Docker

Docker Installation

Docker Installation

Docker Installation in Linux

1.Update the package index

```
sudo yum update -y
```

2.To install docker in linux

```
yum install docker
```

3.Check the status of docker

```
systemctl status docker
```

4.To start the docker

```
systemctl start docker
```

5.Again check the status

```
systemctl status docker
```

8.To verify docker installation

```
docker –version
```

Docker Installation on Ubunutu

1.Update the package index

```
sudo apt-get update -y
```

2.To install docker in Ubunutu

```
apt-get install docker docker-compose -y
```

3.Check the status of docker

```
systemctl status docker
```

4.To start the docker

```
systemctl start docker
```

5.Again check the status

```
systemctl status docker
```

6.To verify docker installation

```
docker –version
```

Docker-Networks

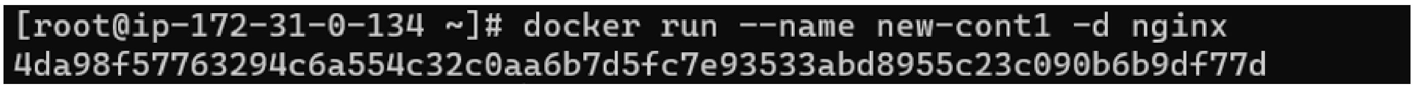
Docker Networks

Easier to set up communication between containers without having to manage IP addresses manually.

1.Default Bridge Network

Docker creates a default bridge network for each host. Containers on the same bridge network can communicate with each other using container names.

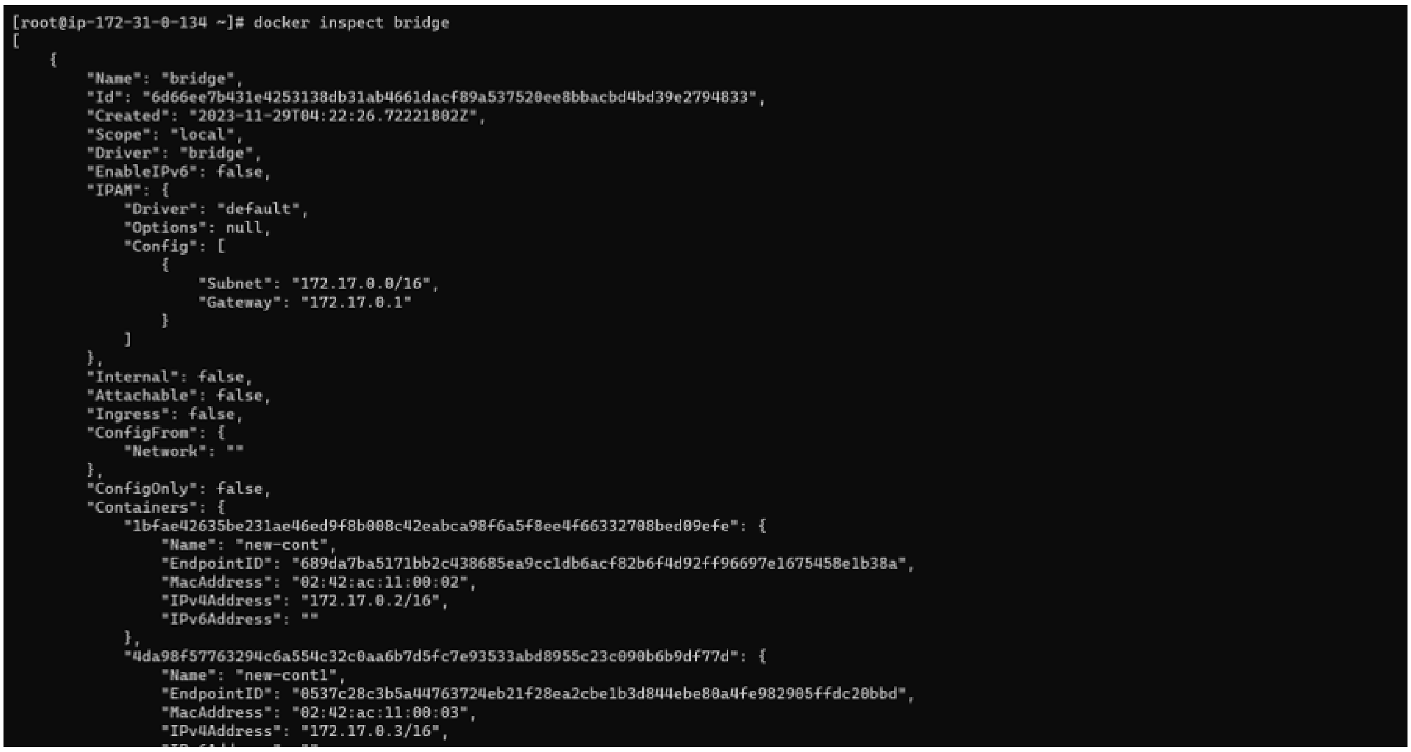
```
docker run --name container1 -d nginx
```



```
[root@ip-172-31-0-134 ~]# docker run --name new-cont1 -d nginx
4da98f57763294c6a554c32c0aa6b7d5fc7e93533abd8955c23c090b6b9df77d
```

To inspect the bridge network, which containers are linked

```
docker inspect bridge
```

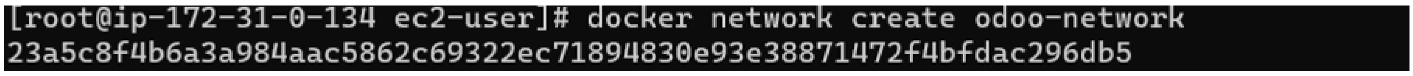


```
[root@ip-172-31-0-134 ~]# docker inspect bridge
[
    {
        "Name": "bridge",
        "Id": "6d66ee7b431e4253138db31ab4661dacf89a537520ee8bbacbd4bd39e2794833",
        "Created": "2023-11-29T04:22:26.72221802Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "1bfae42635be231ae46ed9f8b008c42eabca98f6a5f8ee4f66332708bed09efe": {
                "Name": "new-cont",
                "EndpointID": "689da7ba5171bb2c438685ea9cc1db6acf82b6f4d92ff96697e1675458e1b38a",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            },
            "4da98f57763294c6a554c32c0aa6b7d5fc7e93533abd8955c23c090b6b9df77d": {
                "Name": "new-cont1",
                "EndpointID": "0537c28c3b5a44763724eb21f28ea2cbe1b3d844ebe80a4fe982905ffdc20bbd",
                "MacAddress": "02:42:ac:11:00:03",
                "IPv4Address": "172.17.0.3/16",
```

2.Custom Bridge network

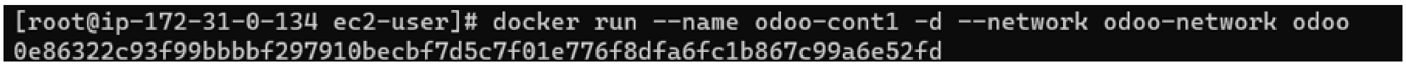
create custom bridge networks to isolate containers and control communication between them.

```
docker network create odoo-network
```



```
[root@ip-172-31-0-134 ec2-user]# docker network create odoo-network
23a5c8f4b6a3a984aac5862c69322ec71894830e93e38871472f4bfdac296db5
```

```
docker run --name container1 -d --network odoo-network odoo
```

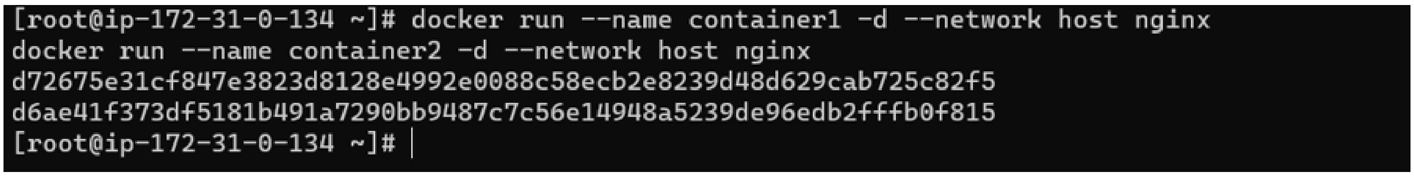


```
[root@ip-172-31-0-134 ec2-user]# docker run --name odoo-cont1 -d --network odoo-network odoo
0e86322c93f99bbbbf297910becbf7d5c7f01e776f8dfa6fc1b867c99a6e52fd
```

3.Host Network:

Containers share the host network stack, meaning they can communicate directly using localhost.

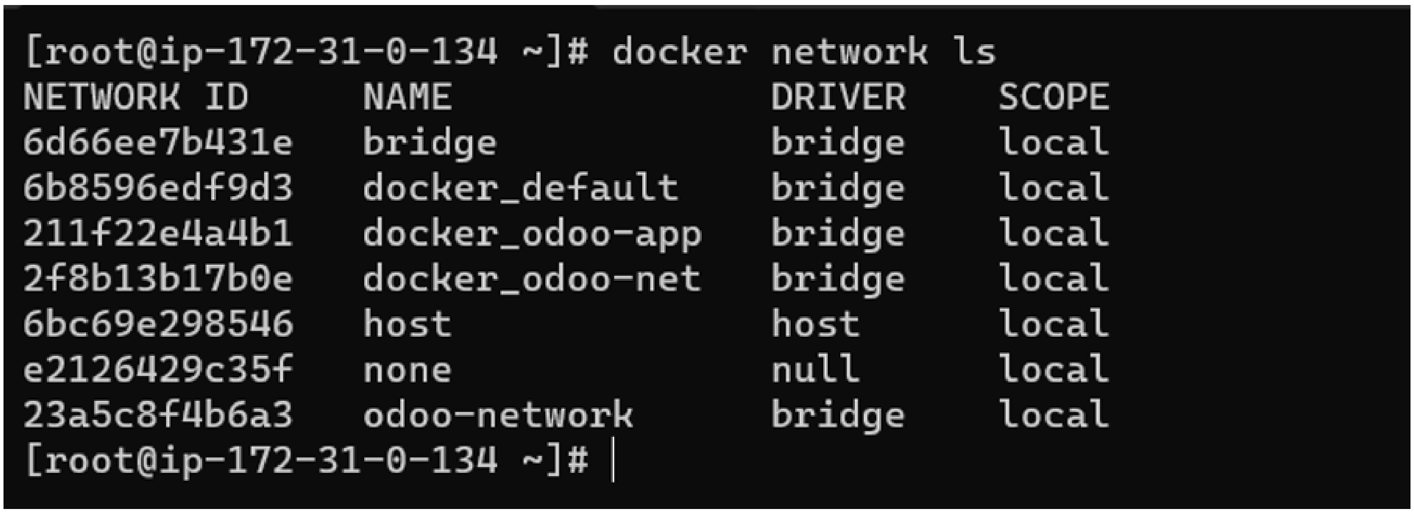
```
docker run --name container1 -d --network host nginx

docker run --name container2 -d --network host nginx
```



```
[root@ip-172-31-0-134 ~]# docker run --name container1 -d --network host nginx
docker run --name container2 -d --network host nginx
d72675e31cf847e3823d8128e4992e0088c58ecb2e8239d48d629cab725c82f5
d6ae41f373df5181b491a7290bb9487c7c56e14948a5239de96edb2fffb0f815
[root@ip-172-31-0-134 ~]#
```

4.Check the Created Networks

```
docker network ls
```



```
[root@ip-172-31-0-134 ~]# docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
6d66ee7b431e   bridge            bridge    local
6b8596edf9d3   docker_default    bridge    local
211f22e4a4b1   docker_odoo-app   bridge    local
2f8b13b17b0e   docker_odoo-net   bridge    local
6bc69e298546   host              host      local
e2126429c35f   none              null      local
23a5c8f4b6a3   odoo-network      bridge    local
[root@ip-172-31-0-134 ~]#
```

Docker-Volumes

Docker Volumes

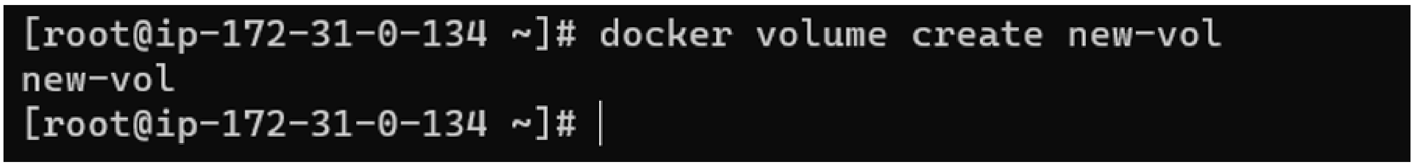
Volumes allow you to persist data generated or modified by containers.

Multiple containers can use the same volume, making it easier to design modular and scalable applications

1.Create a Volume

You can create a volume using the docker volume create command.

```
docker volume create my_volume
```

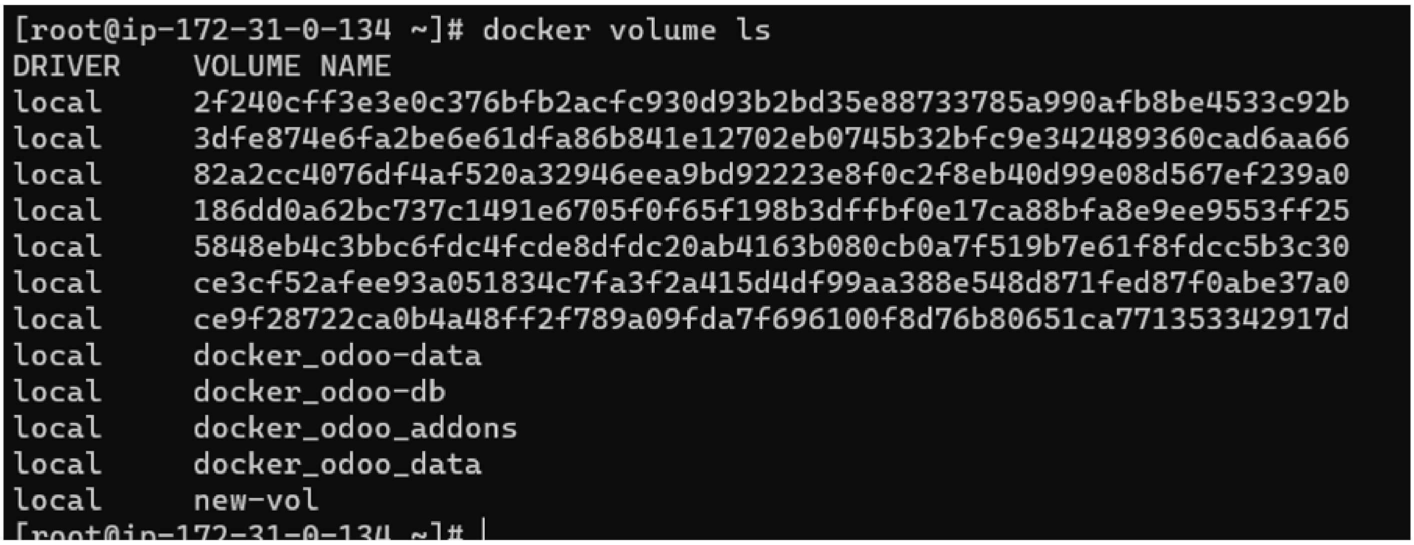


```
[root@ip-172-31-0-134 ~]# docker volume create new-vol
new-vol
[root@ip-172-31-0-134 ~]#
```

2.Check the Created Volume

Verify that the volume has been created by running.

```
docker volume ls
```

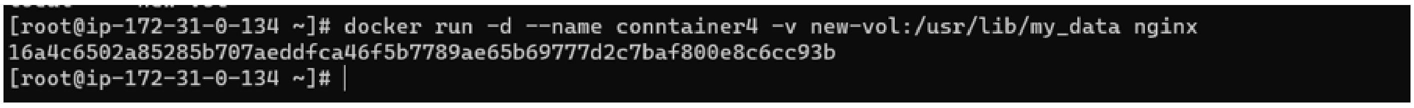


```
[root@ip-172-31-0-134 ~]# docker volume ls
DRIVER    VOLUME NAME
local     2f240cff3e3e0c376bfb2acfc930d93b2bd35e88733785a990afb8be4533c92b
local     3dfe874e6fa2be6e61dfa86b841e12702eb0745b32bfc9e342489360cad6aa66
local     82a2cc4076df4af520a32946eea9bd92223e8f0c2f8eb40d99e08d567ef239a0
local     186dd0a62bc737c1491e6705f0f65f198b3dffbf0e17ca88bfa8e9ee9553ff25
local     5848eb4c3bbc6fdc4fcde8dfdc20ab4163b080cb0a7f519b7e61f8fdcc5b3c30
local     ce3cf52afee93a051834c7fa3f2a415d4df99aa388e548d871fed87f0abe37a0
local     ce9f28722ca0b4a48ff2f789a09fda7f696100f8d76b80651ca771353342917d
local     docker_odoo-data
local     docker_odoo-db
local     docker_odoo_addons
local     docker_odoo_data
local     new-vol
[root@ip-172-31-0-134 ~]#
```

3.Run a Container with a Volume

You can use the -v or --volume option to mount a volume when running a container

```
docker run -d --name conntainer4 -v /path/your/vol: /usr/lib/my_data nginx
```

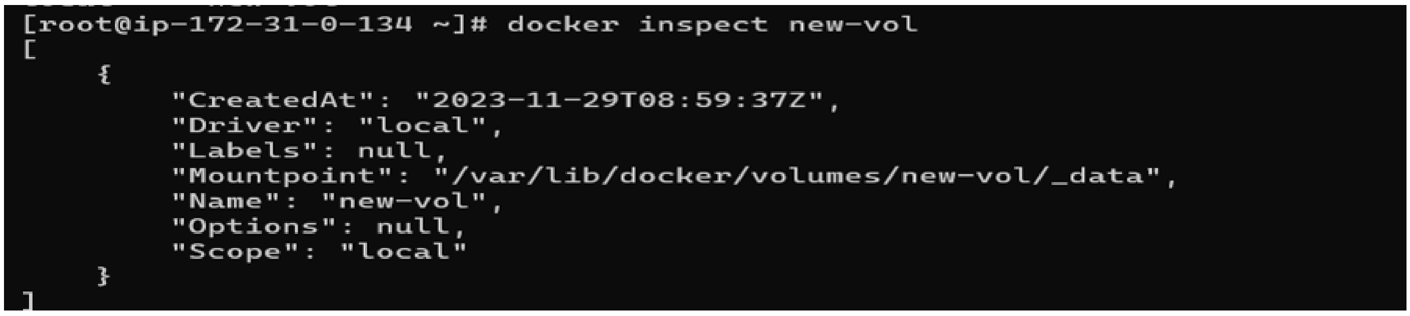


```
[root@ip-172-31-0-134 ~]# docker run -d --name conntainer4 -v new-vol:/usr/lib/my_data nginx
16a4c6502a85285b707aeddfca46f5b7789ae65b69777d2c7baf800e8c6cc93b
[root@ip-172-31-0-134 ~]#
```

4.View Volume Mounts:

To see the volumes mounted in a running container

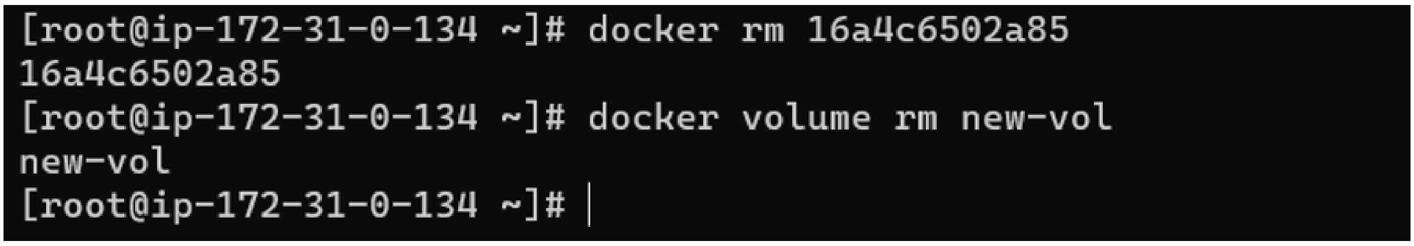
```
docker inspect conntainer4
```



```
[root@ip-172-31-0-134 ~]# docker inspect new-vol
[
    {
        "CreatedAt": "2023-11-29T08:59:37Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/new-vol/_data",
        "Name": "new-vol",
        "Options": null,
        "Scope": "local"
    }
]
```

5.To remove a volume

```
docker volume rm my_volume
```

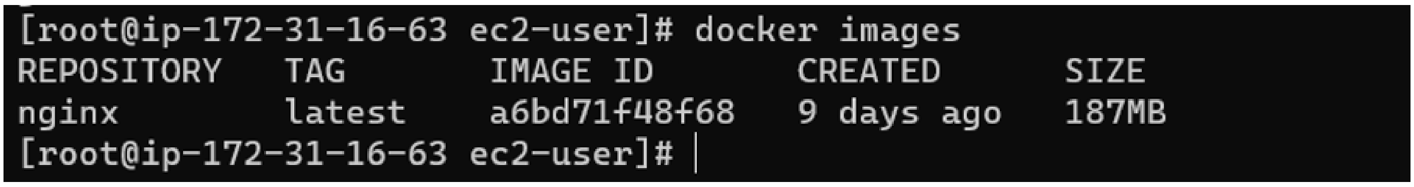


```
[root@ip-172-31-0-134 ~]# docker rm 16a4c6502a85
16a4c6502a85
[root@ip-172-31-0-134 ~]# docker volume rm new-vol
new-vol
[root@ip-172-31-0-134 ~]#
```

Create a soft link for container using volumes

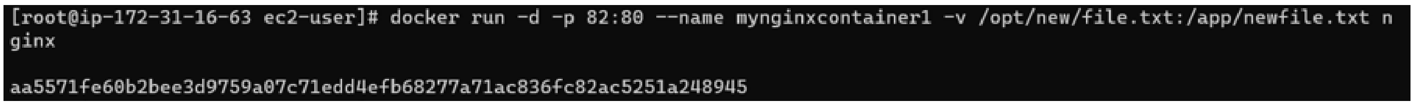
1.To download nginx docker image

```
docker images
```


```
[root@ip-172-31-16-63 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
nginx        latest    a6bd71f48f68   9 days ago   187MB
[root@ip-172-31-16-63 ec2-user]# 
```

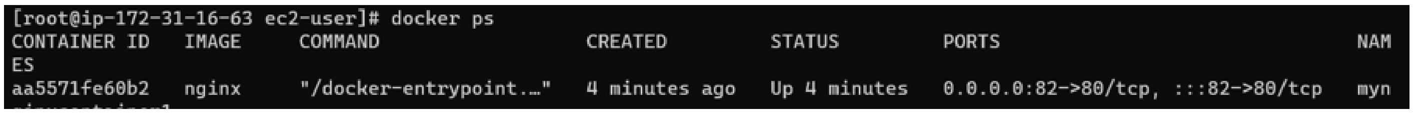
2.To run the container using volume to add which path will link

```
docker run -d -p 82:80 --name mynginxcontainer1 -v /opt/new/file.txt:/app/newfile.txt nginx
```


```
[root@ip-172-31-16-63 ec2-user]# docker run -d -p 82:80 --name mynginxcontainer1 -v /opt/new/file.txt:/app/newfile.txt n
ginx

aa5571fe60b2bee3d9759a07c71edd4efb68277a71ac836fc82ac5251a248945
```

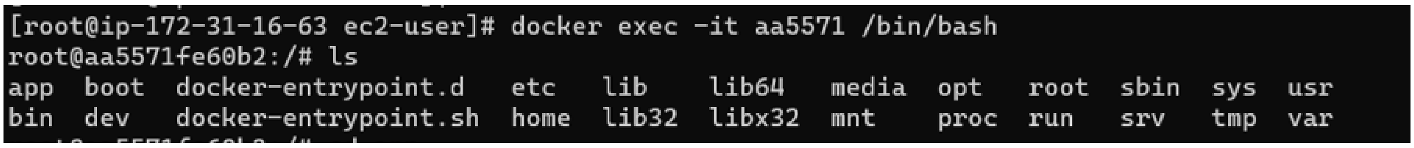
3.To check the process

```
docker ps
```


```
[root@ip-172-31-16-63 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                               NAM
ES
aa5571fe60b2   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   0.0.0.0:82->80/tcp, :::82->80/tcp   myn
```

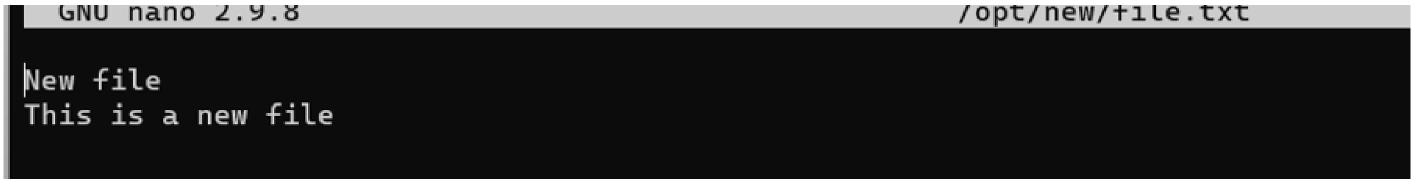
4.Now login the container

```
docker exec -it aa5571 /bin/bash
```


```
[root@ip-172-31-16-63 ec2-user]# docker exec -it aa5571 /bin/bash
root@aa5571fe60b2:/# ls
app  boot  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin  sys  usr
bin  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv   tmp  var
```

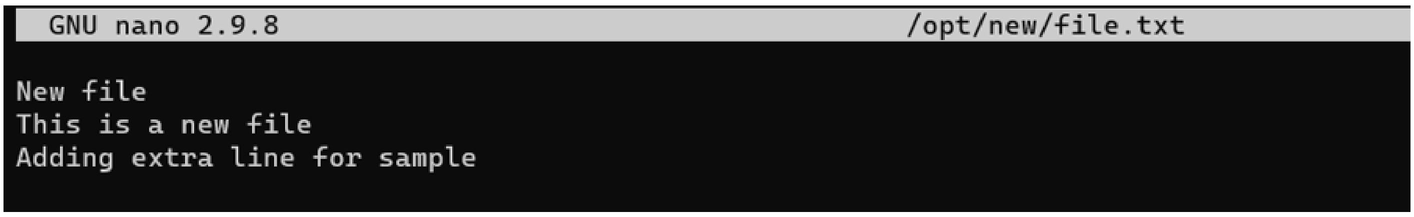
5.Now check the file in container

```
nano /app/newfile.txt
```


```
  GNU nano 2.9.8                              /opt/new/file.txt

New file
This is a new file
```

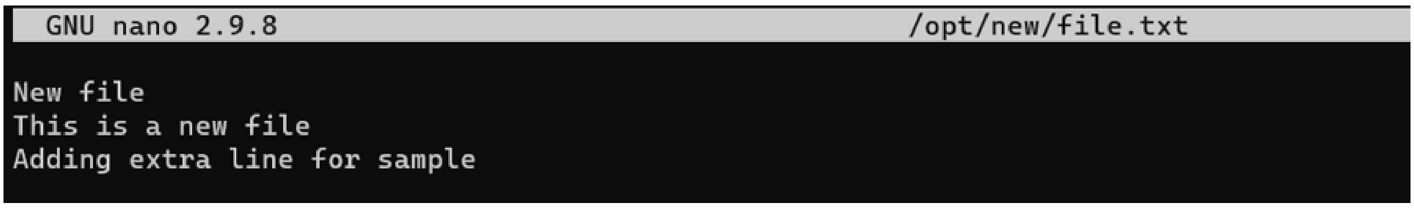
6.Check the file in local machine

```
nano /opt/new/file.txt
```



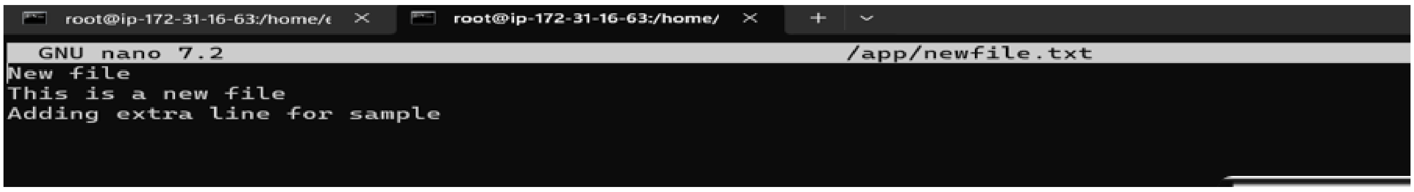
```
  GNU nano 2.9.8                                    /opt/new/file.txt

New file
This is a new file
Adding extra line for sample
```

7.Now changes in local file



```
  GNU nano 2.9.8                                    /opt/new/file.txt

New file
This is a new file
Adding extra line for sample
```

8.Check the container



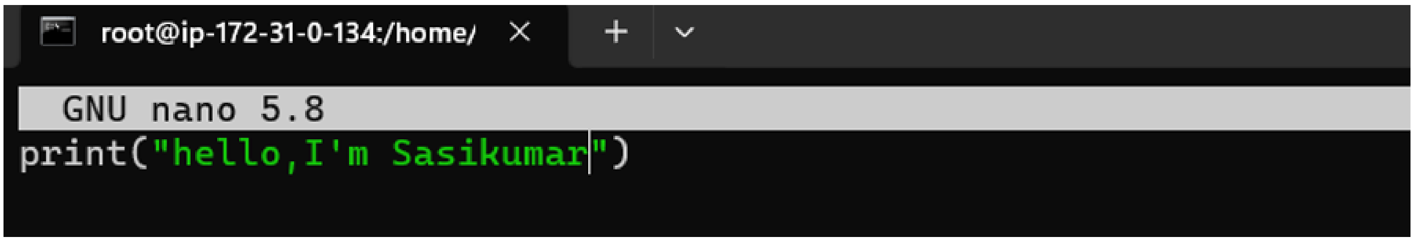
```
  GNU nano 7.2                                      /app/newfile.txt
New file
This is a new file
Adding extra line for sample
```

Docker-files

Docker file

1.Create a project with sample python file

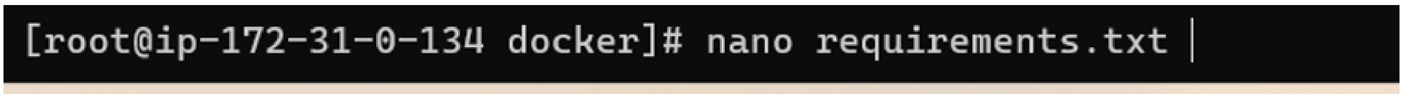
```
nano app.py
print("Hello, World")
```



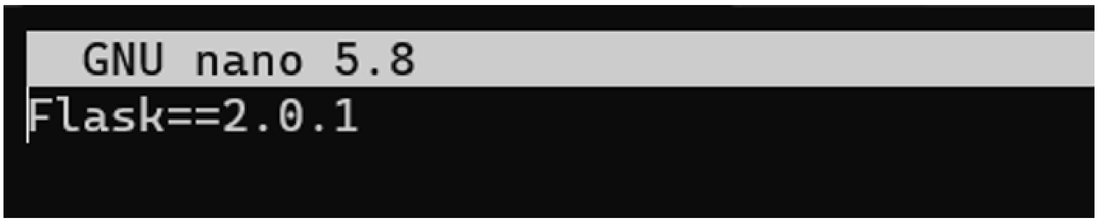
```
root@ip-172-31-0-134:/home/
GNU nano 5.8
print("hello,I'm Sasikumar")
```

2.Create a sample requirements.txt

```
nano requirements.txt
Flask==2.0.1
```



```
[root@ip-172-31-0-134 docker]# nano requirements.txt
```



```
GNU nano 5.8
Flask==2.0.1
```

3.Now create a docker file

```
nano Dockerfile
```

Add the below steps from 4 to 8

4.Start with a Base Image

```
FROM ubuntu:20.04
```

5.Set the Working Directory

Specify a working directory inside the container where your application will be placed

```
WORKDIR /app
```

6.Copy Files

Copy the necessary files from your local machine to the container.

```
COPY . /app
```

7.Run Commands:

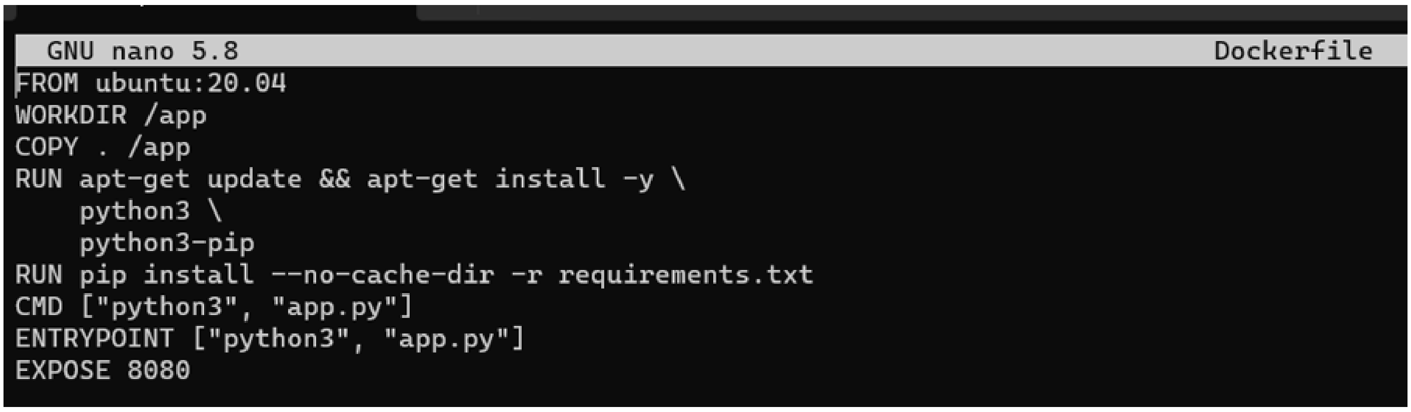
Specify the command to run when the container starts. This is often the command to start your application.

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

8.Run Commands:

Specify the command to run when the container starts. This is often the command to start your application.

```
EXPOSE 8080
```

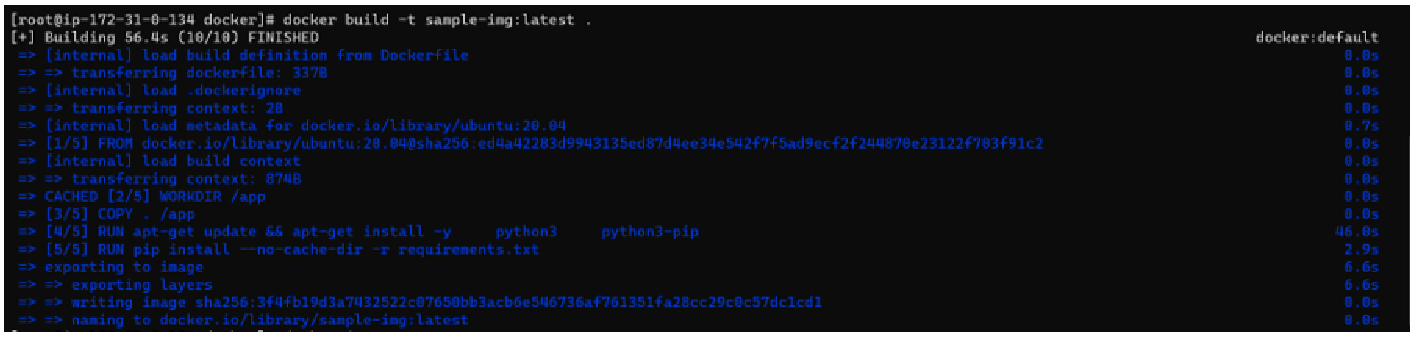


```
  GNU nano 5.8                                                          Dockerfile
FROM ubuntu:20.04
WORKDIR /app
COPY . /app
RUN apt-get update && apt-get install -y \
    python3 \
    python3-pip
RUN pip install --no-cache-dir -r requirements.txt
CMD ["python3", "app.py"]
ENTRYPOINT ["python3", "app.py"]
EXPOSE 8080
```

9.Build the Docker Image:

Save the Dockerfile in your project directory and run the following command in the same directory to build the Docker image.

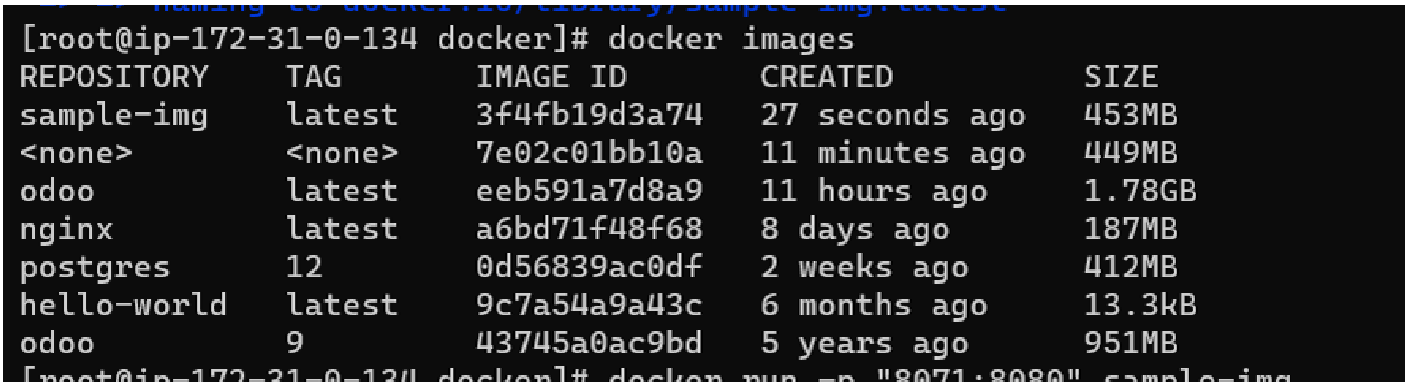
```
docker build -t sample-img:latest .
```



```
[root@ip-172-31-0-134 docker]# docker build -t sample-img:latest .
[+] Building 56.4s (10/10) FINISHED                                                                                           docker:default
 => [internal] load build definition from Dockerfile                                                                                   0.0s
 => => transferring dockerfile: 337B                                                                                                   0.0s
 => [internal] load .dockerignore                                                                                                      0.0s
 => => transferring context: 2B                                                                                                        0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                                                        0.7s
 => [1/5] FROM docker.io/library/ubuntu:20.04@sha256:ed4a42283d9943135ed87d4ee34e542f7f5ad9ecf2f244870e23122f703f91c2                  0.0s
 => [internal] load build context                                                                                                      0.0s
 => => transferring context: 874B                                                                                                      0.0s
 => CACHED [2/5] WORKDIR /app                                                                                                          0.0s
 => [3/5] COPY . /app                                                                                                                  0.0s
 => [4/5] RUN apt-get update && apt-get install -y     python3     python3-pip                                                        46.0s
 => [5/5] RUN pip install --no-cache-dir -r requirements.txt                                                                           2.9s
 => exporting to image                                                                                                                 6.6s
 => => exporting layers                                                                                                                6.6s
 => => writing image sha256:3f4fb19d3a7432522c07650bb3acb6e546736af761351fa28cc29c0c57dc1cd1                                           0.0s
 => => naming to docker.io/library/sample-img:latest                                                                                   0.0s
```

10.Check the image

```
docker images
```

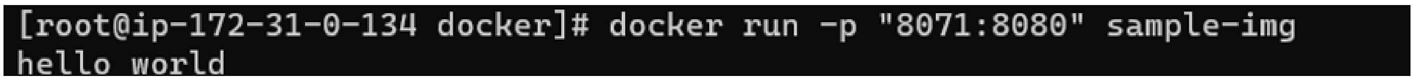


```
[root@ip-172-31-0-134 docker]# docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
sample-img     latest    3f4fb19d3a74   27 seconds ago   453MB
<none>         <none>    7e02c01bb10a   11 minutes ago   449MB
odoo           latest    eeb591a7d8a9   11 hours ago     1.78GB
nginx          latest    a6bd71f48f68   8 days ago       187MB
postgres       12        0d56839ac0df   2 weeks ago      412MB
hello-world    latest    9c7a54a9a43c   6 months ago     13.3kB
odoo           9         43745a0ac9bd   5 years ago      951MB
```

11.Run the Docker Container

Once the image is built, you can run a container based on that image.

```
docker run -p 8071:8080 sample-img:latest
```



```
[root@ip-172-31-0-134 docker]# docker run -p "8071:8080" sample-img
hello world
```

Docker-Compose

Docker Compose

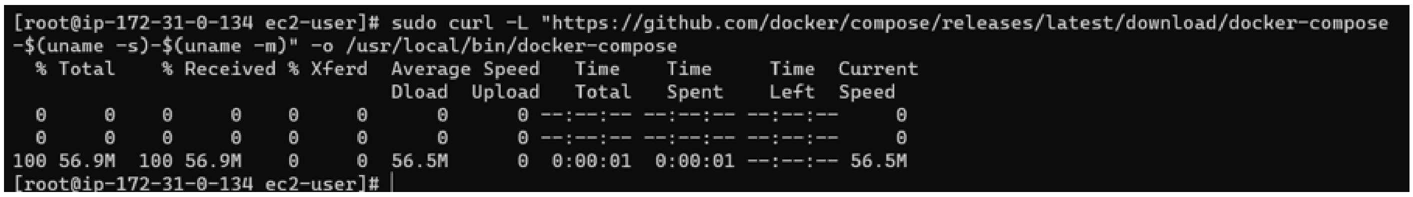
Docker Compose is a tool for defining and running multi-container Docker applications.

Easy to manage and deploy complex applications composed of multiple containers.

Docker-compose installation on linux

Go to web browser and check the docker compose website and below the download command to copy the linux

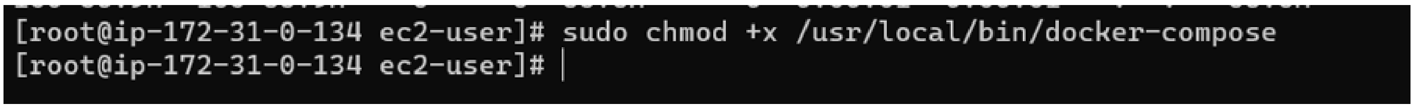
```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```



```
[root@ip-172-31-0-134 ec2-user]# sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose
-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 56.9M  100 56.9M    0     0  56.5M      0  0:00:01  0:00:01 --:--:-- 56.5M
[root@ip-172-31-0-134 ec2-user]# 
```

2.After downloading the docker-compose and make this are executable

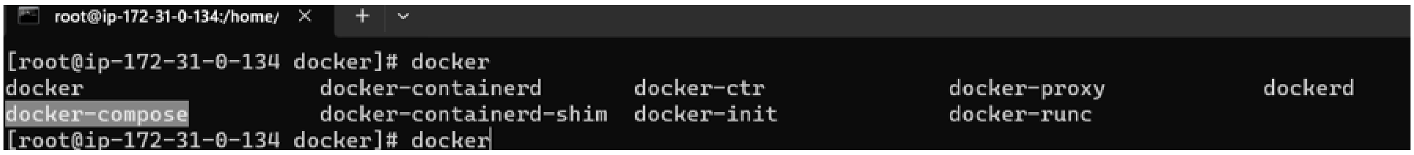
```
sudo chmod +x /usr/local/bin/docker-compose
```



```
[root@ip-172-31-0-134 ec2-user]# sudo chmod +x /usr/local/bin/docker-compose
[root@ip-172-31-0-134 ec2-user]# 
```

3.Check the docker-compose is installed

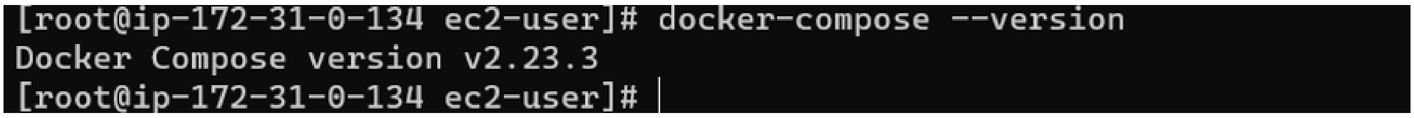
```
docker (double tab)
```



```
[root@ip-172-31-0-134 docker]# docker
docker                docker-containerd      docker-ctr            docker-proxy          dockerd
docker-compose        docker-containerd-shim docker-init           docker-runc
[root@ip-172-31-0-134 docker]# docker
```

4.To check the docker-compose version

```
docker-compose --version
```



```
[root@ip-172-31-0-134 ec2-user]# docker-compose --version
Docker Compose version v2.23.3
[root@ip-172-31-0-134 ec2-user]#
```

5.Docker-compose Commands:

Command Description

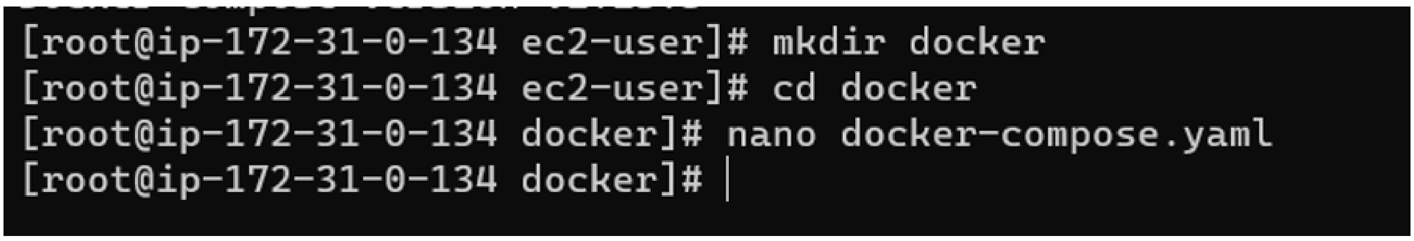
```
build	Builds or rebuilds services
config	Parses, resolves, and renders compose file in canonical format
cp	Copies files/folders between a service container and the local filesystem
create	Creates containers for a service
down	Stops and removes containers, networks
events	Receives real-time events from containers
exec	Executes a command in a running container
images	Lists images used by the created containers
kill	Force-stops service containers
logs	Views output from containers
ls	Lists running compose projects
pause	Pauses services
port	Prints the public port for a port binding
ps	Lists containers
pull	Pulls service images
push	Pushes service images
restart	Restarts service containers
rm	Removes stopped service containers
run	Runs a one-off command on a service
scale	Scales services
start	Starts services
stop	Stops services
top	Displays the running processes
unpause	Unpauses services
up	Creates and starts containers
version	Shows the Docker Compose version information
wait	Blocks until the first service container stops
watch	Watches build context for service and rebuild/refresh containers when files are updated
```

```
```

Docker compose file Creation

1.Create a docker-compose yaml file odoo setup

```
nano docker-compose.yaml
```



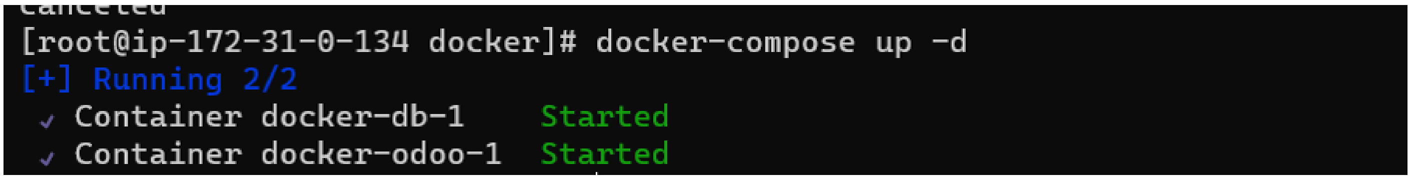
```
[root@ip-172-31-0-134 ec2-user]# mkdir docker
[root@ip-172-31-0-134 ec2-user]# cd docker
[root@ip-172-31-0-134 docker]# nano docker-compose.yaml
[root@ip-172-31-0-134 docker]#
```

2.Inside yaml file to add this content

```
version: '3'
services:
  odoo:
    image: odoo:9
    ports:
      - "8070:8069"
    environment:
      - POSTGRES_USER=test
      - POSTGRES_PASSWORD=test
      - PGDATA=/var/lib/postgresql/data/pgdata
    volumes:
      - odoo-data:/var/lib/odoo
    depends_on:
      - db
    networks:
      - odoo-app

  db:
    image: postgres:12
    environment:
      - POSTGRES_USER=test
      - POSTGRES_PASSWORD=test
      - POSTGRES_DB=postgres
    volumes:
      - odoo-db:/var/lib/postgresql/data
    networks:
      - odoo-app

networks:
  odoo-app:
    driver: bridge

volumes:
  odoo-data:
  odoo-db:
```

3.To up the docker-compose

```
docker-compose up -d
```



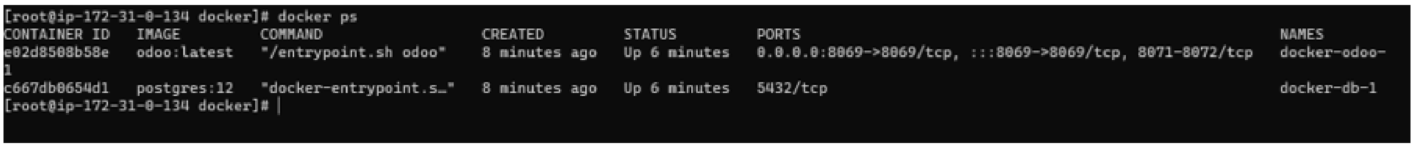
```
[root@ip-172-31-0-134 docker]# docker-compose up -d
[+] Running 2/2
 ✓ Container docker-db-1     Started
 ✓ Container docker-odoo-1   Started
```

If create a file for another name, it should do command

```
docker-compose -f file.yaml up -d
```

4.Check the process will run and check ports

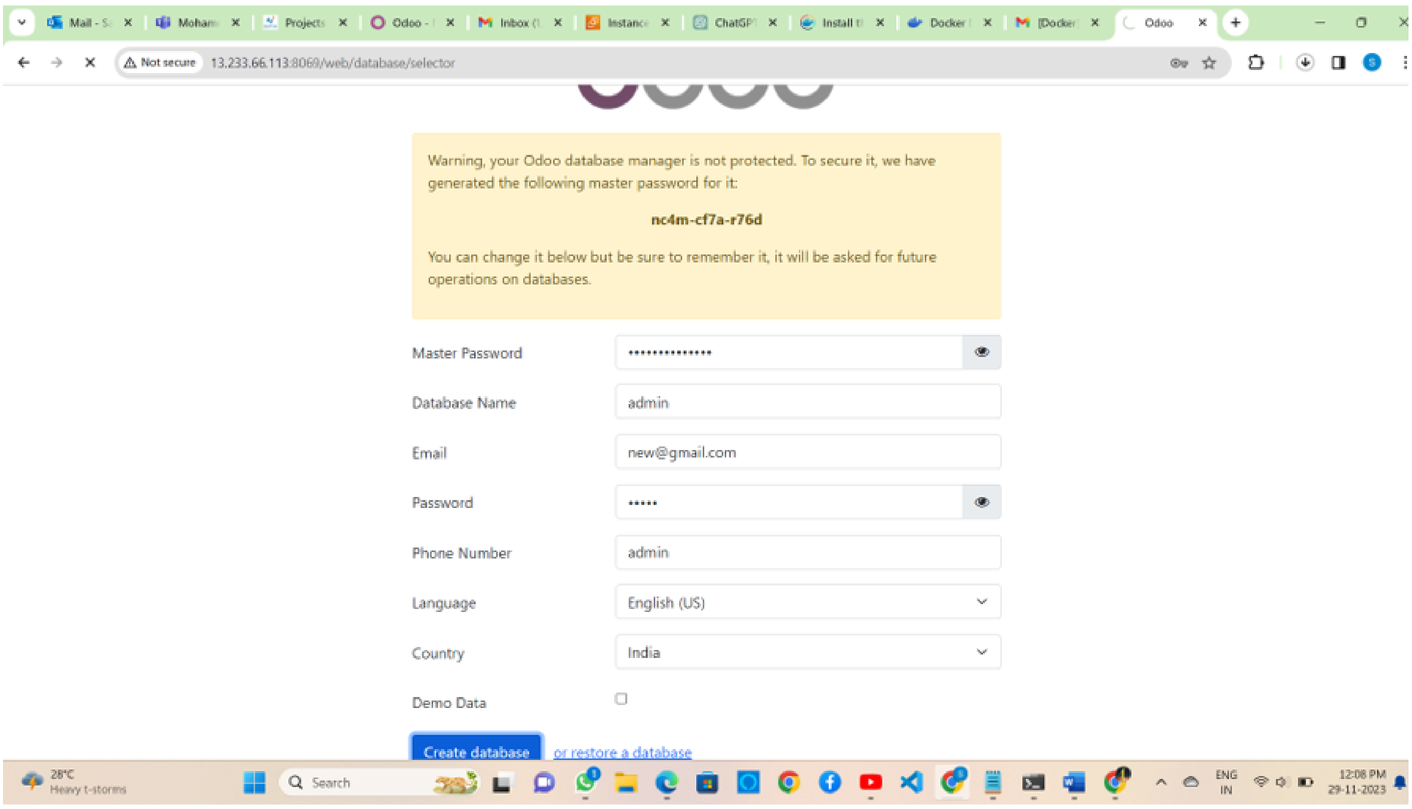
```
docker ps
```



```
[root@ip-172-31-0-134 docker]# docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                                                  NAMES
e02d8508b58e   odoo:latest   "/entrypoint.sh odoo"    8 minutes ago   Up 6 minutes   0.0.0.0:8069->8069/tcp, :::8069->8069/tcp, 8071-8072/tcp   docker-odoo-
1
c667db0654d1   postgres:12   "docker-entrypoint.s…"   8 minutes ago   Up 6 minutes   5432/tcp                                               docker-db-1
[root@ip-172-31-0-134 docker]# |
```

5.To test the process in browser

```
http://localhost:8069/
```


13.233.66.113:8069/web/database/selector

Warning, your Odoo database manager is not protected. To secure it, we have generated the following master password for it:

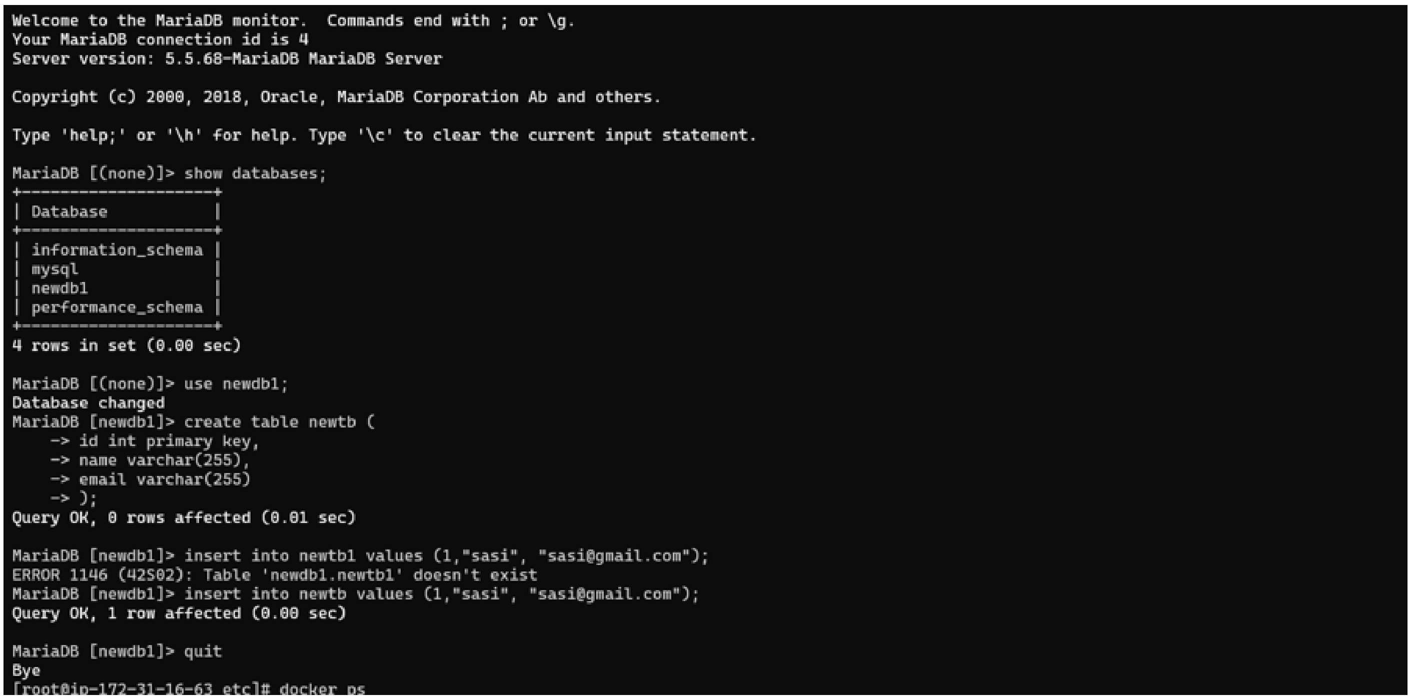
nc4m-cf7a-r76d

You can change it below but be sure to remember it, it will be asked for future operations on databases.

Master Password	••••••••••••••
Database Name	admin
Email	new@gmail.com
Password	•••••
Phone Number	admin
Language	English (US)
Country	India
Demo Data	☐

Attach local DB to Container

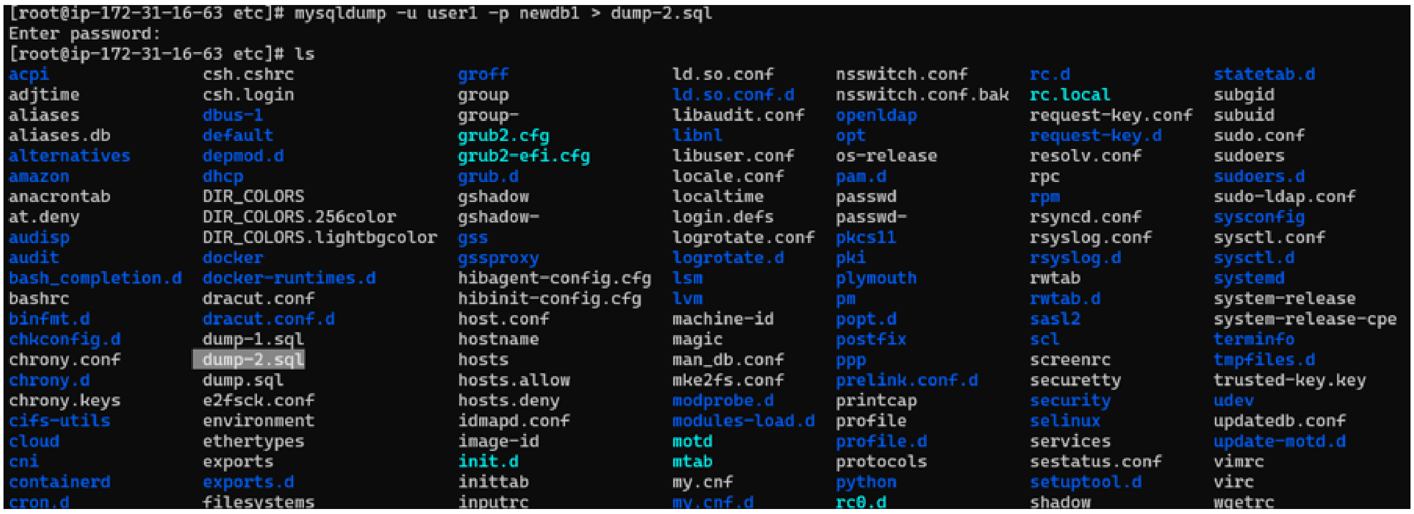
To attach database to docker container 1.Create a database in local machine



```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| newdb1             |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]> use newdb1;
Database changed
MariaDB [newdb1]> create table newtb (
    -> id int primary key,
    -> name varchar(255),
    -> email varchar(255)
    -> );
Query OK, 0 rows affected (0.01 sec)

MariaDB [newdb1]> insert into newtbl values (1,"sasi", "sasi@gmail.com");
ERROR 1146 (42S02): Table 'newdb1.newtbl' doesn't exist
MariaDB [newdb1]> insert into newtb values (1,"sasi", "sasi@gmail.com");
Query OK, 1 row affected (0.00 sec)

MariaDB [newdb1]> quit
Bye
[root@ip-172-31-16-63 etc]# docker ps
```

2.To dump the database

```
mysqldump -u user1 -p newdb1 > dump-2.sql
```



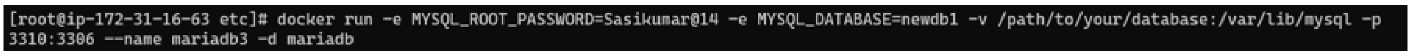
```
[root@ip-172-31-16-63 etc]# mysqldump -u user1 -p newdb1 > dump-2.sql
Enter password:
[root@ip-172-31-16-63 etc]# ls
acpi                csh.cshrc                 groff                ld.so.conf      nsswitch.conf      rc.d              statetab.d
adjtime             csh.login                 group                ld.so.conf.d    nsswitch.conf.bak  rc.local          subgid
aliases             dbus-1                    group-               libaudit.conf   openldap           request-key.conf  subuid
aliases.db          default                   grub2.cfg            libnl           opt                request-key.d     sudo.conf
alternatives        depmod.d                  grub2-efi.cfg        libuser.conf    os-release         resolv.conf       sudoers
amazon              dhcp                      grub.d               locale.conf     pam.d              rpc               sudoers.d
anacrontab          DIR_COLORS                gshadow              localtime       passwd             rpm               sudo-ldap.conf
at.deny             DIR_COLORS.256color       gshadow-             login.defs      passwd-            rsyncd.conf       sysconfig
audisp              DIR_COLORS.lightbgcolor   gss                  logrotate.conf  pkcs11             rsyslog.conf      sysctl.conf
audit               docker                    gssproxy             logrotate.d     pki                rsyslog.d         sysctl.d
bash_completion.d   docker-runtimes.d         hibagent-config.cfg  lsm             plymouth           rwtab             systemd
bashrc              dracut.conf               hibinit-config.cfg   lvm             pm                 rwtab.d           system-release
binfmt.d            dracut.conf.d             host.conf            machine-id      popt.d             sasl2             system-release-cpe
chkconfig.d         dump-1.sql                hostname             magic           postfix            scl               terminfo
chrony.conf         dump-2.sql                hosts                man_db.conf     ppp                screenrc          tmpfiles.d
chrony.d            dump.sql                  hosts.allow          mke2fs.conf     prelink.conf.d     securetty         trusted-key.key
chrony.keys         e2fsck.conf               hosts.deny           modprobe.d      printcap           security          udev
cifs-utils          environment               idmapd.conf          modules-load.d  profile            selinux           updatedb.conf
cloud               ethertypes                image-id             motd            profile.d          services          update-motd.d
cni                 exports                   init.d               mtab            protocols          sestatus.conf     vimrc
containerd          exports.d                 inittab              my.cnf          python             setuptool.d       virc
cron.d              filesystems               inputrc              my.cnf.d        rc0.d              shadow            wgetrc
```

To restore the database

```
mysql -P 3306 -u root -p newdb1 < dump-2.sql
```

3.Now create a docker container using mariadb

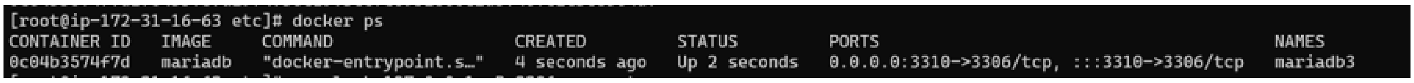
```
docker run -e MYSQL_ROOT_PASSWORD=your_mysql_passwd -e MYSQL_DATABASE=newdb1 -v
/path/to/your/database:/var/lib/mysql -p 3310:3306 --name mariadb3 -d mariadb
```



```
[root@ip-172-31-16-63 etc]# docker run -e MYSQL_ROOT_PASSWORD=Sasikumar@14 -e MYSQL_DATABASE=newdb1 -v /path/to/your/database:/var/lib/mysql -p
3310:3306 --name mariadb3 -d mariadb
```

4.To check the docker process

```
docker ps
```



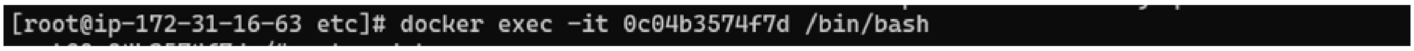
```
[root@ip-172-31-16-63 etc]# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                       NAMES
0c04b3574f7d   mariadb   "docker-entrypoint.s…"   4 seconds ago   Up 2 seconds   0.0.0.0:3310->3306/tcp, :::3310->3306/tcp   mariadb3
```

5.Login the docker container

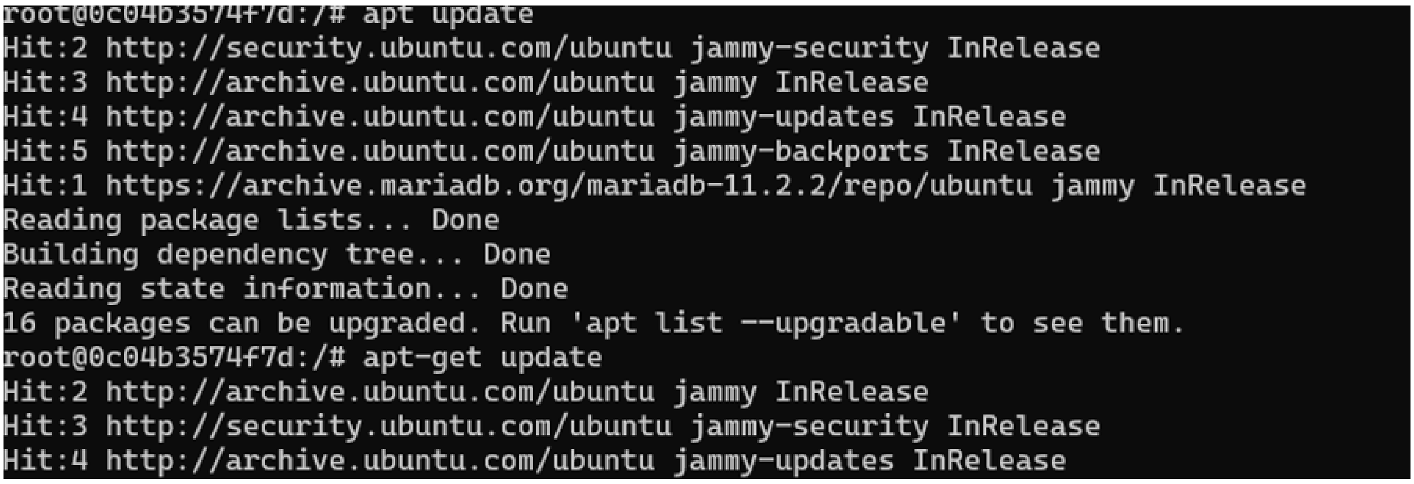
```
docker exec -it 0c04b3574f7d /bin/bash
```

6.To update the container

```
apt update
```



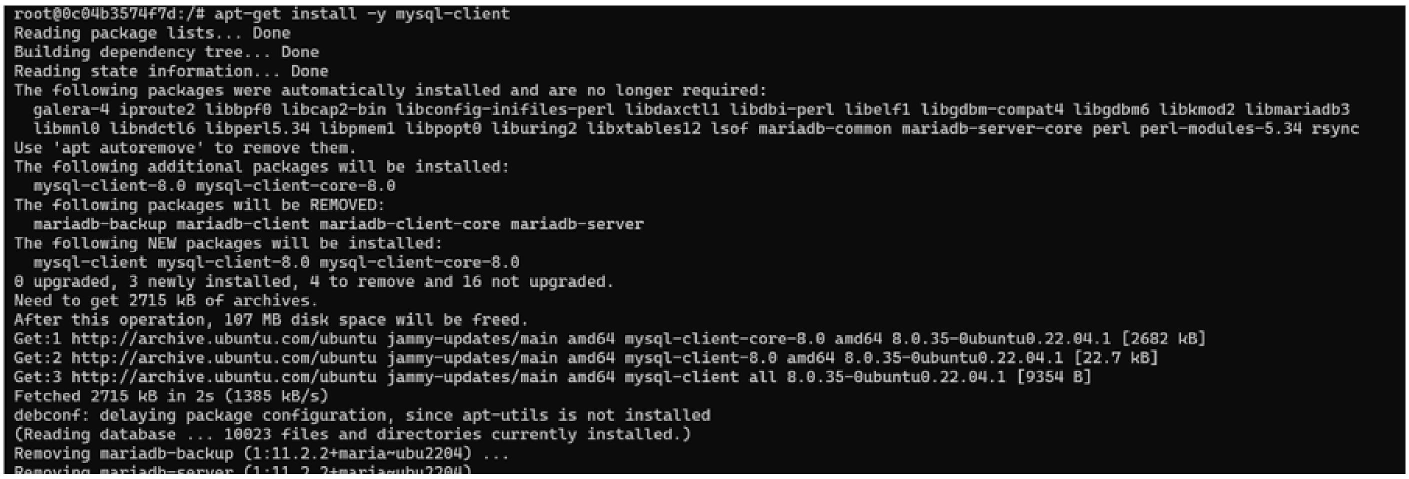
```
[root@ip-172-31-16-63 etc]# docker exec -it 0c04b3574f7d /bin/bash
```



```
root@0c04b3574f7d:/# apt update
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:1 https://archive.mariadb.org/mariadb-11.2.2/repo/ubuntu jammy InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
16 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@0c04b3574f7d:/# apt-get update
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
```

7.To install mysql in container

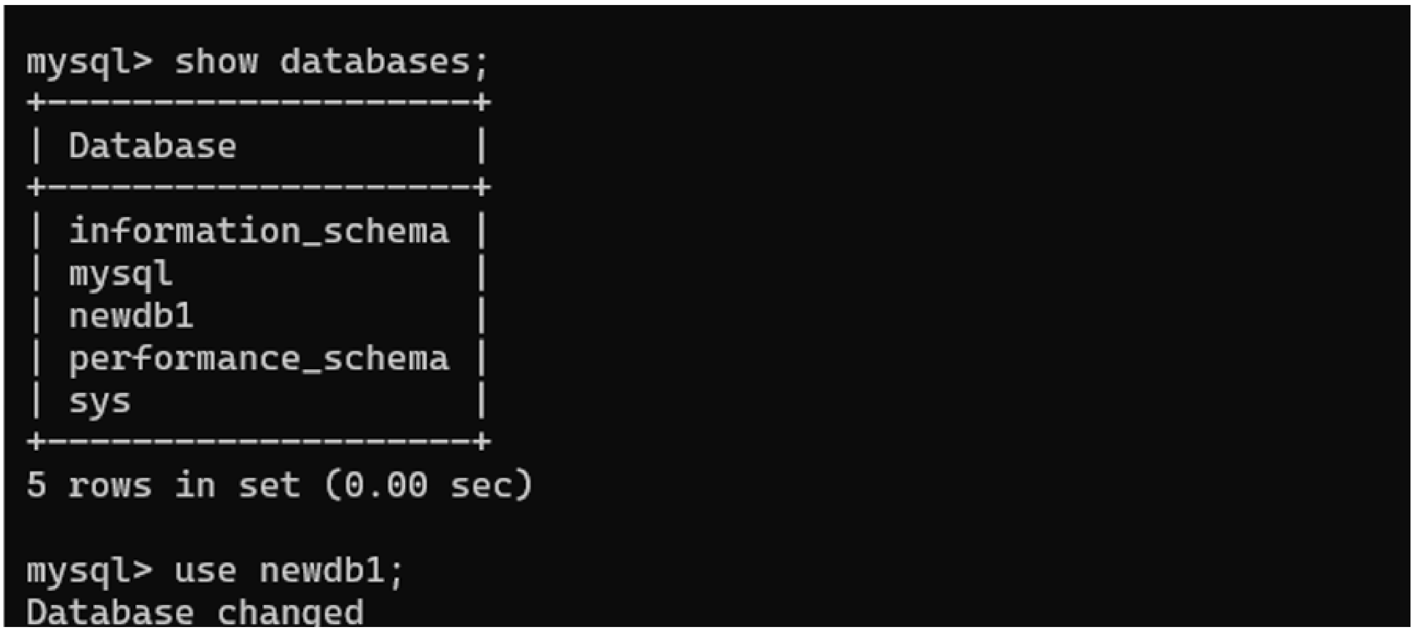
```
apt-get install -y mysql-client
```



```
root@0c04b3574f7d:/# apt-get install -y mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  galera-4 iproute2 libbpf0 libcap2-bin libconfig-inifiles-perl libdaxctl1 libdbi-perl libelf1 libgdbm-compat4 libgdbm6 libkmod2 libmariadb3
  libmnl0 libndctl6 libperl5.34 libpmem1 libpopt0 liburing2 libxtables12 lsof mariadb-common mariadb-server-core perl perl-modules-5.34 rsync
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0
The following packages will be REMOVED:
  mariadb-backup mariadb-client mariadb-client-core mariadb-server
The following NEW packages will be installed:
  mysql-client mysql-client-8.0 mysql-client-core-8.0
0 upgraded, 3 newly installed, 4 to remove and 16 not upgraded.
Need to get 2715 kB of archives.
After this operation, 107 MB disk space will be freed.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-core-8.0 amd64 8.0.35-0ubuntu0.22.04.1 [2682 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client-8.0 amd64 8.0.35-0ubuntu0.22.04.1 [22.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 mysql-client all 8.0.35-0ubuntu0.22.04.1 [9354 B]
Fetched 2715 kB in 2s (1385 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 10023 files and directories currently installed.)
Removing mariadb-backup (1:11.2.2+maria~ubu2204) ...
Removing mariadb-server (1:11.2.2+maria~ubu2204)
```

8.Login as mysql

9.Check the in container



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| newdb1             |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use newdb1;
Database changed
```

Attaching two docker containers

Attaching two docker containers

Docker Compose to create two containers that communicate with each other over a network.

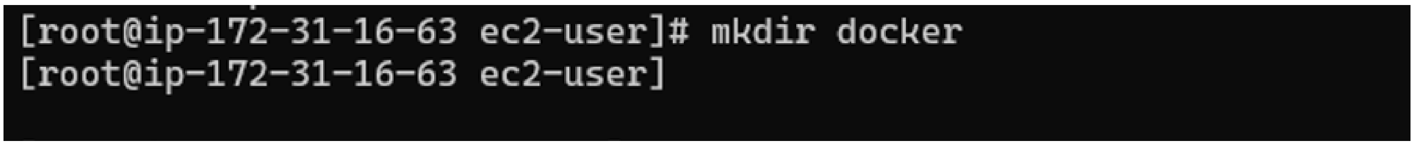
For example:

Container1: Simple flask container

Container2: Nginx

1.Create a new Directory

```
mkdir docker
cd docker
```



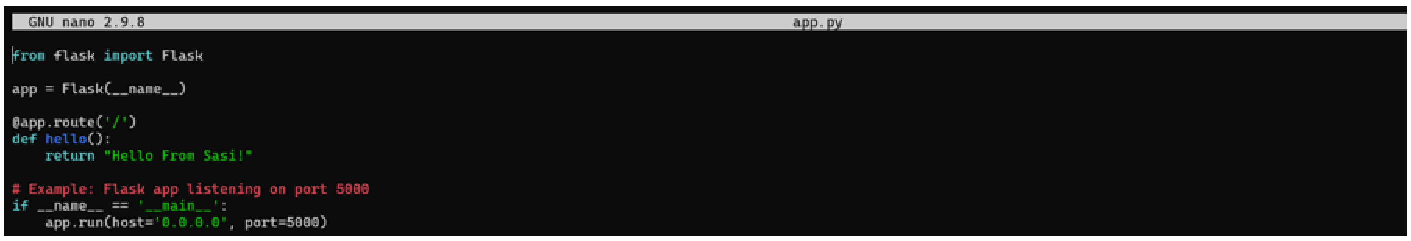
```
[root@ip-172-31-16-63 ec2-user]# mkdir docker
[root@ip-172-31-16-63 ec2-user]
```

2.Now create a Sample python file

```
nano appy.py
```

```
from flask import Flask


app = Flask(__name__)


@app.route("/")
def hello():
    return "Hello From Sasil"


# Example: Flask app listening on port 5008
```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

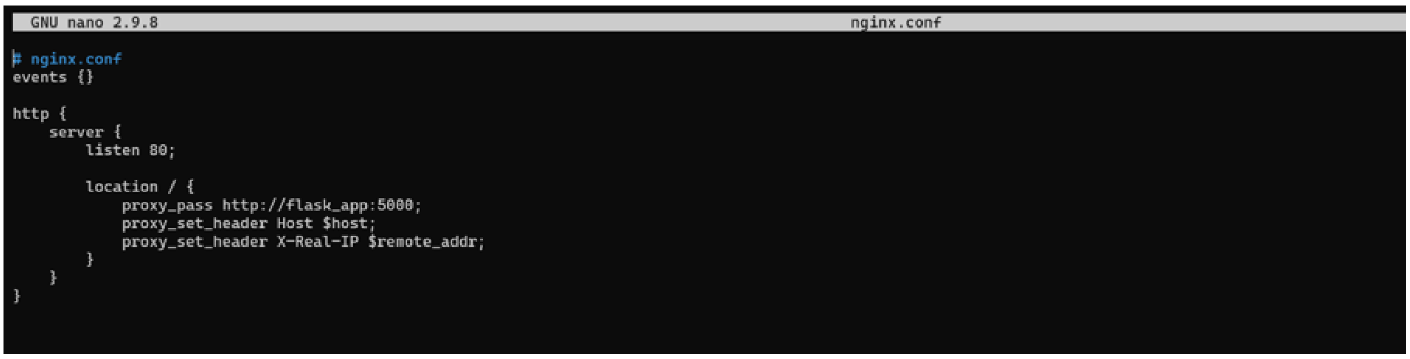


```
GNU nano 2.9.8                                app.py

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello From Sasi!"

# Example: Flask app listening on port 5000
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

3.To create a requirements text file

```
nano requirements.txt
```

```
Flask==1.1.4
MarkupSafe==1.1.1
```



```
GNU nano 2.9.8                                nginx.conf

# nginx.conf
events {}

http {
    server {
        listen 80;

        location / {
            proxy_pass http://flask_app:5000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
}
```

4.Now create nginx configuration file

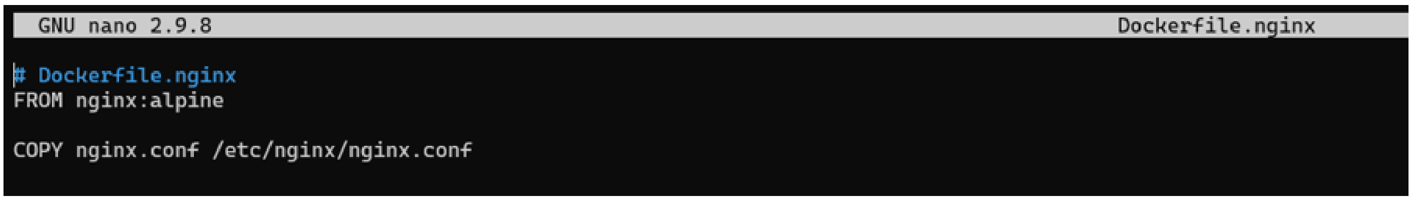
```
nano nginx.conf
```

```
#nginx.conf
events {}


http {
server {
    listen 80;
        location / {
        proxy pass http://flask_app: 5000;
            proxy set_header Host $host;
            proxy_set header X-Real-IP $remote_addr;
            }
       }
}
```

5.Create a docker nginx file

```
nano Dockerfile.nginx
```

```
FROM nginx:alpine


COPY nginx.conf /etc/nginx/nginx.conf
```



```
  GNU nano 2.9.8                                              Dockerfile.nginx

# Dockerfile.nginx
FROM nginx:alpine

COPY nginx.conf /etc/nginx/nginx.conf
```

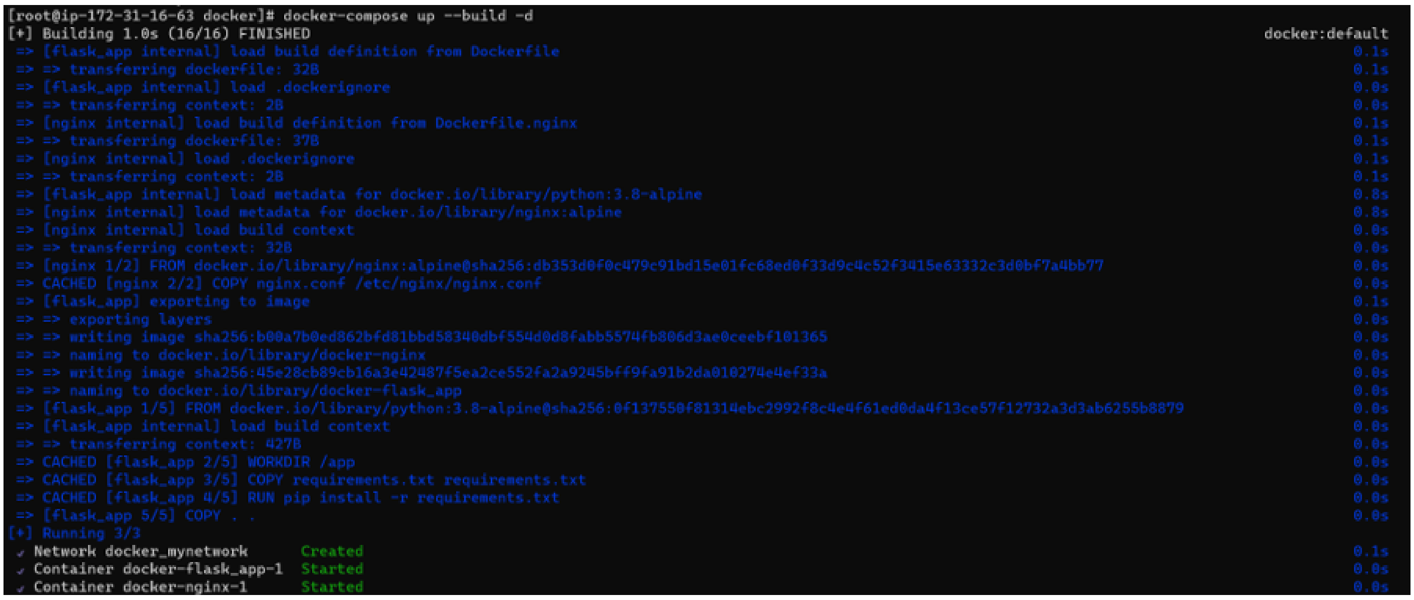
6.To create docker-compose file

```
nano docker-compose.yaml
```

```
version: '3'
services:
  flask_app:
    build:
      context: .
      dockerfile: Dockerfile
    networks:
      - mynetwork
  nginx:
    build:
      context: .
      dockerfile: Dockerfile.nginx
    networks:
      - mynetwork
    ports:
      - "8080:80"
networks:
  mynetwork:
```

7.After creating this file, now build the containers using docker compose

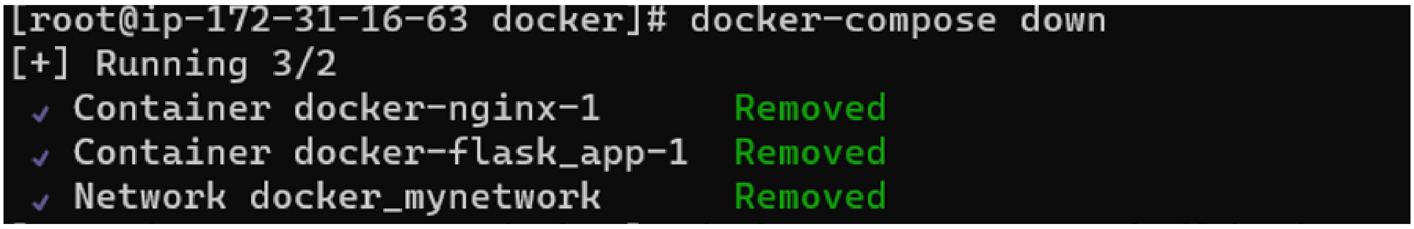
```
docker-compose up --build -d
```



```
[root@ip-172-31-16-63 docker]# docker-compose up --build -d
[+] Building 1.0s (16/16) FINISHED                                                                                              docker:default
 => [flask_app internal] load build definition from Dockerfile                                                                           0.1s
 => => transferring dockerfile: 32B                                                                                                      0.1s
 => [flask_app internal] load .dockerignore                                                                                              0.0s
 => => transferring context: 2B                                                                                                          0.0s
 => [nginx internal] load build definition from Dockerfile.nginx                                                                         0.1s
 => => transferring dockerfile: 37B                                                                                                      0.1s
 => [nginx internal] load .dockerignore                                                                                                  0.1s
 => => transferring context: 2B                                                                                                          0.1s
 => [flask_app internal] load metadata for docker.io/library/python:3.8-alpine                                                           0.8s
 => [nginx internal] load metadata for docker.io/library/nginx:alpine                                                                    0.8s
 => [nginx internal] load build context                                                                                                  0.0s
 => => transferring context: 32B                                                                                                         0.0s
 => [nginx 1/2] FROM docker.io/library/nginx:alpine@sha256:db353d0f0c479c91bd15e01fc68ed0f33d9c4c52f3415e63332c3d0bf7a4bb77             0.0s
 => CACHED [nginx 2/2] COPY nginx.conf /etc/nginx/nginx.conf                                                                             0.0s
 => [flask_app] exporting to image                                                                                                       0.1s
 => => exporting layers                                                                                                                  0.0s
 => => writing image sha256:b00a7b0ed862bfd81bbd58340dbf554d0d8fabb5574fb806d3ae0ceebf101365                                             0.0s
 => => naming to docker.io/library/docker-nginx                                                                                          0.0s
 => => writing image sha256:45e28cb89cb16a3e42487f5ea2ce552fa2a9245bff9fa91b2da010274e4ef33a                                             0.0s
 => => naming to docker.io/library/docker-flask_app                                                                                      0.0s
 => [flask_app 1/5] FROM docker.io/library/python:3.8-alpine@sha256:0f137550f81314ebc2992f8c4e4f61ed0da4f13ce57f12732a3d3ab6255b8879     0.0s
 => [flask_app internal] load build context                                                                                              0.0s
 => => transferring context: 427B                                                                                                        0.0s
 => CACHED [flask_app 2/5] WORKDIR /app                                                                                                  0.0s
 => CACHED [flask_app 3/5] COPY requirements.txt requirements.txt                                                                        0.0s
 => CACHED [flask_app 4/5] RUN pip install -r requirements.txt                                                                           0.0s
 => [flask_app 5/5] COPY . .                                                                                                             0.0s
[+] Running 3/3
 ✓ Network docker_mynetwork      Created                                                                                                 0.1s
 ✓ Container docker-flask_app-1  Started                                                                                                 0.0s
 ✓ Container docker-nginx-1      Started                                                                                                 0.0s
```

8.If anything will change means, use down the container and again up

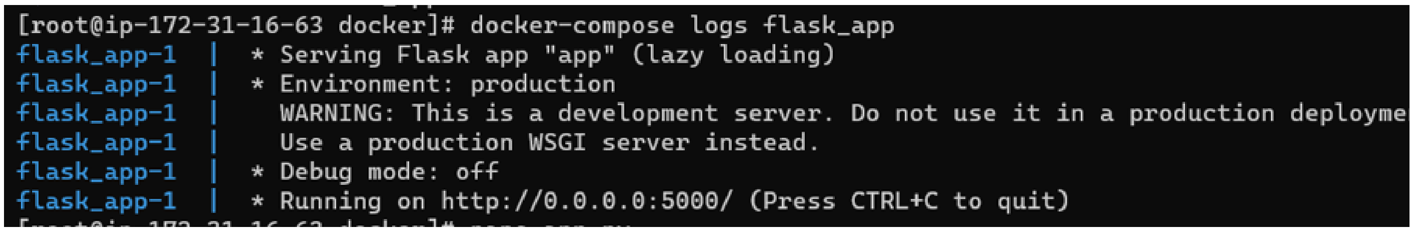
```
docker-compose down
```



```
[root@ip-172-31-16-63 docker]# docker-compose down
[+] Running 3/2
 ✓ Container docker-nginx-1      Removed
 ✓ Container docker-flask_app-1  Removed
 ✓ Network docker_mynetwork      Removed
```

9.If any error,check the logs using docker logs

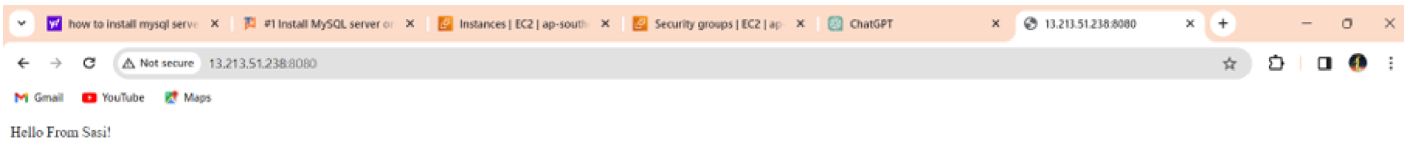
```
docker logs continer_name
```



```
[root@ip-172-31-16-63 docker]# docker-compose logs flask_app
flask_app-1  |  * Serving Flask app "app" (lazy loading)
flask_app-1  |  * Environment: production
flask_app-1  |    WARNING: This is a development server. Do not use it in a production deployme
flask_app-1  |    Use a production WSGI server instead.
flask_app-1  |  * Debug mode: off
flask_app-1  |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

10.To test the application in browser

```
http://localost:8080/
```


how to install mysql serve
#1 Install MySQL server o
Instances | EC2 | ap-south
Security groups | EC2 | ap
ChatGPT
13.213.51.238:8080
Not secure
13.213.51.238:8080
Gmail
YouTube
Maps

Hello From Sasi!

Docker-Commads-sample

Docker save

Save one or more images to a tar archive (streamed to STDOUT by default).

This can be useful when you want to transfer Docker images between systems or store them for backup purposes

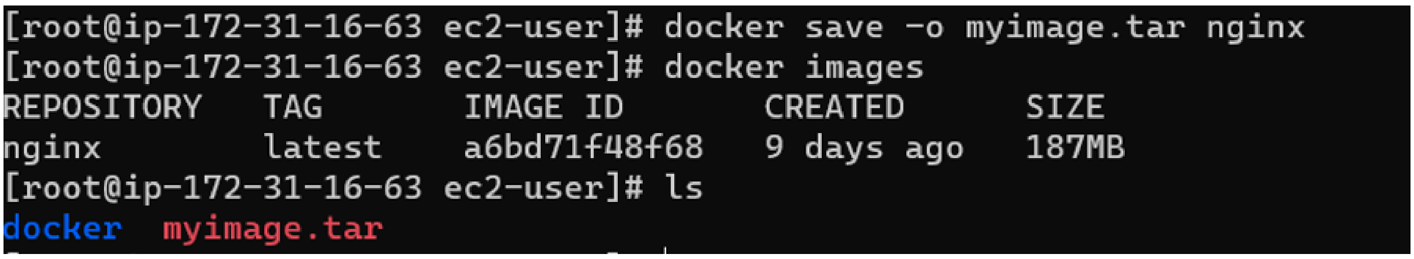
Syntax

```
Usage:  docker save [OPTIONS] IMAGE [IMAGE...]
```

Options -o, --output string Write to a file, instead of STDOUT

Example To save an docker image in tar format

```
docker save -o myimage.tar nginx
```



```
[root@ip-172-31-16-63 ec2-user]# docker save -o myimage.tar nginx
[root@ip-172-31-16-63 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
nginx        latest    a6bd71f48f68   9 days ago   187MB
[root@ip-172-31-16-63 ec2-user]# ls
docker  myimage.tar
```

Docker load

It is used to load Docker images from a tarball archive. It allows you to restore Docker images that were previously saved using the docker save

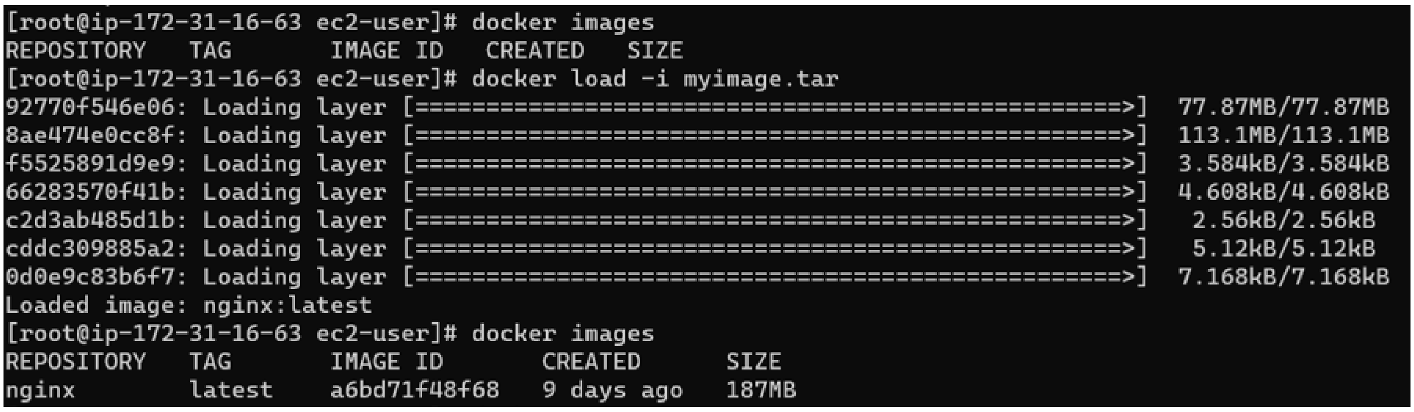
Syntax

```
docker load [OPTIONS]
```

Example

1.Load a Docker image from a tarball archive

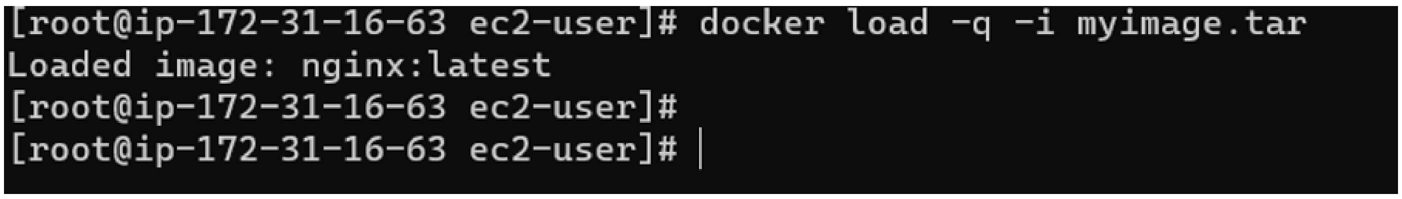
```
docker load -i myimage.tar
```



```
[root@ip-172-31-16-63 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
[root@ip-172-31-16-63 ec2-user]# docker load -i myimage.tar
92770f546e06: Loading layer [==================================================>]  77.87MB/77.87MB
8ae474e0cc8f: Loading layer [==================================================>]  113.1MB/113.1MB
f5525891d9e9: Loading layer [==================================================>]  3.584kB/3.584kB
66283570f41b: Loading layer [==================================================>]  4.608kB/4.608kB
c2d3ab485d1b: Loading layer [==================================================>]   2.56kB/2.56kB
cddc309885a2: Loading layer [==================================================>]   5.12kB/5.12kB
0d0e9c83b6f7: Loading layer [==================================================>]  7.168kB/7.168kB
Loaded image: nginx:latest
[root@ip-172-31-16-63 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
nginx        latest    a6bd71f48f68   9 days ago   187MB
```

2.Load a Docker image from a tarball archive and display only the image ID

```
docker load -q -i myimage.tar
```



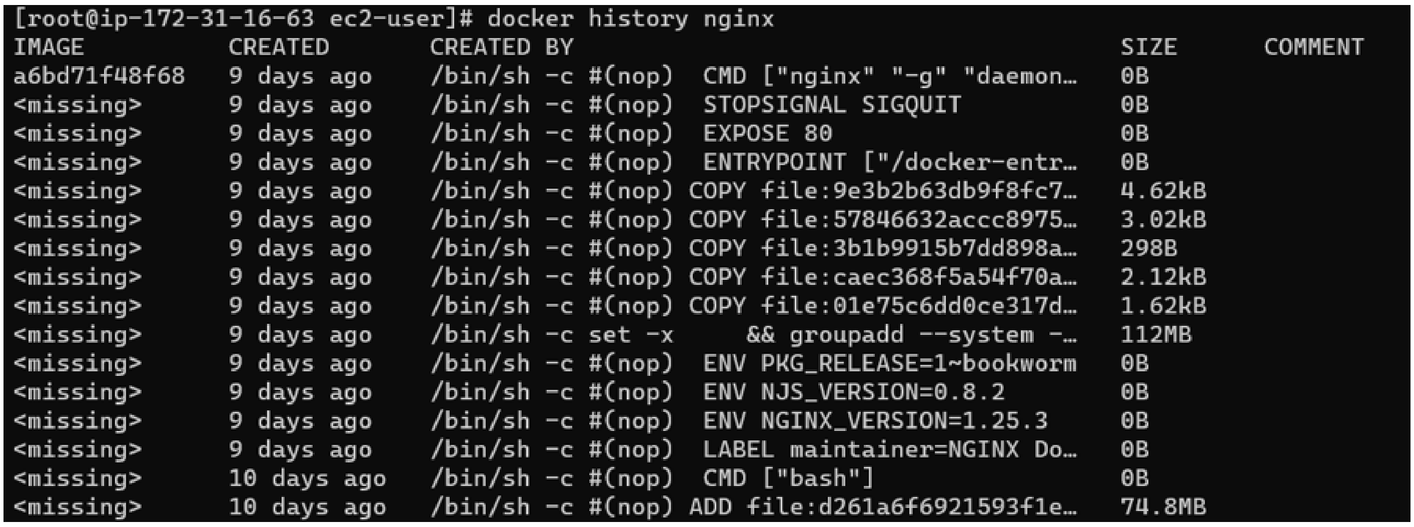
```
[root@ip-172-31-16-63 ec2-user]# docker load -q -i myimage.tar
Loaded image: nginx:latest
[root@ip-172-31-16-63 ec2-user]#
[root@ip-172-31-16-63 ec2-user]# |
```

Docker history

Show the history of an image

Syntax

```
docker history [OPTIONS] IMAGE
```



```
[root@ip-172-31-16-63 ec2-user]# docker history nginx
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
a6bd71f48f68   9 days ago     /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>      9 days ago     /bin/sh -c #(nop)  STOPSIGNAL SIGQUIT           0B
<missing>      9 days ago     /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>      9 days ago     /bin/sh -c #(nop)  ENTRYPOINT ["/docker-entr…   0B
<missing>      9 days ago     /bin/sh -c #(nop) COPY file:9e3b2b63db9f8fc7…   4.62kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY file:57846632accc8975…   3.02kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY file:3b1b9915b7dd898a…   298B
<missing>      9 days ago     /bin/sh -c #(nop) COPY file:caec368f5a54f70a…   2.12kB
<missing>      9 days ago     /bin/sh -c #(nop) COPY file:01e75c6dd0ce317d…   1.62kB
<missing>      9 days ago     /bin/sh -c set -x     && groupadd --system -…   112MB
<missing>      9 days ago     /bin/sh -c #(nop)  ENV PKG_RELEASE=1~bookworm   0B
<missing>      9 days ago     /bin/sh -c #(nop)  ENV NJS_VERSION=0.8.2        0B
<missing>      9 days ago     /bin/sh -c #(nop)  ENV NGINX_VERSION=1.25.3     0B
<missing>      9 days ago     /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>      10 days ago    /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      10 days ago    /bin/sh -c #(nop) ADD file:d261a6f6921593f1e…   74.8MB
```

Docker cp

To copy the file or folder in local to container

Syntax

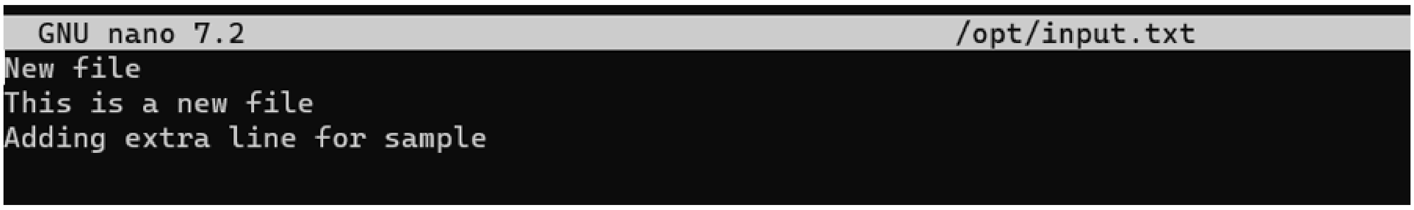
```
docker cp /source_path CONTAINER_ID:/destination_path
```

Example

1.To copy the file using docker

```
docker cp /opt/new/file.txt my-new:/opt/input.txt
```

2.To check the file will copy or not



```
  GNU nano 7.2                                          /opt/input.txt
New file
This is a new file
Adding extra line for sample
```