

# Generic CI/CD Pipeline Reference Guide

This document serves as a **standard reference** for creating a **CI/CD pipeline** for any type of project (MERN, Django, Python, Java, etc.) and deploying it to a **Kubernetes cluster** using **Jenkins, Docker, SonarQube, Nexus, and Kubernetes**.

It is designed so that **any team member** can follow the steps and adapt the files for their own project.

---

## 1. Overall CI/CD Flow

**Code → Build → Analyze → Package → Deploy (Automated via Jenkins)**

### Unified Access Model

- All deployments and cluster interactions happen **only through Jenkins**
- Jenkins has the required Kubernetes permissions to deploy any project
- No user requires direct Kubernetes access for normal operations

### Student Responsibility

1. Developer pushes code to GitHub
2. Jenkins pipeline is triggered
3. Application is built and tested
4. Code quality is checked
5. Docker image is created
6. Image is pushed to registry
7. Jenkins applies Kubernetes manifests and deploys the project

### Special Support Role (Exception Only)

- Lens is used **only when required** for:
  - Viewing logs
  - Inspecting pod/events
  - Debugging cluster-level issues

 **Important:** Kubernetes access is abstracted via Jenkins. Direct cluster access is not part of the normal workflow.

---

## 2. Prerequisites

### For Students

- GitHub account
- Basic Docker knowledge
- Jenkins pipeline understanding
- Access to shared Jenkins (provided by college)

### Common Infrastructure (Provided)

- Jenkins server (pre-configured)
  - Nexus / private registry
  - SonarQube
  - Kubernetes cluster (through jenkins only)
- 

## 3. Standard Project Structure (Mandatory)

Every student **must follow this structure** so that Project can be deployed easily:

- project-root/
    - app/ # Application source code
    - Dockerfile # Mandatory
    - Jenkinsfile # Mandatory
    - k8s/
      - deployment.yaml
      - service.yaml
      - pvc.yaml (only if required)
      - ingress.yaml
    - sonar-project.properties (optional)
    - requirements.txt / package.json
    - README.md
- 

## 4. Dockerfile

[Refrence Docker files](#)

---

## **5. Jenkinsfile (Generic Pipeline)**

[Refrence Jenkins Pipeline](#)

---

## **6. Kubernetes Deployment Template**

[Refrence deployment.yaml](#)

---

## **7. Kubernetes Service Template**

[Refrence service.yaml](#)

---

## **8. PersistentVolumeClaim (Optional – if app needs storage)**

[Refrence PVC.yaml](#)

---

## **9. Ingress Template**

[Refrence ingress.yaml](#)

---

## 10. Common Errors & Fixes

### **ImagePullBackOff**

- Image tag mismatch
- Wrong registry URL
- Missing imagePullSecrets

### **CrashLoopBackOff**

- Application not starting
- Wrong CMD or PORT
- Missing environment variables

### **Jenkins Pod Not Starting**

- YAML indentation issues
  - Privileged mode missing
- 

## 11. Access & Responsibility Model

### **What Students MUST Do**

- Create GitHub repository
- Add Dockerfile, Jenkinsfile, and k8s folder
- Ensure Jenkins pipeline:
  - Builds successfully
  - Pushes Docker image to registry
  - Deploys automatically via Jenkins

### **What Students MUST NOT Do**

- Do not manually deploy to Kubernetes
- Do not rely on Lens for deployment

### **Special Case: Operational Debugging**

- Lens access is used **only if required**
- Used strictly for investigation and logging
- No configuration changes are performed outside Jenkins

---

## 12. Best Practices

- Always use **versioned image tags**
  - Keep manifests in `k8s/` folder
  - Test Docker image locally before pipeline
  - Use `roll_nos` for namespaces for each project
- 

## 13. Summary

This document defines a **clear separation of responsibilities**:

By enforcing a **standard structure and pipeline**, projects from any stack (MERN, Django, Python, Java) can be deployed consistently on the college server.

This ensures:

- Security
  - Consistency
  - Faster deployments
  - Easier troubleshooting
- 

## 14. Submission Checklist (For Students)

Before submission, ensure:

- Jenkins pipeline is GREEN
  - Image pushed to registry
  - Deployment stage executed successfully
  - Application pod is running
  - Dockerfile works locally
  - Jenkinsfile uses correct registry & credentials
  - k8s YAMLs are present in repo
  - README explains app, port, and health check
- 

## 15. Operational Rule

- Jenkins is the **single source of truth** for Kubernetes access
- All users interact with Kubernetes **indirectly via Jenkins**
- Lens is a **read/debug-only tool** for exceptional situations
- Any permanent fix must be applied through GitHub + Jenkins