Jenkins

What Is a CI/CD Pipeline?

A **CI/CD pipeline** automates the process of:

- **Continuous Integration (CI)**: Code is built and tested every time it's pushed to a shared repo.
- **Continuous Delivery/Deployment (CD)**: The app is packaged and deployed automatically to staging or production

What is Jenkins?

Jenkins is an open-source automation server used to **build, test, and deploy** software.

Key Features:

- Supports **Continuous Integration (CI)** and **Continuous Delivery (CD)**
- Automates build/test pipelines for any language or platform
- Integrates with 1000+ tools like GitHub, Docker, Kubernetes, Slack, etc.
- Has a large plugin ecosystem
- Supports distributed builds (Master-Agent architecture)

Types of Jenkins Jobs

Type	Description
Freestyle Project	GUI-based job configuration. Good for simple builds, shell scripts, or basic tasks
Pipeline Project	Code-defined CI/CD pipelines using Jenkinsfile. Scalable and reusable

---

Freestyle vs. Pipeline

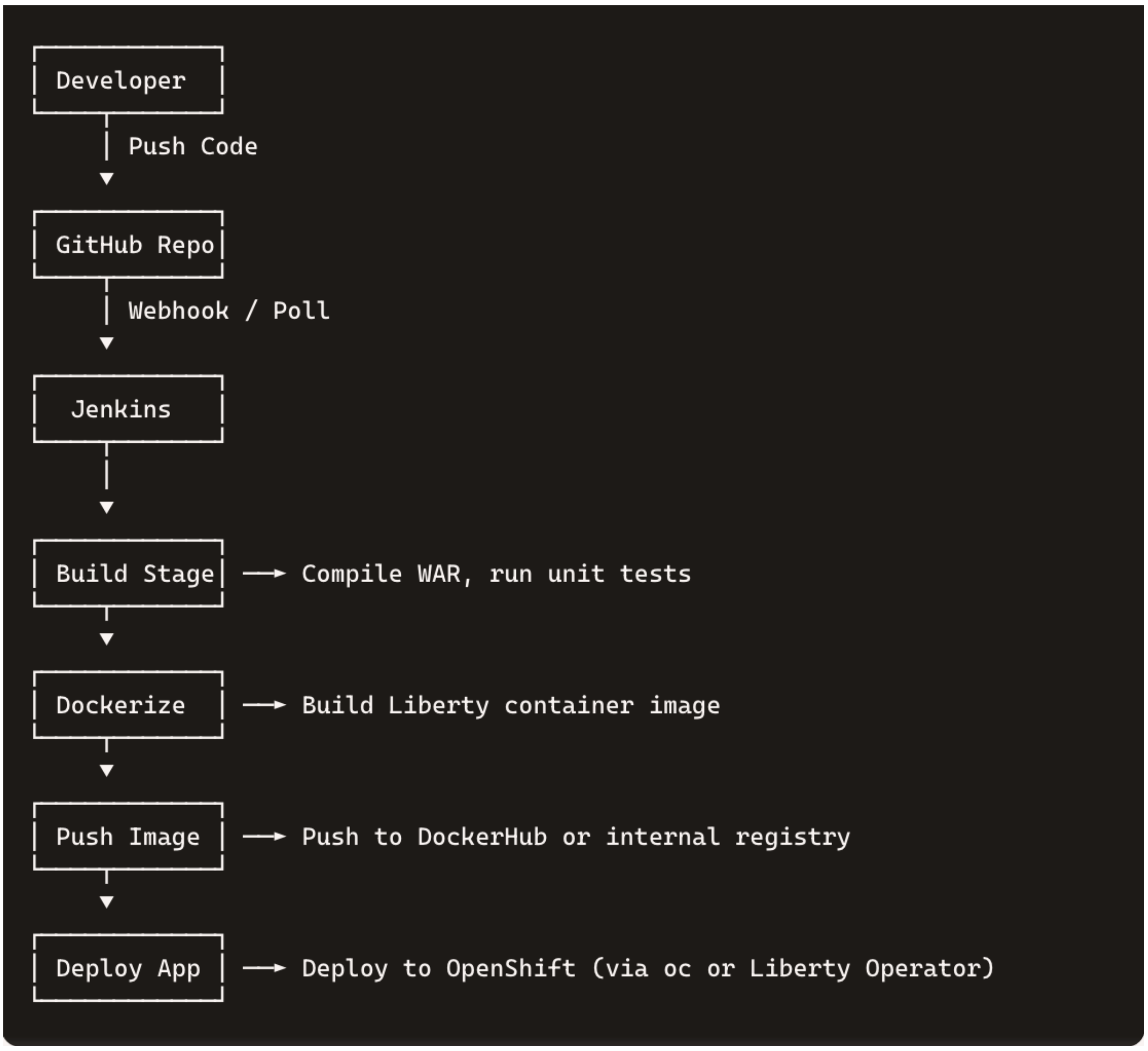
Feature	Freestyle	Pipeline (Declarative/Scripted)
Setup	GUI-based	Code-defined (Jenkinsfile)
Complexity	Simple	Can handle complex workflows
Version Control Friendly	❌ Not stored in Git	✅ Stored in Git alongside code
Reusability	Low	High (shareable templates, stages)
Portability	Harder to port	Easy to copy/maintain across environments

Feature	Freestyle	Pipeline (Declarative/Scripted)
Plugins	Depends on plugins	More flexible, but needs syntax knowledge

Alternatives to Jenkins

Tool	Best Use Case	Key Features
GitHub Actions	GitHub-hosted repos	Easy CI/CD built into GitHub, YAML-based
GitLab CI/CD	Full DevOps platform	Built-in CI/CD with GitLab repos
CircleCI	Fast cloud-based CI/CD	YAML-based, great for container workflows
Travis CI	Open-source projects	Simple YAML config, GitHub-native
Azure DevOps Pipelines	Enterprise-ready CI/CD	Supports Git, TFVC, containers, hybrid deploy
TeamCity (JetBrains)	IntelliJ/enterprise-friendly builds	Strong UI, test reporting
Tekton (Kubernetes-native)	Cloud-native CI/CD for K8s	Flexible pipelines, YAML CRDs
ArgoCD (for CD only)	GitOps-based deployment for K8s	Declarative deployment model
Spinnaker	Multi-cloud deployment pipelines	Great for large-scale delivery workflows

Jenkins CI/CD Pipeline Architecture



```
┌───────────┐
│ Developer │
└───────────┘
     │ Push Code
     ▼
┌───────────┐
│GitHub Repo│
└───────────┘
     │ Webhook / Poll
     ▼
┌───────────┐
│  Jenkins  │
└───────────┘
     │
     ▼
┌───────────┐
│Build Stage│ ──▶ Compile WAR, run unit tests
└───────────┘
     ▼
┌───────────┐
│ Dockerize │ ──▶ Build Liberty container image
└───────────┘
     ▼
┌───────────┐
│ Push Image│ ──▶ Push to DockerHub or internal registry
└───────────┘
     ▼
┌───────────┐
│ Deploy App│ ──▶ Deploy to OpenShift (via oc or Liberty Operator)
└───────────┘
```

GitHub Setup

- Create a repo: `liberty-hello-world`
- Add:
  - `hello.war` or source code
  - `server.xml`
  - `Dockerfile`
  - `Jenkinsfile`
  - `openshift/deployment.yaml`

Jenkins Overview

- **Jenkins** is a leading open-source automation tool for **continuous integration and continuous delivery (CI/CD)**.
- It supports deployments across:
  - 🪟 **Windows**
  - 🐧 **Linux**
  - 📦 **Containers**
  - ☸️ **Kubernetes**

Jenkins Installation

Step	Instruction
Download Jenkins	Use official repo: https://pkg.jenkins.io/debian-stable/
Start Jenkins	Access via browser: `http://<external-ip>:8080`
Unlock Jenkins	Retrieve unlock key: `cat /var/lib/jenkins/secrets/initialAdminPassword`
Set Up Admin User	Create username & password after unlock
Plugin Installation	Choose **Suggested Plugins** (Git, Maven, Pipeline, Docker, etc.)

Freestyle Project Setup (Manual CI)

Use for scripting tasks, basic compilation, shell commands

1. Login to Jenkins UI
2. Click **New Item** → Name: `vanakkam-world` → Choose **Freestyle Project**
3. Configure build steps:
   - Git repo: `https://github.com/mgsgoms/vanakkam-world`
   - Build step: Shell commands (`echo`, `mkdir`, etc.)
4. Trigger manually via **“Build Now”**

Integration Examples

Tool	Usage
GitHub	SCM trigger, auto-build on code push
Maven	Compile Java projects with `pom.xml`
Tomcat	Deploy WAR to Tomcat app server on Linux

Tomcat Setup Steps

1. Download & install Tomcat on VM (default port: `8080`)
2. Configure user in `tomcat-users.xml` with admin role
3. Bind Tomcat to external IP (not just `127.0.0.1`)
4. Start Tomcat and verify access: `http://<vm-ip>:8080`

Jenkinsfile & Pipeline Syntax

The **Jenkinsfile** defines steps to automate CI/CD and can follow two styles:

Syntax Type	Target Audience	Format
Scripted Pipeline	Developers	`node {}` blocks in Groovy
Declarative Pipeline	DevOps / Infra teams	`pipeline {}` structured

Declarative Sections

Directive	Purpose
`pipeline`	Start of pipeline
`agent`	Where to run the job
`stages`	Contains multiple stages
`stage`	Individual named stage
`steps`	Shell or build steps

Optional Directives

- `retry`, `timeout`, `post`, `credentials`
- `cron`, `pollSCM`, `parallel`
- `tools`, `input`, `environment`

Sample Pipelines

Single Stage Pipeline

```
pipeline {
  agent any
  stages {
    stage('Code') {
      steps {
        sh 'echo Hello'
      }
    }
  }
}
```

Multi-Stage Pipeline

```
pipeline {
  agent any
  stages {
    stage('Build') { steps { echo 'Building...' } }
    stage('Test')  { steps { echo 'Testing...' } }
    stage('Deploy'){ steps { echo 'Deploying...' } }
  }
}
```

Windows-Friendly Pipeline

```
pipeline {
  agent any
  stages {
    stage('Setup') { steps { bat 'type nul > abc.txt' } }
    stage('Create') { steps { bat 'mkdir test' } }
  }
}
```

Retry + Timeout Example

```
pipeline {
  agent any
  stages {
    stage('Deploy') {
      steps {
        retry(2) {
          timeout(time: 2, unit: 'MINUTES') {
            sh 'kubectl create -f
https://github.com/mgsgoms/myproject/prometheus.yaml'
          }
        }
      }
    }
  }
}
```

Credentials Management

1. Jenkins UI → **Manage Credentials** → Add new
   - Type: `Secret Text`, `Username & Password`, `File`
   - ID: `mysecret`
2. Use in Pipeline:

```
pipeline {
  agent any
  environment {
    secret = credentials('mysecret')
  }
  stages {
    stage('Example') {
      steps {
        sh 'echo $secret'
      }
    }
  }
}
```

Trigger Other Jobs

```
pipeline {
  agent any
  stages {
    stage('Trigger') {
      steps {
        build('job1')
        build('job2')
      }
    }
  }
}
```