Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

CLOUD COMPUTING

BSE (V-B)

Submitted By:

Musfira Farooq

Roll No:

2023-BSE-045

ASSIGNMENT 02

Advanced Terraform & Nginx Multi-Tier Architecture

Table of Contents

1. Executive Summary

This assignment focuses on the design, deployment, testing, and cleanup of a production-ready multi-tier web infrastructure using Terraform and Nginx on Amazon Web Services (AWS). The main objective was to implement Infrastructure as Code (IaC) .

The infrastructure consists of a custom VPC, public subnets, security groups, EC2 instances running Apache web servers, and Nginx proxy acting as a load balancer. Terraform modules were used to ensure modularity, reusability, and clean project organization.

Key features include:

- Fully automated infrastructure provisioning using Terraform
- Modular Terraform design (networking, security, webserver)
- Nginx load balancing with SSL, caching, and security headers
- High availability testing with backend failover
- Proper infrastructure cleanup to avoid AWS charges

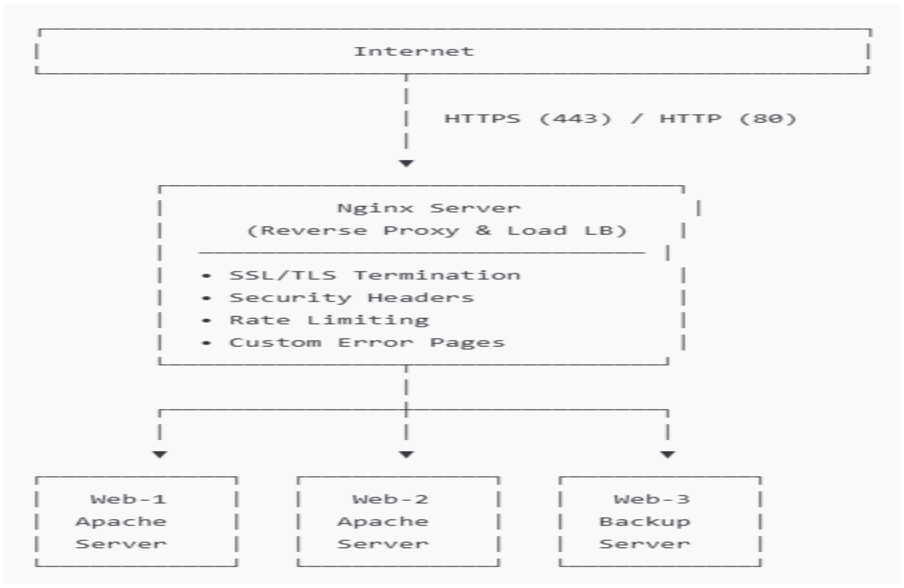
2. Architecture Design

2.1 Architecture Overview

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

- Client Layer: End users accessing the application via browser
- Load Balancer Layer: Nginx reverse proxy handling traffic
- Application Layer: Multiple Apache web servers

2.2 Architecture Diagram



2.3 Component Description

Nginx Server (Reverse Proxy & Load Balancer)

The Nginx server handles all incoming user requests. It manages HTTP/HTTPS traffic, performs SSL/TLS termination, balances load across backend servers, applies caching and rate limiting, and displays custom error pages when needed.

Web Servers (Web-1, Web-2, Web-3)

The web servers run Apache and serve application content. Web-1 and Web-2 handle normal traffic, while Web-3 works as a backup server. All servers are monitored using health checks.

2.4 Network Topology

The infrastructure uses a custom VPC with public subnets. An Internet Gateway enables internet access, route tables manage traffic flow, and security groups control network access.

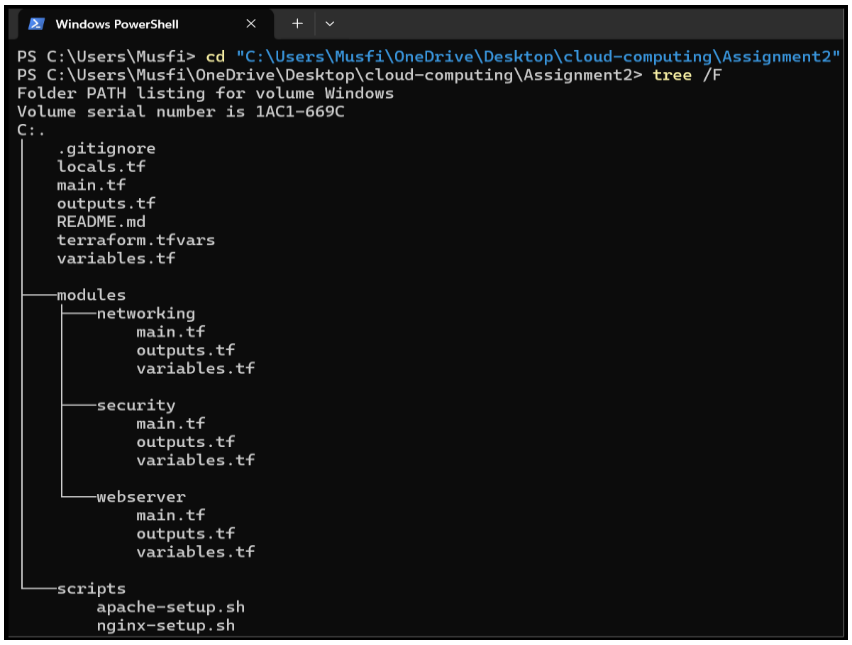
2.5 Security Design

Security groups restrict access to required ports only. SSH access is limited, backend servers are accessible only through Nginx, and sensitive files are protected using `.gitignore`.

3. Implementation Details

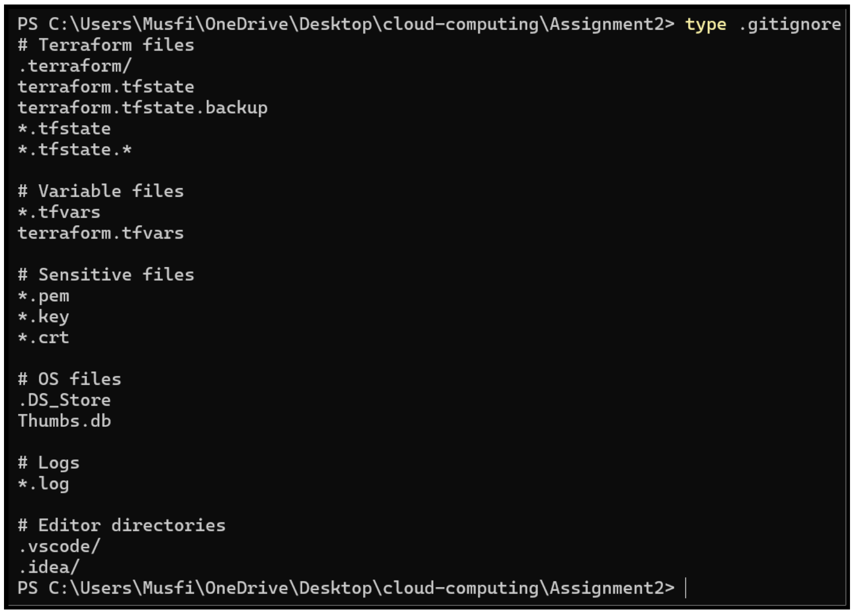
Part 1 : Infrastructure Setup

1.1-Project Structure:



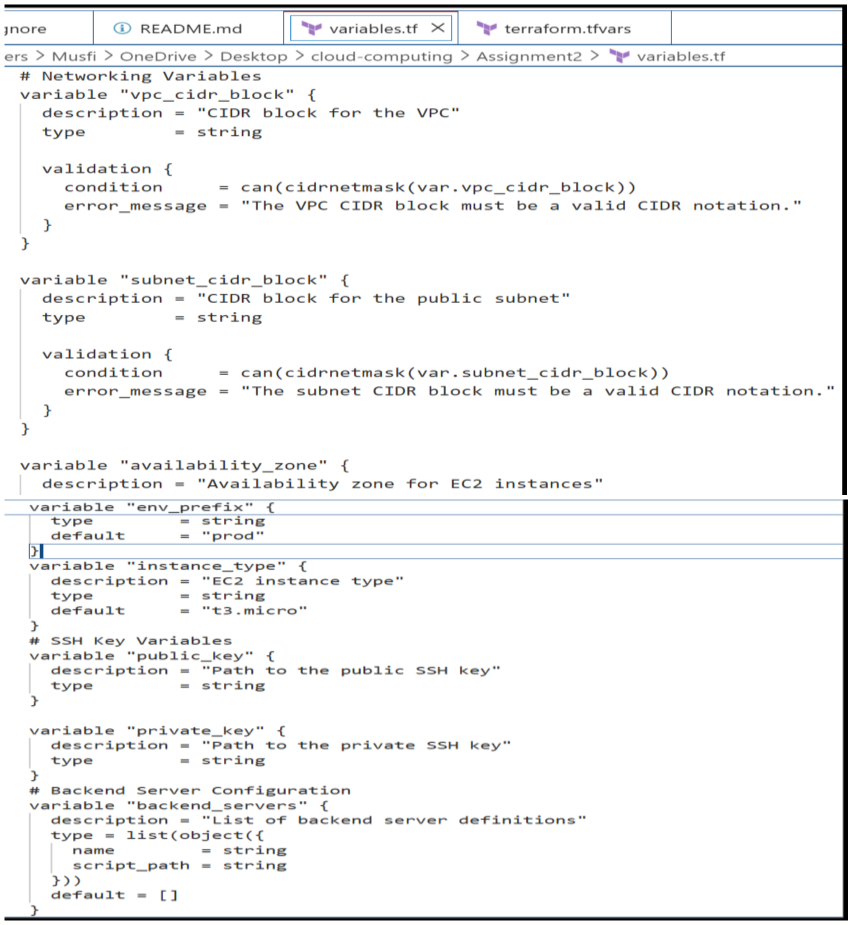
```
PS C:\Users\Musfi> cd "C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2"
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> tree /F
Folder PATH listing for volume Windows
Volume serial number is 1AC1-669C
C:.
│   .gitignore
│   locals.tf
│   main.tf
│   outputs.tf
│   README.md
│   terraform.tfvars
│   variables.tf
│
├───modules
│   ├───networking
│   │       main.tf
│   │       outputs.tf
│   │       variables.tf
│   │
│   ├───security
│   │       main.tf
│   │       outputs.tf
│   │       variables.tf
│   │
│   └───webserver
│           main.tf
│           outputs.tf
│           variables.tf
│
└───scripts
        apache-setup.sh
        nginx-setup.sh
```

- **content of .gitignore:**



```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> type .gitignore
# Terraform files
.terraform/
terraform.tfstate
terraform.tfstate.backup
*.tfstate
*.tfstate.*

# Variable files
*.tfvars
terraform.tfvars

# Sensitive files
*.pem
*.key
*.crt

# OS files
.DS_Store
Thumbs.db

# Logs
*.log

# Editor directories
.vscode/
.idea/
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2>
```

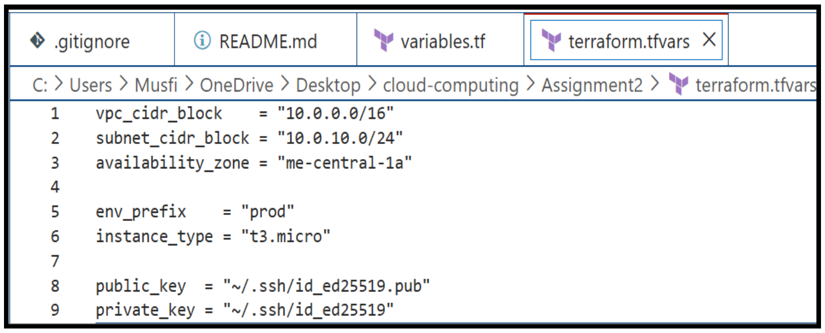
1.2-Variable Configuration:



ɪnore
README.md
variables.tf
terraform.tfvars
ers > Musfi > OneDrive > Desktop > cloud-computing > Assignment2 > variables.tf

```
# Networking Variables
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string

  validation {
    condition     = can(cidrnetmask(var.vpc_cidr_block))
    error_message = "The VPC CIDR block must be a valid CIDR notation."
  }
}

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

  validation {
    condition     = can(cidrnetmask(var.subnet_cidr_block))
    error_message = "The subnet CIDR block must be a valid CIDR notation."
  }
}

variable "availability_zone" {
  description = "Availability zone for EC2 instances"
variable "env_prefix" {
  type        = string
  default     = "prod"
}
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
}
# SSH Key Variables
variable "public_key" {
  description = "Path to the public SSH key"
  type        = string
}

variable "private_key" {
  description = "Path to the private SSH key"
  type        = string
}
# Backend Server Configuration
variable "backend_servers" {
  description = "List of backend server definitions"
  type = list(object({
    name        = string
    script_path = string
  }))
  default = []
}
```

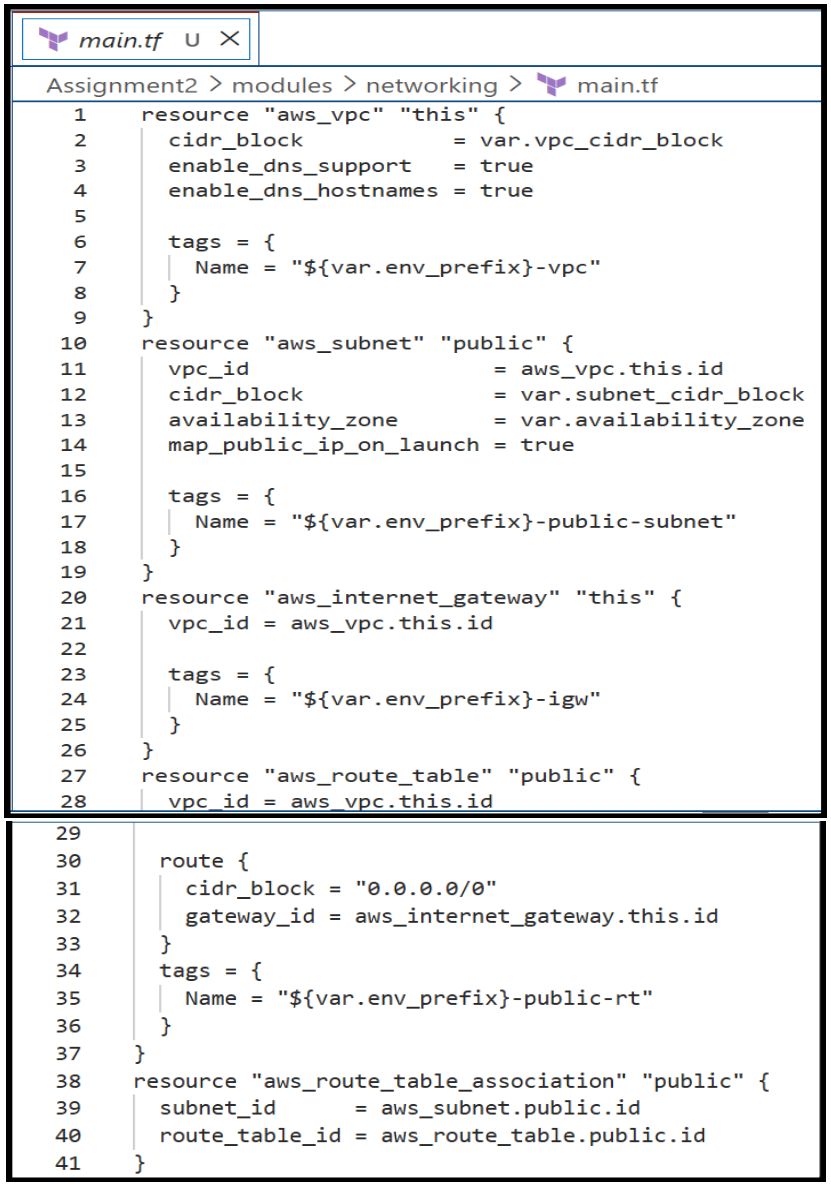
- **terraform_tfvars:**


.gitignore | README.md | variables.tf | terraform.tfvars

C: > Users > Musfi > OneDrive > Desktop > cloud-computing > Assignment2 > terraform.tfvars

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

env_prefix    = "prod"
instance_type = "t3.micro"

public_key  = "~/.ssh/id_ed25519.pub"
private_key = "~/.ssh/id_ed25519"
```

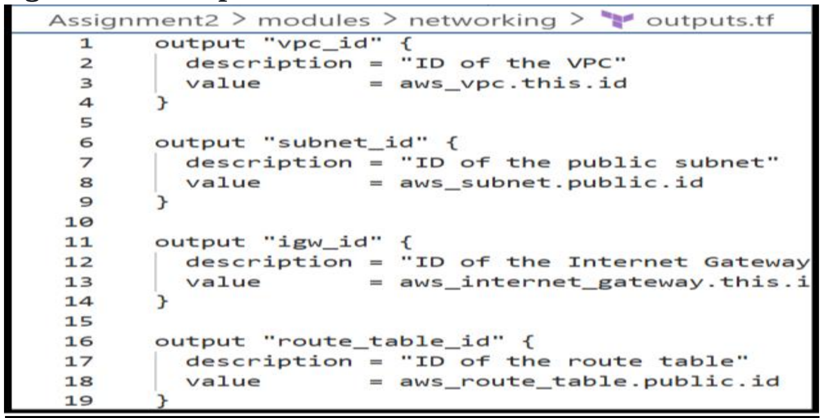
1.3 Networking Module:


main.tf U

Assignment2 > modules > networking > main.tf

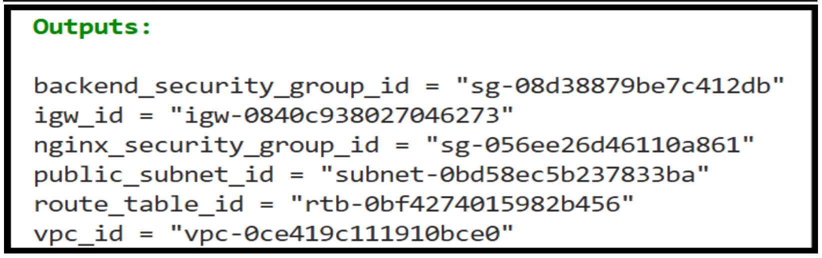
```
resource "aws_vpc" "this" {
  cidr_block           = var.vpc_cidr_block
  enable_dns_support   = true
  enable_dns_hostnames = true

  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}
resource "aws_subnet" "public" {
  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}-public-subnet"
  }
}
resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id

  tags = {
    Name = "${var.env_prefix}-igw"
  }
}
resource "aws_route_table" "public" {
  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}-public-rt"
  }
}
resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.public.id
  route_table_id = aws_route_table.public.id
}
```

- **networking_module_outputs:**



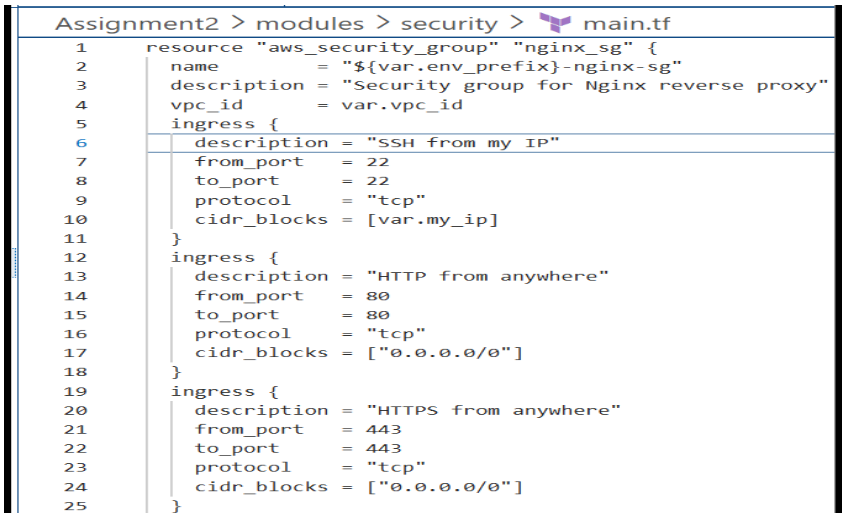
Assignment2 > modules > networking > outputs.tf

```
output "vpc_id" {
  description = "ID of the VPC"
  value       = aws_vpc.this.id
}

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

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

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



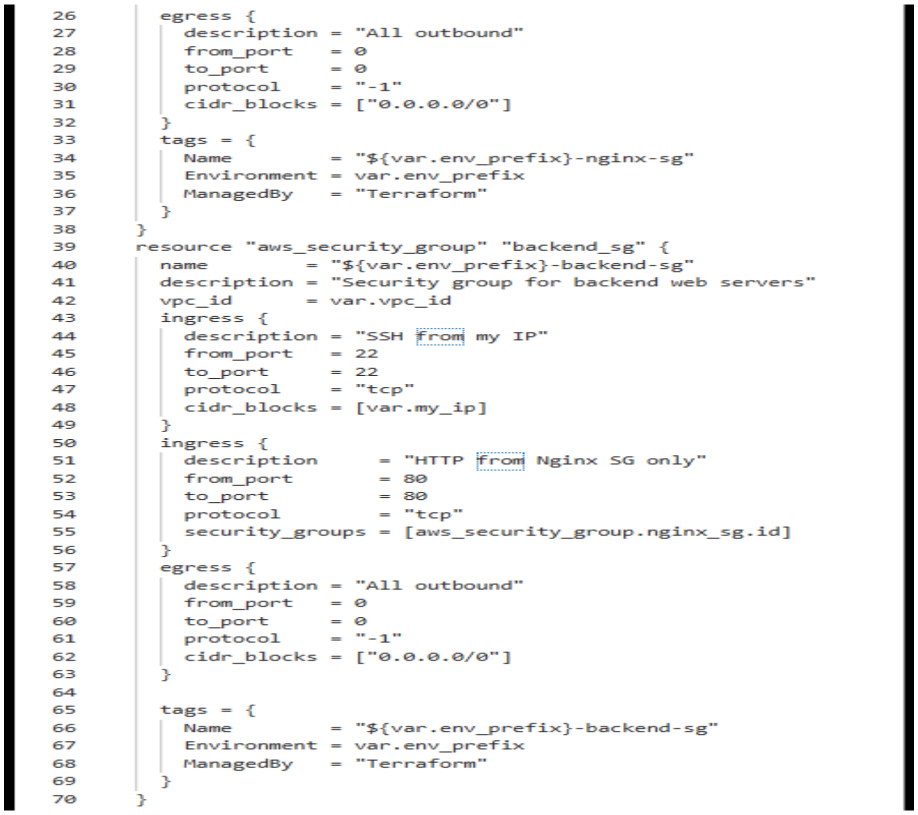
```
Outputs:

backend_security_group_id = "sg-08d38879be7c412db"
igw_id = "igw-0840c938027046273"
nginx_security_group_id = "sg-056ee26d46110a861"
public_subnet_id = "subnet-0bd58ec5b237833ba"
route_table_id = "rtb-0bf4274015982b456"
vpc_id = "vpc-0ce419c111910bce0"
```

1.4 Security Module:



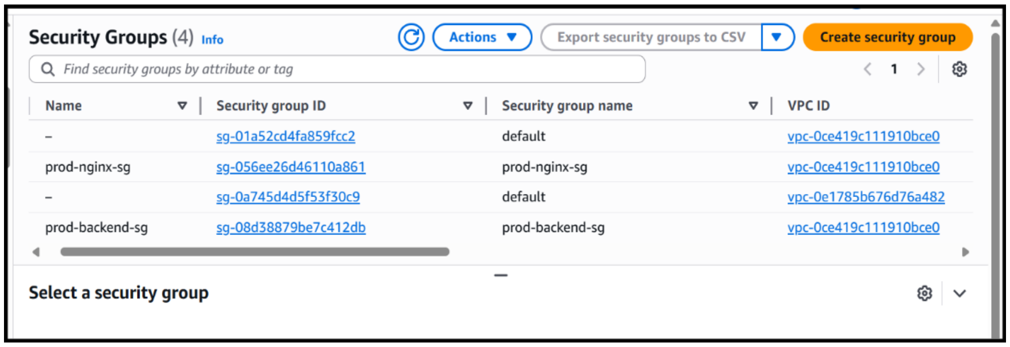
Assignment2 > modules > security > main.tf

```
resource "aws_security_group" "nginx_sg" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Security group for Nginx reverse proxy"
  vpc_id      = var.vpc_id
  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }
  ingress {
    description = "HTTP from anywhere"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    description = "HTTPS from anywhere"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```



```
  egress {
    description = "All outbound"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name        = "${var.env_prefix}-nginx-sg"
    Environment = var.env_prefix
    ManagedBy   = "Terraform"
  }
}
resource "aws_security_group" "backend_sg" {
  name        = "${var.env_prefix}-backend-sg"
  description = "Security group for backend web servers"
  vpc_id      = var.vpc_id
  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }
  ingress {
    description     = "HTTP from Nginx SG only"
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    security_groups = [aws_security_group.nginx_sg.id]
  }
  egress {
    description = "All outbound"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

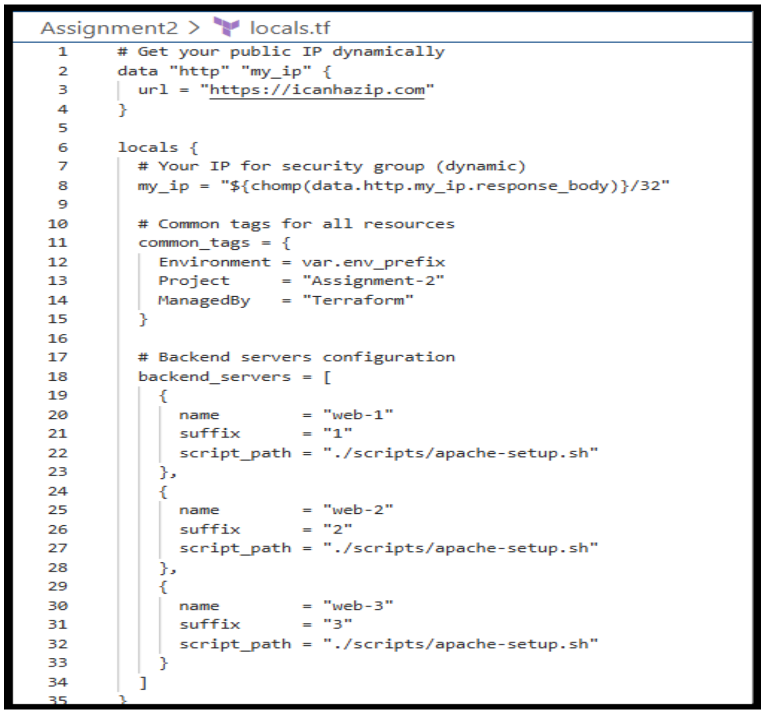
- **AWS Console:**


Security Groups (4) Info
Actions
Export security groups to CSV
Create security group
Find security groups by attribute or tag
1

Name	Security group ID	Security group name	VPC ID
–	sg-01a52cd4fa859fcc2	default	vpc-0ce419c111910bce0
prod-nginx-sg	sg-056ee26d46110a861	prod-nginx-sg	vpc-0ce419c111910bce0
–	sg-0a745d4d5f53f30c9	default	vpc-0e1785b676d76a482
prod-backend-sg	sg-08d38879be7c412db	prod-backend-sg	vpc-0ce419c111910bce0

Select a security group

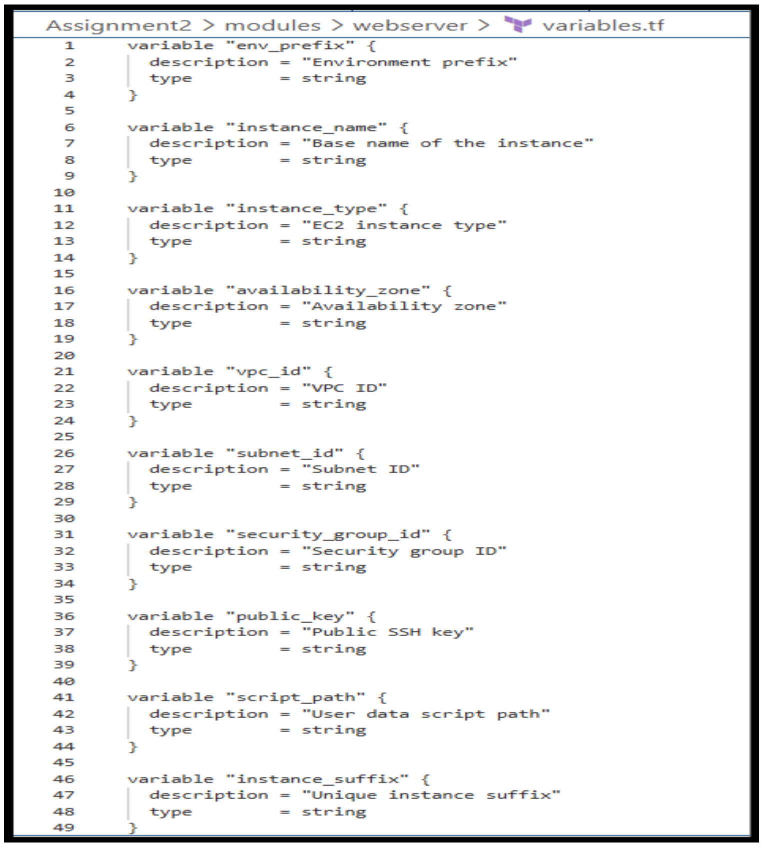
1.5 Locals Configuration:


Assignment2 > locals.tf

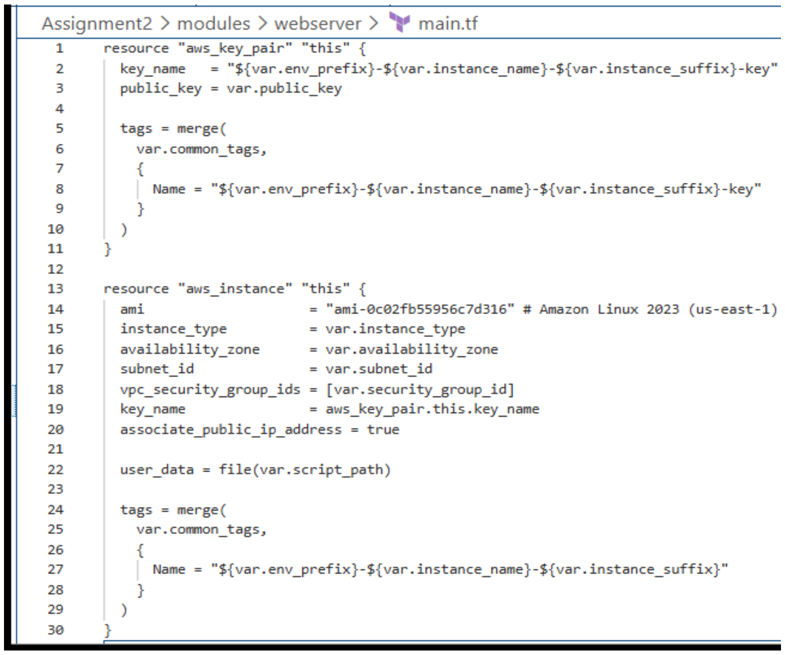
```
# Get your public IP dynamically
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

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

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

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

Part 2 : Webserver Module

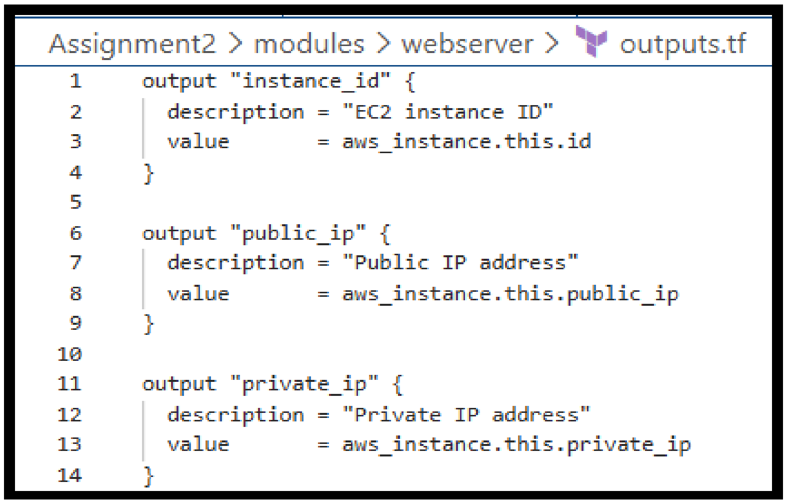
2.1 Module Design:


Assignment2 > modules > webserver > variables.tf

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

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

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

variable "availability_zone" {
  description = "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 SSH key"
  type        = string
}

variable "script_path" {
  description = "User data script path"
  type        = string
}

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

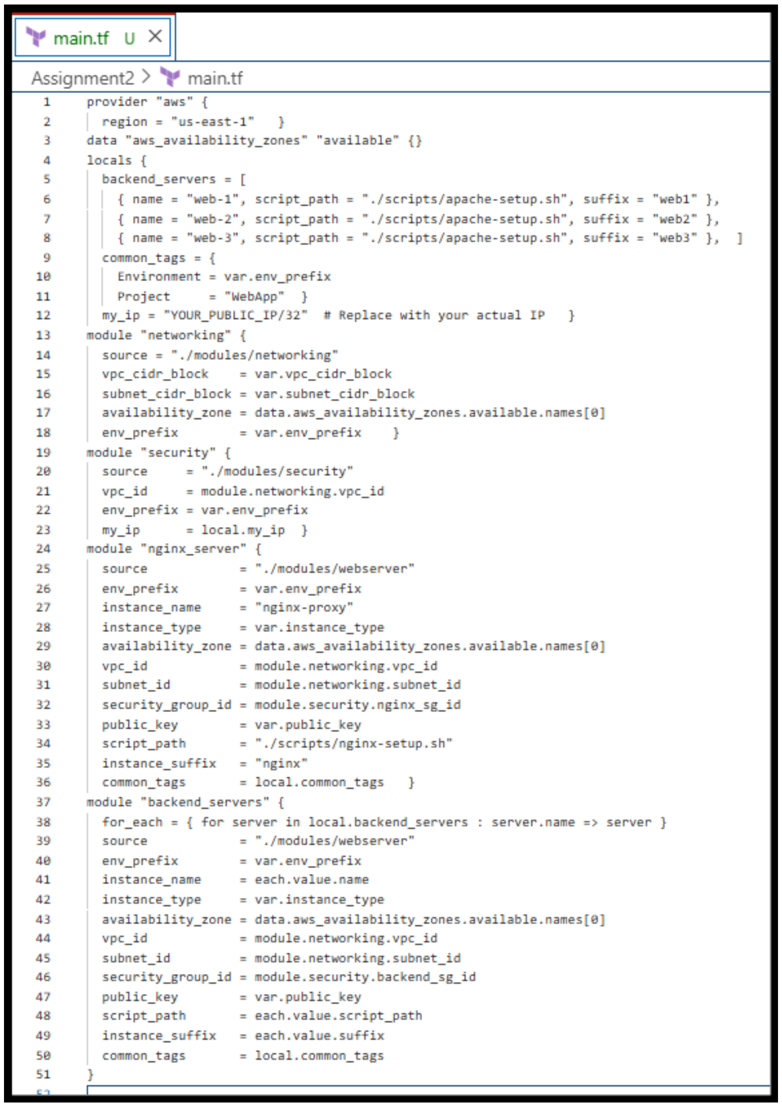

Assignment2 > modules > webserver > main.tf

```hcl
resource "aws_key_pair" "this" {
  key_name   = "${var.env_prefix}-${var.instance_name}-${var.instance_suffix}-key"
  public_key = var.public_key

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

resource "aws_instance" "this" {
  ami                         = "ami-0c02fb55956c7d316" # Amazon Linux 2023 (us-east-1)
  instance_type               = var.instance_type
  availability_zone           = var.availability_zone
  subnet_id                   = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name                    = aws_key_pair.this.key_name
  associate_public_ip_address = true

  user_data = file(var.script_path)

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


Assignment2 > modules > webserver > outputs.tf

```hcl
output "instance_id" {
  description = "EC2 instance ID"
  value       = aws_instance.this.id
}

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

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

2.2 Module Usage:

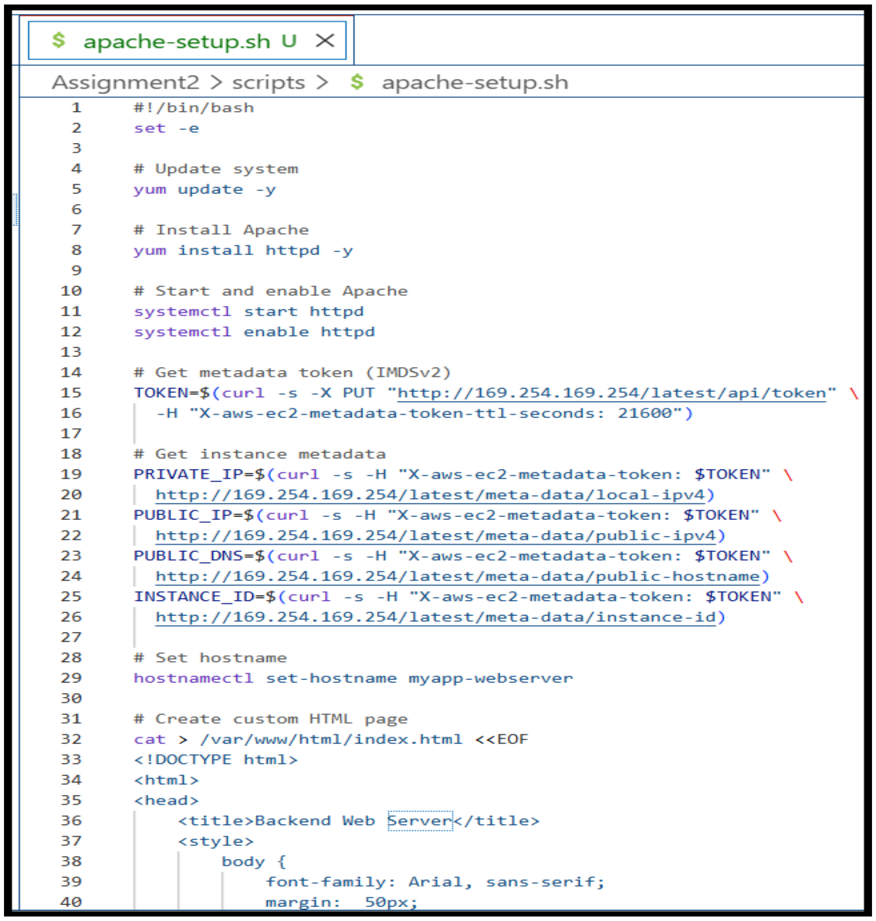

main.tf U

Assignment2 > main.tf

```
provider "aws" {
  region = "us-east-1"   }
data "aws_availability_zones" "available" {}
locals {
  backend_servers = [
    { name = "web-1", 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" },  ]
  common_tags = {
    Environment = var.env_prefix
    Project     = "WebApp"  }
  my_ip = "YOUR_PUBLIC_IP/32"  # Replace with your actual IP   }
module "networking" {
  source = "./modules/networking"
  vpc_cidr_block    = var.vpc_cidr_block
  subnet_cidr_block = var.subnet_cidr_block
  availability_zone = data.aws_availability_zones.available.names[0]
  env_prefix        = var.env_prefix    }
module "security" {
  source     = "./modules/security"
  vpc_id     = module.networking.vpc_id
  env_prefix = var.env_prefix
  my_ip      = local.my_ip  }
module "nginx_server" {
  source            = "./modules/webserver"
  env_prefix        = var.env_prefix
  instance_name     = "nginx-proxy"
  instance_type     = var.instance_type
  availability_zone = data.aws_availability_zones.available.names[0]
  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   }
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 = data.aws_availability_zones.available.names[0]
  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.backend_sg_id
  public_key        = var.public_key
  script_path       = each.value.script_path
  instance_suffix   = each.value.suffix
  common_tags       = local.common_tags
}
```

Part 3 : Server Configuration Scripts

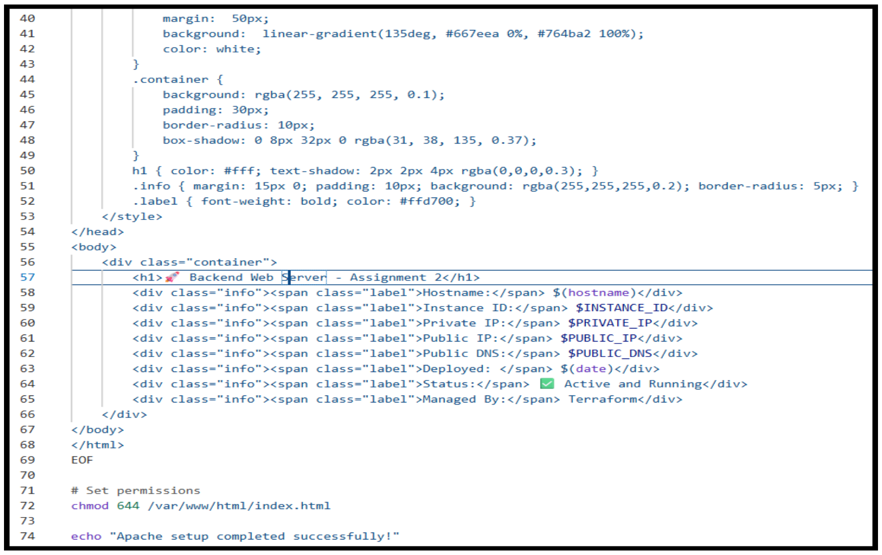
3.1 Apache Backend Server Script:



$ apache-setup.sh U ×

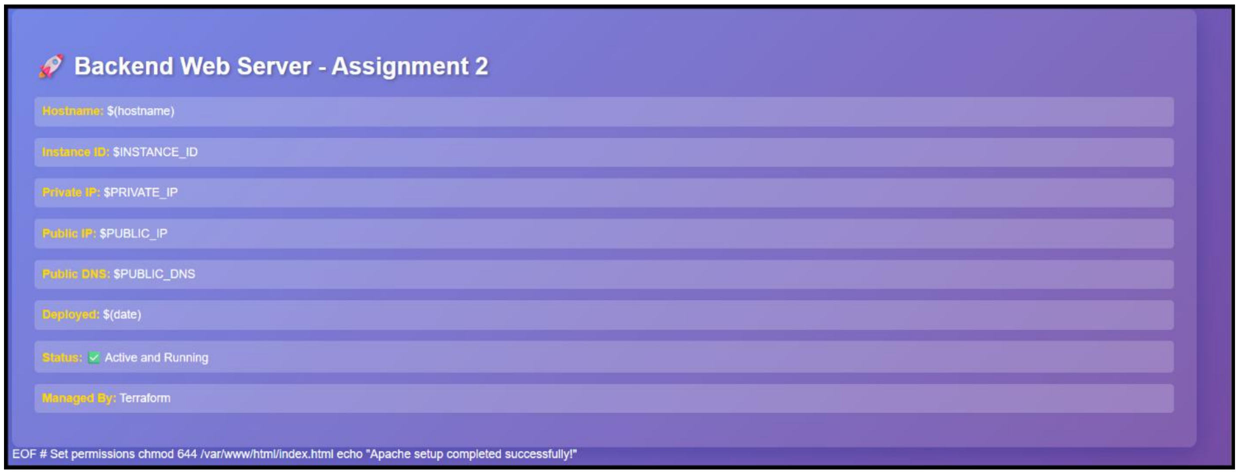
Assignment2 > scripts > $ apache-setup.sh

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

# Update system
yum update -y

# Install Apache
yum install httpd -y

# Start and enable Apache
systemctl start httpd
systemctl enable httpd

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

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

# Set hostname
hostnamectl set-hostname myapp-webserver

# Create custom HTML page
cat > /var/www/html/index.html <<EOF
<!DOCTYPE html>
<html>
<head>
    <title>Backend Web Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin:  50px;
```



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

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

echo "Apache setup completed successfully!"
```

- **browser showing backend server page:**



🚀 Backend Web Server - Assignment 2

Hostname: $(hostname)

Instance ID: $INSTANCE_ID

Private IP: $PRIVATE_IP

Public IP: $PUBLIC_IP

Public DNS: $PUBLIC_DNS

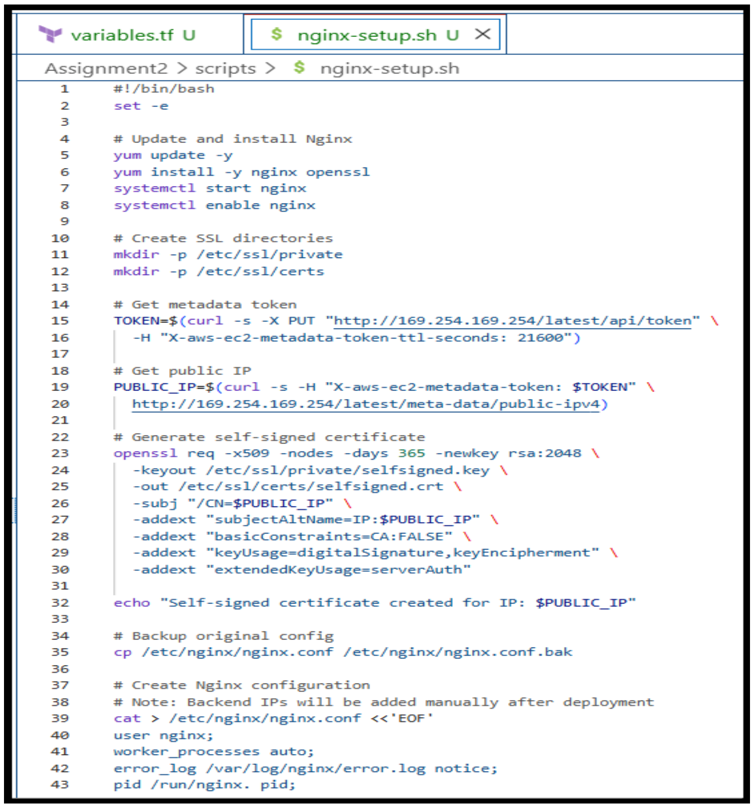
Deployed: $(date)

Status: ✅ Active and Running

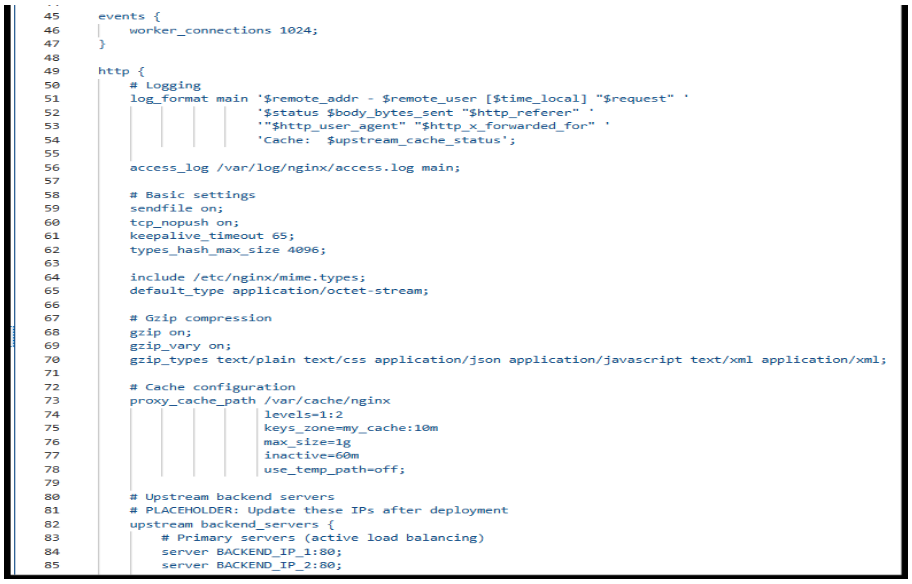
Managed By: Terraform

EOF # Set permissions chmod 644 /var/www/html/index.html echo "Apache setup completed successfully!"

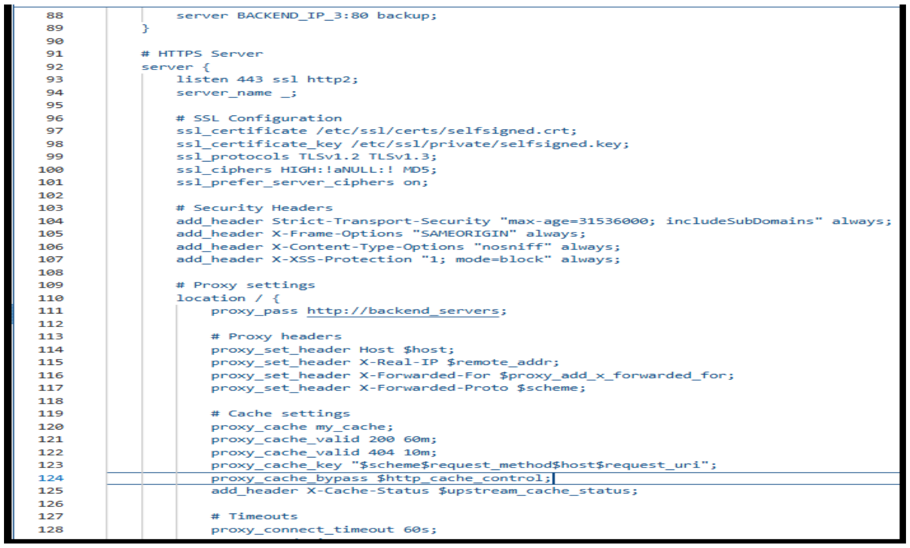
3.2 Nginx Server Setup Script:



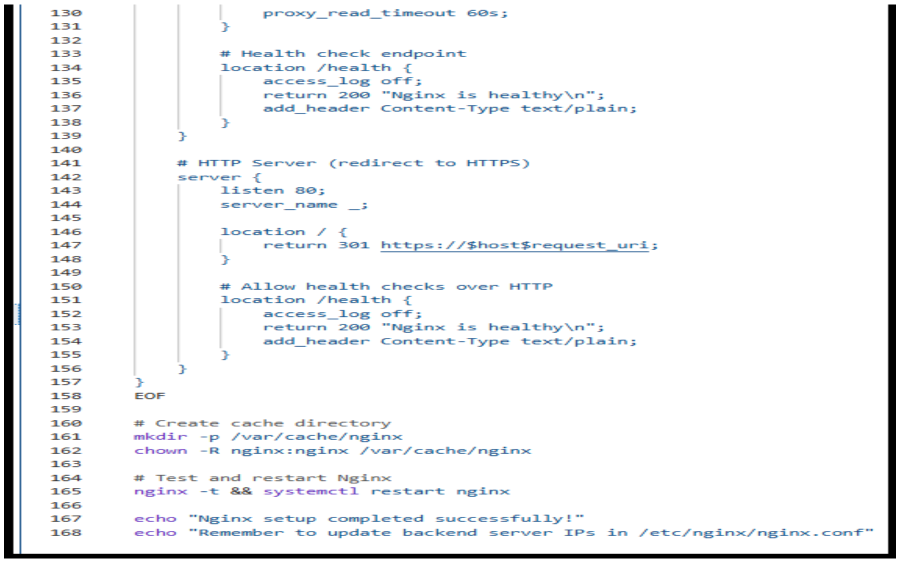
```bash
#!/bin/bash
set -e

# Update and install Nginx
yum update -y
yum install -y nginx openssl
systemctl start nginx
systemctl enable nginx

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

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

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

# Generate self-signed certificate
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/ssl/private/selfsigned.key \
  -out /etc/ssl/certs/selfsigned.crt \
  -subj "/CN=$PUBLIC_IP" \
  -addext "subjectAltName=IP:$PUBLIC_IP" \
  -addext "basicConstraints=CA:FALSE" \
  -addext "keyUsage=digitalSignature,keyEncipherment" \
  -addext "extendedKeyUsage=serverAuth"

echo "Self-signed certificate created for IP: $PUBLIC_IP"

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

# Create Nginx configuration
# Note: Backend IPs will be added manually after deployment
cat > /etc/nginx/nginx.conf <<'EOF'
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx. pid;
```



```
events {
    worker_connections 1024;
}

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

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

    # Basic settings
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 65;
    types_hash_max_size 4096;

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

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

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

    # Upstream backend servers
    # PLACEHOLDER: Update these IPs after deployment
    upstream backend_servers {
        # Primary servers (active load balancing)
        server BACKEND_IP_1:80;
        server BACKEND_IP_2:80;
```

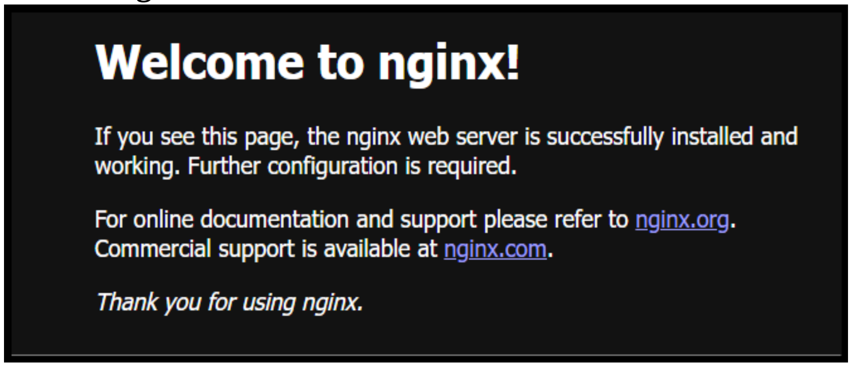


```
        server BACKEND_IP_3:80 backup;
    }

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

        # SSL Configuration
        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;

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

        # Proxy settings
        location / {
            proxy_pass http://backend_servers;

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

            # Cache settings
            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;

            # Timeouts
            proxy_connect_timeout 60s;
```



```
            proxy_read_timeout 60s;
        }

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

    # HTTP Server (redirect to HTTPS)
    server {
        listen 80;
        server_name _;

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

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

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

# Test and restart Nginx
nginx -t && systemctl restart nginx

echo "Nginx setup completed successfully!"
echo "Remember to update backend server IPs in /etc/nginx/nginx.conf"
```

- **before backend configuration:**



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

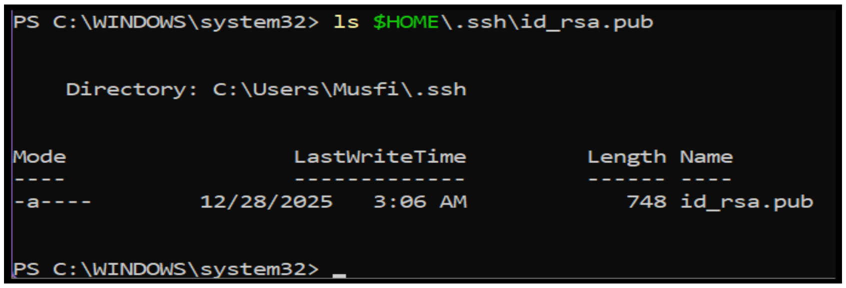
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part 4 : Infrastructure Deployment

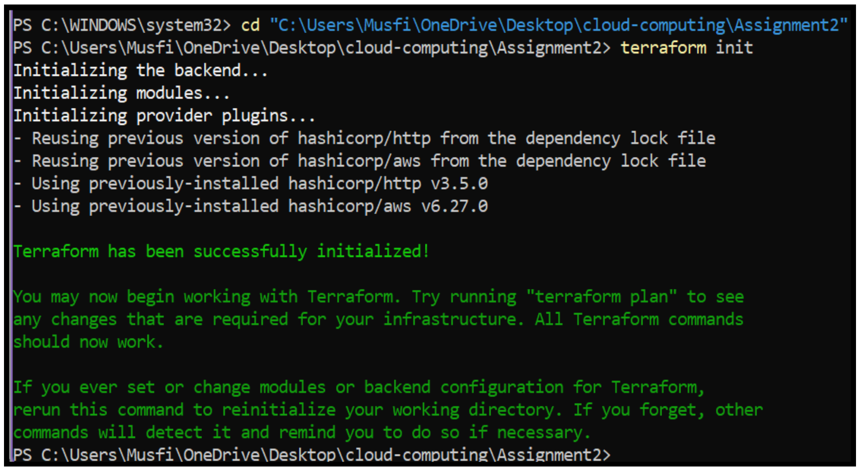
4.1 Initial Deployment:

- **ssh keygen:**



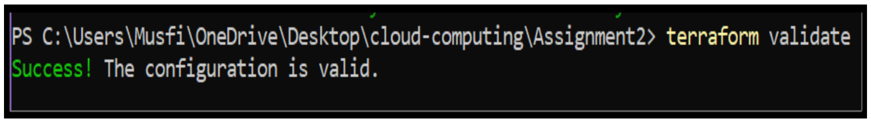
```
PS C:\WINDOWS\system32> ls $HOME\.ssh\id_rsa.pub


    Directory: C:\Users\Musfi\.ssh


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/28/2025   3:06 AM            748 id_rsa.pub


PS C:\WINDOWS\system32> _
```

- **terraform init:**



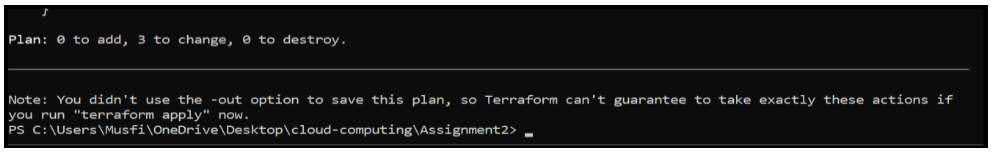
```
PS C:\WINDOWS\system32> cd "C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2"
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/http from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/http v3.5.0
- Using previously-installed hashicorp/aws v6.27.0

Terraform has been successfully initialized!

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

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2>
```

- **terraform validate:**



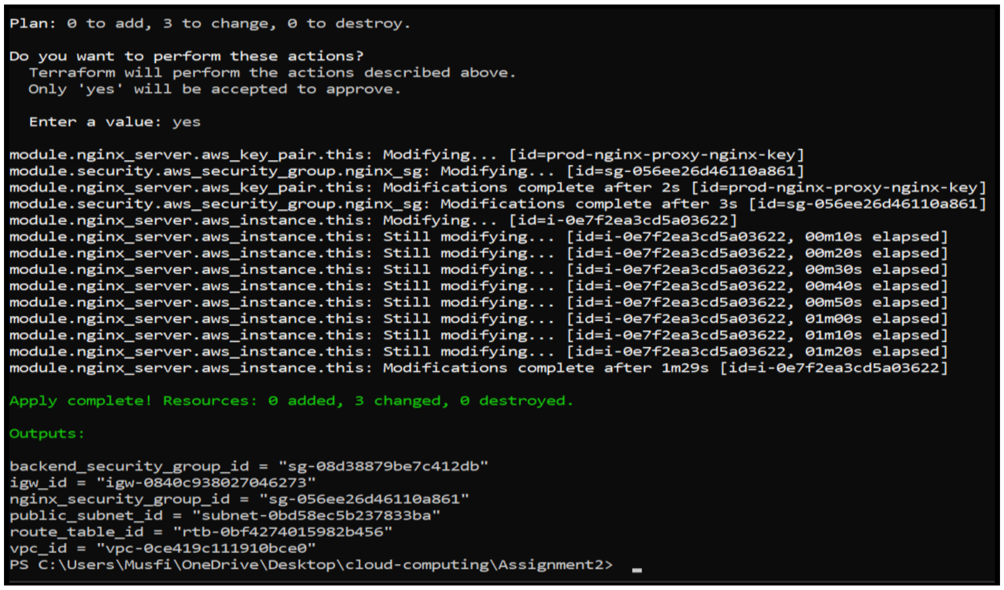
```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> terraform validate
Success! The configuration is valid.
```

- **terraform plan:**



```
    }
Plan: 0 to add, 3 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> _
```

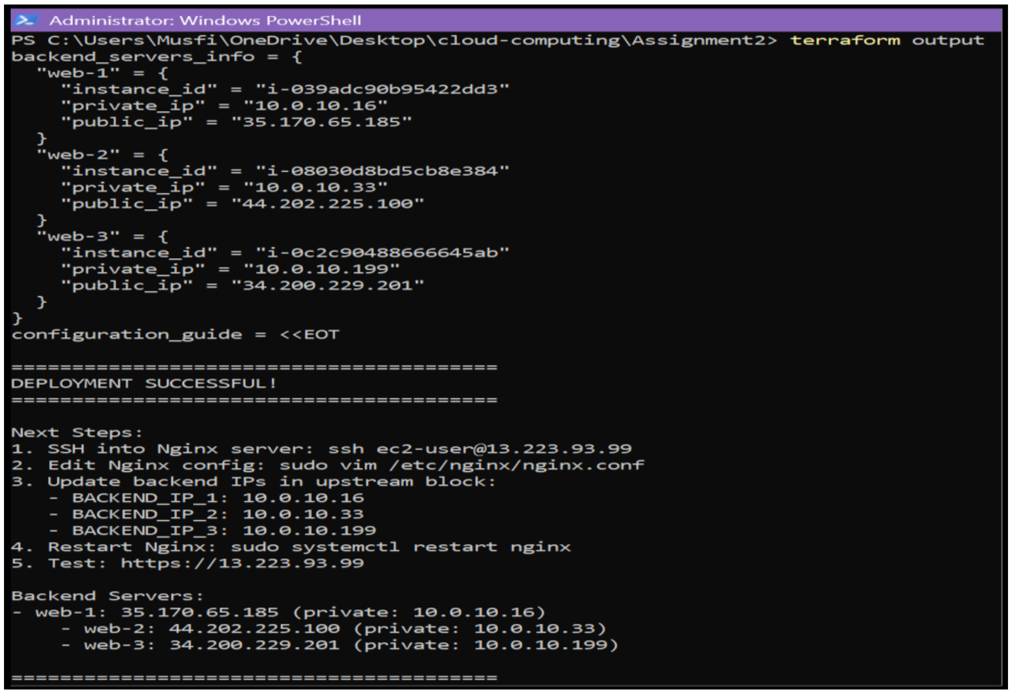
- **terraform apply:**



```
Plan: 0 to add, 3 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_key_pair.this: Modifying... [id=prod-nginx-proxy-nginx-key]
module.security.aws_security_group.nginx_sg: Modifying... [id=sg-056ee26d46110a861]
module.nginx_server.aws_key_pair.this: Modifications complete after 2s [id=prod-nginx-proxy-nginx-key]
module.security.aws_security_group.nginx_sg: Modifications complete after 3s [id=sg-056ee26d46110a861]
module.nginx_server.aws_instance.this: Modifying... [id=i-0e7f2ea3cd5a03622]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 00m10s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 00m20s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 00m30s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 00m40s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 00m50s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 01m00s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 01m10s elapsed]
module.nginx_server.aws_instance.this: Still modifying... [id=i-0e7f2ea3cd5a03622, 01m20s elapsed]
module.nginx_server.aws_instance.this: Modifications complete after 1m29s [id=i-0e7f2ea3cd5a03622]

Apply complete! Resources: 0 added, 3 changed, 0 destroyed.

Outputs:

backend_security_group_id = "sg-08d38879be7c412db"
igw_id = "igw-0840c938027046273"
nginx_security_group_id = "sg-056ee26d46110a861"
public_subnet_id = "subnet-0bd58ec5b237833ba"
route_table_id = "rtb-0bf4274015982b456"
vpc_id = "vpc-0ce419c111910bce0"
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2>  _
```

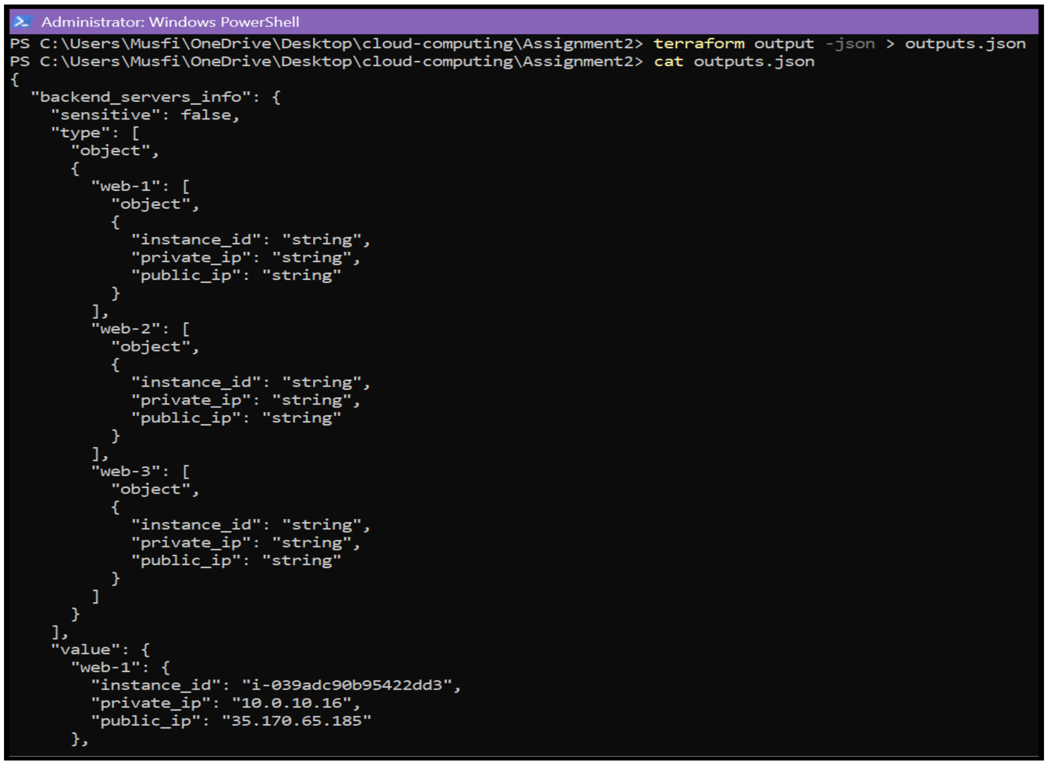
4.2 Output Configuration:

- **Terraform output:**



```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> terraform output
backend_servers_info = {
  "web-1" = {
    "instance_id" = "i-039adc90b95422dd3"
    "private_ip" = "10.0.10.16"
    "public_ip" = "35.170.65.185"
  }
  "web-2" = {
    "instance_id" = "i-08030d8bd5cb8e384"
    "private_ip" = "10.0.10.33"
    "public_ip" = "44.202.225.100"
  }
  "web-3" = {
    "instance_id" = "i-0c2c90488666645ab"
    "private_ip" = "10.0.10.199"
    "public_ip" = "34.200.229.201"
  }
}
configuration_guide = <<EOT

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

Next Steps:
1. SSH into Nginx server: ssh ec2-user@13.223.93.99
2. Edit Nginx config: sudo vim /etc/nginx/nginx.conf
3. Update backend IPs in upstream block:
   - BACKEND_IP_1: 10.0.10.16
   - BACKEND_IP_2: 10.0.10.33
   - BACKEND_IP_3: 10.0.10.199
4. Restart Nginx: sudo systemctl restart nginx
5. Test: https://13.223.93.99

Backend Servers:
- web-1: 35.170.65.185 (private: 10.0.10.16)
    - web-2: 44.202.225.100 (private: 10.0.10.33)
    - web-3: 34.200.229.201 (private: 10.0.10.199)

========================================
```

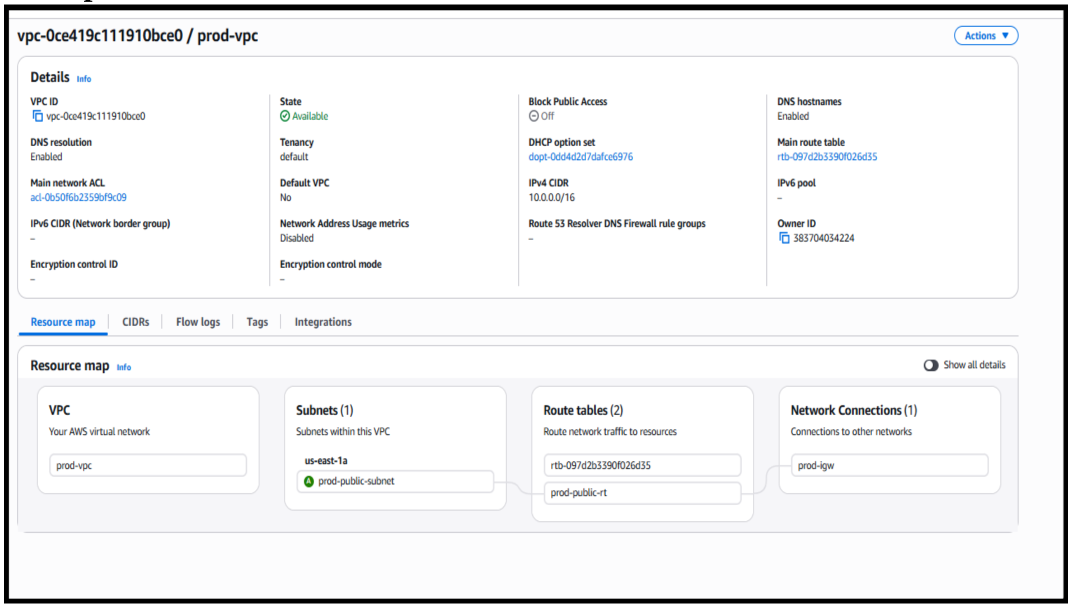
- **Terraform outputs json:**



```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> terraform output -json > outputs.json
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> cat outputs.json
{
  "backend_servers_info": {
    "sensitive": false,
    "type": [
      "object",
      {
        "web-1": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-2": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ],
        "web-3": [
          "object",
          {
            "instance_id": "string",
            "private_ip": "string",
            "public_ip": "string"
          }
        ]
      }
    ],
    "value": {
      "web-1": {
        "instance_id": "i-039adc90b95422dd3",
        "private_ip": "10.0.10.16",
        "public_ip": "35.170.65.185"
      },
```

4.3 AWS Console Verification:

- **aws vpc:**



vpc-0ce419c111910bce0 / prod-vpc

Details

VPC ID vpc-0ce419c111910bce0	State Available	Block Public Access Off	DNS hostnames Enabled
DNS resolution Enabled	Tenancy default	DHCP option set dopt-0dd4d2d7dafce6976	Main route table rtb-097d2b3390f026d35
Main network ACL acl-0b50f6b2359bf9c09	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool –
IPv6 CIDR (Network border group) –	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 383704034224
Encryption control ID –	Encryption control mode –		

Resource map

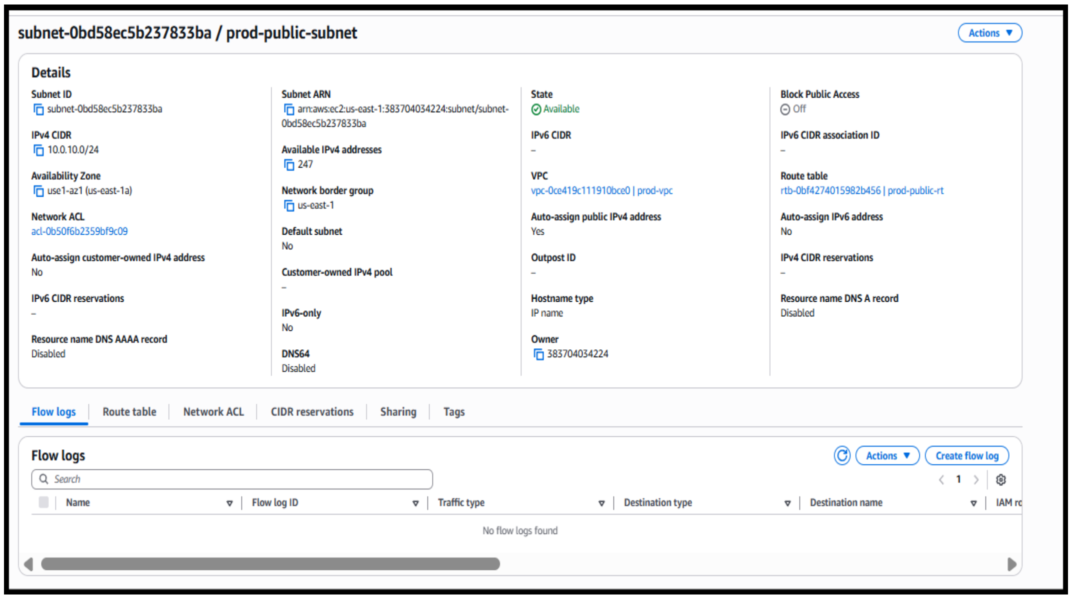
VPC (Your AWS virtual network): prod-vpc

Subnets (1) (Subnets within this VPC): us-east-1a — prod-public-subnet

Route tables (2) (Route network traffic to resources): rtb-097d2b3390f026d35, prod-public-rt

Network Connections (1) (Connections to other networks): prod-igw

- **aws subnet:**



subnet-0bd58ec5b237833ba / prod-public-subnet

Details

Subnet ID: subnet-0bd58ec5b237833ba	Subnet ARN: arn:aws:ec2:us-east-1:383704034224:subnet/subnet-0bd58ec5b237833ba	State: Available	Block Public Access: Off
IPv4 CIDR: 10.0.10.0/24	Available IPv4 addresses: 247	IPv6 CIDR: –	IPv6 CIDR association ID: –
Availability Zone: use1-az1 (us-east-1a)	Network border group: us-east-1	VPC: vpc-0ce419c111910bce0 | prod-vpc	Route table: rtb-0bf4274015982b456 | prod-public-rt
Network ACL: acl-0b50f6b2359bf9c09	Default subnet: No	Auto-assign public IPv4 address: Yes	Auto-assign IPv6 address: No
Auto-assign customer-owned IPv4 address: No	Customer-owned IPv4 pool: –	Outpost ID: –	IPv4 CIDR reservations: –
IPv6 CIDR reservations: –	IPv6-only: No	Hostname type: IP name	Resource name DNS A record: Disabled
Resource name DNS AAAA record: Disabled	DNS64: Disabled	Owner: 383704034224	

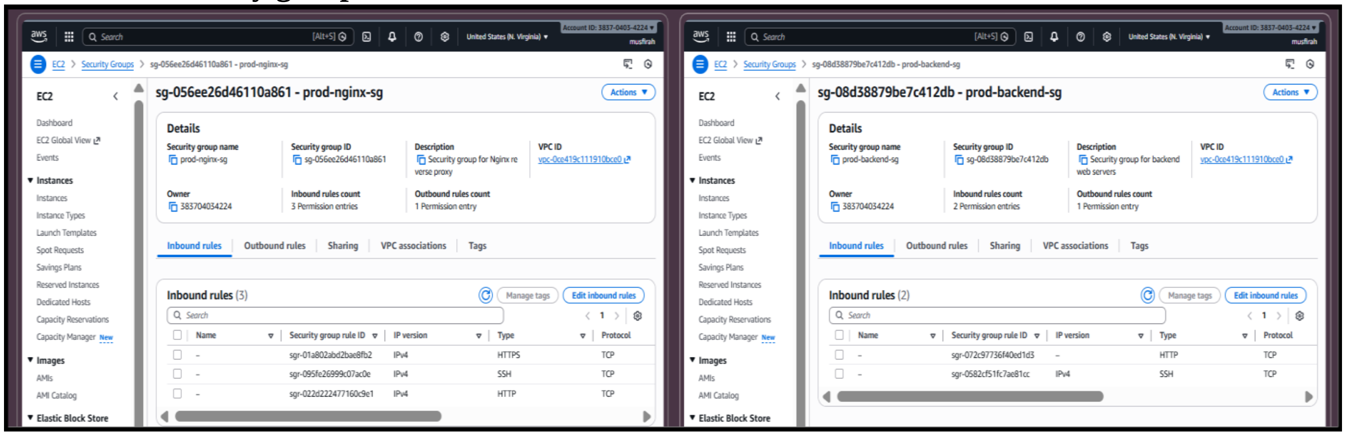
Flow logs | Route table | Network ACL | CIDR reservations | Sharing | Tags

Flow logs

Name	Flow log ID	Traffic type	Destination type	Destination name	IAM ro

No flow logs found

- **aws security groups:**



sg-056ee26d46110a861 - prod-nginx-sg

Details

Security group name	Security group ID	Description	VPC ID
prod-nginx-sg	sg-056ee26d46110a861	Security group for Nginx reverse proxy	vpc-0ce419c111910bce0

Owner	Inbound rules count	Outbound rules count
383704034224	3 Permission entries	1 Permission entry

Inbound rules (3)

Name	Security group rule ID	IP version	Type	Protocol
–	sgr-01a802abd2bae8fb2	IPv4	HTTPS	TCP
–	sgr-095fe26999c07ac0e	IPv4	SSH	TCP
–	sgr-022d222477160c9e1	IPv4	HTTP	TCP

sg-08d38879be7c412db - prod-backend-sg

Details

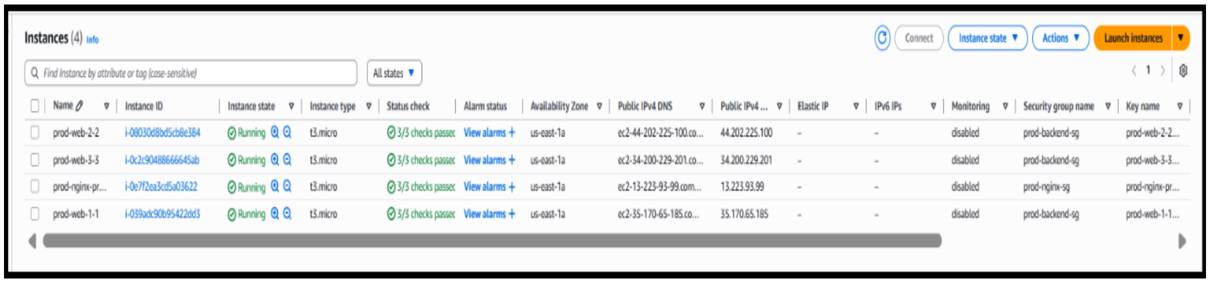
Security group name	Security group ID	Description	VPC ID
prod-backend-sg	sg-08d38879be7c412db	Security group for backend web servers	vpc-0ce419c111910bce0

Owner	Inbound rules count	Outbound rules count
383704034224	2 Permission entries	1 Permission entry

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol
–	sgr-072c97736f40ed1d3	–	HTTP	TCP
–	sgr-0582cf51fc7ae81cc	IPv4	SSH	TCP

- **aws instances:**



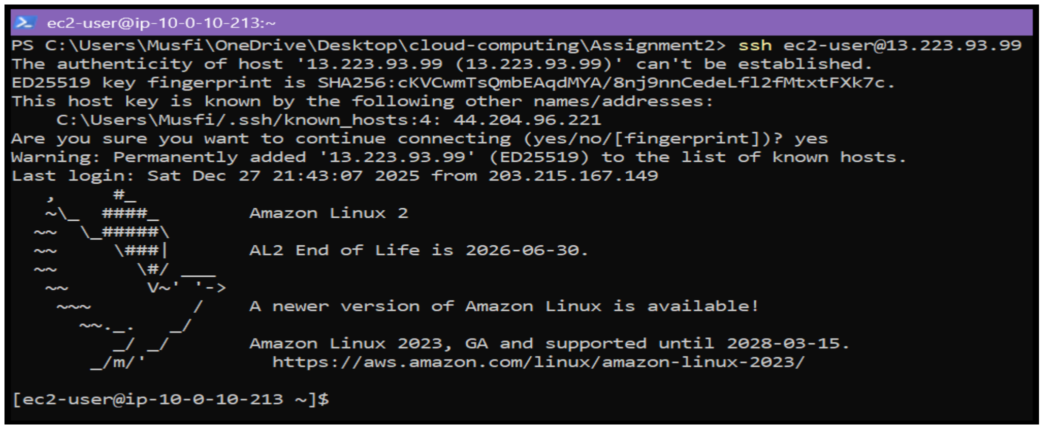
Instances (4)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group name	Key name
prod-web-2-2	i-08030d8bd5cb8e384	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-44-202-225-100.co...	44.202.225.100	–	–	disabled	prod-backend-sg	prod-web-2-2...
prod-web-3-3	i-0c2c90488666645ab	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-34-200-229-201.co...	34.200.229.201	–	–	disabled	prod-backend-sg	prod-web-3-3...
prod-nginx-pr...	i-0e7f2ea3cd5a03622	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-13-223-93-99.com...	13.223.93.99	–	–	disabled	prod-nginx-sg	prod-nginx-pr...
prod-web-1-1	i-039adc90b95422dd3	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-35-170-65-185.co...	35.170.65.185	–	–	disabled	prod-backend-sg	prod-web-1-1...

Part 5 : Nginx Configuration & Testing

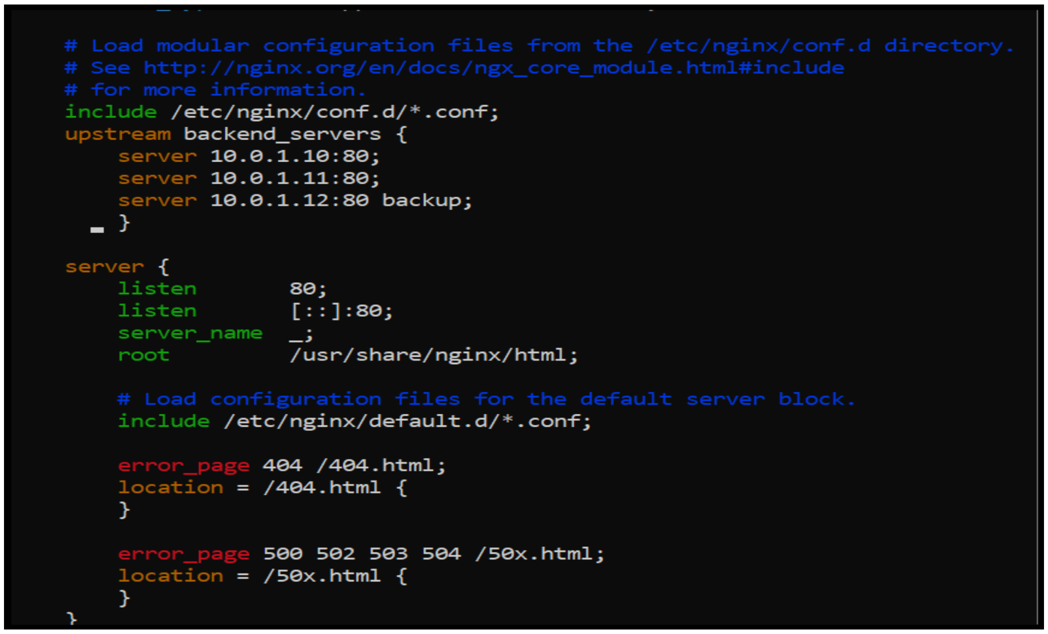
5.1 Update Nginx Backend Configuration:

- **ssh nginx:**



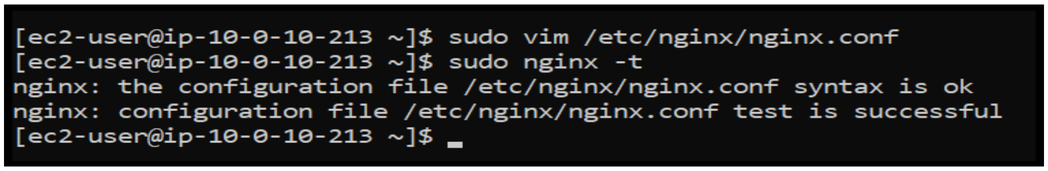
```
ec2-user@ip-10-0-10-213:~
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> ssh ec2-user@13.223.93.99
The authenticity of host '13.223.93.99 (13.223.93.99)' can't be established.
ED25519 key fingerprint is SHA256:cKVCwmTsQmbEAqdMYA/8nj9nnCedeLfl2fMtxtFXk7c.
This host key is known by the following other names/addresses:
    C:\Users\Musfi/.ssh/known_hosts:4: 44.204.96.221
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.223.93.99' (ED25519) to the list of known hosts.
Last login: Sat Dec 27 21:43:07 2025 from 203.215.167.149
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-10-0-10-213 ~]$
```

- **nginx conf updated:**



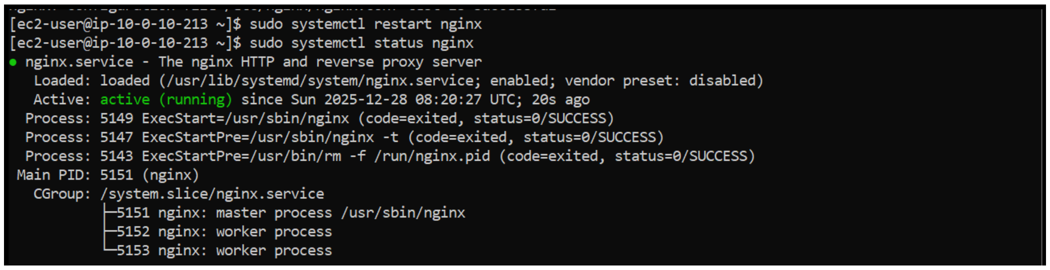
```
    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
    upstream backend_servers {
        server 10.0.1.10:80;
        server 10.0.1.11:80;
        server 10.0.1.12:80 backup;
      }

    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html;
        location = /404.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

- **nginx test:**



```
[ec2-user@ip-10-0-10-213 ~]$ sudo vim /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-213 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-213 ~]$
```

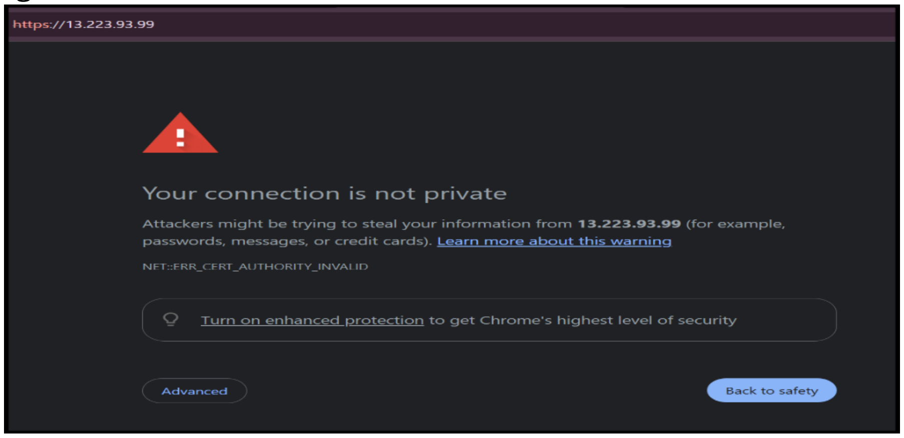
- **nginx restart and status:**



```
[ec2-user@ip-10-0-10-213 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-213 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2025-12-28 08:20:27 UTC; 20s ago
  Process: 5149 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 5147 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 5143 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 5151 (nginx)
   CGroup: /system.slice/nginx.service
           ├─5151 nginx: master process /usr/sbin/nginx
           ├─5152 nginx: worker process
           └─5153 nginx: worker process
```

5.2 Test Load Balancing:

- **ssl warning:**



https://13.223.93.99

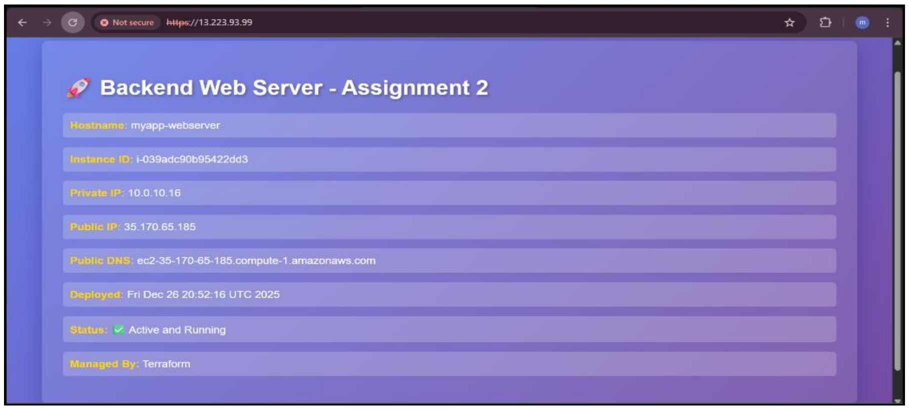
Your connection is not private

Attackers might be trying to steal your information from **13.223.93.99** (for example, passwords, messages, or credit cards). Learn more about this warning

NET::ERR_CERT_AUTHORITY_INVALID

Turn on enhanced protection to get Chrome's highest level of security

- **web1 response:**

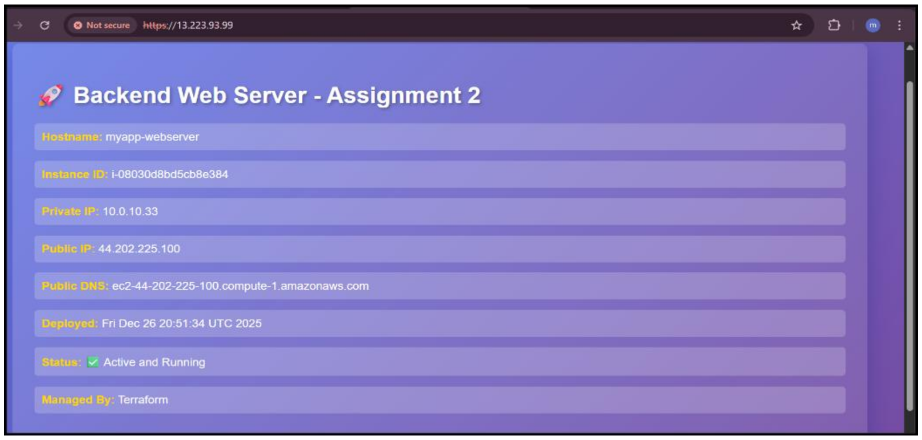


Not secure ~~https~~://13.223.93.99

🚀 Backend Web Server - Assignment 2

Hostname: myapp-webserver
Instance ID: i-039adc90b95422dd3
Private IP: 10.0.10.16
Public IP: 35.170.65.185
Public DNS: ec2-35-170-65-185.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:52:16 UTC 2025
Status: ✅ Active and Running
Managed By: Terraform

- **web2 response:**

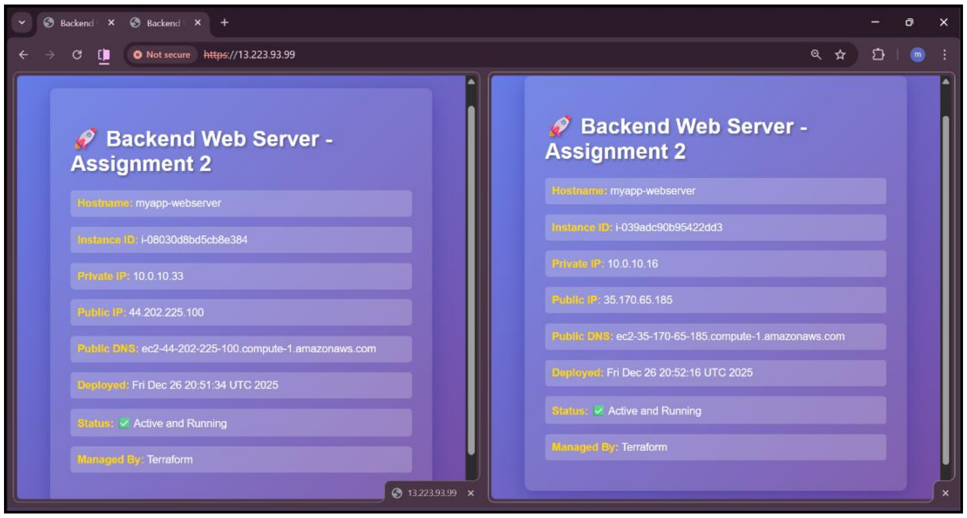


Not secure ~~https~~://13.223.93.99

🚀 Backend Web Server - Assignment 2

Hostname: myapp-webserver
Instance ID: i-08030d8bd5cb8e384
Private IP: 10.0.10.33
Public IP: 44.202.225.100
Public DNS: ec2-44-202-225-100.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:51:34 UTC 2025
Status: ✅ Active and Running
Managed By: Terraform

- **load balancing demo:**


Backend Web Server - Assignment 2

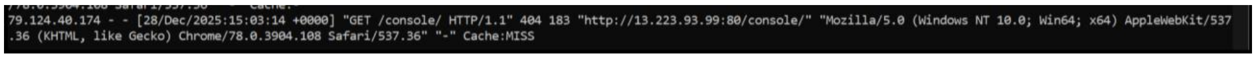
Hostname: myapp-webserver
Instance ID: i-08030d8bd5cb8e384
Private IP: 10.0.10.33
Public IP: 44.202.225.100
Public DNS: ec2-44-202-225-100.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:51:34 UTC 2025
Status: ✅ Active and Running
Managed By: Terraform

Backend Web Server - Assignment 2

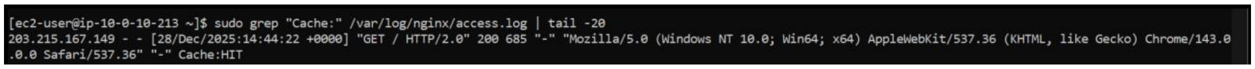
Hostname: myapp-webserver
Instance ID: i-039adc90b95422dd3
Private IP: 10.0.10.16
Public IP: 35.170.65.185
Public DNS: ec2-35-170-65-185.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:52:16 UTC 2025
Status: ✅ Active and Running
Managed By: Terraform

5.3 Test Cache Functionality:

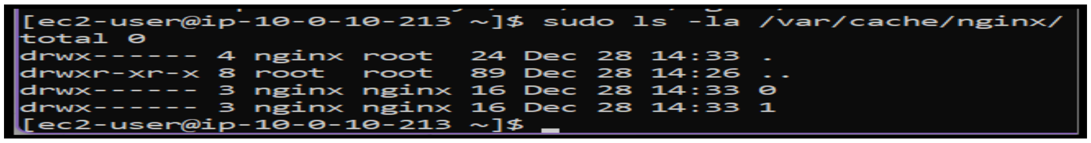
- **cache miss:**


```
79.124.40.174 - - [28/Dec/2025:15:03:14 +0000] "GET /console/ HTTP/1.1" 404 183 "http://13.223.93.99:80/console/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36" "-" Cache:MISS
```

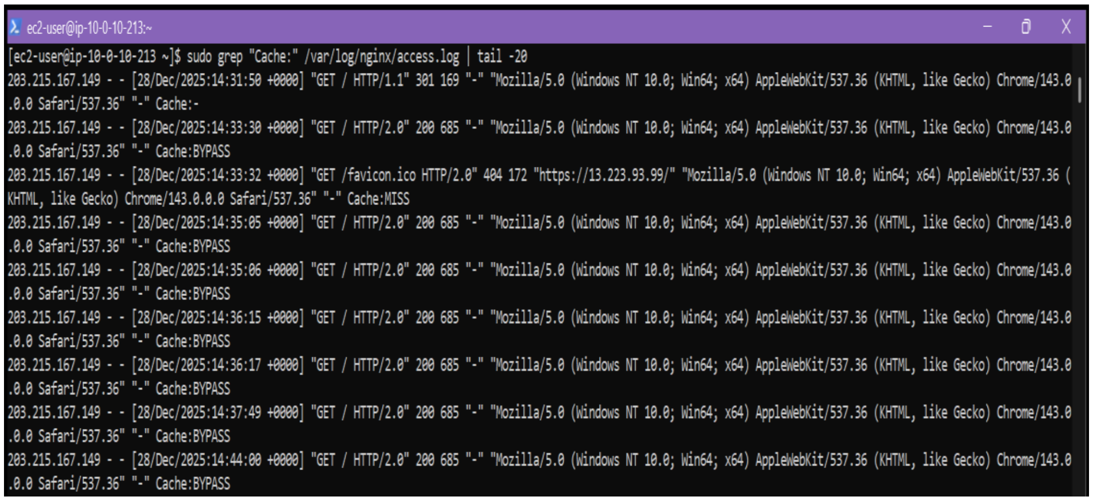
- **cache hit:**


```
[ec2-user@ip-10-0-10-213 ~]$ sudo grep "Cache:" /var/log/nginx/access.log | tail -20
203.215.167.149 - - [28/Dec/2025:14:44:22 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:HIT
```

- **cache directory:**


```
[ec2-user@ip-10-0-10-213 ~]$ sudo ls -la /var/cache/nginx/
total 0
drwx------ 4 nginx root   24 Dec 28 14:33 .
drwxr-xr-x 8 root  root   89 Dec 28 14:26 ..
drwx------ 3 nginx nginx  16 Dec 28 14:33 0
drwx------ 3 nginx nginx  16 Dec 28 14:33 1
[ec2-user@ip-10-0-10-213 ~]$
```

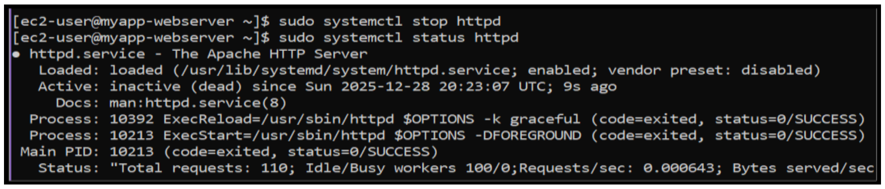
- **access log cache:**



```
[ec2-user@ip-10-0-10-213 ~]$ sudo grep "Cache:" /var/log/nginx/access.log | tail -20
203.215.167.149 - - [28/Dec/2025:14:31:50 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:-
203.215.167.149 - - [28/Dec/2025:14:33:30 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:33:32 +0000] "GET /favicon.ico HTTP/2.0" 404 172 "https://13.223.93.99/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-" Cache:MISS
203.215.167.149 - - [28/Dec/2025:14:35:05 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:35:06 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:36:15 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:36:17 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:37:49 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
203.215.167.149 - - [28/Dec/2025:14:44:00 +0000] "GET / HTTP/2.0" 200 685 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0
.0.0 Safari/537.36" "-" Cache:BYPASS
```

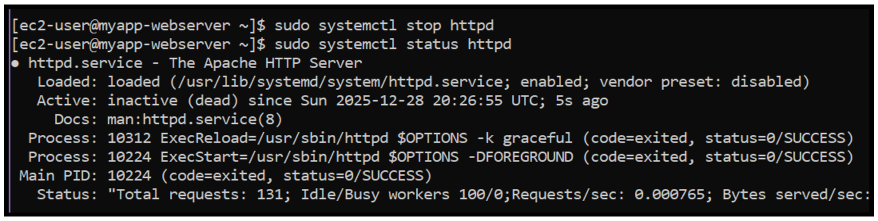
5.4 Test High Availability (Backup Server):

- **web1 stopped:**



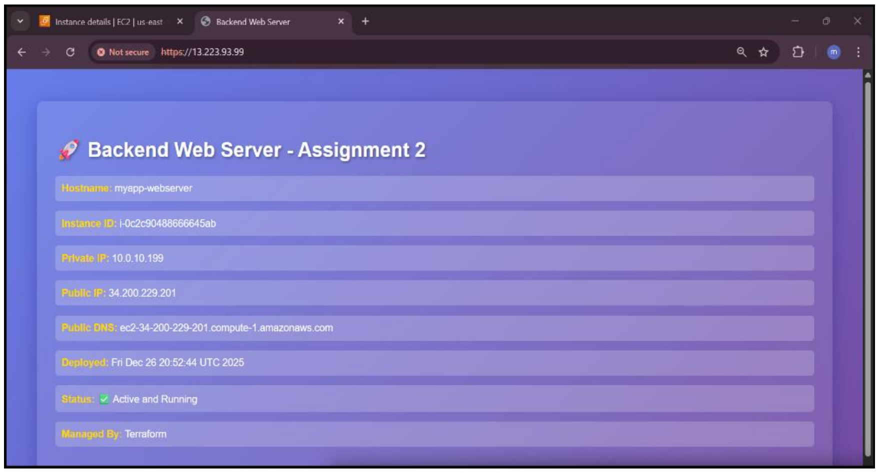
```
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Sun 2025-12-28 20:23:07 UTC; 9s ago
     Docs: man:httpd.service(8)
  Process: 10392 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
  Process: 10213 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
 Main PID: 10213 (code=exited, status=0/SUCCESS)
   Status: "Total requests: 110; Idle/Busy workers 100/0;Requests/sec: 0.000643; Bytes served/sec
```

- **web2 stopped:**



```
[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; vendor preset: disabled)
   Active: inactive (dead) since Sun 2025-12-28 20:26:55 UTC; 5s ago
     Docs: man:httpd.service(8)
  Process: 10312 ExecReload=/usr/sbin/httpd $OPTIONS -k graceful (code=exited, status=0/SUCCESS)
  Process: 10224 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
 Main PID: 10224 (code=exited, status=0/SUCCESS)
   Status: "Total requests: 131; Idle/Busy workers 100/0;Requests/sec: 0.000765; Bytes served/sec:
```

- **backup activated:**



🚀 Backend Web Server - Assignment 2

Hostname: myapp-webserver

Instance ID: i-0c2c90488666645ab

Private IP: 10.0.10.199

Public IP: 34.200.229.201

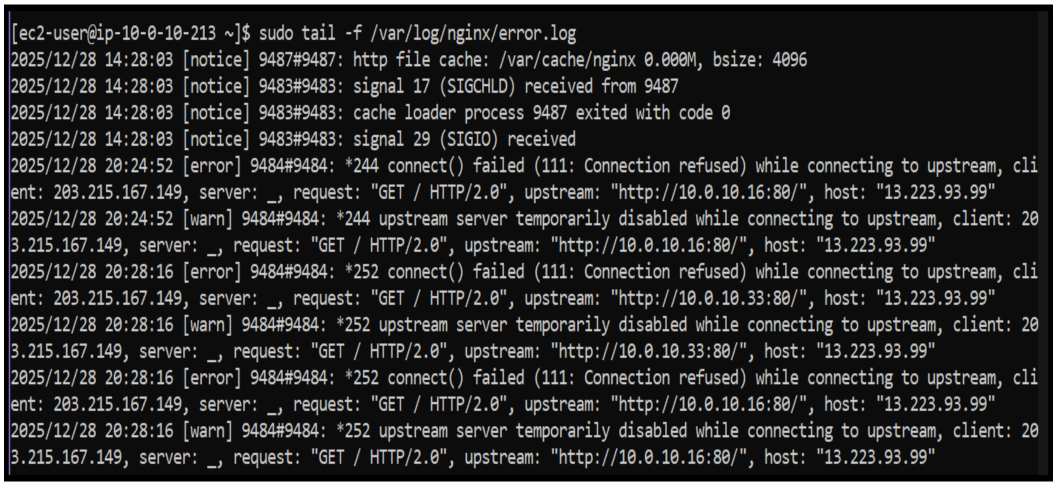
Public DNS: ec2-34-200-229-201.compute-1.amazonaws.com

Deployed: Fri Dec 26 20:52:44 UTC 2025

Status: ✅ Active and Running

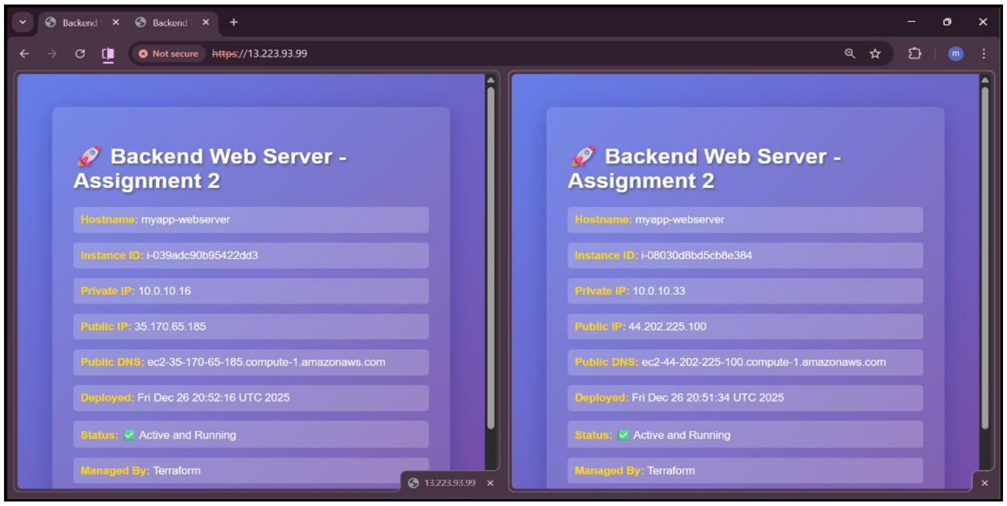
Managed By: Terraform

- **nginx error log:**



```
[ec2-user@ip-10-0-10-213 ~]$ sudo tail -f /var/log/nginx/error.log
2025/12/28 14:28:03 [notice] 9487#9487: http file cache: /var/cache/nginx 0.000M, bsize: 4096
2025/12/28 14:28:03 [notice] 9483#9483: signal 17 (SIGCHLD) received from 9487
2025/12/28 14:28:03 [notice] 9483#9483: cache loader process 9487 exited with code 0
2025/12/28 14:28:03 [notice] 9483#9483: signal 29 (SIGIO) received
2025/12/28 20:24:52 [error] 9484#9484: *244 connect() failed (111: Connection refused) while connecting to upstream, cli
ent: 203.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.16:80/", host: "13.223.93.99"
2025/12/28 20:24:52 [warn] 9484#9484: *244 upstream server temporarily disabled while connecting to upstream, client: 20
3.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.16:80/", host: "13.223.93.99"
2025/12/28 20:28:16 [error] 9484#9484: *252 connect() failed (111: Connection refused) while connecting to upstream, cli
ent: 203.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.33:80/", host: "13.223.93.99"
2025/12/28 20:28:16 [warn] 9484#9484: *252 upstream server temporarily disabled while connecting to upstream, client: 20
3.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.33:80/", host: "13.223.93.99"
2025/12/28 20:28:16 [error] 9484#9484: *252 connect() failed (111: Connection refused) while connecting to upstream, cli
ent: 203.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.16:80/", host: "13.223.93.99"
2025/12/28 20:28:16 [warn] 9484#9484: *252 upstream server temporarily disabled while connecting to upstream, client: 20
3.215.167.149, server: _, request: "GET / HTTP/2.0", upstream: "http://10.0.10.16:80/", host: "13.223.93.99"
```

- **services restored:**



🚀 Backend Web Server - Assignment 2

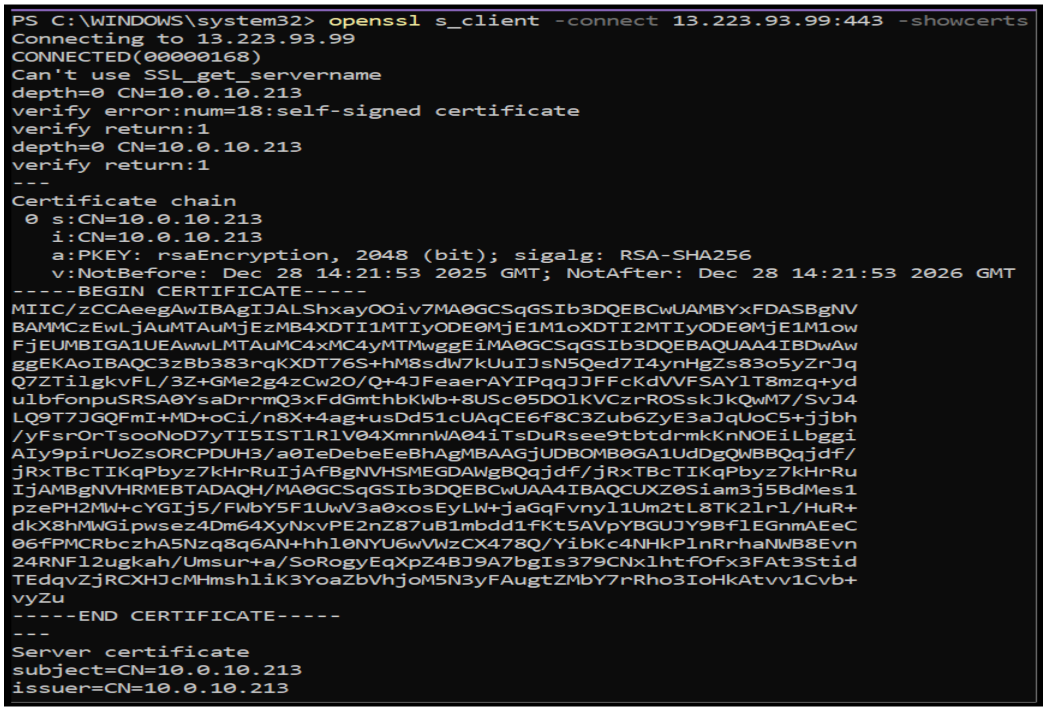
Hostname: myapp-webserver
Instance ID: i-039adc90b95422dd3
Private IP: 10.0.10.16
Public IP: 35.170.65.185
Public DNS: ec2-35-170-65-185.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:52:16 UTC 2025
Status: ✓ Active and Running
Managed By: Terraform

🚀 Backend Web Server - Assignment 2

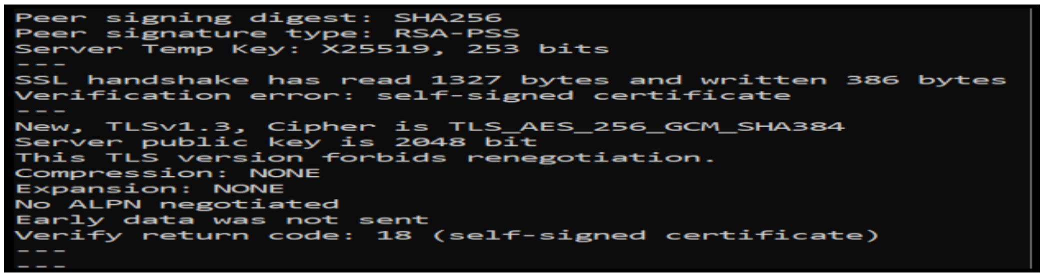
Hostname: myapp-webserver
Instance ID: i-08030d8bd5cb8e384
Private IP: 10.0.10.33
Public IP: 44.202.225.100
Public DNS: ec2-44-202-225-100.compute-1.amazonaws.com
Deployed: Fri Dec 26 20:51:34 UTC 2025
Status: ✓ Active and Running
Managed By: Terraform

5.5 Security & Performance Analysis:

- **ssl certificate:**

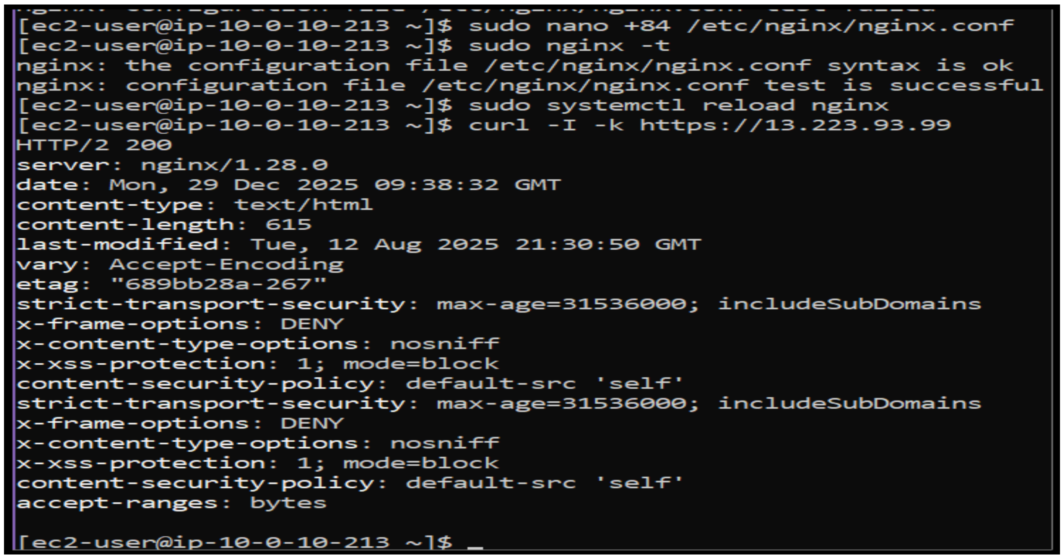


```
PS C:\WINDOWS\system32> openssl s_client -connect 13.223.93.99:443 -showcerts
Connecting to 13.223.93.99
CONNECTED(00000168)
Can't use SSL_get_servername
depth=0 CN=10.0.10.213
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN=10.0.10.213
verify return:1
---
Certificate chain
 0 s:CN=10.0.10.213
   i:CN=10.0.10.213
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Dec 28 14:21:53 2025 GMT; NotAfter: Dec 28 14:21:53 2026 GMT
-----BEGIN CERTIFICATE-----
MIIC/zCCAeegAwIBAgIJALShxayOOiv7MA0GCSqGSIb3DQEBCwUAMBYxFDASBgNV
BAMMCzEwLjAuMTAuMjEzMB4XDTI1MTIyODE0MjE1M1oXDTI2MTIyODE0MjE1M1ow
FjEUMBIGA1UEAwwLMTAuMC4xMC4yMTMwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAw
ggEKAoIBAQC3zBb383rqKXDT76S+hM8sdW7kUuIJsN5Qed7I4ynHgZs83o5yZrJq
Q7ZTilgkvFL/3Z+GMe2g4zCw2O/Q+4JFeaerAYIPqqJJFFcKdVVFSAYlT8mzq+yd
ulbfonpuSRSA0YsaDrrmQ3xFdGmthbKWb+8USc05DOlKVCzrROSskJkQwM7/SvJ4
LQ9T7JGQFmI+MD+oCi/n8X+4ag+usDd51cUAqCE6f8C3Zub6ZyE3aJqUoC5+jjbh
/yFsrOrTsooNoD7yTI5ISTlRlV04XmnnWA04iTsDuRsee9tbtdrmkKnNOEiLbggi
AIy9pirUoZsORCPDUH3/a0IeDebeEeBhAgMBAAGjUDBOMB0GA1UdDgQWBBQqjdf/
jRxTBcTIKqPbyz7kHrRuIjAfBgNVHSMEGDAWgBQqjdf/jRxTBcTIKqPbyz7kHrRu
IjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCUXZ0Siam3j5BdMes1
pzePH2MW+cYGIj5/FWbY5F1UwV3a0xosEyLW+jaGqFvnyl1Um2tL8TK2lrl/HuR+
dkX8hMWGipwsez4Dm64XyNxvPE2nZ87uB1mbdd1fKt5AVpYBGUJY9BflEGnmAEeC
06fPMCRbczhA5Nzq8q6AN+hhl0NYU6wVWzCX478Q/YibKc4NHkPlnRrhaNWB8Evn
24RNFl2ugkah/Umsur+a/SoRogyEqXpZ4BJ9A7bgIs379CNxlhtfOfx3FAt3Stid
TEdqvZjRCXHJcMHmshliK3YoaZbVhjoM5N3yFAugtZMbY7rRho3IoHkAtvv1Cvb+
vyZu
-----END CERTIFICATE-----
---
Server certificate
subject=CN=10.0.10.213
issuer=CN=10.0.10.213
```



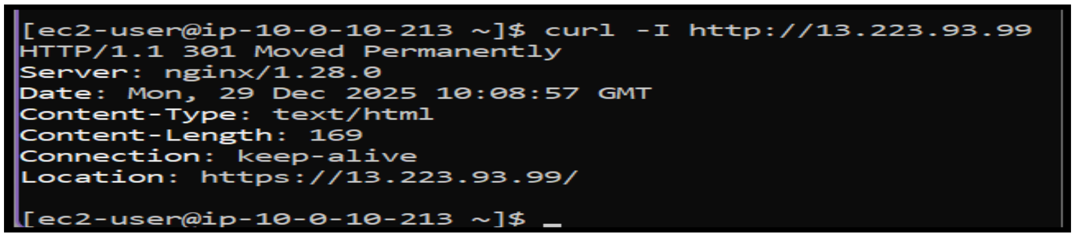
```
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1327 bytes and written 386 bytes
Verification error: self-signed certificate
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 2048 bit
This TLS version forbids renegotiation.
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 18 (self-signed certificate)
---
---
```

- **security headers:**



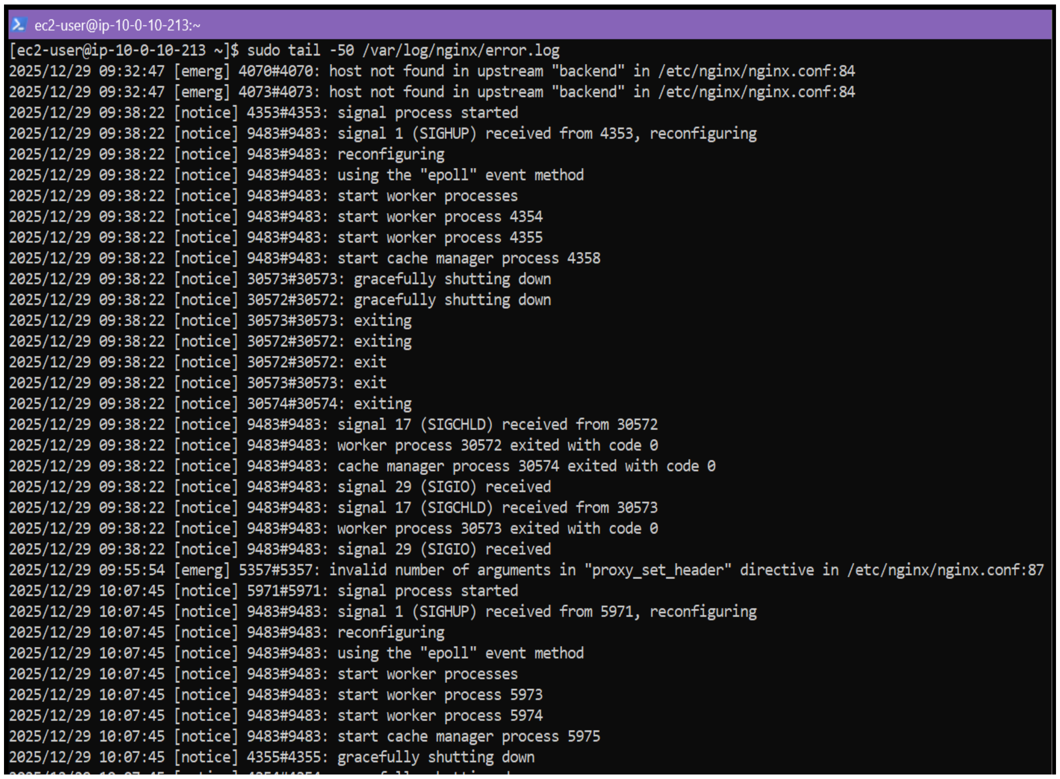
```
[ec2-user@ip-10-0-10-213 ~]$ sudo nano +84 /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-213 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-213 ~]$ sudo systemctl reload nginx
[ec2-user@ip-10-0-10-213 ~]$ curl -I -k https://13.223.93.99
HTTP/2 200
server: nginx/1.28.0
date: Mon, 29 Dec 2025 09:38:32 GMT
content-type: text/html
content-length: 615
last-modified: Tue, 12 Aug 2025 21:30:50 GMT
vary: Accept-Encoding
etag: "689bb28a-267"
strict-transport-security: max-age=31536000; includeSubDomains
x-frame-options: DENY
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
content-security-policy: default-src 'self'
strict-transport-security: max-age=31536000; includeSubDomains
x-frame-options: DENY
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
content-security-policy: default-src 'self'
accept-ranges: bytes

[ec2-user@ip-10-0-10-213 ~]$ _
```

- **http redirect:**



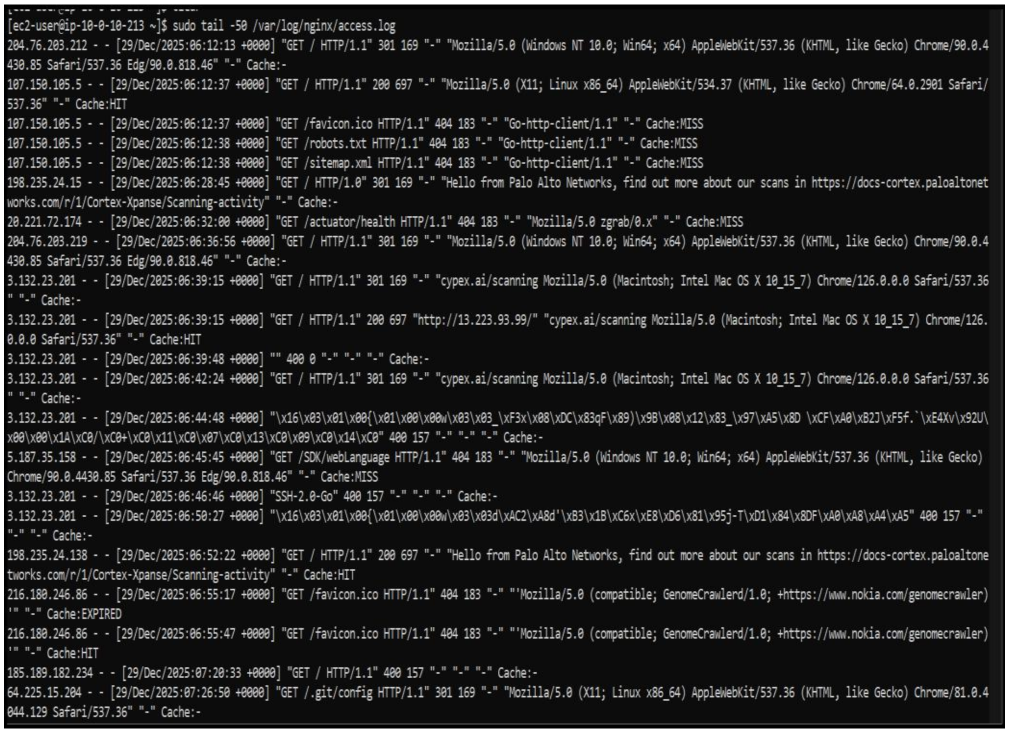
```
[ec2-user@ip-10-0-10-213 ~]$ curl -I http://13.223.93.99
HTTP/1.1 301 Moved Permanently
Server: nginx/1.28.0
Date: Mon, 29 Dec 2025 10:08:57 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://13.223.93.99/

[ec2-user@ip-10-0-10-213 ~]$ _
```

- **error log analysis:**



```
ec2-user@ip-10-0-10-213:~
[ec2-user@ip-10-0-10-213 ~]$ sudo tail -50 /var/log/nginx/error.log
2025/12/29 09:32:47 [emerg] 4070#4070: host not found in upstream "backend" in /etc/nginx/nginx.conf:84
2025/12/29 09:32:47 [emerg] 4073#4073: host not found in upstream "backend" in /etc/nginx/nginx.conf:84
2025/12/29 09:38:22 [notice] 4353#4353: signal process started
2025/12/29 09:38:22 [notice] 9483#9483: signal 1 (SIGHUP) received from 4353, reconfiguring
2025/12/29 09:38:22 [notice] 9483#9483: reconfiguring
2025/12/29 09:38:22 [notice] 9483#9483: using the "epoll" event method
2025/12/29 09:38:22 [notice] 9483#9483: start worker processes
2025/12/29 09:38:22 [notice] 9483#9483: start worker process 4354
2025/12/29 09:38:22 [notice] 9483#9483: start worker process 4355
2025/12/29 09:38:22 [notice] 9483#9483: start cache manager process 4358
2025/12/29 09:38:22 [notice] 30573#30573: gracefully shutting down
2025/12/29 09:38:22 [notice] 30572#30572: gracefully shutting down
2025/12/29 09:38:22 [notice] 30573#30573: exiting
2025/12/29 09:38:22 [notice] 30572#30572: exiting
2025/12/29 09:38:22 [notice] 30572#30572: exit
2025/12/29 09:38:22 [notice] 30573#30573: exit
2025/12/29 09:38:22 [notice] 30574#30574: exiting
2025/12/29 09:38:22 [notice] 9483#9483: signal 17 (SIGCHLD) received from 30572
2025/12/29 09:38:22 [notice] 9483#9483: worker process 30572 exited with code 0
2025/12/29 09:38:22 [notice] 9483#9483: cache manager process 30574 exited with code 0
2025/12/29 09:38:22 [notice] 9483#9483: signal 29 (SIGIO) received
2025/12/29 09:38:22 [notice] 9483#9483: signal 17 (SIGCHLD) received from 30573
2025/12/29 09:38:22 [notice] 9483#9483: worker process 30573 exited with code 0
2025/12/29 09:38:22 [notice] 9483#9483: signal 29 (SIGIO) received
2025/12/29 09:55:54 [emerg] 5357#5357: invalid number of arguments in "proxy_set_header" directive in /etc/nginx/nginx.conf:87
2025/12/29 10:07:45 [notice] 5971#5971: signal process started
2025/12/29 10:07:45 [notice] 9483#9483: signal 1 (SIGHUP) received from 5971, reconfiguring
2025/12/29 10:07:45 [notice] 9483#9483: reconfiguring
2025/12/29 10:07:45 [notice] 9483#9483: using the "epoll" event method
2025/12/29 10:07:45 [notice] 9483#9483: start worker processes
2025/12/29 10:07:45 [notice] 9483#9483: start worker process 5973
2025/12/29 10:07:45 [notice] 9483#9483: start worker process 5974
2025/12/29 10:07:45 [notice] 9483#9483: start cache manager process 5975
2025/12/29 10:07:45 [notice] 4355#4355: gracefully shutting down
```

- **access log analysis:**

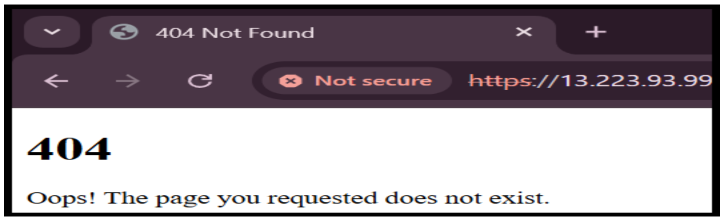


```
[ec2-user@ip-10-0-10-213 ~]$ sudo tail -50 /var/log/nginx/access.log
204.76.203.212 - - [29/Dec/2025:06:12:13 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4
430.85 Safari/537.36 Edg/90.0.818.46" "-" Cache:-
107.150.105.5 - - [29/Dec/2025:06:12:37 +0000] "GET / HTTP/1.1" 200 697 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/534.37 (KHTML, like Gecko) Chrome/64.0.2901 Safari/
537.36" "-" Cache:HIT
107.150.105.5 - - [29/Dec/2025:06:12:37 +0000] "GET /favicon.ico HTTP/1.1" 404 183 "-" "Go-http-client/1.1" "-" Cache:MISS
107.150.105.5 - - [29/Dec/2025:06:12:38 +0000] "GET /robots.txt HTTP/1.1" 404 183 "-" "Go-http-client/1.1" "-" Cache:MISS
107.150.105.5 - - [29/Dec/2025:06:12:38 +0000] "GET /sitemap.xml HTTP/1.1" 404 183 "-" "Go-http-client/1.1" "-" Cache:MISS
198.235.24.15 - - [29/Dec/2025:06:28:45 +0000] "GET / HTTP/1.0" 301 169 "-" "Hello from Palo Alto Networks, find out more about our scans in https://docs-cortex.paloaltonet
works.com/r/1/Cortex-Xpanse/Scanning-activity" "-" Cache:-
20.221.72.174 - - [29/Dec/2025:06:32:00 +0000] "GET /actuator/health HTTP/1.1" 404 183 "-" "Mozilla/5.0 zgrab/0.x" "-" Cache:MISS
204.76.203.219 - - [29/Dec/2025:06:36:56 +0000] "GET / HTTP/1.1" 301 169 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4
430.85 Safari/537.36 Edg/90.0.818.46" "-" Cache:-
3.132.23.201 - - [29/Dec/2025:06:39:15 +0000] "GET / HTTP/1.1" 301 169 "-" "cypex.ai/scanning Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/126.0.0.0 Safari/537.36
" "-" Cache:-
3.132.23.201 - - [29/Dec/2025:06:39:15 +0000] "GET / HTTP/1.1" 200 697 "http://13.223.93.99/" "cypex.ai/scanning Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/126.
0.0.0 Safari/537.36" "-" Cache:HIT
3.132.23.201 - - [29/Dec/2025:06:39:48 +0000] "" 400 0 "-" "-" "-" Cache:-
3.132.23.201 - - [29/Dec/2025:06:42:24 +0000] "GET / HTTP/1.1" 301 169 "-" "cypex.ai/scanning Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Chrome/126.0.0.0 Safari/537.36
" "-" Cache:-
3.132.23.201 - - [29/Dec/2025:06:44:48 +0000] "\x16\x03\x01\x00{\x01\x00\x00w\x03\x03_\xF3x\x08\xDC\x83qF\x89)\x9B\x08\x12\x83_\x97\xA5\x8D \xCF\xA0\xB2J\xF5f.`\xE4Xv\x92U\
x00\x00\x1A\xC0/\xC0+\xC0\x11\xC0\x07\xC0\x13\xC0\x09\xC0\x14\xC0" 400 157 "-" "-" "-" Cache:-
5.187.35.158 - - [29/Dec/2025:06:45:45 +0000] "GET /SDK/webLanguage HTTP/1.1" 404 183 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/90.0.4430.85 Safari/537.36 Edg/90.0.818.46" "-" Cache:MISS
3.132.23.201 - - [29/Dec/2025:06:46:46 +0000] "SSH-2.0-Go" 400 157 "-" "-" "-" Cache:-
3.132.23.201 - - [29/Dec/2025:06:50:27 +0000] "\x16\x03\x01\x00{\x01\x00\x00w\x03\x03d\xAC2\xA8d'\xB3\x1B\xC6x\xE8\xD6\x81\x95j-T\xD1\x84\x8DF\xA0\xA8\xA4\xA5" 400 157 "-"
"-" "-" Cache:-
198.235.24.138 - - [29/Dec/2025:06:52:22 +0000] "GET / HTTP/1.1" 200 697 "-" "Hello from Palo Alto Networks, find out more about our scans in https://docs-cortex.paloaltone
tworks.com/r/1/Cortex-Xpanse/Scanning-activity" "-" Cache:HIT
216.180.246.86 - - [29/Dec/2025:06:55:17 +0000] "GET /favicon.ico HTTP/1.1" 404 183 "-" "'Mozilla/5.0 (compatible; GenomeCrawlerd/1.0; +https://www.nokia.com/genomecrawler)
'" "-" Cache:EXPIRED
216.180.246.86 - - [29/Dec/2025:06:55:47 +0000] "GET /favicon.ico HTTP/1.1" 404 183 "-" "'Mozilla/5.0 (compatible; GenomeCrawlerd/1.0; +https://www.nokia.com/genomecrawler)
'" "-" Cache:HIT
185.189.182.234 - - [29/Dec/2025:07:20:33 +0000] "GET / HTTP/1.1" 400 157 "-" "-" "-" Cache:-
64.225.15.204 - - [29/Dec/2025:07:26:50 +0000] "GET /.git/config HTTP/1.1" 301 169 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4
044.129 Safari/537.36" "-" Cache:-
```

Bonus Tasks

Bonus 1: Custom Error Pages

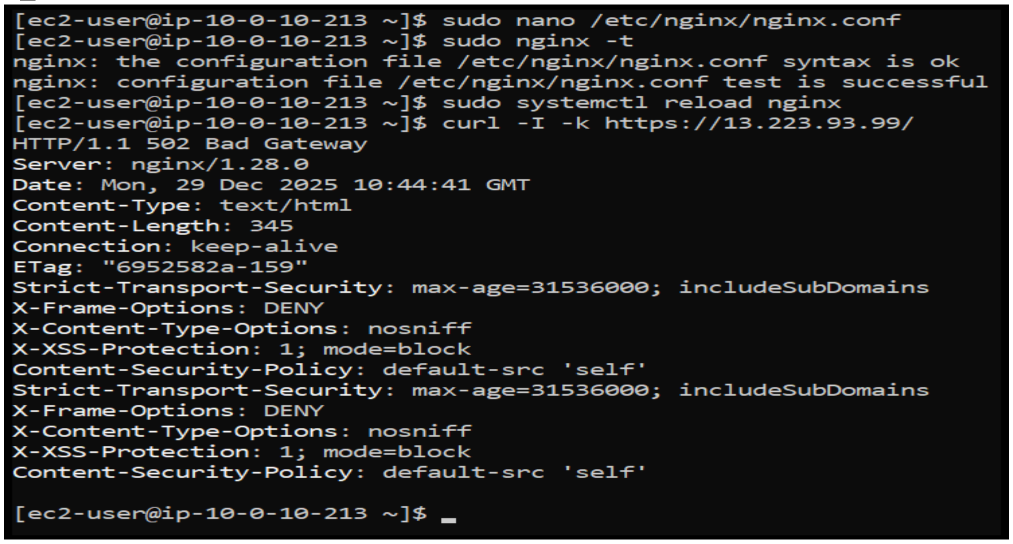
- **custom_404:**



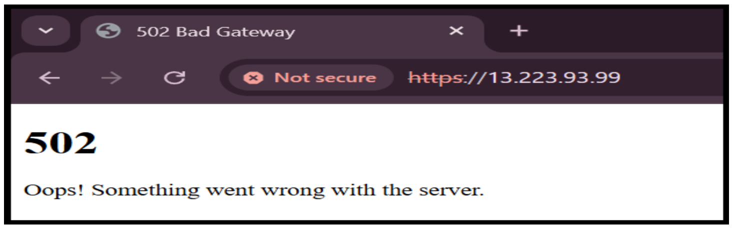
404

Oops! The page you requested does not exist.

- **custom_502:**



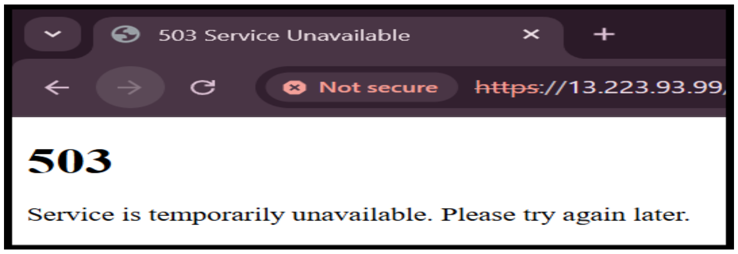
```
[ec2-user@ip-10-0-10-213 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-10-213 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-213 ~]$ sudo systemctl reload nginx
[ec2-user@ip-10-0-10-213 ~]$ curl -I -k https://13.223.93.99/
HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Mon, 29 Dec 2025 10:44:41 GMT
Content-Type: text/html
Content-Length: 345
Connection: keep-alive
ETag: "6952582a-159"
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'

[ec2-user@ip-10-0-10-213 ~]$ _
```



502

Oops! Something went wrong with the server.

- **custom_503:**

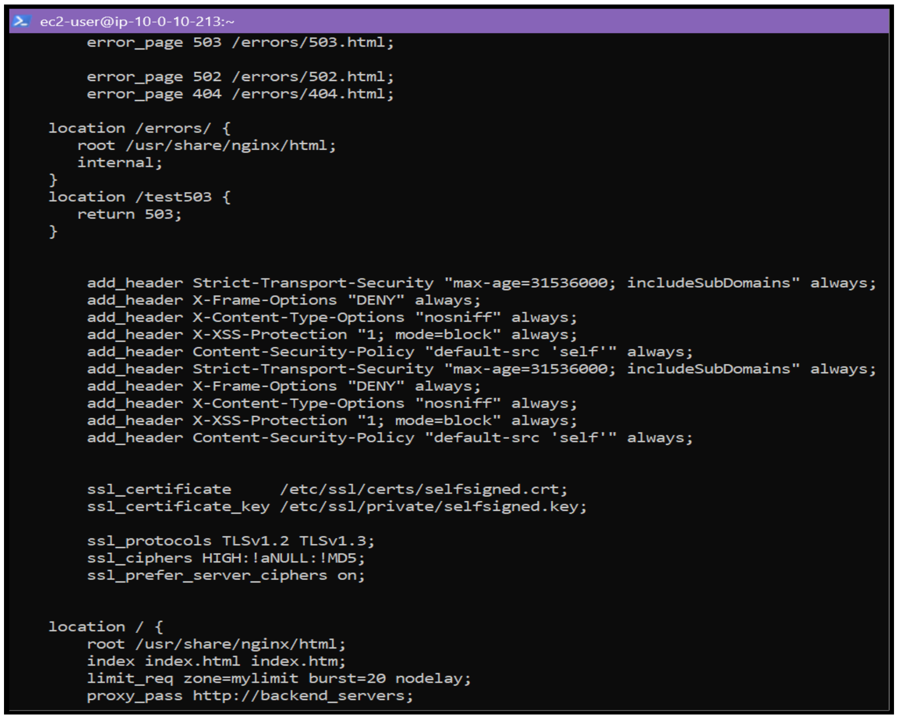


503

Service is temporarily unavailable. Please try again later.

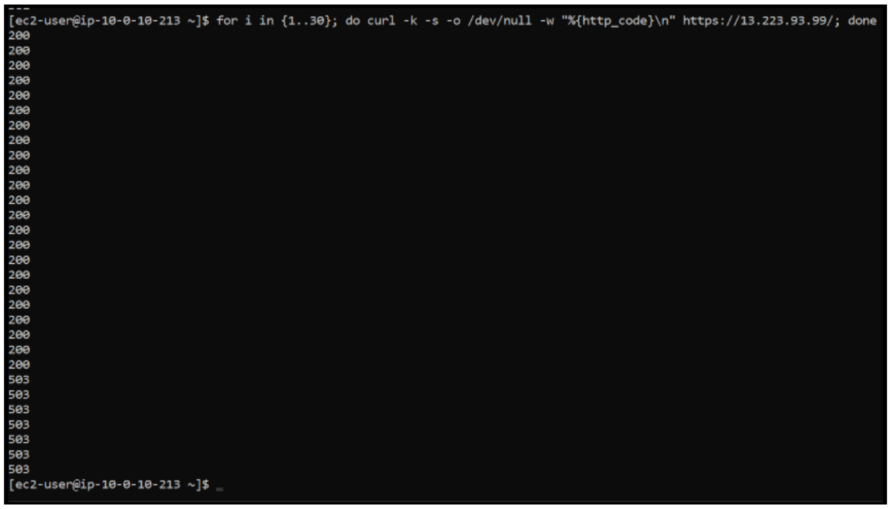
Bonus 2: Implement Rate Limiting

- **rate_limit_config:**



```
ec2-user@ip-10-0-10-213:~
        error_page 503 /errors/503.html;

        error_page 502 /errors/502.html;
        error_page 404 /errors/404.html;

    location /errors/ {
       root /usr/share/nginx/html;
       internal;
    }
    location /test503 {
       return 503;
    }


        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
        add_header X-Frame-Options "DENY" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-XSS-Protection "1; mode=block" always;
        add_header Content-Security-Policy "default-src 'self'" always;
        add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
        add_header X-Frame-Options "DENY" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-XSS-Protection "1; mode=block" always;
        add_header Content-Security-Policy "default-src 'self'" always;


        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;


    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        limit_req zone=mylimit burst=20 nodelay;
        proxy_pass http://backend_servers;
```

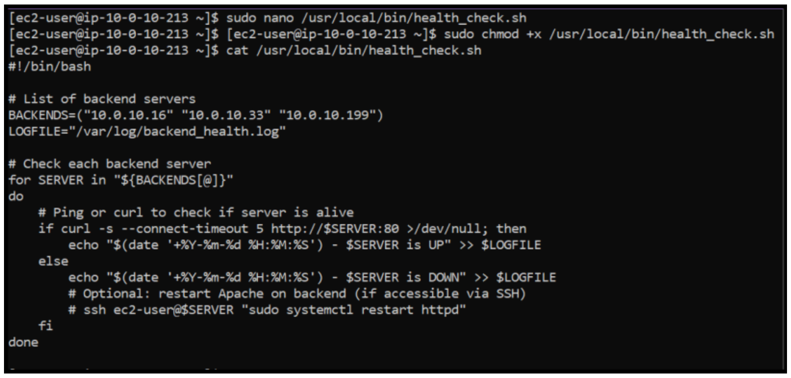
- **rate_limit_test**



```
---
[ec2-user@ip-10-0-10-213 ~]$ for i in {1..30}; do curl -k -s -o /dev/null -w "%{http_code}\n" https://13.223.93.99/; done
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
200
503
503
503
503
503
503
503
[ec2-user@ip-10-0-10-213 ~]$ _
```

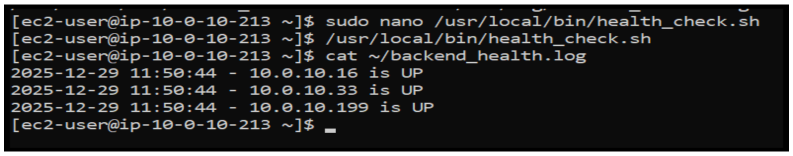
Bonus 3: Health Check Automation

- **health_check_script:**



```
[ec2-user@ip-10-0-10-213 ~]$ sudo nano /usr/local/bin/health_check.sh
[ec2-user@ip-10-0-10-213 ~]$ [ec2-user@ip-10-0-10-213 ~]$ sudo chmod +x /usr/local/bin/health_check.sh
[ec2-user@ip-10-0-10-213 ~]$ cat /usr/local/bin/health_check.sh
#!/bin/bash

# List of backend servers
BACKENDS=("10.0.10.16" "10.0.10.33" "10.0.10.199")
LOGFILE="/var/log/backend_health.log"

# Check each backend server
for SERVER in "${BACKENDS[@]}"
do
    # Ping or curl to check if server is alive
    if curl -s --connect-timeout 5 http://$SERVER:80 >/dev/null; then
        echo "$(date '+%Y-%m-%d %H:%M:%S') - $SERVER is UP" >> $LOGFILE
    else
        echo "$(date '+%Y-%m-%d %H:%M:%S') - $SERVER is DOWN" >> $LOGFILE
        # Optional: restart Apache on backend (if accessible via SSH)
        # ssh ec2-user@$SERVER "sudo systemctl restart httpd"
    fi
done
```

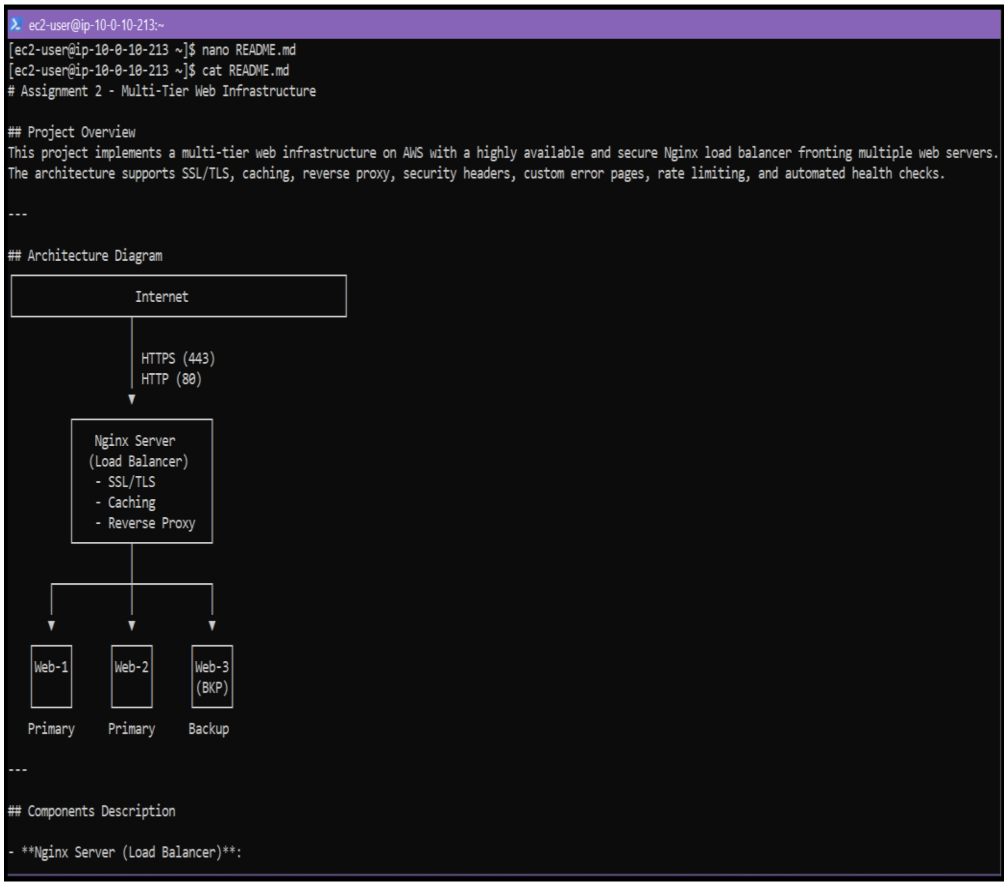
- **health_log:**



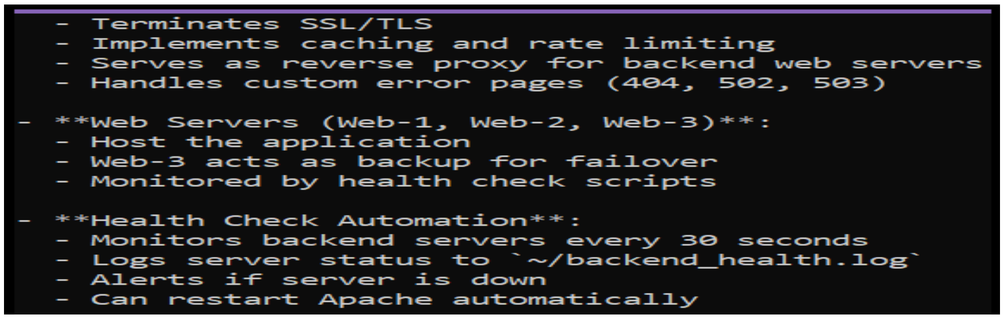
```
[ec2-user@ip-10-0-10-213 ~]$ sudo nano /usr/local/bin/health_check.sh
[ec2-user@ip-10-0-10-213 ~]$ /usr/local/bin/health_check.sh
[ec2-user@ip-10-0-10-213 ~]$ cat ~/backend_health.log
2025-12-29 11:50:44 - 10.0.10.16 is UP
2025-12-29 11:50:44 - 10.0.10.33 is UP
2025-12-29 11:50:44 - 10.0.10.199 is UP
[ec2-user@ip-10-0-10-213 ~]$
```

Part 6 : Documentation & Cleanup

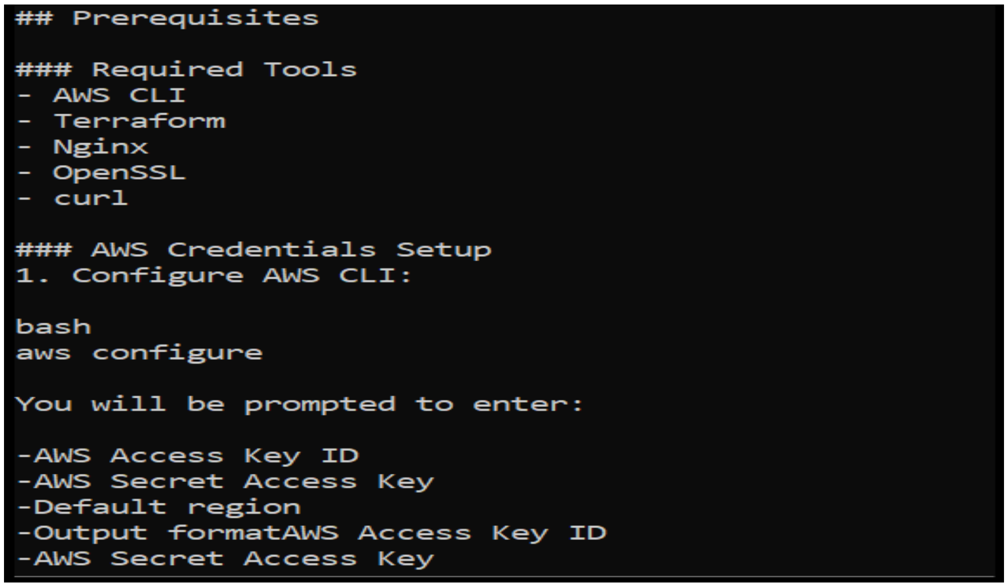
6.1 README Documentation



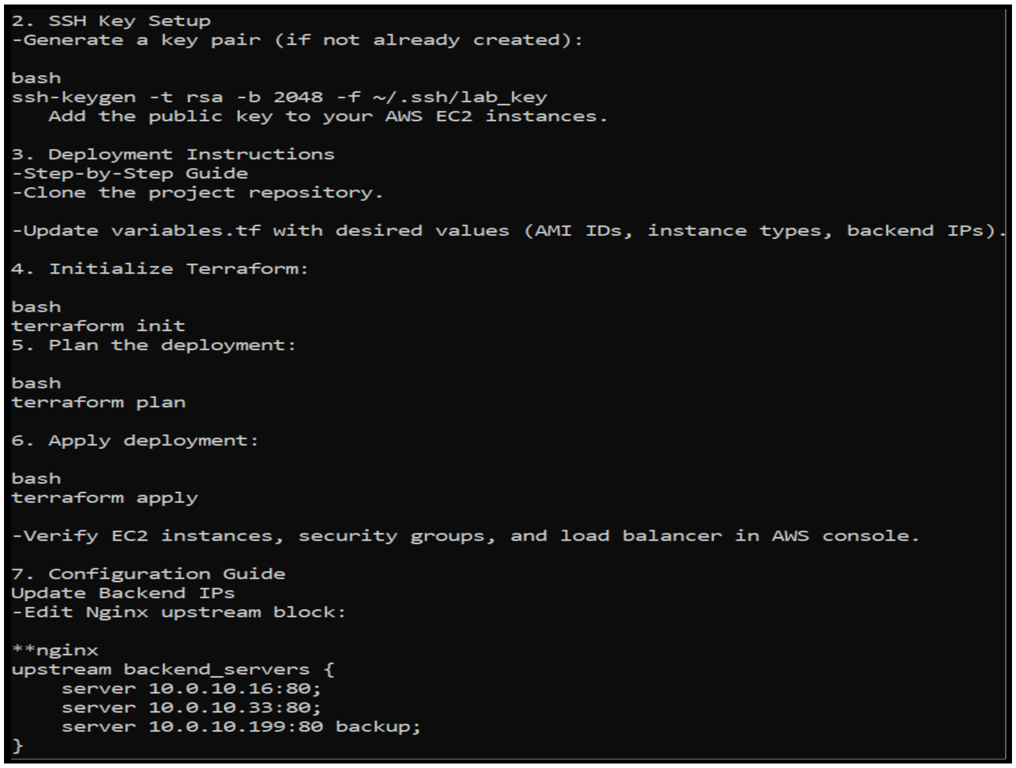
```
[ec2-user@ip-10-0-10-213 ~]$ nano README.md
[ec2-user@ip-10-0-10-213 ~]$ cat README.md
# Assignment 2 - Multi-Tier Web Infrastructure

## Project Overview
This project implements a multi-tier web infrastructure on AWS with a highly available and secure Nginx load balancer fronting multiple web servers.
The architecture supports SSL/TLS, caching, reverse proxy, security headers, custom error pages, rate limiting, and automated health checks.

---

## Architecture Diagram
┌─────────────────────────────────┐
│             Internet            │
└─────────────────────────────────┘
             │
             │ HTTPS (443)
             │ HTTP (80)
             ▼
       ┌──────────────┐
       │ Nginx Server │
       │(Load Balancer)│
       │ - SSL/TLS    │
       │ - Caching    │
       │ - Reverse Proxy│
       └──────────────┘
             │
     ┌───────┼───────┐
     ▼       ▼       ▼
  ┌─────┐ ┌─────┐ ┌─────┐
  │Web-1│ │Web-2│ │Web-3│
  │     │ │     │ │(BKP)│
  └─────┘ └─────┘ └─────┘
  Primary Primary Backup

---

## Components Description

- **Nginx Server (Load Balancer)**:
```



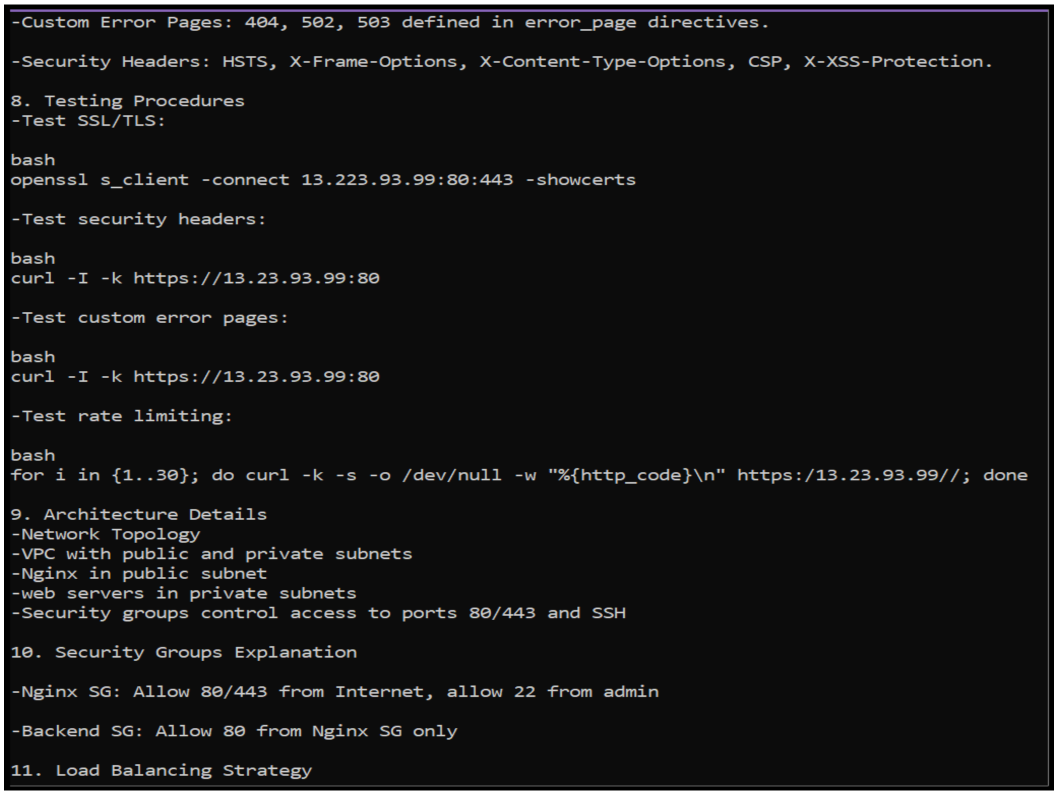
```
   - Terminates SSL/TLS
   - Implements caching and rate limiting
   - Serves as reverse proxy for backend web servers
   - Handles custom error pages (404, 502, 503)

- **Web Servers (Web-1, Web-2, Web-3)**:
   - Host the application
   - Web-3 acts as backup for failover
   - Monitored by health check scripts

- **Health Check Automation**:
   - Monitors backend servers every 30 seconds
   - Logs server status to `~/backend_health.log`
   - Alerts if server is down
   - Can restart Apache automatically
```



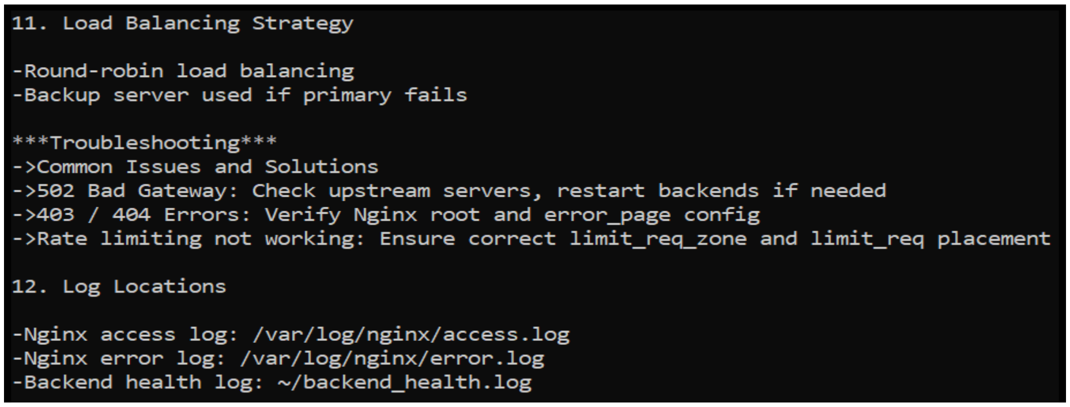
```
## Prerequisites

### Required Tools
- AWS CLI
- Terraform
- Nginx
- OpenSSL
- curl

### AWS Credentials Setup
1. Configure AWS CLI:

bash
aws configure

You will be prompted to enter:

-AWS Access Key ID
-AWS Secret Access Key
-Default region
-Output formatAWS Access Key ID
-AWS Secret Access Key
```



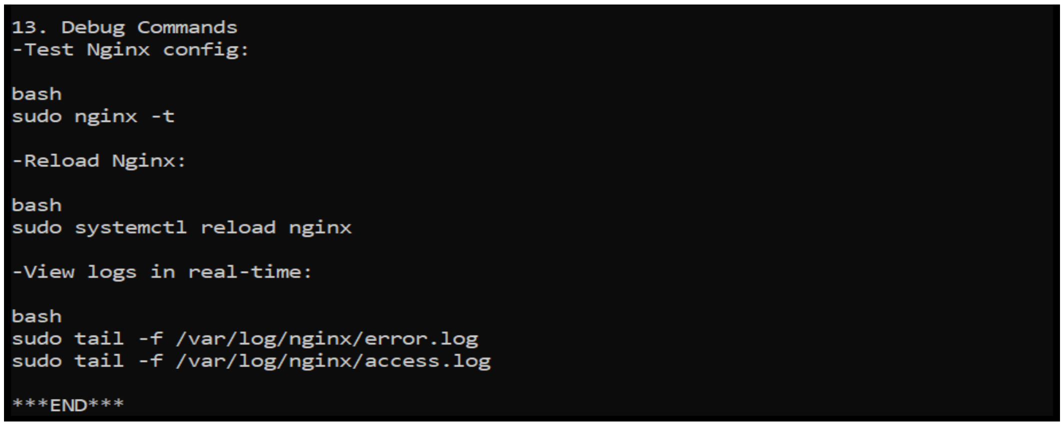
```
2. SSH Key Setup
-Generate a key pair (if not already created):

bash
ssh-keygen -t rsa -b 2048 -f ~/.ssh/lab_key
   Add the public key to your AWS EC2 instances.

3. Deployment Instructions
-Step-by-Step Guide
-Clone the project repository.

-Update variables.tf with desired values (AMI IDs, instance types, backend IPs).

4. Initialize Terraform:

bash
terraform init
5. Plan the deployment:

bash
terraform plan

6. Apply deployment:

bash
terraform apply

-Verify EC2 instances, security groups, and load balancer in AWS console.

7. Configuration Guide
Update Backend IPs
-Edit Nginx upstream block:

**nginx
upstream backend_servers {
    server 10.0.10.16:80;
    server 10.0.10.33:80;
    server 10.0.10.199:80 backup;
}
```



```
-Custom Error Pages: 404, 502, 503 defined in error_page directives.

-Security Headers: HSTS, X-Frame-Options, X-Content-Type-Options, CSP, X-XSS-Protection.

8. Testing Procedures
-Test SSL/TLS:

bash
openssl s_client -connect 13.223.93.99:80:443 -showcerts

-Test security headers:

bash
curl -I -k https://13.23.93.99:80

-Test custom error pages:

bash
curl -I -k https://13.23.93.99:80

-Test rate limiting:

bash
for i in {1..30}; do curl -k -s -o /dev/null -w "%{http_code}\n" https:/13.23.93.99//; done

9. Architecture Details
-Network Topology
-VPC with public and private subnets
-Nginx in public subnet
-web servers in private subnets
-Security groups control access to ports 80/443 and SSH

10. Security Groups Explanation

-Nginx SG: Allow 80/443 from Internet, allow 22 from admin

-Backend SG: Allow 80 from Nginx SG only

11. Load Balancing Strategy
```



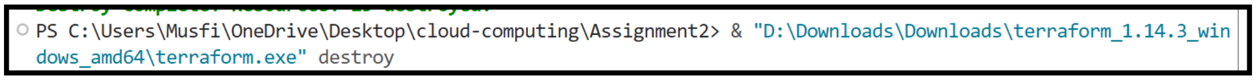
```
11. Load Balancing Strategy

-Round-robin load balancing
-Backup server used if primary fails

***Troubleshooting***
->Common Issues and Solutions
->502 Bad Gateway: Check upstream servers, restart backends if needed
->403 / 404 Errors: Verify Nginx root and error_page config
->Rate limiting not working: Ensure correct limit_req_zone and limit_req placement

12. Log Locations

-Nginx access log: /var/log/nginx/access.log
-Nginx error log: /var/log/nginx/error.log
-Backend health log: ~/backend_health.log
```



```
13. Debug Commands
-Test Nginx config:

bash
sudo nginx -t

-Reload Nginx:

bash
sudo systemctl reload nginx

-View logs in real-time:

bash
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log

***END***
```

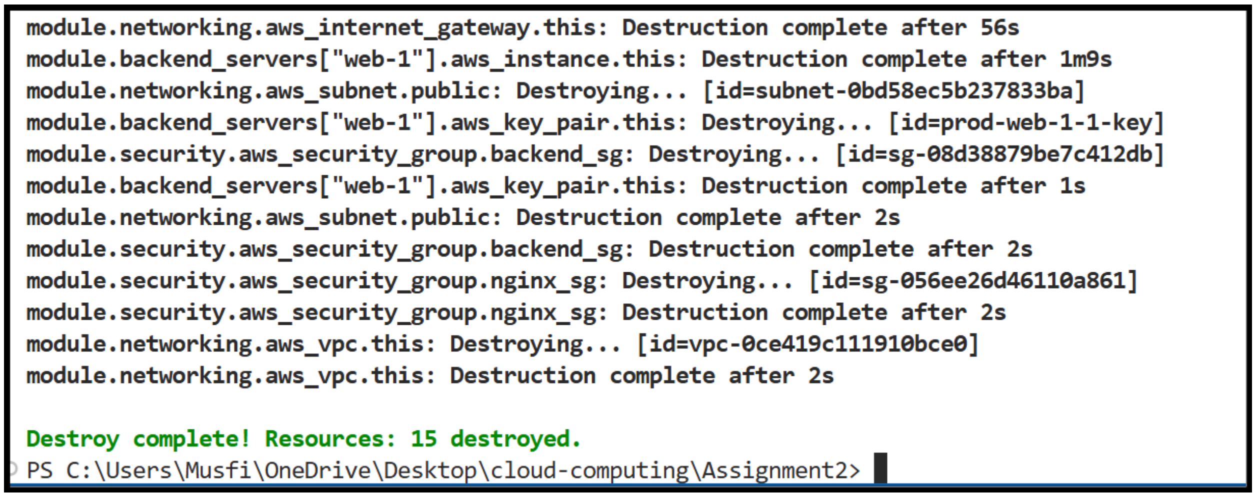
6.2 Infrastructure Cleanup

- **terraform_destroy_prompt:**

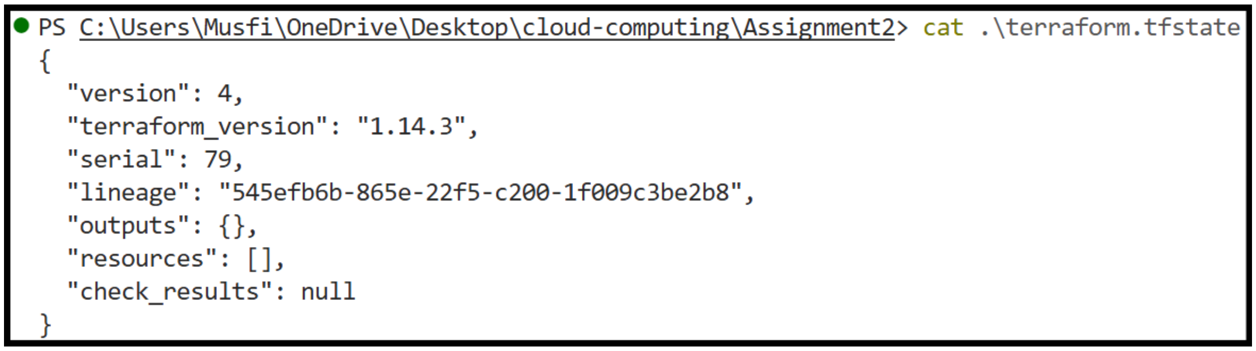


```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> & "D:\Downloads\Downloads\terraform_1.14.3_windows_amd64\terraform.exe" destroy
```

- **terraform_destroy_complete:**

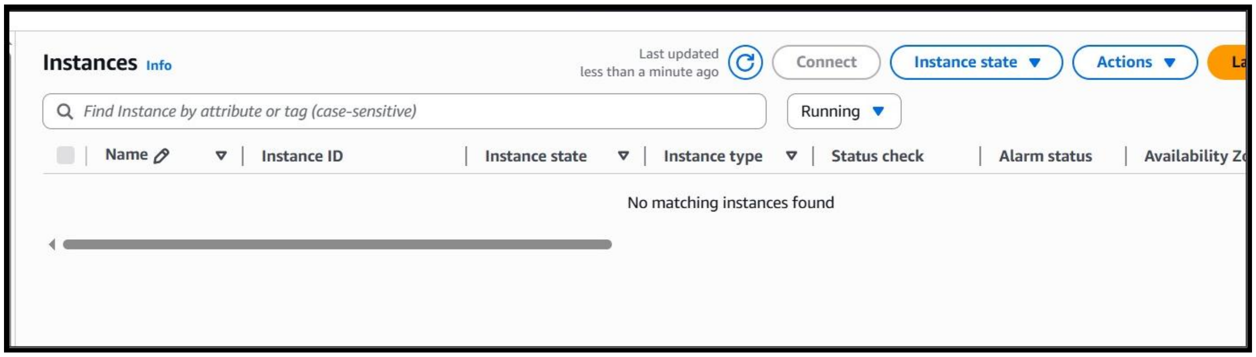


```
module.networking.aws_internet_gateway.this: Destruction complete after 56s
module.backend_servers["web-1"].aws_instance.this: Destruction complete after 1m9s
module.networking.aws_subnet.public: Destroying... [id=subnet-0bd58ec5b237833ba]
module.backend_servers["web-1"].aws_key_pair.this: Destroying... [id=prod-web-1-1-key]
module.security.aws_security_group.backend_sg: Destroying... [id=sg-08d38879be7c412db]
module.backend_servers["web-1"].aws_key_pair.this: Destruction complete after 1s
module.networking.aws_subnet.public: Destruction complete after 2s
module.security.aws_security_group.backend_sg: Destruction complete after 2s
module.security.aws_security_group.nginx_sg: Destroying... [id=sg-056ee26d46110a861]
module.security.aws_security_group.nginx_sg: Destruction complete after 2s
module.networking.aws_vpc.this: Destroying... [id=vpc-0ce419c111910bce0]
module.networking.aws_vpc.this: Destruction complete after 2s

Destroy complete! Resources: 15 destroyed.
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2>
```



```
PS C:\Users\Musfi\OneDrive\Desktop\cloud-computing\Assignment2> cat .\terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.14.3",
  "serial": 79,
  "lineage": "545efb6b-865e-22f5-c200-1f009c3be2b8",
  "outputs": {},
  "resources": [],
  "check_results": null
}
```

- **aws_instances_destroyed:**
-



No matching instances found

4. Testing Results

4.1 Load Balancing Validation

Traffic was successfully shared across multiple backend servers. Server responses and logs confirmed that requests were rotating correctly between available instances.

4.2 Caching Efficiency

Nginx caching improved performance by serving repeated requests faster and reducing the number of requests sent to backend servers.

4.3 High Availability Verification

When one or more primary web servers were stopped, the system continued working without interruption by forwarding traffic to the backup server.

4.4 Security Verification

- HTTPS connections were enforced
- Security headers were properly applied
- Direct access to backend servers was restricted

5. Challenges & Solutions

Challenges Faced

- Managing Terraform state files correctly
- Uploading large numbers of screenshots to GitHub
- Writing and debugging complex Nginx configurations
- Configuring AWS security groups accurately

How These Issues Were Solved

- Used structured and modular Terraform files
- Excluded unnecessary files using .gitignore
- Tested Nginx configurations step-by-step before deployment
- Carefully reviewed and validated security group rules

Key Lessons Learned

- Automation simplifies infrastructure management
- Modular design improves clarity and reusability
- Security planning is critical in cloud deployments

6. Conclusion

This assignment successfully demonstrated the deployment of a secure, scalable, and highly available cloud infrastructure using Terraform and Nginx. Practical experience was gained in cloud networking, infrastructure automation, load balancing, and system administration.

7. Appendices

Appendix A: Code Listings

- All Terraform configuration files (main.tf, variables.tf, outputs.tf, locals.tf)
- All module files (networking/, security/, webserver/)
- Server scripts (nginx-setup.sh, apache-setup.sh)

Appendix B: Configuration Files

- Example terraform.tfvars file
- .gitignore file

Appendix C: Additional Screenshots

- Screenshots for each part of the assignment.
- Screenshots of Nginx configuration, SSL certificates, logs, and Terraform outputs