Intégration Continue : Utilisation de Jenkins – Nexus – Sonar

Cédric Joffroy *Fabrice Ambert*

Ecole Polytechnique

5 Septembre 2013

Département d'Informatique des Systèmes Complexes

Plan

Plan

Objectif de l'atelier

Découvrir des outils

- Jenkins : création de jobs pour le *Build* continu
- Nexus : mise à disposition de librairies `Java`
- Sonar : réalisation de métriques sur le code
- Maven : simplification de la création et de la gestion des dépendances

Découvrir ce qu'est l'intégration continue

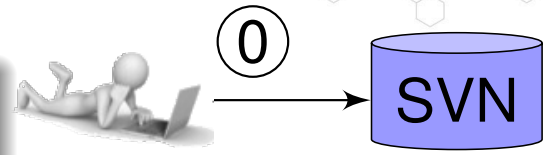
- Scrutation des dépôts (SVN, Git...) pour la construction automatique des projets
- Construction de projets en cascades
- Déploiement automatisé
- ...

Plan

Processus de l'intégration continue

Processus

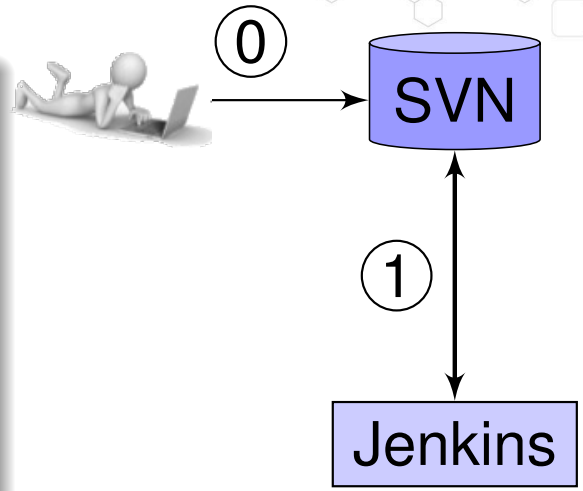
- ⓪ *Commit* des sources



Processus de l'intégration continue

Processus

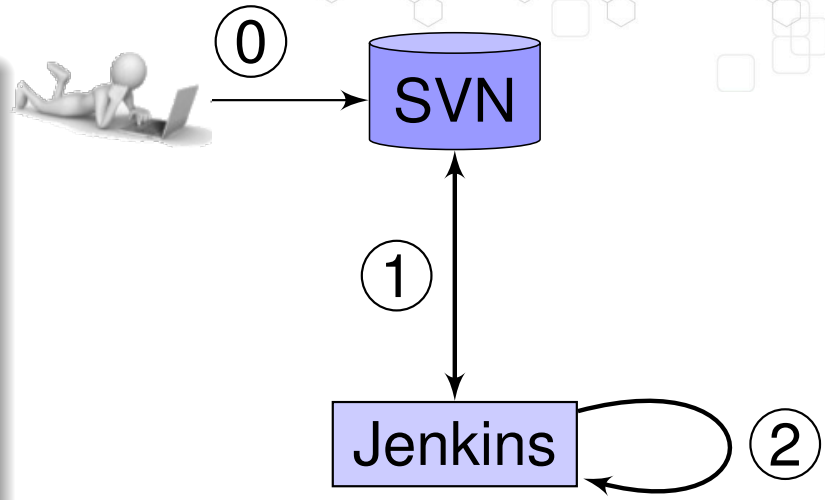
- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources



Processus de l'intégration continue

Processus

- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources
- ② Lancement du Job Jenkins



Processus de l'intégration continue

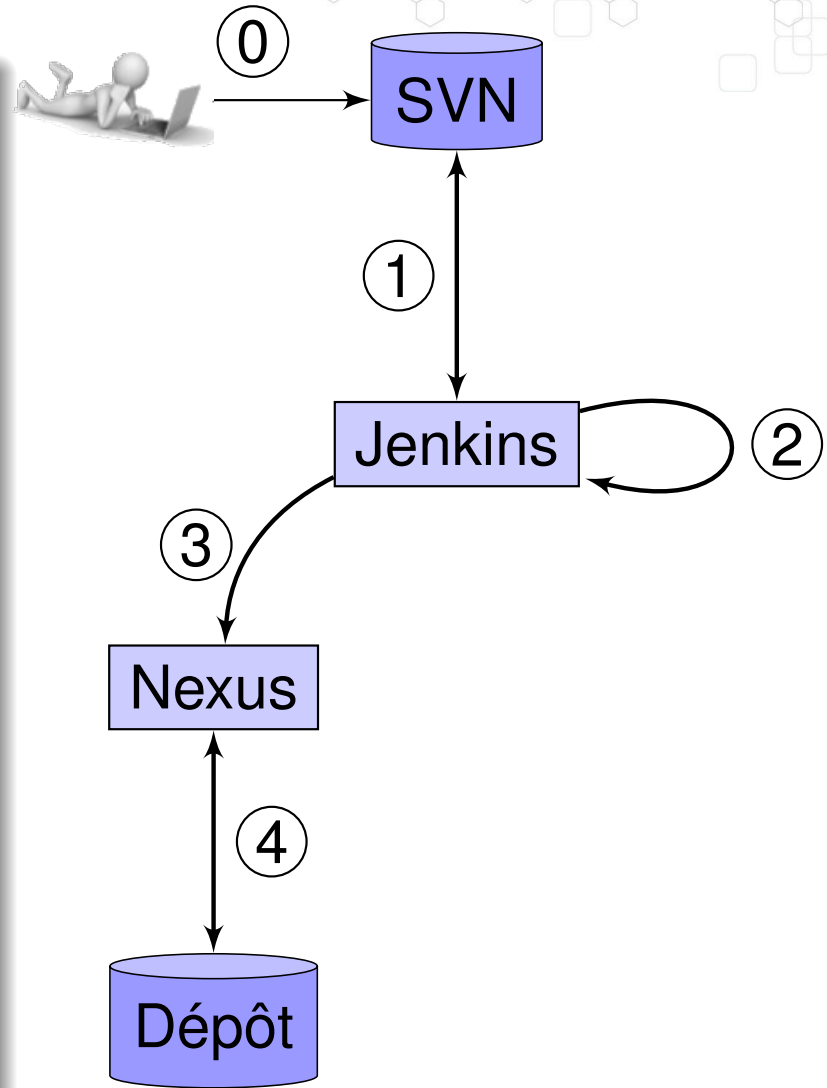
Processus

- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources
- ② Lancement du Job Jenkins
- ③ Demande des librairies à Nexus

Processus de l'intégration continue

Processus

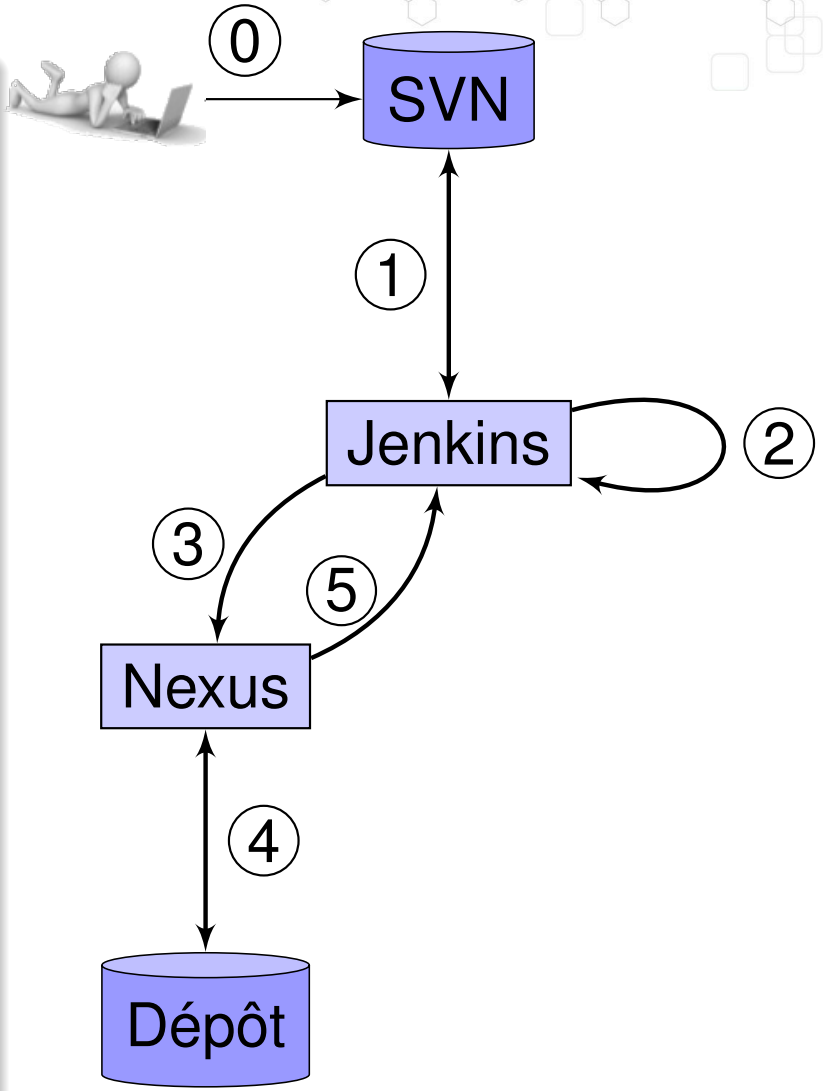
- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources
- ② Lancement du Job Jenkins
- ③ Demande des librairies à Nexus
- ④ Récupération des librairies dans le dépôt



Processus de l'intégration continue

Processus

- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources
- ② Lancement du Job Jenkins
- ③ Demande des librairies à Nexus
- ④ Récupération des librairies dans le dépôt
- ⑤ Envoi des librairies à Jenkins puis construction



Processus de l'intégration continue

Processus

- ⓪ *Commit* des sources
- ① Récupération des sources/Envoi des sources
- ② Lancement du Job Jenkins
- ③ Demande des librairies à Nexus
- ④ Récupération des librairies dans le dépôt
- ⑤ Envoi des librairies à Jenkins puis construction
- ⑥ Envoi des métriques
- ⑦ Publications des librairies

Plan

Plan

Jenkins

- Gestion de différents types de projets
  - Java : Maven, Ant
  - Avec l'ajout de plugins : C++, PHP, . . .
- Gestion des utilisateurs par projet
- Vision d'ensemble des projets en construction continue
- Déclenchement de constructions en cascade

Plan

Nexus – Maven

Nexus

- Gère les librairies (`Java`)
  - Standard intégrées aux dépôts `Maven` officiel
  - Tierces ajoutées et mises à disposition des développeurs
  - Projets déployées en fonction des projets des développeurs
- Gère les accès aux librairies/dépôts
  - Cloisonner les projets et accès aux ressources associées
  - Spécifier qui peut déployer des nouvelles librairies

Maven

- Permet la construction de projets (`Java`)
- S'appuie sur la définition d'un fichier *pom.xml*

Plan

Sonar

- Métriques sur un projet
  - Nombre de lignes de code
  - Pourcentage de code documenter
  - Pourcentage de code dupliquer
- Problèmes dans le code
  - Selon 5 critères : bloquant, critique, majeur, mineur, informatif
  - Possibilité de créer des tickets directement (si plugin installé)
- Tests unitaires
  - Pourcentage de couverture
  - Pourcentage des tests réussis

Plan

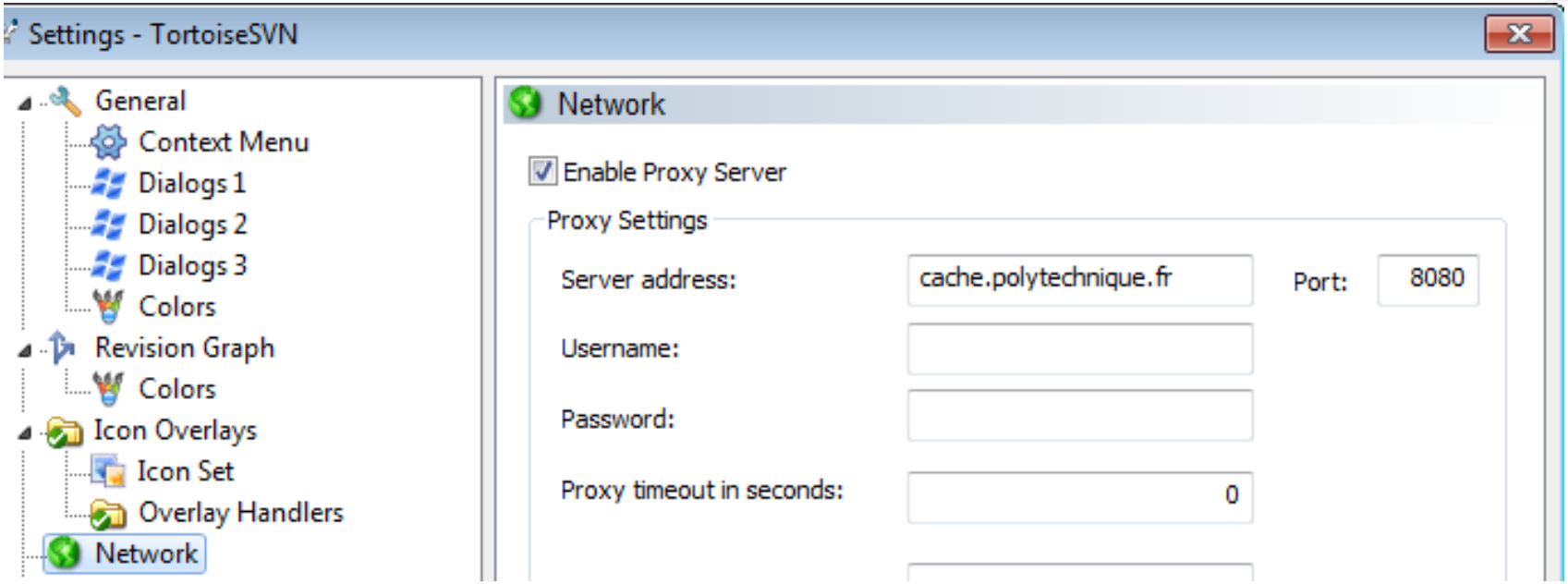
Exercice

- Réalisation d'une calculatrice
  - Prise en main du code SVN
  - Création du "job" `Jenkins`
  - Ajout d'une nouvelle fonction au sein du code
- Réalisation d'une librairie spécifique (par groupe)
  - Ajout au sein du SVN
  - Création du "job" `Jenkins`
  - Utilisation de cette librairie au sein du projet Calculatrice (+ dépendance `Jenkins`)
- Utilisation des librairies faites par les autres groupes
- Création des versions Release des librairies

Merci pour votre attention

Avez-vous des questions ?

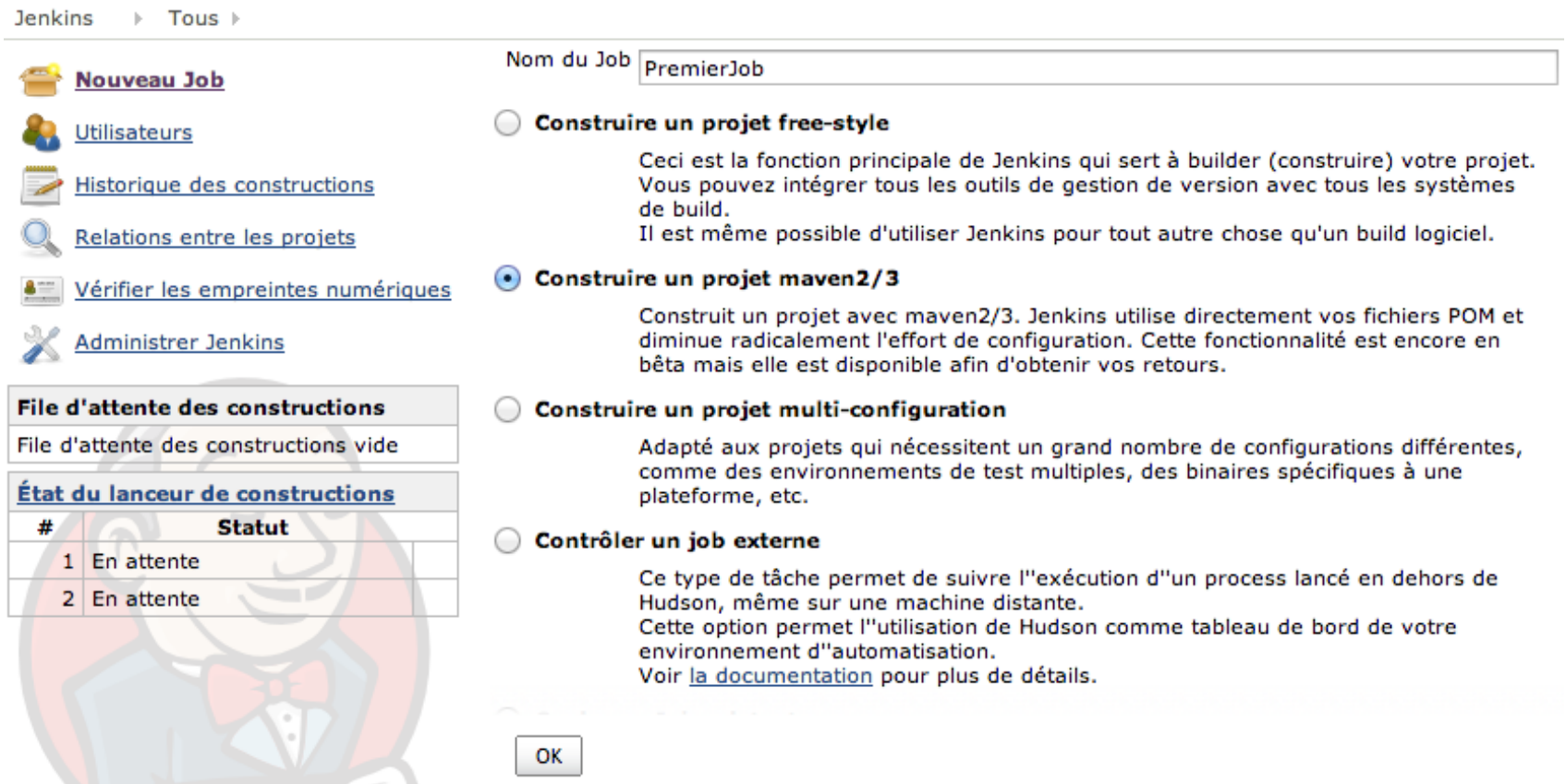
Configuration du Proxy pour SVN



Mac/Linux

- Modifier le fichier ~/.subversion/servers
- Ajouter les lignes suivantes :
  - http-proxy-host = cache.polytechnique.fr
  - http-proxy-port = 8080

Jenkins – Création d'un job (1/3)



Jenkins – Création d'un job (2/3)

Informations générales

- Fournir nom et une description
- Spécifier quand les builds doivent être supprimés
- Spécifier qui a accès au jobs (sécurité)

Gestion du code source

- Spécifier quel est le système de *versioning*
- Spécifier l'URL du dépôt

Ce qui déclenche le build

- Scruter les modification du dépôt
- Construire à la suite d'un autre projet (projet amont)
- Construire périodiquement

Jenkins – Création d'un job (3/3)

Build

- Spécifier le POM racine (cas d'un job `Maven`). Par défaut : "pom.xml"
- Spécifier les objectifs et options du *build* :
  - clean : pour supprimer la construction précédente
  - package : pour créer les paquets
  - verify : pour contrôler la construction

Autres

- Gestion des *Release* `Maven`
- Actions à la suite du *build* :
  - Déclencher d'autres projets
  - Sonar

Exemple d'un fichier *pom.xml*

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.femtost.disc.jdev2013</groupId>
  <artifactId>ToyProject</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>JDevTest</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <distributionManagement>
        <repository>
            <id>nexus-disc</id>
            <url>http://194.57.136.189:8082/nexus/content/repositories/releases/</url>
        </repository>
        <snapshotRepository>
            <id>nexus-disc</id>
            <url>http://194.57.136.189:8082/nexus/content/repositories/snapshots/</url>
        </snapshotRepository>
    </distributionManagement>
</project>
```