OMERO deployment on your platform

roadmap & technical details

Guillaume Gay
February 22

Contents

Introduction

The aim of this document is to outline a generic deployment strategy for microscopy bioimage data management at the various nodes of the France BioImaging infrastructure. It is by no means definitive, and will need a carefull adaptation to your local context.

It should be of interest to anyone involved in the bioimage data life cycle within the microscopy facility.

The source of this document can be found on FBI-data gitlab. The reader is encouraged to contribute corrections or remarks as issues there. The repository linked above will be privately forked for each node in a sub-group of the FBI.data group on the gitlab, and will be used to store site specific information (network mappings, roles indentifications, etc.).

The big picture

In short, our goal is to take the microscopy data out of the institues and store them in a shared facility. The agreed upon technical solution for that is to install an OMERO service in a data-center affiliated with the node.

You can find out more about omero on their website, whether you are a scientist or part of the support staff.

OMERO is composed of a server and a set of tools widely used to manage microscopy data. It is open source and managed by the openmicroscopy consortium.

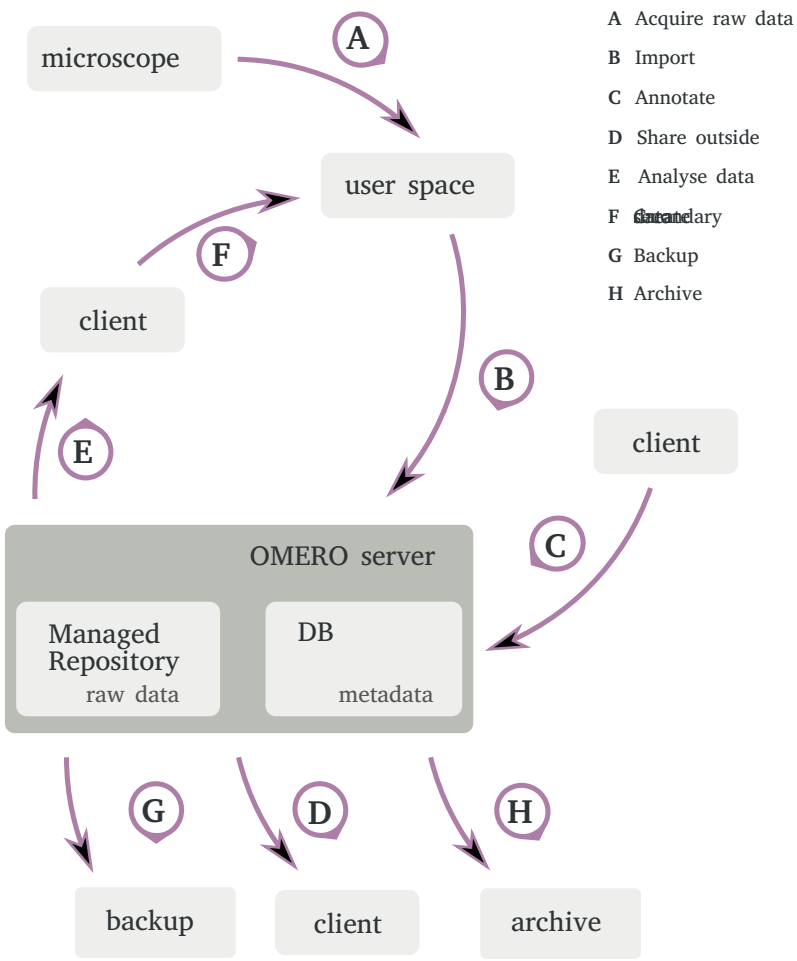
The main expected benefits of this deployment are:

- A more cost and energy effective storage.
- Better data security.
- Mutualised management, lower workload for current data managers and IT in the institutes.
- Better data availability and conservation.
- Easier data sharing with colaborators and data analysts.
- For published data, better complience with the FAIR principles.

Data life cycle

A detailed version of the data life cycle will be given in the platform's Data Management Plan (DMP), which will be constructed with the help of FBI.data in coordination with this project (see here details about that DMP).

Here is a rough sketch of the steps in the microscopy data life cycle:



Outline of the microscopy data cycle

The most complex aspect of this system is the import stage, because potentially big volumes of data have to be copied, displaced over a potentially heterogenous network, and imported into OMERO, which can be a resource intensive step.

Data big and small

Yet, a general observation from microscopy data managers is that most microscopy platform users do not generate large amount of data, as microscopy is not their central research tool, or their experminents do not need data intensive modalities, while a minority of users generate big amounts of data, often on specific instruments.

To account for that, we will distinguish between small and big data workflows. Big data can further be devided into two famillies:

- Many small files — for exemple in screening experiments, or with single molecule microscopy raw data. *this could correspond to 10 000 files or more, less than 10 Mo each*

- Few large files — as in long term 3D + t live imaging, or multiplexed whole slide images
  this could correspond to hundreds of more than 1 Gb files

Both usecases can pose different chalenges and have different bottlenecks.

Let's hope that we do not have to deal with the many large files usecase too much.

In the following, we first define a set of **roles** that will allow us to identify the various actors involved in the project. We'll then go in more details into the proposed (generic) deployment strategy, before discussing advanced import scenarios and methods, and eventually annotations.

Roles

In this first section, we define various "roles" implied during the deployment process and in production. One person can have mutliple roles, and each role can be fullfield by multiple persons.

We assume that the researchers are organized in **Groups** (although one researcher can belong to several groups). We distinguish 5 organisation levels: The FBI.data node, the local node, the datacenter, the platform and the institute or lab. Roles at each of those levels are detailed bellow.

Name	Description	shorthand
FBI.data		
FBIDataHead	Operational manager of the FBI.data Node	FH
FBIOmeroSpecialist	OMERO system expert	FOS
FBIDeploymentSpecialist	Deployment expert	FDS
FBIAppsSpecialist	Analysis worflows integration expert	FAS
—-		
FBI Node		
NodeHead	Local coordinator of the FBI Node	NH
NodeOmeroInstructor	OMERO instructor	NOI
—-		
Datacenter / mesocentre		
DataCenterComputingIT	Computing / virtual machine manager	DCC
DataCenterStorageIT	Storage and network management manager	DCS
—-		
Platform		
PlatformHead	The platform director	PH
PlatformManager	Operational manager of the platform	PM
PlatformOmeroAdmin	OMERO admin at the platform level	POA
PlatformIT	IT manager at the platform level	PIT
—-		
Lab / Institute		
LabSysadmin	The institute's IT department contact	LIT
LabHead	The institute's head	LH
—-		
Research Group / Team		
GroupLeader	The research group leader	GH
GroupOmeroAdmin	OMERO admin for the research group	GOA
GroupExperimenter	The data producer at the microscope	GE

Attribution should be clear to all these roles in your institution or platform, except maybe for the `GroupOmeroAdmin`. This person (who can be the group leader, but often will not) is the point of contact in a given group for the omero adminstration. He / she has restricted priviledges in the omero instance, and will be in charge of orchestrating data and user management at the group level, as detailed in the next section. It is up to the team to appoint this delegate.

This role list is focused on the deployment step, and some aspects of the data life cycle, as rich metadata annotation, data analysis, or data curation roles are not mentionned here.

Authentication and user management

In the short term, we will not provide a global or federated authentication strategy. Rather, authentication will be setup locally. Depending on the context, we might be able to use LDAP (possibly from various institutions) to identify users and groups. In parallel to that, manual user and group creation will be necessary (if only for outside collaborators). Later user management (changing groups, deactivating users) tasks also need to be performed.

This task will be destributed between the PlatformOmeroAdmin, who will be in charge of the **group** creation (as well as the GroupOmeroAdmin user creation), and the GroupOmeroAdmin, who will be in charge of the **user** management within a group. Note that those actions are easily performed on the webclient admin interface. In a small research team with three person, the head of the team will assume the omero admin role, but in bigger teams, it might be an engineer or technician, ideally someone with as much a stable position as possible.

Recommended settings for the priviledges of both roles are detailed bellow — this is configured at the user creation stage (see details on those priviledges here).

PlatformOmeroAdmin priviledges

☐ Sudo
☒ Write Data
☒ Delete Data
☒ Chgrp
☒ Chown
☒ Create and Edit Groups
☒ Create and Edit Users
☒ Add Users to Groups
☒ Upload Scripts

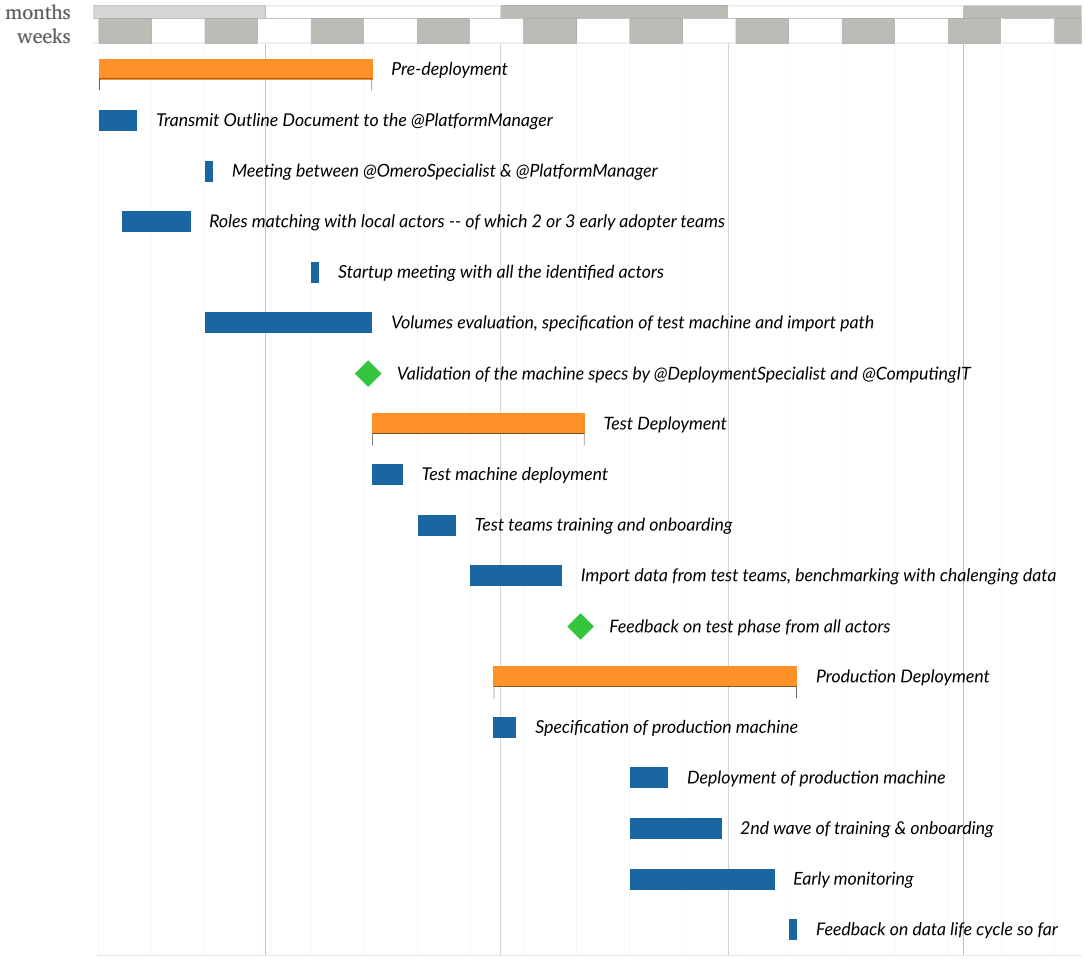
GroupOmeroAdmin priviledges

☐ Sudo
☐ Write Data
☐ Delete Data
☒ Chgrp
☒ Chown
☐ Create and Edit Groups
☒ Create and Edit Users
☒ Add Users to Groups
☐ Upload Scripts

OMERO Deployment

Here is an indicative Gantt chart of the deployment steps, for an overall duration of about 4

months:



Gantt chart of the deployment process

It is comprised of 3 phases that are detailed bellow:

Pre-deployment

During this phase, we study the local context and match the roles outlined above to actual persons, among which two or three test teams who will take part in the next stage of the deployment.

Once this match making is complete, we setup a meeting with all the technical actors and the test teams to launch the test deployment. During this phase, we evaluate the data volumes and the data flow rate for the node, in order to specify the infrastructure.

Task	Involved roles
Transmit Outline Document to the PlatformManager	FOS
Meeting between @OmeroSpecialist & @PlatformManager	FOS, PM
Roles matching with local actors – of which 2 or 3 early adopter teams	PM
Startup meeting with all the identified actors	all local actors, FOS, FDS
Volumes evaluation, specification of test machine and network architecture	FDS, DCC, DCS, PM, LIT
Validation of the system spectification and design	FDS, FOS, DCC, DCS

Test deployment

This omero instance mimics the production deployment and is used as a first step to study standard practice and edge cases. At this step, it is important to identify potentially complex datasets. We setup and test import workflows and perform benchmark on the challenging datasets. Local actors are trained in their role as omero administrators.

During this phase are also sorted out the various aspects of data access permissions and authentication.

At that stage, an access to the deployment infrastructure must be provided to the FBIDeploymentSpecialist (FDS).

The training of the test team will be done by both the NodeOmeroInstructor and the FBIOmeroSpecialist to also train the trainer.

Task	Involved roles
Test machine deployment	FDS, DCC, DCS
Test teams training and onboarding	NOI, FOS
Import data from test teams, benchmarking with chalenging data	GE, GOA, PM, DCC
Feedback on test phase from all actors	all local actors, FOS

Production deployment

Once the lessons from the test deployment have been gathered, and after a feedback session with the test teams and representatives of the whole user community, the production server is deployed and new teams are enrolled in the project. Regular onboarding sessions are then organised to progressively grow the userbase.

Automated monitoring as well as “in person” oversight of the system is initially performed by

the OmeroSpecialist and the PlaftormOmeroAdmin.

We propose a monthly meeting between the FBI.data team and the local team during the first year of the deployment to make sure any issue is caught and solved rapidly.

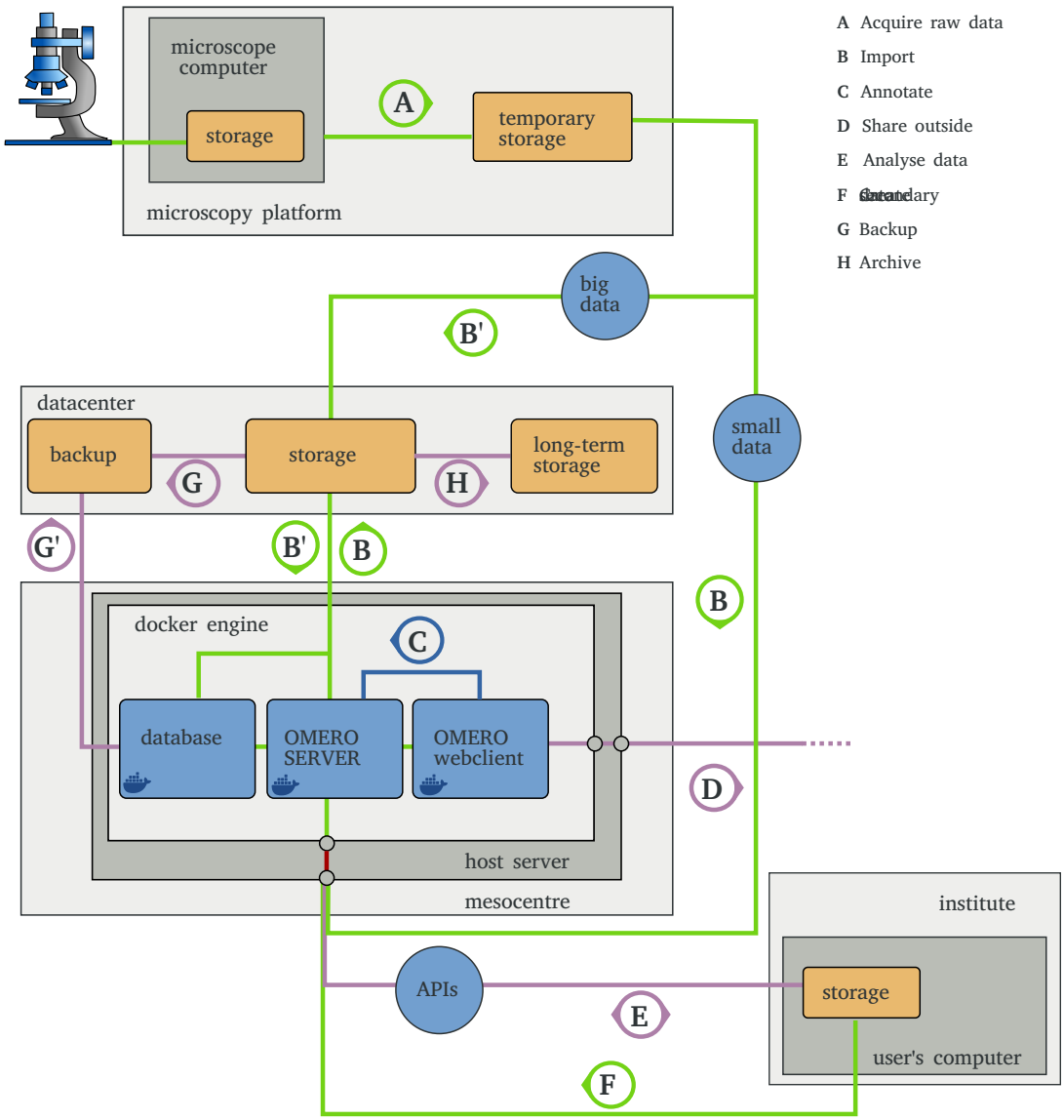
Task	Involved roles
Specification of production machine	FDS
Deployment of production machine	FDS, DCC, DCS
2nd wave of training & onboarding	NOI
Early monitoring	PM, FOS, FDS
Feedback on data life cycle so far	all local actors, FOS

Legacy data

In the nodes where OMERO databases are already in use, an extra step is to transfer the legacy system to its new home. This step will be very dependant on the local context and will surely be work intensive for the FBIOmeroSpecialist and the PlatformOmeroAdmin. Furthermore, we might want to perform this step on the test server, which might delay significatively the deployment of the production server.

Technical overview

In the section we give implementation details for the deployment, administration and maintenance of the service. We discuss data import and dataflow in more details in the next section.



Data cycle with the underlying computing infrastructure

The above sketch depicts 4 local networks: the platform, the datacenter, the mesocenter and the instute, although e.g. datacenter and mesocenter might be merged in some cases.

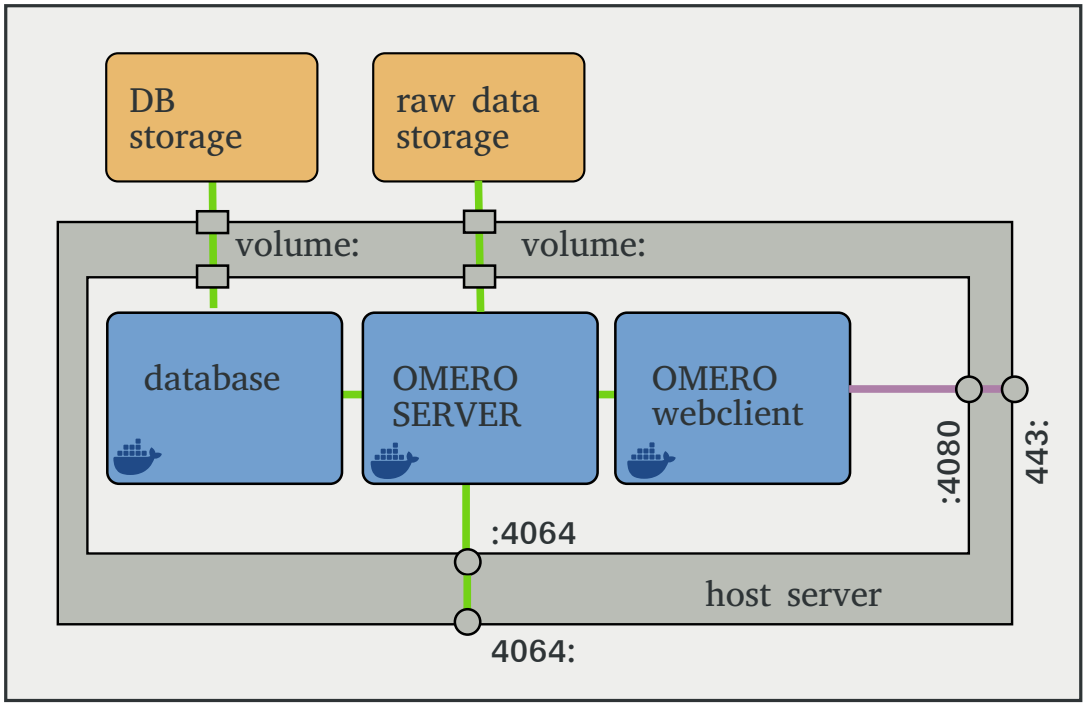
The mesocenter hosts the OMERO host server, and omero itself is deployed as 3 containers.

Docker

We decided to use docker as a distribution platform. As the container layer allows to be independent of the host system, it is easy to propose a standard solution to all the nodes.

We will use a fork of the docker example repository distributed by openmicroscopy. It uses `docker-compose` to create three containers for the omero server itself, the webclient, and the prostgresql database. For increased security podman and `podman-compose` can be used to host trully root-less containers.

At the local level, the main task will be to specify correct environment variables for the domain names, port numbers and possibly authentication methods.



Detailed view of the container and storage architecture

The storage for both the database and the raw data are mounted through a `volume` statement in the docker-compose file (see details here). While the precise mount configuration will likely depend on the datacenter storage vendor, it is important to notice that the (postgresql) DB storage needs to be very I/O efficient, so SSD / Nvme, but will not require high volumes - bellow the Tb level. As data on the raw data storage is more heterogenous, maybe tiered storage if feasible would be interesting. Deduplication strategies where reported to lower the volume of the occupied data.

Hosting

These docker containers will be hosted by the mesocenter, preferably on a virtual machine to ease later upscalling or migration of the service. The mesocenter or the node will also need to provide a fully qualified domain name to the webclient instances (both test and production).

We will use traefik or caddy as reverse proxies for the containers to provide encrypted (https) access to the services.

Benchmarking and monitoring

As discussed, we will setup benchmarks at deployment time to identify possible bottelnecks. We will setup grafana and prometheus to monitor the system use, and help detect issues with the system early.

Software updates

One of the advantages of docker is the ease with which software updates can be performed. Once the parent image is updated, `docker-compose pull` and `docker-compose up` are enough to update the services (with no restart).

We will deploy updates in 3 steps:

1. Update the FBI docker images from upstream, deploy the update on the FBI test and development instances.

2. Deploy the updated images on the local test deployment

3. Deploy on the production services.

Test cases will be setup at each of those steps to ensure service integrity.

Should database updates be performed, we will test them in the same order.

Backup and recovery

Raw data backup is assumed to be a service provided by the datacenter (with onsite and offsite redundancy). On our side, we will provide standardised cron jobs to perform SQL dumps of the OMERO database, which is enough to restore service. Standard daily, weekly and monthly rolling backups will be performed.

A crash and restore test will be regularly performed to ensure data security.

Archiving

The question of long term archiving of the data depends on the local storage infrastructure. We will help setup a strategy to sort short, mid and long term storage data in a transparent way to the user, either through tiered storage if it is available in the infrastructure, or through the configuration of an irods instance between the storage and the omero instance.

Import workflow

Traditional workflow for small data volumes

The traditional method to import data into OMERO consists in using either the desktop client OMERO.Insight or the equivalent ImageJ / Fiji plugin. This is the advised import method for small data volumes. This import can be performed directly from the microscope computer, or at a latter step from a buffer storage within the institute or platform networks. Once imported, the raw data can be deleted by the experimenter.

This method has the disadvantage of requiering a few actions by the users, and thus might not be very favorable to a high adoption rate. Strategies to overcome these limitations will be

discussed after the deployment, along with discussions on annotation and curation methods.

Automated worflow for big data

There are several draw-backs with the legacy import mehod in a big data use case:

- If the client is installed on the microscope, import must be performed during acquisition or after. If import is long, it will block the access to the acquisition machine.

- If the client is installed on the user's computer, this requires the user to perform the import process in a second step. It can be tedious for big or complex datasets. Furthermore, the raw data is then available locally to the user at this stage, lowering the insentive to use OMERO.

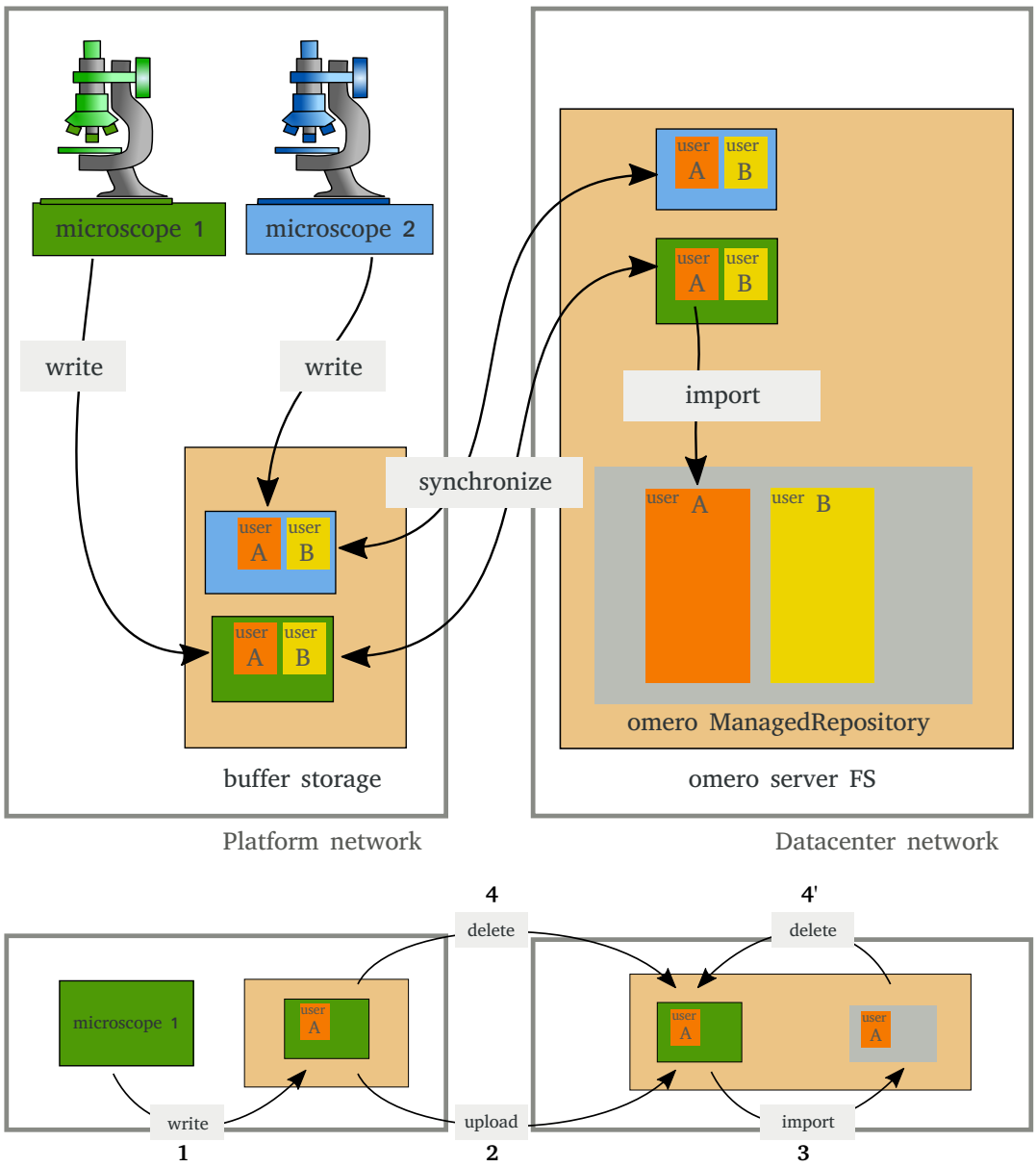
- With that import method, a hard copy of the data is performed at import time. This can be ressource intensive both on the network and on the omero server itself.

More recently, automated import tools have been made available by openmicroscopy.

You can see here a clear presentation of the various import methods available to us. A more detailed discussion of those scenarii is available here.

Our objective is to avoid unnecessary data copies, and prevent the import process and I/O to block the server or the microscope. Here we propose an import workflow with an onsite buffer drive where data is temporary stored before it is imported.

Import steps



Graphical view of the import workflow

Let's detail the 4 steps outlined above:

1. Data is writen to the buffer storage.
A shared directory from the buffer storage is mounted on each microscope (as a Windows network shared drive). Detailed access rights and structure of this directory will depend on the platform policy and network architecture. What is needed here is that the user has write access to a directory in that drive. Ideally, the user should have only *write once* access to that drive.

We propose that this directory has a data lifetime limit (e.g. 2 months) to ensure its volume stays limited.

2. Data upload
This is the most I/O intensive step of the process, as acquired data needs to be transported to the datacenter. Here, infrastructure adaptations might be necessary to provide sufficient bandwidth for efficient transfer.

Pending discussion with all the IT staff involved, we propose that `rsync` is used to synchronise the data between the buffer storage and the datacenter. Fine grained configuration can be performed to optimise that stage. The import daemon needs read / write access to the buffer drive.

3. Import
Once the data is copied to the OMERO server file system, we can use the logs of the upload stage (e.g. rsync logs) to trigger import into omero.

We propose to use the CLI inplace import with hard-linking `import -ln`. This has the benefit of not copying the raw data.

We propose to develop an omero script for the omero web client to allow the user to trigger both data synchronization and data import by choosing the directory to be imported. This action will send an import task, added to a queue in an import manager daemon.

4. Delete
Once the data is imported, it can be deleted from the synched directory (the hard link ensures the data is still accessible to OMERO). Deletion can be performed first on the buffer storage — either after some delay as discussed above, or manually. The next synchronisation will propagate the deletion to the omero server synched directory, while keeping a reference to the raw data in the ManagedRepository.

Alternatively, data deletion can be performed automatically at the end of the import procedure (with `import -ln_rm`).

Once the pointer to the data in the synched directory on the server is deleted, the synchronisation process propagates the deletion to the buffer storage.

We propose to favor the first method, as the synchronization process then only needs read access to the buffer storage, thus keeping data security responsabilities well separated.

> *At the time of this writing, this process is still at the project stage. Although we hope to have proof of concept soon, the rest of the deployment is not dependent on its availability. Should the need arise, custom import script can be deployed for special cases. It is still important to engage the discussion on network and drive architecture at the interface between the datacenter and the platforms with this project in mind.*

Reusing and sharing data

Accessing your data

Webclient

OMERO provides a choice of clients. The first interface to the data will be the webclient. In our setup, it is deployed as a docker container within the docker-compose. It should be accessible form anywhere through https (port 443), with a fully qualified domain name (e.g. https://omero.example.uni). Note that it is possible to spawn multiple webclients for the same omero server, as long as a single one has write access to the DB. Read only webclients can be userfull for public data sharing or display.

By default, several tools will be installed with the webclient, such as `OMERO.figure` that allows to create figures directly from the browser, and is known to be apreciated by the scientists. Such tools will be curated by the FBI.data team in collaboration with the platforms.

The intensive use of webclient scripts to perform analysis tasks is discouraged, as it draws resources from the webserver container. Help with setting up computing workflows will be available from the FBIAppsSpecialist.

ImageJ, napari plugins and APIs

As allready discussed, a ImageJ plugin allows to interact with OMERO. It is also possible to use APIs in multiple languages (Python, Matlab, etc.). For this, OMERO uses the IceSSL transport layer. This secure tool works through a custom port (4064 by default) that needs to be accessible to the users. There must be a discussion with the datacenter administrators to open that port for external access on the OMERO server instance.

Sharing data for analysis

In the framework of the collaboration with a data analyst external to the institute, the easiest way to share data is to manually create an OMERO account for him and add this new user to the group. This will be performed by the GroupOmeroAdmin. See here for documentation on the various data sharing methods.

Publishing data

The solution generally advised to share plublic data is to deploy a second omero-webclient container with readonly access to the database, and then follow the procedure detailed here.

— February 22 —