“Cloud Computing”

Assignment 2

BSE-V (A)

Submitted To: Waqas Saleem

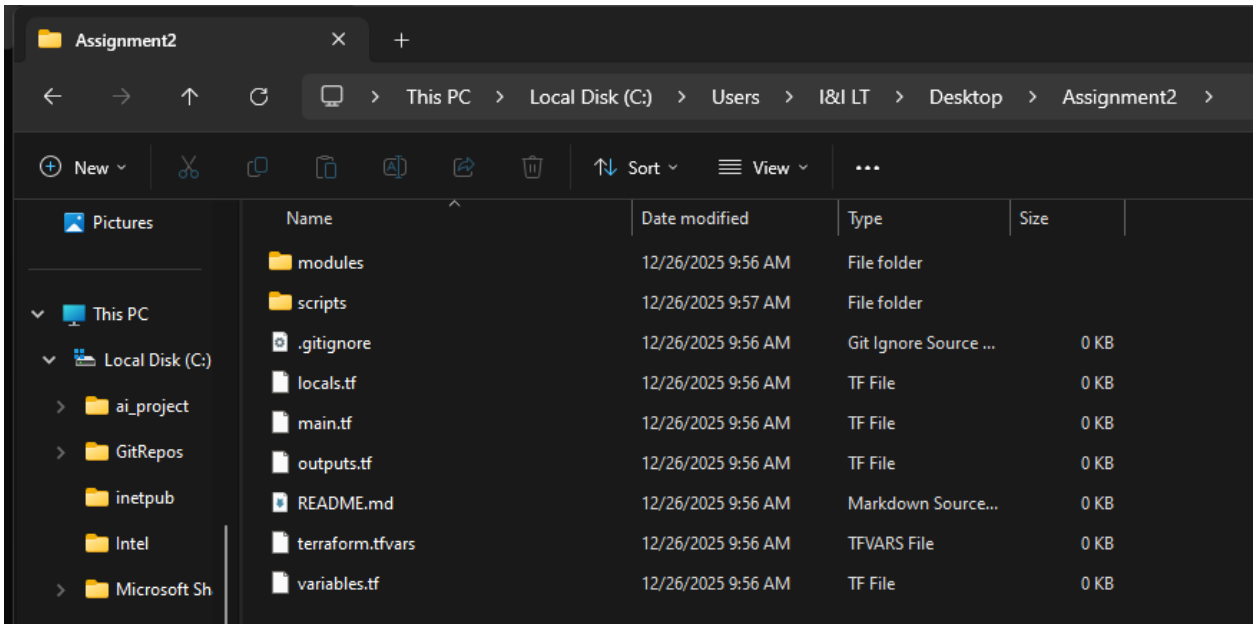
Submitted By: Kainat Shakeel

Roll no: 2023-BSE-029

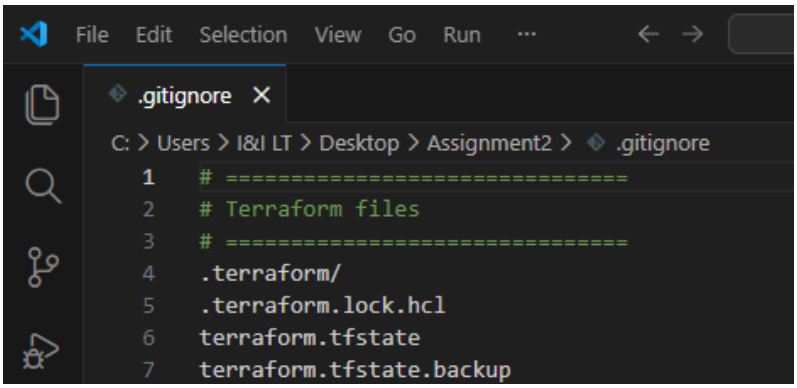
Part 1: Infrastructure Setup

1.1 Project Structure

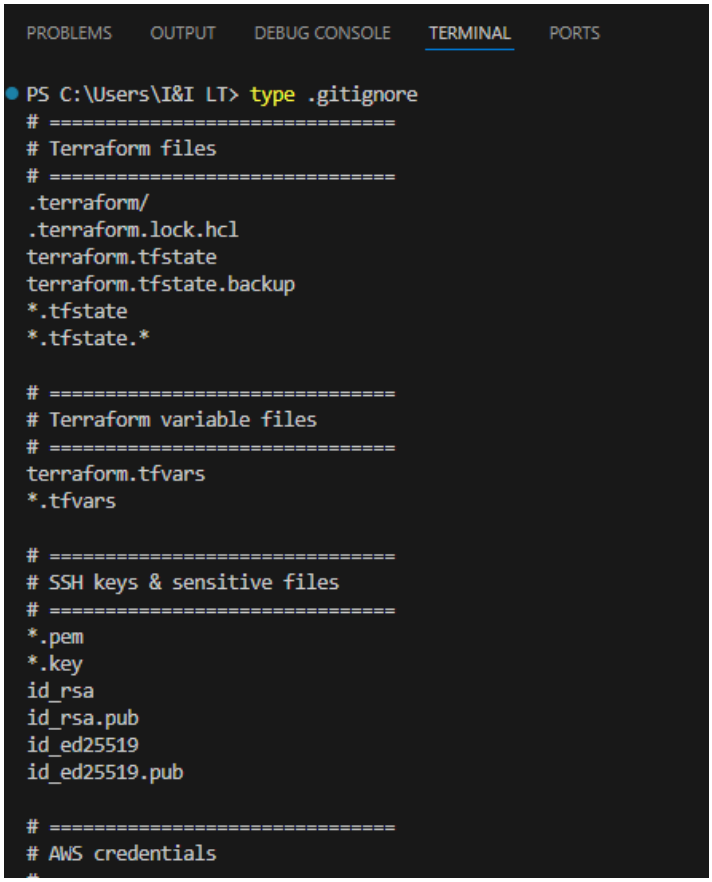
Create a well-organized Terraform project with the following structure:




```
PS C:\Users\I&I LT\Desktop\Assignment2> tree /F
Folder PATH listing
Volume serial number is BEF9-1215
C:.
    .gitignore
    locals.tf
    main.tf
    outputs.tf
    README.md
    terraform.tfvars
    variables.tf

├───modules
    ├───networking
            main.tf
            outputs.tf
            variables.tf

    ├───security
            main.tf
            outputs.tf
            variables.tf

    └───webserver
            main.tf
            outputs.tf
            variables.tf

└───scripts
        apache-setup.sh
        nginx-setup.sh
```

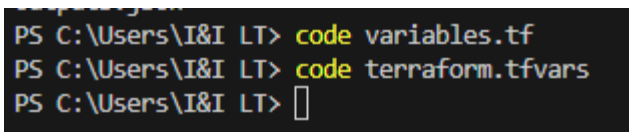


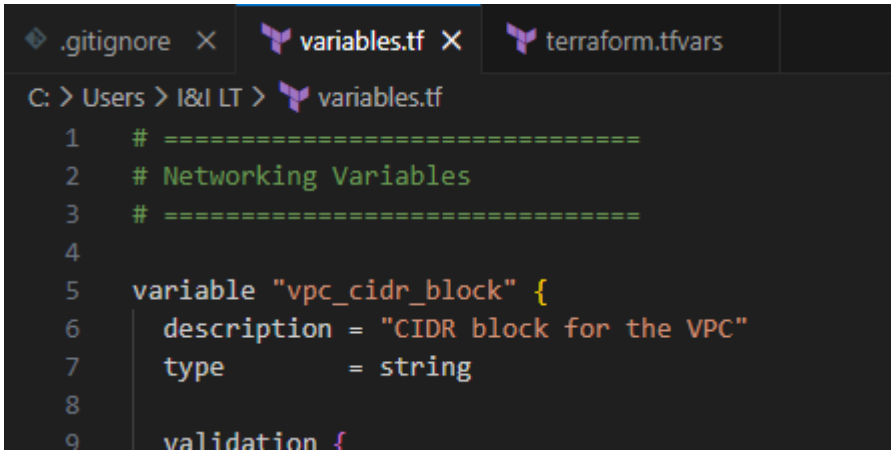
Verify .gitignore Content



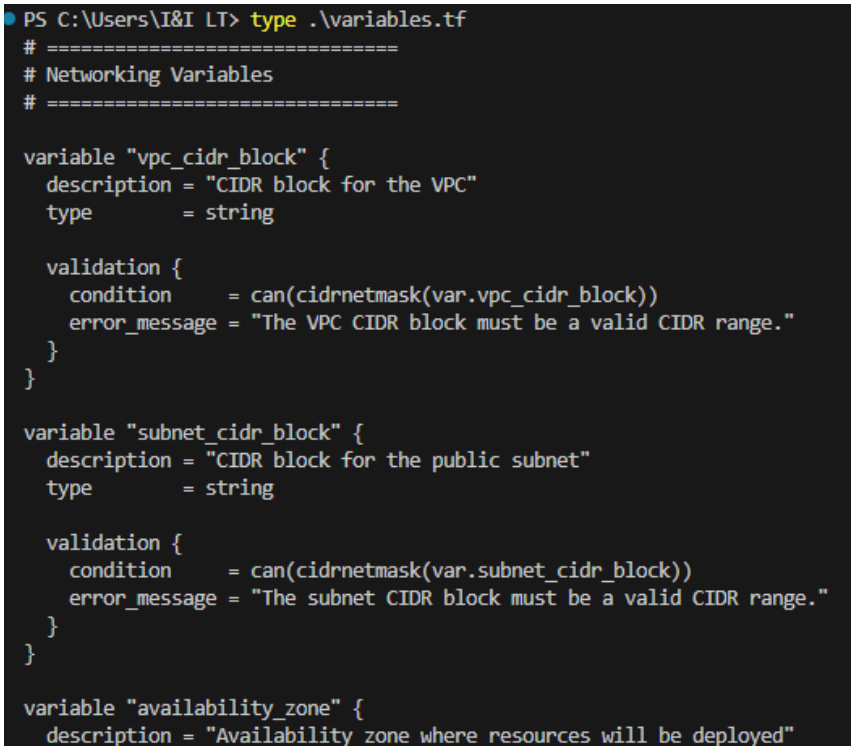
1.2 Variable Configuration

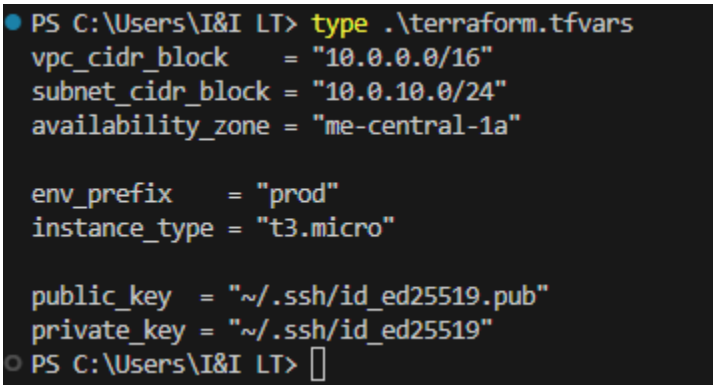
Define all required variables in variables.tf:





Verify the content for both





1.3 Networking Module

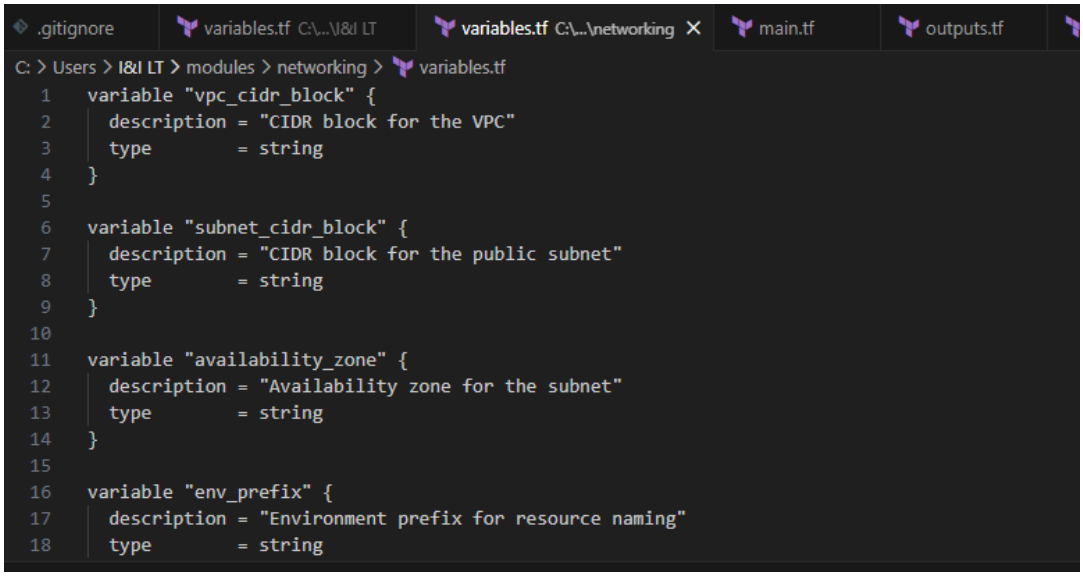
Create a networking module that provisions:

- 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

Open the files

```
PS C:\Users\I&I LT> code modules\networking\variables.tf
PS C:\Users\I&I LT> code modules\networking\main.tf
PS C:\Users\I&I LT> code modules\networking\outputs.tf
```

Write the code for each



Verify the content for each file

Main.tf

```
PS C:\Users\I&I LT> type .\modules\networking\main.tf
# ==============================
# VPC
# ==============================

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

  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

# ==============================
# Public Subnet
# ==============================

resource "aws_subnet" "this" {
  vpc_id                  = aws_vpc.this.id
  cidr_block              = var.subnet_cidr_block
  availability_zone       = var.availability_zone
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.env_prefix}-public-subnet"
  }
}
```

Output.tf

```
PS C:\Users\I&I LT> type .\modules\networking\outputs.tf
output "vpc_id" {
  description = "ID of the VPC"
  value       = aws_vpc.this.id
}

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

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

output "route_table_id" {
  description = "ID of the route table"
  value       = aws_route_table.this.id
}
PS C:\Users\I&I LT>
```

Variable.tf

```
PS C:\Users\I&I LT> type .\modules\networking\variables.tf
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
}

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

variable "availability_zone" {
  description = "Availability zone for the subnet"
  type        = string
}

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

1.4 Security Module

Create a security module that provisions:

Two Security Groups:

1. **Nginx Security Group** (for reverse proxy/load balancer):
2. **Backend Security Group** (for web servers):

open the files

```
PS C:\Users\I&I LT> code modules\security\variables.tf
PS C:\Users\I&I LT> code modules\security\main.tf
PS C:\Users\I&I LT> code modules\security\outputs.tf
```

type the content for each:

variables.tf | main.tf | outputs.tf

C: > Users > I&I LT > modules > security > outputs.tf

```
1   output "nginx_sg_id" {
2     description = "Security group ID for Nginx"
3     value       = aws_security_group.nginx.id
4   }
5
6   output "backend_sg_id" {
7     description = "Security group ID for backend servers"
8     value       = aws_security_group.backend.id
9   }
```

Verify the content:

Variable.tf

```
PS C:\Users\I&I LT> type .\modules\security\variables.tf
variable "vpc_id" {
  description = "VPC ID where security groups will be created"
  type        = string
}

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

variable "my_ip" {
  description = "Your public IP address in CIDR format (e.g. 1.2.3.4/32)"
  type        = string
}
```

Main.tf

```
PS C:\Users\I&I LT> type .\modules\security\main.tf
# ============================
# NGINX Security Group
# ============================

resource "aws_security_group" "nginx" {
  name        = "${var.env_prefix}-nginx-sg"
  description = "Security group for Nginx reverse proxy"
  vpc_id      = var.vpc_id

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

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

Output.tf

```
PS C:\Users\I&I LT> type .\modules\security\outputs.tf
output "nginx_sg_id" {
  description = "Security group ID for Nginx"
  value       = aws_security_group.nginx.id
}

output "backend_sg_id" {
  description = "Security group ID for backend servers"
  value       = aws_security_group.backend.id
}
```

1.5 Locals Configuration

Create locals.tf with:

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

Open file

```
PS C:\Users\I&I LT> code locals.tf
PS C:\Users\I&I LT>
```

Type the content

locals.tf

C: > Users > I&I LT > locals.tf

```
# ================================
# Dynamic Public IP Detection
# ================================

data "http" "my_ip" {
  url = "https://icanhazip.com"
}

# ================================
# Local Values
# ================================

locals {
  # My 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"
```

Now verify the content

```
PS C:\Users\I&I LT> type .\locals.tf
# ==============================
# Dynamic Public IP Detection
# ==============================

data "http" "my_ip" {
  url = "https://icanhazip.com"
}

# ==============================
# Local Values
# ==============================

locals {
  # My 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"
  }

  # Resource naming
  name_prefix = "${var.env_prefix}-assignment2"

  # Backend server configurations
  backend_servers = [
    {
      name        = "web-1"
      suffix      = "1"
      script_path = "./scripts/apache-setup.sh"
    },
```

****

Part 2: Webserver Module

2.1 – Webserver Module

Open the file

```
PS C:\Users\I&I LT> code modules\webserver\variables.tf
PS C:\Users\I&I LT> code modules\webserver\main.tf
PS C:\Users\I&I LT> code modules\webserver\outputs.tf
```

Type the content for each

variables.tf | main.tf | outputs.tf

C: > Users > I&I LT > modules > webserver > outputs.tf

```
1  output "instance_id" {
2    description = "ID of the EC2 instance"
3    value       = aws_instance.this.id
4  }
5
6  output "public_ip" {
7    description = "Public IP address of the instance"
8    value       = aws_instance.this.public_ip
```

Verify the content

Variables.tf

```
PS C:\Users\I&I LT> type .\modules\webserver\variables.tf
variable "env_prefix" {
  description = "Environment prefix for resource naming"
  type        = string
}

variable "instance_name" {
  description = "Base name of the EC2 instance"
  type        = string
}

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

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

variable "availability_zone" {
  description = "Availability zone for the instance"
  type        = string
}
```

Main.tf

```
PS C:\Users\I&I LT> type .\modules\webserver\main.tf
# ============================
# Latest Amazon Linux 2023 AMI
# ============================

data "aws_ami" "amazon_linux_2023" {
  most_recent = true
  owners      = ["amazon"]

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

# ============================
# Key Pair (Unique per Instance)
# ============================

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

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

Output.tf

```
PS C:\Users\I&I LT> type .\modules\webserver\outputs.tf
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.this.id
}

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

output "private_ip" {
  description = "Private IP address of the instance"
  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)

Open the file

```
PS C:\Users\I&I LT> code main.tf
PS C:\Users\I&I LT>
```

Type the content

main.tf

C: > Users > I&I LT > main.tf

```
# ================================
# NGINX SERVER (Reverse Proxy)
# ================================

module "nginx_server" {
  source            = "./modules/webserver"

  env_prefix        = var.env_prefix
  instance_name     = "nginx-proxy"
  instance_type     = var.instance_type
  availability_zone = var.availability_zone

  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.nginx_sg_id

  public_key        = var.public_key
  script_path       = "./scripts/nginx-setup.sh"
  instance_suffix   = "nginx"

```

Verify the content

```
PS C:\Users\I&I LT> type .\main.tf
# ==============================
# NGINX SERVER (Reverse Proxy)
# ==============================

module "nginx_server" {
  source            = "./modules/webserver"

  env_prefix        = var.env_prefix
  instance_name     = "nginx-proxy"
  instance_type     = var.instance_type
  availability_zone = var.availability_zone

  vpc_id            = module.networking.vpc_id
  subnet_id         = module.networking.subnet_id
  security_group_id = module.security.nginx_sg_id

  public_key        = var.public_key
  script_path       = "./scripts/nginx-setup.sh"
  instance_suffix   = "nginx"

  common_tags       = local.common_tags
}
```

****

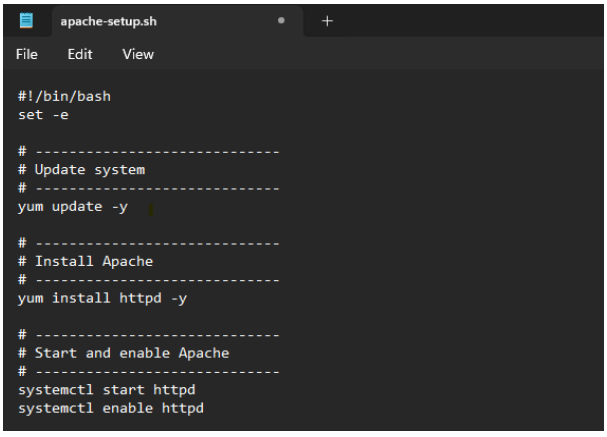
Part 3: Server Configuration Scripts

3.1 Apache Backend Server Script

Open the file

```
PS C:\Users\I&I LT> cd "C:\Users\I&I LT\Desktop\Assignment2\scripts"
PS C:\Users\I&I LT\Desktop\Assignment2\scripts> notepad apache-setup.sh
```

Type the content



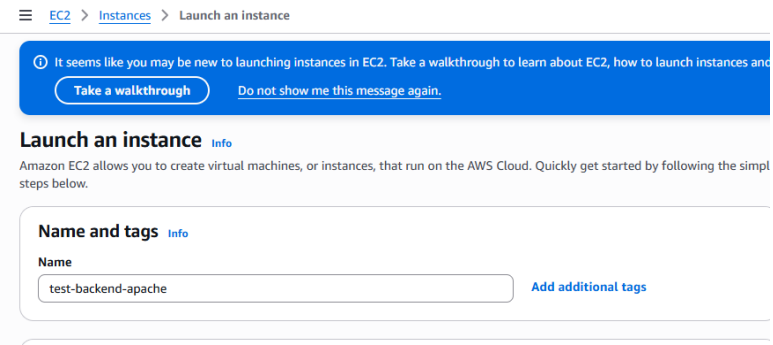
Verify the content

```
PS C:\Users\I&I LT\Desktop\Assignment2\scripts> Get-Content 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/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

# ------------------------------
# Fetch instance metadata
# ------------------------------
PRIVATE_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
```

Test Script Independently

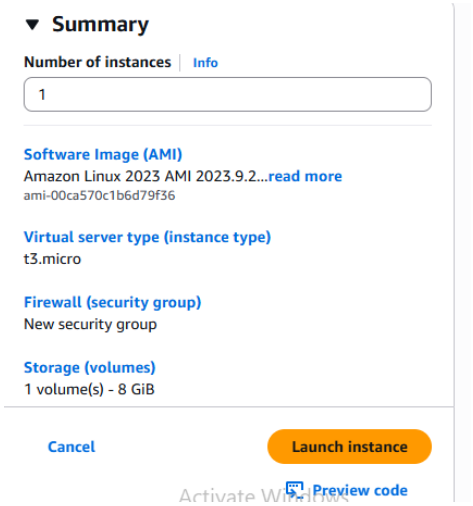
Launch Test EC2 Instance

- Open **AWS Management Console** and navigate to **EC2 → Instances**.
- Click **Launch Instance**.
- **Name and Tags**:
    - Name: test-backend-apache

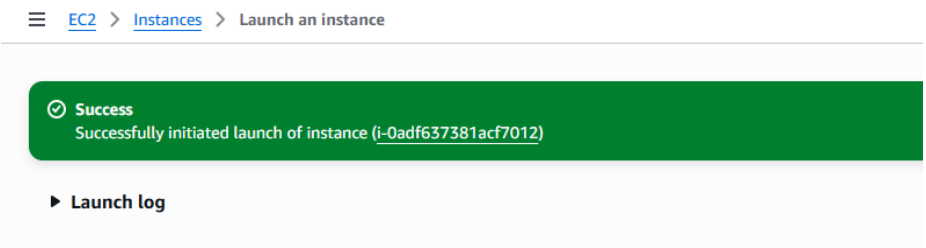


- **Choose Amazon Machine Image (AMI)**:
    - Select **Amazon Linux 2023 AMI (x86_64)**
- **Instance Type**:
    - Choose **t3.micro** (Free tier eligible)
- **Key Pair (login)**:
    - Select existing key pair or create a new one
    - Download .pem file if creating new key
- **Network Settings**:
    - **VPC**: Select your backend VPC

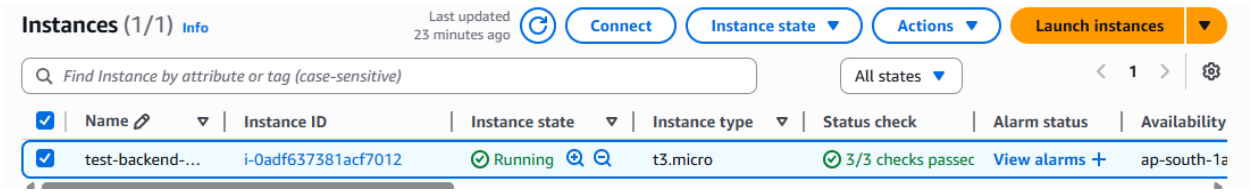
    - **Subnet**: Choose backend subnet (e.g., subnet-0e6477e4da4095d37)
    - **Auto-assign Public IP**: Enable
- **Firewall (Security Group)**:
    - Allow **SSH (Port 22)** from your IP only
    - Allow **HTTP (Port 80)** from anywhere
    - Allow **HTTPS (Port 443)** from anywhere (optional)
- **Storage**:
    - Default 8 GiB (gp3) is sufficient



- Review all settings and click **Launch Instance**



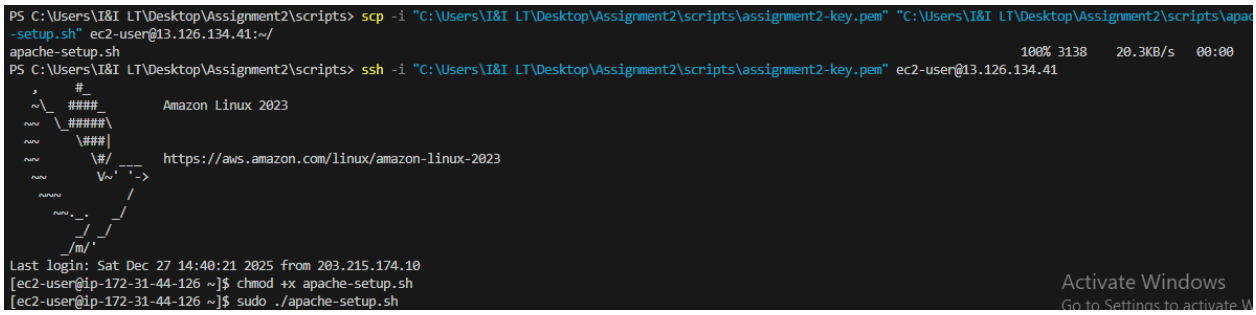
- Verify the instance is in **running** state in the EC2 dashboard



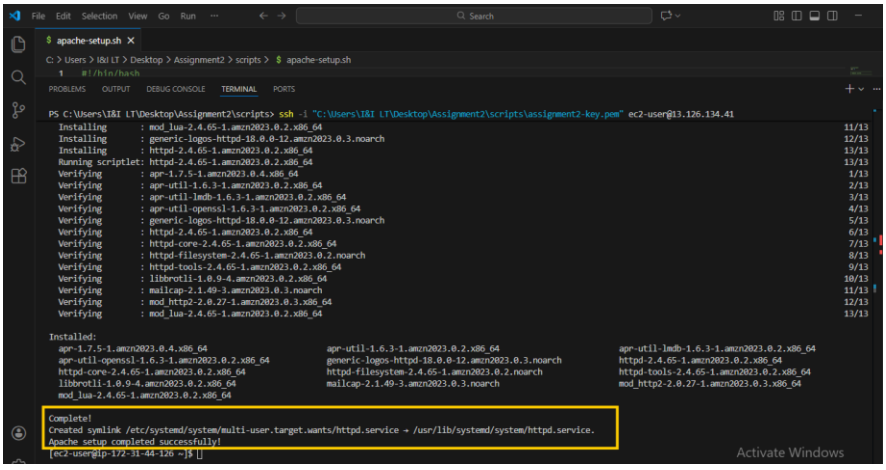
Copy Script to EC2 Instance

- Navigate to your script folder: cd "C:\Users\I&I LT\Desktop\Assignment2\scripts"Copy the script to your EC2 backend instance using scp:
- scp -i ".\assignment2-key.pem" ".\apache-setup.sh" ec2-user@<PUBLIC_IP>:~/

- Replace <PUBLIC_IP> with the public IP of your EC2 instance (e.g., 13.126.134.41).

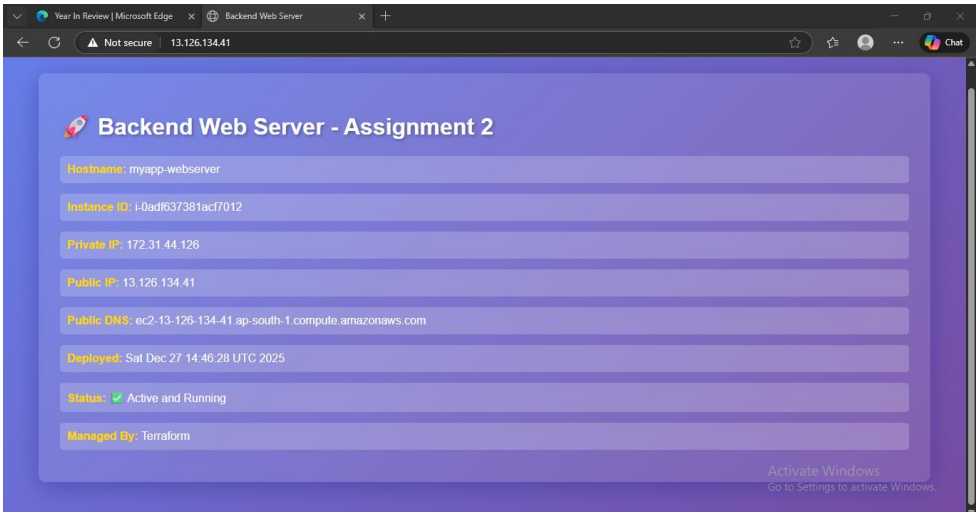


- Connect to EC2 Instance via SSH
- Make Script Executable: chmod +x apache-setup.sh
- Run the Apache Setup Script: sudo ./apache-setup.sh



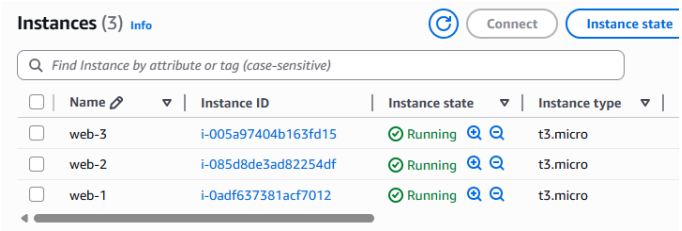
Test the Webpage

1. Open a **browser**.
2. Enter the **public IP** of your EC2 instance: [Backend Web Server]



3.2 Nginx Server Setup Script

- Connect to EC2 via SSH using the .pem key.



- Stop conflicting services (e.g., Apache) on port 80:
- Create SSL directories and generate a self-signed certificate:
- Create cache directory:

```
[ec2-user@myapp-webserver ~]$ sudo systemctl start nginx
sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset: disabled)
     Active: active (running) since Sun 2025-12-28 05:14:30 UTC; 55ms ago
    Process: 64383 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 64384 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 64385 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 64386 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.2M
        CPU: 55ms
     CGroup: /system.slice/nginx.service
             ├─64386 "nginx: master process /usr/sbin/nginx"
             ├─64387 "nginx: worker process"
             └─64388 "nginx: worker process"

Dec 28 05:14:30 myapp-webserver systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Dec 28 05:14:30 myapp-webserver nginx[64384]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Dec 28 05:14:30 myapp-webserver nginx[64384]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Dec 28 05:14:30 myapp-webserver systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.
[ec2-user@myapp-webserver ~]$
```

Test Nginx configuration:

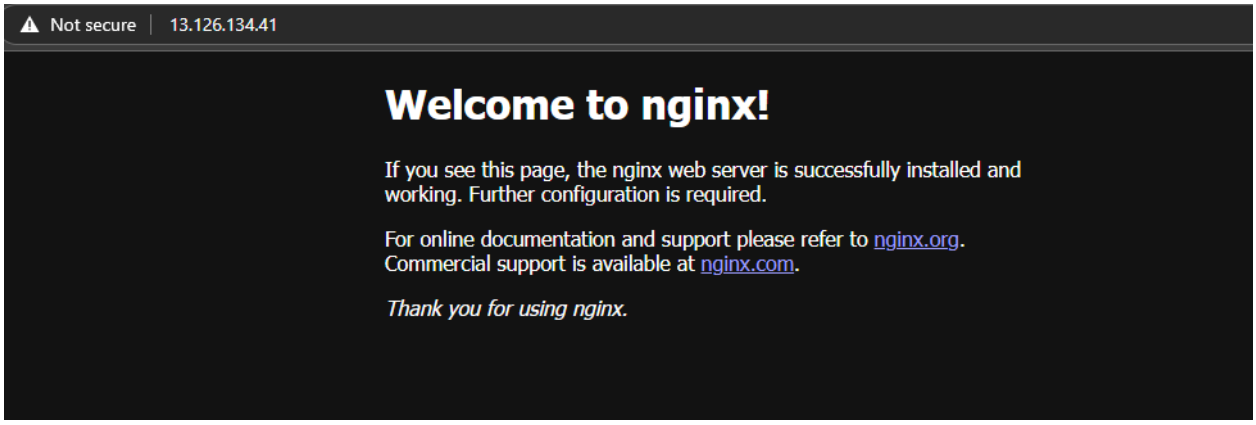
```
[ec2-user@myapp-webserver ~]$ sudo nginx -t
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" directive instead in /etc/nginx/nginx.conf:61
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" directive instead in /etc/nginx/nginx.conf:62
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@myapp-webserver ~]$
```

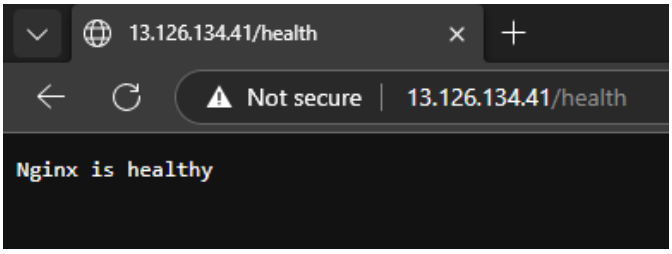
Verify endpoints:

- From EC2:

```
[ec2-user@myapp-webserver ~]$ curl -k https://localhost/
curl http://localhost/health
Nginx is healthy
Nginx is healthy
[ec2-user@myapp-webserver ~]$
```

- From browser:



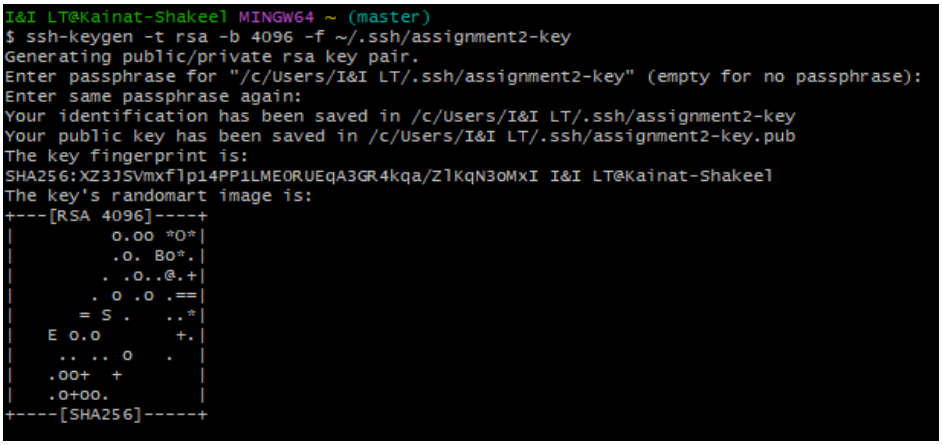


****

Part 4: Infrastructure Deployment

4.1 Initial Deployment

Generate a new SSH key



Initialize Terraform

```
I&I LT@Kainat-Shakeel MINGW64 ~ (master)
$ cd "C:\Users\I&I LT\Desktop\Assignment2"
terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.27.0

Terraform has been successfully initialized!

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

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

Validate Terraform configuration

```
I&I LT@Kainat-Shakeel MINGW64 ~/Desktop/Assignment2 (master)
$ terraform validate
Success! The configuration is valid.
```

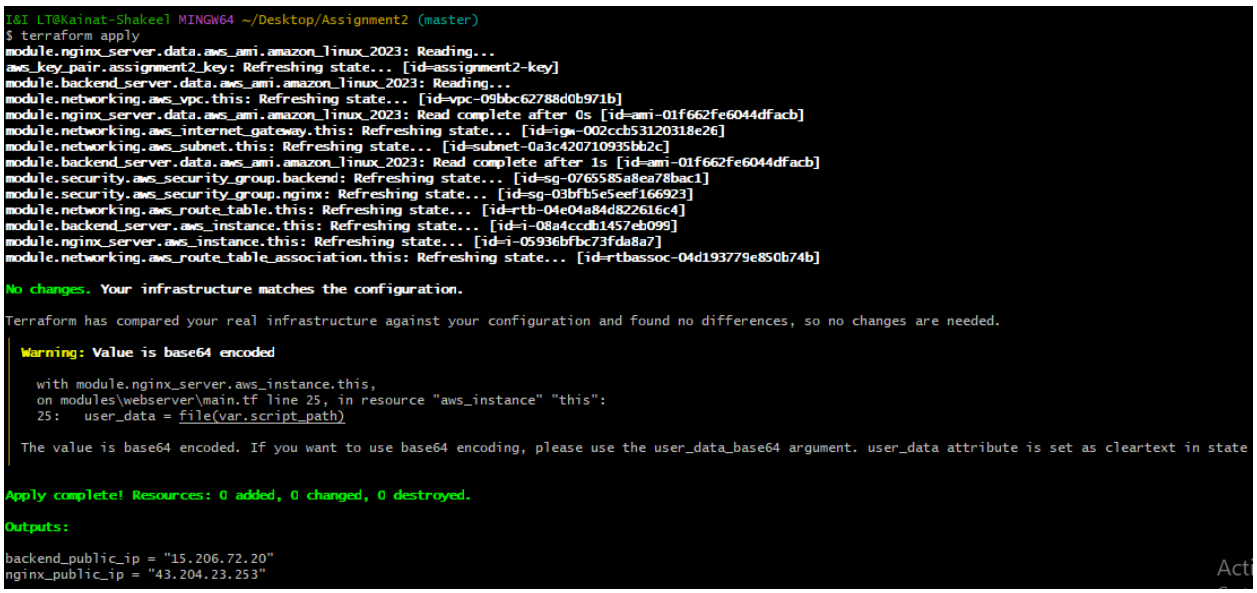
Plan the deployment

```
I&I LT@Kainat-Shakeel MINGW64 ~/Desktop/Assignment2 (master)
$ terraform plan
aws_key_pair.assignment2_key: Refreshing state... [id=assignment2-key]
aws_security_group.web_sg: Refreshing state... [id=sg-016dc271b9a9179a1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.nginx will be created
  + resource "aws_instance" "nginx" {
      + ami                                  = "ami-0de53d8956e8dcf80"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t3.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "assignment2-key"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_group_id                   = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + region                               = "ap-south-1"
      + secondary_private_ips                = (known after apply)
      + security_groups                      = [
```

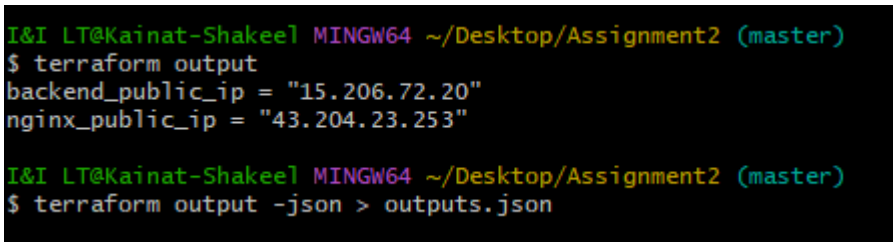
Apply the deployment

4.2 Output Configuration

Apply Terraform

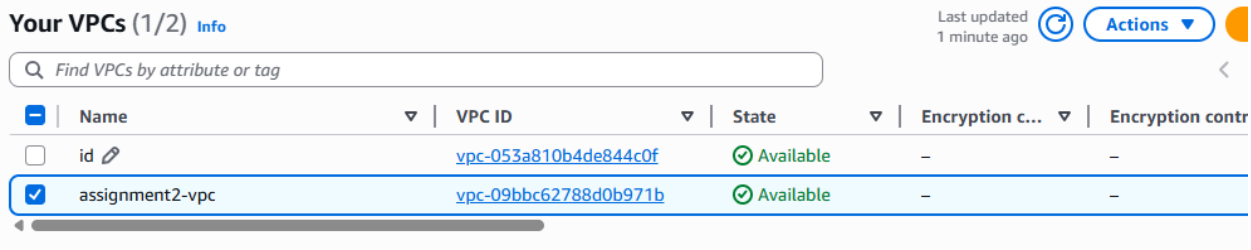


Check Outputs

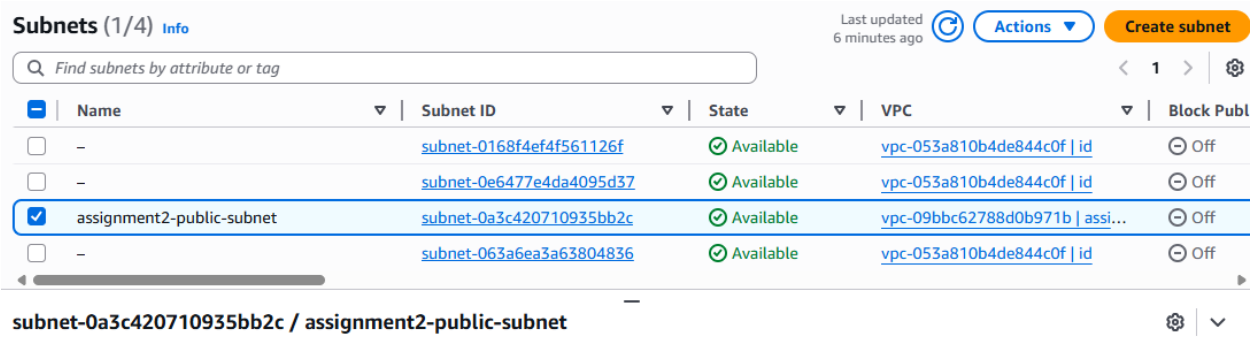


Step 4.3: AWS Console Verification

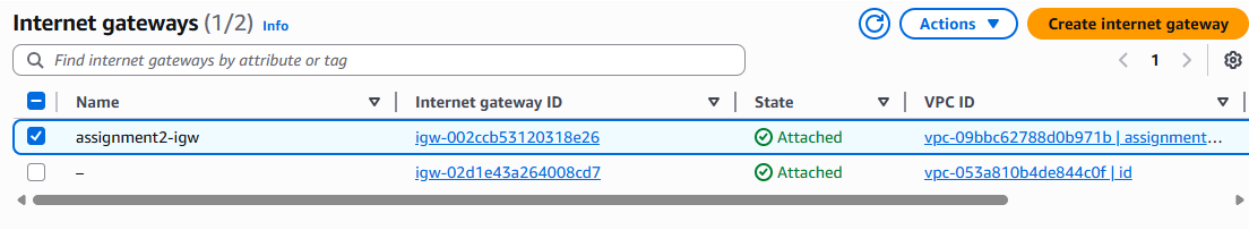
aws_vpc



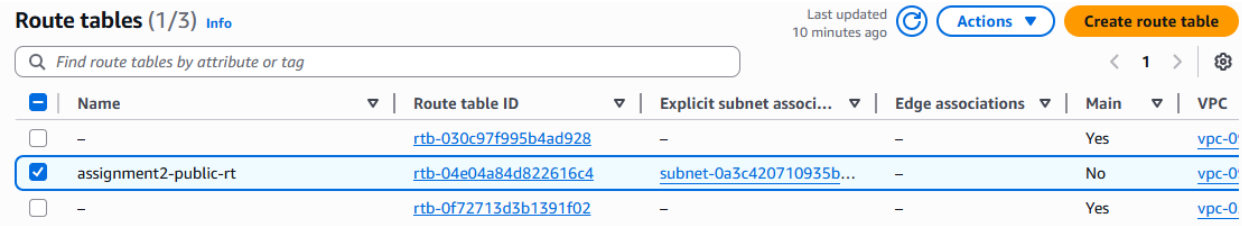
aws_subnet



aws_security_groups



aws_instances



****

Part 5: Nginx Configuration & Testing

5.1 Update Nginx Backend Configuration

SSH into the Nginx Server

```
I&I LT@Kainat-Shakeel MINGW64 ~/Desktop/Assignment2 (master)
$ ssh -i ./scripts/assignment2-key.pem ec2-user@43.204.23.253
The authenticity of host '43.204.23.253 (43.204.23.253)' can't be established.
ED25519 key fingerprint is SHA256:rE7Sy1bwOq1EUe3dgbcy4ThhjAzPFiqmB667HYe1dHs.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '43.204.23.253' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   Amazon Linux 2023 (ECS Optimized)
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

For documentation, visit http://aws.amazon.com/documentation/ecs
[ec2-user@ip-10-0-1-245 ~]$
```

nginx_conf_updated

```
ec2-user@ip-10-0-1-245:~
  GNU nano 8.3                                                                /et
#   * Official English Documentation: http://nginx.org/en/docs/
#   * Official Russian Documentation: http://nginx.org/ru/docs/

user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    # Define backend servers
    upstream backend_servers {
        server 10.0.1.101:80;   # web-1 private IP
        server 10.0.1.102:80;   # web-2 private IP
        server 10.0.1.103:80 backup;   # web-3 private IP
    }

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

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

    # Load modular configuration files from the /etc/nginx/conf.d directory
    include /etc/nginx/conf.d/*.conf;

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

test the Nginx configuration

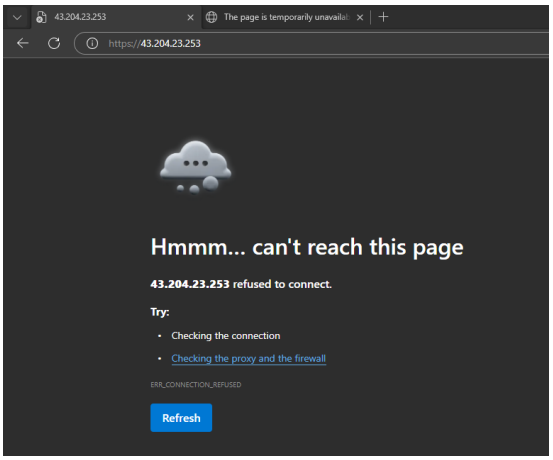
```
[ec2-user@ip-10-0-1-245 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-1-245 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-1-245 ~]$
```

restart Nginx

```
ec2-user@ip-10-0-1-245:~

For documentation, visit http://aws.amazon.com/documentation/ecs
Last login: Sun Dec 28 15:12:12 2025 from 203.215.174.10
[ec2-user@ip-10-0-1-245 ~]$ sudo nano /etc/nginx/nginx.conf
[ec2-user@ip-10-0-1-245 ~]$ 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-10-0-1-245 ~]$ sudo systemctl restart nginx
sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: di>
     Active: active (running) since Sun 2025-12-28 16:25:57 UTC; 56ms ago
    Process: 204684 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, st>
    Process: 204685 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCC>
    Process: 204686 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 204687 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.3M
        CPU: 59ms
     CGroup: /system.slice/nginx.service
             ├─204687 "nginx: master process /usr/sbin/nginx"
             ├─204688 "nginx: worker process"
             └─204689 "nginx: worker process"

Dec 28 16:25:57 ip-10-0-1-245.ap-south-1.compute.internal systemd[1]: Starting >
Dec 28 16:25:57 ip-10-0-1-245.ap-south-1.compute.internal nginx[204685]: nginx:>
Dec 28 16:25:57 ip-10-0-1-245.ap-south-1.compute.internal nginx[204685]: nginx:>
Dec 28 16:25:57 ip-10-0-1-245.ap-south-1.compute.internal systemd[1]: Started n>
lines 1-19/19 (END)...skipping...
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sun 2025-12-28 16:25:57 UTC; 56ms ago
    Process: 204684 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status=0/SUCCESS)
    Process: 204685 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
    Process: 204686 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 204687 (nginx)
      Tasks: 3 (limit: 1067)
     Memory: 3.3M
        CPU: 59ms
     CGroup: /system.slice/nginx.service
             ├─204687 "nginx: master process /usr/sbin/nginx"
             ├─204688 "nginx: worker process"
             └─204689 "nginx: worker process"
```

5.2 Test Load Balancing

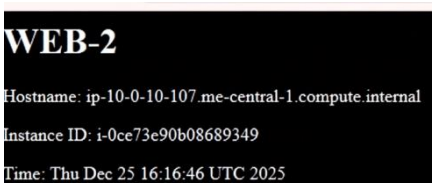
browser security warning



showing web-1 content

```
[ec2-user@ip-10-0-10-75 ~]$
[ec2-user@ip-10-0-10-75 ~]$ curl -k -s "https://localhost/?r=$RANDOM" | grep -oE 'i-[0-9a-f]+' | head -n 1
i-0f0428b79f1efc916
[ec2-user@ip-10-0-10-75 ~]$
```

showing web-2 content



multiple reloads showing alternation

```
i-0cc21e871d1d28299
[ec2-user@ip-10-0-10-75 ~]$ for i in {1..12}; do curl -k -s "https://localhost/?r=$RANDOM" | grep -oE 'i-[0-9a-f]+' | head -n 1; done
i-0f0428b79f1efc916
i-0ce73e90b08689349
i-0f0428b79f1efc916
i-0ce73e90b08689349
i-0f0428b79f1efc916
i-0ce73e90b08689349
i-0f0428b79f1efc916
i-0ce73e90b08689349
i-0f0428b79f1efc916
i-0f0428b79f1efc916
i-0ce73e90b08689349
i-0f0428b79f1efc916
[ec2-user@ip-10-0-10-75 ~]$
```

5.3 Test Cache Functionality

first request - MISS

```
[ec2-user@ip-10-0-10-75 ~]$ curl -k -I "https://localhost/?x=miss1" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   159    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: MISS
```

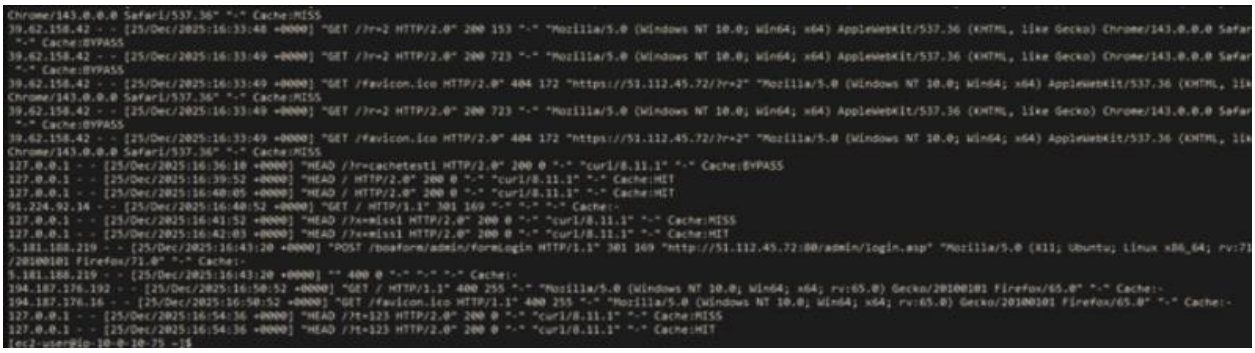
second request – HIT

```
[ec2-user@ip-10-0-10-75 ~]$ curl -k -I "https://localhost/?x=miss1" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   159    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: HIT
```

cache folder contents

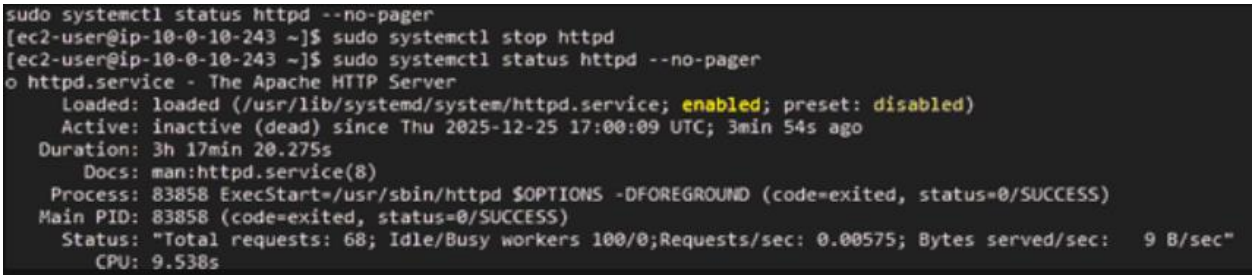
```
x-cache-status: HIT
[ec2-user@ip-10-0-10-75 ~]$ sudo ls -lah /var/cache/nginx
 -type f | head
sudo find /var/cache/nginx -type f | head
total 0
drwx------. 17 nginx root  141 Dec 25 15:12 .
drwxr-xr-x.  9 root  root  101 Dec 24 17:00 ..
drwx------.  5 nginx nginx  36 Dec 25 14:55 0
drwx------.  8 nginx nginx  66 Dec 25 15:12 1
drwx------.  7 nginx nginx  56 Dec 25 14:55 2
drwx------.  6 nginx nginx  46 Dec 25 15:12 4
drwx------.  9 nginx nginx  76 Dec 25 15:12 5
drwx------.  5 nginx nginx  36 Dec 25 15:12 6
drwx------.  7 nginx nginx  56 Dec 25 15:12 7
drwx------.  4 nginx nginx  26 Dec 25 14:55 8
drwx------.  4 nginx nginx  26 Dec 25 15:12 9
drwx------.  6 nginx nginx  46 Dec 25 14:55 a
drwx------.  8 nginx nginx  66 Dec 25 16:41 b
drwx------.  8 nginx nginx  66 Dec 25 15:12 c
drwx------.  6 nginx nginx  46 Dec 25 15:12 d
drwx------.  3 nginx nginx  16 Dec 25 15:12 e
drwx------.  5 nginx nginx  36 Dec 25 14:54 f
[ec2-user@ip-10-0-10-75 ~]$
```

access log showing cache status



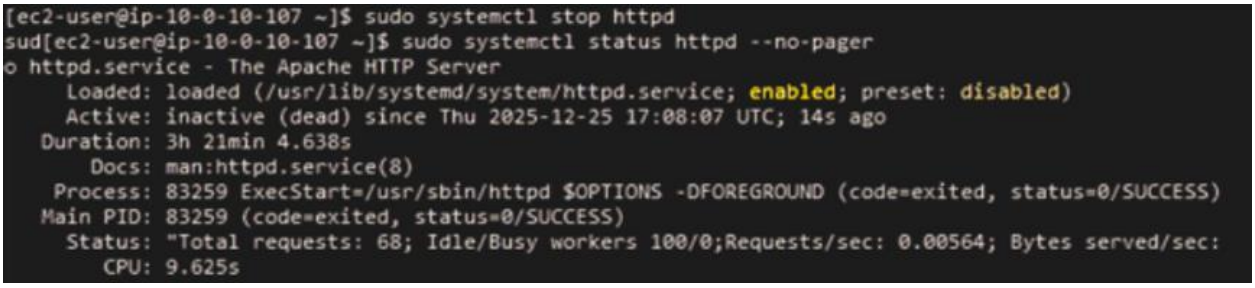
5.4 Test High Availability

web-1 Apache stopped



```
sudo systemctl status httpd --no-pager
[ec2-user@ip-10-0-10-243 ~]$ sudo systemctl stop httpd
[ec2-user@ip-10-0-10-243 ~]$ sudo systemctl status httpd --no-pager
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: inactive (dead) since Thu 2025-12-25 17:00:09 UTC; 3min 54s ago
   Duration: 3h 17min 20.275s
       Docs: man:httpd.service(8)
    Process: 83858 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 83858 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 68; Idle/Busy workers 100/0;Requests/sec: 0.00575; Bytes served/sec:   9 B/sec"
        CPU: 9.538s
```

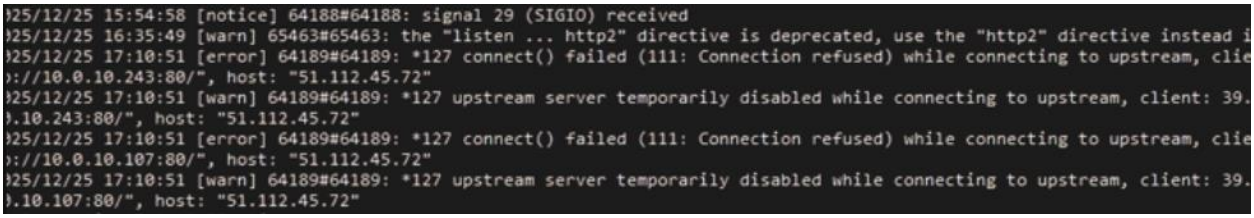
web-2 Apache stopped



```
[ec2-user@ip-10-0-10-107 ~]$ sudo systemctl stop httpd
sud[ec2-user@ip-10-0-10-107 ~]$ sudo systemctl status httpd --no-pager
o httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: inactive (dead) since Thu 2025-12-25 17:08:07 UTC; 14s ago
   Duration: 3h 21min 4.638s
       Docs: man:httpd.service(8)
    Process: 83259 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=0/SUCCESS)
   Main PID: 83259 (code=exited, status=0/SUCCESS)
     Status: "Total requests: 68; Idle/Busy workers 100/0;Requests/sec: 0.00564; Bytes served/sec:
        CPU: 9.625s
```

web-3 serving traffic

error log showing backend failures



```
025/12/25 15:54:58 [notice] 64188#64188: signal 29 (SIGIO) received
025/12/25 16:35:49 [warn] 65463#65463: the "listen ... http2" directive is deprecated, use the "http2" directive instead i
025/12/25 17:10:51 [error] 64189#64189: *127 connect() failed (111: Connection refused) while connecting to upstream, clie
p://10.0.10.243:80/", host: "51.112.45.72"
025/12/25 17:10:51 [warn] 64189#64189: *127 upstream server temporarily disabled while connecting to upstream, client: 39.
).10.243:80/", host: "51.112.45.72"
025/12/25 17:10:51 [error] 64189#64189: *127 connect() failed (111: Connection refused) while connecting to upstream, clie
p://10.0.10.107:80/", host: "51.112.45.72"
025/12/25 17:10:51 [warn] 64189#64189: *127 upstream server temporarily disabled while connecting to upstream, client: 39.
).10.107:80/", host: "51.112.45.72"
```

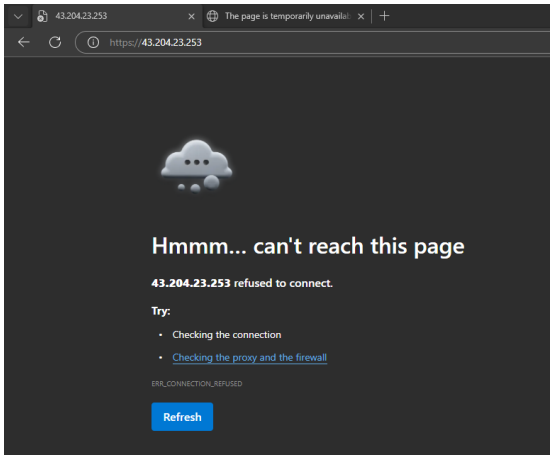
all services back online

```
[ec2-user@ip-10-0-10-243 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-10-243 ~]$ sudo systemctl status httpd --no-pager
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Thu 2025-12-25 17:15:52 UTC; 27s ago
       Docs: man:httpd.service(8)
   Main PID: 93130 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:
      Tasks: 177 (limit: 1067)
     Memory: 13.3M
        CPU: 86ms
     CGroup: /system.slice/httpd.service
             ├─93130 /usr/sbin/httpd -DFOREGROUND
             ├─93131 /usr/sbin/httpd -DFOREGROUND
             ├─93132 /usr/sbin/httpd -DFOREGROUND
             ├─93133 /usr/sbin/httpd -DFOREGROUND
             └─93139 /usr/sbin/httpd -DFOREGROUND
```

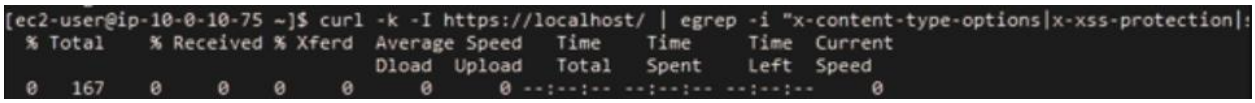
```
[ec2-user@ip-10-0-10-107 ~]$ sudo systemctl start httpd
[ec2-user@ip-10-0-10-107 ~]$ sudo systemctl status httpd --no-pager
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Thu 2025-12-25 17:17:31 UTC; 14s ago
       Docs: man:httpd.service(8)
   Main PID: 92565 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:
      Tasks: 177 (limit: 1067)
     Memory: 13.3M
        CPU: 75ms
     CGroup: /system.slice/httpd.service
             ├─92565 /usr/sbin/httpd -DFOREGROUND
             ├─92566 /usr/sbin/httpd -DFOREGROUND
             ├─92567 /usr/sbin/httpd -DFOREGROUND
             ├─92568 /usr/sbin/httpd -DFOREGROUND
             └─92569 /usr/sbin/httpd -DFOREGROUND
```

5.5 Security & Performance Analysis

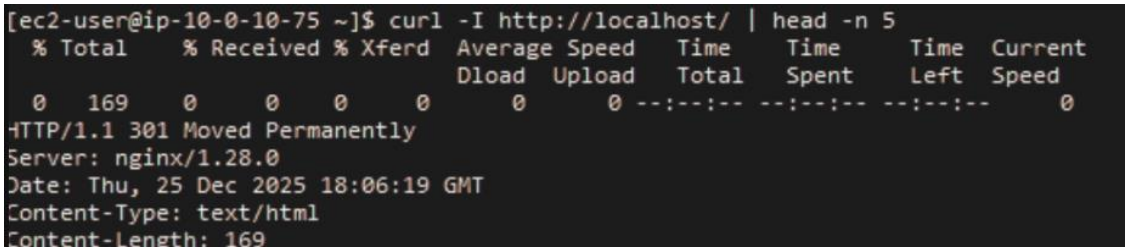
certificate details



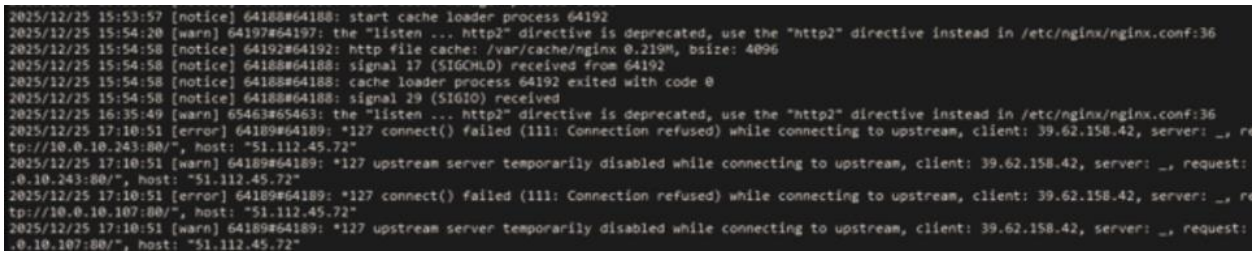
response headers showing security headers



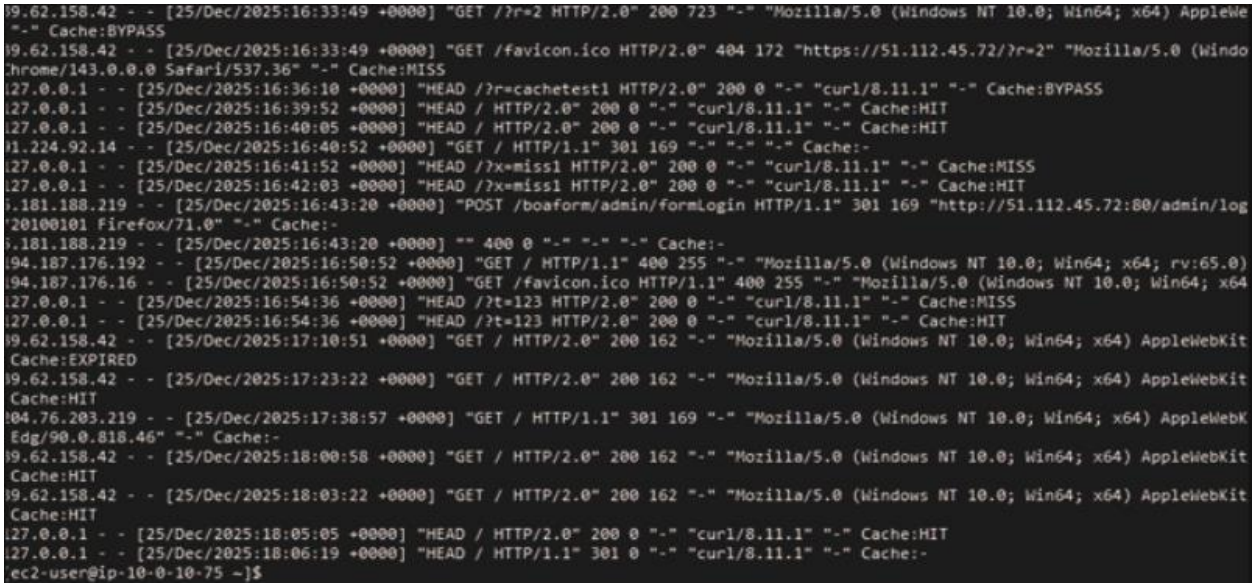
301 redirect test



error log review



access log patterns



****

Bonus Tasks

Bonus 1: Custom Error Pages

Bonus 2 - Rate Limiting

rate limit config

```
[ec2-user@ip-10-0-10-75 ~]$ sudo nginx -t
sudo systemctl reload nginx
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2" directive instead in /etc/nginx/nginx.conf:3
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Rate limit test

```
[ec2-user@ip-10-0-10-75 ~]$ sudo systemctl reload nginx
[ec2-user@ip-10-0-10-75 ~]$ seq 1 120 | xargs -I{} -P40 sh -c 'curl -k -s -o /dev/null -w "%{http_code}\n" '
      3 200
    117 429
```

Bonus 3: Health Check Automation

Health check script

```
[ec2-user@ip-10-0-10-75 ~]$ sudo nginx -t
nginx: [warn] duplicate MIME type "text/html" in /etc/nginx/nginx.conf:12
nginx: [warn] the "listen ... http2" directive is deprecated, use the "http2"
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-10-0-10-75 ~]$ sudo systemctl reload nginx
```

Health log

```
[ec2-user@ip-10-0-10-75 ~]$ sudo systemctl reload nginx
[ec2-user@ip-10-0-10-75 ~]$ curl -k -I -H "Accept-Encoding: gzip" https://localhost/ | grep -i content-encoding
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
content-encoding: gzip
```

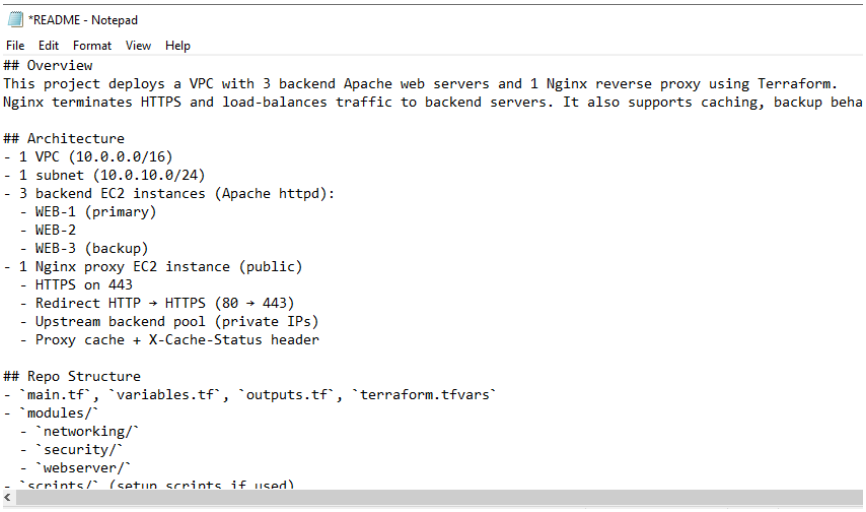
****

Part 6: Documentation & Cleanup

6.1 README Documentation

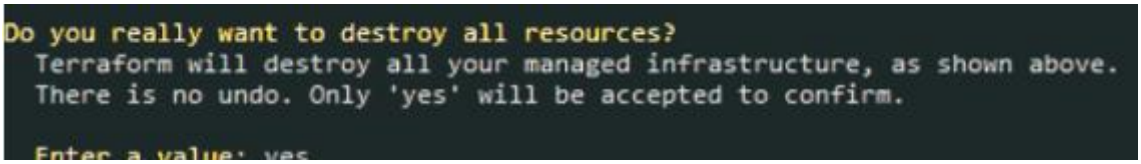
Documented architecture, prerequisites, deployment steps, configuration guide, and troubleshooting in README.md.

README.md content

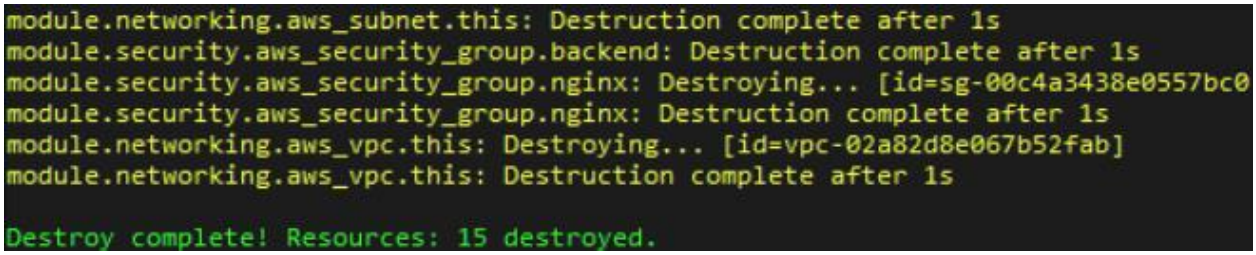


Infrastructure Cleanup

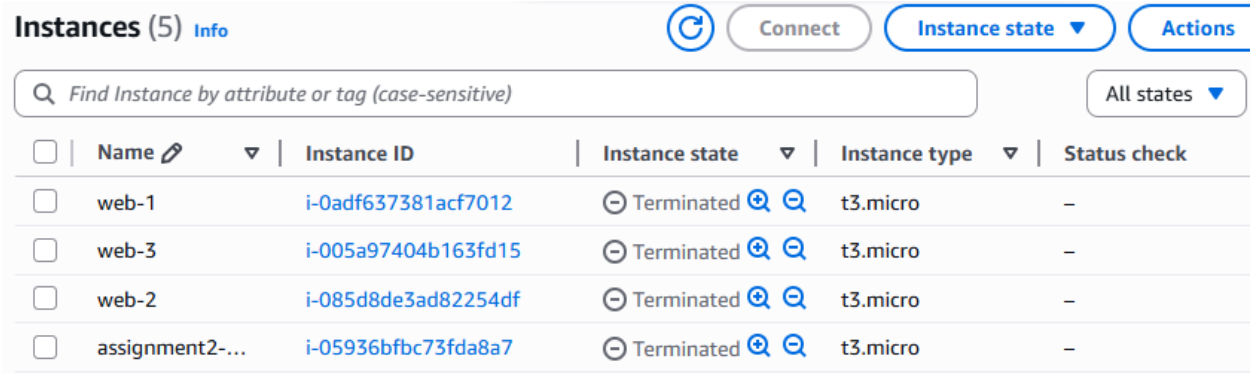
Destroyed all Terraform-managed resources and verified cleanup in AWS Console.



Terraform destroy complete



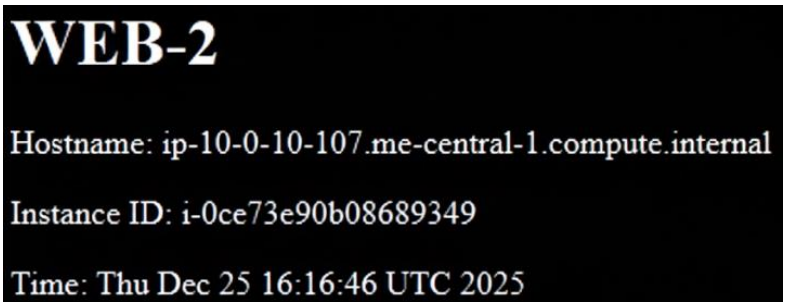
Empty state



terraform.tfstate after destroy showing no resources.

Testing Results Summary

• Load balancing alternated between web-1 and web-2 across multiple reloads.



• Cache status changed from MISS (first request) to HIT (subsequent requests).

```
[ec2-user@ip-10-0-10-75 ~]$ curl -k -I "https://localhost/?x=miss1" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   159    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: MISS
```

Backup

```
[ec2-user@ip-10-0-10-75 ~]$ curl -k -I "https://localhost/?x=miss1" | grep -i "x-cache-status"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   159    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-cache-status: HIT
```

****