



Intégration Continue : Utilisation de Jenkins – Nexus – Sonar

**JDEV2013**

Cédric Joffroy Fabrice Ambert

Ecole Polytechnique

Inria

5 Septembre 2013



Département d'Informatique des Systèmes Complexes



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés
 - Jenkins
 - Nexus – Maven
 - Sonar
- 4 Mise en application



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés
 - Jenkins
 - Nexus – Maven
 - Sonar
- 4 Mise en application

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
- ▶ . . .

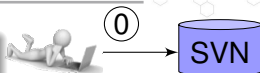


- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés
 - Jenkins
 - Nexus – Maven
 - Sonar
- 4 Mise en application

Processus de l'intégration continue

Processus

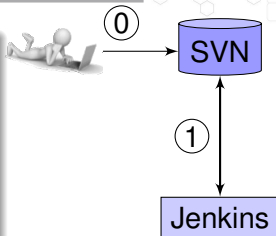
- ▶ ① *Commit* des sources



Processus de l'intégration continue

Processus

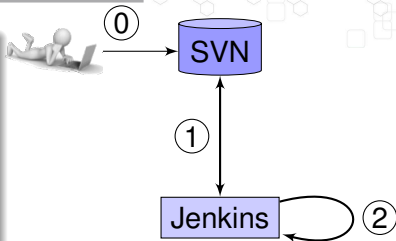
- ▶ ① *Commit* des sources
- ▶ ② Récupération des sources/Envoi des sources



Processus de l'intégration continue

Processus

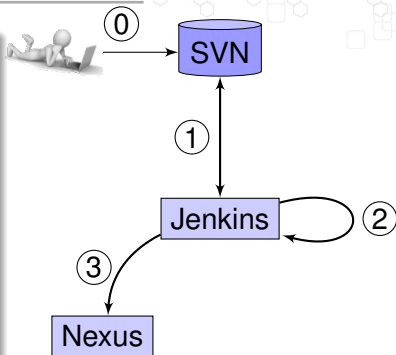
- ▶ ① *Commit* des sources
- ▶ ① Récupération des sources/Envoi des sources
- ▶ ② Lancement du Job Jenkins



Processus de l'intégration continue

Processus

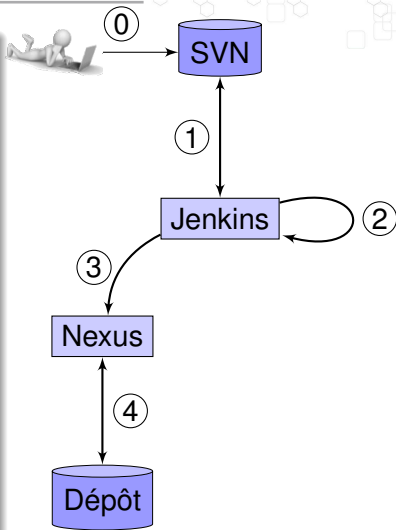
- ▶ ① *Commit* des sources
- ▶ ① Récupération des sources/Envoi des sources
- ▶ ② Lancement du Job Jenkins
- ▶ ③ Demande des librairies à Nexus



Processus de l'intégration continue

Processus

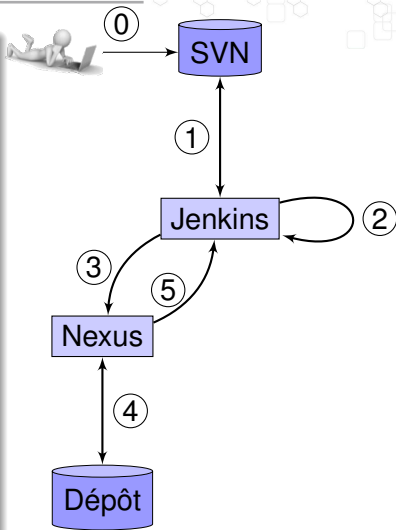
- ▶ ① *Commit* des sources
- ▶ ② Récupération des sources/Envoi des sources
- ▶ ③ Lancement du Job Jenkins
- ▶ ④ Demande des librairies à Nexus
- ▶ ⑤ Récupération des librairies dans le dépôt



Processus de l'intégration continue

Processus

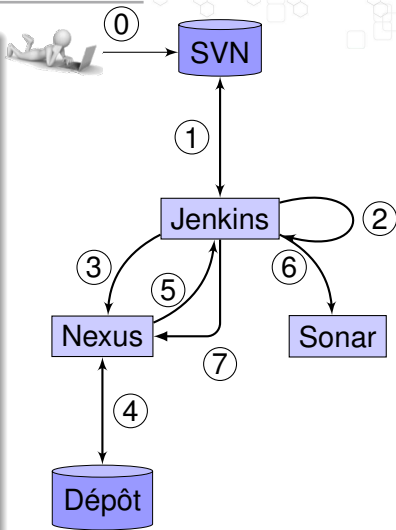
- ▶ ① *Commit* des sources
- ▶ ② Récupération des sources/Envoi des sources
- ▶ ③ Lancement du Job Jenkins
- ▶ ④ Demande des librairies à Nexus
- ▶ ⑤ Récupération des librairies dans le dépôt
- ▶ ⑥ Envoi des librairies à Jenkins puis construction



Processus de l'intégration continue

Processus

- ▶ ① *Commit* des sources
- ▶ ② Récupération des sources/Envoi des sources
- ▶ ③ Lancement du Job Jenkins
- ▶ ④ Demande des librairies à Nexus
- ▶ ⑤ Récupération des librairies dans le dépôt
- ▶ ⑥ Envoi des librairies à Jenkins puis construction
- ▶ ⑦ Envoi des métriques
- ▶ ⑧ Publications des librairies





- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés**
 - Jenkins
 - Nexus – Maven
 - Sonar
- 4 Mise en application



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés**
 - **Jenkins**
 - Nexus – Maven
 - Sonar
- 4 Mise en application



- ▶ 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



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés**
 - Jenkins
 - **Nexus – Maven**
 - Sonar
- 4 Mise en application

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*



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés**
 - Jenkins
 - Nexus – Maven
 - **Sonar**
- 4 Mise en application



- ▶ 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



- 1 Objectif de l'atelier
- 2 Vue d'ensemble du processus
- 3 Les outils utilisés
 - Jenkins
 - Nexus – Maven
 - Sonar
- 4 Mise en application

Exercice



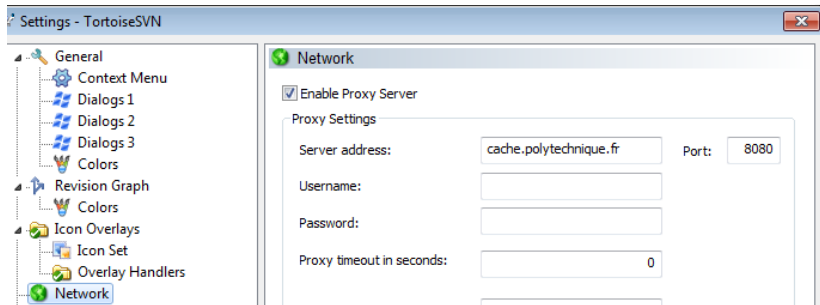
- ▶ Réalisation d'une calculatrice
 - ▶ Prise en main du code SVN
 - ▶ Création du "job" Jenkins
 - ▶ Ajout d'une nouvelle fonction au sein du code
- ▶ Réalisation d'une librairie spécifique (par groupe)
 - ▶ Ajout au sein du SVN
 - ▶ Création du "job" Jenkins
 - ▶ Utilisation de cette librairie au sein 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 > Tous >

[Nouveau Job](#)[Utilisateurs](#)[Historique des constructions](#)[Relations entre les projets](#)[Vérifier les empreintes numériques](#)[Administrer Jenkins](#)

File d'attente des constructions

File d'attente des constructions vide

État du lanceur de constructions

#	Statut
1	En attente
2	En attente

Nom du Job Construire un projet free-style

Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build.

Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.

 Construire un projet maven2/3

Construit un projet avec maven2/3. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.

 Construire un projet multi-configuration

Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.

 Contrôler un job externe

Ce type de tâche permet de suivre l'exécution d'un process lancé en dehors de Hudson, même sur une machine distante.

Cette option permet l'utilisation de Hudson comme tableau de bord de votre environnement d'automatisation.

Voir [la documentation](#) pour plus de détails.

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.femto.st.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>

```