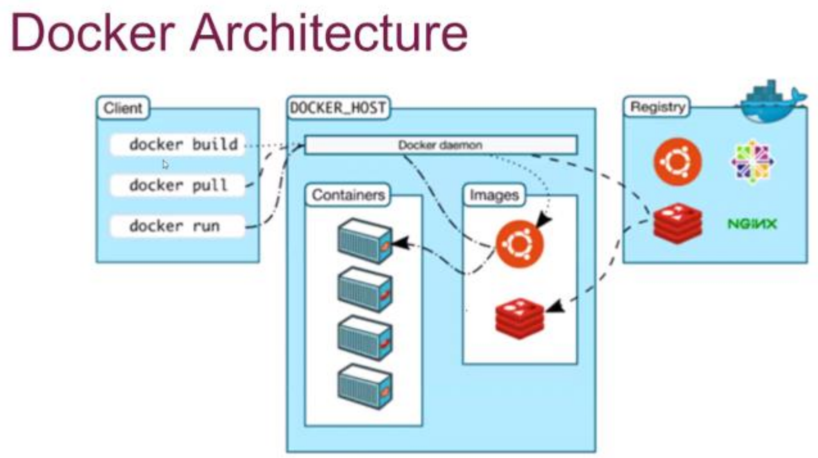
Docker-2 Volume

Wednesday, 8 September 2021 20.08



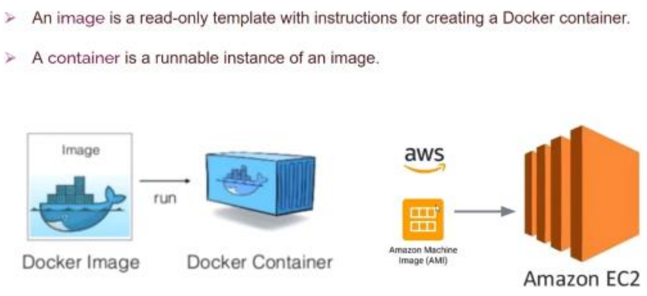
Docker Architecture

Container kisaca hatirlayacak olursak tek uygulama nin calismasi icn hazirlanmis Paketler dir ,
Client server mimaris vardi islemci server mimarisi.

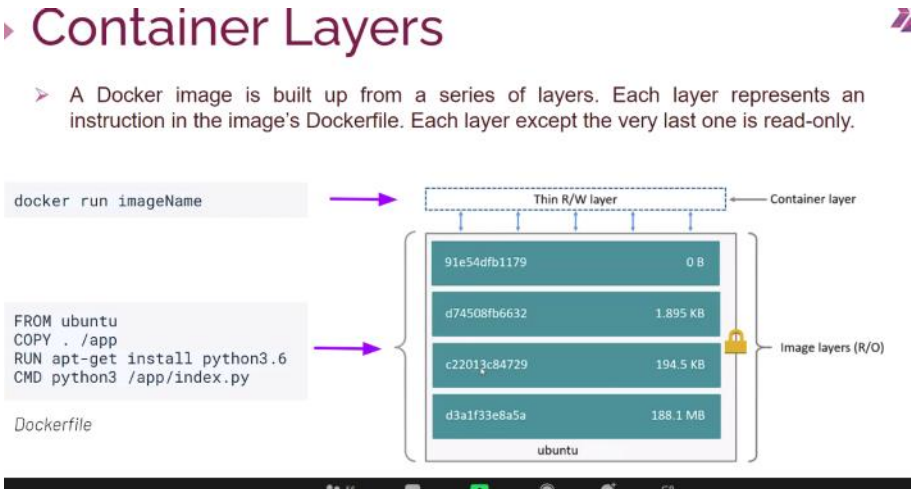
Client docker girilecek terminal
Docker Host; Docker yukledigimiz maKINA
Register repoydu

containerlar imajlardan olusuyorlardi.

Once Docker pull ile imajlar cekiliyor daha sonra Docker run ile calistiriliyor.



- An image is a read-only template with instructions for creating a Docker container.
- A container is a runnable instance of an image.



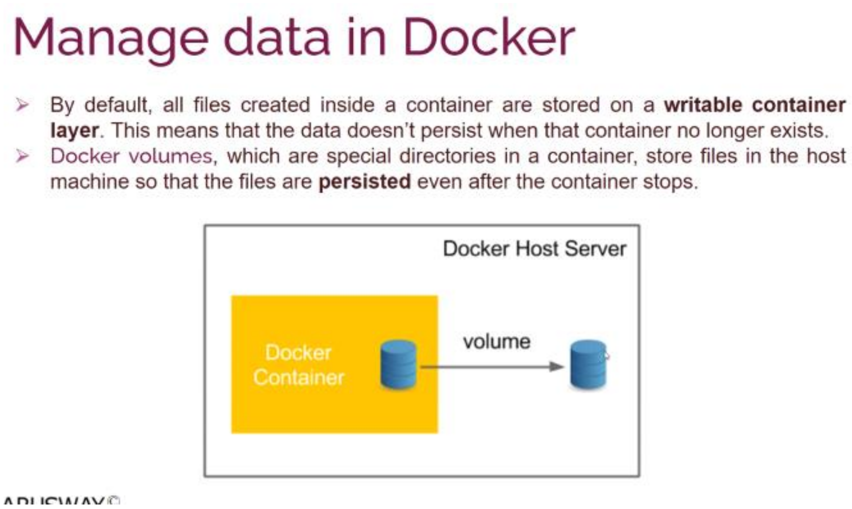
Container Layers

- 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
```

```
FROM ubuntu
COPY . /app
RUN apt-get install python3.6
CMD python3 /app/index.py
```

Dockerfile

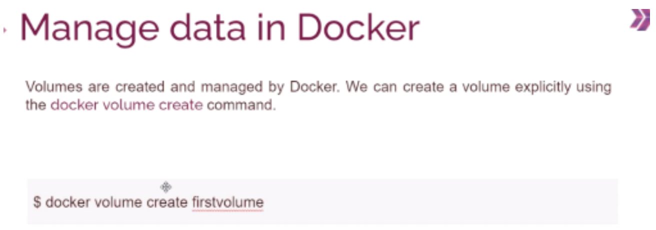


Manage data in Docker

- 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.

Containerda olusturdugumuz veriler conatainer kapandiktan sonra silinir bunlarin silinmemesi icin volumler ilusturulmustur.

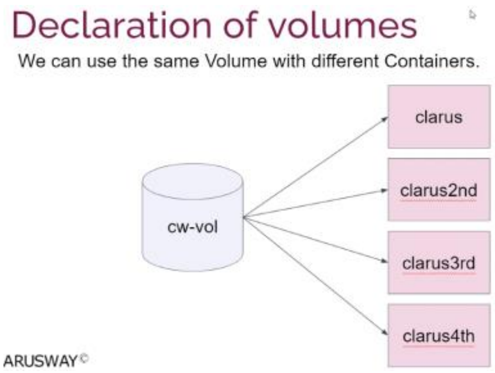
Container da Telefon defterimizin
Docker volume create etmenin ilk yolu asagida



Manage data in Docker

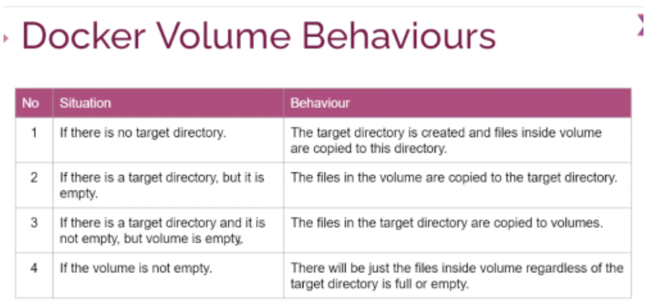
Volumes are created and managed by Docker. We can create a volume explicitly using the docker volume create command.

```
$ docker volume create firstvolume
```



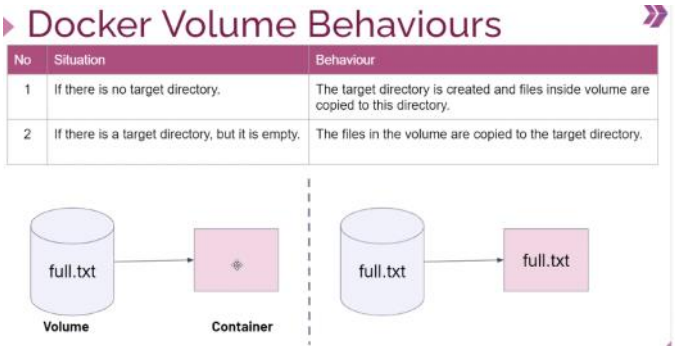
Declaration of volumes

We can use the same Volume with different Containers.



Docker Volume Behaviours

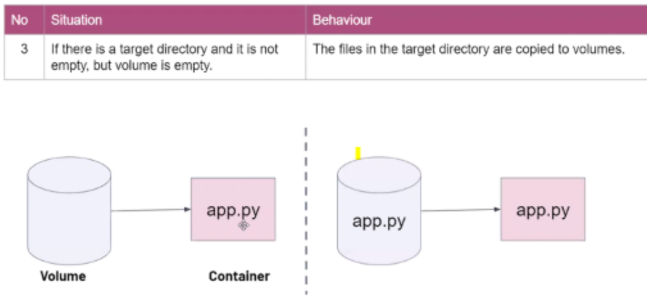
No	Situation	Behaviour
1	If there is no target directory.	The target directory is created and files inside volume are copied to this directory.
2	If there is a target directory, but it is empty.	The files in the volume are copied to the target directory.
3	If there is a target directory and it is not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



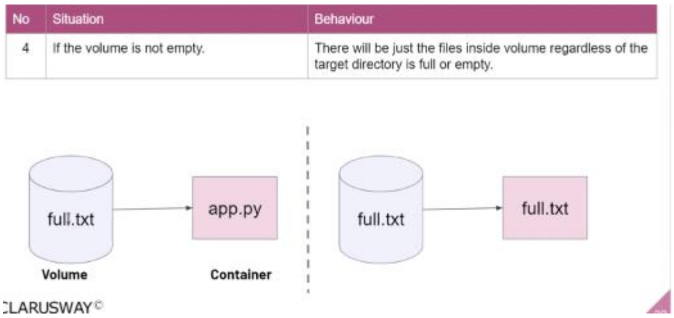
Docker Volume Behaviours

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

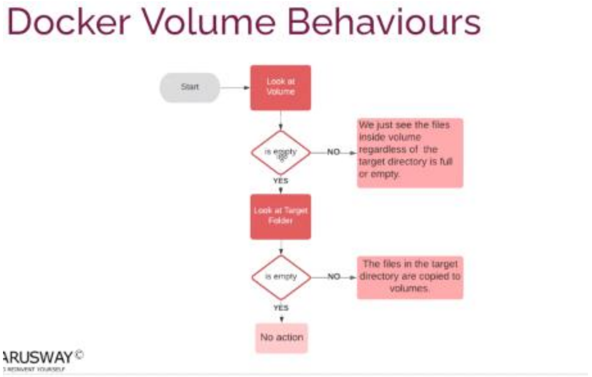
Volume her turlu containeri ezer volu de icerisinde bir sey varsa yukler



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



No	Situation	Behaviour
4	If the volume is not empty.	There will be just the files inside volume regardless of the target directory is full or empty.



Docker Volume Behaviours

Start

Look at Volume

is empty

NO

We just see the files inside volume regardless of the target directory is full or empty.

YES

Look at Target Folder

is empty

NO

The files in the target directory are copied to volumes.

YES

No action

# Hands-on Docker-03 : Handling Docker Volumes

Purpose of the this hands-on training is to teach students how to handle volumes in Docker containers.

## Learning Outcomes

At the end of the this hands-on training, students will be able to;

- explain what Alpine container is and why it is widely used.
- list available volumes in Docker.
- create a volume in Docker.
- inspect properties of a volume in Docker.
- locate the Docker volume mount point.
- attach a volume to a Docker container.
- attach same volume to different containers.
- delete Docker volumes.

## Outline

- Part 1 - Launch a Docker Machine Instance and Connect with SSH
- Part 2 - Data Persistence in Docker Containers
- Part 3 - Managing Docker Volumes
- Part 4 - Using Same Volume with Different Containers
- Part 5 - docker volume behaviours
- Part 6 - Bind Mounts

## Part 1 - Launch a Docker Machine Instance and Connect with SSH

- Launch a Docker machine on Amazon Linux 2 AMI with security group allowing SSH connections using the [Cloudformation Template for Docker Machine Installation](../docker-01-installing-on-ec2-linux2/docker-installation-template.yml).

- Connect to your instance with SSH.

```bash

ssh -i .ssh/call-training.pem ec2-user@ec2-3-133-106-98.us-east-2.compute.amazonaws.com
```

## Part 2 - Data Persistence in Docker Containers

- Check if the docker service is up and running.

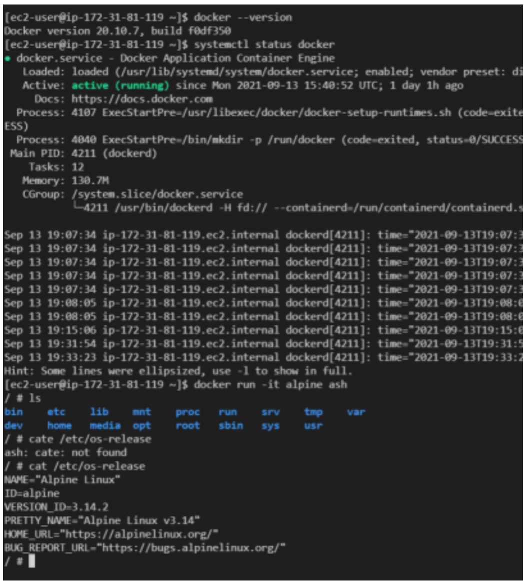
```bash
systemctl status docker  # docker in aktif olup olmadigini görduk
```

- Run a `alpine` container with interactive shell open, and add command to run alpine shell. Here, explain explain what the alpine container is and why it is so popular. (Small size, Secure, Simple, Fast boot)

```bash
docker run -it alpine ash  # docker a run komutu ile container olusturup calistiriyoruz ayrica interaktif sekilde baglaniyoruz ve alpine imajindan create ediyoruz.
```

- Display the os release of the alpine container.

```bash
cat /etc/os-release
```



```
[ec2-user@ip-172-31-81-119 ~]$ docker --version
Docker version 20.10.7, build f0df350
[ec2-user@ip-172-31-81-119 ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: di
   Active: active (running) since Mon 2021-09-13 15:40:52 UTC; 1 day 1h ago
     Docs: https://docs.docker.com
  Process: 4107 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exite
ESS)
  Process: 4040 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS
 Main PID: 4211 (dockerd)
    Tasks: 12
   Memory: 130.7M
   CGroup: /system.slice/docker.service
           └─4211 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.s

Sep 13 19:07:34 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:07:3
Sep 13 19:07:34 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:07:3
Sep 13 19:07:34 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:07:3
Sep 13 19:07:34 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:07:3
Sep 13 19:07:34 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:07:3
Sep 13 19:08:05 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:08:0
Sep 13 19:08:05 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:08:0
Sep 13 19:15:06 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:15:0
Sep 13 19:31:54 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:31:5
Sep 13 19:33:23 ip-172-31-81-119.ec2.internal dockerd[4211]: time="2021-09-13T19:33:2
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-81-119 ~]$ docker run -it alpine ash
/ # ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
/ # cate /etc/os-release
ash: cate: not found
/ # cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.14.2
PRETTY_NAME="Alpine Linux v3.14"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
/ # 
```

- Create a file named `short-life.txt` under `/home` folder

```bash
cd home && touch short-life.txt && ls
```

- Exit the container and return to ec2-user bash shell.

```bash
exit
```

- Show the list of all containers available on Docker machine.

```bash
docker ps -a
```

- Start the alpine container and connect to it.

```bash
docker start 737 && docker attach 737
```

- Show that the file `short-life.txt` is still there, and explain why it is there. (Container holds it data until removed).

```bash
ls /home
```

- Exit the container and return to ec2-user bash shell.

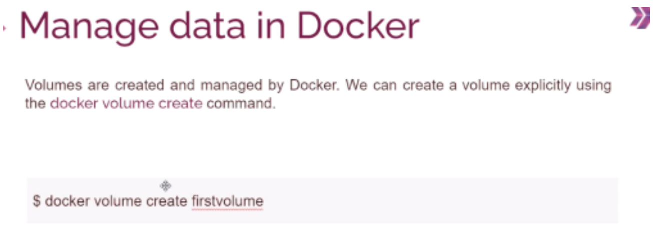
```bash
exit
```

- Remove the alpine container. Sildik ancak container da gitti

```bash
docker rm 737
```

- Show the list of all containers, and the alpine container is gone with its data.

```bash
docker ps -a
```



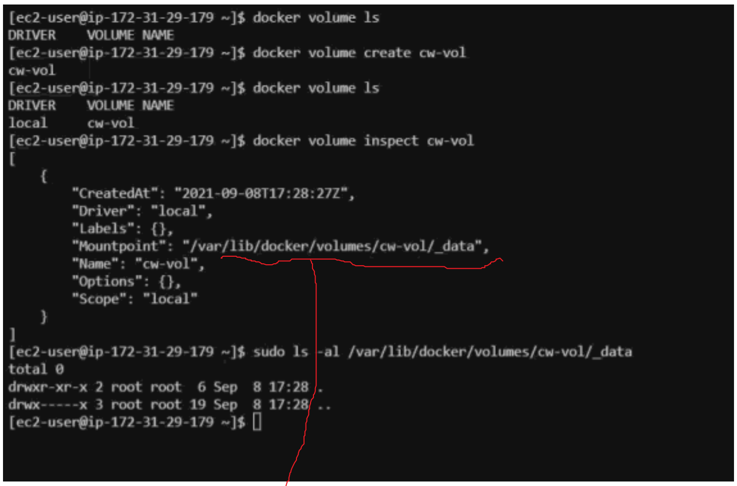
Manage data in Docker

Volumes are created and managed by Docker. We can create a volume explicitly using the docker volume create command.

```
$ docker volume create firstvolume
```

## Part 3 - Managing Docker Volumes
- Explain why we need volumes in Docker.
- List the volumes available in Docker, since not added volume before list should be empty.

```bash
docker volume ls  #  ---'docker da bulunan volume leri listeler
```
- Create a volume named `cw-vol`.
```bash
docker volume create cw-vol  # yeni bir volume create ettik
```
- List the volumes available in Docker, should see local volume `cw-vol` in the list.
```bash
docker volume ls  #   ve listeledik
```
- Show details and explain the volume `cw-vol`. Note the mount point: `/var/lib/docker/volumes/cw-vol/_data`.
```bash
docker volume inspect cw-vol   # inspect komutu ile volumu kontrol ettik inceledik
```
- List all files/folders under the mount point of the volume `cw-vol`, should see nothing listed.



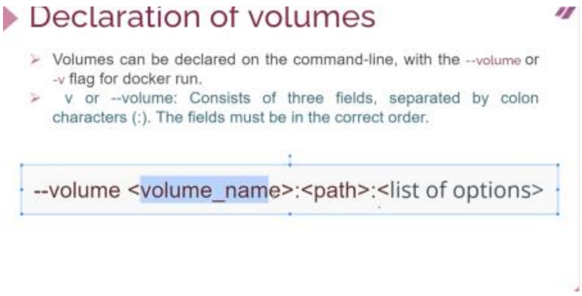
```
[ec2-user@ip-172-31-29-179 ~]$ docker volume ls
DRIVER    VOLUME NAME
[ec2-user@ip-172-31-29-179 ~]$ docker volume create cw-vol
cw-vol
[ec2-user@ip-172-31-29-179 ~]$ docker volume ls
DRIVER    VOLUME NAME
local     cw-vol
[ec2-user@ip-172-31-29-179 ~]$ docker volume inspect cw-vol
[
    {
        "CreatedAt": "2021-09-08T17:28:27Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/cw-vol/_data",
        "Name": "cw-vol",
        "Options": {},
        "Scope": "local"
    }
]
[ec2-user@ip-172-31-29-179 ~]$ sudo ls -al /var/lib/docker/volumes/cw-vol/_data
total 0
drwxr-xr-x 2 root root  6 Sep  8 17:28 .
drwx-----x 3 root root 19 Sep  8 17:28 ..
[ec2-user@ip-172-31-29-179 ~]$ 
```

```bash
sudo ls -al  /var/lib/docker/volumes/cw-vol/_data   # volumu göruntuledik ve icerisine girdik
```
- Run a `alpine` container with interactive shell open, name the container as `clarus`, attach the volume `cw-vol` to `/cw` mount point in the container, and add command to run alpine shell. Here, explain `--volume` and `v` flags.
**```bash**

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

▶ **# ```amacimiz container olusturmak ve olustururken volume baglamak**
Interaktif bir sekilde volume olusturduk ve isim verdik

▶ **-v ile volumu bagliyoruz nereye baglayacaksak volume ismimizi yaziyoruz**
: dan sonra containerda nereye baglamak istiyorsak onu yaziyoruz alpine



Declaration of volumes

- Volumes can be declared on the command-line, with the --volume or -v flag for docker run.
- v or --volume: Consists of three fields, separated by colon characters (:). The fields must be in the correct order.

--volume <volume_name>:<path>:<list of options>

Su ana kadar bir volume olusturduk ve icerisinde alpine imaj bir contaixner olusturduk

Farkli containerlarda ayni volumu baglamayi görecegiz

- List files/folder in `clarus` container, show mounting point `/cw`, and explain the mounted volume `cw-vol`.

```bash
ls
```

- Create a file in `clarus` container under `/cw` folder.

```bash
cd cw && echo "This file is created in the container Clarus" > i-will-persist.txt
```

- List the files in `/cw` folder, and show content of `i-will-persist.txt`.

```bash
ls && cat i-will-persist.txt
```

- Exit the `clarus` container and return to ec2-user bash shell.

```bash
exit
```

- Show the list of all containers available on Docker machine.

```bash
docker ps -a
```

- Remove the `clarus` container.

```bash
docker rm clarus
```

- Show the list of all containers, and the `clarus` container is gone.

```bash
docker ps -a
```

- List all files/folders under the volume `cw-vol`, show that the file `i-will-persist.txt` is there.

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

Farkli containerlarda ayni volumu baglamayi görecegiz

## Part 4 - Using Same Volume with Different Containers

- Run a `alpine` container with interactive shell open, name the container as `clarus2nd`, attach the volume `cw-vol` to `/cw2nd` mount point in the container, and add command to run alpine shell.

Ayni volume baska bir containera bagladik yenir txt dosyasin olusturduk yeni bir data attik icerisine

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

- List the files in `/cw2nd` folder, and show that we can reach the file `i-will-persist.txt`.

```bash
ls -l /cw2nd && cat /cw2nd/i-will-persist.txt
```

- Create an another file in `clarus2nd` container under `/cw2nd` folder.

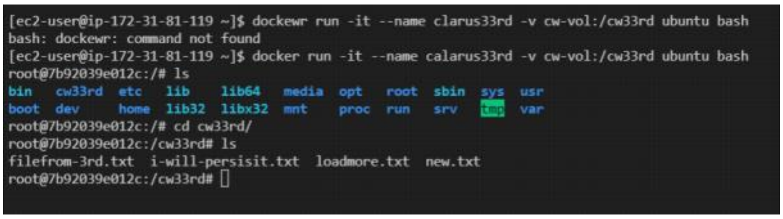
```bash
cd cw2nd && echo "This is a file of the container Clarus2nd" > loadmore.txt
```

- List the files in `/cw2nd` folder, and show content of `loadmore.txt`.

```bash
ls && cat loadmore.txt
```

- Exit the `clarus2nd` container and return to ec2-user bash shell.

```bash
Exit
```



```
[ec2-user@ip-172-31-81-119 ~]$ dockewr run -it --name clarus33rd -v cw-vol:/cw33rd ubuntu bash
bash: dockewr: command not found
[ec2-user@ip-172-31-81-119 ~]$ docker run -it --name calarus33rd -v cw-vol:/cw33rd ubuntu bash
root@7b92039e012c:/# ls
bin   cw33rd  etc   lib    lib64   media  opt   root  sbin  sys  usr
boot  dev     home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@7b92039e012c:/# cd cw33rd/
root@7b92039e012c:/cw33rd# ls
filefrom-3rd.txt  i-will-persisit.txt  loadmore.txt  new.txt
root@7b92039e012c:/cw33rd# 
```

- Run a `ubuntu` container with interactive shell open, name the container as `clarus3rd`, attach the volume `cw-vol` to `/cw3rd` mount point in the container, and add command to run bash shell.

```bash
docker run -it --name clarus3rd -v cw-vol:/cw3rd ubuntu bash
```

- List the files in `/cw3rd` folder, and show that we can reach the all files created earlier.

```bash
ls -l /cw3rd
```

- Create an another file in `clarus3rd` container under `/cw3rd` folder.

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

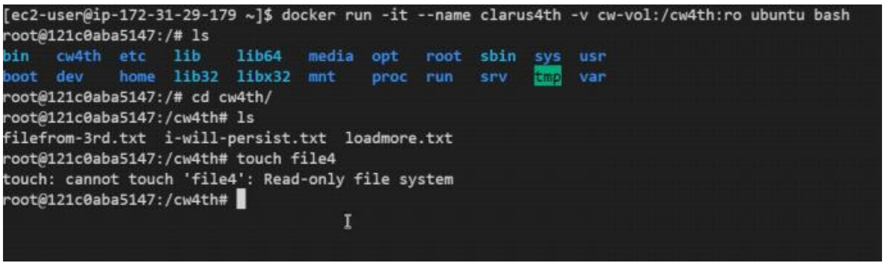
- Exit the `clarus3rd` container and return to ec2-user bash shell.

```bash
exit
```

- Run an another `ubuntu` container with interactive shell open, name the container as `clarus4th`, attach the volume `cw-vol` as read-only to `/cw4th` mount point in the container, and add command to run bash shell.

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

Sadece read only yetkis verdigimiz icin her hangi bir file olusturamiyoruz



```
[ec2-user@ip-172-31-29-179 ~]$ docker run -it --name clarus4th -v cw-vol:/cw4th:ro ubuntu bash
root@121c0aba5147:/# ls
bin   cw4th  etc   lib    lib64   media  opt   root  sbin  sys  usr
boot  dev    home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@121c0aba5147:/# cd cw4th/
root@121c0aba5147:/cw4th# ls
filefrom-3rd.txt  i-will-persist.txt  loadmore.txt
root@121c0aba5147:/cw4th# touch file4
touch: cannot touch 'file4': Read-only file system
root@121c0aba5147:/cw4th# 
```

- List the files in `/cw4th` folder, and show that we can reach the all files created earlier.

```bash
ls -l /cw4th
```

- Try to create an another file under `/cw4th` folder. Should see error `read-only file system`

```bash
cd cw4th && touch file-from-4th.txt
```

- Exit the `clarus4th` container and return to ec2-user bash shell.

```bash
exit
```

- List all containers.

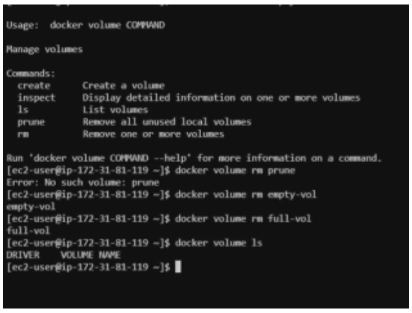
```bash
docker ps -a
```

- Delete `clarus2nd`, `clarus3rd` and `clarus4th` containers.

```bash
docker rm clarus2nd clarus3rd clarus4th
```

- Delete `cw-vol` volume.

```bash
docker volume rm cw-vol
```

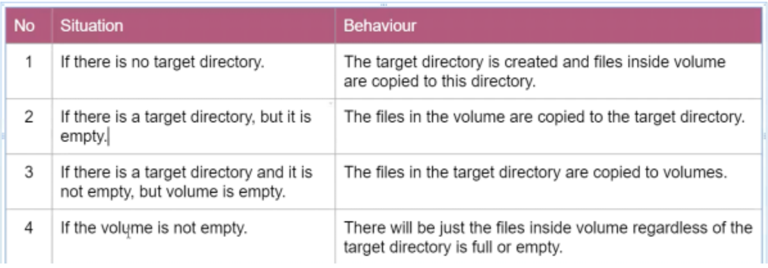


```
Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
[ec2-user@ip-172-31-81-119 ~]$ docker volume rm prune
Error: No such volume: prune
[ec2-user@ip-172-31-81-119 ~]$ docker volume rm empty-vol
empty-vol
[ec2-user@ip-172-31-81-119 ~]$ docker volume rm full-vol
full-vol
[ec2-user@ip-172-31-81-119 ~]$ docker volume ls
DRIVER    VOLUME NAME
[ec2-user@ip-172-31-81-119 ~]$ 
```

## Part 5 - docker volume behaviours   volumlern Davranislari

|No | Situation | Behaviour |

Docker Volume Behaviours

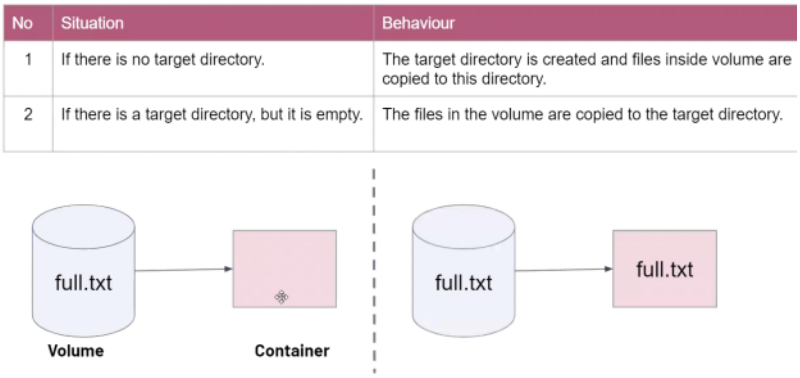


No	Situation	Behaviour
1	If there is no target directory.	The target directory is created and files inside volume are copied to this directory.
2	If there is a target directory, but it is empty.	The files in the volume are copied to the target directory.
3	If there is a target directory and it is not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

4 ayri durumda 4 ayri davranis sergileyebiliyor

| ---- | ----------- | ------------ |
| 1 | If there is no target directory. | The target directory is created and files inside volume are copied to this directory. |

Ilk durumda container yok ise. Volumun icerisindeki file container da bir file create ediyor ve icerisine yerlesiyor

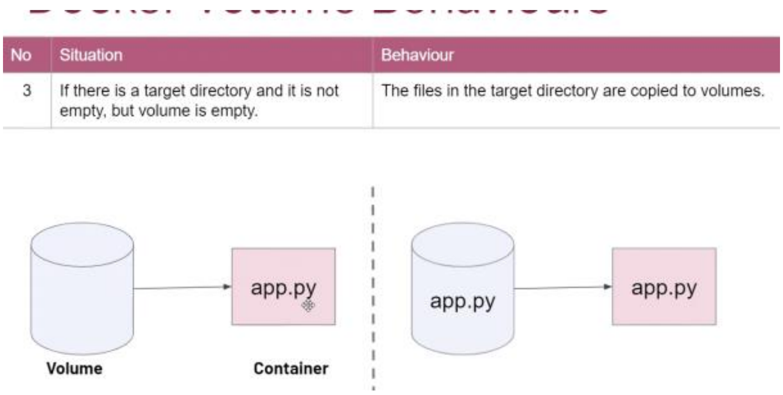


No	Situation	Behaviour
1	If there is no target directory.	The target directory is created and files inside volume are copied to this directory.
2	If there is a target directory, but it is empty.	The files in the volume are copied to the target directory.

| 2 | If there is target directory, but it is empty. | The files in volume are copied to target directory. |

Contanir da bir volume var fakat bos bu durumda da ayni sekilde file icersine yerlesiyoruz

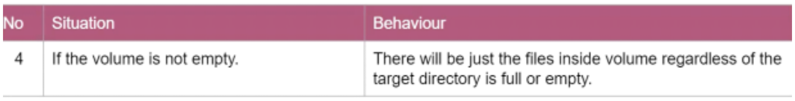
| 3 | If there is target directory and it is not empty, but volume is empty. | The files in the target directory are copied to volumes. |



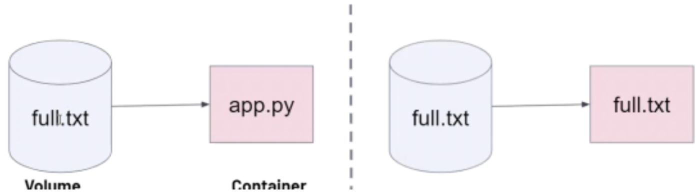
No	Situation	Behaviour
3	If there is a target directory and it is not empty, but volume is empty.	The files in the target directory are copied to volumes.

Container da file var volume de folder yok container da file var ise bu file bizim volumun icerisine kopyalaniyor

| 4 | If the volume is not empty. | There will be just the files inside the volume regardless of the target directory is full or empty. |



No	Situation	Behaviour
4	If the volume is not empty.	There will be just the files inside volume regardless of the target directory is full or empty.



4 uncu durumda volumde ve containerda file var ise containerdaki file siliniyor ve volume icerisindeki containerdaki kopyalaniyor

- Create `empty-vol` and `full-vol` volumes.
```bash
docker volume create empty-vol
docker volume create full-vol
```
- Run an `alpine` container with interactive shell open, name the container as `vol-lesson`, attach the volume `full-vol` to `/cw` mount point in the container, and add command to run alpine shell.
```bash
docker run -it --name vol-lesson -v full-vol:/cw alpine ash
```
- Create a file in `full-vol` container under `/cw` folder.
```bash
cd cw && echo "This file is created in the full-vol volume" > full.txt
```
- Exit the `vol-lesson` container and return to ec2-user bash shell.
```bash
exit
```
- List all files/folders under the volume `full-vol`, show that the file `full.txt` is there.
```bash
sudo ls /var/lib/docker/volumes/full-vol/_data
```
- Run the `clarusway/hello-clarus` container with interactive shell open, name the container as `clarus`, and show the inside of `hello-clarus` directory.
```bash
docker run -it --name clarus clarusway/hello-clarus sh
/ # ls
bin          etc          home         media        opt          root         sbin
    sys          usr
dev          hello-
clarus  lib          mnt          proc         run          srv          tmp          var
/ # cd hello-clarus && ls
app.py
```
- `exit` the container

### Situation-1 and 2:
|No | Situation | Behaviour |
| ---- | ----------- | ------------ |
| 1 | If there is no target directory. | The target directory is created and files inside volume are copied to this directory. |
| 2 | If there is target directory, but it is empty. | The files in volume are copied to target directory. |
![situation 1 and 2](situation-1-and-2.png)
- Run the `clarusways/hello-clarus` container with interactive shell open, name the container as `try1`, attach the volume `full-vol` to `/cw` mount point in the container, and show that `/cw` directory is created and files inside volume are copied to this directory.
```bash
docker run -it --name try1 -v full-vol:/cw clarusway/hello-clarus sh
/ # ls
bin          dev          hello-
clarus  lib          mnt          proc         run          srv          tmp          var
cw           etc          home         media        opt          root         sbin
     sys          usr
/ # cd cw && ls
full.txt
```
- `exit` the container

### Situation-3:
|No| Situation | Behaviour |
| ---- | ----------- | ------------ |

| 3 | If there is target directory and it is not empty, but volume is empty. | The files in the target directory are copied to volumes. |
![situation 3](situation-3.png)
- List all files/folders under the volume `empty-vol`, show that the folder `is empty.
```bash
sudo ls /var/lib/docker/volumes/empty-vol/_data
```
- Run the `clarusway/hello-clarus` container with interactive shell open, name the container as `try2`, attach the volume `empty-vol` to `/hello-clarus` mount point in the container.
```bash
docker run -it --name try2 -v empty-vol:/hello-clarus clarusway/hello-clarus sh
/ # ls
bin          etc          home         media        opt          root         sbin         sys          usr
dev          hello-clarus  lib          mnt          proc         run          srv          tmp          var
/ # cd hello-clarus/ && ls
app.py
```
- `exit` the container.
- List all files/folders under the volume `empty-vol`, show that the file `app.py` is there.
```bash
sudo ls /var/lib/docker/volumes/empty-vol/_data
app.py
```
### Situation-4:
No	Situation	Behaviour
4	If the volume is not empty.	There will be just the files inside the volume regardless of the target directory is full or empty.

![situation 4](situation-4.png)
- List all files/folders under the volume `full-vol`, show that the file `full.txt` is there.
```bash
sudo ls /var/lib/docker/volumes/full-vol/_data
full.txt
```
- Run the `clarusway/hello-clarus` container with interactive shell open, name the container as `try3`, attach the volume `full-vol` to `/hello-clarus` mount point in the container, and show that we just see the files inside volume regardless of the target directory is full or empty.
```bash
docker run -it --name try3 -v full-vol:/hello-clarus clarusway/hello-clarus sh
/ # ls
bin          etc          home         media        opt          root         sbin         sys          usr
dev          hello-clarus  lib          mnt          proc         run          srv          tmp          var
/ # cd hello-clarus/ && ls
full.txt
```
- `exit` the container
- Remove all volumes and containers and list them.
```bash
docker container prune
docker volume prune
docker volume ls
docker container ls
```

## Part 6 - Bind Mounts

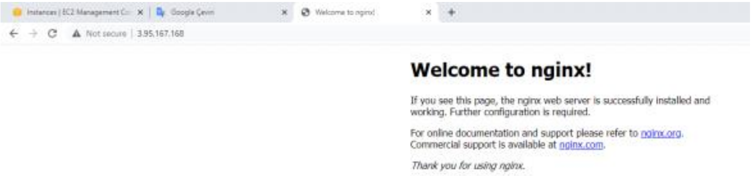
Temel de bir volume Production ortaminda önerilmez Test icin kullanilabiliyor

Lokalde bir folderimizi volume baglamak demektir
Bind Mounts

- Run the `nginx` container at the detached mod, name the container as `nginx-default`, and open <public-ip> on browser and show the nginx default page.
```bash
docker run -d --name nginx-default -p 80:80  nginx  # nginx imajindan bir container olusturuyoruz -d calissin ancak arka planda calissin diyoruz -p hangi portlarda calissin  en son da hangi imajda calisacaksa belirtiyoruz
```
- Add a security rule for protocol HTTP port 80 and show Nginx Web Server is running on Docker Machine.
```text
```

http://<public-ip>:80

Acik olan instancemizin public ip ile :80 portunu yazip calistiriyoruz



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

- Attach the `nginx` container, show the index.html in the /usr/share/nginx/html directory.
```bash
docker exec -it nginx-default bash
root@4a1c7e5f394a:/# cd /usr/share/nginx/html
root@4a1c7e5f394a:/usr/share/nginx/html# ls
50x.html  index.html
root@4a1c7e5f394a:/usr/share/nginx/html# cat index.html
```
- `exit` the container
- Create a folder named webpage, and an index.html file.
```bash
mkdir webpage && cd webpage
echo "<h1>Welcome to Clarusway</h1>" > index.html
```
- Run the `nginx` container at the detached mod, name the container as `nginx-new`, attach the directory `/home/ec2-user/webpage` to `/usr/share/nginx/html` mount point in the container, and open <public-ip> on browser and show the web page.
```bash
docker run -d --name nginx-new -p 8080:80 -v /home/ec2-user/webpage:/usr/share/nginx/html nginx
```
- Add a security rule for protocol HTTP port 8080 and show Nginx Web Server is running on Docker Machine.
```text
http://<public-ip>:8080
```
- Attach the `nginx` container, show the index.html in the /usr/share/nginx/html directory.
```bash
docker exec -it nginx-new bash
root@a7e3d276a147:/# cd usr/share/nginx/html
root@a7e3d276a147:/usr/share/nginx/html# ls
index.html
root@a7e3d276a147:/usr/share/nginx/html# cat index.html
<h1>Welcome to Clarusway</h1>
```
- `exit` the container.
- Add `<h2>
This is added for docker volume lesson</h2>` line to index.html in the /home/ec2-user/webpage folder and check the web page on browser.
```bash
cd /home/ec2-user/webpage
echo "<h2>This is added for docker volume lesson</h2>" >> index.html
```
- Remove the containers.
```bash
docker rm -f nginx-default nginx-new
```
- Remove the volumes.
```bash
 docker volume prune -f
```