Objective:

This assignment focuses on understanding core DevOps principles. You will containerize a web application, provision cloud infrastructure using Terraform, automate the deployment using **Jenkins**, and leverage **AI tools** for security scanning and code remediation.

Scenario:

You are a Devops Engineer. Your manager wants to ensure that the infrastructure we deploy to the cloud is secure by default.

Your task is to:

1] Provision Cloud Infrastructure for a web app.

2] Build a **Jenkins Pipeline** that automatically scans your infrastructure code (Terraform) for security vulnerabilities before deployment.

3] Use **AI** to analyze the security report, explain the risks, and rewrite the code to fix the vulnerabilities.

Requirements:

1. Web Application & Docker

- Choose a simple web application (**Node.js or Python**). Sem End Project
- Create:
    - Dockerfile
    - docker-compose.yml
- Ensure the application runs locally using Docker.

2. Infrastructure as Code (Terraform)

- Write **Terraform code** to provision infrastructure on **any cloud provider** of your choice (for example: AWS, Azure, GCP, etc.).
  Provision at least:
    - A virtual machine / compute instance
    - Networking and security configuration

Intentional Vulnerability (Required)

Initially, your Terraform code **must include a known security flaw**, such as:

SSH (port 22) open to 0.0.0.0/0
Publicly exposed management port
Unencrypted disk volumes

Overly permissive firewall/security rules

- This vulnerability will be fixed later using AI-based remediation.

3. CI/CD Pipeline (Jenkins)

- Run **Jenkins locally using Docker**.
- Create a Jenkins Pipeline with the following stages:

Stage 1: Checkout

- Pull source code from a Git repository.

Stage 2: Infrastructure Security Scan

- Integrate a security scanning tool such as **Trivy**.
- Scan **Terraform files** for misconfigurations and vulnerabilities.
  This stage should:
  Run only if the scan passes **OR**
  Fail on failure but clearly show security warnings
  Update your Terraform code using the AI's recommendations and generate report in jenkins console for fixing the issue
    - Re-run the Jenkins pipeline.
    - Confirm that: The scan passes

- There are **zero critical security issues**

Stage 3: Terraform Plan

- Run terraform plan

4. AI-Driven Security Remediation (Core Task)

1] Run the Jenkins pipeline.
2] The pipeline should **fail or produce warnings** due to the intentional vulnerability.
3] Copy the **Trivy vulnerability report** from the Jenkins console.

5. Documentation

Create a README.md that includes:

Project Overview

- Architecture explanation
- Cloud provider used
- Tools and technologies

Before & After Security Report

Screenshots of:

Initial failing Jenkins scan

Final passing Jenkins scan

AI Usage Log (Mandatory)

The **exact AI prompt** used

-Summary of Identified risks

-How the AI-recommended changes improved security

Submission Guidelines

Submit a **GitHub repository link** containing:

1] Source code & Docker files

2] Jenkins Pipeline

3] terraform/ directory (final secured version)

4] README.md with **GenAI Usage Report**

5] **Video Recording:** Submit a 5–10 minute screen recording demonstrating the Jenkins pipeline execution, security scans, Terraform deployment, and the application running on the cloud public IP (link added in README.md)**.**

Required Screenshots

-Jenkins pipeline success

-Security vulnerability report

-Application running on the **cloud public IP or Domain**

Timeline- 5 days

<u>Evaluation Criteria</u>

Pipeline Automation- Jenkins pipeline successfully pulls code and runs security scans

Security Awareness- Clear identification and remediation of infrastructure vulnerabilities

AI Utilization- Evidence that AI was used to **understand and fix** security flaws
(verified through the AI Usage Log)

Cloud Deployment- Application is accessible via the cloud provider's public IP

Good luck on your DevSecOps journey! 🚀