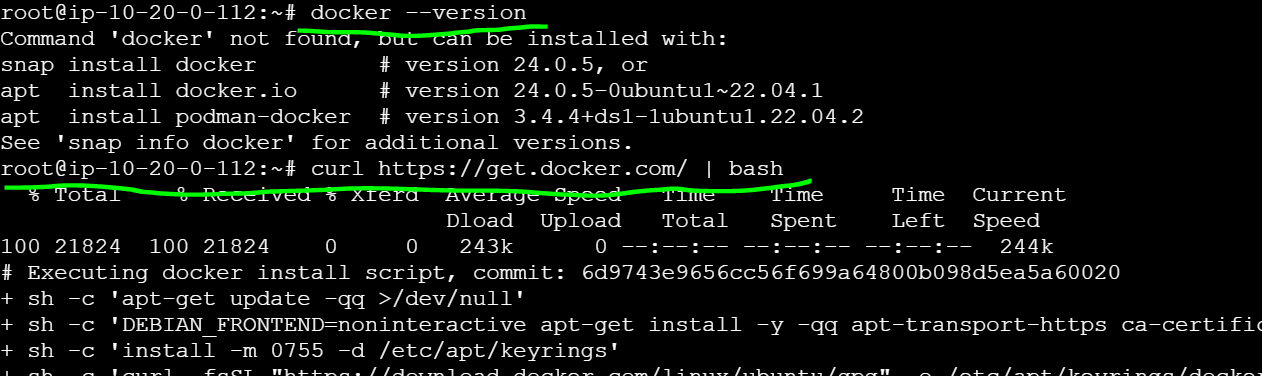
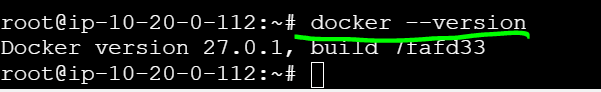
#Install tools

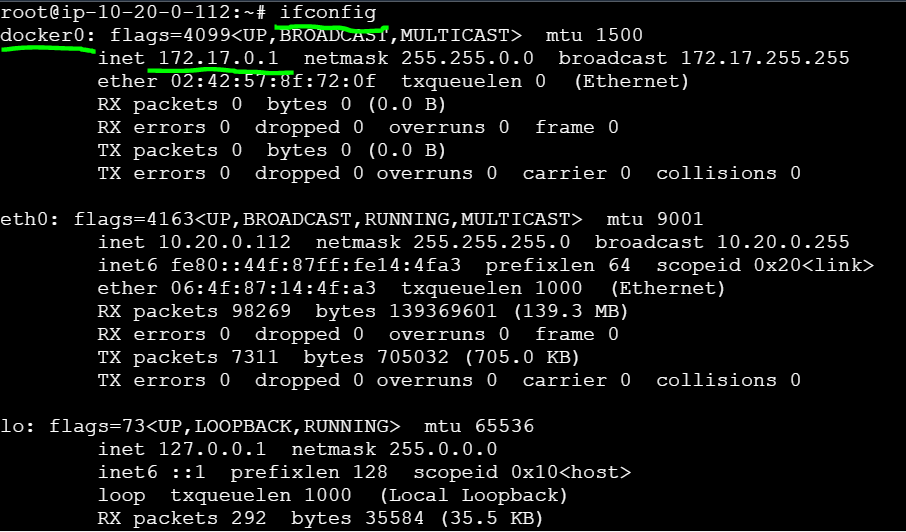
apt update && apt install jq net-tools

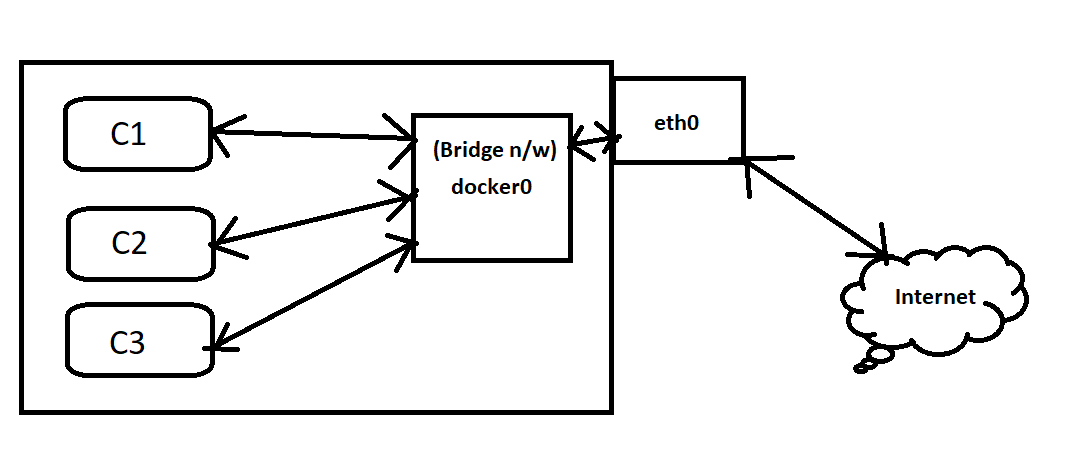
#Docker Installation

curl https://get.docker.com/ |bash

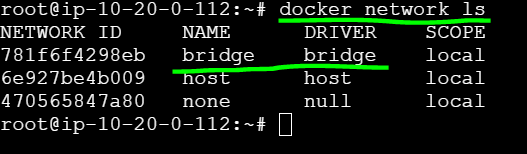


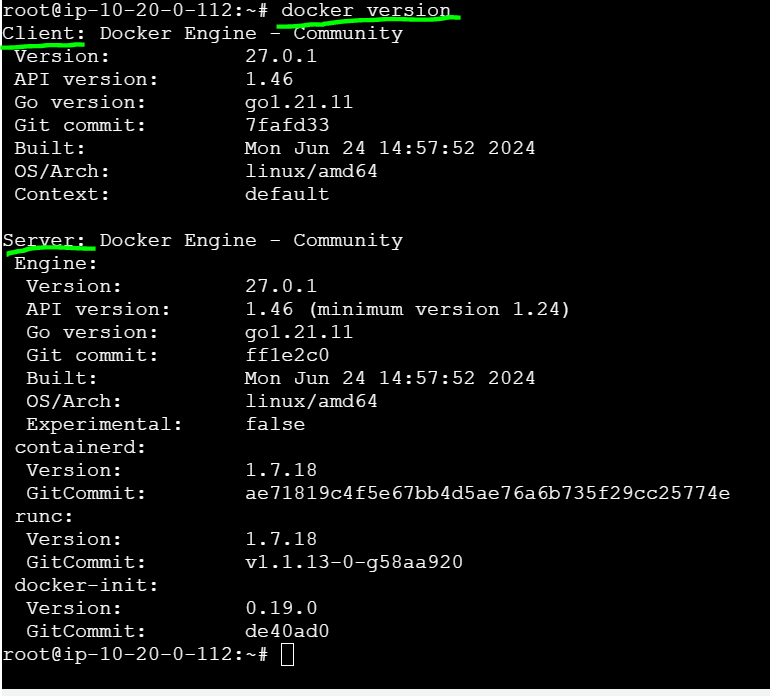


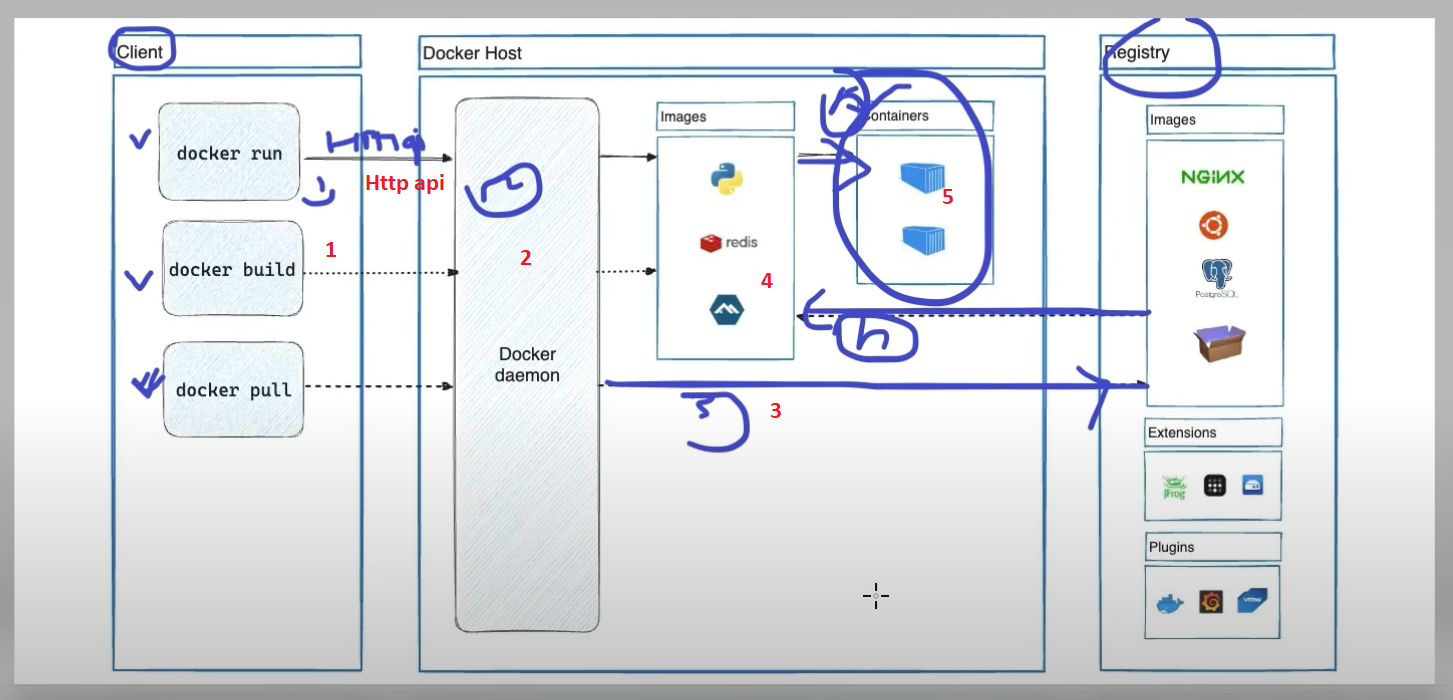




**Docker0 (172.17.0.1) acts as a Bridge N/W between containers & eth0**







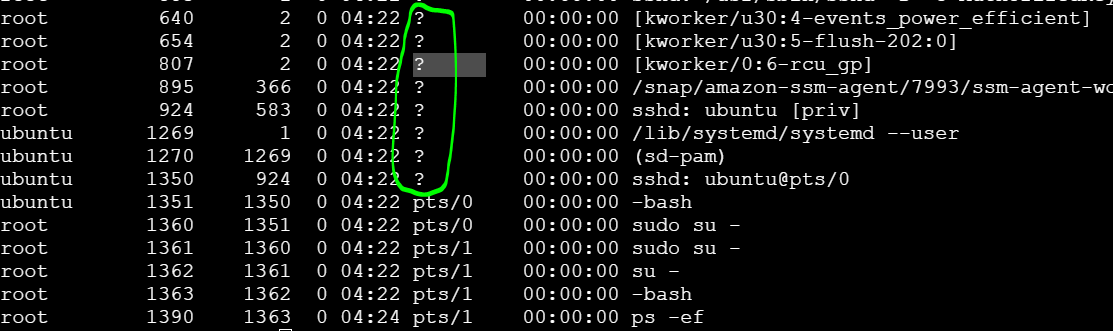
**Process of creating containers**

Docker Run / Build / Pull 🡪 Docker daemon 🡪Registry 🡪 image download 🡪 Container

Docker daemon -- > is a process ( ps -ef you will get the results with ? )

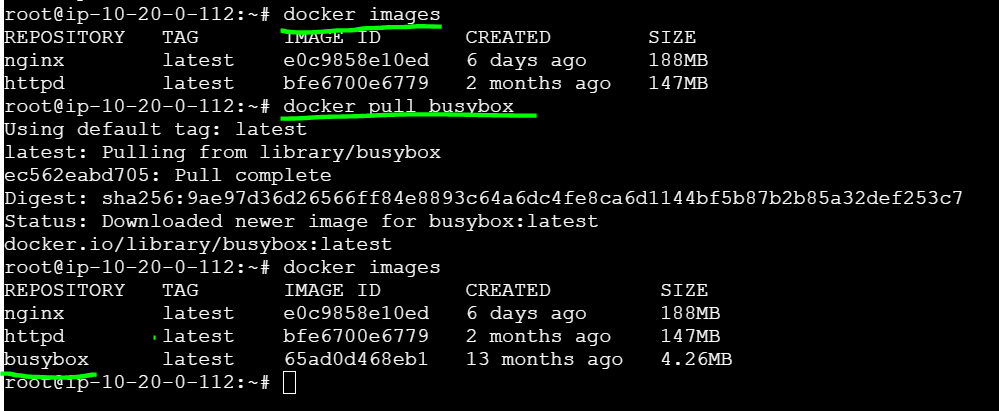
? 🡪 Docker daemon

pts/1 🡪 user



**Downloading images**

Docker pull nginx/busybox/httpd

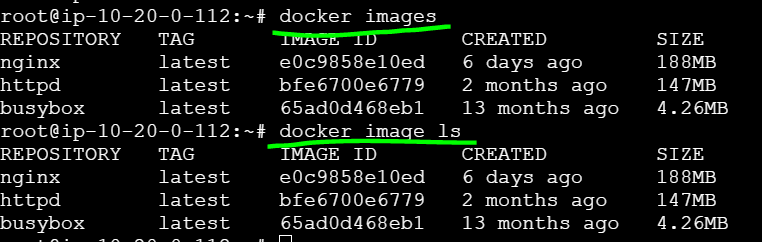


#list available images

docker images



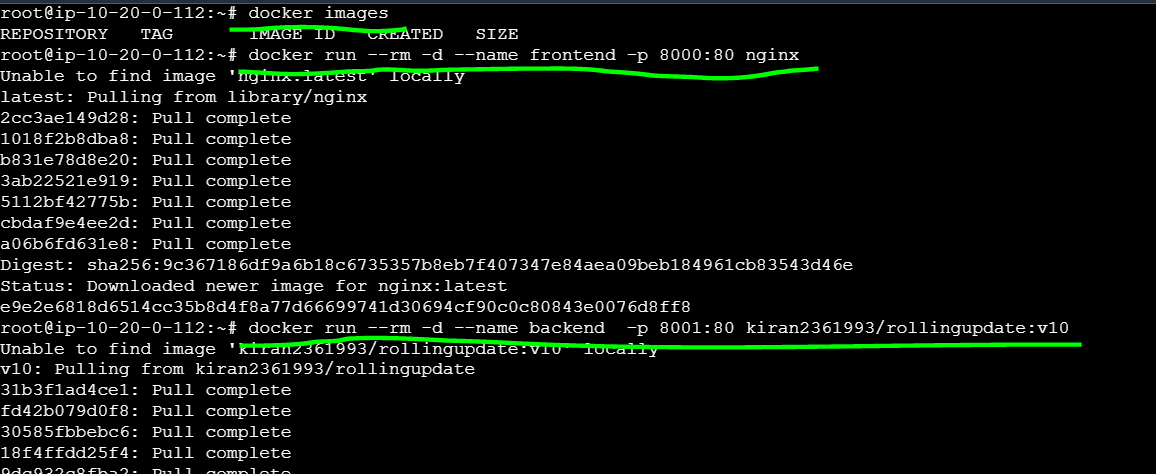


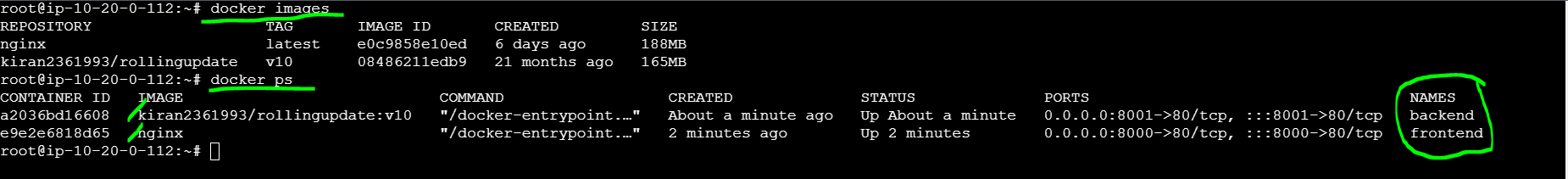


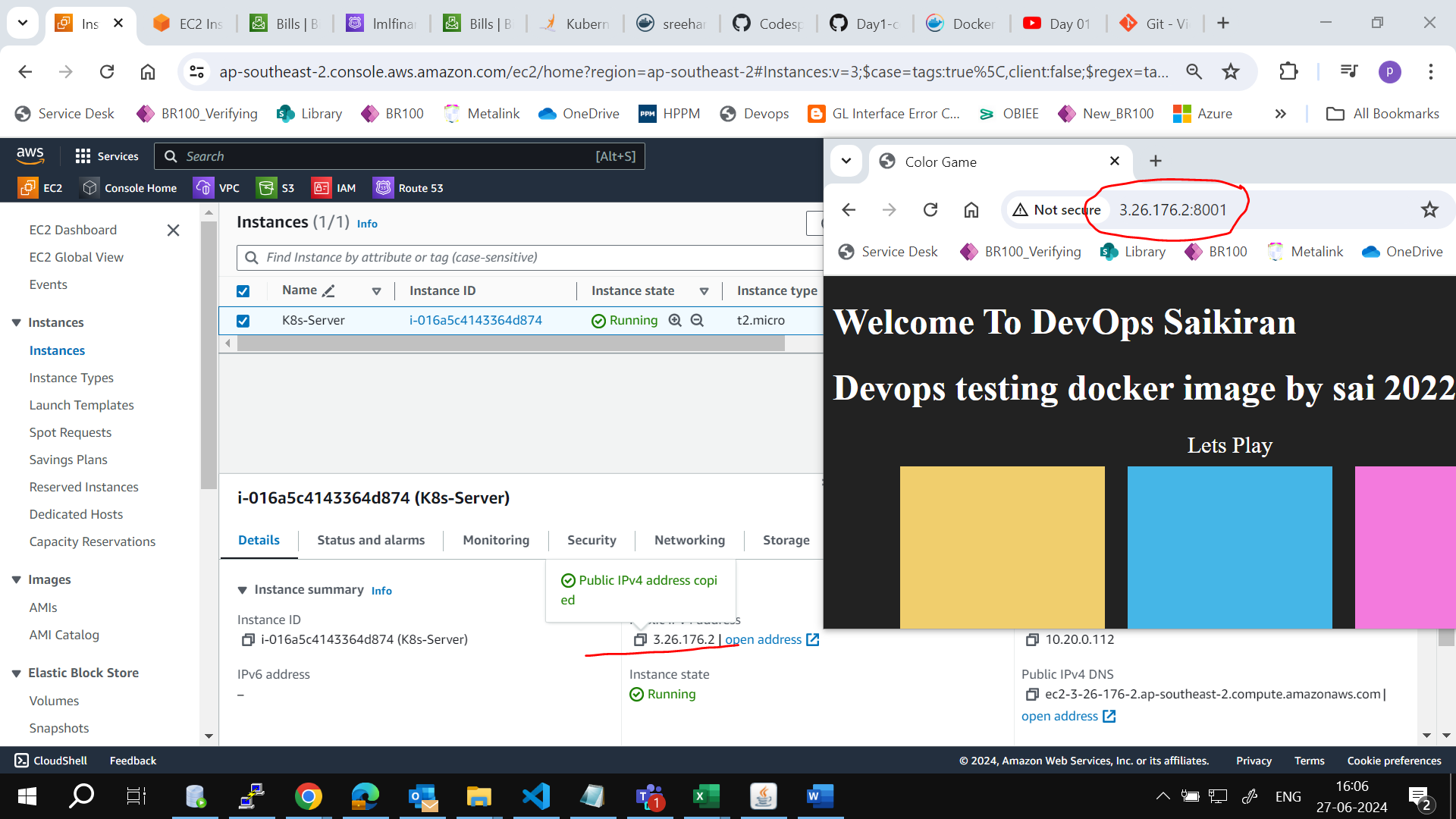
Port-forwarding

docker run --rm -d --name frontend -p 8000:80 nginx

docker run --rm -d --name backend -p 8001:80 kiran2361993/rollingupdate:v10







#containers path

cd /var/lib/docker/containers/

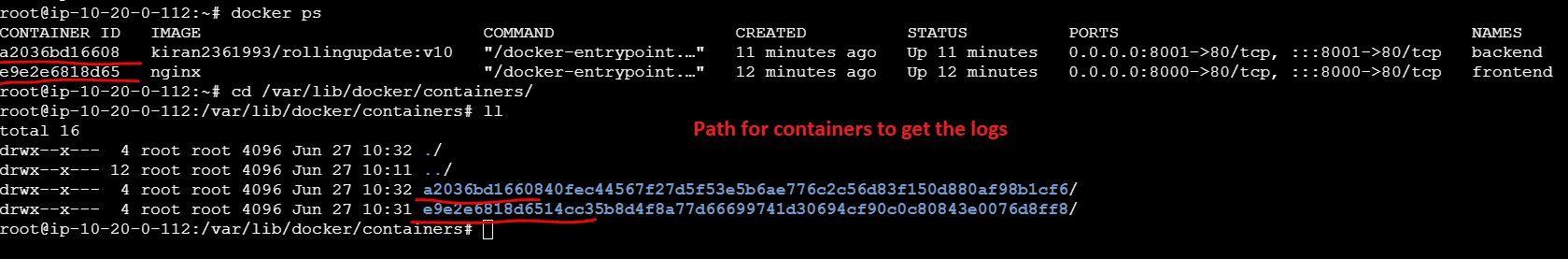
#Once containers are created login to it & get the logs

cd /var/lib/docker/containers/<container\_id>

ll

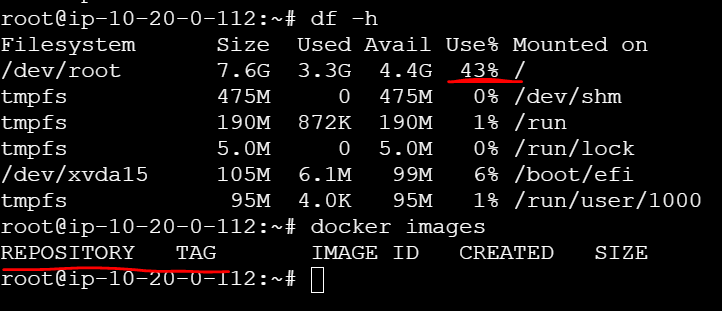
cat a2036bd1660840fec44567f27d5f53e5b6ae776c2c56d83f150d880af98b1cf6-json.log | jq

cat a2036bd1660840fec44567f27d5f53e5b6ae776c2c56d83f150d880af98b1cf6-json.log | jq > INC01478528.log



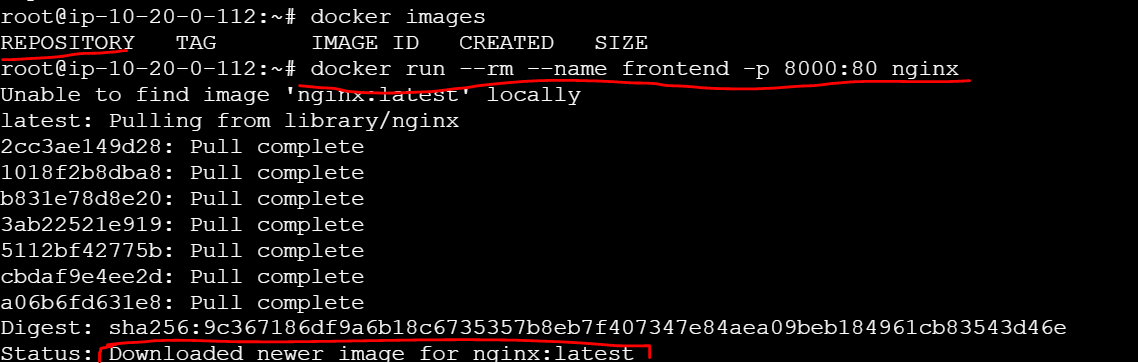
Day 02 Docker Dir Change | Custom Network | Host | AWS EBS Volume |

**Before without images – size usage of the root volume is 43%**

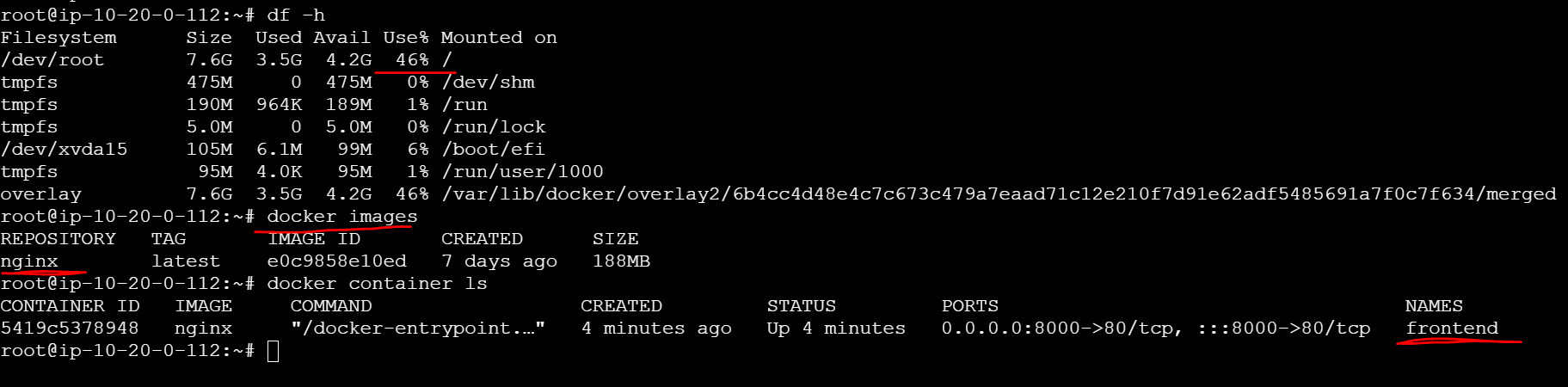


Download image

docker run --rm --name frontend -p 8000:80 nginx



**After downloading the images -- size usage of the root volume is 46%**



#check the current root volume usage  (43%)

df -h

#download image/create a container & check the current root volume usage (46%)

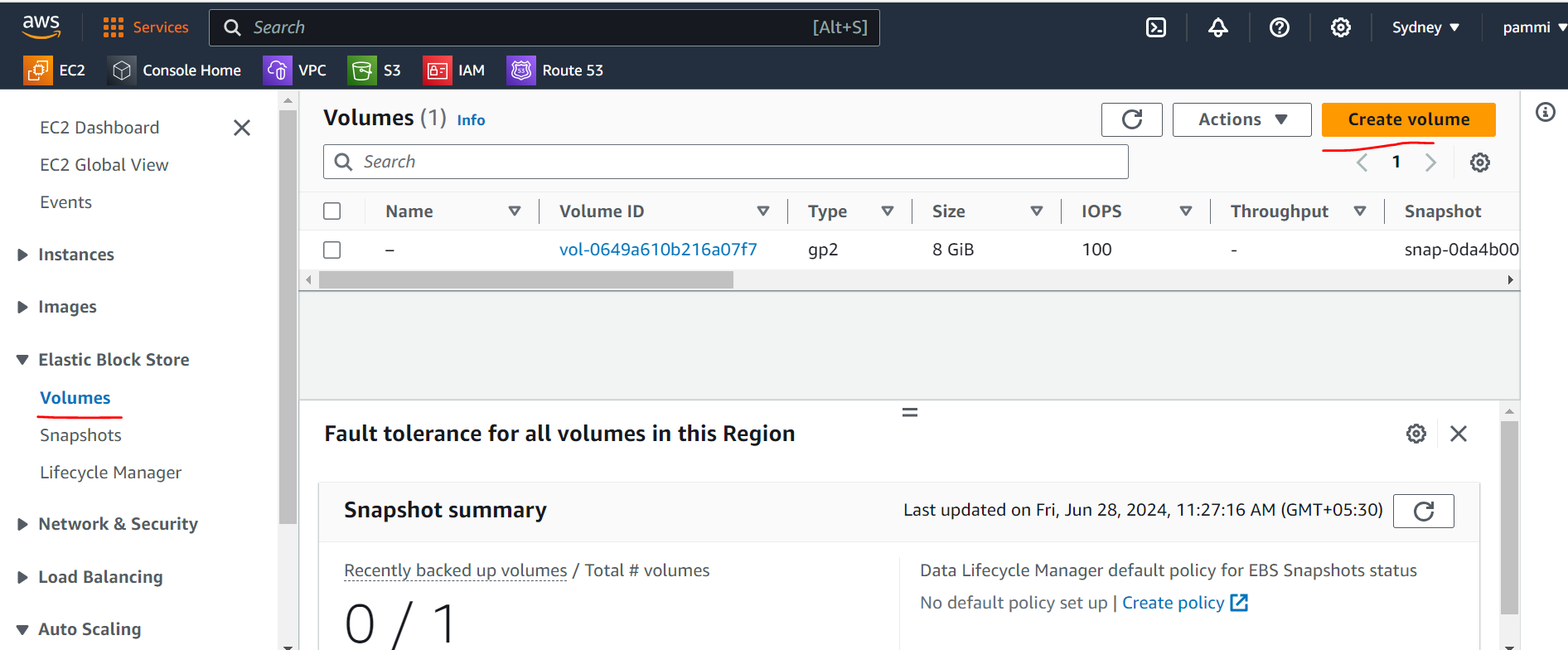
docker run --rm --name frontend -p 8000:80 nginx

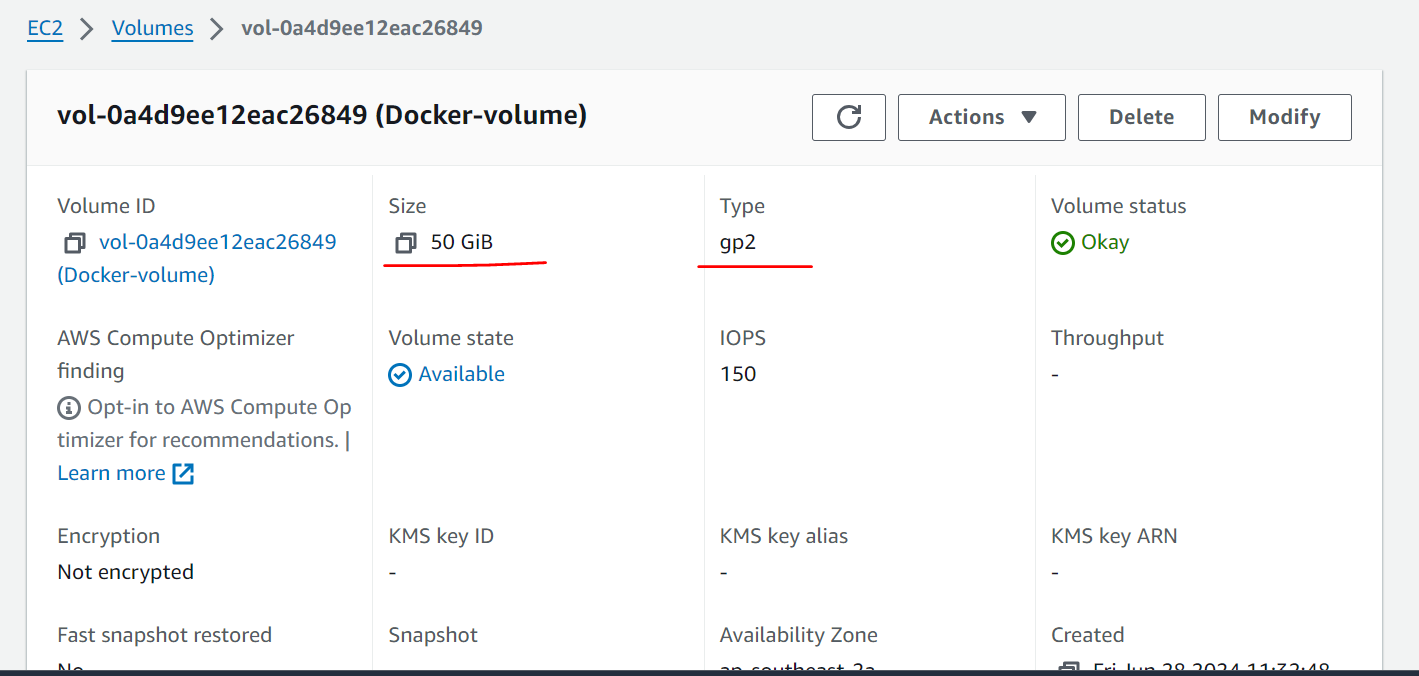
df -h

#if you keep on downloading images/creating containers the root volume reaches to 100%

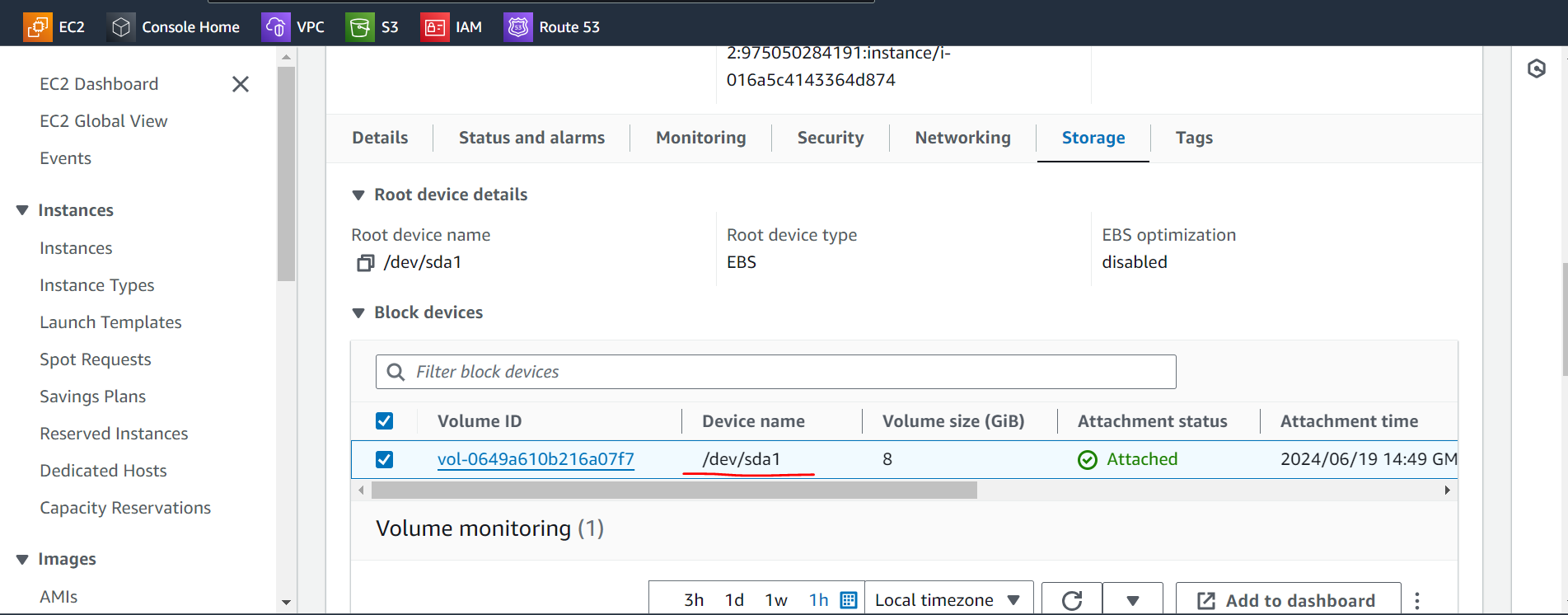
**#Process to change the docker directory**

**Volume creation**

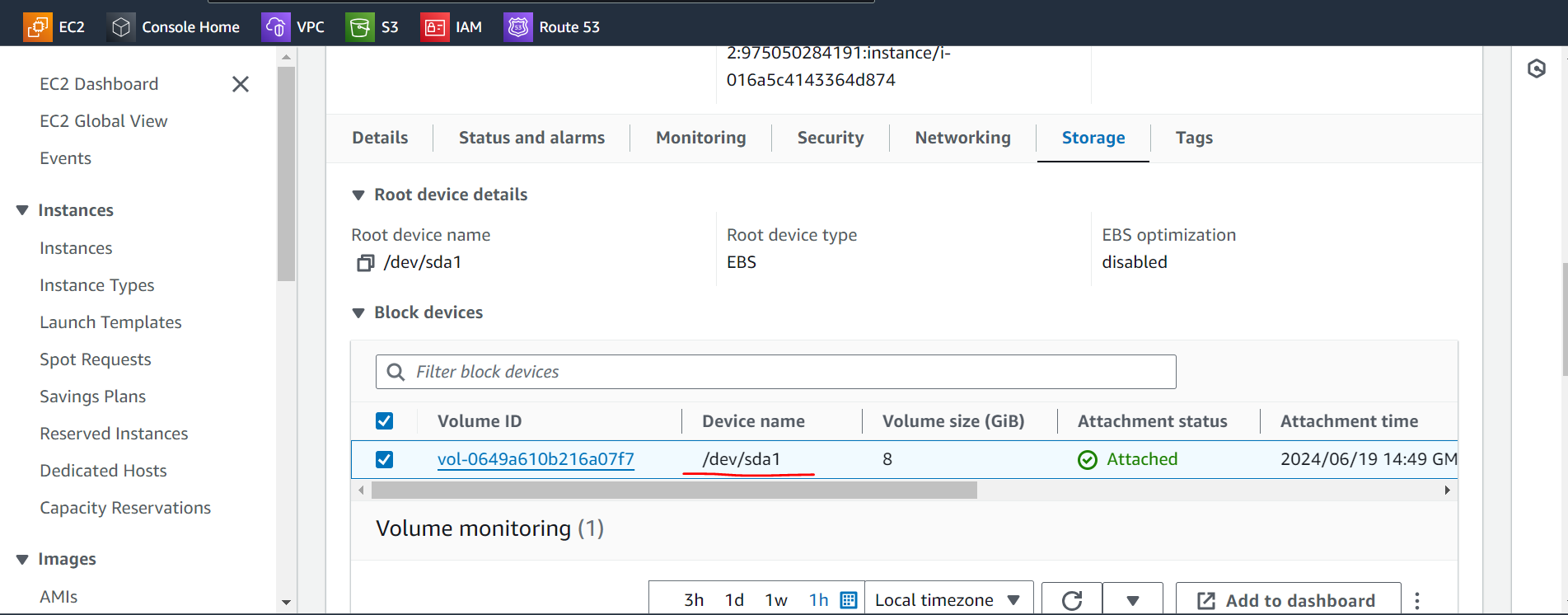




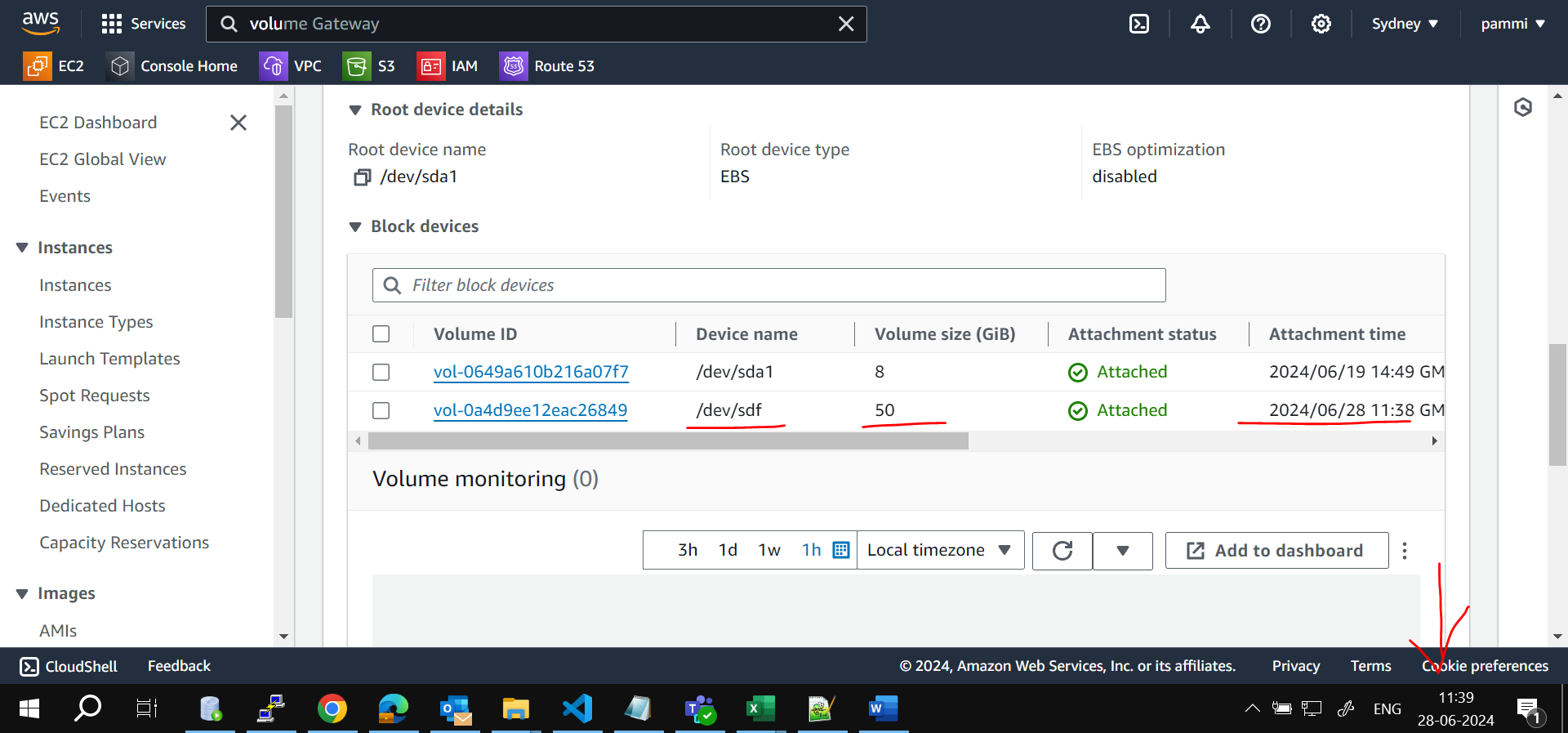
**Attach newly created volume to the instance.**



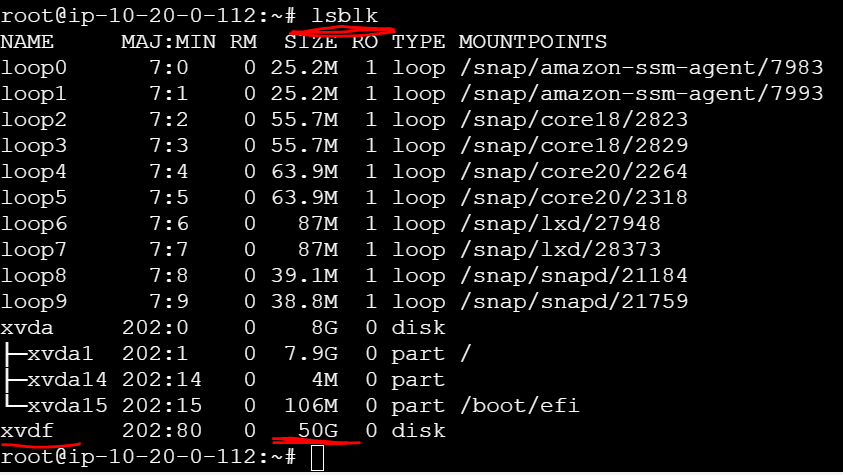
**Before Attaching volume to an instance**



**After attaching a volume to an instance**

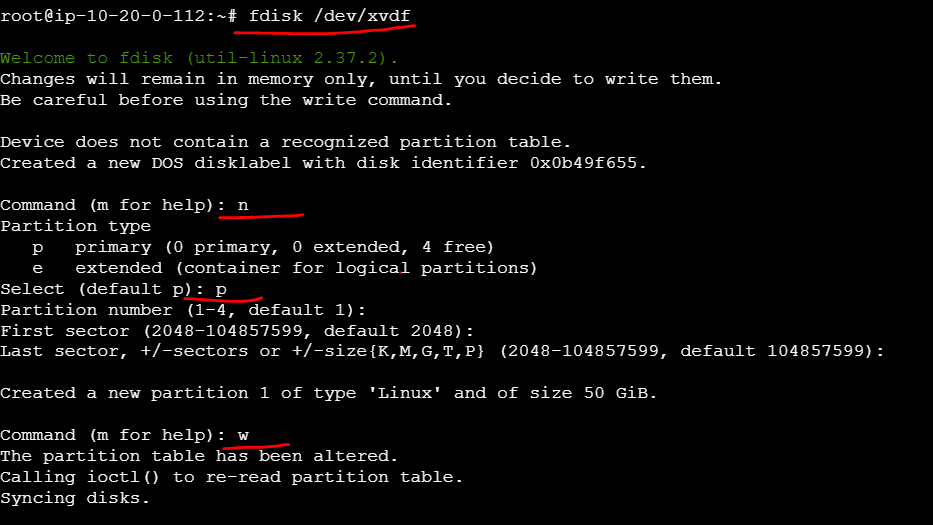


**Validate new volume in linux**

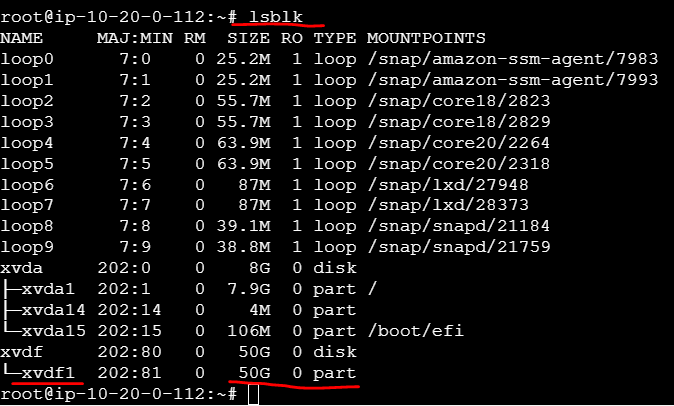


**Format the newly created volume (xvdf)**

**fdisk /dev/xvdf**



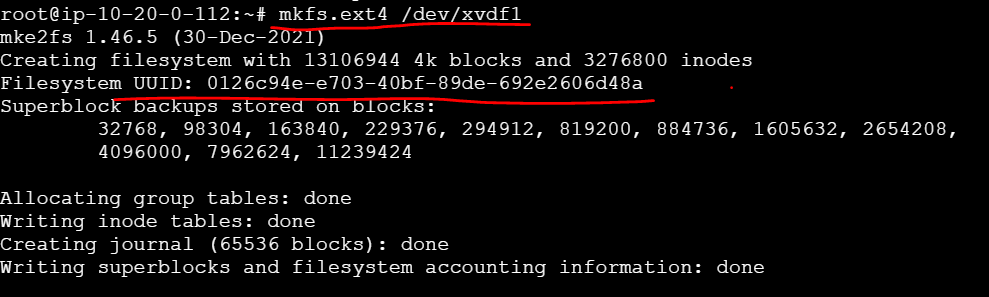
**Validate formatted volume.**



**Create a file system for the newly created volume (/dev/xvdf1)**

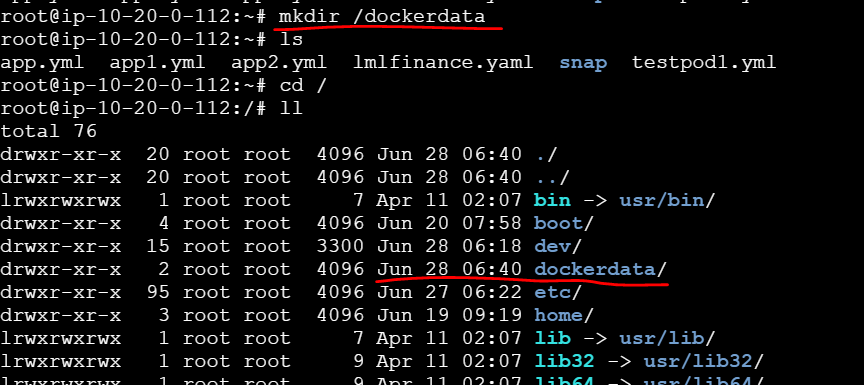
**mkfs.ext4 /dev/xvdf1**

**copy Filesystem UUID: 0126c94e-e703-40bf-89de-692e2606d48a**



**Create directory**

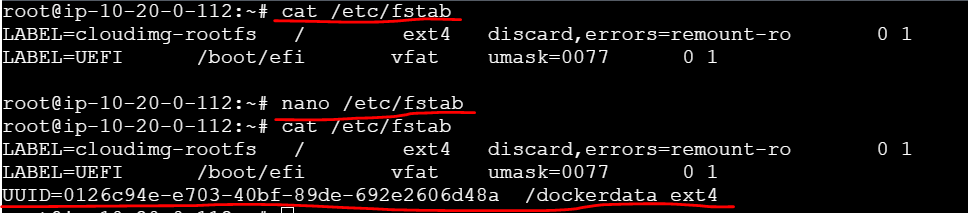
**mkdir /dockerdata**



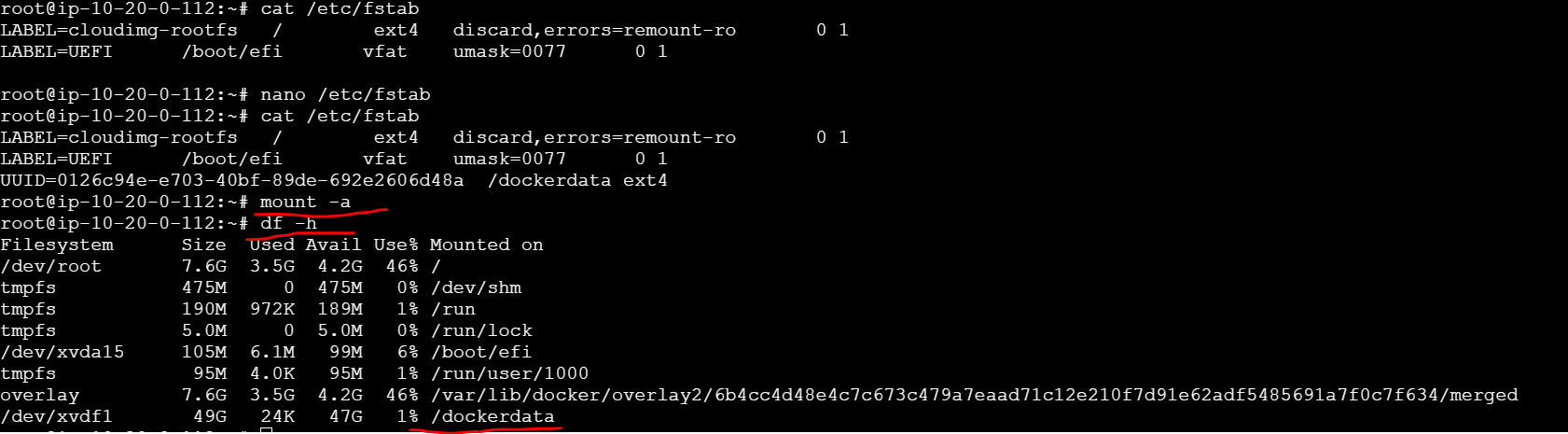
**Add Filesystem UUID: 0126c94e-e703-40bf-89de-692e2606d48a into the below path**

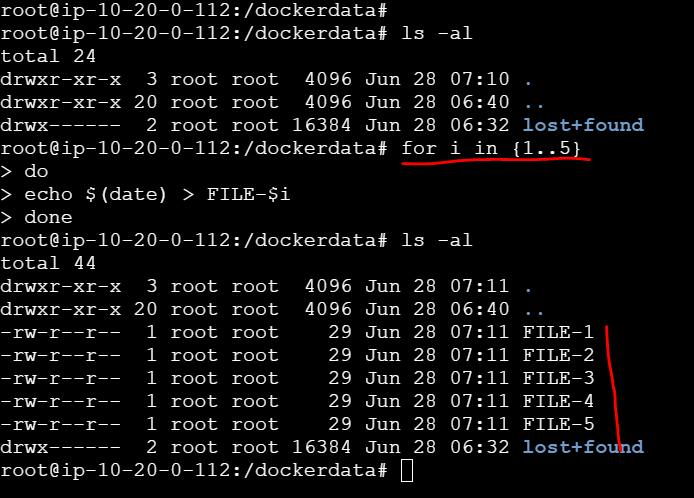
**nano /etc/fstab**

UUID=0126c94e-e703-40bf-89de-692e2606d48a  /dockerdata ext4



**mount -a**





**Change service directory file**

**Create additional EBS Volume , fdisk and mount it.**

**sudo systemctl stop docker.service**

**sudo systemctl stop docker.socket**

**sudo nano /lib/systemd/system/docker.service Add the following line with the custom directo**

**#ExecStart=/usr/bin/dockerd -H fd:// -- containerd=/run/containerd/containerd.sock**

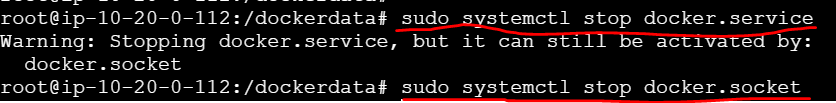
**# ExecStart=/usr/bin/dockerd --data-root /dockerdata -H fd:// -- containerd=/run/containerd/containerd. sock**

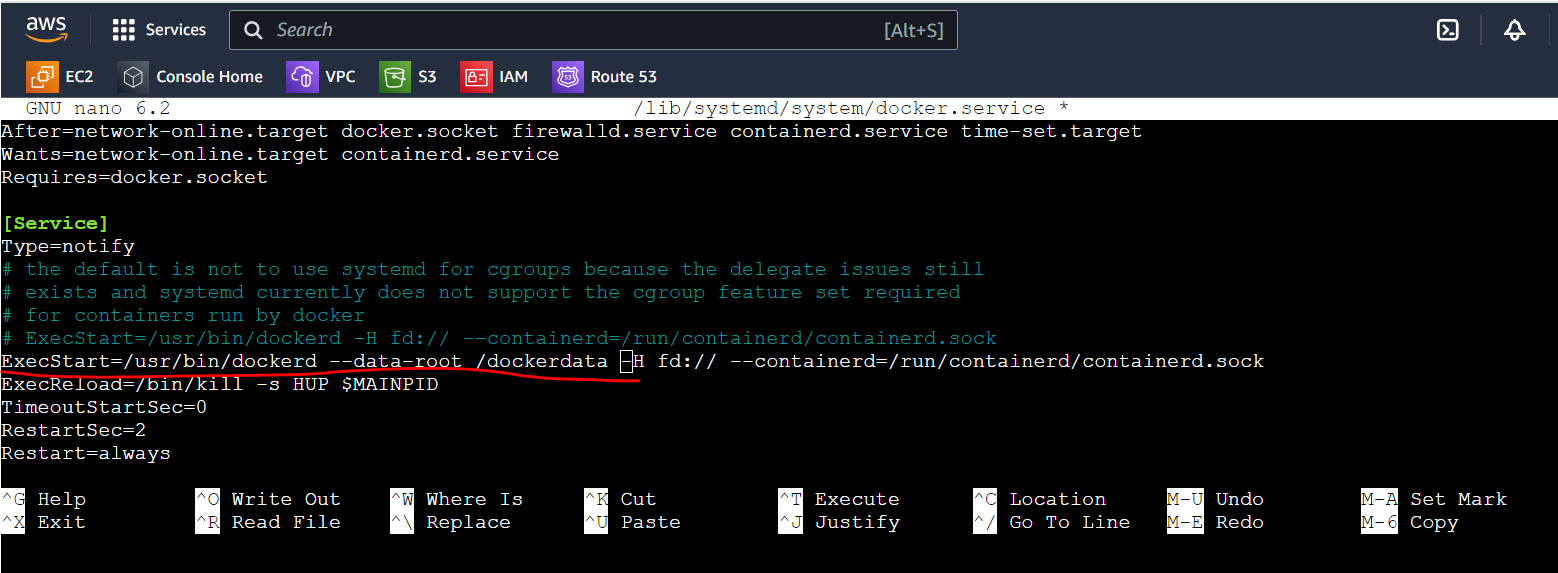
**sudo rsync -aqxP /var/lib/docker/ /dockerdata**

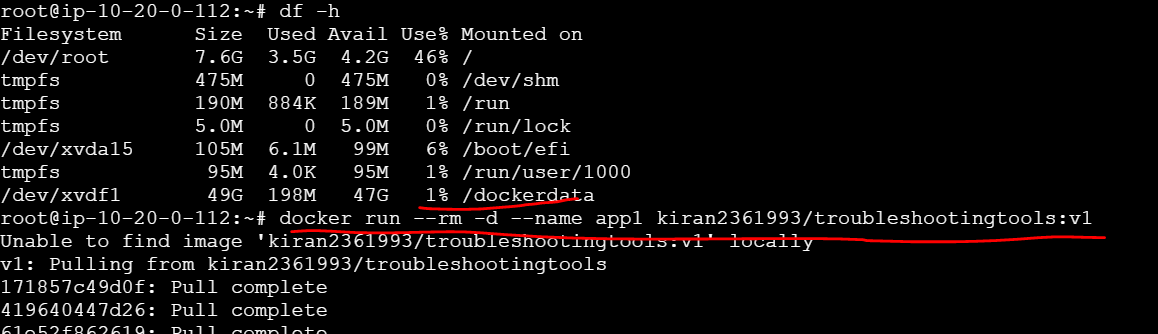
**sudo systemctl daemon-reload && sudo systemctl start docker**

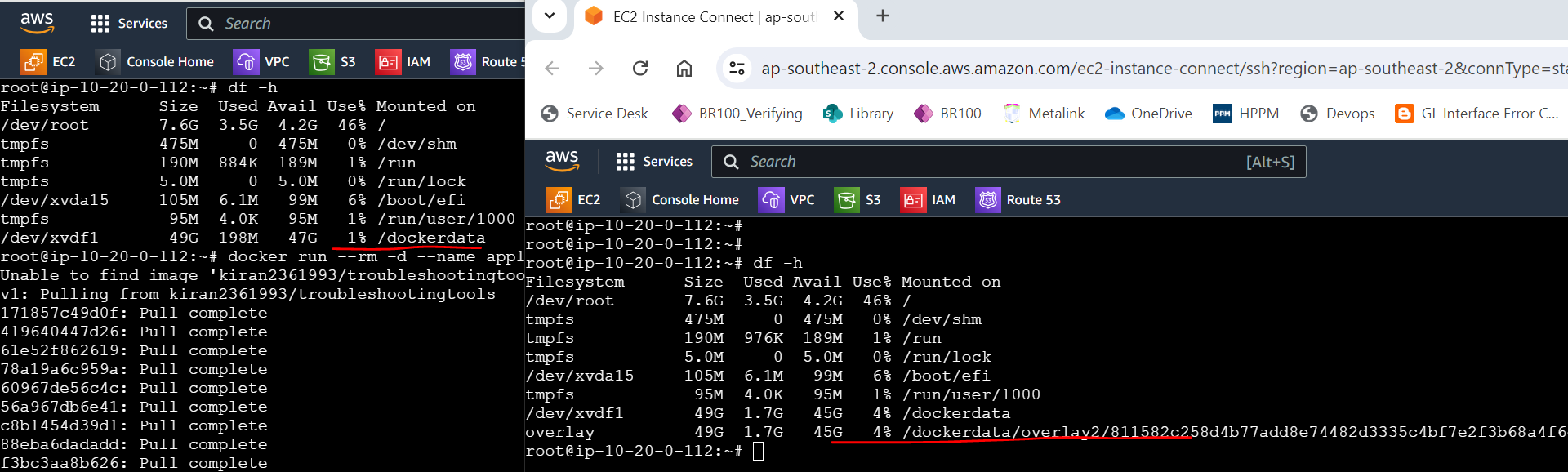
**sudo systemctl status docker --no-pager**

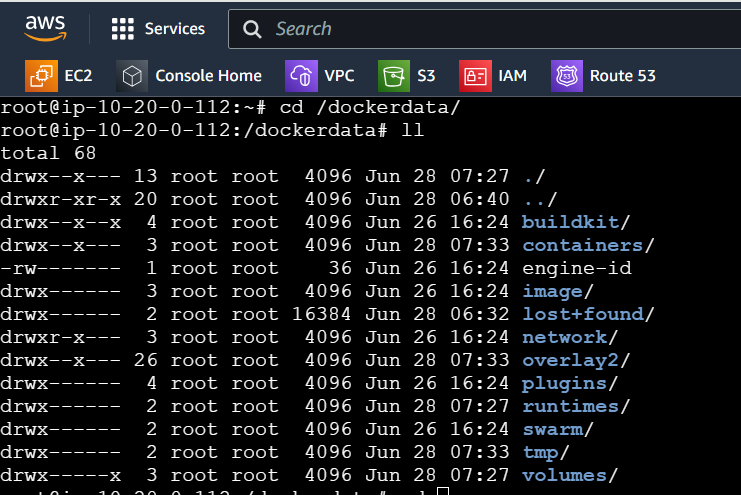
**ps aux | grep -i docker | grep -v grep**

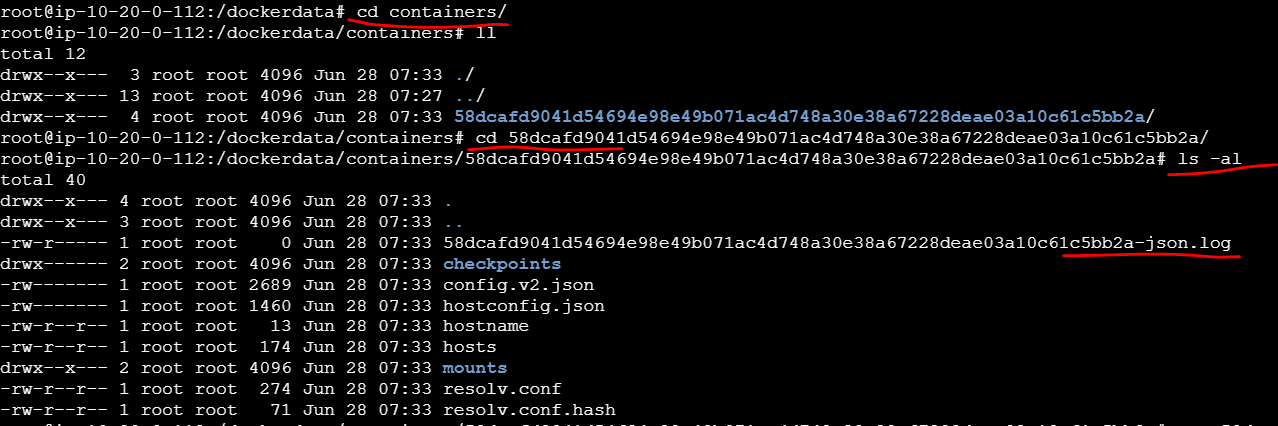








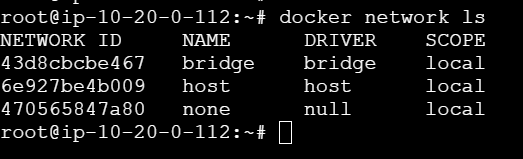




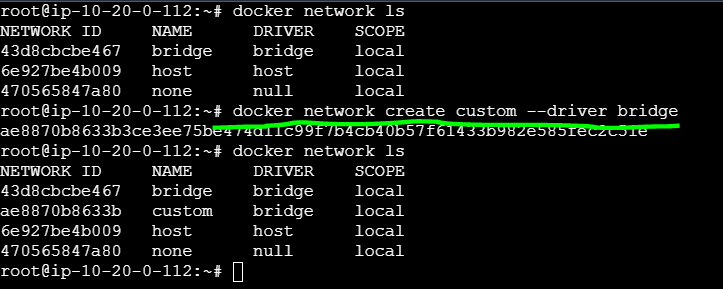
**Docker Network**

#Docker Network

docker network ls

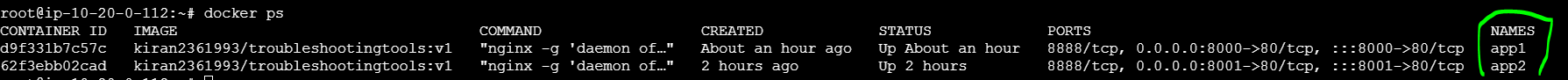


**Custom network creation**

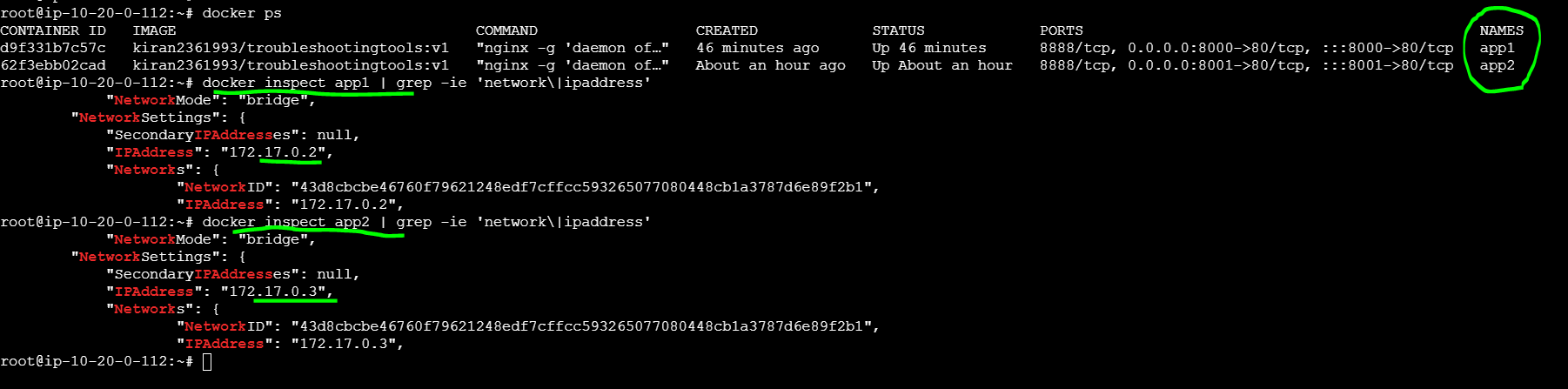


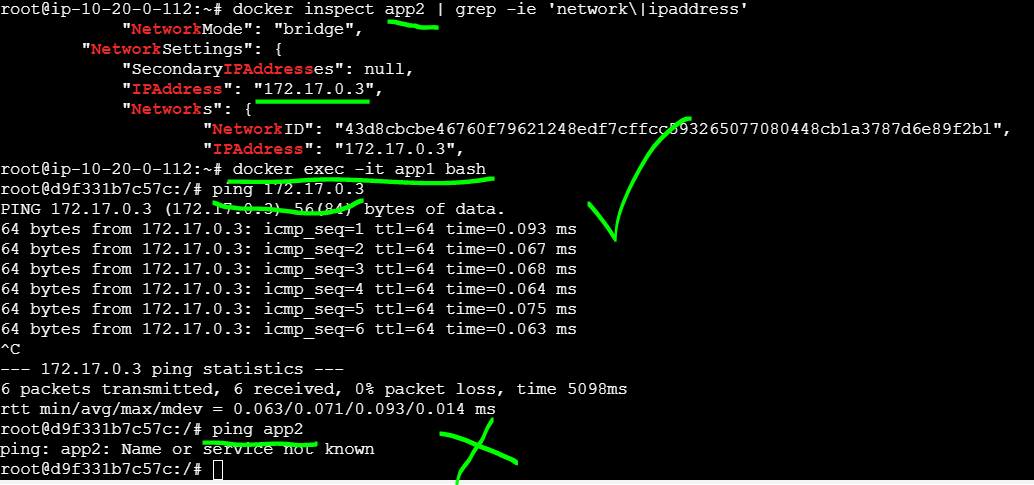
**Check the running containers & get the ipaddress & network details**

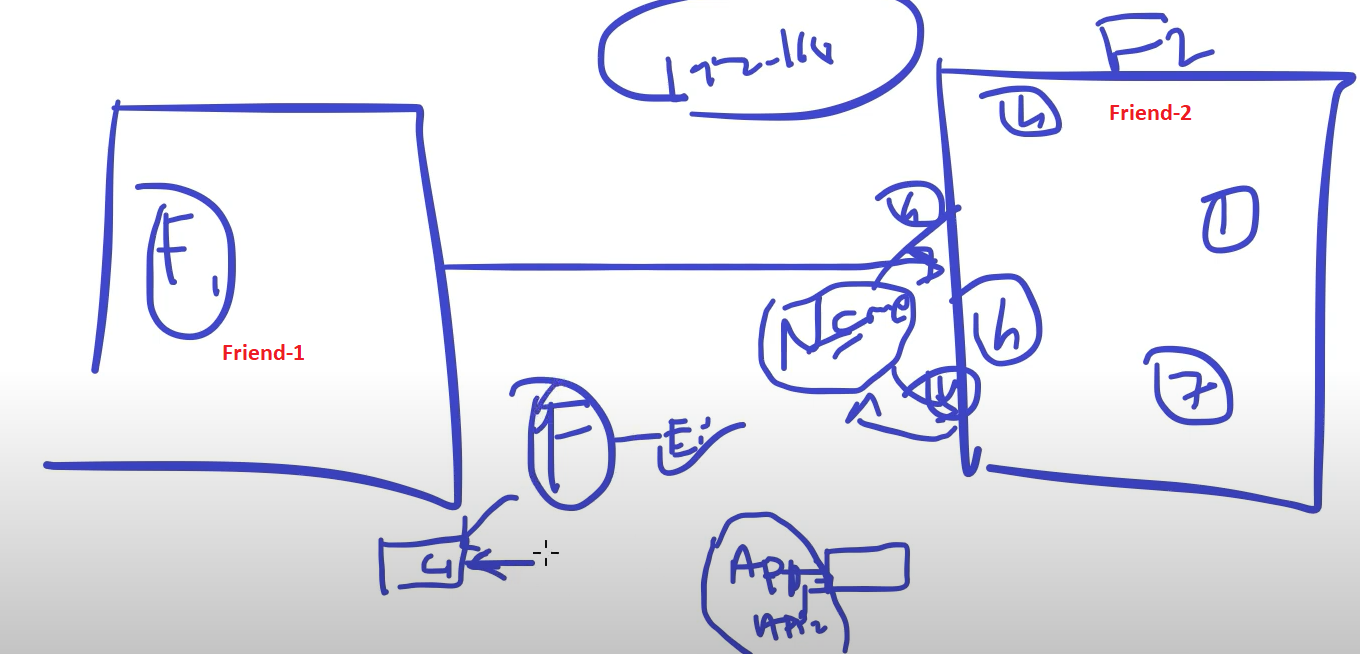
Docker ps



docker inspect app1 | grep -ie 'network\|ipaddress'

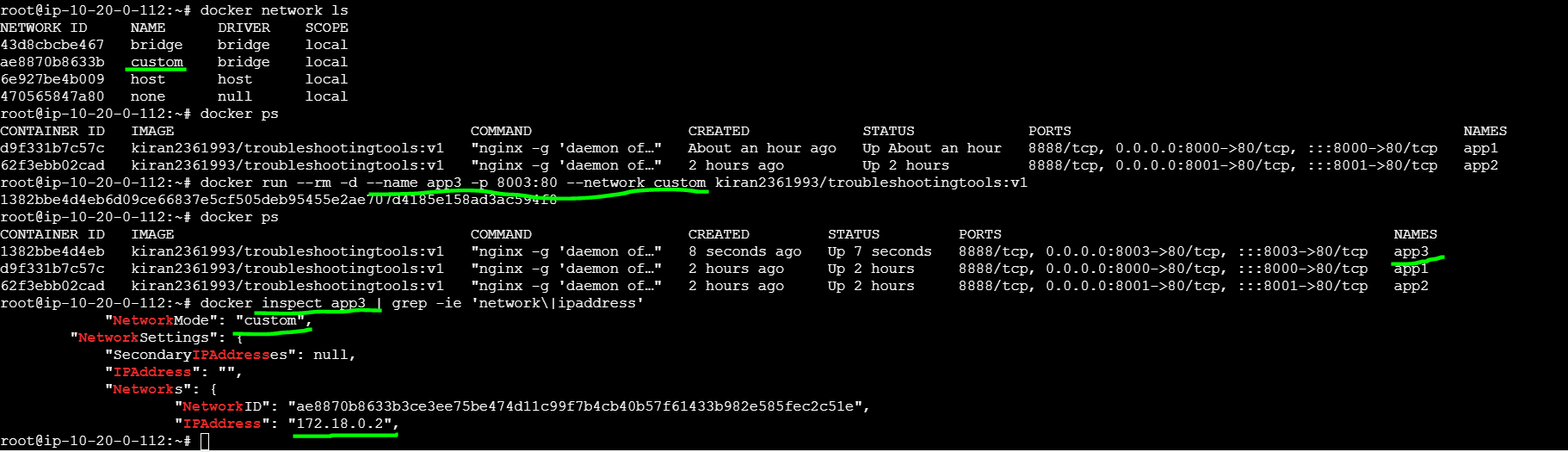




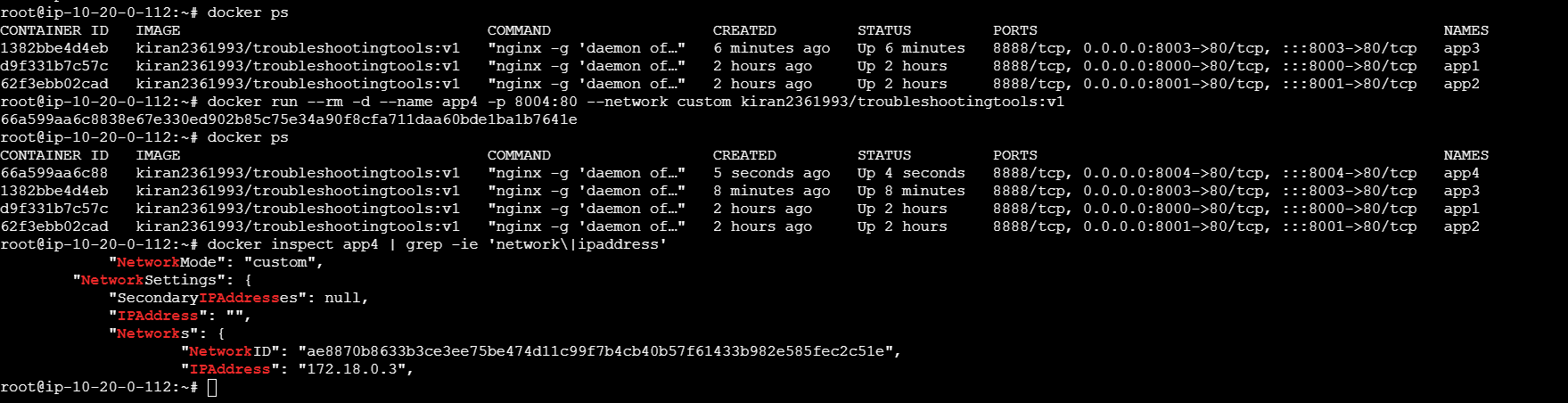


**Create container with custom network**

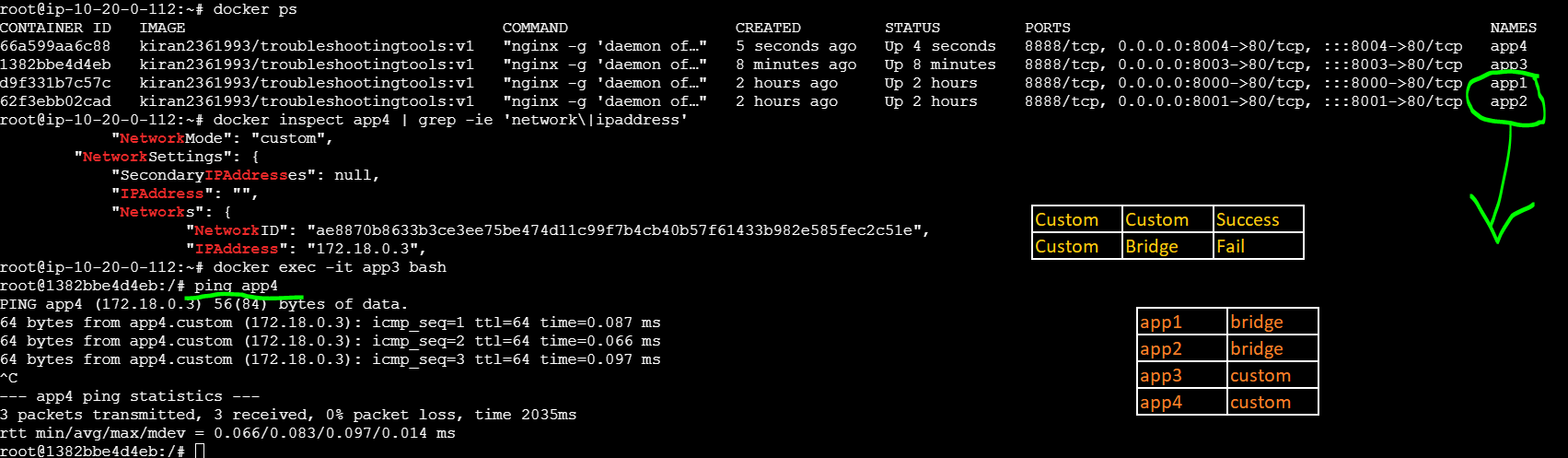
**docker inspect app3 | grep -ie 'network\|ipaddress'**



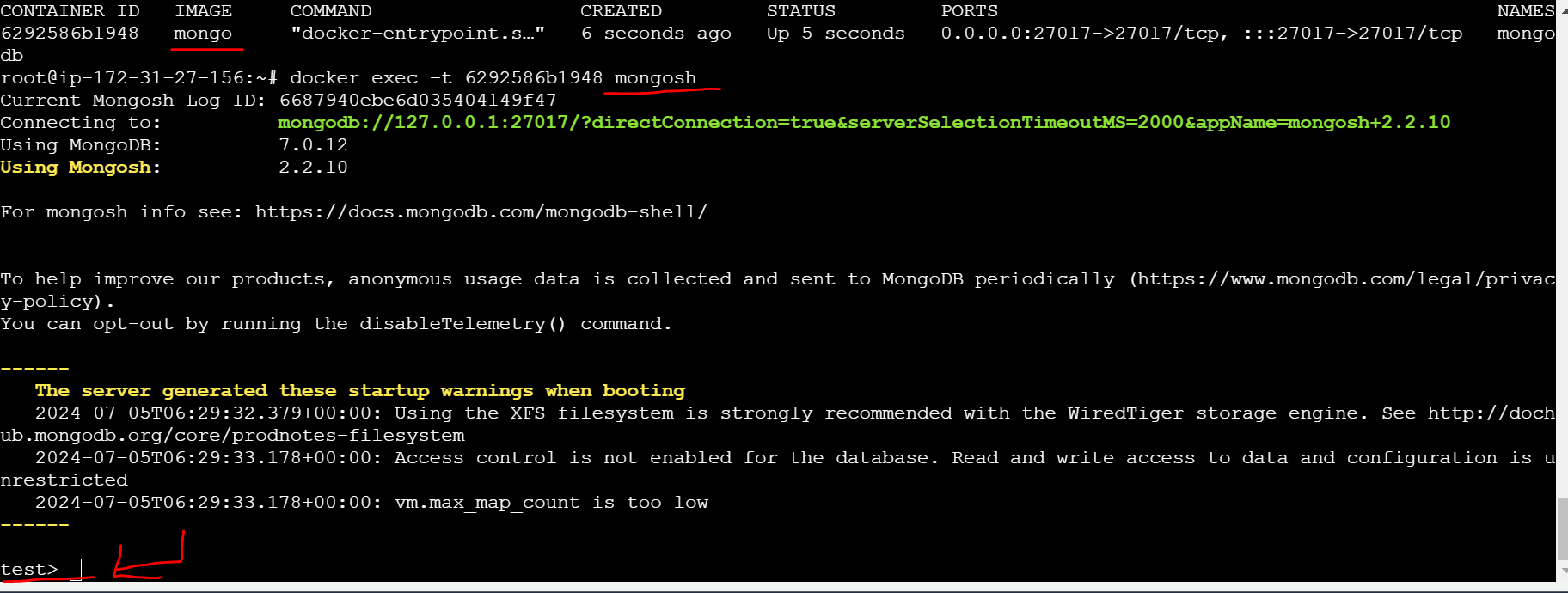
**docker inspect app4 | grep -ie 'network\|ipaddress'**



**Created 2 custom networks & 2 bridge networks**



**DAY-3 -- Docker Volumes | BindMounts | Portainer |**

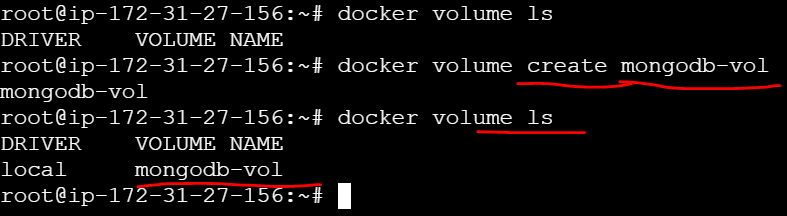


**#list volumes**

docker volume ls

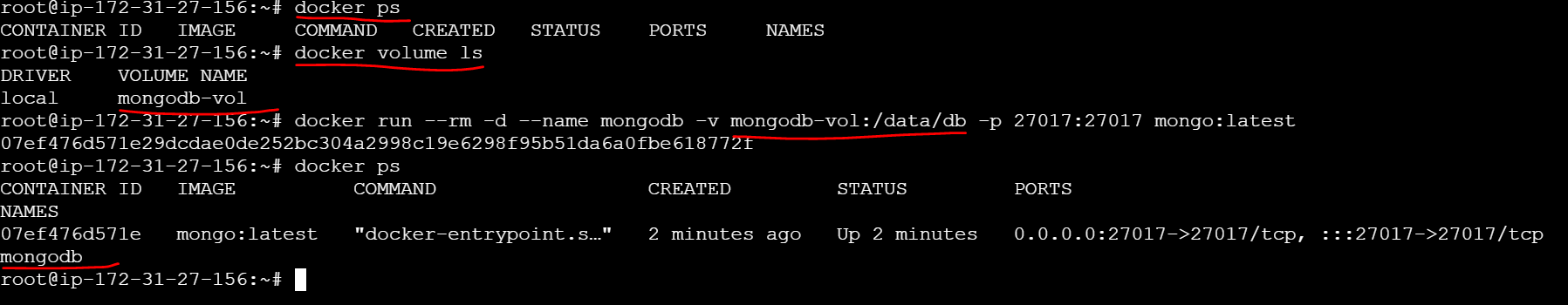
**#create volume**

docker volume create mongodb-vol



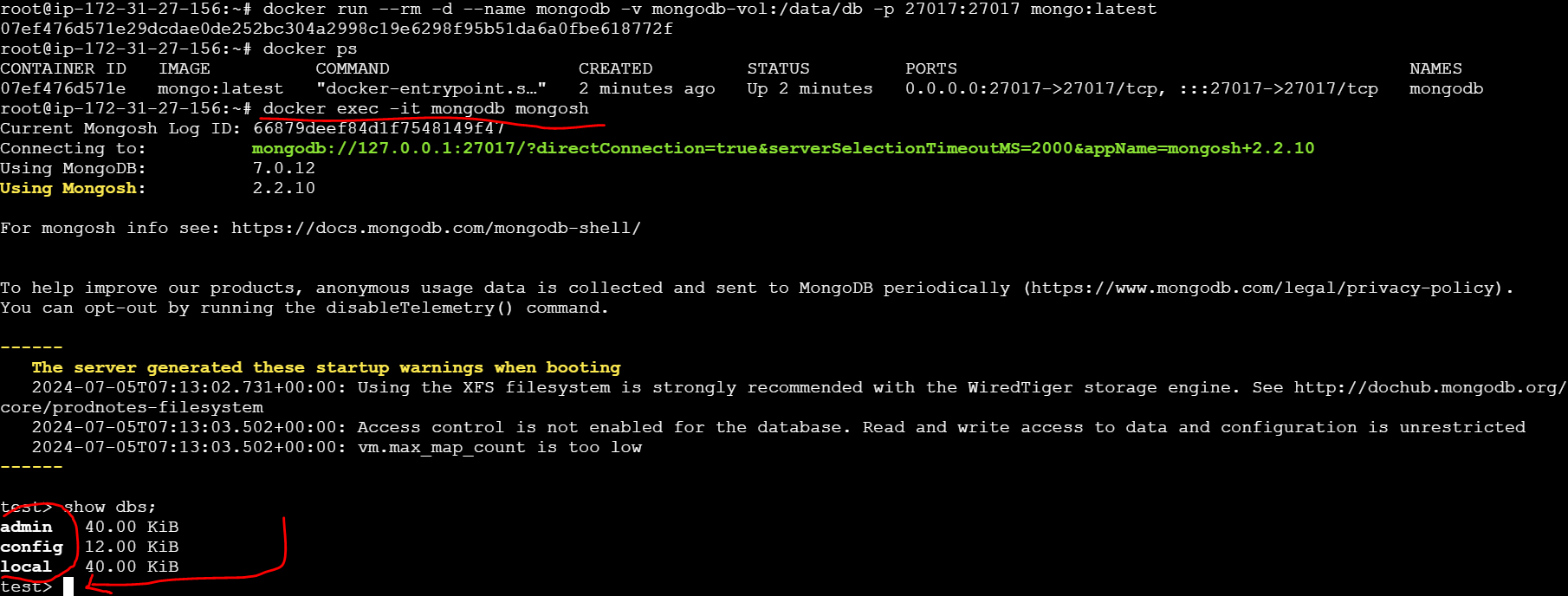
**#Create container & attach volume to it**

docker run --rm -d --name mongodb -v mongodb-vol:/data/db -p 27017:27017 mongo:latest



**#command to enter into Database container**

docker exec -it <Image\_name/container\_id> mongosh



**#Insert sample date into database**

db.helo.insertMany([

{ "\_id" : 1, "name" : "Matt", "status": "active", "level": 12, "score":202},

{ "\_id" : 2, "name" : "Frank", "status": "inactive", "level": 2, "score":9},

{ "\_id" : 3, "name" : "Karen", "status": "active", "level": 7, "score":87},

{ "\_id" : 4, "name" : "Katie", "status": "active", "level": 3, "score":27, "status": "married", "emp": "yes", "kids": 3},

{ "\_id" : 5, "name" : "Matt1", "status": "active", "level": 12, "score":202},

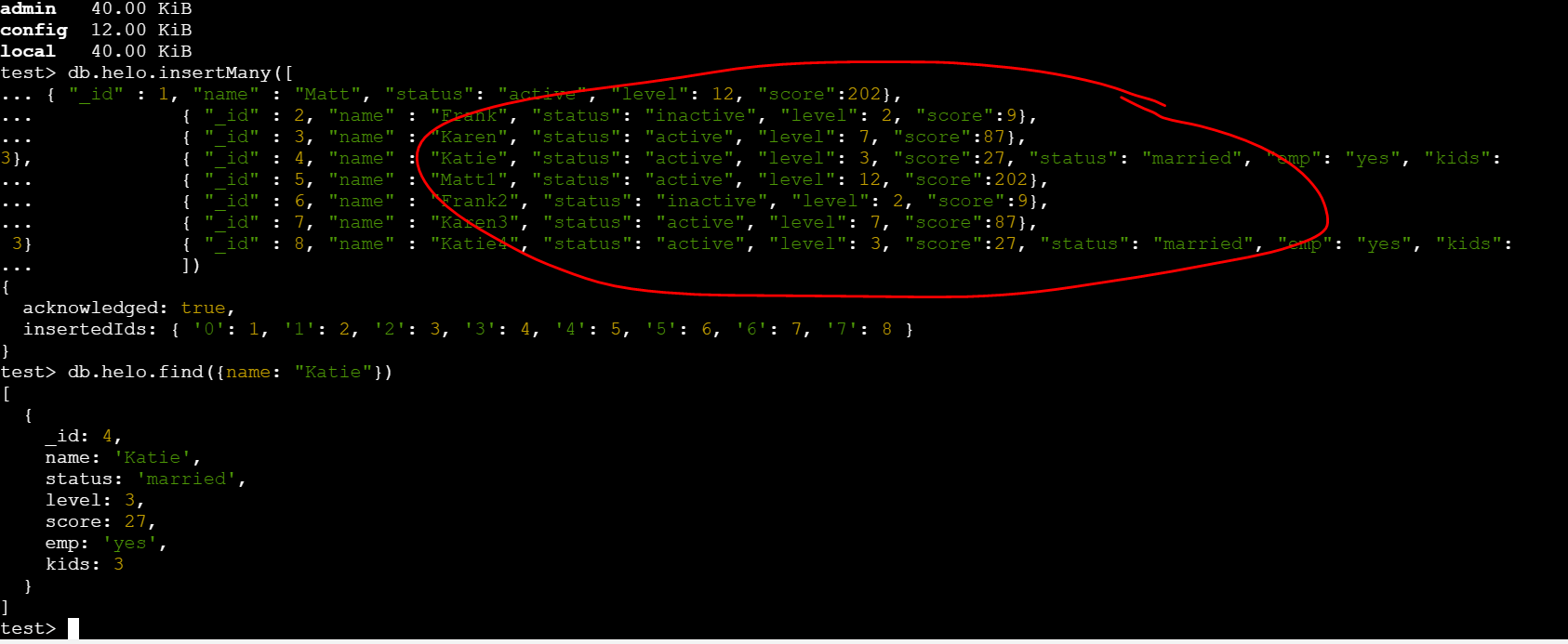
{ "\_id" : 6, "name" : "Frank2", "status": "inactive", "level": 2, "score":9},

{ "\_id" : 7, "name" : "Karen3", "status": "active", "level": 7, "score":87},

{ "\_id" : 8, "name" : "Katie4", "status": "active", "level": 3, "score":27, "status": "married", "emp": "yes", "kids": 3}

])

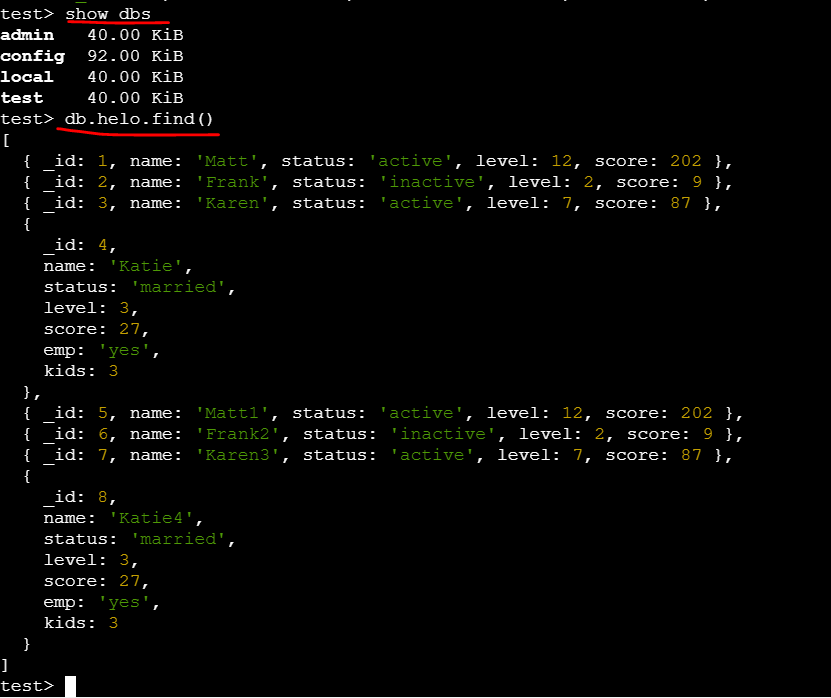
db.helo.find({name: "Katie"})



**#Validate to newly created file**

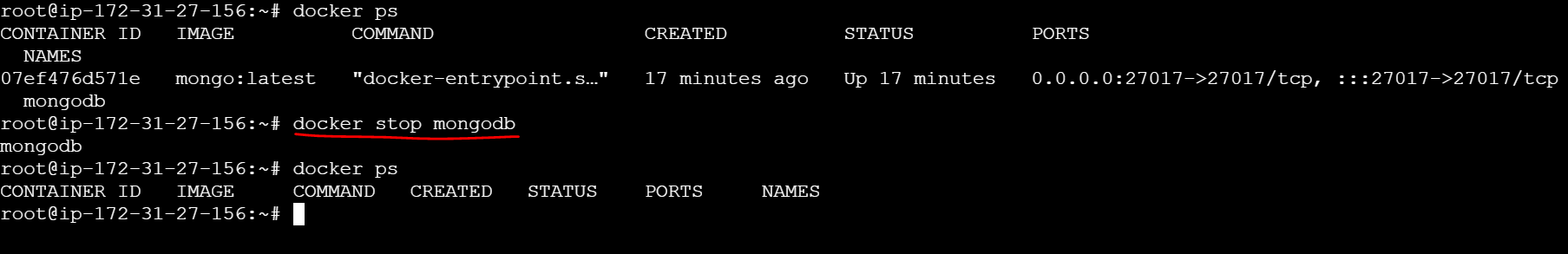
Show dbs;

db.helo.find()



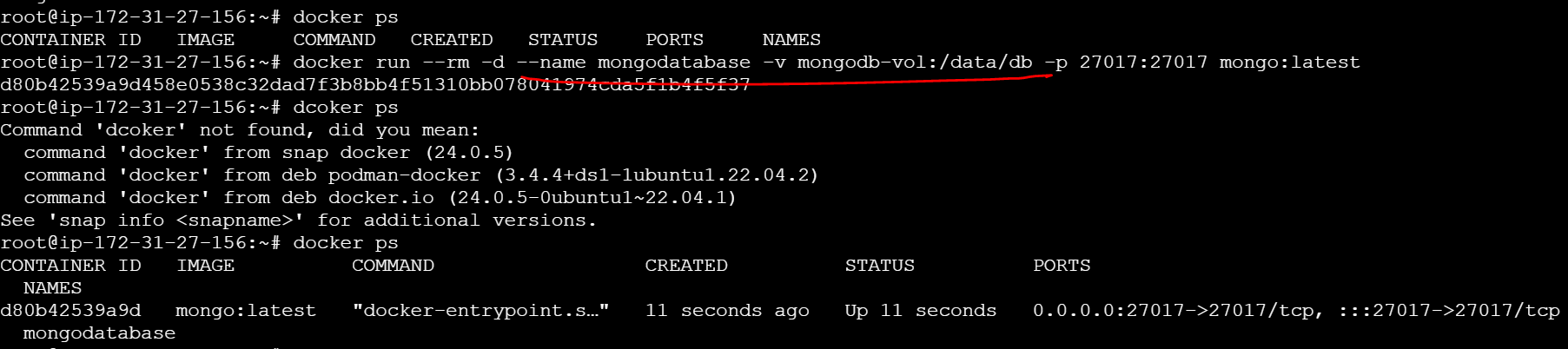
**#stop the container**

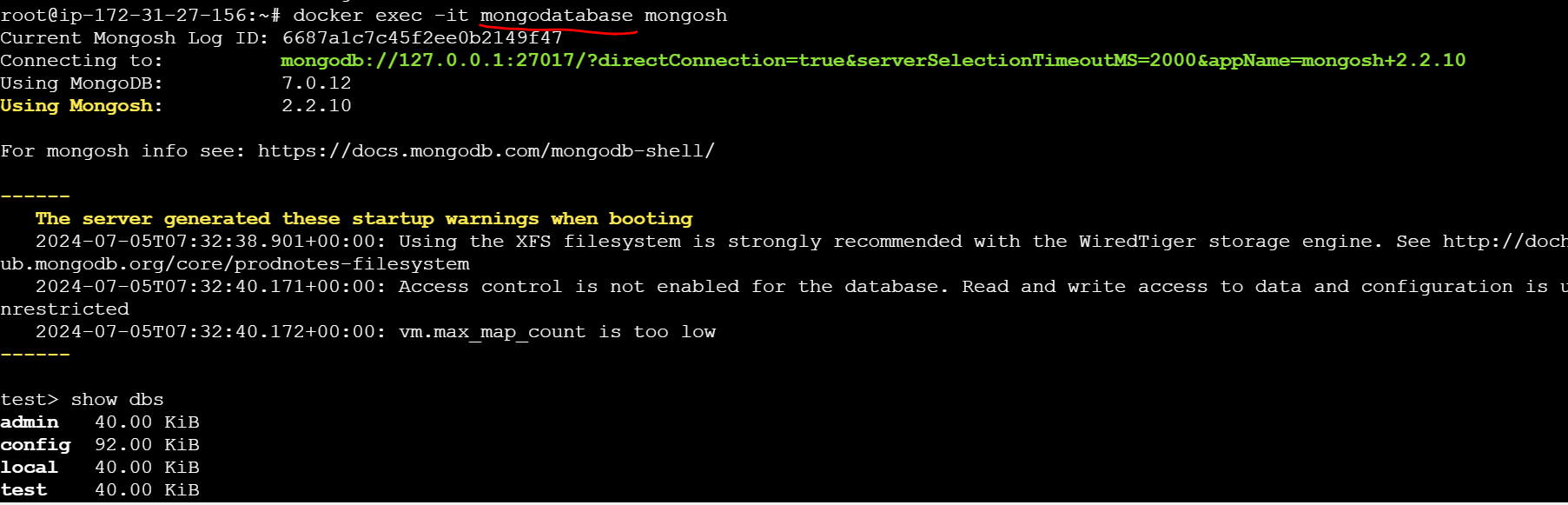
Docker stop mongodb

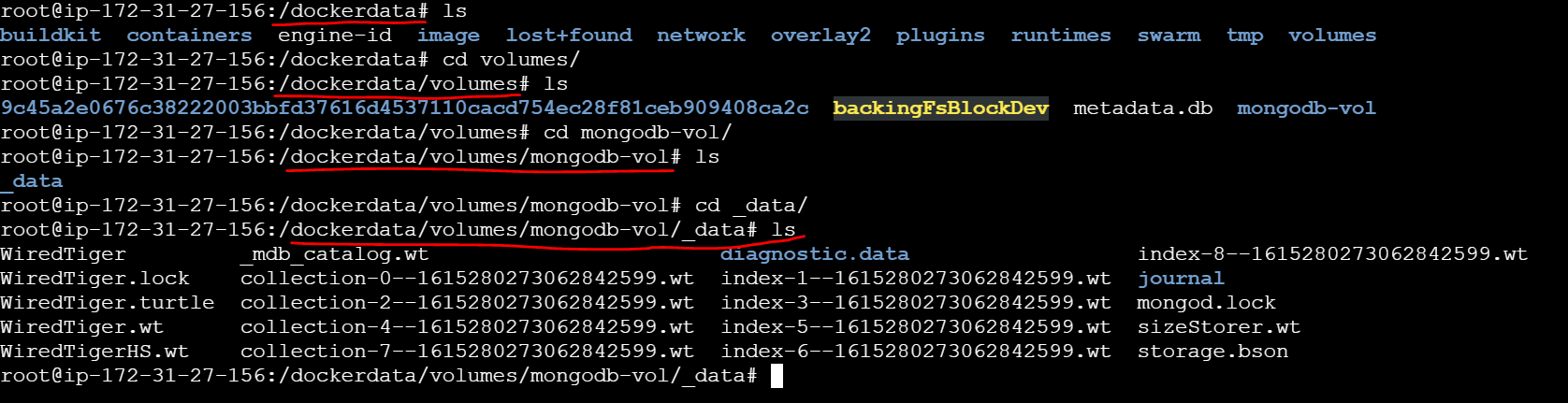


**#create a new container & use the same volume & validate if you get the data/not**

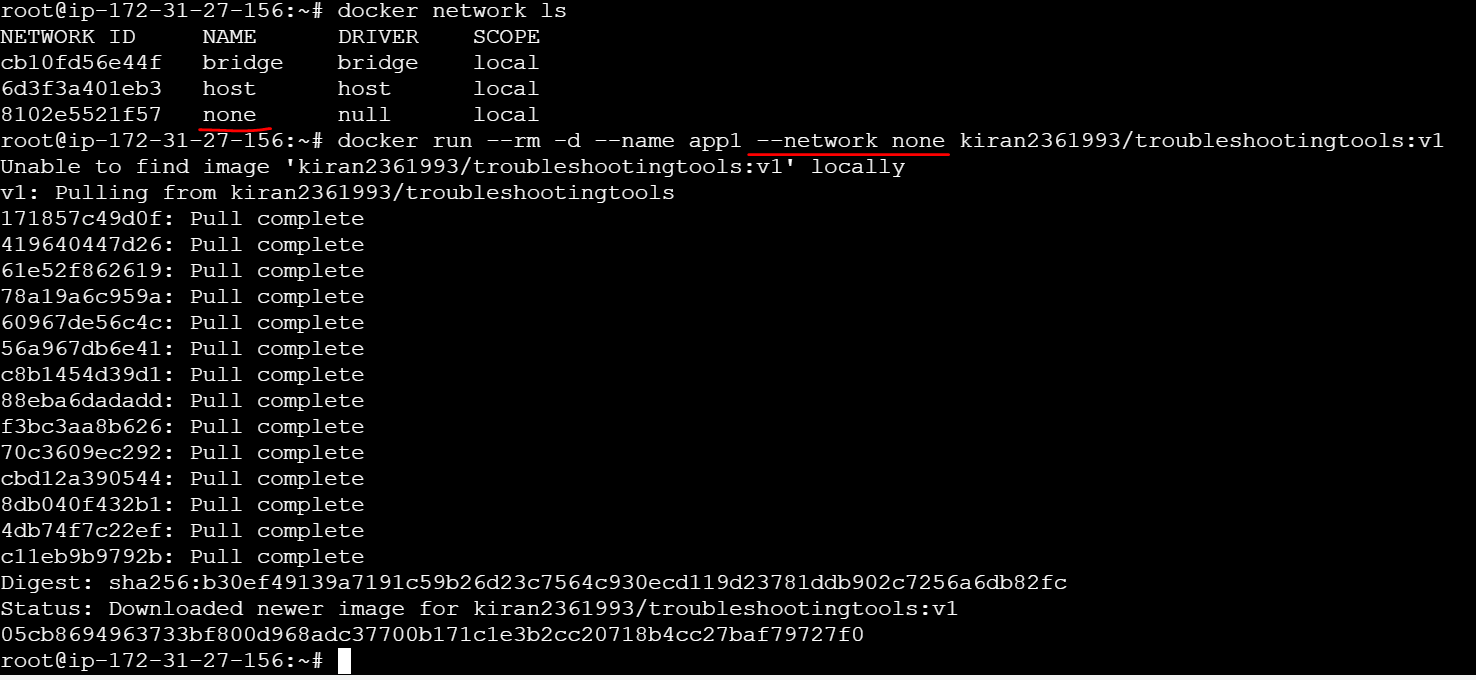
docker run --rm -d --name mongodatabase -v mongodb-vol:/data/db -p 27017:27017 mongo:latest







#creating container will none network



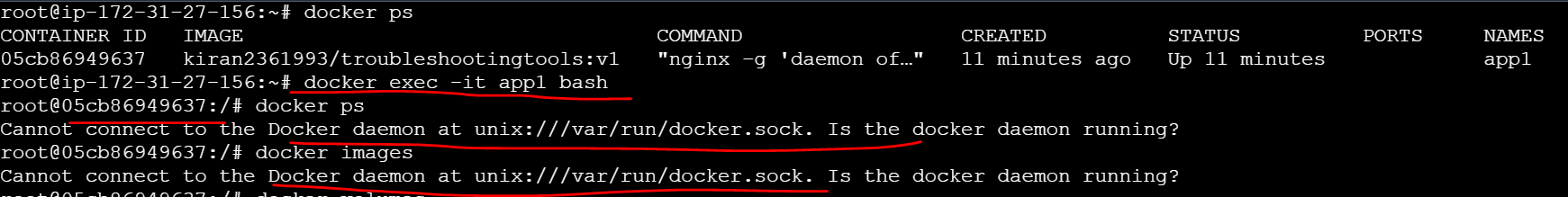
#once you are in inside the container & try to execute thr below commands (it will fail)

root@05cb86949637:/# docker ps

Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

root@05cb86949637:/# docker images

Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?



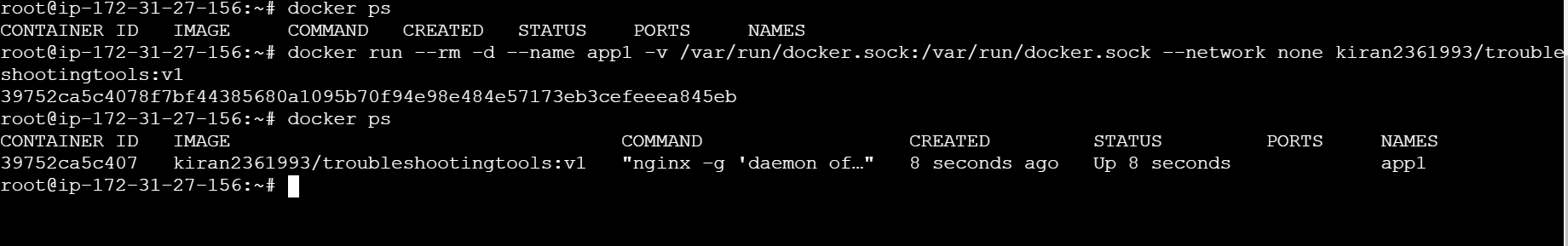
#container does not know what the containers are running on the Host

**Docker socket**

How to fix the attached error (how to get the details of the host from container) by using docker socket

**#Assigning docker socket to container**

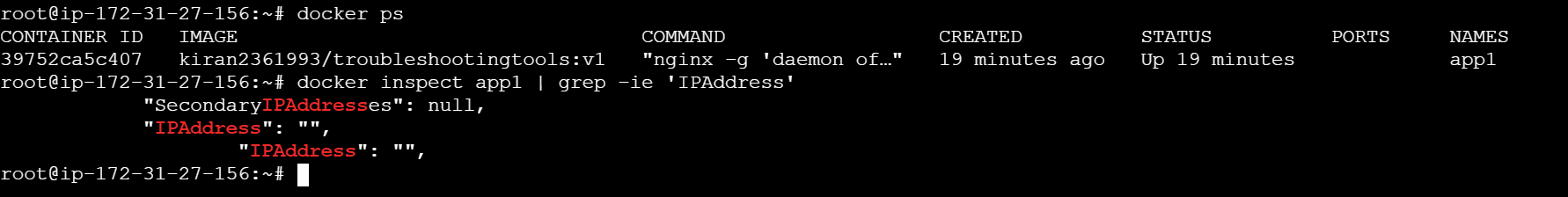
docker run --rm -d --name app1 -v /var/run/docker.sock:/var/run/docker.sock --network none kiran2361993/troubleshootingtools:v1



**# you are getting the details in container only**



**#None network doesn't contain ip address**



**DAY-4**

**Create ubuntu image & install tools like jq, terraform,packer**

**Create docker file**

**Nano dockerfile.dev (paste the below)**

FROM ubuntu:latest

LABEL name="pammi"

ENV AWS\_ACCESS\_KEY=AKIAIOSFODNN7EXAMPLE \

    AWS\_SECRET\_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY \

    AWS\_DEFAULT\_REGION=ap-southeast-1

ARG T\_VERSION='1.6.6' \

    P\_VERSION='1.8.0'

RUN apt update && apt install -y jq net-tools unzip curl wget \

    && apt install -y nginx iputils-ping

RUN wget https://releases.hashicorp.com/terraform/${T\_VERSION}/terraform\_${T\_VERSION}\_linux\_amd64.zip \

    && wget https://releases.hashicorp.com/packer/${P\_VERSION}/packer\_${P\_VERSION}\_linux\_amd64.zip \

    && unzip terraform\_${T\_VERSION}\_linux\_amd64.zip && unzip packer\_${P\_VERSION}\_linux\_amd64.zip \

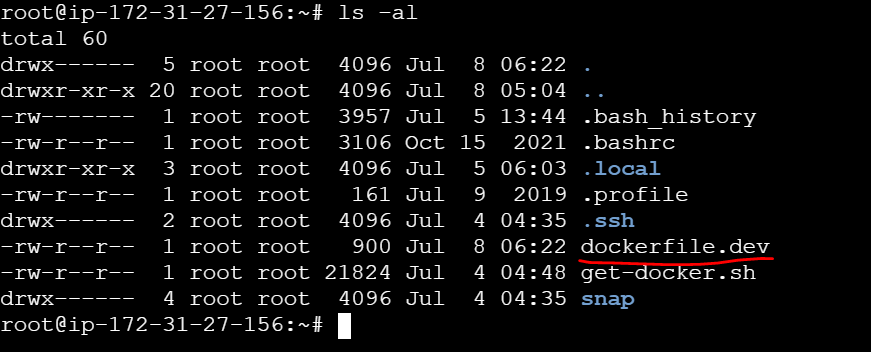
    && chmod 777 terraform && chmod 777 packer \

    && ./terraform version && ./packer version

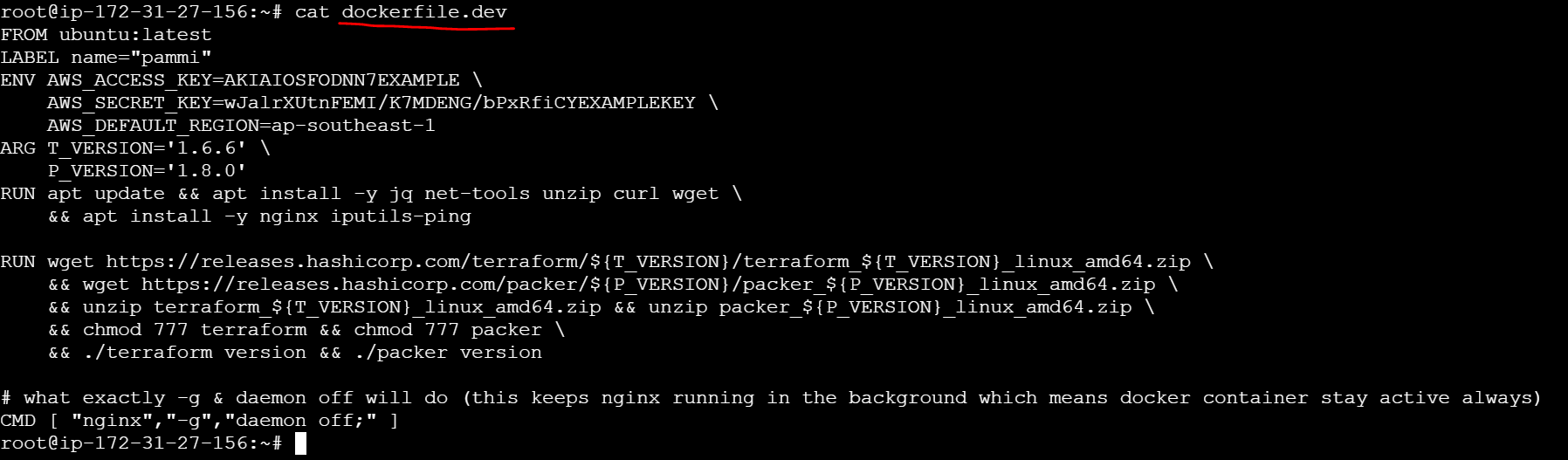
# what exactly -g & daemon off will do (this keeps nginx running in the background which means docker container stay active always)

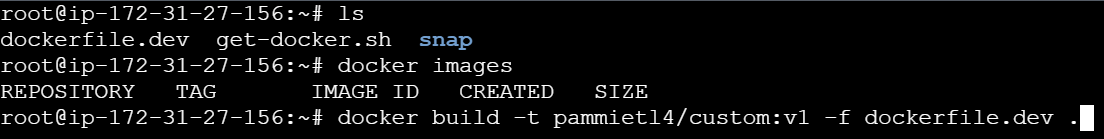
CMD [ "nginx","-g","daemon off;" ]

**Docker build -f dockerfile.dev**

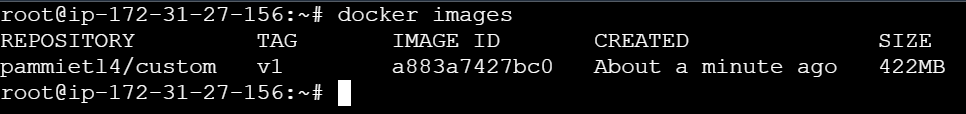


**Building docker image from compose file**





**Validation**



**Build container from above image & do smoke testing**



