Jenkins

Jenkins est un outil d'intégration continue pour un projet informatique. L'"**intégration continue** est le fait de, lors de l'évolution d'un projet informatique, vérifier que les nouvelles modifications s'intègrent bien au programme existant, et n'introduisent pas de bug ou ne supprime pas de fonctionnalité.

Pour ce faire, **Jenkins** permet d'enregistrer des commandes (généralement des commandes terminal) qui seront automatiquement exécutées lorsqu'une modification est apportée au projet. En pratique, **Jenkins** peut être lié à un projet GitLab, et donc exécuter les commandes à chaque commit. Les commandes en question peuvent être de différents types:

Lancer une suite de tests unitaires CUnit.
Utiliser **valgrind(1)** pour détecter les fuites de mémoire (voir la section dédiée pour plus d'informations).
Utiliser **cppcheck** pour analyser le code et détecter de potentiels bugs.

La suite de ce document détaille comment inclure **Jenkins** à un projet GitLab, pour profiter de l'automatisation des tests à chaque commit, et éviter les bugs et la perte de fonctionnalité.

Création du projet sur Jenkins

Se connecter sur **https://jenkins.student.info.ucl.ac.be** et cliquer sur le bouton *S'identifier*.

S'identifier à l'aide du compte UCLouvain.

Une fois connecté, dans le menu à gauche, cliquer sur *Nouveau Item*.

Saisir le nom du projet, choisir *Construire un projet free-style*, et cliquer sur *OK*.

Le menu de configuration s'ouvre. Cocher la case *Activer la sécurité basée projet*.

Ajouter les membres du projet à l'aide de leur Jenkins User ID, visible sur leur profil Jenkins.

Donner tous les droits aux membres du groupe (cocher toutes les cases). Attention à ne pas donner tous les droits aux anonymes !

Dans le menu déroulant *GitLab Connection*, choisir *Forge UCLouvain*.

Dans *Gestion de code source*, choisir l'option *Git*.

Récupérer l'URL du dépôt git sur GitLab, avec l'option *Clone with SSH*.

Indiquer l'URL du dépôt git récupérée à l'étape précédente, et choisir *git (SSH key - jenkins_ingi - forge.uclouvain.be)* comme identifiant.

Dans *Ce qui déclenche le build*, cocher *Build when a change is pushed to GitLab*. **Noter quelque part le web hook URL précisé juste à côté**.

Un cadre s'ouvre, cliquer sur le bouton *Avancé*. En dessous du champ *Secret token* se trouve un bouton *Generate*. **Générer un token et le noter aussi**.

Dans *Build*, ajouter une étape *Exécuter un script shell*. Y introduire :

```
#!/bin/bash

exit 0
```

Dans *Actions à la suite du build*, ajouter une action *Publish build status to GitLab*.

Cliquer sur *Sauver* pour construire le projet.

Déclenchement automatique des builds

Sur GitLab, dans la vue du projet, aller dans le menu *Settings => Webhooks*
Copier le *web hook URL* ainsi que le *Secret token* mis de côté au point précédent. Cocher *Push events* et finalement cliquer sur *Add webhook*.

Au prochain commit, un build sera déclenché automatiquement. Comme le script shell retourne 0, le build passera tout le temps. Si le lien est fonctionnel, une pastille verte s'affichera à côté de la description du commit dans GitLab.

Modifier le script sur Jenkins

Sur Jenkins, dans la vue du projet cliquer sur le menu *Configurer*
Aller jusqu'à la section *Build* et modifier le script shell
Rajouter les commandes nécessaires à l'exécution des tests de votre projet (`make tests` par exemple)

Si le code ne passe pas tous les tests, la valeur de retour de la commande sera différente de 0 et le build sera marqué comme *Failed*, une pastille rouge apparaîtra dans GitLab.

Projet d'exemple pour Jenkins

Nous vous fournissons un projet donnant un exemple plus complet de configuration d'un projet GitLab avec Jenkins. Il est disponible à l'adresse suivante : **https://forge.uclouvain.be/alegay/jenkinslepl1503/**. Le projet contient les fichiers suivants :

- `ex-lepl1503.c`: un fichier C qui contient:
  - `int maxi(int, int)`: une fonction qui calcule le maximum entre deux entiers
  - `void test_maxi(void)`: un test correct (oui le maximum entre 0 et 2 est bien 2)
  - `void test_maxifailed(void)`: un test incorrect (non le maximum entre 0 et 2 n'est pas 0!)
  - `void erreurmalloc(void)`: une fonction qui fait une assignation à une case mémoire non allouée
  - Une procédure main qui :
    - définit une suite de tests CUnit basée sur `test_maxi` et `test_maxifailed`. Pour plus de détails sur CUnit, reportez-vous à la section dédiée.
    - appelle la fonction `erreurmalloc`.
- Un fichier `Makefile` qui comprend plusieurs règles. Pour plus de détails sur la conception du makefile, reportez-vous à la section dédiée.

Voici quelques exemples d'utilisation du projet :

- `make` : compile le programme avec l'option -lcunit et appelle les outils *valgrind* et *cppcheck*. Les résultats sont sauvés dans des fichiers xml.
- `make clean` : efface l'exécutable et les .xml.

Ne pas hésiter à faire un clone de ce projet sur votre machine, et jouer avec la commande `make` pour bien comprendre comment le programme et ses différents outils fonctionnent. Comme vous êtes un·e utilisateur·rice *guest*, vous ne pourrez pas faire de commit dans ce projet. **Attention** : si vous n'avez pas installé CUnit, valgrind ou cppcheck sur votre machine, vous ne pourrez pas compiler le programme ou effectuer les tests.

Vous pouvez voir sur la page principale du projet sur GitLab une croix rouge à côté du dernier commit effectué. Cela vient du projet Jenkins lié à ce GitLab, et montre que ce commit n'a pas réussi tous les tests. Ce projet Jenkins est disponible à l'adresse suivante : **https://jenkins.student.info.ucl.ac.be/job/ex-lepl1503/**. **Attention** : si vous n'êtes pas connecté·e sur Jenkins, vous arriverez sur une page avec une erreur 404. Dans ce cas, retournez sur la page d'accueil de Jenkins (**https://jenkins.student.info.ucl.ac.be**), connectez-vous et réessayez.

Lors de votre connexion, vous verrez qu'un certain nombre de commits ont été faits. Certains avec succès (les bleus) d'autres pas (les rouges). Vous pourrez aussi observer les résultats de CUnit, valgrind et cppcheck. Rendez-vous dans le menu *Configurer*. Vous y trouverez les actions faites avant et après le build : *Ce qui déclenche le build*, *Build* et *Actions à la suite du build*. Vous verrez par exemple dans *Actions à la suite du build*, l'utilisation de plugins pour afficher les rapports XML générés par `make`.

Notez que vous n'avez pas le droit de modifier cette configuration. Cela se vérifie dans l'onglet *Activer la sécurité basée projet* du menu *configure* où seul l'utilisateur Axel Legay a tous les droits. Lorsque vous lierez votre Jenkins à votre projet GitLab, pensez à donner tous les droits à tous les utilisateur·rices mainteneur·ses de votre projet en utilisant *Add user or group*.