Software management: un servizio di distribuzione del software basato su CernVM-FS

Data management: il modello di datalake per la federazione di storage geograficamente distribuiti

Giada Malatesta
giada.malatesta@cnaf.infn.it

20 febbraio 2025

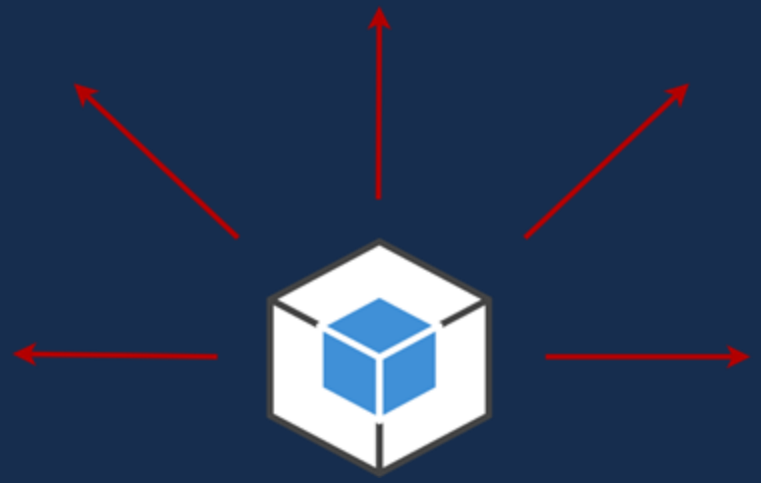
Outline

Software Management

- Software distribution challenge
- The Software Management @DataCloud solution
- Workflow overview
- User perspectives

Data Management

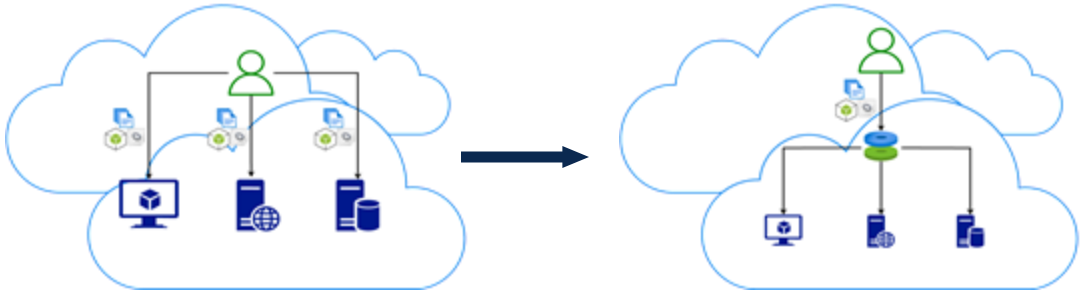
- What is the datalake
- What is RUCIO
- Use case
- Next steps



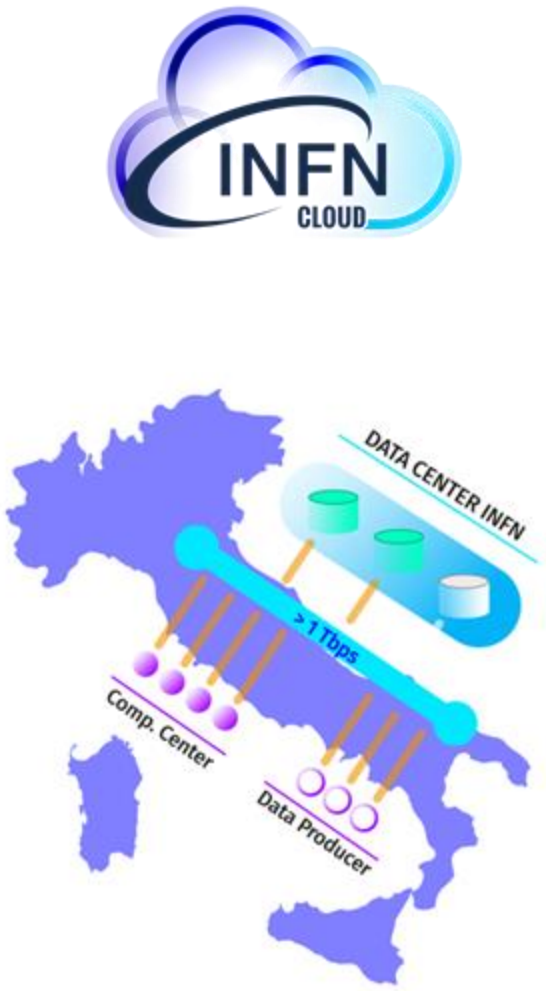
Software Management

Software distribution challenge

- In a **distributed and heterogeneous environment** the **sharing** of software, libraries, configurations and container images in an effective, user-friendly and transparent way can be **challenging**.
- There are already low-level solutions that address this challenge.
- The aim is to further simplify the adoption of a well established technologies such as Cern-VM File System (CVMFS) in a highly **multidisciplinary** environments.



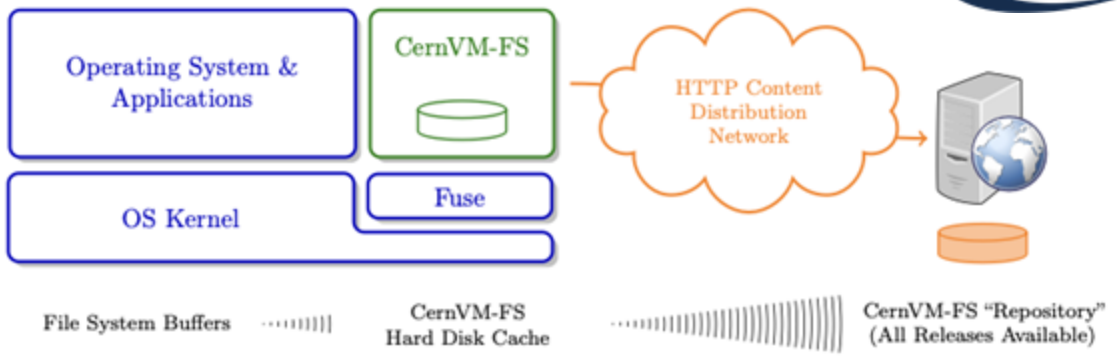
Software Management @DataCloud solution

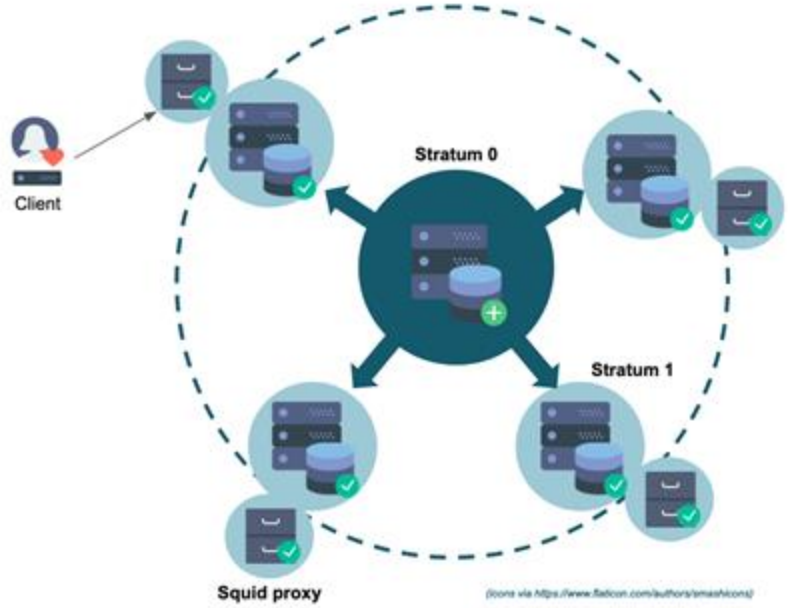


- In order to cope with the challenge in a **Cloud infrastructure**, such as our DataCloud, we implemented a **Software Management service**.
- We build on top of a well established technology known as **CernVM File System** (CVMFS).
- **Abstraction**: what the project adds, is to avoid to know any technical details about CVMFS mechanisms providing **abstractions** in order to let the user accessing the repository in a **simple** and completely **transparent** way.
- **Automation**: in other words we enable the possibility to copy software, libraries and related dependencies, small files, configuration files etc in **S3 cloud storage** and that's it.

CernVM File system

- It's an **open-source**, usable and **customizable** software distribution service.
- It's a network file system implemented as a **POSIX read-only file system**.
- Files and directories are hosted on **standard web servers** and mounted in the universal namespace ***/cvmfs***.
- It uses standard **HTTP transport**, avoiding most of the firewall issues.
- It is a **read-only** files system for those who access it, only the **admin** is able to **modify** its content.





Adopted technologies

CernVM File system

The CernVM File System provides a **scalable**, **reliable** and **low-maintenance software distribution service**. CernVM-FS is implemented as a POSIX read-only file system in user space (a FUSE module). Files and directories are hosted on standard web servers and mounted in the universal namespace /cvmfs.

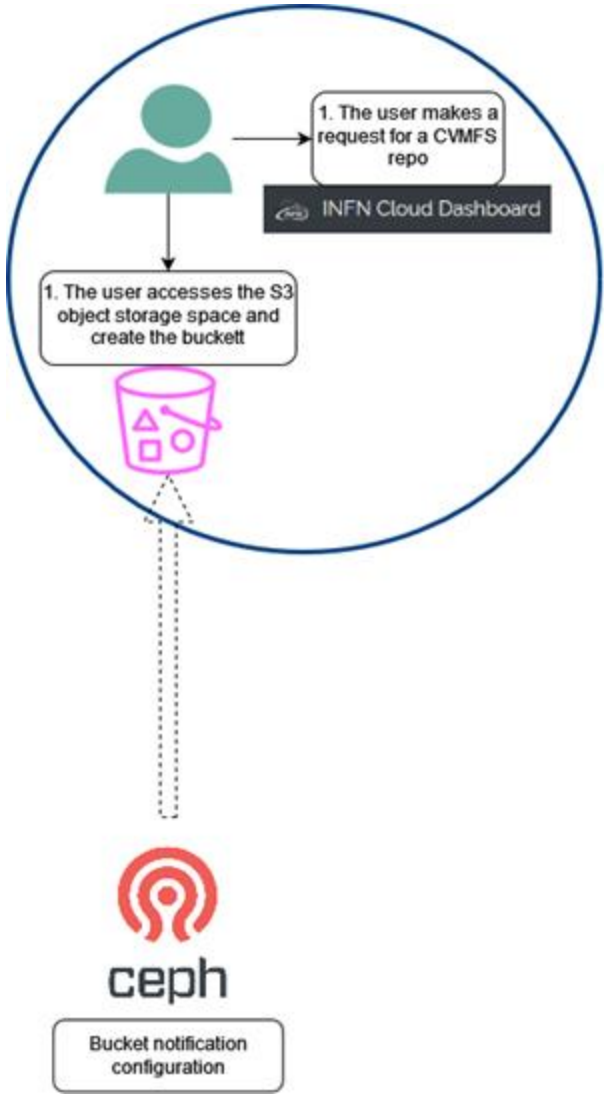
Ceph is an **open-source**, **distributed storage system**.

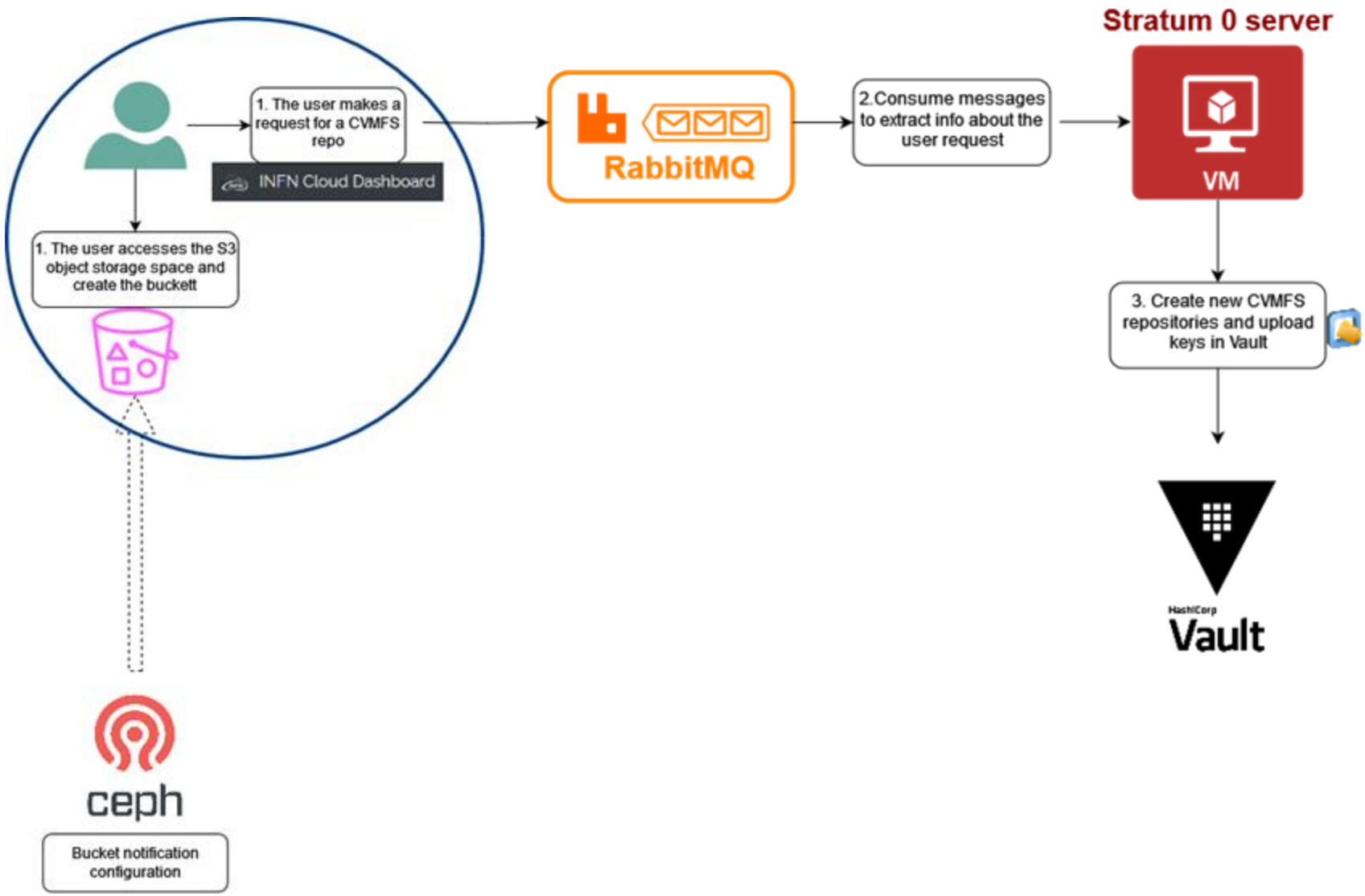
RabbitMQ provides an **open-source**, **reliable**, **scalable** platform for **message delivery**, through features like message acknowledgements, persistence, routing.

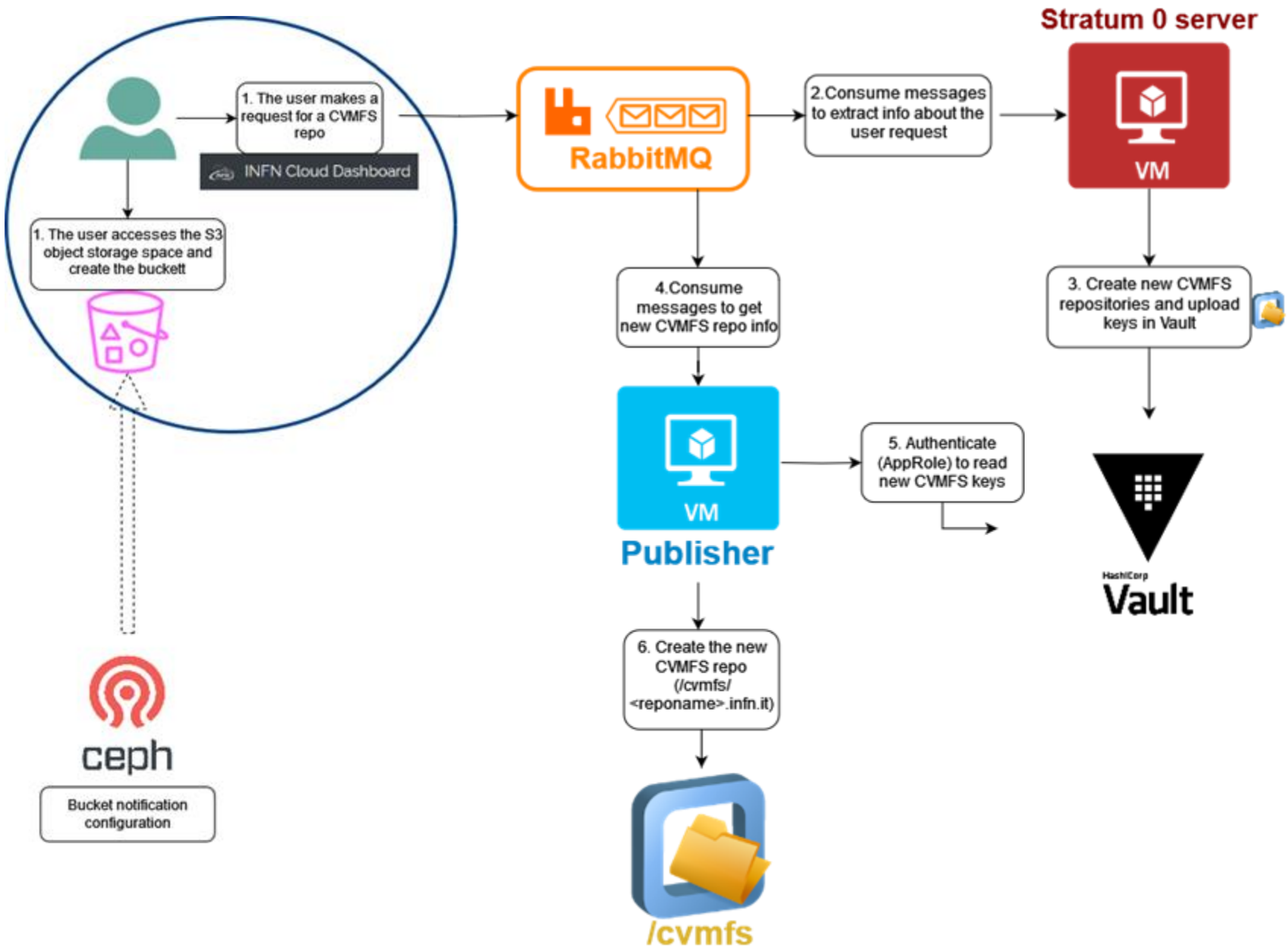
Vault provides organizations with identity-based security to automatically **authenticate** and **authorize** access to **secrets** and other sensitive data.

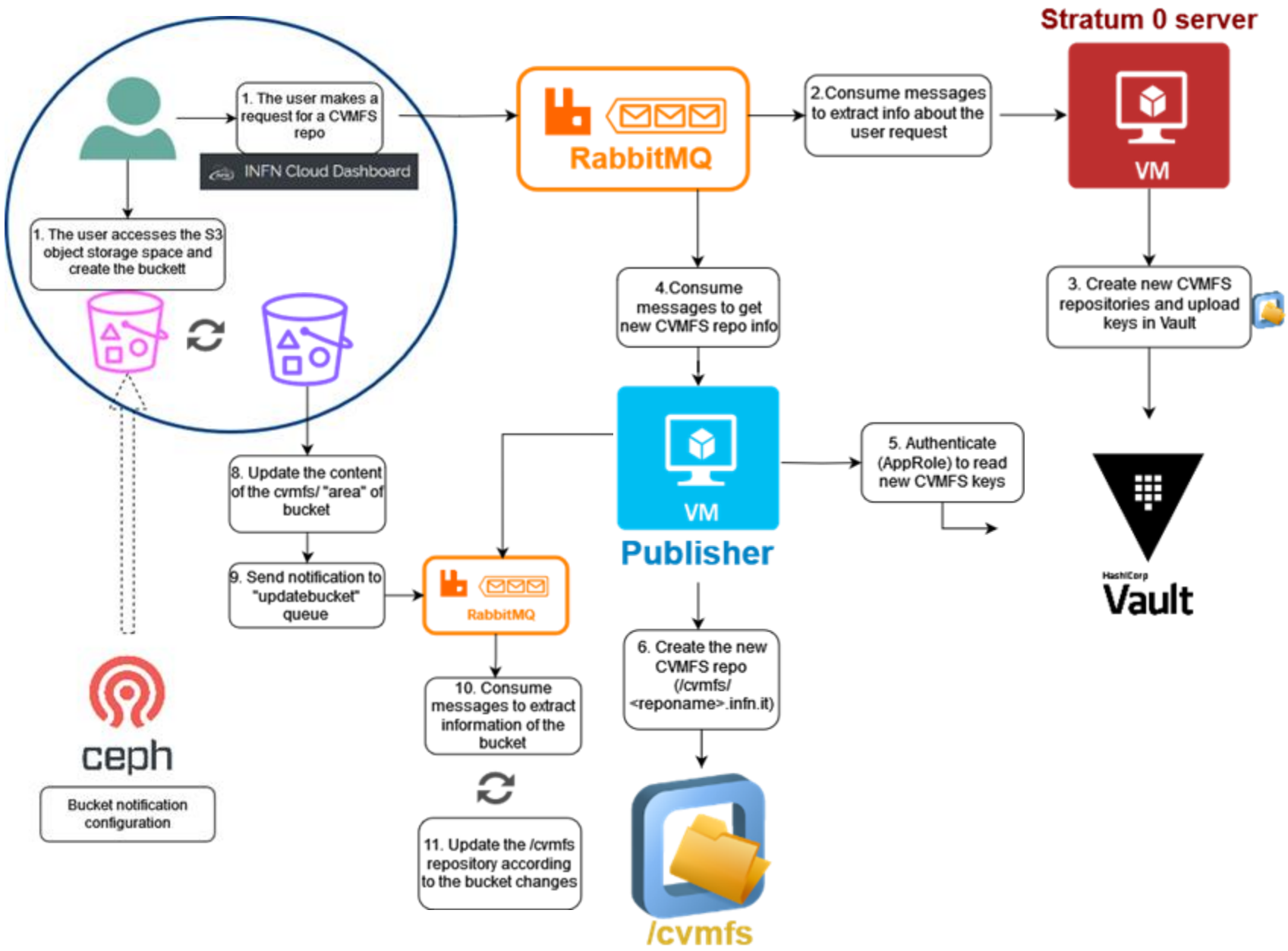
Software distribution: workflow overview

- The user **requests** a CVMFS repository (personal or group) via the **INFN Cloud dashboard**.
- The request is sent to **RabbitMQ** and is elaborated in order to create the repository.
- Once created, the relative **keys** are published in a **Vault system.**
- The user accesses the **S3 object storage** space and creates a **bucket** (personal or group).
- He **uploads** what he wants to **distribute** in a specific area of the bucket named *cvmfs.*
- The S3 bucket service system sends a message to RabbitMQ so that the system get **notified** and can **synchronize** the content of the correspondent CVMFS repository.
- At this point, the user can access the **CVMFS client** in **read** mode to the **distributed** software.
- Expert users can still use the CVMFS mechanisms to publish their software through CVMFS remote publisher.


1. The user makes a request for a CVMFS repo
INFN Cloud Dashboard
1. The user accesses the S3 object storage space and create the buckett
ceph
Bucket notification configuration

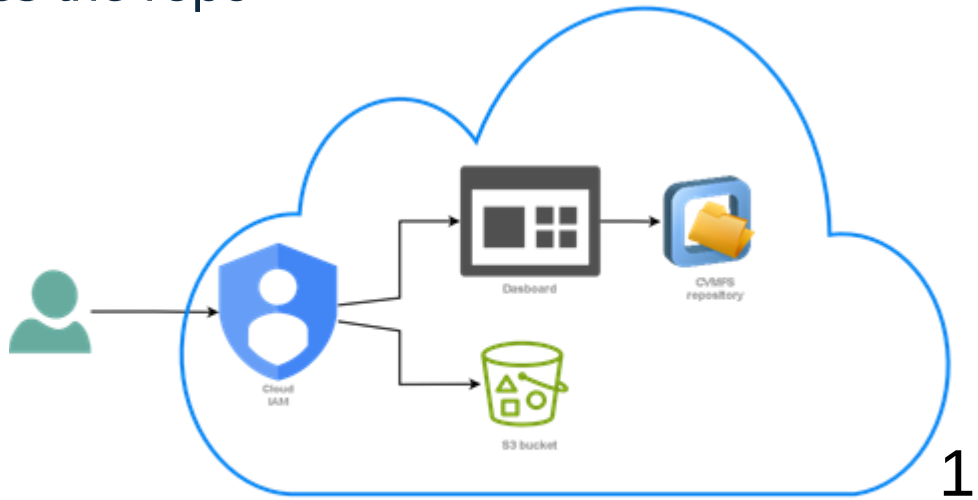

1. The user makes a request for a CVMFS repo
INFN Cloud Dashboard
1. The user accesses the S3 object storage space and create the buckett
ceph
Bucket notification configuration
RabbitMQ
2.Consume messages to extract info about the user request
Stratum 0 server
VM
3. Create new CVMFS repositories and upload keys in Vault
HashiCorp
Vault


1. The user makes a request for a CVMFS repo
INFN Cloud Dashboard
1. The user accesses the S3 object storage space and create the buckett
ceph
Bucket notification configuration
RabbitMQ
2.Consume messages to extract info about the user request
Stratum 0 server
VM
3. Create new CVMFS repositories and upload keys in Vault
HashiCorp
Vault
4.Consume messages to get new CVMFS repo info
VM
Publisher
5. Authenticate (AppRole) to read new CVMFS keys
6. Create the new CVMFS repo (/cvmfs/<reponame>.infn.it)
/cvmfs


Stratum 0 server
1. The user makes a request for a CVMFS repo
INFN Cloud Dashboard
RabbitMQ
2.Consume messages to extract info about the user request
VM
1. The user accesses the S3 object storage space and create the buckett
4.Consume messages to get new CVMFS repo info
3. Create new CVMFS repositories and upload keys in Vault
8. Update the content of the cvmfs/ "area" of bucket
VM
Publisher
5. Authenticate (AppRole) to read new CVMFS keys
HashiCorp
Vault
9. Send notification to "updatebucket" queue
RabbitMQ
6. Create the new CVMFS repo (/cvmfs/ <reponame>.infn.it)
10. Consume messages to extract information of the bucket
ceph
Bucket notification configuration
11. Update the /cvmfs repository according to the bucket changes
/cvmfs

User perspectives

- The user doesn’t need to know CVMFS, he needs a token and an S3 DataCloud account.
- The user must be **authenticated** through JWT based auth N/Z (based on IAM) to access the Cloud Dashboard.
- The user can **request** a personal CVMFS repository via **dashboard** with one click.
- Access to the **CVMFS repository keys**: they can be easily **downloaded** from the dashboard to configure the **CVMFS client** to access the repo in **read-only** mode.
- The user must have access to the **backbone S3 object storage** to upload Software.



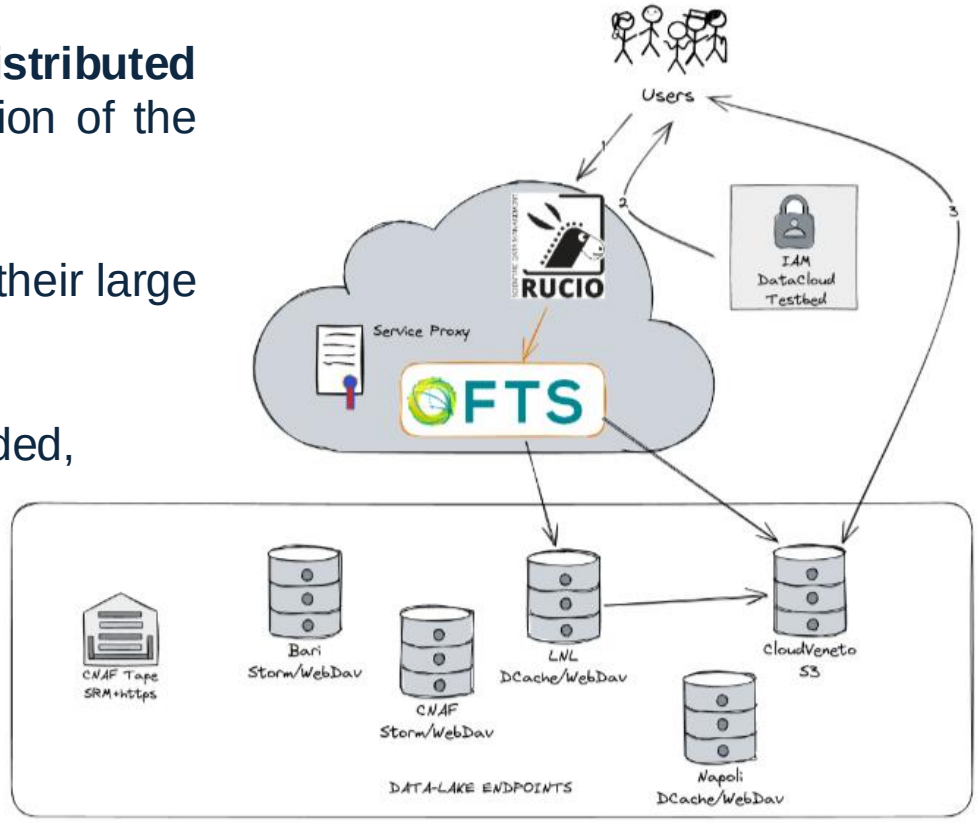
Summary

- Both **abstraction** and **automation** of the underlying CVMFS system are successfully provided by the presented **Software Management service**.
- **Abstraction**: users do not need to know the details of CVMFS, they just **upload** the **software** in their **bucket**.
- **Standard** CVMFS: to expert users is left the possibility to distribute software through a CVMFS **publisher**.
- **Unpacked**: users can use CVMFS to distribute **unpacked container images** via the Harbor registry.
- The Software Managment service is an **open-source** service that can adopted by both single user and group of research.

Data Management

Data Management

- The **federation** of distributed **storage** in the national territory has been discussed extensively in recent years.
- Within INFN, several experiments have been made to federate **distributed** and **heterogeneous** storage and the results is the implementation of the **Datalake model**.
- **Rucio** and **FTS** are the **data management tools** selected due to their large use in some large experiments.
- What we did, was to study how this model can be modified, if needed, in order to be applied in **life-science** use case.
- Having to handle **sensitive data** made us face a significant challenge, since it is important to keep in mind concepts such as **privacy**, **confidentiality** and **information security.**

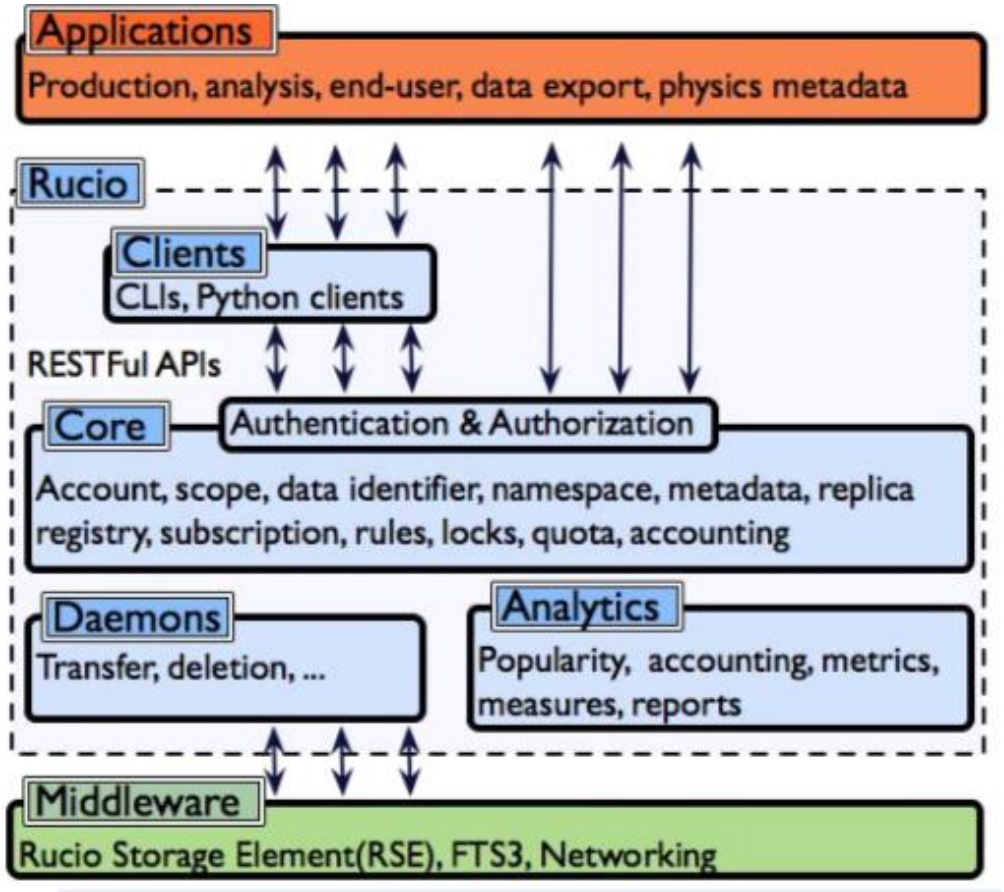


https://agenda.infn.it/event/34683/contributions/197352/attachments/105520/148353/CCR%20-%20Federare%20lo%20storage%20distribuito%20nazionale.pdf

What is Rucio?

- Rucio enables **centralized managment** of large volumes of data **distributed** over **heterogeneous** storage backends.
- Rucio interoperates with heterogeneous storage infrastructures, providing an **interface** that allows users to interact with the storage backends in a **unified way**.
- Rucio enables to **upload**, **download**, and to **declarative manage** groups of files.
- Declarative management is the power of Rucio: users can define **high-level rules** such as "Keep 3 copies on 2 different places".
- If one copy is lost, it will be **automatically re-constructed**.
- Rucio was originally developed to meet the requirements of the high-energy physics experiment ATLAS.
- Now is continuosly extened to support LHC experiments and other diverse scientific communities.

Main Components



Client: consist of components such as the CLI, Python clients, that allow users to interact with Rucio.

Server: manages authentication and provides a common API for interaction with the clients and other external application (the Web UI)

Daemons: are processes in background that take care of the continuous and synchonous workflows.

Persistence layer: Rucio uses PostgreSQL to keep all the logical data and the application states.

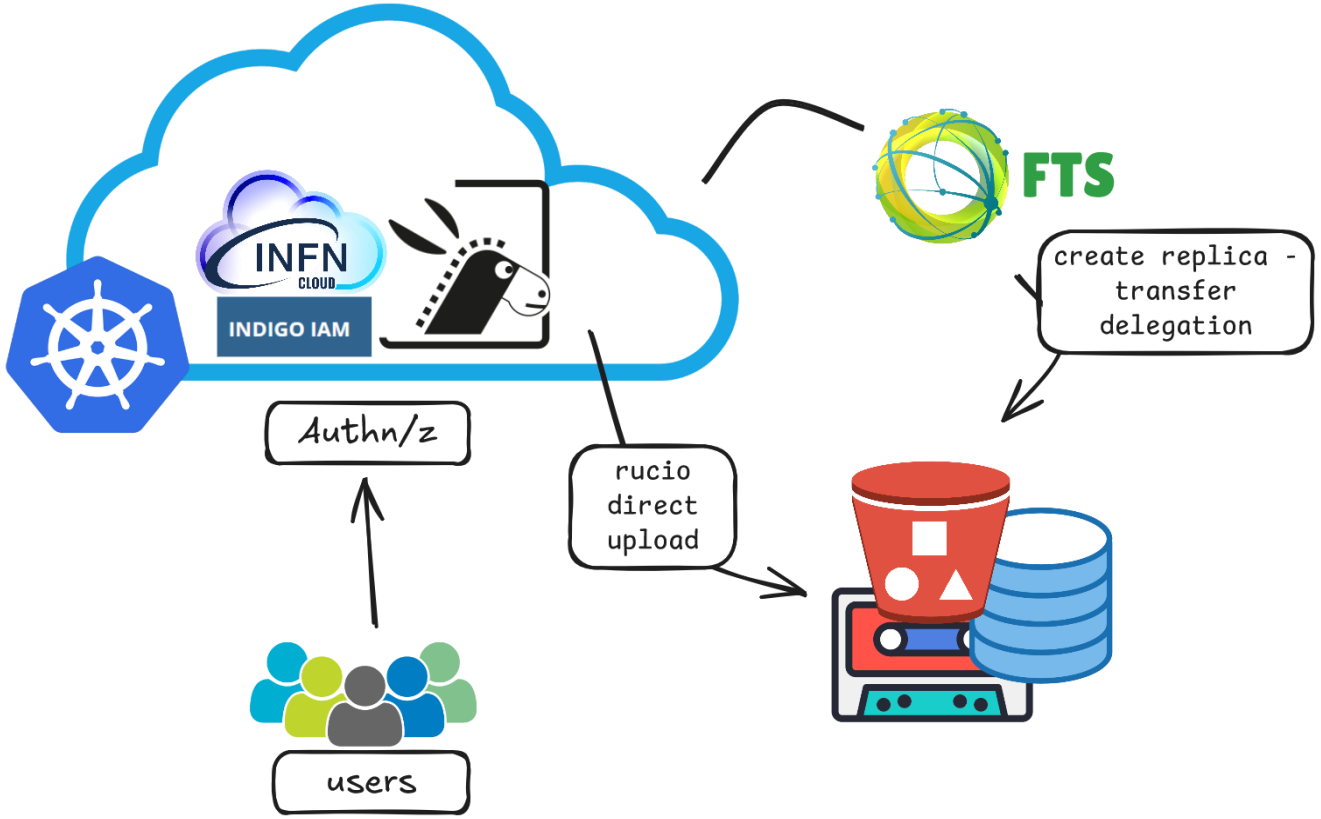
Next to these layers, there are the **storages** resources, and the **transfer tools.**

Rucio deployment

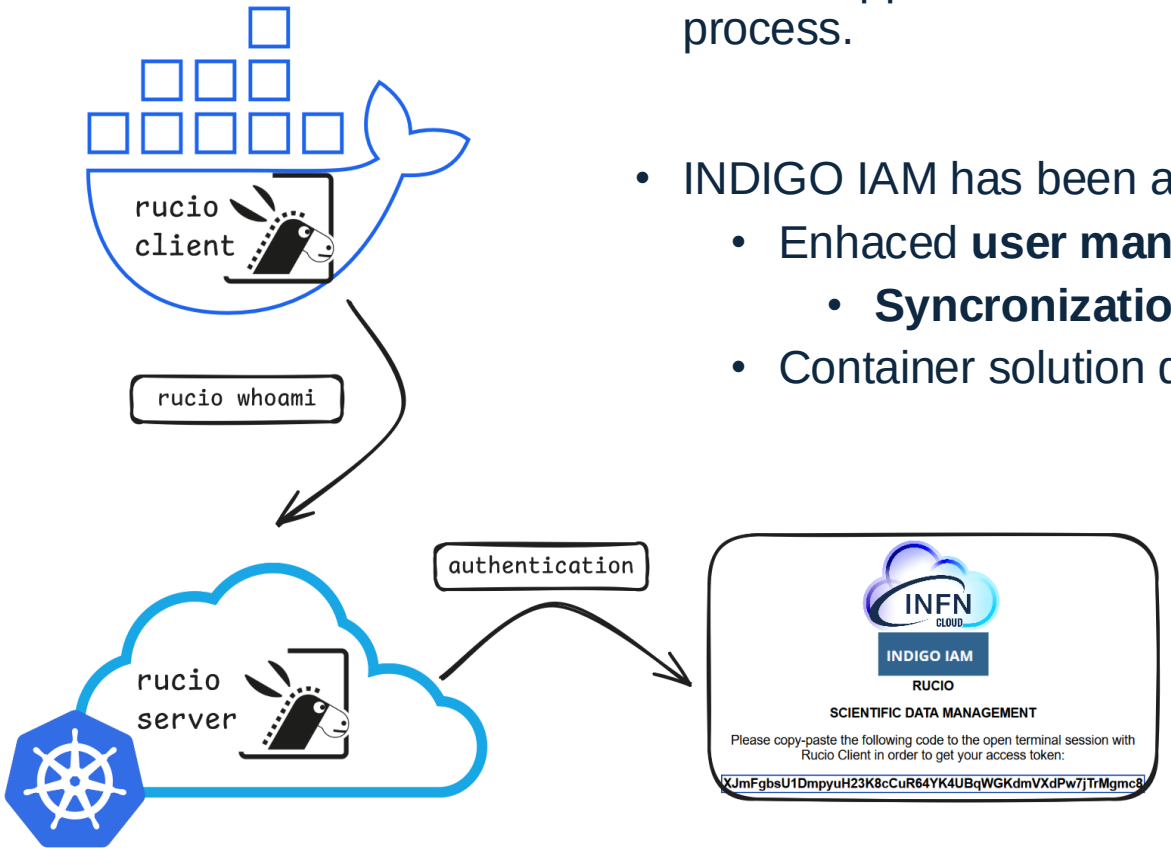
- Cluster kubernetes:
  - Rucio-server
  - Rucio-daemons
  - Indigo-IAM
  - MinIO, object storage

- FTS: open source software for reliable and large-scale data transfers

- Docker: rucio-client for user usage



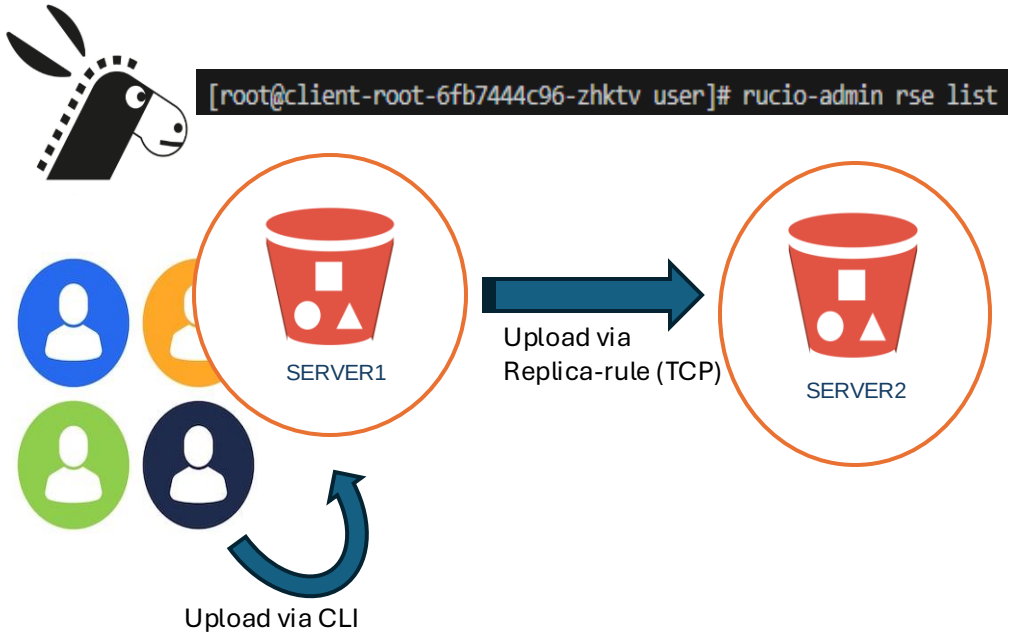
Open ID Connect AuthN/Z



- Rucio support **external identity provider** for the **authentication** process.

- INDIGO IAM has been adopted and configured with Rucio.
  - Enhaced **user management**.
    - **Syncronization** among IAM users and Rucio users.
  - Container solution developed for the the Rucio-Client service.

Use case



- Suppose that the **biomedical community** needs to **transfer** files from one center to another, to **share** information.
- To simulate this scenario, Rucio has been deployed with 2 different MinIO storage providers.
- For each storage provider, different **policies** can be set to share files.

Custom Policy

- Rucio assigns **permissions** to users.
- Permissions are **boolean flags** designating whether an account may perform a certain action (read, write, delete) on a resource (RSE, account, replica, etc.).
- Rucio comes with a generic permission policy including a typical set of **permissions**.

- Policy can be tuned properly to reflect the different community needs.
- e.g. only users belonging to a certain IAM **group** have read/write access.

Auditing

- Auditing is important to check the activity of users in order to **keep track** of their **actions** and to **detect anomalies**
- Deepening rucio logs, we understand it is possible to get information regarding:
  - Timestamp
  - Machine IP of rucio client
  - User access token and rucio user account
  - Tipe of actions + storage endpoint + bucket and DID

- We also find out some **critical issues**:
  - Rucio needs a VOMS-Proxy to delegate transfer to FTS:
    - Apparently they are working on the use of token.
  - FTS web monitoring page was exposing s3 credential:
    - We contact FTS teams and they have now introduced a new variable that hides GFAL credentials.

Summary

- Rucio is an open-source software framework that provides scientific collaborations with the functionality to **organize**, **manage**, and **access large amount of scientific data**.
- Rucio can share data across **distributed locations**, taking advantage of **different technologies** (object storage, block storage, tape, etc).
- The access is controlled by the **authentication** process provided by the external **IdP** and by the **authorization** provided by the costomizable **policies**.
- Rucio has been deployed
  - Locally for testing
  - Within the INFN-Cloud infrastructure
- As a next step there is the integration with other metadata catalog to enrich Rucio functionalities (e.g. Apache Atlas).

Thank you!