Genel Olarak Docker Nedir?:

Docker en net tanımlamayla open source bir ‘container’ teknolojisidir. Docker, aynı işletim sistemi üzerinde, yüzlerce hatta binlerce birbirinden izole ve bağımsız containerlar sayesinde sanallaştırma sağlayan bir teknolojidir. Bunun yanında sunucu maliyetlerini önemli ölçüde azaltır.

Docker’a Neden İhtiyaç Duyarız?:

Docker kullanmadan işlerimizi halledemez miyiz? tabikide yapabiliriz bu gibi teknolojiler zorunluluktan dolayı kullanılmıyor asıl kullanım amacı hayatımızı kolaylaştırmak, yaptığımız işlemleri daha performanslı ve daha az maliyetli hale getirmek zaten bir teknoloji kullanırken ondan beklentilerimiz bunlar değil midir?Peki bir örnek vermek gerekirse bizim işlerimizi nasıl kolaylaştırıyor?Diyelim ki bir web projemiz var ve projenin sonuna geldik artık geriye kalan iş bunu yayına almak. hetzner, amazon, azure gibi bir yerden yayına almak için sunucu aldık ve sunucuya girdik.Yapmamız gereken ilk işlemler nelerdir? tabikide sunucuyu güncellemek, gerekli paketleri kurmak, projenin teknolojisini kurmak,projenin bağımlı olduğu paketleride kurmak,proje database bağlanacaksa ihtiyaç duyduğu databaseleride kurduk, projemizde redisde var o yüzen onuda kurduk ve böyle gider bir çok paket kurduk ve günün sonunda projemizi ayağa kaldırdık buraya kadar çok iyi.Diyelim ki projemizi taşımamız lazım başka bir sunucuya geçicez burda gerekli ortamı kurmak için harcadığımız saatler bu sefer yeni sunucuda yine harcayacağız halbuki yapacağımız işler tamamen aynı. yine paketler kuracaz yine sunucuyu güncelliyecez, yine database kuracaz bu işlemler belli bunları tekrar edicez he işte docker bu konuda bizi kurtarıyor.Docker sayesinde bu yaptığımız işlemleri bir dosyaya adım adım yazıyoruz(Dockerfile). Proje ayağa kalkarkende bu dosyaya bakarak kendi gerekli ortamını oluşturuyor böylece projemizi başka bir ortama taşırken bize saatler kazandırıyor.Tek görevi bu mu? tabikide hayır docker’ın bir çok özelliği var örnek olarak bu teknoloji sayesinde localimizde çalışan projeler sunucuda da aynı localimizde çalıştığı gibi çalışmasını sağlayabiliyoruz böylece “benim bilgisayarda çalışıyordu” cümlesini artık duymuyoruz yada docker container içinde çalıştığı aynı sunucuda birbirinden bağımsız ve farklı ortamlara ihtiyaç duyan projeler ayağa kaldırabiliyoruz. Kaynak tüketimini kontrol edebiliyoruz, kurulumu zor olan teknolojileri tam olarak ihtiyacımızı çözecek şekilde sunucuda kullanabiliyoruz.

Docker Nasıl Çalışıyor?

Docker ile bir proje ayağa kaldırdığımızda aslında olanlar o projenin çalışabileceği bir linux işletim sistemi üzerinde ayağa kalkmasıdır.Biraz daha açıklamak gerekirse bir web projesini ayağa kaldırmak istediğimizde öncelikle bir image oluşturmamız gerekiyor bu image oluşturma işleminde projemizi bir linux işletim sisteminin içine kopyalıyor ve sonra adım adım projenin ayağa kalkması için yapmamız gereken işlemleri teker teker yapıyor ve bunları yaptıktan sonra projenin o durumunu alarak bir image oluşturuyor. bu image'i bir kurulum dosyası gibi düşünebiliriz. Bu image sayesinde projemizi istediğimiz kadar bir çok kez ayağa kaldırabiliyoruz bu image'i kullanarak kaldırdığımız projelerde container denilen yapıların içinde oluyor.

"Docker Image 'ı aslında bir template, şablon olarak tanımlayabiliriz. Docker üzerinde çalışabilecek bir container oluşturmak için gerekli olan template'lere biz aslında Docker Image diyoruz. Kendi özel kullanımınız için kullanabileceğiniz veya diğer Docker kullanıcılarıyla herkese açık olarak paylaşabileceğiniz uygulamaları ve önceden yapılandırılmış sunucu ortamlarını paketlemek için kullanışlı bir yol sağlıyor. "[1]

Örnek Bir Docker Projesi:

Kendi projemize geçmeden önce docker'a giriş seviyesinde olacak bir örnek yapalım.Yapacağımız proje bilgisayarımıza docker aracılığıyla linuxun ubuntu distrosunu kurmak.Burda bir image oluşturmayacağız bunun sebebi ubuntunun zaten docker hubda image'i bulunması.Yapacağımız işler önce docker huba giricez ubuntunun versiyonlardan birini seçip image'ini bilgisayarımıza indiricez bu işlemden sonra bu image'i kullanarak localimizde birden fazla containier oluşturcaz ve bu containerları listeleme,silme, içinde komut çalıştırma işlemlerini yapıcaz.
Adım adım ilerleyelim.

Adım 1 Docker Hub:

Öncelikle https://hub.docker.com/ adresine girip arama yerine ubuntu yazıp imagelerin bulunduğu yere girelim. Bir çok teknolojinin image'ini docker hubda bulabiliriz.

Supported tags and respective `Dockerfile` links

- `18.04`, `bionic-20220315`, `bionic`
- `20.04`, `focal-20220316`, `focal`, `latest`
- `21.10`, `impish-20220316`, `impish`, `rolling`
- `22.04`, `jammy-20220315`, `jammy`, `devel`
- `14.04`, `trusty-20191217`, `trusty`
- `16.04`, `xenial-20210804`, `xenial`

Burda ubuntunun image'ini indiricez ama hangi versiyonunu? bunu taglar ile belirtiyoruz örnek açısından burda ubuntunun 20.04 sürümünü indirmek isteyelim

Adım 2 Image İndirme:

Şimdi sırada bu image'i indirmek var bunun için kullanılan komut formatı;

`docker pull <image adı>:<tagı>`

Bizim yazacağımız komut -> `docker pull ubuntu:20.04`

```
halilkaya@macer ~ % docker pull ubuntu:20.04
20.04: Pulling from library/ubuntu
57d0418fe9dc: Pull complete
Digest: sha256:bea6d19168bbfd6af8d77c2cc3c572114eb5d113e6f422573c93cb605a0e2ffb
Status: Downloaded newer image for ubuntu:20.04
docker.io/library/ubuntu:20.04
halilkaya@macer ~ %
```

Bu komutu yazdıktan sonra localimize 20.04 versiyonlu ubuntunun docker image'i bilgisiayarımıza kurulmuş olucaktır.

Adım 3 imageleri listeleme:

Bu indiridiğimiz docker imagelerini listelemek için `docker images` komutunu girmemiz yeterli böylece localimizdeki docker imagelerini görebiliriz.

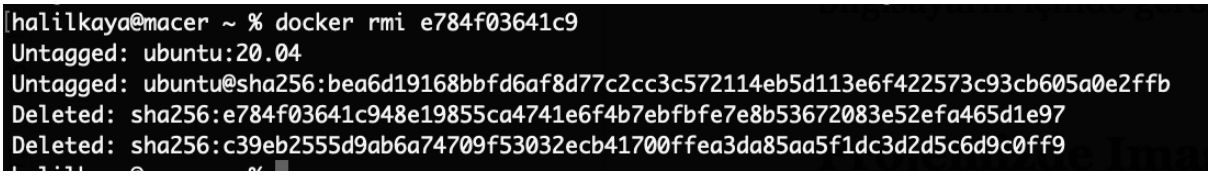
```
halilkaya@macer ~ % docker images
REPOSITORY                TAG                 IMAGE ID       CREATED         SIZE
ubuntu                    20.04               e784f03641c9   4 days ago      65.6MB
```

REPOSITORY -> image'in çekildiği reponun adı
TAG -> image'in versiyonu
IMAGE ID -> image'in id si
CREATED -> image'in oluşturulma tarihi
SIZE -> image'in boyutu

Adım 4 Image silme:

Şimdide örnek olarak bu image'i silmek istersek ne yapmalıyız? bunun içine gereken komut `docker rmi <image_id>` burdaki image_id adı üstünde silinecek olan image'in

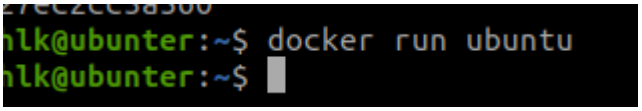
idsi. Silmek istediğimiz ubuntu image'in idsi e784f03641c9 bunu silmek için yazacağımız komut -> `docker rmi e784f03641c9`



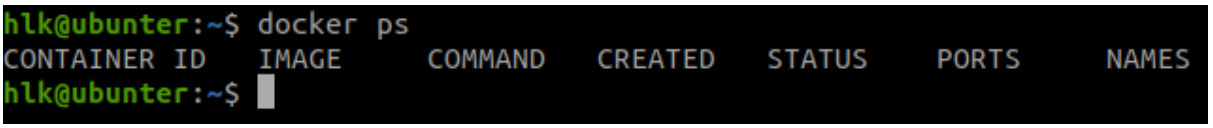
bu komut sayesinde az önce indirdiğimiz image'i silmiş olduk.

Adım 5 image'den container oluşturma:

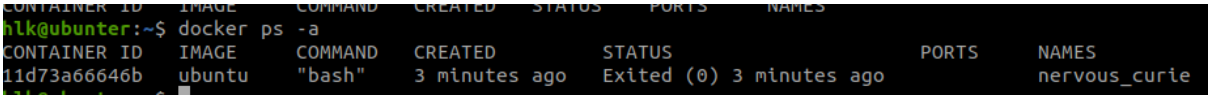
Başlamadan önce ubuntu image'imizi tekrardan indirmemiz gerekiyor yine `docker pull ubuntu:20.04`
komutunu çalıştırıp image'imizi indiriyoruz.
Bu image'den bir container oluşturucaz bunun için kullanılan kod `docker run <image_adı veya image_id>`
bizim yazacağımız kod -> `docker run ubuntu`
komutumuzu yazdıktan sonra hiçbir çıktı gelmedi



peki şu an ne oldu? çalıştı mı? çalıştıysa nerde?
docker'da çalışan container ları görmek için `docker ps` komutunu kullanıyoruz bu komutu çalıştıralım.

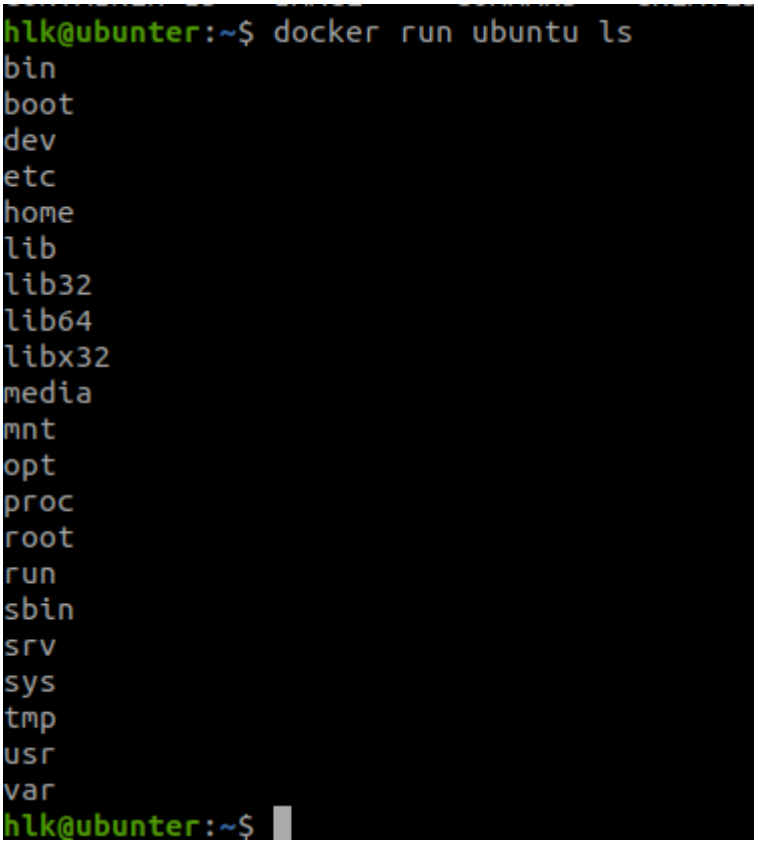


çıktı olarak yine hiçbir şey yok. Docker'da çalışıp işi biten container ları görmek için de `docker ps -a` komutunu kullanıyoruz bide bunun çıktısına bakalım.



Bu sefer bize bir çıktı verdi peki bu tam olarak ne? Neden ubuntu container ı çalışıp kendiliğinden kapandı? ve ismi neden *nervous_curie*?
Çalışıp kapanmasının sebebi ubuntunun ayağa kalkıp bir komut yada program çalıştırmadığı için işi bitip kendini kapatması.Şunu diyebiliriz container lar yapacak bir işleri olmadıklarında kaynak tüketmek yerine kendilerini kapatırlar.

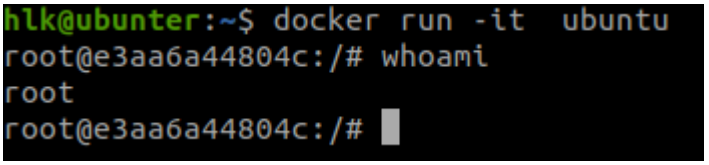
İsminin nervous_curie olmasının sebebi ise her bir container a bir isim verilmesidir yönetmesi daha kolay olsun diye bu ismi biz kendimiz verebiliyoruz eğer biz vermez isek bu sefer sistem kendisi otomatik eşsiz bir isim atar.
hadi bu sefer yeniden bir container oluşturalım ama bu sefer kendisine bir iş verelim
bunun için yazacağımız komut docker run ubuntu ls
burda container ımız çalışacak ls komutunu terminalde çalıştırıp bize çıktı vericek



ve bu komutu çalıştırdıktan sonrada artık yapacak bir işi olmadığı için kendini yine kapatacak.

Adım 6 docker'ın içine girmek:

Şimdi de bir container'ın içine girmek isteyelim bunun için kullanmamız gereken -it (i interactive, t terminal) komutu
yazacağımız komut docker run -it ubuntu



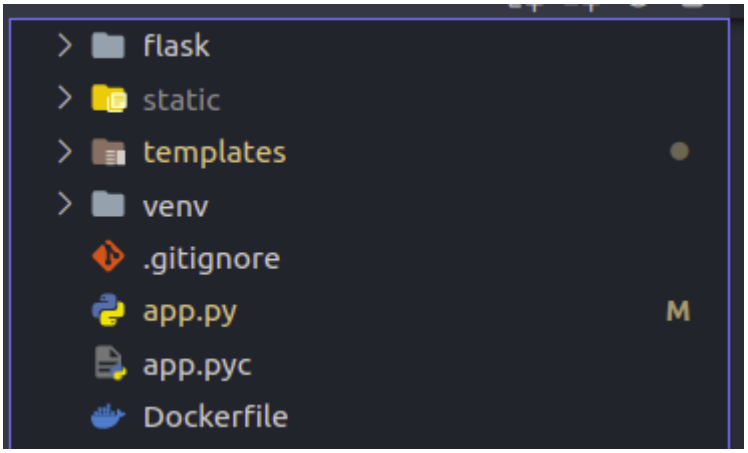
bu komut sayesinde bir container oluşturuyoruz ve bu container'ın içine giriyoruz.

Maske Tespit Etme Projesinde Docker'ın Yeri?

Bizim bu projede docker kullanmamızın sebebi projemizi bulunduğu ortamdan bağımsız bir şekilde çalıştırmak istememiz. Böylece projemiz docker sayesinde hem saniyeler içerisinde ayağa kalkabiliyor hemde çalıştığı bilgisayarın içinde gerekli ortamın bulunmasına gerek kalmıyor.

Projemizde Image'imizi Oluşturalım

Bu sefer projemizde kendi image'imizi oluşturacaz bunu yapmak için ilk yapacağımız işlem projenin bulunduğu dizinde bir **Dockerfile** oluşturmak



Bu Dockerfile'ın içinde projemizin bir bilgisayarda çalışması için yaptığımız işlemleri sırası ile belli bir formatta yazıyoruz.

```
Dockerfile > ...
1   FROM ubuntu
2   WORKDIR /mask-detected
3   ADD . /mask-detected
4   RUN apt update
5   RUN apt upgrade -y
6   RUN apt-get install ffmpeg libsm6 libxext6  -y
7   RUN apt install python3 -y
8   RUN apt install python3-pip -y
9   RUN pip install --upgrade pip
10  RUN pip install opencv-python
11  RUN pip install -r requirements.txt
12  CMD ["python3","app.py"]
```

Böylece image'imizi oluştururken bu dosyaya bakarak bize göre bir image oluşturuyor hadi bu komutları sırası ile ele alalım.

FROM ubuntu

FROM komutu kullanmamızın sebebi image'imiz neyin üstünde çalışacak onun bilgisini veriyoruz. Image'imiz ubuntuyu kendisine temel alıcak

WORKDIR /mask-detected

WORKDIR komutu, herhangi bir zamanda bir Docker konteynerinin çalışma dizinini tanımlamak için kullanılır. Komut, Dockerfile'da belirtilir.
Herhangi bir RUN, CMD, ADD, COPY veya ENTRYPOINT komutu, belirtilen çalışma dizininde yürütülecektir.[3]

bizim burdaki çalışma dizinimize verdiğimiz isim 'mask-detected'

ADD . /mask-detected

ADD komutu dosyaları bi dizine taşımaya yarar. Burda yaptığımız işlem şu anki bütün dosyaları oluşturduğumuz 'mask-detected' çalışma dizinine taşıması.

RUN apt update
RUN apt upgrade -y
RUN apt-get install ffmpeg libsm6 libxext6 -y
RUN apt install python3 -y
RUN apt install python3-pip -y
RUN pip install --upgrade pip
RUN pip install opencv-python
RUN `pip install -r requirements.txt`

RUN komutu terminalden komut çalıştırma mıza olanak sağlıyor.Bu komutu kullanarak projemizin çalışması için gereken paketleri kurmasını sağlıyoruz

CMD ["python3","app.py"]

CMD komutu çalışma dizinimizde çalıştırmak istediğimiz komutları girmemizi sağlıyor

bunları yazdıktan sonra kendi image'imizi oluşturalım

image oluşturma komutu -> **docker build <dizin> -t <oluşacak olan image'in ismi>**

bizim komutumuz -> **docker build . -t mask_detection_image**

```
hlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker build . -t mask_detection_image
Sending build context to Docker daemon  2.062GB
```

bu komutu yazdıktan sonra image'imiz oluşmaya başlar bu işlem 5 dakika kadar sürebilir.

oluşan image'imize bakalım **docker images**

```
^Chlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
mask_detection_image    latest    5c1178c1083c   5 minutes ago    6.37GB
```

image'imiz başarlı bir şekilde oluştu şimdi bu image ile bir container oluşturalım

```
hlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker run --name mask_detection_container mask_detection_image
2022-03-23 17:34:09.160694: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynam
 file or directory; LD_LIBRARY_PATH: /usr/local/lib/python3.8/dist-packages/cv2/../../lib64:
2022-03-23 17:34:09.160711: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if
2022-03-23 17:34:11.235059: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynam
irectory; LD_LIBRARY_PATH: /usr/local/lib/python3.8/dist-packages/cv2/../../lib64:
2022-03-23 17:34:11.235080: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN
2022-03-23 17:34:11.235093: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not ap
```

bu image sayesinde saniyeler içerisinde projemizi ayağa kaldırmayı başardık peki projemize nasıl ulaşıcaz? bizim projemiz 3000 portundan yayın yapmaktadı burda docker ile ayağa kaldırdık ama şu an projemize bağlanabileceğimiz dışardan bir portu yok. Dışarıdan ulaşmak için verdiğimiz komut **-p <dış port:iç port>** komutu.
eğer -p 5000:3000 komutunu verseydik docker içerdeki 3000 portu dinleyip bunu 5000 portuna yönlendircekti böylece bilgisayarımızdaki 5000 portundan projemize ulaşabilecektik. Hadi oluşan konteynarımızı silip bu sefer port ile projemizi ayağa kaldıralım.

```
hlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND           CREATED
1c79e6750bc0   mask_detection_image   "python3 app.py"   10 minutes ago
hlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker rm 1c79e6750bc0
```

docker rm <container_id veya adı>
komutu ile önceki container ımızı siliyoruz ve
docker run -p 5000:3000 --name mask_detection_container mask_detection_image
komutu ile yeni container ımızı bu sefer port vererek ayağa kaldırıyoruz

```
hlk@ubunter:~/Desktop/bitirme/basic-flusk-app$ docker run -p 5000:3000 --name mask_detection_container mask_detection_image
2022-03-23 17:40:20.098353: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcu
 file or directory; LD_LIBRARY_PATH: /usr/local/lib/python3.8/dist-packages/cv2/../../lib64:
2022-03-23 17:40:20.098372: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a
2022-03-23 17:40:22.250270: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcu
irectory; LD_LIBRARY_PATH: /usr/local/lib/python3.8/dist-packages/cv2/../../lib64:
2022-03-23 17:40:22.250292: W tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)
2022-03-23 17:40:22.250305: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be runnin
2022-03-23 17:40:22.250565: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI
```

ve artık böylece http://localhost:5000/ adresinden projemize ulaşabiliyoruz

Kaynaklar

[1]https://www.tayfundeger.com/docker-image-nedir.html#:~:text=Docker%20Image'in%20readonly%20bir,i%C3%A7in%20bunlar%C4%B1%20ba%C5%9Flatamaz%20veya%20%C3%A7al%C4%B1%C5%9Ft%C4%B1ramazs%C4%B1n%C4%B1z.
[2]https://www.youtube.com/watch?v=4XVfmGE1F_w
[3]https://www.educative.io/edpresso/what-is-the-workdir-command-in-docker