FATIMA JINNAH WOMEN UNIVERSITY
RAWALPINDI

**ASSIGNMENT II**

# Advanced Terraform & Nginx Multi-Tier Architecture

Submitted to

**Sir Waqas Saleem**

**Submitted by:** **Manahil Shujah**

**Registration #:** **2023-BSE-043**

**Semester:** **V**

**Section:** **B**

**Department:** **Software Engineering**

**Course Name:** **Cloud Computing**

**Github Repository:**https://github.com/23-22411-043/cc_ManahilShujah_043_Assignment2

## Contents

**Executive Summary** 3

**Architecture Overview** 3

**Part 1: Infrastructure Setup** 4

- **1.1 Project Structure** 4
- **1.2 Variable Configuration** 5
- **1.3 Networking Module** 6
- **1.4 Security Module** 7
- **1.5 Locals Configuration** 7

**Part 2: Webserver Module** 8

- **2.1 Module Design** 8
- **2.2 Module Usage** 9

**Part 3: Server Configuration Scripts** 10

- **3.1 Apache Backend Server Script** 10
- **3.2 Nginx Server Setup Script** 11

**Part 4: Infrastructure Deployment** 12

- **4.1 Initial Deployment** 12
- **4.2 Output Configuration** 13
- **4.3 AWS Console Verification** 14

**Part 5: Nginx Configuration & Testing** 15

- **5.1 Update Nginx Backend Configuration** 15
- **5.2 Test Load Balancing** 16
- **5.3 Test Cache Functionality** 18
- **5.4 Test High Availability (Backup Server)** 19
- **5.5 Security & Performance Analysis** 20

**Part 6: Documentation & Cleanup** 24

- **6.1 README Documentation** 24
- **6.2 Infrastructure Cleanup** 25

**Common Issues and Solutions** 26

**Conclusion** 26

## Executive Summary

This report covers the implementation of Assignment 2, which focuses on deploying a multi-tier web environment on Amazon Web Services (AWS) using Terraform. The goal is to gain hands-on experience with Infrastructure as Code (IaC), cloud networking, and advanced Nginx configurations.

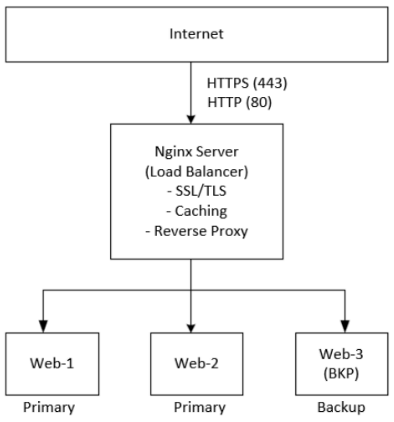
The setup consists of a high-availability web infrastructure on AWS. It includes a single Nginx server functioning as a reverse proxy and load balancer, along with three Apache backend servers. The Nginx server is configured with HTTPS via a self-signed SSL certificate, caching for performance optimization, and load balancing to distribute requests across backend servers. One backend server is designated as a backup to ensure continuity in case primary servers fail.

Terraform modules were utilized to structure the project cleanly, with separate modules for networking, security, and web servers. The networking module sets up the VPC, subnets, internet gateway, and routing. Security groups are defined in the security module to permit only required traffic, following best practices. The webserver module is reused for Nginx and backend servers alike.

Server configuration is automated using shell scripts. Apache backend servers display system and network metadata, while Nginx handles SSL, caching, traffic forwarding, and failover. The deployment is tested for load balancing, caching, backup server functionality, and security compliance.

This assignment reinforced practical skills in Terraform, AWS services, and Nginx configuration, while enhancing understanding of automation, scalability, and high-availability design in cloud systems.

## Architecture Overview

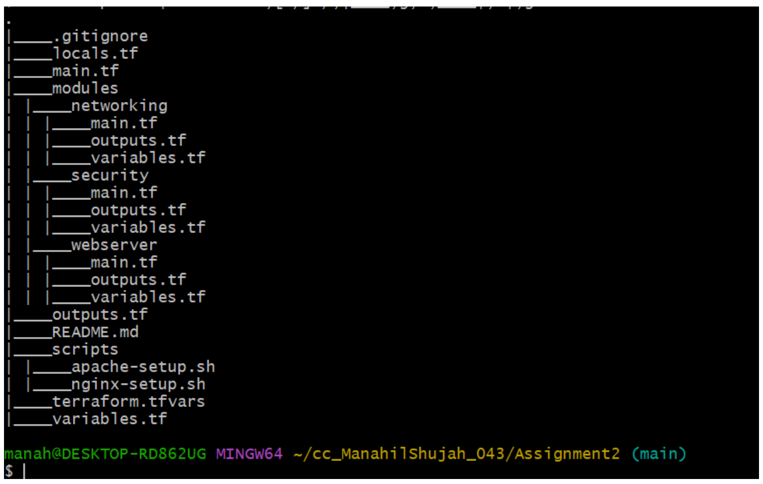


# Part 1: Infrastructure Setup

## 1.1 Project Structure

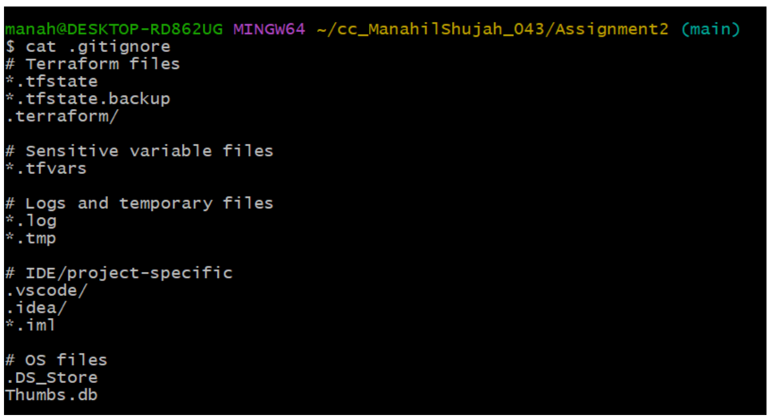
The Terraform project is modularized for clarity and reusability.

### Project Directory Structure:



```
.
|____.gitignore
|____locals.tf
|____main.tf
|____modules
| |____networking
| | |____main.tf
| | |____outputs.tf
| | |____variables.tf
| |____security
| | |____main.tf
| | |____outputs.tf
| | |____variables.tf
| |____webserver
| | |____main.tf
| | |____outputs.tf
| | |____variables.tf
|____outputs.tf
|____README.md
|____scripts
| |____apache-setup.sh
| |____nginx-setup.sh
|____terraform.tfvars
|____variables.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ |
```

### .gitignore Configuration

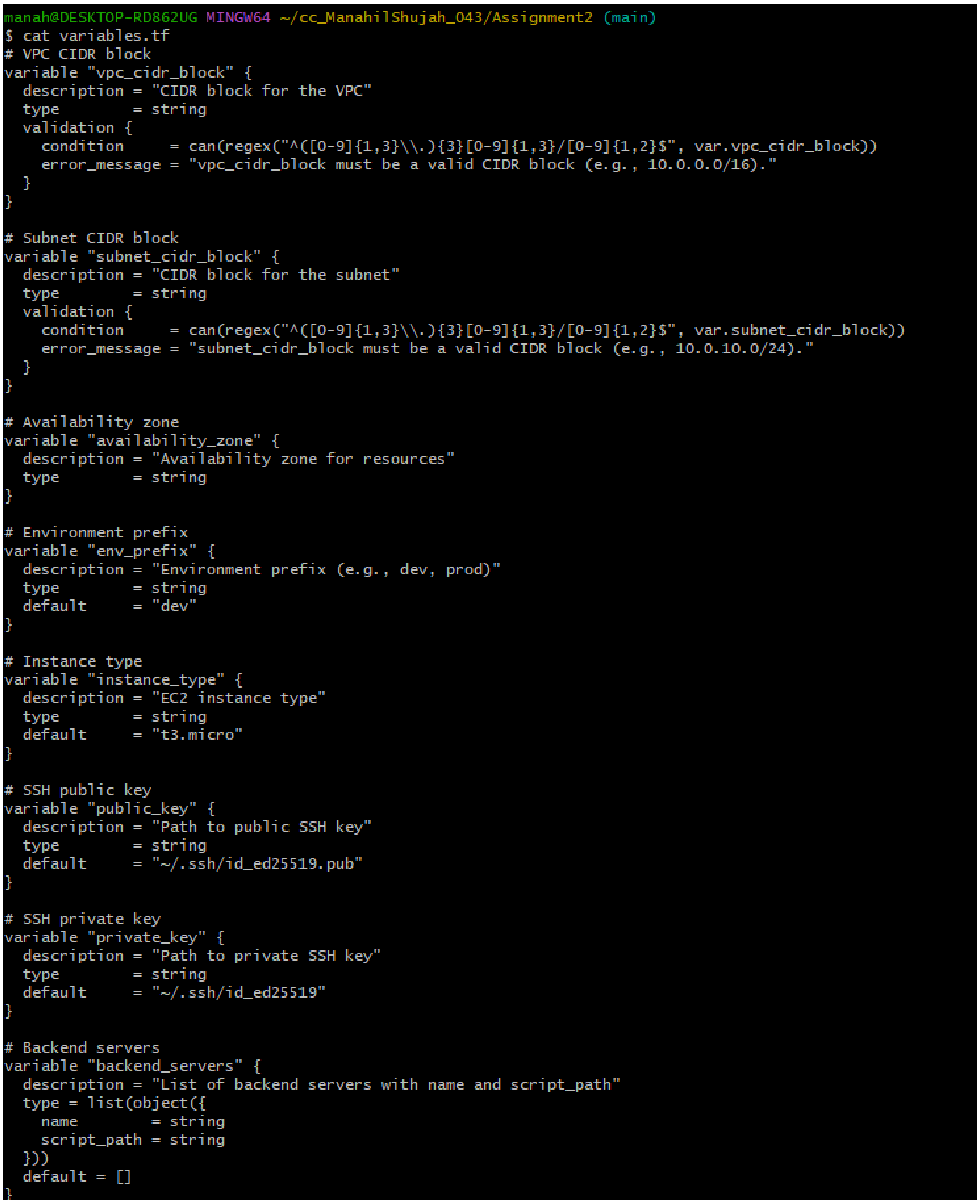


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat .gitignore
# Terraform files
*.tfstate
*.tfstate.backup
.terraform/

# Sensitive variable files
*.tfvars

# Logs and temporary files
*.log
*.tmp

# IDE/project-specific
.vscode/
.idea/
*.iml

# OS files
.DS_Store
Thumbs.db
```

## 1.2 Variable Configuration

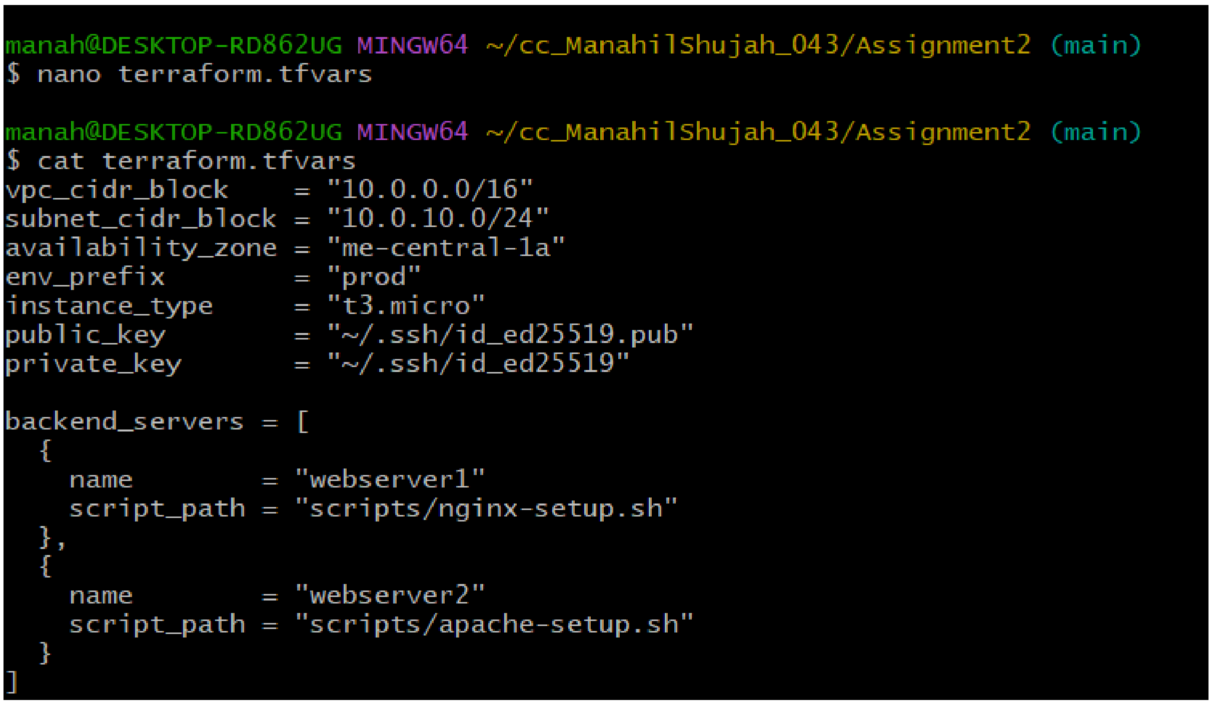
Variables are defined in variables.tf and values are provided using terraform.tfvars.

### Terraform Variables Definition



```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat variables.tf
# VPC CIDR block
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
  validation {
    condition     = can(regex("^([0-9]{1,3}\\.){3}[0-9]{1,3}/[0-9]{1,2}$", var.vpc_cidr_block))
    error_message = "vpc_cidr_block must be a valid CIDR block (e.g., 10.0.0.0/16)."
  }
}

# Subnet CIDR block
variable "subnet_cidr_block" {
  description = "CIDR block for the subnet"
  type        = string
  validation {
    condition     = can(regex("^([0-9]{1,3}\\.){3}[0-9]{1,3}/[0-9]{1,2}$", var.subnet_cidr_block))
    error_message = "subnet_cidr_block must be a valid CIDR block (e.g., 10.0.10.0/24)."
  }
}

# Availability zone
variable "availability_zone" {
  description = "Availability zone for resources"
  type        = string
}

# Environment prefix
variable "env_prefix" {
  description = "Environment prefix (e.g., dev, prod)"
  type        = string
  default     = "dev"
}

# Instance type
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
}

# SSH public key
variable "public_key" {
  description = "Path to public SSH key"
  type        = string
  default     = "~/.ssh/id_ed25519.pub"
}

# SSH private key
variable "private_key" {
  description = "Path to private SSH key"
  type        = string
  default     = "~/.ssh/id_ed25519"
}

# Backend servers
variable "backend_servers" {
  description = "List of backend servers with name and script_path"
  type = list(object({
    name        = string
    script_path = string
  }))
  default = []
}
```

### Terraform Variables Values (terraform.tfvars)

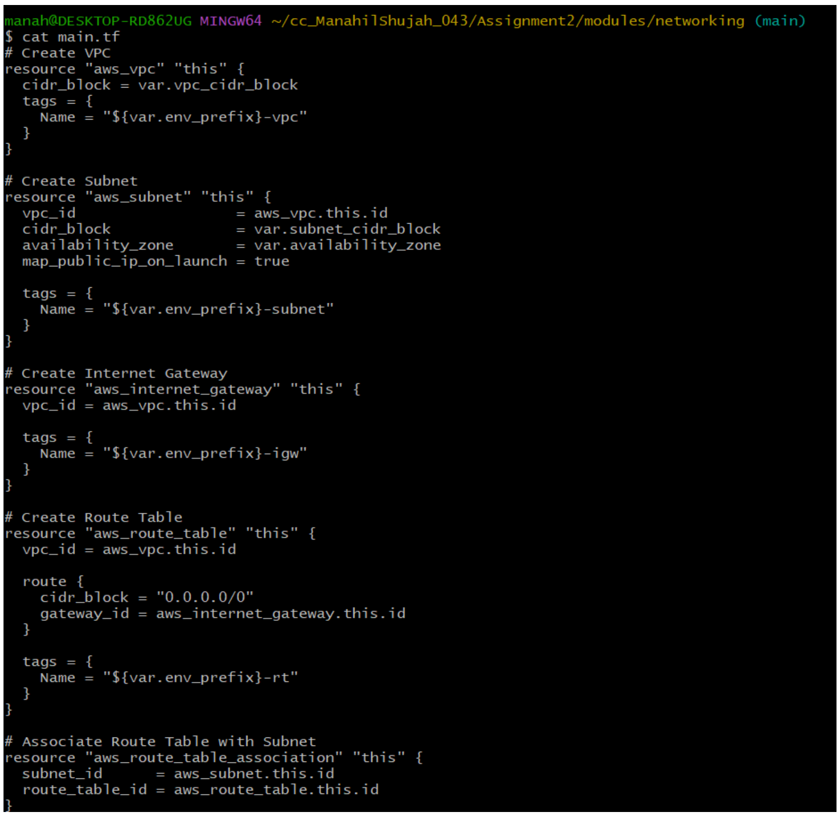


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ nano terraform.tfvars

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat terraform.tfvars
vpc_cidr_block     = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix         = "prod"
instance_type      = "t3.micro"
public_key         = "~/.ssh/id_ed25519.pub"
private_key        = "~/.ssh/id_ed25519"

backend_servers = [
  {
    name        = "webserver1"
    script_path = "scripts/nginx-setup.sh"
  },
  {
    name        = "webserver2"
    script_path = "scripts/apache-setup.sh"
  }
]
```

## 1.3 Networking Module

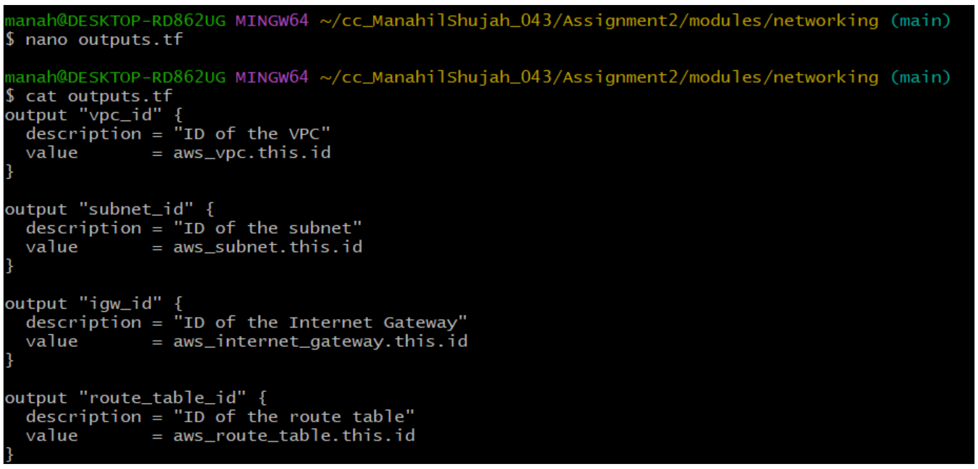
A networking module is used to create VPC, subnet, internet gateway, and route table.

### Networking Module Configuration



```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/networking (main)
$ cat main.tf
# Create VPC
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

# Create Subnet
resource "aws_subnet" "this" {
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.subnet_cidr_block
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.env_prefix}-subnet"
  }
}

# Create Internet Gateway
resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id

  tags = {
    Name = "${var.env_prefix}-igw"
  }
}

# Create Route Table
resource "aws_route_table" "this" {
  vpc_id = aws_vpc.this.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.this.id
  }

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

# Associate Route Table with Subnet
resource "aws_route_table_association" "this" {
  subnet_id      = aws_subnet.this.id
  route_table_id = aws_route_table.this.id
}
```

### Networking Module Outputs

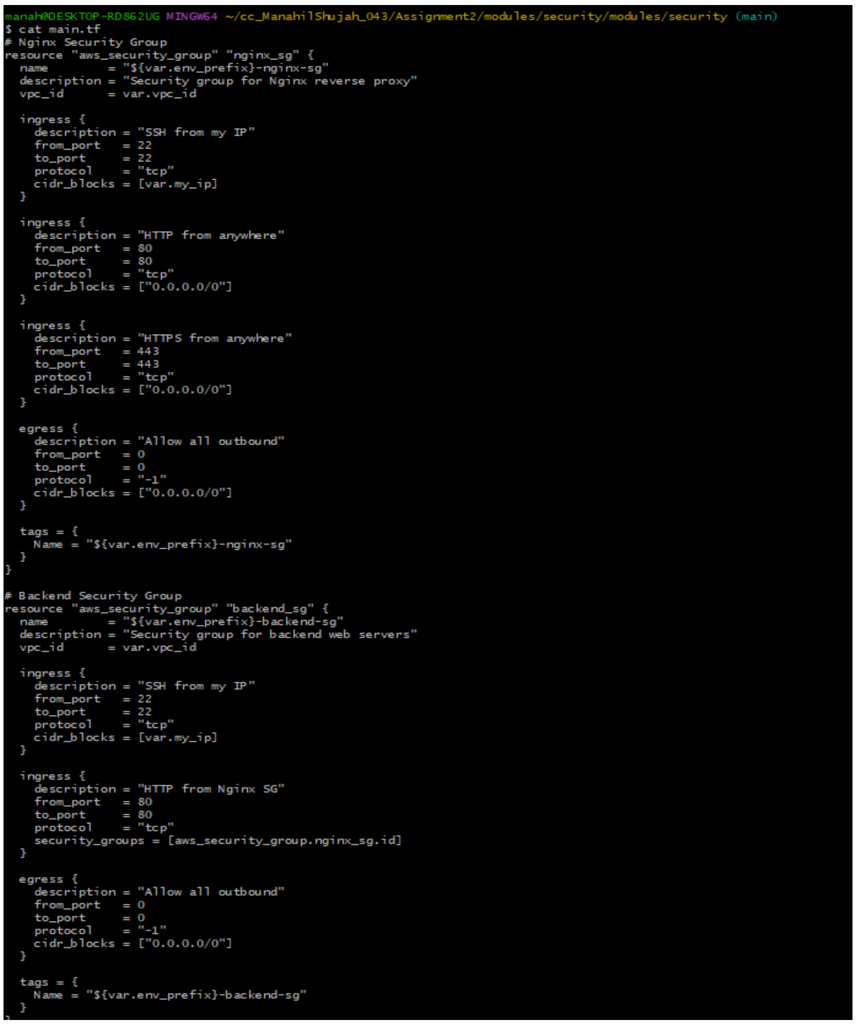


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/networking (main)
$ nano outputs.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/networking (main)
$ cat outputs.tf
output "vpc_id" {
  description = "ID of the VPC"
  value       = aws_vpc.this.id
}

output "subnet_id" {
  description = "ID of the subnet"
  value       = aws_subnet.this.id
}

output "igw_id" {
  description = "ID of the Internet Gateway"
  value       = aws_internet_gateway.this.id
}

output "route_table_id" {
  description = "ID of the route table"
  value       = aws_route_table.this.id
}
```

## 1.4 Security Module

Separate security groups are created for Nginx and backend servers to follow least privilege.

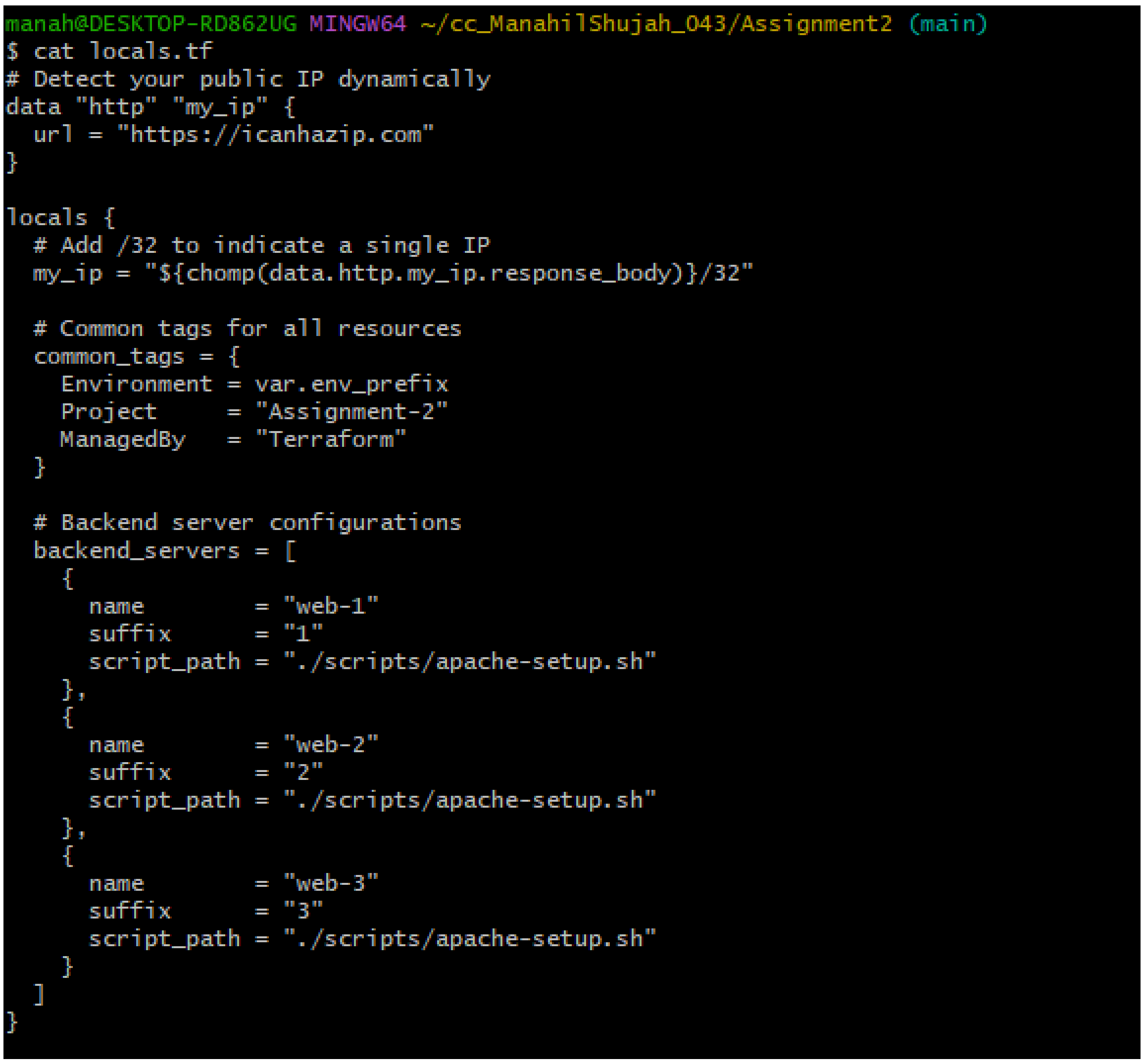
### Security Module Configuration



```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/security/modules/security (main)
$ cat main.tf
# Nginx Security Group
resource "aws_security_group" "nginx_sg" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Security group for Nginx reverse proxy"
  vpc_id      = var.vpc_id

  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }

  ingress {
    description = "HTTP from anywhere"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "HTTPS from anywhere"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    description = "Allow all outbound"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

# Backend Security Group
resource "aws_security_group" "backend_sg" {
  name        = "${var.env_prefix}-backend-sg"
  description = "Security group for backend web servers"
  vpc_id      = var.vpc_id

  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }

  ingress {
    description = "HTTP from Nginx SG"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    security_groups = [aws_security_group.nginx_sg.id]
  }

  egress {
    description = "Allow all outbound"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "${var.env_prefix}-backend-sg"
  }
}
```

### 1.5 Locals Configuration

Locals are used for dynamic IP detection, common tags, and backend server definitions.



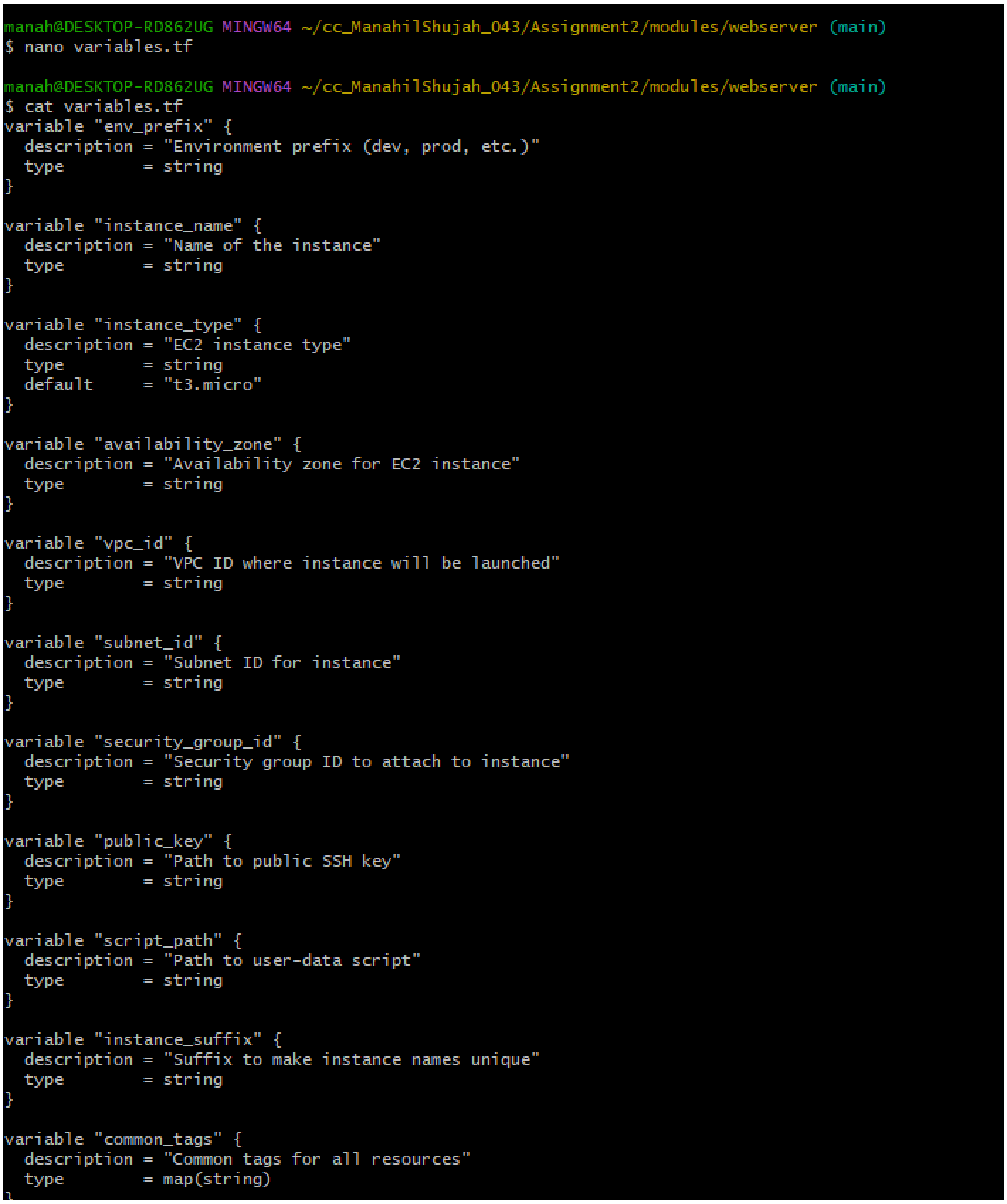
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat locals.tf
# Detect your public IP dynamically
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

locals {
  # Add /32 to indicate a single IP
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"

  # Common tags for all resources
  common_tags = {
    Environment = var.env_prefix
    Project     = "Assignment-2"
    ManagedBy   = "Terraform"
  }

  # Backend server configurations
  backend_servers = [
    {
      name        = "web-1"
      suffix      = "1"
      script_path = "./scripts/apache-setup.sh"
    },
    {
      name        = "web-2"
      suffix      = "2"
      script_path = "./scripts/apache-setup.sh"
    },
    {
      name        = "web-3"
      suffix      = "3"
      script_path = "./scripts/apache-setup.sh"
    }
  ]
}
```

# Part 2: Webserver Module

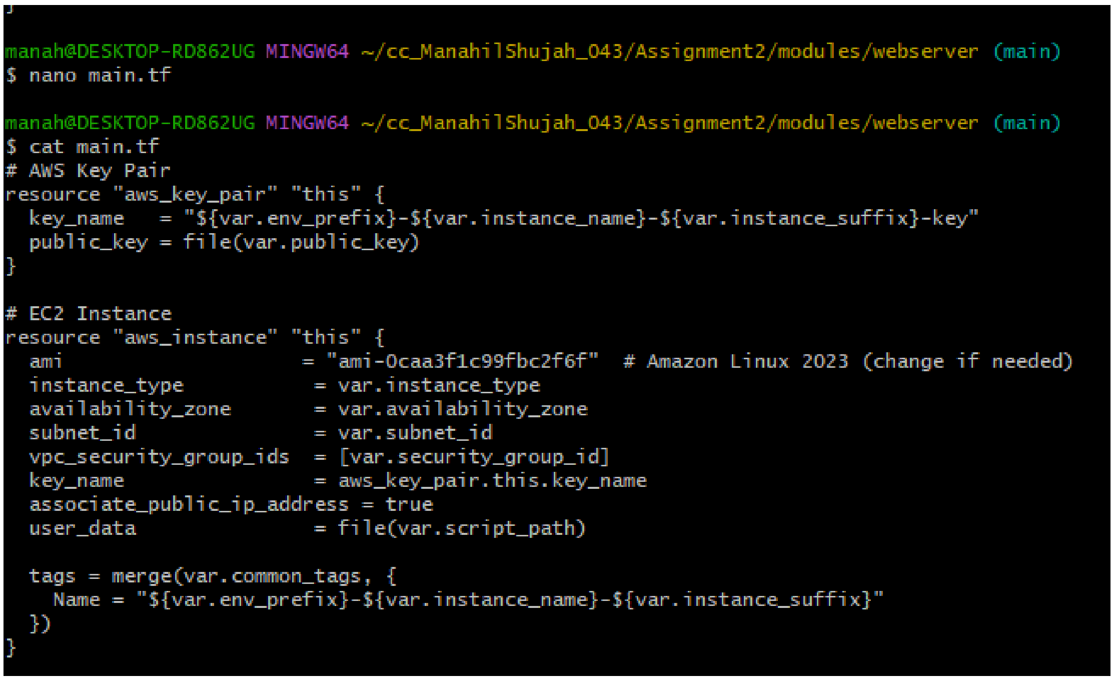
### 2.1 Module Design

A reusable webserver module is created for both Nginx and backend servers.

**Webserver Module Variables**

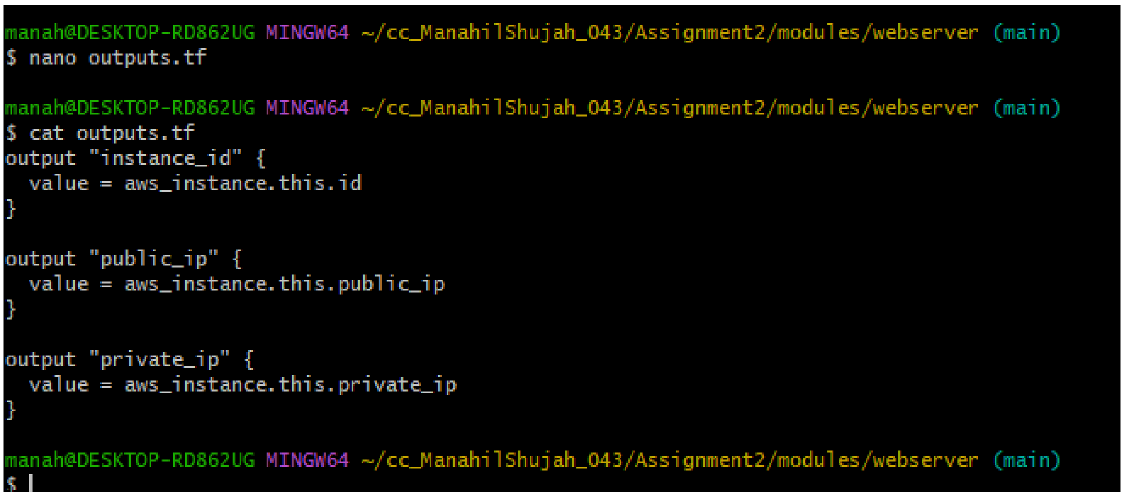

```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ nano variables.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ cat variables.tf
variable "env_prefix" {
  description = "Environment prefix (dev, prod, etc.)"
  type        = string
}

variable "instance_name" {
  description = "Name of the instance"
  type        = string
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
}

variable "availability_zone" {
  description = "Availability zone for EC2 instance"
  type        = string
}

variable "vpc_id" {
  description = "VPC ID where instance will be launched"
  type        = string
}

variable "subnet_id" {
  description = "Subnet ID for instance"
  type        = string
}

variable "security_group_id" {
  description = "Security group ID to attach to instance"
  type        = string
}

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

variable "script_path" {
  description = "Path to user-data script"
  type        = string
}

variable "instance_suffix" {
  description = "Suffix to make instance names unique"
  type        = string
}

variable "common_tags" {
  description = "Common tags for all resources"
  type        = map(string)
```

## Webserver Module Resources



```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ nano main.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ cat main.tf
# AWS Key Pair
resource "aws_key_pair" "this" {
  key_name   = "${var.env_prefix}-${var.instance_name}-${var.instance_suffix}-key"
  public_key = file(var.public_key)
}

# EC2 Instance
resource "aws_instance" "this" {
  ami                    = "ami-0caa3f1c99fbc2f6f"  # Amazon Linux 2023 (change if needed)
  instance_type          = var.instance_type
  availability_zone      = var.availability_zone
  subnet_id              = var.subnet_id
  vpc_security_group_ids  = [var.security_group_id]
  key_name               = aws_key_pair.this.key_name
  associate_public_ip_address = true
  user_data              = file(var.script_path)

  tags = merge(var.common_tags, {
    Name = "${var.env_prefix}-${var.instance_name}-${var.instance_suffix}"
  })
}
```

### Webserver Module Outputs

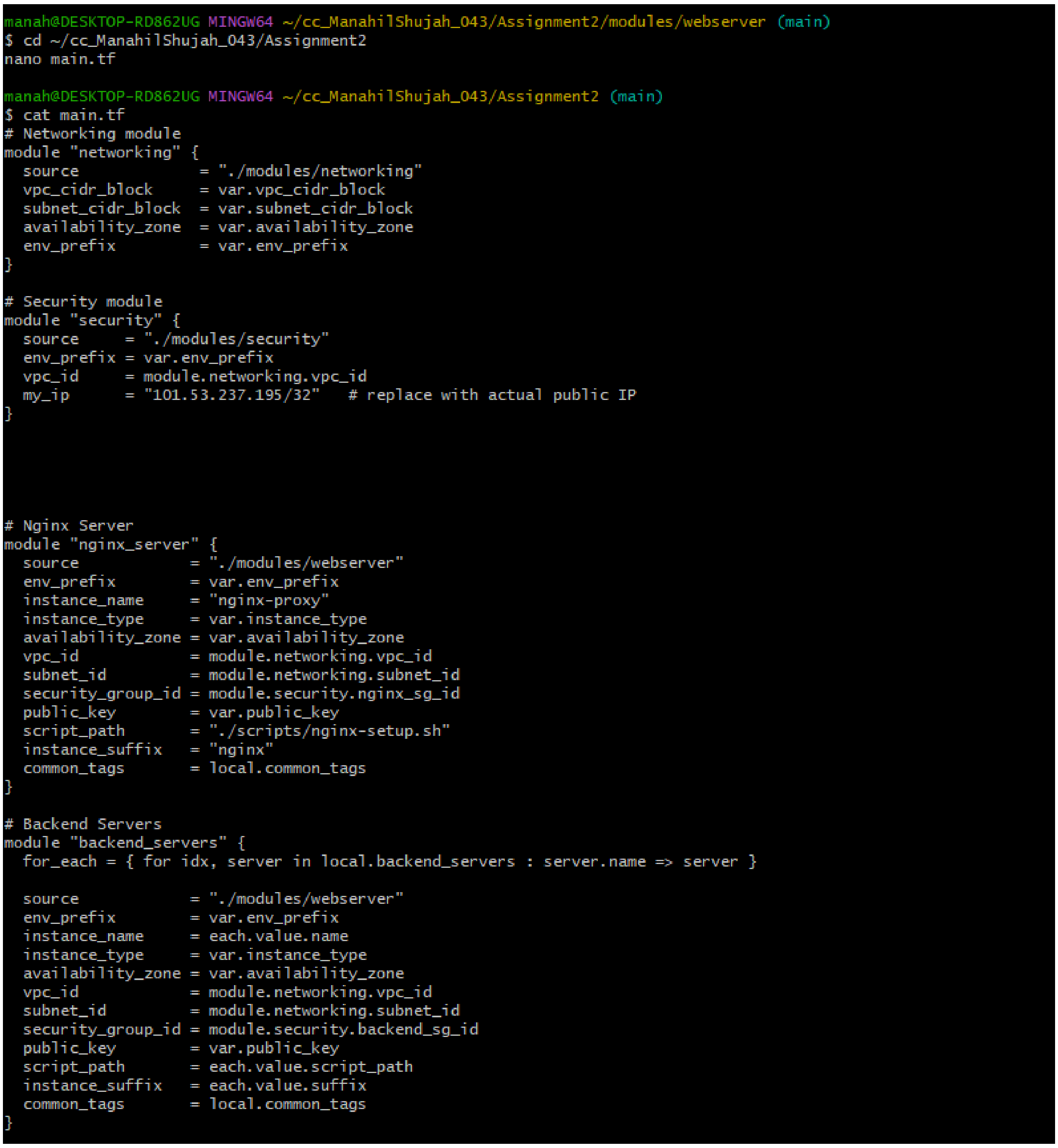


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ nano outputs.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ cat outputs.tf
output "instance_id" {
  value = aws_instance.this.id
}

output "public_ip" {
  value = aws_instance.this.public_ip
}

output "private_ip" {
  value = aws_instance.this.private_ip
}

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ 
```

## 2.2 Module Usage

The module is instantiated once for Nginx and multiple times for backend servers using for each.

### Root Module Webserver Integration



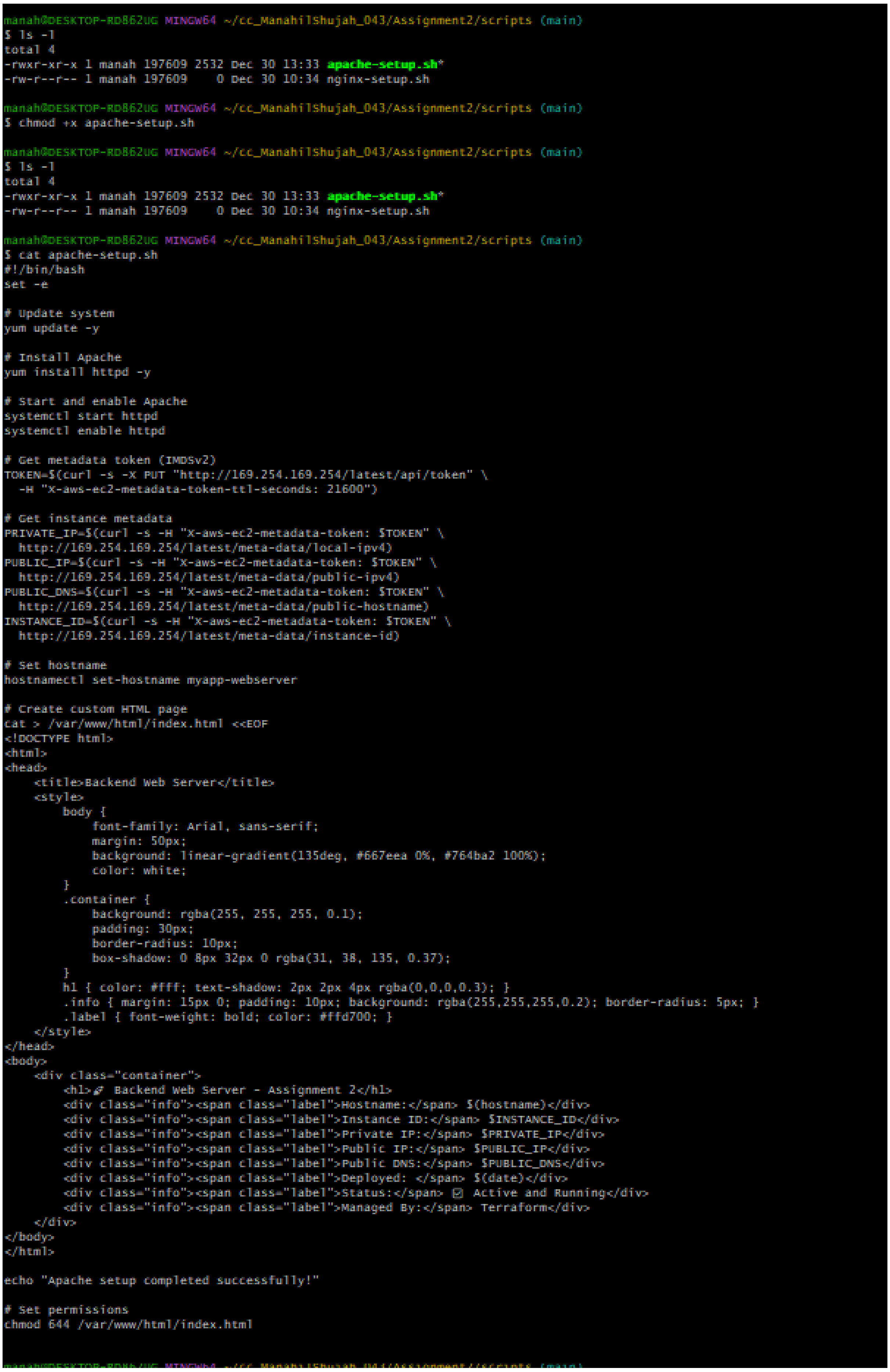
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/modules/webserver (main)
$ cd ~/cc_ManahilShujah_043/Assignment2
nano main.tf

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat main.tf
# Networking module
module "networking" {
  source             = "./modules/networking"
  vpc_cidr_block     = var.vpc_cidr_block
  subnet_cidr_block  = var.subnet_cidr_block
  availability_zone  = var.availability_zone
  env_prefix         = var.env_prefix
}

# Security module
module "security" {
  source     = "./modules/security"
  env_prefix = var.env_prefix
  vpc_id     = module.networking.vpc_id
  my_ip      = "101.53.237.195/32"   # replace with actual public IP
}




# Nginx Server
module "nginx_server" {
  source            = "./modules/webserver"
  env_prefix        = var.env_prefix
  instance_name     = "nginx-proxy"
  instance_type     = var.instance_type
  availability_zone = var.availability_zone
  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.nginx_sg_id
  public_key        = var.public_key
  script_path       = "./scripts/nginx-setup.sh"
  instance_suffix   = "nginx"
  common_tags       = local.common_tags
}

# Backend Servers
module "backend_servers" {
  for_each = { for idx, server in local.backend_servers : server.name => server }

  source            = "./modules/webserver"
  env_prefix        = var.env_prefix
  instance_name     = each.value.name
  instance_type     = var.instance_type
  availability_zone = var.availability_zone
  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.backend_sg_id
  public_key        = var.public_key
  script_path       = each.value.script_path
  instance_suffix   = each.value.suffix
  common_tags       = local.common_tags
}
```

## Part 3: Server Configuration Scripts

### 3.1 Apache Backend Server Script

Apache is installed and a custom HTML page is generated using EC2 metadata.

#### Apache Setup Script

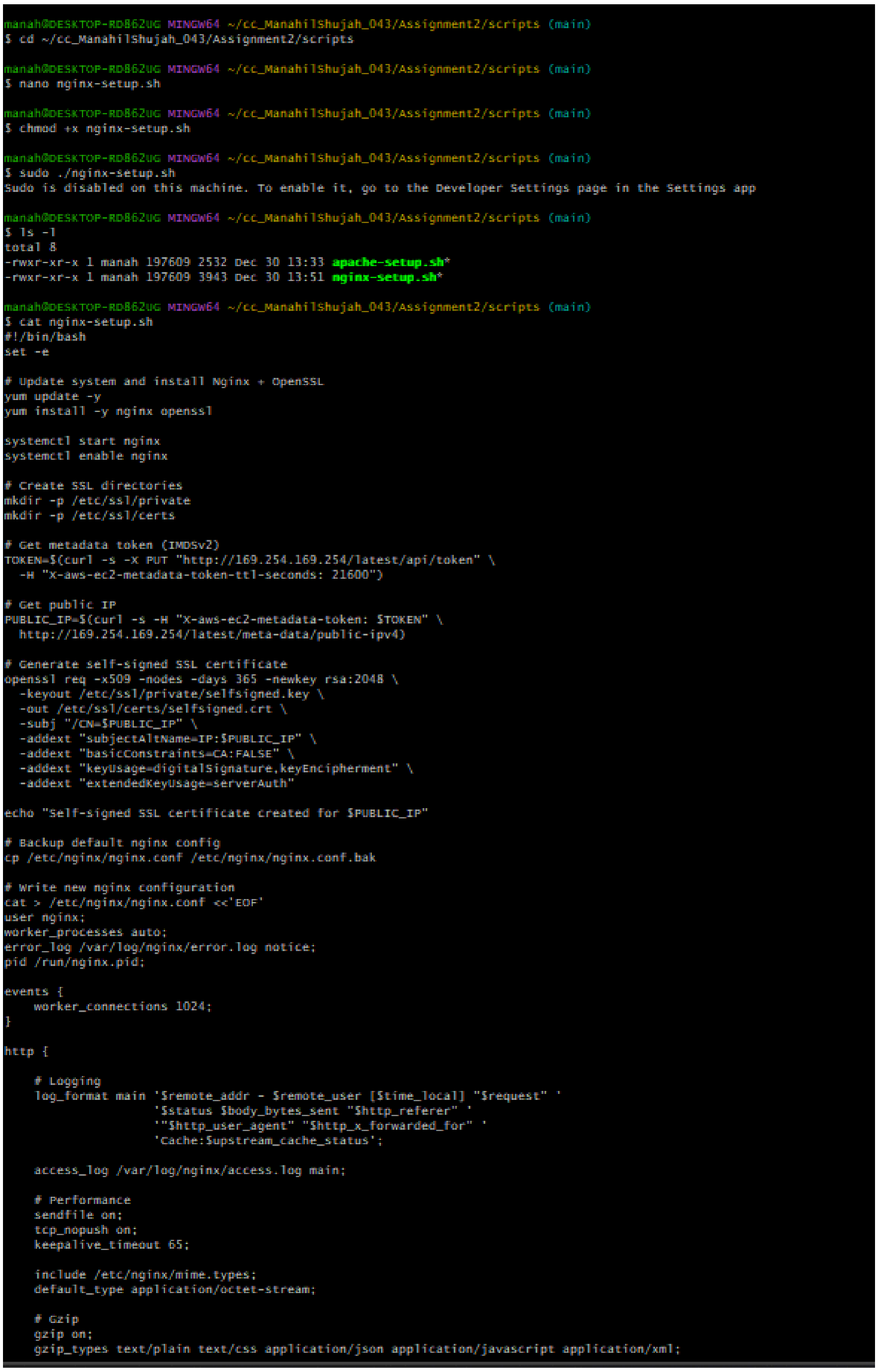


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ ls -l
total 4
-rwxr-xr-x 1 manah 197609 2532 Dec 30 13:33 apache-setup.sh*
-rw-r--r-- 1 manah 197609    0 Dec 30 10:34 nginx-setup.sh

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ chmod +x apache-setup.sh

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ ls -l
total 4
-rwxr-xr-x 1 manah 197609 2532 Dec 30 13:33 apache-setup.sh*
-rw-r--r-- 1 manah 197609    0 Dec 30 10:34 nginx-setup.sh

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ cat apache-setup.sh
#!/bin/bash
set -e

# Update system
yum update -y

# Install Apache
yum install httpd -y

# Start and enable Apache
systemctl start httpd
systemctl enable httpd

# Get metadata token (IMDSv2)
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

# Get instance metadata
PRIVATE_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/local-ipv4)
PUBLIC_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/public-ipv4)
PUBLIC_DNS=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/public-hostname)
INSTANCE_ID=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/instance-id)

# Set hostname
hostnamectl set-hostname myapp-webserver

# Create custom HTML page
cat > /var/www/html/index.html <<EOF
<!DOCTYPE html>
<html>
<head>
    <title>Backend Web Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 50px;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
        }
        h1 { color: #fff; text-shadow: 2px 2px 4px rgba(0,0,0,0.3); }
        .info { margin: 15px 0; padding: 10px; background: rgba(255,255,255,0.2); border-radius: 5px; }
        .label { font-weight: bold; color: #ffd700; }
    </style>
</head>
<body>
    <div class="container">
        <h1>🚀 Backend Web Server - Assignment 2</h1>
        <div class="info"><span class="label">Hostname:</span> $(hostname)</div>
        <div class="info"><span class="label">Instance ID:</span> $INSTANCE_ID</div>
        <div class="info"><span class="label">Private IP:</span> $PRIVATE_IP</div>
        <div class="info"><span class="label">Public IP:</span> $PUBLIC_IP</div>
        <div class="info"><span class="label">Public DNS:</span> $PUBLIC_DNS</div>
        <div class="info"><span class="label">Deployed: </span> $(date)</div>
        <div class="info"><span class="label">Status:</span> ☑ Active and Running</div>
        <div class="info"><span class="label">Managed By:</span> Terraform</div>
    </div>
</body>
</html>

echo "Apache setup completed successfully!"

# Set permissions
chmod 644 /var/www/html/index.html


manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
```

## 3.2 Nginx Server Setup Script

Nginx is configured with SSL, caching, load balancing, and security headers.

### Nginx Setup Script



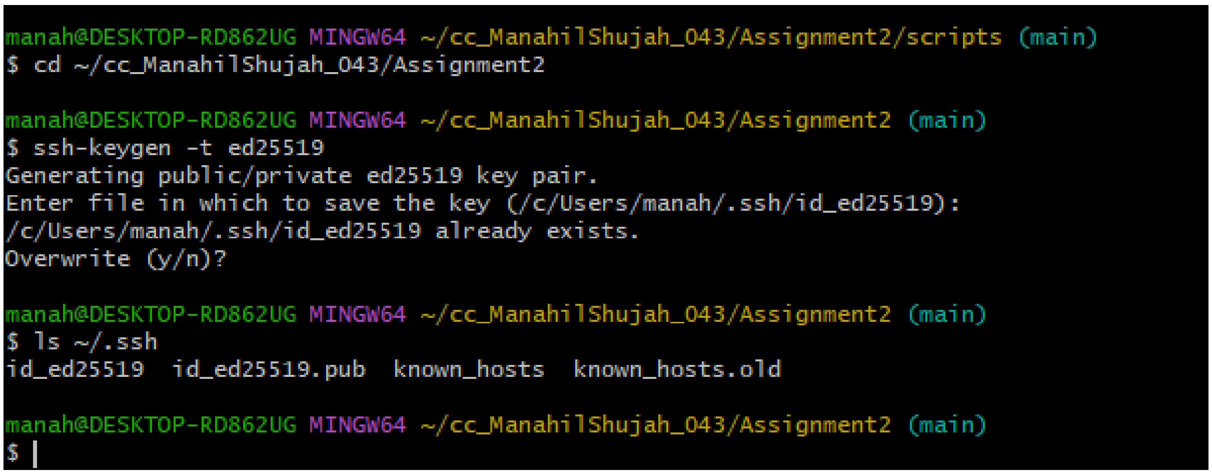
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ cd ~/cc_ManahilShujah_043/Assignment2/scripts

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ nano nginx-setup.sh

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ chmod +x nginx-setup.sh

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ sudo ./nginx-setup.sh
Sudo is disabled on this machine. To enable it, go to the Developer Settings page in the Settings app

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ ls -l
total 8
-rwxr-xr-x 1 manah 197609 2532 Dec 30 13:33 apache-setup.sh*
-rwxr-xr-x 1 manah 197609 3943 Dec 30 13:51 nginx-setup.sh*

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ cat nginx-setup.sh
#!/bin/bash
set -e

# Update system and install Nginx + OpenSSL
yum update -y
yum install -y nginx openssl

systemctl start nginx
systemctl enable nginx

# Create SSL directories
mkdir -p /etc/ssl/private
mkdir -p /etc/ssl/certs

# Get metadata token (IMDSv2)
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

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

# Generate self-signed SSL 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 SSL certificate created for $PUBLIC_IP"

# Backup default nginx config
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak

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

events {
    worker_connections 1024;
}

http {

    # Logging
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for" '
                    'Cache:$upstream_cache_status';

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

    # Performance
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 65;

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

    # Gzip
    gzip on;
    gzip_types text/plain text/css application/json application/javascript application/xml;
```

# Part 4: Infrastructure Deployment

## 4.1 Initial Deployment

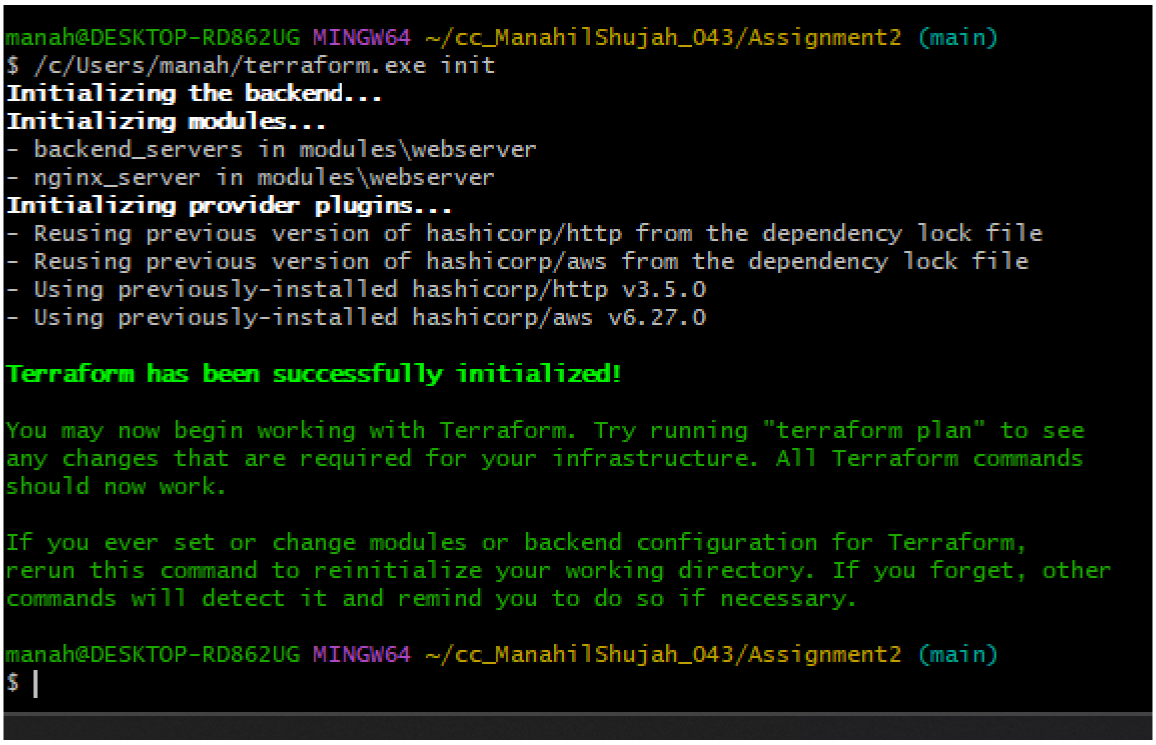
Terraform is initialized, validated, planned, and applied successfully.

### SSH Key Generation



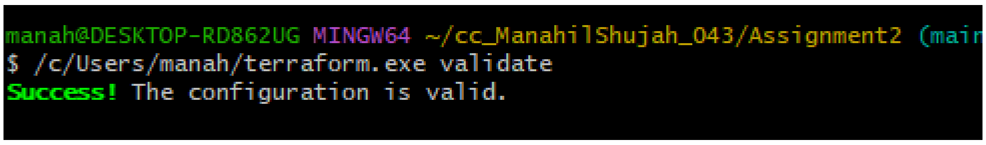
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2/scripts (main)
$ cd ~/cc_ManahilShujah_043/Assignment2

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/manah/.ssh/id_ed25519):
/c/Users/manah/.ssh/id_ed25519 already exists.
Overwrite (y/n)?

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ ls ~/.ssh
id_ed25519  id_ed25519.pub  known_hosts  known_hosts.old

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ 
```

### Terraform Initialization



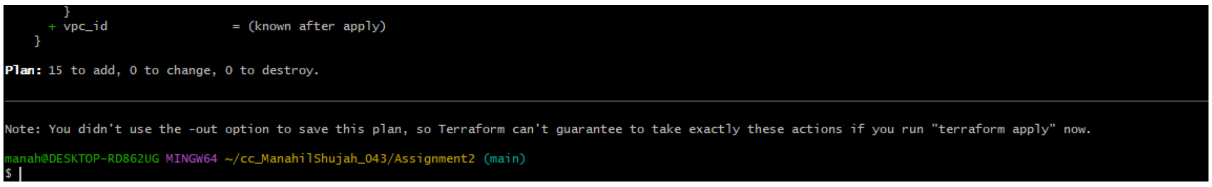
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ /c/Users/manah/terraform.exe init
Initializing the backend...
Initializing modules...
- backend_servers in modules\webserver
- nginx_server in modules\webserver
Initializing provider plugins...
- Reusing previous version of 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.

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ 
```

### Terraform Validation



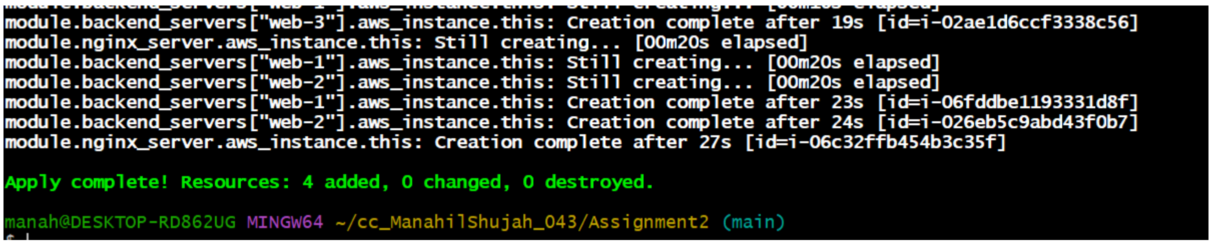
```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (mai
$ /c/Users/manah/terraform.exe validate
Success! The configuration is valid.
```

### Terraform Plan



```
        }
      + vpc_id                 = (known after apply)
    }

Plan: 15 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ |
```

### Terraform Apply

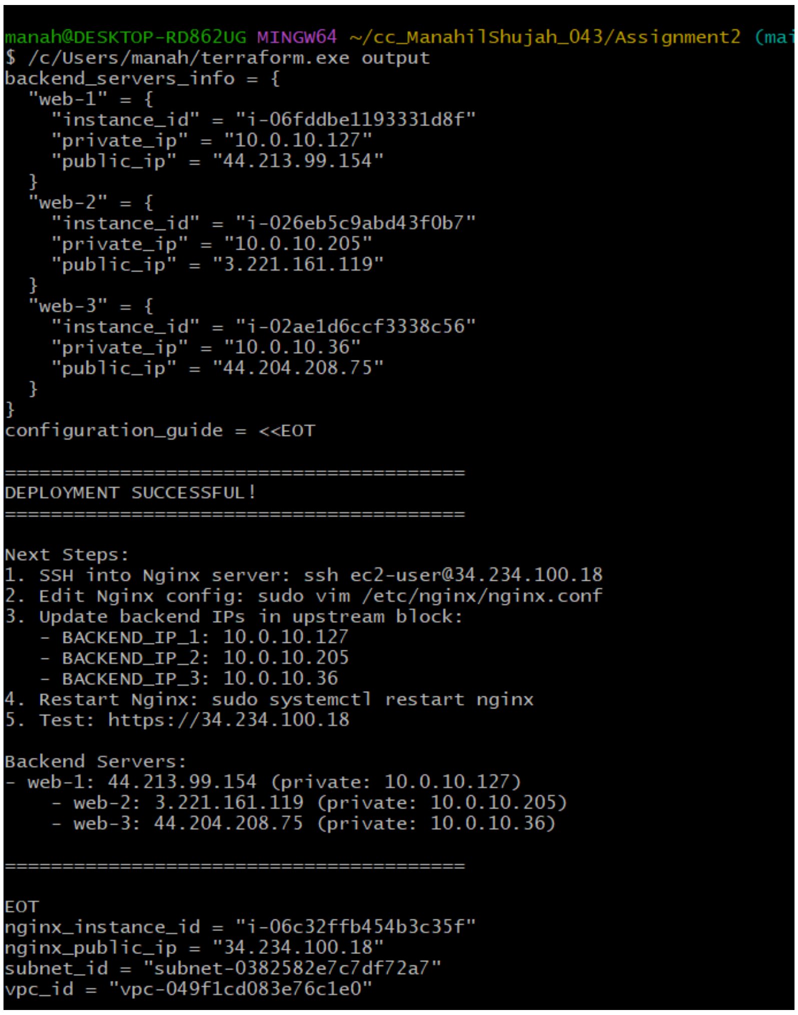


```
module.backend_servers["web-3"].aws_instance.this: Creation complete after 19s [id=i-02ae1d6ccf3338c56]
module.nginx_server.aws_instance.this: Still creating... [00m20s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still creating... [00m20s elapsed]
module.backend_servers["web-2"].aws_instance.this: Still creating... [00m20s elapsed]
module.backend_servers["web-1"].aws_instance.this: Creation complete after 23s [id=i-06fddbe1193331d8f]
module.backend_servers["web-2"].aws_instance.this: Creation complete after 24s [id=i-026eb5c9abd43f0b7]
module.nginx_server.aws_instance.this: Creation complete after 27s [id=i-06c32ffb454b3c35f]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
```

## 4.2 Output Configuration

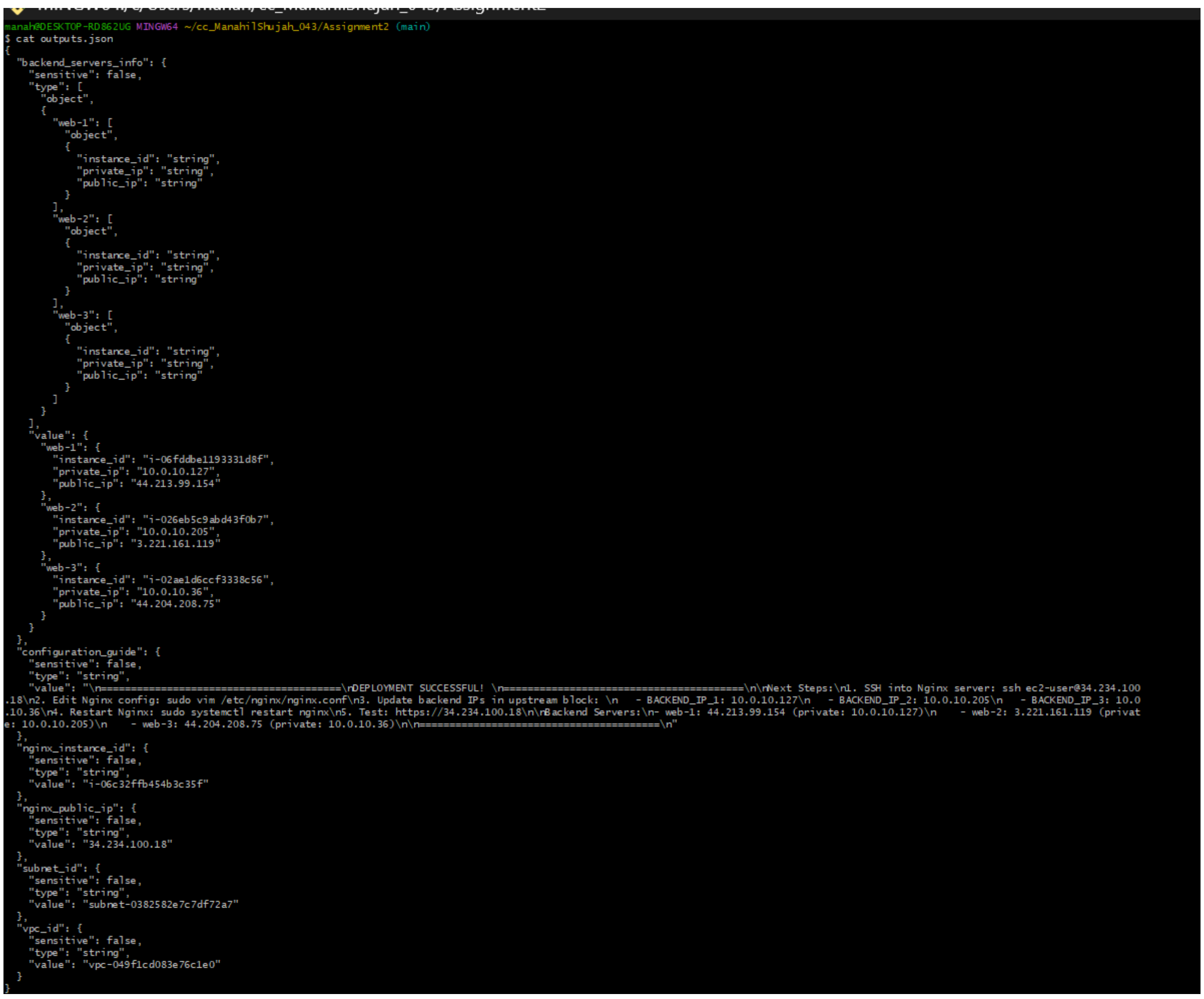
Outputs display server IPs and configuration instructions.

### Terraform Output Display



```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (ma
$ /c/Users/manah/terraform.exe output
backend_servers_info = {
  "web-1" = {
    "instance_id" = "i-06fddbe1193331d8f"
    "private_ip" = "10.0.10.127"
    "public_ip" = "44.213.99.154"
  }
  "web-2" = {
    "instance_id" = "i-026eb5c9abd43f0b7"
    "private_ip" = "10.0.10.205"
    "public_ip" = "3.221.161.119"
  }
  "web-3" = {
    "instance_id" = "i-02ae1d6ccf3338c56"
    "private_ip" = "10.0.10.36"
    "public_ip" = "44.204.208.75"
  }
}
configuration_guide = <<EOT

========================================
DEPLOYMENT SUCCESSFUL!
========================================

Next Steps:
1. SSH into Nginx server: ssh ec2-user@34.234.100.18
2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf
3. Update backend IPs in upstream block:
   - BACKEND_IP_1: 10.0.10.127
   - BACKEND_IP_2: 10.0.10.205
   - BACKEND_IP_3: 10.0.10.36
4. Restart Nginx: sudo systemctl restart nginx
5. Test: https://34.234.100.18

Backend Servers:
- web-1: 44.213.99.154 (private: 10.0.10.127)
    - web-2: 3.221.161.119 (private: 10.0.10.205)
    - web-3: 44.204.208.75 (private: 10.0.10.36)

========================================

EOT
nginx_instance_id = "i-06c32ffb454b3c35f"
nginx_public_ip = "34.234.100.18"
subnet_id = "subnet-0382582e7c7df72a7"
vpc_id = "vpc-049f1cd083e76c1e0"
```

## Terraform Outputs JSON File

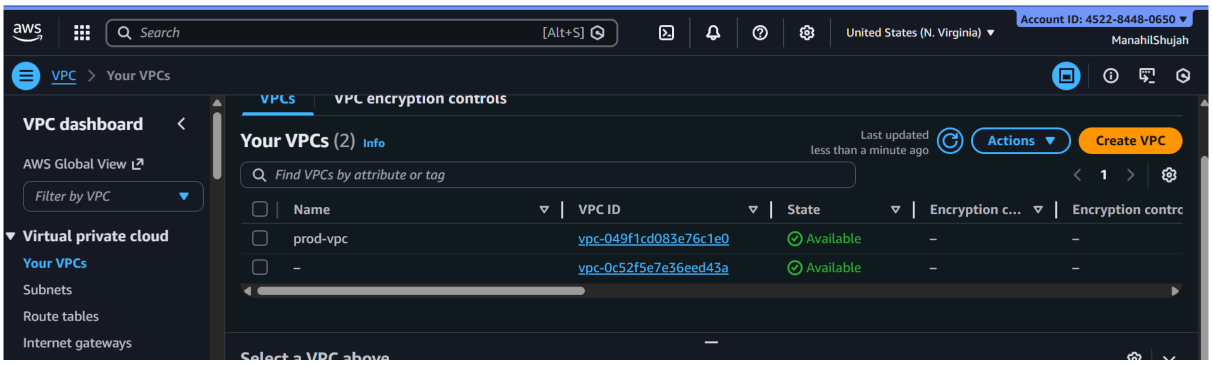


```
manah@DESKTOP-RD862UG MINGW64 ~/cc_ManahilShujah_043/Assignment2 (main)
$ cat outputs.json
{
  "backend_servers_info": {
    "sensitive": false,
    "type": [
      "object",
      {
        "web-1": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-2": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-3": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ]
      }
    ],
    "value": {
      "web-1": {
        "instance_id": "i-06fddbe1193331d8f",
        "private_ip": "10.0.10.127",
        "public_ip": "44.213.99.154"
      },
      "web-2": {
        "instance_id": "i-026eb5c9abd43f0b7",
        "private_ip": "10.0.10.205",
        "public_ip": "3.221.161.119"
      },
      "web-3": {
        "instance_id": "i-02ae1d6ccf3338c56",
        "private_ip": "10.0.10.36",
        "public_ip": "44.204.208.75"
      }
    }
  },
  "configuration_guide": {
    "sensitive": false,
    "type": "string",
    "value": "\n========================================\nDEPLOYMENT SUCCESSFUL! \n========================================\n\nNext Steps:\n1. SSH into Nginx server: ssh ec2-user@34.234.100
.18\n2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf\n3. Update backend IPs in upstream block: \n   - BACKEND_IP_1: 10.0.10.127\n   - BACKEND_IP_2: 10.0.10.205\n   - BACKEND_IP_3: 10.0
.10.36\n4. Restart Nginx: sudo systemctl restart nginx\n5. Test: https://34.234.100.18\n\nBackend Servers:\n- web-1: 44.213.99.154 (private: 10.0.10.127)\n    - web-2: 3.221.161.119 (privat
e: 10.0.10.205)\n    - web-3: 44.204.208.75 (private: 10.0.10.36)\n\n========================================\n"
  },
  "nginx_instance_id": {
    "sensitive": false,
    "type": "string",
    "value": "i-06c32ffb454b3c35f"
  },
  "nginx_public_ip": {
    "sensitive": false,
    "type": "string",
    "value": "34.234.100.18"
  },
  "subnet_id": {
    "sensitive": false,
    "type": "string",
    "value": "subnet-0382582e7c7df72a7"
  },
  "vpc_id": {
    "sensitive": false,
    "type": "string",
    "value": "vpc-049f1cd083e76c1e0"
  }
}
```

## 4.3 AWS Console Verification

All resources were verified in the AWS Console.

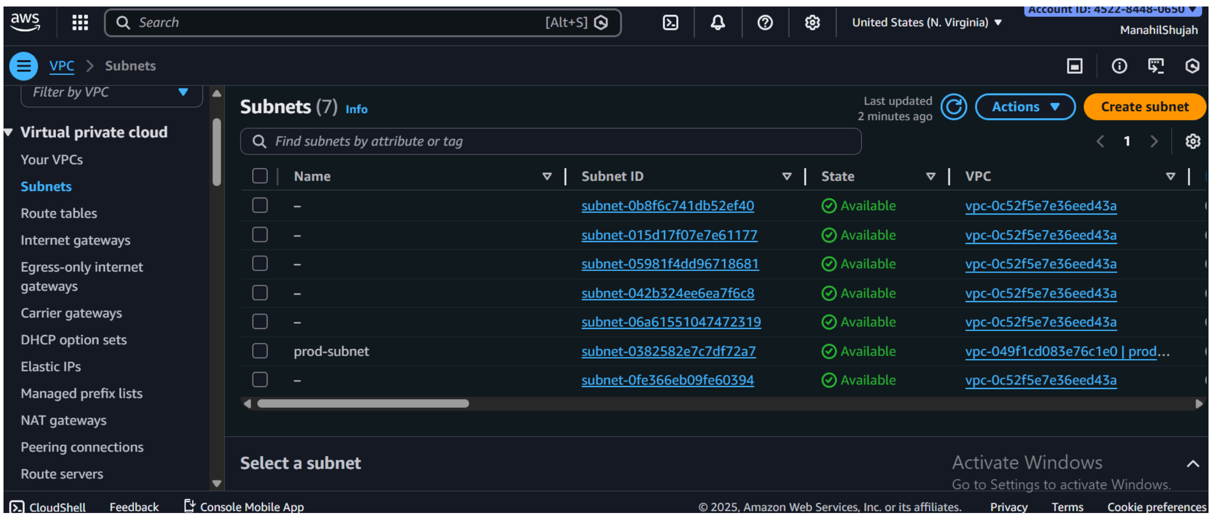
### AWS VPC Verification



Your VPCs (2) Info

| Name | VPC ID | State | Encryption c... | Encryption contro |
|---|---|---|---|---|
| prod-vpc | vpc-049f1cd083e76c1e0 | Available | – | – |
| – | vpc-0c52f5e7e36eed43a | Available | – | – |

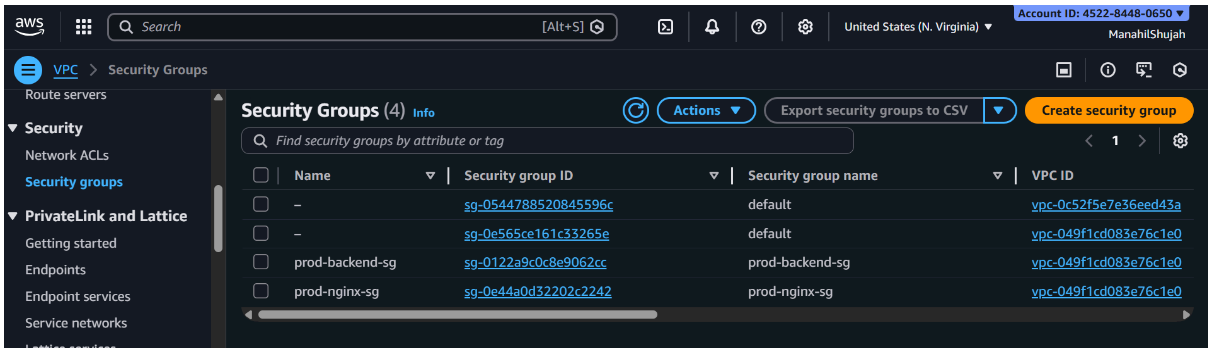
### AWS Subnet Verification



Subnets (7)

| Name | Subnet ID | State | VPC |
|---|---|---|---|
| – | subnet-0b8f6c741db52ef40 | Available | vpc-0c52f5e7e36eed43a |
| – | subnet-015d17f07e7e61177 | Available | vpc-0c52f5e7e36eed43a |
| – | subnet-05981f4dd96718681 | Available | vpc-0c52f5e7e36eed43a |
| – | subnet-042b324ee6ea7f6c8 | Available | vpc-0c52f5e7e36eed43a |
| – | subnet-06a61551047472319 | Available | vpc-0c52f5e7e36eed43a |
| prod-subnet | subnet-0382582e7c7df72a7 | Available | vpc-049f1cd083e76c1e0 \| prod... |
| – | subnet-0fe366eb09fe60394 | Available | vpc-0c52f5e7e36eed43a |

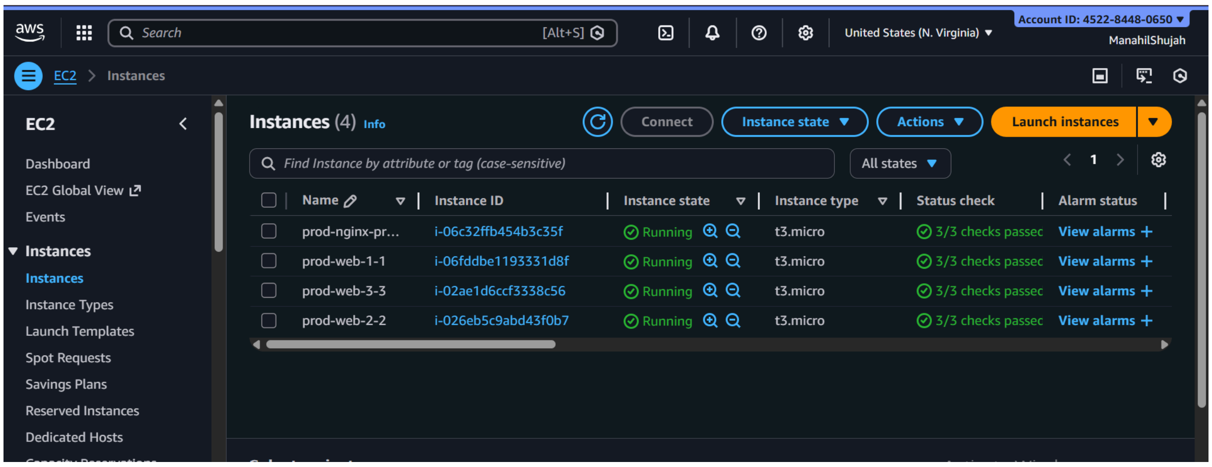
## AWS Security Groups Verification



Security Groups (4)

| Name | Security group ID | Security group name | VPC ID |
|---|---|---|---|
| – | sg-0544788520845596c | default | vpc-0c52f5e7e36eed43a |
| – | sg-0e565ce161c33265e | default | vpc-049f1cd083e76c1e0 |
| prod-backend-sg | sg-0122a9c0c8e9062cc | prod-backend-sg | vpc-049f1cd083e76c1e0 |
| prod-nginx-sg | sg-0e44a0d32202c2242 | prod-nginx-sg | vpc-049f1cd083e76c1e0 |

## AWS EC2 Instances Verification



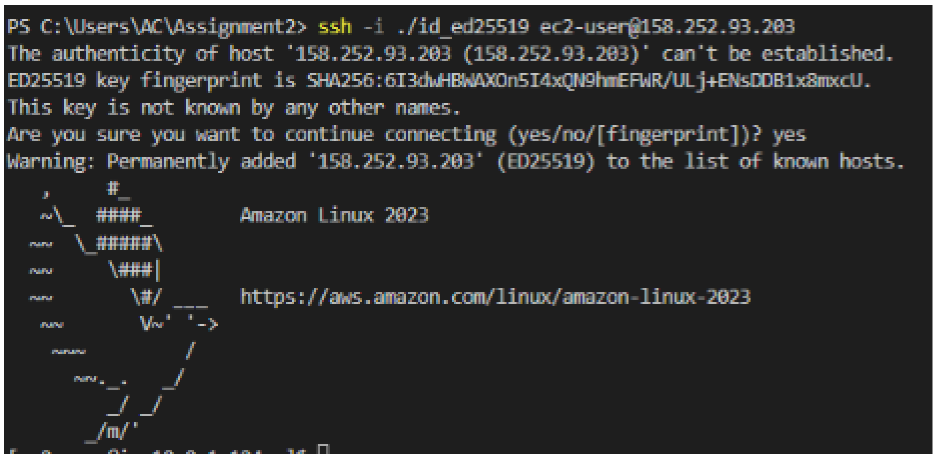
Instances (4)

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status |
|---|---|---|---|---|---|
| prod-nginx-pr... | i-06c32ffb454b3c35f | Running | t3.micro | 3/3 checks passec | View alarms + |
| prod-web-1-1 | i-06fddbe1193331d8f | Running | t3.micro | 3/3 checks passec | View alarms + |
| prod-web-3-3 | i-02ae1d6ccf3338c56 | Running | t3.micro | 3/3 checks passec | View alarms + |
| prod-web-2-2 | i-026eb5c9abd43f0b7 | Running | t3.micro | 3/3 checks passec | View alarms + |

# Part 5: Nginx Configuration & Testing

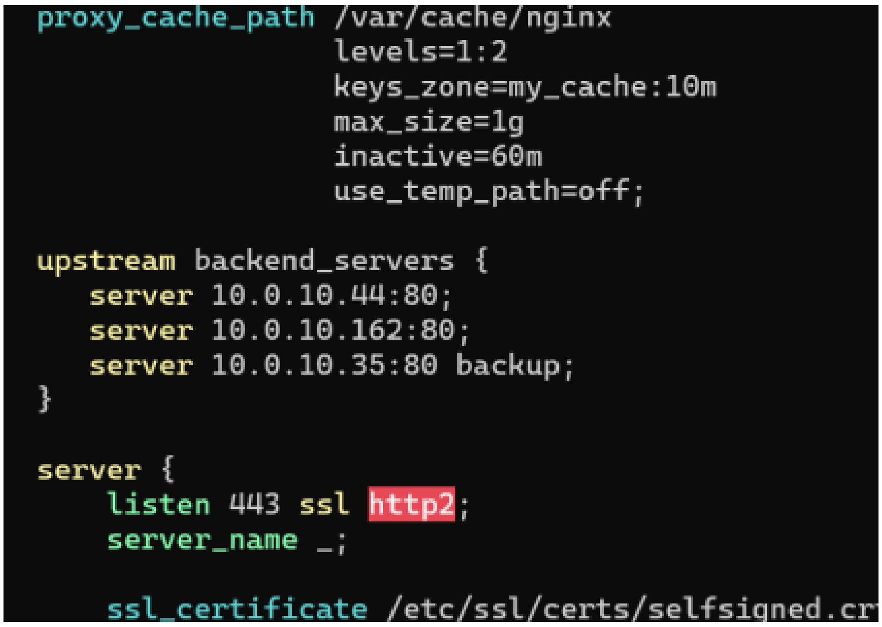
## 5.1 Update Nginx Backend Configuration

### SSH into Nginx Server



```
PS C:\Users\AC\Assignment2> ssh -i ./id_ed25519 ec2-user@158.252.93.203
The authenticity of host '158.252.93.203 (158.252.93.203)' can't be established.
ED25519 key fingerprint is SHA256:6I3dwHBWAXOn5I4xQN9hmEFWR/ULj+ENsDDB1x8mxcU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '158.252.93.203' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

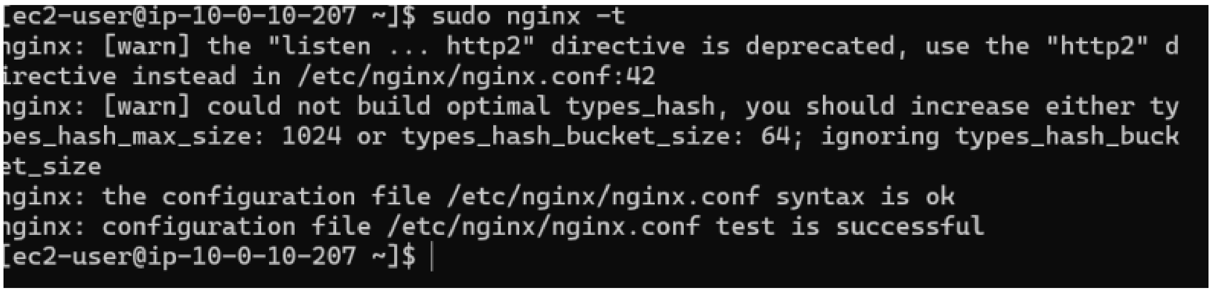
## Updated Nginx Configuration



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

upstream backend_servers {
    server 10.0.10.44:80;
    server 10.0.10.162:80;
    server 10.0.10.35:80 backup;
}

server {
     listen 443 ssl http2;
     server_name _;

     ssl_certificate /etc/ssl/certs/selfsigned.cr
```

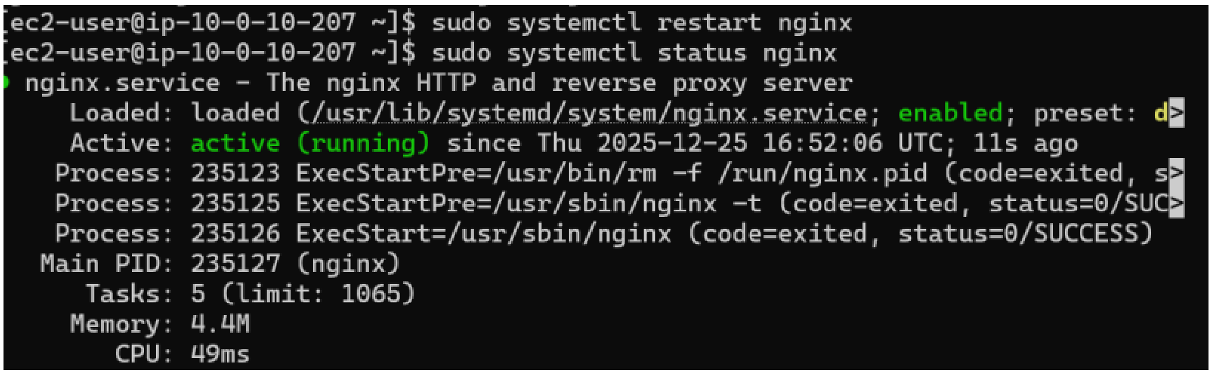
ginx_conf_updated.png

## Nginx Configuration Test



```
[ec2-user@ip-10-0-10-207 ~]$ sudo nginx -t
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" d
irective instead in /etc/nginx/nginx.conf:42
nginx: [warn] could not build optimal types_hash, you should increase either ty
pes_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_buck
et_size
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-207 ~]$ |
```

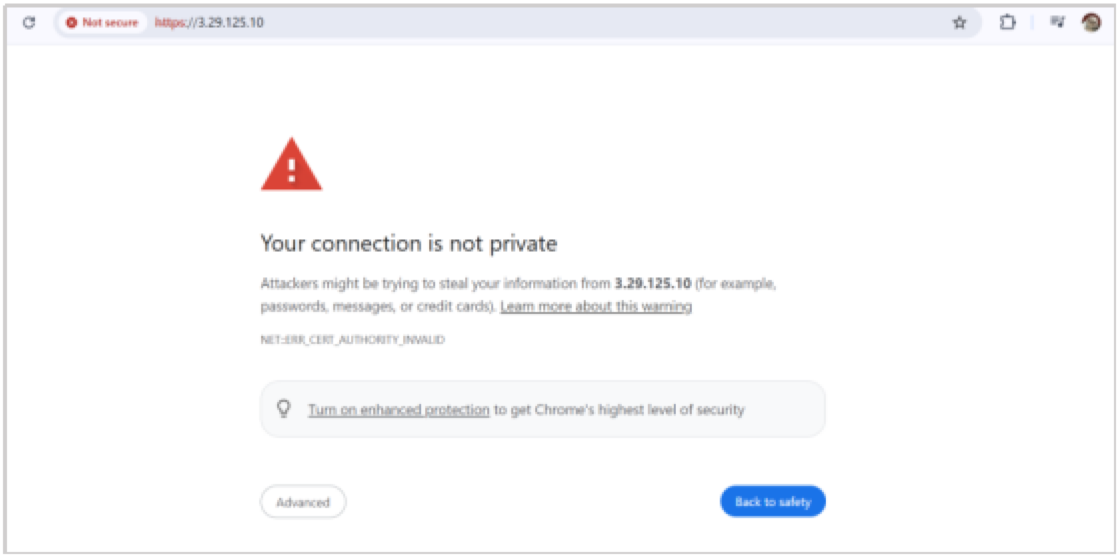
## Nginx Restart



```
[ec2-user@ip-10-0-10-207 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-207 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: d>
     Active: active (running) since Thu 2025-12-25 16:52:06 UTC; 11s ago
    Process: 235123 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, s>
    Process: 235125 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUC>
    Process: 235126 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 235127 (nginx)
      Tasks: 5 (limit: 1065)
     Memory: 4.4M
        CPU: 49ms
```

## 5.2 Test Load Balancing

### SSL Certificate Warning



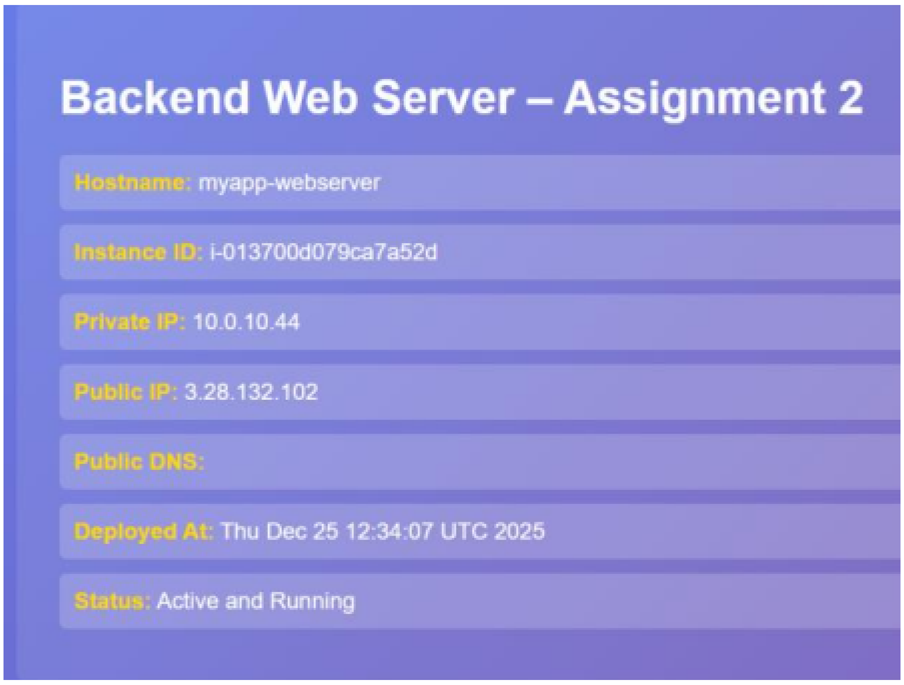
Your connection is not private

Attackers might be trying to steal your information from **3.29.125.10** (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

### Web-1 Response



Backend Web Server – Assignment 2

Hostname: myapp-webserver

Instance ID: i-013700d079ca7a52d

Private IP: 10.0.10.44

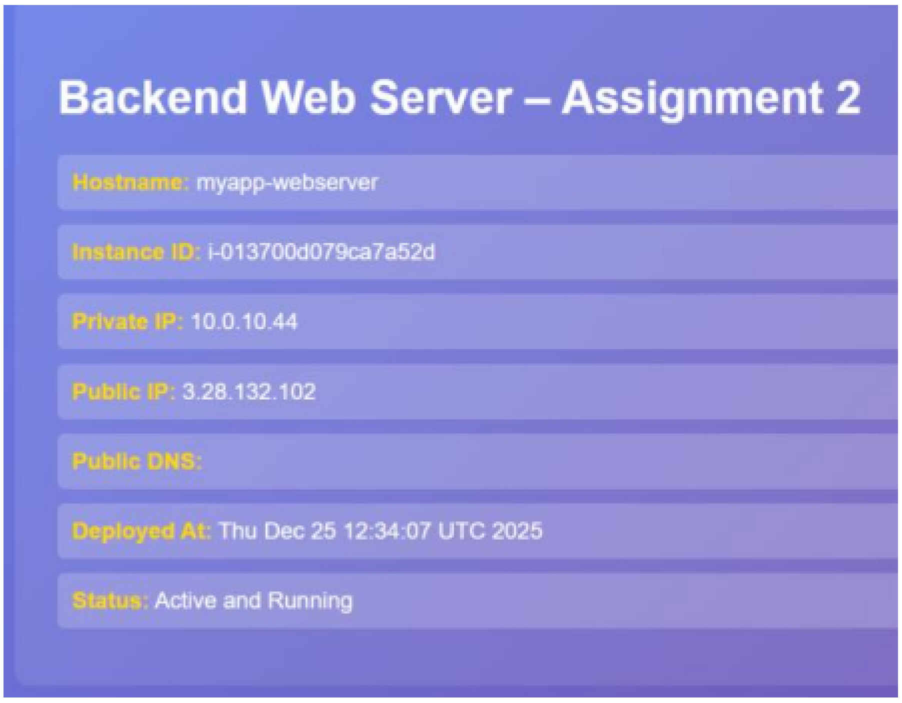
Public IP: 3.28.132.102

Public DNS:

Deployed At: Thu Dec 25 12:34:07 UTC 2025

Status: Active and Running

**Web-2 Response**



Backend Web Server – Assignment 2

Hostname: myapp-webserver

Instance ID: i-013700d079ca7a52d

Private IP: 10.0.10.44

Public IP: 3.28.132.102

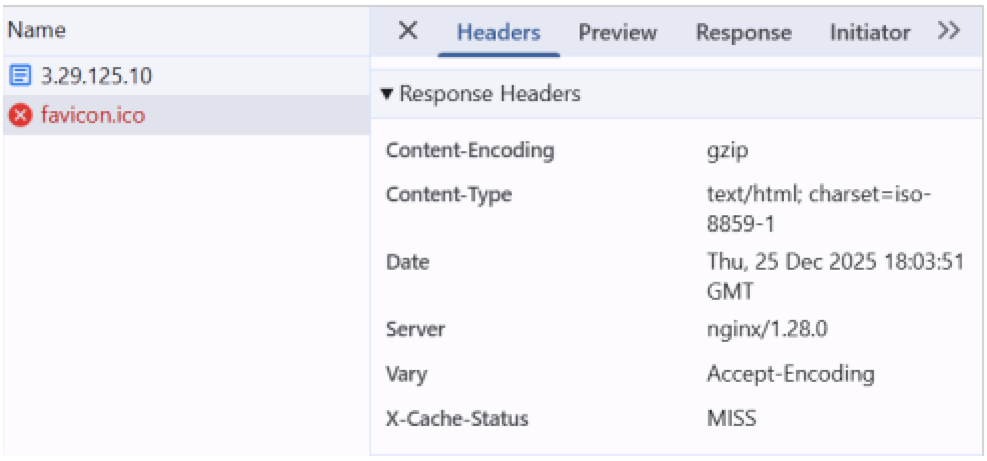
Public DNS:

Deployed At: Thu Dec 25 12:34:07 UTC 2025

Status: Active and Running

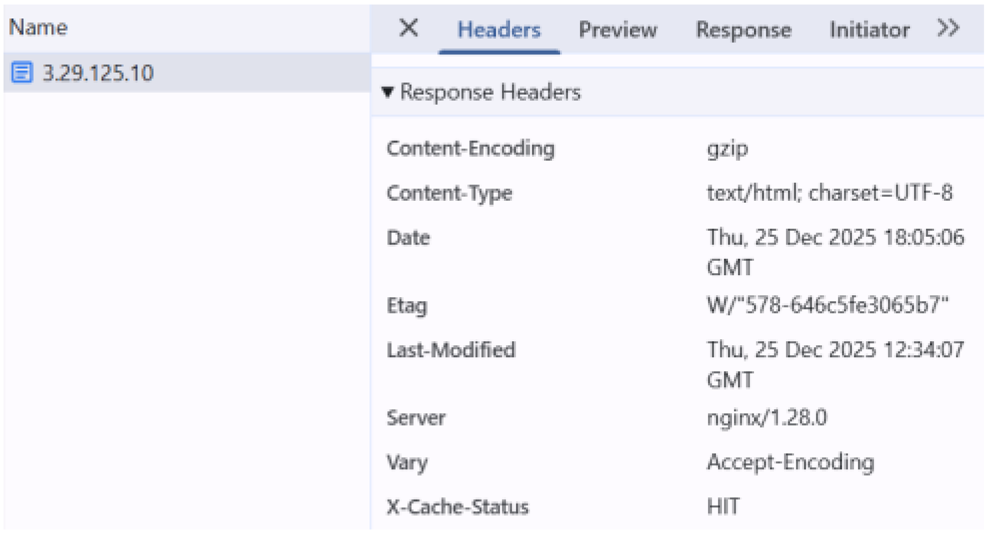
## 5.3 Test Cache Functionality

### Cache MISS Verification



| | |
|---|---|
| Content-Encoding | gzip |
| Content-Type | text/html; charset=iso-8859-1 |
| Date | Thu, 25 Dec 2025 18:03:51 GMT |
| Server | nginx/1.28.0 |
| Vary | Accept-Encoding |
| X-Cache-Status | MISS |

### Cache HIT Verification

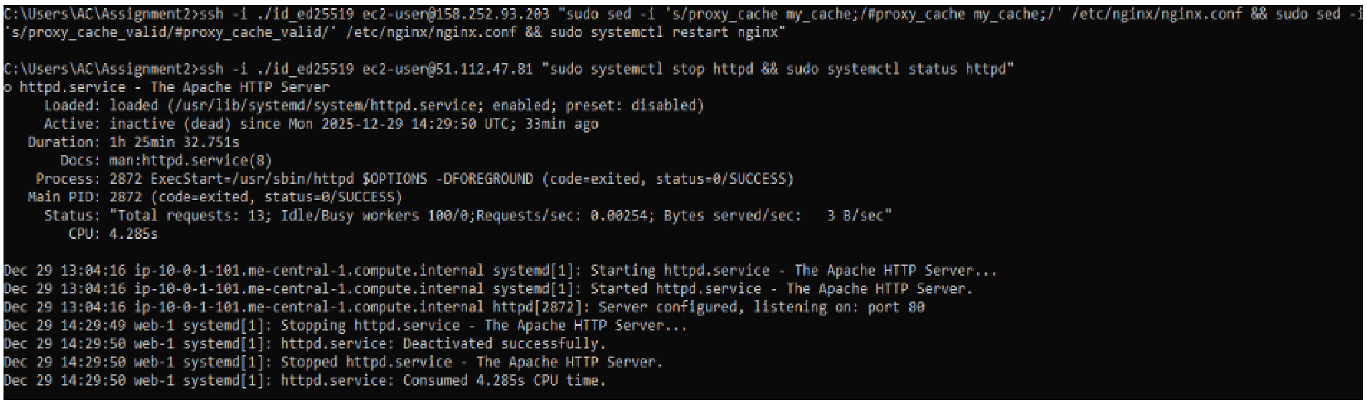


3.29.125.10

▼ Response Headers

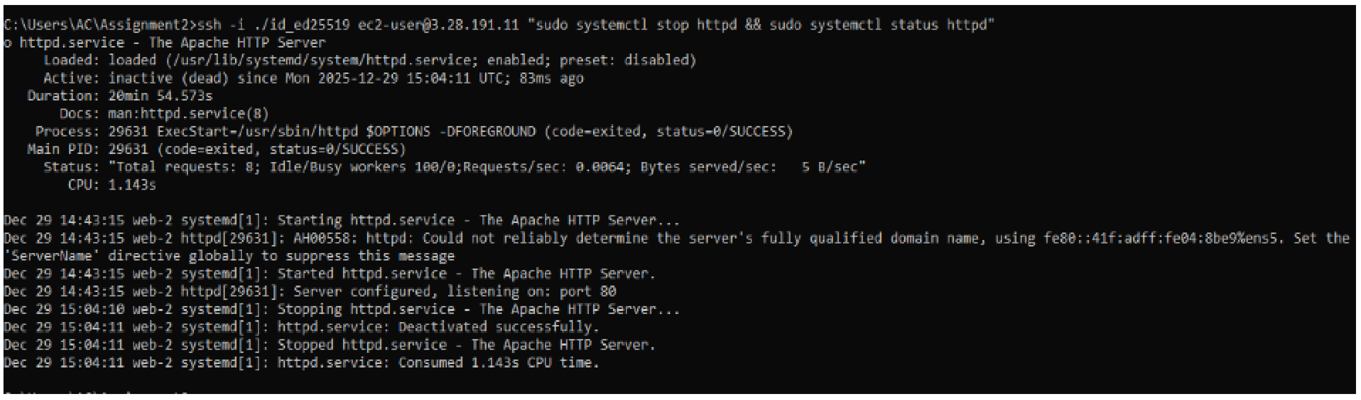
| | |
|---|---|
| Content-Encoding | gzip |
| Content-Type | text/html; charset=UTF-8 |
| Date | Thu, 25 Dec 2025 18:05:06 GMT |
| Etag | W/"578-646c5fe3065b7" |
| Last-Modified | Thu, 25 Dec 2025 12:34:07 GMT |
| Server | nginx/1.28.0 |
| Vary | Accept-Encoding |
| X-Cache-Status | HIT |

### Nginx Cache Directory



```
C:\Users\AC\Assignment2>ssh -i ./id_ed25519 ec2-user@158.252.93.203 "sudo sed -i 's/proxy_cache my_cache;/#proxy_cache my_cache;/' /etc/nginx/nginx.conf && sudo sed -i
's/proxy_cache_valid/#proxy_cache_valid/' /etc/nginx/nginx.conf && sudo systemctl restart nginx"

C:\Users\AC\Assignment2>ssh -i ./id_ed25519 ec2-user@51.112.47.81 "sudo systemctl stop httpd && sudo systemctl status httpd"
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: inactive (dead) since Mon 2025-12-29 14:29:50 UTC; 33min ago
   Duration: 1h 25min 32.751s
       Docs: man:httpd.service(8)
    Process: 2872 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 2872 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 13; Idle/Busy workers 100/0;Requests/sec: 0.00254; Bytes served/sec:   3 B/sec"
        CPU: 4.285s

Dec 29 13:04:16 ip-10-0-1-101.me-central-1.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Dec 29 13:04:16 ip-10-0-1-101.me-central-1.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Dec 29 13:04:16 ip-10-0-1-101.me-central-1.compute.internal httpd[2872]: Server configured, listening on: port 80
Dec 29 14:29:49 web-1 systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Dec 29 14:29:50 web-1 systemd[1]: httpd.service: Deactivated successfully.
Dec 29 14:29:50 web-1 systemd[1]: Stopped httpd.service - The Apache HTTP Server.
Dec 29 14:29:50 web-1 systemd[1]: httpd.service: Consumed 4.285s CPU time.
```

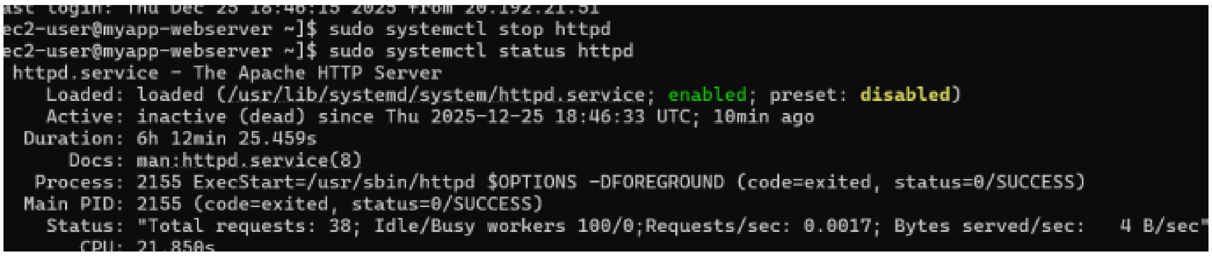
### Access Log Cache Status



```
C:\Users\AC\Assignment2>ssh -i ./id_ed25519 ec2-user@3.28.191.11 "sudo systemctl stop httpd && sudo systemctl status httpd"
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: inactive (dead) since Mon 2025-12-29 15:04:11 UTC; 83ms ago
   Duration: 20min 54.573s
       Docs: man:httpd.service(8)
    Process: 29631 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 29631 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 8; Idle/Busy workers 100/0;Requests/sec: 0.0064; Bytes served/sec:   5 B/sec"
        CPU: 1.143s

Dec 29 14:43:15 web-2 systemd[1]: Starting httpd.service - The Apache HTTP Server...
Dec 29 14:43:15 web-2 httpd[29631]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using fe80::41f:adff:fe04:8be9%ens5. Set the
'ServerName' directive globally to suppress this message
Dec 29 14:43:15 web-2 systemd[1]: Started httpd.service - The Apache HTTP Server.
Dec 29 14:43:15 web-2 httpd[29631]: Server configured, listening on: port 80
Dec 29 15:04:10 web-2 systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Dec 29 15:04:11 web-2 systemd[1]: httpd.service: Deactivated successfully.
Dec 29 15:04:11 web-2 systemd[1]: Stopped httpd.service - The Apache HTTP Server.
Dec 29 15:04:11 web-2 systemd[1]: httpd.service: Consumed 1.143s CPU time.
```

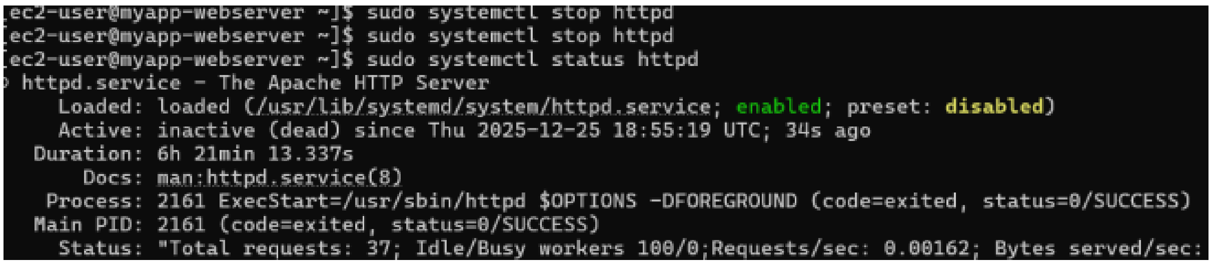
## 5.4 Test High Availability (Backup Server)

### Web-1 Service Stopped



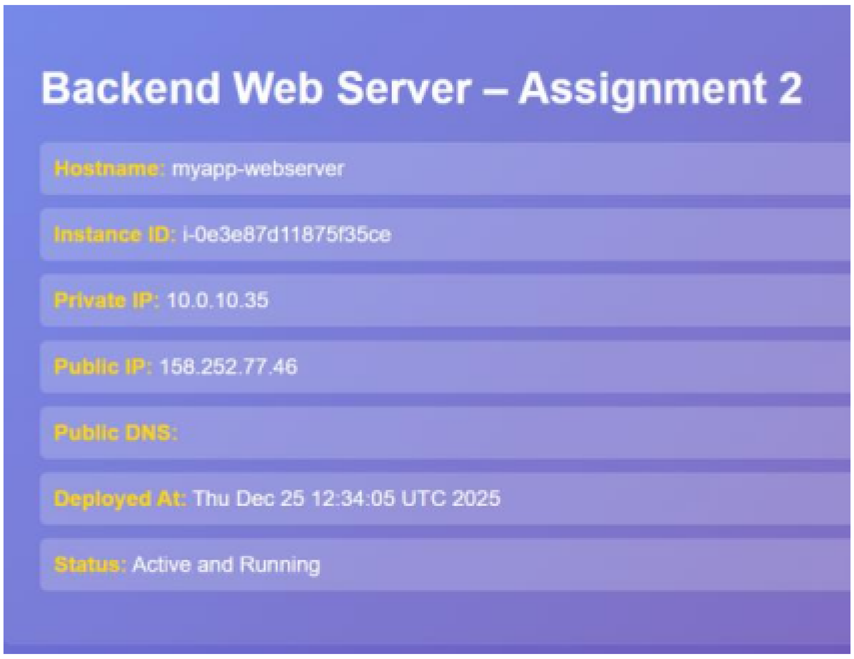
```
ast login: Thu Dec 25 18:46:15 2025 from 20.192.21.51
ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
 httpd.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Active: inactive (dead) since Thu 2025-12-25 18:46:33 UTC; 10min ago
  Duration: 6h 12min 25.459s
      Docs: man:httpd.service(8)
   Process: 2155 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
  Main PID: 2155 (code=exited, status=0/SUCCESS)
    Status: "Total requests: 38; Idle/Busy workers 100/0;Requests/sec: 0.0017; Bytes served/sec:   4 B/sec"
       CPU: 21.850s
```

**Web-2 Service Stopped**



```
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: inactive (dead) since Thu 2025-12-25 18:55:19 UTC; 34s ago
   Duration: 6h 21min 13.337s
       Docs: man:httpd.service(8)
    Process: 2161 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 2161 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 37; Idle/Busy workers 100/0;Requests/sec: 0.00162; Bytes served/sec:
```

**Backup Server Activated**



Backend Web Server – Assignment 2

Hostname: myapp-webserver

Instance ID: i-0e3e87d11875f35ce

Private IP: 10.0.10.35

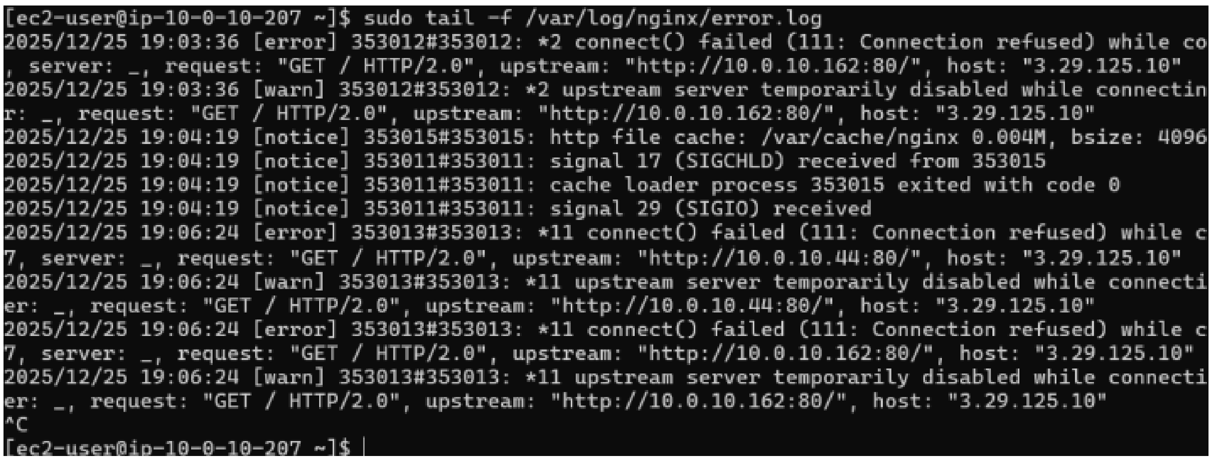
Public IP: 158.252.77.46

Public DNS:

Deployed At: Thu Dec 25 12:34:05 UTC 2025

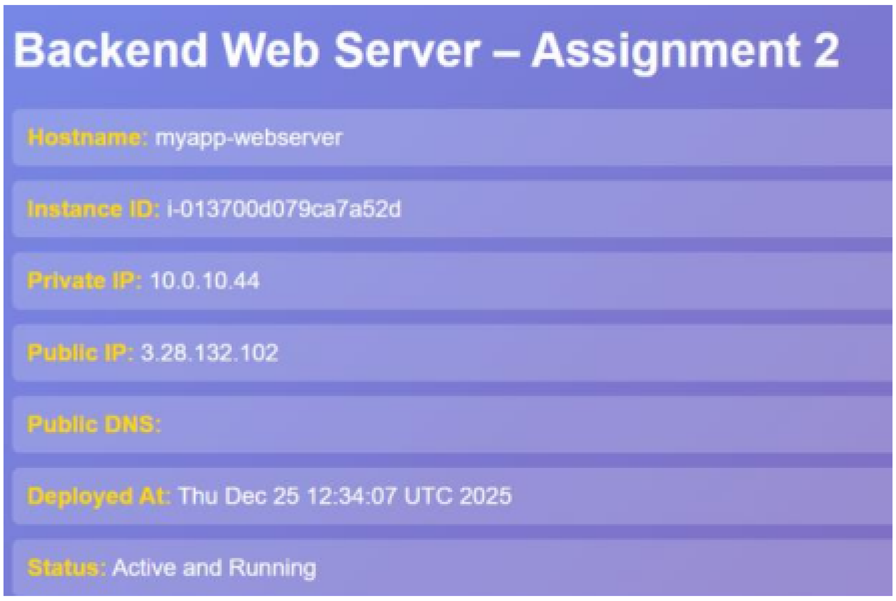
Status: Active and Running

**Nginx Error Log**



```
[ec2-user@ip-10-0-10-207 ~]$ sudo tail -f /var/log/nginx/error.log
2025/12/25 19:03:36 [error] 353012#353012: *2 connect() failed (111: Connection refused) while co
, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.162:80/", host: "3.29.125.10"
2025/12/25 19:03:36 [warn] 353012#353012: *2 upstream server temporarily disabled while connectin
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.162:80/", host: "3.29.125.10"
2025/12/25 19:04:19 [notice] 353015#353015: http file cache: /var/cache/nginx 0.004M, bsize: 4096
2025/12/25 19:04:19 [notice] 353011#353011: signal 17 (SIGCHLD) received from 353015
2025/12/25 19:04:19 [notice] 353011#353011: cache loader process 353015 exited with code 0
2025/12/25 19:04:19 [notice] 353011#353011: signal 29 (SIGIO) received
2025/12/25 19:06:24 [error] 353013#353013: *11 connect() failed (111: Connection refused) while c
7, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.44:80/", host: "3.29.125.10"
2025/12/25 19:06:24 [warn] 353013#353013: *11 upstream server temporarily disabled while connecti
er: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.44:80/", host: "3.29.125.10"
2025/12/25 19:06:24 [error] 353013#353013: *11 connect() failed (111: Connection refused) while c
7, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.162:80/", host: "3.29.125.10"
2025/12/25 19:06:24 [warn] 353013#353013: *11 upstream server temporarily disabled while connecti
er: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.162:80/", host: "3.29.125.10"
^C
[ec2-user@ip-10-0-10-207 ~]$ |
```

**Services Restored**



Backend Web Server – Assignment 2

Hostname: myapp-webserver

Instance ID: i-013700d079ca7a52d

Private IP: 10.0.10.44

Public IP: 3.28.132.102

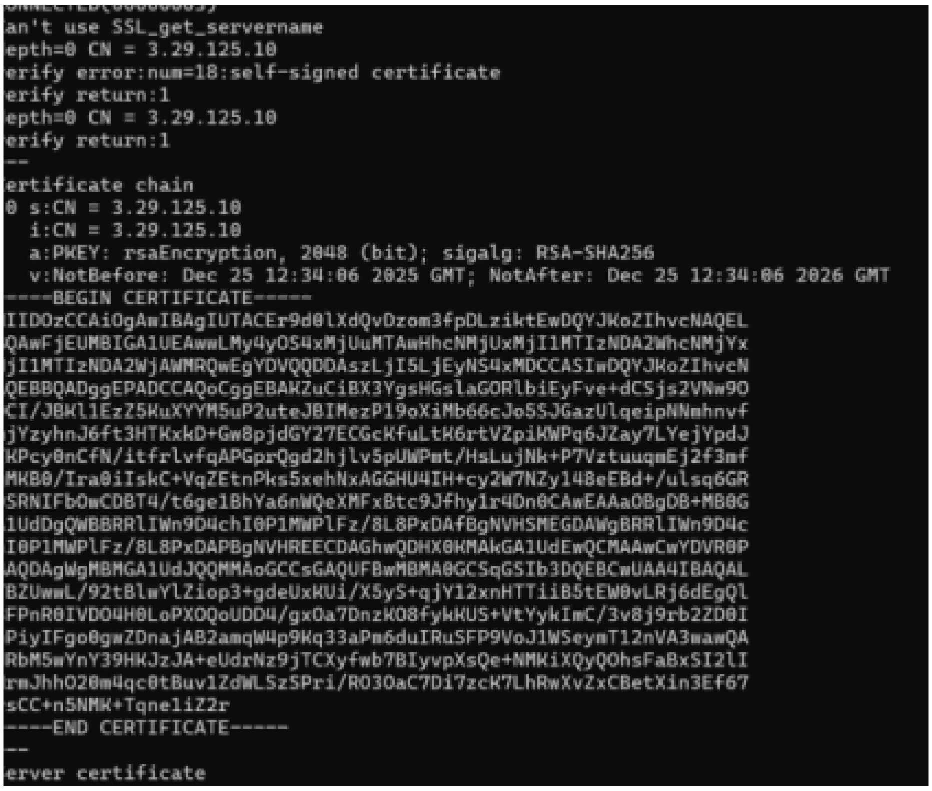
Public DNS:

Deployed At: Thu Dec 25 12:34:07 UTC 2025

Status: Active and Running

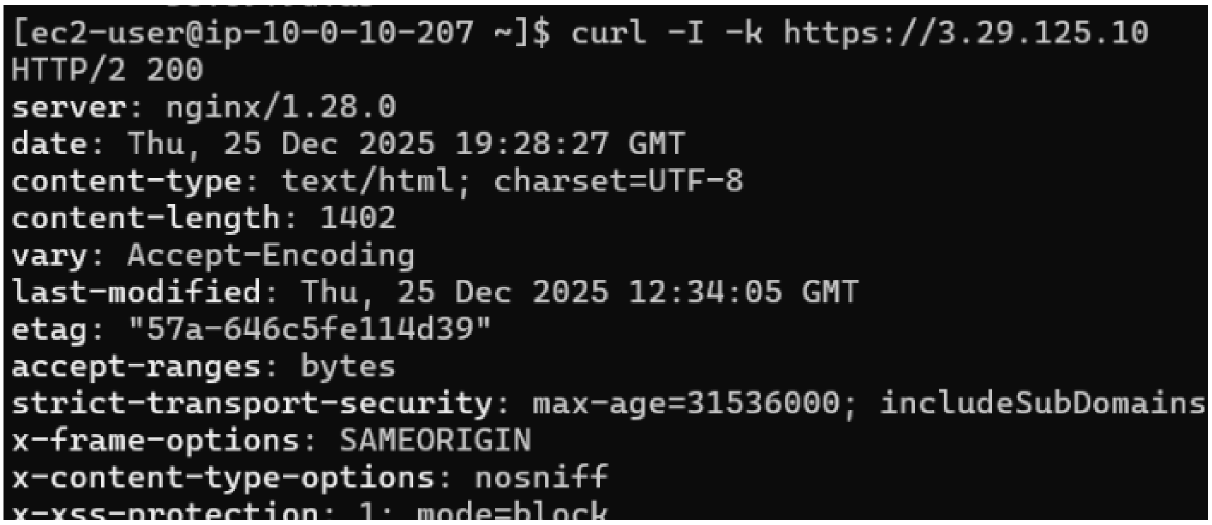
## 5.5 Security & Performance Analysis

### SSL Certificate Details



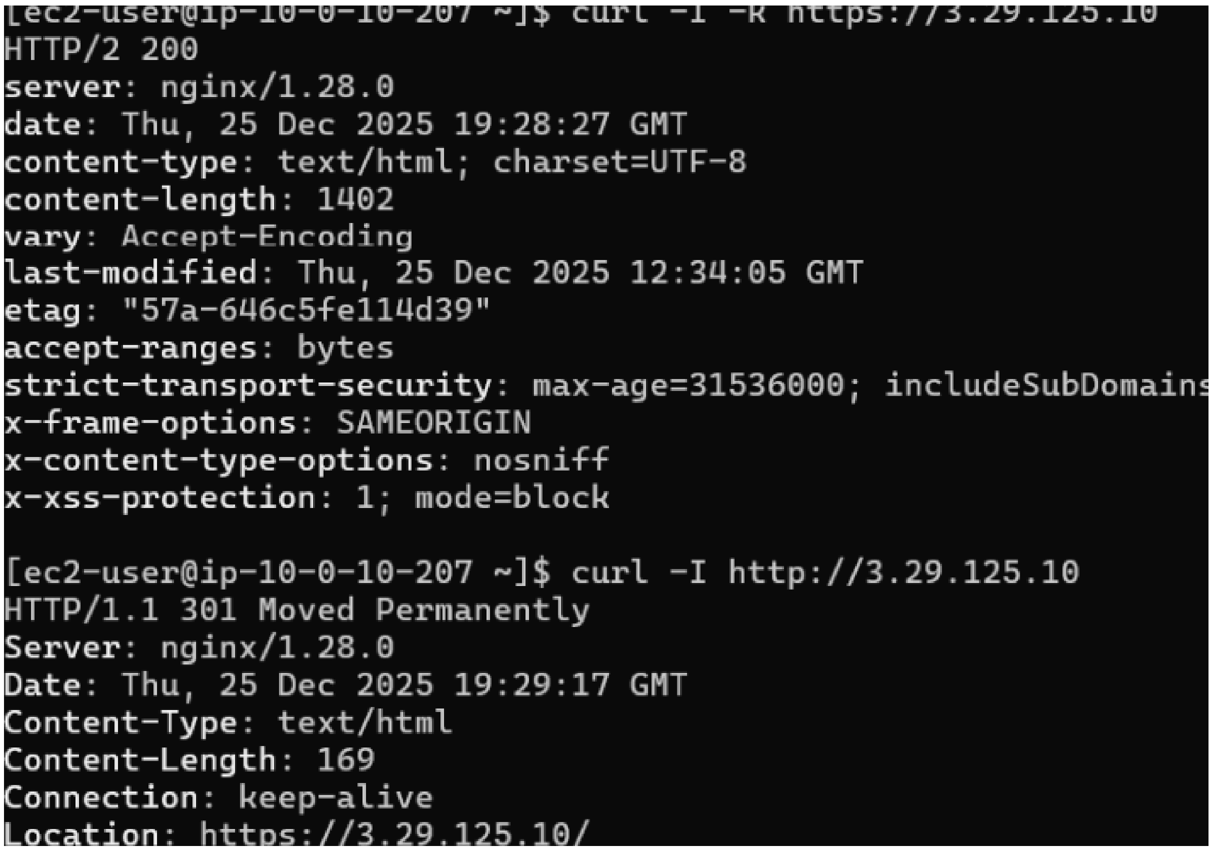
```
an't use SSL_get_servername
epth=0 CN = 3.29.125.10
erify error:num=18:self-signed certificate
erify return:1
epth=0 CN = 3.29.125.10
erify return:1
--
ertificate chain
0 s:CN = 3.29.125.10
  i:CN = 3.29.125.10
  a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Dec 25 12:34:06 2025 GMT; NotAfter: Dec 25 12:34:06 2026 GMT
----BEGIN CERTIFICATE-----
IIDOzCCAiOgAwIBAgIUTACEr9d0lXdQvDzom3fpDLziktEwDQYJKoZIhvcNAQEL
QAwFjEUMBIGA1UEAwwLMy4yOS4xMjUuMTAwHhcNMjUxMjI1MTIzNDA2WhcNMjYx
jI1MTIzNDA2WjAWMRQwEgYDVQQDDAszLjI5LjEyNS4xMDCCASIwDQYJKoZIhvcN
QEBBQADggEPADCCAQoCggEBAKZuCiBX3YgsHGslaGORlbiEyFve+dCSjs2VNw9O
CI/JBKl1EzZ5KuXYYM5uP2uteJBIMezP19oXiMb66cJo5SJGazUlqeipNNmhnvf
jYzyhnJ6ft3HTKxkD+Gw8pjdGY27ECGcKfuLtK6rtVZpiKWPq6JZay7LYejYpdJ
KPcy0nCfN/itfrlvfqAPGprQgd2hjlv5pUWPmt/HsLujNk+P7VztuuqmEj2f3mf
MKB0/Ira0iIskC+VqZEtnPks5xehNxAGGHU4IH+cy2W7NZy148eEBd+/ulsq6GR
SRNIFbOwCDBT4/t6ge1BhYa6nWQeXMFx8tc9Jfhy1r4Dn0CAwEAAaOBgDB+MB0G
1UdDgQWBBRRlIWn9D4chI0P1MWPlFz/8L8PxDAfBgNVHSMEGDAWgBRRlIWn9D4c
I0P1MWPlFz/8L8PxDAPBgNVHREECDAGhwQDHX0KMAkGA1UdEwQCMAAwCwYDVR0P
AQDAgWgMBMGA1UdJQQMMAoGCCsGAQUFBwMBMA0GCSqGSIb3DQEBCwUAA4IBAQAL
BZUwwL/92tBlwYlZiop3+gdeUxKUi/X5yS+qjY12xnHTTiiB5tEW0vLRj6dEgQl
FPnR0IVDO4H0LoPXOQoUDD4/gxOa7DnzKO8fykKUS+VtYykImC/3v8j9rb2ZD0I
PiyIFgo0gwZDnajAB2amqW4p9Kq33aPm6duIRuSFP9VoJ1WSeymT12nVA3wawQA
RbM5wYnY39HKJzJA+eUdrNz9jTCXyfwb7BIyvpXsQe+NMKiXQyQOhsFaBxSI2lI
rmJhhO20m4qc0tBuv1ZdWLSzSPri/RO3OaC7Di7zcK7LhRwXvZxCBetXin3Ef67
sCC+n5NMK+Tqne1iZ2r
----END CERTIFICATE-----
--
erver certificate
```

**Security Headers Verification**



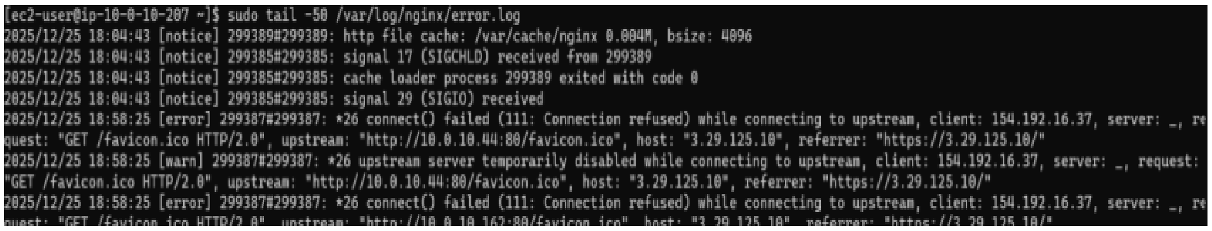
```
[ec2-user@ip-10-0-10-207 ~]$ curl -I -k https://3.29.125.10
HTTP/2 200
server: nginx/1.28.0
date: Thu, 25 Dec 2025 19:28:27 GMT
content-type: text/html; charset=UTF-8
content-length: 1402
vary: Accept-Encoding
last-modified: Thu, 25 Dec 2025 12:34:05 GMT
etag: "57a-646c5fe114d39"
accept-ranges: bytes
strict-transport-security: max-age=31536000; includeSubDomains
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
```

**HTTP to HTTPS Redirect**



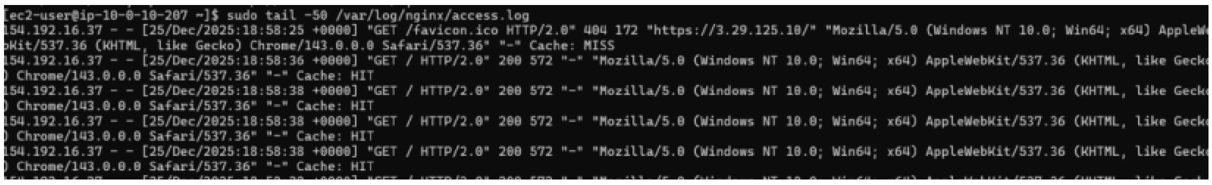
```
[ec2-user@ip-10-0-10-207 ~]$ curl -I -k https://3.29.125.10
HTTP/2 200
server: nginx/1.28.0
date: Thu, 25 Dec 2025 19:28:27 GMT
content-type: text/html; charset=UTF-8
content-length: 1402
vary: Accept-Encoding
last-modified: Thu, 25 Dec 2025 12:34:05 GMT
etag: "57a-646c5fe114d39"
accept-ranges: bytes
strict-transport-security: max-age=31536000; includeSubDomains
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
x-xss-protection: 1; mode=block

[ec2-user@ip-10-0-10-207 ~]$ curl -I http://3.29.125.10
HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Thu, 25 Dec 2025 19:29:17 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://3.29.125.10/
```

### Error Log Analysis



```
[ec2-user@ip-10-0-10-207 ~]$ sudo tail -50 /var/log/nginx/error.log
2025/12/25 18:04:43 [notice] 299389#299389: http file cache: /var/cache/nginx 0.004M, bsize: 4096
2025/12/25 18:04:43 [notice] 299385#299385: signal 17 (SIGCHLD) received from 299389
2025/12/25 18:04:43 [notice] 299385#299385: cache loader process 299389 exited with code 0
2025/12/25 18:04:43 [notice] 299385#299385: signal 29 (SIGIO) received
2025/12/25 18:58:25 [error] 299387#299387: *26 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.16.37, server: _, re
quest: "GET /favicon.ico HTTP/2.0", upstream: "http://10.0.10.44:80/favicon.ico", host: "3.29.125.10", referrer: "https://3.29.125.10/"
2025/12/25 18:58:25 [warn] 299387#299387: *26 upstream server temporarily disabled while connecting to upstream, client: 154.192.16.37, server: _, request:
"GET /favicon.ico HTTP/2.0", upstream: "http://10.0.10.44:80/favicon.ico", host: "3.29.125.10", referrer: "https://3.29.125.10/"
2025/12/25 18:58:25 [error] 299387#299387: *26 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.16.37, server: _, re
quest: "GET /favicon.ico HTTP/2.0", upstream: "http://10.0.10.162:80/favicon.ico", host: "3.29.125.10", referrer: "https://3.29.125.10/"
```

### Access Log Analysis

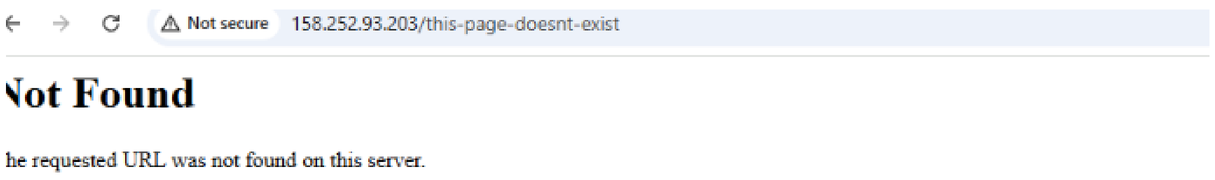


```
[ec2-user@ip-10-0-10-207 ~]$ sudo tail -50 /var/log/nginx/access.log
154.192.16.37 - - [25/Dec/2025:18:58:25 +0000] "GET /favicon.ico HTTP/2.0" 404 172 "https://3.29.125.10/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
bKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache: MISS
154.192.16.37 - - [25/Dec/2025:18:58:36 +0000] "GET / HTTP/2.0" 200 572 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/143.0.0.0 Safari/537.36" "-" Cache: HIT
154.192.16.37 - - [25/Dec/2025:18:58:38 +0000] "GET / HTTP/2.0" 200 572 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/143.0.0.0 Safari/537.36" "-" Cache: HIT
154.192.16.37 - - [25/Dec/2025:18:58:38 +0000] "GET / HTTP/2.0" 200 572 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/143.0.0.0 Safari/537.36" "-" Cache: HIT
154.192.16.37 - - [25/Dec/2025:18:58:38 +0000] "GET / HTTP/2.0" 200 572 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/143.0.0.0 Safari/537.36" "-" Cache: HIT
```

## Bonus Tasks

### Bonus 1: Custom Error Pages

#### Custom 404 Error Page

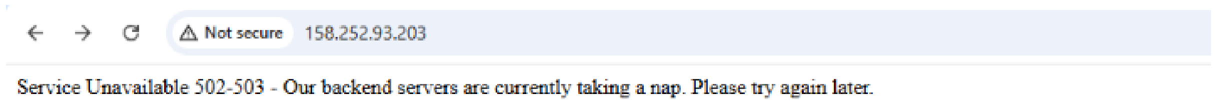


```
Not secure   158.252.93.203/this-page-doesnt-exist
```

Not Found

he requested URL was not found on this server.

### Custom 502 Error Page

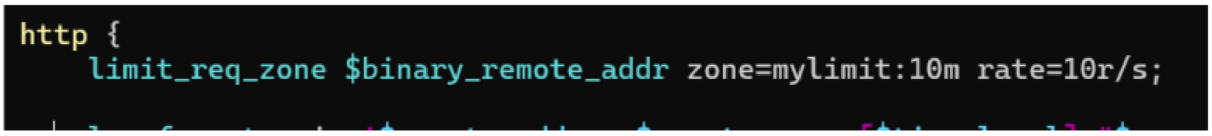


```
Not secure   158.252.93.203
```

Service Unavailable 502-503 - Our backend servers are currently taking a nap. Please try again later.

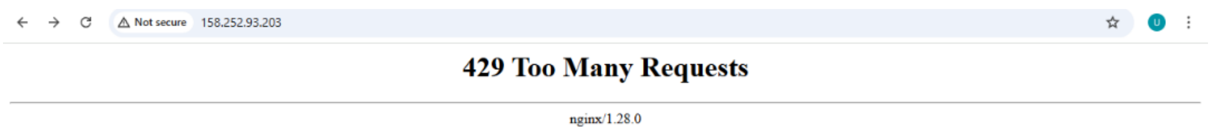
## Bonus 2: Implement Rate Limiting

### Rate Limiting Configuration



```
http {
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;
```

### 429 error



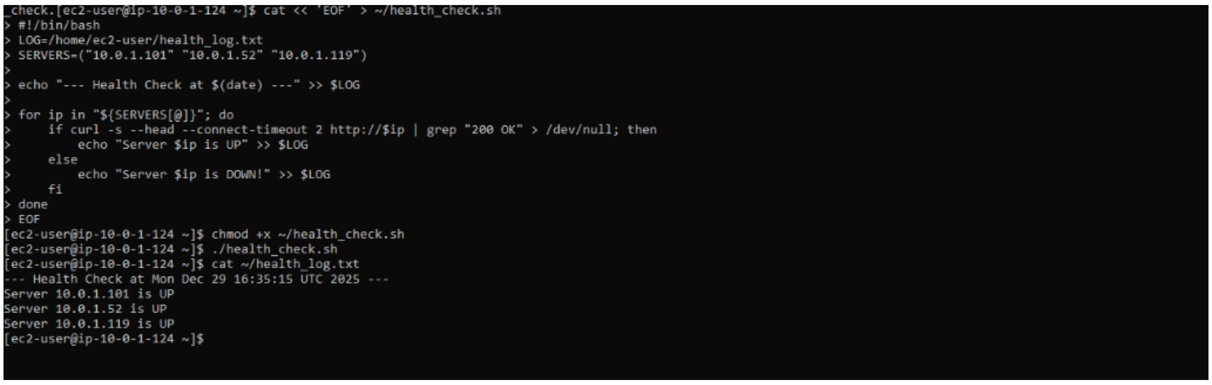
```
Not secure   158.252.93.203
```

429 Too Many Requests

nginx/1.28.0

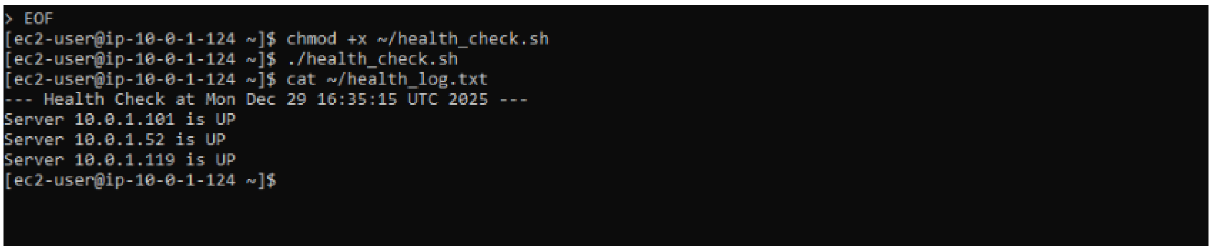
## Bonus 3: Health Check Automation

### Health Check Script



```
_check.[ec2-user@ip-10-0-1-124 ~]$ cat << 'EOF' > ~/health_check.sh
> #!/bin/bash
> LOG=/home/ec2-user/health_log.txt
> SERVERS=("10.0.1.101" "10.0.1.52" "10.0.1.119")
>
> echo "--- Health Check at $(date) ---" >> $LOG
>
> for ip in "${SERVERS[@]}"; do
>     if curl -s --head --connect-timeout 2 http://$ip | grep "200 OK" > /dev/null; then
>         echo "Server $ip is UP" >> $LOG
>     else
>         echo "Server $ip is DOWN!" >> $LOG
>     fi
> done
> EOF
[ec2-user@ip-10-0-1-124 ~]$ chmod +x ~/health_check.sh
[ec2-user@ip-10-0-1-124 ~]$ ./health_check.sh
[ec2-user@ip-10-0-1-124 ~]$ cat ~/health_log.txt
--- Health Check at Mon Dec 29 16:35:15 UTC 2025 ---
Server 10.0.1.101 is UP
Server 10.0.1.52 is UP
Server 10.0.1.119 is UP
[ec2-user@ip-10-0-1-124 ~]$
```

### Health Check Logs

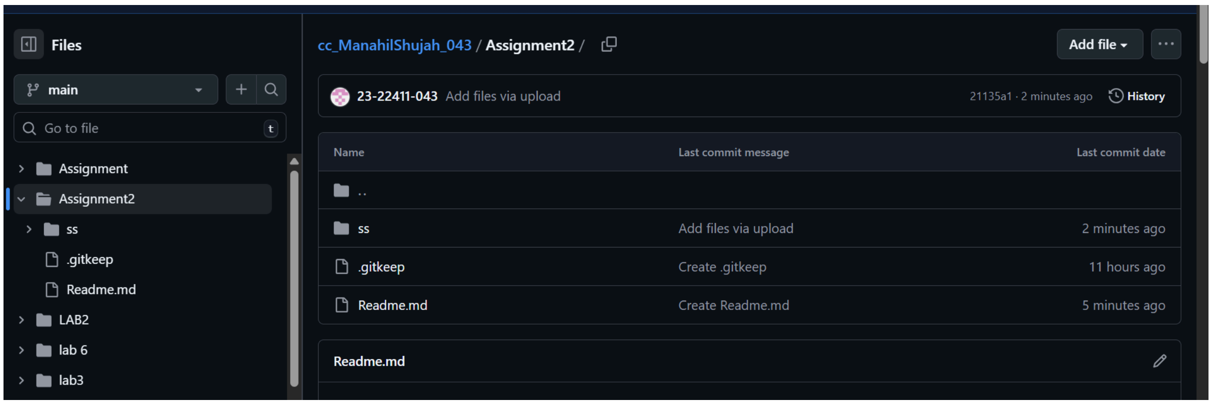


```
> EOF
[ec2-user@ip-10-0-1-124 ~]$ chmod +x ~/health_check.sh
[ec2-user@ip-10-0-1-124 ~]$ ./health_check.sh
[ec2-user@ip-10-0-1-124 ~]$ cat ~/health_log.txt
--- Health Check at Mon Dec 29 16:35:15 UTC 2025 ---
Server 10.0.1.101 is UP
Server 10.0.1.52 is UP
Server 10.0.1.119 is UP
[ec2-user@ip-10-0-1-124 ~]$
```

# Part 6: Documentation & Cleanup

## 6.1 README Documentation

### README File Overview

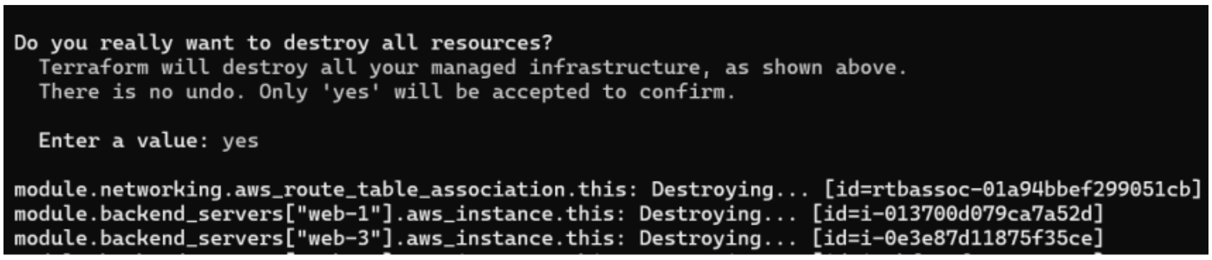


23-22411-043 Add files via upload 21135a1 · 2 minutes ago

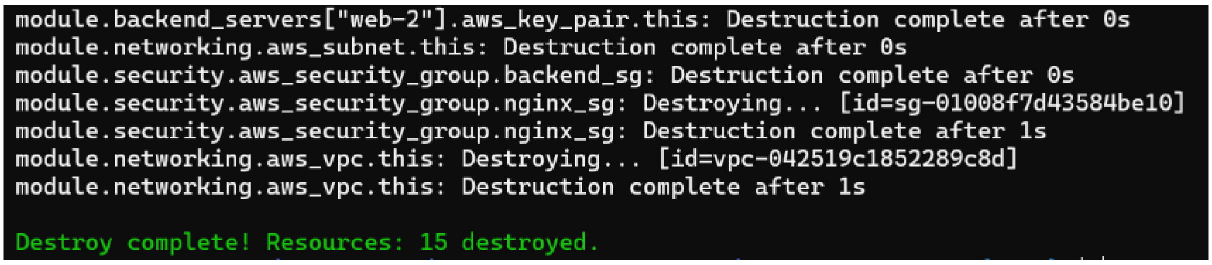
| Name | Last commit message | Last commit date |
|---|---|---|
| .. | | |
| ss | Add files via upload | 2 minutes ago |
| .gitkeep | Create .gitkeep | 11 hours ago |
| Readme.md | Create Readme.md | 5 minutes ago |

Readme.md

## 6.2 Infrastructure Cleanup



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

  Enter a value: yes

module.networking.aws_route_table_association.this: Destroying... [id=rtbassoc-01a94bbef299051cb]
module.backend_servers["web-1"].aws_instance.this: Destroying... [id=i-013700d079ca7a52d]
module.backend_servers["web-3"].aws_instance.this: Destroying... [id=i-0e3e87d11875f35ce]
```



```
module.backend_servers["web-2"].aws_key_pair.this: Destruction complete after 0s
module.networking.aws_subnet.this: Destruction complete after 0s
module.security.aws_security_group.backend_sg: Destruction complete after 0s
module.security.aws_security_group.nginx_sg: Destroying... [id=sg-01008f7d43584be10]
module.security.aws_security_group.nginx_sg: Destruction complete after 1s
module.networking.aws_vpc.this: Destroying... [id=vpc-042519c1852289c8d]
module.networking.aws_vpc.this: Destruction complete after 1s

Destroy complete! Resources: 15 destroyed.
```

## Common Issues and Solutions

- **SSH Access Fails:** Check security groups and key permissions.
- **Backend Not Responding:** Verify Apache service and backend IP configuration.
- **SSL Errors:** Confirm certificate paths and permissions.

## Conclusion

The project enhanced understanding of Terraform modules, AWS networking, and advanced Nginx configurations. The deployed infrastructure effectively supports load balancing, caching, and high availability.