For a DevOps role specialized in Jenkins, it's essential to focus on the following topics and concepts related to CI/CD pipelines:

**Key Topics to Focus On:**

1. **Jenkins Basics:**
   * Jenkins installation and configuration
   * Understanding Jenkins architecture (Master-Slave/Controller-Agent)
   * Jenkins plugins and their management
2. **Pipeline Concepts:**
   * Declarative vs. Scripted Pipelines
   * Pipeline stages, steps, and agents
   * Environment variables and credentials management
3. **Source Code Management:**
   * Integrating Jenkins with Git (GitHub, GitLab, Bitbucket)
   * Understanding webhooks for triggering builds
4. **Build Automation:**
   * Building and testing code using Jenkins
   * Managing dependencies and build tools (Maven, Gradle, npm)
5. **Deployment Strategies:**
   * Deployment to various environments (dev, staging, production)
   * Strategies for blue-green deployments, canary releases, and rolling updates
6. **Security:**
   * Managing credentials securely
   * Role-based access control and user permissions
7. **Monitoring and Troubleshooting:**
   * Configuring build notifications and alerts
   * Analyzing build logs and handling build failures
8. **Integration with Other Tools:**
   * Integrating Jenkins with Docker and Kubernetes
   * Using Jenkins with Terraform for Infrastructure as Code
9. **Pipeline as Code:**
   * Writing and managing Jenkinsfiles
   * Using Jenkins Shared Libraries for reusable code

**Basic Jenkins Pipeline Code**

Here’s a simple Jenkins pipeline script (Declarative Pipeline) to get you started:

pipeline {

agent any

environment {

// Define environment variables here

MY\_ENV\_VAR = 'value'

}

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo.git'

}

}

stage('Build') {

steps {

sh 'mvn clean install'

}

}

stage('Test') {

steps {

sh 'mvn test'

}

}

stage('Deploy') {

steps {

sh './deploy.sh'

}

}

}

post {

success {

echo 'Build and deployment succeeded!'

}

failure {

echo 'Build or deployment failed.'

}

always {

archiveArtifacts artifacts: '\*\*/target/\*.jar', allowEmptyArchive: true

}

}

}

**Real-Time Scenario-Based Pipelines**

**Scenario 1: Deploying a 3-Tier Application to AWS with Terraform and Docker**

pipeline {

agent any

environment {

AWS\_DEFAULT\_REGION = 'us-west-2'

}

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo.git'

}

}

stage('Build Docker Images') {

steps {

script {

docker.build('my-app-backend', 'backend/')

docker.build('my-app-frontend', 'frontend/')

}

}

}

stage('Terraform Plan') {

steps {

withCredentials([usernamePassword(credentialsId: 'aws-credentials', passwordVariable: 'AWS\_SECRET\_KEY', usernameVariable: 'AWS\_ACCESS\_KEY')]) {

sh 'terraform init'

sh 'terraform plan'

}

}

}

stage('Terraform Apply') {

steps {

withCredentials([usernamePassword(credentialsId: 'aws-credentials', passwordVariable: 'AWS\_SECRET\_KEY', usernameVariable: 'AWS\_ACCESS\_KEY')]) {

sh 'terraform apply -auto-approve'

}

}

}

stage('Deploy Docker Containers') {

steps {

script {

docker.image('my-app-backend').push('latest')

docker.image('my-app-frontend').push('latest')

}

}

}

stage('Post-Deployment Tests') {

steps {

sh './integration-tests.sh'

}

}

}

post {

success {

echo 'Deployment succeeded!'

}

failure {

echo 'Deployment failed.'

}

}

}

**Scenario 2: Blue-Green Deployment Strategy**

pipeline {

agent any

environment {

BLUE\_ENV = 'blue'

GREEN\_ENV = 'green'

CURRENT\_ENV = 'blue' // Set current environment

}

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo.git'

}

}

stage('Build') {

steps {

sh 'mvn clean package'

}

}

stage('Deploy Blue Environment') {

when {

environment name: 'CURRENT\_ENV', value: 'blue'

}

steps {

sh './deploy-to-blue.sh'

}

}

stage('Deploy Green Environment') {

when {

environment name: 'CURRENT\_ENV', value: 'green'

}

steps {

sh './deploy-to-green.sh'

}

}

stage('Switch Traffic') {

steps {

sh './switch-traffic.sh'

}

}

stage('Post-Deployment Verification') {

steps {

sh './verify-deployment.sh'

}

}

}

post {

success {

echo 'Blue-Green Deployment succeeded!'

}

failure {

echo 'Deployment failed.'

}

}

}

Inside **/switch-traffic.sh**

#!/bin/bash

# Define variables

OLD\_TARGET\_GROUP\_ARN="arn:aws:elasticloadbalancing:region:account-id:targetgroup/old-target-group-id"

NEW\_TARGET\_GROUP\_ARN="arn:aws:elasticloadbalancing:region:account-id:targetgroup/new-target-group-id"

ALB\_ARN="arn:aws:elasticloadbalancing:region:account-id:loadbalancer/app/your-alb-id"

LISTENER\_ARN="arn:aws:elasticloadbalancing:region:account-id:listener/app/your-alb-id/your-listener-id"

# Register new target group with the ALB listener

echo "Registering new target group with ALB listener..."

aws elbv2 modify-listener \

--listener-arn $LISTENER\_ARN \

--default-actions Type=forward,TargetGroupArn=$NEW\_TARGET\_GROUP\_ARN

# Deregister old target group from the ALB listener

echo "Deregistering old target group from ALB listener..."

aws elbv2 modify-listener \

--listener-arn $LISTENER\_ARN \

--default-actions Type=forward,TargetGroupArn=$OLD\_TARGET\_GROUP\_ARN

# Optional: Clean up old target group if no longer needed

# echo "Deleting old target group..."

# aws elbv2 delete-target-group --target-group-arn $OLD\_TARGET\_GROUP\_ARN

echo "Traffic has been switched to the new environment."

Inside /**deploy-to-green.sh**

#!/bin/bash

# Define variables

GREEN\_DEPLOYMENT\_NAME="my-app-green"

GREEN\_NAMESPACE="production"

DOCKER\_IMAGE="my-app:latest"

K8S\_CONFIG\_FILE="k8s-green-deployment.yaml"

# Apply Kubernetes configuration for the green environment

echo "Deploying to green environment..."

kubectl apply -f $K8S\_CONFIG\_FILE --namespace=$GREEN\_NAMESPACE

# Update Docker image in the deployment

echo "Updating Docker image..."

kubectl set image deployment/$GREEN\_DEPLOYMENT\_NAME my-app-container=$DOCKER\_IMAGE --namespace=$GREEN\_NAMESPACE

# Check deployment status

echo "Waiting for deployment to complete..."

kubectl rollout status deployment/$GREEN\_DEPLOYMENT\_NAME --namespace=$GREEN\_NAMESPACE

echo "Deployment to green environment completed successfully."

**Our Jenkins Code**

def COLOR\_MAP = [

    'SUCCESS': 'good',

    'FAILURE': 'danger',

]

pipeline {

    agent any

    tools {

        jdk "Java11"

        maven "Maven3"

    }

    stages {

        stage('fetch code') {

            steps {

                git branch: 'main', url: '<https://github.com/devopshydclub/vprofile-project.git>'

            }

        }

        stage('Build') {

            steps {

                sh 'mvn install -DskipTests'

            }

            post {

                success {

                    echo 'archiving artifacts'

                    archiveArtifacts artifacts: '\*\*/\*.war'

                }

            }

        }

        stage('unit test') {

            steps {

                sh 'mvn test'

            }

        }

        stage('checkstyle Analysis') {

            steps {

                sh 'mvn checkstyle:checkstyle'

            }

        }

        stage('sonar analysis') {

            environment {

                scannerHome = tool 'SonarQube'

            }

            steps {

                withSonarQubeEnv('sonar') {

                    sh '''

                    ${scannerHome}/bin/sonar-scanner -Dsonar.projectKey=vprofile \

                    -Dsonar.projectName=vprofile-repo \

                    -Dsonar.projectVersion=1.0 \

                    -Dsonar.login=${jenkins} \

                    -Dsonar.sources=src/ \

                    -Dsonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/ \

                    -Dsonar.junit.reportsPath=target/surefire-reports/ \

                    -Dsonar.jacoco.reportPath=target/jacoco.exec \

                    -Dsonar.java.checkstyle.reportPaths=target/checkstyle-result.xml

                    '''

                }

            }

        }

        stage("quality gates"){

            steps{

                timeout(time: 1, unit: 'HOURS'){waitForQualityGate abortPipeline: true}

            }

        }

        stage("UploadArtifact"){

            steps{

                nexusArtifactUploader(

                    nexusVersion: 'nexus3',

                    protocol: 'http',

                    nexusUrl: '172.31.24.8:8081',

                    groupId: 'QA',

                    version: "${env.BUILD\_ID}-${env.BUILD\_TIMESTAMP}",

                    repository: 'our-repo',

                    credentialsId:'Nexus-cred'

                    artifacts: [

                        [artifactId: 'vproapp',

                        classifier: '',

                        file: 'target/vprofile-v2.war',

                        type: 'war']

                    ]

                )

            }

        }

    }

    post{

        always{

            echo 'slack Notification'

            slackSend channel: '#jenkinscicd',

                color: COLOR\_MAP[currentBuild.currentResult],

                message: "\*${currentBuild.cirrentResult}:\*Job ${env.JOB\_NAME} build ${env.BUILD\_NUMBER} \n More info at: ${env.BUILD\_URL}}"

        }

    }

}