“Kubernetes Cluster + Ansible Automation + Jenkins CI/CD + Secure Microservices Deployment”

1) Project Overview

Team 3 will build a complete, secure DevOps platform using:

- Kubernetes cluster (kubeadm) on **3 EC2 instances**
- Ansible for:
    - Node provisioning
    - Cluster initialization
    - Application deployment
    - Monitoring setup
- Jenkins for:
    - CI pipelines
    - CD pipelines (via Ansible)
- Multi-stage Docker builds
- Secure deployments with NetworkPolicies
- Prometheus + Grafana for monitoring
- Full production-grade documentation

2) AWS Infrastructure (Only 3 Instances)

Team 3 must provision:

EC2 Instances (Ubuntu 22.04)

- **k8s-master**
- **k8s-worker1**

- **k8s-worker2**

Security Groups:

- Allow Kubernetes required ports
- Allow internal node-to-node communication
- SSH restricted to your IP only

No other machines.
No Terraform.
No EKS.
No extra services.

3) Ansible Automation (Core of the Project)

Team 3 must write **Ansible playbooks** to automate the entire platform.

1. Node Provisioning

Playbook: `prepare-nodes.yml`

- Disable swap
- Install containerd
- Configure containerd
- Install kubeadm, kubelet, kubectl
- Load required modules
- Apply OS-level security hardening

2. Kubernetes Cluster Bootstrap

Playbook: `kubeadm-init.yml`

- Initialize master node
- Save kubeconfig
- Install CNI (Calico with NetworkPolicy support)
- Join worker nodes
- Validate node status

3. Application Deployment (K8s)

Playbook: `deploy-app.yml`

- Apply Deployments
- Apply Services
- Apply Ingress
- Apply ConfigMaps
- Apply Secrets
- Apply NetworkPolicies
- Rolling deploy logic

4. Monitoring Setup

Playbook: `monitoring.yml`

- Install Helm
- Deploy Prometheus stack
- Deploy Grafana
- Configure dashboards

5) Secure Deployment Requirements

Team 3 must implement:

Containers:

- Multi-stage builds
- Minimal base images
- Non-root users
- Read-only root filesystem
- Drop unnecessary capabilities

Kubernetes:

- Liveness/Readiness probes
- Resource limits
- Secrets in Kubernetes Secrets
- **NetworkPolicies:**

- default-deny baseline
- allow frontend → backend
- allow backend → postgres
- allow backend → redis

CI/CD:

- No plaintext credentials
- Jenkins credential store only
- Secure SSH to master node

6) Jenkins CI Pipeline

Stage 1 — Code Quality

- Code linting (backend + frontend)
- YAML linting
- Dockerfile linting
- Ansible playbook linting (`ansible-lint`)

Stage 2 — Smoke Tests

- Run unit tests
- Small integration test for backend API

Stage 3 — Build Multi-Stage Docker Images

Stage 4 — Push to AWS ECR

7) Jenkins CD Pipeline (Ansible-Based Deployment

Deployment is executed using Ansible, not direct kubectl.

Pipeline stages:

Stage 1 – Get latest artifacts

- Pull ECR image tag
- Update Ansible variables in repo

Stage 2 – Apply Deployment Playbook

ansible-playbook -i hosts deploy-app.yml

10) Documentation Requirements

must deliver a full professional documentation package.