

# Installation et Configuration du serveur d'intégration Jenkins

Version 1.0

**Auteur**  
Laverriere Celim  
*Analyste-programmeur*

# TABLE DES MATIERES

<b>1. Versions</b>	<b>3</b>
<b>2. Introduction</b>	<b>4</b>
2.1 - Objet du document	4
<b>3. Partie 1 - Pré-requis</b>	<b>5</b>
3.1 - Prise de contrôle à distance du serveur	5
3.1.1 - Installation de MobaXterm	5
3.1.2 - Comment accéder au serveur ?	5
3.2 - Installer Java - jdk-1.8.0_222	7
3.3 - Installer Jenkins	7
3.4 - Installer docker-compose	7
3.5 - Installer Maven	7
<b>4. Partie 2 : configurer Jenkins</b>	<b>9</b>
4.1 - Démarrer Jenkins	9
4.2 - Télécharger les plugins manquants	10
4.2.1 - Maven Integration plugin	10
4.2.2 - JaCoCo plugin	10
4.3 - Configuration globale des outils	11
4.3.1 - JDK	11
4.3.2 - Maven	11
4.3.3 - Docker	11
<b>5. Créez un « Nouveau Item »</b>	<b>12</b>
5.1 - Nouveau Item	12
5.2 - Configurer un « Nouveau Item »	12
5.2.1 - Onglet « General »	12
5.2.2 - Onglet « Gestion de code source »	13
5.2.3 - Cloner le dépôt github	13
5.2.4 - Onglet « Ce qui déclenche le build »	13
5.2.5 - Onglet « Pre Steps »	14
5.2.6 - Onglet « Build »	14
5.2.7 - Onglet « Post Steps »	14
5.2.8 - Onglet « Actions à la suite du build »	14
5.3 - Lancez un build	15

# 1. VERSIONS

Auteur	Date	Description	Version
Laverriere Celim	21/11/2019	Cette première version constitue le dossier qui explique comment installer et configurer le serveur d'intégration Jenkins pour le projet MyERP.	1.0

## 2. INTRODUCTION

### 2.1 - Objet du document

Vous trouverez dans ce document les prérogatives à suivre pour l'installation et la configuration du serveur Jenkins pour le projet MyERP sur un serveur dédié loué chez OVH.

Dans une première partie, vous trouverez la distribution à installer sur le serveur et les programmes utiles au fonctionnement de Jenkins.

Puis, dans une deuxième partie, vous trouverez les indications à suivre pour configurer Jenkins et créer un nouveau job pour le projet MyERP.

## 3. PARTIE 1 - PRE-REQUIS

Le serveur d'intégration continue Jenkins sera installé sur un serveur dédié, fourni par OVH parmi leur gamme "Advance", dont vous trouverez les caractéristiques ci-après :

- Gamme : Serveur dédié Advance-1.
- Processeur : Intel Xeon-D 2123IT - 4c/ 8t - 2.2GHz/ 3GHz.
- Mémoire : 32Go DDR4 ECC 2400MHz.
- Disques : 500Go SSD NVMe Soft RAID.
- Réseau public : 1Gbit/s illimité.
- Réseau privé : 100Mbit/s.
- Distribution : Ubuntu Server 18.04 LTS.

La distribution proposée par OVH ne prend pas en charge l'installation de Java, il faudra donc procéder à l'installation de celui-ci et des différents logiciels indispensables pour le fonctionnement du serveur d'intégration.

### 3.1 - Prise de contrôle à distance du serveur

Pour la prise de contrôle à distance du serveur utilisez le logiciel « MobaXterm ».

#### 3.1.1 - Installation de MobaXterm

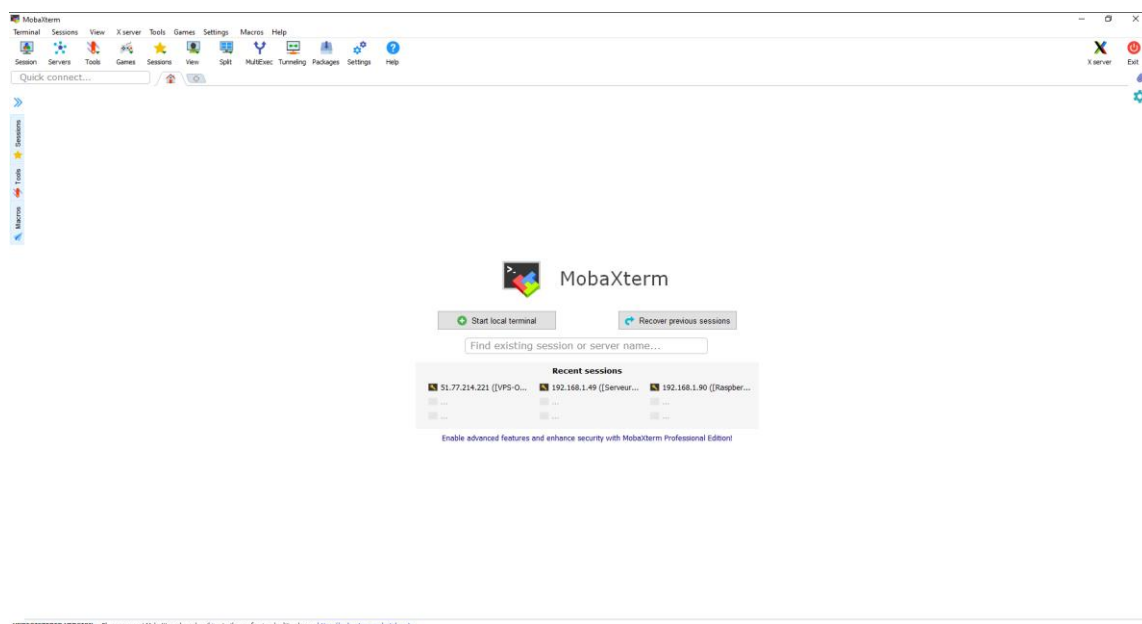
Pour télécharger MobaXterm rendez-vous sur le lien suivant (Récupérer la version portable) :

<https://mobaxterm.mobatek.net/download-home-edition.html>

Aucune installation n'est requise !

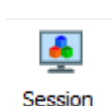
#### 3.1.2 - Comment accéder au serveur ?

À présent, lancez le logiciel !



Paramètres d'accès et compte administrateur :

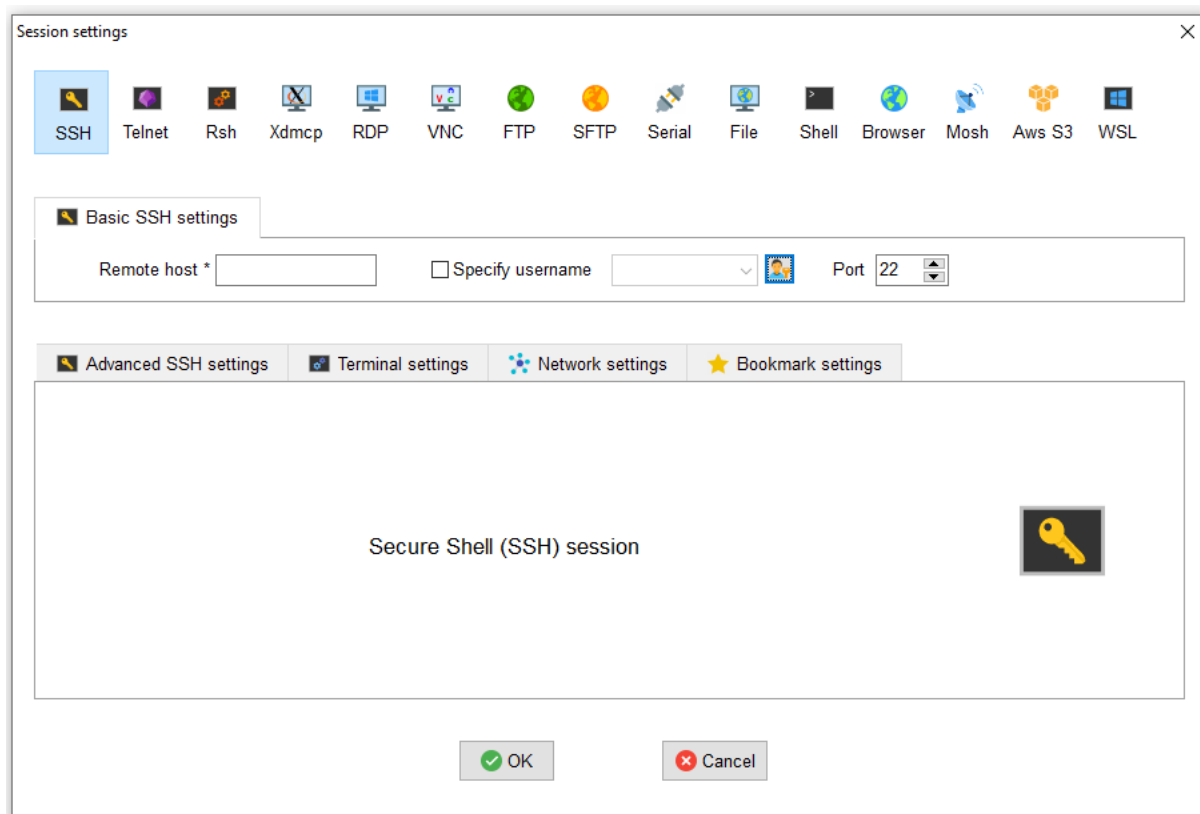
Rendez-vous dans Session




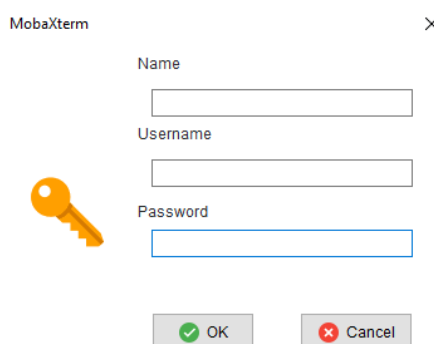
puis sélectionnez SSH



la page suivante s'affiche :



Configurez en premier l'username en cliquant sur le bouton suivant  puis sélectionnez New pour créer un nouvel utilisateur :



- Name : MyERP-SERVEUR-OVH
- Username: root
- Password: fr54Ze9

Une fois « username » créé, renseignez dans « Remote host » l'adresse IPv4 du serveur, puis coché « Specify username » et sélectionnez celui que l'on vient de créer. La connexion à votre serveur est créée, vous pouvez à présent vous y connecter.

### 3.2 - Installer Java - jdk-1.8.0\_222

Ajoutez le repository openjdk :

```
sudo add-apt-repository ppa :openjdk-r/ppa
```

Mettez à jour le repository :

```
sudo apt-get update
```

Puis entrez dans la console la commande suivante pour installer le jdk :

```
sudo apt-get install openjdk-8-jdk
```

### 3.3 - Installer Jenkins

Commencez par exécuter les commandes ci-dessous pour ajouter le référentiel Jenkins à votre système.

Ajoutez la clé du référentiel :

```
cd /tmp && wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key  
| sudo apt-key add -
```

Ensuite, lancez les commandes ci-dessous pour ajouter le référentiel :

```
echo 'deb https://pkg.jenkins.io/debian-stable binary/' | sudo tee -a  
/etc/apt/sources.list.d/jenkins.list
```

Mettez à jour le référentiel et installez Jenkins :

```
sudo apt-get update  
sudo apt-get install jenkins
```

### 3.4 - Installer docker-compose

Entrez la commande suivante pour installer docker-compose :

```
sudo apt-get install docker-compose
```

Autorisez Jenkins à exécuter des commandes docker-compose en exécutant la commande suivante dans la console :

```
sudo usermod -aG docker jenkins
```

Puis, redémarrer Jenkin :

```
sudo service jenkins restart
```

### 3.5 - Installer Maven

Commencez par vous rendre le répertoire opt/ :

```
cd opt/
```

Téléchargez la dernière version stable d'Apache Maven sur le site officiel :

```
sudo wget http://apache.crihan.fr/dist/maven/maven-3/3.6.2/binaries/apache-  
maven-3.6.2-bin.tar.gz
```

Une fois le téléchargement terminé, extrayez l'archive téléchargée :

```
sudo tar -xvzf apache-maven-3.6.2-bin.tar.gz
```

Ensuite, renommez le répertoire extrait :

```
sudo mv apache-maven-3.6.2 maven
```

Configurez la variable d'environnement Maven :

```
sudo nano /etc/environment
```

Indiquez le chemin du bin de Maven comme l'exemple ci-après :

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/opt/maven/bin"
```



## 4. PARTIE 2 : CONFIGURER JENKINS

### 4.1 - Démarrer Jenkins

Connectez-vous à Jenkins en entrant dans votre navigateur l'IPv4 de votre serveur suivi de « :8080 ».

La page suivante va s'afficher, vous demandant d'aller chercher le mot de passe généré dans initialAdminPassword :



Entrez dans la console la commande suivante pour récupérer le mot de passe :

```
sudo nano /var/lib/jenkins/secrets/initialAdminPassword
```

Puis, sélectionnez « installer les plugins suggérés » !

Une fois les plugins installés, créez un utilisateur puis sauvegardez et continuez :



## 4.2 - Télécharger les plugins manquants

### 4.2.1 - Maven Integration plugin

Rendez-vous dans « *Administrer Jenkins* » puis « *Gestion des plugins* » et pour finir sélectionnez l'onglet « *Disponibles* » et saisissez dans la barre de recherche « *Maven Integration* ».

Cochez le plugin et faites « *Installer sans redémarrer* ».

Puis cochez « *Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours* ».

### 4.2.2 - JaCoCo plugin

A présent installez JaCoCo pour l'analyser la couverture du code pour mesurer le nombre de lignes de notre code exécutées lors de tests automatisés.

Pour installer le plugin « *JaCoCo* » suivez la même procédure que pour « *Maven Integration* »

## 4.3 - Configuration globale des outils

A présent, vous devez configurer Jenkins, pour ce faire rendez-vous dans « *Administrer Jenkins* » puis « *Configuration globale des outils* ».

### 4.3.1 - JDK

Cliquez sur « *Ajouter JDK* » et décochez « *Install automatically* » et pour finir renseignez les champs comme suit :

Nom : JAVA\_HOME

JAVA\_HOME : /usr/lib/jvm/java-8-openjdk-amd64

JDK

Installations JDK

Ajouter JDK

JDK

Nom

JAVA\_HOME

JAVA\_HOME

/usr/lib/jvm/java-8-openjdk-amd64

☐ Install automatically

Supprimer JDK

Ajouter JDK

Liste des installations JDK sur ce système

### 4.3.2 - Maven

Cliquez sur « *Ajouter Maven* » et décochez « *Install automatically* » et pour finir renseignez les champs comme suit :

Nom : MAVEN\_HOME

MAVEN\_HOME : /opt/maven

Maven

Installations Maven

Ajouter Maven

Maven

Nom

MAVEN\_HOME

MAVEN\_HOME

/opt/maven

☐ Install automatically

Supprimer Maven

Ajouter Maven

Liste des installations Maven sur ce système

### 4.3.3 - Docker

Cliquez sur « *Ajouter Docker* » et décochez « *Install automatically* » et pour finir renseignez les champs comme suit :

Name : docker

Installation root : /var/lib/docker

Docker

Installations Docker

Ajouter Docker

Docker

Name

docker

Installation root

/var/lib/docker

☐ Install automatically

Supprimer Docker

Ajouter Docker

Liste des installations Docker sur ce système

Pour finir faites, « *Appliquer* » et « *Enregistrer* » !








## 5. CREEZ UN « NOUVEAU ITEM »

### 5.1 - Nouveau Item


Cliquez sur « *nouveau item* » :

Saisissez un nom

» Ce champ ne peut pas être vide. Veuillez saisir un nom valide et appuyer sur OK.

-  **Construire un projet free-style**  
Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.
-  **Construire un projet maven**  
Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.
-  **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **Construire un projet multi-configuration**  
Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.
-  **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
-  **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
-  **Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Si vous voulez créer un nouvel élément à partir d'un autre, vous pouvez utiliser cette option :

 Copier depuis

OK

Nommez le projet, par exemple : `Projet_B4_FR`.

Puis sélectionnez « *Construire un projet maven* » et faites « *OK* ».

### 5.2 - Configurer un « Nouveau Item »

#### 5.2.1 - Onglet « General »

Cochez « *GitHub project* » puis renseignez l'url du projet :

Project url : <https://github.com/Celim-Laverriere/OC-PROJET-9-TESTEZ-VOS-DEVELOPPEMENTS-JAVA/>

☒ GitHub project

Project url

Avancé...

### 5.2.2 - Onglet « Gestion de code source »

Cochez « *Git* » puis renseignez url du dépôt du projet :

Repository URL : <https://github.com/Celim-Laverriere/OC-PROJET-9-TESTEZ-VOS-DEVELOPPEMENTS-JAVA.git>

### 5.2.3 - Cloner le dépôt github

Avant de poursuivre, vous allez cloner le projet sur le serveur.

Faites « *Apply* » et « *Sauver* » puis rendez-vous dans « *Projet\_B4\_FR* » et lancez un build avec « *Lancer un build* » pour importer le projet.

Si tout se passe bien vous verrez dans « *Historique des builds* » le build avec une pastille bleue.

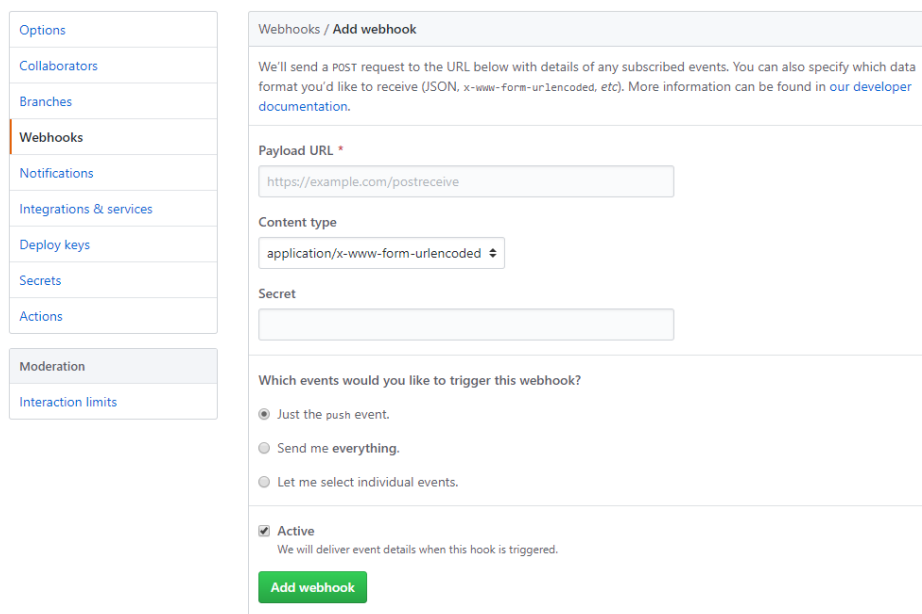
Vous pouvez à présent retourner dans la partie configuration du projet.

### 5.2.4 - Onglet « Ce qui déclenche le build »

Cochez « *GitHub hook trigger for GITScm polling* ».

Le build sera déclenché à chaque fois qu'un commit sera poussé sur le dépôt github il faudra donc configurer notre dépôt github comme suit :

Rendez-vous dans le dépôt github du projet et allez dans « *Settings* » puis sélectionnez « *Webhooks* » puis « *Add webhook* », la page suivante s'affichera :



Dans « *Payload URL* » indiquez l'url de votre serveur où est installé Jenkins suivie de « */github-webhook/* ».

Exemple : <http://XX.XX.XXX.XXX:8080/github-webhook/>

Dans « *Content type* » sélectionnez « *application/json* » et pour finir faites « *Add webhook* ».

Le déclenchement automatique du build est configuré !

### 5.2.5 - Onglet « Pre Steps »

Configurez « *Pre Steps* » pour qu'il crée à partir de docker-compose la base de données.

Dans la liste déroulante « *Ajouter une étape pré-build* » sélectionnez « *Exécuter un script shell* » puis entrez les commandes suivantes :

```
#!/bin/bash  
cd /var/lib/jenkins/workspace/test/projet_B4_FR/docker/dev  
sudo docker-compose up -d
```

### 5.2.6 - Onglet « Build »

Renseignez les informations suivantes :

POM Racine : projet\_B4\_FR/src/pom.xml

Goals et options : clean test -P test-business,test-consumer

### 5.2.7 - Onglet « Post Steps »

Configurez « *Post Steps* » pour qu'il supprime la base de données.

Dans la liste déroulante « *Ajouter une étape pré-build* » sélectionnez « *Exécuter un script shell* » puis entrez les commandes suivantes :

```
cd /var/lib/jenkins/workspace/test/projet_B4_FR/docker/dev  
sudo docker-compose dow
```

### 5.2.8 - Onglet « Actions à la suite du build »

Sélectionnez dans la liste déroulante « Record JaCoCo coverage report », et c'est tout !

Pour que JaCoCo puisse fonctionner il faut ajouter au pom.xml du projet la dépendance suivante :

```
<!-- https://mvnrepository.com/artifact/org.jacoco/jacoco-maven-  
plugin -->  
<dependency>  
  <groupId>org.jacoco</groupId>  
  <artifactId>jacoco-maven-plugin</artifactId>  
  <version>0.8.5</version>  
</dependency>
```

Et le plugin suivant dans les modules où l'on veut que JaCoCo analyse la couverture du code :

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.8.5</version>
  <executions>
    <execution>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
    </execution>
    <!-- attached to Maven test phase -->
    <execution>
      <id>report</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

## 5.3 - Lancez un build

Pour finir, comme précédemment dans le « 5.2.3 - Cloner le dépôt *github* », lancez un build !