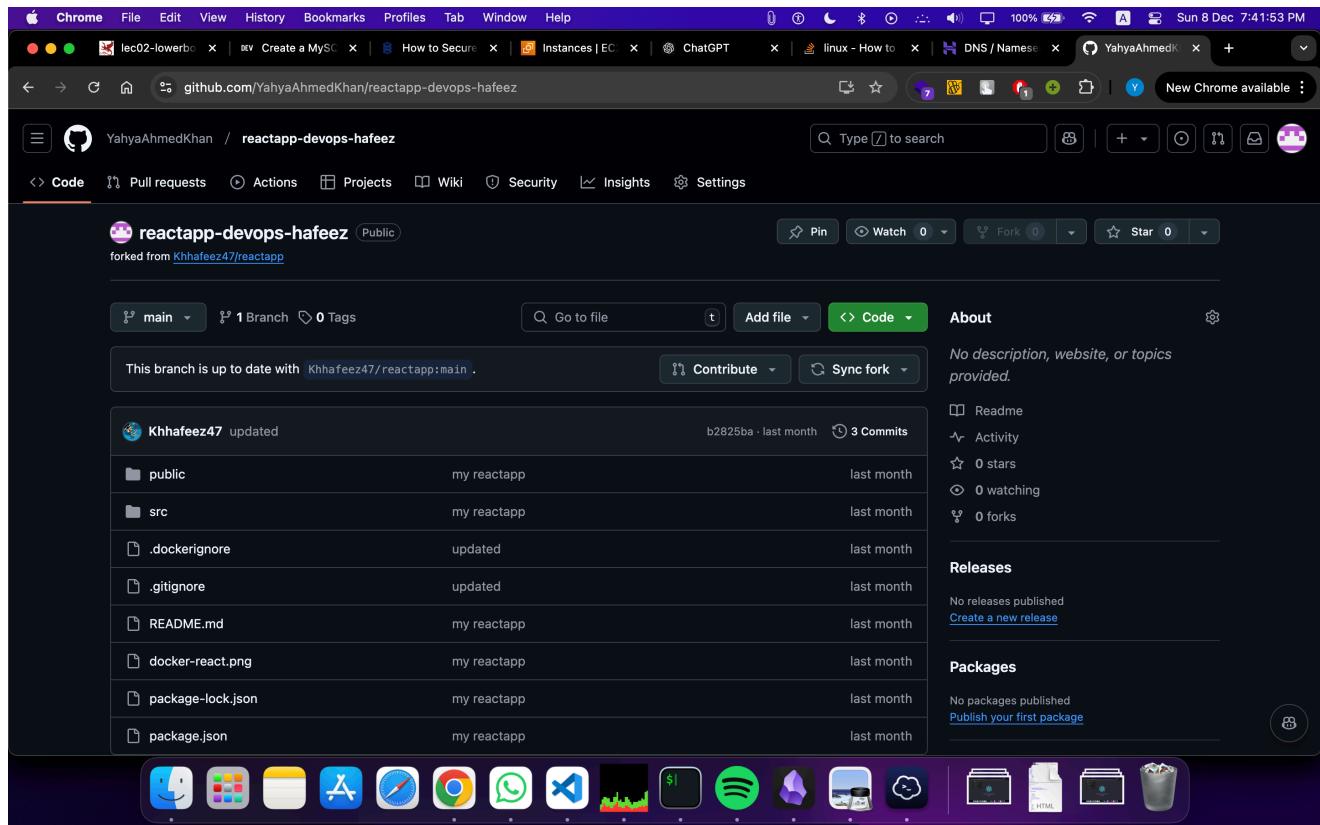
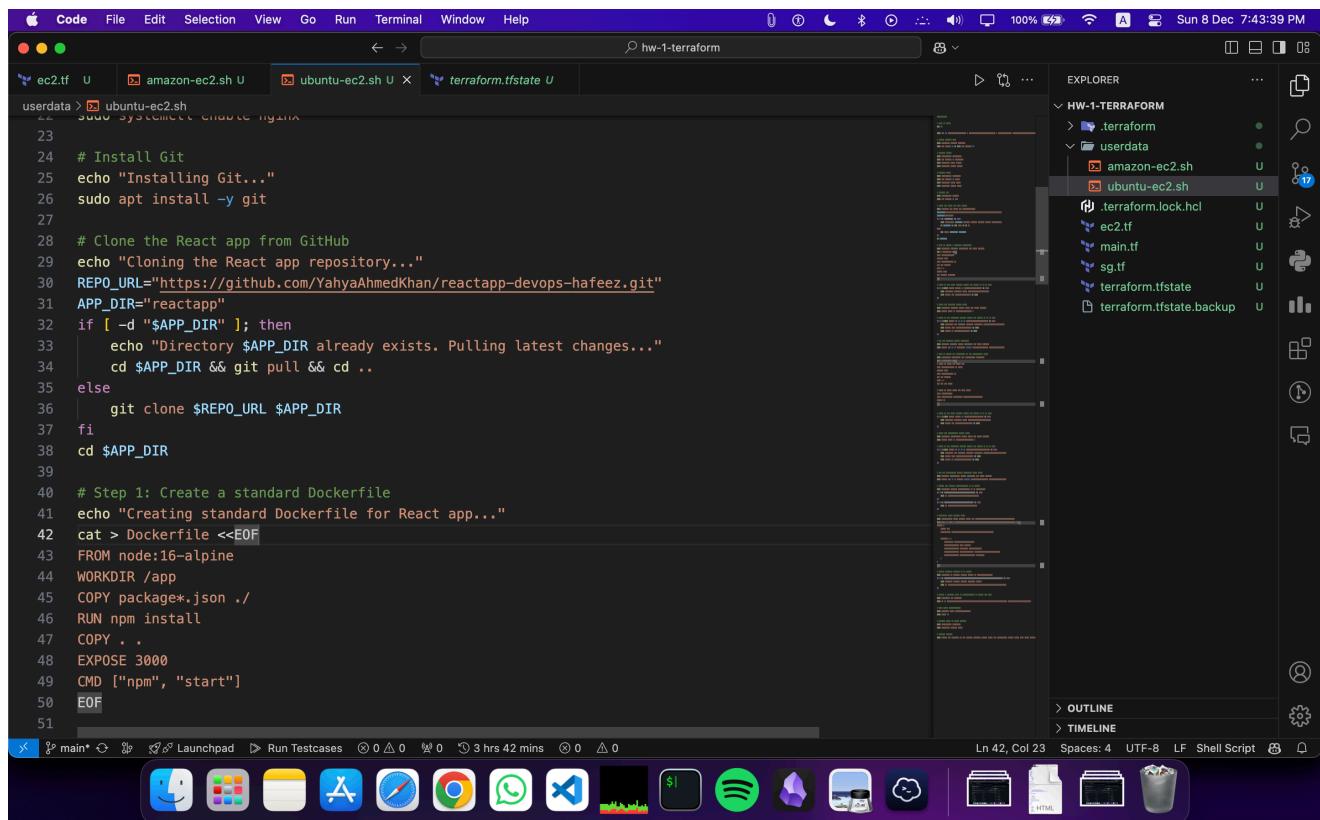


# Terraform HW-1 Yahya Ahmed Khan 24442

## Forking the Repository



## Cloning the Repository



# Creating a Dockerfile to Run React App

A screenshot of the Visual Studio Code interface on a Mac. The top menu bar shows 'Code', 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Window', and 'Help'. The status bar at the bottom right shows 'Sun 8 Dec 7:43:39 PM' and battery level '100%'. The main workspace has several tabs open: 'ec2.tf' (marked as 'U'), 'amazon-ec2.sh' (marked as 'U'), 'ubuntu-ec2.sh' (marked as 'U'), and 'terraform.tfstate' (marked as 'U'). The code editor contains a Dockerfile with the following content:

```
userdata > ubuntu-ec2.sh
23
24 # Install Git
25 echo "Installing Git..."
26 sudo apt install -y git
27
28 # Clone the React app from GitHub
29 echo "Cloning the React app repository..."
30 REPO_URL="https://github.com/YahyaAhmedKhan/reactapp-devops-hafeez.git"
31 APP_DIR="reactapp"
32 if [ -d "$APP_DIR" ]; then
33     echo "Directory $APP_DIR already exists. Pulling latest changes..."
34     cd $APP_DIR && git pull && cd ..
35 else
36     git clone $REPO_URL $APP_DIR
37 fi
38 cd $APP_DIR
39
40 # Step 1: Create a standard Dockerfile
41 echo "Creating standard Dockerfile for React app..."
42 cat > Dockerfile <<EOF
43 FROM node:16-alpine
44 WORKDIR /app
45 COPY package*.json .
46 RUN npm install
47 COPY . .
48 EXPOSE 3000
49 CMD ["npm", "start"]
50 EOF
51
```

The Explorer sidebar on the right shows a tree view of the project structure under 'HW-1-TERRAFORM': '.terraform', 'userdata', 'amazon-ec2.sh', 'ubuntu-ec2.sh', '.terraform.lock.hcl', 'ec2.tf', 'main.tf', 'sg.tf', 'terraform.tfstate', and 'terraform.tfstate.backup'. The bottom status bar shows 'Ln 42, Col 23' and other settings.

## Modifying Dockerfile for Multistage Build

A screenshot of the Visual Studio Code interface on a Mac, similar to the previous one but with a different Dockerfile content. The code editor now contains a Dockerfile with the following content:

```
userdata > ubuntu-ec2.sh
71 sudo docker run -d -p 3000:3000 --name reactapp-standard reactapp-standard
72
73 # Step 2: Modify the Dockerfile to use multi-stage build
74 echo "Modifying Dockerfile for multi-stage build..."
75 cat > Dockerfile <<EOF
76 # Stage 1: Build the React app
77 FROM node:16-alpine AS build
78 WORKDIR /app
79 COPY package*.json .
80 RUN npm install
81 COPY . .
82 RUN npm run build
83
84 # Stage 2: Serve React app with Nginx
85 FROM nginx:alpine
86 COPY --from=build /app/build /usr/share/nginx/html
87 EXPOSE 80
88 EOF
89
90
91 # Check if the image already exists and remove it if it does
92 if [ "$(sudo docker images -q reactapp-multistage)" ]; then
93     echo "Removing existing image 'reactapp-multistage'..."
94     sudo docker rmi reactapp-multistage || true
95 fi
96
97 # Build the multi-stage Docker image
98 echo "Building multi-stage Docker image for React app...."
99 sudo docker build -t reactapp-multistage .
```

The Explorer sidebar on the right shows the same project structure as the previous screenshot. The bottom status bar shows 'Ln 62, Col 65' and other settings.

## Running Multistage Docker Build in Nginx Container

```

82 # Stage 2: Serve React app with Nginx
83 FROM nginx:alpine
84 COPY --from=build /app/build /usr/share/nginx/html
85 EXPOSE 80
86 EOF
87
88 # Check if the image already exists and remove it if it does
89 if [ "$(sudo docker images -q reactapp-multistage)" ]; then
90     echo "Removing existing image 'reactapp-multistage'..."
91     sudo docker rmi reactapp-multistage || true
92 fi
93
94 # Build the multi-stage Docker image
95 echo "Building multi-stage Docker image for React app..."
96 sudo docker build -t reactapp-multistage .
97
98 # Check if the container already exists and remove it if it does
99 if [ "$(sudo docker ps -a -q -f name=reactapp-multistage)" ]; then
100    echo "Stopping and removing existing container 'reactapp-multistage'..."
101    sudo docker stop reactapp-multistage || true
102    sudo docker rm reactapp-multistage || true
103 fi
104
105 # Run the multi-stage Docker container using Nginx
106 echo "Running multi-stage Docker container for React app..."
107 sudo docker run -d -p 8080:80 --name reactapp-multistage reactapp-multistage
108
109 # Configure Nginx for reverse proxy
110 echo "Configuring Nginx reverse proxy for reactapp.yahyaabdullahsaadsubata.online..."
```

Ln 122, Col 6 Spaces: 4 UTF-8 LF Shell Script

## Applying DNS for React App

| Type  | Name                | Priority | Content                        | TTL | Delete                 | Edit                 |
|-------|---------------------|----------|--------------------------------|-----|------------------------|----------------------|
| CNAME | www                 | 0        | yahyaabdullahsaadsubata.online | 300 | <a href="#">Delete</a> | <a href="#">Edit</a> |
| A     | yahya-hwl-ubnt      | 0        | 52.201.227.30                  | 60  | <a href="#">Delete</a> | <a href="#">Edit</a> |
| A     | abdullah-jenkins-mt | 0        | 172.17.5.43                    | 60  | <a href="#">Delete</a> | <a href="#">Edit</a> |
| A     | saad-jenkins-mt     | 0        | 172.17.5.43                    | 60  | <a href="#">Delete</a> | <a href="#">Edit</a> |
| A     | yahya-jenkins-mt    | 0        | 172.17.5.43                    | 60  | <a href="#">Delete</a> | <a href="#">Edit</a> |
| A     | subata-jenkins-mt   | 0        | 172.17.5.43                    | 60  | <a href="#">Delete</a> | <a href="#">Edit</a> |

Sun 8 Dec 8:16:35 PM

hpanel.hostinger.com/domain/yahyaabdullahsaadsabata.online/dns

HOSTINGER

Main menu

Domain Overview

DNS / Nameservers

Domain Ownership

Give feedback

A yahya2-jenkins 0 172.17.5.43 60 Delete Edit

A saad-nodeapp 0 172.17.5.43 60 Delete Edit

A yahya2-nodeapp 0 172.17.5.43 60 Delete Edit

A yahya-hwl-amzn 0 44.202.92.212 60 Delete Edit

A abdullah-react1 0 172.17.5.43 60 Delete Edit

A saad-react1 0 172.17.5.43 60 Delete Edit

A yahya-react1 0 172.17.5.43 60 Delete Edit

A subata-react1 0 172.17.5.43 60 Delete Edit

Refer & Earn 345 USD

13 18

Give feedback

| Type | Name            | TTL | IP Address    | Actions     |
|------|-----------------|-----|---------------|-------------|
| A    | yahya2-jenkins  | 0   | 172.17.5.43   | Delete Edit |
| A    | saad-nodeapp    | 0   | 172.17.5.43   | Delete Edit |
| A    | yahya2-nodeapp  | 0   | 172.17.5.43   | Delete Edit |
| A    | yahya-hwl-amzn  | 0   | 44.202.92.212 | Delete Edit |
| A    | abdullah-react1 | 0   | 172.17.5.43   | Delete Edit |
| A    | saad-react1     | 0   | 172.17.5.43   | Delete Edit |
| A    | yahya-react1    | 0   | 172.17.5.43   | Delete Edit |
| A    | subata-react1   | 0   | 172.17.5.43   | Delete Edit |

## Amazon site

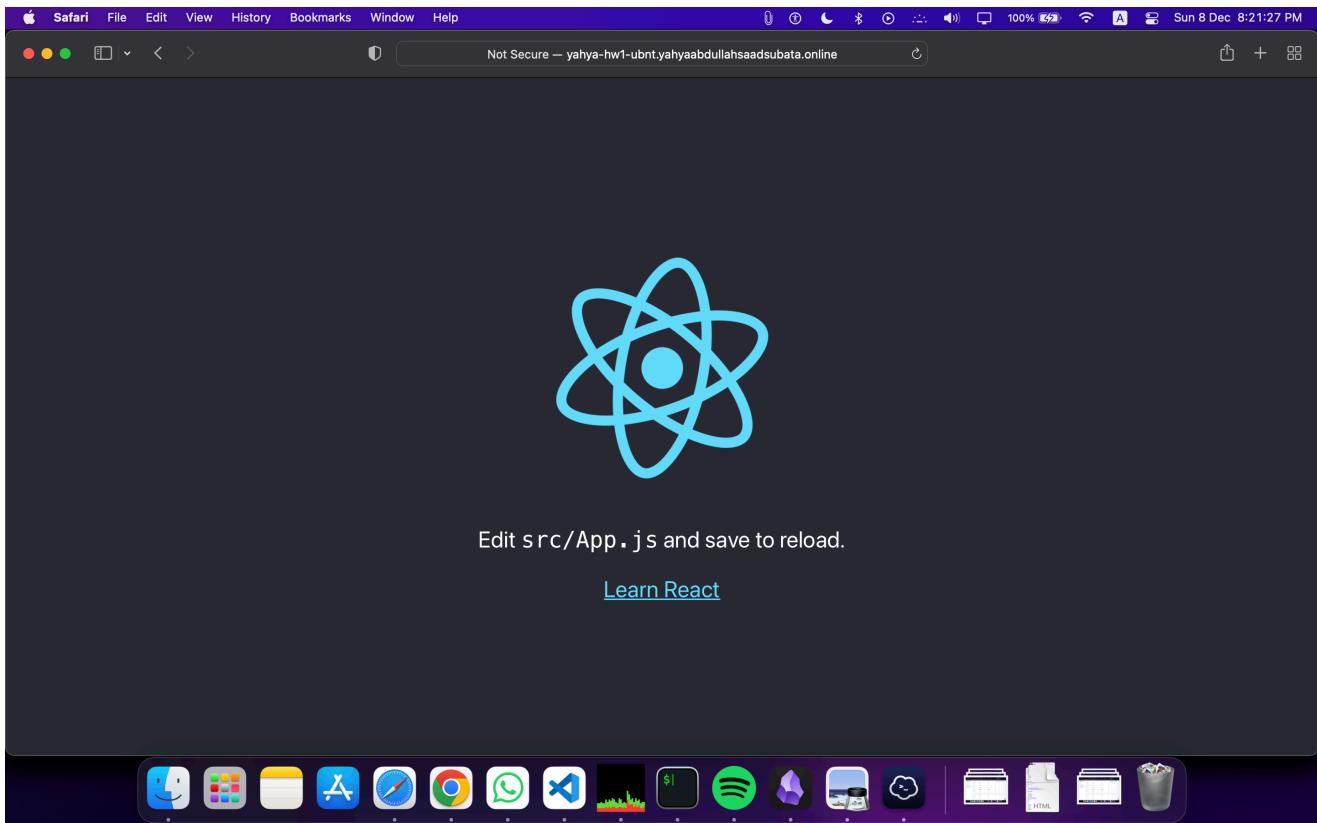
Sun 8 Dec 8:16:53 PM

Not Secure yahya-hwl-amzn.yahyaabdullahsaadsabata.online

Edit `src/App.js` and save to reload.

[Learn React](#)

## Ubuntu site



## Creating Security Group Rule for Port 3000

```
resource "aws_security_group" "my-sg" {
  //vpc_id = "vpc-0d463f4edad65e5"
  vpc_id = data.aws_vpc.default.id
  name   = "my-sg"

  ingress {
    from_port  = 3000
    to_port    = 3000
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port  = 22
    to_port    = 22
  }
}
```

```
Destroy complete! Resources: 3 destroyed.

~/De/d/U/devops/DevOps-Fall-24/hw1-terraform main ?1 > terraform init && terraform plan && terraform apply
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.78.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
data.aws_vpc.default: Reading...
data.aws_vpc.default: Read complete after 2s [id=vpc-030c196b1d96ded76]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
```

## Setting Up EC2 Instances via Terraform

ec2.tf

```
You, 2 minutes ago | 1 author (You)
1 resource "aws_instance" "amazon_linux-ec2-hw1" {
2   ami                  = "ami-045sec754f44f9a4a" # Amazon Linux 2023
3   instance_type        = "t2.micro"
4   subnet_id            = "subnet-070061a2764927c94" # Subnet ID for EC2 instance
5   key_name              = "yahya-ec2-key"
6   vpc_security_group_ids = [aws_security_group.my-sg.id] # Assign EC2 security group
7   associate_public_ip_address = true
8   iam_instance_profile      = "tf_ec2s3" // role made in IAM, allowed services: ec2
9
10  tags = {
11    Name = "amazon_linux-ec2-hw1"
12  }
13
14 user_data = base64encode(file("${path.module}/userdata/amazon-ec2.sh")) # Importing user data from file
15
16 root_block_device {
17   volume_size = 10
18   volume_type = "gp3"
19 }
20
21 You, 2 minutes ago * done, scripts randomly don't work for tf
22
23 resource "aws_instance" "ubuntu-ec2-hw1" {
24   ami                  = "ami-005fc0f236362e99f" # Ubuntu Server 22.04 LTS
25   instance_type        = "t2.micro"
26   subnet_id            = "subnet-070061a2764927c94" # Subnet ID for EC2 instance
27   key_name              = "yahya-ec2-key"
28   vpc_security_group_ids = [aws_security_group.my-sg.id] # Assign EC2 security group
29   associate_public_ip_address = true
30   iam_instance_profile      = "tf_ec2s3" // role made in IAM, allowed services: ec2
31
32   tags = {
33     Name = "ubuntu-ec2-hw1"
34   }
35
36 user_data = base64encode(file("${path.module}/userdata/ubuntu-ec2.sh")) # Importing user data from file
```

SOURCE CONTROL  
Message (Enter to com...)  
✓ Commit  
Changes  
ec2.tf  
M

SOURCE CONTROL... Auto  
done, scripts randomly ... @main  
terraform works, n... @origin/main  
added tf files, not correct abi...  
added resource files shared in w...  
Update README.md Yahya Ahme...  
Initial commit Yahya Ahmed Khan > GITLENS

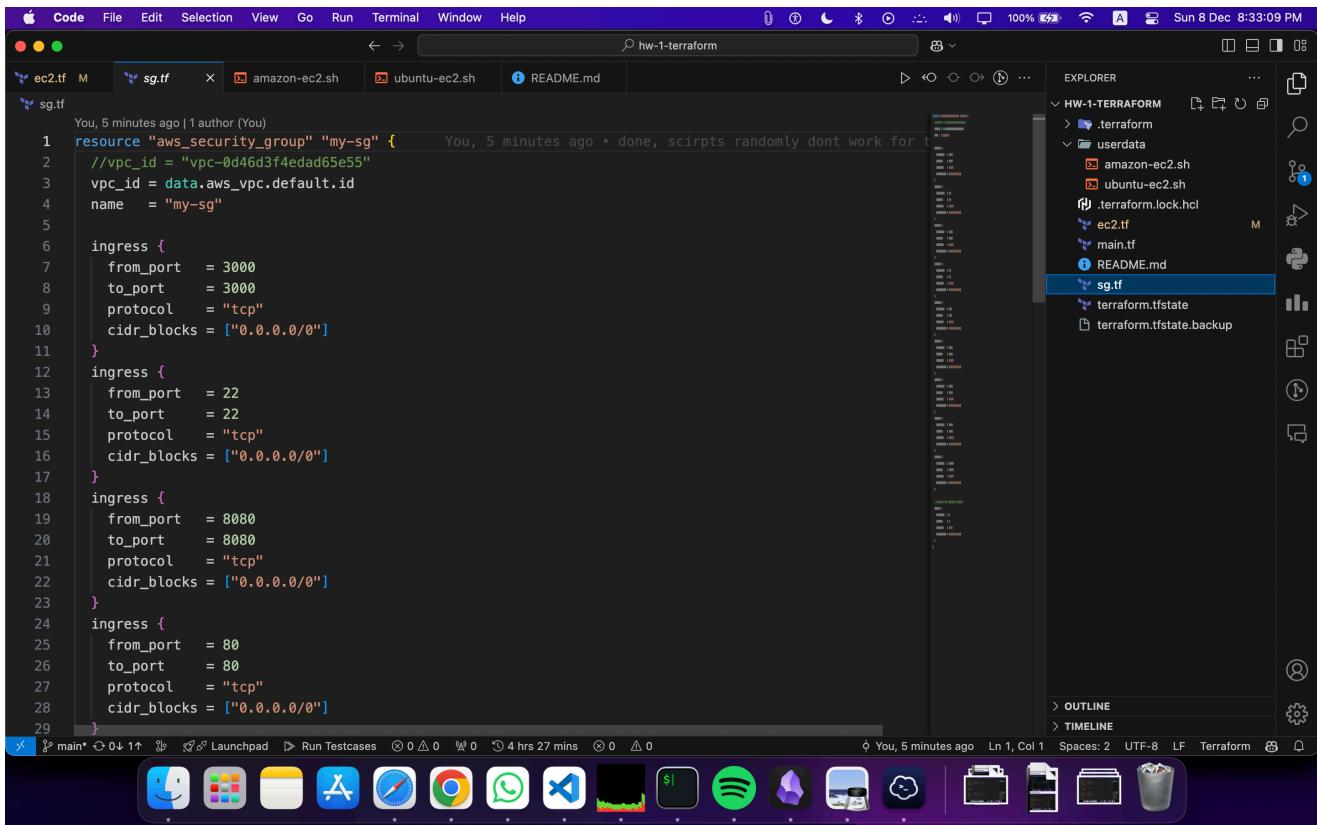
## main.tf

```
You, 4 minutes ago | 1 author (You)
1 # main.tf      You, 4 minutes ago * done, scripts randomly don't work for tf
2
3 # Defining the provider and region
4 provider "aws" {
5   region = "us-east-1"
6 }
7
8 #Default VPC
9
10
11 data "aws_vpc" "default" {
12   default = true
13 }
```

EXPLORER  
HW-1-TERRAFORM .terraform userdata .terraform.lock.hcl ec2.tf main.tf README.md sg.tf terraform.tfstate terraform.tfstate.backup

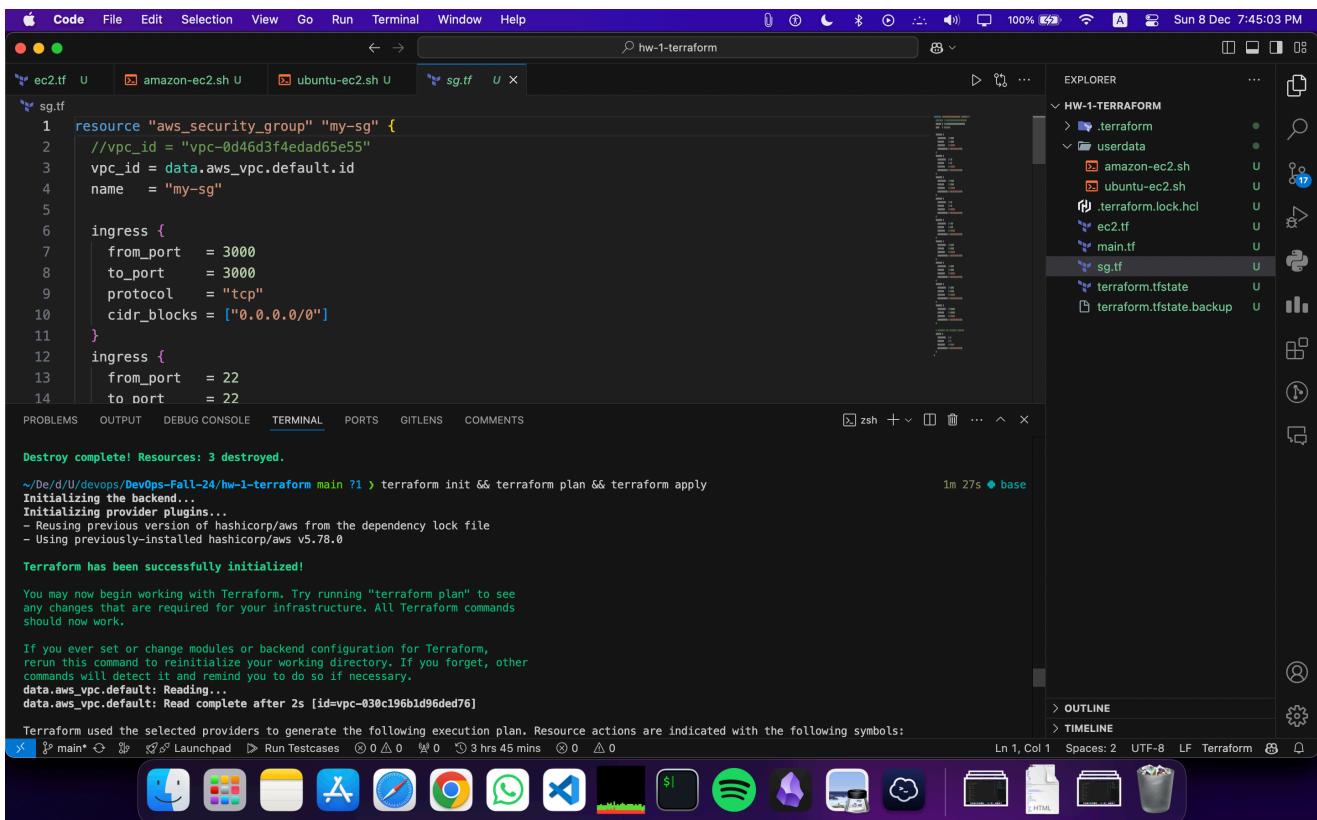
OUTLINE  
TIMELINE

## sg.tf



```
You, 5 minutes ago | 1 author (You)
1 resource "aws_security_group" "my-sg" { You, 5 minutes ago * done, scripts randomly dont work for me
2   //vpc_id = "vpc-0d46d3f4edad65e55"
3   vpc_id = data.aws_vpc.default.id
4   name   = "my-sg"
5
6   ingress {
7     from_port  = 3000
8     to_port    = 3000
9     protocol   = "tcp"
10    cidr_blocks = ["0.0.0.0/0"]
11  }
12  ingress {
13    from_port  = 22
14    to_port    = 22
15    protocol   = "tcp"
16    cidr_blocks = ["0.0.0.0/0"]
17  }
18  ingress {
19    from_port  = 8080
20    to_port    = 8080
21    protocol   = "tcp"
22    cidr_blocks = ["0.0.0.0/0"]
23  }
24  ingress {
25    from_port  = 80
26    to_port    = 80
27    protocol   = "tcp"
28    cidr_blocks = ["0.0.0.0/0"]
29 }
```

## Running terraform



```
Destroy complete! Resources: 3 destroyed.

~/De/d/U/devops/DevOps-Fall-24/hw-1-terraform main ?1 > terraform init && terraform plan && terraform apply
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.78.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
data.aws_vpc.default: Reading...
data.aws_vpc.default: Read complete after 2s [id=vpc-030c196b1d96ded76]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
```

```

resource "aws_security_group" "my-sg" {
  //vpc_id = "vpc-0d46d3f4edad65e55"
  vpc_id = data.aws_vpc.default.id
  name   = "my-sg"

  ingress {
    from_port  = 3000
    to_port    = 3000
    protocol   = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port  = 22
    to_port    = 22
  }
}

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_security_group.my-sg: Creating...
aws_security_group.my-sg: Creation complete after 5s [id=sg-085c8330ef0965240]
aws_instance.amazon_linux-ec2-hw1: Creating...
aws_instance.amazon_linux-ec2-hw1: Still creating... [10s elapsed]
aws_instance.amazon_linux-ec2-hw1: Still creating... [10s elapsed]
aws_instance.amazon_linux-ec2-hw1: Creation complete after 16s [id=i-07e0eb547d51ee515]
aws_instance.amazon_linux-ec2-hw1: Creation complete after 16s [id=i-00d1f02168a3a41fb]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

```

The screenshot shows the VS Code interface with several tabs open: ec2.tf, amazon-ec2.sh, ubuntu-ec2.sh, and sg.tf. The sg.tf file contains Terraform code to create a security group and two instances. The terminal tab shows the Terraform plan and apply process, resulting in three resources being created. The status bar indicates the date and time as Sun 8 Dec 7:45:06 PM.

| Name             | Instance ID         | Instance state | Instance type | Status check | Alarm status                  | Availability Zone |
|------------------|---------------------|----------------|---------------|--------------|-------------------------------|-------------------|
| ubuntu-ec2-hw1   | i-00d1f02168a3a41fb | Running        | t2.micro      | Initializing | <a href="#">View alarms +</a> | us-east-1b        |
| amazon_linux-... | i-07e0eb547d51ee515 | Running        | t2.micro      | Initializing | <a href="#">View alarms +</a> | us-east-1b        |
|                  | i-0c44b849f6f2dd748 | Terminated     | t2.micro      | -            | <a href="#">View alarms +</a> | us-east-1c        |
| amazon_linux-... | i-04876e7b1e74566e2 | Terminated     | t2.micro      | -            | <a href="#">View alarms +</a> | us-east-1b        |
| ubuntu-ec2-hw1   | i-0be56162c4e9e2f9a | Terminated     | t2.micro      | -            | <a href="#">View alarms +</a> | us-east-1b        |

The screenshot shows the AWS CloudWatch Metrics dashboard. The left sidebar has sections for Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The main area displays a table of EC2 instances with their details like instance ID, state, type, and availability zone. The instance 'ubuntu-ec2-hw1' is highlighted in the table. The status bar at the bottom indicates the date and time as Sun 8 Dec 7:45:15 PM.

Chrome File Edit View History Bookmarks Profiles Tab Window Help

lec02-lowerbo | dev Create a MySQL | How to Secure | Instances | EC | ChatGPT | linux - How to | DNS / Names | YahyaAhmedK | Sun 8 Dec 7:45:18 PM

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances:

aws | Search [Option+S]

Dashboard EC2 Global View Events

**Instances**

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations

**Images**

- AMIs
- AMI Catalog

**Elastic Block Store**

- Volumes
- Snapshots
- Lifecycle Manager

CloudShell Feedback

Instances (5) Info Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states ▾

| Name            | Instance ID         | Instance state | Instance type | Status check      | Alarm status  | Availability Zone |
|-----------------|---------------------|----------------|---------------|-------------------|---------------|-------------------|
| ubuntu-ec2-hw1  | i-00d1f02168a3a41fb | Running        | t2.micro      | 2/2 checks passed | View alarms + | us-east-1b        |
| amazon_linux... | i-07e0eb547d51ee515 | Running        | t2.micro      | 2/2 checks passed | View alarms + | us-east-1b        |
|                 | i-0c44b849f6f2dd748 | Terminated     | t2.micro      | -                 | View alarms + | us-east-1c        |
|                 | i-04876e7b1e74566e2 | Terminated     | t2.micro      | -                 | View alarms + | us-east-1b        |
|                 | i-0be56162c4e9e2f9a | Terminated     | t2.micro      | -                 | View alarms + | us-east-1b        |

Select an instance

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms

Chrome File Edit View History Bookmarks Profiles Tab Window Help

lec02-lowerbo | dev Create a MySQL | How to Secure | Instance detail | ChatGPT | linux - How to | DNS / Names | YahyaAhmedK | Sun 8 Dec 7:45:29 PM

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-07e0eb547d51ee515...

aws | Search [Option+S]

EC2 > Instances > i-07e0eb547d51ee515

Dashboard EC2 Global View Events

**Instances**

- Instances
- Instance Types
- Launch Templates
- Spot Requests
- Savings Plans
- Reserved Instances
- Dedicated Hosts
- Capacity Reservations

**Images**

- AMIs
- AMI Catalog

**Elastic Block Store**

- Volumes
- Snapshots
- Lifecycle Manager

CloudShell Feedback

Instance summary for i-07e0eb547d51ee515 (amazon\_linux-ec2-hw1) Info

Updated less than a minute ago

|                                  |                                       |                                 |  |                               |  |
|----------------------------------|---------------------------------------|---------------------------------|--|-------------------------------|--|
| Instance ID                      | i-07e0eb547d51ee515                   | Public IPv4 address             | 3.235.103.152   open address ↗                 | Private IPv4 addresses        | 172.31.7.163   |
| IPv6 address                     | -                                     | Instance state                  | Running  | Public IPv4 DNS               | ec2-3-235-103-152.compute-1.amazonaws.com   open address ↗ |
| Hostname type                    | IP name: ip-172-31-7-163.ec2.internal | Private IP DNS name (IPv4 only) | ip-172-31-7-163.ec2.internal                   | Elastic IP addresses          | -  |
| Answer private resource DNS name | -                                     | Instance type                   | t2.micro                                       | AWS Compute Optimizer finding | Opt-in to AWS Compute Optimizer for recommendations.       |
| Auto-assigned IP address         | 3.235.103.152 [Public IP]             | VPC ID                          | vpc-030c196b1d96ded76 ↗                        | Learn more ↗                  | -  |
| IAM Role                         | tf_ec2s3 ↗                            | Subnet ID                       | subnet-070061a2764927c94 ↗                     | Auto Scaling Group name       | -  |
| IMDSv2                           | Required                              | Instance ARN                    | arn:aws:ec2:us-east-1:831926601934:instance/i- | Managed                       | false  |

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```

ec2.tf U
amazon-ec2.sh U
ubuntu-ec2.sh U
sg.tf U

```

```

userdata > amazon-ec2.sh
99 if [ "$(sudo docker ps -a -q -f name=reactapp-multistage)" ]; then
100     echo "Stopping and removing existing container 'reactapp-multistage'..."
101     sudo docker stop reactapp-multistage || true
102     sudo docker rm reactapp-multistage || true
103 fi
104
105 # Run the multi-stage Docker container using Nginx
106 echo "Running multi-stage Docker container for React app..."
107 sudo docker run -d -p 8080:80 --name reactapp-multistage reactapp-multistage
108
109 # Configure Nginx for reverse proxy
110 echo "Configuring Nginx reverse proxy for reactapp.yahyaabdullahsaadsabata.online..."
111 sudo bash -c 'cat > /etc/nginx/conf.d/reactapp.yahyaabdullahsaadsabata.online.conf <<EOF
112 server {
113     listen 80;
114     server_name yahya-hw1-amzn.yahyaabdullahsaadsabata.online;
115
116     location / {
117         proxy_pass http://localhost:8080;
118         proxy_set_header Host \$host;
119         proxy_set_header X-Real-IP \$remote_addr;
120         proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
121         proxy_set_header X-Forwarded-Proto \$scheme;
122     }
123 }
124 EOF'
125
126 # Test Nginx configuration
127 echo "Testing Nginx configuration..."
128 sudo nginx -t

```

OUTLINE  
TIMELINE

Ln 12, Col 27 Spaces: 4 UTF-8 LF Shell Script

## Comparison of Sizes

```

Termius File Edit View Window Help
Vaults SFTP h1test +

```

```

Removing intermediate container 6b596c9df038
--> 110d355d7e02
Step 7/9 : FROM nginx:alpine
--> 91ca84b4f577
Step 8/9 : COPY --from=build /app/build /usr/share/nginx/html
--> ce88bd27f6b5
Step 9/9 : EXPOSE 80
--> Running in 26756c10a238
Removing intermediate container 26756c10a238
--> 7b243c01fa28
Successfully built 7b243c01fa28
Successfully tagged reactapp-multistage:latest
Stopping and removing the standard React app container...
reactapp-standard
reactapp-standard
Running multi-stage Docker container for React app...
a57e62b5e878e9c4be35d8baf430a18e84866fdac86e9ca85b8a33f2577406c0
Configuring Nginx reverse proxy for reactapp.yahyaabdullahsaadsabata.online...
Testing Nginx configuration...
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
Restarting Nginx...
React app deployed in two stages: standard Docker build and multi-stage Docker build with Nginx reverse proxy!
ubuntu@ip-172-31-93-16:~/scripts$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
reactapp-multistage latest 7b243c01fa28 59 seconds ago 53.1MB
<none> <none> 110d355d7e02 About a minute ago 454MB
reactapp-standard latest 13b3f03dd0b9 About a minute ago 453MB
nginx alpine 91ca84b4f577 11 days ago 52.5MB
node 16-alpine 2573171e0124 16 months ago 118MB
ubuntu@ip-172-31-93-16:~/scripts$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
a57e62b5e878 reactapp-multistage "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:8080-
>80/tcp, :::8080->80/tcp reactapp-multistage
ubuntu@ip-172-31-93-16:~/scripts$ 

```

## Setup letsencrypt for HTTPS

```
Termius File Edit View Window Help
Vaults SFTP 52.201.227.30 + Sun 8 Dec 7:04:49 PM

ubuntu@ip-172-31-82-38:/etc/nginx/sites-available$ sudo python3 -m venv /opt/certbot/
ubuntu@ip-172-31-82-38:/etc/nginx/sites-available$ sudo /opt/certbot/bin/pip install --upgrade pip
Requirement already satisfied: pip in /opt/certbot/lib/python3.10/site-packages (22.0.2)
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
    1.8/1.8 MB 14.9 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.2
    Uninstalling pip-22.0.2:
      Successfully uninstalled pip-22.0.2
Successfully installed pip-24.3.1
ubuntu@ip-172-31-82-38:/etc/nginx/sites-available$ sudo /opt/certbot/bin/pip install certbot certbot-nginx
Collecting certbot
  Downloading certbot-3.0.1-py3-none-any.whl.metadata (8.0 kB)
Collecting certbot-nginx
  Downloading certbot_nginx-3.0.1-py3-none-any.whl.metadata (1.4 kB)
Collecting acme>=3.0.1 (from certbot)
  Downloading acme-3.0.1-py3-none-any.whl.metadata (1.4 kB)
Collecting ConfigArgParse>=1.5.3 (from certbot)
  Downloading ConfigArgParse-1.7-py3-none-any.whl.metadata (23 kB)
Collecting configobj>=5.0.6 (from certbot)
  Downloading configobj-5.0.9-py2.py3-none-any.whl.metadata (3.2 kB)
Collecting cryptography>=3.2.1 (from certbot)
  Downloading cryptography-44.0.0-cp39-manylinux_2_34_x86_64.whl.metadata (5.7 kB)
Collecting distro>=1.0.1 (from certbot)
  Downloading distro-1.9.0-py3-none-any.whl.metadata (6.8 kB)
Collecting josepy<2,>=1.13.0 (from certbot)
  Downloading josepy-1.14.0-py3-none-any.whl.metadata (1.8 kB)
Collecting parsedatetime>=2.4 (from certbot)
  Downloading parsedatetime-2.6-py3-none-any.whl.metadata (4.7 kB)
Collecting pyRFC3339 (from certbot)
  Downloading pyRFC3339-2.0.1-py3-none-any.whl.metadata (2.1 kB)
Collecting pytz>=2019.3 (from certbot)
  Downloading pytz-2024.2-py2.py3-none-any.whl.metadata (22 kB)

[...]
```

```
Termius File Edit View Window Help
Vaults SFTP 52.201.227.30 + Sun 8 Dec 7:04:51 PM

EFF news, campaigns, and ways to support digital freedom.
-----
(Y)es/(N)o: Y
Account registered.

Which names would you like to activate HTTPS for?
We recommend selecting either all domains, or all domains in a VirtualHost/server block.
-----
1: yahya-hw1-ubnt.yahyaabdullahsaadsabata.online
-----
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter '!' to cancel):
Requesting a certificate for yahya-hw1-ubnt.yahyaabdullahsaadsabata.online

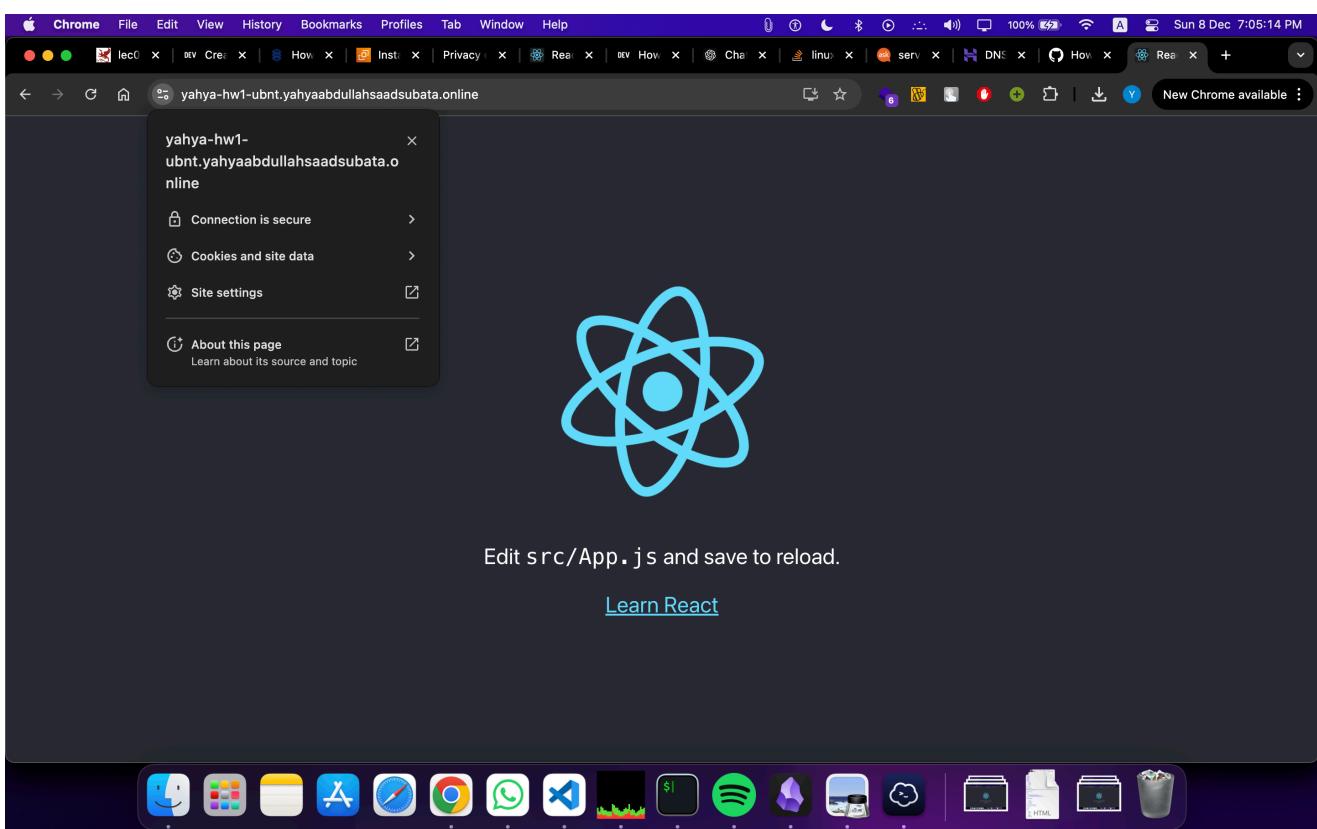
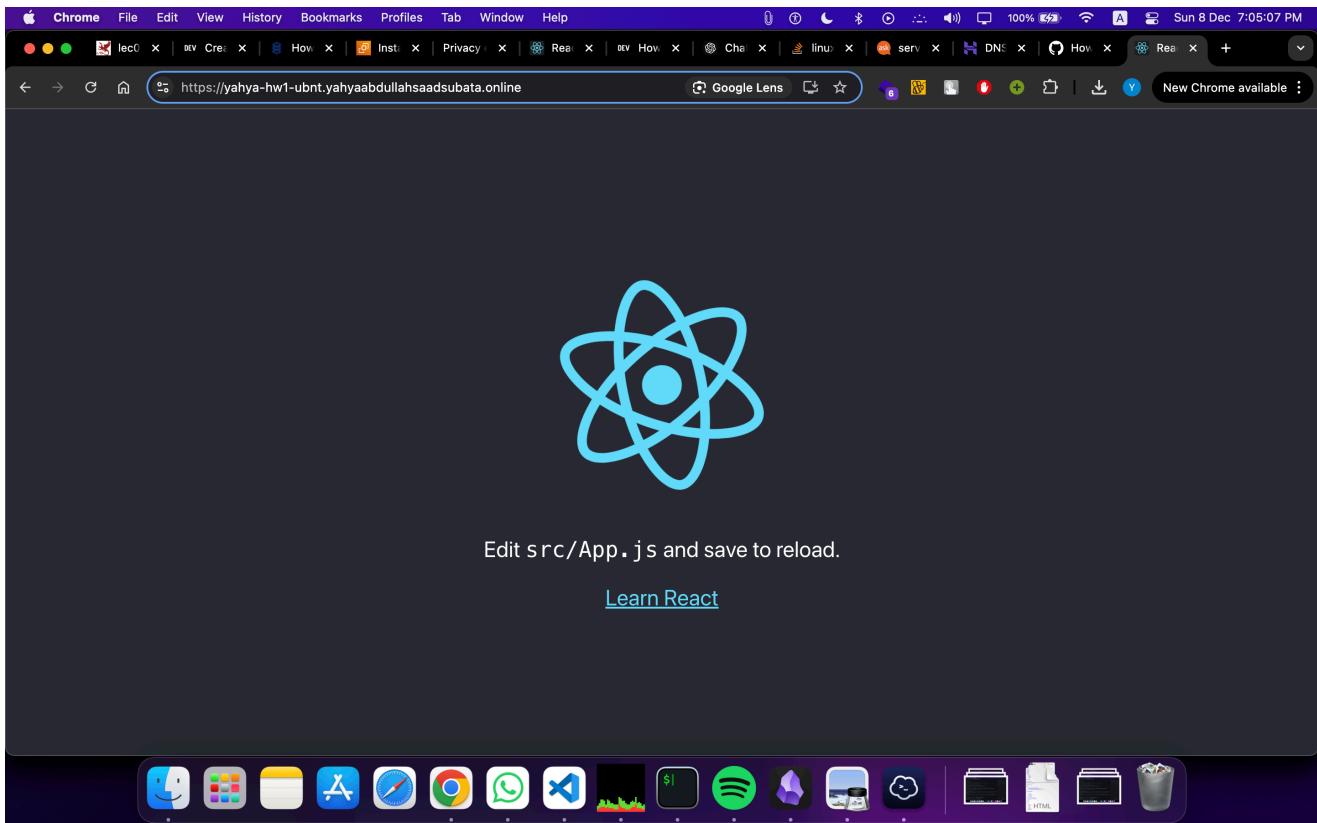
Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/yahya-hw1-ubnt.yahyaabdullahsaadsabata.online/fullchain.pem
Key is saved at: /etc/letsencrypt/live/yahya-hw1-ubnt.yahyaabdullahsaadsabata.online/privkey.pem
This certificate expires on 2025-03-08.
These files will be updated when the certificate renews.

Deploying certificate
Successfully deployed certificate for yahya-hw1-ubnt.yahyaabdullahsaadsabata.online to /etc/nginx/sites-enabled/reactapp.yahyaabdullahsaadsabata.online
Congratulations! You have successfully enabled HTTPS on https://yahya-hw1-ubnt.yahyaabdullahsaadsabata.online

NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can automatically renew the certificate in the
background, but you may need to take steps to enable that functionality. See https://certbot.org/renewal-setup for i
nstructions.

-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
-----
ubuntu@ip-172-31-82-38:/etc/nginx/sites-available$
```

**Showing https version being used**



## Code files

### Server conf for HTTPS

```
server {  
    listen      80;  
    listen      [::]:80;
```

```

server_name yahya-hw1-ubnt.yahyaabdullahsaadsubata.online;
location / {
    proxy_pass http://localhost:8080;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}
}

```

## Amazon Linux 2023 script.sh

```

#!/bin/bash

# Exit on error
set -e

# Update package list
echo "Updating package list..."
sudo dnf update -y

# Install Docker
echo "Installing Docker..."
sudo dnf install -y docker
sudo systemctl start docker
sudo systemctl enable docker

# Install Nginx
echo "Installing Nginx..."
sudo dnf install -y nginx
sudo systemctl start nginx
sudo systemctl enable nginx

# Install Git
echo "Installing Git..."
sudo dnf install -y git

# Clone the React app from GitHub
echo "Cloning the React app repository..."
REPO_URL="https://github.com/YahyaAhmedKhan/reactapp-devops-hafeez.git"
APP_DIR="reactapp"
if [ -d "$APP_DIR" ]; then
    echo "Directory $APP_DIR already exists. Pulling latest changes..."
    cd $APP_DIR && git pull && cd ..
else
    git clone $REPO_URL $APP_DIR

```

```
fi
cd $APP_DIR

# Step 1: Create a standard Dockerfile
echo "Creating standard Dockerfile for React app..."
cat > Dockerfile <<EOF
FROM node:16-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
EOF

# Check if the image already exists and remove it if it does
if [ "$(sudo docker images -q reactapp-standard)" ]; then
    echo "Removing existing image 'reactapp-standard'..."
    sudo docker rmi reactapp-standard || true
fi

# Build the standard Docker image
echo "Building standard Docker image for React app..."
sudo docker build -t reactapp-standard .

# Check if the container already exists and remove it if it does
if [ "$(sudo docker ps -a -q -f name=reactapp-standard)" ]; then
    echo "Stopping and removing existing container 'reactapp-standard'..."
    sudo docker stop reactapp-standard || true
    sudo docker rm reactapp-standard || true
fi

# Run the standard Docker container
echo "Running standard Docker container for React app..."
sudo docker run -d -p 3000:3000 --name reactapp-standard reactapp-standard

# Step 2: Modify the Dockerfile to use multi-stage build
echo "Modifying Dockerfile for multi-stage build..."
cat > Dockerfile <<EOF
# Stage 1: Build the React app
FROM node:16-alpine AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Stage 2: Serve React app with Nginx
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
```

```
EXPOSE 80
```

```
EOF
```

```
# Check if the image already exists and remove it if it does
if [ "$(sudo docker images -q reactapp-multistage)" ]; then
    echo "Removing existing image 'reactapp-multistage'..."
    sudo docker rmi reactapp-multistage || true
fi

# Build the multi-stage Docker image
echo "Building multi-stage Docker image for React app..."
sudo docker build -t reactapp-multistage .

# Check if the container already exists and remove it if it does
if [ "$(sudo docker ps -a -q -f name=reactapp-multistage)" ]; then
    echo "Stopping and removing existing container 'reactapp-
multistage'..."
    sudo docker stop reactapp-multistage || true
    sudo docker rm reactapp-multistage || true
fi

# Run the multi-stage Docker container using Nginx
echo "Running multi-stage Docker container for React app..."
sudo docker run -d -p 8080:80 --name reactapp-multistage reactapp-
multistage

# Configure Nginx for reverse proxy
echo "Configuring Nginx reverse proxy for
reactapp.yahyaabdullahsaadsubata.online..."
sudo bash -c 'cat >
/etc/nginx/conf.d/reactapp.yahyaabdullahsaadsubata.online.conf <<EOF
server {
    listen 80;
    server_name yahya-hw1-amzn.yahyaabdullahsaadsubata.online;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
    }
}
EOF'

# Test Nginx configuration
echo "Testing Nginx configuration..."
sudo nginx -t

# Restart Nginx to apply changes
```

```

echo "Restarting Nginx..."
sudo systemctl restart nginx

# Success message
echo "React app deployed in two stages: standard Docker build and multi-
stage Docker build with Nginx reverse proxy!"

```

## Ubuntu 22.04 script.sh

```

#!/bin/bash

# Exit on error
set -e

sudo sed -i 's/#$nrconf{restart} = "i";/$nrconf{restart} =
"a";/g' /etc/needrestart/needrestart.conf

# Update package list
echo "Updating package list..."
sudo apt update -y && sudo apt upgrade -y

# Install Docker
echo "Installing Docker..."
sudo apt install -y docker.io
sudo systemctl start docker
sudo systemctl enable docker

# Install Nginx
echo "Installing Nginx..."
sudo apt install -y nginx
sudo systemctl start nginx
sudo systemctl enable nginx

# Install Git
echo "Installing Git..."
sudo apt install -y git

# Clone the React app from GitHub
echo "Cloning the React app repository..."
REPO_URL="https://github.com/YahyaAhmedKhan/reactapp-devops-hafeez.git"
APP_DIR="reactapp"
if [ -d "$APP_DIR" ]; then
    echo "Directory $APP_DIR already exists. Pulling latest changes..."
    cd $APP_DIR && git pull && cd ..
else
    git clone $REPO_URL $APP_DIR
fi
cd $APP_DIR

```

```
# Step 1: Create a standard Dockerfile
echo "Creating standard Dockerfile for React app..."
cat > Dockerfile <<EOF
FROM node:16-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
EOF

# Check if the image already exists and remove it if it does
if [ "$(sudo docker images -q reactapp-standard)" ]; then
    echo "Removing existing image 'reactapp-standard'..."
    sudo docker rmi reactapp-standard || true
fi

# Build the standard Docker image
echo "Building standard Docker image for React app..."
sudo docker build -t reactapp-standard .

# Check if the container already exists and remove it if it does
if [ "$(sudo docker ps -a -q -f name=reactapp-standard)" ]; then
    echo "Stopping and removing existing container 'reactapp-standard'..."
    sudo docker stop reactapp-standard || true
    sudo docker rm reactapp-standard || true
fi

# Run the standard Docker container
echo "Running standard Docker container for React app..."
sudo docker run -d -p 3000:3000 --name reactapp-standard reactapp-standard

# Step 2: Modify the Dockerfile to use multi-stage build
echo "Modifying Dockerfile for multi-stage build..."
cat > Dockerfile <<EOF
# Stage 1: Build the React app
FROM node:16-alpine AS build
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Stage 2: Serve React app with Nginx
FROM nginx:alpine
COPY --from=build /app/build /usr/share/nginx/html
EXPOSE 80
EOF
```

```

# Check if the image already exists and remove it if it does
if [ "$(sudo docker images -q reactapp-multistage)" ]; then
    echo "Removing existing image 'reactapp-multistage'..."
    sudo docker rmi reactapp-multistage || true
fi

# Build the multi-stage Docker image
echo "Building multi-stage Docker image for React app..."
sudo docker build -t reactapp-multistage .

# Check if the container already exists and remove it if it does
if [ "$(sudo docker ps -a -q -f name=reactapp-multistage)" ]; then
    echo "Stopping and removing existing container 'reactapp-
multistage'..."
    sudo docker stop reactapp-multistage || true
    sudo docker rm reactapp-multistage || true
fi

# Run the multi-stage Docker container using Nginx
echo "Running multi-stage Docker container for React app..."
sudo docker run -d -p 8080:80 --name reactapp-multistage reactapp-
multistage

# Remove the default configuration if it exists
echo "Removing default configuration if it exists..."
if [ -f /etc/nginx/sites-available/default ]; then
    sudo rm /etc/nginx/sites-available/default
fi
if [ -L /etc/nginx/sites-enabled/default ]; then
    sudo rm /etc/nginx/sites-enabled/default
fi

# Configure Nginx reverse proxy
echo "Configuring Nginx reverse proxy for
reactapp.yahyaabdullahsaadsabata.online..."
sudo bash -c 'cat > /etc/nginx/sites-
available/reactapp.yahyaabdullahsaadsabata.online <<EOF
server {
    listen 80;
    server_name yahya-hw1-ubnt.yahyaabdullahsaadsabata.online;

    location / {
        proxy_pass http://localhost:8080;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto \$scheme;
}
EOF
'

```

```
    }
}

EOF'

# Remove existing symlink if it exists
echo "Checking if symlink already exists in sites-enabled..."
if [ -L /etc/nginx/sites-enabled/reactapp.yahyaabdullahsaadsubata.online
]; then
    echo "Symlink already exists, removing it..."
    sudo rm /etc/nginx/sites-
enabled/reactapp.yahyaabdullahsaadsubata.online
fi

# Create a symbolic link in sites-enabled to enable the site
echo "Enabling the site..."
sudo ln -s /etc/nginx/sites-
available/reactapp.yahyaabdullahsaadsubata.online /etc/nginx/sites-
enabled/

# Test Nginx configuration
echo "Testing Nginx configuration..."
sudo nginx -t

# Restart Nginx to apply changes
echo "Restarting Nginx..."
sudo systemctl restart nginx

# Success message
echo "React app deployed in two stages: standard Docker build and multi-
stage Docker build with Nginx reverse proxy!"
```