DevOps Project Documentation

# 1. Application

Clone the repository and run the application on port 80 (HTTP):

git clone https://github.com/sriram-R-krishnan/devops-build.git  
cd devops-build

# 2. Docker

Dockerfile:

FROM node:14  
WORKDIR /app  
COPY package\*.json ./  
RUN npm install  
COPY . .  
EXPOSE 80  
CMD ["npm", "start"]

docker-compose.yml:

version: "3"  
services:  
 web:  
 build: .  
 ports:  
 - "80:80"

# 3. Bash Scripting

build.sh:

#!/bin/bash  
IMAGE\_NAME="devops-app"  
IMAGE\_TAG="latest"  
  
echo "Building Docker image..."  
docker build -t $IMAGE\_NAME:$IMAGE\_TAG .

deploy.sh:

#!/bin/bash  
IMAGE\_NAME="devops-app"  
IMAGE\_TAG="latest"  
  
echo "Deploying Docker container..."  
docker run -d -p 80:80 $IMAGE\_NAME:$IMAGE\_TAG

# 4. Version Control (GitHub)

Add .gitignore and .dockerignore, then push code to dev branch using CLI:

git init  
git checkout -b dev  
git add .  
git commit -m "Initial commit"  
git remote add origin https://github.com/<your-username>/<repo>.git  
git push origin dev

# 5. Docker Hub

- Create 2 repos: dev (public) and prod (private).  
- Push images using:  
docker login  
docker tag devops-app:latest <your-dockerhub-username>/devops-app-dev:latest  
docker push <your-dockerhub-username>/devops-app-dev:latest

# 6. Jenkins

Install plugins: Docker, GitHub, Pipeline. Configure webhook for auto-build.

Jenkinsfile:

pipeline {  
 agent any  
 environment {  
 DEV\_REPO = "<your-dockerhub-username>/devops-app-dev"  
 PROD\_REPO = "<your-dockerhub-username>/devops-app-prod"  
 }  
 stages {  
 stage('Clone') {  
 steps {  
 git branch: 'dev', url: 'https://github.com/<your-username>/<repo>.git'  
 }  
 }  
 stage('Build Docker Image') {  
 steps {  
 sh 'docker build -t devops-app:latest .'  
 }  
 }  
 stage('Push to DockerHub') {  
 steps {  
 script {  
 if (env.BRANCH\_NAME == "dev") {  
 sh "docker tag devops-app:latest $DEV\_REPO:latest"  
 sh "docker push $DEV\_REPO:latest"  
 } else if (env.BRANCH\_NAME == "master") {  
 sh "docker tag devops-app:latest $PROD\_REPO:latest"  
 sh "docker push $PROD\_REPO:latest"  
 }  
 }  
 }  
 }  
 stage('Deploy to AWS') {  
 steps {  
 sh "ssh -o StrictHostKeyChecking=no ec2-user@<EC2\_PUBLIC\_IP> 'docker run -d -p 80:80 $DEV\_REPO:latest'"  
 }  
 }  
 }  
}

# 7. AWS

- Launch t2.micro EC2 instance  
- Install Docker:  
sudo yum update -y  
sudo amazon-linux-extras install docker -y  
sudo service docker start  
sudo usermod -aG docker ec2-user  
- Security Group rules:  
 \* Port 80 (HTTP) - open to all  
 \* Port 22 (SSH) - only from your IP

# 8. Monitoring

- Use Prometheus + Grafana or AWS CloudWatch for monitoring  
- Setup alerts for high CPU/memory usage or app downtime  
- Notifications via email/SNS/Slack when app goes down

# 9. Workflow Summary

1. Developer pushes code → GitHub (dev branch).  
2. Jenkins auto-build triggers → builds Docker image → pushes to DockerHub dev repo → deploys to EC2.  
3. On master branch → image goes to prod repo (private).  
4. Monitoring system alerts if app goes down.