ReproVIP Report associated to Reproducibility Dashboard (D1.1.2) and CI platform (D 1.2.1)

Hippolyte Blot, Axel Bonnet, Sorina Camarasu-Pop

ReproVIP Report associated to Reproducibility Dashboard (D1.1.2) and CI platform (D 1.2.1)

Hippolyte Blot, Axel Bonnet, Sorina Camarasu-Pop

[1] INSA-Lyon, UJM-Saint Etienne, CNRS, Inserm,
CREATIS UMR 5220, U1294, Lyon, France

Abstract. *The aim of this report is to present the ecosystem established within the framework of the ReproVIP project (ANR project 21-CE45-0024). It specifically concerns the functioning of the dashboard related to the study of the reproducibility of medical imaging applications offered by the VIP platform (vip.creatis.insa-lyon.fr).*

Résumé. *L'objectif de ce rapport est de présenter l'écosystème mis en place dans le cadre du projet ReproVIP (projet ANR 21-CE45-0024). Il concerne plus précisément le fonctionnement du dashboard lié à l'étude de la reproductibilité des applications d'imagerie médicale proposées par la plateforme VIP (vip.creatis.insa-lyon.fr).*

Contents

1. Introduction

La plateforme VIP[1], autour de laquelle repose le projet ReproVIP, permet d'utiliser différentes applications d'imagerie médicale au travers d'un portail web. Ces applications sont exécutées sur les ressources de calcul mises à disposition par l'infrastructure EGI pour l'organisation virtuelle biomed. En plus du portail web, la plateforme VIP dispose d'une API et d'un client Python, permettant d'automatiser le lancement des exécutions, ce qui est particulièrement utile pour le dashboard et la plateforme d'intégration continue.

Le Dashboard lié à VIP consiste en une plateforme web (actuellement accessible à l'adresse suivante : vip.creatis.insa-lyon.fr:9002) permettant de visualiser les résultats de plusieurs exécutions afin d'évaluer leur reproductibilité. Ces données proviennent donc majoritairement de la plateforme VIP à laquelle le dashboard est lié à partir de la plateforme d'intégration continue (CI). Cependant, ces données ne sont pas hébergées durablement sur VIP mais sur une plateforme nommée Girder, dont une instance est maintenue au sein du laboratoire CREATIS (pilot-warehouse.creatis.insa-lyon.fr).

Ce rapport décrira les trois principales entités (le dashboard, la plateforme Girder et le Gitlab CI) de manière individuelle (Section 2), ainsi qu'en les mettant en relation (Section 3).

2. Description des entités

2.1. Le Dashboard

Fonctionnalités utilisateur.

Le Dashboard a pour objectif de mettre en évidence, au travers de graphiques et de métriques, les éventuels problèmes de reproductibilité auxquelles font face les applications d'imagerie médicale. Il est principalement conçu pour s'interfacer avec la plateforme VIP en utilisant des résultats produits à partir de celle-ci, mais il est également capable de traiter des données provenant d'autres sources. Il faut cependant que le format des résultats étudiés soit pris en charge par le Dashboard. Les formats supportés actuellement sont les images NIfTI et les données tabulaires. Dans ce dernier cas, c'est l'utilisateur qui doit paramétrer la visualisation de ses graphiques en indiquant différentes options (cf. figure 2). On peut également configurer des graphiques et métriques spécifiques à une application. Actuellement, le dashboard prend en charge l'application cQUEST[2], pour laquelle il peut générer des graphiques et afficher des statistiques spécifiques, telle que la comparaison des amplitudes des métabolites.

Dans le Dashboard, une expérience contient un ensemble de résultats identifiés par une méta-donnée et correspondant à :

- Une application identifiée par son nom et numéro de version
- Un set de données d'entrée prédéfinies (comprenant l'ensemble des paramètres nécessaires à l'exécution de l'application)
- Un certain nombre de répétitions de l'exécution de cette version de l'application sur l'ensemble des données prédéfinies. En cas de parfaite reproductibilité (pour une application déterministe), les répétitions devraient produire des résultats identiques.

On peut scinder les fonctionnalités de visualisation et d'étude du Dashboard en deux parties relatives à la notion d'expérience :

1. Analyse de résultats intra-expérience (cf. Figure 1)
2. Analyse de résultats inter-expérience (cf. Figure 2)

Ce premier cas permet d'étudier la reproductibilité de l'application en observant la dispersion des résultats produits par la variation computationnelle de l'application, alors que ces résultats devraient être les mêmes pour des données d'entrées identiques.

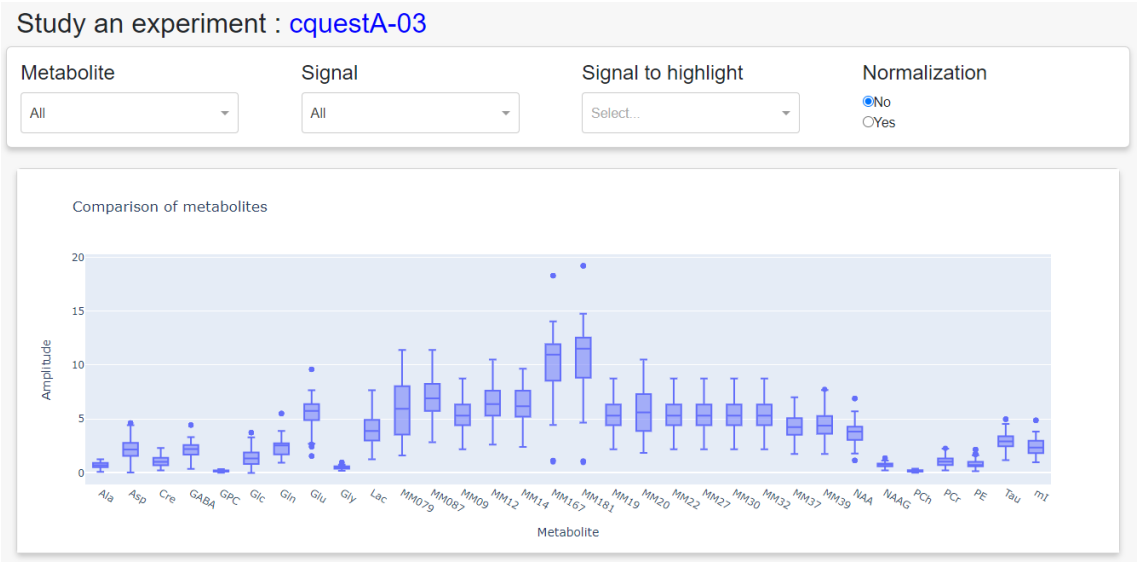

Figure 1. Exemple de graphique permettant d'étudier la reproductibilité d'une expérience

Dans le deuxième scénario, on peut comparer des résultats obtenus avec différentes applications ou avec différentes versions d'une même application, voire des paramètres différents. Ainsi, il est possible d'étudier la reproductibilité à une échelle plus large et de déterminer quels paramètres permettent d'obtenir la meilleure reproductibilité.

Il existe également une fonctionnalité permettant de visualiser des données tabulaires (à partir d'un fichier Excel par exemple) tout en laissant l'utilisateur paramétrer le rendu (choix du type de graphique, des données à afficher, des filtres à appliquer...). Il est ensuite possible d'exporter les paramètres du rendu pour les réappliquer sur d'autres données ayant ce format (cf. Figure 3).

Fonctionnement/Implémentions. Techniquement, le dashboard est réalisé en Python à l'aide d'un framework nommé Dash, développé et maintenu par Plotly.

Le dashboard s'appuie sur une base de données MYSQL référençant l'ensemble des expériences enregistrées, l'emplacement de stockage de leurs résultats ainsi que les applications et versions pouvant être analysées.

Le Dashboard met également à disposition une API permettant de demander une synchronisation de sa base de données avec les données présentes sur Girder. Lorsque l'URL https://vip.creatis.insa-lyon.fr:9002/api/girder_scanner est appelé, le Dashboard analyse les dossiers et fichiers présents sur Girder (présenté dans la partie suivante) dans le répertoire censé contenir les résultats à proposer pour ajouter les nouvelles expériences.

Le code source du dashboard est disponible sur github.com/virtual-imaging-platform/vip-reproducibility-dashboard.

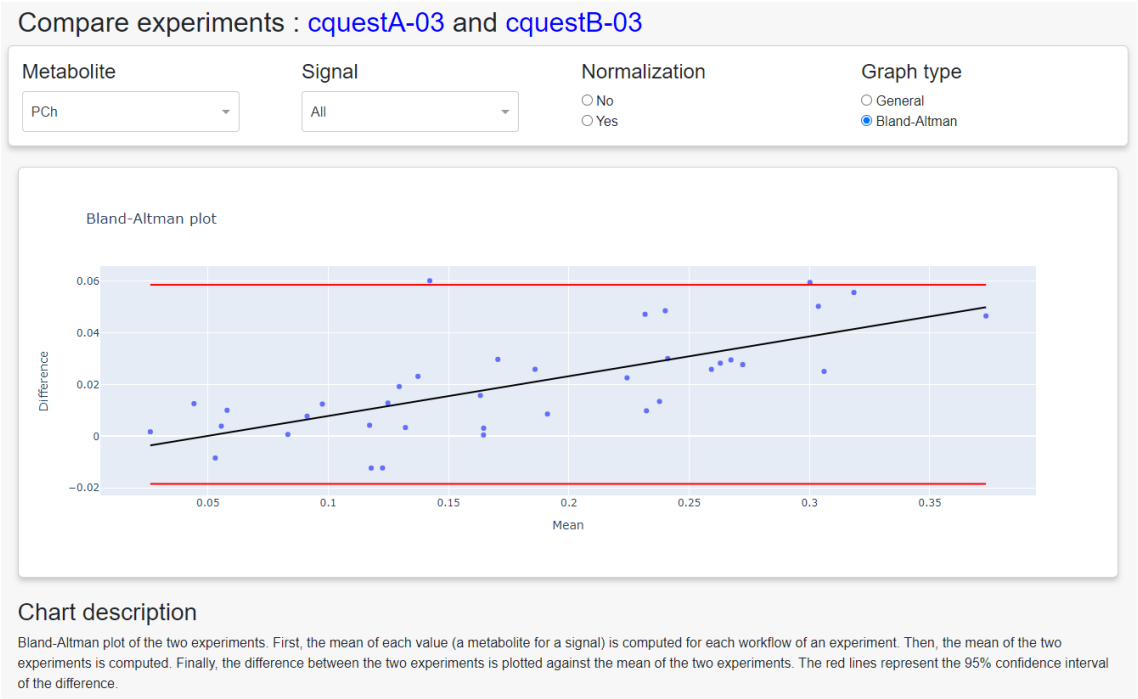

Figure 2. Exemple de graphique (tracé de Bland-Altman) permettant de comparer deux expériences correspondant à une même application avec des paramètres différents

2.2. Girder

Girder est une plateforme de gestion de données. Elle est développée par Kitware, est gratuite ainsi qu'open source. Une instance de ce service est maintenue au sein du laboratoire CREATIS pour la plateforme d'Imagerie multimodale expérimentale sur LyonTech (PILoT). L'atout de ce service est qu'il peut facilement être étendu avec de nouvelles fonctionnalités à l'aide du développement d'extensions, explicitement proposé et facilité par l'architecture de l'application.

VIP est essentiellement une plateforme de calcul ; son système de stockage basé sur des ressources EGI permet d'assurer la disponibilité des entrées/sorties pour les applications, mais il ne dispose pas des fonctionnalités d'une plateforme de gestion de données telle que Girder. Girder donne accès à des fonctionnalités avancées et permet un stockage à plus long terme. Les résultats des expériences définies dans le cadre du Dashboard sont automatiquement déposés par VIP sur Girder, mais leur format peut ne pas être adapté au Dashboard.

Dans le cas de cQUEST, une expérience va généralement contenir plusieurs centaines de fichiers. Si on veut les visualiser, il faut alors tous les télécharger, puis les agréger en un seul fichier pour enfin obtenir le rendu. En effectuant un pré-traitement pour regrouper ces données au préalable, le dashboard n'aura qu'à télécharger un seul fichier contenant l'ensemble des informations nécessaire (cf figure 4)

Un autre problème peut survenir lorsque la représentation que l'on veut obtenir des données nécessite un pré-traitement conséquent. C'est notamment ce qui arrive par exemple avec l'application BraTS[3] pour laquelle il est nécessaire de calculer une métrique

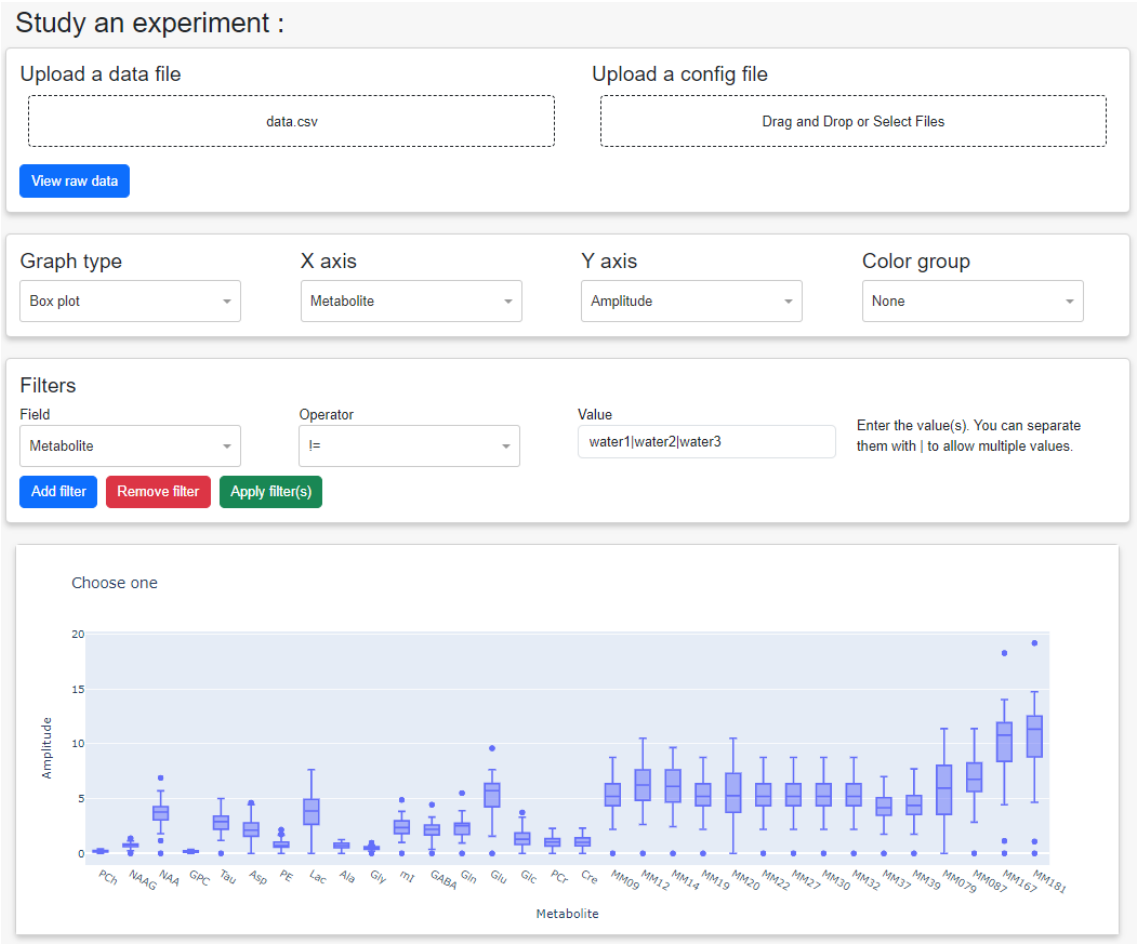

Figure 3. Exemple de graphique pouvant être obtenu en paramétrant un graphique basé sur des données de cQUEST

particulière (les chiffres significatifs tel que décrit dans [4]) demandant un grand temps de traitement.

Une extension pour Girder a donc été développée permettant de pallier ces problèmes. Celle-ci permet d'effectuer un traitement sur les résultats d'une application pour les mettre au bon format et faciliter leur utilisation par le Dashboard (Dépôt Github de l'extension). Ces traitements sont effectués par des conteneurs Docker, ce qui permet de mettre à jour ce traitement sans avoir à modifier directement le code de l'extension intégrée à Girder (Dépôt Github des images Docker).

Techniquement, l'extension repère les données correspondant à une expérience à l'aide des métadonnées ajoutées à un dossier. Si un dossier possède la métadonnée experiment_id (ou autre en fonction du paramétrage), toutes les données qu'il contient seront considérées comme des données à convertir. Un script est alors appelé, dans un conteneur Docker, pour traiter ces données. Le script diffère en fonction de l'application et de la version (qui sont donnés en paramètres du script tel que représenté sur la figure 6), permettant ainsi un traitement différent pour chaque format de donnée.

Plusieurs variables doivent être paramétrées pour s'assurer du bon fonctionnement de l'extension, tel qu'illustré dans la Figure 5.

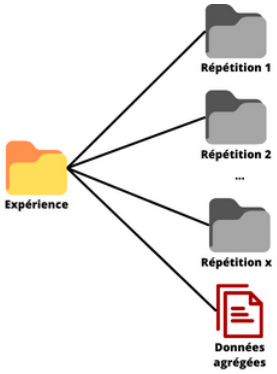

Figure 4. Schéma représentant l'agrégation des résultats de cQUEST

1. Location of the girder storage folder on the host : Le chemin absolu vers le répertoire où les fichiers temporaires générés par les processus de conversion seront stockés
2. Girder id of the folder where the converted experiments will be stored : Identifiant donné par Girder au dossier dans lequel les résultats convertis devront être stocké par l'extension.
3. Name of the attribute in the metadata that identifies experiments : Nom de la métadonnée choisi pour identifier une expérience. Doit être identique au nom choisi dans le script de la CI (parie suivante).
4. Location of the applications.json file on the host : Emplacement où est stocké le fichier référençant les différents conteneurs disponibles pour la conversion par application et par version (exemple de fichier présent sur le dépôt des images).

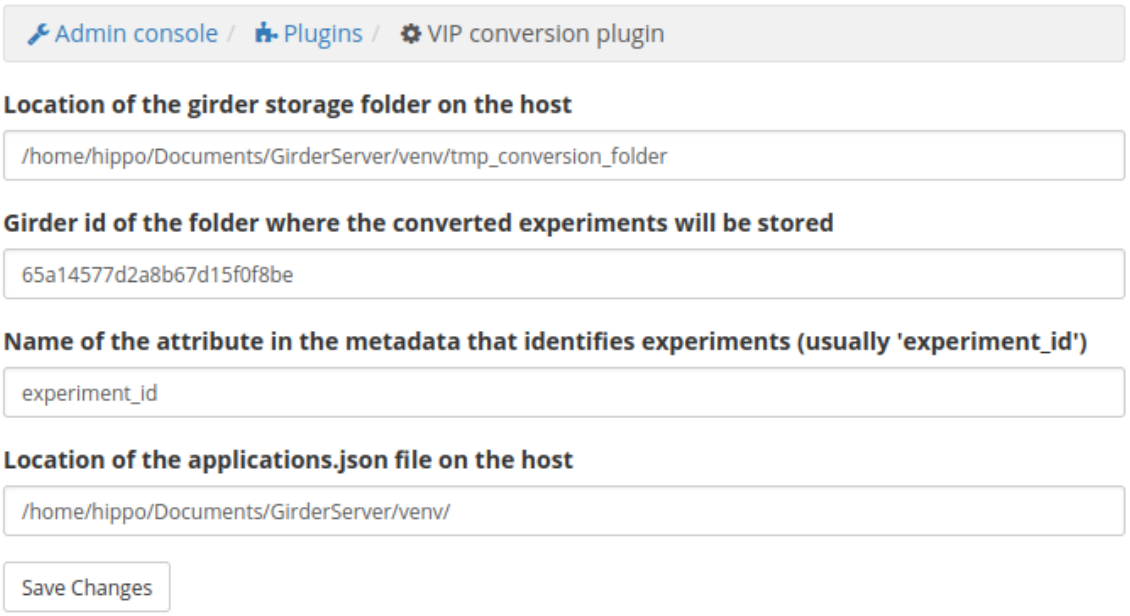

Figure 5. Capture d'écran de l'interface de paramétrage de l'extension

Cette extension propose également une API qui permet de lancer des conversions, en plus d'une IHM présente à l'adresse https://pilot-warehouse.creatis.insa-lyon.fr/#conversion.

Il est donc possible de lancer une conversion en appelant la route convert avec les paramètres "application", "version", "experiment_id" et "container_id" avec une requête GET.

2.3. GitLab CI

Les deux entités précédentes permettent le pré-traitement ainsi que la visualisation des données. Il manque cependant un moyen de lancer une expérience pour ajouter des données au Dashboard. C'est le rôle du CI (Intégration Continue) de GitLab. Il permet aux utilisateurs autorisés d'exécuter une application en plaçant les résultats sur Girder. Le script s'exécutant sur le CI va ensuite envoyer une requête au dashboard pour ajouter l'expérience. Le principal avantage de ce système est qu'il propose une interface utilisateur simple d'utilisation, ce qui permet d'ajouter des expériences facilement.

C'est le dépôt ReproVIP de l'instance IN2P3 de GitLab qui est utilisé. Il faut ensuite accéder à la section Build -> Pipelines -> Run pipelines, puis se positionner sur la branche develop et lancer les pipelines. L'interface propose alors d'instancier des variables d'environnement permettant de paramétrer l'expérience (figure 6) : Celles-ci

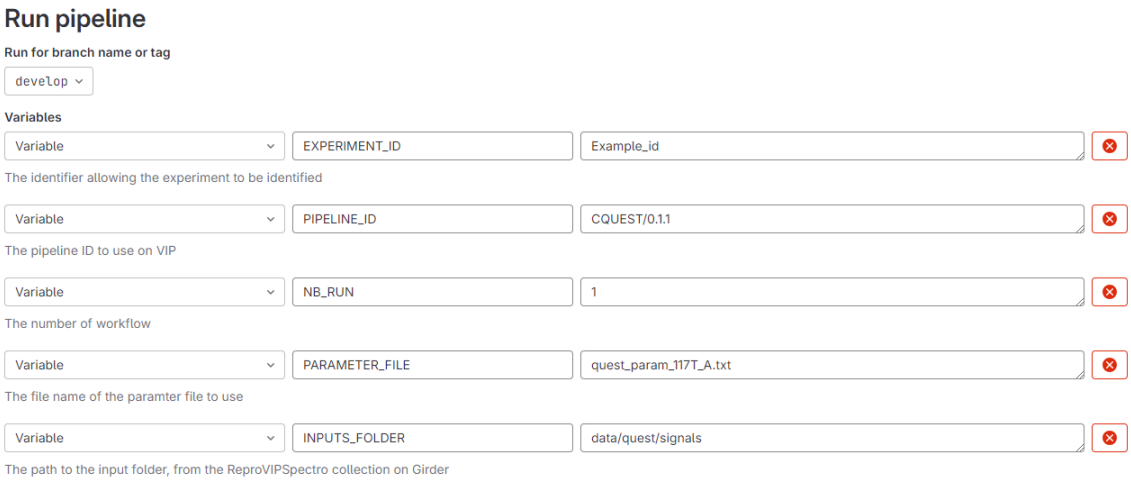

Figure 6. Capture d'écran de l'interface d'instanciation des variables d'environnement de la CI

doivent être renseignées comme suit :

1. EXPERIMENT_ID : L'identifiant de l'expérience qui sera utilisé comme nom sur le Dashboard
2. PIPELINE_ID : Identifiant du pipeline sur VIP. Il s'agit du nom de l'application et de sa version, séparés par un "/".
3. NB_RUN : Nombre d'itérations de l'exécution, c'est à dire le nombre de fois où du même pipeline est exécuté, sur les mêmes données d'entrée, pour étudier la reproductibilité de la version de l'application.
4. PARAMETER_FILE : Fichier de paramètres à utiliser dépendant de l'application.
5. INPUTS_FOLDER : Données d'entrées utilisées par l'application pour produire l'expérience.

En plus de ces 5 paramètres, il est nécessaire de renseigner les clés des API de VIP et Girder. Pour ce faire, il faut se rendre dans les paramètres du projet, CI/CD, puis variables.

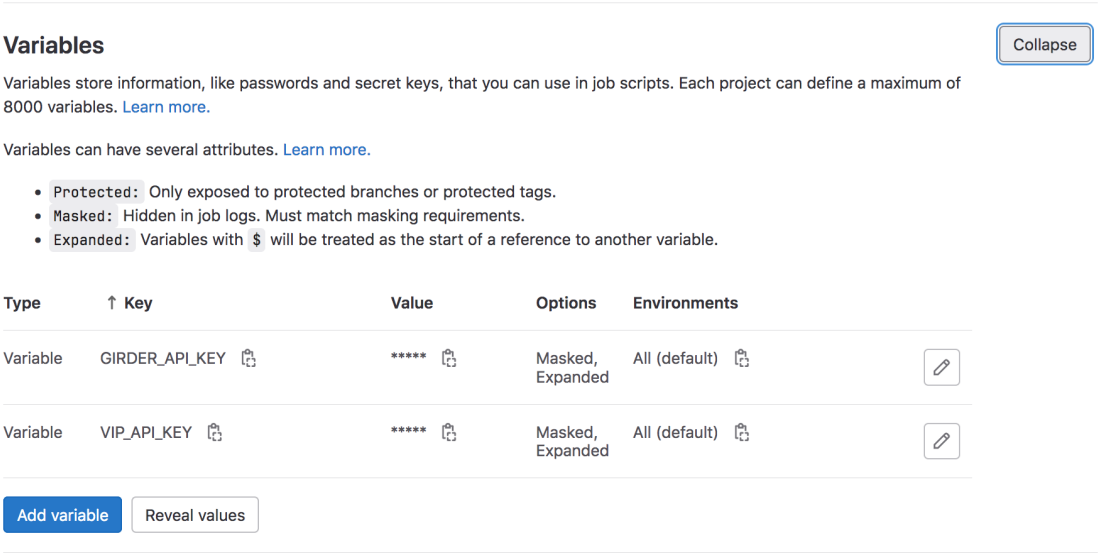

Figure 7. Capture d'écran de l'interface d'instanciation des variables d'environnement de la CI pour le projet

Il faut ensuite créer les deux variables suivantes avec les clés d'API comme valeurs (cf. Figure 7)

La script peut ensuite être lancé, ce qui implique les différentes interactions présentées sur la Figure 8. Ces échanges sont automatiquement engendrés par le script à l'aide des différentes API proposées par les entités avec lesquels il communique.

3. Interactions entre les entités

L'enchaînement d'étape qui permet d'ajouter des données au dashboard est le suivant :

1. Lancement d'une série d'exécutions (autrement appelé expérience) sur VIP à partir du CI / plateforme d'intégration continue de Gitlab.
2. Stockage des données de sorties (outputs) sur Girder
3. Prétraitement de ces données sur Girder (réalisé à l'aide d'un plugin consistant par exemple en l'agrégation de multiples fichiers en un seul, calcul de métriques...)
4. Ajout de ces nouvelles données prétraitées au dashboard

La Figure 8 représente ces interactions de manière détaillée.

Précisions : Dans la Figure 8, l'entité GitLab représente le service d'intégration continue de la plateforme qui est utilisé pour permettre aux utilisateurs de lancer des expériences.

4. Conclusions

Nous avons présenté l'écosystème mis en place dans le cadre du projet ReproVIP pour faciliter l'évaluation de la reproductibilité des résultats scientifiques. Cet écosystème est composé d'un dashboard, d'une platforme de gestion de données et d'un outil d'intégration continue. Il est relié à la platforme VIP, qui permet d'exécuter plus d'une vingtaine d'applications scientifiques sur des ressources de calcul distribuées.

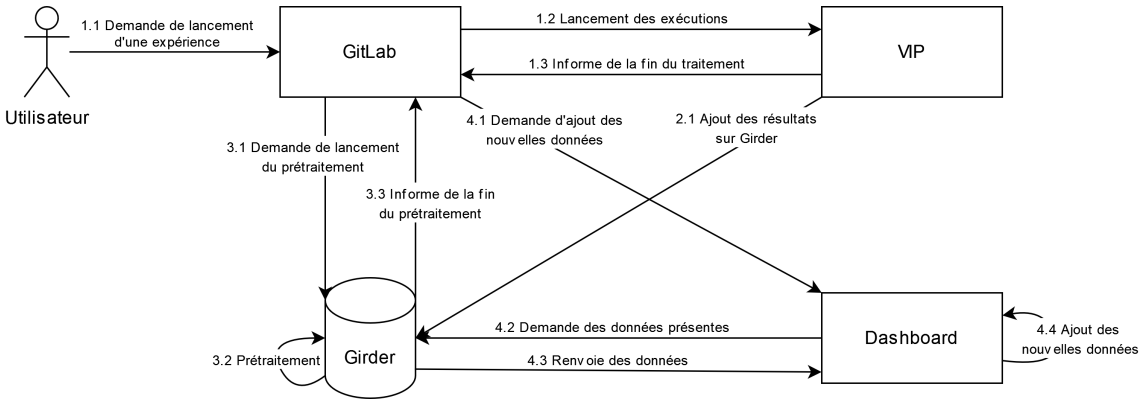

Figure 8. Interactions au sein de l'écosystème du Dashboard

Le Dashboard est accessible librement à l'adresse suivante : vip.creatis.insa-lyon.fr:9002. Il s'agit pour l'instant d'une première version, qui évoluera en fonction des besoins et retours des utilisateurs qui peuvent nous contacter à l'adresse vip-support@creatis.insa-lyon.fr.

References

[1] Tristan Glatard, Carole Lartizien, Bernard Gibaud, et al., "A virtual imaging platform for multi-modality medical image simulation," *IEEE Transactions on Medical Imaging*, vol. 32, pp. 110–118, 2013.

[2] Helene Ratiney, Mark J Albers, Herald Rabeson, and John Kurhanewicz, "Semi-parametric time-domain quantification of hr-mas data from prostate tissue," *NMR in biomedicine*, vol. 23, no. 10, pp. 1146–1157, 2010.

[3] Bjoern H. Menze, Andras Jakab, Stefan Bauer, et al., "The multimodal brain tumor image segmentation benchmark (brats)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015.

[4] Morgane Des Ligneris, Axel Bonnet, Yohan Chatelain, et al., "Reproducibility of Tumor Segmentation Outcomes with a Deep Learning Model," in *International Symposium on Biomedical Imaging (ISBI)*, Cartagena de Indias, Colombia, Apr. 2023.