Docker Architecture 
Client 
docker build 
docker pull 
docker run 
DOCKER_HOST 
Registry 

Client docker kullanıcılarıdır. Docker host docker ın yüklü olduğu makinedir. Registry de image ların saklandığı yerdir.

Konteynır ın amacı tek kullanımlık makinedir. Ec2 kadar önem vermiyoruz. Kağıt bardak gibi. Client

Registry den image çekip kullanabilir ya da kendimiz image oluşturup onunla konteynır ayağa kaldırabiliriz.

Images and Containers 
An image is a read-only template with instructions for creating a Docker container. 
A container is a runnable instance of an image. 
Irnage 
run 
Docker Image 
Docker Container 
aws 
Amazon Machine 
Image (AMI) 
Amazon EC2 

Image read-only bir template tir, Docker konteynır oluşturmak için talimatları barındırır.

Konteynır ise bir image in çalıştırılabilir instance ıdır.

Ekrana hello world yazdırmak da bir uygulamadır ve bunu çalıştıracak bir image kullanırız.

AWS de bir makinemiz olunca çökmemesi için çok özen gösteriririz. Snapshot alırız vs. Ancak konteynır öyle değildir. Bir sürü konteynır anında çalışıp anında silinebilir.

Image şu şekilde oluşturuluyor:

A Docker image is built up from a series of layers. Each layer represents an 
instruction in the image's Dockerfile. Each layer except the very last one is read-only. 
docker run imageName 
Thin R/W layer 
container layer 
FROM ubuntu 
copy 
. /app 
RUN apt-get 
CMD python3 
Dockerfile 
install python3.6 
/ app/ index. py 
91e54dfb1179 
d74508fb6632 
c22013c84729 
d3a1f33e8asa 
u buntu 
0B 
1.895 KB 
Image layers (RIO) 
194.5 KB 
188.1 MB 

Docker image katmanlı yapılarla oluşturulur. Bu katmanlar read-only yapıdadır. Her katman dockerfile dosyasının image i ile ilgili bir yapıyı simgeler. En üstteki katmanda ise yazma yapılabilir.

Docker da data yönetimi:

Konteynır içindeki dosya konteynır gittiğinde silinir.

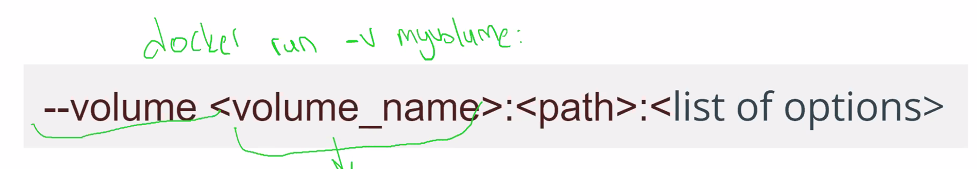
By default, all files created inside a container are stored on a writable container 
layer. This means that the data doesn't persist when that container no longer exists. 
Docker volumes, which are special directories in a container, store files in the host 
machine so that the files are persisted even after the container stops. 
Docker Host Server 
volume 
Docker 
Container 

Volume ler konteynır la kaybolacak datayı elde tutmaya yarar. Çalıştığımız makinede bir storage alanı ya da volume oluşturuyoruz. Konteynır la bu volume ü birbirine bağlıyoruz ve konteynır silinse de data volume de kalır ve başka kaonteynırlara da bağlanabilir.

Docker inspect komutuyla da volume leri inceleyebiliyoruz. Bu komut diğer objelerle de kullanılıyor.

When we create a volume, it is stored within a directory on the Docker host. When we 
mount the volume into a container, this directory is what is mounted into the container. 
$ docker volume inspect firstvolume 
"CreatedAt": "2020-07-12T13: 19:27Z", 
"Driver": "local", 
"Labels": 
"Mountpoint": "Ivar/lib/docker/volumes/firstvolume/_data", 
"Name": "firstvolume", 
"Options": O, 
"Scope": "local" 

Docker volume oluştuktan sonra bunu docker hosttaki bir klasöre bağlıyoruz. Mount ediyoruz. Burada önemli nokta volume oluşturduktan sonra mountpoint te görebiliyoruz.

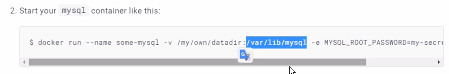


Volume bağlamak için komut bu yapıdadır. --volume ya da –v flagi kullanılır. Sonrasında volume ismi yazılır, : ile ayrılır ve mount edileceği klasör yazılır sonrasında : ile tekrar ayırıp read-only yapabiliyoruz (ro)

path uygulamanın çalışacağı yerdir.

image lerin bir ksımında nereye mount edileceği de hubdocker da yazar:

Where to Store Data 
that in Docker We 
Important rote: are 
images to *ith the 
a o nacemenr clefault ane n 
a sparenttotne 'he 
files may be to for that on system. 
• a data on the system direeory visible from 
This 
on end it 
and 
that 
vstem 
The good 
fw me cglion 
suitable 
like this: 
The i, 
othU on 
oint and vsrietior.s. nd 
9 



bu satırda kodun nasıl yazılacağı belirtilmiş örneğin.

Declaration of volumes 
We can use the same Volume with different Containers. 
clarus 
clarus2nd 
cw-vol 
clarus3rd 
clarus4th 
ARUSWAYO 

volume ün özelliği paylaşılabiliyor olması. bir volume yukarıdaki 4 konteynır a da bağlanabilir. uyglamaya göre konteynıra bağlarken bazılarına ro ile kısıtlama verebiliriz.

Docker Volume Behaviours 
No 
2 
3 
4 
Situation 
If there is no target directory. 
If there is a target directory, but it is 
empty. 
If there is a target directory and it is 
not empty, but volume is empty. 
If the volume is not empty. 
Behaviour 
The target directory is created and files inside volume 
are copied to this directory. 
The files in the volume are copied to the target directory. 
The files in the target directory are copied to volumes. 
There will be just the files inside volume regardless of the 
target direetory is full or empty. 

volume her zaman baskındır.

1. eğer konteynırda hedef klasör yoksa önce o klasörü oluşturur ve volume içindeki datayı oraya aktarır.
2. Eğer konteynırda hedef klasör varsa ve boşsa volume deki data yine oraya aktarılır.
3. hedef klasörde veri var ancak volume boş, bu durumda hedef klasördeki veri volume e kopyalanır.
4. volume de dolu hedef klasör de dolu, volume baskın olacağı için volume deki data oraya aktarılır.

Docker Volume Behaviours 
No 
2 
Situation 
If there is no target directory. 
If there is a target directory, but it is empty. 
full.txt 
Behaviour 
The target directory is created and files inside volume are 
copied to this directory. 
The files in the volume are copied to the target directory. 
full.txt 
full. txt 
Volume 
Container 

Docker Volume Behaviours 
No 
3 
Situation 
If there is a target directory and it is not 
empty, but volume is empty. 
Behaviour 
The files in the target directory are copied to volumes. 
app.py 
app.py 
Volume 
app.py 
Container 

Start 
Look at 
Volume 
is empty 
Look at Target 
Folder 
is empty 
No action 
e just see the file 
inside volume 
regardless of the 
target directory is full 
r empty. 
e files in the targ 
directory are copied 
volumes. 

Command 
docker volume create 
docker volume inspect 
docker volume Is 
docker volume prune 
docker volume rm 
Description 
Create a volume 
Display detailed information on one or more volumes 
List volumes 
Remove all unused local volumes 
Remove one or more volumes 

docker volume komutları

create oluşturur, inspect inceler, ls listeler, prune bütün kullanılmayan volume leri siler, rm bir ya da daha fazla volume siler.

alpine içinde bash shell bile olmayan bir image dir. baz image dir ve boyutu çok küçüktür bu nedenle kullanışlıdır.

docker run -it alpine ash

bu komut alpine konteynır çalıştırır ve içine bağlanır. i interactive t terminal.

T012243-Fatih C. 
7:56 
ash= Almquist shell 
Description: Almquist shell is a lightweight Unix 
shell originally written by Kenneth Almquist in 
the late 1980s. 

gerçek hayatta ise konteynır içine girip bir şey yapılmaz pek. çalıştırılır, çalışmıyorsa silinir.

o ec2-user@docker_instance# docker run -it alpine ash 
Unable to find image alpine: latest' locally 
latest: Pulling from library/ alpine 
63b65145d645: Pull complete 
Digest: sha256:ff6bdca17Ø1f3a8a67e328815ff2346bøe4Ø67d32ec36b7992c1fdc0Ø1dc8517 
Status: Downloaded newer image for alpine: latest 
/ # Is 
bin etc 
dev horne 
lib 
media 
mnt 
opt 
proc 
root 
run 
sbin 
srv 
sys 
tmp 
usr 
var 

ls dediğimizde bir linux image gibi olduğunu görürüz.

cat /etc/os-release

komutu da bir işletim sistemi olduğunu gösterir bize

/ # cat /etc/os-release 
NAME="A1pine Linux" 
ID—alpine 
VERSION ID-3.17.2 
PRETTY Linux v3.17" 
HOME URL="https://alpinelinux.org/" 
BUG REPORT URL="https://git1ab.alpinelinux.org/alpine/aports/-/issues" 

cd home ile home a gidiyoruz:

/ # cd home 
/home # Is 
/home # 

ls dedik ve boş olduğunu gördük.

burada short-life.txt dosyasını oluşturduk ve ls ile gördük:

/home # touch short-life. txt && Is 
short -life. txt 

konteynırdan exit ile çıkalım ve docker ps -a ile konteynırı görelim.

/home # exit 
• ec2-user@docker_instance# docer ps -a 
CONTAINER ID IMAGE 
9b9313e66b9e alpine 
o ec2-use 
COBWIAND CREATED 
"ash " 
4 minutes 
ago 
STATUS 
Exited 
(0) 
3 
seconds 
PORTS 
ago 
WIES 
kind_engelbart 

konteynır çalışmadığı için -a yı kullandık. docker ps sadece çalışanları gösterir. yeni komut sisteminde docker container ls

• ec2-user@docker_instance# docker 
REPOSITORY TAG 
alpine 
latest 
IMAGE ID 
b2aa39c3Ø4c2 
image Is 
CREATED 
4 weeks ago 
SIZE 
7. Ø5MB 
r@docker_instance# 
o ec2-use 

docker image ls ile de alpine image inin boyutunu gördük.

bizim konteynır içinde oluşturduğumuz dosya duruyor mu? konteynır silinmediği için duruyor. şimdi konteynırı start edelim:

• ec2-user@docker_instance# docker start 9b9 
9b9 

o ec2-user@docker_instance# 
cd hom && Is 
ash: 
cd: can't cd to hom: 
cd home && Is 
short -life. txt 
docker exec -it 9b9 ash 
No such file or directory 

exec komutu ile konteynırın terminaline geldik ve home a gidip ls dedik dosyayı gördük. konteynır silinmediği için sadece stop edildiği için data duruyor.

tekrar exit ile çıkıyoruz.

rm komutuyla konteynırı silelim:

docker rm -f 9b9 konteynır çalıştığı için silebilmek -f kullanıyoruz

• ec2-user@docker_instance# docker ps -a 
CONTAINER ID IMAGE 
COBWIAND CREATED 
STATUS 
PORTS 
WIES 
o ec2-user@docker_instance# & : 

çalışan veya stopta olan başka bir konteynır yok.

docker volume create cw-vol

komutuyla cw-vol isimli bir docker volume oluşturalım.

docker volume inspect cw-vol diyerek volume ü inceleyelim:

"CreatedAt : " 
" 2Ø23-Ø3-16T19:22: , 
"Driver": "local" , 
" Labels " 
"Mountpoint " : "/var/lib/docker/volumes/cw-vol/_data" , 
"Name": "cw-vol" , 
"Options " • 
"Scope": "local" 

/var/lib/docker/volumes altında oluşturuyor volume ler.

sudo ls -al ile bakalım volume içine:

• ec2-user@docker_instance# 
total 
drwxr-xr-x 2 root root 6 
drwx- 
-x 3 root root 19 
Mar 
Mar 
sudo Is -al /var/lib/docker/volumes/cw-vol/_data 
16 19:22 . 
16 19:22 . 

içinde hiçbri şey yok. datalar burada toplanacak.

burada volume boyutu instance ımızın boş alanı kadar. limit belirtmediğimiz için kullanabildiği kadar kullanır.

docker run -it --name clarus -v cw-vol:/cw alpine ash

komutuyla alpine image ile bir konteynır oluşturup cw-vol volume u cw klasörüne mount etmesini ve ash terminalini bize vermesini istiyoruz. konteynır içinde cw klasörü yok normalde biz buraya yazdığımız için oluşturacak.

o ec2-user@docker_instance# docker run 
-it --name clarus -v cw-vol: /cw alpine ash 
/ # Is 
bin 
dev 
etc 
horne 
lib 
media 
mnt 
opt 
proc 
root 
run 
sbin 
srv 
sys 
tmp 
usr 
var 

ls ile cw yi gördük. cw ye girip ls diyelim:

/ # cd cw && Is 
/cw # 

cw içinde bir dosya yazdıralım:

/cw # echo I' This file is created in the container Clarus" 
/cw # Is 
i -will -persist . txt 
/cw # cat i-will-persist . txt 
This file is created in the container Clarus 
/cw # 
> i-will-persist. txt 

ls ve cat komutlarıyla da dosyayı ve içeriğini görüyoruz.

exit ile çıkıp docker ps -a diyelim:

• ec2-user@docker_instance# docker ps -a 
CONTAINER ID IMAGE 
ff11Øbe58ecc 
alpine 
COBWIAND 
"ash " 
CREATED 
2 minutes 
ago 
STATUS 
Exited 
(0) 
3 
seconds 
PORTS 
ago 
WIES 
clarus 

şimdi konteynırımızı silersek i-will-persist.txt dosyası duracak mı? deneylim:



• ec2-user@docker_instance# docker ps -a 
CONTAINER ID IMAGE 
COBWIAND CREATED 
r@docker_instance# & 
o ec2-use 
STATUS 
PORTS 
WIES 

başka çalışan konteynırımız da yok. şimdi volume tutulan klasöre gidiyoruz:

sudo ls -al /var/lib/docker/volumes/cw-vol/\_data && sudo cat /var/lib/docker/volumes/cw-vol/\_data/i-will-persist.txt

r@docker_instance# sudo Is -al 
• ec2-use 
/volumes/ cw-vol/ _data/i -will -persist . txt 
total 4 
drwxr-xr-x 2 root root 32 Mar 16 19:34 
/var/lib/docker/volumes/cw-vol/_data && sudo cat /var/lib/docker 
drwx- 
-x 3 root root 19 Mar 16 19:22 . 
-rw-r--r-- 1 root root 45 Mar 16 19:34 i-will-persist.txt 
This file is created in the container Clarus 
o ec2-use 
cker instance# N: 

dosyayı konteynırda volume un mount olduğu dizinde oluşturmuştuk, volume bağlı olduğu için dosyayı volume tuttu. konteynır gittiği halde veri elimizde.

(konteynırlardan da image alınıp, o image dan yeni konteynır oluşturulabilir)

şimdi bir volume ü birden fazla konteynıra bağlayacağız:

bunun için yeni bir konteynır oluşturuyoruz:

docker run -it --name clarus2nd -v cw-vol:/cw2nd alpine ash

docker run -it alpine komutumuzun ham hali bu. araya isim ekliyoruz:

docker run -it --name clarus2nd alpine şimdi de volume komutlarını ekliyoruz:

docker run -it --name clarus2nd -v cw-vol:/cw2nd alpine ash

cw-vol hazırdaki volume, konteynır içinde cw2nd klasörüne bağla dedik. böyle bir klasör olmadığı için oluşturacak:

o ec2-user@docker_instance# docker run 
-it --name clarus2nd 
-v cw-vol: /cw2nd alpine ash 
/ # Is 
bin 
cw2nd 
dev 
etc 
horne 
lib 
media 
mnt 
opt 
proc 
root 
run 
sbin 
srv 
sys 
tmp 
usr 
var 

cw2nd klasörüne gidip ls dersek:

/ # cd cw2nd/ 
/cw2nd # Is 
i -will -persist . txt 
/cw2nd # 

dosyaımızı gördük. cat ile içine bakalım:

/cw2nd # cat i-will-persist . txt 
This file is created in the container Clarus 
/cw2nd # 

şimdi cw2nd içinde başka bir data oluşturalım:

echo "This is a file of the container Clarus2nd" > loadmore.txt

loadmore.txt isimli ikinci bir dosya oluşturduk.

/cw2nd # Is 
i -will-persist . txt loadmore. txt 
/cw2nd # cat loadmore. txt 
This is a file of the container Clarus2nd 
/cw2nd # 

aslında konteynır içindeki uygulamanın çalışıp yeni bir data ürettiği durumu simüle ediyoruz.

exit le çıkıp başka bir konteynır oluşturup aynı volume u bu yeni konteynıra bağlıyoruz:

/cw2nd # exit 
D ec2-user@docker_instance# docker run -it 
Unable to find image 'ubuntu:latest' locally 
latest: Pulling from library/ubuntu 
2abØ9bØ27e7f: Pull complete 
-name clarus3rd 
-v cw-vol: /cw3rd ubuntu bash 
Digest: sha256:67211c14fa74fØ7Ød27cc59d69a7fa9aeff8e28ea118ef3babc295aØ428a6d21 
Status: Downloaded newer image for ubuntu:latest 
root@Ø6d6fØ13852d 
:/# Is -1 /cw3rd 
total 8 
root 45 Mar 16 19:34 i-will-persist.txt 
-rw-r--r-- 1 root 
root 42 Mar 16 19:45 loadmore. txt 
-rw-r--r-- 1 root 
root@Ø6d6fØ13852d : 

bu kez ubuntu image ile bir konteynır oluşturup bash shell ine bağlandık. volume ü cw3rd klasörüne bağladık. ve klasörün içine girip ls dediğimizde volume deki dosyaları burada görebiliyoruz.

root@Ø6d6fØ13852d 
:/# Is 
bin 
cw3rd etc 
lib 
boot dev horne 
lib32 
root@Ø6d6fØ13852d : 
lib64 
libx32 
media 
mnt 
opt 
proc 
root 
run 
sbin 
sys 
tmp 
usr 
var 

biz cw3rd ü root dizininde oluşturduk, ubuntu image içinde bulunan diğer dizinler içinde (örneğin home) de oluşturabilirdik.

şimdi cw3rd içinde başka bir data oluşturalım:

cd cw3rd && touch file-from-3rd.txt && ls

cd cw3rd && touch file-from-3rd . txt && Is 
file-from-3rd . txt i -will-persist . txt loadmore. txt 

exit ile çıkıyoruz. docker image ls diyerek imagelerin boyutlarına bakabiliriz:

• ec2-user@docker_instance# docker 
REPOSITORY TAG 
ubuntu 
alpine 
o ec2-use 
latest 
latest 
IMAGE ID 
08d22cøceb15 
b2aa39c3Ø4c2 
image Is 
CREATED 
8 days ago 
4 weeks ago 
SIZE 
77.8MB 
7. Ø5MB 

bu ubuntu gerçek bir ubuntu image ı değil ancak alpine a kıyasla bayağı büyük.

dördüncü bir konteynır oluşturuyoruz. diğer konteynırlar stopta ancak silinmedi.

docker run -it --name clarus4th -v cw-vol:/cw4th:ro ubuntu bash

dördüncü konteynırı oluştururken optional olarak ro ekledik ( sadece okuyabilir) yani volume ü bağladığımız yere yeni bir data ekleyemez bu konteynır.

root@c81537593Ø3d 
total 8 
-rw-r--r-- 1 root 
-rw-r--r-- 1 root 
-rw-r--r-- 1 root 
:/# Is -1 /cw4th 
root Mar 16 19:51 file-from-3rd . txt 
root 45 Mar 16 19:34 i-will-persist.txt 
root 42 Mar 16 19:45 loadmore. txt 

volume deki dosyalarımızı gördük. burada yeni bir dosya oluşturabilecek miyiz?

cd cw4th && touch file-from-ath . txt 
touch: cannot touch 'file-from-4th.txt' : Read-only file system 

mount ettiğimiz dizinde ro only olduğu için oluşmaz ancak diğer dizinlerde dosya oluşturabilir. ancak ne anlamı var\_ volume buraya bağlı olmadığı için konteynırı silince gidecek.

exit ile çıkıyoruz.

bir volume ü birden fazla konteynıra bağlayabildiğimizi gördük. volume içine bakalım:

• ec2-user@docker_instance# 
total 8 
drwxr-xr-x 2 root root 77 
19:51 
19:22 
sudo Is 
drwx- 
-x 3 root root 19 
-r--r-- 1 root root 
-rw 
-r--r-- 1 root root 45 
-rw 
-r--r-- 1 root root 42 
-rw 
o ec2-use 
cker instance# 
Mar 
Mar 
Mar 
Mar 
Mar 
16 
16 
16 
16 
16 
19. 
51 
19 
: 34 
19. 
• 45 
-al /var/lib/docker/volumes/cw-vol/_data 
file-from-3rd . txt 
i -will -persist . txt 
loadmore. txt 

dataları görebiliyoruz.

• ec2-user@docker_instance# docker ps -a 
CONTAINER ID IMAGE 
c81537593Ø3d 
ubuntu 
Ø6d6fØ13852d 
ubuntu 
233fØ22b0f72 alpine 
o ec2-use 
COBWIAND 
"bash " 
"bash " 
"ash " 
CREATED 
5 minutes ago 
11 minutes ago 
16 minutes ago 
STATUS 
Exited 
Exited 
Exited 
(1) 
(0) 
(0) 
About a minute 
7 minutes ago 
11 minutes ago 
PORTS 
ago 
WIES 
clarus4th 
clarus3rd 
clarus2nd 

konteynırlarımızın hepsi stop durumda. hepsini silelim:

docker rm clarus4th clarus3rd clarus2nd

•$ docker rm clarus4th clarus3rd clarus2nd 
• ec2-user@docker_instance# 
clarus4th 
clarus3rd 
clarus2nd 
o ec2-user@docker 
_instance# 

konteynırlar silindi ancak volume duruyor:

• ec2-user@docker_instance# 
19:51 
19:22 
sudo Is 
total 8 
drwxr-xr-x 2 
drwx- 
root 
root 
root 
root 
root 
root 
root 
root 
root 
77 
19 
45 
42 
Mar 
Mar 
Mar 
Mar 
Mar 
16 
16 
16 
16 
16 
19. 
51 
19 
: 34 
19. 
• 45 
-al /var/lib/docker/volumes/cw-vol/_data 
file-from-3rd . txt 
i -will -persist . txt 
loadmore. txt 

volume ü de silelim:

O 
ec2-user@docker_instance# docker volume rm cw-vol 
cw-vol 
r@docker_instance# sudo Is -al /var/lib/docker/volumes/cw-vol/_data 
ec2-use 
Is: cannot access /var/lib/docker/volumes/cw-vol/_data: No such file or directory 
ec2-use 

şimdi yeni iki tane volume oluşturacaz biri empty-volume diğeri full-volume olacak

docker volume create empty-vol

docker volume create full-vol

şimdi alpine image ından vol-lesson isimli bir konteynır oluşturup içindeki bir dizine bağlayalım:

docker run -it --name vol-lesson -v full-vol:/cw alpine ash

alpine image ı artık sistemimizde var olduğu için cash ten alıyor. registry e gidip çekmyor her seferinde.

cd cw && echo "This file is created in the full-vol volume" > full.txt

komutuyla cw içinde full.txt oluşturduk.

konteynırdan çıkabiliriz.

r@docker_instance# sudo Is /var/lib/docker/volumes/full-vol/_data 
• ec2-use 
full. txt 

full.txt yi volume de gördük.

şimdi clarusway/hello-clarus

image ini kullanacaz. image i tanıyalım:

docker run -it --name clarus clarusway/hello-clarus sh

image ile bir konteynır oluşturup bağlandık:

Unable to find image ' clarusway/hello-clarus:latest' 
latest: Pulling from clarusway/hello-clarus 
df2øfa9351a1: Pull complete 
36b3adc4ff6f: Pull complete 
b3daØ9547f4Ø: Pull complete 
f253d7fd1Ø8d: Pull complete 
eaea9af357a3: Pull complete 
62371e125991: Pull complete 
locally 
Digest: sha256: 7ebafc89ef64a23Ø1d4fa2Øb88eØ5269ef717fea8571Ø33c24990fc147dd7d14 
Status: Downloaded newer image for clarusway/hello-clarus:latest 

lokalden çekemedim diyor ve registry den çekiyor.

/ # Is 
bin 
dev 
etc 
cd 
app. py 
/hello 
hello-clarus 
horne 
lib 
hello-clarus 
Is 
-clarus # 
media 
mnt 
opt 
proc 
run 
sbin 
sys 
tmp 
usr 
var 

içeirisnde hello-clarus diye bir dizin var onun da içinde app.py var. bu image in içinde olan bir dizin ve dosya bu.

exit ile çıkıyoruz.

1. ve 2. duruma bakalım şimdi:

şimdi try1 isimli bir konteynır oluşturacaz ve boş ve olmayan cw dizinine volume ü mount edecez:

docker run -it --name try1 -v full-vol:/cw clarusway/hello-clarus sh

/ # cd cw && Is 
full. txt 
/cw # 

dosyamız geldi.

exit ile çıkıyoruz.

3.durum neydi; volume boş ve target ta dolu bir dosya var:

bizim bu durumu simüle etmek için boş volume ümüz vardı: empty-vol:

sudo ls /var/lib/docker/volumes/empty-vol/\_data

docker run -it --name try2 -v empty-vol:/hello-clarus clarusway/hello-clarus sh

komutu ile konteynır içinde zaten var olan ve dolu olan folder a volume ü mount ediyoruz.

daha sonra konteynırdan çıkıp empty-vol e bakıyoruz:

r@docker_instance# sudo Is /var/lib/docker/volumes/empty-vol/_data 
• ec2-use 
app. py 

empty-vol de dolu bir volume oldu.

1. durumu simüle edelim:

full-vol doluydu zaten:

sudo Is /var/lib/docker/volumes/full-vol/_data 
O 
full. txt 
ec2-user@docker_instance# 

clarusway/hello image ile bir konteynır oluşturuyoruz ismine try3 veriyoruz ve full-vol ü dolu bir dizin olan hello-clarus dizinine bağlıyoruz:

docker run -it --name try3 -v full-vol:/hello-clarus clarusway/hello-clarus sh

/ # cd hello-clarus/ && Is 
full. txt 
/hello-clarus # 

cd ile hello-clarus içine girdiğimizde app.py nin gittiğini ve full-vol içeriği olan full.txt nin geldiğini görüyoruz.

exit ile çıkalım.

şimdi konteynır ve volume leri siliyoruz:

docker container prune

docker volume prune

• ec2-user@docker_instance# docker container prune 
WARNING! This will remove all stopped containers . 
Are you sure you want to continue? [y/N] y 
Deleted Containers: 
Id82e817ab165a95cØd33bc5ffc5Ø1916129d8f4Ø975dØdc8da1a938ec6352b1 
aa24e51b8a1b1faøe64352e68eøcd55c417fØ94d39d9b34Ø6Øb9475d926723fb 
2e1c7e3b86117592Ø29467a566c18eØ13f8de3ef8Øa73b325b56aba7e65f5bd3 
1bØbf7c1fdce83bØ8ef1øa28189478d7eØ8e41542b77f59245f91fcdb6276f81 
3de2fcøe7d8cca91d312dd7770fa13442cØ1dc2673eea2a134c224cd4Ø77fØb2 
Total reclaimed space: 169B 
• ec2-user@docker_instance# docker volume prune 
WARNING! This will remove all local volumes not used by at least one container. 
Are you sure you want to continue? [y/N] y 
Deleted Volumes : 
full-vol 
empty -vol 
Total reclaimed space: 
70B 

ls ile de kontrol edelim:

• ec2-user@docker_instance# docker container Is 
CONTAINER ID IMAGE 
COMMAND CREATED STATUS 
PORTS 
WIES 
• ec2-user@docker_instance# docker volume Is 
DRIVER 
o ec2-use 
VOLUME 
cker instance# N: 

çalışan bir konteynıra bağlanırken docker exec -it name bash ile bağlanırız.

bind mounts konusuna geçiyoruz:

olan bir dizini bağlama işlevini yerine getiriyor.

nginx imajı ile nginx-default isimli bir konteynır kaldırıyoruz ve detached modunda çalıştıracaz. bizim 80 portumuzu konteynırın 80 portuna bağlıyoruz.

docker run -d --name nginx-default -p 80:80 nginx

şimdi gidip public ip yi alırsak nginx in çalıştığını görürüz.

çalışan konteynıra docker exec ile bağlanıyorduk:

docker exec -it nginx-default bash



cd /usr/share/nginx/html 
root@5e11e6adbb45:/usr/share/nginx/html# Is 
50x. html index. html 
root@5e11e6adbb45:/usr/share/nginx/html# 

belirtilen path tekiindex.html dosyası sayesinde browserdaki sayfa görünüyor.

şimdi exit ile çıkalım

webpage isimil bir klasör oluşturup bunun içinde index.html oluşturacaz:

mkdir webpage && cd webpage

echo "<h1>Welcome to Clarusway</h1>" > index.html

mkdir 
webpage : 
webpage : 
webpage && cd webpage 
echo "<h1>We1come to Clarusway</hl>" 
> index. html 

şimdi yeni bir konteynır oluşturacaz ismi nginx-new bu 8080 portundan çalışacak. bu kez volume oluşturmadık dizini bağlayacağız. bu nedenle absolute path yani tam adres yolunu yazıyoruz:

docker run -d --name nginx-new -p 8080:80 -v /home/ec2-user/webpage:/usr/share/nginx/html nginx

volume oluşturduğumuzda onu bağlamak için volume adını yazmak yetiyordu.

yeni index.html i nginx e bağladığımız için browser daki sayfa değişti.



bu özellik sayesinde web sayfasında yapılan bir değişiklik çok basit şekilde gözlemlenebilir.

nginx konteynırlar stop olmadığı için rm -f ile silebilir ya da önce stop edip sonra silerbiliriz.

docker rm -f $(docker ps -aq)

bu komut calisan konteynirlari tek tek id sini yazip silmek yerine hepsini siliyor