Fatima Jinnah Women University

******PROJECT SUBMISSION

# Automated Server Provisioning and Configuration System

*Course Title : Cloud Computing*

# Submitted By :

Saira Ejaz ( 2023-BSE-057 )

Shumail Zahra

( 2023-BSE-061)

# Submitted To:

Prof. Waqas Saleem

# Dated:

January 23, 2026

# Department Of Software Engineering

Fatima Jinnah Women University

# **Project 2**

# *Automated Server Provisioning and Configuration System*

# **Repo Link:**

# **<https://github.com/2023-BSE-057-Saira/Project-2-Automated-Servers>**

# 1. Executive Summary

This project demonstrates the automation of server provisioning and configuration management using Terraform and Ansible. It provisions EC2 instances for web and application servers in AWS and applies configurations using Ansible roles. Configuration drift detection and remediation are implemented to maintain consistency and security. Terraform workspaces manage multiple environments such as dev, staging, and production. End-to-end validation ensures web and app services are running correctly and SSH security is enforced.

# 2. Architecture & Design

# **Terraform Workspaces & Infrastructure**

# Terraform is used to create AWS EC2 instances for web and application servers.Multiple workspaces (dev, staging, production) are used to separate environments, allowing independent deployments. Outputs from Terraform include public and private IP addresses for all servers.

# The main Terraform files used are main.tf, variables.tf, outputs.tf, and backend.tf.

# Terraform GitHub Pipeline-2026-01-24-132849

# Implementation Details

# Terraform initializes and applies infrastructure, generating outputs such as instance IDs and IP addresses. These outputs are used to configure Ansible inventory files. Ansible then connects to the servers, verifies connectivity, and configures web and application services. Both servers were successfully deployed and accessed.

# Part 1: Git Repository Setup (15 marks)

## Repository Structure (5 marks)

# **Tasks:**

# Create all directories and placeholder files.

# Implement .gitignore for Terraform/Ansible/credentials (see 1.3).

# Initialize Git repository and create an initial commit.

# **Deliverables:**

# Screenshot: project2\_part1\_repository\_structure.png

# 

# 

# Screenshot: project2\_part1\_gitignore.png

# 

# Screenshot: project2\_part1\_initial\_commit.png

# 

## Git Branching Strategy (5 marks)

# **Tasks:**

# Create dev and staging branches from main.

# Configure branch protection rules on main and staging.

# Document branching strategy in README.md.

# Create a sample feature branch (e.g., feature/add-ssh-hardening).

# **Deliverables:**

# Screenshot: project2\_part1\_git\_branches.png

# 

# 

# Screenshot: project2\_part1\_branch\_protection.png

# 

# Screenshot: project2\_part1\_branching\_diagram.png

# 

# 

## .gitignore Configuration (5 marks)

# **Tasks:**

# Add .gitignore with the rules above.

# Verify no sensitive files or state are committed.

# Document key ignore rules in README.md.

# **Deliverables:**

# Screenshot: project2\_part1\_gitignore\_content.png

# 

# 

# Screenshot: project2\_part1\_git\_status\_clean.png

# 

# ***Part 2: Terraform Infrastructure (30 marks)***

## 2.1 EC2 Servers Module (15 marks)

# **Tasks:**

# Implement web and app EC2 resources.

# Tag instances with Project, Environment, and Role.

# Output instance IDs and IPs.

# **Deliverables:**

# Screenshot: project2\_part2\_ec2\_module\_variables.png

# 

# 

# Screenshot: project2\_part2\_ec2\_module\_main.png

# 

# 

# Screenshot: project2\_part2\_ec2\_module\_outputs.png

# 

## 2.2 Root Terraform & Workspaces (15 marks)

# **Tasks:**

# Configure Terraform with AWS provider and default tags.

# Implement workspace-dependent configuration via local.env\_config.

# Output summary information (counts, IPs).

# **Deliverables:**

# Screenshot: project2\_part2\_main\_tf.png

# 

# 

# 

# Screenshot: project2\_part2\_variables\_tf.png

# 

# Screenshot: project2\_part2\_terraform\_output.png

# 

# ***Dev Workspace***

# **Heading: Dev Workspace – Terraform Output**

# Description: This workspace provisions a single web and a single application EC2 instance for development. The instances use t3.nano type to fit under our vCPU quota. Private IPs and instance IDs are shown below.

# 

# 

# ***Staging Workspace***

# **Heading: Staging Workspace – Terraform Output**

# Description:

# Staging workspace provisions 2 web and 2 app instance for pre-production validation. Used to test combined configuration and ensure automation works as expected. IPs and instance IDs are displayed below.

# 

# ***Production Workspace***

# **Heading: Production Workspace – Terraform Output**

# Description: Production workspace provisions 1 web and 1 app instance for live environment. Minimal instance counts are used due to vCPU quota limits. Private IPs and instance IDs are listed below.

# 

# Ansible Roles & Playbooks

# Ansible automates configuration management with the following roles:

# hardening: Applies SSH hardening, installs required packages.

# users: Creates the deployer user with SSH key access.

# web: Installs Nginx and deploys a test web page.

# app: Deploys application code and creates a systemd service.

# **Key playbooks:**

# configure-all.yml: Applies all roles.

# detect-drift.yml: Checks for configuration drift, including service status and SSH configuration.

# remediate-drift.yml: Restores the system to the desired configuration if drift is detected.

# Part 3: Ansible Configuration & Dynamic Inventory (25 marks)

## 3.1 Ansible Configuration and Dynamic Inventory (10 marks)

# **Tasks:**

# Create dynamic inventory files for dev, staging, production.

# Configure ansible.cfg.

# Create group\_vars/all.yml with reasonable defaults.

# **Deliverables:**

# Screenshot: project2\_part3\_ansible\_inventory\_dev.png

# 

# 

# 

# Screenshot: project2\_part3\_ansible\_cfg.png

# 

# Screenshot: project2\_part3\_group\_vars\_all.png

# 

# Ansible Dynamic Inventory for All Environments

# Description:

# This dynamic inventory uses the aws\_ec2 plugin to automatically discover EC2 instances for dev, staging, and production environments based on tags. Hosts are grouped by their Role tags (web or app), and connectivity is verified using the Ansible ping module. The inventory ensures that playbooks can run seamlessly across all environments without manually listing hosts.

# 

# 

# 

## 3.2 Ansible Roles for Hardening, Web, App, Users (15 marks)

# **Tasks:**

# Implement roles: hardening, users, web, app.

# Make roles idempotent (re-runs should not fail or re-change unnecessarily).

# Verify that web servers and app servers are correctly configured by role and hardened.

# **Deliverables:**

# Screenshot: project2\_part3\_roles\_structure.png

# 

# Screenshot: project2\_part3\_hardening\_role\_main.png

# 

# 

# 

# 

# 

# Screenshot: project2\_part3\_web\_role\_main.png

# 

# 

# 

# 

# Screenshot: project2\_part3\_app\_role\_main.png

# 

# 

# 

# 

# 

# Screenshot: project2\_part3\_configure\_all\_execution.png

# 

# 

# 

# ***Drift Detection & Remediation***

# Terraform plan is used to detect configuration drift by comparing the desired state with the actual infrastructure. If drift is detected, Terraform apply is executed to restore the correct configuration. This ensures infrastructure consistency and reliability.

# Part 4: Configuration Drift Detection & Remediation (20 marks)

## 4.1 Drift Detection (10 marks)

# **Tasks:**

# Implement detect-drift.yml that surfaces deviations clearly.

# Demonstrate a run where there is no drift and a run where you have manually changed something.

# **Deliverables:**

# Screenshot: project2\_part4\_detect\_drift\_playbook.png

# 

# Screenshot: project2\_part4\_detect\_drift\_execution.png

# 

## 4.2 Drift Remediation (10 marks)

# **Tasks:**

# Implement remediate-drift.yml.

# Show before/after evidence in screenshots/logs.

# **Deliverables:**

# Screenshot: project2\_part4\_remediate\_drift\_playbook.png

# 

# Screenshot: project2\_part4\_remediate\_drift\_execution.png

# 

# Part 5: CI/CD and End-to-End Provisioning (10 marks)

## 5.1 Optional GitHub Actions Workflows (5 marks)

# **Deliverables:**

# Screenshot: project2\_part5\_dev\_workflow\_run.png

# 

# 

# (If not using Actions, describe your manual pipeline in the report.)

# ***End-to-End Provisioning Test***

# An end-to-end test was performed in the development workspace. Terraform provisioned the servers, Ansible verified connectivity using ping, and services were configured successfully. The web service was accessed via browser, and the application service ran correctly.

## 5.2 End-to-End Provisioning Test (5 marks)

# **Deliverables:**

# Screenshot: project2\_part5\_end\_to\_end\_terraform\_apply.png

# 

# Screenshot: project2\_part5\_end\_to\_end\_ansible\_run.png

# 

# Screenshot: project2\_part5\_web\_and\_app\_validation.png

# SSH into the Web server

# 

# 

# Browser Testing

# 

# SSH into the App server

# 

# 

# 

# Browser Testing

# 

# Challenges & Solutions

# SSH access failed initially due to private IPs only; resolved by adding public IP outputs.

# Ansible tasks failed due to permission issues; fixed using become: yes.

# Missing or incorrect Ansible roles caused errors; roles were corrected or removed.

# Terraform module errors occurred due to mismatched variables and outputs; resolved by aligning module definitions.

# Conclusion

# The project successfully demonstrates automated infrastructure provisioning and configuration using Terraform and Ansible. The final system is modular, reliable, and scalable. Practical challenges were resolved through troubleshooting, making the solution suitable for real-world DevOps workflows.

# Appendices

# This section contains Terraform configurations, Ansible playbooks, inventory files, logs, and screenshots demonstrating successful execution.

# **Installations:**

# 

# Files Creation:

# 

# Ansible testing:

# 

# 

# Staging Workflow – .github/workflows/provision-staging.yml

# 

# Production Workflow – .github/workflows/provision-production.yml

# 

# 