ENTRYPOINT et CMD

Nous allons illustrer sur plusieurs exemples l'utilisation des instructions ENTRYPOINT et CMD. Ces instructions sont utilisées dans un Dockerfile pour définir la commande qui sera lancée dans un container.

Format

Dans un Dockerfile, les instructions ENTRYPOINT et CMD peuvent être spécifiées selon 2 formats:

- le format shell, ex: ENTRYPOINT /usr/bin/node index.js.
  Une commande spécifée dans ce format sera exécutée via un shell présent dans l'image. Cela peut notamment poser des problématiques car les signaux ne sont pas forwardés aux processus forkés.

- le format exec, ex: CMD ["node", "index.js"].
  Une commande spécifiée dans ce format ne nécessitera pas la présence d'un shell dans l'image. On utilisera souvent le format exec pour ne pas avoir de problème si aucun shell n'est présent.

Ré-écriture à l'exécution d'un container

ENTRYPOINT et CMD sont 2 instructions du Dockerfile, mais elle peuvent cependant être écrasées au lancement d'un container:

- pour spécifier une autre valeur pour l'ENTRYPOINT, on utilisera l'option --entrypoint, par exemple: docker container run –entrypoint echo alpine
- pour spécifier une autre valeur pour CMD, on précisera celle-ci après le nom de l'image, par exemple: docker container run alpine echo "hello"

Instruction ENTRYPOINT utilisée seule

L'utilisation de l'instruction ENTRYPOINT seule permet de créer un wrapper autour de l'application. Nous pouvons définir une commande de base et lui donner des paramètres suplémentaires, si nécessaire, au lancement d'un container.

Dans ce premier exemple, vous allez créer un fichier Dockerfile-v1 contenant les instructions

suivantes:

```
FROM alpine
ENTRYPOINT ["ping"]
```

Créez ensuite une image, nommée ping:1.0, à partir de ce fichier.

```
$ docker image build -f Dockerfile-v1 -t ping:1.0 .
```

Lancez maintenant un container basé sur l'image ping:1.0

```
$ docker container run ping:1.0
```

La commande ping est lancée dans le container (car elle est spécifiée dans ENTRYPOINT), ce qui produit le message suivant:

```
BusyBox v1.26.2 (2017-05-23 16:46:25 GMT) multi-call binary.

Usage: ping [OPTIONS] HOST

Send ICMP ECHO_REQUEST packets to network hosts

        -4,-6           Force IP or IPv6 name resolution
        -c CNT          Send only CNT pings
        -s SIZE         Send SIZE data bytes in packets (default:56)
        -t TTL          Set TTL
        -I IFACE/IP     Use interface or IP address as source
        -W SEC          Seconds to wait for the first response (default:10)
                        (after all -c CNT packets are sent)
        -w SEC          Seconds until ping exits (default:infinite)
                        (can exit earlier with -c CNT)
        -q              Quiet, only display output at start
                        and when finished
        -p              Pattern to use for payload
```

Par défaut, aucune machine hôte n'est ciblée, et à chaque lancement d'un container il est nécessaire de préciser un FQDN ou une IP. La commande suivante lance un nouveau container en lui donnant l'adresse IP d'un DNS Google (8.8.8.8), nous ajoutons également l'option -c 3 pour limiter le nombre de ping envoyés.

```
$ docker container run ping:1.0 -c 3 8.8.8.8
```

Nous obtenons alors le résultat suivant.

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=37 time=8.731 ms
64 bytes from 8.8.8.8: seq=1 ttl=37 time=8.503 ms
64 bytes from 8.8.8.8: seq=2 ttl=37 time=8.507 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 8.503/8.580/8.731 ms
```

La commande lancée dans le container est donc la concaténation de l'ENTRYPOINT et de la commande spécifiée lors du lancement du container (tout ce qui est situé après le nom de l'image).

Comme nous pouvons le voir dans cet exemple, l'image que nous avons créée est un wrapper autour de l'utilitaire ping et nécessite de spécifier des paramêtres supplémentaires au lancement d'un container.

Instructions CMD utilisée seule

De la même manière, il est possible de n'utiliser que l'instruction CMD dans un Dockerfile, c'est d'ailleurs très souvent l'approche qui est utilisée car il est plus simple de manipuler les instructions CMD que les ENTRYPOINT.

Créez un fichier Dockerfile-v2 contenant les instructions suivantes:

```
FROM alpine
CMD ["ping"]
```

Créez une image, nommée ping:2.0, à partir de ce fichier.

```
$ docker image build -f Dockerfile-v2 -t ping:2.0 .
```

Si nous lançons maintenant un nouveau container, il lancera la commande ping comme c'était

le cas avec l'exemple précédent dans lequel seul l'ENTRYPOINT était défini.

```
$ docker container run ping:2.0
BusyBox v1.26.2 (2017-05-23 16:46:25 GMT) multi-call binary.

Usage: ping [OPTIONS] HOST

Send ICMP ECHO_REQUEST packets to network hosts

        -4,-6           Force IP or IPv6 name resolution
        -c CNT          Send only CNT pings
        -s SIZE         Send SIZE data bytes in packets (default:56)
        -t TTL          Set TTL
        -I IFACE/IP     Use interface or IP address as source
        -W SEC          Seconds to wait for the first response (default:10)
                        (after all -c CNT packets are sent)
        -w SEC          Seconds until ping exits (default:infinite)
                        (can exit earlier with -c CNT)
        -q              Quiet, only display output at start
                        and when finished
        -p              Pattern to use for payload
```

Nous n'avons cependant pas le même comportement que précédemment, car pour spécifier la machine à cibler, il faut redéfinir la commande complète à la suite du nom de l'image.

Si nous ne spécifions que les paramètres de la commande ping, nous obtenons un message d'erreur car la commande lancée dans le container ne peut pas être interpretée.

```
$ docker container run ping:2.0 -c 3 8.8.8.8
```

Vous devriez alors obtenir l'erreur suivante:

```
container_linux.go:247: starting container process caused "exec: \"-c\":
executable file not found in $PATH"
docker: Error response from daemon: oci runtime error: container_linux.go:247:
starting container process ca
used "exec: \"-c\": executable file not found in $PATH".
ERRO[0000] error getting events from daemon: net/http: request canceled
```

Il faut redéfinir la commande dans sa totalité, ce qui est fait en la spécifiant à la suite du nom de l'image

```
$ docker container run ping:2.0 ping -c 3 8.8.8.8
```

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=37 time=10.223 ms
64 bytes from 8.8.8.8: seq=1 ttl=37 time=8.523 ms
64 bytes from 8.8.8.8: seq=2 ttl=37 time=8.512 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 8.512/9.086/10.223 ms
```

Instructions ENTRYPOINT et CMD

Il est également possible d'utiliser ENTRYPOINT et CMD en même temps dans un Dockerfile, ce qui permet à la fois de créer un wrapper autour d'une application et de spécifier un comportement par défaut.

Nous allons illustrer cela sur un nouvel exemple et créer un fichier Dockerfile-v3 contenant les instructions suivantes:

```
FROM alpine
ENTRYPOINT ["ping"]
CMD ["-c3", "localhost"]
```

Ici, nous définissons ENTRYPOINT et CMD, la commande lancée dans un container sera la concaténation de ces 2 instructions: ping -c3 localhost.

Créez une image à partir de ce Dockerfile, nommez la ping:3.0, et lançez un nouveau container à partir de celle-ci.

```
$ docker image build -f Dockerfile-v3 -t ping:3.0 .
$ docker container run ping:3.0
```

Vous devriez alors obtenir le résultat suivant:

```
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: seq=0 ttl=64 time=0.062 ms
64 bytes from 127.0.0.1: seq=1 ttl=64 time=0.102 ms
64 bytes from 127.0.0.1: seq=2 ttl=64 time=0.048 ms

--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.048/0.070/0.102 ms
```

Nous pouvons écraser la commande par défaut et spécifier une autre adresse IP

```
$ docker container run ping:3.0 8.8.8.8
```

Nous obtenons alors le résultat suivant:

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=38 time=9.235 ms
64 bytes from 8.8.8.8: seq=1 ttl=38 time=8.590 ms
64 bytes from 8.8.8.8: seq=2 ttl=38 time=8.585 ms
```

Il faut alors faire un CTRL-C pour arrêter le container car l'option -c3 limitant le nombre de ping n'a pas été spécifiée.

Cela nous permet à la fois d'avoir un comportement par défaut et de pouvoir facilement le modifier en spécifiant une autre commande.