Atelier T6.A2 : Industrialisation des tests – Intégration continue et automatisation des tests

Cédric Joffroy
cedric.joffroy@femto-st.fr

Fabrice Ambert
fabrice.ambert@femto-st.fr

05 Septembre 2013

Objectif de l'atelier

L'objectif de l'atelier est la découverte de l'intégration continue au-travers des outils suivants :

- `Jenkins`,
- `Maven`,
- `Nexus`
- et `Sonar`.

A partir d'un code existant présent au sein de votre dépôt, vous allez tout d'abord l'ajouter dans le *Build* continu de `Jenkins`, puis le compléter avec de nouvelles fonctionnalités. Le code de base fourni correspond à une calculatrice qui ne sait faire que les opérations basiques : addition, soustraction, division et multiplication.

Ensuite, vous allez réaliser une librairie qui correspondra à une fonction mathématique plus avancée (trigonométrique, géométrique, Fibonacci...). Vous utiliserez cette librairie au sein de votre calculatrice.

Enfin, vous utiliserez les librairies réalisées par les autres groupes afin que toutes les calculatrices possèdent l'ensemble des fonctionnalités.

Pour finir, la dernière étape consistera à réaliser une *Release* de la librairie et d'utiliser ces *Releases* au sein de vos calculatrices.

1 Réalisation de la calculatrice V1

La réalisation de la calculatrice va se faire en plusieurs étapes :

1. Récupération des sources sur le dépôt SVN ;
2. Mis en place du *Job* sur `Jenkins` ;
3. Ajout de nouvelles fonctionnalités simples ;
4. Vérification que le *Job* fonctionne bien.

1.1 Récupération des sources

L'adresse du SVN est de la forme suivante : `http://194.57.136.189:8880/svn/repo_userINC/` (avec INC de 01 à 30) et le login/password : userINC/c1l@p1INC (ceux-ci vous seront attribués en début de séance).

Sur Linux/Mac, la récupération des sources se fait via la commande (au préalable, il faut installer SVN `sudo apt-get install subversion` sur Debian/Ubuntu) :

```
svn co http://194.57.136.189:8880/svn/repo_userINC/ --username userINC
```

Sur Windows, il faut passer par TortoiseSVN (ou tout autre outils de gestion SVN) et récupérer les sources au travers de la même commande.

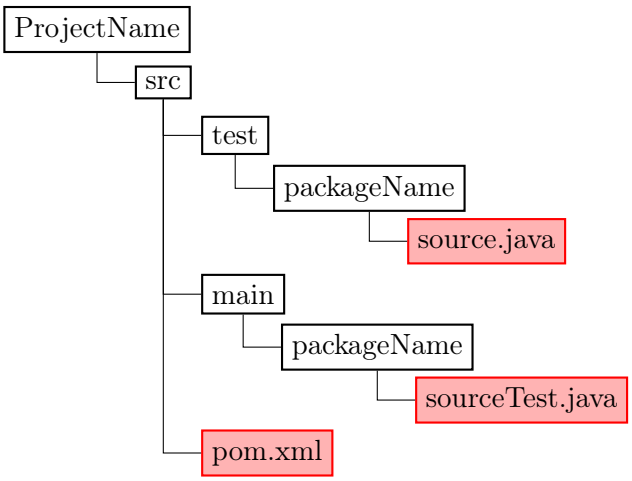

FIGURE 1 – Arborescence d'un projet type Maven

La Figure 1.1 reprend l'arborescence d'un projet type `Maven`. Les tests et les sources sont placés dans le répertoire`src` dans deux répertoires différents : `main` pour les sources et `test` pour les tests. Le nom des *package* est respecté dans les deux sous-arborescences.

Le fichier `pom.xml` qui se trouve dans le projet présent sur le SVN est de la forme suivante (il correspond à un fichier *pom* de base) :

```
<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.jdev2013.userINC</groupId>
 <artifactId>Calculator</artifactId>
 <version>1.0-SNAPSHOT</version>
 <packaging>jar</packaging>

 <name>Calculator</name>
```

```
<dependencies>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>3.8.1</version>
  <scope>test</scope>
 </dependency>
</dependencies>
<build>
 <plugins>
  <plugin>
   <artifactId>maven-compiler-plugin</artifactId>
   <version>2.3.2</version>
   <configuration>
    <source>1.6</source>
    <target>1.6</target>
   </configuration>
  </plugin>
 </plugins>
</build>
<distributionManagement>
 <repository>
  <id>nexus-jdevt6b</id>
  <url>http://192.168.20.3:8080/nexus/content/repositories/releases/</url>
 </repository>
 <snapshotRepository>
  <id>nexus-jdevt6b</id>
  <url>http://192.168.20.3:8080/nexus/content/repositories/snapshots/</url>
 </snapshotRepository>
</distributionManagement>
</project>
```

Listing 1 – Fichier pom.xml basic

Voici en détail ce qui est contenu dans ce fichier :

- *groupId* correspond au nom du package dans lequel se trouve le projet. C'est le paquet de plus haut niveau.
- *artifactId* correspond au nom du projet.
- *version* correspond au numéro de version. Il peut être sous la forme : 1.0-SNAPSHOT ou 1.0.0-SNAPSHOT... pour les versions *Snapshots* et 1.0 ou 1.0.0... pour les versions *Releases*
- *packaging* correspond au type de packaging que l'on souhaite. Ici, *jar*, mais l'on peut également avoir *pom*...
- *dependencies* permet de gérer l'ensemble des dépendances du projet. Il permet ainsi de spécifier quelle est la version de la librairies que l'on souhaite utiliser et à quelle moment (*scope*). Ces librairies sont récupérées via `Nexus`.
- *distributionManagement* permet de spécifier l'URL du serveur `Nexus` (dans notre cas) pour permettre le déploiement du projet (sous la forme ici d'un fichier *jar*). Celui-ci sera alors disponible comme une librairie réutilisable dans d'autres projets.

Il existe de nombreuses autres balises par exemple :
- *parent* pour spécifier un *pom* parent (pour hériter des propriétés) ;
- *module* pour spécifier des *pom* enfants ;
- *build* permet de spécifier des resources ou des plugins à utiliser durant la construction ;
- *profile* pour créer des profiles en fonction d'un environnement ou autre et qui s'exécute dans un cas spécifique.

Pour plus de détails, la documentation sur le fichier `pom.xml` se trouve à l'URL suivante : `http://maven.apache.org/pom.html`.

1.2 Création du Job sur Jenkins

Pour créer le *Job* sur `Jenkins`, il faut se rendre à l'adresse suivante : `http://194.57.136.189:8503/`. Une fois sur la page, il faut se connecter avec les mêmes user/password que pour le SVN. Vous arrivez ensuite sur un tableau de bord où vous pouvez voir l'ensemble des *Jobs* déjà présents.

Pour créer un nouveau *Job*, il suffit de cliquer sur : **Nouveau job**. Il faut ensuite donner un nom de projet sous la forme : UserINC - Calculatrice (par exemple). On choisit ensuite : **Construire un projet maven2/3**. On arrive alors sur une page complète pour spécifier les différentes caractéristiques du *Job*.

Au minimum, vous devez spécifier :
- **Gestion du code source** en choisissant Subversion et en spécifiant l'URL complète où se trouve le projet (**ATTENTION : l'url à mettre ici n'est pas la même que celle du SVN, la voici :** `http://192.168.20.2:8880/svn/repo_userINC/trunk/Calculatrice`) ;
- **Ce qui déclenche le build** en spécifiant une Scrutation de l'outil de gestion de version (basé sur *cron*, donc pour vérifier toutes les 5 minutes : H/5 * * * *)
- **Build** en spécifiant les *Goals et options*. Pour nous : *clean package verify*

On ajoutera également une action post-build pour ajouter l'option : *Sonar*. Vous pourrez le faire sur ce projet ou le suivant à réaliser (ou les deux).

Ensuite, vous n'avez plus qu'à sauver et le *Job* est créé. Il ne reste plus qu'à exécuter celui-ci pour vérifier que tout fonctionne bien.

1.3 Modification du code source existant

Maintenant que le *build* continu est mis en place, le but est de modifier le code source pour ajouter quelques fonctionnalités supplémentaires. Quelques idées de fonctionnalités à créer :
- Le modulo
- La valeur absolue
- La puissance de 2

– ...

On se contente ici de créer des fonctionnalités simples. Les plus complexes seront faites dans la seconde partie de l'atelier.

Pour chaque modification apportée, ne pas oublier de compléter les tests.

Une fois que les modifications auront été *commitées* sur le SVN, la construction de votre projet se fera de manière automatique (si vous avez bien configurer la *Scrutation de l'outil de gestion de version*). Il ne vous reste plus qu'à contrôler que le *build* s'est bien passé.

2 Création d'une librairie de fonctions avancées

L'objectif est de créer un nouveau projet que l'on utilisera comme librairie dans le premier projet (la calculatrice). Ce projet contiendra une fonction mathématique avancée (à se répartir à niveau d'une par personne/binôme). Ces fonctionnalités sont les suivantes :

- Série de Taylor
  - Cosinus : $cos(x) = \sum_{n=1}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \ \forall x$
    - $cos(0) = 1$
    - $cos(1) = 0.54030230586$
  - Sinus : $sin(x) = \sum_{n=1}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \ \forall x$
    - $sin(0) = 0$
    - $sin(1) = 0.8414709848$
  - Arcsinus : $arcsin(x) = \sum_{n=0^{\infty}} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}$ avec $|x| \leq 1$
    - $arcsin(0) = 0$
    - $arcsin(1) = 1.57079633$
  - Arccosinus : $arccos(x) = \frac{\pi}{2} - arcsin(x) = \frac{\pi}{2} - \sum_{n=0^{\infty}} \frac{(2n)!}{4^n (n!)^2 (2n+1)} x^{2n+1}$ avec $|x| \leq 1$
    - $arccos(0) = 1.57079633$
    - $arccos(1) = 0$
  - Sinus hyperbolique : $sinh(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$ avec
    - $sinh(0) = 0$
    - $sinh(1) = 1.17520119364$
  - Cosinus hyperbolique : $cosh(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$ avec
    - $cosh(0) = 1$
    - $cosh(1) = 1.54308063482$
- Factorielle

- Equation du second ordre (dans les réels et/ou dans les complexes) ($ax^2 + bx + c = 0$)
  - $\Delta = b^2 - 4ac$
  - Si $\Delta > 0$, 2 racines
    - $R_1 = \dfrac{-b + \sqrt{\Delta}}{2a}$
    - $R_2 = \dfrac{-b - \sqrt{\Delta}}{2a}$
  - Si $\Delta = 0$, une racine : $R = -\dfrac{b}{2a}$
  - Si $\Delta < 0$, pas de racines dans les réels
- Combinatoire et arrangement : $C_n^p = \dfrac{n!}{p!(n-p)!}$ et $A_n^p = \dfrac{n!}{(n-p)!}$
- Fibonacci : $F_0 = 0, F_1 = F_2 = 1, F_{n+2} = F_{n+1} + F_n$
- Calcul de volume
  - Sphère : $V = \dfrac{4\pi R^3}{3}$
  - Cylindre : $V = \pi R^2 h$
- Calcul des coefficients binomiaux
  - Triangle de Pascal
  - Directement les coefficients binomiaux pour un degré fixé

Attention : ne pas choisir, dans certains cas, des valeurs trop élevées sous peine de dépasser la taille d'encodage et d'avoir des résultats faussés.

2.1 Création du nouveau projet sur Netbeans

Pour créer un nouveau projet `Meven` sur `Netbeans`, on passe par l'assistant de création de projet. On sélectionne ensuite : *Maven* → *Java Application* (cf. Figure 2). La Figure 3 illustre ce qui doit être mis dans les différents champs de création du projet.

Une fois le projet créé, celui-ci devrait suivre l'arborescence décrit dans la Figure 1.1. Deux fichiers `Java` sont présents : `App.java` et `AppTest.java` (respectivement sous l'arborescence de *main* et de *test*).

Il ne reste plus qu'à compléter le code source pour créer la fonctionnalité et la tester et ainsi obtenir la librairie.

Avant de créer le *Job* sur `Jenkins`, il reste deux choses à faire :

1. Modifier le fichier `pom.xml` afin d'ajouter la section *distributionManagement* comme dans l'exemple donné au début.
2. Ajouter au sein du fichier `pom.xml` les informations pour que `Jenkins` et `Maven` puissent gérer la *Release* :

```
<scm>
 <connection>scm:svn:http://192.168.20.2:8880/svn/repo_userINC/
 trunk/Library/Directory</connection>
 <developerConnection>scm:svn:http://192.168.20.2:8880/svn/repo_userINC/
 trunk/Library/Directory</developerConnection>
 <url>http://192.168.20.2:8880/svn/repo_userINC/trunk/
```

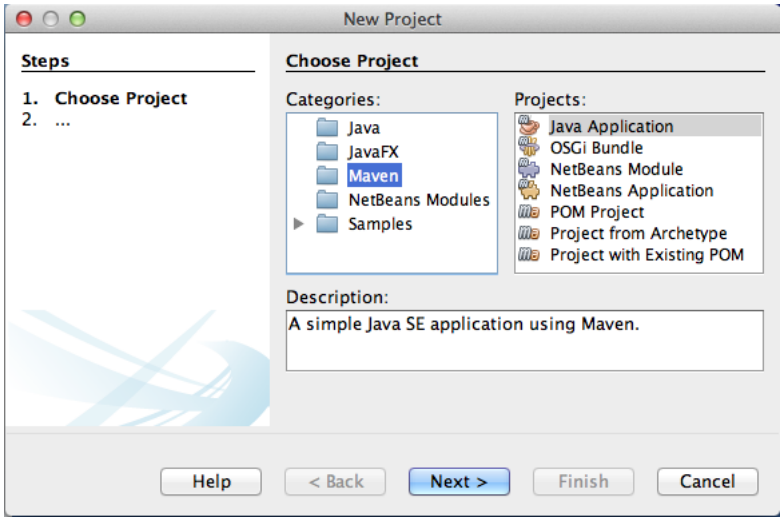

FIGURE 2 – Création d'un nouveau projet sur Netbeans (1/2)

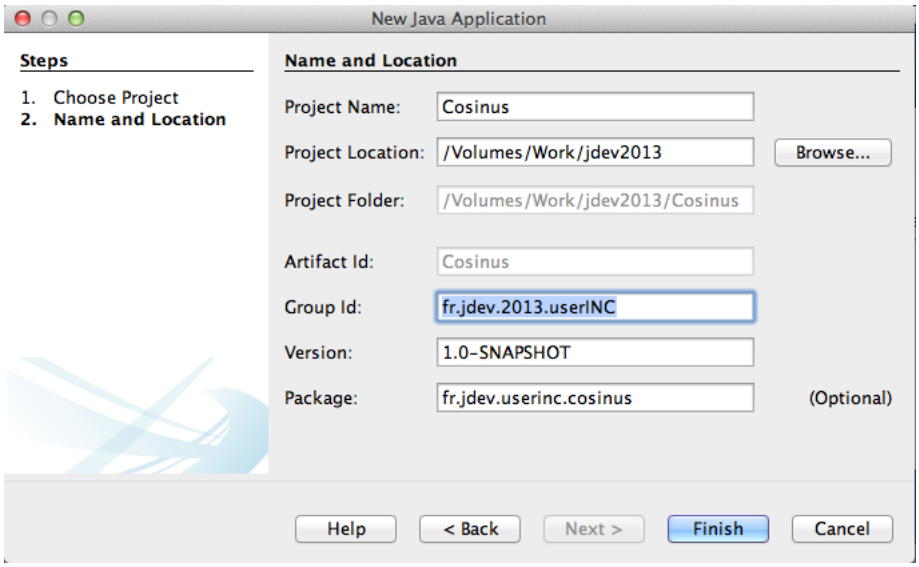

FIGURE 3 – Création d'un nouveau projet sur Netbeans (2/2)

```
Library/Directory</url>
</scm>
```

Listing 2 – A ajouter au fichier pom.xml

3. Ajouter et commiter l'ensemble du projet dans votre dépôt SVN.

2.2 Création du *Job* sur Jenkins

La création du *Job* se fait comme précédemment. La différence va se faire au niveau des *Actions à la suite du build*. On va ajouter l'action suivante : *Déployer les artefacts dans le repository Maven*. Le fait d'ajouter cette option permet de spécifier que le projet doit être déployer au sein de `Nexus` ce qui le mettra ainsi à la disposition des autres utilisateurs.

> Dans le cas de projet où la confidentialité est de mise, il serait important de structurer le dépôt `Nexus` afin que tous les utilisateurs n'aient pas accès à tous les dépôts.

Une fois que le *build* s'est bien passé, la librairie est présente au sein de Nexus.

3 Calculatrice V2 : Réutilisation des librairies

L'objectif de cette partie est de réutiliser dans un premier temps la librairie que vous avez créée puis dans un second temps de réutiliser tout ou partie des librairies qui ont été créées.

3.1 Réutilisation de votre librairie

Pour se faire, il suffit d'aller sur le serveur `Nexus` afin de retrouver votre librairie : `http://194.57.136.189:8083/nexus/`. Celle-ci doit normalement se situer dans le répertoire *Snapshots*. Une fois trouvée, dans le répertoire *1.0-SNAPSHOT* (vu que l'on n'a pas encore fait de *Release*), vous allez voir l'ensemble des versions qui ont été créées suite aux différents *Build* exécuté par `Jenkins`. Il suffit ensuite de cliquer sur une des version qui finit par l'extension *.jar*. Une fenêtre s'ajoute pour donner l'ensemble des informations à ajouter dans le fichier *pom* au niveau des dépendances.

La Figure 4 illustre ce que l'on a lorsque l'on clique sur une version de la librairie que l'on souhaite utiliser. Il suffit alors de copier/coller dans la partie XML et de l'ajouter au fichier *pom* du premier projet au niveau des dépendances. Ainsi, vous serez en mesure de pouvoir utiliser ce qui se trouve dans la librairie.

Il ne reste plus qu'à compléter le code pour ajouter la nouvelle fonctionnalité et commiter les sources.

3.2 Modification du *Job* sur Jenkins

Comme la calculatrice dépend d'un autre projet, il va maintenant falloir créer une dépendance entre ces deux *Jobs* afin que lorsque la librairie est modifiée, le *Job* de la calculatrice soit lancé afin de contrôler que tout se passe

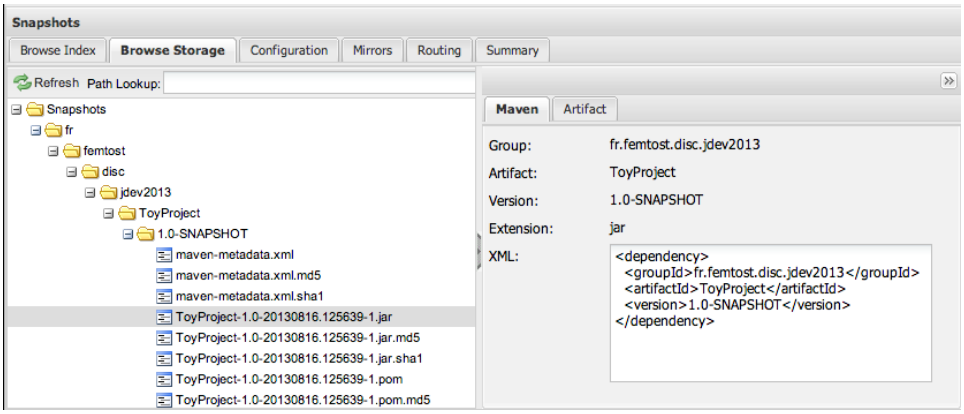

FIGURE 4 – Informations sur un artefact présent dans Nexus

toujours bien. Par contre, l'inverse n'est pas nécessaire puisque la calculatrice n'impacte en rien la librairie.

Ainsi, on va modifier le *Job* du projet de la calculatrice afin de compléter la partie : **Ce qui déclenche le job**. Il faut sélectionner : **Construire à la suite d'autres projets (projets en amont)** et taper le nom du *Job* correspondant à la librairie.

3.3 Réutilisation des autres librairies

Il suffit d'appliquer ce qui a été fait pour votre librairie, pour les librairies qui ont été développées par les autres participants.

De la même manière, il ne faut pas oublier de modifier le *Job* de la calculatrice.

4 Création d'une version Release

La création d'une version *Release* se fait également au travers de `Jenkins`. Pour se faire, il faut compléter le *Job* qui permet de faire le *build* continu de la librairie en ajoutant une option. La Figure 5 correspond à la case qu'il faut cocher au sein du *Job*.

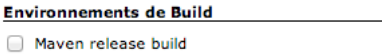

FIGURE 5 – Jenkins – Option Maven Release

Une fois cette case cochée, le menu latéral est modifié comme illustré par la Figure 6. En effet, à partir de maintenant, il est possible de lancer

directement la création d'une *Release*.

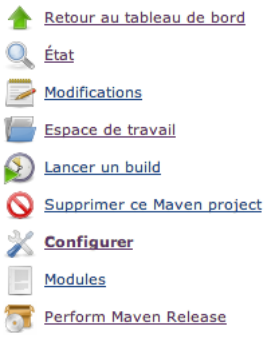

FIGURE 6 – Jenkins – Menu Maven Release

La création d'une *Release* se déroule en plusieurs étapes automatisées :

1. Création d'un tag avec une version stable (sans *Snapshot*)
2. Changement du version de *Snapshot*
3. Build et déploiement de la version stable
4. Build (et déploiement) de la version *Snapshot* (en vue de future modifications)

Lors ce que l'on souhaite réaliser une *Release*, un certain nombre d'informations est demandé. Ces informations sont déjà pré-remplies en fonction de la version actuelle. Il faut également cocher la case : *Specify SCM login/password* (où il faut mettre les identifiants permettant l'accès au SVN). La Figure 7 reprend ces informations. Il ne reste plus alors qu'à cliquer sur *Schedule Maven Release Build*.

Une fois le *Build* réalisé avec succès, il ne reste plus qu'à contrôler que le déploiement a bien eu lieu au sein de `Nexus`. Par la suite, il est possible de modifier le projet de base pour ne plus utiliser la version *Snapshot* mais utiliser la version *Release* de la librairie.

5 Visualisation des métriques sur Sonar

Maintenant que plusieurs *Jobs* ont été créés sur `Jenkins` et que l'on a ajouté en action post-build le fait de mettre les données dans `Sonar`, il est temps d'aller regarder ce que ces métriques donnent.

Pour se faire, il suffit de se rendre à l'adresse suivante : `http://194.57.136.189:8083/sonar/`. La Figure 8 correspond à l'écran d'accueil de `Sonar`. On retrouve ici (de base, sans application de sécurité particulière) l'ensemble des projets dont on a demandé les métriques `Sonar` depuis le *Job* `Jenkins`.

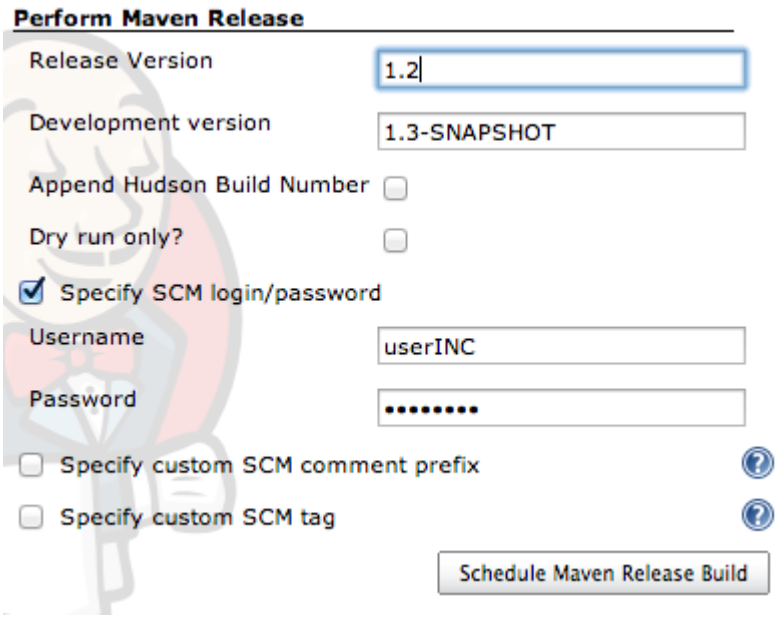

FIGURE 7 – Jenkins – Schedule Maven Release

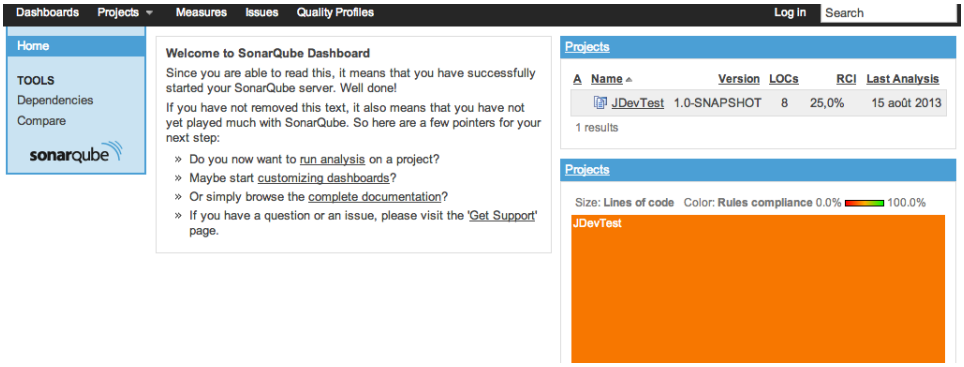

FIGURE 8 – Sonar – Ecran d'accueil

La Figure 10 correspond à un tableau de bord qui reprend l'ensemble des métriques et des informations pour un projet donné. Il est alors possible de rentrer dans le détail de ces métriques en cliquant sur une des données.

La Figure 9 reprend le détails des problèmes majeurs que l'on a dans le code.

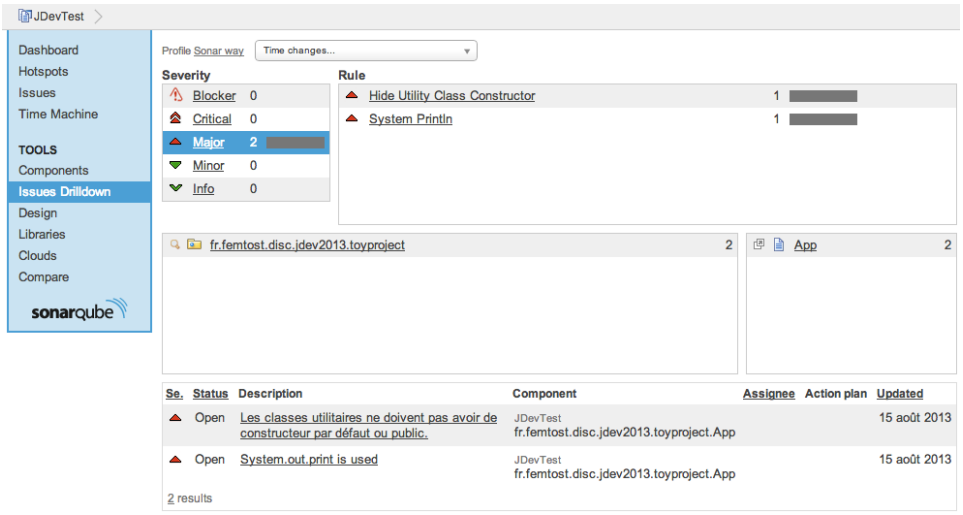

FIGURE 9 – Sonar – Détail sur un problème particulier dans le code

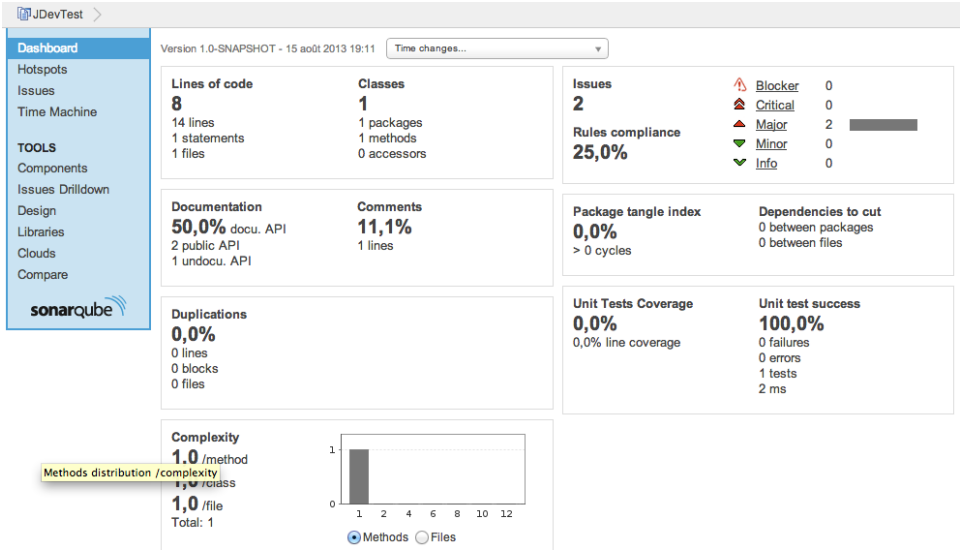

FIGURE 10 – Sonar – Vue de l'ensemble des métriques d'un projet

Conclusion

Ceci conclut cet atelier. Avec d'autres outils tels que `Jira`, il sera possible de lier l'ensemble des outils que l'on a vu (`Jenkins` et `Sonar`) de telle sorte de pouvoir créer des tickets pour les problèmes à résoudre. En effet, ces outils, pour l'intégration continue, ne sont qu'une des briques des outils d'aide au développement et à la gestion de projet.

Annexes

Installation de la partie cliente

Cette section détaille l'installation de `Maven` et de `Netbeans` ainsi que la configuration de ceux-ci.

Installation de Maven

Pour installer `Maven` :

– Sur Windows, télécharger la dernière version de `Maven` à l'adresse suivante : `http://maven.apache.org/download.cgi`. Une fois l'archive *zip* téléchargée, décompressée là dans le répertoire désiré (*e.g.* `C:\apache-maven`. Il faut ensuite compléter le *Path* dans les variables d'environnement. Pour cela, il faut aller dans : *Panneau de configuration* → *Système et Sécurité* → *Système*. A ce moment là, cliquer sur *Paramètres système avancés* (cf. Figure 11). Ensuite, il faut cliquer sur

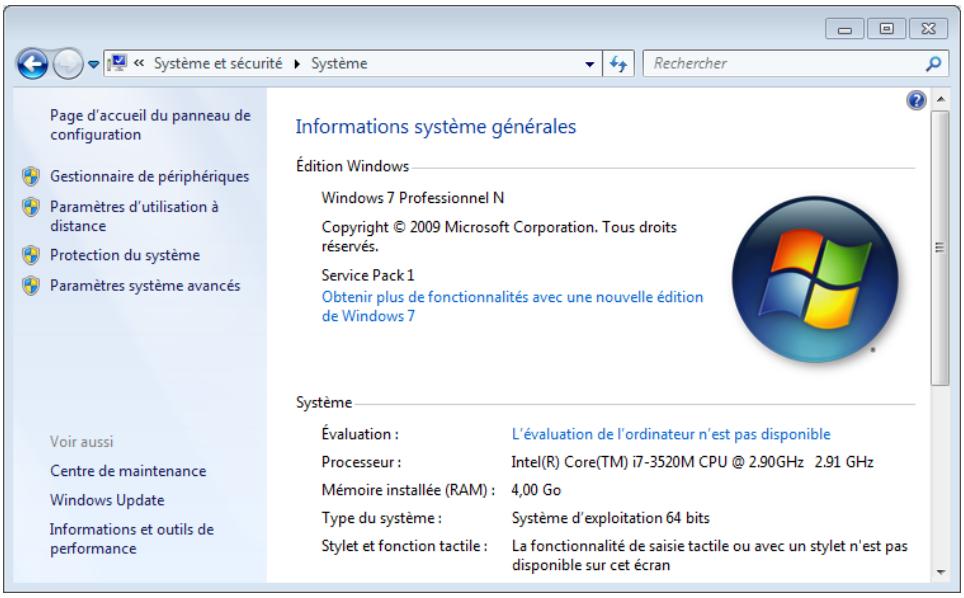

FIGURE 11 – Paramètres systèmes (Windows 7)

le bouton *Variables d'environnement...* (cf. Figure 12). Ajouter au niveau de la variable *Path*, le chemin jusqu'au répertoire *bin* inclus dans le répertoire `Maven`.

– Sur Linux (Debian/Ubuntu), il suffit de faire :
`sudo apt-get install maven`.
Le système se charge ensuite de configurer tout ce qu'il faut. Dans le cas d'autres distributions, il faut télécharger `Maven` et ne pas oublier

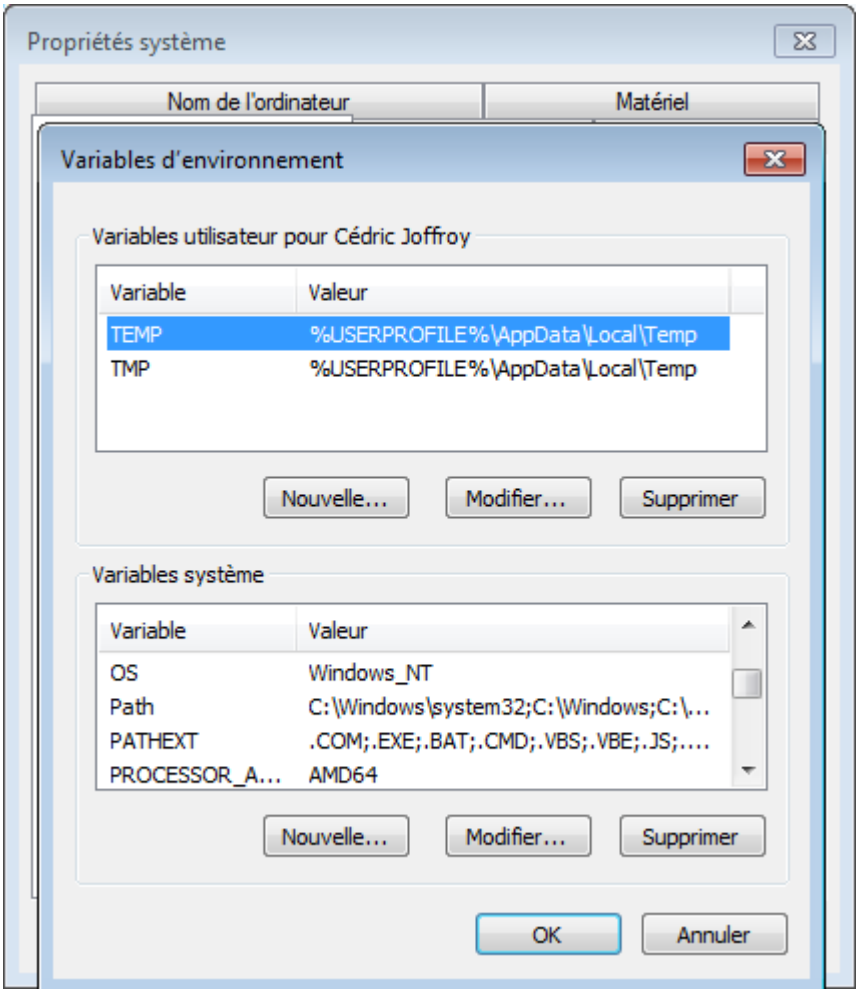

FIGURE 12 – Variables d'environnements (Windows 7)

de l'ajouter ensuite dans le *Path* pour que la commande *mvn* soit accessible (encore une fois, le répertoire *bin* doit être ajouté au *path*).

– Sur Mac, `Maven` est disponible de base.

Configuration de Maven

Pour configurer `Maven` afin que celui-ci utilise `Nexus`, il faut pour cela :

– Soit créer un fichier `settings.xml` dans le répertoire `.m2` qui se trouve dans votre répertoire personnel ;

– Soit modifier le fichier `settings.xml` si celui-ci est déjà présent.

Dans le premier cas, si vous ne possédez pas de répertoire `.m2`, il faut le créer :

– soit dans le répertoire `/home/logi` dans le cas de Linux ;

– soit dans le `C:\Users\Nom Complet` dans le cas de Windows ;
– soit dans le répertoire `/Users/login` dans le cas de Mac.

Une fois le répertoire créé, il faut ajouter le fichier `settings.xml`. Le Listing 20 propose une configuration permettant d'utiliser le `Nexus` installé pour les *JDev* et permet d'accéder à la fois aux librairies qui possèdent une version *Release* mais également aux librairies qui ne possèdent qu'une version *Snapshot*.

Attention : les identifiants doit être modifié pour correspondre à vos identifiants. Ils suivent la même logique que ce qui a été décrit au début. *INC* est un nombre sur deux chiffres allant de 01 à 30.

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
          http://maven.apache.org/xsd/settings-1.0.0.xsd">
 <servers>
  <server>
   <id>nexus-jdevt6b</id>
   <username>userINC</username>
   <password>c1l@p1INC</password>
  </server>
  <server>
   <id>nexus-jdevt6b-snapshots</id>
   <username>userINC</username>
   <password>c1l@p1INC</password>
  </server>
 </servers>
 <mirrors>
  <mirror>
   <id>nexus-jdevt6b</id>
   <mirrorOf>*</mirrorOf>
   <url>http://194.57.136.189:8083/nexus/content/groups/public/</url>
  </mirror>
 </mirrors>
 <profiles>
  <profile>
   <id>nexus-jdevt6b-snapshots</id>
   <activation><activeByDefault>true</activeByDefault></activation>
   <repositories>
    <repository>
     <id>nexus-jdevt6b-snapshots</id>
     <url>http://194.57.136.189:8083/nexus/content/groups/public/</url>
     <releases><enabled>true</enabled></releases>
     <snapshots><enabled>true</enabled></snapshots>
    </repository>
   </repositories>
  </profile>
 </profiles>
</settings>
```

Listing 3 – Fichier settings.xml pour la connexion à Nexus

Installation et configuration de Netbeans

Dans le cadre de l'atelier, nous allons utilisé Netbeans 7.3 (ou plus). Durant l'installation, il ne faut pas oublier de cocher le fait d'installer `JUnit` (cf. Figure 13).

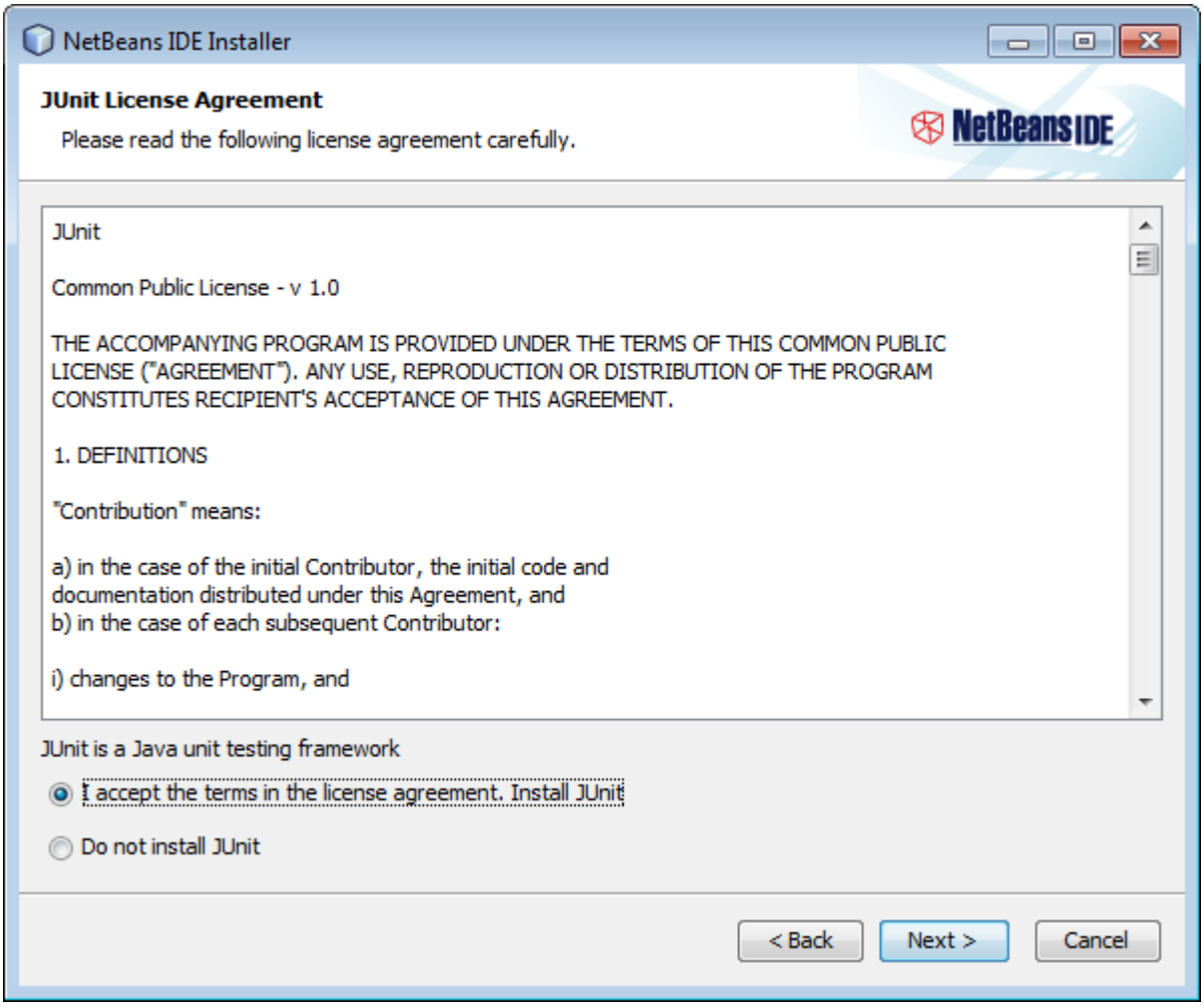

FIGURE 13 – Fenêtre de sélection de JUnit

Une fois Netbeans installé, il faut ensuite modifié la configuration afin qu'il utilise le `Maven` que vous avez installé précédemment. Pour se faire, il faut aller dans :

- Tools → Option (sur Windows et Linux)
- Netbeans → Préférences (sur Mac)

Ensuite, il faut aller dans l'onglet *Java* puis *Maven*. Enfin, il faut modifier la version de `Maven` utilisé en spécifiant ou sélectionnant la version

précédemment installée (cf. Figure 14). Une fois que c'est fait, votre Netbeans est configuré. Il ne reste plus qu'à créer un nouveau projet `Maven` afin de contrôler que tout fonctionne correctement.

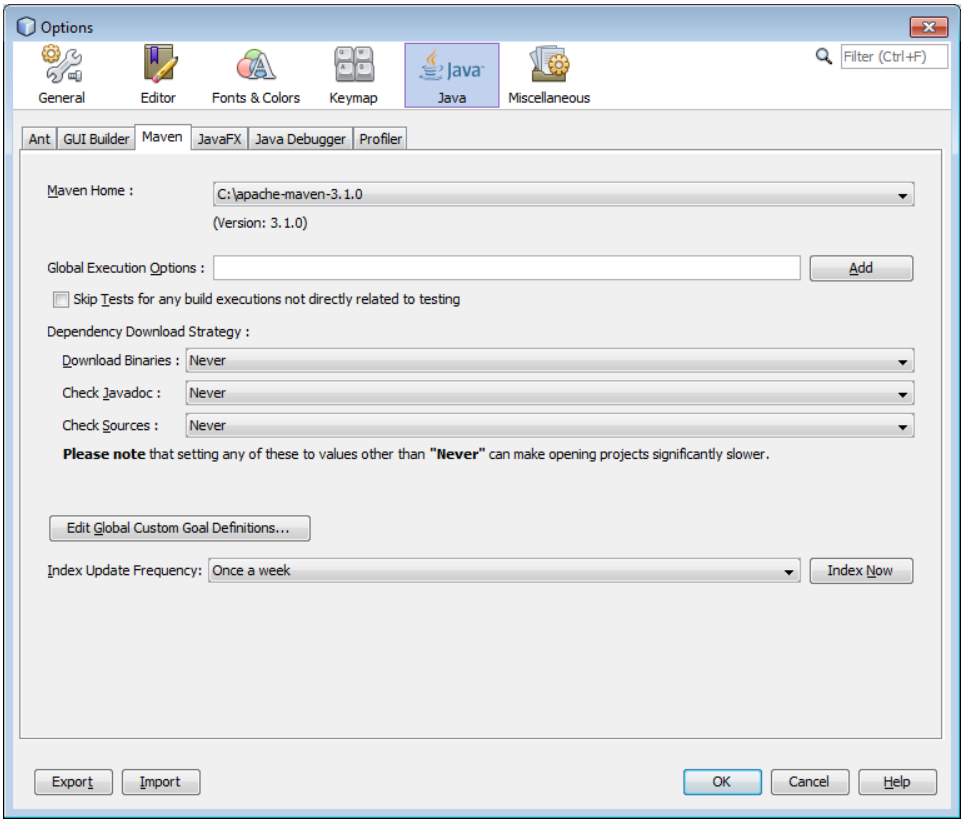

FIGURE 14 – Configuration de Maven dans Netbeans

Installation du serveur d'intégration continue

Le but est l'installation de l'ensemble de la suite d'outils pour l'intégration continue :

- Jenkins
- Nexus
- Maven
- Ant
- Sonar

L'installation a été réalisée sur une Debian (et sur une Ubuntu). La démarche présentée ci-après n'a pas été testée sur d'autres plateformes.

L'installation va être découpée en plusieurs étapes. La configuration des différents outils également

5.0.1 Installations préliminaires

Java 6 (ou autre) JDK et JRE Télécharger la version de java (JDK et JRE) désirée sur le site d'Oracle.

Ensuite, il suffit de le décompresser. On placera le répertoire obtenu dans /opt. Ensuite, et afin de permettre une montée de version de Java par la suite, on crée un lien symbolique sur le répertoire de la version que l'on vient d'installer :

```
cd /opt
sudo ln -s jdk1.6.0_45 current_jdk
sudo ln -s jre1.6.0_45 current_jre
```

Listing 4 – Installation de Java

Maven Pour installer `Maven`, on s'appuie sur les paquets mis à disposition dans la distribution. Pour se faire, il suffit de taper les commandes suivantes :

```
sudo apt-get install maven
```

Listing 5 – Installation de Maven

Ant Pour installer `Ant`, on s'appuie sur les paquets mis à disposition dans la distribution. Pour se faire, il suffit de taper les commandes suivantes :

```
sudo apt-get install ant
```

Listing 6 – Installation de Ant

Git et Subversion Pour installer `Git` et `Subversion`, on s'appuie sur les paquets mis à disposition dans la distribution. Pour se faire, il suffit de taper les commandes suivantes :

```
sudo apt-get install git subversion
```

Listing 7 – Installation de Git et Subversion

Installation de Tomcat7

Pour installer `Tomcat 7`, on s'appuie sur les paquets mis à disposition dans la distribution. Pour se faire, il suffit de taper les commandes suivantes :

```
sudo apt-get install tomcat7 tomcat7-admin
```

Listing 8 – Installation de Maven

Il faut ensuite modifier le fichier `/etc/tomcat7/tomcat-users.xml` afin de pouvoir créer les rôles d'administrateur Pour installer `Maven`, on s'appuie sur les paquets mis à disposition dans la distribution. Pour se faire, il suffit de taper les commandes suivantes :

```
<role rolename="admin"/>
 <role rolename="admin-gui"/>
 <role rolename="manager"/>
 <role rolename="manager-gui"/>
 <user username="tomcat" password="tomcat"
    roles="admin,admin-gui,manager,manager-gui"/>
```

Listing 9 – Configuration des utilisateurs Tomcat

Enfin, il faut modifier le fichier de configuration de `Tomcat 7` afin d'ajouter certaines options `Java`. Le fichier a modifier est le suivant : `/etc/default/tomcat7`.

```
JAVA_OPTS="-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC
-XX:+CMSIncrementalMode -XX:+CMSClassUnloadingEnabled
-XX:+CMSPermGenSweepingEnabled -XX:MaxPermSize=128m"
```

Listing 10 – Fichier de configuration de Tomcat

Jenkins

Pour installer `Jenkins`, il est tout d'abord nécessaire de couper le serveur `Tomcat` car l'installation de `Jenkins` se fait par défaut sur le même port.

C'est pourquoi il faut exécuter la commande : `sudo /etc/init.d/tomcat7 stop`

Ensuite, il faut ajouter une nouvelle clé.

```
wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key |
sudo apt-key add -
ou
wget http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key
sudo apt-key add jenkins-ci.org.key
```

Listing 11 – Ajout de la clé de dépôt de Jenkins

Puis, il faut modifier le fichier `/etc/apt/sources.list` en ajoutant la ligne suivante :

```
# Depot Jenkins
deb http://pkg.jenkins-ci.org/debian binary/
```

Listing 12 – Modification du fichier sources.list

Après la modification du fichier, il faut mettre à jour les paquets disponibles avec la commande :

`sudo apt-get update`

Une fois ceci fait, il suffit d'exécuter la commande suivante :

`sudo apt-get install jenkins`

Une fois `Jenkins` installé, il faut stopper celui-ci pour modifier le port sur lequel il s'exécute. Pour se faire, taper la commande suivante :

`sudo /etc/init.d/jenkins stop`

Il faut ensuite modifier le fichier `/etc/default/jenkins`

```
#HTTP_PORT=8080
HTTP_PORT=8500
```

Listing 13 – Modification du port de Jenkins

Par défaut, le répertoire de travail de `Jenkins` est : `/var/lib/jenkins`. Ce répertoire suit la philosophie Linux sur l'emplacement des différents répertoires, c'est pourquoi nous ne modifions pas le répertoire de travail par défaut (il en est de même pour `Tomcat` : `/var/lib/tomcat7`).

Une fois ceci fait, il est possible de relancer `Jenkins` :

`sudo /etc/init.d/jenkins start`

Installation de Nexus

Selon la version de `Nexus` à installer il est nécessaire de mettre à jour la version de Java (en effet, la dernière version de Nexus : 2.6.1-02, requiert Java 7).

Pour l'installation, il suffit de télécharger directement le fichier war qui se trouve sur `http://www.sonatype.org/nexus/go`

Au préalable, il faut stopper `Tomcat` et copier le fichier `nexus.war` dans le répertoire `/var/lib/tomcat7/webapps/`

`sudo cp nexus.war /var/lib/tomcat7/`

Ensuite, il faut changer le propriétaire du fichier afin qu'il appartienne à `Tomcat`

```
cd /var/lib/tomcat7/
sudo chown tomcat7:tomcat7 nexus.war
```

Listing 14 – Modification du propriétaire du fichier

Il faut ensuite créer un répertoire de travail afin que `Nexus` puisse y déposer ses fichiers. On va donc créer un répertoire nexus dans le répertoire `/var/lib/` qui aura pour propriétaire *tomcat7*.

```
sudo mkdir /var/lib/nexus
sudo chown tomcat7:tomcat7 /var/lib/nexus
```

Listing 15 – Création du répertoire de travail de Nexus

Enfin, il est maintenant possible de relancer `Tomcat`.

A ce moment là, un répertoire `Nexus` va être créé, mais `Nexus` ne sera pas exécuté. En effet, il est nécessaire d'apporter quelques modifications au sein du fichier de configuration.

Il faut donc stopper tomcat et modifier le fichier

`/var/lib/tomcat7/webapps/nexus/WEB-INF/plexus.properties`

```
sudo vim /var/lib/tomcat7/webapps/nexus/WEB-INF/plexus.properties
#nexus-work=${user.home}/sonatype-work/nexus
nexus-work=/var/lib/nexus
```

Listing 16 – Modification du fichier de configuration Nexus

Maintenant que la modification a été faite, il est possible de relancer tomcat. Cette fois-ci `Nexus` devrait s'exécuter comme il faut.

Pour y aller, il suffit de taper dans un navigateur l'url de `Nexus` : `http://tomcat_host:8080/nexus`

Par défaut, pour se connecter à `Nexus`, le login/password administrateur est le suivant : admin/admin123

Maintenant, il faut ajouter un nouvel utilisateur que l'on utilisera pour faire les déploiements. Cet utilisateur est lié à `Jenkins` puisque c'est lui qui se chargera de faire les déploiements au sein de `Nexus` (cf. Figure 15).

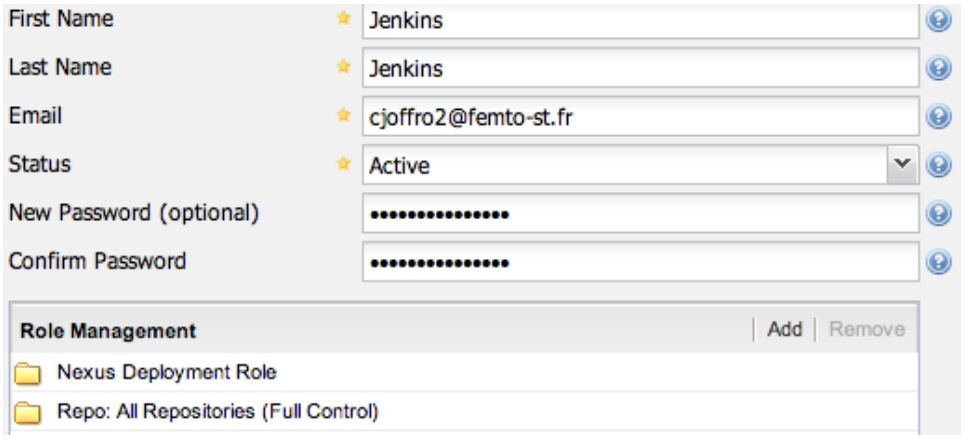

FIGURE 15 – Ajout d'un nouvel utilisateur Jenkins

La configuration de `Nexus` est terminée. Il est temps de repasser sur `Jenkins` pour finir sa configuration.

Configuration de Jenkins

Dans un premier temps (et puisque `Jenkins` est toujours stoppé), nous allons commencer par créer un répertoire `.m2` ainsi qu'un fichier `settings.xml` dans le répertoire `/var/lib/jenkins`.

```
sudo mkdir /var/lib/jenkins/.m2
sudo touch /var/lib/jenkins/.m2/settings.xml
sudo vim /var/lib/jenkins/.m2/settings.xml
```

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">
 <servers>
 <server>
 <id>nexus-ci</id>
 <username>jenkins</username>
 <password>password</password>
```

```
 </server>
 </servers>
 <mirrors>
 <mirror>
 <id>nexus-ci</id>
 <name>Nexus for Continuous Integration</name>
 <url>http://localhost:8080/nexus/content/groups/public/</url>
 <mirrorOf>*</mirrorOf>
 </mirror>
 </mirrors>
</settings>
```

Listing 17 – Création du fichier settings.xml pour Jenkins

Ensuite, on change le propriétaire du répertoire et du fichier nouvellement créé (dans l'exemple le propriétaire est *jenkins :nogroup*, mais il est important de vérifier quel est le propriétaire des fichiers qui se trouvent dans `/var/lib/jenkins`).

```
sudo chown -R jenkins:nogroup /var/lib/jenkins/.m2
```

Il est maintenant possible de relancer `Jenkins`

```
sudo /etc/init.d/jenkins start
```

Il faut ensuite configurer l'emplacement d'installation des différents outils (JDK, Maven, Ant). Pour cela, il faut aller dans : *Administrer Jenkins* → *Configurer le système.*

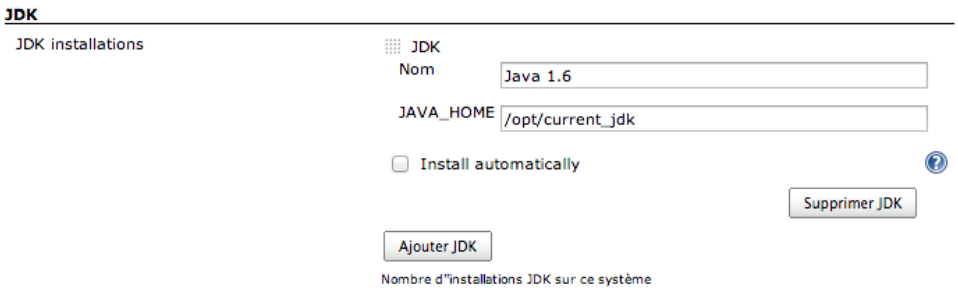

FIGURE 16 – Configuration de la JDK

Il faut ensuite installer certains plugins (il est d'ailleurs recommandé de mettre à jour les plugins déjà installés).

Voici la liste des plugins que nous allons installer :

- *M2 Release Plugin*
- *Github Plugin*
- *Git Plugin*
- *Jenkins Sonar Plugin* (pour faire le lien avec `Sonar` que nous allons installer ensuite, la configuration de celui-ci sera expliqué dans ce qui suit)

Il faut ensuite redémarrer `Jenkins` pour que les modifications soient prises en compte

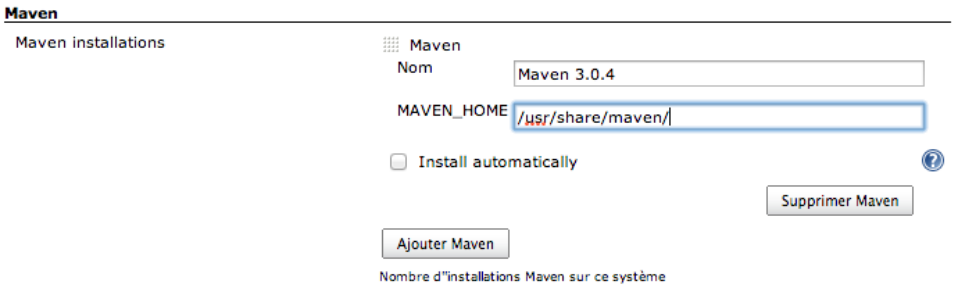

FIGURE 17 – Configuration de la Maven

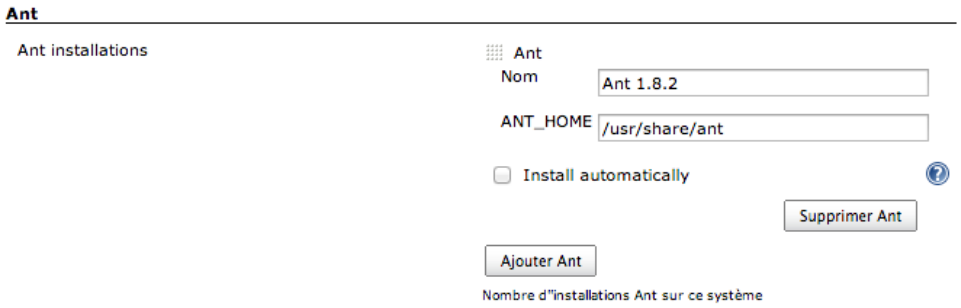

FIGURE 18 – Configuration de la Ant

Installation de Sonar

Avant de pouvoir installer `Sonar`, il faut installer un serveur de base de données. Dans cet exemple, nous allons utiliser `PostgreSQL 9.1`.

Pour installer `PostgreSQL`, il suffit de taper la commande suivante :

`sudo apt-get install postgresql-9.1`

A la fin de l'installation, le serveur `PostgreSQL` est démarré.

Il faut ensuite se connecter avec l'utilisateur *postgres* :

`sudo su - postgres`

De là, on va aller sur l'éditeur `psql` afin de créer un nouvel utilisateur ainsi qu'une nouvelle base de données :

```
postgres@jdevt6b:~$ psql
postgres=# CREATE USER sonar WITH PASSWORD 'sonar';
CREATE ROLE
postgres=# SELECT usename, usesysid FROM PG_USER;
 usename  | usesysid
----------+----------
 postgres |       10
 sonar    |    16384
(2 rows)
postgres=# CREATE DATABASE sonar WITH OWNER sonar ENCODING 'UTF8';
```

```
CREATE DATABASE
postgres=# SELECT datname, datdba, encoding FROM pg_database;
 datname | datdba | encoding
-----------+---------+----------
 template1 | 10 | 6
 template0 | 10 | 6
 postgres | 10 | 6
 sonar | 16384 | 6
(4 rows)
```

Listing 18 – Configuration de la base de données

Enfin *Ctrl+D* pour quitter (et exit pour quitter la session de *postgres*)

Ensuite, il faut télécharger `Sonar` sur le site :`http://www.sonarqube.org/downloads/`

Une fois le fichier télécharger, il faut le décompresser. On le copie ensuite dans le répertoire `/opt/`

```
sudo cp -r sonar-3.7 /opt/sonar
```

Une fois le répertoire copié, il faut modifier le fichier `sonar.properties` qui se trouve dans le répertoire `conf` (dans notre cas `/opt/sonar/conf/sonar` verb+.properties+). Dans ce fichier, nous allons faire la configuration de l'accès à la base de données. De fait, comme nous avons comme login/password de sonar : **sonar**/**sonar**, il n'y aura que deux choses à faire : commenter l'activation par défaut et décommenter l'accès à la base de données `PostgreSQL`.

```
sudo vim sonar.properties
# Comment the following line to deactivate the default
#embedded database.
#sonar.jdbc.url: jdbc:h2:tcp://localhost:9092/sonar
#----- PostgreSQL 8.x/9.x
# Comment the embedded database and uncomment the
#following property to use PostgreSQL
sonar.jdbc.url: jdbc:postgresql://localhost/sonar
```

Listing 19 – Modification du fichier de configuration de Sonar

Il faut ensuite aller dans le répertoire `war` (`/opt/sonar/war`), pour exécuter le fichier `build-war.sh` afin de générer le fichier sonar.war qui sera déployé dans `Tomcat`.

Avant de déployer le fichier obtenu dans `Tomcat`, nous allons tout d'abord stopper `Tomcat`

```
sudo /etc/init.d/tomcat7 stop
```

On change le propriétaire du répertoire sonar pour le faire appartenir à l'utilisateur *tomcat*.

```
sudo chown -R tomcat7:tomcat7 /opt/sonar
```

Puis, on copie le fichier obtenu dans le répertoire des `webapps` :

```
sudo cp sonar.war /var/lib/tomcat7/webapps
```

Attention de bien changer le propriétaire du fichier

```
sudo chown  tomcat7:tomcat7 /var/lib/tomcat7/webapps/sonar.war
```

Maintenant, il est possible de relancer `Tomcat`.

```
sudo /etc/init.d/tomcat7 start
```

Si tout s'est bien passé, un ensemble de tables a été créé. Il est possible de vérifier cela en exécutant les commandes suivantes :

```
psql -d sonar -h localhost -U sonar
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public';
```

Listing 20 – Contrôle de l'installation de Sonar

Vous devriez voir apparaître une cinquantaine de tables.

Configuration de Jenkins (Bis)

Maintenant que `Sonar` est installé, il est nécessaire de configurer `Jenkins` afin que celui-ci connaisse l'emplacement de celui-ci.

Il faut aller dans : *Administrer Jenkins* → *Configurer le système*

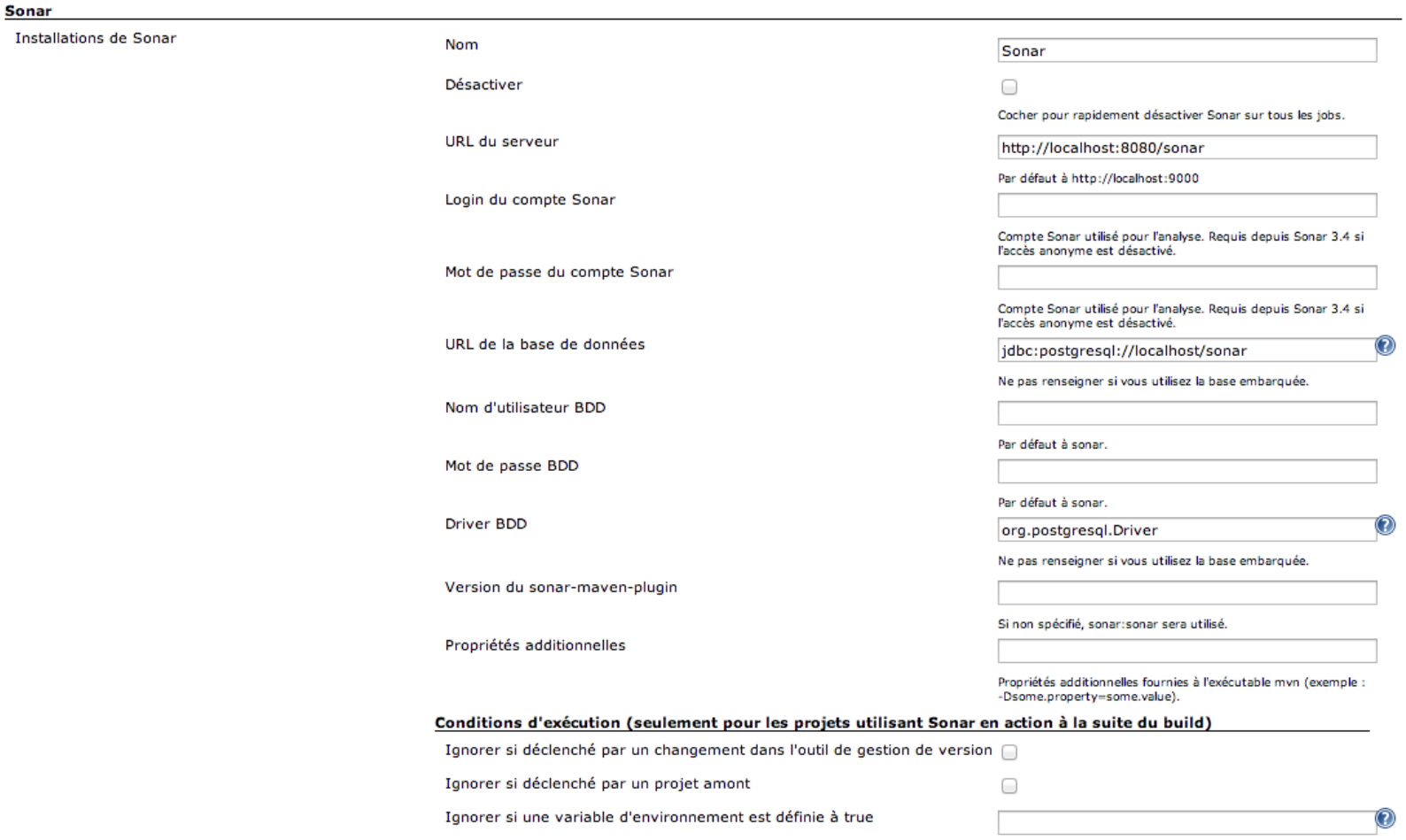

FIGURE 19 – Configuration de Sonar dans Jenkins