Assignment 2

Lab title: Advanced Terraform & Nginx Multi-Tier Architecture

Submitted to: Engr. Mohammad Shoaib

Submitted by: Anara Hayat

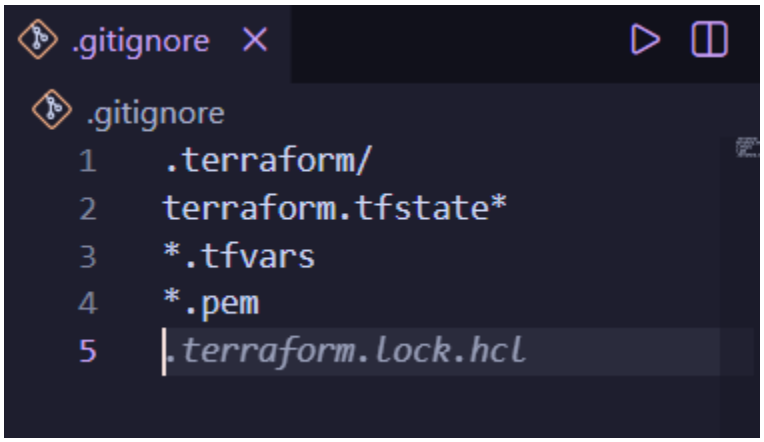
Reg#No: 2023-BSE-008

Requirements

Part 1: Infrastructure Setup

1.1 Project Structure

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

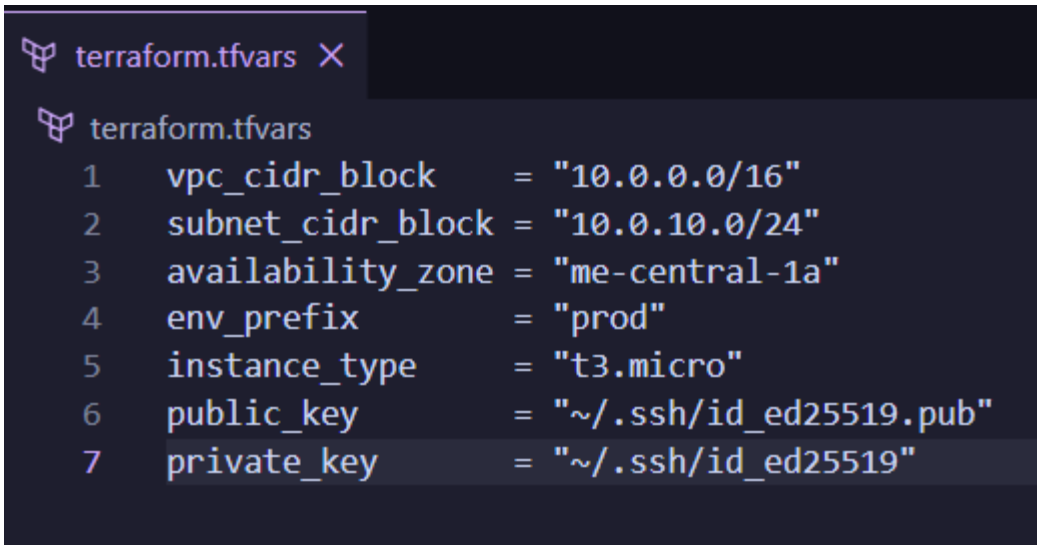


1.2 Variable Configuration (5 marks)

Define all required variables in variables.tf:

variables.tf

```hcl
variable "aws_region" {
  description = "AWS region"
  type        = string
  default     = "me-central-1"
}

variable "vpc_cidr_block" {
  description = "CIDR block for VPC"
  type        = string
  validation {
    condition     = can(cidrnetmask(var.vpc_cidr_block))
    error_message = "Invalid VPC CIDR block"
  }
}

variable "subnet_cidr_block" {
  description = "CIDR block for subnet"
  type        = string
  validation {
    condition     = can(cidrnetmask(var.subnet_cidr_block))
    error_message = "Invalid subnet CIDR block"
  }
}

variable "availability_zone" {
  description = "Availability Zone"
  type        = string
}

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

```



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

Tasks:

- Create VPC resource
- Create subnet with map_public_ip_on_launch = true
- Create and attach Internet Gateway
- Configure routing table
- Add proper tags to all resources using env_prefix

dules > networking > main.tf

```hcl
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr_block
  tags       = merge(var.common_tags, { Name = "${var.env_pre
}

resource "aws_subnet" "this" {
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.subnet_cidr_block
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true
  tags                    = merge(var.common_tags, { Name = "
}

resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id
  tags   = merge(var.common_tags, { Name = "${var.env_prefix}
}

resource "aws_route_table" "this" {
  vpc_id = aws_vpc.this.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.this.id
  }
}

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

modules > networking > outputs.tf

```hcl
output "vpc_id" { value = aws_vpc.this.id }
output "subnet_id" { value = aws_subnet.this.id }
```

1.4 Security Module

Create a security module that provisions:

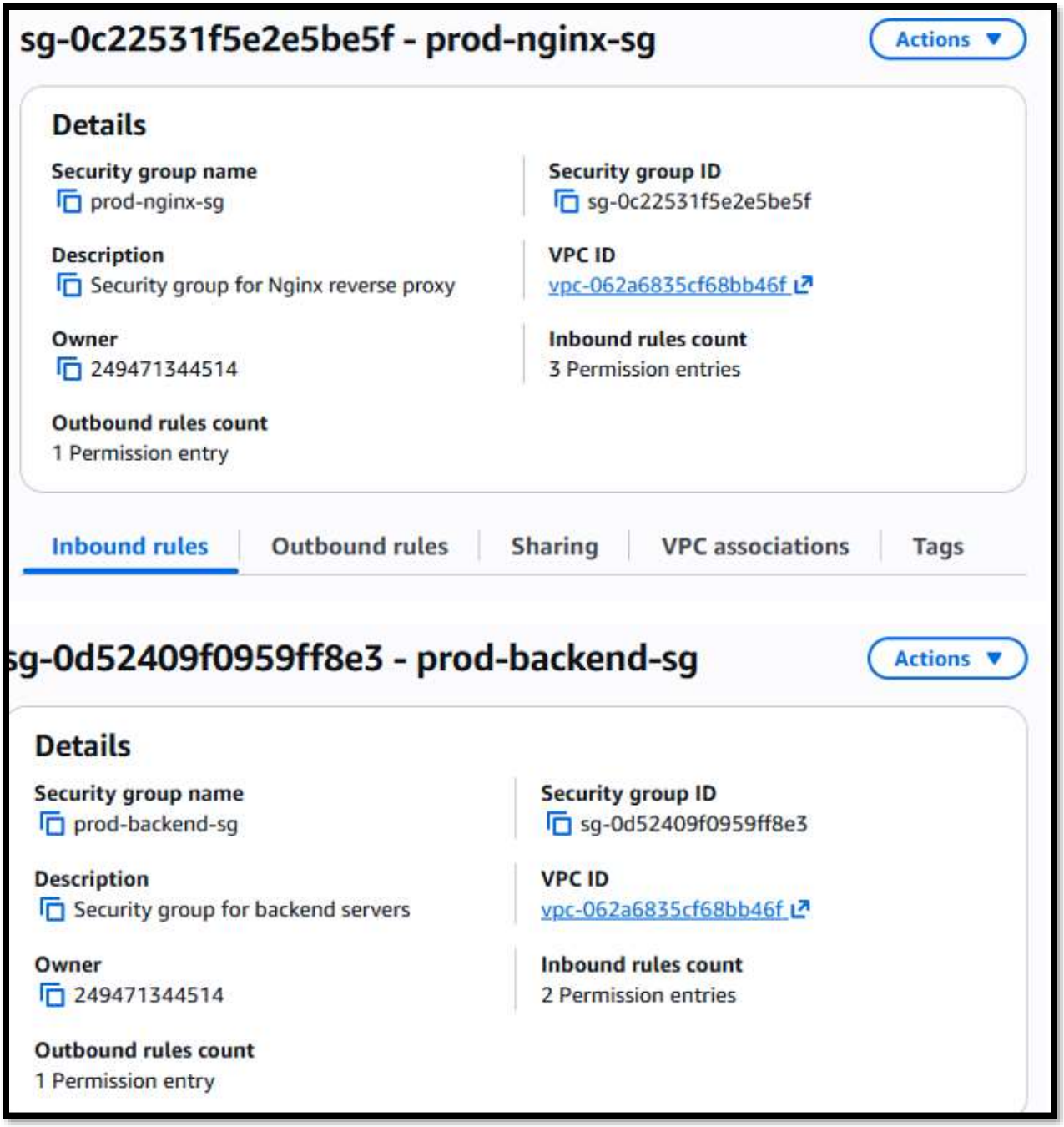
Tasks:

- Create Nginx security group with appropriate rules

- Create backend security group with appropriate rules
- Use security group IDs for backend ingress (not CIDR blocks)
- Add descriptive names and tags

nodules > security > main.tf

```hcl
resource "aws_security_group" "nginx" {
  vpc_id = var.vpc_id
  name   = "${var.env_prefix}-nginx-sg"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.my_ip]
  }

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

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

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

  tags = var.common_tags
}
```



1.5 Locals Configuration (5 marks)

Create locals.tf with:

- Dynamic IP detection for my_ip
- Resource naming conventions
- Common tags

- Backend server configurations

Tasks:

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

```
# Local values
locals {
  # Get current public IP with /32 CIDR notation
  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"
    CreatedDate = timestamp()
  }

  # 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"
```

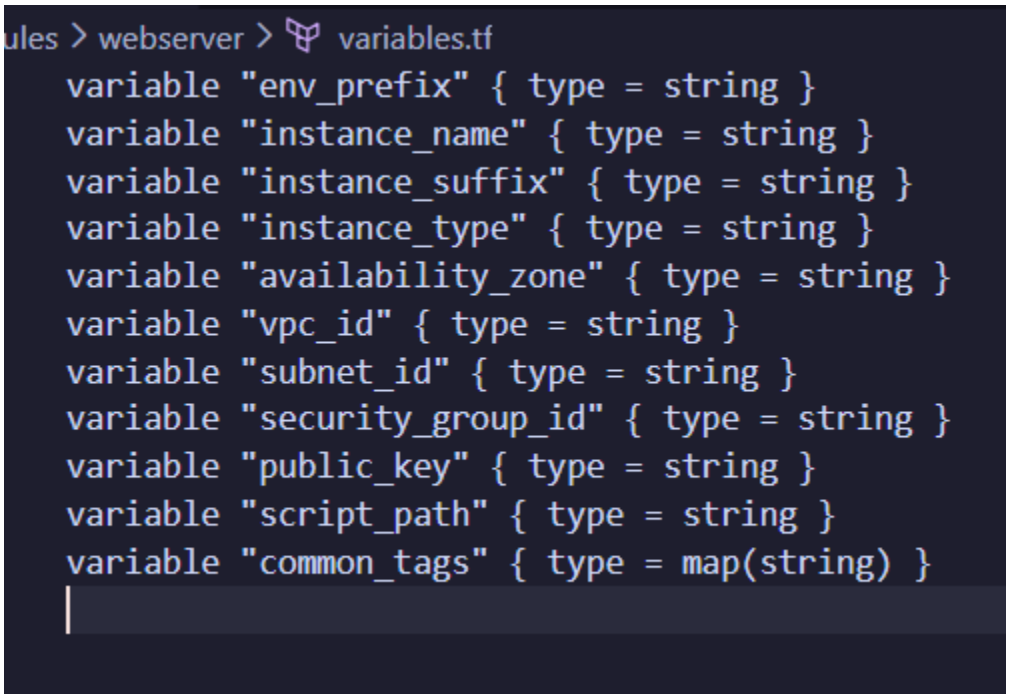
Part 2: Webserver Module

2.1 Module Design (10 marks)

Create a reusable webserver module in modules/webserver/

Tasks:

- Create variables. tf with all required variables
- Create main.tf with key pair and instance resources
- Create outputs.tf with all required outputs
- Ensure module is reusable for both Nginx and backend servers



main.tf

modules > webserver > main.tf

```hcl
data "aws_ami" "amazon_linux" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["al2023-ami-*-x86_64"]
  }
}

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

resource "aws_instance" "this" {
  ami                    = data.aws_ami.amazon_linux.id
  instance_type          = var.instance_type
  availability_zone      = var.availability_zone
  subnet_id              = var.subnet_id
  vpc_security_group_ids = [var.security_group_id]
  key_name               = aws_key_pair.this.key_name
  user_data              = file(var.script_path)

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

nodules > webserver > outputs.tf

```hcl
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 (5 marks)

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:

Tasks:

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

main.tf

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

# BACKEND SERVERS
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
  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
  common_tags         = local.common_tags
}
```

Part 3: Server Configuration Scripts (20 marks)

3.1 Apache Backend Server Script (10 marks)

Create scripts/apache-setup.sh that:

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

Tasks:

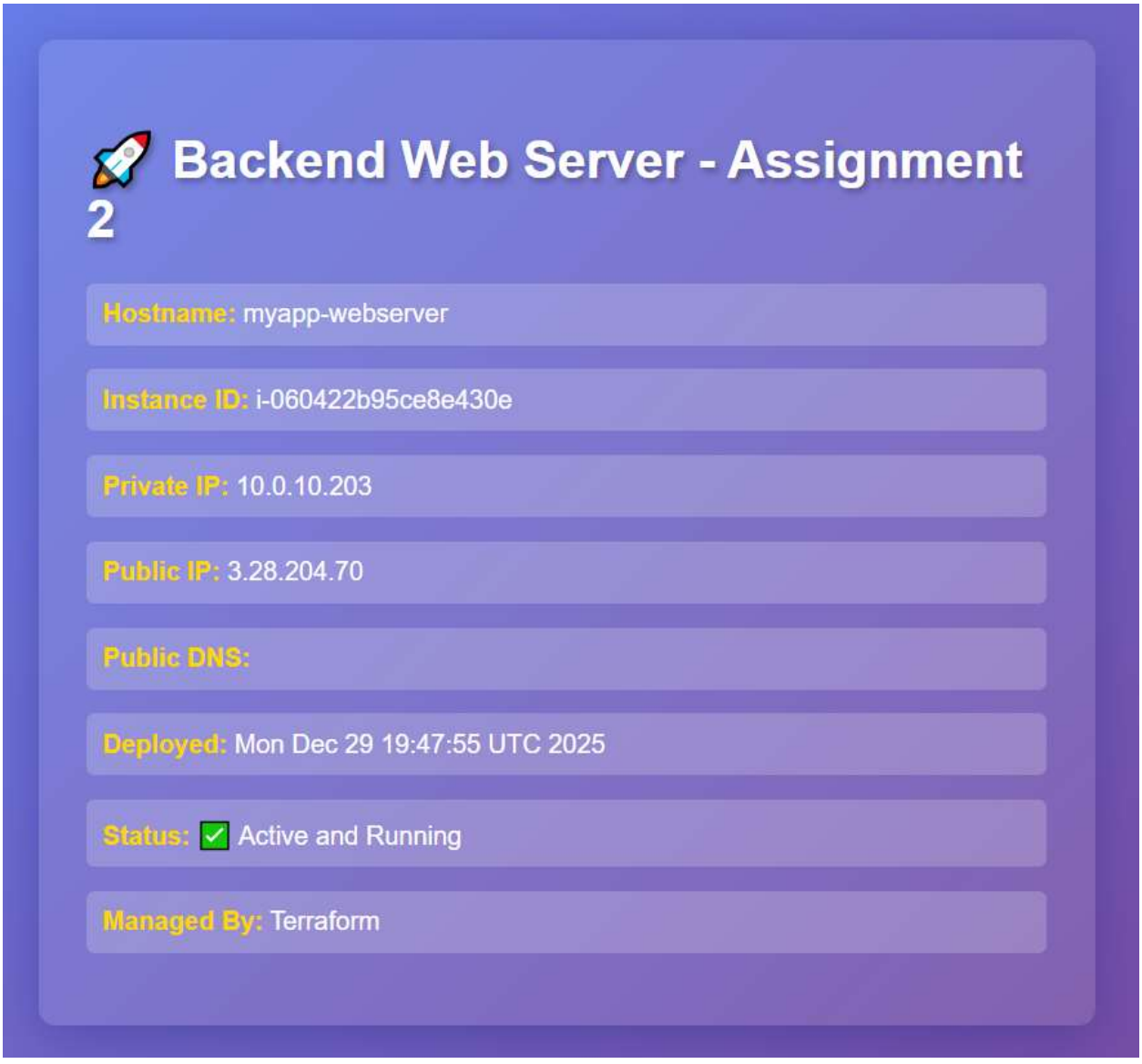
- Create the script with all required features
- Ensure it uses IMDSv2 for metadata
- Create visually appealing HTML output
- Make script executable
- Test script independently


apache-setup.sh
scripts > apache-setup.sh
#!/bin/bash
set -e
Update system
yum update -y
Install Apache
yum install httpd -y
Start and enable Apache
systemctl start httpd
systemctl enable httpd
# Get metadata token (IMDSv2)
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/tok
-H "X-aws-ec2-metadata-token-ttl-seconds: 21600")
# Get instance metadata
PRIVATE_IP=$(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>

Resources Created:

- 1 Nginx reverse proxy server (Public IP: 3.29.231.86)
- 3 Backend web servers:
  - web-1: 10.0.10.203 (Public: 3.28.204.70)
  - web-2: 10.0.10.146 (Public: 158.252.81.36)
  - web-3: 10.0.10.83 (Public: 3.29.64.238)
- VPC with networking infrastructure (subnet, internet gateway, route table)
- Security groups for SSH and HTTP/HTTPS access

The error was fixed by changing the availability zone from `me-central-1` to `me-central-1a` in the terraform.tfvars file.



3.2 Nginx Server Setup Script (10 marks)

Create scripts/nginx-setup.sh that:

- Updates system packages
- Installs Nginx
- Generates self-signed SSL certificate
- Configures Nginx with:
    - HTTPS on port 443
    - HTTP to HTTPS redirect
    - Upstream backend servers (web-1, web-2, web-3)
    - Load balancing configuration
    - Caching configuration
    - Proper logging
    - Security headers

Tasks:

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

scripts > nginx-setup.sh

```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/to
  -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
```

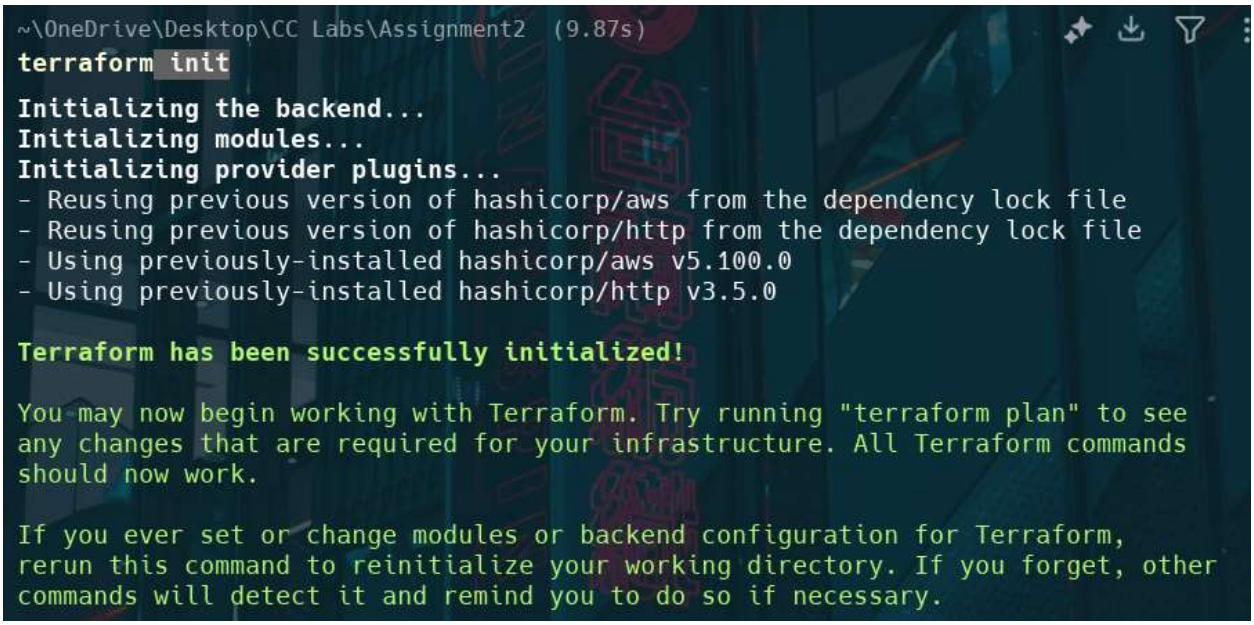
Part 4: Infrastructure Deployment (15 marks)

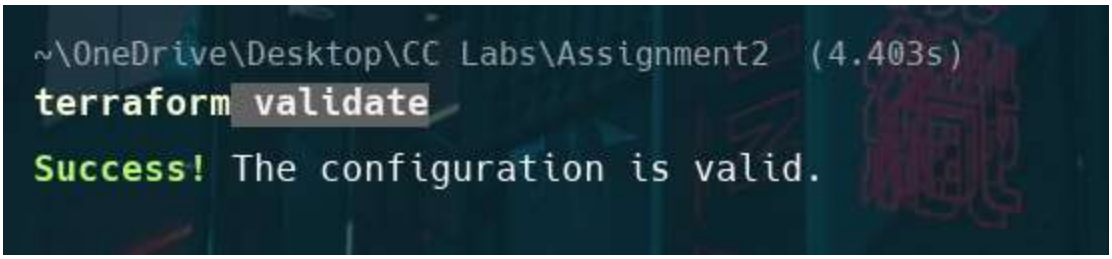
4.1 Initial Deployment (5 marks)

Deploy the infrastructure using Terraform.

Tasks:

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





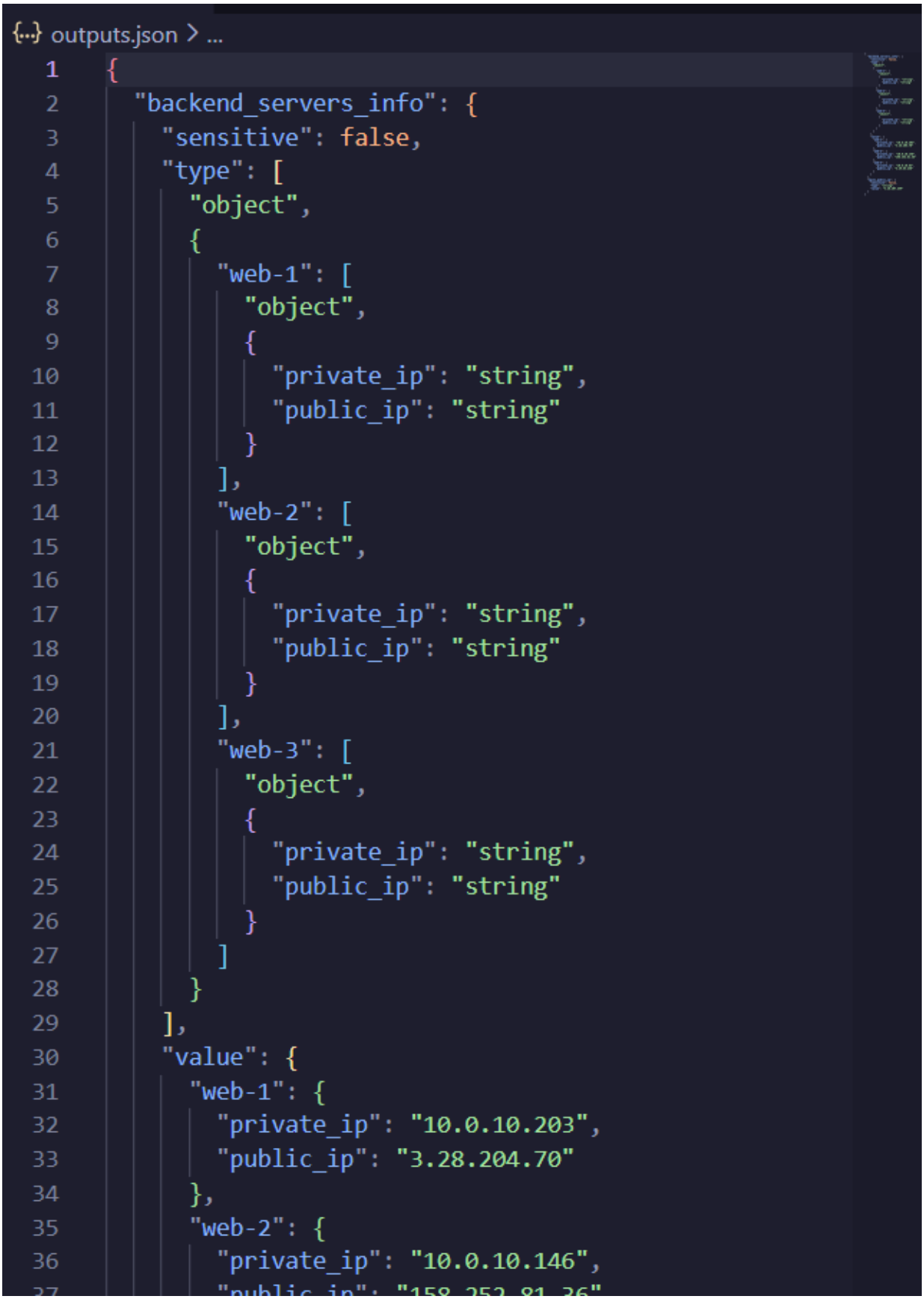

```
~\OneDrive\Desktop\CC Labs\Assignment2  (8.709s)
terraform plan
data.http.my_ip: Read complete after 0s [id=https://ipv4.icanhazip.com]
module.nginx_server.data.aws_ami.amazon_linux: Reading...
module.backend_servers["web-2"].data.aws_ami.amazon_linux: Reading...
module.backend_servers["web-3"].aws_key_pair.this: Refreshing state... [id=prod-3-k
ey]
module.backend_servers["web-2"].aws_key_pair.this: Refreshing state... [id=prod-2-k
ey]
module.backend_servers["web-1"].aws_key_pair.this: Refreshing state... [id=prod-1-k
ey]
module.backend_servers["web-1"].data.aws_ami.amazon_linux: Reading...
module.networking.aws_vpc.this: Refreshing state... [id=vpc-062a6835cf68bb46f]
module.backend_servers["web-3"].data.aws_ami.amazon_linux: Reading...
module.nginx_server.aws_key_pair.this: Refreshing state... [id=prod-nginx-key]
module.backend_servers["web-3"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.backend_servers["web-2"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.backend_servers["web-1"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.nginx_server.data.aws_ami.amazon_linux: Read complete after 1s [id=ami-009ce
8169fc88edf5]
module.networking.aws_internet_gateway.this: Refreshing state... [id=igw-01ca0e5832
13bc718]
module.networking.aws_subnet.this: Refreshing state... [id=subnet-0f4ce09fd93138b7f
]
module.security.aws_security_group.nginx: Refreshing state... [id=sg-0c22531f5e2e5b
e5f]
```

```
terraform apply -auto-approve

data.http.my_ip: Reading...
data.http.my_ip: Read complete after 0s [id=https://ipv4.icanhazip.com]
module.backend_servers["web-3"].data.aws_ami.amazon_linux: Reading...
module.nginx_server.aws_key_pair.this: Refreshing state... [id=prod-nginx-key]
module.backend_servers["web-2"].data.aws_ami.amazon_linux: Reading...
module.backend_servers["web-1"].data.aws_ami.amazon_linux: Reading...
module.backend_servers["web-2"].aws_key_pair.this: Refreshing state... [id=prod-2-k
ey]
module.nginx_server.data.aws_ami.amazon_linux: Reading...
module.backend_servers["web-1"].aws_key_pair.this: Refreshing state... [id=prod-1-k
ey]
module.backend_servers["web-3"].aws_key_pair.this: Refreshing state... [id=prod-3-k
ey]
module.networking.aws_vpc.this: Refreshing state... [id=vpc-062a6835cf68bb46f]
module.backend_servers["web-1"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.nginx_server.data.aws_ami.amazon_linux: Read complete after 1s [id=ami-009ce
8169fc88edf5]
module.backend_servers["web-3"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.backend_servers["web-2"].data.aws_ami.amazon_linux: Read complete after 1s [
id=ami-009ce8169fc88edf5]
module.networking.aws_internet_gateway.this: Refreshing state... [id=igw-01ca0e5832
13bc718]
module.networking.aws_subnet.this: Refreshing state... [id=subnet-0f4ce09fd93138b7f
]
module.security.aws_security_group.nginx: Refreshing state... [id=sg-0c22531f5e2e5b
e5f]
module.networking.aws_route_table.this: Refreshing state... [id=rtb-01b3dc985c94023
```

4.2 Output Configuration (5 marks)

Create comprehensive outputs in outputs.tf

```
~\OneDrive\Desktop\CC Labs\Assignment2  (2.334s)
terraform output

backend_servers_info = {
  "web-1" = {
    "private_ip" = "10.0.10.203"
    "public_ip" = "3.28.204.70"
  }
  "web-2" = {
    "private_ip" = "10.0.10.146"
    "public_ip" = "158.252.81.36"
  }
  "web-3" = {
    "private_ip" = "10.0.10.83"
    "public_ip" = "3.29.64.238"
  }
}
nginx_public_ip = "3.28.207.249"
```

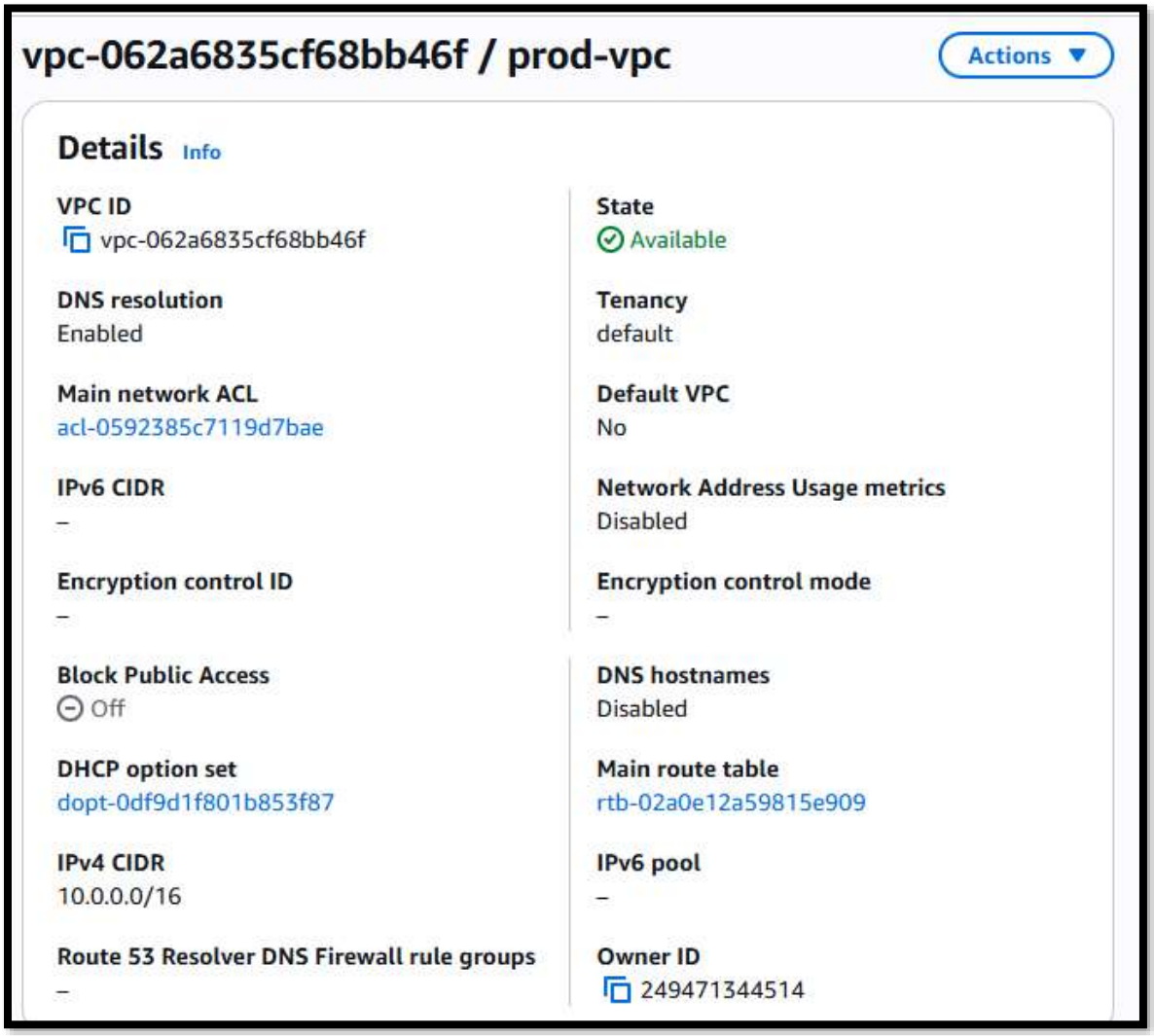

outputs.json > ...
{
"backend_servers_info": {
"sensitive": false,
"type": [
"object",
{
"web-1": [
"object",
{
"private_ip": "string",
"public_ip": "string"
}
],
"web-2": [
"object",
{
"private_ip": "string",
"public_ip": "string"
}
],
"web-3": [
"object",
{
"private_ip": "string",
"public_ip": "string"
}
]
}
],
"value": {
"web-1": {
"private_ip": "10.0.10.203",
"public_ip": "3.28.204.70"
},
"web-2": {
"private_ip": "10.0.10.146",

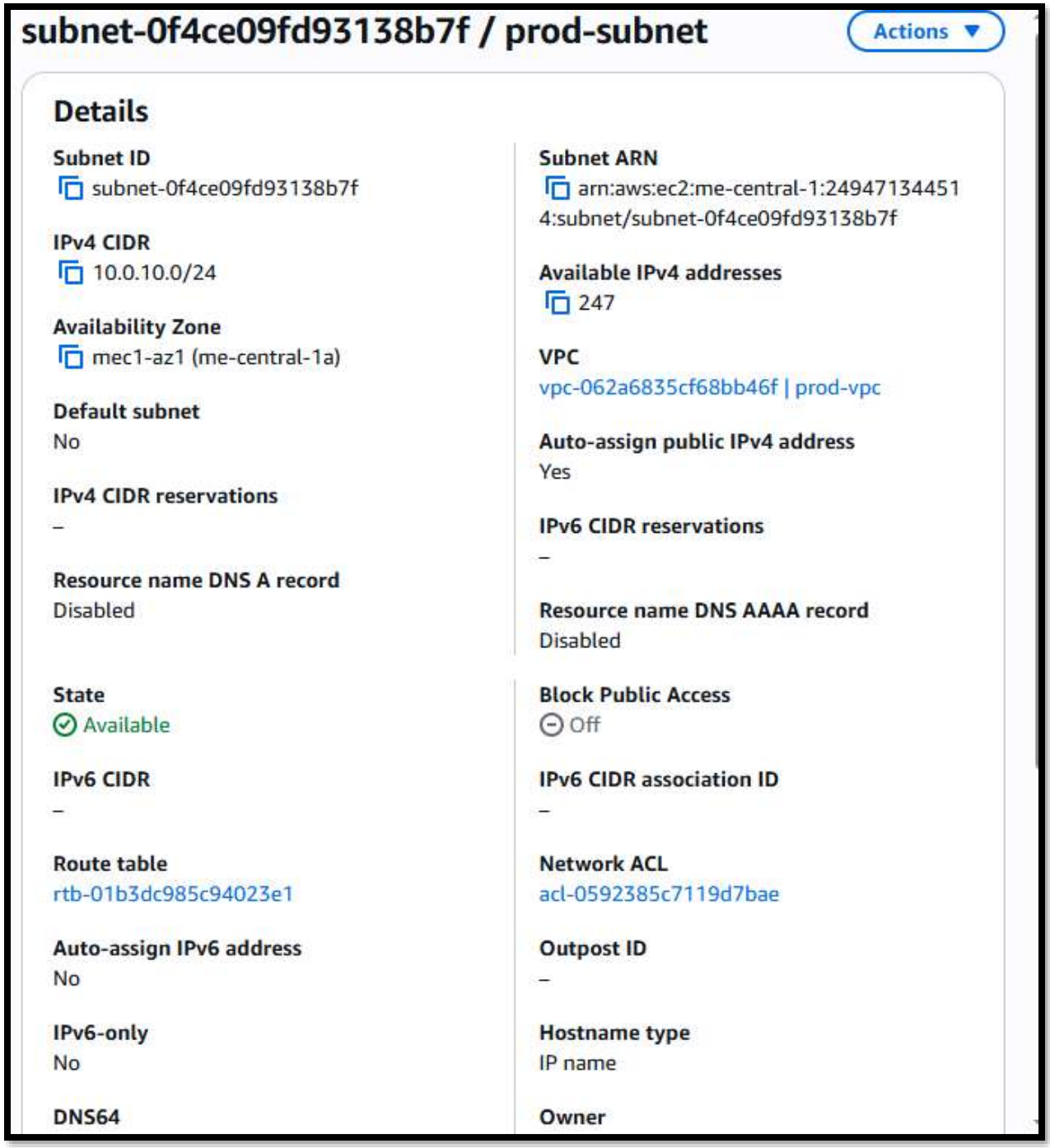
4.3 AWS Console Verification (5 marks)

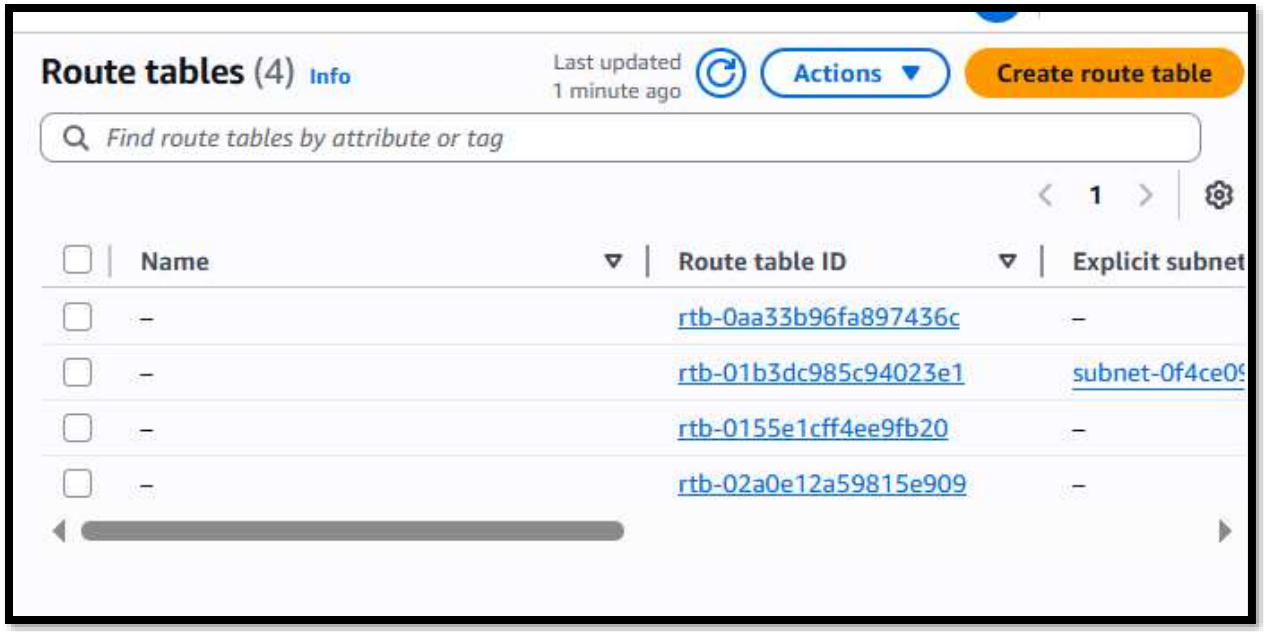
Verify all resources in AWS Console.

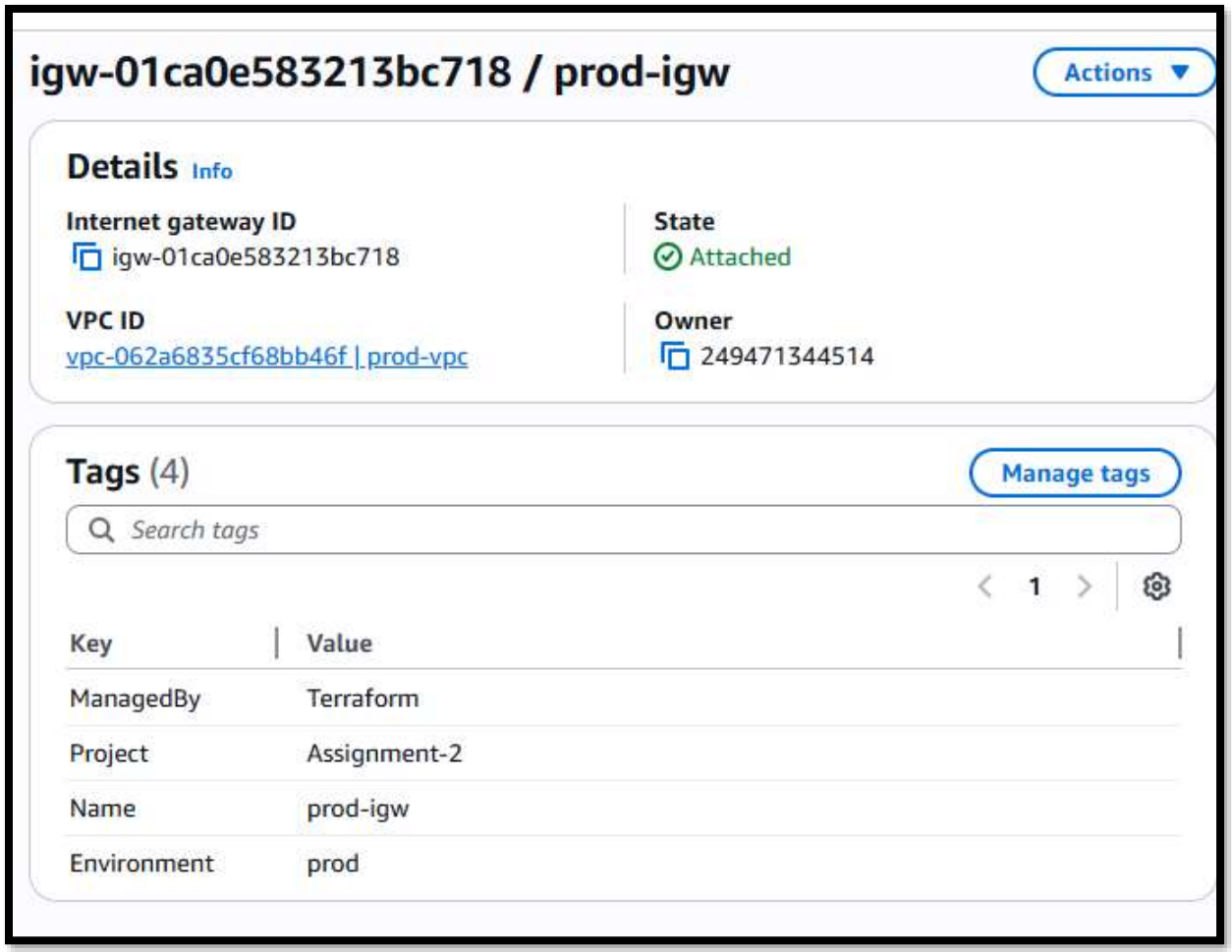
Tasks:

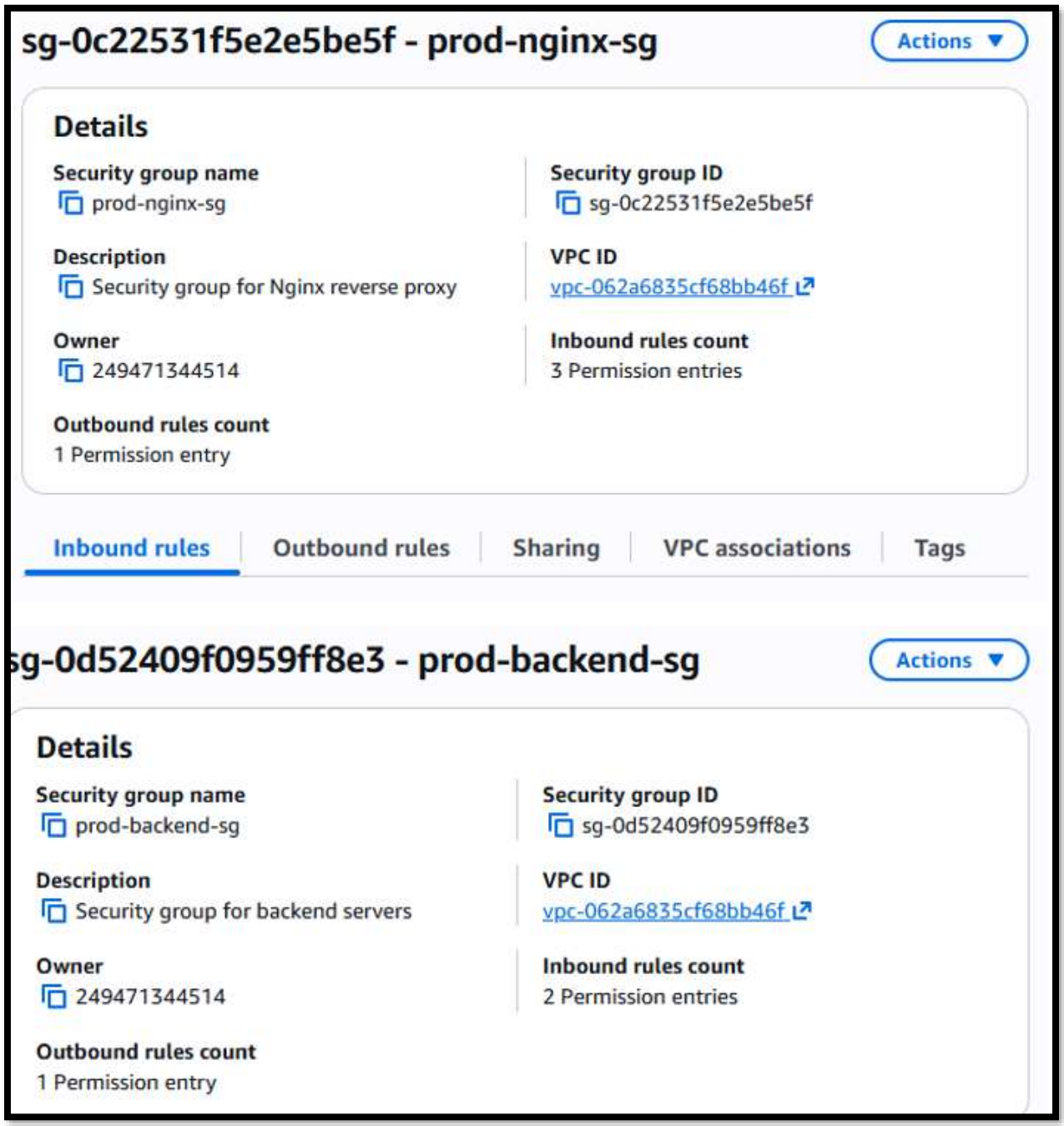
- Verify VPC created
- Verify Subnet created
- Verify Internet Gateway attached
- Verify Route Table configured
- Verify Security Groups created with correct rules
- Verify all 4 EC2 instances running
- Verify Key Pairs created

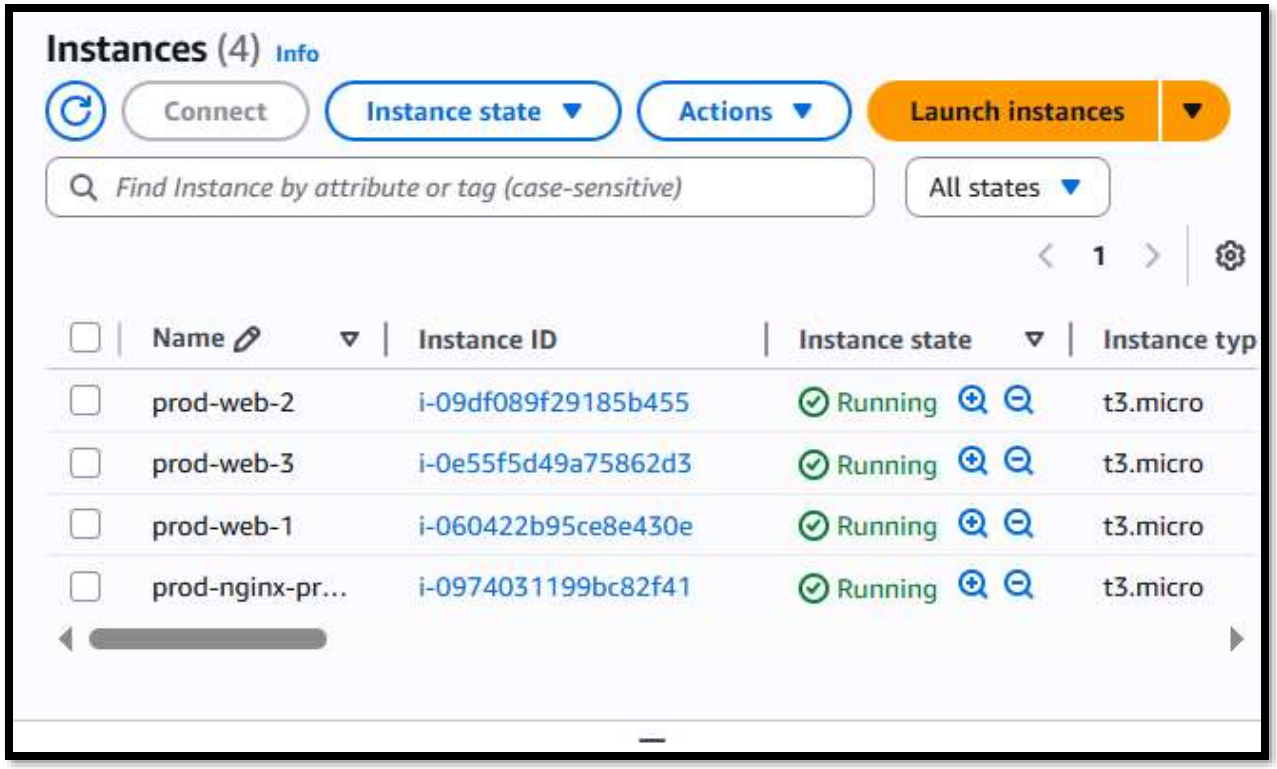



subnet-0f4ce09fd93138b7f / prod-subnet
Actions
Details
Subnet ID
subnet-0f4ce09fd93138b7f
IPv4 CIDR
10.0.10.0/24
Availability Zone
mec1-az1 (me-central-1a)
Default subnet
No
IPv4 CIDR reservations
–
Resource name DNS A record
Disabled
State
Available
IPv6 CIDR
–
Route table
rtb-01b3dc985c94023e1
Auto-assign IPv6 address
No
IPv6-only
No
DNS64
Subnet ARN
arn:aws:ec2:me-central-1:249471344514:subnet/subnet-0f4ce09fd93138b7f
Available IPv4 addresses
247
VPC
vpc-062a6835cf68bb46f | prod-vpc
Auto-assign public IPv4 address
Yes
IPv6 CIDR reservations
–
Resource name DNS AAAA record
Disabled
Block Public Access
Off
IPv6 CIDR association ID
–
Network ACL
acl-0592385c7119d7bae
Outpost ID
–
Hostname type
IP name
Owner


Route tables (4) Info
Last updated
1 minute ago
Actions
Create route table
Find route tables by attribute or tag
1
Name
Route table ID
Explicit subnet
– rtb-0aa33b96fa897436c –
– rtb-01b3dc985c94023e1 subnet-0f4ce0
– rtb-0155e1cff4ee9fb20 –
– rtb-02a0e12a59815e909 –


igw-01ca0e583213bc718 / prod-igw
Actions
Details Info
Internet gateway ID
igw-01ca0e583213bc718
State
Attached
VPC ID
vpc-062a6835cf68bb46f | prod-vpc
Owner
249471344514
Tags (4)
Manage tags
Search tags
1
Key
Value
ManagedBy Terraform
Project Assignment-2
Name prod-igw
Environment prod


sg-0c22531f5e2e5be5f - prod-nginx-sg
Actions
Details
Security group name
prod-nginx-sg
Security group ID
sg-0c22531f5e2e5be5f
Description
Security group for Nginx reverse proxy
VPC ID
vpc-062a6835cf68bb46f
Owner
249471344514
Inbound rules count
3 Permission entries
Outbound rules count
1 Permission entry
Inbound rules
Outbound rules
Sharing
VPC associations
Tags
sg-0d52409f0959ff8e3 - prod-backend-sg
Actions
Details
Security group name
prod-backend-sg
Security group ID
sg-0d52409f0959ff8e3
Description
Security group for backend servers
VPC ID
vpc-062a6835cf68bb46f
Owner
249471344514
Inbound rules count
2 Permission entries
Outbound rules count
1 Permission entry



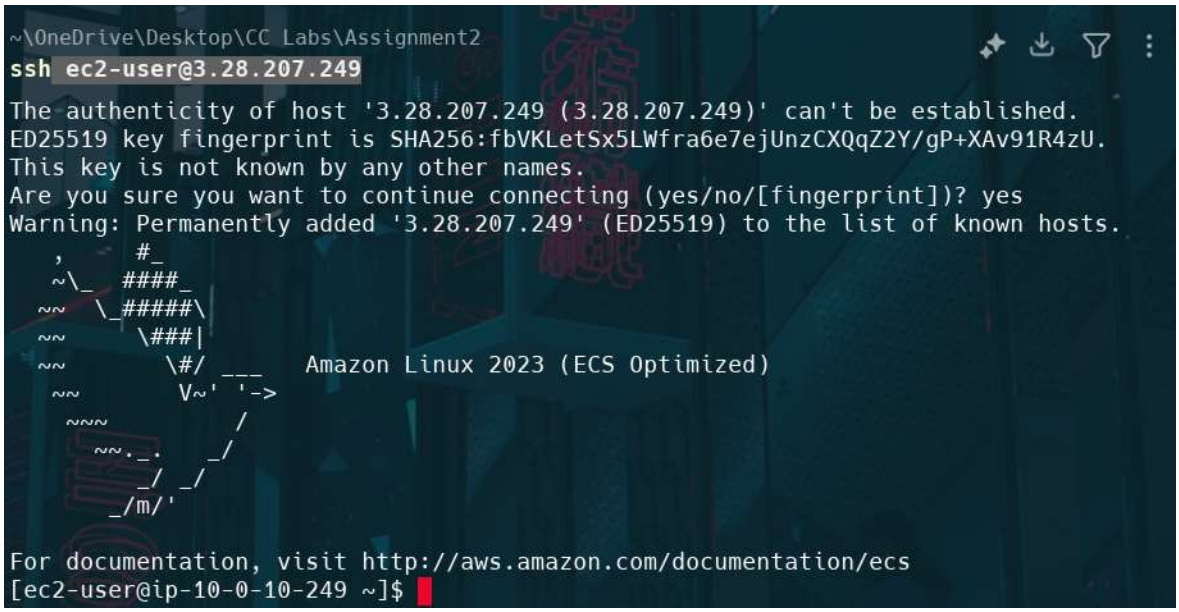
Part 5: Nginx Configuration & Testing (25 marks)

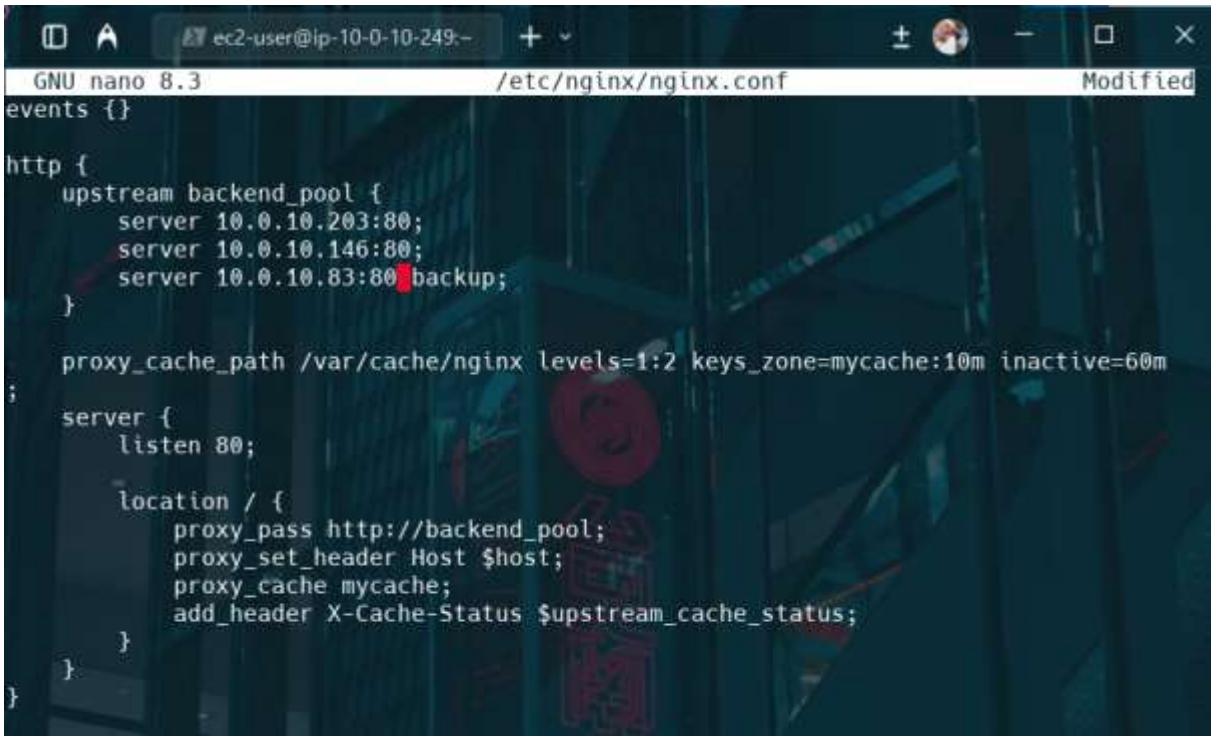
5.1 Update Nginx Backend Configuration (5 marks)

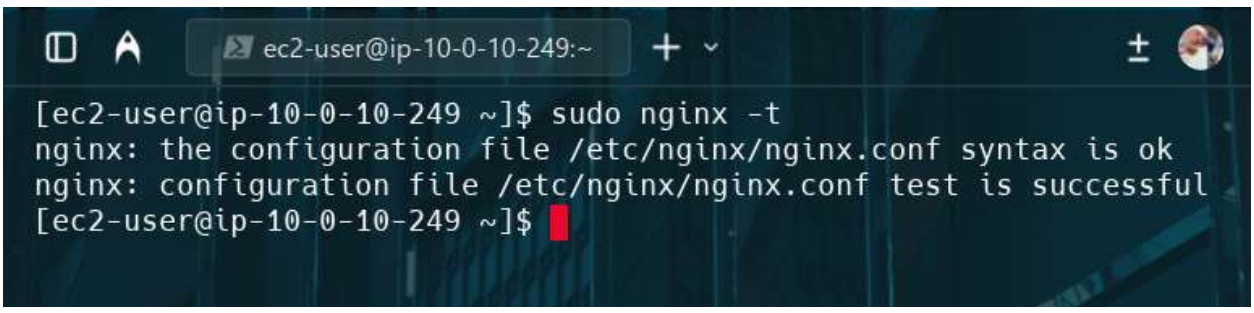
SSH into the Nginx server and update the configuration with actual backend IPs.

Tasks:

- SSH into Nginx server
- Edit /etc/nginx/nginx.conf
- Replace placeholder IPs with actual private IPs of backend servers
- Test Nginx configuration
- Restart Nginx service


~\OneDrive\Desktop\CC Labs\Assignment2
ssh ec2-user@3.28.207.249
The authenticity of host '3.28.207.249 (3.28.207.249)' can't be established.
ED25519 key fingerprint is SHA256:fbVKLetSx5LWfra6e7ejUnzCXQqZ2Y/gP+XAv91R4zU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.28.207.249' (ED25519) to the list of known hosts.
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
[ec2-user@ip-10-0-10-249 ~]$


ec2-user@ip-10-0-10-249:~
GNU nano 8.3 /etc/nginx/nginx.conf Modified
events {}
http {
upstream backend_pool {
server 10.0.10.203:80;
server 10.0.10.146:80;
server 10.0.10.83:80 backup;
}
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=mycache:10m inactive=60m
;
server {
listen 80;
location / {
proxy_pass http://backend_pool;
proxy_set_header Host $host;
proxy_cache mycache;
add_header X-Cache-Status $upstream_cache_status;
}
}
}


ec2-user@ip-10-0-10-249:~
[ec2-user@ip-10-0-10-249 ~]$ 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-10-0-10-249 ~]$

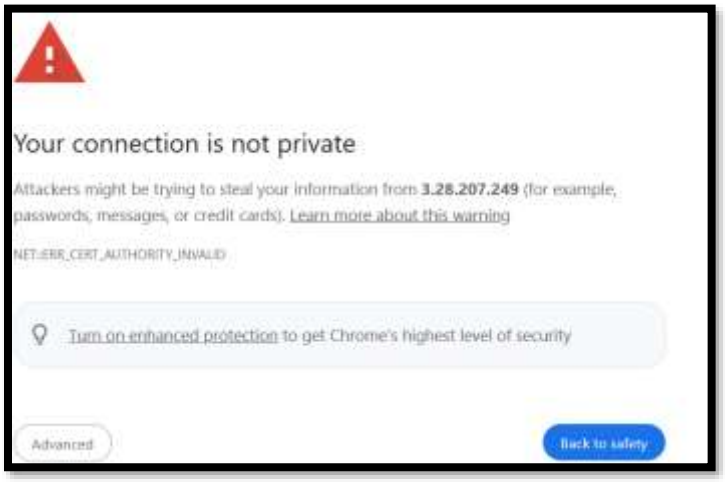
```
[ec2-user@ip-10-0-10-249 ~]$ sudo systemctl restart nginx
[ec2-user@ip-10-0-10-249 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disab>
     Active: active (running) since Mon 2025-12-29 21:15:03 UTC; 21s ago
    Process: 48557 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status>
    Process: 48558 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 48559 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 48560 (nginx)
      Tasks: 4 (limit: 1065)
     Memory: 3.0M
        CPU: 31ms
     CGroup: /system.slice/nginx.service
             ├─48560 "nginx: master process /usr/sbin/nginx"
             ├─48561 "nginx: worker process"
             ├─48562 "nginx: cache manager process"
             └─48563 "nginx: cache loader process"

Dec 29 21:15:03 ip-10-0-10-249.me-central-1.compute.internal systemd[1]: Starting >
Dec 29 21:15:03 ip-10-0-10-249.me-central-1.compute.internal nginx[48558]: nginx: >
Dec 29 21:15:03 ip-10-0-10-249.me-central-1.compute.internal nginx[48558]: nginx: >
Dec 29 21:15:03 ip-10-0-10-249.me-central-1.compute.internal systemd[1]: Started n>
lines 1-20/20 (END)
```

5.2 Test Load Balancing (5 marks)

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

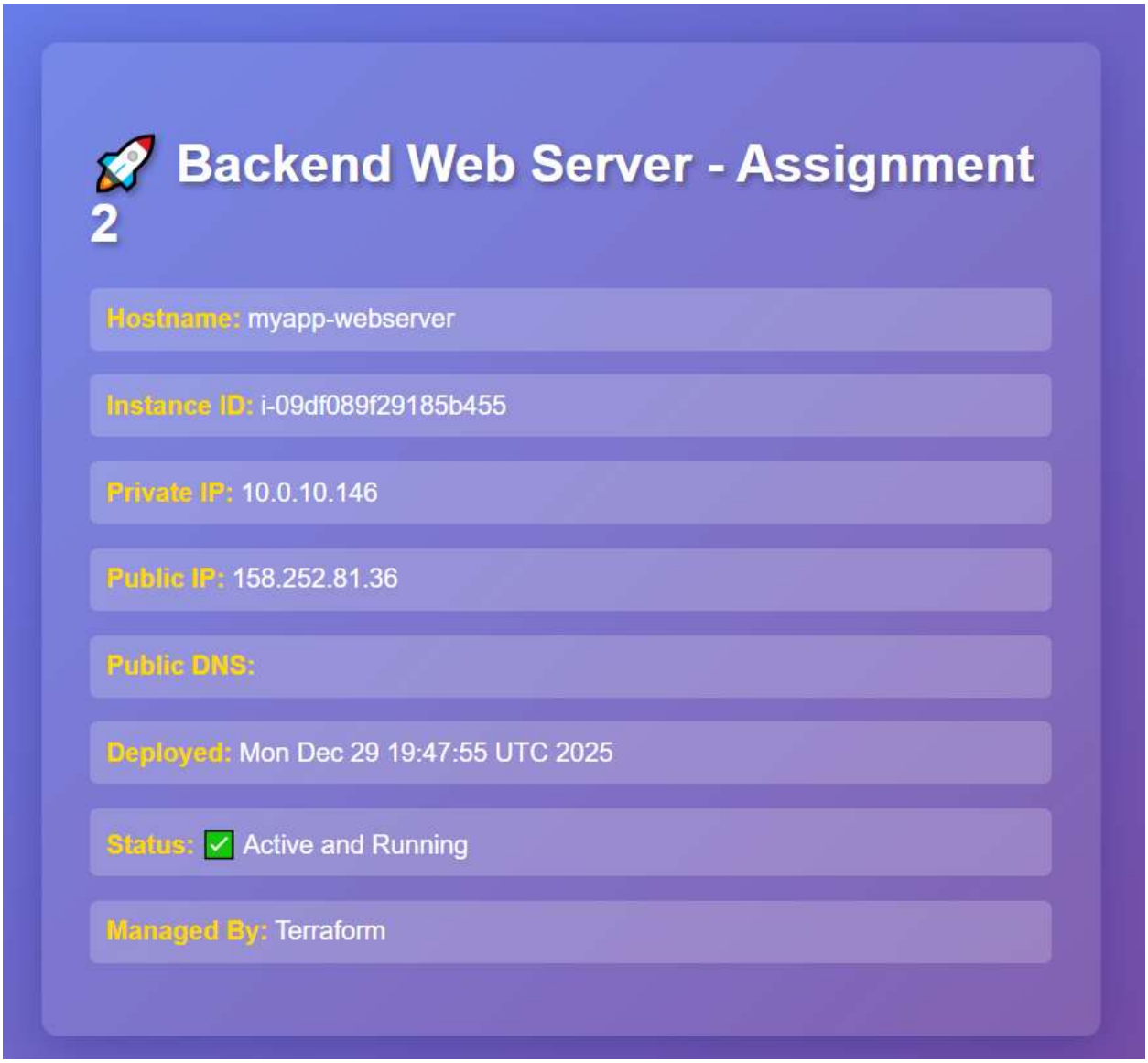
Tasks:

- Open browser to https://<nginx-public-ip>
- Accept the security warning for self-signed certificate
- Reload page multiple times (at least 10 times)
- Verify traffic alternates between web-1 and web-2
- Verify web-3 is NOT serving traffic (it's backup only)




Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-060422b95ce8e430e
Private IP: 10.0.10.203
Public IP: 3.28.204.70
Public DNS:
Deployed: Mon Dec 29 19:47:55 UTC 2025
Status: Active and Running
Managed By: Terraform


Backend Web Server - Assignment 2
Hostname: myapp-webserver
Instance ID: i-09df089f29185b455
Private IP: 10.0.10.146
Public IP: 158.252.81.36
Public DNS:
Deployed: Mon Dec 29 19:47:55 UTC 2025
Status: Active and Running
Managed By: Terraform



5.3 Test Cache Functionality (5 marks)

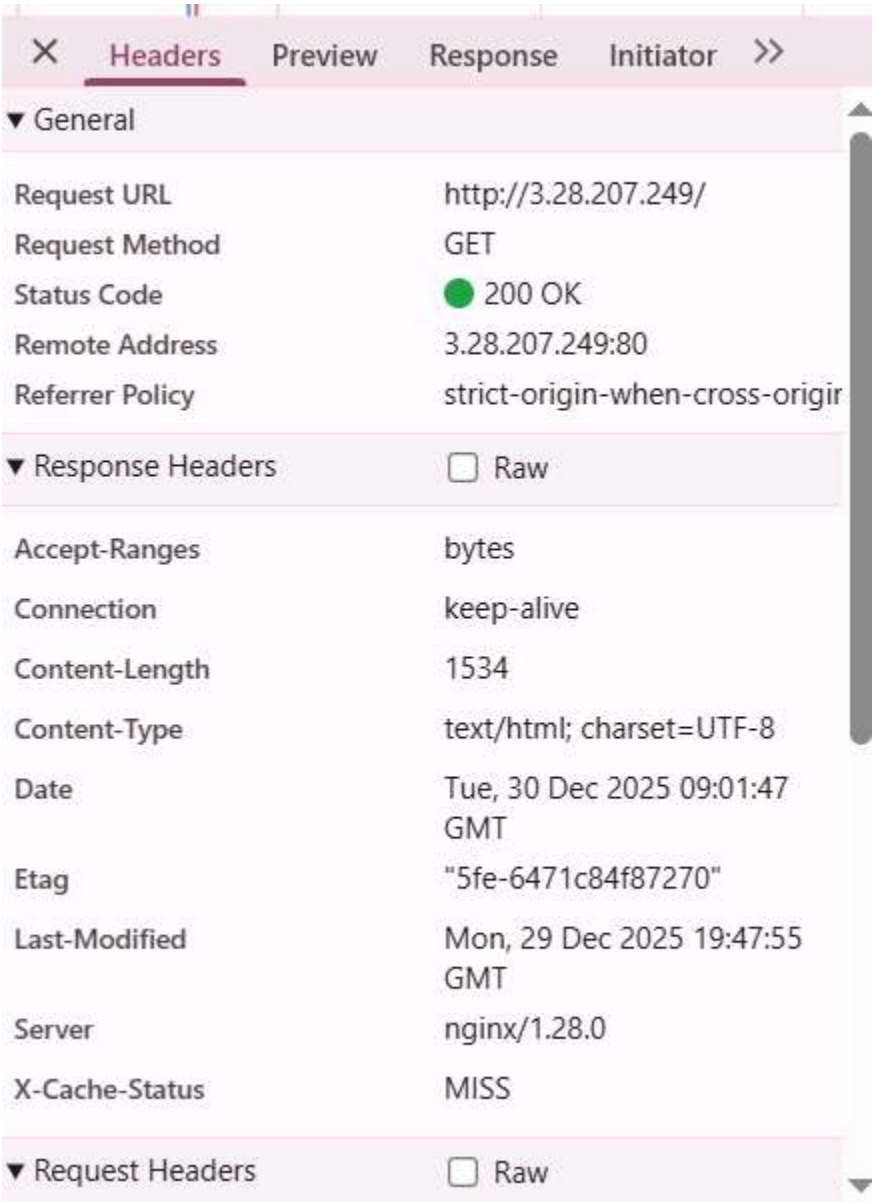
Verify that Nginx caching is working correctly.

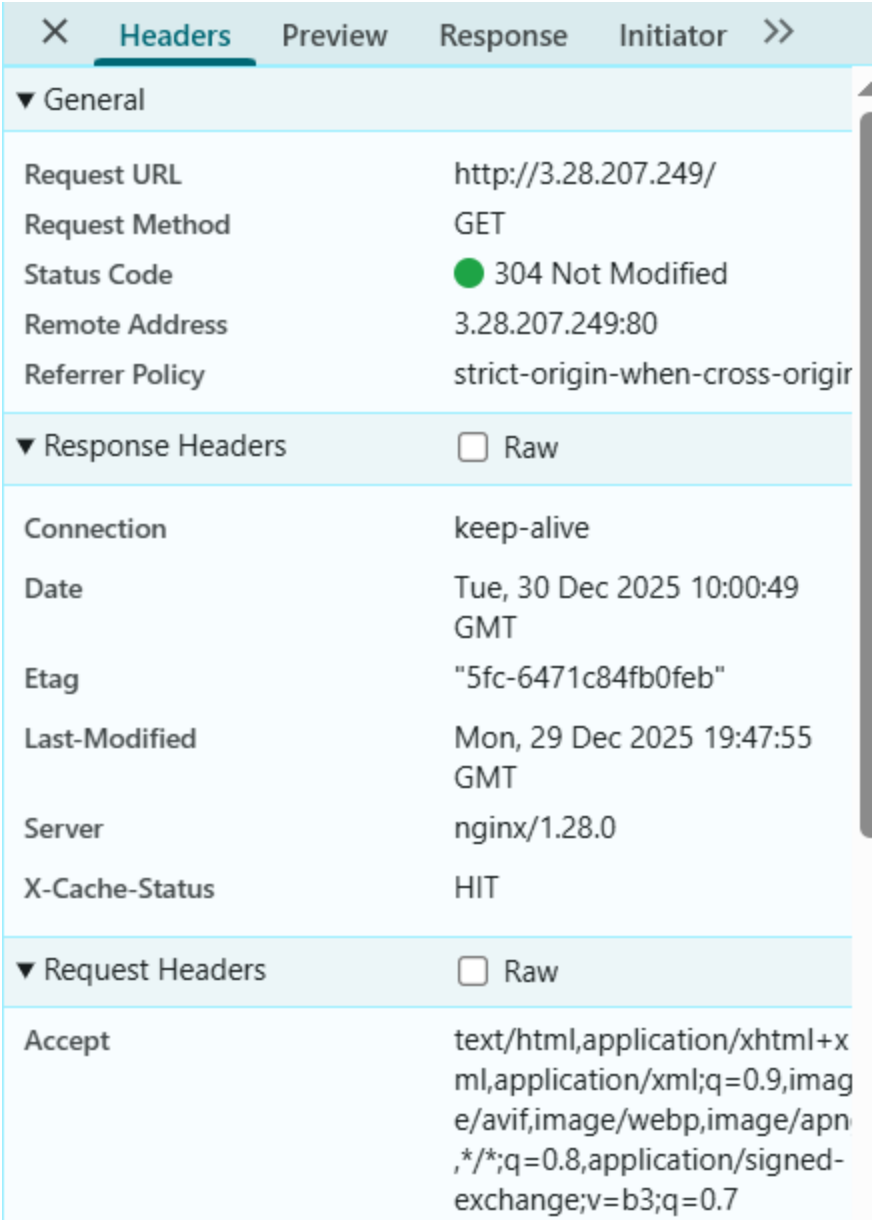
Tasks:

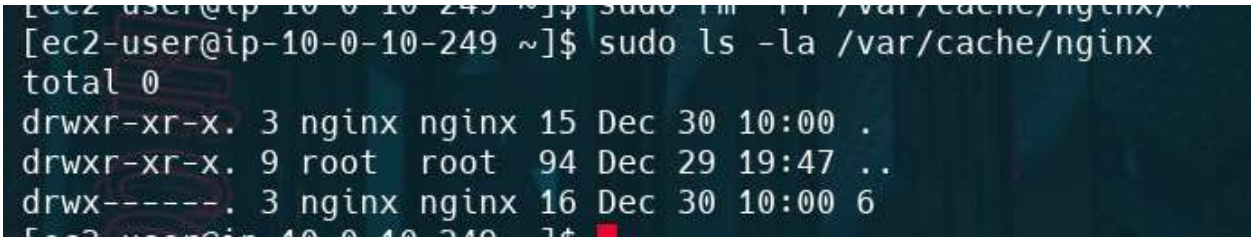
- Open browser developer tools (F12)
- Navigate to Network tab
- Clear browser cache
- Load https://<nginx-public-ip>
- Check response headers for X-Cache-Status: MISS (first request)

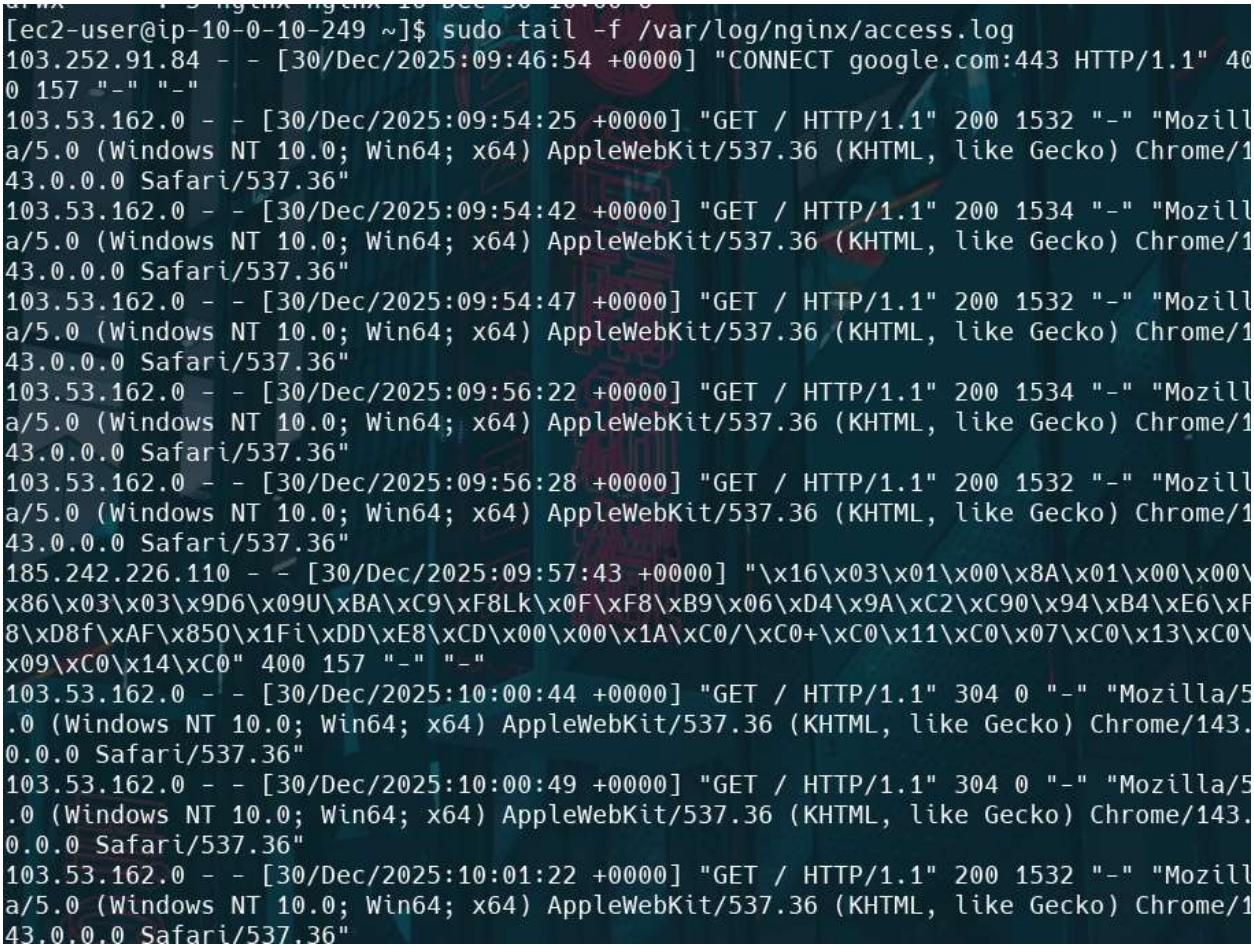
- Reload page
- Check response headers for X-Cache-Status: HIT (cached request)
- Verify cache directory on Nginx server

```
[ec2-user@ip-10-0-10-249 ~]$ sudo ls -la /var/cache/nginx/
total 0
drwx------. 2 nginx root  6 Dec 29 19:47 .
drwxr-xr-x. 9 root  root 94 Dec 29 19:47 ..
```




Headers
Preview
Response
Initiator
General
Request URL
http://3.28.207.249/
Request Method
GET
Status Code
304 Not Modified
Remote Address
3.28.207.249:80
Referrer Policy
strict-origin-when-cross-origir
Response Headers
Raw
Connection
keep-alive
Date
Tue, 30 Dec 2025 10:00:49 GMT
Etag
"5fc-6471c84fb0feb"
Last-Modified
Mon, 29 Dec 2025 19:47:55 GMT
Server
nginx/1.28.0
X-Cache-Status
HIT
Request Headers
Raw
Accept
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apn,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7


[ec2-user@ip-10-0-10-249 ~]$ sudo ls -la /var/cache/nginx
total 0
drwxr-xr-x. 3 nginx nginx 15 Dec 30 10:00 .
drwxr-xr-x. 9 root root 94 Dec 29 19:47 ..
drwx------. 3 nginx nginx 16 Dec 30 10:00 6

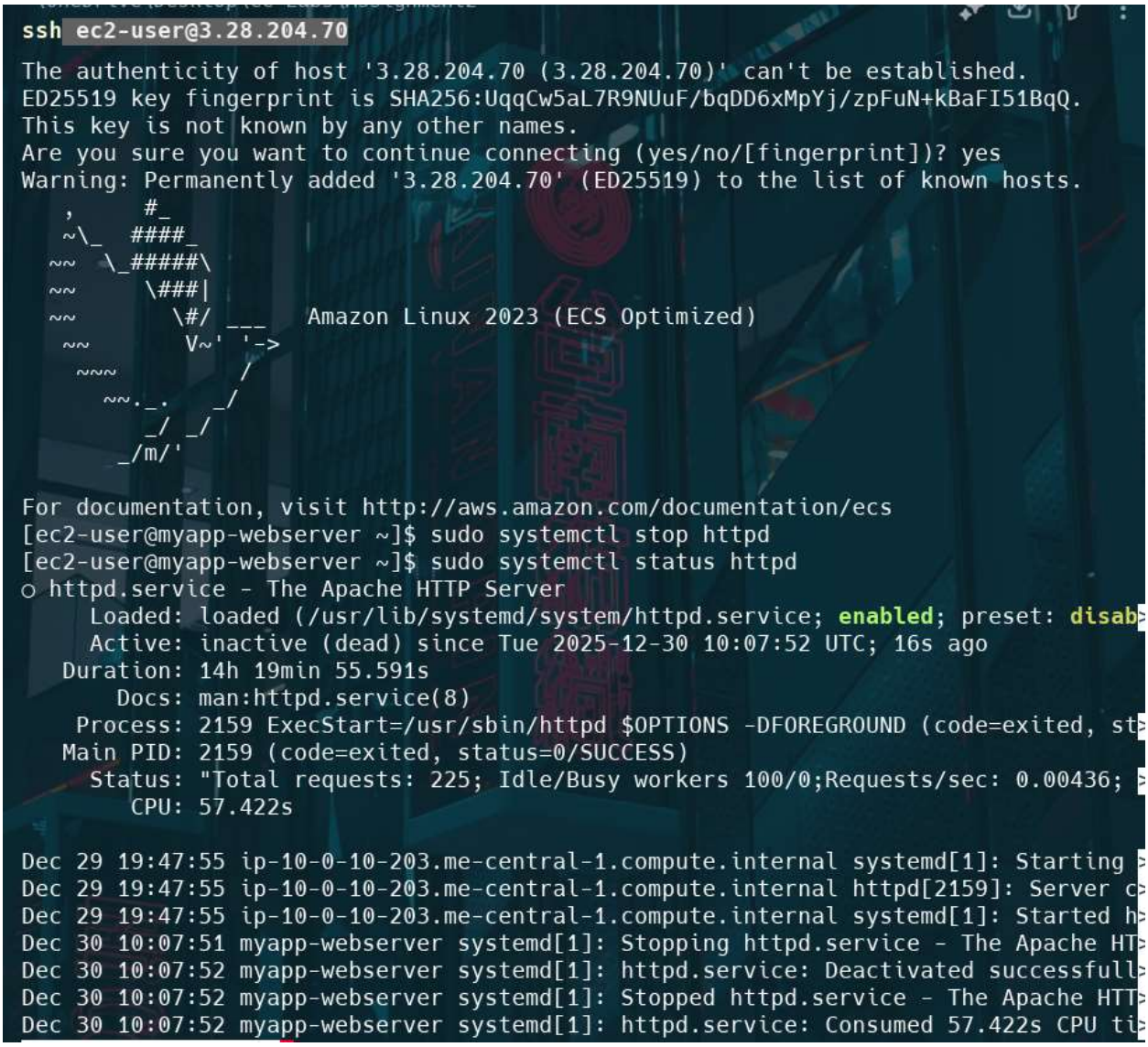


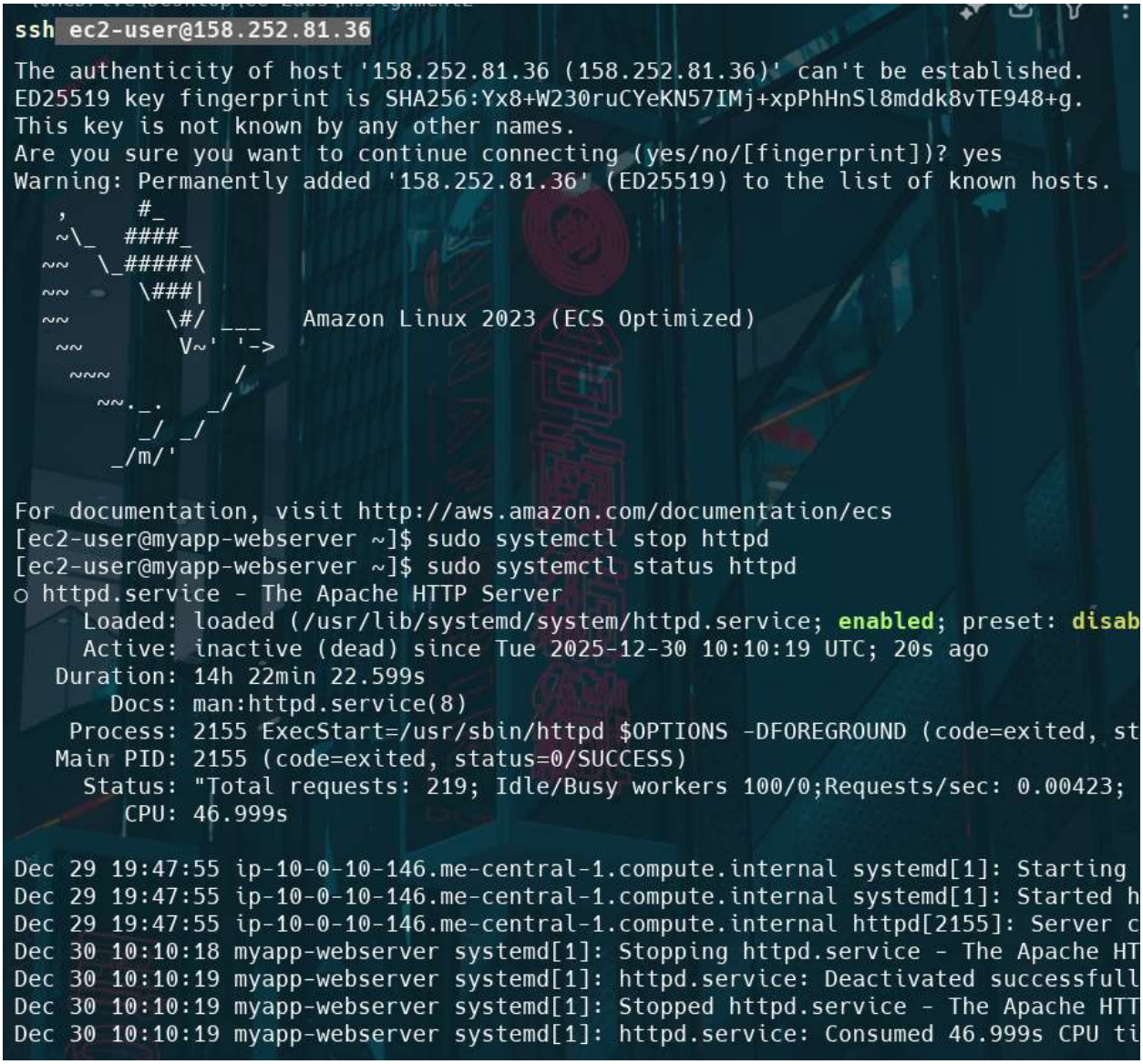
5.4 Test High Availability (Backup Server) (5 marks)

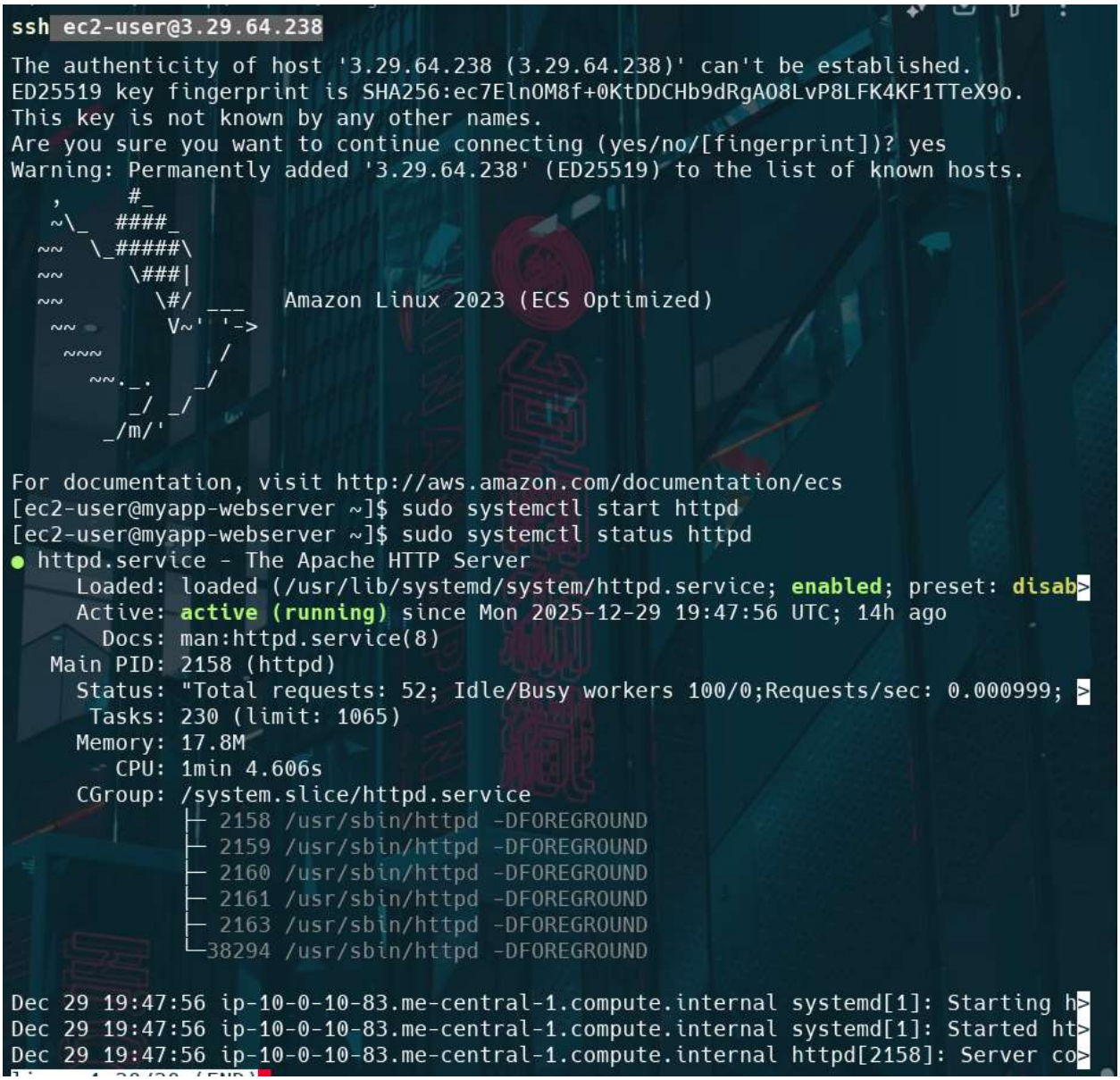
Test the backup server functionality by simulating primary server failure.

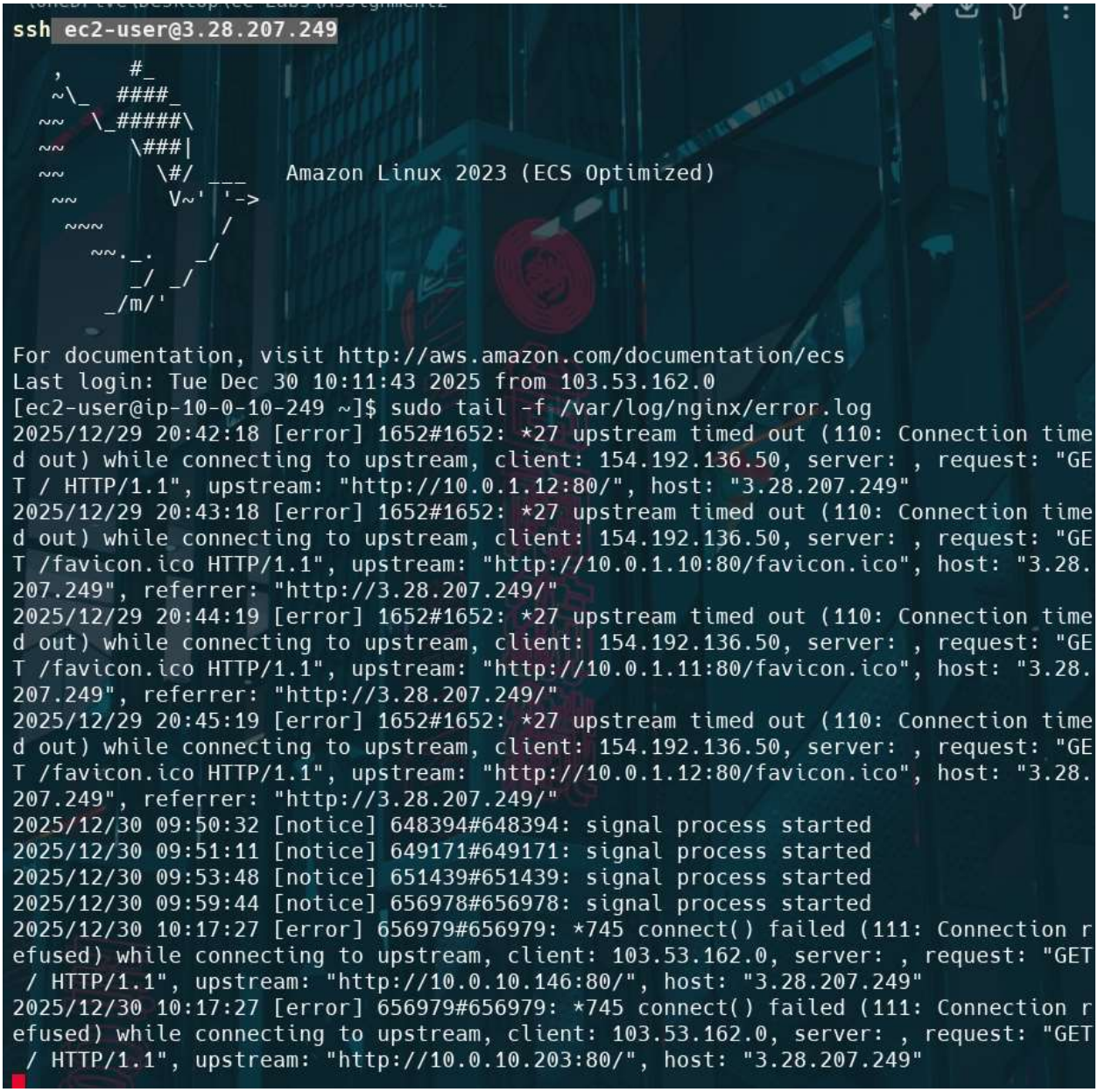
Tasks:

- SSH into web-1 and stop Apache
- Reload Nginx page - should show web-2 only
- SSH into web-2 and stop Apache
- Reload Nginx page - should now show web-3 (backup activated)
- Restart web-1 and web-2
- Verify traffic returns to web-1 and web-2


ssh ec2-user@3.28.204.70
The authenticity of host '3.28.204.70 (3.28.204.70)' can't be established.
ED25519 key fingerprint is SHA256:UqqCw5aL7R9NUuF/bqDD6xMpYj/zpFuN+kBaFI51BqQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.28.204.70' (ED25519) to the list of known hosts.
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disab
Active: inactive (dead) since Tue 2025-12-30 10:07:52 UTC; 16s ago
Duration: 14h 19min 55.591s
Docs: man:httpd.service(8)
Process: 2159 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, st
Main PID: 2159 (code=exited, status=0/SUCCESS)
Status: "Total requests: 225; Idle/Busy workers 100/0;Requests/sec: 0.00436;
CPU: 57.422s
Dec 29 19:47:55 ip-10-0-10-203.me-central-1.compute.internal systemd[1]: Starting
Dec 29 19:47:55 ip-10-0-10-203.me-central-1.compute.internal httpd[2159]: Server c
Dec 29 19:47:55 ip-10-0-10-203.me-central-1.compute.internal systemd[1]: Started h
Dec 30 10:07:51 myapp-webserver systemd[1]: Stopping httpd.service - The Apache HT
Dec 30 10:07:52 myapp-webserver systemd[1]: httpd.service: Deactivated successfull
Dec 30 10:07:52 myapp-webserver systemd[1]: Stopped httpd.service - The Apache HTT
Dec 30 10:07:52 myapp-webserver systemd[1]: httpd.service: Consumed 57.422s CPU ti


ssh ec2-user@158.252.81.36
The authenticity of host '158.252.81.36 (158.252.81.36)' can't be established.
ED25519 key fingerprint is SHA256:Yx8+W230ruCYeKN57IMj+xpPhHnSl8mddk8vTE948+g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '158.252.81.36' (ED25519) to the list of known hosts.
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
[ec2-user@myapp-webserver ~]$ sudo systemctl stop httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disab
Active: inactive (dead) since Tue 2025-12-30 10:10:19 UTC; 20s ago
Duration: 14h 22min 22.599s
Docs: man:httpd.service(8)
Process: 2155 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, st
Main PID: 2155 (code=exited, status=0/SUCCESS)
Status: "Total requests: 219; Idle/Busy workers 100/0;Requests/sec: 0.00423;
CPU: 46.999s
Dec 29 19:47:55 ip-10-0-10-146.me-central-1.compute.internal systemd[1]: Starting
Dec 29 19:47:55 ip-10-0-10-146.me-central-1.compute.internal systemd[1]: Started h
Dec 29 19:47:55 ip-10-0-10-146.me-central-1.compute.internal httpd[2155]: Server c
Dec 30 10:10:18 myapp-webserver systemd[1]: Stopping httpd.service - The Apache HT
Dec 30 10:10:19 myapp-webserver systemd[1]: httpd.service: Deactivated successfull
Dec 30 10:10:19 myapp-webserver systemd[1]: Stopped httpd.service - The Apache HTT
Dec 30 10:10:19 myapp-webserver systemd[1]: httpd.service: Consumed 46.999s CPU ti


ssh ec2-user@3.29.64.238
The authenticity of host '3.29.64.238 (3.29.64.238)' can't be established.
ED25519 key fingerprint is SHA256:ec7ElnOM8f+0KtDDCHb9dRgAO8LvP8LFK4KF1TTeX9o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.29.64.238' (ED25519) to the list of known hosts.
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
[ec2-user@myapp-webserver ~]$ sudo systemctl start httpd
[ec2-user@myapp-webserver ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disab
Active: active (running) since Mon 2025-12-29 19:47:56 UTC; 14h ago
Docs: man:httpd.service(8)
Main PID: 2158 (httpd)
Status: "Total requests: 52; Idle/Busy workers 100/0;Requests/sec: 0.000999;
Tasks: 230 (limit: 1065)
Memory: 17.8M
CPU: 1min 4.606s
CGroup: /system.slice/httpd.service
├─ 2158 /usr/sbin/httpd -DFOREGROUND
├─ 2159 /usr/sbin/httpd -DFOREGROUND
├─ 2160 /usr/sbin/httpd -DFOREGROUND
├─ 2161 /usr/sbin/httpd -DFOREGROUND
├─ 2163 /usr/sbin/httpd -DFOREGROUND
└─38294 /usr/sbin/httpd -DFOREGROUND
Dec 29 19:47:56 ip-10-0-10-83.me-central-1.compute.internal systemd[1]: Starting h
Dec 29 19:47:56 ip-10-0-10-83.me-central-1.compute.internal systemd[1]: Started ht
Dec 29 19:47:56 ip-10-0-10-83.me-central-1.compute.internal httpd[2158]: Server co


ssh ec2-user@3.28.207.249
Amazon Linux 2023 (ECS Optimized)
For documentation, visit http://aws.amazon.com/documentation/ecs
Last login: Tue Dec 30 10:11:43 2025 from 103.53.162.0
[ec2-user@ip-10-0-10-249 ~]$ sudo tail -f /var/log/nginx/error.log
2025/12/29 20:42:18 [error] 1652#1652: *27 upstream timed out (110: Connection timed out) while connecting to upstream, client: 154.192.136.50, server: , request: "GET / HTTP/1.1", upstream: "http://10.0.1.12:80/", host: "3.28.207.249"
2025/12/29 20:43:18 [error] 1652#1652: *27 upstream timed out (110: Connection timed out) while connecting to upstream, client: 154.192.136.50, server: , request: "GET /favicon.ico HTTP/1.1", upstream: "http://10.0.1.10:80/favicon.ico", host: "3.28.207.249", referrer: "http://3.28.207.249/"
2025/12/29 20:44:19 [error] 1652#1652: *27 upstream timed out (110: Connection timed out) while connecting to upstream, client: 154.192.136.50, server: , request: "GET /favicon.ico HTTP/1.1", upstream: "http://10.0.1.11:80/favicon.ico", host: "3.28.207.249", referrer: "http://3.28.207.249/"
2025/12/29 20:45:19 [error] 1652#1652: *27 upstream timed out (110: Connection timed out) while connecting to upstream, client: 154.192.136.50, server: , request: "GET /favicon.ico HTTP/1.1", upstream: "http://10.0.1.12:80/favicon.ico", host: "3.28.207.249", referrer: "http://3.28.207.249/"
2025/12/30 09:50:32 [notice] 648394#648394: signal process started
2025/12/30 09:51:11 [notice] 649171#649171: signal process started
2025/12/30 09:53:48 [notice] 651439#651439: signal process started
2025/12/30 09:59:44 [notice] 656978#656978: signal process started
2025/12/30 10:17:27 [error] 656979#656979: *745 connect() failed (111: Connection refused) while connecting to upstream, client: 103.53.162.0, server: , request: "GET / HTTP/1.1", upstream: "http://10.0.10.146:80/", host: "3.28.207.249"
2025/12/30 10:17:27 [error] 656979#656979: *745 connect() failed (111: Connection refused) while connecting to upstream, client: 103.53.162.0, server: , request: "GET / HTTP/1.1", upstream: "http://10.0.10.203:80/", host: "3.28.207.249"

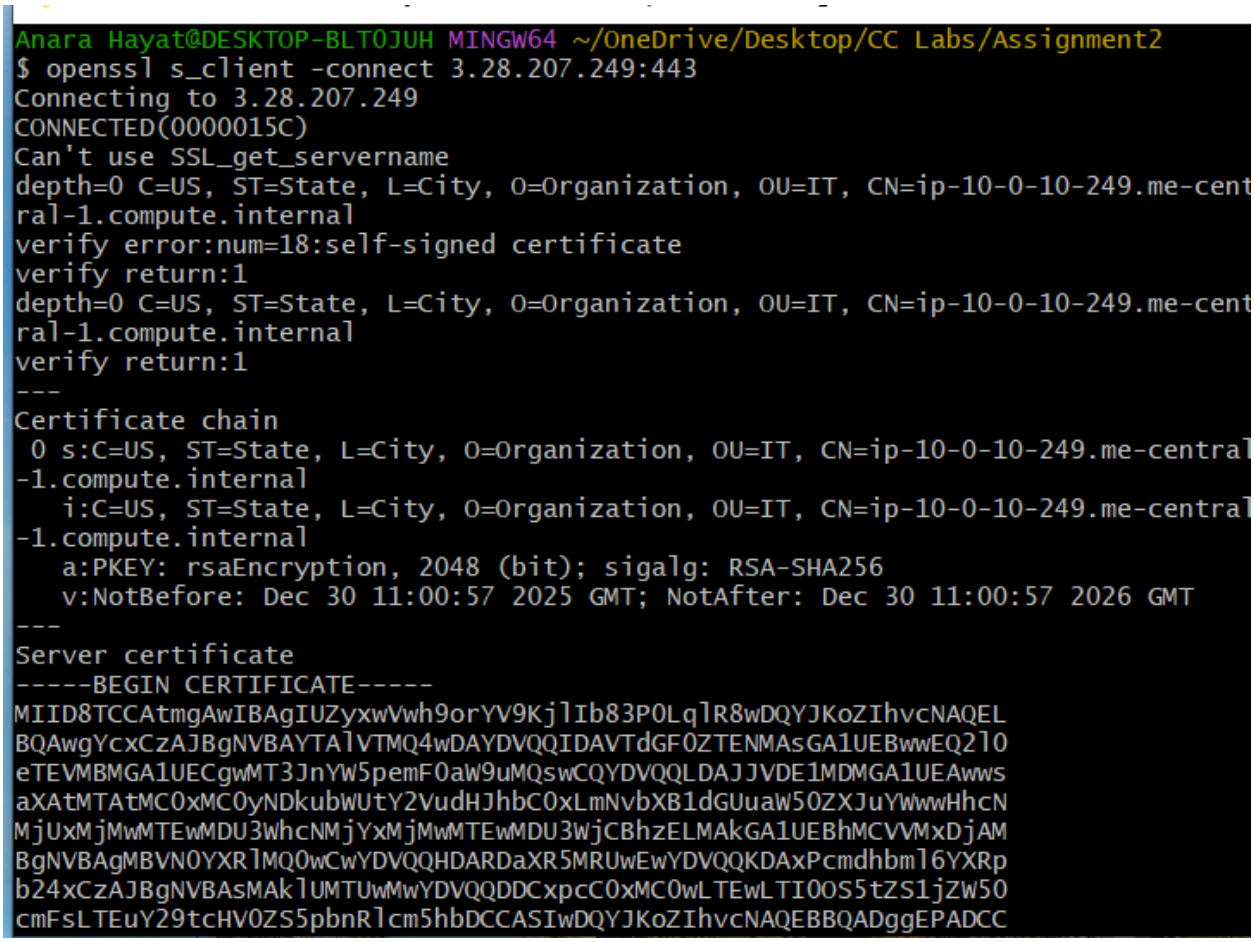
5.5 Security & Performance Analysis (5 marks)

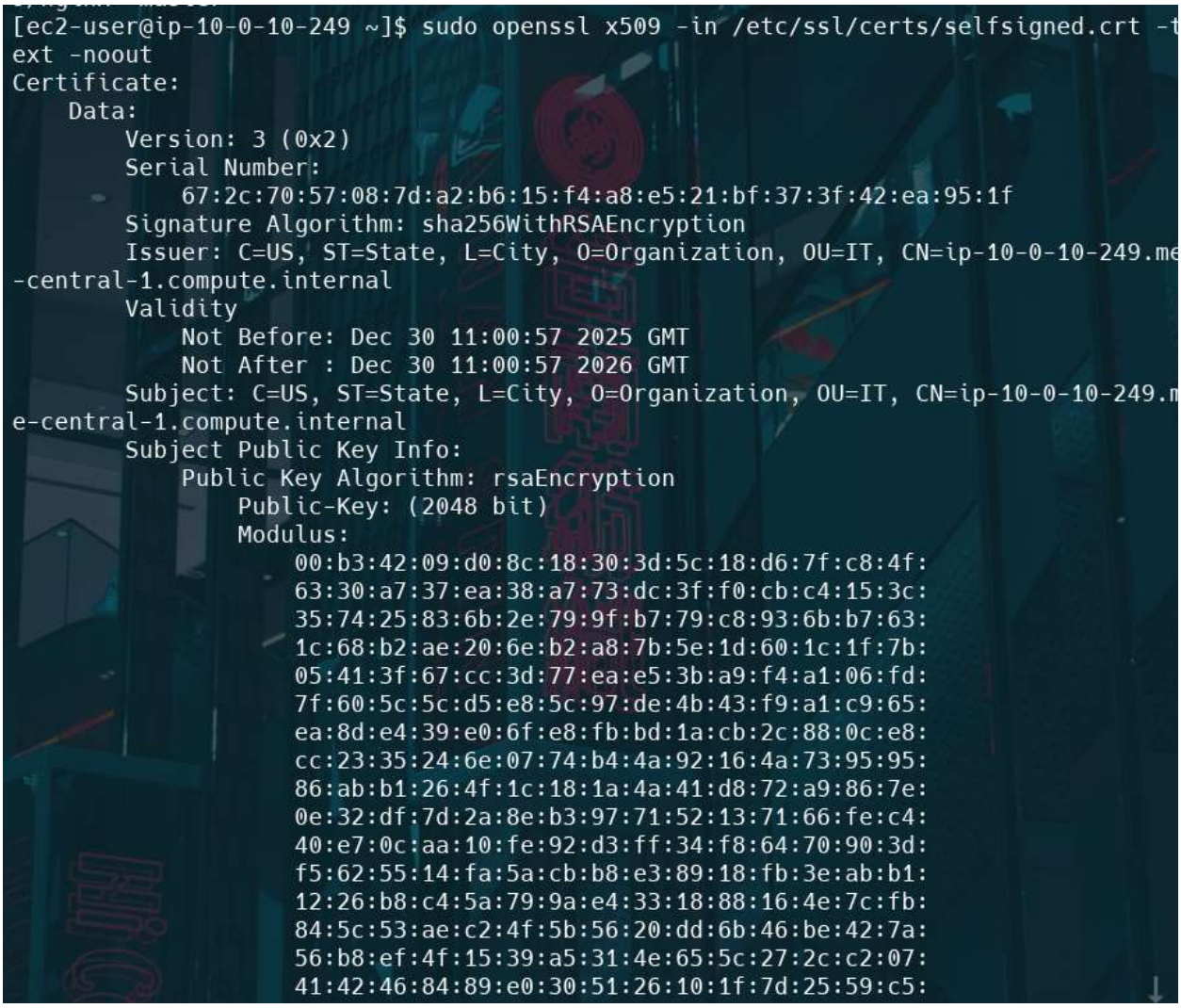
Analyze the security headers and performance of your Nginx setup.

Tasks:

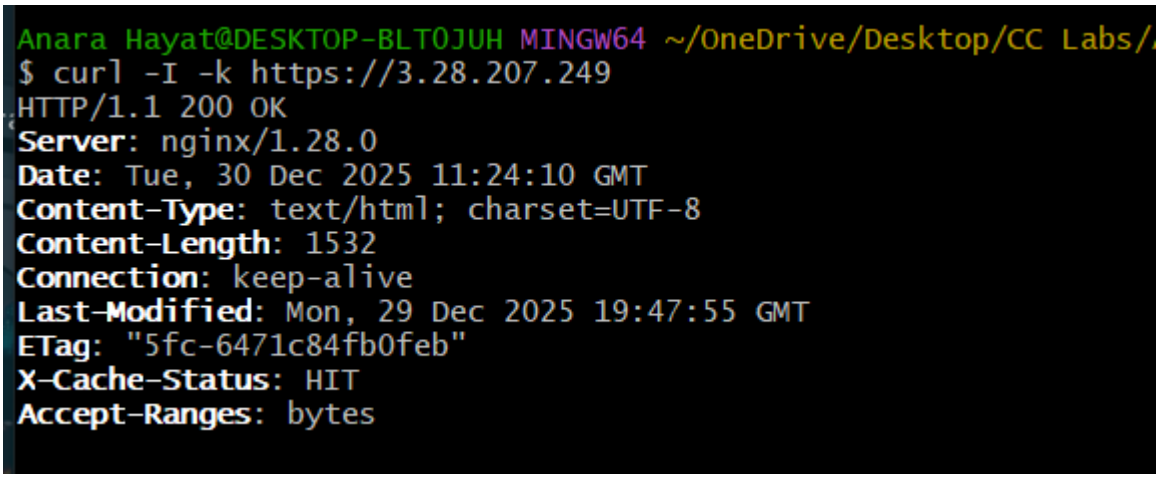
- Check SSL/TLS certificate details
- Verify security headers in response
- Test HTTP to HTTPS redirect
- Check response times
- Analyze Nginx logs

Ssl certificate





Security header



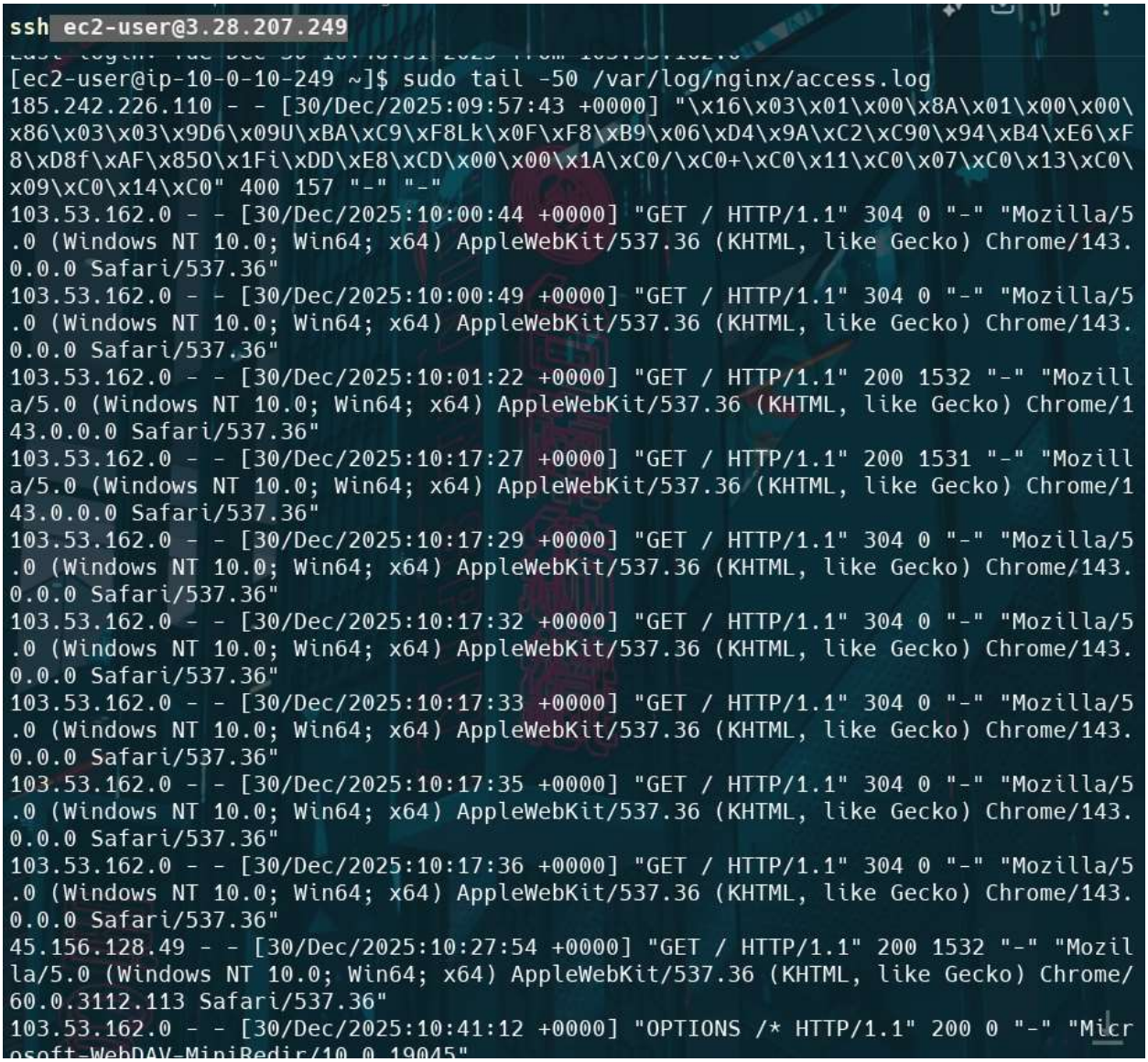
Test http redirect

```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Assignment2
$ curl -Ik https://3.28.207.249
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 11:22:01 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 1532
Connection: keep-alive
Last-Modified: Mon, 29 Dec 2025 19:47:55 GMT
ETag: "5fc-6471c84fb0feb"
X-Cache-Status: MISS
Accept-Ranges: bytes
```

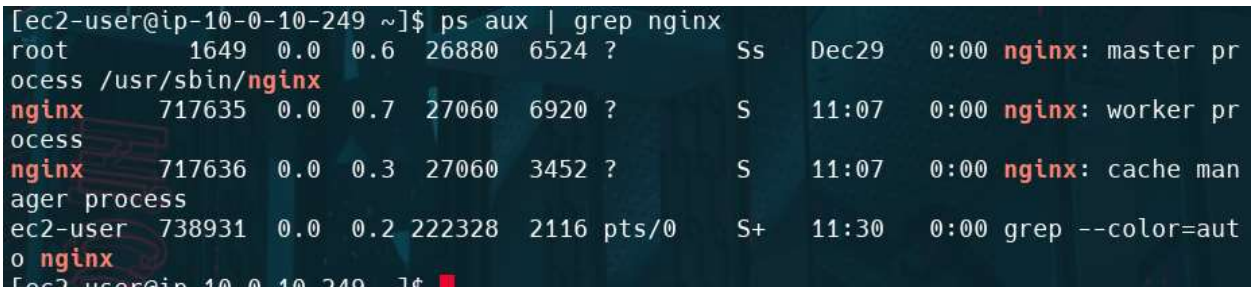
Monitor error logs

```
ssh ec2-user@3.28.207.249

[ec2-user@ip-10-0-10-249 ~]$ sudo tail -50 /var/log/nginx/error.log
2025/12/29 20:04:09 [error] 2029#2029: *23 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T / HTTP/1.1", upstream: "http://10.0.1.10:80/", host: "3.29.231.86"
2025/12/29 20:04:22 [error] 2029#2029: *17 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T /favicon.ico HTTP/1.1", upstream: "http://10.0.1.11:80/favicon.ico", host: "3.29.
231.86", referrer: "http://3.29.231.86/"
2025/12/29 20:05:22 [error] 2029#2029: *17 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T /favicon.ico HTTP/1.1", upstream: "http://10.0.1.12:80/favicon.ico", host: "3.29.
231.86", referrer: "http://3.29.231.86/"
2025/12/29 20:10:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T / HTTP/1.1", upstream: "http://10.0.1.10:80/", host: "3.29.231.86"
2025/12/29 20:11:24 [error] 2029#2029: *32 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 185.16.39.146, server: , request: "GET
 / HTTP/1.1", upstream: "http://10.0.1.10:80/", host: "3.29.231.86"
2025/12/29 20:11:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T / HTTP/1.1", upstream: "http://10.0.1.11:80/", host: "3.29.231.86"
2025/12/29 20:12:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T / HTTP/1.1", upstream: "http://10.0.1.12:80/", host: "3.29.231.86"
2025/12/29 20:13:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T /favicon.ico HTTP/1.1", upstream: "http://10.0.1.11:80/favicon.ico", host: "3.29.
231.86", referrer: "http://3.29.231.86/"
2025/12/29 20:14:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T /favicon.ico HTTP/1.1", upstream: "http://10.0.1.10:80/favicon.ico", host: "3.29.
231.86", referrer: "http://3.29.231.86/"
2025/12/29 20:15:34 [error] 2029#2029: *29 upstream timed out (110: Connection time
d out) while connecting to upstream, client: 154.192.136.50, server: , request: "GE
T /favicon.ico HTTP/1.1", upstream: "http://10.0.1.12:80/favicon.ico", host: "3.29.
231.86", referrer: "http://3.29.231.86/"
```

Monitor access logs



Check Nginx worker processes



Bonus Tasks

Bonus 1: Custom Error Pages

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

Tasks:

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

404

```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Assignment2
$ curl -I http://3.28.207.249/nonexistance
HTTP/1.1 404 Not Found
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 11:54:58 GMT
Content-Type: text/html; charset=iso-8859-1
Content-Length: 196
Connection: keep-alive
X-Cache-Status: MISS
```

502

```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs
$ curl -I -H "Cache-Control: no-cache" http://3.28.207.249/
HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:11:15 GMT
Content-Type: text/html
Content-Length: 192
Connection: keep-alive
ETag: "6953b934-c0"
```

Bonus 2: Implement Rate Limiting (3 marks)

Add rate limiting to prevent abuse.

Tasks:

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

```
events{}
http {
        proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=mycache:10m inactive=60m
;
    # Define a shared memory zone for rate limiting
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;

    upstream backend_pool {
        server 10.0.10.203;
        server 10.0.10.146;
        server 10.0.10.83 backup;
    }

    server {
        listen 80;
        server_name _;

        location / {
            # Apply rate limiting
            limit_req zone=mylimit burst=20 nodelay;

            proxy_pass http://backend_pool;
            proxy_set_header Host $host;

            # Cache settings (optional)
            proxy_cache mycache;
            proxy_cache_key "$scheme$request_method$host$request_uri";
            proxy_cache_valid 200 301 302 10m;
            proxy_cache_valid 404 1m;
            proxy_ignore_headers Cache-Control Expires Set-Cookie;

            add_header X-Cache-Status $upstream_cache_status always;
        }
    }
}
```

```
Anara Hayat@DESKTOP-BLTOJUH MINGW64 ~/OneDrive/Desktop/CC Labs/Assignment2
$ for i in {1..5}; do curl -I http://3.28.207.249/; done
HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:25:54 GMT
Content-Type: text/html
Content-Length: 157
Connection: keep-alive
X-Cache-Status: EXPIRED

HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:25:54 GMT
Content-Type: text/html
Content-Length: 157
Connection: keep-alive
X-Cache-Status: EXPIRED

HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:25:55 GMT
Content-Type: text/html
Content-Length: 157
Connection: keep-alive
X-Cache-Status: EXPIRED

HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:25:55 GMT
Content-Type: text/html
Content-Length: 157
Connection: keep-alive
X-Cache-Status: EXPIRED

HTTP/1.1 502 Bad Gateway
Server: nginx/1.28.0
Date: Tue, 30 Dec 2025 12:25:55 GMT
Content-Type: text/html
Content-Length: 157
Connection: keep-alive
X-Cache-Status: EXPIRED
```

Bonus 3: Health Check Automation

Create a shell script that monitors backend server health.

Script Requirements:

- Check each backend server every 30 seconds
- Log health status

- Alert if server is down
- Restart Apache if needed

```
  GNU nano 8.3              /home/ec2-user/backend_health_check.sh
#!/bin/bash

# List of backend servers (private IPs)
BACKENDS=("10.0.10.203" "10.0.10.146" "10.0.10.83")

# Log file
LOG_FILE="/home/ec2-user/backend_health.log"

# Interval between checks (seconds)
INTERVAL=30

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

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

        if [ "$RESPONSE" == "200" ]; then
            echo "$TIMESTAMP - $SERVER is UP" >> $LOG_FILE
        else
            echo "$TIMESTAMP - $SERVER is DOWN (HTTP $RESPONSE)" >> $LOG_FILE

            # Restart Apache (requires SSH access to backend server)
            echo "$TIMESTAMP - Restarting Apache on $SERVER" >> $LOG_FILE
            ssh -o StrictHostKeyChecking=no ec2-user@$SERVER 'sudo systemctl restart h
        fi
    done

    sleep $INTERVAL
done
```

```
^C
[ec2-user@ip-10-0-10-249 ~]$ tail -f /home/ec2-user/backend_health.log
2025-12-30 12:50:55 - 10.0.10.203 is DOWN (HTTP 000)
2025-12-30 12:50:55 - Restarting Apache on 10.0.10.203
```

Part 6: Documentation & Cleanup

6.1 README Documentation

Create a comprehensive README.md

README.md

nd > # Assignment 2 - Multi-Tier Web Infrastructure > ## 3. Deployment Ins

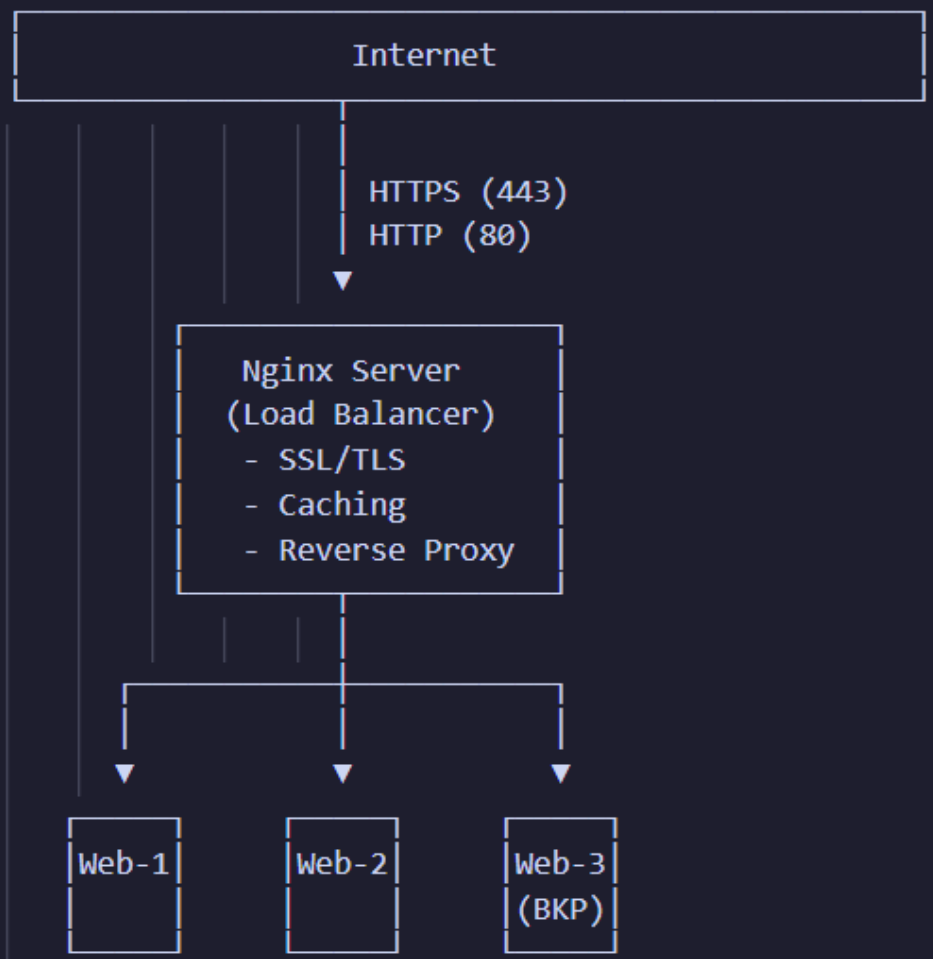
Assignment 2 - Multi-Tier Web Infrastructure

1. Project Overview

This project implements a multi-tier web infrastructure using AWS, Terraform, and Nginx. The architecture includes:

- Nginx as a reverse proxy and load balancer with caching.
- Multiple backend web servers (primary and backup).
- Security groups for network access control.
- Automated health checks and rate limiting.

1.1. Architecture Diagram



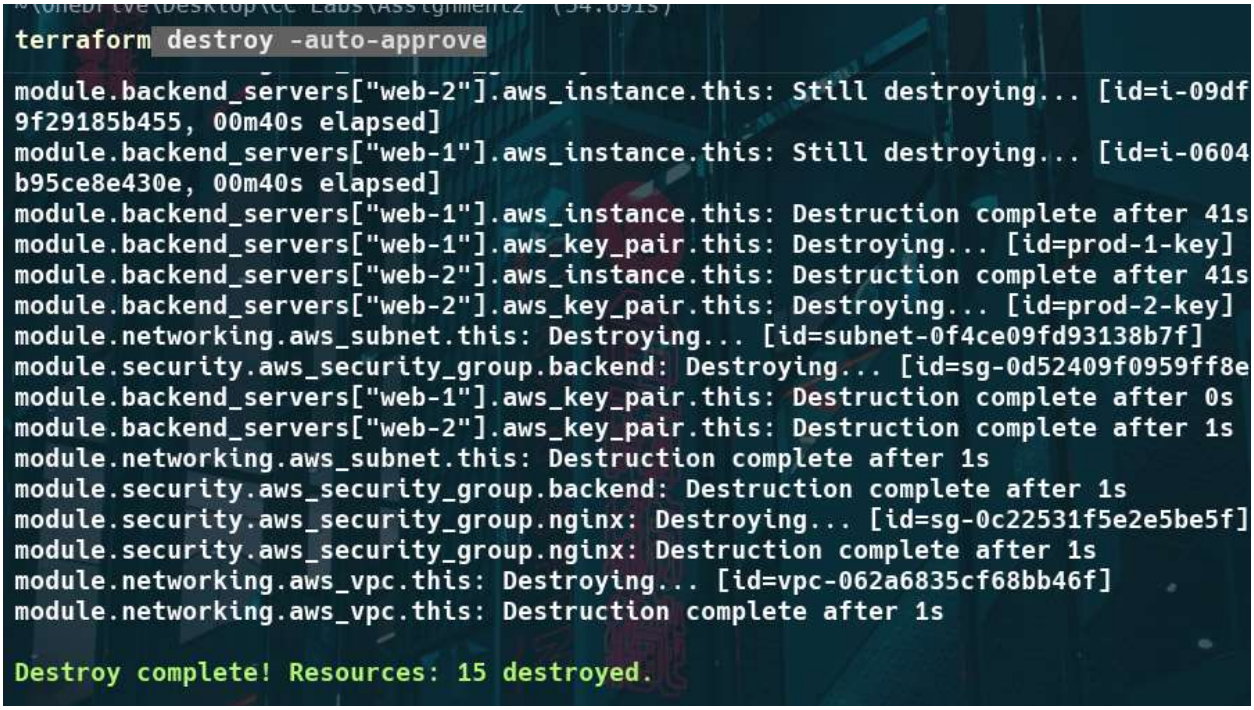
```
## 1.2. Components Description

| Component       | Description |
|-----------------|-------------|
| Nginx Server    | Reverse proxy, load balancer,
caching, SSL/TLS termination |
| Web Servers     | Backend application servers
(HTTPD) |
| Security Groups | Control inbound/outbound
access |
| Terraform       | Infrastructure as code
(provisioning EC2, SG, etc.) |
| Health Check    | Shell script monitoring backend
server health |
| Rate Limiting   | Prevents abuse of web servers
(429 responses) |

---

## 2. Prerequisites

- **Required Tools**:
  - Terraform >= 1.5
  - AWS CLI
  - SSH client
  - cURL (for testing)

- **AWS Setup**:
  - Configure credentials: `aws configure`
  - Ensure IAM user has EC2, VPC, SecurityGroup
  permissions

- **SSH Key Setup**:
  - Generate key pair: `ssh-keygen -t ed25519`
  - Store private key securely
  - Public key used in Terraform EC2 provisioning
```

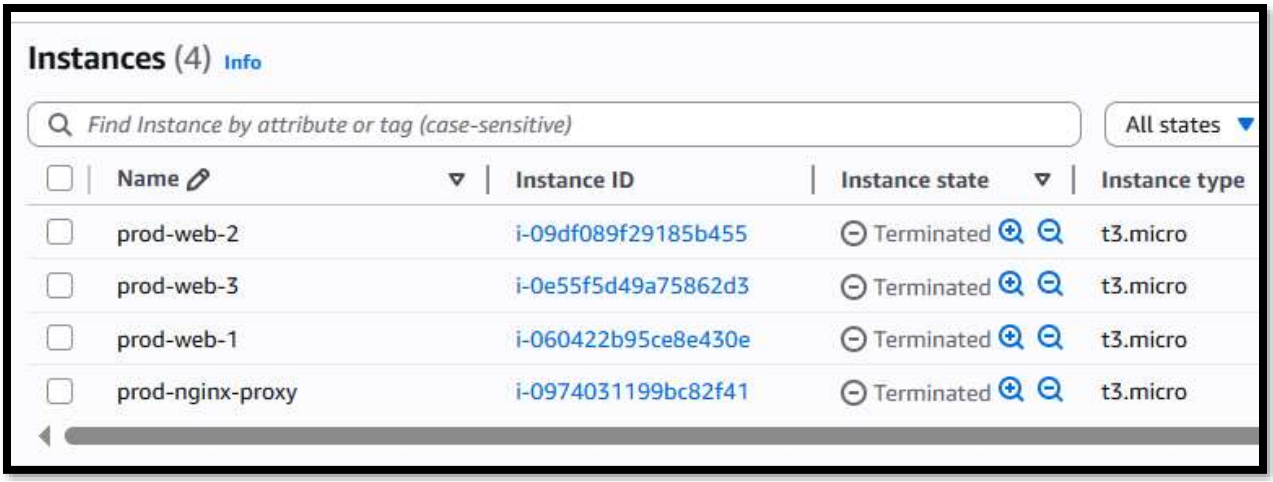
6.2 Infrastructure Cleanup (5 marks)

Properly destroy all resources and verify cleanup.

Tasks:

- Run terraform destroy
- Confirm resource deletion in AWS Console
- Verify no remaining resources
- Document the destruction process




Instances (4) Info
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
prod-web-2
i-09df089f29185b455
Terminated
t3.micro
prod-web-3
i-0e55f5d49a75862d3
Terminated
t3.micro
prod-web-1
i-060422b95ce8e430e
Terminated
t3.micro
prod-nginx-proxy
i-0974031199bc82f41
Terminated
t3.micro


~\OneDrive\Desktop\CC Labs\Assignment2 (0.035s)
cat terraform.tfstate

****