Cloud Computing Project Report

Project 6 – High-Performance Web Hosting Platform

Semester: 5B

Submitted To:
Sir Waqas

Group Members:
Nayab Khazin_046
Safa Jahangir_056
Maria Abdul Malik_076

Submission Date: 27-01-2026

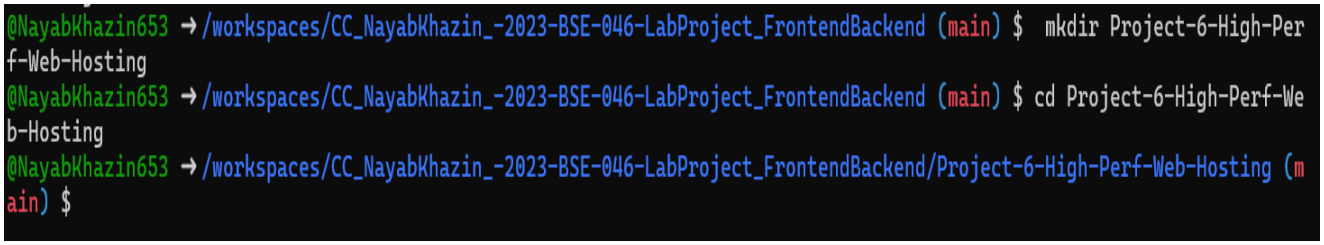
Contents

Part 1: Git Repository Setup

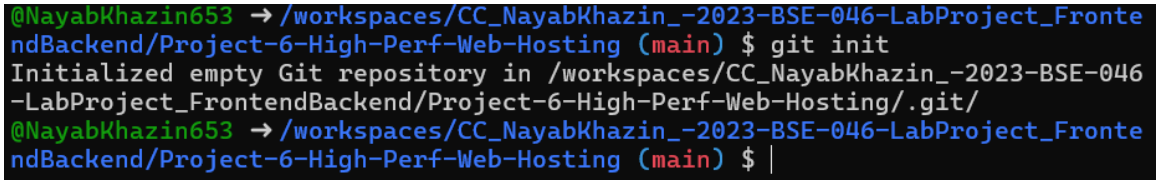
1.1 Repository Structure

The repository was structured according to the project requirements, separating Terraform, Ansible, website content, and documentation. This modular structure improves maintainability and scalability.

- **project6_part1_repository_structure.png**



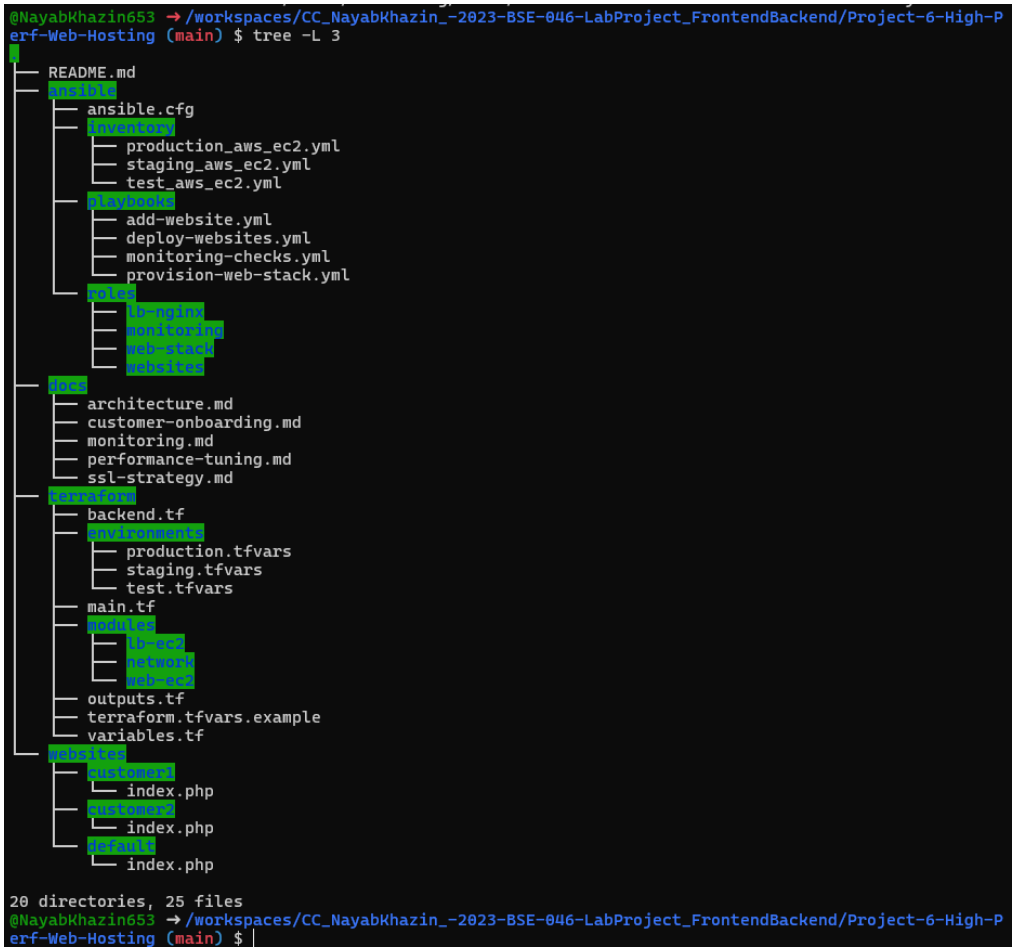
- **project6_part1_initial_commit.png**



1.2 Git Branching Strategy

A three-level branching strategy was followed:
main: production
staging: pre-production
test: internal testing
Feature branches were merged into test before promotion.

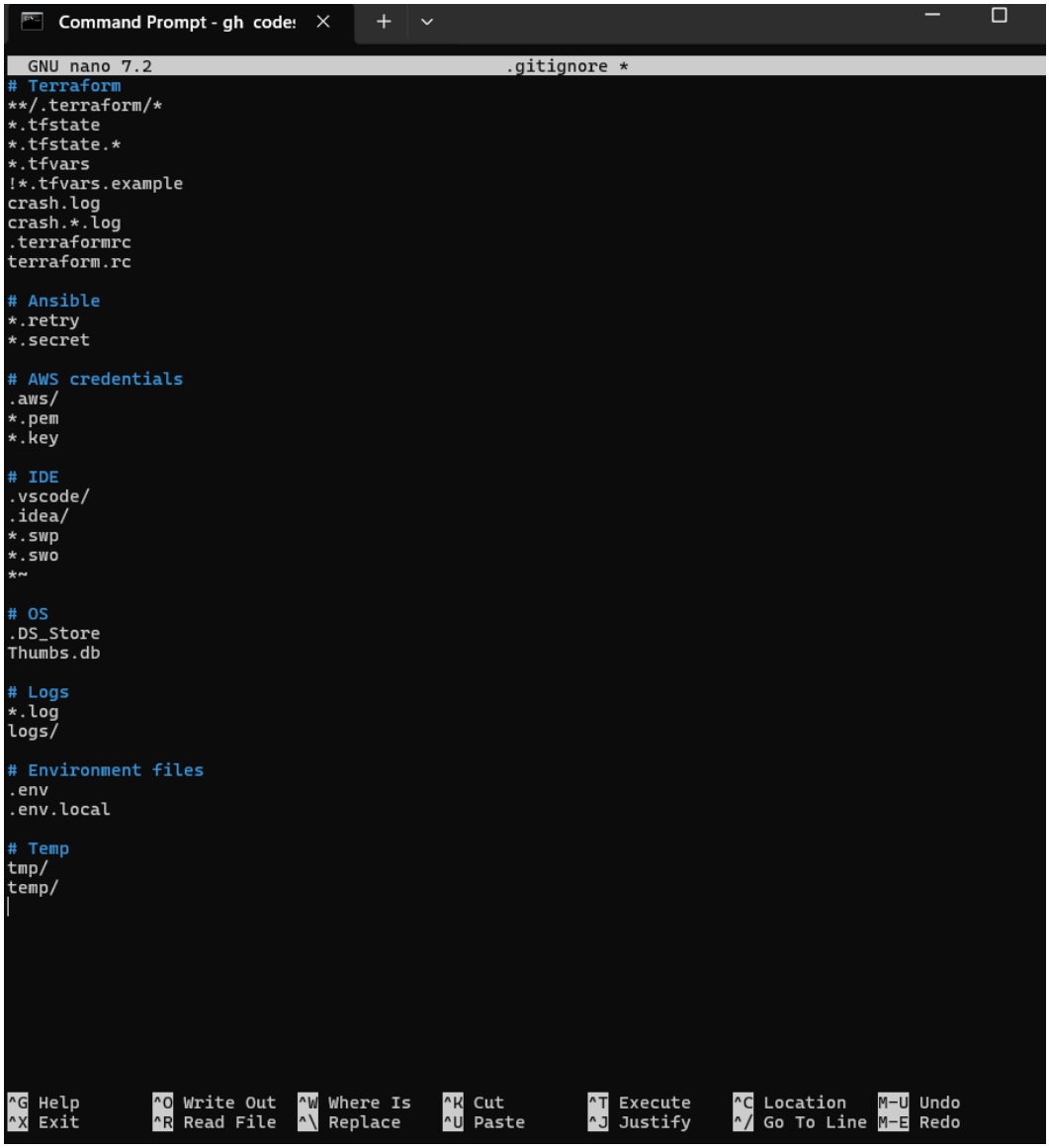
- **project6_part1_git_branches.png**



1.3 .gitignore Configuration

.gitignore was configured to prevent Terraform state files, credentials, keys, and temporary files from being committed.

- **project6_part1_gitignore_content.png**



- **project6_part1_git_status_clean.png**

```
erf-Web-Hosting (main) $ git add .
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/Project-6-High-P
erf-Web-Hosting (main) $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   README.md
        new file:   ansible/ansible.cfg
        new file:   ansible/inventory/production_aws_ec2.yml
        new file:   ansible/inventory/staging_aws_ec2.yml
        new file:   ansible/inventory/test_aws_ec2.yml
        new file:   ansible/playbooks/add-website.yml
        new file:   ansible/playbooks/deploy-websites.yml
        new file:   ansible/playbooks/monitoring-checks.yml
        new file:   ansible/playbooks/provision-web-stack.yml
        new file:   docs/architecture.md
        new file:   docs/customer-onboarding.md
        new file:   docs/monitoring.md
        new file:   docs/performance-tuning.md
        new file:   docs/ssl-strategy.md
        new file:   terraform/backend.tf
        new file:   terraform/main.tf
        new file:   terraform/modules/lb-ec2/main.tf
        new file:   terraform/modules/lb-ec2/outputs.tf
        new file:   terraform/modules/lb-ec2/variables.tf
        new file:   terraform/modules/network/main.tf
        new file:   terraform/modules/network/outputs.tf
        new file:   terraform/modules/network/variables.tf
        new file:   terraform/modules/web-ec2/main.tf
        new file:   terraform/modules/web-ec2/outputs.tf
        new file:   terraform/modules/web-ec2/variables.tf
        new file:   terraform/outputs.tf
        new file:   terraform/terraform.tfvars.example
        new file:   terraform/variables.tf
        new file:   websites/customer1/index.php
        new file:   websites/customer2/index.php
        new file:   websites/default/index.php

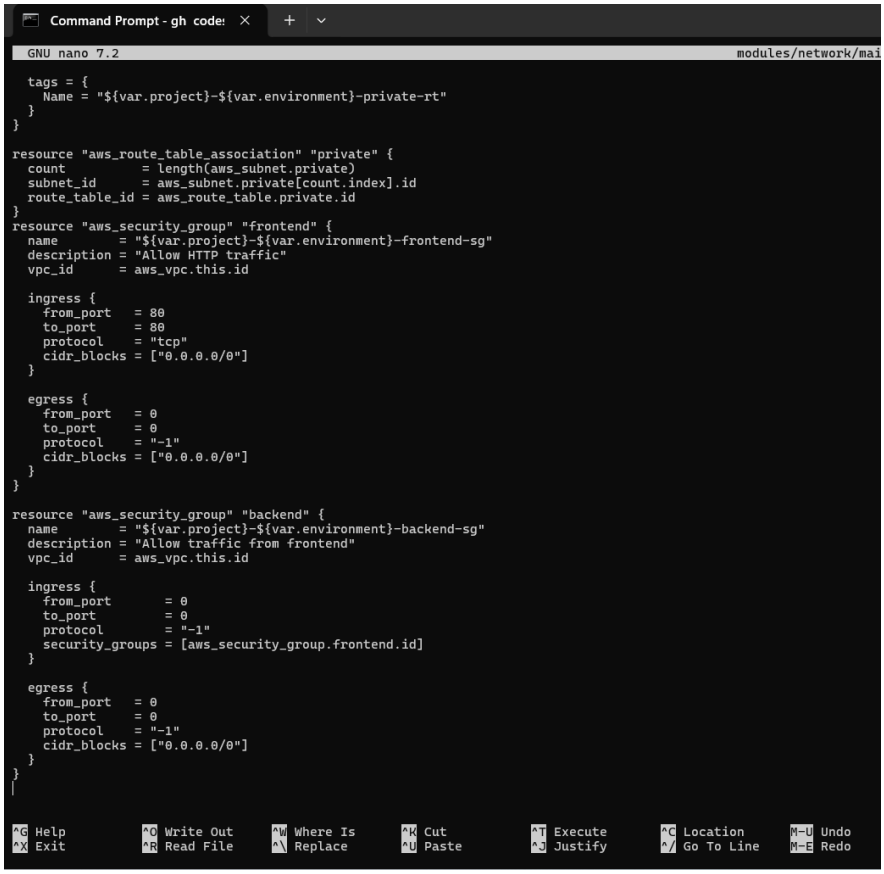
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/Project-6-High-P
erf-Web-Hosting (main) $
```

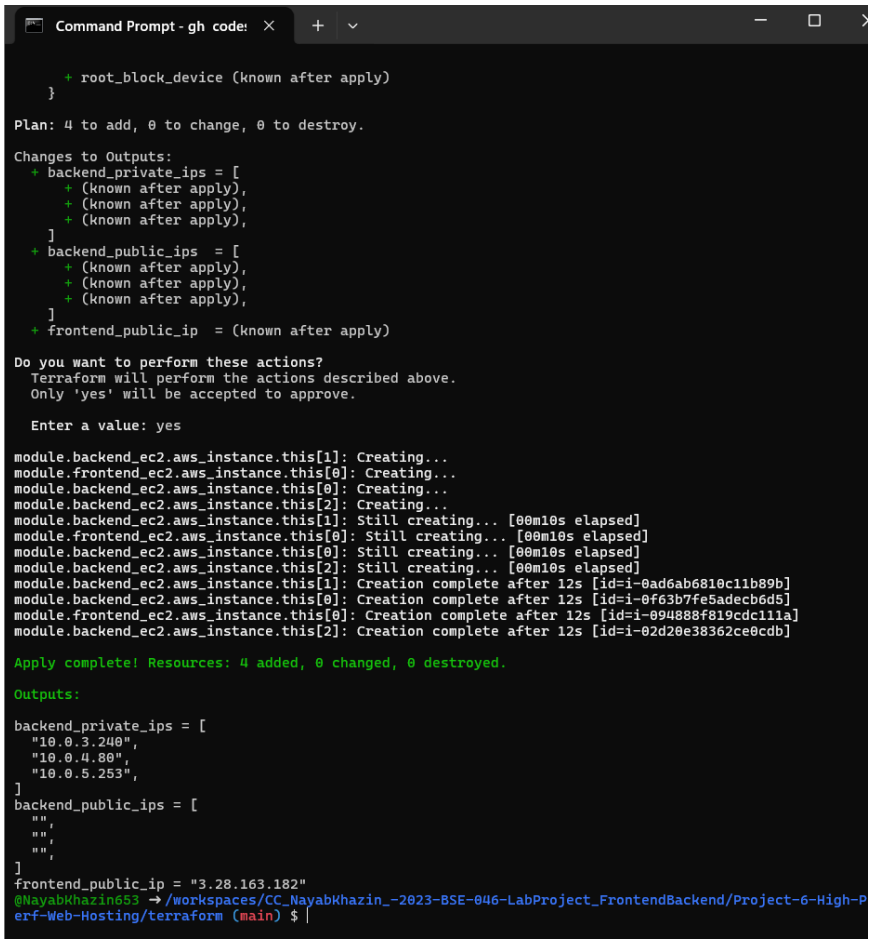
Part 2: Terraform Infrastructure

2.1 Network Module (VPC & Subnets)

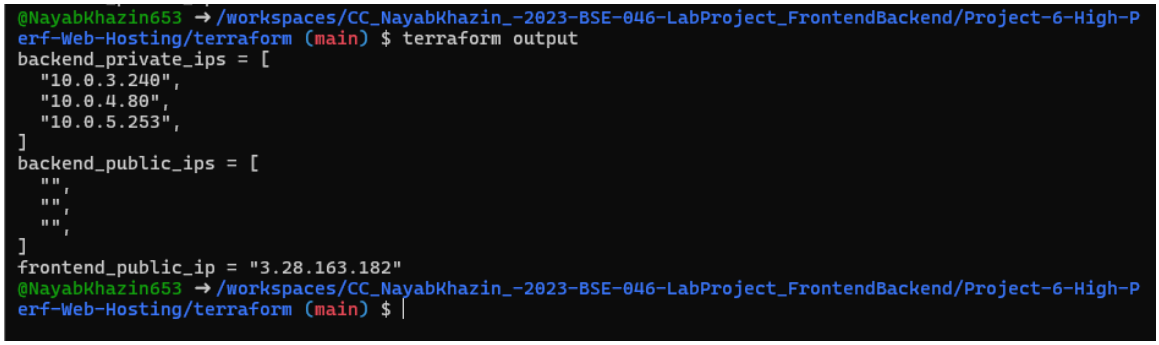
Terraform was used to provision a VPC with public and private subnets across two Availability Zones. An Internet Gateway and route tables were configured. A NAT Gateway was used to allow outbound access for private instances.

- **project6_part2_network_main.png**


Command Prompt - gh codes
GNU nano 7.2
modules/network/mai
tags = {
Name = "${var.project}-${var.environment}-private-rt"
}
}
resource "aws_route_table_association" "private" {
count = length(aws_subnet.private)
subnet_id = aws_subnet.private[count.index].id
route_table_id = aws_route_table.private.id
}
resource "aws_security_group" "frontend" {
name = "${var.project}-${var.environment}-frontend-sg"
description = "Allow HTTP traffic"
vpc_id = aws_vpc.this.id
ingress {
from_port = 80
to_port = 80
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"]
}
}
resource "aws_security_group" "backend" {
name = "${var.project}-${var.environment}-backend-sg"
description = "Allow traffic from frontend"
vpc_id = aws_vpc.this.id
ingress {
from_port = 0
to_port = 0
protocol = "-1"
security_groups = [aws_security_group.frontend.id]
}
egress {
from_port = 0
to_port = 0
protocol = "-1"
cidr_blocks = ["0.0.0.0/0"]
}
}
^G Help
^O Write Out
^W Where Is
^K Cut
^T Execute
^C Location
M-U Undo
^X Exit
^R Read File
^\ Replace
^U Paste
^J Justify
^/ Go To Line
M-E Redo



- **project6_part2_network_outputs.png**



2.2 Load Balancer EC2 Module

Two EC2 instances were provisioned as Nginx-based load balancers in public subnets. Security groups allow HTTP/HTTPS from the internet and SSH from the admin IP.

- **project6_part2_lb_ec2_main.png**

```
module "frontend_ec2" {
  source = "./modules/ec2"

  ami_id             = var.ami_id
  instance_type      = var.instance_type
  key_name           = var.key_name
  subnet_ids         = [module.network.public_subnet_ids[0]]
  security_group_ids = [module.network.frontend_sg_id]
  instance_count     = 1
  security_group_id  = module.network.frontend_sg_id
  name               = "frontend"
}
module "backend_ec2" {
  source = "./modules/ec2"

  ami_id             = var.ami_id
  instance_type      = var.instance_type
  key_name           = var.key_name
  subnet_ids         = module.network.private_subnet_ids
  security_group_ids = [module.network.backend_sg_id]
  instance_count     = 3
  security_group_id  = module.network.frontend_sg_id
  name               = "backend"
}
```

```
Outputs:

backend_private_ips = [
  "10.0.1.36",
  "10.0.2.221",
  "10.0.1.9",
]
backend_public_ips = [
  "3.28.74.111",
  "3.28.128.120",
  "51.112.50.165",
]
frontend_public_ip = "3.28.163.182"
```

- **project6_part2_lb_ec2_outputs.png**

```
  GNU nano 7.2                              modules/ec2/outputs.tf
output "public_ips" {
  value = aws_instance.this[*].public_ip
}

output "private_ips" {
  value = aws_instance.this[*].private_ip
}
```

```
module.network.aws_route_table_association.private[0]: Refreshing state... [id=rtbassoc-05a8c4d920f321
e12]
module.network.aws_route_table_association.private[1]: Refreshing state... [id=rtbassoc-0bcc39afdb0bf4
ff9]

No changes. Your infrastructure matches the configuration.

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

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

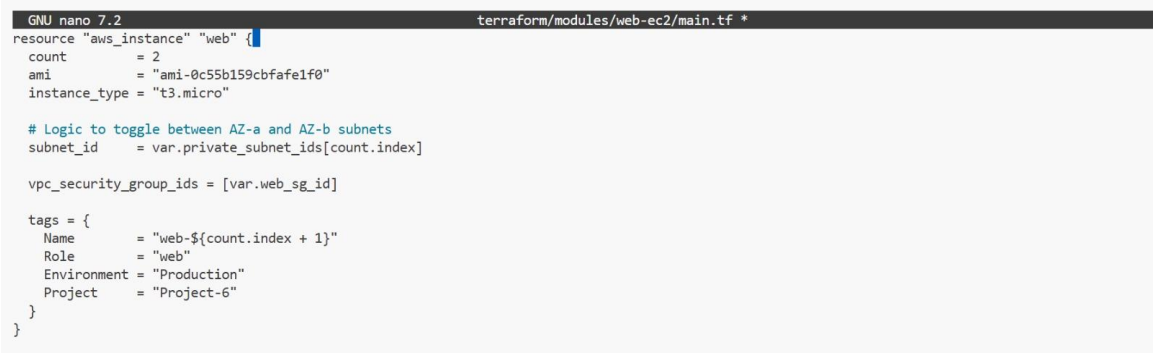
Outputs:

backend_private_ips = [
  "10.0.1.211",
  "10.0.2.109",
  "10.0.1.20",
]
backend_public_ips = [
  "40.172.221.225",
  "158.252.66.106",
  "3.28.74.45",
]
frontend_public_ips = [
  "3.29.113.93",
]
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/Project-6-High-P
erf-Web-Hosting/terraform (main) $
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/Project-6-High-P
erf-Web-Hosting/terraform (main) $
```

2.3 Web EC2 Module

Web servers were deployed in private subnets across multiple AZs. They accept traffic only from the load balancer security group.

- **project6_part2_web_ec2_main.png**



- **project6_part2_web_ec2_outputs.png**

Part 3: Environment Configuration

3.1 Root Terraform Configuration

The root Terraform configuration instantiates all modules and exports outputs such as load balancer public IPs and web server private IPs for Ansible use.

- **project6_part3_main_tf.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting (main) $ cat terraform/main.tf
module "network" {
  source = "./modules/network"

  vpc_cidr       = "10.0.0.0/16"

  public_subnets = [
    "10.0.1.0/24",
    "10.0.2.0/24"
  ]

  private_subnets = [
    "10.0.3.0/24",
    "10.0.4.0/24",
    "10.0.5.0/24"
  ]

  azs = [
    "me-central-1a",
    "me-central-1b",
    "me-central-1c"
  ]

  environment = "dev"
```

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting (main) $ cat terraform/main.tf

module "frontend_ec2" {
  source = "./modules/ec2"

  ami_id             = var.ami_id
  instance_type      = var.instance_type
  key_name           = var.key_name
  subnet_ids         = module.network.public_subnet_ids
  security_group_ids = [module.network.frontend_sg_id]
  instance_count     = 2
  security_group_id  = module.network.frontend_sg_id
  name               = "frontend"
}
module "backend_ec2" {
  source = "./modules/ec2"

  ami_id             = var.ami_id
  instance_type      = var.instance_type
  key_name           = var.key_name
  subnet_ids         = [module.network.public_subnet_ids[0]]
  security_group_ids = [module.network.backend_sg_id]
  instance_count     = 2
  security_group_id  = module.network.backend_sg_id
  name               = "backend"
```

```
 GNU nano 7.2                                                terraform/main.tf *
provider "aws" {
  region = "me-central-1"
}

module "network" {
  source = "./modules/network"

  vpc_cidr       = "10.0.0.0/16"

  public_subnets = [
    "10.0.1.0/24",
    "10.0.2.0/24"
  ]

  private_subnets = [
```

- **project6_part3_outputs_tf.png**

```
@SafaJahangir09 →/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting (main) $ cat terraform/outputs.tf
output "frontend_public_ips" {
  value = module.frontend_ec2.public_ips
}

output "backend_private_ips" {
  value = module.backend_ec2.private_ips
}

output "backend_public_ips" {
  value = module.backend_ec2.public_ips
}
```

3.2 Environment-Specific Variables

Separate tfvars files were created for test, staging, and production environments. These define instance sizes, counts, and admin IP restrictions.

- **project6_part3_test_tfvars.png**

```
@SafaJahangir09 →.../CC_Project/Project-6-High-Perf-Web-Hosting/terraform/environments (main) $ cat test.tfvars
environment       = "test"
web_instance_type = "t3.micro"
web_count         = 2
admin_ip_cidr     = "158.252.78.189/32"
```

- **project6_part3_staging_tfvars.png**

```
@SafaJahangir09 →.../CC_Project/Project-6-High-Perf-Web-Hosting/terraform/environments (main) $ cat staging.tfvars
environment       = "staging"
web_instance_type = "t3.small"
web_count         = 3
admin_ip_cidr     = "40.172.231.171/32"
```

- **project6_part3_production_tfvars.png**

```
@SafaJahangir09 →.../CC_Project/Project-6-High-Perf-Web-Hosting/terraform/environments (main) $ cat production.tfvars
environment       = "production"
web_instance_type = "t3.medium"
web_count         = 2
admin_ip_cidr     = "0.0.0.0/0"
```

- **project6_part3_tfvars_example.png**

```
@SafaJahangir09 →/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/terraform (main) $ cat terraform.tfvars.example
# Template for Project 6 Infrastructure
environment       = "default"
web_instance_type = "t3.micro"
web_count         = 2
```

Part 4: Ansible Web Stack & Websites

4.1 Dynamic Inventory Configuration

Ansible uses the aws_ec2 dynamic inventory plugin to automatically discover instances based on tags.

- **project6_part4_ansible_inventory_graph.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ ansible-inventory -i aws_ec2.yml --graph
[WARNING]: Deprecation warnings can be disabled by setting `deprecation_warnings=False` in ansible.cfg.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature will be removed from ansible-core version 2.23.
@all:
  |--@ungrouped:
  |--@aws_ec2:
  |  |--public-ip-address
  |--@frontend_1:
  |  |--public-ip-address
  |--@backend_2:
  |  |--public-ip-address
  |--@backend_1:
  |  |--public-ip-address
  |--@frontend_2:
  |  |--public-ip-address
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $
```

- **project6_part4_ansible_cfg.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat ansible.cfg
[defaults]
roles_path = ./roles
inventory = ./inventory
```

4.2 Web Stack Role (Nginx + PHP-FPM)

The web-stack role installs and configures Nginx and PHP-FPM with performance tuning. Multiple customer websites are hosted using Nginx server blocks.

- **project6_part4_web_stack_role_main.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/web-stack/tasks/main.yml
---
- name: Install Nginx and PHP-FPM
  apt:
    name:
      - nginx
      - php-fpm
      - php-cli
    state: present
    update_cache: yes

- name: Performance Tuning - Configure PHP-FPM pool
  template:
    src: php-fpm.conf.j2
    dest: /etc/php/{{ php_version }}/fpm/pool.d/www.conf
  notify: restart php-fpm

- name: Ensure web directories exist for customers
  file:
    path: "/var/www/{{ item }}"
    state: directory
    owner: www-data
    group: www-data
    mode: '0755'
  with_items:
```

```
@SafaJahangir09 →/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/web-stack/tasks/main.yml
    state: directory
    owner: www-data
    group: www-data
    mode: '0755'
  with_items:
    - customer1
    - customer2
    - default

- name: Deploy Nginx Virtual Hosts for Customers
  template:
    src: customer-site.conf.j2
    dest: "/etc/nginx/sites-available/{{ item }}.conf"
  with_items:
    - customer1
    - customer2
  notify: restart nginx

- name: Enable customer sites
  file:
    src: "/etc/nginx/sites-available/{{ item }}.conf"
    dest: "/etc/nginx/sites-enabled/{{ item }}.conf"
    state: link
  with_items:
```

- **project6_part4_nginx_server_blocks.png**

```
@SafaJahangir09 →/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/web-stack/templates/php-fpm.conf.j2
user = www-data
group = www-data
listen = /var/run/php/php{{ php_version }}-fpm.sock
listen.owner = www-data
listen.group = www-data

; Requirement: Performance Tuning
pm = dynamic
pm.max_children = {{ php_fpm_max_children | default(5) }}
pm.start_servers = {{ php_fpm_start_servers | default(2) }}
pm.min_spare_servers = 1
pm.max_spare_servers = 3
@SafaJahangir09 →/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/web-stack/templates/customer-site.conf.j2
server {
    listen 80;
    server_name {{ item }}.local;
    root /var/www/{{ item }};
    index index.php index.html;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php{{ php_version }}-fpm.sock;
    }
}
```

4.3 Load Balancer Nginx Role

The load balancer role configures Nginx as a reverse proxy with SSL termination, caching, and a backup server.

- **project6_part4_lb_nginx_conf.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/lb-nginx/templates/lb-nginx.conf.j2
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=my_cache:10m inactive=60m;
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name _;
    return 301 https://$host$request_uri;
}


server {
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    server_name _;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

    location / {
        proxy_cache my_cache;
        proxy_cache_valid 200 60m;
        proxy_pass http://frontend_cluster;
        proxy_set_header Host customer1.local;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    add_header X-Cache-Status $upstream_cache_status;
```

- **project6_part4_lb_ssl_certs.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ ssh -i ./nayab-key.pem ubuntu@158.252.78.189 "ls -l /etc/ng
inx/ssl/"
total 12
-rw------- 1 root root 1679 Jan 27 04:06 frontend1.key
-rw-r--r-- 1 root root 1294 Jan 27 04:06 nginx.crt
-rw------- 1 root root 1704 Jan 27 04:06 nginx.key
```

4.4 Website Deployment & Onboarding

Websites are deployed using Ansible playbooks. New customer sites can be added automatically using variables.

- **project6_part4_add_website_playbook.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ ansible-playbook -i temp_inventory.ini playbooks/add-websit
e.yml \
  -e "site_name=customer3 server_name=customer3.local doc_root=/var/www/customer3" \
  --private-key=./nayab-key.pem
changed: [backend2]

TASK [Create Nginx configuration] ***********************************************************************************************************
changed: [backend1]
changed: [frontend1]
changed: [frontend2]
changed: [backend2]

TASK [Reload Nginx] *************************************************************************************************************************
changed: [frontend2]
changed: [backend1]
changed: [frontend1]
changed: [backend2]

PLAY RECAP **********************************************************************************************************************************
backend1                   : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
backend2                   : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
frontend1                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
frontend2                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

- **project6_part4_new_site_added.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ ansible-playbook -i temp_inventory.ini playbooks/add-website.yml \
  -e "site_name=customer3 server_name=customer3.local doc_root=/var/www/customer3" \
  --private-key=./nayab-key.pem
changed: [backend2]

TASK [Create Nginx configuration] ***********************************************************************************************************************
changed: [backend1]
changed: [frontend1]
changed: [frontend2]
changed: [backend2]

TASK [Reload Nginx] *************************************************************************************************************************************
changed: [frontend2]
changed: [backend1]
changed: [frontend1]
changed: [backend2]

PLAY RECAP **********************************************************************************************************************************************
backend1                   : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
backend2                   : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
frontend1                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
frontend2                  : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Part 5: Monitoring & Customer Onboarding

5.1 Monitoring Scripts

A bash-based monitoring script checks HTTP status and response time for each site and logs results periodically via cron.

- **project6_part5_monitoring_script.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ cat roles/monitoring/files/check-site.sh
#!/bin/bash
# Usage: ./check-site.sh https://customer1.local
SITE=$1
LOG_FILE="/var/log/site_checks.log"
TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

if [ -z "$SITE" ]; then
    echo "Usage: $0 <url>"
    exit 1
fi

# Requirement: Check status code and measure response time
# -w provides the format: [http_code] [time_total]
RESULT=$(curl -k -s -o /dev/null -w "%{http_code} %{time_total}" "$SITE")

echo "[$TIMESTAMP] SITE: $SITE | STATUS/TIME: $RESULT" >> "$LOG_FILE"
```

- **project6_part5_monitoring_log_output.png**

```
@SafaJahangir09 ➜/workspaces/CC_Project/Project-6-High-Perf-Web-Hosting/ansible (main) $ ssh -i ./nayab-key.pem ubuntu@158.252.78.189 "sudo tail -n 10 /var/log/site_checks.log"
[2026-01-27 10:00:01] SITE: https://localhost | STATUS/TIME: 200 0.009993
[2026-01-27 10:05:01] SITE: https://localhost | STATUS/TIME: 200 0.005032
[2026-01-27 10:10:01] SITE: https://localhost | STATUS/TIME: 200 0.005335
[2026-01-27 10:15:01] SITE: https://localhost | STATUS/TIME: 200 0.005214
[2026-01-27 10:20:01] SITE: https://localhost | STATUS/TIME: 200 0.012576
[2026-01-27 10:25:01] SITE: https://localhost | STATUS/TIME: 200 0.005325
[2026-01-27 10:30:01] SITE: https://localhost | STATUS/TIME: 200 0.012079
[2026-01-27 10:35:01] SITE: https://localhost | STATUS/TIME: 200 0.011712
[2026-01-27 10:40:01] SITE: https://localhost | STATUS/TIME: 200 0.009108
[2026-01-27 10:45:01] SITE: https://localhost | STATUS/TIME: 200 0.005572
```

5.2 Customer Onboarding Documentation

The onboarding document explains required inputs, Ansible commands, verification steps, and directory conventions.

- **project6_part5_customer_onboarding_doc.png**

```
@SafaJahangir09 ➜ /workspaces/CC_Project/Project-6-High-Perf-Web-Hosting (main) $ cat docs/customer-onboarding.md
# Project 6: Customer Onboarding Procedure

## 1. Overview
This document outlines the steps to add a new customer website to the High-Performance Web Hosting cluster.

## 2. Prerequisites
- Site Name (e.g., customer3)
- Domain/Server Name (e.g., customer3.local)
- Document Root path (e.g., /var/www/customer3)

## 3. Automation Command
Run the following Ansible command from the management workstation:
```bash
ansible-playbook playbooks/add-website.yml \
  -e "site_name=customer3 server_name=customer3.local doc_root=/var/www/customer3"
```

4. Verification
1. Check Nginx status: `sudo systemctl status nginx`
2. Verify via Load Balancer IP: `curl -H "Host: customer3.local" http://<LB_IP>`
3. Confirm logging: Check `/var/log/nginx/access.log` on backend servers.
@SafaJahangir09 ➜ /workspaces/CC_Project/Project-6-High-Perf-Web-Hosting (main) $
```

Conclusion

This project demonstrates a production-ready, high-performance web hosting platform using Terraform and Ansible. All requirements were implemented within course scope, including high availability, SSL, caching, automation, and monitoring.