**Docker Commands**

Dockerhub’tan imageleri alabiliriz. Birden fazla ‘aynı’ image’yi anda çalıştırabiliriz.

\*docker pull mongo : image’yi bilgisayara kaydeder ama çalıştırmaz.

\*docker run mongo : image’i çalıştırır ama içine girmez. Kullanıma hazır olmaz. Eğer böyle bir image daha önce indirmemişsek de bunun latest versionunu bizim için dockerhubtan indirir.

\*docker run -it mongo : image’i çalıştırır ve içine de girer artık aktif kullanabiliriz.Fakat detach mode da çalıştırmak daha mantıklı bu mod açıklaması aşağıda.

\*docker run -it --name n\_mongo mongo : image’i çalıştırır ve image’ye n\_mongo adını verir.

\*docker images : pc ye daha önce indirdiğimiz çalışan çalışmayan bütün imageler ve onların özelliklerini gösterir.

\*docker ps : aktif olarak o an run edilmiş imageleri gösterir.

\*docker ps -a yada -all : şimdiye kadar çalıştırıp kapatılmış imagelerin loglarını tutar.

\*docker start ${image ıd yada image name} : image’yi ıd yada name ile ayağa kaldırma.

\*docker stop ${image ıd yada image name} : image’yi ıd yada name ile stop etme.

\*docker rm ${image ıd yada image name} : image’yi ıd yada name ile log defterinden siler.Yani artık ps yazınca silinen image gözükmez

\*bütün logları tek seferde silmek için ya docker rm containerıd1 containerıd2 containerıd3 diye bütün ıdleri yada isimleri yazmalıyız.Yada docker rm $(docker container ls -aq) komutunu çalıştırmalıyız.

Tags

\*Docker’da tags demek aslında indireceğimiz imagelerin sürümü demek mesela docker pull mongo diyince yada docker run mongo diyince latest versiyonunu otomatik indirir ama diyelim ki docker pull mongo:5 yada docker run mongo:5 yazarsak mongonun 5 nolu sürümü varsa bunu indirir,Hangi imagelerin hangi sürümü var bunu kontrol etmek için dockerhuba girip o imagenin sayfasında bulabiliriz.

\*Diyelim ki aynı imageden 2 tane indirdik ama bunların isimleri aynı ve ayırabilmek için containerId’sinin yanı sıra imageyede yeni bir isim verebiliriz.bunun için gerekli komut = docker images tag ubuntu my\_new\_ubuntu : bu komut ile yeni bir ubuntu imagesi indir ve adını my\_new\_ubuntu koy dedik.

Detach Mode

\*Eğer docker run mongo yada redis yada başka bir şey gibi images çalıştırdığımızda cmd kitlenir çünkü bir tek onu çalıştırır ve işlevini durdurur. Buna karşı run etmek istediğimiz imageyi detach mod ile run etmeliyiz. Buda şöyle olacak docker run -d mongo .

Port Mapping

\*Dockerda bir imagesi çalıştırmanın mantığı şudur: Her imagenin bir hostu vardır mesela mongonun 27017’dir. Bu değişmezdir ve her zaman bu host üzerinden bağlanırız. Fakat direk olarak bu hosta bağlanmayız ilk önce dockerın hostuna daha sonra bu hostun içinden de mongoya bağlanırız.Bu işleme portmapping deniyor. İşin anlaşılır ve kısa özeti şu ilk önce mongoya bağlanacağımız bir host belirliyoruz ve bu hostu 27017 ye bağlıyoruz bunu da şu komut ile yapıyoruz. docker run -d -p 27018:27017 mongo > bu kod şunu diyor : docker run et,-d yani detach mode ile daha sonra gelen -p 27018 bizim bağlanmak istediğimiz host numarası ve bu 27018 i 27017 ye bağla sonrada çalıştırmak istediğimiz imagenin adı

Volume

\*Docker’da diyelim ki şimdiye kadar anlatılanlar gibi bi images bi mongo kurduk bu mongonun içine databaseler tablolar ekledik ettik.Sonra containerı bi stop ettik tekrar başlattık ama gördük ki eklediğimiz herşey gitmiş.Dockerı bir kere run edip sonra stop edersek kaydettiğimiz herşey gider ama bu mantıklı mı?değil,bunu şöyle çözüyoruz. docker run -v /date/deneme -p 27018:27017 -d mongo.Buradaki -v komutuyla imageyi volüme ediyoruz. Yani dockerın bir köşesine yazıyoruz.Artık stop edip çalıştırsak bile veriler korunur.

İnspect

\*docker inspect ${imageId} = ıdsi girilen image hakkında tüm bilgileri getirir.

Environment Variable

\*docker run --name mysqlserver -e MYSQL\_ROOT\_PASSWORD=1234 -p 3307:3306 -d mysql :buradaki -e ile mysql root password’e parametre veriyoruz demek.Kalanı biliyosun zaten

\*docker run --name phpmyadmin --link mysqlserver:db -p 8000:80 -d phpmyadmin : buradaki link ise mysql’i phpmyadmine bağla demek mysqlserver mysqlin namesidir, :db ise alias anlamını taşır.burada localhost yerine kullanılmıştır. Yani mysqlserverin localhostunu burayla bağla demektir.

\*Docker’da mantık şudur, her bi kullanacağımız program imagedir, İmagelerin birleşimine container denir ve bu imagelerin kendi aralarında haberleşmesi network üzerinden olur. Üstteki –link ile bunu yapabiliriz. Başka ve daha sağlıklı yol da var oda network üzerinden imageleri birleştirmek ve containerlar arası haberleşmek.Onuda şu şekilde yapıcaz ilk önce bir network kurucaz. Aktif olarak default birkaç tane newtork vardır bunları görmek için docker network list yada ls yazabiliriz.

Network

Yeni bir network inşa etmek için docker network create --driver bridge --subnet 182.18.0.1/24 --gateway 182.18.0.1 denemeNetwork komutunu çalıştırıyoruz daha sonra diyelim ki mongoyu kuracaz ama bu network için de,o zaman yazacağımız kod şöyle olmalı : docker run –net denemeNetwork -d mongo

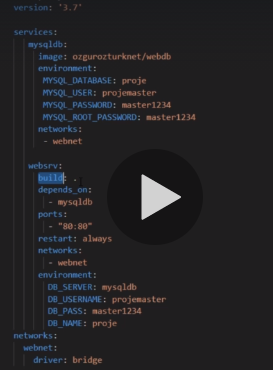
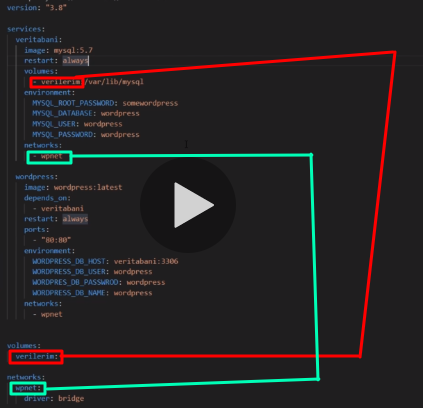
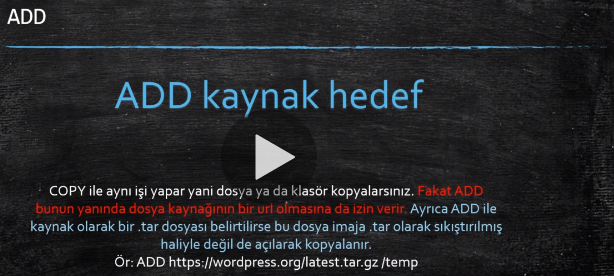
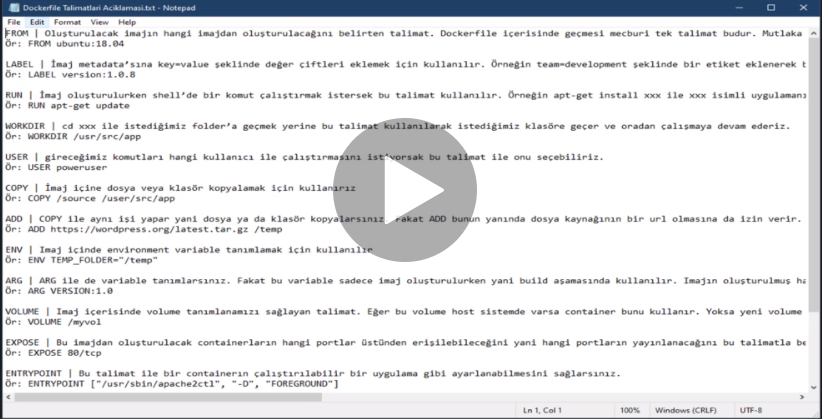
Make Own Image

Docker Compose

\*docker-compose config : o an ki imagemizi konsolda gösterir.

\*docker-compose images : imageleri listeler

\*docker-compose logs : bütün servislerin loğlarını gösterir



**DOCKER COMPOSE.YAML CONFİGURATİONS**

MONGO

**version:** '3.1'

**services:**

**mongo:**

**image:** mongo

**ports:**

- "27017:27017"

**command:** --serviceExecutor adaptive

ELASTİCSEARCH

**version:** '3.1'

**services:**

**elasticsearch:**

**image:** elasticsearch:7.13.2

**ports:**

- "9200:9200"

- "9300:9300"

**environment:**

- discovery.type=single-node

RABBİTMQ

version: "3.9"  
services:  
 rabbitmq:  
 image: rabbitmq:3-management  
 ports:  
 - '5672:5672'  
 - '15672:15672'  
 environment:  
 - RABBITMQ\_DEFAULT\_USER=rabbitmq  
 - RABBITMQ\_DEFAULT\_PASS=1234