Jupyter Lego ou la fin de l'enfer des dépendances ?

Rémi Cailletaud*[1], Glenn Feunteun[2], and Antoine Pons[2]

[1]Observatoire des Sciences de l'Univers de Grenoble – CNRS – France
[2]Zenika – Sans laboratoire – France

Résumé

Gaia Data est un projet qui vise à développer une plateforme intégrée et distribuée de services et données pour l'observation, la modélisation et la compréhension du système Terre, de la biodiversité et de l'Environnement. L'un des nombreux objectifs du projet est de déployer des environnements virtuels de recherche (VRE) intéropérables dans les 8 centres " ossature " qui le composent. Durant une première phase du projet, la cible s'est portée sur JupyterHub qui est largement utilisé dans les communautés et est souvent déjà déployé dans les centres, sous différentes formes (Docker, HPC, Kubernetes). Le format Open Container Initiative (OCI) Image a été choisi pour diffuser les environnements, car il offre de bonnes propriétés (utilisation de registres, environnement immuable, scan de vulnérabilité aisé,...).

Préparer des environnements Jupyter exhaustifs est une véritable gageure, d'autant plus quand la cible est très large : on se heurte rapidement à des problèmes de dépendances entre les codes des différents utilisateur·ices et parfois avec la partie purement système de Jupyter Lab. D'un autre côté, produire une image par cas d'usage n'est pas pertinent pour plusieurs raisons :
- les utilisateur·ices veulent pouvoir interagir avec plusieurs codes sans quitter l'environnement ;
- la maintenance d'un telle flotte d'images peut rapidement devenir compliquée et chronophage.

Une première idée est d'utiliser l'extension *nb_conda_kernels*[1] qui permet à Jupyter Lab d'accéder aux kernels d'autres environnements conda. Cela permet :
1. le découplage de la partie " système " de la partie métier ;
2. l'utilisation de plusieurs codes sans quitter l'environnement VRE.

Cependant, le problème de la maintenance des images reste entier, d'autant qu'avec cette approche, la combinatoire des images possibles explose... Sauf si on peut construire notre environnement dynamiquement, au lancement !

La deuxième idée est de tirer parti d'une fonctionnalité récente de Docker et Kubernetes (déjà existante depuis plus longtemps chez Apptainer/Singularity) qui permet de monter une image OCI dans un conteneur. En les montant à un point de l'arborescence où l'extension *nb_conda_kernels* peut les détecter, nous pouvons composer des images dynamiquement.

Nous pouvons dès lors fournir un ensemble réduit d'image " système " et un plus grand nombre d'image " métier ", laissant le choix et la charge aux différents centres de composer leurs environnements finaux en assemblant les différentes briques. Le premier type d'image

*Intervenant

fournit l'ensemble de la plomberie nécessaire à faire tourner Jupyter Lab alors que le second ne contient que les environnements conda métier. Les deux types d'images sont construits via des pipelines d'intégration continue. Les premiers centres de Gaia Data les ont déjà déployés en test.

(1) https://github.com/anaconda/nb_conda_kernels