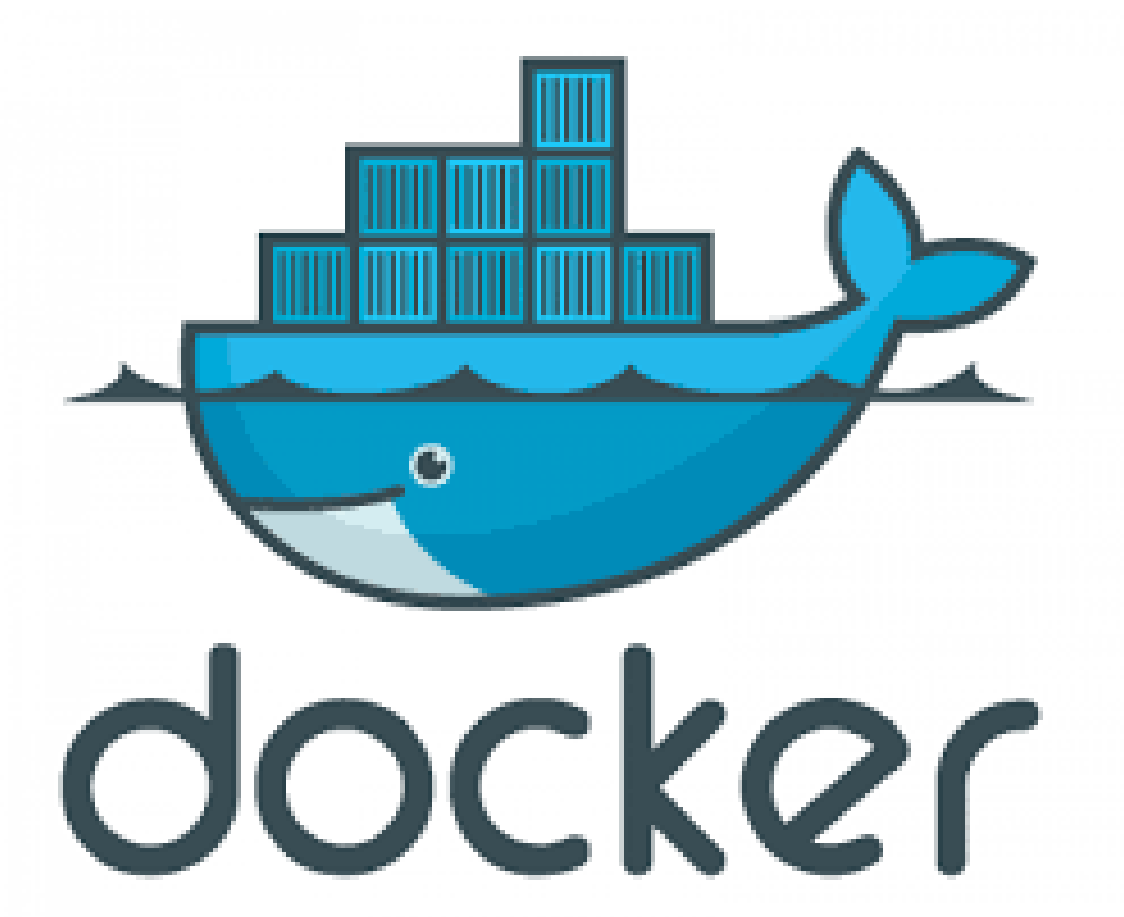


## Compte rendu intervention



# **Sommaire**

- 1- Installation de Jenkins**
- 2- Récupération du projet php**
- 3- Structuration du projet**
- 4- Création du docker-compose.yaml**
- 5- Configuration du fichier Jenkinsfile**
- 6- Démarrer les services**

# 1- Installation de Jenkins

Une fois que vous êtes connecté sur votre machine veuillez créer le fichier jenkins en exécutant cette commande :

```
mkdir jenkins
```

Une fois créé nous rentrons dans jenkins:

```
cd jenkins
```

résultat : `[root@valentin jenkins]`

Une fois dans jenkins, créons le fichiers docker-compose.yml avec cette commande:

```
vim docker-compose.yml
```

Une fois le fichier créé, nous allons le modifier. Appuyer sur la touche i afin de rentrer en mode INSERT (Writing). Une fois modifiés nous pouvons renseigner ce code.

```
version: '2.4'

services:
  jenkinsci:
    image: jenkinsci/blueocean:1.25.5
    ports:
      - 8080:8080
      - 50000:50000
    volumes:
```

```
- ./jenkins_home:/var/jenkins_home
- /var/run/docker.sock:/var/run/docker.sock
user: root
restart: on-failure
```

Pour sortir du mode INSERT, on appuie sur echap et ensuite on rentre la commande:

```
:wq!
```

Cette commande permet d'enregistrer les modifications et de sortir du fichier.

Une fois le fichier créer nous allons pouvoir lancer cette commande :

```
docker-compose up -d
```

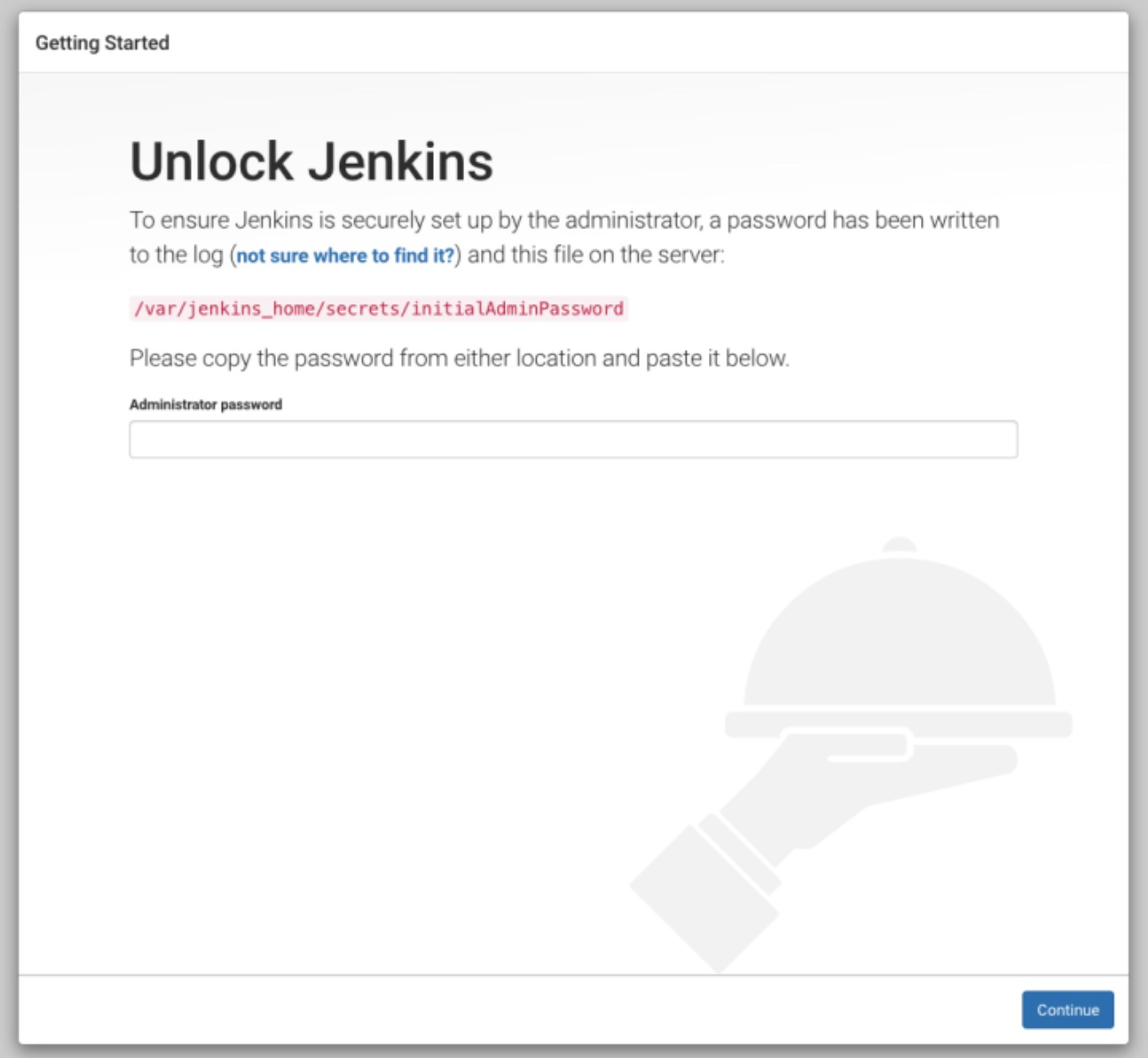
La commande docker-compose up -d permet de démarrer les services définis dans un fichier docker-compose.yml en mode "détaché" (arrière-plan). Cela signifie que les services s'exécutent en tâche de fond, et que vous pouvez continuer à utiliser votre terminal sans être bloqué.

Ici cette commande va se baser sur le fichier docker-compose.yaml que nous avons créé qui se trouve à la racine de jenkins.

Si le fichier est bien renseigné, cette commande doit afficher ce résultat:

```
[+] Running 1/1
Container jenkins-jenkinsci-1 Started
```

Une fois lancée, vous pouvez vous rendre sur l'interface web de votre Jenkins. Au chargement, la première page qui s'affiche:



Getting Started

## 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

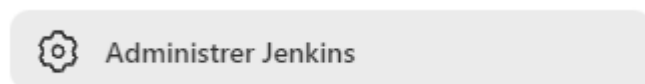
Continue

Pour trouver votre mot de passe, il suffit de rentrer cette commande:

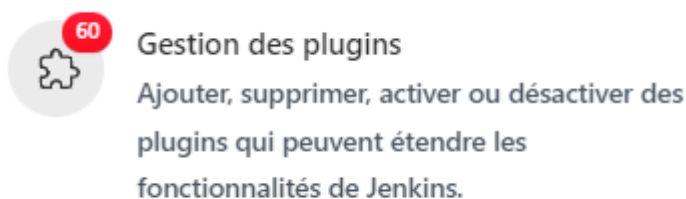
```
docker-compose logs
```

Connectez-vous avec votre mot de passe, faites l'installation des plugins par défauts.

Une fois les plugins installés, créez votre utilisateur sur Jenkins. La connexion étant effectuée vous devez maintenant télécharger les plugins nécessaires qui ne sont par défaut pas présents. Rendez vous dans "Administrer Jenkins".

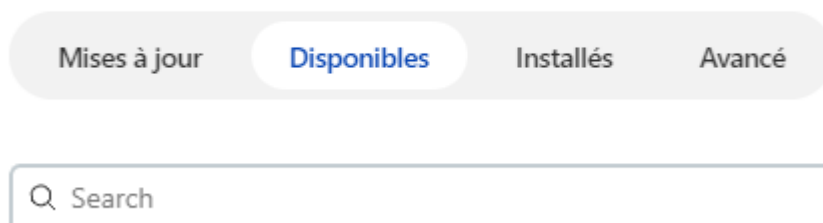


Ensuite cliquer sur "Gestion des plugins".



Veuillez par la suite cliquer sur "Disponible" en dessous de "Plugin manager".

## Plugin Manager



Une fois sur disponible, vous devez entrer "Docker" dans la partie Search.

Install	Name	Released
<input type="checkbox"/>	<b>Docker</b> 1.3.0 <a href="#">Cloud Providers</a> <a href="#">Cluster Management</a> <a href="#">docker</a> This plugin integrates Jenkins with <b>Docker</b> This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	9 j 11 h ago
<input type="checkbox"/>	<b>Docker Commons</b> 1.21 <a href="#">Library plugins (for use by other plugins)</a> <a href="#">docker</a> Provides the common shared functionality for various Docker-related plugins.	5 mo. 10 j ago
<input type="checkbox"/>	<b>Docker Pipeline</b> 563.vd5d2e5c4007f <a href="#">pipeline</a> <a href="#">DevOps</a> <a href="#">Deployment</a> <a href="#">docker</a> Build and use Docker containers from pipelines. This plugin is up for adoption! We are looking for new maintainers. Visit our <a href="#">Adopt a Plugin</a> initiative for more information.	2 mo. 3 j ago

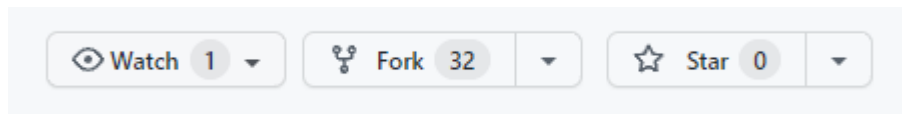
Veuillez cocher ces trois plugins. Et cliquez en bas sur “install without restart”.

**Install without restart**

## 2 - Récupération du projet php

Premièrement il faut se rendre sur le github ou se trouve le projet php. Une fois sur le dépôt, vous devez **fork\*** le projet.

\***Fork** sur Github est un mécanisme de copie d'un dépôt (repository) existant dans votre propre compte. Cela signifie que vous pouvez prendre le code source d'un projet public et en faire une copie complète dans votre propre compte, ce qui vous permet de le modifier comme si c'était votre propre projet. Les modifications que vous apportez à votre fork n'affectent pas le dépôt d'origine.



Vous devez maintenant vous rendre sur votre github et voir que le projet est désormais sur votre dépôt.



### 3- Structuration du projet

#### a/ Installation de git

Vous aurez besoin d'installer git afin de cloner le projet que vous venez de fork.

Pour installer git vous devez vous rendre sur:

```
Su root
```

et lancer la commande:

```
yum install git
```

git est maintenant installé.

#### b/ Utilisateur

Afin de bien structurer le projet nous allons créer un nouvel utilisateur.

```
useradd -g docker Votre_login
```

Connecter vous à votre utilisateur:

```
su Votre_login
```

par exemple:

```
[Valentin@valentin root]
```

#### c/ Création du fichier pour le projet

Pour structurer le projet nous créons un fichier Projet, ou dedans nous allons cloner le projet php.

```
mkdir Projet
```

d/ Récupérer le projet a partir du git.

Se rendre dans notre fichier répertoire projet:

```
cd Projet
```

une fois dans le projet, il vous suffit de cloner votre projet avec la commande “git clone” et suivant votre projet à cloner.

exemple:

```
git clone https://github.com/ValentinHusiaux/app-php.git
```

Le projet cloné, vous devriez le voir en utilisant la commande “ls”.

## 4- Création du docker-compose.yml

Création du fichier docker-compose.yml, ce fichier permet de définir les conteneurs nécessaires pour votre application, ainsi que leurs configurations, les volumes partagés, les réseaux, les dépendances, etc.

```
vim docker-compose.yml
```

Une fois le fichier créé, nous allons le modifier. Appuyer sur la touche i afin de rentrer en mode INSERT (Writing). Une fois modifiés nous pouvons renseigner ce code.

```
version: '3.9' <= Cette ligne spécifie la version de syntaxe de Docker Compose que vous utilisez.
services: <= La section services définit les conteneurs de l'application. Dans ce cas, il y a un seul
conteneur nommé "web".
web:
  image: tutum/apache-php <= ce conteneur web est basé sur l'image "tutum/apache-php".
  hostname: php-web <= La section "hostname" définit le nom d'hôte du conteneur.
  ports: <= La section "ports" spécifie les ports à mapper pour ce conteneur.
    - '9000:80'
  volumes: <= La section "volumes" définit les volumes partagés pour ce conteneur.
    - ./php-data:/var/www/html/
  networks: <= La section "networks" définit les réseaux auxquels ce conteneur appartient.
    - frontend
    - backend

db: <= Cette section définit un autre conteneur nommé "db" basé sur l'image "mariadb".
  image: mariadb
  hostname: db
  volumes:
    - ./db-data:/var/lib/mysql
  environment: <= la section "environment" définit des variables d'environnement pour le conteneur.
    - MARIADB_ROOT_PASSWORD=root
  networks:
```

```
- backend
```

```
networks: <= La section "networks" définit les réseaux Docker pour les conteneurs.
Dans ce cas, il y a deux réseaux nommés "frontend" et "backend". Les conteneurs peuvent
être associés à un ou plusieurs réseaux pour contrôler leur communication.
  frontend:
  backend:
```

Pour sortir du mode INSERT, on appuie sur echap et ensuite on rentre la commande:

```
:wq!
```

Cette commande permet d'enregistrer les modifications et de sortir du fichier.

## 5- Configuration du fichier Jenkinsfile

Création du fichier:

```
vim Jenkinsfile
```

on le modifie et on y ajoute :

```
pipeline { <= définit un pipeline Jenkins
  agent any
  stages {
    stage('version') { <= Étape "version": Exécute la commande docker-compose version pour
afficher la version de docker-compose installée sur le système.
      steps {
        sh 'docker-compose version'
      }
    stage('Build') { <= Étape "Build": Exécute la commande docker-compose up -d pour démarrer
les services définis dans le fichier docker-compose.yml en mode détaché (-d).
      steps {
        sh 'docker-compose up -d'
      }
    stage('curl-create_db.php') { <= Étape "curl-create_db.php": Exécute la commande curl
localhost:9001/create_db.php pour effectuer une requête HTTP GET sur localhost:9001/create_db.php. Cela
peut être utilisé pour tester l'application qui s'exécute dans les conteneurs Docker démarrés dans l'étape
"Build".
  }
```

Si l'on veut tester toutes les pages, il suffit de repérer la dernier stage et d'y remplacer l'adresse après le localhost..