Kubeadm Cluster • Secure Deployment • Monitoring • Jenkins CI/CD

1. Project Overview

The team must build a complete DevOps project on AWS cloud following production-grade practices:

- Provisioning a **kubeadm Kubernetes cluster** on 3 EC2 instances.
- Deploying microservices securely using Kubernetes best practices.
- Enforcing **Network Policies** for zero-trust traffic control.
- Setting up **Prometheus + Grafana** monitoring via Helm.
- Implementing a full **CI/CD pipeline with Jenkins**:
    - Smoke tests
    - Multi-stage secure Docker builds
    - Push images to AWS ECR
    - Deploy to the kubeadm cluster
- Delivering strong, professional documentation of the entire system.

2. Cloud Infrastructure Requirements

The team must provision the following resources on **AWS**:

Compute

- EC2 Instance 1 → Kubernetes Master Node
- EC2 Instance 2 → Worker Node
- EC2 Instance 3 → Worker Node
  OS: **Ubuntu 22.04**

Networking

- VPC with public subnets
- Internet Gateway + Route tables
- Security Groups allowing:
    - Node-to-node communication

    - Kubernetes API Server access
    - SSH from trusted IPs
- Proper inbound/outbound rules for master and worker nodes

Node System Requirements

Each node must be prepared by the team:

- Disable swap
- Install a container runtime
- Install kubeadm, kubelet, kubectl
- Configure CRI
- Prepare the node for cluster initialization or joining

Hint: Use the official Kubernetes documentation for kubeadm installation.

3. Kubernetes Cluster Setup

After all machines are prepared:

Cluster Bootstrapping

- Initialize Kubernetes master with kubeadm
- Join worker nodes to the cluster
- Verify full node registration

CNI Plugin

The team must install a CNI that **supports NetworkPolicy** enforcement.

Hint: Research “Kubernetes CNI with NetworkPolicy support”.

Namespaces

All microservices must be deployed under a dedicated namespace:

vprofile

RBAC

Define:

- ServiceAccounts
- Roles
- RoleBindings

for secure access to cluster resources.

4. Secure Application Deployment

Each microservice must meet the following requirements:

Multi-Stage Docker Images

Each service must:

- Use a multi-stage build
- Run as a non-root user
- Use minimal base images
- Avoid embedding secrets
- Pass a container vulnerability scan (report required)

Kubernetes Manifests

For each microservice, the team must create:

- Deployment
- Service
- ConfigMap
- Secret
- Liveness & Readiness Probes
- Resource Requests & Limits
- PodSecurityContext
- Labels & annotations for monitoring

Network Policies

The deployment must implement a **zero-trust model**:

- Allow only required traffic (app → DB, app → cache, app → broker)
- Deny all other cross-service communication

Hint: Use a “default deny” policy as a baseline.

5. Monitoring Stack (Helm)

Monitoring components must be installed inside a dedicated namespace:

monitoring

Required Components

- Prometheus
- Grafana
- Node Exporters
- Cluster metrics collectors

Hint: Use the official Helm chart for kube-prometheus-stack.

Monitoring Deliverables

The team must create dashboards for:

- Node health
- Pod health
- Application performance
- Cluster CPU/memory usage
- Basic alert rules

Screenshots must be included in the documentation.

6. Jenkins CI/CD Pipeline

The CI/CD pipeline must run entirely on Jenkins.

CI Pipeline (3 Stages)

Stage 1 – Smoke Test

- Validate code structure
- Validate Dockerfiles
- Validate Kubernetes YAMLs (linting)

Stage 2 – Build & Scan

- Build multi-stage Docker images
-

Stage 3 – Push to AWS ECR

- Authenticate to ECR
- Push the built image

CD Pipeline (Automated Deployment)

The deployment is done **from Jenkins to the kubeadm master node** using SSH.

The pipeline must:

- Connect to the master node using a secure SSH key
- Apply updated Kubernetes manifests
- Trigger a rolling update automatically

Hint: Use a Jenkins “Execute shell” step over SSH.

Required Jenkins Secrets

- ECR Access Key
- ECR Secret Key
- Master node SSH private key
- Master node public IP

7. Security Requirements

Cluster Security

- No containers running as root
- Enforce PodSecurity standards
- Read-only root filesystem
- Limit container capabilities
- Apply NetworkPolicies
- Use namespace isolation
- Enable RBAC for all services

Image Security

- Multi-stage images
- Minimal base images
- Remove all unnecessary packages

Deployment Security

- Secrets stored in Kubernetes Secrets
- No plaintext credentials in YAML
- Use dedicated service accounts
- Apply least privilege RBAC rules

8. Documentation Requirements (Mandatory)

The documentation must be **professional**, structured, and detailed.
The team must deliver **one complete documentation package** containing the following sections: