Cloud Computing Lab

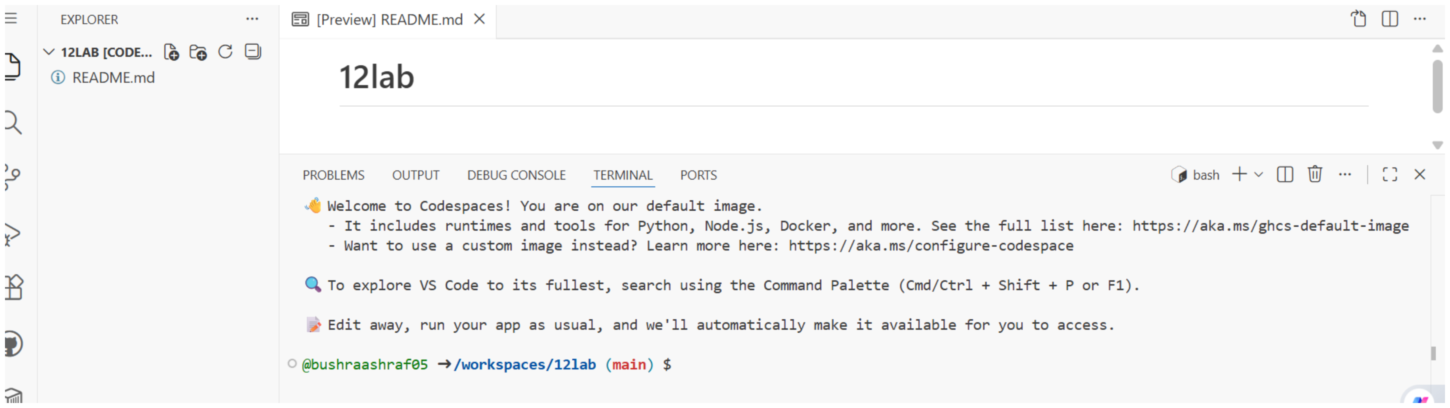
Lab 12 tasks

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

Submitted To: Muhammad Shoaib

Submitted From: Bushra Ashraf Bhatti

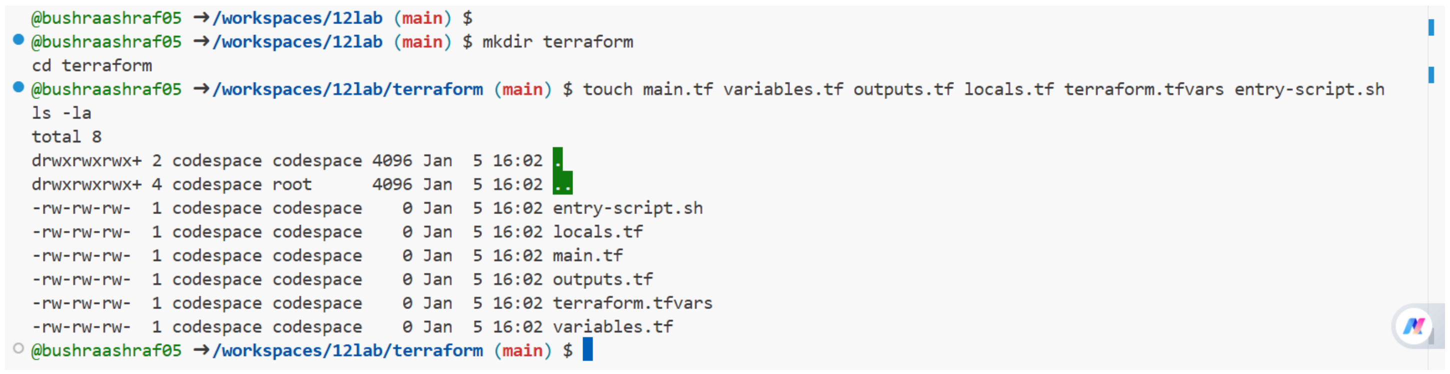
Registration Number: 2023-BSE-015



12lab

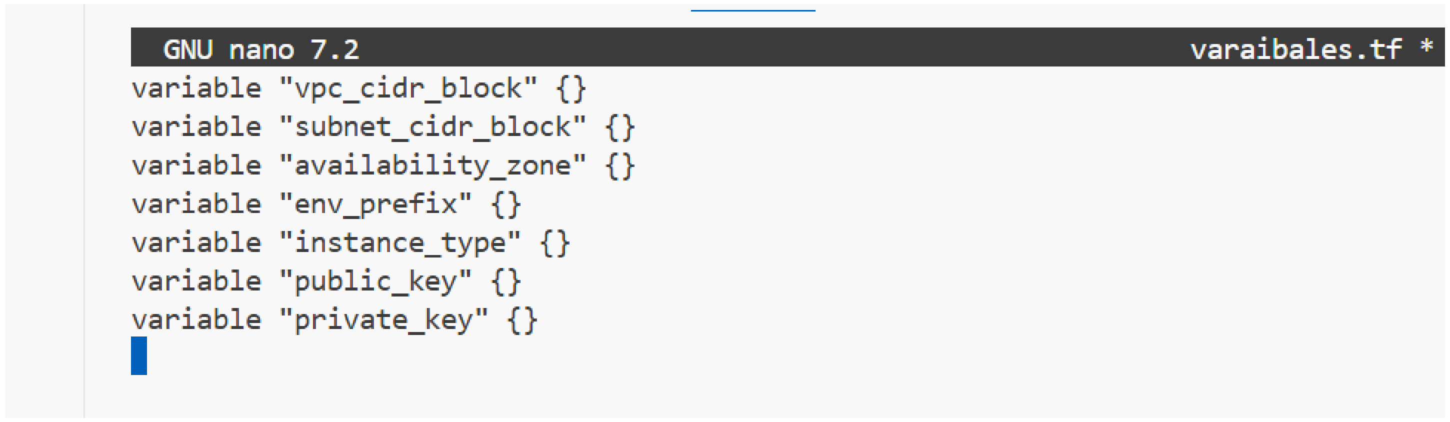
```
👋 Welcome to Codespaces! You are on our default image.
   - It includes runtimes and tools for Python, Node.js, Docker, and more. See the full list here: https://aka.ms/ghcs-default-image
   - Want to use a custom image instead? Learn more here: https://aka.ms/configure-codespace

🔍 To explore VS Code to its fullest, search using the Command Palette (Cmd/Ctrl + Shift + P or F1).

📝 Edit away, run your app as usual, and we'll automatically make it available for you to access.

@bushraashraf05 ➜/workspaces/12lab (main) $
```

Task 1 — Organize Terraform code into separate files



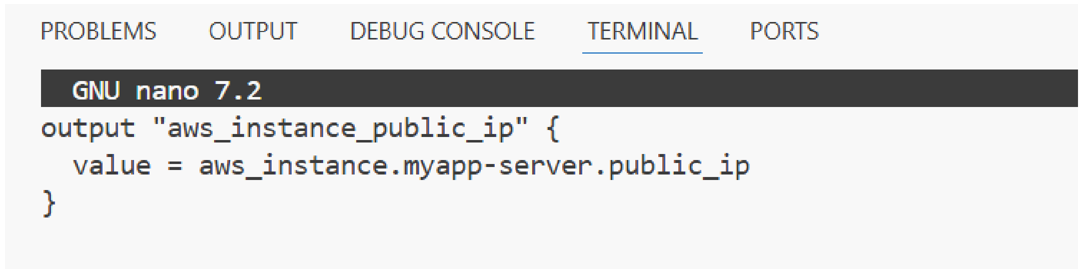
```
@bushraashraf05 ➜/workspaces/12lab (main) $
@bushraashraf05 ➜/workspaces/12lab (main) $ mkdir terraform
cd terraform
@bushraashraf05 ➜/workspaces/12lab/terraform (main) $ touch main.tf variables.tf outputs.tf locals.tf terraform.tfvars entry-script.sh
ls -la
total 8
drwxrwxrwx+ 2 codespace codespace 4096 Jan  5 16:02 .
drwxrwxrwx+ 4 codespace root      4096 Jan  5 16:02 ..
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 entry-script.sh
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 locals.tf
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 main.tf
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 outputs.tf
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 terraform.tfvars
-rw-rw-rw-  1 codespace codespace    0 Jan  5 16:02 variables.tf
@bushraashraf05 ➜/workspaces/12lab/terraform (main) $
```

Varaibles.tf:

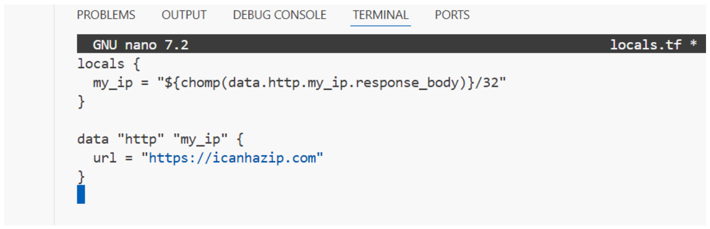


```
  GNU nano 7.2                                                      varaibales.tf *
variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "instance_type" {}
variable "public_key" {}
variable "private_key" {}
```

Outputs.tf:

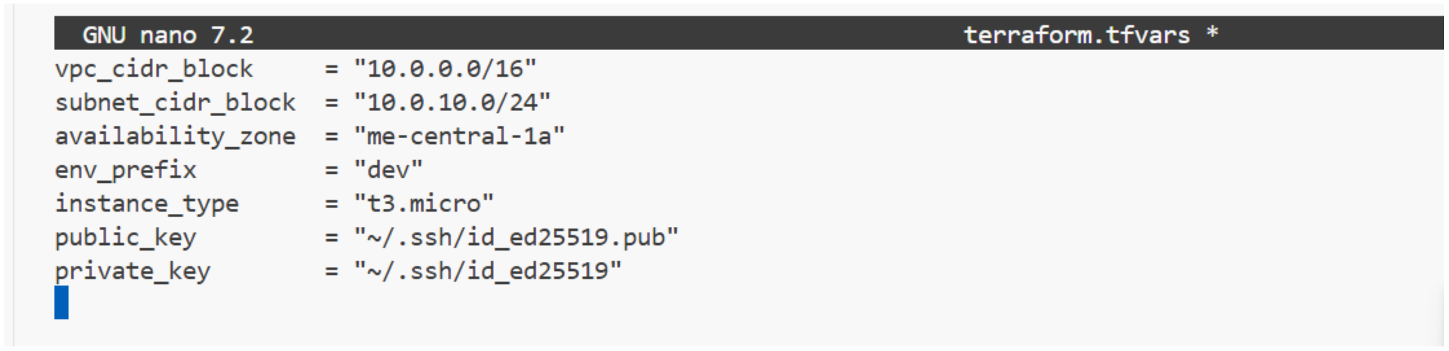


```
  GNU nano 7.2
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
 GNU nano 7.2                                          locals.tf *
locals {
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}

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

Terraform.tfvars:

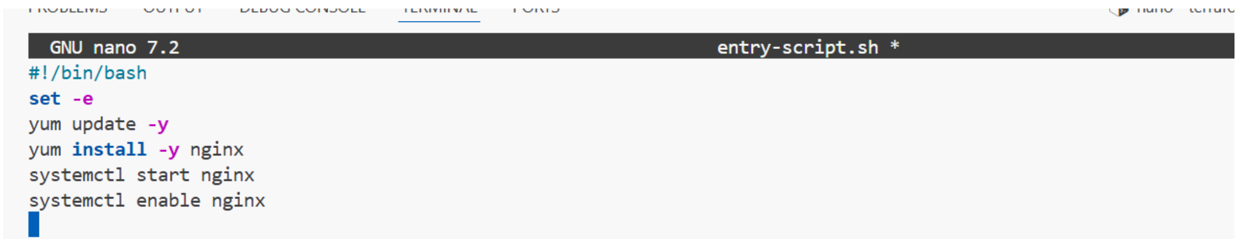


```
 GNU nano 7.2                                     terraform.tfvars *
vpc_cidr_block      = "10.0.0.0/16"
subnet_cidr_block   = "10.0.10.0/24"
availability_zone   = "me-central-1a"
env_prefix          = "dev"
instance_type       = "t3.micro"
public_key          = "~/.ssh/id_ed25519.pub"
private_key         = "~/.ssh/id_ed25519"
```

Main.tf:

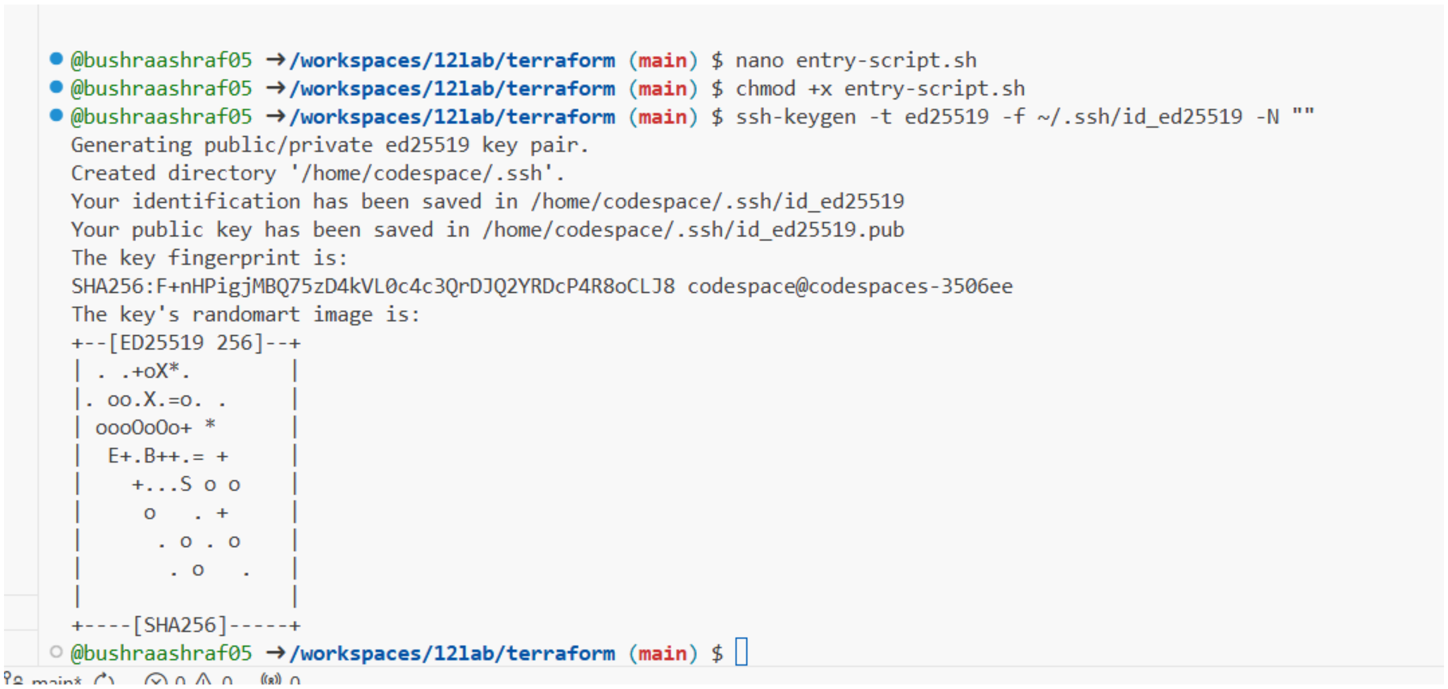
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}

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 = {
     Name = "${var.env_prefix}-rt"
  }
}

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

resource "aws_default_security_group" "default_sg" {
  vpc_id      = aws_vpc.myapp_vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [local.my_ip]
  }
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
    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" # 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"
  }
}
```

Entry script.sh:



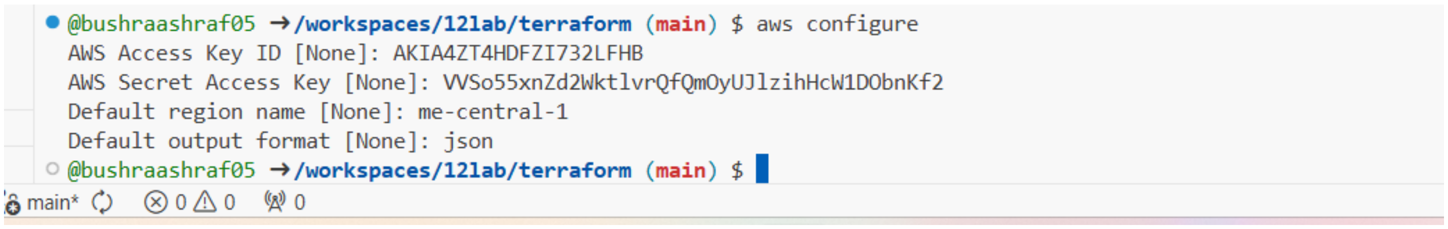
```
 GNU nano 7.2                                  entry-script.sh *
#!/bin/bash
set -e
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
```

GENERATE SSH KEYS (INSIDE CODESPACE):

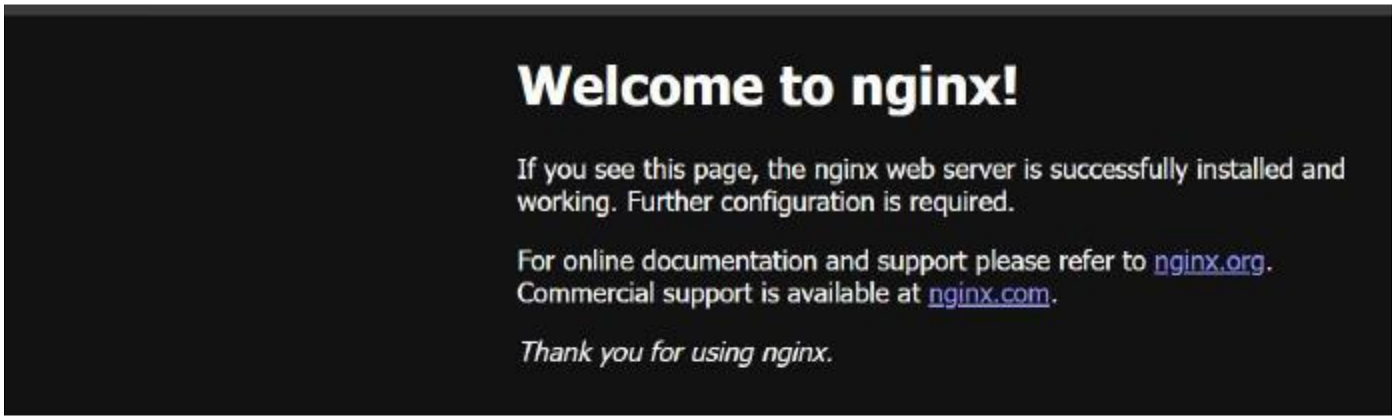


```
● @bushraashraf05 →/workspaces/12lab/terraform (main) $ nano entry-script.sh
● @bushraashraf05 →/workspaces/12lab/terraform (main) $ chmod +x entry-script.sh
● @bushraashraf05 →/workspaces/12lab/terraform (main) $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""
  Generating public/private ed25519 key pair.
  Created directory '/home/codespace/.ssh'.
  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:F+nHPigjMBQ75zD4kVL0c4c3QrDJQ2YRDcP4R8oCLJ8 codespace@codespaces-3506ee
  The key's randomart image is:
  +--[ED25519 256]--+
  | . .+oX*.        |
  |. oo.X.=o. .     |
  | oooOoOo+ *      |
  |  E+.B++.= +     |
  |    +...S o o    |
  |     o    . +    |
  |      . o . o    |
  |       . o   .   |
  |                 |
  +----[SHA256]-----+
○ @bushraashraf05 →/workspaces/12lab/terraform (main) $
```

Then make the iam user and install terraform and aws configure and then:



```
● @bushraashraf05 →/workspaces/12lab/terraform (main) $ aws configure
  AWS Access Key ID [None]: AKIA4ZT4HDFZI732LFHB
  AWS Secret Access Key [None]: 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
  Default region name [None]: me-central-1
  Default output format [None]: json
○ @bushraashraf05 →/workspaces/12lab/terraform (main) $
```



Welcome to nginx!

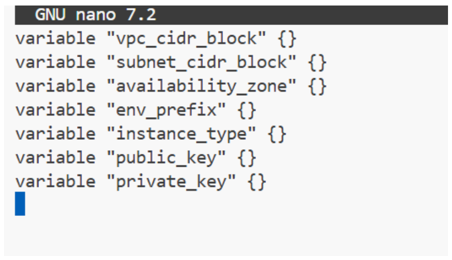
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

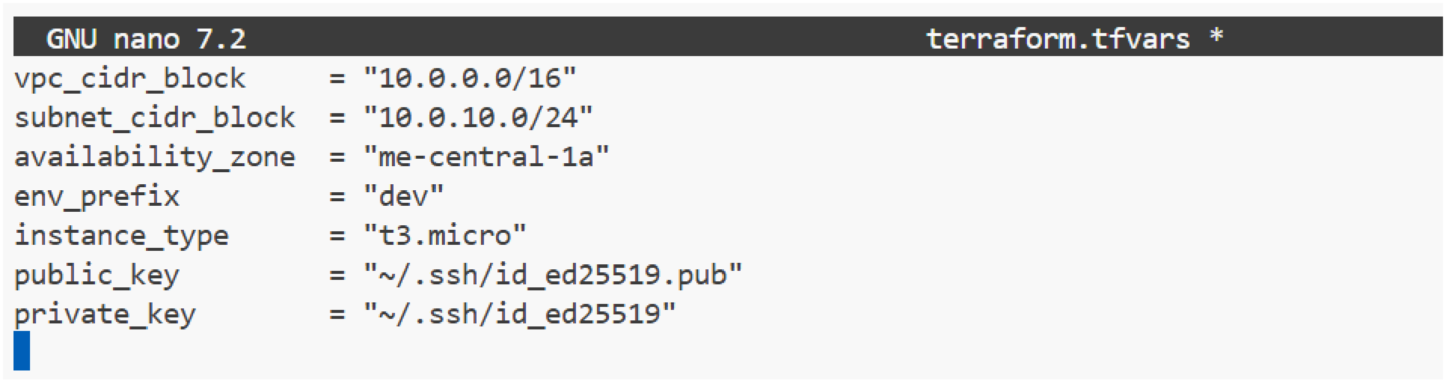
Thank you for using nginx.

Task2: Use remote-exec Provisioner

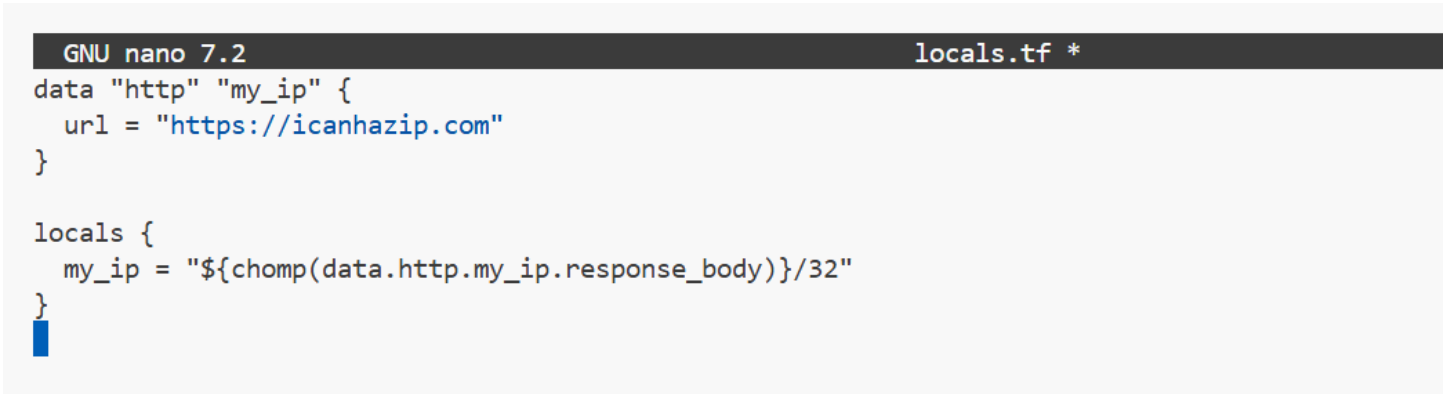
variables.tf:



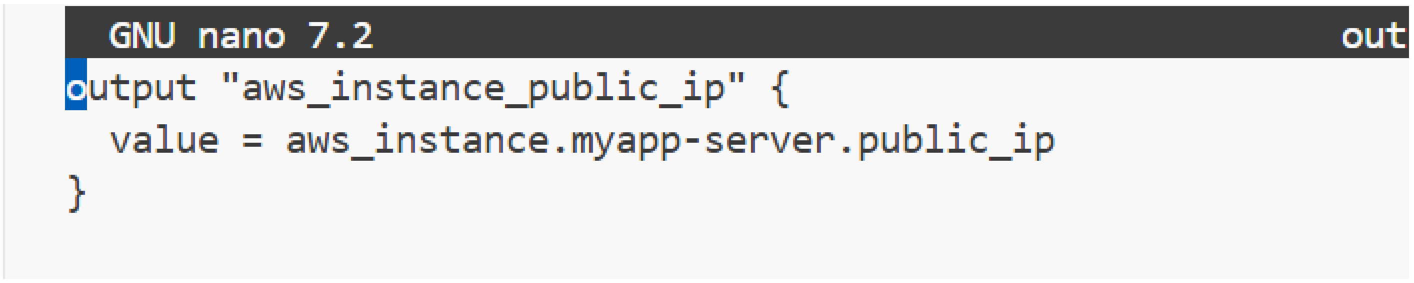
```
  GNU nano 7.2
variable "vpc_cidr_block" {}
variable "subnet_cidr_block" {}
variable "availability_zone" {}
variable "env_prefix" {}
variable "instance_type" {}
variable "public_key" {}
variable "private_key" {}
```



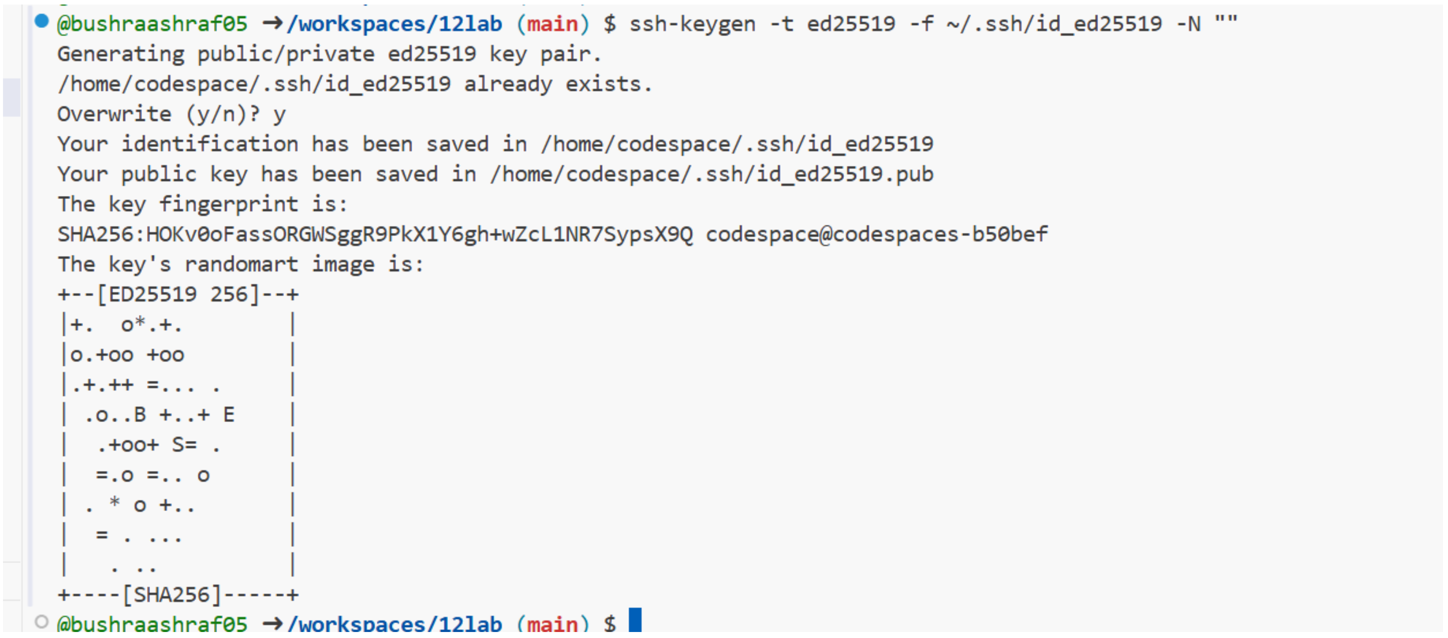
```
  GNU nano 7.2                              terraform.tfvars *
vpc_cidr_block      = "10.0.0.0/16"
subnet_cidr_block   = "10.0.10.0/24"
availability_zone   = "me-central-1a"
env_prefix          = "dev"
instance_type       = "t3.micro"
public_key          = "~/.ssh/id_ed25519.pub"
private_key         = "~/.ssh/id_ed25519"
```



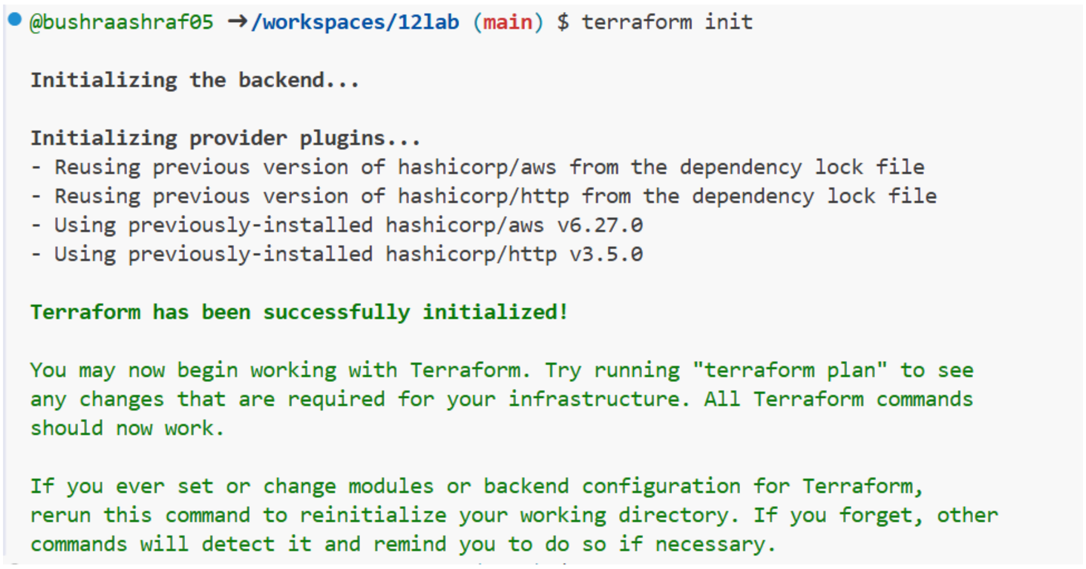
```
  GNU nano 7.2                              locals.tf *
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

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



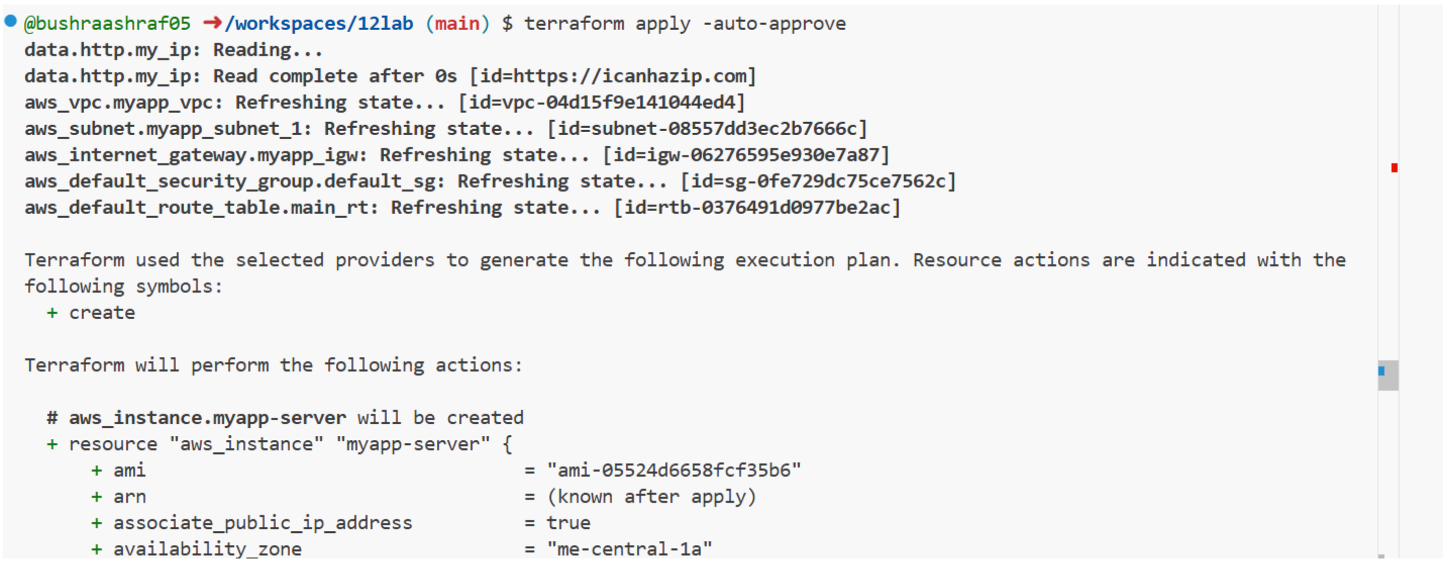
```
  GNU nano 7.2                                     out
output "aws_instance_public_ip" {
  value = aws_instance.myapp-server.public_ip
}
```



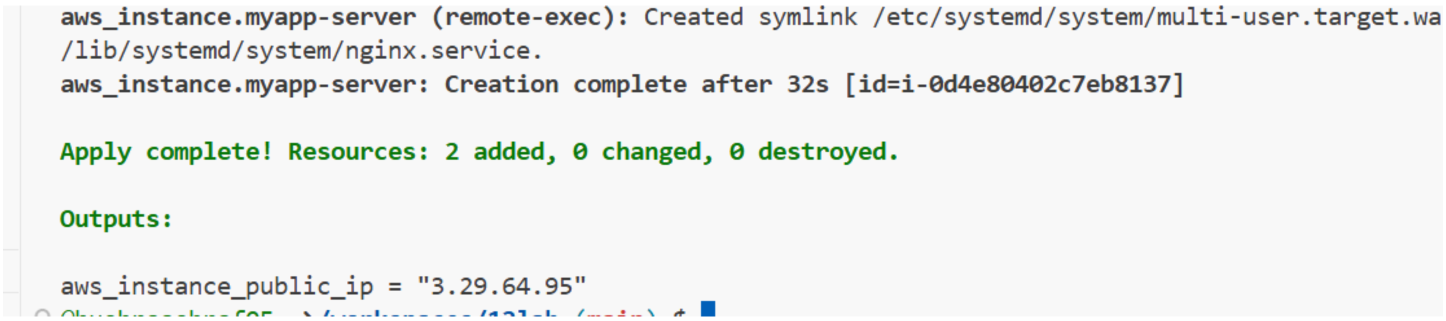
```
● @bushraashraf05 ➜ /workspaces/12lab (main) $ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""
  Generating public/private ed25519 key pair.
  /home/codespace/.ssh/id_ed25519 already exists.
  Overwrite (y/n)? y
  Your identification has been saved in /home/codespace/.ssh/id_ed25519
  Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
  The key fingerprint is:
  SHA256:HOKv0oFassORGWSggR9PkX1Y6gh+wZcL1NR7SypsX9Q codespace@codespaces-b50bef
  The key's randomart image is:
  +--[ED25519 256]--+
  |+.  o*.+.        |
  |o.+oo +oo        |
  |.+.++ =... .     |
  | .o..B +..+ E    |
  |  .+oo+ S= .     |
  |  =.o =.. o      |
  | . * o +..       |
  |  = . ...        |
  |   . ..          |
  +----[SHA256]-----+
○ @bushraashraf05 ➜ /workspaces/12lab (main) $
```



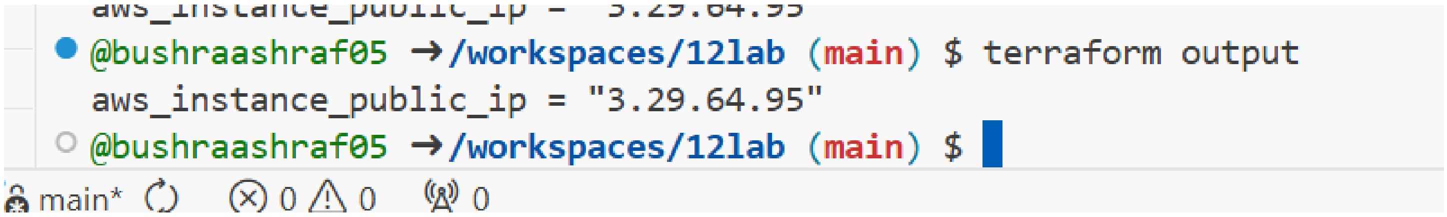
```
@bushraashraf05 →/workspaces/12lab (main) $ terraform init

Initializing the backend...

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.
```



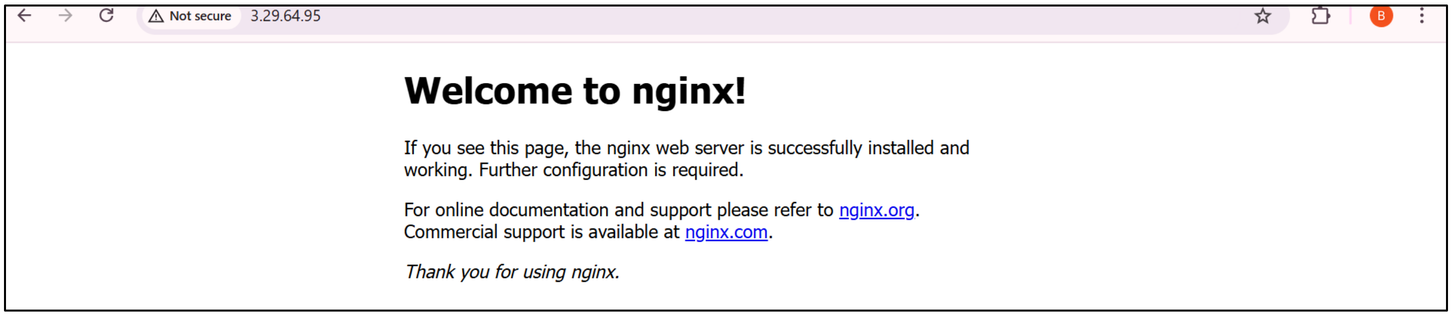
```
@bushraashraf05 →/workspaces/12lab (main) $ terraform apply -auto-approve
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
aws_vpc.myapp_vpc: Refreshing state... [id=vpc-04d15f9e141044ed4]
aws_subnet.myapp_subnet_1: Refreshing state... [id=subnet-08557dd3ec2b7666c]
aws_internet_gateway.myapp_igw: Refreshing state... [id=igw-06276595e930e7a87]
aws_default_security_group.default_sg: Refreshing state... [id=sg-0fe729dc75ce7562c]
aws_default_route_table.main_rt: Refreshing state... [id=rtb-0376491d0977be2ac]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.myapp-server will be created
  + resource "aws_instance" "myapp-server" {
      + ami                                  = "ami-05524d6658fcf35b6"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = "me-central-1a"
```



```
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wa
/lib/systemd/system/nginx.service.
aws_instance.myapp-server: Creation complete after 32s [id=i-0d4e80402c7eb8137]

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

Outputs:

aws_instance_public_ip = "3.29.64.95"
```



```
aws_instance_public_ip = "3.29.64.95"
@bushraashraf05 →/workspaces/12lab (main) $ terraform output
aws_instance_public_ip = "3.29.64.95"
@bushraashraf05 →/workspaces/12lab (main) $
```

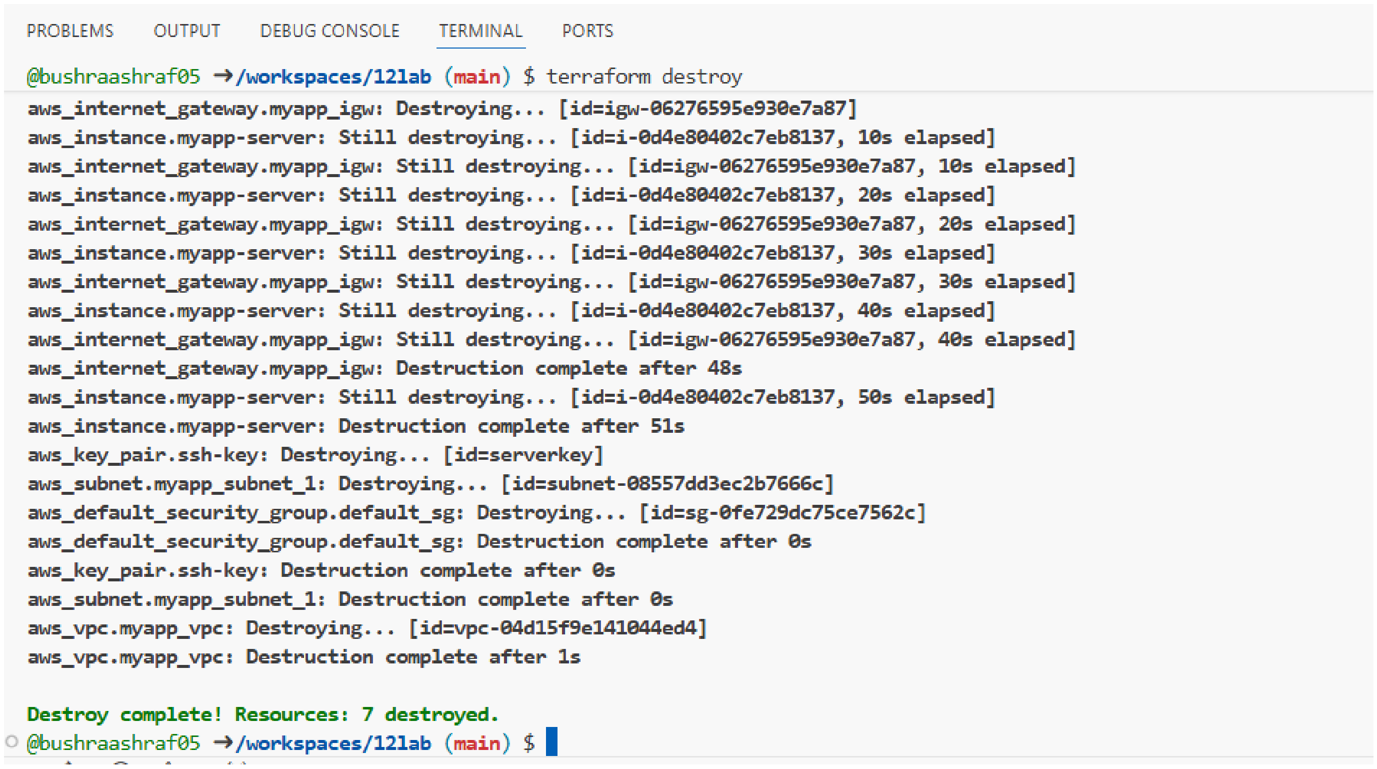

Not secure 3.29.64.95

Welcome to nginx!

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

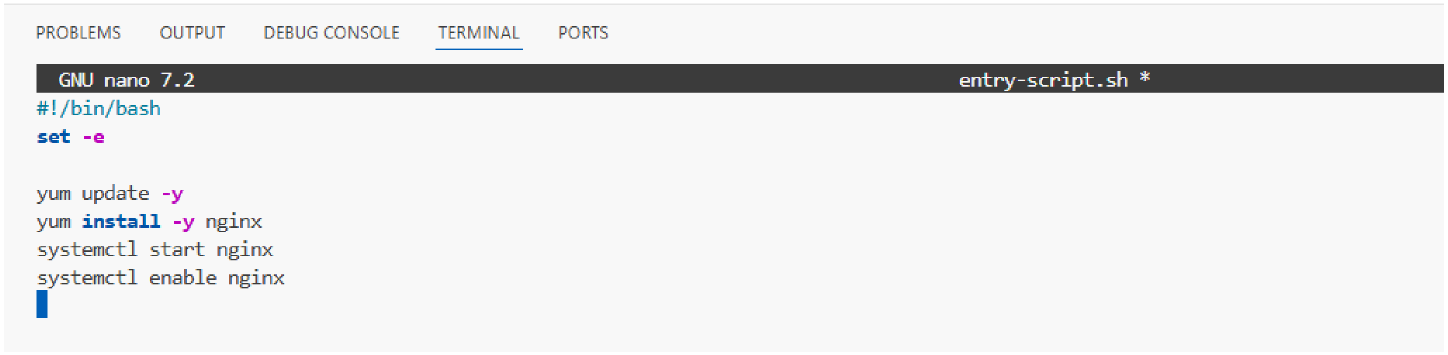
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



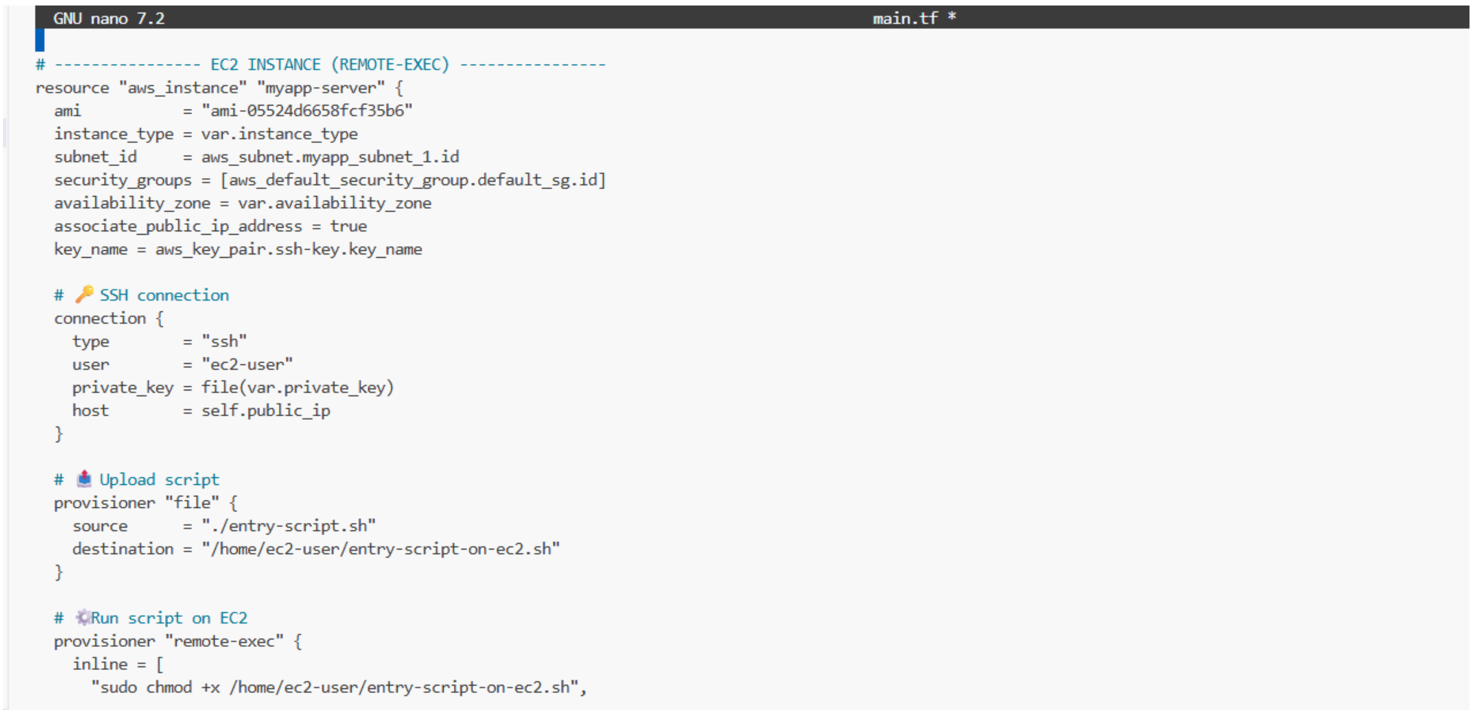
```
@bushraashraf05 →/workspaces/12lab (main) $ terraform destroy
aws_internet_gateway.myapp_igw: Destroying... [id=igw-06276595e930e7a87]
aws_instance.myapp-server: Still destroying... [id=i-0d4e80402c7eb8137, 10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06276595e930e7a87, 10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0d4e80402c7eb8137, 20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06276595e930e7a87, 20s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0d4e80402c7eb8137, 30s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06276595e930e7a87, 30s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0d4e80402c7eb8137, 40s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-06276595e930e7a87, 40s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 48s
aws_instance.myapp-server: Still destroying... [id=i-0d4e80402c7eb8137, 50s elapsed]
aws_instance.myapp-server: Destruction complete after 51s
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-08557dd3ec2b7666c]
aws_default_security_group.default_sg: Destroying... [id=sg-0fe729dc75ce7562c]
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-04d15f9e141044ed4]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@bushraashraf05 →/workspaces/12lab (main) $
```

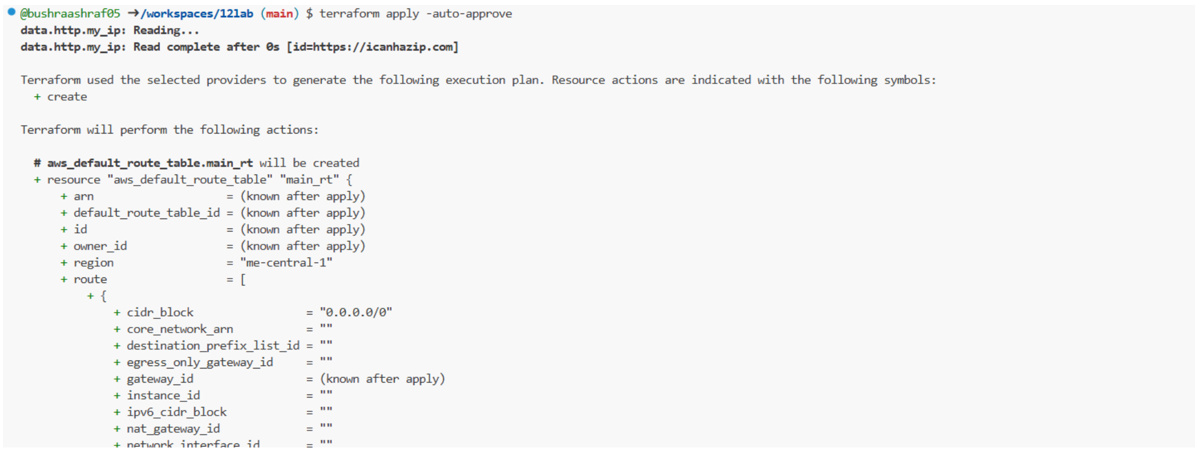
Task 3 — Use file and local-exec provisioners

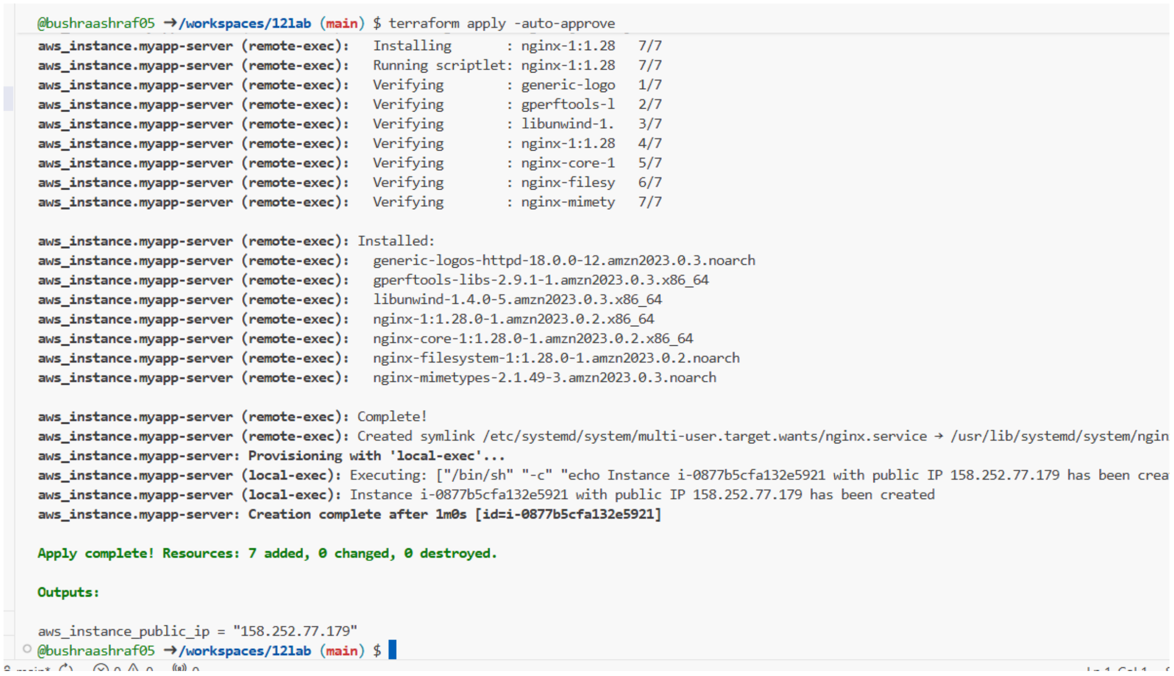


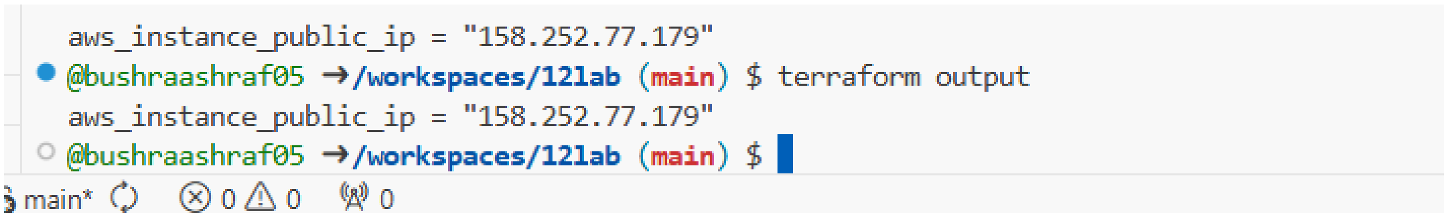
```
#!/bin/bash
set -e

yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx
```

Edit main.tf:

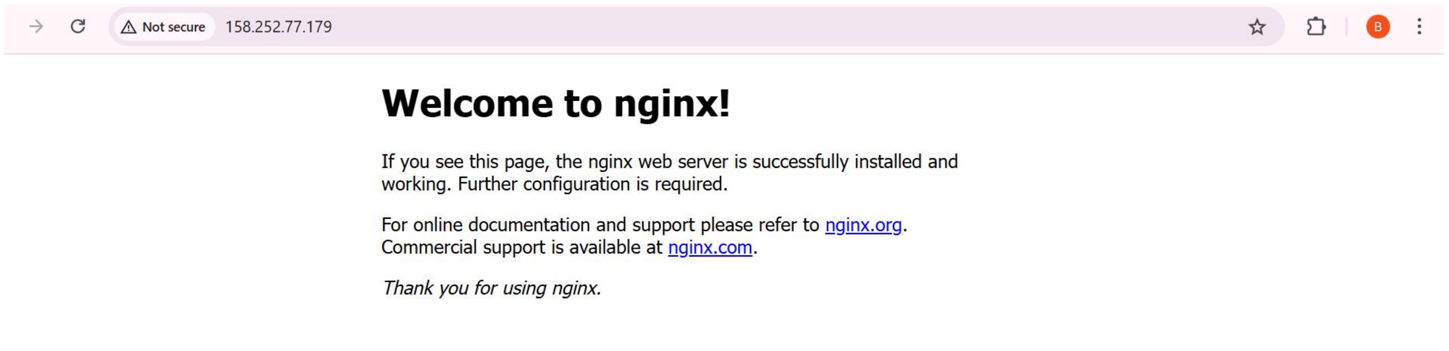

```
GNU nano 7.2                                                                    main.tf *

# ---------------- EC2 INSTANCE (REMOTE-EXEC) ----------------
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

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

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

  # ⚙️Run script on EC2
  provisioner "remote-exec" {
    inline = [
      "sudo chmod +x /home/ec2-user/entry-script-on-ec2.sh",
```


```
@bushraashraf05 →/workspaces/12lab (main) $ terraform apply -auto-approve
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_default_route_table.main_rt will be created
  + resource "aws_default_route_table" "main_rt" {
      + arn                    = (known after apply)
      + default_route_table_id = (known after apply)
      + id                     = (known after apply)
      + owner_id               = (known after apply)
      + region                 = "me-central-1"
      + route                  = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = (known after apply)
              + instance_id                = ""
              + ipv6_cidr_block            = ""
              + nat_gateway_id             = ""
              + network_interface_id       = ""
```


```
@bushraashraf05 →/workspaces/12lab (main) $ terraform apply -auto-approve
aws_instance.myapp-server (remote-exec):   Installing       : nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Running scriptlet: nginx-1:1.28   7/7
aws_instance.myapp-server (remote-exec):   Verifying        : generic-logo   1/7
aws_instance.myapp-server (remote-exec):   Verifying        : gperftools-l   2/7
aws_instance.myapp-server (remote-exec):   Verifying        : libunwind-1.   3/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-1:1.28   4/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-core-1   5/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-filesy   6/7
aws_instance.myapp-server (remote-exec):   Verifying        : nginx-mimety   7/7

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

aws_instance.myapp-server (remote-exec): Complete!
aws_instance.myapp-server (remote-exec): Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/ngin
aws_instance.myapp-server: Provisioning with 'local-exec'...
aws_instance.myapp-server (local-exec): Executing: ["/bin/sh" "-c" "echo Instance i-0877b5cfa132e5921 with public IP 158.252.77.179 has been crea
aws_instance.myapp-server (local-exec): Instance i-0877b5cfa132e5921 with public IP 158.252.77.179 has been created
aws_instance.myapp-server: Creation complete after 1m0s [id=i-0877b5cfa132e5921]

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

Outputs:

aws_instance_public_ip = "158.252.77.179"
@bushraashraf05 →/workspaces/12lab (main) $
```



```
  aws_instance_public_ip = "158.252.77.179"
● @bushraashraf05 ➜ /workspaces/121lab (main) $ terraform output
  aws_instance_public_ip = "158.252.77.179"
○ @bushraashraf05 ➜ /workspaces/121lab (main) $ 
```

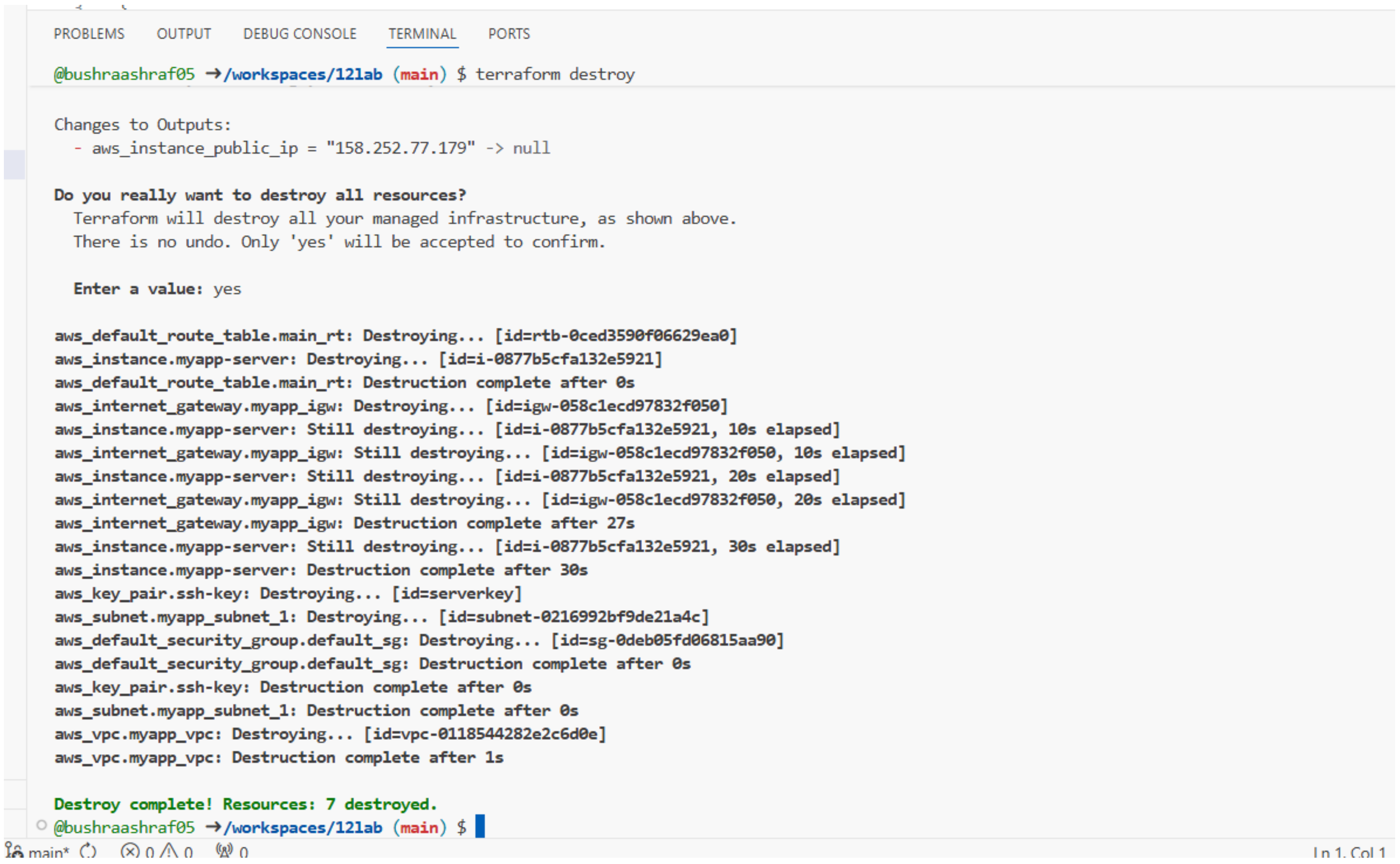


Welcome to nginx!

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

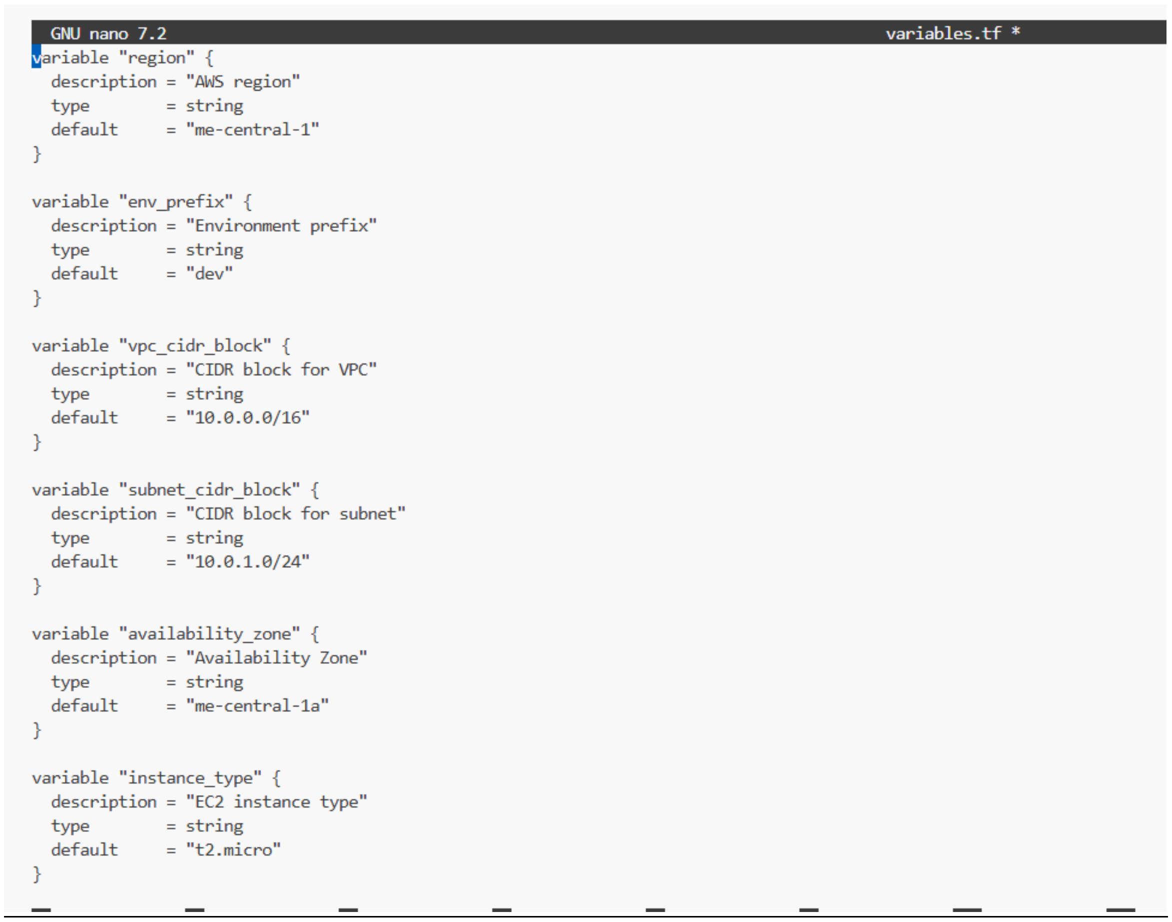
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

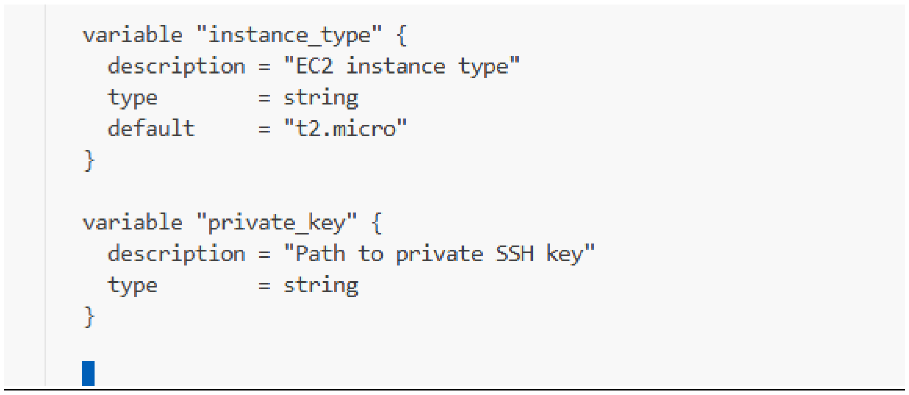


```
@bushraashraf05 ➜ /workspaces/121lab (main) $ terraform destroy

Changes to Outputs:
  - aws_instance_public_ip = "158.252.77.179" -> null

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

  Enter a value: yes

aws_default_route_table.main_rt: Destroying... [id=rtb-0ced3590f06629ea0]
aws_instance.myapp-server: Destroying... [id=i-0877b5cfa132e5921]
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-058c1ecd97832f050]
aws_instance.myapp-server: Still destroying... [id=i-0877b5cfa132e5921, 10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-058c1ecd97832f050, 10s elapsed]
aws_instance.myapp-server: Still destroying... [id=i-0877b5cfa132e5921, 20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-058c1ecd97832f050, 20s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 27s
aws_instance.myapp-server: Still destroying... [id=i-0877b5cfa132e5921, 30s elapsed]
aws_instance.myapp-server: Destruction complete after 30s
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0216992bf9de21a4c]
aws_default_security_group.default_sg: Destroying... [id=sg-0deb05fd06815aa90]
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-0118544282e2c6d0e]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
○ @bushraashraf05 ➜ /workspaces/121lab (main) $ 
```

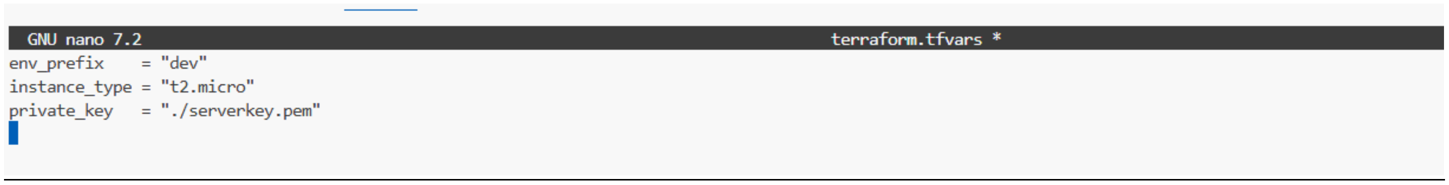
TASK 4 — Use Variables, Locals & Outputs Properly:



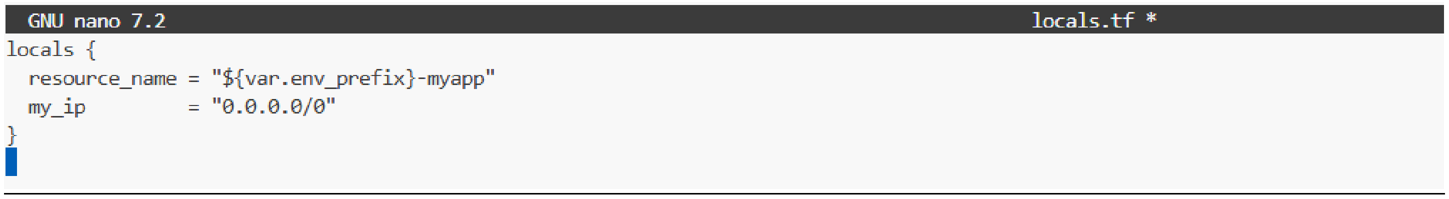
```
GNU nano 7.2                                                      variables.tf *
variable "region" {
  description = "AWS region"
  type        = string
  default     = "me-central-1"
}

variable "env_prefix" {
  description = "Environment prefix"
  type        = string
  default     = "dev"
}

variable "vpc_cidr_block" {
  description = "CIDR block for VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "subnet_cidr_block" {
  description = "CIDR block for subnet"
  type        = string
  default     = "10.0.1.0/24"
}

variable "availability_zone" {
  description = "Availability Zone"
  type        = string
  default     = "me-central-1a"
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}
```



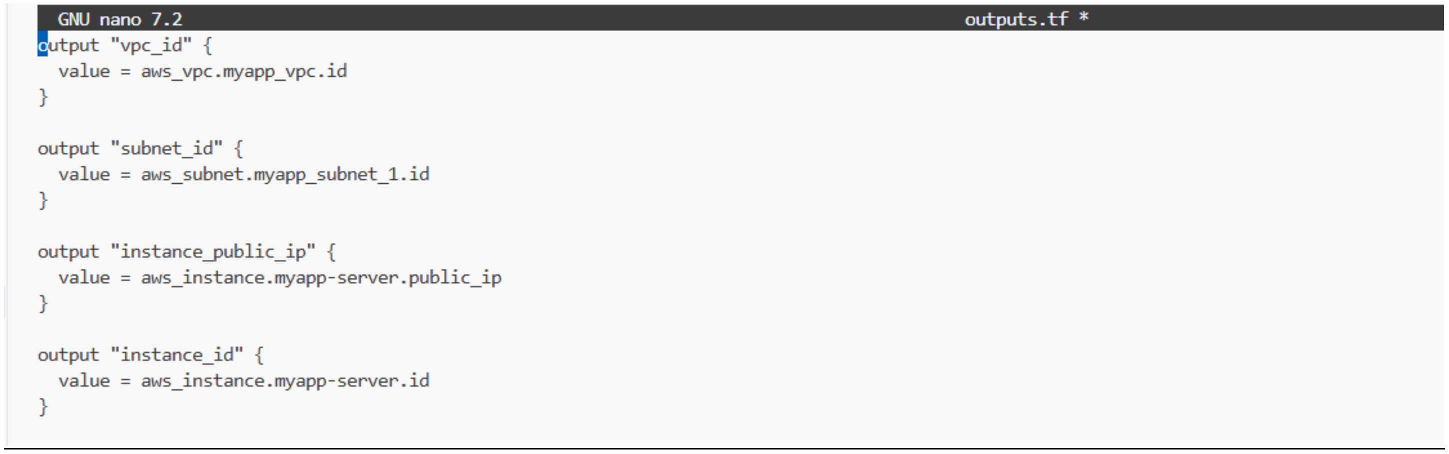
```
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}

variable "private_key" {
  description = "Path to private SSH key"
  type        = string
}
```

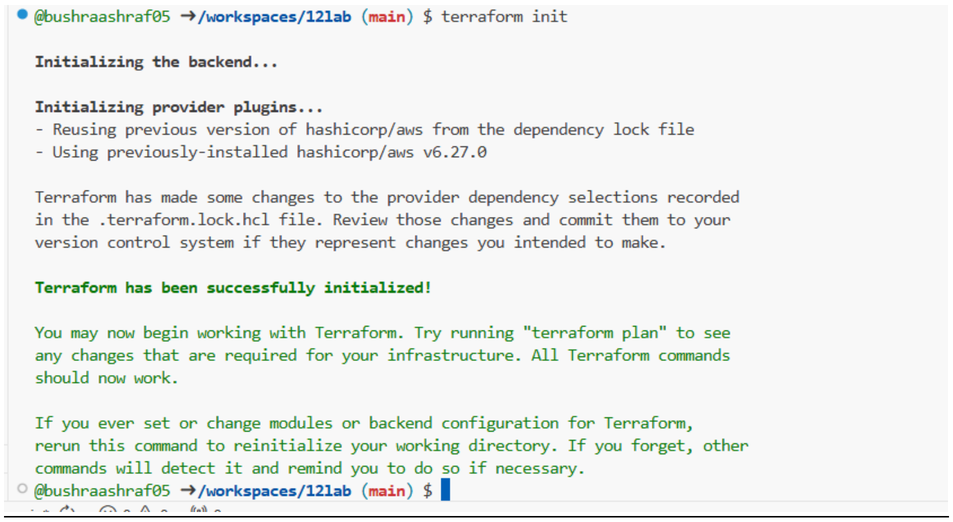


```
GNU nano 7.2                                                terraform.tfvars *
env_prefix    = "dev"
instance_type = "t2.micro"
private_key   = "./serverkey.pem"
```

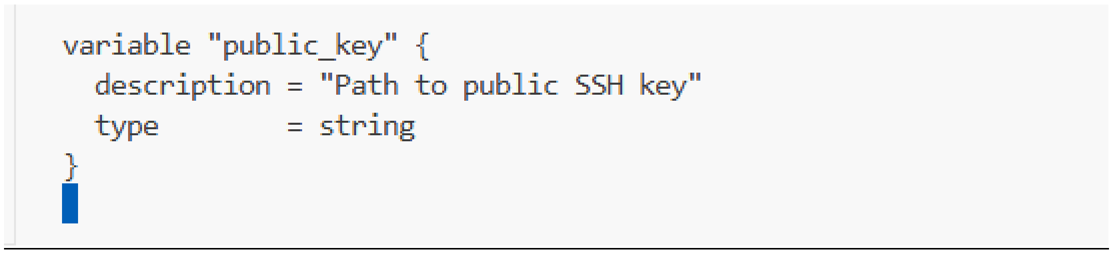


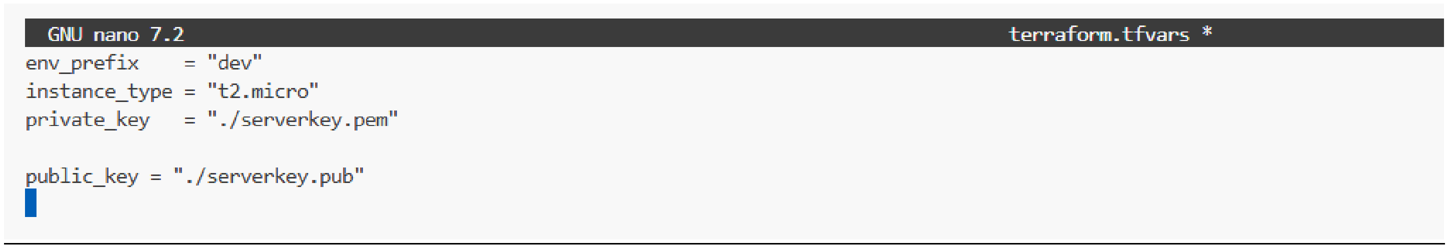
```
GNU nano 7.2                                                      locals.tf *
locals {
  resource_name = "${var.env_prefix}-myapp"
  my_ip         = "0.0.0.0/0"
}
```

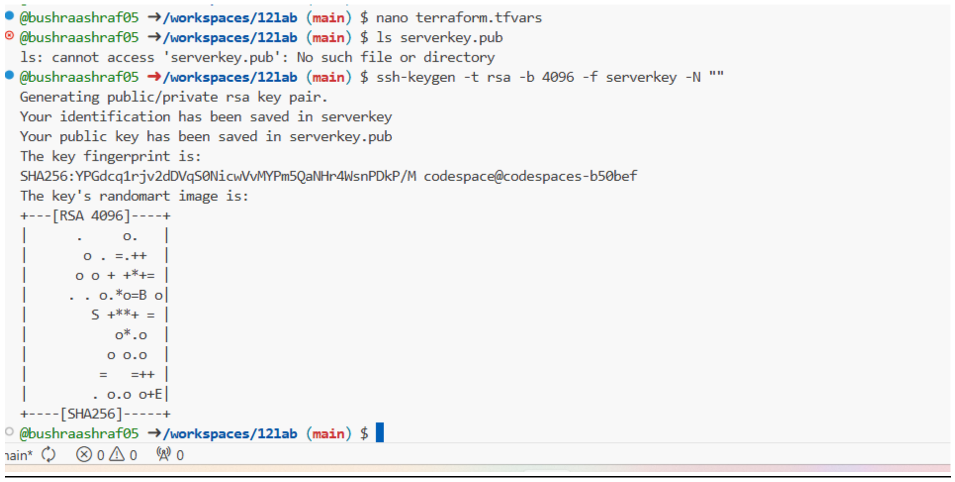

```
 GNU nano 7.2                                                         outputs.tf *
output "vpc_id" {
  value = aws_vpc.myapp_vpc.id
}

output "subnet_id" {
  value = aws_subnet.myapp_subnet_1.id
}

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

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


```
● @bushraashraf05 →/workspaces/12lab (main) $ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.
○ @bushraashraf05 →/workspaces/12lab (main) $
```

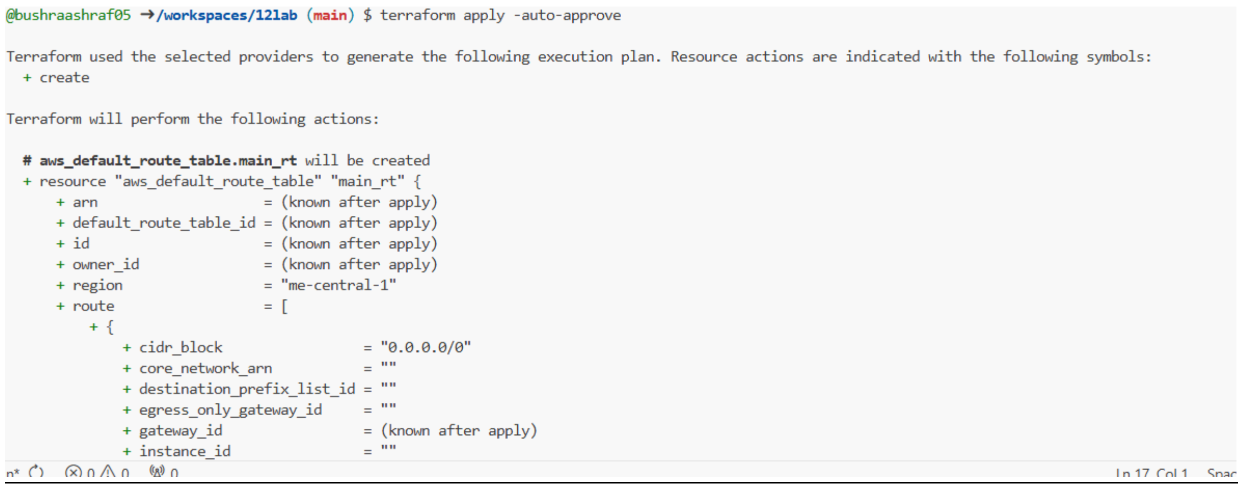
Edit main.tf:


```
variable "public_key" {
  description = "Path to public SSH key"
  type        = string
}
```

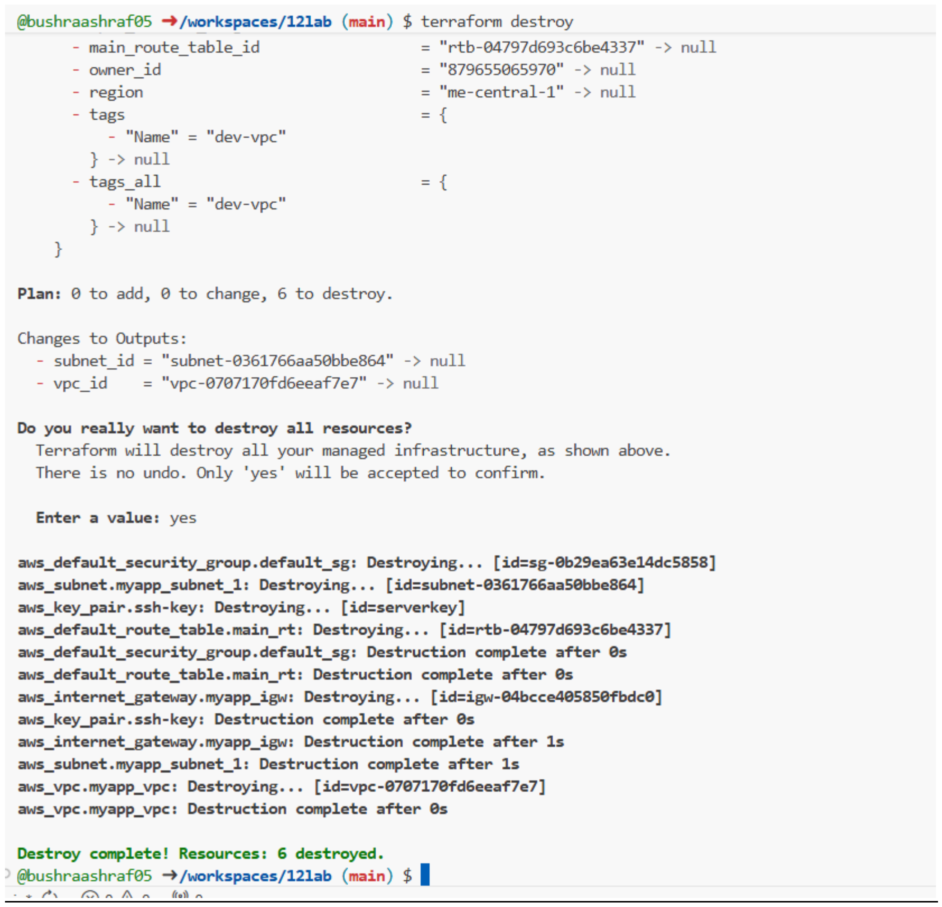

```
 GNU nano 7.2                                                         terraform.tfvars *
env_prefix    = "dev"
instance_type = "t2.micro"
private_key   = "./serverkey.pem"

public_key = "./serverkey.pub"
```



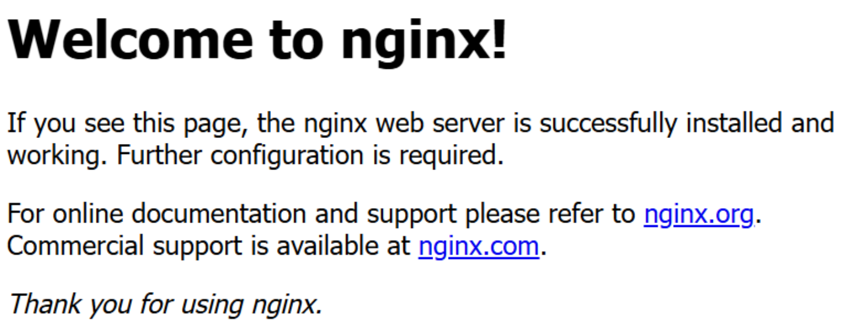
```
@bushraashraf05 →/workspaces/12lab (main) $ nano terraform.tfvars
@bushraashraf05 →/workspaces/12lab (main) $ ls serverkey.pub
ls: cannot access 'serverkey.pub': No such file or directory
@bushraashraf05 →/workspaces/12lab (main) $ ssh-keygen -t rsa -b 4096 -f serverkey -N ""
Generating public/private rsa key pair.
Your identification has been saved in serverkey
Your public key has been saved in serverkey.pub
The key fingerprint is:
SHA256:YPGdcq1rjv2dDVqS0NicwVvMYPm5QaNHr4WsnPDkP/M codespace@codespaces-b50bef
The key's randomart image is:
+---[RSA 4096]----+
|      .     o.   |
|       o . =.++  |
|      o o + +*+= |
|     . . o.*o=B o|
|        S +**+ = |
|           o*.o  |
|          o o.o  |
|         =   =++ |
|        . o.o o+E|
+----[SHA256]-----+
@bushraashraf05 →/workspaces/12lab (main) $
```



```
@bushraashraf05 →/workspaces/12lab (main) $ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_default_route_table.main_rt will be created
  + resource "aws_default_route_table" "main_rt" {
      + arn                    = (known after apply)
      + default_route_table_id = (known after apply)
      + id                     = (known after apply)
      + owner_id               = (known after apply)
      + region                 = "me-central-1"
      + route                  = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = (known after apply)
              + instance_id                = ""
```



```
@bushraashraf05 →/workspaces/12lab (main) $ terraform destroy
      - main_route_table_id              = "rtb-04797d693c6be4337" -> null
      - owner_id                         = "879655065970" -> null
      - region                           = "me-central-1" -> null
      - tags                             = {
          - "Name" = "dev-vpc"
        } -> null
      - tags_all                         = {
          - "Name" = "dev-vpc"
        } -> null
    }

Plan: 0 to add, 0 to change, 6 to destroy.

Changes to Outputs:
  - subnet_id = "subnet-0361766aa50bbe864" -> null
  - vpc_id    = "vpc-0707170fd6eeaf7e7" -> null

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

  Enter a value: yes

aws_default_security_group.default_sg: Destroying... [id=sg-0b29ea63e14dc5858]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-0361766aa50bbe864]
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_default_route_table.main_rt: Destroying... [id=rtb-04797d693c6be4337]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-04bcce405850fbdc0]
aws_key_pair.ssh-key: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destruction complete after 1s
aws_subnet.myapp_subnet_1: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-0707170fd6eeaf7e7]
aws_vpc.myapp_vpc: Destruction complete after 0s

Destroy complete! Resources: 6 destroyed.
@bushraashraf05 →/workspaces/12lab (main) $
```


Welcome to nginx!

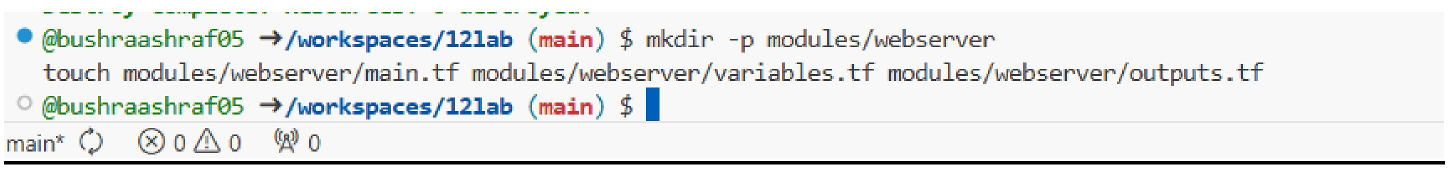
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

Thank you for using nginx.

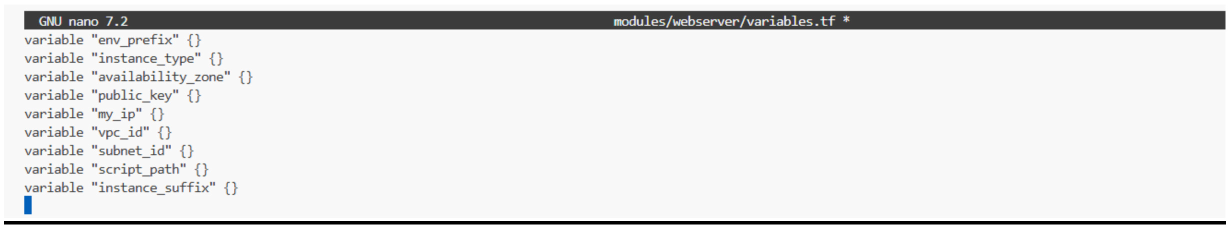
Task 5 — Create webserver module:

CREATE WEB SERVER MODULE FOLDER:

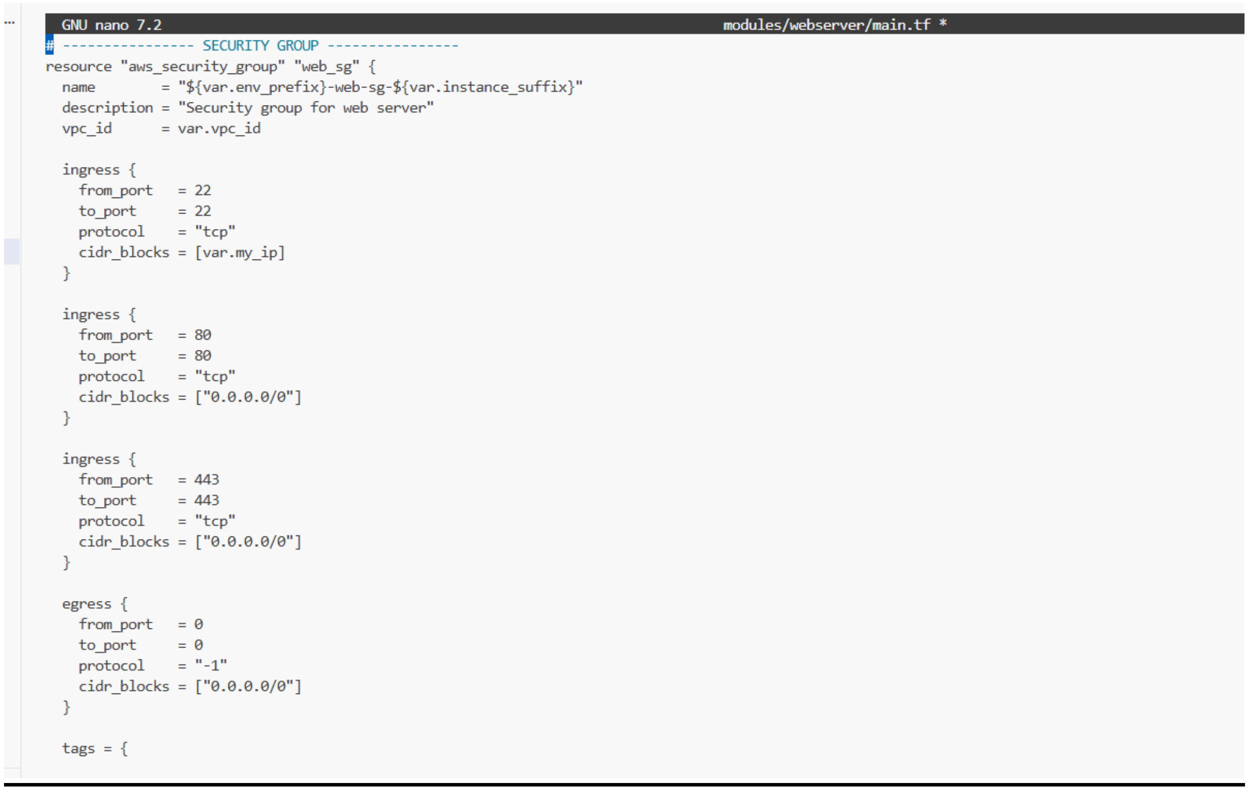


```
@bushraashraf05 →/workspaces/121ab (main) $ mkdir -p modules/webserver
touch modules/webserver/main.tf modules/webserver/variables.tf modules/webserver/outputs.tf
@bushraashraf05 →/workspaces/121ab (main) $
main*  ⓧ 0 ⚠ 0  0
```

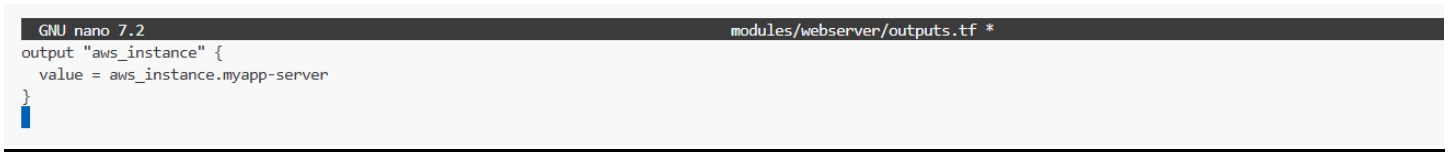
modules/webserver/variables.tf:



```
GNU nano 7.2                                  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" {}
```

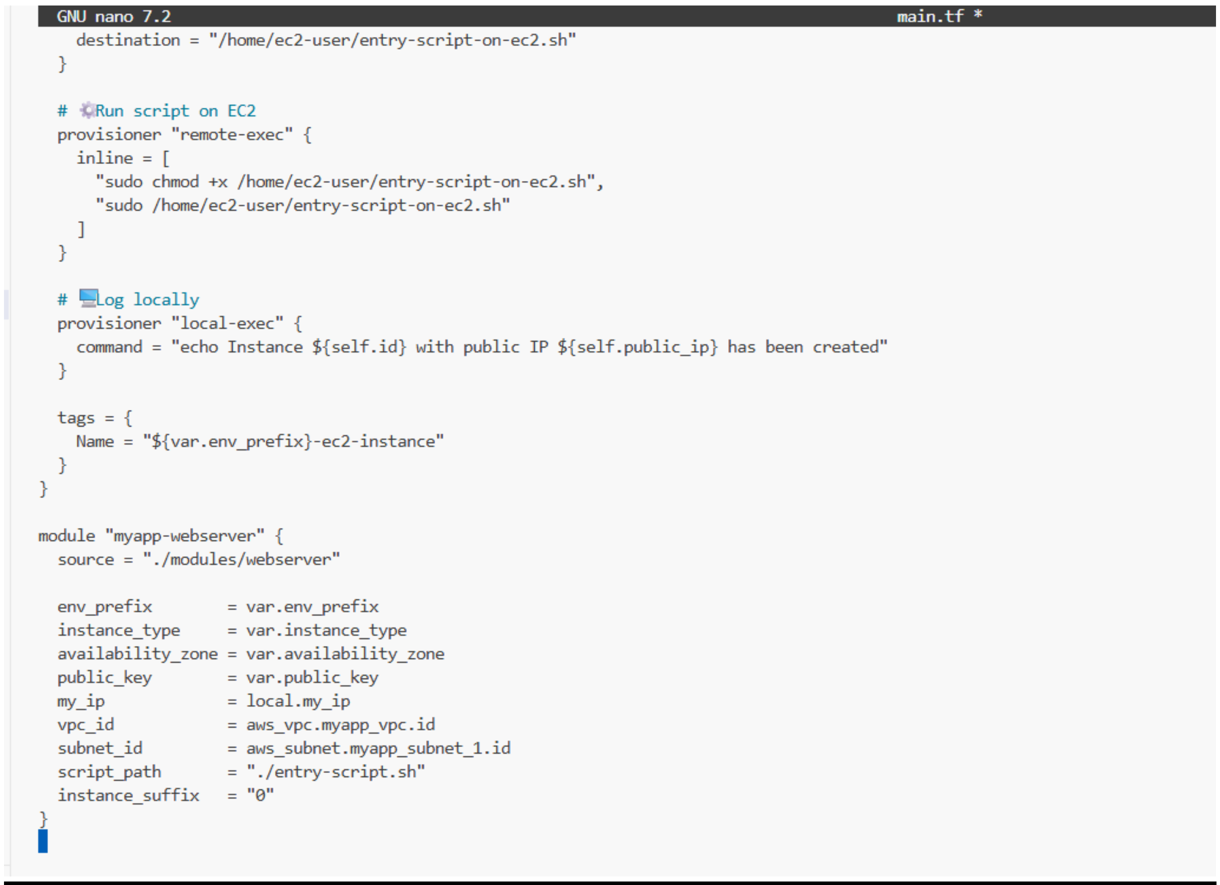


```
GNU nano 7.2                                  modules/webserver/main.tf *
# --------------- SECURITY GROUP ---------------
resource "aws_security_group" "web_sg" {
  name        = "${var.env_prefix}-web-sg-${var.instance_suffix}"
  description = "Security group for web server"
  vpc_id      = var.vpc_id

  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
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
```

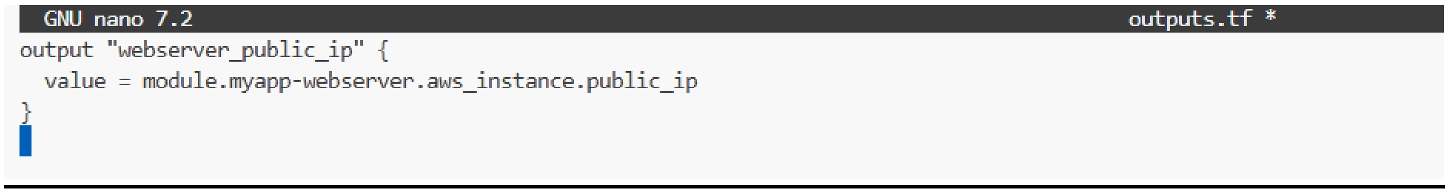


```
GNU nano 7.2                                   modules/webserver/outputs.tf *
output "aws_instance" {
  value = aws_instance.myapp-server
}
```

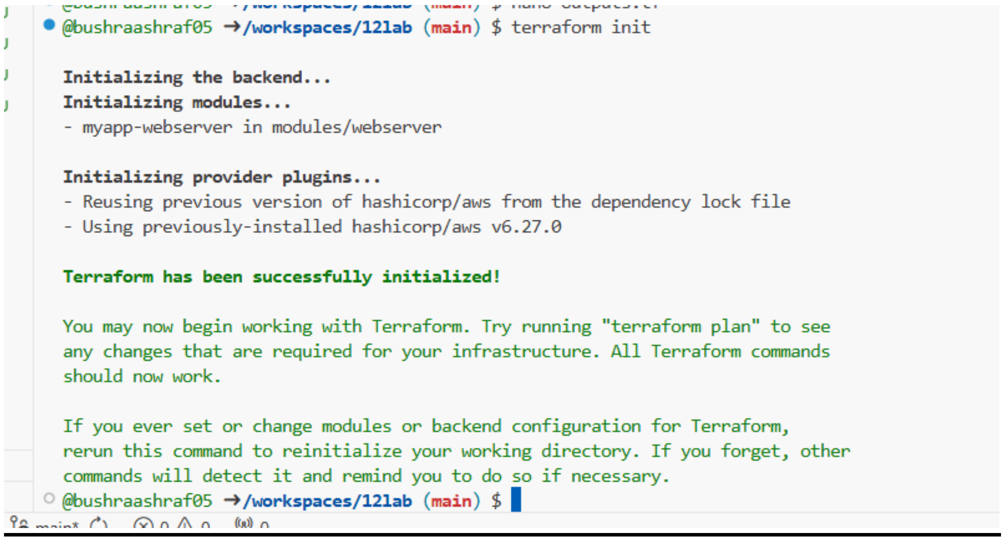
Edit main.tf and add at the end:



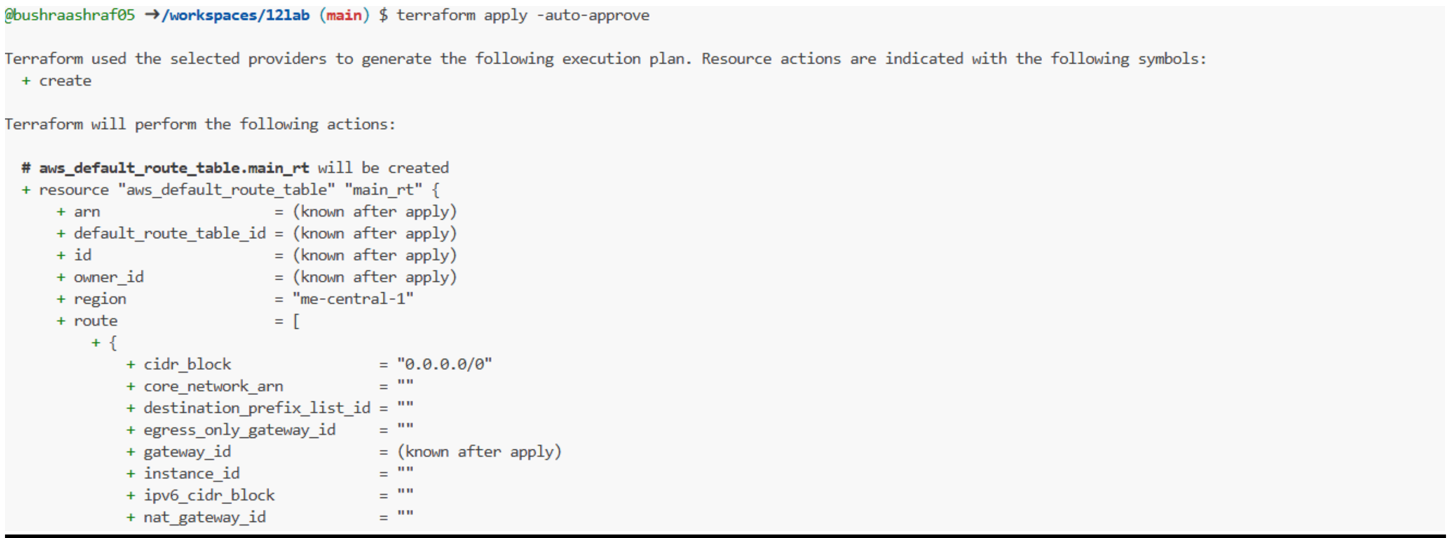
```
GNU nano 7.2                                            main.tf *
    destination = "/home/ec2-user/entry-script-on-ec2.sh"
  }

  # ⚙️Run script on EC2
  provisioner "remote-exec" {
    inline = [
      "sudo chmod +x /home/ec2-user/entry-script-on-ec2.sh",
      "sudo /home/ec2-user/entry-script-on-ec2.sh"
    ]
  }

  # 💻Log locally
  provisioner "local-exec" {
    command = "echo Instance ${self.id} with public IP ${self.public_ip} has been created"
  }

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

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         = aws_subnet.myapp_subnet_1.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
```



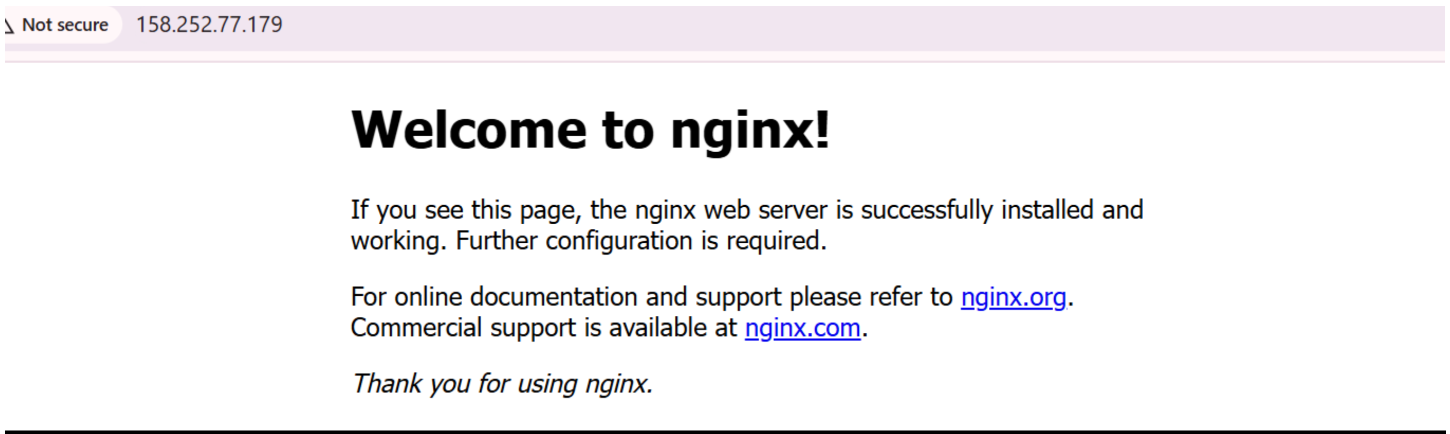
```
GNU nano 7.2                                            outputs.tf *
output "webserver_public_ip" {
  value = module.myapp-webserver.aws_instance.public_ip
}
```



```
● @bushraashraf05 ➜ /workspaces/12lab (main) $ terraform init

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

  Initializing provider plugins...
  - Reusing previous version of hashicorp/aws from the dependency lock file
  - 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.
○ @bushraashraf05 ➜ /workspaces/12lab (main) $
```



```
@bushraashraf05 ➜/workspaces/121ab (main) $ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_default_route_table.main_rt will be created
  + resource "aws_default_route_table" "main_rt" {
      + arn                    = (known after apply)
      + default_route_table_id = (known after apply)
      + id                     = (known after apply)
      + owner_id               = (known after apply)
      + region                 = "me-central-1"
      + route                  = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = (known after apply)
              + instance_id                = ""
              + ipv6_cidr_block            = ""
              + nat_gateway_id             = ""
```



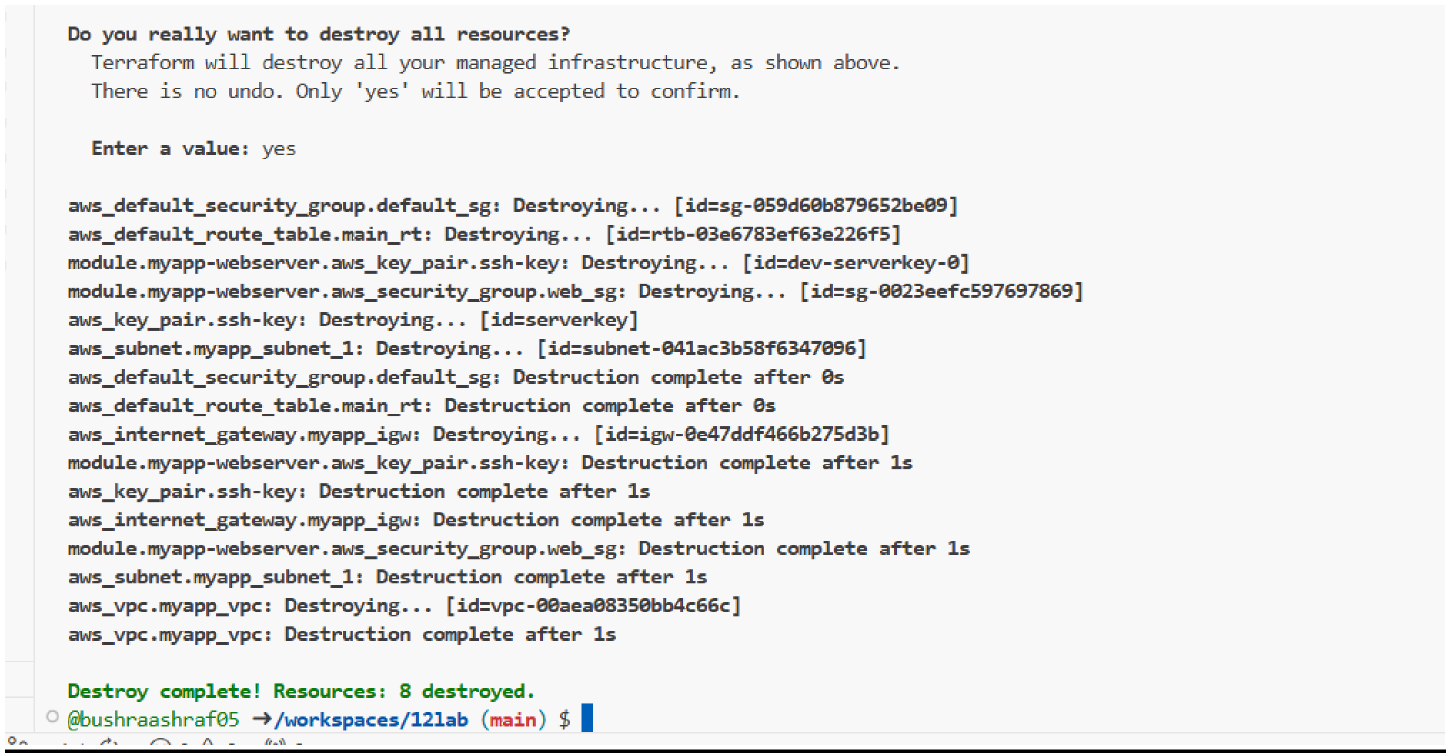
Not secure 158.252.77.179

Welcome to nginx!

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

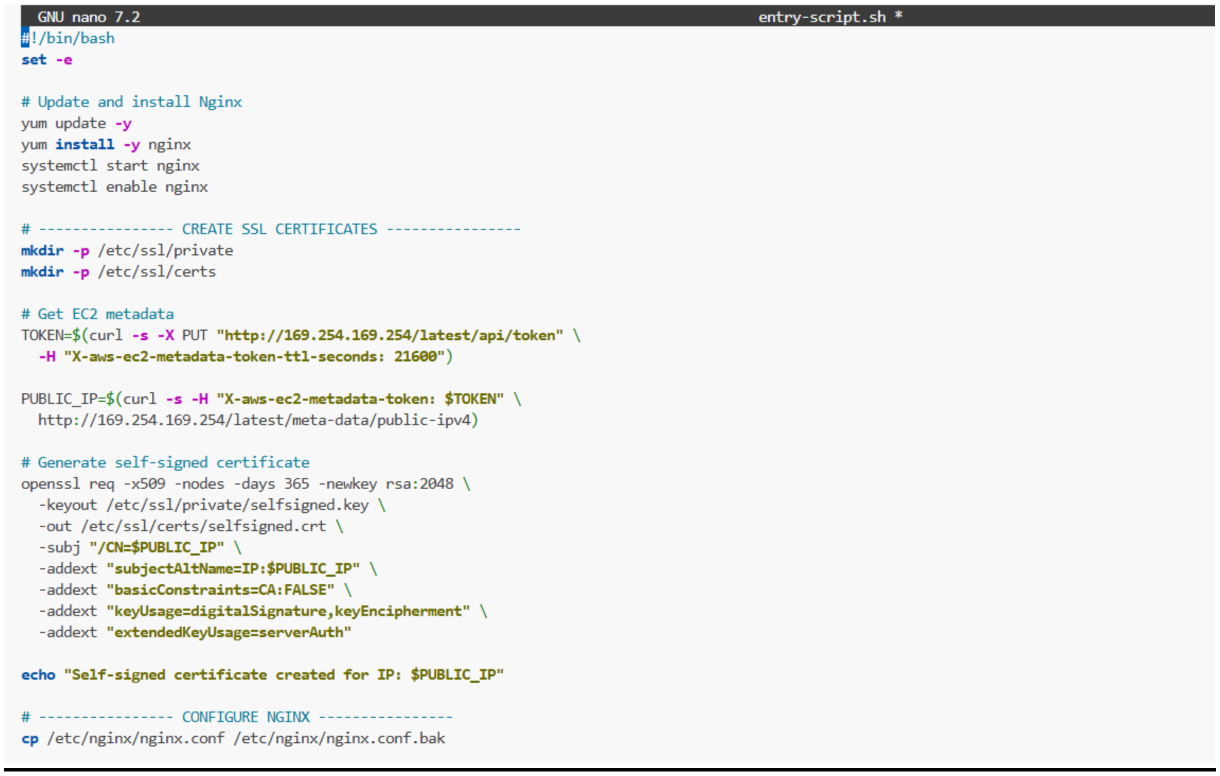
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

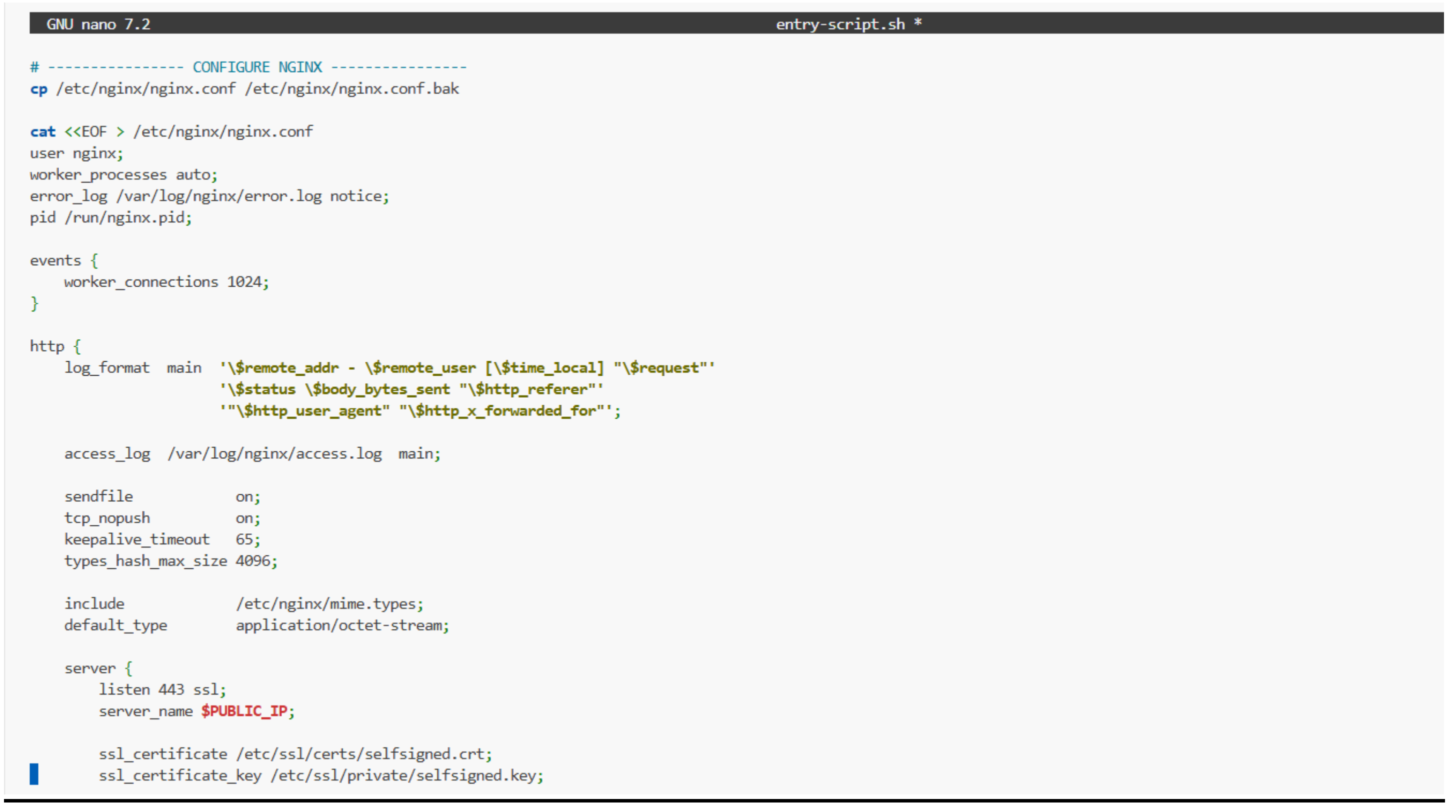


```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_default_security_group.default_sg: Destroying... [id=sg-059d60b879652be09]
aws_default_route_table.main_rt: Destroying... [id=rtb-03e6783ef63e226f5]
module.myapp-webserver.aws_key_pair.ssh-key: Destroying... [id=dev-serverkey-0]
module.myapp-webserver.aws_security_group.web_sg: Destroying... [id=sg-0023eefc597697869]
aws_key_pair.ssh-key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-041ac3b58f6347096]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_default_route_table.main_rt: Destruction complete after 0s
aws_internet_gateway.myapp_igw: Destroying... [id=igw-0e47ddf466b275d3b]
module.myapp-webserver.aws_key_pair.ssh-key: Destruction complete after 1s
aws_key_pair.ssh-key: Destruction complete after 1s
aws_internet_gateway.myapp_igw: Destruction complete after 1s
module.myapp-webserver.aws_security_group.web_sg: Destruction complete after 1s
aws_subnet.myapp_subnet_1: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-00aea08350bb4c66c]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 8 destroyed.
@bushraashraf05 ➜/workspaces/121ab (main) $
```

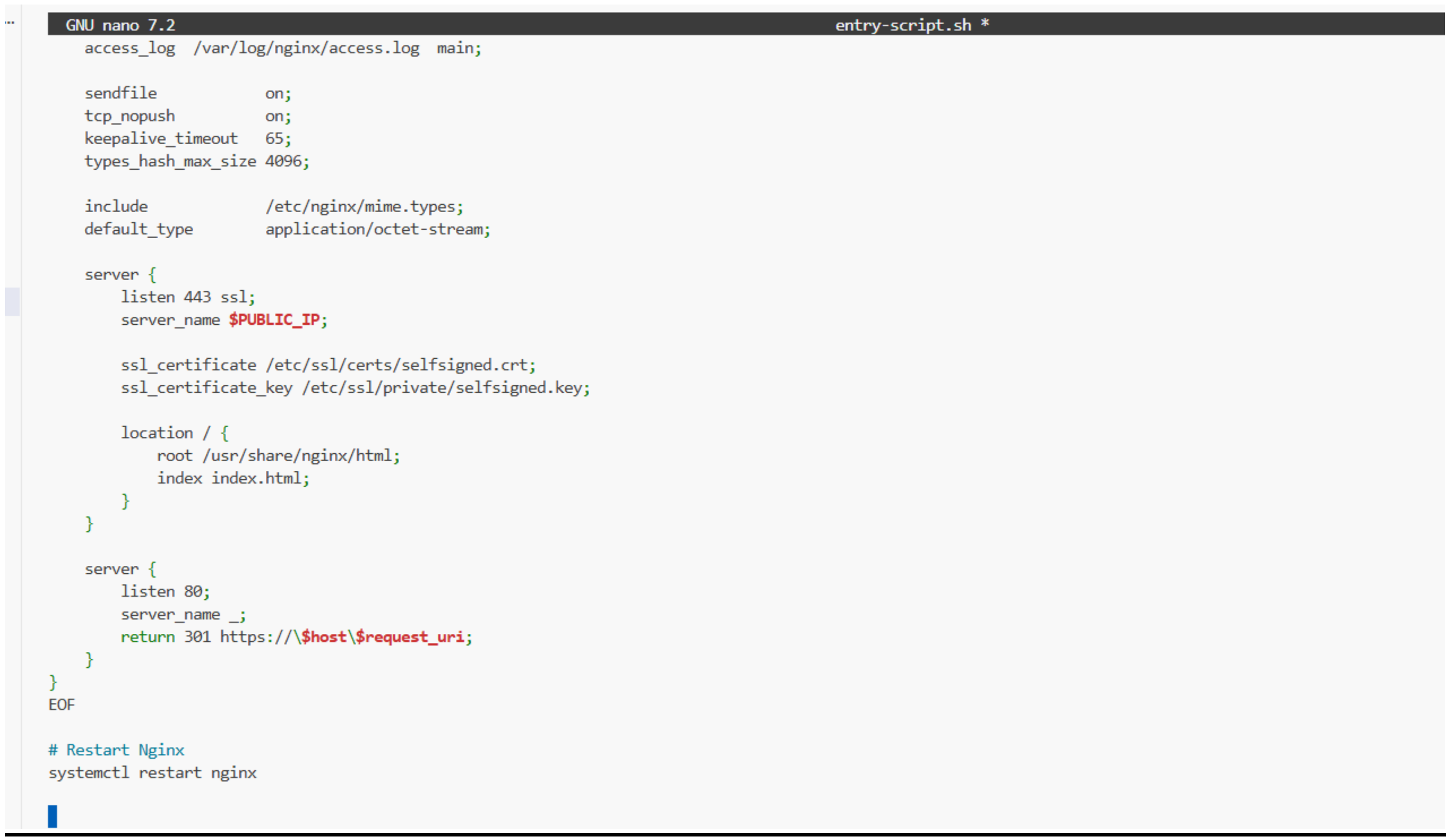
Task 6 — Configure HTTPS with self-signed certificates



```
GNU nano 7.2                                                    entry-script.sh *
#!/bin/bash
set -e

# Update and install Nginx
yum update -y
yum install -y nginx
systemctl start nginx
systemctl enable nginx

# ---------------- CREATE SSL CERTIFICATES ----------------
mkdir -p /etc/ssl/private
mkdir -p /etc/ssl/certs

# Get EC2 metadata
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

PUBLIC_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/public-ipv4)

# Generate self-signed certificate
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/ssl/private/selfsigned.key \
  -out /etc/ssl/certs/selfsigned.crt \
  -subj "/CN=$PUBLIC_IP" \
  -addext "subjectAltName=IP:$PUBLIC_IP" \
  -addext "basicConstraints=CA:FALSE" \
  -addext "keyUsage=digitalSignature,keyEncipherment" \
  -addext "extendedKeyUsage=serverAuth"

echo "Self-signed certificate created for IP: $PUBLIC_IP"

# ---------------- CONFIGURE NGINX ----------------
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak
```

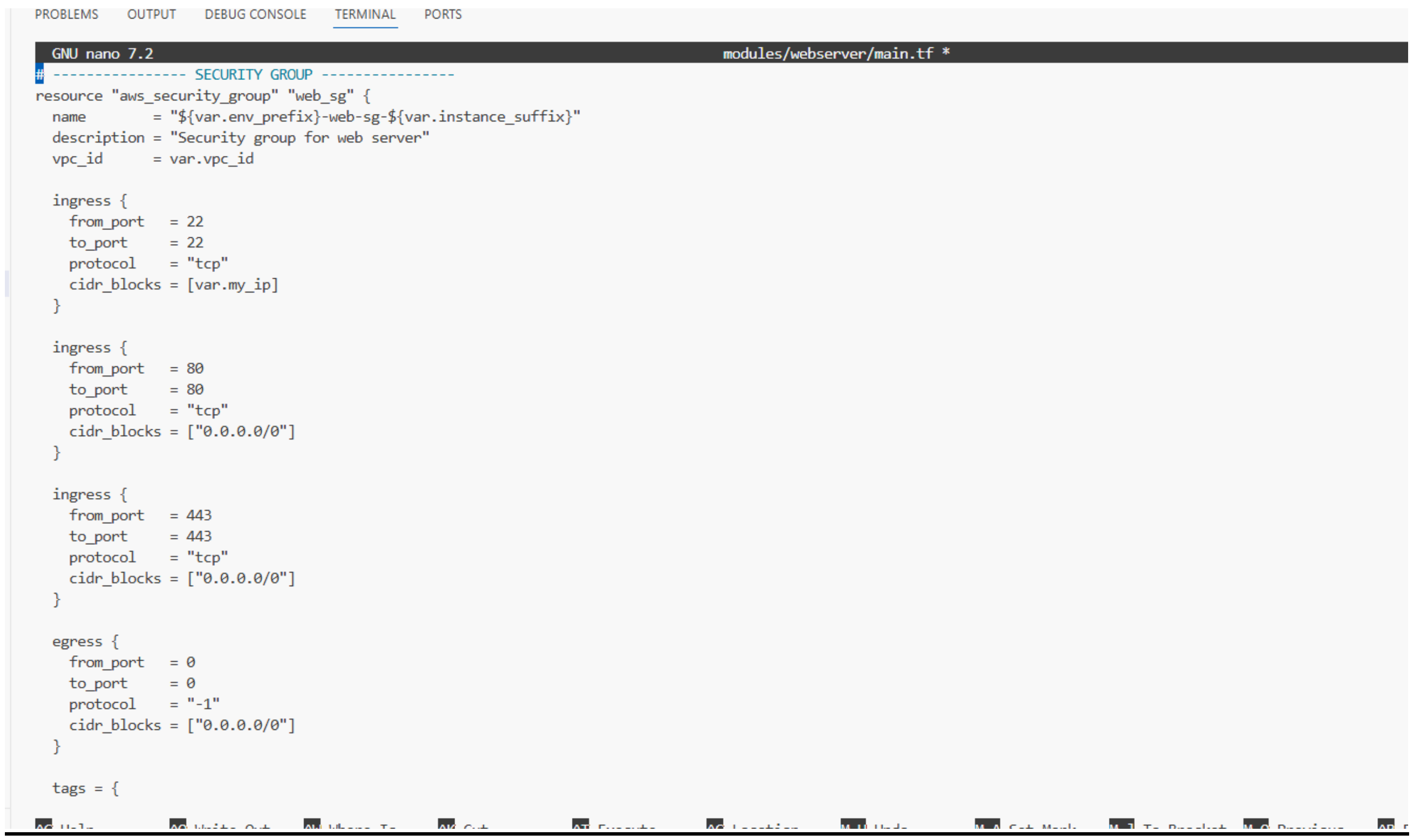


```
GNU nano 7.2                                                    entry-script.sh *

# ---------------- CONFIGURE NGINX ----------------
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak

cat <<EOF > /etc/nginx/nginx.conf
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    log_format  main  '\$remote_addr - \$remote_user [\$time_local] "\$request"'
                      '\$status \$body_bytes_sent "\$http_referer"'
                      '"\$http_user_agent" "\$http_x_forwarded_for"';

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

    sendfile            on;
    tcp_nopush          on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

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

    server {
        listen 443 ssl;
        server_name $PUBLIC_IP;

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

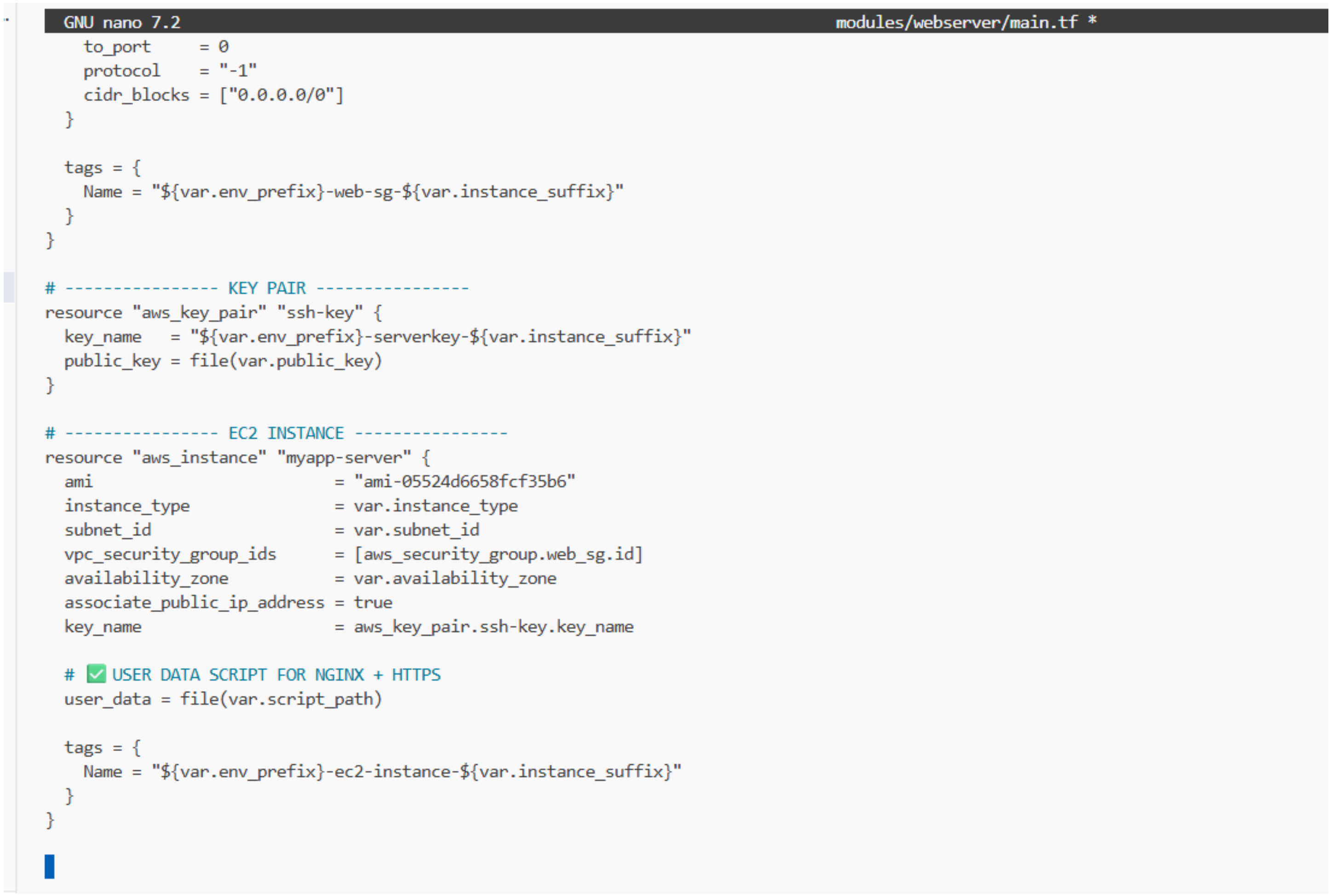


```
    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

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

    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

# Restart Nginx
systemctl restart nginx
```

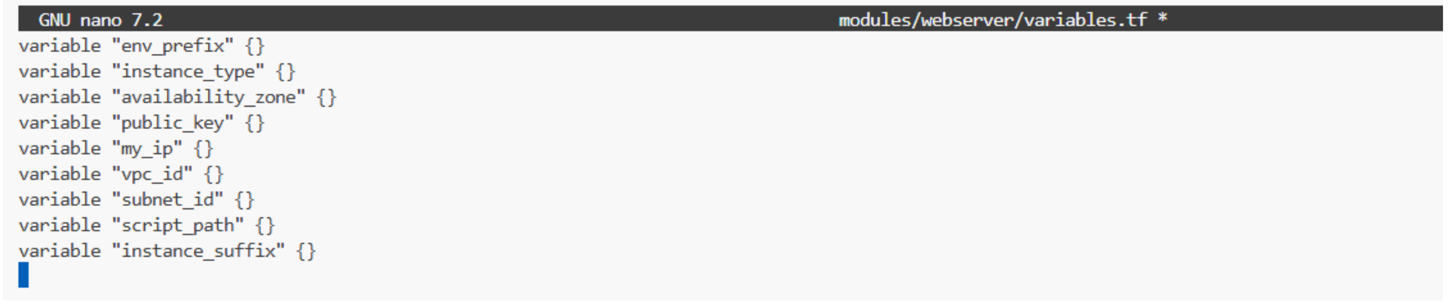
Edit main.tf:



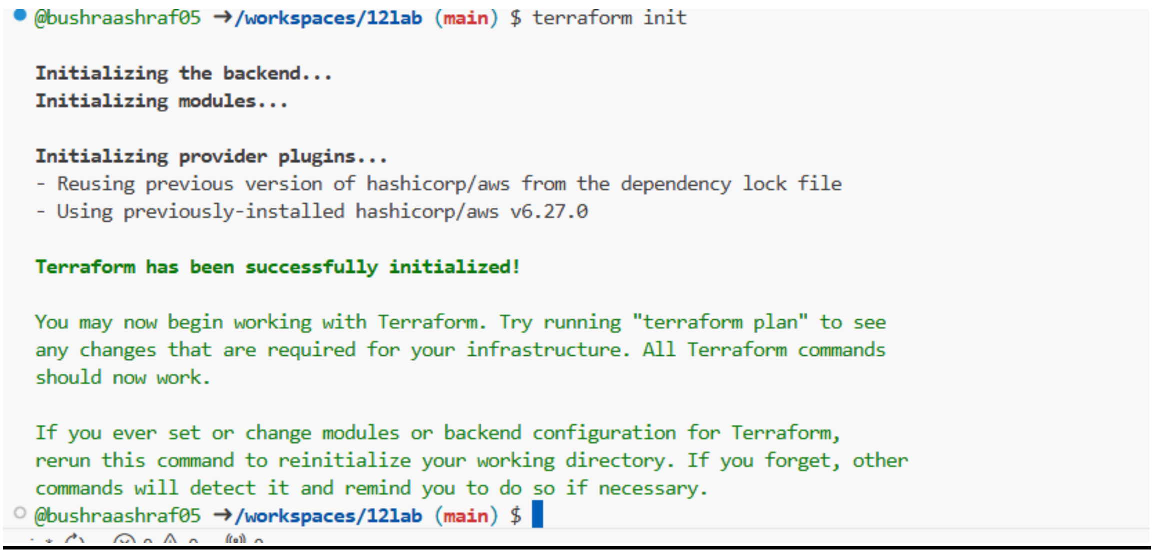
```
# --------------- SECURITY GROUP ---------------
resource "aws_security_group" "web_sg" {
  name        = "${var.env_prefix}-web-sg-${var.instance_suffix}"
  description = "Security group for web server"
  vpc_id      = var.vpc_id

  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
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
```



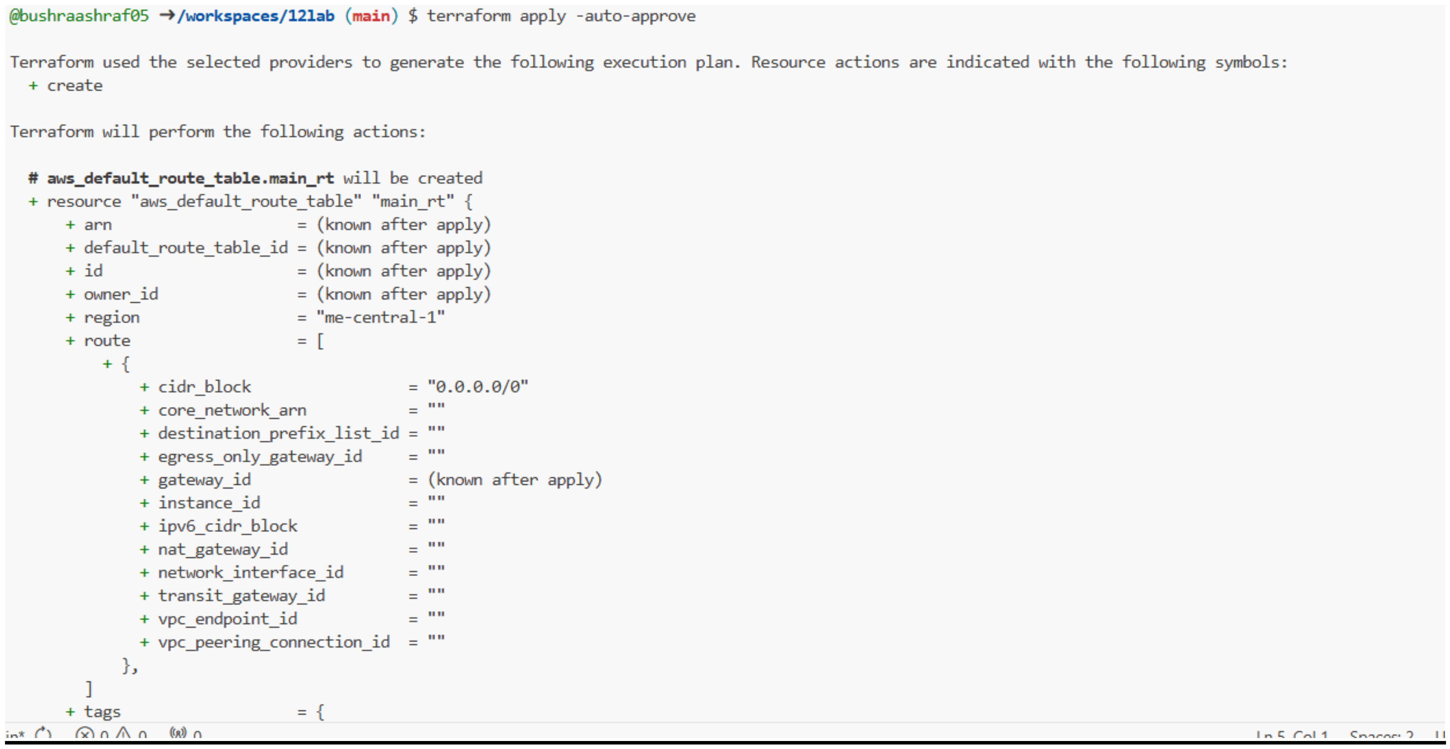
```
GNU nano 7.2                                                    modules/webserver/main.tf *
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.env_prefix}-web-sg-${var.instance_suffix}"
  }
}

# ---------------- KEY PAIR ----------------
resource "aws_key_pair" "ssh-key" {
  key_name   = "${var.env_prefix}-serverkey-${var.instance_suffix}"
  public_key = file(var.public_key)
}

# ---------------- EC2 INSTANCE ----------------
resource "aws_instance" "myapp-server" {
  ami                         = "ami-05524d6658fcf35b6"
  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 SCRIPT FOR NGINX + HTTPS
  user_data = file(var.script_path)

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



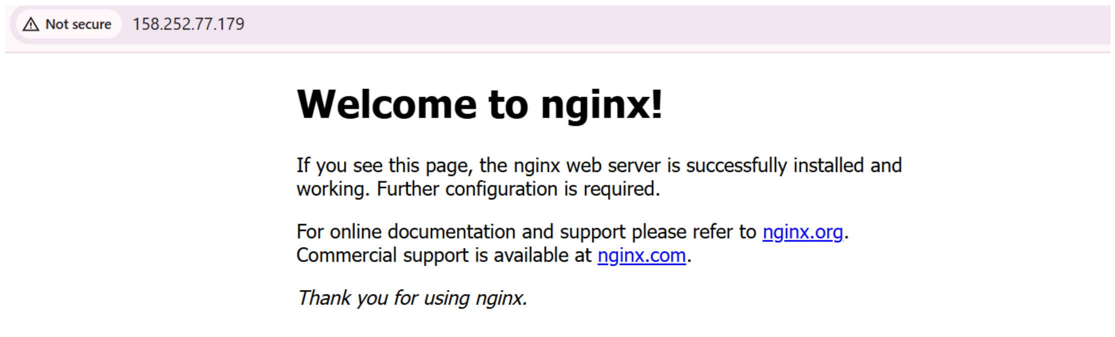
```
GNU nano 7.2                                          modules/webserver/variables.tf *
variable "env_prefix" {}
variable "instance_type" {}
variable "availability_zone" {}
variable "public_key" {}
variable "my_ip" {}
variable "vpc_id" {}
variable "subnet_id" {}
variable "script_path" {}
variable "instance_suffix" {}
```



```
@bushraashraf05 →/workspaces/12lab (main) $ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- 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.
@bushraashraf05 →/workspaces/12lab (main) $
```



```
@bushraashraf05 ➜/workspaces/12lab (main) $ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_default_route_table.main_rt will be created
  + resource "aws_default_route_table" "main_rt" {
      + arn                    = (known after apply)
      + default_route_table_id = (known after apply)
      + id                     = (known after apply)
      + owner_id               = (known after apply)
      + region                 = "me-central-1"
      + route                  = [
          + {
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = (known after apply)
              + instance_id                = ""
              + ipv6_cidr_block            = ""
              + nat_gateway_id             = ""
              + network_interface_id       = ""
              + transit_gateway_id         = ""
              + vpc_endpoint_id            = ""
              + vpc_peering_connection_id  = ""
            },
        ]
      + tags                   = {
```



Not secure 158.252.77.179

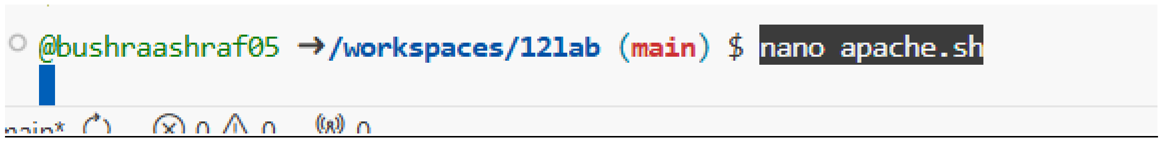
Welcome to nginx!

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

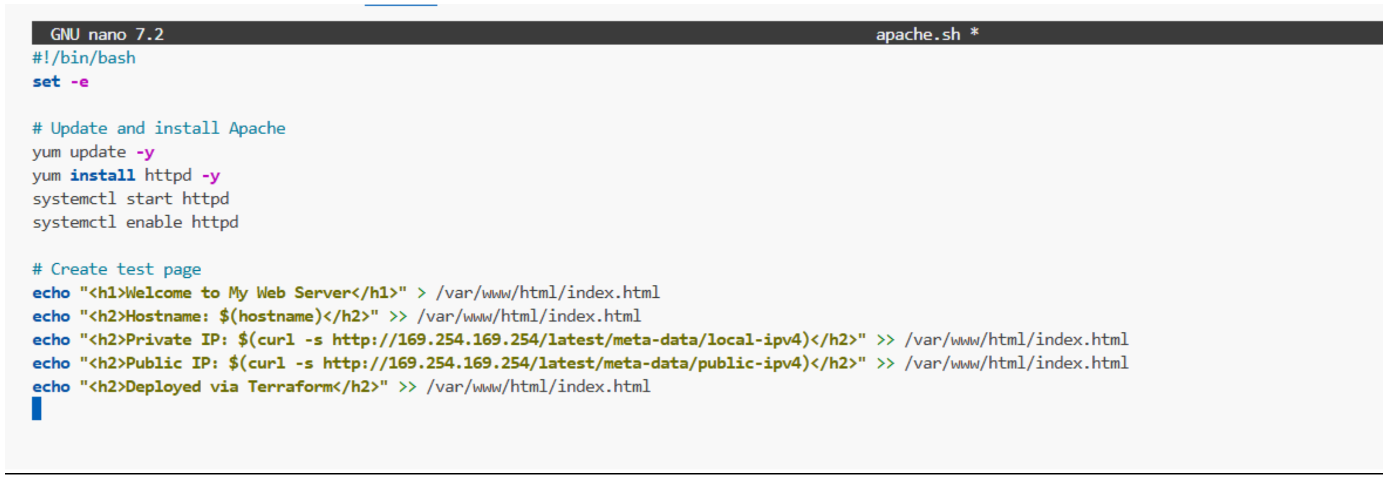
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Task 7 — Configure Nginx as reverse proxy

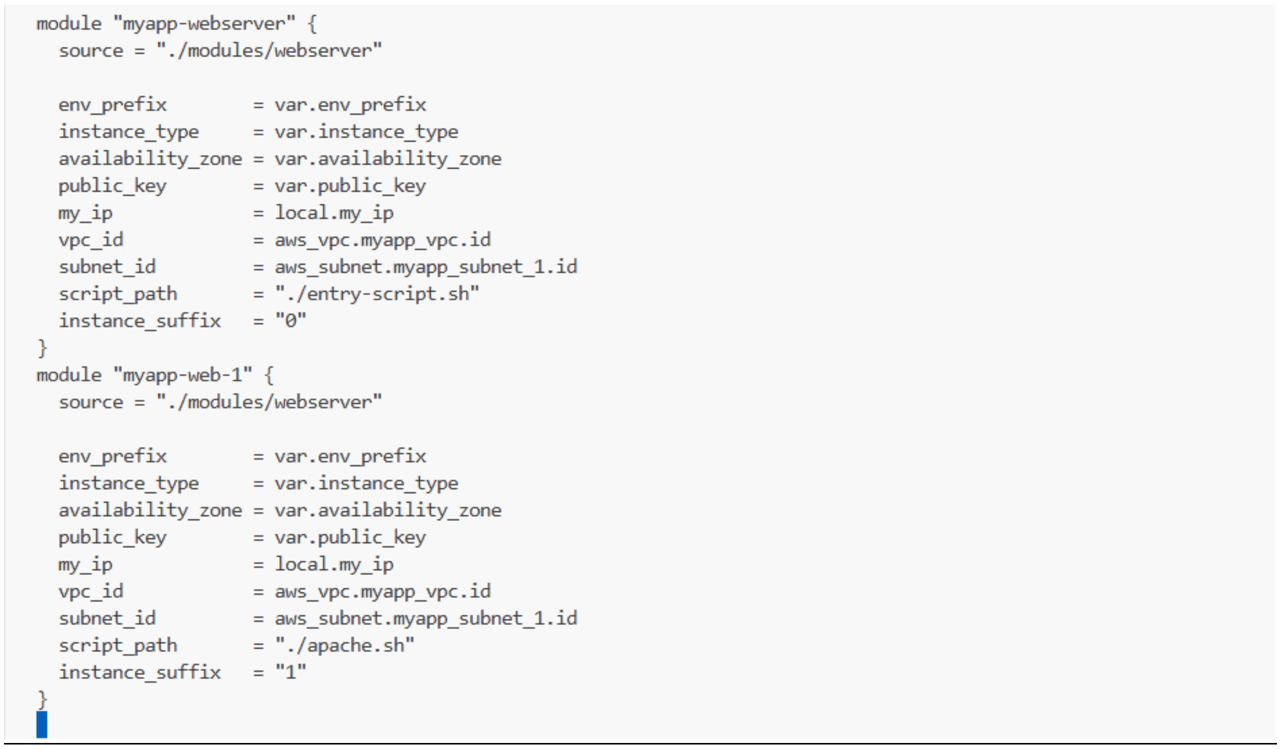


```
@bushraashraf05 ➜/workspaces/12lab (main) $ nano apache.sh
```

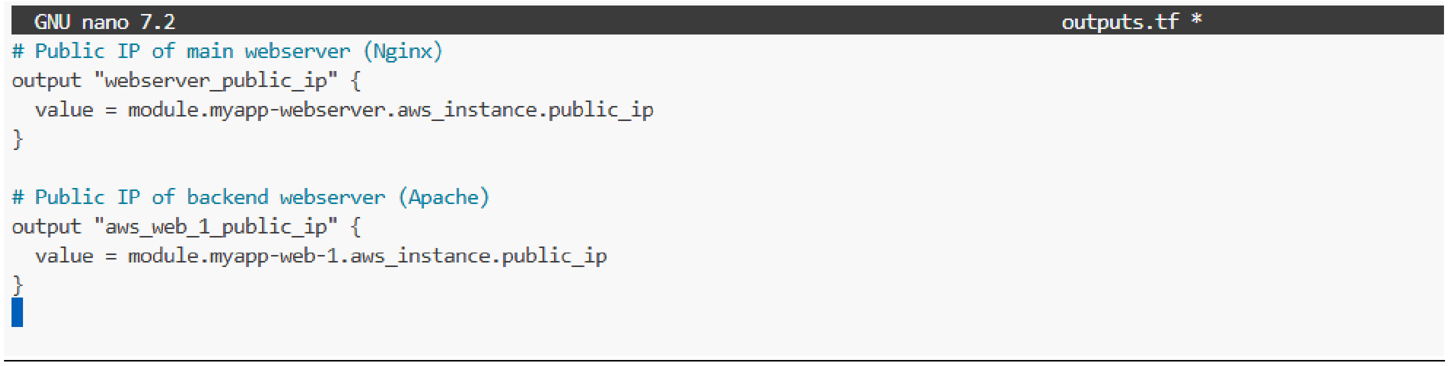


```
GNU nano 7.2                                        apache.sh *
#!/bin/bash
set -e

# Update and install Apache
yum update -y
yum install httpd -y
systemctl start httpd
systemctl enable httpd

# Create test page
echo "<h1>Welcome to My Web Server</h1>" > /var/www/html/index.html
echo "<h2>Hostname: $(hostname)</h2>" >> /var/www/html/index.html
echo "<h2>Private IP: $(curl -s http://169.254.169.254/latest/meta-data/local-ipv4)</h2>" >> /var/www/html/index.html
echo "<h2>Public IP: $(curl -s http://169.254.169.254/latest/meta-data/public-ipv4)</h2>" >> /var/www/html/index.html
echo "<h2>Deployed via Terraform</h2>" >> /var/www/html/index.html
```

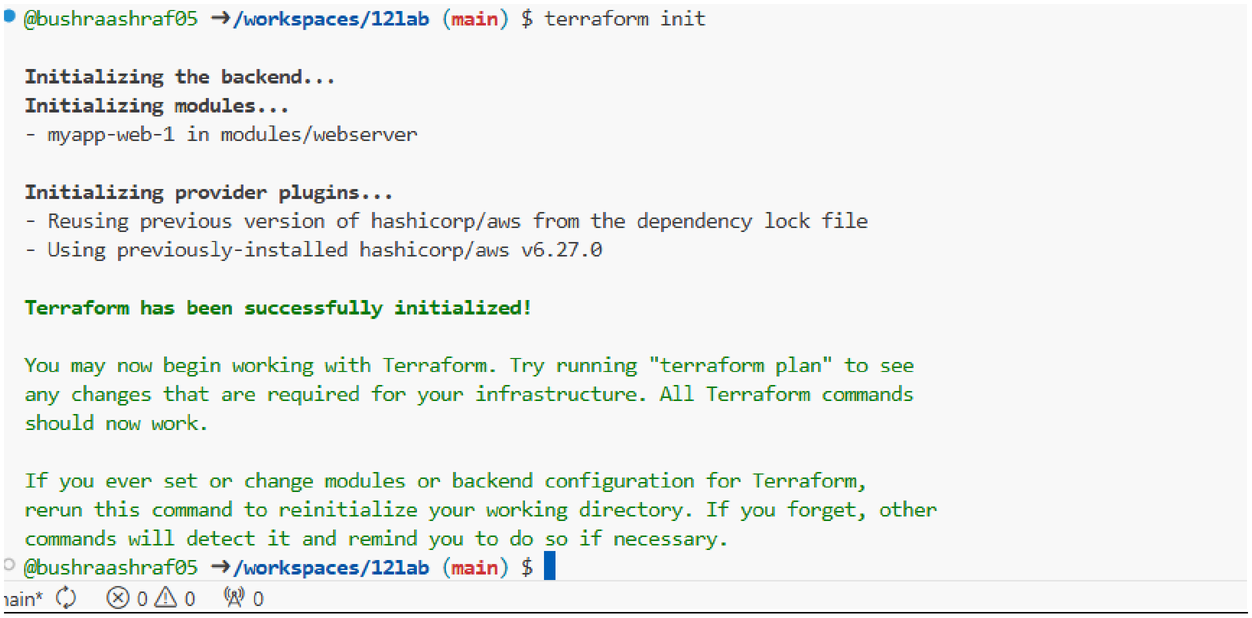
Add this at the end of main.tf:



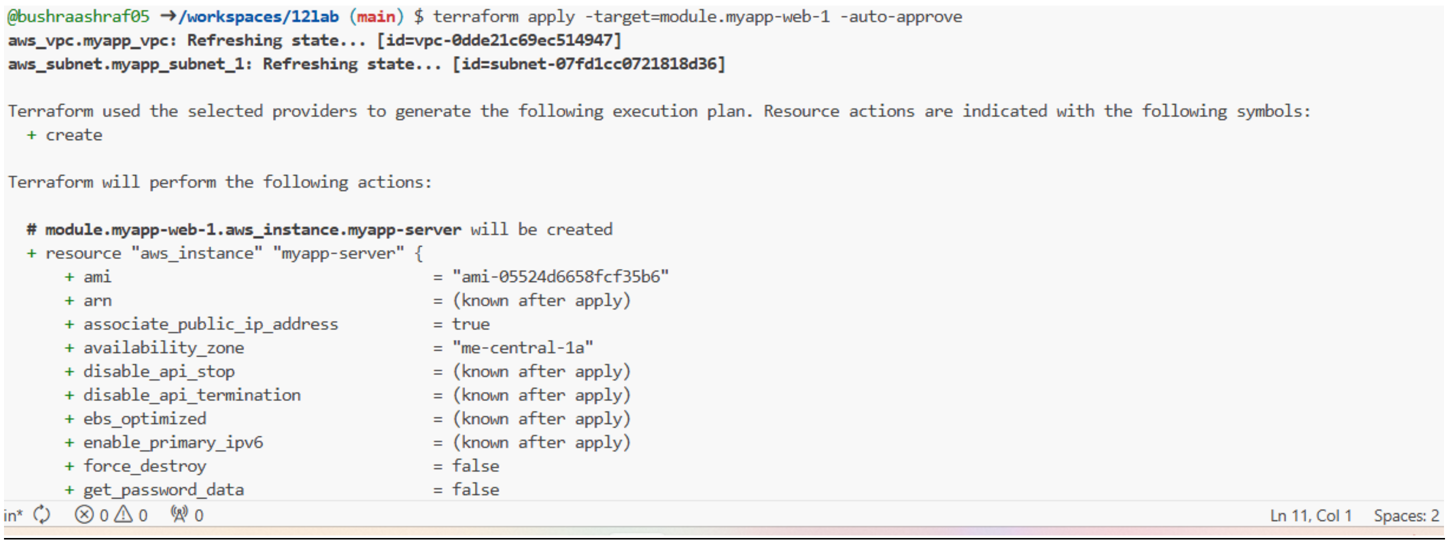
```
module "myapp-webserver" {
  source = "./modules/webserver"

  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         = aws_subnet.myapp_subnet_1.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
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         = aws_subnet.myapp_subnet_1.id
  script_path       = "./apache.sh"
  instance_suffix   = "1"
}
```

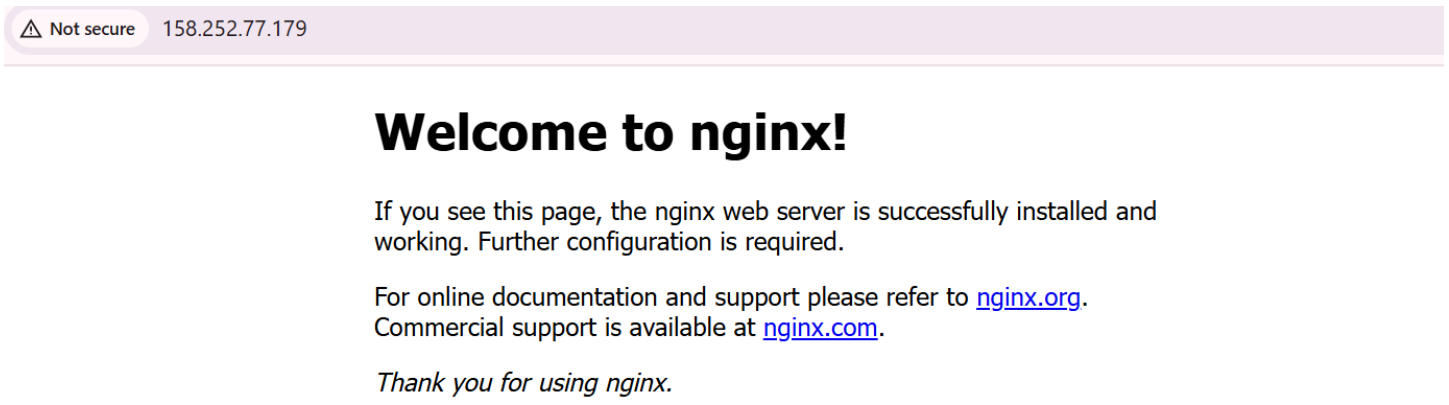


```
 GNU nano 7.2                                                    outputs.tf *
# Public IP of main webserver (Nginx)
output "webserver_public_ip" {
  value = module.myapp-webserver.aws_instance.public_ip
}

# Public IP of backend webserver (Apache)
output "aws_web_1_public_ip" {
  value = module.myapp-web-1.aws_instance.public_ip
}
```



```
@bushraashraf05 ➜ /workspaces/12lab (main) $ terraform init

 Initializing the backend...
 Initializing modules...
 - myapp-web-1 in modules/webserver

 Initializing provider plugins...
 - Reusing previous version of hashicorp/aws from the dependency lock file
 - 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.
@bushraashraf05 ➜ /workspaces/12lab (main) $
ain*   ⊗ 0 ⚠ 0   0
```


```
@bushraashraf05 →/workspaces/121ab (main) $ terraform apply -target=module.myapp-web-1 -auto-approve
aws_vpc.myapp_vpc: Refreshing state... [id=vpc-0dde21c69ec514947]
aws_subnet.myapp_subnet_1: Refreshing state... [id=subnet-07fd1cc0721818d36]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # module.myapp-web-1.aws_instance.myapp-server will be created
  + resource "aws_instance" "myapp-server" {
      + ami                                  = "ami-05524d6658fcf35b6"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = "me-central-1a"
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
```



Welcome to nginx!

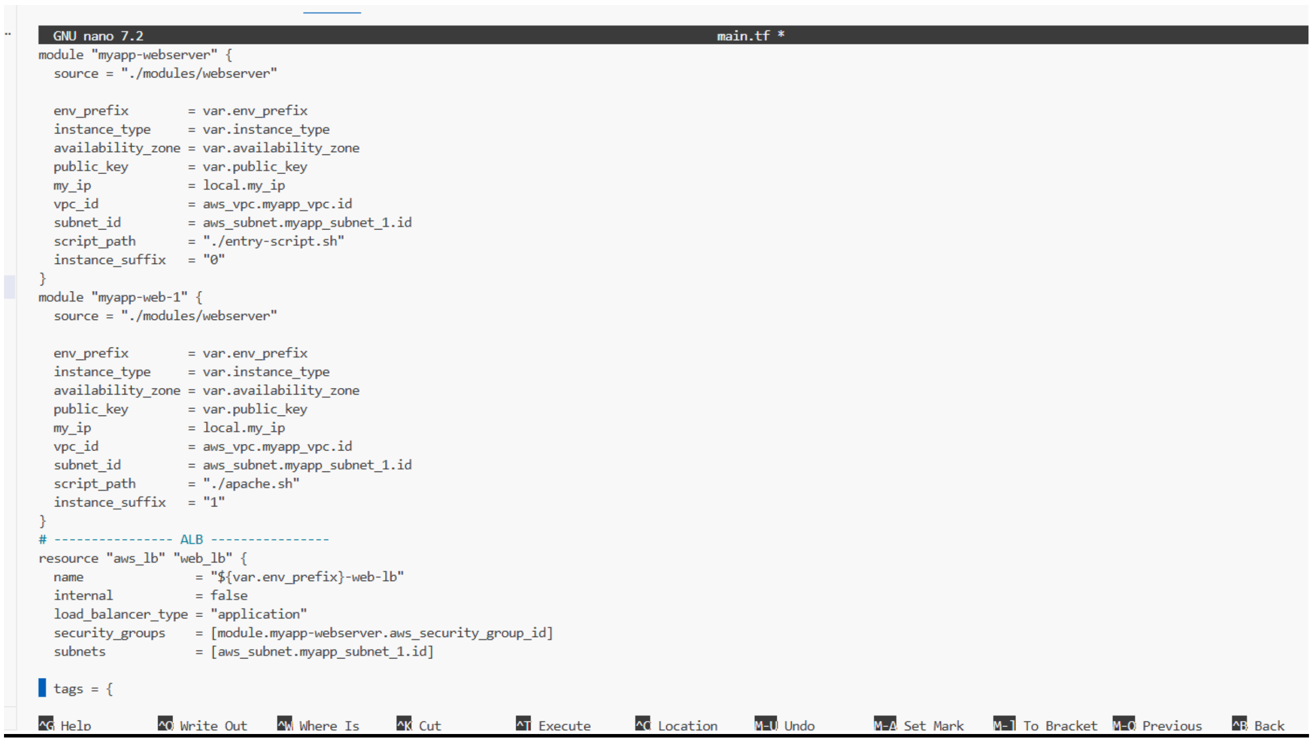
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

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

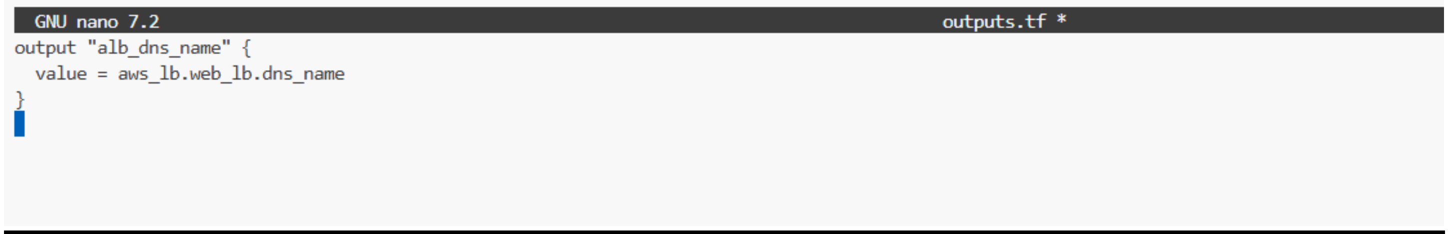
Thank you for using nginx.

Task 8 — Configure Nginx as load balancer:

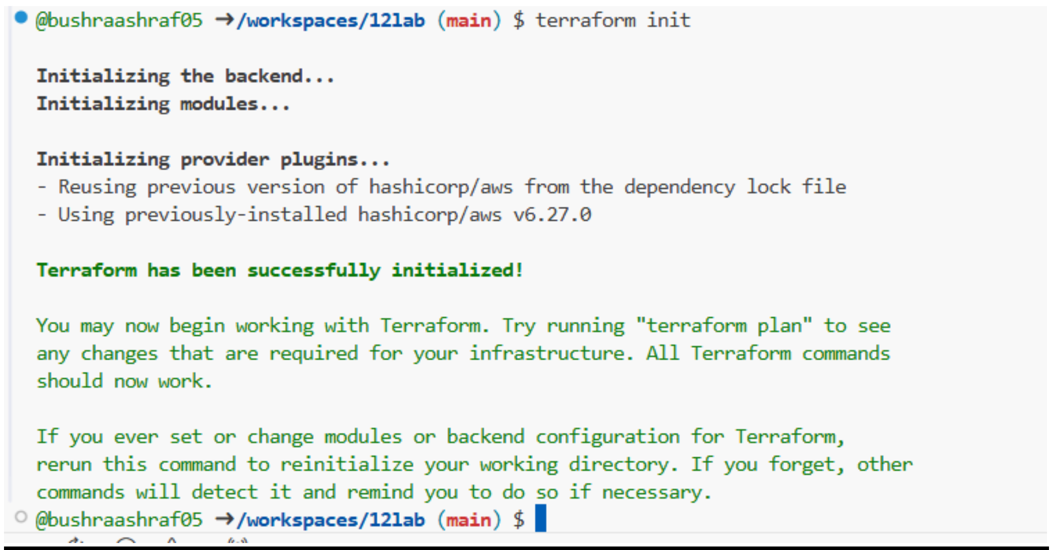
Edit main.tf:



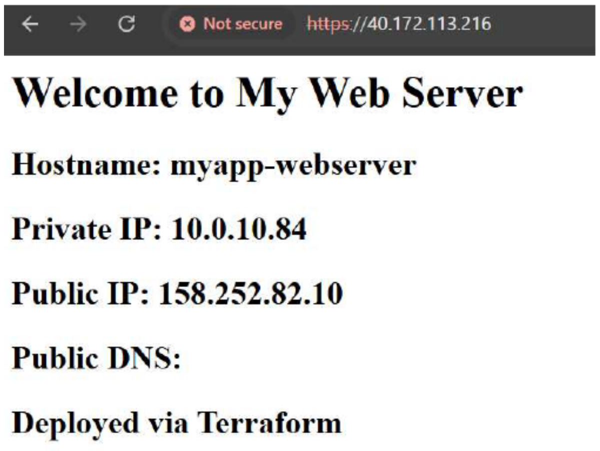
```
module "myapp-webserver" {
  source = "./modules/webserver"

  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         = aws_subnet.myapp_subnet_1.id
  script_path       = "./entry-script.sh"
  instance_suffix   = "0"
}
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         = aws_subnet.myapp_subnet_1.id
  script_path       = "./apache.sh"
  instance_suffix   = "1"
}
# ---------------- ALB ----------------
resource "aws_lb" "web_lb" {
  name               = "${var.env_prefix}-web-lb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [module.myapp-webserver.aws_security_group_id]
  subnets            = [aws_subnet.myapp_subnet_1.id]

  tags = {
```



```
GNU nano 7.2                                                    outputs.tf *
output "alb_dns_name" {
  value = aws_lb.web_lb.dns_name
}
```



```
@bushraashraf05 →/workspaces/12lab (main) $ terraform init

Initializing the backend...
Initializing modules...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- 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.
@bushraashraf05 →/workspaces/12lab (main) $
```



Not secure https://40.172.113.216

Welcome to My Web Server

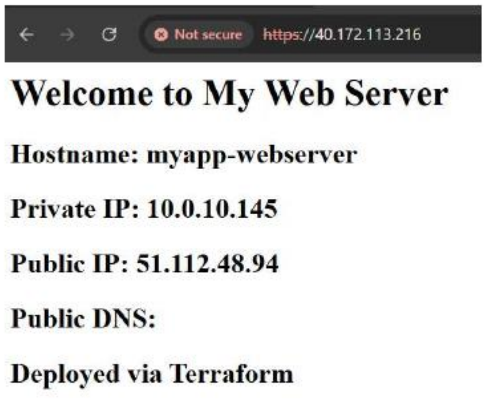
Hostname: myapp-webserver

Private IP: 10.0.10.84

Public IP: 158.252.82.10

Public DNS:

Deployed via Terraform



Not secure https://40.172.113.216

Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.145

Public IP: 51.112.48.94

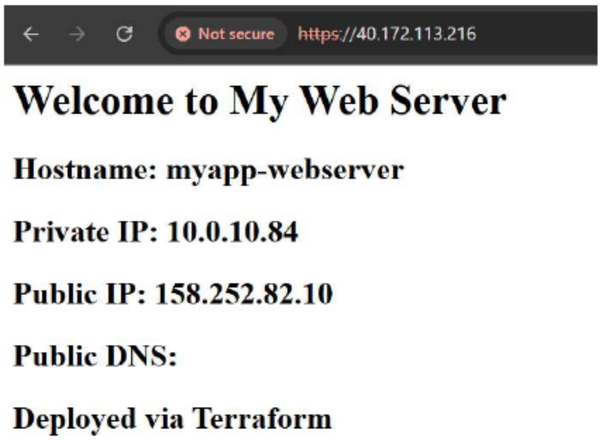
Public DNS:

Deployed via Terraform

TASK 9 — Auto Scaling Group:

Edit main.tf:

```
# ---------------- Launch Template ----------------
resource "aws_launch_template" "web_lt" {
  name_prefix   = "${var.env_prefix}-lt"
  image_id      = data.aws_ami.amazon_linux.id
  instance_type = var.instance_type
  key_name      = module.myapp-webserver.aws_key_pair_name

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

# ---------------- Auto Scaling Group ----------------
resource "aws_autoscaling_group" "web_asg" {
  name                 = "${var.env_prefix}-web-asg"
  max_size              = 3
  min_size              = 1
  desired_capacity      = 2
  vpc_zone_identifier   = [aws_subnet.myapp_subnet_1.id]
  health_check_type     = "ELB"
  launch_template {
    id      = aws_launch_template.web_lt.id
    version = "$Latest"
  }

  target_group_arns = [aws_lb_target_group.web_tg.arn]

  tag {
    key                 = "Name"
    value               = "${var.env_prefix}-asg-instance"
    propagate_at_launch = true
  }
}
```



Not secure https://40.172.113.216

Welcome to My Web Server

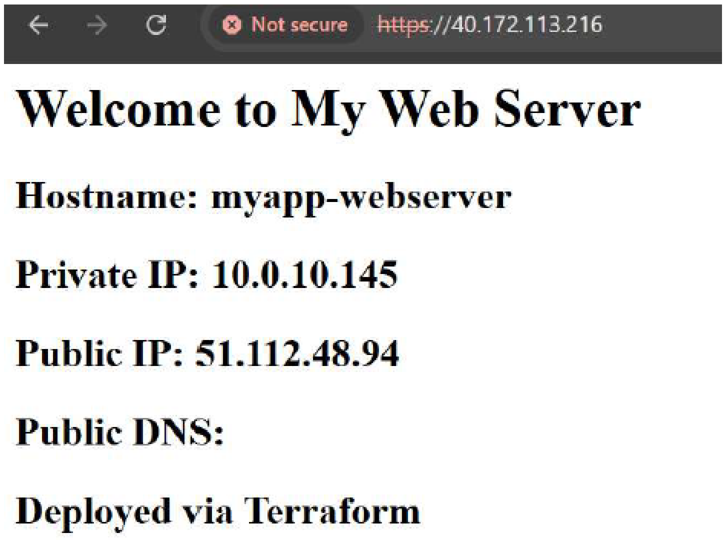
Hostname: myapp-webserver

Private IP: 10.0.10.84

Public IP: 158.252.82.10

Public DNS:

Deployed via Terraform



Not secure https://40.172.113.216

Welcome to My Web Server

Hostname: myapp-webserver

Private IP: 10.0.10.145

Public IP: 51.112.48.94

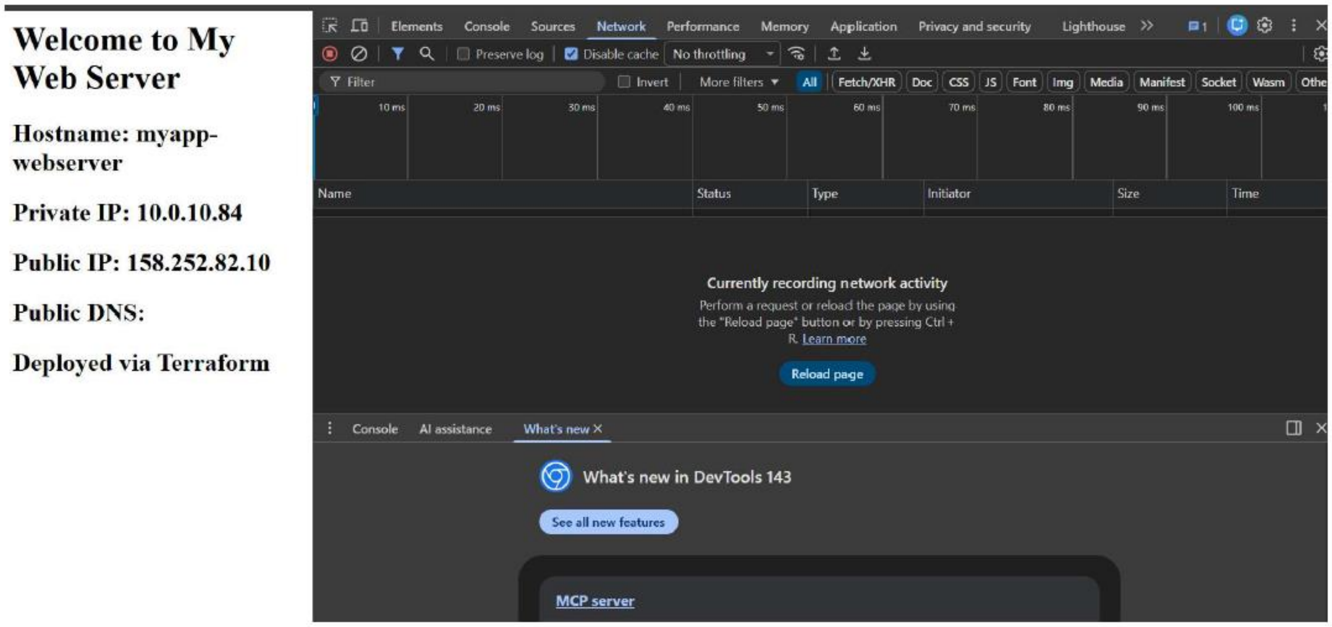
Public DNS:

Deployed via Terraform

TASK 10 — CloudWatch Monitoring:

Edit main.tf:

```
# ---------------- CloudWatch CPU Alarm ----------------
resource "aws_cloudwatch_metric_alarm" "cpu_high" {
  alarm_name          = "${var.env_prefix}-cpu-high"
  comparison_operator = "GreaterThanThreshold"
  evaluation_periods  = 2
  metric_name         = "CPUUtilization"
  namespace           = "AWS/EC2"
  period              = 60
  statistic           = "Average"
  threshold           = 70


  alarm_description   = "This metric monitors EC2 CPU utilization"
  dimensions = {
    InstanceId = module.myapp-webserver.aws_instance.id
  }
}
```



Welcome to My Web Server

Hostname: myapp-webserver

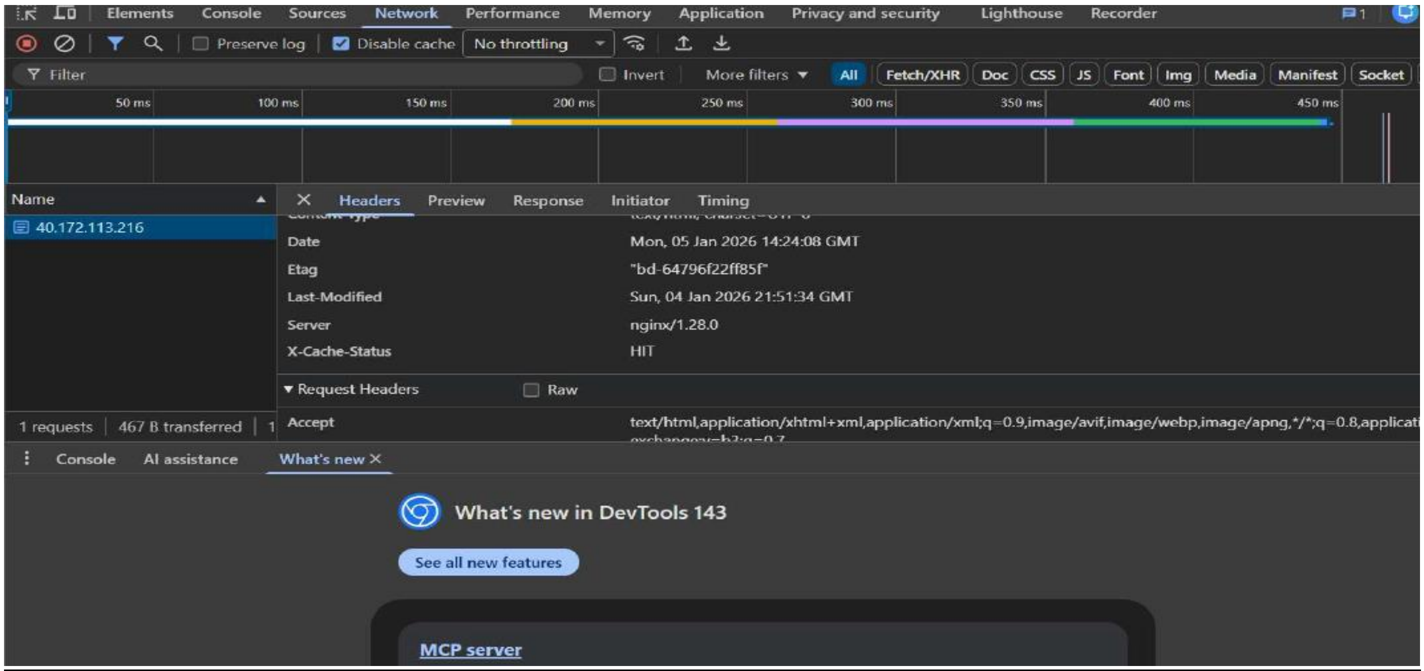
Private IP: 10.0.10.84

Public IP: 158.252.82.10

Public DNS:

Deployed via Terraform

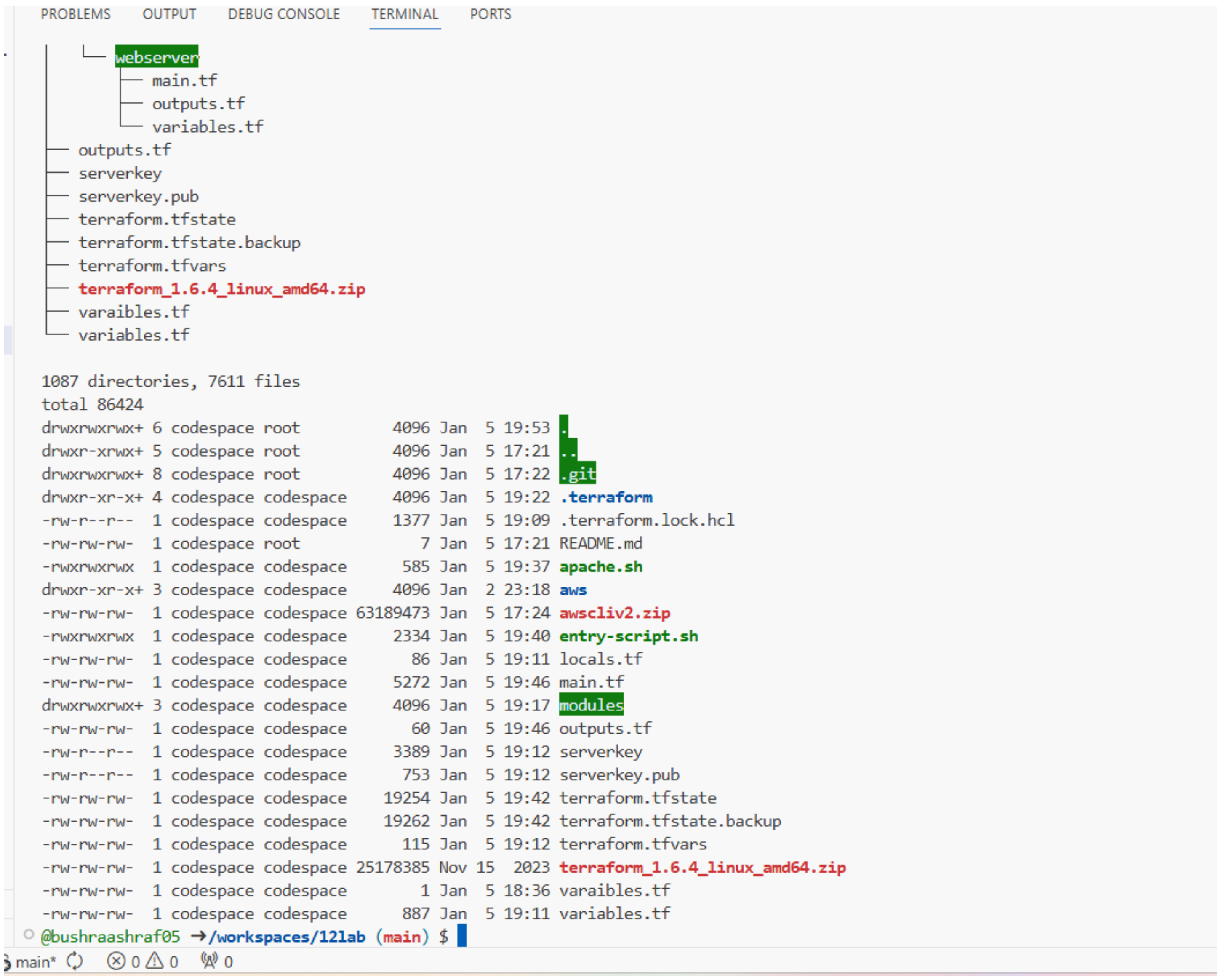
Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket Wasm Othe
Name Status Type Initiator Size Time
Currently recording network activity
Perform a request or reload the page by using the "Reload page" button or by pressing Ctrl + R. Learn more
Reload page
Console AI assistance What's new
What's new in DevTools 143
See all new features
MCP server



Elements Console Sources Network Performance Memory Application Privacy and security Lighthouse Recorder
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All Fetch/XHR Doc CSS JS Font Img Media Manifest Socket
Name
Headers Preview Response Initiator Timing

```
40.172.113.216

Date              Mon, 05 Jan 2026 14:24:08 GMT
Etag              "bd-64796f22ff85f"
Last-Modified     Sun, 04 Jan 2026 21:51:34 GMT
Server            nginx/1.28.0
X-Cache-Status    HIT

▼ Request Headers     Raw
Accept            text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati

1 requests | 467 B transferred | 1
```

Console AI assistance What's new
What's new in DevTools 143
See all new features
MCP server



```
│   └── webserver
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── serverkey
├── serverkey.pub
├── terraform.tfstate
├── terraform.tfstate.backup
├── terraform.tfvars
├── terraform_1.6.4_linux_amd64.zip
├── varaibles.tf
└── variables.tf

1087 directories, 7611 files
total 86424
drwxrwxrwx+ 6 codespace root          4096 Jan  5 19:53 .
drwxr-xrwx+ 5 codespace root          4096 Jan  5 17:21 ..
drwxrwxrwx+ 8 codespace root          4096 Jan  5 17:22 .git
drwxr-xr-x+ 4 codespace codespace     4096 Jan  5 19:22 .terraform
-rw-r--r--  1 codespace codespace     1377 Jan  5 19:09 .terraform.lock.hcl
-rw-rw-rw-  1 codespace root             7 Jan  5 17:21 README.md
-rwxrwxrwx  1 codespace codespace      585 Jan  5 19:37 apache.sh
drwxr-xr-x+ 3 codespace codespace     4096 Jan  2 23:18 aws
-rw-rw-rw-  1 codespace codespace 63189473 Jan  5 17:24 awscliv2.zip
-rwxrwxrwx  1 codespace codespace     2334 Jan  5 19:40 entry-script.sh
-rw-rw-rw-  1 codespace codespace       86 Jan  5 19:11 locals.tf
-rw-rw-rw-  1 codespace codespace     5272 Jan  5 19:46 main.tf
drwxrwxrwx+ 3 codespace codespace     4096 Jan  5 19:17 modules
-rw-rw-rw-  1 codespace codespace       60 Jan  5 19:46 outputs.tf
-rw-r--r--  1 codespace codespace     3389 Jan  5 19:12 serverkey
-rw-r--r--  1 codespace codespace      753 Jan  5 19:12 serverkey.pub
-rw-rw-rw-  1 codespace codespace    19254 Jan  5 19:42 terraform.tfstate
-rw-rw-rw-  1 codespace codespace    19262 Jan  5 19:42 terraform.tfstate.backup
-rw-rw-rw-  1 codespace codespace      115 Jan  5 19:12 terraform.tfvars
-rw-rw-rw-  1 codespace codespace 25178385 Nov 15  2023 terraform_1.6.4_linux_amd64.zip
-rw-rw-rw-  1 codespace codespace        1 Jan  5 18:36 varaibles.tf
-rw-rw-rw-  1 codespace codespace      887 Jan  5 19:11 variables.tf
@bushraashraf05 ➜ /workspaces/12lab (main) $
```

****