Lab 12

Lab title: Terraform Provisioners, Modules & Nginx Reverse Proxy/Load Balancer

Submitted to: Engr. Mohammad Shoaib

Submitted by: Anara Hayat

Reg#No: 2023-BSE-008

Task 0 Lab Setup (Codespace & GH CLI)

All actions below should be executed inside the Codespace shell.

Create Codespace & connect:


```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Lab12 (main)
$ gh codespace create --repo Anara-hayat/Lab12
  ✓ Codespaces usage for this repository is paid for by Anara-hayat
? Choose Machine Type: 2 cores, 8 GB RAM, 32 GB storage
animated-capybara-69xgwgprxr4wc795

Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Lab12 (main)
$ gh codespace list
NAME              DISPLAY NAME   REPOSITORY     BRANCH  STATE      CREATED AT
obscure-space...  obscure sp...  Anara-haya...  main*   Shutdown   about 26 d...
obscure-happi...  obscure ha...  Anara-haya...  main*   Shutdown   about 11 d...
orange-robot-...  orange robot   Anara-haya...  main    Shutdown   about 3 da...
animated-capy...  animated c...  Anara-haya...  main    Available  less than ...
```

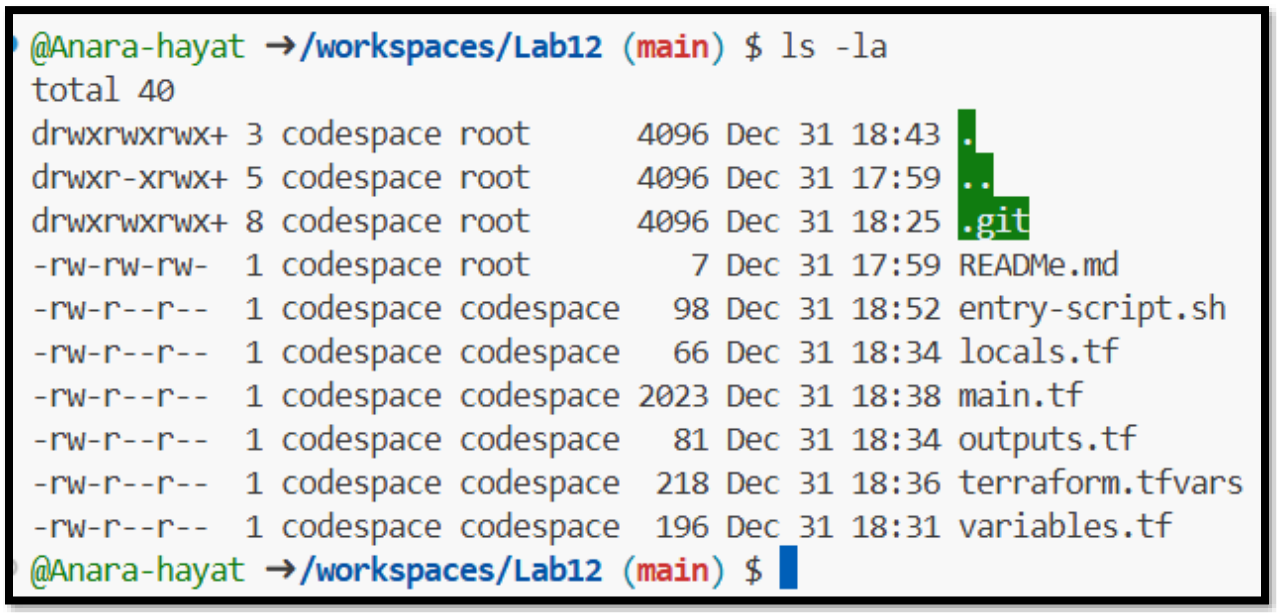
```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Lab12 (main)
$ gh codespace list --json name -q '.[].name'
obscure-space-doodle-6vv75r4rwxvf56pw
obscure-happiness-r45gwgxr54jxcp997
orange-robot-v6jvwv5gjqjrfpjgx
animated-capybara-69xgwgprxr4wc795

Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Lab12 (main)
$ gh codespace ssh -c animated-capybara-69xgwgprxr4wc795
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

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

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

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

@Anara-hayat → /workspaces/Lab12 (main) $
```

Task 1 — Organize Terraform code into separate files

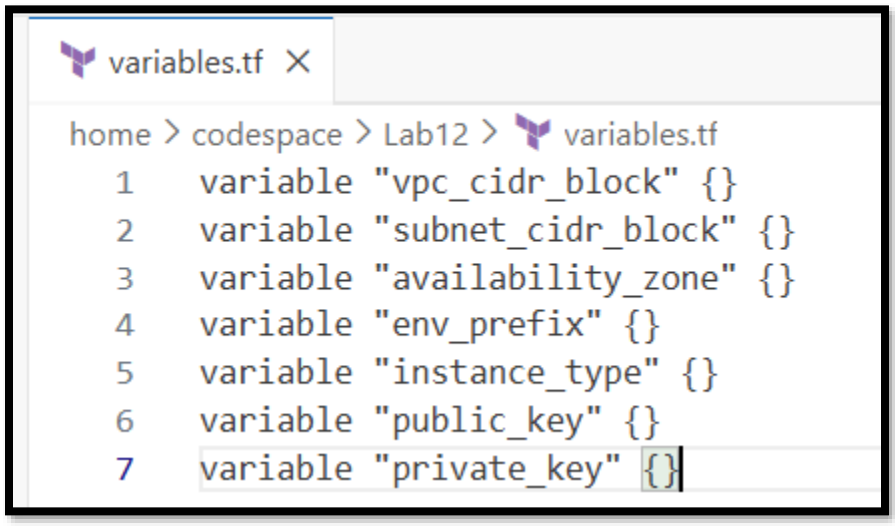
In this task, you will split a monolithic Terraform configuration into separate, well-organized files following best practices.

1. **Create the initial project structure:**

2. **Create all required files:**



3. **Create variables.tf with the following content:**



4. **Create outputs.tf with the following content:**

outputs.tf

home > codespace > Lab12 > outputs.tf

```
1  output "aws_instance_public_ip" {
2    value = aws_instance.myapp-server.public_ip
3  }
```

5. **Create locals.tf with the following content:**

locals.tf

home > codespace > Lab12 > locals.tf

```
1  locals {
2    my_ip = "${chomp(data.http.my_ip.response_body)}/32"
3  }
```

6. **Create terraform.tfvars with the following content**

terraform.tfvars

home > codespace > Lab12 > terraform.tfvars

```
1  vpc_cidr_block = "10.0.0.0/16"
2  subnet_cidr_block = "10.0.10.0/24"
3  availability_zone = "me-central-1a"
4  env_prefix = "dev"
5  instance_type = "t3.micro"
6  public_key = "~/.ssh/id_ed25519.pub"
7  private_key = "~/.ssh/id_ed25519"
```

7. **Create main.tf with the following content:**

main.tf

home > codespace > Lab12 > main.tf

```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

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
```

8. **Create entry-script.sh with the following content:**

$ entry-script.sh

```
#!/bin/bash
set -e
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
```

9. **Generate SSH key pair if not already exists:**

```
@Anara-hayat ➜/workspaces/Lab12 (main) $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""
Generating public/private ed25519 key pair.
Your identification has been saved in /home/codespace/.ssh/id_ed25519
Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:K2w+XJzOpBoThWn8inaMrbpoxLevXf2dEgvDRM3IicA codespace@codespaces-704975
The key's randomart image is:
+--[ED25519 256]--+
|    ... o =      |
|   . E . = o     |
|    = . .        |
|   . o   .       |
|.   . ..S.       |
| o * +  *= .     |
|. = O.+B..o o    |
|.o +.==.o .o. .  |
|=o..++..   ..o   |
+----[SHA256]-----+
@Anara-hayat ➜/workspaces/Lab12 (main) $
```

10. Initialize Terraform:

```
@Anara-hayat ➜/workspaces/Lab12 (main) $ 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.
@Anara-hayat ➜/workspaces/Lab12 (main) $
```

11. Apply the configuration:

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply
Changes to Outputs:
  + aws_instance_public_ip = (known after apply)

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_key_pair.ssh_key: Creating...
aws_vpc.myapp_vpc: Creating...
aws_key_pair.ssh_key: Creation complete after 0s [id=serverkey]
aws_vpc.myapp_vpc: Creation complete after 1s [id=vpc-01450a0f115f7e11e]
aws_internet_gateway.myapp_igw: Creating...
aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-0530745ac74a0e60b]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-0f7a28d243a7ea6eb]
aws_default_route_table.main_rt: Creation complete after 0s [id=rtb-0db36fa78a731fdee]
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-01943214486454290]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 12s [id=i-0931980483c578fde]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_public_ip = "3.28.219.42"
```

12. Display the output:

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "3.28.219.42"
```

13. Test nginx in browser:

14. **Destroy resources:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform destroy

Plan: 0 to add, 0 to change, 7 to destroy.

Changes to Outputs:
  - aws_instance_public_ip = "3.28.219.42" -> null

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_default_route_table.main_rt: Destroying... [id=rtb-0db36fa78a731fdee]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_instance.myapp-server: Destroying... [id=i-0931980483c578fde]
aws_internet_gateway.myapp_igw: Destroying... [id=igw-0530745ac74a0e60b]
aws_instance.myapp-server: Still destroying... [id=i-0931980483c578fde, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0530745ac74a0e60b, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 17s
aws_instance.myapp-server: Destruction complete after 20s
aws_key_pair.ssh_key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0f7a28d243a7ea6eb]
aws_default_security_group.default_sg: Destroying... [id=sg-01943214486454290]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh_key: Destruction complete after 0s
aws_subnet.myapp_subnet_1: Destruction complete after 0s
aws_vpc.myapp_vpc: Destroying... [id=vpc-01450a0f115f7e11e]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```

Task 2 — Use remote-exec provisioner

In this task, you will replace the user_data approach with the remote-exec provisioner to install and configure nginx.

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

Replace the user_data line with the following provisioner block:

main.tf

```
73  resource "aws_instance" "myapp-server" {
74    ami           = "ami-05524d6658fcf35b6"
75    instance_type = var.instance_type
76    subnet_id     = aws_subnet.myapp_subnet_1.id
77    security_groups = [aws_default_security_group.default_sg.id]
78    availability_zone = var.availability_zone
79    associate_public_ip_address = true
80    key_name = aws_key_pair.ssh-key.key_name
81
82    connection {
83      type        = "ssh"
84      user        = "ec2-user"
85      private_key = file(var.private_key)
86      host        = self.public_ip
87    }
88
89    provisioner "remote-exec" {
90      inline = [
91        "sudo yum update -y",
92        "sudo yum install -y nginx",
93        "sudo systemctl start nginx",
94        "sudo systemctl enable nginx"
95      ]
96    }
97
98    tags = {
99      Name = "${var.env_prefix}-ec2-instance"
00    }
```

2. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve
aws_instance.myapp-server (remote-exec):   Installing       : nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Running scriptlet: nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Verifying        : generic-logo   1/7
aws_instance.myapp-server (remote-exec):   Verifying        : gperftools-l   2/7
aws_instance.myapp-server (remote-exec):   Verifying        : libunwind-1.   3/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-1:1.28   4/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-core-1   5/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-filesy   6/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-mimety   7/7

aws_instance.myapp-server (remote-exec): Installed:
aws_instance.myapp-server (remote-exec):   generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
aws_instance.myapp-server (remote-exec):   gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
aws_instance.myapp-server (remote-exec):   libunwind-1.4.0-5.amzn2023.0.3.x86_64
aws_instance.myapp-server (remote-exec):   nginx-1:1.28.0-1.amzn2023.0.2.x86_64
aws_instance.myapp-server (remote-exec):   nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64
aws_instance.myapp-server (remote-exec):   nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
aws_instance.myapp-server (remote-exec):   nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

aws_instance.myapp-server (remote-exec): Complete!
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target
.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
aws_instance.myapp-server: Creation complete after 34s [id=i-0a7b8836cc140287d]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_public_ip = "3.28.226.95"
```

3. **Display the output:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "3.28.226.95"
```

4. **Test nginx in browser:**

Task 3 — Use file and local-exec provisioners

In this task, you will add the file provisioner to upload the script and the local-exec provisioner to log instance information locally.

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

main.tf

```
resource "aws_instance" "myapp-server" {
  ami           = "ami-05524d6658fcf35b6"
  instance_type = var.instance_type
  subnet_id     = aws_subnet.myapp_subnet_1.id
  security_groups = [aws_default_security_group.default_sg.id]
  availability_zone = var.availability_zone
  associate_public_ip_address = true
  key_name = aws_key_pair.ssh-key.key_name

  connection {
    type        = "ssh"
    user        = "ec2-user"
    private_key = file(var.private_key)
    host        = self.public_ip
  }

  provisioner "file" {
    source = "./entry-script.sh"
    destination = "/home/ec2-user/entry-script-on-ec2.sh"
  }

  provisioner "remote-exec" {
    inline = [
      "sudo chmod +x /home/ec2-user/entry-script-on-ec2.sh",
      "sudo /home/ec2-user/entry-script-on-ec2.sh"
    ]
  }

  provisioner "local-exec" {
    command = <<-EOF
      echo Instance ${self.id} with public IP ${self.public_ip} has been
    EOF
  }
```

Loading...

2. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve
aws_instance.myapp-server (remote-exec):   Installing       : nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Running scriptlet: nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Verifying        : generic-logo   1/7
aws_instance.myapp-server (remote-exec):   Verifying        : gperftools-l   2/7
aws_instance.myapp-server (remote-exec):   Verifying        : libunwind-1.   3/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-1:1.28   4/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-core-1   5/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-filesy   6/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-mimety   7/7

aws_instance.myapp-server (remote-exec): Installed:
aws_instance.myapp-server (remote-exec):   generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
aws_instance.myapp-server (remote-exec):   gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
aws_instance.myapp-server (remote-exec):   libunwind-1.4.0-5.amzn2023.0.3.x86_64
aws_instance.myapp-server (remote-exec):   nginx-1:1.28.0-1.amzn2023.0.2.x86_64
aws_instance.myapp-server (remote-exec):   nginx-core-1:1.28.0-1.amzn2023.0.2.x86_64
aws_instance.myapp-server (remote-exec):   nginx-filesystem-1:1.28.0-1.amzn2023.0.2.noarch
aws_instance.myapp-server (remote-exec):   nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch

aws_instance.myapp-server (remote-exec): Complete!
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target
.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
aws_instance.myapp-server: Provisioning with 'local-exec'...
aws_instance.myapp-server (local-exec): Executing: ["/bin/sh" "-c" "echo Instance i-0a97966193
b557c18 with public IP 3.28.136.220 has been created\n"]
aws_instance.myapp-server (local-exec): Instance i-0a97966193b557c18 with public IP 3.28.136.2
20 has been created
aws_instance.myapp-server: Creation complete after 59s [id=i-0a97966193b557c18]

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

Outputs:

aws_instance_public_ip = "3.28.136.220"
```

3. **Display the output:**

```
aws_instance_public_ip = "3.28.136.220"
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "3.28.136.220"
@Anara-hayat →/workspaces/Lab12 (main) $
```

4. **Test nginx in browser:**

5. **Destroy the resources:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform destroy

  Enter a value: yes

aws_default_route_table.main_rt: Destroying... [id=rtb-0655061f6ca9224cd]
aws_instance.myapp-server: Destroying... [id=i-0a97966193b557c18]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-008b204bfca58f04b]
aws_instance.myapp-server: Still destroying... [id=i-0a97966193b557c18, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-008b204bfca58f04b, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 17s
aws_instance.myapp-server: Destruction complete after 20s
aws_key_pair.ssh_key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0a6519f3700ce8a70]
aws_default_security_group.default_sg: Destroying... [id=sg-00886bb0fadb7c39a]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh_key: Destruction complete after 0s
aws_subnet.myapp_subnet_1: Destruction complete after 0s
aws_vpc.myapp_vpc: Destroying... [id=vpc-024ad56fe2bd5ab70]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```

6. **Remove the provisioners and restore user_data:**

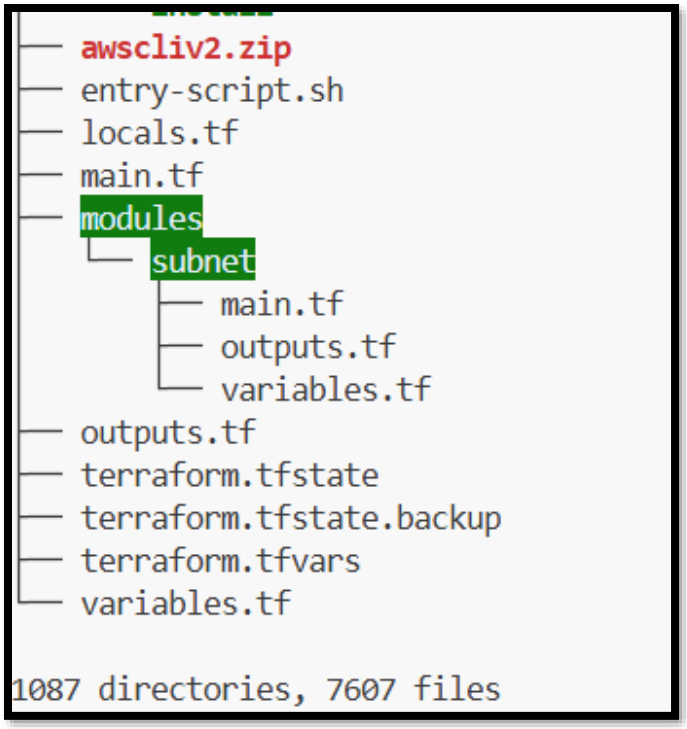
main.tf

```
resource "aws_instance" "myapp-server" {
  subnet_id     = aws_subnet.myapp_subnet_1.id
  security_groups = [aws_default_security_group.default_sg.id]
  availability_zone = var.availability_zone
  associate_public_ip_address = true
  key_name = aws_key_pair.ssh_key.key_name

  connection {
    type        = "ssh"
    user        = "ec2-user"
    private_key = file(var.private_key)
    host        = self.public_ip
  }

  user_data = file("./entry-script.sh")

  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
}

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

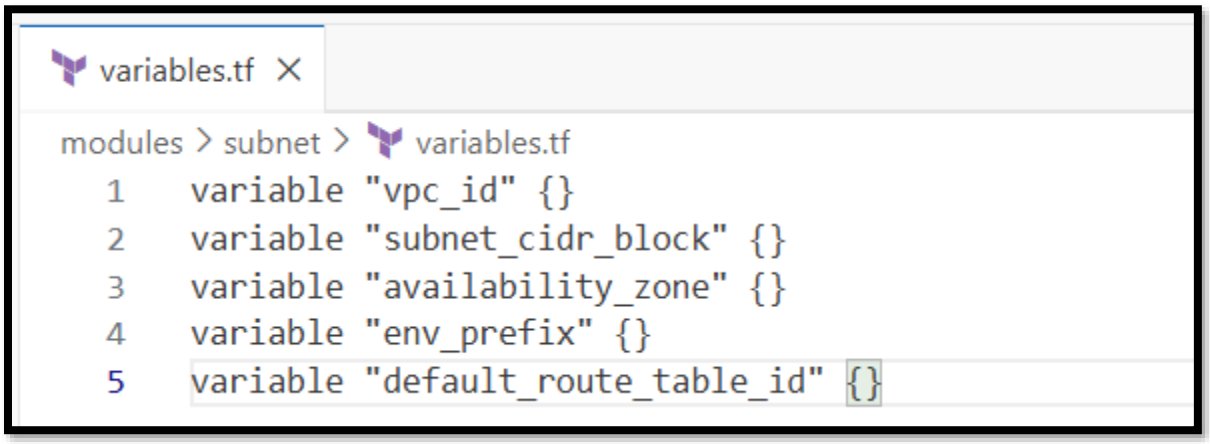
Task 4 — Create Terraform modules (subnet module)

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

1. **Create the module directory structure:**



2. **Create modules/subnet/variables.tf:**



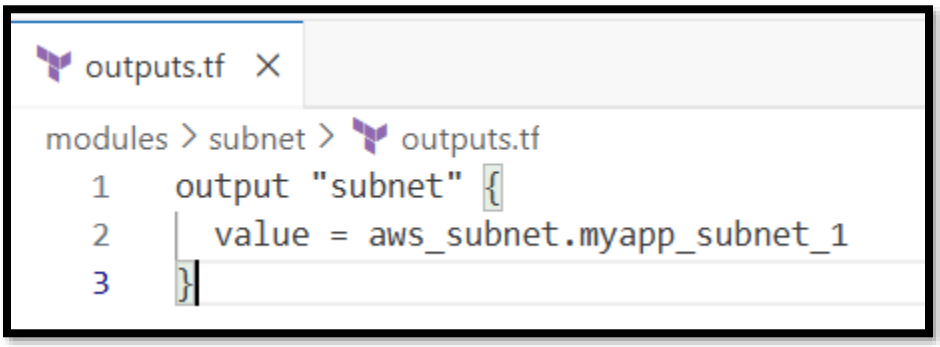
3. **Create modules/subnet/main.tf:**

main.tf

modules > subnet > main.tf

```
resource "aws_subnet" "myapp_subnet_1" {
  vpc_id     = var.vpc_id
  cidr_block = var.subnet_cidr_block
  availability_zone = var.availability_zone
  map_public_ip_on_launch = true
  tags = {
    Name = "${var.env_prefix}-subnet-1"
  }
}

resource "aws_default_route_table" "main_rt" {
  default_route_table_id = var.default_route_table_id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway. myapp_igw.id
  }
    tags = {
    Name = "${var.env_prefix}-rt"
  }
}
```

4. **Create modules/subnet/outputs.tf:**



5. **Modify the root main.tf to use the subnet module:**

Remove the subnet, route table, and internet gateway resources and replace them with:

main.tf

nodules > subnet > main.tf

```
1   module "myapp_subnet" {
2     source = "./modules/subnet"
3     vpc_id = aws_vpc.myapp_vpc. id
4     subnet_cidr_block = var.subnet_cidr_block
5     availability_zone = var.availability_zone
6     env_prefix = var.env_prefix
7     default_route_table_id = aws_vpc.myapp_vpc.default_
8   }
9
10  resource "aws_instance" "myapp-server" {
11    ami                    = "ami-0a0e5d9c7acc336f1"
12    instance_type          = var.instance_type
13    subnet_id              = module.myapp-subnet.subnet
14    vpc_security_group_ids = [aws_security_group.myapp_
15    key_name               = "serverkey"
16
17    tags = {
18      Name = "${var.env_prefix}-server"
19    }
20  }
```

6. **Initialize Terraform to download the module:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/http from the dependency lock
file
- Reusing previous version of hashicorp/aws from the dependency lock f
ile
- Using previously-installed hashicorp/http v3.5.0
- Using previously-installed hashicorp/aws v6.27.0

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 c
ommands
should now work.

If you ever set or change modules or backend configuration for Terrafo
rm.
```

7. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve
aws_vpc.myapp_vpc: Creation complete after 1s [id=vpc-08319ae2e9b79bc8c]
aws_internet_gateway.myapp_igw: Creating...
aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-084c2ac32a5e56122]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-0b96779a6453ebf37]
aws_default_route_table.main_rt: Creation complete after 1s [id=rtb-09be70aea96fb9ce1]
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-03e235901a08f6190]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-0f673ff83d88a7ef8]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

aws_instance_public_ip = "40.172.100.241"
```

8. **Display the output:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "40.172.100.241"
```

9. **Test nginx in browser:**

Task 5 — Create webserver module

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

1. **Create the webserver module directory structure:**

```
├── entry-script.sh
├── locals.tf
├── main.tf
├── modules
│   ├── subnet
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   └── webserver
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── terraform.tfstate
├── terraform.tfstate.backup
├── terraform.tfvars
└── variables.tf
```

2. **Create modules/webserver/variables.tf:**

variables.tf

modules > webserver > variables.tf

```
variable "availability_zone" {}
variable "public_key" {}
variable "my_ip" {}
variable "vpc_id" {}
variable "subnet_id" {}
variable "script_path" {}
variable "instance_suffix" {}
```

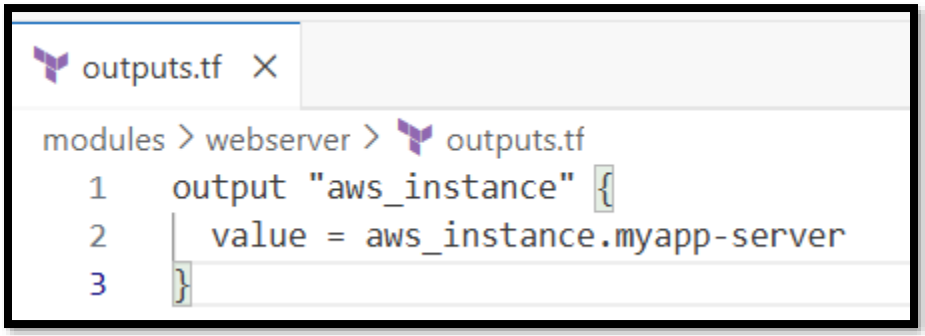
3. **Create modules/webserver/main.tf:**

main.tf U

modules > webserver > main.tf

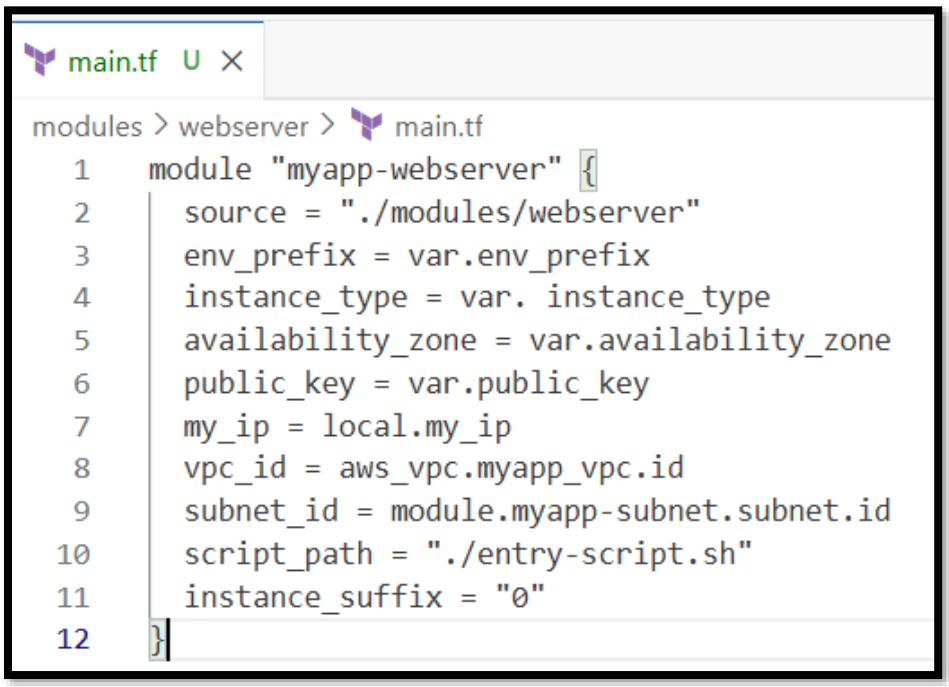
```
resource "aws_security_group" "web_sg" {
  vpc_id      = var.vpc_id
  name        = "${var.env_prefix}-web-sg-${var.instance_suffix}"
  description = "Security group for web server allowing HTTP, HTTPS and S

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
```

4. **Create modules/webserver/outputs.tf:**

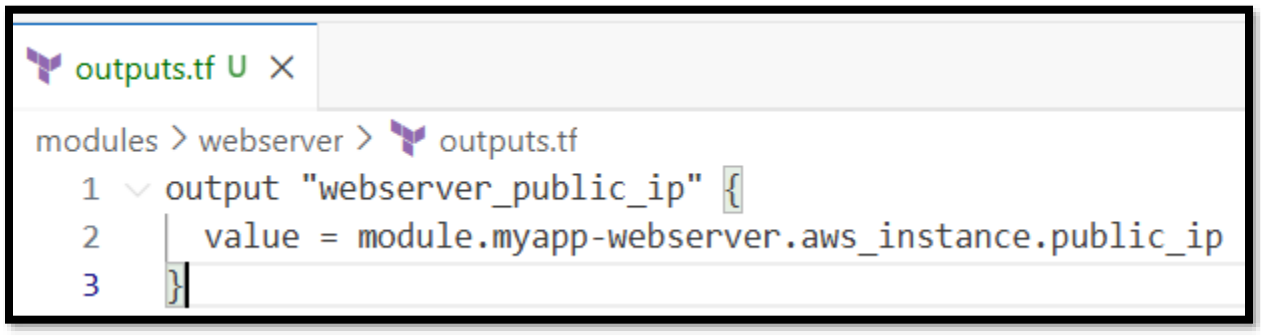


5. **Modify the root main.tf:**

Remove the security group, key pair, and instance resources. Replace them with:



6. **Update outputs.tf:**



7. **Initialize Terraform:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/http from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/http v3.5.0
- Using previously-installed hashicorp/aws v6.27.0

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.
```

8. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve

Changes to Outputs:
  ~ aws_instance_public_ip = "40.172.100.241" -> (known after apply)
aws_instance.myapp-server: Destroying... [id=i-0f673ff83d88a7ef8]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 00m10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 00m20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 00m30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 00m40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 00m50s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 01m00s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 01m10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f673ff83d88a7ef8, 01m20s elapsed]
aws_instance.myapp-server: Destruction complete after 1m20s
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-0eddd0397966b74cf]

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

Outputs:

aws_instance_public_ip = "158.252.93.41"
```

9. **Display the output:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "158.252.93.41"
```

10. Test nginx in browser:

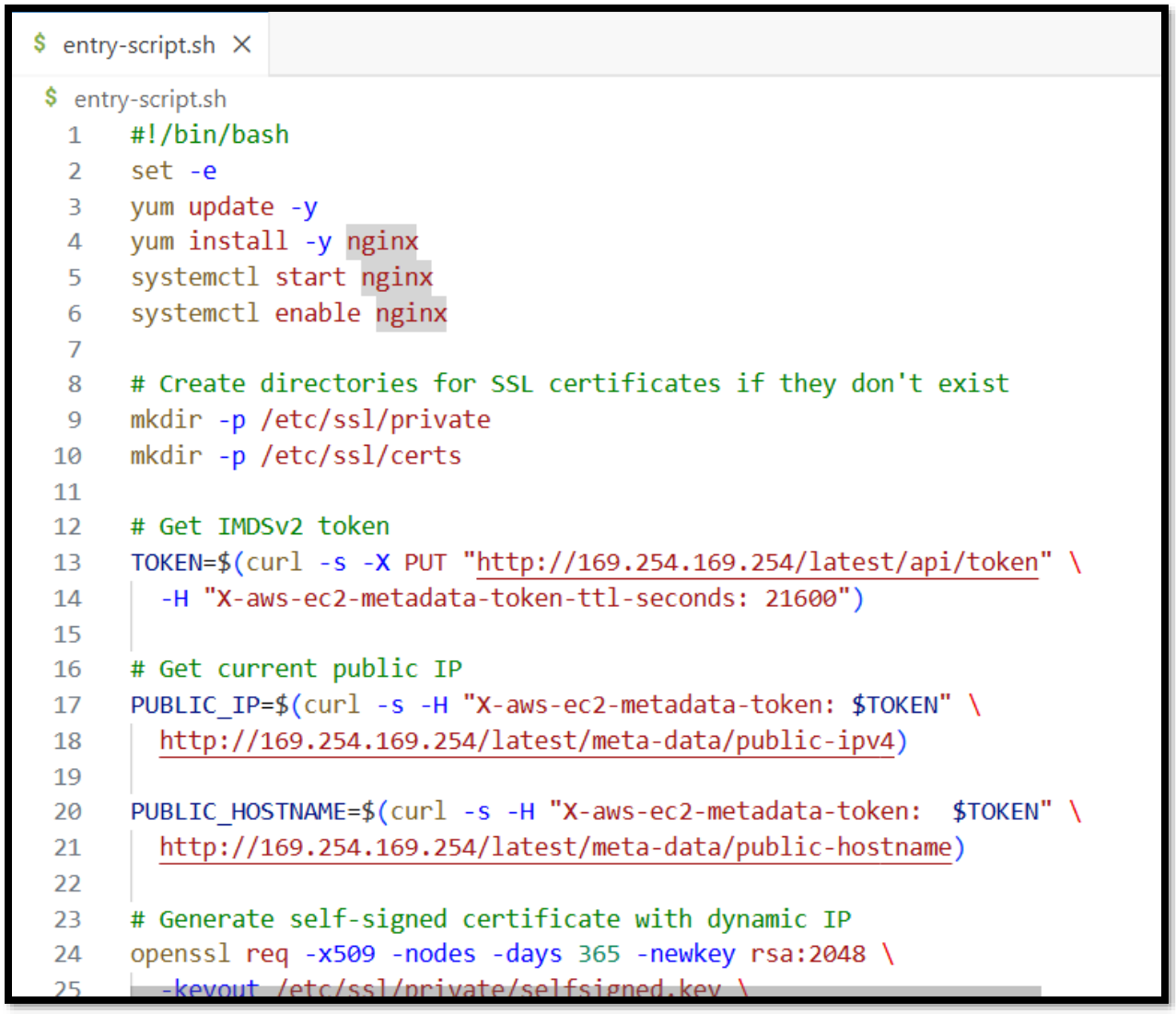
11. Destroy resources:

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform destroy

aws_default_route_table.main_rt: Destroying... [id=rtb-09be70aea96fb9ce1]
aws_instance.myapp-server: Destroying... [id=i-0eddd0397966b74cf]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-084c2ac32a5e56122]
aws_instance.myapp-server: Still destroying... [id=i-0eddd0397966b74cf, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-084c2ac32a5e56122, 00m10s elapsed
aws_instance.myapp-server: Still destroying... [id=i-0eddd0397966b74cf, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-084c2ac32a5e56122, 00m20s elapsed
aws_internet_gateway.myapp_igw: Destruction complete after 29s
aws_instance.myapp-server: Still destroying... [id=i-0eddd0397966b74cf, 00m30s elapsed]
aws_instance.myapp-server: Destruction complete after 31s
aws_key_pair.ssh_key: Destroying... [id=serverkey]
aws_default_security_group.default_sg: Destroying... [id=sg-03e235901a08f6190]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0b96779a6453ebf37]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh_key: Destruction complete after 0s
aws_subnet.myapp_subnet_1: Destruction complete after 0s
aws_vpc.myapp_vpc: Destroying... [id=vpc-08319ae2e9b79bc8c]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
```

Task 6 — Configure HTTPS with self-signed certificates

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

1. **Update entry-script.sh with SSL configuration:**



2. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve
Changes to Outputs:
  + aws_instance_public_ip = (known after apply)
aws_key_pair.ssh_key: Creating...
aws_vpc.myapp_vpc: Creating...
aws_key_pair.ssh_key: Creation complete after 1s [id=serverkey]
aws_vpc.myapp_vpc: Creation complete after 2s [id=vpc-0b8820211b0a292d3]
aws_internet_gateway.myapp_igw: Creating...
aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-0b47a6259398f2a58]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-0cee13893c010dee1]
aws_default_route_table.main_rt: Creation complete after 2s [id=rtb-02bd4fe212be0245b]
aws_default_security_group.default_sg: Creation complete after 4s [id=sg-00aa516e47dad1105]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-0c866567629133228]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
```

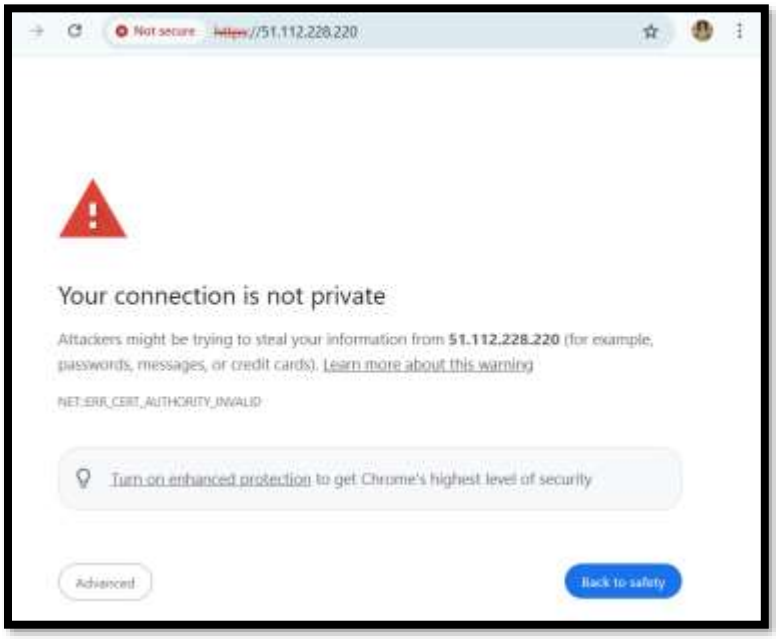
3. **Display the output:**

```
Last login: Thu Jan  1 10:32:32 2026 from ::1
@Anara-hayat → /workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "51.112.228.220"
@Anara-hayat → /workspaces/Lab12 (main) $
```

4. **Test HTTPS in browser:**

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

You will see a warning: "Warning: Potential Security Risk Ahead"



- Click "Advanced" button
- Click "Accept the Risk and Continue"

5. **Verify HTTP to HTTPS redirect:**

Task 7 — Configure Nginx as reverse proxy

In this task, you will create a backend web server and configure Nginx to act as a reverse proxy.

1. **Create apache.sh script for backend web server:**

```
@Anara-hayat → /workspaces/Lab12 (main) $ touch apache.sh
@Anara-hayat → /workspaces/Lab12 (main) $ code apache.sh
```

apache.sh

```
#!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd
echo "<h1>Welcome to My Web Server</h1>" > /var/www/html/index.html
hostnamectl set-hostname myapp-webserver
echo "<h2>Hostname:  $(hostname)</h2>" >> /var/www/html/index.html
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
echo "<h2>Private IP: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN"  ht
echo "<h2>Public IP: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN"  htt
echo "<h2>Public DNS: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN"  ht
echo "<h2>Deployed via Terraform</h2>" >> /var/www/html/index. html
```

PROBLEMS OUTPUT TERMINAL PORTS bash

```
@Anara-hayat →/workspaces/Lab12 (main) $ code apache.sh
```

2. **Add the backend web server module to main.tf:**

main.tf

```
93
94   module "myapp-web-1" {
95     source = "./modules/webserver"
96     env_prefix = var.env_prefix
97     instance_type = var.instance_type
98     availability_zone = var.availability_zone
99     public_key = var. public_key
00     my_ip = local.my_ip
01     vpc_id = aws_vpc. myapp_vpc.id
02     subnet_id = module.myapp-subnet.subnet. id
03     script_path = "./apache.sh"
04     instance_suffix = "1"
05   }
```

3. **Update outputs.tf:**

```
outputs.tf
1   output "webserver_public_ip"{
2     value=module.myapp-webserver.aws_instance.public_ip
3   }
4   output "aws_web-1_public_ip" {
5     value = module.myapp-web-1.aws_instance.public_ip
6   }
```

4. **Apply the configuration:**

```
@Anara-hayat ➜/workspaces/Lab12 (main) $ terraform apply -auto-approv
e
aws_instance.myapp-server: Still destroying... [id=i-0580068b0e01fe664
, 00m40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0580068b0e01fe664
, 00m50s elapsed]
aws_instance.myapp-server: Destruction complete after 50s
aws_default_security_group.default_sg: Destroying... [id=sg-00aa516e47
dad1105]
aws_default_security_group.default_sg: Destruction complete after 0s

Apply complete! Resources: 6 added, 0 changed, 2 destroyed.

Outputs:

aws_web_1_public_ip = "3.28.215.197"
webserver_public_ip = "3.28.163.111"
```

5. **Get the outputs:**

```
@Anara-hayat ➜/workspaces/Lab12 (main) $ terraform output
aws_web_1_public_ip = "3.28.215.197"
webserver_public_ip = "3.28.163.111"
```

6. **SSH into the webserver (Nginx proxy server):**

```
@Anara-hayat ➜/workspaces/Lab12 (main) $ ssh ec2-user@3.28.163.111
The authenticity of host '3.28.163.111 (3.28.163.111)' can't be establ
ished.
ED25519 key fingerprint is SHA256:vDcno0eYKRNA6mwpjoFP3mr4WpI+z5FQkgm2
k8mcw44.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
es
Warning: Permanently added '3.28.163.111' (ED25519) to the list of kno
wn hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-250 ~]$
```

7. **Edit the Nginx configuration:**

```
    server {
        listen 443 ssl;
        server_name 3.28.163.111;

        ssl_certificate     /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;

        location / {
            #root /usr/share/nginx/html;
            #index index.html;
                proxy_pass http://3.28.215.197:80;
                #proxy_pass http://3.28.163.111;
        }
    }
```

8. **Restart Nginx:**

```
[ec2-user@ip-10-0-10-250 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-250 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; >
     Active: active (running) since Fri 2026-01-02 11:00:43 UTC; 1min>
    Process: 25556 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=e>
    Process: 25557 ExecStartPre=/usr/sbin/nginx -t (code=exited, stat>
    Process: 25558 ExecStart=/usr/sbin/nginx (code=exited, status=0/S>
   Main PID: 25559 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.3M
        CPU: 45ms
     CGroup: /system.slice/nginx.service
             ├─25559 "nginx: master process /usr/sbin/nginx"
             ├─25560 "nginx: worker process"
             └─25561 "nginx: worker process"

Jan 02 11:00:43 ip-10-0-10-250.me-central-1.compute.internal systemd[>
Jan 02 11:00:43 ip-10-0-10-250.me-central-1.compute.internal nginx[25>
```

9. **View Nginx logs and configuration files:**

```
[ec2-user@ip-10-0-10-250 ~]$ cat /var/log/nginx/error.log
2026/01/02 10:51:35 [notice] 3203#3203: using the "epoll" event method
2026/01/02 10:51:35 [notice] 3203#3203: nginx/1.28.0
2026/01/02 10:51:35 [notice] 3203#3203: OS: Linux 6.1.158-180.294.amzn
2023.x86_64
2026/01/02 10:51:35 [notice] 3203#3203: getrlimit(RLIMIT_NOFILE): 6553
5:65535
2026/01/02 10:51:35 [notice] 3232#3232: start worker processes
2026/01/02 10:51:35 [notice] 3232#3232: start worker process 3233
2026/01/02 10:51:35 [notice] 3232#3232: start worker process 3234
2026/01/02 10:51:35 [warn] 3309#3309: could not build optimal types_ha
sh, you should increase either types_hash_max_size: 1024 or types_hash
_bucket_size: 64; ignoring types_hash_bucket_size
2026/01/02 10:51:35 [notice] 3232#3232: signal 3 (SIGQUIT) received fr
om 1, shutting down
2026/01/02 10:51:35 [notice] 3233#3233: gracefully shutting down
2026/01/02 10:51:35 [notice] 3233#3233: exiting
2026/01/02 10:51:35 [notice] 3233#3233: exit
2026/01/02 10:51:35 [notice] 3234#3234: gracefully shutting down
```

```
[ec2-user@ip-10-0-10-250 ~]$ cat /var/log/nginx/access.log
23.159.248.90 - - [02/Jan/2026:11:05:00 +0000] "CONNECT www.baidu.com:443 HTTP/1.1" 400 157 "-
" "-"
23.159.248.90 - - [02/Jan/2026:11:05:00 +0000] "CONNECT www.baidu.com:443 HTTP/1.1" 400 157 "-
" "-"
```

```
[ec2-user@ip-10-0-10-250 ~]$ cat /etc/nginx/mime.types
types {
application/A2L                                 a2l;
application/AML                                 aml;
application/andrew-inset                        ez;
application/ATF                                 atf;
application/ATFX                                atfx;
application/ATXML                               atxml;
application/atom+xml                            atom;
application/atomcat+xml                         atomcat;
application/atomdeleted+xml                     atomdeleted;
application/atomsvc+xml                         atomsvc;
application/atsc-dwd+xml                        dwd;
application/atsc-held+xml                       held;
application/atsc-rsat+xml                       rsat;
application/auth-policy+xml                     apxml;
application/bacnet-xdd+zip                      xdd;
application/calendar+xml                        xcs;
application/cbor                                cbor;
application/cccex                               c3ex;
application/ccmp+xml                            ccmp;
application/ccxml+xml                           ccxml;
application/CDFX+XML                            cdfx;
application/cdmi-capability                     cdmia;
application/cdmi-container                      cdmic;
```

```
cat: crt: No such file or directory
[ec2-user@ip-10-0-10-250 ~]$ sudo cat /etc/ssl/certs/selfsigned.crt
-----BEGIN CERTIFICATE-----
MIIDIDCCAgigAwIBAgIUbWlxz3YxStd8i1IwhyxB6RWWTRwwDQYJKoZIhvcNAQEL
BQAwFzEVMBMGA1UEAwwMMy4yOC4xNjMuMTExMB4XDTI2MDEwMjEwNTEzNVoXDTI3
MDEwMjEwNTEzNVowFzEVMBMGA1UEAwwMMy4yOC4xNjMuMTExMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmSFt3l6s5pk5+W8qcwMyXz3vobmebuV7TdJu
45kCyvkokx+sLOwskB71CKDK300bFVtbykwzxLE1uDVGeNZSqltZXCWaz57cnzjf
fTZqeDRA+VjKHArefA0cxYji8ZE5BVjFhNfpxxWK1z7pQc+GflGs5i0YKtvb0gvz
xF8RpmS3ehcHQW2DqDvpkaE5td/gUyE0oyanPBYhgcSkDOTeplTPJ6JJ1DiASzS+
zHlHxcNha+ri0tX7KjlfywiRG3bkYo1jyEiLi+wixtCfWjt+H7/8HWyJbuw2wIaP
+A24att8GuRbu1erOVSu/xtaNJnrEKFY1R+BHsBpPISGDIgCbwIDAQABo2QwYjAd
BgNVHQ4EFgQUSA14ceUzKl+3JPlPWXnTa/EArXcwHwYDVR0jBBgwFoAUSA14ceUz
Kl+3JPlPWXnTa/EArXcwDwYDVR0TAQH/BAUwAwEB/zAPBgNVHREECDAGhwQDHKNv
MA0GCSqGSIb3DQEBCwUAA4IBAQB/1tDNoOl1V/Cye1LRdvS/l1C4lGavSYzM+vHh
INBW63Bf7IVQk/UQ+dGRhzDJkcXmoEzcvzv+ii1MZRSrVPWUaPNQfNc3iOjHJ+qo
0sOvY/qMDhwgckZ5lS3SXFGSuMqLhaUPxhdz/PYkq9uKD57u3OXaswyvFSd+gRFg
a2ffSJ2R83eyQYoynMEOj9KiLnV0F39cov/dJIwd+E3MUHo+qfXv9qlqlMYF/CVh
2QnthdsC5UgUR4ZO2VQiHY/Fe35URaRM84i7E2wYxvdclL2iskv2yWYYOuYJLHSs
sO9Yocn0a/BeZd557ZOe2uHf7hE8TMz/jJYYcH+jn/upxo/6
-----END CERTIFICATE-----
```

```
[ec2-user@ip-10-0-10-250 ~]$ sudo cat /etc/ssl/private/selfsigned.key
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCZIW3eXqzmmTn5
bypzAzJfPe+huZ5u5XtN0m7jmQLK+SiTH6ws7CyQHvUIoMrfTRsVW1vKTDPEsTW4
NUZ41lKqW1lcJZrPntyfON99Nmp4NED5WMocCt58DRzFiOLxkTkFWMWE1+nHFYrX
PulBz4Z+UazmLRgq29vSC/PEXxGmZLd6FwdBbYOoO+mRoTm13+BTITSjJqc8FiGB
xKQM5N6mVM8noknUOIBLNL7MeUfFw2Fr6uLS1fsqOV/LCJEbduRijWPISIuL7CLG
0J9aO34fv/wdbIlu7DbAho/4Dbhq23wa5Fu7V6s5VK7/G1o0mesQoVjVH4EewGk8
hIYMiAJvAgMBAAECggEAB+y/993PIDwBJmncuN5ge/yuQHYPv04nByGt1XTfZ8u5
5YCic3dhiFqMdH7bIO/88n+rjMOhe5DEO/Kd4WEOHtI4GgRkMlHWucgYC4MxGA7J
kgid+meMQ2qWTkQa9+97vk1teMYydg16Wvte16YQICIaoEA925yh5V6dpfsEOdKG
CGc4eL+6MzU9cTER9y0hfzeuVT/orfsEpXyrifXmTZpP1ua6ZY2nCJEZ8U2xlXJD
AOmpPJuRJHSvDNv7d9YM4mrL5/6WAsF4p10DhYIZQsXF7CBZ48+vrgiIMw58OrXz
9icEw1XmWKDpRoatFiU58rQztLesg2yLdOOOTsdPMQKBgQDOIfMyEwSU3E5WJXV5
eBql8JMpdJesjqKJ96kvnkSZNcPaPPIKy8Cr2ohjW9f3gE5Nua1CfUjCvkhewexI
UW5msuJaIYsQ03HXdhUFzE215OAHiLzf2pUuP9TE1VdgXl+8/DSfCcbRMb8ODMdd
+bHlPAOyHBXKPl/uXOXhtgIWVwKBgQC+LP9CjYYSU7T3wIST9335rE9Ha2BdFeMa
ToFTErWv+nIAhkImZ9hF5NNQ99c20wR6fFLZjV7lJCyE1NnwSoKC7zhvnT4yLz7W
CIY+qc8uIvIElwfSH+hecbvQjQK7fYMeMMCtCu7ypXE0ZlfWuv72HGkJWtyV6O1B
7fW0cKr1qQKBgAITbXOPpXgqT8+mSOyUPRB2v8SnG+/s784b5GRKX757QCGug3GA
Y8xXQxBdGGuxqxg1Sn1k7Zqod+ocYdjImSgHnSfzNodQ2dClq8iHKjEuwOLSibRn
ayEwG2BwFZUu3h/1GkKPwqTQr2/gyRE1NolsdU0R41ZKEoVAAkCnqJW5AoGARKjg
O7n2trUr9+q0sus+ux5/vOxCbTkSthOz0zdFC1TjRdczGMSTK6DSvj/MnomO613p
V4wXv6jQveBPEtcDyZvuhzWBGANAr2uCdCtAloqdUBTrCOzmKvrq8P+IBgLpjMPT
PGZfIJGkjmOIwQjgNiDJt0vrqd/fkNcFx3CJ+IkCgYAQ9jfshNL/ZBVWCu5T4pbL
UbBXrydWOZyUcmxSsOGPRv4JrbzKIkiRruQ7okriW/qqPHYXUS9rGuGQVRzWMuPT
4TRqsbEpD4cRMTV4eaS/i5EC16J73dmDJFd+zpNTxxpDkaYkIYjMcggRFY8el66J
```

10. Test reverse proxy in browser:

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

You should see the web-1 Apache page through the Nginx proxy

Task 8 — Configure Nginx as load balancer

In this task, you will add a second backend server and configure Nginx to load balance between them.

1. **Add the second web server module to main.tf:**

main.tf

```
69   module "myapp_web_1" {
77     subnet_id          = aws_subnet.myapp_subnet_1.id
78     script_path        = "./apache.sh"
79     instance_suffix    = "1"
80   }
81   module "myapp-web-2" {
82     source = "./modules/webserver"
83     env_prefix = var.env_prefix
84     instance_type = var.instance_type
85     availability_zone = var. availability_zone
86     public_key = var.public_key
87     my_ip = local.my_ip
88     vpc_id = aws_vpc.myapp_vpc. id
89     subnet_id = module.myapp-subnet.subnet. id
90     script_path = "./apache.sh"
91     instance_suffix = "2"
92   }
```

2. **Update outputs.tf**

outputs.tf

```
1   output "webserver_public_ip"{
2     value=module.myapp_webserver.aws_instance.public_ip
3   }
4   output "aws_web_1_public_ip" {
5     value = module.myapp_web_1.aws_instance.public_ip
6   }
7   output "aws_web-2_public_ip" {
8     value = module. myapp-web-2.aws_instance.public_ip
9   }
```

3. **Apply the configuration:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ terraform apply -auto-approve
20s elapsed]
module.myapp_webserver.aws_instance.myapp-server: Still destroying... [id=i-0a94712326c37e5d0,
 00m20s elapsed]
module.myapp_web_1.aws_instance.myapp-server: Destruction complete after 30s
module.myapp_web_1.aws_instance.myapp-server: Creating...
module.myapp_webserver.aws_instance.myapp-server: Destruction complete after 30s
module.myapp_webserver.aws_instance.myapp-server: Creating...
module.myapp_web_1.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp_webserver.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp_web_1.aws_instance.myapp-server: Creation complete after 12s [id=i-03d3d695465252
026]
module.myapp_webserver.aws_instance.myapp-server: Creation complete after 12s [id=i-0be3eeab38
95a6e0c]

Apply complete! Resources: 5 added, 0 changed, 2 destroyed.

Outputs:

aws_web_1_public_ip = "158.252.78.204"
aws_web_2_public_ip = "3.29.231.98"
webserver_public_ip = "51.112.51.163"
```

4. **Get all outputs:**

```
webserver_public_ip =  51.112.51.163
@Anara-hayat →/workspaces/Lab12 (main) $ terraform output
aws_web_1_public_ip = "158.252.78.204"
aws_web_2_public_ip = "3.29.231.98"
webserver_public_ip = "51.112.51.163"
```

5. **SSH into the webserver (Nginx proxy):**

```
@Anara-hayat →/workspaces/Lab12 (main) $ ssh ec2-user@51.112.51.163
The authenticity of host '51.112.51.163 (51.112.51.163)' can't be established.
ED25519 key fingerprint is SHA256:n818lGyvEGGN35yhVvbnZLB0pFND5wGIka0tR5prnK8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.51.163' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-56 ~]$
```

6. **Edit Nginx configuration for load balancing:**

```
############################
upstream backend_servers {
    server 158.252.78.204:80;
    server 3.29.231.98:80;
}

server {
    listen       80;
    server_name  localhost;

    # Default location
    location / {
            #root/usr/share/nginx/html;
            #index index.html;
            #proxy_pass http://158.252.78.204:80;
        proxy_pass http://backend_servers;
    }
```

7. **Restart Nginx:**

```
[ec2-user@ip-10-0-10-56 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-56 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Fri 2026-01-02 11:41:47 UTC; 6s ago
    Process: 25936 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 25937 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 25938 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 25939 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.2M
        CPU: 40ms
     CGroup: /system.slice/nginx.service
             ├─25939 "nginx: master process /usr/sbin/nginx"
             ├─25940 "nginx: worker process"
             └─25941 "nginx: worker process"

Jan 02 11:41:47 ip-10-0-10-56.me-central-1.compute.internal systemd[1]: Starting nginx.servic
```

8. **Test load balancing in browser:**

- Open browser and navigate to https://<webserver-public-ip>

- Reload the page multiple times
- You should see the content alternating between web-1 and web-2 (check the hostname/IP in the page)

Task 9 — Configure high availability with backup servers

In this task, you will configure one server as primary and another as backup for high availability.

1. **SSH into the webserver:**

```
@Anara-hayat →/workspaces/Lab12 (main) $ ssh ec2-user@51.112.51.163
The authenticity of host '51.112.51.163 (51.112.51.163)' can't be established.
ED25519 key fingerprint is SHA256:n818lGyvEGGN35yhVvbnZLB0pFND5wGIka0tR5prnK8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.51.163' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

2. **Edit Nginx configuration for high availability:**

```
access_log  /var/log/nginx/access.log  main;

sendfile        on;
keepalive_timeout  65;

upstream backend_servers {
    server 158.252.78.204:80;   # aws_web_1_public_ip
    server 3.29.231.98:80 backup;      # aws_web_2_public_ip
}
server {
    listen       80;
    server_name  localhost;

    location / {
        proxy_pass http://backend_servers;
        proxy_set_header Host $host;
```

3. **Restart Nginx:**

```
[ec2-user@ip-10-0-10-56 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-56 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disab>
     Active: active (running) since Fri 2026-01-02 12:12:57 UTC; 21s ago
    Process: 26872 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status>
    Process: 26874 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 26875 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 26876 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.2M
        CPU: 43ms
     CGroup: /system.slice/nginx.service
             ├─26876 "nginx: master process /usr/sbin/nginx"
             ├─26877 "nginx: worker process"
             └─26878 "nginx: worker process"
```

4. **Test in browser:**

- **Open browser and navigate to https://<webserver-public-ip>**
- **Reload multiple times**
- **You should ONLY see web-1 (primary server)**

5. **Switch backup configuration:**

```
sendfile        on;
keepalive_timeout  65;

upstream backend_servers {
    server 158.252.78.204:80 backup;    # aws_web_1_public_ip
    server 3.29.231.98:80;      # aws_web_2_public_ip
}
server {
    listen       80;
    server_name  localhost;

    location / {
        proxy_pass http://backend_servers;
        proxy_set_header Host $host;
```

6. **Restart Nginx:**

```
[ec2-user@ip-10-0-10-56 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-56 ~]$
```

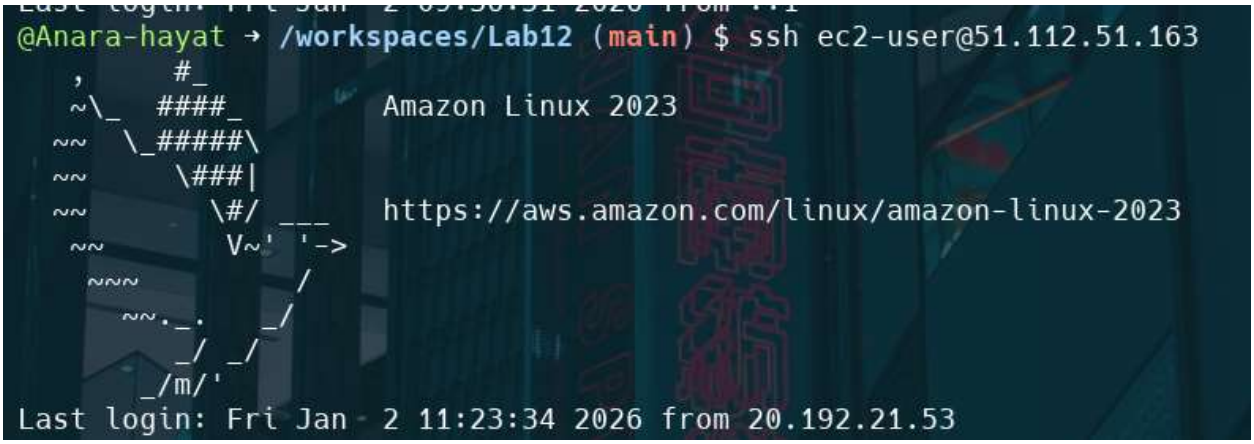
7. **Test in browser:**

- **Reload multiple times**
- **You should ONLY see web-2 (now the primary server)**

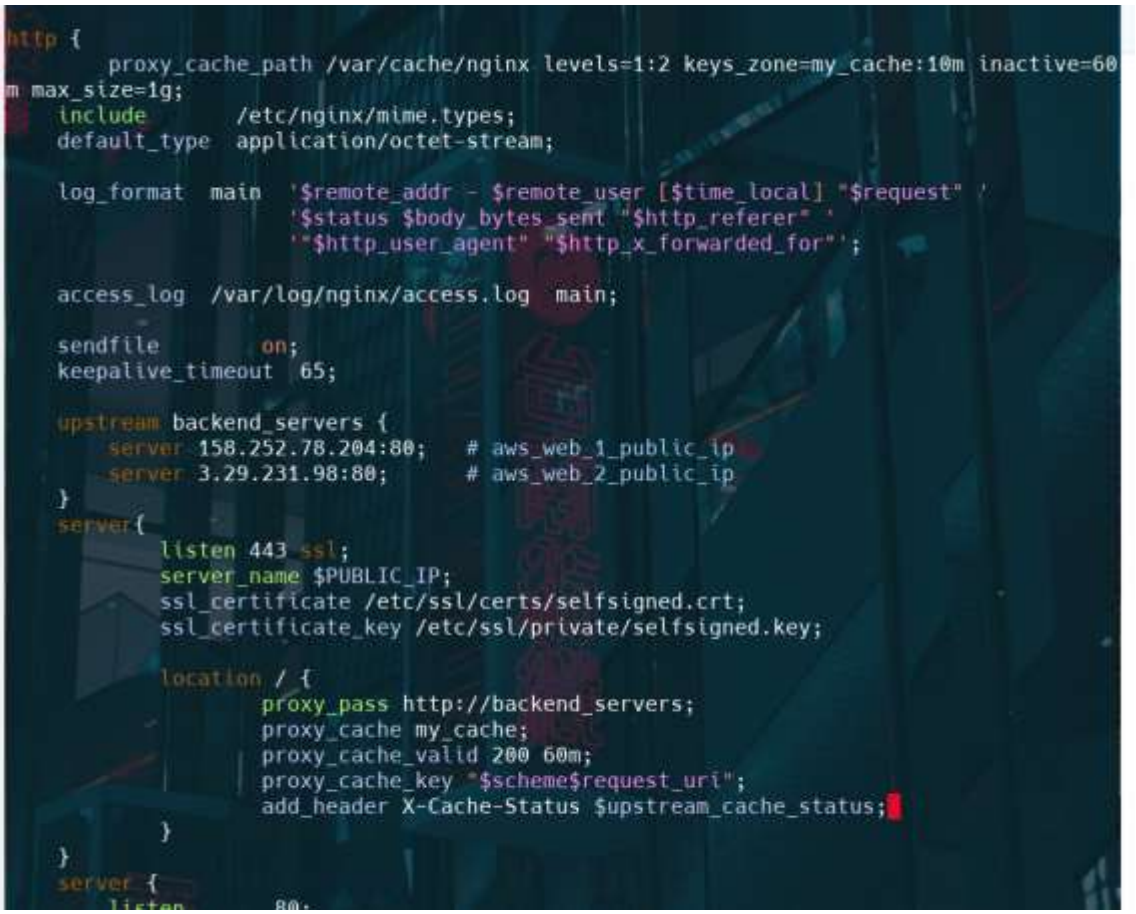
Task 10 — Enable Nginx caching

In this task, you will enable caching in Nginx to improve performance.

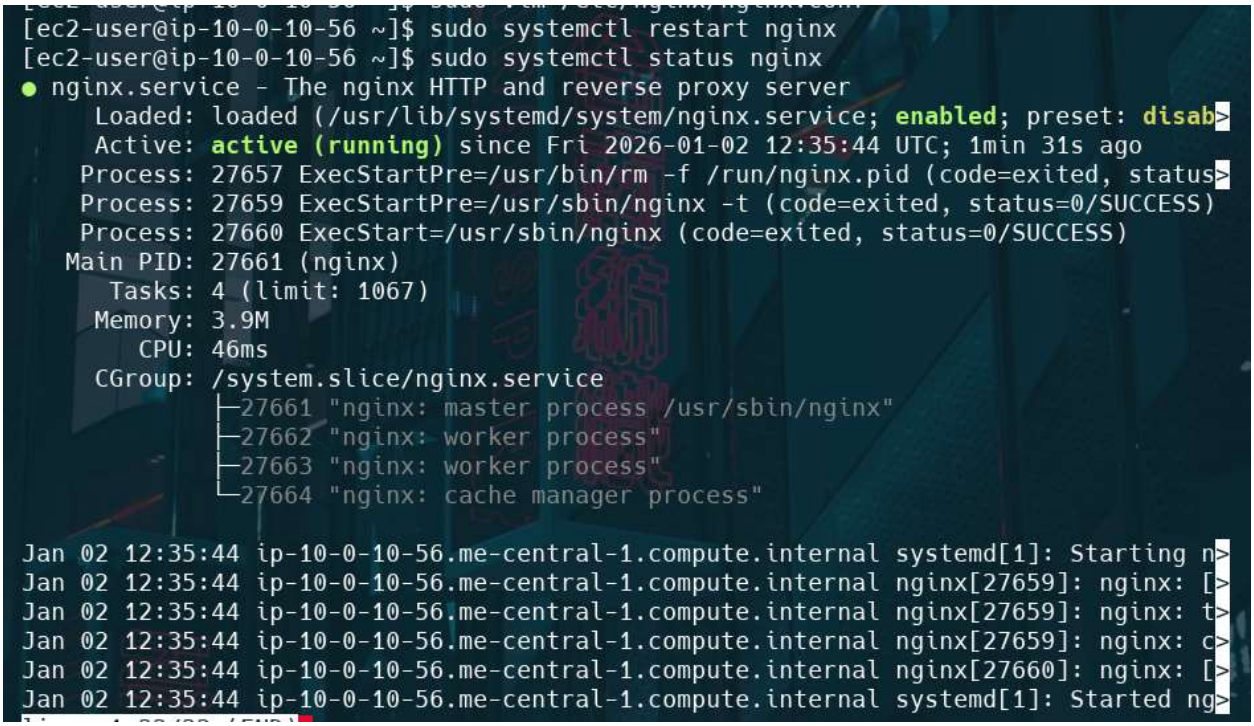
1. **SSH into the webserver:**



2. **Edit Nginx configuration to enable caching:**

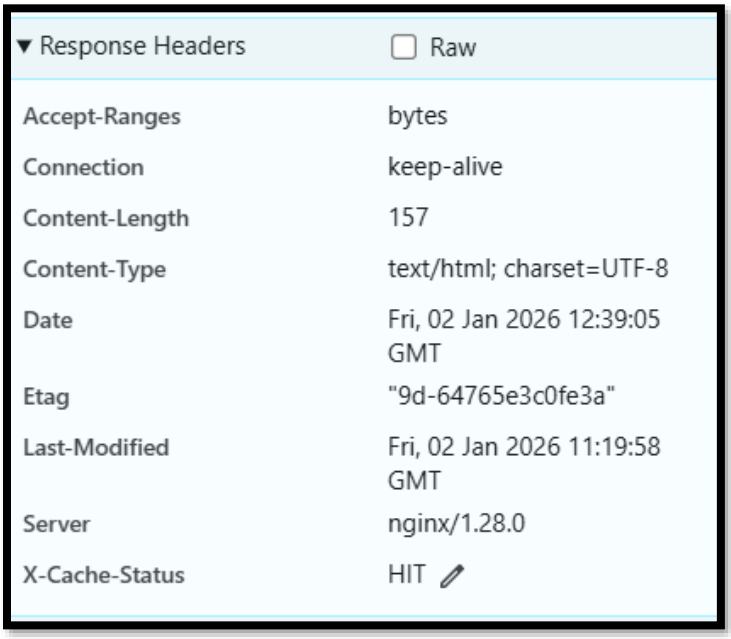


3. **Restart Nginx:**



4. **Test caching in browser:**

- **Open browser developer tools (F12)**
- **Navigate to Network tab**
- **Visit https://<webserver-public-ip>**
- **Check response headers for X-Cache-Status**
- **First request should show MISS**
- **Reload the page**
- **Second request should show HIT**



5. **Verify cache directory:**

```
[ec2-user@ip-10-0-10-56 ~]$ sudo ls -la /var/cache/nginx/
total 0
drwx------. 3 nginx root   15 Jan  2 12:38 .
drwxr-xr-x. 9 root  root  101 Jan  2 12:35 ..
drwx------. 3 nginx nginx  16 Jan  2 12:38 4
```

Cleanup

1. **Exit SSH session:**

```
[ec2-user@ip-10-0-10-56 ~]$ exit
logout
Connection to 51.112.51.163 closed.
@Anara-hayat → /workspaces/Lab12 (main) $
```

2. **Destroy all resources:**

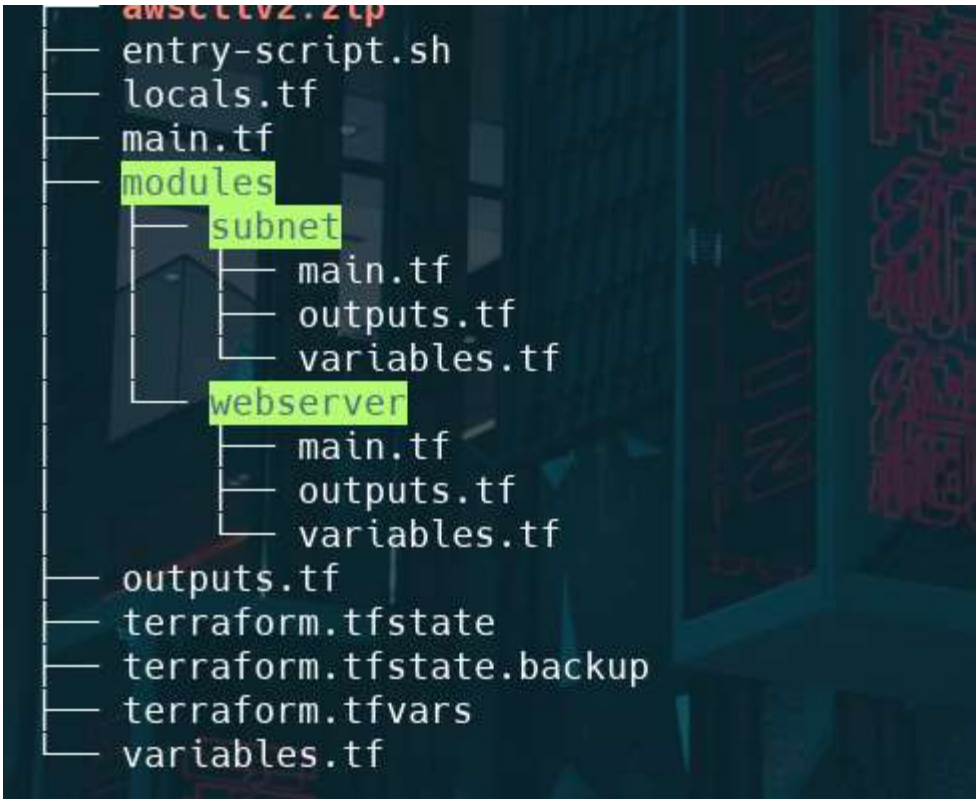
```
Plan: 0 to add, 0 to change, 14 to destroy.

Changes to Outputs:
  - aws_web_1_public_ip = "158.252.78.204" -> null
  - aws_web_2_public_ip = "3.29.231.98" -> null
  - webserver_public_ip = "51.112.51.163" -> null

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
```

```
module.myapp_webserver.aws_instance.myapp-server: Still destroying... [id=i-0be3eea
b3895a6e0c, 01m00s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0b47a6259398f2a58, 01m0
0s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 1m7s
module.myapp_webserver.aws_instance.myapp-server: Still destroying... [id=i-0be3eea
b3895a6e0c, 01m10s elapsed]
module.myapp_webserver.aws_instance.myapp-server: Destruction complete after 1m10s
module.myapp_webserver.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-0]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0cee13893c010dee1]
module.myapp_webserver.aws_security_group.web_sg: Destroying... [id=sg-0f16f39c9cbc
e36db]
module.myapp_webserver.aws_key_pair.ssh-key: Destruction complete after 1s
aws_subnet.myapp_subnet_1: Destruction complete after 1s
module.myapp_webserver.aws_security_group.web_sg: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0b8820211b0a292d3]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 14 destroyed.
```

3. **Verify state files:**

```
@Anara-hayat → /workspaces/Lab12 (main) $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.3",
  "serial": 152,
  "lineage": "14d02829-edc2-95c1-7dff-d34874e444e2",
  "outputs": {},
  "resources": [],
  "check_results": null
}
```

4. **List all project files:**


awscliv2.zip
entry-script.sh
locals.tf
main.tf
modules
subnet
main.tf
outputs.tf
variables.tf
webserver
main.tf
outputs.tf
variables.tf
outputs.tf
terraform.tfstate
terraform.tfstate.backup
terraform.tfvars
variables.tf

****