Core Docker Concepts

- Subtopics: Containers, Volumes, Ports, Inspect, Dockerfile, Network

<u>NOTES</u>:

https://drive.google.com/drive/folders/1SYIE-gBImytTLPjfOyfhPX84JrKnCS3Q?usp=share_link

What is Docker?

- • Docker is a platform for developing, shipping, and running applications inside containers.
- • Containers package an application and its dependencies into a single lightweight unit.
- • Ensures consistency across development, testing, and production environments.

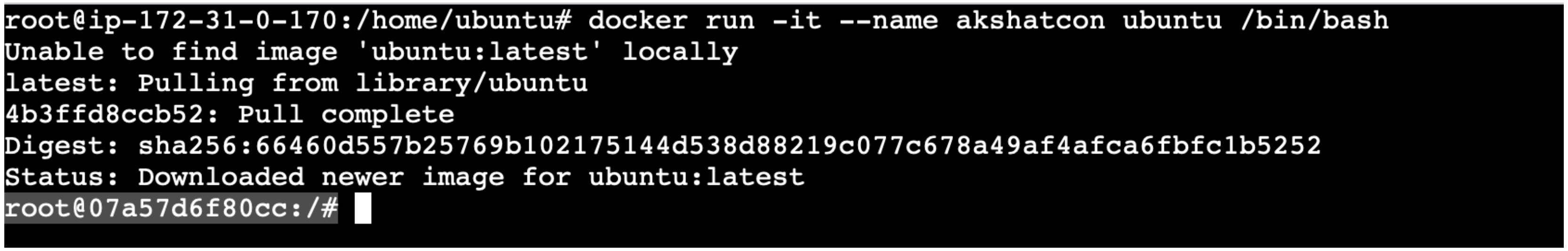
Docker Container

- • A Docker container is a runnable instance of a Docker image.
- • Containers are isolated environments sharing the host OS kernel.
- • Each container can run applications independently.

- Key Commands:
- docker run – create and start a container
- docker ps – list running containers
- docker stop <container> – stop a container
- docker rm <container> – remove a container

- Example:
- docker run -it ubuntu –name c01 /bin/bash

Create a new container

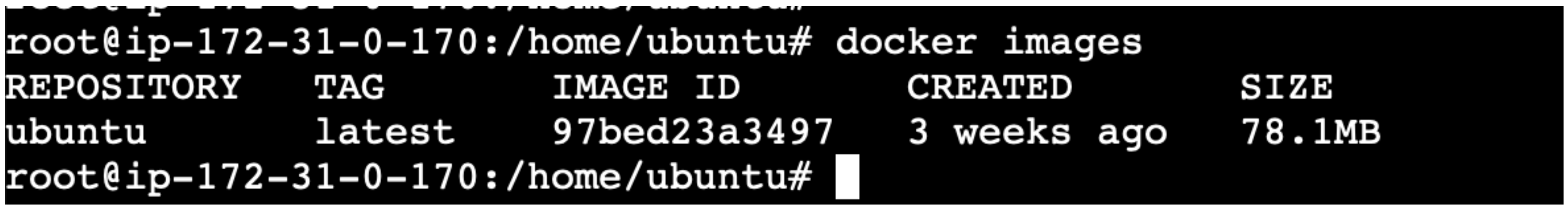


```
root@ip-172-31-0-170:/home/ubuntu# docker run -it --name akshatcon ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
4b3ffd8ccb52: Pull complete
Digest: sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Status: Downloaded newer image for ubuntu:latest
root@07a57d6f80cc:/# 
```

How will we come out the container:

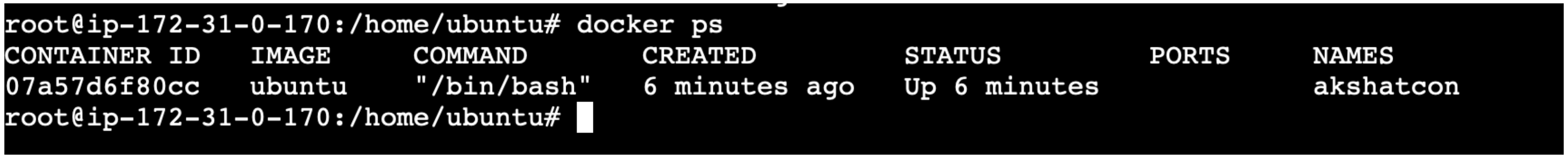
Press ctrl p -> save the process
Press ctrl q -> quit the process

See the images present in local registry : docker images
# you will have ubuntu image because we already created one container from ubuntu image ...so it first pulled the image in local registry and then created the container



```
root@ip-172-31-0-170:/home/ubuntu# docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    97bed23a3497   3 weeks ago    78.1MB
root@ip-172-31-0-170:/home/ubuntu# 
```

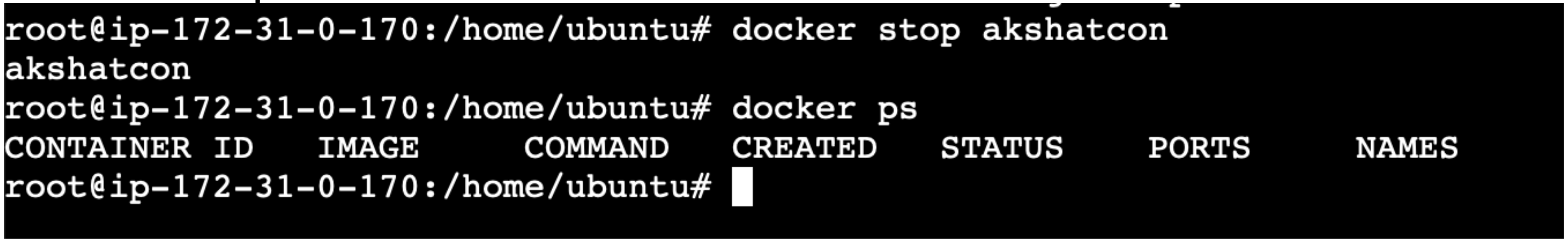
- **We want to see the list of running container:** docker ps



```
root@ip-172-31-0-170:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS         PORTS     NAMES
07a57d6f80cc   ubuntu    "/bin/bash"   6 minutes ago   Up 6 minutes             akshatcon
root@ip-172-31-0-170:/home/ubuntu# 
```

- We can **stop a container** (release the ram and vcpu) :

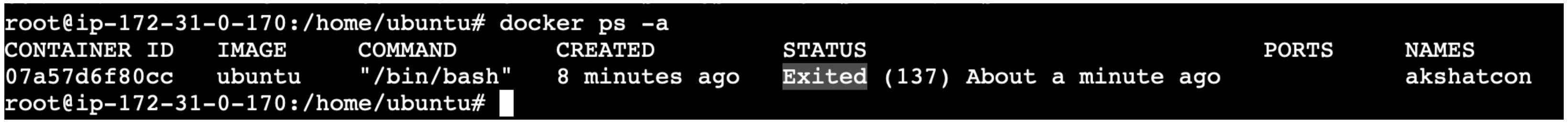
  docker stop containername or id



```
root@ip-172-31-0-170:/home/ubuntu# docker stop akshatcon
akshatcon
root@ip-172-31-0-170:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-0-170:/home/ubuntu# 
```

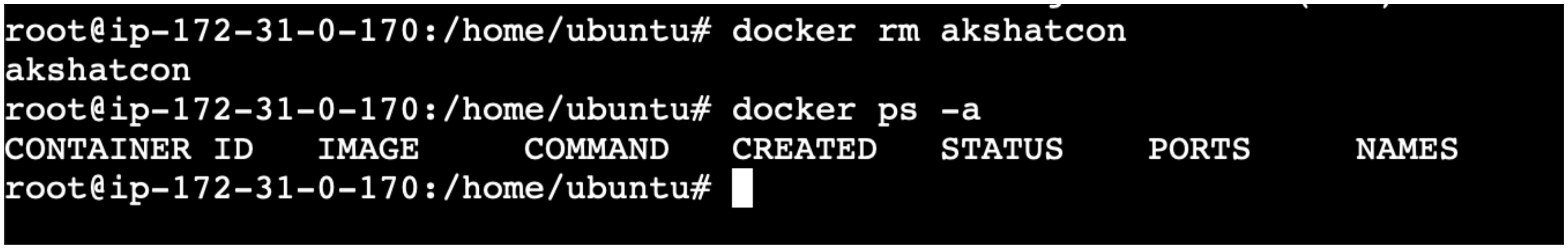
- When we run docker ps it shows me no container but I want to see stopped container as well. We will **list all the container**

  so we will run docker ps –a   # a -> all



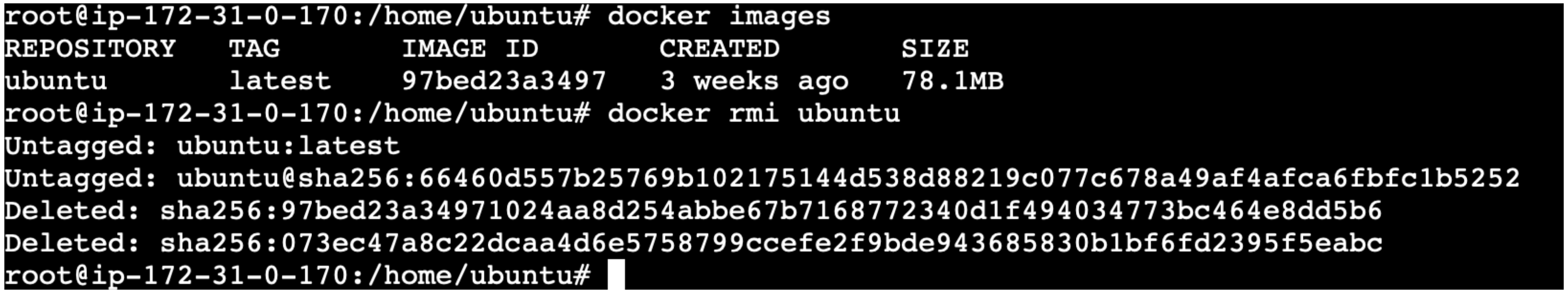
```
root@ip-172-31-0-170:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                            PORTS     NAMES
07a57d6f80cc   ubuntu    "/bin/bash"   8 minutes ago   Exited (137) About a minute ago             akshatcon
root@ip-172-31-0-170:/home/ubuntu# 
```

- **To remove a container:** docker rm containername



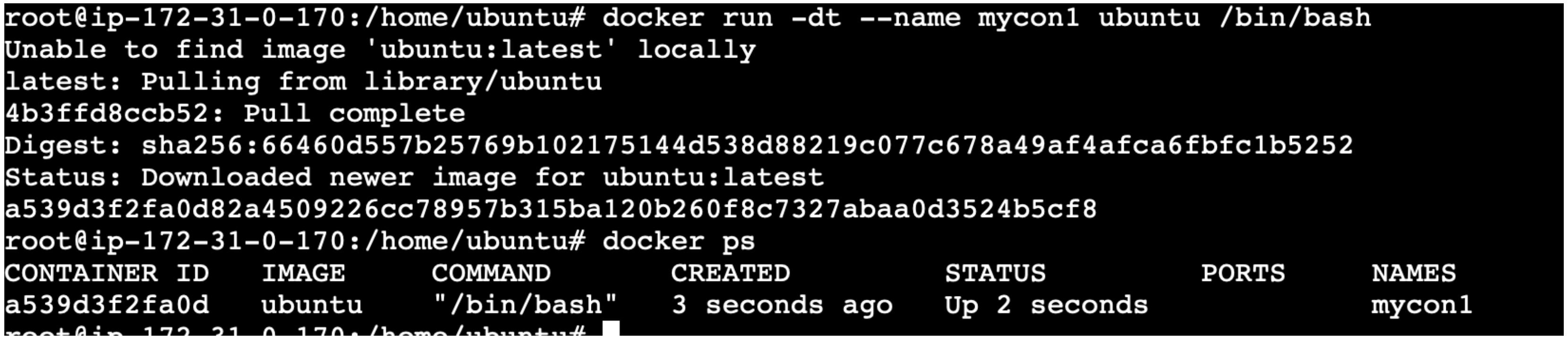
```
root@ip-172-31-0-170:/home/ubuntu# docker rm akshatcon
akshatcon
root@ip-172-31-0-170:/home/ubuntu# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@ip-172-31-0-170:/home/ubuntu# 
```

- **To remove a image from local registry:** docker rmi imagename



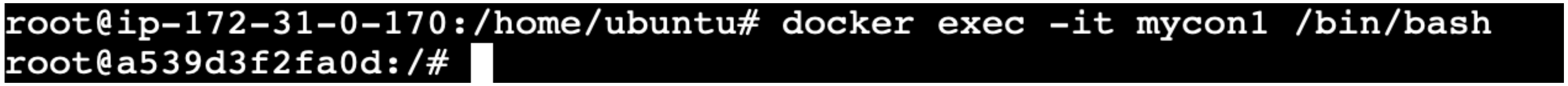
```
root@ip-172-31-0-170:/home/ubuntu# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    97bed23a3497   3 weeks ago   78.1MB
root@ip-172-31-0-170:/home/ubuntu# docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Deleted: sha256:97bed23a34971024aa8d254abbe67b7168772340d1f494034773bc464e8dd5b6
Deleted: sha256:073ec47a8c22dcaa4d6e5758799ccefe2f9bde943685830b1bf6fd2395f5eabc
root@ip-172-31-0-170:/home/ubuntu# 
```

- **We want to create a new container without entering inside it:**



```
root@ip-172-31-0-170:/home/ubuntu# docker run -dt --name mycon1 ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
4b3ffd8ccb52: Pull complete
Digest: sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b5252
Status: Downloaded newer image for ubuntu:latest
a539d3f2fa0d82a4509226cc78957b315ba120b260f8c7327abaa0d3524b5cf8
root@ip-172-31-0-170:/home/ubuntu# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS         PORTS     NAMES
a539d3f2fa0d   ubuntu    "/bin/bash"   3 seconds ago   Up 2 seconds             mycon1
root@ip-172-31-0-170:/home/ubuntu# 
```

- ***You want to enter inside the existing containers:***



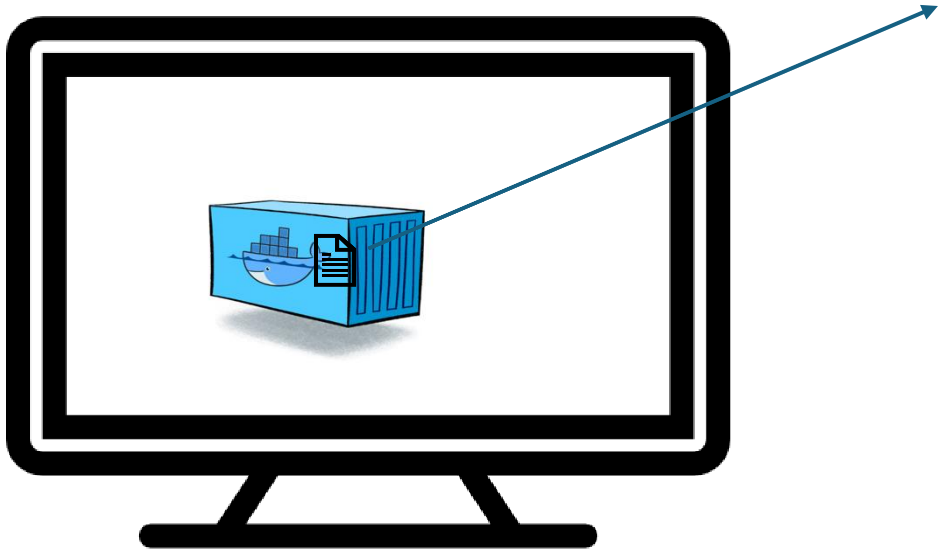
```
root@ip-172-31-0-170:/home/ubuntu# docker exec -it mycon1 /bin/bash
root@a539d3f2fa0d:/# 
```

Points we should know

- Containers cannot have same id or container name

Docker Volume

- In Docker, a volume is a persistent storage mechanism that allows data to live outside the container's writable layer — meaning your data is not lost when the container stops, restarts, or is deleted.

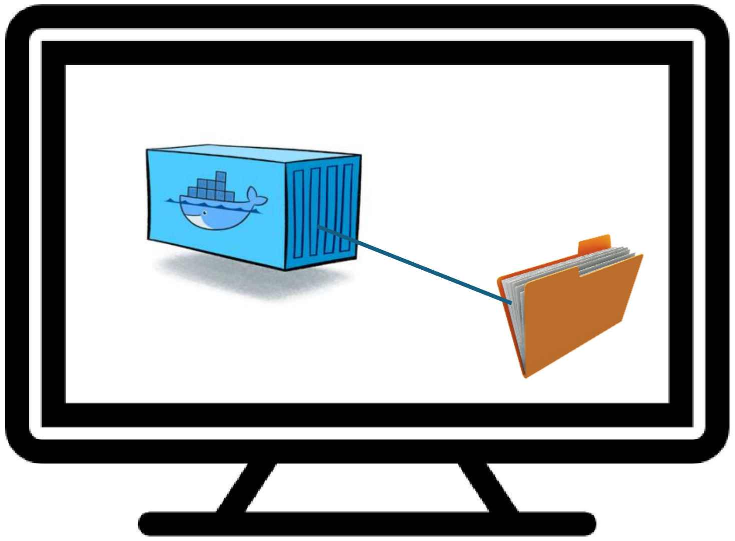


Inside the container there are some logs/metrics/important files which is very important for our infra

- But due to some reason (like server overload / machine restart) the container is deleted. And due to which the files are lost !!!

- As these files are present in container if the container is deleted the files are removed ...

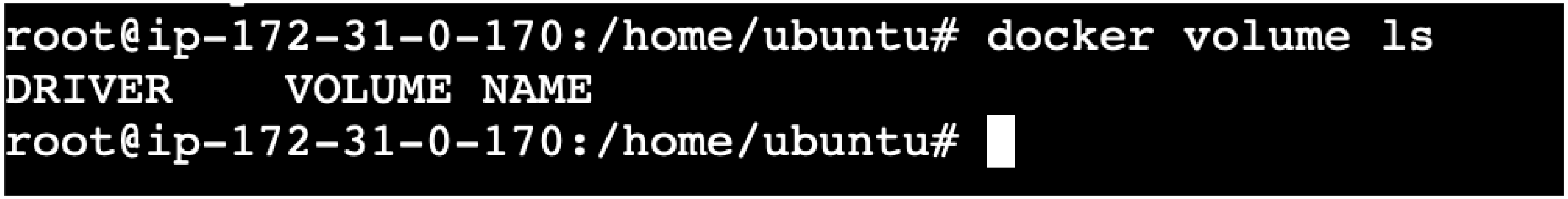
- Solution is docker volume



We can map a directory of the container (for example log directory) with the docker volume which is created in the machine within which your docker is running .

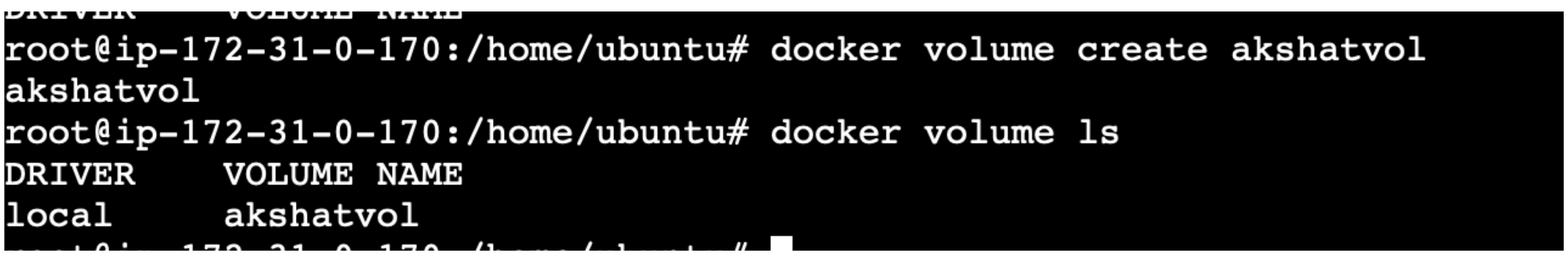
Now if your file is deleted still the files will be present in docker volume.

- See the list of volumes : docker volume ls



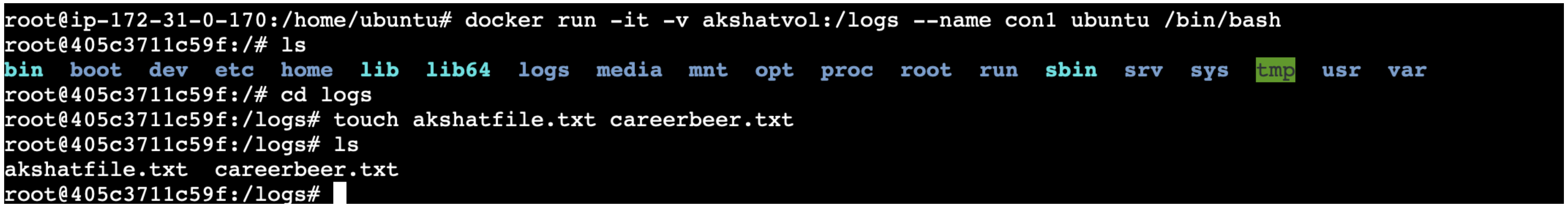
```
root@ip-172-31-0-170:/home/ubuntu# docker volume ls
DRIVER     VOLUME NAME
root@ip-172-31-0-170:/home/ubuntu# 
```

- Create a docker volume :



```
DRIVER     VOLUME NAME
root@ip-172-31-0-170:/home/ubuntu# docker volume create akshatvol
akshatvol
root@ip-172-31-0-170:/home/ubuntu# docker volume ls
DRIVER     VOLUME NAME
local      akshatvol
root@ip-172-31-0-170:/home/ubuntu# 
```

- Create a container of which a directory named logs (anyname)will be mapped with akshatvol



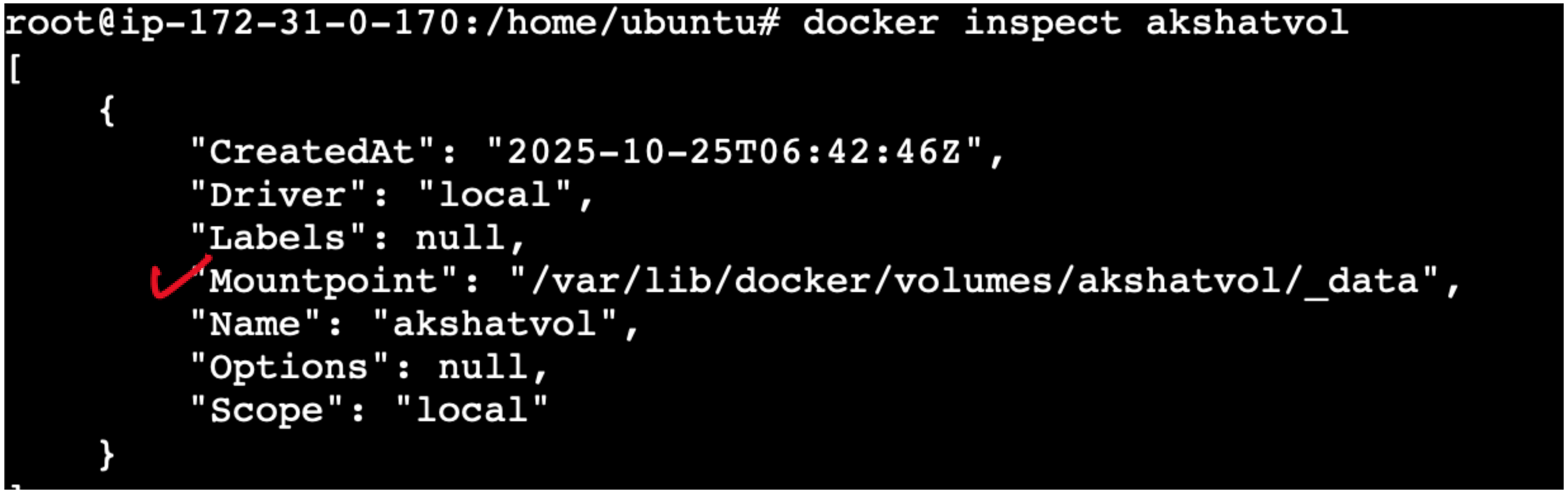
```
root@ip-172-31-0-170:/home/ubuntu# docker run -it -v akshatvol:/logs --name con1 ubuntu /bin/bash
root@405c3711c59f:/# ls
bin  boot  dev  etc  home  lib  lib64  logs  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@405c3711c59f:/# cd logs
root@405c3711c59f:/logs# touch akshatfile.txt careerbeer.txt
root@405c3711c59f:/logs# ls
akshatfile.txt  careerbeer.txt
root@405c3711c59f:/logs# 
```

(you will see that akshatvol which is a dockervolume is mapped with logs directory of the container)

Ctrl p and ctrl q to come out of the container

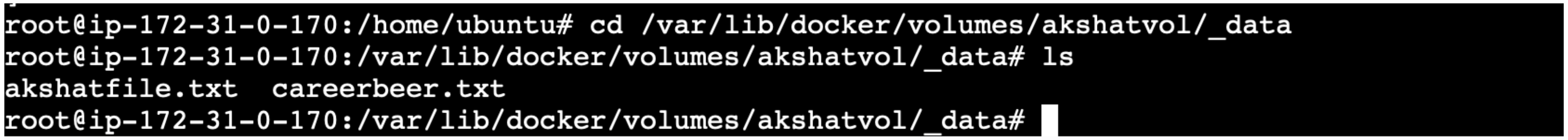
- I want to see where the volume is created:

  docker inspect volumename



```
root@ip-172-31-0-170:/home/ubuntu# docker inspect akshatvol
[
    {
        "CreatedAt": "2025-10-25T06:42:46Z",
        "Driver": "local",
        "Labels": null,
      ✓ "Mountpoint": "/var/lib/docker/volumes/akshatvol/_data",
        "Name": "akshatvol",
        "Options": null,
        "Scope": "local"
    }
```

- See the files in the docker volume:



```
root@ip-172-31-0-170:/home/ubuntu# cd /var/lib/docker/volumes/akshatvol/_data
root@ip-172-31-0-170:/var/lib/docker/volumes/akshatvol/_data# ls
akshatfile.txt  careerbeer.txt
root@ip-172-31-0-170:/var/lib/docker/volumes/akshatvol/_data#
```

Point you should know about dockervolume

- Docker volume cannot be mapped to existing container

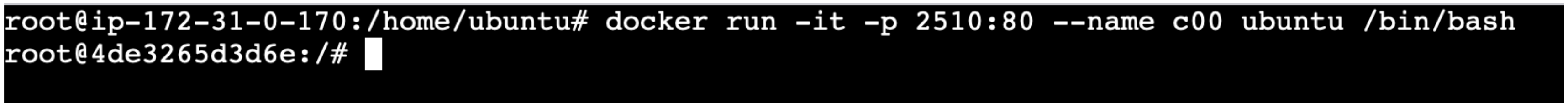
Docker Port Expose

- • Ports allow external communication with containers.
- • EXPOSE in Dockerfile documents the port, while '-p' publishes it.
- Port expose can only be performed while creating the container

- Example:
- docker run -p 8080:80 nginx
- Maps host port 8080 → container port 80.

- Command:
- EXPOSE <port> (in Dockerfile)

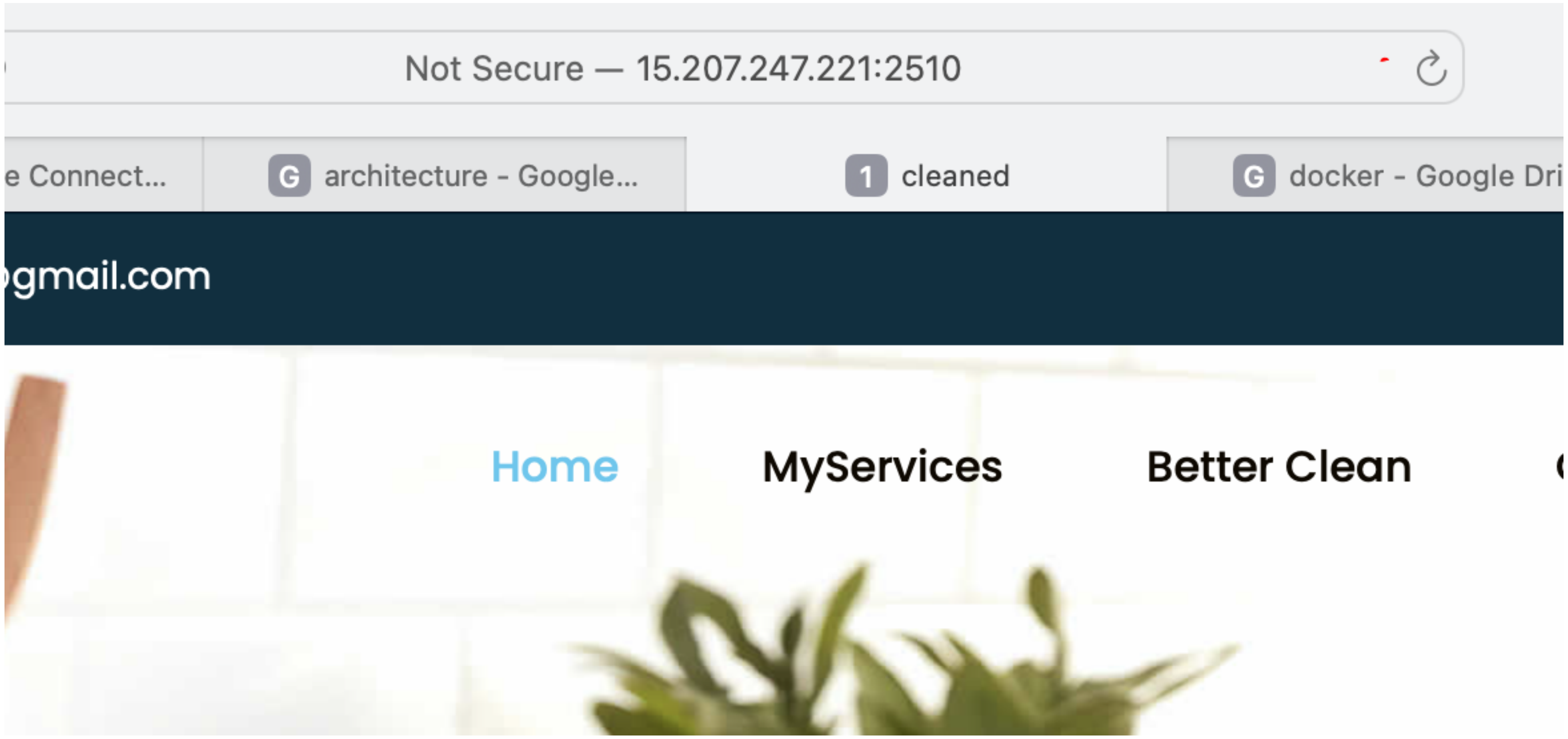
- Port expose example:



```
root@ip-172-31-0-170:/home/ubuntu# docker run -it -p 2510:80 --name c00 ubuntu /bin/bash
root@4de3265d3d6e:/#
```

- You can now deploy apache website inside the container

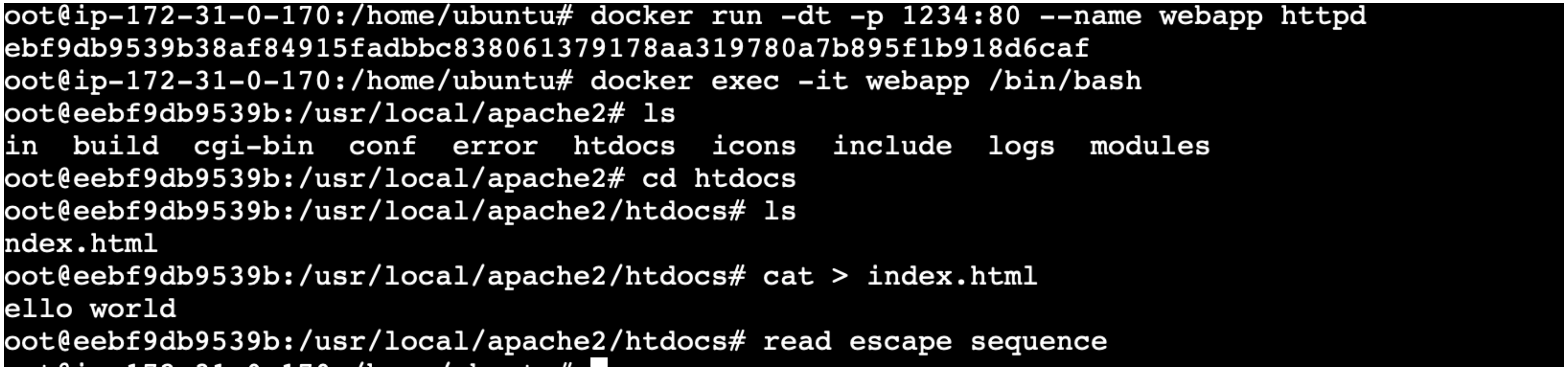
(in the end ...do not forget to start apache via service apache2 start)



Not Secure — 15.207.247.221:2510

- We can directly use apache image also

https://hub.docker.com/_/httpd



```
oot@ip-172-31-0-170:/home/ubuntu# docker run -dt -p 1234:80 --name webapp httpd
ebf9db9539b38af84915fadbbc838061379178aa319780a7b895f1b918d6caf
oot@ip-172-31-0-170:/home/ubuntu# docker exec -it webapp /bin/bash
oot@eebf9db9539b:/usr/local/apache2# ls
in  build  cgi-bin  conf  error  htdocs  icons  include  logs  modules
oot@eebf9db9539b:/usr/local/apache2# cd htdocs
oot@eebf9db9539b:/usr/local/apache2/htdocs# ls
ndex.html
oot@eebf9db9539b:/usr/local/apache2/htdocs# cat > index.html
ello world
oot@eebf9db9539b:/usr/local/apache2/htdocs# read escape sequence
```

docker run -dt -p 1234:80 --name webapp httpd

docker exec -it webapp /bin/bash

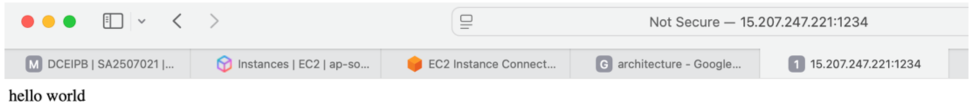
-> cd htdocs

-> ls

-> cat > index.html

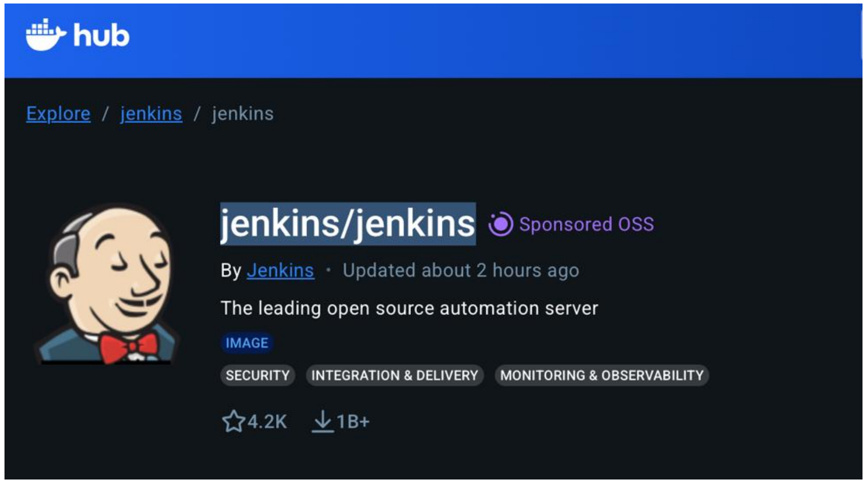
hello world

-> ctrl p and ctrl q



hello world

- Deploy Jenkins container:

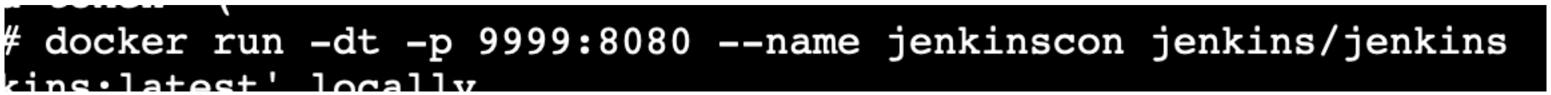


Explore / jenkins / jenkins

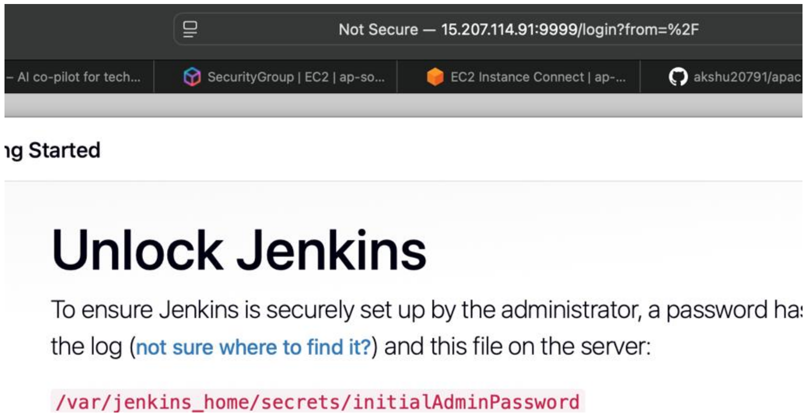
jenkins/jenkins Sponsored OSS

By Jenkins · Updated about 2 hours ago

The leading open source automation server



```
# docker run -dt -p 9999:8080 --name jenkinscon jenkins/jenkins
kins:latest' locally
```



Not Secure — 15.207.114.91:9999/login?from=%2F

– AI co-pilot for tech...
SecurityGroup | EC2 | ap-so...
EC2 Instance Connect | ap-...
akshu20791/apac

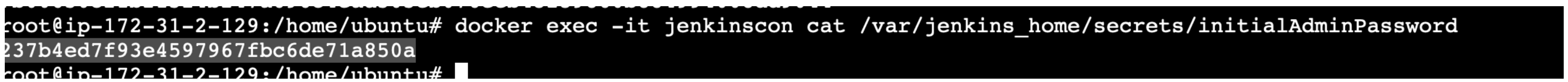
ıg Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password ha

the log (not sure where to find it?) and this file on the server:

`/var/jenkins_home/secrets/initialAdminPassword`



```
root@ip-172-31-2-129:/home/ubuntu# docker exec -it jenkinscon cat /var/jenkins_home/secrets/initialAdminPassword
237b4ed7f93e4597967fbc6de71a850a
root@ip-172-31-2-129:/home/ubuntu#
```

Docker Inspect

- • docker inspect returns detailed information about Docker objects.
- • Provides low-level configuration and runtime data in JSON format.

- Usage:
- docker inspect <container_name>

- Outputs:
- Container ID, Mounts, Networks, Ports, Image Info, Environment Variables.

Dockerfile

- • A Dockerfile is a text file containing instructions to build a Docker image.
- • Defines environment setup, dependencies, and commands to run.

- Common Instructions:
- FROM – base image
- RUN – execute commands
- COPY / ADD – copy files
- EXPOSE – specify ports
- CMD – define start command

REMEMBER : DOCKER FILE NAME IS ALWAYS Dockerfile only

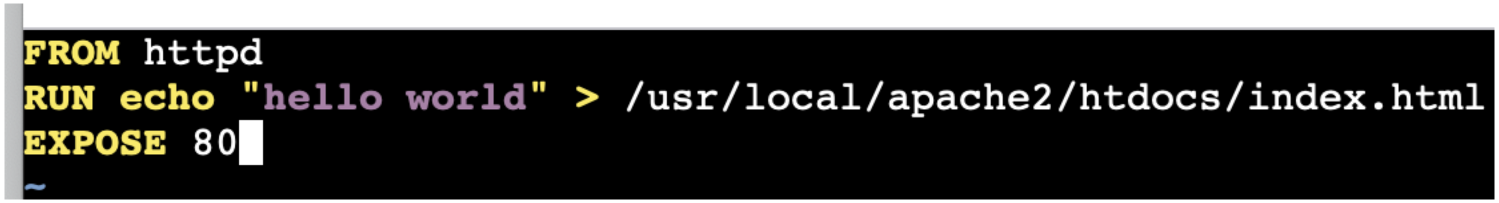
- Example:

```
FROM ubuntu
RUN apt-get update && apt-get install -y apache2
COPY . /var/www/html
EXPOSE 80
CMD ["apachectl", "-D", "FOREGROUND"]
```

To create custom image: docker build –t myimg .

- **Create a Docker file (apache)**

vi Dockerfile



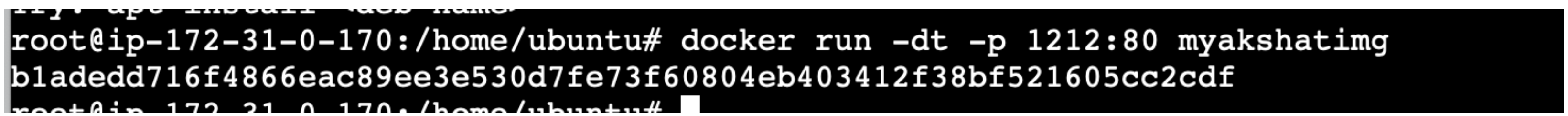
```
FROM httpd
RUN echo "hello world" > /usr/local/apache2/htdocs/index.html
EXPOSE 80
~
```

- **Execute the Dockerfile**

  docker build –t myakshatimg .

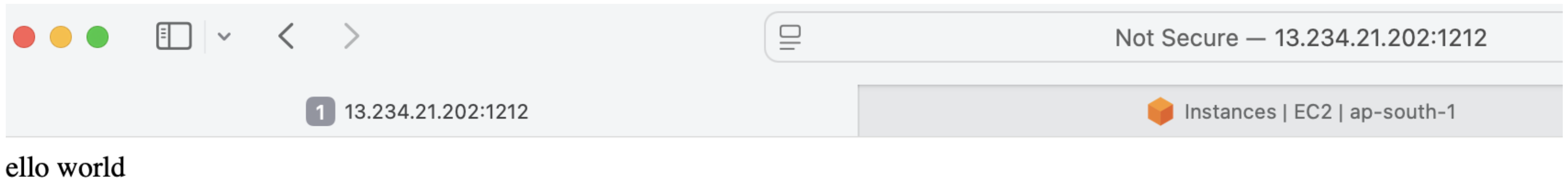
  docker images #will show myakshatimg to you

- Now you can create a container from the custom image



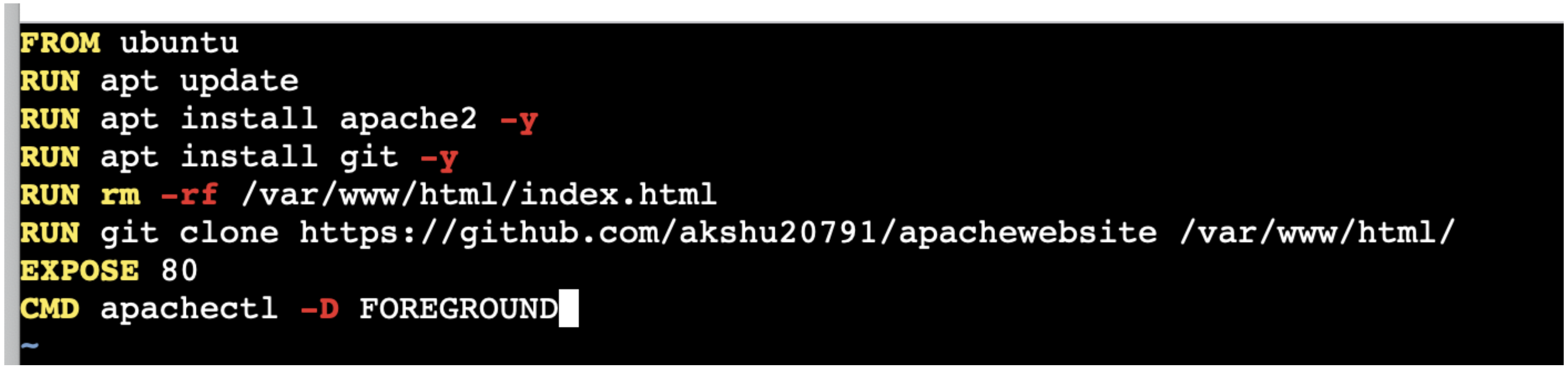
```
root@ip-172-31-0-170:/home/ubuntu# docker run -dt -p 1212:80 myakshatimg
b1adedd716f4866eac89ee3e530d7fe73f60804eb403412f38bf521605cc2cdf
root@ip-172-31-0-170:/home/ubuntu#
```

- Lets see the port 1212



ello world

- Docker file with ubuntu as base image

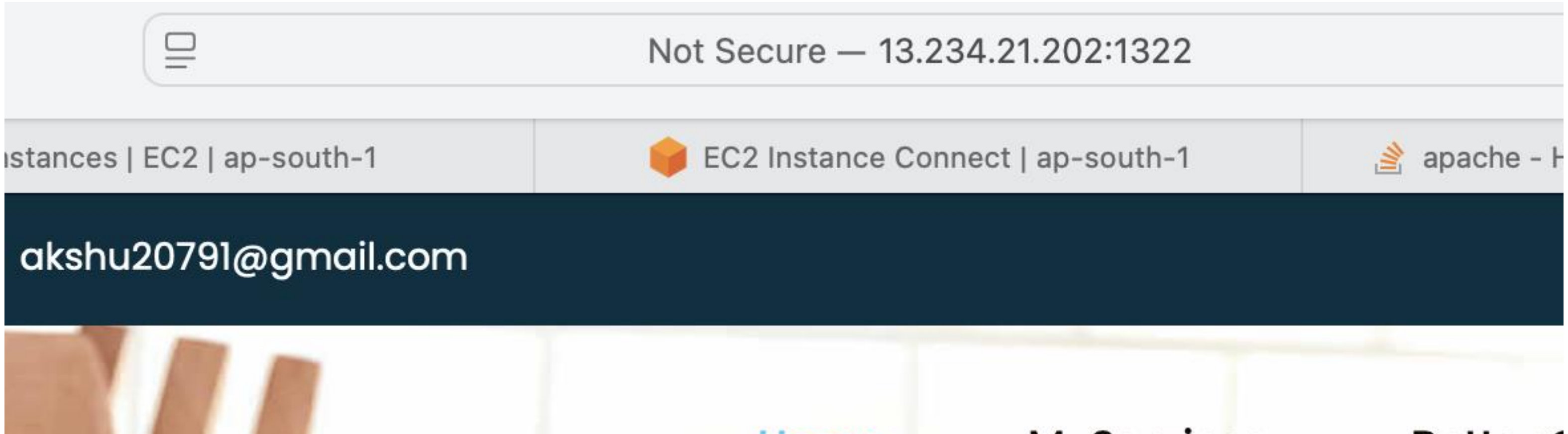


```
FROM ubuntu
RUN apt update
RUN apt install apache2 -y
RUN apt install git -y
RUN rm -rf /var/www/html/index.html
RUN git clone https://github.com/akshu20791/apachewebsite /var/www/html/
EXPOSE 80
CMD apachectl -D FOREGROUND
~
```

docker build –t mysecondimg1 .

docker images

docker run –dt –p 1322:80 mysecondimg1



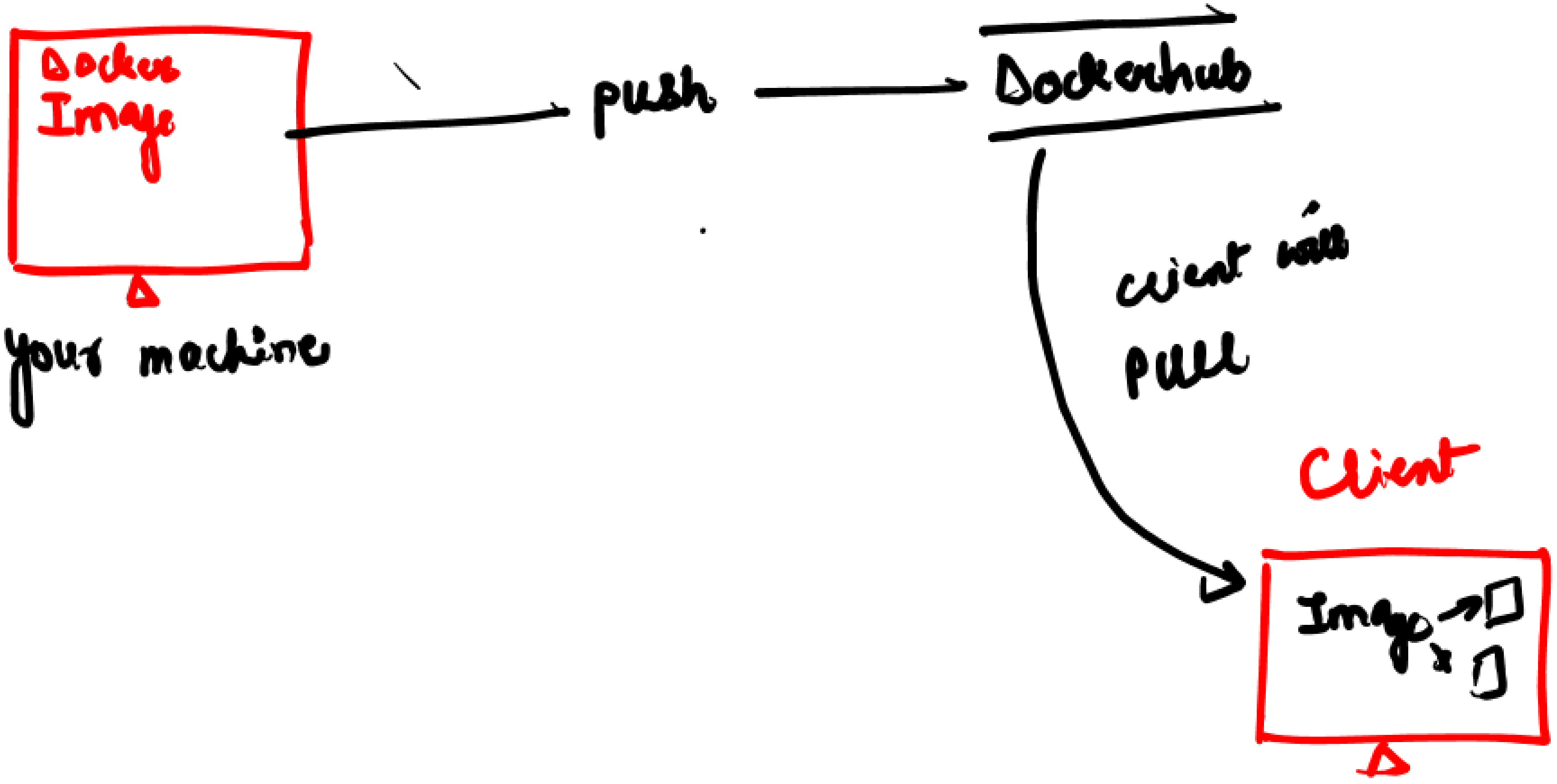
Not Secure — 13.234.21.202:1322

istances | EC2 | ap-south-1
EC2 Instance Connect | ap-south-1
apache - H

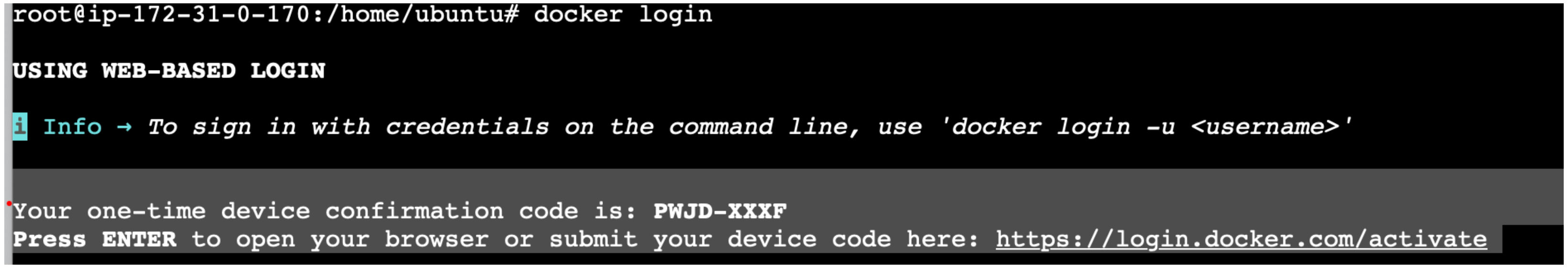
akshu20791@gmail.com

Docker logging

Lets suppose you are working a service provider and you created a custom image and now you want to share it with the client .

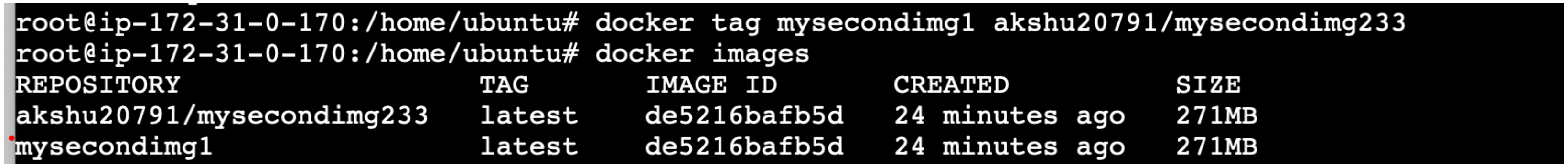


- Login to dockerhub from Developer machine



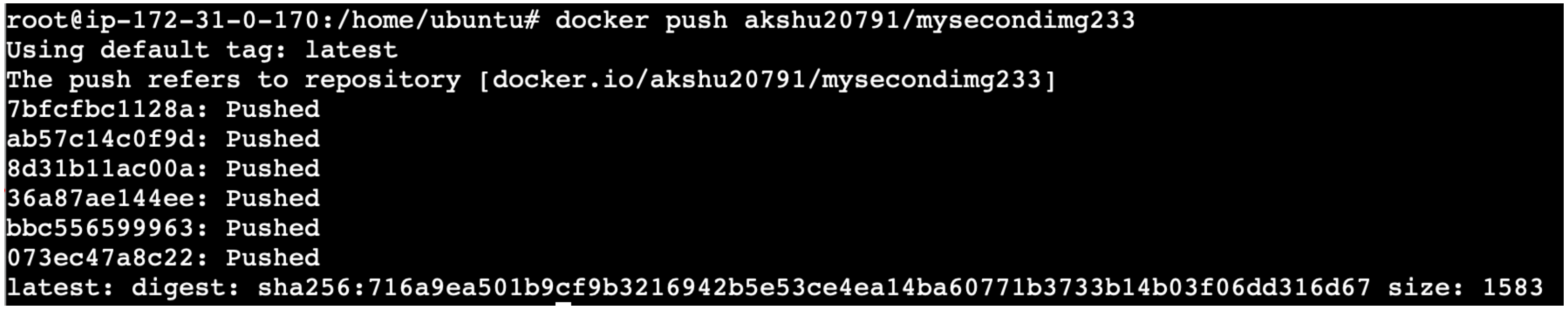
```
root@ip-172-31-0-170:/home/ubuntu# docker login

USING WEB-BASED LOGIN

i Info → To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: PWJD-XXXF
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate
```

- Remember: You can only push the images in the format dockerhubusername/imagename

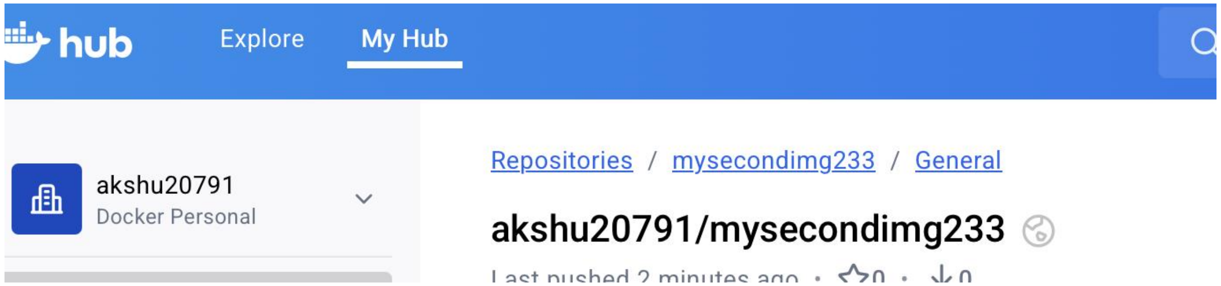


```
root@ip-172-31-0-170:/home/ubuntu# docker tag mysecondimg1 akshu20791/mysecondimg233
root@ip-172-31-0-170:/home/ubuntu# docker images
REPOSITORY                  TAG       IMAGE ID       CREATED          SIZE
akshu20791/mysecondimg233   latest    de5216bafb5d   24 minutes ago   271MB
mysecondimg1                latest    de5216bafb5d   24 minutes ago   271MB
```

- Push the image:



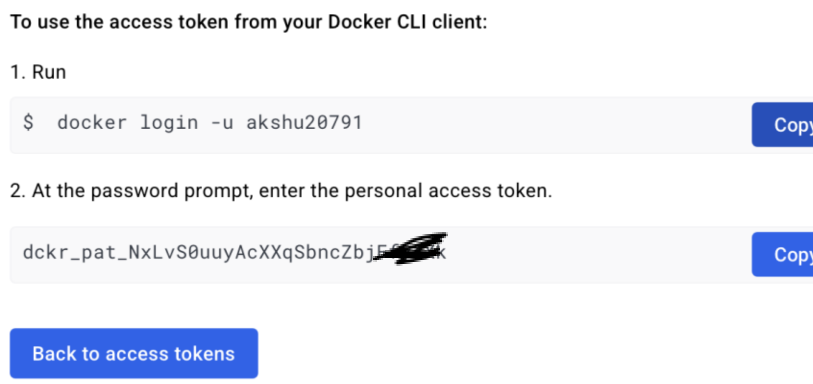
```
root@ip-172-31-0-170:/home/ubuntu# docker push akshu20791/mysecondimg233
Using default tag: latest
The push refers to repository [docker.io/akshu20791/mysecondimg233]
7bfcfbc1128a: Pushed
ab57c14c0f9d: Pushed
8d31b11ac00a: Pushed
36a87ae144ee: Pushed
bbc556599963: Pushed
073ec47a8c22: Pushed
latest: digest: sha256:716a9ea501b9cf9b3216942b5e53ce4ea14ba60771b3733b14b03f06dd316d67 size: 1583
```



akshu20791/mysecondimg233

Last pushed 2 minutes ago · 0 · 0

- We will create a client machine (instance with docker installed) and we will pull the image
  But if the image is private we cannot directly pull as we need permissions to do it .
  We can generate PAT :
  Go to account setting -> PAT -> Generate new token ->
  Token name : anyname
  Access permission: Read, write, Delete



To use the access token from your Docker CLI client:

1. Run

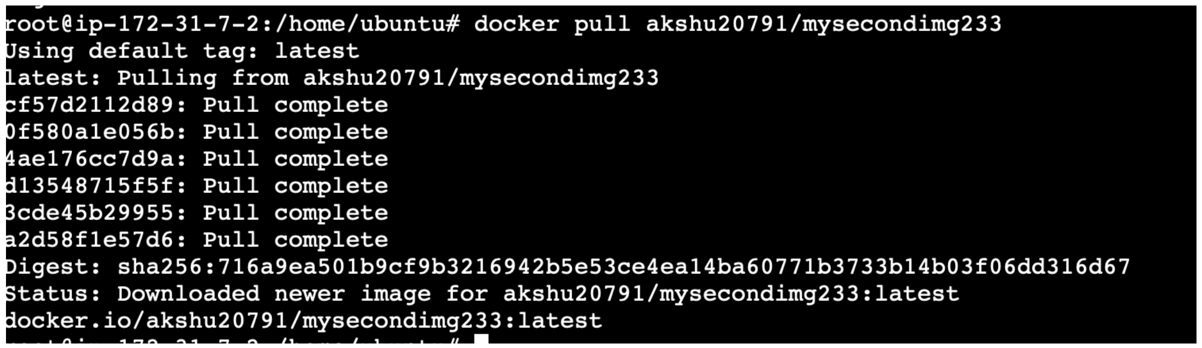
```
$ docker login -u akshu20791
```

2. At the password prompt, enter the personal access token.

```
dckr_pat_NxLvS0uuyAcXXqSbncZbj[redacted]
```

- Now share it with the client .

  Now client will connect to the Client instance:
  client will run the login commands



```
root@ip-172-31-7-2:/home/ubuntu# docker pull akshu20791/mysecondimg233
Using default tag: latest
latest: Pulling from akshu20791/mysecondimg233
cf57d2112d89: Pull complete
0f580a1e056b: Pull complete
4ae176cc7d9a: Pull complete
d13548715f5f: Pull complete
3cde45b29955: Pull complete
a2d58f1e57d6: Pull complete
Digest: sha256:716a9ea501b9cf9b3216942b5e53ce4ea14ba60771b3733b14b03f06dd316d67
Status: Downloaded newer image for akshu20791/mysecondimg233:latest
docker.io/akshu20791/mysecondimg233:latest
```

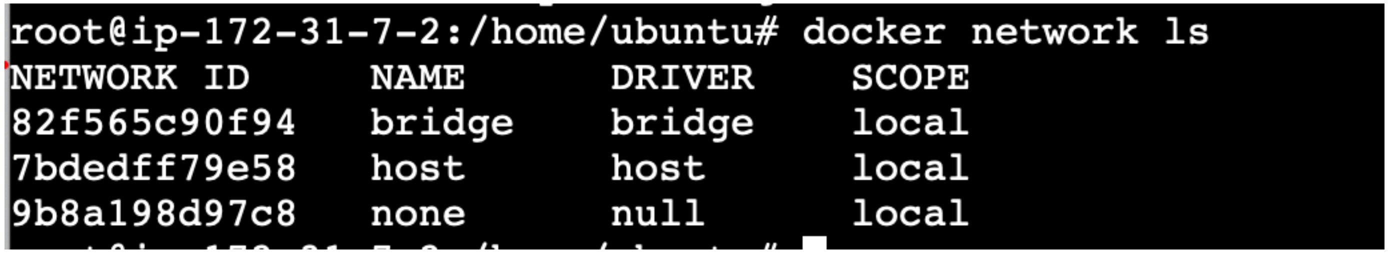
Docker Network

- Docker networks allow communication between containers.
- Containers on the same network can connect using container names.

- Types of Networks:
- Bridge (default)
- Host
- Overlay
- None networ

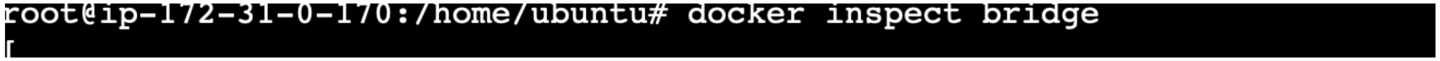
- Commands:
- docker network ls
- docker network create mynetwork
- docker run -d --network=mynetwork nginx

- Get the list of all the network :



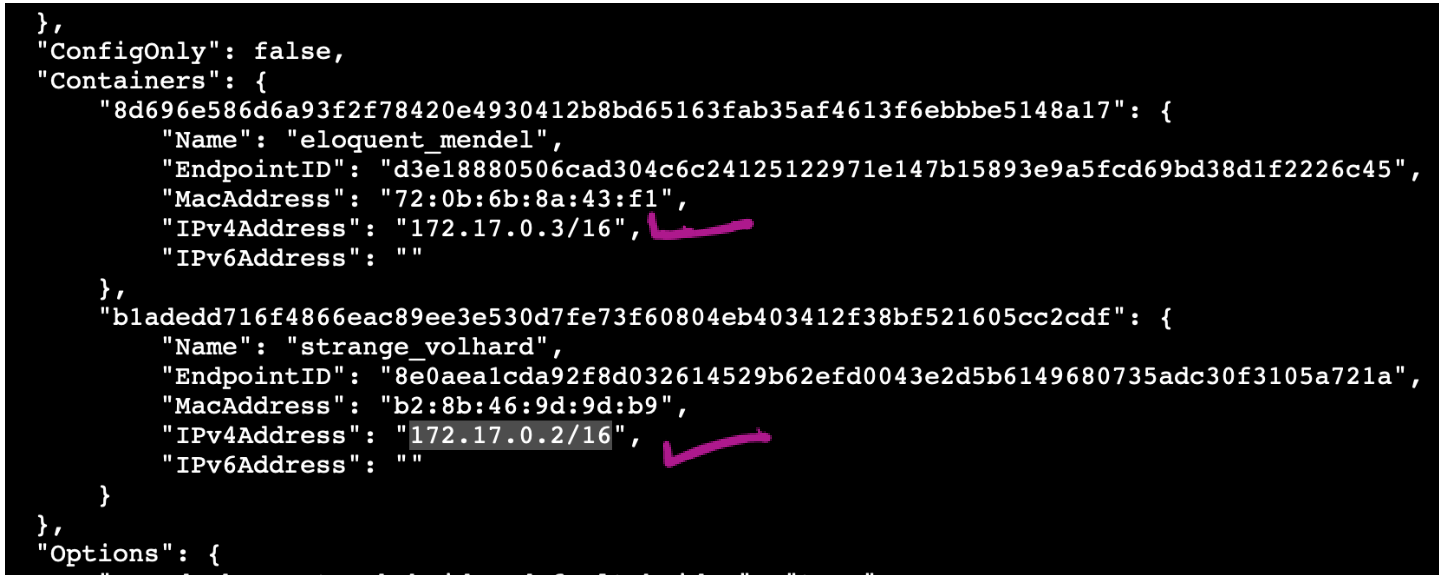
```
root@ip-172-31-7-2:/home/ubuntu# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
82f565c90f94   bridge    bridge    local
7bdedff79e58   host      host      local
9b8a198d97c8   none      null      local
```

- If you want to see the containers in bridge network :



```
root@ip-172-31-0-170:/home/ubuntu# docker inspect bridge
[
```

With above command you will be able to see the ip address of the container as well



```
    },
    "ConfigOnly": false,
    "Containers": {
        "8d696e586d6a93f2f78420e4930412b8bd65163fab35af4613f6ebbbe5148a17": {
            "Name": "eloquent_mendel",
            "EndpointID": "d3e18880506cad304c6c24125122971e147b15893e9a5fcd69bd38d1f2226c45",
            "MacAddress": "72:0b:6b:8a:43:f1",
            "IPv4Address": "172.17.0.3/16",
            "IPv6Address": ""
        },
        "b1adedd716f4866eac89ee3e530d7fe73f60804eb403412f38bf521605cc2cdf": {
            "Name": "strange_volhard",
            "EndpointID": "8e0aea1cda92f8d032614529b62efd0043e2d5b6149680735adc30f3105a721a",
            "MacAddress": "b2:8b:46:9d:9d:b9",
            "IPv4Address": "172.17.0.2/16",
            "IPv6Address": ""
        }
    },
    "Options": {
```

Documentation for Docker network:
https://drive.google.com/file/d/13hyjXW4h3eOJheH5QbZOBWi2ACOnjye4/view?usp=sharing

Summary

- Concept | Purpose
- Container | Runs isolated application instance
- Volume | Stores persistent data
- Port Expose | Enables external access
- Inspect | Displays detailed metadata
- Dockerfile | Defines image build steps
- Network | Connects containers together

Thank You

- Contact: Akshat Gupta
- Topic Recap: Containers, Volumes, Ports, Inspect, Dockerfile, Networks