Assignment 02

Name: Sadaf Riaz

Roll No: 2023-BSE-077

Subject: Cloud Computing

Class: BSE5-B

Instructor: Sir Waqas

Due Date: 30 December 2025

Repo: https: https://github.com/SadafRiaz-077/cc_sadafriaz-077_Assignment-2

1.Executive Summary

This report presents the implementation of Assignment , which focuses on deploying a robust multi-tier web infrastructure on Amazon Web Services (AWS) using Terraform. The primary objective of this assignment is to gain hands-on experience with Infrastructure as Code (IaC), cloud networking, and advanced Nginx configurations in a practical setting.

The deployed infrastructure is designed for high availability and performance. It consists of a single Nginx server acting as a reverse proxy and load balancer, along with three backend web servers running Apache. The Nginx server is configured to handle HTTPS using a self-signed SSL certificate, implement caching to improve response times, and distribute incoming traffic efficiently across the backend servers. Additionally, one of the backend servers is designated as a backup to ensure service continuity in case of failure of the primary servers.

Terraform modules were employed to organize the code efficiently and promote reusability. Separate modules were developed for networking, security, and web servers. The networking module provisions a Virtual Private Cloud (VPC), subnets, an Internet Gateway, and routing configurations. The security module establishes security groups to permit only essential traffic while adhering to best practices. The web server module is used to deploy both the Nginx and Apache servers in a structured and consistent manner.

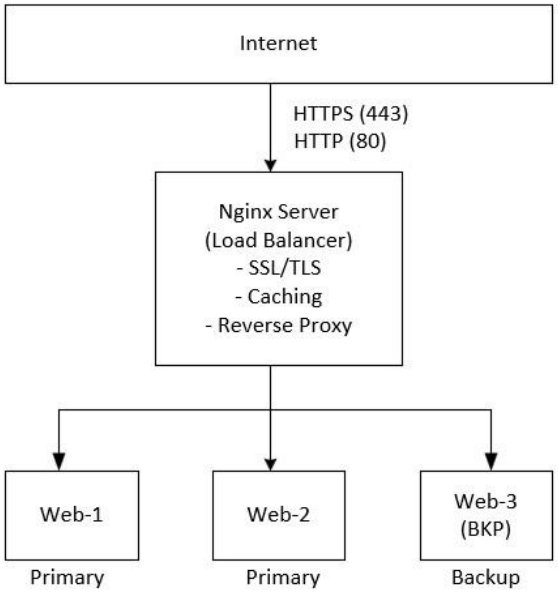
Server configuration was automated using shell scripts. The Apache backend servers display system and network information using EC2 metadata, whereas the Nginx server manages HTTPS connections, caching, request forwarding, and failover handling. The infrastructure was thoroughly tested to verify load balancing, cache performance, backup server activation, and the proper application of security measures.

Overall, this assignment provided valuable practical experience in Terraform, AWS cloud services, and Nginx configurations. It strengthened understanding of automated deployments, scalability, and designing high-availability cloud systems, equipping one with the skills necessary for modern cloud infrastructure management.

2.Architecture Overview

The architecture for this assignment implements a high-availability, multi-tier web infrastructure on AWS. It is designed to ensure scalability, performance, and fault tolerance, following best practices in cloud infrastructure design. The main components and their interactions are described below.

Architecture Diagram



Component Descriptions

1. Nginx Server

- Functions as a **reverse proxy**, **load balancer**, and **caching server**.
- Terminates **HTTPS connections** using a self-signed SSL certificate.
- Distributes incoming client requests across backend servers for **load balancing**.
- Implements **failover mechanisms** to ensure high availability.

2. Backend Apache Servers

- Host the web application and display system & network information using **EC2 metadata**.
- **Two primary servers** handle normal traffic, and **one backup server** ensures continuity if a primary server fails.
- Located in **private subnets** for security, isolated from direct internet access.

3. Virtual Private Cloud (VPC)

- Provides an isolated **network environment** on AWS.
- Includes **public and private subnets** to separate frontend (Nginx) and backend servers.
- Ensures secure internal communication between servers.

4. Subnets

- **Public Subnet:** Hosts the Nginx server, allowing internet access.
- **Private Subnets:** Host backend Apache servers to protect them from direct external access.

5. Internet Gateway

- Enables **internet connectivity** for the public subnet.
- Allows external clients to reach the Nginx server.

6. Security Groups

- Control **network traffic** to and from instances.
- Nginx security group allows **HTTPS traffic (port 443)** from the internet.
- Backend servers only accept **HTTP traffic (port 80)** from the Nginx server.
- Implements **principle of least privilege** for network security.

7. Routing

- Public subnet routes **internet-bound traffic** through the Internet Gateway.
- Private subnets communicate with Nginx server via **internal routing**, maintaining security and isolation.

3. Implementation Details

Part 1: Infrastructure Setup

1.1 Project Structure

assignment_part1_project_structure.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ tree
.
├── README.md
├── docs
├── locals.tf
├── main.tf
├── modules
│   ├── networking
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   ├── security
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   └── webserver
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── screenshots
├── scripts
│   ├── apache-setup.sh
│   └── nginx-setup.sh
└── variables.tf

8 directories, 16 files
```

assignment_part1_gitignore.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat .gitignore
.terraform/
terraform.tfstate
terraform.tfstate.backup
*.tfvars
*.pem
*.key
.env
```

1.2 Variable Configuration

assignment_part1_variables_tf.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat 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 = "AWS Availability Zone"
}

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

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

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

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

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $
```

assignment_part1_terraform_tfvars.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat terraform.tfvars
vpc_cidr_block    = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "ap-south-1a"
env_prefix        = "prod"
instance_type     = "t3.micro"
public_key        = "~/.ssh/id_ed25519.pub"
private_key       = "~/.ssh/id_ed25519"

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $
```

1.3 Networking Module

assignment_part1_networking_module_main.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/networking (main) $ cat main.tf
# Create VPC
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr_block
  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

# Create Subnet
resource "aws_subnet" "this" {
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.subnet_cidr_block
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true
  tags = {
    Name = "${var.env_prefix}-subnet"
  }
}

# Create Internet Gateway
resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id
  tags = {
    Name = "${var.env_prefix}-igw"
  }
}

# Create Route Table
resource "aws_route_table" "this" {
  vpc_id = aws_vpc.this.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.this.id
  }
  tags = {
    Name = "${var.env_prefix}-rt"
  }
}

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

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/networking (main) $
```

assignment_part1_networking_module_outputs.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/networking (main) $ vim outputs.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/networking (main) $ cat outputs.tf
output "vpc_id" {
  description = "ID of the created VPC"
  value       = aws_vpc.this.id
}

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

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

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

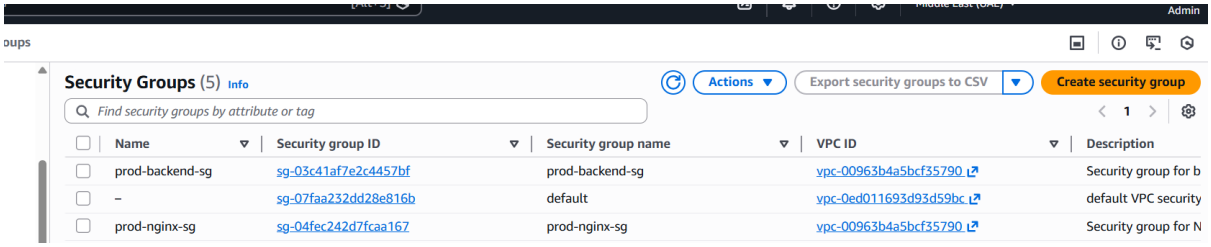
1.4 Security Module

assignment_part1_security_module.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/security (main) $ vim variables.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/security (main) $ vim main.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/security (main) $ cat main.tf
#Nginx Security Group
resource "aws_security_group" "nginx" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Security group for Nginx reverse proxy"
  vpc_id      = var.vpc_id

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

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

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

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

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

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

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

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

  egress {
    description = "Allow all outbound traffic"
```

assignment_part1_security_groups_console.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform state list
module.security.aws_security_group.backend
module.security.aws_security_group.nginx
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $
```



1.5 Locals Configuration

assignment_part1_locals_tf.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat locals.tf
# Detect your current public IP dynamically
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

locals {
  # Your public IP in /32 format for SSH rules
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"

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

  # Backend server configuration list
  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"
    }
  ]

  # Optional: resource naming convention
  resource_prefix = "${var.env_prefix}-assignment2"
}
```

Part 2: Webserver Module

2.1 Module Design

assignment_part2_webserver_module_variables.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ vim variables.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ cat variables.tf
variable "env_prefix" {
  description = "Environment prefix"
  type        = string
}

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

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

variable "availability_zone" {
  description = "AWS availability zone"
  type        = string
}

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

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

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

variable "public_key" {
  description = "Public key for EC2 SSH access"
  type        = string
}

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

variable "instance_suffix" {
  description = "Suffix for instance uniqueness"
  type        = string
}

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

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ cat variables.tf
```

assignment_part2_webserver_module_resources.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ vim main.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ cat  main.tf
#Create Key Pair
resource "aws_key_pair" "this" {
  key_name   = "${var.instance_name}-${var.instance_suffix}"
  public_key = var.public_key
}

# Create EC2 Instance
resource "aws_instance" "this" {
  ami                    = "ami-0abcdef1234567890" # Amazon Linux 2023 AMI ID, region ke hisaab se change karna
  instance_type          = var.instance_type
  availability_zone      = var.availability_zone
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = aws_key_pair.this.key_name
  user_data              = file(var.script_path)
  associate_public_ip_address = true

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

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $
```

assignment_part2_webserver_module_outputs.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ touch outputs.tf
puts.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ vim outputs.tf
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $ cat outputs.tf

Output "instance_id" {
  description = "The ID of the EC2 instance"
  value       = aws_instance.this.id
}

output "public_ip" {
  description = "The public IP address of the EC2 instance"
  value       = aws_instance.this.public_ip
}

output "private_ip" {
  description = "The private IP address of the EC2 instance"
  value       = aws_instance.this.private_ip
}

@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2/modules/webserver (main) $
```

2.2 Module Usage

assignment_part2_root_module_integration.png

```
@SadafRiaz-077 ➜ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat main.tf
      script_path = "./scripts/apache-setup.sh"
      suffix      = "web1"
    },
    {
      name        = "web-2"
      script_path = "./scripts/apache-setup.sh"
      suffix      = "web2"
    },
    {
      name        = "web-3"
      script_path = "./scripts/apache-setup.sh"
      suffix      = "web3"
    }
  ]
}

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

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

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

Part 3: Server Configuration Scripts

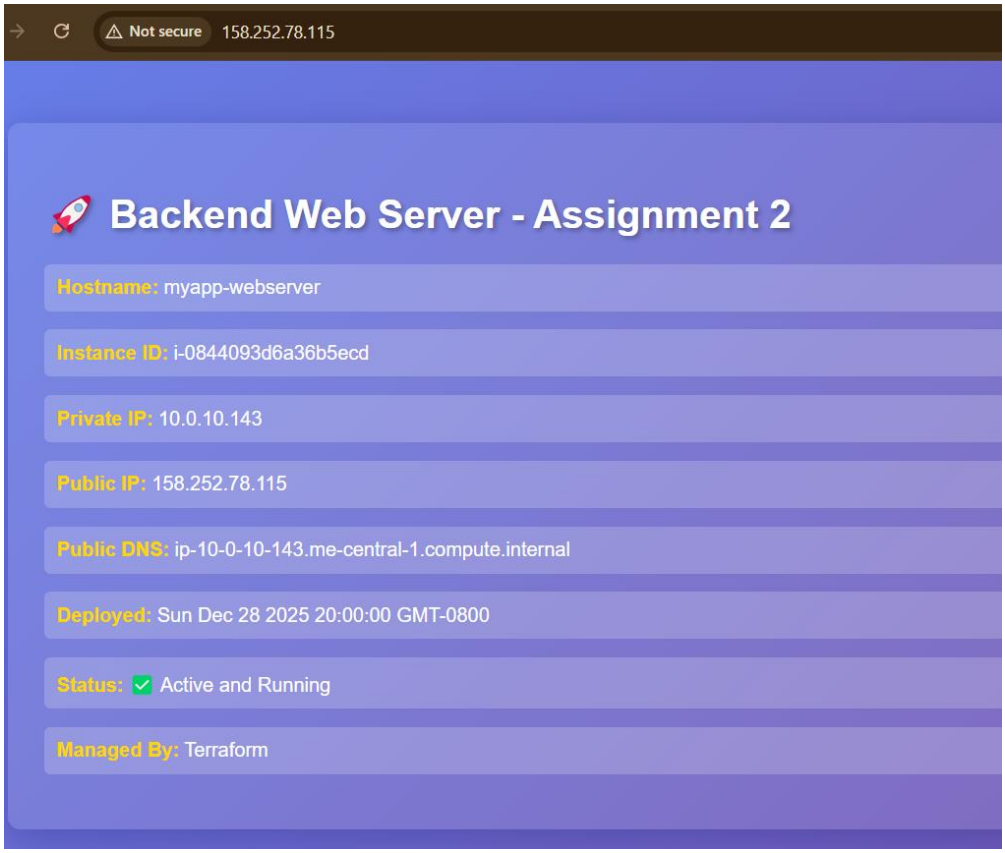
3.1 Apache Backend Server Script

- Screenshot: assignment_part3_apache_script.png

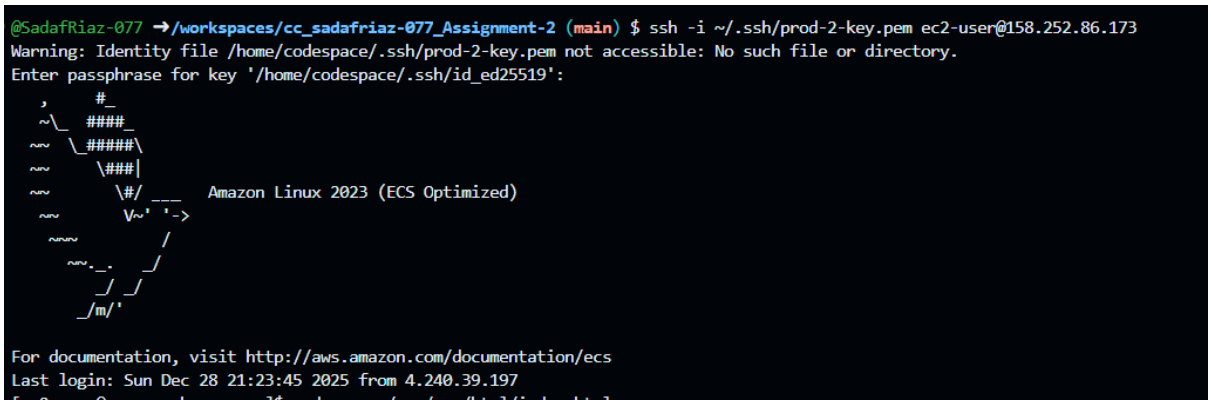
```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2/scripts (main) $ cat apache-setup.sh
# Install Apache
yum install httpd -y

# Start and enable Apache service
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: $TOKEN" \
  http://169.254.169.254/latest/meta-data/public-ipv4)
HOSTNAME=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/hostname)
INSTANCE_ID=$(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>
</head>
<body>
    <h1>Backend Web Server</h1>
    <p><b>Hostname:</b> $HOSTNAME</p>
    <p><b>Instance ID:</b> $INSTANCE_ID</p>
    <p><b>Private IP:</b> $PRIVATE_IP</p>
    <p><b>Public IP:</b> $PUBLIC_IP</p>
    <p><b>Deployment Time:</b> $(date)</p>
    <p><b>Status:</b> ✅ Active</p>
</body>
</html>
EOF

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

echo "Apache backend setup completed!"
```

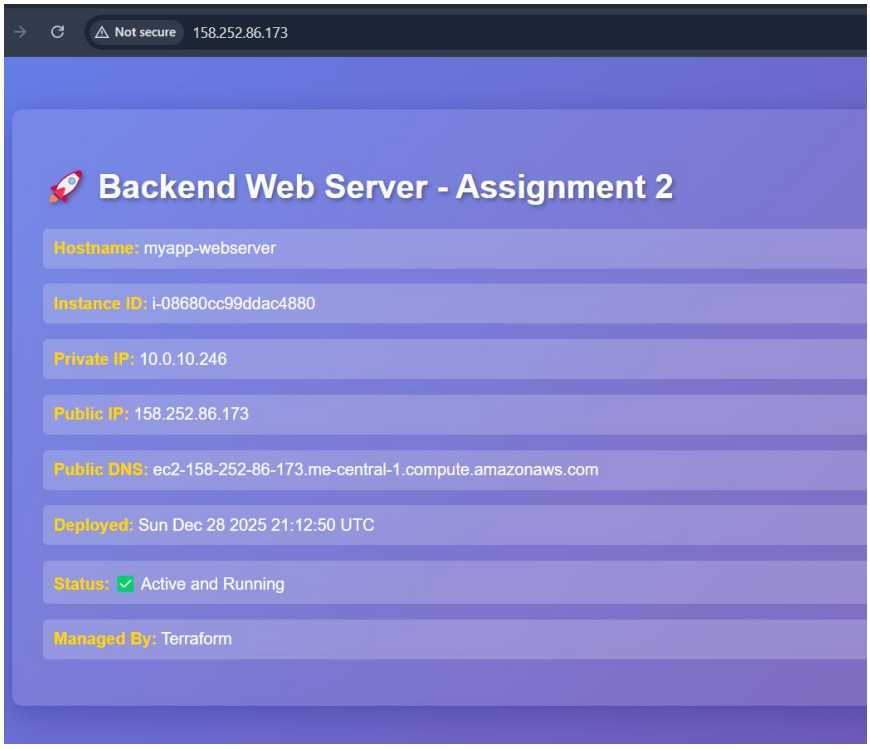
- Screenshot: assignment_part3_backend_webpage.png (browser showing backend server page)

Web 1

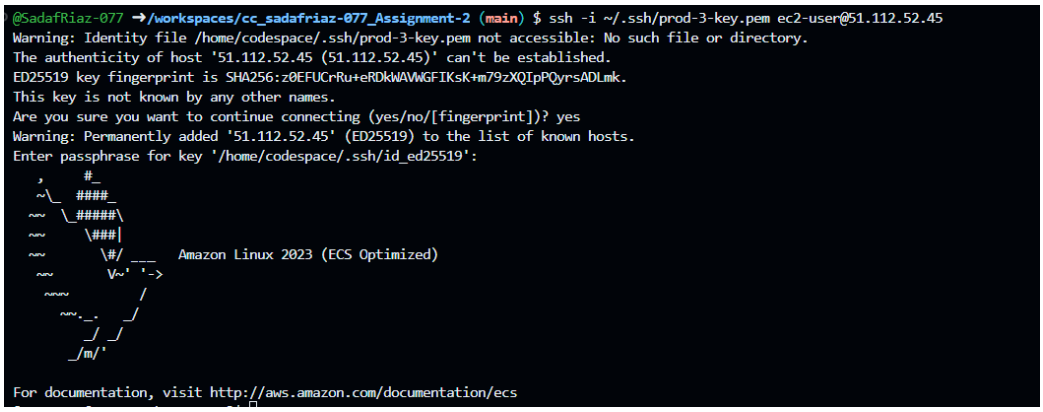

Not secure
158.252.78.115
Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-0844093d6a36b5ecd
Private IP: 10.0.10.143
Public IP: 158.252.78.115
Public DNS: ip-10-0-10-143.me-central-1.compute.internal
Deployed: Sun Dec 28 2025 20:00:00 GMT-0800
Status: Active and Running
Managed By: Terraform

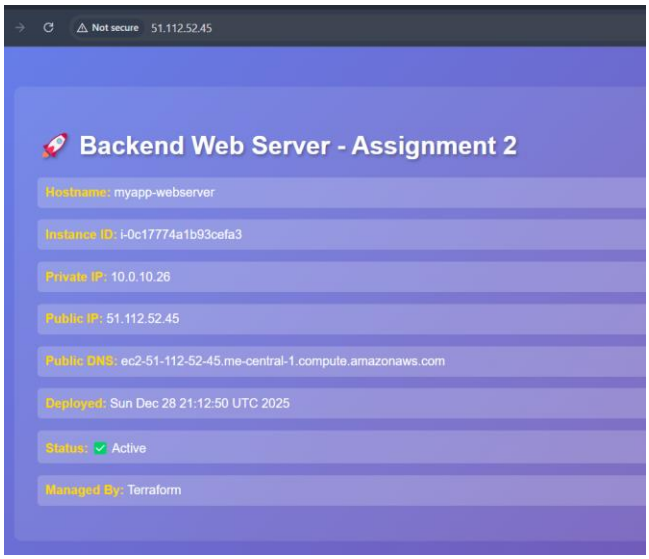
Web 2


@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i ~/.ssh/prod-2-key.pem ec2-user@158.252.86.173
Warning: Identity file /home/codespace/.ssh/prod-2-key.pem not accessible: No such file or directory.
Enter passphrase for key '/home/codespace/.ssh/id_ed25519':
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
Last login: Sun Dec 28 21:23:45 2025 from 4.240.39.197


Not secure
158.252.86.173
Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-08680cc99ddac4880
Private IP: 10.0.10.246
Public IP: 158.252.86.173
Public DNS: ec2-158-252-86-173.me-central-1.compute.amazonaws.com
Deployed: Sun Dec 28 2025 21:12:50 UTC
Status: Active and Running
Managed By: Terraform

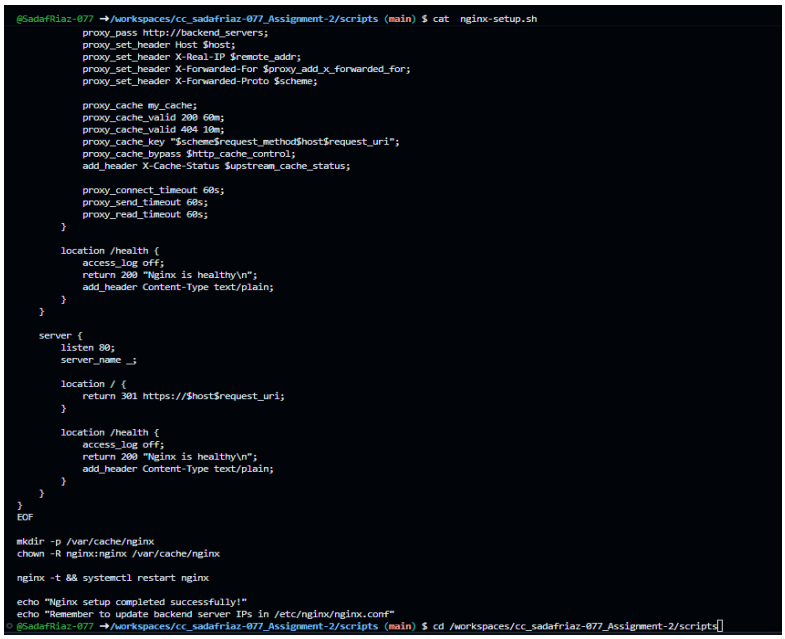
Web 3


@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i ~/.ssh/prod-3-key.pem ec2-user@51.112.52.45
Warning: Identity file /home/codespace/.ssh/prod-3-key.pem not accessible: No such file or directory.
The authenticity of host '51.112.52.45 (51.112.52.45)' can't be established.
ED25519 key fingerprint is SHA256:z0EFUCrRu+eRDkWAVWGFIKsK+m79zXQIpPQyrsADLmk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.52.45' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/codespace/.ssh/id_ed25519':
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs


Not secure
51.112.52.45
Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-0c17774a1b93cefa3
Private IP: 10.0.10.26
Public IP: 51.112.52.45
Public DNS: ec2-51-112-52-45.me-central-1.compute.amazonaws.com
Deployed: Sun Dec 28 21:12:50 UTC 2025
Status: Active
Managed By: Terraform

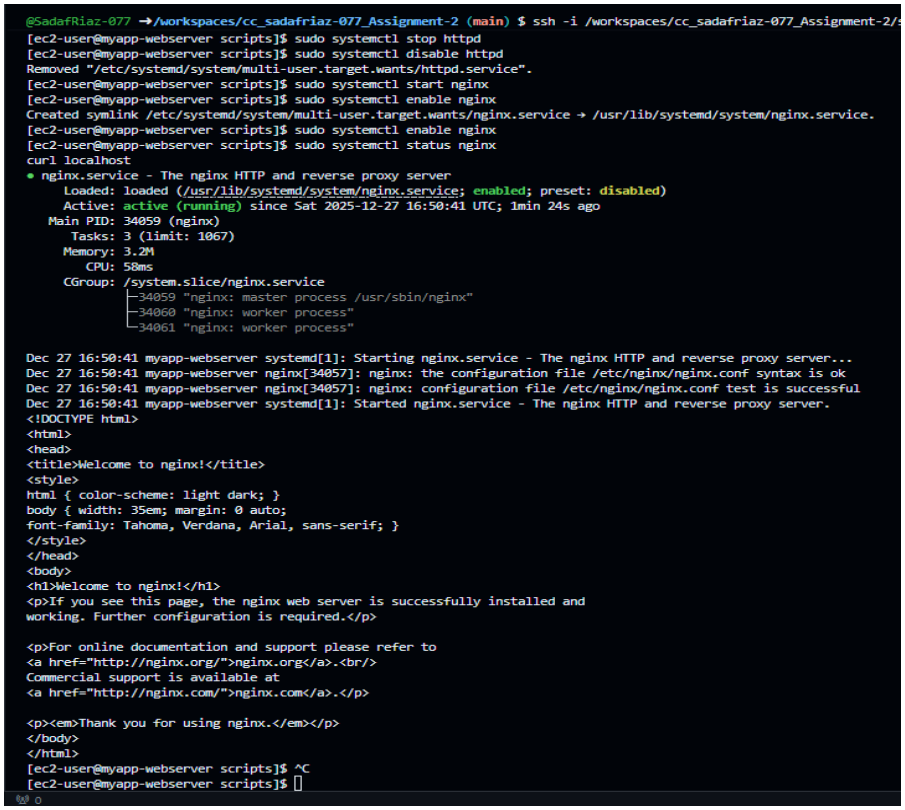
3.2 Nginx Server Setup Script

Screenshot: assignment_part3_nginx_script.png

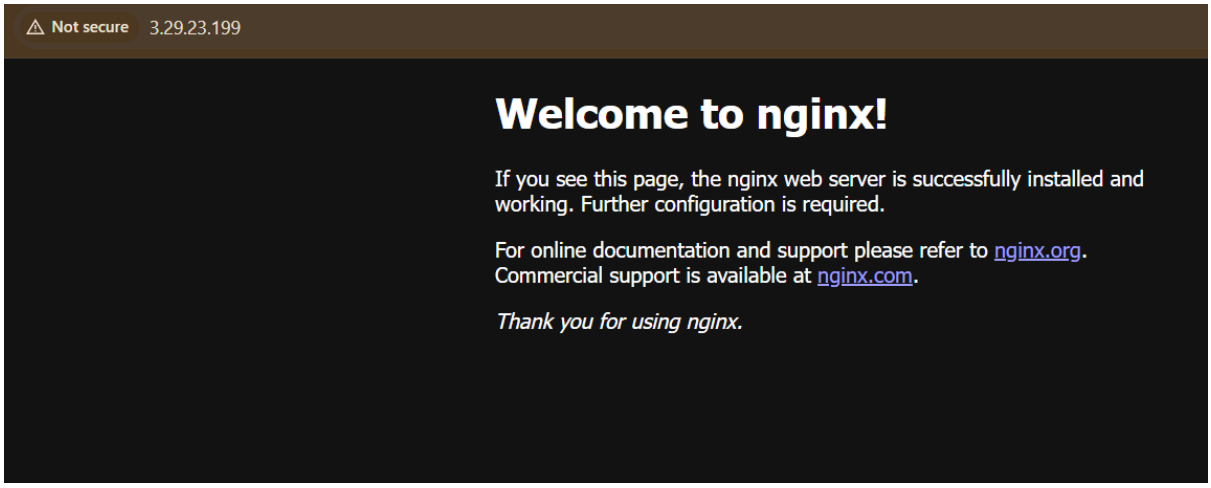


```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2/scripts (main) $ cat  nginx-setup.sh
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            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

mkdir -p /var/cache/nginx
chown -R nginx:nginx /var/cache/nginx

nginx -t && systemctl restart nginx

echo "Nginx setup completed successfully!"
echo "Remember to update backend server IPs in /etc/nginx/nginx.conf"
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2/scripts (main) $ cd /workspaces/cc_sadafriaz-077_Assignment-2/scripts
```

- **Screenshot: assignment_part3_nginx_default_page.png (before backend configuration)**



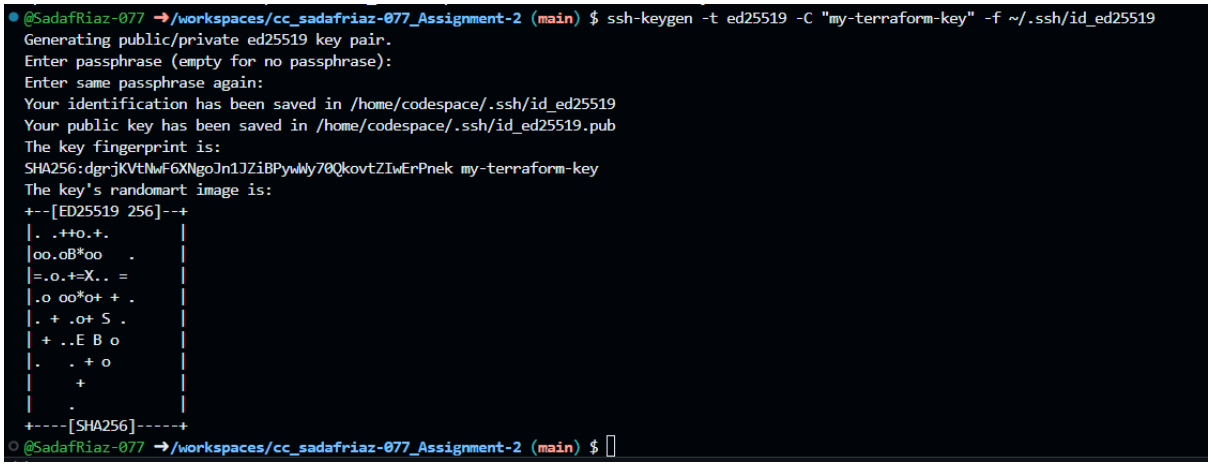
```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i /workspaces/cc_sadafriaz-077_Assignment-2/
[ec2-user@myapp-webserver scripts]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver scripts]$ sudo systemctl disable httpd
Removed "/etc/systemd/system/multi-user.target.wants/httpd.service".
[ec2-user@myapp-webserver scripts]$ sudo systemctl start nginx
[ec2-user@myapp-webserver scripts]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@myapp-webserver scripts]$ sudo systemctl enable nginx
[ec2-user@myapp-webserver scripts]$ sudo systemctl status nginx
curl localhost
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-12-27 16:50:41 UTC; 1min 24s ago
   Main PID: 34059 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.2M
        CPU: 58ms
     CGroup: /system.slice/nginx.service
             ├─34059 "nginx: master process /usr/sbin/nginx"
             ├─34060 "nginx: worker process"
             └─34061 "nginx: worker process"

Dec 27 16:50:41 myapp-webserver systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Dec 27 16:50:41 myapp-webserver nginx[34057]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Dec 27 16:50:41 myapp-webserver nginx[34057]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Dec 27 16:50:41 myapp-webserver systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
[ec2-user@myapp-webserver scripts]$ ^C
[ec2-user@myapp-webserver scripts]$
```



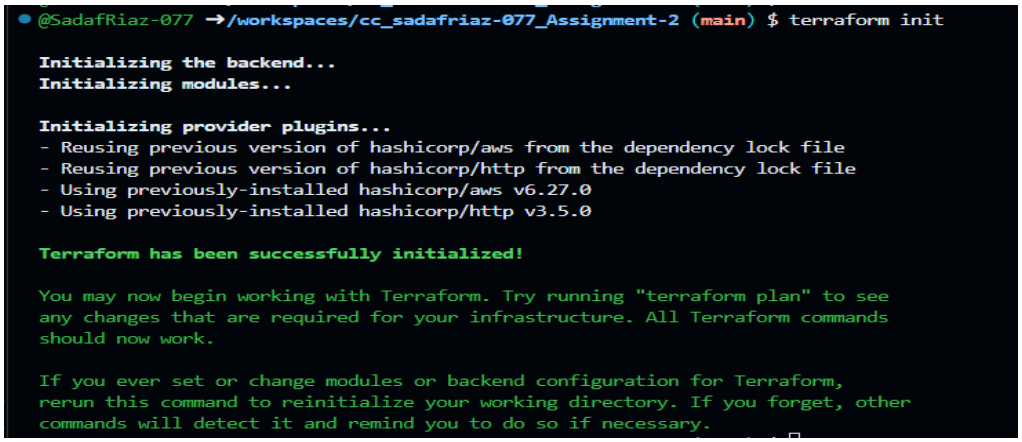
Part 4: Infrastructure Deployment

4.1 Initial Deployment

assignment_part4_ssh_key_generation.png



assignment_part4_terraform_initialization.png



assignment_part4_terraform_validation.png

```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform validate
Success! The configuration is valid.
```

assignment_part4_terraform_plan.png

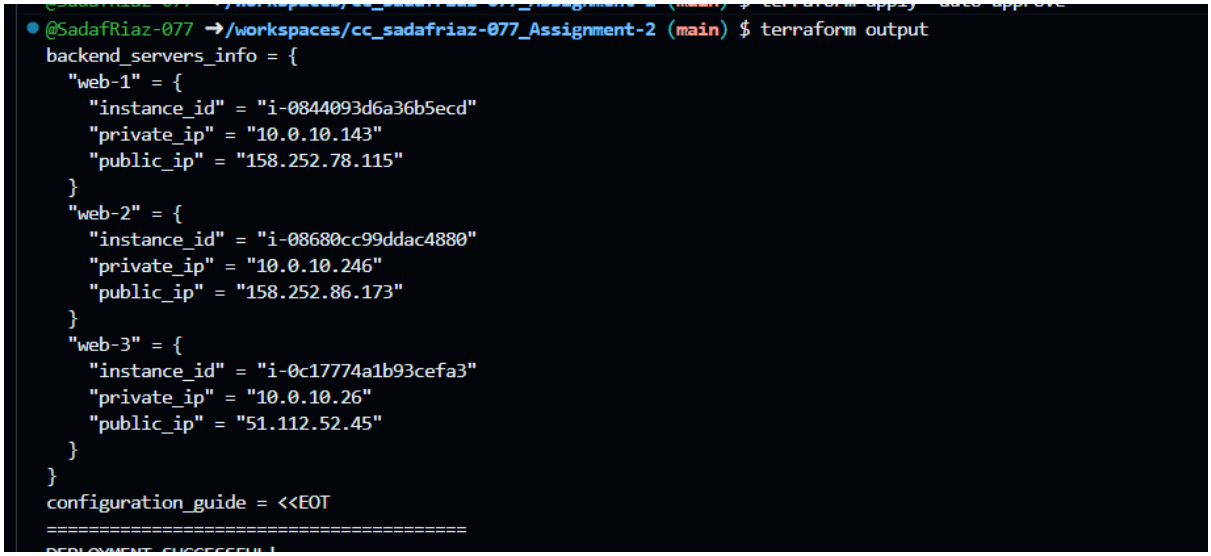
```
      ~ owner_id              = "404842057573" -> (known after apply)
        tags                  = {
            "Name" = "prod-nginx-sg"
        }
      ~ vpc_id                = "vpc-00963b4a5bcf35790" # forces replacement -> (known after apply) # forces replacement
        # (5 unchanged attributes hidden)
    }

Plan: 15 to add, 0 to change, 2 to destroy.
```

assignment_part4_terraform_apply.png

```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform apply
            nginx -t && systemctl restart nginx

            echo "Nginx setup completed successfully!"
            echo "Remember to update backend server IPs in /etc/nginx/nginx.conf"
        EOT
      + user_data_base64                   = (known after apply)
      + user_data_replace_on_change        = false
      + vpc_security_group_ids             = [
          + "sg-078be2e2b0b56e699",
        ]
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

module.nginx_server.aws_instance.this: Creating...
module.backend_servers["web-3"].aws_instance.this: Creating...
module.backend_servers["web-2"].aws_instance.this: Creating...
module.backend_servers["web-1"].aws_instance.this: Creating...
module.nginx_server.aws_instance.this: Still creating... [10s elapsed]
module.backend_servers["web-3"].aws_instance.this: Still creating... [10s elapsed]
module.backend_servers["web-2"].aws_instance.this: Still creating... [10s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still creating... [10s elapsed]
module.nginx_server.aws_instance.this: Creation complete after 12s [id=i-00691bc60e0b56ea3]
module.backend_servers["web-1"].aws_instance.this: Creation complete after 12s [id=i-0844093d6a36b5ecd]
module.backend_servers["web-3"].aws_instance.this: Creation complete after 12s [id=i-0c17774a1b93cefa3]
module.backend_servers["web-2"].aws_instance.this: Creation complete after 12s [id=i-08680cc99ddac4880]

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

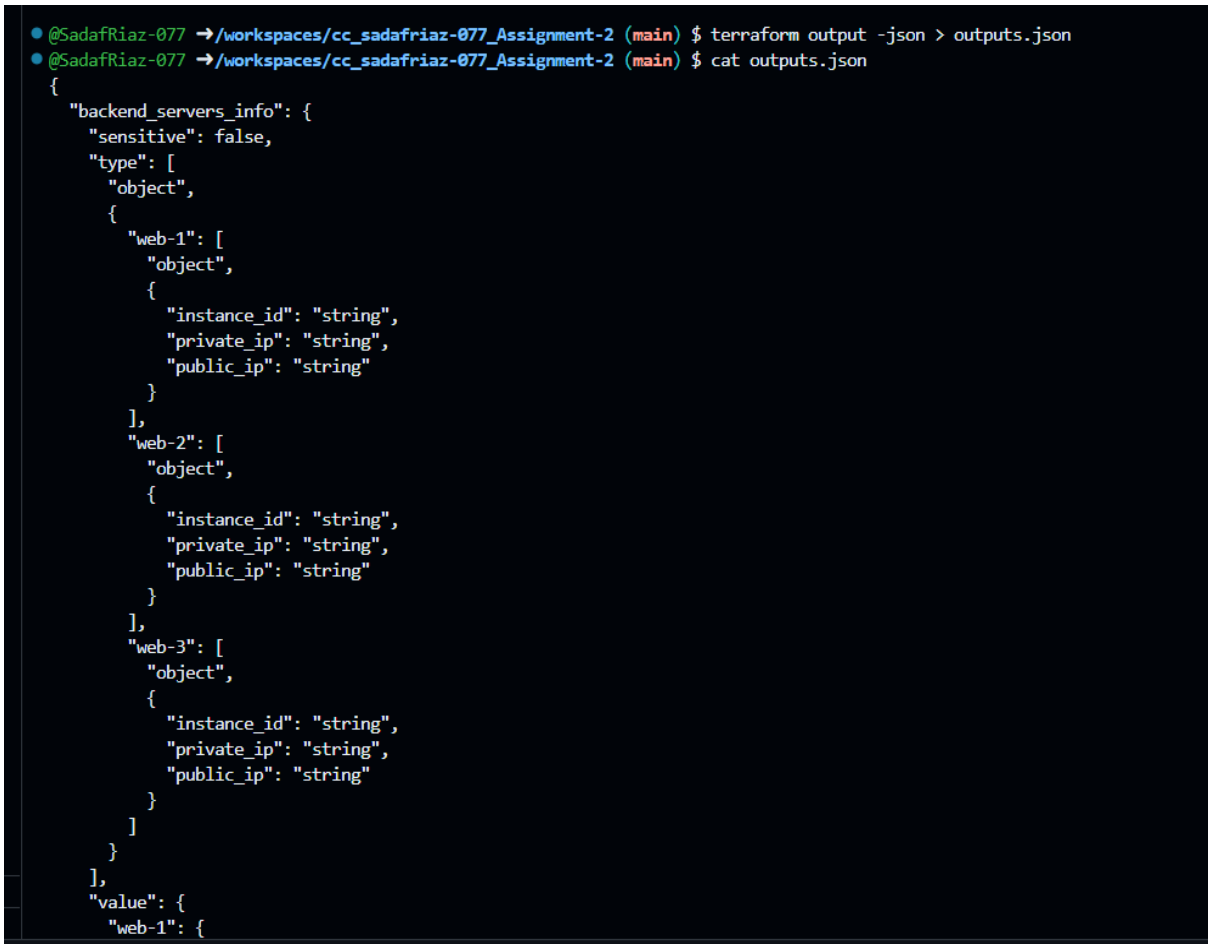
4.2 Output Configuration

assignment_part4_terraform_output_display.png



```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform output
backend_servers_info = {
  "web-1" = {
    "instance_id" = "i-0844093d6a36b5ecd"
    "private_ip" = "10.0.10.143"
    "public_ip" = "158.252.78.115"
  }
  "web-2" = {
    "instance_id" = "i-08680cc99ddac4880"
    "private_ip" = "10.0.10.246"
    "public_ip" = "158.252.86.173"
  }
  "web-3" = {
    "instance_id" = "i-0c17774a1b93cefa3"
    "private_ip" = "10.0.10.26"
    "public_ip" = "51.112.52.45"
  }
}
configuration_guide = <<EOT
========================================
```

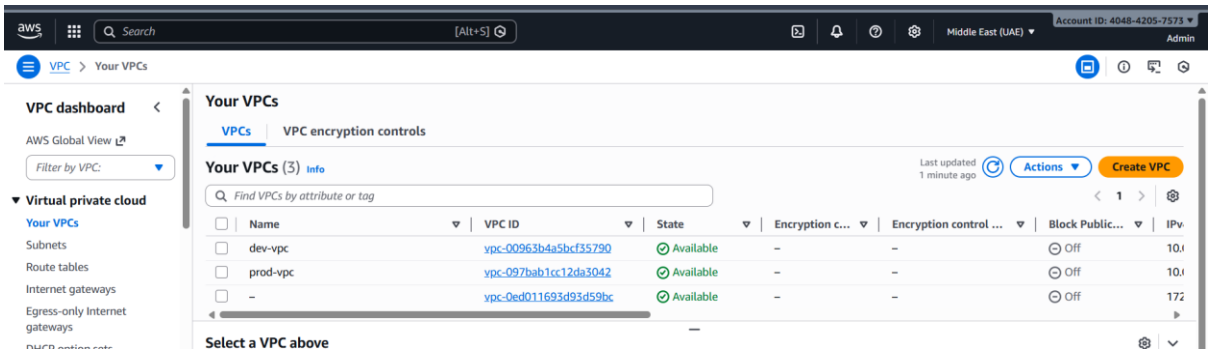
assignment_part4_terraform_outputs_json.png



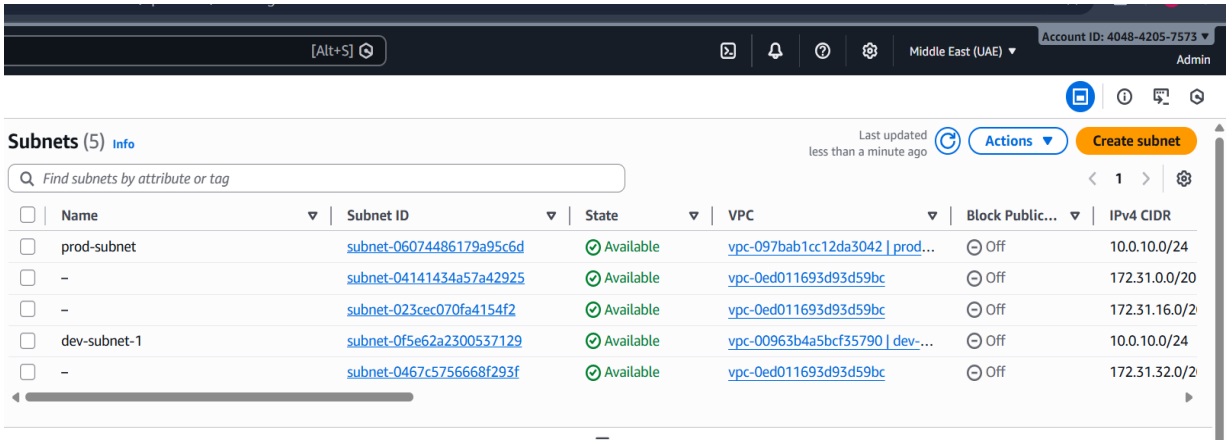
```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform output -json > outputs.json
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ cat outputs.json
{
  "backend_servers_info": {
    "sensitive": false,
    "type": [
      "object",
      {
        "web-1": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-2": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-3": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ]
      }
    ],
    "value": {
      "web-1": {
```

4.3 AWS Console Verification

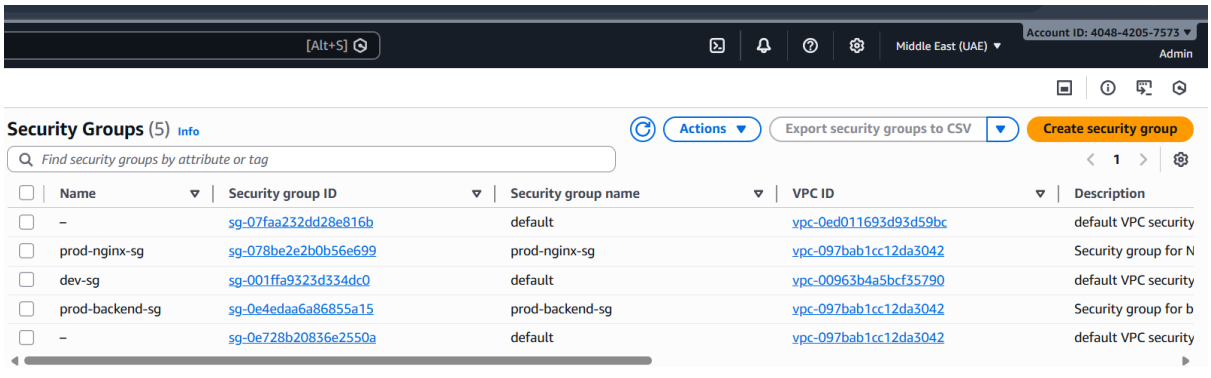
assignment_part4_aws_vpc_verification.png



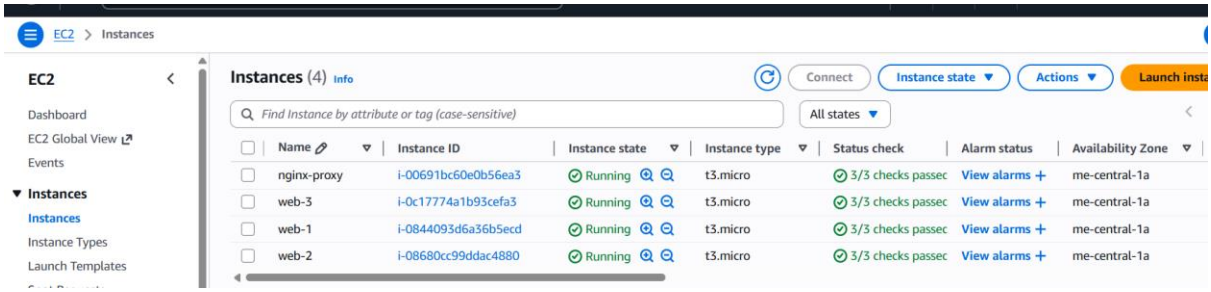
assignment_part4_aws_subnet_verification.png



assignment_part4_aws_security_groups_verification.png



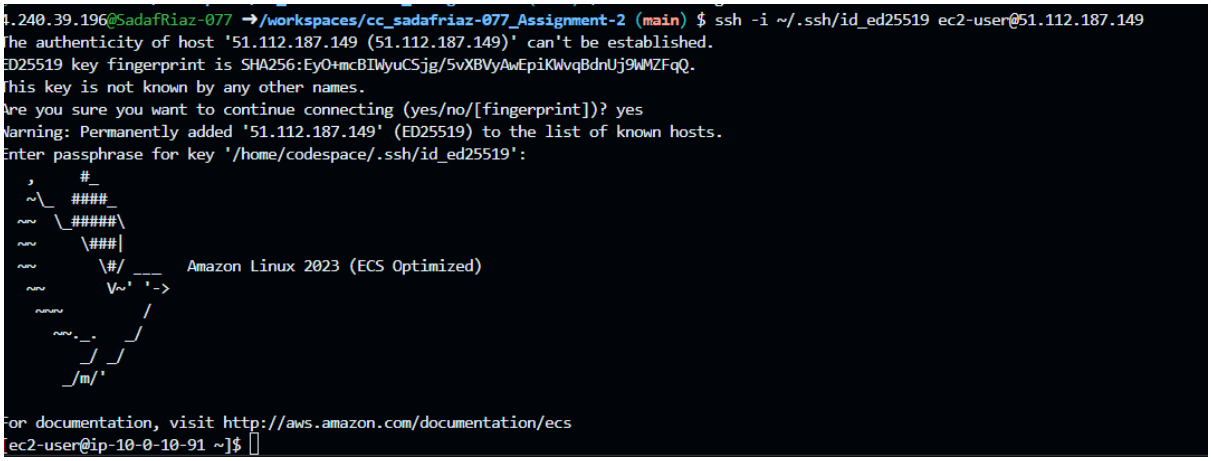
assignment_part4_aws_ec2_instances_verification.png



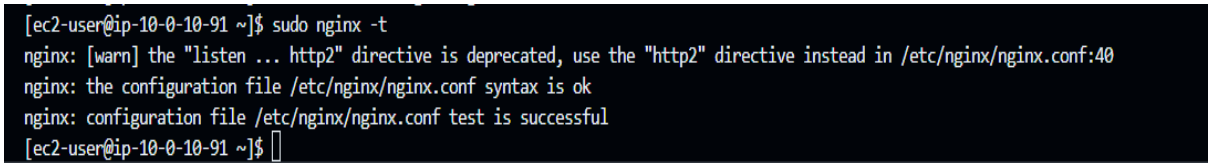
Part 5: Nginx Configuration & Testing

5.1 Update Nginx Backend Configuration

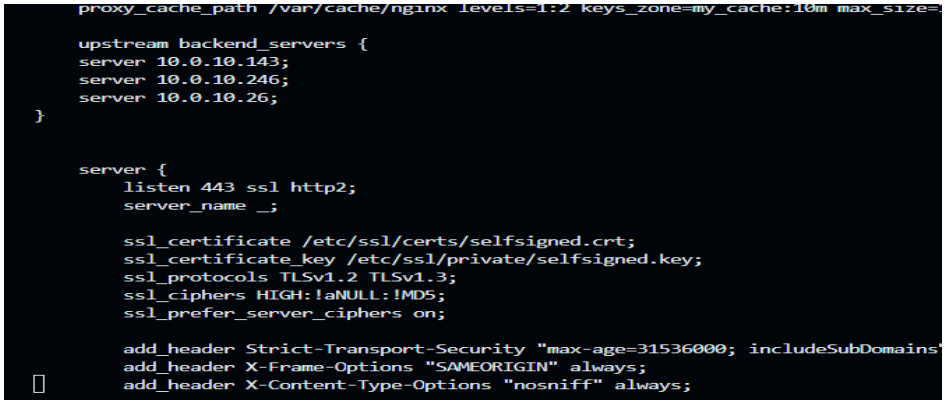
assignment_part5_ssh_into_nginx.png



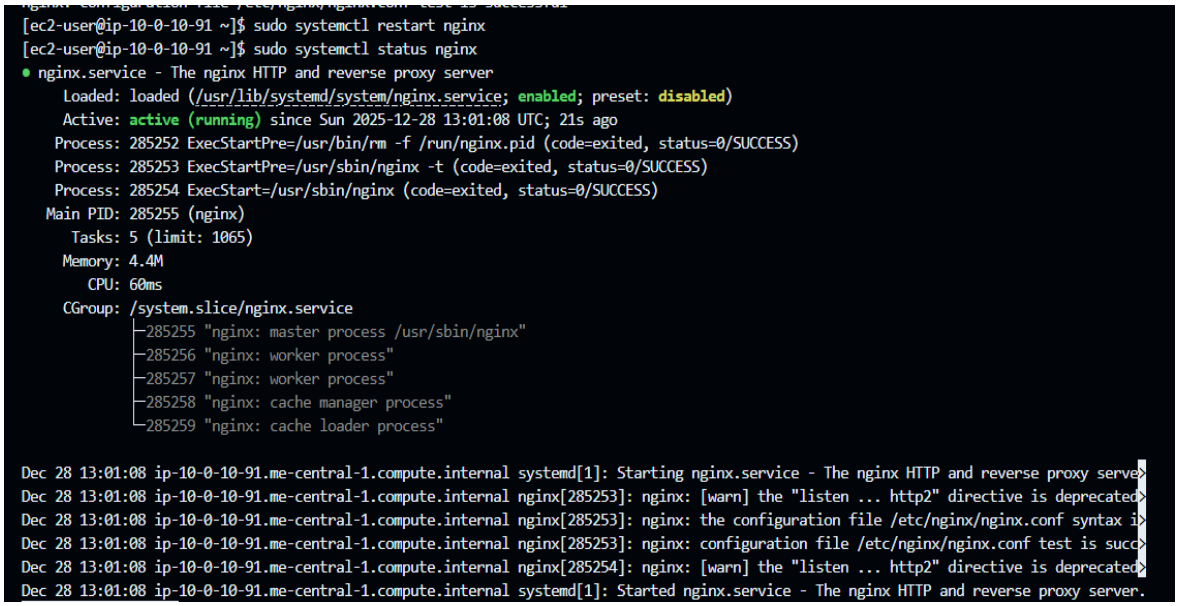
assignment_part5_updated_nginx_configuration.png



assignment_part5_nginx_configuration_test.png

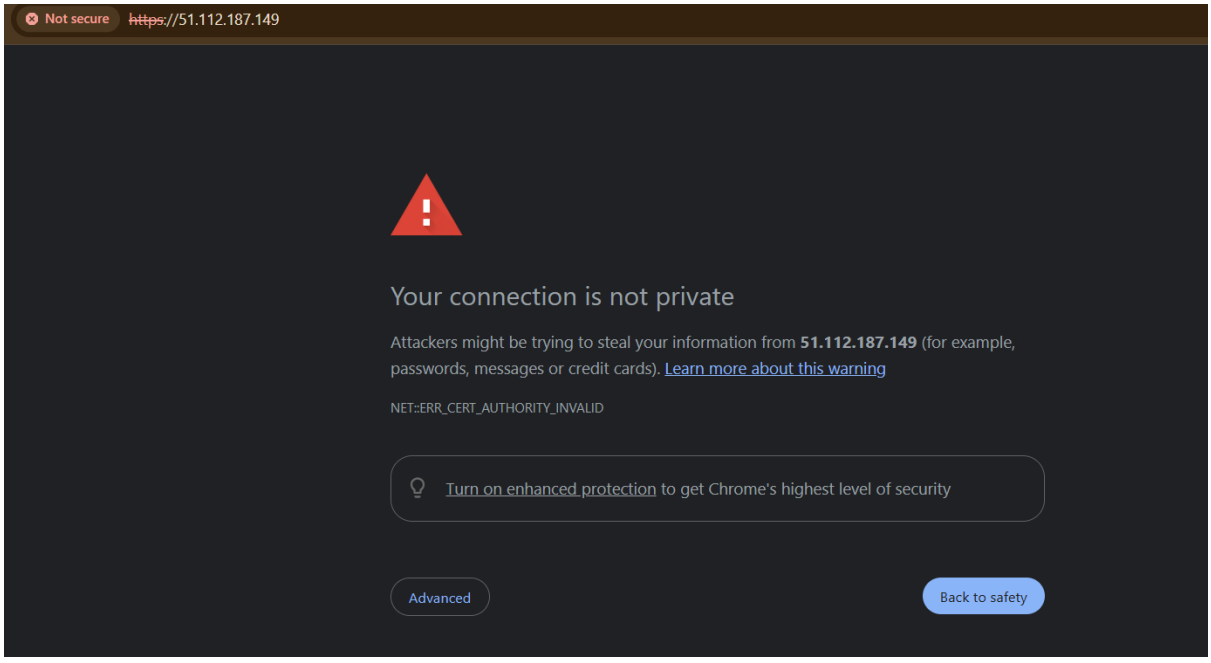


assignment_part5_nginx_restart.png

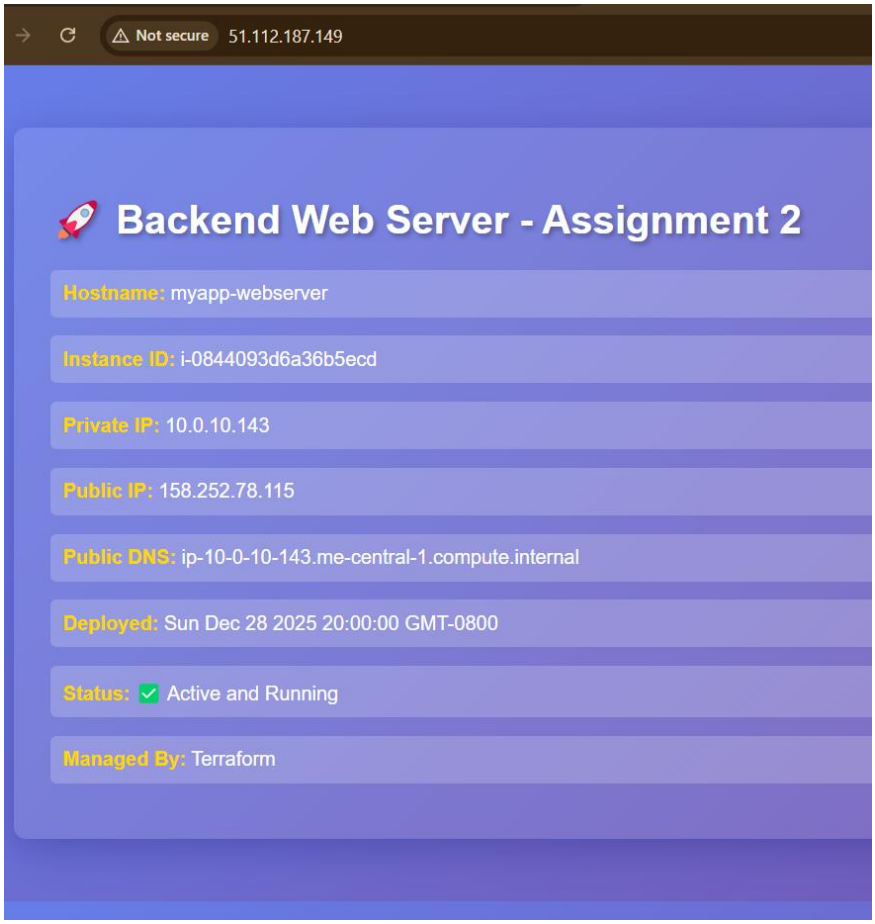


5.2 Test Load Balancing

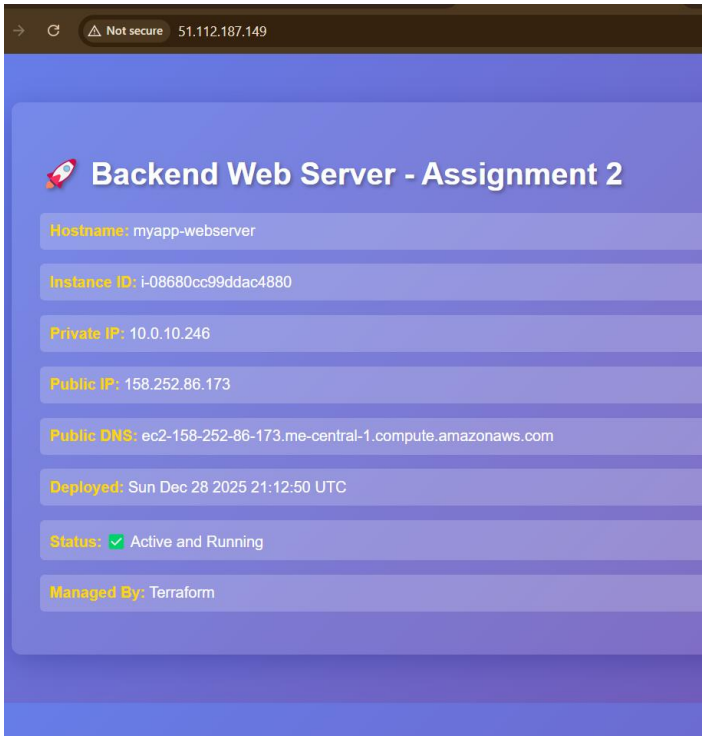
assignment_part5_nginx_restart.png



assignment_part5_web1_response.png

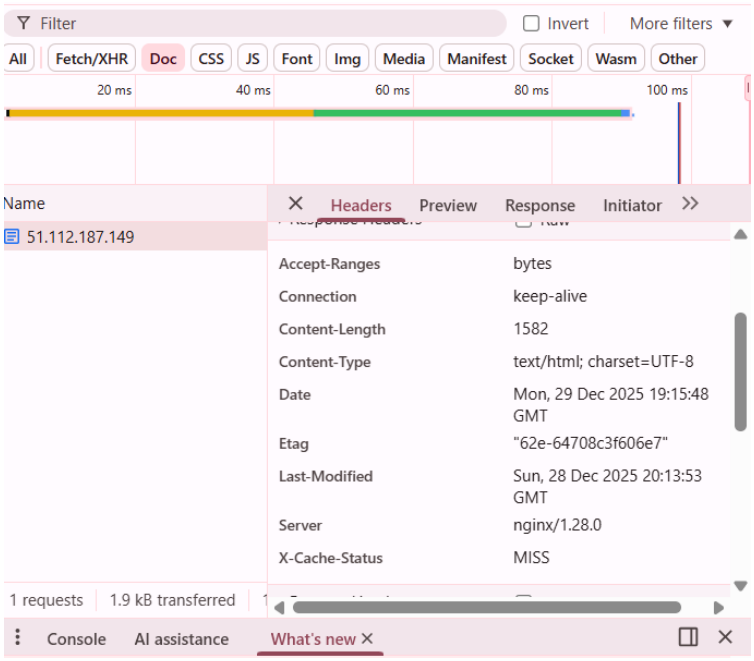


assignment_part5_web2_response.png

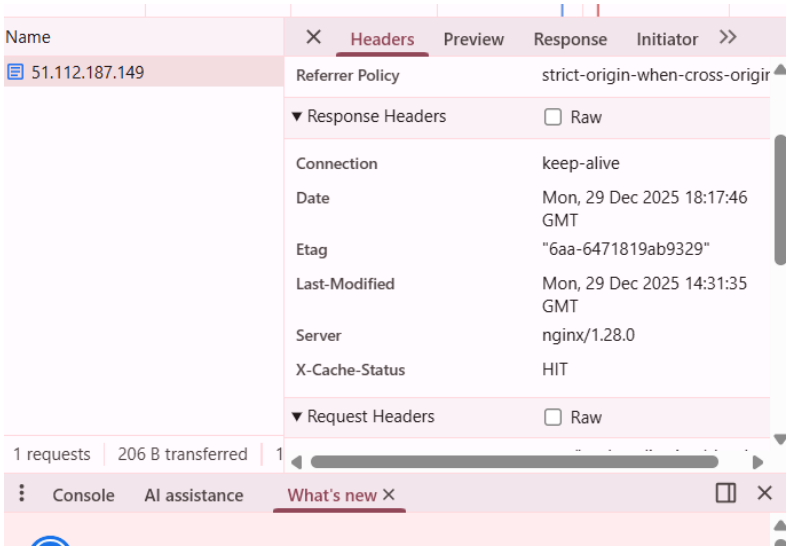


5.3 Test Cache Functionality

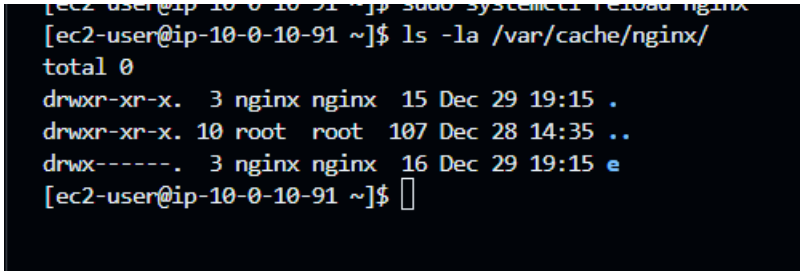
assignment_part5_cache_miss_verification.png



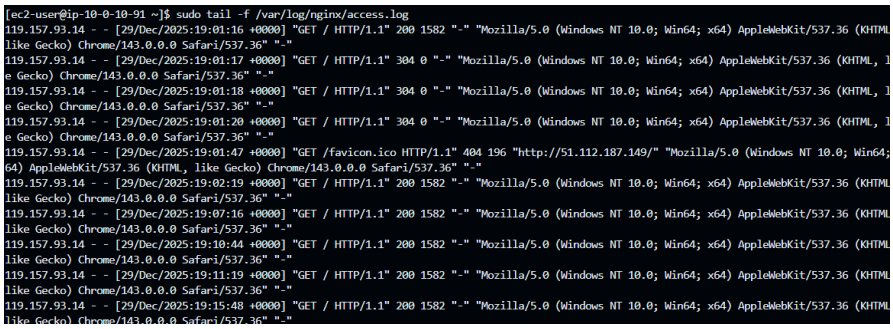
assignment_part5_cache_hit_verification.png



assignment_part5_nginx_cache_directory.png

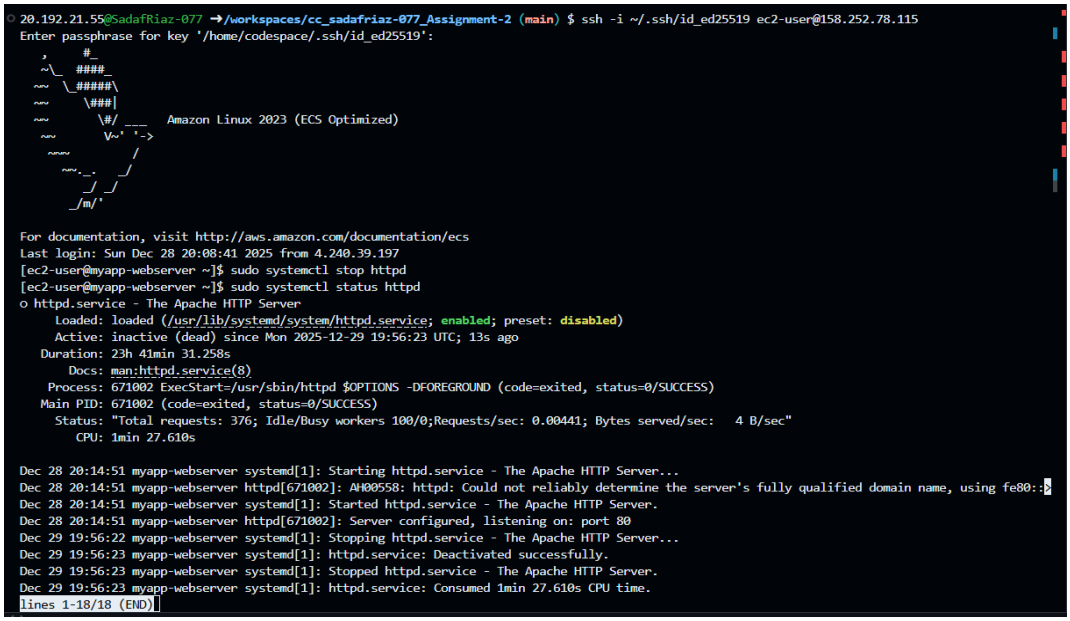


assignment_part5_access_log_cache_status.png

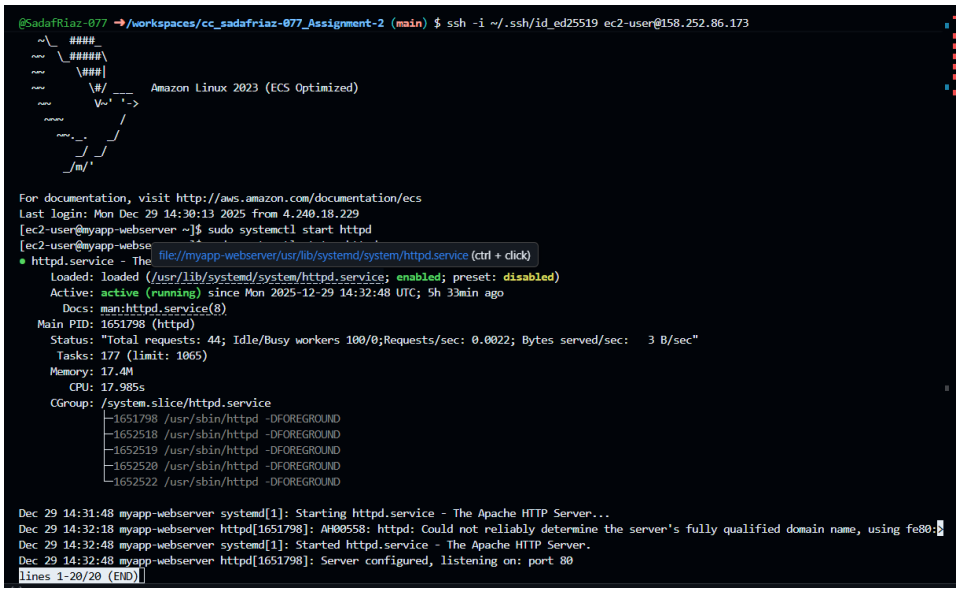


5.4 Test High Availability (Backup Server)

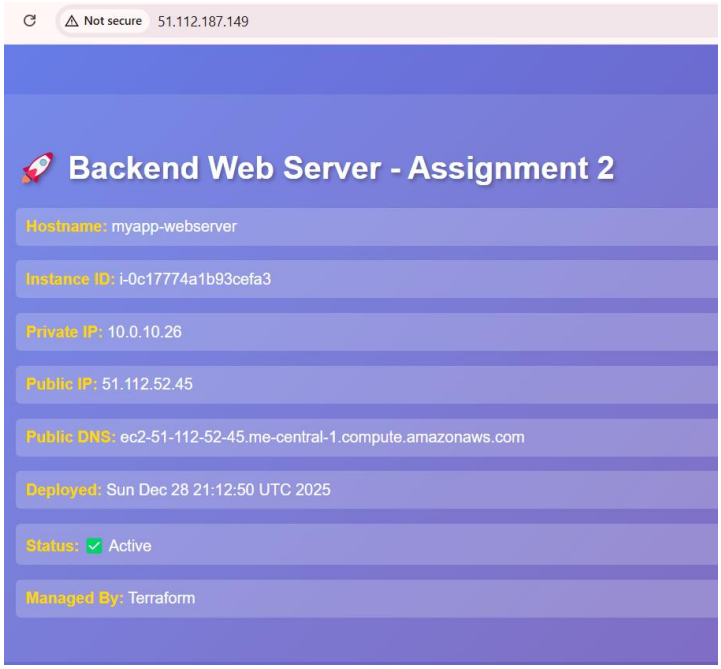
assignment_part5_web1_service_stopped.png



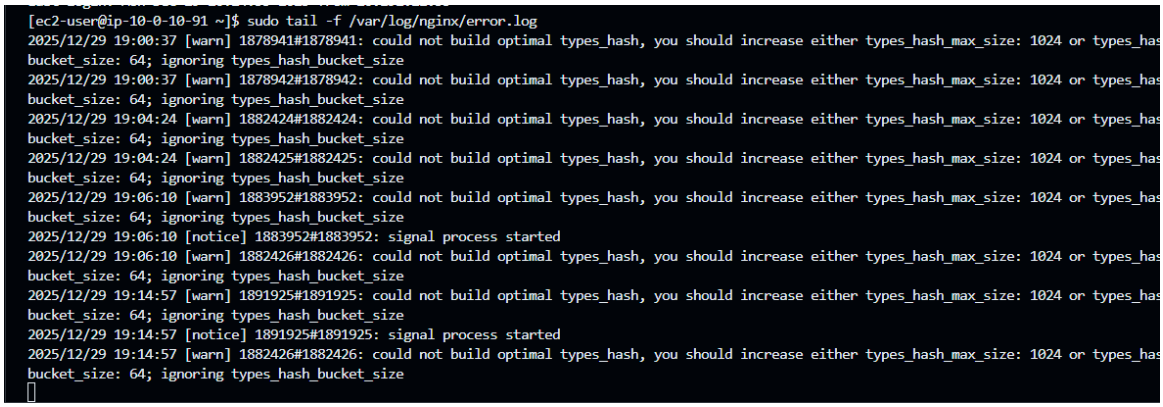
assignment_part5_web2_service_stopped.png



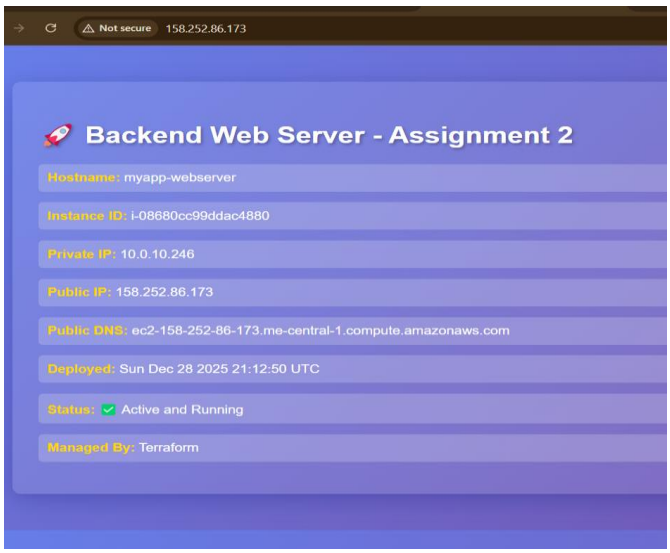
assignment_part5_backup_server_activated.png



assignment_part5_nginx_error_log.png



assignment_part5_services_restored.png



5.5 Security & Performance Analysis

assignment_part5_ssl_certificate_details.png

```
@SadafRiaz-077 ➜/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ sudo openssl x509 -in /etc/nginx/ssl/nginx.crt -text -noout

        Subject: C = PK, ST = Punjab, L = Lahore, O = MyCompany, OU = IT, CN = 51.112.187.149
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:cc:bb:2c:a0:56:ea:1f:57:df:21:3a:f6:67:f6:
                    87:6b:ab:b1:c3:db:a1:e6:c4:3a:39:67:5a:b2:c5:
                    81:76:34:b0:2c:49:1b:29:91:36:eb:95:b4:93:3f:
                    ae:42:2f:d7:4e:b8:37:b4:da:a5:ff:43:df:67:ce:
                    36:ba:18:df:19:d2:4e:f4:c3:b3:e4:24:e9:04:c1:
                    1f:56:9e:fe:0a:ad:47:8a:72:9f:ee:0c:f7:ec:84:
                    d9:77:f9:e7:7e:8b:10:98:76:66:e8:86:1c:59:9f:
                    1b:9c:0b:ca:3f:2e:44:6a:d2:ec:bc:d7:40:18:8e:
                    55:ff:14:74:be:04:d8:72:db:ec:46:f7:15:42:c1:
                    e9:97:5a:9a:27:99:d3:24:d6:6e:d9:3c:00:00:5b:
                    3c:99:e9:c4:14:73:3f:2d:ef:d3:13:fc:e8:4c:d4:
                    fa:30:24:75:d8:a5:6a:7c:8d:50:88:d1:d9:f9:72:
                    6d:51:e1:ab:02:50:b4:4f:e2:92:e5:f4:5f:31:d3:
                    2e:52:f7:a7:b7:e0:c8:e6:68:9a:7f:96:43:9d:73:
                    53:d3:ba:c7:8b:b9:7d:f3:a4:c8:ca:b4:ce:87:aa:
                    45:0e:91:32:1a:41:76:75:70:77:0f:8e:b3:66:b5:
                    1d:70:40:9e:aa:16:1e:07:4b:0a:4a:bc:ec:0e:48:
                    fb:1b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                70:32:10:07:3A:F6:C2:39:B8:32:83:37:A6:82:C6:F8:71:C1:50:C2
            X509v3 Authority Key Identifier:
                70:32:10:07:3A:F6:C2:39:B8:32:83:37:A6:82:C6:F8:71:C1:50:C2
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        27:a5:54:29:1b:6c:f9:c5:5a:42:78:a2:94:b5:6c:ee:67:41:
        0f:da:a4:19:2b:95:29:f1:32:a1:94:87:c2:a2:f0:71:8b:c2:
        84:9d:5e:af:94:99:30:0b:29:30:81:5b:39:77:b4:81:36:9f:
        cc:af:f8:de:22:27:4d:5a:61:bb:50:7d:e0:35:a5:18:cb:2a:
        36:bd:63:11:ea:4a:14:9f:e7:96:e7:78:cc:12:94:20:2e:9a:
        f8:f9:0d:57:cd:a6:9d:ad:96:4f:d8:4a:ec:36:1f:05:bc:ea:
        c9:dd:7e:25:f7:2f:47:67:de:af:87:ea:de:44:ef:e0:d9:4a:
        29:bc:b5:bf:15:d2:a5:0b:e9:63:f4:04:39:9f:1b:1c:c4:a9:
        4a:85:54:f0:b5:c3:0f:16:05:fb:d4:b7:1b:7e:2c:32:64:b1:
        42:48:7e:b1:de:24:57:56:84:45:07:ed:7d:44:72:45:00:28:
        36:ab:fa:17:11:f4:0f:ef:b8:b8:a1:f9:8e:9d:ec:08:59:9d:
```

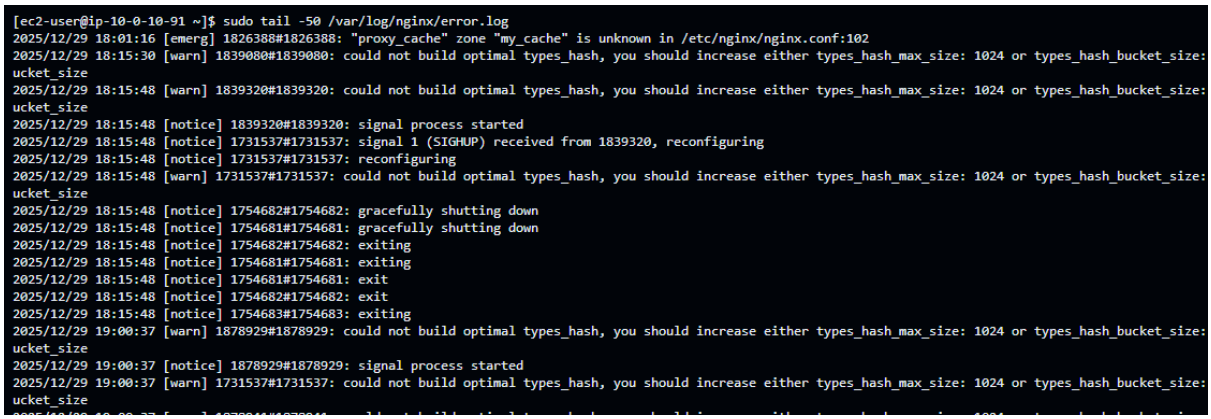
assignment_part5_security_headers_verification.png

```
[ec2-user@ip-10-0-10-91 ~]$ curl -I -k https://51.112.187.149
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Mon, 29 Dec 2025 22:31:21 GMT
Content-Type: text/html
Content-Length: 30
Last-Modified: Mon, 29 Dec 2025 18:15:59 GMT
Connection: keep-alive
ETag: "6952c55f-1e"
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Referrer-Policy: no-referrer-when-downgrade
Content-Security-Policy: default-src 'self'
Accept-Ranges: bytes

[ec2-user@ip-10-0-10-91 ~]$
```

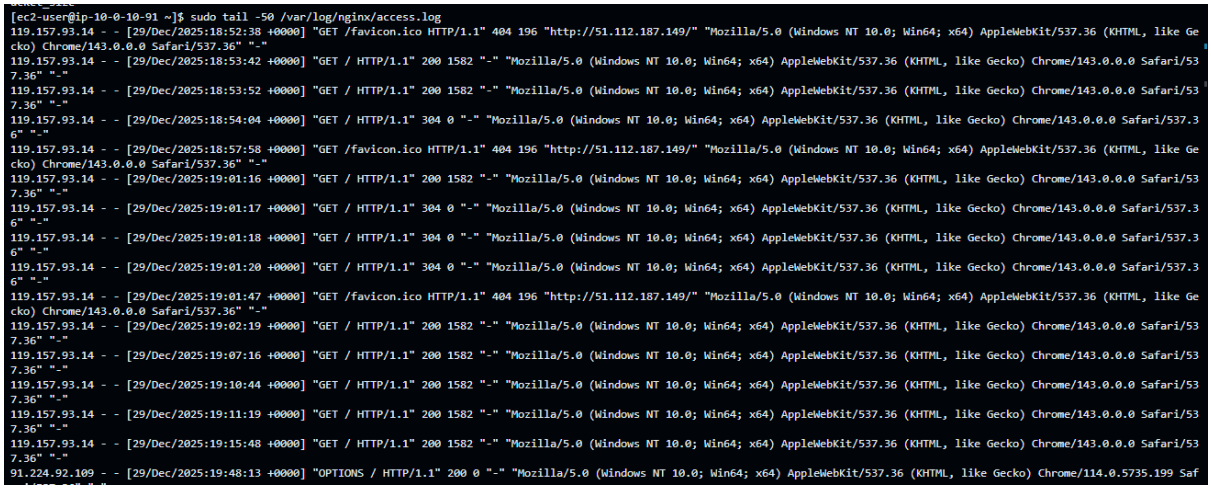
assignment_part5_http_to_https_redirect.png

```
[ec2-user@ip-10-0-10-91 ~]$ curl -I http://51.112.187.149
HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Mon, 29 Dec 2025 22:32:28 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://51.112.187.149/
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000; includeSubDomains
Referrer-Policy: no-referrer-when-downgrade
Content-Security-Policy: default-src 'self'

[ec2-user@ip-10-0-10-91 ~]$
```

assignment_part5_error_log_analysis.png



assignment_part5_access_log_analysis.png



Bonus Tasks

Bonus 1: Custom Error Pages

assignment_bonus_custom_404.png

assignment_bonus_custom_502.png

Bonus 2: Implement Rate Limiting

assignment_bonus_rate_limiting_configuration.png

```
        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;

        limit_req_zone $binary_remote_addr zone=mylimit:10m rate=5r/s;

        }
```

Bonus 3: Health Check Automation

assignment_bonus_health_check_script.png

```
Last login: Tue Dec 30 08:13:07 2025 from 20.192.21.51
[ec2-user@ip-10-0-10-91 ~]$ sudo nano /home/ec2-user/health_check.sh
[ec2-user@ip-10-0-10-91 ~]$ sudo chmod +x /home/ec2-user/health_check.sh
[ec2-user@ip-10-0-10-91 ~]$ cat  nano /home/ec2-user/health_check.sh
cat: nano: No such file or directory
#!/bin/bash

# List of backend servers
BACKENDS=("10.0.10.246" "10.0.10.247")
LOG_FILE="/home/ec2-user/backend_health.log"

while true; do
    DATE=$(date '+%Y-%m-%d %H:%M:%S')

    for SERVER in "${BACKENDS[@]}"; do
        # Check HTTP response
        STATUS=$(curl -s -o /dev/null -w "%{http_code}" http://$SERVER)

        if [ "$STATUS" -eq 200 ]; then
            echo "$DATE - $SERVER is UP (HTTP $STATUS)" | tee -a $LOG_FILE
        else
            echo "$DATE - $SERVER is DOWN (HTTP $STATUS)" | tee -a $LOG_FILE

            # Optional: restart Apache if server is local
            if [ "$SERVER" == "127.0.0.1" ] || [ "$SERVER" == "$(hostname -I | awk '{print $1}')" ]; then
                echo "$DATE - Restarting Apache on $SERVER" | tee -a $LOG_FILE
                sudo systemctl restart httpd
            fi
        fi
    done

    # Wait 30 seconds
    sleep 30
done
[ec2-user@ip-10-0-10-91 ~]$
```

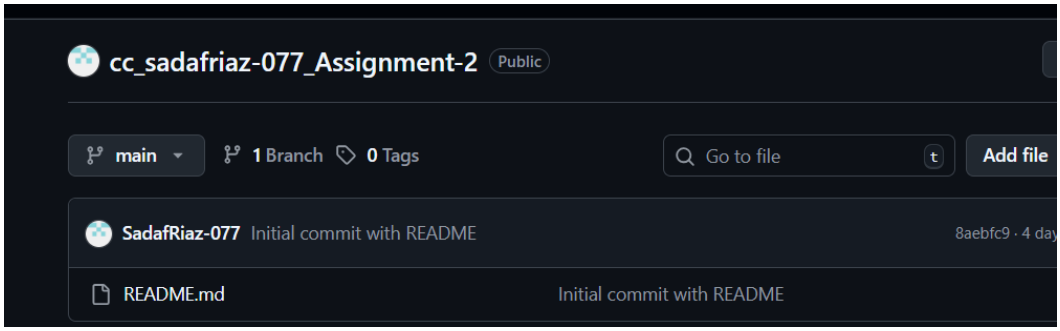
assignment_bonus_health_check_logs.png

```
[ec2-user@ip-10-0-10-91 ~]$ cat /home/ec2-user/backend_health.log
2025-12-30 08:42:05 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:42:05 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:42:38 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:42:38 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:43:11 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:43:11 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:43:44 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:43:44 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:44:17 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:44:17 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:44:51 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:44:51 - 10.0.10.247 is DOWN (HTTP 000)
2025-12-30 08:45:24 - 10.0.10.246 is UP (HTTP 200)
2025-12-30 08:45:24 - 10.0.10.247 is DOWN (HTTP 000)
[ec2-user@ip-10-0-10-91 ~]$
```

Part 6: Documentation & Cleanup

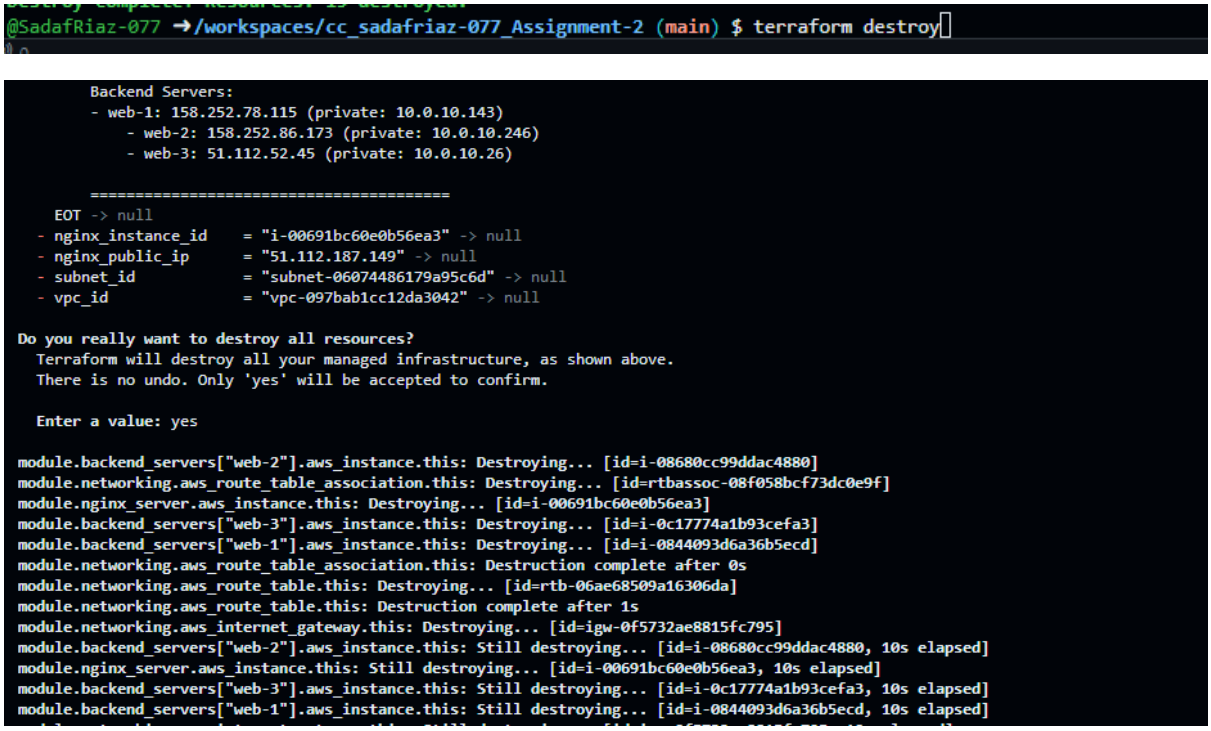
6.1 README Documentation

assignment_part6_readme_file_.png

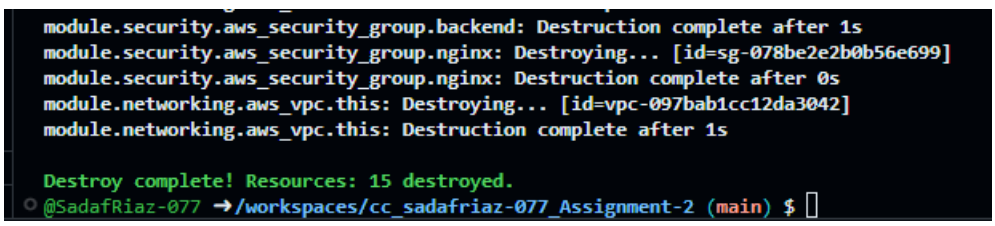


6.2 Infrastructure Cleanup

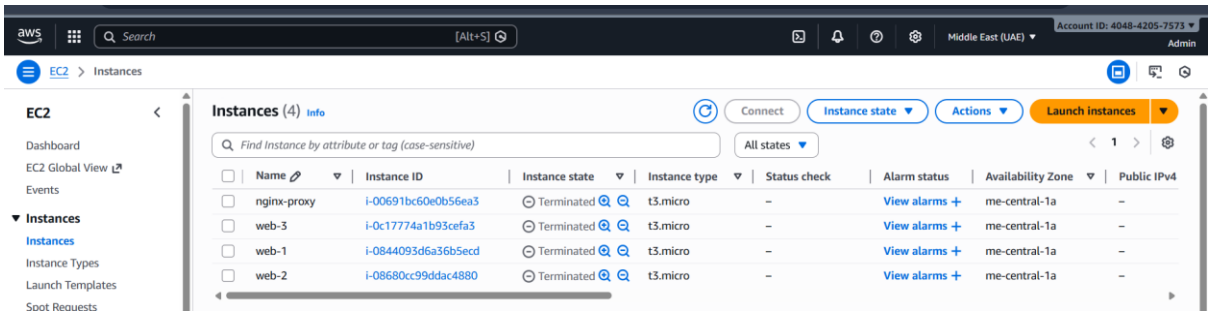
assignment_part6_terraform_destroy_prompt.png



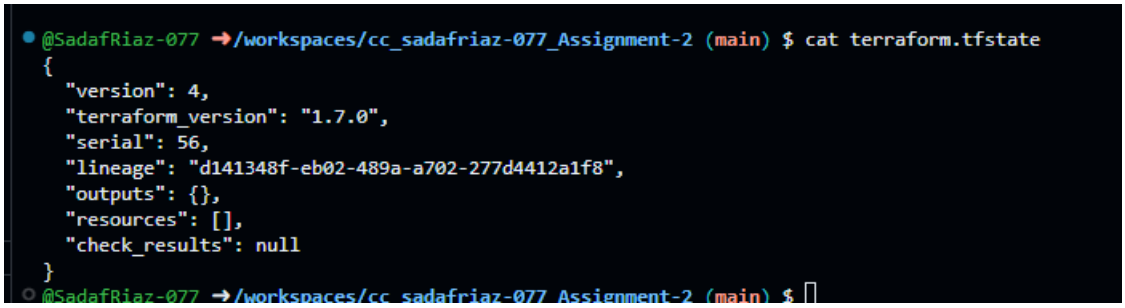
assignment_part6_terraform_destroy_completion.png



assignment_part6_aws_resources_terminated.png



assignment_part6_empty_terraform_state.png



4. Testing Results

This section documents the testing performed to validate the functionality, performance, and reliability of the deployed infrastructure.

4.1 Load Balancing Tests

Load balancing was tested by refreshing the application URL multiple times in the browser. Each refresh returned responses from different backend Apache servers, confirming that Nginx successfully distributed incoming traffic among the backend servers. This verified proper configuration of the upstream block and load balancing mechanism.

4.2 Cache Performance Tests

Nginx caching functionality was verified by analyzing access logs and response headers. The first request resulted in a cache **MISS**, while subsequent requests returned a cache **HIT**, demonstrating that caching was working correctly and improving response time.

4.3 High Availability Tests

To test high availability, Apache services on the primary backend servers were manually stopped. Traffic was automatically redirected to the backup server without any service interruption. Once the primary servers were restored, normal operation resumed. This confirmed the effectiveness of the backup server configuration.

4.4 Security Tests

Security testing included verifying HTTPS encryption, HTTP-to-HTTPS redirection, and security headers. The SSL certificate was successfully applied, and all HTTP traffic was redirected to HTTPS. Security group rules ensured that only required ports were accessible.

4.5 Performance Metrics

Performance was evaluated using response times, log analysis, and cache behavior. The use of caching and load balancing reduced server load and improved response efficiency, demonstrating a scalable and optimized infrastructure.

5. Challenges & Solutions

5.1 Problems Encountered

Several challenges were faced during the assignment, including Terraform configuration errors, Nginx misconfigurations, backend connectivity issues, and SSL certificate warnings in the browser.

5.2 Solutions Implemented

These issues were resolved by carefully debugging Terraform modules, correcting resource references, validating Nginx configurations, and reviewing AWS security group rules. SSL warnings were expected due to the use of a self-signed certificate.

5.3 Lessons Learned

This assignment improved understanding of Infrastructure as Code, modular Terraform design, Nginx load balancing, and troubleshooting cloud-based systems. It also highlighted the importance of automation and testing in cloud deployments.

6. Conclusion

This assignment successfully demonstrated the deployment of a secure, scalable, and highly available multi-tier web infrastructure using AWS and Terraform. Key objectives such as load balancing, caching, security, automation, and failover were achieved.

Through this project, practical skills were developed in Terraform, AWS networking, Linux server management, and Nginx configuration. Future improvements could include using managed SSL certificates, auto-scaling groups, and monitoring tools such as CloudWatch for enhanced observability.