Experiences building a common system

Mario.Lassnig@cern.ch
for the Rucio team

Future Trends in Nuclear Physics Computing
29 Sep 2020 - 1 Oct 2020
https://indico.bnl.gov/event/9023/

Software and Computing Round Table
7 Apr 2020
https://www.jlab.org/indico/event/356/

System overview

A look back

Discussion and interpretation

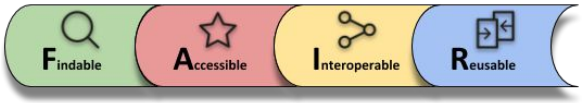
Rucio in a nutshell

- 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 **global single namespace** 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*
- Open community-driven development process

Rucio main functionalities

- Provides many features that can be enabled selectively
  - Horizontally scalable catalog for files, collections, and metadata
  - Transfers between facilities including disk, tapes, clouds, HPCs
  - Authentication and authorisation for users and groups
  - Web-UI, CLI, FUSE, and REST API
  - Extensive monitoring for all dataflows
  - Expressive policy engines with rules, subscriptions, and quotas
  - Automated corruption identification and recovery
  - Transparent support for multihop, caches, and CDN dataflows
  - Data-analytics based flow control and SDNs
  - …

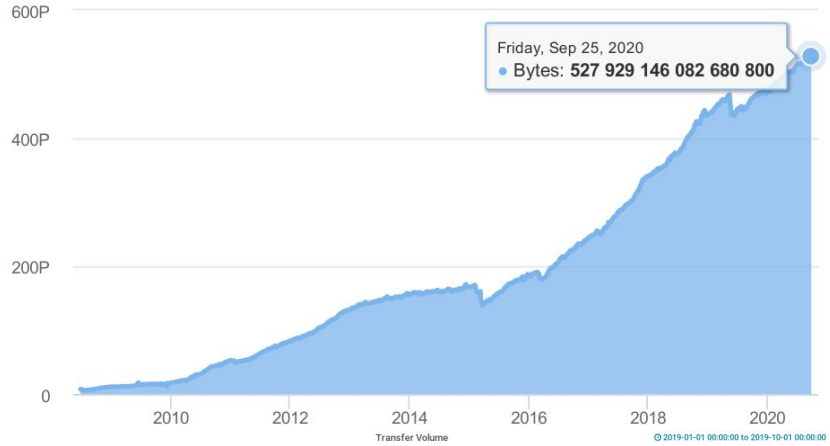
More advanced features

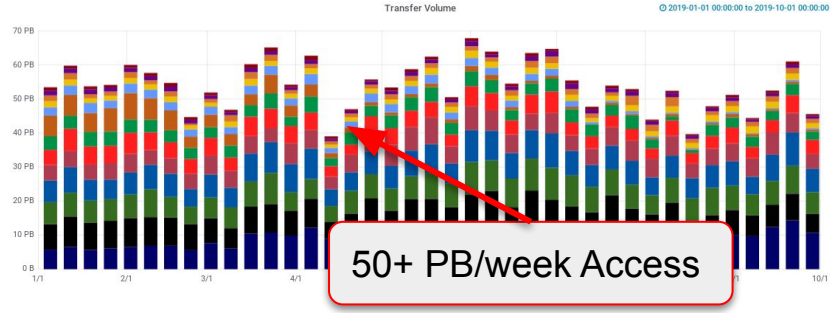


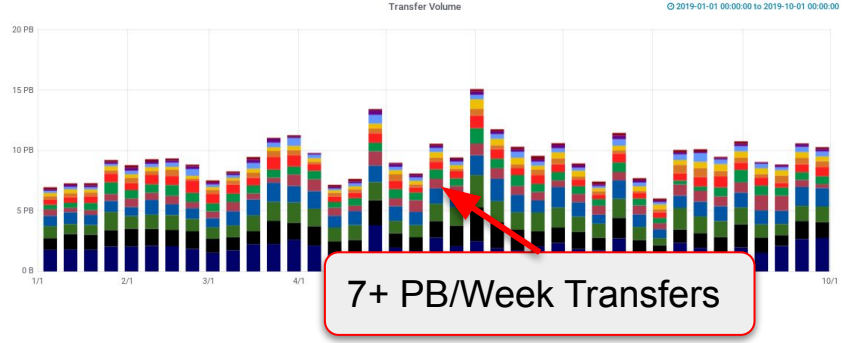
- Rucio is not a distributed file system, it connects existing storage infrastructure
  - No Rucio software needs to run at the data centres
  - Data centres are free to choose which storage system suits them best

Rucio for ATLAS

- A few numbers to set the scale
  - 1B+ files, 500+ 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







System overview

A look back

Discussion and interpretation

*What we're particularly **interested in** for this talk is understanding the **benefits of the common software approach** and **how Rucio became a success.***

*What's the **secret sauce**?*

- Paul J. Laycock

Secret sauce?

Rucio was not started with the intention of becoming a common system!

- We had **a particular problem to solve** and we wanted to solve it in the best possible way
  - The predecessor system DQ2 had **technical and operational limitations**
  - The ATLAS computing model was evolving **from static to dynamic**
  - The gap between adapting DQ2 and **having to reinvent DDM** was too big
- A group of **excited software engineers**
  - Building something new is always **fun!**
  - Especially if these people have **experience operating** the system
  - With an **open-source** mindset

Secret sauce?

- **Early adoption of good practices** and tools in software engineering
  - Distributed coding with **git** → We skipped SVN
  - Mandatory **code review** → No more cowboy coding
  - Mandatory **unit tests** → A validated core system
  - Validated master **release process** → Makes quick deployments possible

- Favour **sustainable long-term** decisions and developments
  - Since we had a quasi-working system, we were **double-developing** on DQ2 and Rucio
  - Long-term Rucio project timeline from 2011 to 2014 allowed us to **do things right without pressure** of having to deliver a particular feature next week
  - Willing to explore and **adopt ideas and paradigms** from communities beyond HEP

Secret sauce?

- Be as open as possible while doing all that
  - Design discussions with **ALICE (AliEN)**, **LHCb (DIRAC)**, and **CMS (PheDEx)**
  - **Continuous integration** with the services provided by **CERN IT**
  - **Integration** of interested people into the development team

- All of this would not have been possible without the great support from our bosses!

- And then… Rucio took over in December 2014!
  - But it was still **missing features** that DQ2 had
  - And quite a **few bits were a bit** *wonky…* despite all the software engineering
  - But the Rucio **platform was ready** for adaptation and extension

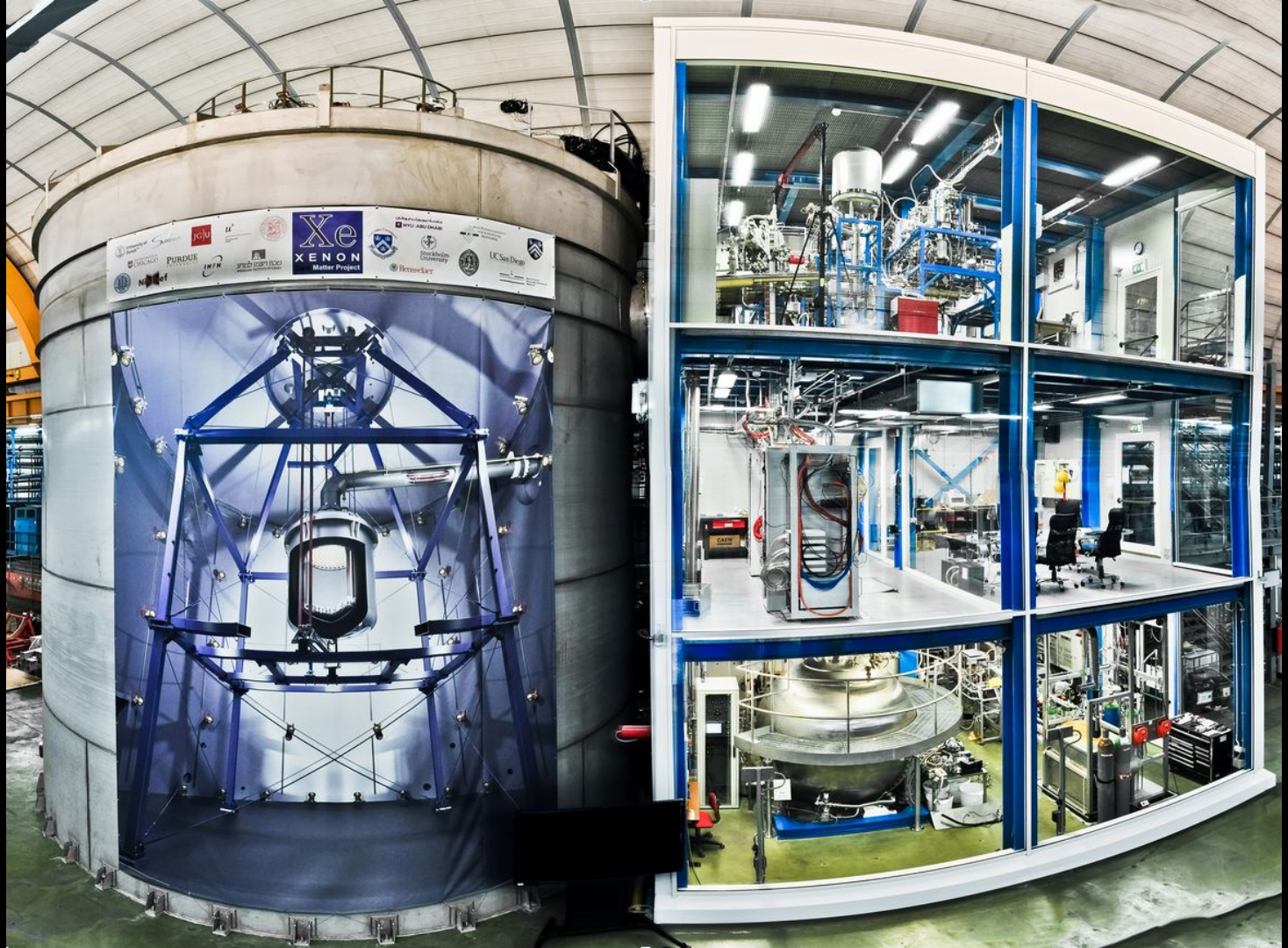
The next 4 years

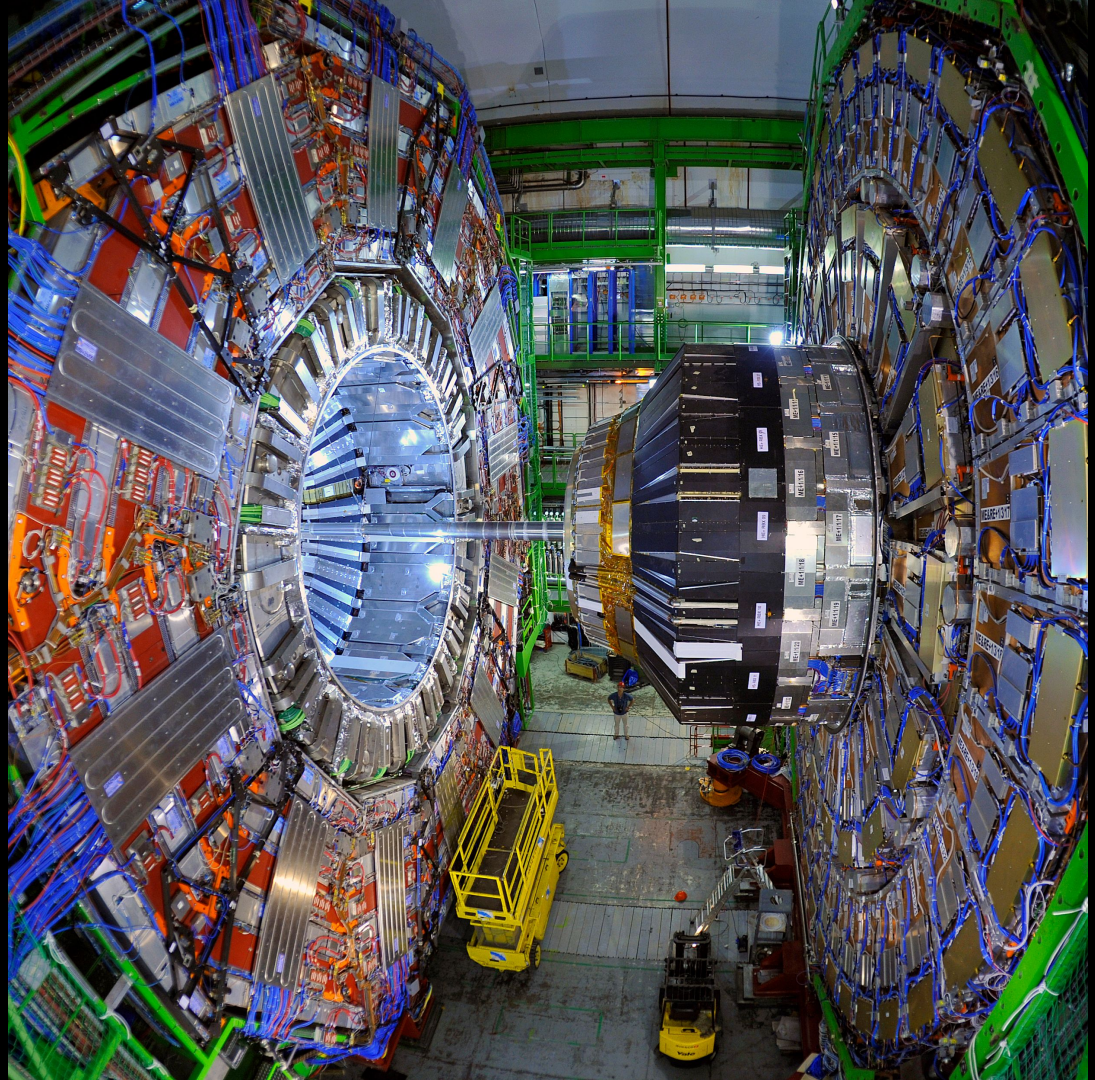
- **Continuous improvements** on the DDM system were now possible
  - With **reasonable effort** by a fraction of the previously needed people
  - **Without a multi-month startup** period for new people to be productive
  - Modular concept eased system integration and **gradual improvements**
  - **Easier deployment** became a large timesaver for central and the sites

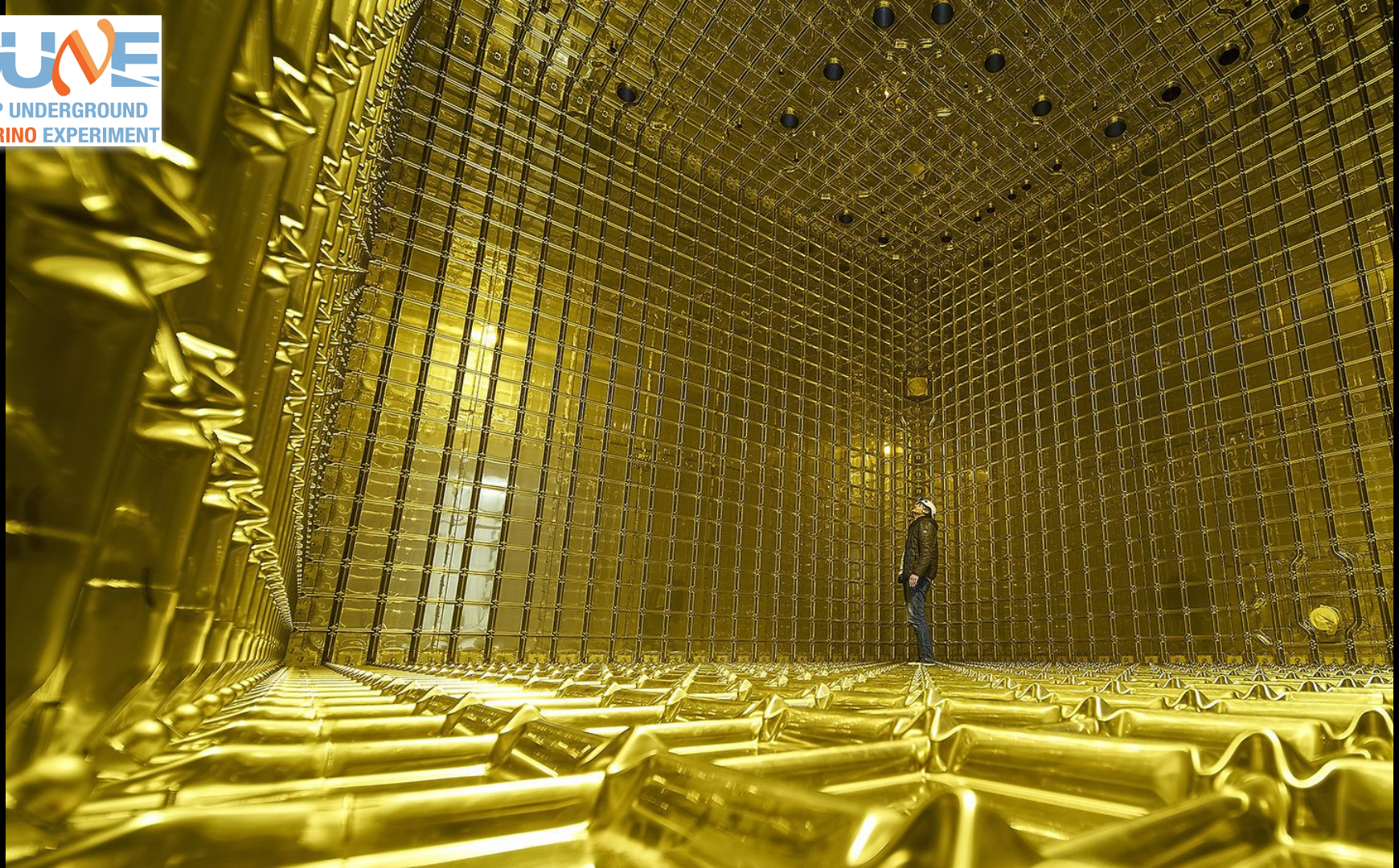
- **Beyond** required technical improvements
  - **Integration of R&D programmes** from PhD students, e.g. networks, clouds, transfer modelling
  - Attending conferences and **conferences beyond HEP**, e.g., Supercomputing or CCGRID
  - Establishing **technical contacts from industry**, e.g., via Google, Amazon, CERN KT

TWGrid ASGC

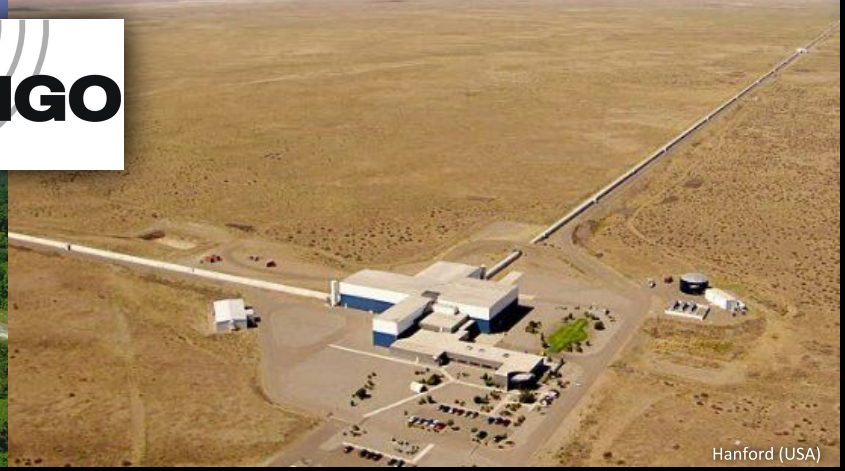
DiCOS
Distributed Cloud Operating System

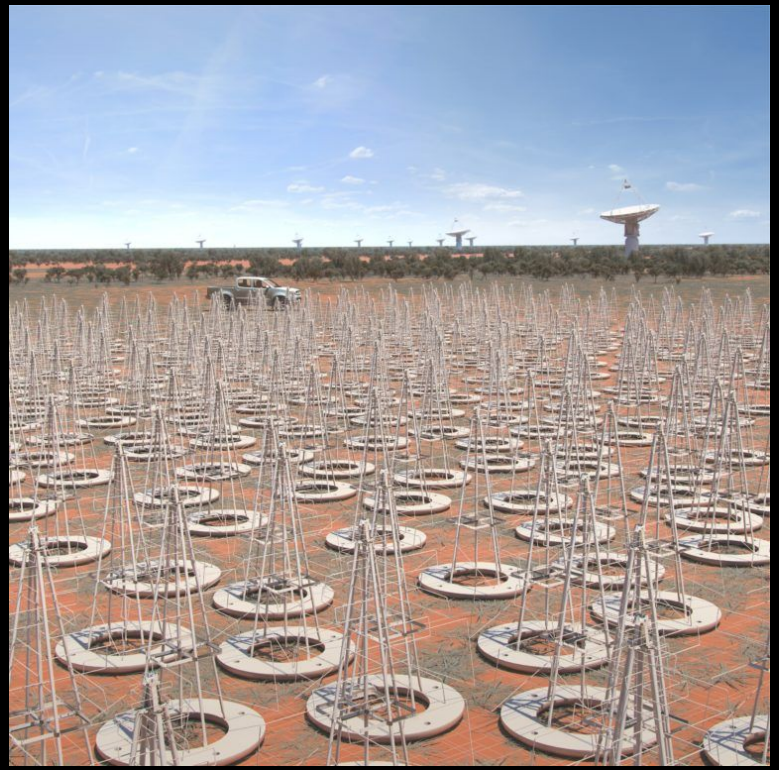

Xe
XENON
Matter Project
PURDUE
UC San Diego
Stockholm University
Princeton




DUNE
DEEP UNDERGROUND
NEUTRINO EXPERIMENT

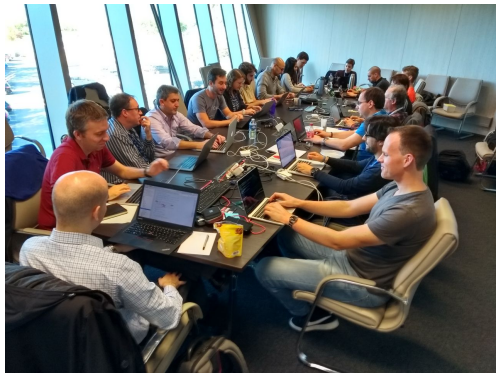
LIGO

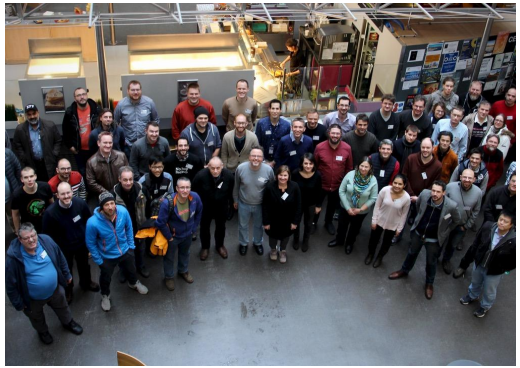




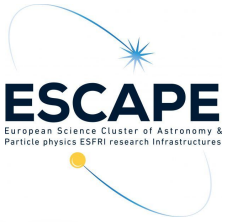
Regular events

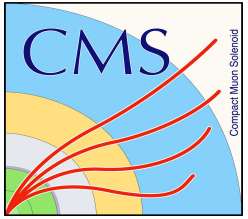
- Community Workshops
  - CERN, Switzerland [2018]
  - University of Oslo, Norway [2019]
  - Fermilab, USA [2020]
- Coding Camps [2018] [2019] [2020]
- Development Meetings [Weekly]

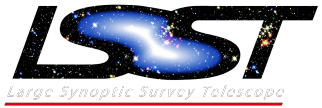




A growing community







System overview

A look back

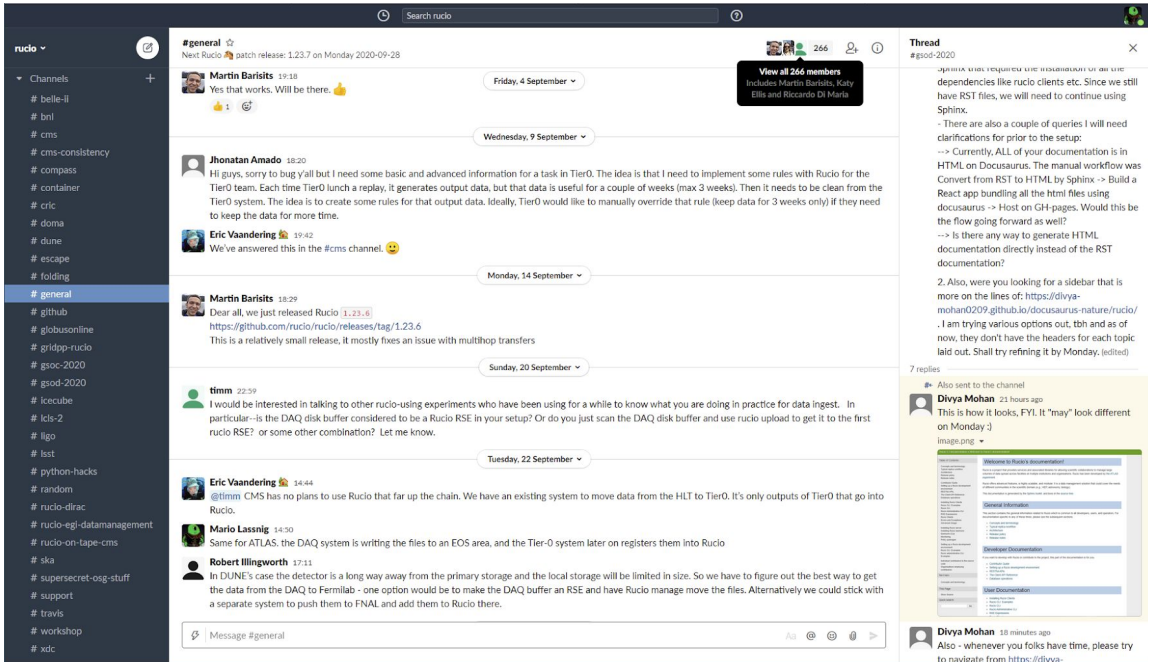
Discussion and interpretation

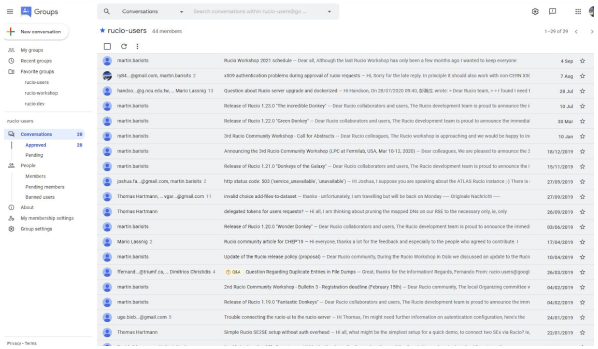
Why a common data management solution?

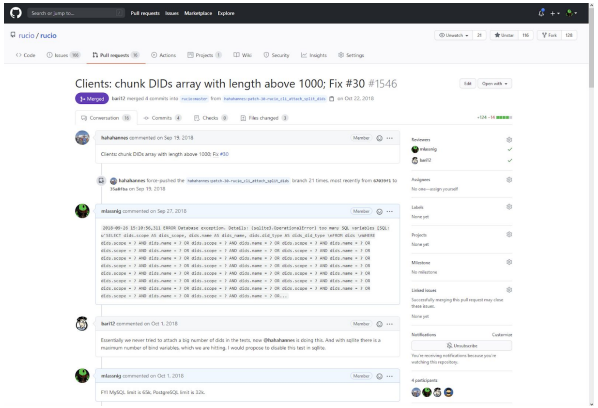
- Shared use of the global research infrastructures will become the norm, especially with sciences at the scale of HL-LHC, DUNE, and SKA
  - Competing requests on a **limited set of storage and network**, data centres will be **multi-experiment**
  - **Compute** seems well-covered, e.g., good scheduling systems, interfaces, and specifications exist
  - **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**

Communication channels







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

- Variety of different topics addressed by focus groups
  - Third-party-copy, Data carousel, Quality of Service, Token-based authn/z, SDN and Networks, …

A few quotes

"... major part of it has been you and the team's willingness to invest the time to help others understand the system"

"... a core which is not ATLAS, even if some of the optional pieces were. ...a willingness to work on them not being so"

"... support for metadata is going to be a big deal for us"

"I like Rucio because it solves a real problem and is not just a product."

"... the general architecture makes it easy to outsource aspects"

"... you basically remove / redistribute the admin burden / labour"

"... the team responsiveness / engagement is a major selling point!"

"The outstanding and friendly support.. ;)"

Summary

- Rucio was built as a scalable and adaptable data management platform
  - It seemed to solve the right problems at the right time
  - Open communication was key
  - Lots of effort from the core team beyond their initial ATLAS commitment
- Several experiments and communities went from evaluation to production
  - Strong participation for support, development, and deployment from many groups
  - Successful integrations with existing software and computing infrastructures
- Emerging strong cooperation between HEP and multiple other fields
- Community-driven innovations to enlarge functionality and address common needs
- Rucio has become a common standard as a collaborative open source project

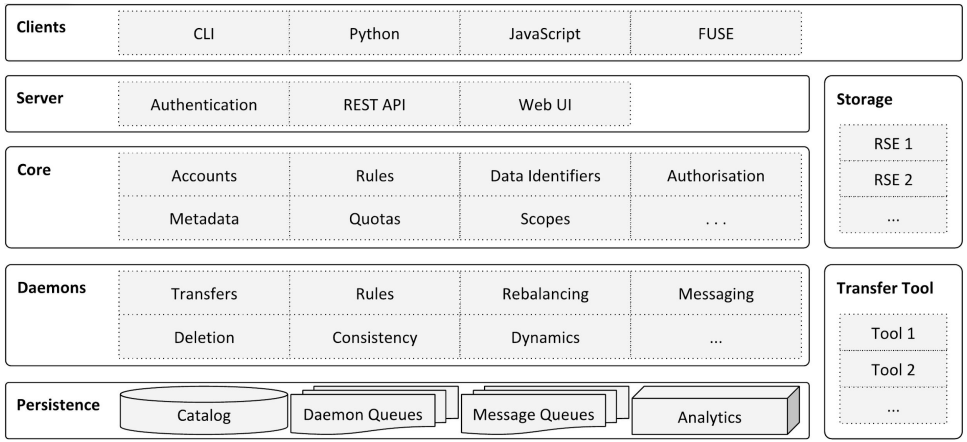
Fresh off the press - IEEE Data Engineering article:
http://sites.computer.org/debull/A20mar/A20MAR-CD.pdf

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
Publications	https://rucio.cern.ch/publications
Twitter	https://twitter.com/RucioData

Backup

Architecture

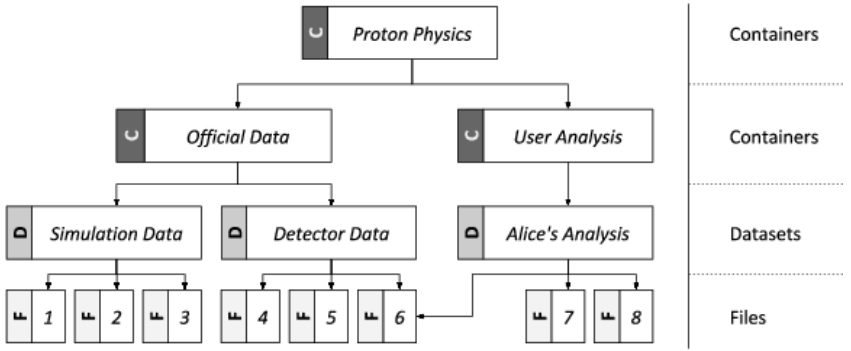


- **Servers**
  - HTTP REST/JSON APIs
  - Token-based security (x509, ssh, kerberos, …)
  - Horizontally scalable
- **Daemons**
  - Orchestrates the collaborative work
    e.g., transfers, deletion, recovery, policy
  - Horizontally scalable
- **Messaging**
  - STOMP / ActiveMQ-compatible
- **Persistence**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

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 - 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 - 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 - 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

Monitoring & analytics

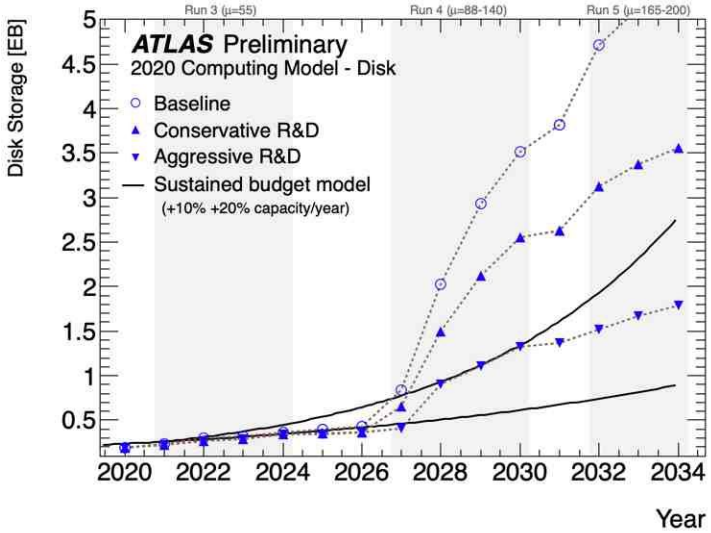
- RucioUI
  - Provides several views for different types of users
  - Normal users: Data discovery and details, transfer requests and monitoring
  - Site admins: Quota management and transfer approvals
  - Central administration: Account / Identity / Site management
- Monitoring
  - Internal system health 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
- LTS releases are always at least 12 month overlapping
- 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

Ongoing topics

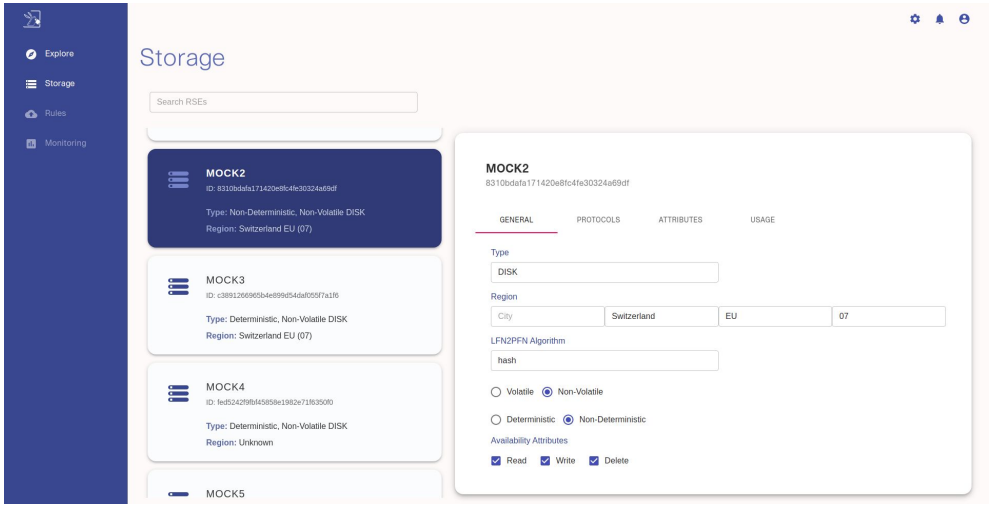
- Tackling the HL-LHC challenge
  - DOMA, ESCAPE, and more!
  - New protocols and third-party-copy
  - Security, authentication, tokens
  - Caching and access
  - Quality of Service & Data Carousel
  - Commercial clouds
  - …
- Better deployment and documentation
  - Docker, Singularity, Kubernetes
  - "Stack Overflow"-like community site
- Adoption by more communities



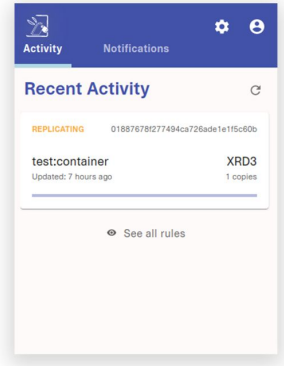
https://indico.cern.ch/event/773049/contributions/3474416/

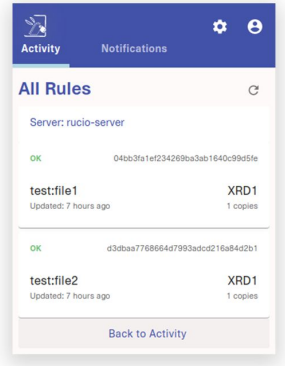
GSoC sneak peek

Rucio Desktop

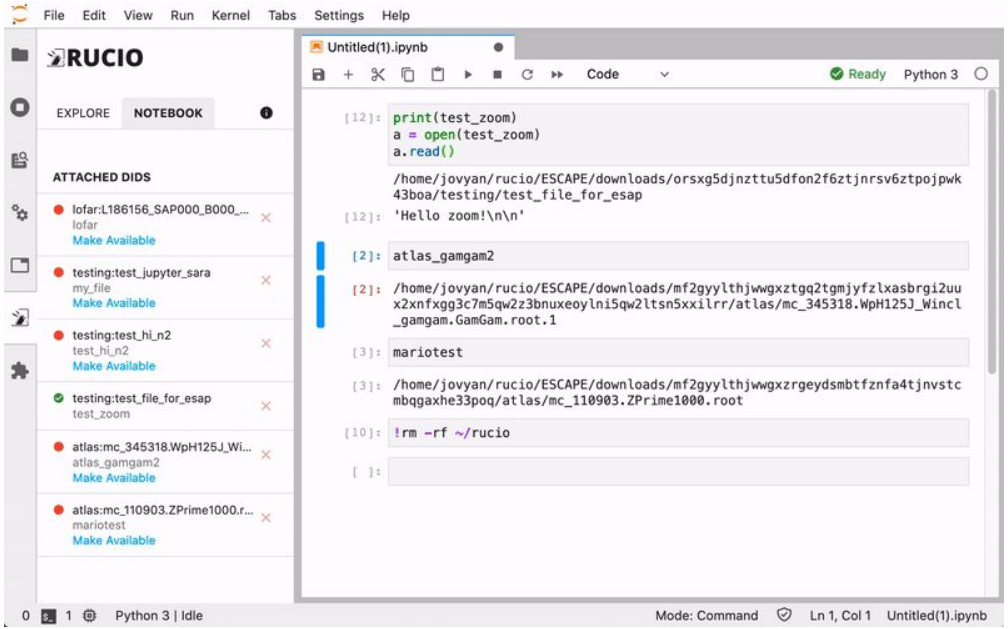


Rucio Notifier





GSoC sneak peek



GSoC sneak peek

```
Checking model availability...
Loading AnswerDetector...
Loading SearchEngines...
DonkeyBot ready to be asked!

CTRL+C to exit donkeybot
ask question: How are Rucio users authenticated?
how many answers: 1
Predicting answers from 2 document(s)...
100%|                                                                                | 2/2 [00:07<00:00,  3.69s/it]
Predicting answers from 2 document(s)...
100%|                                                                                | 2/2 [00:06<00:00,  3.24s/it]
Total inference time: 13.91 seconds

ANSWERS: (descending order)

Question: 'How are Rucio users authenticated?'
Answer: 'Rucio user is authenticated by credentials, such as X509 certificates,
. For more info check: https://github.com/rucio/rucio/blob/master/doc/source/overview_Rucio_account.rst
Confidence: 0.8403105424895546

Question: 'How are Rucio users authenticated?'
Most Similar FAQ Answer: 'You can find the contact information for all authors and contributors to Rucio on:
https://github.com/rucio/rucio/blob/master/AUTHORS.rst
Most similar FAQ question: Who are the Rucio authors?
Author: Vasilis
```