FBI DATA : une gestion FAIR et libre des données de bio-imagerie

Théo Barnouin, Emmanuel Faure, Perrine Gilloteaux, Jean-François Guillaume, Marc Mongy, Guillaume Maucort, Raphaël Braud-Mussi, Guillaume Gay

FBI.data : une gestion FAIR et libre des données de bio-imagerie

Théo Barnouin
CNRS / France BioImaging
Plateforme MicroPICell
IRS UN - 8 Quai Moncousu
44000 Nantes

Jean-François Guillaume
CNRS / France BioImaging
Plateforme MicroPICell
IRS UN - 8 Quai Moncousu
44000 Nantes

Perrine Paul-Gilloteaux
CNRS / France BioImaging
Plateforme MicroPICell
IRS UN - 8 Quai Moncousu
44000 Nantes

Marc Mongy
INSERM / France BioImaging
Plateforme MicroPICell
IRS UN - 8 Quai Moncousu
44000 Nantes

Guillaume Gay
LIRMM / France BioImaging
LIRMM - 161 Rue Ada
34095 Montpellier

Emmanuel Faure
LIRMM / France BioImaging
LIRMM - 161 Rue Ada
34095 Montpellier

Guillaume Maucort
Bordeaux Imaging Center / France BioImaging
Centre Broca Nouvelle Aquitaine
146 rue Léo Saignat
33076 Bordeaux

Raphaël Braud-Mussi
CNRS / France BioImaging
Institut Jacques Monod
15 Rue Hélène Brion
75013 Paris

Résumé

France-BioImaging (FBI) est une Infrastructure Nationale en Biologie et Santé du Programme National d'Investissements d'Avenir dans le domaine de l'imagerie biologique. FBI est à la croisée des domaines entre la biologie moléculaire et cellulaire, la biophysique et l'ingénierie, les mathématiques et l'informatique. Cette infrastructure coordonnée unique regroupe plusieurs grandes plateformes d'imagerie biologique et des laboratoires spécialisés dans la R&D pour l'imagerie dans 10 nœuds géographiques et un nœud transversal pour l'analyse et la gestion des données images. Elle assure une mission de développement et de mise à disposition des dernières technologies et expertises en imagerie biologique, ainsi que de dissémination et de formation. Elle est par ailleurs le Nœud français de l'ERIC EuroBioimaging et est impliquée dans des réseaux et programmes européens et internationaux.

France-BioImaging a notamment développé un axe fort de son programme autour de la gestion des données avec le projet FBI.data auquel sont affectés plusieurs ingénieurs. FBI.data a pour but de fournir les outils pour la gestion des données d'imagerie biologique, d'acculturer et de faciliter l'ouverture des données en suivant et en participant aux standards internationaux de son domaine d'expertise.

Mots-clefs

Systèmes et réseaux, FAIR, Open Data, Mésocentres

1 Contexte

L'infrastructure informatique France-BioImaging a été conçue pour répondre a différents besoins :

- accéder en visualisation à des images de formats et dimensions très variables ;
- accéder aux ressources en stockage et calcul des mésocentres régionaux ;
- libérer les machines d'acquisition des données au plus vite ;
- valoriser les données par des annotations adéquates dès le démarrage des projets.

Cette infrastructure doit également faire face à des contraintes fortes. Elle doit permettre une facilité d'utilisation pour des usagers non informaticiens, un maintien de plusieurs instances de gestion distribuée nationalement, être disponible aux utilisateurs au niveau national et international, respecter des standards internationaux du domaine dans la description des données et s'accommoder de la prédominance des machines Windows parmi les utilisateurs et les machines d'acquisition.

2 Présentation de la solution

Le scénario d'utilisation retenu est donc le déploiement de serveurs tampons au plus proche des machines. L'utilisateur des stations de microscopie dépose ses données après l'acquisition dans un répertoire sur le serveur tampon après s'être authentifié via un fédérateur d'identité. Ces données sont annotées en respectant un standard propre à l'imagerie biologique et poussées automatiquement dans le mésocentre, et également enregistrées dans une base de données d'images offrant des fonctionnalités de visualisation et traitement à distance. L'utilisateur peut gérer de manière autonome le partage de ces données. Lors de la publication de son projet, les données peuvent être transmises avec un minimum d'effort sur un entrepôt public européen respectant les principes FAIR.

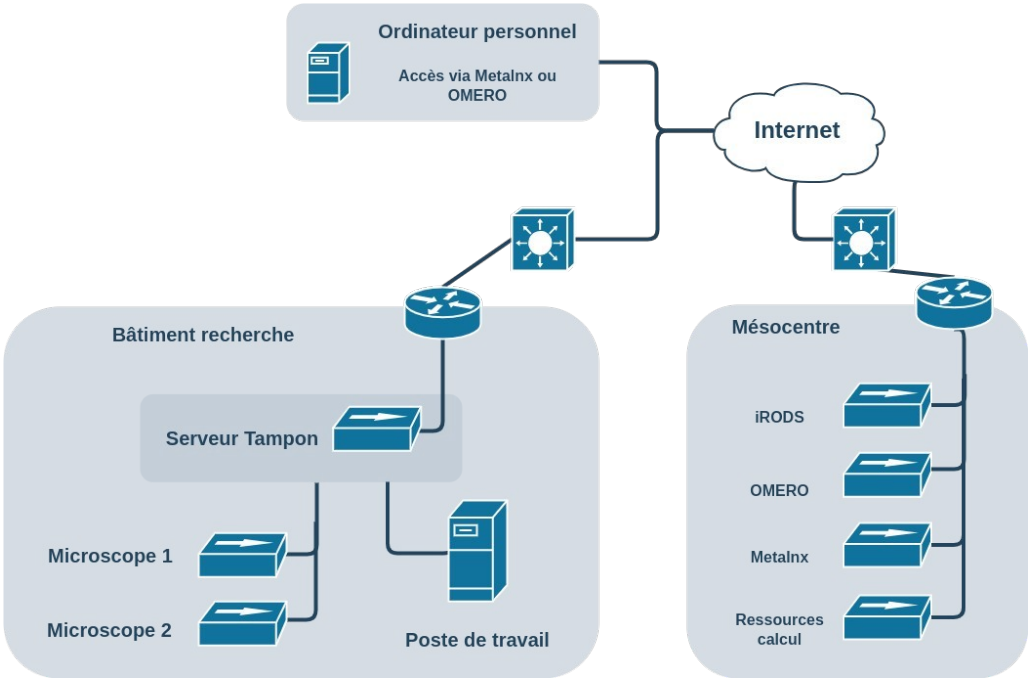


Schéma simplifié de la vue utilisateur

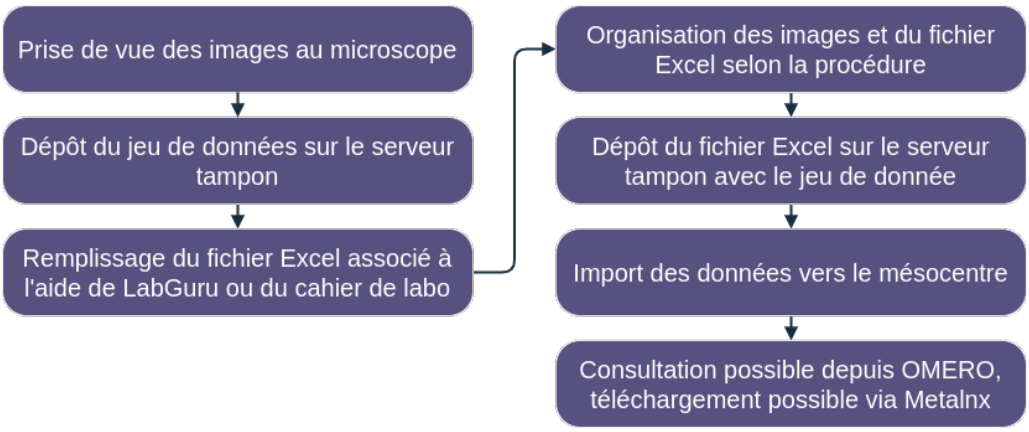


Schéma présentant le chemin effectué par une donnée de microscope

3 Présentation de l'infrastructure

Le projet est divisé en deux parties distinctes : une partie hébergée dans un mésocentre hébergeant les données et les services web, et une partie plateforme de recherche constituée d'un serveur "tampon" sur lesquels les données de microscopies sont acquises. Cela permet aux chercheurs de pouvoir travailler rapidement sur leurs données localement avant de les rendre accessibles plus largement. La puissance de calcul des mésocentres permettra aussi dans un deuxième temps de fournir des ressources pour l'analyse de ces données.

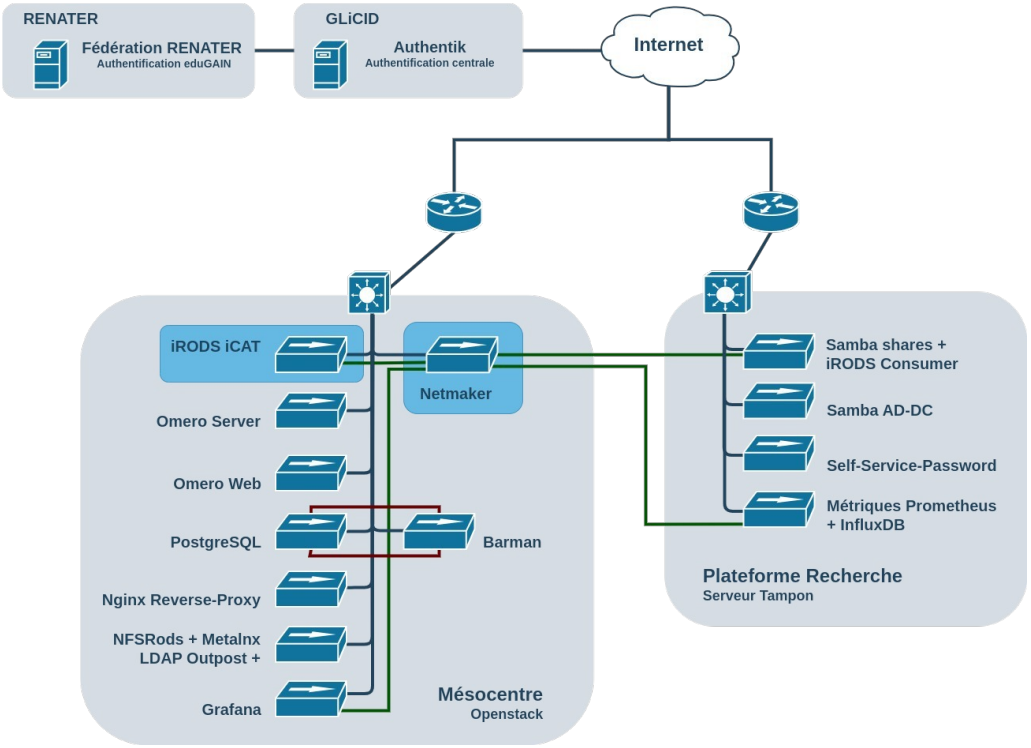


Schéma de l'architecture réseau

3.1 Aux mésocentres

Pour déployer les services en mésocentre, nous nous appuyons sur les infrastructures existantes proposées par les mésocentres nationaux avec lesquels nous travaillons. Il s'agit par exemple de GliCID (Groupement Ligérien pour le Calcul Intensif Distribué à Nantes) ou du CROcc (Cloud Recherche Occitanie) pour Montpellier.

Ces mésocentres proposent une infrastructure cloud basée sur OpenStack, sur laquelle nous déployons des machines virtuelles (VM) pré-configurées par cloud-init. Ces VM sont quasi intégralement basées sur Debian 12, à l'exception des services OMERO Server et OMERO Web, basées sur Ubuntu 22.04 dans un souci de compatibilité.

Les services déployés sont les suivants, divisés en deux catégories pour plus de lisibilité :

Le **backend**, l'hébergement des données et gestion du réseau avec :

- **iRODS** pour l'hébergement des données de microscopie, propose un stockage distribué entre les plateformes de recherche et le mésocentre pour simplifier le transit des données ;
- **NFSRods** qui permet de monter des données **iRODS** via **NFS ;**

- **OMERO Server** où les données **iRODS** sont montées en **NFS ;**
- Une base de données **Postgresql** pour **iRODS** et **OMERO**, couplée à **Barman** pour nous permettre du Point-In-Time Recovery grâce au WAL ;
- Récupération des métriques et logs via **Rsyslog** + **Prometheus** + **Loki** + **InfluxDB ;**
- **Netmaker**, permettant de créer un réseau privé virtuel entre les mésocentres et les plateformes de recherche ;
- **Authentik** qui permet une authentification fédérée via **OIDC** ou **OpenLDAP** selon la compatibilité des logiciels.

Le **frontend**, la porte d'entrée pour les utilisateurs, accessible sur le web comprenant :

- Portail Authentik comprenant un lien vers les différents services proposés ;
- **OMERO Web**, l'interface Web de visualisation de données présentes sur OMERO Server ;
- **Metalnx**, une interface web permettant de télécharger et uploader des fichiers vers iRODS ;
- Documentation utilisateur et admin avec **Mkdocs ;**
- Visualisation de logs et métriques avec **Grafana ;**
- Gestion de tickets avec **Zammad.**

Les deux composants au cœur de cette infrastructure sont iRODS et OMERO.

3.1.1 iRODS

iRODS est un logiciel de gestion de données ayant de nombreuses fonctionnalités : stockage distribué, système de règles pour automatiser le transit de données, gestion poussée des métadonnées, etc.

iRODS fonctionne grâce à un serveur central, situé au mésocentre dans notre cas, nommé **catalogue** ou **iCat**. Ce serveur est relié à la base de données Postgresql et centralise les comptes utilisateurs et les métadonnées. À ce catalogue viennent se greffer des instances "clientes" d'iRODS, des **consumers**, qui correspondent dans notre cas à nos plateformes de recherche. Ces instances permettent également d'héberger des données mais ne sont pas directement reliées à la base Postgresql et dépendent donc du catalogue pour fonctionner.

Puisque iRODS propose un **système de fichiers distribués**, il importe peu à l'utilisateur de savoir où se situe physiquement la donnée : en interrogeant n'importe quel serveur iRODS (le catalogue ou un consumer), le logiciel se charge de faire transiter la donnée jusqu'à l'utilisateur de manière transparente. Grace à Netmaker, les différentes plateformes de recherche seront connectées entre elles via un réseau privé virtual (VPN) et pourront donc dialoguer sur un réseau dédié sans pour autant être accessible via une IP publique.

Une autre force d'iRODS est son système de règles. On peut écrire un ensemble de règles, dans un langage spécifique à iRODS ou en python selon le moteur utilisé, que iRODS fera respecter. Dans notre cas, nous avons une règle faisant transiter les données depuis la plateforme de recherche vers le mésocentre quand celle-ci est importée dans OMERO et qu'elle est stockée sur le tampon depuis un temps donné en paramètre. Ces moteurs de règles sont très puissants et permettent une gestion très précise et granulaire des données, d'automatiser la gestion des métadonnées, de l'indexation, etc.

Puisqu'il faut passer par la ligne de commande pour utiliser iRODS, il a été crucial de proposer un outil graphique pour que les utilisateurs les moins avancés en informatique puissent au moins visualiser les données stockées dans iRODS. Cet outil doit aussi pouvoir permettre aux utilisateurs de partager les données qu'ils souhaitent avec d'autres personnes.

L'outil retenu dans un premier temps était **SFTPGo**, plus particulièrement un fork développé et maintenu par **Cyverse** permettant l'interfaçage avec iRODS. Cet outil très complet permet un accès aux données via HTTP, WebDav ou encore SFTP. Cependant, SFTPGo n'est pas conçu spécialement pour iRODS et manque donc de certaines fonctionnalités qui nous sont essentielles. Nous avons d'abord essayé d'ajouter ces fonctionnalités en contribuant au code de Cyverse mais, les résultats n'étant pas convaincants, nous avons changé de cap pour finalement déployer **Metanlx**, un outil bien moins complet ne proposant qu'un accès aux données via HTTP mais spécialement conçu pour iRODS et développé par **DELL EMC**. Nous avons donc considérablement facilité le déploiement en changeant d'outil, n'ayant plus à maintenir et configurer plusieurs services (SFTP, WebDav…) ni à développer de fonctionnalités supplémentaires.

3.1.2 OMERO

OMERO est un logiciel principalement dédié à la publication d'images au sein d'une équipe (permettant un travail collaboratif) ou publiquement. Il facilite également la publication d'articles scientifiques notamment grâce à la traçabilité et l'accessibilité des images, mais aussi grâce à la présence d'un module permettant de créer facilement et rapidement des figures pour publications scientifiques. Il permet enfin d'analyser des images grâce à un système de scripts, exécutables par l'interface web. Il est divisé en deux parties, un serveur et une interface web, installés séparément. OMERO est la dernière étape du système : c'est ici que les utilisateurs vont pouvoir visionner, partager et analyser leurs données sur une interface Web pensée pour l'analyse de données.

La partie serveur est accessible pour les utilisateurs via **websocket** sur une adresse dédiée et accessible publiquement. Cela leur permet de se connecter sans passer par l'interface web et via des clients lourds tel que **OMERO Insight**. Ces clients permettent la visualisation, l'analyse, le téléchargement et le téléversement d'images. Pour les développeurs et administrateurs systèmes, il est possible de s'interfacer avec OMERO directement, en ligne de commande ou via une API python. Cette API nous permet notamment d'importer des images depuis iRODS via **OMERO Quay**, un logiciel que nous développons en interne et dont le fonctionnement est détaillé plus bas dans cet article. Le développement de ce logiciel a été nécessaire puisque OMERO ne peut pas directement s'interfacer avec iRODS.

L'interface web est accessible via un navigateur web pour les utilisateurs et fournit plusieurs outils de visualisation, recherche, d'annotation et de partages de données. Un certain nombre de plugins sont disponibles et permettent d'augmenter les capacités d'OMERO, qui propose un framework pour développer son propre plugin en python. C'est via ce framework que nous avons développé un plugin pour créer une interface web à notre outil d'import de données Quay. Cet outil est ainsi disponible directement depuis l'interface web d'OMERO, ce qui simplifie l'accès pour l'utilisateur.

3.2 Sur les plateformes de recherche produisant les données

Nous préconisons de déployer les services sur des machines virtuelles, dans un hyperviseur . Cela permet une plus grande facilité de mises à jour des services et de maintenance générale. Toutefois, il nous paraît important que les différentes unités de recherche puissent utiliser le matériel existant. Les services installés sur les plateformes ne nécessitent pas beaucoup de puissance de calcul ni même forcément un grand espace disque. Cela permet à des plateformes de toutes tailles d'utiliser l'architecture proposée.

3.2.1 SAMBA et Self-Service-Password

Le cœur de l'infrastructure sur les plateformes de recherche est un disque réseau pouvant être monté sur un poste Windows, Linux ou MacOS. Côté serveur il s'agit d'un service SAMBA, configuré en mode Active Directory. Les utilisateurs sont gérés localement puisque SAMBA ne peut pas s'interfacer avec Authentik. Nous avons développé pour cette raison un script python permettant de récupérer les utilisateurs auprès de l'API Authentik, par provenance, pour les créer en local dans SAMBA.

Les données de microscopie sont ainsi stockées et accessibles en local dans un premier temps, avant d'être intégrées dans le mésocentre. Cela permet aux chercheurs de faire un tri et d'éventuelles analyses localement avant de rendre ces données accessibles sur le cloud, dans le mésocentre.

Self-Service-Password est un petit service web, écrit en PHP et permettant la gestion des mots de passe SAMBA. Il permet notamment aux utilisateurs de changer leur mot de passe mais aussi de le réinitialiser via un lien envoyé par mail. Ce service nous a très rapidement été indispensable et a grandement facilité la gestion des mots de passe puisqu'il autonomise l'utilisateur, qui n'a plus besoin de contacter l'administrateur système en cas d'oubli de mot de passe.

3.2.2 OpenZFS

Pour gérer le stockage des différents serveurs installés sur les plateformes, nous avons fait le choix d'utiliser OpenZFS. Il s'agit d'un système de fichiers et d'un ensemble d'outils permettant de créer des RAID logiciels, des volumes virtuels, de gérer des quotas, de créer des partages réseau, etc. C'est un système de fichiers très robuste et relativement standard, ce qui facilite son adoption par les différentes plateformes.

L'avantage majeur de ZFS est qu'il utilise de la RAM pour créer un cache qui améliore grandement les performances, notamment en lecture. Cela permet aux chercheurs d'accéder à leurs données de manière fluide localement.

L'utilisation de RAID logiciel offre une meilleure visibilité aux administrateurs des plateformes sur l'état des disques à distance et de faciliter d'éventuels dépannages. Il est possible d'ajouter un disque de rechange directement dans ZFS (un **hot spare**) et ainsi de pouvoir changer un disque défectueux par ce disque de rechange, à distance simplement en se connectant au serveur en SSH.

ZFS propose également un certain nombre de métriques, exportables via prometheus. Cela permet de consulter ces métriques dans un tableau de bord (dashboard) Grafana et donc de rendre lisible l'état des différents serveurs sur les plateformes pour les administrateurs qui ne seraient pas sur site.

3.3 Automatisation du déploiement avec Ansible et OpenTofu

L'architecture proposée est complexe : une vingtaine de rôles Ansible et 720 lignes de playbook pour déployer 13 VM au mésocentre et 3 sur la plateforme de recherche. Les outils utilisés doivent donc être assez standards et le code produit le plus lisible possible pour que le projet soit utilisable et compréhensible par le plus grand nombre.

Nous avons fait le choix d'utiliser OpenTofu (le fork libre de terraform) avec le provider OpenStack pour le provisionnement des VM. Il a principalement été choisi pour sa facilité d'utilisation : OpenTofu accède à l'API OpenStack via un token, et le provider OpenStack permet d'écrire une configuration pour les VM que l'on souhaite déployer.

Il s'agit ici de la deuxième force d'OpenTofu, sa configuration déclarative : plutôt que de décrire les étapes à suivre pour déployer une VM ("crée une VM à partir de tel template, ajoute-la dans ce groupe, ajoute-lui tant de RAM, etc."), on décrit l'état final que l'on souhaite déployer ("je veux 13 VM dont tant sont dans ce groupe, ayant tant de RAM, etc."). Cela permet aux administrateurs

d’avoir une grande lisibilité sur l’infrastructure puisque tout est inscrit dans les fichiers de configuration.

Cela permet enfin à OpenTofu de pouvoir comparer l’état des VM déjà installées à l’état souhaité. S’il manque des VM, on les ajoute et si certaines sont déjà présentes on ne change rien.

Pour la configuration et l’installation de logiciels sur ces VM, nous utilisons Ansible. C’est un choix assez classique puisqu’Ansible est aujourd’hui un logiciel assez standard et très largement utilisé par les administrateurs systèmes et réseaux. Le code produit sera donc facilement réutilisable et lisible par des tiers. De plus, Ansible étant un logiciel très utilisé, de nombreux **playbooks** et **rôles** sont accessibles sur Github ce qui permet de réutiliser du code existant pour installer certains composants.

Ansible est aussi léger et simple d’installation. Il utilise SSH pour se connecter aux VM et les configurer : il n’y a donc pas de composants supplémentaires à installer sur les VM.

Ces deux outils nous permettent de faciliter l’intégration continue avec Gitlab CI. L’IN2P3 (Institut national de physique nucléaire et de physique des particules, CNRS Nucléaire et Particules) nous met à disposition une forge Gitlab, ainsi que des runners qui permettent de déployer et tester le code sur des conteneurs dédiés. Nous automatisons aujourd’hui le déploiement de VM dans un environnement de test à chaque commit sur notre forge Gitlab. Nous pouvons ainsi détecter les erreurs au déploiement et pourrons par la suite automatiser une série de tests sur l’infrastructure déployée.

4 Présentation de l’environnement de développement

4.1 Développement spécifique : OMERO-Quay

Dans son ensemble, le système implique des actions asynchrones et distribuées sur les données : transport, droits d’accès, importation dans OMERO. Nous avons développé l’utilitaire OMERO-Quay en Python pour effectuer l’ensemble de ces tâches dans le cadre d’une API cohérente. Dans notre solution, les données sont finalement accessibles à travers deux interfaces : iRODS et OMERO. Pour iRODS, il s’agit d’une vue « fichier » sous forme de hiérarchie de Collections et Data Objects. La profondeur de l’arborescence est arbitraire. Sous OMERO, la hiérarchie est limitée à trois niveaux : Group / Project / Dataset. Les fichiers images sont hébergés au niveau Dataset. Pour conserver la cohérence de ces deux systèmes, nous définissons une hiérarchie commune basée sur le modèle **Investigation** / **Study** / **Assay** (**ISA**) qui organise les données dans un arbre à 3 niveaux. Cette structure est formalisée dans un schéma défini dans le langage LinkML. Ce schéma modélise les jeux de données, les personnes associées à une investigation et leurs rôles, les métadonnées (paires de clef/valeurs, étiquettes) et l’état d’avancement du workflow d’import. La racine du modèle est appelée Manifest. Depuis le modèle **LinkML** (un fichier au format YAML), on produit automatiquement des classes de données pydantic. À chaque importation par un·e utilisateur·ice, une instance de la classe **Manifest** est crée et munie d’un identifiant unique, puis serialisée en **JSON**. Ce message JSON est ensuite échangé entre les différents systèmes au cours des étapes de transformation des données. La première étape de création du Manifest est pour le moment effectuée par l’utilisateur·ice à partir d’un fichier excel structuré à partir d’un patron, décrivant les éléments essentiels de la hiérarchie et validé au moment de l’importation. Ce fichier excel est téléversé par l’utilisateur sur une interface web simple attachée au client web d’OMERO. Au mésocentre, un service OMERO-Quay expose un server tornado qui reçoit une requête POST contenant le JSON. Ensuite, OMERO-Quay s’appuie sur la bibliothèque **zeroMQ** pour échanger le manifeste, mit à jour à chaque étape (réorganisation des données dans iRODS, import dans OMERO). Pour s’assurer de la traçabilité des données entre les outils, on s’appuie sur les systèmes

de métadonnées de iRODS et OMERO pour garder une référence aux indices de ces objets dans les deux bases de données. À chaque étape, le manifeste JSON est mis à jour dans une base de documents MongoDB. On peut interroger cette base via une requête GET sur le serveur tornado, ce qui permet d'effectuer le suivi de l'import depuis l'interface web.

Enfin, la gestion automatique des utilisateurs·ices entre SAMBA et Authentik évoquée plus haut est intégrée à OMERO-Quay.

4.2 Intégration continue avec Gitlab CI

Pour assurer la robustesse du développement d'Omero-Quay, nous avons mis en place un environnement de développement *sandbox* intégré au code source. Dans cet environnement, nous utilisons **Docker compose** pour déployer cinq conteneurs :

- un conteneur postgresql pour les bases iRODS et OMERO ;
- un conteneur pour le serveur iRODS ;
- un conteneur pour le serveur NFSRODS qui expose les données comme point de montage NFS ;
- un conteneur pour le serveur OMERO, qui joue aussi le rôle de serveur Omero-Quay ;
- un conteneur jouant le rôle de client Ces conteneurs sont orchestrés par Docker compose.

Un *job* (déclenché manuellement pour économiser temps et ressources) permet de re-construire les images Docker de ces composants. Dans une deuxième étape, des tests unitaires et d'intégration sont réalisés. Pour cela, une archive zenodo contenant un jeu de données de test est téléchargée.

5 Quelques problèmes rencontrés et leurs solutions

5.1 Authentification fédérée et appartenance des données

Une des principales contraintes du projet est l'authentification des utilisateurs aux différents services. Le projet étant amené à être utilisé par un grand nombre d'utilisateurs de différentes provenances, il est crucial de centraliser ces utilisateurs et de les identifier de la même manière sur les différents logiciels qu'ils vont être amenés à utiliser. De plus, puisque nous hébergeons des données de recherche, il est important de pouvoir retracer et identifier à qui appartiennent ces données, dans le respect des principes FAIR.

Un effort de centralisation au niveau national a déjà été effectué par le réseau RENATER qui propose aujourd'hui la **Fédération Éducation-Recherche**, un service d'authentification centralisé et sécurisé. Ce service s'inscrit plus largement dans la fédération internationale **eduGAIN**, bien connue et utilisés par la majorité les universités au niveau international. Nous nous appuyons donc sur ce service d'authentification pour permettre d'accéder à nos services, ce qui veut dire en pratique que n'importe quel utilisateur ayant un compte dans un organisme rattaché à eduGAIN peut accéder à nos services.

Pour autant il n'est pas possible de connecter directement nos multiples services directement à la Fédération RENATER ou à eduGain. Les services utilisant OIDC et l'authentification unique (comme c'est le cas pour Zammad ou Grafana par exemple) pourraient se connecter directement la Fédération, mais RENATER ne délivre des tokens de connexion que sous certaines conditions (à raison). Aussi, la plupart de nos services ne peuvent pas utiliser OIDC directement et doivent passer par un annuaire LDAP pour centraliser leurs utilisateurs. C'est le cas d'OMERO ou iRODS par exemple.

Pour pallier ce manque, il nous a donc fallu installer un fournisseur d'identité (Identity Provider, IdP abrégé en anglais). Si le choix a d'abord été fait d'utiliser Keycloak, nous avons finalement changé d'avis pour un logiciel plus récent, au développement actif et très prometteur : Authentik. Il s'agit d'un logiciel proposant de l'authentification unique (Single Sign On, ou SSO) proposant plusieurs fournisseurs d'identité comme OIDC (Oauth2) ou SAML, mais aussi de backend plus "traditionnel" comme LDAP ou Radius.

Nous avons fait le choix d'installer une instance unique d'Authentik accessible publiquement et à laquelle vont se connecter toutes les instances de FBI.data installées en mésocentre. Pour les plateformes de recherche, Authentik propose le déploiement d'outpost LDAP, c'est-à-dire d'un conteneur Docker proposant un simple service LDAP récupérant les utilisateurs depuis l'instance Authentik publique via un token d'API. Ces conteneurs sont déployés dans chaque mésocentre et les différents services s'y connectent sur un réseau local. Cela permet de ne pas avoir à exposer un annuaire LDAP sur une IP publique.

Authentik proposant une API très bien documentée, cela nous a également permis de développer une fonctionnalité supplémentaire à Omero-Quay pour créer automatiquement les utilisateurs présents dans l'annuaire LDAP sur les serveurs SAMBA des plateformes. Authentik ne propose pas à ce jour de fournisseur d'identité compatible avec SAMBA.

5.2 Faire de l'Active Directory sans Windows

Pour rendre notre système le plus simple d'accès et le plus interopérable possible, nous avons décidé que la porte d'entrée pour un utilisateur serait un partage réseau, accessible sur un réseau local et depuis un ordinateur Windows, MacOS ou Linux. Avec ces contraintes, le choix le plus logique est le logiciel SAMBA, une implémentation du protocole SMB initialement développé par Microsoft et disponible sous Linux.

Concrètement, le serveur installé sur les plateformes sera un serveur Samba Active Directory Domain Controller (AD-DC), un contrôleur Active Directory gérant à la fois les utilisateurs et le partage de fichiers. Pour faciliter l'installation du service Samba nous avons d'abord pensé utiliser une distribution Linux dédiée au déploiement d'un AD-DC comme **Zentyal** par exemple. Malheureusement l'intégration de iRODS et Omero-Quay étant difficile sur ces distributions, nous avons décidé de développer des playbooks Ansible pour déployer Samba.

Ce serveur Samba doit être indépendant des services installés au mésocentre pour que les plateformes de recherche puissent continuer de fonctionner même en cas de coupure d'internet. Il est donc possible pour les utilisateurs de continuer l'acquisition et l'analyse d'image localement même en cas de coupure des services au mésocentre.

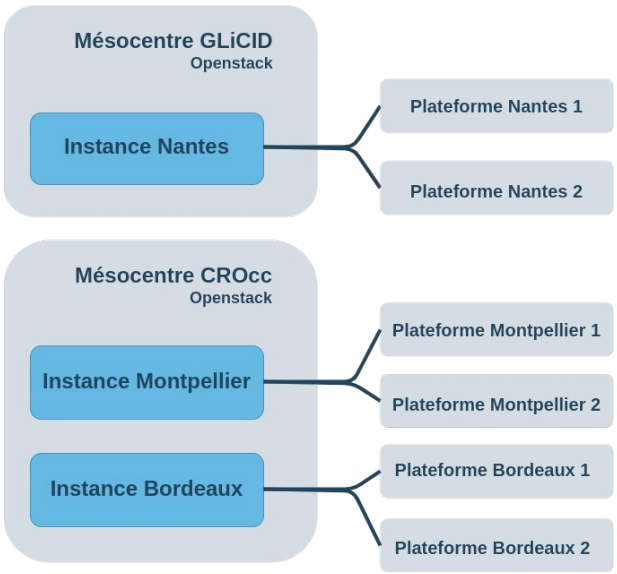
Bien que le serveur soit indépendant, il faut que les utilisateurs présents sur celui-ci correspondent aux utilisateurs présents sur les services du mésocentre. La création d'utilisateurs se faisant de manière centralisée sur Authentik, il faut alors récupérer ces utilisateurs pour les créer, les supprimer ou les mettre à jour sur l'instance SAMBA. Pour cela, nous avons développé un utilitaire Python, intégré a Omero-Quay, permettant de récupérer via l'API Authentik l'état des utilisateurs présents sur Authentik. En cas de coupure, cette mise à jour des utilisateurs sera donc le seul service impacté.

La mise en place d'un serveur Samba AD-DC est relativement complexe et demande une connaissance étendue des systèmes Linux. En effet, un serveur Samba AD-DC est à la fois un annuaire de nom, un serveur d'authentification, un serveur DNS et un système de fichiers en réseau. Il est donc indispensable de bien planifier l'architecture réseau en amont, pour le serveur comme pour les clients. Une erreur que nous avons faite et que nous aurions pu facilement éviter est celle du nom de domaine utilisé par le serveur AD-DC.

Lors de l'installation, nous avons configuré le domaine DNS du serveur pour qu'il gère la zone omero-fbi.fr, le serveur étant smb.omero-fbi.fr. Évidemment, ce nom de domaine est celui utilisé pour tous nos autres services web, et au moment de connecter notre Consumer iRODS au serveur central en irods-nte.omero-fbi.fr, aucun record n'est enregistré pour ce nom de domaine et le serveur est injoignable. Il aurait été plus judicieux d'utiliser une adresse en .local, type omerofbi.local pour que les requêtes vers des noms de domaines résolvables soit forwardés à un autre DNS.

5.3 Gérer un réseau multi-site interconnecté

Le projet FBI.data s'appuie sur les infrastructures de mésocentres nationaux pour faire fonctionner le cœur de son système. Ainsi il nous est possible de déployer plusieurs instances pour plusieurs sites qui souhaitent utiliser le projet. Aujourd'hui, une instance est disponible à Nantes et deux autres sont déployées à Montpellier : une pour une antenne de FBI à Montpellier et une autre pour une antenne à Bordeaux. Certains des services installés au mésocentre sont donc disponibles publiquement, notamment iRODS et OMERO. Pour chaque antenne de FBI, donc pour chaque ville, il existe plusieurs plateformes de recherches. Un diagramme simplifié résume l'architecture décrite ici :



Exemple d'interaction entre plateformes et mésocentres

Les plateformes de recherche communiquent donc avec leur mésocentre sur des interfaces publiques pour les services qui sont accessibles via un nom de domaine pleinement qualifié (FQDN). Pourtant, il n'est pas toujours possible ni souhaitable de faire transiter publiquement le trafic de certaines applications. Aussi, certaines plateformes de recherches ne sont pas accessibles sur un réseau public et ne sont donc pas accessible par machines hébergées aux mésocentres. Ce cas de figure s'est notamment posé pour iRODS, qui a besoin de pouvoir accéder aux ressources citées sur les consumers iRODS situés sur les plateformes pour pouvoir fonctionner. La question s'est également posée pour les logs et métriques, qui doivent transiter des plateformes vers le mésocentre mais qu'il n'est pas souhaitable de faire communiquer via des adresses publiques.

Une solution commune dans ce cas de figure est de créer un réseau privé virtuel (ou VPN) entre les différents sites.

Nous aurions pu utiliser des logiciels comme Wireguard ou OpenVPN pour créer ce genre de tunnels, mais ils nécessitent d'avoir un point d'entrée avec une IP publique, donc directement accessible sur internet et non derrière un *Network Address Translation (*NAT). Les différentes

configurations réseaux des mésocentres ne nous permettant pas toujours d'avoir une telle configuration, nous nous sommes tournés vers le logiciel **Netmaker**. Celui-ci est basé sur Wireguard, un VPN moderne, robuste et léger, mais propose autour de cette base un écosystème complet et très pratique. Le principal avantage de Netmaker est la possibilité d'utiliser un serveur *Session Traversal Utilities for NAT* (STUN) public afin d'exposer un serveur VPN qui se trouverait derrière un NAT, réglant ainsi notre problème de réseau.

Netmaker propose aussi une interface Web de gestion des configurations VPN, clients et serveurs, facilitant ainsi la création de nouvelles configurations pour les administrateurs systèmes des différentes plateformes.

6 Conclusion et perspective

Dans le cadre d'une infrastructure nationale en Biologie-Santé regroupant un ensemble de plateformes de microscopies pour la recherche fondamentale, nous avons proposé une architecture complète répondant aux contraintes posées par notre communauté et par l'évolution du paysage numérique. Notre objectif était de faciliter la transition des utilisateurs de ces plateformes vers l'utilisation des mésocentres régionaux, et vers l'application des principes FAIR en science ouverte. Nous passons désormais à une phase de production pendant laquelle l'accompagnement des utilisateurs va être critique.

La perspective immédiate pour 2025, travaillée avec les chercheurs et ingénieurs développant les solutions d'analyses de ces données produites, est le développement de l'offre de calcul sur ces données. Nos contraintes seront alors liées à différents usages de l'écosystème d'analyse d'image de microscopie en biologie. Les solutions envisagées sont de proposer différents type d'interface suivant le profil des utilisateurs et des caractéristiques des outils (notamment liée à l'interactivité et la visualisation): Jupyter notebook, VMs, interfaçage web direct via OMERO, accès SSH, en interfaçant les données avec le résultat des analyses de manière FAIR. La collaboration avec les mésocentres et l'appui sur l'offre qu'ils développent déjà sera donc essentielle pour offrir à notre communauté d'imagerie en biologie les outils nécessaires et adaptés.