CLOUD COMPUTING

ASSIGNMENT: 02

Submitted To:

Sir Waqas Saleem

Submitted By:

Komal Kashif

BSE V-A

2023-BSE-031

Submission Date: December 30, 2025

1) EXECUTIVE SUMMARY

This assignment focuses on designing and deploying a secure, high-availability multi-tier web infrastructure on Amazon Web Services (AWS) using Terraform for Infrastructure as Code and Nginx as a reverse proxy and load balancer.

The main goal was to show practical understanding of cloud automation, Terraform modules, Nginx configuration, and high-availability web architecture. The entire infrastructure was deployed automatically using reusable Terraform modules, making it scalable, consistent, and easy to manage.

Infrastructure Overview

The system follows a three-tier architecture:

- **Networking Layer:** A custom VPC with a public subnet, Internet Gateway, and routing for external access.
- **Security Layer:** Separate security groups for Nginx and backend servers, restricted SSH access, and backend servers accessible only through Nginx.
- **Compute Layer:**
  - One Nginx server acting as a secure reverse proxy and load balancer with HTTPS, caching, security headers, and failover support.
  - Three Apache backend servers displaying dynamic system information.

All EC2 instances were created using a reusable Terraform module to maintain consistency.

Key Achievements

- Implemented Infrastructure as Code using Terraform modules.
- Built a high-availability setup with load balancing and backup servers.
- Configured Nginx with SSL, caching, compression, and security features.
- Tested failover and verified caching behavior.
- Applied security best practices such as HTTPS enforcement and restricted access.

Overall, this assignment demonstrates a complete DevOps workflow, from infrastructure design and automation to deployment and testing, following modern cloud and DevOps best practices.

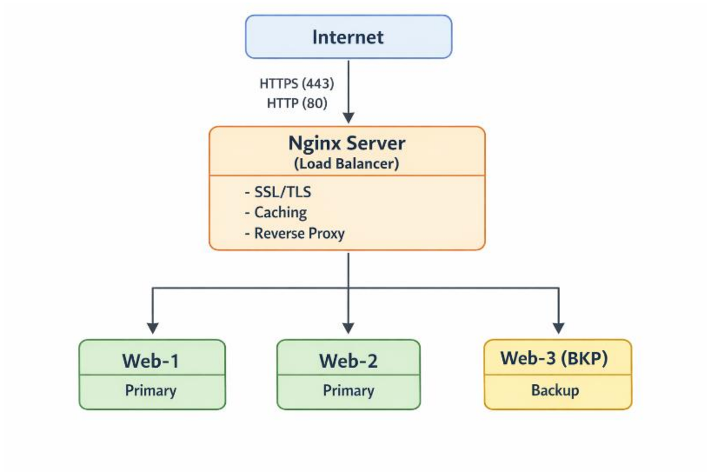
2) ARCHITECTURE DESIGN

The architecture for this assignment is a multi-tier web application deployed on Amazon Web Services (AWS) using Terraform for automation and Nginx as a reverse proxy and load balancer. The main goal of the design is to ensure security, availability, and scalability while following cloud best practices demonstrated in class.

The architecture is divided into the following layers:

- Client layer (users accessing the application)
- Load balancing layer (Nginx server)
- Application layer (Apache backend servers)
- Network and security layer (VPC, subnet, routing, security groups)

Architecture Diagram



Component Descriptions

Nginx Server:

The Nginx server acts as the main entry point for users. It handles HTTPS connections, redirects HTTP traffic to HTTPS, and distributes requests across multiple backend servers. It also provides basic caching and adds security headers to incoming responses.

Backend Web Servers:

Three Apache web servers are deployed as backend servers:

- Two primary servers (web-1 and web-2)
- One backup server (web-3)

These servers host a simple web page and are only accessible through the Nginx server, not directly from the internet.

Network Topology

A custom VPC is created to isolate the infrastructure. A single public subnet is used to host all EC2 instances. An Internet Gateway is attached to the VPC to allow external access, and routing is configured so that only the Nginx server is exposed to the public internet.

Security Design

Security is implemented using AWS security groups:

- The Nginx security group allows HTTP (80) and HTTPS (443) from anywhere, and SSH access only from the administrator's IP address.
- The backend security group allows HTTP traffic only from the Nginx server and restricts SSH access.

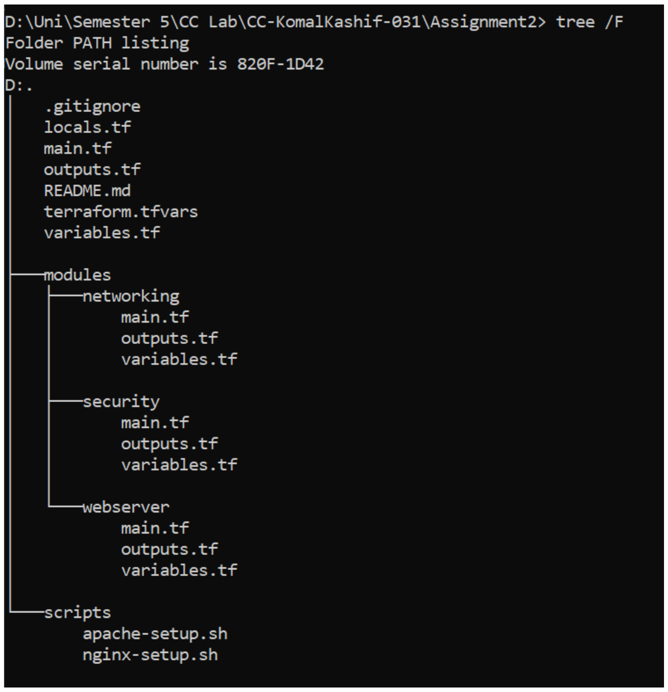
This setup ensures that backend servers remain protected and cannot be accessed directly by users.

3) IMPLEMENTATION DETAILS

- **Part 1: Infrastructure setup**

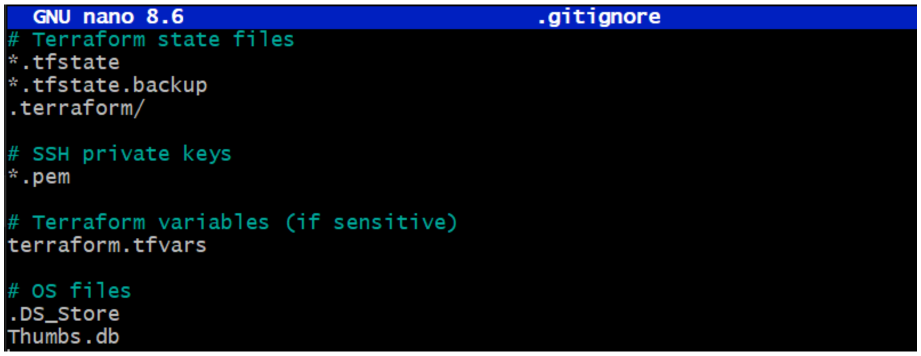
○ **1.1 Project Setup**

All the necessary commands were run for the proper setup of the project. The screenshot below shows the structure of project:



```
D:\Uni\Semester 5\CC Lab\CC-KomalKashif-031\Assignment2> tree /F
Folder PATH listing
Volume serial number is 820F-1D42
D:.
│   .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
```

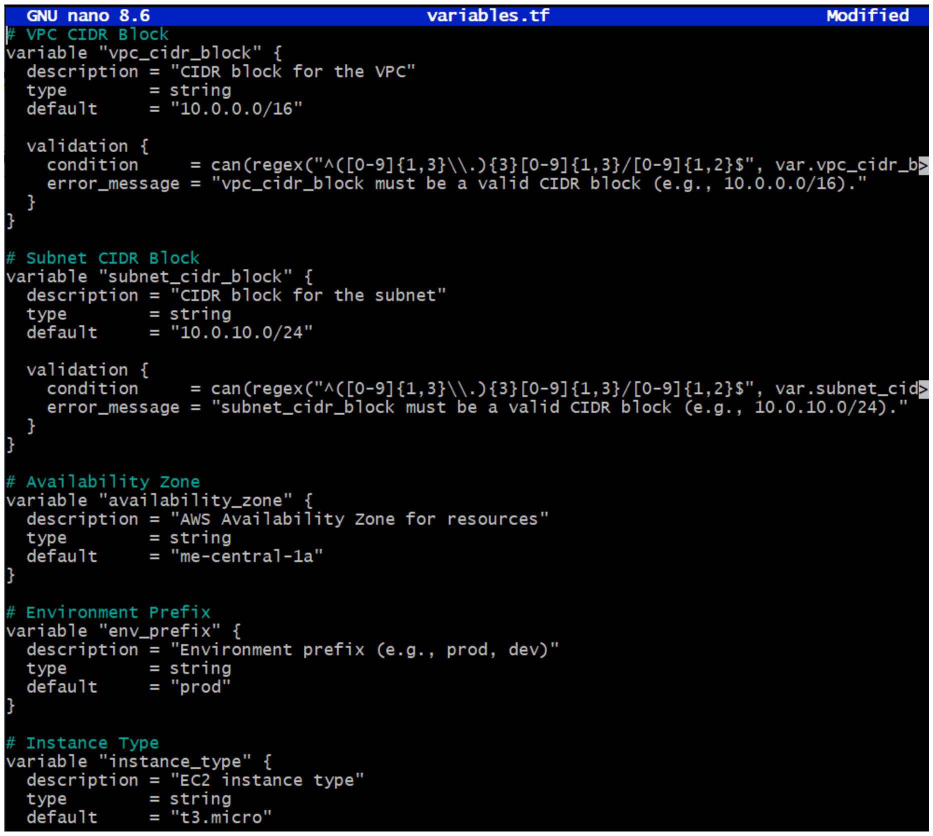
Contents of .gitignore:



```
GNU nano 8.6                              .gitignore
# Terraform state files
*.tfstate
*.tfstate.backup
.terraform/

# SSH private keys
*.pem

# Terraform variables (if sensitive)
terraform.tfvars

# OS files
.DS_Store
Thumbs.db
```

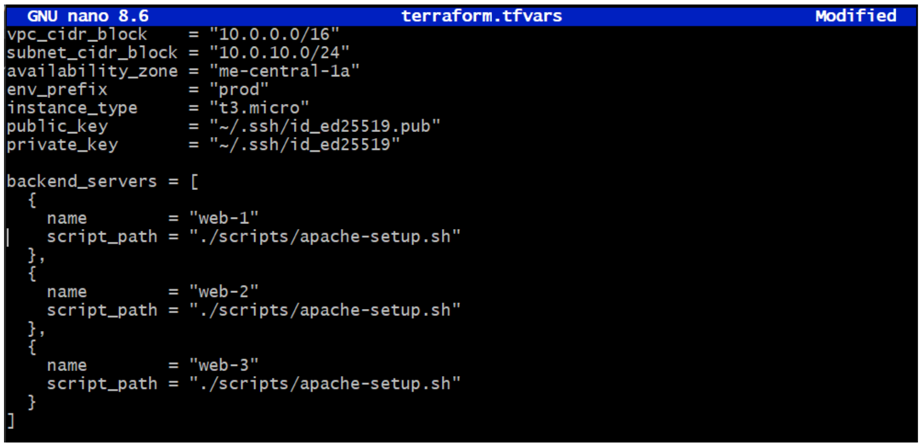
- 1.2 Variable Configuration

Task: Add validation rules for CIDR blocks. Add descriptions for all variables. Set appropriate defaults where applicable.



```
GNU nano 8.6                        variables.tf                        Modified
# VPC CIDR Block
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
  default     = "10.0.0.0/16"

  validation {
    condition     = can(regex("^([0-9]{1,3}\\.){3}[0-9]{1,3}/[0-9]{1,2}$", var.vpc_cidr_b>
    error_message = "vpc_cidr_block must be a valid CIDR block (e.g., 10.0.0.0/16)."
  }
}

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

  validation {
    condition     = can(regex("^([0-9]{1,3}\\.){3}[0-9]{1,3}/[0-9]{1,2}$", var.subnet_cid>
    error_message = "subnet_cidr_block must be a valid CIDR block (e.g., 10.0.10.0/24)."
  }
}

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

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

# Instance Type
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
```

Populate terraform.tfvars with your values:



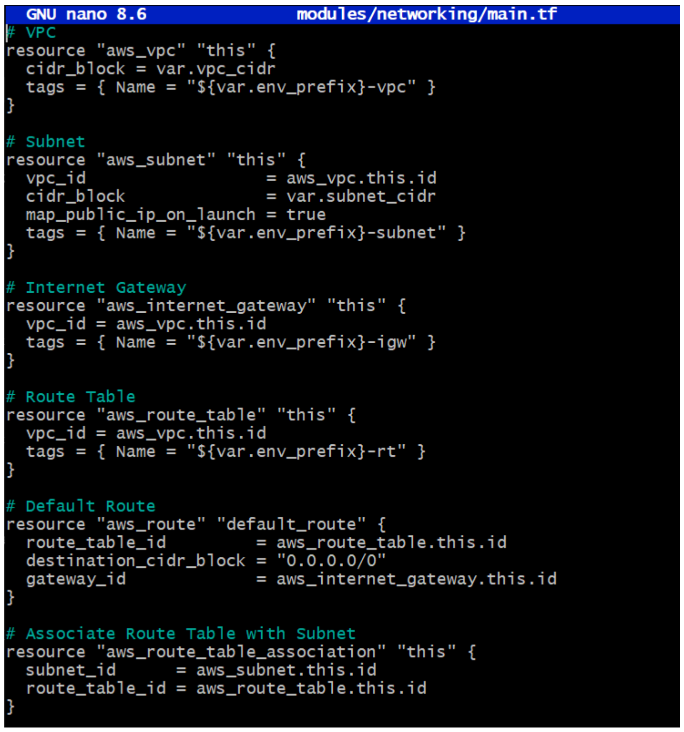
```
  GNU nano 8.6                     terraform.tfvars                     Modified
vpc_cidr_block    = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix        = "prod"
instance_type     = "t3.micro"
public_key        = "~/.ssh/id_ed25519.pub"
private_key       = "~/.ssh/id_ed25519"

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

- **1.3 Networking Module**

Create a networking module that provisions:

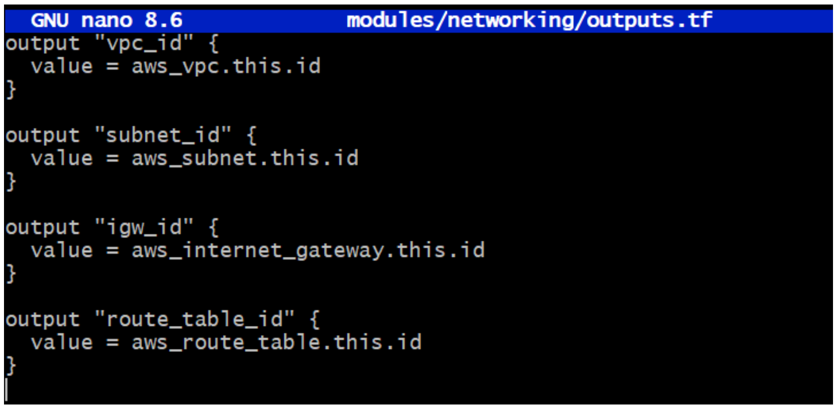
- VPC with specified CIDR block
- Subnet with public IP assignment enabled
- Internet Gateway
- Route table with default route to IGW
- Associate route table with subnet

In **modules/networking/main.tf**:



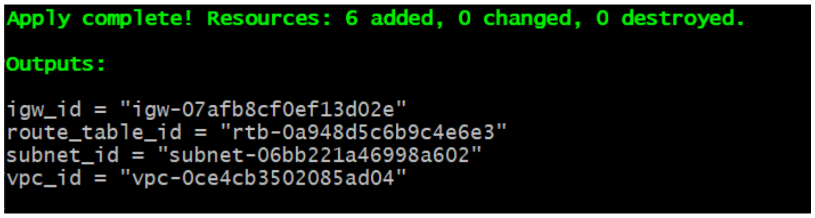
```
  GNU nano 8.6                modules/networking/main.tf
# VPC
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr
  tags = { Name = "${var.env_prefix}-vpc" }
}

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

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

# Route Table
resource "aws_route_table" "this" {
  vpc_id = aws_vpc.this.id
  tags = { Name = "${var.env_prefix}-rt" }
}

# Default Route
resource "aws_route" "default_route" {
  route_table_id         = aws_route_table.this.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.this.id
}

# 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
}
```

In **modules/networking/outputs.tf**:



```
  GNU nano 8.6              modules/networking/outputs.tf
output "vpc_id" {
  value = aws_vpc.this.id
}

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

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

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

After terraform is applied, following are the required outputs**:**

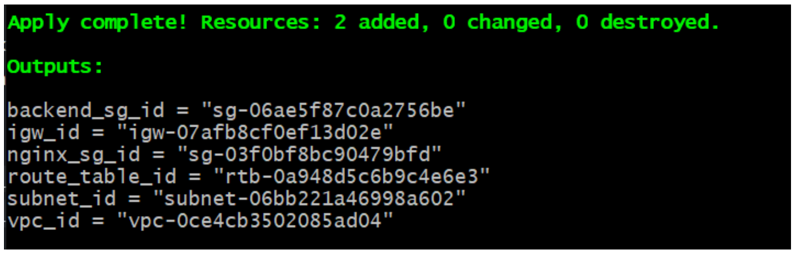


```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

igw_id = "igw-07afb8cf0ef13d02e"
route_table_id = "rtb-0a948d5c6b9c4e6e3"
subnet_id = "subnet-06bb221a46998a602"
vpc_id = "vpc-0ce4cb3502085ad04"
```

- ## 1.4 Security Module

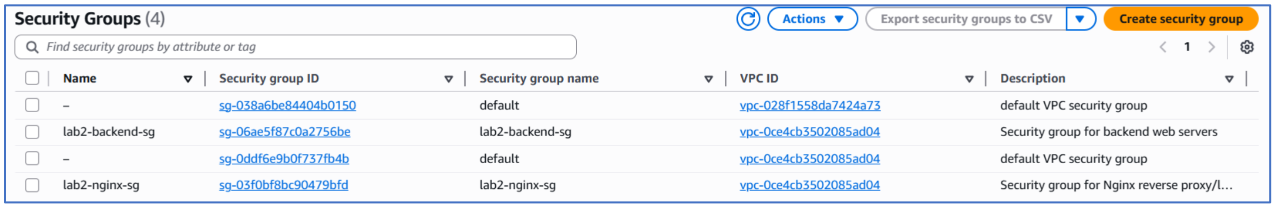
Create a security module:

Output from security module:



```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

backend_sg_id = "sg-06ae5f87c0a2756be"
igw_id = "igw-07afb8cf0ef13d02e"
nginx_sg_id = "sg-03f0bf8bc90479bfd"
route_table_id = "rtb-0a948d5c6b9c4e6e3"
subnet_id = "subnet-06bb221a46998a602"
vpc_id = "vpc-0ce4cb3502085ad04"
```

On AWS Console

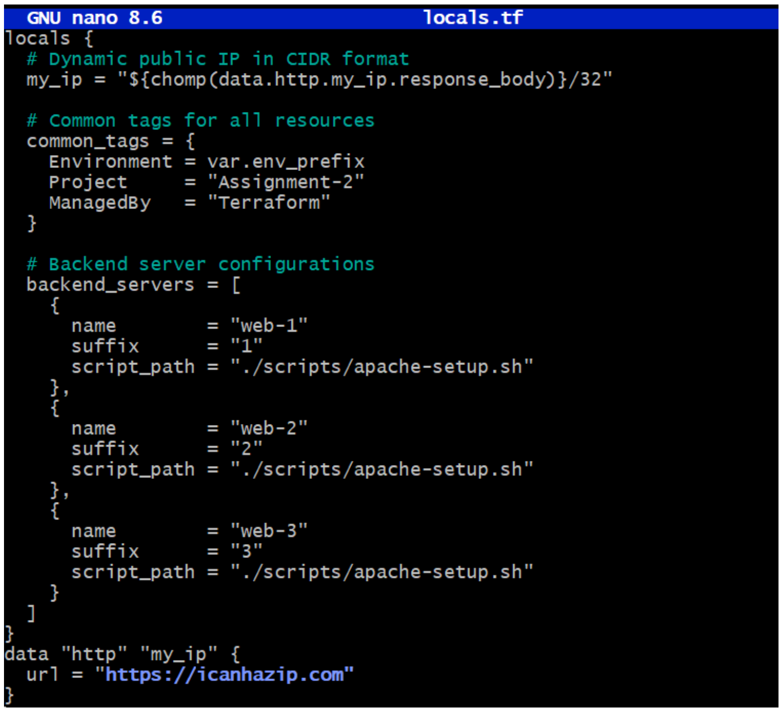

Security Groups (4)
Actions
Export security groups to CSV
Create security group
Find security groups by attribute or tag
1

Name	Security group ID	Security group name	VPC ID	Description
–	sg-038a6be84404b0150	default	vpc-028f1558da7424a73	default VPC security group
lab2-backend-sg	sg-06ae5f87c0a2756be	lab2-backend-sg	vpc-0ce4cb3502085ad04	Security group for backend web servers
–	sg-0ddf6e9b0f737fb4b	default	vpc-0ce4cb3502085ad04	default VPC security group
lab2-nginx-sg	sg-03f0bf8bc90479bfd	lab2-nginx-sg	vpc-0ce4cb3502085ad04	Security group for Nginx reverse proxy/l…

- **1.5 Locals Configuration**

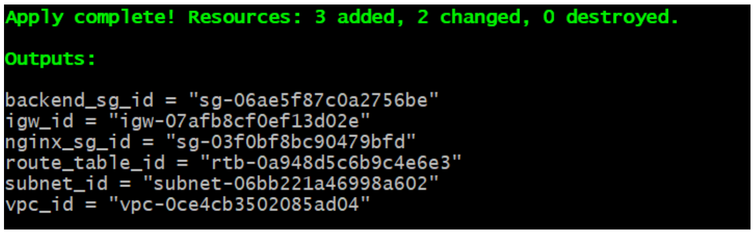
Create locals.tf with:

- Dynamic IP detection for my_ip
- Resource naming conventions
- Common tags
- Backend server configurations



```
  GNU nano 8.6                         locals.tf
locals {
  # Dynamic public IP in CIDR format
  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 server configurations
  backend_servers = [
    {
      name        = "web-1"
      suffix      = "1"
      script_path = "./scripts/apache-setup.sh"
    },
    {
      name        = "web-2"
      suffix      = "2"
      script_path = "./scripts/apache-setup.sh"
    },
    {
      name        = "web-3"
      suffix      = "3"
      script_path = "./scripts/apache-setup.sh"
    }
  ]
}
data "http" "my_ip" {
  url = "https://icanhazip.com"
}
```

After terraform is applied adding necessary modules to main.tf and adding required configurations to locals.tf:

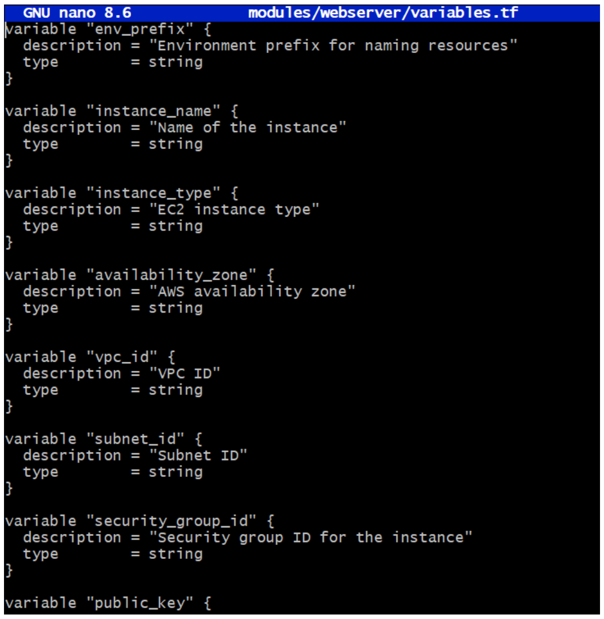


```
Apply complete! Resources: 3 added, 2 changed, 0 destroyed.

Outputs:

backend_sg_id = "sg-06ae5f87c0a2756be"
igw_id = "igw-07afb8cf0ef13d02e"
nginx_sg_id = "sg-03f0bf8bc90479bfd"
route_table_id = "rtb-0a948d5c6b9c4e6e3"
subnet_id = "subnet-06bb221a46998a602"
vpc_id = "vpc-0ce4cb3502085ad04"
```

- **Part 2: Webserver Module**
  - **2.1 Module Design**

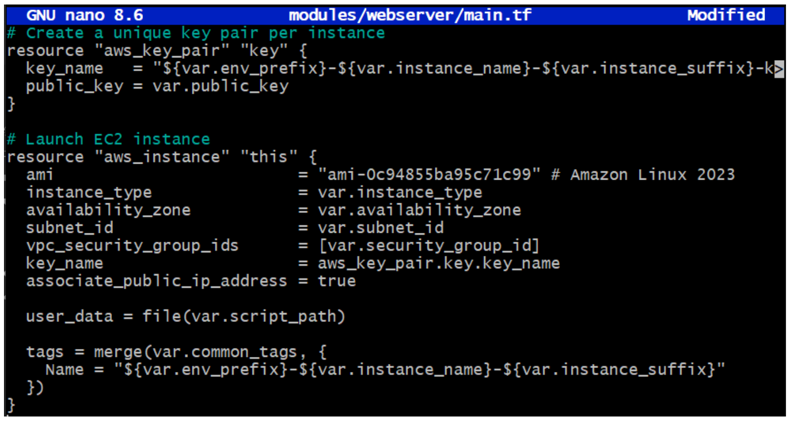
Create a reusable webserver module in modules/webserver

In modules/webserver/variables.tf:



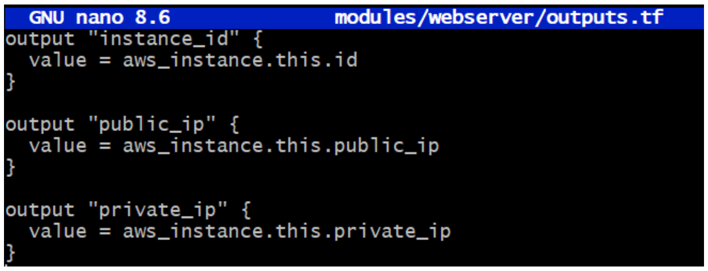
```
GNU nano 8.6                modules/webserver/variables.tf
variable "env_prefix" {
  description = "Environment prefix for naming resources"
  type        = string
}

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 for the instance"
  type        = string
}

variable "public_key" {
```

In modules/webserver/main.tf:



```
GNU nano 8.6                modules/webserver/main.tf                Modified
# Create a unique key pair per instance
resource "aws_key_pair" "key" {
  key_name   = "${var.env_prefix}-${var.instance_name}-${var.instance_suffix}-k>
  public_key = var.public_key
}

# Launch EC2 instance
resource "aws_instance" "this" {
  ami                         = "ami-0c94855ba95c71c99" # Amazon Linux 2023
  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.key.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}"
  })
}
```

In modules/webserver/outputs.tf:



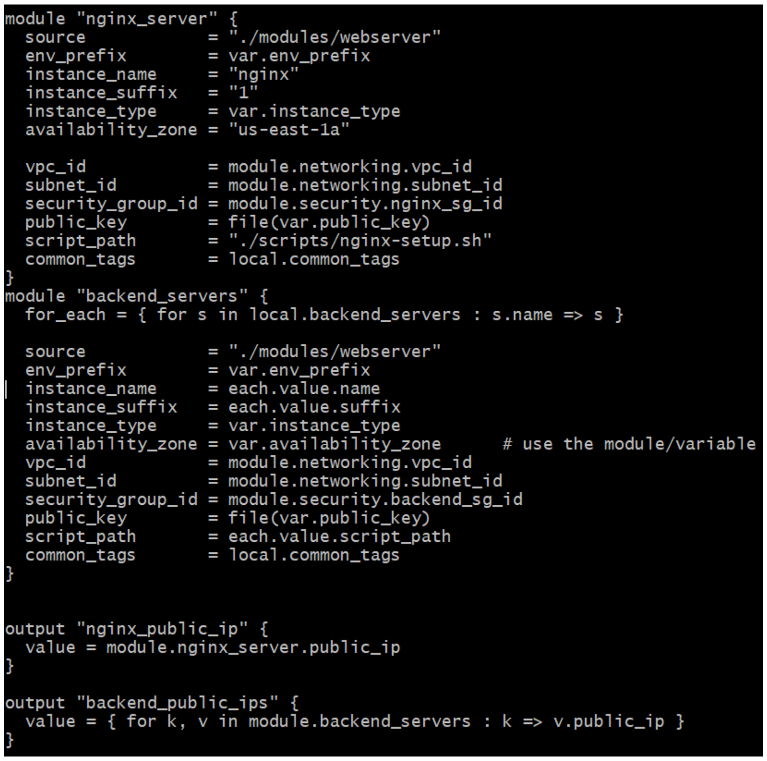
```
GNU nano 8.6            modules/webserver/outputs.tf
output "instance_id" {
  value = aws_instance.this.id
}

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

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

- **2.2 Module Usage**

In root main.tf, instantiate the webserver module for:

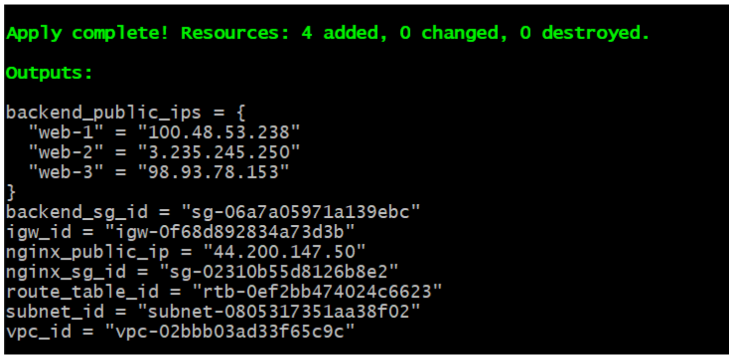
1. One Nginx server (using nginx-setup.sh)
2. Three backend servers (web-1, web-2, web-3 using apache-setup.sh)

Use dynamic blocks or for_each for backend servers



```
module "nginx_server" {
  source            = "./modules/webserver"
  env_prefix        = var.env_prefix
  instance_name     = "nginx"
  instance_suffix   = "1"
  instance_type     = var.instance_type
  availability_zone = "us-east-1a"

  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.nginx_sg_id
  public_key        = file(var.public_key)
  script_path       = "./scripts/nginx-setup.sh"
  common_tags       = local.common_tags
}
module "backend_servers" {
  for_each = { for s in local.backend_servers : s.name => s }

  source            = "./modules/webserver"
  env_prefix        = var.env_prefix
  instance_name     = each.value.name
  instance_suffix   = each.value.suffix
  instance_type     = var.instance_type
  availability_zone = var.availability_zone      # use the module/variable
  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.backend_sg_id
  public_key        = file(var.public_key)
  script_path       = each.value.script_path
  common_tags       = local.common_tags
}


output "nginx_public_ip" {
  value = module.nginx_server.public_ip
}

output "backend_public_ips" {
  value = { for k, v in module.backend_servers : k => v.public_ip }
}
```

After applying terraform:



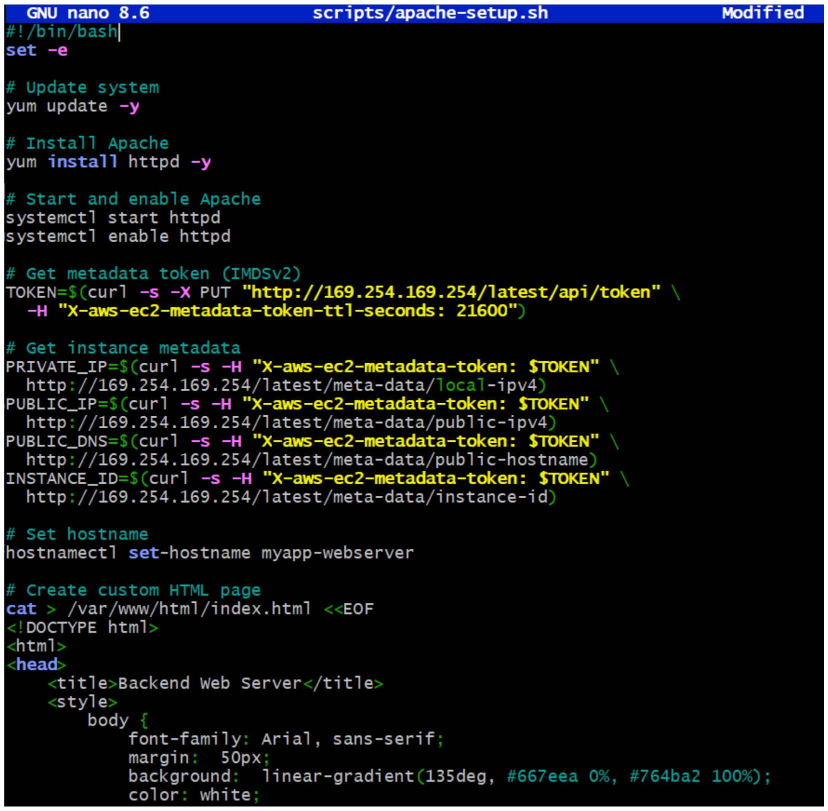
```
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:

backend_public_ips = {
  "web-1" = "100.48.53.238"
  "web-2" = "3.235.245.250"
  "web-3" = "98.93.78.153"
}
backend_sg_id = "sg-06a7a05971a139ebc"
igw_id = "igw-0f68d892834a73d3b"
nginx_public_ip = "44.200.147.50"
nginx_sg_id = "sg-02310b55d8126b8e2"
route_table_id = "rtb-0ef2bb474024c6623"
subnet_id = "subnet-0805317351aa38f02"
vpc_id = "vpc-02bbb03ad33f65c9c"
```

- **Part 3: Server Configuration Scripts**
    - **3.1 Apache Backend Server Script**

Create scripts/apache-setup.sh

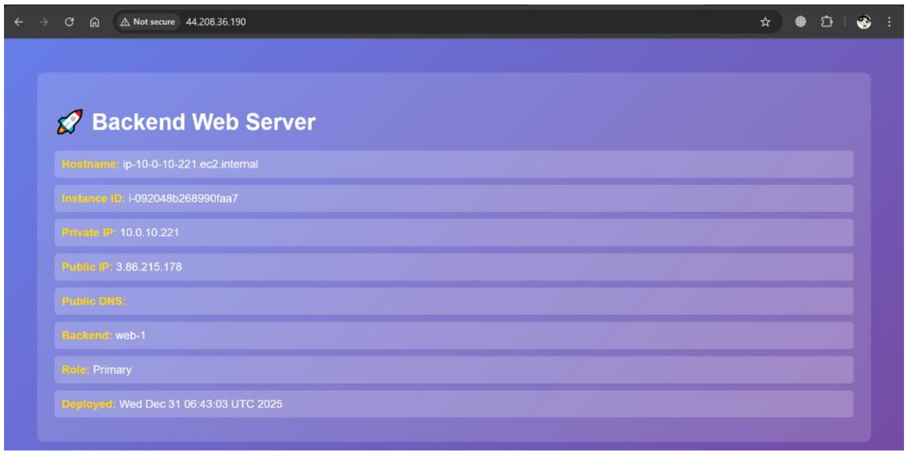
Apache Shell script code:

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

# Update system
yum update -y

# Install Apache
yum install httpd -y

# Start and enable Apache
systemctl start httpd
systemctl enable httpd

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

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

# Set hostname
hostnamectl set-hostname myapp-webserver

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

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

echo "Apache setup completed successfully!"
```



```
  GNU nano 8.6                 scripts/apache-setup.sh                 Modified
#!/bin/bash
set -e

# Update system
yum update -y

# Install Apache
yum install httpd -y

# Start and enable Apache
systemctl start httpd
systemctl enable httpd

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

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

# Set hostname
hostnamectl set-hostname myapp-webserver

# Create custom HTML page
cat > /var/www/html/index.html <<EOF
<!DOCTYPE html>
<html>
<head>
    <title>Backend Web Server</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 50px;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
```

Webpage:



🚀 Backend Web Server

Hostname: ip-10-0-10-221.ec2.internal

Instance ID: i-092048b268990faa7

Private IP: 10.0.10.221

Public IP: 3.86.215.178

Public DNS:

Backend: web-1

Role: Primary

Deployed: Wed Dec 31 06:43:03 UTC 2025

- ### 3.2 Nginx Server Setup Script

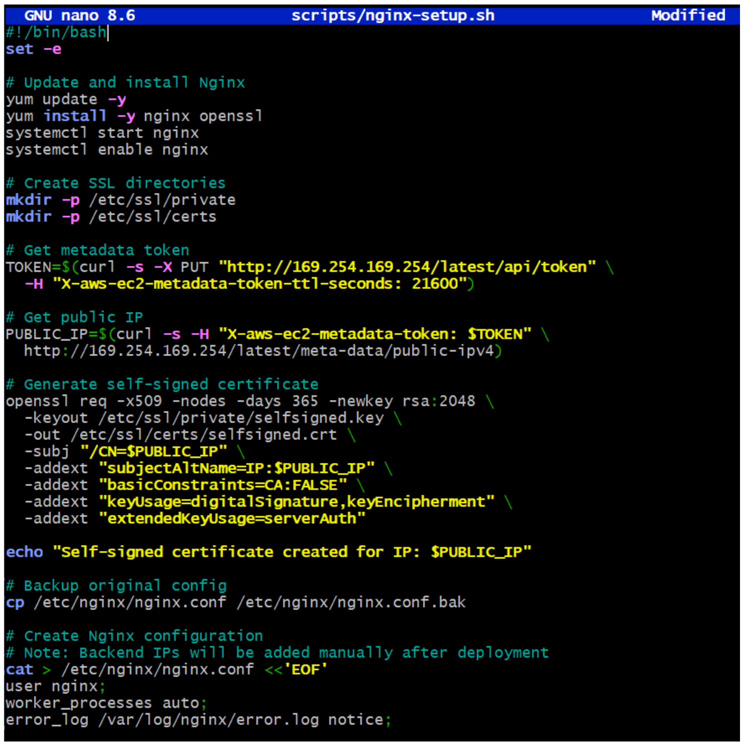
Create scripts/nginx-setup.sh

Enhanced Script Template:

```
#!/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;

    # Backup server (only used when primary servers are down)
    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_send_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"
```

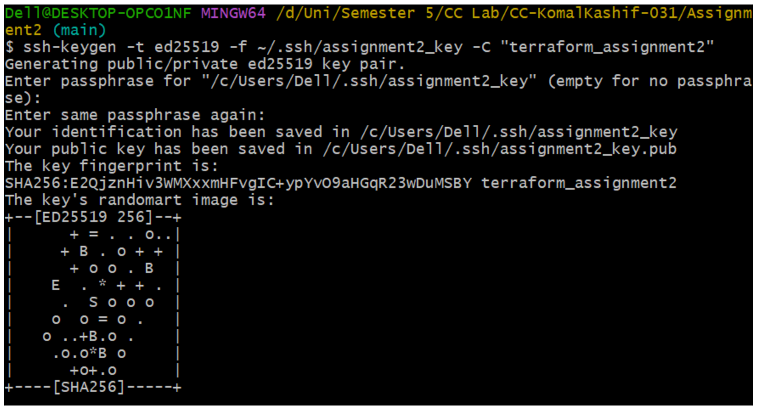


```
  GNU nano 8.6                  scripts/nginx-setup.sh                  Modified
#!/bin/bash
set -e

# 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;
```

- **Part 4: Infrastructure Deployment**
    - **4.1 Initial Deployment**

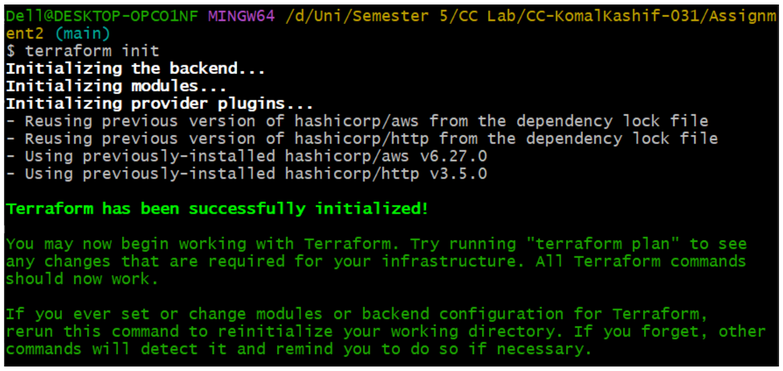
Deploy the infrastructure using Terraform.

- Generate SSH key pair if not exists



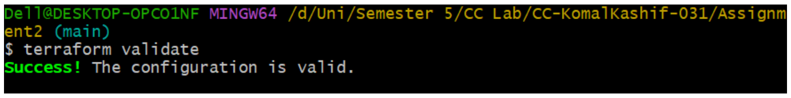
```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ ssh-keygen -t ed25519 -f ~/.ssh/assignment2_key -C "terraform_assignment2"
Generating public/private ed25519 key pair.
Enter passphrase for "/c/Users/Dell/.ssh/assignment2_key" (empty for no passphra
se):
Enter same passphrase again:
Your identification has been saved in /c/Users/Dell/.ssh/assignment2_key
Your public key has been saved in /c/Users/Dell/.ssh/assignment2_key.pub
The key fingerprint is:
SHA256:E2QjznHiv3WMXxxmHFvgIC+ypYvO9aHGqR23wDuMSBY terraform_assignment2
The key's randomart image is:
+--[ED25519 256]--+
|      + = . . o..|
|     + B . o + + |
|      + o o . B  |
|    E  . * + + . |
|     .  S o o o  |
|    o  o = o .   |
|   o ..+B.o .    |
|    .o.o*B o     |
|      +o+.o      |
+----[SHA256]-----+
```

- Initialize Terraform



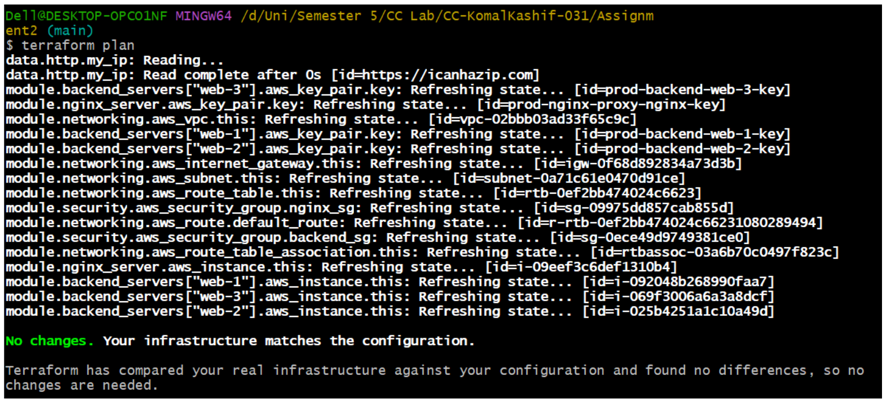
```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of 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.
```

- Validate configuration



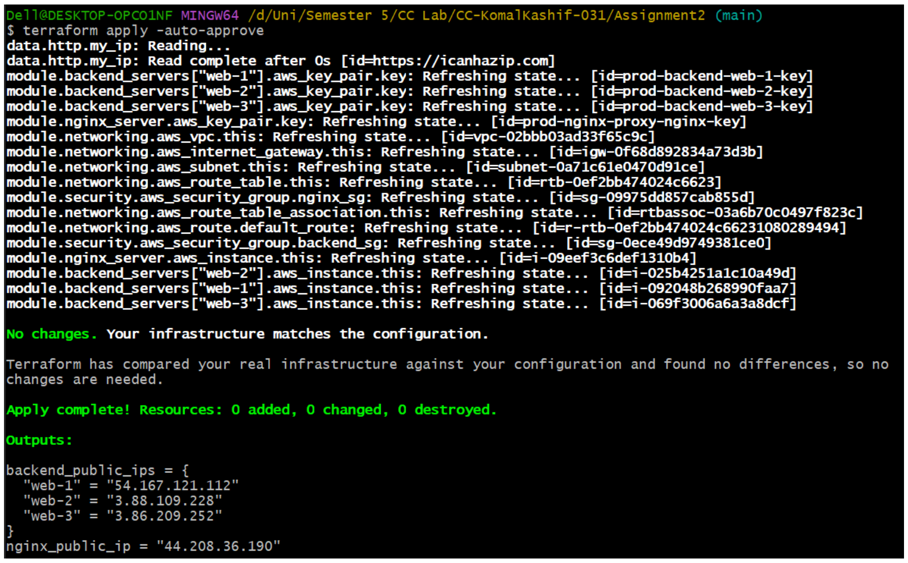
```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ terraform validate
Success! The configuration is valid.
```

- Plan deployment



```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ terraform plan
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
module.backend_servers["web-3"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-3-key]
module.nginx_server.aws_key_pair.key: Refreshing state... [id=prod-nginx-proxy-nginx-key]
module.networking.aws_vpc.this: Refreshing state... [id=vpc-02bbb03ad33f65c9c]
module.backend_servers["web-1"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-1-key]
module.backend_servers["web-2"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-2-key]
module.networking.aws_internet_gateway.this: Refreshing state... [id=igw-0f68d892834a73d3b]
module.networking.aws_subnet.this: Refreshing state... [id=subnet-0a71c61e0470d91ce]
module.networking.aws_route_table.this: Refreshing state... [id=rtb-0ef2bb474024c6623]
module.security.aws_security_group.nginx_sg: Refreshing state... [id=sg-09975dd857cab855d]
module.networking.aws_route.default_route: Refreshing state... [id=r-rtb-0ef2bb474024c66231080289494]
module.security.aws_security_group.backend_sg: Refreshing state... [id=sg-0ece49d9749381ce0]
module.networking.aws_route_table_association.this: Refreshing state... [id=rtbassoc-03a6b70c0497f823c]
module.nginx_server.aws_instance.this: Refreshing state... [id=i-09eef3c6def1310b4]
module.backend_servers["web-1"].aws_instance.this: Refreshing state... [id=i-092048b268990faa7]
module.backend_servers["web-3"].aws_instance.this: Refreshing state... [id=i-069f3006a6a3a8dcf]
module.backend_servers["web-2"].aws_instance.this: Refreshing state... [id=i-025b4251a1c10a49d]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no
changes are needed.
```

- Apply configuration

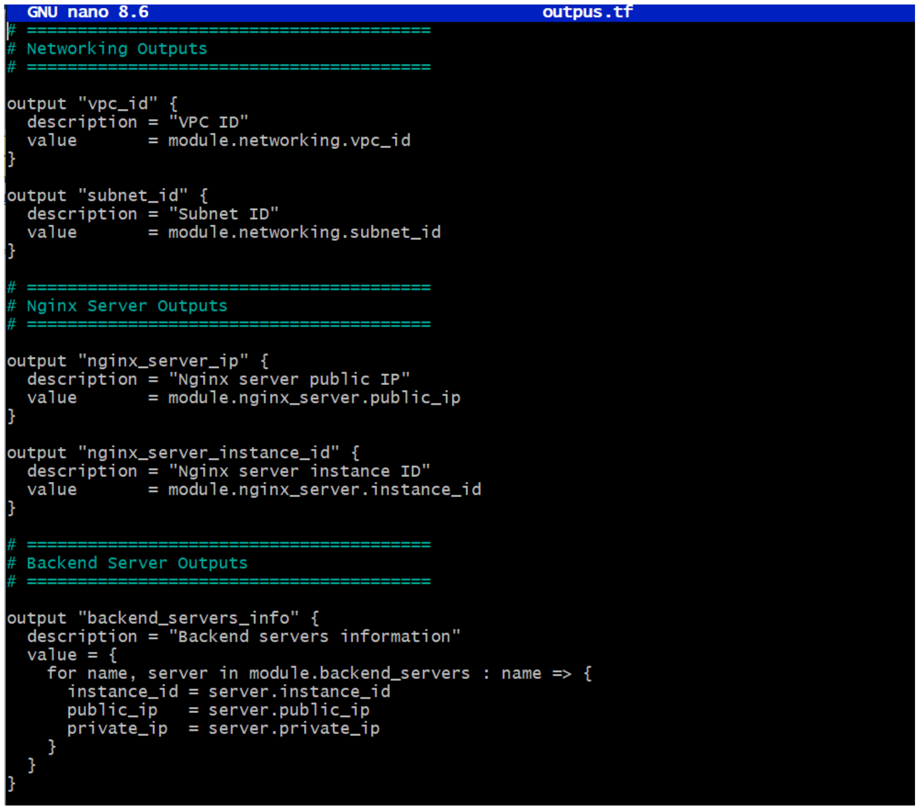


```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ terraform apply -auto-approve
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
module.backend_servers["web-1"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-1-key]
module.backend_servers["web-2"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-2-key]
module.backend_servers["web-3"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-3-key]
module.nginx_server.aws_key_pair.key: Refreshing state... [id=prod-nginx-proxy-nginx-key]
module.networking.aws_vpc.this: Refreshing state... [id=vpc-02bbb03ad33f65c9c]
module.networking.aws_internet_gateway.this: Refreshing state... [id=igw-0f68d892834a73d3b]
module.networking.aws_subnet.this: Refreshing state... [id=subnet-0a71c61e0470d91ce]
module.networking.aws_route_table.this: Refreshing state... [id=rtb-0ef2bb474024c6623]
module.security.aws_security_group.nginx_sg: Refreshing state... [id=sg-09975dd857cab855d]
module.networking.aws_route_table_association.this: Refreshing state... [id=rtbassoc-03a6b70c0497f823c]
module.networking.aws_route.default_route: Refreshing state... [id=r-rtb-0ef2bb474024c66231080289494]
module.security.aws_security_group.backend_sg: Refreshing state... [id=sg-0ece49d9749381ce0]
module.nginx_server.aws_instance.this: Refreshing state... [id=i-09eef3c6def1310b4]
module.backend_servers["web-2"].aws_instance.this: Refreshing state... [id=i-025b4251a1c10a49d]
module.backend_servers["web-1"].aws_instance.this: Refreshing state... [id=i-092048b268990faa7]
module.backend_servers["web-3"].aws_instance.this: Refreshing state... [id=i-069f3006a6a3a8dcf]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no
changes are needed.

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

Outputs:

backend_public_ips = {
  "web-1" = "54.167.121.112"
  "web-2" = "3.88.109.228"
  "web-3" = "3.86.209.252"
}
nginx_public_ip = "44.208.36.190"
```

- **4.2 Output Configuration**

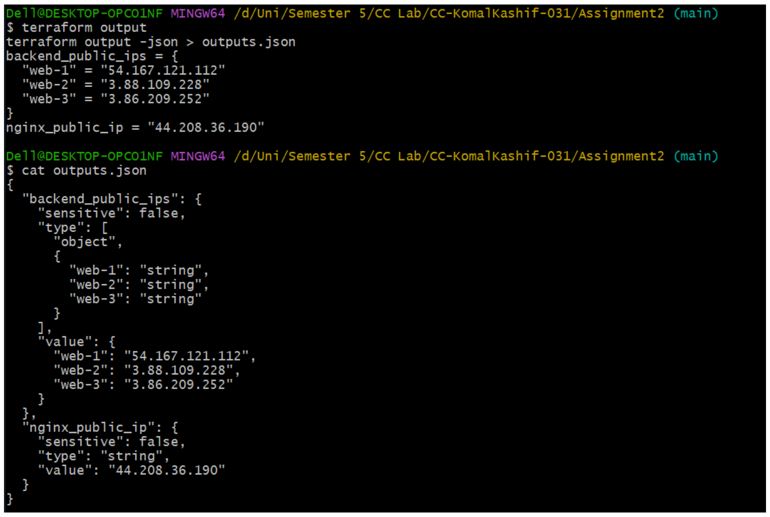
Create comprehensive outputs in outputs.tf:

```
# ========================================
# Networking Outputs
# ========================================

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

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

# ========================================
# Nginx Server Outputs
# ========================================

output "nginx_server_ip" {
  description = "Nginx server public IP"
  value       = module.nginx_server.public_ip
}

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

# ========================================
# Backend Server Outputs
# ========================================

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

output "configuration_guide" {
  value = <<-EOT

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

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

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

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



```
  GNU nano 8.6                                    outpus.tf
# ========================================
# Networking Outputs
# ========================================

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

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

# ========================================
# Nginx Server Outputs
# ========================================

output "nginx_server_ip" {
  description = "Nginx server public IP"
  value       = module.nginx_server.public_ip
}

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

# ========================================
# Backend Server Outputs
# ========================================

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

Display outputs after apply:

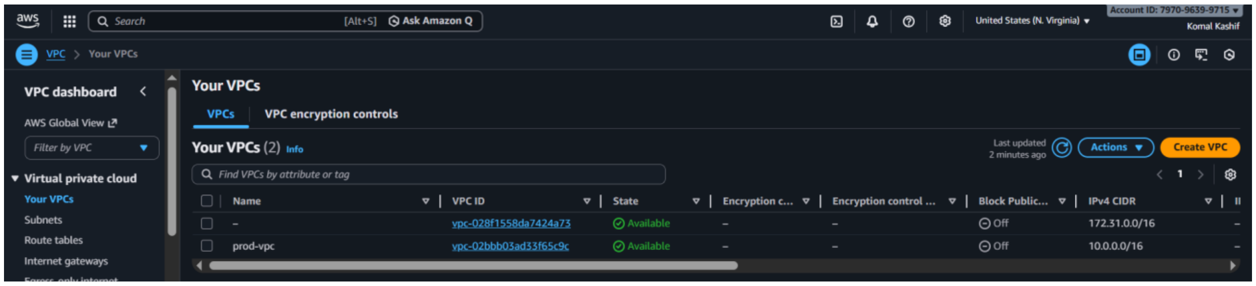


```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ terraform output
terraform output -json > outputs.json
backend_public_ips = {
  "web-1" = "54.167.121.112"
  "web-2" = "3.88.109.228"
  "web-3" = "3.86.209.252"
}
nginx_public_ip = "44.208.36.190"

Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ cat outputs.json
{
  "backend_public_ips": {
    "sensitive": false,
    "type": [
      "object",
      {
        "web-1": "string",
        "web-2": "string",
        "web-3": "string"
      }
    ],
    "value": {
      "web-1": "54.167.121.112",
      "web-2": "3.88.109.228",
      "web-3": "3.86.209.252"
    }
  },
  "nginx_public_ip": {
    "sensitive": false,
    "type": "string",
    "value": "44.208.36.190"
  }
}
```

- **4.3 AWS Console Verification**

  Verify all resources in AWS Console.

VPC Console:

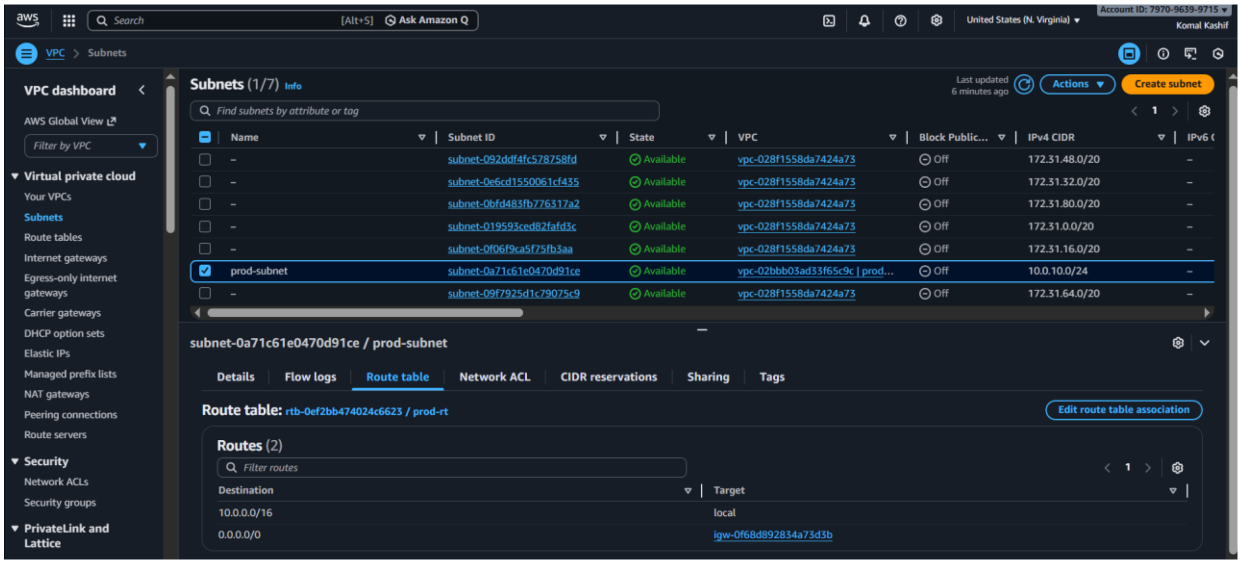


Your VPCs

Your VPCs (2) Info

Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv4 CIDR
–	vpc-028f1558da7424a73	Available	–	–	Off	172.31.0.0/16
prod-vpc	vpc-02bbb03ad33f65c9c	Available	–	–	Off	10.0.0.0/16

Subnets:



Subnets (1/7) Info

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 (
–	subnet-092ddf4fc578758fd	Available	vpc-028f1558da7424a73	Off	172.31.48.0/20	–
–	subnet-0e6cd1550061cf435	Available	vpc-028f1558da7424a73	Off	172.31.32.0/20	–
–	subnet-0bfd483fb776317a2	Available	vpc-028f1558da7424a73	Off	172.31.80.0/20	–
–	subnet-019593ced82fafd3c	Available	vpc-028f1558da7424a73	Off	172.31.0.0/20	–
–	subnet-0f06f9ca5f75fb3aa	Available	vpc-028f1558da7424a73	Off	172.31.16.0/20	–
prod-subnet	subnet-0a71c61e0470d91ce	Available	vpc-02bbb03ad33f65c9c | prod...	Off	10.0.10.0/24	–
–	subnet-09f7925d1c79075c9	Available	vpc-028f1558da7424a73	Off	172.31.64.0/20	–

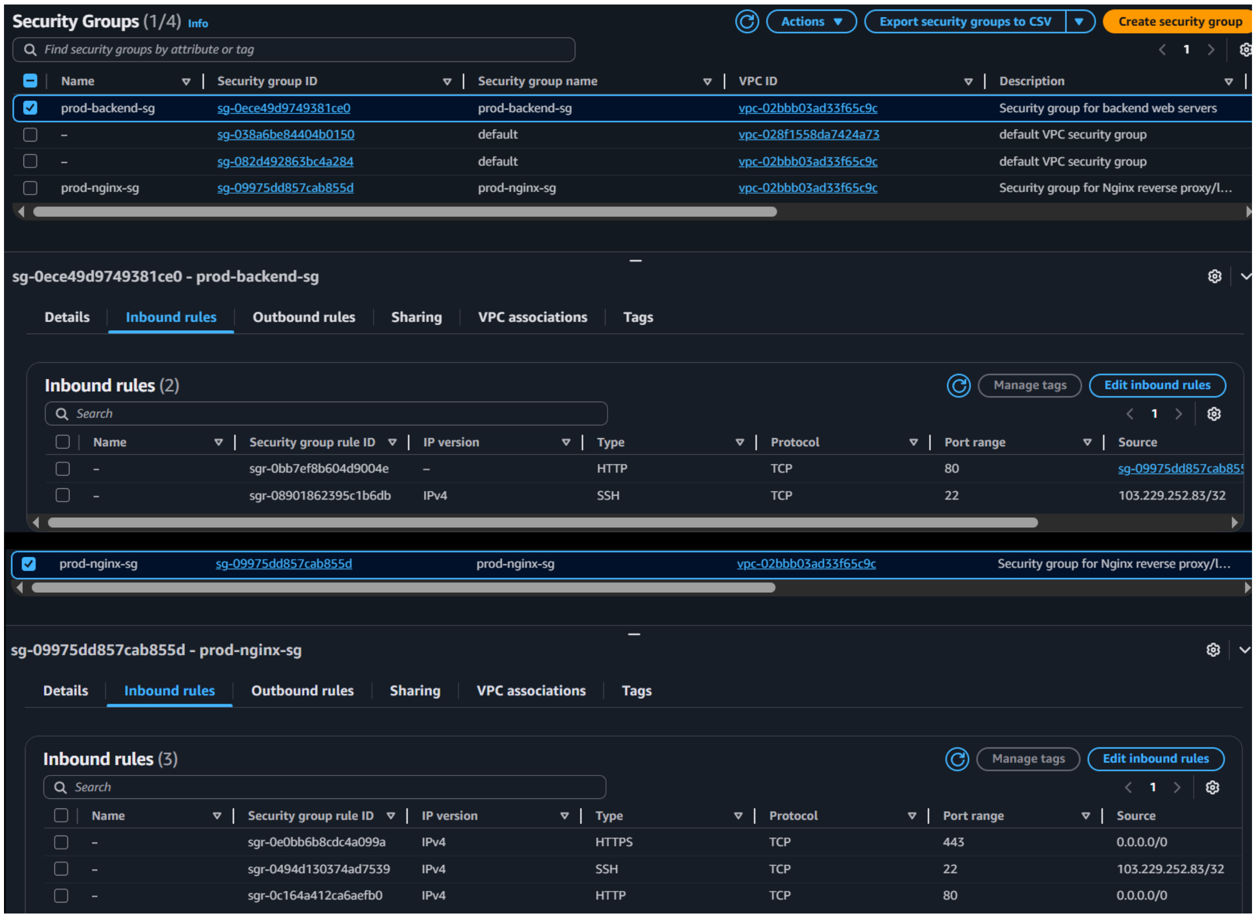
subnet-0a71c61e0470d91ce / prod-subnet

Route table: rtb-0ef2bb474024c6623 / prod-rt

Routes (2)

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-0f68d892834a73d3b

Security Groups:



Security Groups (1/4) Info

Name	Security group ID	Security group name	VPC ID	Description
prod-backend-sg	sg-0ece49d9749381ce0	prod-backend-sg	vpc-02bbb03ad33f65c9c	Security group for backend web servers
–	sg-038a6be84404b0150	default	vpc-028f1558da7424a73	default VPC security group
–	sg-082d492863bc4a284	default	vpc-02bbb03ad33f65c9c	default VPC security group
prod-nginx-sg	sg-09975dd857cab855d	prod-nginx-sg	vpc-02bbb03ad33f65c9c	Security group for Nginx reverse proxy/l…

sg-0ece49d9749381ce0 - prod-backend-sg

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
–	sgr-0bb7ef8b604d9004e	–	HTTP	TCP	80	sg-09975dd857cab85
–	sgr-08901862395c1b6db	IPv4	SSH	TCP	22	103.229.252.83/32

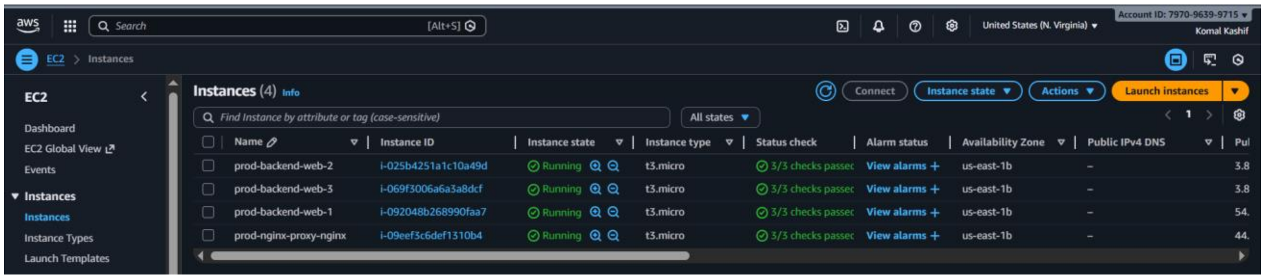
Name	Security group ID	Security group name	VPC ID	Description
prod-nginx-sg	sg-09975dd857cab855d	prod-nginx-sg	vpc-02bbb03ad33f65c9c	Security group for Nginx reverse proxy/l…

sg-09975dd857cab855d - prod-nginx-sg

Inbound rules (3)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
–	sgr-0e0bb6b8cdc4a099a	IPv4	HTTPS	TCP	443	0.0.0.0/0
–	sgr-0494d130374ad7539	IPv4	SSH	TCP	22	103.229.252.83/32
–	sgr-0c164a412ca6aefb0	IPv4	HTTP	TCP	80	0.0.0.0/0

Instances:



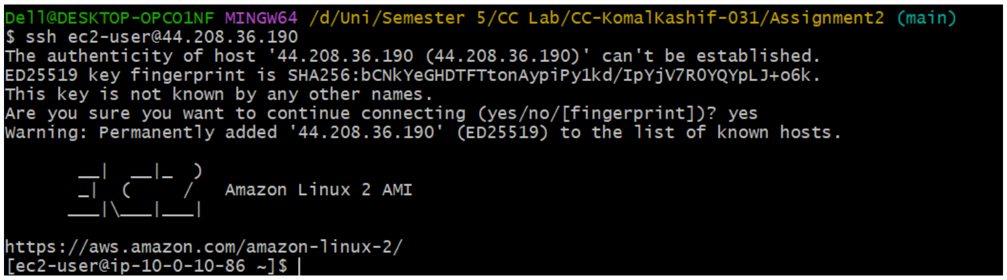
Instances (4) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Pul
prod-backend-web-2	i-025b4251a1c10a49d	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b	–	3.8
prod-backend-web-3	i-069f3006a6a3a8dcf	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b	–	3.8
prod-backend-web-1	i-092048b268990faa7	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b	–	54.
prod-nginx-proxy-nginx	i-09eef3c6def1310b4	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1b	–	44.

- **Part 5: Nginx Configuration & Testing**
  - **5.1 Update Nginx Backend Configuration**

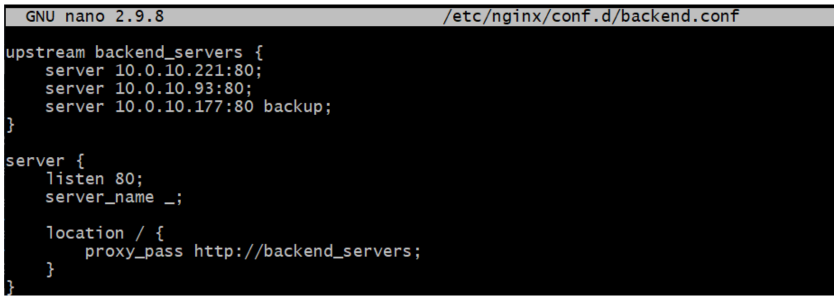
SSH into the Nginx server and update the configuration with actual backend IPs.

- SSH into Nginx server



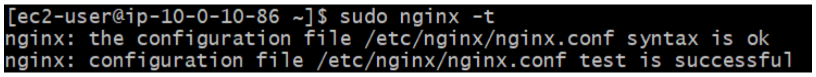
```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ ssh ec2-user@44.208.36.190
The authenticity of host '44.208.36.190 (44.208.36.190)' can't be established.
ED25519 key fingerprint is SHA256:bCNkYeGHDTFTtonAypiPy1kd/IpYjV7R0YQYpLJ+o6k.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.208.36.190' (ED25519) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-10-86 ~]$ |
```

- Edit /etc/nginx/nginx.conf
- Replace placeholder IPs with actual private IPs of backend servers



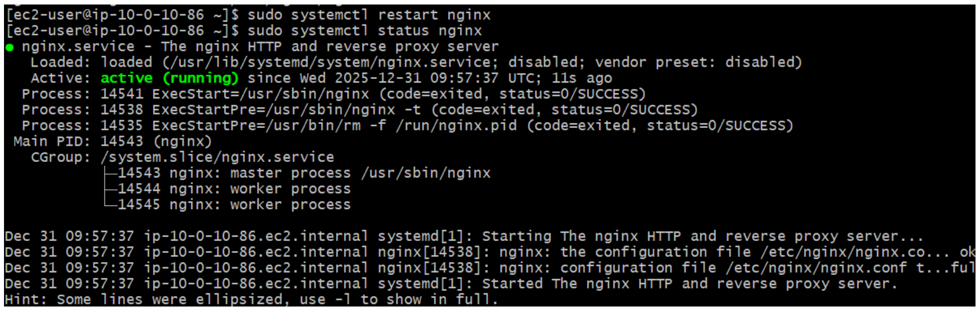
```
  GNU nano 2.9.8                          /etc/nginx/conf.d/backend.conf

upstream backend_servers {
    server 10.0.10.221:80;
    server 10.0.10.93:80;
    server 10.0.10.177:80 backup;
}

server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://backend_servers;
    }
}
```

- Test Nginx configuration



```
[ec2-user@ip-10-0-10-86 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

- Restart Nginx service



```
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-86 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2025-12-31 09:57:37 UTC; 11s ago
  Process: 14541 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 14538 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 14535 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
 Main PID: 14543 (nginx)
   CGroup: /system.slice/nginx.service
           ├─14543 nginx: master process /usr/sbin/nginx
           ├─14544 nginx: worker process
           └─14545 nginx: worker process

Dec 31 09:57:37 ip-10-0-10-86.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
Dec 31 09:57:37 ip-10-0-10-86.ec2.internal nginx[14538]: nginx: the configuration file /etc/nginx/nginx.co... ok
Dec 31 09:57:37 ip-10-0-10-86.ec2.internal nginx[14538]: nginx: configuration file /etc/nginx/nginx.conf t...ful
Dec 31 09:57:37 ip-10-0-10-86.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
Hint: Some lines were ellipsized, use -l to show in full.
```

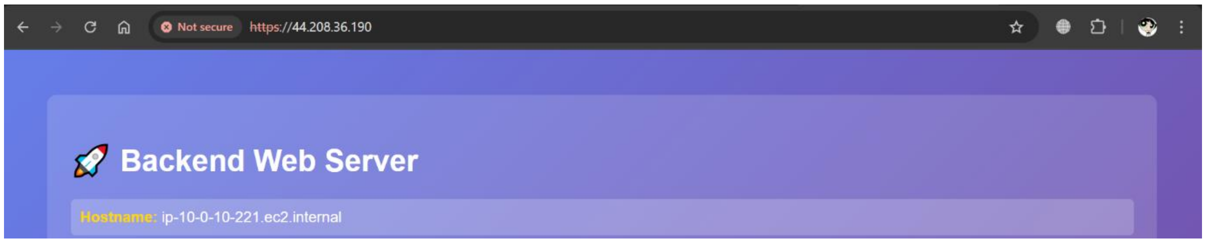
- ### 2.2 Test Load Balancing

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

- Open browser to https://<nginx-public-ip>

First replace the content in backend.conf with:

```
server {
   listen 443 ssl;
   server_name _;

   ssl_certificate /etc/nginx/ssl/nginx.crt;
   ssl_certificate_key /etc/nginx/ssl/nginx.key;

   location / {
      proxy_pass http://backend_servers;
   }
}
```

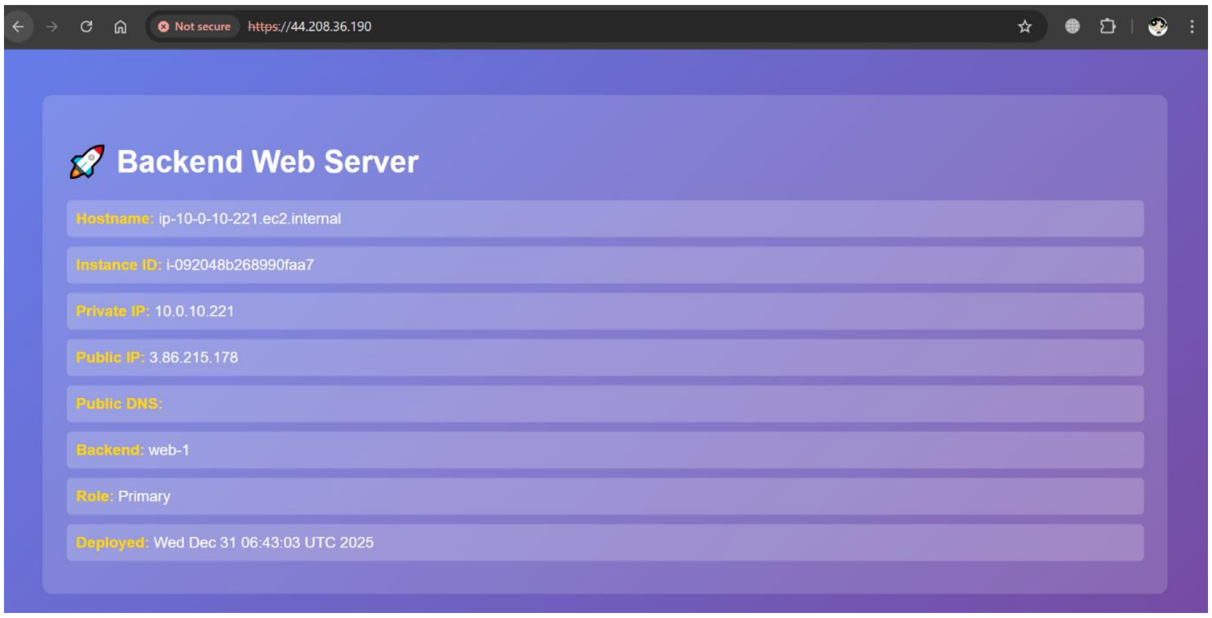


Not secure https://44.208.36.190

🚀 Backend Web Server

Hostname: ip-10-0-10-221.ec2.internal

- Reload page multiple times (at least 10 times). Verify traffic alternates between web-1 and web-2



Not secure https://44.208.36.190

🚀 Backend Web Server

Hostname: ip-10-0-10-221.ec2.internal

Instance ID: i-092048b268990faa7

Private IP: 10.0.10.221

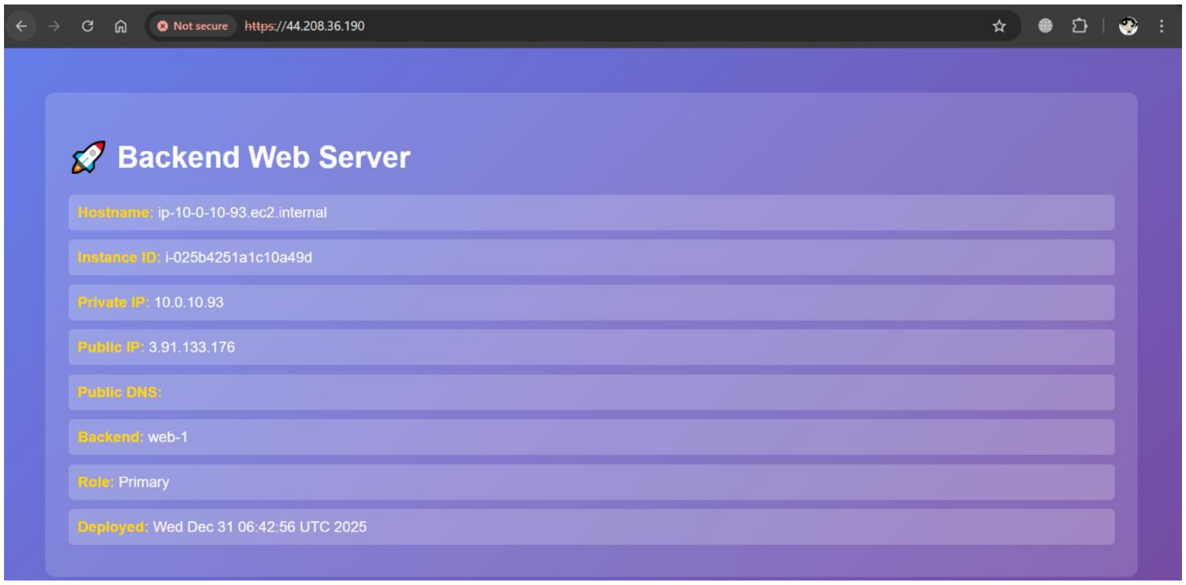
Public IP: 3.86.215.178

Public DNS:

Backend: web-1

Role: Primary

Deployed: Wed Dec 31 06:43:03 UTC 2025


Not secure https://44.208.36.190

🚀 Backend Web Server

Hostname: ip-10-0-10-93.ec2.internal

Instance ID: i-025b4251a1c10a49d

Private IP: 10.0.10.93

Public IP: 3.91.133.176

Public DNS:

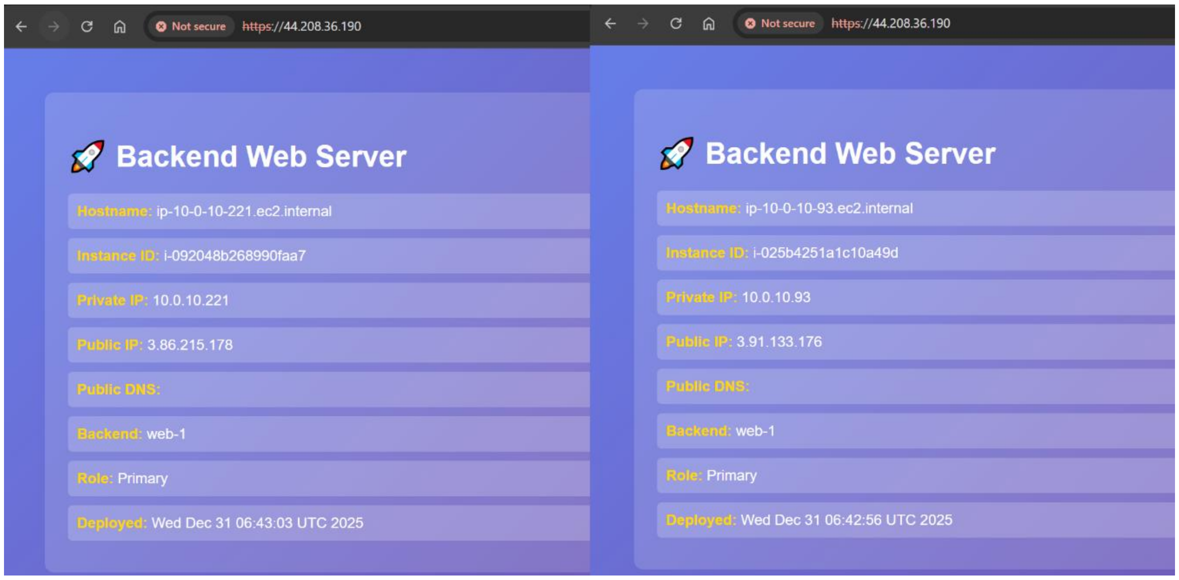
Backend: web-1

Role: Primary

Deployed: Wed Dec 31 06:42:56 UTC 2025

- Verify web-3 is NOT serving traffic (it's backup only)

As there are only two alternations in the webpage, hence web-3 is only backup.


Not secure https://44.208.36.190

🚀 Backend Web Server

Hostname: ip-10-0-10-221.ec2.internal

Instance ID: i-092048b268990faa7

Private IP: 10.0.10.221

Public IP: 3.86.215.178

Public DNS:

Backend: web-1

Role: Primary

Deployed: Wed Dec 31 06:43:03 UTC 2025

Not secure https://44.208.36.190

🚀 Backend Web Server

Hostname: ip-10-0-10-93.ec2.internal

Instance ID: i-025b4251a1c10a49d

Private IP: 10.0.10.93

Public IP: 3.91.133.176

Public DNS:

Backend: web-1

Role: Primary

Deployed: Wed Dec 31 06:42:56 UTC 2025

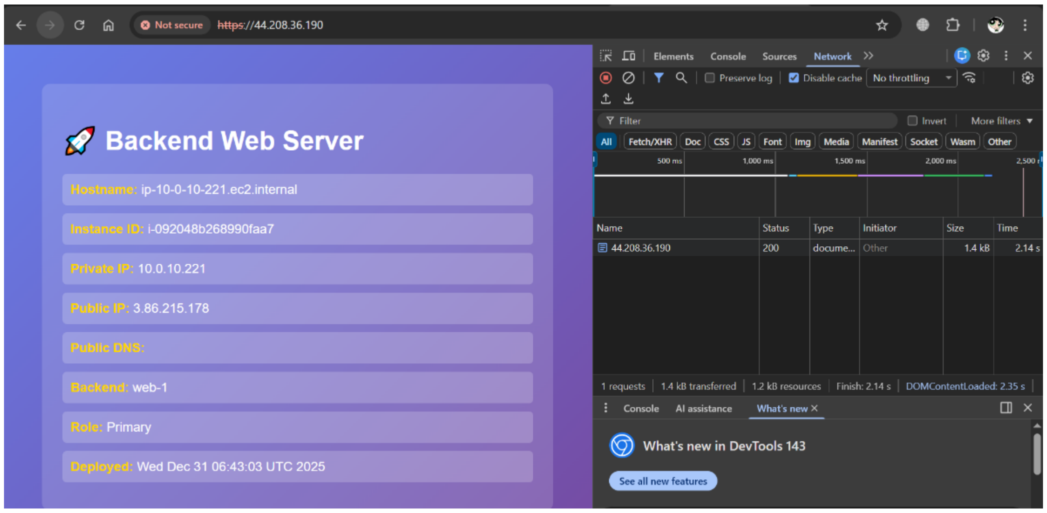
- ### 5.3 Test Cache Functionality

Verify that Nginx caching is working correctly.

First update the nano /etc/nginx/conf.d/backend.conf with:

```
 location / {
        proxy_pass http://backend_servers;

         proxy_cache my_cache;
        proxy_cache_valid 200 10m;
        proxy_cache_use_stale error timeout updating;

        add_header X-Cache-Status $upstream_cache_status;

    }
}
```

- Open browser developer tools (F12). Navigate to Network tab. Clear browser cache. Load https://<nginx-public-ip>
- Check response headers for X-Cache-Status: MISS (first request)


Not secure ~~https~~://44.208.36.190

🚀 Backend Web Server

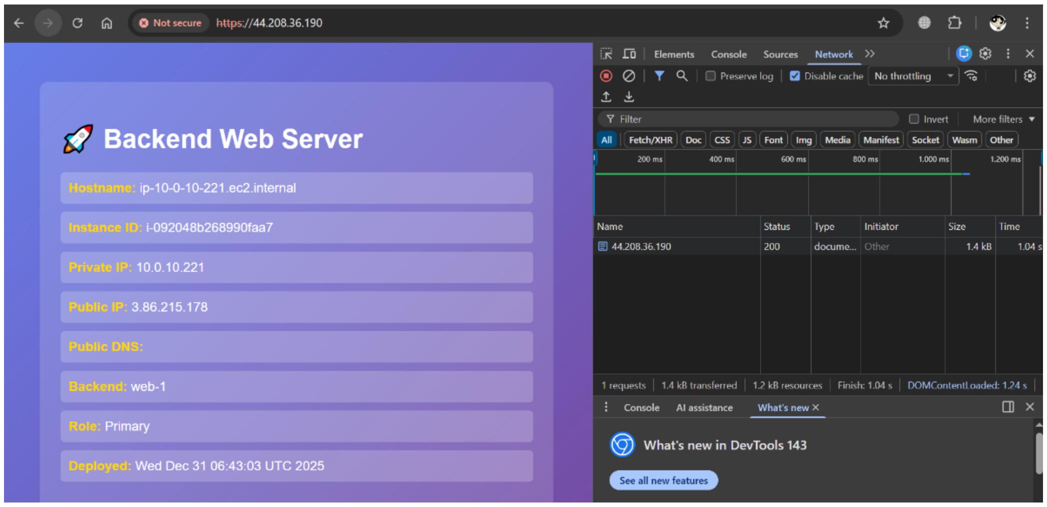
Hostname: ip-10-0-10-221.ec2.internal
Instance ID: i-092048b268990faa7
Private IP: 10.0.10.221
Public IP: 3.86.215.178
Public DNS:
Backend: web-1
Role: Primary
Deployed: Wed Dec 31 06:43:03 UTC 2025

Name	Status	Type	Initiator	Size	Time
44.208.36.190	200	docume...	Other	1.4 kB	2.14 s

1 requests | 1.4 kB transferred | 1.2 kB resources | Finish: 2.14 s | DOMContentLoaded: 2.35 s

What's new in DevTools 143

- Check response headers for X-Cache-Status: HIT (cached request)


Not secure https://44.208.36.190

🚀 Backend Web Server

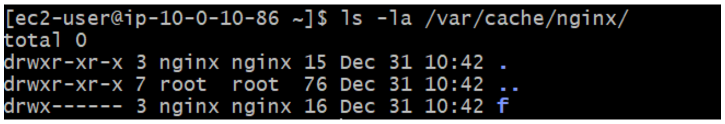
Hostname: ip-10-0-10-221.ec2.internal
Instance ID: i-092048b268990faa7
Private IP: 10.0.10.221
Public IP: 3.86.215.178
Public DNS:
Backend: web-1
Role: Primary
Deployed: Wed Dec 31 06:43:03 UTC 2025

Name	Status	Type	Initiator	Size	Time
44.208.36.190	200	docume...	Other	1.4 kB	1.04 s

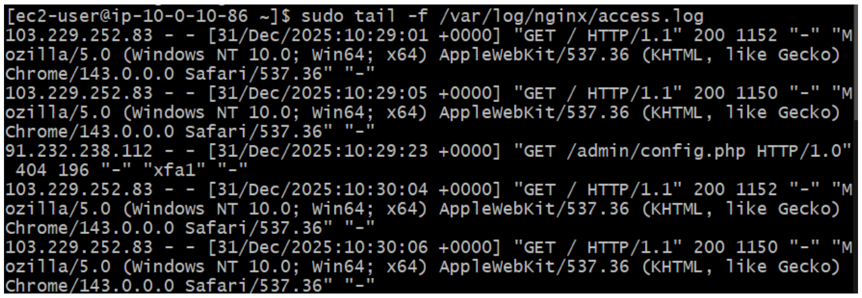
1 requests | 1.4 kB transferred | 1.2 kB resources | Finish: 1.04 s | DOMContentLoaded: 1.24 s

What's new in DevTools 143

- Verify cache directory on Nginx server



```
[ec2-user@ip-10-0-10-86 ~]$ ls -la /var/cache/nginx/
total 0
drwxr-xr-x 3 nginx nginx 15 Dec 31 10:42 .
drwxr-xr-x 7 root  root  76 Dec 31 10:42 ..
drwx------ 3 nginx nginx 16 Dec 31 10:42 f
```

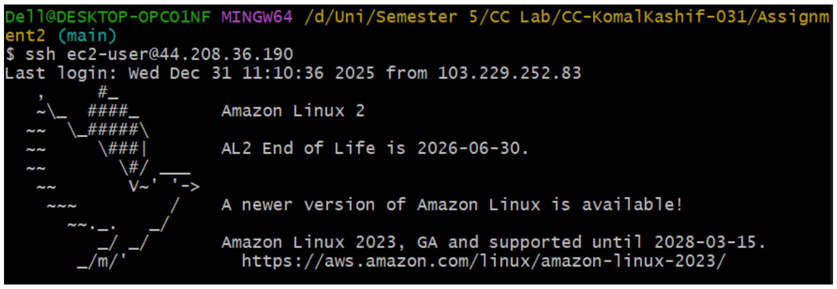


```
[ec2-user@ip-10-0-10-86 ~]$ sudo tail -f /var/log/nginx/access.log
103.229.252.83 - - [31/Dec/2025:10:29:01 +0000] "GET / HTTP/1.1" 200 1152 "-" "M
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:29:05 +0000] "GET / HTTP/1.1" 200 1150 "-" "M
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36" "-"
91.232.238.112 - - [31/Dec/2025:10:29:23 +0000] "GET /admin/config.php HTTP/1.0"
 404 196 "-" "xfa1" "-"
103.229.252.83 - - [31/Dec/2025:10:30:04 +0000] "GET / HTTP/1.1" 200 1152 "-" "M
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:30:06 +0000] "GET / HTTP/1.1" 200 1150 "-" "M
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/143.0.0.0 Safari/537.36" "-"
```

  - **5.4 Test High Availability (Backup Server)**

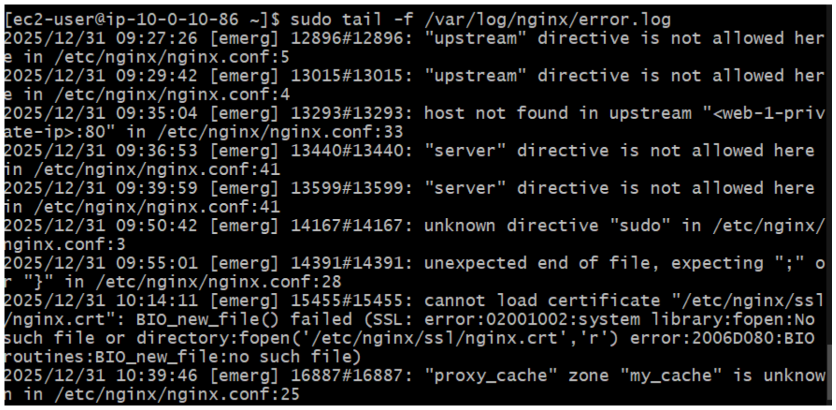
Test the backup server functionality by simulating primary server failure.

After services are stopped from web-1 and web-2, ssh nginx:



```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ ssh ec2-user@44.208.36.190
Last login: Wed Dec 31 11:10:36 2025 from 103.229.252.83
   ,     #_
   ~\_  ####_        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/
```

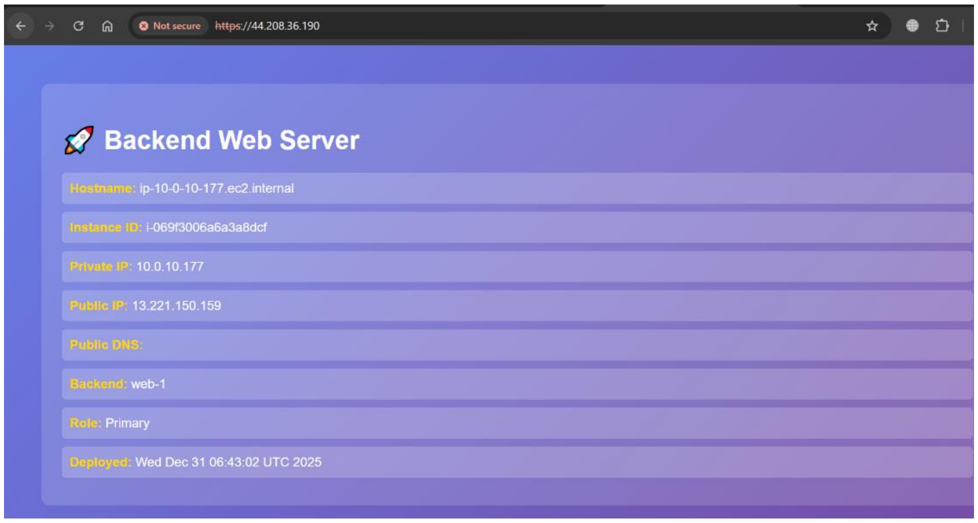
sudo tail -f /var/log/nginx/error.log



```
[ec2-user@ip-10-0-10-86 ~]$ sudo tail -f /var/log/nginx/error.log
2025/12/31 09:27:26 [emerg] 12896#12896: "upstream" directive is not allowed her
e in /etc/nginx/nginx.conf:5
2025/12/31 09:29:42 [emerg] 13015#13015: "upstream" directive is not allowed her
e in /etc/nginx/nginx.conf:4
2025/12/31 09:35:04 [emerg] 13293#13293: host not found in upstream "<web-1-priv
ate-ip>:80" in /etc/nginx/nginx.conf:33
2025/12/31 09:36:53 [emerg] 13440#13440: "server" directive is not allowed here
in /etc/nginx/nginx.conf:41
2025/12/31 09:39:59 [emerg] 13599#13599: "server" directive is not allowed here
in /etc/nginx/nginx.conf:41
2025/12/31 09:50:42 [emerg] 14167#14167: unknown directive "sudo" in /etc/nginx/
nginx.conf:3
2025/12/31 09:55:01 [emerg] 14391#14391: unexpected end of file, expecting ";" o
r "}" in /etc/nginx/nginx.conf:28
2025/12/31 10:14:11 [emerg] 15455#15455: cannot load certificate "/etc/nginx/ssl
/nginx.crt": BIO_new_file() failed (SSL: error:02001002:system library:fopen:No
such file or directory:fopen('/etc/nginx/ssl/nginx.crt','r') error:2006D080:BIO
routines:BIO_new_file:no such file)
2025/12/31 10:39:46 [emerg] 16887#16887: "proxy_cache" zone "my_cache" is unknow
n in /etc/nginx/nginx.conf:25
```

Now activation web-3 which is backup server:

sudo systemctl start httpd



Backend Web Server

Hostname: ip-10-0-10-177.ec2.internal
Instance ID: i-069f3006a6a3a8dcf
Private IP: 10.0.10.177
Public IP: 13.221.150.159
Public DNS:
Backend: web-1
Role: Primary
Deployed: Wed Dec 31 06:43:02 UTC 2025

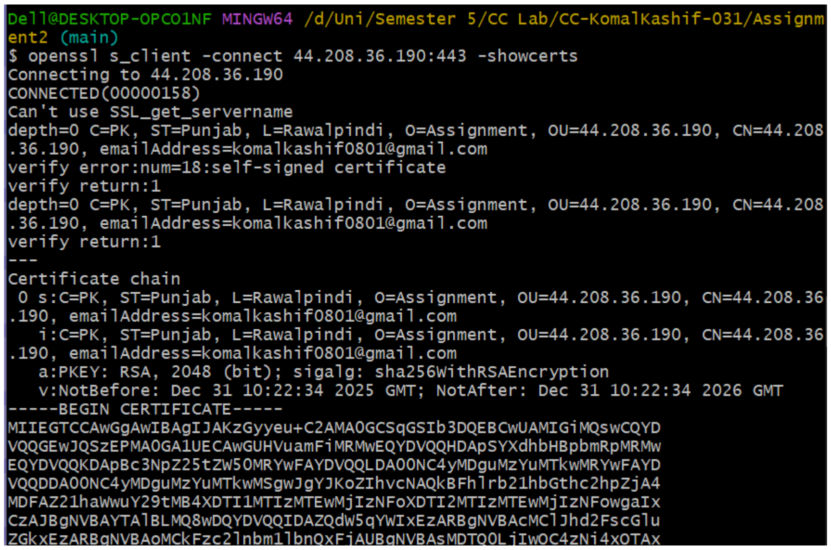
The services are now working on 10.0.10.177 which is backup server.

- ### 5.5 Security & Performance Analysis

Analyze the security headers and performance of your Nginx setup.

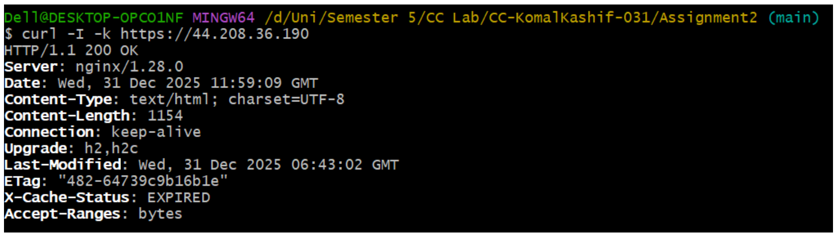
Tasks:

- Check SSL/TLS certificate details



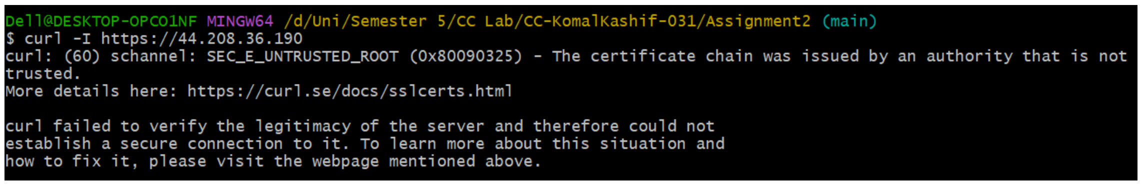
```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignm
ent2 (main)
$ openssl s_client -connect 44.208.36.190:443 -showcerts
Connecting to 44.208.36.190
CONNECTED(00000158)
Can't use SSL_get_servername
depth=0 C=PK, ST=Punjab, L=Rawalpindi, O=Assignment, OU=44.208.36.190, CN=44.208
.36.190, emailAddress=komalkashif0801@gmail.com
verify error:num=18:self-signed certificate
verify return:1
depth=0 C=PK, ST=Punjab, L=Rawalpindi, O=Assignment, OU=44.208.36.190, CN=44.208
.36.190, emailAddress=komalkashif0801@gmail.com
verify return:1
---
Certificate chain
 0 s:C=PK, ST=Punjab, L=Rawalpindi, O=Assignment, OU=44.208.36.190, CN=44.208.36
.190, emailAddress=komalkashif0801@gmail.com
   i:C=PK, ST=Punjab, L=Rawalpindi, O=Assignment, OU=44.208.36.190, CN=44.208.36
.190, emailAddress=komalkashif0801@gmail.com
   a:PKEY: RSA, 2048 (bit); sigalg: sha256WithRSAEncryption
   v:NotBefore: Dec 31 10:22:34 2025 GMT; NotAfter: Dec 31 10:22:34 2026 GMT
-----BEGIN CERTIFICATE-----
MIIEGTCCAwGgAwIBAgIJAKzGyyeu+C2AMA0GCSqGSIb3DQEBCwUAMIGiMQswCQYD
VQQGEwJQSzEPMA0GA1UECAwGUHVuamFiMRMwEQYDVQQHDApSYXdhbHBpbmRpMRMw
EQYDVQQKDApBc3NpZ25tZW50MRYwFAYDVQQLDA00NC4yMDguMzYuMTkwMRYwFAYD
VQQDDA00NC4yMDguMzYuMTkwMSgwJgYJKoZIhvcNAQkBFhlrb21hbGthc2hpZjA4
MDFAZ21haWwuY29tMB4XDTI1MTIzMTEwMjIzNFoXDTI2MTIzMTEwMjIzNFowgaIx
CzAJBgNVBAYTAlBLMQ8wDQYDVQQIDAZQdW5qYWIxEzARBgNVBAcMClJhd2FscGlu
ZGkxEzARBgNVBAoMCkFzc2lnbm1lbnQxFjAUBgNVBAsMDTQ0LjIwOC4zNi4xOTAx
```

- Verify security headers in response



```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ curl -I -k https://44.208.36.190
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Wed, 31 Dec 2025 11:59:09 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 1154
Connection: keep-alive
Upgrade: h2,h2c
Last-Modified: Wed, 31 Dec 2025 06:43:02 GMT
ETag: "482-64739c9b16b1e"
X-Cache-Status: EXPIRED
Accept-Ranges: bytes
```

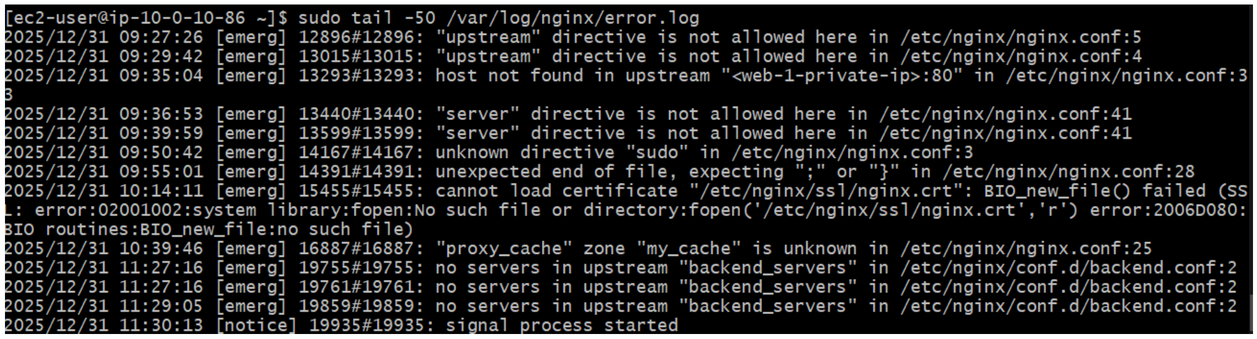
- Test HTTP to HTTPS redirect



```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ curl -I https://44.208.36.190
curl: (60) schannel: SEC_E_UNTRUSTED_ROOT (0x80090325) - The certificate chain was issued by an authority that is not
trusted.
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the webpage mentioned above.
```

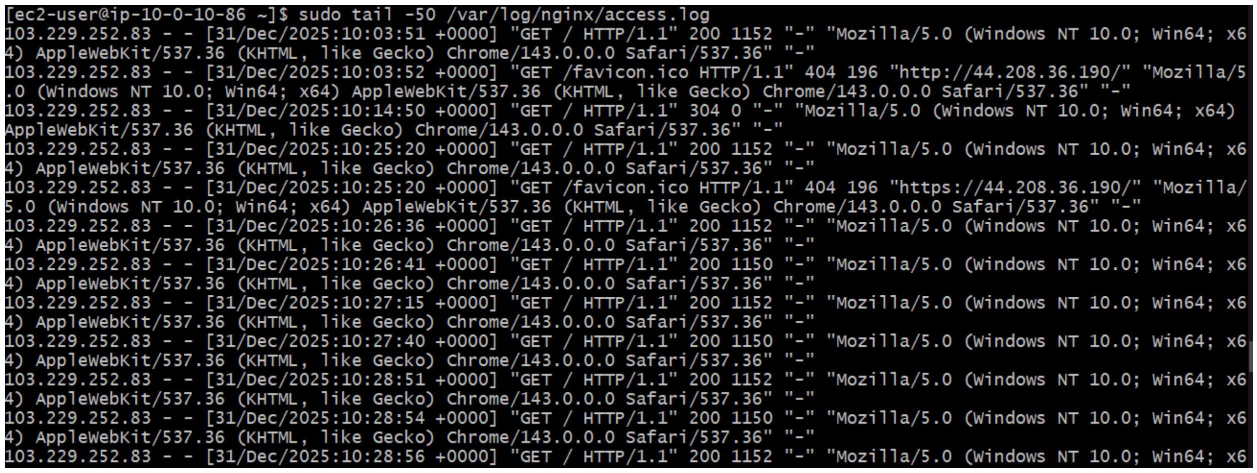
- Analyze Nginx logs

Error Log:



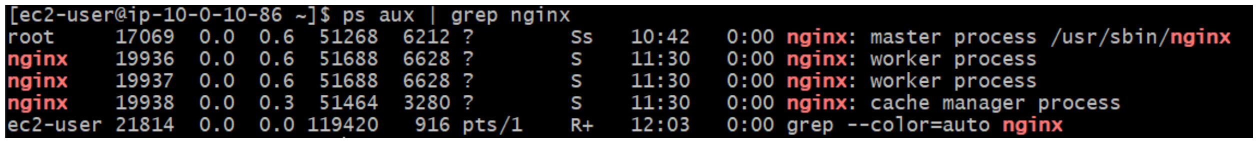
```
[ec2-user@ip-10-0-10-86 ~]$ sudo tail -50 /var/log/nginx/error.log
2025/12/31 09:27:26 [emerg] 12896#12896: "upstream" directive is not allowed here in /etc/nginx/nginx.conf:5
2025/12/31 09:29:42 [emerg] 13015#13015: "upstream" directive is not allowed here in /etc/nginx/nginx.conf:4
2025/12/31 09:35:04 [emerg] 13293#13293: host not found in upstream "<web-1-private-ip>:80" in /etc/nginx/nginx.conf:3
3
2025/12/31 09:36:53 [emerg] 13440#13440: "server" directive is not allowed here in /etc/nginx/nginx.conf:41
2025/12/31 09:39:59 [emerg] 13599#13599: "server" directive is not allowed here in /etc/nginx/nginx.conf:41
2025/12/31 09:50:42 [emerg] 14167#14167: unknown directive "sudo" in /etc/nginx/nginx.conf:3
2025/12/31 09:55:01 [emerg] 14391#14391: unexpected end of file, expecting ";" or "}" in /etc/nginx/nginx.conf:28
2025/12/31 10:14:11 [emerg] 15455#15455: cannot load certificate "/etc/nginx/ssl/nginx.crt": BIO_new_file() failed (SS
L: error:02001002:system library:fopen:No such file or directory:fopen('/etc/nginx/ssl/nginx.crt','r') error:2006D080:
BIO routines:BIO_new_file:no such file)
2025/12/31 10:39:46 [emerg] 16887#16887: "proxy_cache" zone "my_cache" is unknown in /etc/nginx/nginx.conf:25
2025/12/31 11:27:16 [emerg] 19755#19755: no servers in upstream "backend_servers" in /etc/nginx/conf.d/backend.conf:2
2025/12/31 11:27:16 [emerg] 19761#19761: no servers in upstream "backend_servers" in /etc/nginx/conf.d/backend.conf:2
2025/12/31 11:29:05 [emerg] 19859#19859: no servers in upstream "backend_servers" in /etc/nginx/conf.d/backend.conf:2
2025/12/31 11:30:13 [notice] 19935#19935: signal process started
```

Access Log:



```
[ec2-user@ip-10-0-10-86 ~]$ sudo tail -50 /var/log/nginx/access.log
103.229.252.83 - - [31/Dec/2025:10:03:51 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:03:52 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "http://44.208.36.190/" "Mozilla/5
.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:14:50 +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" "-"
103.229.252.83 - - [31/Dec/2025:10:25:20 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:25:20 +0000] "GET /favicon.ico HTTP/1.1" 404 196 "https://44.208.36.190/" "Mozilla/
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:26:36 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:26:41 +0000] "GET / HTTP/1.1" 200 1150 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:27:15 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:27:40 +0000] "GET / HTTP/1.1" 200 1150 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:28:51 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:28:54 +0000] "GET / HTTP/1.1" 200 1150 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36" "-"
103.229.252.83 - - [31/Dec/2025:10:28:56 +0000] "GET / HTTP/1.1" 200 1152 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x6
```

Processes running in nginx:

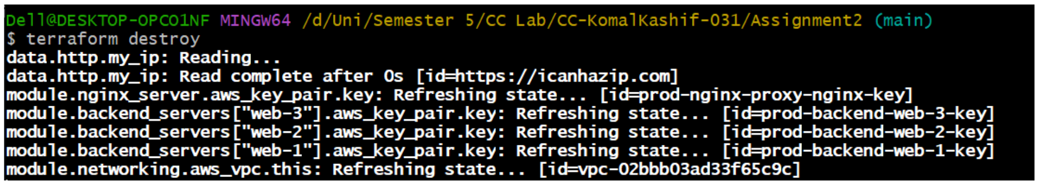


```
[ec2-user@ip-10-0-10-86 ~]$ ps aux | grep nginx
root      17069  0.0  0.6  51268  6212 ?        Ss   10:42   0:00 nginx: master process /usr/sbin/nginx
nginx     19936  0.0  0.6  51688  6628 ?        S    11:30   0:00 nginx: worker process
nginx     19937  0.0  0.6  51688  6628 ?        S    11:30   0:00 nginx: worker process
nginx     19938  0.0  0.3  51464  3280 ?        S    11:30   0:00 nginx: cache manager process
ec2-user 21814  0.0  0.0 119420   916 pts/1    R+   12:03   0:00 grep --color=auto nginx
```

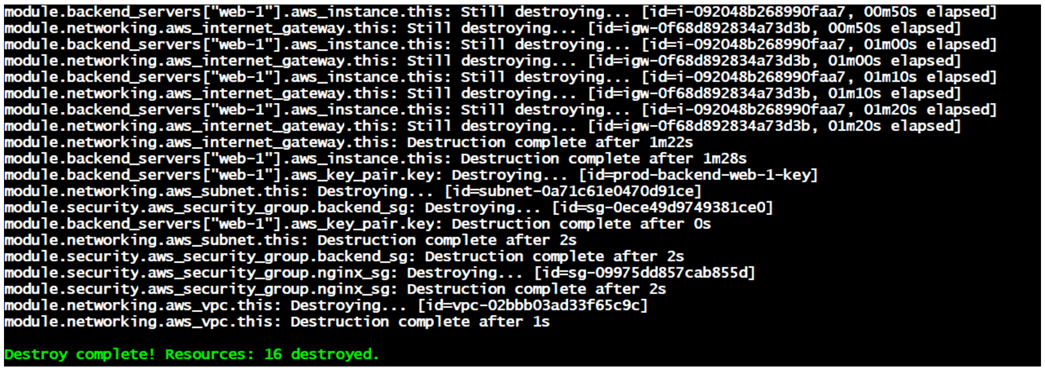
- **Part 6: Documentation and Cleanup:**
    - **6.2. Infrastructure cleanup**

  Properly destroy all resources and verify cleanup.

terraform destroy

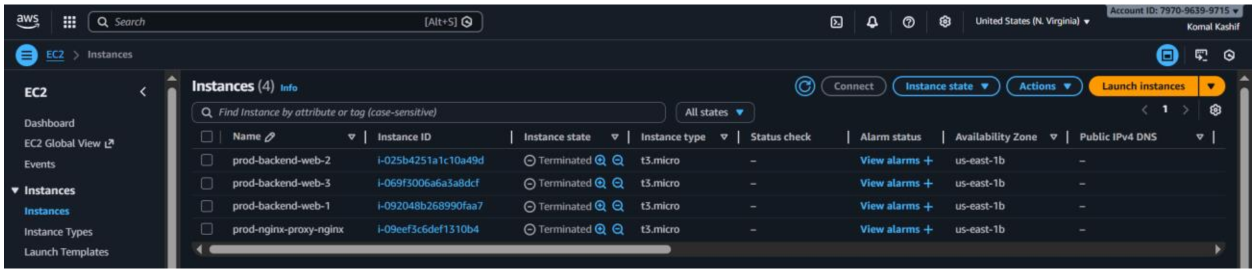


```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ terraform destroy
data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://icanhazip.com]
module.nginx_server.aws_key_pair.key: Refreshing state... [id=prod-nginx-proxy-nginx-key]
module.backend_servers["web-3"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-3-key]
module.backend_servers["web-2"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-2-key]
module.backend_servers["web-1"].aws_key_pair.key: Refreshing state... [id=prod-backend-web-1-key]
module.networking.aws_vpc.this: Refreshing state... [id=vpc-02bbb03ad33f65c9c]
```



```
module.backend_servers["web-1"].aws_instance.this: Still destroying... [id=i-092048b268990faa7, 00m50s elapsed]
module.networking.aws_internet_gateway.this: Still destroying... [id=igw-0f68d892834a73d3b, 00m50s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still destroying... [id=i-092048b268990faa7, 01m00s elapsed]
module.networking.aws_internet_gateway.this: Still destroying... [id=igw-0f68d892834a73d3b, 01m00s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still destroying... [id=i-092048b268990faa7, 01m10s elapsed]
module.networking.aws_internet_gateway.this: Still destroying... [id=igw-0f68d892834a73d3b, 01m10s elapsed]
module.backend_servers["web-1"].aws_instance.this: Still destroying... [id=i-092048b268990faa7, 01m20s elapsed]
module.networking.aws_internet_gateway.this: Still destroying... [id=igw-0f68d892834a73d3b, 01m20s elapsed]
module.networking.aws_internet_gateway.this: Destruction complete after 1m22s
module.backend_servers["web-1"].aws_instance.this: Destruction complete after 1m28s
module.backend_servers["web-1"].aws_key_pair.key: Destroying... [id=prod-backend-web-1-key]
module.networking.aws_subnet.this: Destroying... [id=subnet-0a71c61e0470d91ce]
module.security.aws_security_group.backend_sg: Destroying... [id=sg-0ece49d9749381ce0]
module.backend_servers["web-1"].aws_key_pair.key: Destruction complete after 0s
module.networking.aws_subnet.this: 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-09975dd857cab855d]
module.security.aws_security_group.nginx_sg: Destruction complete after 2s
module.networking.aws_vpc.this: Destroying... [id=vpc-02bbb03ad33f65c9c]
module.networking.aws_vpc.this: Destruction complete after 1s

Destroy complete! Resources: 16 destroyed.
```

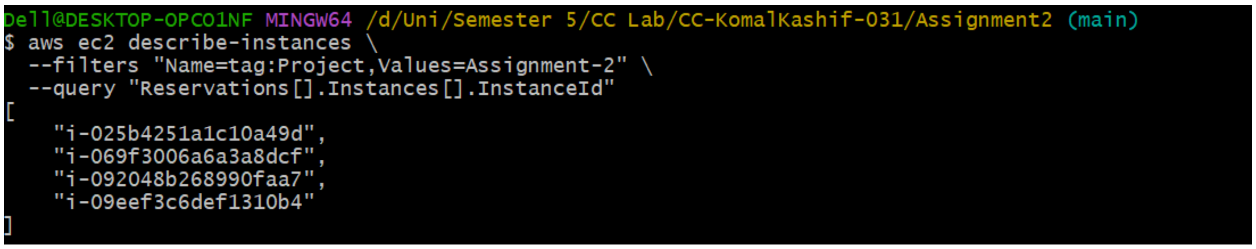
cat terraform.tfstate



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
☐	prod-backend-web-2	i-025b4251a1c10a49d	Terminated	t3.micro	–	View alarms +	us-east-1b	–
☐	prod-backend-web-3	i-069f3006a6a3a8dcf	Terminated	t3.micro	–	View alarms +	us-east-1b	–
☐	prod-backend-web-1	i-092048b268990faa7	Terminated	t3.micro	–	View alarms +	us-east-1b	–
☐	prod-nginx-proxy-nginx	i-09eef3c6def1310b4	Terminated	t3.micro	–	View alarms +	us-east-1b	–

The instances are still visible on the Console but are Terminated which means they are not operable and hence destroyed.

aws ec2 describe-instances --filters "Name=tag:Project,Values=Assignment-2" --query "Reservations[].Instances[].InstanceId"



```
Dell@DESKTOP-OPCO1NF MINGW64 /d/Uni/Semester 5/CC Lab/CC-KomalKashif-031/Assignment2 (main)
$ aws ec2 describe-instances \
  --filters "Name=tag:Project,Values=Assignment-2" \
  --query "Reservations[].Instances[].InstanceId"
[
    "i-025b4251a1c10a49d",
    "i-069f3006a6a3a8dcf",
    "i-092048b268990faa7",
    "i-09eef3c6def1310b4"
]
```

These are the instance ids of the instances from above. As they are destroyed but still in the console, their ids are here.

4) Testing Results

- ### 4.1 Load Balancing Tests

The load balancing functionality was tested by repeatedly refreshing the application URL accessed through the Nginx server. Each backend server displays unique system information, allowing easy identification of which server handled the request.

Results confirmed that traffic alternated between web-1 and web-2, while web-3 remained inactive under normal conditions, validating the correct implementation of the load balancing strategy.

- ### 4.2 Cache Performance Tests

Nginx caching was tested using browser developer tools. The first request returned X-Cache-Status: MISS, while subsequent requests returned HIT, confirming effective caching behavior.

This significantly reduced backend load and improved response times.

- ### 4.3 High Availability Tests

To test failover:

- Apache services on web-1 and web-2 were stopped
- Nginx logs were monitored
- Traffic was successfully routed to web-3

This validated the backup server configuration.

- ### 4.4 Security Tests

- HTTPS encryption verified using browser certificate inspection
- HTTP requests redirected to HTTPS
- Security headers confirmed in response headers
- Backend servers inaccessible directly from the internet

- ### 4.5 Performance Metrics

- Reduced latency due to caching
- Stable response times under load
- Efficient traffic distribution

5) Challenges & Solutions

Challenges Encountered

- Managing dynamic backend IPs
- Nginx SSL configuration

- Terraform module dependencies
- Security group misconfigurations

Solutions

- Used Terraform outputs for backend IP mapping
- Implemented self-signed certificates
- Modularized Terraform code
- Applied least-privilege security rules

Lessons Learned

- Importance of modular Infrastructure as Code
- Real-world Nginx troubleshooting
- Security-first cloud design principles

6) Conclusion

This assignment successfully demonstrated the deployment of a secure, automated, and highly available web infrastructure using AWS, Terraform, and Nginx.

Summary of Work Completed

- Automated infrastructure provisioning
- Configured secure load balancing
- Implemented caching and failover
- Verified performance and security

Skills Acquired

- Terraform module design
- AWS networking and security
- Nginx reverse proxy configuration
- Cloud troubleshooting and testing

Future Improvements

- Auto-scaling groups
- Managed load balancer (ALB)
- CI/CD integration
- Monitoring with CloudWatch