

# JENKINS INSTALLATION

**Jenkins** est un outil open source d'intégration continue, fork de l'outil Hudson après les différents entre son auteur, Kohsuke Kawaguchi, et Oracle. Écrit en Java, Jenkins fonctionne dans un conteneur de servlets tels que Apache Tomcat ou en mode autonome avec son propre serveur Web embarqué.



## Objectif du document

L'objectif de ce document est de comprendre comment installer l'outil Jenkins, ainsi que construire un projet Jenkins. Le document sera réalisé dans l'optique de construire un projet sur la mise en place du site internet développé en PHP.

La présentation se fera en 5 étapes :

- Installation de Jenkins
- Installation des modules complémentaires
- Configuration de Jenkins
- Création d'un projet sur Jenkins.
- Lancement et analyse des résultats.

## Installation de Jenkins

Pour installer jenkins rien de plus simple. Il suffit de télécharger le package jenkins puis de l'installer

```
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
```

Après ces commandes, jenkins est correctement installé sur votre ordinateur.

**/\ Par défaut Jenkins s'installe et utilise le port 8080. Si celui-ci est déjà utilisé par une autre application, n'hésitez pas à modifier son port. Pour cela il suffit d'aller modifier le fichier **/etc/default/jenkins**, en remplaçant le port par défaut :**

HTTP\_PORT=8080 => HTTP\_PORT=8081

Et voilà ! Jenkins est installé (adresse : localhost:8080)

## Installation des modules complémentaires

Jenkins est maintenant installé, ce qui est sympa, cependant dans l'état actuel, Jenkins est simplement une coquille vide. Mais on va résoudre ce problème, en installant plusieurs modules qui vont nous permettre de construire une plateforme d'intégration complète.

Il faut savoir qu'aujourd'hui le projet qui nous intéresse est un projet PHP qui est hébergé sur github. On retrouve aussi à l'intérieur des tests PHPUnit.

L'objectif est donc d'installer plusieurs modules afin de répondre aux besoins du projet Pour cela nous allons installer :

- un module Github
- un module PHPUnit
- un module Sonar

Pour installer un module dans Jenkins il faut aller :

Manage Jenkins > Manage Plugins > Onglet Available

Puis il suffit de chercher grâce à la barre de recherche le module en question

---

### **MODULE GITHUB (GitHub+Plugin)**

Lien : <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+Plugin>

Ce module est important car il va permettre de télécharger à chaque build (lancement de l'ensemble des tâches définies dans un projet Jenkins) l'ensemble des sources se trouvant dans une branche qui sera définie lors de la création du projet.

### **MODULE PHPUnit (Clover+PHP+Plugin JENKINS/xUnit+Plugin)**

Lien : <https://wiki.jenkins-ci.org/display/JENKINS/Clover+PHP+Plugin>

Lien : <https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin>

Ces deux modules vont permettre d'exécuter les tests unitaires ainsi que l'affichage des résultats sur l'interface graphique.

### **MODULE SonarQube (SonarQube+plugin)**

Lien : <https://wiki.jenkins-ci.org/display/JENKINS/SonarQube+plugin>

Ce module va permettre de réaliser une analyse sur la qualité du code.

**Il faudra aussi installer Sonar sur le serveur (se référer au document sur l'installation de sonar)**

## Configuration de Jenkins

Maintenant que Jenkins est installé et que les plugins sont aussi installés, il faut configurer jenkins afin de pouvoir utiliser et créer un projet sans problème par la suite.

Pour cela, il faut pour commencer aller sur :

Manage Jenkins > Configure System

### Configuration de GIT

Modifier cette partie pour indiquer la position de git sur votre machine ou serveur.

**Global properties**

☐ Environment variables

☒ Tool Locations

List of tool locations

Name	(Git) Default	
Home	/usr/local/bin/git	<a href="#">Delete</a>

[Add](#)

**SonarQube servers**

### Configuration de SonarQube

**SonarQube servers**

Environment variables

☒ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	Sonar
Server URL	http://192.168.99.116:9000/
Server version	5.3 or higher
Server authentication token	*****
SonarQube account login	
SonarQube account password	

[Advanced...](#)

[Delete SonarQube](#)

Il faut remplir le formulaire avec les informations du serveur (URL, version de sonar) puis il faut donner le token permettant l'authentification. Pour récupérer ce token, il faut :

- aller sur Sonar
- se connecter en tant qu'administrateur.
- aller ensuite dans administration > Security > User
- Puis récupérer le token qui sera sous la forme 2de6755cb8cacb3c7941a736b089bb4a52c27539

Pour finir il faut aller dans :

Manage Jenkins > Global Tool Configuration

## Configuration de SonarQube Scanner

**SonarQube Scanner**

SonarQube Scanner installations

SonarQube Scanner

Name

SonarQube Scanner 2.8

☒ Install automatically

Install from Maven Central

Version

SonarQube Scanner 2.8

Add Installer

Delete Installer

Delete SonarQube Scanner

Il faut ajouter un scanner, pour cela cliquer sur le bouton “SonarQube Scanner installations...” qui vous ouvrira l’interface ci-dessus. A partir de là, il suffira de rajouter un SonarQube avec les mêmes informations que l’image ci-dessus. Ceci permettra de réaliser des actions sur sonar à partir de Jenkins.

## Création d'un projet sur Jenkins.

C'est bon, tout est prêt, on va pouvoir enfin créer le projet Jenkins et le configurer !!

La création et la configuration d'un projet se fait en plusieurs étapes. Les différentes étapes suivantes vous montreront comment créer un projet autour de notre fil rouge à savoir le site PHP.

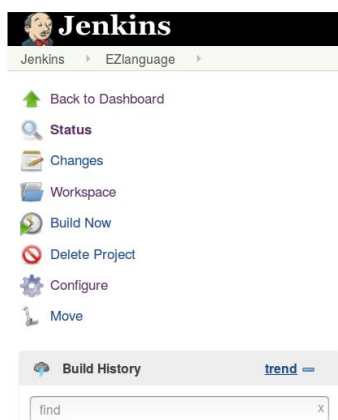
### Etape 1 : Création du projet

Aller sur la page d'accueil de jenkins (localhost:8080) puis cliquez dans le menu sur le lien "New Item" qui va permettre la création d'un Job (un job correspond à un projet).

Pour la création de ce projet, nous allons partir sur un projet "**Freestyle project**" afin d'avoir une liberté totale dans la création du projet.

Vous pouvez voir qu'un projet s'est ajouté sur votre page d'accueil de Jenkins.

### Etape 2 : Configuration du projet



Cliquer sur le projet qui se trouve sur votre page d'accueil, vous allez arriver sur la page d'accueil de votre projet avec à gauche un menu qui va vous permettre d'avoir accès aux informations du projet.

Le build History permettra par la suite d'avoir un historique de tous les builds (lancement des actions définies sur un projet).

Pour configurer le projet il suffit d'aller sur l'élément CONFIGURE du menu.

The screenshot shows the 'General' tab of the Jenkins configuration interface. At the top, there are tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is active. It contains a 'Project name' field with the value 'EZlanguage' and a 'Description' text area. Below the description area, there is a link that says '[Plain text] Preview'.

## -> GIT :

Pour la configuration de git, c'est assez simple, il suffit de rajouter l'URL du dépôt sur votre page de configuration. Cela permettra d'avoir un raccourci sur le projet (page Jenkins)

This screenshot shows the 'GitHub project' configuration section. It has a checkbox labeled 'GitHub project' which is checked. Below it is a 'Project url' field containing the text 'https://github.com/eZlanguage/website/'. To the right of the field is a help icon. At the bottom right of this section is a button labeled 'Advanced...'.

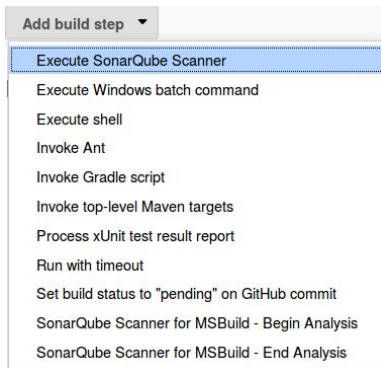
Puis il faut configurer git pour créer un lien entre le dépôt git et jenkins pour cela il faut remplir le formulaire ci-dessous avec les informations propre du repository.

The screenshot shows the 'Source Code Management' configuration page. At the top, there are radio buttons for 'None' and 'Git', with 'Git' being selected. Below this is a section for 'Repositories'. It contains a 'Repository URL' field with the value 'https://github.com/eZlanguage/website.git', a 'Credentials' dropdown menu set to '- none -' with an 'Add' button next to it, and buttons for 'Advanced...' and 'Add Repository'. Below the 'Repositories' section is a 'Branches to build' section with a 'Branch Specifier (blank for 'any')' field containing '\*/master' and an 'Add Branch' button. Below that is a 'Repository browser' dropdown menu set to '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button and a radio button for 'Subversion'.

Le choix de la "Branches to build" va permettre de choisir sur quelle branche sera réalisé le build. Cela permet par exemple de lancer une série de tests sur une branche pas encore mergée avec le master.

Il sera ensuite possible de rajouter des options pour, par exemple exécuter un build à chaque commit sur la branche principale (définie dans la configuration).

## -> Sonar



Il faut ajouter une tâche “Execute SonarQube Scanner”. cela va ouvrir un formulaire qui devra être rempli avec les informations futures du projet.

Pour ajouter un projet Sonar, il suffit de rajouter les propriétés du projet (à définir soi-même) .

La propriété projectKey et projectName sont à définir par vous même et correspondront par la suite au lien entre Jenkins et Sonar (id du projet).

**Build**

Execute SonarQube Scanner

Task to run

JDK

JDK to be used for this SonarQube analysis

Path to project properties

Analysis properties

Additional arguments

JVM Options

(Inherit From Job)

#Required metadata  
sonar.projectKey=EZlanguagePHP2  
sonar.projectName=EZlanguagePHP  
sonar.projectVersion=1.\$BUILD\_NUMBER  
sonar.language=php  
sonar.sources=src  
  
sonar.sourceEncoding=UTF-8

Execute SonarQube Scanner

Task to run

JDK

JDK to be used for this SonarQube analysis

Path to project properties

Analysis properties

Additional arguments

JVM Options

(Inherit From Job)

#Required metadata  
sonar.projectKey=EZlanguageHTML  
sonar.projectName=EZlanguageHTML  
sonar.projectVersion=1.\$BUILD\_NUMBER  
sonar.language=web  
sonar.sources=src  
  
sonar.sourceEncoding=UTF-8

Save

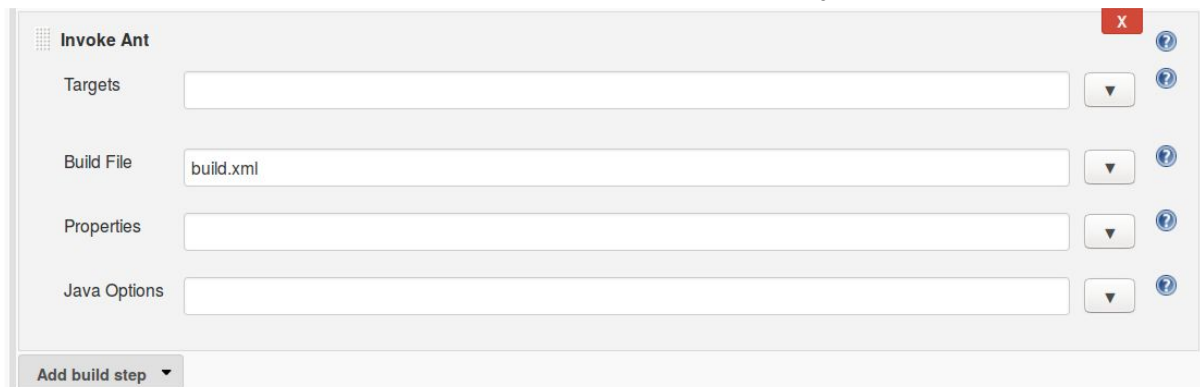
Apply

Dans la configuration proposée ci-dessus, nous avons configuré deux projets pour pouvoir gérer sonar autour de plusieurs langages : le premier sera une analyse du PHP et le deuxième sera une analyse du Web (HTML). Pour les sources nous avons stipulé “src” car dans un projet Symfony seul le dossier “src” contient des sources pertinentes à analyser (cela permet de ne pas regarder tous les fichiers PHP qui sont liés au framework).



## -> PHPUnit

Pour configurer sur le projet PHPUnit il faut réaliser plusieurs tâches dans un premier temps il faut ajouter dans le Build, la tâche Invoke Ant. Cette tâche va permettre d'exécuter Ant à partir d'un fichier xml (build.xml) qui se trouve à la racine du projet (code source)

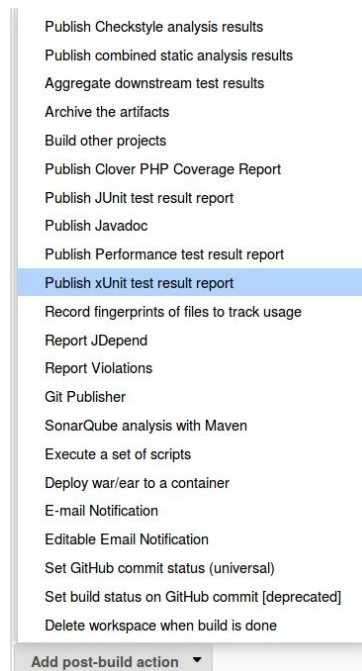


Cela suffit pour lancer les tests PHPUnit, mais il faut quand même pour cela construire et remplir le fichier build.xml qui va permettre de réaliser différentes tâches. Voici l'exemple du fichier build.xml utilisé.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <project name="EZlanguage" default="build">
4   <property name="workspace" value="${basedir}" />
5   <property name="sourcedir" value="${basedir}/src" />
6   <property name="builddir" value="${workspace}/app/build" />
7
8   <target name="build" depends="prepare,composer,phpunit"/>
9
10  <target name="clean" description="Cleanup build artifacts">
11    <delete dir="${builddir}/logs"/>
12    <delete dir="${builddir}/docs/*"/>
13  </target>
14
15  <target name="prepare" depends="clean" description="Prepare for build">
16    <mkdir dir="${builddir}/logs"/>
17  </target>
18
19  <target name="composer" description="Installing composer dependencies">
20    <exec executable="composer" failonerror="true">
21      <arg value="install" />
22    </exec>
23  </target>
24
25  <target name="phpunit" description="Run unit tests with PHPUnit">
26    <exec executable="phpunit" failonerror="true">
27      <arg value="-c" />
28      <arg path="${basedir}/app/phpunit.xml" />
29    </exec>
30  </target>
31
32 </project>
```

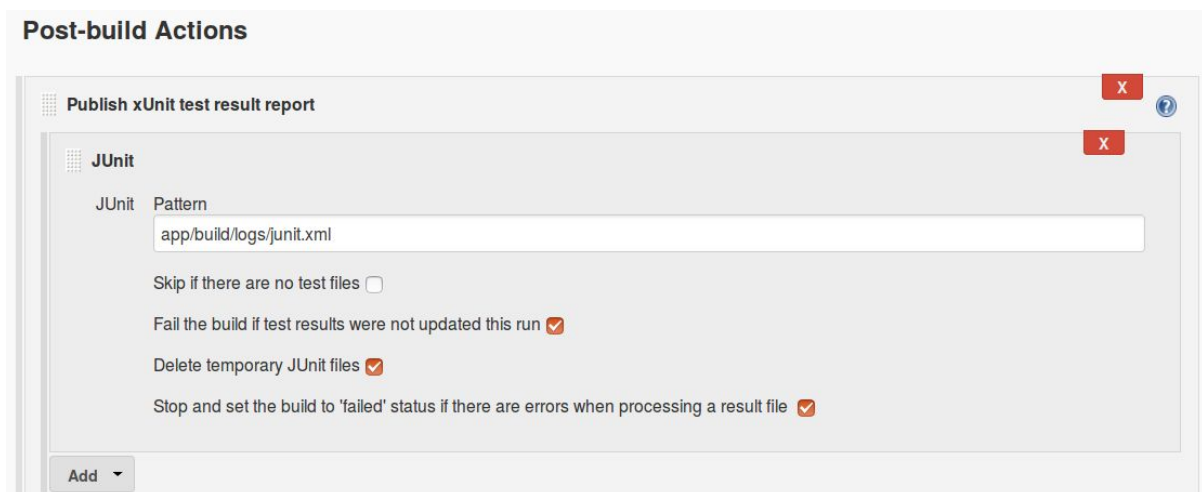
Donc ici on va dans un premier temps supprimer certains dossiers (tâche clean) puis créer un dossier logs (logs) et pour finir la tâche phpunit va exécuter les tests à partir du fichier

phpunit.xml qui est donné par défaut par Symfony2 (il va indiquer où se trouvent tous les fichiers de test)



Maintenant que tout cela est fonctionnel, il est intéressant de réaliser des rapports sur les exécutions des tests afin d'avoir une idée plus précise. Pour cela il faut rajouter une action "post-build"

Cela va ajouter un formulaire, qui va permettre de stipuler le lien du fichier où se trouvent les logs (lien se trouvant dans le fichier build.xml).



Cette fonctionnalité va permettre d'avoir de jolis rapports (cf à la partie suivante)

# Résultats.

## -> Jenkins



Ici, nous sommes sur la page d'accueil du projet EZLanguage, dont on retrouve à droite un résumé des résultats des tests (PHPUnit). On retrouve en bas à gauche, la liste des builds réalisés. On retrouve aussi au centre de la page, deux icônes de sonar qui correspondent aux deux projets que l'on a rajouté durant la configuration de jenkins-sonar.

## -> Rapport PHP Unit

### Test Result



### All Failed Tests

Test Name	Duration	Age
<a href="#">AppBundle\Tests\Controller\DefaultControllerTest::testThatIndexIsSuccessful</a>	0.77 sec	1
<a href="#">AppBundle\Tests\Controller\DefaultControllerTest::testThatFrenchIndexIsSuccessful</a>	0.52 sec	1
<a href="#">AppBundle\Tests\Controller\DefaultControllerTest::testFormContactShouldValidate</a>	19 ms	1
<a href="#">AppBundle\Tests\Controller\DefaultControllerTest::testFormContactShouldNotValidateWithIncorrectEmail</a>	10 ms	1
<a href="#">AppBundle\Tests\Controller\DefaultControllerTest::testFrenchFormContact</a>	10 ms	1

### All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
<a href="#">(root)</a>	1.5 sec	5	+5	0		4	+4	9	+9

Sur le tableau de bord on peut voir un résumé des différents tests (voir ci-dessus) mais il possible de voir en détail les tests qui ont échoué et réussi.

## -> Sonar

The screenshot displays the SonarQube web interface. At the top, there is a navigation bar with links to 'sonarqube', 'Dashboards', 'Issues', 'Measures', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. A 'Home' button is located on the right side of the navigation bar. Below the navigation bar, the main content area is titled 'PROJECTS'. It features a table with columns: 'QG', 'Name', 'Version', 'LOC', 'Bugs', 'Vulnerabilities', 'Code Smells', and 'Last Analysis'. Two projects are listed: 'EZLanguageHTML' and 'EZLanguagePHP'. Below the table, there is a section for 'EZLANGUAGEPHP' showing detailed metrics: 'Complexity' (166), 'Comments (%)' (21.0%), 'Issues' (26), and 'Lines' (2,159). A 'Quality Gate Status' is shown with a green checkmark icon.

QG	Name	Version	LOC	Bugs	Vulnerabilities	Code Smells	Last Analysis
★	EZLanguageHTML	1.38	508	0	0	0	Dec 07 2016
★	EZLanguagePHP	1.38	1,094	0	0	26	Dec 07 2016

2 results

**EZLANGUAGEPHP**

Complexity	Comments (%)	Issues	Lines
166	21.0%	26	2,159

Quality Gate Status

✓

On peut voir les deux projets créés avec des informations sur le code et des indicateurs de qualité

Lien pour ajouter des permissions permettant la configuration de Symfony 2. Et la mise en place d'un déploiement par jenkins

Permission dossier cache et log

[http://symfony.com/doc/2.8/setup/file\\_permissions.html](http://symfony.com/doc/2.8/setup/file_permissions.html)