Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 2 : DockerFile

V2023.1

Table des matières

1 - Paramètres génériques

Pour réaliser la construction d'une nouvelle image Docker (Build), il faut créer un fichier de description nommé « Dockerfile » à la racine du projet.

Ce fichier peut contenir entre autre les directives suivantes :

- FROM : qui indique de quoi dérive notre image (debian:jessie, debian:wheezy, ubuntu:16.04, …), exemple :

```
FROM debian:jessie
```

- ENV : qui permet de positionner des variables d'environnement. Par exemple pour que l'outil « debconf » ne pose pas de questions pendant les installations :

```
ENV DEBIAN_FRONTEND noninteractive
```

- WORKDIR : permet de changer de répertoire de travail, exemple :

```
WORKDIR /usr/local
```

- MAINTAINER : qui indique la personne ayant réalisé l'image, exemple :

```
MAINTAINER Cyril Drocourt <cyril@drocourt.info>
```

Remarque - Cette directive est aujourd'hui obsolète et remplacée par LABEL qui est plus générique :

```
LABEL maintainer="Cyril Drocourt <cyril@drocourt.info>"
```

2 - Exécution de commandes

Il est possible d'exécuter des commandes spécifiques dans le nouveau conteneur à l'aide de la directive « RUN » qui indique l'action ou les actions à réaliser :

```
RUN apt-get update && apt-get install -y curl && rm -rf
/var/lib/apt/lists/*
RUN curl -LO "http://monsite/" && tar -xzf fichier.tar.gz -C
/usr/local && rm fichier.tar.gz
```

Il faut noter que plusieurs directives RUN sont utilisables, et qu'il est d'ailleurs conseillé de séparer les commandes en groupes. En effet, lors d'un Build ultérieur, ce dernier pourra être reprit à partir du dernier RUN effectif.

3 - Ajout de ressources externes

Il est possible d'ajouter une ressource externe au conteneur :

- COPY : qui permet de copier des fichiers du répertoire courant vers l'image :

```
COPY apache2.conf /etc/apache2/
```

- ADD : qui permet aussi la copie, mais qui permet également certaines choses supplémentaires comme le désarchivage ou la récupération automatiques d'URL, même si la méthode préconisée reste l'utilisation de « wget » et « curl » :

```
ADD fichier.tar.gz /
ADD http://server/file.tar.gz /usr/local
```

4 - Les ports et volumes

Les paramètres liés aux ports et aux volumes sont :

- EXPOSE : pour exposer un port qui pourra ensuite être utilisé à l'extérieur de l'image, par exemple pour le port 3000 :

```
EXPOSE 3000
```

- VOLUME : pour indique un répertoire qui sera disponible à l'extérieur de l'images :

```
VOLUME /app/log
```

5 - Le point d'entrée

Le point d'entrée du conteneur est géré par :

- ENTRYPOINT : pour indiquer le point d'entrée de l'image, c'est à dire l'instruction qui sera exécutée. Cette dernière ne peut pas être écrasée par les arguments du conteneur :

```
ENTRYPOINT ["macommande"]
```

- CMD : idem, sauf que ce dernier sera écrasé par un éventuel argument de lancement du conteneur :

```
CMD ["ps","-ax"]
```

Il est possible de cumuler les deux éléments pour par exemple spécifier une commande et le fichier sur lequel porte la commande, qui pourra être écrasé :

```
ENTRYPOINT ["commande"]
CMD ["--help"]
```

6 - Construction

Une fois ces paramètres définit, il est possible de construire l'image :

```
[root@drocourt ~]# docker build -t monnom .
```

Pour le lancer :

```
[root@drocourt ~]# docker run -d -p 3000:3000 -v ./log:/app/log monnom
```

Remarque :

Si il est nécessaire de passer sur un autre utilisateur on n'utilise pas « sudo » mais :

```
gosu user cmd
```

7 - Exercices

7.1 - Exercice 3

1. Vous allez réaliser un Dockerfile basé sur Debian ou Alpine,
2. Qui contient une installation de Apache et de PHP,
3. Le port utilisé sera 80,
4. Le service « apache » sera démarré automatiquement,
5. Vous ajouterez de manière automatique le fichier « index.php » dans cette image,
6. Testez la construction et l'exécution de cette image de manière automatique,

Fichier index.php :

```
<?php
  header('Content-Type: text/plain');
  echo "hostname : ".gethostname()."\n";
  echo "IP server : ".$_SERVER['SERVER_ADDR']."\n";
  echo "IP client : ".$_SERVER['REMOTE_ADDR']."\n";
                  echo         "X-Forwarded-for:         ".
$_SERVER['HTTP_X_FORWARDED_FOR']."\n";
  echo "PHP Version : ".phpversion()."\n";
?>
```

8 - Services multiples

Dans une architecture orienté conteneur, donc micro-services, un conteneur n'est censé contenir qu'un seul service lié à son rôle. Toutefois, il peut arriver d'avoir besoin de services annexes au service principal dans le même conteneur, ce qui complique le point d'entrée.

L'une des solutions est d'utiliser « supervisor », qui va se charger d'exécuter tous les services voulus. Pour cela il faut :

- Installer le paquet « supervisor » lors du Build,
- Créer un fichier de configuration « supervisord.conf » par exemple, pour définir son utilisation,
- Le recopier lors du Build dans le répertoire « /etc/supervisor/conf.d/supervisord.conf »,
- Définir le point d'entrée « /usr/bin/supervisord »
- Ne pas oublier d'exposer tous les ports liés à tous les services,

Le fichier de configuration de « supervisor » est constitué de la manière suivante :

```
[supervisord]
nodaemon=true

[program:<nom1>]
command=<commande qui lance le service en avant plan>

[program:<nom2>]
command=<commande qui lance le service en avant plan>
```