# İlk Kısım - Dersteki Docker Komutlarının Açıklaması

## sudo docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

- Docker file çalıştırma komutu
- OPTIONS:
  - -d:
- docker file'ı arka planda çalıştırır, böylece docker file çalışırken terminali kullanabiliriz
- --rm:
  - docker file çalışmasını bitirdikten sonra container'ı siler
- -e:
- env değişkenleri tanımlama
- --name:
  - container'a isim verme
- COMMAND:
  - herhangi bir terminal komutu çalıştırılabilir

```
emre@emre-VirtualBox:~

emre@emre-VirtualBox:~

emre@emre-VirtualBox:~

emre@emre-VirtualBox:~

emre@emre-VirtualBox:~

esh -c 'echo $MY_ENV_VAR'

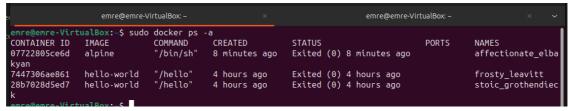
HelloWorld

emre@emre-VirtualBox:~

emre@emre-Virtu
```

# sudo docker ps [OPTIONS]

- Çalışan container'ları listeler
- OPTIONS:
  - -a:
- Tüm container'ları listeler



# sudo docker stop CONTAINER\_ID

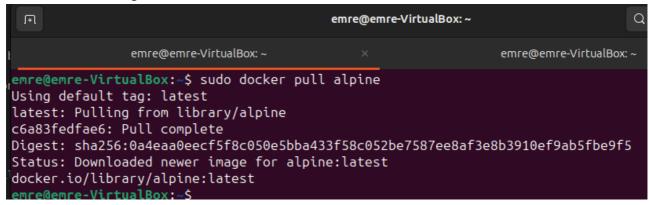
Çalışan bir container'ı durdurur

```
emre@emre-VirtualBox:~ × emre@emre-VirtualBox:~ × × 

<a href="mailto:color: blue color: white;">color: color: blue color: blu
```

sudo docker pull IMAGE\_NAME

• Docker hub'dan image indirme komutu



## sudo docker images

mevcut docker image'larını listeler

```
emre@emre-VirtualBox: ~
                                                                              emre@emre-VirtualBox: ~
emre@emre-VirtualBox:~$ sudo docker images
REPOSITORY
                          IMAGE ID
                                          CREATED
                                                           SIZE
               TAG
                                          45 hours ago
alpine
               latest
                          324bc02ae123
                                                            7.8MB
hello-world
                                                            13.3kB
                          d2c94e258dcb
                                          14 months ago
               latest
 emre@emre-VirtualBox:
```

## sudo docker rmi IMAGE\_ID

mevcut bir docker image'ı siler

```
emre@emre-VirtualBox:~ × emre@emre-VirtualBox:~ × ✓

emre@emre-VirtualBox:~$ sudo docker rmi d2c94e258dcb --force

Untagged: hello-world:latest

Untagged: hello-world@sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6

Deleted: sha256:d2c94e258dcb3c5ac2798d32e1249e42ef01cba4841c2234249495f87264ac5a

emre@emre-VirtualBox:~$
```

sudo docker build -t IMAGE\_NAME:TAG .

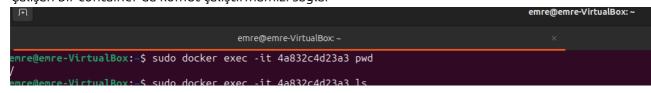
• docker file'dan image oluşturma komutu

## sudo docker push IMAGE\_NAME:TAG

• docker hub'a image yükleme komutu

# sudo docker exec -it CONTAINER\_ID COMMAND

• Calışan bir container'da komut çalıştırmamızı sağlar



#### sudo docker version

• docker versiyonunu döner

```
emre@emre-VirtualBox: ~ ×

cemre@emre-VirtualBox: ~ 

cemre@emre-VirtualBox: ~ 

Docker version 27.1.1, build 6312585

cemre@emre-VirtualBox: ~ $
```

## sudo docker-compose [OPTIONS] [SERVICES]

- OPTIONS:
  - up:
- docker-compose.yml file'ındaki servisleri çalıştırır
- down:
  - docker-compose.yml file'ındaki çalışan servisleri durdurur
- build:
  - docker-compose.yml file'ındaki servislerden image oluşturur
- ps:
- docker-compose.yml file'ındaki çalışan servisleri listeler

#### sudo docker network ls

Docker network'lerini listeler

```
emre@emre-VirtualBox: ~/Desktop/dockerludockerize_me/dockerize_me
     emre@emre-VirtualBox: ~/Desktop/dockerlidockerize_me/dockerize_me
                                                                                        emre@emre-VirtualBox: ~
emre@emre-VirtualBox:~/Desktop/dockerlidockerize_me/dockerize_me$ sudo docker network ls
                            DRIVER
NETWORK ID
                 NAME
                                        SCOPE
c35c0b9f405c
                 bridge
                            bridge
                                        local
4b909b1b260d
                 host
                            host
                                        local
c7378d8ba2ce
                                        local
                 none
```

# sudo docker network create NETWORK\_NAME

docker network oluşturur

## sudo docker network rm NETWORK\_NAME

docker network silme komutu

```
emre@emre-VirtualBox: ~/Desktop/dockerludockerize_me / Q =

emre@emre-VirtualBox: ~/Desktop/dockerludockerize_me × emre@emre-VirtualBox: ~

genre@emre-VirtualBox: ~/Desktop/dockerludockerize_me/dockerize_me$ sudo docker network create homework d4e7fd0d39847f247b9733d67fad72f6cb3197097c65bf8d74bf425e910b897c

emre@emre-VirtualBox: ~/Desktop/dockerludockerize_me/dockerize_me$ sudo docker network rm homework ehomework

ehomework

emre@emre-VirtualBox: ~/Desktop/dockerludockerize_me/dockerize_me$
```

# sudo docker volume ls

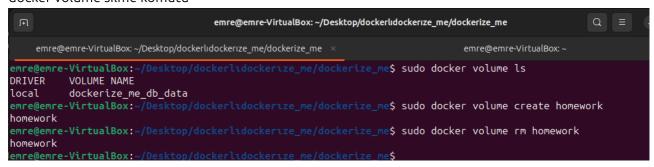
docker volume'lerini listeleme(volume: backup tarzı bir olay)

#### sudo docker volume create VOLUME NAME

docker volume oluşturma komutu

sudo docker volume rm VOLUME NAME

docker volume silme komutu



# İkinci Kısım - Gruba Atılan Dosya İçin Açıklamalar

# Docker File

# FROM php:7.4-apache

• Base image'i belirtmek için yazılan satır

#### WORKDIR /var/www/html

• diğer tüm komutların çalışacağı directory'i belirtiyoruz

# COPY ./app .

• Ana host'taki directory ve file'ları, container'ın file sistemine yükler, yani üstte belirttiğimiz container'ın /var/www/html dizini

# RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

 Container'daki apache configurasyonunda ServerName'i localhost olarak ayarlıyoruz, servername localhost olmayınca hata aldığımız için

#### RUN apt-get update

• container'daki paketleri günceller, amaç: kullanılan paketlerin son versiyonlarını kullanmak

# RUN docker-php-ext-install pdo pdo\_mysql

php'nin mysql ile iletişim kurması için gerekli extention'ların kurulumu

#### EXPOSE 80

 container 80 port'unu dinlemeye başlar ama server 80 portunda başlatılmaz, container çalıştırılacağında -p argümanı ile host için belirleyeceğimiz port'ta başlatılır.

# docker-compose.yml

#### version: '3'

• docker-compose file format'ını belirtir

```
services:
   app:
    build:
       context: .
       dockerfile: Dockerfile
   depends_on:
       - db
   ports:
       - "80:80"
   networks:
       - net
```

- **services içinde**, hangi container'ların çalışacağını belirtiyoruz, buna göre sadece app container'ı çalıştırılacak
  - app -> servisin adı
    - **build içinde**, app servisi için nasıl image build(oluşturmak) edileceği belirtiliyor, buna göre bulunulan directory'deki Dockerfile'a göre oluşturulacak
    - depends\_on içinde, app servisinin hangi servisler çalıştıktan sonra çalışacağını belirtiyoruz, buna göre db servisi başladıktan sonra app servisi başlatılacak
    - ports içinde, host:container şeklinde, app servisinin host'ta çalışacağı port'u belirtiyoruz,
       buna göre app servisi host'un 80 port'unda çalışacak
    - networks içinde, app servisinin hangi network'e bağlanacağı belirtiliyor, buna göre app servisi net network'üne bağlanacak

```
db:
    image: mysql:latest
    environment:
        - MYSQL_DATABASE=yavuzlar
        - MYSQL_ROOT_PASSWORD=1
    volumes:
        - db_data:/var/lib/mysql
        - ./yavuzlar_messages.sql:/docker-entrypoint-
initdb.d/yavuzlar_messages.sql
    ports:
        - "8080:3306"
    networks:
        - net
```

- **db** -> servisin adı
  - image ile db servisinde kullanılacak image'ı belirtiyoruz, buna göre db servisinde kullanılacak image mysql
  - **environment** ile environment variable'larını belirtiyoruz, buna göre db servisi başlatıldığında yavuzlar adında bir db oluşturulacak ve root password'u 1 olacak
  - **volumes** ile volume belirtilir; buna göre yavuzlar\_messages.sql file, host /var/lib/mysql içinde de tutulacak, yapılan her değişiklikten etkilenecek şekilde

• **ports** ile host:container şeklinde, db servisinin host'ta çalışcağı port belirleniyor, buna göre db servisi host üzerinde 8080 portunda çalışacak

• **networks**, db servisinin hangi network'e bağlanacağı belirtiliyor, buna göre app servisi net network'üne bağlanacak

networks:
net:

driver: bridge

- networks ile servisler için özel bir network oluşturuluyor
  - **net** -> network'ün adı
    - **driver** ile kullanılacak network sürücüsünü belirtiyoruz, buna göre network'te bridge driver kullanılacak, bu servislerin birbirleri ile private bir network'te iletişim kurmasına olanak sağlar

volumes:
 db\_data:

• db\_data adında volume oluşturuyoruz, böylece container'lar kapansa, durdurulsa bile data kaybının önüne geçiyoruz