Université Gaston Berger
UFR SAT
CFPP – MaDSI (2017)

Docker

Commandes de base

1. Commande “info”

Donne des informations sur la configuration du docker sur votre machine.

```
$ docker info
```

La sortie de la commande ressemblera à la liste ci-dessous

```
Containers: 0
Images: 301
Server Version: 1.9.0
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 301
 Dirperm1 Supported: false
Execution Driver: native-0.2
Logging Driver: json-file
Kernel Version: 3.13.0-32-generic
Operating System: Ubuntu 14.04.1 LTS
CPUs: 4
Total Memory: 3.141 GiB
Name: ubuntu
ID: CS3B:XJ7B:DEWA:NRXG:VGDY:HYWO:N4NA:FGXM:WYME:FW5C:WYJN:IHVJ
Username: rajdeepd
Registry: https://index.docker.io/v1/
WARNING: No swap limit support
```

2. Créez un conteneur et entrez son shell

```
$ sudo docker run -i -t debian / bin / bash
```

Cela devrait vous donner une nouvelle invite de commande dans le conteneur, très similaire à si vous aviez été installé dans une machine distante. Dans ce cas, les drapeaux -i et -t dire à Docker que nous voulons une session interactive avec un tty attaché. La commande / bin / bash donne un shell bash. Lorsque vous quittez le shell, le conteneur s'arrêtera - les conteneurs ne fonctionnent que pendant leur processus principal.

```
$ docker run debian echo hello-world
```

Hello World

3. Créez un conteneur avec un nom

Vous pouvez utiliser le paramètre de ligne de commande -h pour spécifier un nom de conteneur.

```
$ docker run -h CONTAINER1 -i -t debian /bin/bash
```

La sortie de la commande ci-dessus ouvrira une tty à l'intérieur du conteneur:

```
root@CONTAINER1: / #
```

4. Créer un conteneur avec un mode réseau

Le mode conteneur peut être spécifié à l'aide du flag:

code: “-net= <NETWORK_MODE>

où

```
$ docker run -h CONTAINER2 -i -t --net = "pont" debian/bin/bash
```

5. Liste des conteneurs docker en cours d'exécution

```
$ docker ps-a
```

6. Inspecter un conteneur

```
$ docker inspect hopeful_pare
```

La sortie sera un fichier JSON.

7. Démarrer un conteneur arrêté

```
$ docker start hopeful_pare
```

Où hopeful_pare est le nom du conteneur.

8. Entrez le shell d'un conteneur lancé

```
$ docker attach hopeful_pare
```

Où hopeful_pare est le nom du conteneur.

9. Détacher d'un conteneur

`Docker run -t -i` → peut être détaché avec ^ P ^ Q et rattaché à l'attache du docker
`Docker run -i` → ne peut pas être détaché avec ^ P ^ Q; Perturbera stdin
`Docker run` → ne peut pas être détachée avec ^ P ^ Q;
Peut le client SIGKILL; Peut se rattacher à l'attache du docker

10. Docker Logs

Si vous exécutez cette commande avec le nom de votre conteneur, vous devez obtenir une liste des commandes exécutées dans le conteneur.

```
$ docker logs hopeful_pare
```

Où hopeful_pare est le nom du conteneur.

11. Retrait d'un seul conteneur

```
$ docker rm hopeful_pare
```

12. Suppression de tous les conteneurs

```
$ docker rm `docker ps --no-trunc -aq`
```