# Jenkins Tutorial – A Comprehensive Guide

## What is Jenkins?

Jenkins is an **open-source automation server** used for **Continuous Integration (CI) and Continuous Deployment (CD)**. It helps automate software development's **build, test, and deployment** phases.

- **Official Website:** Jenkins
- **Developed in:** Java
- **Supports:** CI/CD, pipeline as code, automation testing, integrations with various tools

---

# 1️⃣ Why Use Jenkins?

✅ **Automates Build & Deployment** – Saves time and effort
✅ **Supports CI/CD** – Automates testing and deployment workflows
✅ **Integrates with DevOps Tools** – Works with Git, Docker, Kubernetes, etc.
✅ **Scalability** – Distributed builds using Jenkins agents
✅ **Plugins** – 1000+ plugins for various integrations

---

# 2️⃣ Jenkins Architecture

### 🔹 Components of Jenkins

1. **Jenkins Master** – Main Jenkins server that handles UI, job scheduling, and build execution
2. **Jenkins Slave (Agent)** – Executes build jobs assigned by the master
3. **Jobs/Pipelines** – Define CI/CD workflows
4. **Plugins** – Extend Jenkins functionality (Git, Docker, Kubernetes, etc.)

---

# 3️⃣ Installation of Jenkins

## 🔹 Pre-requisites

✅ **Java (JDK 11 or later)**
✅ **Git (if using Git integration)**

## 🔹 Installing Jenkins on Windows

1. **Download Jenkins:** Jenkins Download Page
2. **Install Jenkins:** Run `.msi` or `.war` file

**Start Jenkins:**

```
java -jar jenkins.war --httpPort=8080
```

3. **Access Jenkins:** Open `http://localhost:8080/` in the browser

## 🔹 Installing Jenkins on Linux (Ubuntu)

```
sudo apt update
sudo apt install openjdk-11-jdk -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
echo "deb http://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins

```

✅ **Access Jenkins at:** `http://localhost:8080/`

---

# 4️⃣ Jenkins Dashboard Overview

Once Jenkins is installed and running:

1. **Unlock Jenkins** – Use the password from `/var/lib/jenkins/secrets/initialAdminPassword`
2. **Install Plugins** – Recommended plugins (Git, Pipeline, Docker, etc.)
3. **Create Admin User** – Set up a new Jenkins admin account
4. **Jenkins Dashboard** – Main interface where jobs and builds are managed

---

# 5️⃣ Creating a Simple Jenkins Job

## 🔷 Steps to Create a Jenkins Job

1. **Go to Jenkins Dashboard → Click on "New Item"**
2. **Select "Freestyle project" → Click OK**
3. **Configure the job:**
    - **Source Code Management (SCM)** → Select Git and add repository URL
    - **Build Triggers** → Choose how the job should be triggered (e.g., Poll SCM, Webhook)
    - **Build Steps** → Add a script (Maven, Gradle, Shell)

## 🔷 Example: Build Job with Shell Script

```
echo "Building the project..."
mvn clean install
echo "Build completed!"
```

4. **Save & Build** → Click **"Build Now"**

✅ **Check Console Output** for build logs

---

# 6️⃣ Jenkins Pipeline (CI/CD Automation)

Jenkins **Pipelines** allow defining the entire CI/CD process using code (`Jenkinsfile`).

## 🔷 Types of Pipelines

- **Declarative Pipeline** – Uses a structured format
- **Scripted Pipeline** – Uses Groovy scripting for advanced scenarios

---

# 🔷 Example: Declarative Jenkins Pipeline

Create a `Jenkinsfile` in your repository:

```
pipeline {
    agent any  // Run on any available agent
    stages {
        stage('Clone Repository') {
            steps {
                git 'https://github.com/example/repository.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'scp target/*.jar user@server:/deploy/path/'
            }
        }
    }
}
```

✅ **Steps in Pipeline:**

1. **Clone Repository** – Pulls code from Git
2. **Build** – Uses Maven to build
3. **Test** – Runs unit tests
4. **Deploy** – Deploys artifact to a remote server

---

# 7️⃣ Jenkins Integration with GitHub (Webhook for Automation)

## 🔷 Steps

1. **Install "Git Plugin" in Jenkins**
2. **Go to "Manage Jenkins" → "Configure System" → Add GitHub Webhook URL**
3. **In GitHub Repo:**
    - Go to **Settings → Webhooks**

Add Jenkins URL:

```
http://your-jenkins-url/github-webhook/
```

    - Choose **"Just the push event"**

✅ **Now, Jenkins will trigger builds on every GitHub push!**

---

# 8️⃣ Jenkins Integration with Docker

Jenkins can run builds inside **Docker containers** for isolated environments.

## 🔷 Steps

1. **Install Docker on Jenkins Server**
2. **Install "Docker Pipeline Plugin" in Jenkins**
3. **Use Docker in Pipeline:**

```
pipeline {
    agent {
        docker { image 'maven:3.8.5-openjdk-11' }
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn clean install'
            }
        }
    }
}
```

✅ **This runs the build inside a Docker container with Maven!**

---

# 9️⃣ Jenkins Distributed Builds (Master-Slave Setup)

Jenkins supports **distributed builds** using **slave agents**.

### 🔷 Steps to Add a Jenkins Agent

1. **Go to "Manage Jenkins" → "Manage Nodes and Clouds"**
2. **Click "New Node" → Choose "Agent"**
3. **Configure Agent:**
    - Set agent **name, number of executors**
    - Choose launch method: **SSH, JNLP, or Docker**

**Start the agent** using:

```
java -jar agent.jar -jnlpUrl http://jenkins-server/computer/node-name/slave-agent.jnlp -secret SECRET_KEY

```

✅ **Now, builds can be distributed across multiple machines!**

---

# 🔟 Jenkins Best Practices

✅ Use **Pipelines Instead of Freestyle Jobs**
✅ Implement **Webhooks for Auto Builds**
✅ Use **Jenkins Agents** for Scalability
✅ Run Builds in **Docker Containers**
✅ Use **Parameterized Builds** for flexibility
✅ Configure **RBAC (Role-Based Access Control) for Security**

---

# Conclusion

Jenkins is a **powerful CI/CD tool** that automates software development workflows.
✅ **Key Features:** CI/CD Pipelines, GitHub/Docker Integration, Master-Slave Architecture
✅ **Next Steps:** Learn **Advanced Pipelines**, integrate with **Kubernetes**, and explore **Jenkins X** 🚀

Let's break down **advanced CI/CD pipeline setup** and **specific integrations** based on your needs.

### 🔷 Advanced CI/CD Pipeline Setup - What We Can Cover

✅ **Multi-Branch Pipelines** (for GitHub/GitLab repos)
✅ **Parameterized Pipelines** (for flexible builds)
✅ **Parallel Test Execution** (faster execution)
✅ **Integration with Docker & Kubernetes** (containerized builds & deployments)
✅ **Secrets Management with Vault or Akeyless**
✅ **Notifications (Slack, Email, Teams) on Build Failures**

### 🔷 Specific Integrations - What We Can Set Up

✔ **Playwright + Jenkins for UI Automation**
✔ **Selenium + TestNG/Cucumber Integration**
✔ **Rest Assured API Testing in Jenkins**
✔ **Cypress Integration with Jenkins (for JS Automation)**
✔ **AWS/GCP Deployment with Jenkins Pipelines**
✔ **Jenkins + SonarQube for Code Quality Analysis**
✔ **Jenkins + Docker/Kubernetes for Scalable Deployments**

### 🤔 What do you want to focus on first?

- CI/CD pipeline for **Test Automation**?
- Jenkins + **Docker/Kubernetes** setup?
- CI/CD pipeline for **Web App Deployment**?

---

**NOTE: If you have jenkins already install and you forget the password, you can uninstall and set the credential again using below steps:**

### Uninstalling Jenkins on Mac & Installing it from Scratch
**[Optional Section]**

## 1️⃣ Uninstall Jenkins from Mac

If you installed Jenkins via **Homebrew**, follow these steps:

### 🔸 Step 1: Stop Jenkins Service

Run the following command to stop Jenkins:

```
brew services stop jenkins-lts
```

### 🔸 Step 2: Uninstall Jenkins

```
brew uninstall jenkins-lts
```

### 🔶 Step 3: Remove Jenkins Configuration and Data

To completely remove all Jenkins data, delete the Jenkins home directory:

```
sudo rm -rf /Users/Shared/Jenkins
sudo rm -rf ~/.jenkins
```

If Jenkins was installed as a **launch agent**, remove the launch script:

```
sudo rm -rf /Library/LaunchDaemons/org.jenkins-ci.plist
```

### 🔶 Step 4: Remove Jenkins Logs (Optional)

```
sudo rm -rf /var/log/jenkins.log
```

✅ **Jenkins is now completely removed from your Mac!**

---

## 2️⃣ Installing Jenkins from Scratch on Mac

### 🔶 Step 1: Install Java (if not installed)

Jenkins requires **Java 11 or later**. Check your Java version:

```
java -version
```

If Java is not installed, install OpenJDK:

```
brew install openjdk@11
echo 'export PATH="/opt/homebrew/opt/openjdk@11/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

---

### 🔶 Step 2: Install Jenkins via Homebrew

```
brew install jenkins-lts
```

---

### 🔶 Step 3: Start Jenkins

```
brew services start jenkins-lts
```

By default, Jenkins runs on **port 8080**. Check if it is running:

```
brew services list
```

---

### 🔶 Step 4: Unlock Jenkins

Open your browser and go to:

```
http://localhost:8080
```

Jenkins will ask for an **initial administrator password**.
Find it using:

```
cat /Users/Shared/Jenkins/Home/secrets/initialAdminPassword
```

---

### 🔶 Step 5: Install Plugins & Set Up Admin User

- Select **"Install Suggested Plugins"**
- Create an **Admin User**
- Configure **Jenkins Home Directory** (Default: `/Users/Shared/Jenkins`)

---

### 🔶 Step 6: Verify Jenkins Installation

To check the Jenkins version:

```
jenkins --version
```

✅ **Jenkins is now installed and running from scratch on your Mac!**

---

## Bonus: Useful Commands

✔ **Restart Jenkins:**

```
brew services restart jenkins-lts
```

✔ **Stop Jenkins:**

```
brew services stop jenkins-lts
```

✔ **Check Jenkins Logs:**

```
tail -f /Users/Shared/Jenkins/Home/logs/jenkins.log
```

---

### 🚀 Next Steps

- **Do you want to set up a CI/CD pipeline now?**
- **Need help with Jenkins integration (GitHub, Docker, Playwright, Cypress, etc.)?**

---

## 🔷 1️⃣ Check the Correct Jenkins Home Directory

Jenkins stores its data in a **home directory**, which may vary depending on how it was installed. Run this command to find the actual location:

```
echo $JENKINS_HOME
```

If nothing appears, try:

```
brew info jenkins-lts
```

The output may show a path like `/usr/local/var/jenkins_home` or `/opt/homebrew/var/jenkins_home`.

Once you find the correct path, navigate to it and check for the secrets folder:

```
ls -l /usr/local/var/jenkins_home/secrets/
```

Try retrieving the password using:

```
cat /usr/local/var/jenkins_home/secrets/initialAdminPassword
```

---

## 🔷 2️⃣ Check Jenkins Logs for Password

If the file is missing, Jenkins may have logged the password during the first startup. Check the logs:

```
tail -f /usr/local/var/log/jenkins.log
```

or

```
tail -f /opt/homebrew/var/log/jenkins.log
```

Look for a line that says:

```
************************************************************
Jenkins initial setup is required. An admin user has been created
and a password is required to proceed.
Please use the following password to proceed to installation:

xxxxx-xxxxx-xxxxx-xxxxx

This may also be found at:
/path/to/jenkins_home/secrets/initialAdminPassword
************************************************************
```

Copy the password and use it to unlock Jenkins.

---

## 🔷 3️⃣ Restart Jenkins & Generate a New Password

If the file is missing, you can **restart Jenkins** and regenerate the password:

```
brew services restart jenkins-lts
```

Then check again:

```
cat /usr/local/var/jenkins_home/secrets/initialAdminPassword
```

---

## 🔷 4️⃣ Reset Jenkins Admin Password (If All Else Fails)

If you **can't retrieve the password**, you can **disable security** and create a new admin user.

### 🔶 Step 1: Disable Security in Jenkins

```
brew services stop jenkins-lts
```

---

***Happy Learning!***