



## **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](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.

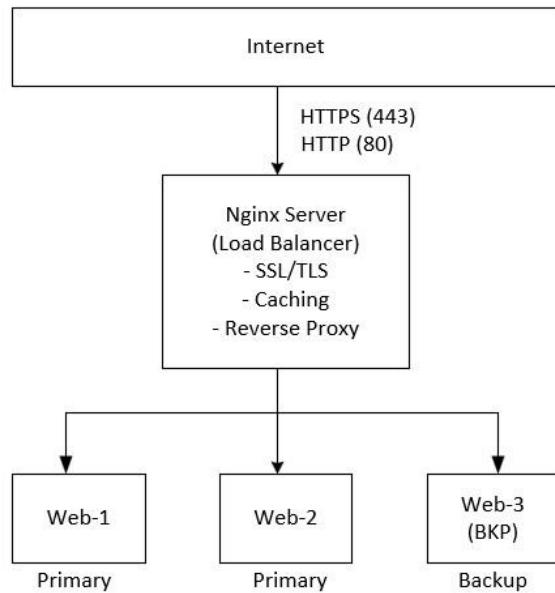
Server configuration was automated using shell scripts. The Apache backend servers display system and network information using EC2 metadata, whereas the Nginx server manages HTTPS connections, caching, request forwarding, and failover handling. The infrastructure was thoroughly tested to verify load 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
    └── local.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) $

```

Security Groups (5) <a href="#">Info</a>					
<input type="text"/> Find security groups by attribute or tag		<a href="#">Actions</a> <a href="#">Export security groups to CSV</a>		<a href="#">Create security group</a>	
<input type="checkbox"/>	Name	Security group ID	Security group name	VPC ID	Description
<input type="checkbox"/>	prod-backend-sg	<a href="#">sg-03c41af7e2c4457bf</a>	prod-backend-sg	<a href="#">vpc-00963b4a5bcf35790</a>	Security group for b
<input type="checkbox"/>	-	<a href="#">sg-07faa232dd28e816b</a>	default	<a href="#">vpc-0ed011693d93d59bc</a>	default VPC security
<input type="checkbox"/>	prod-nginx-sg	<a href="#">sg-04fec242d7fcaa167</a>	prod-nginx-sg	<a href="#">vpc-00963b4a5bcf35790</a>	Security group for N

## 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

```
yum update -y
@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)

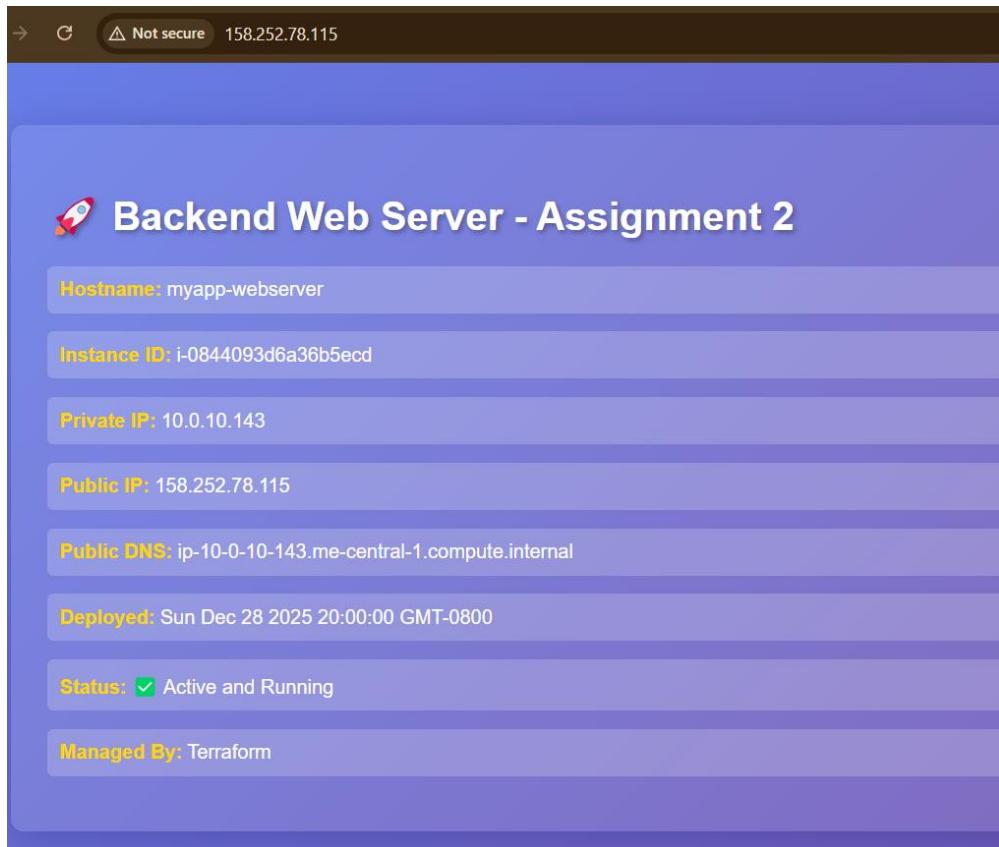
# Set hostname
hostnamectl set-hostname myapp-webserver

# 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!"
```

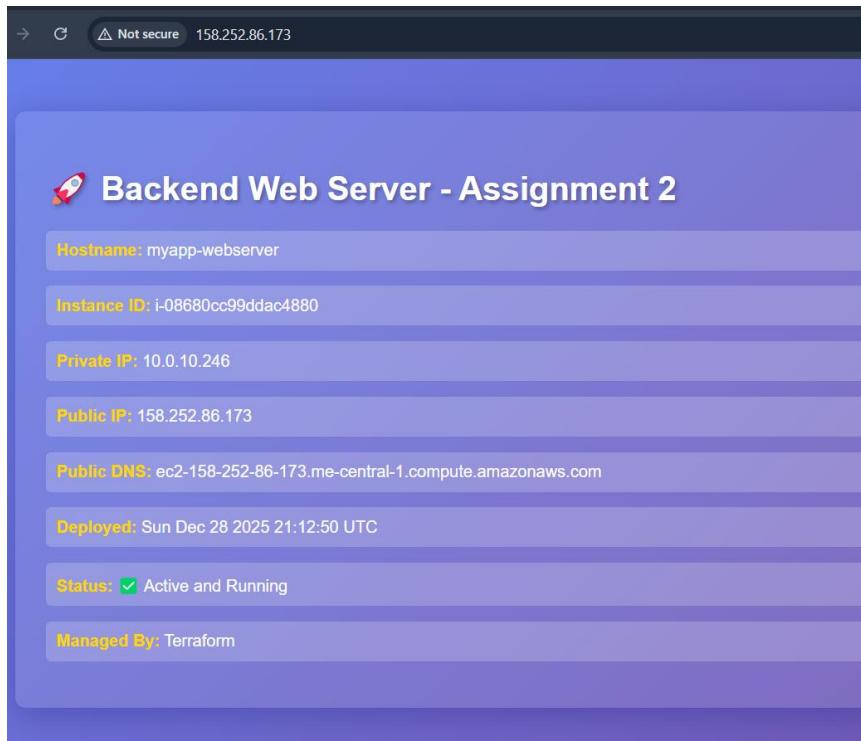
- Screenshot: assignment\_part3\_backend\_webpage.png (browser showing backend server page)



## Web 2

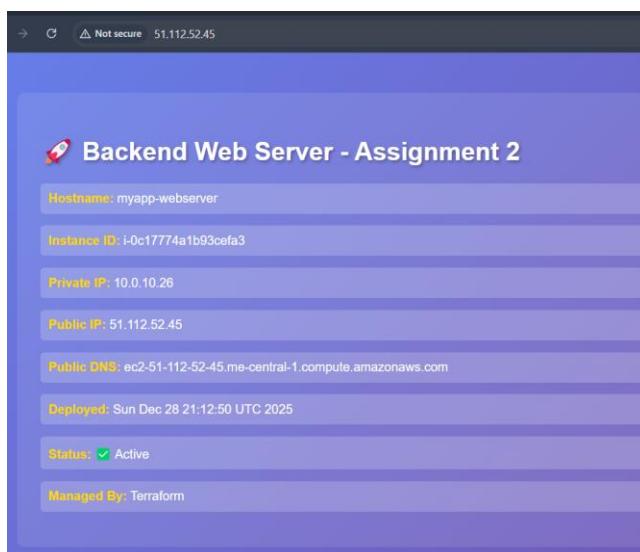
```
@SadafRiaz-077 ~/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i ~/.ssh/prod-2-key.pem ec2-user@158.252.86.173
Warning: Identity file /home/codespace/.ssh/prod-2-key.pem not accessible: No such file or directory.
Enter passphrase for key '/home/codespace/.ssh/id_ed25519':
,
  #
~\_ ####_
~~ \#####\
~~\###|
~~ \|/   Amazon Linux 2023 (ECS Optimized)
~~` .->
~~\_
~~\_
  /_
  /_
  /m/'
```

For documentation, visit <http://aws.amazon.com/documentation/ecs>  
Last login: Sun Dec 28 21:23:45 2025 from 4.240.39.197



## Web 3

```
@SadaRiaz-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+eRDkWAVMGFIKsK+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)  
~~ V.' ' ->  
~~ /  
~~ .- /  
~~ / /  
~/m/'  
For documentation, visit http://aws.amazon.com/documentation/ecs
```



## 3.2 Nginx Server Setup Script

### Screenshot: assignment\_part3\_nginx\_script.png

```
@SadaRiaz-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_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

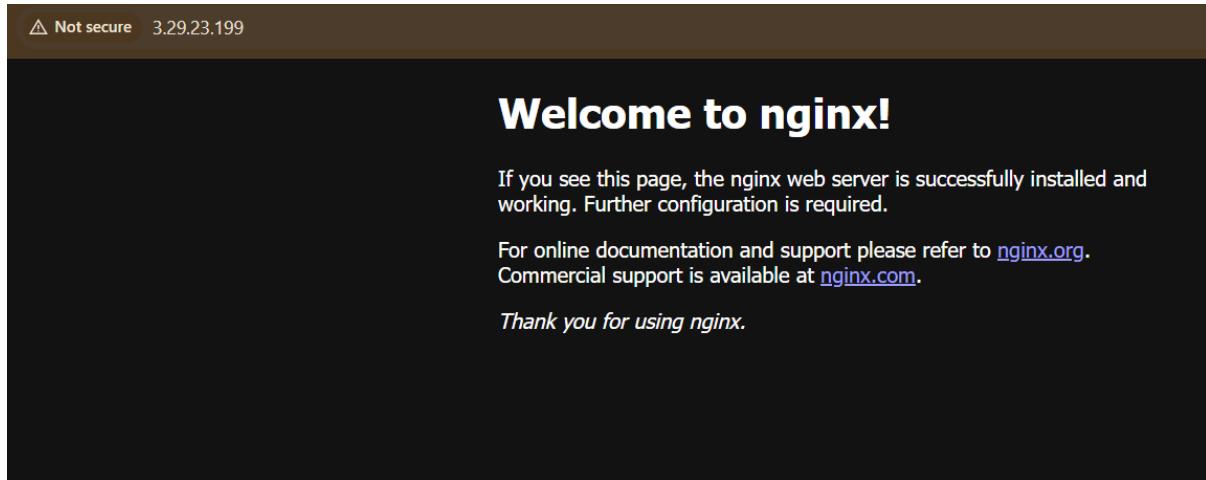
mkdir -p /var/cache/nginx
chown -R nginx:nginx /var/cache/nginx
nginx -t && systemctl restart nginx

echo "Nginx setup completed successfully!"
echo "Remember to update backend server IPs in /etc/nginx/nginx.conf"
@SadaRiaz-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)

```
@SadaRiaz-077 ~>/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i /workspaces/cc_sadafriaz-077_Assignment-2/ [ec2-user@app-webserver scripts]$ sudo systemctl stop httpd
[ec2-user@app-webserver scripts]$ sudo systemctl disable httpd
Removed "/etc/systemd/system/multi-user.target.wants/httpd.service".
[ec2-user@app-webserver scripts]$ sudo systemctl start nginx
[ec2-user@app-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@app-webserver scripts]$ sudo systemctl enable nginx
[ec2-user@app-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@app-webserver scripts]$ ^C
[ec2-user@app-webserver scripts]$ []
```



## Part 4: Infrastructure Deployment

### 4.1 Initial Deployment

assignment\_part4\_ssh\_key\_generation.png

```
● @SadafRiaz-077 ➔ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh-keygen -t ed25519 -C "my-terraform-key" -f ~/.ssh/id_ed25519
Generating public/private ed25519 key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/codespace/.ssh/id_ed25519
Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:dgrjKVtNwFGXNg0Jn1JZiBPwMy70QkovtZIwErPnek my-terraform-key
The key's randomart image is:
+--[ED25519 256]--+
|.. .++o+.
|oo.oB*oo .
|=o.++X.. =
|o oo*o+ +
|.. + .o+ S .
| + ..E B o
|.. . + o
|   +
|   .
+---[SHA256]---+
○ @SadafRiaz-077 ➔ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ 
```

assignment\_part4\_terraform\_initialization.png

```
● @SadafRiaz-077 ➔ /workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/http from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0
- Using previously-installed hashicorp/http v3.5.0

Terraform has been successfully initialized!

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

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

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
=====
REDEPLOYMENT SUCCESSFUL!
```

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

The screenshot shows the AWS VPC dashboard with the title "Your VPCs". It lists three VPCs:

Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR
dev-vpc	vpc-00963b4a5bcf35790	Available	-	-	Off	10.0.10.0/24
prod-vpc	vpc-097bab1cc12da3042	Available	-	-	Off	10.0.10.0/24
-	vpc-0ed011693d93d59bc	Available	-	-	Off	172.31.32.0/2

assignment\_part4\_aws\_subnet\_verification.png

The screenshot shows the AWS Subnets dashboard with the title "Subnets (5) Info". It lists five subnets across three VPCs:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
prod-subnet	subnet-06074486179a95c6d	Available	vpc-097bab1cc12da3042   prod...	Off	10.0.10.0/24
-	subnet-0414134a57a42925	Available	vpc-0ed011693d93d59bc	Off	172.31.0.0/20
-	subnet-023cec070fa4154f2	Available	vpc-0ed011693d93d59bc	Off	172.31.16.0/2
dev-subnet-1	subnet-0f5e62a2300537129	Available	vpc-00963b4a5bcf35790   dev...	Off	10.0.10.0/24
-	subnet-0467c5756668f293f	Available	vpc-0ed011693d93d59bc	Off	172.31.32.0/2

assignment\_part4\_aws\_security\_groups\_verification.png

The screenshot shows the AWS Security Groups dashboard with the title "Security Groups (5) Info". It lists six security groups across three VPCs:

Name	Security group ID	Security group name	VPC ID	Description
-	sg-07faa232dd28e816b	default	vpc-0ed011693d93d59bc	default VPC security
prod-nginx-sg	sg-078be2e2b0b56e699	prod-nginx-sg	vpc-097bab1cc12da3042	Security group for N
dev-sg	sg-001ffa9323d334dc0	default	vpc-00963b4a5bcf35790	default VPC security
prod-backend-sg	sg-0e4edaa6a86855a15	prod-backend-sg	vpc-097bab1cc12da3042	Security group for b
-	sg-0e728b20836e2550a	default	vpc-097bab1cc12da3042	default VPC security

assignment\_part4\_aws\_ec2\_instances\_verification.png

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
nginx-proxy	i-00691bc60e0b56ea3	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	me-central-1a
web-3	i-0c17774a1b93cefa3	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	me-central-1a
web-1	i-0844093d6a36b5ec0d	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	me-central-1a
web-2	i-08680cc99ddac4880	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	me-central-1a

## Part 5: Nginx Configuration & Testing

### 5.1 Update Nginx Backend Configuration

assignment\_part5\_ssh\_into\_nginx.png

```
1.240.39.196@sadafRiaz-077 ~]$/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ ssh -i ~/.ssh/id_ed25519 ec2-user@51.112.187.149
The authenticity of host '51.112.187.149 (51.112.187.149)' can't be established.
ED25519 key fingerprint is SHA256:Eye+mcBIlyuCSjg/5vXBVwAWEpiKwqBdnUj9WNZFqQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '51.112.187.149' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/codespace/.ssh/id_ed25519':
      #
      _#####
      \#####
      \###|
      \#/ __ Amazon Linux 2023 (ECS Optimized)
      V~' .->
      /_
      /_/
      /m/'
```

For documentation, visit <http://aws.amazon.com/documentation/ecs>

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

assignment\_part5\_updated\_nginx\_configuration.png

```
[ec2-user@ip-10-0-10-91 ~]$ sudo nginx -t
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" directive instead in /etc/nginx/nginx.conf:40
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-91 ~]$ ]
```

assignment\_part5\_nginx\_configuration\_test.png

```
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=my_cache:10m max_size=
```

```
upstream backend_servers {
    server 10.0.10.143;
    server 10.0.10.246;
    server 10.0.10.26;
}
```

```
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";
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
```

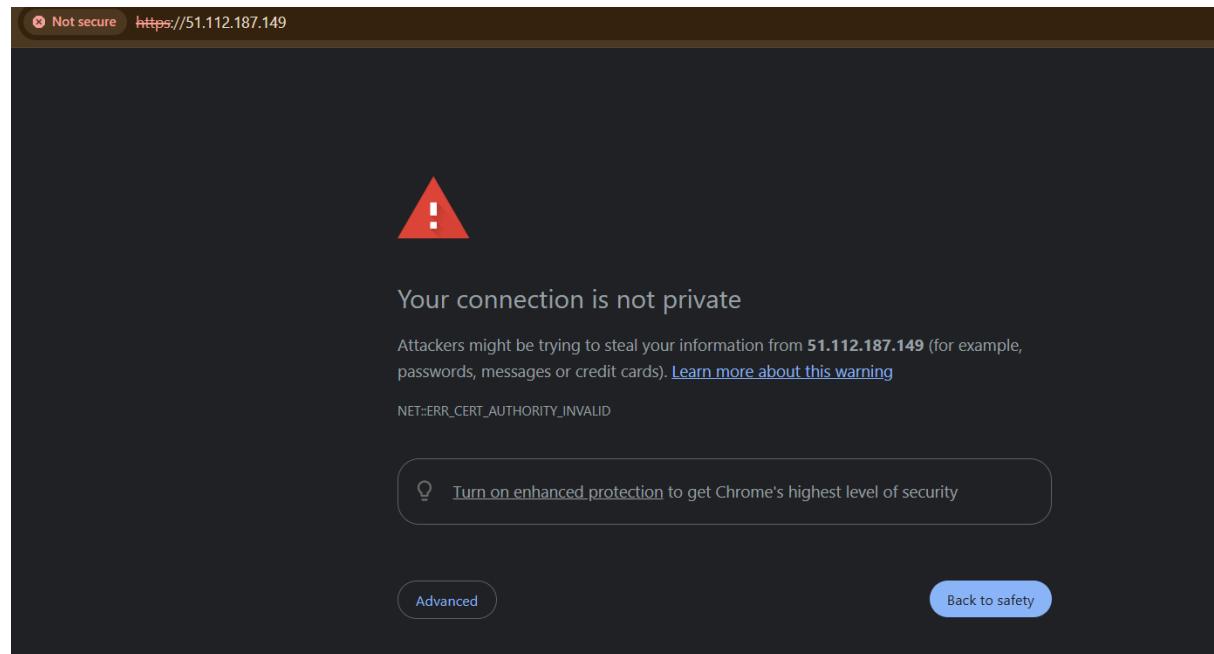
### assignment\_part5\_nginx\_restart.png

```
[ec2-user@ip-10-0-10-91 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-91 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
   Active: active (running) since Sun 2025-12-28 13:01:08 UTC; 21s ago
     Process: 285252 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 285253 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 285254 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 285255 (nginx)
      Tasks: 5 (limit: 1065)
        Memory: 4.4M
         CPU: 60ms
      CGroup: /system.slice/nginx.service
          ├─285255 "nginx: master process /usr/sbin/nginx"
          ├─285256 "nginx: worker process"
          ├─285257 "nginx: worker process"
          ├─285258 "nginx: cache manager process"
          └─285259 "nginx: cache loader process"

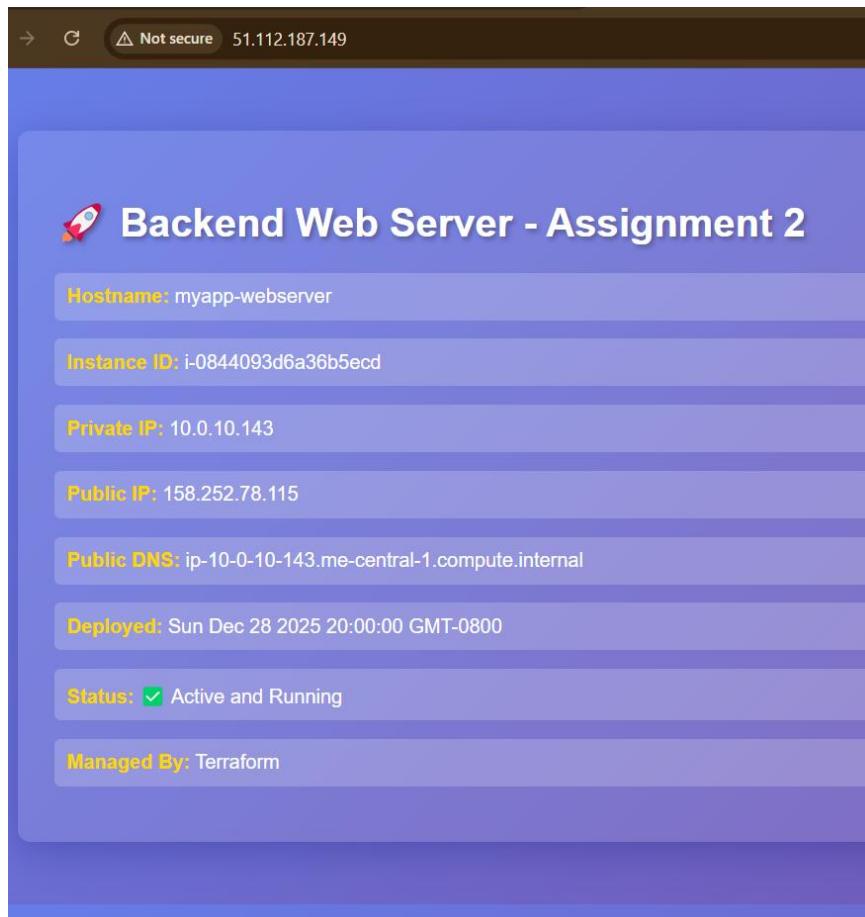
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal nginx[285253]: nginx: [warn] the "listen ... http2" directive is deprecated
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal nginx[285253]: nginx: the configuration file /etc/nginx/nginx.conf syntax is
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal nginx[285253]: nginx: configuration file /etc/nginx/nginx.conf test is succ
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal nginx[285254]: nginx: [warn] the "listen ... http2" directive is deprecated
Dec 28 13:01:08 ip-10-0-10-91.me-central-1.compute.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
```

## 5.2 Test Load Balancing

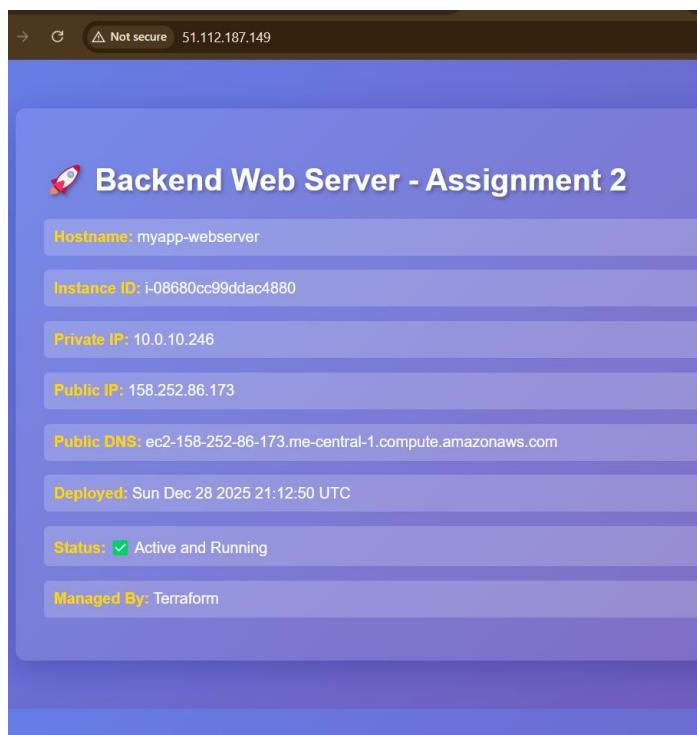
### assignment\_part5\_nginx\_restart.png



assignment\_part5\_web1\_response.png

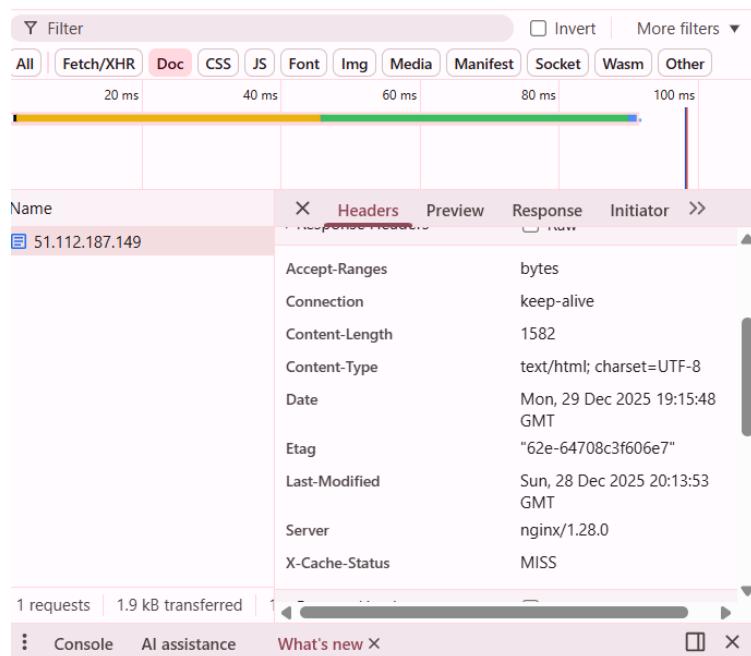


assignment\_part5\_web2\_response.png

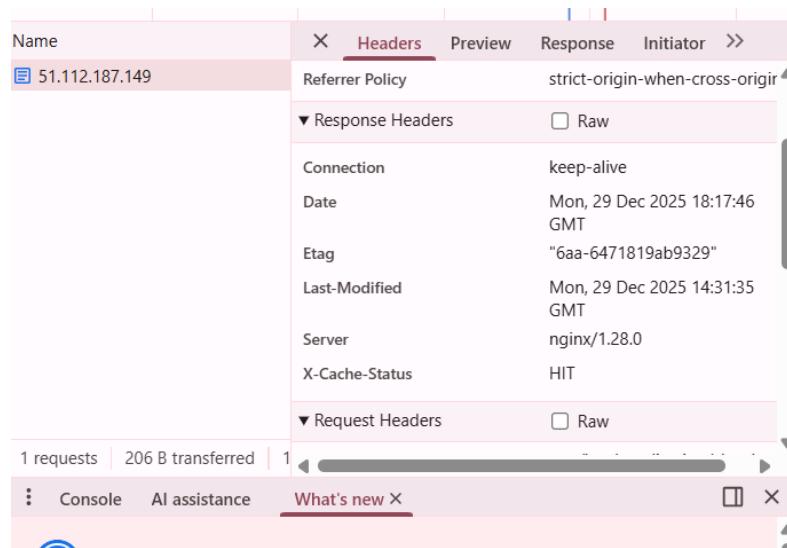


### 5.3 Test Cache Functionality

assignment\_part5\_cache\_miss\_verification.png



assignment\_part5\_cache\_hit\_verification.png



assignment\_part5\_nginx\_cache\_directory.png

```
[ec2-user@ip-10-0-10-91 ~]$ sudo systemctl reload nginx
[ec2-user@ip-10-0-10-91 ~]$ ls -la /var/cache/nginx/
total 0
drwxr-xr-x. 3 nginx nginx 15 Dec 29 19:15 .
drwxr-xr-x. 10 root root 107 Dec 28 14:35 ..
drwx----- 3 nginx nginx 16 Dec 29 19:15 e
[ec2-user@ip-10-0-10-91 ~]$ [ ]
```

assignment\_part5\_access\_log\_cache\_status.png

## 5.4 Test High Availability (Backup Server)

assignment\_part5\_web1\_service\_stopped.png

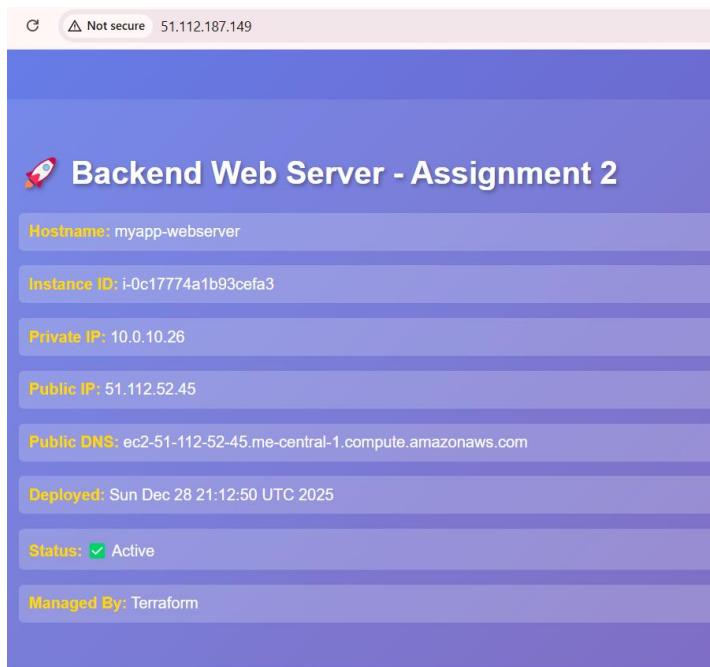
```
○ 20.192.21.55$SafdarRiaz-077 → /workspaces/cc_safdarfiaz-077_Assignment-2 (main) $ ssh -i ~/.ssh/id_ed25519 ec2-user@158.252.78.115
Enter passphrase for key '/home/codespace/.ssh/id_ed25519':
,
#
~\_ ####_
~~ \ #####
~~ \###|
~~ \|/ ____ Amazon Linux 2023 (ECS Optimized)
~~ \v\-'-->
~~ /_
~~ /_/
~~ /_/
/`m'

For documentation, visit http://aws.amazon.com/documentation/ecs
Last login: Sun Dec 28 20:08:41 2025 from 4.240.39.197
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: inactive (dead) since Mon 2025-12-29 19:56:23 UTC; 13s ago
     Duration: 23h 41min 31.258s
     Docs: man:httpd.service(8)
   Process: 671002 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
 Main PID: 671002 (code=exited, status=0/SUCCESS)
   Status: "Total requests: 376; Idle/Busy workers 100/0;Requests/sec: 0.00441; Bytes served/sec: 4 B/sec"
    CPU: 1min 27.610s

Dec 28 20:14:51 myapp-webserver systemd[1]: Starting httpd.service - The Apache HTTP Server...
Dec 28 20:14:51 myapp-webserver httpd[671002]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using fe80::...
Dec 28 20:14:51 myapp-webserver systemd[1]: Started httpd.service - The Apache HTTP Server.
Dec 28 20:14:51 myapp-webserver httpd[671002]: Server configured, listening on: port 80
Dec 29 19:56:22 myapp-webserver systemd[1]: Stopping httpd.service - The Apache HTTP Server...
Dec 29 19:56:23 myapp-webserver systemd[1]: httpd.service: Deactivated successfully.
Dec 29 19:56:23 myapp-webserver systemd[1]: Stopped httpd.service - The Apache HTTP Server.
Dec 29 19:56:23 myapp-webserver systemd[1]: httpd.service: Consumed 1min 27.610s CPU time.
Lines 1-18/18 (END)
```

assignment\_part5\_web2\_service\_stopped.png

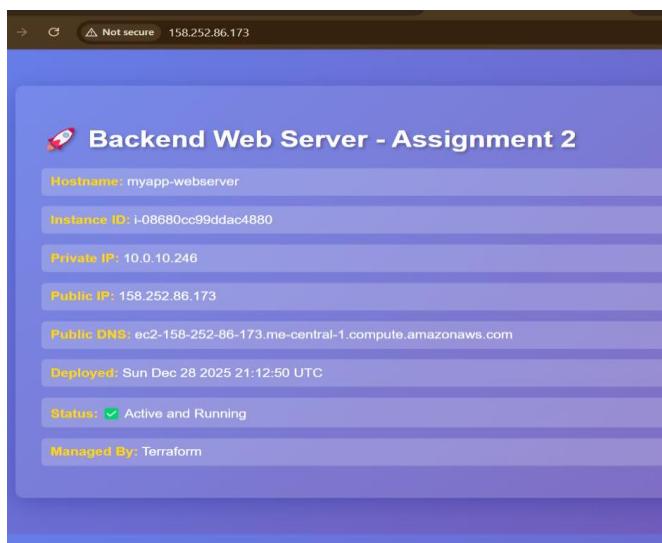
### assignment\_part5\_backup\_server\_activated.png



### assignment\_part5\_nginx\_error\_log.png

```
[ec2-user@ip-10-0-10-91 ~]$ sudo tail -f /var/log/nginx/error.log
2025/12/29 19:00:37 [warn] 1878941#1878941: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:00:37 [warn] 1878942#1878942: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:04:24 [warn] 1882424#1882424: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:04:24 [warn] 1882425#1882425: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:06:10 [warn] 1883952#1883952: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:06:10 [notice] 1883952#1883952: signal process started
2025/12/29 19:06:10 [warn] 1882426#1882426: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:14:57 [warn] 1891925#1891925: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
2025/12/29 19:14:57 [notice] 1891925#1891925: signal process started
2025/12/29 19:14:57 [warn] 1882426#1882426: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
```

### assignment\_part5\_services\_restored.png



## 5.5 Security & Performance Analysis

assignment\_part5\_ssl\_certificate\_details.png

```
@SadafRiaz-077 ~$ 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:
                1f:56:9e:fe:0a:ad:47:8a:72:9f:ee:0c:f7:ec:84:
                d9:77:f9:fe:7:7e: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:ob: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:f:c:e8:4c:d4:
                fa:30:24:75:d8:a5:6a:7c:8d:50:88:d1:d9:f9:72:
                6d:51:e1:a8:02:50: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:c1:22:94:20:2e:9a:
    f8:f9:0d:57:cda:6:9d: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:d4:44:72:45:00:28:
    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

```
[ec2-user@ip-10-0-10-91 ~]$ sudo tail -50 /var/log/nginx/error.log
2025/12/29 18:01:16 [emerg] 1826388#1826388: "proxy_cache" zone "my_cache" is unknown in /etc/nginx/nginx.conf:102
2025/12/29 18:15:30 [warn] 1839080#1839080: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
2025/12/29 18:15:48 [warn] 1839320#1839320: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
2025/12/29 18:15:48 [notice] 1839320#1839320: signal process started
2025/12/29 18:15:48 [notice] 1731537#1731537: signal 1 (SIGHUP) received from 1839320, reconfiguring
2025/12/29 18:15:48 [notice] 1731537#1731537: reconfiguring
2025/12/29 18:15:48 [warn] 1731537#1731537: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
2025/12/29 18:15:48 [notice] 1754682#1754682: gracefully shutting down
2025/12/29 18:15:48 [notice] 1754681#1754681: gracefully shutting down
2025/12/29 18:15:48 [notice] 1754682#1754682: exiting
2025/12/29 18:15:48 [notice] 1754681#1754681: exiting
2025/12/29 18:15:48 [notice] 1754681#1754681: exit
2025/12/29 18:15:48 [notice] 1754682#1754682: exit
2025/12/29 18:15:48 [notice] 1754683#1754683: exiting
2025/12/29 19:00:37 [warn] 1878929#1878929: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
2025/12/29 19:00:37 [notice] 1878929#1878929: signal process started
2025/12/29 19:00:37 [warn] 1731537#1731537: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
2025/12/29 19:00:37 [warn] 1731537#1731537: could not build optimal types_hash, you should increase either types_hash_max_size: 1024 or types_hash_bucket_size: ucket_size
```

## assignment\_part5\_access\_log\_analysis.png

```
[ec2-user@ip-10-0-10-91 ~]$ sudo tail -50 /var/log/nginx/access.log
119.157.93.14 - - [29/Dec/2025:18:52:38 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:18:53:42 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:18:53:52 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:18:54:04 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:18:57:58 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:16 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:17 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:18 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:26 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:28 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:01:47 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:02:19 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:07:16 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:10:44 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:11:19 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
119.157.93.14 - - [29/Dec/2025:19:15:48 +0000] "GET / HTTP/1.1" 200 1582 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
91.224.92.109 - - [29/Dec/2025:19:48:13 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.5735.199 Safari/537.36" 
```

## 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 $LOG_FILE
                sudo systemctl restart httpd
            fi
        fi
    done
    done
    # Wait 30 seconds
    sleep 30
done
[ec2-user@ip-10-0-10-91 ~]$ []
```

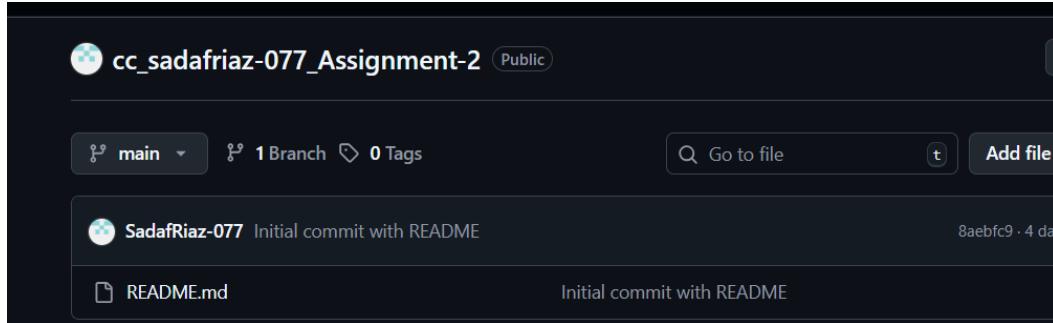
assignment\_bonus\_health\_check\_logs.png

```
[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

```
destroy completed! Resources: 15 destroyed.
@SadafRiaz-077 →/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ terraform destroy[]

Backend Servers:
- web-1: 158.252.78.115 (private: 10.0.10.143)
  - web-2: 158.252.86.173 (private: 10.0.10.246)
  - web-3: 51.112.52.45 (private: 10.0.10.26)

=====
EOT -> null
- nginx_instance_id      = "i-00691bc60e0b56ea3" -> null
- nginx_public_ip        = "51.112.187.149" -> null
- subnet_id               = "subnet-06074486179a95c6d" -> null
- vpc_id                  = "vpc-097bab1cc12da3042" -> null

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

Enter a value: yes

module.backend_servers["web-2"].aws_instance.this: Destroying... [id=i-08680cc99ddac4880]
module.networking.aws_route_table_association.this: Destroying... [id=rtbassoc-08f058bcf73dc0e9f]
module.nginx_server.aws_instance.this: Destroying... [id=i-00691bc60e0b56ea3]
module.backend_servers["web-3"].aws_instance.this: Destroying... [id=i-0c17774a1b93cefa3]
module.backend_servers["web-1"].aws_instance.this: Destroying... [id=i-0844093d6a36b5ecd]
module.networking.aws_route_table_association.this: Destruction complete after 0s
module.networking.aws_route_table.this: Destroying... [id=rtb-06ae68509a16306da]
module.networking.aws_route_table.this: Destruction complete after 1s
module.networking.aws_internet_gateway.this: Destroying... [id=igw-0f5732ae8815fc795]
module.backend_servers["web-2"].aws_instance.this: Still destroying... [id=i-08680cc99ddac4880, 10s elapsed]
module.nginx_server.aws_instance.this: Still destroying... [id=i-00691bc60e0b56ea3, 10s elapsed]
module.backend_servers["web-3"].aws_instance.this: Still destroying... [id=i-0c17774a1b93cefa3, 10s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still destroying... [id=i-0844093d6a36b5ecd, 10s elapsed]
```

assignment\_part6\_terraform\_destroy\_completion.png

```
module.security.aws_security_group.backend: Destruction complete after 1s
module.security.aws_security_group.nginx: Destroying... [id=sg-078be2e2b0b56e699]
module.security.aws_security_group.nginx: Destruction complete after 0s
module.networking.aws_vpc.this: Destroying... [id=vpc-097bab1cc12da3042]
module.networking.aws_vpc.this: Destruction complete after 1s

Destroy complete! Resources: 15 destroyed.
@SadafRiaz-077 →/workspaces/cc_sadafriaz-077_Assignment-2 (main) $ []
```

assignment\_part6\_aws\_resources\_terminated.png

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main area displays a table titled 'Instances (4) Info'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. All four instances listed are in the 'Terminated' state. The instance types are all 't3.micro'. The availability zones are 'me-central-1a' for all. The public IPv4 column shows '-' for all instances.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
nginx-proxy	i-00691bc60e0b56e3	Terminated	t3.micro	-	<a href="#">View alarms</a>	me-central-1a	-
web-3	i-0c17774a1b93cefa3	Terminated	t3.micro	-	<a href="#">View alarms</a>	me-central-1a	-
web-1	i-0844093d6a36b5ecd	Terminated	t3.micro	-	<a href="#">View alarms</a>	me-central-1a	-
web-2	i-08680cc99ddac4880	Terminated	t3.micro	-	<a href="#">View alarms</a>	me-central-1a	-

assignment\_part6\_empty\_terraform\_state.png

The screenshot shows a terminal window with the command `cat terraform.tfstate` run. The output is a JSON object representing an empty state file:

```
{ "version": 4, "terraform_version": "1.7.0", "serial": 56, "lineage": "d141348f-eb02-489a-a702-277d4412a1f8", "outputs": {}, "resources": [], "check_results": null }
```

## 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.