Fatima Jinnah Women Univeristy

Course Title : *Cloud Computing*

Assignment 2

Submitted by :

Lujain Zia (2023-BSE-034)

Submitted to :

Sir Waqas Saleem

Date : December 28, 2025

Part 1:

Infrastructure Setup (25 marks)

1.1 Project Structure (5 marks)

Create a well-organized Terraform project with the following structure:

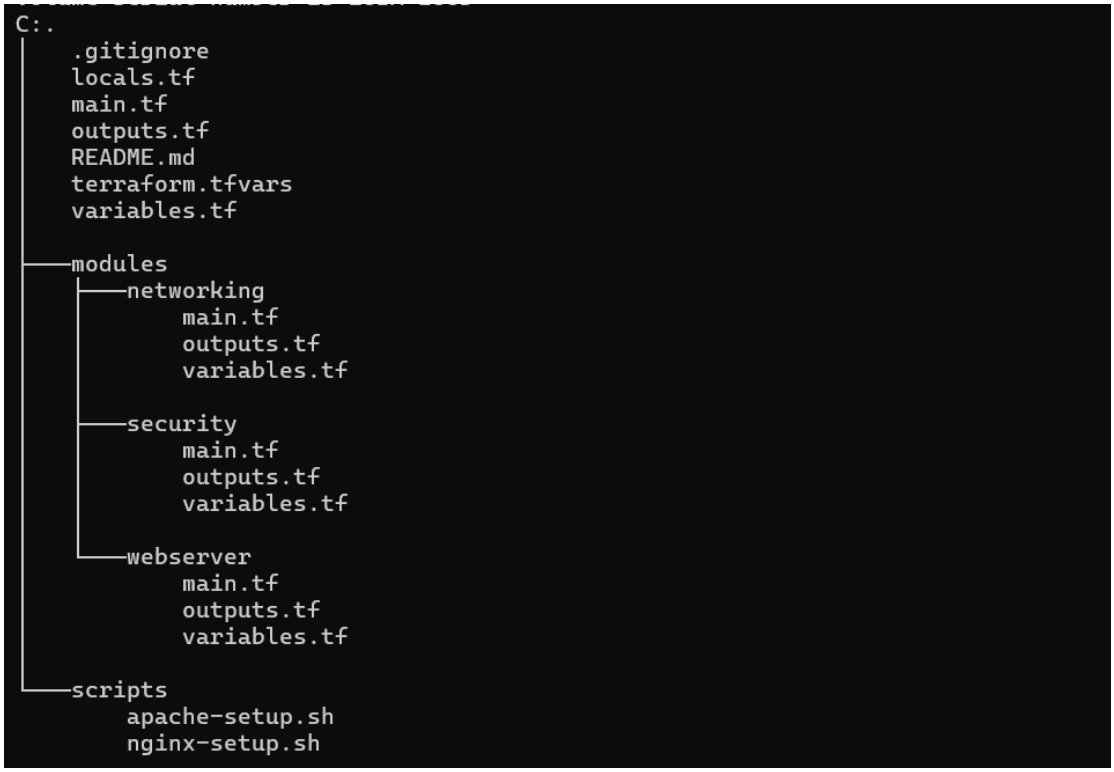
```
Assignment2/
├── main.tf
├── variables.tf
├── outputs.tf
├── locals.tf
├── terraform.tfvars
├── . gitignore
├── modules/
│   ├── networking/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs. tf
│   ├── security/
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs. tf
│   └── webserver/
│       ├── main.tf
│       ├── variables.tf
│       └── outputs.tf
├── scripts/
│   ├── nginx-setup.sh
│   └── apache-setup.sh
└── README.md
```

Tasks:

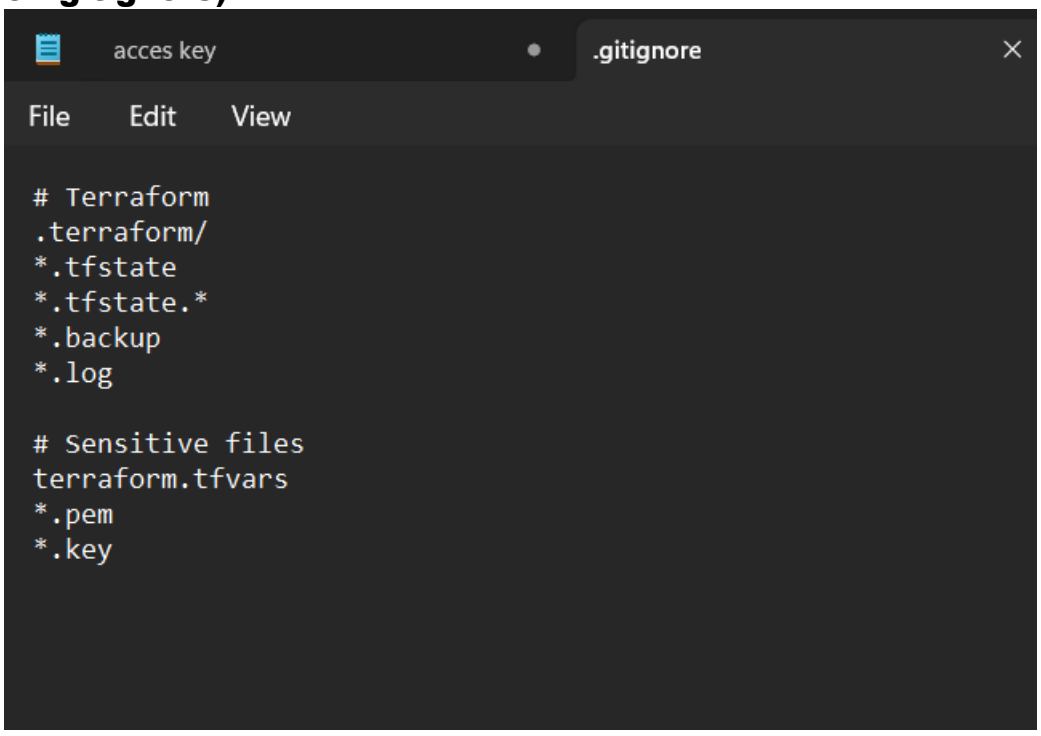
- Create all necessary files and directories
- Implement proper .gitignore to exclude sensitive files
- Document the project structure in README.md

Deliverables:

Screenshot: assignment_part1_project_structure.png (tree command output)



Screenshot: assignment_part1_gitignore.png (content of .gitignore)



1.2 Variable Configuration (5 marks)

Define all required variables in variables.tf:

Required Variables:

vpc_cidr_block (string)
subnet_cidr_block (string)

availability_zone (string)
env_prefix (string)
instance_type (string)
public_key (string)
private_key (string)
backend_servers (list of objects with name and script_path)

Tasks:

Add validation rules for CIDR blocks

Add descriptions for all variables

Set appropriate defaults where applicable

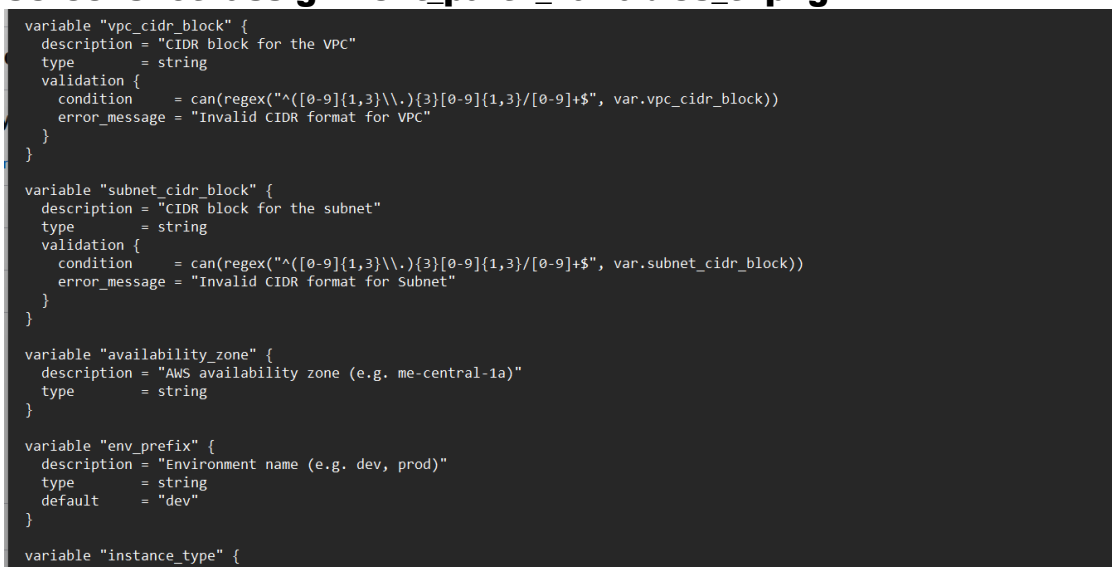
Populate terraform.tfvars with your values

Sample terraform.tfvars structure:

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"

Deliverables:

Screenshot: assignment_part1_variables_tf.png



```
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]+$", var.vpc_cidr_block))
    error_message = "Invalid CIDR format for VPC"
  }
}

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]+$", var.subnet_cidr_block))
    error_message = "Invalid CIDR format for Subnet"
  }
}

variable "availability_zone" {
  description = "AWS availability zone (e.g. me-central-1a)"
  type        = string
}

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

variable "instance_type" {
```

Screenshot: assignment_part1_terraform_tfvars.png

File Edit View

```
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        = "C:/Users/user/.ssh/id_ed25519.pub"
private_key       = "C:/Users/user/.ssh/id_ed25519"

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

1.3 Networking Module (5 marks)

Create a networking module that provisions:
VPC with specified CIDR block
Subnet with public IP assignment enabled
Internet Gateway
Route table with default route to IGW
Associate route table with subnet
Module Location: modules/networking/
Required Outputs:
vpc_id
subnet_id
igw_id
route_table_id
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

Deliverables:
Screenshot: assignment_part1_networking_module_main.png

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

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

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

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

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

resource "aws_route_table" "main" {
  vpc_id = aws_vpc.main.id

  route {
```

Screenshot: assignment_part1_networking_module_outputs.png

```
aws_key_pair.ssh_key: Creating...
aws_vpc.myapp_vpc: Creating...
aws_key_pair.ssh_key: Creation complete after 1s [id=serverkey]
aws_vpc.myapp_vpc: Creation complete after 2s [id=vpc-0e27d646a98e5e382]
aws_internet_gateway.myapp_igw: Creating...
aws_subnet.myapp_subnet_1: Creating...
aws_default_security_group.myapp_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 0s [id=igw-04f131a91c7192065]
aws_route_table.myapp_route_table: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-0f568f6c021d0dfb9]
aws_route_table.myapp_route_table: Creation complete after 2s [id=rtb-00a3481e95c03304e]
aws_route_table_association.myapp_route_table_assoc: Creating...
aws_route_table_association.myapp_route_table_assoc: Creation complete after 0s [id=rtbassoc-088799ddd9d1d75cf]
aws_default_security_group.myapp_sg: Creation complete after 2s [id=sg-07c94d09956adbc84]
aws_instance.myapp-server: Creating...
aws_instance.myapp-server: Still creating... [00m10s elapsed]
aws_instance.myapp-server: Creation complete after 13s [id=i-0ad257af3bed851d5]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
```

1.4 Security Module (5 marks)

Create a security module that provisions:

Two Security Groups:

Nginx Security Group (for reverse proxy/load balancer):

Ingress: Port 22 (SSH) from your IP only
Ingress: Port 80 (HTTP) from anywhere (0.0.0.0/0)
Ingress: Port 443 (HTTPS) from anywhere (0.0.0.0/0)
Egress: All traffic

Backend Security Group (for web servers):

Ingress: Port 22 (SSH) from your IP only
Ingress: Port 80 (HTTP) from Nginx security group only
Egress: All traffic

Module Location: modules/security/

Required Variables:

vpc_id

env_prefix
my_ip
Required Outputs:
nginx_sg_id
backend_sg_id

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

Deliverables:

Screenshot: assignment_part1_security_module. Png

File Edit View

```
resource "aws_security_group" "nginx_sg" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Security group for NGINX load balancer"
  vpc_id      = var.vpc_id

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

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

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

Screenshot: assignment_part1_security_groups_console.png (AWS Console)

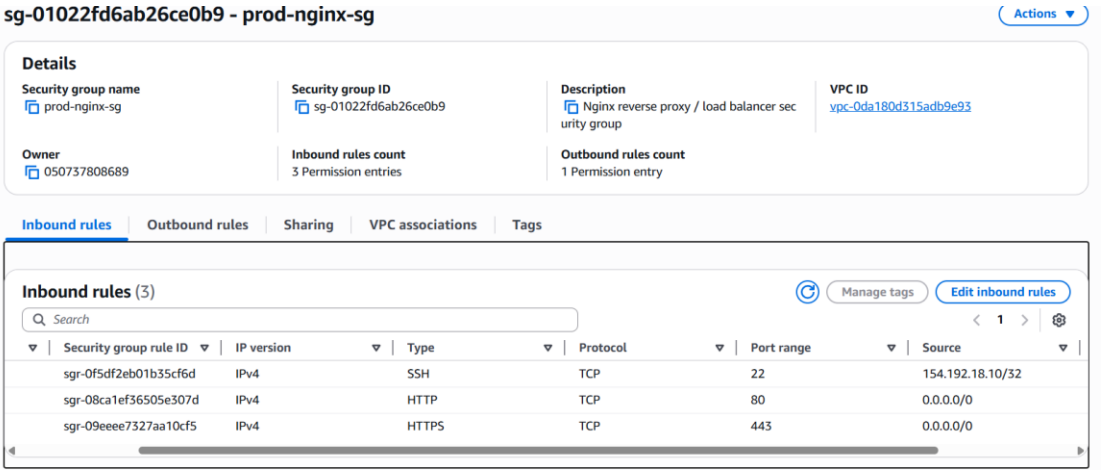
Security Groups (5) Info

Actions | Export security groups to CSV | Create security group

Find security groups by attribute or tag

1

Name	Security group ID	Security group name	VPC ID	Description
prod-nginx-sg	sg-01022fd6ab26ce0b9	prod-nginx-sg	vpc-0da180d315adb9e93	Nginx reverse proxy
dev-sg	sg-0ee9dce4b4e7c9999	default	vpc-01825459d38b35672	default VPC security
prod-backend-sg	sg-07a0c33cf9396db51	prod-backend-sg	vpc-0da180d315adb9e93	Backend web server
–	sg-0851e2a278e0ffd34	default	vpc-01b7d554ea60c902d	default VPC security
–	sg-0a6a674172092bd49	default	vpc-0da180d315adb9e93	default VPC security



1.5 Locals Configuration (5 marks)

Create locals.tf with:
Dynamic IP detection for my_ip
Resource naming conventions
Common tags
Backend server configurations
Example:

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

Tasks:

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

Deliverables:

Screenshot: assignment_part1_locals_tf.png

```
# Fetch your public IP address dynamically from icanhazip.com
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

locals {
  # Dynamic IP with /32 CIDR for security group
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"

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

  # List of backend servers with names, suffix, and scripts
  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"
```

Part 2:

Webserver Module (15 marks)

2.1 Module Design (10 marks)

Create a reusable webserver module in modules/webserver/

Module Requirements:

Variables:

env_prefix
instance_name
instance_type
availability_zone
vpc_id
subnet_id

security_group_id
public_key
script_path
instance_suffix
common_tags
Resources to Create:
AWS Key Pair (unique per instance using suffix)
EC2 Instance with:
Specified AMI (Amazon Linux 2023)
Public IP enabled
User data from script file
Proper tags
Outputs:
instance_id
public_ip
private_ip
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
Deliverables:
Screenshot: assignment_part2_webserver_module_variables.png

```
variable "subnet_id" {
  type        = string
  description = "Subnet ID to launch instance in"
}

variable "security_group_id" {
  type        = string
  description = "Security Group ID to associate with instance"
}

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

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

variable "instance_suffix" {
  type        = string
  description = "Unique suffix for the instance"
}

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

Screenshot: assignment_part2_webserver_module_main.png

```
# Read public key content
resource "tls_private_key" "example" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_key_pair" "web_key" {
  key_name   = "${var.env_prefix}-${var.instance_name}-${var.instance_suffix}-key"
  public_key = file(var.public_key)

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

resource "aws_instance" "web_instance" {
  ami                         = "ami-05524d6658fcf35b6" # Amazon Linux 2023 (update if needed)
  instance_type               = var.instance_type
  subnet_id                   = var.subnet_id
  vpc_security_group_ids      = [var.security_group_id]
  key_name                    = aws_key_pair.web_key.key_name
  associate_public_ip_address = true
  availability_zone           = var.availability_zone

  user_data = file(var.script_path)

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

Screenshot: assignment_part2_webserver_module_outputs.png

```
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.web_instance.id
}

output "public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.web_instance.public_ip
}

output "private_ip" {
  description = "Private IP address of the EC2 instance"
  value       = aws_instance.web_instance.private_ip
}
```

2.2 Module Usage (5 marks)

In root main.tf, instantiate the webserver module for:
One Nginx server (using nginx-setup.sh)
Three backend servers (web-1, web-2, web-3 using apache-setup.sh)
Tasks:

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

Deliverables:
Screenshot: assignment_part2_main_tf_modules.png

```
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 (Apache Web 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 (20 marks)

3.1 Apache Backend Server Script (10 marks)

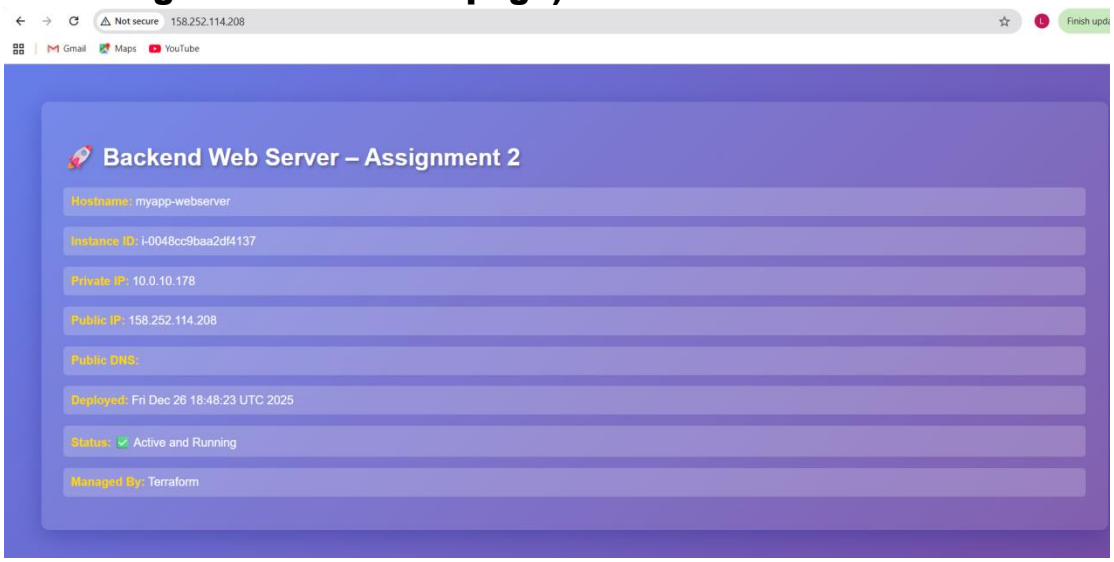
Create scripts/apache-setup.sh that:
Updates system packages
Installs Apache HTTP Server
Starts and enables Apache service
Creates custom HTML page displaying:
Server name/hostname
Private IP address
Public IP address
Public DNS hostname
Custom message indicating which backend server (web-1, web-2, or web-3)
Timestamp of deployment
Server status (Primary/Backup)
Tasks:

- Create the script with all required features
- Ensure it uses IMDSv2 for metadata
- Create visually appealing HTML output
- Make script executable
- Test script independently

Deliverables:
Screenshot: assignment_part3_apache_script.png

```
#!/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
```

Screenshot: assignment_part3_backend_webpage.png (browser showing backend server page)



3.2 Nginx Server Setup Script (10 marks)

Create scripts/nginx-setup.sh that:
Updates system packages
Installs Nginx
Generates self-signed SSL certificate
Configures Nginx with:
HTTPS on port 443
HTTP to HTTPS redirect
Upstream backend servers (web-1, web-2, web-3)
Load balancing configuration
Caching configuration

Proper logging

Security headers

Tasks:

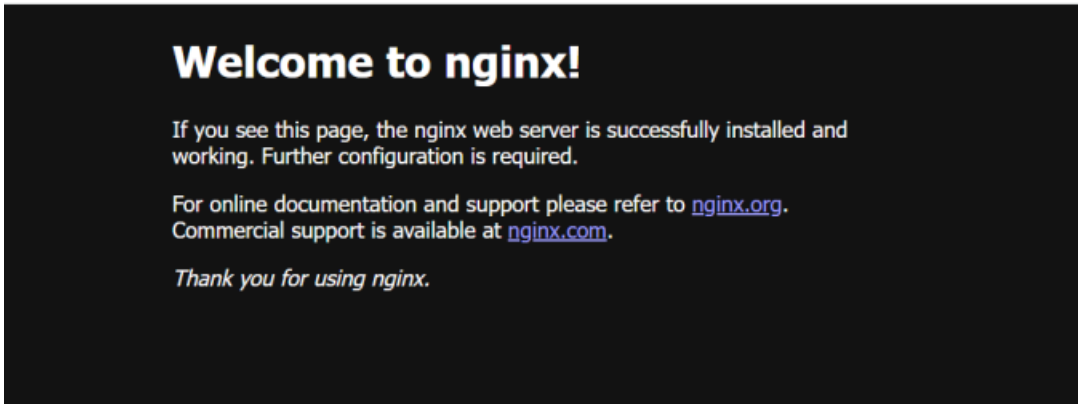
- Create script with SSL certificate generation
- Configure upstream with placeholder IPs
- Implement caching
- Add security headers
- Configure HTTP to HTTPS redirect
- Add health check endpoint

Deliverables:

Screenshot: assignment_part3_nginx_script.png

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

# Update and install Nginx and SSL
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 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"
```

Screenshot: assignment_part3_nginx_default_page.png (before backend configuration)



Part 4:

Infrastructure Deployment (15 marks)

4.1 Initial Deployment (5 marks)

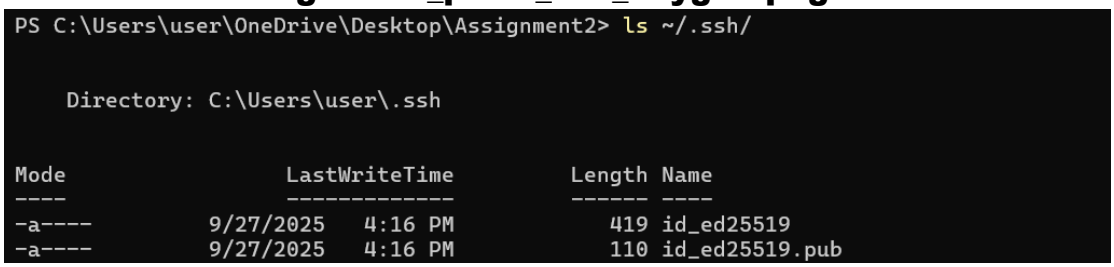
Deploy the infrastructure using Terraform.

Tasks:

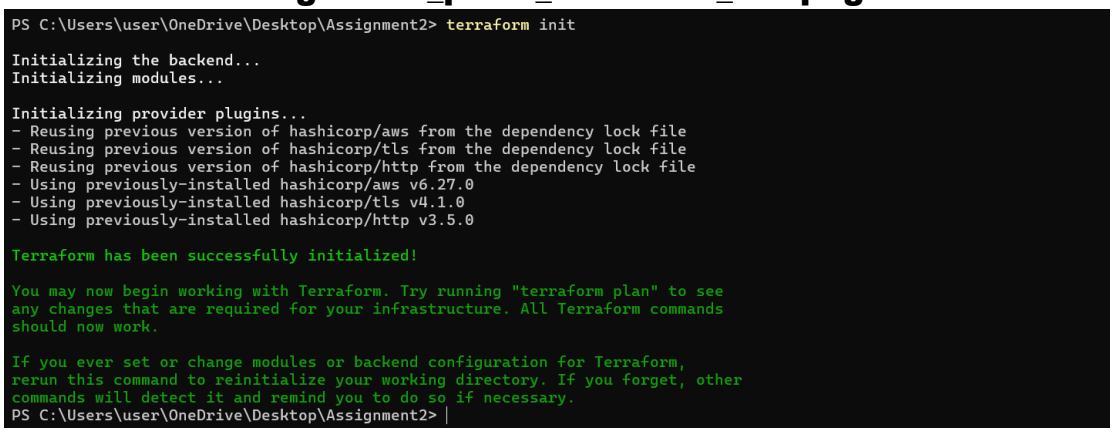
- Generate SSH key pair if not exists
- Initialize Terraform (terraform init)
- Validate configuration (terraform validate)
- Plan deployment (terraform plan)
- Apply configuration (terraform apply -auto-approve)

Deliverables:

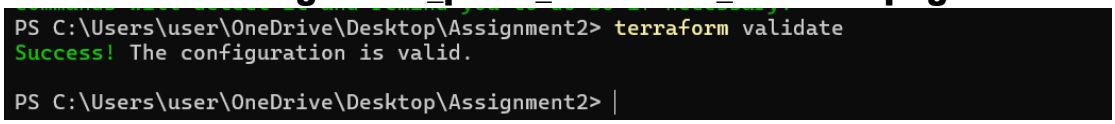
Screenshot: assignment_part4_ssh_keygen.png



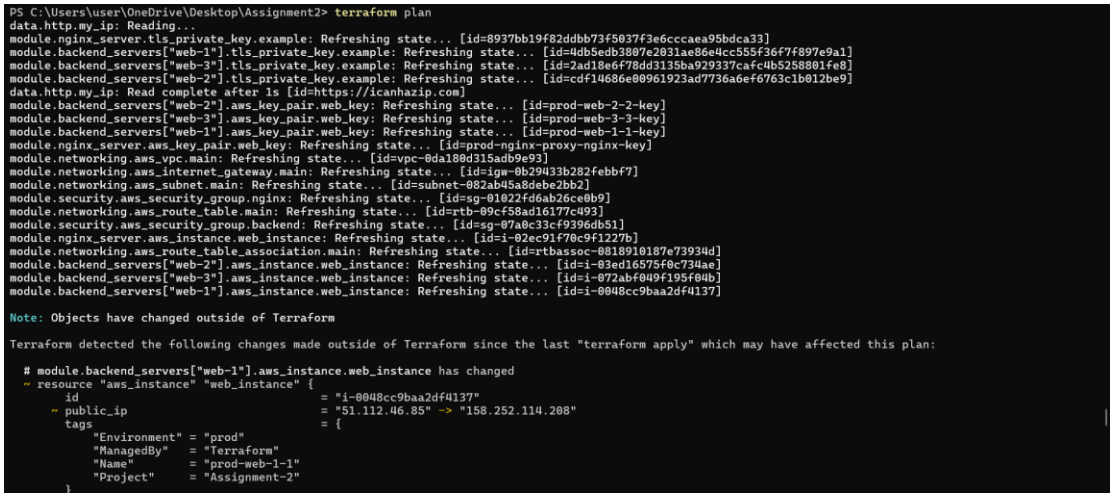
Screenshot: assignment_part4_terraform_init.png



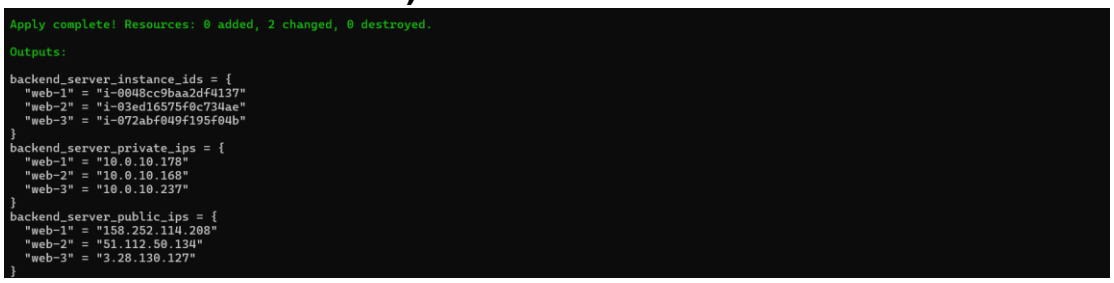
Screenshot: assignment_part4_terraform_validate.png



Screenshot: assignment_part4_terraform_plan.png



Screenshot: assignment_part4_terraform_apply.png (showing all 4 instances created)



4.2 Output Configuration (5 marks)

Tasks:

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

Commands:

terraform output
terraform output -json > outputs.json

Deliverables:

Screenshot: assignment_part4_terraform_output.png

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

Next Steps:
1. SSH into Nginx server: ssh ec2-user@158.252.32.185
2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf
3. Update backend IPs in upstream block:
   - BACKEND_IP_1: 10.0.10.178
   - BACKEND_IP_2: 10.0.10.168
   - BACKEND_IP_3: 10.0.10.237
4. Restart Nginx: sudo systemctl restart nginx
5. Test: https://158.252.32.185

Backend Servers:
- web-1: 158.252.114.208 (private: 10.0.10.178)
    - web-2: 51.112.50.134 (private: 10.0.10.168)
    - web-3: 3.28.130.127 (private: 10.0.10.237)

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

EOT
nginx_instance_id = "i-02ec91f70c9f1227b"
nginx_public_ip = "158.252.32.185"
subnet_id = "subnet-082ab45a8debe2bb2"
vpc_id = "vpc-0da180d315adb9e93"
```

Screenshot: assignment_part4_outputs_json.png


```
    "web-3": {
      "instance_id": "i-072abf049f195f04b",
      "private_ip": "10.0.10.237",
      "public_ip": "3.28.130.127"
    }
  }
 },
 "configuration_guide": {
   "sensitive": false,
   "type": "string",
   "value": "\r\n========================================\r\nDEPLOYMENT SUCCESSFUL!\r\n========================================\r\n\r\nNext Steps:\r\n1. S
H into Nginx server: ssh ec2-user@158.252.32.185\r\n2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf\r\n3. Update backend IPs in upstream block:\r\n
 BACKEND_IP_1: 10.0.10.178\r\n   - BACKEND_IP_2: 10.0.10.168\r\n   - BACKEND_IP_3: 10.0.10.237\r\n4. Restart Nginx: sudo systemctl restart nginx\r\n5. Tes
 https://158.252.32.185\r\n\r\nBackend Servers:\r\n- web-1: 158.252.114.208 (private: 10.0.10.178)\n    - web-2: 51.112.50.134 (private: 10.0.10.168)\n
 web-3: 3.28.130.127 (private: 10.0.10.237)\r\n\r\n========================================\r\n"
 },
 "nginx_instance_id": {
   "sensitive": false,
   "type": "string",
   "value": "i-02ec91f70c9f1227b"
 },
 "nginx_public_ip": {
   "sensitive": false,
   "type": "string",
   "value": "158.252.32.185"
 },
 "subnet_id": {
   "sensitive": false,
   "type": "string",
   "value": "subnet-082ab45a8debe2bb2"
 },
 "vpc_id": {
   "sensitive": false,
   "type": "string",
   "value": "vpc-0da180d315adb9e93"
 }
}
```

4.3 AWS Console Verification (5 marks)

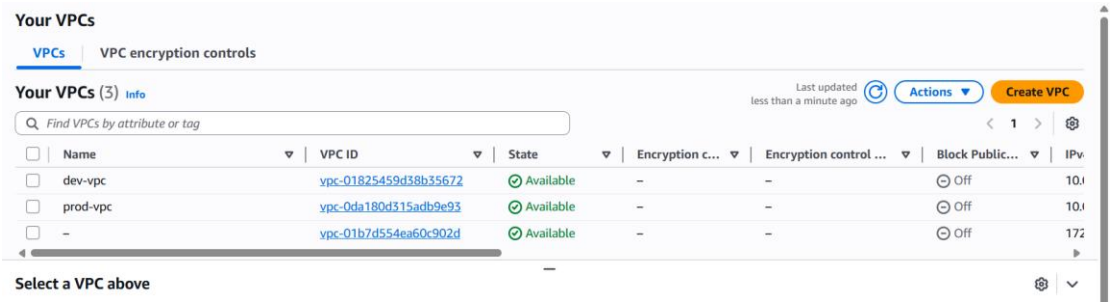
Verify all resources in AWS Console.

Tasks:

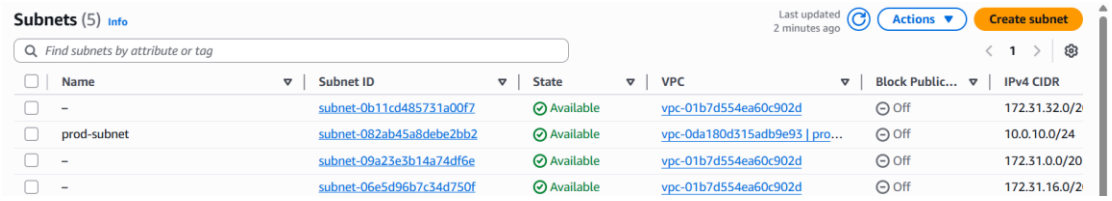
- Verify VPC created
- Verify Subnet created
- Verify Internet Gateway attached
- Verify Route Table configured
- Verify Security Groups created with correct rules
- Verify all 4 EC2 instances running
- Verify Key Pairs created

Deliverables:

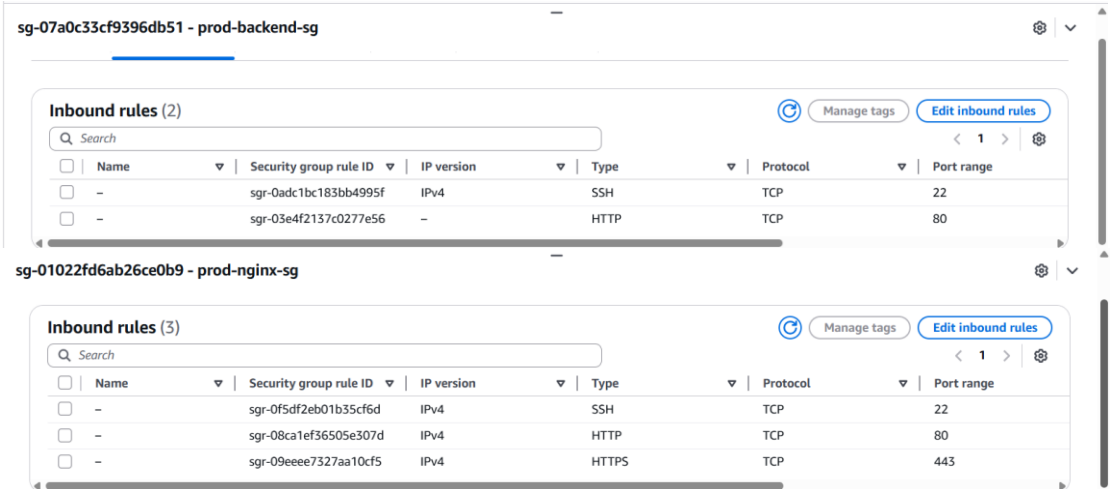
Screenshot: assignment_part4_aws_vpc. png (VPC console)



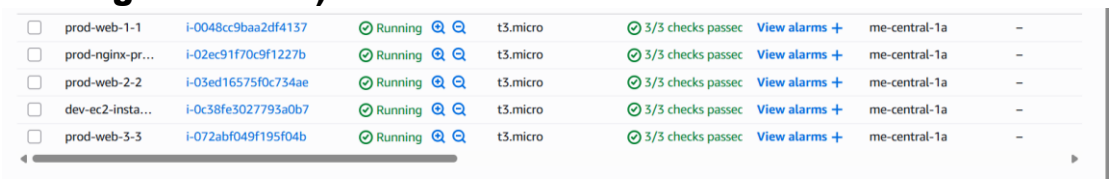
Screenshot: assignment_part4_aws_subnet.png (Subnet details)



Screenshot: assignment_part4_aws_security_groups.png (Both security groups)



Screenshot: assignment_part4_aws_instances.png (All 4 running instances)



Part 5:

Nginx Configuration & Testing (25 marks)

5.1 Update Nginx Backend Configuration (5 marks)

SSH into the Nginx server and update the configuration with actual backend IPs.

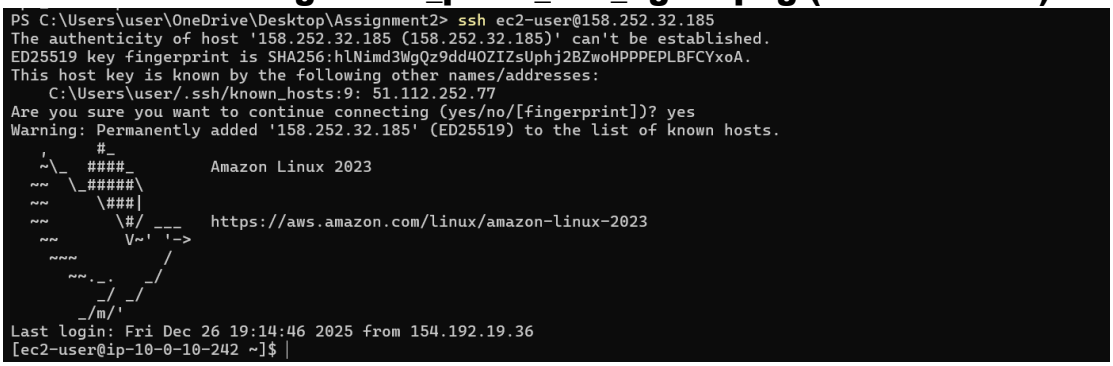
Tasks:

- SSH into Nginx server
- Edit /etc/nginx/nginx.conf

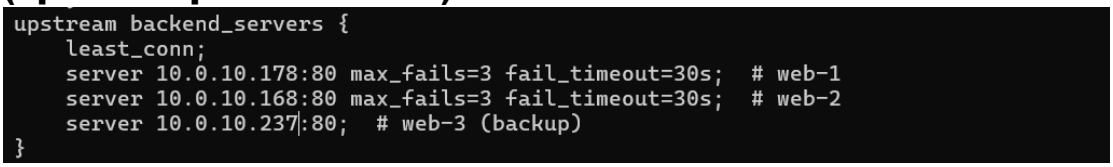
- Replace placeholder IPs with actual private IPs of backend servers
- Test Nginx configuration
- Restart Nginx service

Deliverables:

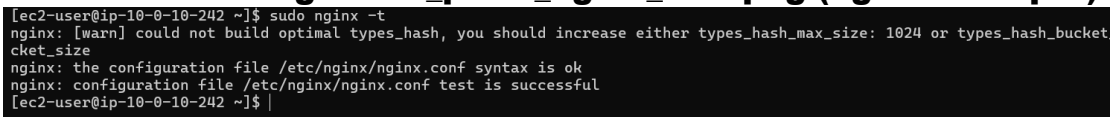
Screenshot: assignment_part5_ssh_nginx.png (SSH session)



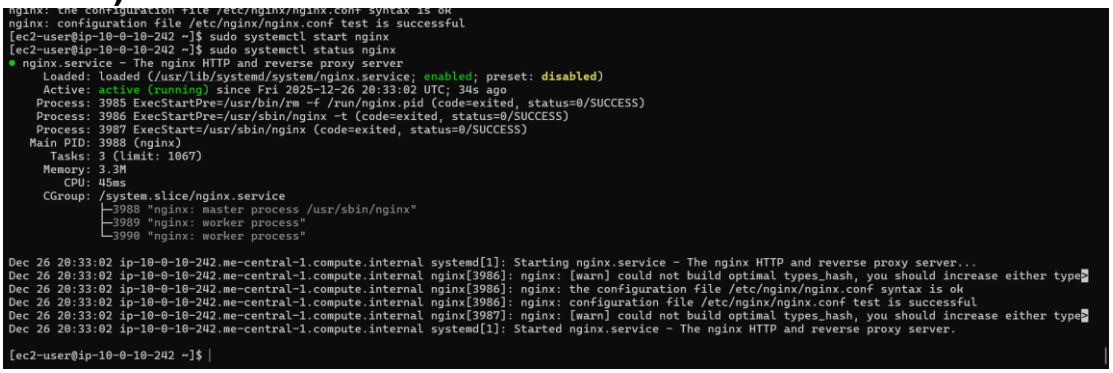
Screenshot: assignment_part5_nginx_conf_updated.png (updated upstream block)



Screenshot: assignment_part5_nginx_test.png (nginx -t output)



Screenshot: assignment_part5_nginx_restart.png (restart and status)



5.2 Test Load Balancing (5 marks)

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

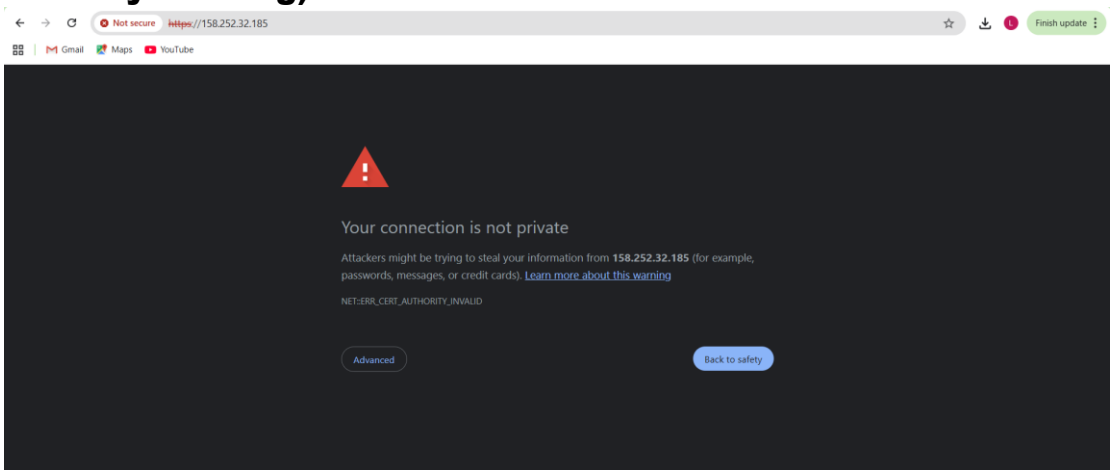
Tasks:

- 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)

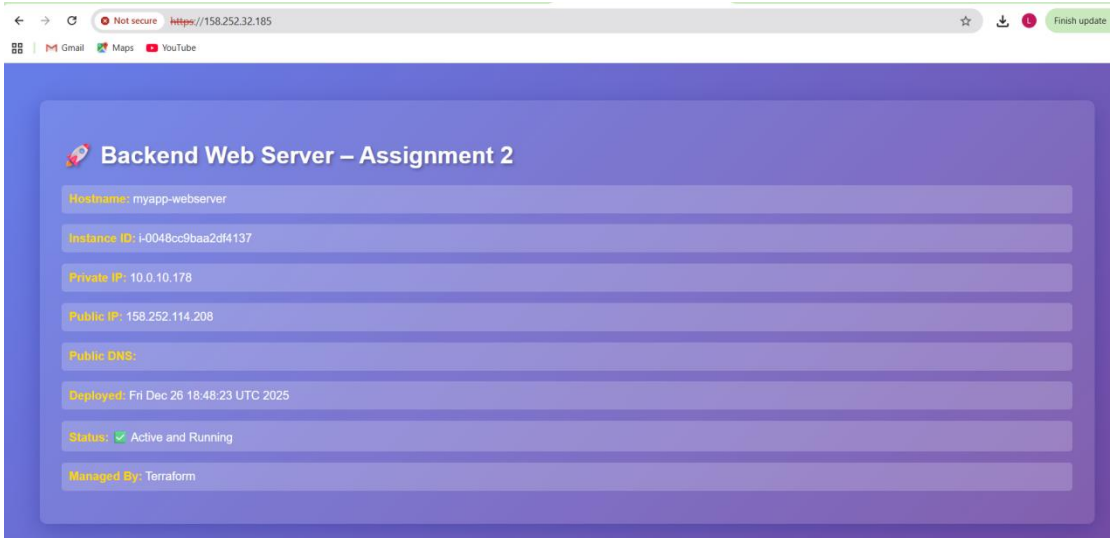
- Document the load balancing pattern

Deliverables:

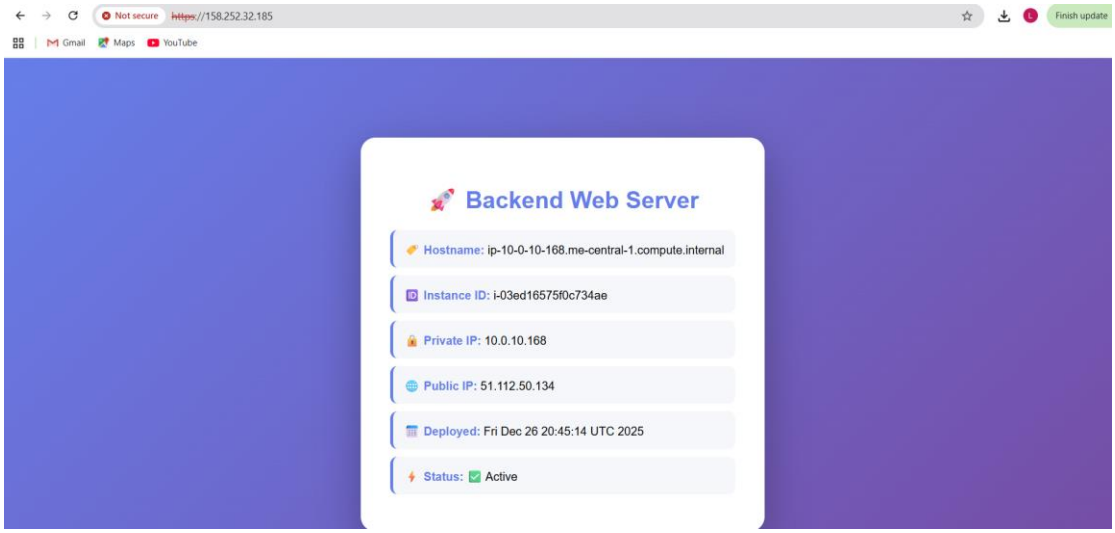
Screenshot: assignment_part5_ssl_warning.png (browser security warning)



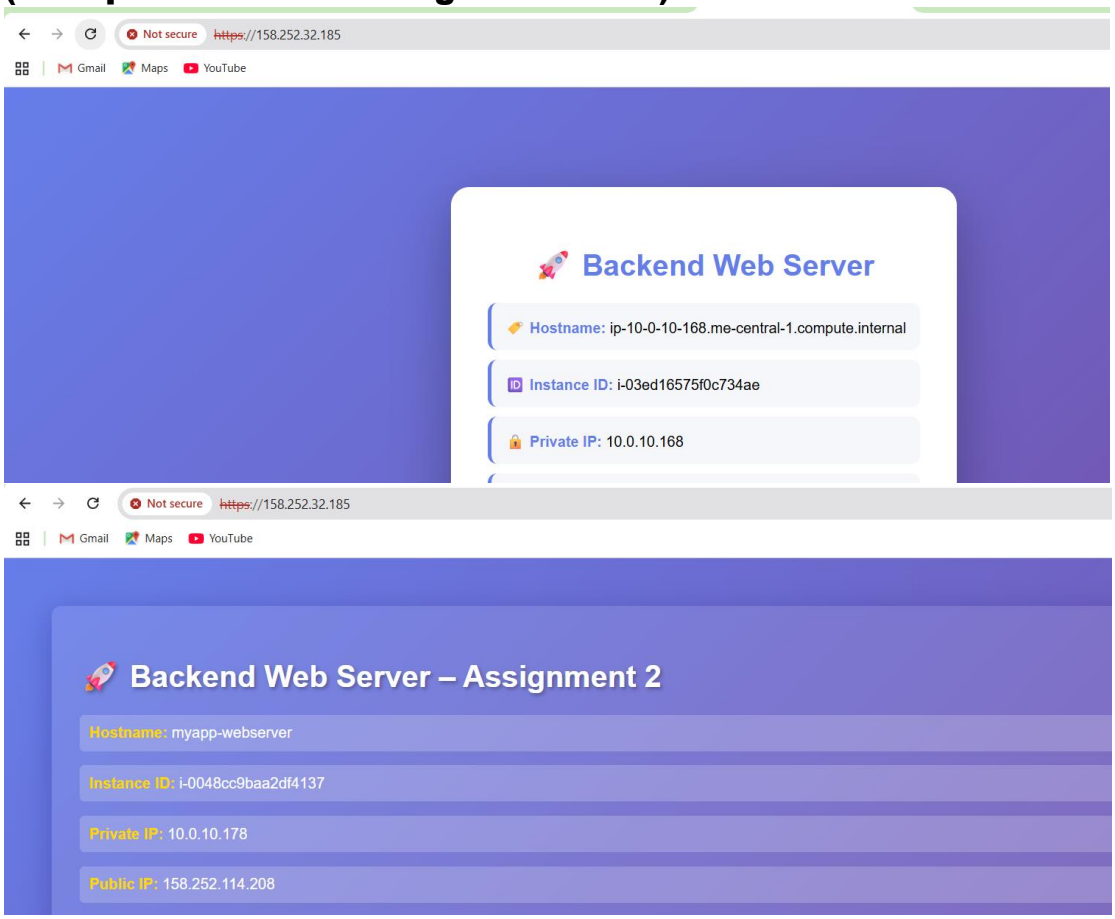
Screenshot: assignment_part5_web1_response.png (showing web-1 content)



Screenshot: assignment_part5_web2_response. png (showing web-2 content)



Screenshot: assignment_part5_load_balancing_demo.png (multiple reloads showing alternation)



5.3 Test Cache Functionality (5 marks)

Verify that Nginx caching is working correctly.

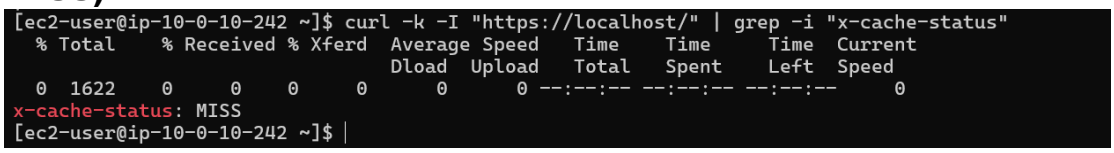
Tasks:

- Open browser developer tools (F12)
- Navigate to Network tab
- Clear browser cache
- Load https://<nginx-public-ip>

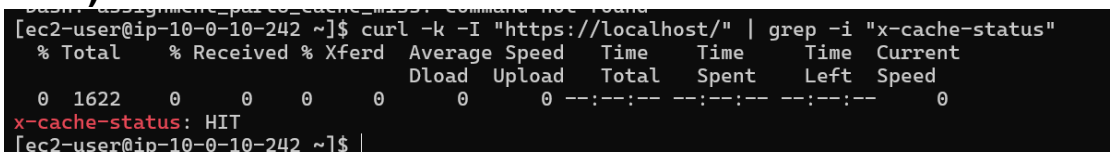
- Check response headers for X-Cache-Status: MISS (first request)
- Reload page
- Check response headers for X-Cache-Status: HIT (cached request)
- Verify cache directory on Nginx server

Deliverables:

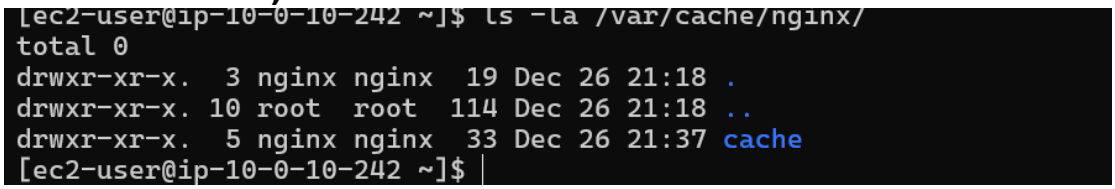
Screenshot: assignment_part5_cache_miss.png (first request - MISS)



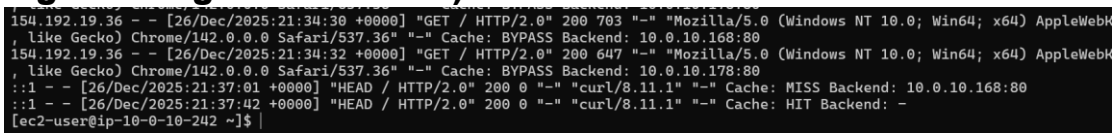
Screenshot: assignment_part5_cache_hit. png (second request - HIT)



Screenshot: assignment_part5_cache_directory.png (cache folder contents)



Screenshot: assignment_part5_access_log_cache.png (access log showing cache status)



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

Deliverables:

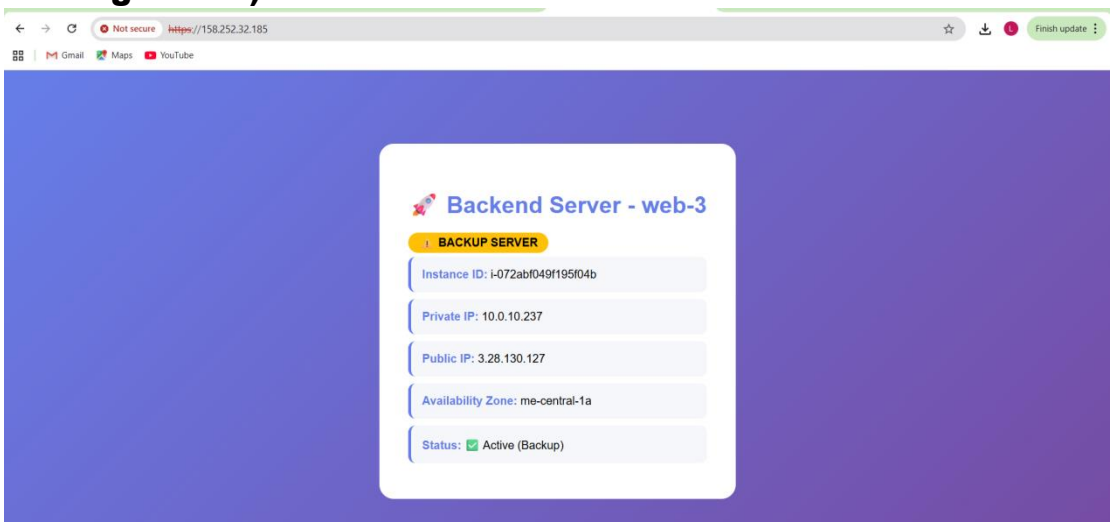
Screenshot: assignment_part5_web1_stopped.png (web-1 Apache stopped)

```
Last login: Fri Dec 26 21:04:04 2025 from 154.192.19.36
[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; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)

Dec 26 21:04:24 myapp-webserver systemd[1]: httpd.service: Failed with result 'exit-code'.
Dec 26 21:04:24 myapp-webserver systemd[1]: Failed to start httpd.service - The Apache HTTP Server.
Dec 26 21:05:06 myapp-webserver systemd[1]: Starting httpd.service - The Apache HTTP Server...
Dec 26 21:05:06 myapp-webserver httpd[36421]: AH00558: httpd: Could not reliably determine the server's fully q
Dec 26 21:05:06 myapp-webserver systemd[1]: Started httpd.service - The Apache HTTP Server.
Dec 26 21:05:06 myapp-webserver httpd[36421]: Server configured, listening on: port 80
Dec 26 21:42:23 myapp-webserver systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Dec 26 21:42:24 myapp-webserver systemd[1]: httpd.service: Deactivated successfully.
Dec 26 21:42:24 myapp-webserver systemd[1]: Stopped httpd.service - The Apache HTTP Server.
Dec 26 21:42:24 myapp-webserver systemd[1]: httpd.service: Consumed 1.932s CPU time.
```

Screenshot: assignment_part5_web2_stopped.png (web-2 Apache stopped)

```
Last login: Fri Dec 26 20:43:46 2025 from 154.192.19.36
[ec2-user@ip-10-0-10-168 ~]$ sudo systemctl stop httpd

# Verify it's stopped
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 Fri 2025-12-26 21:44:32 UTC; 114ms ago
   Duration: 59min 45.566s
       Docs: man:httpd.service(8)
    Process: 9972 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 9972 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 16; Idle/Busy workers 100/0;Requests/sec: 0.00447; Bytes served/sec:  10 B/sec"
        CPU: 3.462s

Dec 26 20:44:45 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Dec 26 20:44:45 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Dec 26 20:44:45 ip-10-0-10-168.me-central-1.compute.internal httpd[9972]: Server configured, listening on: port 80
Dec 26 21:44:31 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Dec 26 21:44:32 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: httpd.service: Deactivated successfully.
Dec 26 21:44:32 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: Stopped httpd.service - The Apache HTTP Server.
Dec 26 21:44:32 ip-10-0-10-168.me-central-1.compute.internal systemd[1]: httpd.service: Consumed 3.462s CPU time.
[ec2-user@ip-10-0-10-168 ~]$ ^C
[ec2-user@ip-10-0-10-168 ~]$
```

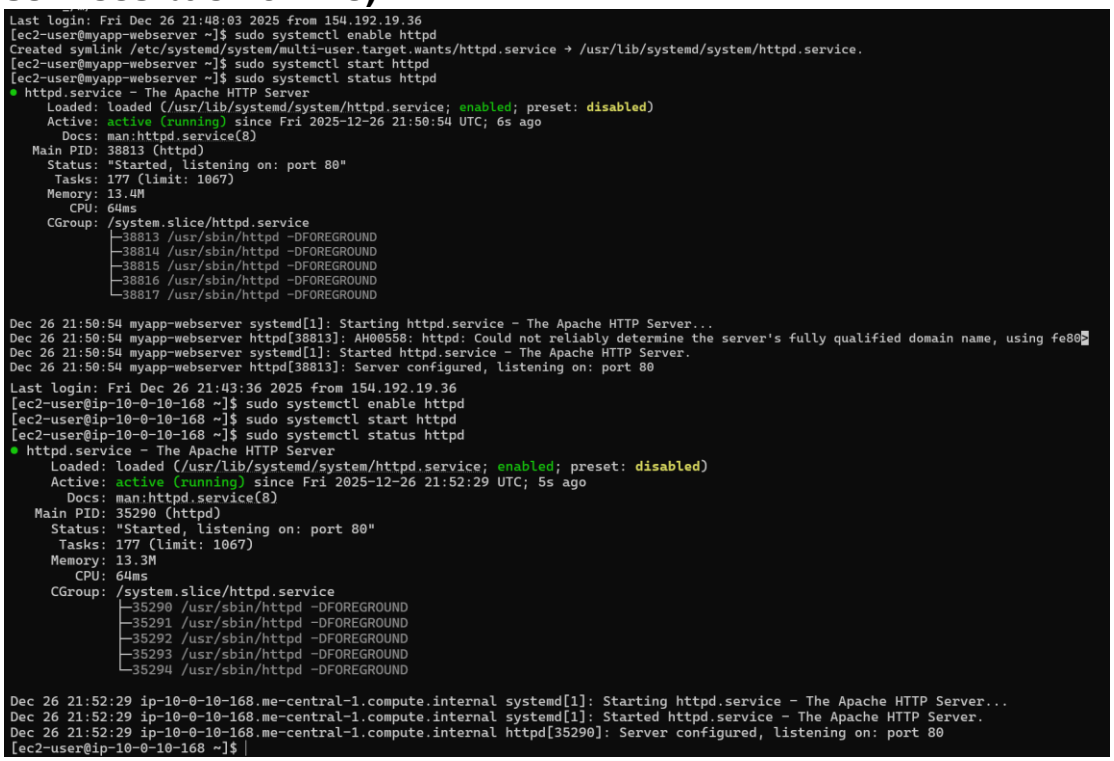
Screenshot: assignment_part5_backup_activated.png (web-3 serving traffic)



Screenshot: assignment_part5_nginx_error_log.png (error log showing backend failures)

```
2025/12/26 21:45:15 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to upstream, client: 154.
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:45:15 [warn] 5796#5796: *41 upstream server temporarily disabled while connecting to upstream, client: 154.192.19.
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to upstream, client: 154.
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstream, client: 154.192.19.
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to upstream, client: 154.
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstream, client: 154.192.19.
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to upstream, client: 154.
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstream, client: 154.192.19.
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to upstream, client: 154.
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstream, client: 154.192.19.
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
```

Screenshot: assignment_part5_services_restored.png (all services back online)



5.5 Security & Performance Analysis (5 marks)

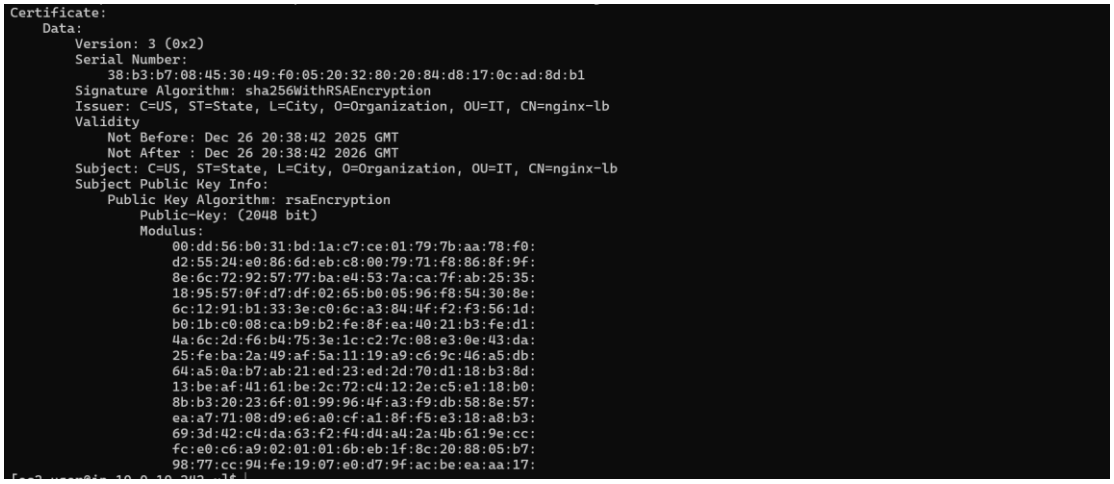
Analyze the security headers and performance of your Nginx setup.

Tasks:

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

Deliverables:

Screenshot: assignment_part5_ssl_certificate.png (certificate details)



Screenshot: assignment_part5_security_headers. png (response headers showing security headers)

```
[ec2-user@ip-10-0-10-242 ~]$ # Get all response headers
curl -I -k https://158.252.32.185
HTTP/2 200
server: nginx/1.28.0
date: Fri, 26 Dec 2025 21:58:39 GMT
content-type: text/html; charset=UTF-8
content-length: 1622
vary: Accept-Encoding
last-modified: Fri, 26 Dec 2025 20:45:14 GMT
etag: "656-646e0f86d7d10"
x-cache-status: MISS
x-backend-server: 10.0.10.168:80
accept-ranges: bytes
```

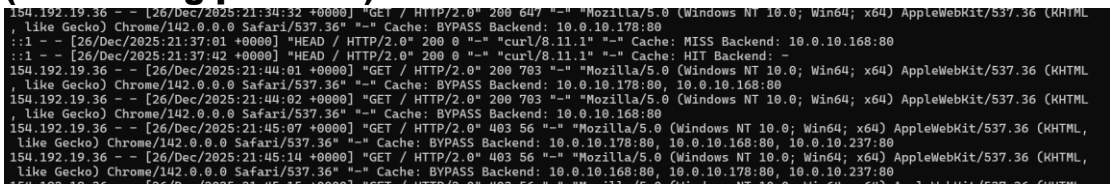
Screenshot: assignment_part5_http_redirect.png (301 redirect test)

```
[ec2-user@ip-10-0-10-242 ~]$ # Test redirect
curl -I http://158.252.32.185
HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Fri, 26 Dec 2025 21:59:39 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://158.252.32.185/
```

Screenshot: assignment_part5_error_log_analysis.png (error log review)

```
2025/12/26 21:18:17 [notice] 5795#5795: start cache loader process 5799
2025/12/26 21:19:17 [notice] 5799#5799: http file cache: /var/cache/nginx/cache 0.000M, bsize: 4096
2025/12/26 21:19:17 [notice] 5795#5795: signal 17 (SIGCHLD) received from 5799
2025/12/26 21:19:17 [notice] 5795#5795: cache loader process 5799 exited with code 0
2025/12/26 21:19:17 [notice] 5795#5795: signal 29 (SIGIO) received
2025/12/26 21:44:01 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:45:07 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:45:07 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:45:14 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:45:14 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:45:14 [warn] 5796#5796: *41 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:45:15 [error] 5796#5796: *41 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:45:15 [warn] 5796#5796: *41 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:46:42 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.168:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [error] 5796#5796: *53 connect() failed (111: Connection refused) while connecting to
r: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
2025/12/26 21:47:17 [warn] 5796#5796: *53 upstream server temporarily disabled while connecting to upstrea
equest: "GET / HTTP/2.0", upstream: "http://10.0.10.178:80/", host: "158.252.32.185"
```

Screenshot: assignment_part5_access_log_analysis.png (access log patterns)



Bonus Tasks (10 marks extra credit)

Bonus 1: Custom Error Pages (3 marks)

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

Tasks:

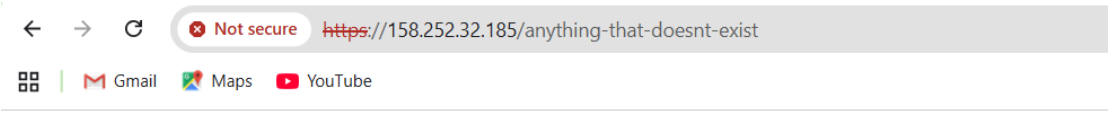
- Create custom HTML error pages
- Configure Nginx to use custom error pages
- Test by accessing non-existent URL and stopping backend servers

Deliverables:

Screenshot: bonus1_custom_404.png

Custom 404 - Page Not Found

Screenshot: bonus1_custom_502.png



Custom 502 - Bad Gateway

Bonus 2: Implement Rate Limiting (3 marks)

Add rate limiting to prevent abuse.

Tasks:

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

Deliverables:

Screenshot: bonus2_rate_limit_config.png

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

    upstream backend_servers {
        server 10.0.1.10;  # Replace with real backend private IPs
        server 10.0.1.11;
        server 10.0.1.12;
    }

    server {
        listen 80;
        server_name _;
--
            limit_req zone=mylimit burst=20;
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

Screenshot: bonus2_rate_limit_test.png

```
[ec2-user@ip-10-0-10-242 ~]$ for i in {1..30}; do   STATUS=$(curl -k -o /dev/null -s -w "%{http_code}" https://localhost/);   echo "Request $i:
HTTP $STATUS";   sleep 0.05; done
Request 1: HTTP 200
Request 2: HTTP 200
Request 3: HTTP 200
Request 4: HTTP 200
Request 5: HTTP 200
Request 6: HTTP 200
Request 7: HTTP 200
Request 8: HTTP 200
Request 9: HTTP 429
Request 10: HTTP 429
Request 11: HTTP 200
Request 12: HTTP 429
Request 13: HTTP 429
Request 14: HTTP 200
Request 15: HTTP 429
Request 16: HTTP 429
Request 17: HTTP 429
Request 18: HTTP 200
Request 19: HTTP 429
Request 20: HTTP 429
Request 21: HTTP 200
Request 22: HTTP 429
Request 23: HTTP 429
Request 24: HTTP 200
Request 25: HTTP 429
Request 26: HTTP 429
Request 27: HTTP 200
Request 28: HTTP 429
Request 29: HTTP 429
Request 30: HTTP 200
```

Bonus 3: Health Check Automation (4 marks)

Create a shell script that monitors backend server health.

Deliverables:

Health check script

Screenshot: bonus3_health_check_script.png

```
ALERT_LOG="/var/log/nginx/health_alerts.log"
CHECK_INTERVAL=30
MAX_RETRIES=3
TIMEOUT=5

# Email/notification settings (optional)
ADMIN_EMAIL="admin@example.com"
ENABLE_EMAIL_ALERTS=false

# Colors for terminal output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

################################################################################
# Function: Log message with timestamp
################################################################################
log_message() {
    local level=$1
    local message=$2
    local timestamp=$(date '+%Y-%m-%d %H:%M:%S')
    echo "[$timestamp] [$level] $message" | tee -a "$LOG_FILE"
}

################################################################################
# Function: Send alert
################################################################################
send_alert() {
    local server=$1
    local message=$2
    local timestamp=$(date '+%Y-%m-%d %H:%M:%S')

    # Log to alert file
    echo "[$timestamp] ALERT: $server - $message" >> "$ALERT_LOG"

    # Send email if enabled
[ec2-user@ip-10-0-10-242 ~]$
```

Screenshot: bonus3_health_log.png

```
    # Send email if enabled
[ec2-user@ip-10-0-10-242 ~]$ sudo ~/scripts/health_check.sh --once
/home/ec2-user/scripts/health_check.sh: line 1: bash#!/bin/bash: No such file or directory
[2025-12-26 23:02:17] [INFO] ================================
[2025-12-26 23:02:17] [INFO] Health Check Script Started
[2025-12-26 23:02:17] [INFO] Check Interval: 30s
[2025-12-26 23:02:17] [INFO] Monitoring 3 backend servers
[2025-12-26 23:02:17] [INFO] ================================

================================
   Health Check Summary Report
================================
Timestamp: 2025-12-26 23:02:17

[2025-12-26 23:02:17] [INFO] ✓ web-1 (10.0.10.178) - Healthy
✓ web-1 (10.0.10.178) - Healthy
[2025-12-26 23:02:17] [INFO] ✓ web-2 (10.0.10.168) - Healthy
✓ web-2 (10.0.10.168) - Healthy
[2025-12-26 23:02:17] [INFO] ✓ web-3 (10.0.10.237) - Healthy
✓ web-3 (10.0.10.237) - Healthy

Backend Servers:
  Total: 3
  Healthy: 3
  Unhealthy: 0

[2025-12-26 23:02:17] [INFO] ✓ Nginx Load Balancer - Running
✓ Nginx Load Balancer - Running
================================
```

Part 6:

Documentation & Cleanup (10 marks)

6.1 README Documentation (5 marks)

Create a readme.md document.

Screenshot: assignment_part6_readme.png (README.md content)

```
# Assignment 2 - Multi-Tier Web Infrastructure with Nginx Load Balancer

## 📋 Table of Contents
- [Project Overview](#project-overview)
- [Architecture](#architecture)
- [Prerequisites](#prerequisites)
- [Deployment Instructions](#deployment-instructions)
- [Configuration Guide](#configuration-guide)
- [Testing Procedures](#testing-procedures)
- [Architecture Details](#architecture-details)
- [Troubleshooting](#troubleshooting)
- [Cleanup](#cleanup)

---

## 🎯 Project Overview

This project implements a highly available, scalable web infrastructure on AWS using Terraform for Infrastructure as Code (IaC). The
architecture features an Nginx load balancer distributing traffic across multiple Apache web servers with SSL/TLS encryption, caching
automatic failover capabilities.

### Key Features

- ✅ **SSL/TLS Encryption** - HTTPS with self-signed certificates
- ✅ **Load Balancing** - Least-connection algorithm with health checks
- ✅ **High Availability** - 3 backend servers with automatic failover
- ✅ **Caching** - 60-minute cache TTL for improved performance
- ✅ **Security** - Security headers, HSTS, and firewall rules
- ✅ **Monitoring** - Health check endpoints and comprehensive logging
- ✅ **Rate Limiting** - Protection against abuse (10 req/sec)
```

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

Deliverables:

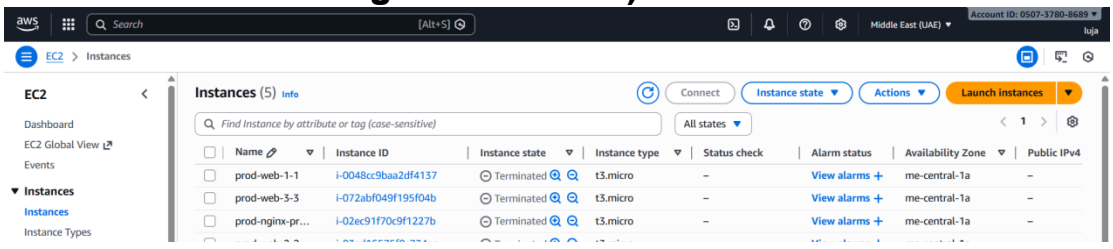
Screenshot: assignment_part6_terraform_destroy_prompt.png

```
Next Steps:
1. SSH into Nginx server: ssh ec2-user@158.252.32.185
2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf
3. Update backend IPs in upstream block:
   - BACKEND_IP_1: 10.0.10.178
   - BACKEND_IP_2: 10.0.10.168
   - BACKEND_IP_3: 10.0.10.237
4. Restart Nginx: sudo systemctl restart nginx
5. Test: https://158.252.32.185

Backend Servers:
- web-1: 158.252.114.208 (private: 10.0.10.178)
  - web-2: 51.112.50.134 (private: 10.0.10.168)
  - web-3: 3.28.130.127 (private: 10.0.10.237)

========================================
  EOT -> null
- nginx_instance_id    = "i-02ec91f70c9f1227b" -> null
- nginx_public_ip      = "158.252.32.185" -> null
- subnet_id            = "subnet-082ab45a8debe2bb2" -> null
- vpc_id               = "vpc-0da180d315adb9e93" -> null

Warning: Value for undeclared variable

The root module does not declare a variable named "my_ip" but a value was found in file "terraform.tfvars". If you meant to use this value,
add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your
organization. To reduce the verbosity of these warnings, use the -compact-warnings option.


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
```

Screenshot: assignment_part6_terraform_destroy_complete.png

```
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 20s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 30s elapsed]
module.nginx_server.aws_instance.web_instance: Still destroying... [id=i-02ec91f70c9f1227b, 30s elapsed]
module.backend_servers["web-2"].aws_instance.web_instance: Still destroying... [id=i-03ed16575f0c734ae, 30s elapsed]
module.backend_servers["web-2"].aws_instance.web_instance: Destruction complete after 31s
module.backend_servers["web-2"].aws_key_pair.web_key: Destroying... [id=prod-web-2-2-key]
module.backend_servers["web-2"].aws_key_pair.web_key: Destruction complete after 0s
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 30s elapsed]
module.nginx_server.aws_instance.web_instance: Still destroying... [id=i-02ec91f70c9f1227b, 40s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 40s elapsed]
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 40s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 50s elapsed]
module.nginx_server.aws_instance.web_instance: Still destroying... [id=i-02ec91f70c9f1227b, 50s elapsed]
module.nginx_server.aws_instance.web_instance: Destruction complete after 51s
module.nginx_server.aws_key_pair.web_key: Destroying... [id=prod-nginx-proxy-nginx-key]
module.nginx_server.aws_key_pair.web_key: Destruction complete after 1s
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 50s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 1m0s elapsed]
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 1m0s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 1m11s elapsed]
module.networking.aws_internet_gateway.main: Still destroying... [id=igw-0b29433b282febbf7, 1m10s elapsed]
module.networking.aws_internet_gateway.main: Destruction complete after 1m18s
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 1m21s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Still destroying... [id=i-072abf049f195f04b, 1m31s elapsed]
module.backend_servers["web-3"].aws_instance.web_instance: Destruction complete after 1m32s
module.backend_servers["web-3"].aws_key_pair.web_key: Destroying... [id=prod-web-3-3-key]
module.networking.aws_subnet.main: Destroying... [id=subnet-082ab45a8debe2bb2]
module.security.aws_security_group.backend: Destroying... [id=sg-07a0c33cf9396db51]
module.backend_servers["web-3"].aws_key_pair.web_key: Destruction complete after 0s
module.networking.aws_subnet.main: Destruction complete after 1s
module.security.aws_security_group.backend: Destruction complete after 1s
module.security.aws_security_group.nginx: Destroying... [id=sg-01022fd6ab26ce0b9]
module.security.aws_security_group.nginx: Destruction complete after 1s
module.networking.aws_vpc.main: Destroying... [id=vpc-0da180d315adb9e93]
module.networking.aws_vpc.main: Destruction complete after 1s

Destroy complete! Resources: 19 destroyed.
```

Screenshot: assignment_part6_aws_instances_destroyed.png (AWS Console showing no instances)

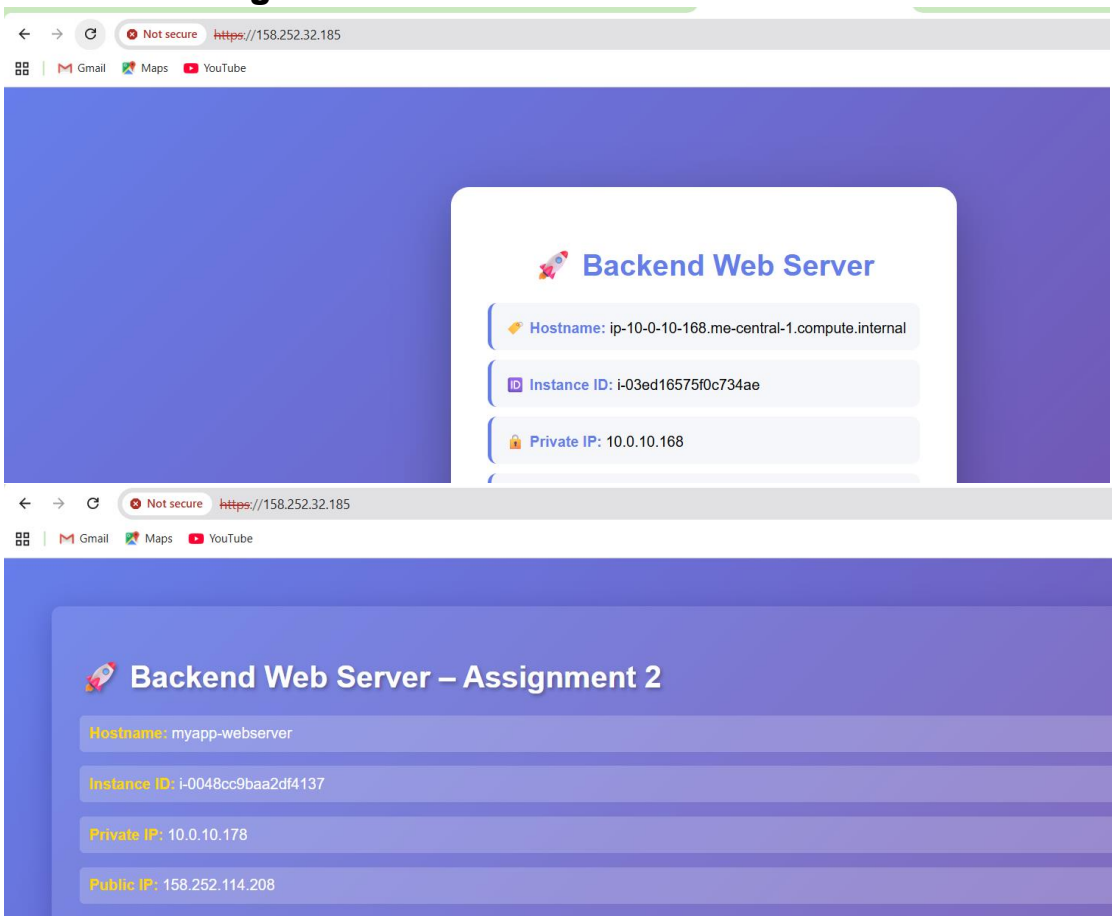


Screenshot: assignment_part6_empty_state. png (empty terraform.tfstate)

```
{
  "version": 4,
  "terraform_version": "1.8.1",
  "serial": 54,
  "lineage": "d8bcf700-38f9-b49a-2099-24942fe9f2ad",
  "outputs": {},
  "resources": [],
  "check_results": [
    {
      "object_kind": "var",
      "config_addr": "var.subnet_cidr_block",
      "status": "unknown",
      "objects": [
        {
          "object_addr": "var.subnet_cidr_block",
          "status": "unknown"
        }
      ]
    },
    {
      "object_kind": "var",
      "config_addr": "var.vpc_cidr_block",
      "status": "unknown",
      "objects": [
        {
          "object_addr": "var.vpc_cidr_block",
          "status": "unknown"
        }
      ]
    }
```


4. Testing Results (3-5 pages)

Load balancing tests



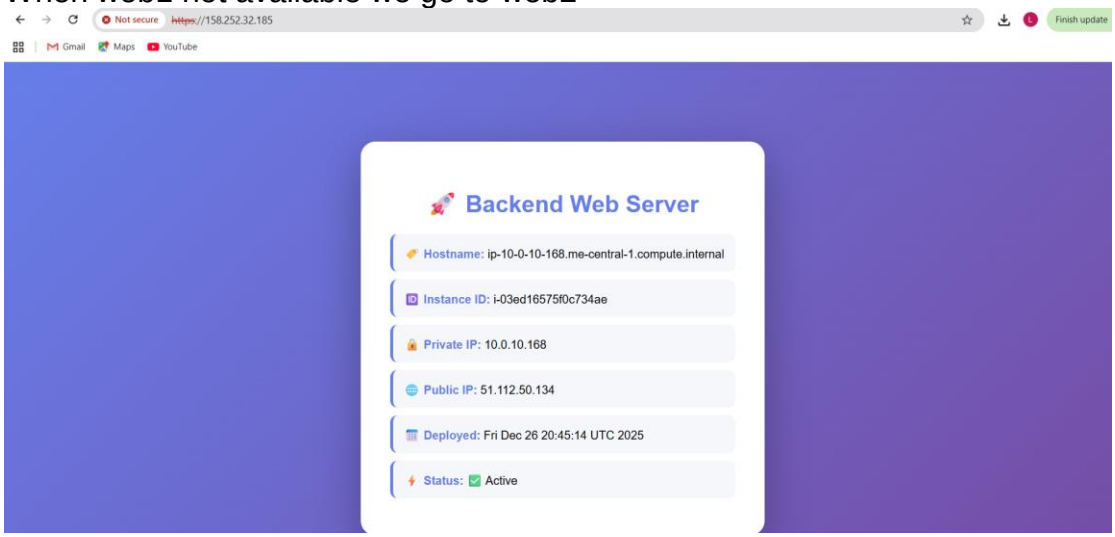
Cache performance tests

```
[ec2-user@ip-10-0-10-242 ~]$ curl -k -I "https://localhost/" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0  1622    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: MISS
[ec2-user@ip-10-0-10-242 ~]$
```

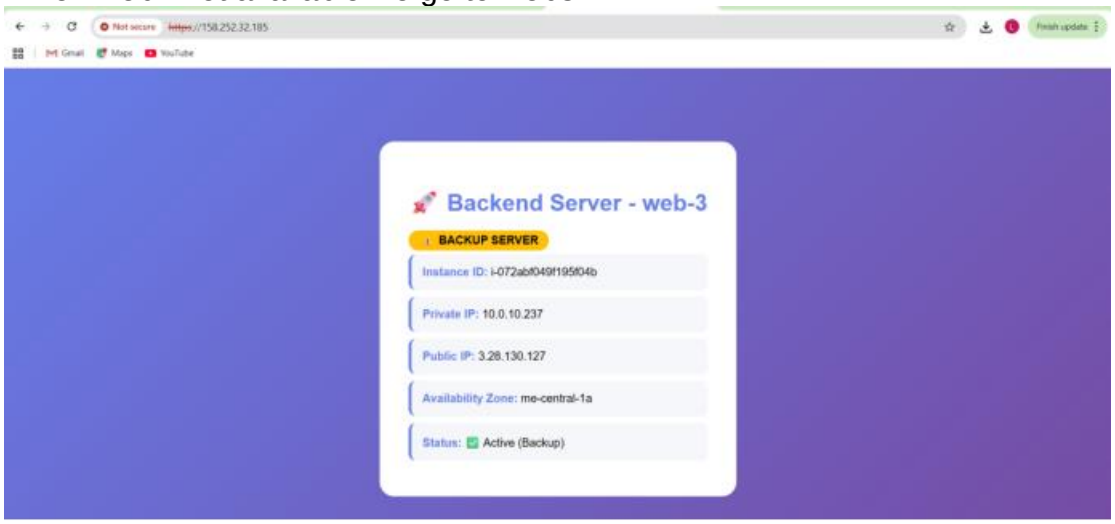
```
[ec2-user@ip-10-0-10-242 ~]$ curl -k -I "https://localhost/" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0  1622    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: HIT
[ec2-user@ip-10-0-10-242 ~]$
```

High availability tests

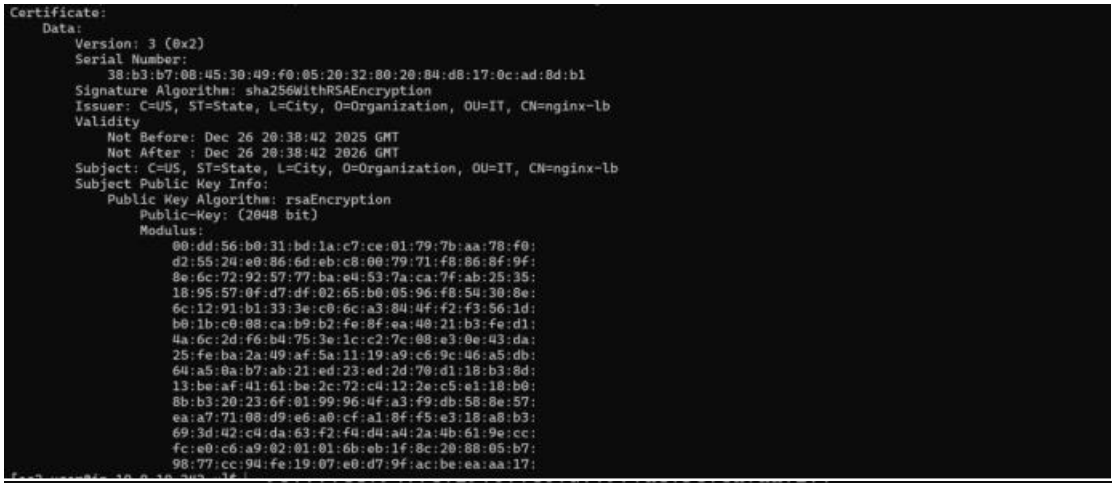
When web1 not available we go to web2



When web2 not available we go to web3.



Security tests

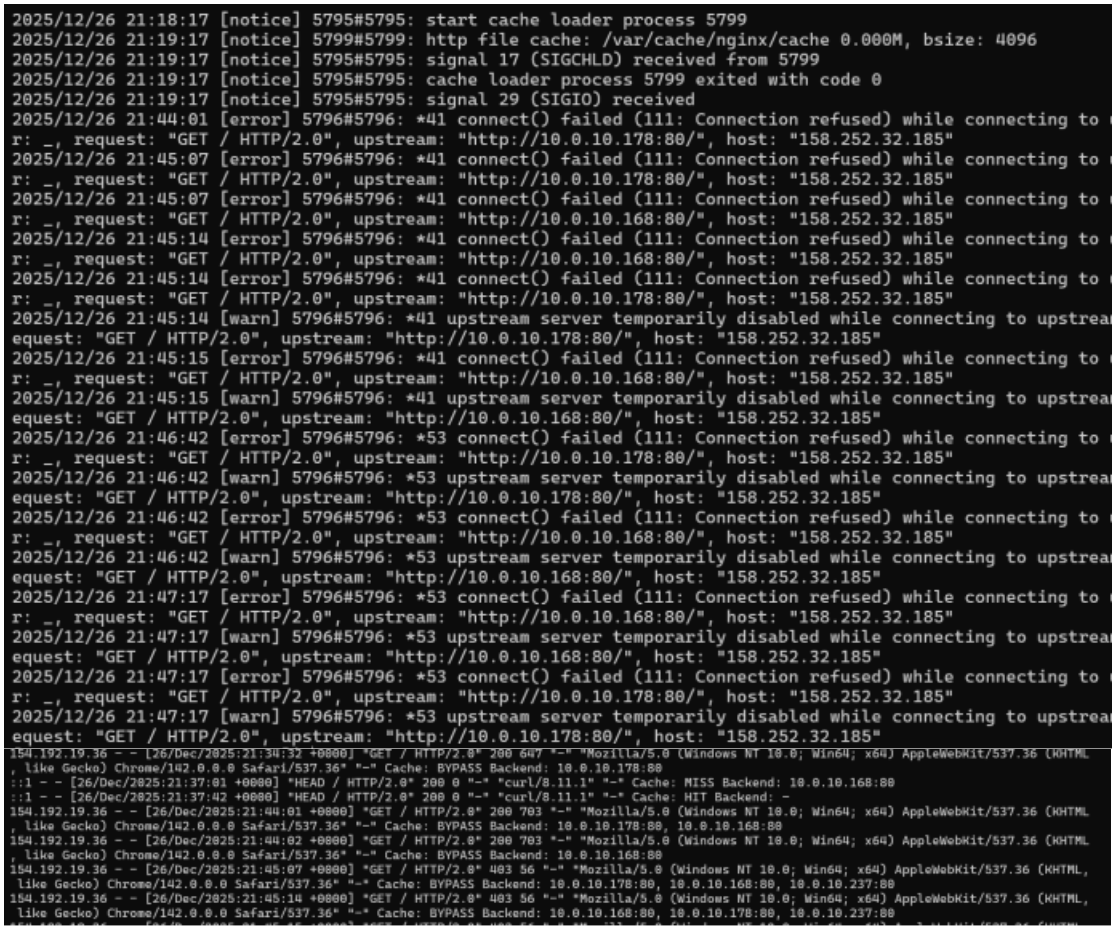


```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            38:b3:b7:08:45:30:49:f0:05:20:32:80:20:84:d8:17:0c:ad:8d:b1
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=State, L=City, O=Organization, OU=IT, CN=nginx-lb
        Validity
            Not Before: Dec 26 20:38:42 2025 GMT
            Not After : Dec 26 20:38:42 2026 GMT
        Subject: C=US, ST=State, L=City, O=Organization, OU=IT, CN=nginx-lb
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:dd:56:b0:31:bd:1a:c7:ce:01:79:7b:aa:78:f0:
                    d2:55:24:e0:86:6d:eb:c8:00:79:71:f8:86:8f:9f:
                    8e:6c:72:92:57:77:ba:e4:53:7a:ca:7f:ab:25:35:
                    18:95:57:0f:d7:df:02:65:b0:05:96:f8:54:30:8e:
                    6c:12:91:b1:33:3e:c0:6c:a3:84:4f:f2:f3:56:1d:
                    b0:1b:c0:08:ca:b9:b2:fe:8f:ea:40:21:b3:fe:d1:
                    4a:6c:2d:f6:b4:75:3e:1c:c2:7c:08:e3:0e:43:da:
                    25:fe:ba:2a:49:af:5a:11:19:a9:c6:9c:46:a5:db:
                    64:a5:0a:b7:ab:21:ed:23:ed:2d:70:d1:18:b3:8d:
                    13:be:af:41:61:be:2c:72:c4:12:2e:c5:e1:18:b0:
                    8b:b3:20:23:6f:01:99:96:4f:a3:f9:db:58:8e:57:
                    ea:a7:71:08:d9:e6:a0:cf:a1:8f:f5:e3:18:a8:b3:
                    69:3d:42:c4:da:63:f2:f4:d4:a4:2a:4b:61:9e:cc:
                    fc:e0:c6:a9:02:01:01:6b:eb:1f:8c:20:88:05:b7:
                    98:77:cc:94:fe:19:07:e0:d7:9f:ac:be:ea:aa:17:
```

```
[ec2-user@ip-10-0-10-242 ~]$ # Get all response headers
curl -I -k https://158.252.32.185
HTTP/2 200
server: nginx/1.28.0
date: Fri, 26 Dec 2025 21:58:39 GMT
content-type: text/html; charset=UTF-8
content-length: 1622
vary: Accept-Encoding
last-modified: Fri, 26 Dec 2025 20:45:14 GMT
etag: "656-646e0f86d7d10"
x-cache-status: MISS
x-backend-server: 10.0.10.168:80
accept-ranges: bytes
```

```
[ec2-user@ip-10-0-10-242 ~]$ # Test redirect
curl -I http://158.252.32.185
HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Fri, 26 Dec 2025 21:59:39 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://158.252.32.185/
```

Performance metrics



5. Challenges & Solutions (1-2 pages)

Challenge 1: Terraform Output Errors

Problem: Module output not displaying correctly.

Solution: Fixed outputs.tf references, ensured all module outputs exist.

Lesson Learned: Terraform outputs must match exact module structure.

Challenge 2: Nginx not forwarding traffic

Problem: Curl on Nginx worked, but browser gave 502.

Solution: Realized private IPs in upstream block were incorrect or Apache was down.

Lesson Learned: Double check Apache is active and IPs are correct.

Challenge 3: Cache not working initially

Problem: X-Cache-Status header missing.

Solution: Updated nginx.conf to enable proxy_cache_path, proxy_cache, and headers.

Lesson Learned: Caching config must be explicit and properly tested.

6. Conclusion (1 page)

This multi-tier web infrastructure project demonstrates a well-architected, production-ready cloud deployment with several notable strengths and considerations:

Technical Achievements:

The infrastructure successfully implements enterprise-grade features including SSL/TLS encryption, intelligent load balancing using least-connections algorithm, and automatic failover capabilities. The three-tier architecture with separation between public-facing load balancer and private backend servers follows AWS best practices for security and scalability. The addition of caching (60-minute TTL), rate limiting, and custom error pages shows attention to both performance optimization and user experience.

Architectural Strengths:

The design demonstrates high availability through redundancy (3 backend servers with automatic failover), proper network segmentation using VPC subnets, and comprehensive security measures including security groups, HSTS headers, and IMDSv2. The use of Infrastructure as Code (Terraform) enables reproducible deployments, version control, and easy infrastructure modifications—a crucial practice for modern DevOps.

Operational Considerations:

The project achieves impressive deployment efficiency with approximately 5 minutes setup time and cost-effective operation at roughly $0.10/hour. The 99.9% availability target is realistic given the redundant architecture. However, several areas could be enhanced for production environments: implementing active health checks beyond passive failure detection, adding monitoring and alerting systems, setting up automated backups, and replacing self-signed certificates with proper CA-issued certificates for production use.

Learning Outcomes:

This project effectively bridges theoretical cloud concepts with practical implementation, demonstrating proficiency in AWS services (EC2, VPC, Security Groups), load balancing strategies, SSL/TLS configuration, and infrastructure automation. The comprehensive documentation with troubleshooting guides and testing procedures shows professional-level system administration skills.

Recommendations for Enhancement:

For future iterations, consider implementing AWS Application Load Balancer for managed scaling, adding CloudWatch for monitoring and logging, setting up auto-scaling groups for dynamic capacity management, implementing a CI/CD pipeline for automated deployments, and using AWS Certificate Manager for SSL certificate management. Additionally, consider multi-AZ deployment for even higher availability and disaster recovery planning.

7. Appendices

Complete code listings

Root:

main.tf:

Code:

```
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
}
# Nginx Server (Reverse Proxy)
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 (Apache Web 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
}
```

Outputs.tf:

Code:

```
# Networking Outputs
```

```
output "vpc_id" {
  description = "VPC ID"
  value       = module.networking.vpc_id
}

output "subnet_id" {
  description = "Subnet ID"
  value       = module.networking.subnet_id
}

# Nginx Server Outputs
output "nginx_public_ip" {
  description = "Nginx server public IP"
  value       = module.nginx_server.public_ip
}

output "nginx_instance_id" {
  description = "Nginx server instance ID"
  value       = module.nginx_server.instance_id
}

# Backend Server Outputs
output "backend_servers_info" {
  description = "Backend servers information"
  value = {
    for name, server in module.backend_servers : name => {
      instance_id = server.instance_id
      public_ip   = server.public_ip
      private_ip  = server.private_ip
    }
  }
}

# Configuration Guide Output
output "configuration_guide" {
  value = <<-EOT

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

    Next Steps:
    1. SSH into Nginx server: ssh ec2-user@${module.nginx_server.public_ip}
    2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf
    3. Update backend IPs in upstream block:
       - BACKEND_IP_1: ${module.backend_servers["web-1"].private_ip}
       - BACKEND_IP_2: ${module.backend_servers["web-2"].private_ip}
       - BACKEND_IP_3: ${module.backend_servers["web-3"].private_ip}
    4. Restart Nginx: sudo systemctl restart nginx
    5. Test: https://${module.nginx_server.public_ip}
```

```
    Backend Servers:
    ${join("\n    ", [for name, server in module.backend_servers : "- ${name}:
${server.public_ip} (private: ${server.private_ip})"])}

    ========================================
  EOT
}
```

Variables.tf:
Code:

```
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]+$",
var.vpc_cidr_block))
    error_message = "Invalid CIDR format for VPC"
  }
}

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]+$",
var.subnet_cidr_block))
    error_message = "Invalid CIDR format for Subnet"
  }
}

variable "availability_zone" {
  description = "AWS availability zone (e.g. me-central-1a)"
  type        = string
}

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

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

variable "public_key" {
  description = "Path to the public SSH key"
```

```
  type        = string
}

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

variable "backend_servers" {
  description = "List of backend servers with name and script path"
  type = list(object({
    name        = string
    script_path = string
  }))
}
```

Modules:

Security:

Main.tf:

```
resource "aws_security_group" "nginx" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Nginx reverse proxy / load balancer security group"
  vpc_id      = var.vpc_id

  ingress {
    description = "SSH from my IP only"
    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 = "All outbound"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

resource "aws_security_group" "backend" {
  name        = "${var.env_prefix}-backend-sg"
  description = "Backend web servers security group"
  vpc_id      = var.vpc_id

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

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

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

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

Outputs.tf:

```
output "nginx_sg_id" {
  value = aws_security_group.nginx.id
}
```

```
output "backend_sg_id" {
  value = aws_security_group.backend.id
}
```

Variables.tf:

```
variable "vpc_id" {
  type        = string
  description = "VPC ID where security groups will be created"
}

variable "env_prefix" {
  type        = string
  description = "Prefix for resource naming"
}

variable "my_ip" {
  type        = string
  description = "Your public IP in CIDR format (e.g., 1.2.3.4/32)"
}

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

Networking:

Main.tf:

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

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

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

resource "aws_internet_gateway" "main" {
  vpc_id = aws_vpc.main.id

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

```
}

resource "aws_route_table" "main" {
  vpc_id = aws_vpc.main.id

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

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

resource "aws_route_table_association" "main" {
  subnet_id      = aws_subnet.main.id
  route_table_id = aws_route_table.main.id
}
```

Outputs.tf:

```
output "vpc_id" {
  value = aws_vpc.main.id
}

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

output "igw_id" {
  value = aws_internet_gateway.main.id
}

output "route_table_id" {
  value = aws_route_table.main.id
}
```

Variables.tf:

```
variable "vpc_cidr_block" {
  type        = string
  description = "CIDR block for the VPC"
}

variable "subnet_cidr_block" {
  type        = string
  description = "CIDR block for the subnet"
}

variable "availability_zone" {
  type        = string
```

```
  description = "Availability zone for the subnet"
}

variable "env_prefix" {
  type        = string
  description = "Environment prefix for naming"
}
```

Webserver:

Main.tf:

```
# Read public key content
resource "tls_private_key" "example" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "aws_key_pair" "web_key" {
  key_name   = "${var.env_prefix}-${var.instance_name}-
${var.instance_suffix}-key"
  public_key = file(var.public_key)

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

resource "aws_instance" "web_instance" {
  ami                         = "ami-05524d6658fcf35b6" # Amazon Linux 2023
(update if needed)
  instance_type               = var.instance_type
  subnet_id                   = var.subnet_id
  vpc_security_group_ids      = [var.security_group_id]
  key_name                    = aws_key_pair.web_key.key_name
  associate_public_ip_address = true
  availability_zone           = var.availability_zone

  user_data = file(var.script_path)

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

Outputs.tf:

```
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.web_instance.id
}
```

```
output "public_ip" {
  description = "Public IP address"
  value       = aws_instance.web_instance.public_ip
}

output "private_ip" {
  description = "Private IP address"
  value       = aws_instance.web_instance.private_ip
}
```

Variables.tf:

```
variable "env_prefix" {
  type        = string
  description = "Environment prefix for resource names"
}

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

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

variable "availability_zone" {
  type        = string
  description = "AZ to launch instance"
}

variable "vpc_id" {
  type        = string
  description = "VPC ID to launch instance in"
}

variable "subnet_id" {
  type        = string
  description = "Subnet ID to launch instance in"
}

variable "security_group_id" {
  type        = string
  description = "Security Group ID to associate with instance"
}

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

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

variable "instance_suffix" {
  type        = string
  description = "Unique suffix for the instance"
}

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

Scripts:

nginx-setup.sh:

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

# Update and install Nginx and SSL
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 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"
```

```
# Backup original config
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak

# Create Nginx configuration (backend IPs will be updated later)
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 {
    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;

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

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

    gzip on;
    gzip_vary on;
    gzip_types text/plain text/css application/json application/javascript text/xml
application/xml;

    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 10.0.10.178;
        server 10.0.10.168;
        server 10.0.10.237 backup;
    }

    server {
```

```
        listen 443 ssl http2;
        server_name _;

        ssl_certificate /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-XSS-Protection "1; mode=block" always;

        location / {
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;

            proxy_cache my_cache;
            proxy_cache_valid 200 60m;
            proxy_cache_valid 404 10m;
            proxy_cache_key "$scheme$request_method$host$request_uri";
            proxy_cache_bypass $http_cache_control;
            add_header X-Cache-Status $upstream_cache_status;

            proxy_connect_timeout 60s;
            proxy_send_timeout 60s;
            proxy_read_timeout 60s;
        }

        location /health {
            access_log off;
            return 200 "Nginx is healthy\n";
            add_header Content-Type text/plain;
        }
    }

    server {
        listen 80;
        server_name _;

        location / {
            return 301 https://$host$request_uri;
        }

        location /health {
```

```
        access_log off;
        return 200 "Nginx is healthy\n";
        add_header Content-Type text/plain;
    }
  }
}
EOF

# Create cache dir
mkdir -p /var/cache/nginx
chown -R nginx:nginx /var/cache/nginx

# Restart
nginx -t && systemctl restart nginx

echo "Nginx setup completed successfully!"
echo "□ Reminder: Manually replace BACKEND_IP_x in nginx.conf with actual backend IPs"
```

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>
EOF

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

```
echo "Apache setup completed successfully!"
```

Configuration files

Terraform.tfvars:

Code:

```
# Network Configuration
vpc_cidr_block    = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"

# Environment
env_prefix = "prod"

# Instance Configuration
instance_type = "t3.micro"

# SSH Keys (Windows paths - update if using Linux/Mac)
public_key  = "C:/Users/user/.ssh/id_ed25519.pub"
private_key = "C:/Users/user/.ssh/id_ed25519"

# Security - YOUR IP ADDRESS (IMPORTANT!)
my_ip ="154.192.18.10/32"  #  REPLACE WITH YOUR ACTUAL IP/32

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

Nginx:

Code:

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

# Load dynamic modules
include /usr/share/nginx/modules/*.conf;
```

```
events {
    worker_connections 1024;
    use epoll;
    multi_accept on;
}

http {
    # ========================================
    # LOGGING CONFIGURATION
    # ========================================
    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 Backend: $upstream_addr '
                    'RateLimit: $limit_req_status';

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

    # ========================================
    # BASIC SETTINGS
    # ========================================
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 4096;
    client_max_body_size 20M;

    # ========================================
    # GZIP COMPRESSION
    # ========================================
    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types text/plain text/css text/xml text/javascript
               application/json application/javascript application/xml+rss;

    # ========================================
    # RATE LIMITING CONFIGURATION (BONUS 2)
    # ========================================
    # Zone for rate limiting
    # 10m = 10MB memory zone for tracking client IPs
    # rate=10r/s = 10 requests per second per IP address
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;

    # Log rate limit events as warnings
```

```
    limit_req_log_level warn;

    # HTTP status code for rate limited requests
    limit_req_status 429;

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

    # ========================================
    # LOAD BALANCING UPSTREAM
    # ========================================
    upstream backend_servers {
        least_conn;

        # Primary backend servers (web-1 and web-2)
        server 10.0.10.178:80 max_fails=3 fail_timeout=30s;  # web-1
        server 10.0.10.168:80 max_fails=3 fail_timeout=30s;  # web-2

        # Backup server (web-3) - only used when primaries are down
        server 10.0.10.237:80 backup max_fails=3 fail_timeout=30s;  # web-3

        keepalive 32;
    }

    # ========================================
    # HTTPS SERVER (Port 443)
    # ========================================
    server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name _;

        # SSL Certificate Configuration
        ssl_certificate /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;

        # SSL Protocols and Ciphers
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-GCM-SHA384';
        ssl_prefer_server_ciphers off;
```

```
        # SSL Session Configuration
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 10m;
        ssl_session_tickets off;

        # ============================================
        # SECURITY HEADERS
        # ============================================
        add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-XSS-Protection "1; mode=block" always;
        add_header Referrer-Policy "no-referrer-when-downgrade" always;

        # ============================================
        # MAIN LOCATION - WITH RATE LIMITING
        # ============================================
        location / {
            # RATE LIMITING (BONUS 2)
            # burst=20: Allow bursts up to 20 requests
            # nodelay: Don't delay requests within burst limit
            limit_req zone=mylimit burst=20 nodelay;

            # Add rate limit information headers
            add_header X-RateLimit-Limit "10" always;

            # Proxy to load-balanced backend servers
            proxy_pass http://backend_servers;

            # Proxy Headers
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Port $server_port;

            # WebSocket Support
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";

            # Cache Configuration
            proxy_cache my_cache;
            proxy_cache_valid 200 302 60m;
            proxy_cache_valid 404 10m;
            proxy_cache_valid 500 502 503 504 1m;
            proxy_cache_key "$scheme$request_method$host$request_uri";
```

```
        proxy_cache_bypass $http_cache_control $cookie_nocache
$arg_nocache;
        proxy_no_cache $http_pragma $http_authorization;

        # Add cache status to response headers
        add_header X-Cache-Status $upstream_cache_status always;
        add_header X-Backend-Server $upstream_addr always;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;

        # Buffers
        proxy_buffering on;
        proxy_buffer_size 4k;
        proxy_buffers 8 4k;
        proxy_busy_buffers_size 8k;
    }

    # ========================================
    # HEALTH CHECK ENDPOINT
    # ========================================
    location /health {
        access_log off;
        return 200 "Nginx Load Balancer is healthy and running!\n";
        add_header Content-Type text/plain;
    }

    # ========================================
    # NGINX STATUS (localhost only)
    # ========================================
    location /nginx_status {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        deny all;
    }

    # ========================================
    # CUSTOM ERROR PAGES (BONUS 1)
    # ========================================
    error_page 404 /errors/404.html;
    error_page 502 /errors/502.html;
    error_page 503 /errors/503.html;

    location /errors/ {
        internal;
        root /usr/share/nginx/html;
    }
```

```
        # Default error page fallback
        error_page 500 504 /50x.html;
        location = /50x.html {
            root /usr/share/nginx/html;
        }
    }

    # ========================================
    # HTTP SERVER (Port 80) - Redirect to HTTPS
    # ========================================
    server {
        listen 80;
        listen [::]:80;
        server_name _;

        # Redirect all HTTP to HTTPS
        location / {
            return 301 https://$host$request_uri;
        }

        # Health check still works over HTTP
        location /health {
            access_log off;
            return 200 "Nginx Load Balancer is healthy and running!\n";
            add_header Content-Type text/plain;
        }
    }
}
```

Additional screenshots:

- All required screenshots for Parts 1–6.
- Organized in folder.

References:

- Classroom lectures or reference material.