Jenkins

(@theshivanshvasu)

Jenkins Overview:

Purpose: Automation server for CI/CD in software development.
Language: Written in Java.
Key Features: Automation, Continuous Integration (CI), Continuous Delivery (CD).

Jenkins Architecture:

Master:

// Master controls Jenkins environment
// Schedules and monitors build jobs

Agent:

// Executes build scripts assigned by the master
// Reports back results to the master

Jenkins Workflow:

Source Code Management:

// Supports Git, SVN, Mercurial, etc.
// Developers commit code changes to the repository

Build Execution:

// Master assigns build job to agent
// Agent executes build script (e.g., Maven, Gradle)

Testing:

// Automated tests run during build process
// Test results recorded and reported back

Post-Build Actions:

// Notifications sent (email, Slack)
// Successful builds deployed automatically

Jenkins Use Cases:

CI/CD: Automate code integration to production deployment
Testing:Run unit, integration, and automated tests
Build Automation:Compile code, run scripts, create builds automatically
DevOps: Integrate with DevOps tools for streamlined processes

Example Jenkins Pipeline (Jenkinsfile):

```
pipeline {
    agent any
    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 'scp target/your-app.jar user@server:/path/to/deploy'
            }
        }
    }
}
```

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery.

Resources:

- https://www.youtube.com/watch?v=FX322RVNGj4&t=7679s
- https://www.youtube.com/watch?v=7KCS70sCoK0&t=718s
- https://www.youtube.com/watch?v=3a8KsB5wJDE
- https://www.youtube.com/c/CloudBeesTV
- https://www.youtube.com/watch?v=Ei_Nk14vruE
- https://www.youtube.com/watch?v=pMO26j2OUME&list=PLy7NrYWoggjw_LIiDK1LXdNN82uYuuuiC
-

Links:

- https://www.jenkins.io/doc/
- https://www.jenkins.io/sigs/docs/
- https://www.jenkins.io/user-handbook.pdf

Courses:

- https://www.udemy.com/course/devops-and-continuous-integration-with-jenkins-pipelines/
- https://www.udemy.com/course/learn-devops-ci-cd-with-jenkins-using-pipelines-and-docker/
- https://www.simplilearn.com/jenkins-certification-training-course
- https://www.udemy.com/course/jenkins-from-zero-to-hero/
- https://www.coursera.org/learn/uva-darden-continous-delivery-devops

Getting Started:

- Installing Jenkins
- Getting started with pipelines
- Jenkins on AWS
- Jenkins Installation in kubernetes with Helm

Jenkins:

- Create a jenkins setup in kubernetes using helm
- Configure jenkins using configuration as code.
- Run jenkins jobs as ephemeral pods in kubernetes which gets destroyed after the job
- Write your first jenkinsfile to run a basic test to calculate Pi
- Write parallel build steps to calculate pi as 10 jobs
- Configure a bitbucket pipeline to automatically run the above job when commit is pushed to bitbucket (you need to figure out how to set this up)
- Create an alert when job passed/failed to a slack channel

Solution:

- Create a jenkins setup in kubernetes using helm

```
 helm install jenkins jenkins/jenkins
 kubectl get secret jenkins -o jsonpath='{.data.jenkins-admin-password}'
| base64 --decode
 kubectl port-forward service/jenkins 8080:8080 --namespace default
```

```
NAME                                                    READY   STATUS             RESTARTS         AGE
jenkins-54488c88d-qvbgv                                 2/2     Running            0                6h1m
mynginx-nginx-ingress-controller-7d9f97fdb-prnbt        0/1     CrashLoopBackOff   38 (3m23s ago)   6h37m
mynginx-nginx-ingress-default-backend-bd4b5488f-mb6lb   1/1     Running            1 (5h59m ago)    6h37m
myrelease-mychart-555fd6ccd5-55wdf                      1/1     Running            1 (5h59m ago)    6h39m
nginx-deployment-7c79c4bf97-cjwqz                       1/1     Running            1 (5h59m ago)    7h3m
```

```
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

- Configure jenkins using configuration as code.

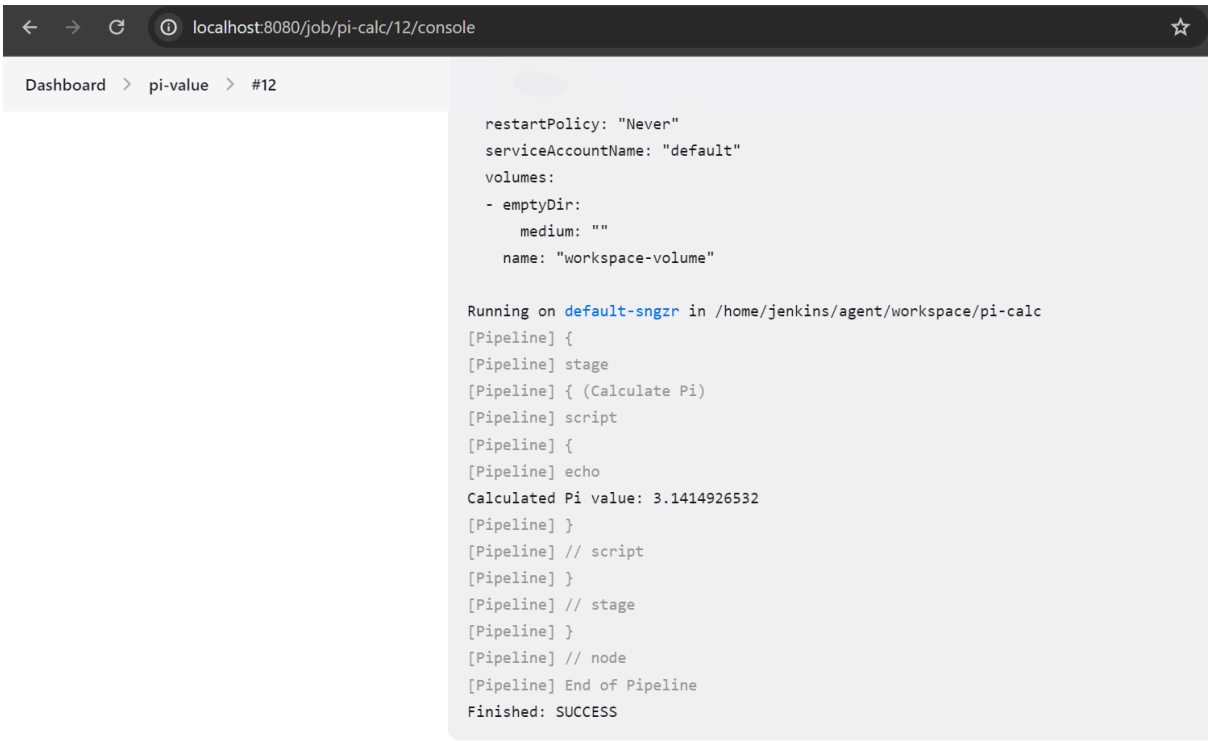
```
jenkins:
  systemMessage: "Welcome to Jenkins configured with JCasC!"
  securityRealm:
    local:
      allowsSignup: false
      users:
        - id: admin
          password: admin_password
  authorizationStrategy:
    loggedInUsersCanDoAnything:
      allowAnonymousRead: false
  unclassified:
    location:
      url: "http://jenkins.example.com/"
  tool:
    git:
      installations:
        - name: "Default Git"
          home: "/usr/bin/git"
  credentials:
    system:
      domainCredentials:
        - credentials:
            - basicSSHUserPrivateKey:
                scope: SYSTEM
                id: "git-ssh-key"
                username: "git"
                privateKeySource:
                  directEntry:
                    privateKey: |
                      -----BEGIN RSA PRIVATE KEY-----
                      [Your SSH private key]
                      -----END RSA PRIVATE KEY-----
```

- Run jenkins jobs as ephemeral pods in kubernetes which gets destroyed after the job

```
pipeline {
    agent {
        kubernetes {
            yaml """
            apiVersion: v1
            kind: Pod
            metadata:
              labels:
                jenkins: ephemeral
```

```
            spec:
              containers:
                - name: jnlp
                  image: jenkins/jnlp-slave
                  tty: true
            """
        }
    }
    stages {
        stage('Build') {
            steps {
                echo 'Hello World!'
            }
        }
    }
}
```

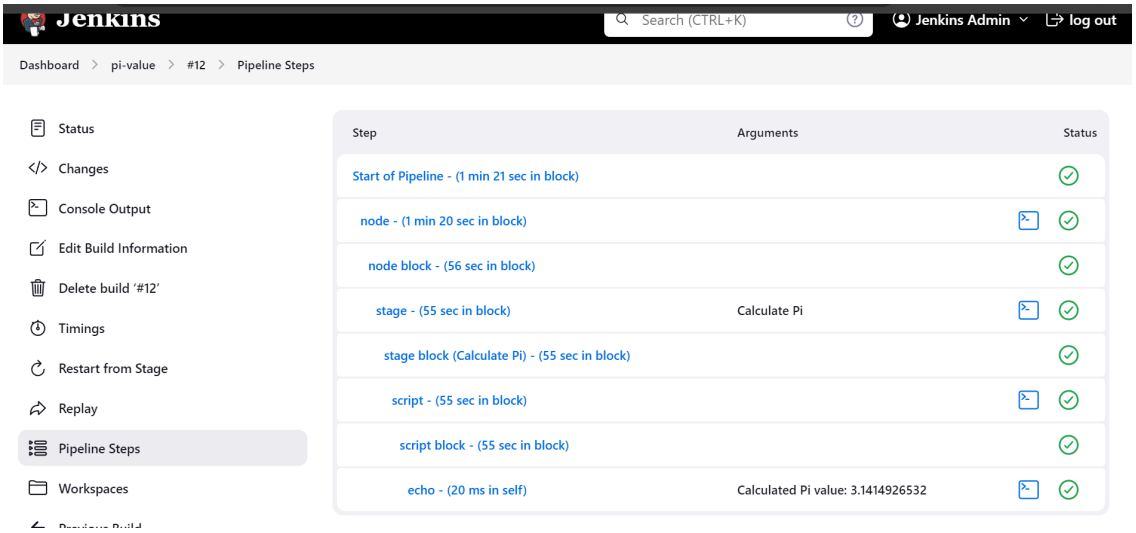
- Write your first jenkinsfile to run a basic test to calculate Pi




```
pipeline {
    agent any

    stages {
        stage('Calculate Pi') {
```

```
            steps {
                script {
                    def pi = 0
                    def n = 10000 // Number of iterations (adjust for
accuracy)
                    for (int i = 0; i < n; i++) {
                        def sign = (i % 2 == 0) ? 1 : -1
                        pi += sign * (4.0 / (2 * i            steps {
                script {
                    // Calculate Pi using bc command
                    def piValue = sh(script: 'echo "scale=10; 4*a(1)" |
bc -lq', returnStdout: true).trim()
                    echo "The value of Pi is: ${piValue}"
                }
            }
        }
    }
}
```



- Configure a bitbucket pipeline to automatically run the above job when commit is pushed to bitbucket (you need to figure out how to set this up)

```
Configure Bitbucket Pipeline:

Open your Bitbucket repository.
Navigate to Settings > Pipelines > Repository settings.
Create or update the bitbucket-pipelines.yml file in your repository.
Define Pipeline Steps:

Use a script block in the bitbucket-pipelines.yml file.
Install required tools (e.g., curl) for triggering Jenkins job.
Trigger the Jenkins job using curl or HTTP request.
Example Configuration:
Here's an example bitbucket-pipelines.yml file:
```

```
yaml
Copy code
image: node:14.17.6  # Use an appropriate Docker image

pipelines:
  branches:
    master:
      - step:
          name: Trigger Jenkins Job
          script:
            - apt-get update && apt-get install -y curl  # Install curl
            - curl -X POST "JENKINS_JOB_URL/build?token=YOUR_AUTH_TOKEN"
Replace "JENKINS_JOB_URL" with the actual URL of your Jenkins job that accepts remote
triggers, and replace "YOUR_AUTH_TOKEN" with the authentication token configured in
Jenkins.

Commit Changes:

Save the changes to the bitbucket-pipelines.yml file.
Commit and push the changes to your Bitbucket repository.
Test Pipeline Trigger:

Make a commit and push it to the specified branch (e.g., master).
Check Jenkins to ensure that the job is triggered automatically.
```

- Create an alert when job passed/failed to a slack channel

```
Install Jenkins Slack Plugin:

Use Jenkins Plugin Manager.
Search and install "Slack Notification" plugin.
Configure Slack Integration:

Go to Jenkins > Manage Jenkins > Configure System.
In "Slack" section, add Slack Team details (domain, token).
Test connection to verify.
Update Jenkins Job:

Open job configuration.
Go to "Post-build Actions" > "Slack Notifications".
Configure Slack settings (team, channel, token).
Choose notification options (build start, failure, success).
Test Notifications:

Trigger a build in Jenkins (manually or via commit).
Check the configured Slack channel for notifications.
Verify notifications for job start, failure, and success.
```