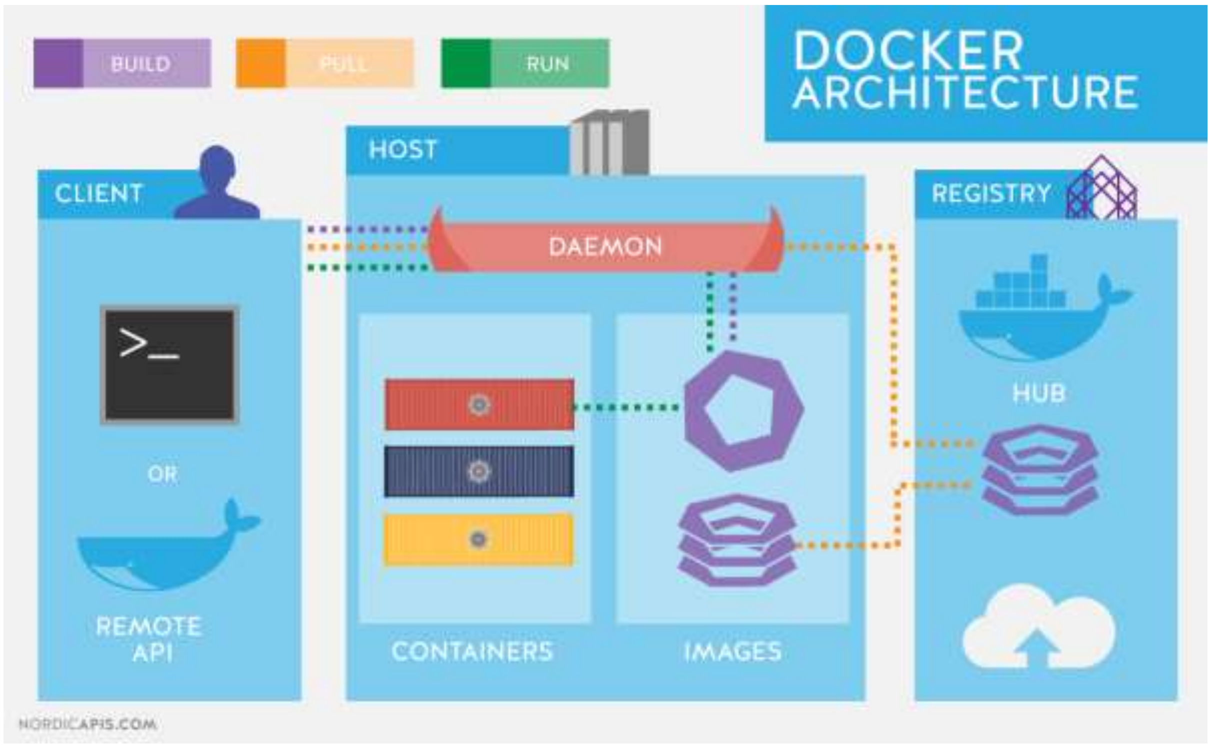
Docker Notes - Best Practices

Table Content

1. Docker Architecture



2. Setup

- CentOS

3. Cheat Sheet

The Ultimate Docker Cheat Sheet | dockerlabs (collabnix.com)

4. Docker CLI

4.1. List all docker image

docker images

4.2 Running ubuntu bash

- image --> docker run --> running container --> stopped container --> docker commit --> new images
- image is not change.

-ti = terminal keyboardInteractive

docker run -ti ubuntu:latest

4.3. List of containers

docker ps

List of docker with format

docker ps --format $FORMAT

List all (-a)

docker ps -a

List last (-l)

docker ps -l

4.4. Docker commit

- create new image from container

docker commit <docker id | name> [<new-image-name>]

- set tag for new images

docker tag < sha256 > < tag-name >

ex:

docker tag 52caa40054059fc07e4148337efa0a937799dc25ddc6b2e9f4d7deec4cf63177 my-image

test:

docker run -ti my-image

5. Running processes in container

- --rm : do not keep container after finish process

docker run --rm -ti ubuntu sleep 5

docker run --rm -ti ubuntu cat /etc/hosts

docker run -ti ubuntu bash -c "sleep 5; echo done"

- -d : (detach) run docker process in background

docker run -d -ti ubuntu bash

- attach

docker ps -l [--format $FORMAT]

docker attach <container-name>

- detach, leave it running in background Control P + Control Q
- add another process in existed container

docker exec -ti <container-name> <command>

ex:

docker exec -ti container_name bash

5.1. View output of container

docker logs <container-name>

ex:

docker run --name my-container -d ubuntu bash -c "more /etc/hosts"

docker logs my-container

5.2. Kill a container

docker kill <container-name>

6. Manage container

6.1. Memory limits

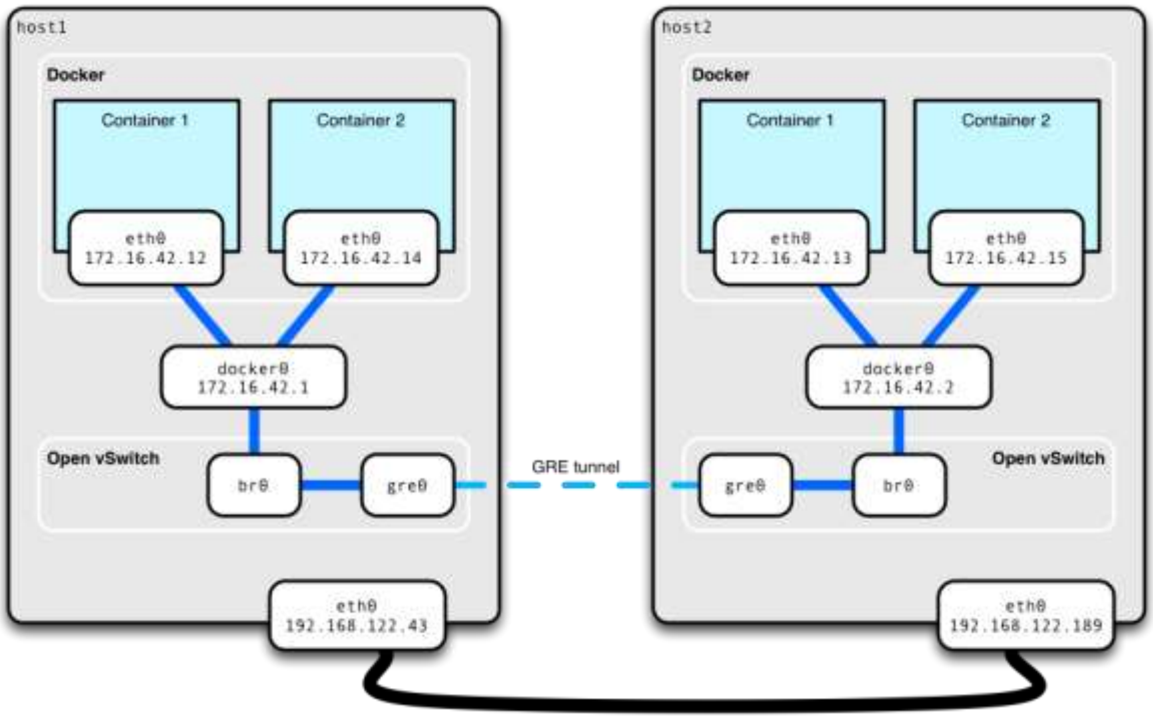
docker run --memory <maximum-allowed> <image-image> <command>

6.2. CPU limits

docker run --cpu-shares relative to other containers

docker run --cpu-quota to limit it in general

7. Networking



ex:

- echo-server (-p port_in[:port_out])

```
docker run --rm -ti -p 45678:45678 -p 45679:45679 --name echo-server ubuntu:14.04 bash


nc -lp 45678 | nc -lp 45679
```

7.1. Getting container ip

```
docker ps
docker inspect <container-id> | grep IP
nc <container-ip> <port>
```

ex1:

```
nc 172.17.0.2 45678
nc 172.17.0.2 45679
```

ex2:

```
docker run --rm -ti -p 45678 -p 45679 --name echo-server ubuntu:14.04 bash


docker port echo-server
```

7.2. UDP ports

```
docker run -p ousite-port:insite-port/protocol(tcp/udp)
```

ex:

```
docker run -p 1234:1234/udp
```

8. Connecting between Containers

Client Container-->Host Network--->Virtual Network ---> Server Container

ex:

```
docker run -ti --rm -p 1234:1234 unbuntu:14.04 bash
nc -lp 1234
```

8.1. host -> container

```
docker ps -l
docker inspect <container-id> | grep IP
nc <container-ip> 1234
```

8.2. container --> container

```
docker run -ti --rm ubuntu:14.04 bash
nc <container-id> 1234
```

9. container connects to another container directly.

- server

```
docker run -ti --rm --name server ubuntu:14.04 bash
nc -lp 1234
```

- client

```
docker run --rm -ti --link server --name client ubuntu:14.04 bash
nc server 1234
```

- Link directly:
- A service with its DB - not good
- Automatically assigns a hot name
- That links can break when containers restart

9.1. Making Links Not Break

- Docker has private networks.
- Fix the Links
- Must create the networks in advance

```
docker network create <network-name>
```

ex:

- server

```
docker network create example
docker run --rm -ti --net=example --name server ubuntu:14.04 bash
nc
nc -lp 1234
```

- client

```
docker run --rm -ti --link server --net=example --name client ubuntu:14.04 bash
nc server 1234
```

- Now kill the server and restart again.
- The link between server and client does not break.

9.2. Limiting access to only host

```
docker run -p 127.0.0.1:1234:1234/tcp
```

10. Listing images

- List downloaded images

docker images

- Tagging gives images

docker commit <container-id|name> <new-image-name>[:<tag>]

ex:

docker ps -l

docker commit b5938fe91f4c my-image-now

docker images

10.1. Getting images

- for offline work

docker pull

10.2. Removing images

docker rmi <image-name|id>

ex:

docker images

docker rmi my-image

11. Volumes

- Sharing data between containers and containers and host.
- Virtual "dicsc"
- Two types:
  - Persistent : Keep when container went away.
  - Ephemeral: exists in container life.
- Volumes is not a part of image.

11.1. Sharing data with the host

- like VMware.
- Sharing folders with the host ex:

mkdir /home/docker/my-volume

docker run -ti -v=/home/docker/my-volume:/shared-folder ubuntu bash

cd /shared-folder

touch my-data

Press Crtl + D

ls ./my-volume/shared-folder

- Sharing a "single file" into a container

11.2. Sharing Data between Containers

- volumes-from
- Shared disks that exist only as long as they are being used
- Can be shared between containers

ex

- Container #1

docker run -ti -v /shared-data ubuntu bash

echo "hello, is it great!" > /shared-data/my-file

- Container #2

docker ps -l

docker run -ti --volumes-from volume_name ubuntu bash

cat /shared-data/my-file

12. Docker Registries

- Registries and distributes images.

12.1. Finding Images

https://hub.docker.com

docker search centos

- Push image to the world.

docker login

docker pull centos

docker tag centos:123 test/test-image-32:v123.1234

docker push test/test-image-32:v123.1234

Note: Do not push password with the image

13. Dockerfile

- What is it ?
- code to create image

docker build -t name-of-result .

- each line takes the image of previous line and makes another image.
- the previous image is unchanged.

14. References

1. https://docs.docker.com/engine/reference/builder/
2. http://apachebooster.com/kb/wp-content/uploads/2017/09/docker-architecture.png
3. The Ultimate Docker Cheat Sheet | dockerlabs (collabnix.com)
4. http://extremeautomation.io/img/cheatsheets/cheat_sheet_docker_page_1.png