**Jenkins**

Jenkins is a powerful and widely-used open-source automation server, primarily known for \*\*Continuous Integration (CI)\*\* and \*\*Continuous Delivery (CD)\*\*. Jenkins helps automate the repetitive tasks of building, testing, and deploying applications. While Jenkins is fairly easy to start with, it has many advanced features and deep concepts that provide scalability, flexibility, and integration with other tools. Here’s an overview of Jenkins' deeper concepts:

---

### 1. \*\*Jenkins Pipeline\*\*

A \*\*Pipeline\*\* is a fundamental concept in Jenkins that defines the process of building, testing, and deploying your application. Jenkins Pipelines are written as \*\*Jenkinsfile\*\*, a script that defines the sequence of stages and steps involved in the CI/CD process.

There are two types of Pipelines:

- \*\*Declarative Pipeline\*\*: A more structured, user-friendly syntax to define pipelines.

- \*\*Scripted Pipeline\*\*: A more flexible but complex pipeline syntax using Groovy scripts.

#### Key Elements of a Pipeline:

- \*\*Stages\*\*: A pipeline is divided into stages such as `Build`, `Test`, and `Deploy`.

- \*\*Steps\*\*: Each stage contains steps, such as executing shell commands, running tests, or triggering other jobs.

- \*\*Agent\*\*: Defines where the pipeline will run. It can be a specific machine, Docker container, or Kubernetes pod.

\*\*Declarative Example\*\*:

```groovy

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

sh 'mvn clean install'

}

}

stage('Test') {

steps {

echo 'Testing...'

sh 'mvn test'

}

}

stage('Deploy') {

steps {

echo 'Deploying...'

sh 'ansible-playbook deploy.yml'

}

}

}

}

```

---

### 2. \*\*Jenkins Nodes and Executors\*\*

- \*\*Jenkins Node\*\*: A machine that is part of the Jenkins environment, either the master or a \*\*slave\*\* (also called a \*\*worker\*\*).

- \*\*Jenkins Master\*\*: The central controller that manages the scheduling of jobs, monitors build progress, and serves the web interface.

- \*\*Jenkins Slave/Agent\*\*: Machines that can be configured to run jobs in parallel with the master node. Slaves help distribute workload, improving scalability.

A \*\*node\*\* can run one or more \*\*executors\*\*. An executor is a computational resource on a node that is capable of executing tasks. Each executor runs a single build at a time, and jobs are distributed to available executors.

---

### 3. \*\*Jenkins Blue Ocean\*\*

\*\*Blue Ocean\*\* is a modern and visually appealing user interface for Jenkins. It provides an intuitive experience for creating and managing pipelines, visualizing build flows, and identifying problems in your CI/CD process.

Some features of Blue Ocean:

- \*\*Pipeline Visualization\*\*: Visual representation of pipeline stages and steps.

- \*\*Simplified Pipeline Creation\*\*: Streamlined setup and visualization for declarative pipelines.

- \*\*Improved Logs and Error Handling\*\*: Easier-to-read logs with rich feedback.

---

### 4. \*\*Jenkinsfile\*\*

A \*\*Jenkinsfile\*\* is a text file that contains the definition of a Jenkins Pipeline. It is stored in the source code repository alongside your application code, which makes versioning and collaboration easier.

- \*\*Declarative Jenkinsfile\*\*: Provides a simplified syntax for defining pipeline stages and steps.

- \*\*Scripted Jenkinsfile\*\*: Offers more flexibility with Groovy scripting, enabling dynamic and complex behaviors.

\*\*Example of a Jenkinsfile (Declarative)\*\*:

```groovy

pipeline {

agent any

environment {

APP\_NAME = 'myapp'

}

stages {

stage('Build') {

steps {

script {

sh 'make build'

}

}

}

stage('Deploy') {

steps {

sh "deploy.sh ${env.APP\_NAME}"

}

}

}

}

```

---

### 5. \*\*Jenkins Plugins\*\*

One of the most powerful features of Jenkins is its \*\*plugin ecosystem\*\*. Jenkins plugins extend its functionality, integrating with source control systems, notification services, deployment tools, and more.

#### Common Types of Jenkins Plugins:

- \*\*Source Code Management (SCM) Plugins\*\*: Integrate Jenkins with tools like Git, Subversion, Mercurial.

- \*\*Build Tools Plugins\*\*: Integrate with build systems like Maven, Gradle, or Ant.

- \*\*Notification Plugins\*\*: Send notifications to Slack, email, or other services.

- \*\*Containerization Plugins\*\*: Integrate with Docker or Kubernetes.

- \*\*Security Plugins\*\*: Integrate with LDAP, Active Directory, or OAuth for authentication and authorization.

For example, the \*\*Git Plugin\*\* integrates Git with Jenkins, enabling Jenkins to pull code from a Git repository to perform the build.

---

### 6. \*\*Jenkins Shared Libraries\*\*

A \*\*Shared Library\*\* is a way to define reusable pieces of code, such as functions, steps, or utilities, that can be used across multiple Jenkinsfiles. This helps to avoid repetition and makes it easier to maintain complex pipelines.

You can define shared libraries in a separate repository or inside the Jenkins workspace, and include them in your Jenkinsfile with:

```groovy

@Library('my-shared-library') \_

```

Shared libraries provide functions and scripts that can be reused across multiple Jenkinsfiles and pipelines, making your CI/CD process more modular and maintainable.

---

### 7. \*\*Jenkins Credentials Management\*\*

Jenkins provides a secure way to store and manage sensitive data such as passwords, API tokens, and SSH keys. This is done through the \*\*Credentials Plugin\*\*.

Credentials can be:

- \*\*Global credentials\*\*: Available for all Jenkins jobs and pipelines.

- \*\*Job-specific credentials\*\*: Available only within a specific job or pipeline.

Jenkins can securely store:

- \*\*Username and password\*\* pairs.

- \*\*SSH private keys\*\* for Git or other services.

- \*\*Secret text\*\* for API tokens or other sensitive information.

Credentials are injected into the pipeline using environment variables or pipeline steps.

```groovy

withCredentials([usernamePassword(credentialsId: 'my-credentials', usernameVariable: 'USER', passwordVariable: 'PASS')]) {

sh 'git clone https://$USER:$PASS@github.com/repository.git'

}

```

---

### 8. \*\*Jenkins Build Triggers\*\*

A \*\*build trigger\*\* is a condition or event that causes a Jenkins job or pipeline to run. Jenkins supports a variety of build triggers, such as:

- \*\*SCM Polling\*\*: Jenkins periodically checks the version control system (e.g., Git) for changes.

- \*\*Webhook Triggers\*\*: External systems can send HTTP POST requests to Jenkins (e.g., GitHub or GitLab).

- \*\*Scheduled Builds\*\*: Jenkins can trigger builds based on cron-like schedules.

- \*\*Manual Triggers\*\*: A user can manually trigger a job through the Jenkins UI.

```groovy

triggers {

cron('H 0 \* \* \*') // Trigger daily at midnight

}

```

---

### 9. \*\*Jenkins Master-Slave Architecture\*\*

Jenkins follows a \*\*master-slave\*\* architecture for distributing workloads:

- The \*\*master\*\* node handles scheduling of jobs, managing the UI, and maintaining job configurations.

- The \*\*slave\*\* (or agent) nodes are responsible for running jobs in parallel to the master. They can be physical or virtual machines, or even Docker containers.

Jenkins master-slave architecture helps distribute the load, scale jobs, and provide isolation for specific tasks. For example, you may have a node dedicated to running tests or another node specifically for deployment tasks.

---

### 10. \*\*Jenkins Artifact Management\*\*

Jenkins supports artifact management and can store build artifacts (such as `.jar`, `.war`, or Docker images) using the \*\*Artifact Manager\*\*. Artifacts are typically stored in external systems, like \*\*Artifactory\*\*, \*\*Nexus\*\*, or \*\*S3\*\*, and they can be promoted through different stages of the pipeline (e.g., from `build` to `staging` to `production`).

You can archive and deploy artifacts as part of your build process:

```groovy

post {

success {

archiveArtifacts '\*\*/target/\*.jar'

}

}

```

---

### 11. \*\*Jenkins Monitoring and Logging\*\*

To ensure the health and performance of Jenkins, monitoring and logging are essential:

- \*\*Jenkins Metrics Plugin\*\*: Exposes various metrics related to job execution, system performance, and resource usage.

- \*\*Build Logs\*\*: Jenkins provides detailed logs for each build, which help with debugging and tracing build problems.

- \*\*External Monitoring\*\*: Tools like \*\*Prometheus\*\* and \*\*Grafana\*\* can be integrated with Jenkins for advanced monitoring and alerting.

---

### 12. \*\*Jenkins Security\*\*

Security is a crucial aspect of Jenkins:

- \*\*Authentication\*\*: Jenkins supports multiple authentication mechanisms like \*\*LDAP\*\*, \*\*OAuth\*\*, and \*\*Active Directory\*\*.

- \*\*Authorization\*\*: Control access to Jenkins features with \*\*Role-Based Access Control (RBAC)\*\* or \*\*Matrix-based security\*\*.

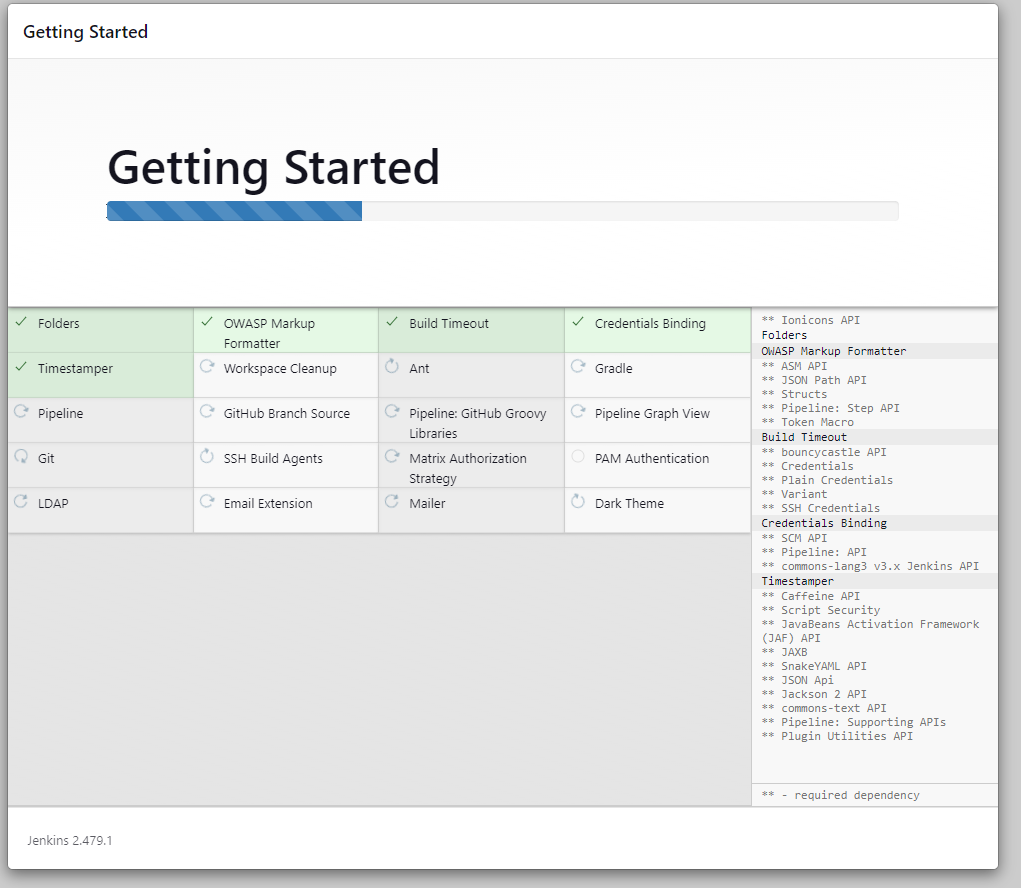
- \*\*Audit Logs\*\*: Track user actions and access to the Jenkins system.

---

### Conclusion

Jenkins is a highly flexible and powerful automation server, with many advanced features that allow it to scale and integrate with complex DevOps pipelines. Concepts such as \*\*Pipelines\*\*, \*\*Nodes and Executors\*\*, \*\*Shared Libraries\*\*, \*\*Credentials Management\*\*, and \*\*Monitoring\*\* are essential for mastering Jenkins and using it efficiently in larger, more complex CI/CD environments.

Let me know if you'd like more details on any of these advanced Jenkins concepts!



For checking the initial password when jenkins started

→ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

eba3103172fd40169a382e7d021d2735

First start the jenkins

→ sudo systemctl start jenkins

2.create the git repository and create a index.html file in it

3. Then make a dir in local machine and clone the repo in this dir

4. Give the sudo permission that directory.

→sudo visudo

→jenkins ALL=(ALL) NOPASSWD:ALL

15/11/2024

Node is the machine to provide resource.

What is the agent in jenkins

→agent is the builder one can handle the job.

Multicast address ?

# **Jenkins, From Zero To Hero: Become a DevOps Jenkins Master**

Become a DevOps Master learning Jenkins & integrations with powerful tools like Docker, Ansible, AWS, GIT & more!