SOFTWARE ENGINEERING

UNIT – 3

TOPIC – 10

CONTINUOUS INTEGRATION USING JENKINS, INSTALLATION OF JENKINS

Introduction to Jenkins

Jenkins is a free tool that helps automate tasks in software development. It's mainly used to build, test, and deploy code whenever there are changes. This process is called **Continuous Integration (CI)** and **Continuous Delivery (CD)**. Jenkins helps developers work faster by handling repetitive tasks and finding mistakes early. It's like an assistant that takes care of the routine steps, allowing developers to focus on more important tasks.

Jenkins Plugins

In **Jenkins**, plugins are like "add-ons" that give it extra abilities, like **apps on a smartphone**. They help Jenkins do more things and work with other tools that developers use.

Here's how plugins help:

1. **Add New Skills**: Plugins let Jenkins do new things. For example, if you need Jenkins to work with a cloud service, there's probably a plugin for that.
2. **Connect to Other Tools**: Plugins help Jenkins connect to things like **Git** (where code is stored) or **Slack** (for team messages). So, Jenkins can check code from Git or send a message in Slack when a task is done.
3. **Create Custom Steps**: If you want Jenkins to do something specific in a project, like run extra tests, plugins make it easy to add these steps.
4. **Easy to Install**: Jenkins has a **Plugin Manager** where you can find and install plugins with just a few clicks.
5. **Popular Plugins**:
    - **Git Plugin**: Helps Jenkins get the latest code.

- **Slack Plugin**: Sends a message to your team when a task is done.
- **Pipeline Plugin**: Lets you create step-by-step workflows in Jenkins.

Main Features of Jenkins

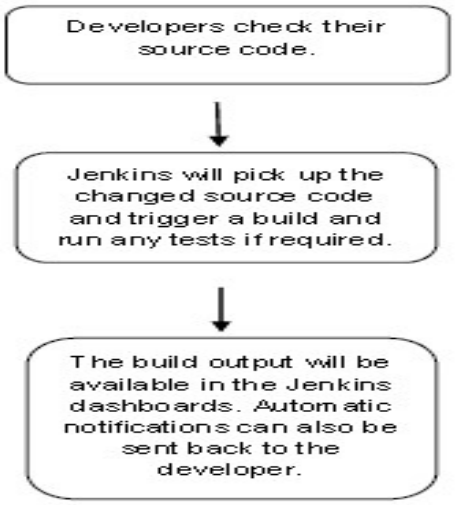
Jenkins is popular because of the following features:

- **Automation**: Jenkins automatically checks code for errors, builds the project, and runs tests, which saves developers time.
- **Integration**: It works well with tools like Git (to manage code) and Maven (to build projects).
- **Plugins**: Jenkins has many plugins that help it connect to other tools, like cloud services or testing tools.
- **Distributed Builds**: Jenkins can split tasks across multiple computers to get things done faster.
- **Customizable**: Developers can add their own plugins to make Jenkins fit their specific needs.
- **Monitoring and Reporting**: Jenkins gives updates on what's happening, so you know if something went wrong.
- **Pipeline Support**: A "pipeline" is a series of steps to build, test, and release software. Jenkins automates these steps, speeding up the whole process.

How Jenkins Workflow Works

The **workflow** of Jenkins generally follows these steps:

1. **Code Repository**: Developers store their code in a shared place, like **Git**.
2. **Jenkins Trigger**: Jenkins watches the code repository for changes. When someone updates the code, Jenkins sees it and starts working.
3. **Code Checkout**: Jenkins pulls the updated code from the repository.
4. **Build**: Jenkins compiles the code and turns it into a usable format.
5. **Testing**: Jenkins automatically runs tests to make sure the code works as expected.
6. **Artifact Archiving**: Jenkins saves the final build (called an artifact) for later use.
7. **Notification**: Jenkins sends notifications to let developers know if the process worked or if there was a problem.



Example: Imagine you're building a website. Whenever a team member adds new features, Jenkins automatically checks if everything works, sends an email with the results, and saves the final product.

Continuous Integration (CI)

Continuous Integration (CI) means that developers regularly add their code changes to a shared project. Each time they make changes, Jenkins tests and builds the code automatically. This helps catch issues early and ensures that everyone's code works well together.

In simple terms, CI allows teams to check their work frequently so they don't have to wait until the end of the project to find and fix problems.

Continuous Delivery with Jenkins

Continuous Delivery (CD) is a practice where code is not only integrated frequently (like in Continuous Integration) but also delivered to a staging or production environment automatically. This ensures that new features, bug fixes, and other improvements are always ready to be deployed. Jenkins plays a key role here by automating each step, from code building to testing and finally deploying. CD with Jenkins helps developers release software quickly and reliably by automating repetitive tasks and reducing the risks associated with manual deployments.

Setting Up Continuous Integration in Jenkins

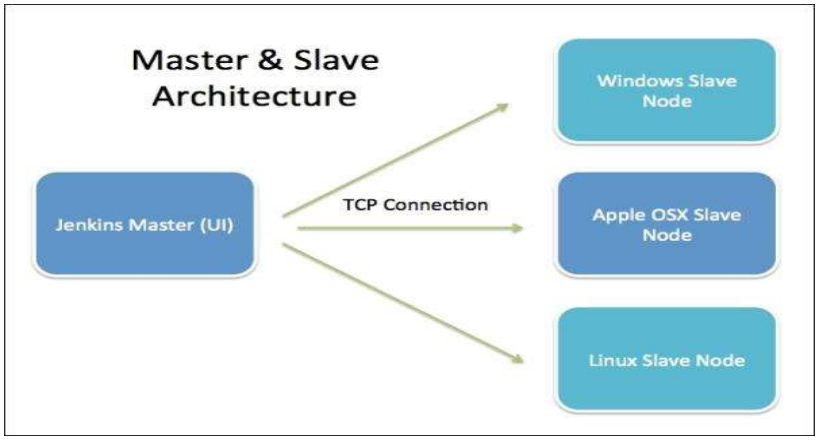
Follow these steps to set up **Continuous Integration (CI)** in Jenkins, so that your code is automatically built and tested every time you make changes:

1. **Install Jenkins**: First, install Jenkins on your computer or server. This will handle all the automation.
2. **Create a New Jenkins Job**:
   - Open Jenkins in your web browser.
   - Click "New Item" and name your job (for example, "MyCIJob").
   - Select the **"Freestyle project"** option, which is a basic setup for CI tasks.
3. **Configure Source Code Management**:
   - In the job settings, go to the **"Source Code Management"** section.
   - Select **Git** or another version control system.
   - Enter the **repository URL** (the link to your code) and add credentials if needed.
4. **Set Up Build Steps**:
   - In **"Build"**, set up what Jenkins should do to build your project.
   - For example, if you're using **Maven**, choose "Invoke top-level Maven targets" and enter goals like `clean install`.
   - For **Gradle**, select "Invoke Gradle script" and specify tasks like `clean build`.
5. **Post-Build Actions**:
   - After building, you can set up additional steps in **"Post-build Actions"**.
   - You can tell Jenkins to **save artifacts** (built files) or **send notifications** about the build status.
6. **Save and Run the Job**:
   - Save your setup and run the job to make sure it works. Jenkins will pull the code, build it, and report the result.
7. **Set Up Webhooks for Automatic Builds (Optional)**:
   - If you want Jenkins to automatically build every time the code changes, set up **webhooks** in your Git repository.
   - Webhooks tell Jenkins, "A new change was made—start building!" Alternatively, Jenkins can check for changes at regular intervals.

By following these steps, Jenkins will automatically build and test your code whenever you make changes.

Jenkins Architecture

Jenkins uses a **distributed architecture** that helps it handle many tasks at once. It consists of two main parts: the **Jenkins Master** and **Jenkins Slaves**.



Jenkins Master

The **Jenkins Master** is the main part of Jenkins. It does the following jobs:

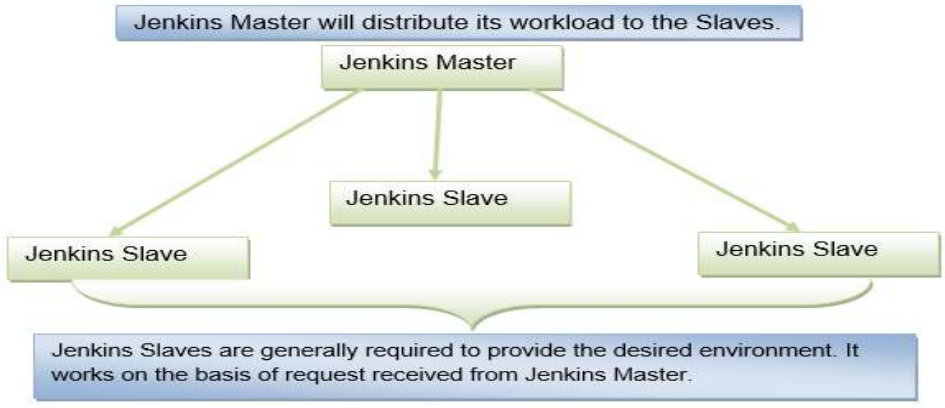
- **Scheduling jobs**: The master decides when tasks (builds) should run.
- **Distributing tasks**: The master sends tasks to slave computers to do the actual work.
- **Monitoring slaves**: It keeps track of the slave machines and manages their availability.
- **Displaying results**: The master shows the results of the builds, so developers know if they were successful.

The Jenkins Master can also run jobs itself, but it usually sends them to slaves to spread the workload.

Jenkins Slaves

Jenkins Slaves are extra computers that do the actual work of building and testing. This allows Jenkins to handle more tasks at once. Each slave runs a **Jenkins agent**, which communicates with the master. The master sends tasks to the slave, and the slave does the job (like building or testing) and sends back the results.

Slaves can be set up on different machines (physical or virtual), and you can choose whether a project should always run on a specific slave or let Jenkins pick an available one.

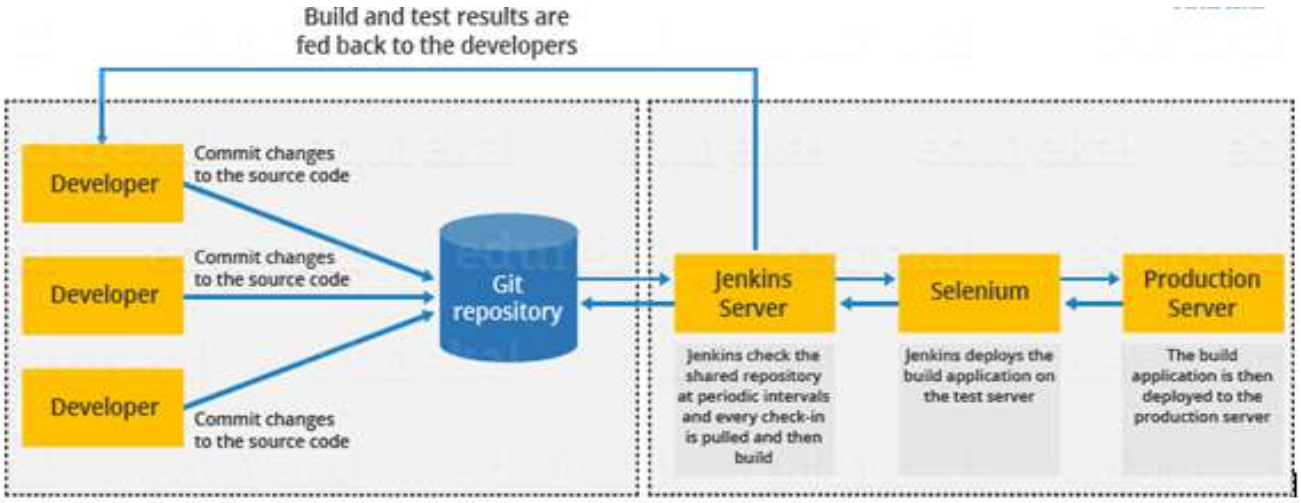


Example: Imagine Jenkins is like a factory. The **master** is the manager that organizes and assigns tasks, and the **slaves** are the workers that actually complete the tasks. The manager keeps track of all the workers and checks if the tasks are done correctly.

Jenkins Flow Diagram for Continuous Integration

The Continuous Integration (CI) process with Jenkins generally works like this:

1. A developer updates the code in a **source code repository** (like Git).
2. Jenkins detects the code change and starts a new build.
3. If there's a problem, Jenkins alerts the team. If everything works, Jenkins moves on to the next steps.
4. If the build is successful, Jenkins deploys it to a test server to verify everything works.
5. Jenkins then informs developers of the results and repeats this process for each new code update.



Example: Imagine you're writing code for an online store. Each time you make a change, Jenkins checks if it's compatible with existing features and tells you if anything needs fixing.

Advantages of Using Jenkins

- **Free**: Jenkins is open-source, so it's free to use.
- **Easy to Set Up**: It's simple to install and doesn't require complicated setup.
- **Customizable**: With many plugins available, Jenkins can fit different needs.
- **Works Across Platforms**: It runs on Windows, macOS, and Linux.
- **Cloud Support**: Jenkins can also run on cloud servers, making it flexible for remote use.

Drawbacks of Jenkins

- **Outdated Interface**: Jenkins looks a bit old-fashioned and can be tricky to navigate.
- **Requires Some Technical Knowledge**: Since Jenkins runs on a server, it might require some basic server skills to manage.
- **Can Be Fragile**: Small configuration changes can sometimes cause issues, so it needs regular attention.

Steps to Install Jenkins on Windows, macOS, and Linux

Jenkins can be installed on Windows, macOS, and Linux systems. Below are simple step-by-step instructions for each operating system.

Installing Jenkins on Windows

1. **Download Jenkins:**
   - Go to Jenkins' official download page and download the Windows installer (Jenkins Windows Installer .msi file).
2. **Install Java (if not already installed):**
   - Jenkins requires Java. Download Java from the Oracle website.
   - Install it by following the prompts and set the `JAVA_HOME` environment variable in Windows:
     - Right-click "This PC" > "Properties" > "Advanced System Settings."
     - Click "Environment Variables" > "New" and add `JAVA_HOME` with the path where Java is installed (e.g., `C:\Program Files\Java\jdk-xx`).
3. **Run the Jenkins Installer:**
   - Open the downloaded `.msi` file and follow the installation steps.
   - During installation, you can select the folder where Jenkins will be installed.
   - Choose the option to install Jenkins as a Windows service if prompted (this allows Jenkins to start automatically).
4. **Complete Setup and Start Jenkins:**
   - After installation, open a web browser and go to `http://localhost:8080`.
   - Enter the administrator password located in the `C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword` file.
   - Complete the setup by installing the recommended plugins and creating an admin user.

Installing Jenkins on macOS

1. **Install Homebrew (if not already installed):**
   - Homebrew is a package manager for macOS. If it's not installed, open the Terminal and paste this command:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. **Install Java (if not already installed):**
   - Jenkins requires Java. Use Homebrew to install Java:

```
brew install openjdk@11
```

- Set up the `JAVA_HOME` environment variable by adding this to your shell configuration file (e.g., `.zshrc` or `.bash_profile`):

```
export JAVA_HOME=$(/usr/libexec/java_home -v 11)
source ~/.zshrc  # or source ~/.bash_profile
```

3. **Install Jenkins using Homebrew:**
   - Use Homebrew to install Jenkins with this command:

```
brew install jenkins-lts
```

4. **Start Jenkins:**
   - Start Jenkins using:

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

   - Open a web browser and go to `http://localhost:8080`.
   - Locate the password file at `/Users/your-username/.jenkins/secrets/initialAdminPassword`.
   - Use this password to unlock Jenkins, install plugins, and create an admin user.

Installing Jenkins on Linux

1. **Update Packages and Install Java (if not already installed):**
   - Open a terminal and update your package manager:

```
sudo apt update
sudo apt install openjdk-11-jdk -y
```

   - Check Java is installed by running:

```
java -version
```

2. **Add Jenkins Repository:**
   - Import the Jenkins GPG key:

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

- Add the Jenkins package repository:

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

3. **Install Jenkins:**
   - Update the package index and install Jenkins:

```
sudo apt update
sudo apt install jenkins -y
```

4. **Start and Enable Jenkins:**
   - Start Jenkins and enable it to start on boot:

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

5. **Access Jenkins:**
   - Open a browser and go to `http://localhost:8080`.
   - Retrieve the initial admin password with:

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

   - Complete the setup by installing recommended plugins and creating an admin account.