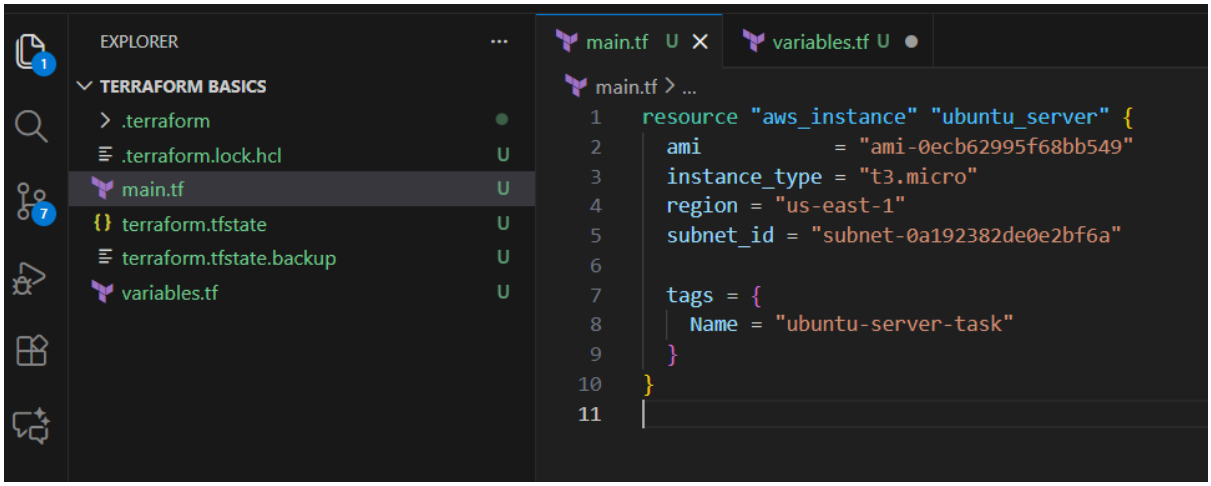
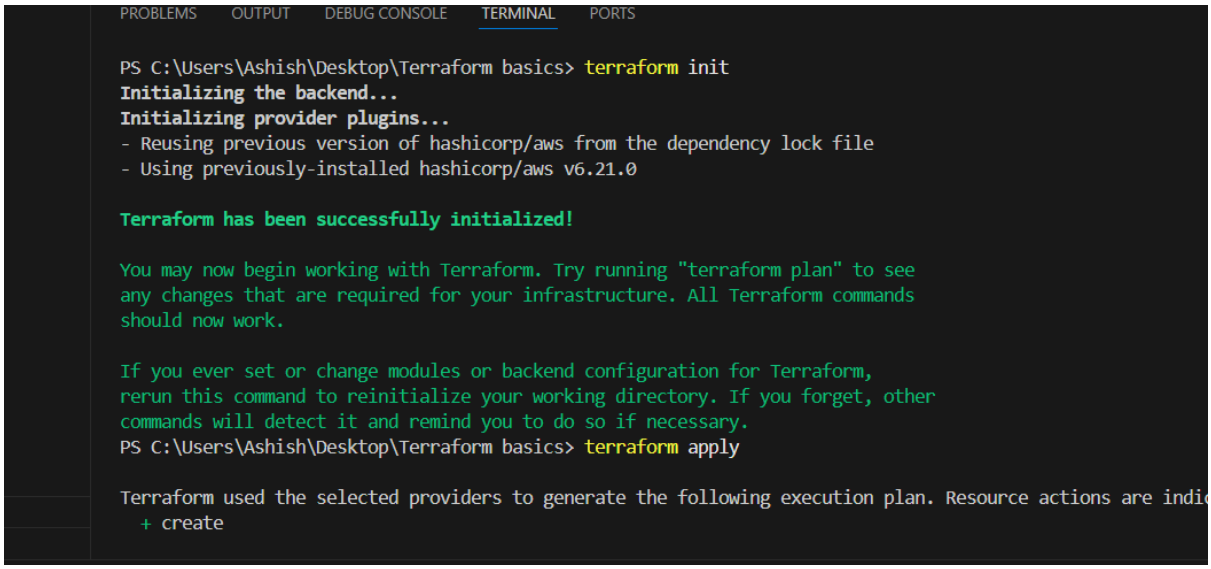
Write a code using Terraform that deploys and achieves the following:-

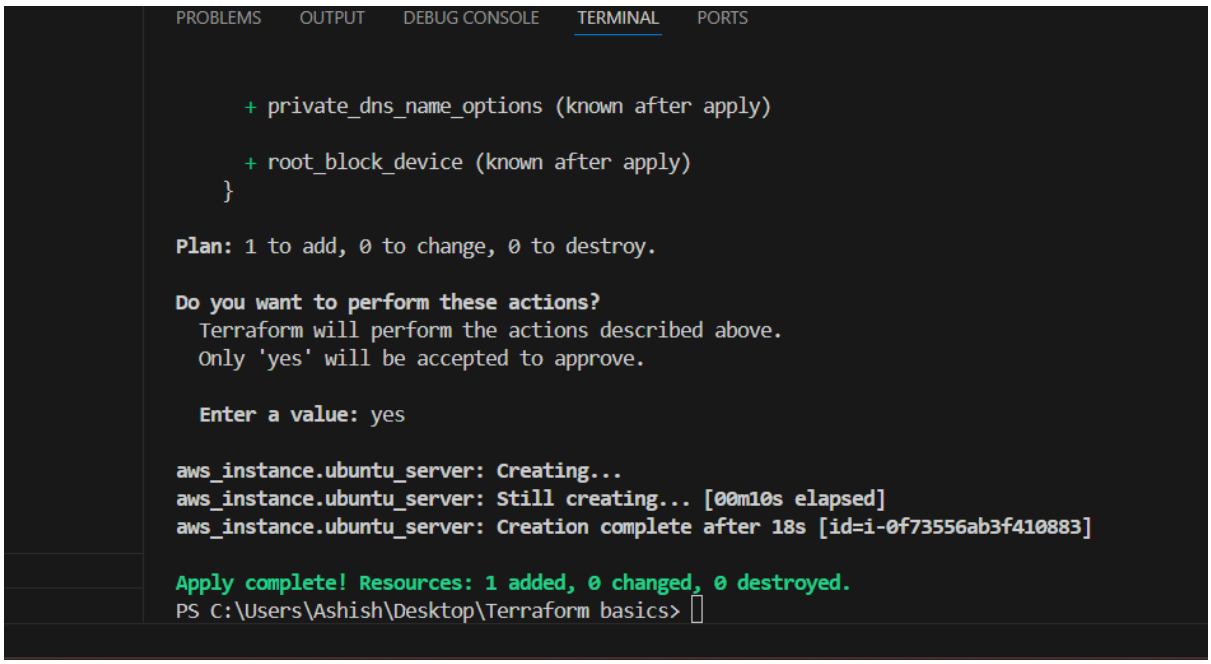
1. **Create an EC2 Instance on AWS Cloud using the below-mentioned ubuntu AMI and instance_type. Or, (You can use of your choice as well) ami = " ami-0ecb62995f68bb549" instance_type = "t3.micro"**

Give this script in main.tf

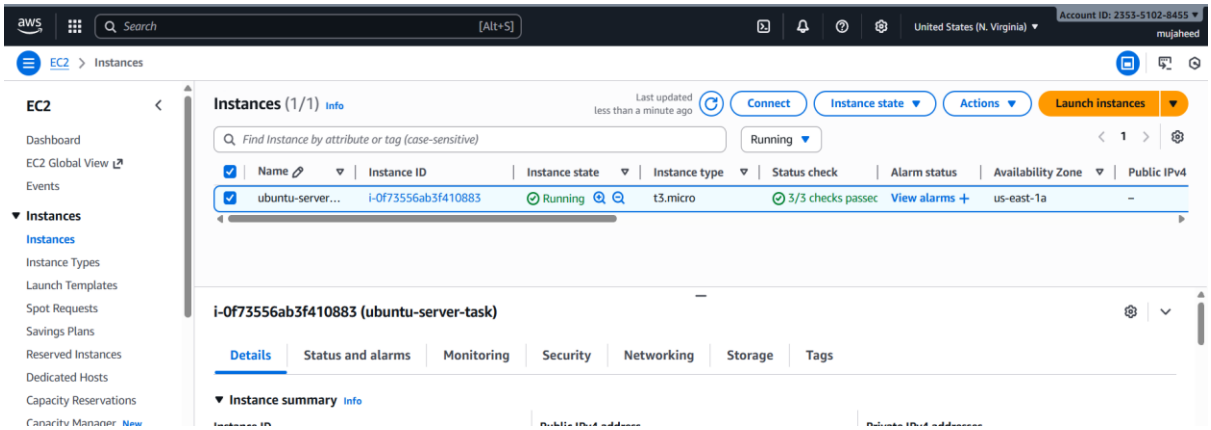
```
resource "aws_instance" "ubuntu_server" {
  ami           = "ami-0ecb62995f68bb549"
  instance_type = "t3.micro"
  region = "us-east-1"
  subnet_id = "subnet-0a192382de0e2bf6a"

  tags = {
    Name = "ubuntu-server-task"
  }
}
```







An ubuntu server instance has been created.



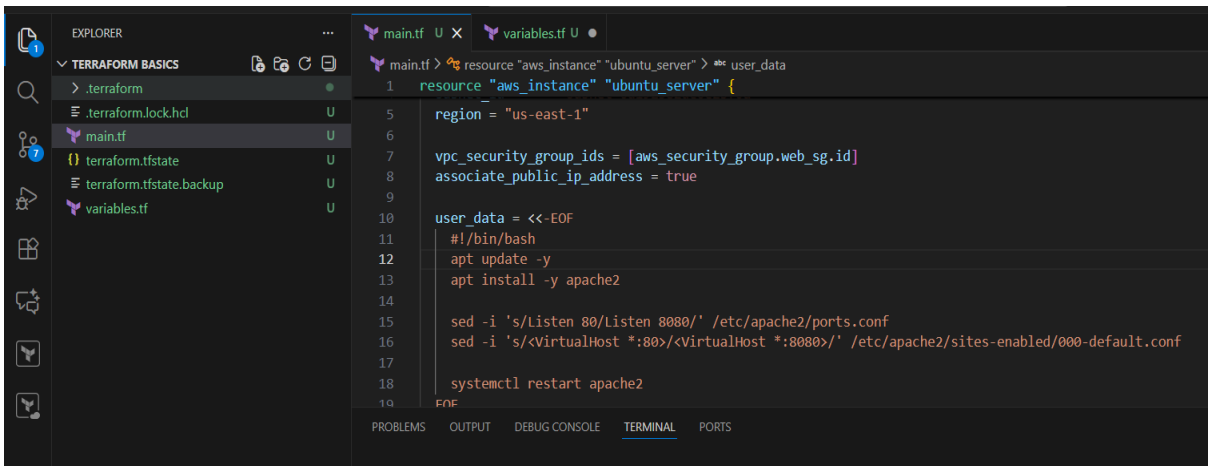
2. Install a web server on the same EC2 Instance. The web server should listen on port 8080.

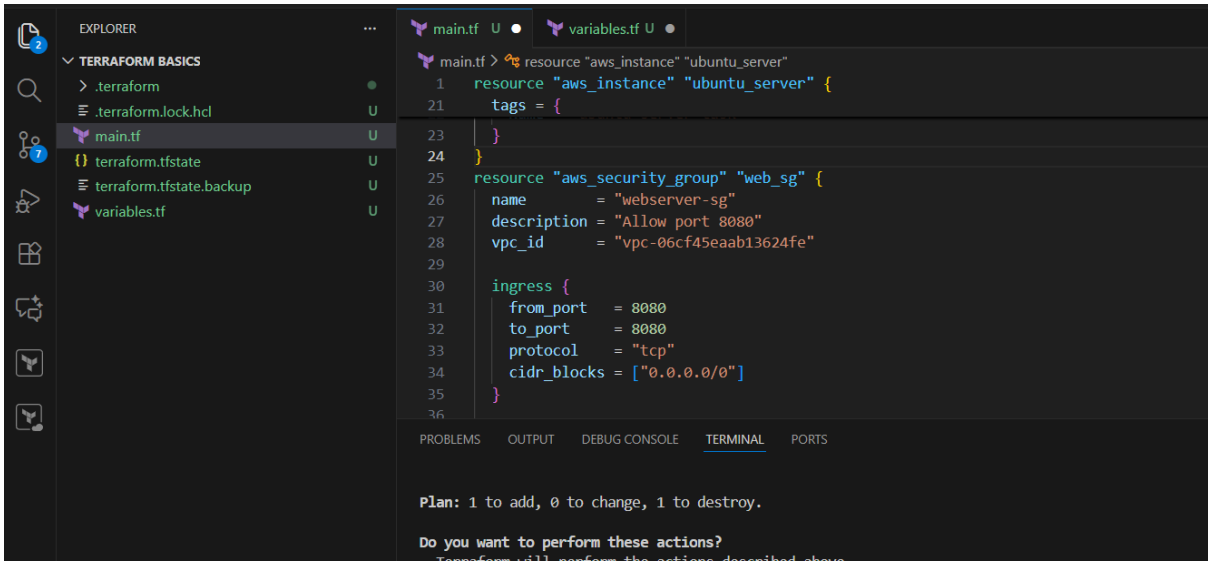
Give the code like this in resource block in main.tf

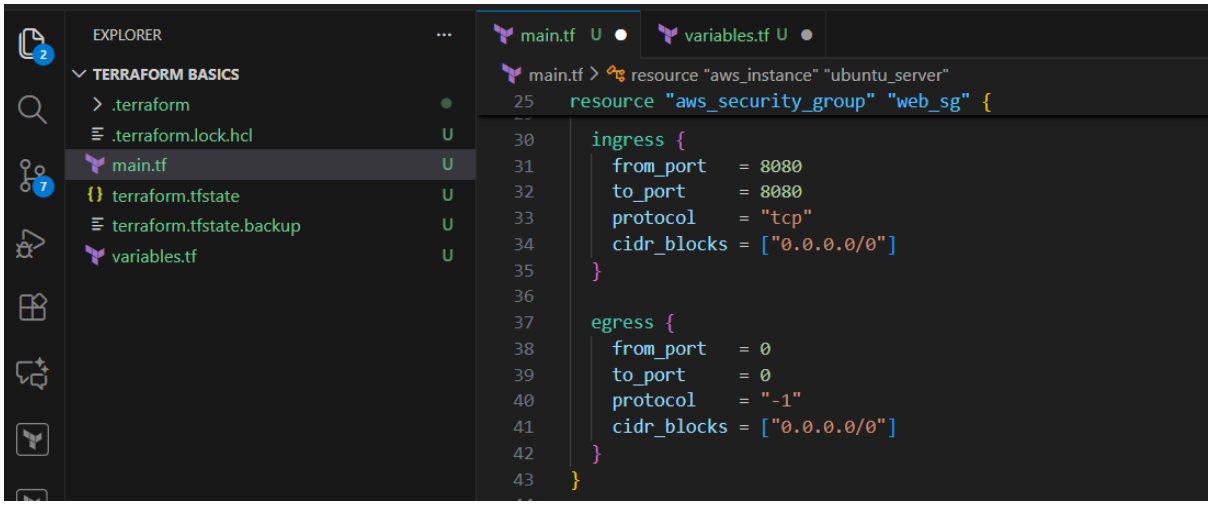
```
resource "aws_instance" "ubuntu_server" {
  ami           = "ami-0ecb62995f68bb549"
  instance_type = "t3.micro"
  subnet_id     = "subnet-0a192382de0e2bf6a"
  region = "us-east-1"

  vpc_security_group_ids = [aws_security_group.web_sg.id]
  associate_public_ip_address = true

  user_data = <<-EOF
    #!/bin/bash
    apt update -y
```

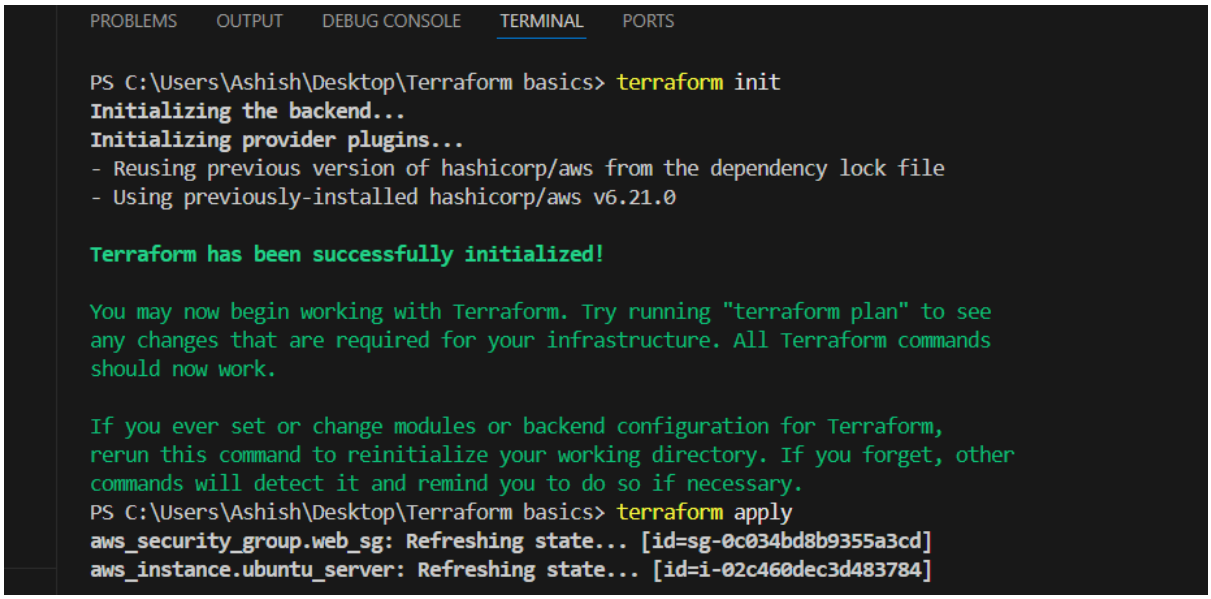
```
    apt install -y apache2

    sed -i 's/Listen 80/Listen 8080/' /etc/apache2/ports.conf
    sed -i 's/<VirtualHost *:80>/<VirtualHost *:8080>/'
/etc/apache2/sites-enabled/000-default.conf

    systemctl restart apache2
  EOF

  tags = {
    Name = "ubuntu-server-task"
  }
}
resource "aws_security_group" "web_sg" {
  name        = "webserver-sg"
  description = "Allow port 8080"
  vpc_id      = "vpc-06cf45eaab13624fe"

  ingress {
    from_port   = 8080
    to_port     = 8080
```

```
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```




EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl
main.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
main.tf U
variables.tf U
main.tf > resource "aws_instance" "ubuntu_server"
1 resource "aws_instance" "ubuntu_server" {
21 tags = {
23 }
24 }
25 resource "aws_security_group" "web_sg" {
26 name = "webserver-sg"
27 description = "Allow port 8080"
28 vpc_id = "vpc-06cf45eaab13624fe"
30 ingress {
31 from_port = 8080
32 to_port = 8080
33 protocol = "tcp"
34 cidr_blocks = ["0.0.0.0/0"]
35 }
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Plan: 1 to add, 0 to change, 1 to destroy.
Do you want to perform these actions?


EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl
main.tf
terraform.tfstate
terraform.tfstate.backup
variables.tf
main.tf U
variables.tf U
main.tf > resource "aws_instance" "ubuntu_server"
25 resource "aws_security_group" "web_sg" {
30 ingress {
31 from_port = 8080
32 to_port = 8080
33 protocol = "tcp"
34 cidr_blocks = ["0.0.0.0/0"]
35 }
37 egress {
38 from_port = 0
39 to_port = 0
40 protocol = "-1"
41 cidr_blocks = ["0.0.0.0/0"]
42 }
43 }


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.21.0
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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_security_group.web_sg: Refreshing state... [id=sg-0c034bd8b9355a3cd]
aws_instance.ubuntu_server: Refreshing state... [id=i-02c460dec3d483784]

```
Plan: 1 to add, 0 to change, 1 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_instance.ubuntu_server: Destroying... [id=i-02c460dec3d483784]
aws_instance.ubuntu_server: Still destroying... [id=i-02c460dec3d483784, 00m10s elapsed]
aws_instance.ubuntu_server: Still destroying... [id=i-02c460dec3d483784, 00m20s elapsed]
aws_instance.ubuntu_server: Still destroying... [id=i-02c460dec3d483784, 00m30s elapsed]
aws_instance.ubuntu_server: Destruction complete after 33s
aws_instance.ubuntu_server: Creating...
aws_instance.ubuntu_server: Still creating... [00m10s elapsed]
aws_instance.ubuntu_server: Creation complete after 15s [id=i-09d4d87fd245422e9]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
PS C:\Users\Ashish\Desktop\Terraform basics>
```

Copy your public ip and search in browser with the port number 8080. You will find the apche2 webpage.

3. Create a Docker-compose file for the project: https://github.com/betawins/cricbuzz-docker-hello-world.git and point it to port 8000.

* https://github.com/mujaheed00/cricbuzz-docker-hello-world.git

Give this script in main.tf

```
provider "aws" {
region = "us-east-1"
}
resource "aws_security_group" "web_sg" {
name = "webserver-sg"
description = "Allow SSH and port 8000"
vpc_id = "vpc-06cf45eaab13624fe"
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
ingress {
from_port = 8000
to_port = 8000
protocol = "tcp"
```

```
cidr_blocks = ["0.0.0.0/0"]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
resource "aws_instance" "ubuntu_server" {
ami = "ami-0ecb62995f68bb549"
instance_type = "t3.micro"
subnet_id = "subnet-0a192382de0e2bf6a"
associate_public_ip_address = true
vpc_security_group_ids = [aws_security_group.web_sg.id]
user_data = <<-USERDATA
#!/bin/bash
set -e
# Update packages
apt update -y
# Install Docker
```

apt install -y docker.io

systemctl enable docker

systemctl start docker

 # Install Git

apt install -y git

 # Install Docker Compose v2

**curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) **

-o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

 # Clone your repository

cd /home/ubuntu

git clone https://github.com/mujaheed00/cricbuzz-docker-hello-world.git

cd cricbuzz-docker-hello-world

 # Fix permissions

chown -R ubuntu:ubuntu /home/ubuntu/cricbuzz-docker-hello-world

 # Create docker-compose.yml

cat <<'COMPOSE' > docker-compose.yml

services:

cricbuzz-app:

build: .

container_name: cricbuzz-app

ports:

- "8000:8080"

restart: always

COMPOSE

 # Build and run the app

/usr/local/bin/docker-compose up --build -d

USERDATA

tags = {

Name = "ubuntu-server-task"

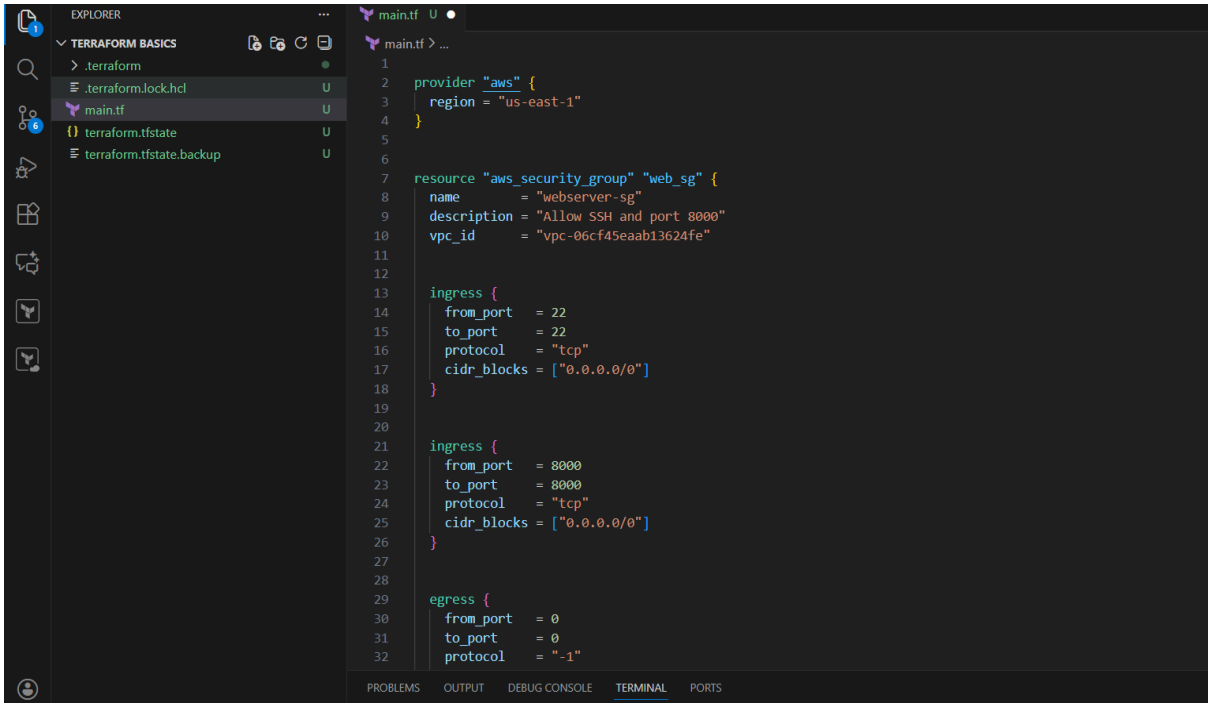
}

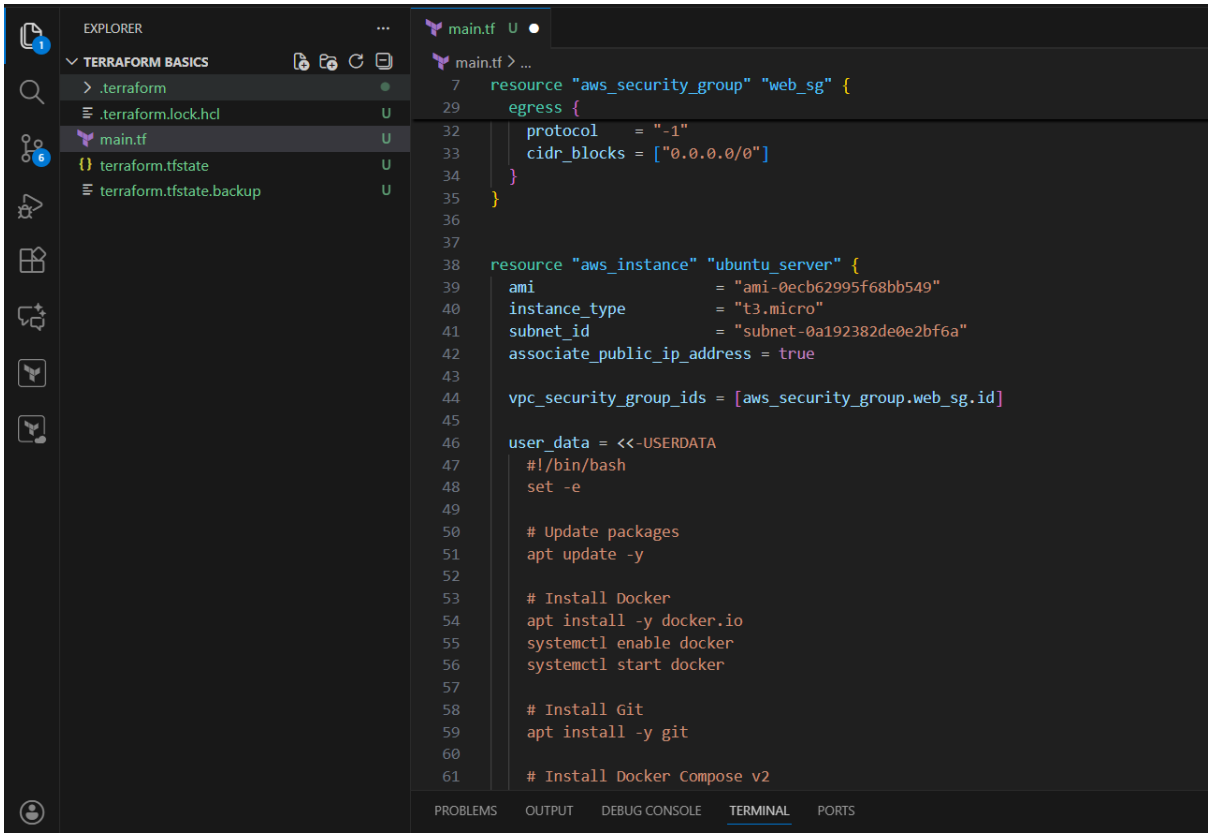
}

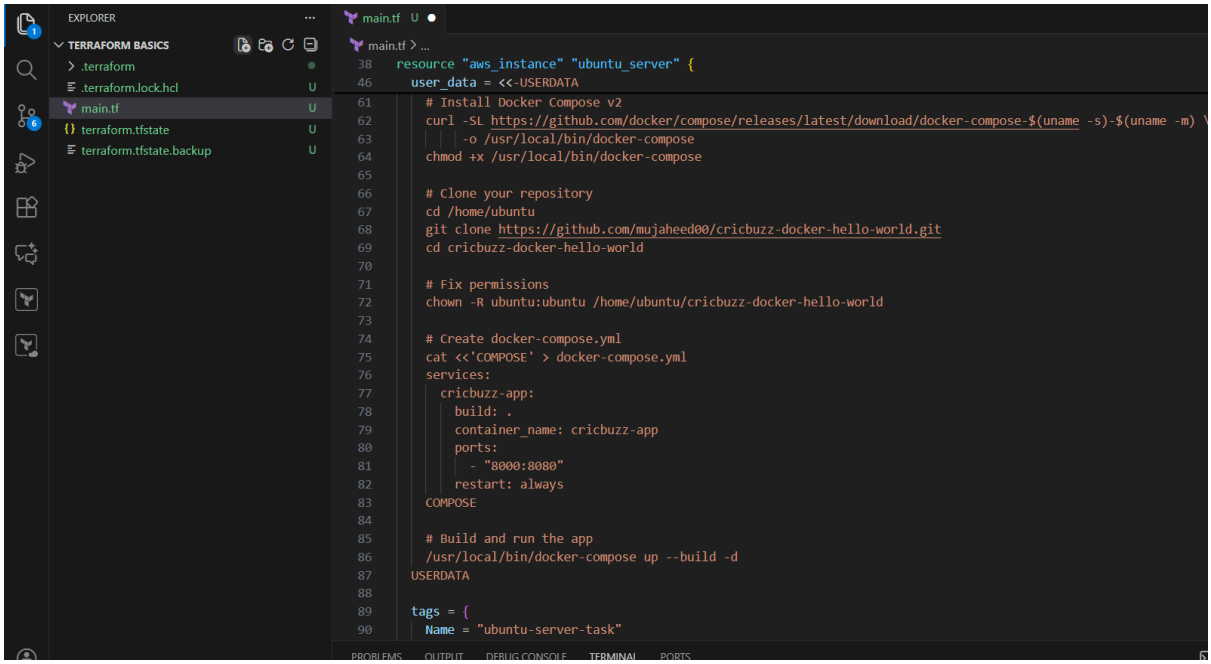
output "public_ip" {

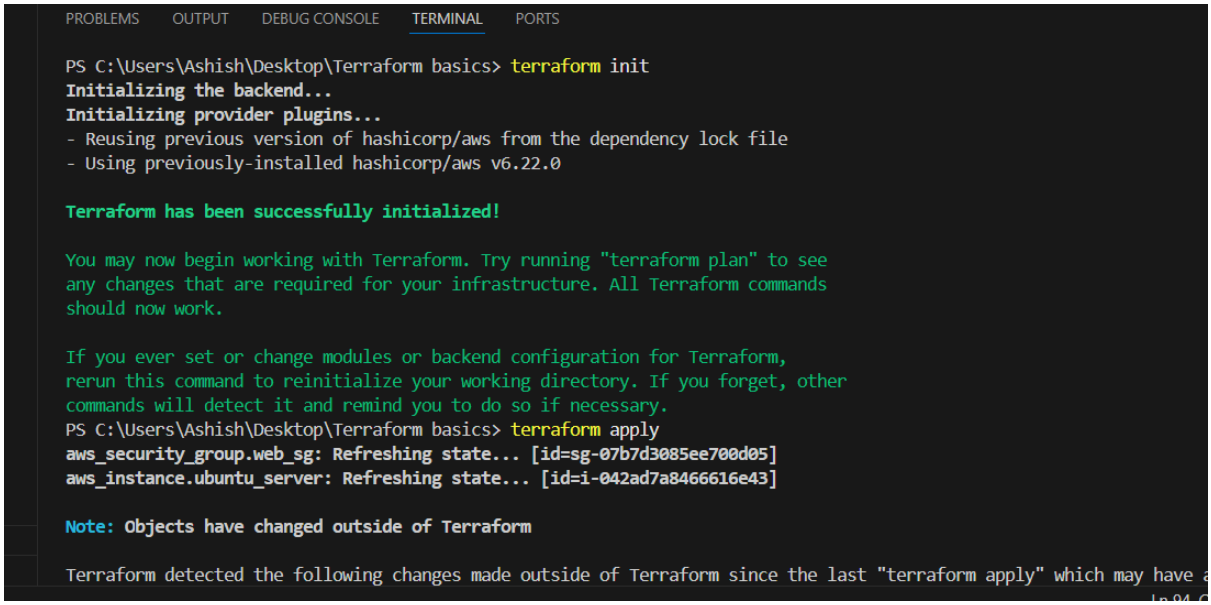
value = aws_instance.ubuntu_server.public_ip

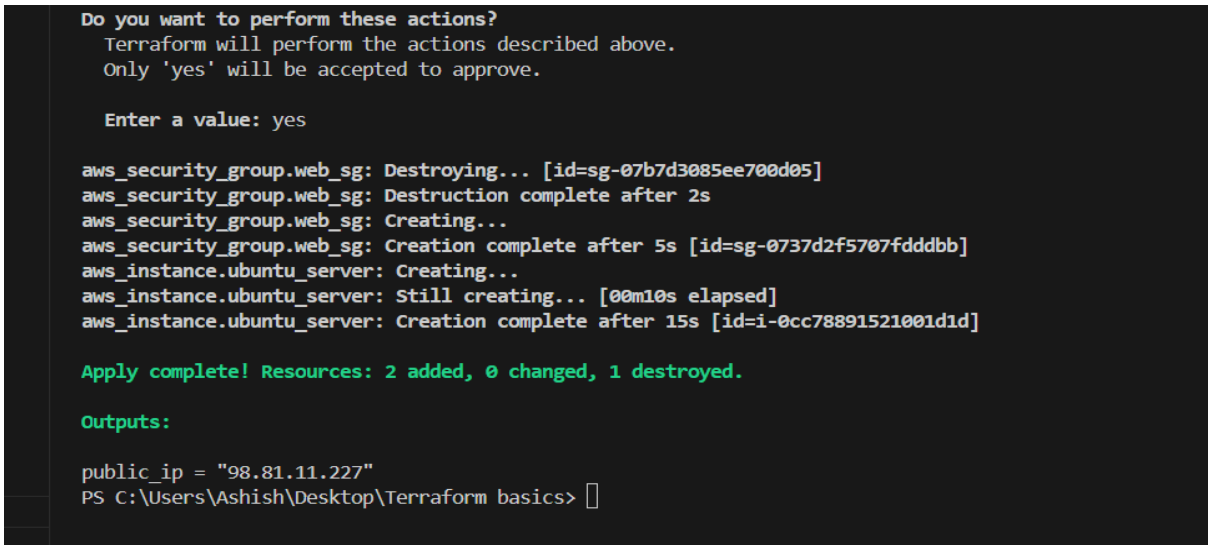
}


EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl U
main.tf U
terraform.tfstate U
terraform.tfstate.backup U
main.tf U
main.tf > ...
provider "aws" {
region = "us-east-1"
}
resource "aws_security_group" "web_sg" {
name = "webserver-sg"
description = "Allow SSH and port 8000"
vpc_id = "vpc-06cf45eaab13624fe"
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
ingress {
from_port = 8000
to_port = 8000
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

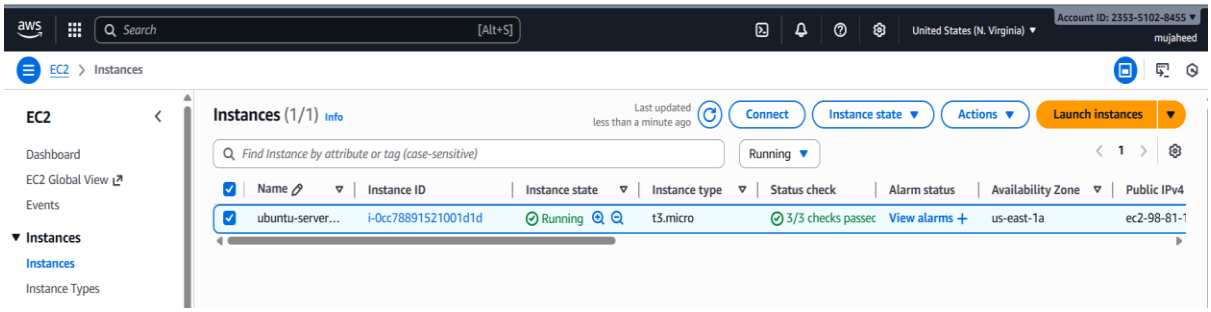

EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl U
main.tf U
terraform.tfstate U
terraform.tfstate.backup U
main.tf U
main.tf > ...
resource "aws_security_group" "web_sg" {
egress {
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
resource "aws_instance" "ubuntu_server" {
ami = "ami-0ecb62995f68bb549"
instance_type = "t3.micro"
subnet_id = "subnet-0a192382de0e2bf6a"
associate_public_ip_address = true
vpc_security_group_ids = [aws_security_group.web_sg.id]
user_data = <<-USERDATA
#!/bin/bash
set -e
Update packages
apt update -y
Install Docker
apt install -y docker.io
systemctl enable docker
systemctl start docker
Install Git
apt install -y git
Install Docker Compose v2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS


EXPLORER
TERRAFORM BASICS
.terraform
.terraform.lock.hcl U
main.tf U
terraform.tfstate U
terraform.tfstate.backup U
main.tf U
main.tf > ...
38 resource "aws_instance" "ubuntu_server" {
46 user_data = <<-USERDATA
61 # Install Docker Compose v2
62 curl -SL https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)
63 -o /usr/local/bin/docker-compose
64 chmod +x /usr/local/bin/docker-compose
65
66 # Clone your repository
67 cd /home/ubuntu
68 git clone https://github.com/mujaheed00/cricbuzz-docker-hello-world.git
69 cd cricbuzz-docker-hello-world
70
71 # Fix permissions
72 chown -R ubuntu:ubuntu /home/ubuntu/cricbuzz-docker-hello-world
73
74 # Create docker-compose.yml
75 cat <<'COMPOSE' > docker-compose.yml
76 services:
77 cricbuzz-app:
78 build: .
79 container_name: cricbuzz-app
80 ports:
81 - "8000:8080"
82 restart: always
83 COMPOSE
84
85 # Build and run the app
86 /usr/local/bin/docker-compose up --build -d
87 USERDATA
88
89 tags = {
90 Name = "ubuntu-server-task"
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.22.0
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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_security_group.web_sg: Refreshing state... [id=sg-07b7d3085ee700d05]
aws_instance.ubuntu_server: Refreshing state... [id=i-042ad7a8466616e43]
Note: Objects have changed outside of Terraform
Terraform detected the following changes made outside of Terraform since the last "terraform apply" which may have


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.web_sg: Destroying... [id=sg-07b7d3085ee700d05]
aws_security_group.web_sg: Destruction complete after 2s
aws_security_group.web_sg: Creating...
aws_security_group.web_sg: Creation complete after 5s [id=sg-0737d2f5707fdddbb]
aws_instance.ubuntu_server: Creating...
aws_instance.ubuntu_server: Still creating... [00m10s elapsed]
aws_instance.ubuntu_server: Creation complete after 15s [id=i-0cc78891521001d1d]
Apply complete! Resources: 2 added, 0 changed, 1 destroyed.
Outputs:
public_ip = "98.81.11.227"
PS C:\Users\Ashish\Desktop\Terraform basics>

An instance has been created.



Open the instance and execute this commands.

- cd ~/cricbuzz-docker-hello-world
- sudo docker-compose down
- sudo docker-compose up --build -d

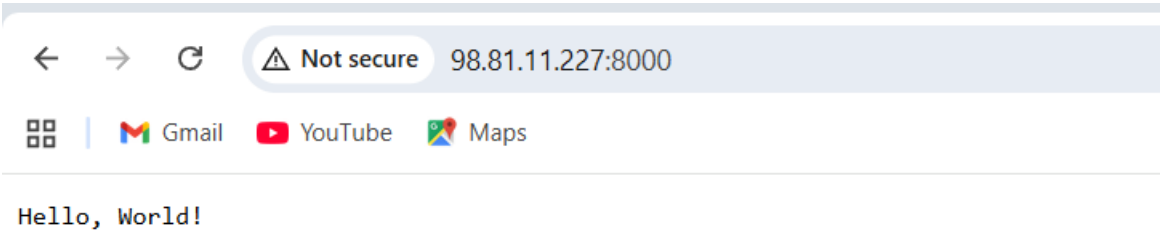
```
ubuntu@ip-172-31-30-76:~$ cd ~/cricbuzz-docker-hello-world
ubuntu@ip-172-31-30-76:~/cricbuzz-docker-hello-world$ sudo docker-compose down
[+] Running 2/2
 ✓ Container cricbuzz-app                      Removed
 ✓ Network cricbuzz-docker-hello-world_default  Removed
ubuntu@ip-172-31-30-76:~/cricbuzz-docker-hello-world$ sudo docker-compose up --build -d
no such service: -d
ubuntu@ip-172-31-30-76:~/cricbuzz-docker-hello-world$ sudo docker-compose up --build -d
WARN[0000] Docker Compose is configured to build using Bake, but buildx isn't installed
[+] Building 0.4s (12/12) FINISHED
 => [cricbuzz-app internal] load build definition from Dockerfile
 => => transferring dockerfile: 249B
 => [cricbuzz-app internal] load metadata for ghcr.io/vincetse/scratch:latest
 => [cricbuzz-app internal] load metadata for docker.io/library/golang:1.22-alpine
 => [cricbuzz-app internal] load .dockerignore
 => => transferring context: 2B
 => [cricbuzz-app builder 1/4] FROM docker.io/library/golang:1.22-alpine@sha256:1699c10032ca2582ec89
 => [cricbuzz-app internal] load build context
 => => transferring context: 2.49kB
 => CACHED [cricbuzz-app builder 2/4] WORKDIR /app
 => CACHED [cricbuzz-app builder 3/4] COPY . .
 => CACHED [cricbuzz-app builder 4/4] RUN go mod tidy && go build -o hello_world
 => CACHED [cricbuzz-app stage-1 1/2] COPY --from=builder /app/hello_world .
 => [cricbuzz-app] exporting to image
 => => exporting layers
 => => writing image sha256:e09ee1824041d9b5f828478f941ba8b558264d25bc2dca3b68a5b16b40a0d1e8
```

```
 => CACHED [cricbuzz-app builder 4/4] RUN go mod tidy && go build -o hello_world
 => CACHED [cricbuzz-app stage-1 1/2] COPY --from=builder /app/hello_world .
 => [cricbuzz-app] exporting to image
 => => exporting layers
 => => writing image sha256:e09ee1824041d9b5f828478f941ba8b558264d25bc2dca3b68a5b16b40a0d1e8
 => => naming to docker.io/library/cricbuzz-docker-hello-world-cricbuzz-app
 => [cricbuzz-app] resolving provenance for metadata file
[+] Running 3/3
 ✓ cricbuzz-app                                Built
 ✓ Network cricbuzz-docker-hello-world_default  Created
 ✓ Container cricbuzz-app                      Started
ubuntu@ip-172-31-30-76:~/cricbuzz-docker-hello-world$
```

i-0cc78891521001d1d (ubuntu-server-task)

PublicIPs: 98.81.11.227 PrivateIPs: 172.31.30.76

Search in browser with public ip and port number 8000



4. Make the web server point to the above service so that 8080 returns the response from the Docker service.

Add this script in main.tf

```
provider "aws" {
region = "us-east-1"
}
resource "aws_security_group" "web_sg" {
name = "webserver-sg"
description = "Allow SSH and port 8080"
vpc_id = "vpc-06cf45eaab13624fe"
ingress {
from_port = 22
to_port = 22
protocol = "tcp"
```

```
cidr_blocks = ["0.0.0.0/0"]
}
ingress {
from_port = 8080
to_port = 8080
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
resource "aws_instance" "ubuntu_server" {
ami = "ami-0ecb62995f68bb549"
instance_type = "t3.micro"
subnet_id = "subnet-0a192382de0e2bf6a"
associate_public_ip_address = true
vpc_security_group_ids = [aws_security_group.web_sg.id]
```

```
user_data = <<-USERDATA
#!/bin/bash
set -e
apt update -y
# Install Docker
apt install -y docker.io
systemctl enable docker
systemctl start docker
apt install -y git
# Install Docker Compose v2
curl -SL
"https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \
-o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
cd /home/ubuntu
git clone https://github.com/mujaheed00/cricbuzz-docker-hello-world.git
cd cricbuzz-docker-hello-world
chown -R ubuntu:ubuntu /home/ubuntu/cricbuzz-docker-hello-world
# Create docker-compose.yml
```

```
cat <<'COMPOSE' > docker-compose.yml
services:
cricbuzz-app:
build: .
container_name: cricbuzz-app
ports:
- "8000:8080"
restart: always
COMPOSE
/usr/local/bin/docker-compose up --build -d
apt install -y nginx
rm -f /etc/nginx/sites-enabled/default
cat <<'NGINXCONF' > /etc/nginx/sites-available/reverse-proxy
server {
listen 8080;
location / {
proxy_pass http://localhost:8000;
proxy_set_header Host \$host;
proxy_set_header X-Real-IP \$remote_addr;
}
```

}

NGINXCONF

ln -s /etc/nginx/sites-available/reverse-proxy /etc/nginx/sites-enabled/reverse-proxy

systemctl restart nginx

USERDATA

tags = {

Name = "ubuntu-server-task"

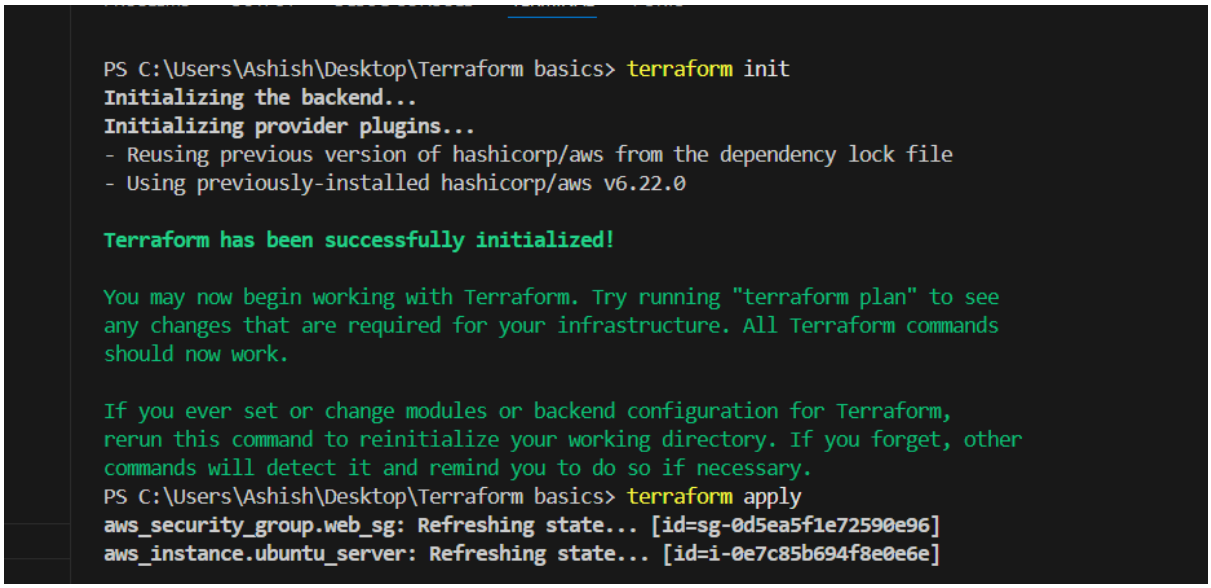
}

}

output "public_ip" {

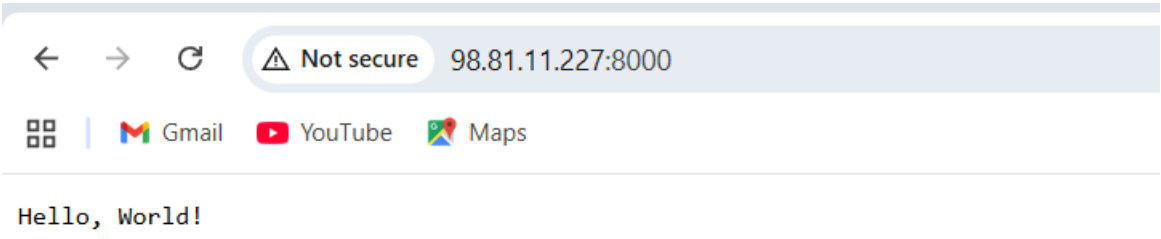
value = aws_instance.ubuntu_server.public_ip

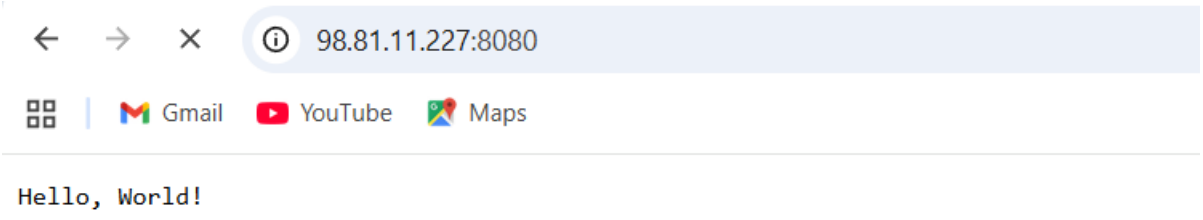
}



```
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.ubuntu_server: Modifying... [id=i-0e7c85b694f8e0e6e]
aws_instance.ubuntu_server: Still modifying... [id=i-0e7c85b694f8e0e6e, 00m10s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-0e7c85b694f8e0e6e, 00m20s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-0e7c85b694f8e0e6e, 00m30s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-0e7c85b694f8e0e6e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-0e7c85b694f8e0e6e, 00m50s elapsed]
aws_instance.ubuntu_server: Modifications complete after 1m0s [id=i-0e7c85b694f8e0e6e]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

If you type port number it will be redirected to docker webserver.





5. Configure to Save the Logs in the location /var/log/cb.log

This was the script in main.tf

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_security_group" "web_sg" {
  name        = "webserver-sg"
  description = "Allow SSH and port 8000"
  vpc_id      = "vpc-06cf45eaab13624fe"

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

  ingress {
    from_port   = 8000
    to_port     = 8000
```

```
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "ubuntu_server" {
  ami                         = "ami-0ecb62995f68bb549"
  instance_type               = "t3.micro"
  subnet_id                   = "subnet-0a192382de0e2bf6a"
  associate_public_ip_address = true
  vpc_security_group_ids      =
[aws_security_group.web_sg.id]

  user_data = <<-USERDATA
```

```
#!/bin/bash
set -e

apt update -y
apt install -y docker.io git
systemctl enable docker
systemctl start docker

curl -SL "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \
  -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose

cd /home/ubuntu
git clone https://github.com/mujaheed00/cricbuzz-docker-hello-world.git
cd cricbuzz-docker-hello-world
chown -R ubuntu:ubuntu /home/ubuntu/cricbuzz-docker-hello-world

cat << 'EOF' > docker-compose.yml
```

```
services:
  cricbuzz-app:
    build: .
    container_name: cricbuzz-app
    ports:
      - "8000:8080"
    restart: always
EOF

# Start the Docker container
/usr/local/bin/docker-compose up --build -d

################################################
# SAVE DOCKER LOGS → /var/log/cb.log
################################################
mkdir -p /var/log
touch /var/log/cb.log
chmod 666 /var/log/cb.log

cat << 'SERVICE' > /etc/systemd/system/cb-log.service
[Unit]
```

Description=Cricbuzz Docker Log Saver

After=docker.service

[Service]

Type=simple

ExecStart=/bin/bash -c "/usr/bin/docker logs -f cricbuzz-app >> /var/log/cb.log 2>&1"

Restart=always

[Install]

WantedBy=multi-user.target

SERVICE

systemctl daemon-reload

systemctl enable cb-log.service

systemctl start cb-log.service

USERDATA

```
  tags = {
    Name = "ubuntu-server-task"
```

```
  }
}
```

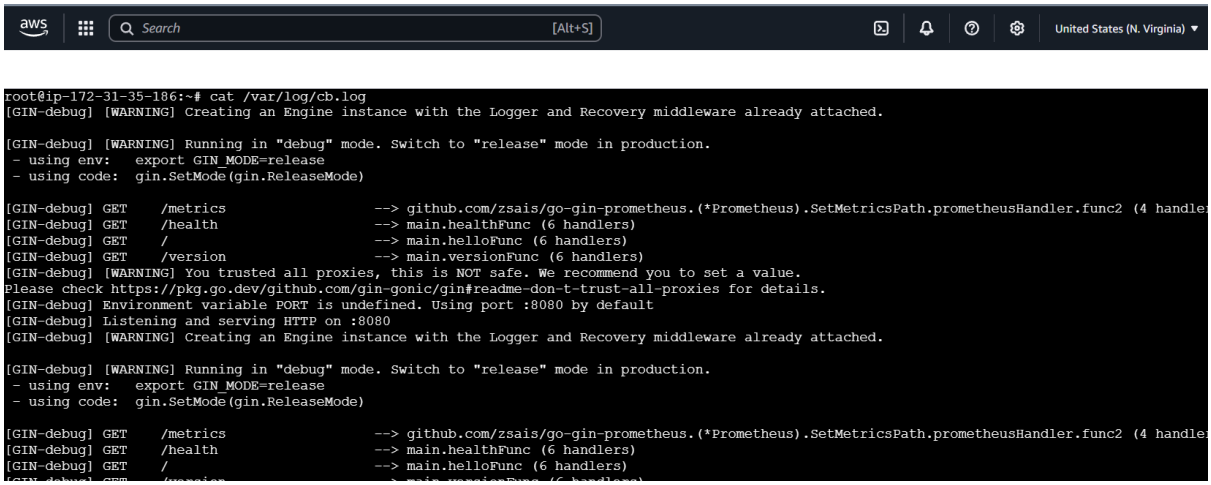
```
output "public_ip" {
  value = aws_instance.ubuntu_server.public_ip
}
```

```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_security_group.web_sg: Refreshing state... [id=sg-0d5ea5f1e72590e96]
aws_instance.ubuntu_server: Refreshing state... [id=i-07f3c7730bb24226e]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicat
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.ubuntu_server will be updated in-place
  ~ resource "aws_instance" "ubuntu_server" {
        id                                   = "i-07f3c7730bb24226e"
      ~ public_dns                           = "ec2-3-94-6-224.compute-1.amazonaws.com" -> (known after appl
      ~ public_ip                            = "3.94.6.224" -> (known after apply)
        tags                                 = {
            "Name" = "ubuntu-server-task"
        }
      ~ user_data                            = <<-EOT
            #!/bin/bash
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

  Enter a value: yes

aws_instance.ubuntu_server: Modifying... [id=i-07f3c7730bb24226e]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m10s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m20s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m30s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m50s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 01m00s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 01m10s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 01m20s elapsed]
aws_instance.ubuntu_server: Modifications complete after 1m29s [id=i-07f3c7730bb24226e]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:

public_ip = "98.85.224.222"
PS C:\Users\Ashish\Desktop\Terraform basics>
```

Login into ec2

- **/var/log/cb.log**



6. Write the Log rotate configuration for the above file

Give this in your user data

 # LOGROTATE CONFIGURATION

##

cat << 'EOF' > /etc/logrotate.d/cb-log

/var/log/cb.log {

daily

rotate 7

compress

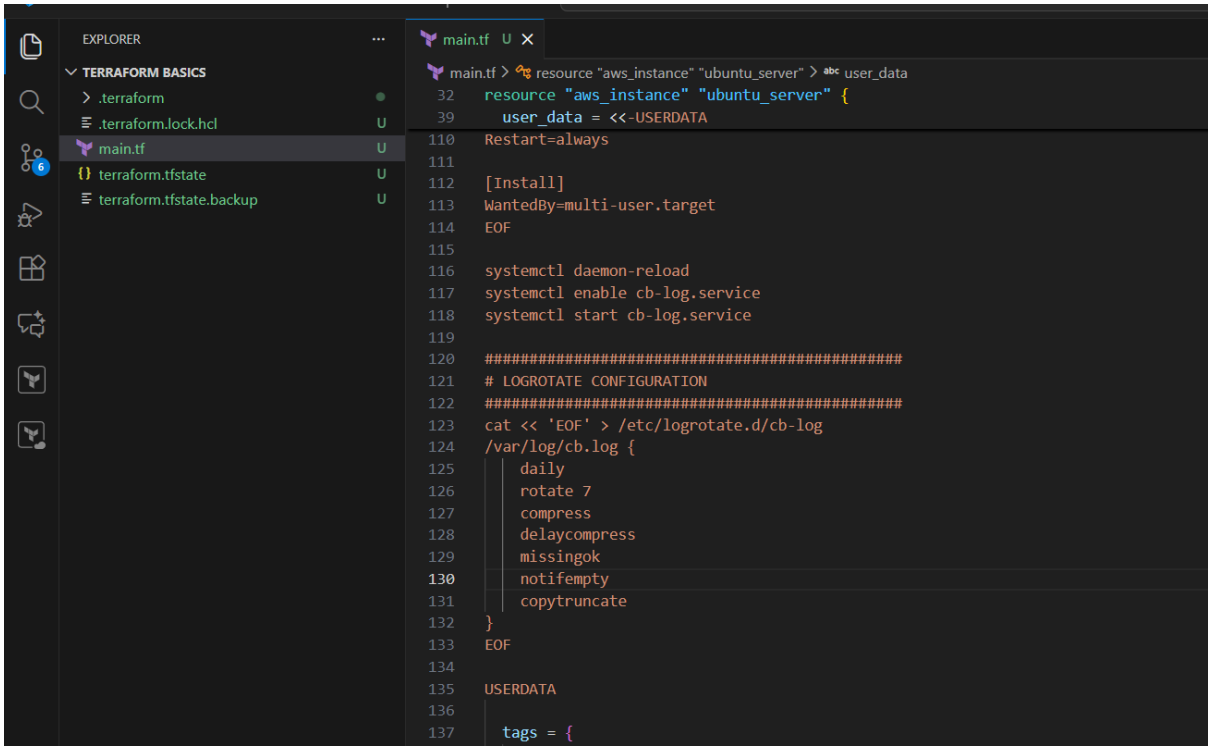
delaycompress

missingok

notifempty

copytruncate

}




```
PS C:\Users\Ashish\Desktop\Terraform basics> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.22.0

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.
PS C:\Users\Ashish\Desktop\Terraform basics> terraform apply
aws_security_group.web_sg: Refreshing state... [id=sg-0d5ea5f1e72590e96]
aws_instance.ubuntu_server: Refreshing state... [id=i-07f3c7730bb24226e]

Terraform used the selected providers to generate the following execution plan. Resource acti
  ~ update in-place
```

```
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m20s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m30s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m30s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m40s elapsed]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m50s elapsed]
aws_instance.ubuntu_server: Modifications complete after 59s [id=i-07f3c7730bb24226e]
aws_instance.ubuntu_server: Still modifying... [id=i-07f3c7730bb24226e, 00m50s elapsed]
aws_instance.ubuntu_server: Modifications complete after 59s [id=i-07f3c7730bb24226e]
aws_instance.ubuntu_server: Modifications complete after 59s [id=i-07f3c7730bb24226e]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Outputs:

public_ip = "13.217.87.105"
PS C:\Users\Ashish\Desktop\Terraform basics>
```

login to ssh and

- ls -lh /var/log/cb.log

```
root@ip-172-31-35-186:~# ls -lh /var/log/cb.log*
-rw-r--r-- 1 root root 5.9K Nov 21 15:02 /var/log/cb.log
root@ip-172-31-35-186:~#
```