# Jenkins Documentation

## Basics of Jenkins Automation Server

Jenkins is an open-source automation server used to automate tasks involved in building, testing, and deploying software. It is widely used for implementing Continuous Integration and Continuous Delivery (CI/CD).

### Key Features

- Automates repetitive tasks in software development  
- Integrates with various development and deployment tools  
- Supports pipeline-as-code through Jenkinsfile  
- Highly customizable with plugins

### Core Concepts

|  |  |
| --- | --- |
| Concept | Description |
| Job/Project | A unit of work executed by Jenkins |
| Pipeline | A script-defined workflow used for CI/CD processes |
| Node/Agent | Machines that run jobs assigned by Jenkins |
| Master/Controller | Controls scheduling and delegating tasks to agents |
| Plugin | Adds additional functionality to Jenkins |

## 1. Common CI/CD Deployment Method Using Jenkins

### Continuous Integration (CI)

Continuous Integration is a practice where developers frequently push code into a shared repository. Jenkins helps by automatically pulling the latest code, building the application, running automated tests, and generating build artifacts.

### Continuous Delivery/Deployment (CD)

After integration, Jenkins can deploy applications to staging or production servers, integrate with automation tools like Ansible or Docker, and notify teams about deployment statuses.

### Typical CI/CD Pipeline Flow

1. Developer pushes code to version control (GitHub/GitLab/Bitbucket)  
2. Jenkins detects the update  
3. Jenkins pulls the latest code  
4. Builds the application using a build tool (Maven, npm, etc.)  
5. Runs tests  
6. Packages and stores artifacts  
7. Deploys the application to the desired environment  
8. Sends a notification to the team

### Common Plugins Used

|  |  |
| --- | --- |
| Plugin | Purpose |
| Git | Pulls source code from repositories |
| Maven/Gradle | Builds Java-based projects |
| Docker | Creates and deploys containers |
| Pipeline | Enables writing Jenkins pipelines |
| Slack/Email | Sends build notifications |

## 2. Install Java (OpenJDK 17)

Jenkins requires Java to run. Install Java 17 using the following command:  
  
Amazon Linux 2023:  
sudo dnf install java-17-amazon-corretto -y  
  
Amazon Linux 2:  
sudo yum install java-17-amazon-corretto -y  
  
Verify installation:  
java -version

## 3. Install Jenkins

Add Jenkins repository and install Jenkins using the following commands:  
  
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo  
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key  
sudo yum install jenkins -y  
  
Enable and start Jenkins:  
sudo systemctl enable jenkins  
sudo systemctl start jenkins  
  
Check Jenkins status:  
sudo systemctl status jenkins

## 4. Access Jenkins Dashboard

Open Jenkins by visiting: http://<your-server-public-ip>:8080  
  
Retrieve admin password:  
sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
  
Paste the password in the browser and install the suggested plugins.

## 5. Install and Configure Git

Install Git:  
sudo yum install git -y  
  
Verify installation:  
git --version  
  
Configure Git (optional):  
git config --global user.name "Your Name"  
git config --global user.email "your-email@example.com"

## 6. Integrate Jenkins with GitHub

1. Go to GitHub → Settings → Developer Settings → Personal Access Tokens → Tokens (classic)  
2. Click 'Generate new token' and select scopes: repo, workflow, admin:repo\_hook  
3. Copy the token  
4. In Jenkins → Manage Jenkins → Credentials → Global → Add Credentials  
5. Select 'Secret Text' and paste the token  
6. Give ID name: github-token

## 7. Create index.html

Create a Jenkinsfile in the root of your GitHub project with the following content:

Echo “Hello from Jenkins”

## 8. Push Jenkinsfile to GitHub

## git add index.html

git commit -m "Initial index.html for Jenkins deployment"

git push origin master

9. Configure Jenkins Pipeline Job

1. Jenkins Dashboard → New Item → Enter name 'linux-cicd-pipeline'  
2. Select 'Pipeline' → OK  
3. Under Pipeline section:  
 - Definition: Pipeline script from SCM  
 - SCM: Git  
 - Repository URL: https://github.com/itzzmeanjali/simple-web-deploy.git  
 - Credentials: Select GitHub token  
 - Branch: master  
4. Save and click 'Build Now'

## 10. Troubleshooting Build Failures

- Check Console Output for red lines or 'ERROR:' messages.  
- Common issues:  
 \* Wrong Git repo URL or branch name  
 \* Jenkins permission errors  
 \* Jenkinsfile syntax errors  
  
Allow Jenkins to run sudo commands:  
sudo visudo  
Add line:  
jenkins ALL=(ALL) NOPASSWD: ALL

## 11. Verify Deployment

Visit http://<your-server-public-ip>/ to verify that your PHP app is deployed successfully.

## 12. Optional: Webhook Setup for Automation

1. In GitHub repo → Settings → Webhooks → Add Webhook  
2. Payload URL: http://<jenkins-public-ip>:8080/github-webhook/  
3. Content type: application/json  
4. Trigger: Just the push event  
5. Save

## 13. CI/CD Flow Summary

1. Developer commits code → GitHub  
2. Jenkins fetches the latest code  
3. Jenkins builds and deploys it to /var/www/html/  
4. The PHP app is live instantly

## 14. Conclusion

You have successfully configured Jenkins on Amazon Linux to automate the CI/CD process for PHP web applications. This setup enables efficient continuous deployment and integration for faster delive  