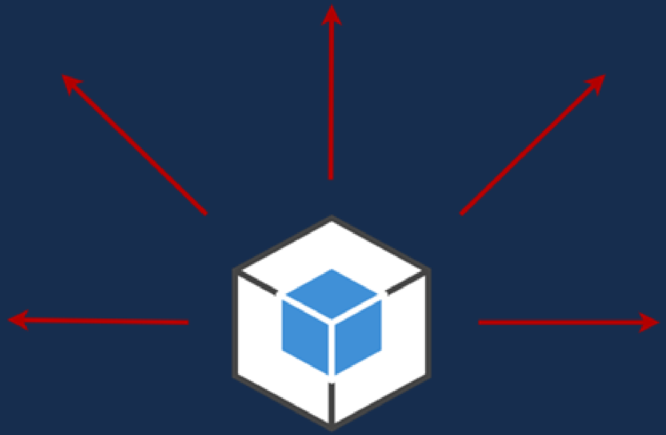
Software management: un servizio di distribuzione del software basato su CernVM-FS

Giada Malatesta giada.malatesta@cnaf.infn.it
Francesca Del Corso francesca.delcorso@pg.infn.it
Alessandro Costantini alessandro.costantini@cnaf.infn.it
Daniele Spiga Daniele.Spiga@pg.infn.it
Massimo Sgaravatto massimo.sgaravatto@pd.infn.it
Sergio Traldi sergio.traldi@pd.infn.it
Marco Verlato marco.verlato@pd.infn.it

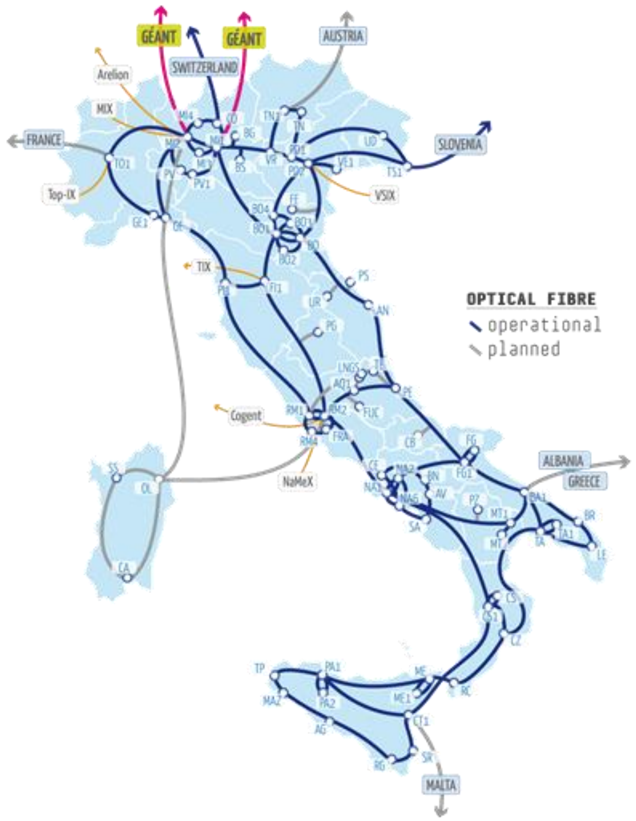
Workshop sul Calcolo dell'INFN – Palau (Sassari)
22/05/2024

Outline

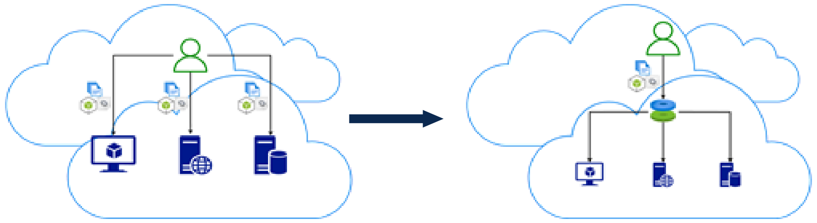


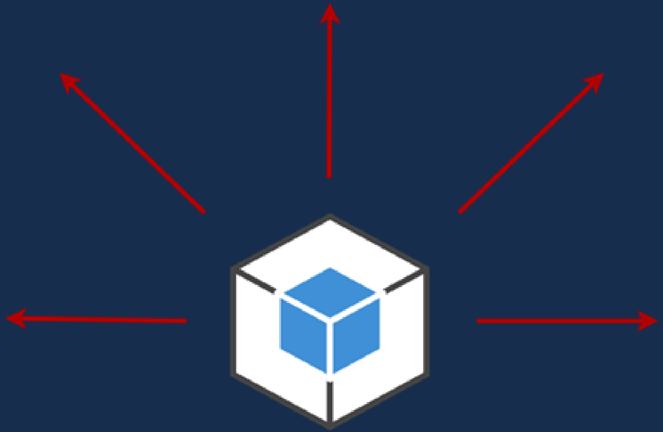
Software distribution challenge

User software distribution: the challenge



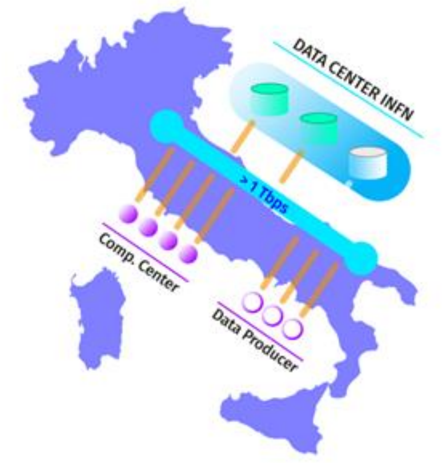
- In a **distributed and heterogeneous environment** the **sharing** of software, libraries, configurations and container images in an effective, user-friendly and transparent way can be **challenging**.
- There are already low-level solutions that address this challenge.
- Our aim is to further simplify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





Software Management solution

Software Management @DataCloud: the strategy

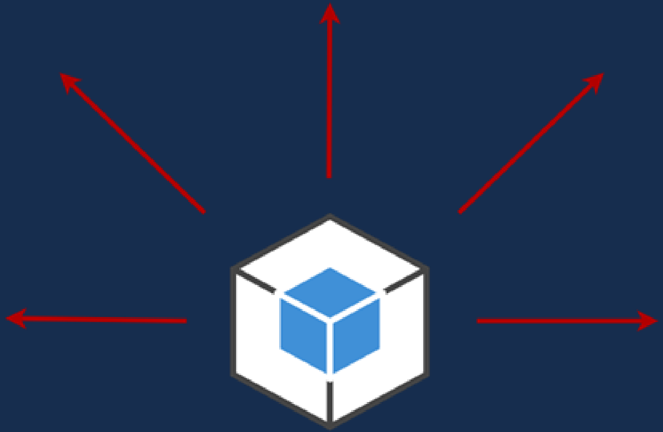


- In order to cope with the challenge in a **Cloud infrastructure**, such as our DataCloud, we implemented a **Software Management service**.

- 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.

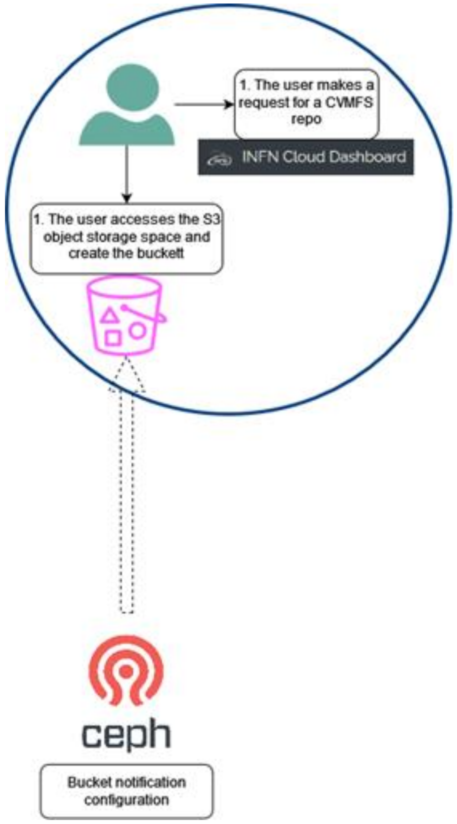
- **Automation**: in other words we enable the possibility to copy software, libraries and related dependencies, small files, configuration files etc in **S3 cloud storage** and that's it.

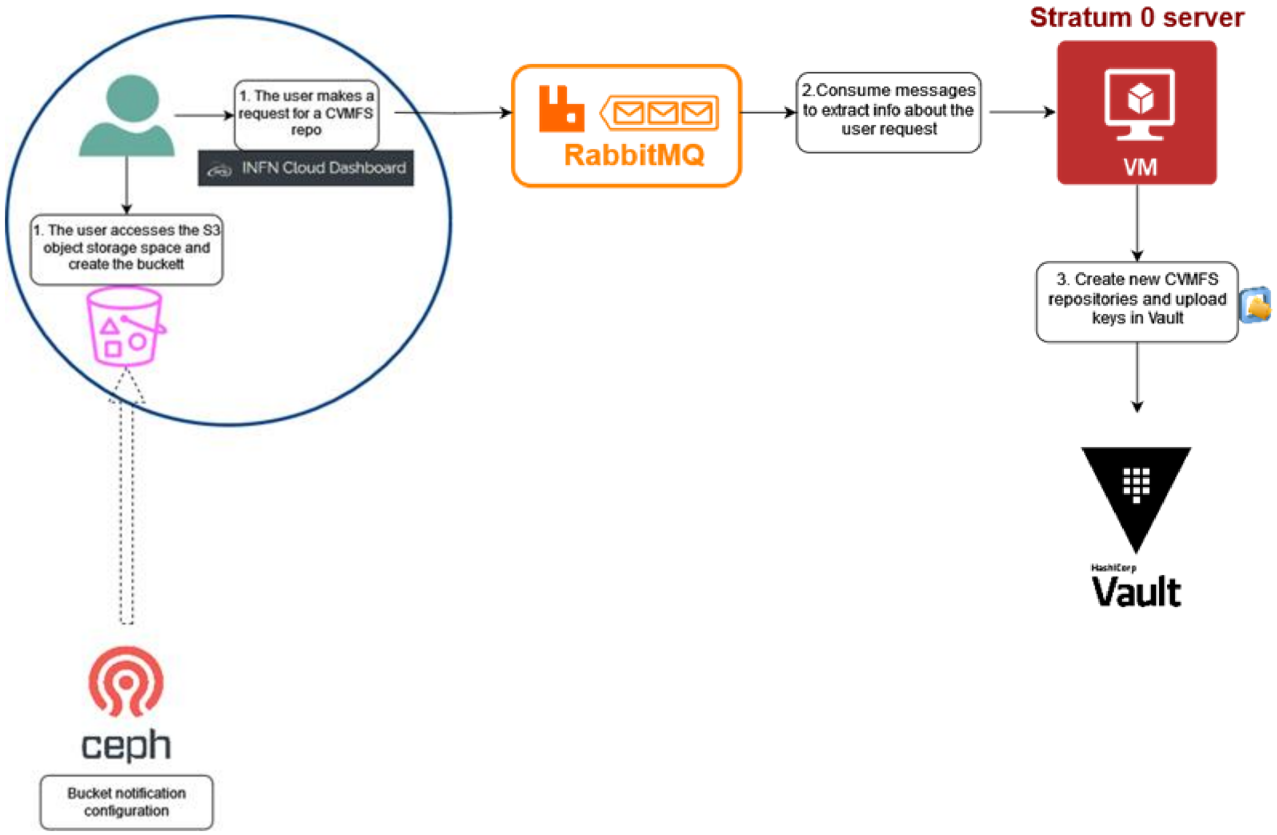


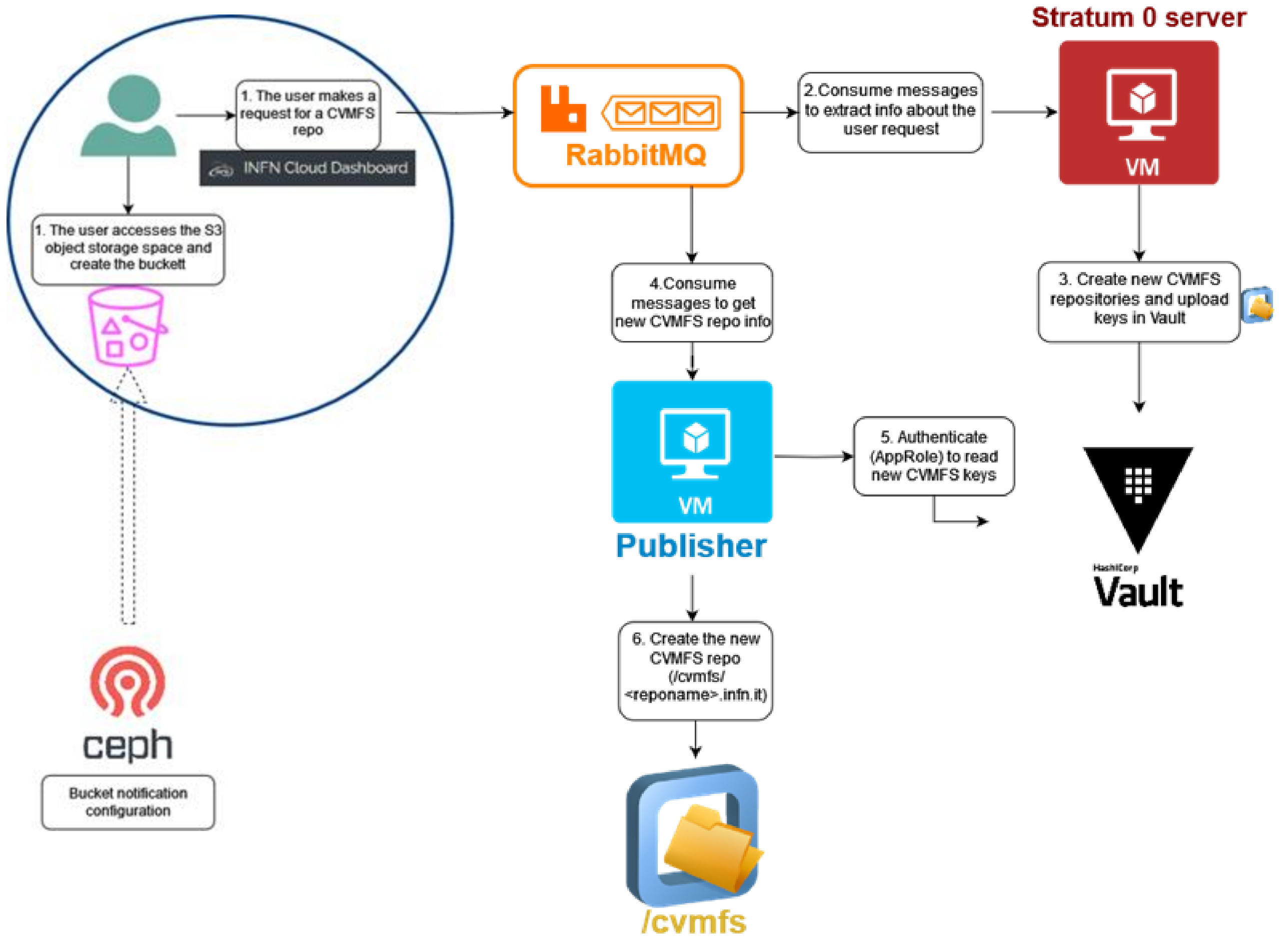
Workflow overview

Software distribution: workflow overview

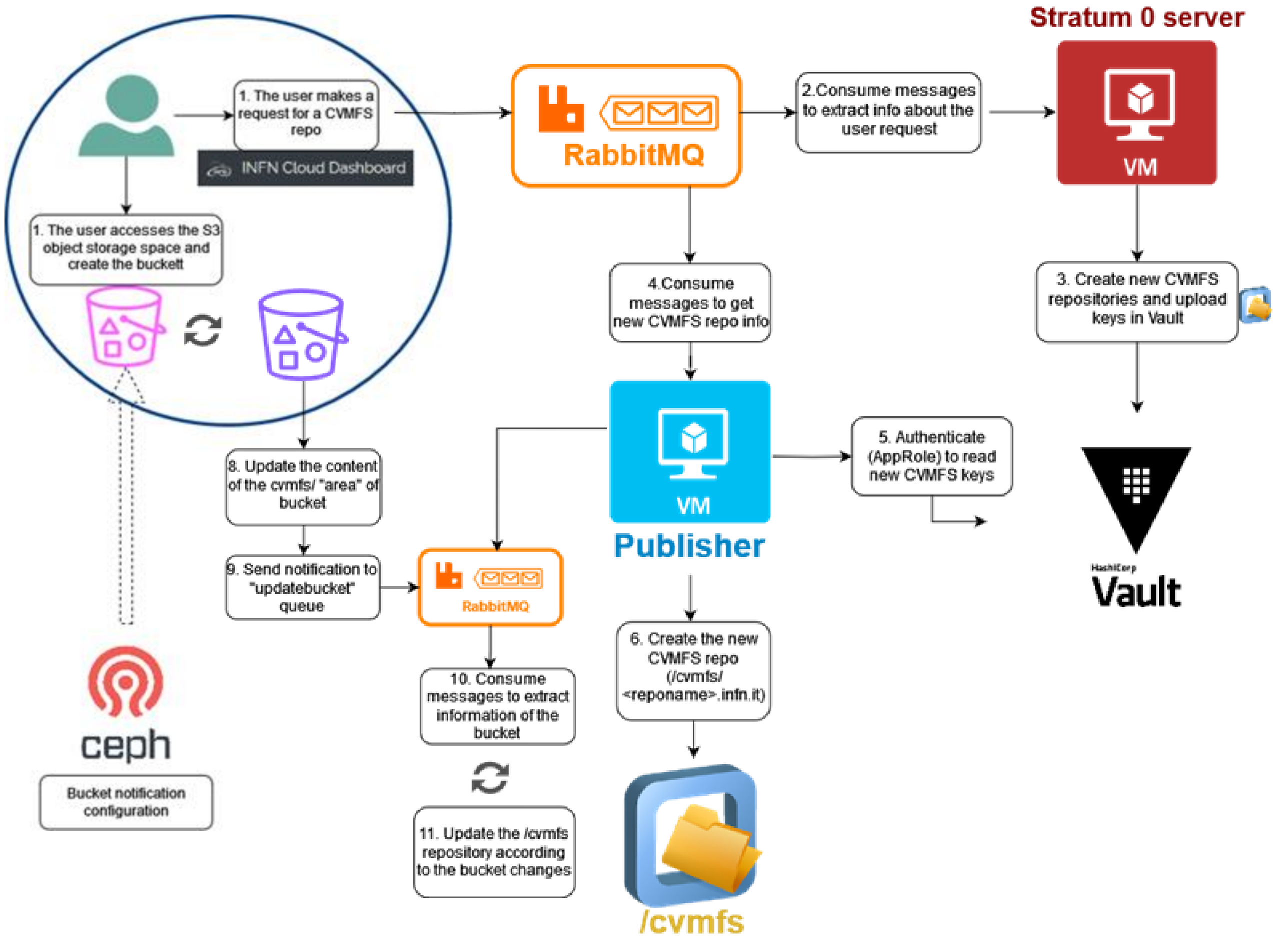
- The user **requests** a CVMFS repository (personal or group) via the **INFN Cloud dashboard**.
- The request is sent to **RabbitMQ** and is elaborated in order to create the repository.
- Once created, the relative **keys** are published in a **Vault system.**
- The user accesses the **S3 object storage** 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
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
ceph
Bucket notification configuration

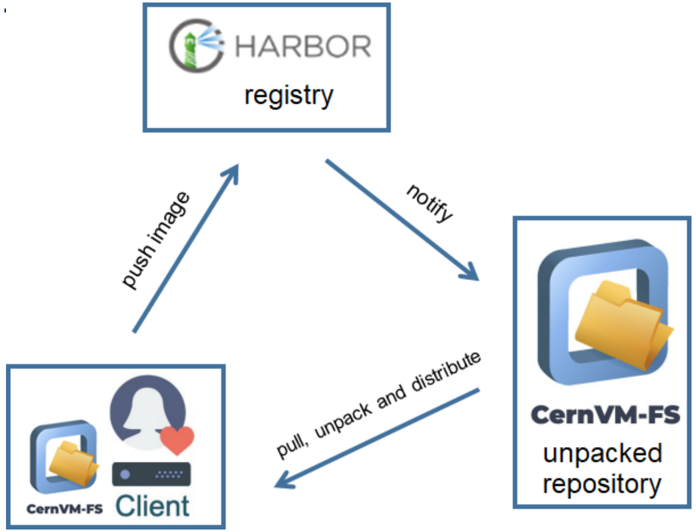


Stratum 0 server
1. The user makes a request for a CVMFS repo
INFN Cloud Dashboard
1. The user accesses the S3 object storage space and create the buckett
RabbitMQ
2.Consume messages to extract info about the user request
VM
3. Create new CVMFS repositories and upload keys in Vault
4.Consume messages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


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
VM
Publisher
5. Authenticate (AppRole) to read new CVMFS keys
8. Update the content of the cvmfs/ "area" of bucket
9. Send notification to "updatebucket" queue
RabbitMQ
HashiCorp
Vault
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

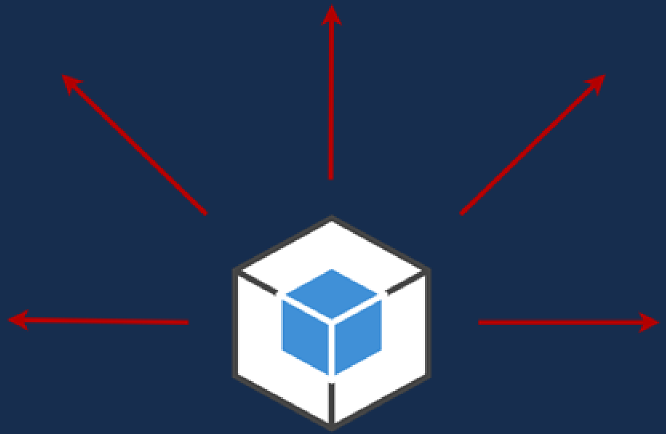
CVMFS to distribute container images: workflow overview

- CernVM-FS can be used to distribute **unpacked container images** via the **Harbor** registry.
- Minutes after the container images have been pushed to the Harbor registry, they are **unpacked** and **readily available** in CVMFS to be run with **apptainer**.
- No need to wait for minutes for pulling and decrompessing the complete image before run.
- An experimental solution is also in place for Running container in Kubernetes



- **User experience**

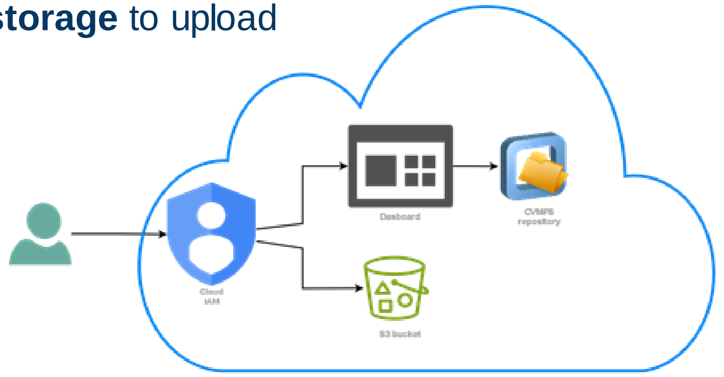
```
docker push harbor.cloud.infn.it/unpacked/my-image:1.0

apptainer exec '/cvmfs/unpacked.infn.it/harbor.cloud.infn.it/unpacked/my-image:1.0' /bin/bash
```



User perspectives

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.
- The user must have access to the **Harbor** registry to **push** the image.



Summary

Summary

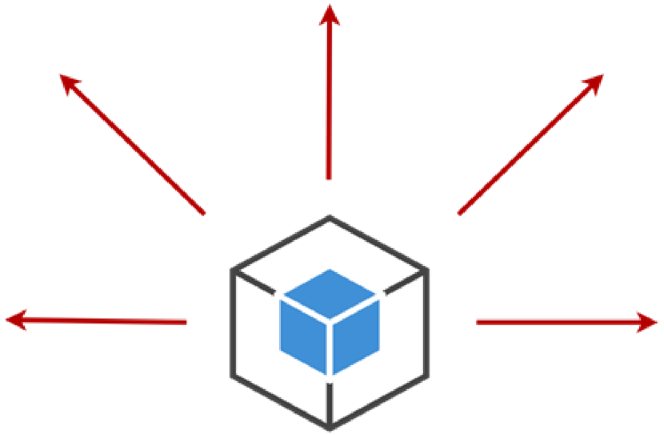
- Both **abstraction** and **automation** of the underlying CVMFS system are successfully provided by the presented **Software Management service**.

- **Abstraction**: users do not need to know the details of CVMFS, they just **upload** the **software** in their **bucket**.

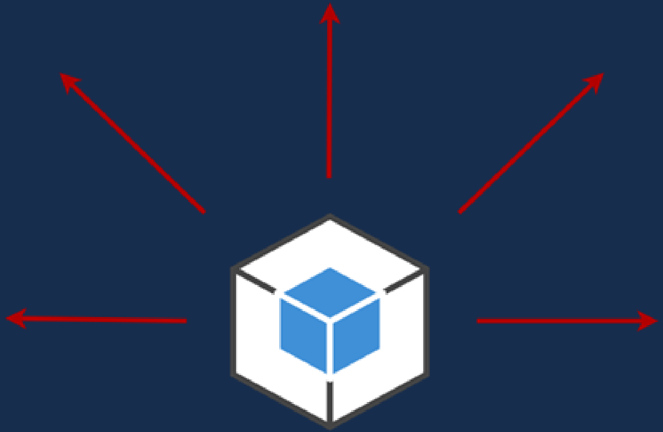
- **Standard** CVMFS: to expert users is left the possibility to distribute software through a CVMFS **publisher**.

- **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.



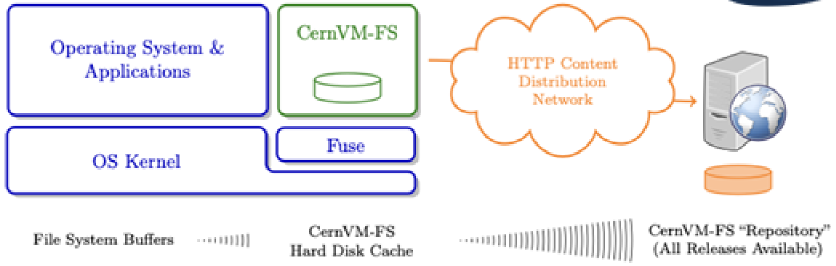
Thank You!

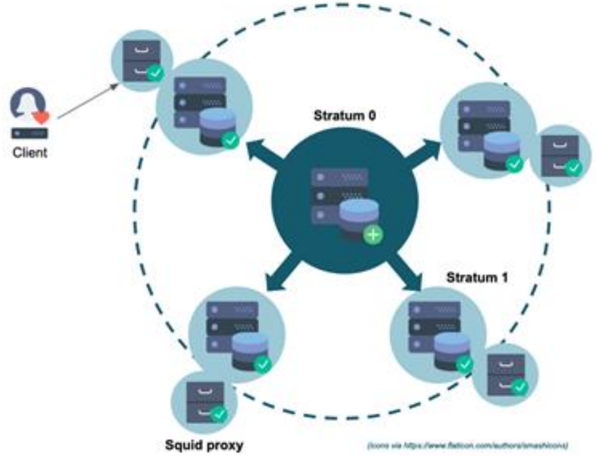


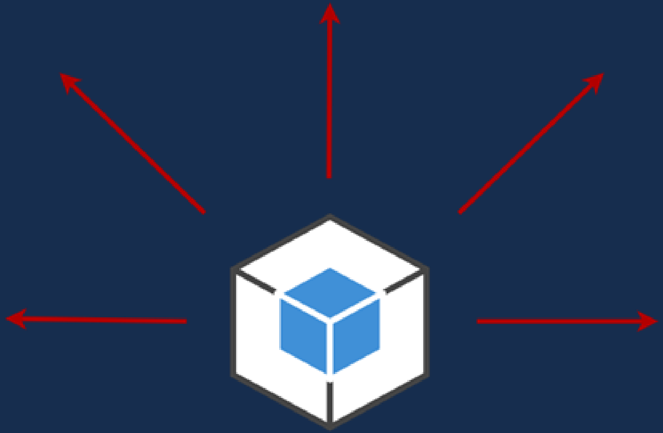
About CVMFS

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.


Operating System & Applications
CernVM-FS
OS Kernel
Fuse
HTTP Content Distribution Network
File System Buffers
CernVM-FS Hard Disk Cache
CernVM-FS "Repository" (All Releases Available)





Adopted technologies

Adopted technologies

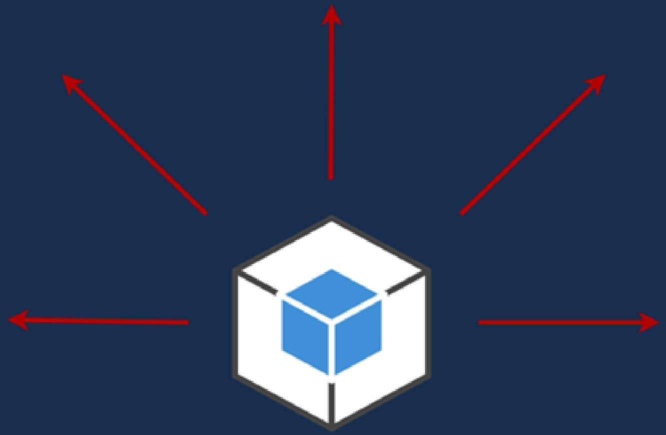
The <u>CernVM File System</u> provides a **scalable**, **reliable** and **low-maintenance software distribution service**. CernVM-FS is implemented as a POSIX read-only file system in user space (a FUSE module). Files and directories are hosted on standard web servers and mounted in the universal namespace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

<u>Ceph</u> is an **open-source**, **distributed storage system**.

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

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

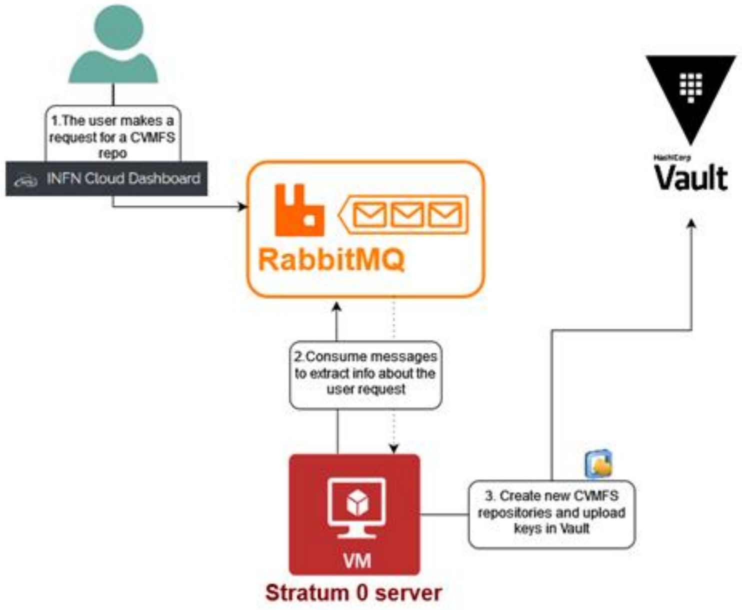
<u>Harbor</u> is an open source trusted cloud native registry project that stores, signs, and scans content.



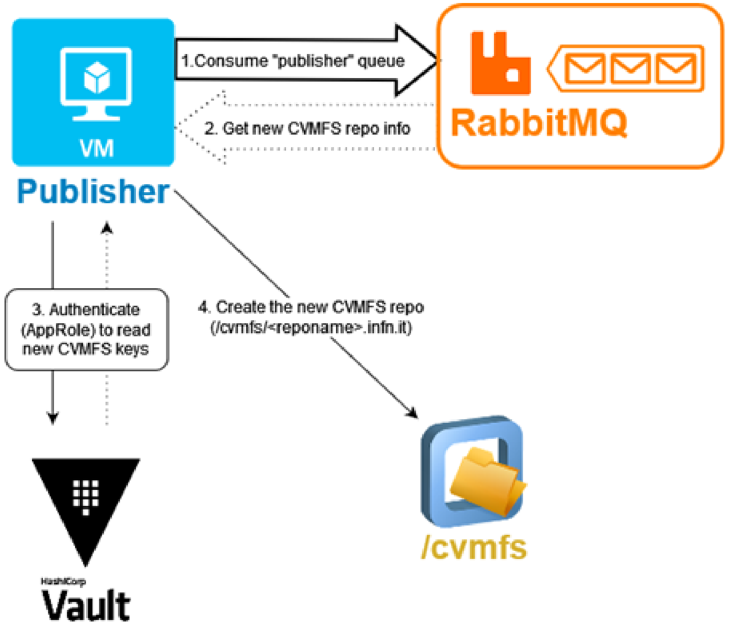
Implementation

CVMFS stratum 0 - Vault - RabbitMQ interaction

- This interaction allows the **CVMFS stratum 0** server to get **notified** when a user **requests** a personal/group **CVMFS repository**.
- It takes this information from a **RabbitMQ** queue.
- With this information, it **creates** the **CVMFS repository** and the relative **keys.**
- CVMFS stratum 0 server authenticates to **Vault** and copies the **secrets** in a specific path of the service.

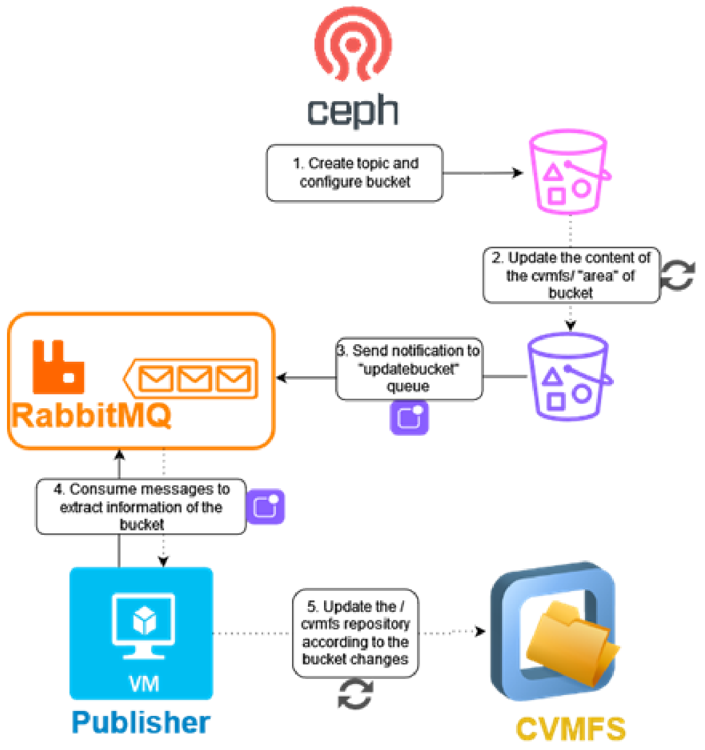

1.The user makes a request for a CVMFS repo
INFN Cloud Dashboard
RabbitMQ
HashiCorp Vault
2.Consume messages to extract info about the user request
VM
Stratum 0 server
3. Create new CVMFS repositories and upload keys in Vault

CVMFS publisher - Vault - RabbitMQ interaction


1.Consume "publisher" queue
2. Get new CVMFS repo info
RabbitMQ
VM
Publisher
3. Authenticate (AppRole) to read new CVMFS keys
4. Create the new CVMFS repo (/cvmfs/<reponame>.infn.it)
HashiCorp
Vault
/cvmfs

- This interaction allows the **Publisher** to understand if a user have requested a **new CVMFS** repositories.
- It takes this information from a **RabbitMQ queue**.
- The information is sent to the queue when the stratum 0 server creates the CVMFS repository and the keys.
- Using the given information, the publisher authenticates to **Vault** and takes the **keys** needed to write in the repository via gateway.

CVMFS publisher - Ceph - RabbitMQ interaction


ceph
1. Create topic and configure bucket
2. Update the content of the cvmfs/ "area" of bucket
3. Send notification to "updatebucket" queue
RabbitMQ
4. Consume messages to extract information of the bucket
VM
Publisher
5. Update the / cvmfs repository according to the bucket changes
CVMFS

- This interaction allows the Publisher to get information about **changes** in the cvmfs/ “area” of the **buckets** and therefore to **synchronize** the content of the CVMFS repository.

- **Notification messages** are sent to a RabbitMQ queue with informations about bucket owner, object key and event type.

- Using those informations, the publisher **distribute** the software in the correct CVMFS repository.