Open-Ended Lab Report

Terraform & Ansible – Frontend Backend High Availability Architecture

Student Name:
Maria Abdul Malik
Registration No:
2023-BSE-076
Course:
Cloud Computing
Lab Type:
Open Ended Lab

1. Introduction

This report presents the complete implementation of an open-ended cloud computing lab using Terraform and Ansible.
The goal of this lab is to provision infrastructure automatically and configure a highly available frontend-backend web architecture.

2. GitHub Repository Initialization

A dedicated GitHub repository was created to maintain version control and ensure clean separation of this lab from previous assignments

```
@mariamalik11 ➜ /workspaces/CC_Maria_076 (main) $ cd LabProject_FrontendBackend
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $
```

3. GitHub Codespaces Environment Setup

GitHub Codespaces was used as the development environment to provide a consistent Linux- based setup.

```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\A>gh codespace list
NAME                                 DISPLAY NAME         REPOSITORY                BRANCH  STATE     CREATED AT
refactored-umbrella-v6j6vpxjwwx...   refactored umbrella  mariamalik11/Lab-9        main*   Shutdown  about 12 days agofi
ncake    mariamalik11/CC_Maria_076  main*   Available  about 2 days ago

C:\Users\A>gh codespace ssh -c fictional-pancake-wrqr7vwqxg6rhq6g
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro
Last login: Sun Jan 18 04:47:26 2026 from ::1
@mariamalik11 /workspaces/CC_Maria_076 (main) $ cd LabProject_FrontendBackend
```

4. Tool Installation & Verification

Required tools including Terraform, AWS CLI, Python, and Ansible were installed and verified before implementation.

```
@mariamalik11 ➜ /workspaces/CC_Maria_076 (main) $ aws --version
aws-cli/2.33.1 Python/3.13.11 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
@mariamalik11 ➜ /workspaces/CC_Maria_076 (main) $ terraform version
Terraform v1.14.3
on linux_amd64
@mariamalik11 ➜ /workspaces/CC_Maria_076 (main) $ ansible --version
ansible [core 2.16.3]
  config file = None
  configured module search path = ['/home/codespace/.ansible/plugins/modules', '/usr/share/ansible/plugins/modu
les']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/codespace/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Nov  6 2025, 13:44:16) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.6
  libyaml = True
@mariamalik11 ➜ /workspaces/CC_Maria_076 (main) $ python3 --version
Python 3.12.1
```

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ mkdir -p ansible/{inventory,playbooks,roles/{frontend/{tasks,handlers,templates
},backend/{tasks,handlers,templates}}}
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ mkdir -p modules/subnet
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ touch main.tf variables.tf locals.tf outputs.tf terraform.tfvars
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ touch ansible/ansible.cfg
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ touch ansible/inventory/hosts
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ touch ansible/playbooks/site.yaml
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $
```

5. Final Project Directory Structure

The final directory structure follows Terraform module-based design and Ansible role-based configuration.

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ cat
itignore
# Terraform
.terraform/
*.tfstate
*.tfstate.*
*.tfvars
.terraform.lock.hcl
.crash.log

# AWS credentials
.aws/

# SSH keys
.ssh/
*.pem

# Ansible
*.retry

# OS / Editor
.vscode/
.DS_Store
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $
```

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ tree -L 4
.
├── README.md
├── ansible
│   ├── ansible.cfg
│   ├── inventory
│   │   └── hosts
│   ├── playbooks
│   │   └── site.yaml
│   └── roles
│       ├── backend
│       │   ├── handlers
│       │   ├── tasks
│       │   └── templates
│       └── frontend
│           ├── handlers
│           ├── tasks
│           └── templates
├── locals.tf
├── main.tf
├── modules
│   └── subnet
├── outputs.tf
├── terraform.tfvars
└── variables.tf

15 directories, 9 files
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $
```

6. Terraform Configuration

6.1 Provider & Variables Definition

Terraform provider configuration specifies the AWS region and credentials. Variables are used for reusability and flexibility.

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ c
at variables.tf
variable "project_name" {
  description = "Project name prefix"
  type        = string
  default     = "frontend-backend-lab"
}

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

variable "vpc_cidr" {
  description = "CIDR block for VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "public_subnet_cidrs" {
  description = "Public subnet CIDRs"
  type        = list(string)
  default     = [
    "10.0.1.0/24",
    "10.0.2.0/24"
  ]
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
}

variable "key_name" {
  description = "EC2 SSH key pair name"
  type        = string
}
```

6.2 Local Values & IP Restriction

Local values dynamically retrieve the user's public IP to restrict SSH access securely.

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $ c
at locals.tf
locals {
  frontend_count = 1
  backend_count  = 3

  common_tags = {
    Project = var.project_name
    Managed = "Terraform"
  }
}
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend (main) $
```

6.3 Networking (VPC, Subnet, Routing)

A VPC, public subnet, internet gateway, and route table are provisioned to allow internet access.

```
 GNU nano 7.2                                                    main.tf
ovider "aws" {
 region = var.region


ata "aws_ami" "amazon_linux" {
 most_recent = true
 filter {
   name   = "name"
   values = ["amzn2-ami-hvm-*-x86_64-gp2"]
 }
 owners = ["amazon"]


odule "network" {
 source = "./modules/subnet"

 vpc_cidr_block     = var.vpc_cidr_block
 subnet_cidr_block  = var.subnet_cidr_block
 availability_zone  = var.availability_zone
 env_prefix         = var.env_prefix


esource "aws_security_group" "web_sg" {
 name   = "${var.env_prefix}-sg"
 vpc_id = module.network.vpc_id

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

 ingress {
   from_port   = 80
   to_port     = 80
   protocol    = "tcp"
```

6.4 EC2 Frontend & Backend Provisioning

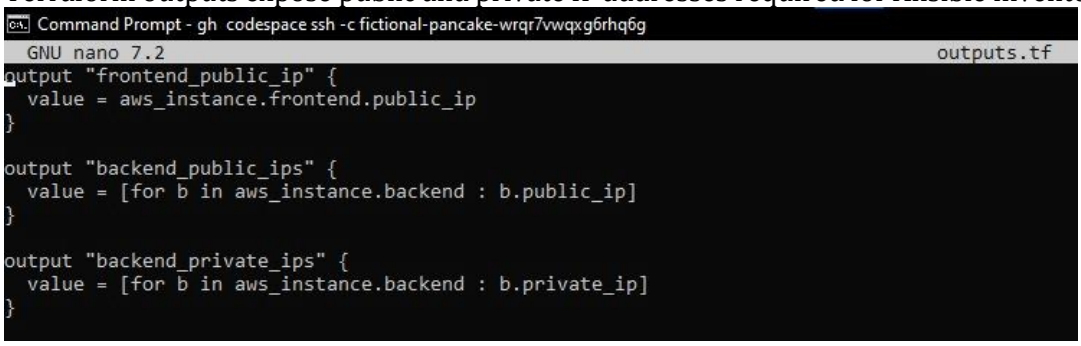
One frontend EC2 instance and tree backend EC2 instances are created using reusable Terraform modules.

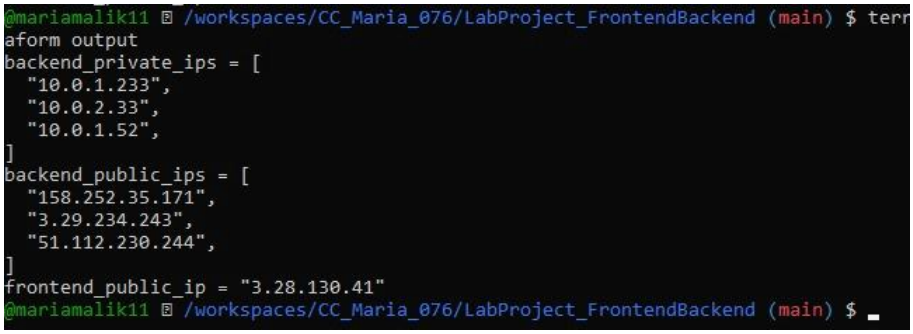
GNU nano 7.2 main.tf

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

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

  ami_id             = data.aws_ami.amazon_linux.id
  instance_type      = var.instance_type
  subnet_id          = module.network.subnet_id
  security_group_ids = [aws_security_group.web_sg.id]
  key_name           = aws_key_pair.key.key_name
  name               = "${var.env_prefix}-frontend"
  count              = 1
}

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

  ami_id             = data.aws_ami.amazon_linux.id
  instance_type      = var.instance_type
  subnet_id          = module.network.subnet_id
  security_group_ids = [aws_security_group.web_sg.id]
  key_name           = aws_key_pair.key.key_name
  name               = "${var.env_prefix}-backend"
  count              = 3
}

resource "null_resource" "ansible_run" {
  depends_on = [
    module.frontend,
    module.backend
  ]
```

6.5 Terraform Outputs

Terraform outputs expose public and private IP addresses required for Ansible inventory and verification.





6.6 Terraform Modules Implementation

Terraform modules were used to improve code reusability, readability, and separation of concerns. Instead of defining all resources in a single file, networking and compute logic were encapsulated into independent modules.

This approach aligns with infrastructure-as-code best practices and simplifies future scaling.

6.6.1 Subnet Module (Networking Layer)

The subnet module is responsible for provisioning the core networking components required for the project. It creates the Virtual Private Cloud (VPC), public subnet, internet gateway, route table, and route table association.

This ensures that all EC2 instances are deployed in a properly configured public network with internet access

```
GNU nano 7.2                                                    main.t

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
  tags = merge(local.common_tags, {
    Name = "${var.project_name}-igw"
  })
}
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main_vpc.id
  tags = merge(local.common_tags, {
    Name = "${var.project_name}-public-rt"
  })
}

resource "aws_route" "default_route" {
  route_table_id         = aws_route_table.public_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.igw.id
}

resource "aws_route_table_association" "public_assoc" {
  count          = length(aws_subnet.public)
  subnet_id      = aws_subnet.public[count.index].id
  route_table_id = aws_route_table.public_rt.id
}

resource "aws_instance" "frontend" {
  ami                    = data.aws_ami.amazon_linux.id # Amazon Linux 2 (ME Central)
  instance_type          = "t3.micro"
  subnet_id              = aws_subnet.public[0].id
  key_name               = aws_key_pair.deployer.key_name
  vpc_security_group_ids = [aws_security_group.web_sg.id]
  associate_public_ip_address = true

  tags = merge(local.common_tags, {
    Name = "${var.project_name}-frontend"
    Role = "frontend"
  })
}
```

```
Command Prompt - gh  codespace ssh -c fictional-pancake-wrqr7vwqxg6rhq6g
  GNU nano 7.2                                                  variables.tf
variable "project_name" {
  description = "Project name prefix"
  type        = string
  default     = "frontend-backend-lab"
}

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

variable "vpc_cidr" {
  description = "CIDR block for VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "public_subnet_cidrs" {
  description = "Public subnet CIDRs"
  type        = list(string)
  default     = [
    "10.0.1.0/24",
    "10.0.2.0/24"
  ]
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t3.micro"
}
```

```
 GNU nano 7.2                                  modules/subnet/variables.tf *
variable "vpc_cidr_block" {
  type = string
}

variable "subnet_cidr_block" {
  type = string
}

variable "availability_zone" {
  type = string
}

variable "env_prefix" {
  type = string
}
```

```
 GNU nano 7.2                                  modules/subnet/outputs.tf
output "vpc_id" {
  value = aws_vpc.this.id
}

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

6.6.2 Webserver Module (EC2 Abstraction Layer)

The webserver module abstracts EC2 instance creation into a reusable component.
It is used to provision both frontend and backend instances by passing different parameters such as instance count and naming prefix.

This modular design avoids duplication and ensures consistent EC2 configuration.

7. Ansible Configuration:

7.1 Ansible Configuration File

The ansible.cfg file disables host key checking and specifies Python interpreter settings.

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ @mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main)
$ cat ansible.cfg
[defaults]
# Disables SSH host key checking for automated connections
host_key_checking = False

# Path to your roles directory
roles_path = ./roles

# Specifies the Python interpreter for consistency across environments
interpreter_python = auto_silent

# Default remote user for AWS EC2 instances
remote_user = ec2-user

# Private key file for authentication (ensure the path matches your .pem file)
private_key_file = ~/.ssh/id_ed25519
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $
```

7.2 Inventory File

The inventory file groups frontend and backend servers to enable role-based execution.

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat inventory/hosts
[frontend]
3.28.130.41

[backends]
51.112.230.244
3.29.234.243
158.252.35.171


[all:vars]
ansible_user=ec2-user
ansible_ssh_private_key_file=~/.ssh/id_ed25519
```

7.3 Backend Role (Apache HTTPD)

The backend role installs Apache HTTPD and deploys unique index pages on each backend server.

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat roles/backend/tasks/main.yml
---
- name: Install Apache (httpd)
  dnf:
    name: httpd
    state: present

- name: Start and Enable Apache
  service:
    name: httpd
    state: started
    enabled: yes

- name: Create basic web page
  copy:
    content: |
      <html>
      <body>
        <h1>Backend Server: {{ inventory_hostname }}</h1>
      </body>
      </html>
    dest: /var/www/html/index.html
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $
```

7.4 Frontend Role (Nginx Load Balancer)

The frontend role installs Nginx and configures it as a reverse proxy with two primary and one backup backend.

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/tasks/main.yml
---
- name: Install Nginx
  yum:
    name: nginx
    state: present

- name: Configure Nginx load balancer
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
    force: yes
  notify: Restart Nginx

- name: Start and enable Nginx
  service:
    name: nginx
    state: started
    enabled: yes
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $
```

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/templates/nginx.conf.j
events {
    worker_connections 1024;
}

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

    upstream backend_servers {
        {% for host in groups['backends'] %}
        # Backend {{ loop.index }}
        server {{ hostvars[host].ansible_host }}:80{% if loop.index == 3 %} backup{% endif %};
        {% endfor %}
    }

    server {
        listen 80;
        server_name _;

        location / {
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;

            # Add a custom header to see which backend handled the request
            add_header X-Backend-Server $upstream_addr;
        }
    }
}
```

```
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/handlers/main.yaml
---
- name: restart nginx
service:
name:nginx
state:restarted
@mariamalik11 /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $
```

7.5 Main Playbook (site.yaml)

The main playbook applies backend configuration first, followed by frontend setup.

```
@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $ cat playbooks/site.yaml
---
- hosts: backends
  become: yes
  roles:
    - backend

- hosts: frontend
  become: yes
  roles:
    - frontend

@mariamalik11 ➜ /workspaces/CC_Maria_076/LabProject_FrontendBackend/ansible (main) $
```

8. Terraform Apply & Automation Execution

Terraform apply provisions all resources and automatically triggers Ansible using a null_resource.

```
ok: [3.29.234.243]
ok: [158.252.35.171]

TASK [backend : Start and Enable Apache] ***************************************************************************************************************
ok: [3.29.234.243]
ok: [158.252.35.171]
ok: [51.112.230.244]

TASK [backend : Create basic web page] *****************************************************************************************************************
ok: [158.252.35.171]
ok: [3.29.234.243]
ok: [51.112.230.244]

PLAY [frontend] ****************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************
[WARNING]: Platform linux on host 3.28.130.41 is using the discovered Python interpreter at /usr/bin/python3.9, but future installation of another Python interpreter
could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.16/reference_appendices/interpreter_discovery.html for more information.
ok: [3.28.130.41]

TASK [frontend : Install Nginx] ************************************************************************************************************************
ok: [3.28.130.41]

TASK [frontend : Configure Nginx load balancer] ********************************************************************************************************
changed: [3.28.130.41]

TASK [frontend : Start and enable Nginx] ***************************************************************************************************************
changed: [3.28.130.41]

RUNNING HANDLER [frontend : Restart Nginx] *************************************************************************************************************
changed: [3.28.130.41]

PLAY RECAP *********************************************************************************************************************************************
158.252.35.171             : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
3.28.130.41                : ok=5    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
3.29.234.243               : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
51.112.230.244             : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

9. Backend Server Verification

The following unlabeled screenshots show direct access to backend servers using their public IPs.

Backend Server

Backend Server

Backend Server

10. Frontend Load Balancer Verification

Frontend Response

Frontend Response

11. Load Balancing Behaviour Validation

Refresh-1

Refresh-2

Refresh-3

Success! This is Backend Server ip-10-0-1-52

Refresh-4

12. Failover Testing (Backup Backend)

Success! This is Backend Server ip-10-0-1-233

Backup Backend Response

13. Conclusion

This lab successfully demonstrates automated infrastructure provisioning and configuration management. The architecture meets all requirements of high availability, modularity, and automation.