Name: Rughma Malik

Reg no: 2023-BSE-54

Course: Cloud Computing Lab

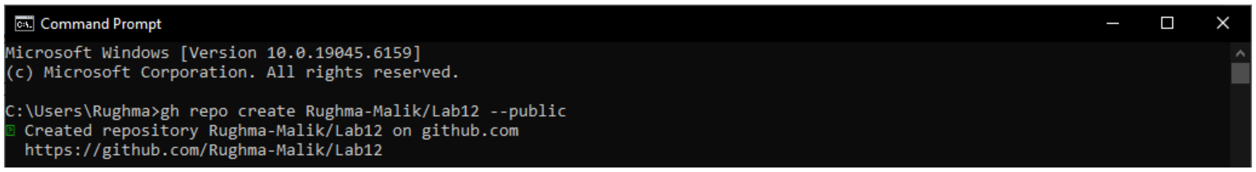
Section: V-B

LAB 12

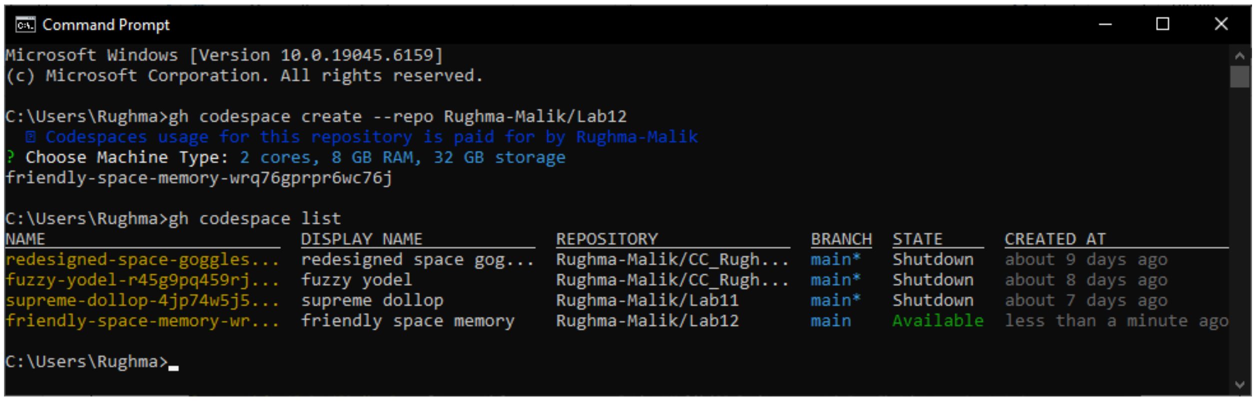
Terraform Provisioners, Modules & Nginx Reverse Proxy/Load Balancer

Task 0 Lab Setup (Codespace & GH CLI)

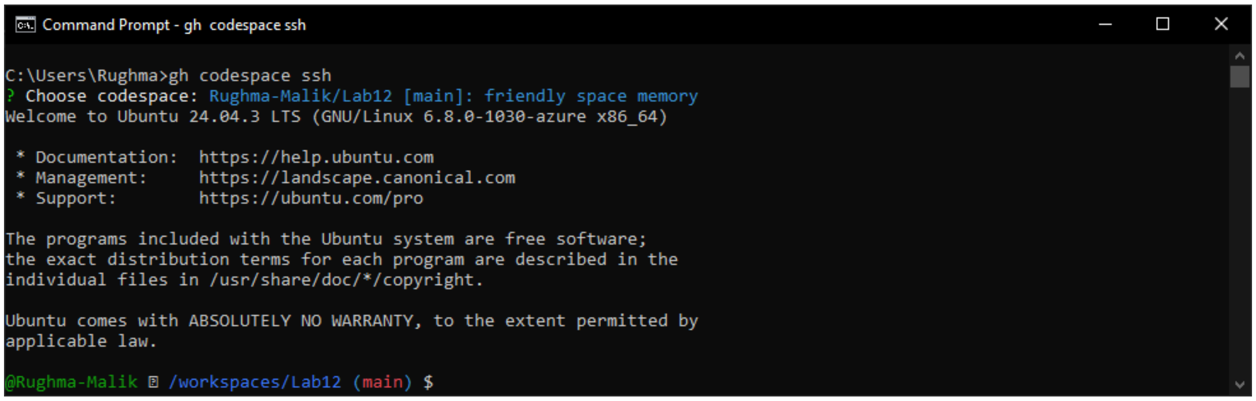
Create Codespace & connect:



```
Microsoft Windows [Version 10.0.19045.6159]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rughma>gh repo create Rughma-Malik/Lab12 --public
✓ Created repository Rughma-Malik/Lab12 on github.com
  https://github.com/Rughma-Malik/Lab12
```



```
Microsoft Windows [Version 10.0.19045.6159]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rughma>gh codespace create --repo Rughma-Malik/Lab12
  ✓ Codespaces usage for this repository is paid for by Rughma-Malik
? Choose Machine Type: 2 cores, 8 GB RAM, 32 GB storage
friendly-space-memory-wrq76gprpr6wc76j

C:\Users\Rughma>gh codespace list
NAME                          DISPLAY NAME              REPOSITORY                BRANCH  STATE      CREATED AT
redesigned-space-goggles...   redesigned space gog...   Rughma-Malik/CC_Rugh...   main*   Shutdown   about 9 days ago
fuzzy-yodel-r45g9pq459rj...   fuzzy yodel               Rughma-Malik/CC_Rugh...   main*   Shutdown   about 8 days ago
supreme-dollop-4jp74w5j5...   supreme dollop            Rughma-Malik/Lab11        main*   Shutdown   about 7 days ago
friendly-space-memory-wr...   friendly space memory     Rughma-Malik/Lab12        main    Available  less than a minute ago

C:\Users\Rughma>_
```



```
C:\Users\Rughma>gh codespace ssh
? Choose codespace: Rughma-Malik/Lab12 [main]: friendly space memory
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.

@Rughma-Malik ➜ /workspaces/Lab12 (main) $
```

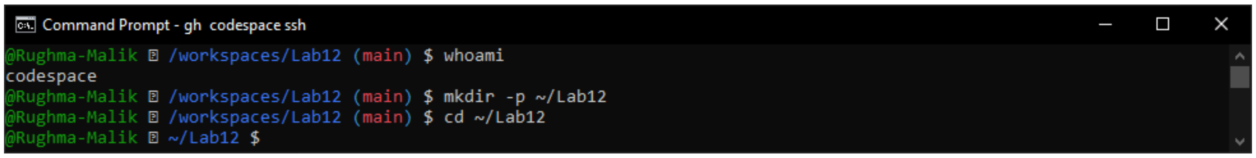
Task 1 — Organize Terraform code into separate files

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

1. Create the initial project structure:

mkdir -p ~/Lab12

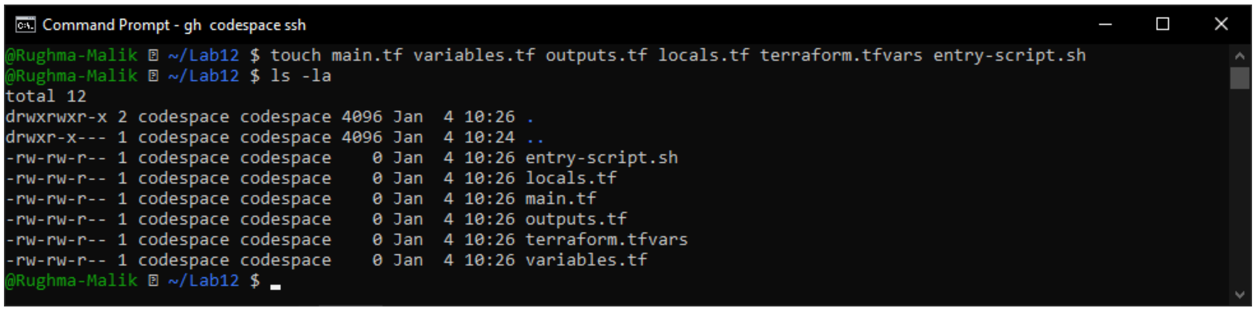
cd ~/Lab12



```
@Rughma-Malik ➜ /workspaces/Lab12 (main) $ whoami
codespace
@Rughma-Malik ➜ /workspaces/Lab12 (main) $ mkdir -p ~/Lab12
@Rughma-Malik ➜ /workspaces/Lab12 (main) $ cd ~/Lab12
@Rughma-Malik ➜ ~/Lab12 $
```

2. Create all required files:

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



```
@Rughma-Malik ➜ ~/Lab12 $ touch main.tf variables.tf outputs.tf locals.tf terraform.tfvars entry-script.sh
@Rughma-Malik ➜ ~/Lab12 $ ls -la
total 12
drwxrwxr-x 2 codespace codespace 4096 Jan  4 10:26 .
drwxr-x--- 1 codespace codespace 4096 Jan  4 10:24 ..
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 entry-script.sh
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 locals.tf
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 main.tf
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 outputs.tf
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 terraform.tfvars
-rw-rw-r-- 1 codespace codespace    0 Jan  4 10:26 variables.tf
@Rughma-Malik ➜ ~/Lab12 $
```

3. Create variables.tf with the following content:

variable "vpc_cidr_block" {}

variable "subnet_cidr_block" {}

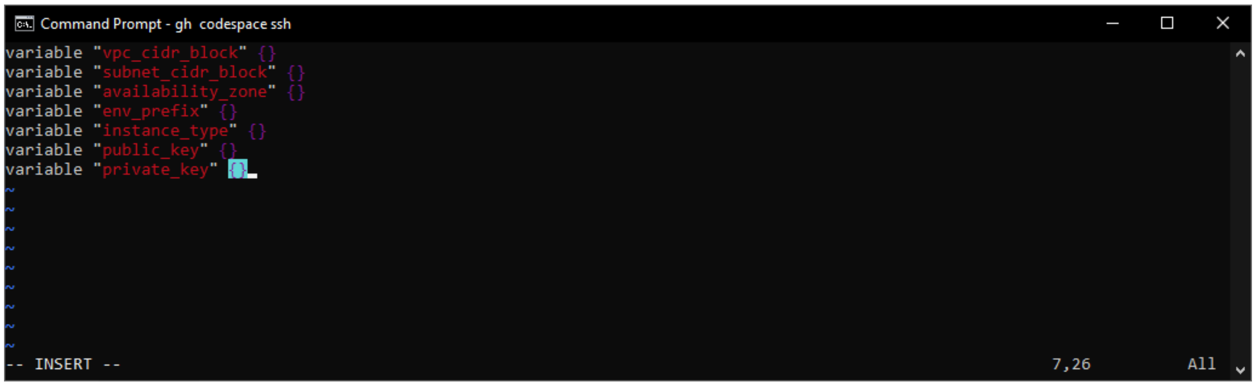
variable "availability_zone" {}

variable "env_prefix" {}

variable "instance_type" {}

variable "public_key" {}

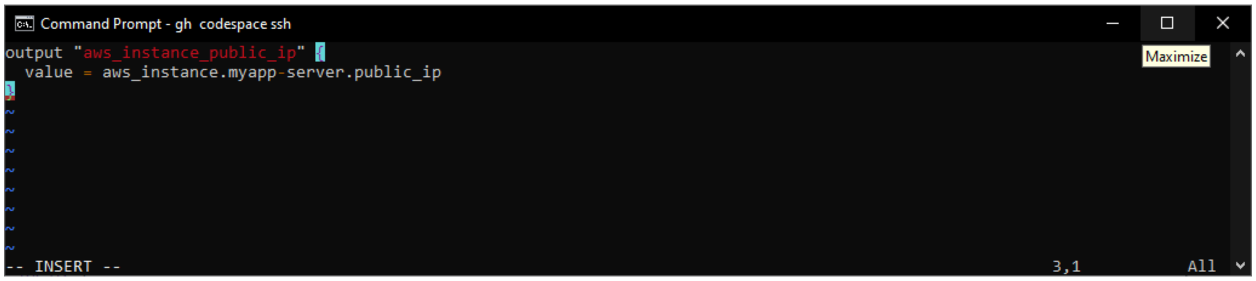
variable "private_key" {}


Command Prompt - gh codespace ssh

```
variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "instance_type" {}
variable "public_key" {}
variable "private_key" {}
~
~
~
~
~
~
~
~
~
-- INSERT --                                                                                   7,26          All
```

4. Create outputs.tf with the following content:

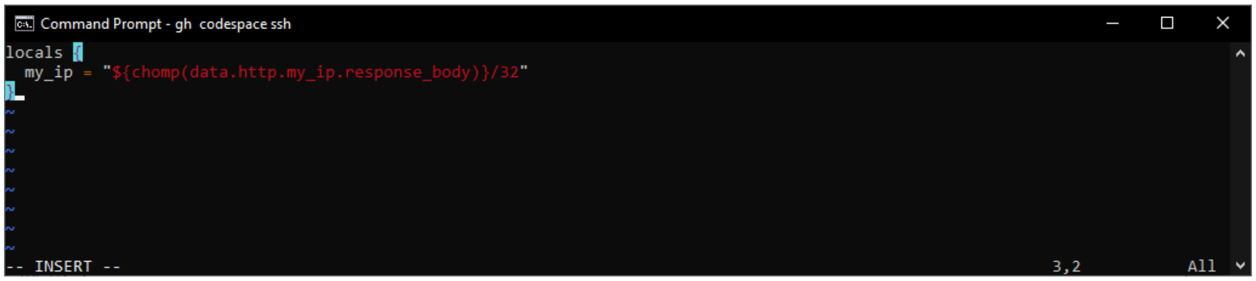
```
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```


Command Prompt - gh codespace ssh

```
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
~
~
~
~
~
~
~
~
-- INSERT --                                                                                   3,1           All
```

5. Create locals.tf with the following content:

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


Command Prompt - gh codespace ssh

```
locals {
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}
~
~
~
~
~
~
~
~
-- INSERT --                                                                                   3,2           All
```

6. Create terraform.tfvars with the following content:

```
vpc_cidr_block = "10.0.0.0/16"

subnet_cidr_block = "10.0.10.0/24"
```

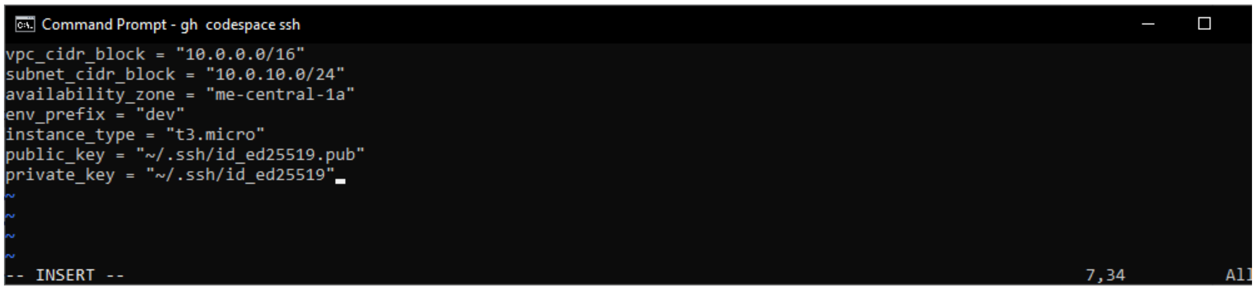
availability_zone = "me-central-1a"

env_prefix = "dev"

instance_type = "t3.micro"

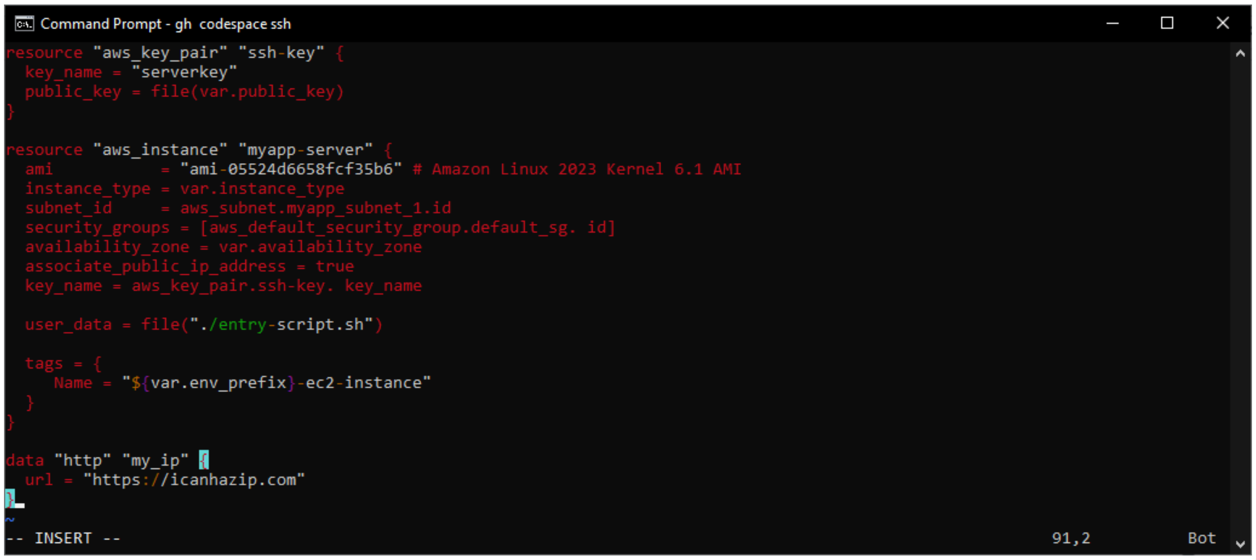
public_key = "~/.ssh/id_ed25519.pub"

private_key = "~/.ssh/id_ed25519"



```
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"
~
~
~
~
-- INSERT --                                                              7,34          All
```

7. Create main.tf with the following content:



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

resource "aws_instance" "myapp-server" {
  ami             = "ami-05524d6658fcf35b6" # Amazon Linux 2023 Kernel 6.1 AMI
  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"
}
~
-- INSERT --                                                          91,2          Bot
```

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

#!/bin/bash

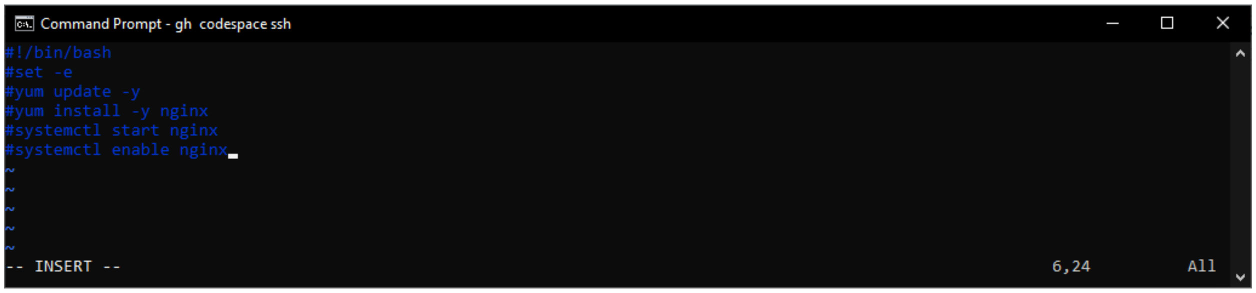
set -e

yum update -y

yum install -y nginx

systemctl start nginx

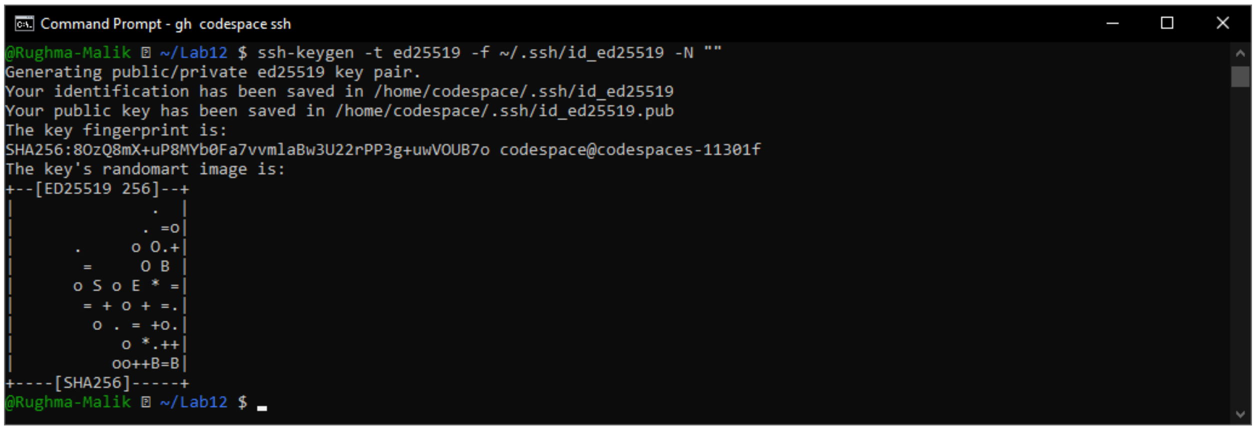
systemctl enable nginx



```
#!/bin/bash
#set -e
#yum update -y
#yum install -y nginx
#systemctl start nginx
#systemctl enable nginx
~
~
~
~
~
-- INSERT --                                                    6,24          All
```

9. Generate SSH key pair if not already exists:

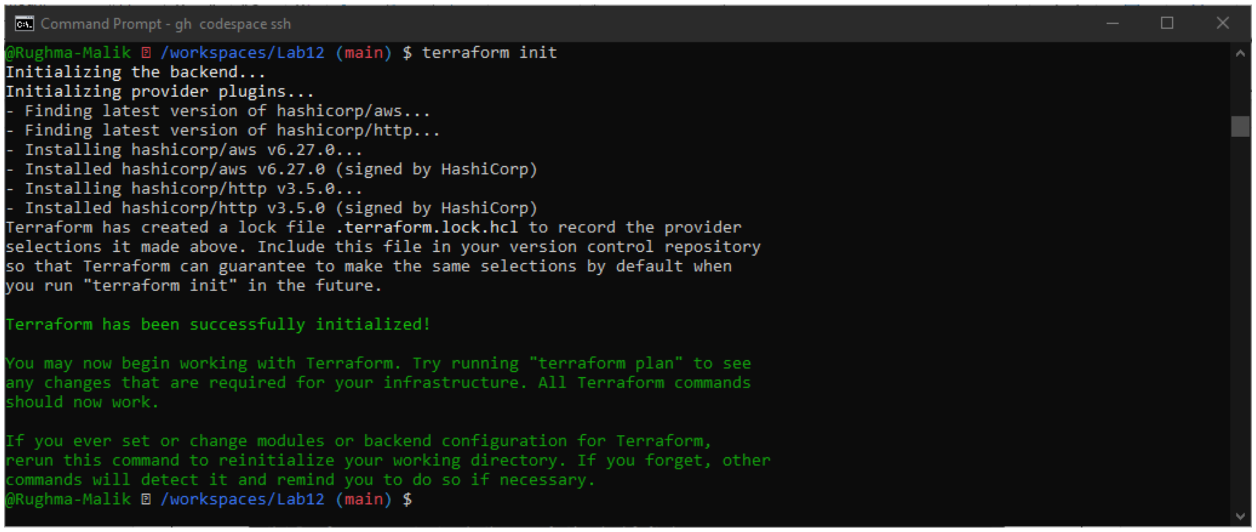
ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""



```
@Rughma-Malik ➜ ~/Lab12 $ 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:
SHA256:8OzQ8mX+uP8MYb0Fa7vvmlaBw3U22rPP3g+uwVOUB7o codespace@codespaces-11301f
The key's randomart image is:
+--[ED25519 256]--+
|               . |
|             . =o|
|       .     o O.+|
|        =     O B |
|      o S o E * =|
|       = + o + =.|
|        o . = +o.|
|           o *.++|
|          oo++B=B|
+----[SHA256]-----+
@Rughma-Malik ➜ ~/Lab12 $
```

10. Initialize Terraform:

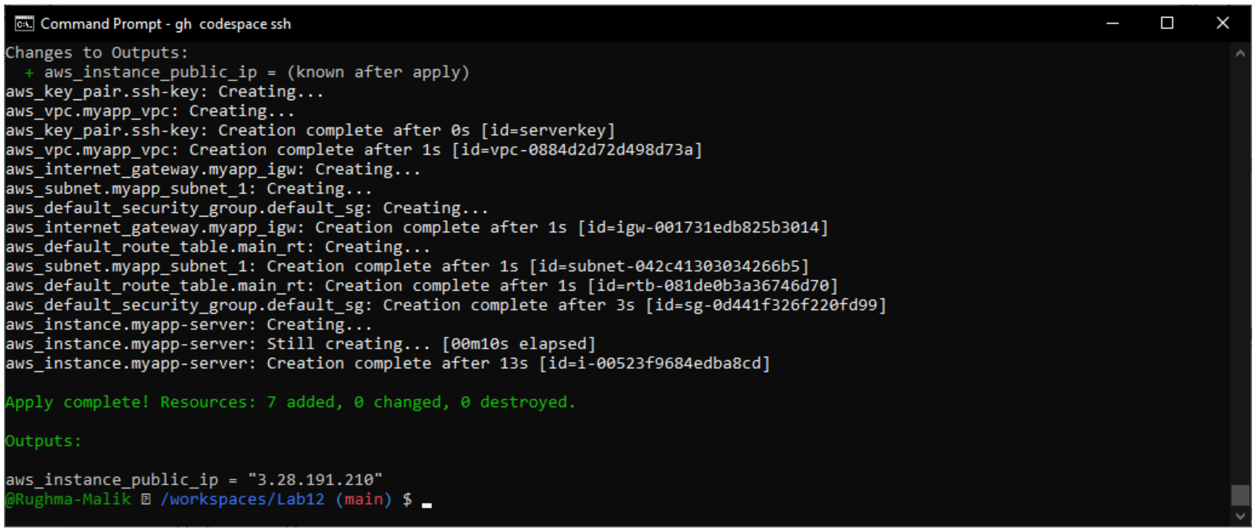
terraform init



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

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
@Rughma-Malik ➜ /workspaces/Lab12 (main) $
```

11. Apply the configuration:

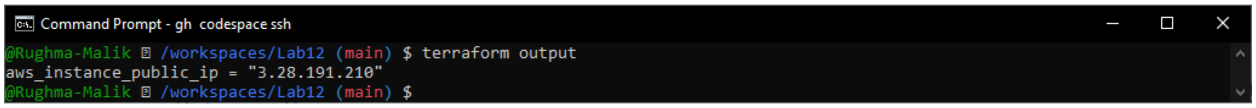
terraform apply -auto-approve



```
Changes to Outputs:
  + aws_instance_public_ip = (known after apply)
aws_key_pair.ssh-key: Creating...
aws_vpc.myapp_vpc: Creating...
aws_key_pair.ssh-key: Creation complete after 0s [id=serverkey]
aws_vpc.myapp_vpc: Creation complete after 1s [id=vpc-0884d2d72d498d73a]
aws_internet_gateway.myapp_igw: Creating...
aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-001731edb825b3014]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-042c41303034266b5]
aws_default_route_table.main_rt: Creation complete after 1s [id=rtb-081de0b3a36746d70]
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-0d441f326f220fd99]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-00523f9684edba8cd]

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

Outputs:

aws_instance_public_ip = "3.28.191.210"
@Rughma-Malik ➜ /workspaces/Lab12 (main) $ _
```

12. Display the output:

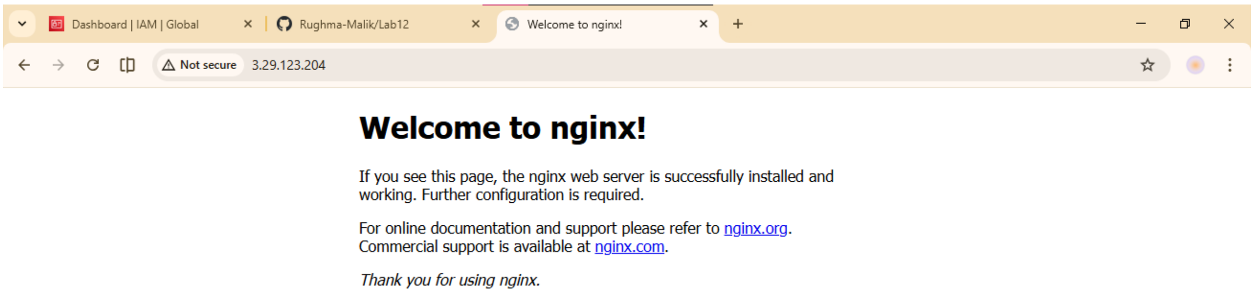
terraform output



```
@Rughma-Malik ➜ /workspaces/Lab12 (main) $ terraform output
aws_instance_public_ip = "3.28.191.210"
@Rughma-Malik ➜ /workspaces/Lab12 (main) $
```

13. Test nginx in browser:

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



Welcome to nginx!

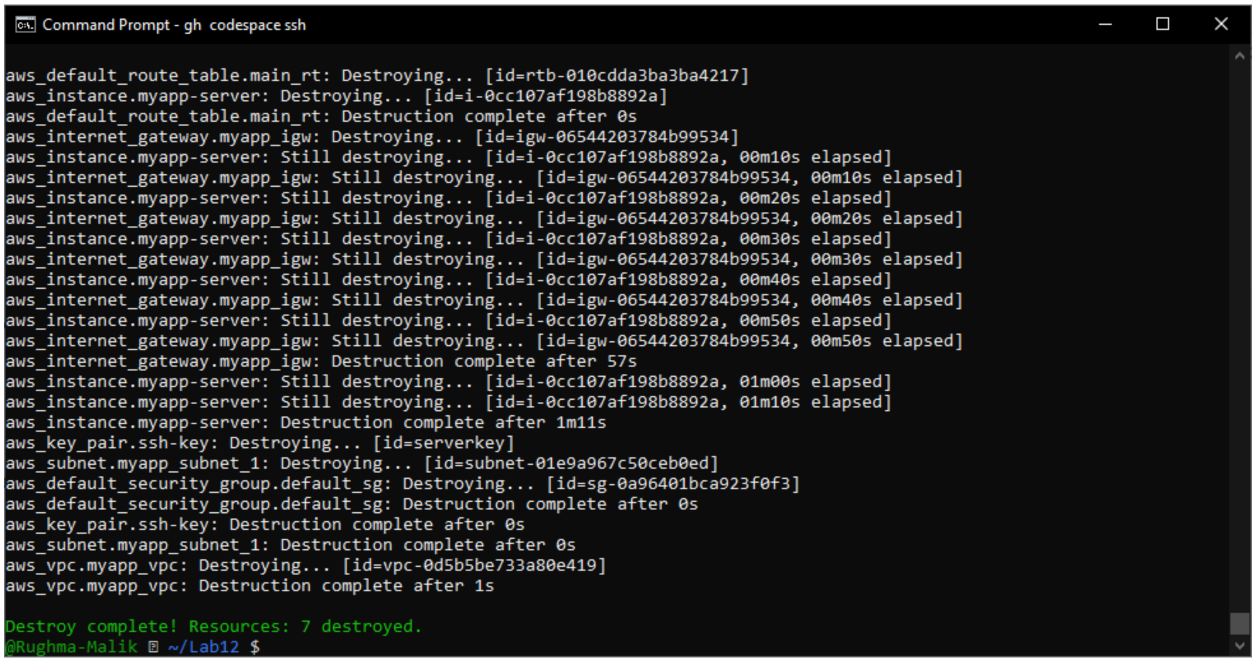
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

Thank you for using nginx.

14. Destroy resources:

terraform destroy



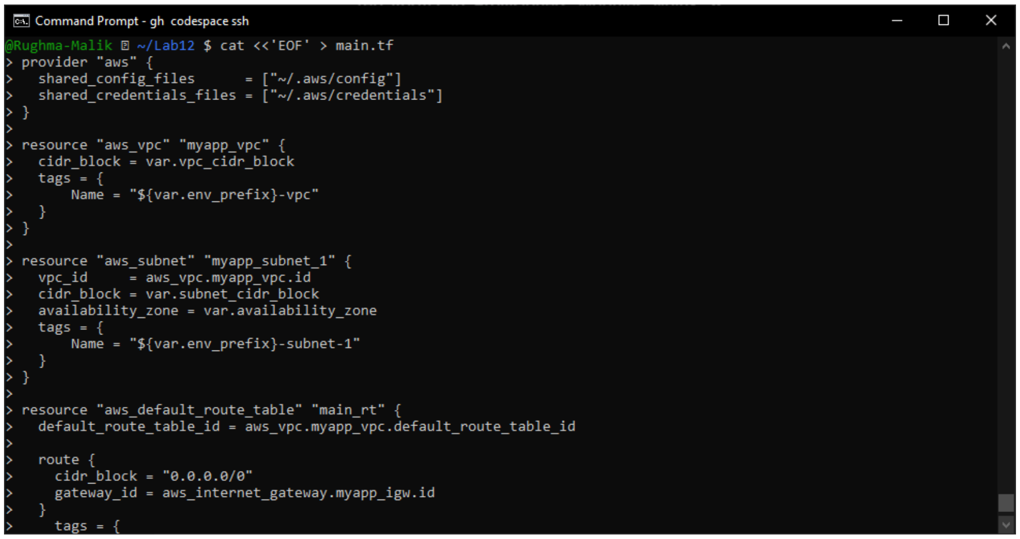
```
aws_default_route_table.main_rt: Destroying... [id=rtb-010cdda3ba3ba4217]
aws_instance.myapp-server: Destroying... [id=i-0cc107af198b8892a]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-06544203784b99534]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06544203784b99534, 00m10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06544203784b99534, 00m20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 00m30s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06544203784b99534, 00m30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 00m40s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06544203784b99534, 00m40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 00m50s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06544203784b99534, 00m50s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete 57s
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 01m00s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0cc107af198b8892a, 01m10s elapsed]
aws_instance.myapp-server: Destruction complete after 1m11s
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-01e9a967c50ceb0ed]
aws_default_security_group.default_sg: Destroying... [id=sg-0a96401bca923f0f3]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh-key: Destruction complete after 0s
aws_subnet.myapp_subnet_1: Destruction complete after 0s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0d5b5be733a80e419]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@Rughma-Malik ➜ ~/Lab12 $
```

Task 2 — Use remote-exec provisioner

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

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

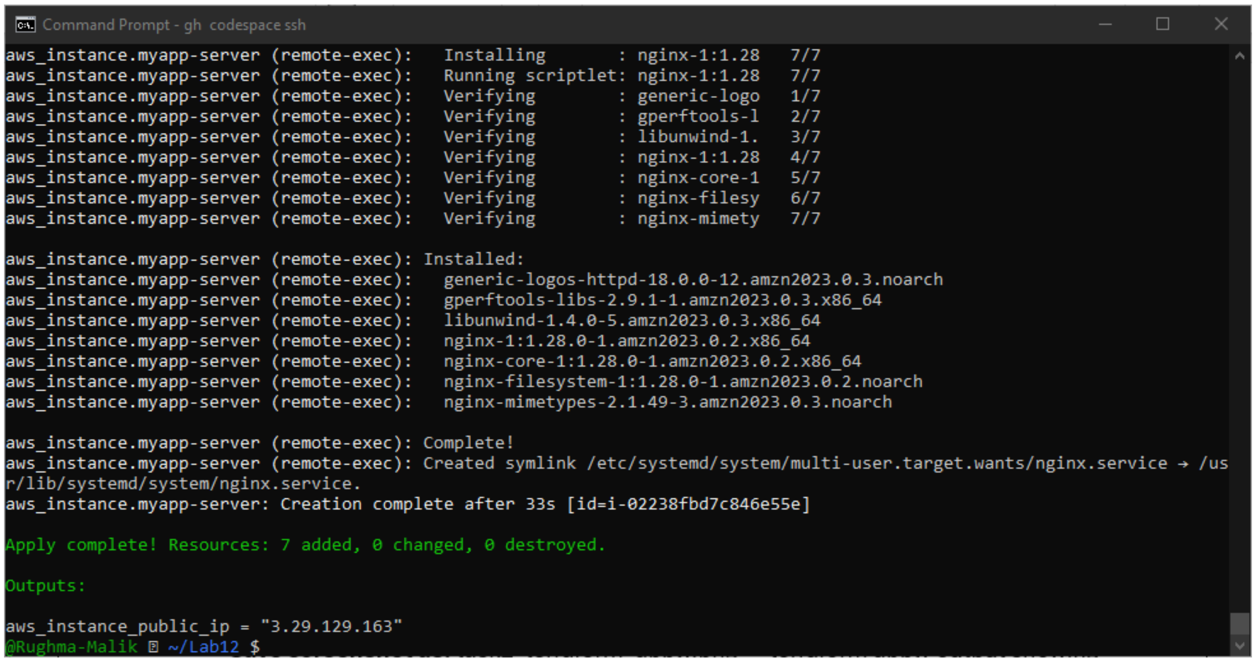
Replace the user_data line with the following provisioner block:



```
@Rughma-Malik ➜ ~/Lab12 $ cat <<'EOF' > 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
>   }
>     tags = {
```

2. Apply the configuration:

terraform apply -auto-approve

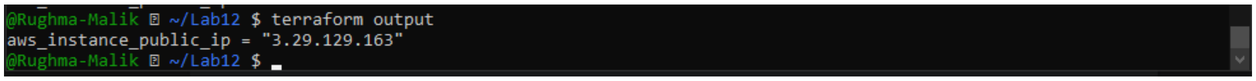


Command Prompt - gh codespace ssh

```
aws_instance.myapp-server (remote-exec):   Installing       : nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Running scriptlet: nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Verifying        : generic-logo   1/7
aws_instance.myapp-server (remote-exec):   Verifying        : gperftools-l   2/7
aws_instance.myapp-server (remote-exec):   Verifying        : libunwind-1.   3/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-1:1.28   4/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-core-1   5/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-filesy   6/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-mimety   7/7

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

aws_instance.myapp-server (remote-exec): Complete!
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /us
r/lib/systemd/system/nginx.service.
aws_instance.myapp-server: Creation complete after 33s [id=i-02238fbd7c846e55e]

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

Outputs:

aws_instance_public_ip = "3.29.129.163"
@Rughma-Malik ➜ ~/Lab12 $
```

3. Display the output:

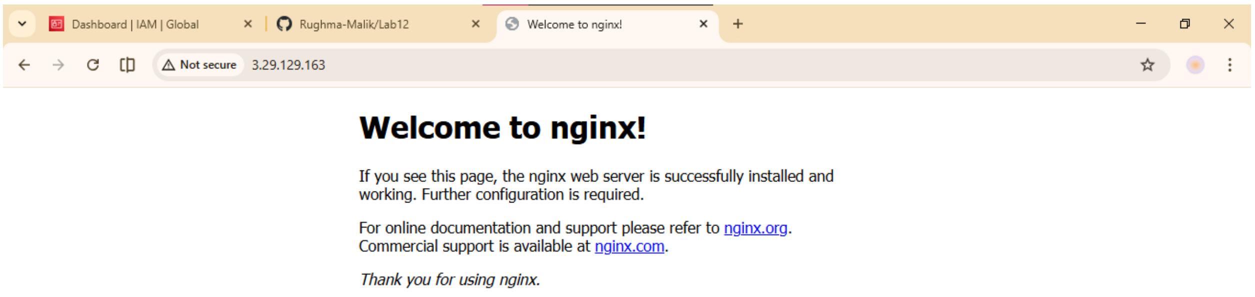
terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
aws_instance_public_ip = "3.29.129.163"
@Rughma-Malik ➜ ~/Lab12 $ _
```

4. Test nginx in browser:

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



Welcome to nginx!

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

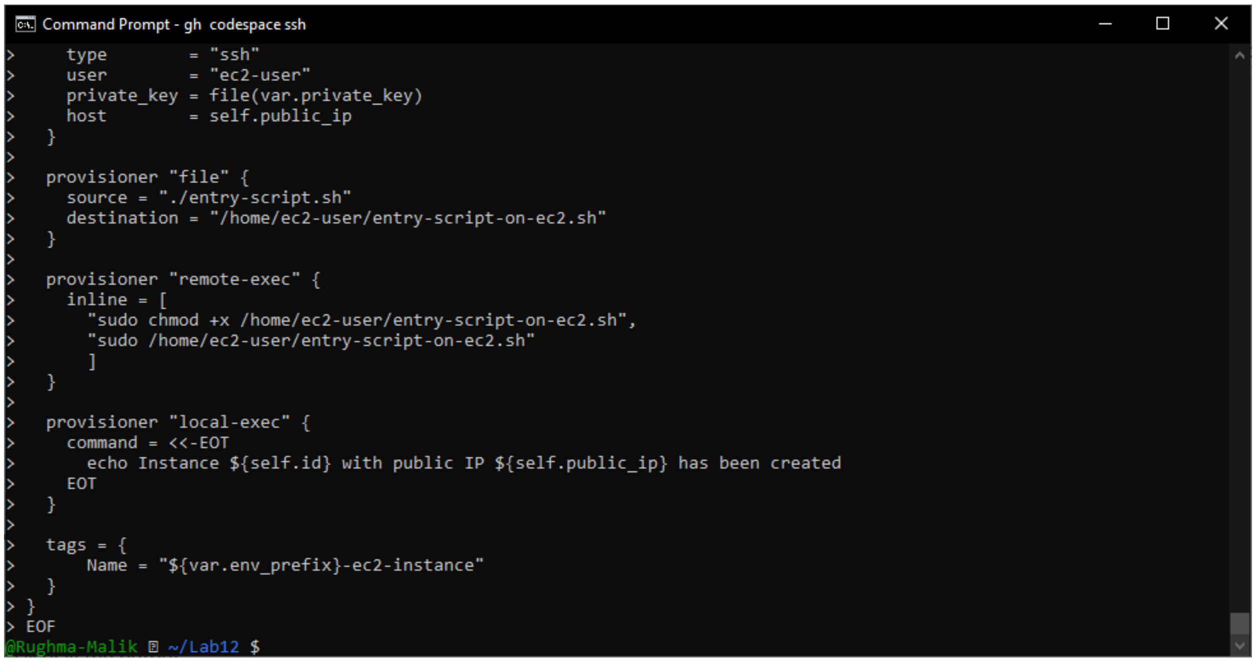
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Task 3 — Use file and local-exec provisioners

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

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

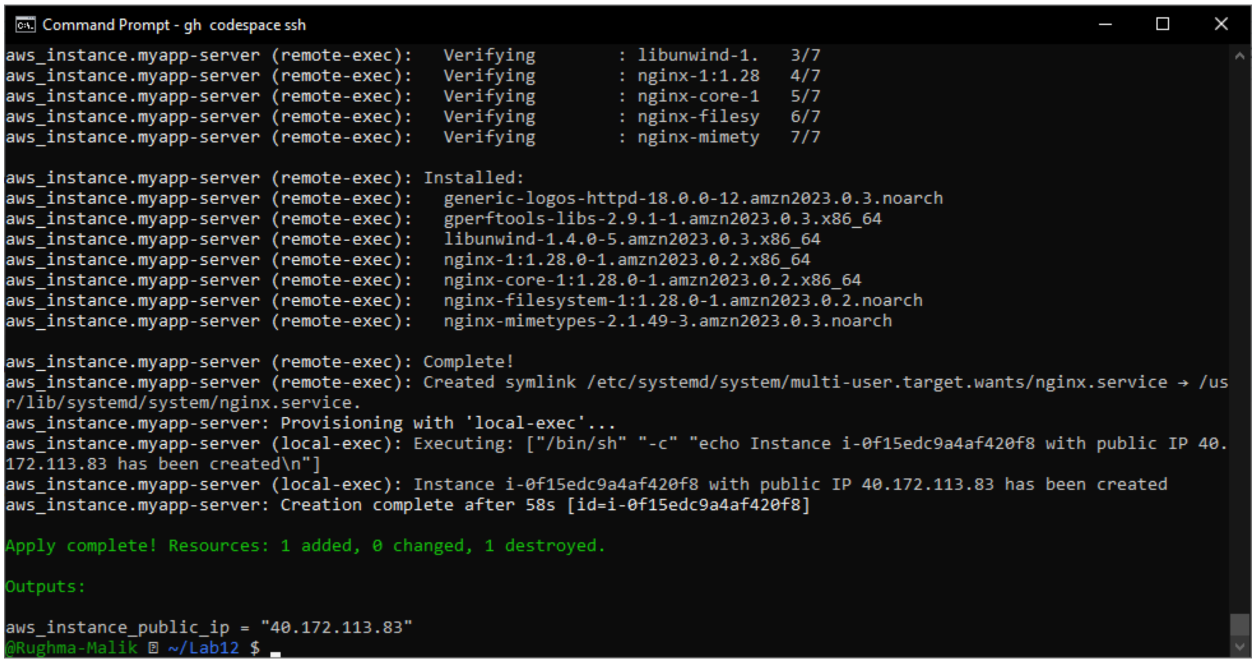


Command Prompt - gh codespace ssh

```
>     type        = "ssh"
>     user        = "ec2-user"
>     private_key = file(var.private_key)
>     host        = self.public_ip
>   }
>
>   provisioner "file" {
>     source = "./entry-script.sh"
>     destination = "/home/ec2-user/entry-script-on-ec2.sh"
>   }
>
>   provisioner "remote-exec" {
>     inline = [
>       "sudo chmod +x /home/ec2-user/entry-script-on-ec2.sh",
>       "sudo /home/ec2-user/entry-script-on-ec2.sh"
>       ]
>   }
>
>   provisioner "local-exec" {
>     command = <<-EOT
>       echo Instance ${self.id} with public IP ${self.public_ip} has been created
>     EOT
>   }
>
>   tags = {
>       Name = "${var.env_prefix}-ec2-instance"
>   }
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

2. Apply the configuration:

terraform apply -auto-approve

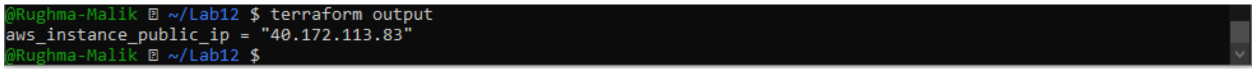


Command Prompt - gh codespace ssh

```
aws_instance.myapp-server (remote-exec):   Verifying        : libunwind-1.   3/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-1:1.28   4/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-core-1   5/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-filesy   6/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-mimety   7/7

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

aws_instance.myapp-server (remote-exec): Complete!
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /us
r/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-0f15edc9a4af420f8 with public IP 40.
172.113.83 has been created\n"]
aws_instance.myapp-server (local-exec): Instance i-0f15edc9a4af420f8 with public IP 40.172.113.83 has been created
aws_instance.myapp-server: Creation complete after 58s [id=i-0f15edc9a4af420f8]

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

Outputs:

aws_instance_public_ip = "40.172.113.83"
@Rughma-Malik ➜ ~/Lab12 $
```

3. Display the output:

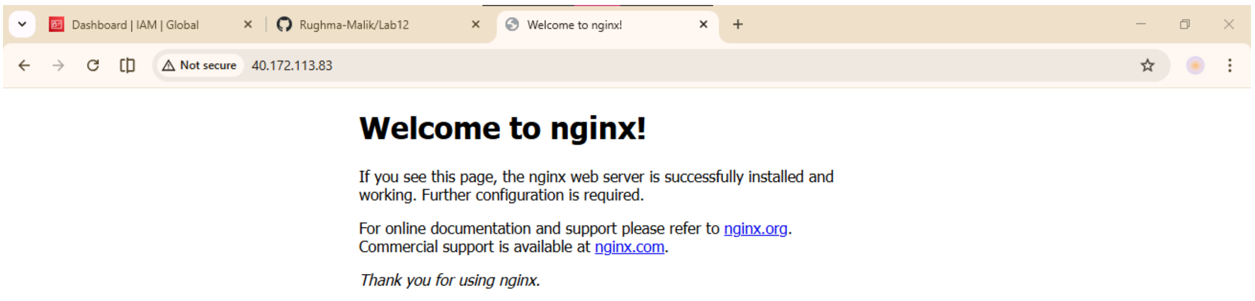
terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
aws_instance_public_ip = "40.172.113.83"
@Rughma-Malik ➜ ~/Lab12 $
```

4. Test nginx in browser:

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



Welcome to nginx!

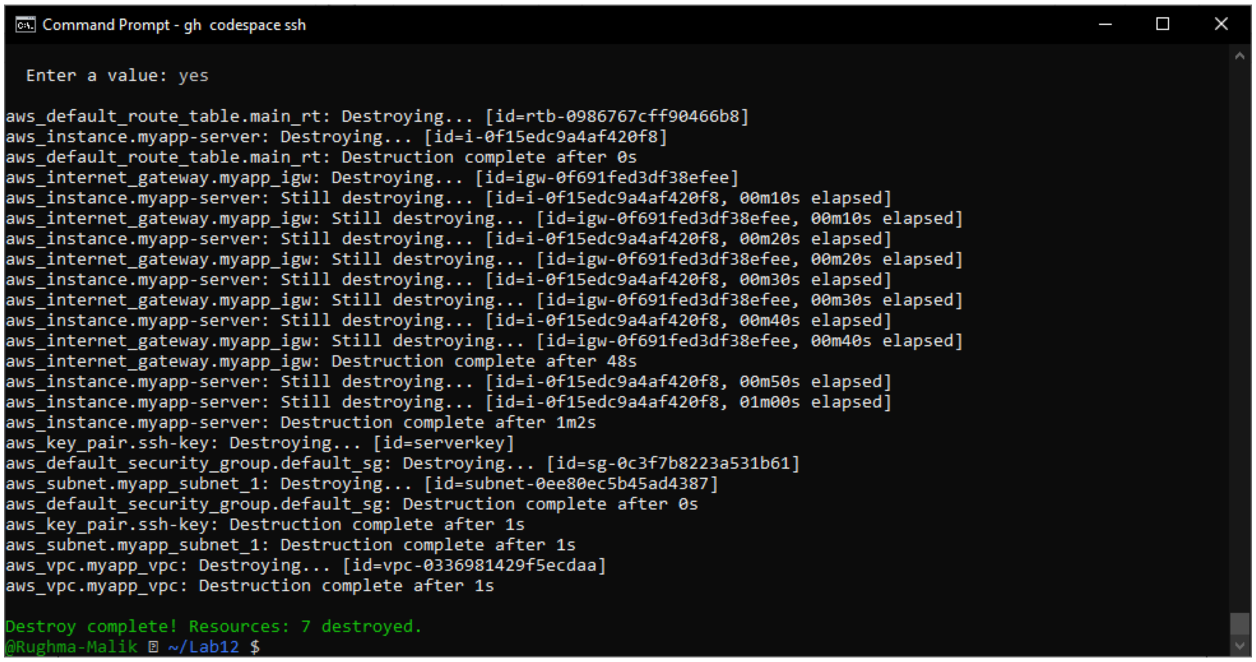
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

Thank you for using nginx.

5. Destroy the resources:

terraform destroy

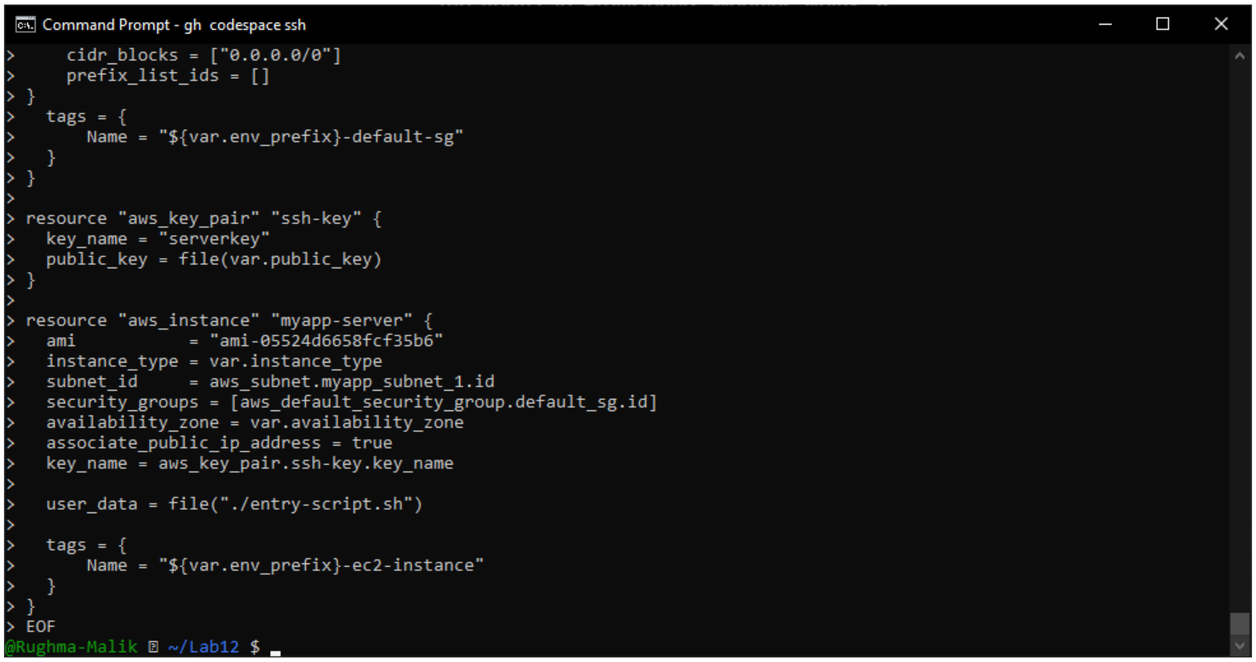


```
  Enter a value: yes

aws_default_route_table.main_rt: Destroying... [id=rtb-0986767cff90466b8]
aws_instance.myapp-server: Destroying... [id=i-0f15edc9a4af420f8]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-0f691fed3df38efee]
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0f691fed3df38efee, 00m10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0f691fed3df38efee, 00m20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 00m30s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0f691fed3df38efee, 00m30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 00m40s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0f691fed3df38efee, 00m40s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 48s
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 00m50s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0f15edc9a4af420f8, 01m00s elapsed]
aws_instance.myapp-server: Destruction complete after 1m2s
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_default_security_group.default_sg: Destroying... [id=sg-0c3f7b8223a531b61]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0ee80ec5b45ad4387]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh-key: Destruction complete after 1s
aws_subnet.myapp_subnet_1: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0336981429f5ecdaa]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@Rughma-Malik ➜ ~/Lab12 $
```

6. Remove the provisioners and restore user_data:

Replace the connection and provisioner blocks with:

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



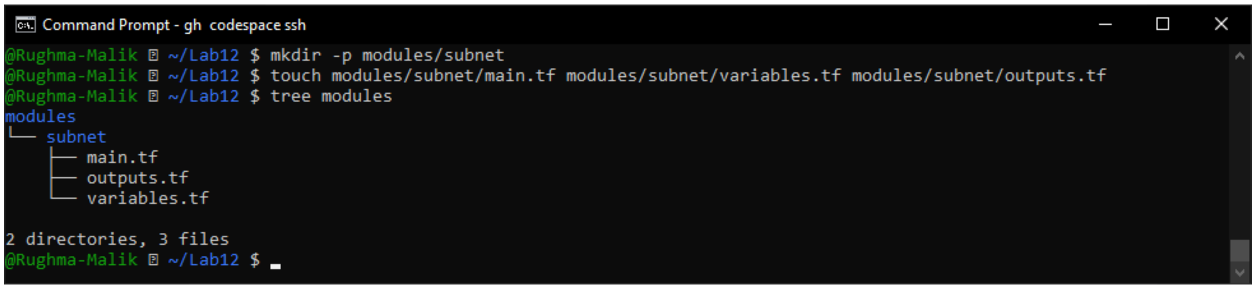
Command Prompt - gh codespace ssh

```
>     cidr_blocks = ["0.0.0.0/0"]
>     prefix_list_ids = []
> }
>   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"
>   }
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $ _
```

Task 4 — Create Terraform modules (subnet module)

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

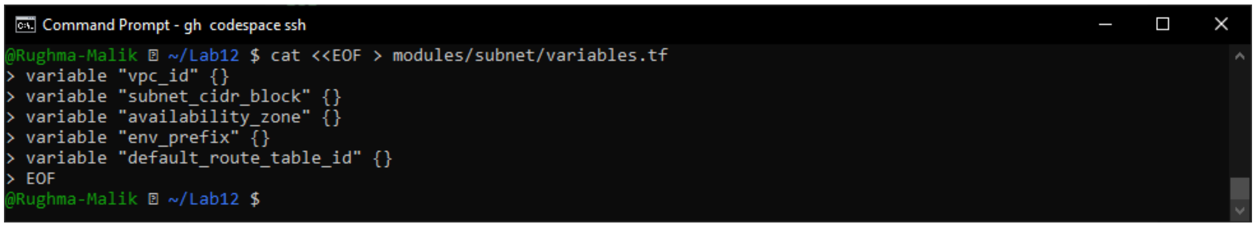
1. Create the module directory structure:



Command Prompt - gh codespace ssh

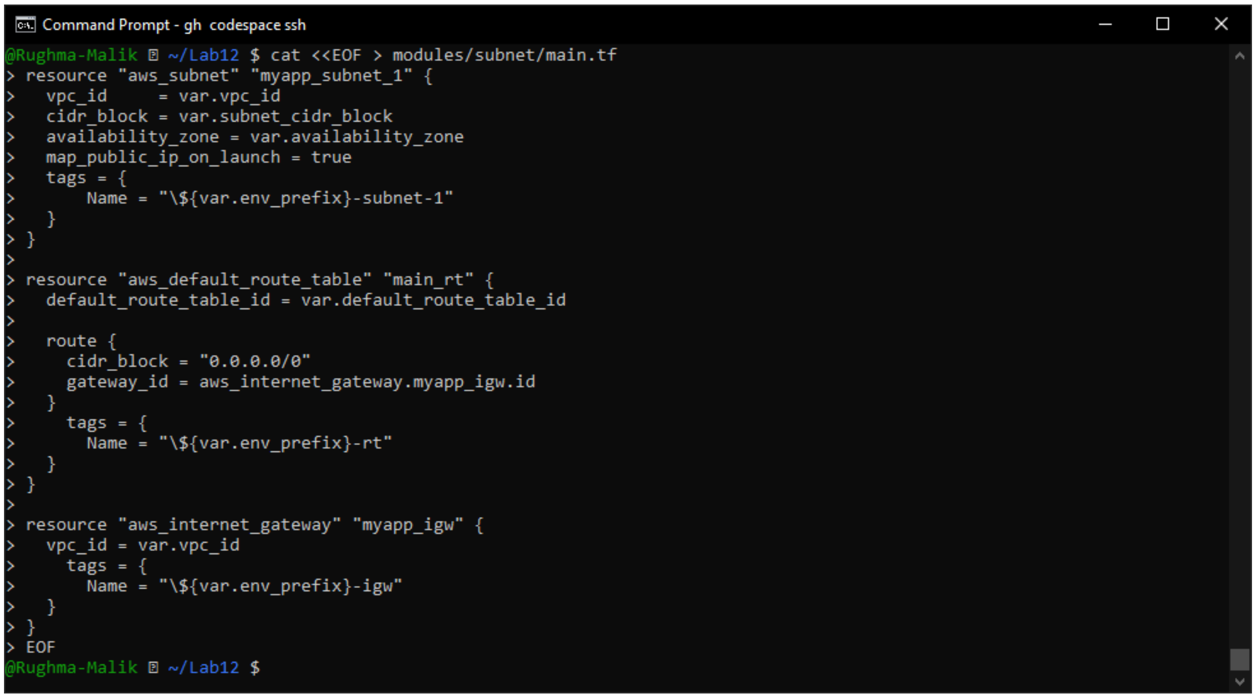
```
@Rughma-Malik ➜ ~/Lab12 $ mkdir -p modules/subnet
@Rughma-Malik ➜ ~/Lab12 $ touch modules/subnet/main.tf modules/subnet/variables.tf modules/subnet/outputs.tf
@Rughma-Malik ➜ ~/Lab12 $ tree modules
modules
└── subnet
    ├── main.tf
    ├── outputs.tf
    └── variables.tf

2 directories, 3 files
@Rughma-Malik ➜ ~/Lab12 $ _
```

2. Create modules/subnet/variables.tf:



```
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > modules/subnet/variables.tf
> variable "vpc_id" {}
> variable "subnet_cidr_block" {}
> variable "availability_zone" {}
> variable "env_prefix" {}
> variable "default_route_table_id" {}
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

3. Create modules/subnet/main.tf:



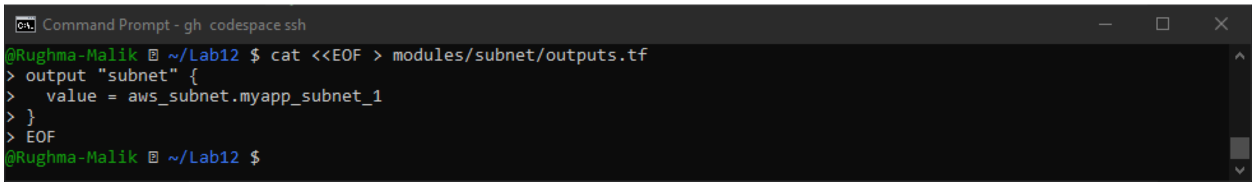
```
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > modules/subnet/main.tf
> resource "aws_subnet" "myapp_subnet_1" {
>   vpc_id     = var.vpc_id
>   cidr_block = var.subnet_cidr_block
>   availability_zone = var.availability_zone
>   map_public_ip_on_launch = true
>   tags = {
>       Name = "\${var.env_prefix}-subnet-1"
>   }
> }
>
> 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"
>   }
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

4. Create modules/subnet/outputs.tf:

output "subnet" {

 value = aws_subnet.myapp_subnet_1

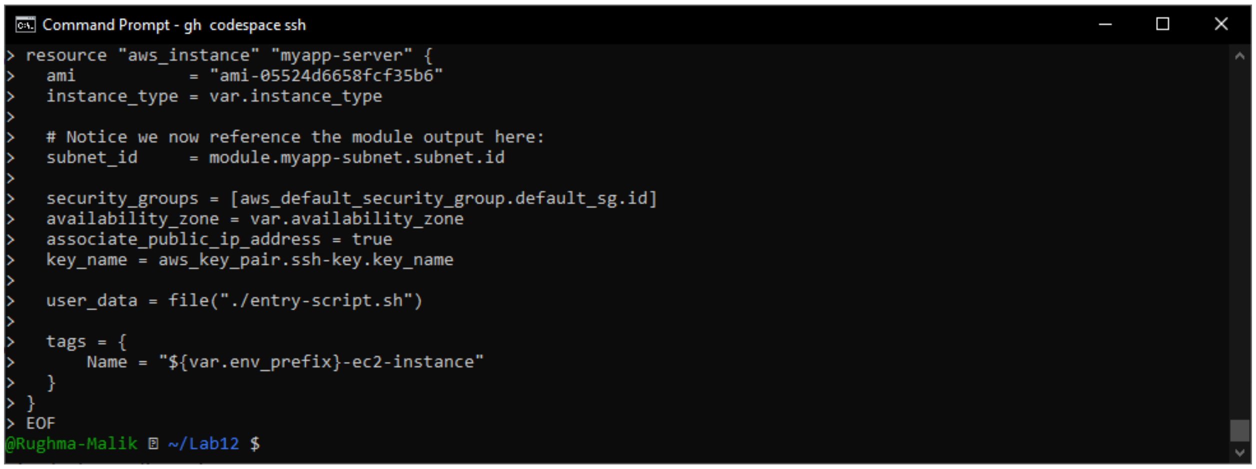
}



```
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > modules/subnet/outputs.tf
> output "subnet" {
>   value = aws_subnet.myapp_subnet_1
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

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

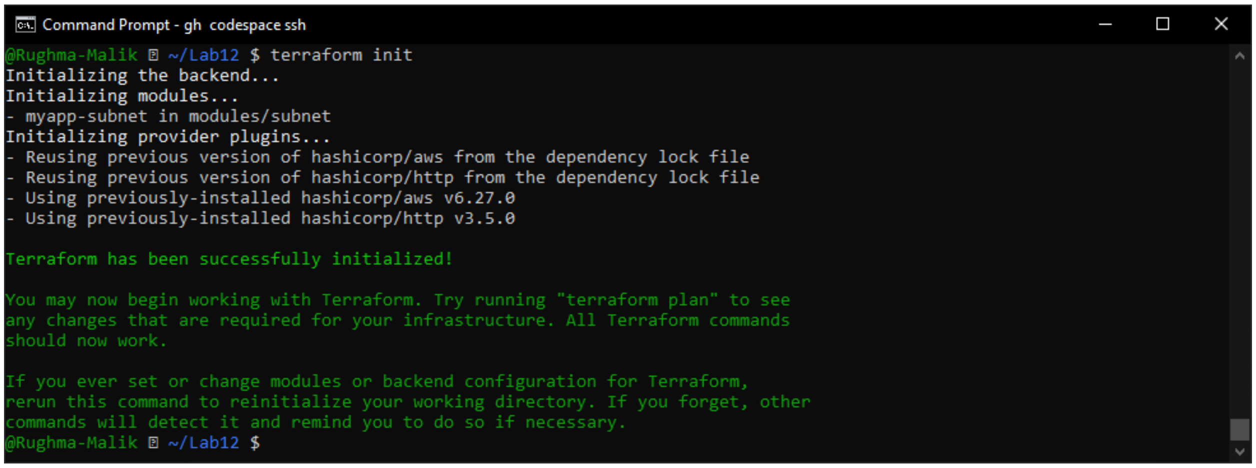
Remove the subnet, route table, and internet gateway resources and replace them with:



```
> resource "aws_instance" "myapp-server" {
>   ami           = "ami-05524d6658fcf35b6"
>   instance_type = var.instance_type
>
>   # Notice we now reference the module output here:
>   subnet_id     = module.myapp-subnet.subnet.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"
>   }
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

6. Initialize Terraform to download the module:

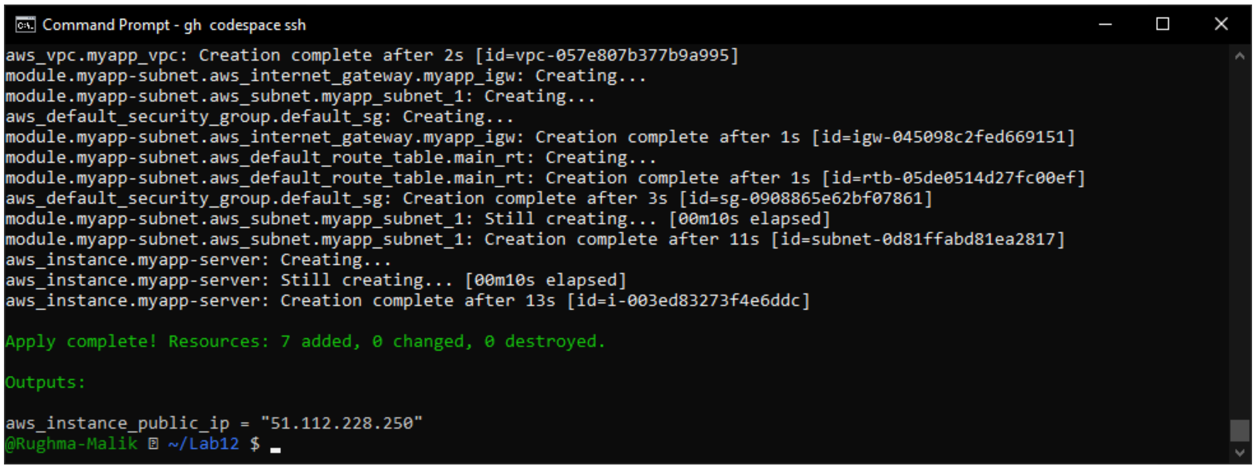
terraform init



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

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
@Rughma-Malik ➜ ~/Lab12 $
```

7. Apply the configuration:

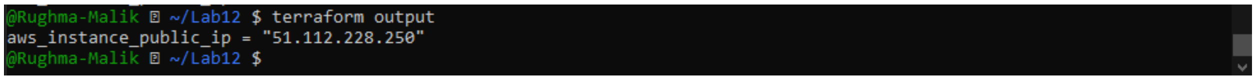
terraform apply -auto-approve



```
aws_vpc.myapp_vpc: Creation complete after 2s [id=vpc-057e807b377b9a995]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Creating...
module.myapp-subnet.aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.default_sg: Creating...
module.myapp-subnet.aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-045098c2fed669151]
module.myapp-subnet.aws_default_route_table.main_rt: Creating...
module.myapp-subnet.aws_default_route_table.main_rt: Creation complete after 1s [id=rtb-05de0514d27fc00ef]
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-0908865e62bf07861]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Still creating... [00m10s elapsed]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Creation complete after 11s [id=subnet-0d81ffabd81ea2817]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-003ed83273f4e6ddc]

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

Outputs:

aws_instance_public_ip = "51.112.228.250"
@Rughma-Malik ➜ ~/Lab12 $
```

8. Display the output:

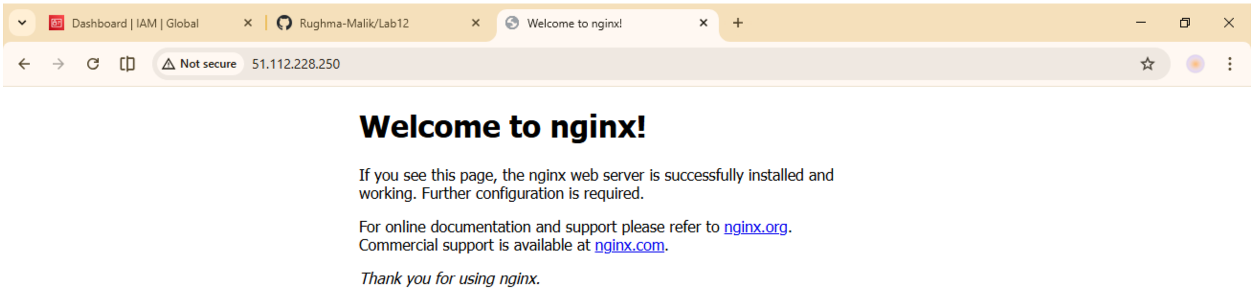
terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
aws_instance_public_ip = "51.112.228.250"
@Rughma-Malik ➜ ~/Lab12 $
```

9. Test nginx in browser:

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



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.

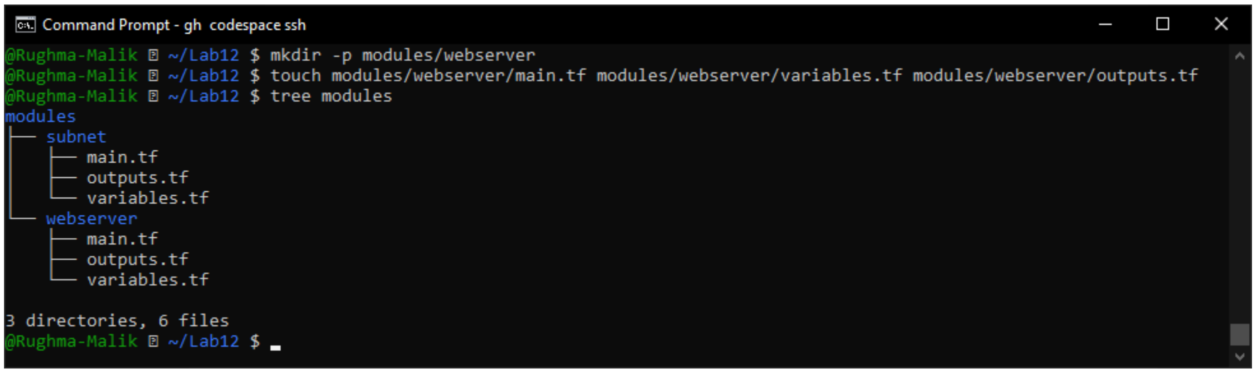
Thank you for using nginx.

Task 5 — Create webserver module

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

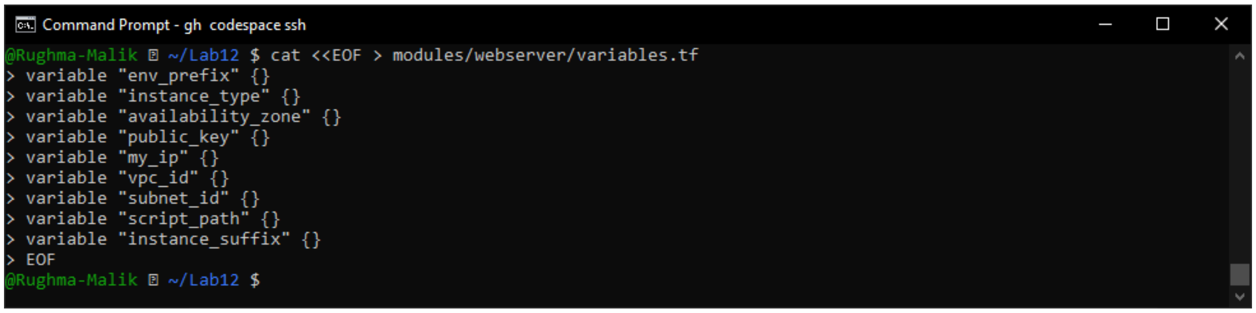
1. Create the webserver module directory structure:

mkdir -p modules/webserver



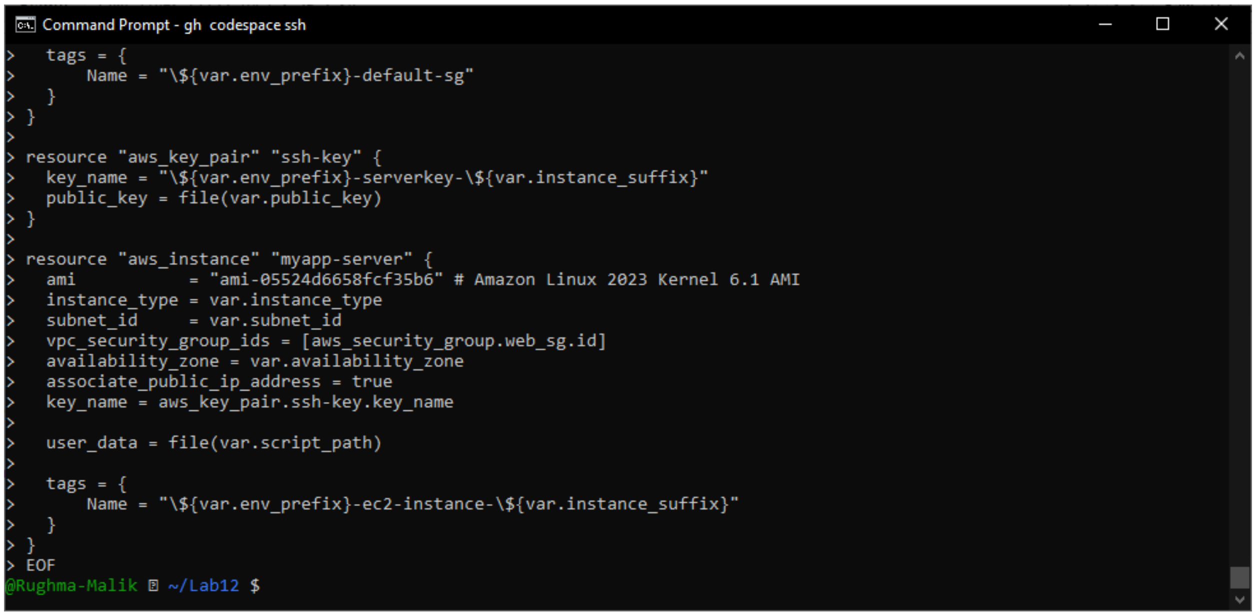
```
@Rughma-Malik ➜ ~/Lab12 $ mkdir -p modules/webserver
@Rughma-Malik ➜ ~/Lab12 $ touch modules/webserver/main.tf modules/webserver/variables.tf modules/webserver/outputs.tf
@Rughma-Malik ➜ ~/Lab12 $ tree modules
modules
├── subnet
│   ├── main.tf
│   ├── outputs.tf
│   └── variables.tf
└── webserver
    ├── main.tf
    ├── outputs.tf
    └── variables.tf

3 directories, 6 files
@Rughma-Malik ➜ ~/Lab12 $
```

2. Create modules/webserver/variables.tf:



```
Command Prompt - gh  codespace ssh
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > 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" {}
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

3. Create modules/webserver/main.tf:



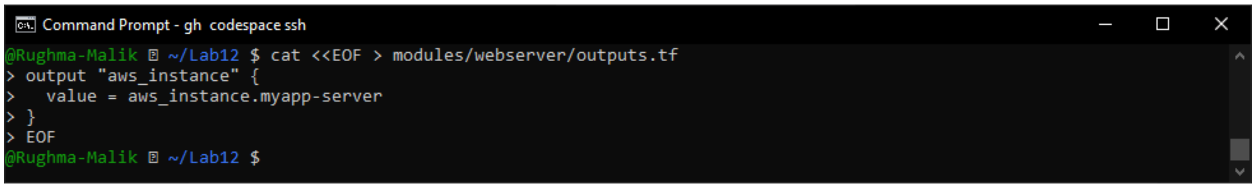
```
Command Prompt - gh  codespace ssh
>   tags = {
>       Name = "\${var.env_prefix}-default-sg"
>   }
> }
>
> resource "aws_key_pair" "ssh-key" {
>   key_name = "\${var.env_prefix}-serverkey-\${var.instance_suffix}"
>   public_key = file(var.public_key)
> }
>
> resource "aws_instance" "myapp-server" {
>   ami           = "ami-05524d6658fcf35b6" # Amazon Linux 2023 Kernel 6.1 AMI
>   instance_type = var.instance_type
>   subnet_id     = var.subnet_id
>   vpc_security_group_ids = [aws_security_group.web_sg.id]
>   availability_zone = var.availability_zone
>   associate_public_ip_address = true
>   key_name = aws_key_pair.ssh-key.key_name
>
>   user_data = file(var.script_path)
>
>   tags = {
>       Name = "\${var.env_prefix}-ec2-instance-\${var.instance_suffix}"
>   }
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

4. Create modules/webserver/outputs.tf:

output "aws_instance" {

  value = aws_instance.myapp-server

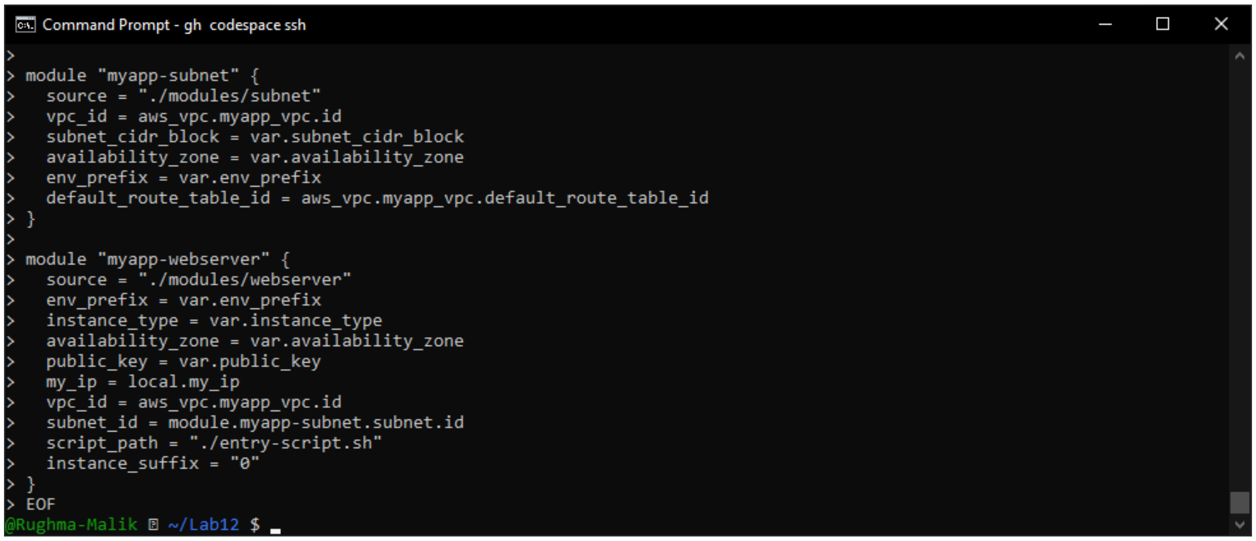
}



```
Command Prompt - gh  codespace ssh
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > modules/webserver/outputs.tf
> output "aws_instance" {
>   value = aws_instance.myapp-server
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

5. Modify the root main.tf:

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

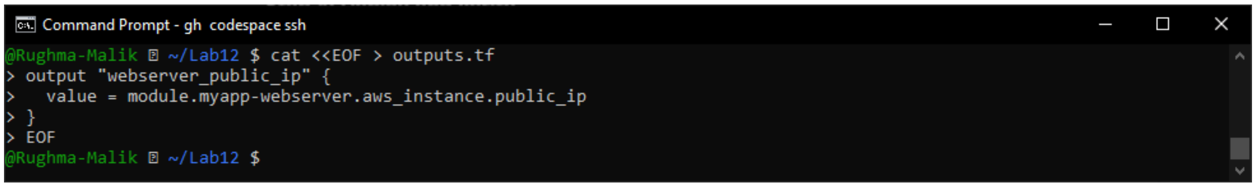

```
Command Prompt - gh  codespace ssh
>
> module "myapp-subnet" {
>   source = "./modules/subnet"
>   vpc_id = aws_vpc.myapp_vpc.id
>   subnet_cidr_block = var.subnet_cidr_block
>   availability_zone = var.availability_zone
>   env_prefix = var.env_prefix
>   default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id
> }
>
> 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"
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

6. Update outputs.tf:

output "webserver_public_ip" {

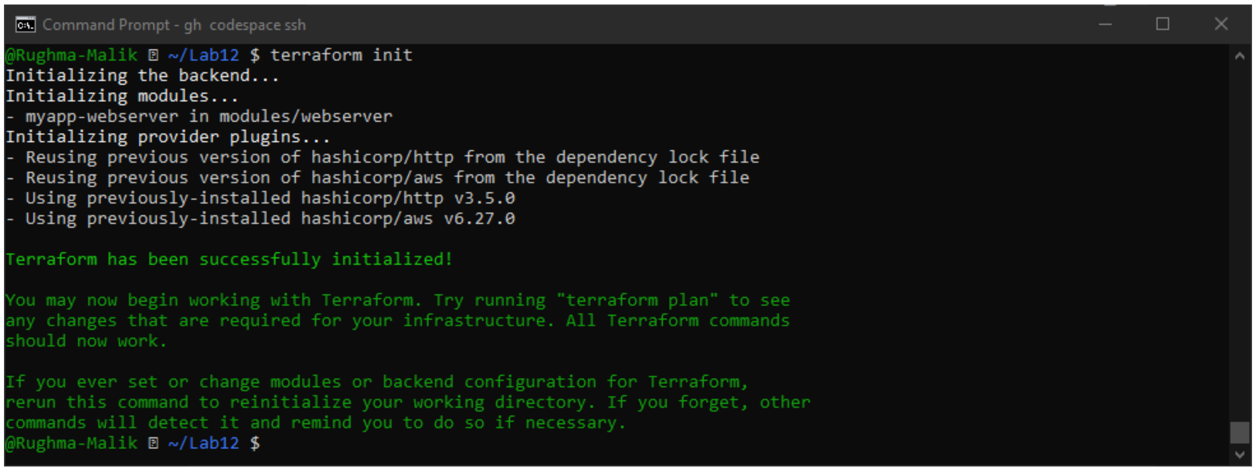
  value = module.myapp-webserver.aws_instance.public_ip

}


```
Command Prompt - gh  codespace ssh
@Rughma-Malik ➜ ~/Lab12 $ cat <<EOF > outputs.tf
> output "webserver_public_ip" {
>   value = module.myapp-webserver.aws_instance.public_ip
> }
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

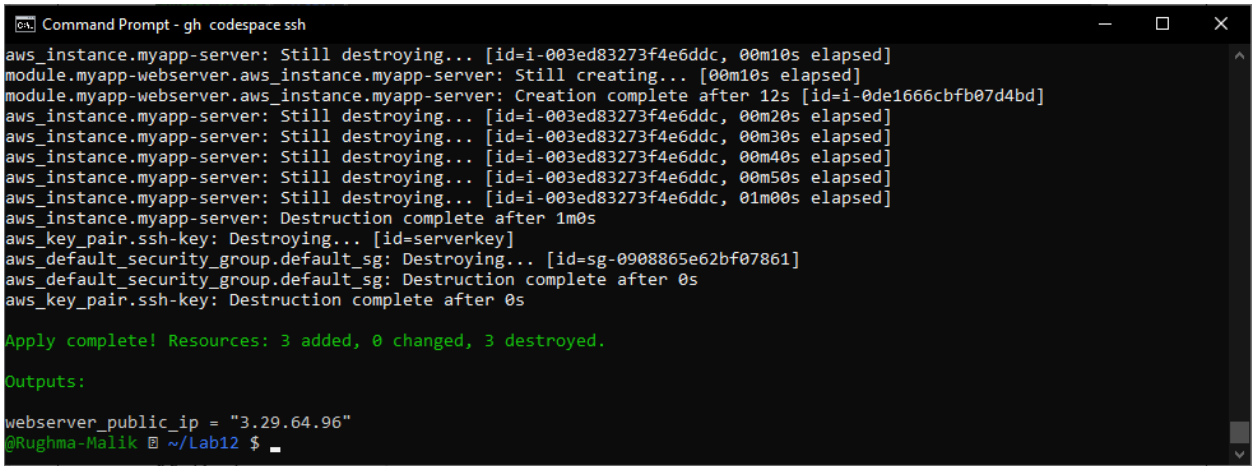
7. Initialize Terraform:

terraform init


```
Command Prompt - gh  codespace ssh
@Rughma-Malik ➜ ~/Lab12 $ terraform init
Initializing the backend...
Initializing modules...
- myapp-webserver in modules/webserver
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.
@Rughma-Malik ➜ ~/Lab12 $
```

8. Apply the configuration:

terraform apply -auto-approve

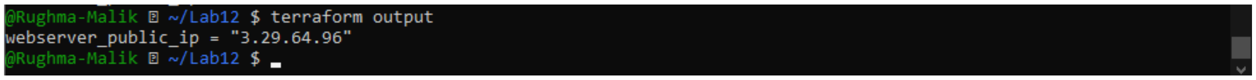


Command Prompt - gh codespace ssh

```
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 00m10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 12s [id=i-0de1666cbfb07d4bd]
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 00m20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 00m30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 00m40s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 00m50s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-003ed83273f4e6ddc, 01m00s elapsed]
aws_instance.myapp-server: Destruction complete after 1m0s
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_default_security_group.default_sg: Destroying... [id=sg-0908865e62bf07861]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh-key: Destruction complete after 0s

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

Outputs:

webserver_public_ip = "3.29.64.96"
@Rughma-Malik ➜ ~/Lab12 $
```

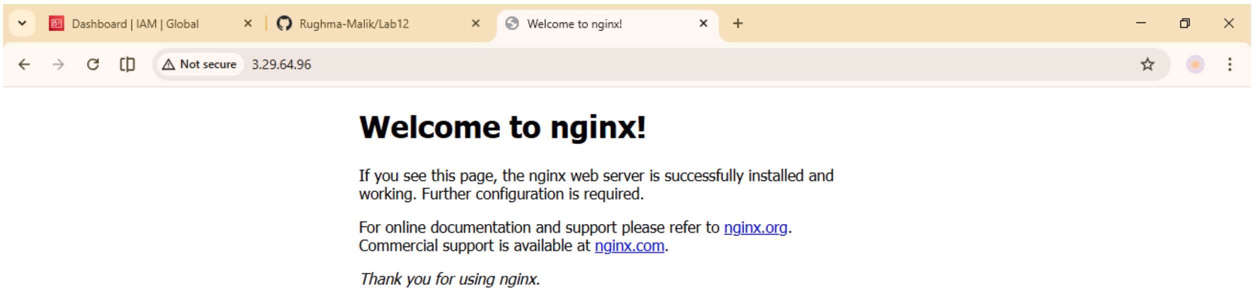
9. Display the output:

terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
webserver_public_ip = "3.29.64.96"
@Rughma-Malik ➜ ~/Lab12 $
```

10. Test nginx in browser:



Welcome to nginx!

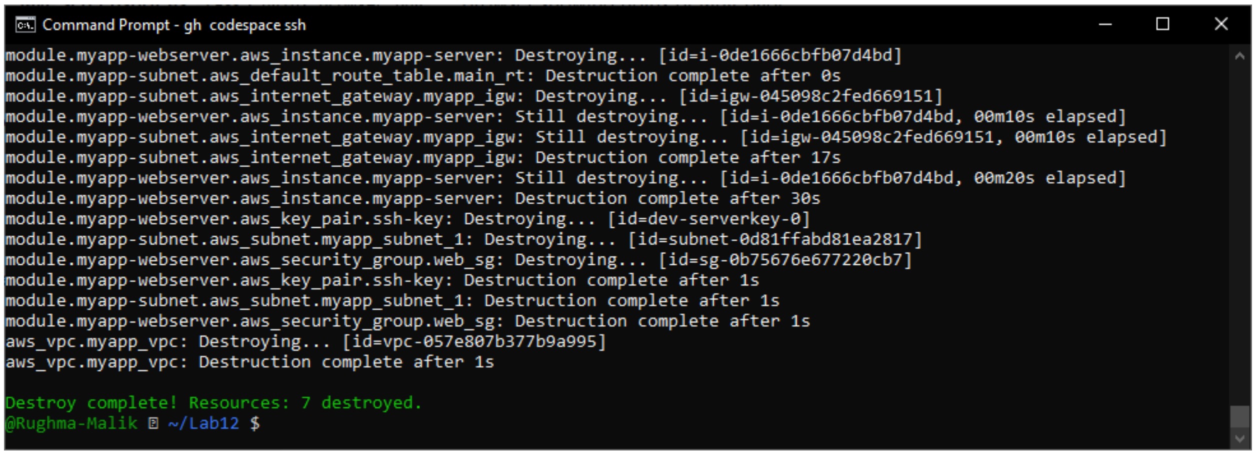
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

Thank you for using nginx.

11. Destroy resources:

terraform destroy



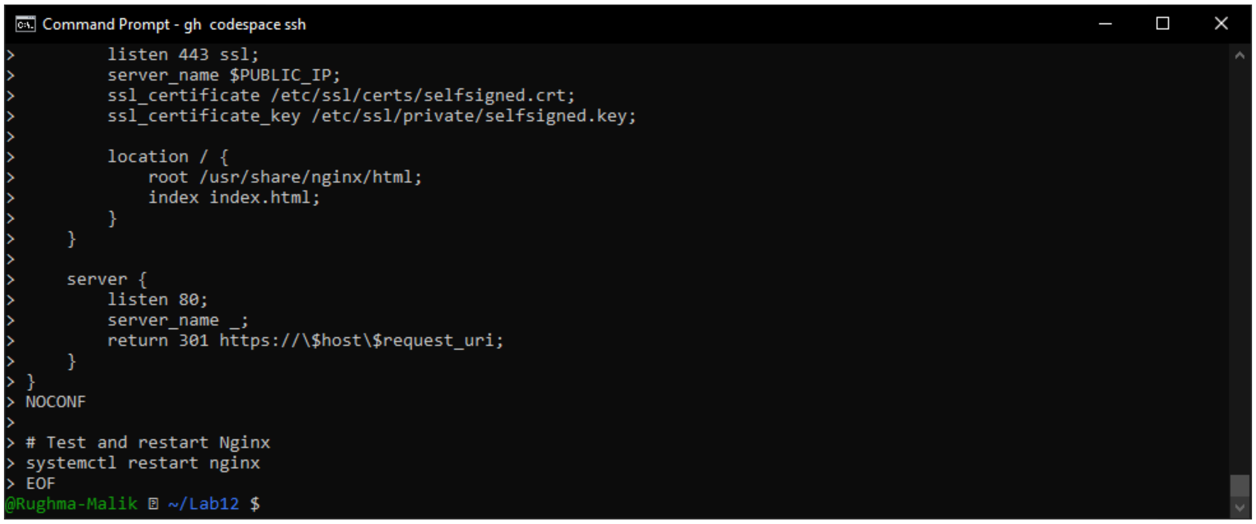
```
module.myapp-webserver.aws_instance.myapp-server: Destroying... [id=i-0de1666cbfb07d4bd]
module.myapp-subnet.aws_default_route_table.main_rt: Destruction complete after 0s
module.myapp-subnet.aws_internet_gateway.myapp_igw: Destroying... [id=igw-045098c2fed669151]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de1666cbfb07d4bd, 00m10s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-045098c2fed669151, 00m10s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Destruction complete after 17s
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-0de1666cbfb07d4bd, 00m20s 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-subnet.aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0d81ffabd81ea2817]
module.myapp-webserver.aws_security_group.web_sg: Destroying... [id=sg-0b75676e677220cb7]
module.myapp-webserver.aws_key_pair.ssh-key: Destruction complete after 1s
module.myapp-subnet.aws_subnet.myapp_subnet_1: Destruction complete after 1s
module.myapp-webserver.aws_security_group.web_sg: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-057e807b377b9a995]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@Rughma-Malik ➜ ~/Lab12 $
```

Task 6 — Configure HTTPS with self-signed certificates

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

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

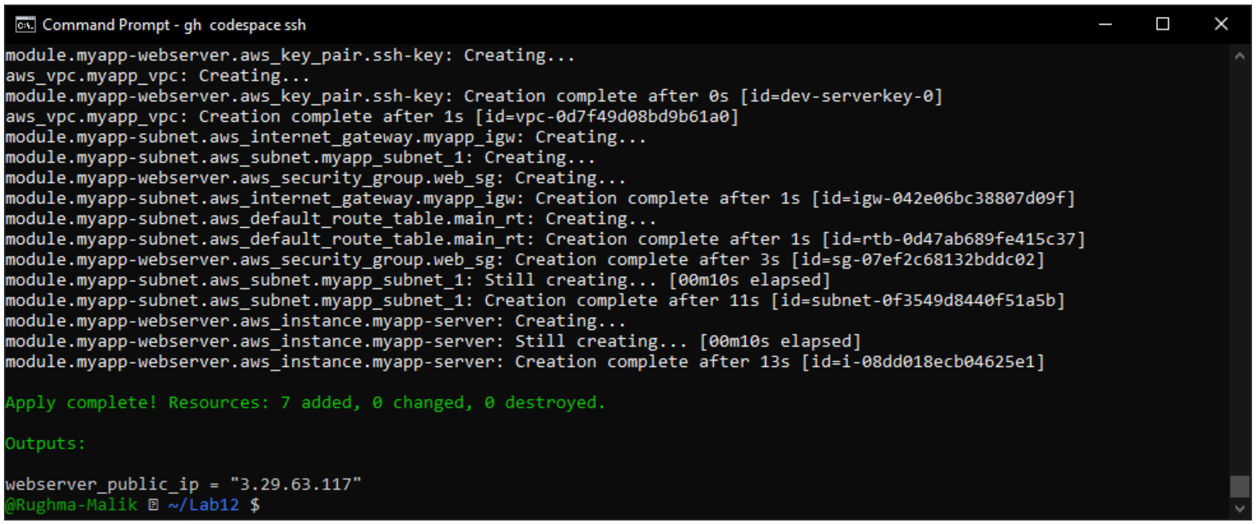
systemctl restart nginx



```
>         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;
>     }
> }
> NOCONF
>
> # Test and restart Nginx
> systemctl restart nginx
> EOF
@Rughma-Malik ➜ ~/Lab12 $
```

2. Apply the configuration:

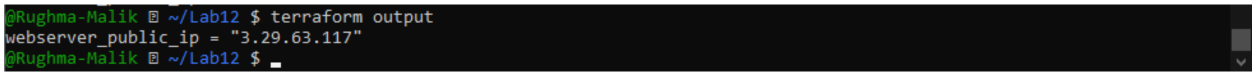
terraform apply -auto-approve


Command Prompt - gh codespace ssh

```
module.myapp-webserver.aws_key_pair.ssh-key: Creating...
aws_vpc.myapp_vpc: Creating...
module.myapp-webserver.aws_key_pair.ssh-key: Creation complete after 0s [id=dev-serverkey-0]
aws_vpc.myapp_vpc: Creation complete after 1s [id=vpc-0d7f49d08bd9b61a0]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Creating...
module.myapp-subnet.aws_subnet.myapp_subnet_1: Creating...
module.myapp-webserver.aws_security_group.web_sg: Creating...
module.myapp-subnet.aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-042e06bc38807d09f]
module.myapp-subnet.aws_default_route_table.main_rt: Creating...
module.myapp-subnet.aws_default_route_table.main_rt: Creation complete after 1s [id=rtb-0d47ab689fe415c37]
module.myapp-webserver.aws_security_group.web_sg: Creation complete after 3s [id=sg-07ef2c68132bddc02]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Still creating... [00m10s elapsed]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Creation complete after 11s [id=subnet-0f3549d8440f51a5b]
module.myapp-webserver.aws_instance.myapp-server: Creating...
module.myapp-webserver.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Creation complete after 13s [id=i-08dd018ecb04625e1]

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

Outputs:

webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $
```

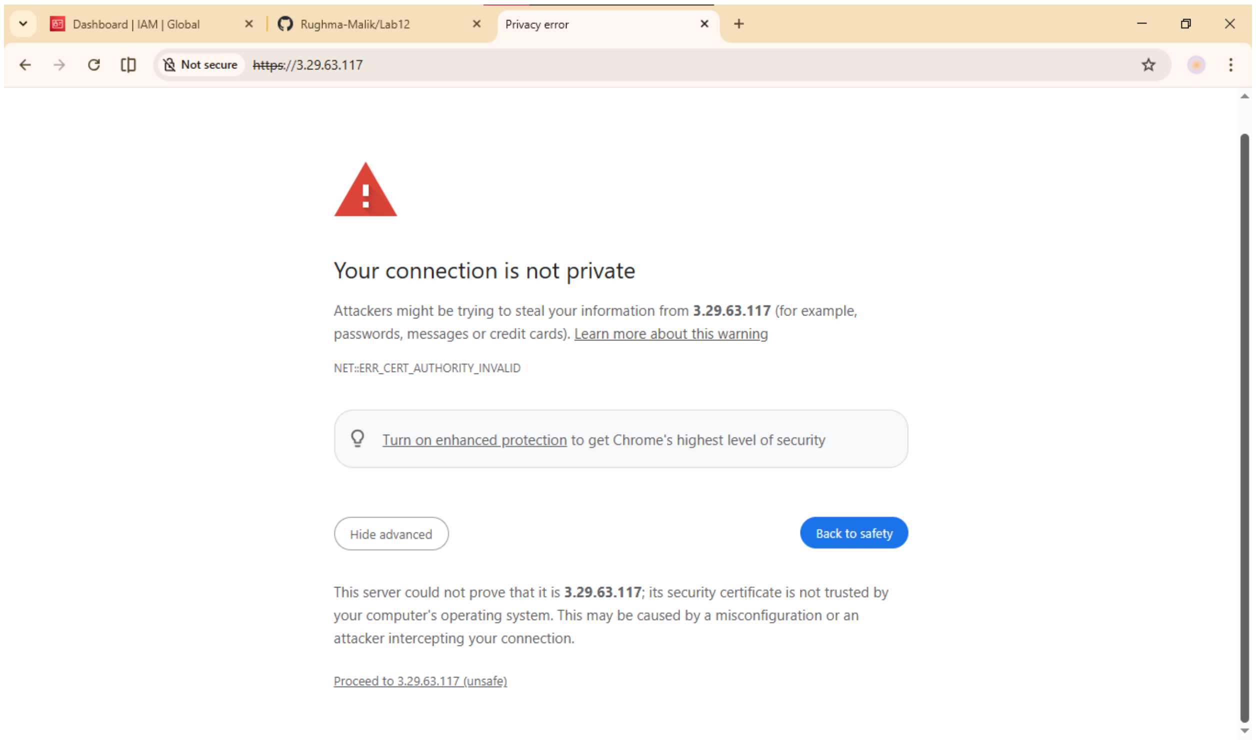
3. Display the output:

terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $ _
```

4. Test HTTPS in browser:



Your connection is not private

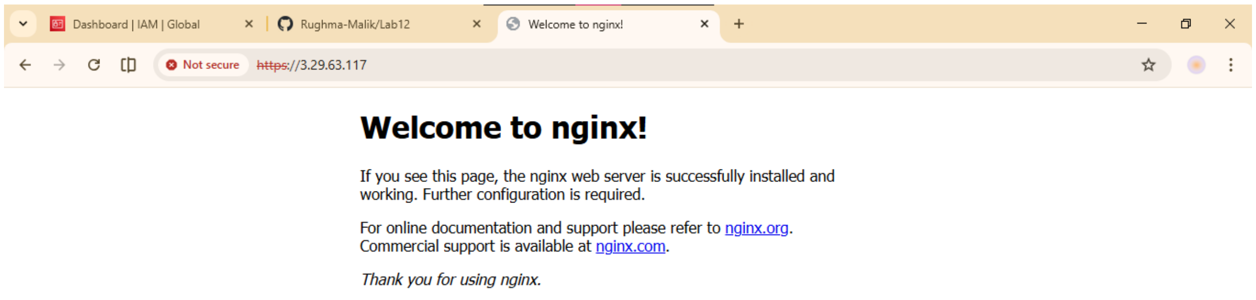
Attackers might be trying to steal your information from **3.29.63.117** (for example, passwords, messages or credit cards). Learn more about this warning

NET::ERR_CERT_AUTHORITY_INVALID

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

This server could not prove that it is **3.29.63.117**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to 3.29.63.117 (unsafe)

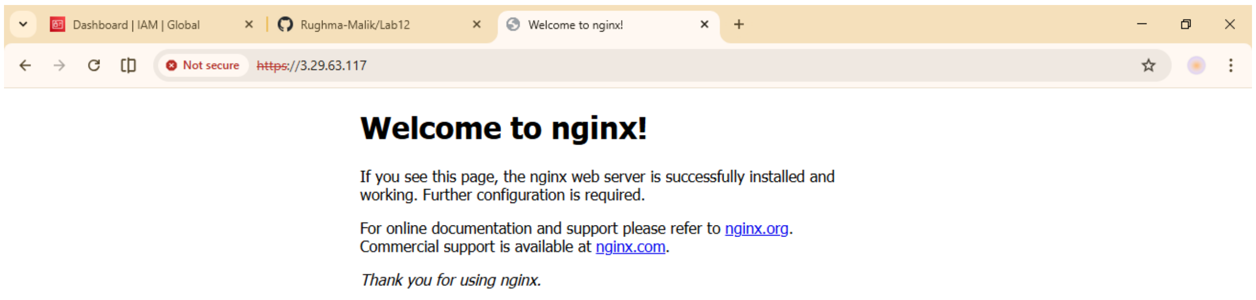

Welcome to nginx!

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.

5. Verify HTTP to HTTPS redirect:


Welcome to nginx!

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

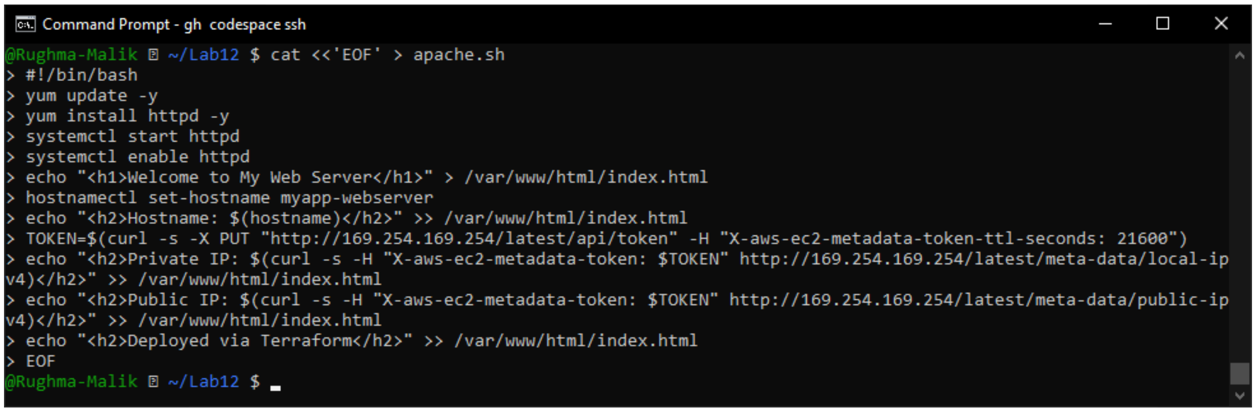
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Task 7 — Configure Nginx as reverse proxy

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

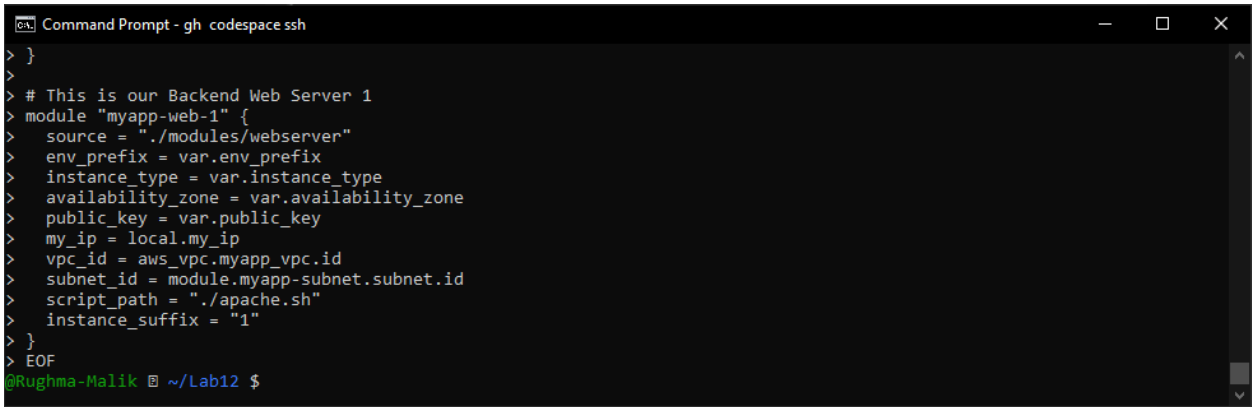
1. Create apache.sh script for backend web server:



Command Prompt - gh codespace ssh

```
@Rughma-Malik ⊡ ~/Lab12 $ cat <<'EOF' > apache.sh
> #!/bin/bash
> yum update -y
> yum install httpd -y
> systemctl start httpd
> systemctl enable httpd
> echo "<h1>Welcome to My Web Server</h1>" > /var/www/html/index.html
> hostnamectl set-hostname myapp-webserver
> echo "<h2>Hostname: $(hostname)</h2>" >> /var/www/html/index.html
> TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
> echo "<h2>Private IP: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/local-ip
v4)</h2>" >> /var/www/html/index.html
> echo "<h2>Public IP: $(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" http://169.254.169.254/latest/meta-data/public-ip
v4)</h2>" >> /var/www/html/index.html
> echo "<h2>Deployed via Terraform</h2>" >> /var/www/html/index.html
> EOF
@Rughma-Malik ⊡ ~/Lab12 $
```

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



Command Prompt - gh codespace ssh

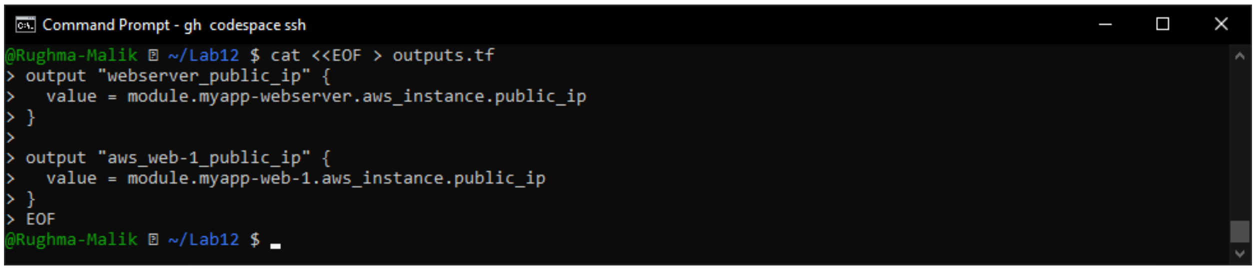
```
> }
>
> # This is our Backend Web Server 1
> module "myapp-web-1" {
>   source = "./modules/webserver"
>   env_prefix = var.env_prefix
>   instance_type = var.instance_type
>   availability_zone = var.availability_zone
>   public_key = var.public_key
>   my_ip = local.my_ip
>   vpc_id = aws_vpc.myapp_vpc.id
>   subnet_id = module.myapp-subnet.subnet.id
>   script_path = "./apache.sh"
>   instance_suffix = "1"
> }
> EOF
@Rughma-Malik ⊡ ~/Lab12 $
```

3. Update outputs.tf:

output "aws_web-1_public_ip" {

value = module.myapp-web-1.aws_instance.public_ip

}

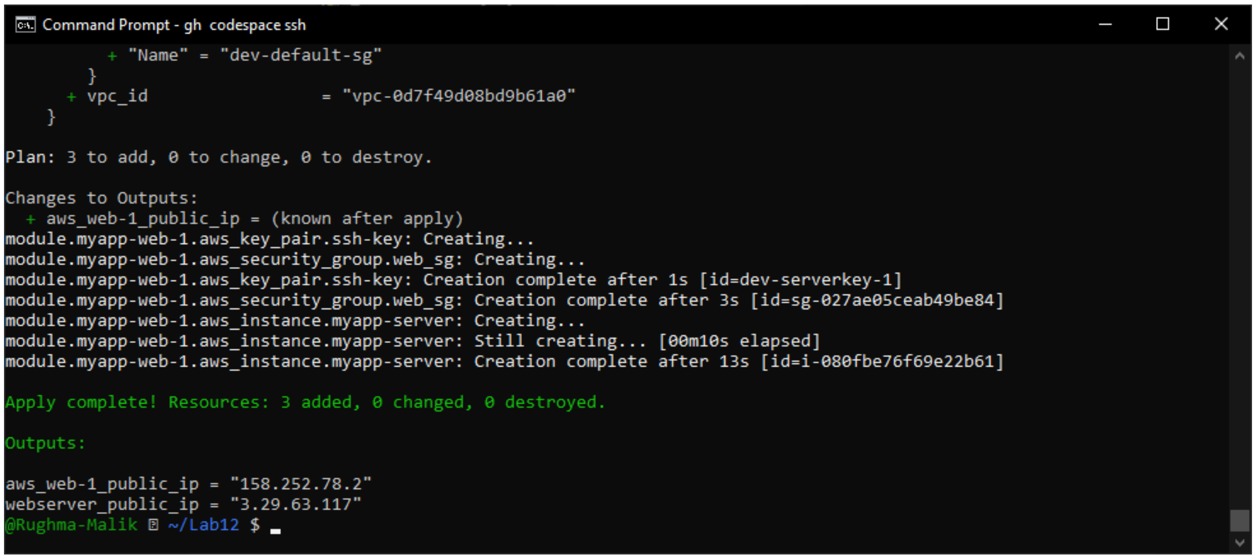


Command Prompt - gh codespace ssh

```
@Rughma-Malik ⊡ ~/Lab12 $ cat <<EOF > outputs.tf
> output "webserver_public_ip" {
>   value = module.myapp-webserver.aws_instance.public_ip
> }
>
> output "aws_web-1_public_ip" {
>   value = module.myapp-web-1.aws_instance.public_ip
> }
> EOF
@Rughma-Malik ⊡ ~/Lab12 $
```

4. Apply the configuration:

terraform apply -auto-approve

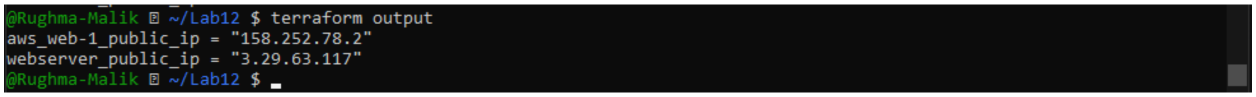


Command Prompt - gh codespace ssh

```
          + "Name" = "dev-default-sg"
        }
      + vpc_id                 = "vpc-0d7f49d08bd9b61a0"
    }

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

Changes to Outputs:
  + aws_web-1_public_ip = (known after apply)
module.myapp-web-1.aws_key_pair.ssh-key: Creating...
module.myapp-web-1.aws_security_group.web_sg: Creating...
module.myapp-web-1.aws_key_pair.ssh-key: Creation complete after 1s [id=dev-serverkey-1]
module.myapp-web-1.aws_security_group.web_sg: Creation complete after 3s [id=sg-027ae05ceab49be84]
module.myapp-web-1.aws_instance.myapp-server: Creating...
module.myapp-web-1.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Creation complete after 13s [id=i-080fbe76f69e22b61]

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

Outputs:

aws_web-1_public_ip = "158.252.78.2"
webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $
```

5. Get the outputs:

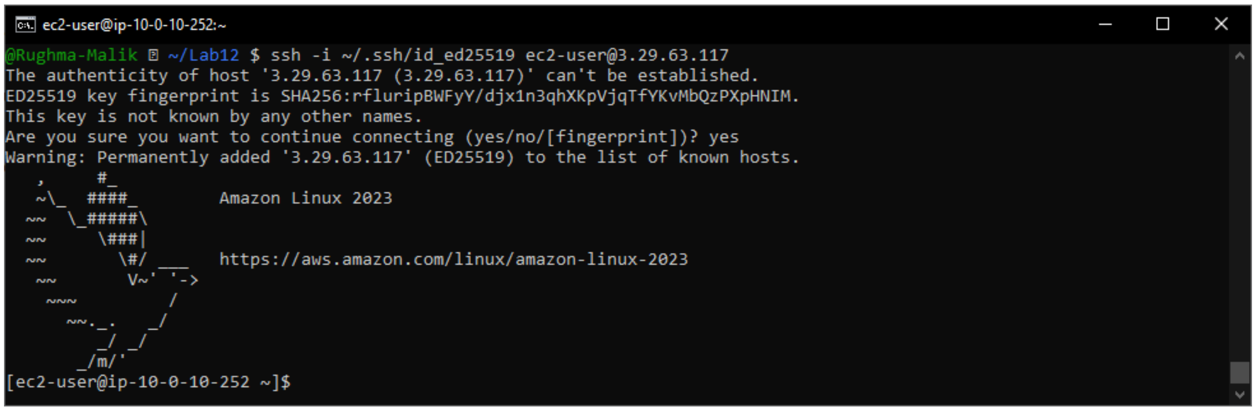
terraform output



```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
aws_web-1_public_ip = "158.252.78.2"
webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $
```

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

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

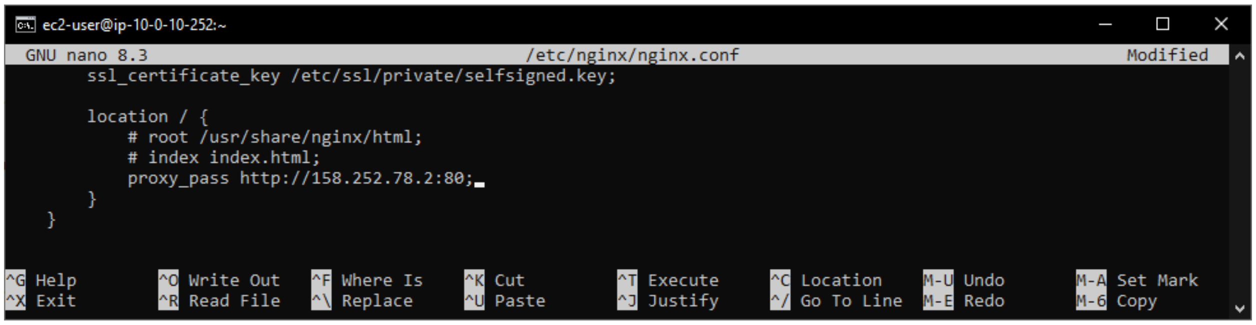


ec2-user@ip-10-0-10-252:~

```
@Rughma-Malik ➜ ~/Lab12 $ ssh -i ~/.ssh/id_ed25519 ec2-user@3.29.63.117
The authenticity of host '3.29.63.117 (3.29.63.117)' can't be established.
ED25519 key fingerprint is SHA256:rfluripBWFyY/djx1n3qhXKpVjqTfYKvMbQzPXpHNIM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.29.63.117' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-252 ~]$
```

7. Edit the Nginx configuration:

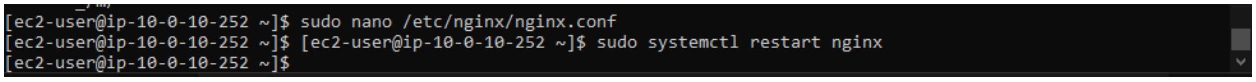
sudo vim /etc/nginx/nginx.conf



```
ec2-user@ip-10-0-10-252:~
 GNU nano 8.3                              /etc/nginx/nginx.conf                              Modified
        ssl_certificate_key /etc/ssl/private/selfsigned.key;

        location / {
            # root /usr/share/nginx/html;
            # index index.html;
            proxy_pass http://158.252.78.2:80;
        }
    }

^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

8. Restart Nginx:

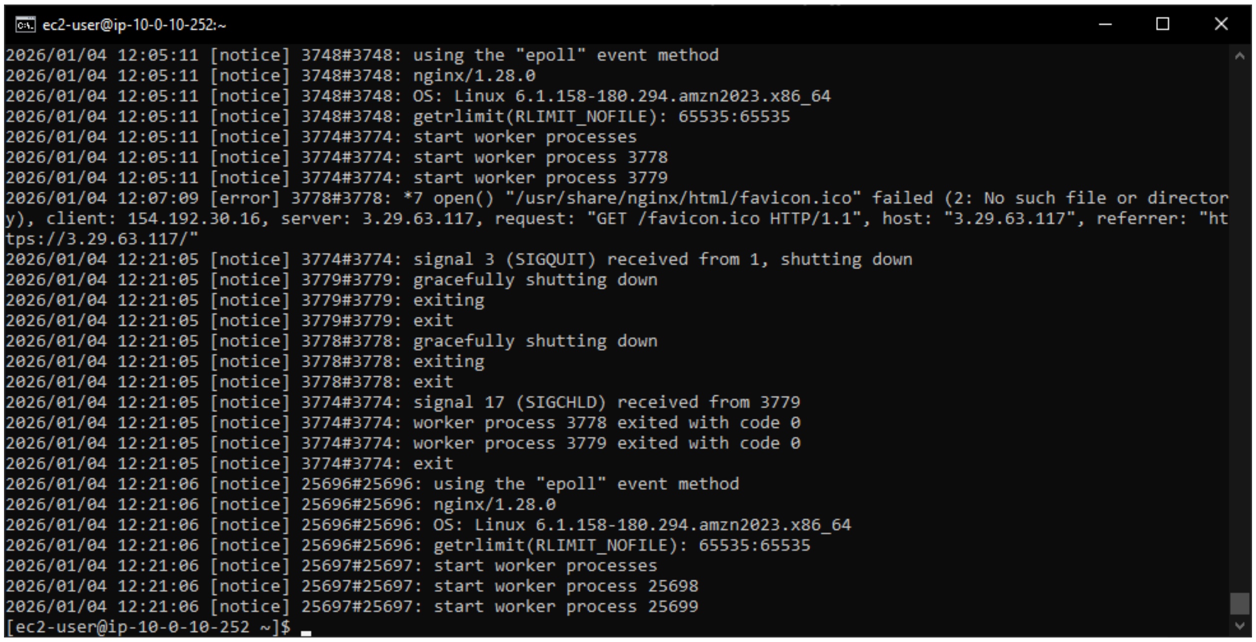
sudo systemctl restart nginx



```
[ec2-user@ip-10-0-10-252 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-252 ~]$ [ec2-user@ip-10-0-10-252 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-252 ~]$
```

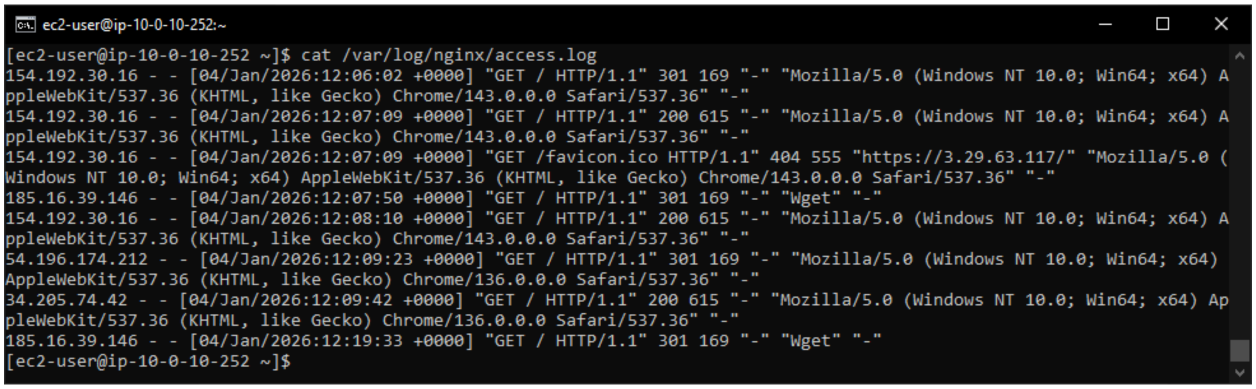
9. View Nginx logs and configuration files:

cat /var/log/nginx/error.log



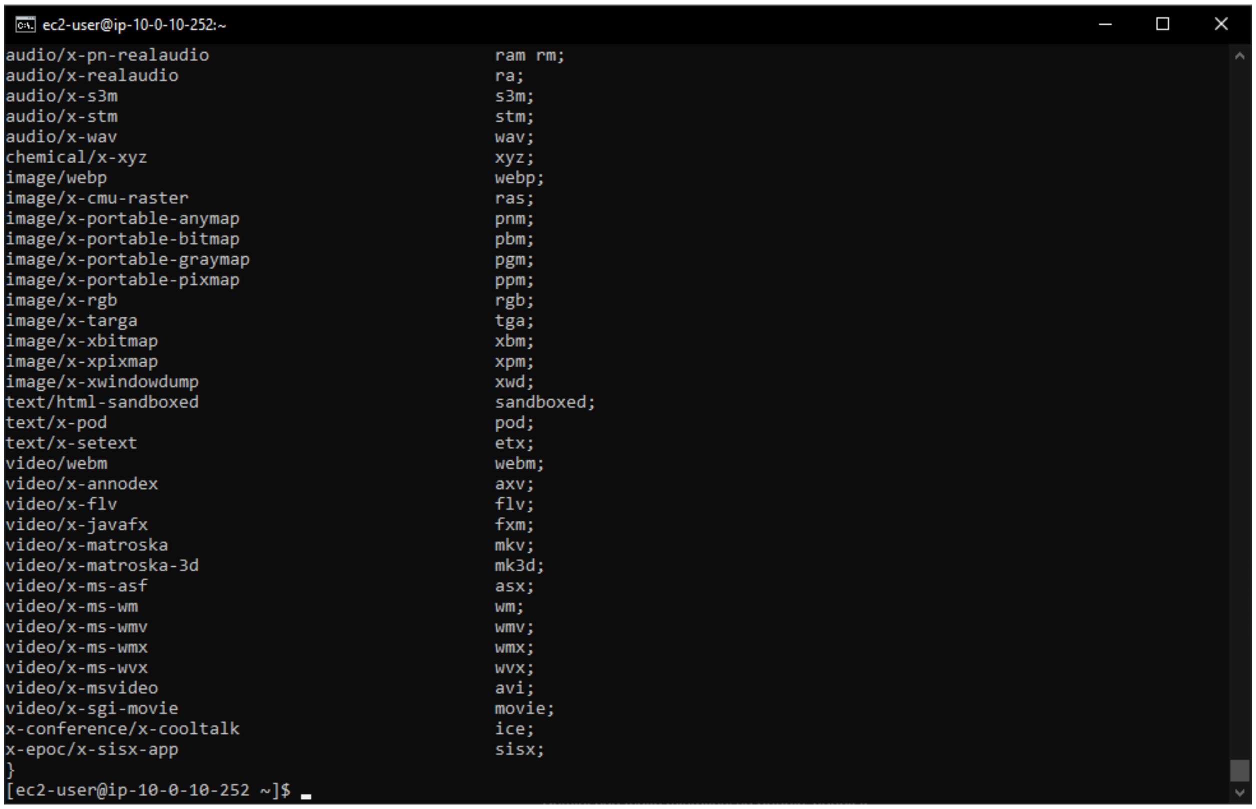
```
ec2-user@ip-10-0-10-252:~
2026/01/04 12:05:11 [notice] 3748#3748: using the "epoll" event method
2026/01/04 12:05:11 [notice] 3748#3748: nginx/1.28.0
2026/01/04 12:05:11 [notice] 3748#3748: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2026/01/04 12:05:11 [notice] 3748#3748: getrlimit(RLIMIT_NOFILE): 65535:65535
2026/01/04 12:05:11 [notice] 3774#3774: start worker processes
2026/01/04 12:05:11 [notice] 3774#3774: start worker process 3778
2026/01/04 12:05:11 [notice] 3774#3774: start worker process 3779
2026/01/04 12:07:09 [error] 3778#3778: *7 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or director
y), client: 154.192.30.16, server: 3.29.63.117, request: "GET /favicon.ico HTTP/1.1", host: "3.29.63.117", referrer: "ht
tps://3.29.63.117/"
2026/01/04 12:21:05 [notice] 3774#3774: signal 3 (SIGQUIT) received from 1, shutting down
2026/01/04 12:21:05 [notice] 3779#3779: gracefully shutting down
2026/01/04 12:21:05 [notice] 3779#3779: exiting
2026/01/04 12:21:05 [notice] 3779#3779: exit
2026/01/04 12:21:05 [notice] 3778#3778: gracefully shutting down
2026/01/04 12:21:05 [notice] 3778#3778: exiting
2026/01/04 12:21:05 [notice] 3778#3778: exit
2026/01/04 12:21:05 [notice] 3774#3774: signal 17 (SIGCHLD) received from 3779
2026/01/04 12:21:05 [notice] 3774#3774: worker process 3778 exited with code 0
2026/01/04 12:21:05 [notice] 3774#3774: worker process 3779 exited with code 0
2026/01/04 12:21:05 [notice] 3774#3774: exit
2026/01/04 12:21:06 [notice] 25696#25696: using the "epoll" event method
2026/01/04 12:21:06 [notice] 25696#25696: nginx/1.28.0
2026/01/04 12:21:06 [notice] 25696#25696: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2026/01/04 12:21:06 [notice] 25696#25696: getrlimit(RLIMIT_NOFILE): 65535:65535
2026/01/04 12:21:06 [notice] 25697#25697: start worker processes
2026/01/04 12:21:06 [notice] 25697#25697: start worker process 25698
2026/01/04 12:21:06 [notice] 25697#25697: start worker process 25699
[ec2-user@ip-10-0-10-252 ~]$
```

cat /var/log/nginx/access.log



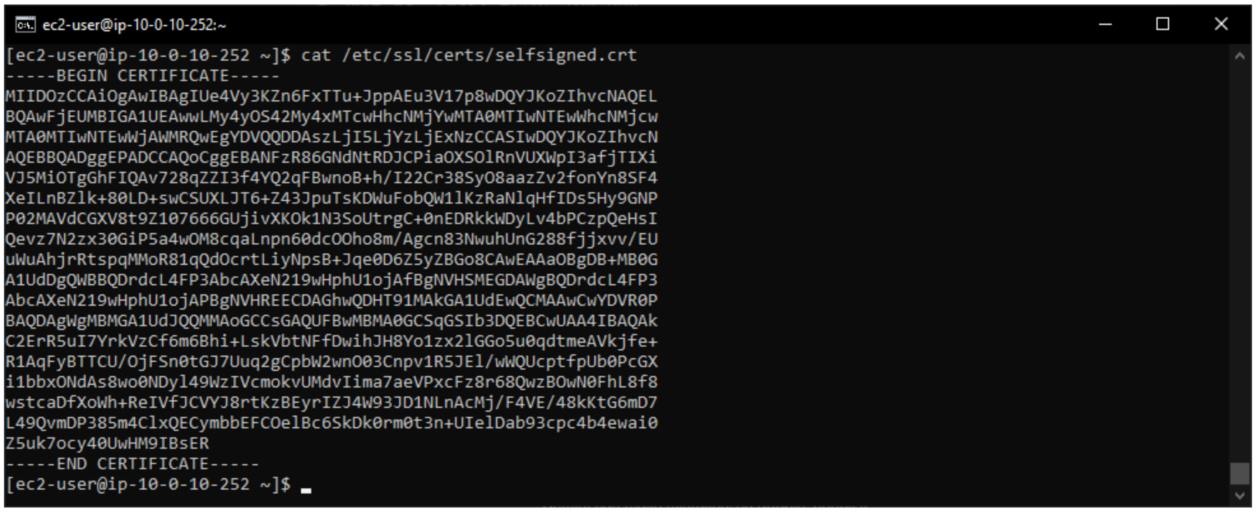
```
ec2-user@ip-10-0-10-252:~
[ec2-user@ip-10-0-10-252 ~]$ cat /var/log/nginx/access.log
154.192.30.16 - - [04/Jan/2026:12:06:02 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) A
ppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
154.192.30.16 - - [04/Jan/2026:12:07:09 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) A
ppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
154.192.30.16 - - [04/Jan/2026:12:07:09 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "https://3.29.63.117/" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
185.16.39.146 - - [04/Jan/2026:12:07:50 +0000] "GET / HTTP/1.1" 301 169 "-" "Wget" "-"
154.192.30.16 - - [04/Jan/2026:12:08:10 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) A
ppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
54.196.174.212 - - [04/Jan/2026:12:09:23 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36" "-"
34.205.74.42 - - [04/Jan/2026:12:09:42 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) Ap
pleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36" "-"
185.16.39.146 - - [04/Jan/2026:12:19:33 +0000] "GET / HTTP/1.1" 301 169 "-" "Wget" "-"
[ec2-user@ip-10-0-10-252 ~]$
```

cat /etc/nginx/mime.types



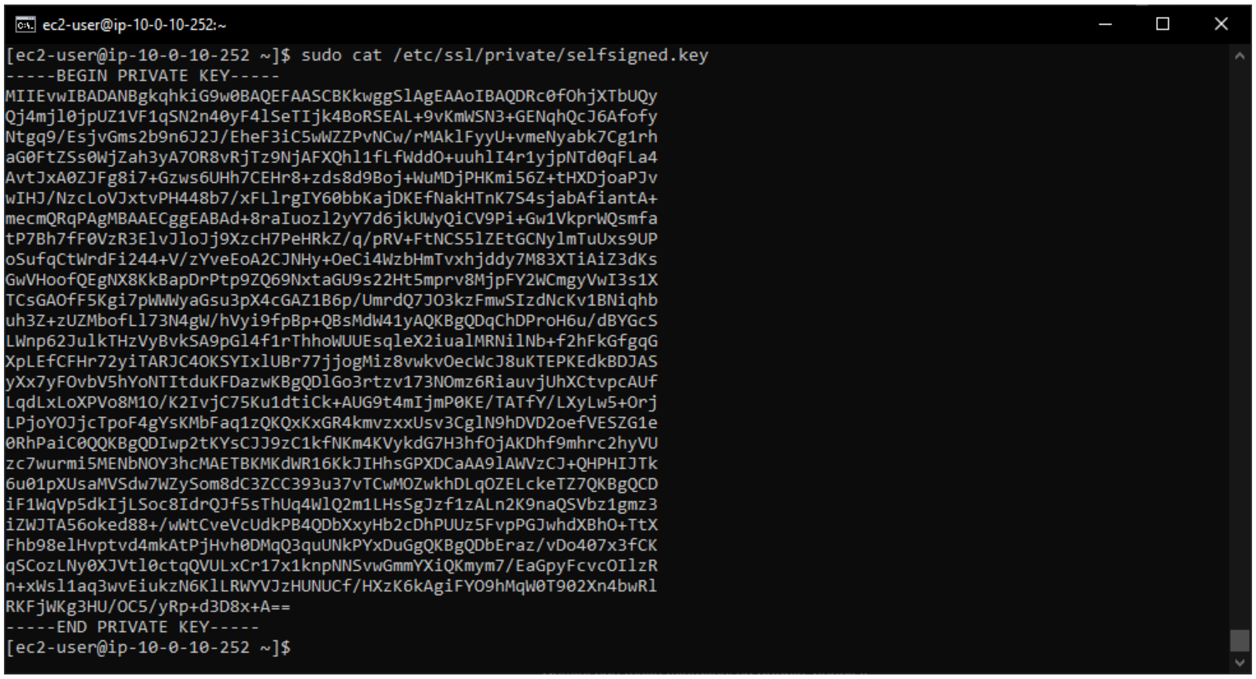
```
ec2-user@ip-10-0-10-252:~
audio/x-pn-realaudio                             ram rm;
audio/x-realaudio                                ra;
audio/x-s3m                                      s3m;
audio/x-stm                                      stm;
audio/x-wav                                      wav;
chemical/x-xyz                                   xyz;
image/webp                                       webp;
image/x-cmu-raster                               ras;
image/x-portable-anymap                          pnm;
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-10-0-10-252 ~]$
```

cat /etc/ssl/certs/selfsigned.crt



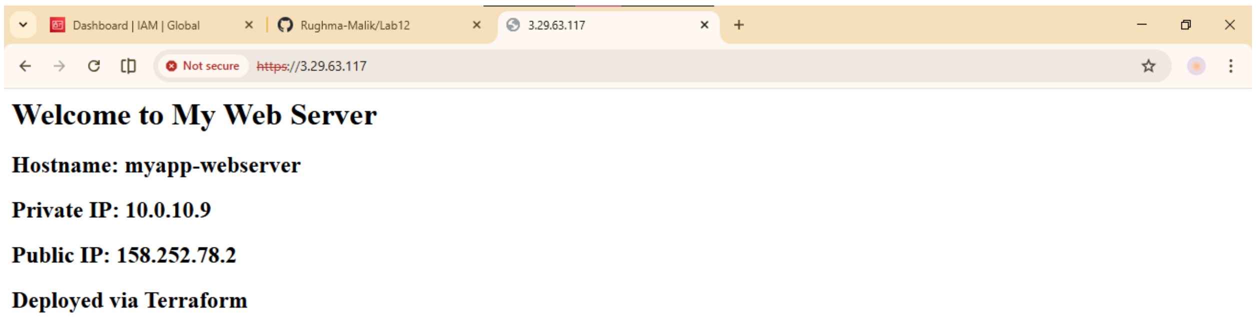
```
[ec2-user@ip-10-0-10-252 ~]$ cat /etc/ssl/certs/selfsigned.crt
-----BEGIN CERTIFICATE-----
MIIDOzCCAiOgAwIBAgIUe4Vy3KZn6FxTTu+JppAEu3V17p8wDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAwwLMy4yOS42My4xMTcwHhcNMjYwMTA0MTIwNTEwWhcNMjcw
MTA0MTIwNTEwWjAWMRQwEgYDVQQDDAszLjI5LjYzLjExNzCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANFzR86GNdNtRDJCPiaOXSOlRnVUXWpI3afjTIXi
VJ5MiOTgGhFIQAv728qZZI3f4YQ2qFBwnoB+h/I22Cr38SyO8aazZv2fonYn8SF4
XeILnBZlk+80LD+swCSUXLJT6+Z43JpuTsKDWuFobQW1lKzRaNlqHfIDs5Hy9GNP
P02MAVdCGXV8t9Z107666GUjivXKOk1N3SoUtrgC+0nEDRkkWDyLv4bPCzpQeHsI
Qevz7N2zx30GiP5a4wOM8cqaLnpn60dcOOho8m/Agcn83NwuhUnG288fjjxvv/EU
uWuAhjrRtspqMMoR81qQdOcrtLiyNpsB+Jqe0D6Z5yZBGo8CAwEAAaOBgDB+MB0G
A1UdDgQWBBQDrdcL4FP3AbcAXeN219wHphU1ojAfBgNVHSMEGDAWgBQDrdcL4FP3
AbcAXeN219wHphU1ojAPBgNVHREECDAGhwQDHT91MAkGA1UdEwQCMAAwCwYDVR0P
BAQDAgWgMBMGA1UdJQQMMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQAk
C2ErR5uI7YrkVzCf6m6Bhi+LskVbtNFfDwihJH8Yo1zx2lGGo5u0qdtmeAVkjfe+
R1AqFyBTTCU/OjFSn0tGJ7Uuq2gCpbW2wnO03Cnpv1R5JEl/wWQUcptfpUb0PcGX
i1bbxONdAs8wo0NDyl49WzIVcmokvUMdvIima7aeVPxcFz8r68QwzBOwN0FhL8f8
wstcaDfXoWh+ReIVfJCVYJ8rtKzBEyrIZJ4W93JD1NLnAcMj/F4VE/48kKtG6mD7
L49QvmDP385m4ClxQECymbbEFCOelBc6SkDk0rm0t3n+UIelDab93cpc4b4ewai0
Z5uk7ocy40UwHM9IBsER
-----END CERTIFICATE-----
[ec2-user@ip-10-0-10-252 ~]$
```

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



```
[ec2-user@ip-10-0-10-252 ~]$ sudo cat /etc/ssl/private/selfsigned.key
-----BEGIN PRIVATE KEY-----
MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQDRc0fOhjXTbUQy
Qj4mjl0jpUZ1VF1qSN2n40yF4lSeTIjk4BoRSEAL+9vKmWSN3+GENqhQcJ6Afofy
Ntgq9/EsjvGms2b9n6J2J/EheF3iC5wWZZPvNCw/rMAklFyyU+vmeNyabk7Cg1rh
aG0FtZSs0WjZah3yA7OR8vRjTz9NjAFXQhl1fLfWddO+uuhlI4r1yjpNTd0qFLa4
AvtJxA0ZJFg8i7+Gzws6UHh7CEHr8+zds8d9Boj+WuMDjPHKmi56Z+tHXDjoaPJv
wIHJ/NzcLoVJxtvPH448b7/xFLlrgIY60bbKajDKEfNakHTnK7S4sjabAfiantA+
mecmQRqPAgMBAAECggEABAd+8raIuozl2yY7d6jkUWyQiCV9Pi+Gw1VkprWQsmfa
tP7Bh7fF0VzR3ElvJloJj9XzcH7PeHRkZ/q/pRV+FtNCS5lZEtGCNylmTuUxs9UP
oSufqCtWrdFi244+V/zYveEoA2CJNHy+OeCi4WzbHmTvxhjddy7M83XTiAiZ3dKs
GwVHoofQEgNX8KkBapDrPtp9ZQ69NxtaGU9s22Ht5mprv8MjpFY2WCmgyVwI3s1X
TCsGAOfF5Kgi7pWWWyaGsu3pX4cGAZ1B6p/UmrdQ7JO3kzFmwSIzdNcKv1BNiqhb
uh3Z+zUZMbofLl73N4gW/hVyi9fpBp+QBsMdW41yAQKBgQDqChDProH6u/dBYGcS
LWnp62JulkTHzVyBvkSA9pGl4f1rThhoWUUEsqleX2iualMRNilNb+f2hFkGfgqG
XpLEfCFHr72yiTARJC4OKSYIxlUBr77jjogMiz8vwkvOecWcJ8uKTEPKEdkBDJAS
yXx7yFOvbV5hYoNTItduKFDazwKBgQDlGo3rtzv173NOmz6RiauvjUhXCtvpcAUf
LqdLxLoXPVo8M1O/K2IvjC75Ku1dtiCk+AUG9t4mIjmP0KE/TATfY/LXyLw5+Orj
LPjoYOJjcTpoF4gYsKMbFaq1zQKQxKxGR4kmvzxxUsv3CglN9hDVD2oefVESZG1e
0RhPaiC0QQKBgQDIwp2tKYsCJJ9zC1kfNKm4KVykdG7H3hfOjAKDhf9mhrc2hyVU
zc7wurmi5MENbNOY3hcMAETBKMKdWR16KkJIHhsGPXDCaAA9lAWVzCJ+QHPHIJTk
6u01pXUsaMVSdw7WZySom8dC3ZCC393u37vTCwMOZwkhDLqOZELckeTZ7QKBgQCD
iF1WqVp5dkIjLSoc8IdrQJf5sThUq4WlQ2m1LHsSgJzf1zALn2K9naQSVbz1gmz3
iZWJTA56oked88+/wWtCveVcUdkPB4QDbXxyHb2cDhPUUz5FvpPGJwhdXBhO+TtX
Fhb98elHvptvd4mkAtPjHvh0DMqQ3quUNkPYxDuGgQKBgQDbEraz/vDo407x3fCK
qSCozLNy0XJVtl0ctqQVULxCr17x1knpNNSvwGmmYXiQKmym7/EaGpyFcvcOIlzR
n+xWsl1aq3wvEiukzN6KlLRWYVJzHUNUCf/HXzK6kAgiFYO9hMqW0T902Xn4bwRl
RKFjWKg3HU/OC5/yRp+d3D8x+A==
-----END PRIVATE KEY-----
[ec2-user@ip-10-0-10-252 ~]$
```

10. Test reverse proxy in browser:



Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.9

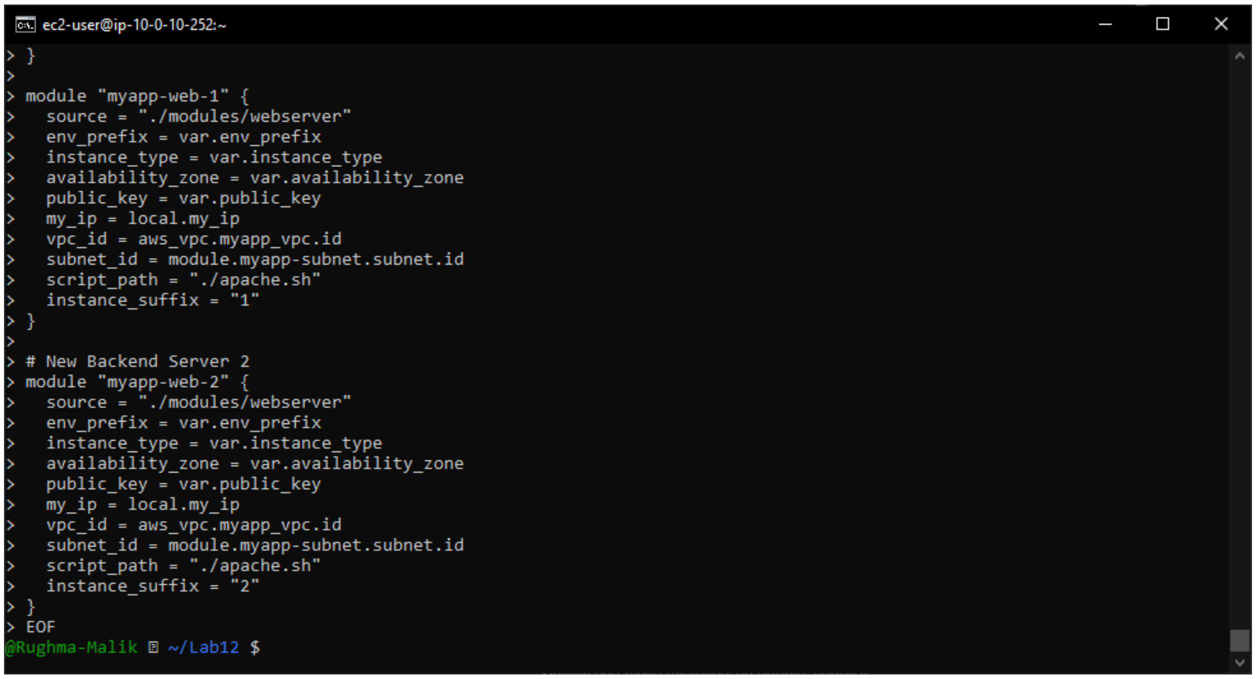
Public IP: 158.252.78.2

Deployed via Terraform

Task 8 — Configure Nginx as load balancer

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

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



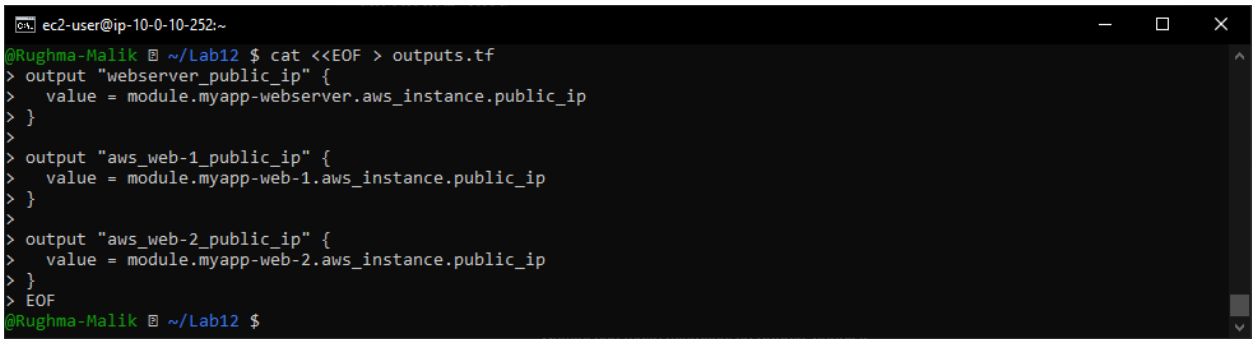
```
ec2-user@ip-10-0-10-252:~
> }
>
> module "myapp-web-1" {
>   source = "./modules/webserver"
>   env_prefix = var.env_prefix
>   instance_type = var.instance_type
>   availability_zone = var.availability_zone
>   public_key = var.public_key
>   my_ip = local.my_ip
>   vpc_id = aws_vpc.myapp_vpc.id
>   subnet_id = module.myapp-subnet.subnet.id
>   script_path = "./apache.sh"
>   instance_suffix = "1"
> }
>
> # New Backend Server 2
> 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"
> }
> EOF
@Rughma-Malik ~/Lab12 $
```

2. Update outputs.tf:

output "aws_web-2_public_ip" {

 value = module. myapp-web-2.aws_instance.public_ip

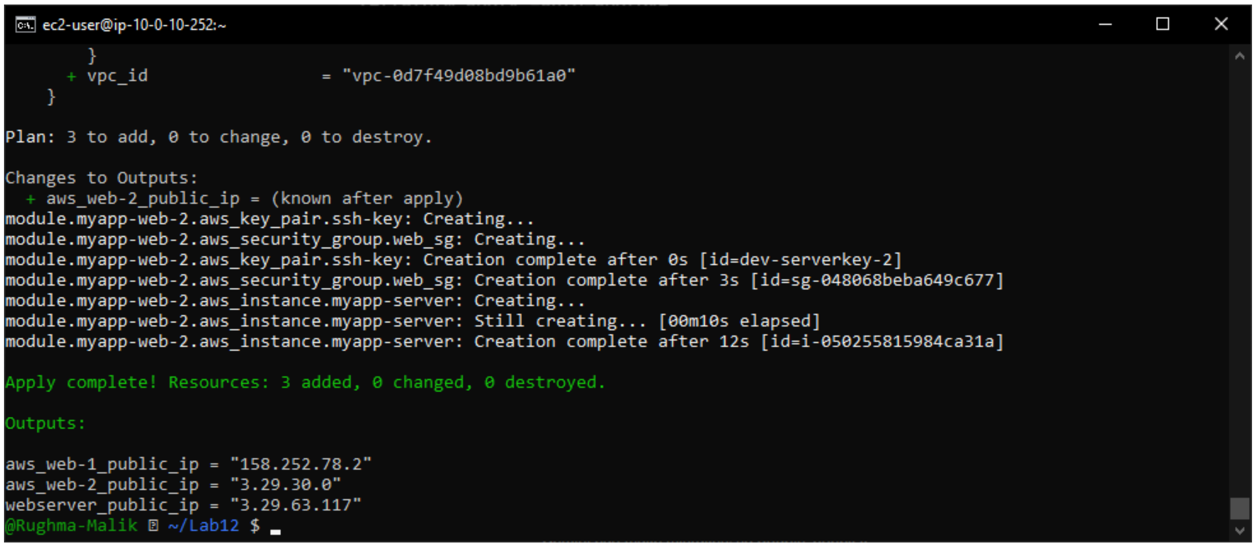
}



```
ec2-user@ip-10-0-10-252:~
@Rughma-Malik ~/Lab12 $ cat <<EOF > outputs.tf
> output "webserver_public_ip" {
>   value = module.myapp-webserver.aws_instance.public_ip
> }
>
> output "aws_web-1_public_ip" {
>   value = module.myapp-web-1.aws_instance.public_ip
> }
>
> output "aws_web-2_public_ip" {
>   value = module.myapp-web-2.aws_instance.public_ip
> }
> EOF
@Rughma-Malik ~/Lab12 $
```

3. Apply the configuration:

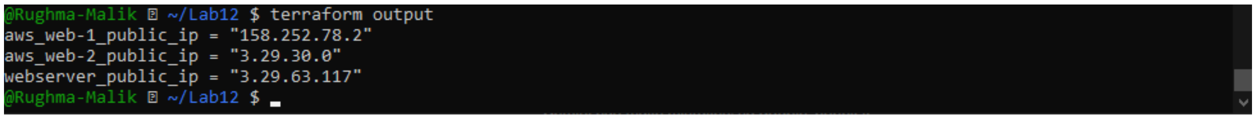
terraform apply -auto-approve



```
      }
    + vpc_id                 = "vpc-0d7f49d08bd9b61a0"
  }

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

Changes to Outputs:
  + aws_web-2_public_ip = (known after apply)
module.myapp-web-2.aws_key_pair.ssh-key: Creating...
module.myapp-web-2.aws_security_group.web_sg: Creating...
module.myapp-web-2.aws_key_pair.ssh-key: Creation complete after 0s [id=dev-serverkey-2]
module.myapp-web-2.aws_security_group.web_sg: Creation complete after 3s [id=sg-048068beba649c677]
module.myapp-web-2.aws_instance.myapp-server: Creating...
module.myapp-web-2.aws_instance.myapp-server: Still creating... [00m10s elapsed]
module.myapp-web-2.aws_instance.myapp-server: Creation complete after 12s [id=i-050255815984ca31a]

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

Outputs:

aws_web-1_public_ip = "158.252.78.2"
aws_web-2_public_ip = "3.29.30.0"
webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $
```

4. Get all outputs:

terraform output

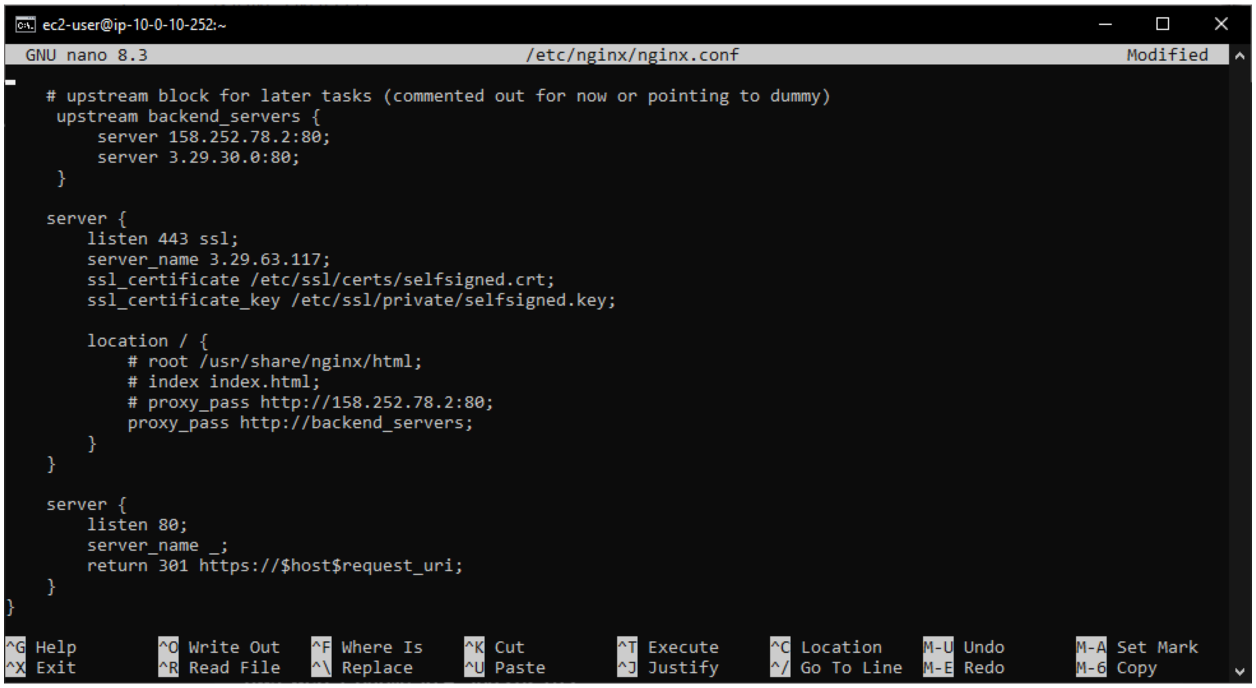


```
@Rughma-Malik ➜ ~/Lab12 $ terraform output
aws_web-1_public_ip = "158.252.78.2"
aws_web-2_public_ip = "3.29.30.0"
webserver_public_ip = "3.29.63.117"
@Rughma-Malik ➜ ~/Lab12 $
```

5. SSH into the webserver (Nginx proxy):

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

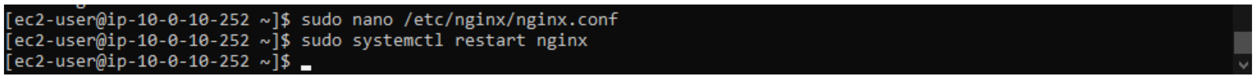
6. Edit Nginx configuration for load balancing:



```
  GNU nano 8.3                    /etc/nginx/nginx.conf                    Modified

    # upstream block for later tasks (commented out for now or pointing to dummy)
     upstream backend_servers {
         server 158.252.78.2:80;
         server 3.29.30.0:80;
     }

    server {
        listen 443 ssl;
        server_name 3.29.63.117;
        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://158.252.78.2:80;
            proxy_pass http://backend_servers;
        }
    }

    server {
        listen 80;
        server_name _;
        return 301 https://$host$request_uri;
    }
}
^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```

7. Restart Nginx:

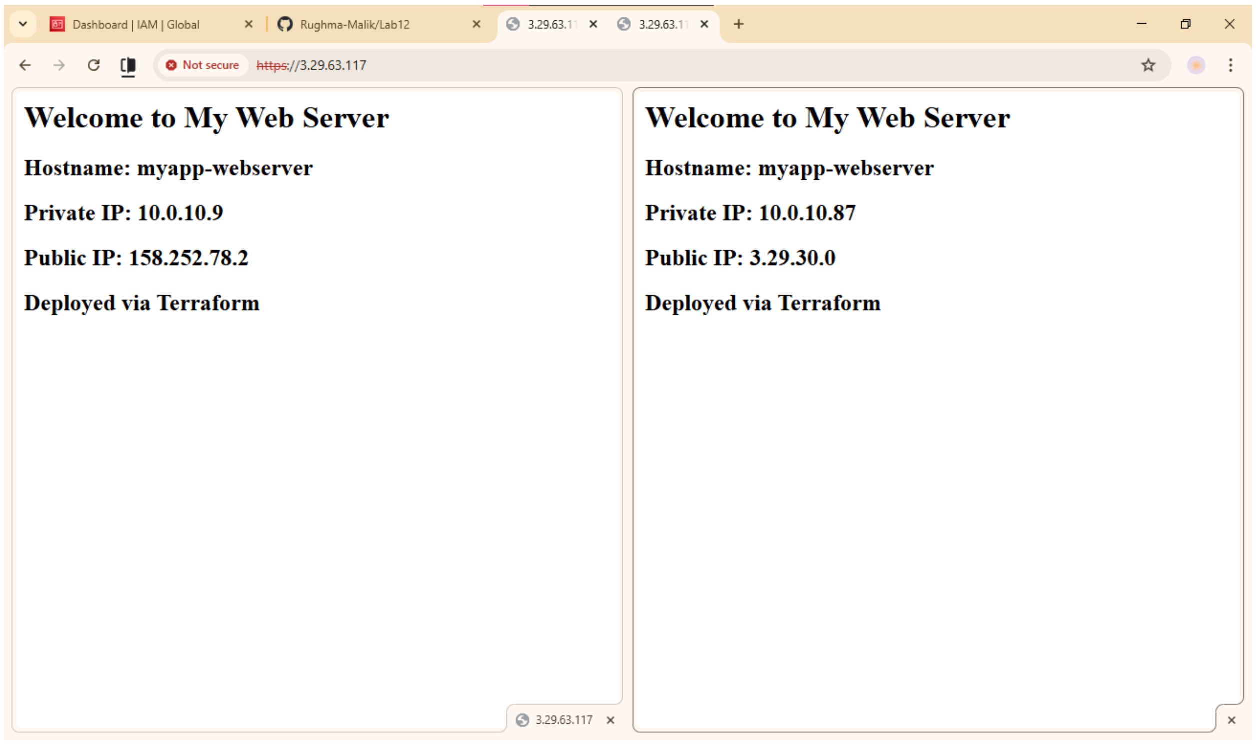
sudo systemctl restart nginx



```
[ec2-user@ip-10-0-10-252 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-252 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-252 ~]$ _
```

8. Test load balancing in browser:

- Open browser and navigate to https://<webserver-public-ip>
- Reload the page multiple times
- You should see the content alternating between web-1 and web-2 (check the hostname/IP in the page)



Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.9

Public IP: 158.252.78.2

Deployed via Terraform

Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.87

Public IP: 3.29.30.0

Deployed via Terraform

Task 9 — Configure high availability with backup servers

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

1. SSH into the webserver:

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

2. Edit Nginx configuration for high availability:

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

Update the upstream block to make web-2 a backup:

```
  upstream backend_servers {
      server <web-1-public-ip>:80;
      server <web-2-public-ip>:80 backup;
  }
```

- **Save screenshot as:** task9_nginx_conf_ha_web1_primary.png — nginx.conf with web-2 as backup.

3. Restart Nginx:

```
sudo systemctl restart nginx
```

4. Test in browser:

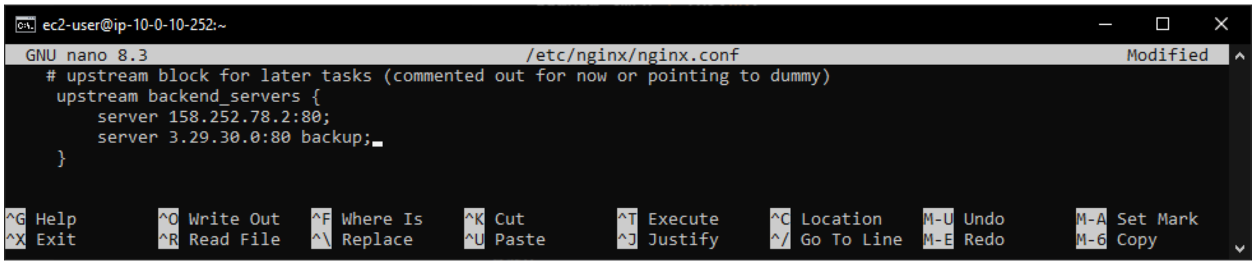
- Open browser and navigate to https://<webserver-public-ip>
- Reload multiple times
- You should ONLY see web-1 (primary server)
- **Save screenshot as:** task9_ha_web1_only.png — browser showing only web-1 content on multiple reloads.

5. Switch backup configuration:

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

Update to make web-1 backup:

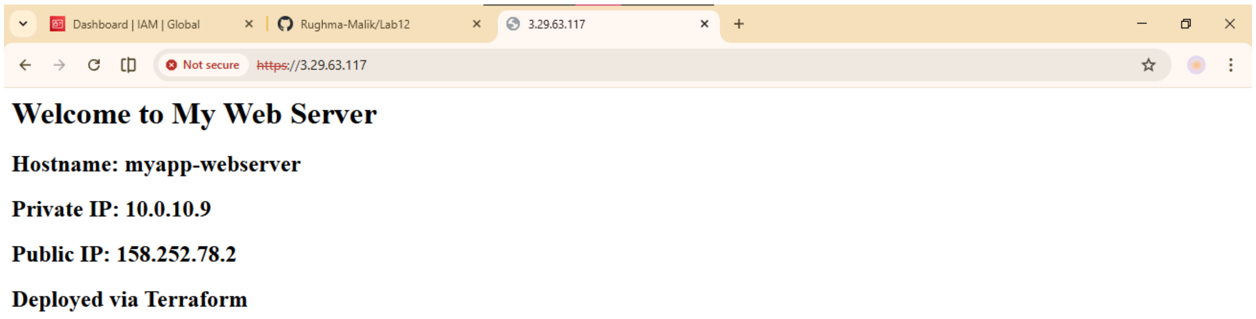
```
  upstream backend_servers {
      server <web-1-public-ip>:80 backup;
      server <web-2-public-ip>:80;
  }
```



```
ec2-user@ip-10-0-10-252:~
  GNU nano 8.3                      /etc/nginx/nginx.conf                          Modified
    # upstream block for later tasks (commented out for now or pointing to dummy)
     upstream backend_servers {
         server 158.252.78.2:80;
         server 3.29.30.0:80 backup;
     }

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

6. Restart Nginx:

sudo systemctl restart nginx



Welcome to My Web Server

Hostname: myapp-webserver

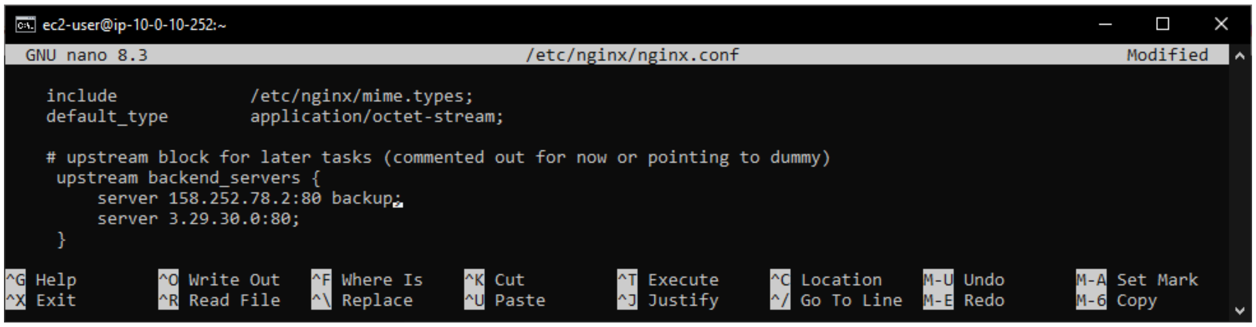
Private IP: 10.0.10.9

Public IP: 158.252.78.2

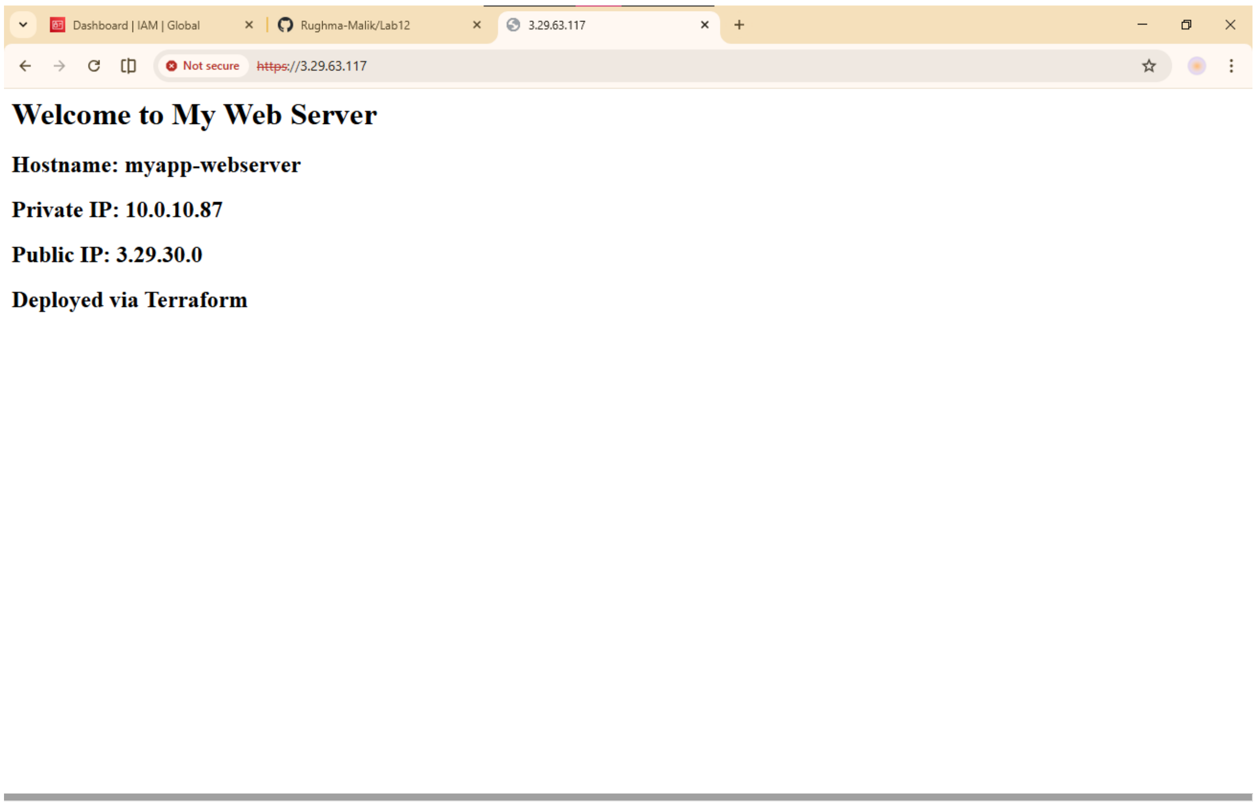
Deployed via Terraform

7. Test in browser:

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



```
ec2-user@ip-10-0-10-252:~
  GNU nano 8.3                      /etc/nginx/nginx.conf                          Modified

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

    # upstream block for later tasks (commented out for now or pointing to dummy)
     upstream backend_servers {
         server 158.252.78.2:80 backup;
         server 3.29.30.0:80;
     }
^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.87

Public IP: 3.29.30.0

Deployed via Terraform

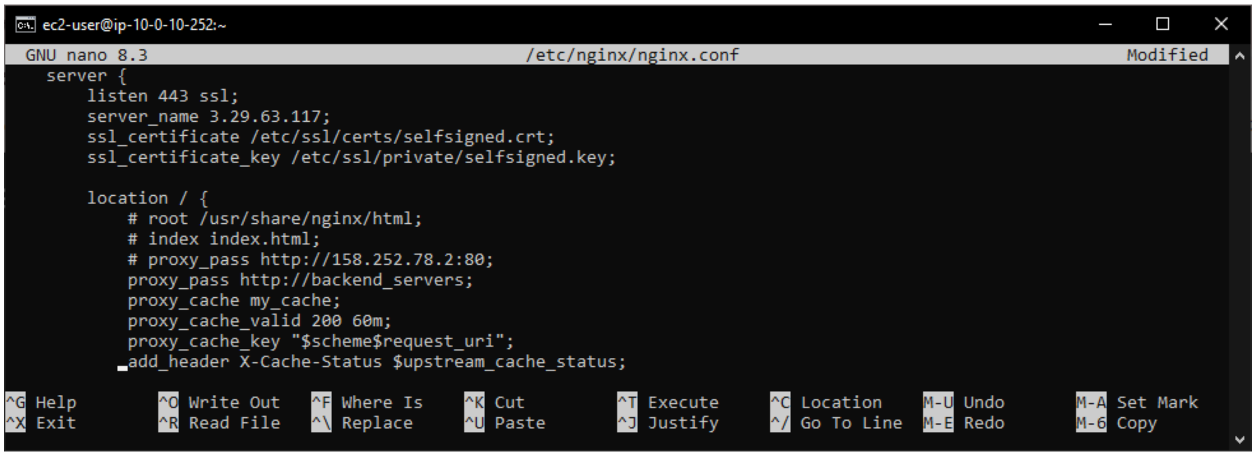
Task 10 — Enable Nginx caching

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

1. SSH into the webserver:

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

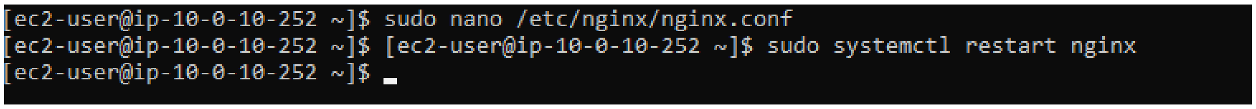
2. Edit Nginx configuration to enable caching:



```
    server {
        listen 443 ssl;
        server_name 3.29.63.117;
        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://158.252.78.2:80;
            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;
```

3. Restart Nginx:

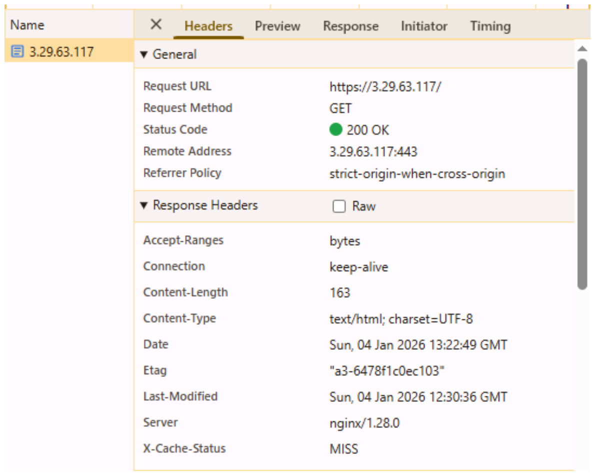
sudo systemctl restart nginx



```
[ec2-user@ip-10-0-10-252 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-252 ~]$ [ec2-user@ip-10-0-10-252 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-252 ~]$ _
```

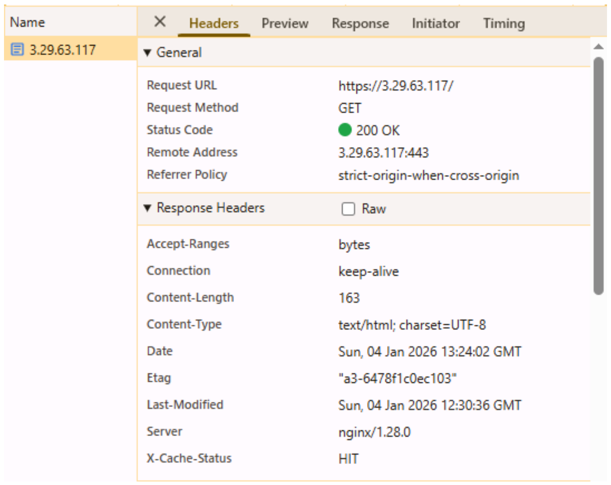
4. Test caching in browser:

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



```
Request URL        https://3.29.63.117/
Request Method     GET
Status Code        200 OK
Remote Address     3.29.63.117:443
Referrer Policy    strict-origin-when-cross-origin

Accept-Ranges      bytes
Connection         keep-alive
Content-Length     163
Content-Type       text/html; charset=UTF-8
Date               Sun, 04 Jan 2026 13:22:49 GMT
Etag               "a3-6478f1c0ec103"
Last-Modified      Sun, 04 Jan 2026 12:30:36 GMT
Server             nginx/1.28.0
X-Cache-Status     MISS
```

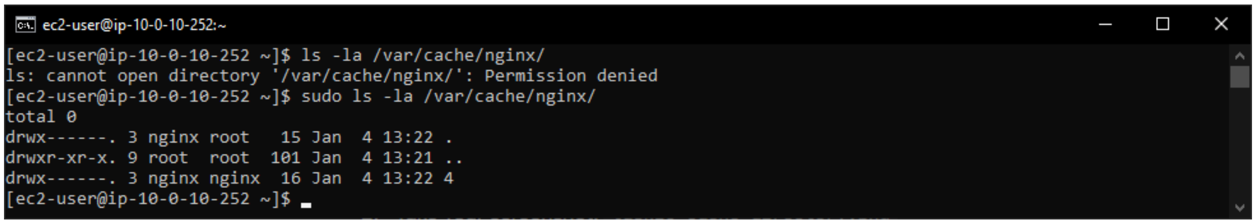
- Reload the page
- Second request should show HIT



```
Request URL        https://3.29.63.117/
Request Method     GET
Status Code        200 OK
Remote Address     3.29.63.117:443
Referrer Policy    strict-origin-when-cross-origin

Accept-Ranges      bytes
Connection         keep-alive
Content-Length     163
Content-Type       text/html; charset=UTF-8
Date               Sun, 04 Jan 2026 13:24:02 GMT
Etag               "a3-6478f1c0ec103"
Last-Modified      Sun, 04 Jan 2026 12:30:36 GMT
Server             nginx/1.28.0
X-Cache-Status     HIT
```

5. Verify cache directory:

ls -la /var/cache/nginx/

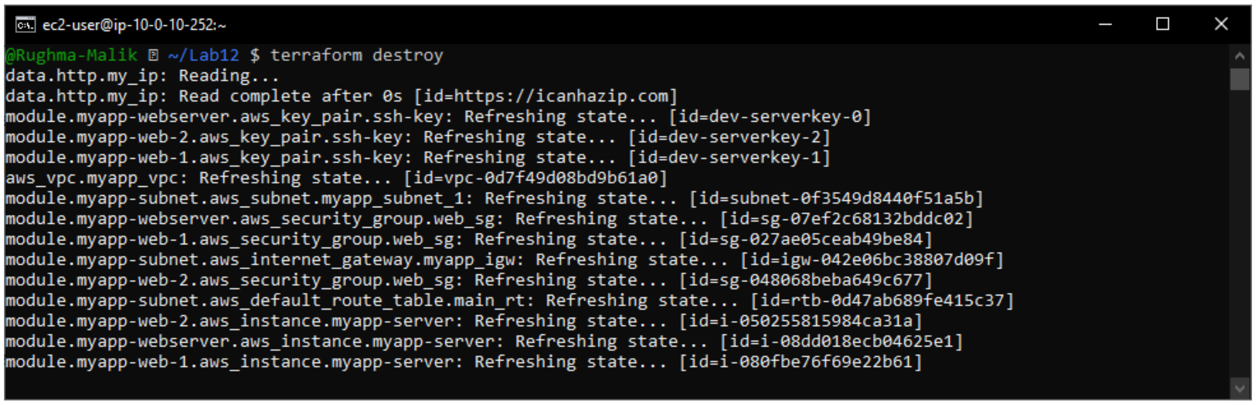


```
[ec2-user@ip-10-0-10-252 ~]$ ls -la /var/cache/nginx/
ls: cannot open directory '/var/cache/nginx/': Permission denied
[ec2-user@ip-10-0-10-252 ~]$ sudo ls -la /var/cache/nginx/
total 0
drwx------. 3 nginx root   15 Jan  4 13:22 .
drwxr-xr-x. 9 root  root  101 Jan  4 13:21 ..
drwx------. 3 nginx nginx  16 Jan  4 13:22 4
[ec2-user@ip-10-0-10-252 ~]$ 
```

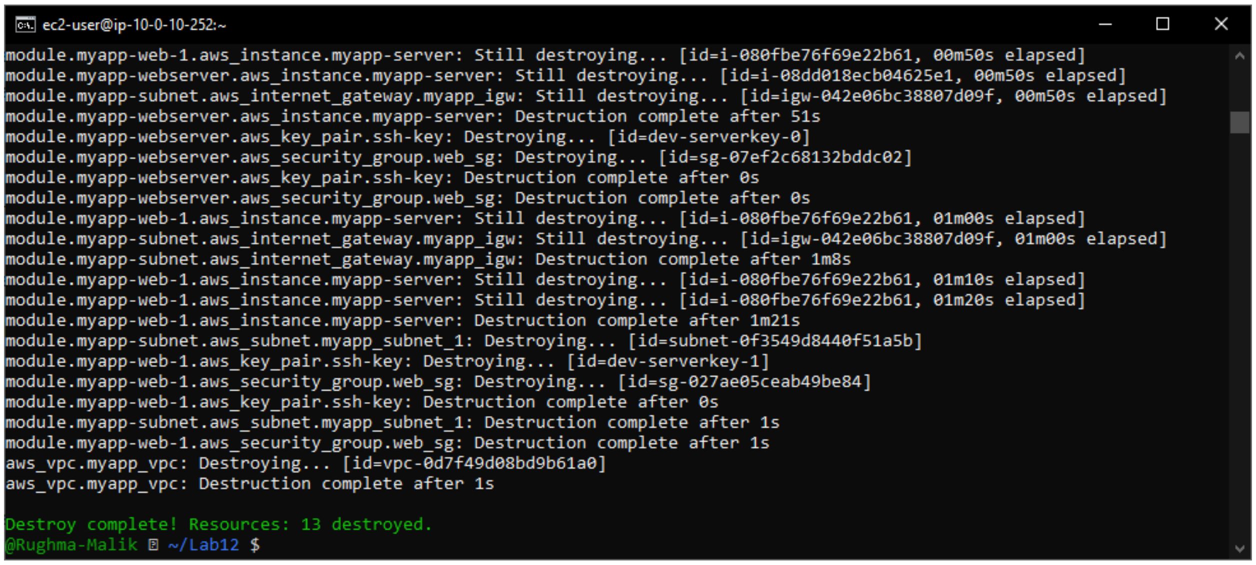
Cleanup

1. Exit SSH sessionDestroy all resources:

terraform destroy



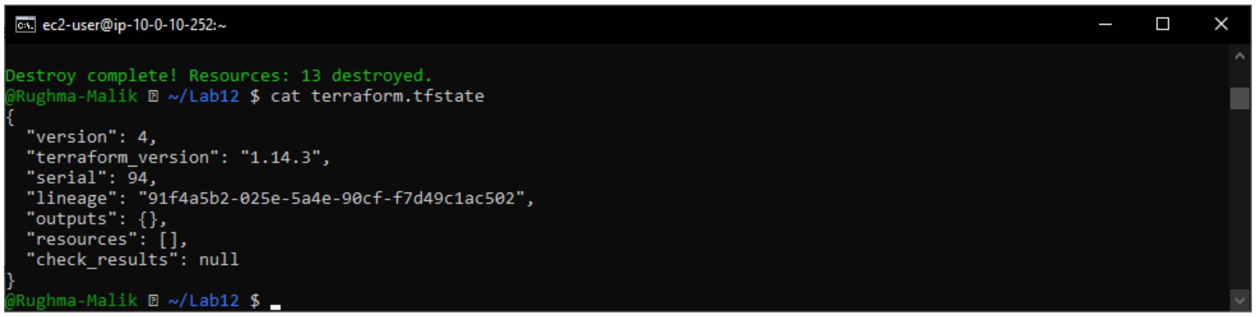
```
@Rughma-Malik ~/Lab12 $ terraform destroy
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
module.myapp-webserver.aws_key_pair.ssh-key: Refreshing state... [id=dev-serverkey-0]
module.myapp-web-2.aws_key_pair.ssh-key: Refreshing state... [id=dev-serverkey-2]
module.myapp-web-1.aws_key_pair.ssh-key: Refreshing state... [id=dev-serverkey-1]
aws_vpc.myapp_vpc: Refreshing state... [id=vpc-0d7f49d08bd9b61a0]
module.myapp-subnet.aws_subnet.myapp_subnet_1: Refreshing state... [id=subnet-0f3549d8440f51a5b]
module.myapp-webserver.aws_security_group.web_sg: Refreshing state... [id=sg-07ef2c68132bddc02]
module.myapp-web-1.aws_security_group.web_sg: Refreshing state... [id=sg-027ae05ceab49be84]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Refreshing state... [id=igw-042e06bc38807d09f]
module.myapp-web-2.aws_security_group.web_sg: Refreshing state... [id=sg-048068beba649c677]
module.myapp-subnet.aws_default_route_table.main_rt: Refreshing state... [id=rtb-0d47ab689fe415c37]
module.myapp-web-2.aws_instance.myapp-server: Refreshing state... [id=i-050255815984ca31a]
module.myapp-webserver.aws_instance.myapp-server: Refreshing state... [id=i-08dd018ecb04625e1]
module.myapp-web-1.aws_instance.myapp-server: Refreshing state... [id=i-080fbe76f69e22b61]
```



```
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-080fbe76f69e22b61, 00m50s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Still destroying... [id=i-08dd018ecb04625e1, 00m50s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-042e06bc38807d09f, 00m50s elapsed]
module.myapp-webserver.aws_instance.myapp-server: Destruction complete after 51s
module.myapp-webserver.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Destroying... [id=sg-07ef2c68132bddc02]
module.myapp-webserver.aws_key_pair.ssh-key: Destruction complete after 0s
module.myapp-webserver.aws_security_group.web_sg: Destruction complete after 0s
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-080fbe76f69e22b61, 01m00s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Still destroying... [id=igw-042e06bc38807d09f, 01m00s elapsed]
module.myapp-subnet.aws_internet_gateway.myapp_igw: Destruction complete after 1m8s
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-080fbe76f69e22b61, 01m10s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Still destroying... [id=i-080fbe76f69e22b61, 01m20s elapsed]
module.myapp-web-1.aws_instance.myapp-server: Destruction complete after 1m21s
module.myapp-subnet.aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0f3549d8440f51a5b]
module.myapp-web-1.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-1]
module.myapp-web-1.aws_security_group.web_sg: Destroying... [id=sg-027ae05ceab49be84]
module.myapp-web-1.aws_key_pair.ssh-key: Destruction complete after 0s
module.myapp-subnet.aws_subnet.myapp_subnet_1: Destruction complete after 1s
module.myapp-web-1.aws_security_group.web_sg: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0d7f49d08bd9b61a0]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 13 destroyed.
@Rughma-Malik ~/Lab12 $
```

3. Verify state files:

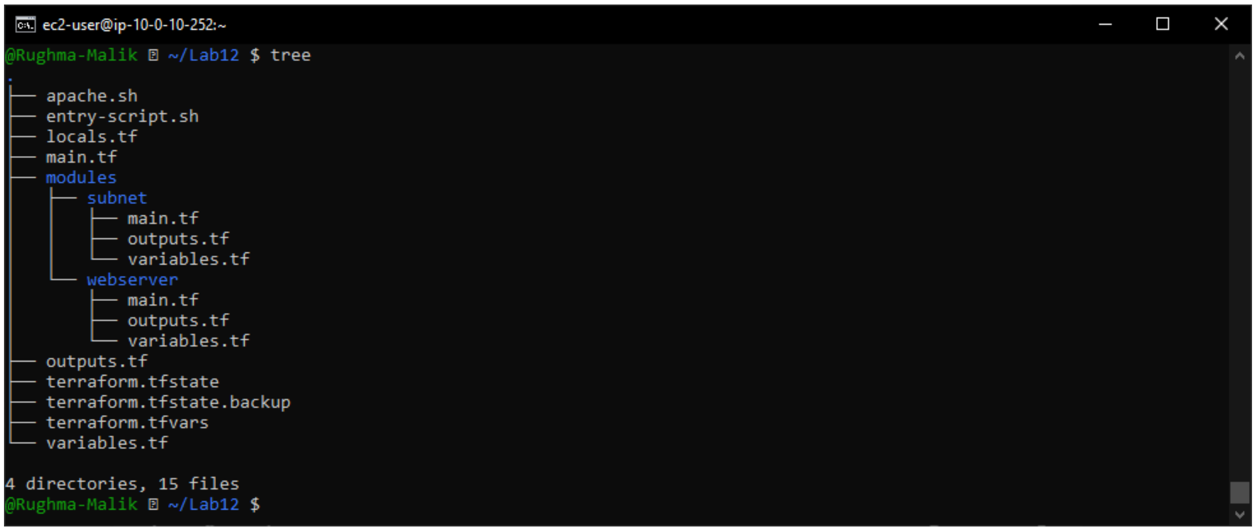
cat terraform.tfstate



```
ec2-user@ip-10-0-10-252:~

Destroy complete! Resources: 13 destroyed.
@Rughma-Malik ▣ ~/Lab12 $ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.3",
  "serial": 94,
  "lineage": "91f4a5b2-025e-5a4e-90cf-f7d49c1ac502",
  "outputs": {},
  "resources": [],
  "check_results": null
}
@Rughma-Malik ▣ ~/Lab12 $ _
```

4. List all project files:

tree



```
ec2-user@ip-10-0-10-252:~
@Rughma-Malik ▣ ~/Lab12 $ tree
.
├── apache.sh
├── entry-script.sh
├── locals.tf
├── main.tf
├── modules
│   ├── subnet
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   └── webserver
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── terraform.tfstate
├── terraform.tfstate.backup
├── terraform.tfvars
└── variables.tf

4 directories, 15 files
@Rughma-Malik ▣ ~/Lab12 $
```