Rucio

Scientific Data Management

Dr. Martin Barisits
on behalf of the Rucio team

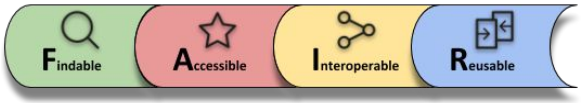
RUCIO
SCIENTIFIC DATA MANAGEMENT

Rucio in a nutshell

- Rucio provides a mature and modular scientific data management federation
  - **Seamless integration** of **scientific and commercial** storage and their network systems
  - Data is stored in files and can contain **any potential payload**
  - 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
  - Orchestration of dataflows with both low-level and high-level policies
- Principally developed by and for ATLAS, now with many more communities
- Rucio is open-source software licenced under *Apache v2.0*
- Makes use of established open-source toolchains

Rucio main functionalities

- Provides many features that can be enabled selectively



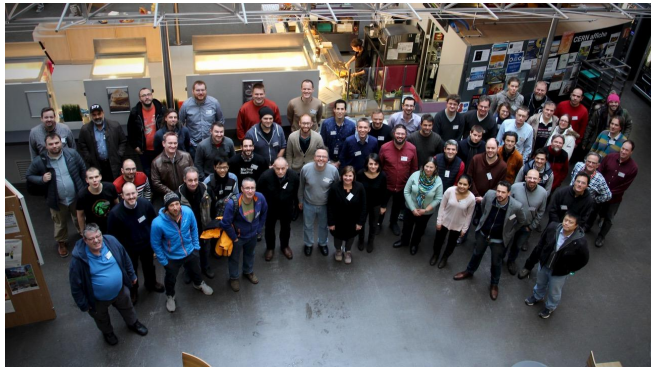
More advanced features

  - 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
  - …

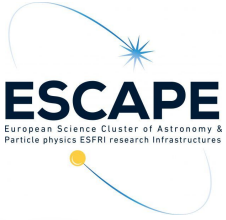
- Rucio can be integrated with Workload and Workflow Management System
  - Already supporting PanDA, the ATLAS WFMS
  - Communities evaluate & develop integrations, e.g., CRAB/WMAgent, DIRAC, Pegasus, or Condor

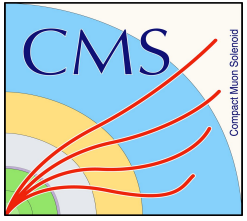
Regular events

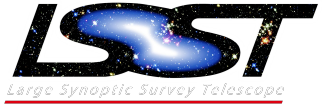
- Rucio Community Workshops [2018] [2019]
- Rucio Coding Camps [2018] [2019]
- Development Meetings [Weekly]



A growing community



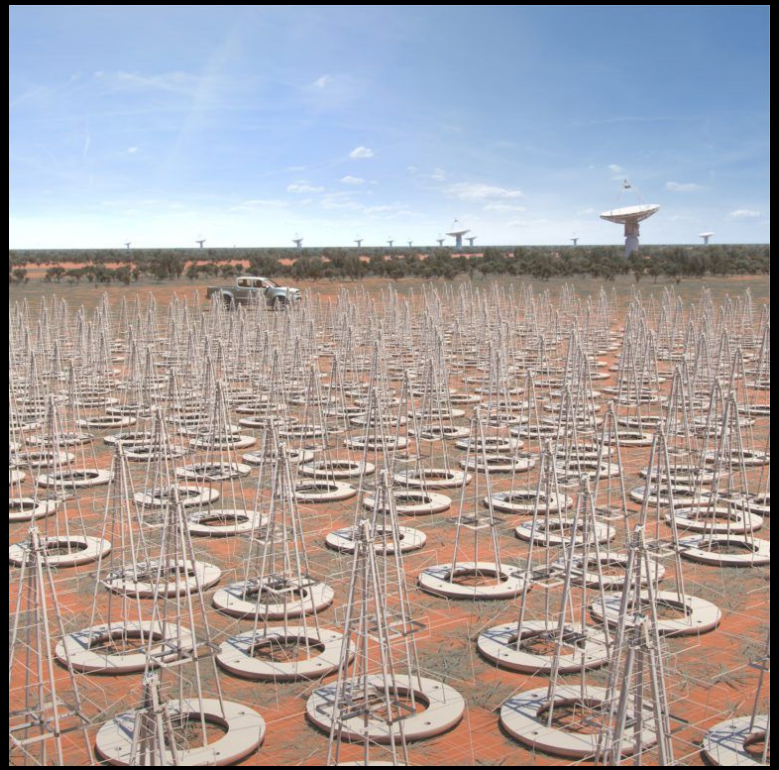




Why a common data management solution?

- Shared use of the global research infrastructures will become the norm, especially with sciences at the scale of HL-LHC, DUNE, and SKA
  - Competing requests on a **limited set of storage and network**, data centres will be **multi-experiment**
  - **Compute** is usually well-covered, e.g., via common scheduling, interfaces, and specifications
  - **Data** was always missing **a common open-source solution** to tackle our **shared challenges**

- Ensure more efficient use of available data resources across multiple experiments
  - **Allocate storage and network based on science needs**, not based on administrative domains
  - **Orchestrate dataflow policies across experiments**
  - Dynamically support compute workflows with **adaptive data allocations**
  - **Unify monitoring**, reporting and analytics to data centres and administration
  - Potential for **shared operations across experiments**



Rohini Joshi (SKAO)

SKA Regional Centres

- SRCs will provide a **platform for transparent data** access, data distribution, post-processing, archive storage, and software development
- **Up to 1 PB/day** to be ingested from each telescope, and made available for access and post-processing
- Need a way to **manage data in a federated way** across many physical sites transparent to the user

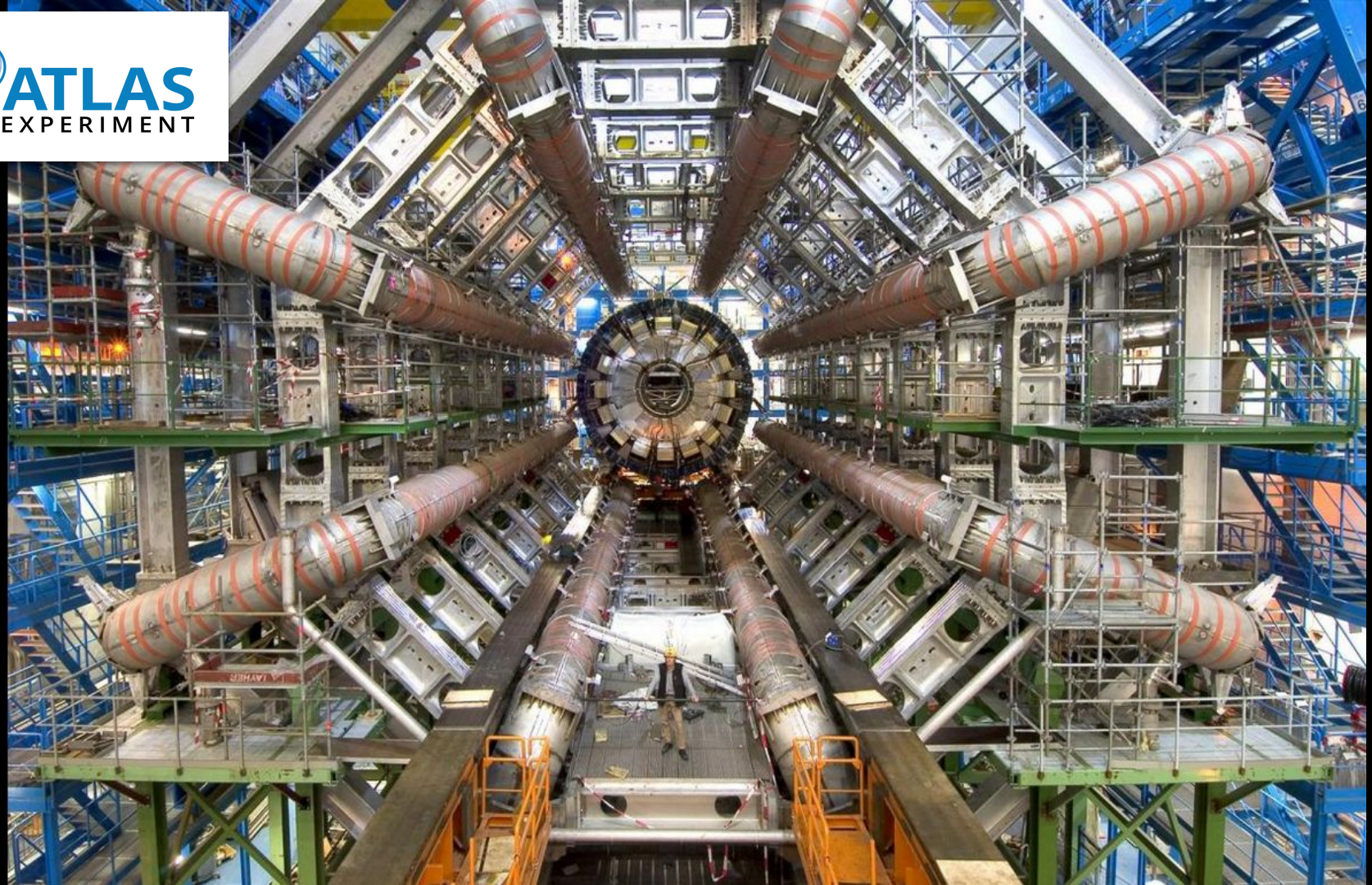
ARCHIVE

DATA DISCOVERY

DISTRIBUTED DATA PROCESSING

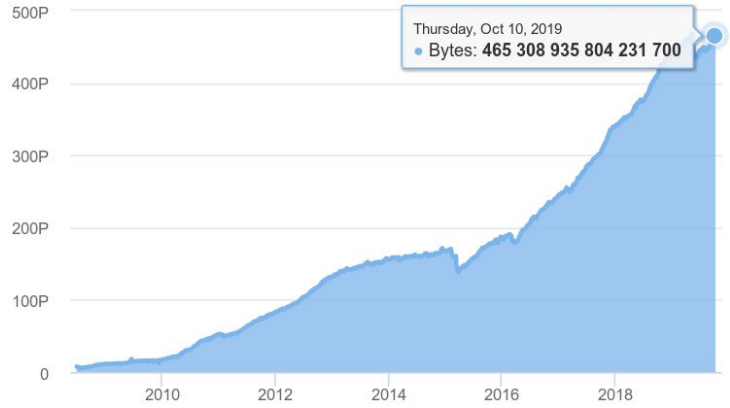
USER SUPPORT

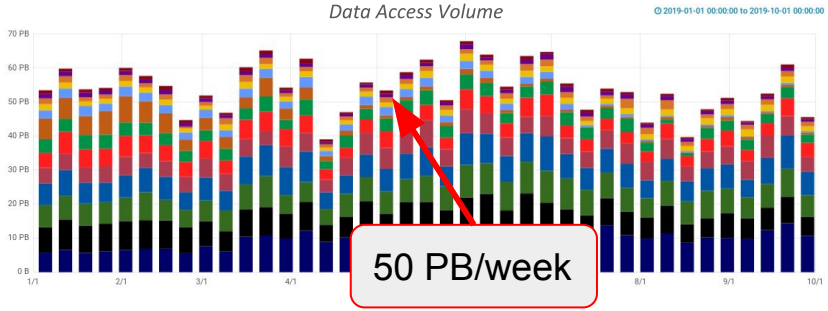
INTEROPERABILITY

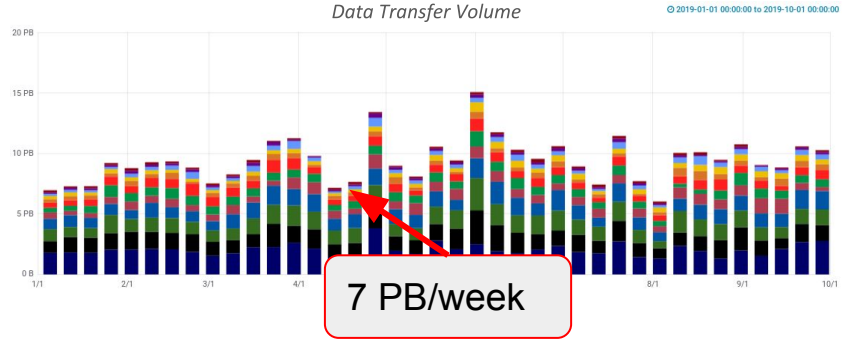

ATLAS
EXPERIMENT

Data management for ATLAS

- A few numbers to set the scale
  - 1B+ files, 450+ PB of data, 400+ Hz interaction
  - 120 data centres, 5 HPCs, 2 clouds, 1000 users
  - 500 Petabytes/year transferred & deleted
  - 2.5 Exabytes/year uploaded & downloaded
- Increase 1+ order of magnitude for HL-LHC

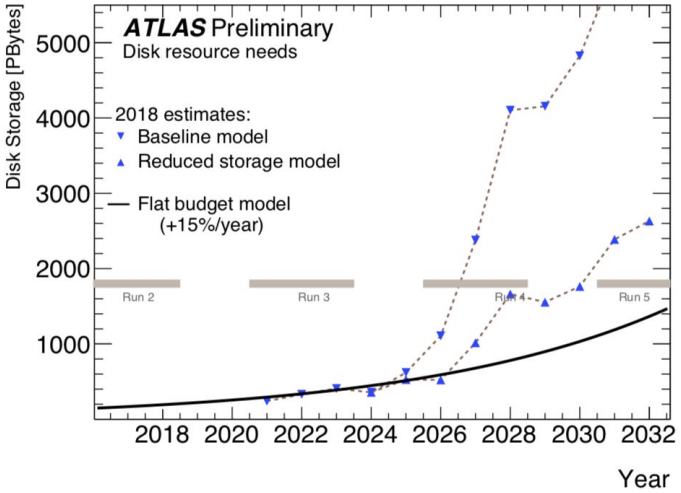






Data management for ATLAS at HL-LHC

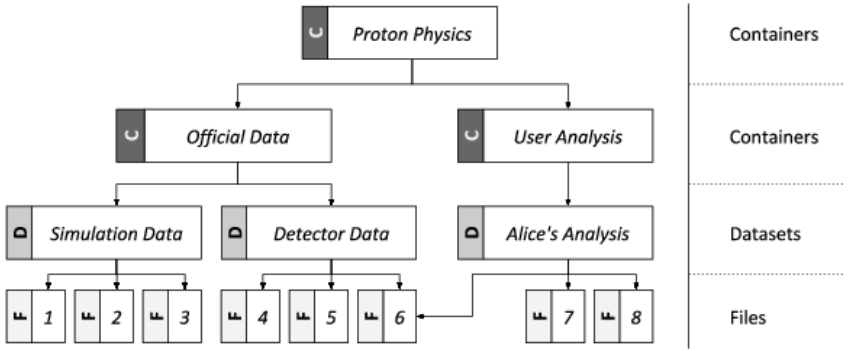
- Rucio is a central component to tackle HL-LHC data
  - Smart orchestration of the dataflow
  - Easy integration of new systems, ideas, and components
- Several combined effort R&D activities launched
  - Distributed storage and caching *Data Lakes*
  - Fine-grained data delivery services *iDDS & ServiceX*
  - Commercial cloud integration *Google & Co*



- R&D Highlight for HL-LHC: *Data Carousel*
  - Tight integration of workflow and dataflow for more **efficient use of high-latency storage** (i.e., tape)
  - New algorithms on **multi-site I/O scheduling** for both writing and reading
  - **Smart placement** of data on based on estimated access patterns

Rucio concepts - Namespace

- All data stored in Rucio is identified by a **D**ata **ID**entifier (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*

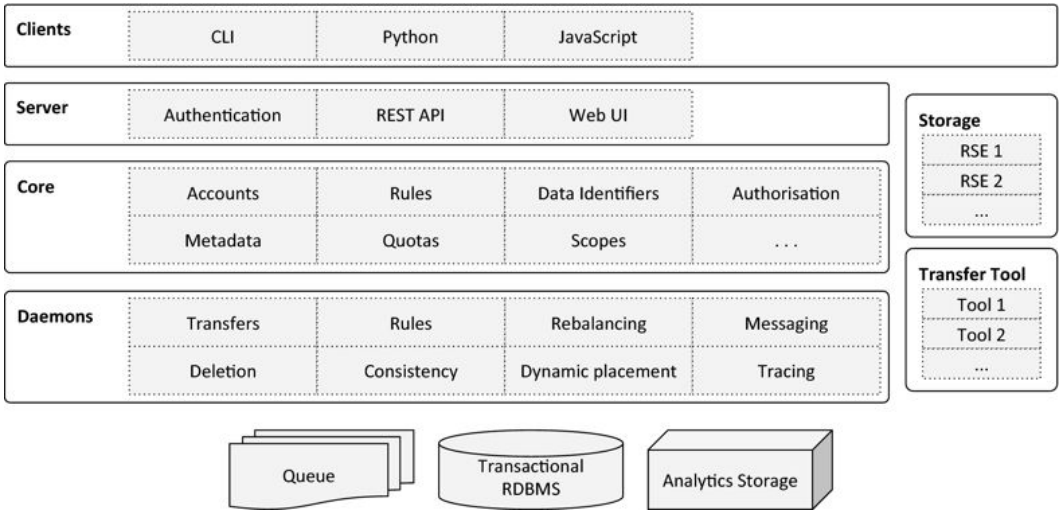
Rucio concepts - RSEs

- 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

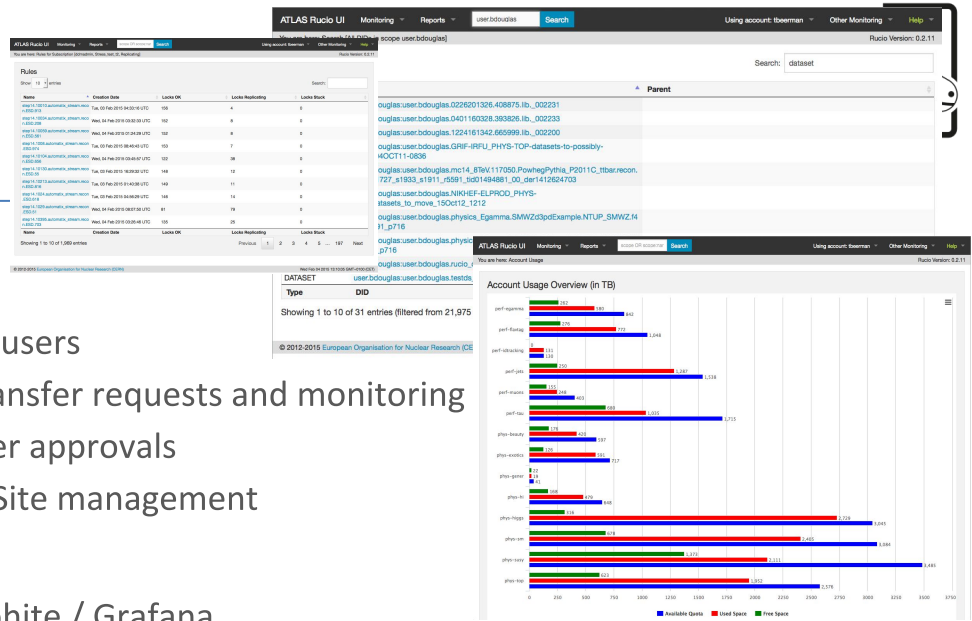


- **Servers**
  - HTTP REST/JSON APIs
  - Token-based security (x509, ssh, kerberos, …)
  - Horizontally scalable
- **Daemons**
  - Orchestrates the collaborative work
    e.g., transfers, deletion, recovery, policy
  - Horizontally scalable
- **Messaging**
  - STOMP / ActiveMQ-compatible
- **Persistence**
  - 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

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

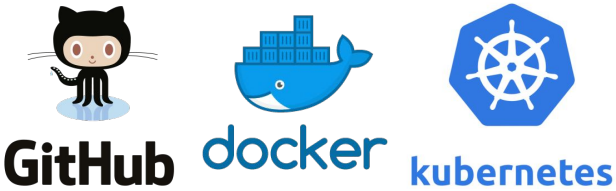
Monitoring & analytics



- 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

Community-driven development

- We have successfully moved to **community-driven development**
  - Requirements, features, issues, release are **publicly discussed** (e.g., weekly meetings, GitHub, Slack)
  - The core team is usually only **providing guidance** for architecture/design/tests
  - Usually 1-2 persons from a **community then take responsibility** to **develop** the software extension and also its **continued maintenance**
- Communities are helping each other **across experiments**
  - Effective across time zones due to US involvement
  - Automation and containerisation of development **lowers barrier of entry** for newcomers
  - Core team then only takes care about the management and packaging of the releases



Summary

- Several experiments and communities went from evaluation to production
  - AMS and Xenon as early adopters
  - Adoption by CMS was a decisive moment
  - Strong US and UK participation for support, development, and deployment
  - Successful integrations with existing software and computing infrastructures
- Emerging strong cooperation between HEP and multiple other fields
  - Notably neutrino and astronomy, with growing interest from more diverse sciences
- Community-driven innovations to enlarge functionality and address common needs
- Rucio is developing into a common standard for scientific data management
  - A successful collaborative open source project

Thank you!

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
Journal article	https://doi.org/10.1007/s41781-019-0026-3
Twitter	https://twitter.com/RucioData