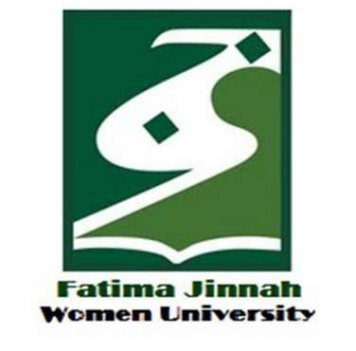
CLOUD COMPUTING
PROJECT



SUBMITTED TO

SIR WAQAS SALEEM

SUBMITTED BY

EMAAN HANIF 2023-BSE-017
HAMNA MAHMOOD 2023-BSE-025
KOMAL KASHIF 2023-BSE-031

BSE V-A

TABLE OF CONTENTS

Project 7 – Automated Development and Testing Environment

1. Executive Summary

This project focuses on building an automated development and testing environment using Infrastructure as Code and configuration management principles. Terraform was used to provision AWS infrastructure in a modular and reusable manner, while Ansible was used to configure web servers and application stacks consistently across environments. Multiple isolated environments (dev, qa, uat) were implemented to support parallel development and testing without resource conflicts. Automation scripts enabled self-service provisioning, health validation, and cost-efficient teardown of environments. Overall, the project demonstrates practical application of DevOps practices for scalable, repeatable, and reliable environment management.

2. Architecture Design

- project7_part1_repository_structure.png

- project7_part1_initial_commit.png

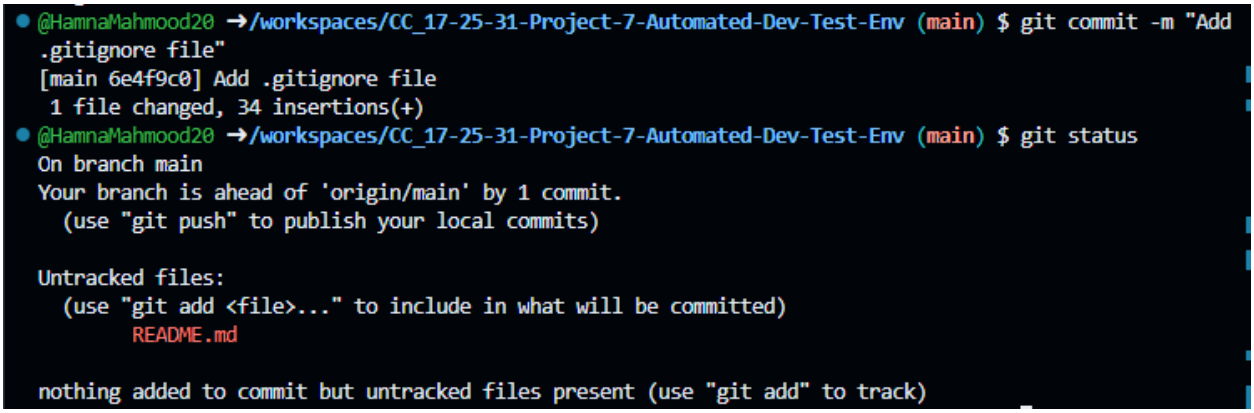
```
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Project7/
        README.md

nothing added to commit but untracked files present (use "git add" to track)
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git add Project7/
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git commit -m "Init
ial commit: Project7 repository structure with Terraform and Ansible files"
[main 03227e6] Initial commit: Project7 repository structure with Terraform and Ansible files
 42 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Project7/.gitignore
 create mode 100644 Project7/README.md
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git log --oneline
03227e6 (HEAD -> main) Initial commit: Project7 repository structure with Terraform and Ansible files
cf84fb7 (origin/main, origin/HEAD) Initial commit
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git push origin mai
n
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 2 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (23/23), 2.03 KiB | 1.01 MiB/s, done.
Total 23 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/HamnaMahmood20/CC_17-25-31-Project-7-Automated-Dev-Test-Env
   cf84fb7..03227e6  main -> main
```

1.2 .gitignore & Basic Hygiene

- project7_part1_gitignore_content.png

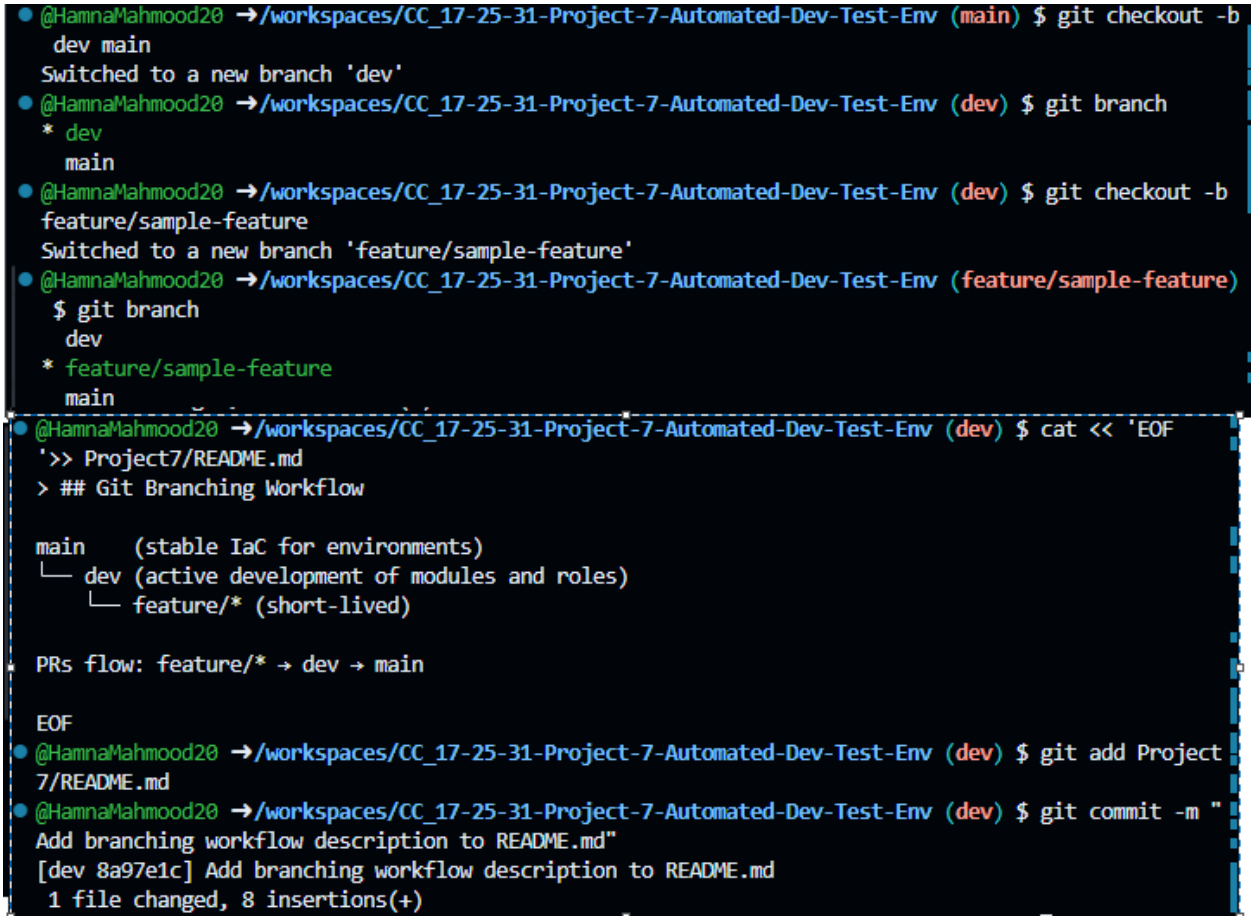
```
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ cat Project7/.gitig
nore
# Terraform
**/.terraform/*
*.tfstate
*.tfstate.*
*.tfvars
!*.tfvars.example

# Ansible
*.retry
.vault_pass
*.secret

# AWS credentials
.aws/
*.pem
*.key

# IDE / OS / Logs / Temp
.vscode/
.idea/
*.swp
*.swo
*~
.DS_Store
Thumbs.db
*.log
logs/
.env
```

- project7_part1_git_status_clean.png



1.3 Branching & Simple Workflow

- project7_part1_git_branches.png



PART II - Terraform Infrastructure Modules

Module Path: terraform/modules/network/

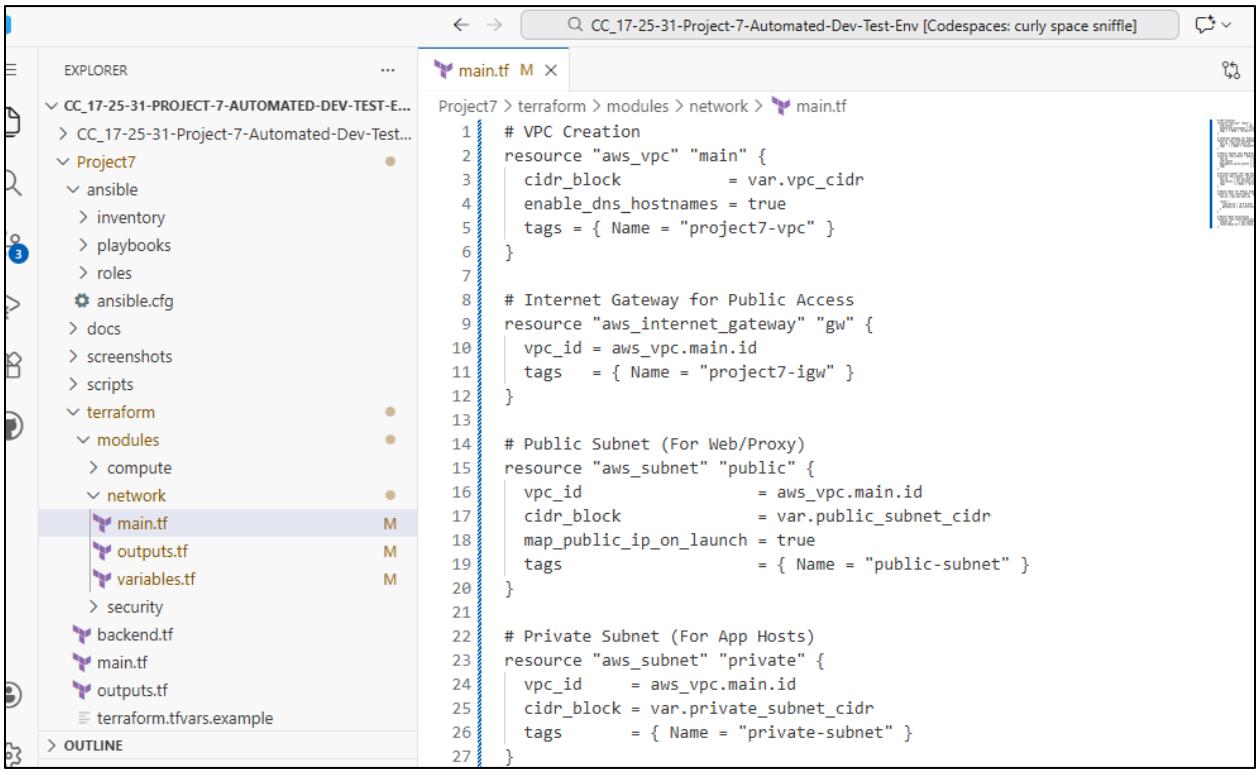
The primary objective of this module is to provision a secure, isolated, and scalable network foundation on AWS to support the multi-tier automated development environment. It follows the principle of segregation to ensure that public-facing services and private application logic are isolated.

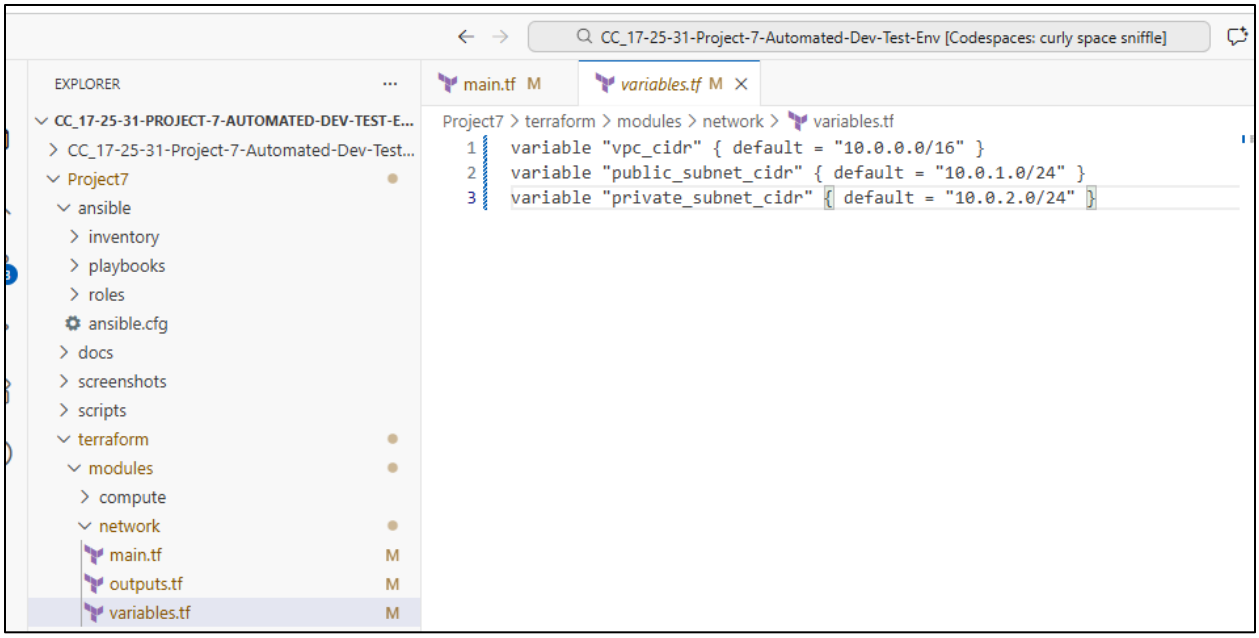
Infrastructure Components Implemented

- **VPC (Virtual Private Cloud):** A dedicated virtual network created with a CIDR block of 10.0.0.0/16 to provide ample IP address space for multiple environments (Dev, QA, UAT).
- **Public Subnet:** Designed for the Nginx reverse proxy and external-facing web servers. It is configured to assign public IP addresses automatically to instances launched within it.
- **Private Subnet:** Reserved for application hosts (PHP-FPM) that do not require direct internet exposure, enhancing the security posture of the environment.
- **Internet Gateway (IGW):** Deployed and attached to the VPC to enable communication between resources in the public subnet and the internet.
- **Public Route Table:** A custom routing table configured to direct outbound traffic (0.0.0.0/0) through the Internet Gateway. This table is explicitly associated with the public subnet.

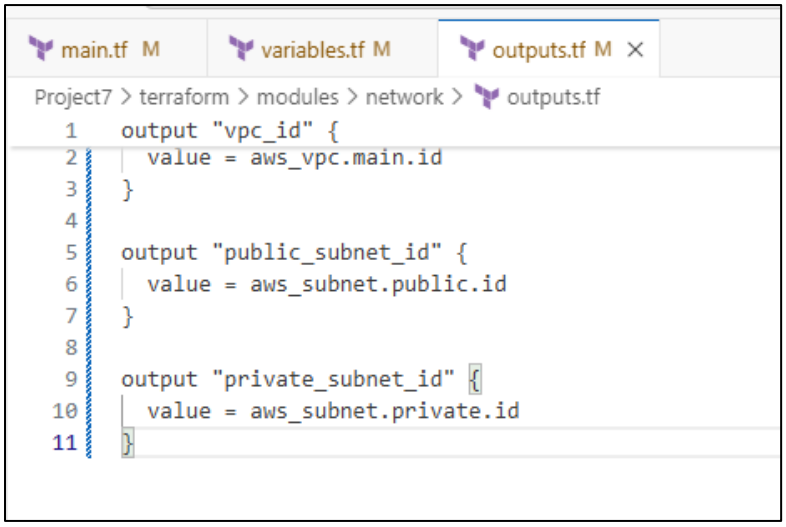
2.1 Network Module – VPC, Subnets, Routing

- project7_part2_network_main.png

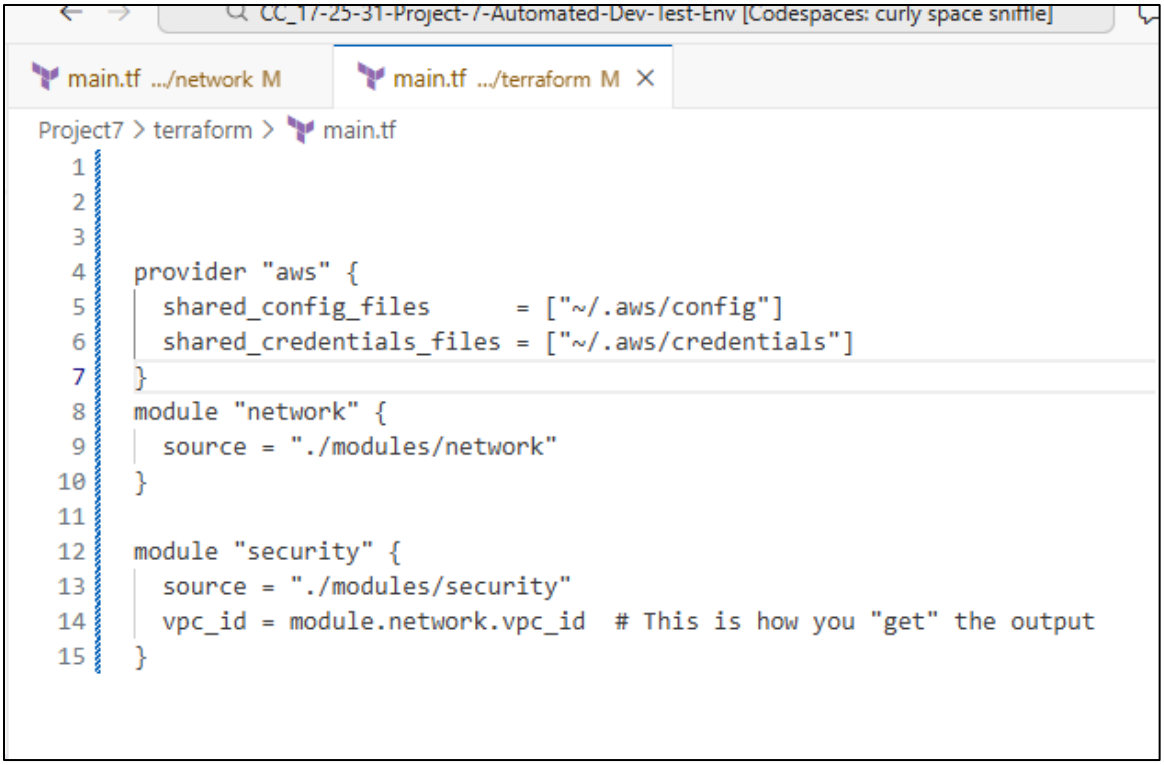




- project7_part2_network_outputs.png

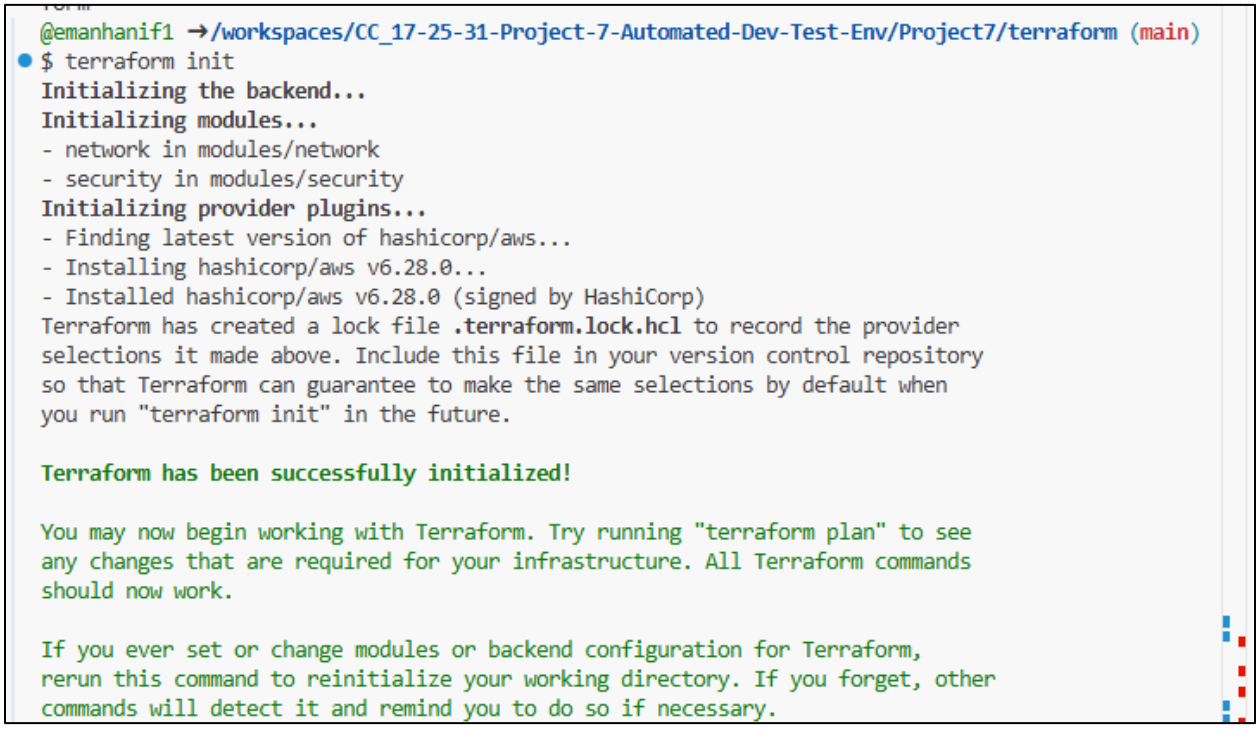


Call in main



Initialize the directory

terraform init

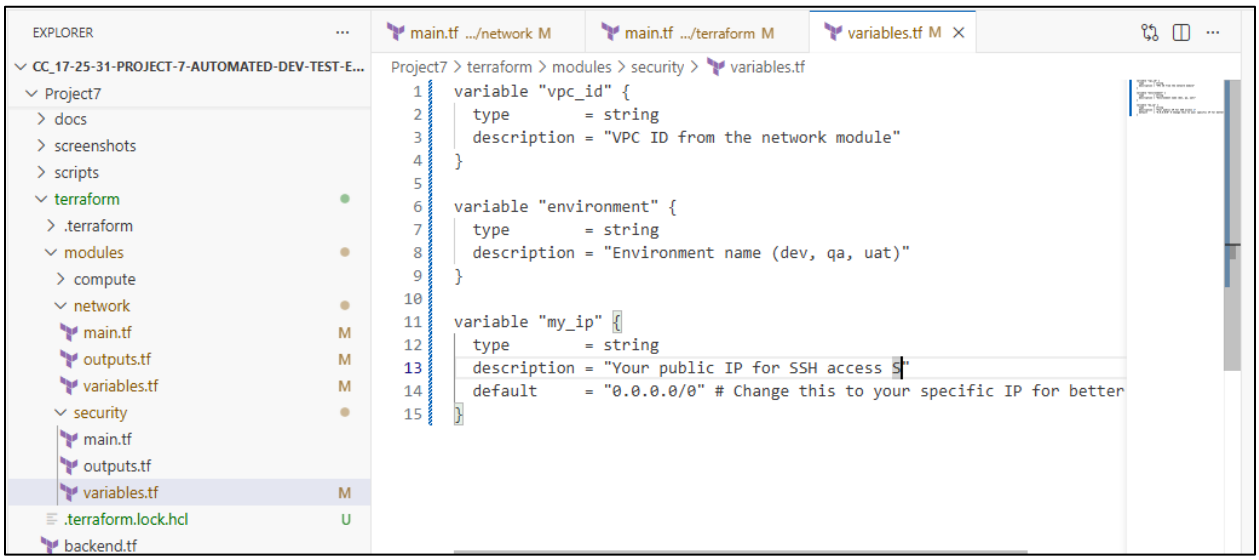


Output

1. **VPC ID:** Required for the creation of Security Groups in Part 2.2.
2. **Public Subnet ID:** Required for deploying the Nginx proxy and Apache web stacks.
3. **Private Subnet ID:** Required for deploying backend app instances.

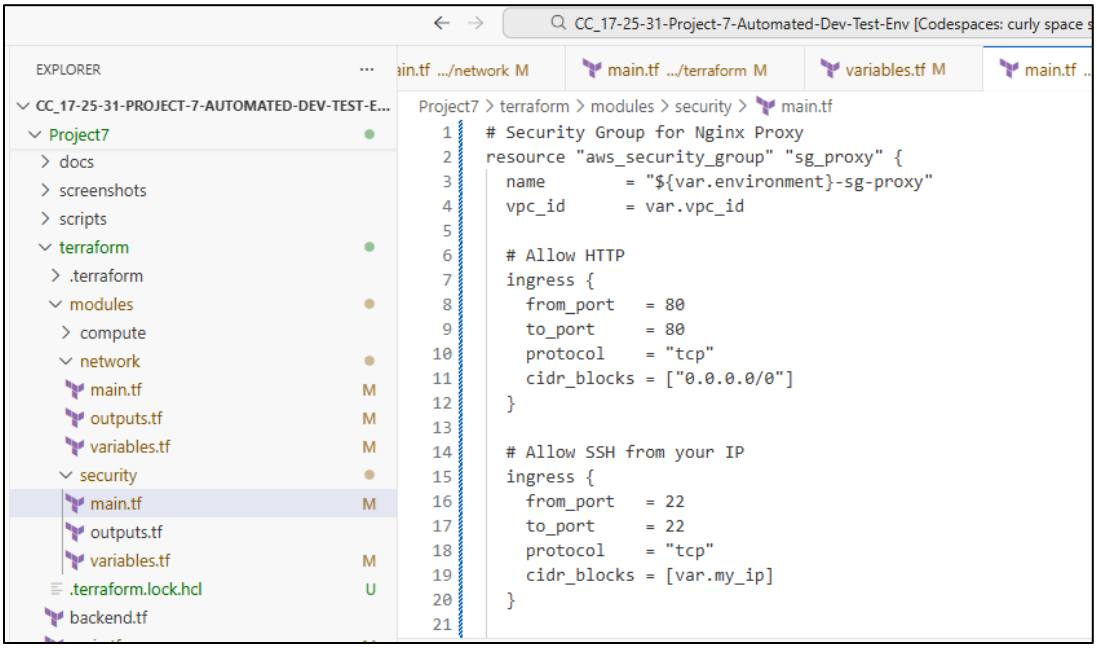
```
private_subnet_id = "subnet-0e9bd14962c5d881d"
public_subnet_id = "subnet-0b6c79a37f3acae46"
vpc_id = "vpc-0b330b9636fa90c7b"
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
```

2.2 - Security Module – Security Groups



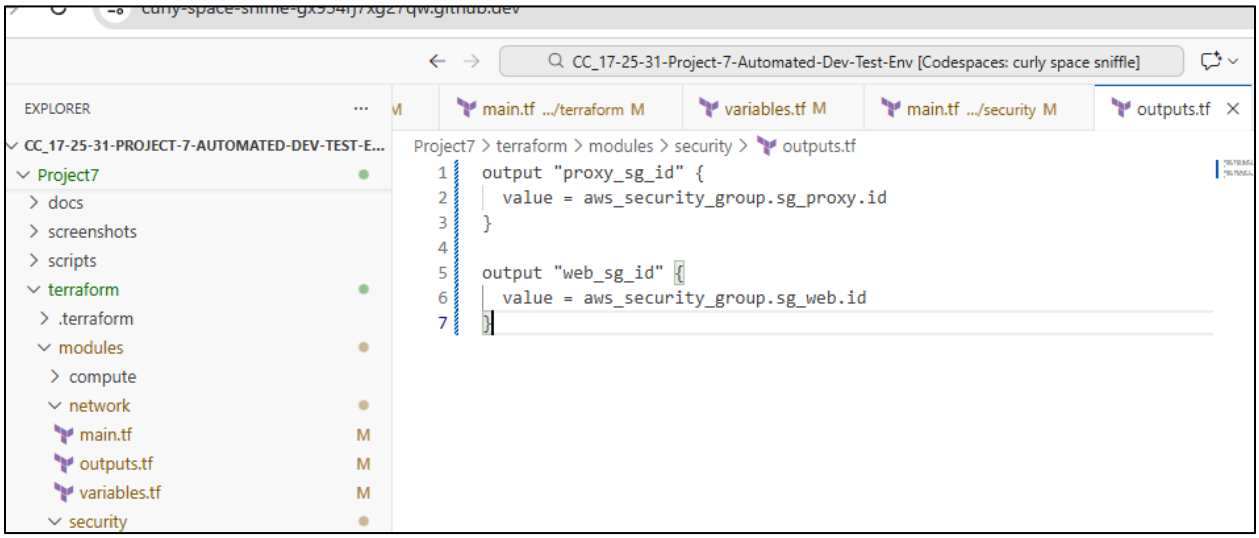
Security Module Main Code (main.tf)

Creating the Proxy and Web security groups with the required tags,



Security Module Outputs (outputs.tf)

Open Project7/terraform/modules/security/outputs.tf. These are required for the Compute (EC2) module:

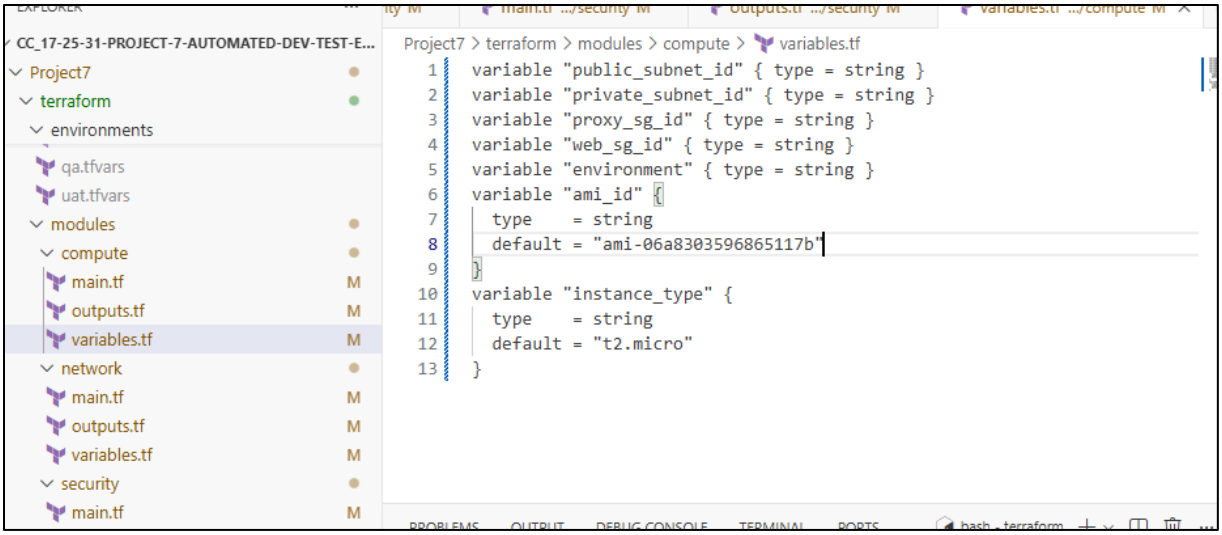


- **sg_proxy**: Configured to allow inbound HTTP (Port 80) and HTTPS (Port 443) traffic from the public internet. SSH access is restricted to the administrator's specific IP address for management.
- **sg_web**: Highly secure group that allows HTTP traffic **only** if it originates from the Proxy Security Group. This ensures that the web servers are not directly exposed to the public internet, satisfying the project's isolation requirements.

2.3 Compute Module – Environment Instances

Compute Module Main Code (variables.tf)

This is terraform/modules/compute/ **variables**.tf. This code provisions the three required instances with their specific tags,



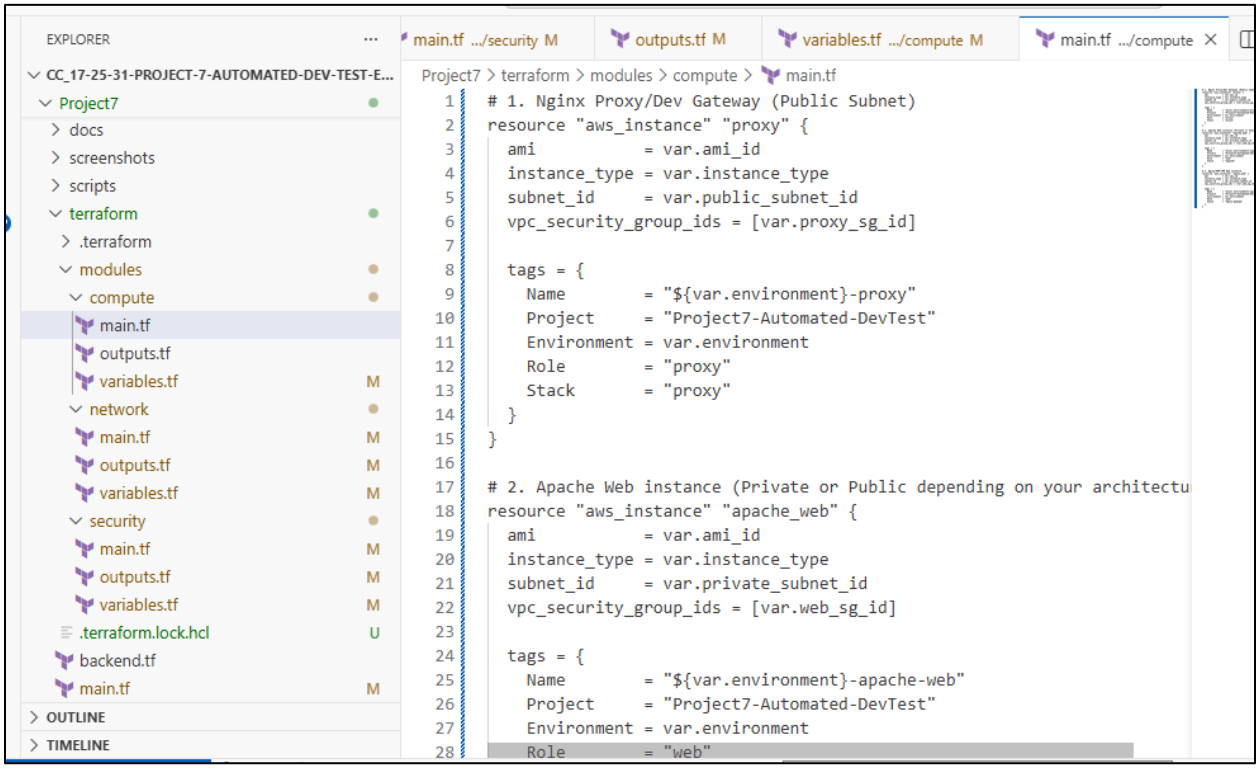
OUTPUTS

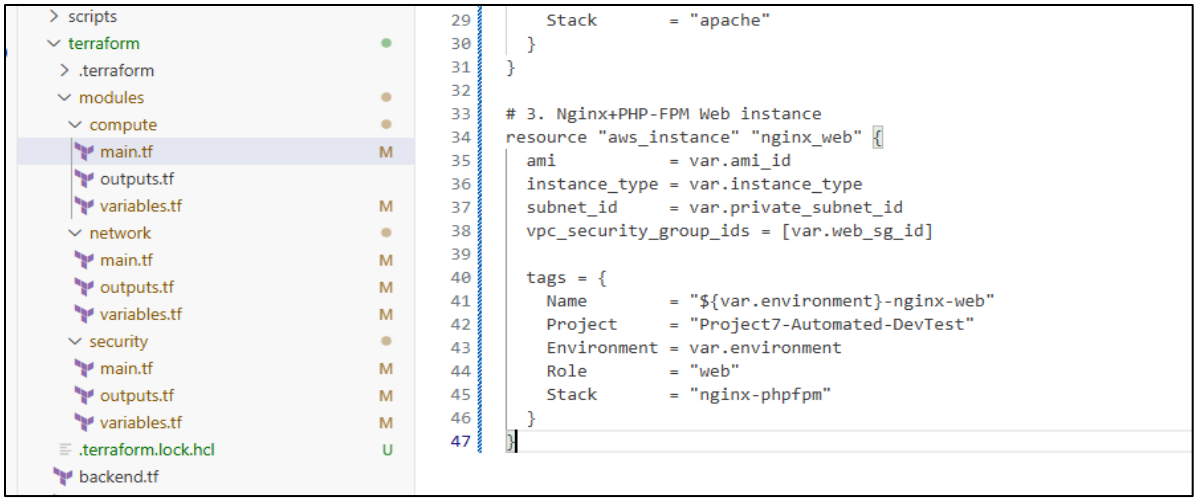
SG IDs for compute module,

```
proxy_security_group_id = "sg-087d98fe4c537f95f"
public_subnet_id = "subnet-0b6c79a37f3acae46"
vpc_id = "vpc-0b330b9636fa90c7b"
web_security_group_id = "sg-08e5482e3d8d6b697"
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$
```

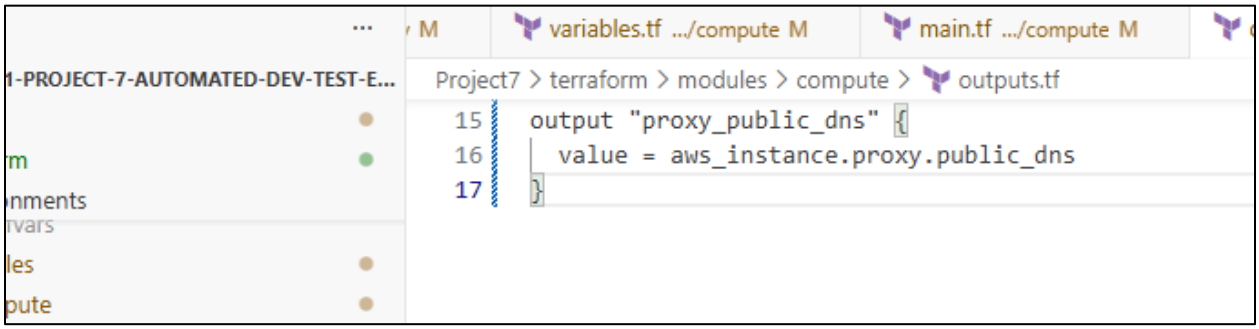
Compute Module Main Code (main.tf)

terraform/modules/compute/main.tf provisions the three required instances with their specific tags,

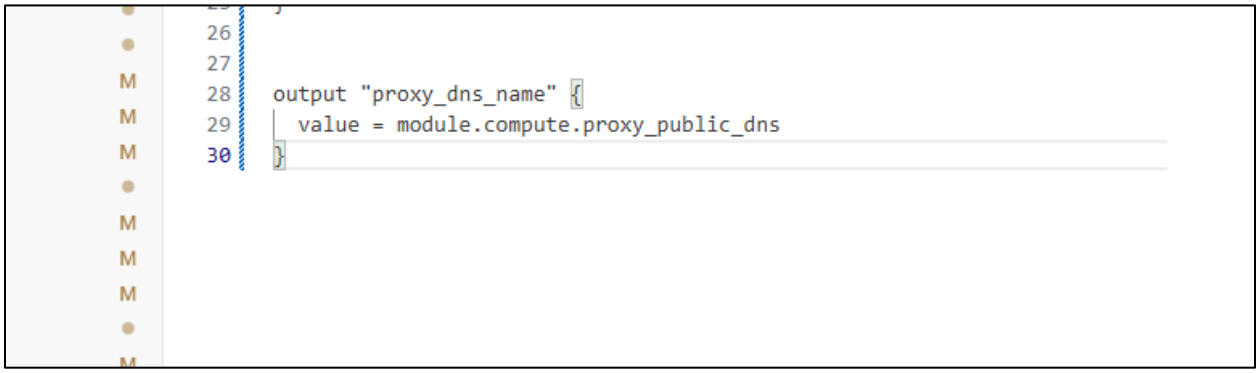




Specifically need the **Public DNS** (e.g., ec2-3-29-134-133.me-central-1.compute.amazonaws.com), you can add this to your **Compute Module's** outputs.tf first:



root outputs.tf,



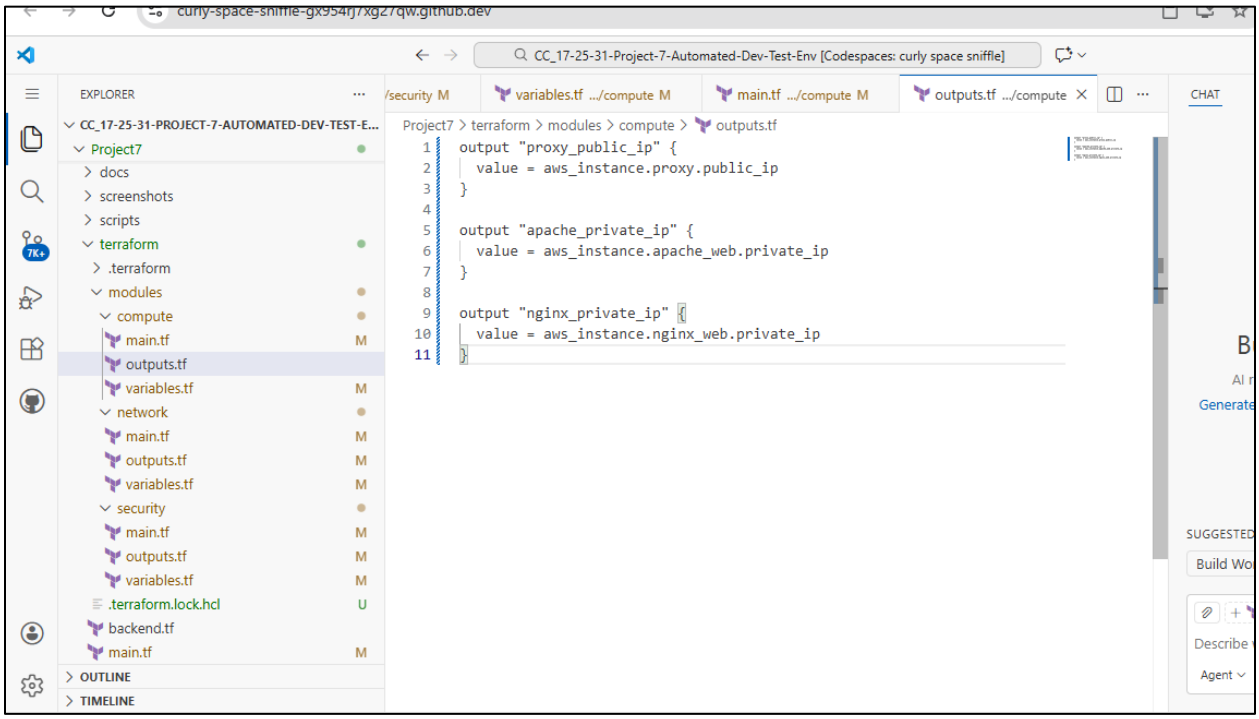
OUTPUT

- Public IP/DNS of proxy
- Private IPs of web instances

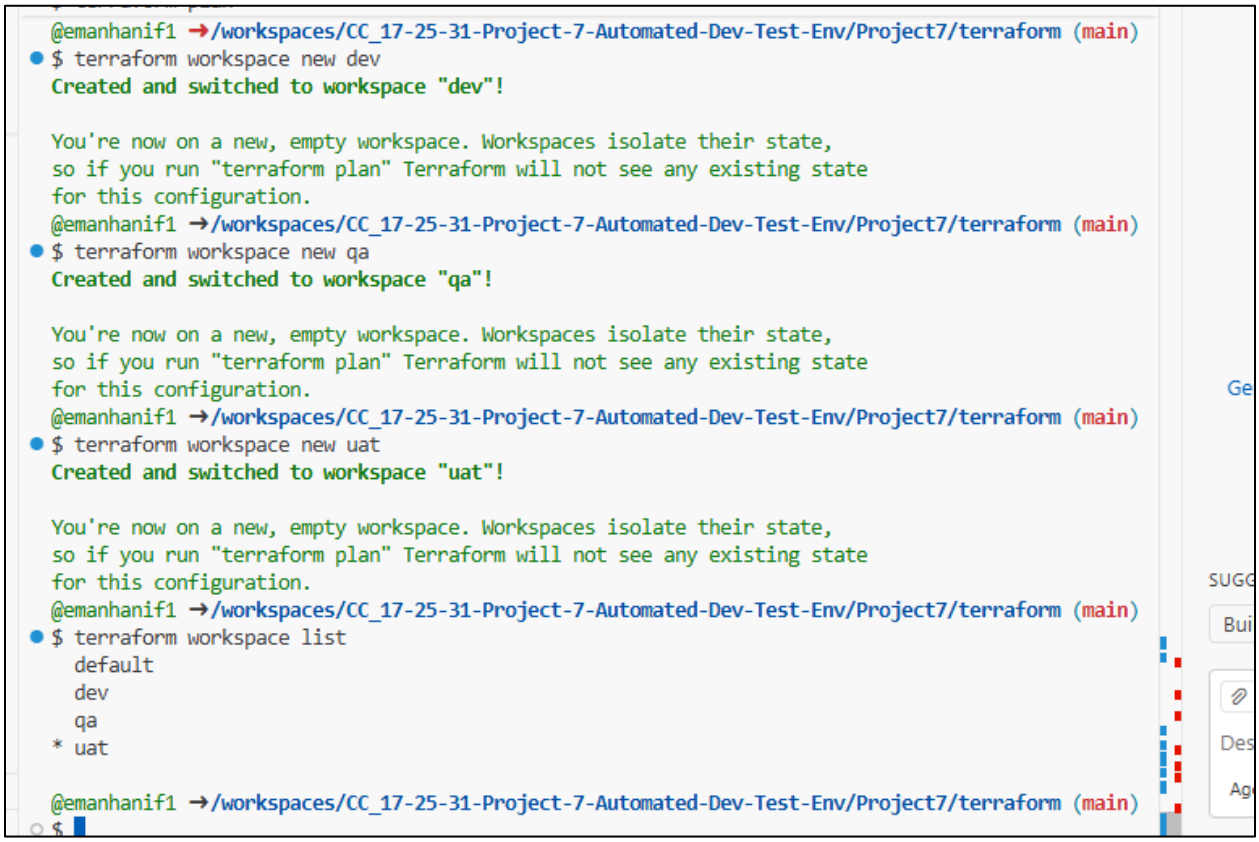
```
env_apache_private_ip = "10.0.2.107"
env_nginx_private_ip = "10.0.2.232"
env_proxy_public_ip = "3.29.134.133"
private_subnet_id = "subnet-0e9bd14962c5d881d"
proxy_dns_name = "ec2-3-29-134-133.me-central-1.compute.amazonaws.com"
```

Compute Module Outputs (outputs.tf)

terraform/modules/compute/outputs.tf, these are essential for Ansible to know which IPs to configure:



PART III - State Isolation & Environments



3.1 State Isolation

Method Chosen: Option A – Terraform Workspaces

To manage the deployment of the **Automated Dev-Test Environment**, we implemented **Terraform Workspaces**. This approach ensures that the state of each environment (Dev, QA, and UAT) is stored separately within the terraform.tfstate.d directory.

Key Benefits of Isolation:

- **Conflict Prevention:** Resources in the dev workspace (like VPCs or EC2 instances) are tracked independently from those in qa or uat, allowing us to create, modify, or destroy one environment without affecting the others.
- **Resource Tagging:** By using the ${terraform.workspace} variable in our code, we automatically tag resources with the correct environment name.
- **Simplified Workflow:** Developers can switch between environments using a single command (terraform workspace select), making the infrastructure management highly efficient.

3.2 Environment-Specific Variables

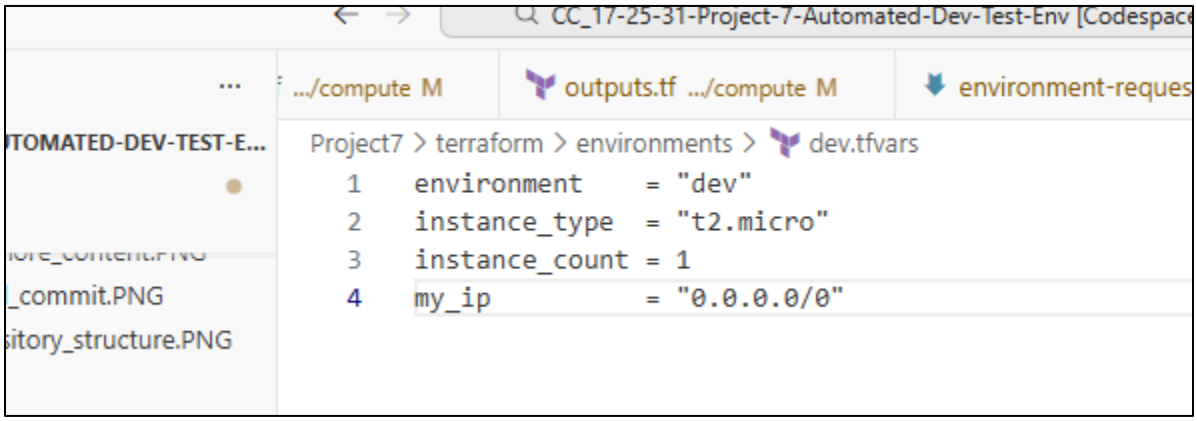
Create the Directory and Files,

```
bash: cd: Project7/terraform: No such file or directory
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ mkdir -p environments
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ touch environments/dev.tfvars environments/qa.tfvars environments/uat.tfvars
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$
```

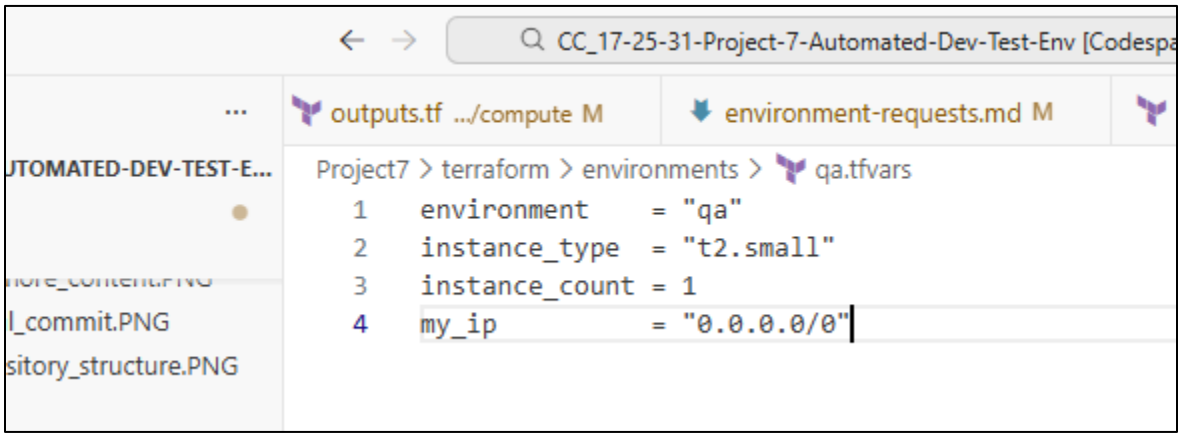
Populate the Files,

Now, open each file in the Codespace editor and paste the following content to match the project requirements:

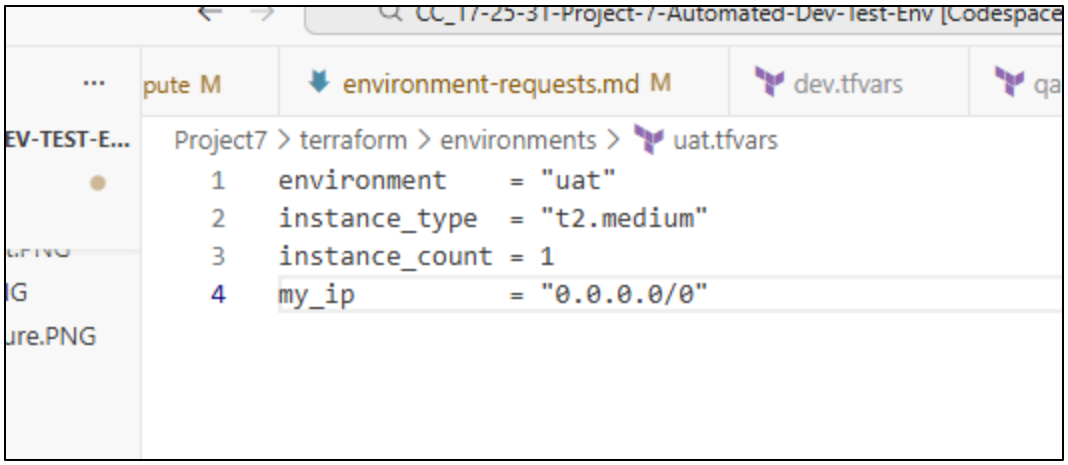
environments/dev.tfvars,



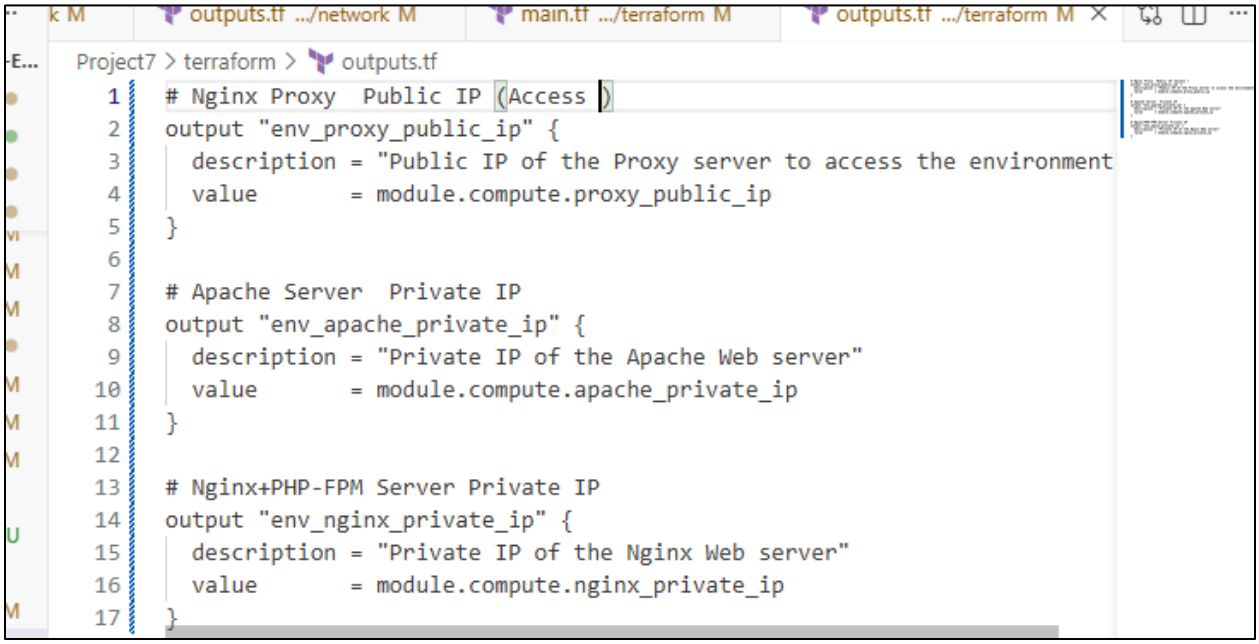
environments/qa.tfvars,



environments/uat.tfvars,

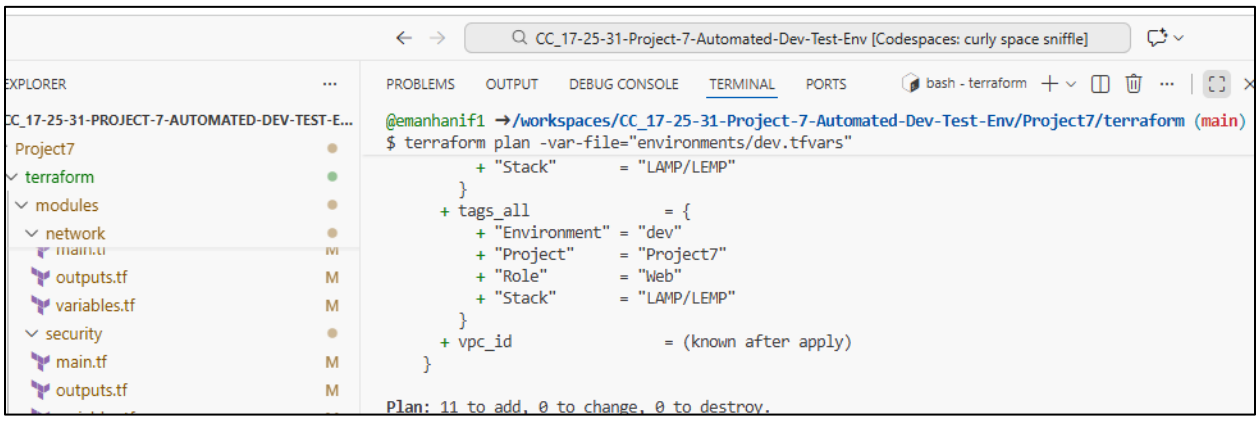


Root outputs,

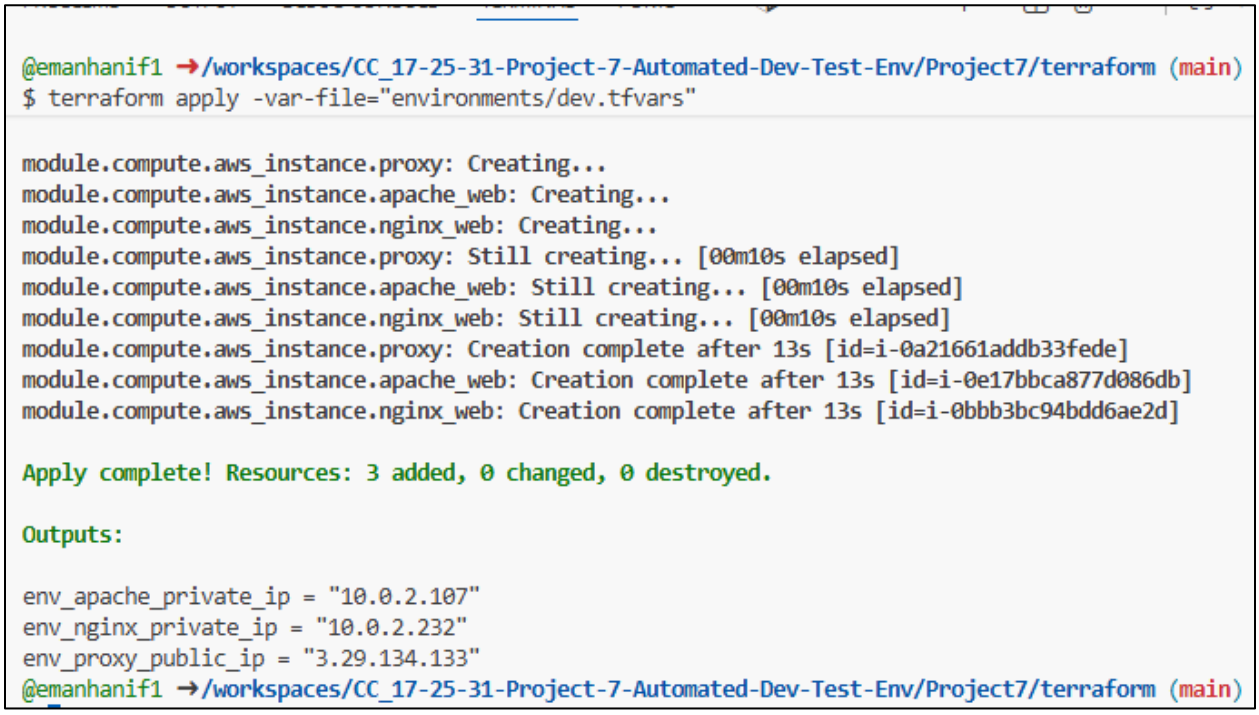


Running this command,

terraform plan -var-file="environments/dev.tfvars"



OUTPUTS



PART IV - Ansible Roles (Apache + Nginx + PHP-FPM + Proxy)

4.1 Base Tools & Inventory

- project7_part4_inventory_graph.png

```
@HamnaMahmood20 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/ansible
(main) $ ansible-inventory -i inventory/aws_ec2.yml --graph
This feature will be removed from ansible-core version 2.24. Use ansible.module_utils.common
.text.converters instead.
[DEPRECATION WARNING]: Importing 'to_native' from 'ansible.module_utils._text' is deprecated
. This feature will be removed from ansible-core version 2.24. Use ansible.module_utils.comm
on.text.converters instead.
[DEPRECATION WARNING]: Passing `disable_lookups` to `template` is deprecated. This feature w
ill be removed from ansible-core version 2.23.
@all:
  |--@ungrouped:
  |--@aws_ec2:
  |  |--ip-10-0-2-123.me-central-1.compute.internal
  |  |--ec2-51-112-253-222.me-central-1.compute.amazonaws.com
  |  |--ip-10-0-2-44.me-central-1.compute.internal
  |--@env_dev:
  |  |--ip-10-0-2-123.me-central-1.compute.internal
  |  |--ec2-51-112-253-222.me-central-1.compute.amazonaws.com
  |  |--ip-10-0-2-44.me-central-1.compute.internal
  |--@role_web:
  |  |--ip-10-0-2-123.me-central-1.compute.internal
  |  |--ip-10-0-2-44.me-central-1.compute.internal
  |--@stack_nginx_phpfpm:
  |  |--ip-10-0-2-123.me-central-1.compute.internal
  |--@role_proxy:
  |  |--ec2-51-112-253-222.me-central-1.compute.amazonaws.com
  |--@stack_proxy:
  |  |--ec2-51-112-253-222.me-central-1.compute.amazonaws.com
  |--@stack_apache:
  |  |--ip-10-0-2-44.me-central-1.compute.internal
```

4.2 Apache Web Stack (HTTPD + PHP)

- project7_part4_apache_role_main.png

```
- name: Ensure Apache is enabled and started
  service:
    name: httpd
    state: started
    enabled: yes
```

4.3 Nginx+ PHP-FPM Web Stack

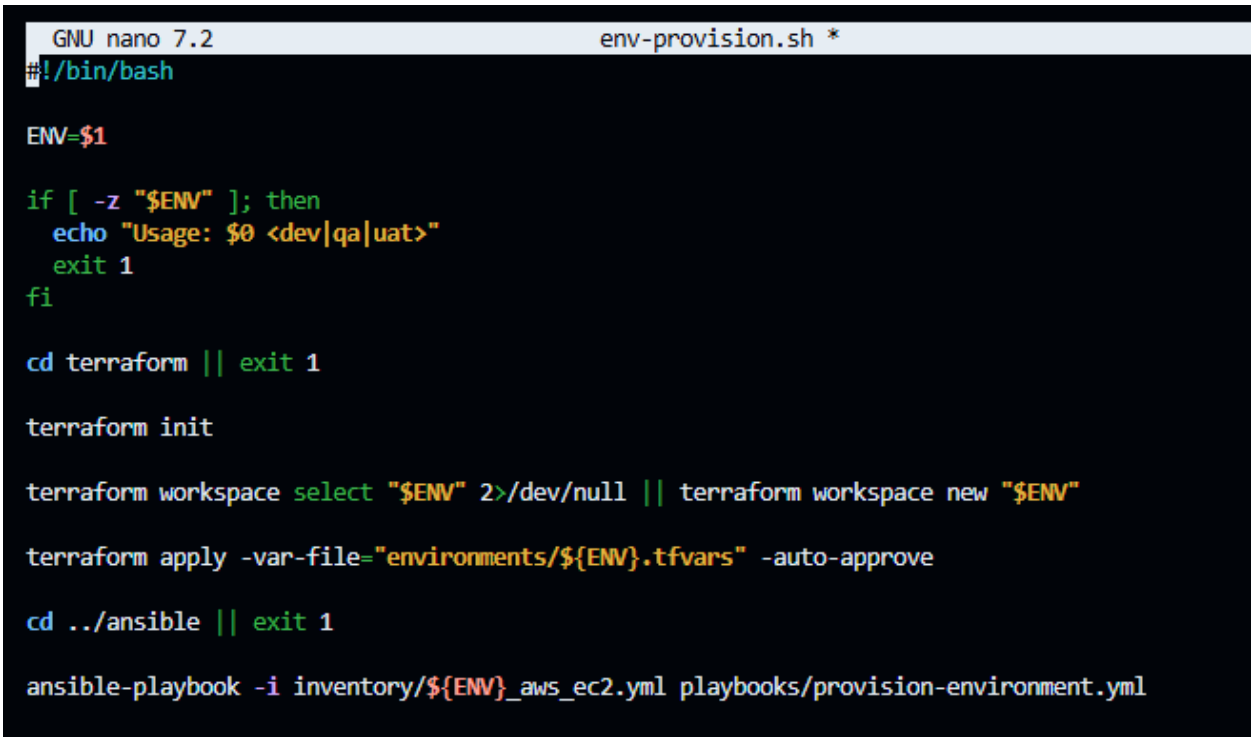
- project7_part4_nginx_phpfpm_role_main.png

```
 GNU nano 7.2              ansible/roles/nginx-phpfpm-web/handlers/main.yml
---
- name: Reload Nginx
  service:
    name: nginx
    state: reloaded
```

PART V Automation Scripts, Health Checks & Terraform

5.1 Provision Script

- project7_part5_env_provision_script.png

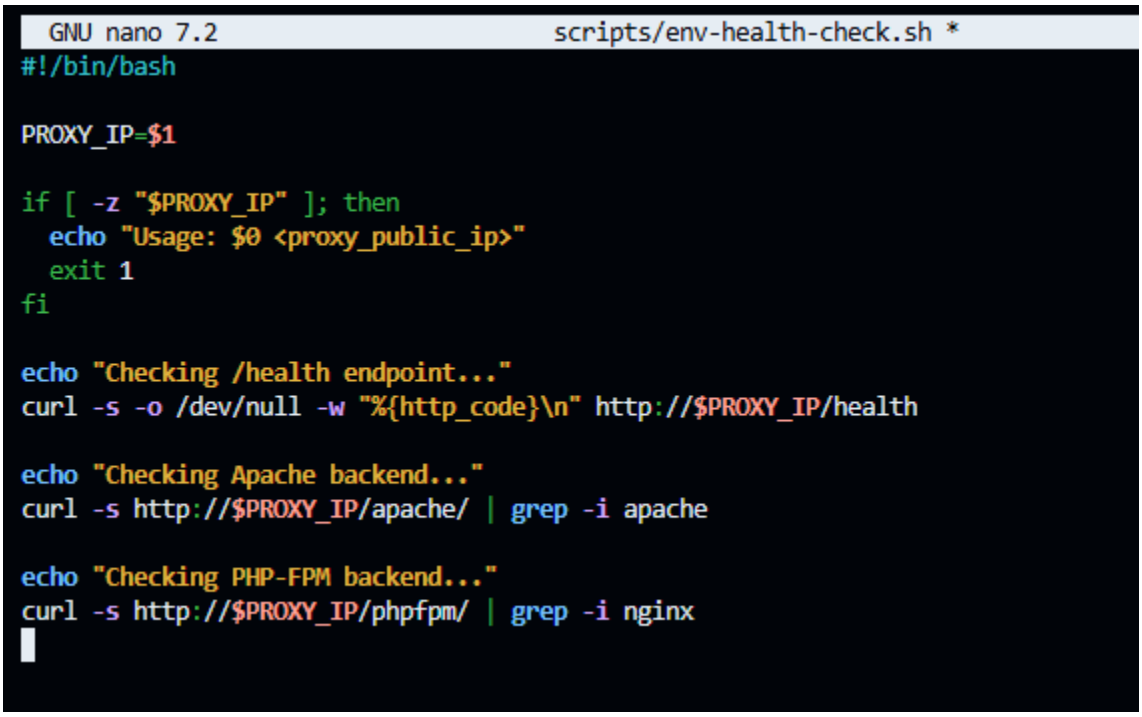


5.2 Teardown Script

- project7_part5_env_destroy_script.png

5.3 Health Checks

- project7_part5_health_check_script.png



PART VI - Teardown / Cost Optimization

These are actions that prevent unnecessary AWS spend by destroying unused resources.

In *our* practical work, Teardown / Cost Optimization includes:

1. **scripts/env-destroy.sh**
    - Explicitly destroys all infrastructure for an environment
    - Prevents idle EC2, VPC, and NAT costs
    - Primary cost-optimization mechanism

2. **Environment-Based Lifecycle**
    - Short-lived dev / qa / uat environments
    - Created only when needed
    - Destroyed after testing

```
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform apply -var-file="environments/dev.tfvars"

module.compute.aws_instance.proxy: Creating...
module.compute.aws_instance.apache_web: Creating...
module.compute.aws_instance.nginx_web: Creating...
module.compute.aws_instance.proxy: Still creating... [00m10s elapsed]
module.compute.aws_instance.apache_web: Still creating... [00m10s elapsed]
module.compute.aws_instance.nginx_web: Still creating... [00m10s elapsed]
module.compute.aws_instance.proxy: Creation complete after 13s [id=i-0a21661addb33fede]
module.compute.aws_instance.apache_web: Creation complete after 13s [id=i-0e17bbca877d086db]
module.compute.aws_instance.nginx_web: Creation complete after 13s [id=i-0bbb3bc94bdd6ae2d]

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

Outputs:

env_apache_private_ip = "10.0.2.107"
env_nginx_private_ip = "10.0.2.232"
env_proxy_public_ip = "3.29.134.133"
@emanhanif1 ➜/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
```

PART VII - Challenges & Solutions

1. Environment State Isolation

Challenge:
Using the same Terraform configuration for dev, qa, and uat risked state conflicts.

Solution:
Terraform workspaces were used to isolate state per environment, ensuring resources did not overlap.

2. Terraform Module Integration

Challenge:
Passing outputs (VPC IDs, subnet IDs, security groups) between modules caused initial misconfigurations.

Solution:
Clearly defined variables and outputs were implemented for each module with consistent naming.

3. Security Group Least Privilege

Challenge:
Restricting access while allowing required traffic between proxy and web servers.

Solution:
Separate security groups were created, allowing backend access only from the proxy and SSH only from the admin IP.

4. Nginx and PHP-FPM Configuration

Challenge:
PHP files were downloaded instead of executed due to incorrect FastCGI configuration.

Solution:
Correct fastcgi_pass and SCRIPT_FILENAME settings were applied and PHP-FPM was ensured to be running.

5. Reverse Proxy Routing

Challenge:
Routing multiple backend services through a single Nginx proxy required careful path handling.

Solution:
Dedicated Nginx location blocks were configured for each backend (/apache/, /phpfpm/).

PART VIII - Conclusion

This project successfully demonstrated the design and implementation of an automated development and testing environment using Terraform and Ansible. Reusable Terraform modules and isolated state management enabled safe provisioning of multiple environments, while Ansible roles ensured consistent, production-like configuration across all stacks. Automation scripts simplified environment provisioning, validation, and teardown, improving reliability and cost efficiency. Overall, the project reinforced key DevOps concepts such as infrastructure automation, configuration management, environment isolation, and repeatable deployment workflows