



# PROCÉDURES D'EXPLOITATION

Projet : Intégration continue d'OCS avec Jenkins	Auteur : Nathanaël TOUCHARD
Objet : Document d'exploitation	Date : Mai/Juin 2018

# Table des matières

1-Installation Jenkins.....	4
1.1-Problème d'installation.....	4
1.2-Commandes d'installation.....	4
1.3-Accès à jenkins.....	4
1.4-Récupération du login.....	5
1.5-Accès en CLI.....	6
2-Configuration de Jenkins de base.....	6
2.1-Plugins de base.....	6
2.2-Premier utilisateur Administrateur.....	7
2.3-Installation de nouveaux plugins.....	10
2.4-Configuration système.....	10
Configuration de GitLab.....	10
2.5-Configuration des outils.....	11
2.6-Configuration de la sécurité.....	12
2.7-Configurer une clé SSH.....	12
Configuration de la clé ssh sur GitLab.....	12
3-Créer un projet GitLab/GitHub.....	12
3.1-Plugin à installer.....	12
Pour un projet PHP :.....	12
3.2-Créer un nouveau job.....	13
3.3-Projet GitLab.....	14
3.4-Projet Github.....	15
3.5-Scrutation de Git.....	15
4-Configuration des builds.....	16
4.1-Pour un projet PHP.....	16
PHP Lint.....	16
PHP_CodeSniffer :.....	16
4.2-Configuration de Phing.....	19
4.3-Script de construction.....	20
4.4-Résultat du build.....	21
4.5-Publication des résultats de build.....	23
Analyse checkstyle pour PHP_CodeSniffer.....	24
5- Installations requises sur le ssh.....	27
5.1-Installations de base.....	27
5.2-Installation de version multiple de php.....	27
Téléchargement et décompression des fichiers binaires php.....	27
Compilation de PHP.....	28
6-Configuration pour OCS.....	28
6.1-Configuration système.....	28
6.2-Configuration de la sécurité.....	29
Exemple rôles .....	31
6.3-Configuration du job PHP.....	32
6.4-Configuration MySQL.....	34
6.5-Notification par email.....	35
6.6-Configuration Selenium.....	36
7-Tests automatisés dans OCS.....	37
7.1-Tests MySQL.....	37
7.2-Tests Selenium.....	37
7.3-Test installation du serveur OCS incluant les tests Selenium sur des serveurs SSH.....	39
8-Conclusion.....	39
9-Annexes.....	40
10-Bibliographies.....	41



# 1 Installation Jenkins

## 1.1 Problème d'installation

Il se peut qu'il y est un problème au moment de lancer les commandes : `-bash: sudo: command not found`

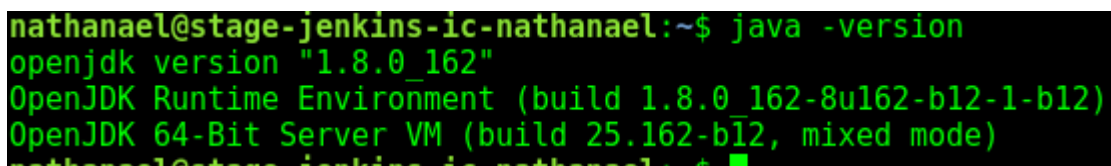
Il suffit alors de lancer la commande suivante pour installer sudo :

`apt-get install sudo`

Il vous faut aussi une bonne version de java pour pouvoir lancer jenkins, comme il s'agit d'un logiciel en java.

Regardez votre version avec la commande : `java -version`

Et la version doit être la suivante :



```
nathanael@stage-jenkins-ic-nathanael:~$ java -version
openjdk version "1.8.0_162"
OpenJDK Runtime Environment (build 1.8.0_162-8u162-b12-1-b12)
OpenJDK 64-Bit Server VM (build 25.162-b12, mixed mode)
```

*Illustration 1.1: Version de java*

Si jamais vous êtes sur une version plus récente vous pouvez choisir votre config avec la commande suivante : `sudo update-alternatives --config java`

## 1.2 Commandes d'installation

Installation Jenkins sur le ssh :

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

`sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'`

`sudo apt-get update`

`sudo apt-get install jenkins`

## 1.3 Accès à jenkins

<http://172.18.26.211:8080>

serveur dev : 172.18.26.211

Vous obtiendrez la page suivante : (Illustration 1.2 : Page de déblocage de Jenkins)

# Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

*Illustration 1.2: Page de déblocage de Jenkins*

Si ce n'est pas le cas vous pouvez tenter d'utiliser la commande `sudo systemctl restart jenkins` pour restart jenkins. Ou utiliser `sudo systemctl stop jenkins` pour stopper la connexion et ensuite vous pouvez restart.

Il vous faudra récupérer le mot de passe administrateur pour débloquer Jenkins.

Vous pouvez changer le port http en vous plaçant dans le dossier : `etc/default/` et en faisant un `sudo nano` du fichier jenkins vous pouvez ainsi modifier le port à la ligne `HTTP_PORT=9090` par exemple.

## 1.4 Récupération du login

Vous pouvez retrouver le login de l'admin avec la commande : `cat /var/log/jenkins/jenkins.log` depuis la console et vous obtiendrez le mot de passe administrateur entre les 2 morceaux d'astérisques. (voir Illustration 1.3 : Récupération du mot de passe administrateur)

```
INFO:
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
7035073050924daeb83b919bcf7007d3
This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
*****
*****
*****
```

*Illustration 1.3: Récupération du mot de passe administrateur*

Ensuite cliquez sur continuez.

## 1.5 Accès en CLI

Il existe un accès en ligne de commande, qui est surtout utile pour scripter des opérations de maintenance.

Téléchargement du fichier .jar permettant l'accès en CLI :

```
wget http://172.18.26.215:8080/jnlpJars/jenkins-cli.jar
```

Exemple de commande pour afficher toutes les commandes disponibles :

```
java -jar jenkins-cli.jar -s http://172.18.26.215:8080/ help
```

Il est possible qu'une erreur s'affiche, vous devrez alors sûrement vous authentifier en ajoutant votre username et password après la commande :

```
java -jar jenkins-cli.jar -s http://172.18.26.211:8081/ help --username user --password pwd
```

## 2 Configuration de Jenkins de base

### 2.1 Plugins de base

Pour démarrer Jenkins vous pouvez soit installer les plugins suggérés, c'est à dire les plus souvent utilisés, ou bien sélectionner vous même les plugins à installer selon vos besoins (voir Illustration 2.1 : Première installations des plugins).

Pour commencer on peut choisir l'installation de base, on pourra par la suite installer d'autres plugins plus en détails.



Illustration 2.1: Première installations des plugins

## 2.2 Premier utilisateur Administrateur

Une fois la configuration des plugins effectuée, vous pouvez désormais créer le 1<sup>er</sup> utilisateur qui sera Administrateur et pourra ainsi configurer le système Jenkins. (voir Illustration 2.2 : Création du 1<sup>er</sup> Administrateur)

Démarrage

# Créer le 1er utilisateur Administrateur

Nom d'utilisateur:

Mot de passe:

Confirmation du mot de passe:

Nom complet:

Adresse courriel:

Illustration 2.2: Création du 1<sup>er</sup> Administrateur

Vous vous trouverez alors sur la page d'accueil de Jenkins. (voir Illustration 2.3 : Page d'accueil de Jenkins)

 **Jenkins**

rechercher

Administrateur | se déconnecter

Jenkins

Nouveau Item

Utilisateurs

Historique des constructions

Administrer Jenkins

Mes vues

Identifiants

New View

File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations

1 Au repos

2 Au repos

Bienvenue sur Jenkins !

Veillez **créer un nouveau job** pour démarrer.

Rafraîchissement automatique

Ajouter une description

Page générée: 3 mai 2018 12:54:38 UTC

[REST API](#)

Jenkins ver. 2.107.2

Illustration 2.3: Page d'accueil de Jenkins

## 2.3 Installation de nouveaux plugins

Vous pouvez installer d'autres plugins une fois sur Jenkins, pour cela rendez vous sur « **Administrer Jenkins > Gestion des plugins** » et ensuite vous verrez 4 onglets :

Mises à jour : très explicite vous pouvez vérifier s'il existe des mises à jour.

Disponibles : c'est l'onglet intéressant qui va permettre de trouver et installer de nouveaux plugins.

Installés : vous trouverez les plugins que vous avez installé, et vous pourrez aussi les désinstaller si vous les jugez non nécessaire.

Avancé : vous trouverez des options avancées pour les plugins.

## 2.4 Configuration système

Dans la partie « **Administrer Jenkins > Configurer le système** », vous pouvez laisser tel quel pour l'instant, la configuration de base fait l'affaire pour des petits projets, il vous faudra juste configurer la partie GitLab si vous en avez besoin.

C'est ici que vous pouvez gérer les chemins vers les différents outils que utiliserez dans vos builds. Plusieurs plugins installés devront être aussi configuré ici.

### Configuration de GitLab

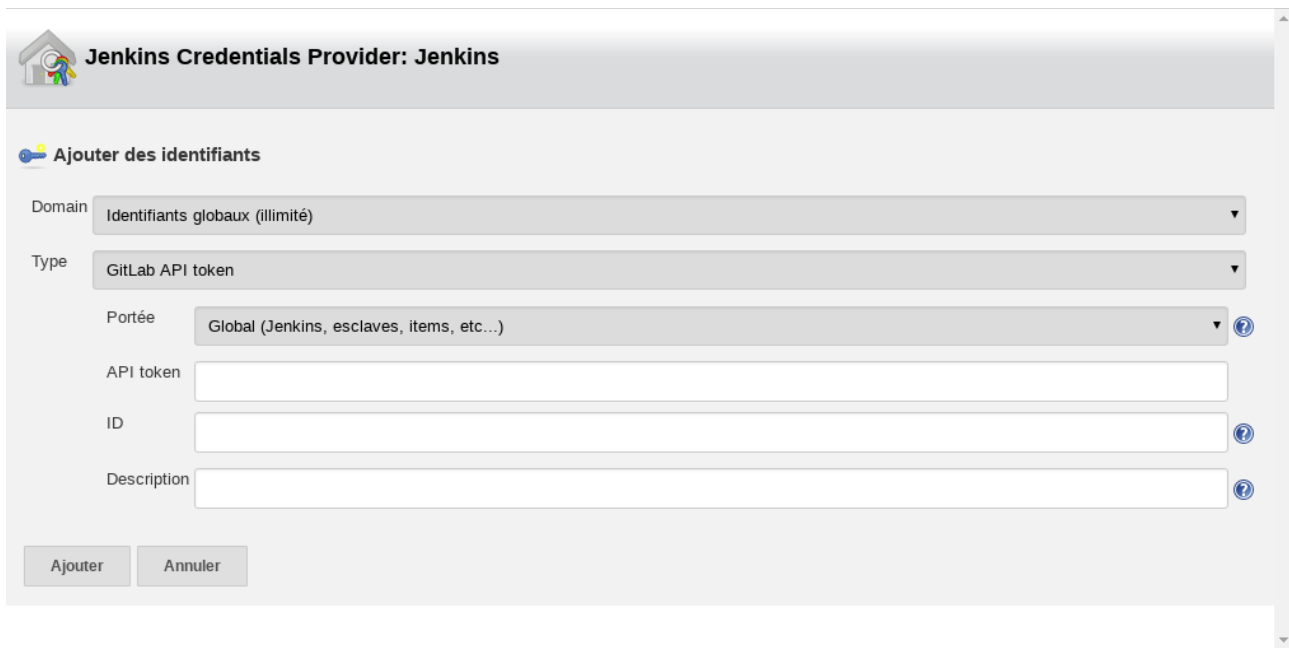
Pour configurer GitLab sur Jenkins , vous devez vous rendre dans la partie Administrer Jenkins et dans la partie Gitlab informez un nom et l'URL http du Gitlab. (voir Illustration 2.4 : Configuration du GitLab)

The screenshot shows the 'GitLab' configuration section in Jenkins. At the top, there is a checkbox 'Enable authentication for "/>

*Illustration 2.4: Configuration du GitLab*

Ensuite il faudra ajouter l'API du Gitlab dans la partie « **Credentials** »(voir Illustration 2.5 : Ajout de l'API pour GitLab). Vous pouvez retrouver l'API dans les settings de GitLab et dans l'onglet account.



The screenshot shows the 'Jenkins Credentials Provider: Jenkins' interface. At the top, there's a header with the Jenkins logo and the title. Below it, a section titled 'Ajouter des identifiants' (Add credentials) contains a form. The form has several fields: 'Domain' with a dropdown menu showing 'Identifiants globaux (illimité)', 'Type' with a dropdown menu showing 'GitLab API token', 'Portée' (Scope) with a dropdown menu showing 'Global (Jenkins, esclaves, items, etc...)', 'API token' with a text input field, 'ID' with a text input field, and 'Description' with a text input field. Each of the last three fields has a help icon (a question mark in a circle) to its right. At the bottom of the form, there are two buttons: 'Ajouter' (Add) and 'Annuler' (Cancel).

*Illustration 2.5: Ajout de l'API pour GitLab*

Enfin vous pouvez cliquer sur le bouton Test Connection, et si vous avez suivi la démarche vous devez obtenir un succès.

## 2.5 Configuration des outils

Vous pouvez configurer aussi les outils dans « [Administrer Jenkins](#) > [Configuration globale](#) » des outils, pour indiquer la localisation d'outils dans votre serveur ou leur installateurs automatique pour les ajouter à votre serveur. Pour l'instant nous avons juste besoin de Git, que vous pouvez installer avec la commande (voir partie 5).

## 2.6 Configuration de la sécurité

Vous pouvez configurer la sécurité de votre système Jenkins « [Administrer Jenkins](#) > [Configurer la sécurité globale](#) », pour la configuration de base, celle ci correspond très bien, on verra plus tard pour la configurer avec un projet plus important.

Dans cette configuration vous pouvez contrôler les accès et la gérer la protection.

## 2.7 Configurer une clé SSH

Il vous faut configurer la clé ssh pour établir une connexion entre GitLab et votre système Jenkins.

### Configuration de la clé ssh sur GitLab

Tout d'abord vous pouvez utiliser la commande : `cat ~/.ssh/id_rsa.pub` pour voir si une clé ssh est déjà existante.

Sinon vous devez en générer une : `ssh-keygen -t rsa -C "votre adresse gitlab" -b 4096`

Ensuite vous pouvez retrouver la clé ssh générée dans votre terminal avec la commande :

```
cat ~/.ssh/id_rsa.pub
```

Et ensuite vous pouvez ajouter votre clé ssh dans la partie « **Settings > Key SSH** » de GitLab.

## 3 Créer un projet GitLab/GitHub

### 3.1 Plugin à installer

GitLab Plugin : pour intégrer Jenkins avec des projets GitLab

GitHub Plugin : pour intégrer Jenkins avec des projets GitHub

Git Plugin : pour l'intégration des dépôts Git

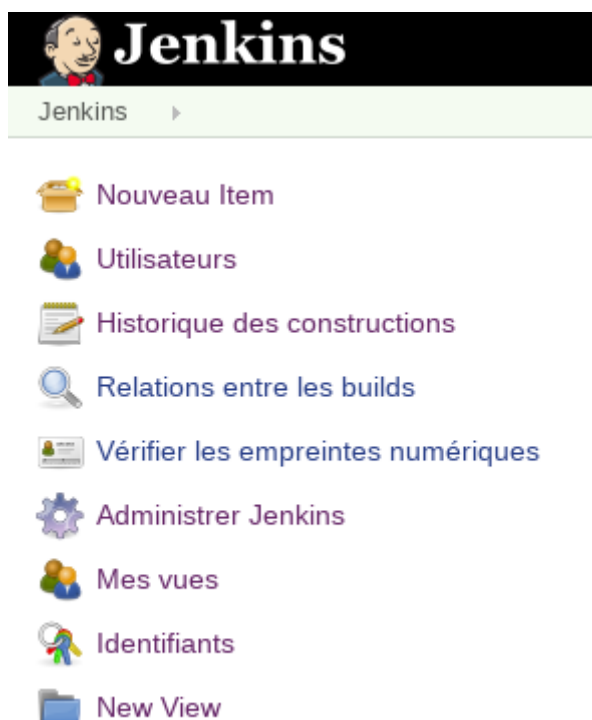
Display Console Output Plugin : permet d'avoir un affichage du dernier build sur la page du projet

### Pour un projet PHP :

Phing : Outil d'automatisation de tâches, cet outil va exécuter des tâches à la suite qui sont décrites dans un fichier **build.xml**

### 3.2 Créer un nouveau job

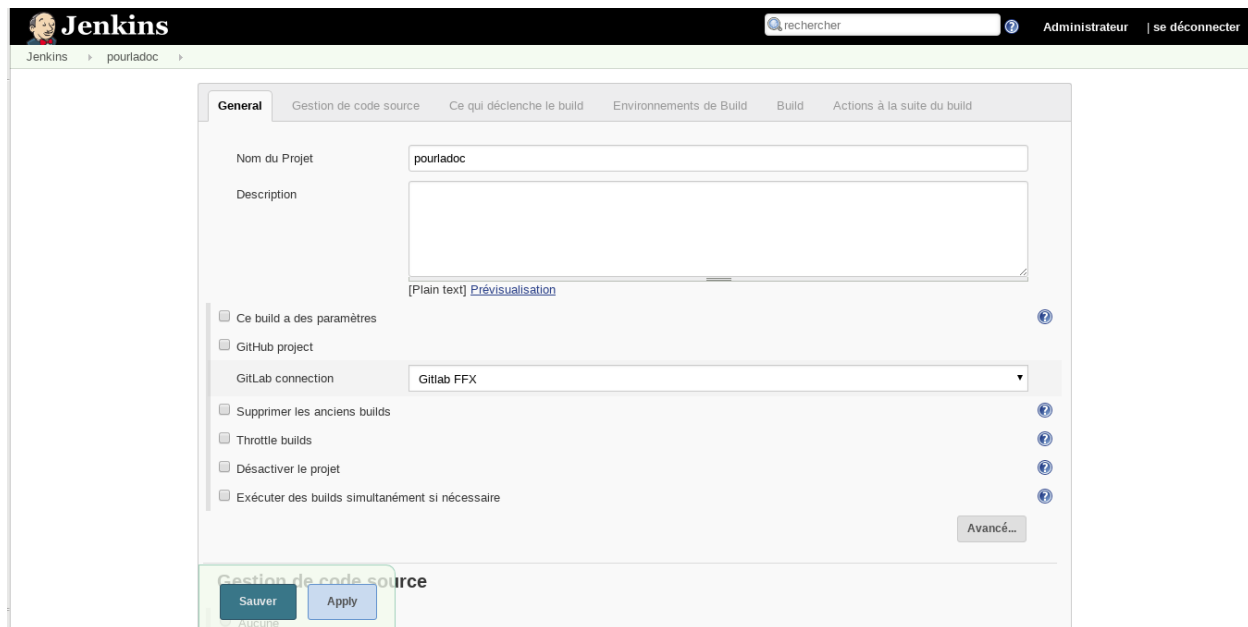
Cliquez sur « **Nouveau Item** » pour pouvoir créer un nouveau projet (voir Illustration 3.1: Menu de Jenkins).



*Illustration 3.1: Menu de Jenkins*

Ensuite vous vous trouverez sur la page de création d'un job, saisissez un nom pour votre projet et après vous devez choisir votre type de projet, pour intégrer vos outils choisissez de construire un « **projet free-style** ». Cliquez sur OK.

Vous serez alors sur la page pour configurer votre projet. (Illustration 3.2 : Page de configuration d'un projet)

The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar labeled 'rechercher', and links for 'Administrateur' and 'se déconnecter'. Below the header, a breadcrumb trail shows 'Jenkins' > 'pourladoc'. The main content area is titled 'General' and contains several tabs: 'Gestion de code source', 'Ce qui déclenche le build', 'Environnements de Build', 'Build', and 'Actions à la suite du build'. The 'General' tab is active. It features a 'Nom du Projet' field with the value 'pourladoc' and a 'Description' text area. Below these, there are several checkboxes: 'Ce build a des paramètres', 'GitHub project', 'Supprimer les anciens builds', 'Throttle builds', 'Désactiver le projet', and 'Exécuter des builds simultanément si nécessaire'. A 'GitLab connection' dropdown menu is set to 'Gitlab FFX'. On the right side of the configuration area, there are five question mark icons. At the bottom of the configuration area, there are 'Sauver' and 'Apply' buttons. Below the configuration area, there's a section titled 'Gestion de code source' with a 'Sauver' button. The overall layout is clean and professional, typical of a web-based configuration tool.

*Illustration 3.2: Page de configuration d'un projet*

### 3.3 Projet GitLab

Pour intégrer un projet GitLab dans votre système Jenkins, il vous suffit de vous rendre dans Gestion de code source, cochez le bouton Git, et ici vous devez entrer l'URL de votre projet GitLab, ensuite il vous faut remplir la partie « **Credentials** », ajouter un identifiant de type Nom d'utilisateur et mot de passe. (voir Illustration 3.3 : Ajout d'un identifiant GitLab)

*Illustration 3.3: Ajout d'un identifiant GitLab*

Une fois avoir configurer votre GitLab vous pouvez sauvegarder et alors votre Jenkins aura intégrer votre GitLab. Vous n'aurez plus qu'a configurer les builds (voir partie 4).

### 3.4 Projet Github

Pour intégrer un projet GitHub ça sera sensiblement la même chose que pour GitLab. Dans la partie Gestion de code source, il vous suffit de rentrer l'URL de votre projet GitHub et s'il s'agit d'un projet privé, il vous faut rentrer votre identifiant comme pour GitLab, sinon il n'y a rien à faire.

### 3.5 Scrutation de Git

Pour déclencher un build vous pouvez utiliser l'outil dans votre configuration du projet **Ce qui déclenche le build > Scrutation de l'outil de gestion de version**. Vous pouvez définir un planning d'horaires pour scruter le git et voir si des changements sont apportés, si oui le build se déclenchera.

Par exemple ici on choisi de scruter toute les minutes (voir Illustration 3.4: Scrutation de l'outil de gestion de version).

*Illustration 3.4: Scrutation de l'outil de gestion de version*

Le format est de type : MINUTE HOUR DOM MONTH DOW

- MINUTE : minutes entre 0 et 59
- HOUR : heure entre 0 et 23
- DOM : jour du mois entre 1 et 31
- MONTH : mois entre 1 et 12
- DOW : jour de la semaine entre 0 et 7 (0 et 7 sont le dimanche)
  - \* : représente toutes les valeurs
  - M-N : représente une plage de valeurs
  - M-N/X ou \*/X effectue des pas de X dans la plage spécifiée ou dans toute la plage
  - A,B,...,Z énumère plusieurs valeurs

Vous pouvez cliquer sur la petite bulle d'aide à droite pour plus d'informations.

## 4 Configuration des builds

### 4.1 Pour un projet PHP

#### PHP Lint

Soit vous pouvez exécuter un script shell pendant votre build sur vos fichiers php avec **PHP Lint** (voir Illustration 4.1: Exécution d'un script shell).

Sinon vous pouvez créer un fichier **build.xml** que le plugin Phing va utiliser pour exécuter des tâches (voir partie 4.2).

#### PHP\_CodeSniffer :

Pour **PHP\_CodeSniffer** vous pouvez aussi utiliser une commande shell et l'exécuter (voir Illustration 4.1: Exécution d'un script shell) mais le mieux est d'utiliser le plugin Phing (voir partie 4.2).

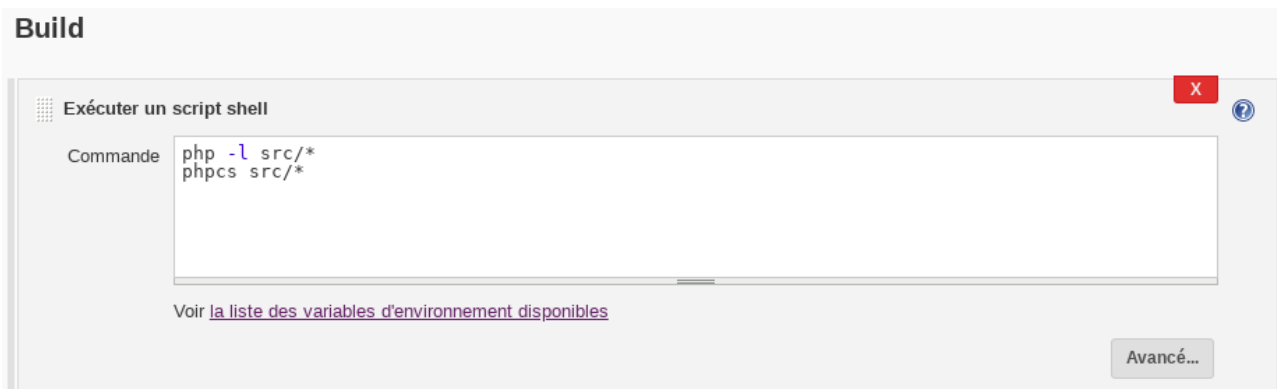


Illustration 4.1: Exécution d'un script shell

## 4.2 Configuration de Phing

Comme vous avez pu voir précédemment Phing va exécuter des tâches décrites dans un fichier **build.xml**. Vous allez donc pouvoir ajouter une étape au build, qui invoquera une target phing (voir Illustration 4.2: Invocation d'une target Phing).

Et ensuite vous pouvez renseigner la liste « **Targets** » par exemple dans le build de ce projet, le build entier est invoqué depuis la target build (voir **build.xml**) et dans avancé il vous faut rentrer le chemin vers le script de build dans le cadre « Phing Build File ».

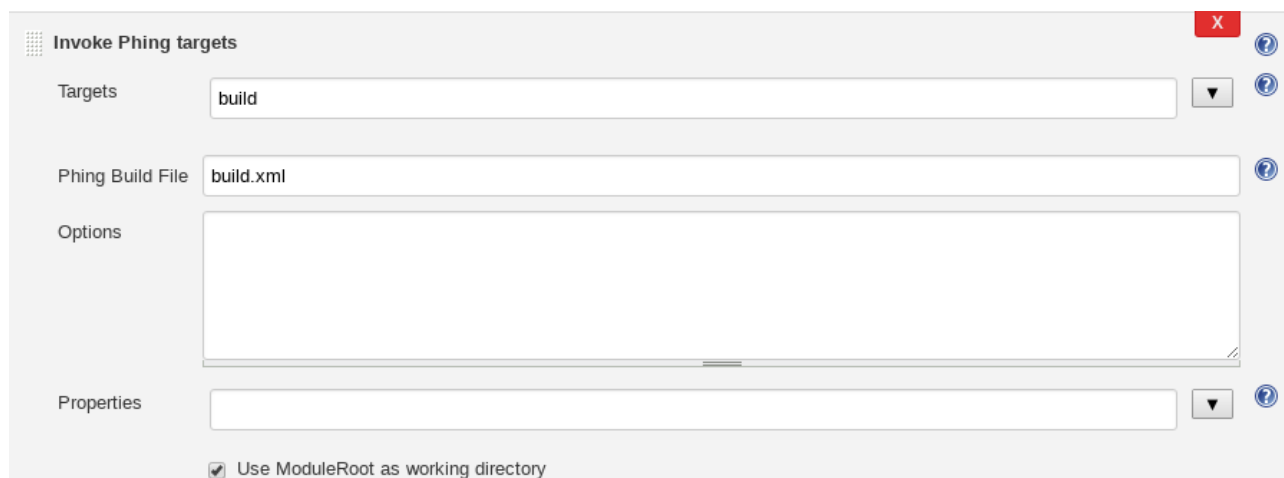


Illustration 4.2: Invocation d'une target Phing

Après ce qui est intéressant est de configurer des actions à effectuer à la suite d'un build, par exemple publier les résultats d'analyse de PHP\_CodeSniffer, mais vous verrez ça dans une prochaine partie.

Vous pouvez donc sauver votre configuration. Vous vous trouverez alors sur la page d'accueil de votre job (voir Illustration 4.3: Page d'accueil d'un job).

The screenshot shows the Jenkins web interface for a project named 'pourladoc'. The top navigation bar includes the Jenkins logo, a search bar, and links for 'Administrateur' and 'se déconnecter'. The left sidebar contains various management links. The main area displays project-specific information, including a list of permanent links to build history and a console output window showing the execution of shell commands and the resulting errors.

Illustration 4.3: Page d'accueil d'un job

Vous pouvez voir dans l'encadré en bas à gauche l'Histoire des builds, qui vous indiquera si vos builds réussissent ou échouent, la date du build.

Pour l'instant vous pouvez lancer un build, mais il échouera car vous n'avez pas mis en place le fichier **build.xml**.

Vous pouvez cliquer sur le build pour obtenir des détails sur la construction de ce build. Par exemple la console qui vous écrira les résultats de vos commandes shell. (voir Illustration 4.4: Détails de construction d'un build)

This screenshot shows the detailed view of a Jenkins build. The top section displays the build's name, number, and timestamp. Below this, the console output is shown, indicating that no changes were detected and the build was triggered by the administrator. The revision hash and the Git repository path are also visible.

Illustration 4.4: Détails de construction d'un build

## 4.3 Script de construction

Dans la configuration de notre job plus haut vous avez défini la target phing build contenu dans le fichier **build.xml**.

La target build dépend de plusieurs targets :

- **prepare** : qui va nettoyer les répertoires de reportings (dossier où les outils d'analyse enregistreront leurs rapports) et recréer ces répertoires.

- **lint** : va exécuter la commande `php -l` sur les répertoires que vous lui indiquez pour faire une analyse syntaxique, ici on va chercher la commande `php` des différentes versions (c'est à dire la 5.4.24 ; 5.6.36 ; 7.0.30) d'où le chemin `/opt/php-version/sapi/cli/php`, c'est pour cela que `lint` va dépendre de `lint 5.4.24`, `lint 5.6.36` et `lint 7.0.30`.
- **phpcs-ci** : va exécuter la commande `phpcs` pour vérifier le code et rendre les violations du code standard. De plus dans cette target, on va générer un fichier `checkstyle.xml` que l'on va placer dans les dossiers que l'on a préparé avant, et ensuite vous pourrez publier ces résultats avec un plugin.

## 4.4 Résultat du build

Une fois ce fichier commit, attendez quelques instants et Jenkins lancera un build, vous obtiendrez un succès et la console vous affichera ceci :

### Sortie de la console

```
Started by user Administrateur
Building in workspace /var/lib/jenkins/workspace/pourladoc
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://gitlab.factorfx.com/jenkins-ffx/testJenkins # timeout=10
Fetching upstream changes from https://gitlab.factorfx.com/jenkins-ffx/testJenkins
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --tags --progress https://gitlab.factorfx.com/jenkins-ffx/testJenkins +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision a39742d6e5ff59b1fc97947d6fd29b0e185edb82 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f a39742d6e5ff59b1fc97947d6fd29b0e185edb82
Commit message: "correction build"
> git rev-list --no-walk 3d28b49aa03a52e121a942d475c29daa89f11e95 # timeout=10
looking for '/var/lib/jenkins/workspace/pourladoc/build.xml' ...
use '/var/lib/jenkins/workspace/pourladoc' as a working directory.
[pourladoc] $ phing -buildfile /var/lib/jenkins/workspace/pourladoc/build.xml build -logger phing.listener.DefaultLogger
Buildfile: /var/lib/jenkins/workspace/pourladoc/build.xml

PHPJenkins > clean:

[delete] Directory /var/lib/jenkins/workspace/pourladoc/build/logs does not exist or is not a directory.

PHPJenkins > prepare:

[mkdir] Created dir: /var/lib/jenkins/workspace/pourladoc/build/logs

PHPJenkins > lint 5.4.24:

PHPJenkins > lint 5.6.36:

PHPJenkins > lint 7.0.30:

PHPJenkins > lint:

PHPJenkins > phpcs-ci:

PHPJenkins > build:

BUILD FINISHED

Total time: 0.5840 seconds
Finished: SUCCESS
```

*Illustration 4.5: Succès de construction d'un build et la console de sortie*



Cela signifie que votre build a réussi, que le fichier **build.xml** existe, et correspond à un fichier Phing valide.

Ensuite les tâches ont été lancées. Si vous regardez votre console on peut voir que les targets ont bien été invoquées.

## 4.5 Publication des résultats de build

Maintenant que votre projet PHP est intégré dans Jenkins, que vous avez créé votre script de build qui lance des outils d'analyse. Vous pouvez alors publier les résultats de ces analyses.

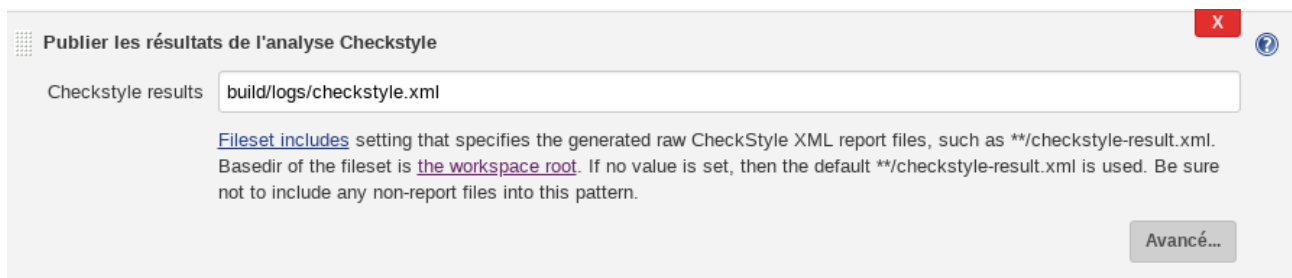
Il suffit de retourner sur la page de configuration de votre job, et vous trouverez dans les onglets (ou en bas de page) la section « **Actions à la suite du Build** ».

Vous pourrez alors configurer les actions de publications correspondant aux outils d'analyse que vous avez mis en place dans votre build.

### Analyse checkstyle pour PHP\_CodeSniffer

Dans votre fichier **build.xml** vous invoquez l'outil **PHP\_CodeSniffer**, qui vous génère un fichier **build/logs/checkstyle.xml**.

Pour obtenir un affichage vous devez renseigner le chemin du fichier **checkstyle.xml** dans l'action suivante :



*Illustration 4.6: Action après un build pour publier les résultats d'analyse PHP\_CodeSniffer*

Dans la section « **Avancé...** » vous pouvez accéder à toute la configuration de votre action, mais pour l'instant celle de base suffit. Enregistrez votre projet. Lancez un build, et dans la console vous verrez que Jenkins a analysé le fichier **checkstyle.xml** (voir Illustration 4.7: Sortie de console pour l'analyse du checkstyle).

```
[CHECKSTYLE] Collecting checkstyle analysis files...
[CHECKSTYLE] Searching for all files in /var/lib/jenkins/workspace/pourladoc that match
the pattern build/logs/checkstyle.xml
[CHECKSTYLE] Parsing 1 file in /var/lib/jenkins/workspace/pourladoc
[CHECKSTYLE] Successfully parsed file
/var/lib/jenkins/workspace/pourladoc/build/logs/checkstyle.xml with 20 unique warnings
and 0 duplicates.
<Git Blamer> Using GitBlamer to create author and commit information for all warnings.
<Git Blamer> GIT_COMMIT=a39742d6e5ff59b1fc97947d6fd29b0e185edb82,
workspace=/var/lib/jenkins/workspace/pourladoc
> git rev-parse a39742d6e5ff59b1fc97947d6fd29b0e185edb82^{commit} # timeout=10
Finished: SUCCESS
```

Illustration 4.7: Sortie de console pour l'analyse du checkstyle

Et vous trouverez un onglet « **Résultats Checkstyle** » dans le menu à gauche. Si vous cliquez dessus, vous trouverez plusieurs écrans qui sont les résultats de votre analyse avec **PHP\_CodeSniffer** avec les catégories des erreurs, le type, les priorités, l'origine des erreurs et les détails (voir Illustration 4.8: Résultat de l'analyse checkstyle).

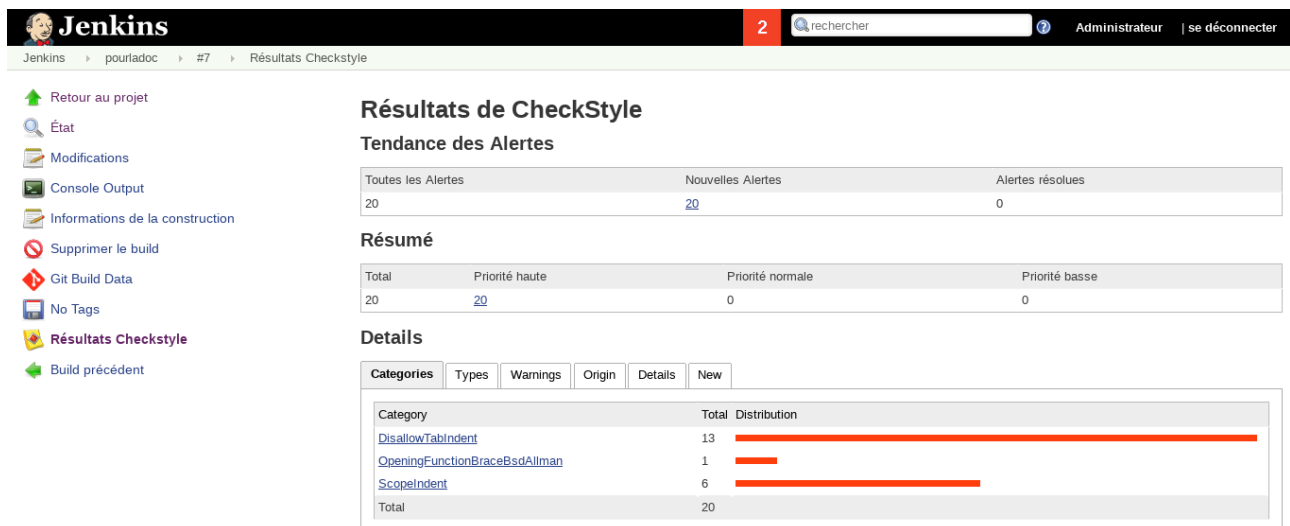


Illustration 4.8: Résultat de l'analyse checkstyle

Si vous re-lancez un build, vous trouverez à gauche un graphe montrant l'évolution des résultats d'analyse (voir Illustration 4.9: Graphe d'évolution de l'analyse avec PHP\_CodeSniffer).

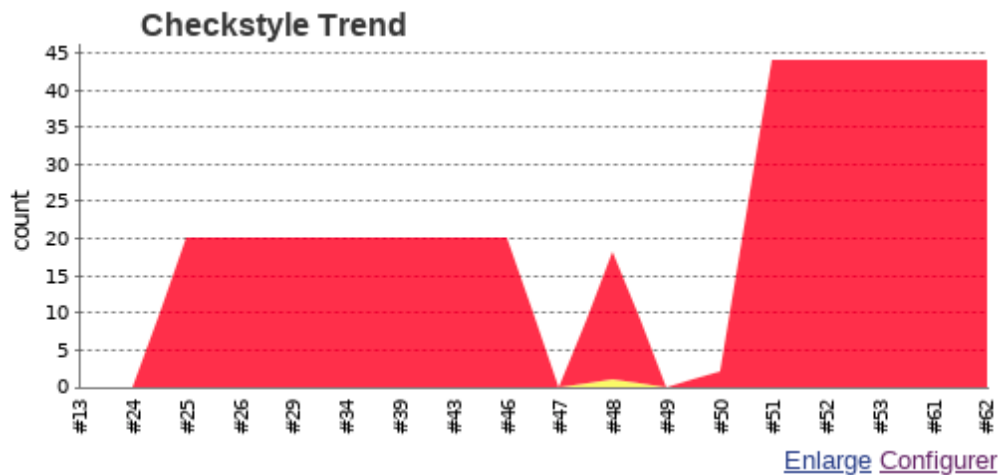


Illustration 4.9: Graphe d'évolution de l'analyse avec PHP\_CodeSniffer

## 5 Installations requises sur le ssh

### 5.1 Installations de base

#### Installation de Git :

```
sudo apt-get install git-core
```

#### Installation de php :

```
sudo apt install apache2 php mysql-server libapache2-mod-php php-mysql
```

#### Installation de pear:

```
sudo apt install php-pear
```

#### Installation de Xvfb :

```
sudo apt install xvfb
```

### 5.2 Installation de version multiple de php

Pour éviter les problèmes de version php on va installer plusieurs version pour s'adapter à toutes les versions de php.

#### **Téléchargement et décompression des fichiers binaires php**

Tout d'abord placez vous de le dossier /opt/.

#### PHP 5.4 :

```
$ wget -c http://fr2.php.net/get/php-5.4.24.tar.gz/from/this/mirror
```

#### PHP 5.6 :

```
$ wget -c http://fr2.php.net/get/php-5.6.36.tar.gz/from/this/mirror
```

### **PHP 7.0 :**

```
$ wget -c http://fr2.php.net/get/php-7.0.30.tar.gz/from/this/mirror
```

Ensuite il faut extraire le contenu du fichier tar en utilisant la commande suivante :

```
$ tar -zxvf fichier.tar.gz
```

Vous avez désormais les dossiers contenant les différentes versions de php.

## **Compilation de PHP**

Avant de compiler php, vérifiez que vous avez bien les installations suivante :

```
$ sudo apt install autoconf
```

```
$ sudo apt install make gcc
```

Maintenant vous pouvez compiler php. D'abord placez vous dans le dossier php, vous devez construire la commande **configure** avec la commande suivante :

```
$ sudo ./buildconf --force
```

Une fois la commande créée vous pouvez l'utiliser pour installer php. Vous pouvez exécuter `./configure --help` et vérifier s'il y a quelque chose que vous avez besoin. La commande suivante installera PHP :

```
$ sudo ./configure --prefix=/opt/php-5.4.24/php
```

Si vous rencontrez l'erreur suivante :

```
configure: error: xml2-config not found. Please check your libxml2 installation.
```

Il vous suffit d'installer la librairie suivante :

```
$ sudo apt-get install libxml2-dev
```

Pour finir il faut compiler php avec la commande suivante :

```
$ sudo make
```

Vous pouvez désormais utiliser la commande php avec la version que vous venez d'installer en la trouvant dans le dossier `sapi/cli/php`

Vous devrez refaire cette manipulation pour les autres versions.

## **6 Configuration pour OCS**

### **6.1 Configuration système**







Notification par mails

Installez le plugin Email Extension Plugin

## 6.2 Configuration de la sécurité

Pour la sécurité d'OCS vous pouvez gérer les autorisations soit selon une matrice, pour cela rendez vous dans la partie « **Administrer Jenkins > Configurer la sécurité globale > Autorisation > Stratégie d'autorisation matricielle basée sur les projets** ». Dans cette matrice vous pouvez gérer les autorisations d'utilisateurs par exemple les utilisateurs connectés peuvent lire des jobs et des builds mais pas les utilisateurs non connectés (voir Illustration 6.1: Matrice d'autorisations).

● Stratégie d'autorisation matricielle basée sur les projets

Utilisateur/groupe	Global		Identifiants				Agent								Job								
	Administer	Read	Create	Delete	Manage	Domains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace
 Anonymous Users	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 Authenticated Users	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
 Chef-de-projet	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 Administrateur	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
 dev	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
 Nathanael Touchard	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Utilisateur/groupe à ajouter:		<input type="text"/>	<div>Ajouter</div>																				

Utilisateur/groupe à ajouter:

Ajouter

Illustration 6.1: Matrice d'autorisations

Ce qui est intéressant dans cette matrice c'est que vous pouvez ajouter des groupes, c'est à dire que plusieurs personnes pourront se connecter au groupe, et auront tous les mêmes droits. Par exemple un groupe « Chef de projet » qui auront plus de droit d'administration qu'un groupe « dev ».

Sinon vous pouvez utiliser le plugin « Role-based Authorization Strategy » pour définir des rôles à vos utilisateurs. Ensuite cliquez sur « **Administrer Jenkins > Configurer la sécurité globale > Autorisation > Stratégie basée sur les rôles** » (voir Illustration 6.2: Stratégie basée sur les rôles).

### Autorisations

● Les utilisateurs connectés peuvent tout faire



● Mode legacy



● **Stratégie basée sur les rôles**



● Stratégie d'autorisation matricielle basée sur les projets



● Sécurité basée sur une matrice



● Tout le monde a accès à toutes les fonctionnalités



Illustration 6.2: Stratégie basée sur les rôles

Maintenant dans les onglets d'administration de Jenkins vous trouverez « **Gérer et assigner les rôles** » pour gérer les permissions (voir Illustration 6.3: Nouvel onglet de gestion de rôle).



### Gérer et assigner les rôles

Gérer les permissions en créant des rôles et en les assignant aux utilisateurs/groupes

Illustration 6.3: Nouvel onglet de gestion de rôle

Si vous vous connectez sur un utilisateur autre que l'administrateur vous verrez qu'aucunes permissions n'est assignées pour l'instant. (voir Illustration 6.4: Utilisateur sans permissions).

## Access Denied

— user1 is missing the Global/Read permission

*Illustration 6.4: Utilisateur sans permissions*

Allez dans « [Administrer Jenkins](#) > [Gérer et assigner les rôles](#) » vous y trouverez 3 liens, le premier « [Gérer les rôles](#) » vous permet de créer des rôles et gérer leurs permissions. Dans le deuxième « [Assigner les rôles](#) » vous pourrez assigner un rôle à des utilisateurs. Et le troisième « [Role Strategy Macros](#) » fournit des infos sur l'utilisation et la disponibilités des macros.



## Gérer et assigner les rôles



[Gérer les rôles](#)

Manage Roles



[Assigner les rôles](#)

Assign Roles



[Role Strategy Macros](#)

Provides info about macro usage and available macros

Cliquez sur « [Gérer les rôles](#) », ensuite vous trouverez plusieurs types de rôles :

- les rôles globaux qui regroupent la plupart des permissions du Jenkins
- les rôles de projets pour donner des permissions juste sur les projets
- et les slaves roles pour mettre des permissions concernant les nœuds (on ne s'y intéressera pas pour le moment)

Chaque case représente une permission (l'admin ayant tout de coché). Vous pouvez ajouter des rôles et ainsi leur donner des droits.

### Exemple rôles

On se rend dans la partie « [Gérer les rôles](#) ».



## Gérer et assigner les rôles

### Rôles globaux

Rôle	Global	Identifiants					Agent							
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provis
Employee	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Rôle à ajouter

Ajouter

Illustration 6.5: Ajout d'un rôle global

Ici on a ajouté un rôle « Employee » auquel on a donné la permission de voir la plupart des pages du Jenkins (voir Illustration 6.5: Ajout d'un rôle global).

Job									
Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Illustration 6.6: Permissions pour les jobs

Et aussi la possibilité de voir les jobs et les codes utilisés dans les jobs (voir Illustration 6.6: Permissions pour les jobs).

Ensuite on ajoute un rôle de projets « développer » où les utilisateurs qui auront ce rôle pourront gérer entièrement un job (créer, supprimer, configurer, lire, etc.).

Le « **Patron** » représente la correspondance avec les noms de projets. C'est à dire que seulement les permissions associés à ce rôle ne seront valides que pour les projets dont le nom concorde avec ce patron.

On enregistre la configuration.

Pour finir on se rend dans « **Assigner les rôles** », on va donner à chaque utilisateurs un rôle. Ici on ajoute les identifiants d'utilisateurs auxquels on va assigner un rôle. On ajoute l'utilisateur « user1 », et on va lui donner le rôle « Employee » vu plus haut (voir Illustration 6.7: Assigner un rôle global).

## Assigner les rôles

### Rôles globaux

Utilisateur/groupe	Employee	admin	
 Administrateur	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
 User1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 Anonyme	<input type="checkbox"/>	<input type="checkbox"/>	

Utilisateur/groupe à ajouter

*Illustration 6.7: Assigner un rôle global*

Ensuite on peut assigner des rôles de projet, on va ajouter l'utilisateur « user1 » en tant que « developper » (voir Illustration 6.8: Assigner un rôle pour les projets).

### Item roles

Utilisateur/groupe	developper	tester	
 User1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 User2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
 Anonyme	<input type="checkbox"/>	<input type="checkbox"/>	

Utilisateur/groupe à ajouter

*Illustration 6.8: Assigner un rôle pour les projets*

## 6.3 Configuration du job PHP

Pour la configuration PHP utilisez Phing vu dans la partie 4.2, ici notre build va appeler PHP Lint avec les différentes versions PHP et PHP CodeSniffer (voir [build.xml](#)).

Utiliser Phing va permettre au build d'échouer si il y a des erreurs avec PHP Lint.

Si vous souhaitez obtenir un affichage de vos PHP Lint, vous pouvez installer le plugins : **HTML Publisher plugin**. Qui va publier un fichier html, pour le générer vous pouvez écrire par exemple un script qui va écrire dans un fichiers les balises html et les erreurs de vos PHP Lint (voir [report.sh](#)).

*Rappel pour appeler un script : **Build > Exécuter un script shell** .*

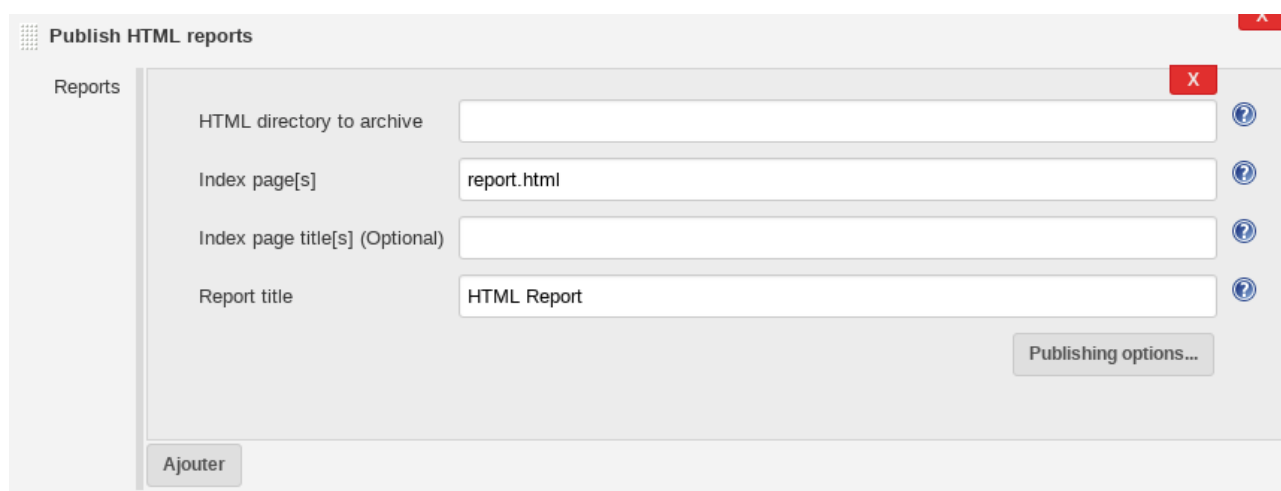


Pour PHP Lint, vous pouvez faire un script qui appelle l'exécutable `php -l` (voir [lint-5.4.sh](#) ;[lint-5.6.sh](#);[lint-7.0.sh](#)), et il vous suffira d'appeler ce script dans votre script pour le report (voir Illustration 6.9: Appel du script dans le report).

```
20      <h1>Result of PHP Lint analysis</h1>
21      <h2>lint 5.4 :</h2>
22      <p>' >> $file
23
24      bash script/lint-5.4.sh >> $file
25
26      echo ' </p>
```

*Illustration 6.9: Appel du script dans le report*

Ensuite pour publier votre report HTML dans Jenkins, ils vous faut configurer votre job, dans **Actions à la suite du build > Publish HTML reports**, rentrez le nom de votre report dans Index page[s] et vous pouvez aussi modifier le titre de votre report qui sera affiché sur Jenkins (voir Illustration 6.10: Plugin HTML Publisher plugin).



*Illustration 6.10: Plugin HTML Publisher plugin*

Sauvegardez votre configuration, et ensuite si vous buildez vous verrez apparaître un onglet HTML Report (titre que vous avez donnée) sur la gauche et un au centre de votre page (voir Illustration 6.11: Onglets HTML Report).

**Projet integration-ocs**  
Système d'intégration continue d'OCS Inventory

[HTML Report](#)  
[Espace de travail](#)  
[Changements récents](#)

**Liens permanents**

- [Dernier build \(#36\), il y a 6 j 16 h](#)
- [Dernier build stable \(#1\), il y a 1 mo 4 j](#)
- [Dernier build avec succès \(#1\), il y a 1 mo 4 j](#)
- [Dernier build en échec \(#36\), il y a 6 j 16 h](#)
- [Dernier build non réussi \(#36\), il y a 6 j 16 h](#)
- [Last completed build \(#36\), il y a 6 j 16 h](#)

Build	Status	Time
#36	Failed	12 juin 2018 15:44
#35	Failed	12 juin 2018 15:32
#34	Failed	12 juin 2018 15:06

*Illustration 6.11: Onglets HTML Report*

Si vous cliquez dessus vous aurez alors le résultat de votre page HTML contenant les résultats des scripts lint par exemple (voir Illustration 6.13: Résultats du HTML Report).

[Back to integration-ocs](#) [report](#)

## Result of PHP Lint analysis

lint 5.4 :

Fatal error: Can't use method return value in write context in ./require/search/TranslationSearch.php on line 218 Errors parsing ./require/search/TranslationSearch.php

*Illustration 6.12: Résultats du HTML Report*

Bien sûr vous pouvez ajouter plus d'outils pour vos fichiers PHP avec Phing, tels que PHPUnit, Xdebug, PHP Mess Detector, etc.

## 6.4 Configuration MySQL

Pour démarrer le serveur MySQL, tapez la commande suivante dans un terminal:

```
sudo service mysql start
```

Configurer un login-path, pour éviter les messages d'erreurs de connexion :mysql dans Jenkins:

[Warning] Using a password on the command line interface can be insecure.

```
mysql_config_editor set --login-path=jenkins --host=localhost --user=root --password
```

```
mysql --login-path=jenkins
```

Pour que cette commande se lance dans votre système jenkins, il vous faut d'abord lui donner la configuration et les droits.

Placez vous dans le home : `cd`

Ensuite copier la configuration des logins mysql dans le répertoire user de jenkins :

```
sudo cp .mylogin.cnf /var/lib/jenkins/ (/var/lib/jenkins/ étant le user.home)
```

Pour finir remplacer l'utilisateur (et le groupe) propriétaire du fichier config.

```
sudo chown jenkins:jenkins /var/lib/jenkins/.mylogin.cnf
```

Ainsi vous n'aurez pas d'erreurs pour vous connecter au mysql depuis votre système jenkins.

Pour effectuer des tests avec MySQL dans Jenkins vous pouvez écrire des scripts qui vous connecte à la base, et récupérer les erreurs de syntaxes par exemple. Voici un moyen d'appeler votre serveur mysql avec votre login-path configuré (voir au dessus) et en vous connectant à une base de test :

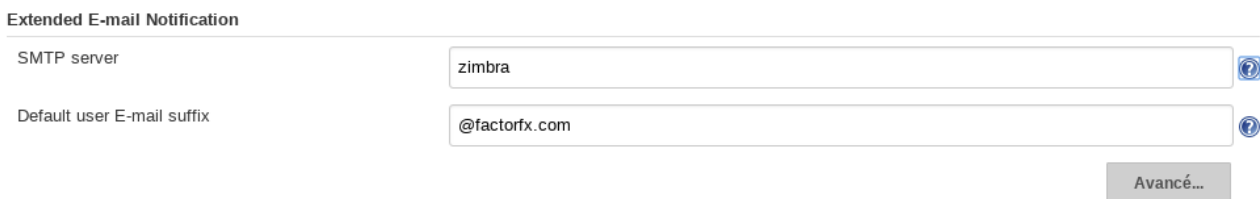
```
mysql --login-path=jenkins -D ocs_check_SQL_jenkins --force < $file 2>>$error
```

Avec cette commande, vous allez vous connecter sur le serveur mysql à la base de données `ocs_check_SQL_jenkins`, ce qui suit permet d'importer un fichier source à la base (par exemple un fichier update sql) et la commande `2>>$error` permet d'écrire les erreurs dans un fichiers erreur.

## 6.5 Notification par email

Installez le plugin **Extended E-mail Notification**, ce plugin vous permettra de configurer un envoi de mail lors des builds.

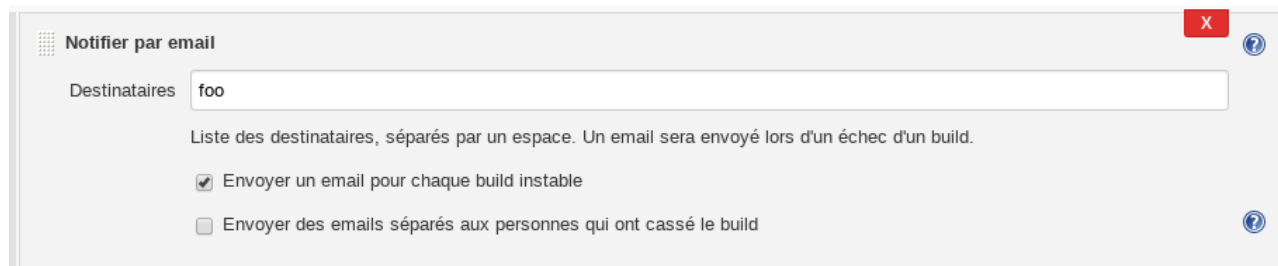
Tout d'abord rendez vous dans la section **Administrer Jenkins > Configurer le système > Extended E-mail Notification**, ici vous pouvez configurer le serveur mail sur lequel vous voulez envoyer les mails, le suffixe du E-mail par défaut, vous pourrez par exemple juste écrire le nom utilisateur (foo), et cela l'enverra à [foo@factorfx.com](mailto:foo@factorfx.com) (voir Illustration 6.13: Plugin Extended E-mail Notification).



*Illustration 6.13: Plugin Extended E-mail Notification*

De plus vous pouvez changer le sujet, le contenu du mail à votre guise (n'hésitez pas à lire les petites aides à droite). Pour l'instant on laisse tel quel, le mail nous donnera les infos sur un builds qui a échoué.

Maintenant pour que les projets envoient des mails aux utilisateurs, il vous faut configurer le projet, allez dans Actions à la suite du build > Notifier par email, et ajoutez les noms d'utilisateurs, quand un build aura échoué, il sera envoyé à cet utilisateur (voir Illustration 6.14: Notification par email).



*Illustration 6.14: Notification par email*

## 6.6 Configuration Selenium

Tout d'abord pour comprendre Selenium vous pouvez installer l'extension Chrome ou Firefox :**Katalon Automation Recorder**, elle vous permettra de créer des suites de tests qui vont simuler des actions sur une page web. Il vous suffit d'enregistrer faire vos actions, stopper et ensuite, vous pourriez rejouer ces tests, la simulation se fera toute seule.

Pourquoi installer cet outil ? Car vous pourrez importer les tests que vous effectués dans un langage que vous préférez parmi le java, python, C#, ruby, XML, Robot Framework. Ainsi vous pourrez ajouter ces tests dans vos codes. Ici nous utilisons JUnit une librairie de Java pour exécuter nos tests.

Il vous faut installer la librairie **Selenium Standalone Server** qui est nécessaire pour exécuter vos tests automatisés. Vous trouverez la dernière version sur le site suivant :

<https://www.seleniumhq.org/download/>

De plus vous devez avoir la librairie **Hamcrest Core**, téléchargez la avec cette commande :

**wget http://central.maven.org/maven2/org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3.jar**

Placez le dans **/usr/share/java/**. Ensuite il vous faut juste Java à jour et JUnit.

Enfin ajoutez chromedriver, et geckodriver dans **/usr/bin/**, voici les commandes pour les installer (n'hésitez pas à les mettre à jour) :

**wget https://chromedriver.storage.googleapis.com/2.40/chromedriver\_linux64.zip**

**wget https://github.com/mozilla/geckodriver/releases/download/v0.21.0/geckodriver-v0.21.0-linux64.tar.gz**

Ensuite il vous suffit d'extraire les fichiers dans les paquets.

Avec cette configuration vous pourrez exécuter vos tests Selenium sans aucun problème.

## 7 Tests automatisés dans OCS

### 7.1 Tests MySQL

Ici dans notre système Jenkins, on veut pouvoir faire des vérifications de syntaxes sur les fichiers sql et des vérifications des tables c'est à dire, voir si des tables utilisées dans un fichiers php sont bien présentes dans la bases. Pour cela dans chaque scripts de vérifications, je supprime la base de données de tests et je la recréer à partir du fichiers source de la base. Comme ça nous avons une base de données de tests clean (voir [mysql\\_database\\_test.sh](#)).

Ensuite j'appelle ce script dans mes scripts de vérifications pour enfin effectuer une vérification sur la base de données clean (voir [check\\_syntax\\_request.sh](#) ;[check\\_table.sh](#) ;[check\\_update.sh](#)).

Pour finir vous pouvez ajouter les résultats de vos scripts au report HTML de la même manière vu au dessus.

### 7.2 Tests Selenium

Pour exécuter vos tests Selenium avec JUnit, tout d'abord vous devez compiler vos fichiers (voir [ChromeTest.java](#) [FirefoxTest.java](#)) :

```
javac -cp ./usr/share/java/junit-4.12.jar:usr/share/java/hamcrest-core-1.3.jar:selenium-server-standalone-3.12.0.jar ChromeTest.java
```

```
javac -cp ./usr/share/java/junit-4.12.jar:usr/share/java/hamcrest-core-1.3.jar:selenium-server-standalone-3.12.0.jar FirefoxTest.java
```

Ensuite vous pouvez lancer les tests avec les commandes suivante :

```
java -cp ./usr/share/java/junit-4.12.jar:usr/share/java/hamcrest-core-1.3.jar:selenium-server-standalone-3.12.0.jar org.junit.runner.JUnitCore ChromeTest
```

```
java -cp ./usr/share/java/junit-4.12.jar:usr/share/java/hamcrest-core-1.3.jar:selenium-server-standalone-3.12.0.jar org.junit.runner.JUnitCore FirefoxTest
```

Les tests vont s'exécuter, mais sur un serveur ssh il n'y pas d'affichage pour Chrome ou Firefox, alors vous devez lancer vos webDriver en mode headless. Pour cela dans le langage de votre choix, vous devez définir les options au lancement du chromedriver ou geckodriver, en donnant en paramètre le mode headless.

Conseil : ajoutez les commandes ci-dessus dans un script shell (voir [run-test-selenium.sh](#))

**Exemple java :**

*Pour Chrome :*

```
ChromeOptions options = new ChromeOptions();  
options.addArguments("--headless");
```

Pour Firefox :

```
FirefoxOptions options = new FirefoxOptions();  
options.setHeadless(true);
```

Ainsi Chrome ou Firefox se lancera en mode headless, donc il n'y aura pas de problème lié à l'affichage.

Voici quelques commandes java utiles à connaître pour vos tests Selenium (elles sont assez similaire dans les autres langages) :

- `driver.get("http://172.18.26.211/ocsreports/");` : récupère l'url et lancera la page donnée en paramètre
- `driver.findElement(By.id("menu_settings")).click();` : cherche un élément html présent sur la page, vous pouvez chercher l'id ou le nom de classe ou le xpath par exemple, tout dépend de votre balise html
- `new Select(driver.findElement(By.name("MODE"))).selectByVisibleText("DEBUG");` : effectue une selection en cherchant l'élément html
- `WebElement element = driver.findElement(By.id("menu_settings"));` : créer un WebElement où l'on peut placer un élément html
- `WebDriverWait wait = new WebDriverWait(driver, 5);`  
`wait.until(ExpectedConditions.elementToBeClickable(By.linkText("Packages on servers groups"))).click();` : WebDriverWait permet d'attendre qu'un élément soit cliquable, ça permet d'éviter les problèmes lorsqu'un élément n'est pas encore trouvable

Maintenant on veut récupérer les erreurs ou warnings PHP, lors de la simulation avec Selenium, pour ça vous pouvez écrire un script qui récupère le fichier error.log de apache et il vous suffit de récupérer les PHP Warnings dans un fichiers temporaire, et vous l'afficher entre des balises html pour l'ajouter au HTML Report (voir [print-php-errors.sh](#)).

Pour que votre script shell s'exécute sans problème de permission car vous devez accéder à /var/log/apache2/error.log .Vous devez donner les droits d'exécutions de commandes à Jenkins pour cela vous devez modifier le fichier sudoers : `sudo nano /etc/sudoers`

Ajoutez : `jenkins ALL=(ALL) NOPASSWD: ALL` (voir Illustration 7.1: Donner toutes les permissions à un utilisateur)

```
# Allow members of group sudo to execute any command  
%sudo    ALL=(ALL:ALL) ALL  
jenkins  ALL=(ALL) NOPASSWD: ALL
```

Illustration 7.1: Donner toutes les permissions à un utilisateur

Enfin il vous suffit d'appeler vos scripts dans votre configuration du projet Jenkins.

### 7.3 Test installation du serveur OCS incluant les tests Selenium sur des serveurs SSH

Ce tests est incomplets mais je vous donne l'idée.

Pour pouvoir faire des tests Selenium automatisés avec Jenkins sur des serveurs où on installe le serveur OCS, vous pouvez créer un script qui installe tout ce que OCS requiert pour que le Jenkins installe le serveur OCS sur chaque serveur SSH. Ensuite vous installez avec ce script tout ce qui est nécessaire pour Selenium (voir install-ocs.sh). A la fin vous supprimez toutes les installations faites pour nettoyer votre serveur (voir remove-ocs.sh).

Il vous faut aussi importer la base de donnée contenant les machines de tests (voir la commande mysql dans install-ocs.sh).

Et le but étant de lancer le script bash d'installation avec le Jenkins mais sur les serveurs SSH, pour cela vous pouvez avec un script envoyer les fichiers de tests Selenium récupérés depuis le dépôt GitHub sur chaque serveurs (voir send-file.sh), ensuite vous lancez le tests Selenium comme vu ci-dessus.

Pour lancer une commande shell sur un serveur SSH depuis le Jenkins vous devez installer le plugin **SSH Plugin**, qui va vous permettre dans **Administrer Jenkins > Configurer le système > SSH remote hosts**, d'ajouter des serveurs SSH en ajoutant le **Hostname**, le **port** (généralement 22), le **User Name** et le **Password**.

Une fois la configuration terminé, dans la configuration de votre projet, vous pouvez définir une action pendant le build qui est **Execute shell script on remote host using ssh**, où vous pourrez exécuter vos scripts que vous avez envoyé.

Pour ce qu'il s'agit des tests Selenium, je pensais à l'idée de mettre **localhost** à la place de **172.18.26.211** dans la commande : `driver.get("http://172.18.26.211/ocsreports/");`

Pour récupérer l'application web lié à chaque serveurs.

Voir les scripts dans le dépôts suivant :

<https://github.com/nathanaeltouchard/testJenkins>

## 8 Conclusion

Normalement grâce à cette doc vous avez toutes les informations et petits conseils nécessaire pour mettre en place votre système Jenkins.

## 9 Annexes

**OCSInventory-ocsreports :**

<https://github.com/nathanaeltouchard/OCSInventory-ocsreports>

**build.xml :**

<https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/build.xml>

**lint-5.4.sh :**

<https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/lint-5.4.sh>

**lint-5.6.sh :**

<https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/lint-5.6.sh>

**lint-7.0.sh :**

<https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/lint-7.0.sh>

**check\_syntax\_request.sh :**

[https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/mysql\\_database\\_test.sh](https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/mysql_database_test.sh)

**check\_table.sh :**

[https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/check\\_table.sh](https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/check_table.sh)

**check\_update.sh :**

[https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/check\\_update.sh](https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/check_update.sh)

**mysql\_database\_test.sh :**

[https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/mysql\\_database\\_test.sh](https://github.com/nathanaeltouchard/OCSInventory-ocsreports/blob/master/script/mysql_database_test.sh)

**report.sh :**

<https://gitlab.factorfx.com/jenkins-ffx/test-selenium>

**ChromeTest.java**

<https://github.com/nathanaeltouchard/testJenkins/blob/master/ChromeTest.java>

**FirefoxTest.java**

<https://github.com/nathanaeltouchard/testJenkins/blob/master/FirefoxTest.java>

**print-php-errors.sh**

<https://github.com/nathanaeltouchard/testJenkins/blob/master/print-php-errors.sh>

**run-test-selenium.sh**

<https://github.com/nathanaeltouchard/testJenkins/blob/master/run-test-selenium.sh>



## 10 Bibliographies

**Voici les sources qui m'ont beaucoup aidé pendant ce projet :**

*Installation et configuration :*

<https://jenkins.io/doc/book/installing/>

<https://blog.pascal-martin.fr/post/integration-continue-jenkins-installation-configuration.html#installation-jenkins-ensuite>

*La suivante est très complète je vous la conseille si vous avez des problèmes :*

<https://jenkins-le-guide-complet.github.io/html/book.html>

*PHP :*

<https://blog.erlem.fr/programmation/developpement-web/php/72-php-integration-continue-avec-jenkins>

<https://fr.slideshare.net/hhamon/intgration-continue-des-projets-php-avec-jenkins>

<http://objis.com/tuto-selenium-php-jenkins/>

<https://www.phing.info/phing/guide/en/output/chunkhtml/index.html>

*Selenium :*

<http://www.automationtestinghub.com/selenium-headless-chrome-firefox/>

<https://blog.softhints.com/ubuntu-16-04-server-install-headless-google-chrome/>