Docker compose introduction

Exécution d'un conteneur avec Docker Compose

Docker Hub, comprend une simple image «Hello World» à des fins de démonstration et de test. Il illustre la configuration minimale requise pour exécuter un conteneur à l'aide de Docker Compose: un fichier YAML qui appelle une seule image.

Créez d'abord un répertoire pour notre fichier YAML:

```
[root@localhost ~]# mkdir hello-world && cd hello-world
```

Créez maintenant le fichier YAML:

```
[root@localhost hello-world]# vim docker-compose.yml

services:
  my-test:
    image: hello-world
```

Dans le bloc « services : », la première ligne fera partie du nom du conteneur. La deuxième ligne spécifie l'image à utiliser pour créer le conteneur. Lorsque vous exécutez la commande, *docker compose up* il recherchera une image par le nom spécifié, hello-world.

Maintenant, toujours dans le répertoire ~/hello-world, exécutez la commande suivante pour créer le conteneur:

```
[root@localhost hello-world]# docker compose up
Pulling my-test (hello-world:)...
Trying to pull repository docker.io/library/hello-world ...
latest: Pulling from docker.io/library/hello-world
1b930d010525: Pull complete
Digest:
sha256:f9dfddf63636d84ef479d645ab5885156ae030f611a56f3a7ac7f2fdd86d7e4e
Status: Downloaded newer image for docker.io/hello-world:latest
Creating hello-world_my-test_1 ... done
Attaching to hello-world_my-test_1
my-test_1  |
my-test_1  | Hello from Docker!
my-test_1  | This message shows that your installation appears to be
working correctly.
...
hello-world_my-test_1 exited with code 0
```

Utilisez la commande *docker ps -a* pour afficher tous les conteneurs

```
[root@localhost hello-world]# docker ps -a
CONTAINER ID  IMAGE        COMMAND   CREATED  STATUS    PORTS         NAMES
3ef03bc24e53  hello-world  "/hello"  2 min    Exited  hello-world_my-test_1
```

Apprendre les commandes Docker compose

La commande docker compose fonctionne sur une base par répertoire. Vous pouvez avoir plusieurs groupes de conteneurs Docker exécutés sur une seule machine - créez simplement un répertoire pour chaque conteneur et un fichier docker-compose.yml pour chaque répertoire.

Lors de l'exécution en production, il est plus robuste d'avoir docker compose qui agit comme un service. Une façon simple de le faire est d'ajouter l'option -d lors du *up* de votre session:

```
[root@localhost hello-world]# docker compose up -d
Starting hello-world_my-test_1 ... done
```

Pour afficher votre groupe de conteneurs Docker (arrêtés et en cours d'exécution), utilisez la commande suivante:

```
[root@localhost hello-world]# docker compose ps -a
        Name            Command   State    Ports
-------------------------------------------------
hello-world_my-test_1   /hello    Exit 0
```

L'exemple «Hello World» se ferme après son exécution, donc pour tester plus, démarrez un conteneur qui continuera de fonctionner. Pour les besoins de ce Lab, utilisez l'image Nginx de Docker Hub.

Créez un nouveau répertoire nommé nginx et accédez-y:

```
mkdir ~/nginx
cd ~/nginx
```

Ensuite, créez un fichier docker-compose.yml dans votre répertoire et éditez-le :

```
[root@localhost nginx]# vim docker-compose.yml
nginx:
  image: nginx
```

Lors de l'écriture d'un fichier docker-compose, nous ne sommes pas à l'abri d'une erreur. Pour éviter au maximum cela, vous devez utiliser la commande :

```
[root@localhost nginx]# docker compose config
services:
  nginx:
    image: nginx
    network_mode: bridge
version: '2.1'
```

Démarrez le conteneur Nginx en tant que processus d'arrière-plan avec la commande suivante:

```
[root@localhost nginx]# docker compose up -d
Creating nginx_nginx_1 ... done
[root@localhost nginx]# docker compose ps
    Name              Command          State   Ports
-----------------------------------------------------
nginx_nginx_1   nginx -g daemon off;   Up      80/tcp
```

```
[root@localhost nginx]# docker container inspect nginx_nginx_1 | grep -i
ipaddress
            "IPAddress": "172.17.0.2",
[root@localhost nginx]# curl http://172.17.0.2
<!DOCTYPE html>
<html>
...
<h1>Welcome to nginx!</h1>
...
</html>
```

Votre stack Docker Compose est maintenant fonctionnelle, et l'ensemble des services répondent bien ; mais vous pourriez avoir besoin de voir les logs de vos conteneurs. Pour cela, vous devez utiliser la commande:

```
[root@localhost nginx]# docker compose logs
Attaching to nginx_nginx_1
nginx_1  | 172.17.0.1 - - [24/Mar/2020:00:54:23 +0000] "GET / HTTP/1.1" 200
612 "-" "curl/7.29.0" "-"
```

Si vous souhaitez arrêter une stack Docker Compose sans supprimera les différentes ressources créées par votre stack :

```
[root@localhost nginx]# docker compose stop
Stopping nginx_nginx_1 ... done
```

Si vous lancez à nouveau un *docker compose up -d*, l'ensemble de votre stack sera tout de suite à nouveau fonctionnelle.

Si vous souhaitez supprimer l'ensemble de la stack Docker Compose, vous devez utiliser la commande *docker compose down* qui détruira l'ensemble des ressources créées.

```
[root@localhost nginx]# docker compose down
Removing nginx_nginx_1 ... done
[root@localhost nginx]# docker compose ps -a
Name   Command   State   Ports
------------------------------
```