

# Rucio

## ErUM-DATA Collaboration Meeting

---







[Thomas Beermann](#)

on behalf of the Rucio team



# Rucio in a nutshell

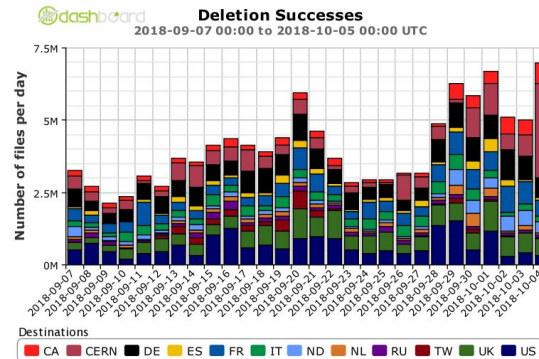
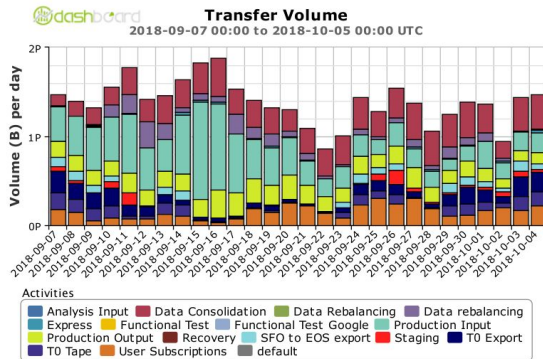
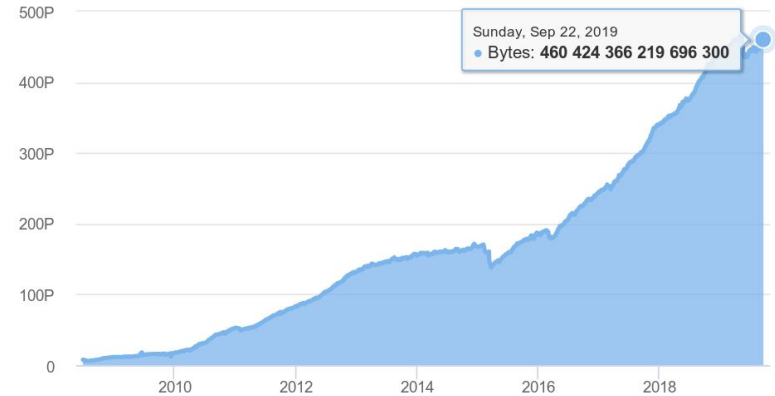
---

- Rucio provides a complete and generic scientific data management service
  - Seamless integration of scientific and commercial storage and network systems
  - Data can be scientific observations, measurements, objects, events, images saved in files
  - Facilities can be distributed at multiple locations belonging to different administrative domains
  - Designed with more than a decade of operational experience in very large-scale data management
- Rucio manages location-aware data in a heterogeneous distributed environment
  - Creation, location, transfer, deletion, and annotation of data
  - Orchestration of dataflows with both low-level and high-level policies
- Principally developed by and for the High-Energy Physics experiment [ATLAS](#)
- Rucio is open source and available under Apache 2.0 license
- Make use of established toolchains for and with the community      



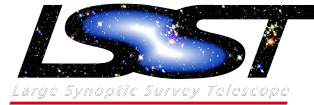
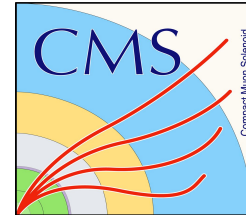
# Data management for ATLAS

- A few numbers to set the scale
  - 1B+ files, 450 PB of data, 400+ Hz interaction rate
  - 120 data centres, 5 HPCs, 600 storage areas
  - Up to 4M files/2.5 PB transferred per day
  - 10PB access from clients per day; >1000 active users
- Increase 1+ order of magnitude for LHC Run 4





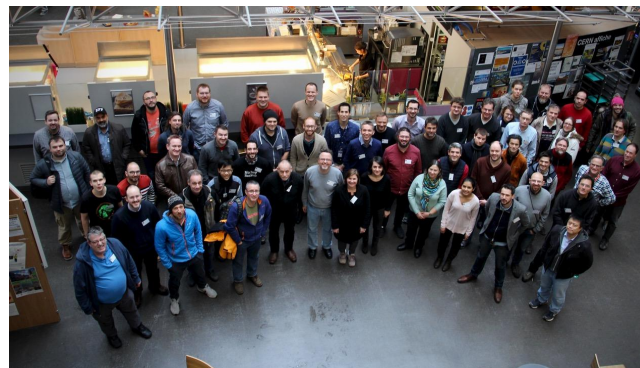
# Community





# Community

- Rucio Community Workshops [[2018](#)] [[2019](#)]
- Rucio Coding Camps 2018 [[2018](#)] [[2019](#)]
- Development Meetings [[Weekly](#)]





# Rucio main functionalities

- Provides many features that can be enabled selectively
  - File and dataset catalog
  - Transfers between facilities including disk, tapes, clouds, HPCs
  - Web-UI, CLI, and API to discover/download/upload/transfer/annotate data
  - Extensive monitoring for all dataflows
  - Support for caches and CDN workflows
  - Expressive policy engines with rules and subscriptions
  - Automated corruption identification and recovery
  - Data popularity based replication
  - ...
- Rucio can be integrated with Workload and Workflow Management System
  - Already supporting PanDA, the ATLAS WFMS
  - Belle-II is driving the integration with DIRAC



# Operations model

---

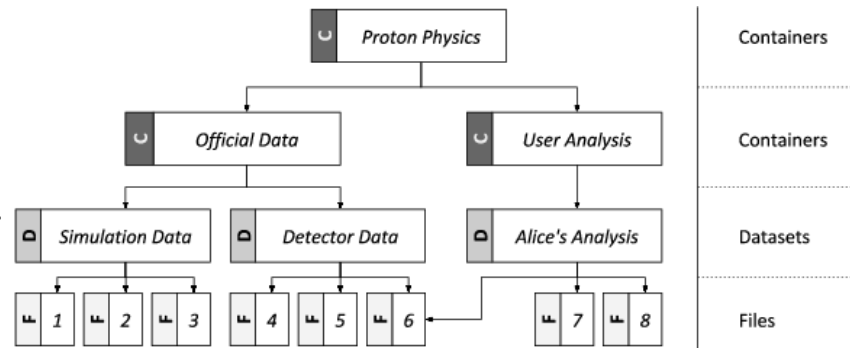
- Objective was to minimise the amount of human intervention necessary
- Large-scale and repetitive operational tasks can be automated
  - Bulk migrating/deleting/rebalancing data across facilities at multiple institutions
  - Popularity driven replication and deletion based on data access patterns
  - Management of disk spaces and data lifetime
  - Identification of lost data and automatic consistency recovery
- Administrators at the sites are not operating any Rucio service
  - Sites only operate their storage exposed via protocols (POSIX, ROOT, HTTP, WebDAV, S3, gsiftp, ... )
  - Users have transparent access to all data in a federated way
- Easy to deploy
  - PIP packages, Docker containers, Kubernetes (Helm Charts)



# Rucio concepts - Namespace

- All data stored in Rucio is identified by a **Data Identifier (DID)**

- There are different types of DIDs
  - Files
  - Datasets    Collection of files
  - Container    Collection of dataset and/or container
- Each DID is uniquely identified and composed of a scope and name, e.g.:



`detector_raw.run34:observation_123.root`

scope

name





# Rucio concepts - Declarative data management

---

- Express what you want, not how you want it
  - *e.g., "Three copies of this dataset, distributed evenly across multiple continents, with at least one copy on TAPE"*
- Replication rules
  - Rules can be dynamically added and removed by all users, some pending authorisation
  - Evaluation engine resolves all rules and tries to satisfy them by requesting transfers and deletions
  - Lock data against deletion in particular places for a given lifetime
  - Primary replicas have indefinite lifetime rules
  - Cached replicas are dynamically created replicas based on traced usage and popularity
  - Workflow system can drive rules automatically, e.g., job to data flows or vice-versa
- Subscriptions
  - Automatically generate rules for newly registered data matching a set of filters or metadata
  - *e.g., project=data17\_13TeV and data\_type=AOD uniformly across T1s*



# Rucio concepts - RSEs

---

- Rucio Storage Elements (RSEs) are logical entities of space
  - No software needed to run at the facility except the storage system, e.g., EOS/dCache/S3, ...
  - RSE names are arbitrary, e.g., "CERN-PROD\_DATADISK", "AWS\_REGION\_USEAST", ...
  - Common approach is one RSE per storage class at the site
- RSEs collect all necessary metadata for a storage system
  - Protocols, hostnames, ports, prefixes, paths, implementations, ...
  - Data access priorities can be set, e.g., to prefer a different protocol for LAN-only access
- RSEs can be assigned metadata as well
  - Key/Value pairs, e.g., *country=UK, type=TAPE, is\_cached=False, ...*
  - You can use RSE expressions to describe a list of RSEs, e.g. *country=FR&type=DISK* for the replication rules



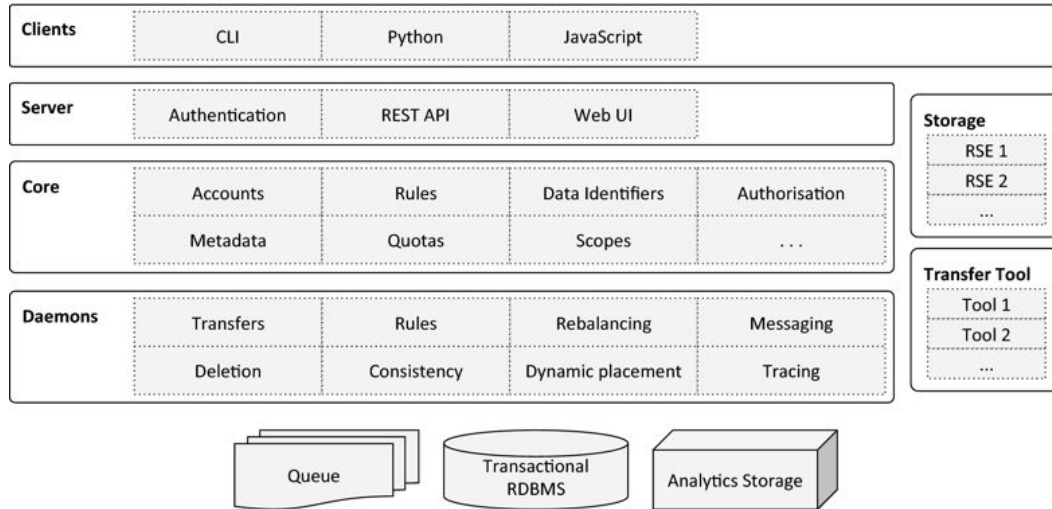
# Rucio concepts - Metadata

---

- Rucio supports different kinds of metadata
  - File internal metadata, e.g., *size, checksum, creation time, status*
  - Fixed physics metadata, e.g., *number of events, lumiblock, cross section, ...*
  - Internal metadata necessary for the organisation of data, e.g., *replication factor, job-id,*
  - Generic metadata that can be set by the users
- Generic metadata can be restricted
  - Enforcement possible by types and schemas
  - Naming convention enforcement and automatic metadata extraction
- Provides additional namespace to organise the data
  - Searchable via name and metadata
  - Aggregation based on metadata searches
  - Can also be used for long-term reporting, e.g., *evolution of particular metadata selection over time*



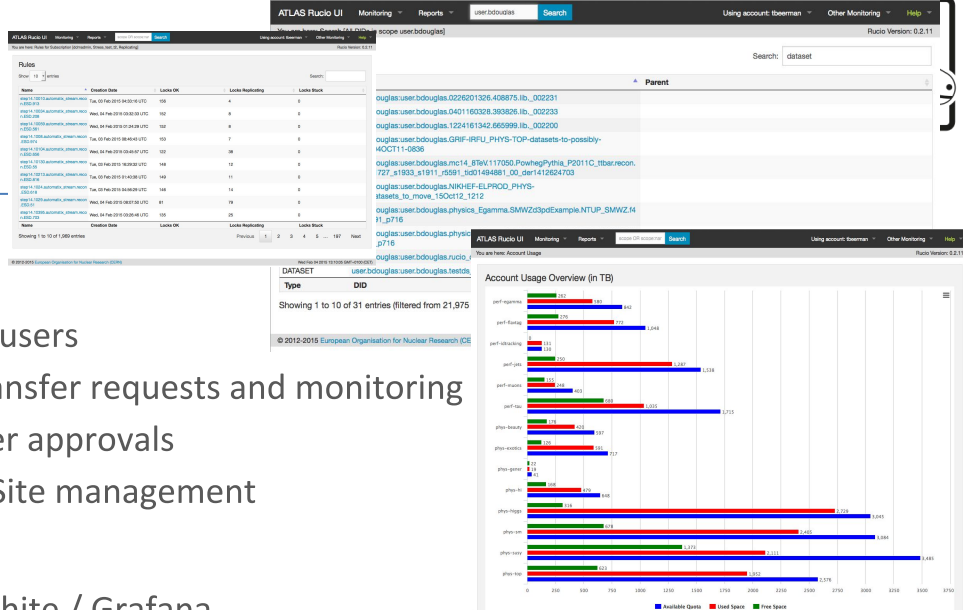
# Architecture



- **Servers**
  - HTTP REST/JSON APIs
  - Token-based security (x509, ssh, kerberos, ...)
  - Horizontally scalable
- **Daemons**
  - Orchestrates the collaborative work e.g., transfers, deletion, recovery, policy
  - Horizontally scalable
- **Messaging**
  - STOMP / ActiveMQ-compatible
- **Persistence**
  - Object relational mapping
  - Oracle, PostgreSQL, MySQL/MariaDB, SQLite
- **Middleware**
  - Connects to well-established products, e.g., FTS3, DynaFed, dCache, EOS, S3, ...
- **Python**
  - Support for Python2 and Python3

# Monitoring & analytics

- RucioUI
  - Provides several views for different types of users
  - Normal users: Data discovery and details, transfer requests and monitoring
  - Site admins: Quota management and transfer approvals
  - Central administration: Account / Identity / Site management
- Monitoring
  - Internal system health monitoring with Graphite / Grafana
  - Transfer / Deletion / ... monitoring built on HDFS, ElasticSearch, and Spark
  - Messaging with STOMP
- Analytics and accounting
  - e.g., Show which the data is used, where and how space is used, ...
  - Data reports for long-term views
  - Built on Hadoop and Spark





# Development

---

- Release cycle and support period
  - Bi-weekly patch releases (Bugfixes, minor enhancements)
  - ~3 feature (named) releases per year (Features, major changes)
  - Once a year a feature version is designated as a Long-Term Support (LTS) release
- Development organized as open-source community project
  - Weekly development meetings; Release roadmap for each feature release
  - Contributors describe their planned developments, receive comments from community
  - Extensive integration and unit testing across all supported databases



# Current developments

---

- Multi-experiment data management
  - Smart sharing of infrastructure across experiments
  - Smart sharing of data across experiments
- Quality of Service
  - Replication rules can take storage quality descriptions into account
- Expanding support for commercial cloud providers
  - Transparent Google Cloud and S3 (OpenStack, AWS) integration already supported
- Capability-based authentication, authorisation, and permissions
  - Data embargoes, JWT bearer tokens, Macaroons, OpenID, EduGain
- Network interface integration with SDNs
- Integration with research publication databases, e.g., Zenodo



# More information

---

Website



<http://rucio.cern.ch>

Documentation



<https://rucio.readthedocs.io>

Repository



<https://github.com/rucio/>

Images



<https://hub.docker.com/r/rucio/>

Online support



<https://rucio.slack.com/messages/#support/>

Developer contact



[rucio-dev@cern.ch](mailto:rucio-dev@cern.ch)

Journal article



<https://doi.org/10.1007/s41781-019-0026-3>

Twitter



<https://twitter.com/RucioData>