Atelier 1 : Installation des logiciels Hadoop avec Docker

Pour cette séance, on utilisera la technologie Docker qui est aussi bien utilisée en phase de prototypage qu'en phase de production dans les projets Big Data.

Présentation générale de Docker

Docker est une plateforme open source qui assure la portabilité des applications distribuées qu'elle héberge. En effet, l'application devient indépendante des systèmes d'exploitation sous-jacents. Docker fournit, grâce à la notion de container une granularité plus fine que les solutions de virtualisation dans les systèmes classiques.

Le container est une sorte de VM allégée mais beaucoup plus rapide à l'exécution des applications et moins consommatrice de ressources. En plus de l'économie en surcoût CPU, Docker fournit des fonctionnalités intéressantes de contrôle de version et de portabilité.

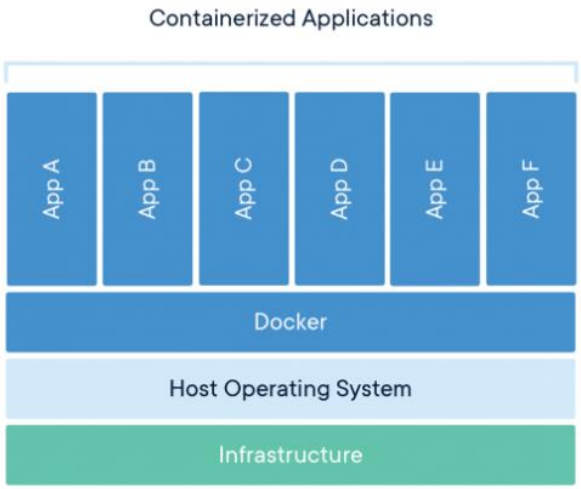

Figure 1Virtualisation avec Docker

Propriétés de Docker

Installation simple : les conteneurs sont lancés à partir d'images portables contenant les différents fichiers nécessaires à l'installation des logiciels ciblés (librairies, programmes, données de configuration, etc.). L'installation se fait via une seule ligne de commande.

Déploiement pratique : l'intégration d'une application dans un conteneur permet de faciliter son déploiement et la gestion des versions notamment le retour à une version antérieure.

Isolation des applications : chaque application contenue dans un conteneur fonctionne indépendamment des autres conteneurs présents sur le même système d'exploitation. L'avantage est donc de pouvoir gérer plusieurs applications n'ayant pas les mêmes

contraintes et exigences sur une même machine. Ceci est pratique pour réaliser rapidement des prototypes et faire des premiers tests fonctionnels.

Amélioration de la productivité des développeurs : les nouveaux environnements se configurent plus facilement et en peu de temps.

Migration plus simple des applications : la migration d'applications basées sur des conteneurs Docker situées en local sur vos machines se fait directement à moindre coût vers l'environnement de production.

Il est recommandé de travailler sur Linux pour ce TP.

Fonctionnement de Docker

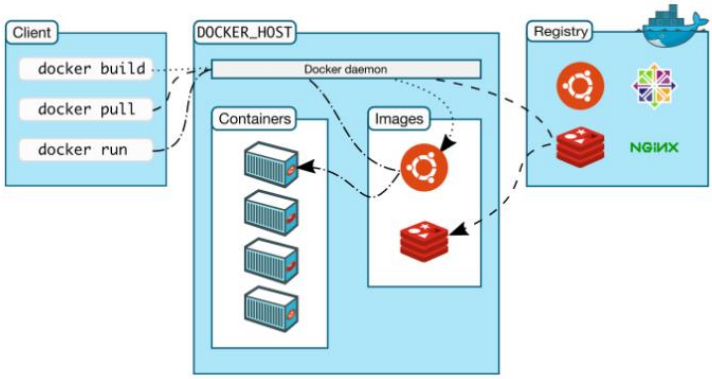

Figure 2 Création de conteneurs Docker

Docker introduit la notion d'image (snapshot) qui permet d'installer le logiciel nécessaire au fonctionnement d'un conteneur. Dockerfile est le fichier de base utilisé pour lister et automatiser la création d'image Docker.

Docker Compose permet, comme son nom l'indique, de composer plusieurs conteneurs dans un même système et de les relier par un réseau. Docker Compose permet d'orchestrer les conteneurs et ainsi de simplifier le déploiement sur de multiples environnements. Docker Compose est un outil écrit en Python qui permet de décrire, dans un fichier YAML, plusieurs conteneurs comme un ensemble de services.

Pour plus d'information sur Docker compose, on peut visiter le site de Docker ou les tutoriaux sur Internet.

Installations

Téléchargement de Docker et les fichiers de configuration

Sites web :

https://docs.docker.com/install/

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

Quelques commandes utiles de Docker :

Pour tester que l'installation est réussie, vérifier sur votre terminal :

```
docker --version
Docker version 20.10.12, build e91ed57
docker-compose --version
docker-compose version 1.29.2, build 5becea4c
```

Pour consulter les conteneurs installés :

```
docker container ps
```

Pour consulter les images utilisées dans les conteneurs :

```
docker container ls -a
```

Pour effacer tous les containers :

```
docker ps -a
docker rm -f containerID
```

Effacer toutes les images :

```
docker images
docker rmi -f imageID
```

Pour avoir la liste des ID des conteneurs crées :

```
docker container ls -aq
```

Pour tuer un conteneur particulier :

```
docker container kill <containerID>
```

Pour arrêter tous les containers en cours

```
docker container stop $(docker container ls -aq)
```

Pour arrêter tous les conteneurs et nettoyer Docker de toutes les images téléchargées:

```
docker container stop $(docker container ls -aq) && docker system prune -af --volumes
docker container rm $(docker container ls -aq)
```

Pour démarrer la création de l'image et du cluster installé par docker-compose :

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

Pour arrêter le cluster et supprimer ses conteneurs proprement :

```
docker-compose down
```

Installation d'un cluster Hadoop

Le but de ce TP est d'installer un cluster Hadoop en utilisant Docker-compose. Ce cluster doit permettre d'installer les composants de Hadoop dans des conteneurs différents : le NameNode, le DataNode, le Resource Manager et un Node Manager.

Nous rappelons que NameNode permet de gérer le stockage réparti et que Ressource Manager permet de répartir le traitement. Les Data Node permettent le stockage réparti et les Node Manager gèrent les ressources nécessaires pour le stockage et pour le traitement des tâches.

Dans ce TP on va utiliser l'environnement docker-compose d'un cluster Hadoop donné en classe.

Dans votre dossier de travail, visualiser docker-compose

```
nano docker-compose.yml
```

Quelques commandes utiles de Docker

Pour démarrer la création de l'image et du cluster :

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

L'option -d permet de lancer la commande en tâche de fond (background).

Pour vérifier les images installées :

```
docker container ls
```

Pour vérifier les containers crées :

```
docker container ps
```

Pour se connecter sur le container du namenode :

```
docker exec -it namenode bash
```

Pour sortir de ce container:

```
exit
```

Pour une visualisation web du container namenode, il faut connaître l'adresse de la machine et vérifier sur quel port le container est monté. Pour ce faire utiliser :

```
ifconfig
```

Et pour le port :

```
docker container ls
```

Explorer les différents ports des NameNode er ResourceManager. Pour ce faire, dans un navigateur taper **http://@IP:PortNum**.

Par exemple:

http://172.18.0.3:9000

Partage des données :

Créer un dossier **/Data** dans **namenode** où on va mettre les fichiers que l'on va mettre par la suite dans HDFS.

```
mkdir /Data

exit
```

On rappelle que dans le fichier de configuration **docker-compose.yml**, pour chaque container, on dispose d'une partie « volumes » de la configuration qui consiste à faire correspondre un endroit de stockage des données en local avec un emplacement sur le conteneur master du cluster. Par exemple, sur ma machine, j'ai la correspondance entre le dossier « /home/hadoop/BigData/Data » et /Data sur le conteneur namenode. Ceci est exprimé dans le fichier docker-compose.yml par la ligne suivante :

```
volumes:
        -/home/hadoop/BigData/Data:/Data
```

Il faut mettre à jour cette ligne dans docker-comopse.yml pour mettre le chemin vers votre dossier des données en local.

On teste avec un petit fichier file.txt que vous avez en local (dans le dossier Data) et qui est automatiquement recopié sous /Data sur la machine du cluster.

Remarque : A chaque fois que vous mettez à jour le chemin de votre dossier, il faut refaire un build (l'image ne sera pas recréée donc ceci ne prendra pas beaucoup de temps) :

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

Sinon utiliser la commande Docker pour copier un fichier file.txt d'un dossier local à un dossier sur le dossier container.

```
docker cp /home/hadoop/BigData/Data/file.txt namenode:/Data/file.txt
```

Il faut se connecter sur le container du namenode:

```
docker exec -it namenode bash
```

Vérifier que file.txt est dans le dossier /Data comme prévu dans la configuration.

```
ls /Data
```

Remarques :

Pour mettre à jour l'environnement docker :

```
docker-compose down

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

En cas de création de nouvelles images, il arrive des fois que les containers démarrés soient dans un état 'Unhealthy' suite à des confusions dans les ports utilisés. Il est conseillé de nettoyer dans ce cas tous les containers et toutes les images de Docker et de refaire un build :

```
docker container stop $(docker container ls -aq) && docker system prune -af --volumes

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