CLOUD COMPUTING
PROJECT

**Fatima Jinnah
Women University**
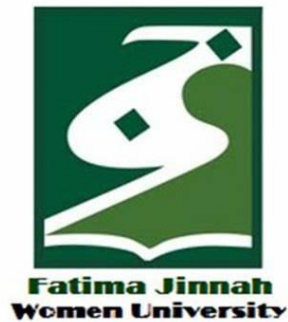
SUBMITTED TO

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

```
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git commit -m "Add
.gitignore file"
[main 6e4f9c0] Add .gitignore file
 1 file changed, 34 insertions(+)
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

nothing added to commit but untracked files present (use "git add" to track)
```

1.3 Branching & Simple Workflow

- project7_part1_git_branches.png

```
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (main) $ git checkout -b
  dev main
 Switched to a new branch 'dev'
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (dev) $ git branch
 * dev
   main
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (dev) $ git checkout -b
 feature/sample-feature
 Switched to a new branch 'feature/sample-feature'
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (feature/sample-feature)
   $ git branch
   dev
 * feature/sample-feature
   main
```

```
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (dev) $ cat << 'EOF
'>> Project7/README.md
> ## Git Branching Workflow

 main    (stable IaC for environments)
 └── dev (active development of modules and roles)
     └── feature/* (short-lived)

 PRs flow: feature/* → dev → main

 EOF
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (dev) $ git add Project
7/README.md
@HamnaMahmood20 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env (dev) $ git commit -m "
Add branching workflow description to README.md"
[dev 8a97e1c] Add branching workflow description to README.md
 1 file changed, 8 insertions(+)
```

PART II - Terraform Infrastructure Modules
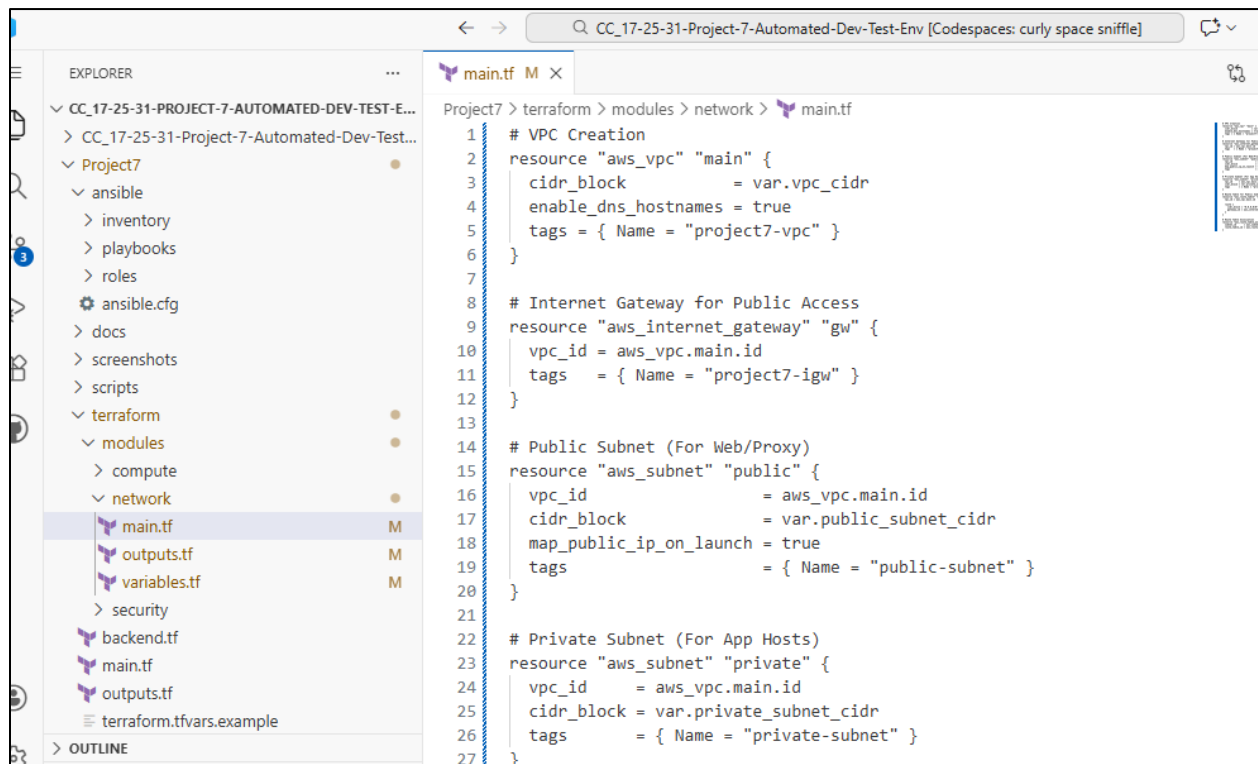
Module Path: terraform/modules/network/

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.
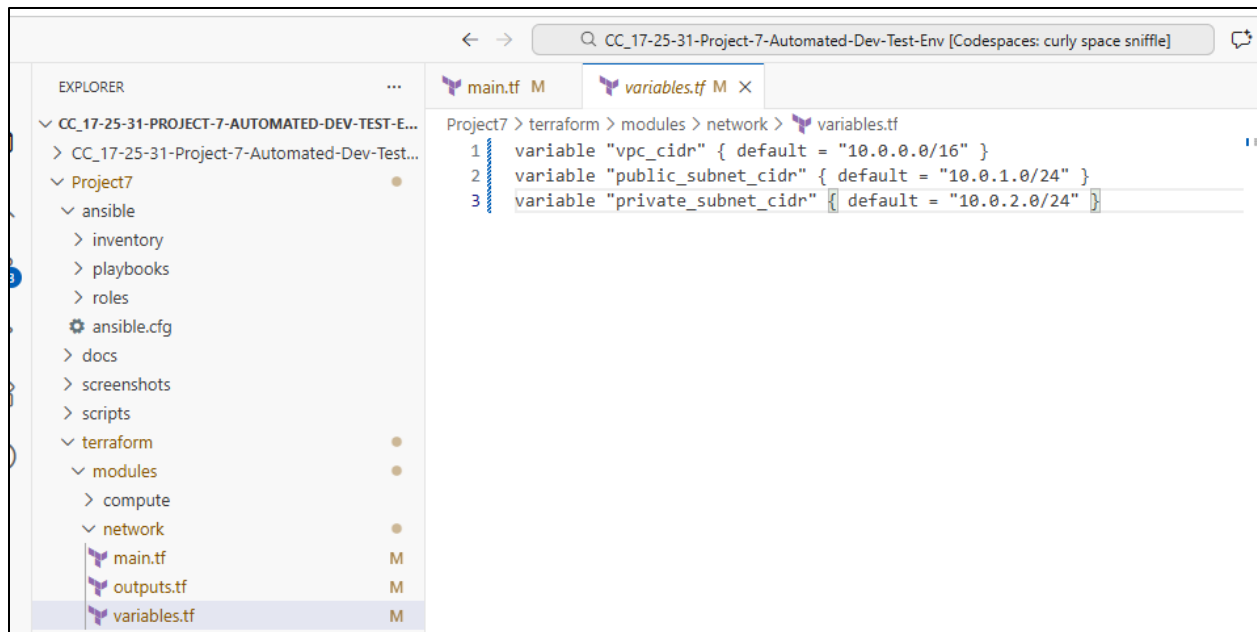
Infrastructure Components Implemented

- **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.
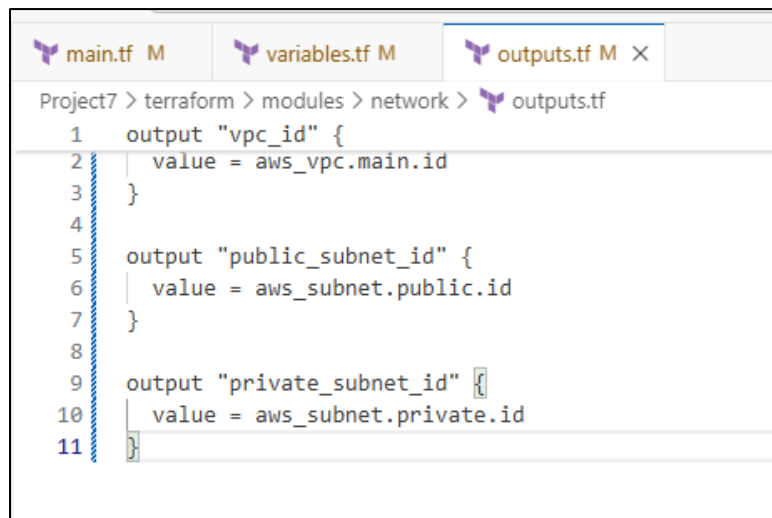
2.1 Network Module – VPC, Subnets, Routing

- project7_part2_network_main.png

- project7_part2_network_outputs.png

Call in main

🇹 main.tf .../network M   🇹 main.tf .../terraform M ✕

Project7 > terraform > 🇹 main.tf

```
 1
 2
 3
 4     provider "aws" {
 5         shared_config_files      = ["~/.aws/config"]
 6         shared_credentials_files = ["~/.aws/credentials"]
 7     }
 8     module "network" {
 9         source = "./modules/network"
10     }
11
12     module "security" {
13         source = "./modules/security"
14         vpc_id = module.network.vpc_id  # This is how you "get" the output
15     }
```

Initialize the directory

terraform init

```
@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform init
Initializing the backend...
Initializing modules...
- network in modules/network
- security in modules/security
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```
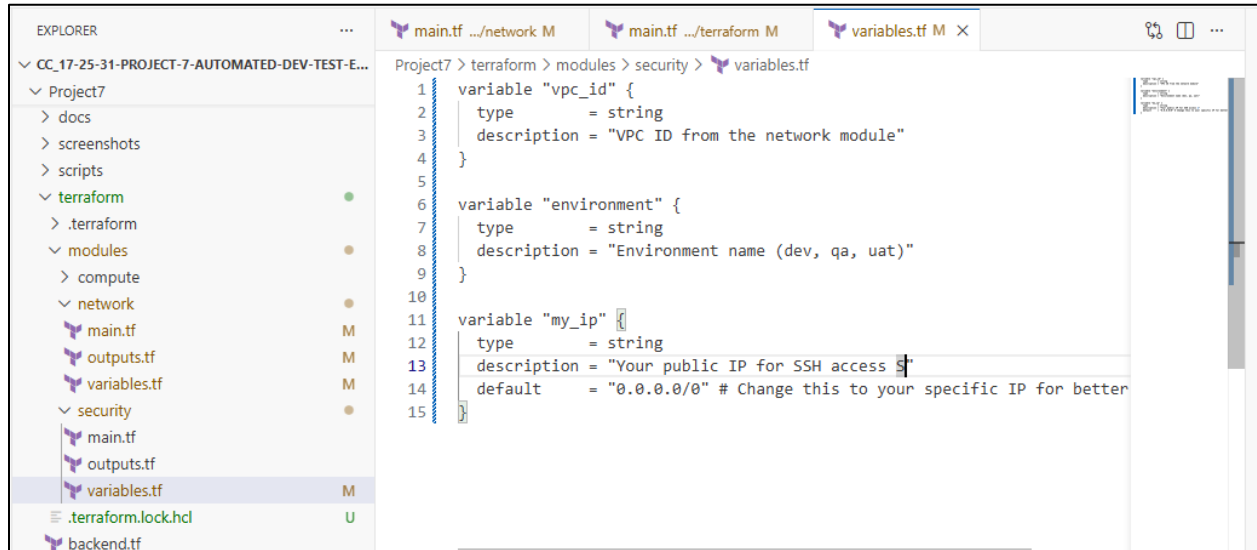
Output

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)
```
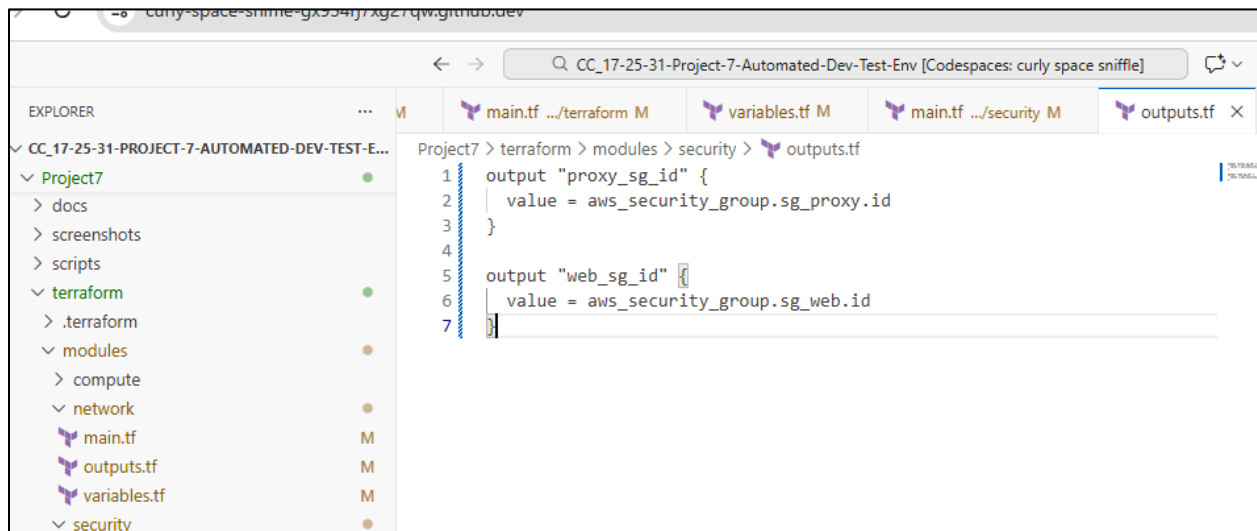
2.2 - Security Module – Security Groups

Security Module Main Code (main.tf)

Creating the Proxy and Web security groups with the required tags,
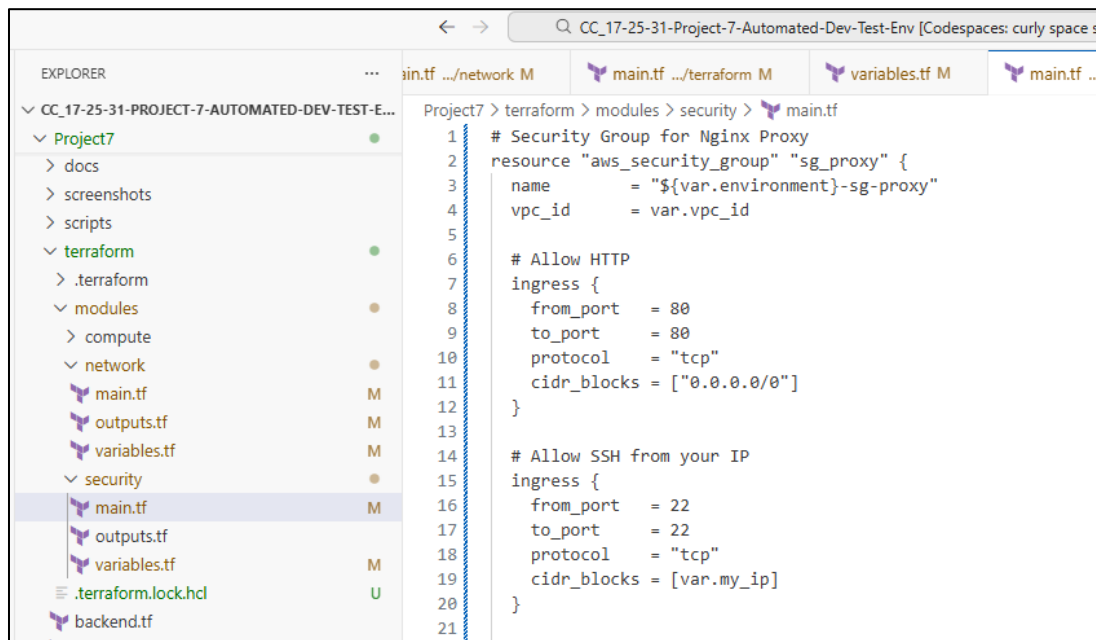
Security Module Outputs (outputs.tf)

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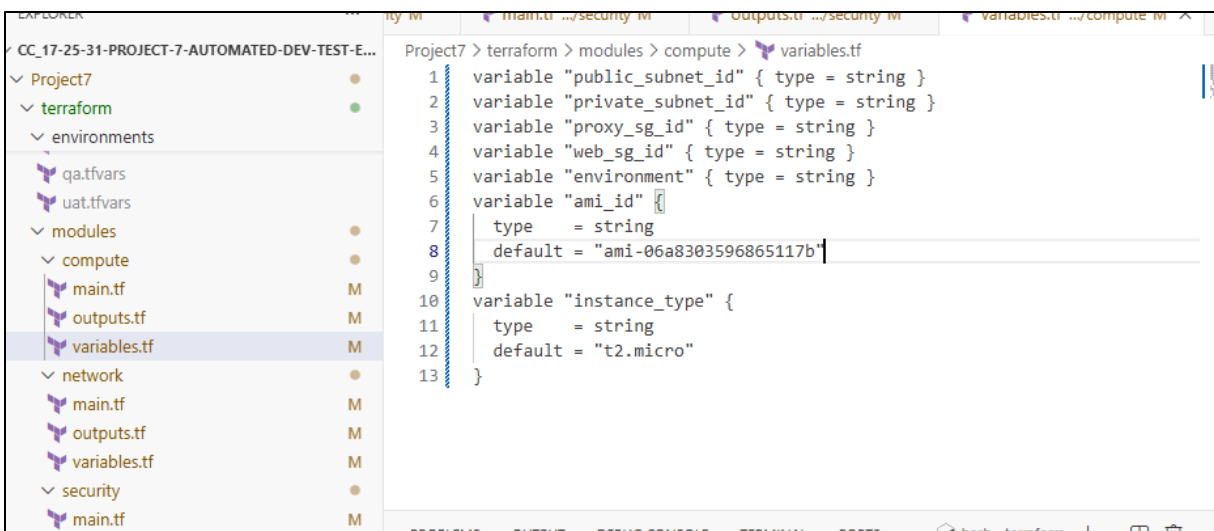

- **sg_proxy**: Configured to allow inbound HTTP (Port 80) and HTTPS (Port 443) traffic from the public internet. SSH access is restricted to the administrator's specific IP address for management.
- **sg_web**: Highly secure group that allows HTTP traffic **only** if it originates from the Proxy Security 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)

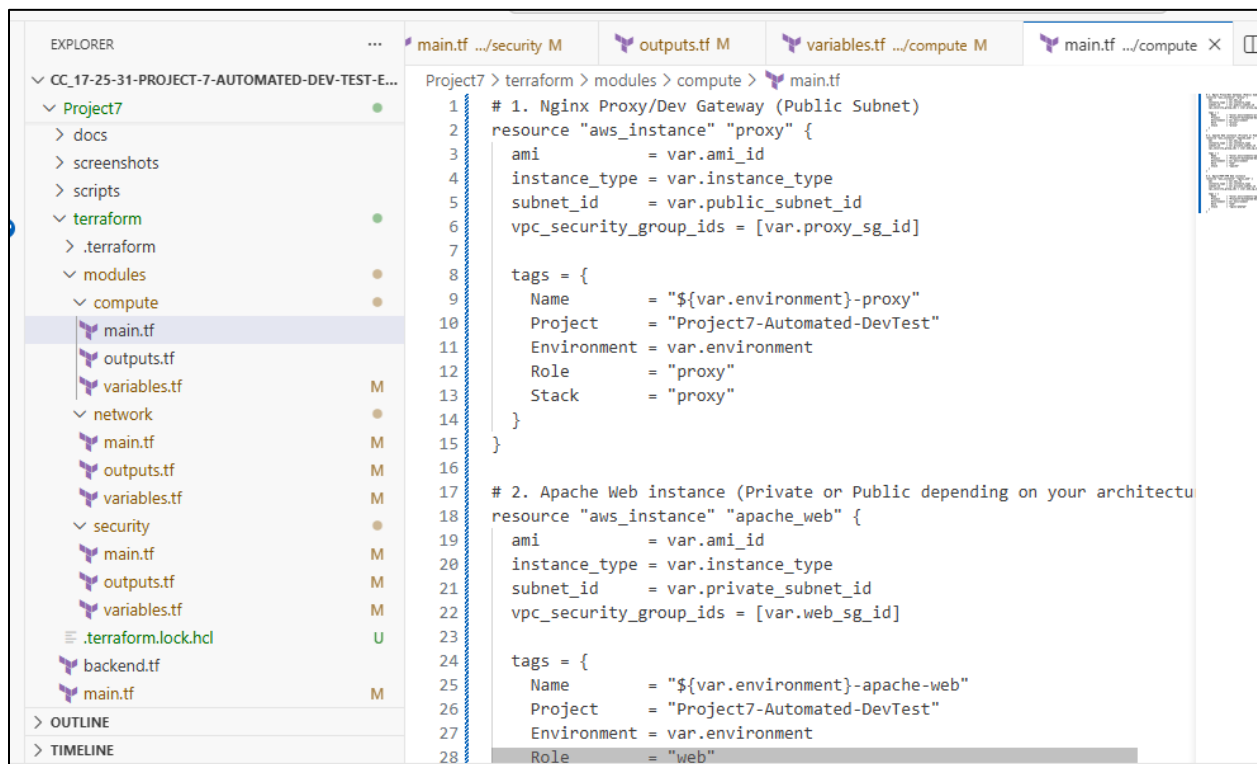This is terraform/modules/compute/ **variables**.tf. This code provisions the three required instances with their specific tags,

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,

```
# 1. Nginx Proxy/Dev Gateway (Public Subnet)
resource "aws_instance" "proxy" {
    ami                    = var.ami_id
    instance_type          = var.instance_type
    subnet_id              = var.public_subnet_id
    vpc_security_group_ids = [var.proxy_sg_id]

    tags = {
        Name        = "${var.environment}-proxy"
        Project     = "Project7-Automated-DevTest"
        Environment = var.environment
        Role        = "proxy"
        Stack       = "proxy"
    }
}

# 2. Apache Web instance (Private or Public depending on your architectu
resource "aws_instance" "apache_web" {
    ami                    = var.ami_id
    instance_type          = var.instance_type
    subnet_id              = var.private_subnet_id
    vpc_security_group_ids = [var.web_sg_id]

    tags = {
        Name        = "${var.environment}-apache-web"
        Project     = "Project7-Automated-DevTest"
        Environment = var.environment
        Role        = "web"
```

```
> scripts
∨ terraform                    ●
  > .terraform
  ∨ modules                    ●
    ∨ compute                  ●
      🔹 main.tf               M
      🔹 outputs.tf
      🔹 variables.tf          M
    ∨ network                  ●
      🔹 main.tf               M
      🔹 outputs.tf            M
      🔹 variables.tf          M
    ∨ security                 ●
      🔹 main.tf               M
      🔹 outputs.tf            M
      🔹 variables.tf          M
  ≡ .terraform.lock.hcl        U
  🔹 backend.tf
```

```
29      Stack        = "apache"
30    }
31  }
32
33  # 3. Nginx+PHP-FPM Web instance
34  resource "aws_instance" "nginx_web" {
35    ami             = var.ami_id
36    instance_type = var.instance_type
37    subnet_id       = var.private_subnet_id
38    vpc_security_group_ids = [var.web_sg_id]
39
40    tags = {
41      Name        = "${var.environment}-nginx-web"
42      Project     = "Project7-Automated-DevTest"
43      Environment = var.environment
44      Role        = "web"
45      Stack       = "nginx-phpfpm"
46    }
47  }
```

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:

```
···        M        🔹 variables.tf .../compute M        🔹 main.tf .../compute M        🔹

1-PROJECT-7-AUTOMATED-DEV-TEST-E...    Project7 > terraform > modules > compute > 🔹 outputs.tf

m                                ●    15    output "proxy_public_dns" {
                                 ●    16      value = aws_instance.proxy.public_dns
nments                                17    }
vars
les                              ●
pute                             ●
```

root outputs.tf,

```
                ●
               ●    26
                    27
               M    28    output "proxy_dns_name" {
               M    29      value = module.compute.proxy_public_dns
               M    30    }
               ●
               M
               M
               M
               ●
               M
```

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:
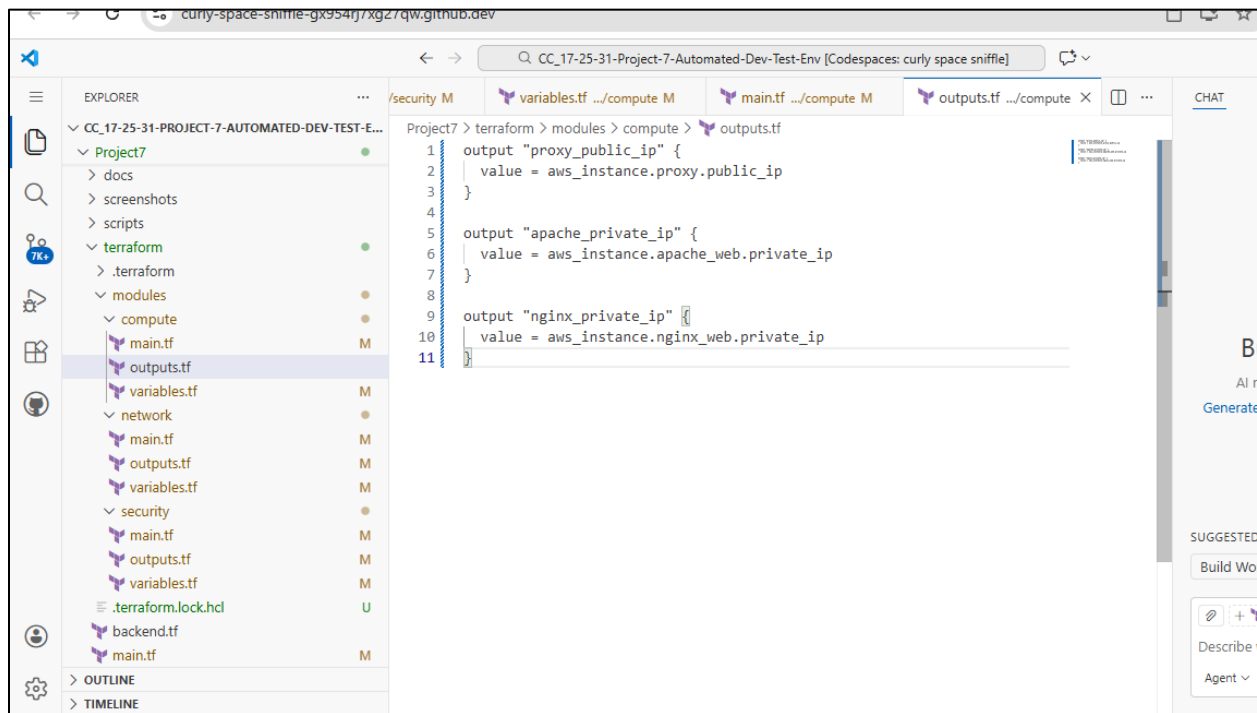
PART III - State Isolation & Environments

```
@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform workspace new dev
Created and switched to workspace "dev"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform workspace new qa
Created and switched to workspace "qa"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform workspace new uat
Created and switched to workspace "uat"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$ terraform workspace list
  default
  dev
  qa
* uat

@emanhanif1 →/workspaces/CC_17-25-31-Project-7-Automated-Dev-Test-Env/Project7/terraform (main)
$
```

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.
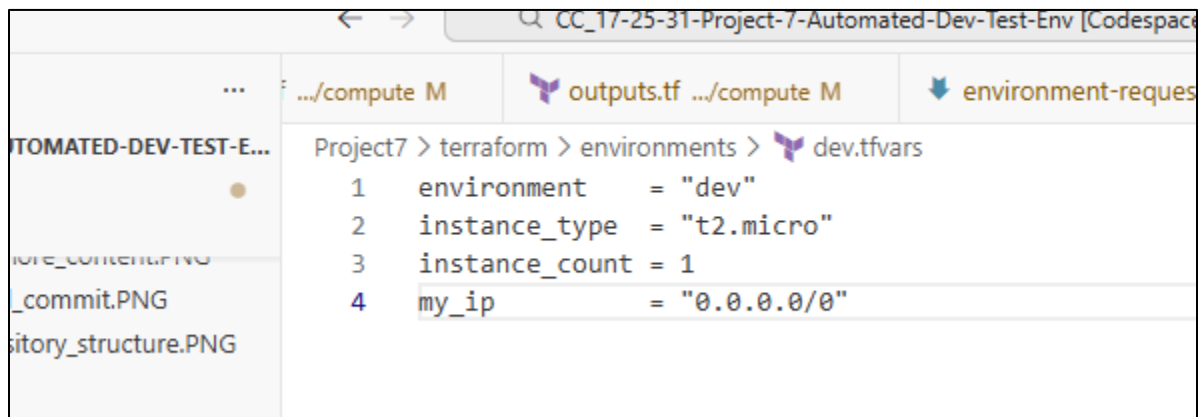
14

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:
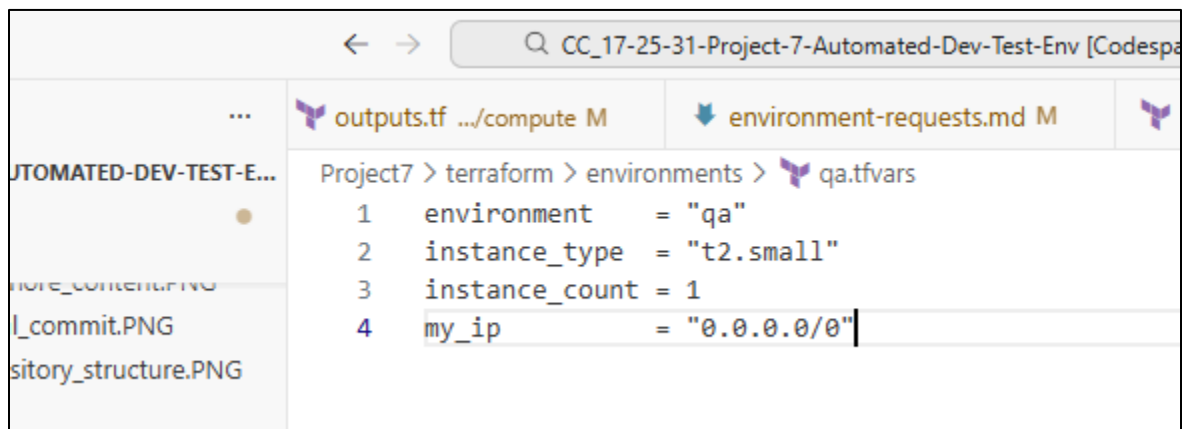
environments/dev.tfvars,

```
Project7 > terraform > environments > dev.tfvars
1    environment    = "dev"
2    instance_type  = "t2.micro"
3    instance_count = 1
4    my_ip          = "0.0.0.0/0"
```

environments/qa.tfvars,

```
Project7 > terraform > environments > qa.tfvars
1    environment    = "qa"
2    instance_type  = "t2.small"
3    instance_count = 1
4    my_ip          = "0.0.0.0/0"
```
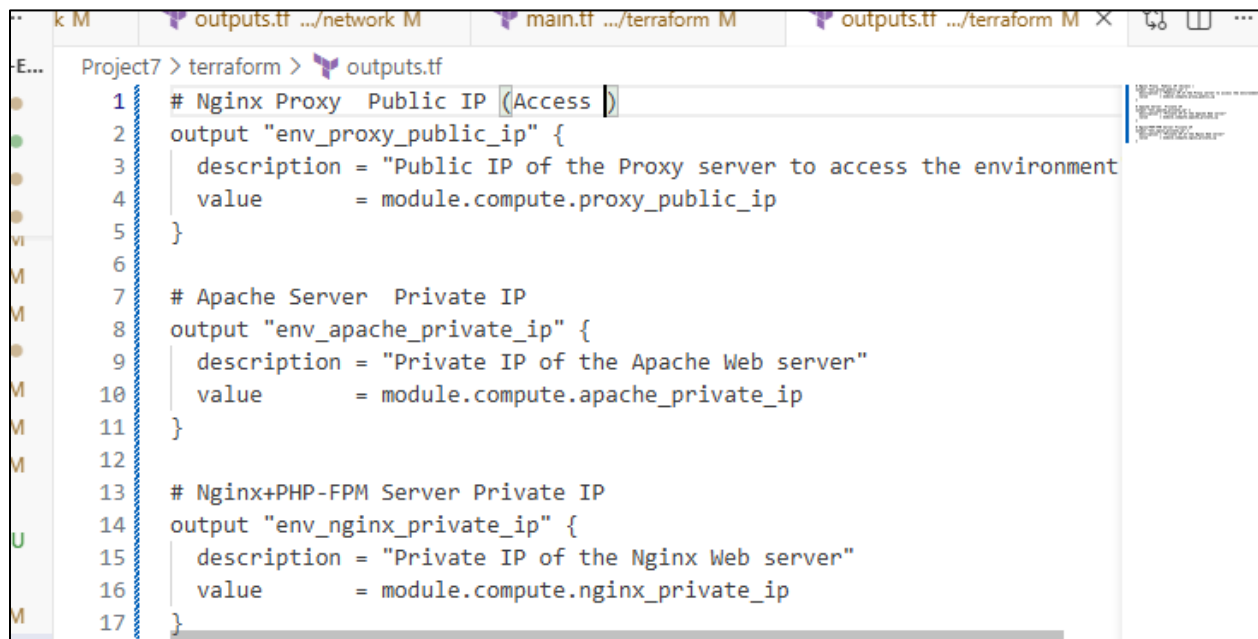
environments/uat.tfvars,

```
Project7 > terraform > environments > uat.tfvars
1    environment     = "uat"
2    instance_type   = "t2.medium"
3    instance_count = 1
4    my_ip           = "0.0.0.0/0"
```

Root outputs,

```
Project7 > terraform > outputs.tf
1    # Nginx Proxy  Public IP (Access )
2    output "env_proxy_public_ip" {
3      description = "Public IP of the Proxy server to access the environment
4      value       = module.compute.proxy_public_ip
5    }
6
7    # Apache Server  Private IP
8    output "env_apache_private_ip" {
9      description = "Private IP of the Apache Web server"
10     value       = module.compute.apache_private_ip
11   }
12
13   # Nginx+PHP-FPM Server Private IP
14   output "env_nginx_private_ip" {
15     description = "Private IP of the Nginx Web server"
16     value       = module.compute.nginx_private_ip
17   }
```

Running this command,

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

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

4.1 Base Tools & Inventory

- project7_part4_inventory_graph.png

4.2 Apache Web Stack (HTTPD + PHP)

- project7_part4_apache_role_main.png

4.3 Nginx+ PHP-FPM Web Stack

- project7_part4_nginx_phpfpm_role_main.png

PART V Automation Scripts, Health Checks & Terraform

5.1 Provision Script

- project7_part5_env_provision_script.png

```
  GNU nano 7.2                              env-provision.sh *
#!/bin/bash

ENV=$1

if [ -z "$ENV" ]; then
  echo "Usage: $0 <dev|qa|uat>"
  exit 1
fi

cd terraform || exit 1

terraform init

terraform workspace select "$ENV" 2>/dev/null || terraform workspace new "$ENV"

terraform apply -var-file="environments/${ENV}.tfvars" -auto-approve

cd ../ansible || exit 1

ansible-playbook -i inventory/${ENV}_aws_ec2.yml playbooks/provision-environment.yml
```

5.2 Teardown Script

- project7_part5_env_destroy_script.png

```
  GNU nano 7.2                          scripts/env-destroy.sh *
#!/bin/bash

ENV=$1

if [ -z "$ENV" ]; then
  echo "Usage: $0 <dev|qa|uat>"
  exit 1
fi

cd terraform || exit 1

terraform workspace select "$ENV"

terraform destroy -var-file="environments/${ENV}.tfvars" -auto-approve
```

5.3 Health Checks

- project7_part5_health_check_script.png

```
  GNU nano 7.2                          scripts/env-health-check.sh *
#!/bin/bash

PROXY_IP=$1

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

echo "Checking /health endpoint..."
curl -s -o /dev/null -w "%{http_code}\n" http://$PROXY_IP/health

echo "Checking Apache backend..."
curl -s http://$PROXY_IP/apache/ | grep -i apache

echo "Checking PHP-FPM backend..."
curl -s http://$PROXY_IP/phpfpm/ | grep -i nginx
```

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**

   o Explicitly destroys all infrastructure for an environment

   o Prevents idle EC2, VPC, and NAT costs

   o Primary cost-optimization mechanism

```
  GNU nano 7.2                          scripts/env-destroy.sh *
#!/bin/bash

ENV=$1

if [ -z "$ENV" ]; then
  echo "Usage: $0 <dev|qa|uat>"
  exit 1
fi

cd terraform || exit 1

terraform workspace select "$ENV"

terraform destroy -var-file="environments/${ENV}.tfvars" -auto-approve
```

20

2. **Environment-Based Lifecycle**

   o   Short-lived dev / qa / uat environments

   o   Created only when needed

   o   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

Project Link:

https://github.com/HamnaMahmood20/CC_17-25-31-Project-7-Automated-Dev-Test-Env