Docker et l'intégration continue

Alain Tchana, Boris Teabe ENSEEIHT 2016-2017, IRIT/Equipe SEPIA

email: alain.tchana@enseeiht.fr

10 novembre 2016

Table des matières

1 Docker

Cette section est consacrée à la prise en main de docker.

1.1 Environnement de travail

L'installation de docker nécessite un système 64-bit, un noyau linux de version supérieure à 3.10. Pour obtenir la version de votre noyau linux, utilisez la commande "uname -r".

1.2 Installation de docker

```
#Installation des paquets nécessaires.
#Installation de curl.
sudo apt-get update
sudo apt-get install curl
#Il est récommandé d'utiliser ces commandes pour l'installation de docker
#afin d'avoir la dernière version stable.
curl -sSL https://get.docker.com/ | sh
#Vérifier que docker est bien installé. Cette commande télécharge une image test et exécute
#le programme hello-world dans cette dernière.
sudo docker run hello-world
```

1.3 Utilisation de docker

Il est question ici de présenter quelques commandes de base.

```
#Nous allons télécharger une image depuis le hub
docker pull ubuntu
#Dans la suite nous utilisons cette image.

#Lister les images présentes localment:
docker images
#Cette commande retourne entre autre le nom et l'identifiant de chaque image.

#La suppression d'une image se fait avec la commande:
docker rmi <Nom de limage>

#Le démarrage d'un conteneur avec la commande docker run.
docker run -t -i  --name <Nom du conteneur> ubuntu /bin/bash
#Vous venez de démarrer votre conteneur.
#Pour avoir la liste de tous les conteneurs, exécutez depuis un autre terminal la commande docker ps -a.
docker ps -a
#Cette commande retourne l'identifiant des conteneurs

#Customisation de l'image d'un conteneur. Installer par exemple les packages apache.
#Ensuite, enregistrer l'image modifiée
#sur votre poste en utilisant la commande.
docker commit <ID du conteneur> <nom image>
#Ouvrir un autre terminal et arrêter votre conteneur grâce à la commande.
docker stop <Nom du conteneur>

#Suppression d'un conteneur précédemment démarré.
docker rm <Nom du conteneur>
#Lancer un job dans un conteneur et afficher les logs.

JOB1=$(docker run -d --name echo ubuntu echo "echo")
docker logs $JOB1
```

1.4 Prise en main du hub docker

Le hub docker est un dépôt d'images.

```
#Pour réaliser une recherche depuis le dépot, utiliser la commande docker search
#Dans notre cas, nous recherchons les images ubuntu.
docker search ubuntu
#Pour télécharger une image, utiliser la commande: docker pull <nom de l'image>:<version>
docker pull ubuntu:14.10
docker images
#L'image est maintenant disponible en local.
```

1.5 Création d'un image from scratch

Un dockerfile est un document texte contenant toutes les commandes qu'un utilisateur aurait exécuté pour construire son conteneur. Utiliser un dockerfile permet a l'utilisateur de créer des fichiers transportables, facilitant ainsi la reproduction de l'environnement d'un conteneur. Dockerfile utilise un DSL (Domain Specific Language) basique avec des directives bien définies. Les principales directives sont :

- MAINTAINER : nom et email du mainteneur du conteneur ;
- FROM : image de base (Ubuntu,Debian) ;
- VOLUME : point de montage ;
- RUN : commande à exécuter pour installer le conteneur ;
- ENTRYPOINT : commande qui s'exécute au démarrage du conteneur ;
- CMD : commande qui s'exécute au démarrage du conteneur ;
- ADD : copier un fichier du répertoire vers le système de fichiers du conteneur ;
- USER : utilisateur qui exécute les commandes dans le conteneur ;
- EXPOSE : port(s) à exposer à l'extérieur.

```
#Création du répertoire du contexte de notre image
mkdir context
cd context
#Création de notre fichier Dockerfile.
cat << EOF > Dockerfile
FROM ubuntu
MAINTAINER tpcloud "toto@titi.fr"
RUN apt-get update
RUN apt-get install -y apache2
RUN echo 'Je suis dans le conteneur' > /var/www/html/index.html
EXPOSE 80
EOF
#Création de l'image à partir du Dockerfile
docker build --tag="tpcloud" .
#Vous pouvez voir l'image créée en utilisant docker images.
#Démarrer un conteneur à partir de l'image.
docker run -t -i -p 8089:80 tpcloud /bin/bash
#Démarrer apache dans votre conteneur
service apache2 start
#Depuis un navigateur de cumulus ou stratus, accéder à la page http:/<adresse IP de votre serveur>:8089/
```

1.6 Liaison de conteneurs

Dans cette section nous illustrons la liaison de conteneurs à travers une application multi-tier.

```
#Les liaisons permettent une communication sécurisée entre conteneur.
#La liaison entre deux conteneurs créée un tunnel
#entre le conteneur source et le conteneur client.
# Le conteneur client peut ainsi accéder aux données du conteneur source. La création
#d'une liaison se fait en utilisant l'option --link. Nous allons créer un conteneur
#db qui jouera le role de base de données. Un autre conteneur web qui sera notre conteneur
#serveur d'applications. Nous définissons une liaison entre les deux conteneurs.
docker run -t -i --name db --expose=3306 ubuntu /bin/bash
#Installation du server mysql dans le conteneur db.
apt-get update
apt-get install mysql-server
```

```
#Création du conteneur web en ajoutant l'option link pour créer le lien avec le conteneur db.
docker run -t -i  -P --name web --link db ubuntu /bin/bash
#Installation de mysql client dans le conteneur web.
apt-get update
apt-get install mysql-client
#Nous allons maintenant autoriser les connexions externes à mysql.
#Pour se faire éditer le fichier /etc/mysql/mysql.conf.d/mysqld.cnf du conteneur db et remplacer la ligne
#bind-address = 127.0.0.1 par  bind-address = 0.0.0.0
#Redémarrer votre serveur mysql.
/etc/init.d/mysql restart
#Connecter vous à votre service mysql et saisir le mot de passe root de mysql
mysql -u root -p
#saisir les commandes suivantes sous mysql
CREATE USER 'tpcloud'@'%' IDENTIFIED BY 'tpcloud';
#RAPPEL: "tpcloud" est le mot de passe
GRANT ALL ON *.* to 'tpcloud'@'%';
FLUSH PRIVILEGES;
#Redémarrer votre service mysql dans votre conteneur db.
/etc/init.d/mysql restart
#depuis votre conteneur web saisir la commande env pour avoir
#la liste des variables d'environnement. Vous aurez DB_PORT_3306_TCP_ADDR
#Connecter vous au container mysql en utilisant la commande et rentrer le mot de passe
mysql -u tpcloud -h $DB_PORT_3306_TCP_ADDR -p
```

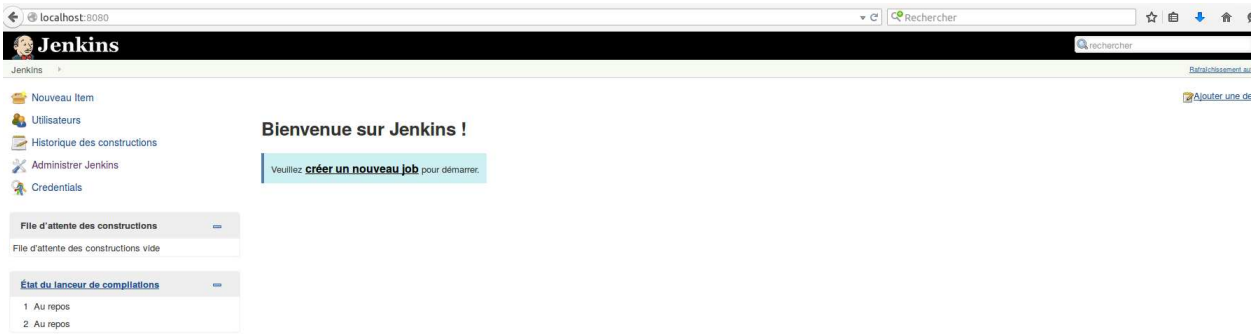
2 Intégration continue avec Jenkins

Jenkins est un serveur d'intégration continue. L'intégration continue consiste à vérifier que toute modification du code source d'une application ne produit pas une régression.

Installation de jenkins

```
apt-get install openjdk-7-jdk
wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
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
#Modifier le fichier /var/lib/jenkins/config.xml et modifier <useSecurity>true</useSecurity>
#à <useSecurity>false</useSecurity>
```

Démarrer un navigateur depuis stratus ou cumuls et accéder à la page d'accueil http ://<adresse IP de votre serveur> :8080/



3 Intégration de Docker à Jenkins

L'idée est de toujours effectuer dans un environnement sain et contrôlé le build (compilation et packaging) d'une application.

— Jenkins démarre un conteneur docker
— Jenkins démarrer un slave temporaire dans le conteneur
— Le build se fait dans le conteneur. Jenkins récupère les résultats (logs, métadonnées, artifacts)
— Le conteneur docker est détruit à la fin du build.

Installation du pluging docker dans Jenkins

Nous allons installer le plugin docker de Jenkins via l'interface web. L'installation se fait à partir de *configuration > gestion des plugings*. Cocher *docker pluging*, installer le pluging et redémarrer Jenkins.

Configuration de docker et création de notre image

```
echo "DOCKER_OPTS=\"-H 0.0.0.0:4243 -H unix:///var/run/docker.sock\"" >> /etc/default/docker
#Redémarrez le service Docker
sudo service docker restart
#Création de notre image "jenkins" pour Jenkins.
#Nous allons utiliser une image bien configurée et disponbile sur le docker hub.
docker search jenkins
docker pull evarga/jenkins-slave
#Rassurez vous que votre image est bien disponible en locale.
docker images
```

Déclaration de l'image docker sous Jenkins

Dans la configuration générale, *configuration>cloud*, nous enregistrons l'image docker précédemment créée dans Jenkins. Il est nécessaire de vérifier que la configuration est bonne en réalisant un test de connexion vers votre service docker. Rajouter un credential "Username with password" afin que Jenkins puisse se connecter

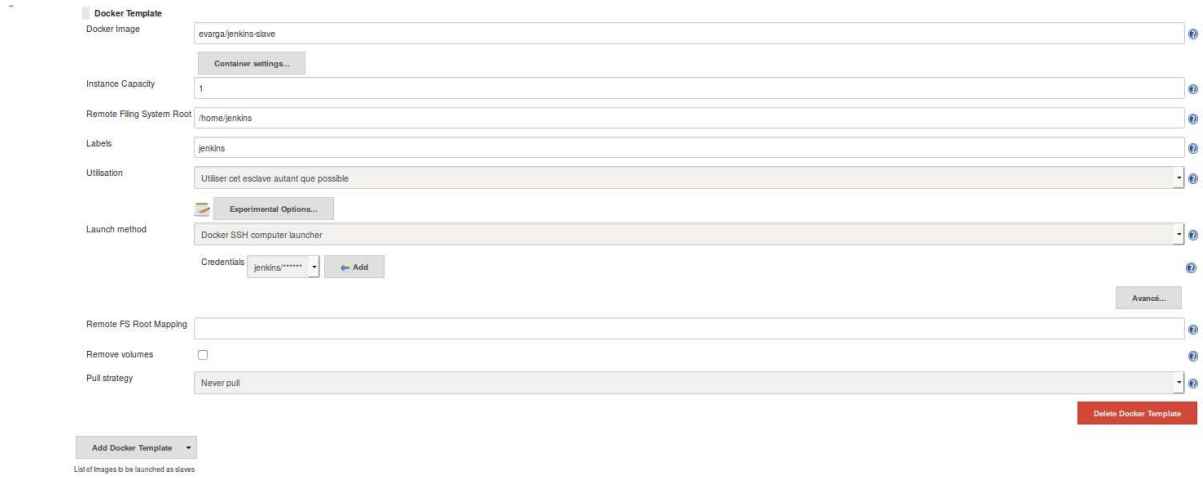
en ssh sur le conteneur (username : jenkins, password : jenkins). Renseigner "jenkins" comme label du conteneur.

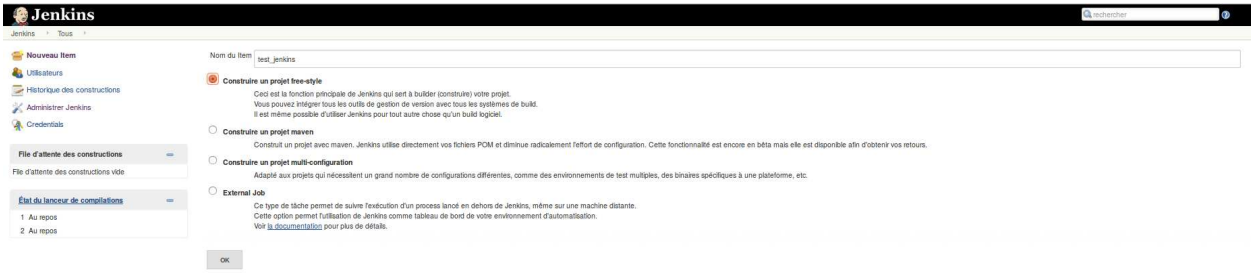
Création d'un job sous jenkins

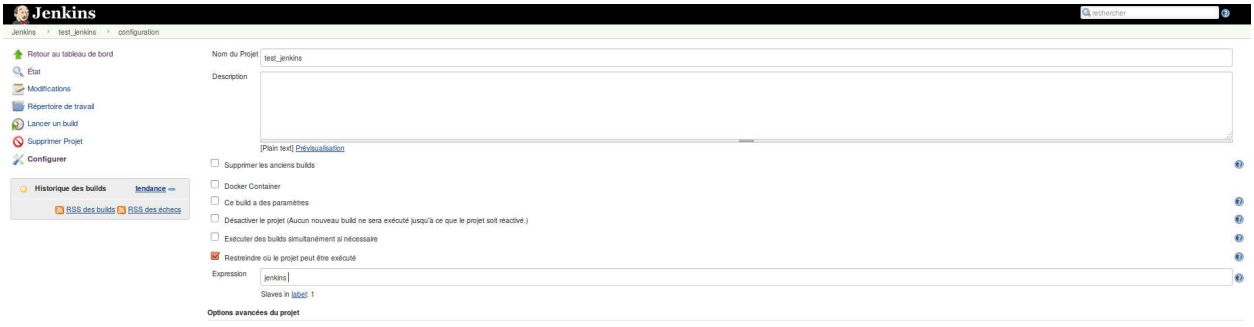
Nous créons un nouveau job sous Jenkins,*Nouveau item*.
Ensuite, configurer le job afin qu'il se démarre automatiquement dans un conteneur "jenkins". Nous pouvons à présent sauvegarder le job et exécuter le projet. Au niveau de la console nous aurons comme sortie.

4 Illustration à travers un micro projet JEE

Nous allons préparer l'image du conteneur en installant tous les éléments nécessaires à notre projet.








```
#Nous allons tout d'abord commencer par une configuration ssh
#Lancer un autre terminal et démarrer votre conteneur evarga/jenkins-slave
docker run -t -i -p 0.0.0.0:2222:22 evarga/jenkins-slave /bin/bash
/etc/init.d/ssh start
#A partir d'un terminal de votre serveur de travail, connectez vous via ssh à votre conteneur.
ssh jenkins@localhost -p 2222
```

```
#Le mot de passe est jenkins.
#Si vous parvenez à vous connecter au conteneur, alors vous êtes sur la bonne voie.
#Générez une clé publique ssh depuis votre conteneur.
ssh-keygen -t rsa
#Enregistrement de la clé sur votre serveur.
cat ~/.ssh/id_rsa.pub | ssh root@<Votre adresse IP de votre serveur>  'cat >> ~/.ssh/authorized_keys'
#installation de "pache ant" dans le conteneur.
apt-get install ant
#Depuis un autre terminal de votre serveur, enregistrer le conteneur avec les nouvelles modifications.
docker commit <ID du conteneur>  evarga/jenkins-slave
#Nous allons maintenant modifier notre projet jenkins , test_jenkins  pour y ajouter
#les éléments de notre projet JEE. Il s'agit d'un micro projet de chat. L'objectif
#est  de réaliser le build de notre application avec Jenkins.
#Dans la configuration de notre projet, aller à Ajouter une étape au build.
#Sélectionner Exécuter un script Shell. Copiez les commandes ci-dessous et
#collez dans la zone de script en renseignant bien votre nom d'utilisateur et votre adresse IP.
scp -r root@<Votre adresse IP>:~/chat .
cd /home/jenkins/workspace/test_jenkins/chat/
ant build
#Enregistrez les modifications du job et exécutez le projet Jenkins.

#Pour la suite vous pouvez imaginer ce qui est possible de réaliser avec jenkins et docker
```