docker image ls

docker run -it ubuntu

cat /etc/os-release

exit

docker image pull ubuntu:18.04

docker image ls

mkdir techpro\_web

cd techpro\_web

techpro\_web adlı bir klasör aç ve bu klasör içine welcome.py adlı dosya oluştur ve aşağıdaki kodları yapıştır.

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def hello():

    return "<h1>Welcome to techpro Devops team</h1>"

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=80)

techpro\_web altında Dockerfile dosyası oluştur(Dosyanın yeri önemli)Aşağıdaki komutları içine yaz.

FROM ubuntu

RUN apt-get update -y

RUN apt-get install python3 -y

RUN apt-get install python3-pip -y

RUN pip3 install flask

COPY . /app

WORKDIR /app

CMD python3 ./welcome.py

Aşağıdaki kodu uygulamak için Dockerhub websitesine gidip orada hesap açıp bir repo oluşturmanız gerekiyor. dockerhubusername/flask-app:1.0 bu kısım ilk yer dockerhubdaki kullanıcı ismi ikinci yer repo ismi ve :1.0 kısmı ise tag yani version tagı

docker build -t dockerhubusername/flask-app:1.0 .

docker image ls

docker run -d --name welcome -p 80:80 <dockerhubusername>/flask-app:1.0

docker login

(Bunu yazdığımızda çalıştığımız alanda kullanıcı adı ve şifre isteyecektir.Biz image dockerhub a göndermek için login oluruz.)

docker push <dockerhubusername>/flask-app:1.0

Aynı dosya dizininde bir dockerfile dosyası oluşturduğumuzda ve içine aşağıdaki kodları eklediğimizde bu sefer image kısmında python kısmı otomatik yüklü olan python:alpine image kullanırız.

Dockerfile-alpine

FROM python:alpine

RUN pip install flask

COPY . /app

WORKDIR /app

EXPOSE 80

CMD python ./welcome.py

docker build -t "<dockerhubusername>/flask-app:2.0" -f ./Dockerfile-alpine .

docker run -d --name welcome2 -p 8080:80 <dockerhubusername>/flask-app:2.0

docker stop welcome && docker rm welcome

docker push <dockerhubusername>/flask-app:2.0

docker image tag <dockerhubusername>/flask-app:2.0 techpro/flask-app:latest

docker image rm image\_id

Home dizininde başka bir klasör açıp dockerfile oluşturun ve aşağıdaki kodları kopyalıp yapıştırılam.

FROM ubuntu

RUN apt-get update

RUN apt-get install figlet

ENTRYPOINT figlet Merhaba B149 aws-devops

# CMD can also be used here

docker build -t fig-app .

docker images

docker run fig-app

Aynı dosya dizininde farklı bir docker file oluşturduğumuzda

FROM ubuntu

RUN apt-get update

RUN apt-get install figlet

ENTRYPOINT ["figlet", "Merhaba"]

docker build -t fig-app2 .

docker run fig-app2 Dockerseverler!

Ekranda hem merhaba hem deDockerseverler! Yazar.Bu ENTRYPOINT özelliği.

Elbette, `ENTRYPOINT` ve `CMD` komutlarının farkını daha iyi anlamak için örneklerle açıklayalım:

Diyelim ki basit bir Flask uygulamasını çalıştırmak için bir Docker imajı oluşturuyorsunuz. Flask uygulamanız `app.py` adlı bir Python betiğini çalıştırmak için `python app.py` komutunu kullanıyor.

CMD VE ENTRYPOINT FARKI

Örnek 1: `CMD` Kullanımı

```Dockerfile

FROM python:alpine

# Flask uygulamanızı kopyala

COPY . /app

# Çalışma dizinini ayarla

WORKDIR /app

# Flask uygulamasını çalıştır

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

```

Bu Dockerfile'da `CMD` komutu, Docker konteyneri başlattığınızda otomatik olarak `python app.py` komutunu çalıştırır. Ancak, Docker konteynerini başlatırken farklı bir komut veya argüman belirtebilirsiniz ve bu `CMD` komutunu geçersiz kılabilirsiniz:

Örnek 2: `ENTRYPOINT` Kullanımı

```Dockerfile

FROM python:alpine

# Flask uygulamanınızı kopyala

COPY . /app

# Çalışma dizinini ayarla

WORKDIR /app

# Flask uygulamasını çalıştırmak için temel komutu belirt

ENTRYPOINT ["python", "app.py"]

```

Bu Dockerfile'da `ENTRYPOINT` komutu, Docker konteynerini başlattığınızda otomatik olarak `python app.py` komutunu çalıştırır. Ancak, kullanıcı Docker konteynerini başlatırken bu komutu geçersiz kılamaz. Kullanıcı yalnızca ek argümanlar ekleyebilir:

Örnek 3: `ENTRYPOINT` ve `CMD` Kullanımı

```Dockerfile

FROM python:alpine

# Flask uygulamanınızı kopyala

COPY . /app

# Çalışma dizinini ayarla

WORKDIR /app

# Flask uygulamasını çalıştırmak için temel komutu belirt

ENTRYPOINT ["python", "app.py"]

# Varsayılan argümanları belirt

CMD ["--host", "0.0.0.0"]

```

Bu Dockerfile'da `ENTRYPOINT` ve `CMD` bir arada kullanılır. `ENTRYPOINT`, temel komutu belirtirken, `CMD` varsayılan argümanları sağlar. Kullanıcı, Docker konteynerini başlatırken `CMD` ile belirtilen argümanları değiştirebilir veya ekleyebilir.

Özetle:

- `CMD`, Docker konteynerini başlatırken çalışacak varsayılan komutu ve argümanları belirtir, ancak kullanıcı bunu geçersiz kılabilir.

- `ENTRYPOINT`, Docker konteynerini başlatırken çalışacak temel komutu ve argümanları belirtir ve kullanıcı bunu değiştiremez, yalnızca ek argümanlar ekleyebilir.