FATIMA JINNAH WOMEN UNIVERSITY

Department of Software Engineering

Assignment 2

Advanced Terraform & Nginx Multi-Tier Architecture

Course Title:

Cloud Compting

Submitted to:

Mr. Waqas Saleem

Submitted by:

Rughma Malik

2023-BSE-054

V-B

Submission Date:

31-12-2025

Table of Contents

Advanced Terraform & Nginx Multi-Tier Architecture

1. Executive Summary

This assignment focuses on the design, deployment, and testing of a highly available multi-tier web infrastructure on Amazon Web Services (AWS) using Terraform as an Infrastructure as Code (IaC) tool.

The deployed infrastructure consists of a Virtual Private Cloud (VPC), a public subnet, security groups, three Apache backend web servers, and an Nginx reverse proxy acting as a load balancer. Secure communication is enforced using HTTPS with a self-signed SSL certificate, and performance is enhanced through Nginx caching.

Key achievements include:

- Automated infrastructure provisioning using Terraform
- Load balancing between multiple backend servers
- High availability using a backup web server
- Secure access with SSL/TLS and security headers
- Successful validation through extensive testing

This project demonstrates practical knowledge of cloud infrastructure, DevOps automation, and web server configuration.

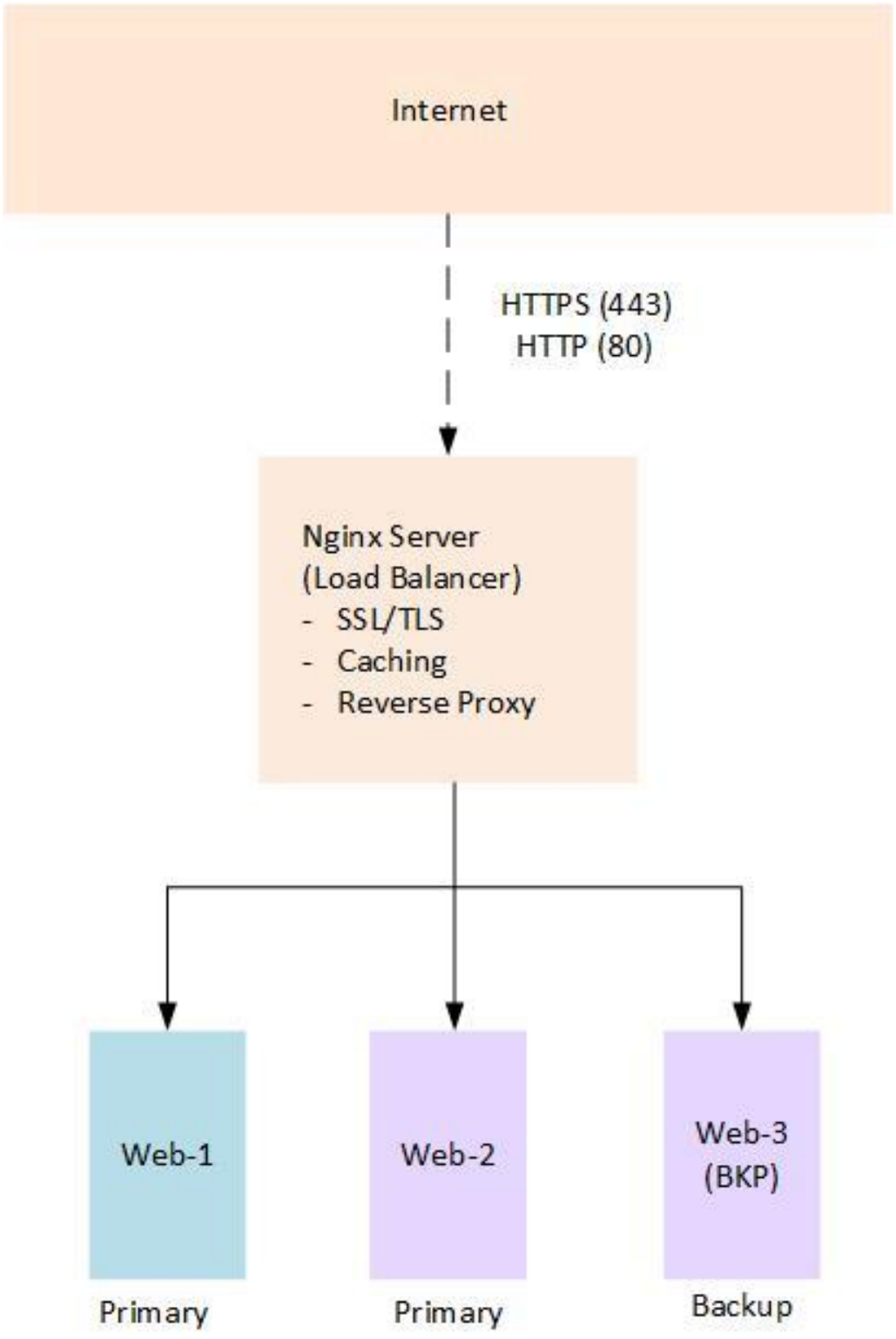
2. Architecture Design

2.1 Architecture Overview

The system follows a **three-tier architecture**:

- **Client Layer:** End users accessing the application via browser
- **Application Layer:** Nginx server acting as reverse proxy and load balancer
- **Backend Layer:** Apache web servers serving application content

2.2 Architecture Diagram



2.3 Network Topology

- Single VPC
- One public subnet
- Internet Gateway for external access
- Route table allowing outbound internet traffic

2.4 Security Design

- Separate security groups for Nginx and backend servers
- Restricted backend access (HTTP allowed only from Nginx)
- SSH access enabled for management
- HTTPS enforced using SSL/TLS

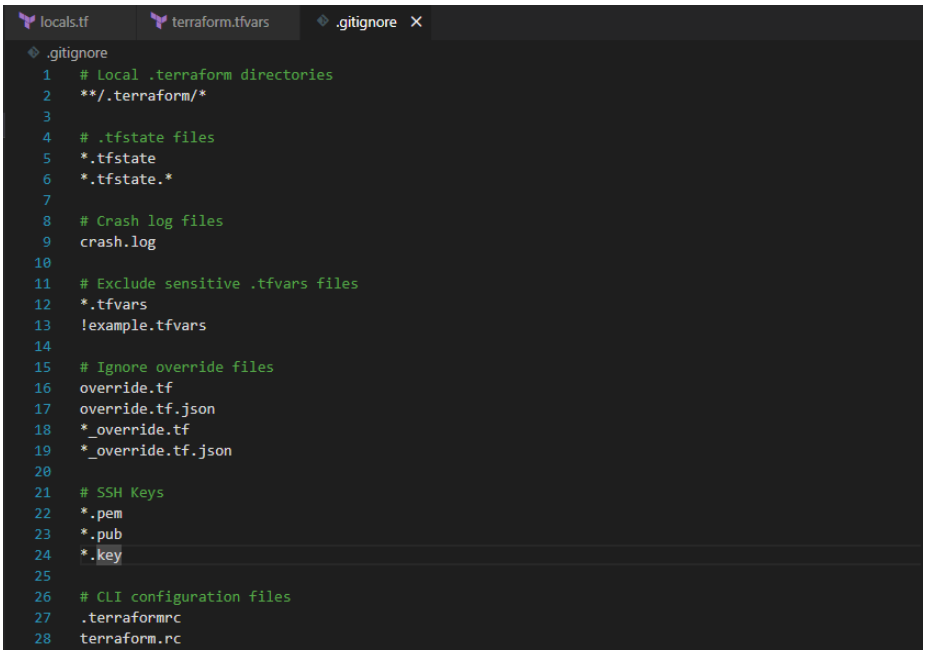
3. Implementation Details

Part 1: Infrastructure Setup (25 marks)

1.1 Project Structure (5 marks)

Tasks:

- Create all necessary files and directories
- Implement proper .gitignore to exclude sensitive files
- Document the project structure in README.md



```
PS D:\docs\Assignment2> tree /F
Volume serial number is EE83-994E
    .gitignore
    locals.tf
    main.tf
    outputs.tf
    README.md
    terraform.tfvars
    variables.tf

├───docs
├───modules
    ├───networking
            main.tf
            outputs.tf
            variables.tf

    ├───security
            main.tf
            outputs.tf
            variables.tf

    └───webserver
            main.tf
            outputs.tf
            variables.tf

├───screenshots
    ├───bonus
    ├───part1
    ├───part2
    ├───part3
    ├───part4
    ├───part5
    └───part6
└───scripts
        apache-setup.sh
        nginx-setup.sh
```

Ln 24, Col 6 Spaces: 4 UTF-8 CRLF

1.2 Variable Configuration (5 marks)

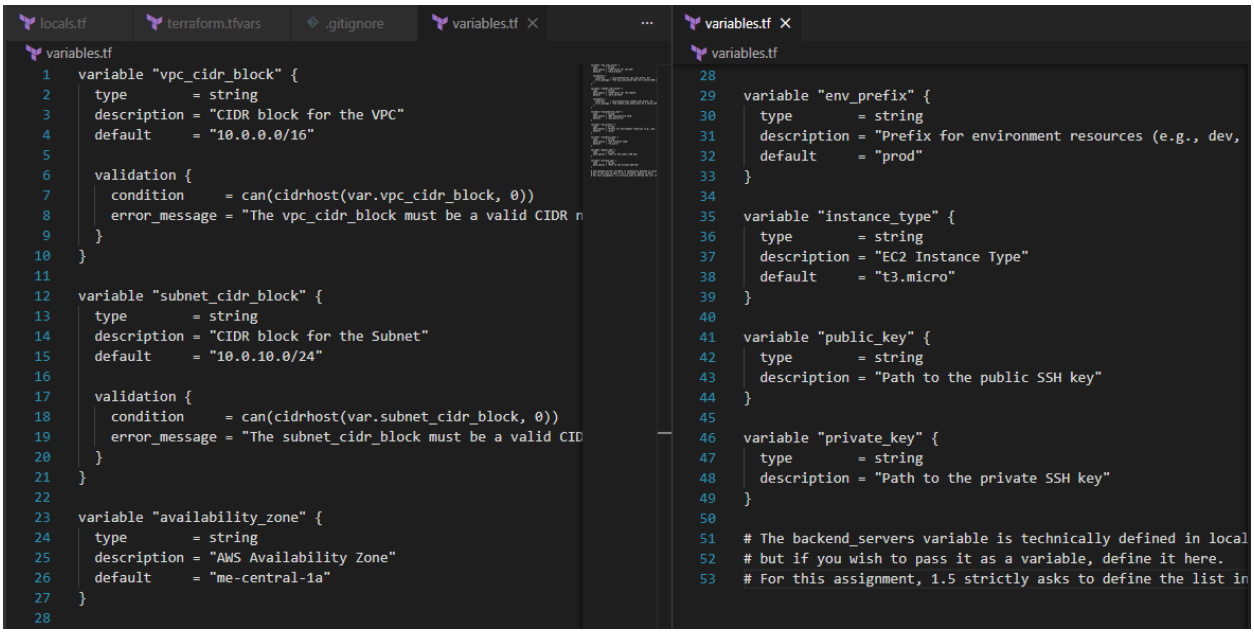
Tasks:

- Add validation rules for CIDR blocks
- Add descriptions for all variables
- Set appropriate defaults where applicable
- Populate terraform.tfvars with your values

locals.tf terraform.tfvars

terraform.tfvars

```
1   vpc_cidr_block       = "10.0.0.0/16"
2   subnet_cidr_block    = "10.0.10.0/24"
3   availability_zone    = "us-east-1a"
4   env_prefix           = "prod"
5   instance_type        = "t3.micro"
6
7   public_key_location  = "/c/Users/Rughma/.ssh/id_ed25519.pub"
8   private_key_location = "/c/Users/Rughma/.ssh/id_ed25519"
```



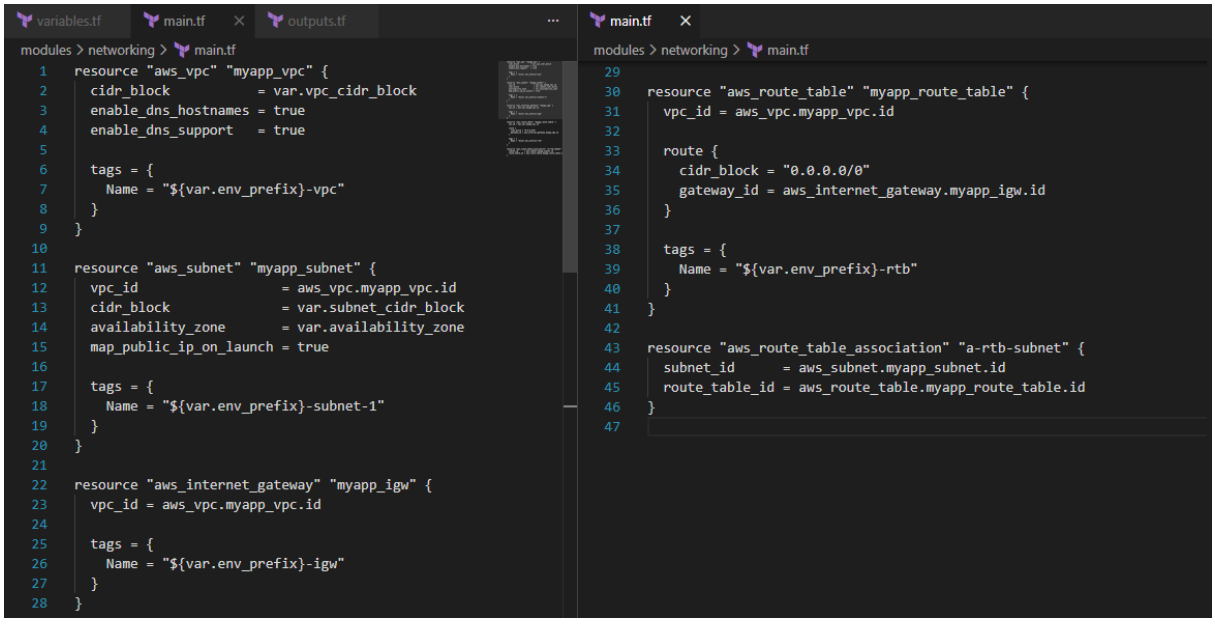
```
variable "vpc_cidr_block" {
  type        = string
  description = "CIDR block for the VPC"
  default     = "10.0.0.0/16"

  validation {
    condition     = can(cidrhost(var.vpc_cidr_block, 0))
    error_message = "The vpc_cidr_block must be a valid CIDR n
  }
}

variable "subnet_cidr_block" {
  type        = string
  description = "CIDR block for the Subnet"
  default     = "10.0.10.0/24"

  validation {
    condition     = can(cidrhost(var.subnet_cidr_block, 0))
    error_message = "The subnet_cidr_block must be a valid CID
  }
}

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

```
variable "env_prefix" {
  type        = string
  description = "Prefix for environment resources (e.g., dev,
  default     = "prod"
}

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

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

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

# The backend_servers variable is technically defined in local
# but if you wish to pass it as a variable, define it here.
# For this assignment, 1.5 strictly asks to define the list in
```

1.3 Networking Module (5 marks)

Tasks:

- Create VPC resource
- Create subnet with map_public_ip_on_launch = true
- Create and attach Internet Gateway
- Configure routing table
- Add proper tags to all resources using env_prefix



```
resource "aws_vpc" "myapp_vpc" {
  cidr_block           = var.vpc_cidr_block
  enable_dns_hostnames = true
  enable_dns_support   = true

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

resource "aws_subnet" "myapp_subnet" {
  vpc_id                  = aws_vpc.myapp_vpc.id
  cidr_block              = var.subnet_cidr_block
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.env_prefix}-subnet-1"
  }
}

resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = aws_vpc.myapp_vpc.id

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

```
resource "aws_route_table" "myapp_route_table" {
  vpc_id = aws_vpc.myapp_vpc.id

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

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

resource "aws_route_table_association" "a-rtb-subnet" {
  subnet_id      = aws_subnet.myapp_subnet.id
  route_table_id = aws_route_table.myapp_route_table.id
}
```

variables.tf | outputs.tf

modules > networking > outputs.tf

```hcl
output "vpc_id" {
  value = aws_vpc.myapp_vpc.id
}
output "subnet_id" {
  value = aws_subnet.myapp_subnet.id
}
output "igw_id" {
  value = aws_internet_gateway.myapp_igw.id
}
output "route_table_id" {
  value = aws_route_table.myapp_route_table.id
}
```

1.4 Security Module (5 marks)

Tasks:

- Create Nginx security group with appropriate rules
- Create backend security group with appropriate rules
- Use security group IDs for backend ingress (not CIDR blocks)
- Add descriptive names and tags

variables.tf | main.tf | outputs.tf

modules > security > main.tf

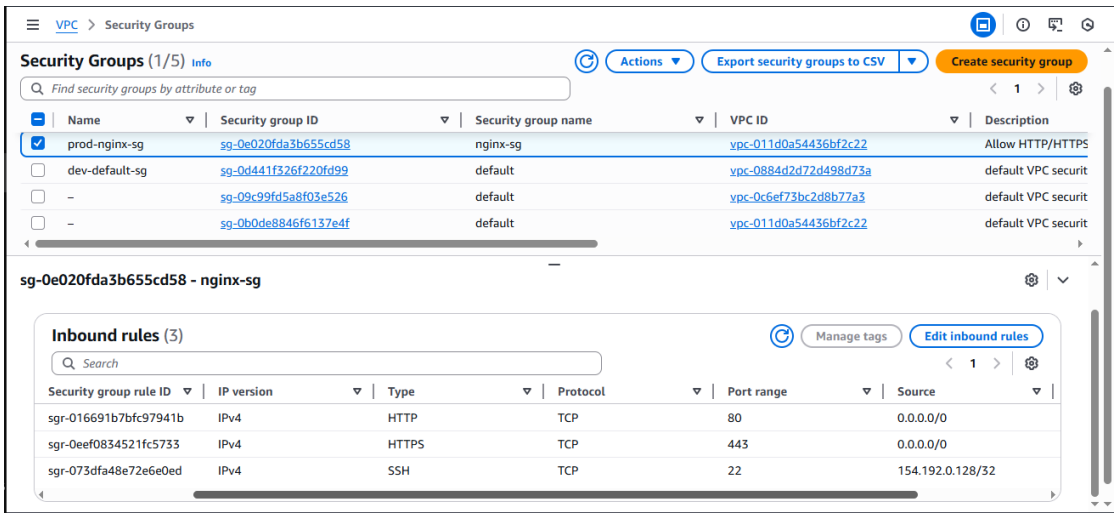
```hcl
# Nginx Security Group
resource "aws_security_group" "nginx_sg" {
  name        = "nginx-sg"
  description = "Allow SSH, HTTP, HTTPS"
  vpc_id      = var.vpc_id

  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"]
```

1.5 Locals Configuration (5 marks)

Tasks:

- Implement dynamic IP detection
- Define common tags
- Define backend server list
- Add any other reusable local values

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

locals {
  # Dynamically fetch current public IP and append CIDR mask
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"

  common_tags = {
    Environment = var.env_prefix
    Project     = "Assignment-2"
    ManagedBy   = "Terraform"
  }

  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"
    }
  ]
}
```

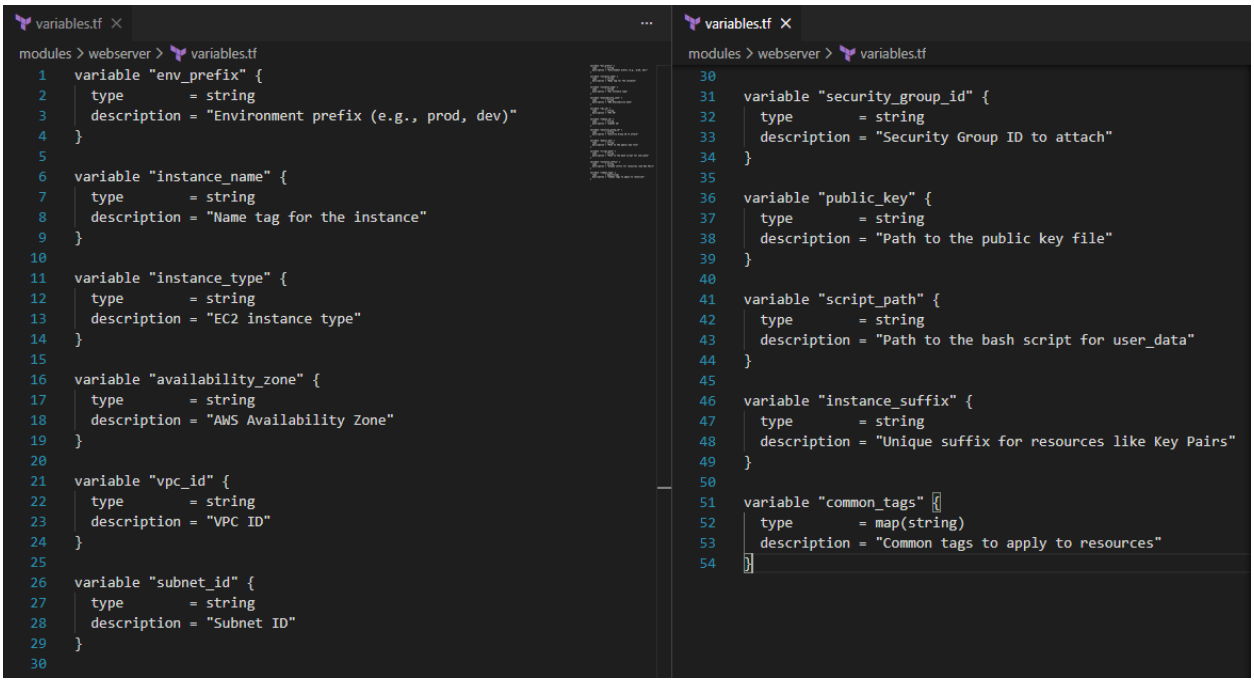


Part 2: Webserver Module (15 marks)

2.1 Module Design (10 marks)

Tasks:

- Create variables. tf with all required variables
- Create main.tf with key pair and instance resources
- Create outputs.tf with all required outputs
- Ensure module is reusable for both Nginx and backend servers



variables.tf

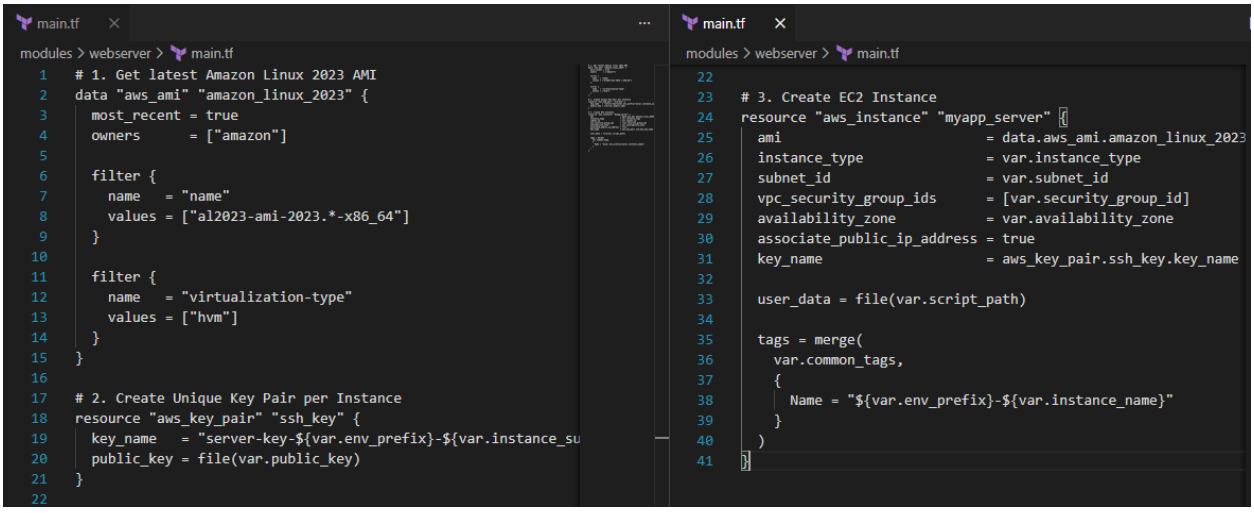
modules > webserver > variables.tf

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

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

variable "instance_type" {
  type        = string
  description = "EC2 instance type"
}

variable "availability_zone" {
  type        = string
  description = "AWS Availability Zone"
}

variable "vpc_id" {
  type        = string
  description = "VPC ID"
}

variable "subnet_id" {
  type        = string
  description = "Subnet ID"
}
```

variables.tf

modules > webserver > variables.tf

```
variable "security_group_id" {
  type        = string
  description = "Security Group ID to attach"
}

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

variable "script_path" {
  type        = string
  description = "Path to the bash script for user_data"
}

variable "instance_suffix" {
  type        = string
  description = "Unique suffix for resources like Key Pairs"
}

variable "common_tags" {
  type        = map(string)
  description = "Common tags to apply to resources"
}
```



main.tf

modules > webserver > main.tf

```
# 1. Get latest Amazon Linux 2023 AMI
data "aws_ami" "amazon_linux_2023" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["al2023-ami-2023.*-x86_64"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}

# 2. Create Unique Key Pair per Instance
resource "aws_key_pair" "ssh_key" {
  key_name   = "server-key-${var.env_prefix}-${var.instance_su
  public_key = file(var.public_key)
}
```

main.tf

modules > webserver > main.tf

```
# 3. Create EC2 Instance
resource "aws_instance" "myapp_server" {
  ami                         = data.aws_ami.amazon_linux_2023
  instance_type               = var.instance_type
  subnet_id                   = var.subnet_id
  vpc_security_group_ids      = [var.security_group_id]
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = aws_key_pair.ssh_key.key_name

  user_data = file(var.script_path)

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

main.tf outputs.tf

modules > webserver > outputs.tf

```hcl
output "instance_id" {
  value = aws_instance.myapp_server.id
}

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

output "private_ip" {
  value = aws_instance.myapp_server.private_ip
}
```

2.2 Module Usage (5 marks)

Tasks:

- Create Nginx server module instance
- Create backend server module instances using for_each
- Ensure proper security group assignment
- Pass all required variables

main.tf

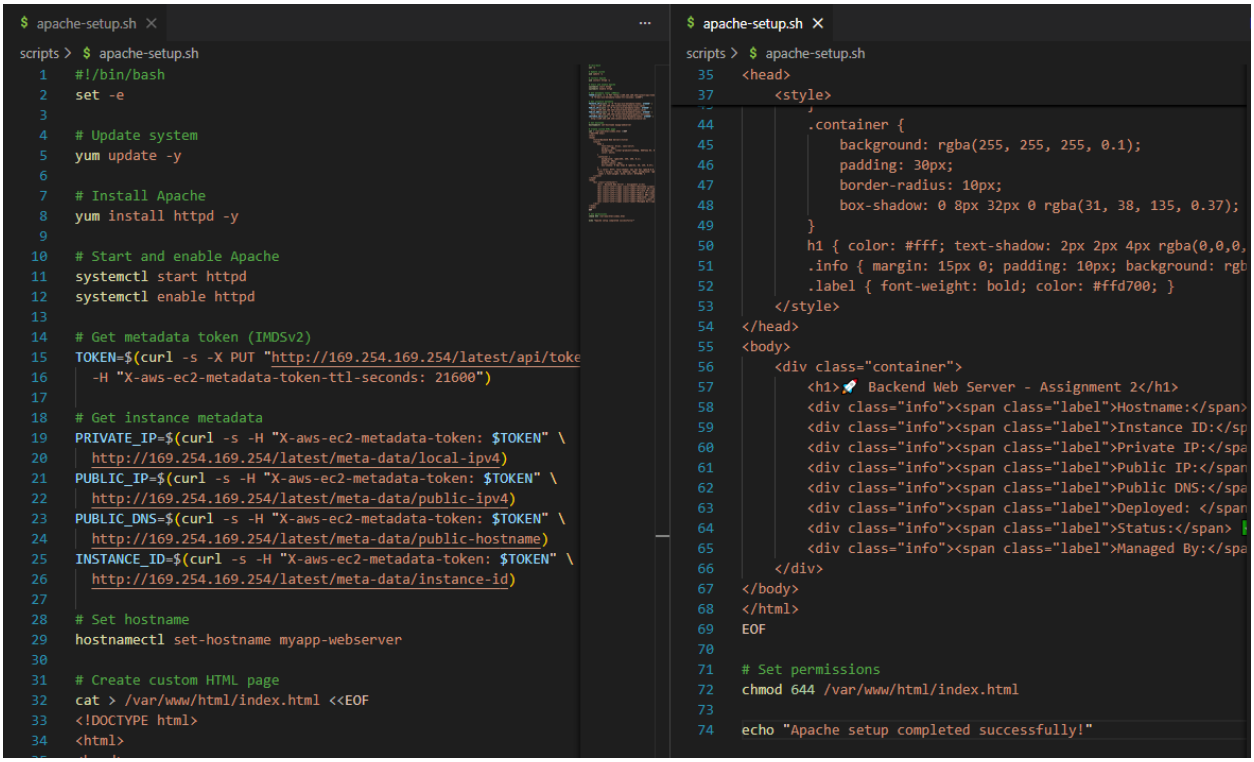
```hcl
provider "aws" {
  region = "me-central-1"
}

# --- Part 1: Networking & Security ---

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
}

module "security" {
  source     = "./modules/security"
  vpc_id     = module.networking.vpc_id
  env_prefix = var.env_prefix
  my_ip      = local.my_ip
}

# --- Part 2: Webservers ---

# 1. Nginx Server (Single Instance)
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
```

main.tf

```hcl
module "nginx_server" {
  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
}

# 2. Backend Servers (Dynamic Loop using for_each)
module "backend_servers" {
  for_each = { for idx, server in local.backend_servers : serve

  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
  # Note: Using backend_sg_id here (restricted access)
  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 (20 marks)

3.1 Apache Backend Server Script (10 marks)

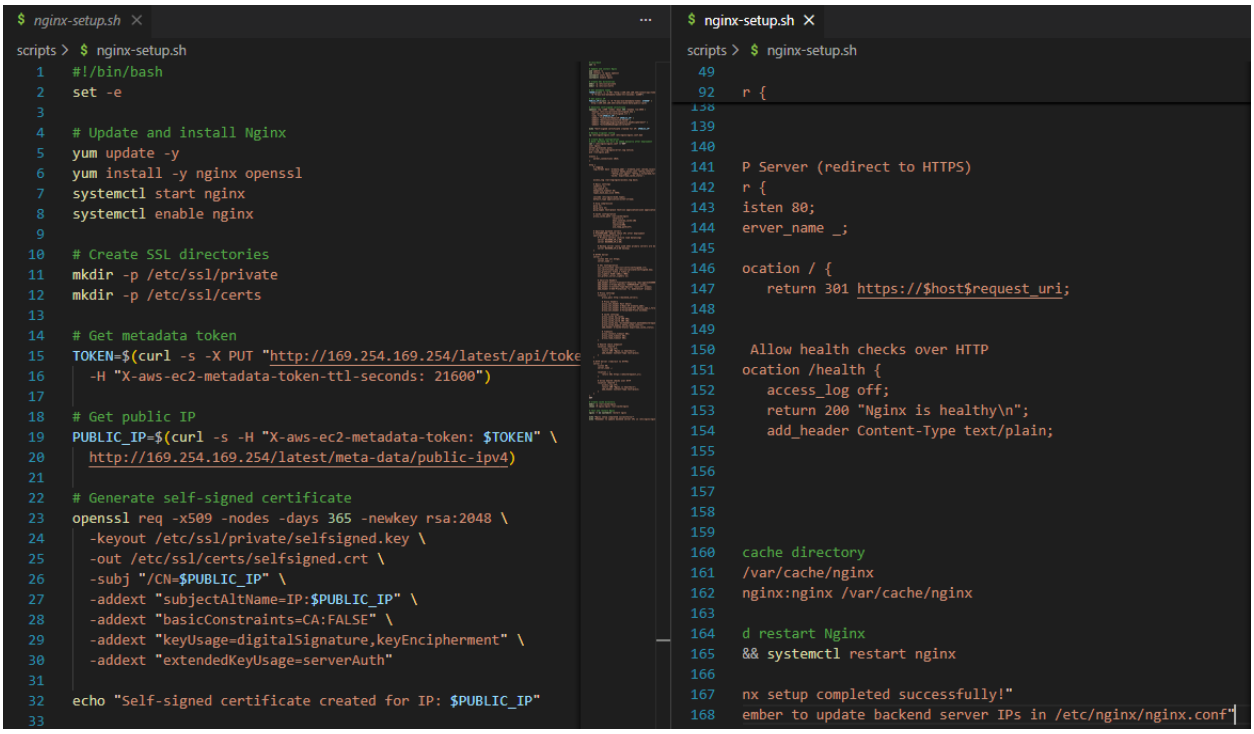


```
#!/bin/bash
set -e

# 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/toke
  -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>
    <style>
        .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,
        .info { margin: 15px 0; padding: 10px; background: rgb
        .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>
        <div class="info"><span class="label">Instance ID:</sp
        <div class="info"><span class="label">Private IP:</spa
        <div class="info"><span class="label">Public IP:</span
        <div class="info"><span class="label">Public DNS:</spa
        <div class="info"><span class="label">Deployed: </span
        <div class="info"><span class="label">Status:</span>
        <div class="info"><span class="label">Managed By:</spa
    </div>
</body>
</html>
EOF

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

echo "Apache setup completed successfully!"
```

3.2 Nginx Server Setup Script (10 marks)

Tasks:

- Create script with SSL certificate generation
- Configure upstream with placeholder IPs
- Implement caching
- Add security headers
- Configure HTTP to HTTPS redirect
- Add health check endpoint



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

# Update and install Nginx
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
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/toke
  -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 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"
```

```
r {

P Server (redirect to HTTPS)
r {
isten 80;
erver_name _;

ocation / {
   return 301 https://$host$request_uri;


 Allow health checks over HTTP
ocation /health {
   access_log off;
   return 200 "Nginx is healthy\n";
   add_header Content-Type text/plain;




cache directory
/var/cache/nginx
nginx:nginx /var/cache/nginx

d restart Nginx
&& systemctl restart nginx

nx setup completed successfully!"
ember to update backend server IPs in /etc/nginx/nginx.conf"
```

Part 4: Infrastructure Deployment (15 marks)

4.1 Initial Deployment (5 marks)

Deploy the infrastructure using Terraform.

Tasks:

- Generate SSH key pair if not exists
- Initialize Terraform (terraform init)
- Validate configuration (terraform validate)
- Plan deployment (terraform plan)
- Apply configuration (terraform apply -auto-approve)

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ ssh-keygen -t ed25519 -f ~/.ssh/id_ed25519 -N ""
Generating public/private ed25519 key pair.
/c/Users/Rughma/.ssh/id_ed25519 already exists.
Overwrite (y/n)? n
```

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0
- Using previously-installed hashicorp/http v3.5.0

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$
```

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform validate
Success! The configuration is valid.
```

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform plan

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + primary_network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 6 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.

Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$
```

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform apply -auto-approve
module.networking.aws_subnet.myapp_subnet: Still creating... [00m10s elapsed]
module.networking.aws_subnet.myapp_subnet: Creation complete after 12s [id=subnet-001cfc4f830f4d588]
module.networking.aws_route_table_association.a-rtb-subnet: Creating...
module.backend_servers["web-2"].aws_instance.myapp_server: Creating...
module.backend_servers["web-1"].aws_instance.myapp_server: Creating...
module.nginx_server.aws_instance.myapp_server: Creating...
module.backend_servers["web-3"].aws_instance.myapp_server: Creating...
module.networking.aws_route_table_association.a-rtb-subnet: Creation complete after 0s [id=rtbassoc-0a1ac9583a4d3fcba]
module.backend_servers["web-2"].aws_instance.myapp_server: Still creating... [00m10s elapsed]
module.backend_servers["web-1"].aws_instance.myapp_server: Still creating... [00m10s elapsed]
module.nginx_server.aws_instance.myapp_server: Still creating... [00m10s elapsed]
module.backend_servers["web-3"].aws_instance.myapp_server: Still creating... [00m10s elapsed]
module.backend_servers["web-1"].aws_instance.myapp_server: Creation complete after 13s [id=i-0bf37fdce3ea62d4d]
module.backend_servers["web-2"].aws_instance.myapp_server: Creation complete after 13s [id=i-0c87ebcfb4d5976fa]
module.backend_servers["web-3"].aws_instance.myapp_server: Creation complete after 13s [id=i-0ddead9a50b5e91b3]
module.nginx_server.aws_instance.myapp_server: Creation complete after 13s [id=i-0eff4e877ccb53f4b]

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

Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$
```

4.2 Output Configuration (5 marks)

Tasks:

- Create all required outputs
- Add helpful descriptions
- Include configuration guide
- Display outputs after apply

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform output

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

Next Steps:
1. SSH into Nginx server:
   ssh -i ~/.ssh/id_ed25519 ec2-user@158.252.77.128

2. Edit Nginx config:
   sudo vim /etc/nginx/nginx.conf

3. Update backend IPs in 'upstream backend_servers' block:
   - BACKEND_IP_1: 10.0.10.185
   - BACKEND_IP_2: 10.0.10.194
   - BACKEND_IP_3: 10.0.10.93

4. Restart Nginx:
   sudo systemctl restart nginx

5. Test:
   https://158.252.77.128

Backend Servers List:
- web-1: Public: 3.28.44.253 | Private: 10.0.10.185
    - web-2: Public: 40.172.177.31 | Private: 10.0.10.194
    - web-3: Public: 40.172.215.101 | Private: 10.0.10.93

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

EOT
nginx_instance_id = "i-0eff4e877ccb53f4b"
nginx_public_ip = "158.252.77.128"
subnet_id = "subnet-001cfc4f830f4d588"
vpc_id = "vpc-011d0a54436bf2c22"

Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
```

terraform.tfvars | outputs.tf | outputs.json

outputs.json > ...

```
{
  "backend_servers_info": {
    "sensitive": false,
    "type": [
      "object",
      {
        "web-1": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-2": [
          "object",
          {
```

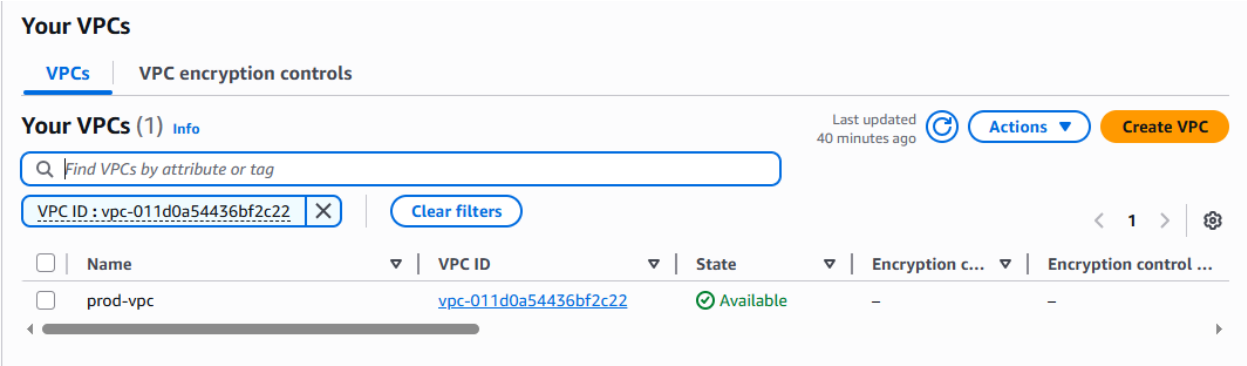
PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

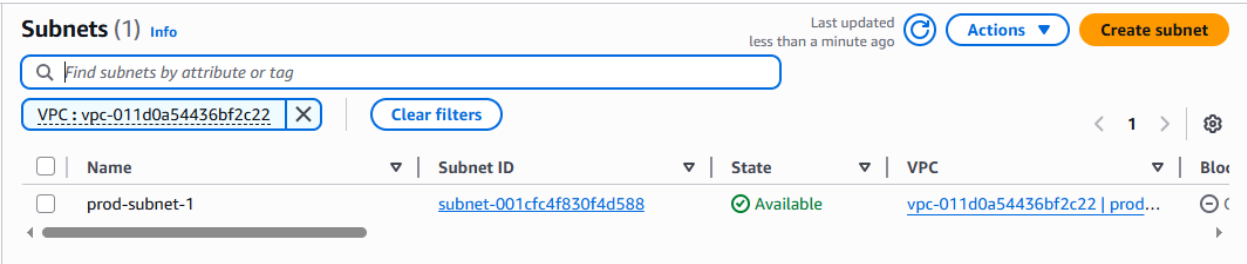
```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ terraform output -json > outputs.json
```

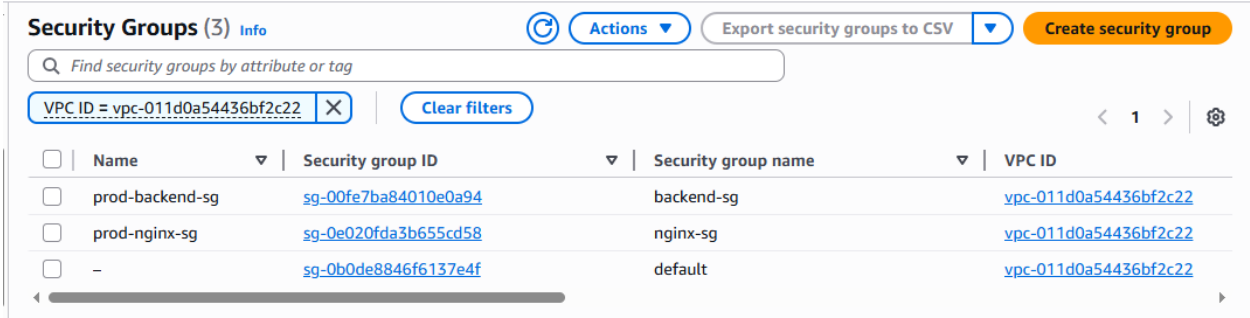
4.3 AWS Console Verification (5 marks)

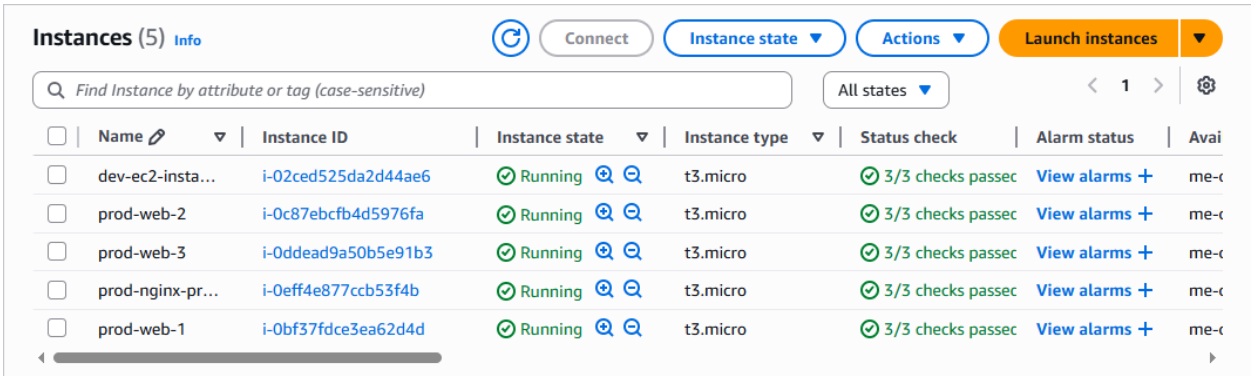
Tasks:

- Verify VPC created
- Verify Subnet created
- Verify Internet Gateway attached
- Verify Route Table configured
- Verify Security Groups created with correct rules
- Verify all 4 EC2 instances running
- Verify Key Pairs created









Part 5: Nginx Configuration & Testing (25 marks)

5.1 Update Nginx Backend Configuration (5 marks)

Tasks:

- SSH into Nginx server
- Edit /etc/nginx/nginx.conf
- Replace placeholder IPs with actual private IPs of backend servers
- Test Nginx configuration
- Restart Nginx service

```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ ssh -i ~/.ssh/id_ed25519 ec2-user@158.252.77.128
The authenticity of host '158.252.77.128 (158.252.77.128)' can't be established.
ED25519 key fingerprint is SHA256:q0r7Q4ditp+mfTKqwnfWbQkbQLCeXdWPoPqKScBCQCA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '158.252.77.128' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-10-89 ~]$
```

```
    # Upstream backend servers
    # PLACEHOLDER: Update these IPs after deployment
    upstream backend_servers {
        # Primary servers (active load balancing)
        server 10.0.10.185:80;
        server 10.0.10.194:80;

        # Backup server (only used when primary servers are down)
        server 10.0.10.93:80 backup;
    }

    # HTTPS Server
    server {
        listen 443 ssl http2;
        server_name _;
-- INSERT --
```

```
[ec2-user@ip-10-0-10-89 ~]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-89 ~]$ sudo nginx -t
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" directive instead in /etc/nginx/nginx.conf:54
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-89 ~]$
```

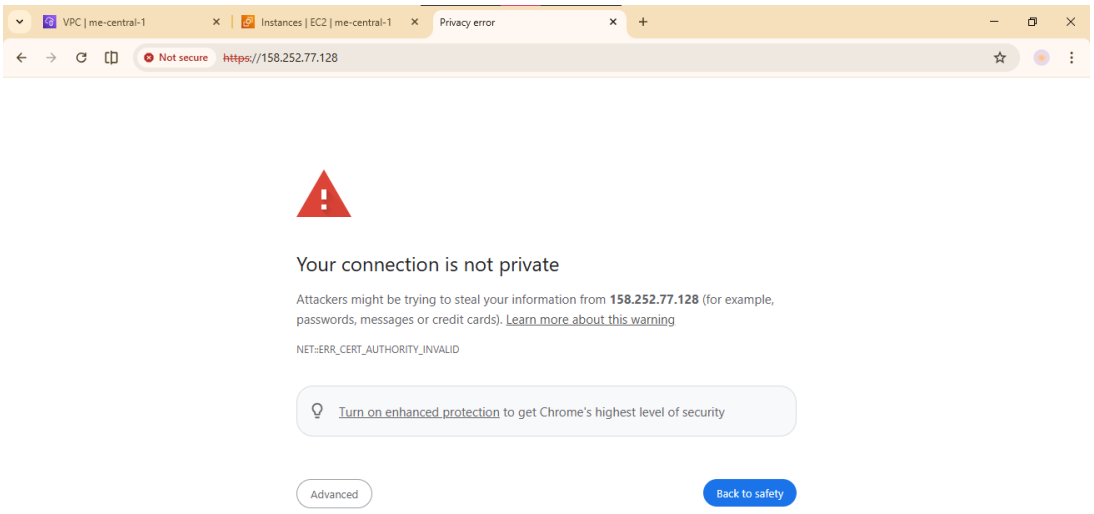
```
[ec2-user@ip-10-0-10-89 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-89 ~]$ sudo systemctl status nginx
\● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Tue 2025-12-30 14:14:11 UTC; 8s ago
    Process: 27337 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 27338 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 27339 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 27340 (nginx)
      Tasks: 5 (limit: 1067)
     Memory: 4.5M
        CPU: 59ms
```

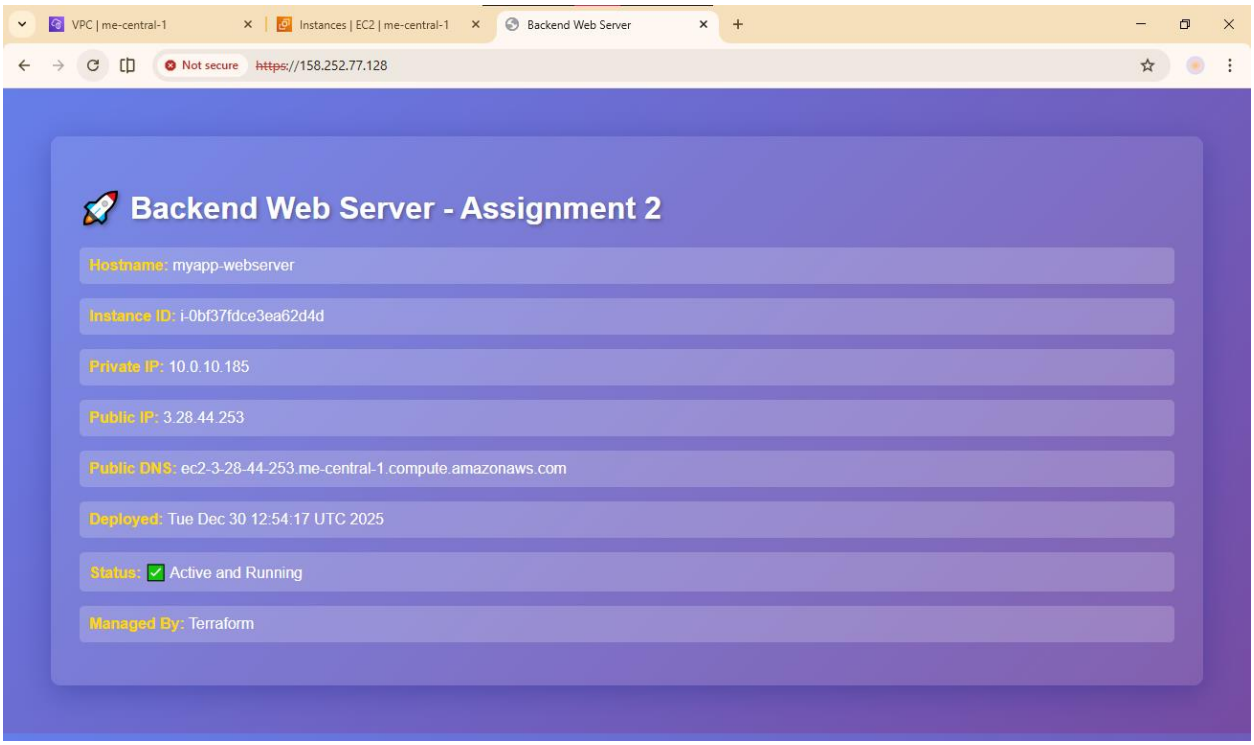
5.2 Test Load Balancing (5 marks)

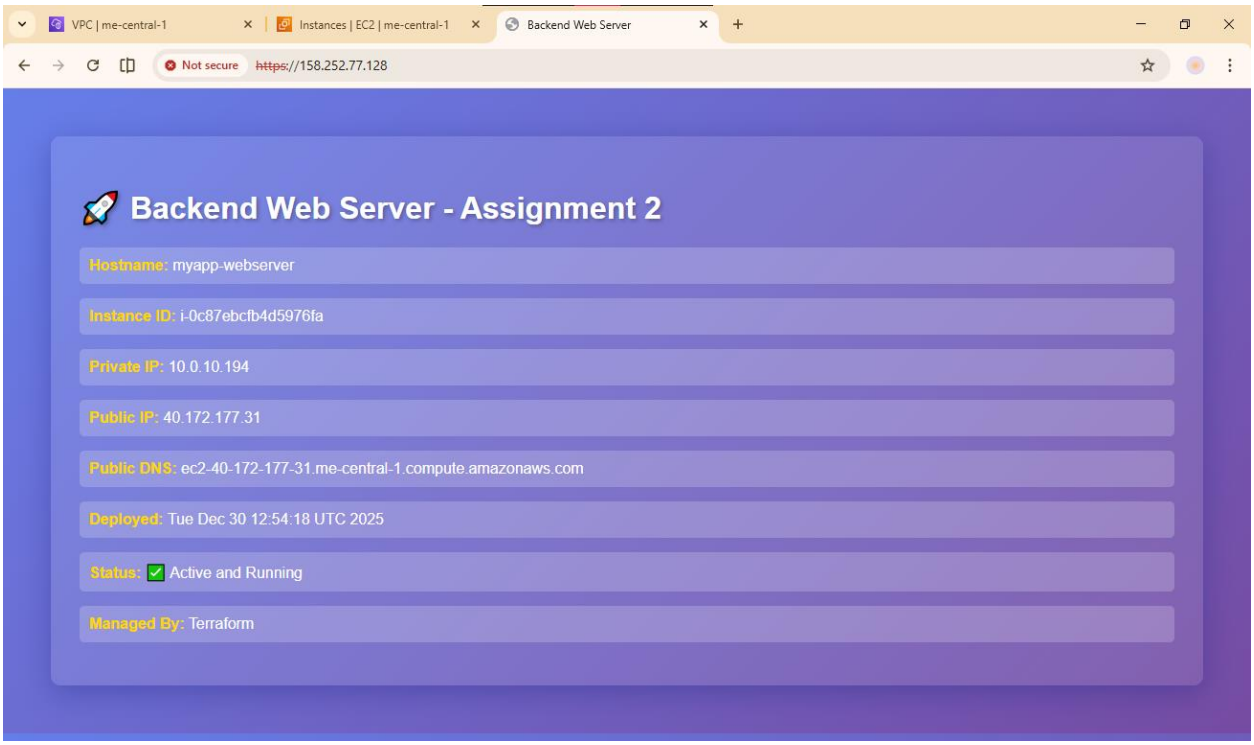
Test that Nginx is properly load balancing between web-1 and web-2.

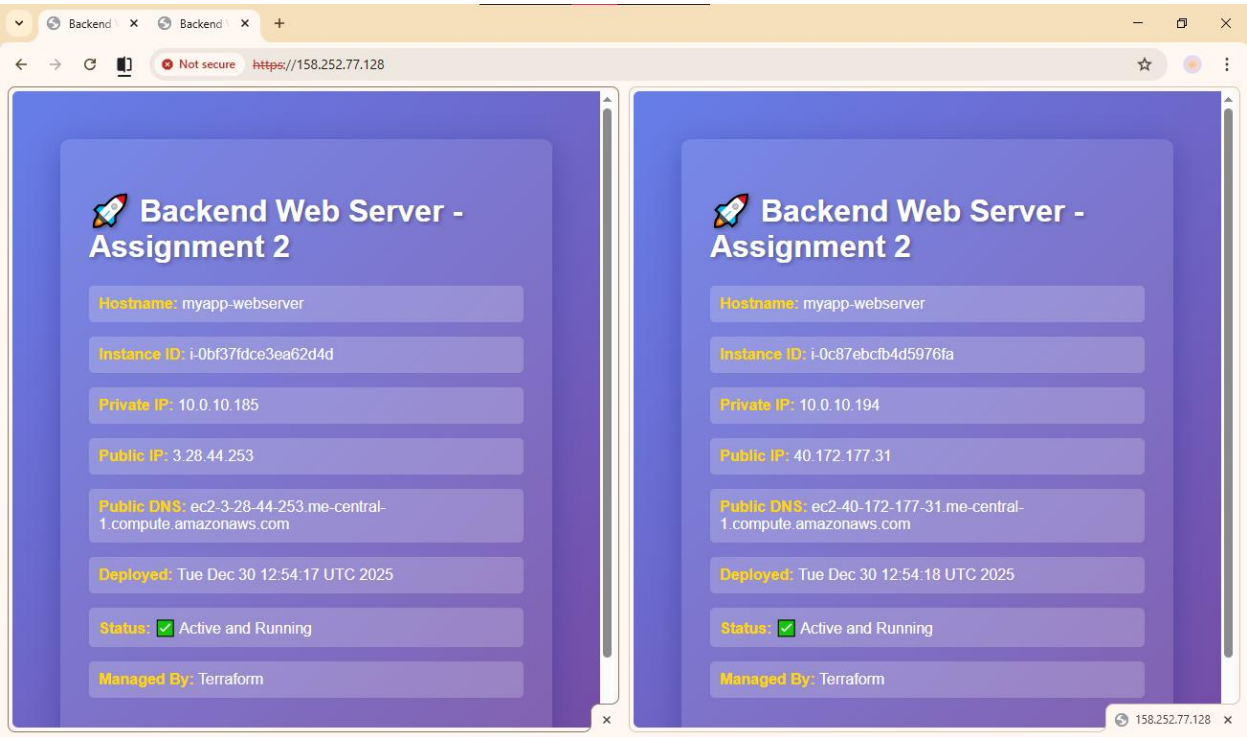
Tasks:

- Open browser to https://<nginx-public-ip>
- Accept the security warning for self-signed certificate
- Reload page multiple times (at least 10 times)
- Verify traffic alternates between web-1 and web-2
- Verify web-3 is NOT serving traffic (it's backup only)
- Document the load balancing pattern




VPC | me-central-1
Instances | EC2 | me-central-1
Backend Web Server
Not secure
https://158.252.77.128
Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-0bf37fdce3ea62d4d
Private IP: 10.0.10.185
Public IP: 3.28.44.253
Public DNS: ec2-3-28-44-253.me-central-1.compute.amazonaws.com
Deployed: Tue Dec 30 12:54:17 UTC 2025
Status: Active and Running
Managed By: Terraform

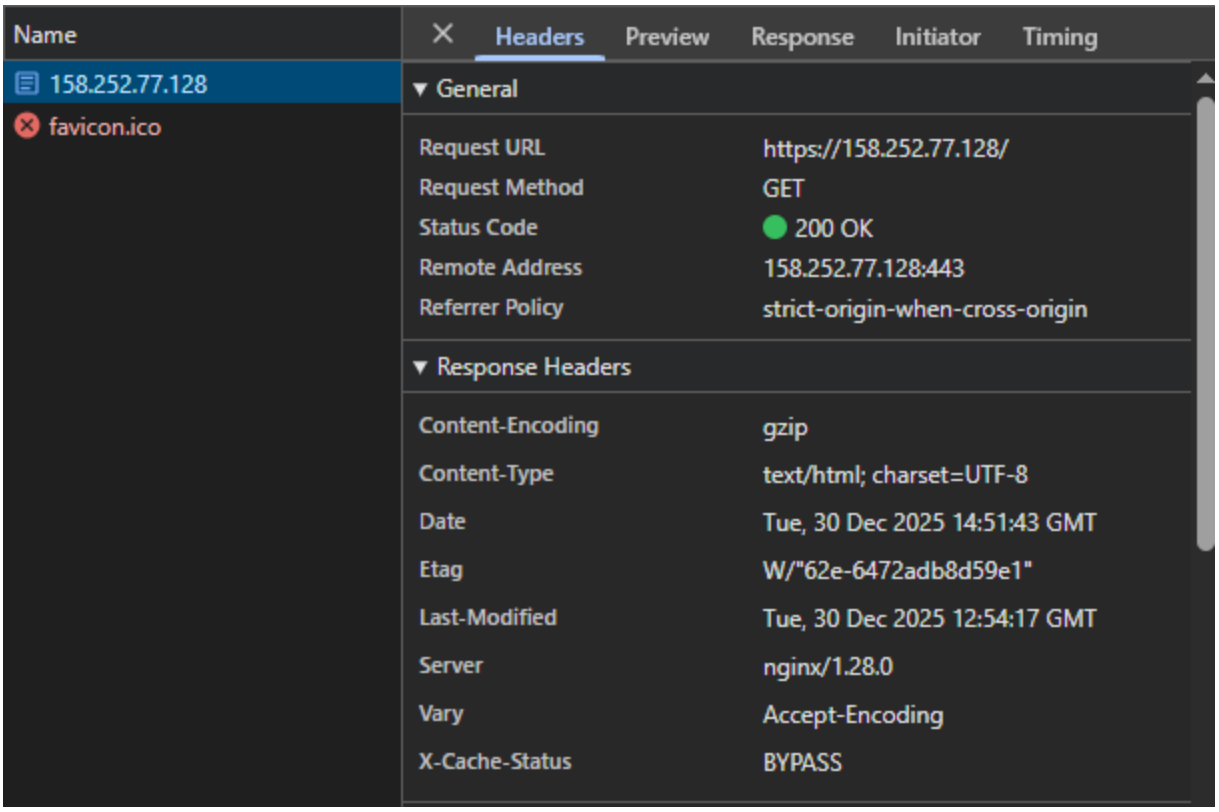

VPC | me-central-1
Instances | EC2 | me-central-1
Backend Web Server
Not secure
https://158.252.77.128
Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-0c87ebcfb4d5976fa
Private IP: 10.0.10.194
Public IP: 40.172.177.31
Public DNS: ec2-40-172-177-31.me-central-1.compute.amazonaws.com
Deployed: Tue Dec 30 12:54:18 UTC 2025
Status: Active and Running
Managed By: Terraform

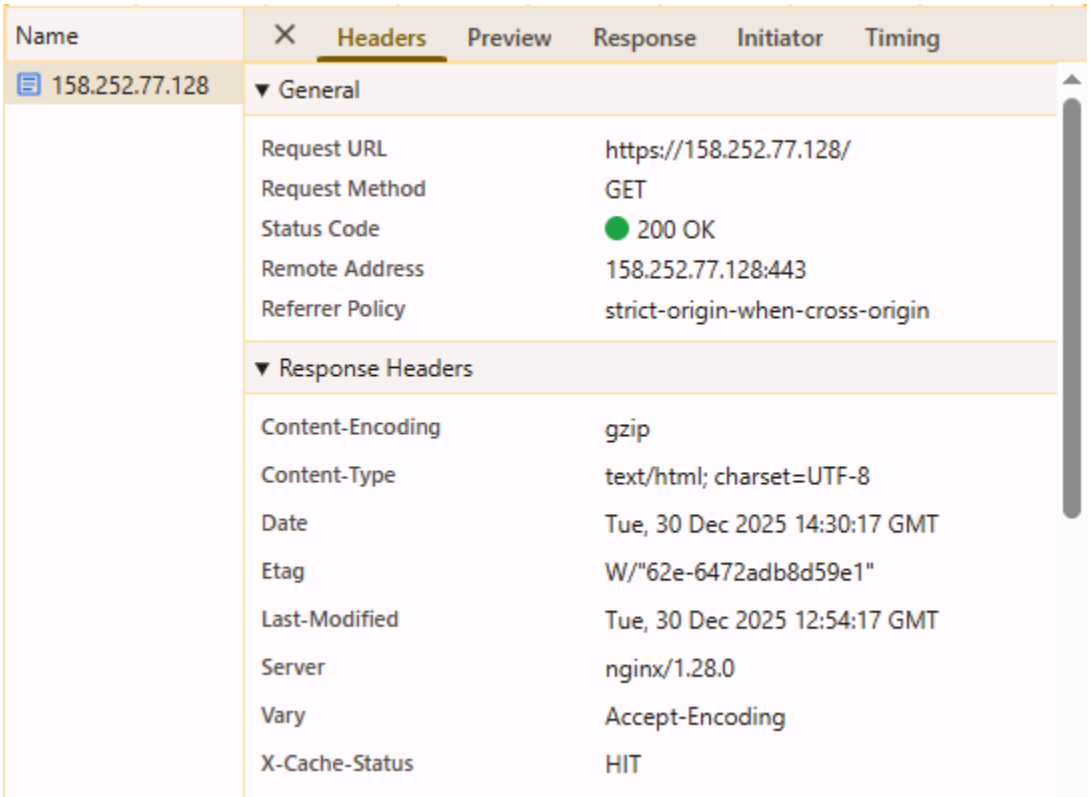


5.3 Test Cache Functionality (5 marks)

Tasks:

- Open browser developer tools (F12)
- Navigate to Network tab
- Clear browser cache
- Load https://<nginx-public-ip>
- Check response headers for X-Cache-Status: MISS (first request)
- Reload page
- Check response headers for X-Cache-Status: HIT (cached request)
- Verify cache directory on Nginx server


Name
158.252.77.128
favicon.ico
Headers
Preview
Response
Initiator
Timing
General
Request URL
https://158.252.77.128/
Request Method
GET
Status Code
200 OK
Remote Address
158.252.77.128:443
Referrer Policy
strict-origin-when-cross-origin
Response Headers
Content-Encoding
gzip
Content-Type
text/html; charset=UTF-8
Date
Tue, 30 Dec 2025 14:51:43 GMT
Etag
W/"62e-6472adb8d59e1"
Last-Modified
Tue, 30 Dec 2025 12:54:17 GMT
Server
nginx/1.28.0
Vary
Accept-Encoding
X-Cache-Status
BYPASS


Name
158.252.77.128
Headers
Preview
Response
Initiator
Timing
General
Request URL
https://158.252.77.128/
Request Method
GET
Status Code
200 OK
Remote Address
158.252.77.128:443
Referrer Policy
strict-origin-when-cross-origin
Response Headers
Content-Encoding
gzip
Content-Type
text/html; charset=UTF-8
Date
Tue, 30 Dec 2025 14:30:17 GMT
Etag
W/"62e-6472adb8d59e1"
Last-Modified
Tue, 30 Dec 2025 12:54:17 GMT
Server
nginx/1.28.0
Vary
Accept-Encoding
X-Cache-Status
HIT

```
[ec2-user@ip-10-0-10-89 ~]$ ls -la /var/cache/nginx/
total 0
drwxr-xr-x. 6 nginx nginx  42 Dec 30 15:03 .
drwxr-xr-x. 9 root  root  101 Dec 30 12:54 ..
drwx------. 3 nginx nginx  16 Dec 30 15:03 3
drwx------. 3 nginx nginx  16 Dec 30 14:14 7
drwx------. 3 nginx nginx  16 Dec 30 14:39 8
drwx------. 3 nginx nginx  16 Dec 30 14:17 a
[ec2-user@ip-10-0-10-89 ~]$
```

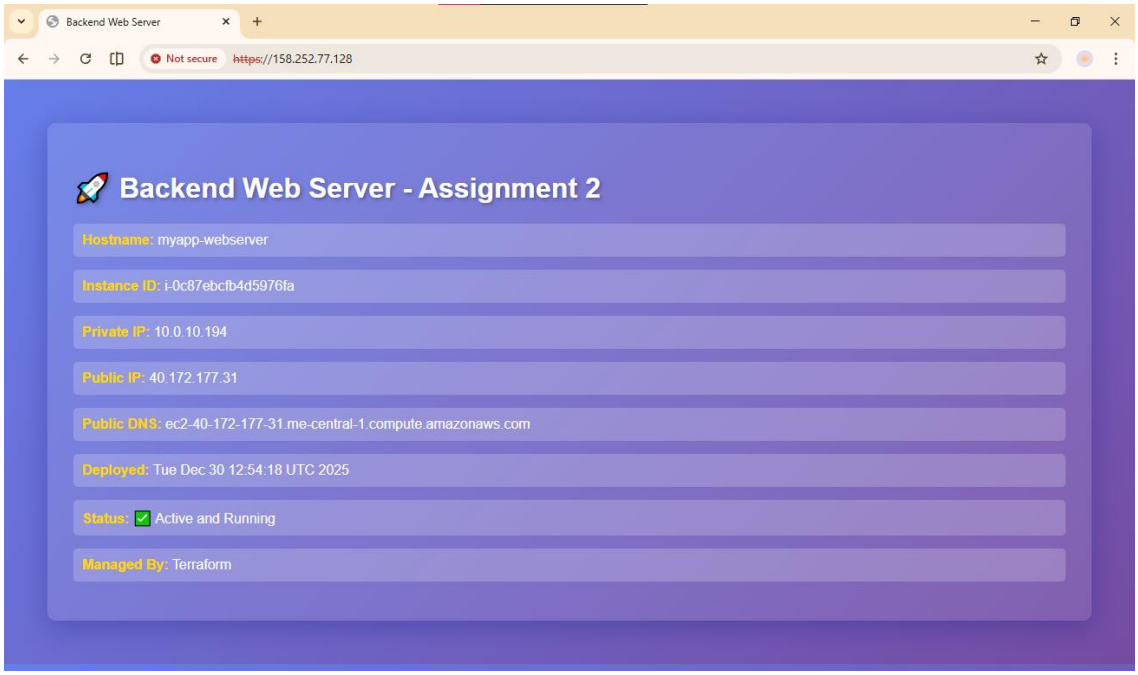
```
[ec2-user@ip-10-0-10-89 ~]$ sudo grep "Cache:" /var/log/nginx/access.log | tail -5
154.192.0.128 - - [30/Dec/2025:14:51:43 +0000] "GET /favicon.ico HTTP/2.0" 404 172 "https://158.252.77.128/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache: EXPIRED
154.192.0.128 - - [30/Dec/2025:14:53:40 +0000] "GET / HTTP/2.0" 200 691 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
154.192.0.128 - - [30/Dec/2025:14:54:05 +0000] "GET / HTTP/2.0" 200 693 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
154.192.0.128 - - [30/Dec/2025:14:54:06 +0000] "GET / HTTP/2.0" 200 691 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
48.217.87.78 - - [30/Dec/2025:15:03:43 +0000] "GET /ecp/Current/exporttool/microsoft.exchange.ediscovery.exporttool.application HTTP/1.1" 404 183 "-" "Mozilla/5.0 zgrab/0.x" "-" Cache: MISS
[ec2-user@ip-10-0-10-89 ~]$
```

5.4 Test High Availability (Backup Server) (5 marks)

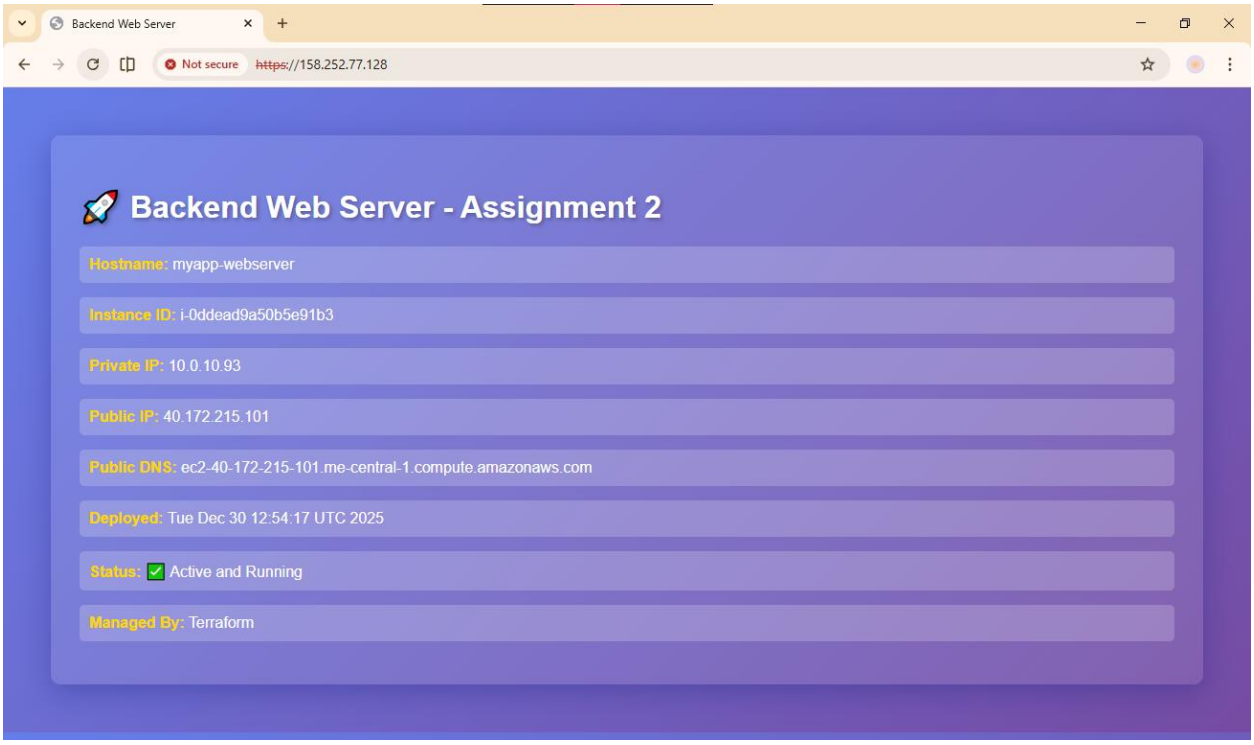
Test the backup server functionality by simulating primary server failure.

Tasks:

- SSH into web-1 and stop Apache
- Reload Nginx page - should show web-2 only
- SSH into web-2 and stop Apache
- Reload Nginx page - should now show web-3 (backup activated)
- Restart web-1 and web-2
- Verify traffic returns to web-1 and web-2

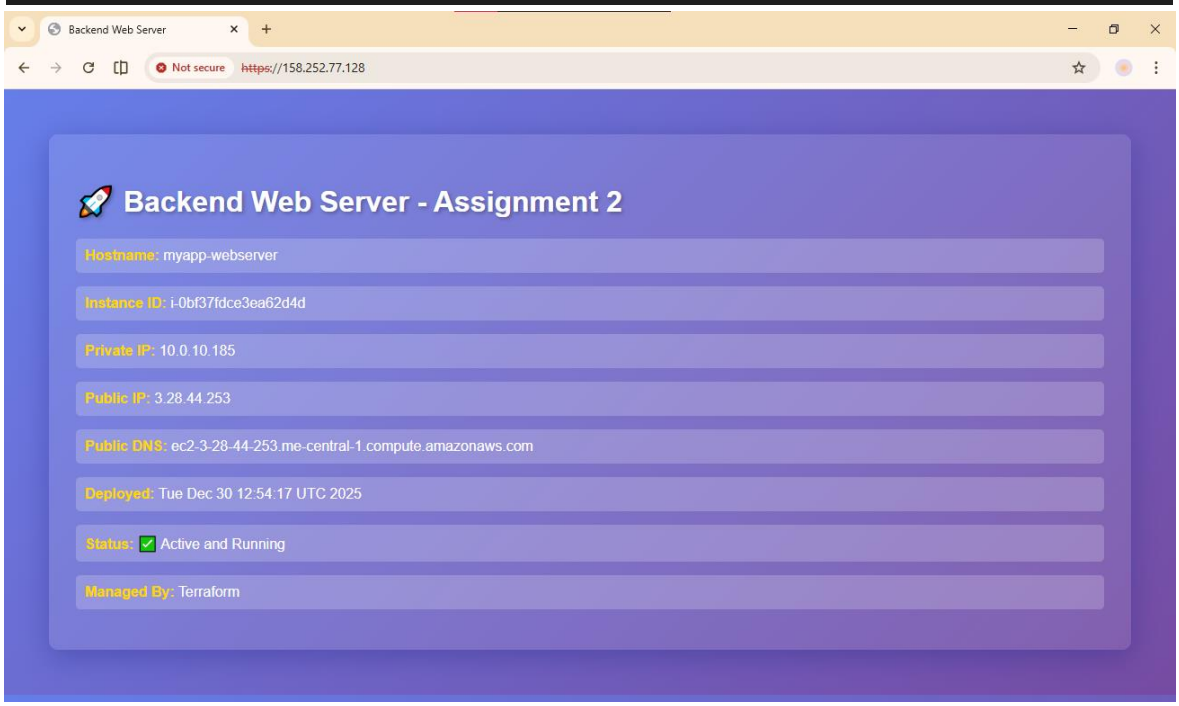


```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ ssh -i ~/.ssh/id_ed25519 ec2-user@40.172.177.31
The authenticity of host '40.172.177.31 (40.172.177.31)' can't be established.
ED25519 key fingerprint is SHA256:wXQwskHgcbnoXI2O8lPXQnKOSDxUHs648uZcpVpij2A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '40.172.177.31' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$
```




```
[ec2-user@ip-10-0-10-89 ~]$ sudo tail -n 20 /var/log/nginx/error.log
2025/12/30 14:14:11 [notice] 27339#27339: using the "epoll" event method
2025/12/30 14:14:11 [notice] 27339#27339: nginx/1.28.0
2025/12/30 14:14:11 [notice] 27339#27339: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/30 14:14:11 [notice] 27339#27339: getrlimit(RLIMIT_NOFILE): 65535:65535
2025/12/30 14:14:11 [notice] 27340#27340: start worker processes
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27341
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27342
2025/12/30 14:14:11 [notice] 27340#27340: start cache manager process 27343
2025/12/30 14:14:11 [notice] 27340#27340: start cache loader process 27344
2025/12/30 14:15:11 [notice] 27344#27344: http file cache: /var/cache/nginx 0.004M, bsize: 4096
2025/12/30 14:15:12 [notice] 27340#27340: signal 17 (SIGCHLD) received from 27344
2025/12/30 14:15:12 [notice] 27340#27340: cache loader process 27344 exited with code 0
2025/12/30 14:15:12 [notice] 27340#27340: signal 29 (SIGIO) received
2025/12/30 14:26:24 [crit] 27342#27342: *73 SSL_do_handshake() failed (SSL: error:0A000172:SSL routines::wrong signature type) while SSL handshaking, client:
 50.116.49.249, server: 0.0.0.0:443
2025/12/30 15:44:19 [error] 27342#27342: *154 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:44:19 [warn] 27342#27342: *154 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
[ec2-user@ip-10-0-10-89 ~]$
```

```
[ec2-user@myapp-webserver ~]$ sudo systemctl start httpd
[ec2-user@myapp-webserver ~]$
```



5.5 Security & Performance Analysis (5 marks)

Analyze the security headers and performance of your Nginx setup.

Tasks:

- Check SSL/TLS certificate details
- Verify security headers in response
- Test HTTP to HTTPS redirect
- Check response times
- Analyze Nginx logs

```
[ec2-user@ip-10-0-10-89 ~]$ curl -I http://158.252.77.128
\\HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 16:04:38 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://158.252.77.128/

[ec2-user@ip-10-0-10-89 ~]$
```

```
[ec2-user@ip-10-0-10-89 ~]$ curl -I -k https://158.252.77.128
HTTP/2 200
server: nginx/1.28.0
date: Tue, 30 Dec 2025 16:05:34 GMT
content-type: text/html; charset=UTF-8
content-length: 1586
vary: Accept-Encoding
last-modified: Tue, 30 Dec 2025 12:54:18 GMT
etag: "632-6472adb9ca5aa"
x-cache-status: MISS
accept-ranges: bytes

[ec2-user@ip-10-0-10-89 ~]$
```

```
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 729F5EF620634B1BF39F82402855BB36AAF4B32363F063B76AF0B81672D3AC9B
    Session-ID-ctx:
    Resumption PSK: A3B74206CC70757A8EE63791524EBB721976E40A3794429D99EEA558193D1BB358DD53AC47B1D3EAF88FA5602AADED09
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - a5 80 d2 62 3f 60 da 16-fa 8f c4 47 46 fe de e2   ...b?`.....GF...
    0010 - 6b 58 7f 8f 43 02 b8 aa-27 d7 c6 5b b1 42 49 f4   kX..C...'..[.BI.
    0020 - 36 b0 ca f5 bf 60 22 b1-8a b6 e3 6b 0c 48 6b 9b   6....`".....k.Hk.
    0030 - dd 59 68 07 78 cf bc 02-74 69 a8 70 5f c0 4c 08   .Yh.x...ti.p_.L.
    0040 - 24 33 25 48 8c c7 c8 b5-3f 66 04 77 19 e9 c2 f0   $3%H....?f.w....
    0050 - be d2 79 b5 53 0b 48 0b-81 82 3f 09 c3 a3 5b cc   ..y.S.H...?...[.
    0060 - 54 96 a3 f0 ae 22 2f bb-b2 c1 cc 05 ee cc 0f 34   T...."/........4
    0070 - 91 fc af 2d e6 23 45 9e-9c 2f cc 0a cc a9 60 d7   ...-.#E../....`.
    0080 - cf 39 72 dc 94 78 54 76-67 f1 84 7f a4 95 f9 ba   .9r..xTvg.......
    0090 - ed 08 ec ce 22 1f 03 df-fb 7e 2b df 79 9b 86 b6   ....".....~+.y...
    00a0 - f9 b3 85 c3 83 76 80 88-00 56 b1 3a be 9d 41 09   .....v...V.:..A.
    00b0 - 4c 6e a5 63 1a 4d bd 79-9e b9 c7 88 6c 29 4e e0   Ln.c.M.y....l)N.
    00c0 - b4 0c c7 08 1f fb bf 43-ad 88 1c d4 e0 9a e2 47   .......C.......G
    00d0 - aa 49 82 bd 49 60 6a 42-b3 e3 6b eb 18 33 9a e2   .I..I`jB..k..3..

    Start Time: 1767110810
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
```

```
 Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
154.192.0.128 - - [30/Dec/2025:15:50:45 +0000] "GET / HTTP/2.0" 200 693 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
154.192.0.128 - - [30/Dec/2025:15:50:46 +0000] "GET / HTTP/2.0" 200 693 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
154.192.0.128 - - [30/Dec/2025:15:50:47 +0000] "GET / HTTP/2.0" 200 691 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/143.0.0.0 Safari/537.36" "-" Cache: BYPASS
176.32.195.85 - - [30/Dec/2025:15:53:43 +0000] "\x16\x03\x02\x01o\x01\x00\x01k\x03\x02RH\xC5\x1A#\xF7:N\xDF\xE2\xB4\x82/\xFF\x09T\x9F\xA7\xC4y\xB0h\xC6\x13\x
8C\xA4\x1C=\x22\xE1\x1A\x98 \x84\xB4,\x85\xAFn\xE3Y\xBBbhl\xFF(=':\xA9\x82\xD9o\xC8\xA2\xD7\x93\x98\xB4\xEF\x80\xE5\xB9\x90\x00(\xC0" 400 157 "-" "-" "-" Cac
he: -
95.215.0.144 - - [30/Dec/2025:15:54:36 +0000] "GET / HTTP/1.1" 301 169 "-" "fasthttp" "-" Cache: -
46.161.50.108 - - [30/Dec/2025:15:54:38 +0000] "\x16\x03\x01\x05\xA8\x01\x00\x05\xA4\x03\x03S= \x0FY\x89_\xD5\xC3\x12\xC5N\xF3\xB4(\xE4\x8EF\xA3t-^BM\x99|" 4
00 157 "-" "-" "-" Cache: -
46.161.50.108 - - [30/Dec/2025:15:54:38 +0000] "GET /aaa9 HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/70.0.3538.77 Safari/537.36" "-" Cache: -
46.161.50.108 - - [30/Dec/2025:15:54:38 +0000] "\x16\x03\x01\x05\xA8\x01\x00\x05\xA4\x03\x03\x7F\x04V\xD2g\xD82\x80\xD0" 400 157 "-" "-" "-" Cache: -
46.161.50.108 - - [30/Dec/2025:15:54:38 +0000] "GET /aab9 HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
cko) Chrome/70.0.3538.77 Safari/537.36" "-" Cache: -
45.91.64.8 - - [30/Dec/2025:15:55:00 +0000] "\x16\x03\x01\x05\xA8\x01\x00\x05\xA4\x03\x03\xA1\xE2\x016u\xAFr\x8C,\x98\x0F\x01\xB6\xD5\x89Pe\xE3\xC5?\xB1F\xF1
\x22\x03Qz\xB4\xAC\xEC,\xA4 \x95\xC4*\x94/\xAEU\x8E\x8F\xCBd\xFF\x0C\x83\xCA\xF3Dy" 400 157 "-" "-" "-" Cache: -
45.91.64.8 - - [30/Dec/2025:15:55:00 +0000] "GET /aaa9 HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/70.0.3538.77 Safari/537.36" "-" Cache: -
45.91.64.8 - - [30/Dec/2025:15:55:00 +0000] "\x16\x03\x01\x05\xA8\x01\x00\x05\xA4\x03\x033\xD8\xF9\xF7\xB4\x90\x89\x99\xDF\x91s\xBD\xA6\xCF\xBAMk\x12\xEE\xC2
qX\xA0\xFC\xDBa\xB9\xFA\x87\x09\xCF\xD6 v\xA1Zxn2\xA5\xC5)\xA2J$\xC2\x9DT\xF3e\x9E\x0B\xE7\x1D41\x8F\x1FL\xBA\xED\xB4\xFE\xE9\xB7\x00\x1A\xC0+\xC0/\xC0,\xC00
\xCC\xA9\xCC\xA8\xC0\x09\xC0\x13\xC0" 400 157 "-" "-" "-" Cache: -
45.91.64.8 - - [30/Dec/2025:15:55:00 +0000] "GET /aab9 HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
) Chrome/70.0.3538.77 Safari/537.36" "-" Cache: -
93.174.93.12 - - [30/Dec/2025:15:58:16 +0000] "GET / HTTP/1.0" 301 169 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.
0.3889.0 Safari/537.36" "-" Cache: -
129.212.186.47 - - [30/Dec/2025:16:01:00 +0000] "SSTP_DUPLEX_POST /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/ HTTP/1.1" 400 157 "-" "-" "-" Cache: -
158.252.77.128 - - [30/Dec/2025:16:04:38 +0000] "HEAD / HTTP/1.1" 301 0 "-" "curl/8.11.1" "-" Cache: -
158.252.77.128 - - [30/Dec/2025:16:05:34 +0000] "HEAD / HTTP/2.0" 200 0 "-" "curl/8.11.1" "-" Cache: MISS
20.119.75.60 - - [30/Dec/2025:16:05:39 +0000] "GET /login HTTP/1.1" 404 183 "-" "Mozilla/5.0 zgrab/0.x" "-" Cache: MISS
[ec2-user@ip-10-0-10-89 ~]$
```

```
0.3889.0 Safari/537.36" "-" Cache: -
129.212.186.47 - - [30/Dec/2025:16:01:00 +0000] "SSTP_DUPLEX_POST /sra_{BA195980-CD49-458b-9E23-C84EE0ADCD75}/ HTTP/1.1" 400 157 "-" "-" "-" Cache: -
158.252.77.128 - - [30/Dec/2025:16:04:38 +0000] "HEAD / HTTP/1.1" 301 0 "-" "curl/8.11.1" "-" Cache: -
158.252.77.128 - - [30/Dec/2025:16:05:34 +0000] "HEAD / HTTP/2.0" 200 0 "-" "curl/8.11.1" "-" Cache: MISS
20.119.75.60 - - [30/Dec/2025:16:05:39 +0000] "GET /login HTTP/1.1" 404 183 "-" "Mozilla/5.0 zgrab/0.x" "-" Cache: MISS
[ec2-user@ip-10-0-10-89 ~]$ sudo tail -n 20 /var/log/nginx/error.log
2025/12/30 14:14:11 [notice] 27339#27339: using the "epoll" event method
2025/12/30 14:14:11 [notice] 27339#27339: nginx/1.28.0
2025/12/30 14:14:11 [notice] 27339#27339: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/30 14:14:11 [notice] 27339#27339: getrlimit(RLIMIT_NOFILE): 65535:65535
2025/12/30 14:14:11 [notice] 27340#27340: start worker processes
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27341
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27342
2025/12/30 14:14:11 [notice] 27340#27340: start cache manager process 27343
2025/12/30 14:14:11 [notice] 27340#27340: start cache loader process 27344
2025/12/30 14:15:11 [notice] 27344#27344: http file cache: /var/cache/nginx 0.004M, bsize: 4096
2025/12/30 14:15:12 [notice] 27340#27340: signal 17 (SIGCHLD) received from 27344
2025/12/30 14:15:12 [notice] 27340#27340: cache loader process 27344 exited with code 0
2025/12/30 14:15:12 [notice] 27340#27340: signal 29 (SIGIO) received
2025/12/30 14:26:24 [crit] 27342#27342: *73 SSL_do_handshake() failed (SSL: error:0A000172:SSL routines::wrong signature type) while SSL handshaking, client:
 50.116.49.249, server: 0.0.0.0:443
2025/12/30 15:44:19 [error] 27342#27342: *154 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:44:19 [warn] 27342#27342: *154 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
[ec2-user@ip-10-0-10-89 ~]$
```

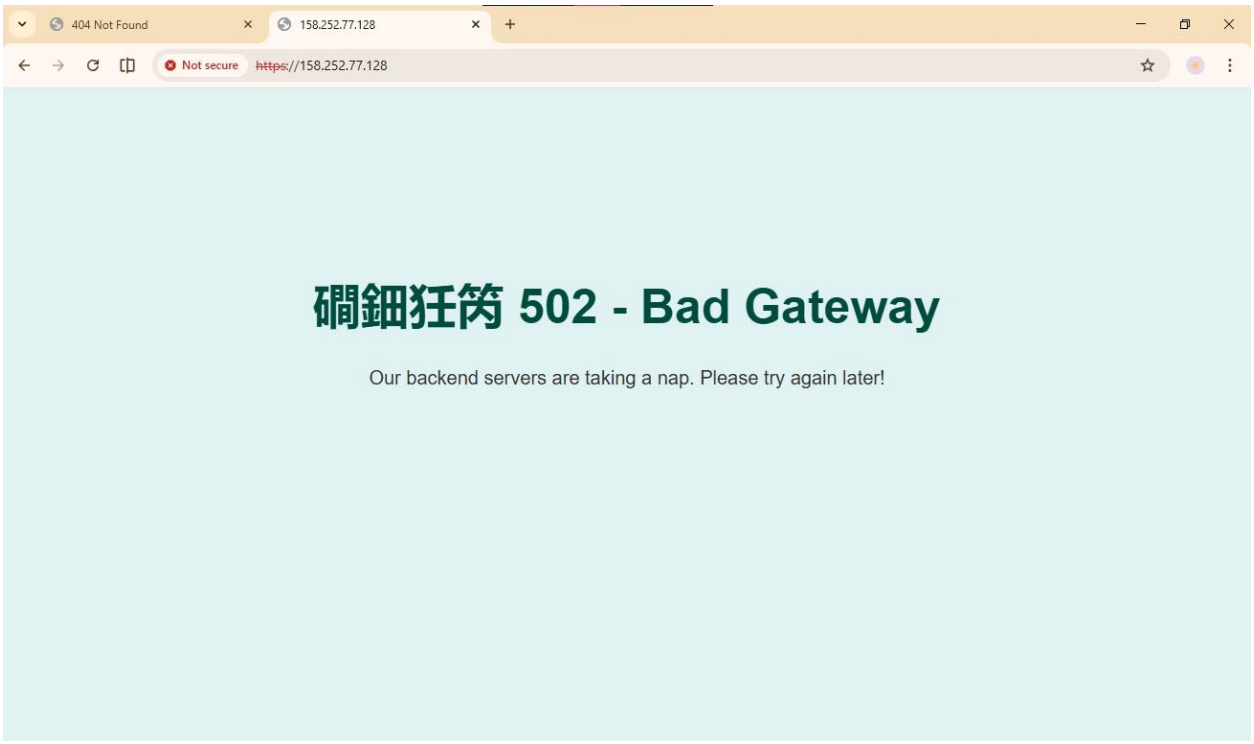
Bonus Tasks (10 marks extra credit)

Bonus 1: Custom Error Pages (3 marks)

Create custom error pages for Nginx (404, 502, 503).

Tasks:

- Create custom HTML error pages
- Configure Nginx to use custom error pages
- Test by accessing non-existent URL and stopping backend servers



Bonus 2: Implement Rate Limiting (3 marks)

Tasks:

- Implement rate limiting
- Test with rapid requests
- Show 429 (Too Many Requests) response

```
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;
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=1r/s;

    # Basic settings
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 65;
"/etc/nginx/nginx.conf" 133L, 4019B
```

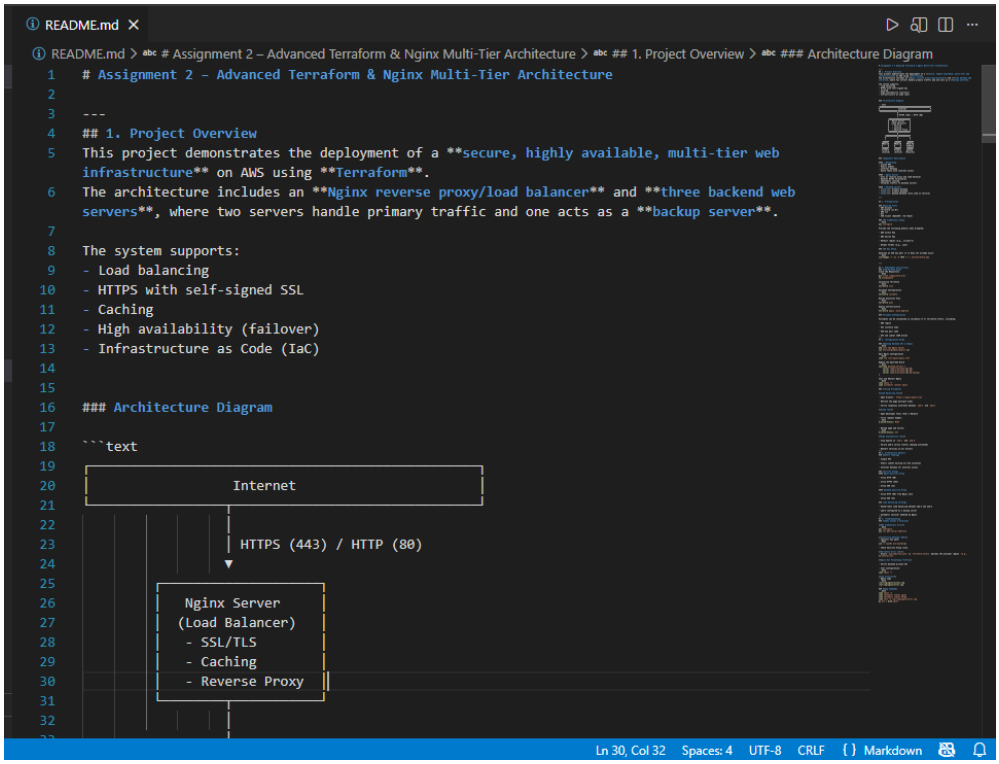
Bonus 3: Health Check Automation (4 marks)

```
[ec2-user@myapp-webserver ~]$ cat health_monitor.sh
#!/bin/bash

LOG_FILE="/home/ec2-user/health_monitor.log"

echo "Starting Health Monitor..." >> $LOG_FILE

while true; do
    # Check HTTP status code
    HTTP_STATUS=$(curl -o /dev/null -s -w "%{http_code}\n" http://localhost)
    TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

    if [ "$HTTP_STATUS" == "200" ]; then
        echo "[$TIMESTAMP] ✅ Status: $HTTP_STATUS - Healthy" >> $LOG_FILE
    else
        echo "[$TIMESTAMP] ❌ Status: $HTTP_STATUS - DOWN! Restarting Apache..." >> $LOG_FILE
        sudo systemctl start httpd

        # Verify restart
        sleep 2
        NEW_STATUS=$(curl -o /dev/null -s -w "%{http_code}\n" http://localhost)
        echo "[$TIMESTAMP]   Restart result: $NEW_STATUS" >> $LOG_FILE
    fi

    # Wait 10 seconds (using 10 instead of 30 for faster testing)
    sleep 10
done
[ec2-user@myapp-webserver ~]$ ./health_monitor.sh &
[2] 37861
[ec2-user@myapp-webserver ~]$
```

```
[ec2-user@myapp-webserver ~]$ tail -f health_monitor.log
[2025-12-30 17:38:56] ✅ Status: 200 - Healthy
[2025-12-30 17:39:02] ✅ Status: 200 - Healthy
[2025-12-30 17:39:06] ✅ Status: 200 - Healthy
[2025-12-30 17:39:12] ✅ Status: 200 - Healthy
[2025-12-30 17:39:16] ✅ Status: 200 - Healthy
[2025-12-30 17:39:22] ✅ Status: 200 - Healthy
[2025-12-30 17:39:26] ✅ Status: 200 - Healthy
[2025-12-30 17:39:32] ❌ Status: 000 - DOWN! Restarting Apache...
[2025-12-30 17:39:32]   Restart result: 200
[2025-12-30 17:39:36] ✅ Status: 200 - Healthy
[2025-12-30 17:39:44] ✅ Status: 200 - Healthy
[2025-12-30 17:39:46] ✅ Status: 200 - Healthy
[2025-12-30 17:39:54] ✅ Status: 200 - Healthy
[2025-12-30 17:39:56] ✅ Status: 200 - Healthy
[2025-12-30 17:40:04] ✅ Status: 200 - Healthy
[2025-12-30 17:40:07] ✅ Status: 200 - Healthy
[2025-12-30 17:40:14] ✅ Status: 200 - Healthy
[2025-12-30 17:40:17] ✅ Status: 200 - Healthy
[2025-12-30 17:40:24] ✅ Status: 200 - Healthy
[2025-12-30 17:40:27] ✅ Status: 200 - Healthy
```

Part 6: Documentation & Cleanup (10 marks)

6.1 README Documentation (5 marks)



6.2 Infrastructure Cleanup (5 marks)

Properly destroy all resources and verify cleanup.

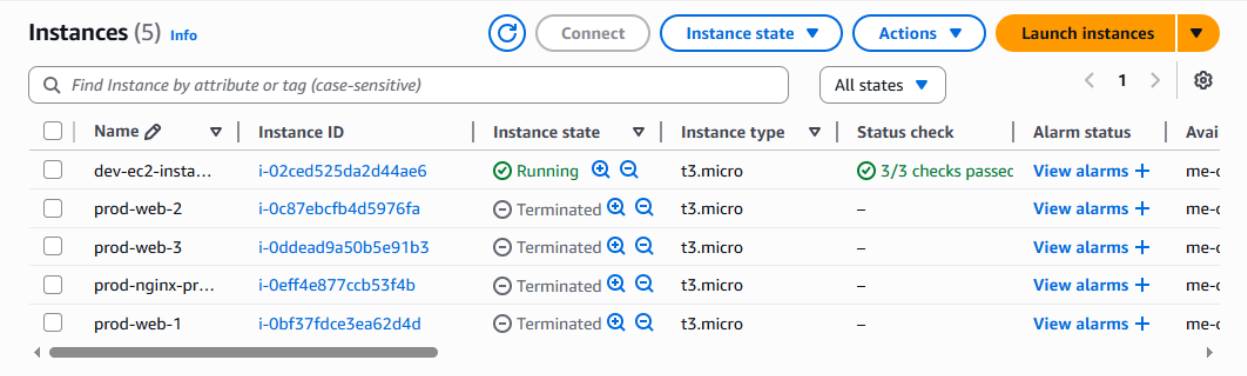
Tasks:

- Run terraform destroy
- Confirm resource deletion in AWS Console
- Verify no remaining resources
- Document the destruction process

```
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_subnet.myapp_subnet: Destruction complete after 1s
module.security.aws_security_group.nginx_sg: Destruction complete after 1s
module.networking.aws_vpc.myapp_vpc: Destroying... [id=vpc-011d0a54436bf2c22]
module.networking.aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 15 destroyed.

Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$
```

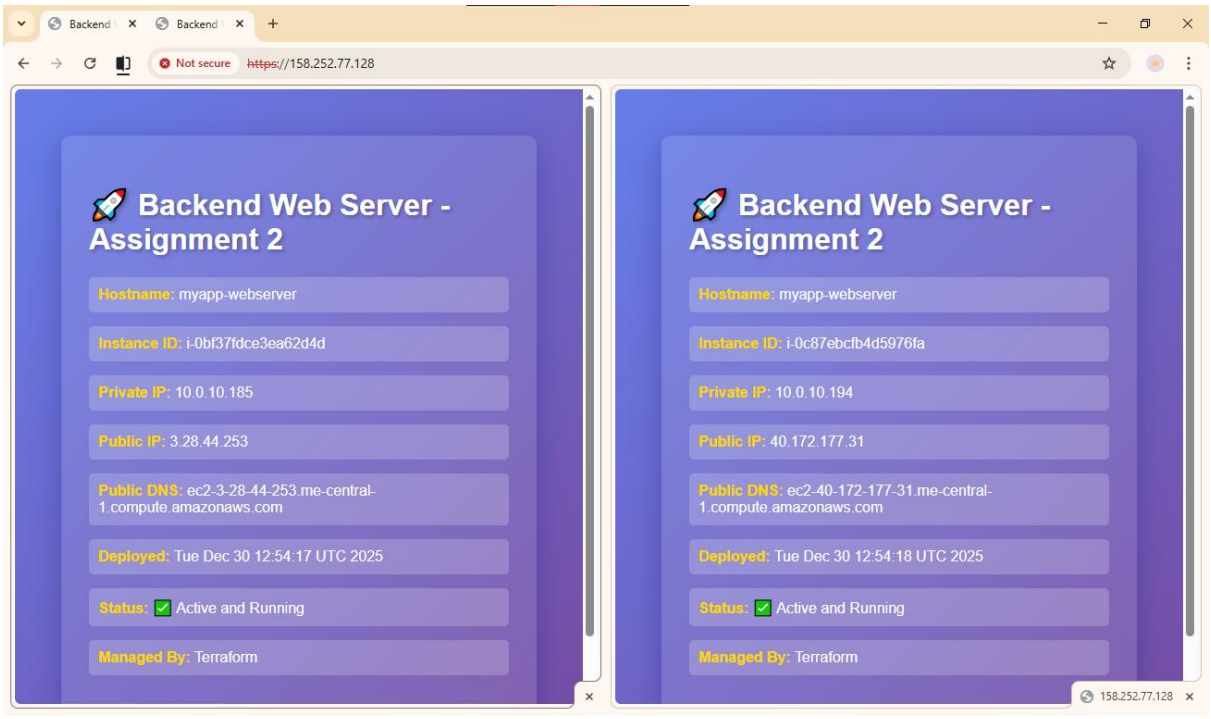
```
Rughma@DESKTOP-LJFKHUN MINGW64 /d/docs/Assignment2
$ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.3",
  "serial": 40,
  "lineage": "5e4c4843-6ab3-3a11-0dde-f9b10742df8f",
  "outputs": {},
  "resources": [],
  "check_results": [
    {
      "object_kind": "var",
      "config_addr": "var.vpc_cidr_block",
      "status": "unknown",
      "objects": null
    },
    {
      "object_kind": "var",
      "config_addr": "var.subnet_cidr_block",
      "status": "unknown",
      "objects": null
    }
  ]
}
```



4. Testing Results

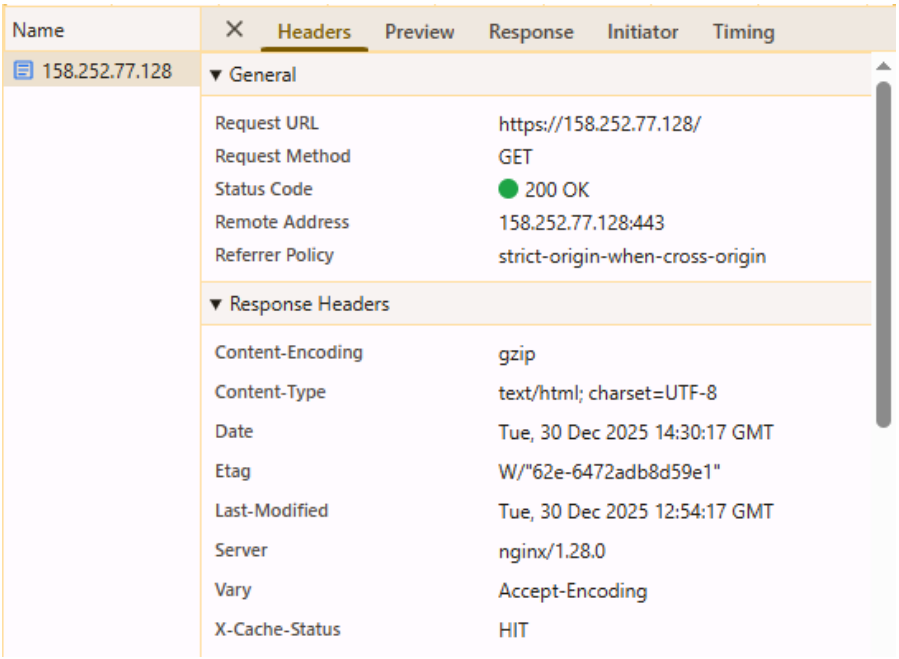
4.1 Load Balancing Test

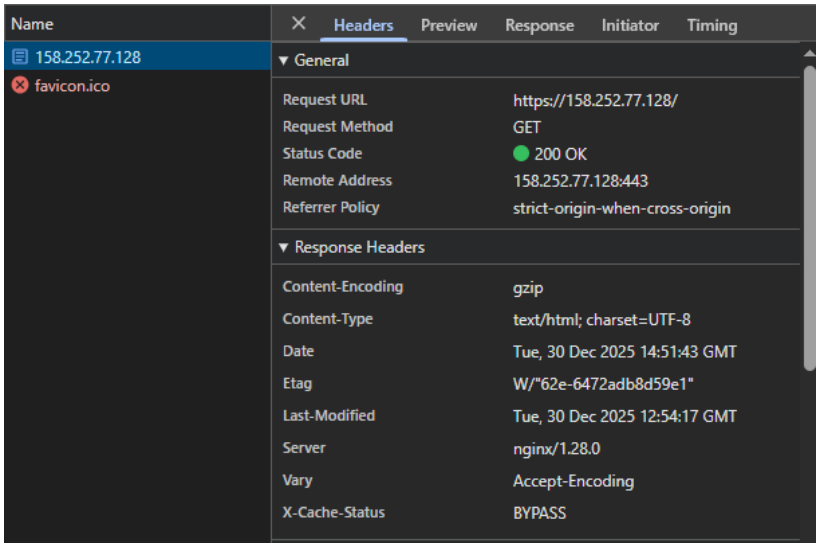
- Requests alternated between web-1 and web-2
- web-3 remained inactive as backup



4.2 Cache Performance Test

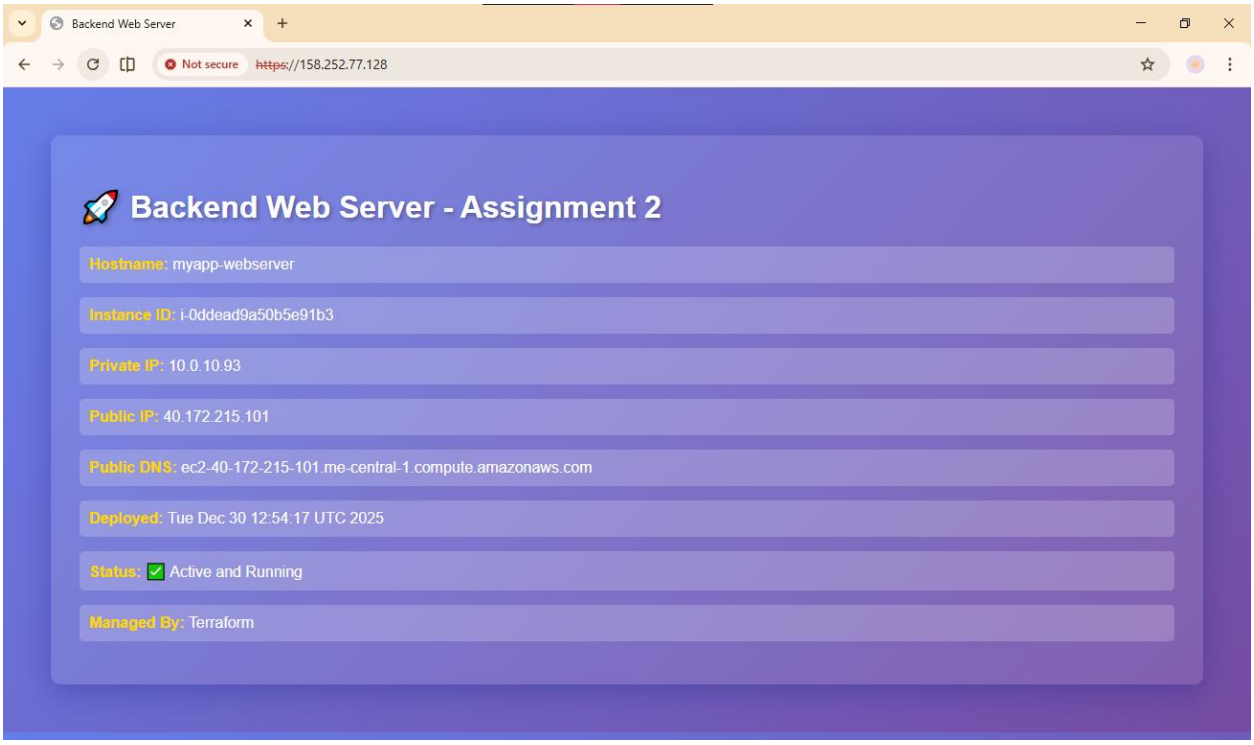
- First request: X-Cache-Status: MISS
- Subsequent requests: X-Cache-Status: HIT





4.3 High Availability Test

- Stopping Apache on web-1 and web-2 activated web-3
- Traffic resumed normally after restart



```
[ec2-user@ip-10-0-10-89 ~]$ sudo tail -n 20 /var/log/nginx/error.log
2025/12/30 14:14:11 [notice] 27339#27339: using the "epoll" event method
2025/12/30 14:14:11 [notice] 27339#27339: nginx/1.28.0
2025/12/30 14:14:11 [notice] 27339#27339: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/30 14:14:11 [notice] 27339#27339: getrlimit(RLIMIT_NOFILE): 65535:65535
2025/12/30 14:14:11 [notice] 27340#27340: start worker processes
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27341
2025/12/30 14:14:11 [notice] 27340#27340: start worker process 27342
2025/12/30 14:14:11 [notice] 27340#27340: start cache manager process 27343
2025/12/30 14:14:11 [notice] 27340#27340: start cache loader process 27344
2025/12/30 14:15:11 [notice] 27344#27344: http file cache: /var/cache/nginx 0.004M, bsize: 4096
2025/12/30 14:15:12 [notice] 27340#27340: signal 17 (SIGCHLD) received from 27344
2025/12/30 14:15:12 [notice] 27340#27340: cache loader process 27344 exited with code 0
2025/12/30 14:15:12 [notice] 27340#27340: signal 29 (SIGIO) received
2025/12/30 14:26:24 [crit] 27342#27342: *73 SSL_do_handshake() failed (SSL: error:0A000172:SSL routines::wrong signature type) while SSL handshaking, client:
 50.116.49.249, server: 0.0.0.0:443
2025/12/30 15:44:19 [error] 27342#27342: *154 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:44:19 [warn] 27342#27342: *154 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.194:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [error] 27342#27342: *167 connect() failed (111: Connection refused) while connecting to upstream, client: 154.192.0.128, server: _, requ
est: "GET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
2025/12/30 15:45:59 [warn] 27342#27342: *167 upstream server temporarily disabled while connecting to upstream, client: 154.192.0.128, server: _, request: "G
ET / HTTP/2.0", upstream: "http://10.0.10.185:80/", host: "158.252.77.128"
[ec2-user@ip-10-0-10-89 ~]$
```

4.4 Security Testing

- SSL certificate verified
- Security headers confirmed
- HTTP redirected to HTTPS

```
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 729F5EF620634B1BF39F82402855BB36AAF4B32363F063B76AF0B81672D3AC9B
    Session-ID-ctx:
    Resumption PSK: A3B74206CC70757A8EE63791524EBB721976E40A3794429D99EEA558193D1BB358DD53AC47B1D3EAF88FA5602AADED09
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - a5 80 d2 62 3f 60 da 16-fa 8f c4 47 46 fe de e2   ...b?`.....GF...
    0010 - 6b 58 7f 8f 43 02 b8 aa-27 d7 c6 5b b1 42 49 f4   kX..C...'..[.BI.
    0020 - 36 b0 ca f5 bf 60 22 b1-8a b6 e3 6b 0c 48 6b 9b   6....`"....k.Hk.
    0030 - dd 59 68 07 78 cf bc 02-74 69 a8 70 5f c0 4c 08   .Yh.x...ti.p_.L.
    0040 - 24 33 25 48 8c c7 c8 b5-3f 66 04 77 19 e9 c2 f0   $3%H....?f.w....
    0050 - be d2 79 b5 53 0b 48 0b-81 82 3f 09 c3 a3 5b cc   ..y.S.H...?...[.
    0060 - 54 96 a3 f0 ae 22 2f bb-b2 c1 cc 05 ee cc 0f 34   T...."/........4
    0070 - 91 fc af 2d e6 23 45 9e-9c 2f cc 0a cc a9 60 d7   ...-.#E../....`.
    0080 - cf 39 72 dc 94 78 54 76-67 f1 84 7f a4 95 f9 ba   .9r..xTvg.......
    0090 - ed 08 ec ce 22 1f 03 df-fb 7e 2b df 79 9b 86 b6   ....".....~+.y...
    00a0 - f9 b3 85 c3 83 76 80 88-00 56 b1 3a be 9d 41 09   .....v...V.:..A.
    00b0 - 4c 6e a5 63 1a 4d bd 79-9e b9 c7 88 6c 29 4e e0   Ln.c.M.y....l)N.
    00c0 - b4 0c c7 08 1f fb bf 43-ad 88 1c d4 e0 9a e2 47   .......C.......G
    00d0 - aa 49 82 bd 49 60 6a 42-b3 e3 6b eb 18 33 9a e2   .I..I`jB..k..3..

    Start Time: 1767110810
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
```

```
[ec2-user@ip-10-0-10-89 ~]$ curl -I -k https://158.252.77.128
HTTP/2 200
server: nginx/1.28.0
date: Tue, 30 Dec 2025 16:05:34 GMT
content-type: text/html; charset=UTF-8
content-length: 1586
vary: Accept-Encoding
last-modified: Tue, 30 Dec 2025 12:54:18 GMT
etag: "632-6472adb9ca5aa"
x-cache-status: MISS
accept-ranges: bytes

[ec2-user@ip-10-0-10-89 ~]$
```

```
[ec2-user@ip-10-0-10-89 ~]$ curl -I http://158.252.77.128
\\HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 16:04:38 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://158.252.77.128/

[ec2-user@ip-10-0-10-89 ~]$
```

5. Challenges & Solutions

Challenge 1: Terraform command not found

- **Solution:** Added Terraform to system PATH and restarted terminal

Challenge 2: AWS credential errors

- **Solution:** Installed AWS CLI and configured credentials

Challenge 3: Incorrect Availability Zone

- **Solution:** Updated AZ to match the configured region

Challenge 4: Nginx Syntax Error

- **Solution:** Sudo nginx -t which pointed to a syntax error on line 53. Upon inspection, I found an accidental typo (i}) caused by a vim keystroke error.

Challenge 5: Availability Zone Mismatch

- **Solution:** Updated the availability_zone variable to me-central-1a to match the provider's region

Lessons Learned

- Importance of region consistency
- Benefits of Infrastructure as Code
- Debugging using logs is critical

6. Conclusion

This assignment successfully demonstrated the deployment of a secure, scalable, and highly available web infrastructure using Terraform and AWS. Through this project, practical skills in cloud networking, automation, server configuration, and testing were developed.

Future improvements may include:

- Using ACM certificates
- Auto Scaling Groups
- CloudWatch monitoring
- CI/CD integration

7. Appendices

Appendix A – Repository Link with Complete Code Listings

Github Repository-assignment2

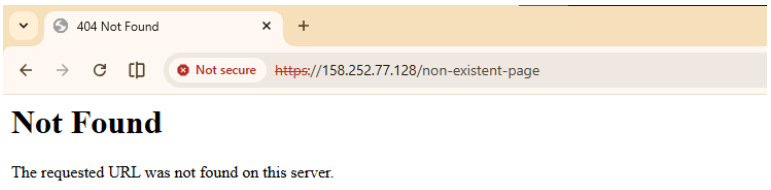
Appendix B: Configuration Files

Final Nginx Configuration (/etc/nginx/nginx.conf) *The upstream block modified after deployment:*

```
    # Upstream backend servers
    # PLACEHOLDER: Update these IPs after deployment
    upstream backend_servers {
        # Primary servers (active load balancing)
        server 10.0.10.185:80;
        server 10.0.10.194:80;

        # Backup server (only used when primary servers are down)
        server 10.0.10.93:80 backup;
    }
```

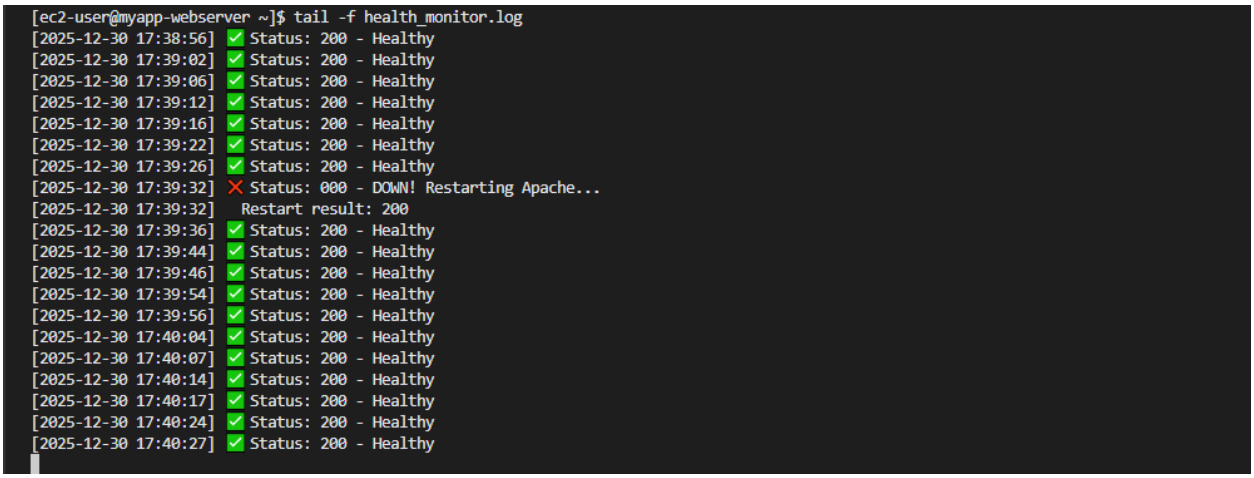
Appendix C: Additional Screenshots

Bonus 1: Custom 404 Error Page



Bonus 2: Rate Limit Testing (429 Too Many Requests)

Bonus 3: Health Monitor Log



Appendix D: References

1. **Terraform Documentation:** https://registry.terraform.io/providers/hashicorp/aws/latest/docs
2. **Nginx Documentation:** https://nginx.org/en/docs/
3. **AWS EC2 User Guide:** https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/
4. **Amazon Linux 2023:** https://docs.aws.amazon.com/linux/al2023/

THE END