All You Need to Know - Docker

- Containers: They are sealed, self-contained units of software that have everything needed to run a service.
  What you should know(Viewed)

1. Installing Docker

https://docs.docker.com/get-docker/

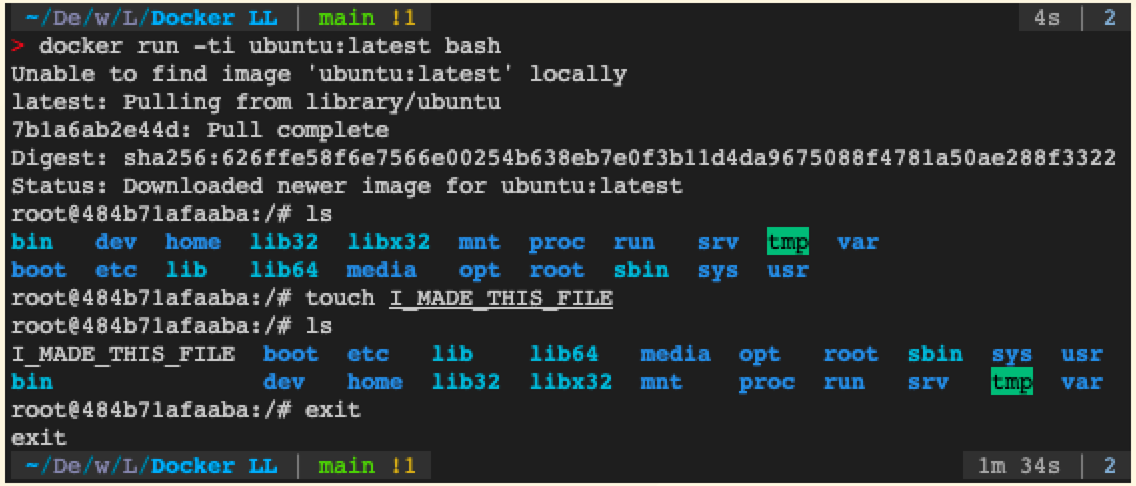
2. Using Docker

The Docker flow: Images to containers**

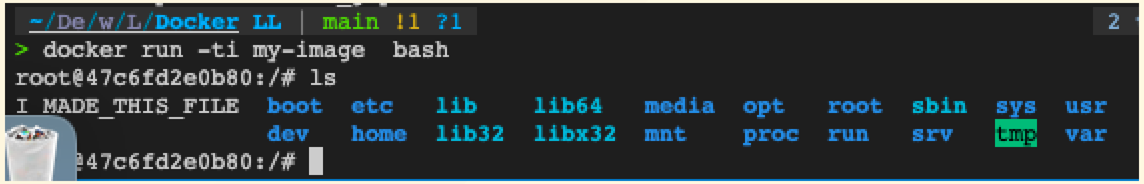
- Image -> running container
- `docker run docker-name`
- `docker ps` -> get information on running container
- image is fixed and does not change

The Docker flow: Containers to images**

- running container -> image (files are stored in stopped container)
- `docker ps -a` all container
- `docker ps -l` last container
- topped container has the software installed in it
- docker commit -> container to new image
- Run bash image `docker run -ti ubuntu:latest bash`
- create and give images name
- Run processes in containers



```
 ~/De/w/L/Docker LL | main !1                                                   4s | 2
> docker run -ti ubuntu:latest bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
7b1a6ab2e44d: Pull complete
Digest: sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a50ae288f3322
Status: Downloaded newer image for ubuntu:latest
root@484b71afaaba:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@484b71afaaba:/# touch I_MADE_THIS_FILE
root@484b71afaaba:/# ls
I_MADE_THIS_FILE  boot  etc   lib    lib64   media  opt   root  sbin  sys  usr
bin               dev   home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@484b71afaaba:/# exit
exit
 ~/De/w/L/Docker LL | main !1                                               1m 34s | 2
```



```
 ~/De/w/L/Docker LL | main !1 ?1                                                   2
> docker run -ti my-image  bash
root@47c6fd2e0b80:/# ls
I_MADE_THIS_FILE  boot  etc   lib    lib64   media  opt   root  sbin  sys  usr
                  dev   home  lib32  libx32  mnt    proc  run   srv   tmp  var
@47c6fd2e0b80:/# 
```

Running process in Docker**

- containers have a min proceed
- one main process that has name
- -d run detached container in the background

3. Under the hood

Registries in detail

- Kernel : controls and organizes storage, programs and so. allocate resources and so on
- docker manges kernel
- cgroup to group process together
- can be client and server
- docker control socket

```
docker run -ti --rm -v /var/run/docker.sock docker sh
```

Docker network

- Ethernet : move frame on wire of wifi
- IP layer : move packets on local network
- routing : forwards packets between networks
- ports: address particular program on a computer
- bridges -> virtual network on computer
- controls the ethenes layer
- docker builds bridges to create virtual network
- Routing -> moving packets
- create fireball rule moves between network

```
docker run -ti --rm --net=host --privileged=true ubuntu bash

apt-get update
apt-get install iptables
```

Exposing port is essentially port forwarding

Process in Docker**

- process init 1-> to many process
- shell -> runs other process -> continuer vanishes
  Intro to orchestration

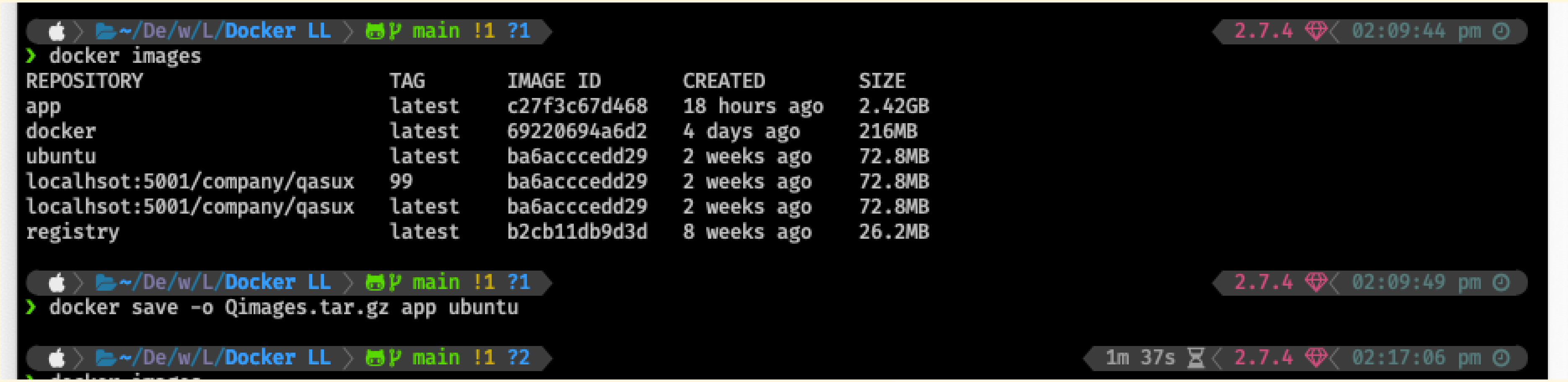
```
-> docker run -ti --rm --name qasim ubuntu bash
-> docker inspect --format '{{.State.Pid}}' qasim
 3134

-> docker run -ti --rm --privileged=true --pid=host ubuntu bash
root@e63d21be00c6:/# kill 3134

root@99fb3caf034e:/# exit?
```

storage in docker**

- actual storage
- forms in logical groups
- filesystem : which file belongs where
- COWS!!! -> copy on write
- base image -> write stuff container sees it with changes no change to the image
- like a file system layer on the image !!! COW COPY ON WRITE
- managing cows
    - layers -> splits them to gzip files and then puts them on container
    - containers are indigent of storage engine
    - some have limited layers
- get the right mount order correct

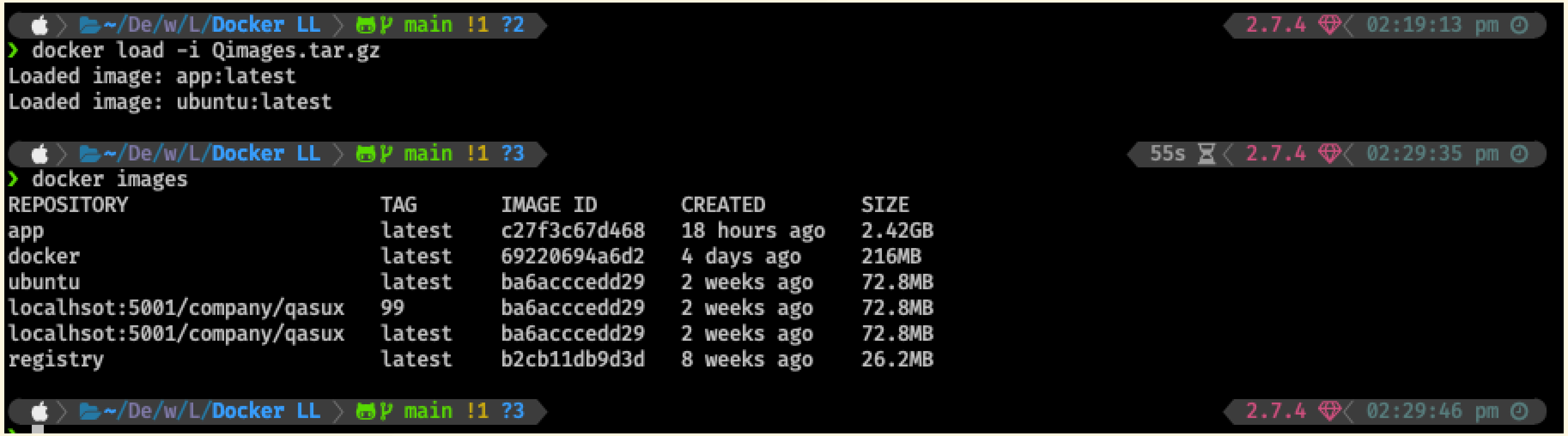
save abd load docker images**

1. save imaged locally



```
 ~/De/w/L/Docker LL   main !1 ?1                                                        2.7.4   02:09:44 pm
❯ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED         SIZE
app                            latest    c27f3c67d468   18 hours ago    2.42GB
docker                         latest    69220694a6d2   4 days ago      216MB
ubuntu                         latest    ba6acccedd29   2 weeks ago     72.8MB
localhsot:5001/company/qasux   99        ba6acccedd29   2 weeks ago     72.8MB
localhsot:5001/company/qasux   latest    ba6acccedd29   2 weeks ago     72.8MB
registry                       latest    b2cb11db9d3d   8 weeks ago     26.2MB

 ~/De/w/L/Docker LL   main !1 ?1                                                        2.7.4   02:09:49 pm
❯ docker save -o Qimages.tar.gz app ubuntu

 ~/De/w/L/Docker LL   main !1 ?2                                           1m 37s   2.7.4   02:17:06 pm
```

2. load Images



```
 ~/De/w/L/Docker LL   main !1 ?2                                                        2.7.4   02:19:13 pm
❯ docker load -i Qimages.tar.gz
Loaded image: app:latest
Loaded image: ubuntu:latest

 ~/De/w/L/Docker LL   main !1 ?3                                              55s   2.7.4   02:29:35 pm
❯ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED         SIZE
app                            latest    c27f3c67d468   18 hours ago    2.42GB
docker                         latest    69220694a6d2   4 days ago      216MB
ubuntu                         latest    ba6acccedd29   2 weeks ago     72.8MB
localhsot:5001/company/qasux   99        ba6acccedd29   2 weeks ago     72.8MB
localhsot:5001/company/qasux   latest    ba6acccedd29   2 weeks ago     72.8MB
registry                       latest    b2cb11db9d3d   8 weeks ago     26.2MB

 ~/De/w/L/Docker LL   main !1 ?3                                                        2.7.4   02:29:46 pm
```