Dockerfile

Dockerfile Nedir?

Docker file, image oluşturmak için gerekli işletim sistemi, database, kurulucak araçlar, uygulamanın çalışması için gerekli dosyalar ve kütüphaneleri içeren dosyadır. Örnek Script:

```
FROM nginx:latest
COPY index.html /usr/share/nginx/html
COPY index.html /usr/share/nginx/html
CMD [“nginx”, “-g” “daemon off;”]
```

Web sunucusu nginx: lates

kopyalanacak dosyalar: index.html ve Linux.png

port: 80 olarak belirlenmiştir.

Bu Docker dosyası Docker üzerinden build edildiğinde, adımları tek tek gerçekleştirerek image dosyasını oluşturacaktır.

Docker File ve Image Nasıl Oluşturulur?

Docker file oluşturmak için kullanabileceğimiz belli taglar bulunmaktadır.
Bunlar:
FROM: Kullanılacak sistem belirtilir.
RUN: Belirtilen komutları image içerisinde çalıştırır.
LABEL: Image dosyasına aranabilir meta tagları eklememizi sağlar.
COPY: Belirtilen dosyaları image içerisinde belirlenen hedefe kopyalar.
EXPOSE: Container’ın docker içinde çalışacağı port adresi belirlenir.
WORKDIR: Sonrasında gelen komutları verilen klasör içerisindeymiş gibi çalıştırır.
CMD: İlk önce çağrılır. Çalışma sırasını düzenler. İçine yazılan komut ve parametreleri çalıştırır.
ENV: Image içerisine dışarıdan veri girileceği zaman kullanılır.

Kendi Image Dosyamızı Nasıl Hazırlarız?

1. Docker file oluşturulalım.

```
FROM nginx
ENV AUTHOR=DockerEnv
WORKDIR /usr/share/nginx/html
COPY Color_docker.html /usr/share/nginx/html
CMD cd /usr/share/nginx/html && sed -e s/DockerEnv/"$AUTHOR"/ Color_docker.html >
index.html ; nginx -g 'daemon off;'
```

2. Background değişikliği yapabileceğimiz bir html dosyası hazırlayalım.

```
<html>
<head><style>
body {background-color: DockerEnv;}
</style></head>
<body></body>
</html>
```

3. Klasör içinde terminal açıp, build komutunu girerek image dosyasını oluşturalım.

```
$ Docker build -t myimage .
```

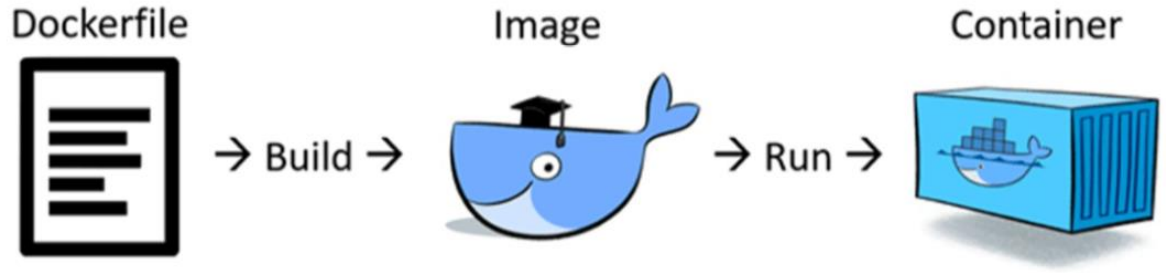
```
ecelik@DESKTOP-HKO3VLQ MINGW64 ~/Desktop/Image
$ docker build -t myimage .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM nginx
 ---> 4bb46517cac3
Step 2/5 : ENV AUTHOR=DockerEnv
 ---> Running in 333f8d3bae26
Removing intermediate container 333f8d3bae26
 ---> 87798d0e7d53
Step 3/5 : WORKDIR /usr/share/nginx/html
 ---> Running in ccae0de46758
Removing intermediate container ccae0de46758
 ---> d7f437496862
Step 4/5 : COPY Color_docker.html /usr/share/nginx/html
 ---> d20223a39f34
Step 5/5 : CMD cd /usr/share/nginx/html && sed -e s/DockerEnv/"$AUTHOR"/ Color_d
ocker.html > index.html ; nginx -g 'daemon off;'
 ---> Running in bcc57f42b510
Removing intermediate container bcc57f42b510
 ---> f970ab62e1f5
Successfully built f970ab62e1f5
Successfully tagged myimage:latest
```

Docker File içerisindeki tüm adımlar tek tek gerçekleşti ve “myimage” adında bir Docker image oluşturulmuş oldu.

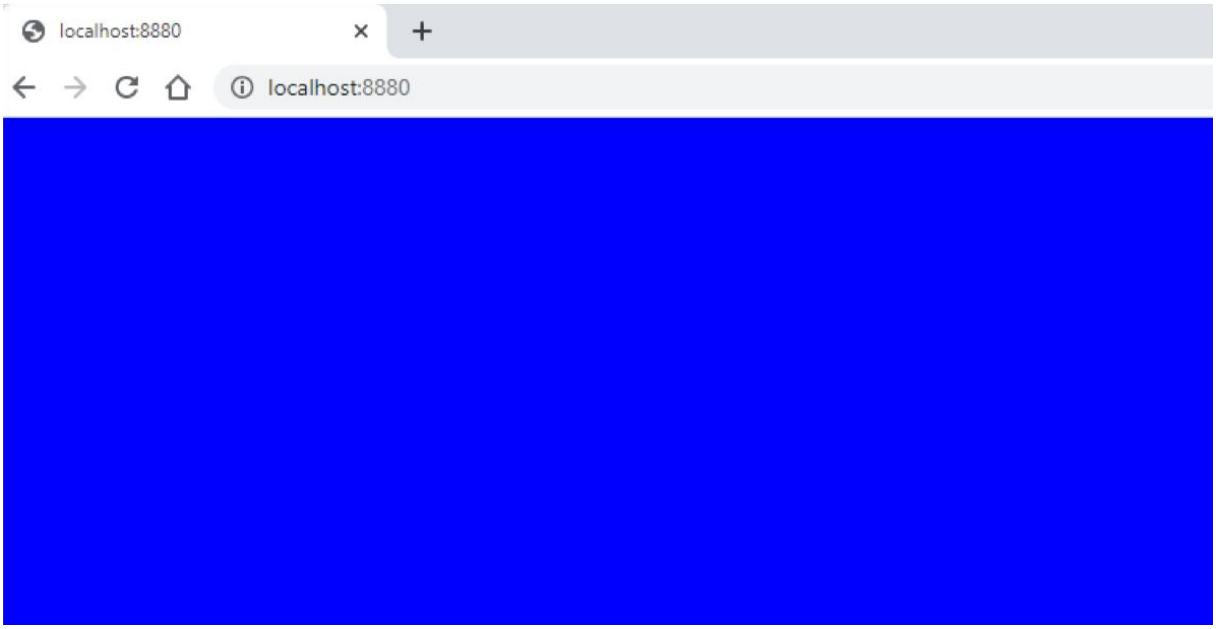
4. Docker image, run komutu kullanılarak container haline getirilir.
   `$ Docker run -d -p 8880:80 -e AUTHOR=”blue” myimage`

Kullanılan Komutlar Nelerdir?

run: image’ı çalıştırmak için kullandığımız komut
-d: –detach in kısa halidir. Container arka planda başlatılır.
-p: docker container’ a bir port verir. Buradaki 80 portu docker içinde geçerlidir, tarayıcı üzerinde görüntülenemez. Bu yüzden 8880 portunu 80 portuna expose ettik, artık localhost:8880 portuna giderek tarayıcı üzerinde görüntüleyebiliriz.
-e: Docker file içerisindeki ENV AUTHOR ile dışarıdan almak istenilen veri gönderilir.
Son olarak çalıştıracak olan image adı girilir.



Localhost:8880 portunu tarayıcıdan açtığımızda arka plan renginin mavi olduğunu görüntüledik.



Docker Container İçerisindeki Verilerin Saklanması

Container içerisine sonradan yüklenen her şey container silindiği anda tamamen yok olacaktır. Bunu önlemenin birkaç yolu vardır:

1. Bind Mount

Kendi bilgisayarımızdan belirleyeceğimiz klasörü kaynak olarak gösterebiliriz. Örnek syntax;

```
$ Docker run -d –mount type=bind, source=/tmp/docker/mydocuments/temp , target=/myte
```

Ancak Bind Mount genelde yazılım aşamasında kullanılır. Bunun sebebi dosyaların kolayca değişebilmesi ve hassas dosyalara erişimin açık olmasıdır.

2. Volume

Bir kaynak klasörü göstermek yerine sadece isim verilir. Docker /var/lib/docker/volumes/ içerisine oluşturur ve yönetir. Oluşturmak için:

```
$ Docker volume create mydocuments
```

Volumeleri listelemek için:

```
$ Docker volume ls
```

İstenen volüme'ü görüntülemek için:

```
$ Docker volume inspect mydocuments
```

Çalıştırma sırasında:

```
$ Docker run -d –mount source=mydocuments, target =/mytemp myimage
```

Canlıya çıkmış ürünlerde kullanılması önerilir. Kolayca back-up alınabilir.
3. Tmpsf Bellek hafızasını kullanarak verileri saklar, yer belirtmeye gerek yoktur.

```
$ Docker run -d –mount type=tmpfs , target =/mytemp myimage
```

Docker Compose Nedir?

Docker compose ile basit yapıdaki uygulamalarımızı herhangi bir işletim sistemi veya web sunucusu image'ında çalıştırabiliriz; ancak daha kompleks uygulamalarda birbirinden farklı image'lara ihtiyacımız olabilir. Compose ile birlikte birden fazla Container tanımı tek bir dosyada yapılabilir. Bunun için basit bir yml dosyası oluşturmamız yeterlidir. Bu dosya ile birlikte tek bir komut ile uygulamanızın ihtiyaç duyduğu tüm gereksinimleri ayağa kaldırarak uygulamayı çalıştırabiliriz.

Kaynaklar;

https://keytorc.com/blog/docker-file-ve-image-nasil-olusturulur/