

# DEVOPS LAB MANUAL

## CI/CD Pipelines using Maven, React, Flask, Docker Swarm & Kubernetes

### PROGRAM 1: Maven + Java + Docker Swarm (Manual)

Aim:

Implement manual CI/CD pipeline for Java app.

Tools Required:

Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application

```
mvn archetype:generate -DgroupId=com.devops -DartifactId=javaapp -DarchetypeArtifactId=maven-archetype
```

2. Dockerfile

```
FROM openjdk:17
```

```
WORKDIR /app
```

```
COPY target/javaapp-1.0-SNAPSHOT.jar app.jar
```

```
CMD ["java", "-jar", "app.jar"]
```

3. Push to GitHub

```
git init
```

```
git add .
```

```
git commit -m "Initial Commit"
```

```
git remote add origin <repo-url>
```

```
git push -u origin master
```

4. Jenkinsfile

```
pipeline {
```

```
  agent any
```

```
  stages{
```

```
    stage('Clone'){steps{git '<repo-url>'}}
```

```
    stage('Build'){steps{sh 'mvn clean package'}}
```

```
    stage('Docker Build'){steps{sh 'docker build -t javaapp:1.0 .'}}
```

```
    stage('Deploy'){steps{sh 'docker service create --name javaapp -p 8081:8080 javaapp:1.0'}}
```

```
  }
```

```
}
```

5. Deployment

```
docker swarm init
```

Conclusion:

Manual pipeline successfully implemented.

### PROGRAM 2: Maven + Java + Docker Swarm (Cron)

Aim:

Automated Java CI/CD pipeline.

Tools Required:

Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application

Same as Program 1

2. Dockerfile

Same as Program 1

3. Push to GitHub

```
git init
```

```
git add .
```

```
git commit -m "Initial Commit"
```

```
git remote add origin <repo-url>
git push -u origin master
```

4. Jenkinsfile  
Same Jenkinsfile + Jenkins Trigger: H/5 \* \* \* \*

5. Deployment  
docker swarm init

Conclusion:  
Automated pipeline executed.

## PROGRAM 3: Maven + Java + Kubernetes (Manual)

Aim:  
Deploy Java app on Kubernetes.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:  
1. Create Application  
Same as Program 1

2. Dockerfile  
Same as Program 1

3. Push to GitHub  
git init  
git add .  
git commit -m "Initial Commit"  
git remote add origin <repo-url>  
git push -u origin master

4. Jenkinsfile  
pipeline {... Deploy using kubectl apply -f deployment.yaml }

5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
Java app deployed on Kubernetes.

## PROGRAM 4: Maven + Java + Kubernetes (Cron)

Aim:  
Automated Kubernetes pipeline.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:  
1. Create Application  
Same as Program 3

2. Dockerfile  
Same as Program 3

3. Push to GitHub  
git init  
git add .  
git commit -m "Initial Commit"  
git remote add origin <repo-url>  
git push -u origin master

4. Jenkinsfile  
Same Jenkinsfile + Cron Trigger

5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
Cron based pipeline executed.

## PROGRAM 5: React + Docker Swarm (Manual)

Aim:  
Manual CI/CD for React.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
`npm create-react-app reactapp`
2. Dockerfile  
`FROM node:18`  
`WORKDIR /app`  
`COPY . .`  
`RUN npm install`  
`RUN npm run build`  
`CMD ["npm", "serve", "-s", "build"]`
3. Push to GitHub  
`git init`  
`git add .`  
`git commit -m "Initial Commit"`  
`git remote add origin <repo-url>`  
`git push -u origin master`
4. Jenkinsfile  
`pipeline { stage('Clone')... }`
5. Deployment  
`docker service create --name reactapp -p 3000:3000 reactapp:1.0`

Conclusion:  
React deployed on Swarm.

## PROGRAM 6: React + Docker Swarm (Cron)

Aim:  
Automated React pipeline.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
Same as Program 5
2. Dockerfile  
Same as Program 5
3. Push to GitHub  
`git init`  
`git add .`  
`git commit -m "Initial Commit"`  
`git remote add origin <repo-url>`  
`git push -u origin master`
4. Jenkinsfile  
Same Jenkinsfile + Cron Trigger
5. Deployment  
`docker service create ...`

Conclusion:  
Cron pipeline executed.

## PROGRAM 7: React + Kubernetes (Manual)

Aim:  
Deploy React on Kubernetes.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
Same as Program 5
2. Dockerfile  
Same as Program 5
3. Push to GitHub  
git init  
git add .  
git commit -m "Initial Commit"  
git remote add origin <repo-url>  
git push -u origin master
4. Jenkinsfile  
Deploy using kubectl
5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
React deployed on Kubernetes.

## PROGRAM 8: React + Kubernetes (Cron)

Aim:  
Automated React Kubernetes pipeline.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
Same as Program 7
2. Dockerfile  
Same as Program 7
3. Push to GitHub  
git init  
git add .  
git commit -m "Initial Commit"  
git remote add origin <repo-url>  
git push -u origin master
4. Jenkinsfile  
Same Jenkinsfile + Cron
5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
Cron pipeline executed.

## PROGRAM 9: Flask + Docker Swarm (Manual)

Aim:  
CI/CD for Flask app.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
Create app.py
2. Dockerfile

```

FROM python:3.10
WORKDIR /app
COPY . .
RUN pip install flask
CMD ["python","app.py"]

3. Push to GitHub
git init
git add .
git commit -m "Initial Commit"
git remote add origin <repo-url>
git push -u origin master

4. Jenkinsfile
pipeline { stage('Clone')... }

5. Deployment
docker service create --name flaskapp -p 5000:5000 flaskapp:1.0

Conclusion:
Flask deployed on Swarm.

```

## PROGRAM 10: Flask + Docker Swarm (Cron)

```

Aim:
Automated Flask pipeline.

Tools Required:
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:
1. Create Application
Same as Program 9

2. Dockerfile
Same as Program 9

3. Push to GitHub
git init
git add .
git commit -m "Initial Commit"
git remote add origin <repo-url>
git push -u origin master

4. Jenkinsfile
Same Jenkinsfile + Cron

5. Deployment
docker service create ...

Conclusion:
Cron pipeline executed.

```

## PROGRAM 11: Flask + Kubernetes (Manual)

```

Aim:
Deploy Flask on Kubernetes.

Tools Required:
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:
1. Create Application
Same as Program 9

2. Dockerfile
Same as Program 9

3. Push to GitHub
git init
git add .
git commit -m "Initial Commit"
git remote add origin <repo-url>

```

```
git push -u origin master
```

4. Jenkinsfile  
Deploy using kubectl

5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
Flask deployed on Kubernetes.

## PROGRAM 12: Flask + Kubernetes (Cron)

Aim:  
Automated Flask Kubernetes pipeline.

Tools Required:  
Git, GitHub, Jenkins, Docker, Java, Maven/Node/Python, Kubernetes/Docker Swarm

Procedure:

1. Create Application  
Same as Program 11
2. Dockerfile  
Same as Program 11
3. Push to GitHub  
git init  
git add .  
git commit -m "Initial Commit"  
git remote add origin <repo-url>  
git push -u origin master
4. Jenkinsfile  
Same Jenkinsfile + Cron
5. Deployment  
kubectl apply -f deployment.yaml

Conclusion:  
Cron pipeline executed.