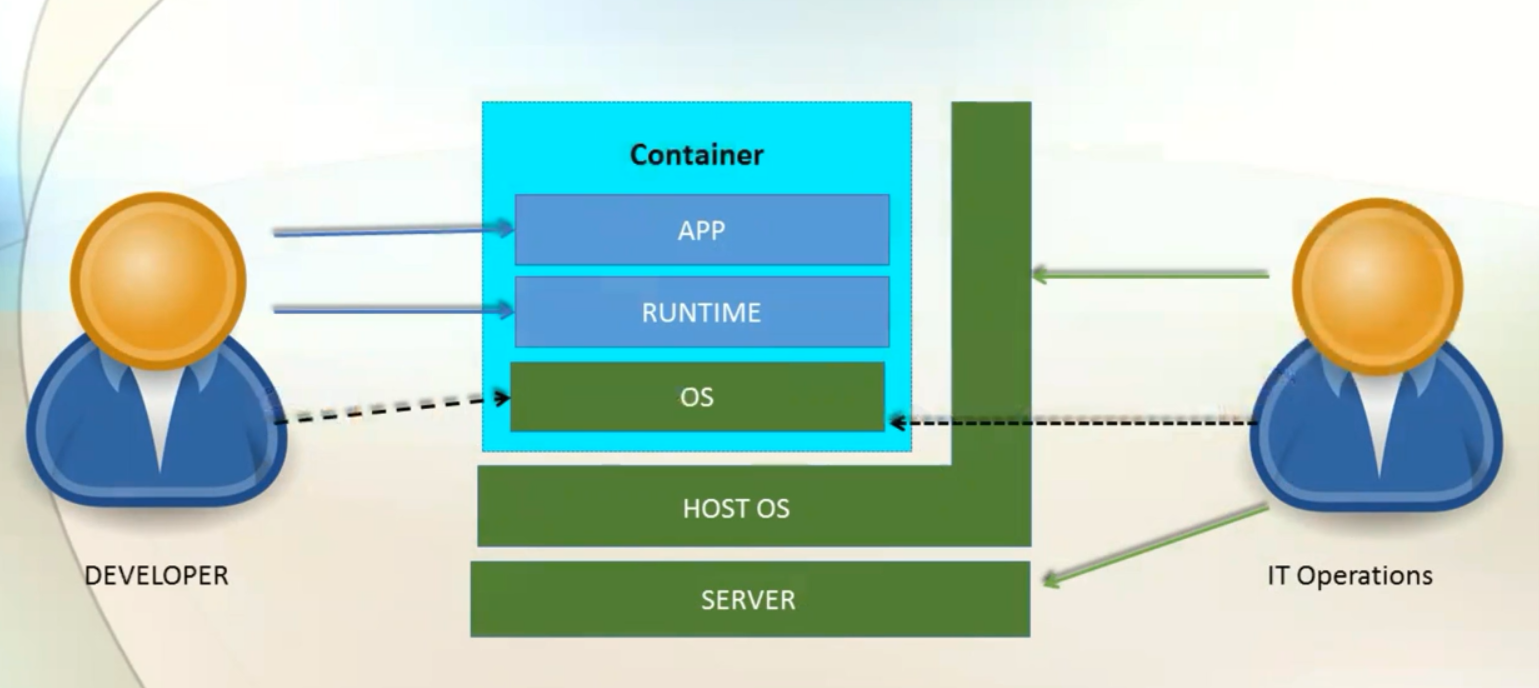


* Containers are isolated from Host OS
* Container Runtime=Docker deamon
* Containers are only use required OS components
* Virtual environment> each app has own OS
* Docker image contains dependencies for applications(i.e package manager for applications)
* Virtualized systems are good at isolating systems and containerized systems are good at isolating applications
* Container=Running image



**Docker enterprise edition(docker EE)** releases are every 3 months

+ Certified images and plugins

+ docker data center

+ vulnerability scans

+ official support

**Docker community edition**

Edge: releases are every 1 month

Stable: release are every 3 month (bugfix 4months later)

**Method1: docker install windows**

https://docs.docker.com/docker-for-windows/install/

! enable container and hyperv feature from windows advanced features

(type1 virtualization is needed)

Docker deamon runs locally and accessable from localhost

**Method2: docker install windows**

<https://github.com/docker/toolbox/releases> (download .exe)

**Docker Engine**

Deamon name: dockerd

REST API: client use to give action to container

Client docker CLI: client use to give action to container (for example containers, networks, images, data volumes)

Docker client >> (daemon) Docker Engine (via rest api or CLI)

**Docker Community Edition (stable-edge version)**

yum install -y yum-utils device-mapper-persistent-data lvm2

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

cat /etc/yum.repos.d/docker-ce.repo

yum install docker -y

systemctl enable docker

systemctl status docker

systemctl start docker

docker info (systemwide info)

docker system df

TYPE TOTAL ACTIVE SIZE RECLAIMABLE

**Images**  5 1 333.8 MB 96.67 MB (28%)

**Containers**  3 3 99.17 MB 0 B (0%)

**Local Volumes** 3 1 6 B 0 B (0%)

docker ps (running containers)

docker ps -a (all containers)

docker images(available images and their tags on local system)

docker pull containername:tag (generally tag=latest is used)

docker run -it --name=containername -p 8080:80/tcp imagename

|  |  |
| --- | --- |
| -p 8080:80 | Map TCP port 80 in the container (insider) to port 8080 on the Docker host. |

(outsider)

docker port containername

80/tcp -> 0.0.0.0:8080

additional flags

e: for environment path

-p: port (can be multiple p use)

--rm: remove container after stopped

-d : container runs in background

--restart on-failure (host reboot’a dayanıklı olmasını sağlar)

-v (mounting volume) $PWD:/containerfolder

--memory=4g (The maximum amount of memory the container can use)

--cpus= "1.5" (guarantees 1.5CPU for the container)

docker rm containername >> removes only specific container

docker rm $(docker ps -a -q) >> removes all stopped containers

docker stop $(docker container ls -a -q) >> stops all running containers

docker start containerid/containername

docker stop containerid/containername

docker exec “containername” CMD (execute a command on container without attaching)

docker attach “containername”

(removing docker image)

docker images

docker rmi imagename:version

docker rmi imageid

seeing container logs

docker container logs -f “containername” (it is tail logs)

getting stats from container

docker container stats

attach Attach to a running container

build Build an image from a Dockerfile

commit Create a new image from a container's changes

cp Copy files/folders between a container and the local filesystem

create Create a new container

diff Inspect changes on a container's filesystem

events Get real time events from the server

exec Run a command in a running container

export Export a container's filesystem as a tar archive

history Show the history of an image

images List images

import Import the contents from a tarball to create a filesystem image

info Display system-wide information

inspect Return low-level information on a container or image

kill Kill a running container

load Load an image from a tar archive or STDIN

login Log in to a Docker registry

logout Log out from a Docker registry

logs Fetch the logs of a container

network Manage Docker networks

pause Pause all processes within a container

port List port mappings or a specific mapping for the CONTAINER

ps List containers

pull Pull an image or a repository from a registry

push Push an image or a repository to a registry

rename Rename a container

restart Restart a container

rm Remove one or more containers

rmi Remove one or more images

run Run a command in a new container

save Save one or more images to a tar archive

search Search the Docker Hub for images

start Start one or more stopped containers

stats Display a live stream of container(s) resource usage statistics

stop Stop a running container

tag Tag an image into a repository

top Display the running processes of a container

unpause Unpause all processes within a container

update Update configuration of one or more containers

version Show the Docker version information

volume Manage Docker volumes

wait Block until a container stops, then print its exit code

**Creating own image on docker**

docker login

username:

password:

docker commit basecontainername username/imagename:v1 (creating custom image)

docker images (verify this image created)

docker push username/imagename:v1

(downloading custom image)

docker pull yaydog/testing:v1

docker run -it --name=mycontainer yaydog/testing:v1

(alternative)

docker commit containername/id (generates code)

docker tag CODE imagename

**GETTING DETAILED INFO ABOUT CONTAINER**

docker inspect containername/id

**NETWORKING**

**(on host)**

sysctl -w net.ipv4.ip\_forward=1

vi /etc/sysctl.conf

net.ipv4.ip\_forward=1

docker pull ubuntu

docker run -it --name=mycontainer -p 222:22 ubuntu

iptables -L > before

apt-get update

apt-get install net-tools

apt-get install iproute2

apt-get install iputils-ping

apt-get install vim

apt-get install openssh-server

ipconfig : get container IP

vim /etc/ssh/sshd\_config

PermitRootLogin yes

/etc/init.d/ssh start

**(changing network settings of container)**

[root@centos1 ~]# docker network ls

**Bridge**, host, none

[root@centos1 ~]# ifconfig

**docker0**: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet **172.17.0.1** netmask 255.255.0.0 broadcast 0.0.0.0

Vethaeef0bd1, Vethaeef0bd2 interface is attached to docker0 bridge

Vethaeef0bd1 attached to eth0 of container1

Vethaeef0bd2 attached to eth0 of container2

docker network inspect bridge : bridge baglı ve calısan container’ların aldıgı IP’ler görülebilir

docker run -it -d --network=none --name=mycontainer2 ubuntu:latest (without internet access)

docker run -it -d --network=bridge --name=mycontainer3 ubuntu:latest (default with internet access)

(linking containers)

docker start mycontainer

docker run -it --link mycontainer:ubuntu --name mycontainer2 ubuntu

mycontainer from Ubuntu image is linked to mycontainer with Ubuntu image

mycontainer2 has mycontainer info in its /etc/hosts

**CREATING NEW NETWORK**

docker network create --driver bridge firstnetwork

docker run -it --name=dev1 -d --restart on-failure --network=firstnetwork centos

docker run -it --name=dev2 -d --restart on-failure --network=firstnetwork centos

docker network inspect firstnetwork (yeni yaratılan makinenin bu network’e oldugu görülür)

docker exec dev1 ping -c 3 dev2 (dev1 makinesinden dev2 makinesine ping yapılabilirliği görülür)

docker DNS server: nameserver 127.0.0.11

**INSTALLATION OF DOCKER CLIENT (insecure)**

(docker host)

# Create the directory to store the configuration file.

mkdir -p /etc/systemd/system/docker.service.d

# Create a new file to store the daemon options.

vi /etc/systemd/system/docker.service.d/options.conf

# Now make it look like this and save the file when you're done:

[Service]

ExecStart=

ExecStart=/usr/bin/dockerd -H unix:// -H tcp://0.0.0.0:2375

# Reload the systemd daemon.

sudo systemctl daemon-reload

# Restart Docker.

sudo systemctl restart docker

(docker client)

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

yum install docker-ce-cli -y

echo "export DOCKER\_HOST=tcp://X.X.X.X:2375" >> ~/.bashrc && source ~/.bashrc

**DOCKER IMAGE BUİLD FROM DOCKER FILE**

docker image build -t web1 .

(first cd to docker image location)

docker image inspect web1 (getting detailed information of image)

(removing images)

docker image rm -f IMAGEID

docker image tag web1 yaydog/web1:latest

docker push yaydog/web1:latest

docker pull yaydog/web1:latest

docker run -it --name=dev -p 5002:5000 -e FLASK\_APP=app.py web1

docker run -it --name=dev3 -p 5002:5000 -d --restart on-failure -e FLASK\_APP=app.py -v $PWD:/app web1

**CREATING DATA VOLUMES AND ATTACH**

docker volume create hostvol

docker volume ls

docker volume inspect hostvol

/var/lib/docker/volumes/hostvol/\_data

(attaching volume to container)

docker run -it --name=dev1 -d --restart on-failure --network=firstnetwork -v hostvol:/home centos

docker volume rm hostvol (removing host data volume)

**COPYING DATA FROM DOCKER HOST TO DOCKER CONTAINER**

docker cp /path/of/the/file <Container\_ID>:/path/of/he/container/folder

**SHARING DATA BETWEEN CONTAINERS**

docker volume create hostvol

docker run -it --name=dev1 -d --restart on-failure --network=firstnetwork -v hostvol:/home centos

docker run -it --name=dev2 -d --restart on-failure --network=firstnetwork ---volumes-from dev1 centos

host üzerindeki /var/lib/docker/volumes/hostvol/\_data

dev1 ve dev2 nin /home altında ortak olarak kullanılır

**RUNNING SCRIPT AT STARTUP OF CONTAINER --dockerfile**

COPY docker-entrypoint.sh /

RUN chmod +x /docker-entrypoint.sh

ENTRYPOINT ["/docker-entrypoint.sh"]

**RUNNING SCRIPT ON RUNNING CONTAINER**

(syntax)

docker run -it dev3 bash -c "cmd1;cmd2;cmd3"

docker exec -it dev3 bash -c "cmd1;cmd2;cmd3"

docker cp /home/script.sh <Container\_ID>:/home

docker exec -it dev3 bash -c "chmod u+x /home/script.sh"

docker exec -it dev3 bash -c "bash /home/script.sh"

**DOCKER CLEANING**

docker system prune -f

WARNING! This will remove:

- all stopped containers

- all volumes not used by at least one container

- all networks not used by at least one container

- all dangling images

docker system prune -a

WARNING! This will remove:

- all stopped containers

- all volumes not used by at least one container

- all networks not used by at least one container

- all images without at least one container associated to them

**DOCKER LOGGING**

docker logs “containername”

**GOLDEN CENTOS IMAGE**

docker pull centos

dockerfile

FROM centos:7

ENV container docker

RUN (cd /lib/systemd/system/sysinit.target.wants/; for i in \*; do [ $i == \

systemd-tmpfiles-setup.service ] || rm -f $i; done); \

rm -f /lib/systemd/system/multi-user.target.wants/\*;\

rm -f /etc/systemd/system/\*.wants/\*;\

rm -f /lib/systemd/system/local-fs.target.wants/\*; \

rm -f /lib/systemd/system/sockets.target.wants/\*udev\*; \

rm -f /lib/systemd/system/sockets.target.wants/\*initctl\*; \

rm -f /lib/systemd/system/basic.target.wants/\*;\

rm -f /lib/systemd/system/anaconda.target.wants/\*;

VOLUME [ "/sys/fs/cgroup" ]

RUN yum install -y openssh-server openssh-clients; systemctl enable sshd

EXPOSE 22

CMD ["/usr/sbin/init"]

docker image build -t mycentos7 .

docker run -it --name=dev -d --restart on-failure mycentos7

docker exec -it -u root dev /bin/bash

passwd root

(install required packages)

docker login

username:

password:

docker commit dev yaydog/mycentos

docker images (verify this image created)

docker push yaydog/mycentos

**SAMPLE DOCKERFILE REFERENCE**

**<https://docs.docker.com/engine/reference/builder/>**

**FROM** centos:latest

**LABEL** maintainer="Yigit Aydog <yaydog@gmail.com>" \

version="1.0" \

description="sample image"

**RUN** yum install -y epel-release passwd openssh-server openssh-clients wget nmap lsof unzip

**RUN** useradd yaydog -d /home/yaydog -m

**EXPOSE** 22/tcp

**RUN** mkdir /project

**WORKDIR** /project

**COPY** . .

**CMD** ["/bin/bash"]