Fatima Jinnah
Women University

Name: Reena Qureshi

Roll Number: 2023-BSE-052

Section: V-B

Course: Cloud Computing

Lab Project Report

Contents

Executive Summary

This lab project demonstrates the design and implementation of a highly available multi-tier web infrastructure on AWS using Infrastructure as Code (IaC) principles. The project successfully integrates Terraform for infrastructure provisioning and Ansible for automated configuration management, showcasing modern DevOps practices.

1. **Project Overview**

1.1 Objectives

The primary objective of this project was to demonstrate proficiency in:

- Designing multi-tier AWS infrastructure using Terraform
- Implementing configuration management using Ansible roles
- Configuring high-availability load balancing with Nginx
- Integrating IaC tools for end-to-end automation

1.2 Learning Outcomes Achieved

- **Infrastructure as Code:** Translated architecture requirements into Terraform configurations
- **Configuration Management:** Separated concerns using Ansible role-based structure
- **High Availability:** Implemented active-backup load balancing pattern
- **Automation:** Achieved zero-touch deployment with integrated Terraform-Ansible workflow
- **Best Practices:** Followed industry standards for code organization and documentation

1.3 Technologies Used

Technology	Version	Purpose
Terraform	~> 5.0	Infrastructure provisioning
Ansible	2.15.12	Configuration management
AWS	-	Cloud infrastructure provider
Amazon Linux 2	Latest	Operating system for EC2 instances
Nginx	1.x	Load balancer and reverse proxy
Apache HTTPD	2.4.x	Backend web servers

2. Architecture Design

2.1 Network Architecture

VPC Configuration

- **VPC CIDR Block:** 10.0.0.0/16
- **Public Subnet:** 10.0.15.0/24
- **Availability Zone:** us-east-1a
- **Internet Gateway:** Enabled for public internet access
- **Route Table:** Default route (0.0.0.0/0) to Internet Gateway

Security Groups

Web Security Group:

- **Inbound Rules:**
  - SSH (port 22): Restricted to developer's IP address
  - HTTP (port 80): Open to internet (0.0.0.0/0)
  - Internal VPC traffic: All protocols allowed within VPC CIDR
- **Outbound Rules:**
  - All traffic allowed (0.0.0.0/0)

2.2 Compute Architecture

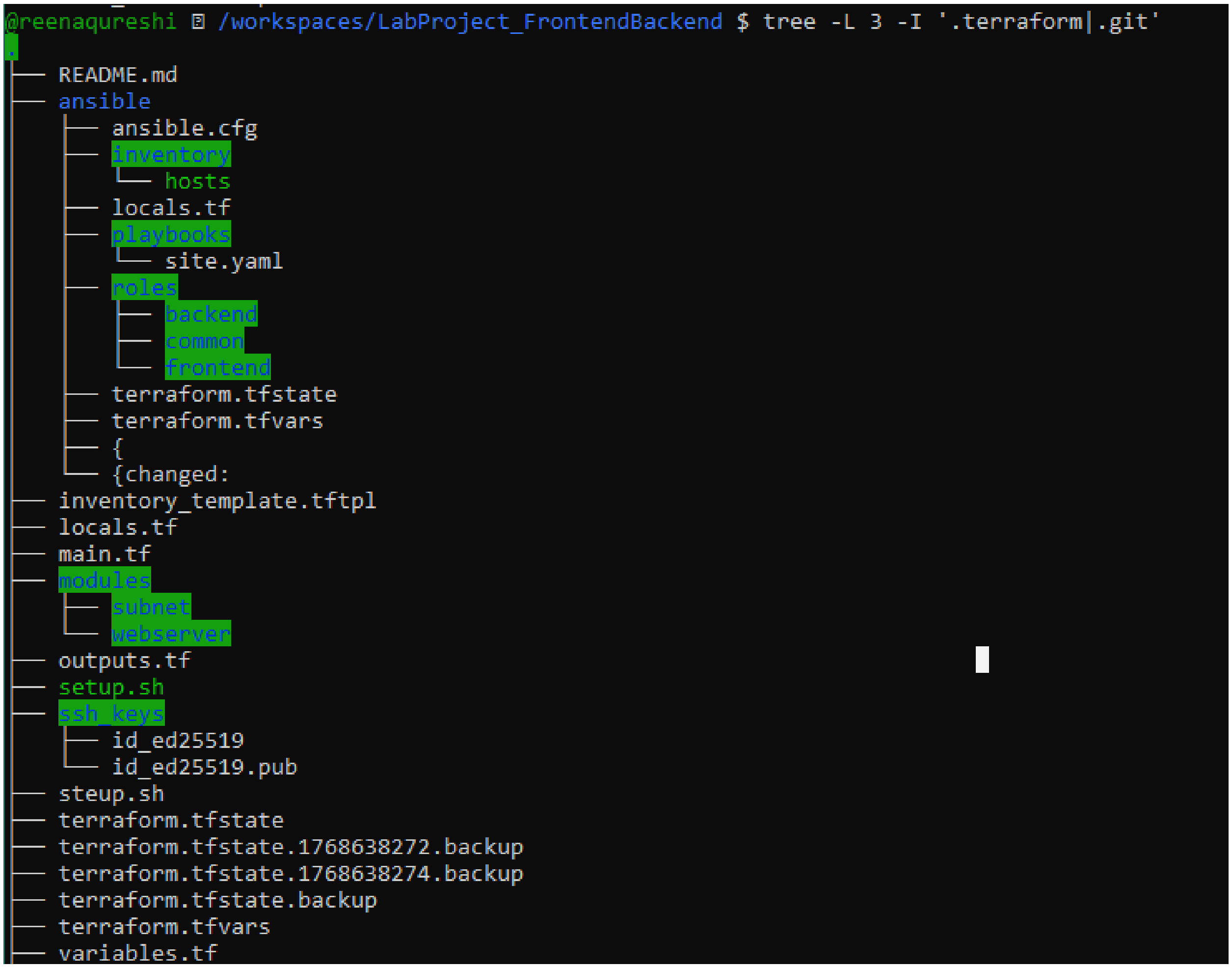
Instance Configuration

Instance Type	Count	Role	Purpose
t2.micro	1	Frontend	Nginx load balancer
t2.micro	3	Backend	Apache HTTPD web servers

Load Balancing Strategy:

- **Normal Operation:** Round-robin between Backend-0 and Backend-1
- **Failover Mode:** Backend-2 activates when primaries are unavailable
- **Health Checks:** Nginx automatically detects failed backends

2.3 File Structure



```
@reenaqureshi ⍰ /workspaces/LabProject_FrontendBackend $ tree -L 3 -I '.terraform|.git'
.
├── README.md
├── ansible
│   ├── ansible.cfg
│   ├── inventory
│   │   └── hosts
│   ├── locals.tf
│   ├── playbooks
│   │   └── site.yaml
│   ├── roles
│   │   ├── backend
│   │   ├── common
│   │   └── frontend
│   ├── terraform.tfstate
│   ├── terraform.tfvars
│   ├── {
│   └── {changed:
├── inventory_template.tftpl
├── locals.tf
├── main.tf
├── modules
│   ├── subnet
│   └── webserver
├── outputs.tf
├── setup.sh
├── ssh_keys
│   ├── id_ed25519
│   └── id_ed25519.pub
├── steup.sh
├── terraform.tfstate
├── terraform.tfstate.1768638272.backup
├── terraform.tfstate.1768638274.backup
├── terraform.tfstate.backup
├── terraform.tfvars
├── variables.tf
```

3. Implementation Details

3.1 Terraform Infrastructure

Key Components Implemented

1. VPC and Networking (main.tf)

hcl

- VPC with DNS support enabled

- Public subnet with auto-assign public IP

- Internet Gateway for internet connectivity

- Route table with default route to IGW

- Security group with appropriate firewall rules

2. EC2 Instances

hcl

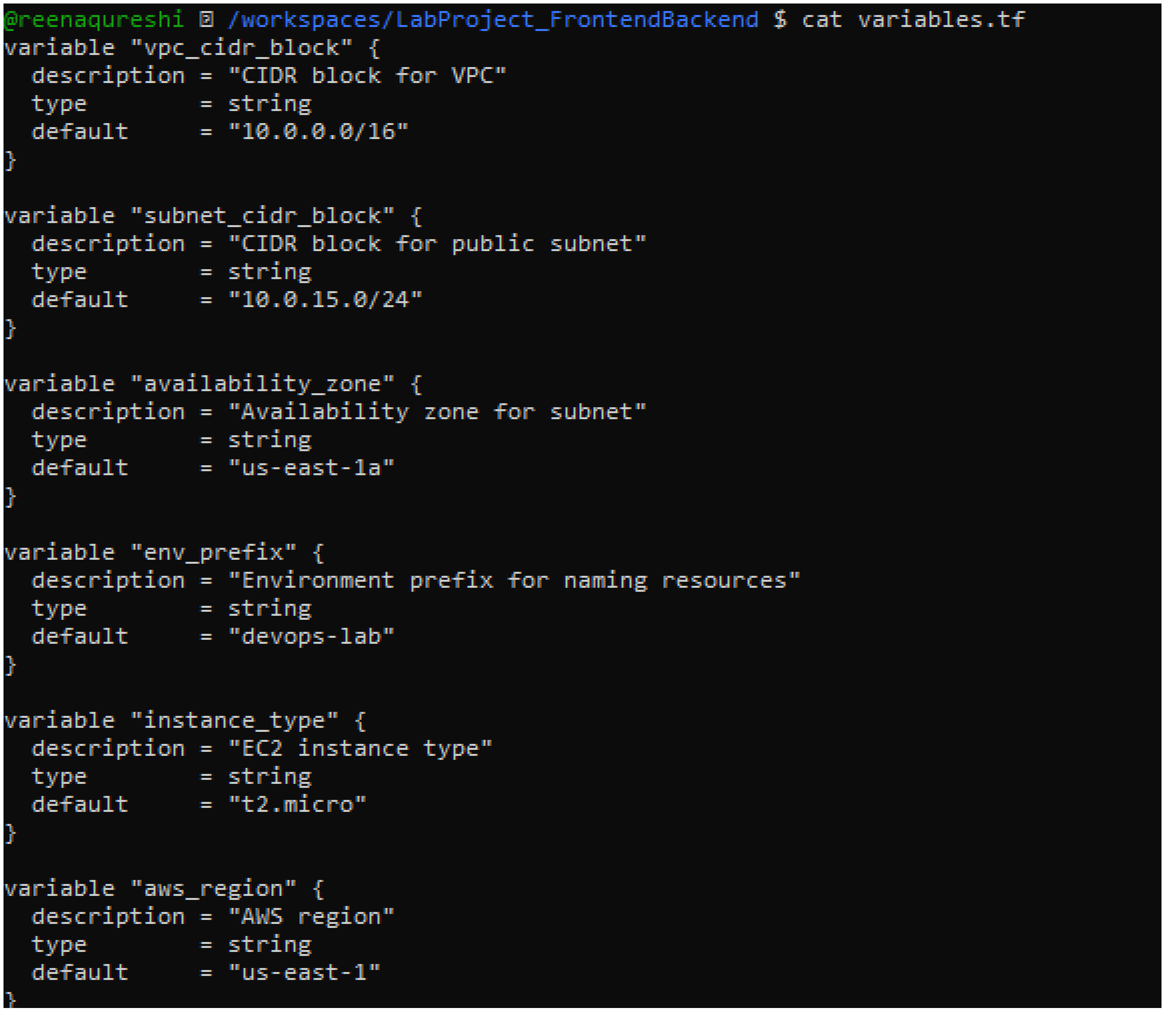
- Frontend instance: Single Nginx load balancer
- Backend instances: 3 Apache HTTPD servers (using count)
- SSH key pair: Automated key deployment
- Tags: Meaningful naming for resource identification

3. Automation Integration

hcl

- Inventory generation: Dynamic Ansible inventory from Terraform
- Wait mechanism: Ensures EC2 instances are ready before configuration
- Automatic execution: null_resource triggers Ansible playbook

Variables Configuration



```
@reenaqureshi ⍰ /workspaces/LabProject_FrontendBackend $ cat variables.tf
variable "vpc_cidr_block" {
  description = "CIDR block for VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "subnet_cidr_block" {
  description = "CIDR block for public subnet"
  type        = string
  default     = "10.0.15.0/24"
}

variable "availability_zone" {
  description = "Availability zone for subnet"
  type        = string
  default     = "us-east-1a"
}

variable "env_prefix" {
  description = "Environment prefix for naming resources"
  type        = string
  default     = "devops-lab"
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}

variable "aws_region" {
  description = "AWS region"
  type        = string
  default     = "us-east-1"
}
```

3.2 Ansible Configuration Management

Backend Role (roles/backend/)

Purpose: Configure Apache HTTPD web servers with distinct content

Tasks Implemented:

1. Install Apache HTTPD package
2. Ensure HTTPD service is running and enabled
3. Deploy custom HTML page using Jinja2 template
4. Configure service handlers for automatic restart

Template Features:

- Displays backend server hostname
- Shows private IP address
- Unique identification for each backend
- Professional HTML/CSS styling

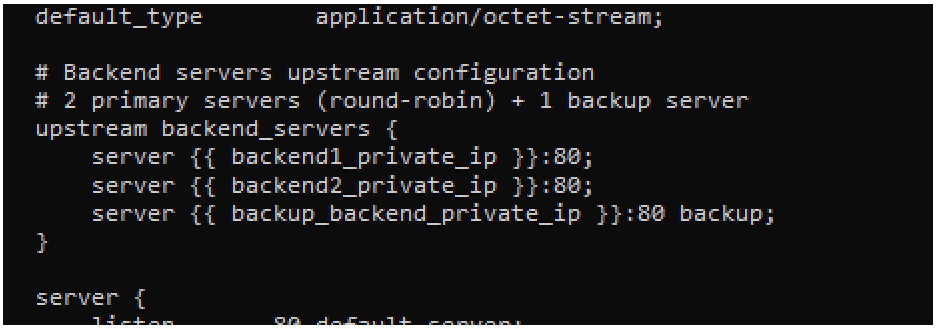
Frontend Role (roles/frontend/)

Purpose: Configure Nginx as reverse proxy load balancer

Tasks Implemented:

1. Install Nginx via amazon-linux-extras
2. Start and enable Nginx service
3. Deploy custom Nginx configuration
4. Configure upstream with HA settings

Nginx Upstream Configuration:



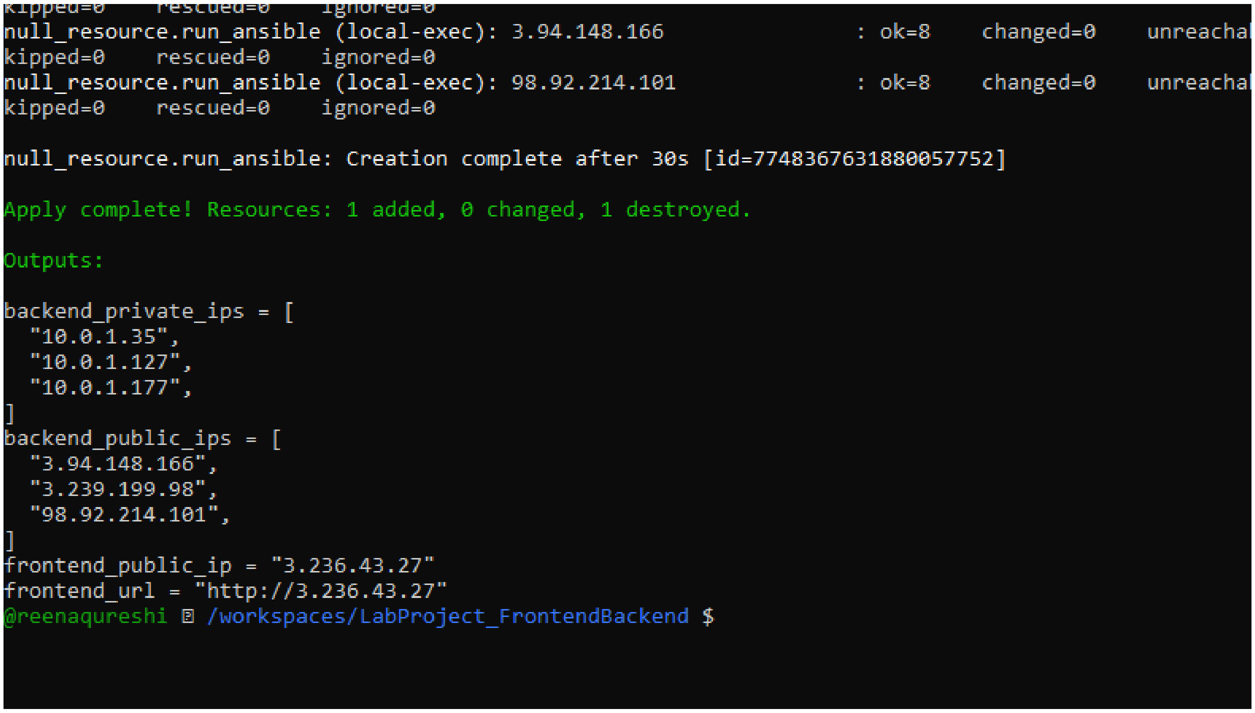
```
  default_type        application/octet-stream;

  # Backend servers upstream configuration
  # 2 primary servers (round-robin) + 1 backup server
  upstream backend_servers {
      server {{ backend1_private_ip }}:80;
      server {{ backend2_private_ip }}:80;
      server {{ backup_backend_private_ip }}:80 backup;
  }

  server {
      listen       80 default_server;
```

Main Playbook (playbooks/site.yaml)

Playbook Structure:

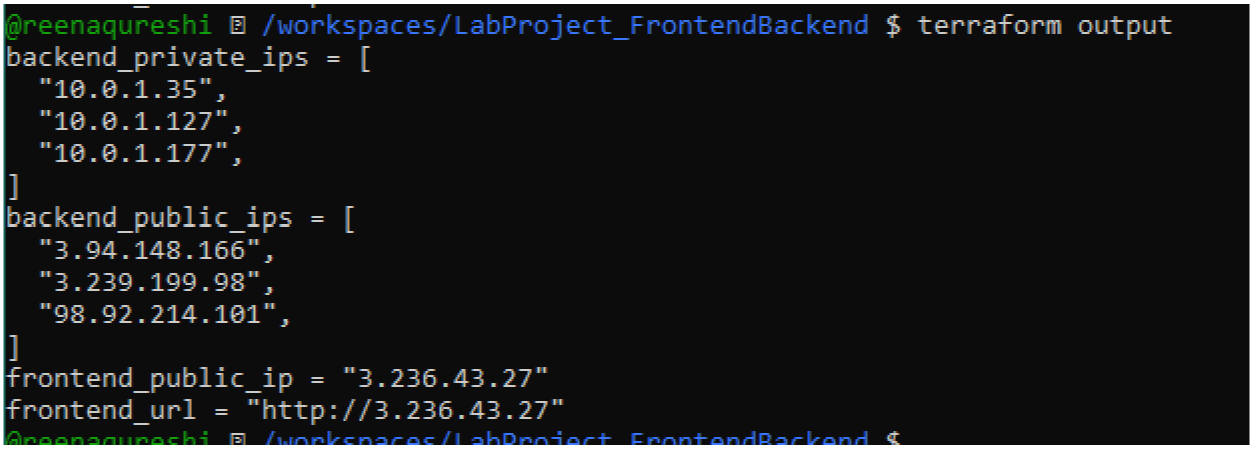
1. **Play 1:** Ensure Python 3 availability on all hosts
2. **Play 2:** Configure backend HTTPD servers (role: backend)
3. **Play 3:** Gather backend facts for frontend configuration
4. **Play 4:** Configure frontend Nginx (role: frontend) with dynamic backend IPs

4. Testing & Validation



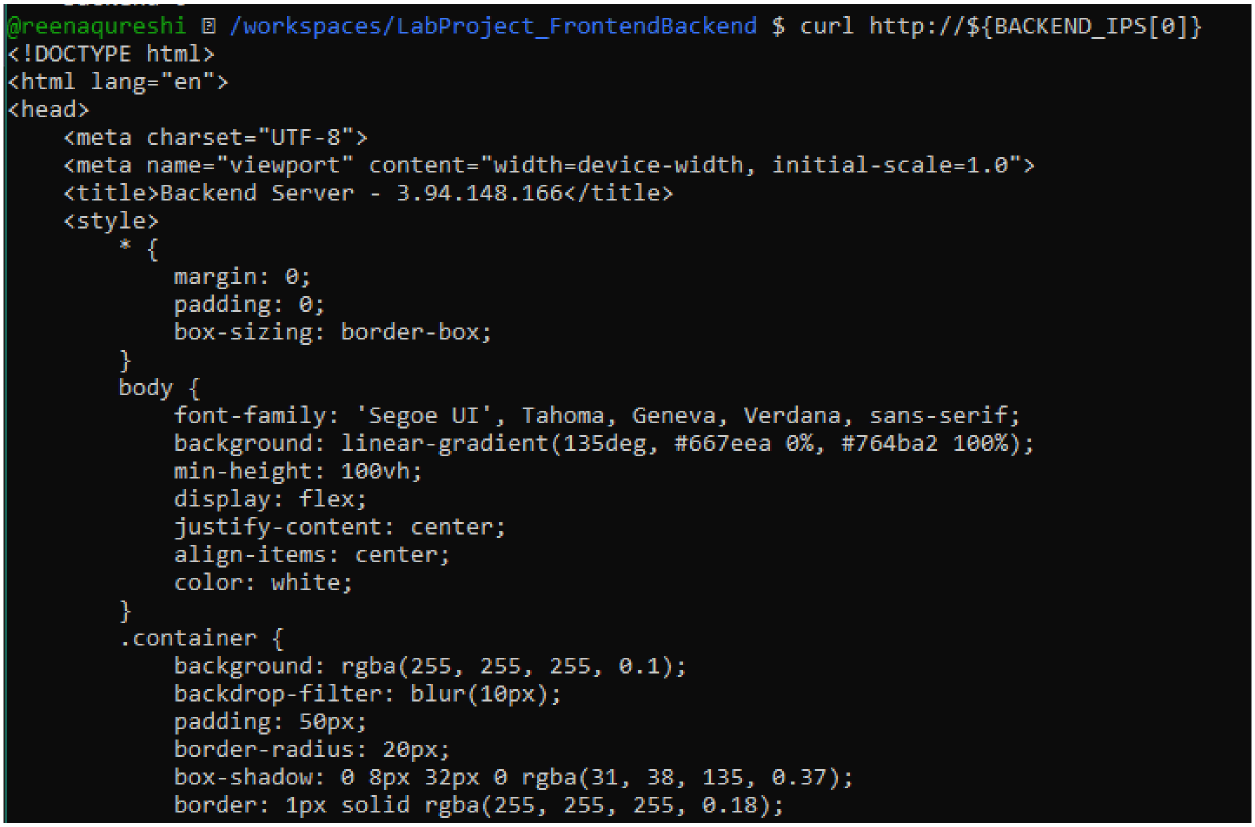
```
kipped=0    rescued=0    ignored=0
null_resource.run_ansible (local-exec): 3.94.148.166              : ok=8    changed=0    unreachal
kipped=0    rescued=0    ignored=0
null_resource.run_ansible (local-exec): 98.92.214.101             : ok=8    changed=0    unreachal
kipped=0    rescued=0    ignored=0

null_resource.run_ansible: Creation complete after 30s [id=7748367631880057752]

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

Outputs:

backend_private_ips = [
  "10.0.1.35",
  "10.0.1.127",
  "10.0.1.177",
]
backend_public_ips = [
  "3.94.148.166",
  "3.239.199.98",
  "98.92.214.101",
]
frontend_public_ip = "3.236.43.27"
frontend_url = "http://3.236.43.27"
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $
```

Terraform Outputs:

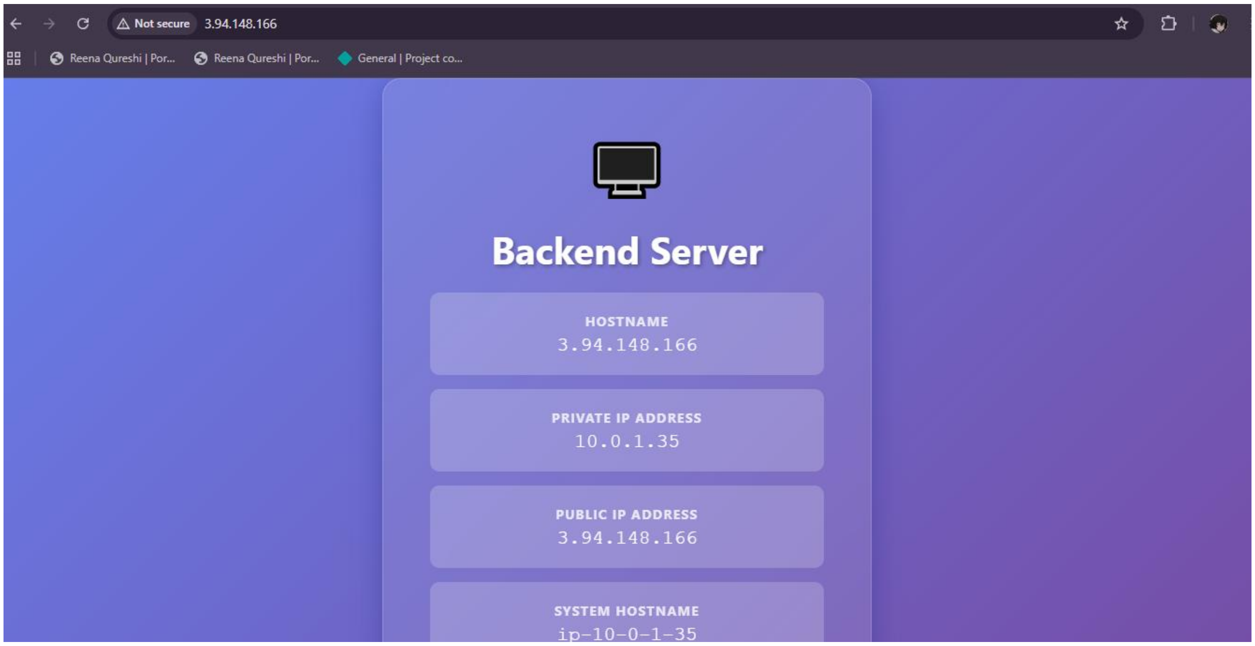


```
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $ terraform output
backend_private_ips = [
  "10.0.1.35",
  "10.0.1.127",
  "10.0.1.177",
]
backend_public_ips = [
  "3.94.148.166",
  "3.239.199.98",
  "98.92.214.101",
]
frontend_public_ip = "3.236.43.27"
frontend_url = "http://3.236.43.27"
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $
```

Individual Backend Testing:



```
@reenaqureshi ➜ /workspaces/LabProject_FrontendBackend $ curl http://${BACKEND_IPS[0]}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Backend Server - 3.94.148.166</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            color: white;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            padding: 50px;
            border-radius: 20px;
            box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
            border: 1px solid rgba(255, 255, 255, 0.18);
```



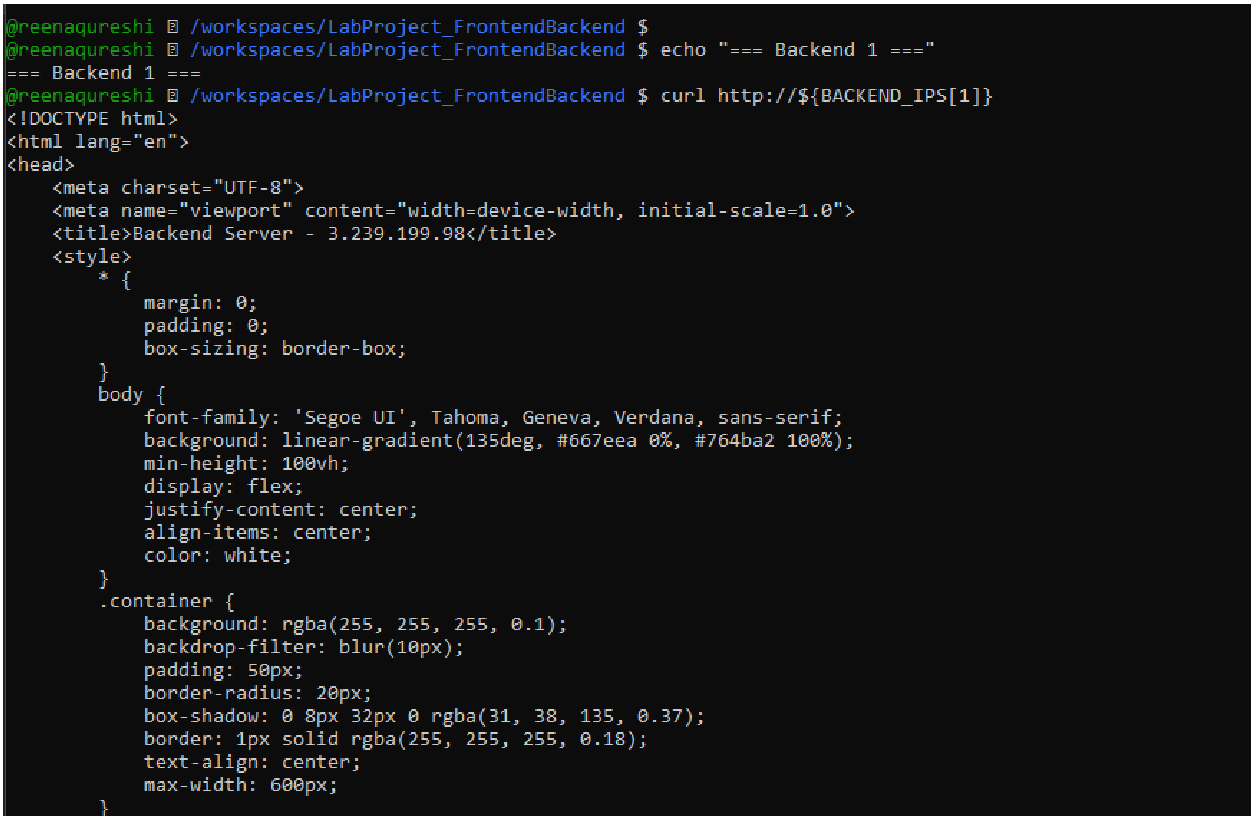
Backend Server

HOSTNAME
3.94.148.166

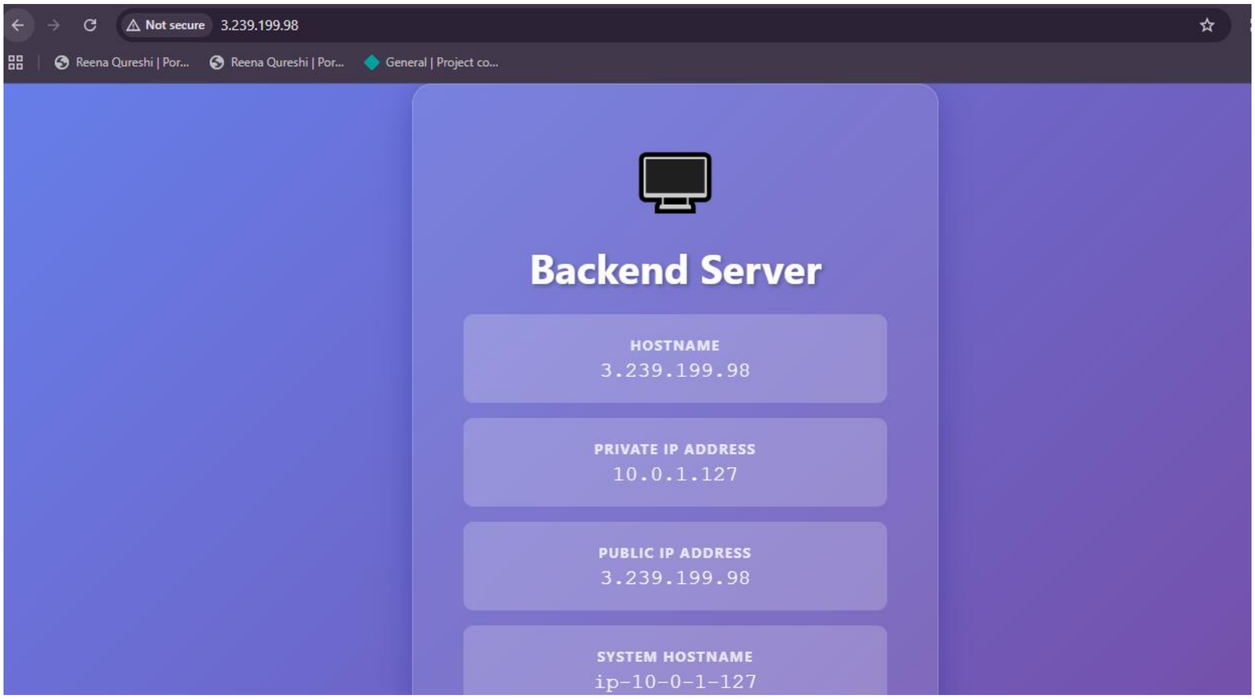
PRIVATE IP ADDRESS
10.0.1.35

PUBLIC IP ADDRESS
3.94.148.166

SYSTEM HOSTNAME
ip-10-0-1-35



```
@reenaqureshi ➜ /workspaces/LabProject_FrontendBackend $
@reenaqureshi ➜ /workspaces/LabProject_FrontendBackend $ echo "=== Backend 1 ==="
=== Backend 1 ===
@reenaqureshi ➜ /workspaces/LabProject_FrontendBackend $ curl http://${BACKEND_IPS[1]}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Backend Server - 3.239.199.98</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            color: white;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            padding: 50px;
            border-radius: 20px;
            box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
            border: 1px solid rgba(255, 255, 255, 0.18);
            text-align: center;
            max-width: 600px;
        }
```



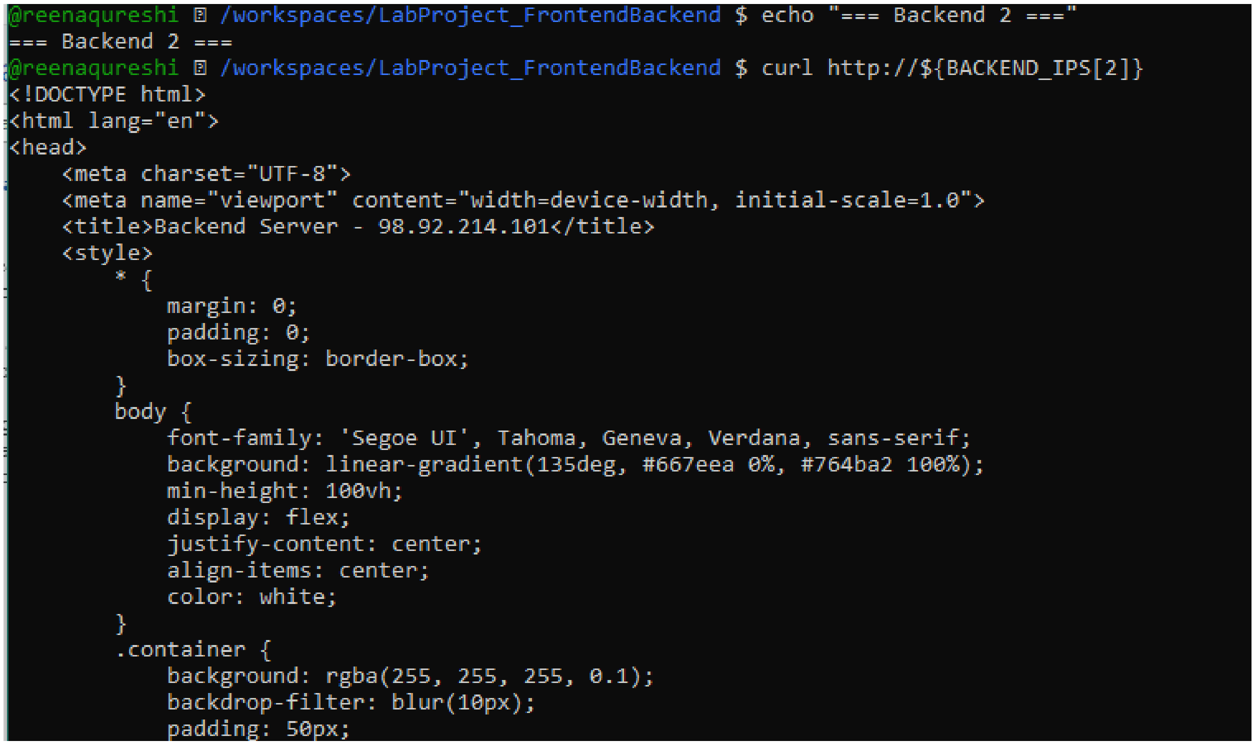
Backend Server

HOSTNAME
3.239.199.98

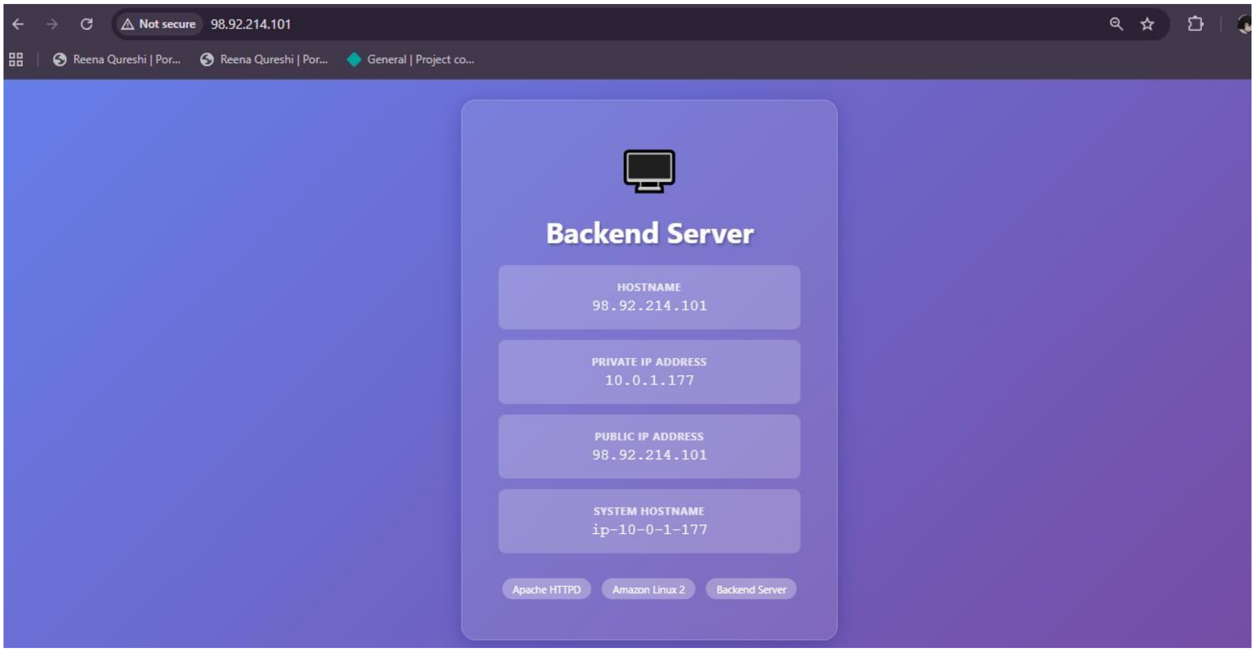
PRIVATE IP ADDRESS
10.0.1.127

PUBLIC IP ADDRESS
3.239.199.98

SYSTEM HOSTNAME
ip-10-0-1-127



```
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $ echo "=== Backend 2 ==="
=== Backend 2 ===
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $ curl http://${BACKEND_IPS[2]}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Backend Server - 98.92.214.101</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            color: white;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            padding: 50px;
```



Backend Server

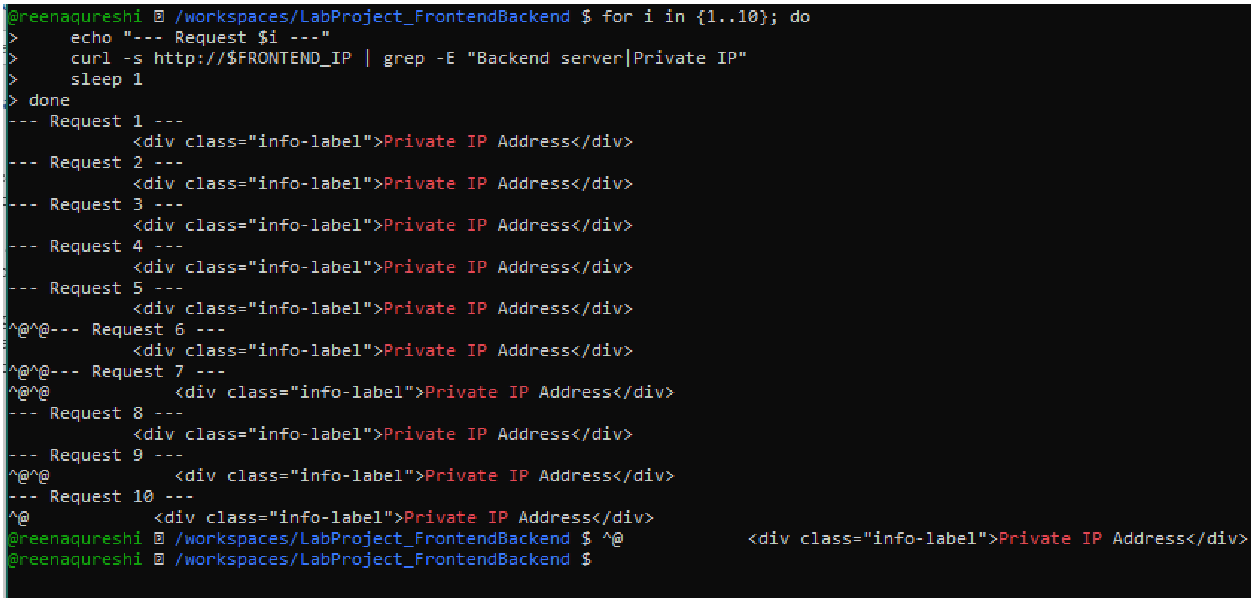
HOSTNAME
98.92.214.101

PRIVATE IP ADDRESS
10.0.1.177

PUBLIC IP ADDRESS
98.92.214.101

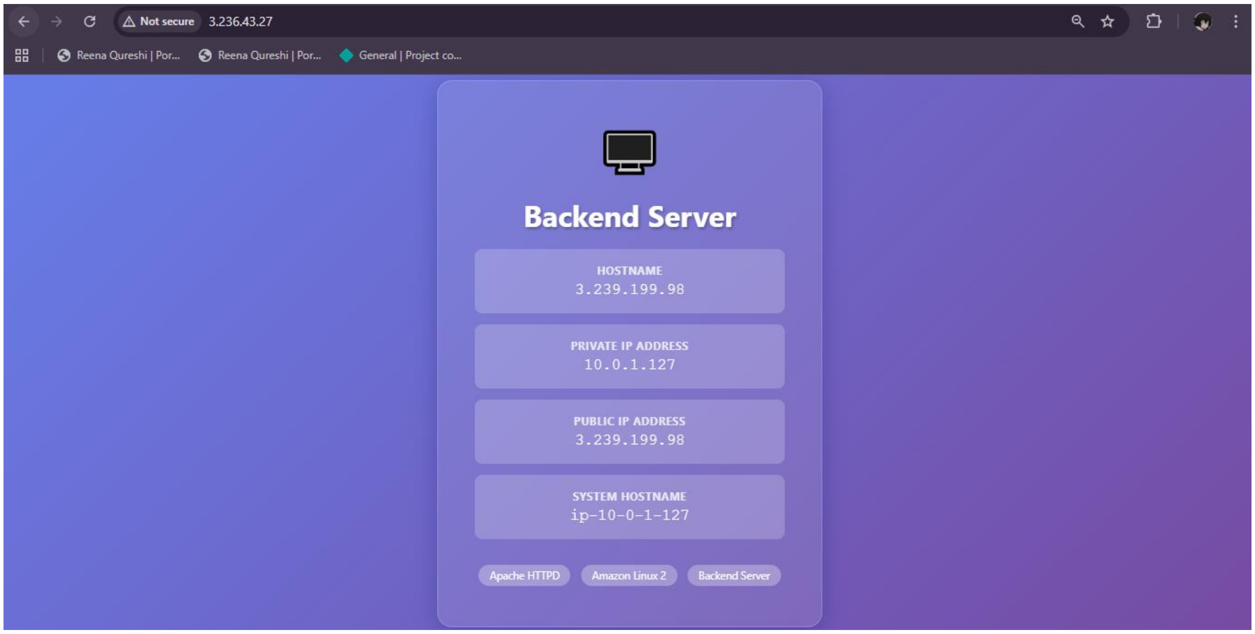
SYSTEM HOSTNAME
ip-10-0-1-177

Load Balancer Testing (Normal Operation)



```
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $ for i in {1..10}; do
>     echo "--- Request $i ---"
>     curl -s http://$FRONTEND_IP | grep -E "Backend server|Private IP"
>     sleep 1
> done
--- Request 1 ---
            <div class="info-label">Private IP Address</div>
--- Request 2 ---
            <div class="info-label">Private IP Address</div>
--- Request 3 ---
            <div class="info-label">Private IP Address</div>
--- Request 4 ---
            <div class="info-label">Private IP Address</div>
--- Request 5 ---
            <div class="info-label">Private IP Address</div>
^@^@--- Request 6 ---
            <div class="info-label">Private IP Address</div>
^@^@--- Request 7 ---
^@^@            <div class="info-label">Private IP Address</div>
--- Request 8 ---
            <div class="info-label">Private IP Address</div>
--- Request 9 ---
^@^@            <div class="info-label">Private IP Address</div>
--- Request 10 ---
^@            <div class="info-label">Private IP Address</div>
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $ ^@           <div class="info-label">Private IP Address</div>
@reenaqureshi □ /workspaces/LabProject_FrontendBackend $
```

Browser Testing:



Backend Server

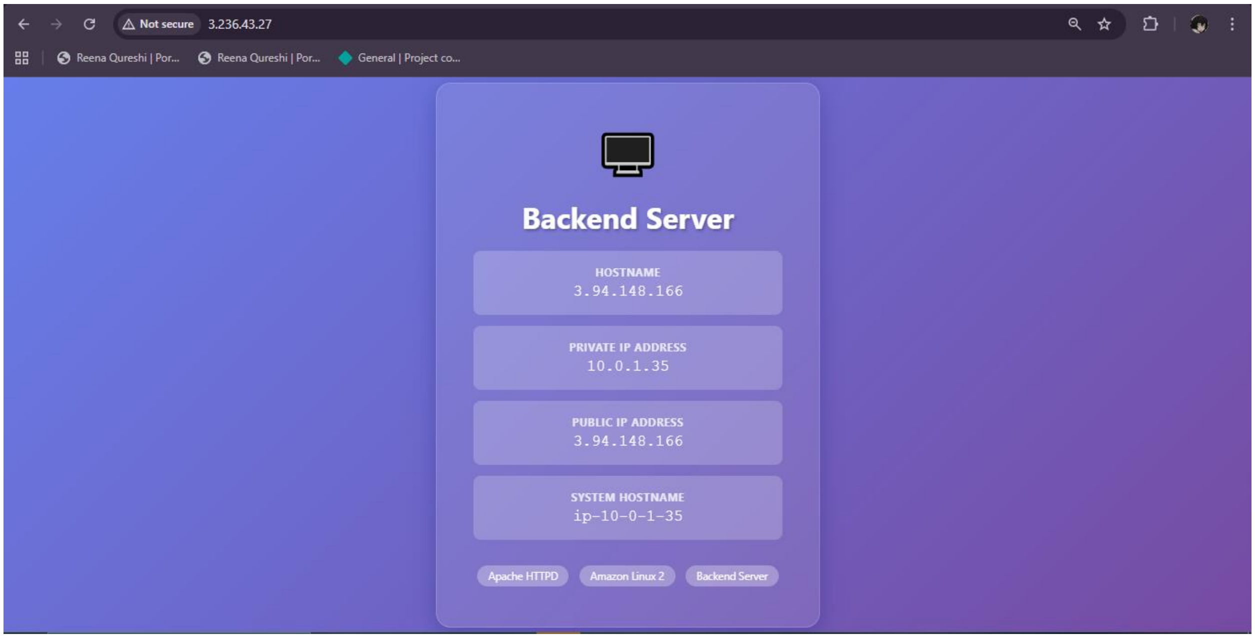
HOSTNAME
3.239.199.98

PRIVATE IP ADDRESS
10.0.1.127

PUBLIC IP ADDRESS
3.239.199.98

SYSTEM HOSTNAME
ip-10-0-1-127

After Refreshing it few times:



Backend Server

HOSTNAME
3.94.148.166

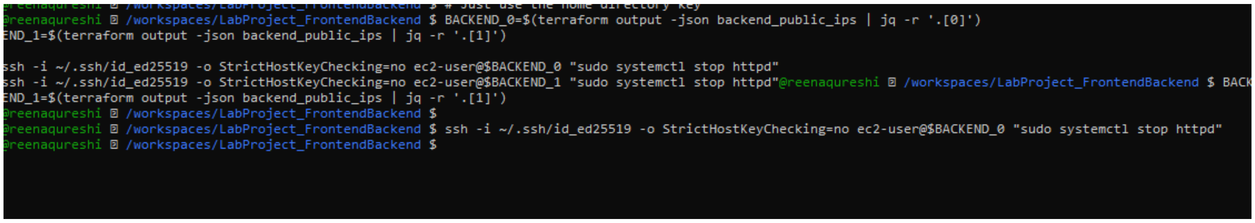
PRIVATE IP ADDRESS
10.0.1.35

PUBLIC IP ADDRESS
3.94.148.166

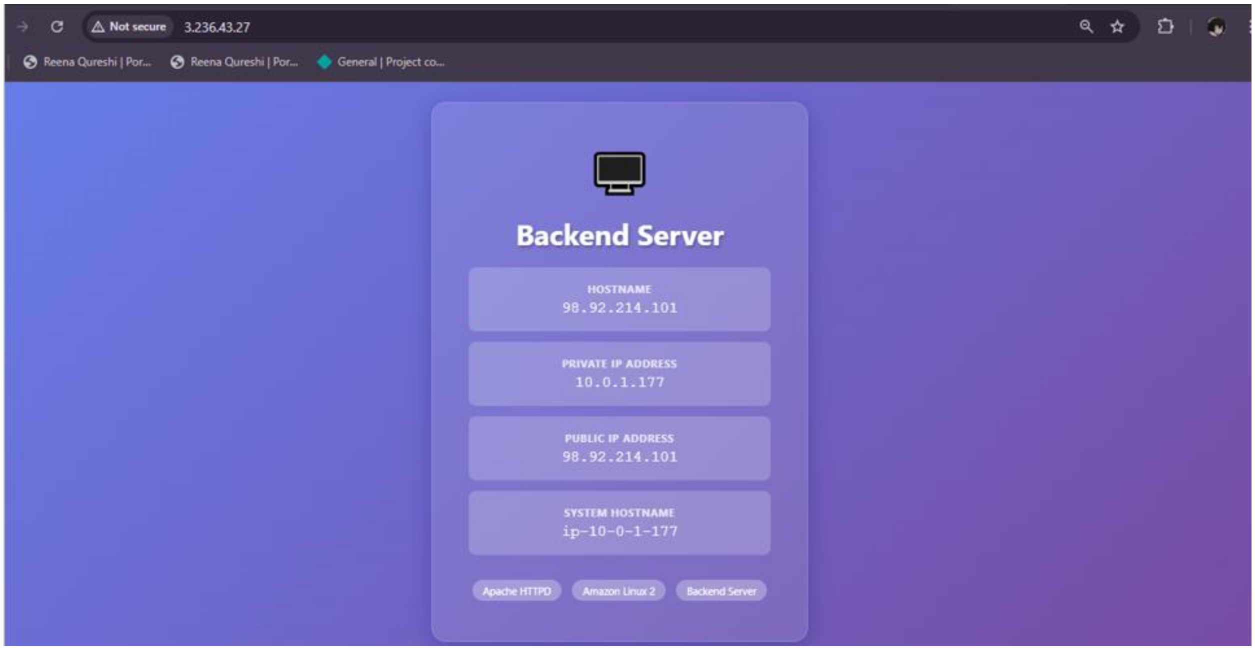
SYSTEM HOSTNAME
ip-10-0-1-35

Backup Server Testing

Stopping Backend 0 and 1 server to use the backup server:



```
@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $ # Just use the home directory key
@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $ BACKEND_0=$(terraform output -json backend_public_ips | jq -r '.[0]')
END_1=$(terraform output -json backend_public_ips | jq -r '.[1]')

ssh -i ~/.ssh/id_ed25519 -o StrictHostKeyChecking=no ec2-user@$BACKEND_0 "sudo systemctl stop httpd"
ssh -i ~/.ssh/id_ed25519 -o StrictHostKeyChecking=no ec2-user@$BACKEND_1 "sudo systemctl stop httpd"@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $ BACK
END_1=$(terraform output -json backend_public_ips | jq -r '.[1]')
@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $
@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $ ssh -i ~/.ssh/id_ed25519 -o StrictHostKeyChecking=no ec2-user@$BACKEND_0 "sudo systemctl stop httpd"
@reenaqureshi ⊡ /workspaces/LabProject_FrontendBackend $
```



Backend Server

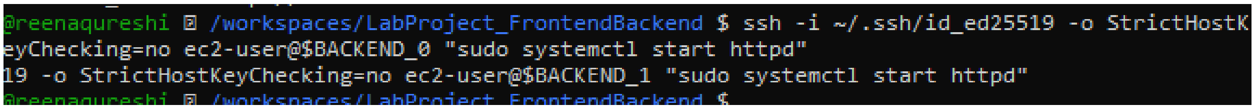
HOSTNAME
98.92.214.101

PRIVATE IP ADDRESS
10.0.1.177

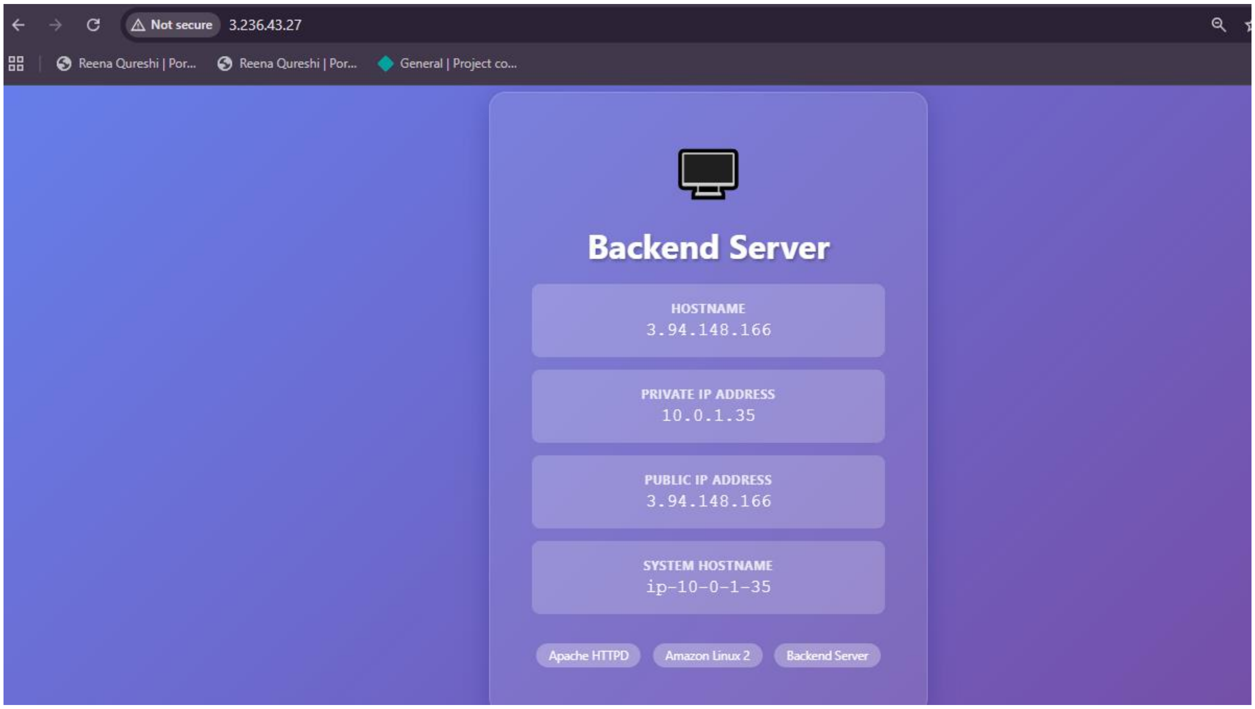
PUBLIC IP ADDRESS
98.92.214.101

SYSTEM HOSTNAME
ip-10-0-1-177

Restarted the backend servers:



```
@reenaqureshi ⊠ /workspaces/LabProject_FrontendBackend $ ssh -i ~/.ssh/id_ed25519 -o StrictHostK
eyChecking=no ec2-user@$BACKEND_0 "sudo systemctl start httpd"
19 -o StrictHostKeyChecking=no ec2-user@$BACKEND_1 "sudo systemctl start httpd"
@reenaqureshi ⊠ /workspaces/LabProject_FrontendBackend $
```



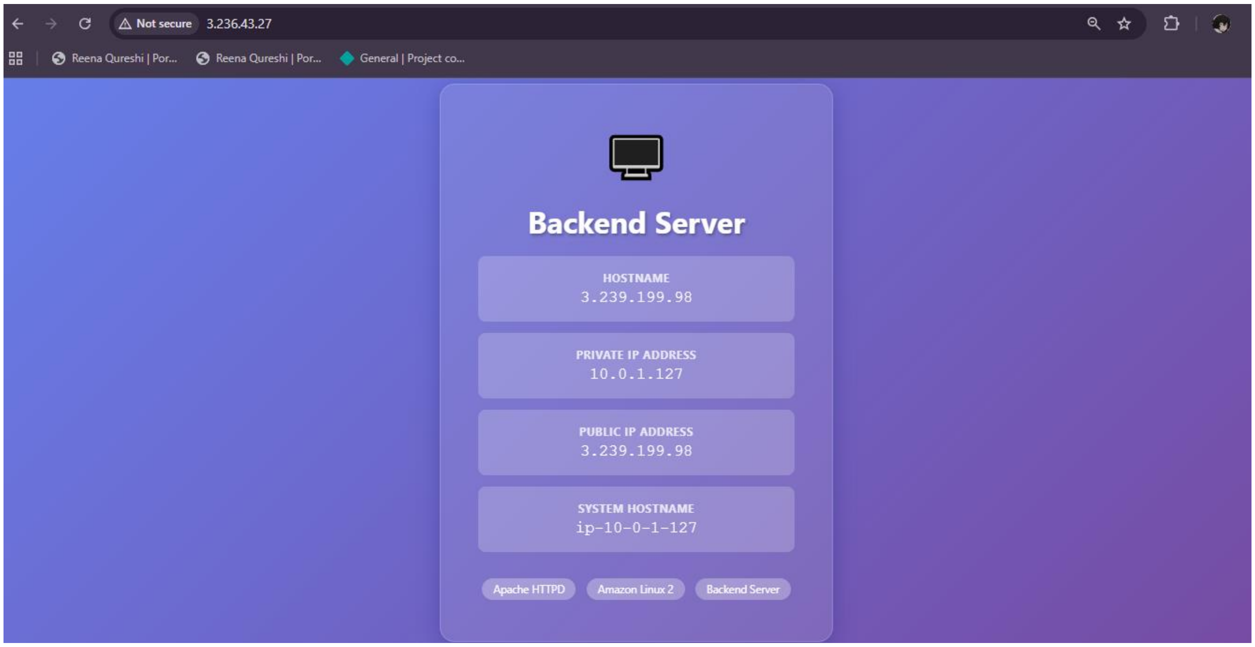
Backend Server

HOSTNAME
3.94.148.166

PRIVATE IP ADDRESS
10.0.1.35

PUBLIC IP ADDRESS
3.94.148.166

SYSTEM HOSTNAME
ip-10-0-1-35



Backend Server

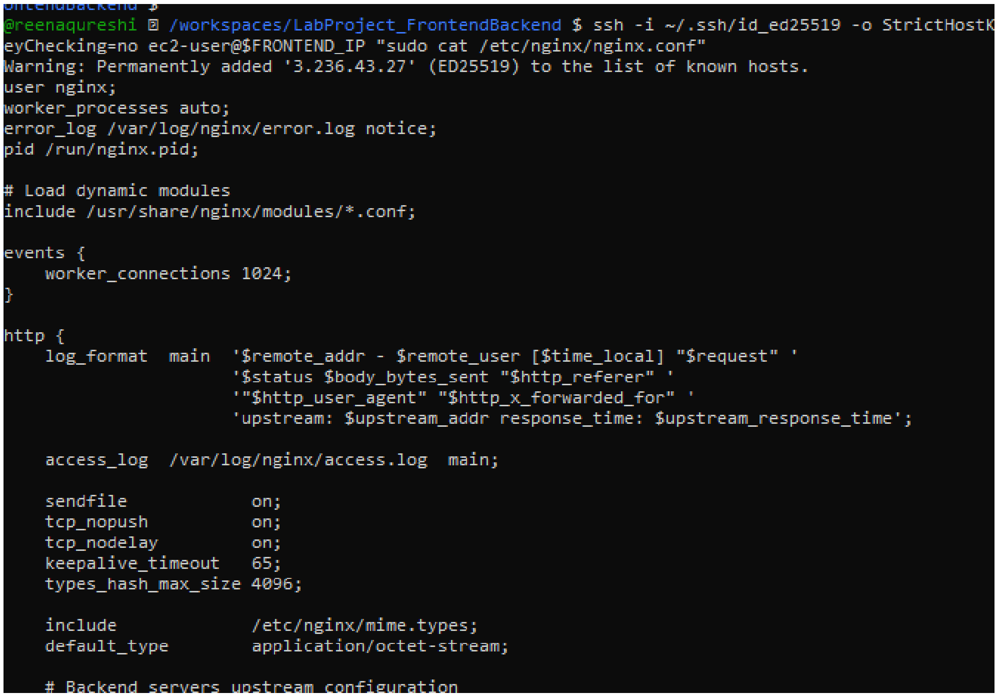
HOSTNAME
3.239.199.98

PRIVATE IP ADDRESS
10.0.1.127

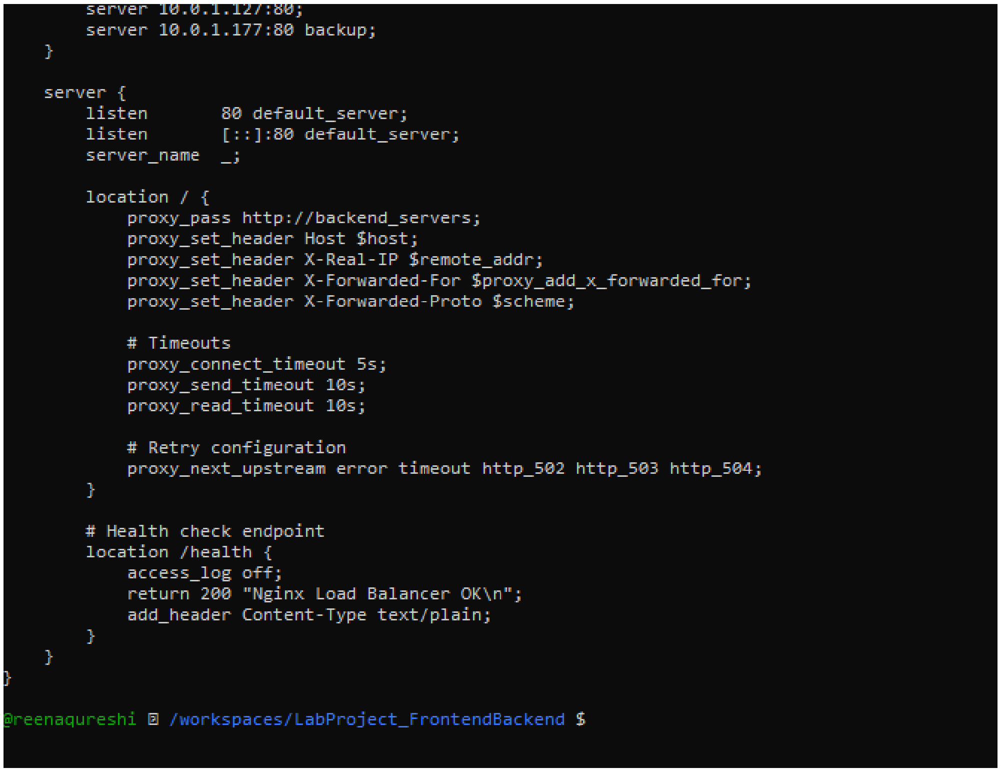
PUBLIC IP ADDRESS
3.239.199.98

SYSTEM HOSTNAME
ip-10-0-1-127

Nginx Configuration

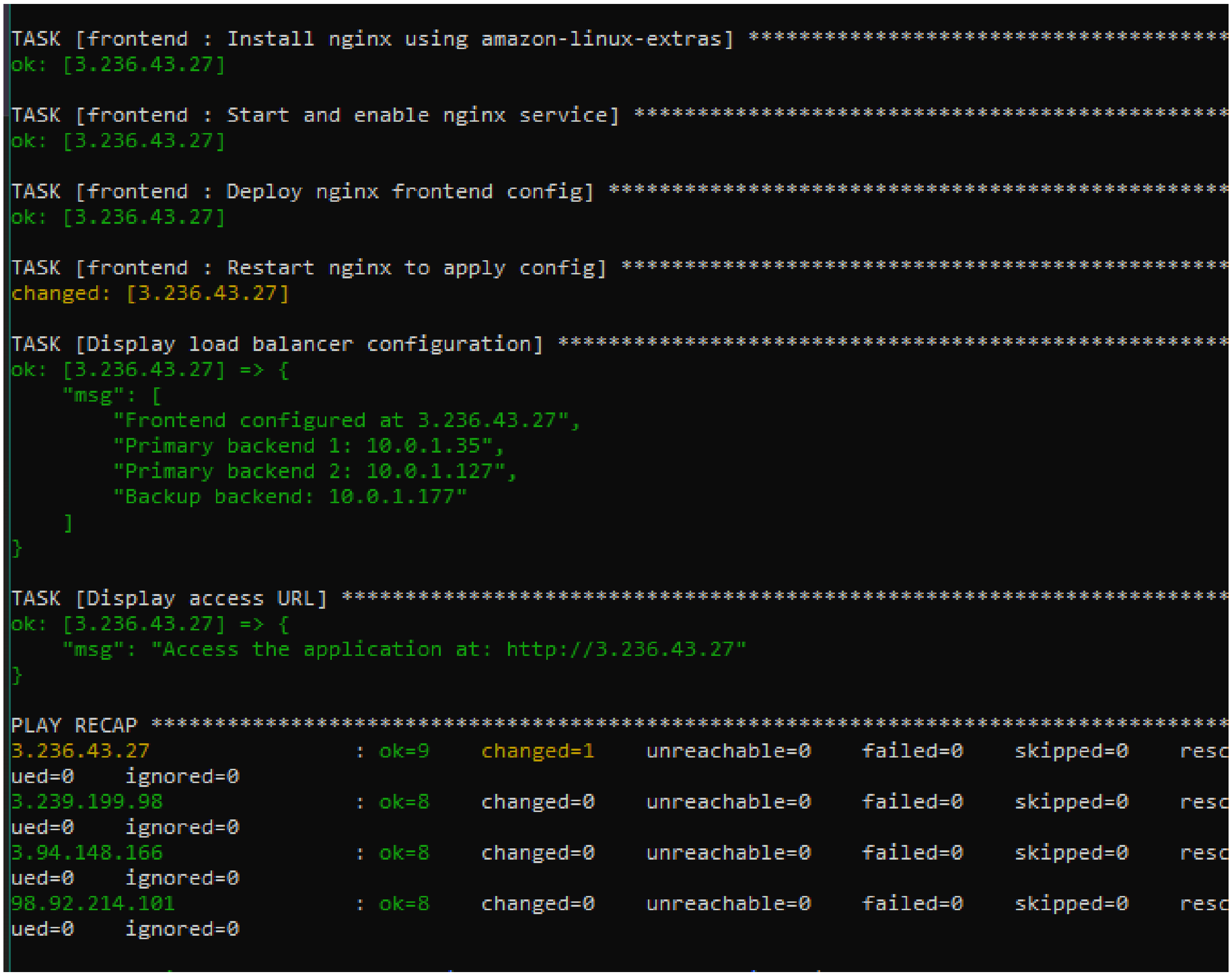


```
@reenaqureshi ▯ /workspaces/LabProject_FrontendBackend $ ssh -i ~/.ssh/id_ed25519 -o StrictHostK
eyChecking=no ec2-user@$FRONTEND_IP "sudo cat /etc/nginx/nginx.conf"
Warning: Permanently added '3.236.43.27' (ED25519) to the list of known hosts.
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log notice;
pid /run/nginx.pid;

# Load dynamic modules
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for" '
                      'upstream: $upstream_addr response_time: $upstream_response_time';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 4096;

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

    # Backend servers upstream configuration
```



```
        server 10.0.1.127:80;
        server 10.0.1.177:80 backup;
    }

    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;

        location / {
            proxy_pass http://backend_servers;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;

            # Timeouts
            proxy_connect_timeout 5s;
            proxy_send_timeout 10s;
            proxy_read_timeout 10s;

            # Retry configuration
            proxy_next_upstream error timeout http_502 http_503 http_504;
        }

        # Health check endpoint
        location /health {
            access_log off;
            return 200 "Nginx Load Balancer OK\n";
            add_header Content-Type text/plain;
        }
    }
}

@reenaqureshi ▯ /workspaces/LabProject_FrontendBackend $
```

Ansible Playbook Execution



```
TASK [frontend : Install nginx using amazon-linux-extras] ****************************************
ok: [3.236.43.27]

TASK [frontend : Start and enable nginx service] *************************************************
ok: [3.236.43.27]

TASK [frontend : Deploy nginx frontend config] ***************************************************
ok: [3.236.43.27]

TASK [frontend : Restart nginx to apply config] **************************************************
changed: [3.236.43.27]

TASK [Display load balancer configuration] *******************************************************
ok: [3.236.43.27] => {
    "msg": [
        "Frontend configured at 3.236.43.27",
        "Primary backend 1: 10.0.1.35",
        "Primary backend 2: 10.0.1.127",
        "Backup backend: 10.0.1.177"
    ]
}

TASK [Display access URL] ************************************************************************
ok: [3.236.43.27] => {
    "msg": "Access the application at: http://3.236.43.27"
}

PLAY RECAP ***************************************************************************************
3.236.43.27                : ok=9    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
3.239.199.98               : ok=8    changed=0    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
3.94.148.166               : ok=8    changed=0    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
98.92.214.101              : ok=8    changed=0    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
```

6. Challenges & Solutions

Challenge 1: Ansible Version Compatibility

Problem: Ansible 2.17+ uses Python 3.8+ syntax features, incompatible with Amazon Linux 2's Python 3.7

Solution:

- Downgraded Ansible to version 2.15.12
- Verified compatibility with Python 3.7
- Documented version requirements in README

Challenge 2: SSH Key Path Issues

Problem: Mismatch between SSH key locations in Terraform and actual key files

Solution:

- Created dedicated `ssh_keys/` directory in project root
- Updated `locals.tf` to use project-relative paths
- Added `ssh_keys/` to `.gitignore` to prevent committing private keys

Challenge 3: Nginx Package Installation

Problem: Nginx not available via standard `yum install nginx` on Amazon Linux 2

Solution:

- Used `amazon-linux-extras install nginx1 -y` command
- Modified Ansible frontend role to use shell module
- Added idempotency check with `creates` parameter

Challenge 4: Dynamic Backend IP Passing

Problem: Frontend Nginx needed backend private IPs that are only known after backend instances are created

Solution:

- Used Ansible's `hostvars` and `groups` to dynamically retrieve backend IPs
- Implemented fact gathering play before frontend configuration
- Passed IPs as variables to frontend role

---

7. Conclusion

7.1 Project Success

This lab project successfully demonstrates the implementation of a production-ready, highly available web infrastructure using modern Infrastructure as Code and Configuration Management tools. All objectives were met:

Infrastructure Design: VPC, subnets, security groups, and EC2 instances correctly provisioned **Ansible Roles:** Modular, reusable roles for frontend and backend configuration **High Availability:** Nginx load balancer with 2 active + 1 backup backend configuration **Automation:** Single-command deployment with Terraform-Ansible integration **Best Practices:** Clean code structure, proper documentation, secure credential handling

7.2 Key Learnings

1. **Infrastructure as Code Benefits:**
    - Repeatable, consistent deployments

   - Version-controlled infrastructure
   - Reduced manual configuration errors
2. **Separation of Concerns:**
   - Terraform handles infrastructure provisioning
   - Ansible manages configuration
   - Clear boundaries improve maintainability
3. **High Availability Patterns:**
   - Active-backup configuration provides resilience
   - Automatic failover ensures service continuity
   - Load balancing improves performance and reliability
4. **Automation Value:**
   - Single-command deployment saves time
   - Idempotent operations prevent configuration drift
   - Integrated workflows reduce human error

8. References

Documentation

1. Terraform AWS Provider Documentation
2. Ansible Documentation
3. Nginx Upstream Module
4. Amazon Linux 2 User Guide
5. Apache HTTP Server Documentation