Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

Open-Ended Lab Report
Terraform & Ansible – Frontend Backend High Availability Architecture

Student Name: Nayab Khazin

Registration No: 2023-BSE-046

Course: Cloud Computing

Lab Type: Open Ended Lab

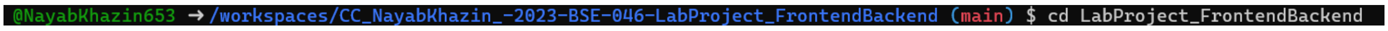
Contents

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.

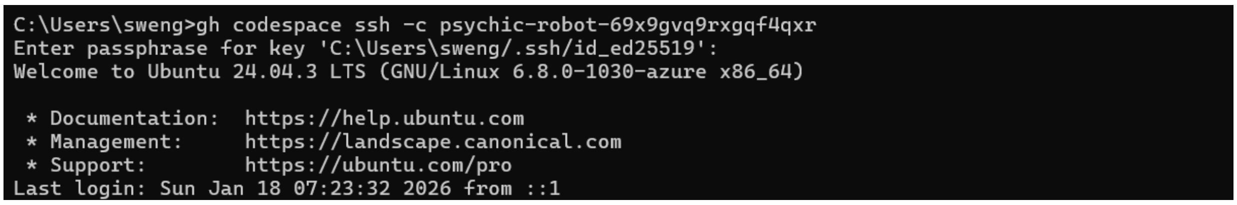


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ cd LabProject_FrontendBackend
```

(Figure 1: GitHub Repository Creation)

3. GitHub Codespaces Environment Setup

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

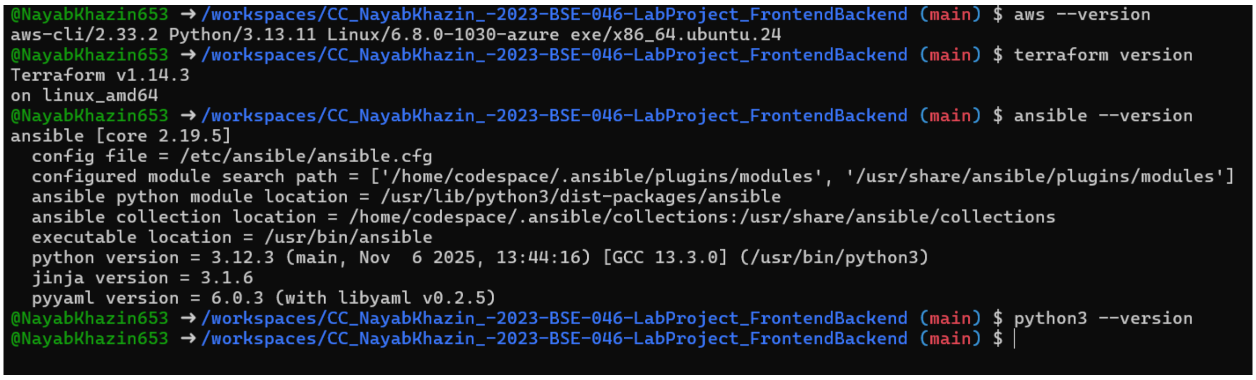


```
C:\Users\sweng>gh codespace ssh -c psychic-robot-69x9gvq9rxgqf4qxr
Enter passphrase for key 'C:\Users\sweng/.ssh/id_ed25519':
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-1030-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro
Last login: Sun Jan 18 07:23:32 2026 from ::1
```

(Figure 2: Codespaces Environment)

4. Tool Installation & Verification

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

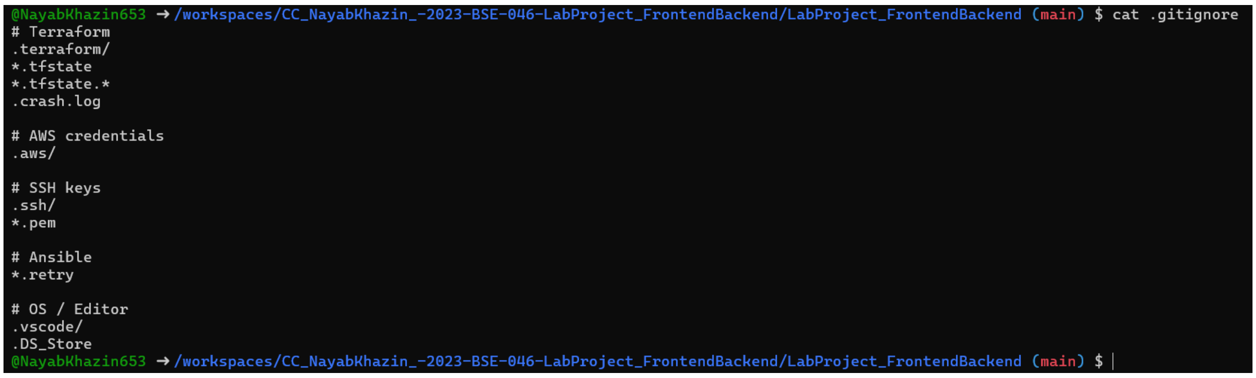


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ aws --version
aws-cli/2.33.2 Python/3.13.11 Linux/6.8.0-1030-azure exe/x86_64.ubuntu.24
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ terraform version
Terraform v1.14.3
on linux_amd64
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ ansible --version
ansible [core 2.19.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/codespace/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  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
  pyyaml version = 6.0.3 (with libyaml v0.2.5)
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ python3 --version
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ |
```

(Figure 3: Terraform Version, Ansible Version)

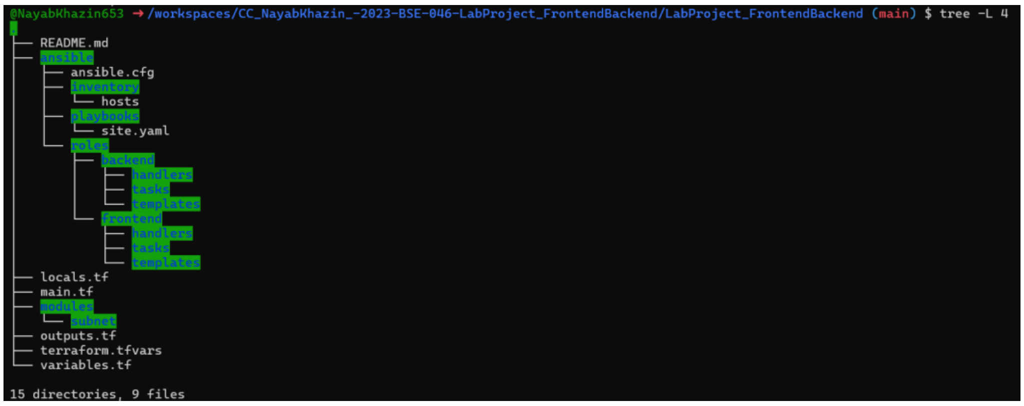
5. Final Project Directory Structure

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



```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ cat .gitignore
# Terraform
.terraform/
*.tfstate
*.tfstate.*
.crash.log

# AWS credentials
.aws/

# SSH keys
.ssh/
*.pem

# Ansible
*.retry

# OS / Editor
.vscode/
.DS_Store
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ |
```

(Figure 4: .gitignore file)



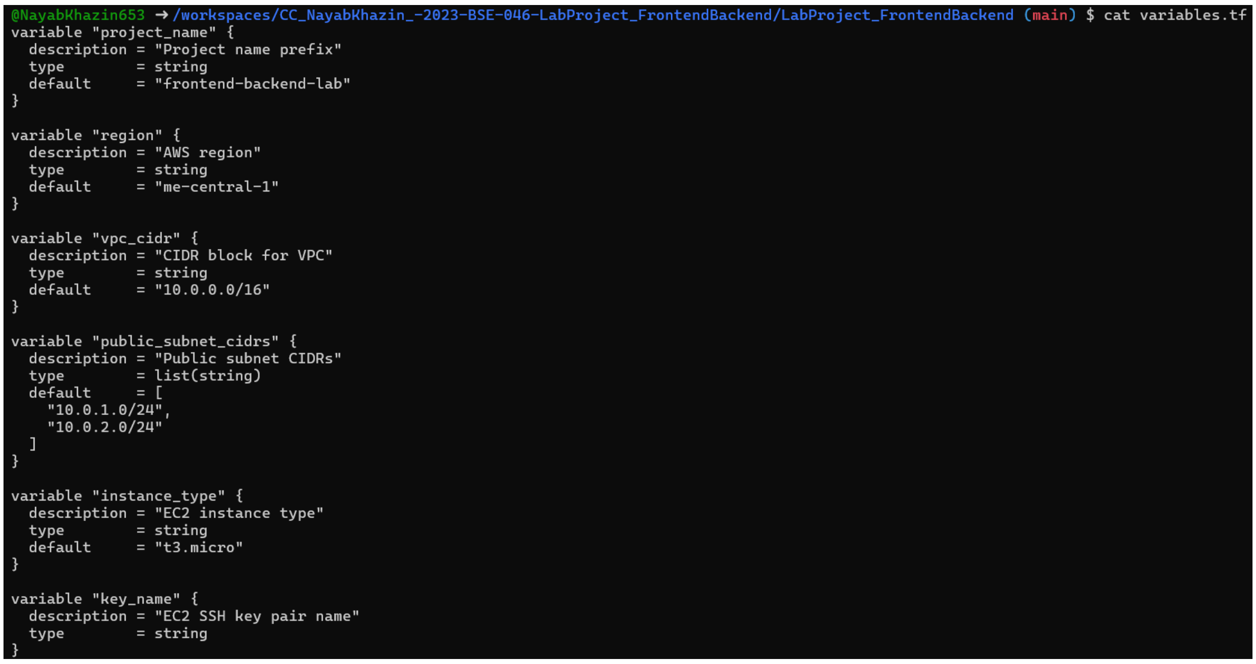
```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ tree -L 4
.
├── README.md
├── ansible
│   ├── ansible.cfg
│   ├── inventory
│   │   └── hosts
│   ├── playbooks
│   │   └── site.yaml
│   └── roles
│       ├── backend
│       │   ├── handlers
│       │   ├── tasks
│       │   └── templates
│       └── frontend
│           ├── handlers
│           ├── tasks
│           └── templates
├── locals.tf
├── main.tf
├── modules
│   └── subnet
├── outputs.tf
├── terraform.tfvars
└── variables.tf

15 directories, 9 files
```

(Figure 5: Project Directory Tree)

6. Terraform Configuration

6.1 Provider & Variables Definition

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

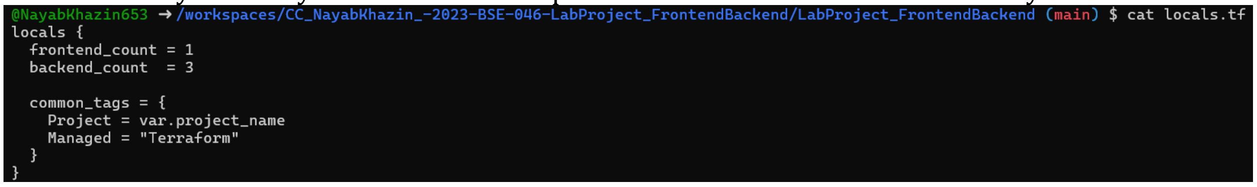


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ cat variables.tf
variable "project_name" {
  description = "Project name prefix"
  type        = string
  default     = "frontend-backend-lab"
}

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
}
```

(Figure 6: variables.tf)

6.2 Local Values & IP Restriction

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

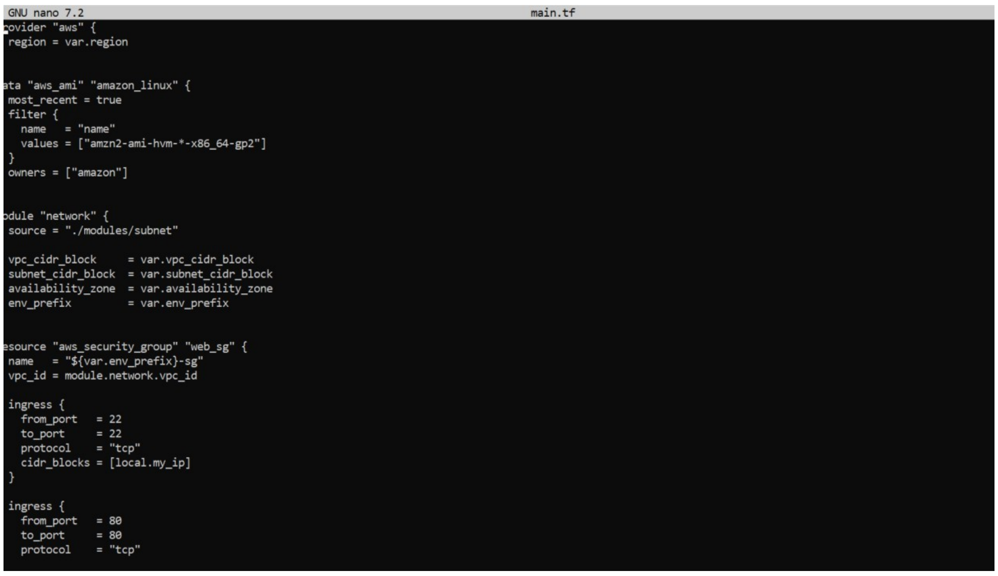


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ cat locals.tf
locals {
  frontend_count = 1
  backend_count  = 3

  common_tags = {
    Project = var.project_name
    Managed = "Terraform"
  }
}
```

(Figure 7: locals.tf)

6.3 Networking (VPC, Subnet, Routing)

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

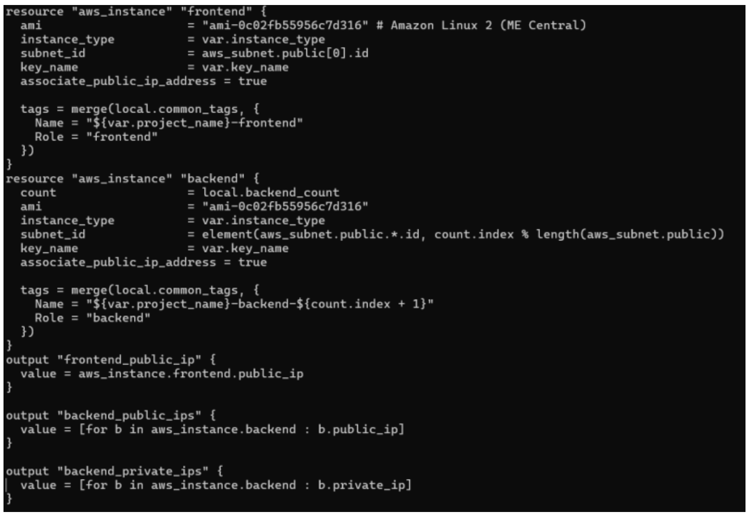


```
GNU nano 7.2                                                        main.tf
ovider "aws" {
 region = var.region


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


dule "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"
```

(Figure 8: VPC & Subnet Configuration)

6.4 EC2 Frontend & Backend Provisioning

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

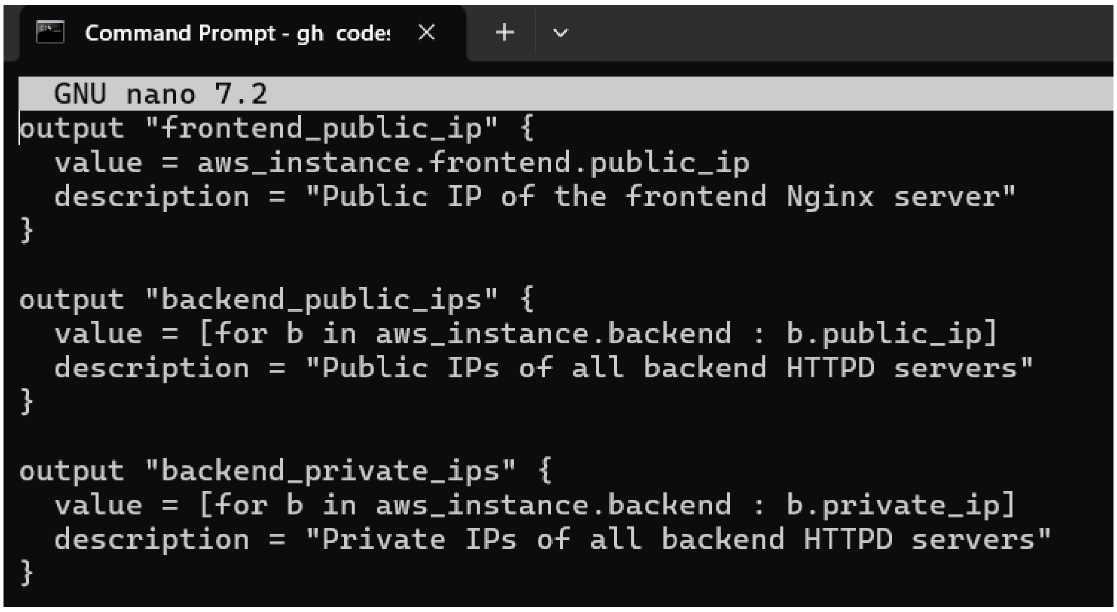


```
resource "aws_instance" "frontend" {
  ami                         = "ami-0c02fb55956c7d316" # Amazon Linux 2 (ME Central)
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.public[0].id
  key_name                    = var.key_name
  associate_public_ip_address = true

  tags = merge(local.common_tags, {
    Name = "${var.project_name}-frontend"
    Role = "frontend"
  })
}
resource "aws_instance" "backend" {
  count                       = local.backend_count
  ami                         = "ami-0c02fb55956c7d316"
  instance_type               = var.instance_type
  subnet_id                   = element(aws_subnet.public.*.id, count.index % length(aws_subnet.public))
  key_name                    = var.key_name
  associate_public_ip_address = true

  tags = merge(local.common_tags, {
    Name = "${var.project_name}-backend-${count.index + 1}"
    Role = "backend"
  })
}
output "frontend_public_ip" {
  value = aws_instance.frontend.public_ip
}

output "backend_public_ips" {
  value = [for b in aws_instance.backend : b.public_ip]
}

output "backend_private_ips" {
  value = [for b in aws_instance.backend : b.private_ip]
}
```

(Figure 9: EC2 Instance Creation)

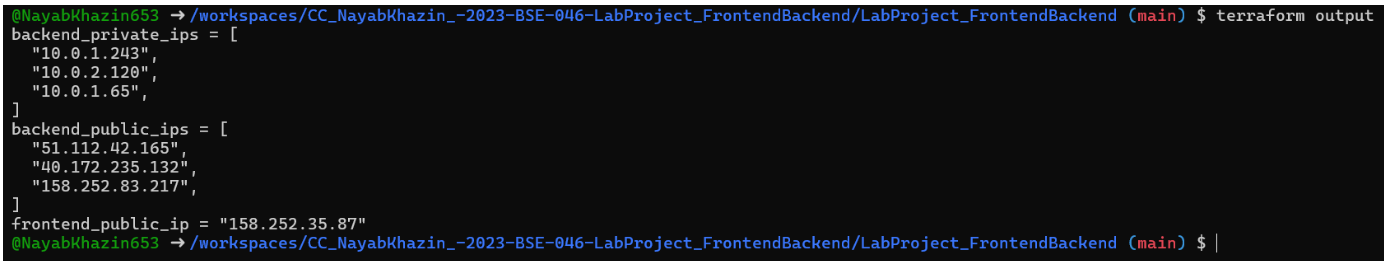
6.5 Terraform Outputs

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



```
  GNU nano 7.2
output "frontend_public_ip" {
  value = aws_instance.frontend.public_ip
  description = "Public IP of the frontend Nginx server"
}

output "backend_public_ips" {
  value = [for b in aws_instance.backend : b.public_ip]
  description = "Public IPs of all backend HTTPD servers"
}

output "backend_private_ips" {
  value = [for b in aws_instance.backend : b.private_ip]
  description = "Private IPs of all backend HTTPD servers"
}
```

(Figure 10-A: outputs.tf)



```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ terraform output
backend_private_ips = [
  "10.0.1.243",
  "10.0.2.120",
  "10.0.1.65",
]
backend_public_ips = [
  "51.112.42.165",
  "40.172.235.132",
  "158.252.83.217",
]
frontend_public_ip = "158.252.35.87"
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/LabProject_FrontendBackend (main) $ |
```

(Figure 10-B: outputs.tf)

6.6 Terraform Modules Implementation

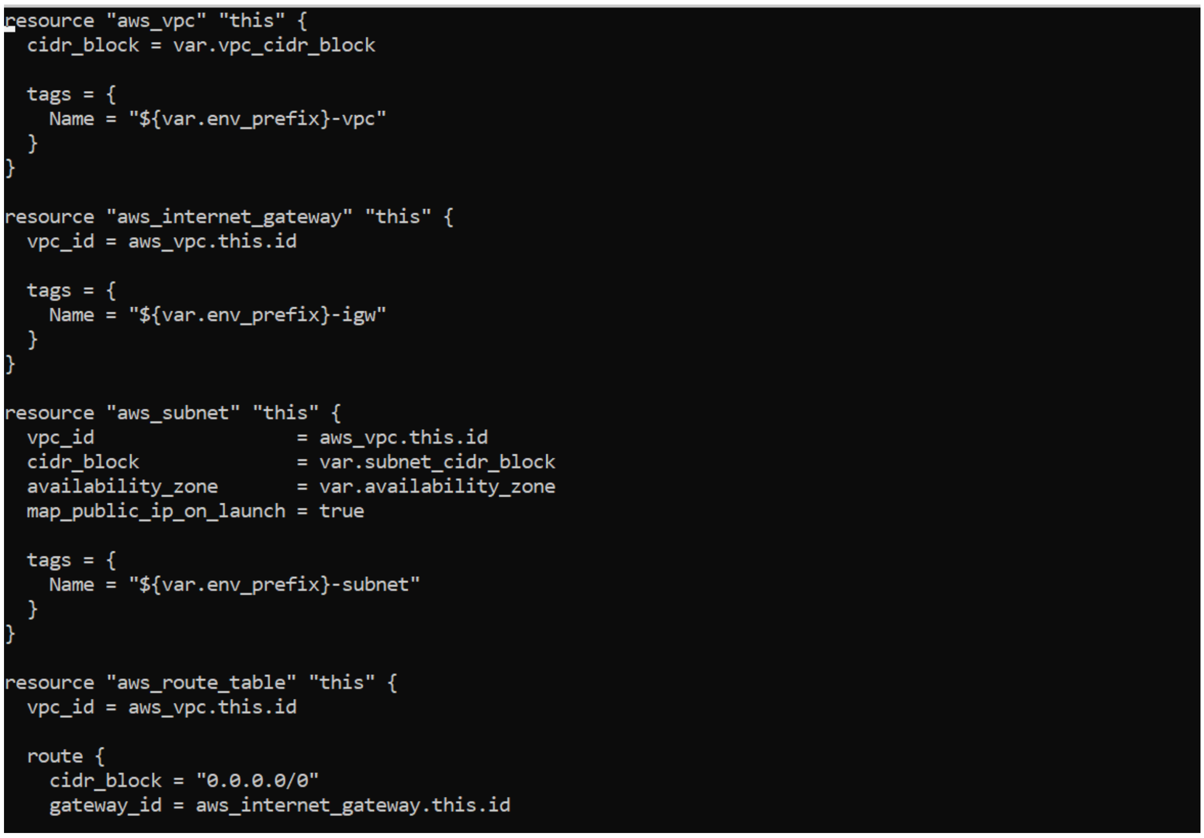
Terraform modules were used to improve code reusability, readability, and separation of concerns.
Instead of defining all resources in a single file, networking and compute logic were encapsulated into independent modules.

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

6.6.1 Subnet Module (Networking Layer)

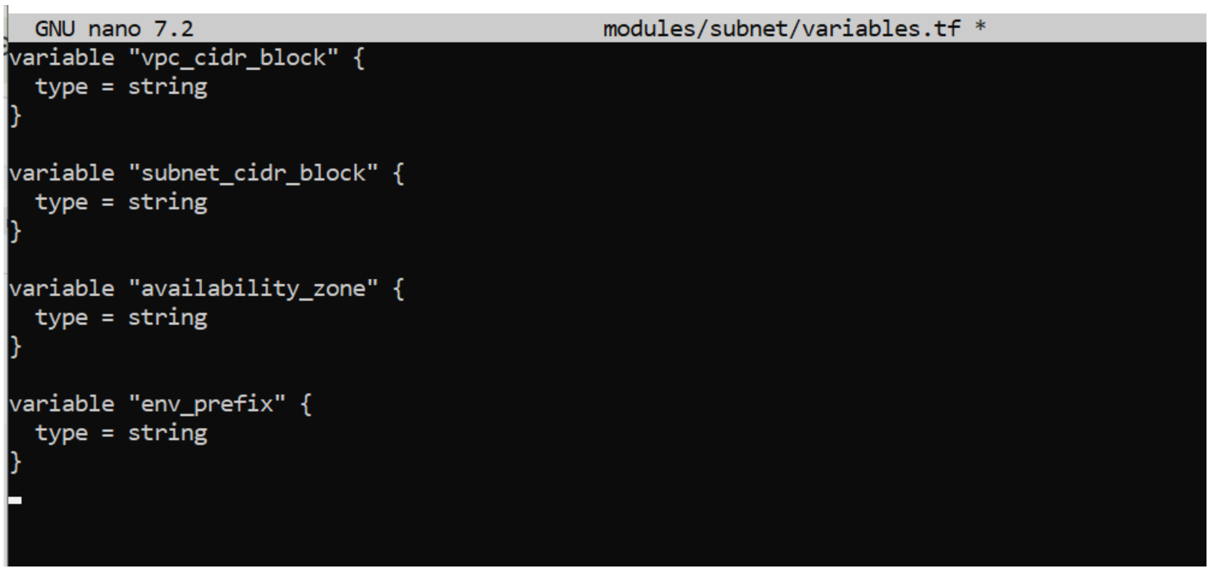
The subnet module is responsible for provisioning the core networking components required for the project.
It creates the Virtual Private Cloud (VPC), public subnet, internet gateway, route table, and route table association.

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



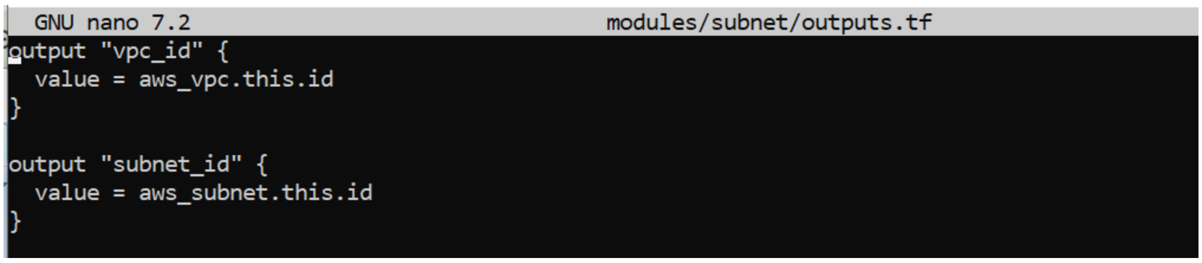
```
resource "aws_vpc" "this" {
  cidr_block = var.vpc_cidr_block

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

resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id

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

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

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

resource "aws_route_table" "this" {
  vpc_id = aws_vpc.this.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.this.id
```

(Figure 12: Subnet module main.tf – VPC, subnet, and routing configuration)



```
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
}
```

(Figure 13: Subnet module variables.tf)



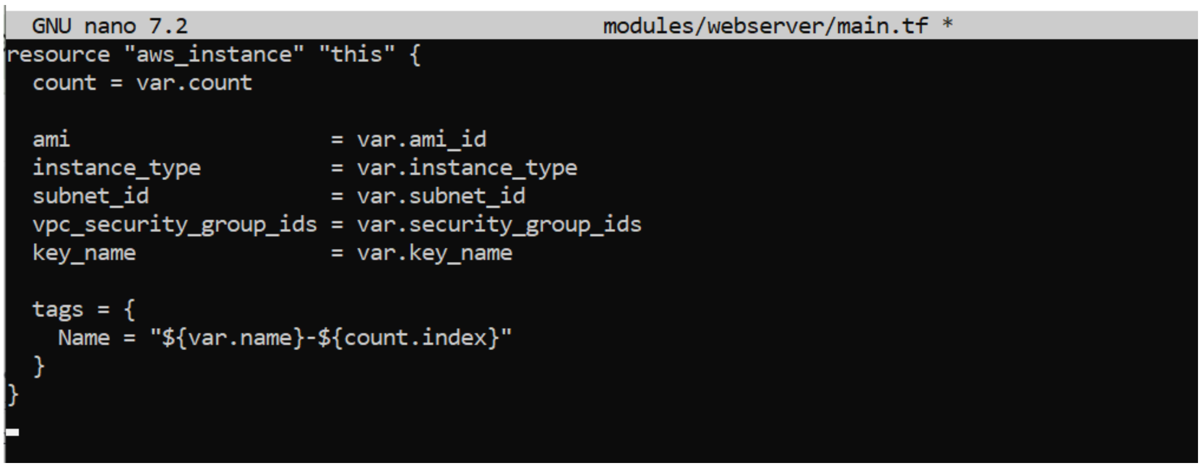
```
GNU nano 7.2                                 modules/subnet/outputs.tf
output "vpc_id" {
  value = aws_vpc.this.id
}

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

(Figure 14: Subnet module outputs.tf)

6.6.2 Webserver Module (EC2 Abstraction Layer)

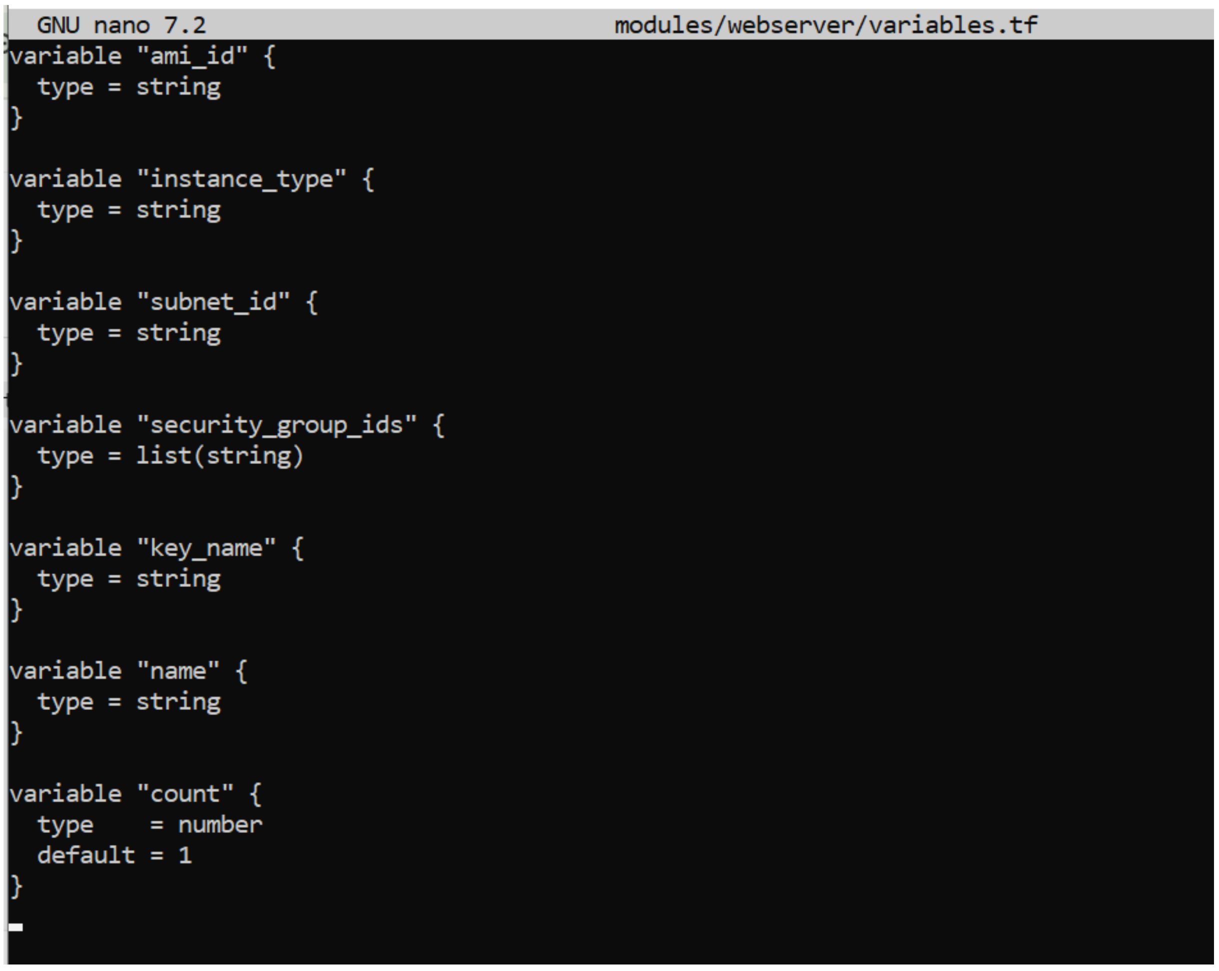
The webserver module abstracts EC2 instance creation into a reusable component.
It is used to provision both frontend and backend instances by passing different parameters such as instance count and naming prefix.

This modular design avoids duplication and ensures consistent EC2 configuration.



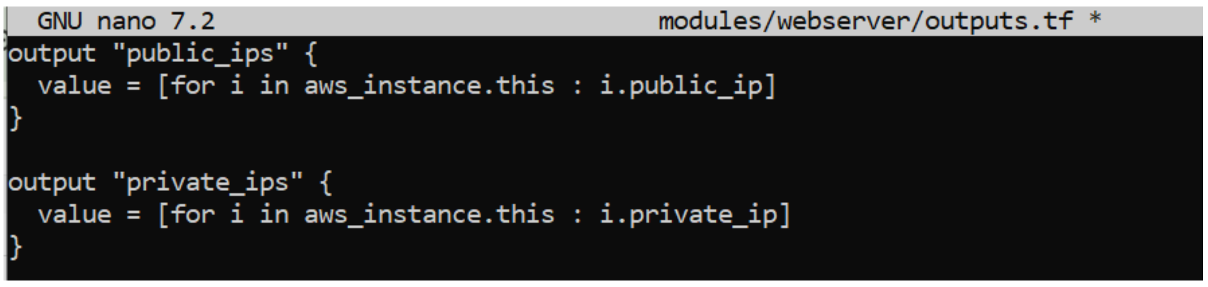
```
GNU nano 7.2                                 modules/webserver/main.tf *
resource "aws_instance" "this" {
  count = var.count

  ami                    = var.ami_id
  instance_type          = var.instance_type
  subnet_id              = var.subnet_id
  vpc_security_group_ids = var.security_group_ids
  key_name               = var.key_name

  tags = {
    Name = "${var.name}-${count.index}"
  }
}
```

(Figure 15: Webserver module main.tf – EC2 instance resource definition)



```
  GNU nano 7.2                              modules/webserver/variables.tf
variable "ami_id" {
  type = string
}

variable "instance_type" {
  type = string
}

variable "subnet_id" {
  type = string
}

variable "security_group_ids" {
  type = list(string)
}

variable "key_name" {
  type = string
}

variable "name" {
  type = string
}

variable "count" {
  type    = number
  default = 1
}
```

(Figure 16: Webserver module variables.tf)



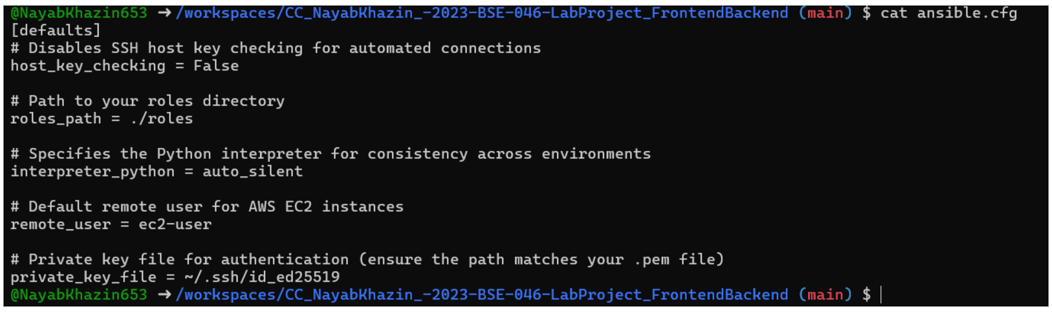
```
  GNU nano 7.2                              modules/webserver/outputs.tf *
output "public_ips" {
  value = [for i in aws_instance.this : i.public_ip]
}

output "private_ips" {
  value = [for i in aws_instance.this : i.private_ip]
}
```

(Figure 17: Webserver module outputs.tf)

7. Ansible Configuration

7.1 Ansible Configuration File

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

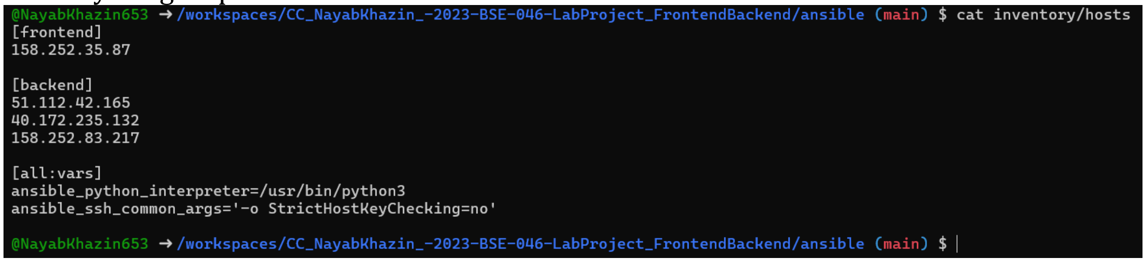


```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ cat ansible.cfg
[defaults]
# Disables SSH host key checking for automated connections
host_key_checking = False

# 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
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend (main) $ |
```

(Figure 18: ansible.cfg)

7.2 Inventory File

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

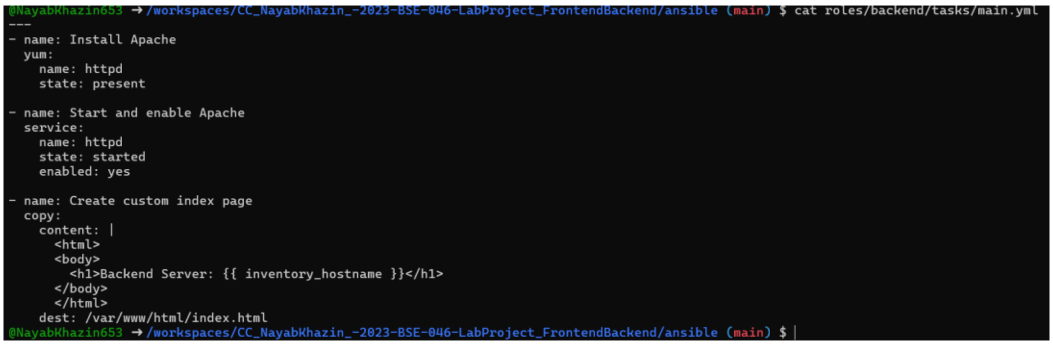


```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat inventory/hosts
[frontend]
158.252.35.87

[backend]
51.112.42.165
40.172.235.132
158.252.83.217

[all:vars]
ansible_python_interpreter=/usr/bin/python3
ansible_ssh_common_args='-o StrictHostKeyChecking=no'

@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 19: Inventory hosts file)

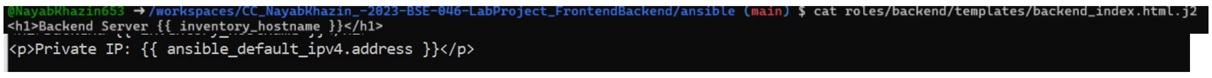
7.3 Backend Role (Apache HTTPD)

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



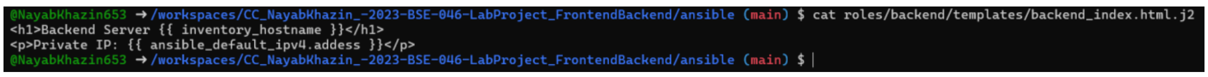
```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/backend/tasks/main.yml
---
- name: Install Apache
  yum:
    name: httpd
    state: present

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

- name: Create custom index page
  copy:
    content: |
      <html>
      <body>
        <h1>Backend Server: {{ inventory_hostname }}</h1>
      </body>
      </html>
    dest: /var/www/html/index.html
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 20-A: Backend Role Tasks)



```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/backend/templates/backend_index.html.j2
<h1>Backend Server {{ inventory_hostname }}</h1>
<p>Private IP: {{ ansible_default_ipv4.address }}</p>
```

(Figure 20-B: Backend Role Templates)

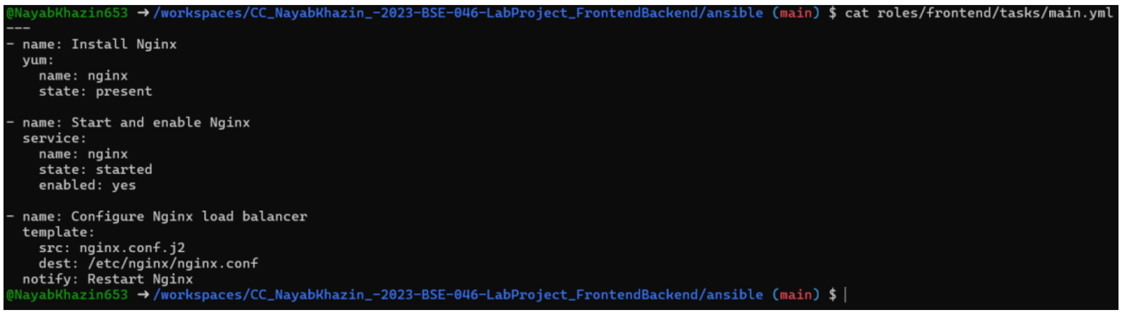


```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/backend/templates/backend_index.html.j2
<h1>Backend Server {{ inventory_hostname }}</h1>
<p>Private IP: {{ ansible_default_ipv4.addess }}</p>
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 20-C: Backend Role Handler)

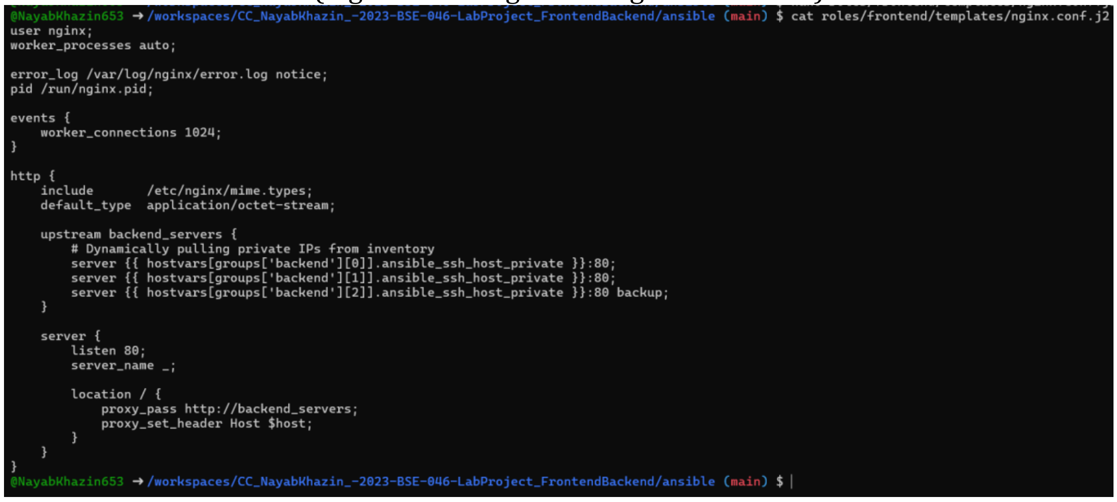
7.4 Frontend Role (Nginx Load Balancer)

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



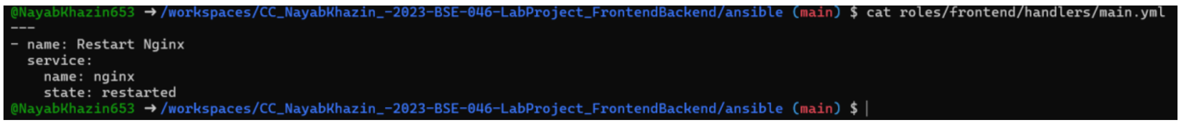
```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/tasks/main.yml
---
- name: Install Nginx
  yum:
    name: nginx
    state: present

- name: Start and enable Nginx
  service:
    name: nginx
    state: started
    enabled: yes

- name: Configure Nginx load balancer
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
  notify: Restart Nginx
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 21-A: Nginx Configuration Tasks)



```
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/templates/nginx.conf.j2
user nginx;
worker_processes auto;

error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

events {
    worker_connections 1024;
}

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

    upstream backend_servers {
        # Dynamically pulling private IPs from inventory
        server {{ hostvars[groups['backend'][0]].ansible_ssh_host_private }}:80;
        server {{ hostvars[groups['backend'][1]].ansible_ssh_host_private }}:80;
        server {{ hostvars[groups['backend'][2]].ansible_ssh_host_private }}:80 backup;
    }

    server {
        listen 80;
        server_name _;

        location / {
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
        }
    }
}
@NayabKhazin653 → /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 21-B: Nginx Configuration Template)

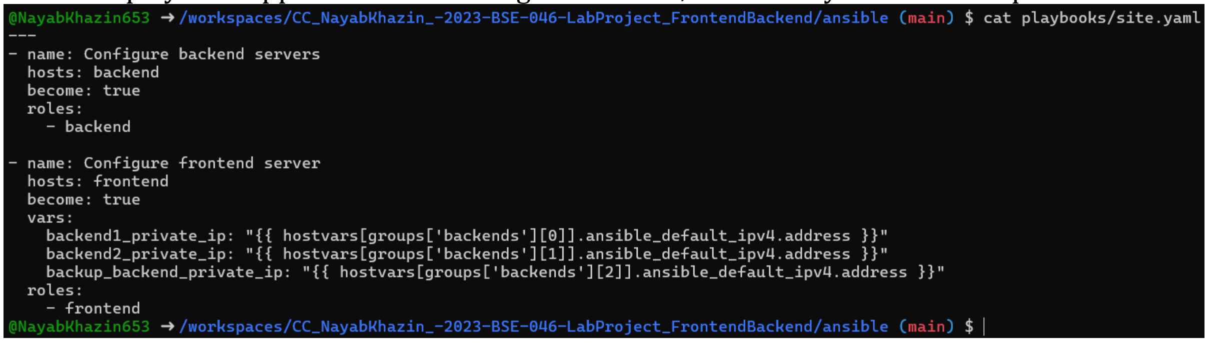


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat roles/frontend/handlers/main.yml
---
- name: Restart Nginx
  service:
    name: nginx
    state: restarted
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 21-C: Nginx Configuration Handlers)

7.5 Main Playbook (site.yaml)

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

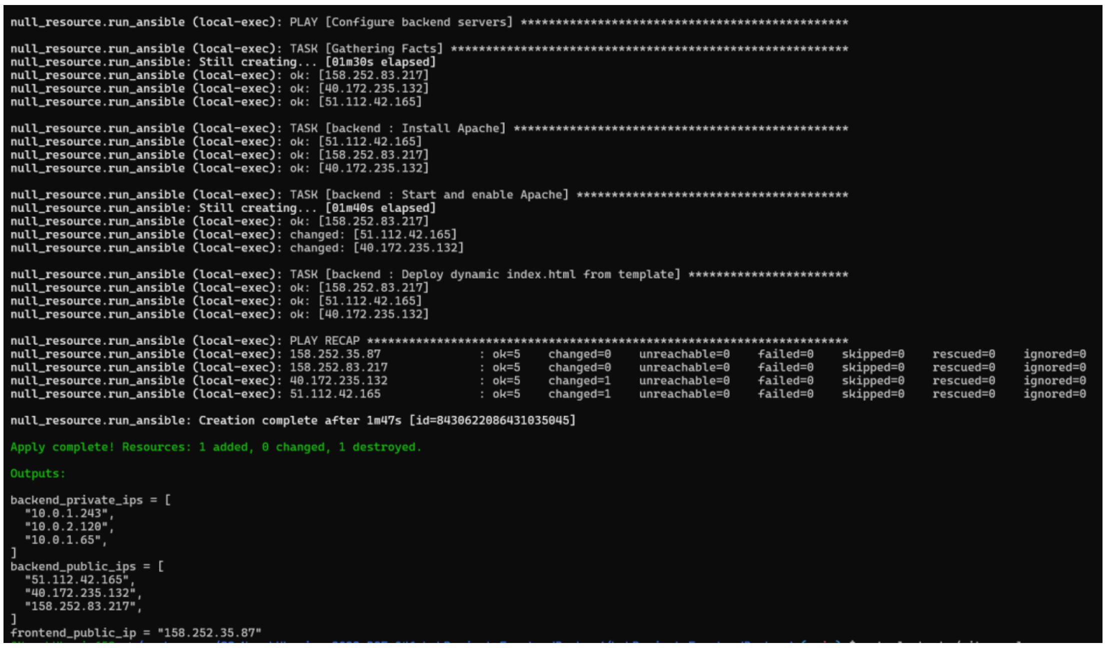


```
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ cat playbooks/site.yaml
---
- name: Configure backend servers
  hosts: backend
  become: true
  roles:
    - backend

- name: Configure frontend server
  hosts: frontend
  become: true
  vars:
    backend1_private_ip: "{{ hostvars[groups['backends'][0]].ansible_default_ipv4.address }}"
    backend2_private_ip: "{{ hostvars[groups['backends'][1]].ansible_default_ipv4.address }}"
    backup_backend_private_ip: "{{ hostvars[groups['backends'][2]].ansible_default_ipv4.address }}"
  roles:
    - frontend
@NayabKhazin653 ➜ /workspaces/CC_NayabKhazin_-2023-BSE-046-LabProject_FrontendBackend/ansible (main) $ |
```

(Figure 22: site.yaml)

8. Terraform Apply & Automation Execution

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

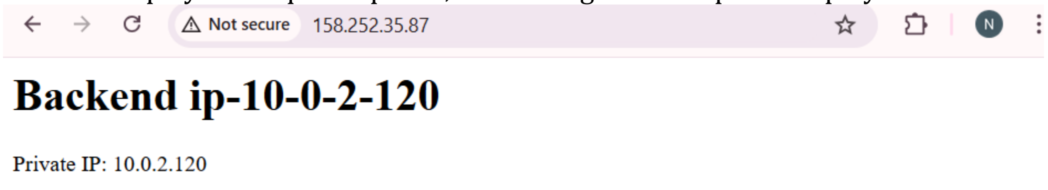


```
null_resource.run_ansible (local-exec): PLAY [Configure backend servers] ***********************************************

null_resource.run_ansible (local-exec): TASK [Gathering Facts] *********************************************************
null_resource.run_ansible: Still creating... [01m30s elapsed]
null_resource.run_ansible (local-exec): ok: [158.252.83.217]
null_resource.run_ansible (local-exec): ok: [40.172.235.132]
null_resource.run_ansible (local-exec): ok: [51.112.42.165]

null_resource.run_ansible (local-exec): TASK [backend : Install Apache] ************************************************
null_resource.run_ansible (local-exec): ok: [51.112.42.165]
null_resource.run_ansible (local-exec): ok: [158.252.83.217]
null_resource.run_ansible (local-exec): ok: [40.172.235.132]

null_resource.run_ansible (local-exec): TASK [backend : Start and enable Apache] ***************************************
null_resource.run_ansible: Still creating... [01m40s elapsed]
null_resource.run_ansible (local-exec): ok: [158.252.83.217]
null_resource.run_ansible (local-exec): changed: [51.112.42.165]
null_resource.run_ansible (local-exec): changed: [40.172.235.132]

null_resource.run_ansible (local-exec): TASK [backend : Deploy dynamic index.html from template] ***********************
null_resource.run_ansible (local-exec): ok: [158.252.83.217]
null_resource.run_ansible (local-exec): ok: [51.112.42.165]
null_resource.run_ansible (local-exec): ok: [40.172.235.132]

null_resource.run_ansible (local-exec): PLAY RECAP *********************************************************************
null_resource.run_ansible (local-exec): 158.252.35.87              : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
null_resource.run_ansible (local-exec): 158.252.83.217             : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
null_resource.run_ansible (local-exec): 40.172.235.132             : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
null_resource.run_ansible (local-exec): 51.112.42.165              : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

null_resource.run_ansible: Creation complete after 1m47s [id=8430622086431035045]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

backend_private_ips = [
  "10.0.1.243",
  "10.0.2.120",
  "10.0.1.65",
]
backend_public_ips = [
  "51.112.42.165",
  "40.172.235.132",
  "158.252.83.217",
]
frontend_public_ip = "158.252.35.87"
```

(Figure 23: Terraform Apply Output)

9. Backend Server Verification

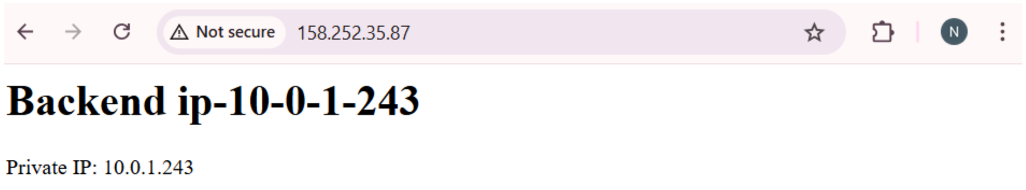
The following unlabeled screenshots show direct access to backend servers using their public IPs.
Each backend displays a unique response, confirming correct Apache deployment.



Backend ip-10-0-2-120

Private IP: 10.0.2.120

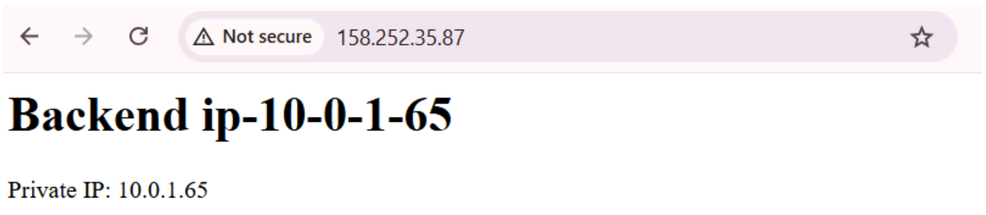
(Figure 24: Backend Server 1 Output)



Backend ip-10-0-1-243

Private IP: 10.0.1.243

(Figure 25: Backend Server 2 Output)



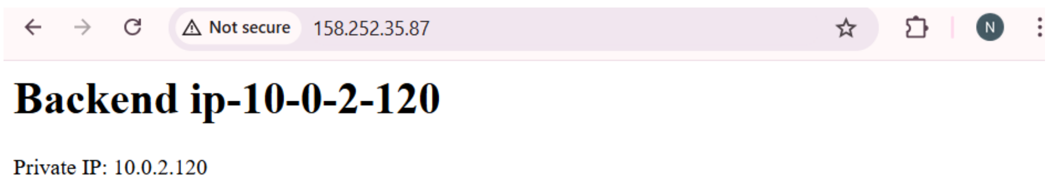
Backend ip-10-0-1-65

Private IP: 10.0.1.65

(Figure 26: Backend Server 3 Output)

10. Frontend Load Balancer Verification

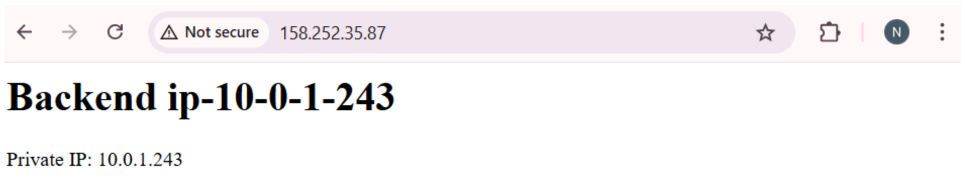
These unlabeled screenshots show frontend access via the Nginx load balancer public IP. Responses alternate between primary backend servers.


⚠ Not secure 158.252.35.87

Backend ip-10-0-2-120

Private IP: 10.0.2.120

(Figure 27: Frontend Response – Backend A)


⚠ Not secure 158.252.35.87

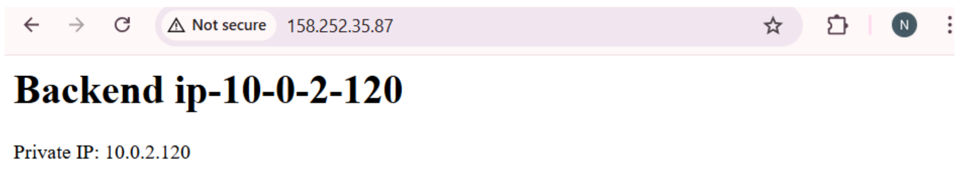
Backend ip-10-0-1-243

Private IP: 10.0.1.243

(Figure 28: Frontend Response – Backend B)

11. Load Balancing Behaviour Validation

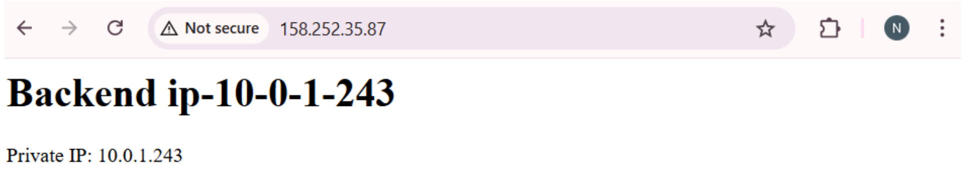
Repeated refreshes confirm round-robin behavior between the two primary backend servers.


⚠ Not secure 158.252.35.87

Backend ip-10-0-2-120

Private IP: 10.0.2.120

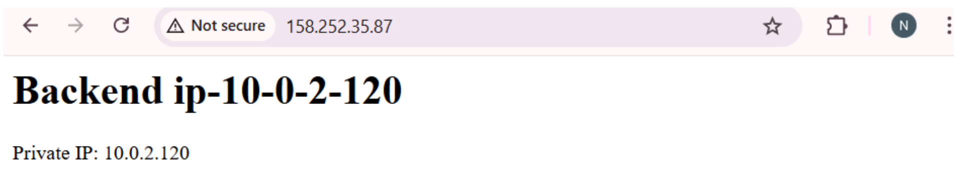
(Figure 29-A: Refresh-1)


⚠ Not secure 158.252.35.87

Backend ip-10-0-1-243

Private IP: 10.0.1.243

(Figure 29-B: Refresh-2)

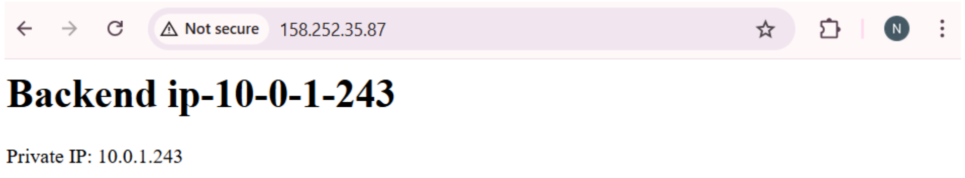


Not secure 158.252.35.87

Backend ip-10-0-2-120

Private IP: 10.0.2.120

(Figure 29-C: Refresh-3)



Not secure 158.252.35.87

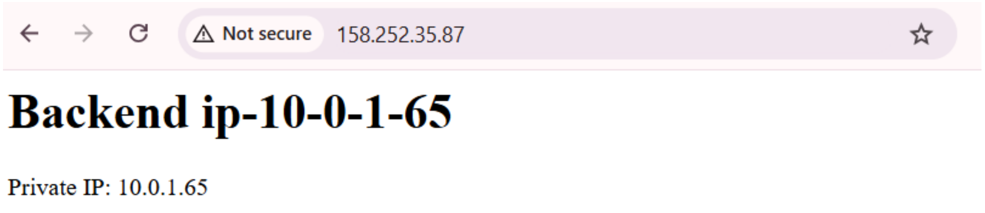
Backend ip-10-0-1-243

Private IP: 10.0.1.243

(Figure 29-D: Refresh-4)

12. Failover Testing (Backup Backend)

Primary backend services were stopped to validate failover. Traffic was successfully routed to the backup backend.



Not secure 158.252.35.87

Backend ip-10-0-1-65

Private IP: 10.0.1.65

(Figure 30: 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.