osones

Travaux Pratiques
Docker

Version du 04/2017

Introduction

Ce TP permet de se familiariser avec la CLI Docker et sur les fonctions de base des conteneurs Docker.

Nous montrerons comment :

- créer une image
- lancer un conteneur
- exposer des ports
- monter des volumes

Installation

Plusieurs méthodes d'installation sont disponibles pour Docker. La plupart des distributions possèdent un package officiel. Néanmoins ce package n'est pas tout le temps à jour, il peut apparaitre utile d'utiliser les dépots fournis par Docker afin de disposer d'un package à jour.

Exemple pour Ubuntu :
`https://docs.docker.com/engine/installation/linux/ubuntu/#install-docker`

Préparation de l'environnement

Afin de travailler dans un environnement vierge, créez un dossier et placez vous à l'intérieur. Toutes les commandes qui suivront seront lancées depuis ce répertoire.

```
~ $ mkdir docker
~ $ cd docker
```

Construire une image Docker

Nous utiliserons le serveur web Nginx comme démonstrateur.

Copier/coller ce dockerfile dans un fichier Dockerfile :

```
FROM ubuntu:16.04
RUN apt update \
      && apt install -yf \
      nginx
EXPOSE 80
CMD ["nginx, “-g”, “daemon off;"]
```

Pour construire notre image :

```
$ docker build -t mynginx .
```

Une fois que les différentes layers ont été construites, vous devriez retrouver votre image en local :

```
$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
mynginx             latest              62d27f54b98b        About a minute ago   212MB
```

Différence CMD et ENTRYPOINT

Utilisons le package mtr comme démonstrateur, celui ci permet d’obtenir un traceroute userfriendly.

Créer deux dossiers cmd et entrypoint :

```
$ mkdir cmd entrypoint
$ cd cmd
```

Copiez y ce Dockerfile :

```
FROM ubuntu:16.04
RUN apt update \
      && apt install -yf \
      mtr
CMD ["mtr", “8.8.8.8”]
```

Builder l’image :

```
$ docker build -t mtr-cmd .
```

Déplacer vous dans l’autre dossier et copiez y ce Dockerfile :

```
FROM ubuntu:16.04
RUN apt update \
      && apt install -yf \
      mtr
ENTRYPOINT ["mtr"]
```

Builder le :

```
$ docker build -t mtr-entrypoint .
```

Vous disposez de deux images : mtr-cmd et mtr-entrypoint

Lancer chacune des deux images :

```
$ docker run -it mtr-cmd
$ docker run -it mtr-entrypoint
```

Que se passe t-il ?

Lancer les images de cette façon maintenant :

```
$ docker run -it mtr-cmd 8.8.8.8
$ docker run -it mtr-entrypoint 8.8.8.8
```

Que remarquez vous ?
Quelles sont les différences entre CMD et ENTRYPOINT

Lancer un conteneur

$ docker [OPTIONS] COMMAND [ARG…]

Les premières options à exploiter sont : -t , -i et -d
Elles permettent de choisir le “mode” du conteneur. Notamment entre le fait d’être exécuté au premier plan ou en arrière plan.

Différence -t et -i

```
$ docker run -i -t ubuntu /bin/bash
root@ebc138d8cdc9:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin
srv  sys  tmp  usr  var
```

Le comportement est normal.

```
$ docker run -i ubuntu /bin/bash
ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

Que constate t-on ?

```
$ docker run -t ubuntu /bin/bash
root@f0f4c4dcc7da:/# ls
```

Que constate t-on ?

Exposer un port

Reprenons notre image mynginx

```
$ docker run -d -p 8000:80 mynginx
```

Nous exposer le port 8000 de notre host vers le port 80 de notre conteneur.

Vérifions que notre conteneur est bien en écoute :

```
$ curl http://localhost:8000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>

[...]

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Peut-on accéder à cette page html avec une autre URL ? un autre port ?

On peut ne pas vouloir fixer le port sur notre host :

```
$ docker run -d -p 80 mynginx
```

Comment savoir le port associé à notre host ?

Comme nous avons spécifier un EXPOSE dans notre Dockerfile, nous pouvons aussi utiliser le paramètre `-P` :

```
$ docker run -d -P mynginx
```

Comment vérifier le mappage de port entre notre host et notre conteneur ?

Monter un fichier

Le répertoire dans lequel Nginx va, par défaut, chercher les pages html est /var/www/html

Créons cet index.html dans notre dossier courant :

```
<html>
<h1> Osones, l'expertise cloud </h1>
</html>
```

Montons le dans notre conteneur :

```
$ docker run -d -p 8000:80 -v $PWD/index.html:/var/www/html/index.html mynginx
```

Vérifions que notre page est bien accessible :

```
$ curl http://localhost:8000
<html>
  <h1> Osones, l'expertise cloud </h1>
</html>
```

Le test peut aussi être effectué sur votre navigateur. Les balises HTML sont correctement interprétées.

Monter un volume

Créons notre volume au préalable :

```
$ docker volume create myvolume
myvolume
```

Par défaut, les volumes sont stockés dans /var/lib/docker/volumes/

```
# ls /var/lib/docker/volumes
myvolume metadata.db
# cd /var/lib/docker/volumes/myvolume/_data
# echo "<html> Hello Osones </html>" > index.html
```

Montons ce volume dans un conteneur :

```
$ docker run -d -p 8001:80 -v myvolume:/var/www/html mynginx
$ curl http://localhost:8001
<html> Hello Osones </html>
```

Notre volume est correctement monté !