Bruno Simões <brunogsimoes@gmail.com>

Legend

docker-machines

myserver

Image names

ubuntu, nginx, redis, myredis

Container names or commit ids

mydb container name
c7777 commit id (see docker ps -a)

Docker Machine

Start a machine

```
$> docker-machine start myserver
```

Configure docker to use a specific machine

```
$> eval $(docker-machine env myserver)
```

Manage Images

Build an image from Dockerfile in current directory

```
$> docker build --tag ubuntu .
```

Force rebuild of Docker image

```
$> docker build --no-cache .
```

Convert a container to image

```
$> docker commit mydb myredis
```

List local available images

```
$> docker images
```

Delete an image from local store

```
$> docker rmi ubuntu
```

Remove all unused images

```
$> docker rmi $(docker images -q -f "dangling=true"
```

Manage Containers

List running containers

```
$> docker ps
```

List all containers (running & stopped)

```
$> docker ps -a
```

Delete all stopped containers

```
$> docker rm $(docker ps --filter status=exited -q)
```

List all containers with a specific label

```
$> docker ps –filter label=mykey
```

Inspect container metadata

```
$> docker inspect mydb
```

Query a specific metadata of a running container

```
$> docker inspect -f '{{ .NetworkSettings.IPAddress }}'
mydb
```

Manage Volumes

Create a local volume

```
$> docker volume create --name myvol
```

Destroy a volume

```
$> docker volume rm myvol
```

List volumes

```
$> docker volume ls
```

Use Containers

Start a container in background (-d)

```
$> docker run -d nginx
```

Start in background, expose port 5000 externally and map to port 80

```
$> docker run -p 5000:80 -d nginx
```

Start an interactive container and connect it to the terminal

```
$> docker run -it ubuntu bash
```

Start a container and remove automatically after it exits

```
$> docker run --rm ubuntu bash
```

Start a container with a specific name

```
$> docker run --name mydb redis
```

Restart a stopped container

```
$> docker start mydb
```

Stop a running container through SIGTERM

```
$> docker stop mydb
```

Stop a running container through SIGKILL

```
$> docker kill mydb
```

Combine multiple instructions

```
$> docker run
            --rm  remove automatically after it exits
             -it  connect container to the terminal
 --name webserver  name the container
        -p 80:80  expose port 80 externally
           nginx  the image for the container
         /bin/sh  the command to run inside
```

Debug

Run another process in running container

```
$> docker exec -it mydb bash
```

Follow live logs (-f) with timestamps (-t) of container

```
$> docker logs -ft mydb
```

Print the last 100 lines of a container

```
$> docker logs --tail 100 mydb
```

Show exposed ports of a container

```
$> docker port mydb
```

Security

Limit the max amount of memory a container can use

```
$> docker run -m 512m ubuntu
```

Change the container's CPU share weighting relative to the weighting of all other running containers (1024 by default).

```
$> docker run -d -c 512 ubuntu
```

Manage Networks

List all local network

```
$> docker network ls
```

Create a local network

```
$> docker network create mynet
```

Create a network and specify a subnet

```
$> docker network create --subnet 10.1.0.0/24 --gateway
10.1.0.1 -d overlay mynet
```

Attach a container to a network on start

```
$> docker run -d --net mynet redis
```

Connect a running container from a network

```
$> docker network connect mynet mydb
```

Disconnect container to a network

```
$> docker network disconnect mynet mydb
```

Manage Swarms

Initialize a swarm mode and listen on a specific interface

```
$> docker swarm init --advertise-addr 10.1.0.2
```

Join an existing swarm as a manager node

```
$> docker swarm join --token manager-token
10.1.0.2:2377
```

Join an existing swarm as a worker node

```
$> docker swarm join --token worker-token 10.1.0.2:2377
```

List all nodes participating in a swarm

```
$> docker node ls
```

Create a service and deploy 4 instances

```
$> docker service create --replicas 4 -p 80:80 –name serv
nginx
```

Re-scale a service

```
$> docker service scale serv=6
```

List the services running in a swarm

```
$> docker service ls
```

List the tasks of a service

```
$> docker service tasks serv
```

Ship

Pull an image from a registry

```
$> docker pull alpine:3.4
```

Retag a local image

```
$> docker tag alpine:3.4 myrepo/alpine:3.4
```

Login into a registry (the Docker Hub by default)

```
$> docker login my.registry.com:8000
```

Push an image to a registry

```
$> docker push myrepo/alpine:3.4
```