# Project Title: Automating configuration management across environments with ansible

**Overview**

This project automates configuration management across multiple environments using Ansible, ensuring consistency, scalability, and security. By defining infrastructure as code, it simplifies server provisioning, deployment, and maintenance, reducing manual effort and misconfigurations.

Ansible’s agentless architecture makes it lightweight, while YAML-based playbooks ensure repeatable, idempotent configurations. Using modular roles, the setup enhances reusability and maintainability. Ansible Vault secures sensitive data like credentials.

This framework helps organizations streamline IT operations, enforce compliance, and maintain high availability with minimal manual intervention.

# Key Components

## Playbooks

* + Define automation tasks for configuring servers.
  + The main playbook orchestrates all configurations, while separate playbooks handle web and database server setup.
  + Roles are used for modular automation.

## Inventory

* + Defines the target hosts for different environments like production and staging.
  + Helps in managing multiple servers efficiently.

## Variables

* + Stores customizable values for configurations such as server settings, ports, and credentials.
  + Enables easy modifications without changing playbooks.

## Roles

* + Provides a structured way to organize tasks, handlers, templates, and variables.
  + Ensures modularity and reusability of automation scripts.

## Ansible Vault

* + Secures sensitive data such as passwords and API keys.
  + Ensures safe configuration management across environments.

# Deploying Automated Configuration Management Across Environments with Ansible

This project automates configuration management using **Ansible**, ensuring **consistent and scalable deployments** across different environments like **staging and production**.

# Deployment Workflow

## Define Infrastructure & Inventory

* + The inventory/ directory contains files specifying **staging** and **production** environments.
  + Hosts (servers) are grouped to target specific configurations for each environment.

## Automate Configuration with Playbooks

* + The playbooks/ directory includes:
    - **site.yml** – The main playbook orchestrating all tasks.
    - **webserver.yml** – Configures web servers with required dependencies.
    - **dbserver.yml** – Sets up and secures database servers.
    - **roles/** – Organizes reusable configurations for modular deployment.

## Customize Variables for Flexibility

* + The vars/ directory stores environment-specific configurations, enabling customization without modifying playbooks.

## Deploy Using Ansible

Run Ansible commands to execute playbooks and automate configuration: ansible-playbook -i inventory/production playbooks/site.yml

or for staging:

ansible-playbook -i inventory/staging playbooks/site.yml

This ensures **automated and consistent configuration** across environments.

## Security & Compliance

* + Use **Ansible Vault** to encrypt sensitive variables (passwords, API keys).
  + Implement **firewall rules and security policies** using Ansible tasks.

# Features and Benefits of Automating Configuration Management with Ansible

**Features :**

* + Agentless architecture eliminates the need for additional software on managed nodes.
  + Inventory management organizes servers into different environments like staging and production.
  + Declarative playbooks use YAML to define configurations, ensuring repeatability.
  + Role-based automation allows modular and reusable configurations.
  + Variable management enables customization without modifying playbooks.
  + Ansible Vault secures sensitive data such as passwords and API keys.
  + Scalable and flexible to manage large infrastructures efficiently.
  + Seamless integration with CI/CD tools like Jenkins, GitHub Actions, and GitLab CI/CD.
  + Idempotency ensures consistent and stable configurations without unintended changes.

# Benefits :

* + Ensures consistency across all environments, reducing configuration drift.
  + Automates deployments, minimizing manual intervention and speeding up processes.
  + Enhances security by enforcing best practices and maintaining compliance.
  + Reduces operational costs by eliminating manual setup and maintenance efforts.
  + Simplifies infrastructure management across cloud, on-premises, and hybrid environments.
  + Scales efficiently to handle thousands of servers with minimal effort.

# Github link:

ttps://github.com/Zakiya-Tasneem/Automating-configuration-management-across-environments-with-ansible