources: ["root://us1:1094//rucio/rcarpa/73/f4/analysis_input"],
              destinations: ["root://eu1:1094//rucio/rcarpa/73/f4/analysis_input"],
              checksum: ...}],
params: {"multihop": True, "overwrite": true, "priority": ...}}
```

- **While the reasons of inconsistencies are obvious to us, they are far from being obvious to newcomers.**

Feedback from ATLAS ops

The Rucio + FTS combo:

- works very well and gets the job done (according to many sources)
- makes questionable decisions when obvious (to a human) better choices exists (according to one anonymous source, that nobody will, definitely not, recognize)

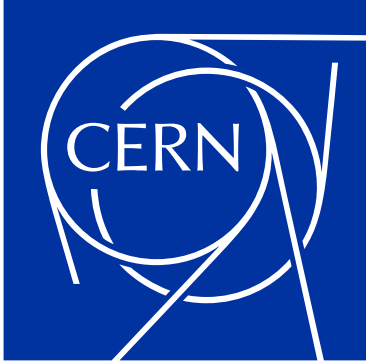
I, personally, agree with both statements. Most of the time it works very well. But it requires too much baby-sitting from the ops in some corner cases (tweaking limits; forcing specific rules manually)

Summary

The FTS development team is always very responsive and helpful. Especially considering their limited resource.

How do we proceed to improve the state of (rare, but impactful) obviously sub-optimal submissions? In Rucio? in FTS? Both?

Would be very helpful to have first-party FTS containers



home.cern