Exécuter du code utilisateur dans un bac à sable

Une bonne manière d'introduire un nouveau concept est de faire pratiquer l'utilisateur avec du contenu interactif. Pour tester un langage de programmation dans le navigateur internet, il est courant d'utiliser des sites proposant un *"bac à sable"*. L'utilisateur soumet son code au serveur, qui compile le code, exécute et retourne le résultat au client.

Créer un bac à sable

Moshell est un langage de script, créé au sein de l'IUT. Dans le but de créer une documentation attractive, l'idée est venue de proposer un *bac à sable* pour tester le langage et l'exécuter. Étant donné qu'il s'agit d'un langage dans l'esprit du shell, il nous paraît important que l'utilisateur puisse créer, lire et supprimer des fichiers. Pour éviter que l'utilisateur supprime définitivement des fichiers indispensables au fonctionnement de l'application, il est nécessaire d'utiliser une technologie de conteneurisation, telle que *Docker* et ses variantes, ou un hyperviseur. Exécuter du code que l'utilisateur a pu arbitrairement saisir est dangereux, et les conteneurs ne sont pas une technologie de *sandboxing*. Il faut alors les coupler avec des outils tel que *Bubblewrap* et limiter les appels systèmes disponibles.

Ce système peut être étendu à d'autres langages de programmation.

Distribuer les ressources

Le code fournit par l'utilisateur n'est pas à l'abri de contenir une boucle infinie et un acteur malveillant peut tout à fait tenter de surcharger le *bac à sable* de requêtes non légitimes dans un intervalle de temps très court. Il convient alors au backend du *bac à sable* de limiter les ressources allouées aux conteneurs (mémoire, temps d'exécution, CPU...) et d'établir une file d'attente au cas où le site est saturé.

Il est par ailleurs intéressant de distribuer les *runners* sur plusieurs machines, ce qui demande alors au site internet de savoir vers quel serveur envoyer la demande d'exécution. C'est un protocole de communication à définir avec un système de messagerie tel *RabbitMQ*.

Réaliser un frontend et un backend web

L'utilisateur est invité à saisir son code directement sur le site web, avec un éditeur intégré comme *Ace* ou *Monaco* en *JavaScript*. Il peut voir le code s'exécuter, soit en direct, soit lorsque le conteneur a complétement terminé son exécution. Le temps de chargement du *bundle* peut être lent sur les petites connexions, quels chargements peut-on différer plutôt que de tout charger à l'ouverture de la page ?

Le backend sait notifier le client quand son code est exécuté, demandant l'utilisation de *server-sent events* ou *WebSockets*.

Déployer en continu

Un langage de programmation subit des modifications constamment, le bac à sable doit savoir rester à jour. L'image OCI utilisée doit pouvoir être reconstruite facilement, dans des environnements d'intégration et de déploiement continu par exemple.