

# PlatformOps – End-to-End DevOps Project Planner

This document provides a realistic, phase-by-phase execution plan for building a production-style DevOps platform named **PlatformOps**. The goal is to integrate Infrastructure as Code, CI/CD, Kubernetes, Monitoring, and Zero-Downtime Deployments in a way that is interview-ready and industry-aligned.

## Phase 0 – Project Foundation & Planning

- Finalize project name: PlatformOps.
- Create a Git repository with a clean folder structure (terraform/, ci-cd/, k8s/, monitoring/, docs/).
- Write a short project vision explaining the problem PlatformOps solves.
- Decide target cloud (AWS or Azure) and CI/CD tool (Jenkins recommended).
- Prepare architecture and flow diagrams (hand-drawn or digital).

## Phase 1 – Infrastructure Provisioning with Terraform

- Install Terraform and configure cloud provider credentials.
- Provision base infrastructure (VPC/VNet, subnets, security groups).
- Create compute resources (EC2/VM) to host the Kubernetes cluster.
- Install Docker and Kind on provisioned machines.
- Use Terraform provisioners or scripts to bootstrap the cluster.
- Validate cluster access using kubectl.

## Phase 2 – Kubernetes Cluster Setup & Baseline

- Verify nodes, namespaces, and core Kubernetes components.
- Install metrics-server and validate resource metrics.
- Create namespaces for frontend, backend, monitoring, and ingress.
- Set up RBAC for namespace-level access.
- Define baseline resource quotas and limits.

## Phase 3 – Application & Microservices

- Choose a simple microservice application (frontend, backend, database).
- Dockerize each service with proper tagging and versioning.
- Push images to Docker Hub or another container registry.
- Create Kubernetes Deployments, Services, ConfigMaps, and Secrets.
- Validate internal service-to-service communication.

## Phase 4 – CI/CD Pipeline with Jenkins

- Install Jenkins (VM or container-based).
- Configure Jenkins credentials for Git and Docker registry.
- Create CI pipeline: build, test, dockerize, push image.
- Create CD pipeline: deploy updated images to Kubernetes.
- Implement parameterized builds for different environments.

## Phase 5 – Advanced Kubernetes Deployments

- Implement readiness and liveness probes.
- Configure Horizontal Pod Autoscalers (HPA) for frontend and backend.
- Apply rolling update strategies.
- Implement Blue-Green deployment using separate deployments and service switching.
- Validate zero-downtime deployments.

## Phase 6 – Monitoring & Observability

- Install Prometheus and Grafana using Helm.
- Configure dashboards for nodes, pods, and services.
- Monitor HPA behavior under load.
- Set up basic alerting rules.
- Document key metrics and dashboards.

## Phase 7 – Security & Governance

- Implement namespace-level RBAC.
- Secure Secrets using Kubernetes Secrets and best practices.
- Apply Network Policies for pod-to-pod communication.
- Harden container images (non-root, minimal base images).

## Phase 8 – Documentation & Interview Readiness

- Write a detailed README explaining architecture and workflows.
- Add diagrams for CI/CD, Kubernetes flow, and Blue-Green deployment.
- Prepare interview explanations for each design decision.
- Add future roadmap (VPA, GitOps, ArgoCD, Service Mesh).
- Finalize the project as a portfolio-ready showcase.