# Introduction

## What is Jenkins?

Jenkins is a **self-contained, open source, Jav-based** [**CI/CD**](https://devopscube.com/continuous-integration-delivery-deployment/)**tool** which can be used to automate all sorts of tasks related to building, testing, and deploying software.

# Installation

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

## Prerequisites

* Hardware:
  + 256 MB of RAM, although more than 2 GB is recommended
  + 10 GB of drive space (for Jenkins and your Docker image)
* Software:
  + Java 17 or 21 if Jenkins 2.479.1 (October 2024). OpenJDK is allowed. For details, check [support policy](https://www.jenkins.io/doc/book/platform-information/support-policy-java/).
  + Docker

## Download and Run Jenkins

1. [Download Jenkins Generic Java package (.war)](https://www.jenkins.io/download)
2. Open up a terminal in the download directory
3. Run java -jar jenkins.war --httpPort=8080
4. Browse to http://localhost:8080
5. Follow the instructions to complete the installation

# Glossary

## Agent

An agent is typically a machine, or container, which connects to a Jenkins controller and executes tasks when directed by the controller.

## Artifact

An immutable file generated during a Build or Pipeline run which is archived onto the Jenkins Controller for later retrieval by users.

## Stage

Stage defines a distinct subset of tasks performed through the entire Pipeline, for example "Build", "Test", and "Deploy".

## Step

Step defines a single task in a stage. It tell Jenkins what to do inside of a Pipeline or job.

Think of a step like a single command which performs a single action. When a step succeeds, it moves onto the next step. When a step fails to execute correctly, the Pipeline will fail.

For example, to execute the shell command make, use the sh step: sh 'make'.

## Job

A job is a collection of stages. We normally call it a Jenkins Pipeline.

# Jenkins Architecture

## Key Components

### Jenkins Server (or Master Node)

Server holds all key configurations. It's like a **control server** that orchestrates all the workflow defined in the pipelines.

Jenkins server is where you can find Jobs, Users, Plugins, Credentials, Global Configs, Logs, Jenkins Data.

#### Plugins

Jenkins has a rich ecosystem of plugins to extend its functionality:

* Integrate with various version control systems (e.g., Git, SVN).
* Support different build tools (e.g., Maven, Gradle).
* Enable notifications (e.g., email, Slack).
* Add reporting capabilities.
* Add UI enhancements.

#### Configuration

Configuration in Jenkins server covers job, system, plugin, security, environment, user views, etc..

#### Jenkins Data

All the Jenkins data gets stored in /var/lib/jenkins/. Data includes all jobs config files, plugins configs, secrets, node info, etc. It makes Jenkins migration very easy.

Most configurations are in XML format.

### Jenkins Agent (or Slave Node)

An agent is a worker node (physic or virtual) that actually **execute all the steps** mentioned in a job. When you create a Jenkins job, you have to assign an agent to it. Every agent has a label as a unique identifier.

You can have any number of Jenkins agents attached to a master with a combination of Windows, Linux servers, and even containers as build agents.

There are two types of Jenkins agents:

* **Static agent**: Servers (Windows/Linux) running all the time and stay connected to the Jenkins server.
* **Dynamic agent**: Gets deployed as a VM/container on demand and gets deleted once the job is completed.

**Note:** You can run jobs in Jenkins server without a Jenkins agent. Master nodes will act as the agent. However, recommended approach is [master-agent setup](https://devopscube.com/setup-agents-on-jenkins-2/) for different job requirements. You won't end up corrupting the server for any system-wide configuration changes.

### Jenkins Web Interface

Jenkins 2.0 introduced a very intuitive web interface called "**Blue Ocean"**. It has a good visual representation of all the pipelines.

### Build Triggers

Jenkins supports various mechanisms to trigger builds.

* Poll SCM (Source Code Management) for changes.
* Trigger builds based on webhooks (e.g., from GitHub).
* Scheduled builds (cron jobs).
* Manual build triggers via the UI.

# Jenkins Pipeline

## What Is Jenkins Pipeline?

Jenkins Pipeline is a set of plugins that supports Continuous Integration and Continuous Deployment (CI/CD) through automation of build, test and deployment processes.

A Jenkins Pipeline **allows** **users to define their build process as CODE**, using a Domain-Specific Language (DSL) based on [Groovy](https://groovy-lang.org/syntax.html). It is typically written in a text file called Jenkinsfile.

## Type of Jenkins Pipeline

* **Declarative Pipeline**: Simple and easy to read. Focuses on stages and steps.
* **Scripted Pipeline**: More flexible and powerful. Allows for complete control over the pipeline execution using Groovy scripting.

## Pipeline Syntax

### Declarative Pipeline

The pipeline block defines all the work done throughout your entire Pipeline.

Basic structure:

**pipeline** {

    agent any

    stages {

        stage('Build') {

            steps {

                //

            }

        }

        stage('Test') {

            steps {

                //

            }

        }

        stage('Deploy') {

            steps {

                //

            }

        }

    }

}

Example:

**pipeline** {

    agent any

    options {

        skipStagesAfterUnstable()

    }

    stages {

        stage('Build') {

            steps {

                sh 'make'

            }

        }

        stage('Test'){

            steps {

                sh 'make check'

                junit 'reports/\*\*/\*.xml'

            }

        }

        stage('Deploy') {

            steps {

                sh 'make publish' //

            }

        }

    }

}

**Note**:

* In order to use Declarative Pipeline, install the [Pipeline: Declarative Plugin](https://plugins.jenkins.io/pipeline-model-definition).
* Full syntax, check [Pipeline Syntax](https://www.jenkins.io/doc/book/pipeline/syntax/)

### Scripted Pipeline

One or more node blocks do the core work throughout the entire Pipeline. Although this is not a mandatory requirement of Scripted Pipeline syntax, confining your Pipeline’s work inside of a node block does two things:

1. Schedules the steps contained within the block to run by adding an item to the Jenkins queue. As soon as an executor is free on a node, the steps will run.
2. Creates a workspace (a directory specific to that particular Pipeline) where work can be done on files checked out from source control.  
   **Caution:** Depending on your Jenkins configuration, some workspaces may not get automatically cleaned up after a period of inactivity. See tickets and discussion linked from [JENKINS-2111](https://issues.jenkins.io/browse/JENKINS-2111) for more information.

Basic structure:

**node** {

    stage('Build') {

        //

    }

    stage('Test') {

        //

    }

    stage('Deploy') {

        //

    }

}