BUT 2

Docker en pratique : cours 2

Créer ses propres conteneurs

Samuel Delepoulle & Nicolas Condette

Images Docker

- nombreuses images disponibles sur le Docker Hub : https://hub.docker.com/
- plus de 100.000 images disponibles dont de nombreuses officielles
  - systèmes : ubuntu, debian, alpine ...
  - serveurs : apache, nginx, tomcat ...
  - bases de données : mariadb, mysql, postgres, mongoDB ...
  - langages : python, php, java, node, ...
- mais il est souvent nécessaire de personnaliser les images
- utilisation du fichier `Dockerfile` (« recette » de fabrication)

Dockerfile

Format général :

```
# Comment
INSTRUCTION arguments
```

Exemple

```
FROM debian:latest
RUN apt update && apt install -y iproute2 iputils-ping
CMD ["/bin/echo","Le conteneur est démarré"]
```

Utilisation du Dockerfile

construction du conteneur

```
docker build -t mydebian .
```

exécution

```
docker run mydebian
```

Mais la tâche principale se termine arrêt du conteneur

```
docker run --entrypoint /bin/sh -itd mydebian
```

Variables d'environnement

- Déclarées avec `ENV`
- Utilisation `$ENV` ou `${ENV}`

Exemple :

```
FROM busybox
ENV FOO=/bar
WORKDIR ${FOO}   # WORKDIR /bar
ADD . $FOO       # ADD . /bar
COPY \$FOO /quux # COPY $FOO /quux
```

Instructions du Dockerfile

Instruction `FROM`

- Permet d'utiliser une image de base. Souvent debian, Ubuntu, alpine.
- commande **obligatoire**.

image	version	size
debian	Debian GNU/Linux 11	124 MB
ubuntu	Ubuntu 22.04.1 LTS	77 MB
alpine	Alpine Linux 3.19	7,38 MB

Instruction `COPY`

```
COPY [--chown=<user>:<group>] <src>... <dest>
COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

- Copie des fichiers depuis un répertoire local dans le système de fichier du conteneur.

exemple

```
COPY hom* /mydir/
```

Recopie tous les fichiers qui commencent par "hom" du système dans le répertoire `/mydir/` de l'image.

Instruction `ADD`

Fonctionnement identique mais plus d'options :

- peut utiliser des URLs
- peut utiliser des fichiers compressés

Exemples :

```
# syntax=docker/dockerfile-upstream:master-labs
FROM alpine
ADD --keep-git-dir=true https://github.com/moby/buildkit.git#v0.10.1 /buildkit
```

Instruction `RUN`

- Permet d'exécuter des commandes shell dans le conteneur (pour installer ou configurer)

Exemple :

```
RUN apt update && apt install -y iproute2 iputils-ping
```

Deux formes :

- forme "shell" : `RUN <command>` qui appelle automatiquement `/bin/sh -c`
- forme "exec" : `RUN ["executable", "param1", "param2"]` = permet d'utiliser un autre interpréteur :

```
RUN ["/bin/bash", "-c", "echo hello"]
```

Instruction `WORKDIR`

- Permet de spécifier le répertoire de travail qui sera utilisé (pour RUN, CMD, ENTRYPOINT, COPY et ADD).

Exemple

```
WORKDIR /usr/src/app
```

Si le répertoire n'existe pas, il est alors créé.

Instruction `EXPOSE`

Permet de spécifier le protocole (TCP par défaut) et les ports écoutés

Exemple : pour exposer le port 80 en TPC et UDP :

```
EXPOSE 80/tcp
EXPOSE 80/udp
```

attention

Expose ne publie pas le port, uniquement signaler qu'il est exposé. Pour publier il faut le faire à l'exécution :

`docker run -p 80:80/tcp -p 80:80/udp ...`

ou

`docker run -P` = publier tous les ports exposés

Instruction `CMD`

- Permet de spécifier l'instruction utilisée pour exécuter le conteneur
- Un seul CMD dans le DockerFile (en général = dernière instruction)

Exemple

```
CMD  echo "This is a test." | wc -
```

Peut-être redéfini à l'exécution :

`docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]`

3 formes :

- `CMD ["executable","param1","param2"]` (à préférer : "exec form")
- `CMD ["param1","param2"]` (avec ENTRYPOINT)
- `CMD command param1 param2` ("shell form")

Instruction `ENTRYPOINT`

Idem que `CMD` mais ne peut pas être redéfini à l'exécution

Peuvent être combinés :

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

Exemple :

```
FROM python:3
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install --no-cache-dir --upgrade pip \
  && pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 80
CMD ["python", "./your-daemon-or-script.py"]
```