Rucio: State of the Union

Martin Barisits
on behalf of the Rucio team

RUCIO
SCIENTIFIC DATA MANAGEMENT

Overview

- Project organization, communication, and contributions
- Contribution evolution of the last years
- Release plan
- Selected developments from 2020 & 2021
  - Lots of important contributions, due to time I cannot mention all of them
    - In no way is this a judgement on importance or complexity of the contribution
- Development outlook for 2022

Organization 1/2

- Discussions in weekly Rucio meetings
  - News, DevOps roundtable, hot topics, developers roundtable
  - Planning meeting (3-4 month plan) for each feature release
  - Everyone is welcome to join!
- Component leads take formal responsibility of maintenance of their components
  - Plan patches and features, guide new contributors, review code
  - All contributions are welcome!
- Yearly Rucio community workshops
- Yearly Rucio coding camps
  - Introduce new developers to the project; Spend some focused days on discussing ideas and implement them
  - Unfortunately none in 2020 & 2021 due to ongoing pandemic

Organization 2/2

- Communication
  - Slack
    - Open to everyone: Users, Operators, Developers
    - 350+ users
  - eMail lists
    - rucio-dev@cern.ch for developer contact
    - rucio-users@googlegroups.com for users, not massively used - most people head to Slack
- Special Interest Groups
  - Launched two SIGs this year
    - Forums to specifically discuss ongoing activities of these areas
    - Open to anyone (organized via public CERN e-groups, message me if you want to join without CERN account)
  - RUCIO-SIG-Metadata - Metadata evolution in Rucio
  - RUCIO-SIG-QoS - Quality of Service evolution in Rucio (and beyond)

Contributions (to the main repository)

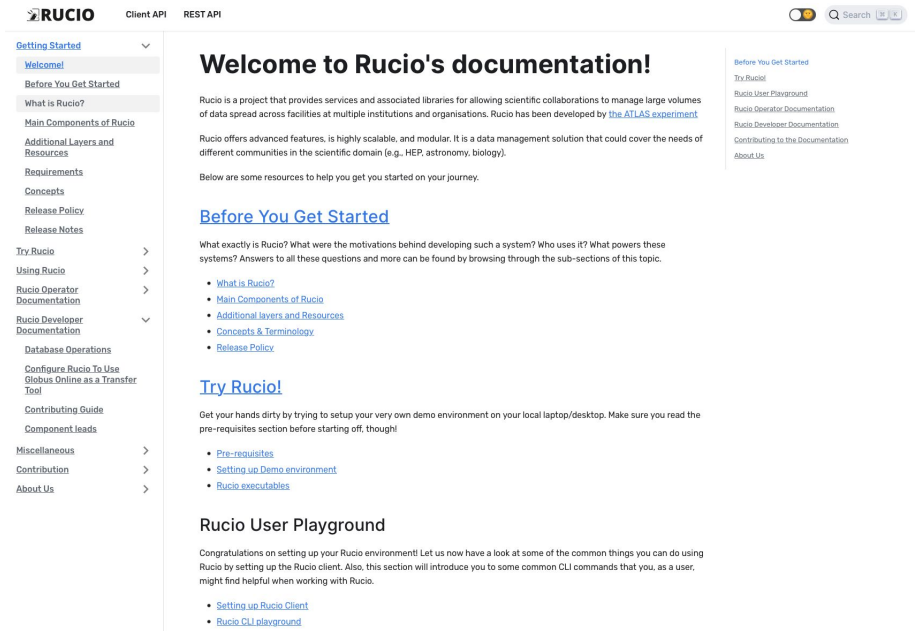
- 2017
  - **540** commits, **33k** LOC, from **12** contributors
- 2018
  - **775** commits, **64k** LOC, from **29** contributors
- 2019
  - **954** commits, **53k** LOC, from **31** contributors
- 2020
  - **559** commits, **54k** LOC, from **29** contributors
- 2021 (so far)
  - **344** commits, **39k** LOC, from **32** contributors
- Commits and LOC not a good indicator to judge complexity of contributions
- Top-contributors produce the majority of the code

Release plan

- 2019
  - **1.19** “Fantastic Donkey” February 2019
  - **1.20 LTS** “Wonder Donkey” June 2019 **EOL 07-2021**
  - **1.21** “Donkeys of the Galaxy” October 2019
- 2020
  - **1.22** “Green Donkey” March 2020
  - **1.23 LTS** “The Incredible Donkey” June 2020 **EOL 07-2022**
  - **1.24** “Aquadonkey” November 2020
- 2021
  - **1.25** “Rat-Donkey” March 2021
  - **1.26 LTS** “Donkey League of La Mancha” July 2021 **EOL at least 07-2023**
  - **1.27** “Batdonkey v Superdonkey” November 2021
- Long Term Support model works well - LTS releases actively used by the community

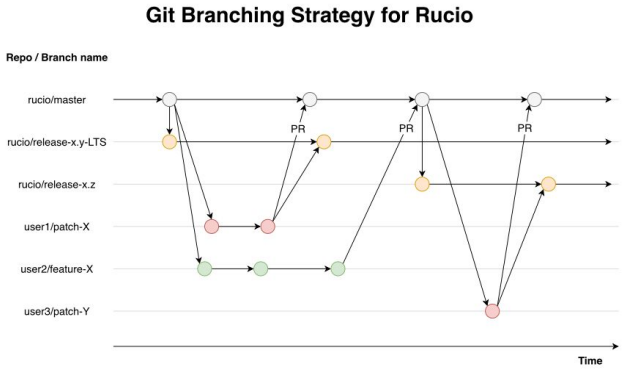
Documentation

- New documentation https://rucio.cern.ch/documentation/
  - Markdown based
  - Separate repository to contribute https://github.com/rucio/documentation
  - Initial work done by Google Season of Docs student
- Old RTD documentation still available
  - Will be removed eventually (TBD)
- Very easy to contribute now!
  - Please do so!



New contribution model and testing

- Previous contribution model was complex and difficult for newcomers
- Changed contribution model to a workflow similar to most other open source projects
  - One contribution branch (master)
  - Commits are cherry-picked to release branches by librarian
    - Shifts complexity from developer to librarian
- Switched all CI testing to github actions (From Travis)
  - Several parallel test suites
  - Integration testing and unit testing



New transfer system

- Transfer system evolved massively over the last years
  - New FTS features
  - Multi Hop
  - New transfertools (Globus ~~Online~~)
- This evolution introduced issues since it was not anticipated in the original design
- Incremental re-write of the entire system (still ongoing)
  - Separate transfertool (FTS, GO) specific code from the core-transfer system
  - Make Multi Hop a primary citizen of the transfer logic
  - Use common code path for stage-in and all transfers
  - Testing
- In the future
  - Transparently multihop between transfer tools (eg. FTS to GO)

Lightweight clients and policies

- Rucio clients come with a long list of dependencies
  - Some were conflicting in the environments of some communities
- Massively reduced the python dependencies in the clients
  - Now dependencies are loaded based on the required use-case (Optional extras)
- Will possibly further reduce the dependencies if needed
  - However, the current required dependencies are all well-tested, widely compatible and widely used python projects
- Policy packages
  - Now also available for Multi-VO installations
  - Eventually will remove hard-coded policies (ATLAS, CMS, …) from the core repository
    - Should be fully replaced by a community-managed policy package similar to DUNE

Python 2 and Python 3

- Rucio 1.24 release dropped support for Python 2 on the server side
- Clients are still Python 2.7 compatible
  - Rucio server ⟶ py3
  - Rucio clients ⟶ py2.7, py3
- If your environment requires Python 2.7, the 1.23 LTS release line will be supported until **07-2022**
- Need to discuss deprecation of python2.7 for the clients
  - Which communities still need it?

Rucio - DIRAC Interface

- RucioFileCatalog plugin was added to DIRAC
  - Enables communities using DIRAC as a WFMS to use Rucio as a DDM system
  - Used by Belle II in production
- Mostly uses existing Rucio REST endpoints
  - One Rucio REST endpoint was added to support a specific atomic DIRAC workflow
- More on this in tomorrow's WFMS session

Reaper 2

- Reaper is the deletion agent of Rucio, responsible for all physical replica deletion on storage
- Due to scalability issues with reaper1, it was re-written to address these issues
  - More dynamic and distributed model
  - Continuously assess workload situation and prioritizes accordingly
- Reaper2 was available in parallel to reaper1
- Rucio 1.25 release fully deprecated reaper1
  - Reaper2 was renamed to reaper

MultiVO

- Enable one Rucio instance to host multiple Virtual Organizations (VOs)
- Goal was to adapt Rucio code without impacting SingleVO functionality
  - Both in functionality as well as maintainability of the code
  - This was done by addressing most MultiVO functionality in the REST/API layer of Rucio, while leaving lower level functionality (Rucio core) the same for both ways of running Rucio
- More on this in the “Long tail of science” session on Thursday

Metadata unification

- Original Rucio metadata implementation was very much driven by the needs of ATLAS
  - Limited metadata information in the Rucio catalog, since ATLAS has it's own full-fledged metadata system
  - Metadata stored in hard-coded database columns (Very much HEP focused)
- Generic JSON-based metadata functionality was added to Rucio later on
- This development introduced a metadata plugin-approach in Rucio
  - Queries/Insertions are made to one REST interface, Rucio relays the requests to the right plugins
    - Eg. column-based-plugin, json-plugin, etc.
  - Communities can introduce their own plugins to connect to their own metadata systems
    - E.g. MetaCat project for DUNE
- More on this in the Astronomy session & metadata panel

Community chep paper

- Published "Experience with Rucio in the wider HEP community" at CHEP 2021
- Includes contributions from
  - ATLAS
  - Belle II
  - CMS
  - ESCAPE
  - IGWN
  - LDMX
  - Folding@Home
  - MultiVO Rucio

Misc.

- Moved the entire web framework from Web.py to FLASK
  - Lack of maintenance of web.py, large community behind FLASK
  - Rucio 1.26 now only supports FLASK
- Scalability
  - Several scalability issues with Rule creation were addressed
    - Rule creation was memory limited, thus very large rules would not succeed
    - New rule algorithms are memory constant
- Containers & Helm charts
  - Dockerhub dropped support for free auto-builds, even for open-source communities
    - All Rucio container builds are now done by GH Actions
  - Introduced versioning for helm-charts, in order to offer compatible helm charts for LTS releases
    - Helm charts of version 1.26 are compatible with (any) Rucio version of 1.26 etc.
      - Only true for major version, no need to match minor version (1.26.x≠1.26.y)

Google Summer of Code

- Rucio involved in GSOC since 2017
  - Very good experience with very bright students
- 2020
  - Native Desktop Application for Rucio - Vivek Nigam
  - Integration of Rucio in JupyterLab for SWAN - Muhammad Aditya Hilmy
    - More on this in the ESCAPE Data Lake talk tomorrow
  - Support for Rucio Users with Native Language Processing - Vasilis Mageirakos
- 2021
  - Rucio and CS3API to enable data management for the ScienceMesh cloud - Rahul Chauhan
    - More on this in the CS3 talk tomorrow
  - New protocols for exascale data management with Rucio - Rakshita Varadarajan

Token workflows 1/2

- Large shift in our community/infrastructure from X509 to JWT authz
  - Involves the entire stack: Storage, Compute, WFMS, DDM, Users
- Rucio supports OIDC JWT since 1.22
- However, most of this support was done having X509 concepts in mind
  - User uses 1 token instead of 1 certificate - client passes token instead of proxy
    - Token is used for all interactions: auth to Rucio, auth to storage
    - Naturally these tokens are very powerful
  - Permissions are largely enforced by IAM (VOMS) or Storage

Token workflows 2/2

- In the future Rucio needs to take a more central role in these workflows
  - Rucio will be in charge of enforcing permissions
  - Rucio will give limited, specifically scoped tokens to the client for storage interaction
    - Tokens are only useful for this specific download/upload
- These workflows introduce new complexities
  - Can users directly interact with storage without interacting with Rucio?
  - VOs might deal with these workflows differently, Rucio needs to support that
- But also give us unique opportunities we did not have before
  - Rucio can actually prevent storage access to embargoed data!
  - Enforce real data access permissions
- More on this in Andrea Ceccantis talk this afternoon

Quality of Service

- The need for storage quality of service is voiced by almost all data-heavy communities
  - Details are however less clear
- Many complexities on different levels: conceptual, organizational, technical
  - What QoS “classes” are needed beyond Disk and Tape?
  - Which data workflows would use which classes, when would they transition?
  - How do we account these pledged ressources, since hardware cost might be significantly different?
  - How does an orchestration engine (Like Rucio) communicate QoS transitions? Which protocol?
  - …
- Progress very difficult due to the intermix of these issues
  - Entire community, across experiments, needs to be involved: Storage (Sites), Data Management, Workflows

Quality of Service 2/2

- With Rucio we follow an incremental approach to QoS
  - Discussions are happening in the Rucio-SIG-QoS
  - Pick one use-case, prototype it with available resources and technologies, test it and show outcomes
- As the first use-case we picked “storage-managed QoS” with the BNL MAS system which is currently ongoing
  - Tape system with very large disk buffer
  - Rucio replication rules will transfer data to disk
  - MAS is allowed to transition the data to tape after a certain duration
    - Selection criteria is based on popularity/access metrics
  - New replication rules, for existing data, will transition it back to disk (until duration is met)
  - This usecase can be fully developed with existing technologies, no new QoS protocols are needed
  - More on this in Matt Snyder’s talk tomorrow

Thanks for participating and let’s get started!