

## ATELIER INTEGRATION CONTINUE AVEC LE SERVEUR JENKINS

### Conception de workflow de Build (Pipeline)

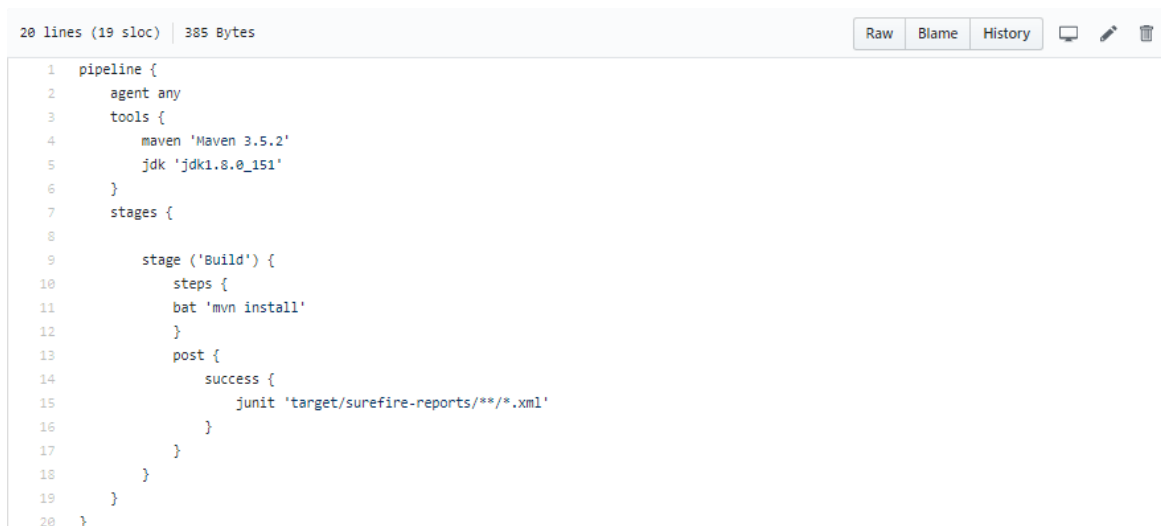
Jusqu'à présent vous avez lancé des jobs simples, chacun de ces derniers appelait une cible Maven pour réaliser une fonction. Par exemple, vous avez lancé une tâche de build capable de récupérer le code source depuis un gestionnaire de code source (GitHub) et vous avez généré une application fonctionnelle.

Dans le deuxième atelier, vous avez prolongé le concept d'intégration continue au déploiement continu. C'est-à-dire au lieu de déployer l'application en local, il est judicieux de pouvoir le déployer directement dans le dépôt de l'entreprise (Dans notre cas, nous avons étudié le cas de Nexus).

Dans l'atelier d'aujourd'hui, vous allez expérimenter le principe de pipeline de build (workflow). Un pipeline de déploiement est une façon de définir d'orchestrer vos build au travers d'une série de passages garantissant la qualité. Le passage d'une étape à une autre peut être automatisé comme il peut être manuel.

### Etapes à suivre:

- 🖥️ Faire un fork du projet: <https://github.com/spring-projects/spring-petclinic.git>
- 🖥️ Dans le dépôt github ajouter un fichier appelé Jenkinsfile contenant la syntaxe indiquée dans la figure ci-dessous: il s'agit de spécifier explicitement le cycle de vie d'un build sous forme d'un script GROOVY
- 🖥️ Commiter les modifications apportées sur le fichier Jenkinsfile



```
20 lines (19 sloc) | 385 Bytes
1 pipeline {
2   agent any
3   tools {
4     maven 'Maven 3.5.2'
5     jdk 'jdk1.8.0_151'
6   }
7   stages {
8
9     stage ('Build') {
10      steps {
11        bat 'mvn install'
12      }
13      post {
14        success {
15          junit 'target/surefire-reports/**/*.xml'
16        }
17      }
18    }
19  }
20 }
```

- 🖥️ Dans Jenkins, créer un job de type Pipeline.
- 🖥️ Configurer le job de façon à scruter le SCM (GitHub) : scruter le source code management veut dire récupérer la dernière version du code source du logiciel disponible sur votre dépôt github.

**Pipeline**

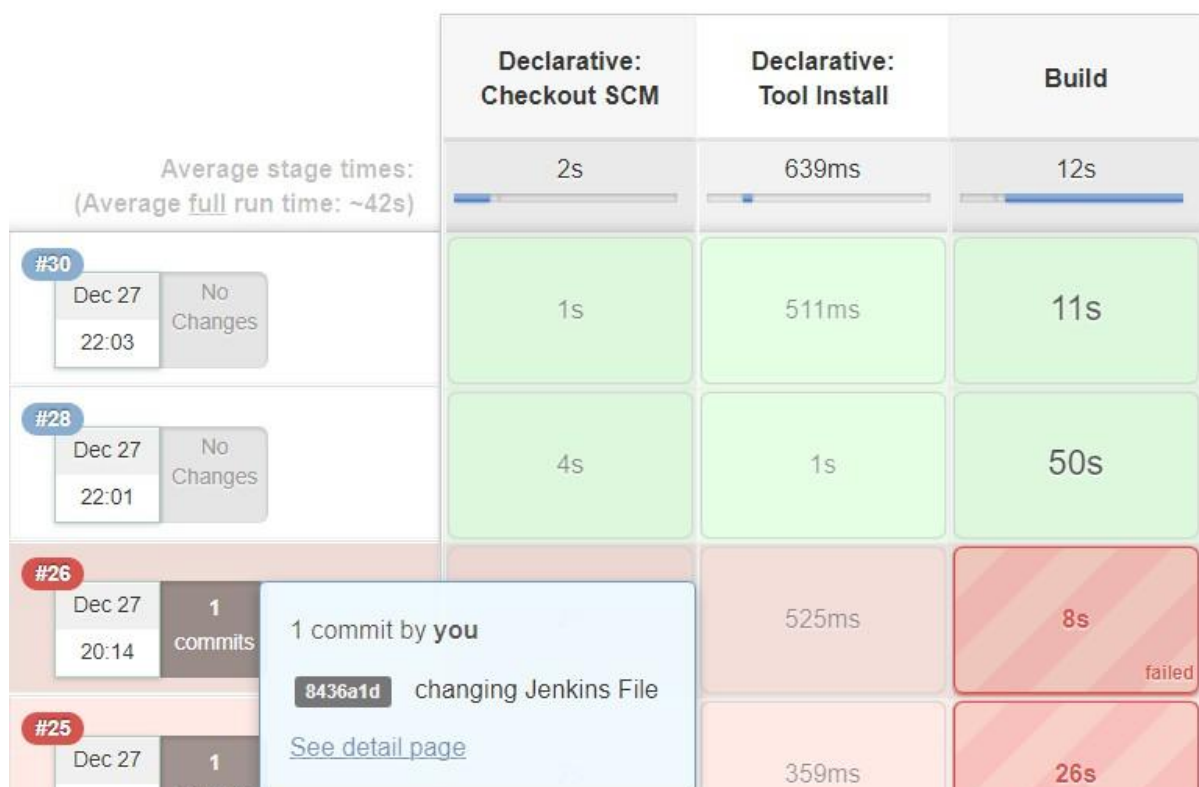
Definition

SCM

📁 Mentionner à Jenkins l'endroit du script à exécuter.

Script Path

Exemple de workflow simple avec détection de quel commit a engendré l'erreur et quel membre de l'équipe.



## Travail à rendre

- 📁 Concevoir un workflow dans le serveur d'Intégration Continue Jenkins qui va dans un premier temps lancer une tâche de compilation du code source du projet.
- 📁 Ensuite, le serveur lancera une série de tests chacun dans un nœud :
  - Nœud pour les tests unitaires.
  - Nœud pour la couverture de code.
  - Nœud pour la génération de la documentation

Astuce : revoir l'atelier maven

- 🖥 Dans la même étape, le serveur doit générer un rapport de tests et le publier dans un site web en invoquant la cible mvn site.
- 🖥 Dans la troisième étape, Jenkins doit emballer le projet dans un .jar ou .war ou autre selon le type d'application à déployer.
- 🖥 Si tout se passe bien, il faut planifier un déploiement du projet dans le dépôt (vous pouvez utiliser nexus, artifactory, ou autre).
- 🖥 S'il y'a une erreur dans une des étapes de build, envoyer une notification par mail à l'administrateur.

**Astuce: pour chaque étape, penser à ajouter le plugin nécessaire au « POM.xml » et aussi à Jenkins pour afficher les résultats dans la page d'accueil de votre tâche de build. Le tester en local avec une commande mvn avant de le lancer dans le processus d'intégration continue.**

Voici des exemples de plugins et goals à appeler

**Build : mvn compile**

**Test :**

- Couverture de code : **mvn cobertura:cobertura** (Plugin : cobertura-maven-plugin)
- Tests Unitaires : **mvn test**
- Tests de Performance : **mvn gatling:test** (Plugin : gatling-maven-plugin)
  - Analyse de code:
- CheckStyle: **mvn checkstyle:checkstyle** (Plugin : maven-checkstyle-plugin)
- FindBugs: **mvn findbugs:findbugs**
- PMD: mvn pmd:pmd (Plugin : maven-pmd-plugin)
  - JavaDoc: **mvn site**
  - Packaging: **mvn package**
  - Archivage Nexus: **mvn deploy** ou avec le plugin Nexus de Jenkins. (DistributionManagement)

Voici à quoi doit ressembler votre première pipeline jenkins:

## Stage View



Figure 1: Exemple de Pipeline

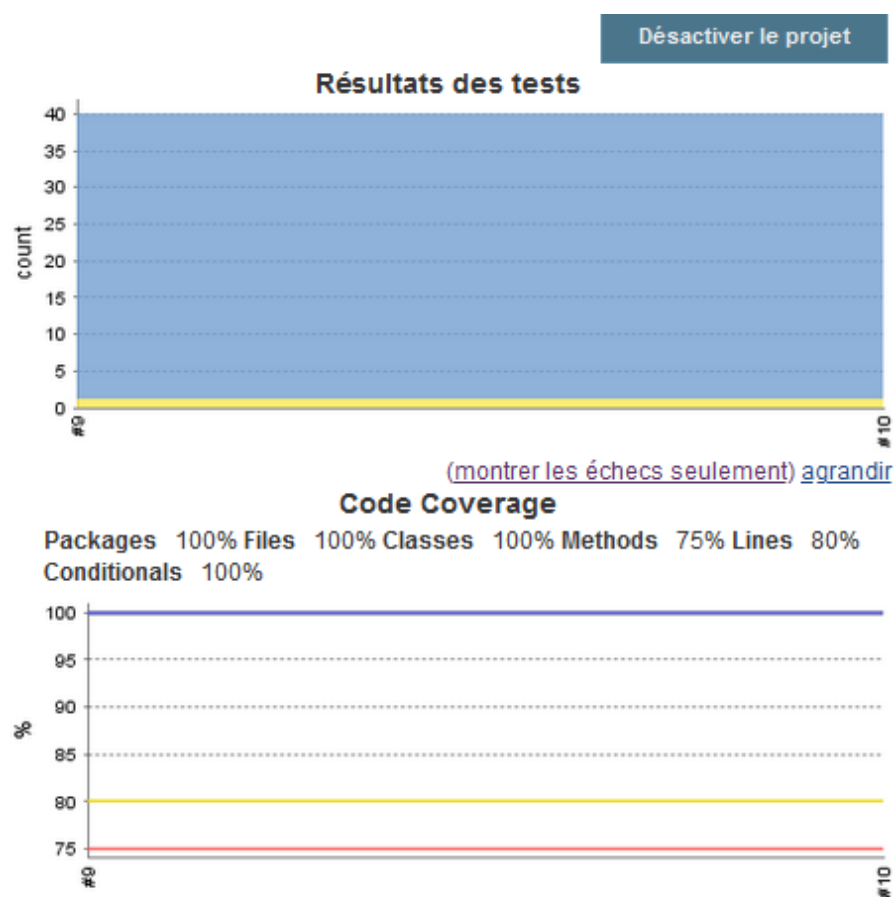


Figure 2: publication des résultats des test dans le serveur d'IC