

Compte rendu examen Docker

Gaspard BRISSET

(01/02/2023)

1/ On se connecte au serveur docker et on ajoute un utilisateur

- Pour se connecter

```
ssh root@gaspard
```

```
root@gaspard's password:
```

- Ajouter un utilisateur

```
useradd -g docker gaspard
```

```
su - gaspard
```

2/ On ajoute un dossier jenkins et on installe un fichier docker-compose.yml

- On creer un dossier Jenkins

```
mkdir jenkins
```

```
cd jenkins
```

- On installe le dossier docker-compose.yml

```
cat > docker-compose.yml
```

```
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
```

```
docker-compose
```

3/ On regarde le fonctionnement et on récupère le mot de passe ainsi que l'adresse ip

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

```
docker ps
```

```
docker-compose logs
```

4/ On se connecte grâce à l'adresse ip puis on gère l'installation des packages

- Se connecter à Jenkins

formation.ludovic.tech:8080/

- On télécharge les packages dans administrer Jenkins puis dans gestion des plugins:
 - Docker Commons Plugin
 - Docker Pipeline
 - Docker Plugin

5/ On manage les credentials et on crée un nouveau identifiants globaux

Update credentials

Portée ?

Global (Jenkins, agents, items, etc...)

Nom d'utilisateur ?

brisset

☐ Treat username as secret ?

Mot de passe ?

Concealed Change Password

ID ?

hub_docker_id

Description ?


6/ On crée un nouveau pipeline avec des paramètres


- Créer le pipeline


Saisissez un nom


NouveauPipeline


» Champ obligatoire


 **Construire un projet free-style**
Ceci est la fonction principale de Jenkins qui sert à build (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.

 **Pipeline**
Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.

 **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.

 **Dossier**
Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

 **Pipeline Multibranches**
Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

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

OK

- On modifie Pipeline script et on donne l'url du git ainsi que le fichier à lire

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Navigateur de la base de code ?

(Auto) ▼

Additional Behaviours

Ajouter ▼

Script Path ?

Jenkinsfile.groovy

☒ Lightweight checkout ?

7/ On crée le fichier Jenkinsfile et on lui donne un script

app-php / Jenkinsfile.groovy in main

<> Edit file

Preview changes

```
1  node {
2
3      def registryProjet='brisset/'
4
5
6      stage('Clone') {
7          checkout scm
8      }
9
10     stage('Push') {
11         sh 'apk update'
12         sh 'apk add docker-compose'
13         sh 'docker-compose up -d'
14         sh 'docker ps'
15     }
16 }
17 }
18 }
```

8/ On execute le Pipeline Jenkins

↑ Back to Project

🔍 Status

📄 Changes

📄 Console Output

📄 View as plain text

📄 Edit Build Information

🗑 Supprimer le build "#27"

🔍 Git Build Data

🌊 Open Blue Ocean

🔄 Replay

⚙ Pipeline Steps

📁 Workspaces

← Previous Build

✔ Sortie de la console

Started by user [gaspard](#)
Obtained Jenkinsfile.groovy from git <https://github.com/GaspardBrisset/app-php.git>
[Pipeline] Start of Pipeline
[Pipeline] node
Running on **Jenkins** in /var/jenkins_home/workspace/DockerPipelineGaspardExam
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/DockerPipelineGaspardExam/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url <https://github.com/GaspardBrisset/app-php.git> # timeout=10
Fetching upstream changes from <https://github.com/GaspardBrisset/app-php.git>
> git --version # timeout=10
> git --version # 'git version 2.36.1'
> git fetch --tags --force --progress -- <https://github.com/GaspardBrisset/app-php.git> +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 39d52f5905b7b6e95fa93d6f85a284e9d2ad2eeb (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 39d52f5905b7b6e95fa93d6f85a284e9d2ad2eeb # timeout=10
Commit message: "Update Jenkinsfile.groovy"
> git rev-list --no-walk 39d52f5905b7b6e95fa93d6f85a284e9d2ad2eeb # timeout=10