TD outil de développement logiciel : Intégration continue

Rendre un rapport au format pdf + code sous forme d'archive à la fin du TP sous Celene ; un rapport par étudiant

Consignes générales

- Tout ce qui est nécessaire au TP est installé dans la VM. Il y a un compte "sti" mot de passe "sti"
- Vous rendez vos rapports pour le contrôle continu sur Celene, avant minuit le jour du TP, sous forme d'archive contenant un rapport en pdf ainsi que tout code utile
- **Par défaut le clavier est en anglais,** `setxkbmap fr` **résoudra ce problème.**
- Aujourd'hui la VM vous servira de Runner ! Vous pouvez au choix aussi vous en servir pour développer ou utiliser votre station de travail.

Prologue au TP

Le but de cette séance est de vous familiariser avec les pratiques et l'écosystème logiciel typiques d'une entreprise dont l'activité principale est le développement.

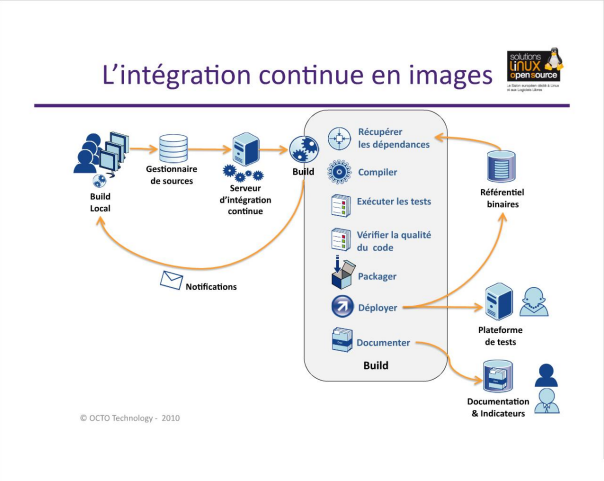
En entreprise, on ne développe jamais seul : le plus souvent, on fait partie d'une équipe de développement qui produit du code. La structuration d'une équipe et sa manière de travailler peut être très variée ; on retrouve souvent un certain nombre de rôles clefs (architecte, chef de projet, responsable qualité, développeur, expert), les activités des divers acteurs étant coordonnées par l'application d'une méthode de développement (de nos jours celle-ci est souvent *agile*).

Les activitités sont variées : produire, tester le code de manière unitaire, le mettre à disposition, l'intégrer et effectuer des tests d'intégration, pour qu'il puisse *in fine* être déployé dans un environnement de production.

Lors des derniers TP, vous avez appris les grandes familles de tests, à coder des tests en utilisant JUnit et évaluer la couverture de vos tests. Vous avez mis votre code à disposition en utilisant un gestionnaire de version de code (le gitlab de l'école). L'étape suivante est de le testet en utilisant l'outil d'intégration continue qui lui est adjoint : gitlab runner. On utilise toujours la même VM. Cette fois ci elle nous servira de Runner, le code pouvant au choix être géré depuis la machine hôte, la VM, ou via l'interface Web de gitlab qui vous permet de faire des modifs en ligne.

Intégration continue

Le schéma général de l'intégration continue (IC ou CI en anglais) est donné à la figure suivante.



L'idée est simple : automatiser côté serveur le passage des tests, la vérification de le documentation, etc, etc. Pour éviter de surcharger le serveur, gitlab utilise des Runners : ce sont eux qui vont faire tout le boulot, le serveur gitlab se contentant de coordonner le fait qu'un Runner effectue une tâche.

Nous allons nous contenter, lors de ce TP, d'automatiser le build et le passage des tests unitaires que vous avez développés pour votre cher projet arithmetics. Pour cela, la VM que nous avons utilisé jusque là a été configurée pour pouvoir être un runner (installation d'un paquet, pas très compliqué à faire ...).

Introduction à gitlab CI

le fonctionnement de l'intégration continue (*Continuous Integration* ou CI) de Gitlab repose sur 3 notions importantes :

— Les runners sont des ressources capables d'exécuter des tâches. Typiquement, un runner peut être une VM, un container (un pool de ressources isolées dans une machine), une machine physique ...

— Les tâches qu'un runner effectue sont appelés des jobs. Quand l'intégration et le test d'un projet requièrt l'exécution d'un job, celui-ci est mis dans une liste d'actions à effectuer. Un runner libre vient régulièrement quémander du travail, celui-ci lui sera envoyé par le serveur.

Les tâches à effectuer pour un projet sont regroupées au sein d'un Pipeline. Les pipelines en cours et terminés sont visibles sur la page d'accueil de votre projet (un onglet à droite).

Pour configurer un Pipeline, il va falloir le décrire. Un pipeline est constitué de 4 *stages* (étapes) : *build, test, staging* et *production*. Nous nous arrêterons aux 2 premiers, les autres étant dédiés à des projets où le code est déployé sur une infrastructure complexe.

Ces étapes sont décrites de manière très simple dans un fichier spécial appelé gitlab-ci.yml. La syntaxe est très simple (la doc complète est là : `https://docs.gitlab.com/ce/ci/yaml/`). Le principe est le suivant : on donne d'abord les stages que l'on va définir, puis on décrit des tâches qui vont appartenir à l'un ou l'autre des stages. Voilà un exemple simple :

```
stages:
  - build
  - test

ci_build:
  stage: build
  script:
    - ant clean
    - ant jar

ci_test:
  stage: test
  script:
    - ant clean jar junit
```

Ce petit fichier donne 2 stages (build et test) et deux tâches (ci_build et ci_test) qui appartiennent chacune à un des stages. Le script est une commande qui va être lancée sur le Runner. Par chance la VM contient déjà ant, donc cela devrait passer. Les appels au cible ant correspondent bien sûr à des cibles existantes dans le build.xml du projet.

Une fois ce fichier écrit, il ne reste plus que 2 choses à faire : 1) configurer le CI sur le serveur et récupérer un token spécifique au projet que l'on utilise pour 2) Configurer le Runner pour qu'il se connecte au serveur et dise qu'il est prêt à exécuter les tâches des pipelines d'un projet spécifique.

Configuration du CI sur gitlab

2 possibilités : soit éditer localement le fichier `.gitlab-ci.yml` soit passer par l'interface web gitlab :

— Aller sur le projet, cliquer à droite sur "setup CI"

— Editer le fichier puis le commiter.

Configuration du Runner

pour configurer le Runner :
— Exécuter la commande `sudo gitlab-runner register`
— Rentrez le nom du serveur
— Donnez le token que vous trouverz via l'interface Web gitlab dans Project puis Setting tout à droite, et ensuite Pipelines sur la ligne en dessous.
— Répondez au reste des questions jusqu'à la fin, et c'est fini.

1 La Question

Votre but aujourd'hui est de lancer les tests sur le Runner (cible ant -> junit) pour votre petit projet personnel contenant vos propres tests. Le rendu sera l'archive de votre projet que vous clonerez une fois fini et que vous rendrez sur Celene.

Pour que vous ne soyez pas perdu, vous trouverez dans le groupe OutilDeDEv un projet - arithmetics - qui contient un petit exemple de projet pour lequel un Runner fonctionne. Celui-ci contient juste un test pour la classe TestHelloWorld. A vous d'adapter la démarche à votre projet...