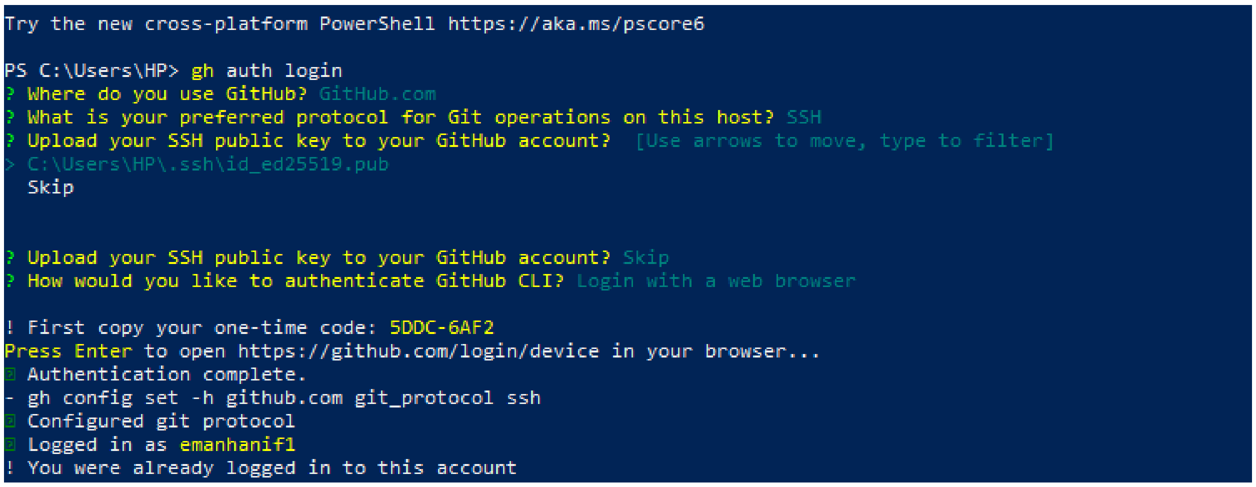
Lab 12 – Terraform Provisioners, Modules & Nginx Reverse Proxy/Load Balancer

Objective

- In this lab you will:

- Use GH CLI to work inside a Codespace.
- Organize Terraform code into separate files (variables. tf, outputs.tf, locals.tf, main.tf, terraform.tfvars).
- Practice Terraform provisioners: user_data, remote-exec, file, and local-exec.
- Create reusable Terraform modules for subnet and webserver resources.
- Configure Nginx as a reverse proxy and load balancer.
- Implement SSL/TLS with self-signed certificates.
- Configure Nginx caching for improved performance.
- Implement high availability patterns with backup servers

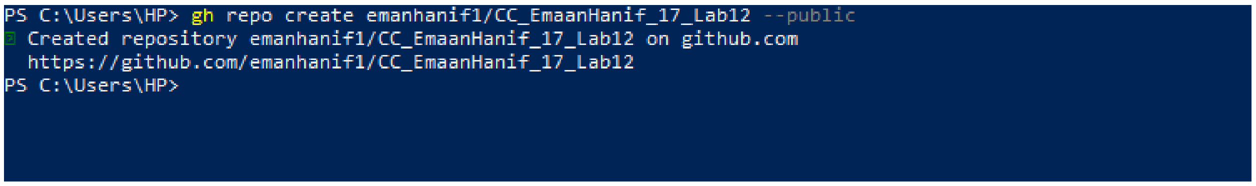


```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\HP> gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account?  [Use arrows to move, type to filter]
> C:\Users\HP\.ssh\id_ed25519.pub
  Skip


? Upload your SSH public key to your GitHub account? Skip
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 5DDC-6AF2
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Logged in as emanhanif1
! You were already logged in to this account
```

Task 0: Lab Setup (Codespace & GitHub CLI)

1. Repository Creation

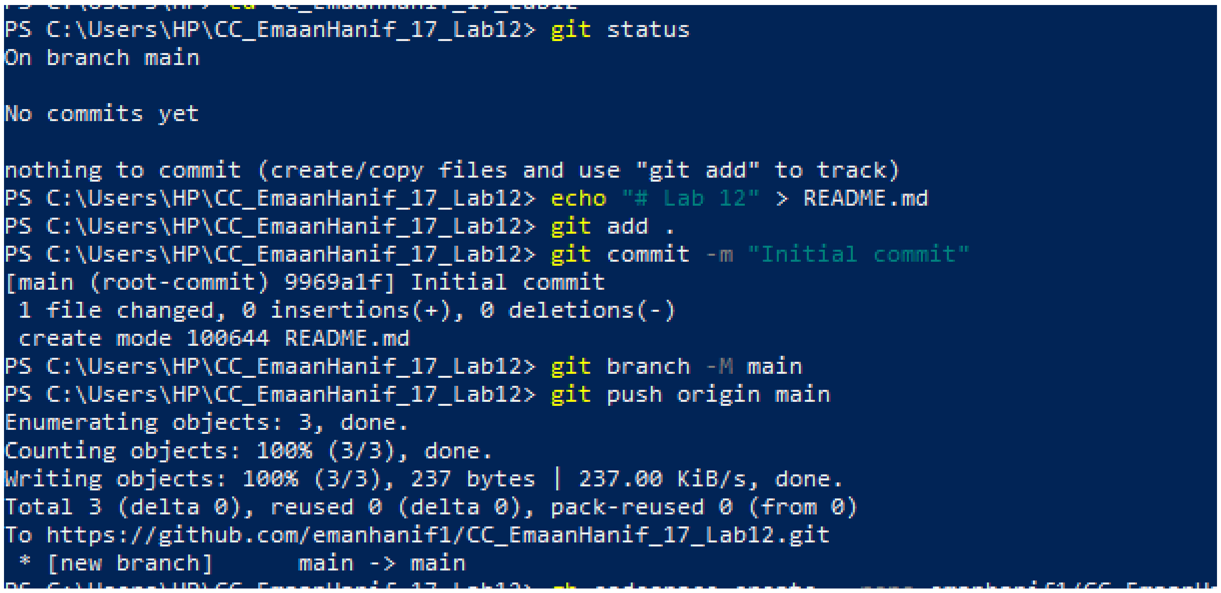
- Using GitHub CLI, a new repository was created under the GitHub username emanhanif1 with the name CC_EmaanHanif_17_Lab12.
- Command used:
- gh repo create emanhanif1/CC_EmaanHanif_17_Lab12 --public
- This repository will host all lab assignments and act as the source for the Codespace.



```
PS C:\Users\HP> gh repo create emanhanif1/CC_EmaanHanif_17_Lab12 --public
✓ Created repository emanhanif1/CC_EmaanHanif_17_Lab12 on github.com
  https://github.com/emanhanif1/CC_EmaanHanif_17_Lab12
PS C:\Users\HP>
```

Repository Initialization

- Codespace creation failed because the repository **did not have any branch or initial commit**.
- To resolve this:
    1. Clone the repository locally.
    2. Add a README.md file.
    3. Make the initial commit.
    4. Push to the main branch

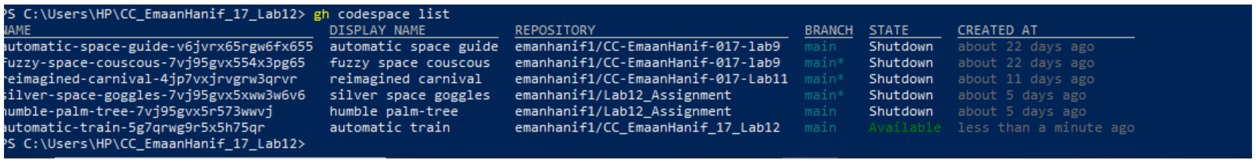


```
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> echo "# Lab 12" > README.md
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> git add .
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> git commit -m "Initial commit"
[main (root-commit) 9969a1f] Initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> git branch -M main
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 237 bytes | 237.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/emanhanif1/CC_EmaanHanif_17_Lab12.git
 * [new branch]      main -> main
```

Verification and SSH Access

- Codespace list verified the Codespace creation:

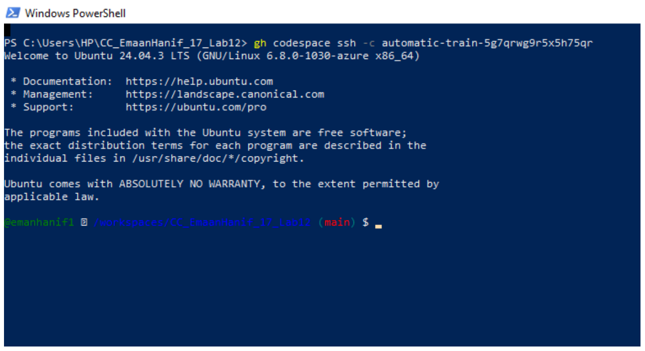
  gh codespace list



```
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> gh codespace list
NAME                                 DISPLAY NAME           REPOSITORY                               BRANCH  STATE      CREATED AT
automatic-space-guide-v6jvrx65rgw6fx655  automatic space guide  emanhanif1/CC-EmaanHanif-017-lab9    main    Shutdown   about 22 days ago
fuzzy-space-couscous-7vj95gvx554x3pg65   fuzzy space couscous   emanhanif1/CC-EmaanHanif-017-lab9    main*   Shutdown   about 22 days ago
reimagined-carnival-4jp7vxjrvgrw3qrvr    reimagined carnival    emanhanif1/CC-EmaanHanif-017-Lab11   main*   Shutdown   about 11 days ago
silver-space-goggles-7vj95gvx5xww3w6v6   silver space goggles   emanhanif1/Lab12_Assignment          main*   Shutdown   about 5 days ago
humble-palm-tree-7vj95gvx5r573wwvj       humble palm-tree       emanhanif1/Lab12_Assignment          main    Shutdown   about 5 days ago
automatic-train-5g7qrwg9r5x5h75qr        automatic train        emanhanif1/CC_EmaanHanif_17_Lab12    main    Available  less than a minute ago
PS C:\Users\HP\CC_EmaanHanif_17_Lab12>
```

SSH into the Codespace shell:

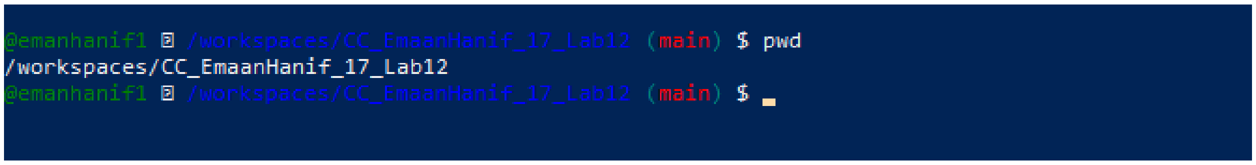
gh codespace ssh -c <codespace_name>



```
Windows PowerShell
PS C:\Users\HP\CC_EmaanHanif_17_Lab12> gh codespace ssh -c automatic-train-5g7qrwg9r5x5h75qr
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.

@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12 (main) $ _
```

Inside the Codespace, the working directory confirmed the repository path:

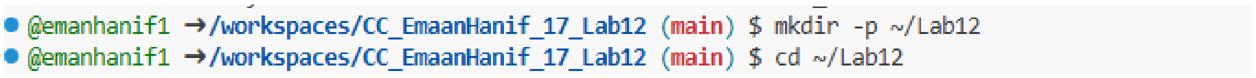
pwd



```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12 (main) $ pwd
/workspaces/CC_EmaanHanif_17_Lab12
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12 (main) $ _
```

Task1- **Organize Terraform code into separate files**

Step 1: Create Project Directory

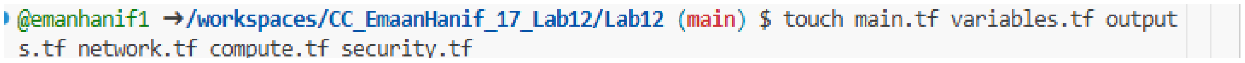


```
● @emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12 (main) $ mkdir -p ~/Lab12
● @emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12 (main) $ cd ~/Lab12
```

- Created a new project directory named **Lab12** using the `mkdir -p` command.
- Navigated into the **Lab12** directory using the `cd` command.
- This directory serves as the root folder for all files and configurations of the project.

Step 2: Create Terraform File Structure

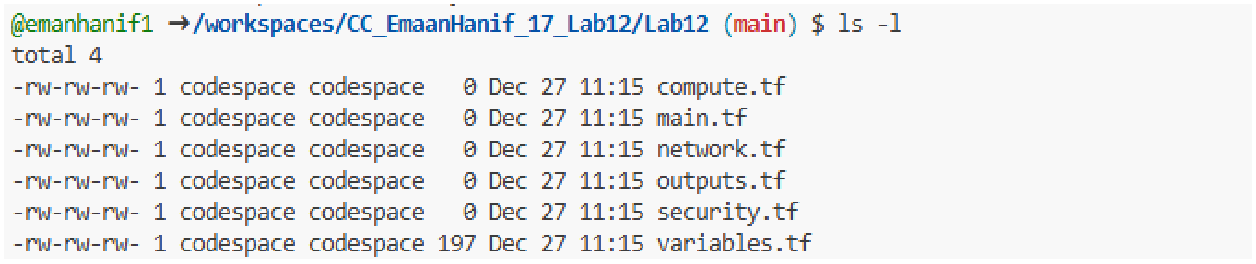
Main Terraform files



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ touch main.tf variables.tf output
s.tf network.tf compute.tf security.tf
```

- Created Terraform configuration files (**main.tf**, **variables.tf**, **outputs.tf**, **locals.tf**) to define infrastructure resources, inputs, outputs, and local values.
- Created **terraform.tfvars** to store variable values separately from the main configuration.
- Created **entry-script.sh** to automate instance initialization tasks during deployment.

- Listed all files and directories with detailed permissions using `ls -la`.
- Confirmed that all required Terraform and script files were successfully created in the project directory.



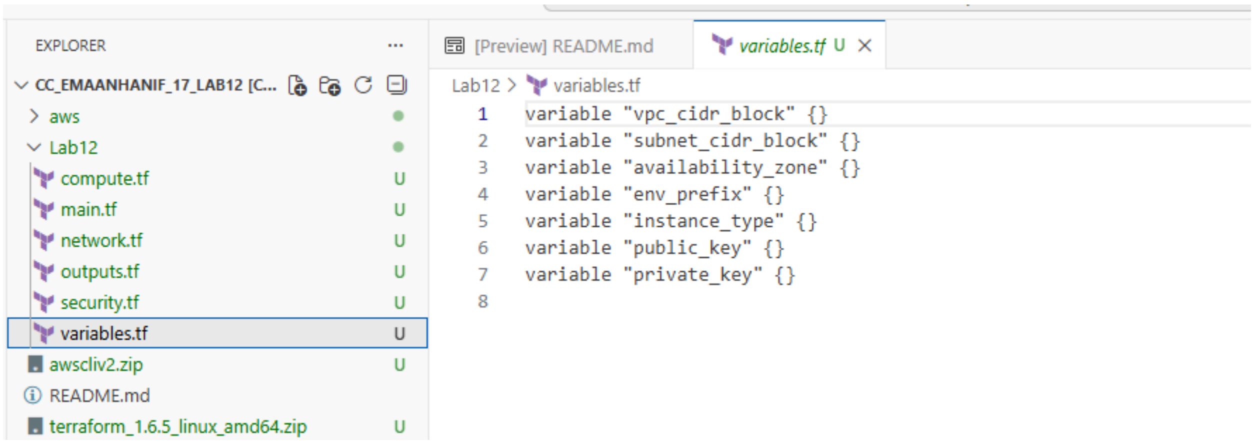
```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ls -l
total 4
-rw-rw-rw- 1 codespace codespace   0 Dec 27 11:15 compute.tf
-rw-rw-rw- 1 codespace codespace   0 Dec 27 11:15 main.tf
-rw-rw-rw- 1 codespace codespace   0 Dec 27 11:15 network.tf
-rw-rw-rw- 1 codespace codespace   0 Dec 27 11:15 outputs.tf
-rw-rw-rw- 1 codespace codespace   0 Dec 27 11:15 security.tf
-rw-rw-rw- 1 codespace codespace 197 Dec 27 11:15 variables.tf
```

Now your Lab12 folder is ready to organize Terraform code into separate files.

2)

Add Variables to variables.tf

- ❖ Opened the `variables.tf` file in the project directory.
- ❖ Defined input variables for VPC CIDR, subnet CIDR, availability zone, environment prefix, instance type, and SSH keys.
- ❖ Add the following content to variables.tf


EXPLORER
CC_EMAANHANIF_17_LAB12 [C...
aws
Lab12
compute.tf U
main.tf U
network.tf U
outputs.tf U
security.tf U
variables.tf U
awscliv2.zip U
README.md
terraform_1.6.5_linux_amd64.zip U

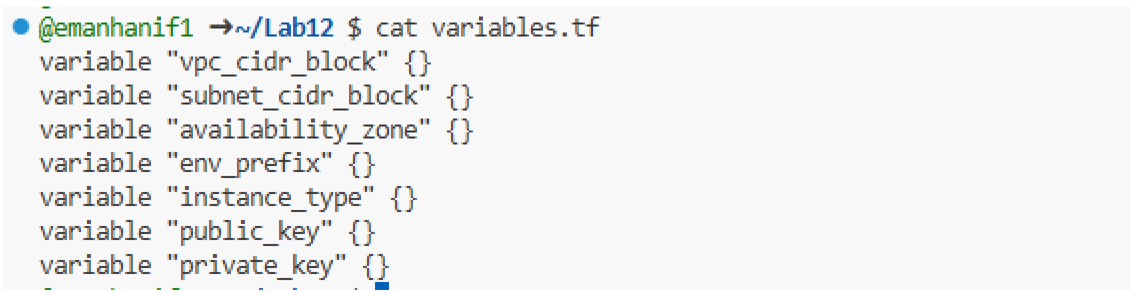
[Preview] README.md | variables.tf U

Lab12 > 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" {}
```

- Verify:

cat variables.tf



```
@emanhanif1 ➜~/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" {}
```

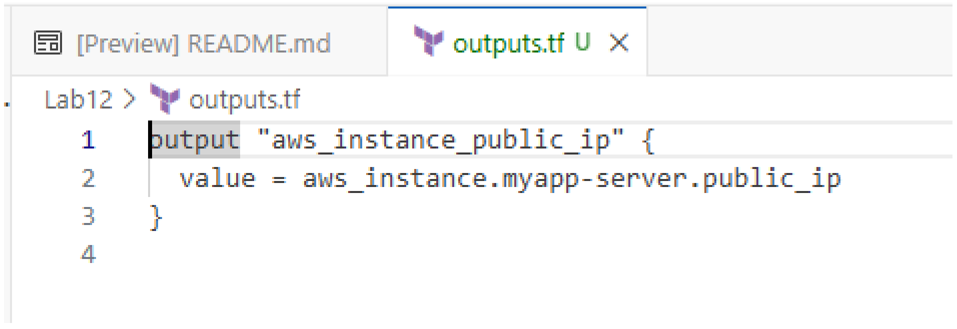
- ❖ Verified the contents of **variables.tf** using the `cat variables.tf` command to ensure all variable definitions were correctly added.

Defines all the input variables that Terraform will use for VPC, subnets, availability zone, instance type, and SSH keys.

3) Create `outputs.tf` with the following content:

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

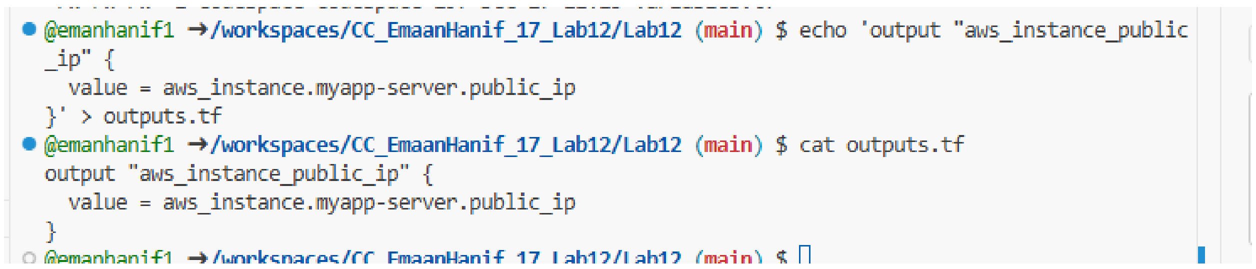
- Opened the `outputs.tf` file.
- Defined an output to display the public IP address of the EC2 instance after deployment.



[Preview] README.md | outputs.tf U ×

Lab12 > outputs.tf

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

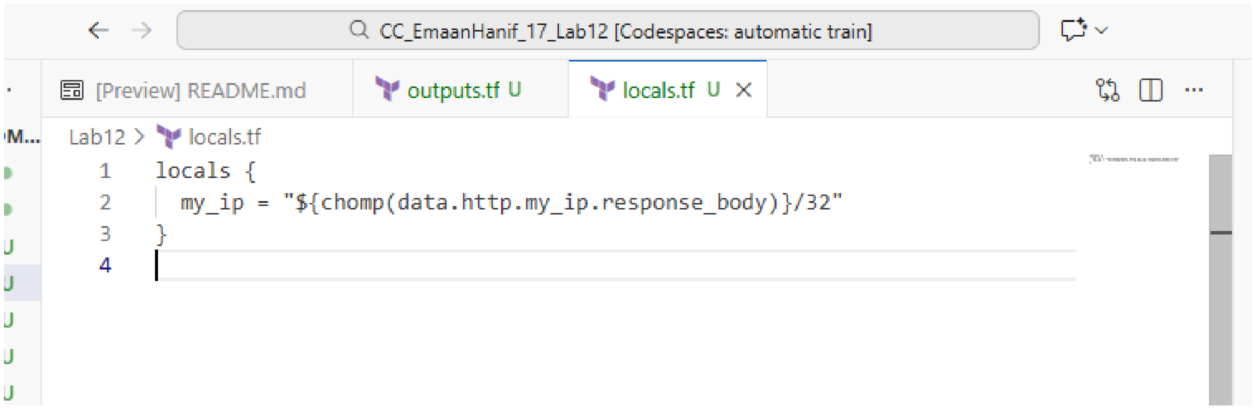


```
● @emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ echo 'output "aws_instance_public
  _ip" {
    value = aws_instance.myapp-server.public_ip
  }' > outputs.tf
● @emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat outputs.tf
  output "aws_instance_public_ip" {
    value = aws_instance.myapp-server.public_ip
  }
○ @emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ 
```

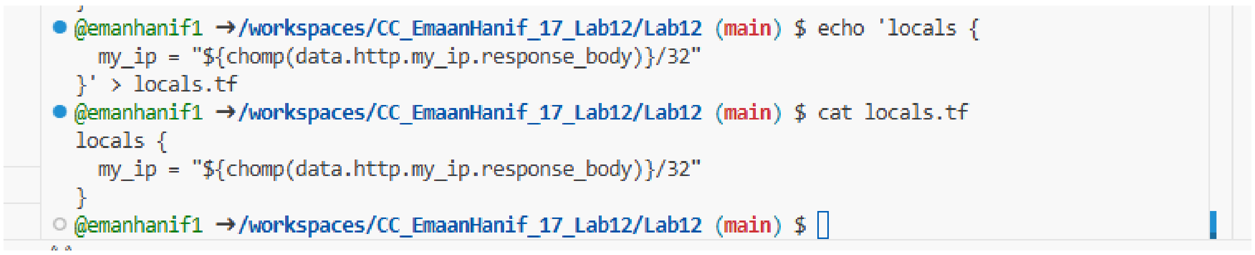
- Defines what Terraform will **output after deployment**.
- Here, it will display the **public IP** of the EC2 instance myapp-server.

locals.tf

- ❖ Created the **locals.tf** file in the project directory.
- ❖ Defined a local variable `my_ip` to automatically detect the current public IP address and append `/32` for CIDR notation.
- ❖ Added an HTTP data source to fetch the public IP from an external service.



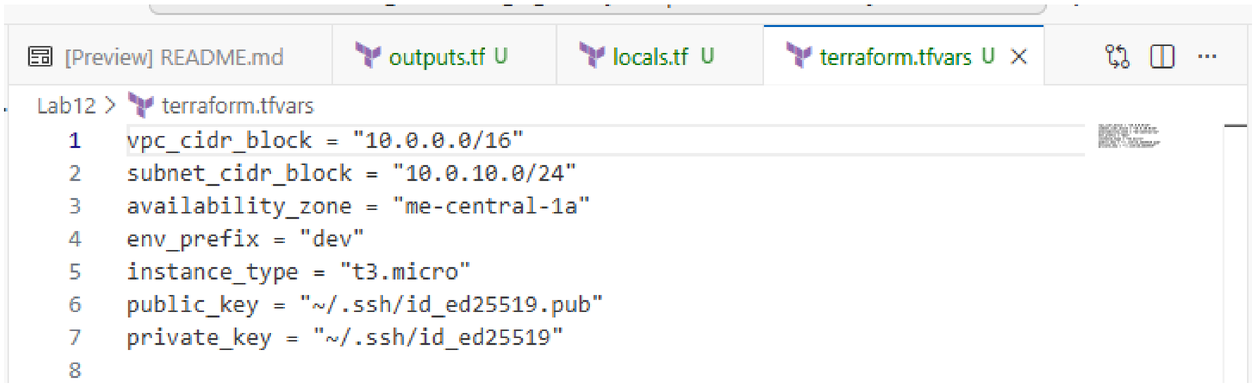
```
locals {
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}
```



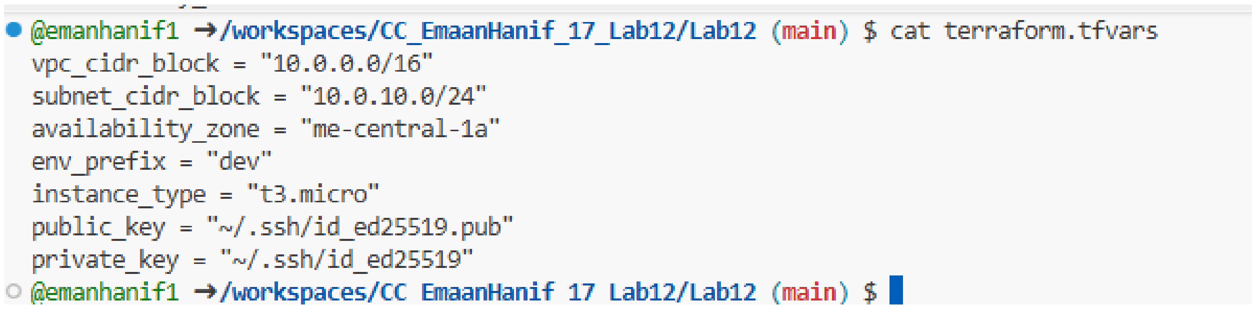
```
● @emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ echo 'locals {
    my_ip = "${chomp(data.http.my_ip.response_body)}/32"
  }' > locals.tf
● @emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat locals.tf
  locals {
    my_ip = "${chomp(data.http.my_ip.response_body)}/32"
  }
○ @emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ 
```

- Here, my_ip stores your **current public IP** (from HTTP request) with /32 CIDR, useful for **security group rules**.

Create terraform.tfvars



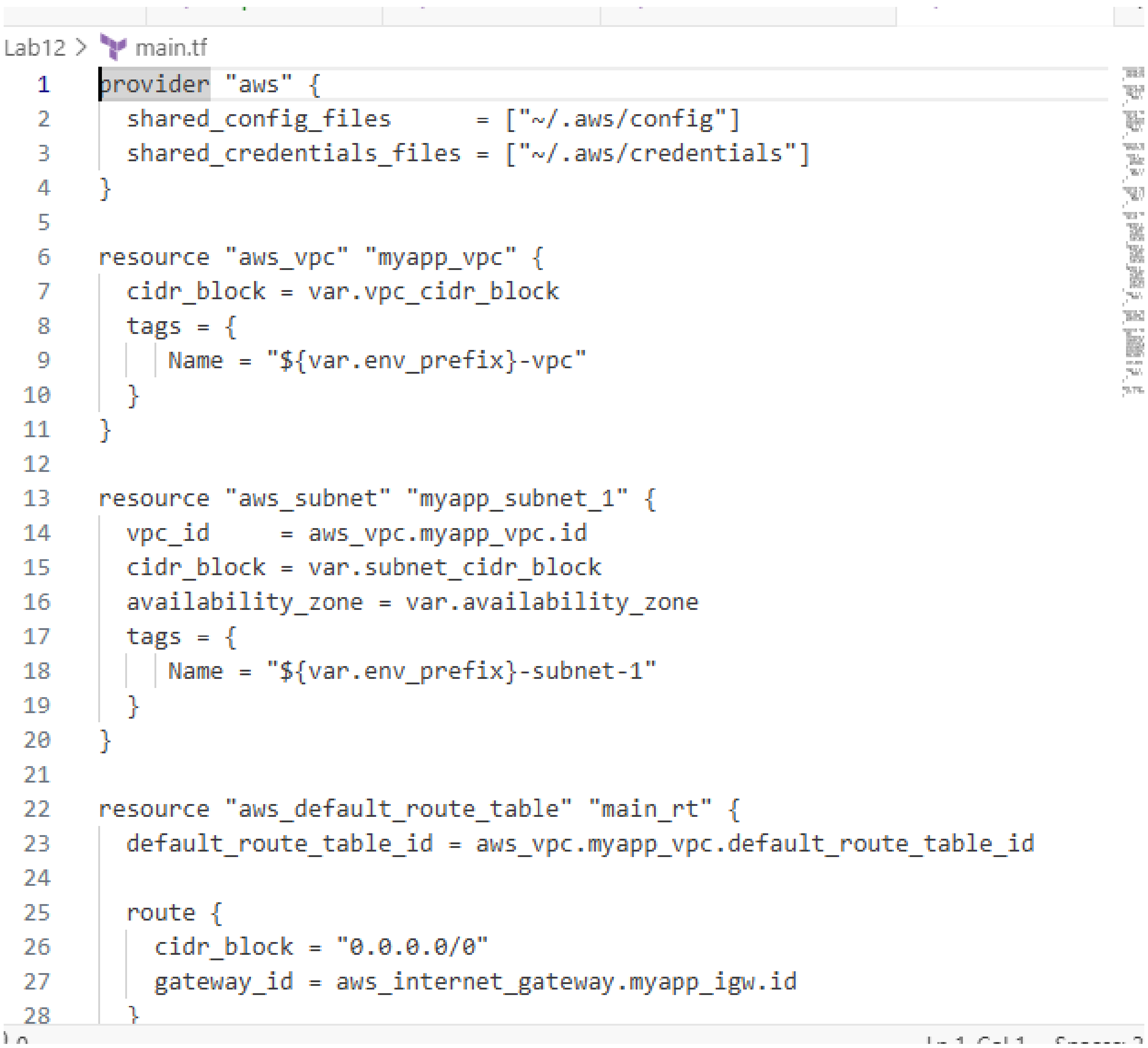
```
vpc_cidr_block = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix = "dev"
instance_type = "t3.micro"
public_key = "~/.ssh/id_ed25519.pub"
private_key = "~/.ssh/id_ed25519"
```



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat terraform.tfvars
vpc_cidr_block = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix = "dev"
instance_type = "t3.micro"
public_key = "~/.ssh/id_ed25519.pub"
private_key = "~/.ssh/id_ed25519"
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

1. Purpose of terraform.tfvars
   - Provides **values for the variables** defined in variables.tf.
   - Example:
     - vpc_cidr_block corresponds to variable "vpc_cidr_block"
     - instance_type corresponds to variable "instance_type"
2. Why separate file
   - Makes it easy to **change values** without editing main Terraform files.
   - Terraform automatically reads this file and **fills the variables** when applying.
3. SSH keys
   - public_key and private_key point to your local SSH key pair, used for EC2 access.

7)MAIN.TF



Lab12 > main.tf

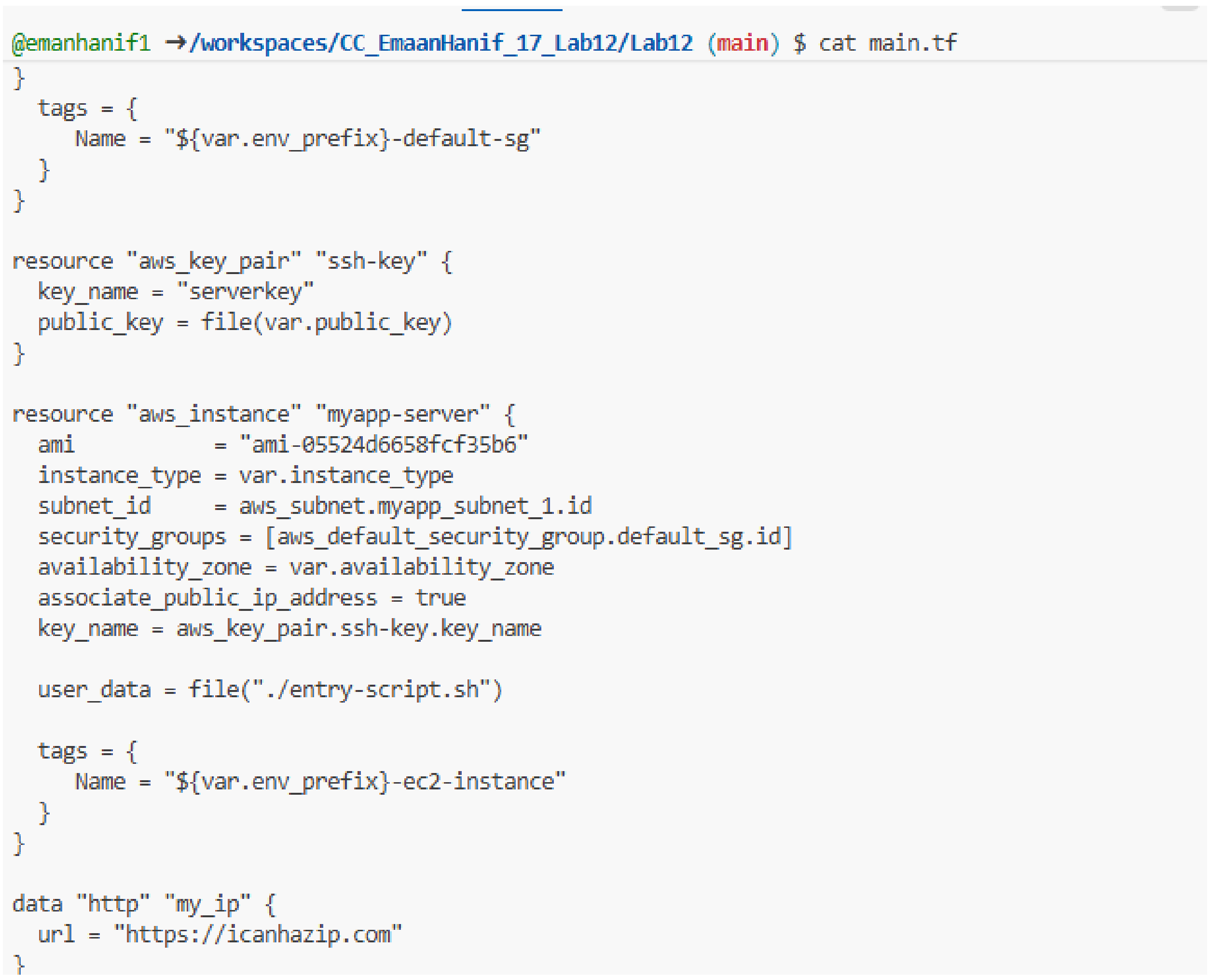
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

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
  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"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }
```

- ❖ Created the **main.tf** file in directory.
- ❖ Configured the **AWS provider** to use local AWS credentials and config files.
- ❖ Defined a **VPC** with a CIDR block from `var.vpc_cidr_block`
- ❖ Created a **subnet** within the VPC
- ❖ Configured a **default route table** with a route to the internet via an **Internet Gateway**.
- ❖ Created an **Internet Gateway** for the VPC
- ❖ Configured the **default security group** to allow SSH from the local IP and HTTP from all IPs, with unrestricted egress.
- ❖ Added an **AWS key pair** using the public key file for SSH access.
- ❖ Created an **EC2 instance**

Verify content

cat main.tf

Output should display **the full content of main.tf**.



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat main.tf
}
  tags = {
     Name = "${var.env_prefix}-default-sg"
  }
}

resource "aws_key_pair" "ssh-key" {
  key_name = "serverkey"
  public_key = file(var.public_key)
}

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

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

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

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

- ❖ provider "aws"
  - o Specifies AWS **credentials and config** files.
- ❖ aws_vpc
  - o Creates a **VPC** where all resources will reside.

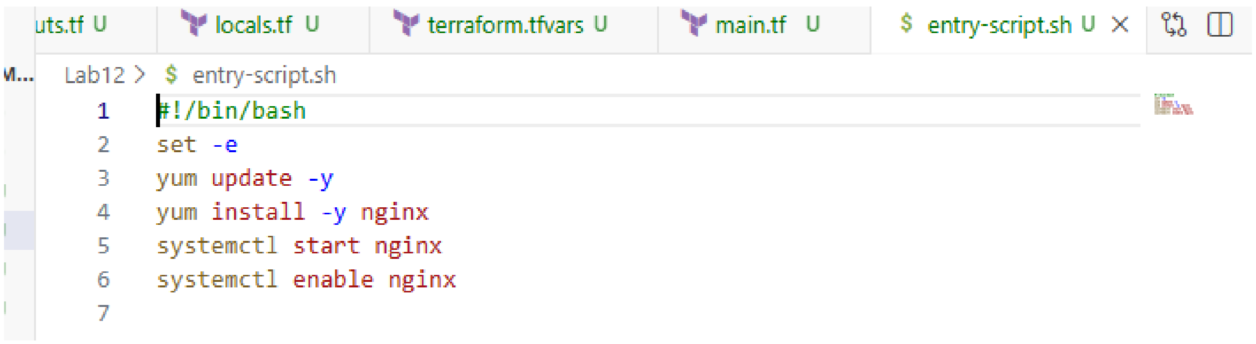
- aws_subnet
  - Creates a **subnet** inside the VPC.
- aws_internet_gateway & default route table
  - Provides **internet access** for the VPC.
- aws_default_security_group
  - Defines **security rules**:
    - SSH (port 22) from **your current IP**
    - HTTP (port 80) open to all
  - Uses local.my_ip for dynamic IP.

- aws_key_pair
  - Defines **SSH key** for EC2 access.
- aws_instance
  - Creates an **EC2 instance** with public IP, security groups, and user_data script.
- data "http" "my_ip"
  - Fetches your **current public IP** dynamically.

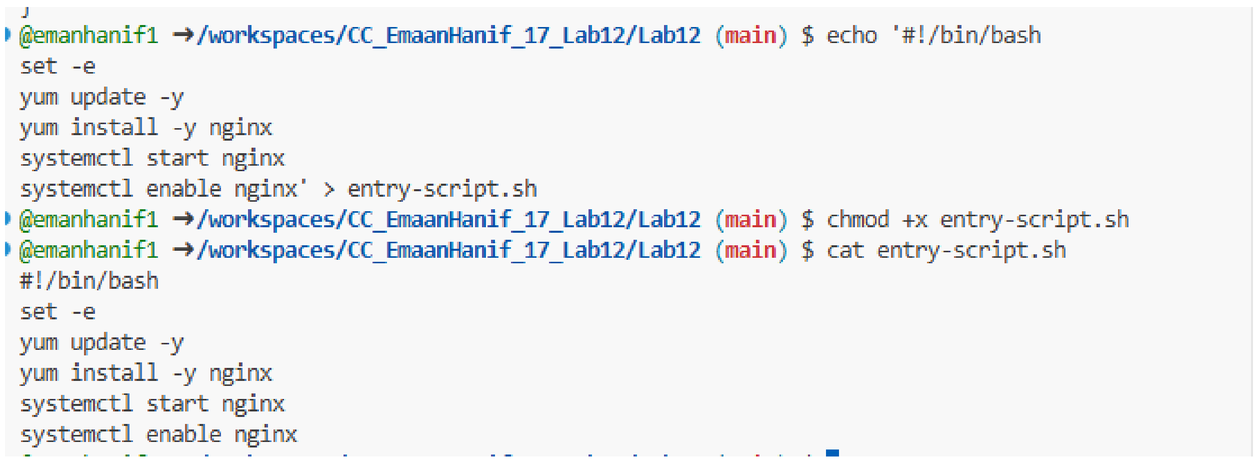
Summary

- main.tf contains **all core resources and provider configuration**.
- Other files (network.tf, compute.tf, security.tf) can later be used to **split resources** for organization.

8)



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

```

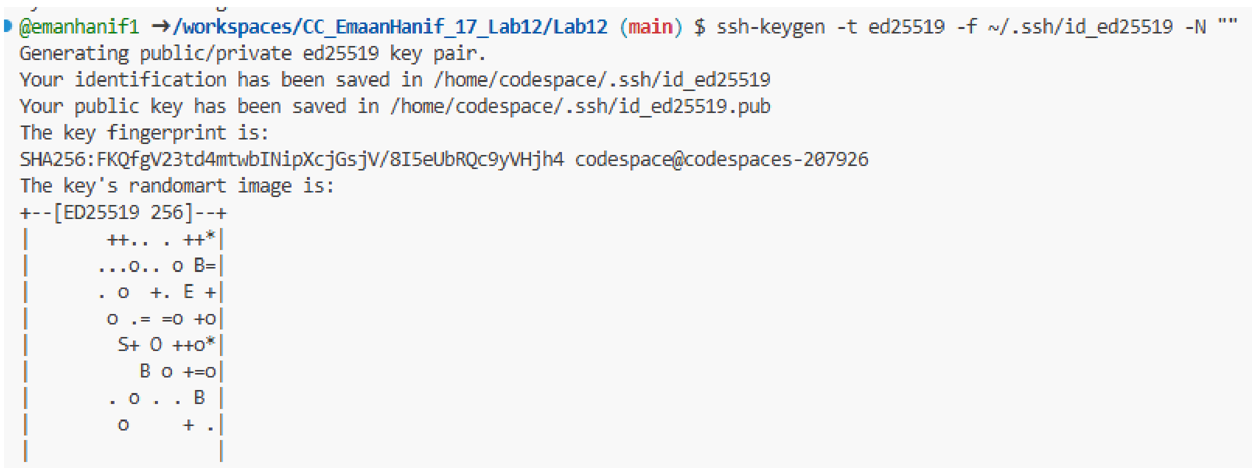


```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ echo '#!/bin/bash
set -e
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx' > entry-script.sh
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ chmod +x entry-script.sh
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat entry-script.sh
#!/bin/bash
set -e
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
```

- #!/bin/bash
  - Tells the system that this is a **bash script**.
- set -e
  - Stops the script if **any command fails**.
- yum update -y
  - Updates all packages automatically.
- yum install -y nginx
  - Installs **nginx web server** without asking confirmation.
- systemctl start nginx
  - Starts the **nginx service** immediately.
- systemctl enable nginx
  - Ensures nginx **starts automatically** on instance boot.

Why: This script runs when EC2 instance launches (user_data). It automatically sets up **nginx**, so the instance is ready without manual intervention.

9) Generate SSH key pair if not already exists:



```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/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:FKQfgV23td4mtwbINipXcjGsjV/8I5eUbRQc9yVHjh4 codespace@codespaces-207926
The key's randomart image is:
+--[ED25519 256]--+
|       ++.. . ++*|
|      ...o.. o B=|
|      . o  +. E +|
|       o .= =o +o|
|        S+ O ++o*|
|          B o +=o|
|       . o . . B |
|        o     + .|
|                 |
```

❖ ssh-keygen
   ○ Generates a **secure key pair** for SSH login.
❖ -t ed25519
   ○ Uses **ed25519 algorithm** (modern and secure).
❖ -f ~/.ssh/id_ed25519
   ○ Saves the key in **.ssh folder** as id_ed25519 (private) and id_ed25519.pub (public).
❖ -N ""
   ○ No passphrase, login will be automatic with the key.

PURPOSE

- EC2 instances require **SSH keys** for login.
- Public key is added to the instance (aws_key_pair resource).
- Private key stays with you → you use it to login securely.

9)

1. Initialize Terraform

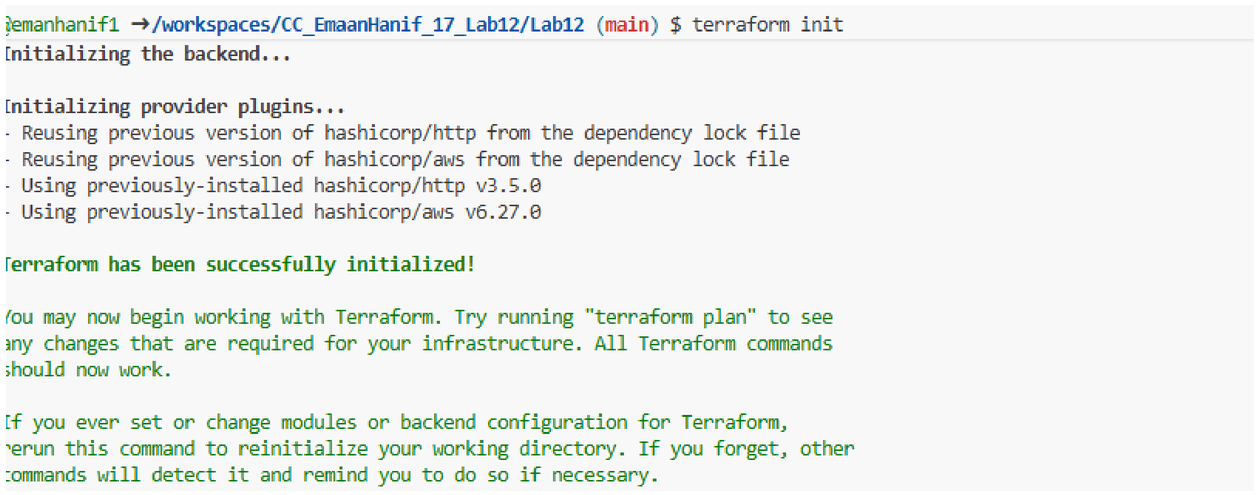
9. Generate SSH key pair if not already exists:

terraform init

What it does:

- Downloads **provider plugins** (AWS in this case).

- Prepares the folder for **Terraform execution**.
- Checks the **configuration files** for syntax errors.



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/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.
```

Screenshot should show **Terraform initialization success**.

2. Apply Terraform Configuration

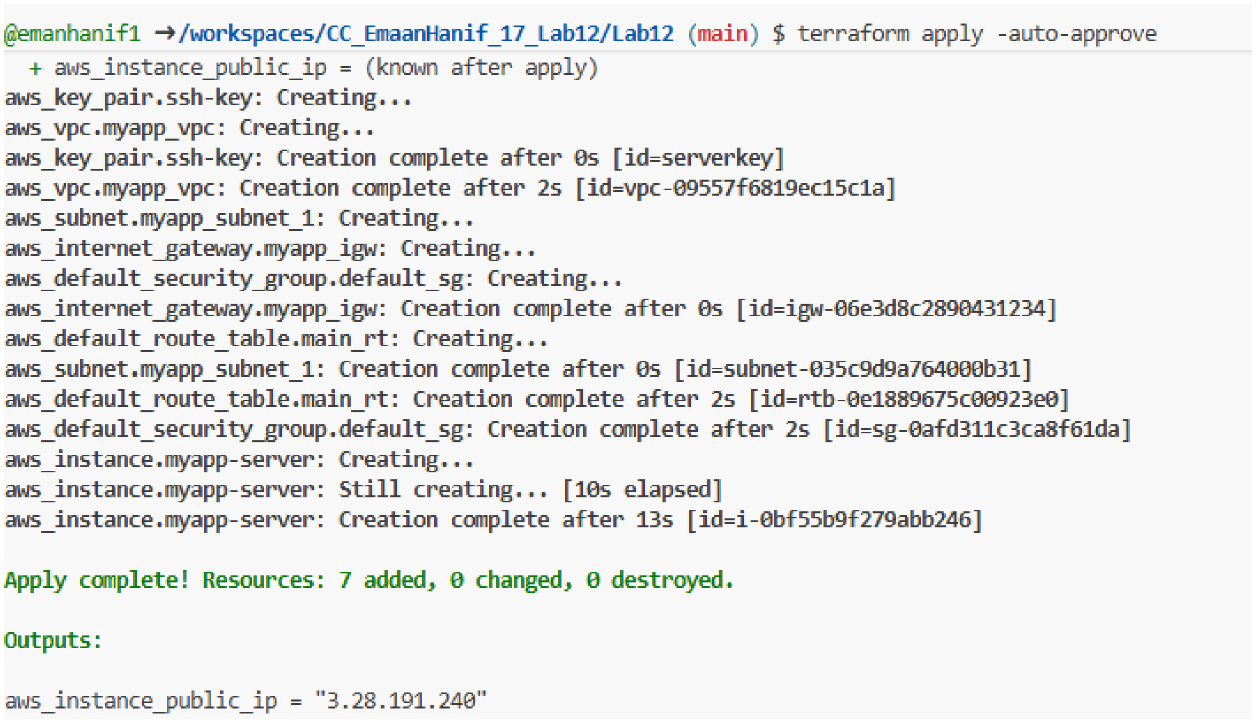
Command:

terraform apply -auto-approve

- -auto-approve → applies configuration **without asking confirmation**.

What it does:

- ❖ Terraform reads all .tf files (main.tf, variables.tf, etc.)
- ❖ Creates **all resources on AWS**:
    - o VPC, Subnet, Internet Gateway
    - o Security Group, Key Pair
    - o EC2 Instance with nginx installed
- ❖ Uses terraform.tfvars for **variable values**.



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
  + 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 0s [id=serverkey]
aws_vpc.myapp_vpc: Creation complete after 2s [id=vpc-09557f6819ec15c1a]
aws_subnet.myapp_subnet_1: Creating...
aws_internet_gateway.myapp_igw: Creating...
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 0s [id=igw-06e3d8c2890431234]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 0s [id=subnet-035c9d9a764000b31]
aws_default_route_table.main_rt: Creation complete after 2s [id=rtb-0e1889675c00923e0]
aws_default_security_group.default_sg: Creation complete after 2s [id=sg-0afd311c3ca8f61da]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-0bf55b9f279abb246]

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

Outputs:

aws_instance_public_ip = "3.28.191.240"
```

Screenshot should show **resources being created and applied successfully**.

3. Display Terraform Output

Command:

terraform output

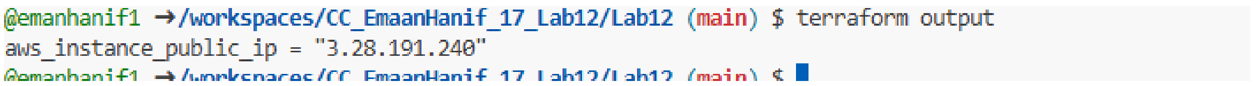
What it does:

- Shows **values defined in outputs.tf**
- Example: aws_instance_public_ip → public IP of EC2 instance.

📷 **Screenshot name:**

task1_terraform_output.png

Screenshot should show **EC2 public IP** returned by Terraform.

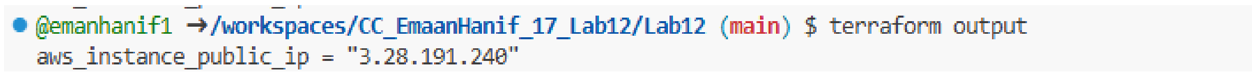


```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
aws_instance_public_ip = "3.28.191.240"
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

Test Nginx in Browser

1. **Get the public IP of your EC2 instance**

terraform output

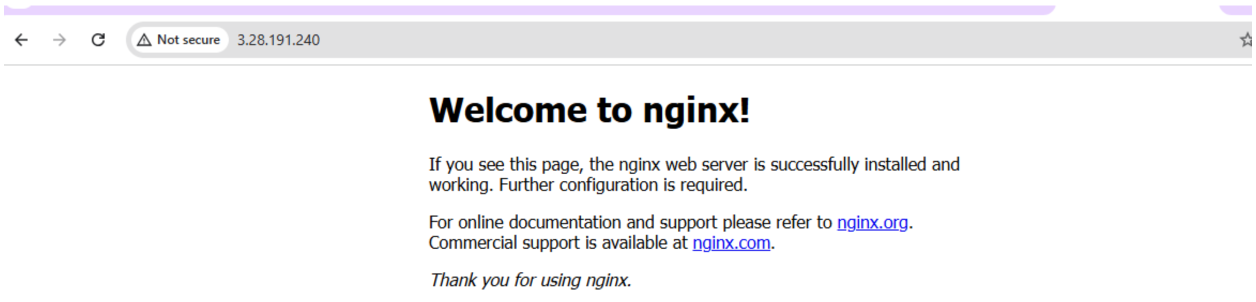


```
● @emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
  aws_instance_public_ip = "3.28.191.240"
```

2. **Open browser** → navigate to:

http:3.28.191.240

- Replace <public-ip> with the value from Terraform output.
- You should see the **default Nginx page**.



Not secure 3.28.191.240

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Screenshot should show the **Nginx welcome page** in your browser.

Destroy resources

1. **Command:**

terraform destroy

2. **Confirm destruction**

- Type yes when prompted.
- Terraform will **delete all AWS resources** it created:
  - EC2 instance
  - VPC, Subnet, Internet Gateway
  - Security Group
  - Key Pair

3. **Verify destruction**

- After completion, Terraform shows **"Destroy complete"**.
- You can also check AWS console → **no resources left from this Terraform run**.

Summary

1. terraform output → get public IP.
2. Open **browser** → test Nginx is running.
3. terraform destroy → clean up all resources.

Always destroy resources after testing to **avoid AWS charges**.

Task 2: Use Remote-Exec Provisioner

Step 1: Modify main.tf

What to do:

- Replace the **user_data line** in your aws_instance resource with a **remote-exec provisioner** and connection block.

Example aws_instance block:

```
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 "remote-exec" {
  inline = [
   "sudo yum update -y",
   "sudo yum install -y nginx",
   "sudo systemctl start nginx",
   "sudo systemctl enable nginx"
  ]
 }

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

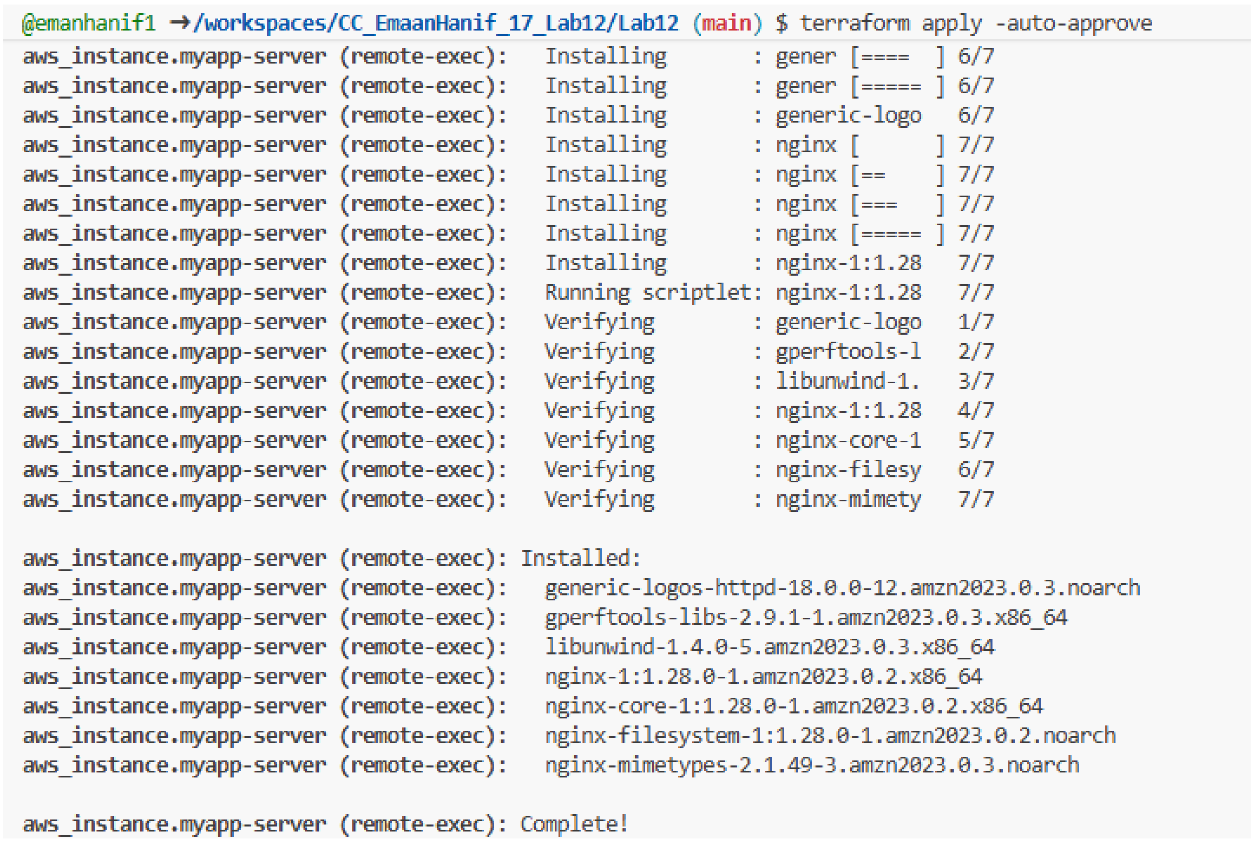
Apply the configuration

Command:

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

What it does:

- Terraform creates the EC2 instance (if not already created).
- Uses **SSH connection** to run commands in inline array:
  - Updates OS packages
  - Installs nginx
  - Starts and enables nginx service



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
aws_instance.myapp-server (remote-exec):   Installing       : gener [====   ] 6/7
aws_instance.myapp-server (remote-exec):   Installing       : gener [===== ] 6/7
aws_instance.myapp-server (remote-exec):   Installing       : generic-logo   6/7
aws_instance.myapp-server (remote-exec):   Installing       : nginx [       ] 7/7
aws_instance.myapp-server (remote-exec):   Installing       : nginx [==     ] 7/7
aws_instance.myapp-server (remote-exec):   Installing       : nginx [===    ] 7/7
aws_instance.myapp-server (remote-exec):   Installing       : nginx [===== ] 7/7
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!
```

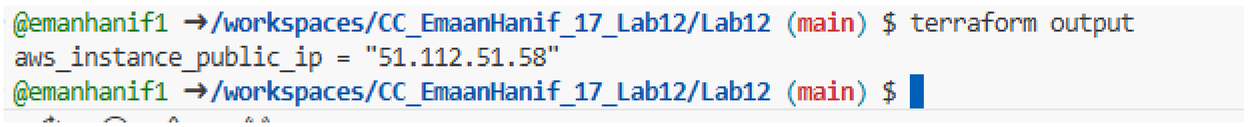
task2_terraform_apply.png

Screenshot should show **remote-exec commands executing successfully**.

Step 3: Display Terraform Output

Command:
terraform output

- Shows **EC2 public IP** just like Task 1.



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
aws_instance_public_ip = "51.112.51.58"
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

📷 **Screenshot name:**

task2_terraform_output.png

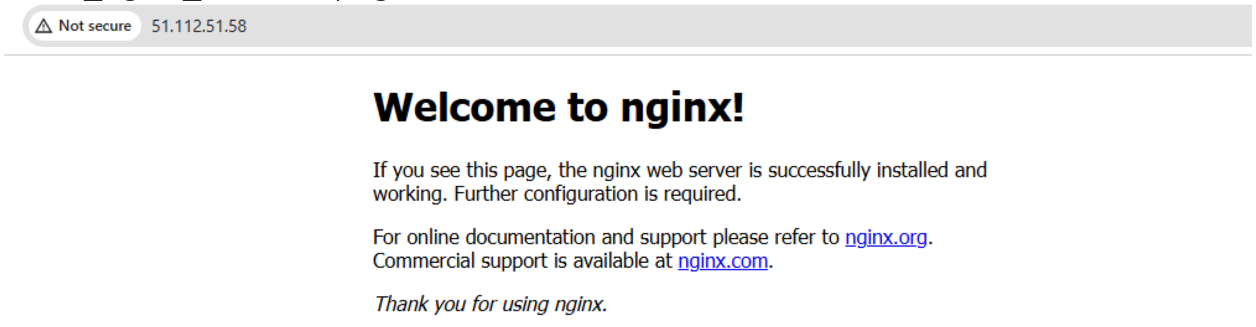
Step 4: Test Nginx in Browser

1. Open browser → navigate to:

- http://51.112.51.58
- Should display **Nginx default page**.

📷 Screenshot name:

task2_nginx_browser.png



Not secure 51.112.51.58

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Explanation

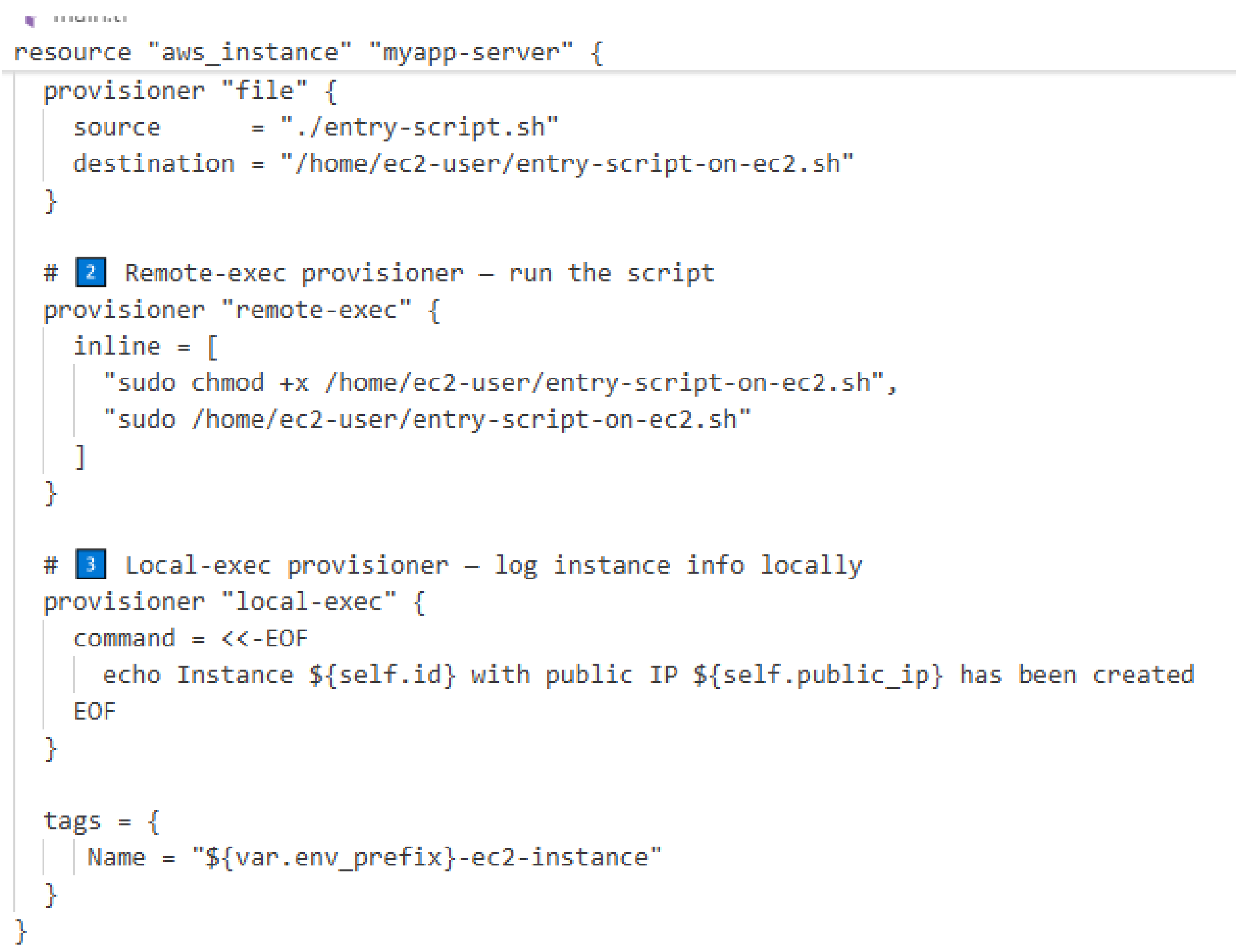
1. **remote-exec vs user_data**
   - user_data runs **only once at instance launch**
   - remote-exec runs via **SSH from Terraform**, can run on existing instances
2. **connection block**
   - Defines **how Terraform connects** to EC2 via SSH
   - Uses your private key and instance public IP
3. **inline commands**
   - Executes commands **directly on the EC2 instance**
   - Same as running commands manually after SSH
4. **Benefit**
   - More flexible → can **update or configure existing instance**

Task3- Use file and local-exec provisioners

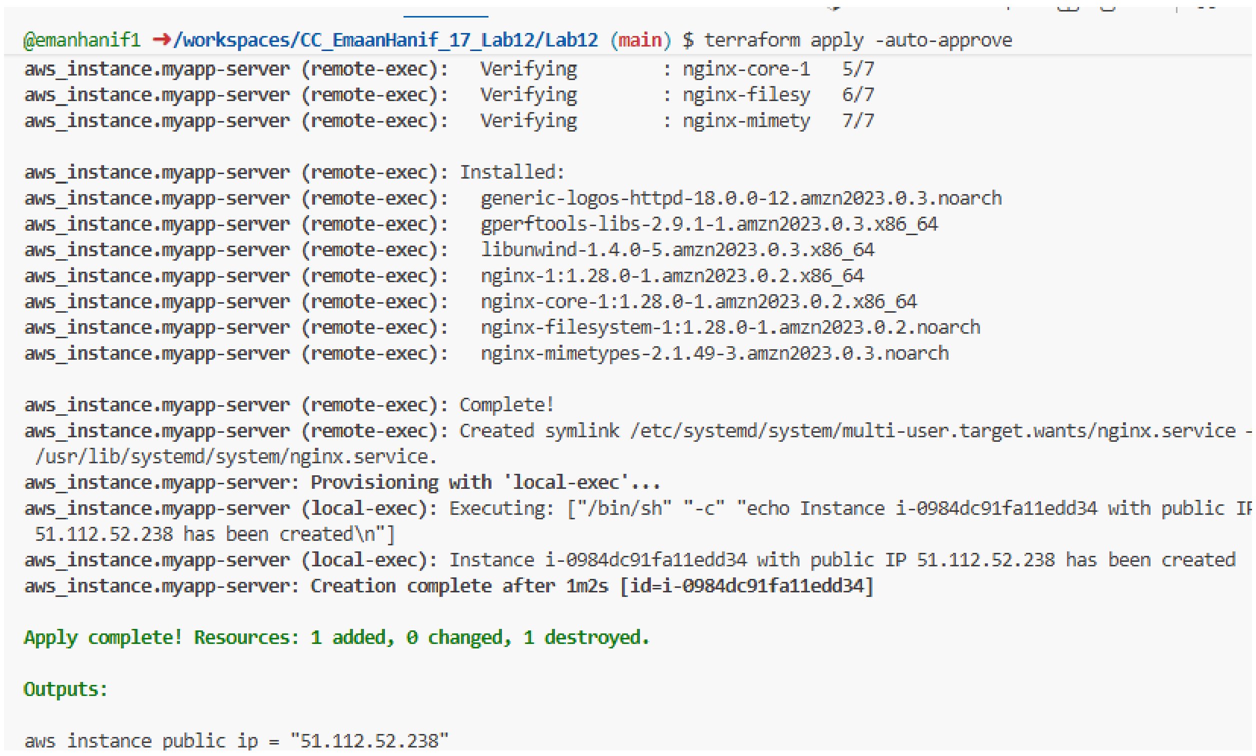
Modify aws_instance resource

Replace the previous remote-exec block with **all three provisioners**:

- **file** → Copies entry-script.sh from your machine to the EC2 instance.
- **remote-exec** → Runs the script on EC2 (install nginx, start service).
- **local-exec** → Prints/logs the EC2 instance ID and public IP **on your local machine**.



```
resource "aws_instance" "myapp-server" {
  provisioner "file" {
    source      = "./entry-script.sh"
    destination = "/home/ec2-user/entry-script-on-ec2.sh"
  }

  # 2️⃣ Remote-exec provisioner – run the script
  provisioner "remote-exec" {
    inline = [
      "sudo chmod +x /home/ec2-user/entry-script-on-ec2.sh",
      "sudo /home/ec2-user/entry-script-on-ec2.sh"
    ]
  }

  # 3️⃣ Local-exec provisioner – log instance info locally
  provisioner "local-exec" {
    command = <<-EOF
      echo Instance ${self.id} with public IP ${self.public_ip} has been created
    EOF
  }

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



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
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-0984dc91fa11edd34 with public IP
 51.112.52.238 has been created\n"]
aws_instance.myapp-server (local-exec): Instance i-0984dc91fa11edd34 with public IP 51.112.52.238 has been created
aws_instance.myapp-server: Creation complete after 1m2s [id=i-0984dc91fa11edd34]

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

Outputs:

aws instance public ip = "51.112.52.238"
```

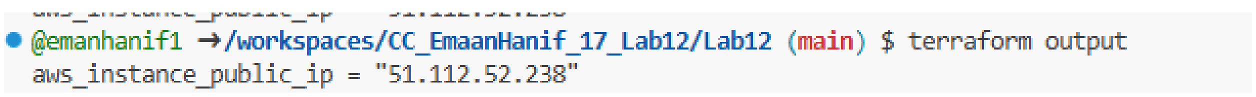
- Terraform created and provisioned all defined resources automatically without prompting for approval.

- Monitored the output to ensure all resources, including VPC, subnet, security group, key pair, internet gateway, and EC2 instance, were successfully created.

Display output

terraform output

- Shows **public IP** of the EC2 instance.



```
● @emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
  aws_instance_public_ip = "51.112.52.238"
```

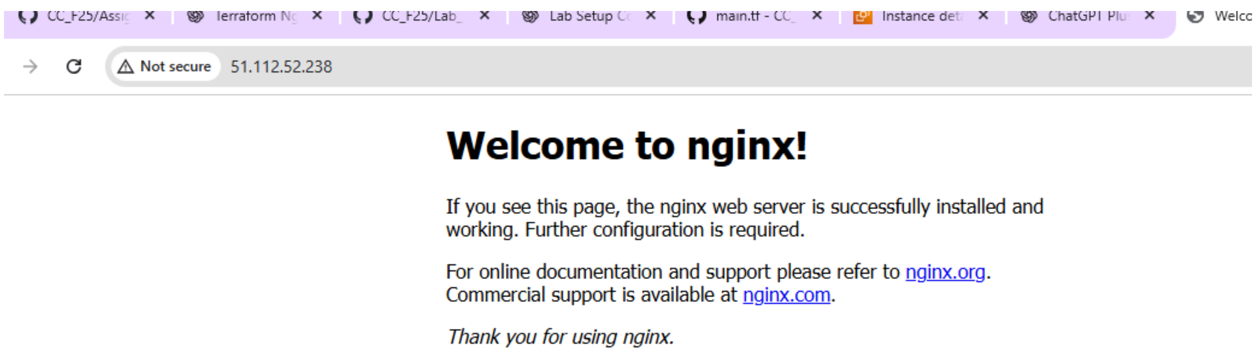
Confirmed that the EC2 instance was successfully provisioned and accessible via the retrieved public IP.

Test Nginx

- Open browser:

http://<public-ip>

- Replace <public-ip> with the 51.112.52.238
- You should see the **Nginx default page**.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

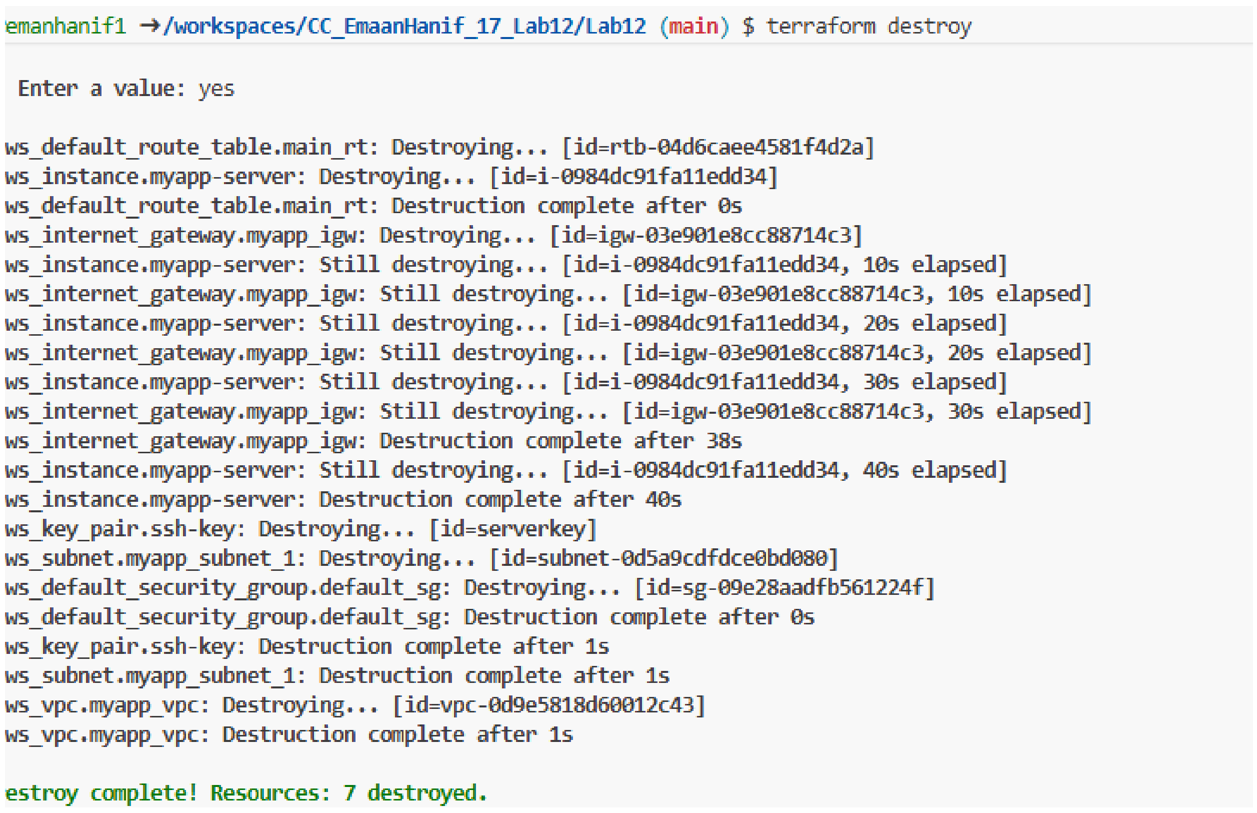
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Destroy resources

terraform destroy

- Type yes when prompted.
- Deletes all AWS resources created in this task.



```
emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform destroy

 Enter a value: yes

ws_default_route_table.main_rt: Destroying... [id=rtb-04d6caee4581f4d2a]
ws_instance.myapp-server: Destroying... [id=i-0984dc91fa11edd34]
ws_default_route_table.main_rt: Destruction complete after 0s
ws_internet_gateway.myapp_igw: Destroying... [id=igw-03e901e8cc88714c3]
ws_instance.myapp-server: Still destroying... [id=i-0984dc91fa11edd34, 10s elapsed]
ws_internet_gateway.myapp_igw: Still destroying... [id=igw-03e901e8cc88714c3, 10s elapsed]
ws_instance.myapp-server: Still destroying... [id=i-0984dc91fa11edd34, 20s elapsed]
ws_internet_gateway.myapp_igw: Still destroying... [id=igw-03e901e8cc88714c3, 20s elapsed]
ws_instance.myapp-server: Still destroying... [id=i-0984dc91fa11edd34, 30s elapsed]
ws_internet_gateway.myapp_igw: Still destroying... [id=igw-03e901e8cc88714c3, 30s elapsed]
ws_internet_gateway.myapp_igw: Destruction complete after 38s
ws_instance.myapp-server: Still destroying... [id=i-0984dc91fa11edd34, 40s elapsed]
ws_instance.myapp-server: Destruction complete after 40s
ws_key_pair.ssh-key: Destroying... [id=serverkey]
ws_subnet.myapp_subnet_1: Destroying... [id=subnet-0d5a9cdfdce0bd080]
ws_default_security_group.default_sg: Destroying... [id=sg-09e28aadfb561224f]
ws_default_security_group.default_sg: Destruction complete after 0s
ws_key_pair.ssh-key: Destruction complete after 1s
ws_subnet.myapp_subnet_1: Destruction complete after 1s
ws_vpc.myapp_vpc: Destroying... [id=vpc-0d9e5818d60012c43]
ws_vpc.myapp_vpc: Destruction complete after 1s

estroy complete! Resources: 7 destroyed.
```

Restore user_data in main.tf

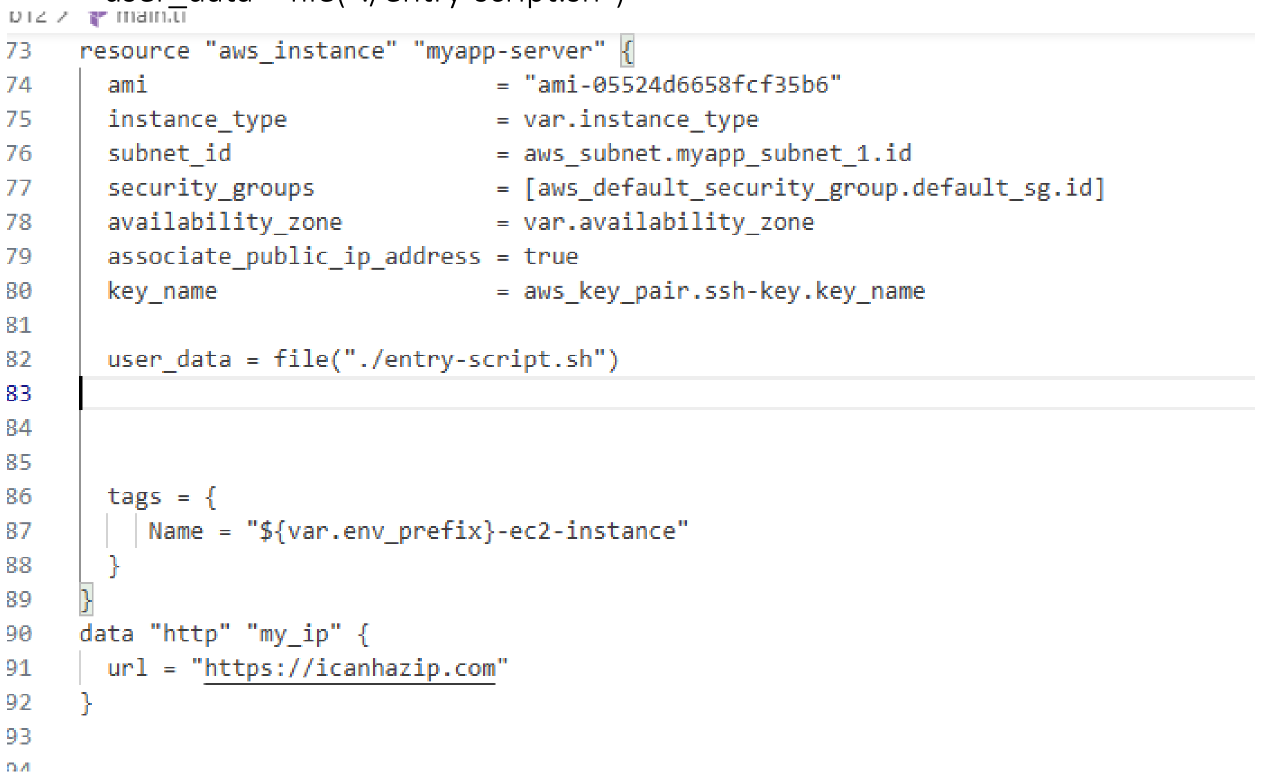
After using **file, remote-exec, and local-exec** provisioners, the last step is to **simplify your EC2 resource** back to the original user_data approach.

Edit main.tf

- Open main.tf.
- **Remove these blocks** from aws_instance:

- connection { … }
-
- provisioner "file" { … }
-
- provisioner "remote-exec" { … }

- provisioner "local-exec" { … }
- **Replace them with**:
- user_data = file("./entry-script.sh")



```
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

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



  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
}
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

```

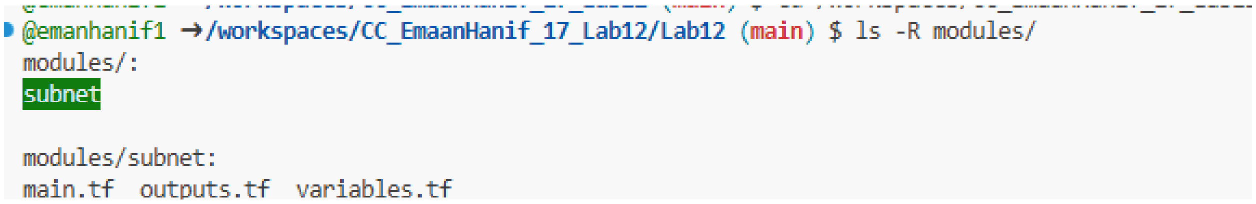
- It should clearly show user_data restored and **no connection/provisioner blocks**.

Task 4 — Create Terraform Subnet Module

The goal: **Make a reusable subnet module** so your infrastructure code is organized and can be reused for multiple subnets.

Create module directory structure

In your Lab12 workspace:



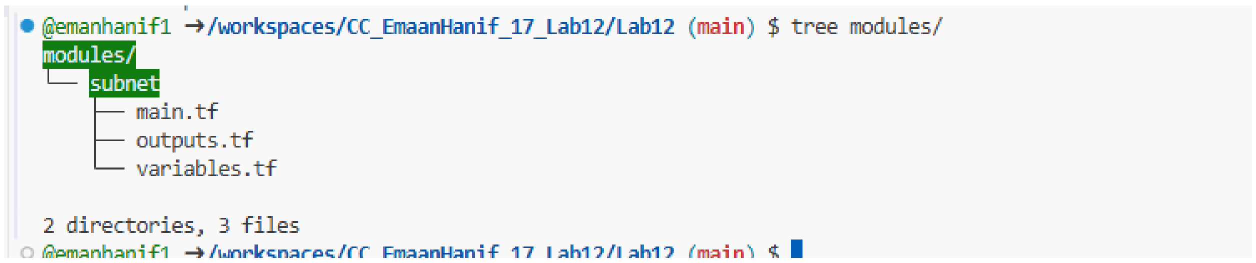
```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ls -R modules/
modules/:
subnet

modules/subnet:
main.tf  outputs.tf  variables.tf
```

- modules/subnet/main.tf → resource definitions (subnet creation).
- modules/subnet/variables.tf → variables the module accepts.
- modules/subnet/outputs.tf → outputs the module returns (e.g., subnet ID)

Verify structure

Or, if tree is installed:

tree modules/



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ tree modules/
modules/
└── subnet
    ├── main.tf
    ├── outputs.tf
    └── variables.tf

2 directories, 3 files
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

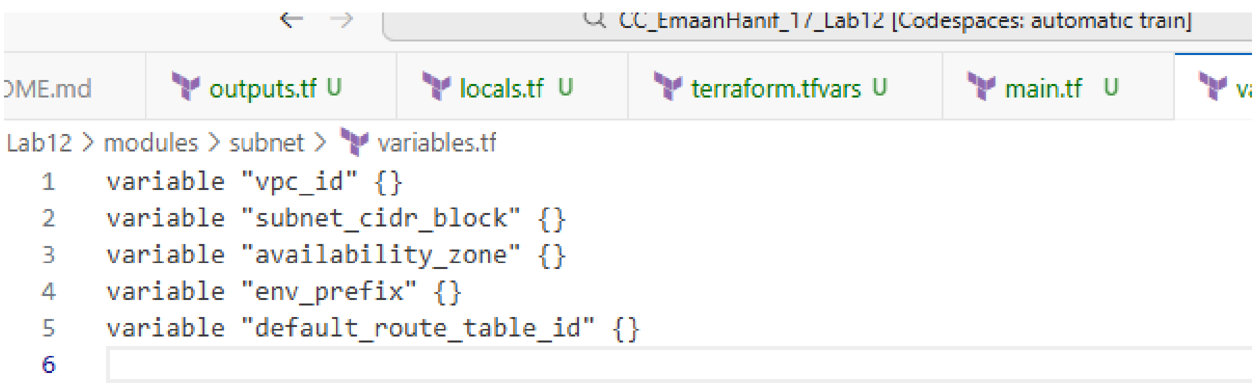
- Screenshot required:

task4_module_structure.png

Why use a module?

- **Reusability:** You can create multiple subnets without repeating code.
- **Organization:** Keeps main.tf clean.
- **Maintainability:** Any change to subnet logic is done in one place.

2) Create `modules/subnet/variables.tf`:

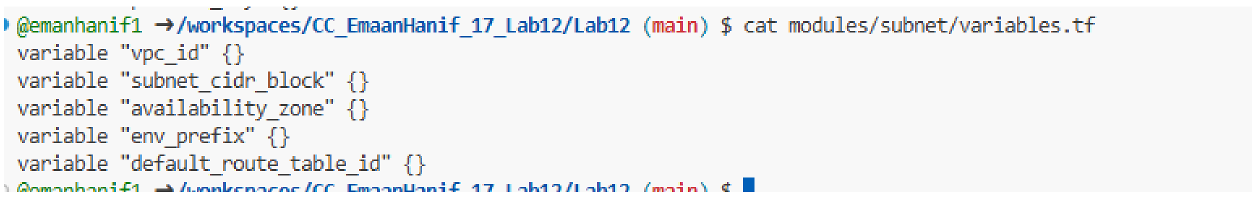

CC_EmaanHanif_17_Lab12 [Codespaces: automatic train]
DME.md
outputs.tf U
locals.tf U
terraform.tfvars U
main.tf U
Lab12 > modules > subnet > variables.tf

```
variable "vpc_id" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "default_route_table_id" {}
```

variables.tf **defines all the inputs** that your subnet module will need.

- Instead of hardcoding values, you **pass them from the root main.tf** when you call the module.

Variable	Purpose
vpc_id	The VPC where the subnet will be created
subnet_cidr_block	CIDR block for this subnet (like 10.0.10.0/24)
availability_zone	Which AZ the subnet will be in
env_prefix	Prefix for naming the subnet (like dev or prod)
default_route_table_id	ID of the default route table to attach routes



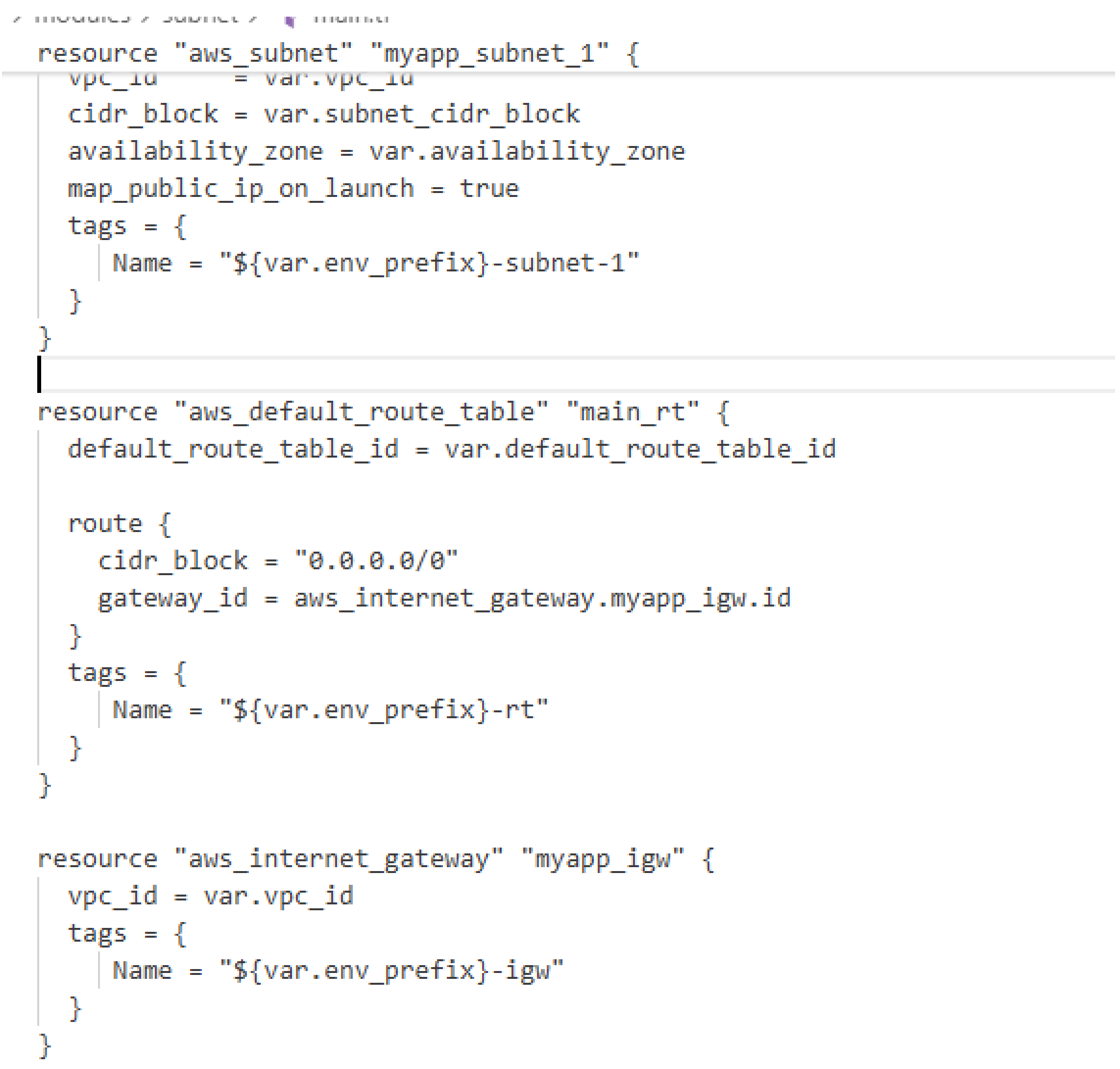
```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ cat modules/subnet/variables.tf
variable "vpc_id" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "default_route_table_id" {}
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

You can **view file content from anywhere** using cat or echo like this\

Purpose:

- These variables make the module **reusable** for different VPCs, CIDRs, and environments.
- The root module will pass values for these variables.

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

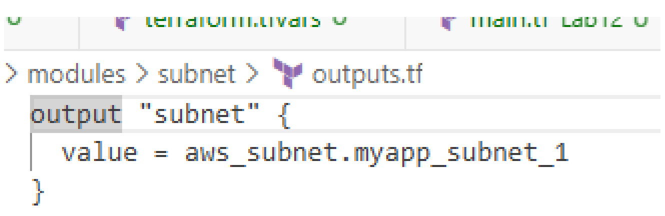


```
resource "aws_subnet" "myapp_subnet_1" {
  vpc_id      = var.vpc_id
  cidr_block = var.subnet_cidr_block
  availability_zone = var.availability_zone
  map_public_ip_on_launch = true
  tags = {
    Name = "${var.env_prefix}-subnet-1"
  }
}

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"
  }
}

resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = var.vpc_id
  tags = {
    Name = "${var.env_prefix}-igw"
  }
}
```

Creates a **subnet**, **internet gateway**, and **route table** inside the module.

Module handles **network setup** automatically.

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

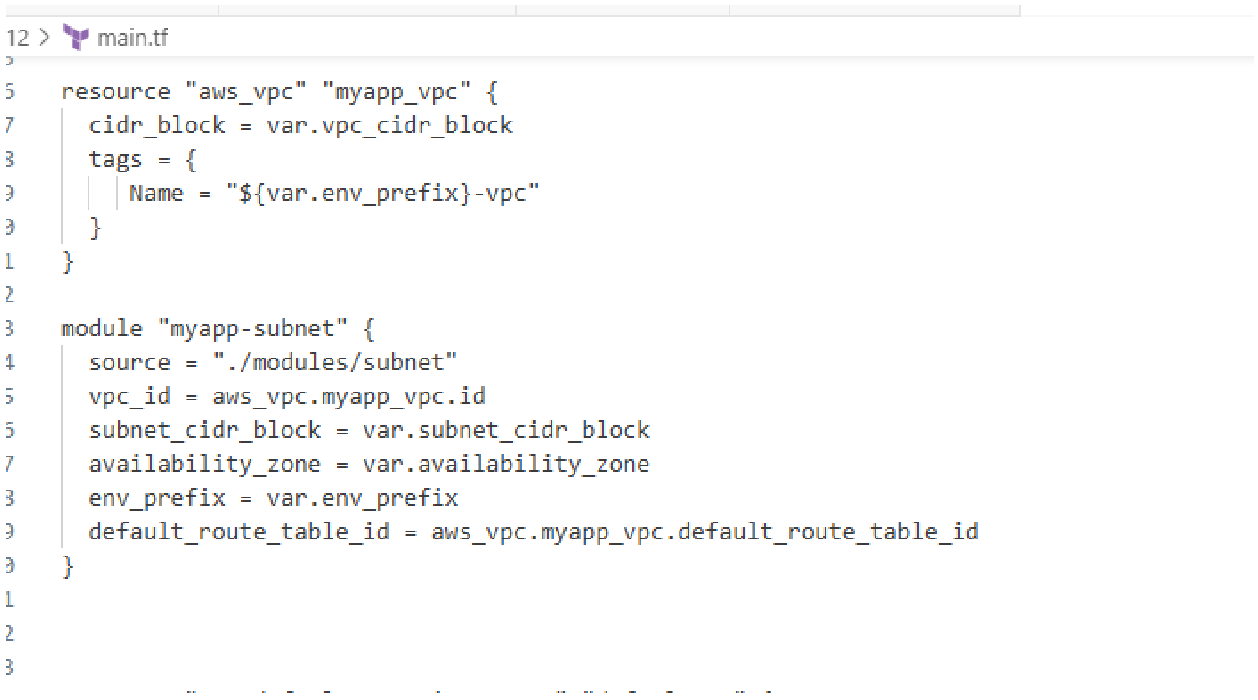


```
> modules > subnet > outputs.tf
output "subnet" {
  value = aws_subnet.myapp_subnet_1
}
```

Exposes the subnet ID so **root main.tf can use it** for EC2 instances.

Update Root main.tf

- **Remove** subnet, route table, and IGW resources from root.
- **Add module call:**



```
12 > main.tf
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

module "myapp-subnet" {
  source = "./modules/subnet"
  vpc_id = aws_vpc.myapp_vpc.id
  subnet_cidr_block = var.subnet_cidr_block
  availability_zone = var.availability_zone
  env_prefix = var.env_prefix
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id
}
```

Update EC2 instance to use module output:

```
subnet_id                    =  module.myapp-subnet.subnet.id
```

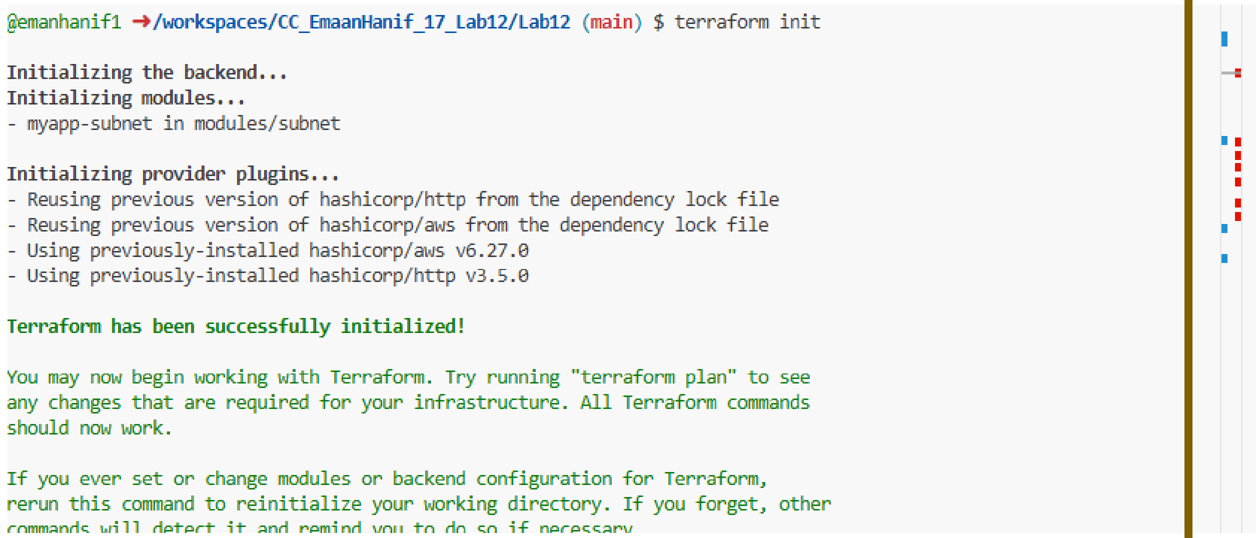
Purpose:

- Root main.tf is now **clean and modular**.
- EC2 instance automatically uses subnet from module.

• Screenshot: task4_main_tf_with_module.png

Initialize Terraform

terraform init



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform init

Initializing the backend...
Initializing modules...
- myapp-subnet in modules/subnet

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/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.
```

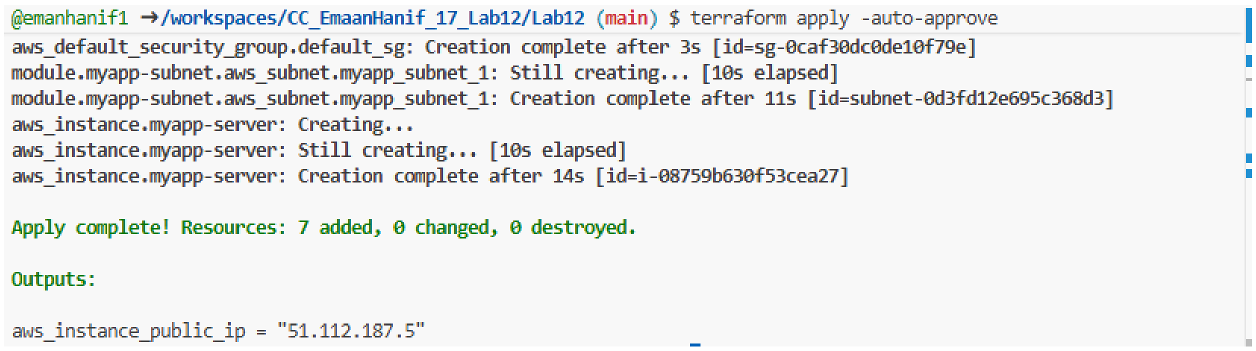
Downloads modules and providers.

- Must run **before apply** after adding a new module.

• Screenshot: task4_terraform_init.png

Apply the Configuration

terraform apply -auto-approve



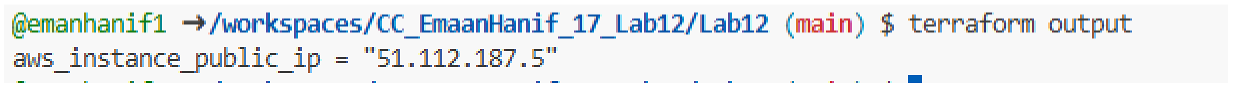
```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-0caf30dc0de10f79e]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Still creating... [10s elapsed]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Creation complete after 11s [id=subnet-0d3fd12e695c368d3]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [10s elapsed]
aws_instance.myapp-server: Creation complete after 14s [id=i-08759b630f53cea27]

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

Outputs:

aws_instance_public_ip = "51.112.187.5"
```

- Terraform creates:
  - VPC
  - Subnet, IGW, route table (via module)
  - Security group
  - EC2 instance
- Screenshot: task4_terraform_apply.png

View Outputs

terraform output

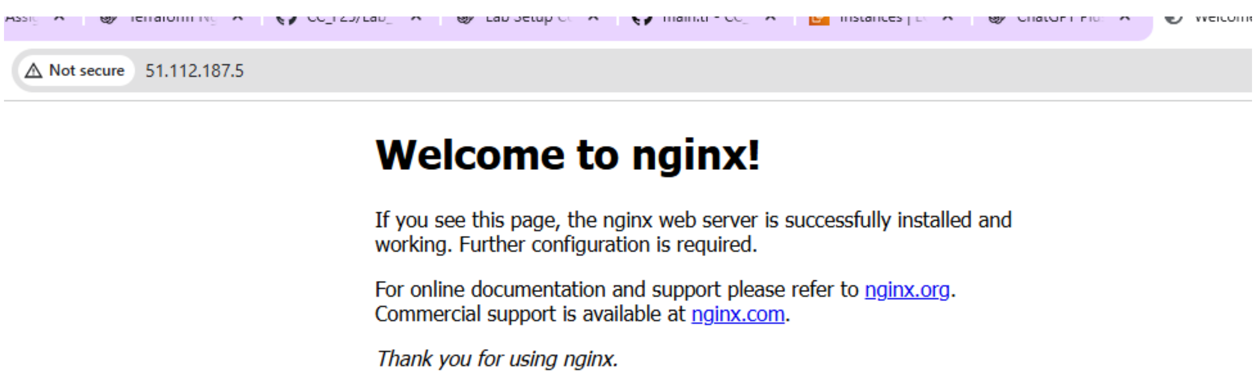
- Shows the **public IP** of your EC2 instance.



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
aws_instance_public_ip = "51.112.187.5"
```

9. Test nginx in browser:

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


Not secure 51.112.187.5

Welcome to nginx!

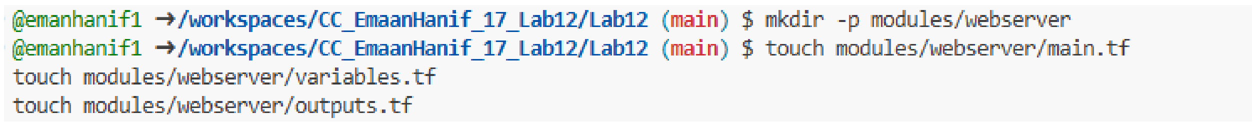
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Task5- Create webserver module

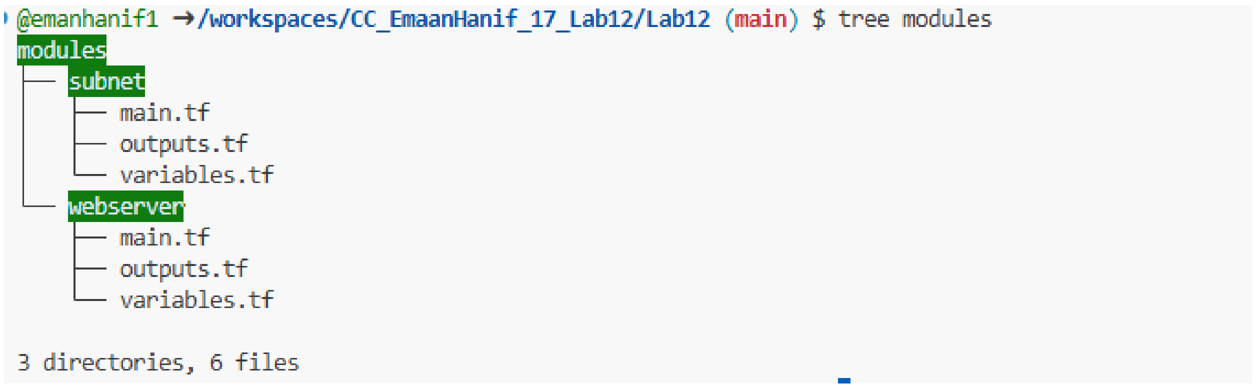
Create Webserver Module Directory



```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ mkdir -p modules/webserver
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ touch modules/webserver/main.tf
touch modules/webserver/variables.tf
touch modules/webserver/outputs.tf
```

Purpose

- modules/webserver/ → module folder
- main.tf → EC2, security group, key pair
- variables.tf → inputs from root module
- outputs.tf → values returned to root (like public IP)



```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ tree modules
modules
├── subnet
│   ├── main.tf
│   ├── outputs.tf
│   └── variables.tf
└── webserver
    ├── main.tf
    ├── outputs.tf
    └── variables.tf

3 directories, 6 files
```

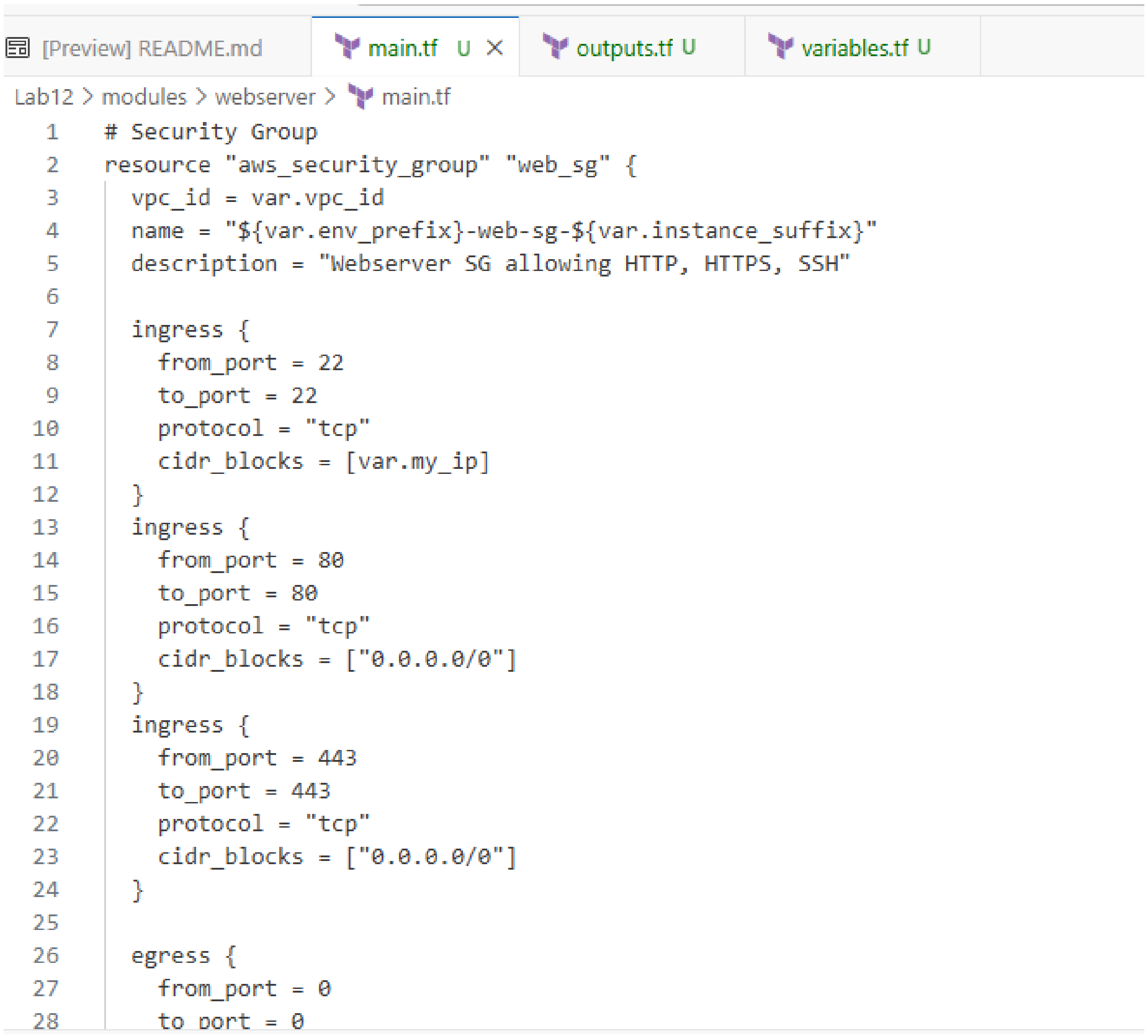
Define Module Variables

modules/webserver/variables.tf

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

Step 3 — Create EC2 + SG + Key in Module

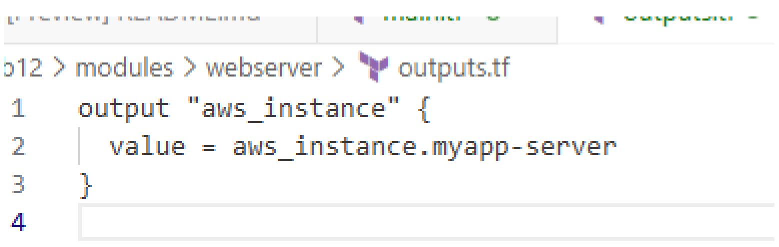
modules/webserver/main.tf:



```
# Security Group
resource "aws_security_group" "web_sg" {
  vpc_id = var.vpc_id
  name = "${var.env_prefix}-web-sg-${var.instance_suffix}"
  description = "Webserver SG allowing HTTP, HTTPS, SSH"

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

  egress {
    from_port = 0
    to_port = 0
```

Step 4 — Module Outputs

modules/webserver/outputs.tf:



012 > modules > webserver > outputs.tf

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

In the root main.tf file:

Remove the following resources (if they exist):

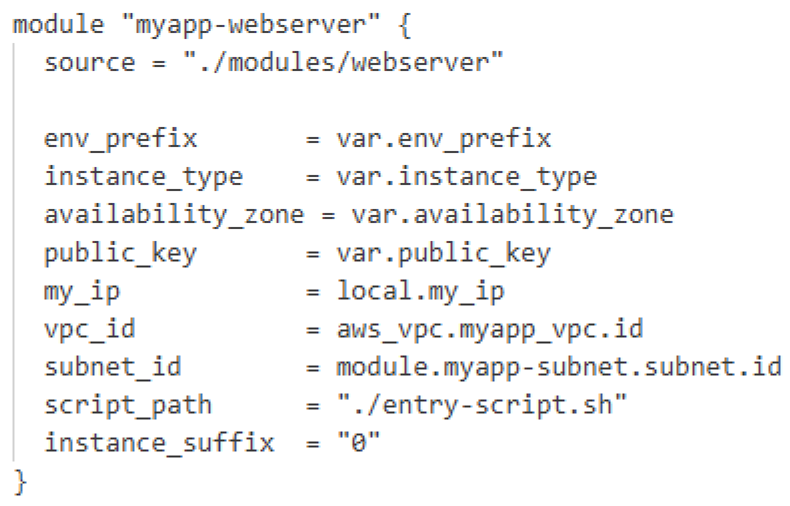
- aws_security_group
- aws_key_pair
- aws_instance

These resources are now **handled inside the webserver module**, so they should **not** be in the root file anymore.

Add the webserver module block

Paste the following code into your **root main.tf**:

```
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            = aws_vpc.myapp_vpc.id
  subnet_id         = module.myapp-subnet.subnet.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
```



```
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            = aws_vpc.myapp_vpc.id
  subnet_id         = module.myapp-subnet.subnet.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
```

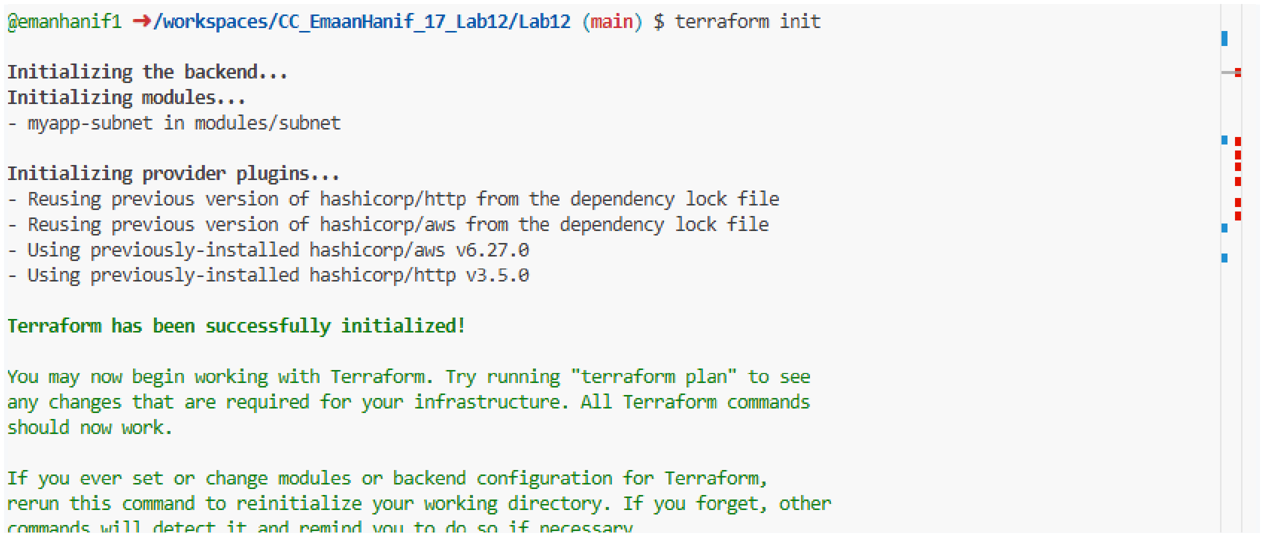
6. Update outputs.tf:

```
output "webserver_public_ip" {
  value = module.myapp-webserver.aws_instance.public_ip
}
```

- **Save screenshot as:** task5_outputs_updated.png — updated outputs.tf.

7. Initialize Terraform:

```
terraform init
```



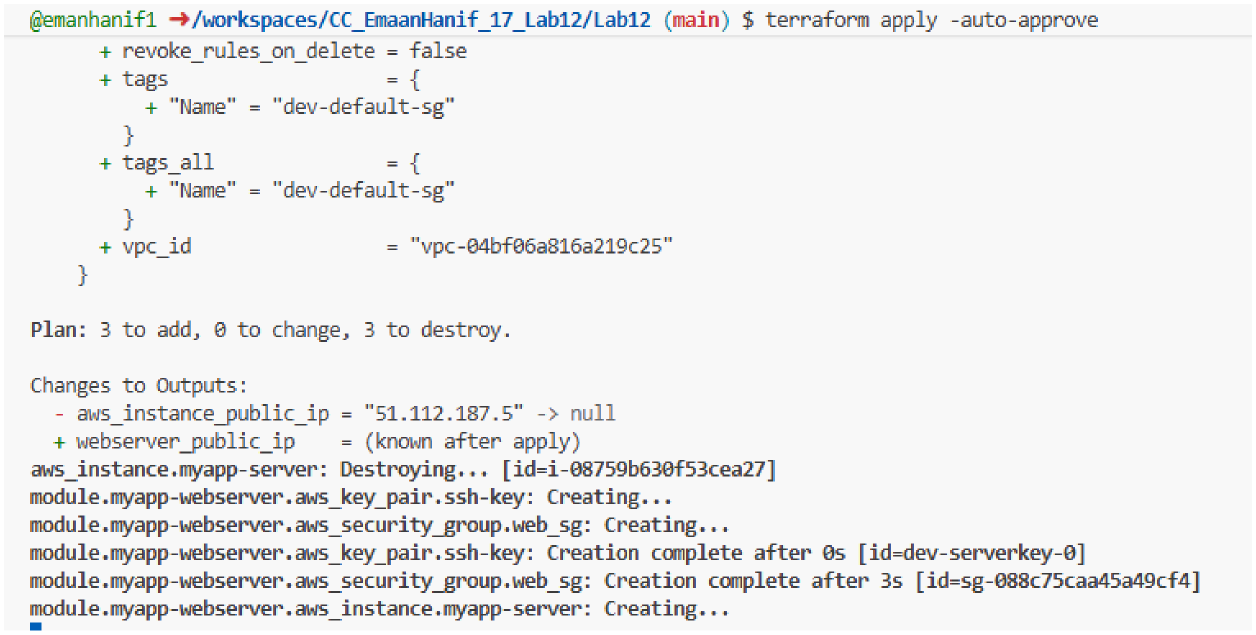
```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform init

Initializing the backend...
Initializing modules...
- myapp-subnet in modules/subnet

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/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.
```

- **Save screenshot as:** task5_terraform_init.png — terraform init output.

8. Apply the configuration:

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

- **Save screenshot as:** task5_terraform_apply.png — terraform apply output with webserver module.



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
      + revoke_rules_on_delete = false
      + tags                   = {
          + "Name" = "dev-default-sg"
        }
      + tags_all               = {
          + "Name" = "dev-default-sg"
        }
      + vpc_id                 = "vpc-04bf06a816a219c25"
    }

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

Changes to Outputs:
  - aws_instance_public_ip = "51.112.187.5" -> null
  + webserver_public_ip    = (known after apply)
aws_instance.myapp-server: Destroying... [id=i-08759b630f53cea27]
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 0s [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Creation complete after 3s [id=sg-088c75caa45a49cf4]
module.myapp-webserver.aws_instance.myapp-server: Creating...
```

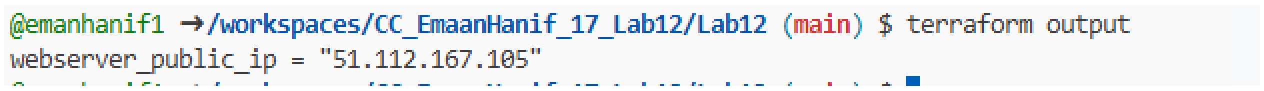
6. Update outputs.tf:

```
output "webserver_public_ip" {
  value = module.myapp-webserver.aws_instance.public_ip
}
```

9. Display the output:

```
terraform output
```

- **Save screenshot as:** task5_terraform_output.png — terraform output showing webserver public IP.

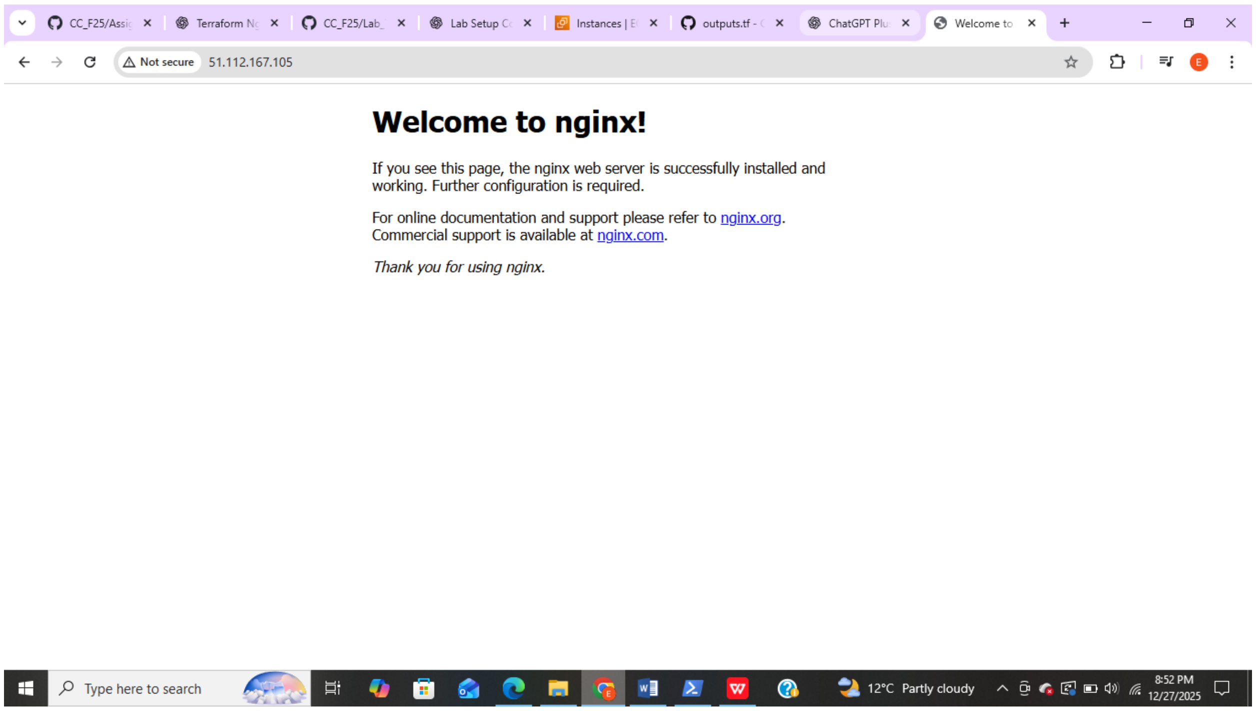


```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
webserver_public_ip = "51.112.167.105"
```

10. Test nginx in browser:

- Open browser and navigate to http://<public-ip>
- **Save screenshot as:** task5_nginx_browser.png — browser showing nginx default page.

.



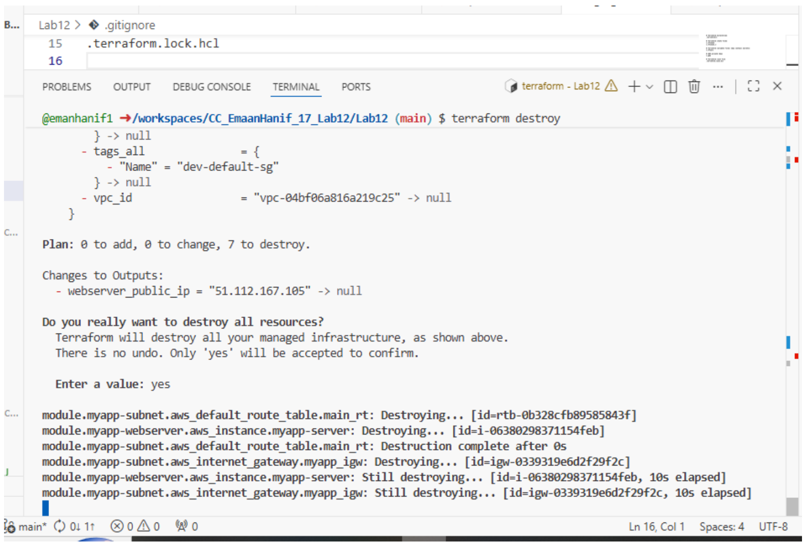
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Terraform destroy



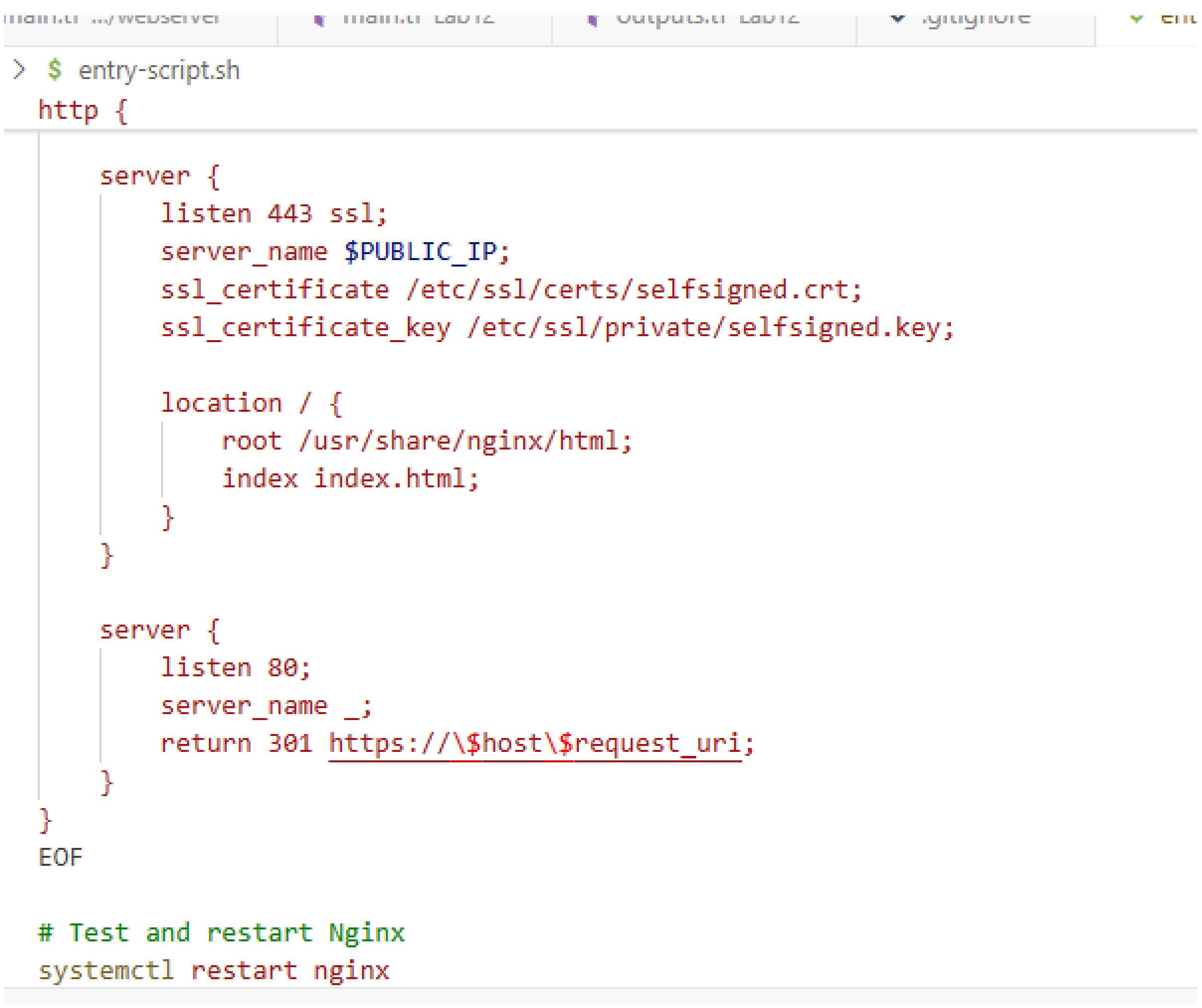
```
.terraform.lock.hcl
```

```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform destroy
        } -> null
      - tags_all                = {
          - "Name" = "dev-default-sg"
        } -> null
      - vpc_id                  = "vpc-04bf06a816a219c25" -> null
    }

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

Changes to Outputs:
  - webserver_public_ip = "51.112.167.105" -> 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-subnet.aws_default_route_table.main_rt: Destroying... [id=rtb-0b328cfb89585843f]
module.myapp-webserver.aws_instance.myapp-server: Destroying... [id=i-06380298371154feb]
module.myapp-subnet.aws_default_route_table.main_rt: Destruction complete after 0s
module.myapp-subnet.aws_internet_gateway.myapp_igw: Destroying... [id=igw-0339319e6d2f29f2c]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-06380298371154feb, 10s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0339319e6d2f29f2c, 10s elapsed]
```

Task 6 — Configure HTTPS with Self-Signed Certificates

Update entry-script.sh

Replace your old script or append the HTTPS configuration with the following **key points**:



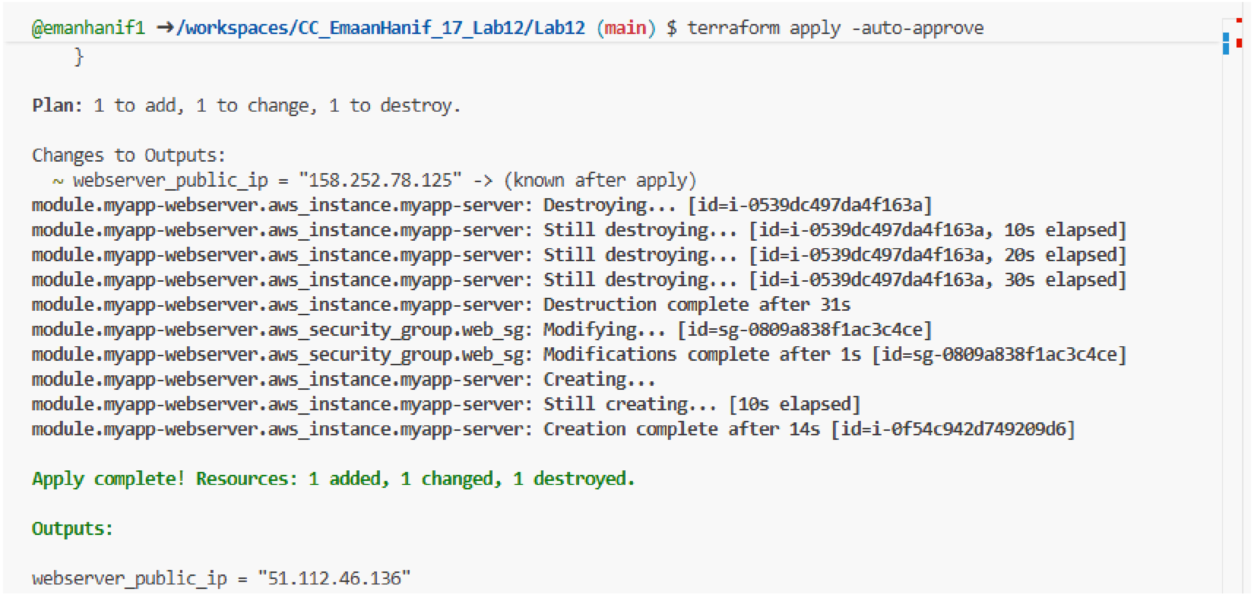
```
> $ entry-script.sh
  http {

      server {
          listen 443 ssl;
          server_name $PUBLIC_IP;
          ssl_certificate /etc/ssl/certs/selfsigned.crt;
          ssl_certificate_key /etc/ssl/private/selfsigned.key;

          location / {
              root /usr/share/nginx/html;
              index index.html;
          }
      }

      server {
          listen 80;
          server_name _;
          return 301 https://\$host\$request_uri;
      }
  }
  EOF

  # Test and restart Nginx
  systemctl restart nginx
```

- ❖ **IMDSv2 token** → securely fetch EC2 metadata (IP/hostname)
- ❖ **Self-signed cert** → HTTPS without buying a certificate

- **Dynamic CN / SAN** → certificate matches the public IP
- **Nginx config** →
- Port 443 for HTTPS
- Port 80 redirects to HTTPS
- Optional upstream backend for load balancing
- **systemctl restart nginx** → apply new SSL configuration

Step 1 — Apply Terraform configuration

terraform apply -auto-approve

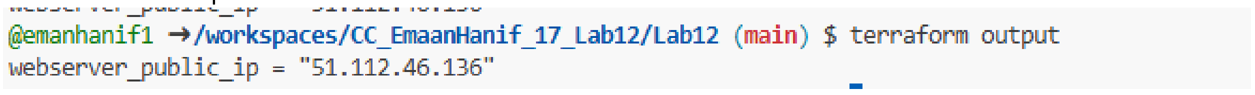


```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
    }

Plan: 1 to add, 1 to change, 1 to destroy.

Changes to Outputs:
  ~ webserver_public_ip = "158.252.78.125" -> (known after apply)
module.myapp-webserver.aws_instance.myapp-server: Destroying... [id=i-0539dc497da4f163a]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0539dc497da4f163a, 10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0539dc497da4f163a, 20s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0539dc497da4f163a, 30s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Destruction complete after 31s
module.myapp-webserver.aws_security_group.web_sg: Modifying... [id=sg-0809a838f1ac3c4ce]
module.myapp-webserver.aws_security_group.web_sg: Modifications complete after 1s [id=sg-0809a838f1ac3c4ce]
module.myapp-webserver.aws_instance.myapp-server: Creating...
module.myapp-webserver.aws_instance.myapp-server: Still creating... [10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 14s [id=i-0f54c942d749209d6]

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

Outputs:

webserver_public_ip = "51.112.46.136"
```

- This will create the EC2 instance with **HTTPS-ready Nginx**
- Screenshot: **task6_terraform_apply.png**

Step 2 — Display Terraform outputs

terraform output



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
webserver_public_ip = "51.112.46.136"
```

- Copy the **public IP** of the webserver
- Screenshot: **task6_terraform_output.png**

Step 3 — Test HTTPS in browser

1. Open browser
2. Navigate to:

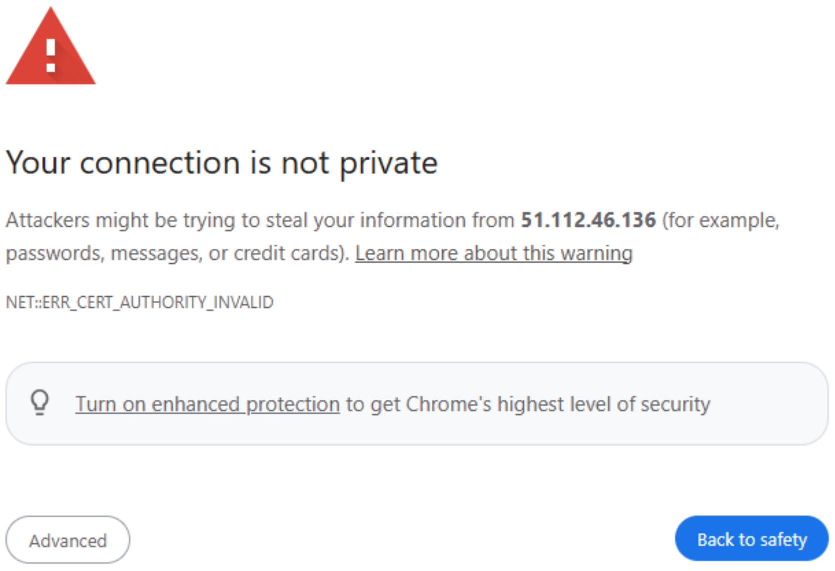
https://<public-ip>

3. Browser will show:

Warning: Potential Security Risk Ahead

4. Click **Advanced → Accept the Risk and Continue**

- Screenshot: **task6_browser_security_warning.png**



Your connection is not private

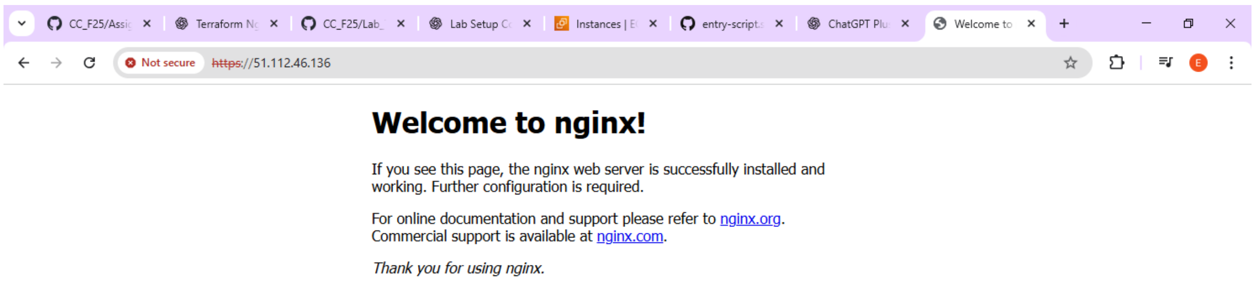
Attackers might be trying to steal your information from **51.112.46.136** (for example, passwords, messages, or credit cards). Learn more about this warning

NET::ERR_CERT_AUTHORITY_INVALID

Turn on enhanced protection to get Chrome's highest level of security

-

5. After accepting, Nginx default page over HTTPS will load


Not secure ~~https~~://51.112.46.136

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

- Screenshot: **task6_nginx_https_browser.png**

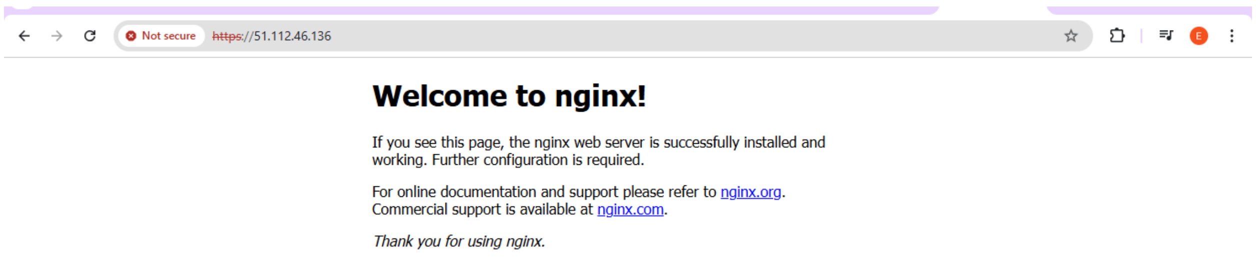
Step 4 — Verify HTTP to HTTPS redirect

1. Open browser
2. Navigate to:

http://<public-ip>

3. Confirm it **automatically redirects** to:

https://<public-ip>


Not secure ~~https~~://51.112.46.136

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

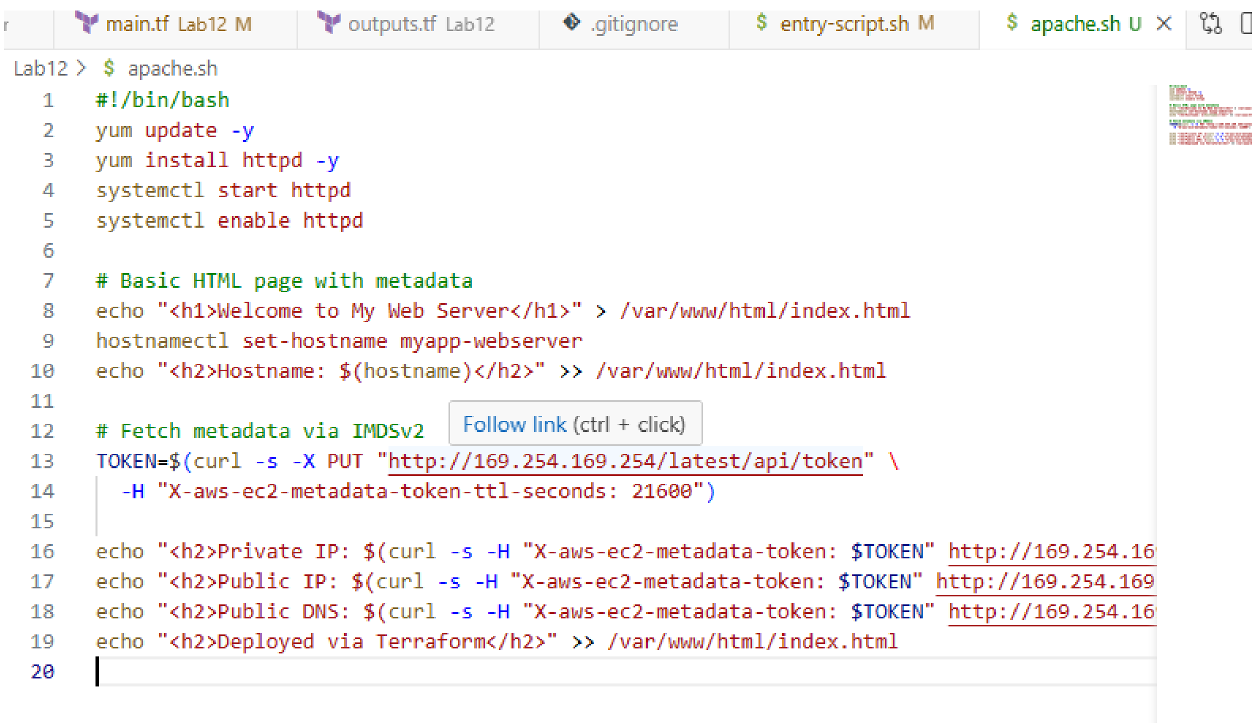
- Screenshot: **task6_http_redirect.png**

Task7-**Configure Nginx as reverse proxy**

Configure Nginx as Reverse Proxy

Step 1 — Create backend script (apache.sh)

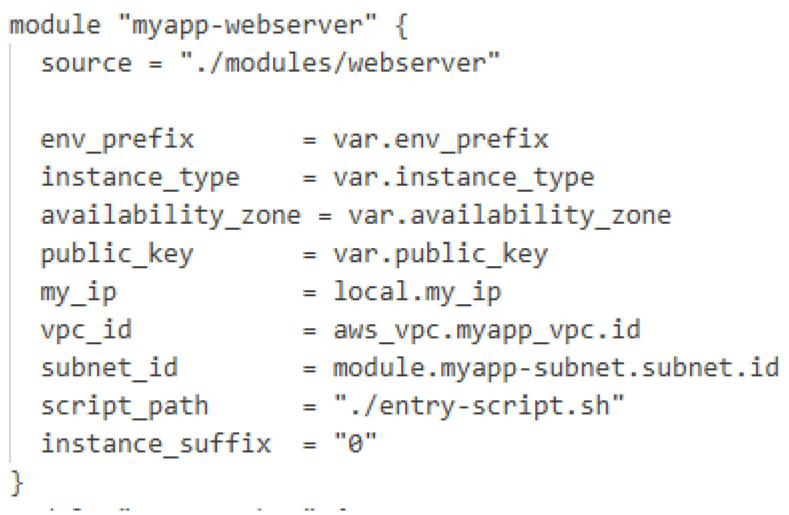
This script will configure a **backend web server** (Apache) with dynamic metadata info:



```
#!/bin/bash
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd

# Basic HTML page with metadata
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

# Fetch metadata via IMDSv2
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" http://169.254.16
echo "<h2>Public IP: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169
echo "<h2>Public DNS: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.16
echo "<h2>Deployed via Terraform</h2>" >> /var/www/html/index.html

```

Add backend web server module in main.tf

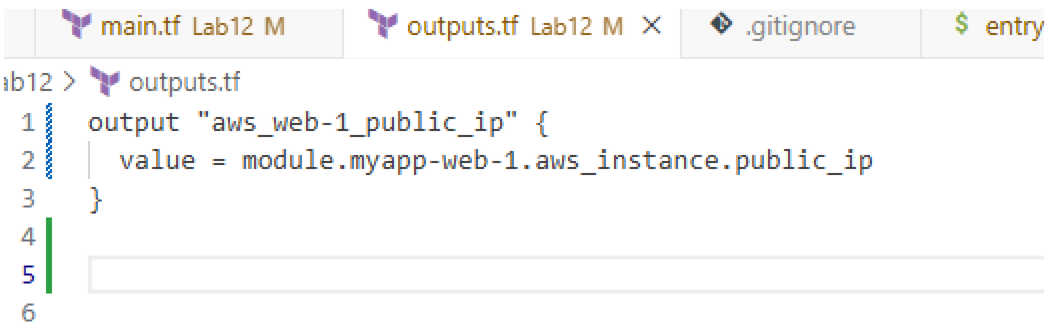
Add a **new module block** for myapp-web-1:



```
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            = aws_vpc.myapp_vpc.id
  subnet_id         = module.myapp-subnet.subnet.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
```

Update outputs.tf

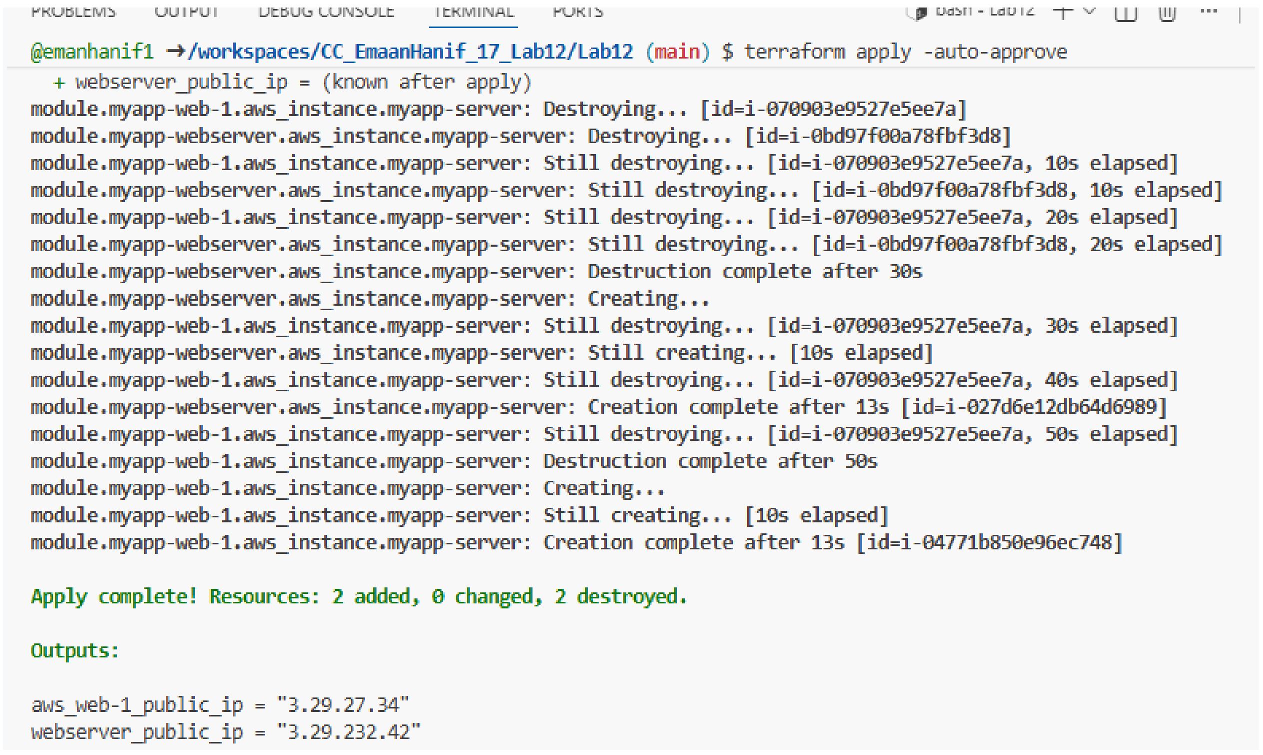
Expose the backend web server's public IP:



```
output "aws_web-1_public_ip" {
  value = module.myapp-web-1.aws_instance.public_ip
}
```

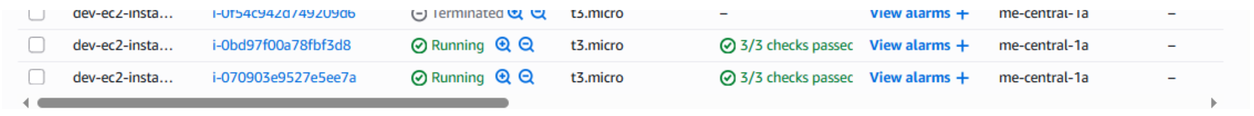
4. Apply the configuration:

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

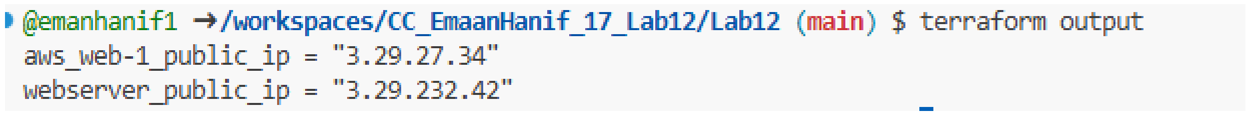


```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
  + webserver_public_ip = (known after apply)
module.myapp-web-1.aws_instance.myapp-server: Destroying... [id=i-070903e9527e5ee7a]
module.myapp-webserver.aws_instance.myapp-server: Destroying... [id=i-0bd97f00a78fbf3d8]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-070903e9527e5ee7a, 10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0bd97f00a78fbf3d8, 10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-070903e9527e5ee7a, 20s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0bd97f00a78fbf3d8, 20s elapsed]
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 destroying... [id=i-070903e9527e5ee7a, 30s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still creating... [10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-070903e9527e5ee7a, 40s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 13s [id=i-027d6e12db64d6989]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-070903e9527e5ee7a, 50s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Destruction complete after 50s
module.myapp-web-1.aws_instance.myapp-server: Creating...
module.myapp-web-1.aws_instance.myapp-server: Still creating... [10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Creation complete after 13s [id=i-04771b850e96ec748]

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

Outputs:

aws_web-1_public_ip = "3.29.27.34"
webserver_public_ip = "3.29.232.42"
```

- **Save screenshot as:** task7_terraform_apply.png — terraform apply output showing both instances created.

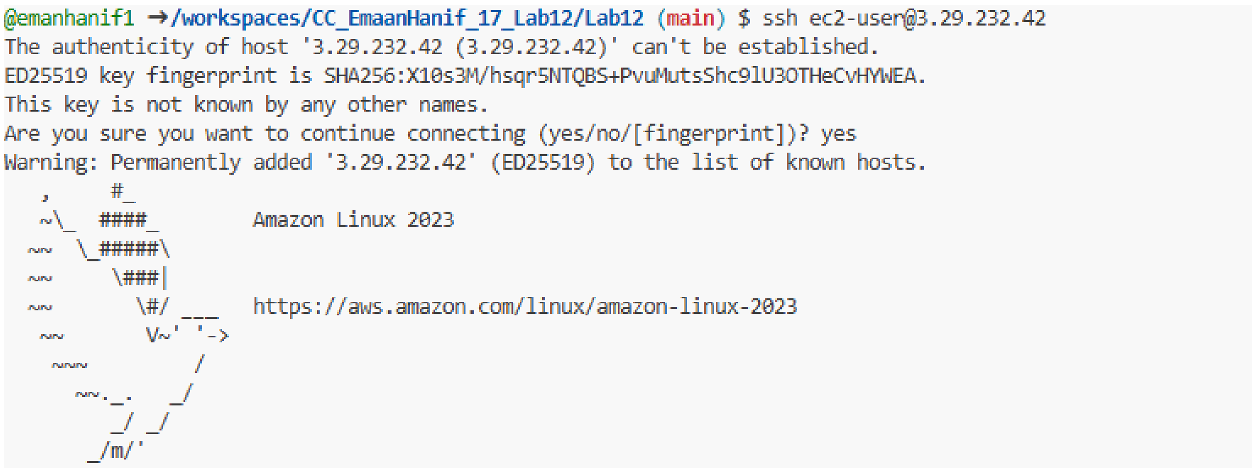


☐	dev-ec2-insta...	i-0f54c942d749209d6	Terminated	t3.micro	–	View alarms +	me-central-1a	–
☐	dev-ec2-insta...	i-0bd97f00a78fbf3d8	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	me-central-1a	–
☐	dev-ec2-insta...	i-070903e9527e5ee7a	✓ Running	t3.micro	✓ 3/3 checks passec	View alarms +	me-central-1a	–



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform output
aws_web-1_public_ip = "3.29.27.34"
webserver_public_ip = "3.29.232.42"
```

SSH into the main webserver



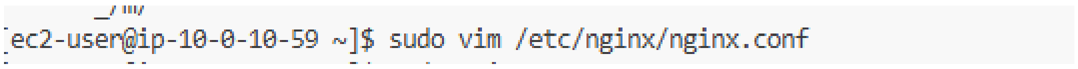
```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ssh ec2-user@3.29.232.42
The authenticity of host '3.29.232.42 (3.29.232.42)' can't be established.
ED25519 key fingerprint is SHA256:X10s3M/hsqr5NTQBS+PvuMutsShc9lU3OTHeCvHYWEA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.29.232.42' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

— **Verify connection**

Once connected, you should see something like:

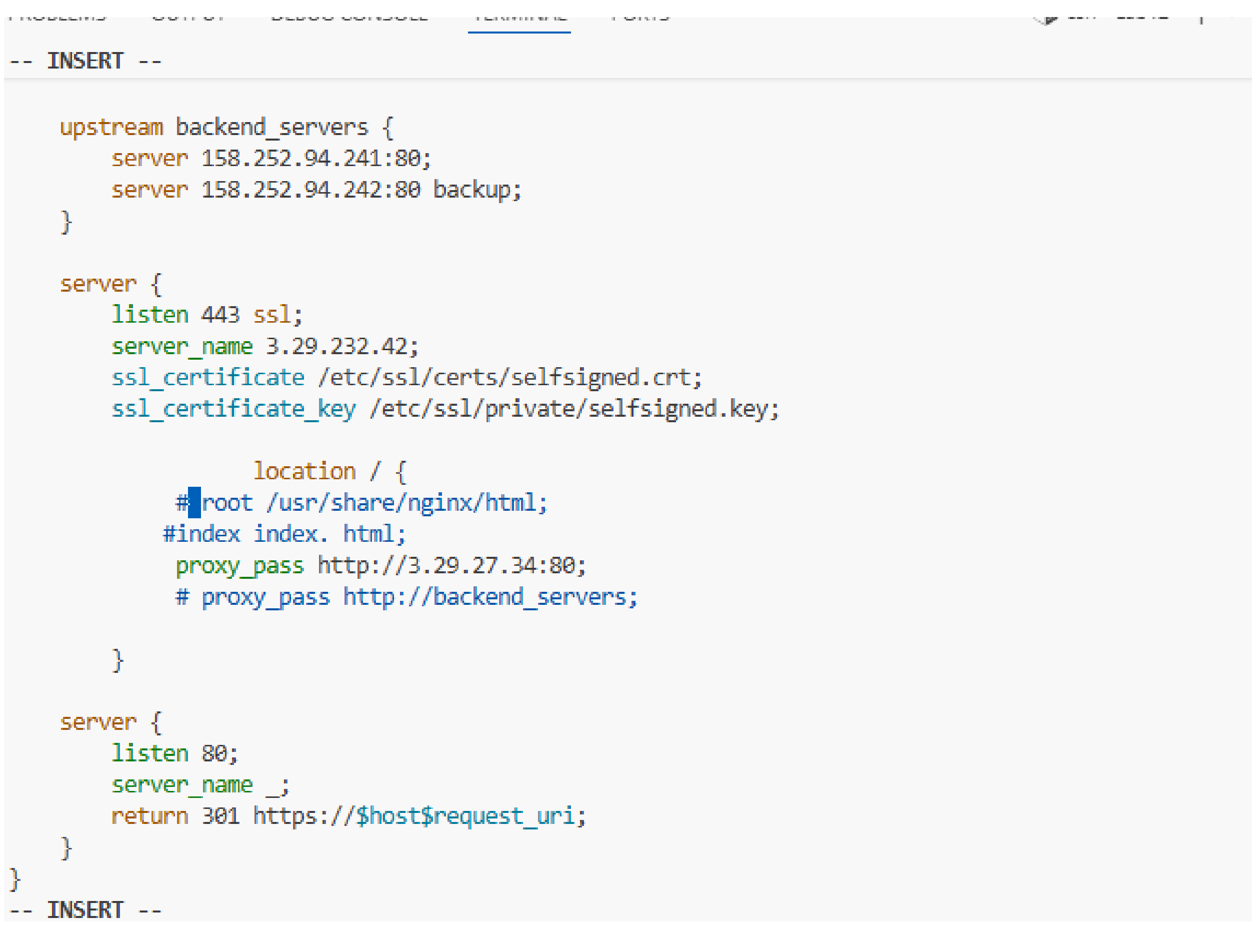
[ec2-user@ip-3-29-232-42 ~]$

- This means you are now **inside the webserver**
- **Open Nginx config**
- sudo vim /etc/nginx/nginx.conf



```
[ec2-user@ip-10-0-10-59 ~]$ sudo vim /etc/nginx/nginx.conf
```

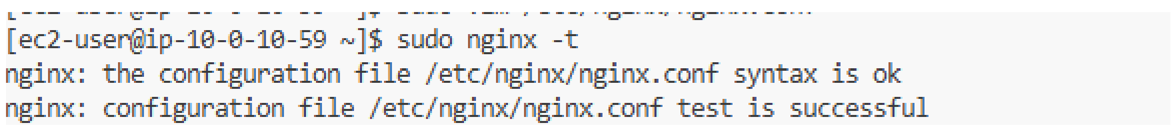
- **Step 3 — Modify the location / block**
- Find the block:



```
-- INSERT --

    upstream backend_servers {
        server 158.252.94.241:80;
        server 158.252.94.242:80 backup;
    }

    server {
        listen 443 ssl;
        server_name 3.29.232.42;
        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://3.29.27.34:80;
             # proxy_pass http://backend_servers;

        }

    server {
        listen 80;
        server_name _;
        return 301 https://$host$request_uri;
    }
}
-- INSERT --
```

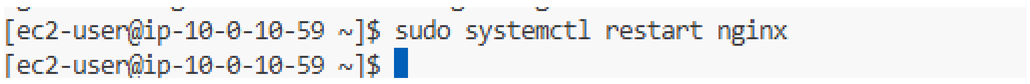
- Comment out the old root and index lines
- Replace <web-1-public-ip> with **actual backend IP** (3.29.27.34)

Test configuration



```
[ec2-user@ip-10-0-10-59 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Reload Nginx

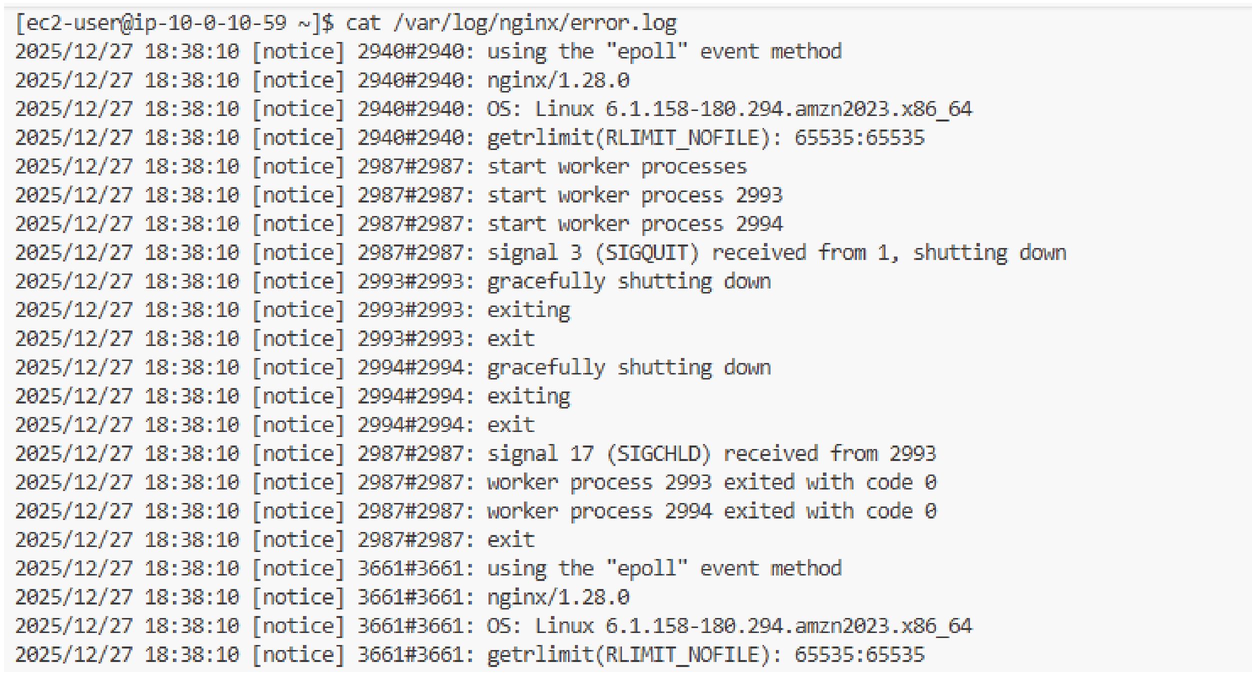


```
[ec2-user@ip-10-0-10-59 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-59 ~]$
```

Now HTTPS + reverse proxy + HTTP→HTTPS redirect should work perfectly.

View Nginx logs

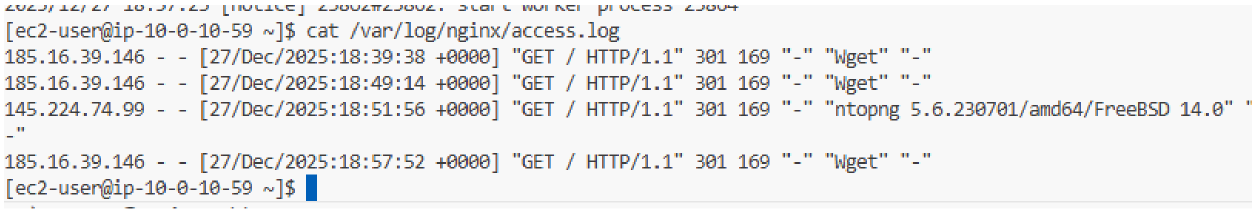
cat /var/log/nginx/error.log



```
[ec2-user@ip-10-0-10-59 ~]$ cat /var/log/nginx/error.log
2025/12/27 18:38:10 [notice] 2940#2940: using the "epoll" event method
2025/12/27 18:38:10 [notice] 2940#2940: nginx/1.28.0
2025/12/27 18:38:10 [notice] 2940#2940: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/27 18:38:10 [notice] 2940#2940: getrlimit(RLIMIT_NOFILE): 65535:65535
2025/12/27 18:38:10 [notice] 2987#2987: start worker processes
2025/12/27 18:38:10 [notice] 2987#2987: start worker process 2993
2025/12/27 18:38:10 [notice] 2987#2987: start worker process 2994
2025/12/27 18:38:10 [notice] 2987#2987: signal 3 (SIGQUIT) received from 1, shutting down
2025/12/27 18:38:10 [notice] 2993#2993: gracefully shutting down
2025/12/27 18:38:10 [notice] 2993#2993: exiting
2025/12/27 18:38:10 [notice] 2993#2993: exit
2025/12/27 18:38:10 [notice] 2994#2994: gracefully shutting down
2025/12/27 18:38:10 [notice] 2994#2994: exiting
2025/12/27 18:38:10 [notice] 2994#2994: exit
2025/12/27 18:38:10 [notice] 2987#2987: signal 17 (SIGCHLD) received from 2993
2025/12/27 18:38:10 [notice] 2987#2987: worker process 2993 exited with code 0
2025/12/27 18:38:10 [notice] 2987#2987: worker process 2994 exited with code 0
2025/12/27 18:38:10 [notice] 2987#2987: exit
2025/12/27 18:38:10 [notice] 3661#3661: using the "epoll" event method
2025/12/27 18:38:10 [notice] 3661#3661: nginx/1.28.0
2025/12/27 18:38:10 [notice] 3661#3661: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/27 18:38:10 [notice] 3661#3661: getrlimit(RLIMIT_NOFILE): 65535:65535
```

- Screenshot: task7_error_log.png

cat /var/log/nginx/access.log

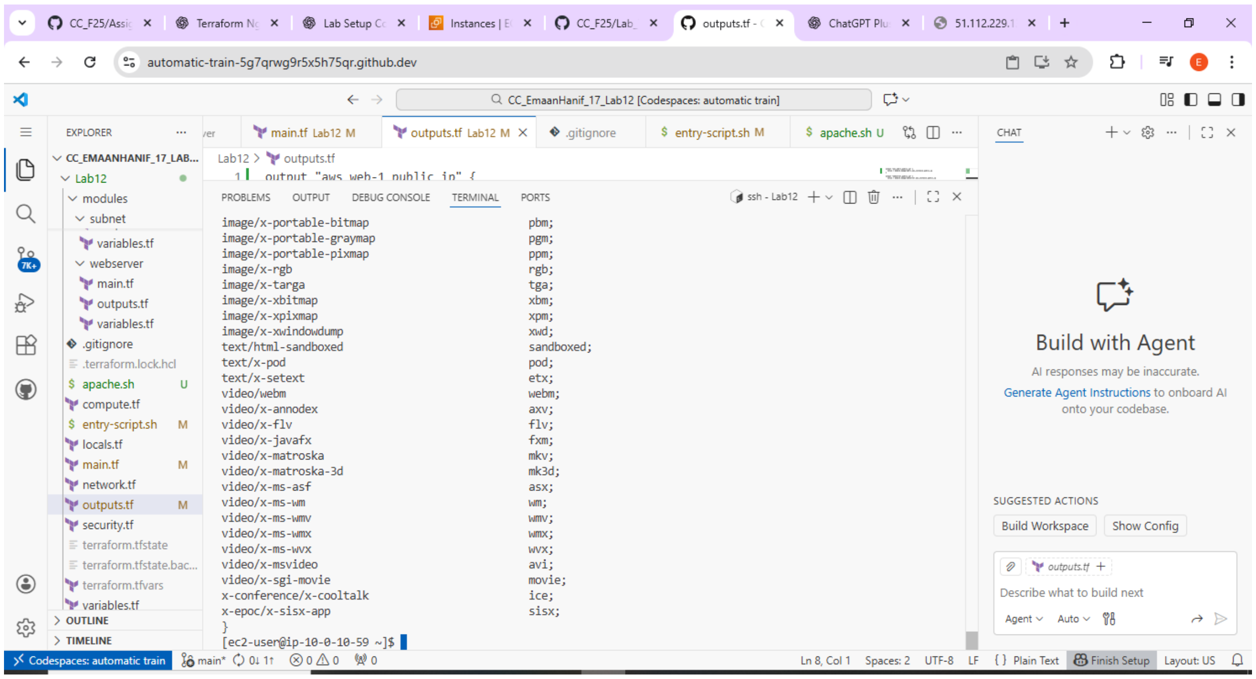


```
[ec2-user@ip-10-0-10-59 ~]$ cat /var/log/nginx/access.log
185.16.39.146 - - [27/Dec/2025:18:39:38 +0000] "GET / HTTP/1.1" 301 169 "-" "Wget" "-"
185.16.39.146 - - [27/Dec/2025:18:49:14 +0000] "GET / HTTP/1.1" 301 169 "-" "Wget" "-"
145.224.74.99 - - [27/Dec/2025:18:51:56 +0000] "GET / HTTP/1.1" 301 169 "-" "ntopng 5.6.230701/amd64/FreeBSD 14.0" "
-"
185.16.39.146 - - [27/Dec/2025:18:57:52 +0000] "GET / HTTP/1.1" 301 169 "-" "Wget" "-"
[ec2-user@ip-10-0-10-59 ~]$
```

- Screenshot: task7_access_log.png

2 View configuration files

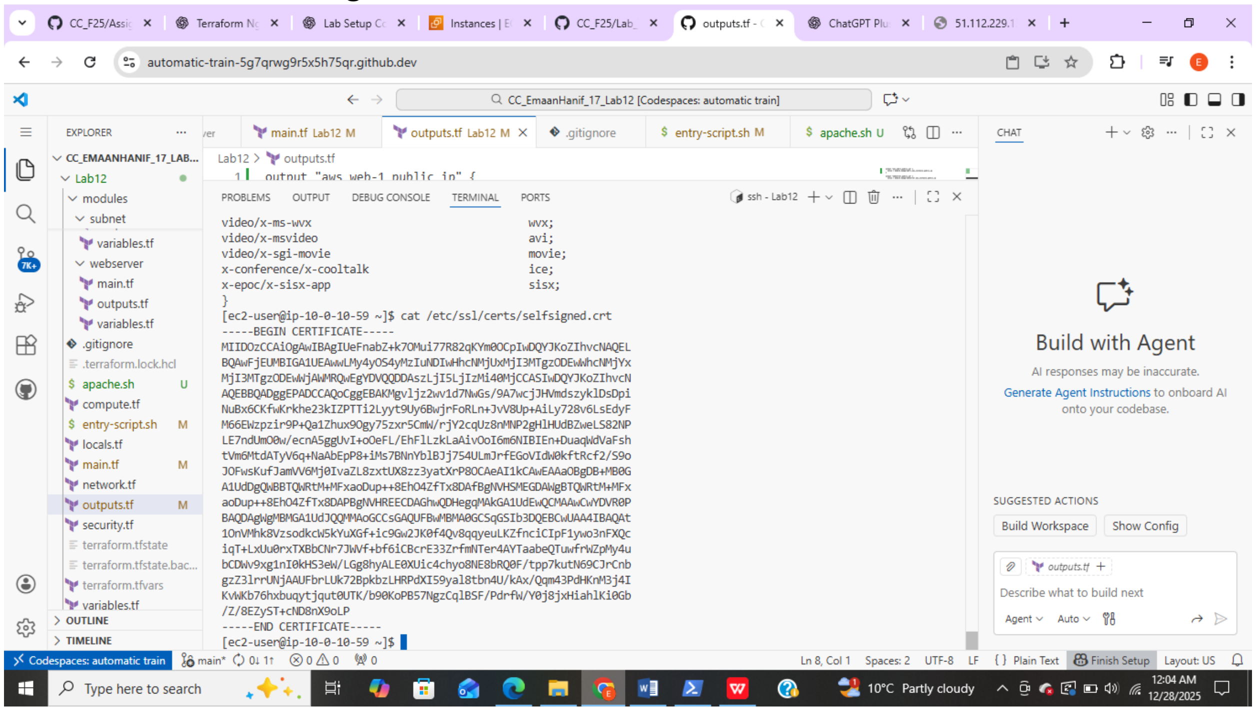
cat /etc/nginx/mime.types



```
image/x-portable-bitmap                         pbm;
image/x-portable-graymap                        pgm;
image/x-portable-pixmap                         ppm;
image/x-rgb                                     rgb;
image/x-targa                                   tga;
image/x-xbitmap                                 xbm;
image/x-xpixmap                                 xpm;
image/x-xwindowdump                             xwd;
text/html-sandboxed                             sandboxed;
text/x-pod                                      pod;
text/x-setext                                   etx;
video/webm                                      webm;
video/x-annodex                                 axv;
video/x-flv                                     flv;
video/x-javafx                                  fxm;
video/x-matroska                                mkv;
video/x-matroska-3d                             mk3d;
video/x-ms-asf                                  asx;
video/x-ms-wm                                   wm;
video/x-ms-wmv                                  wmv;
video/x-ms-wmx                                  wmx;
video/x-ms-wvx                                  wvx;
video/x-msvideo                                 avi;
video/x-sgi-movie                               movie;
x-conference/x-cooltalk                         ice;
x-epoc/x-sisx-app                               sisx;
}
[ec2-user@ip-10-0-10-59 ~]$
```

- Screenshot: task7_mime_types.png

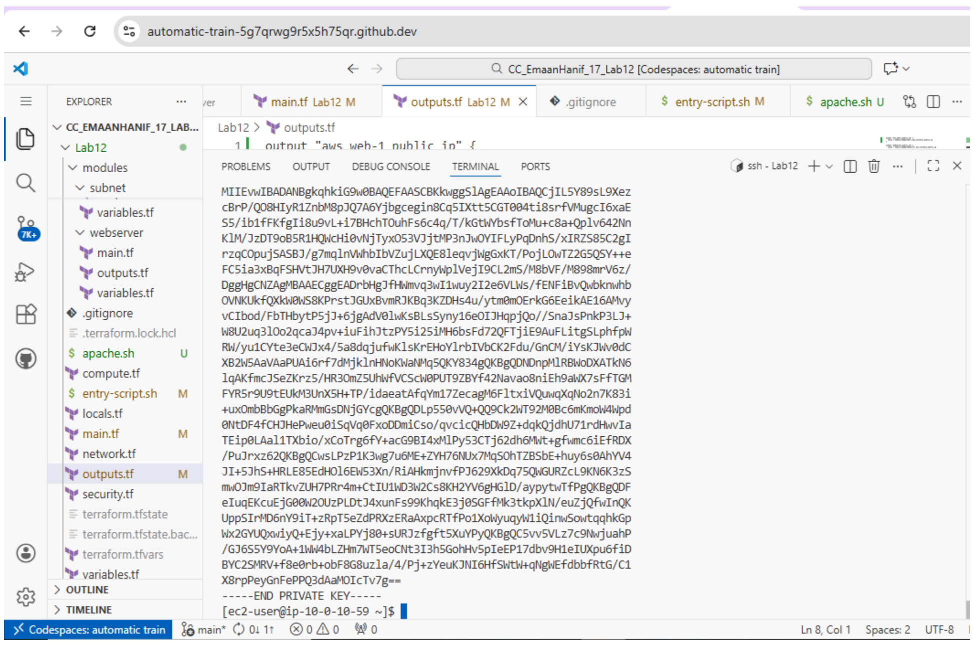
cat /etc/ssl/certs/selfsigned.crt



```
video/x-ms-wvx                                  wvx;
video/x-msvideo                                 avi;
video/x-sgi-movie                               movie;
x-conference/x-cooltalk                         ice;
x-epoc/x-sisx-app                               sisx;
}
[ec2-user@ip-10-0-10-59 ~]$ cat /etc/ssl/certs/selfsigned.crt
-----BEGIN CERTIFICATE-----
MIIDOzCCAiOgAwIBAgIUeFnabZ+k7OMui77R82qKYm0OCpIwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAwwLMy4yOS4yMzIuNDIwHhcNMjUxMjI3MTgzODEwWhcNMjYx
MjI3MTgzODEwWjAWMRQwEgYDVQQDDAszLjI5LjIzMi40MjCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKMgvljz2wv1d7NwGs/9A7wcjJHVmdszyklDsDpi
NuBx6CKfwKrkhe23kIZPTTi2Lyyt9Uy6BwjrFoRLn+JvV8Up+AiLy728v6LsEdyF
M66EWzpzir9P+Qa1Zhux9Ogy75zxr5CmW/rjY2cqUz8nMNP2gHlHUdBZweLS82NP
LE7ndUmO0w/ecnA5ggUvI+oOeFL/EhFlLzkLaAivOoI6m6NIBIEn+DuaqWdVaFsh
tVm6MtdATyV6q+NaAbEpP8+iMs7BNnYblBJj754ULmJrfEGoVIdW0kftRcf2/S9o
JOFwsKufJamVV6Mj0IvaZL8zxtUX8zz3yatXrP8OCAeAI1kCAwEAAaOBgDB+MB0G
A1UdDgQWBBTQWRtM+MFxaoDup++8EhO4ZfTx8DAfBgNVHSMEGDAWgBTQWRtM+MFx
aoDup++8EhO4ZfTx8DAPBgNVHREECDAGhwQDHegqMAkGA1UdEwQCMAAwCwYDVR0P
BAQDAgWgMBMGA1UdJQQMMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQAt
1OnVMhk8VzsodkcW5kYuXGf+ic9Gw2JK0f4Qv8qqyeuLKZfnciCIpF1ywo3nFXQc
iqT+LxUu0rxTXBbCNr7JWVf+bf6iCBcrE33ZrfmNTer4AYTaabeQTuwfrWZpMy4u
bCDWv9xg1nI0kHS3eW/LGg8hyALE0XUic4chyo8NE8bRQ0F/tpp7kutN69CJrCnb
gzZ3lrrUNjAAUFbrLUk72BpkbzLHRPdXI59yal8tbn4U/kAx/Qqm43PdHKnM3j4I
KvWKb76hxbuqytjqut0UTK/b90KoPB57NgzCqlBSF/PdrfW/Y0j8jxHiahlKi0Gb
/Z/8EZyST+cND8nX9oLP
-----END CERTIFICATE-----
[ec2-user@ip-10-0-10-59 ~]$
```

- Screenshot: task7_ssl_cert.png

Sudo cat /etc/ssl/private/selfsigned.key



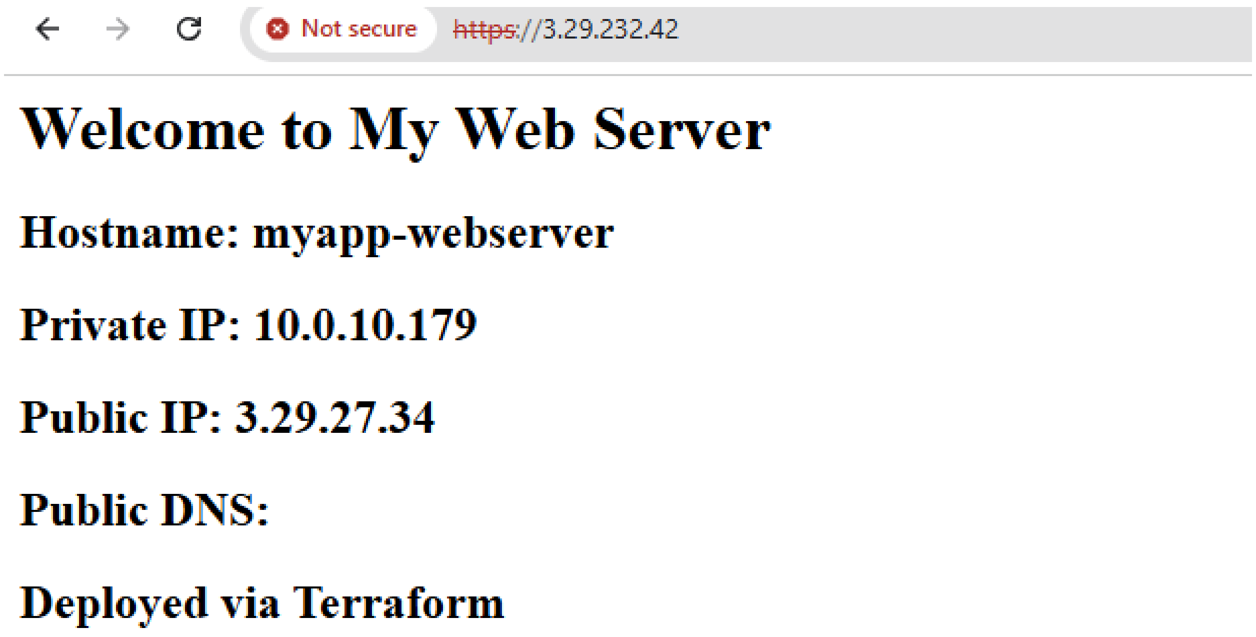
```
output "aws_web-1_public_ip" {
```

```
MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQCjIL5Y89sL9Xez
cBrP/QO8HIyR1ZnbM8pJQ7A6Yjbgcegin8Cq5IXtt5CGT004ti8srfVMugcI6xaE
S5/ib1fFKfgIi8u9vL+i7BHchTOuhFs6c4q/T/kGtWYbsfToMu+c8a+Qplv642Nn
KlM/JzDT9oB5R1HQWcHi0vNjTyxO53VJjtMP3nJwOYIFLyPqDnhS/xIRZS85C2gI
rzqCOpujSASBJ/g7mqlnVWhbIbVZujLXQE8leqvjWgGxKT/PojLOwTZ2G5QSY++e
FC5ia3xBqFSHVtJH7UXH9v0vaCThcLCrnyWplVejI9CL2mS/M8bVF/M898mrV6z/
DggHgCNZAgMBAAECggEADrbHgJfHWmvq3wI1wuy2I2e6VLWs/fENFiBvQwbknwhb
OVNKUkfQXkW0WS8KPrstJGUxBvmRJKBq3KZDHs4u/ytm0mOErkG6EeikAE16AMvy
vCIbod/FbTHbytP5jJ+6jgAdV0lwKsBLsSyny16eOIJHqpjQo//SnaJsPnkP3LJ+
W8U2uq3lOo2qcaJ4pv+iuFihJtzPY5i25iMH6bsFd72QFTjiE9AuFLitgSLphfpW
RW/yu1CYte3eCWJx4/5a8dqjufwKlsKrEHoYlrbIVbCK2Fdu/GnCM/iYsKJWv0dC
XB2W5AaVAaPUAi6rf7dMjklnHNoKWaNMq5QKY834gQKBgQDNDnpMlRBWoDXATkN6
lqAKfmcJSeZKrz5/HR3OmZ5UhWfVCScW0PUT9ZBYf42Navao8niEh9aWX7sFfTGM
FYR5r9U9tEUkM3UnX5H+TP/idaeatAfqYm17ZecagM6FltxiVQuwqXqNo2n7K83i
+uxOmbBbGgPkaRMmGsDNjGYcgQKBgQDLp550vVQ+QQ9Ck2WT92M0Bc6mKmoW4Wpd
0NtDF4fCHJHePweu0iSqVq0FxoDDmiCso/qvcicQHbDW9Z+dqkQjdhU71rdHwvIa
TEip0LAal1TXbio/xCoTrg6fY+acG9BI4xMlPy53CTj62dh6MWt+gfwmc6iEfRDX
/PuJrxz62QKBgQCwsLPzP1K3wg7u6ME+ZYH76NUx7MqSOhTZBSbE+huy6s0AhYV4
JI+5JhS+HRLE85EdHOl6EW53Xn/RiAHkmjnvfPJ629XkDq75QWGURZcL9KN6K3zS
mwOJm9IaRTkvZUH7PRr4m+CtIU1WD3W2Cs8KH2YV6gHGlD/aypytwTfPgQKBgQDF
eIuqEKcuEjG00W2OUzPLDtJ4xunFs99KhqkE3j0SGFfMk3tkpXlN/euZjQfwInQK
UppSIrMD6nY9iT+zRpT5eZdPRXzERaAxpcRTfPo1XoWyuqyW1iQinwSowtqqhkGp
Wx2GYUQxwiyQ+Ejy+xaLPYj80+sURJzfgft5XuYPyQKBgQC5vv5VLz7c9NwjuahP
/GJ6S5Y9YoA+1WW4bLZHm7WT5eoCNt3I3h5GohHv5pIeEP17dbv9H1eIUXpu6fiD
BYC2SMRV+f8e0rb+obF8G8uzla/4/Pj+zYeuKJNI6HfSWtW+qNgWEfdbbfRtG/C1
X8rpPeyGnFePPQ3dAaMOIcTv7g==
-----END PRIVATE KEY-----
[ec2-user@ip-10-0-10-59 ~]$
```

- Screenshot: task7_ssl_key.png

Test reverse proxy in browser

1. Open browser → https://3.29.232.42 (your main webserver IP)
2. Accept **self-signed certificate warning**
3. You should see **Apache page from backend web-1**
4. Screenshot: task7_reverse_proxy_browser.png



Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.179

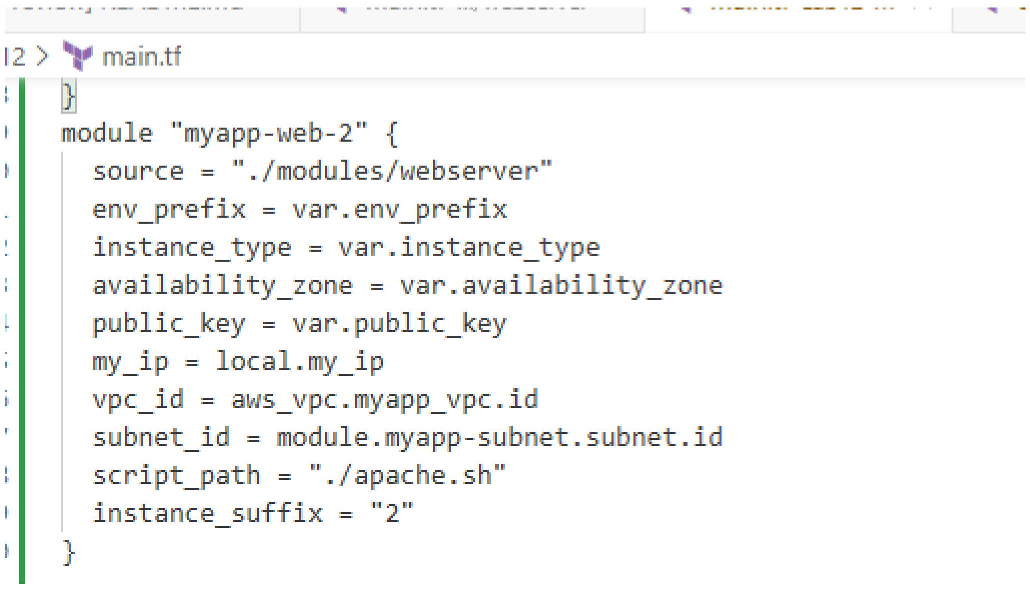
Public IP: 3.29.27.34

Public DNS:

Deployed via Terraform

Task8-Configure Nginx as load balancer

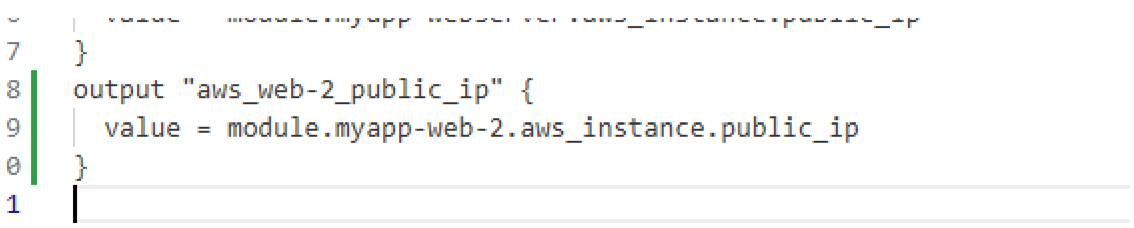
1. Add second backend server
   - In main.tf, create a new module myapp-web-2 just like myapp-web-1 but with instance_suffix = "2"
   - This launches a second EC2 running Apache.


main.tf

```
}
module "myapp-web-2" {
  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 = aws_vpc.myapp_vpc.id
  subnet_id = module.myapp-subnet.subnet.id
  script_path = "./apache.sh"
  instance_suffix = "2"
}
```

Update outputs.tf

- Add output for the new web-2 public IP:

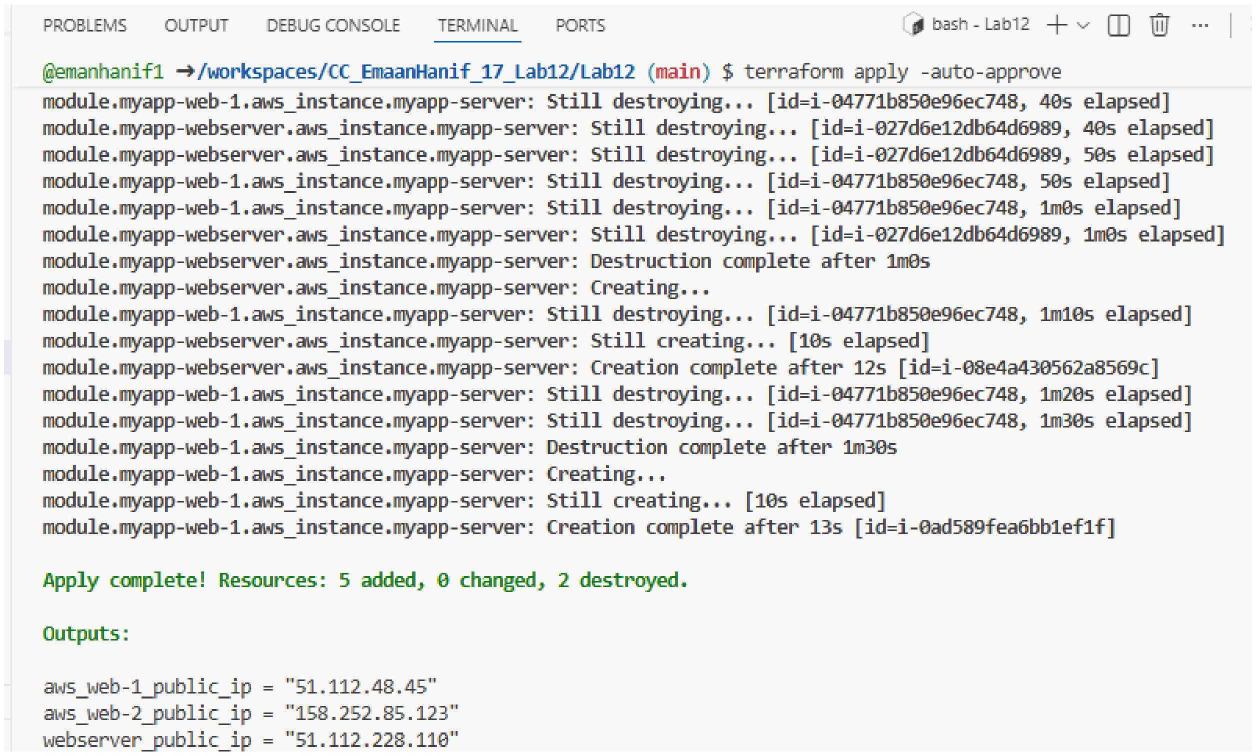


```
}
output "aws_web-2_public_ip" {
  value = module.myapp-web-2.aws_instance.public_ip
}
```

Terraform commands to run

# Initialize Terraform
terraform init

# Apply changes (creates web-2 and updates outputs)
terraform apply -auto-approve

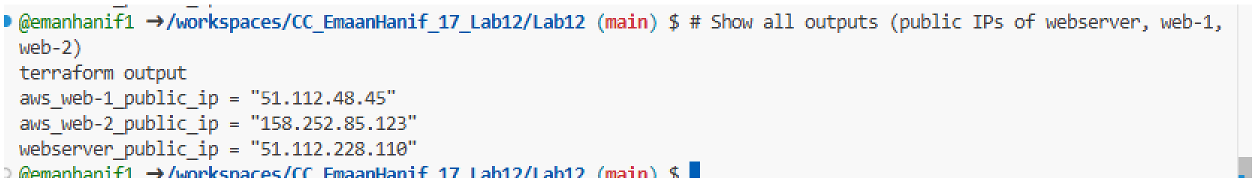

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - Lab12

```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform apply -auto-approve
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 40s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-027d6e12db64d6989, 40s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-027d6e12db64d6989, 50s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 50s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 1m0s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-027d6e12db64d6989, 1m0s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Destruction complete after 1m0s
module.myapp-webserver.aws_instance.myapp-server: Creating...
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 1m10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still creating... [10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 12s [id=i-08e4a430562a8569c]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 1m20s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-04771b850e96ec748, 1m30s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Destruction complete after 1m30s
module.myapp-web-1.aws_instance.myapp-server: Creating...
module.myapp-web-1.aws_instance.myapp-server: Still creating... [10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Creation complete after 13s [id=i-0ad589fea6bb1ef1f]

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

Outputs:

aws_web-1_public_ip = "51.112.48.45"
aws_web-2_public_ip = "158.252.85.123"
webserver_public_ip = "51.112.228.110"
```

- Screenshot: task8_terraform_apply.png

terraform output

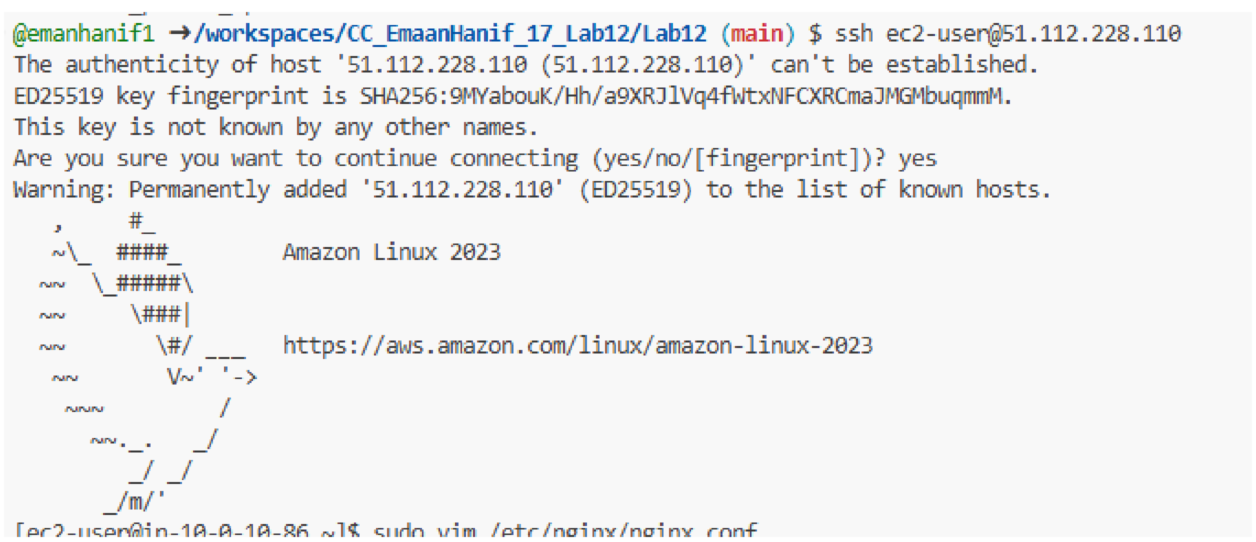
- Screenshot: task8_terraform_output.png



```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ # Show all outputs (public IPs of webserver, web-1,
web-2)
terraform output
aws_web-1_public_ip = "51.112.48.45"
aws_web-2_public_ip = "158.252.85.123"
webserver_public_ip = "51.112.228.110"
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $
```

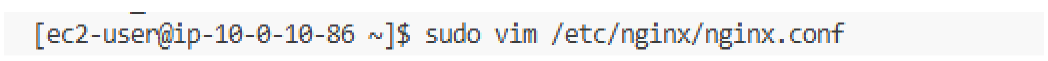
SSH into Nginx webserver

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

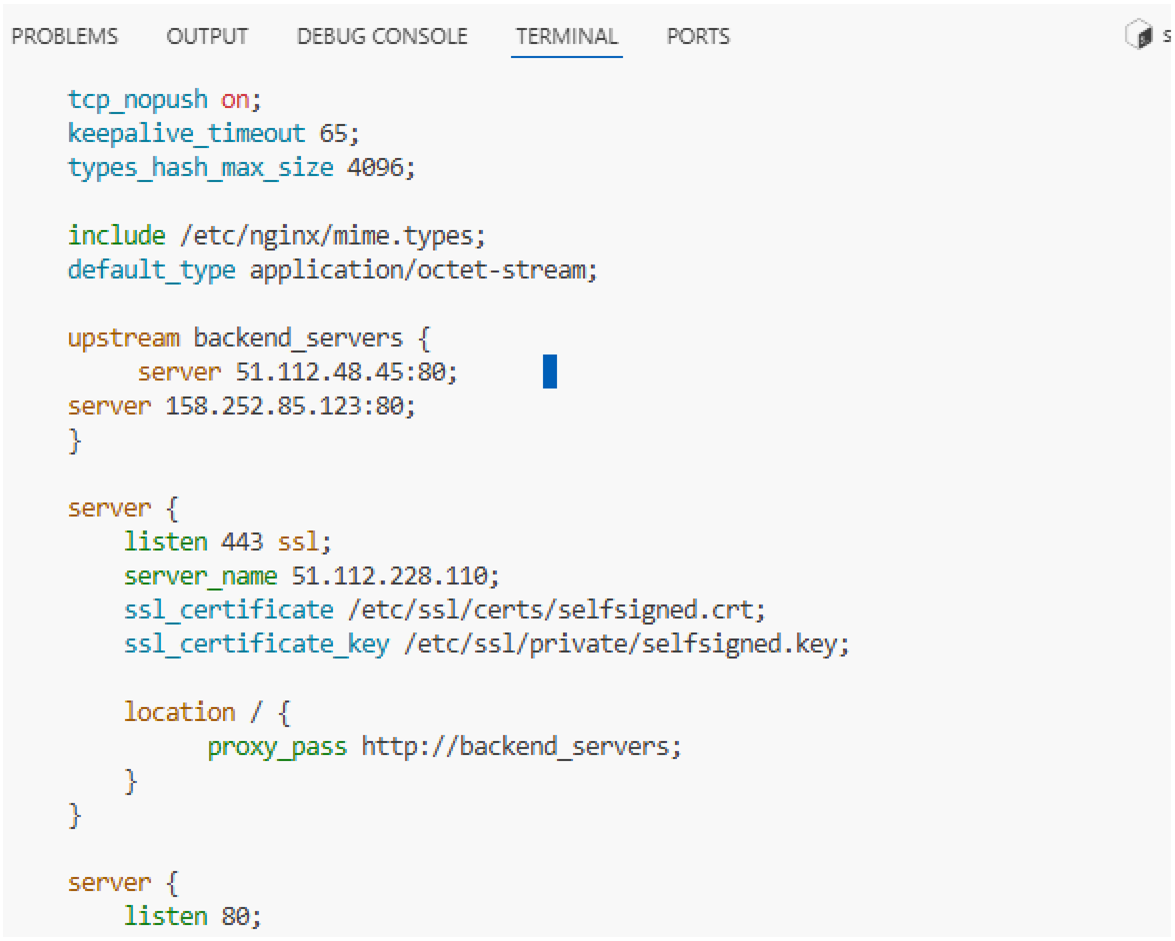

```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ssh ec2-user@51.112.228.110
The authenticity of host '51.112.228.110 (51.112.228.110)' can't be established.
ED25519 key fingerprint is SHA256:9MYabouK/Hh/a9XRJlVq4fWtxNFCXRCmaJMGMbuqmmM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.228.110' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-86 ~]$ sudo vim /etc/nginx/nginx.conf
```

Edit Nginx configuration

sudo vim /etc/nginx/nginx.conf


```
[ec2-user@ip-10-0-10-86 ~]$ sudo vim /etc/nginx/nginx.conf
```

Replace placeholders with your real public IPs:



```
    tcp_nopush on;
    keepalive_timeout 65;
    types_hash_max_size 4096;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    upstream backend_servers {
         server 51.112.48.45:80;
    server 158.252.85.123:80;
    }

    server {
        listen 443 ssl;
        server_name 51.112.228.110;
        ssl_certificate /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;

        location / {
              proxy_pass http://backend_servers;
        }
    }

    server {
        listen 80;
```

Update location block

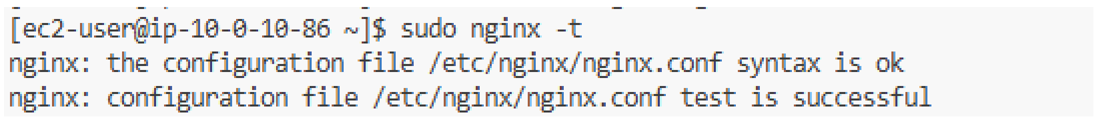
Make sure your location / points to backend_servers:

```
location / {
   proxy_pass http://backend_servers;
}
```

- **Remove any old proxy_pass lines** like proxy_pass http://<web-1-public-ip>:80;

Test and restart Nginx

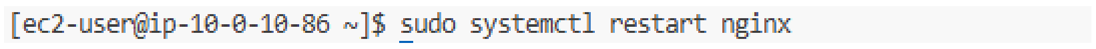
```
sudo nginx -t         # test configuration
```



```
[ec2-user@ip-10-0-10-86 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

sudo systemctl restart nginx

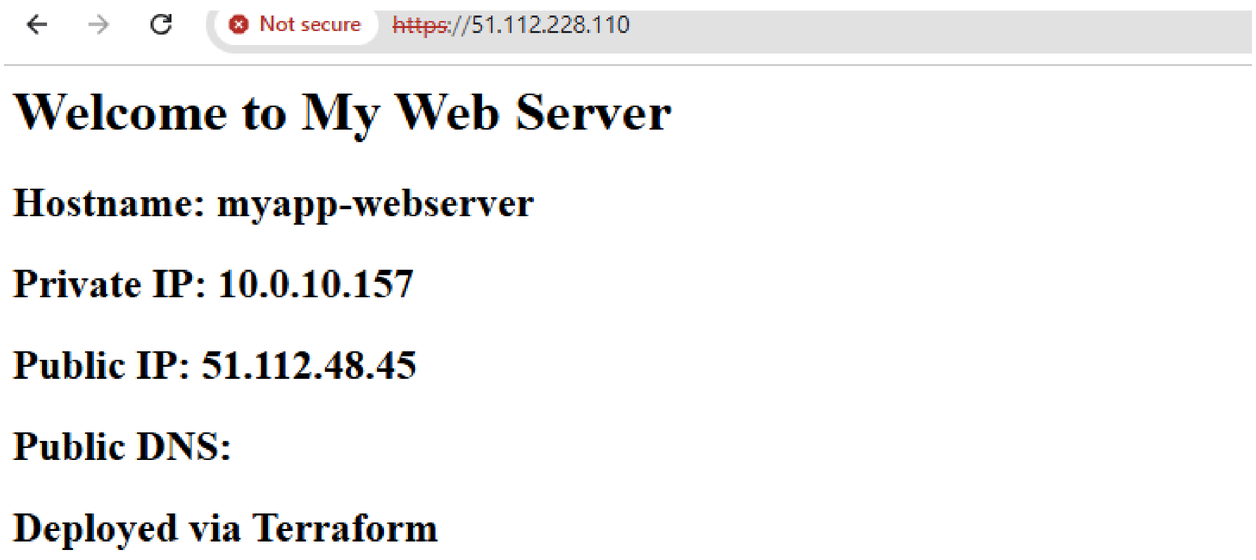
- If syntax is correct, Nginx will reload.



```
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl restart nginx
```

Test in browser

- Open https://<webserver-public-ip>
- Reload several times → should alternate between web-1 and web-2 content.



Welcome to My Web Server

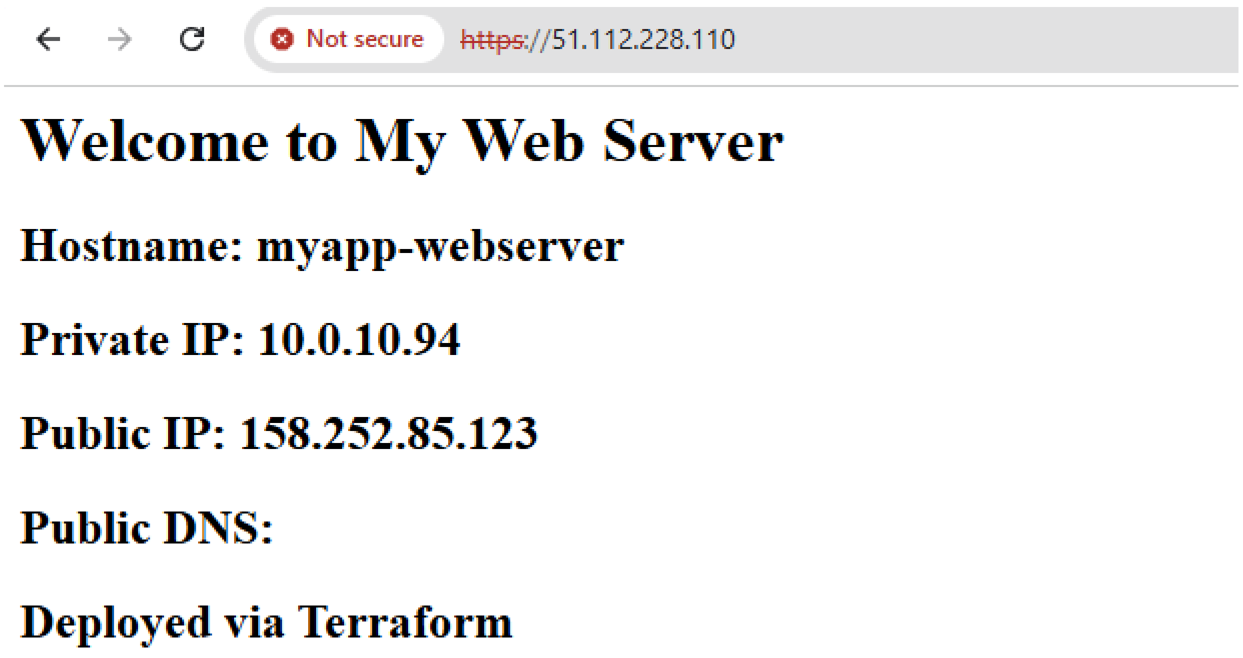
Hostname: myapp-webserver

Private IP: 10.0.10.157

Public IP: 51.112.48.45

Public DNS:

Deployed via Terraform



~~https~~://51.112.228.110

Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.94

Public IP: 158.252.85.123

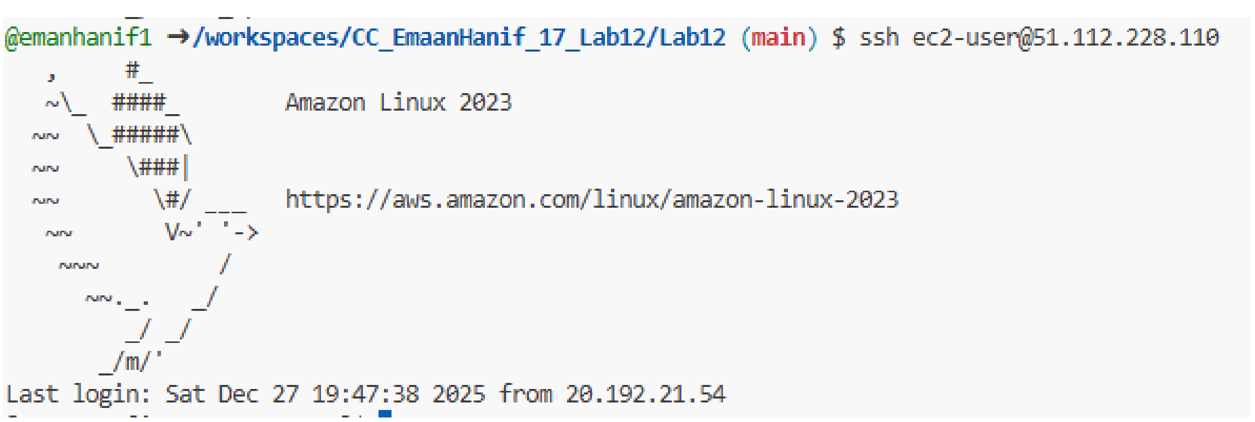
Public DNS:

Deployed via Terraform

Task 9-Configure high availability with backup servers

1. SSH into the webserver:

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

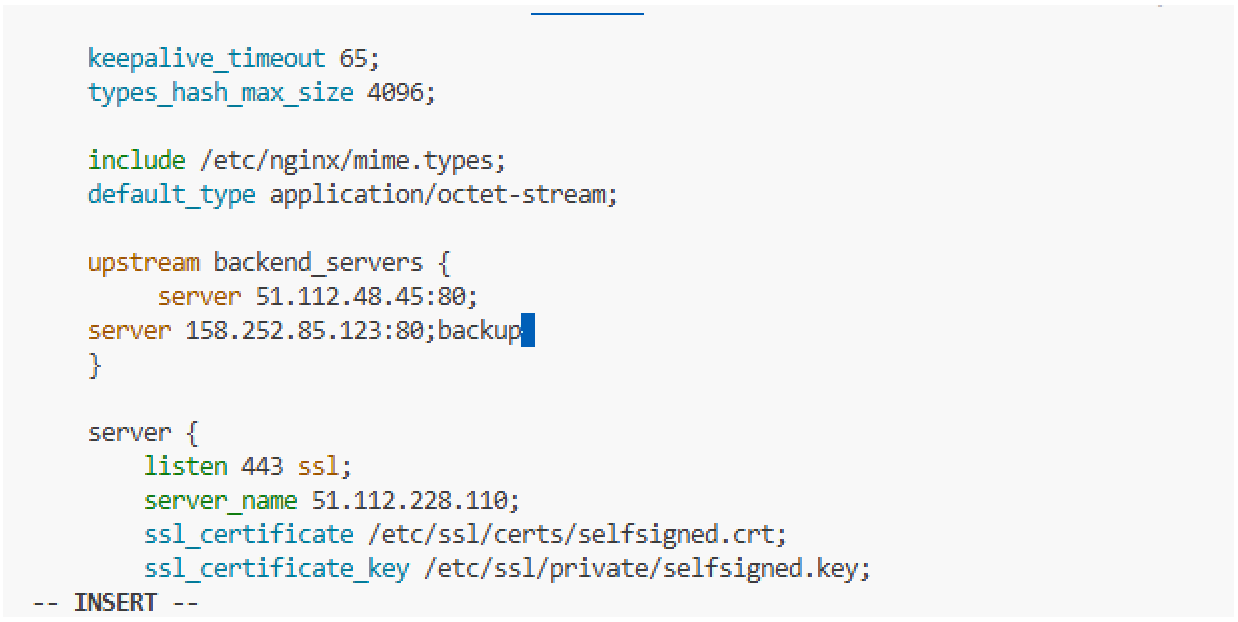


```
@emanhanif1 ➜ /workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ssh ec2-user@51.112.228.110
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Dec 27 19:47:38 2025 from 20.192.21.54
```

Configure web-1 as primary, web-2 as backup

sudo vim /etc/nginx/nginx.conf

Update **upstream block**:



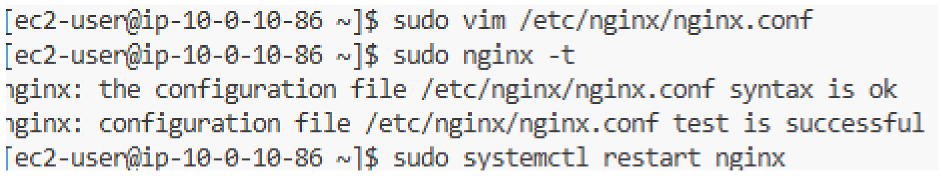
```
    keepalive_timeout 65;
    types_hash_max_size 4096;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    upstream backend_servers {
         server 51.112.48.45:80;
    server 158.252.85.123:80;backup
    }

    server {
        listen 443 ssl;
        server_name 51.112.228.110;
        ssl_certificate /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;
  -- INSERT --
```

Restart Nginx

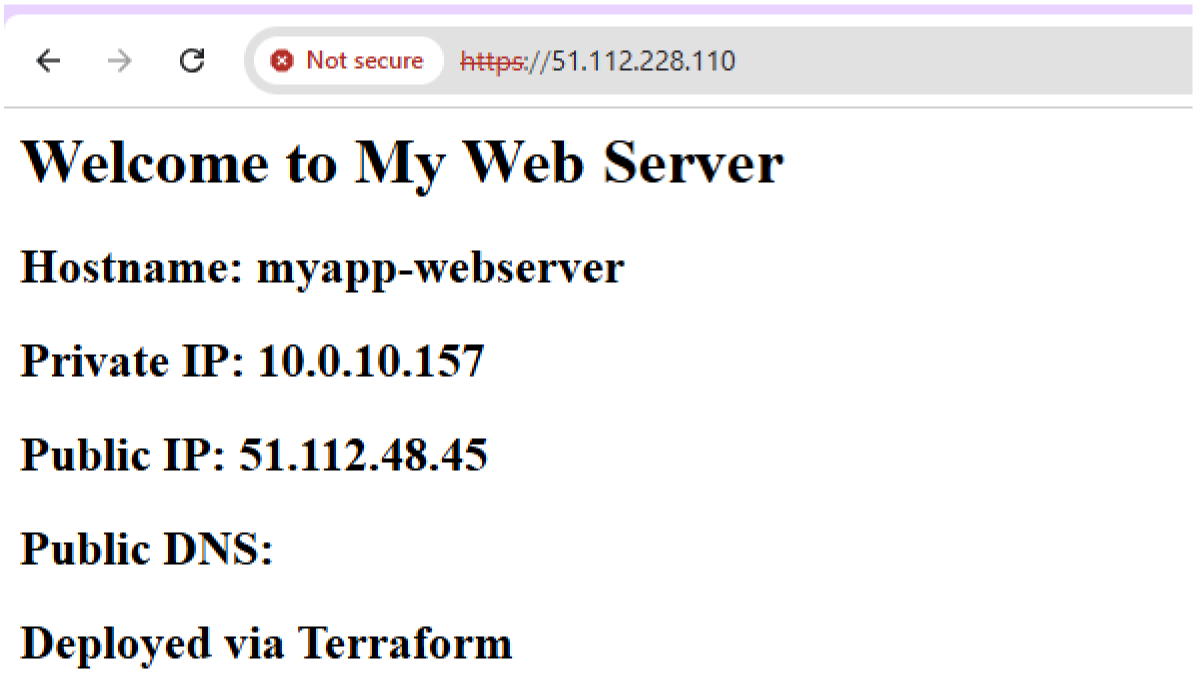
sudo nginx -t
sudo systemctl restart nginx

- Test in browser: https://<webserver-public-ip>

- Reload multiple times → **only web-1 content**



```
[ec2-user@ip-10-0-10-86 ~]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-86 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl restart nginx
```



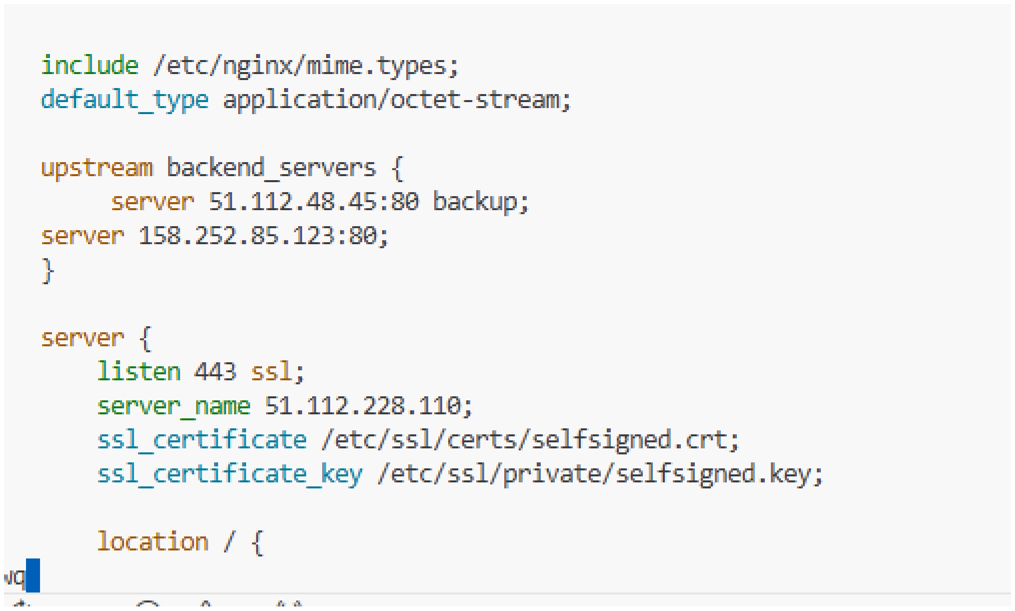
```
Welcome to My Web Server
Hostname: myapp-webserver
Private IP: 10.0.10.157
Public IP: 51.112.48.45
Public DNS:
Deployed via Terraform
```

Switch to web-2 as primary, web-1 as backup

Edit again:

sudo vim /etc/nginx/nginx.conf

Update upstream:



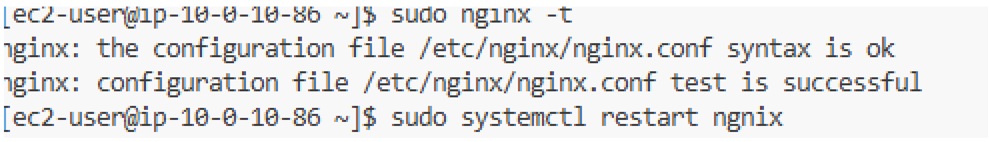
```
include /etc/nginx/mime.types;
default_type application/octet-stream;

upstream backend_servers {
     server 51.112.48.45:80 backup;
server 158.252.85.123:80;
}

server {
    listen 443 ssl;
    server_name 51.112.228.110;
    ssl_certificate /etc/ssl/certs/selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/selfsigned.key;

    location / {
wq
```

Restart Nginx

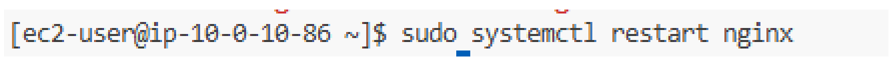
sudo nginx -t

sudo systemctl restart nginx

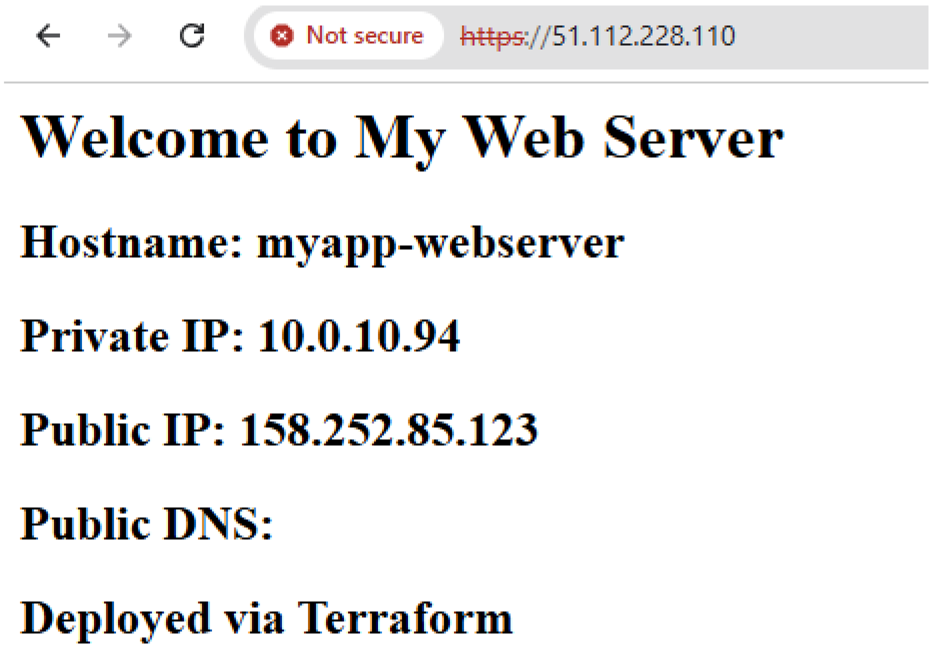
- Test in browser → reload multiple times → **only web-2 content**



```
[ec2-user@ip-10-0-10-86 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl restart ngnix
```



```
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl restart nginx
```


Not secure ~~https~~://51.112.228.110

Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.94

Public IP: 158.252.85.123

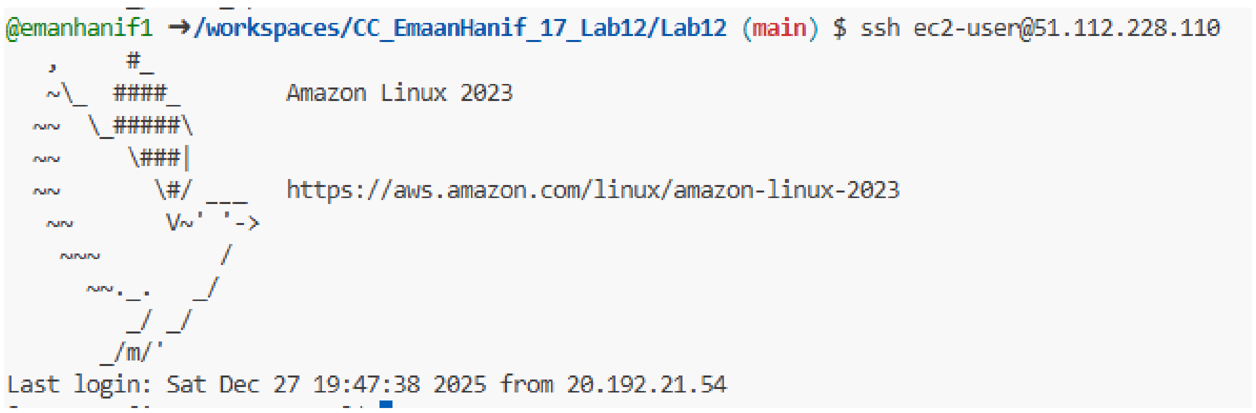
Public DNS:

Deployed via Terraform

Task 10— Enable Nginx caching

- SSH into the EC2 webserver using the public IP:

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



```
@emanhanif1 →/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ ssh ec2-user@51.112.228.110
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Dec 27 19:47:38 2025 from 20.192.21.54
```

Opened the Nginx configuration file for editing:

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

- Made the necessary changes in **nginx.conf** to enable caching.
- Saved and exited the file after editing.
- Verified the changes to ensure Nginx caching was correctly configured.

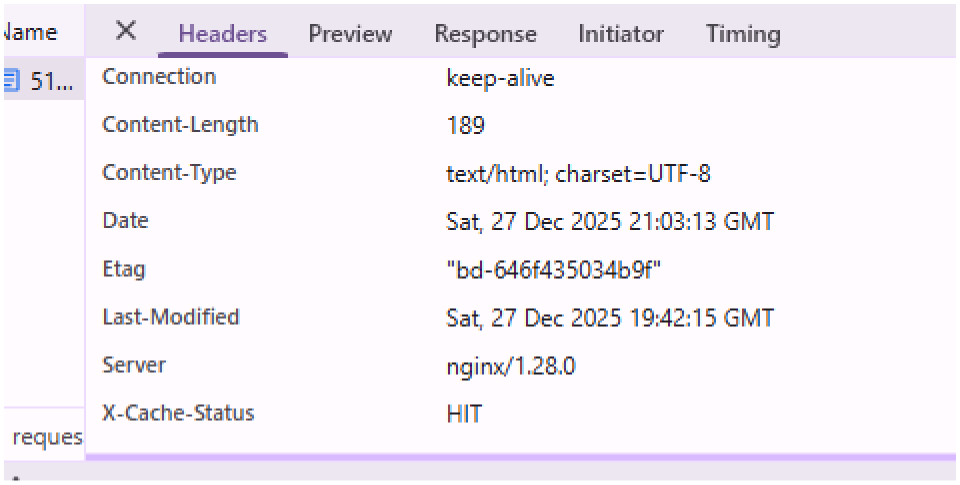
```
http {
        proxy_cache_path /var/cache/nginx
levels=1:2
keys_zone=my_cache:10m
inactive=60m
max_size=1g;
```

Defined a cache path `/var/cache/nginx` with `keys_zone=my_cache:10m`, `inactive=60m`, and `max_size=1g`.

- Configured `log_format` to track requests.
- Updated the `server` and `location /` blocks:
- Set `proxy_pass` to `http://backend_servers`.
- Enabled `proxy_cache my_cache` and `proxy_cache_valid 200 60m`.
- Added `proxy_cache_key` and `add_header X-Cache-Status $upstream_cache_status`.

```
    location / {
  proxy_pass http://backend_servers;

  proxy_cache my_cache;
  proxy_cache_valid 200 60m;
  proxy_cache_key "$scheme$request_uri";
  add_header X-Cache-Status $upstream_cache_status;
}
  }
```

- Added an **upstream block** for backend servers: `<web-1-public-ip>` and `<web-2-public-ip>`.



Connection	keep-alive
Content-Length	189
Content-Type	text/html; charset=UTF-8
Date	Sat, 27 Dec 2025 21:03:13 GMT
Etag	"bd-646f435034b9f"
Last-Modified	Sat, 27 Dec 2025 19:42:15 GMT
Server	nginx/1.28.0
X-Cache-Status	HIT

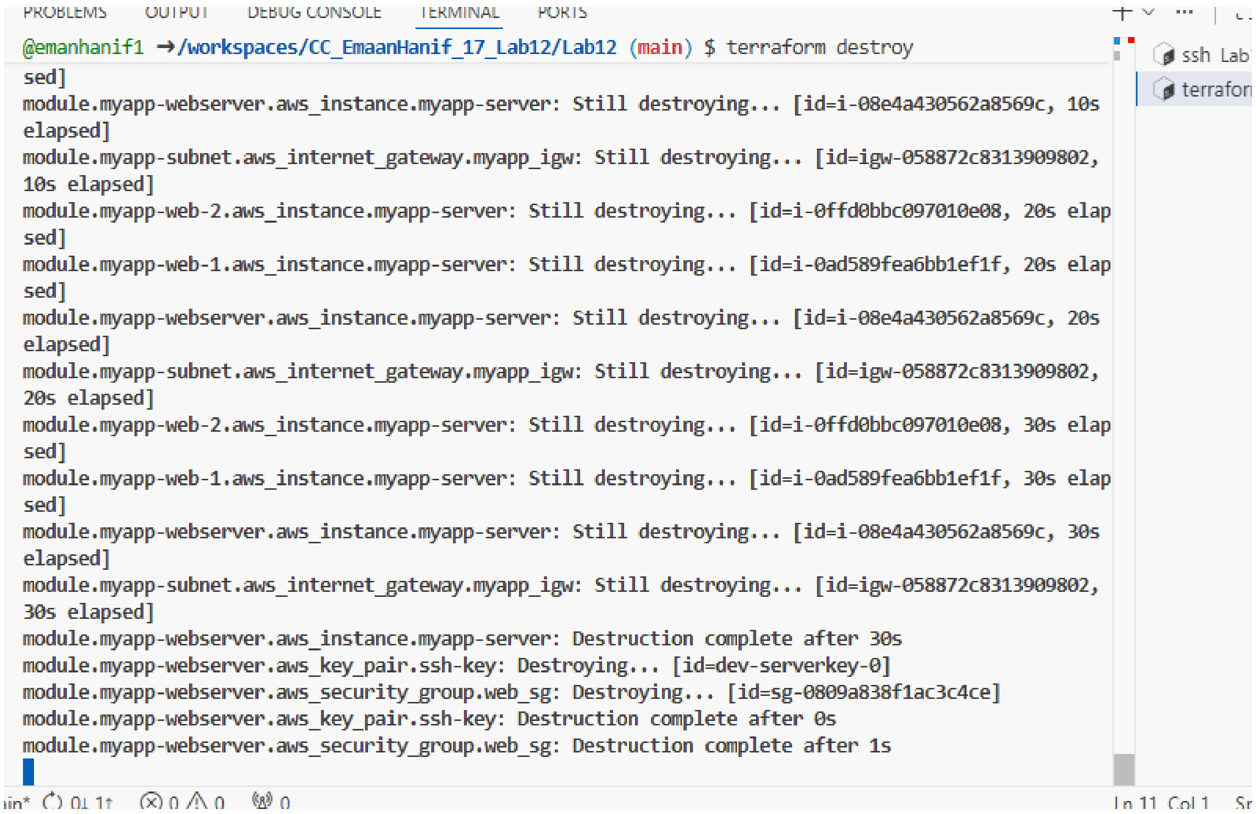
CLEANUP

1. Exit SSH session:

```
exit
```

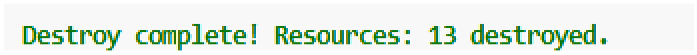
2. Destroy all resources:

```
terraform destroy
```

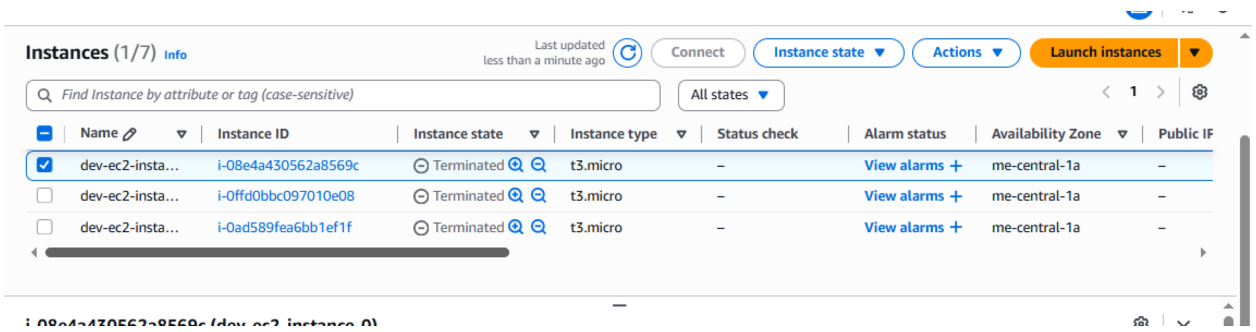


```
@emanhanif1 ➜/workspaces/CC_EmaanHanif_17_Lab12/Lab12 (main) $ terraform destroy
sed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-08e4a430562a8569c, 10s
elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-058872c8313909802,
10s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Still destroying... [id=i-0ffd0bbc097010e08, 20s elap
sed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-0ad589fea6bb1ef1f, 20s elap
sed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-08e4a430562a8569c, 20s
elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-058872c8313909802,
20s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Still destroying... [id=i-0ffd0bbc097010e08, 30s elap
sed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-0ad589fea6bb1ef1f, 30s elap
sed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-08e4a430562a8569c, 30s
elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-058872c8313909802,
30s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Destruction complete after 30s
module.myapp-webserver.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Destroying... [id=sg-0809a838f1ac3c4ce]
module.myapp-webserver.aws_key_pair.ssh-key: Destruction complete after 0s
module.myapp-webserver.aws_security_group.web_sg: Destruction complete after 1s
```

Destroy all resources



```
Destroy complete! Resources: 13 destroyed.
```


Instances (1/7) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
✓	dev-ec2-insta…	i-08e4a430562a8569c	Terminated	t3.micro	–	View alarms +	me-central-1a	–
	dev-ec2-insta…	i-0ffd0bbc097010e08	Terminated	t3.micro	–	View alarms +	me-central-1a	–
	dev-ec2-insta…	i-0ad589fea6bb1ef1f	Terminated	t3.micro	–	View alarms +	me-central-1a	–

i-08e4a430562a8569c (dev-ec2-instance-0)

VERIFIED THAT NO RESOURCES RUNNING IN CONSOLE