Name : Reena Qureshi

Reg No : 2023-BSE-052

Section: V-B

LAB 12

Task 1 — Organize Terraform code into separate files

Create the initial project structure:

```
mkdir -p ~/Lab12
```

```
@reenaqureshi /workspaces/cc_ReenQureshi_052 (main) $ mkdir -p ~/Lab12
@reenaqureshi /workspaces/cc_ReenQureshi_052 (main) $ cd ~/Lab12
@reenaqureshi ~/Lab12 $
```

Create all required files:

touch main.tf variables.tf outputs.tf locals.tf terraform.tfvars entry-script.sh

Create variables.tf with the following content

```
@reenaqureshi ~/Lab12 $ cat variables.tf
variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "instance_type" {}
variable "public_key" {}
variable "private_key" {}
@reenaqureshi ~/Lab12 $
```

Create `outputs.tf` with the following content:

```
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```

```
tputs.tf@reenaqureshi ~/Lab12 $ cat outputs.tf
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```

Create `locals.tf` with the following content:

```
@reenaqureshi ~/Lab12 $ cat locals.tf
locals {
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}

data "http" "my_ip" {
  url = "https://icanhazip.com"
}
```

Create `terraform.tfvars` with the following content:

```
@reenaqureshi ~/Lab12 $ cat terraform.tfvars
vpc_cidr_block = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "us-east-1a"
env_prefix = "dev"
instance_type = "t3.micro"
public_key = "~/.ssh/id_ed25519.pub"
private_key = "~/.ssh/id_ed25519"
```

Create `main.tf` with the following content:

```
> EOF
@reenaqureshi ~/Lab12 $ cat main.tf
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block
  tags = {
     Name = "${var.env_prefix}-vpc"
  }
}

resource "aws_subnet" "myapp_subnet_1" {
  vpc_id     = aws_vpc.myapp_vpc.id
  cidr_block = var.subnet_cidr_block
  availability_zone = var.availability_zone
  tags = {
     Name = "${var.env_prefix}-subnet-1"
  }
}

resource "aws_default_route_table" "main_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

  route {
    cidr_block = "0.0.0.0/0"
```

Create `entry-script.sh` with the following content:

```
@reenaqureshi ~/Lab12 $ cat entry-script.sh
#!/bin/bash
set -e
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
@reenaqureshi ~/Lab12 $ chmod +x entry-script.sh
@reenaqureshi ~/Lab12 $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519
~/.ssh/@reenaqureshi ~/Lab12 $ ls -la ~/.ssh/
```

Initialize Terraform:

```
@reenaqureshi 🖹 ~/Lab12 $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/http...
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (signed by HashiCorp)
- Installing hashicorp/http v3.5.0...
- Installed hashicorp/http v3.5.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

Apply the configuration:

```
@reenaqureshi 🖹 ~/Lab12 $ @reenaqureshi 🖹 ~/Lab12 $ terraform apply -auto-approve
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 1s [id=https://icanhazip.com]
aws_key_pair.ssh-key: Refreshing state... [id=serverkey]
aws_vpc.myapp_vpc: Refreshing state... [id=vpc-037a2b5f96140533f]
aws_subnet.myapp_subnet_1: Refreshing state... [id=subnet-01a103908f7624a14]
aws_internet_gateway.myapp_igw: Refreshing state... [id=igw-0820c6d8fc5e74bf0]
aws_default_security_group.default_sg: Refreshing state... [id=sg-0e28cdb94003274fe]
aws_default_route_table.main_rt: Refreshing state... [id=rtb-028ce09473ca6d6c3]

Terraform used the selected providers to generate the following execution plan. Resource acti
following symbols:
  + create
```

Display the output:

```
ws_instance_public_ip = "3.231.50.188"
```

Test nginx in browser:

Open browser and navigate to `http://<public-ip>`

Destroy:

```
      - vpc_id                      = "vpc-0a9005120b91ce98c" -> null
        # (1 unchanged attribute hidden)
    }

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

Changes to Outputs:
  - aws_instance_public_ip = "52.3.233.109" -> null
  - ssh_command            = "ssh -i ~/.ssh/id_rsa_lab12 ubuntu@52.3.233.109" ->

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.myapp-server: Destroying... [id=i-0372abda970304c73]
aws_instance.myapp-server: Still destroying... [id=i-0372abda970304c73, 00m10s el
aws_instance.myapp-server: Still destroying... [id=i-0372abda970304c73, 00m20s el
aws_instance.myapp-server: Still destroying... [id=i-0372abda970304c73, 00m30s el
aws_instance.myapp-server: Still destroying... [id=i-0372abda970304c73, 00m40s el
aws_instance.myapp-server: Still destroying... [id=i-0372abda970304c73, 00m50s el
aws_instance.myapp-server: Destruction complete after 52s
aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-lab12]
aws_security_group.web_sg: Destroying... [id=sg-0db2d26bea296f02f]
aws_key_pair.ssh-key: Destruction complete after 1s
aws_security_group.web_sg: Destruction complete after 2s

Destroy complete! Resources: 3 destroyed.
```

Task 2 — Use remote-exec provisioner

Modify the `aws_instance` resource in `main.tf` to use `remote-exec` provisioner:

Apply the configuration:

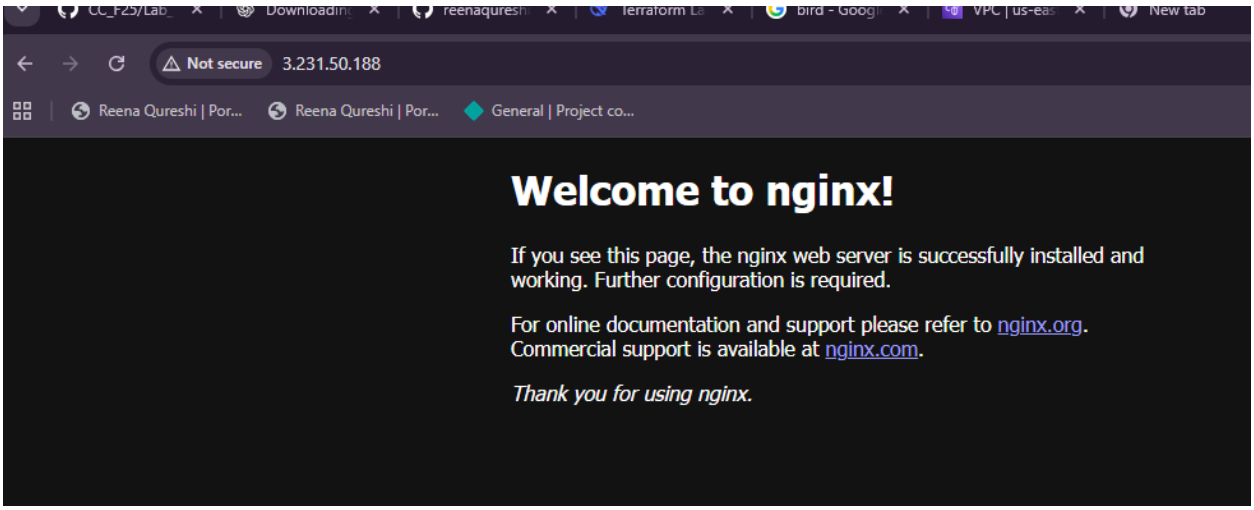
```
30
curl http://$IP@reenaqureshi ~/Lab12 $ terraform apply -auto-approve

data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
data.aws_vpc.default: Reading...
aws_key_pair.ssh-key: Refreshing state... [id=serverkey]
data.aws_vpc.default: Read complete after 2s [id=vpc-0a9005120b91ce98c]
data.aws_subnet.default: Reading...
aws_security_group.default_sg: Refreshing state... [id=sg-010af2dd39e1db8db]
data.aws_subnet.default: Read complete after 0s [id=subnet-00447af4668f7212a]
aws_instance.myapp-server: Refreshing state... [id=i-06a42a3d0b7f34cf1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.myapp-server will be updated in-place
  ~ resource "aws_instance" "myapp-server" {
        id                                   = "i-06a42a3d0b7f34cf1"
      ~ public_dns                           = "ec2-54-236-40-82.compute-1.amazonaws.com" -> (known after apply)
      ~ public_ip                            = "54.236.40.82" -> (known after apply)
        tags                                 = {
            "Name" = "dev-ec2-instance"
        }
      - user_data                            = <<-EOT
            #!/bin/bash
            set -e
```

Display the output:

`terraform output`

```
aws_instance_public_ip = "3.231.50.188"
```

Test nginx in browser:



Task 3 — Use file and local-exec provisioners

Modify the aws_instance resource in main.tf to include all three provisioners

```
sg"
  description = "Defa@reenaqureshi ~/Lab12 $ cat > main.tf << 'EOF'
> provider "aws" {
>   region = "us-east-1"
> }
>
> # Use default VPC
> data "aws_vpc" "default" {
>   default = true
> }
>
> # Use default subnet
> data "aws_subnet" "default" {
>   vpc_id            = data.aws_vpc.default.id
>   availability_zone = var.availability_zone
>   default_for_az    = true
> }
>
> # Security Group
> resource "aws_security_group" "default_sg" {
>   name        = "${var.env_prefix}-default-sg"
>   description = "Default security group for lab"
>   vpc_id      = data.aws_vpc.default.id
>
>   ingress {
>     from_port   = 22
>     to_port     = 22
>     protocol    = "tcp"
```

Apply the configuration:

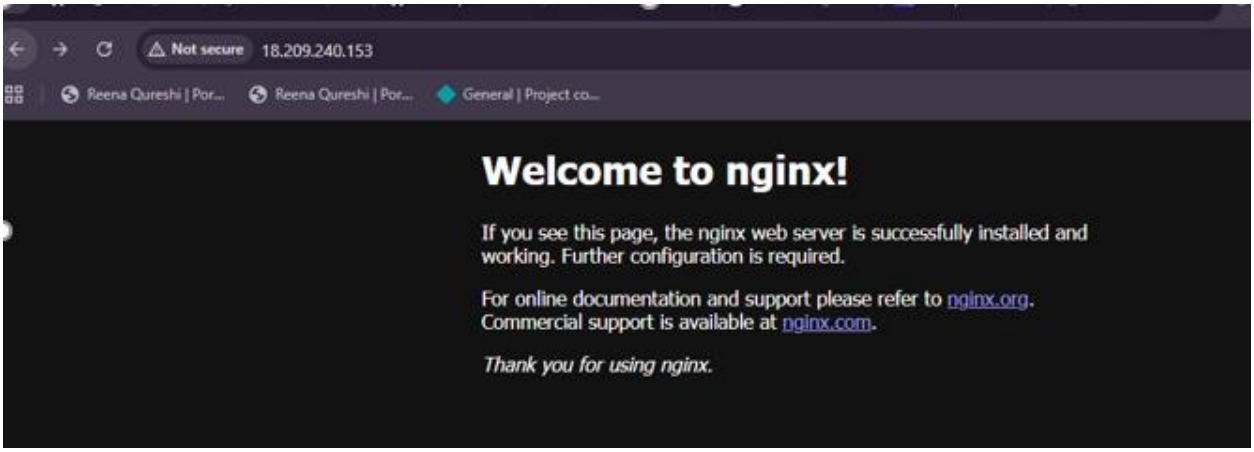
```
ws_instance.myapp-server (remote-exec): Created symlink from /etc/systemd/system/multi-use
o /usr/lib/systemd/system/nginx.service.
ws_instance.myapp-server (remote-exec): Nginx installation complete from uploaded script!
ws_instance.myapp-server (remote-exec): ● nginx.service - The nginx HTTP and reverse proxy
ws_instance.myapp-server (remote-exec):    Loaded: loaded (/usr/lib/systemd/system/nginx.s
: disabled)
ws_instance.myapp-server (remote-exec):    Active: active (running) since Sun 2026-01-04 1
ws_instance.myapp-server (remote-exec):  Main PID: 5699 (nginx)
ws_instance.myapp-server (remote-exec):    CGroup: /system.slice/nginx.service
ws_instance.myapp-server (remote-exec):            ├─5699 nginx: master proce...
ws_instance.myapp-server (remote-exec):            ├─5701 nginx: worker proce...
ws_instance.myapp-server (remote-exec):            └─5702 nginx: worker proce...

ws_instance.myapp-server (remote-exec): Jan 04 10:35:28 ip-172-31-2-214.ec2.internal syste
ws_instance.myapp-server (remote-exec): Jan 04 10:35:28 ip-172-31-2-214.ec2.internal nginx
ws_instance.myapp-server (remote-exec): Jan 04 10:35:28 ip-172-31-2-214.ec2.internal nginx
ws_instance.myapp-server (remote-exec): Jan 04 10:35:28 ip-172-31-2-214.ec2.internal syste
ws_instance.myapp-server (remote-exec): Hint: Some lines were ellipsized, use -l to show i
ws_instance.myapp-server: Provisioning with 'local-exec'...
ws_instance.myapp-server (local-exec): Executing: ["/bin/sh" "-c" "echo \"Instance i-064d1
.209.240.153 has been created at $(date)\" >> instance_creation.log\necho \"SSH command: s
user@18.209.240.153\" >> instance_creation.log\n"]
ws_instance.myapp-server: Creation complete after 58s [id=i-064d1a62869946103]

pply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

Display the output:

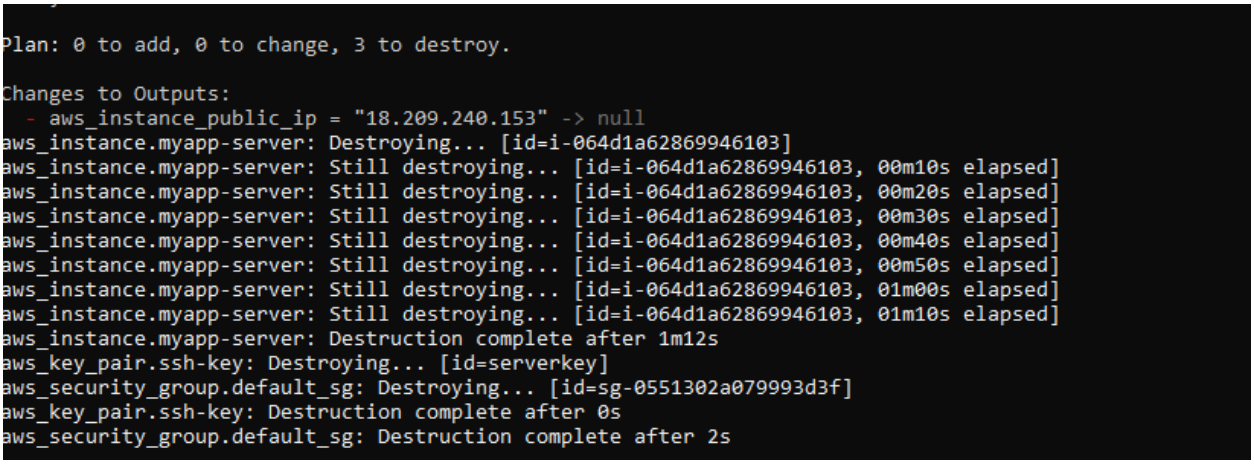
`terraform output`

```
aws_instance_public_ip = "18.209.240.153"
```

Test nginx in browser:



Destroy the resources:



Task 4 — Create Terraform modules (subnet module)

In this task, you will create a reusable subnet module to organize your infrastructure code better.

Create the module directory structure:

```
mkdir -p modules/subnet

touch modules/subnet/main.tf

touch modules/subnet/variables.tf

touch modules/subnet/outputs.tf
```

```
@reenaqureshi ~/Lab12 $ tree
.
├── entry-script.sh
├── instance_creation.log
├── locals.tf
├── main.tf
├── main.tf.task1
├── modules
│   └── subnet
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── screenshots
│   ├── task3_main_tf_all_provisioners.txt
│   ├── task3_main_tf_restored.txt
│   ├── task3_nginx_browser.txt
│   ├── task3_terraform_apply.txt
│   ├── task3_terraform_destroy.txt
│   └── task3_terraform_output.txt
├── terraform.tfstate
├── terraform.tfstate.backup
├── terraform.tfvars
├── test_url.txt
└── variables.tf

4 directories, 20 files
6. Create modules/subnet/main.tf :
```

Create `modules/subnet/variables.tf`:

```
variable "vpc_id" {}

variable "subnet_cidr_block" {}

variable "availability_zone" {}

variable "env_prefix" {}

variable "default_route_table_id" {}
```

Create modules/subnet/outputs.tf

```
in.tf@reenaqureshi ~/Lab12 $ cat modules/subnet/main.tf
data "aws_vpc" "default" {
  default = true
}

data "aws_subnet" "default" {
  vpc_id            = data.aws_vpc.default.id
  availability_zone = var.availability_zone
  default_for_az    = true
}
```

Initialize Terraform to download the module:

terraform init

```
@reenaqureshi ~/Lab12 $ terraform init
Initializing the backend...
Initializing modules...
- myapp-subnet in modules/subnet
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0
- Using previously-installed hashicorp/http v3.5.0

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.
```

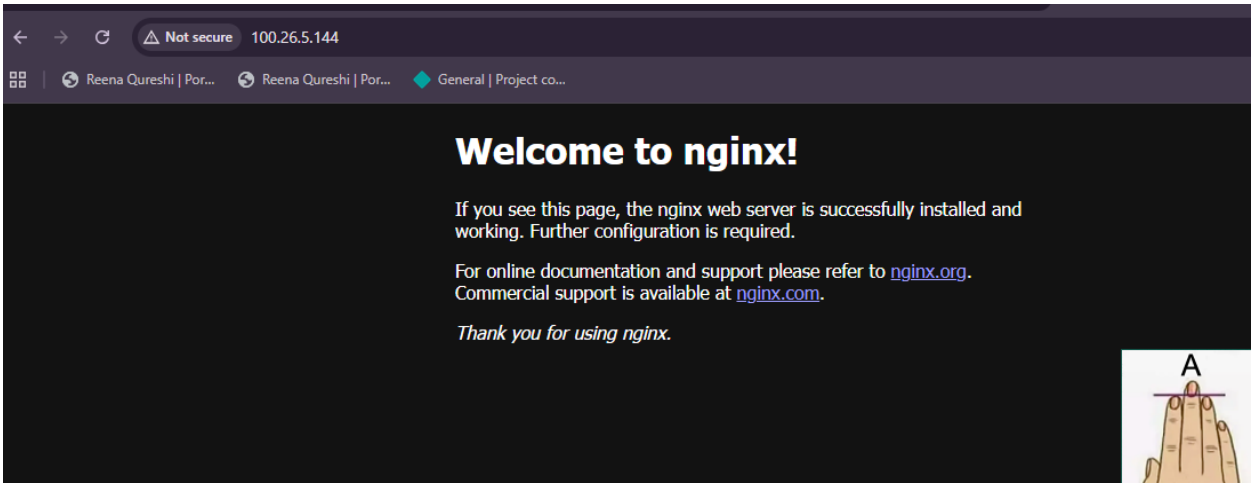
Configure it:

```
      + vpc_id                      = "vpc-0a9005120b91ce98c"
    }

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

Changes to Outputs:
  + aws_instance_public_ip = (known after apply)
aws_key_pair.ssh-key: Creating...
aws_security_group.default_sg: Creating...
aws_key_pair.ssh-key: Creation complete after 1s [id=serverkey]
aws_security_group.default_sg: Creation complete after 5s [id=sg-0d784d473193227ab]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 15s [id=i-035202f3bcf507518]

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

Outputs:

aws_instance_public_ip = "100.26.5.144"
```

OutPut:

```
aws_instance_public_ip = "100.26.5.144"
@reenaqureshi ~/Lab12 $ client_loop: send disconnect: Connection abor
shell closed: exit status 255
```

BROWSER:



Task 5 — Create webserver module

In this task, you will create a reusable webserver module for EC2 instances.

Create the webserver module directory structure:

```
mkdir -p modules/webserver
```

```
touch modules/webserver/main.tf
```

```
touch modules/webserver/variables.tf
```

```
touch modules/webserver/outputs.tf
```

Modify the root `main.tf`:

Remove the security group, ke

```
@reenaqureshi ~/Lab12 $ # Update root main.tf to use webserver module
@reenaqureshi ~/Lab12 $ cat > main.tf << 'EOF'
> provider "aws" {
>   region = "us-east-1"
> }
>
> module "myapp-subnet" {
>   source = "./modules/subnet"
>   availability_zone = var.availability_zone
>   env_prefix = var.env_prefix
> }
>
> module "myapp-webserver" {
>   source = "./modules/webserver"
>   env_prefix = var.env_prefix
>   instance_type = var.instance_type
>   availability_zone = var.availability_zone
>   public_key = var.public_key
>   my_ip = local.my_ip
>   vpc_id = module.myapp-subnet.vpc_id
>   subnet_id = module.myapp-subnet.subnet.id
>   script_path = "./entry-script.sh"
>   instance_suffix = "0"
> }
> EOF
```

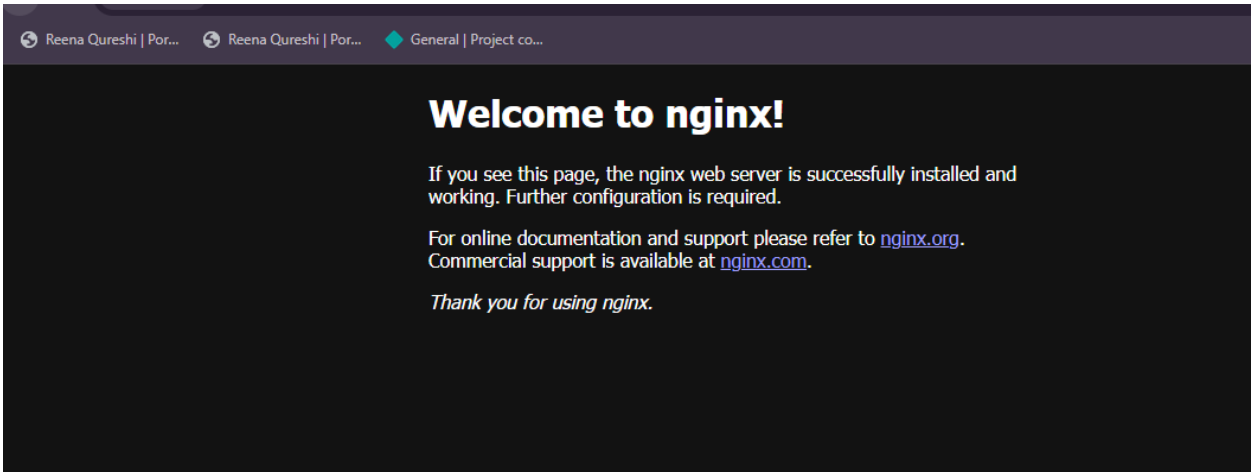
Configuring:

```
      + owner_id              = (known after apply)
      + region                = "us-east-1"
      + revoke_rules_on_delete = false
      + tags                  = {
          + "Name" = "dev-web-sg-0"
        }
      + tags_all              = {
          + "Name" = "dev-web-sg-0"
        }
      + vpc_id                = "vpc-0a9005120b91ce98c"
    }

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

Changes to Outputs:
  + instance_id        = (known after apply)
  + webserver_public_ip = (known after apply)
module.myapp-webserver.aws_key_pair.ssh-key: Creating...
module.myapp-webserver.aws_security_group.web_sg: Creating...
module.myapp-webserver.aws_key_pair.ssh-key: Creation complete after 1s [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Creation complete after 5s [id=sg-03cf0ef7bb487c907]
module.myapp-webserver.aws_instance.myapp-server: Creating...
^[[6~^[[A^[[Amodule.myapp-webserver.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 15s [id=i-0de4cf3723d982a6f]

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

Outputs:

instance_id = "i-0de4cf3723d982a6f"
```

OutPut:

```
@reenaqureshi ~/Lab12 $ cat screenshots/task5_n
Task 5 Instance IP: 13.219.245.239
Testing Nginx with webserver module...
```

Browser test:



Destroying Resources:

```
  Enter a value: yes

module.myapp-webserver.aws_instance.myapp-server: Destroying... [id=i-0de4cf3723d982a6f]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de4cf3723d982a6f, 00m1
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de4cf3723d982a6f, 00m2
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de4cf3723d982a6f, 00m3
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de4cf3723d982a6f, 00m4
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de4cf3723d982a6f, 00m5
module.myapp-webserver.aws_instance.myapp-server: Destruction complete after 52s
module.myapp-webserver.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Destroying... [id=sg-03cf0ef7bb487c907]
module.myapp-webserver.aws_key_pair.ssh-key: Destruction complete after 0s
module.myapp-webserver.aws_security_group.web_sg: Destruction complete after 2s
```

Task 6 — Configure HTTPS with self-signed certificates

In this task, you will configure Nginx to serve traffic over HTTPS using self-signed certificates.

Update `entry-script.sh` with SSL configuration:

```
# Update and install required @reenaqureshi 🖹 ~/Lab12 $ cat > entry-script.sh << 'EOF'
> #!/bin/bash
> set -e
>
> # Update and install required packages
> sudo yum update -y
> sudo amazon-linux-extras enable nginx1
> sudo yum install -y nginx openssl
>
> # Create directories for SSL certificates
> sudo mkdir -p /etc/ssl/private
> sudo mkdir -p /etc/ssl/certs
>
> # Get IMDSv2 token
> TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
>   -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
>
> # Get current public IP and hostname
> PUBLIC_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
```

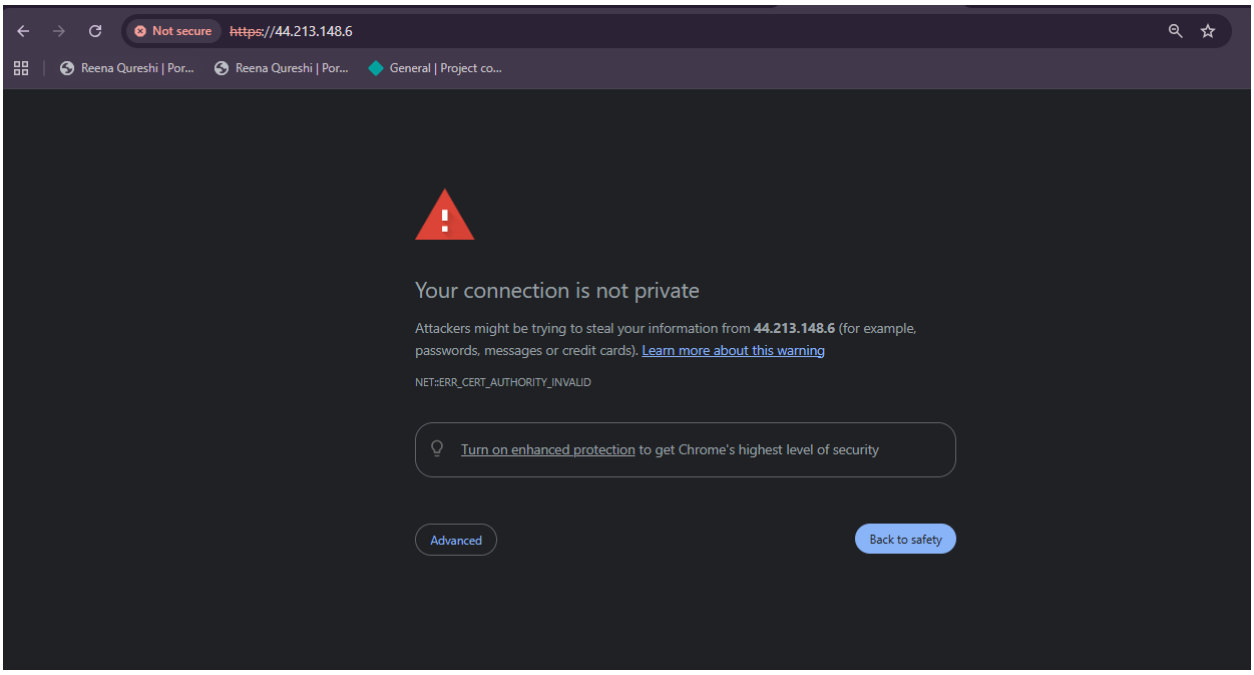
Apply the configuration:

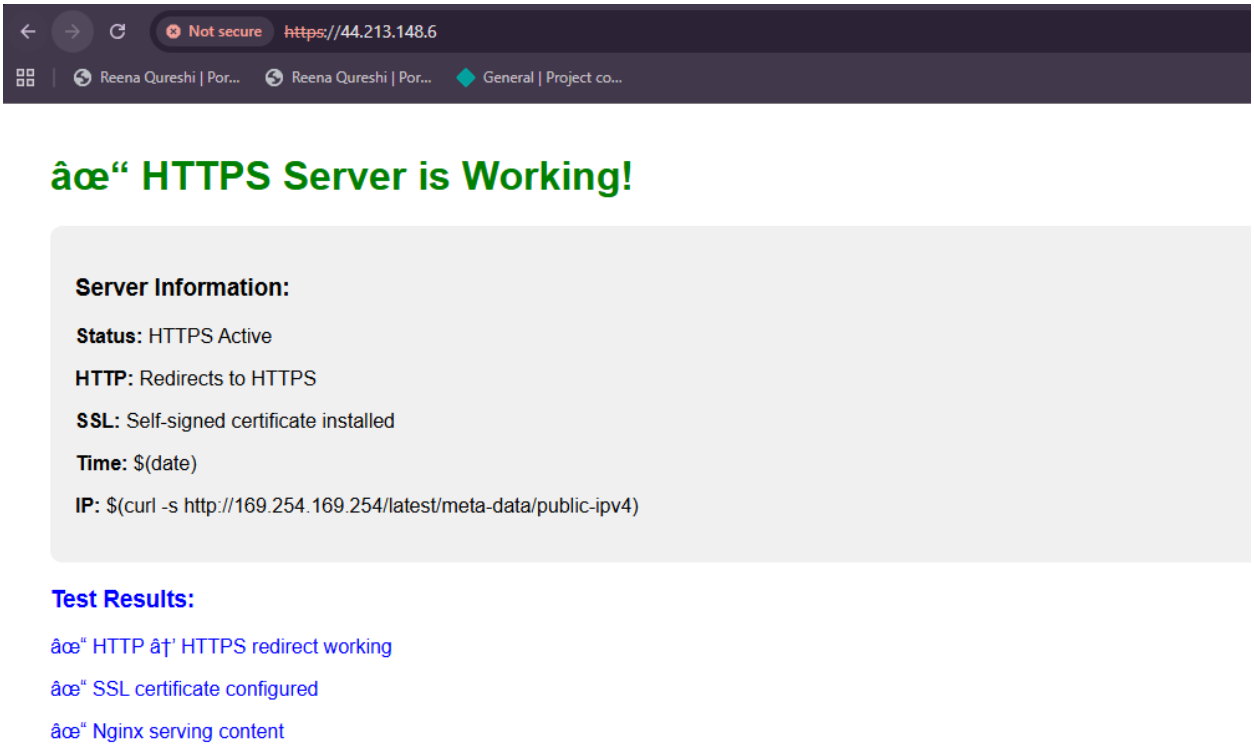
`terraform apply -auto-approve`

```
@reenaqureshi 🖹 ~/Lab12 $ terraform apply
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
module.myapp-subnet.data.aws_vpc.default: Reading...
module.myapp-webserver.aws_key_pair.ssh-key: Refreshing state... [id=dev-serverkey-0]
module.myapp-subnet.data.aws_vpc.default: Read complete after 2s [id=vpc-0a9005120b91ce98c]
module.myapp-subnet.data.aws_subnet.default: Reading...
module.myapp-webserver.aws_security_group.web_sg: Refreshing state... [id=sg-0d646d1910f202a86]
module.myapp-subnet.data.aws_subnet.default: Read complete after 1s [id=subnet-00447af4668f7212a]
module.myapp-webserver.aws_instance.myapp-server: Refreshing state... [id=i-04e12a68150b3fa35]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no c
needed.
```

Outputs:

```
instance_id = "i-04f7da1b82a4ba3cd"
webserver_public_ip = "44.213.148.6"
```



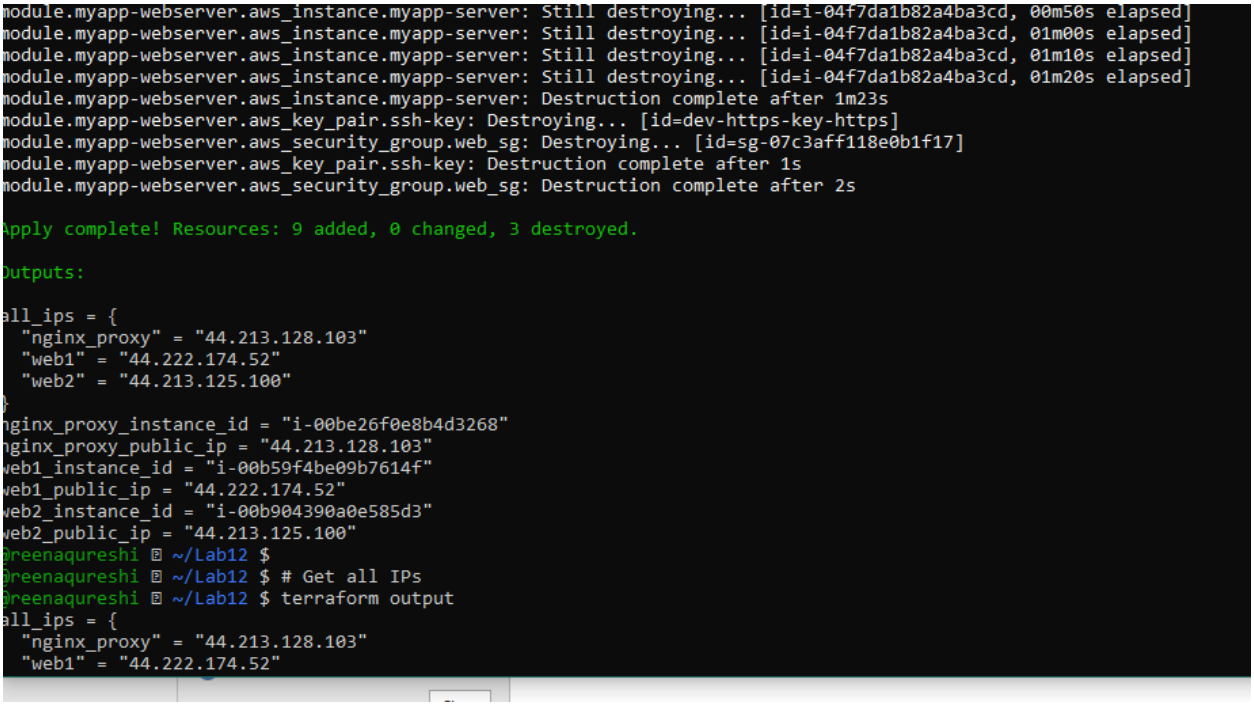


Task 7 — Configure Nginx as reverse proxy

Create `apache.sh` script for backend web server:

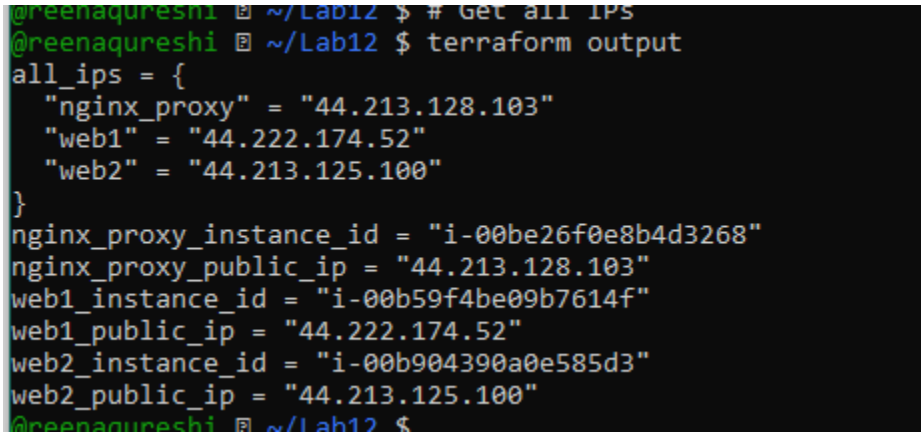
Apply the configuration:

```
terraform apply -auto-approve
```



Get the outputs:

```
terraform output
```



SSH into the webserver (Nginx proxy server)

```
@reenaqureshi ~/Lab12 $ ssh ec2-user@44.222.174.52
The authenticity of host '44.222.174.52 (44.222.174.52)' can't be established.
ED25519 key fingerprint is SHA256:ySsSyHWpyDRYYohcNQl3XoSjMwqH7T817sBwWbCxA2Y.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.222.174.52' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@backend-server-00b5 ~]$
```

Edit the Nginx configuration:

`sudo vim /etc/nginx/nginx.conf`

```
        # TASK 7: Reverse proxy to web1
        location / {
            proxy_pass http://44.213.125.100:80;  # WEB1_IP
            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;
        }
    }
}
NGINX
```

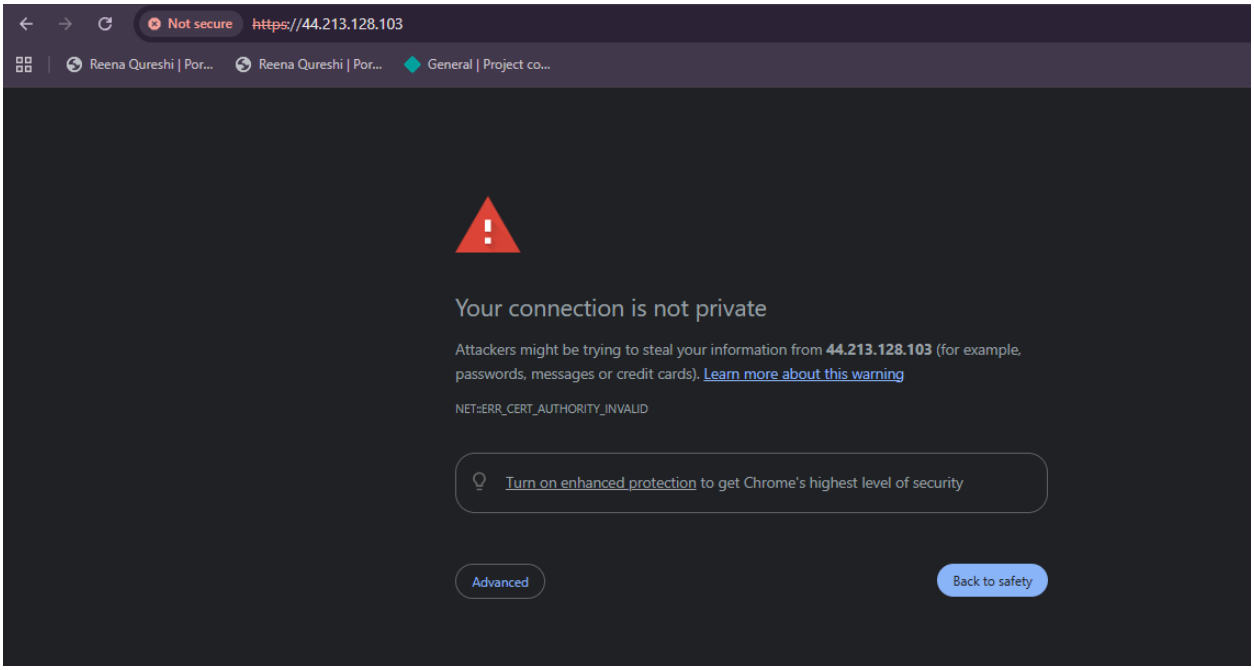
Restart Nginx:

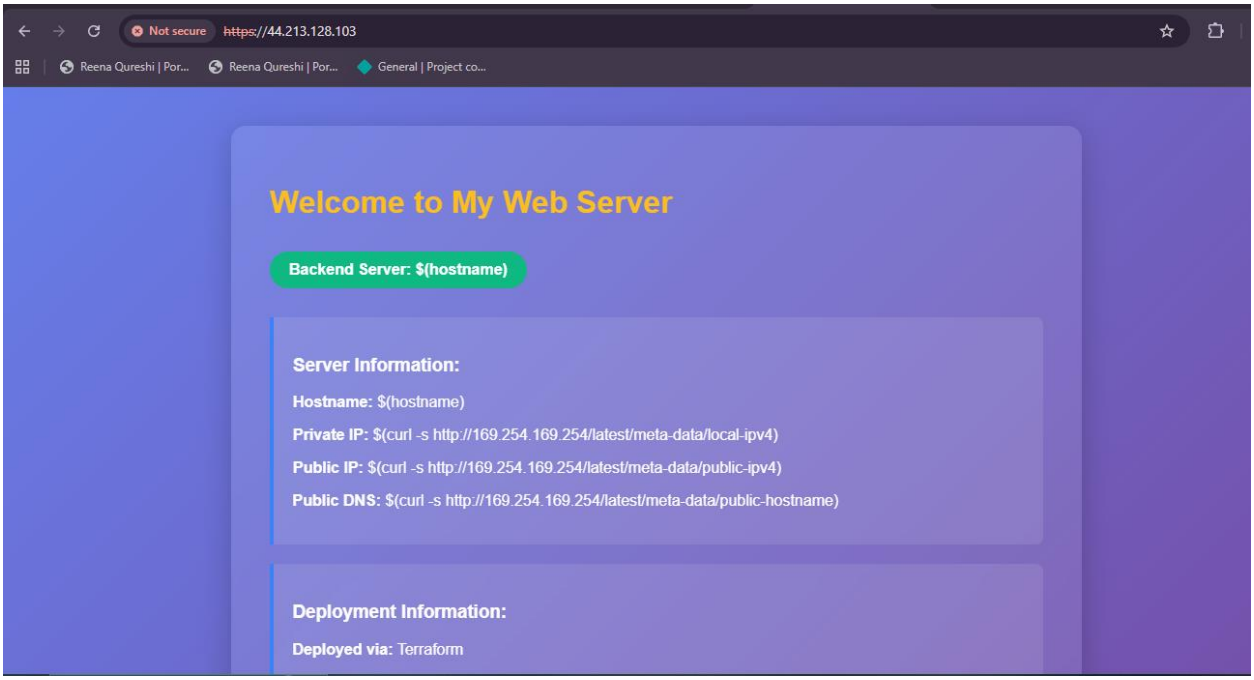
`sudo systemctl restart nginx`

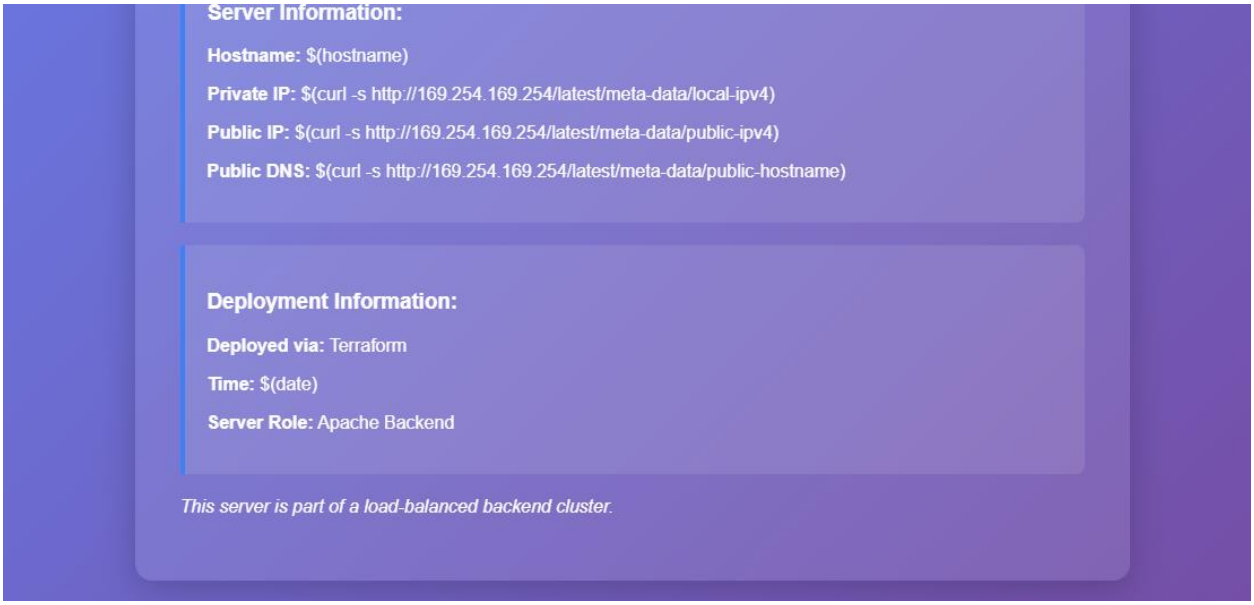
```
[ec2-user@ip-172-31-8-106 ~]$ # Restart nginx
[ec2-user@ip-172-31-8-106 ~]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-8-106 ~]$
```

View Nginx logs and configuration files:

`cat /var/log/nginx/error.log`


Not secure
https://44.213.128.103
Reena Qureshi | Por...
Reena Qureshi | Por...
General | Project co...
Your connection is not private
Attackers might be trying to steal your information from 44.213.128.103 (for example, passwords, messages or credit cards). Learn more about this warning
NET::ERR_CERT_AUTHORITY_INVALID
Turn on enhanced protection to get Chrome's highest level of security
Advanced
Back to safety


Not secure
https://44.213.128.103
Reena Qureshi | Por...
Reena Qureshi | Por...
General | Project co...
Welcome to My Web Server
Backend Server: $(hostname)
Server Information:
Hostname: $(hostname)
Private IP: $(curl -s http://169.254.169.254/latest/meta-data/local-ipv4)
Public IP: $(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)
Public DNS: $(curl -s http://169.254.169.254/latest/meta-data/public-hostname)
Deployment Information:
Deployed via: Terraform



Task 8 — Configure Nginx as load balancer

In this task, you will add a second backend server and configure Nginx

Add the second web server module to `main.tf`:

Update `outputs.tf`:

Apply the configuration:

```
module.nginx-proxy.aws_security_group.web_sg: Creation complete after 5s [id=sg-0c9d0e2df7a355f4c]
module.myapp-web-2.aws_instance.myapp-server: Creating...
module.nginx-proxy.aws_instance.myapp-server: Creating...
module.myapp-web-1.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.nginx-proxy.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still creating... [00m20s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Still creating... [00m20s elapsed]
module.nginx-proxy.aws_instance.myapp-server: Still creating... [00m20s elapsed]
module.nginx-proxy.aws_instance.myapp-server: Creation complete after 24s [id=i-0656851113f46f24c]
module.myapp-web-1.aws_instance.myapp-server: Still creating... [00m30s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Still creating... [00m30s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Creation complete after 34s [id=i-04b86cb564f01ae0a]
module.myapp-web-1.aws_instance.myapp-server: Creation complete after 34s [id=i-0a32a30c6aad1ee29]
```

Get all outputs:

```
all_servers = {
  "backend_1" = "3.233.237.119"
  "backend_2" = "13.220.9.105"
  "nginx_proxy" = "34.239.93.60"
}
nginx_proxy_public_ip = "34.239.93.60"
web1_public_ip = "3.233.237.119"
web2_public_ip = "13.220.9.105"
```

Nginx Proxy Server

Status: Ready for Tasks 7-10 Configuration

IP: 3.233.237.119

Tasks:

- Task 7: Reverse Proxy
- Task 8: Load Balancer
- Task 9: High Availability
- Task 10: Caching

Configure via: /etc/nginx/nginx.conf

Nginx Proxy Server

Status: Ready for Tasks 7-10 Configuration

IP: 13.220.9.105

Tasks:

- Task 7: Reverse Proxy
- Task 8: Load Balancer
- Task 9: High Availability
- Task 10: Caching

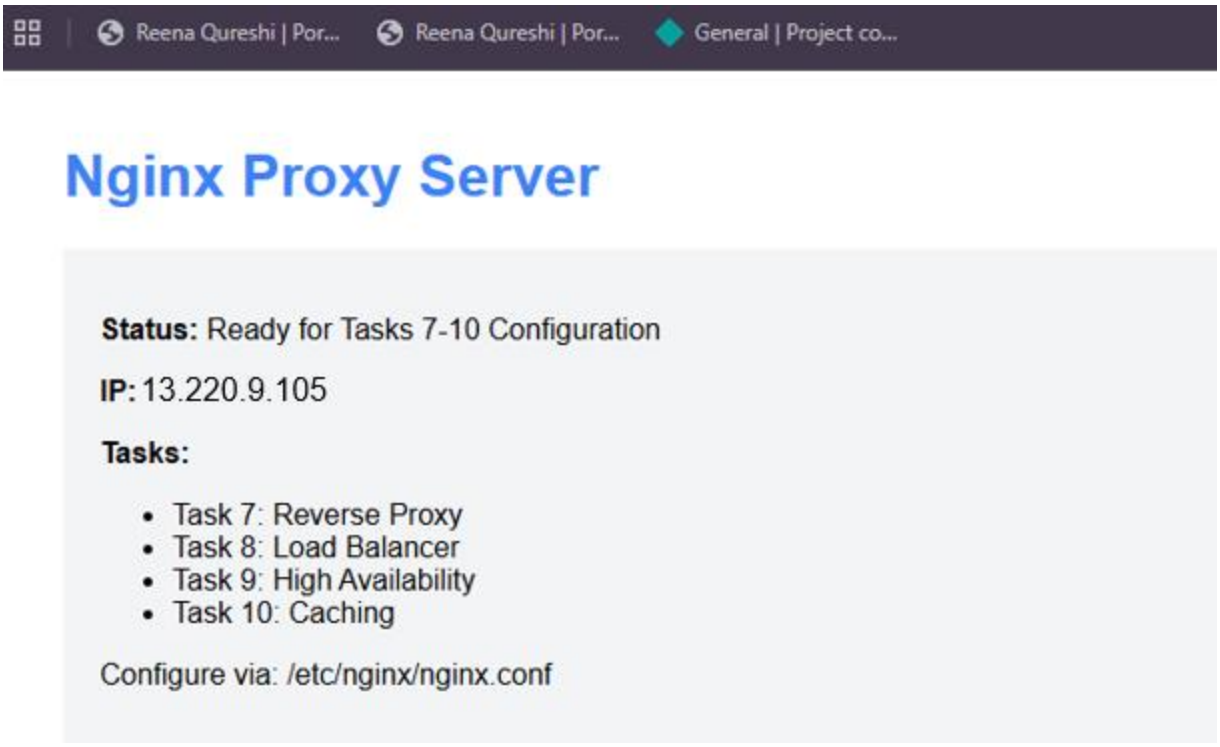
Configure via: /etc/nginx/nginx.conf

TASK 9: Configure high availability with backup servers

```
@reenaqureshi ~/Lab12 $ ssh ec2-user@34.239.93.60
The authenticity of host '34.239.93.60 (34.239.93.60)' can't be established.
ED25519 key fingerprint is SHA256:9/qHZCWWy2xTWWiHFZR1G2GfTfnhO3OqbbTwNeSavjs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.239.93.60' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-14-17 ~]$
```

```
[ec2-user@ip-172-31-14-17 ~]$ systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2026-01-04 13:48:21 UTC; 32min ago
  Process: 6419 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 6417 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 6415 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 6422 (nginx)
   CGroup: /system.slice/nginx.service
           ├─6422 nginx: master process /usr/sbin/nginx
           ├─6425 nginx: worker process
           └─6426 nginx: cache manager process

Jan 04 13:48:22 ip-172-31-14-17.ec2.internal nginx[6417]: nginx: the configuration file /etc/nginx/nginx.
Jan 04 13:48:22 ip-172-31-14-17.ec2.internal nginx[6417]: nginx: configuration file /etc/nginx/nginx.conf
Jan 04 13:48:21 ip-172-31-14-17.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server
Jan 04 13:48:21 ip-172-31-14-17.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-14-17 ~]$
```

Server running ON web 2:



Task 10 — Enable Nginx caching

1. SSH into the webserver:

```
ssh ec2-user@<webserver-public-ip>
```

```
@reenaqureshi ~/Lab12 $ ssh ec2-user@34.239.93.60
The authenticity of host '34.239.93.60 (34.239.93.60)' can't be established.
ED25519 key fingerprint is SHA256:9/qHZCWWy2xTWWiHFZR1G2GfTfnhO3OqbbTwNeSavjs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.239.93.60' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-14-17 ~]$
```