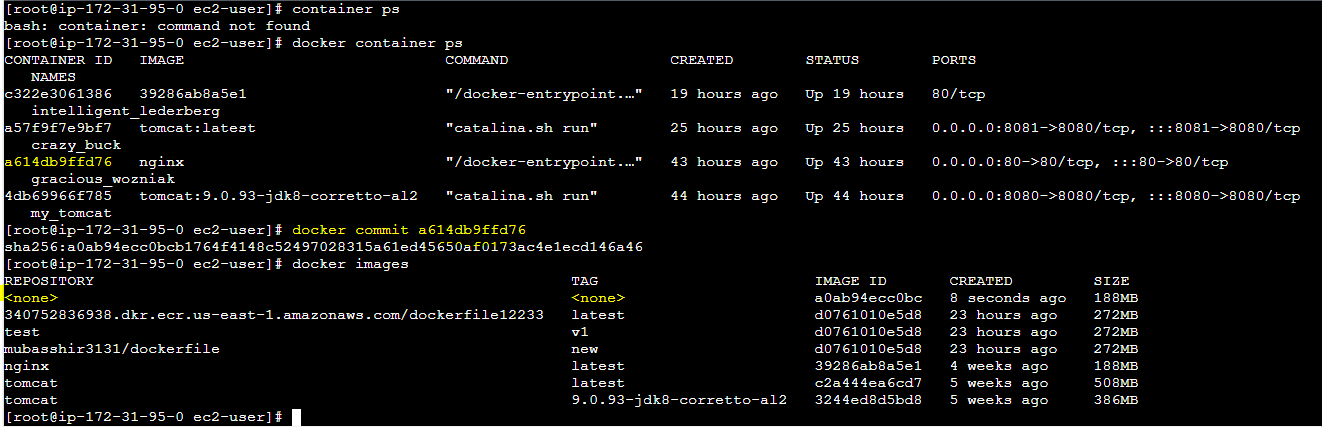
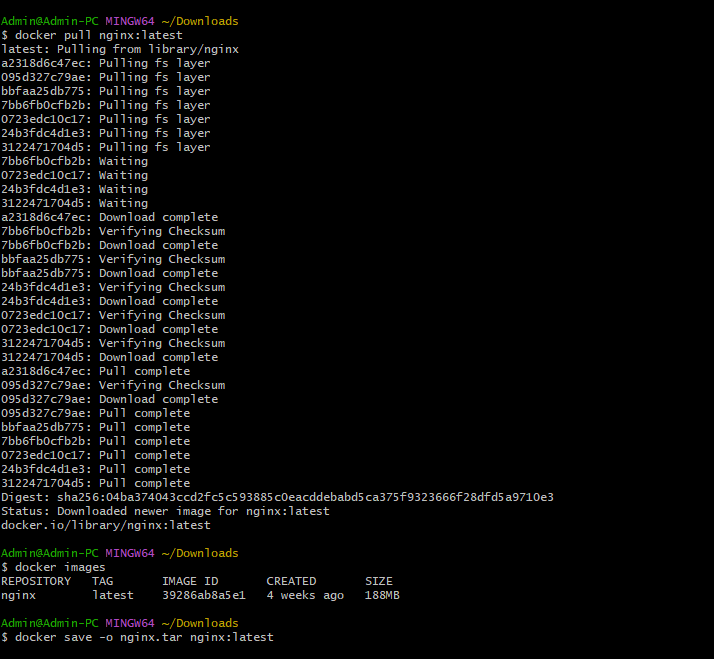
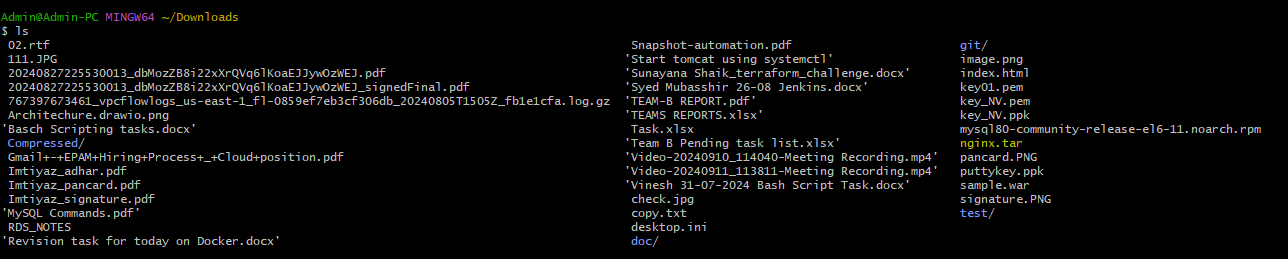
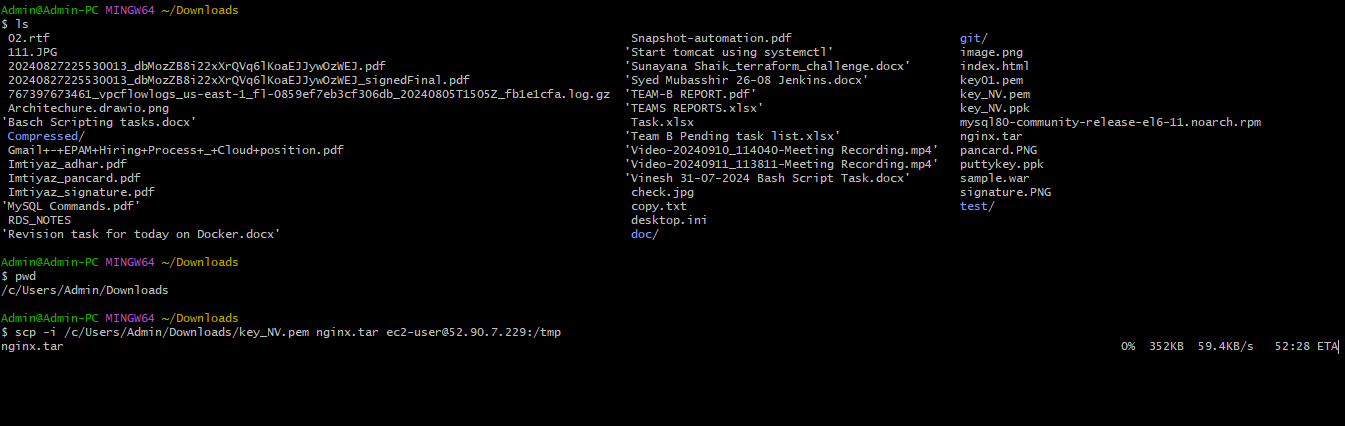
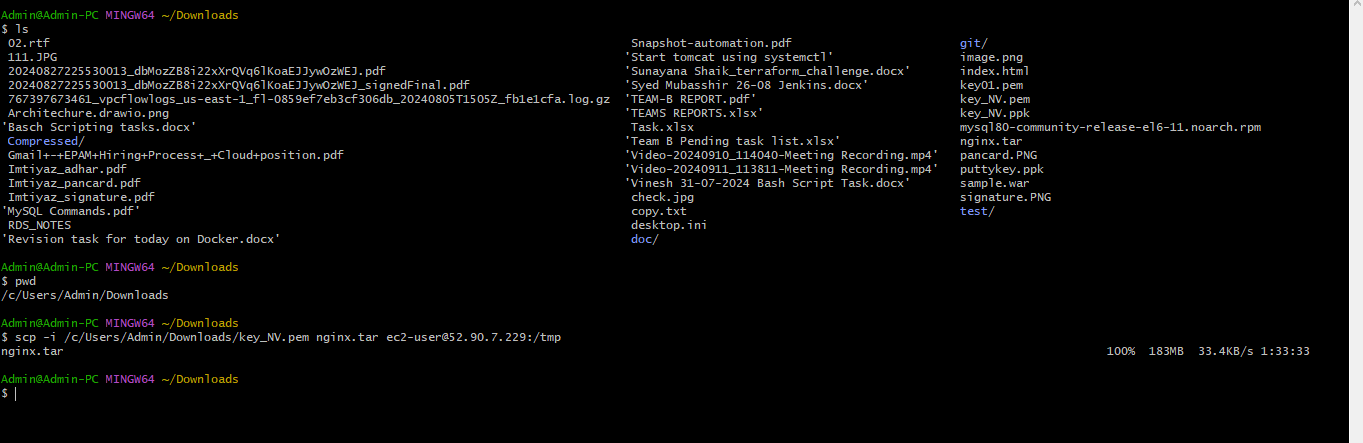
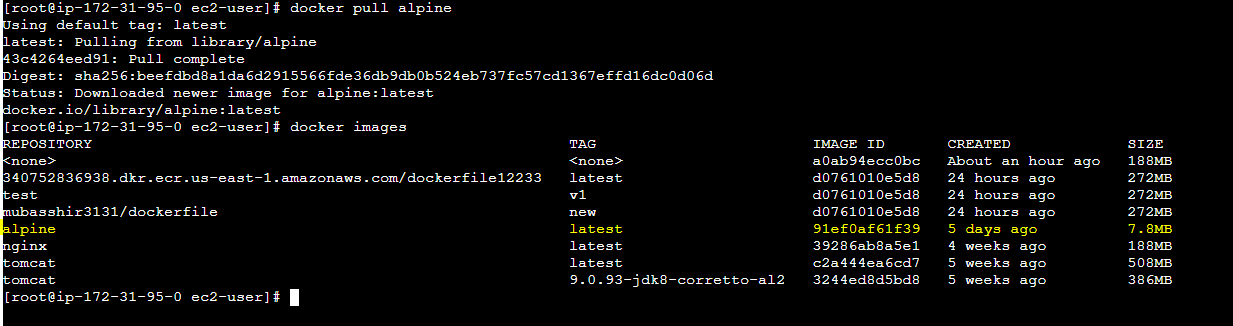
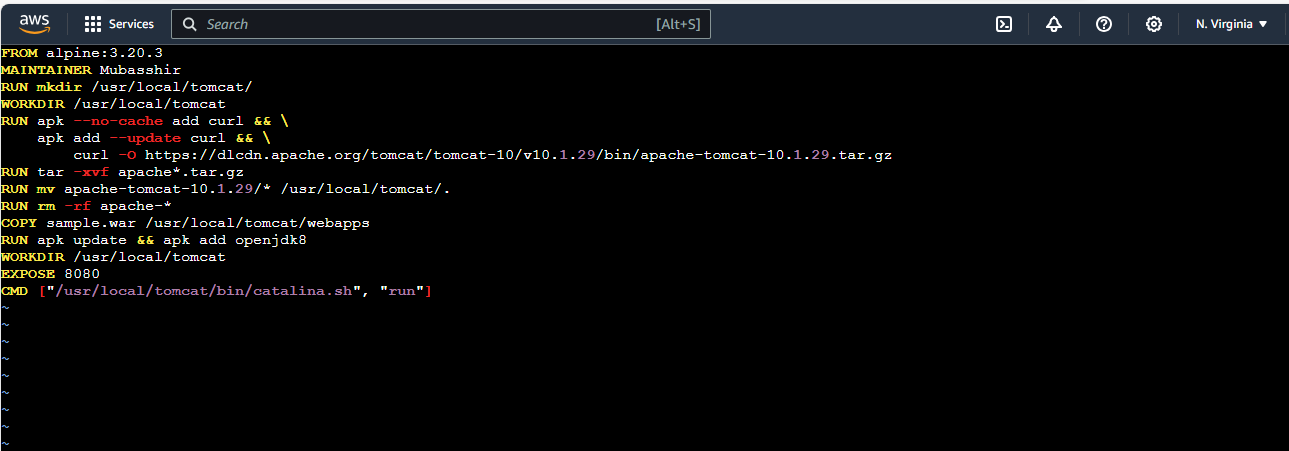
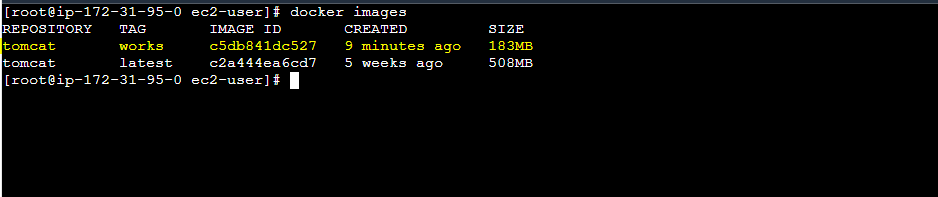
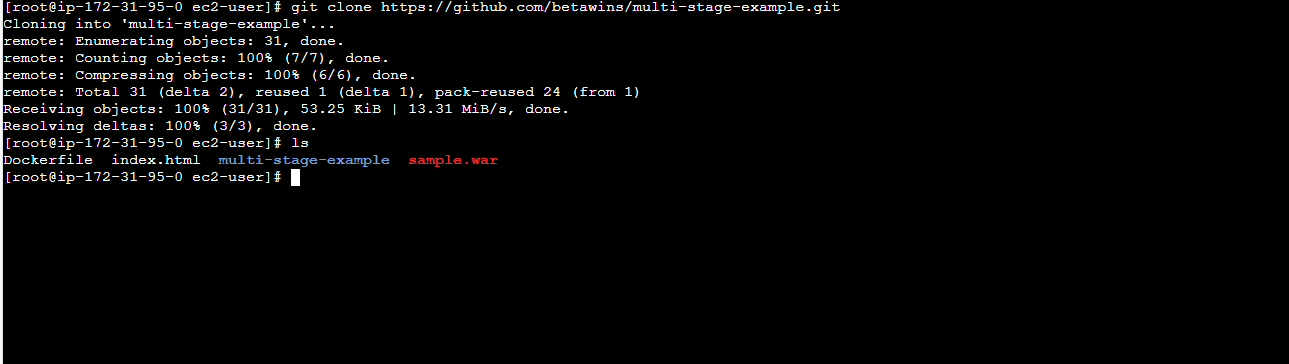
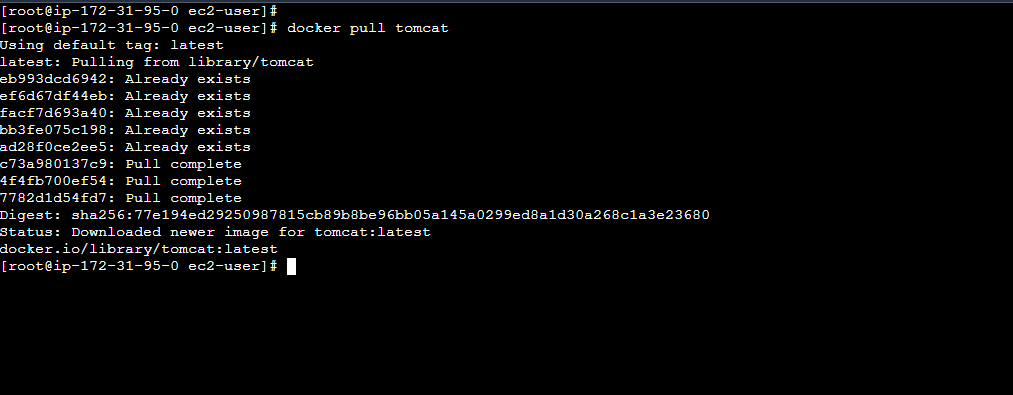
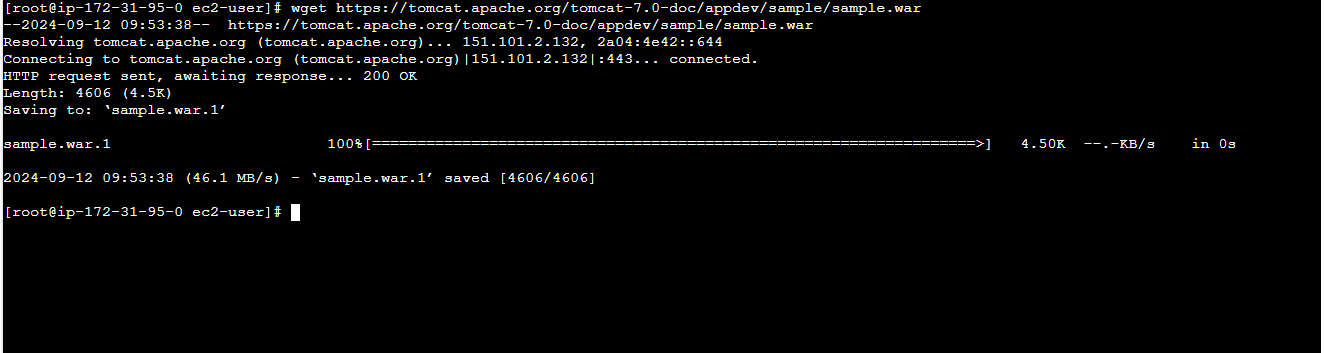
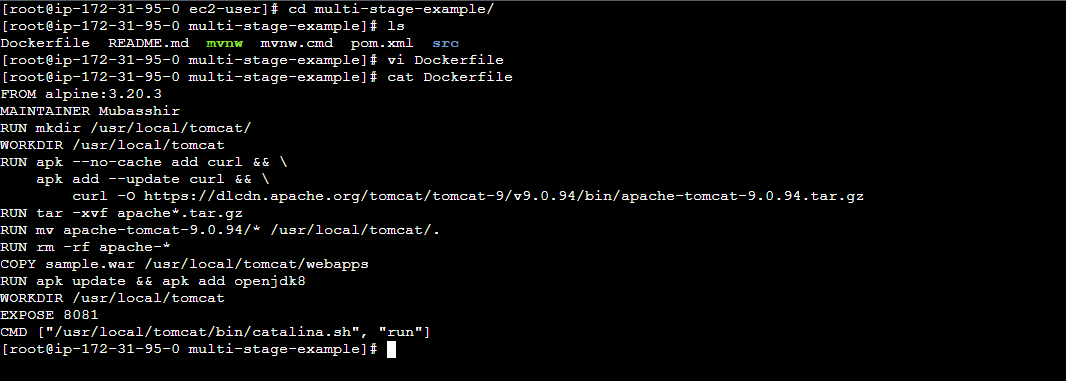
1) Create a image from running container.  
selected the nginx container and created the image out of it:  


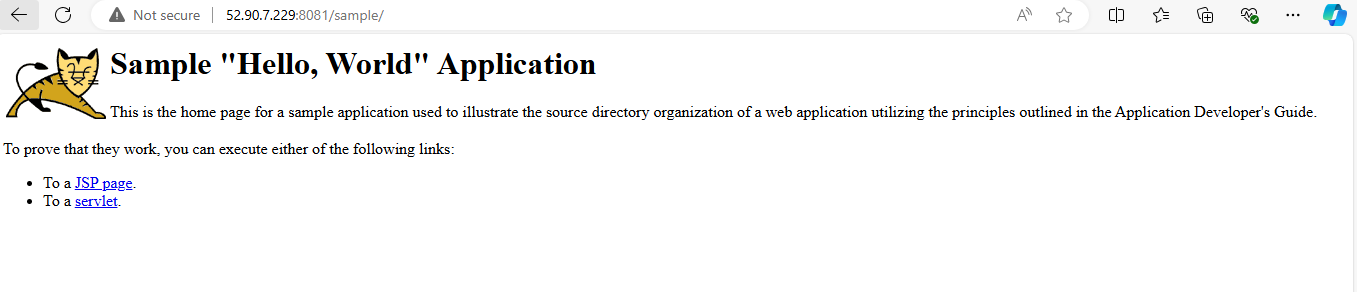
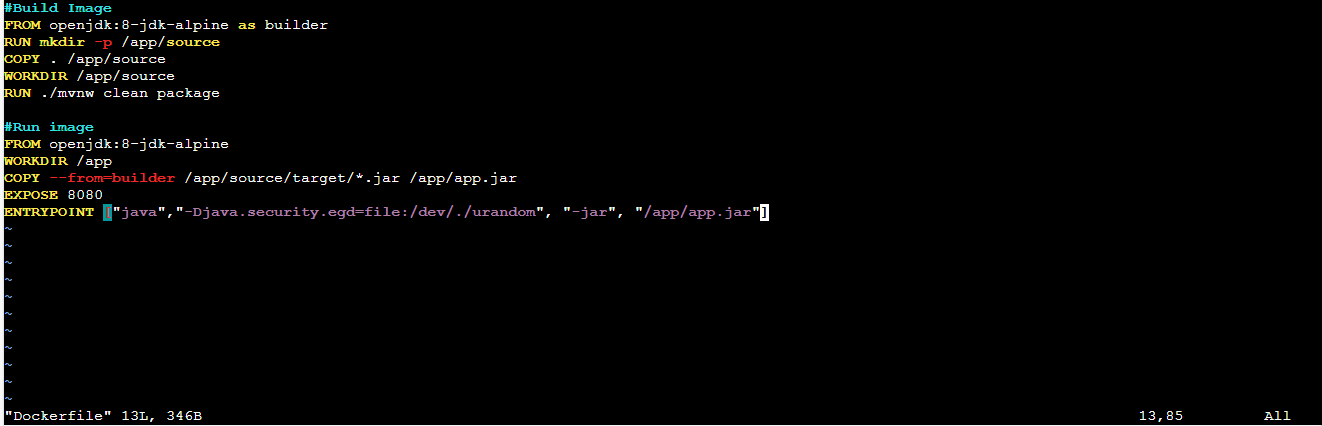
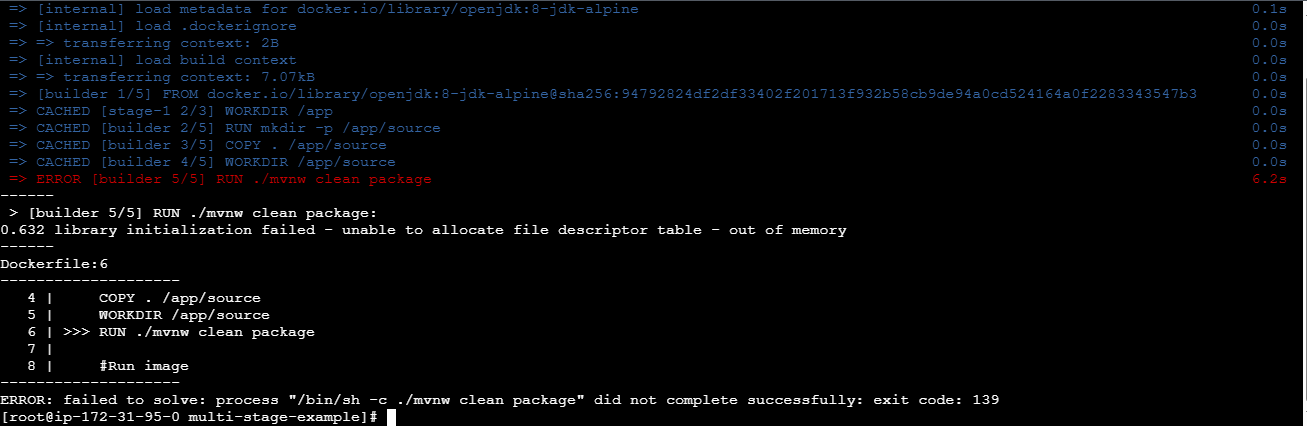
2) Copy image from local machine to docker server and load the image.  
  
create a tar file with above command (save):   
  
pushed the image to remote:  
  
Done;  


3) Create Docker image using alpine and customize with tomcat.  
pulled alpine image:  
  
With the help of Dockerfile:   
building the tomcat with build command:  
  
by using 🡪$ docker images I have checked the list and found out that the docker image has been created :  


4) Create single stage and multi stage docker file using the below source code.

<https://github.com/betawins/multi-stage-example.git>  
  
**FOR SINGLESTAGE:**

Firstly, I have clone the git repo to my server and also downloaded the tomcat image and war package which is required to my tomcat:  
   
war package:  
  
Secondly, Used the Docker file and made some changes according to me: 

Done:   
working fine:  
  
  
FOR MULTISTAGE FILE:  
multisatge file code:  
  
  
got an error:  
  
  
=> ERROR [builder 5/5] RUN ./mvnw clean package 6.3s

------

> [builder 5/5] RUN ./mvnw clean package:

0.556 library initialization failed - unable to allocate file descriptor table - out of memory

------

Dockerfile:6

--------------------

4 | COPY . /app/source

5 | WORKDIR /app/source

6 | >>> RUN ./mvnw clean package

7 |

8 | #Run image  
  
resolved the error:

1. Adjusting System Settings

Freeing System Resources

To ensure Docker has enough resources, you might need to manage other system resources and processes:

1. Close Unnecessary Applications:

o Terminate or reduce the resource usage of non-essential applications to free up memory and CPU.

2. Increase Swap Space:

o If you’re running out of memory, increasing swap space can help. You can add or resize swap space with the following steps:

Create a Swap File:

sudo fallocate -l 2G /swapfile

Set Permissions:

sudo chmod 600 /swapfile

Set Up Swap Area:

sudo mkswap /swapfile

Enable Swap:

sudo swapon /swapfile

Make Swap Permanent:

Add the following line to /etc/fstab:

/swapfile none swap sw 0 0

Docker Daemon Configuration

You might need to adjust Docker daemon settings to optimize performance:

1. Locate or Create the Daemon Configuration File:

o Docker's configuration file is located at /etc/docker/daemon.json. If it does not exist, you can create it.

2. Edit the Configuration File:

o Open or create /etc/docker/daemon.json with a text editor like nano or vim:

sudo nano /etc/docker/daemon.json

o Configure daemon options such as logging or storage drivers. Example configuration:

{

"storage-driver": "overlay2",

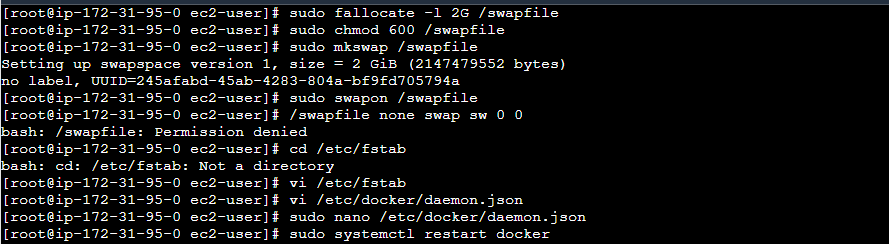
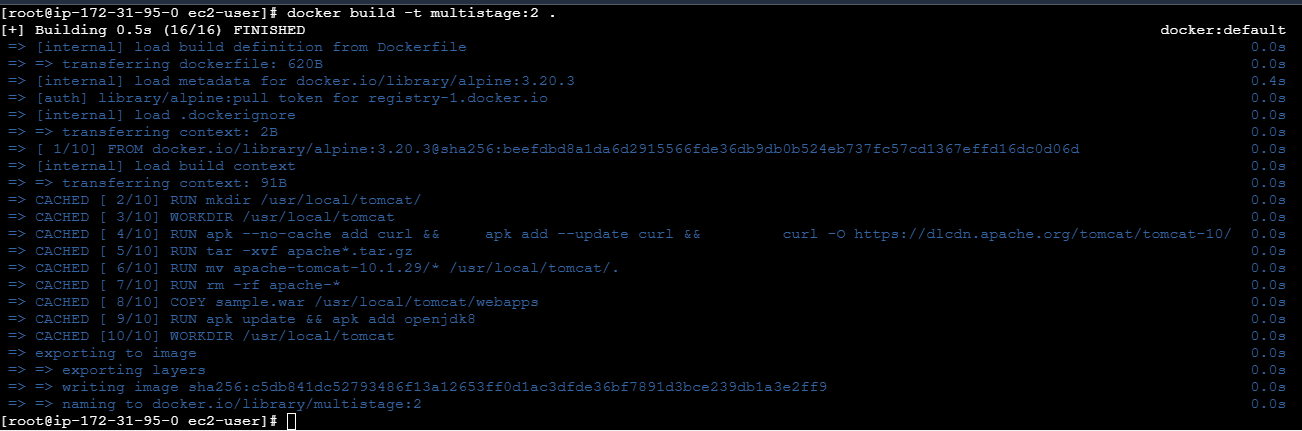
"log-level": "warn",

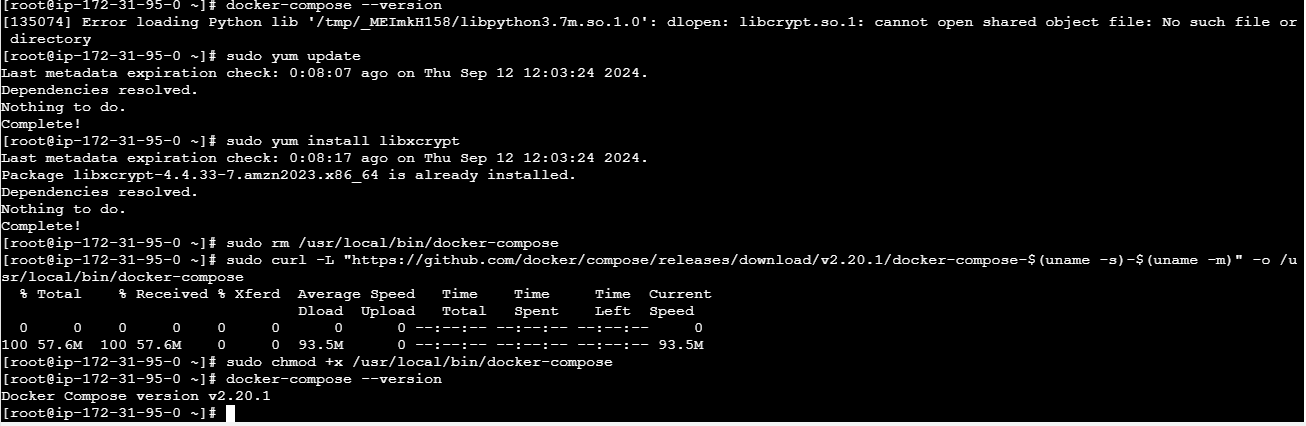
"max-concurrent-downloads": 10

}

3. Restart Docker Daemon:

o Apply the changes by restarting the Docker service:

sudo systemctl restart docker  
  
  


5) Install docker compose and execute sample applciation.  
  
  
created a file named it as “docker-compose-yml” & added this script:  
version: '3'

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=somewordpress

- MYSQL\_DATABASE=wordpress

- MYSQL\_USER=wordpress

- MYSQL\_PASSWORD=wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

ports:

- "8000:80"

restart: always

environment:

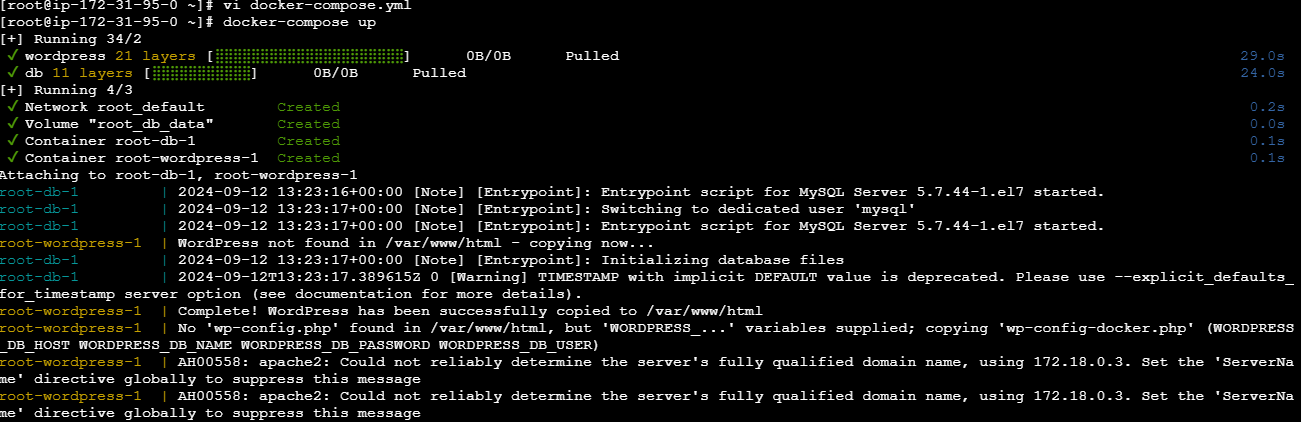
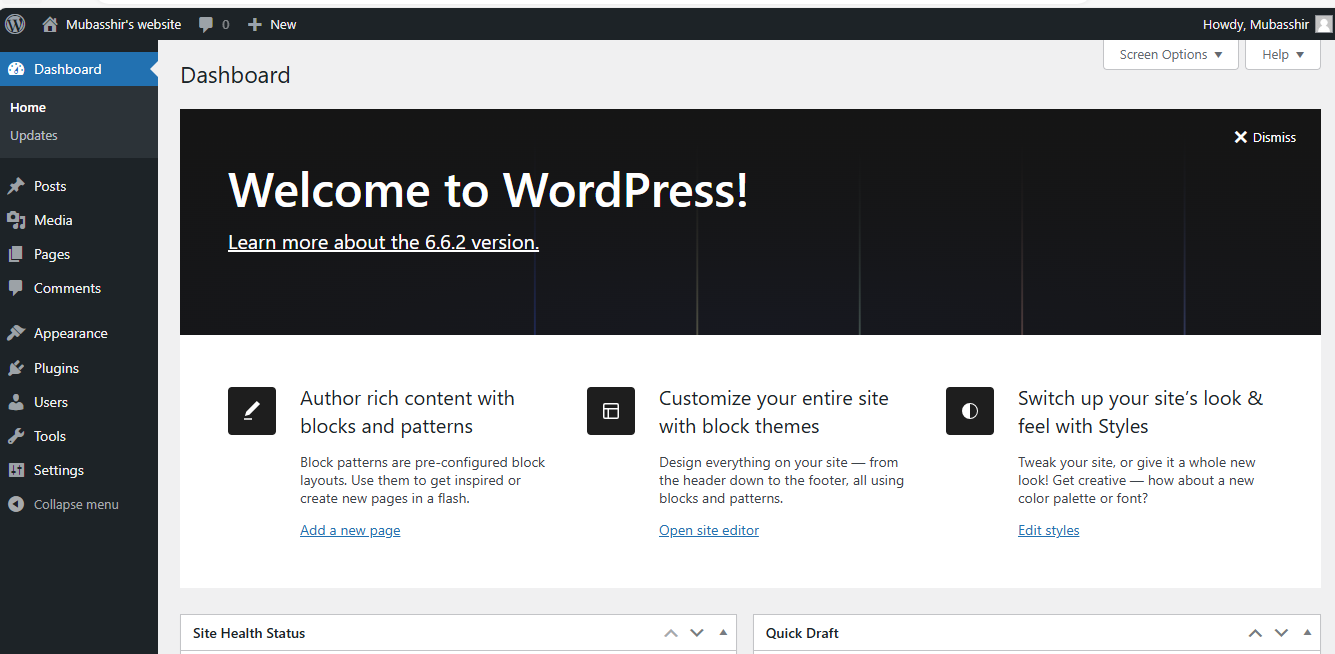
- WORDPRESS\_DB\_HOST=db:3306

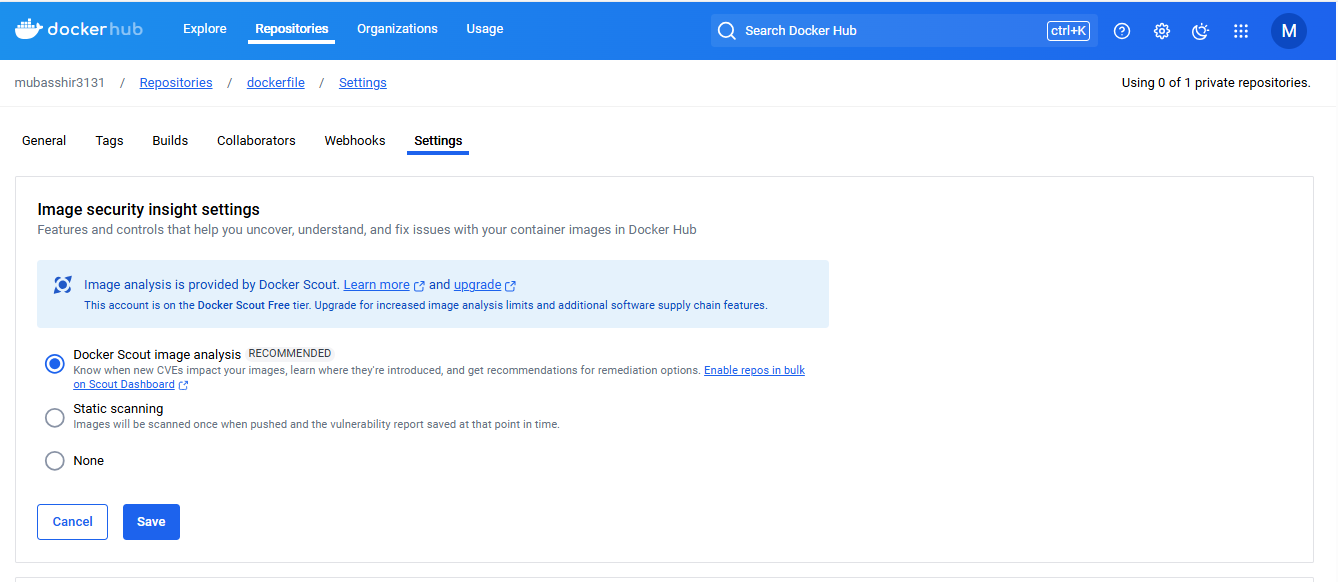
- WORDPRESS\_DB\_USER=wordpress

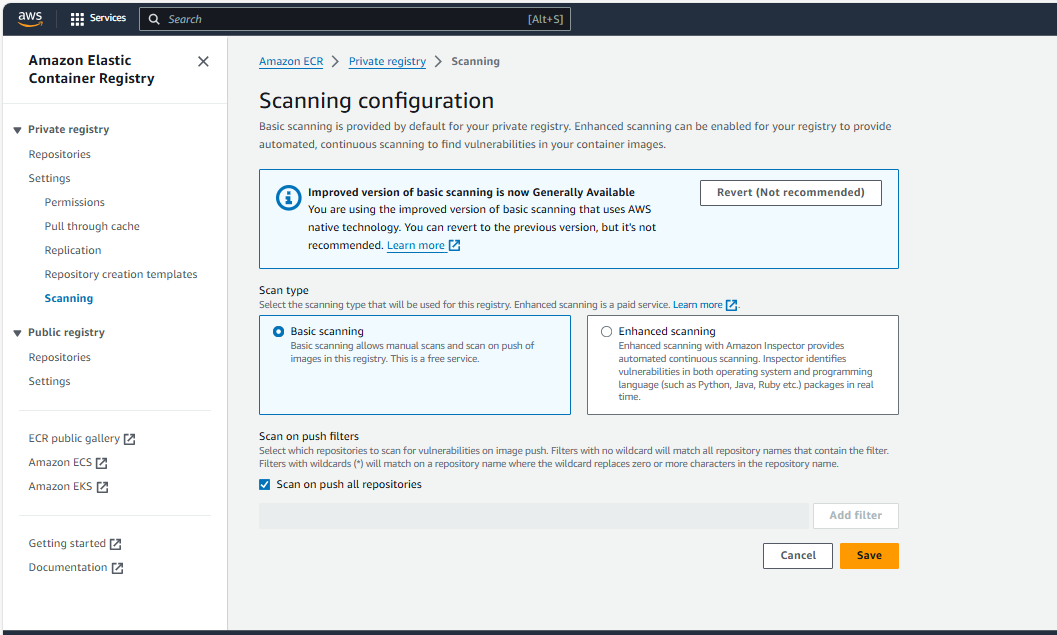
- WORDPRESS\_DB\_PASSWORD=wordpress

- WORDPRESS\_DB\_NAME=wordpress

volumes:

db\_data: { }  
  
spublic ip:8000  


6) Implement solution to scan images when pushed to docker registry.   


7) Implement solution to scan images when pushed to aws ecr.   


8) Create a jenkins pipeline to create a docker image and push the image to dockerhub.