

# Configuration du Serveur d'Intégration – Jenkins

## Table des matières

Introduction.....	2
Pré-requis.....	2
Information système.....	2
Système d'exploitation.....	2
Docker.....	2
Docker Compose.....	2
Java.....	2
Maven.....	2
Jenkins.....	2
Procédures d'installation.....	3
Clone du projet GitHub MyERP.....	3
Installation de Docker.....	3
Installation de Docker Compose.....	4
Installation de Java.....	5
Installation de Jenkins.....	5
Procédures de configuration.....	5
Création de docker-compose.yml.....	5
Lancement et configuration de Jenkins.....	5
Configuration de pipeline Jenkins.....	7
Initialisation et configuration de Webhook Relay.....	9
Installation de la CLI.....	9
Inscription et authentification.....	9
Configuration de Webhooks Github.....	10
Test !.....	11

Auteur	Date	Description	Version
Benoît Nérin	23/12/2020	Initialisation du dossier de configuration	1.0

## Introduction

---

Le document présente la configuration du serveur d'intégration Jenkins via Docker Compose qui permet d'implémenter et automatiser les tests de l'application MyERP. Les tests sont lancés via un build Maven à chaque commit réalisé sur le dépôt Github.

## Pré-requis

---

### Information système

Mémoire : 11,1 GiB

Processeur : Intel® Core™ i5-6300U CPU @ 2.40GHz × 4

Carte graphique : Mesa Intel® HD Graphics 520 (SKL GT2)

Capacité du disque : 256,1 GB

### Système d'exploitation

Nom du système d'exploitation : Pop!\_OS 20.10 Linux

Type de système d'exploitation : 64 bits

Version de GNOME : 3.38.1

Système de fenêtrage : X11

### Docker

Docker version 19.03.13

### Docker Compose

docker-compose version 1.25.0

### Java

OpenJDK version 1.8.0\_275

OpenJDK Runtime Environment

OpenJDK 64-Bit Server VM

### Maven

Maven version 3.6.3

### Jenkins

Jenkins version 2.264

## Procédures d'installation

---

### Clone du projet GitHub MyERP

```
$ sudo git clone https://github.com/BenoitNE/projet_B4_FR.git
```

Créez un nouveau repository du projet dans votre session GitHub.

### Installation de Docker

<https://devimalplanet.com/how-to-install-docker-on-linux-pop-os>

1. Assurez-vous qu'aucune installation de docker précédente n'existe dans le système :

```
$ sudo apt-get remove docker docker-engine docker.io containerd  
runc
```

2. Mettez à jour les packages apt :

```
$ sudo apt-get update
```

3. Activez HTTPS pour apt :

```
$ sudo apt-get install apt-transport-https ca-certificates curl  
gnupg-agent software-properties-common
```

4. Ajoutez la clé d'identification du Docker :

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg \  
| sudo apt-key add - \  
&& sudo apt-key fingerprint 0EBFCD88
```

```
OK  
pub rsa4096 2017-02-22 [SCEA]  
9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88  
uid [ unknown] Docker Release (CE deb) <docker@docker.com>  
sub rsa4096 2017-02-22 [S]
```

5. Ajoutez le référentiel de version stables à votre gestionnaire de packages :

```
$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"

$ sudo apt-get update && sudo apt-get install docker-ce docker-ce-
cli containerd.io
```

7. Vérifiez que l'installation s'est terminée correctement :

```
$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally

[... some installation lines]

Hello from Docker!
This message shows that your installation appears to be working
correctly.

[... some more lines]
```

## Installation de Docker Compose

<https://docs.docker.com/compose/install/>

1. Téléchargez la version stable actuelle de Docker Compose :

```
$ sudo curl -L
"https://github.com/docker/compose/releases/download/1.27.4/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Appliquez les autorisations exécutables au binaire :

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

3. Testez l'installation :

```
$ docker-compose --version
docker-compose version 1.27.4, build 1110ad01
```

## Installation de Java

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

```
$ sudo apt search openjdk
$ sudo apt install openjdk-8-jdk
$ java -version

openjdk version "1.8.0_275"
OpenJDK Runtime Environment (build 1.8.0_275-8u275-b01-
0ubuntu1~20.10-b01)
OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)
```

## Procédures de configuration

---

### Création de docker-compose.yml

Créez avec un éditeur de texte le fichier docker-compose.yml :

```
version: '2'
services:
  jenkins:
    image: 'jenkins/jenkins:lts'
    labels:
      kompose.service.type: nodeport
    ports:
      - "127.0.0.2:8080:5432"
    volumes:
      - 'jenkins_data:/jenkins_config'
volumes:
  jenkins_data:
    driver: local
```

### Lancement et configuration de Jenkins

Accédez au dossier contenant le fichier docker-compose.yml. Montez l'image via Docker Compose :

```
$ sudo docker-compose up
```

## Démarrage

## Débloquer Jenkins

Pour être sûr que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs (où le trouver) ainsi que dans ce fichier sur le serveur :

`C:\Users\utilisateur\.jenkins\secrets\initialAdminPassword`

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

[Continuer](#)

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 ! ». Créez un utilisateur.

Administrez Jenkins :

### [Dashboard](#) - Configuration globale des outils

Ajoutez une installation JDK et une installation Maven.

### [Dashboard](#) - Gestion des plugins

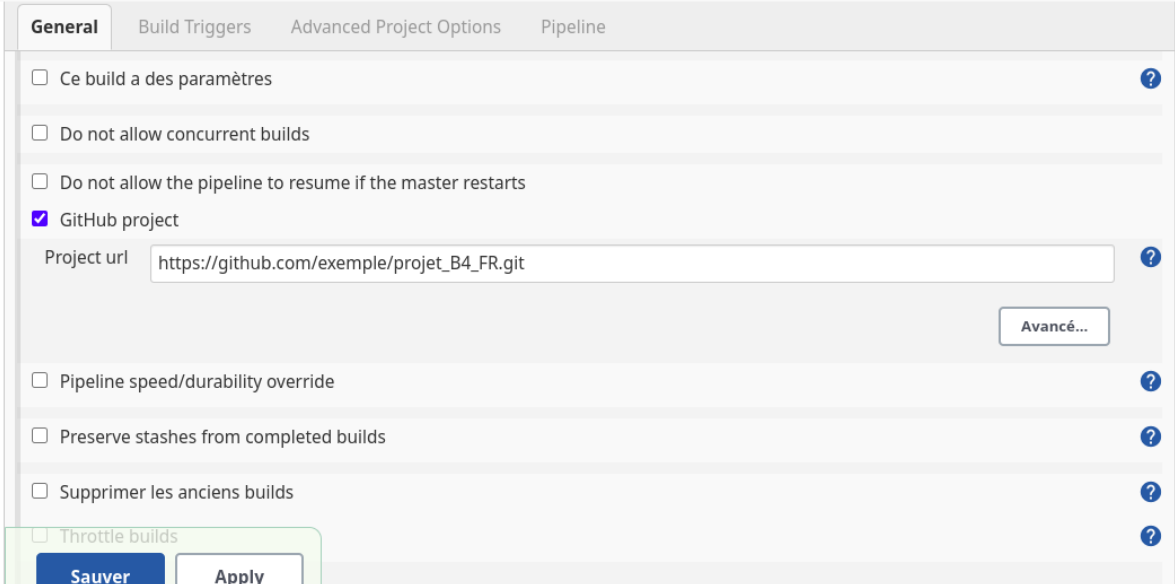
Installez les plugins suivants : GitHub Plugin et GitHub Intégration Plugin.

## Configuration de la Pipeline Jenkins

### [Dashboard](#) - Tous

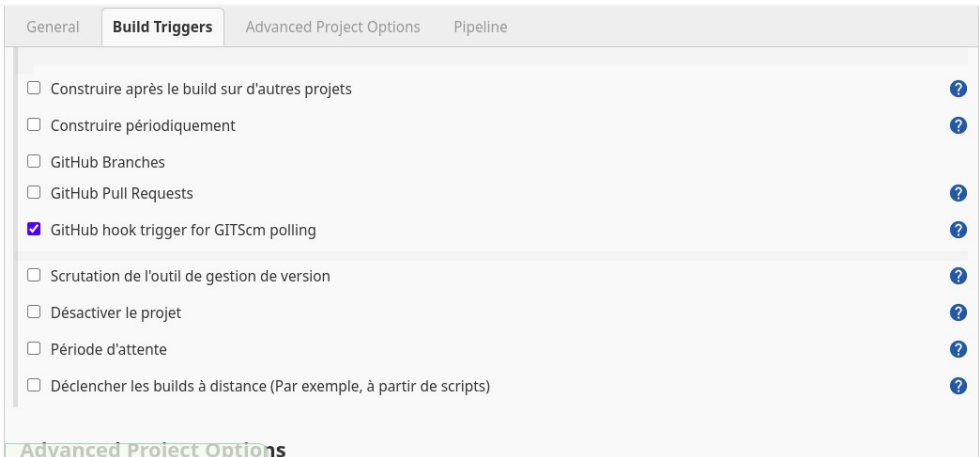
Saisissez un nom et créez une Pipeline.

Ajoutez l'URL de votre repository GitHub :



The screenshot shows the Jenkins Pipeline configuration page, specifically the 'General' tab. The 'Project url' field is set to 'https://github.com/exemple/projet\_B4\_FR.git'. The 'GitHub project' checkbox is checked. Other options like 'Ce build a des paramètres', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the master restarts', 'Pipeline speed/durability override', 'Preserve stashes from completed builds', 'Supprimer les anciens builds', and 'Throttle builds' are unchecked. There are 'Sauver' and 'Apply' buttons at the bottom.

Cochez la case GitHub hook :



The screenshot shows the Jenkins Pipeline configuration page, specifically the 'Build Triggers' tab. The 'GitHub hook trigger for GITScm polling' checkbox is checked. Other options like 'Construire après le build sur d'autres projets', 'Construire périodiquement', 'GitHub Branches', 'GitHub Pull Requests', 'Scrutation de l'outil de gestion de version', 'Désactiver le projet', 'Période d'attente', and 'Déclencher les builds à distance' are unchecked. The 'Advanced Project Options' section is visible at the bottom.

Créez un credential avec votre identifiant et mot de passe GitHub :

### Dashboard - Identifiants

Récupérez l'ID du credential créé pour l'utiliser dans la Pipeline.

Ajoutez le script suivant :

```
pipeline {
  agent any
  tools {
    maven 'Maven'
    jdk 'jdk8'
  }
  stages {
    stage('Clone'){
      steps {
        git branch: 'main',
          credentialsId: 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX',
          url: 'https://github.com/exemple/projet_B4_FR.git'
      }
    }
    stage('Build') {
      steps {
        dir("src") {
          sh 'mvn clean package'
        }
      }
    }
  }
}
```

Sauvegardez la configuration.



## Initialisation et configuration de Webhook Relay

<https://webhookrelay.com/v1/installation/cli>

### Installation de la CLI

Installez CLI pour gérer les ressources :

```
$ sudo wget -O /usr/local/bin/relay  
https://storage.googleapis.com/webhookrelay/downloads/relay-linux-  
amd64
```

Donnez lui les autorisations de s'exécuter et de se mettre à jour :

```
$ sudo chmod +wx /usr/local/bin/relay
```

### Inscription et authentification

Pour commencer à utiliser le service, vous devrez créer un compte. Rendez-vous sur la page d'inscription <https://my.webhookrelay.com/register/> et inscrivez-vous.

Relay CLI a besoin d'informations d'identification pour votre compte. Une fois enregistré, générez un jeton et utilisez la clé de jeton et le secret comme nom d'utilisateur et mot de passe pour l'authentification CLI:

#### New token created

Key

0c58c52d-3b99-43f2-b297-d56460d34827

Secret

2t3HAMxbVQE9

Configure relay command line access by running the following command:

```
relay login -k 0c58c52d-3b99-43f2-b297-d56460d34827 -s 2t3HAMxbVQE9
```

Or you can also set them as an environment variables:

```
export RELAY_KEY=0c58c52d-3b99-43f2-b297-d56460d34827  
export RELAY_SECRET=2t3HAMxbVQE9
```

[CLOSE](#)

Lancez-le :

```
$ relay login -k token-key-here -s token-secret-here
```

## Configuration de Webhooks Github

<https://webhookrelay.com/blog/2017/11/23/github-jenkins-guide/#Step-5-Configuring-Webhook-Relay>

Transférez les webhooks vers Jenkins :

```
$ relay forward --bucket github-jenkins
http://localhost:8080/github-webhook/
Forwarding:
https://my.webhookrelay.com/v1/webhooks/6edf55c7-e774-46f8-a058-f4d9c527a6a7 -> http://localhost:8080/github-webhook/
Starting webhook relay agent...
1.511438424864371e+09 info webhook relay ready...
{"host": "api.webhookrelay.com:8080"}
```

Une fois que vous avez votre URL d'entrée (ici

<https://my.webhookrelay.com/v1/webhooks/6edf55c7-e774-46f8-a058-f4d9c527a6a7>), allez à la configuration de webhook de GitHub et définissez-la :

rusenask / jenkins-whr-demo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Options Collaborators Branches Webhooks Integrations & services Deploy keys

### Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \*** your input should go here

https://my.webhookrelay.com/v1/webhooks/0a702a5d-39bd-4d

**Content type**

application/json

**Secret**

By default, we verify SSL certificates when delivering payloads. [Disable SSL verification](#)

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

## Test !

Modifiez le projet, réalisez un commit et poussez le vers GitHub. Vous devriez obtenir le résultat suivant sur Jenkins :

**Recent Changes**

### Stage View

Average stage times:  
(Average full run time: ~16s)

	Declarative: Tool Install	Clone	Build
#48 Dec 23 09:50 1 commit	334ms	1s	11s
#47 Dec 23 09:49 3 commits	262ms	1s	13s
	406ms	2s	9s

**Historique des builds** *tendance* ^

find X

#48 23 déc. 2020 09:50

**Liens permanents**