

1. Project Title

Deployment of React E-commerce Application using Docker, Jenkins, and AWS

2. Candidate Details

- **Name:** Junaid Ahamed
 - **Project Type:** DevOps – Application Deployment
 - **Organization:** GUVI Geeks Network Pvt Ltd
-

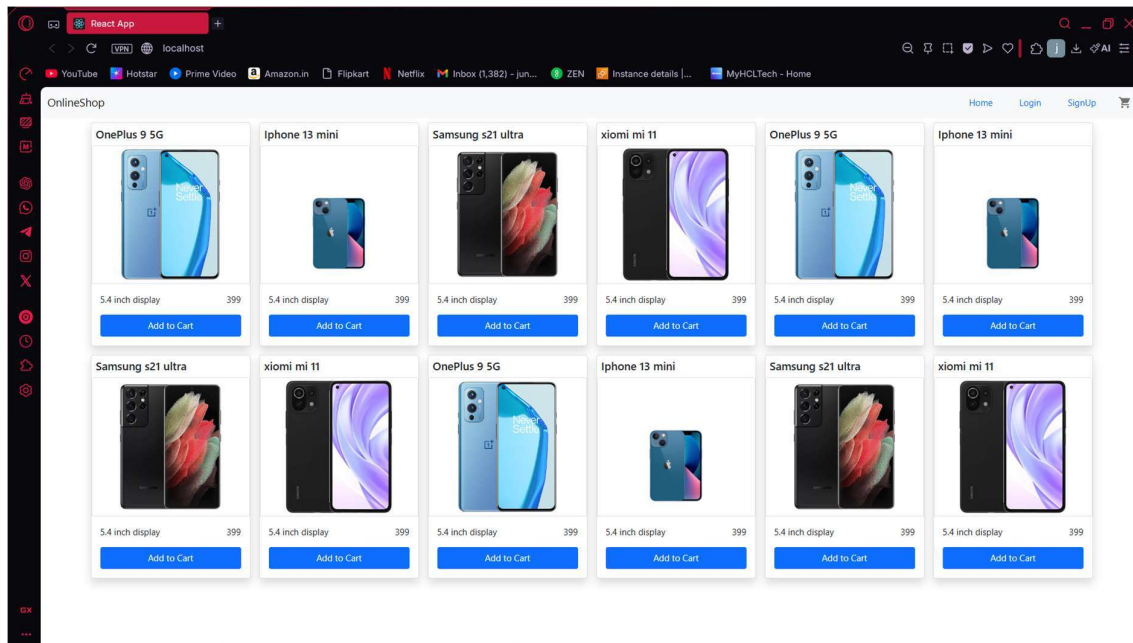
3. Project Objective

The objective of this project is to deploy a production-ready React application using DevOps best practices. The project includes containerization using Docker, CI/CD automation using Jenkins, cloud deployment on AWS EC2, Docker image management using Docker Hub, version control using GitHub CLI, and application health monitoring.

4. Application Details

- **Application Type:** ReactJS (Static Build – E-commerce Application)
- **Source Repository:**
 - <https://github.com/sriram-R-krishnan/devops-build>
- **Deployment Port:**
 - HTTP – Port 80


The repository contains pre-built React static files which are directly served using Nginx for production readiness.



5. Docker Implementation

5.1 Docker Installation & Environment Setup

Docker and Docker Compose were installed inside Ubuntu (WSL and EC2). The environment was verified using Docker version commands.

 **Screenshot:** Docker & Docker-Compose Installation Verification
(Insert terminal screenshot showing `docker --version` and `docker-compose --version`)

5.2 Dockerfile Creation

A Dockerfile was created using the `nginx:alpine` base image to serve the React build files.

Key Points:

- Lightweight Nginx image
- Static files copied to `/usr/share/nginx/html`
- Port 80 exposed

```

junaaid@LAPTOP-GU5B805P:~$ git clone https://github.com/sriram-R-krishnan/devops-build.git
Cloning into 'devops-build'...
remote: Enumerating objects: 21, done.
remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21 (from 1)
Receiving objects: 100% (21/21), 720.09 KiB | 7.50 MiB/s, done.
junaaid@LAPTOP-GU5B805P:~$ cd devops-build
junaaid@LAPTOP-GU5B805P:~/devops-build$ pwd
/home/junaaid/devops-build
junaaid@LAPTOP-GU5B805P:~/devops-build$ ls
build
junaaid@LAPTOP-GU5B805P:~/devops-build$ nano Dockerfile
junaaid@LAPTOP-GU5B805P:~/devops-build$ rm -f Dockerfile
junaaid@LAPTOP-GU5B805P:~/devops-build$ nano Dockerfile
junaaid@LAPTOP-GU5B805P:~/devops-build$ rm -f Dockerfile
junaaid@LAPTOP-GU5B805P:~/devops-build$ cat <<EOF > Dockerfile
FROM nginx:alpine

RUN rm -rf /usr/share/nginx/html/*

COPY build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
EOF
junaaid@LAPTOP-GU5B805P:~/devops-build$ cat Dockerfile
FROM nginx:alpine

RUN rm -rf /usr/share/nginx/html/*

COPY build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
junaaid@LAPTOP-GU5B805P:~/devops-build$ ^[[200~docker build -t react-app .
^[[201~docker: command not found
junaaid@LAPTOP-GU5B805P:~/devops-build$ docker build -t react-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.419MB
Step 1/5 : FROM nginx:alpine
alpine: Pulling from library/nginx
1074353eec0d: Pulling fs layer
25f453064fd3: Pulling fs layer
567f84da6fbd: Pulling fs layer
da7c973d8b92: Pulling fs layer
33f95a0f3229: Pulling fs layer

```

5.3 Docker Image Build

The Docker image was built successfully using the Dockerfile.

```

junaid@LAPTOP-GU58805P:~/devops-build$ cat <<EOF > build.sh
#!/bin/bash

echo "Building Docker image..."
docker build -t react-app .
EOF
junaid@LAPTOP-GU58805P:~/devops-build$ chmod +x build.sh
junaid@LAPTOP-GU58805P:~/devops-build$ ./build.sh
Building Docker image...
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.421MB
Step 1/5 : FROM nginx:alpine
--> 04da2b0513cd
Step 2/5 : RUN rm -rf /usr/share/nginx/html/*
--> Using cache
--> 29926da7f24e
Step 3/5 : COPY build /usr/share/nginx/html
--> Using cache
--> 2a38f7bf7f85
Step 4/5 : EXPOSE 80
--> Using cache
--> 8b2cc724ad0f
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
--> Using cache
--> 7a7ef912cf1f
Successfully built 7a7ef912cf1f
Successfully tagged react-app:latest
junaid@LAPTOP-GU58805P:~/devops-build$ cat <<EOF > deploy.sh
#!/bin/bash

echo "Stopping old container..."
docker stop react-app || true
docker rm react-app || true

echo "Starting new container..."
docker run -d -p 80:80 --name react-app react-app
EOF
junaid@LAPTOP-GU58805P:~/devops-build$ chmod +x deploy.sh
junaid@LAPTOP-GU58805P:~/devops-build$ ./deploy.sh
Stopping old container...
Error response from daemon: No such container: react-app
Error response from daemon: No such container: react-app
Starting new container...
dab4421a61733ca2c6a012ff7ff78880f425414f46ae89aca16f838a18c9db06
docker: Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint
react-app (018e6778814fc3eaa1244b862354ab6aab53f94988630ceaab55371e29720b9a): Bind for 0.0.0.0:80 failed: port is already allocated

```

6. Docker Compose Implementation

A docker-compose.yml file was created to manage container lifecycle.

Features:

- Application runs on port 80
- Restart policy enabled
- Simplified deployment

```

Successfully tagged react-app:latest
junaid@LAPTOP-GU5B805P:~/devops-build$ docker run -d -p 80:80 --name react-app react-app
98ea5ec9fd0e98a60dc2b97c2a0abd03e12e3323ae70eaf27e4b795cfb79affd
junaid@LAPTOP-GU5B805P:~/devops-build$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
98ea5ec9fd0e   react-app   "/docker-entrypoint..."  19 seconds ago Up 18 seconds  0.0.0.0:80->80/tcp, [::]:80->80/tcp   react-app
junaid@LAPTOP-GU5B805P:~/devops-build$ Sdocker stop react-app
Command 'Sdocker' not found, did you mean:
  command 'docker' from snap docker (28.4.0)
  command 'kdocker' from deb kdocker (5.4-1)
  command 'docker' from deb docker.io (28.2.2-0ubuntu1~24.04.1)
  command 'docker' from deb podman-docker (4.9.3+ds1-1ubuntu0.2)
See 'snap info <snapname>' for additional versions.
junaid@LAPTOP-GU5B805P:~/devops-build$ docker stop react-app
react-app
junaid@LAPTOP-GU5B805P:~/devops-build$ docker rm react-app
react-app
junaid@LAPTOP-GU5B805P:~/devops-build$ cat <<EOF > docker-compose.yml
version: "3.8"

services:
  web:
    image: react-app
    ports:
      - "80:80"
    restart: always
EOF
junaid@LAPTOP-GU5B805P:~/devops-build$ docker-compose up -d
Creating network "devops-build_default" with the default driver
Creating devops-build_web_1 ... done
junaid@LAPTOP-GU5B805P:~/devops-build$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
8a2931205c2a   react-app   "/docker-entrypoint..."  10 seconds ago Up 9 seconds   0.0.0.0:80->80/tcp, [::]:80->80/tcp   devops-build_web_1
junaid@LAPTOP-GU5B805P:~/devops-build$

```

7. Bash Scripting

Two Bash scripts were created for automation.

7.1 build.sh

- Automates Docker image build

7.2 deploy.sh

- Stops old containers
- Deploys application using Docker Compose

```

junaid@LAPTOP-GU58805P:~/devops-build$ cat <<EOF > build.sh
#!/bin/bash

echo "Building Docker image..."
docker build -t react-app .
EOF
junaid@LAPTOP-GU58805P:~/devops-build$ chmod +x build.sh
junaid@LAPTOP-GU58805P:~/devops-build$ ./build.sh
Building Docker image...
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
             Install the buildx component to build images with BuildKit:
             https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.421MB
Step 1/5 : FROM nginx:alpine
--> 04da2b0513cd
Step 2/5 : RUN rm -rf /usr/share/nginx/html/*
--> Using cache
--> 29926da7f24e
Step 3/5 : COPY build /usr/share/nginx/html
--> Using cache
--> 2a38f7bf7f85
Step 4/5 : EXPOSE 80
--> Using cache
--> 8b2cc724ad0f
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
--> Using cache
--> 7a7ef912cf1f
Successfully built 7a7ef912cf1f
Successfully tagged react-app:latest
junaid@LAPTOP-GU58805P:~/devops-build$ cat <<EOF > deploy.sh
#!/bin/bash

echo "Stopping old container..."
docker stop react-app || true
docker rm react-app || true

echo "Starting new container..."
docker run -d -p 80:80 --name react-app react-app
EOF
junaid@LAPTOP-GU58805P:~/devops-build$ chmod +x deploy.sh
junaid@LAPTOP-GU58805P:~/devops-build$ ./deploy.sh
Stopping old container...
Error response from daemon: No such container: react-app
Error response from daemon: No such container: react-app
Starting new container...
dab4421a61733ca2c6a012ff7ff78880f425414f46ae89aca16f838a18c9db06
docker: Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint
react-app (018e6778814fc3eaa1244b862354ab6aab53f94988630ceaab55371e29720b9a): Bind for 0.0.0.0:80 failed: port is already allocated

```

8. Version Control (GitHub – CLI Only)

Git operations were performed strictly using CLI.

Steps Performed:

- .gitignore and .dockerignore created
- dev branch created
- Code committed and pushed using CLI
- SSH authentication configured

```

junaid@LAPTOP-GU5B805P:~/devops-build$ git status
On branch dev
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore
    modified:   docker-compose.yml

junaid@LAPTOP-GU5B805P:~/devops-build$ git commit -m "Dockerized app with compose, scripts and Jenkins pipeline"
[dev 04973cb] Dockerized app with compose, scripts and Jenkins pipeline
 2 files changed, 4 insertions(+), 3 deletions(-)
junaid@LAPTOP-GU5B805P:~/devops-build$ git push -u origin dev
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 445 bytes | 445.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:Junaid-dot-max/devops-build.git
   0507b38..04973cb  dev -> dev
branch 'dev' set up to track 'origin/dev'.
junaid@LAPTOP-GU5B805P:~/devops-build$ git status
On branch dev
Your branch is up to date with 'origin/dev'.

nothing to commit, working tree clean

```

GitHub Repository URL:

<https://github.com/Junaid-dot-max/devops-build>

9. Docker Hub Integration

Two Docker Hub repositories were created as per requirements:

Environment	Repository	Visibility
DEV	ahamedjunaid/devops-build	Public
PROD	ahamedjunaid/devops-build-prod	Private

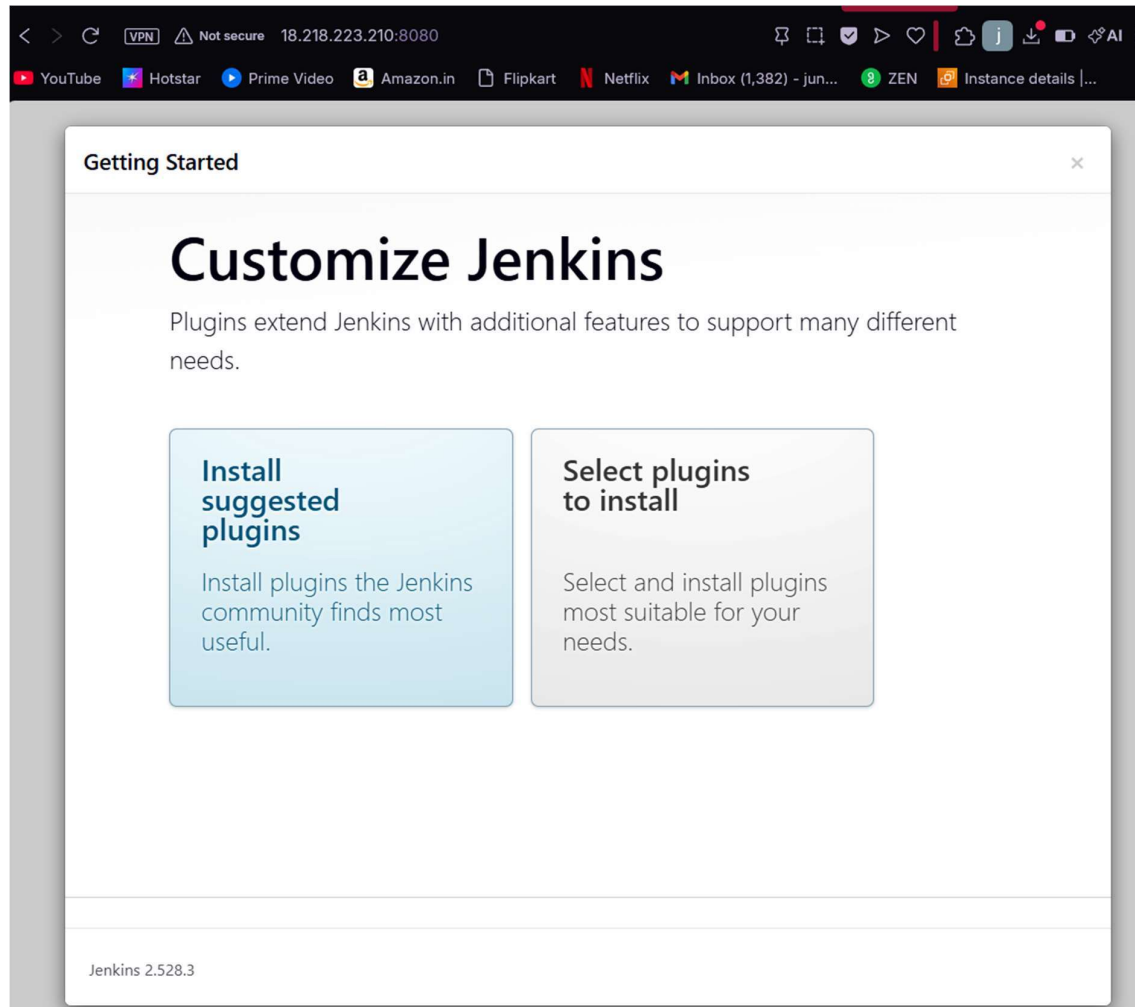
Docker images were tagged and pushed successfully.

The screenshot displays the Docker Hub interface for the repository `ahamedjunaid/devops-build`. The main content area shows the repository's general information, including the last push time (29 minutes ago) and repository size (23.4 MB). Below this, the 'Tags' section lists two tags: `latest` and `dev`, both pushed less than 1 day ago. The 'Repository overview' section is partially visible at the bottom, indicating that an overview can be added to describe the image and its usage.

10. Jenkins CI/CD Pipeline

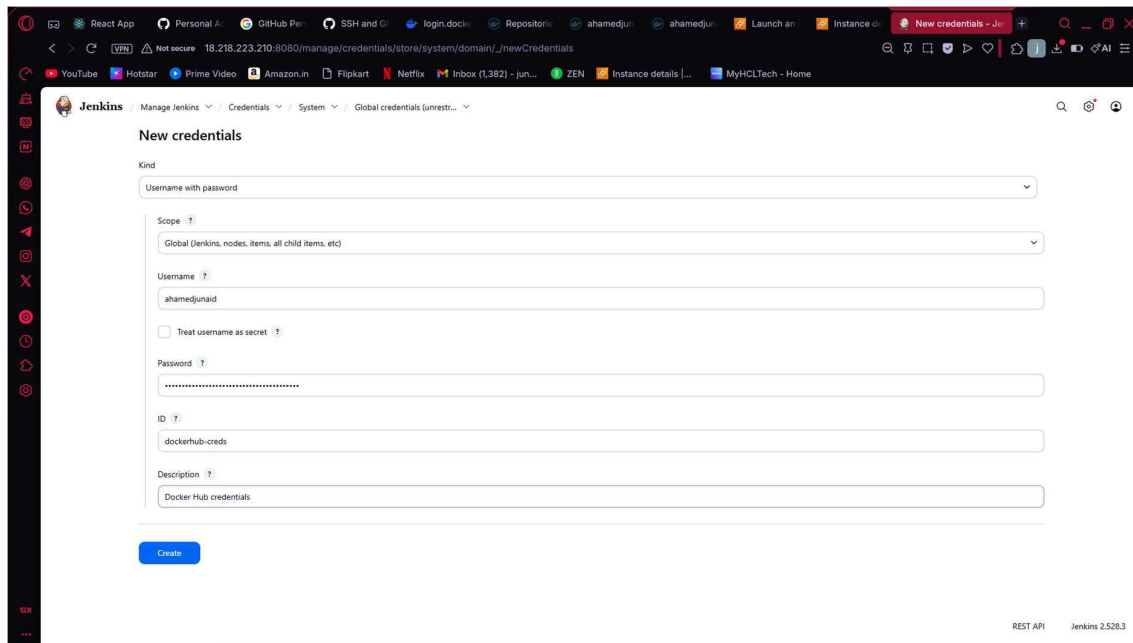
10.1 Jenkins Installation

Jenkins was installed on AWS EC2 and accessed via port 8080.



10.2 Jenkins Credentials Configuration

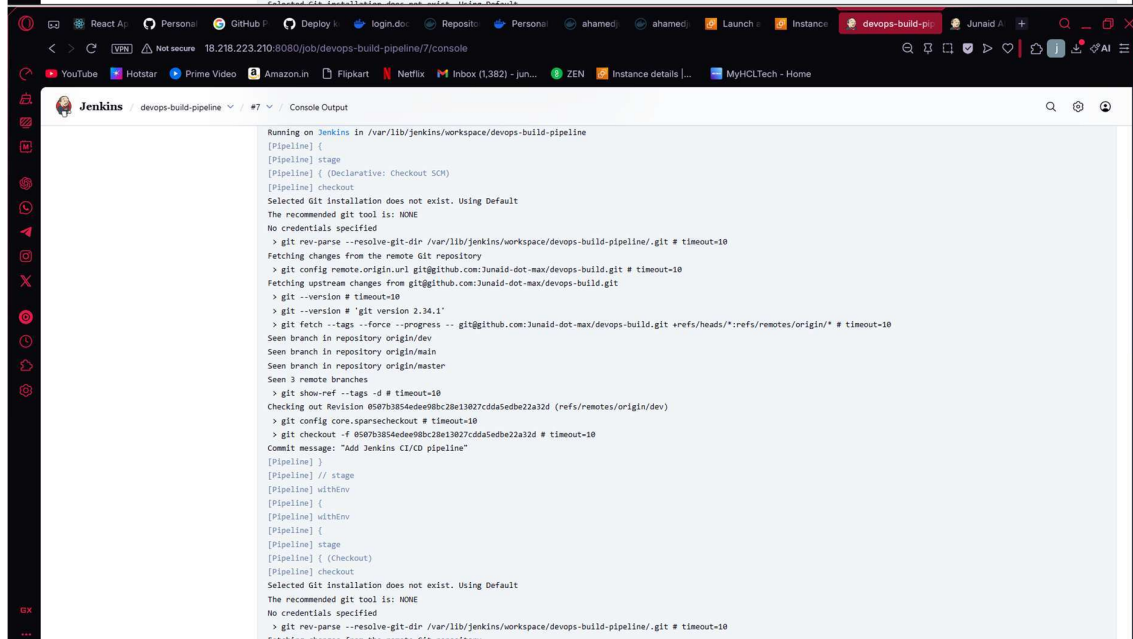
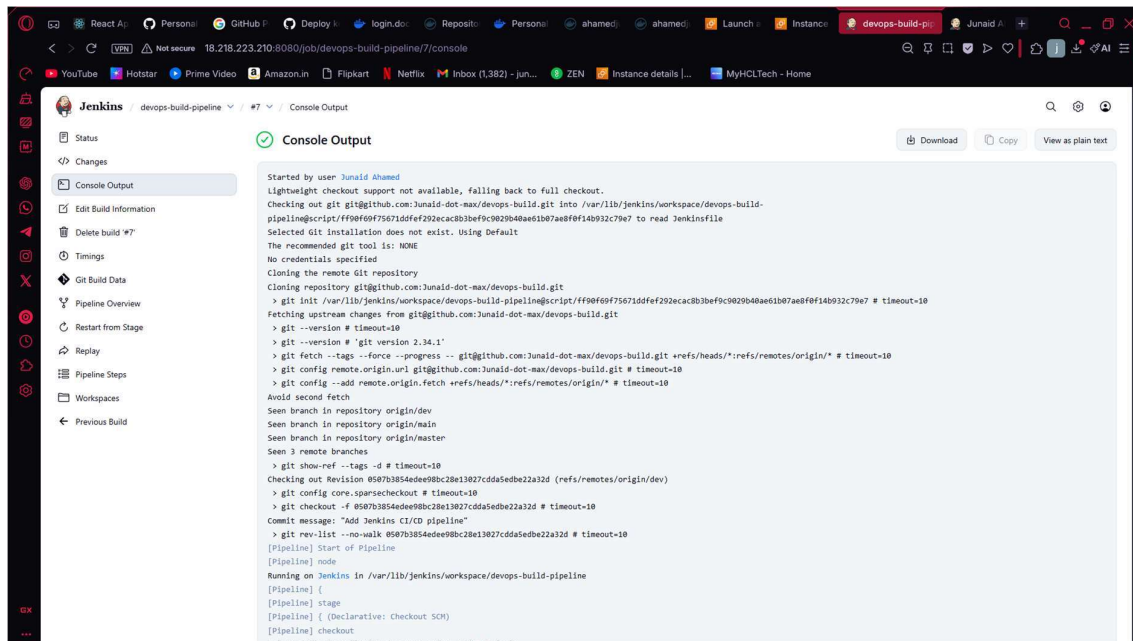
- Docker Hub credentials added
- GitHub SSH key configured
- Secure authentication enabled



10.3 Jenkins Pipeline Execution

Pipeline Logic:

- Push to dev → Build & push DEV image
- Merge to main → Build & push PROD image



The top screenshot displays the Jenkins console output for a build named 'devops-build-pipeline' with job number #7. The output shows the following steps and results:

- Step 4/5: EXPOSE 80
- Step 5/5: CMD ["nginx", "-g", "daemon off;"]
- Successfully built 60cf5cb000a
- Successfully tagged react-app:latest
- [Pipeline] // stage
- [Pipeline] stage
- [Pipeline] { (Push to Docker Hub)
- [Pipeline] script
- [Pipeline] {
- [Pipeline] withCredentials
- Masking supported pattern matches of \$DOCKER_PASS
- [Pipeline] {
- [Pipeline] sh
- + echo ****
- + docker login -u ahamedjunaid --password-stdin
- Login Succeeded
- [Pipeline] }
- [Pipeline] // withCredentials
- [Pipeline] }
- [Pipeline] // script
- [Pipeline] }
- [Pipeline] // stage
- [Pipeline] }
- [Pipeline] // withEnv
- [Pipeline] }
- [Pipeline] // withEnv
- [Pipeline] }
- [Pipeline] // node
- [Pipeline] End of Pipeline
- Finished: SUCCESS

The bottom screenshot shows the 'Build History of Jenkins' page. It lists several builds with their status and time taken:

S	Build	Time Since 1	Status
✓	devops-build-pipeline #7	2 min 53 sec	stable
✓	devops-build-pipeline #6	10 min	back to normal
✗	devops-build-pipeline #5	18 min	broken for a long time
✗	devops-build-pipeline #4	20 min	broken for a long time
✗	devops-build-pipeline #3	34 min	broken for a long time
✗	devops-build-pipeline #2	39 min	broken for a long time
✗	devops-build-pipeline #1	39 min	broken since this build

11. AWS EC2 Deployment

11.1 EC2 Instance Details

- Instance Type: t3.micro
- OS: Ubuntu 22.04
- Docker & Jenkins installed

aws

Search

[Alt+S]

United States (Ohio)

Account ID: 7275-9813-4512

Junaid_Ahmed

EC2

Instances

i-0dd1a7bf7eb2ec1a7

EC2

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Capacity Manager

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Instance summary for i-0dd1a7bf7eb2ec1a7 (Jenkins-server)

Connect Instance state Actions

Updated less than a minute ago

Instance ID

i-0dd1a7bf7eb2ec1a7

Public IPv4 address

18.218.223.210 | open address

Private IPv4 addresses

172.31.37.47

IPv6 address

-

Instance state

Running

Public DNS

ec2-18-218-223-210.us-east-2.compute.amazonaws.com | open address

Hostname type

IP name: ip-172-31-37-47.us-east-2.compute.internal

Private IP DNS name (IPv4 only)

ip-172-31-37-47.us-east-2.compute.internal

Answer private resource DNS name

IPv4 (A)

Instance type

t3.micro

Auto-assigned IP address

18.218.223.210 [Public IP]

VPC ID

vpc-0a7525c5de367426e

Elastic IP addresses

-

IAM Role

-

Subnet ID

subnet-0ee588ac4248abcc9

AWS Compute Optimizer finding

Opt-in to AWS Compute Optimizer for recommendations. | Learn more

IMDSv2

Required

Instance ARN

arn:aws:ec2:us-east-2:727598134512:instance/i-0dd1a7bf7eb2ec1a7

Auto Scaling Group name

-

Operator

-

Managed

false

11.2 Security Group Configuration

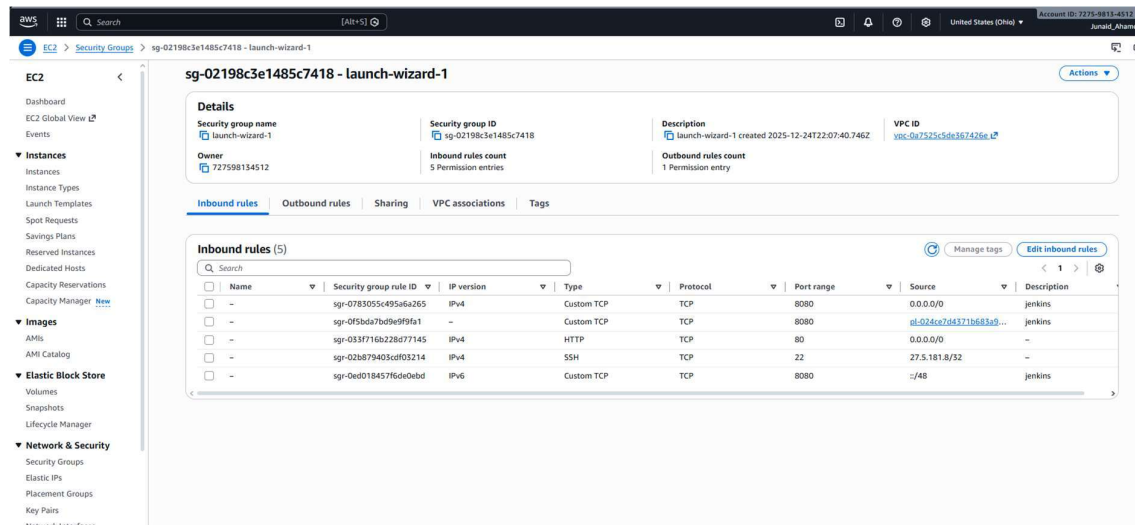
Security Groups were configured as per requirement:

Port	Purpose	Access
------	---------	--------

80	Application	0.0.0.0/0
----	-------------	-----------

22	SSH	My IP
----	-----	-------

8080	Jenkins	My IP
------	---------	-------

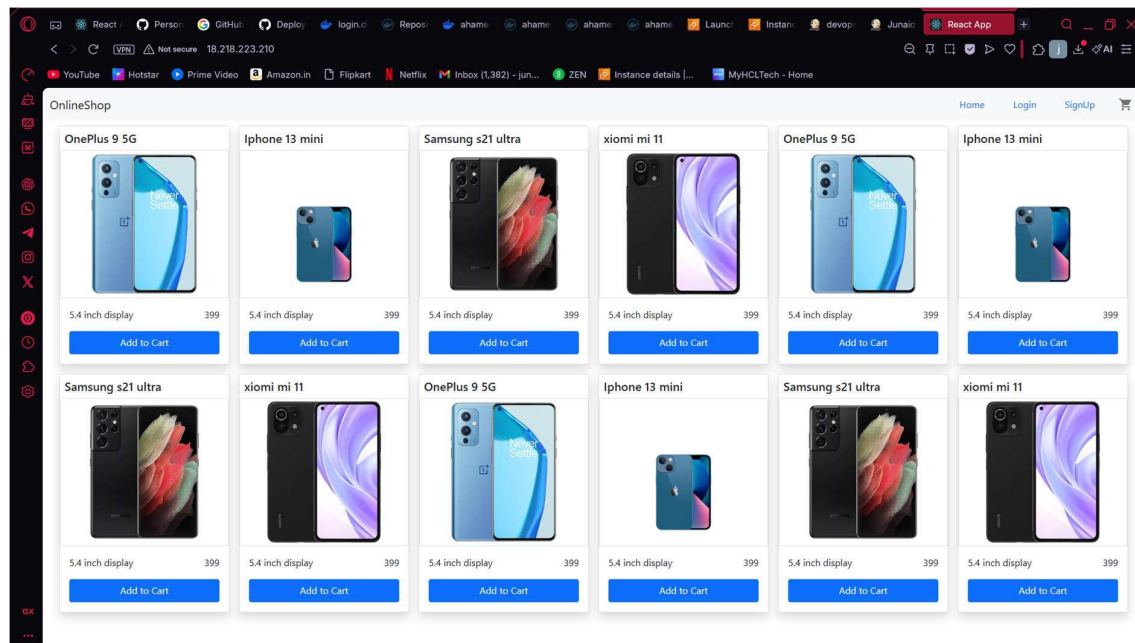


12. Application Deployment Verification

The application was successfully deployed and accessed via EC2 public IP.

Application URL:

http://18.218.223.210



13. Monitoring & Health Check

Application health was monitored by:

- Docker container status
- HTTP availability check

The application remained continuously available, confirming deployment stability.

```
ubuntu@ip-172-31-37-47:~$ sudo systemctl start docker
ubuntu@ip-172-31-37-47:~$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-37-47:~$ newgrp docker
ubuntu@ip-172-31-37-47:~$ docker ps
docker images
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
REPOSITORY    TAG        IMAGE ID          CREATED        SIZE
ahamedjunaid/react-app    prod       60cf5cb0600a      About an hour ago    56.4MB
react-app      latest     60cf5cb0600a      About an hour ago    56.4MB
nginx          alpine     04da2b0513cd      6 days ago         53.7MB
ubuntu@ip-172-31-37-47:~$ docker pull ahamedjunaid/devops-build:dev
dev: Pulling from ahamedjunaid/devops-build
1074353eec0d: Already exists
25f453064fd3: Already exists
567f84da6fbd: Already exists
da7c973d8b92: Already exists
33f95a0f3229: Already exists
085c5e5aaa8e: Already exists
0abf9e567266: Already exists
de54cb821236: Already exists
ff20fd7fb92b: Pull complete
13caef3ab764: Pull complete
Digest: sha256:49616d83149a6f72f4c7ddfd7184fbf681dc9ec02c0be98ee0c0f57a86e70fab
Status: Downloaded newer image for ahamedjunaid/devops-build:dev
docker.io/ahamedjunaid/devops-build:dev
ubuntu@ip-172-31-37-47:~$ docker run -d \
  --name react-app \
  -p 80:80 \
  ahamedjunaid/devops-build:dev
637214ca035dfb94eab1c2e7613833df7dbf968e2386e2d9eb52b5f3b53b4ea9
ubuntu@ip-172-31-37-47:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
637214ca035d   ahamedjunaid/devops-build:dev  "/docker-entrypoint..."  12 seconds ago    Up 11 seconds    0.0.0.0:80->80/tcp, [::]:80->80/tcp    react-app
ubuntu@ip-172-31-37-47:~$
```

14. Conclusion

This project demonstrates a complete DevOps lifecycle implementation including containerization, CI/CD automation, cloud deployment, secure access control, and monitoring. All requirements specified by GUVI were successfully implemented and validated.

15. Tools & Technologies Used

- Docker
- Docker Compose
- Jenkins
- Git & GitHub
- Docker Hub
- AWS EC2
- Ubuntu Linux

 **END OF REPORT**