

## Project: Reactjs E-commerce Application

### 1.Docker:

1.Docker and Docker-compose Installed:

```
root@ip-172-31-24-31:~# docker --version && docker-compose --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
docker-compose version 1.29.2, build unknown
root@ip-172-31-24-31:~# █
```

### 2.Source Code File:

```
root@ip-172-31-24-31:~/Project1# ls
build
root@ip-172-31-24-31:~/Project1# █
```

### 3.Creating Dockerfile:

```
root@ip-172-31-24-31:~/Project1# nano Dockerfile█
```

#### 4.DockerFile Code:

```
GNU nano 7.2                                            Dockerfile
FROM nginx:alpine
RUN rm -rf /usr/share/nginx/html/*
COPY build/ /usr/share/nginx/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

#### 5.Creating Docker Image:

```
root@ip-172-31-24-31:~/Project1# docker build -t static-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 1/6 : FROM nginx:alpine
alpine: Pulling from library/nginx
9824c27679d3: Pull complete
6bc572a340ec: Pull complete
403e3f251637: Pull complete
9adfbae99cb7: Pull complete
7a8a46741e18: Pull complete
c9ebe2ff2d2c: Pull complete
a992fbc61ecc: Pull complete
cb1ff4086f82: Pull complete
Digest: sha256:42a516af16b852e33b7682d5ef8acbd5d13fe08fecadc7ed98605ba5e3b26ab8
Status: Downloaded newer image for nginx:alpine
--> 4a86014ec699
Step 2/6 : WORKDIR /usr/share/nginx/html
--> Running in df7b442476fa
--> Removed intermediate container df7b442476fa
--> afbdd68e0ffd
Step 3/6 : RUN rm -rf ./*
```

#### 6.Docker Image Created:

```
root@ip-172-31-24-31:~/Project1# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
static-app      latest       96a05b569493   About a minute ago  55.9MB
nginx           alpine       4a86014ec699   9 days ago    52.5MB
root@ip-172-31-24-31:~/Project1#
```

## 7.Docker-compose Code:

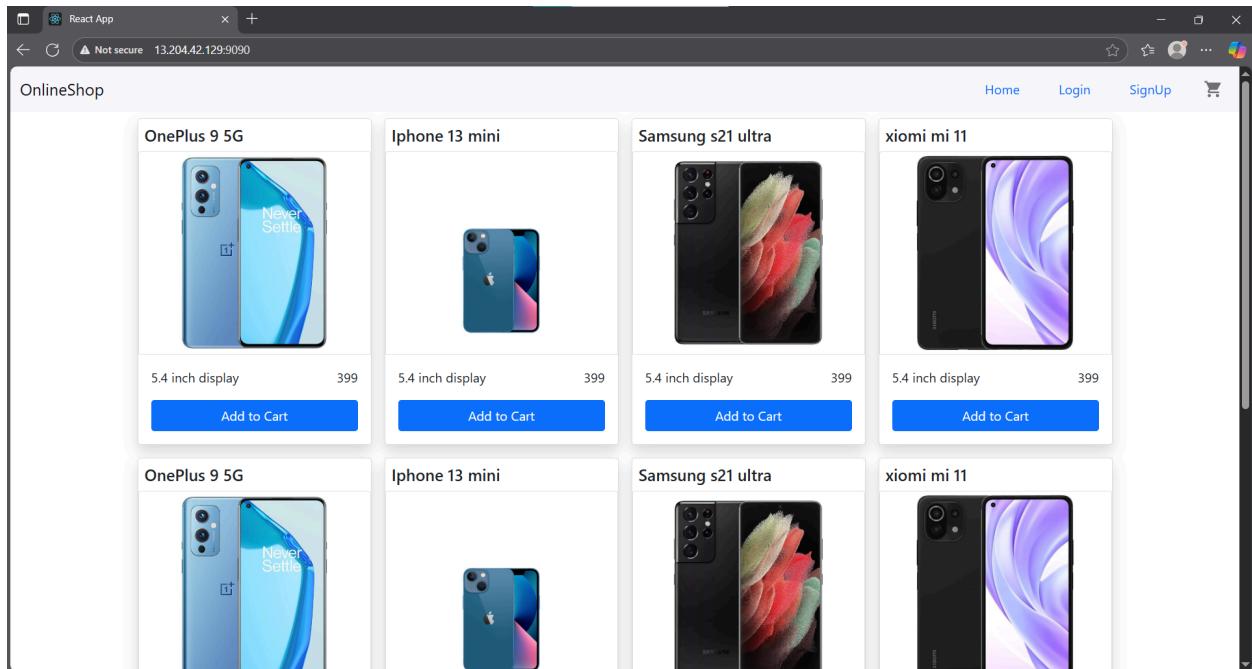
```
GNU nano 7.2                                     docker-compose.yaml
version: '3.8'

services:
  Dev:
    image: static-app:latest
    ports:
      - "9090:80"
```

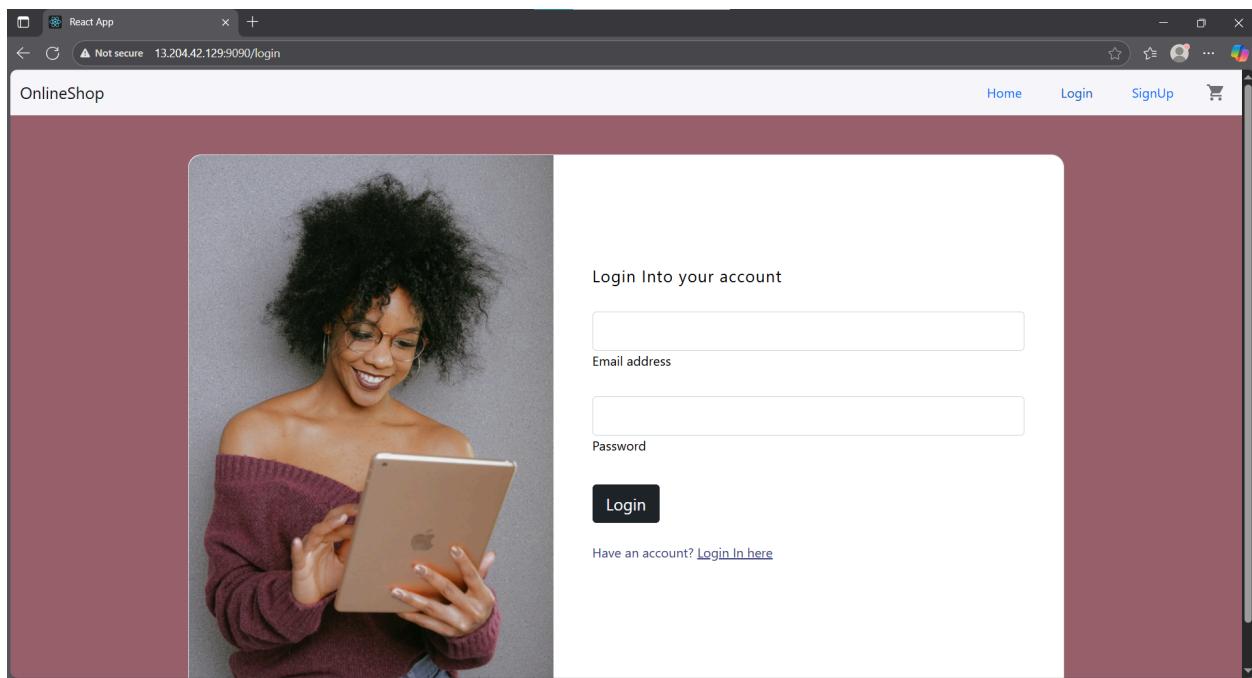
## 8.Running The Docker-compose:

```
root@ip-172-31-24-31:~/Project1# docker-compose up -d
Creating network "project1_default" with the default driver
Creating project1_Dev_1 ... done
root@ip-172-31-24-31:~/Project1#
```

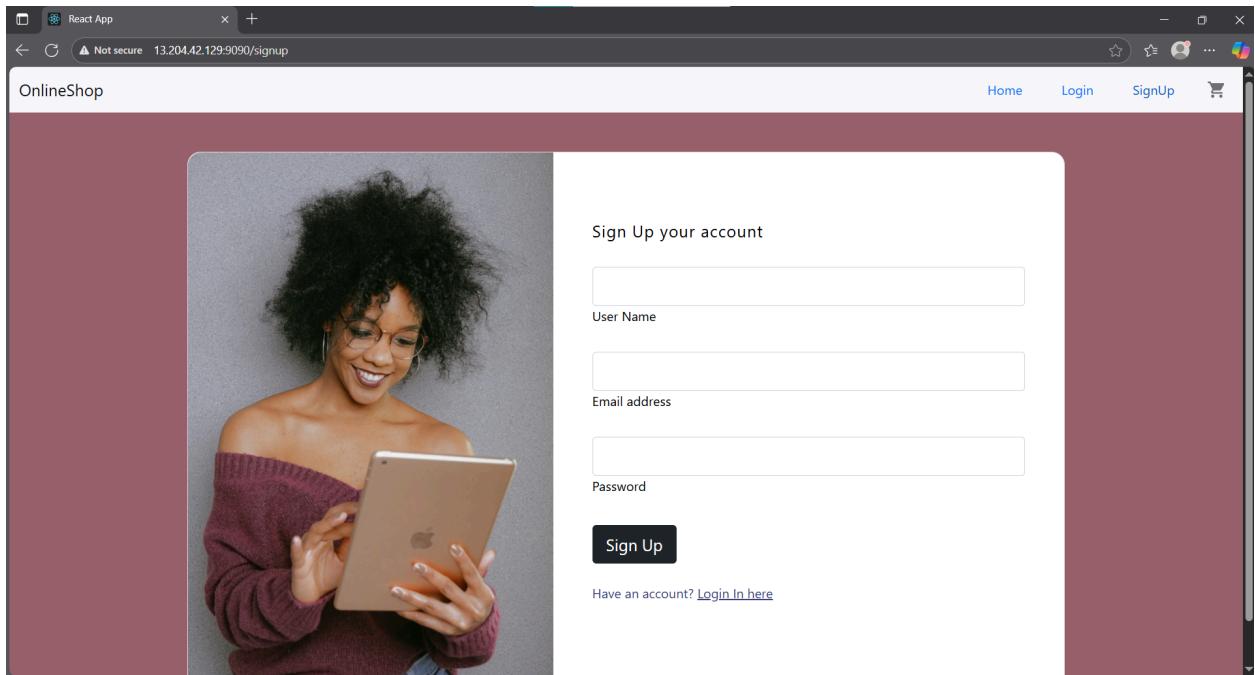
## 9.Docker Application Output 1:



## 10.Docker Application Output 2:



## 11.Docker Application Output 3:



## 2.Bash\_Scripting:

### 1.Creating Script file to Build Docker Image:

```
GNU nano 7.2                                     build.sh *
#!/bin/bash
IMAGE_NAME="static-web"
docker build -t $IMAGE_NAME .
```

## 2.Creating Script file to Run Docker Container:

```
GNU nano 7.2                                            deploy.sh *
```

```
#!/bin/bash

IMAGE_NAME="static-web"
CONTAINER_NAME="static_application"

docker rm -f $CONTAINER_NAME 2>/dev/null || true
docker run -idt -p "9091:80" --name $CONTAINER_NAME $IMAGE_NAME
```

## 3.Using Docker Ignore to ignore some files in Docker Build:

```
GNU nano 7.2                                            .dockerignore *
```

```
*.sh
*.yaml
```

## 4.Giving Executable permission to Build and Deploy file:

```
root@ip-172-31-24-31:~/Project1# chmod +x build.sh deploy.sh
root@ip-172-31-24-31:~/Project1#
```

## 5.Running Build File:

```
root@ip-172-31-24-31:~/Project1# ./build.sh
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 3.42MB
Step 1/5 : FROM nginx:alpine
--> 4a86014ec699
Step 2/5 : RUN rm -rf /usr/share/nginx/html/*
--> Using cache
--> dcf90923529c
Step 3/5 : COPY build/ /usr/share/nginx/html/
--> Using cache
--> 8ae700e55a18
Step 4/5 : EXPOSE 80
--> Using cache
--> 8d976824691c
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
--> Using cache
--> 836000789d26
Successfully built 836000789d26
Successfully tagged static-web:latest
root@ip-172-31-24-31:~/Project1#
```

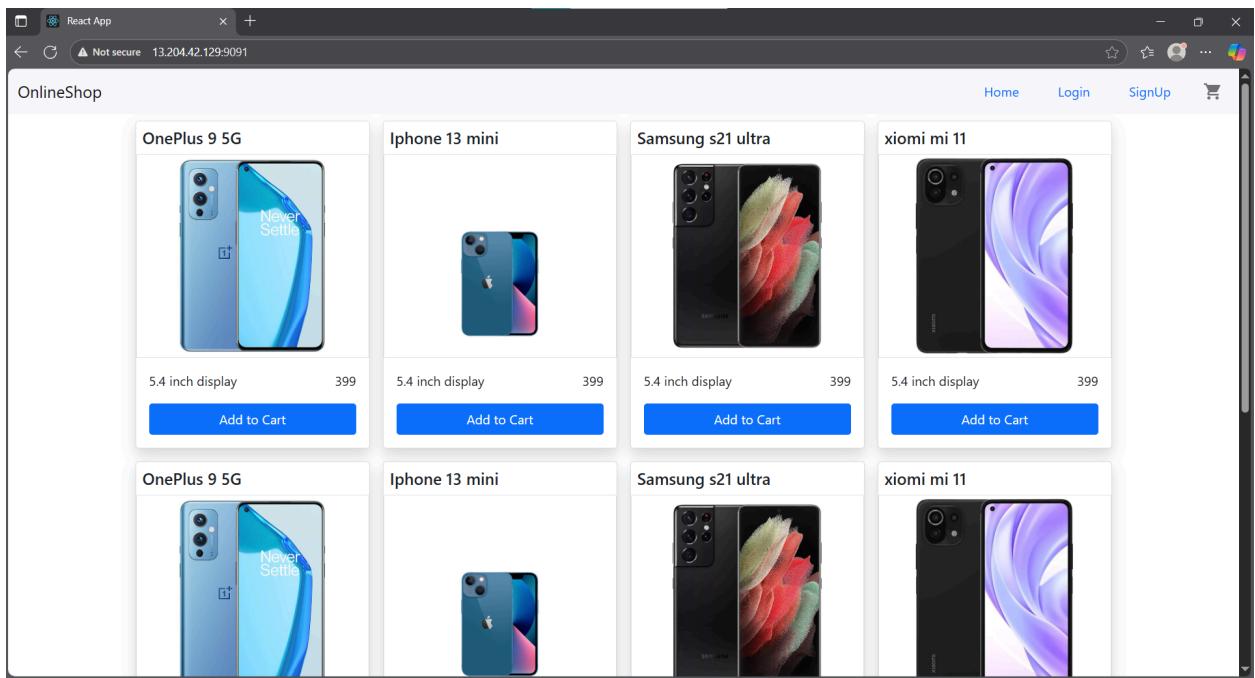
## 6.Docker Image created:

```
root@ip-172-31-24-31:~/Project1# docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
static-web      latest       fded10026b8c  17 seconds ago  55.1MB
nginx           alpine       4a86014ec699  9 days ago    52.5MB
root@ip-172-31-24-31:~/Project1#
```

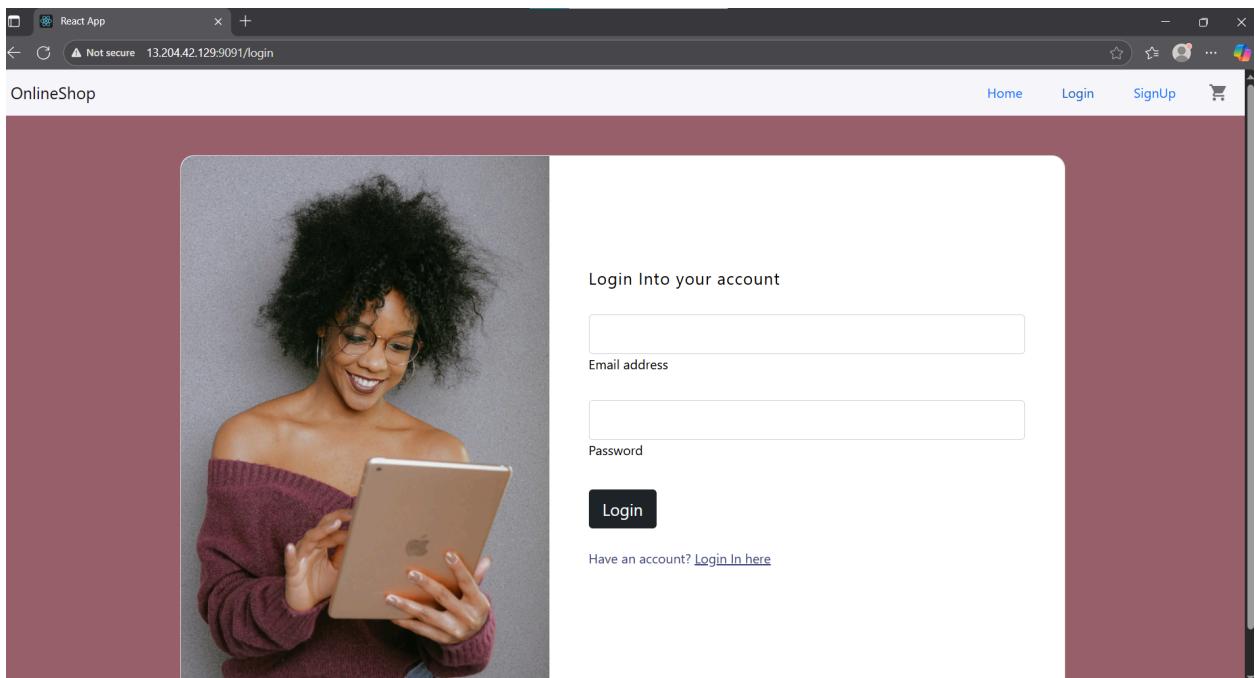
## 7.Docker Container Created:

```
root@ip-172-31-24-31:~/Project1# ./deploy.sh
76e6a35d87d1fead112387509a039673b2892befbbce524b03231cd2fff83f72
root@ip-172-31-24-31:~/Project1# docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED        STATUS        PORTS          NAMES
76e6a35d87d1   static-web     "/docker-entrypoint..."  7 seconds ago   Up 6 seconds  0.0.0.0:9091->80/tcp, [::]:9091->80/tcp   static_application
root@ip-172-31-24-31:~/Project1#
```

## 8.Output 1:



## 9.Output 2:



## 10.Output 3:



OnlineShop

Home Login SignUp

Sign Up your account

User Name

Email address

Password

Sign Up

Have an account? [Login In here](#)

A screenshot of a web browser showing a sign-up form for an "OnlineShop". The page has a dark red header and footer. On the left side of the main content area is a photograph of a woman with curly hair, wearing a maroon off-the-shoulder sweater, smiling and holding a gold-colored tablet. To the right of the photo is a white sign-up form with three input fields labeled "User Name", "Email address", and "Password", each with a corresponding text input box. Below the input fields is a black "Sign Up" button. At the bottom of the form is a link "Have an account? [Login In here](#)". The browser's address bar shows "Not secure 13.204.42.129:9091/signup".

### 3.Version Control:

#### 1.Creating GitHub Repository:

Create a new repository [Preview](#) [Switch back to classic experience](#)

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

1 General

Owner \* Anand-kumar-git / Repository name \* Project1  Project1 is available.

Great repository names are short and memorable. How about [super-duper-octo-rotary-phone](#)?

Description

0 / 350 characters

2 Configuration

Choose visibility \* Public

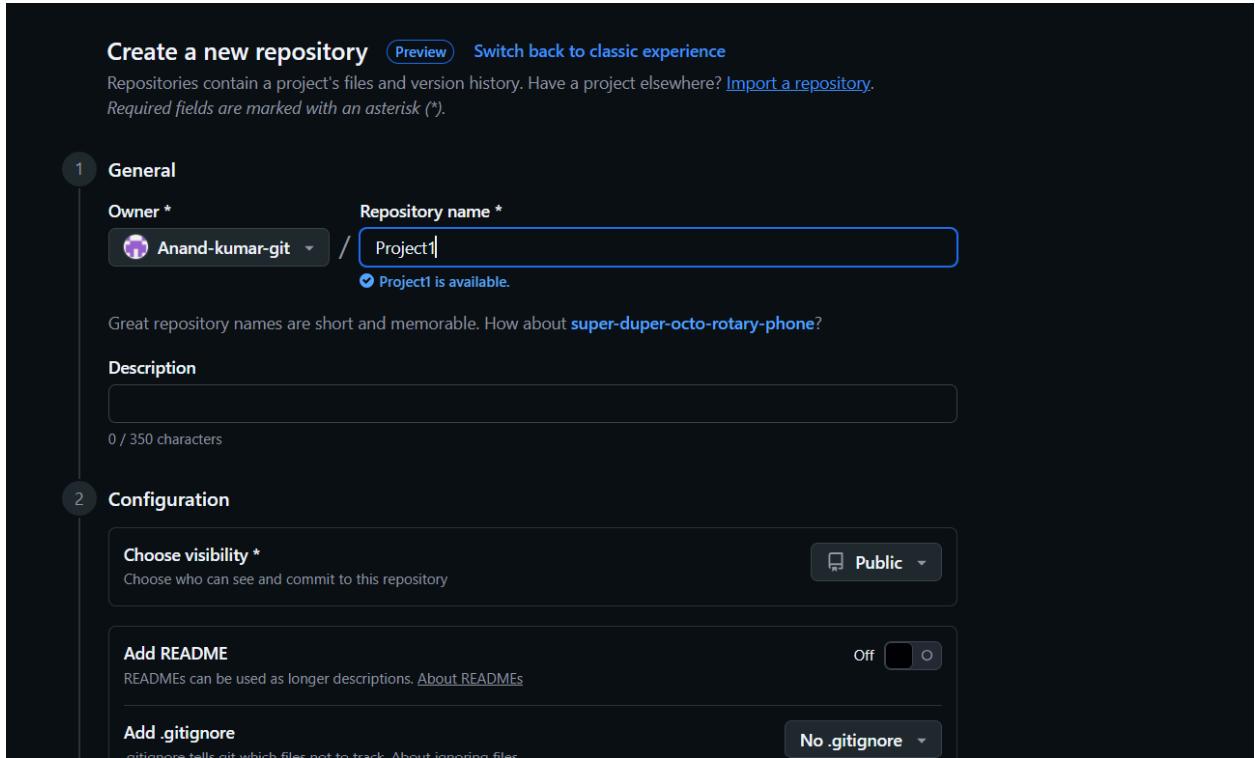
Choose who can see and commit to this repository

Add README Off

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore No .gitignore

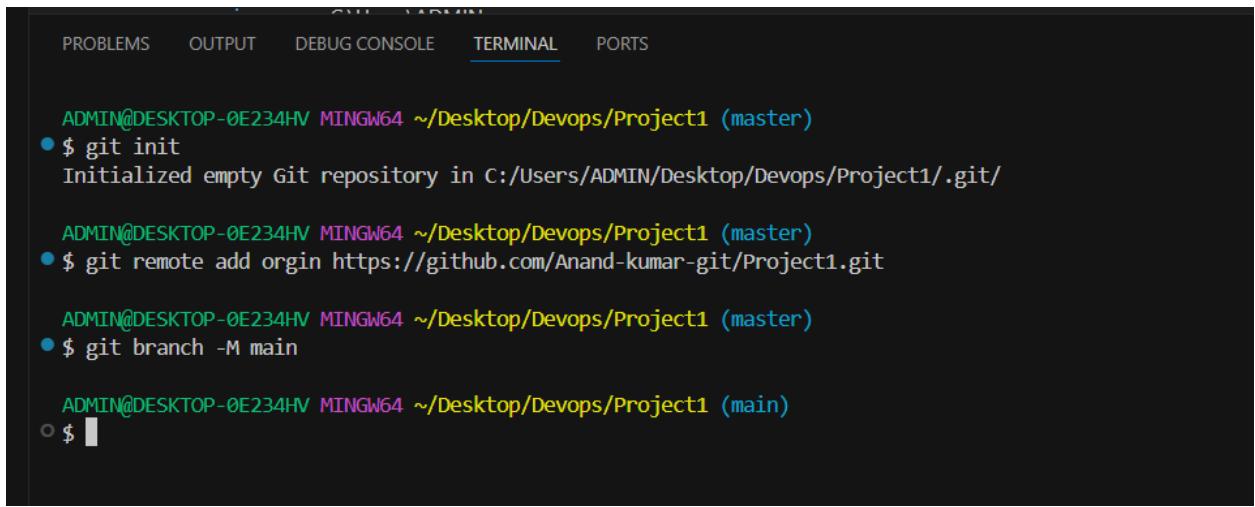
.gitignore tells git which files not to track. [About ignoring files](#)



#### 2.Cloning the git Repository:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (master)
• $ git init
Initialized empty Git repository in C:/Users/ADMIN/Desktop/Devops/Project1/.git/
ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (master)
• $ git remote add origin https://github.com/Anand-kumar-git/Project1.git
ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (master)
• $ git branch -M main
ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (main)
```

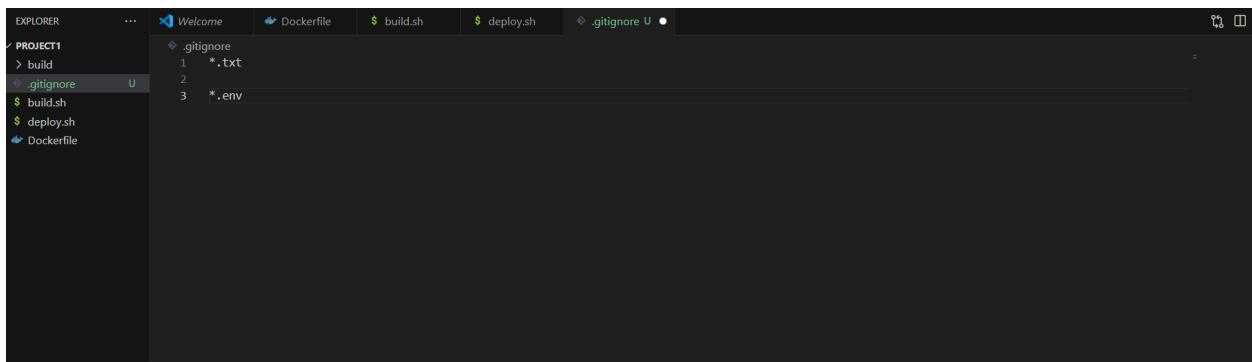


### 3.Creating Dev Branch:

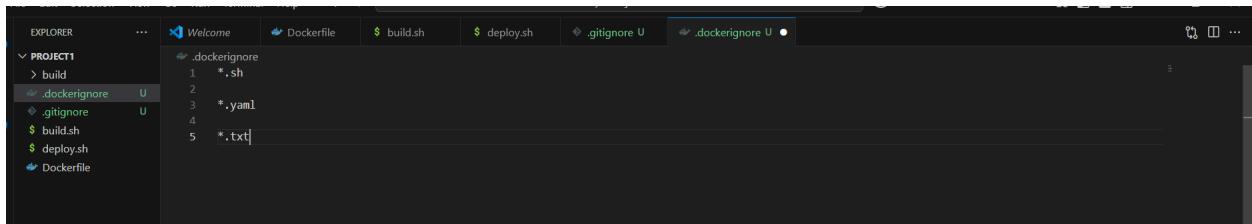
```
ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (main)
$ git checkout -b dev
Switched to a new branch 'dev'

ADMIN@DESKTOP-0E234HV MINGW64 ~/Desktop/Devops/Project1 (dev)
$
```

### 4.Using Git ignore file:



### 5.Using .dockerignore file:



## 4.Docker\_Hub:

### 1.Creating Dev Repository in Dockerhub:

The screenshot shows the 'Create repository' form on the Docker Hub website. The repository name is 'dev'. The visibility is set to 'Public' (radio button selected). The 'Pushing images' section contains CLI commands: `docker tag local-image:tagname new-repo:tagname` and `docker push new-repo:tagname`. A note says to replace `tagname` with the desired image repository tag. The 'Create' button is visible at the bottom right.

### 2.Creating Prod Private Repository:

The screenshot shows the 'Create repository' form on the Docker Hub website. The repository name is 'prod'. The visibility is set to 'Private' (radio button selected). The 'Pushing images' section contains CLI commands: `docker tag local-image:tagname new-repo:tagname` and `docker push new-repo:tagname`. A note says to replace `tagname` with the desired image repository tag. The 'Create' button is visible at the bottom right.

### 3.Dev and Prod Repository Created:

The screenshot shows the Docker Hub interface under the 'My Hub' tab. On the left, there's a sidebar with options like 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The 'Repositories' option is selected. The main area displays a table of repositories. The first row shows 'anand20003/prod' with 'Last Pushed' at '2 minutes ago', 'Visibility' as 'Private', and 'Scout' status as 'Inactive'. The second row shows 'anand20003/dev' with 'Last Pushed' at '4 minutes ago', 'Visibility' as 'Public', and 'Scout' status as 'Inactive'. A search bar at the top right says 'Search Docker Hub'.

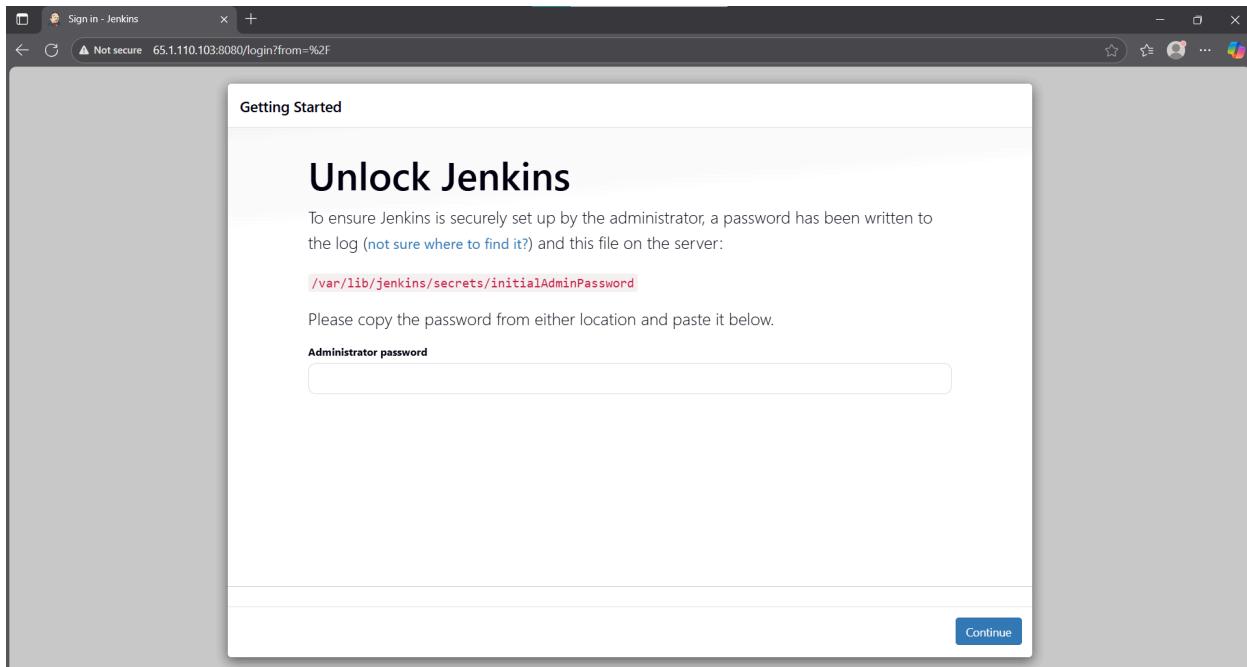
## 5.Jenkins:

### 1.Jenkins Installed:

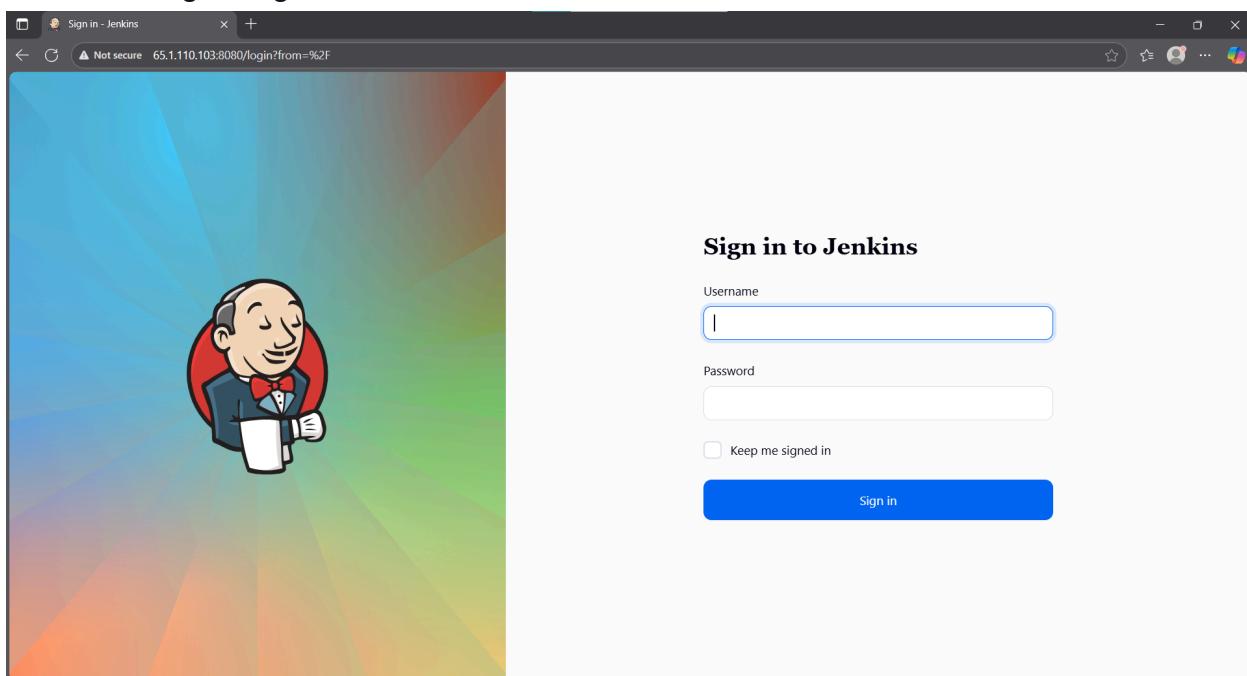
```
root@ip-172-31-28-7:~# systemctl status jenkins
● Jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Sat 2025-08-23 09:19:53 UTC; 1min 37s ago
    Main PID: 4754 (java)
       Tasks: 45 (limit: 4515)
      Memory: 618.5M (peak: 632.3M)
        CPU: 23.118s
       CGroup: /system.slice/jenkins.service
           └─4754 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 23 09:19:46 ip-172-31-28-7 jenkins[4754]: cd3d0ca35989445680e5bb27639bd7a8
Aug 23 09:19:46 ip-172-31-28-7 jenkins[4754]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 23 09:19:46 ip-172-31-28-7 jenkins[4754]: ****
Aug 23 09:19:46 ip-172-31-28-7 jenkins[4754]: 2025-08-23 09:19:53.063+0000 [id=33]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
Aug 23 09:19:53 ip-172-31-28-7 jenkins[4754]: 2025-08-23 09:19:53.082+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Aug 23 09:19:53 ip-172-31-28-7 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Aug 23 09:19:54 ip-172-31-28-7 jenkins[4754]: 2025-08-23 09:19:54.682+0000 [id=49]      INFO      h.m.DownloadService$Downloadable#load: Obtained the updated data
Aug 23 09:19:54 ip-172-31-28-7 jenkins[4754]: 2025-08-23 09:19:54.683+0000 [id=49]      INFO      hudson.util.Retrier#start: Performed the action check updates se
lines 1-20/20 (END)
```

## 2. Accessing Jenkins With Port 8080:



## 3. Jenkins Login Page:



#### 4. Installing Required plugins:

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Plugins' section. A sidebar on the left lists options: 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The 'Download progress' option is currently selected. The main area displays a table of installed plugins with their status as 'Success' indicated by green checkmarks. The listed plugins include Matrix Authorization Strategy, LDAP, Email Extension, Mailer, Dark Theme, Loading plugin extensions, Cloud Statistics, Authentication Tokens API, Docker Commons, Apache HttpComponents Client 5.x API, Commons Compress API, Docker API, Docker, Loading plugin extensions, CloudBees Docker Hub/Registry Notification, and Loading plugin extensions.

→ Go back to the top page  
(you can start using the installed plugins right away)

→  Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.524

#### 5. Adding DockerHub Credential in Jenkins:

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Credentials' section. A breadcrumb navigation path is visible: Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...). The main area is titled 'New credentials' and shows a 'Kind' dropdown set to 'Username with password'. The form fields are as follows: 'Scope' dropdown set to 'Global (Jenkins, nodes, items, all child items, etc)'; 'Username' input field containing 'anand20003'; a checked checkbox 'Treat username as secret'; 'Password' input field containing '\*\*\*\*\*'; 'ID' input field containing 'Docker-Hub' (which is highlighted with a blue border); and an empty 'Description' input field. A blue 'Create' button is at the bottom left.

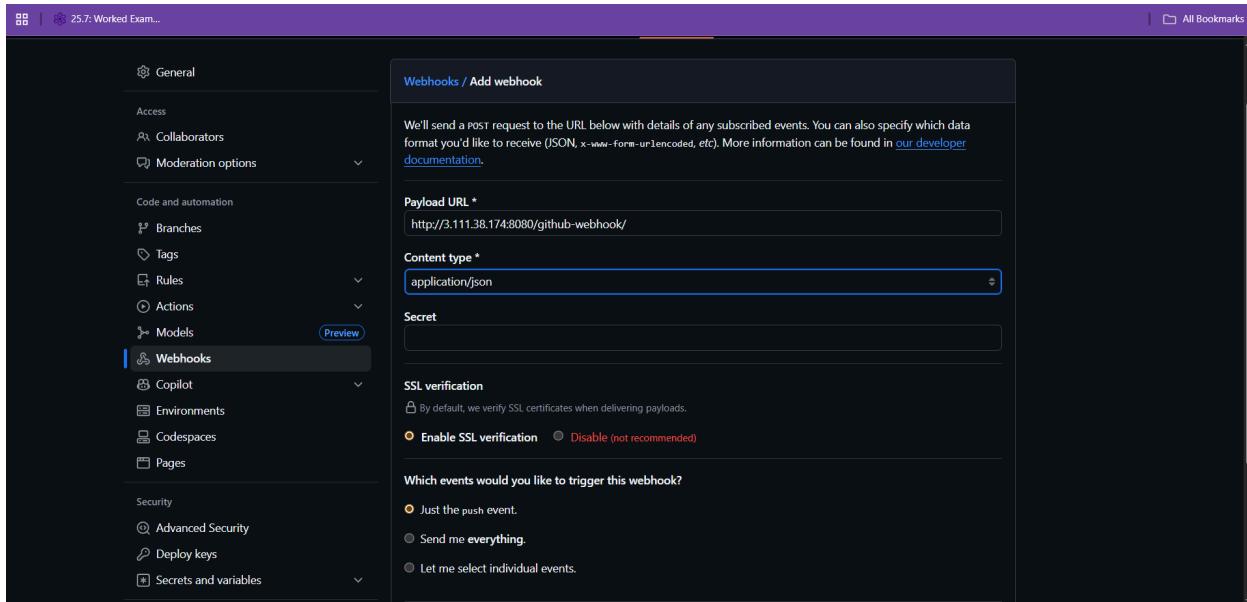
## 6.Creating Jenkins MultiBranch Pipeline:

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there is a search bar and a 'Not secure' warning. Below it, the URL is 3.111.38.174:8080/view/all/newJob. The main area is titled 'New Item' and has a sub-section 'Select an item type'. It lists several options: 'Freestyle project' (classic job type), 'Pipeline' (orchestrates long-running activities), 'Multi-configuration project' (suitable for multiple configurations), 'Folder' (creates a container for nested items), and 'Multibranch Pipeline' (creates a set of Pipeline projects according to detected branches). The 'Multibranch Pipeline' option is highlighted with a blue border. At the bottom of the form is a blue 'OK' button.

## 7.Multibranch Pipeline Configuration:

The screenshot shows the Jenkins 'Configuration' screen for a Multibranch Pipeline project named 'Project1'. The left sidebar lists configuration options: General, Branch Sources (which is selected and highlighted in grey), Build Configuration, Scan Multibranch Pipeline Triggers, Orphaned Item Strategy, Appearance, Health metrics, and Properties. The main panel is titled 'Branch Sources' and contains a 'Git' section with a 'Project Repository' field containing the URL https://github.com/Anand-kumar-git/Project1.git. Below this is a 'Credentials' dropdown set to 'none'. A 'Behaviors' section includes a 'Discover branches' field with an 'Add' button. At the bottom are 'Save' and 'Apply' buttons.

## 8.Adding Github Webhook:



## 9.Jenkinsfile First part:

The screenshot shows a code editor displaying a Jenkinsfile. The file defines a pipeline with several stages: 'Cleanup Workspace', 'Checkout Code', and 'Build Docker Image'. In the 'Build Docker Image' stage, a script block uses conditional logic based on the environment variable `env.BRANCH_NAME` to run different Docker commands ('tag static-web' or 'tag static-web'). The code editor interface includes tabs for other files like `build.sh`, `deploy.sh`, `.dockerignore`, `.gitignore`, and `Dockerfile`. The bottom status bar indicates the code is in Groovy syntax, with 17 lines and 33 columns.

```
1 v pipeline {
2   agent any
3
4 v   environment {
5     DEV_REPO = "anand20003/dev"
6     PROD_REPO = "anand20003/prod"
7     IMAGE_TAG = "latest"
8   }
9
10 v   stages {
11     stage('Cleanup Workspace') {
12       steps {
13         deleteDir()
14     }
15   }
16
17 v   stage('Checkout Code') {
18     steps {
19       checkout scm
20     }
21   }
22
23 v   stage('Build Docker Image') {
24     steps {
25       script {
26         sh 'chmod +x build.sh'
27         sh './build.sh'
28
29         if (env.BRANCH_NAME == "dev") {
30           sh "docker tag static-web $DEV_REPO:$IMAGE_TAG"
31         } else if (env.BRANCH_NAME == "main") {
32           sh "docker tag static-web $PROD_REPO:$IMAGE_TAG"
33         }
34       }
35     }
36   }
37 }
```

## 10.Jenkinsfile Second part:

```
38     stage('Login to DockerHub') {
39         steps [
40             withCredentials([usernamePassword(credentialsId: 'Docker-Hub',
41                                         usernameVariable: 'DOCKER_USER',
42                                         passwordVariable: 'DOCKER_PASS')])
43             sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
44         ]
45     }
46 }
47
48 stage('Push Docker Image') {
49     steps {
50         script {
51             if (env.BRANCH_NAME == "dev") {
52                 sh "docker push $DEV_REPO:$IMAGE_TAG"
53             } else if (env.BRANCH_NAME == "main") {
54                 sh "docker push $PROD_REPO:$IMAGE_TAG"
55             }
56         }
57     }
58 }
59
60 stage('Deploy To AWS') {
61     steps {
62         script {
63             sh 'chmod +x deploy.sh'
64             sh './deploy.sh'
65         }
66     }
67 }
68 }
69 }
70 }
```

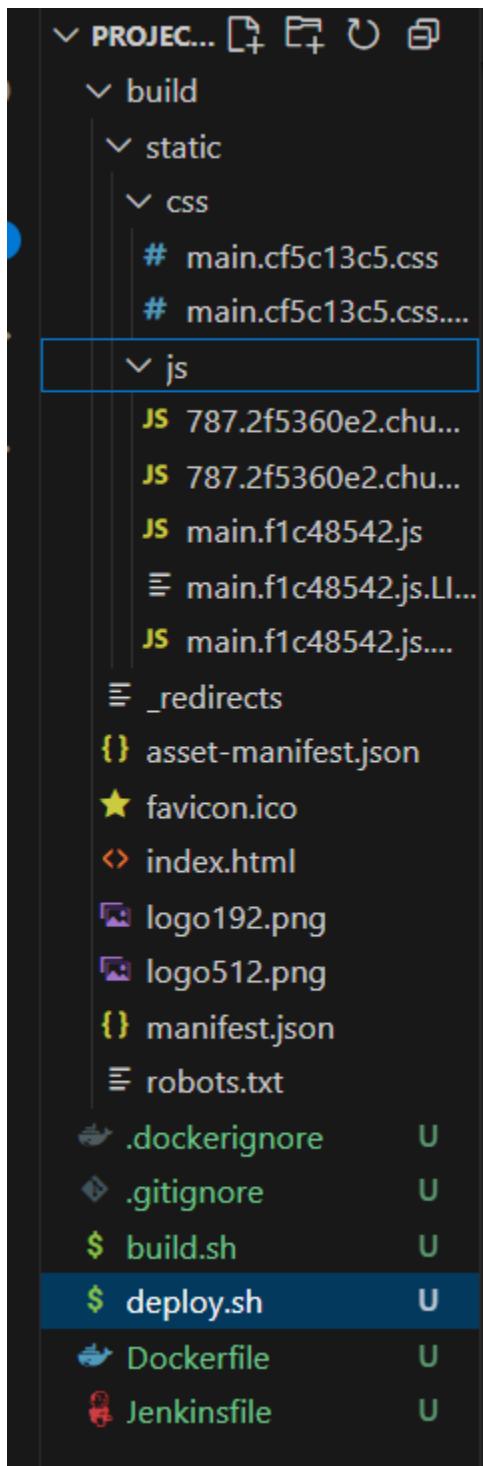
Ln 45, Col 14 Spaces: 4 UTF-8 LF {} Groovy

## 11.Dockerfile:

```
FROM nginx:alpine
RUN rm -rf /usr/share/nginx/html/*
# Copy only prebuilt React app
COPY build/. /usr/share/nginx/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Ln 10, Col 1 Spaces: 4 UTF-8 LF {} Docker

## 12.File Structure:



### 13.Adding Jenkins to Docker Group:

```
root@ip-172-31-28-7:~# usermod -aG docker jenkins
root@ip-172-31-28-7:~#
```

### 14.Pushed to dev Branch:

```
ADMIN@DESKTOP-0E234HV MINGW64 ~/Project1 (dev)
● $ git commit -m "First Commit"
[dev b1a4281] First Commit
 1 file changed, 1 insertion(+), 1 deletion(-)

ADMIN@DESKTOP-0E234HV MINGW64 ~/Project1 (dev)
● $ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 288 bytes | 288.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Anand-kumar-git/Project1.git
 46d7993..b1a4281 dev -> dev

ADMIN@DESKTOP-0E234HV MINGW64 ~/Project1 (dev)
○ $
```

### 15.Pipeline Build in dev Branch:

The screenshot shows the Jenkins interface for the Project1/dev pipeline. The top navigation bar includes links for Jenkins, Project1, and dev. The main content area displays the following information:

- Status:** A green checkmark icon next to the text "dev".
- Full project name:** Project1/dev.
- Permalinks:** A list of recent builds:
  - Last build (#4), 7 min 8 sec ago
  - Last stable build (#4), 7 min 8 sec ago
  - Last successful build (#4), 7 min 8 sec ago
  - Last failed build (#3), 21 min ago
  - Last unsuccessful build (#3), 21 min ago
  - Last completed build (#4), 7 min 8 sec ago
- Builds:** A table showing the last four builds, each with a status icon (green for success, red for failure) and a dropdown arrow.

Build	Timestamp	Action
#4	9:15 PM	▼
#3	9:01 PM	▼
#2	8:57 PM	▼
#1	8:55 PM	▼

## 16.Docker Image Pushed to Dockerhub dev Repository:

Repositories / dev / General

anand20003/dev

Last pushed 9 minutes ago • Repository size: 22.1 MB

Add a description *(1)*  
Add a category *(1)*

General Tags Image Management BETA Collaborators Webhooks Settings

Tags

This repository contains 0 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	Image		less than 1 day	9 minutes

See all

Using 1 of 1 private repositories. [Get more](#)

Docker commands

To push a new tag to this repository:  
docker push anand20003/dev:tagname

Public view

**buildcloud**  
Build with Docker Build Cloud  
Accelerate image build times with access to cloud-based builders and shared cache.  
Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.  
Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without

## 17.Merging dev to main:

```
ADMIN@DESKTOP-0E234HV MINGW64 ~/Project1 (main)
● $ git merge dev
Updating f2fce62..b1a4281
Fast-forward
 .dockerignore |  9 ++++++++
 .gitignore    |  1 +
 Dockerfile    |  9 ++++++++
 Jenkinsfile   | 69 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 build.sh      |  5 +++
 deploy.sh     | 10 ++++++++
 6 files changed, 103 insertions(+)
 create mode 100644 .dockerignore
 create mode 100644 .gitignore
 create mode 100644 Dockerfile
 create mode 100644 Jenkinsfile
 create mode 100644 build.sh
 create mode 100644 deploy.sh

ADMIN@DESKTOP-0E234HV MINGW64 ~/Project1 (main)
● $ git push origin main
```

## 18.Pipeline Build in Main branch:

The screenshot shows the Jenkins interface for the 'Project1' pipeline. The main header displays 'Status' and 'main'. Below it, a 'Permalinks' section lists recent builds: 'Last build (#1), 1 min 57 sec ago', 'Last stable build (#1), 1 min 57 sec ago', 'Last successful build (#1), 1 min 57 sec ago', and 'Last completed build (#1), 1 min 57 sec ago'. A 'Builds' section shows a single build entry for '#1' at 9:29 PM. At the bottom right, there are links for 'REST API' and 'Jenkins 2.524'.

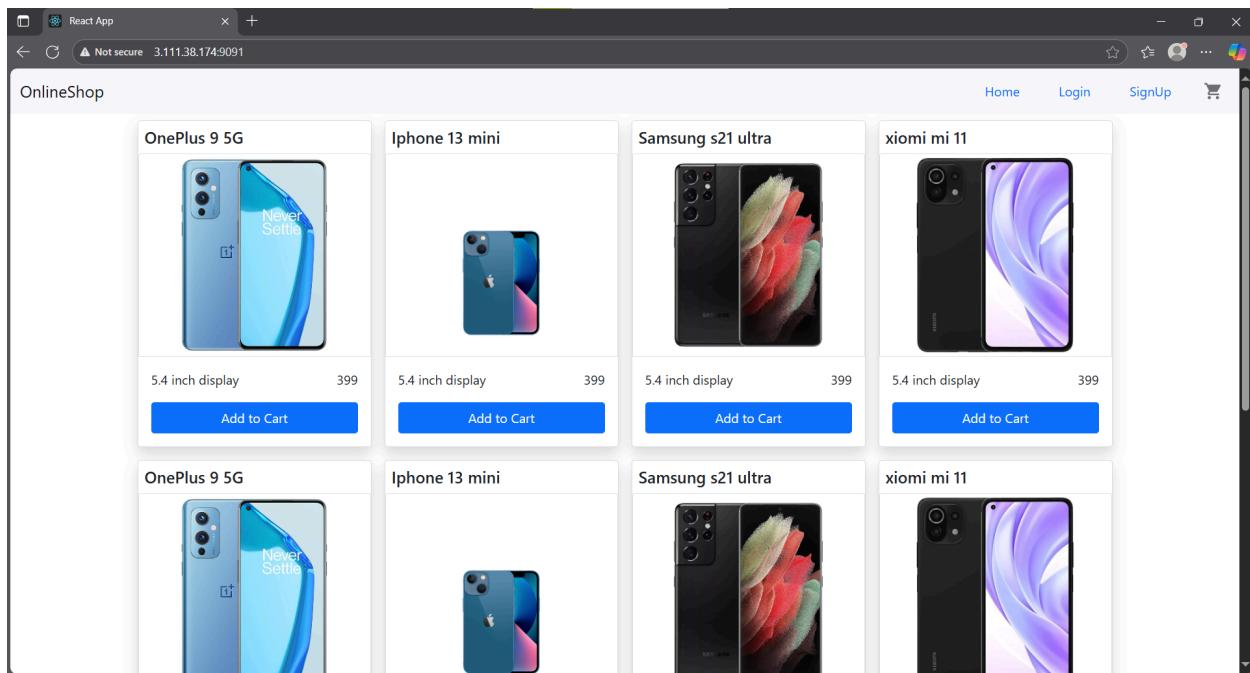
## 19.Docker Image Pushed to DockerHub prod Private Repository:

The screenshot shows the DockerHub interface for the private repository 'anand20003/prod'. The left sidebar shows the user profile 'anand20003' and navigation options like 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area shows the repository details for 'anand20003/prod', last pushed 4 minutes ago, with a repository size of 22.1 MB. It includes sections for 'Add a description', 'Add a category', and tabs for 'General', 'Tags', 'Image Management', 'Collaborators', 'Webhooks', and 'Settings'. The 'Tags' section shows one tag: 'latest' (OS: alpine, Type: Image, Pulled: less than 1 day, Pushed: 4 minutes). On the right, there's a sidebar for 'Docker commands' with the command 'docker push anand20003/prod:tagname'. A 'buildcloud' advertisement is also present.

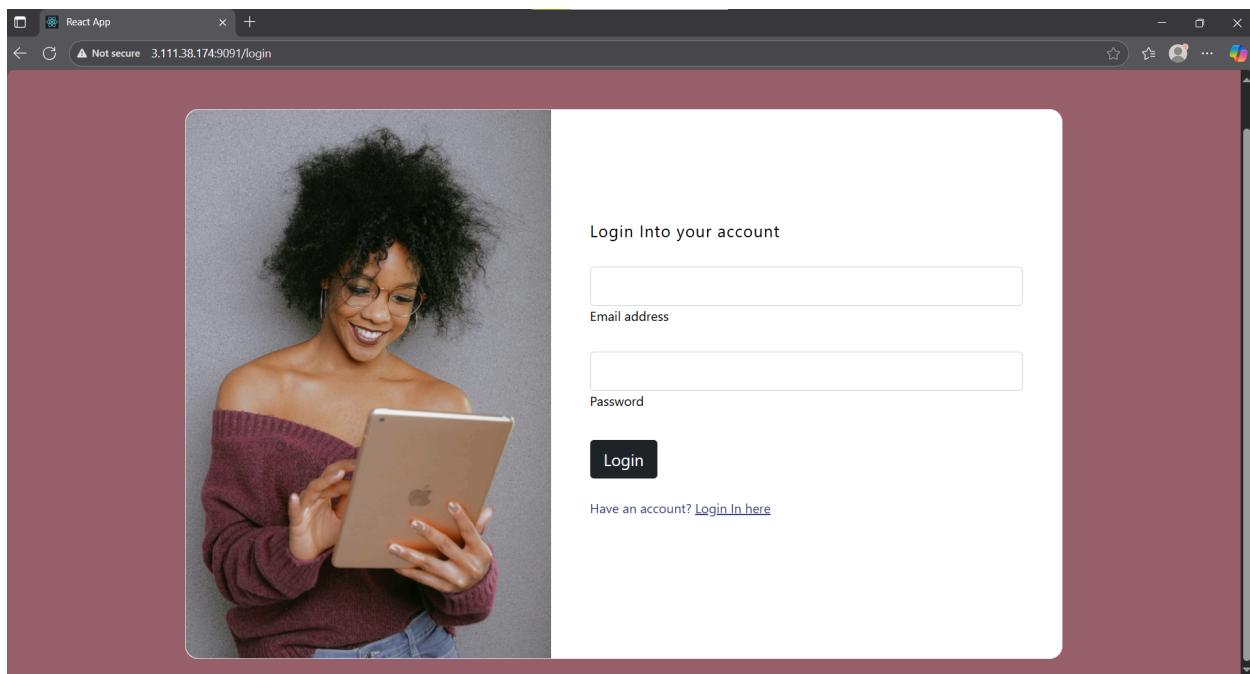
## 20.Docker Image and Container Build Locally:

```
root@ip-172-31-28-7:~# docker images
REPOSITORY          TAG        IMAGE ID      CREATED             SIZE
static-web          latest     8f53e9886939  14 minutes ago   55.1MB
anand20003/dev      latest     8f53e9886939  14 minutes ago   55.1MB
anand20003/prod      latest     8f53e9886939  14 minutes ago   55.1MB
nginx              alpine     4a86014ec699  13 days ago    52.5MB
root@ip-172-31-28-7:~# docker ps
CONTAINER ID        IMAGE       COMMAND                  CREATED             STATUS              PORTS               NAMES
03azdd603e79        static-web   "/docker-entrypoint..."  4 minutes ago     Up 4 minutes      0.0.0.0:9091->80/tcp, [::]:9091->80/tcp   static_application
root@ip-172-31-28-7:~#
```

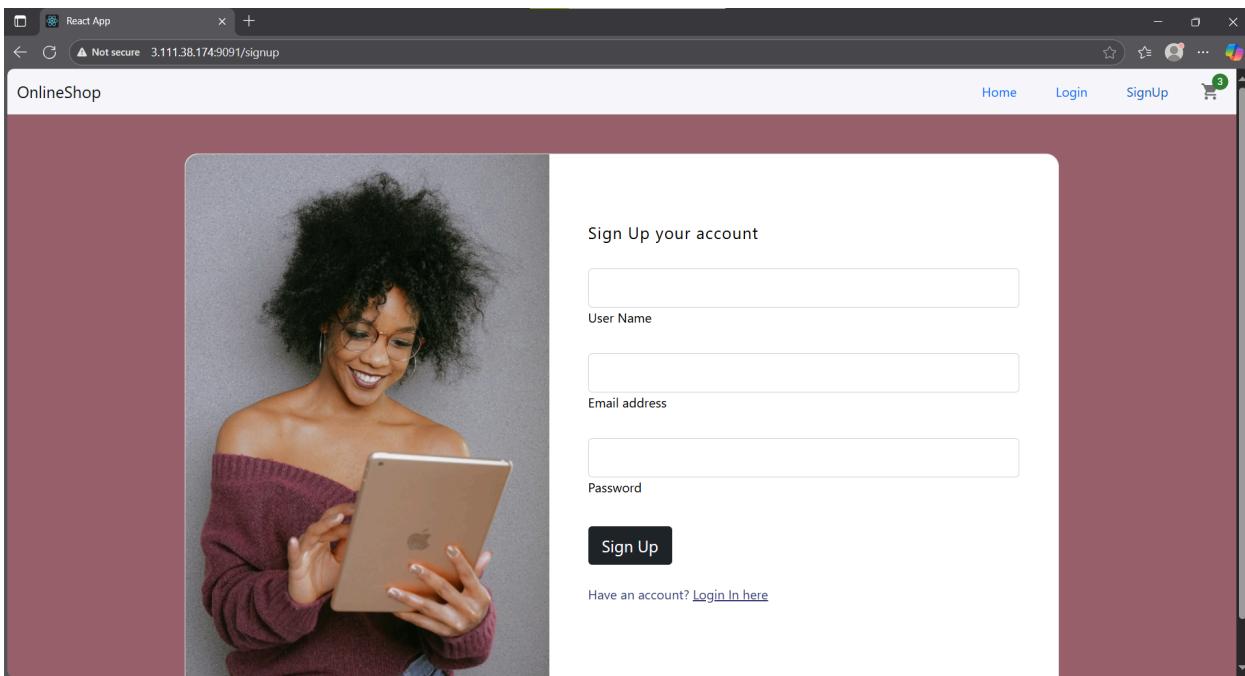
## 21.Output 1:



## 22.Output 2:



## 23.Output 3:



## 24.Output 4:

The screenshot shows a React application interface for a shopping cart. At the top, there's a dark header bar with a 'React App' logo, a 'Not secure' warning, and the URL '3.111.38.174:9091/cart'. Below the header is a maroon header with the text 'OnlineShop'. The main content area is titled 'Shopping Cart' and displays a table of items. The table has columns for '#', 'Product', 'Price', 'Quantity', and 'Total'. There are three items listed: 1 OnePlus 9 5G (5.4 inch display) at \$399, 1 Iphone 13 mini (5.4 inch display) at \$399, and 1 Samsung s21 ultra (5.4 inch display) at \$399. Each item row includes a small image, a description, a price, a quantity selector (with values -, 1, +), and a 'Remove' link. At the bottom of the table, it says 'SubTotal: \$1197'. Below the table, there's a note 'Taxes and Shipping Calculated' and three buttons: a red 'Clear Cart' button, a blue 'Login to CheckOut' button, and a white 'Continue Shopping' button.

#	Product	Price	Quantity	Total
1	OnePlus 9 5G 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399
1	Iphone 13 mini 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399
1	Samsung s21 ultra 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399

SubTotal: \$1197

Taxes and Shipping Calculated

[Login to CheckOut](#)

[Continue Shopping](#)

## 6.AWS:

### 1.Launching EC2 Instance with t2.micro:

The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. The process is at the 'Name and tags' step, where 'Project1' is entered. It then moves to the 'Application and OS Images (Amazon Machine Image)' step, where 'Ubuntu' is selected from a grid of AMI icons. The summary on the right shows one instance being launched with the Amazon Linux 2023 AMI, instance type t2.micro, and 8 GiB storage. The final step is 'Launch instance'.

### 2. Security Group Configurations:

The screenshot shows the 'Edit inbound rules' page for a security group. It lists four rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0232081b632cb7f85	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0d76c7fb6afdd8e1d	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-00974c005bd504d8a	SSH	TCP	22	My IP	Login Access
-	Custom TCP	TCP	3000	Anyw...	27.61.51.70/32

A warning message at the bottom notes that rules with source 0.0.0.0/0 or ::/0 allow all IP addresses to access the instance, and recommends setting security group rules to allow access from known IP addresses only. Buttons for 'Add rule', 'Preview changes', and 'Save rules' are visible.

### 3.Login into the server with my IP:

The screenshot shows a terminal window titled "ubuntu@ip-172-31-26-202: ~". The session starts with a password prompt and then displays the standard Ubuntu 24.04.3 LTS welcome message. It then provides system information as of Tuesday, August 26, 2025, including load average, memory usage, and swap usage. It also checks for security updates and ESM Apps. The terminal ends with a copyright notice and a warning about no warranty, followed by instructions for sudo usage.

```
ubuntu@ip-172-31-26-202: ~
login as: ubuntu
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Tue Aug 26 22:07:09 UTC 2025

 System load: 0.16          Processes:           110
 Usage of /: 25.8% of 6.71GB   Users logged in:    0
 Memory usage: 21%           IPv4 address for enX0: 172.31.26.202
 Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-26-202:~$
```

### 4.Installing Docker:

The screenshot shows a terminal window titled "root@ip-172-31-26-202: ~". The root user runs the command "apt install docker.io -y" to install Docker. The terminal is black, indicating the command is still executing or has just completed.

```
root@ip-172-31-26-202:~# apt install docker.io -y
```

## 5.Docker Installed:

```
root@ip-172-31-26-202:~# docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
root@ip-172-31-26-202:~#
```

## 6.DockerHub Login Done:

```
root@ip-172-31-26-202:~# docker login
USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username>'

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

Waiting for authentication in the browser...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
root@ip-172-31-26-202:~#
```

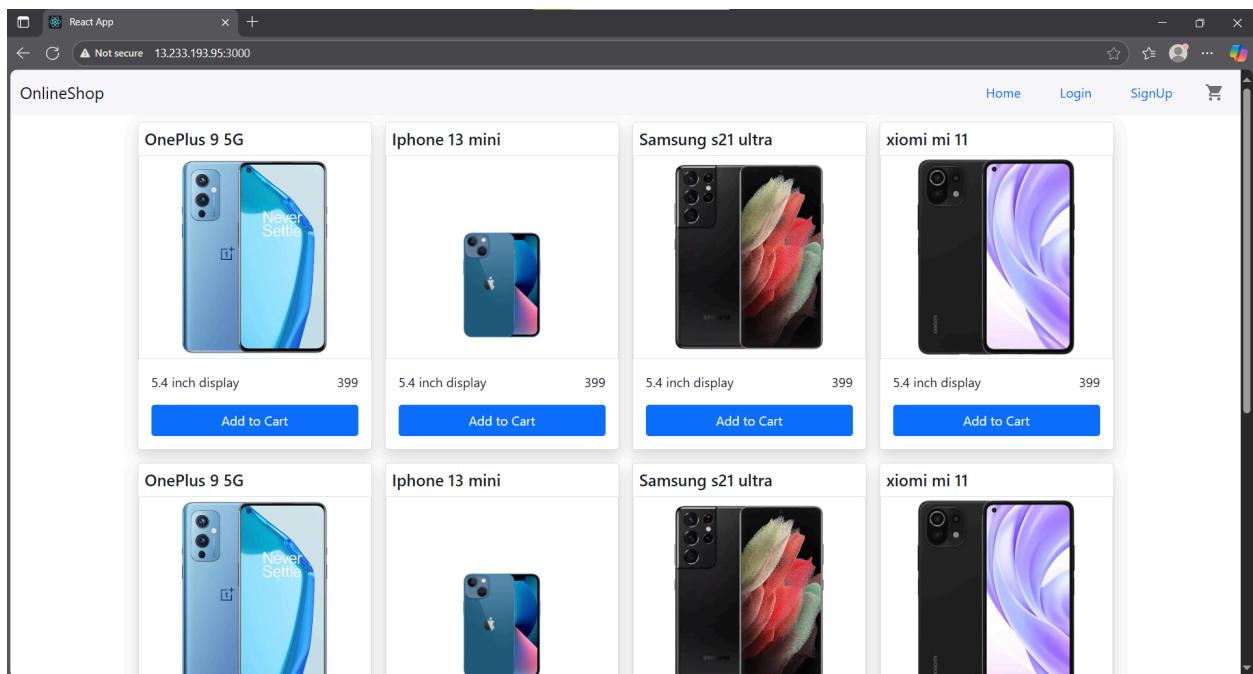
## 7.Pulling Docker Image From Dockerhub:

```
root@ip-172-31-26-202:~# docker pull anand20003/prod:latest
latest: Pulling from anand20003/prod
9824c27679d3: Pull complete
6bc572a340ec: Pull complete
403e3f251637: Pull complete
9adfbae99cb7: Pull complete
7a8a46741e18: Pull complete
c9ebbe2ff2d2c: Pull complete
a992fbcc61ecc: Pull complete
cb1ff4086f82: Pull complete
7252e8e57291: Pull complete
8e6b521f19db: Pull complete
Digest: sha256:f11b94cb4989c2d6062c45385ab49c49163fdadfe2752a3ad7ac1df6995f7c64
Status: Downloaded newer image for anand20003/prod:latest
docker.io/anand20003/prod:latest
root@ip-172-31-26-202:~#
```

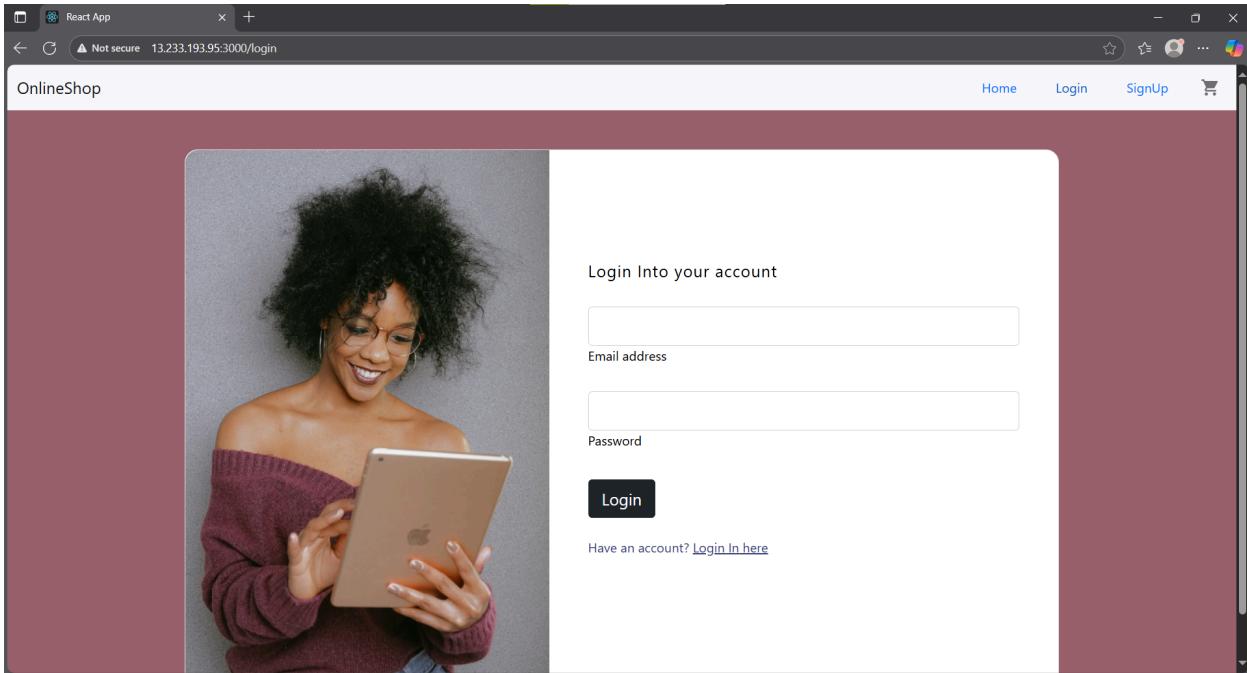
## 8.Container Created With Pulled Image:

```
root@ip-172-31-26-202:~# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
anand20003/prod  latest   8f5369886939  About an hour ago  55.1MB
root@ip-172-31-26-202:~# docker run -itd -p "3000:80" --name Reactjs_Application anand20003/prod
44455847f161ad69616edcb1ff9dd1cb03783418d542415e56314fc89d027d32
root@ip-172-31-26-202:~#
```

## 9.Output 1:



## 10.Output 2:



## 11.Output 3:

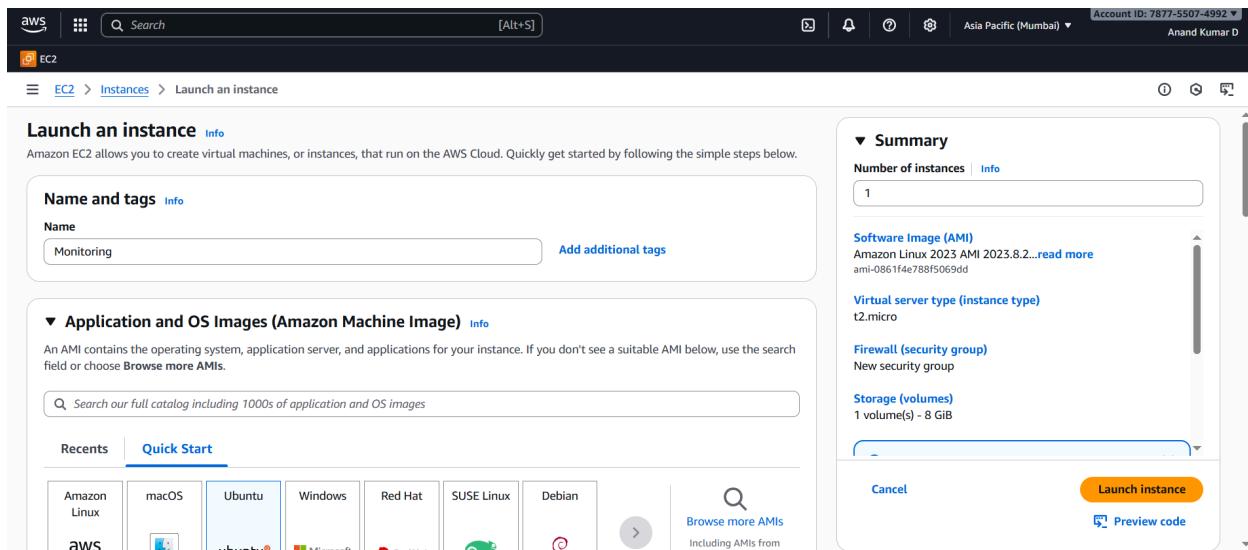
A screenshot of a web browser showing the shopping cart page of an online shop. The URL is 13.233.193.95:3000/cart. The page has a dark red header with the text "OnlineShop". The main content area is titled "Shopping Cart" and displays a table of items in the cart. The table has columns for "#", "Product", "Price", "Quantity", and "Total". There are four items listed:

#	Product	Price	Quantity	Total
1	Samsung s21 ultra 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399
1	Iphone 13 mini 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399
1	OnePlus 9 5G 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399
1	Samsung s21 ultra 5.4 inch display <a href="#">Remove</a>	\$399	- 1 +	\$399

At the bottom left is a "Clear Cart" button. At the bottom right are buttons for "SubTotal: \$1596", "Taxes and Shipping Calculated", and "Login to CheckOut".

## 7.Monitoring:

### 1.Launching Instance For Monitoring:



### 2.Installing Docker-compose:

```
root@ip-172-31-23-1:~# apt install docker-compose
```

### 3.Docker-compose.yaml file for Uptime Kuma:

```
GNU nano 7.2                                            docker-compose.yml
version: '3'
services:
  uptime-kuma:
    image: louislam/uptime-kuma:1
    container_name: uptime-kuma
    restart: always
    ports:
      - "3001:3001"
    volumes:
      - ./data:/app/data
```

#### 4. Running Docker-compose File:

```
root@ip-172-31-23-1:~# docker-compose up -d
Creating network "root_default" with the default driver
Pulling uptime-kuma (louislam/uptime-kuma:1)...
1: Pulling from louislam/uptime-kuma
b33856f40a7: Pull complete
874bf4d93720: Pull complete
b16337721583: Pull complete
7d955db85b85: Pull complete
2c706596bd17: Pull complete
88a5c59ed14f: Pull complete
5a1d0a896c33: Pull complete
e68c2f25b946: Pull complete
2e6c90f010d6: Pull complete
ff15b10fabbb: Pull complete
4f4fb700ef54: Pull complete
d2a400cc8adb: Pull complete
Digest: sha256:431fee3be822b04861cf0e35daf4beef6b7cb37391c5f26c3ad6e12ce280fe18
Status: Downloaded newer image for louislam/uptime-kuma:1
Creating uptime-kuma ... done
root@ip-172-31-23-1:~#
```

#### 5. Enabling Port 3001 to access Uptime-Kuma for Monitoring:

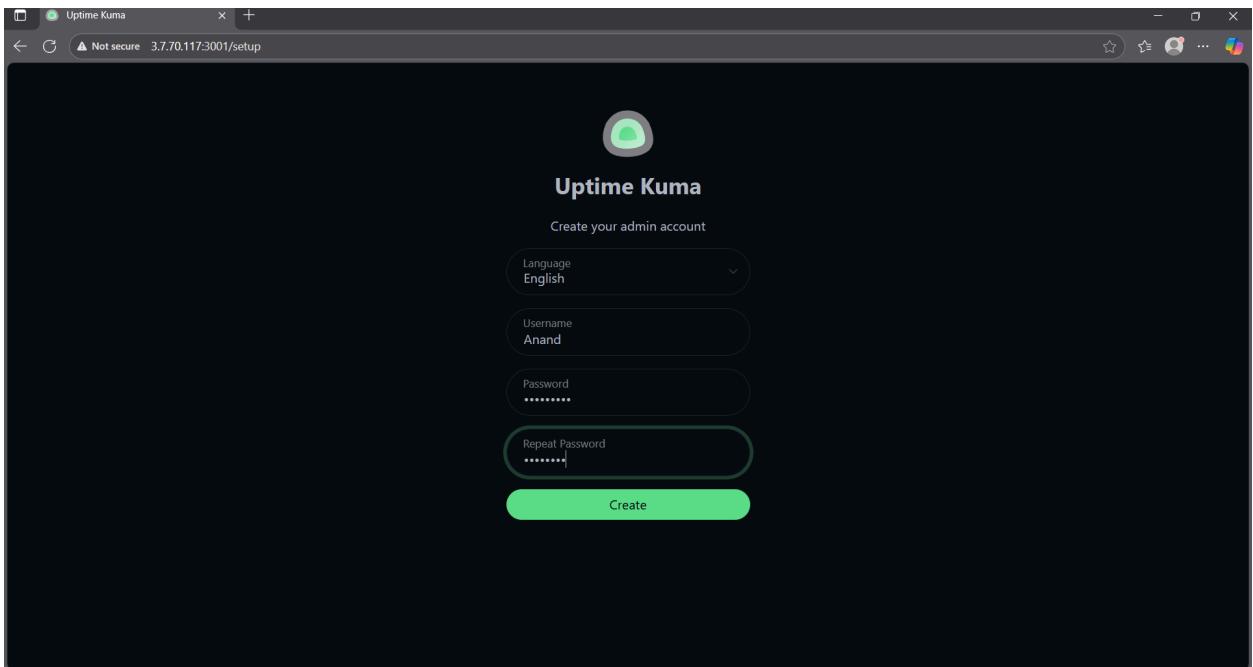
The screenshot shows the AWS Management Console interface for managing security group inbound rules. The URL in the address bar is `EC2 > Security Groups > sg-081a02413cadd936a - launch-wizard-11 > Edit inbound rules`. The page title is "Edit inbound rules".

**Inbound rules**

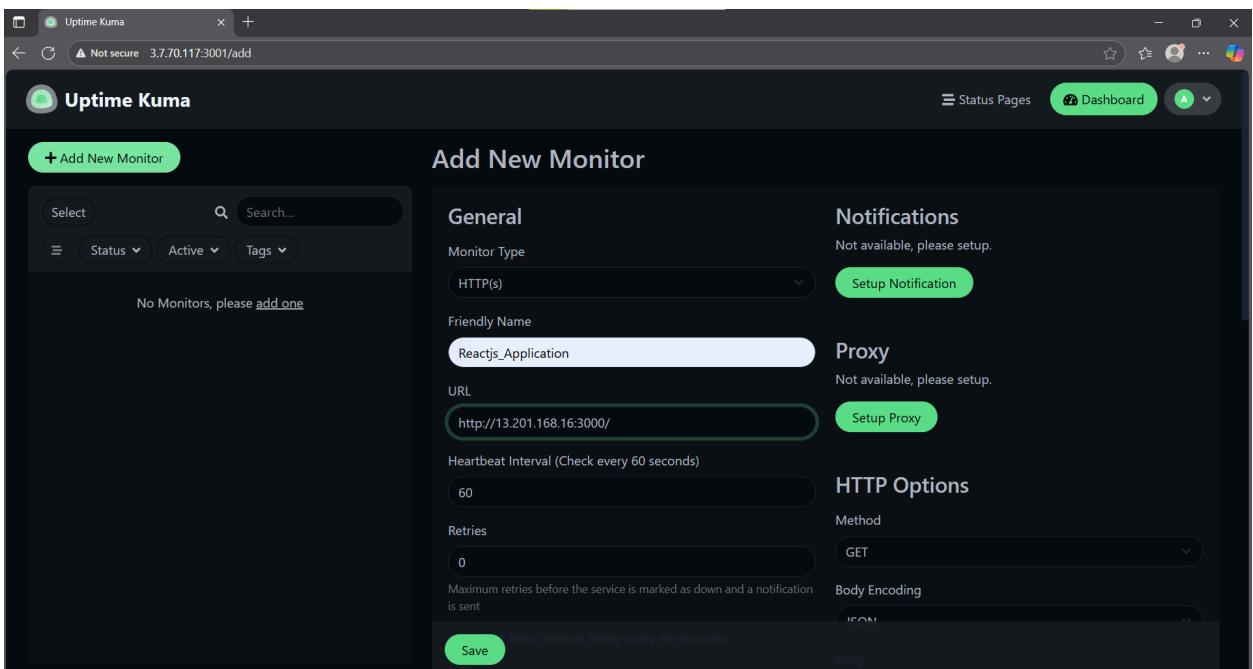
Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Action
sgr-0fc6d94bce5a0203a	HTTP	TCP	80	Custom	0.0.0.0/0	Delete
sgr-058e1012a600291df	SSH	TCP	22	Custom	0.0.0.0/0	Delete
sgr-0830b1dc228de0203	HTTPS	TCP	443	Custom	0.0.0.0/0	Delete
sgr-0fb0af36272c87dfb	Custom TCP	TCP	3001	Custom	0.0.0.0/0	Add rule

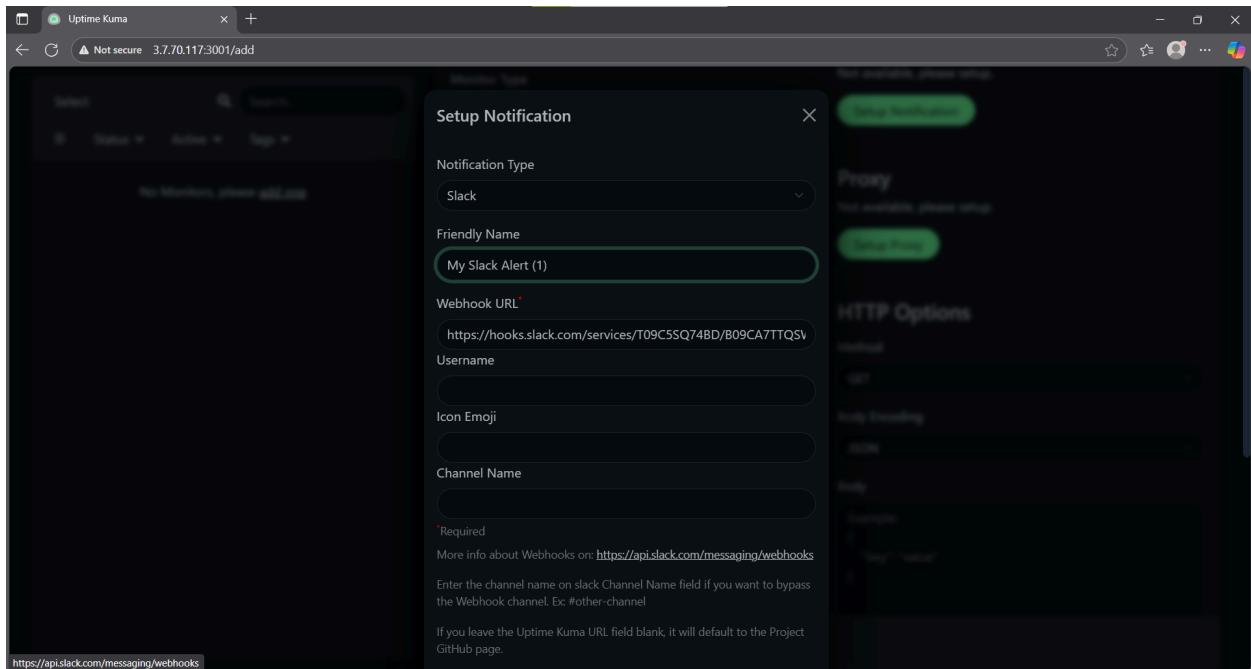
## 6.Creating Account in Uptime-Kuma for Monitoring:



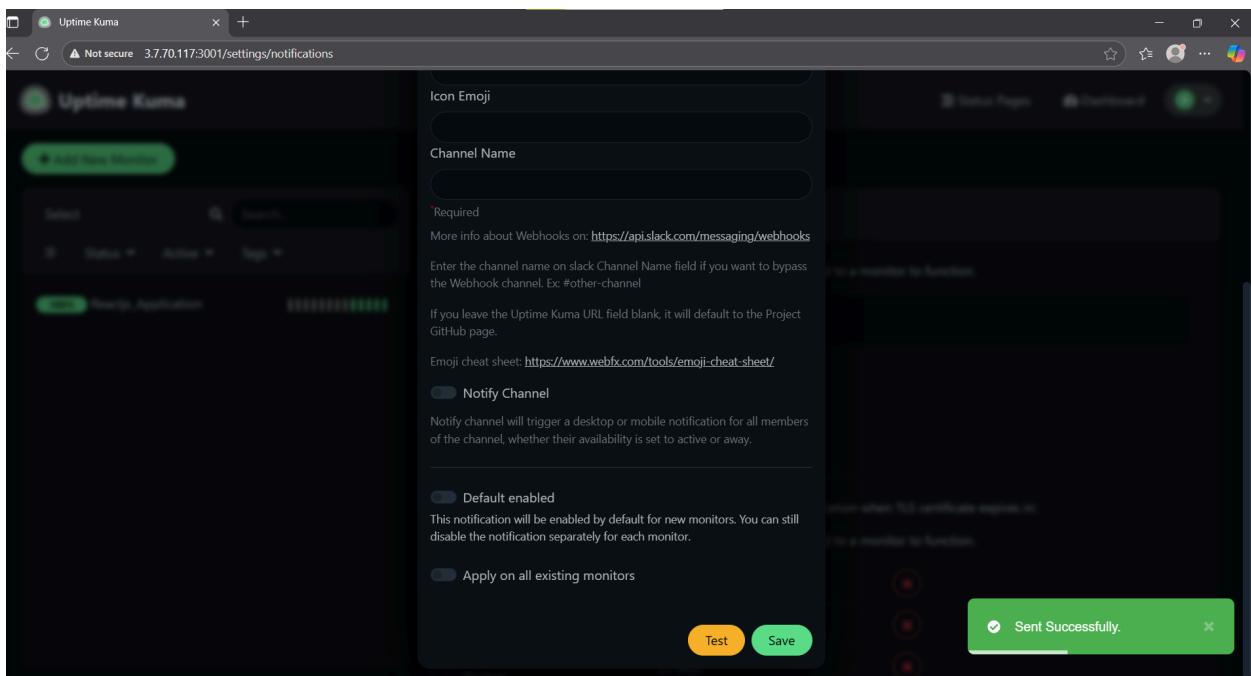
## 7.Adding New Monitor:



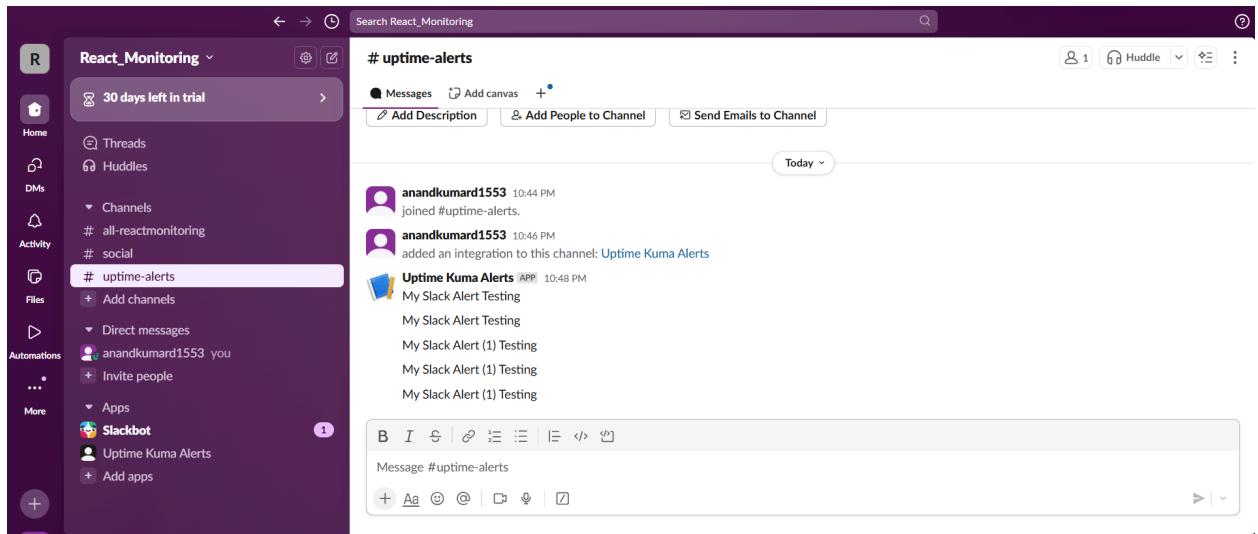
## 8. Setting up Slack Notification:



## 9. Testing the Notification:



## 10.Notification Received in Slack:



## 11.Monitor Dashboard in Uptime Kuma:

