pipeline {

    agent any

    parameters {

        string(name: 'CONTAINER\_NAME', defaultValue: 'saikart-app', description: 'Name of the Docker container')

        string(name: 'HOST\_PORT', defaultValue: '9090', description: 'Port to expose the application')

    }

    environment {

        IMAGE\_NAME = 'java-web'

        IMAGE\_TAG = 'latest'

        INTERNAL\_PORT = '8080'  // Tomcat default port

    }

    stages {

        stage('Checkout Code') {

            steps {

                git branch: 'main', credentialsId: 'git-creds', url: 'https://github.com/pc-nexc/contido-java.git'

            }

        }

        stage('Build Java Project') {

            steps {

                echo "Building Java application..."

                sh 'mvn clean package -DskipTests'

            }

        }

        stage('Build Docker Image') {

            steps {

                echo "Building Docker image with tag ${IMAGE\_TAG}..."

                sh "docker build -t ${IMAGE\_NAME}:${IMAGE\_TAG} ."

            }

        }

        stage('Remove Existing Container') {

            steps {

                echo "Removing existing container if it exists..."

                sh "docker rm -f ${params.CONTAINER\_NAME} || true"

            }

        }

        stage('Run Docker Container') {

            steps {

                echo "Running Docker container with persistent shell..."

                sh """

                    docker run -itd --name $CONTAINER\_NAME -p $HOST\_PORT:$INTERNAL\_PORT $IMAGE\_NAME /bin/sh

                """

            }

        }

        stage('Wait for Container') {

            steps {

                echo 'Waiting for container to initialize...'

                sh 'sleep 10'  // Adjust if needed

            }

        }

    }

    post {

        success {

            echo "✅ Deployment completed successfully."

            echo "App running at: http://your.jenkins.server.ip:${params.HOST\_PORT}"

        }

        failure {

            echo "❌ Build failed. Check logs above."

            script {

                sh "docker logs ${params.CONTAINER\_NAME} || true"

            }

        }

    }

}

pipeline {

    agent any

    environment {

        AWS\_REGION  = "ap-south-1"

        AWS\_CREDS   = "aws-creds"        // Jenkins AWS credential ID

        SECRET\_NAME = "ecr-secrets"      // JSON secret in AWS Secrets Manager

        IMAGE\_NAME  = "contido-dev-repo"

    }

    stages {

        stage('Checkout Code') {

            steps {

                git branch: 'dev', credentialsId: 'git-creds', url: 'https://github.com/ValipiReddy-web/terraform-aug2025.git'

            }

        }

        stage('Get ECR Info from Secrets Manager') {

            steps {

                withAWS(credentials: "${AWS\_CREDS}", region: "${AWS\_REGION}") {

                    script {

                        // Fetch secret JSON

                        def secretOutput = sh(

                            script: "aws secretsmanager get-secret-value --secret-id ${SECRET\_NAME} --region ${AWS\_REGION} --query SecretString --output text",

                            returnStdout: true

                        ).trim()

                        // Parse JSON values using jq

                        env.DOCKER\_USER = sh(script: "echo '${secretOutput}' | jq -r .username", returnStdout: true).trim()

                        env.DOCKER\_REG  = sh(script: "echo '${secretOutput}' | jq -r .registry", returnStdout: true).trim()

                        echo "✅ Docker info loaded: $DOCKER\_USER @ $DOCKER\_REG"

                    }

                }

            }

        }

        stage('Docker Login to ECR') {

            steps {

                withAWS(credentials: "${AWS\_CREDS}", region: "${AWS\_REGION}") {

                    sh """

                        # Use dynamic ECR login password

                        aws ecr get-login-password --region ${AWS\_REGION} | \

                        docker login --username $DOCKER\_USER --password-stdin $DOCKER\_REG

                    """

                }

            }

        }

        stage('Build & Push Docker Image') {

            steps {

                sh """

                    docker build -t ${IMAGE\_NAME} .

                    docker tag ${IMAGE\_NAME}:latest $DOCKER\_REG/${IMAGE\_NAME}:latest

                    docker push $DOCKER\_REG/${IMAGE\_NAME}:latest

                """

            }

        }

    }

    post {

        success {

            echo "✅ Docker image built and pushed successfully!"

        }

        failure {

            echo "❌ Build or push failed!"

        }

    }

}