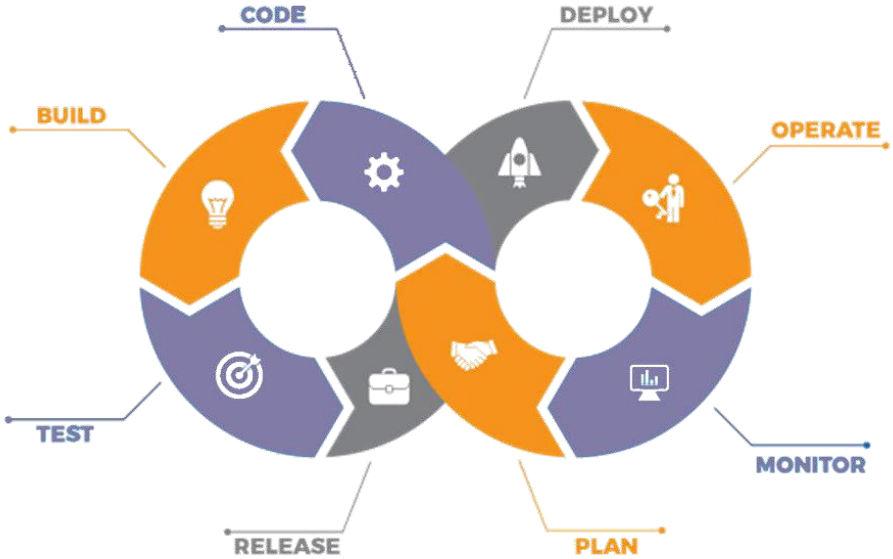
IntelliPaat

Continuous Integration Using Jenkins & Maven™



Agenda

01 Why Maven?

02 What is Maven?

03 What does Maven do?

04 What is a POM file?

05 Maven Life Cycles

06 Introduction to Jenkins

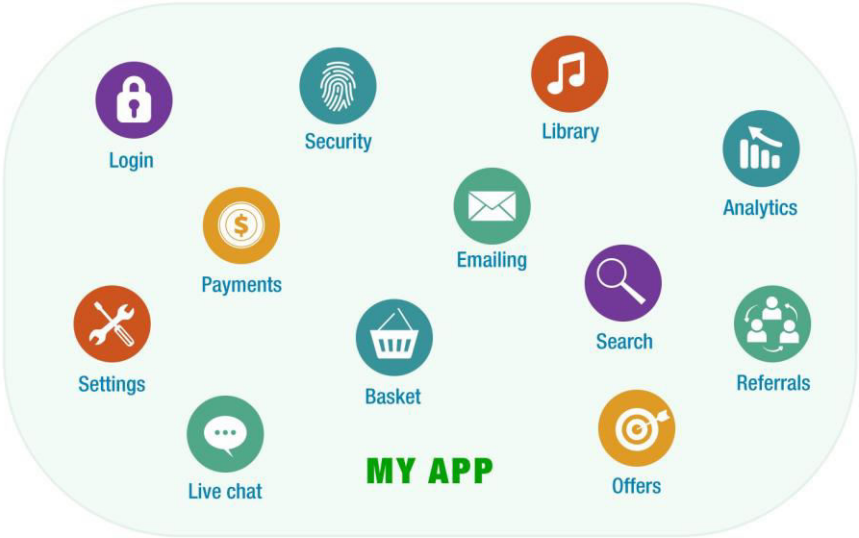
07 Maven Installation

08 Maven Hands-on

Why Maven?

Why Maven?

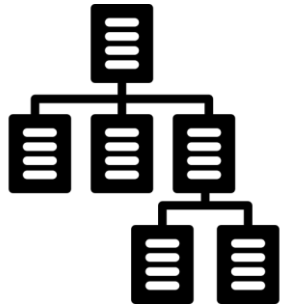
The written code by developers has to be packaged and converted into a executable format, this is done with the help of Maven



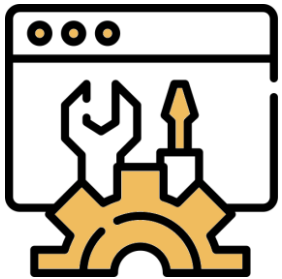
Using Maven

Since Maven is a build tool, it can help in building an application that has several different modules. It helps doing so by:

Downloading dependencies



Enforcing a directory structure

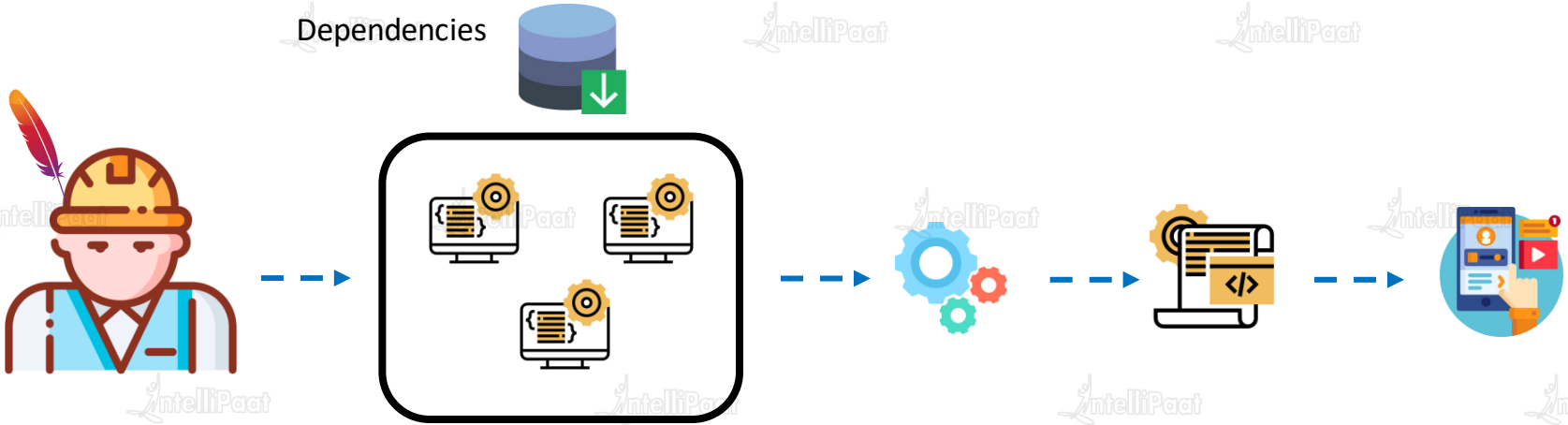


Building an executable with all the dependencies

What is Maven?

What is Maven?

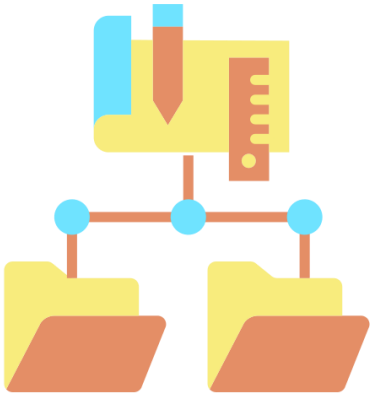
Maven is commonly referred to as a build tool that is used to manage the entire life cycle of a project, generate reports, and store documents with its POM repository



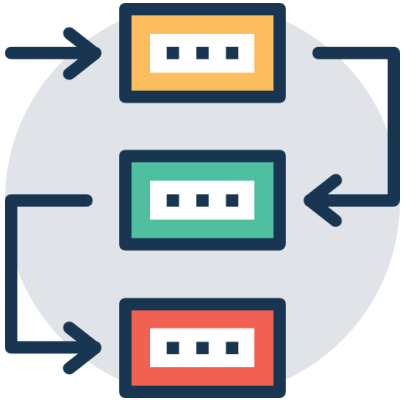
What does Maven do?

What does Maven do?

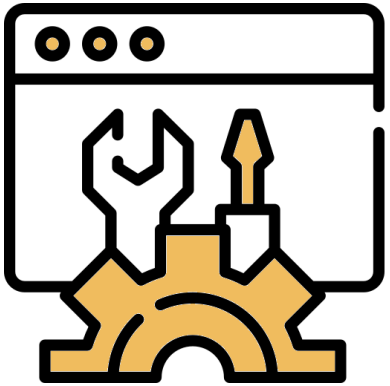
Enforce a Directory Structure

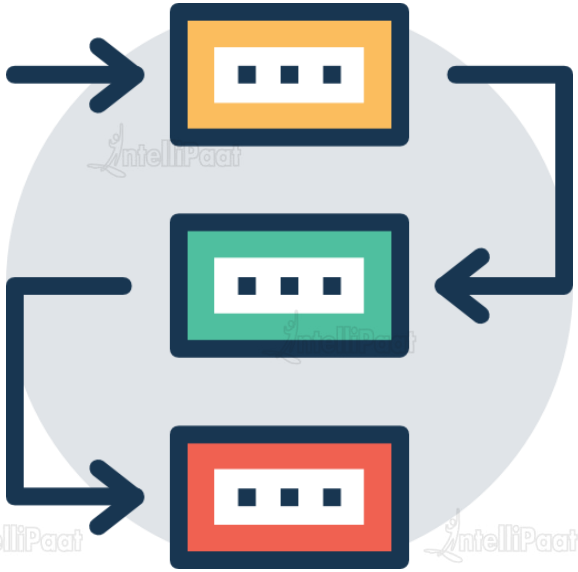


Manage and Download Project Dependencies



Build an executable for the code

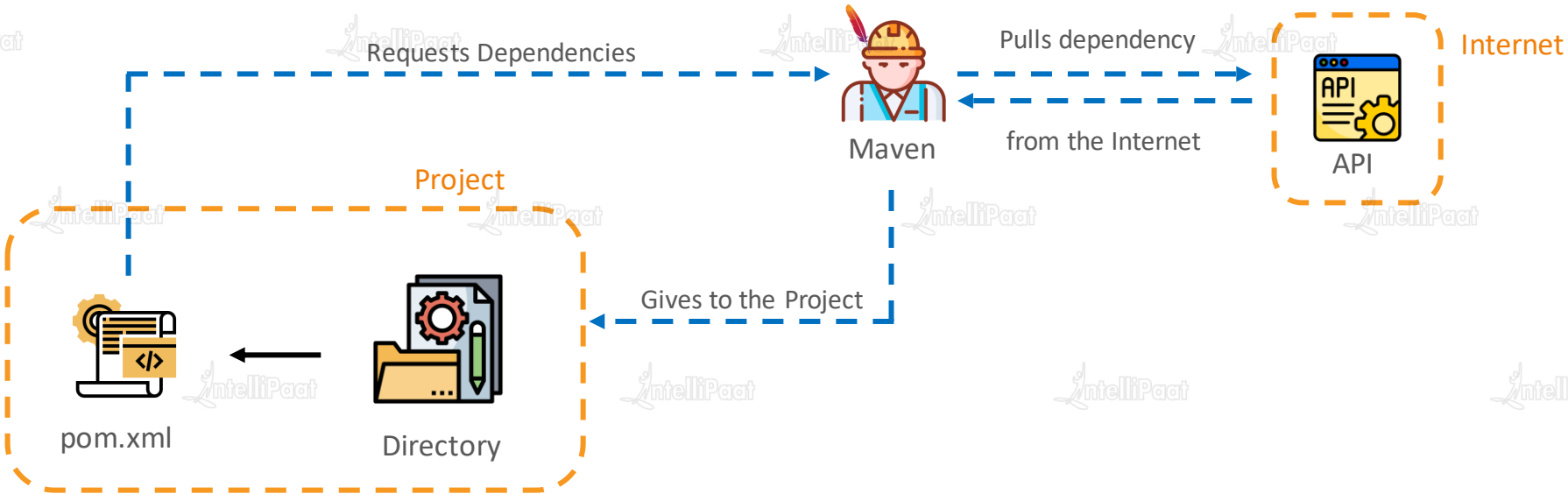




Managing and Downloading Project Dependencies

Project Dependencies

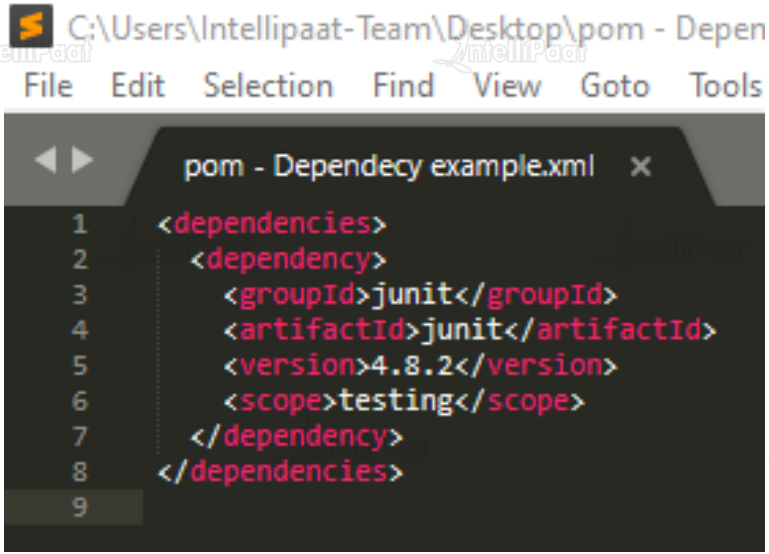
Projects may need Java APIs or frameworks that are packaged in their own JAR files. These JAR files are needed in the class path when a project code is compiled

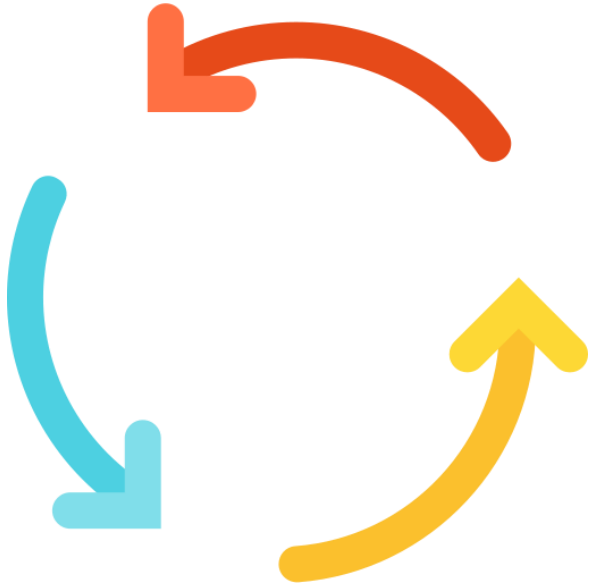


Building POM Files

POM (Project Object Model) File

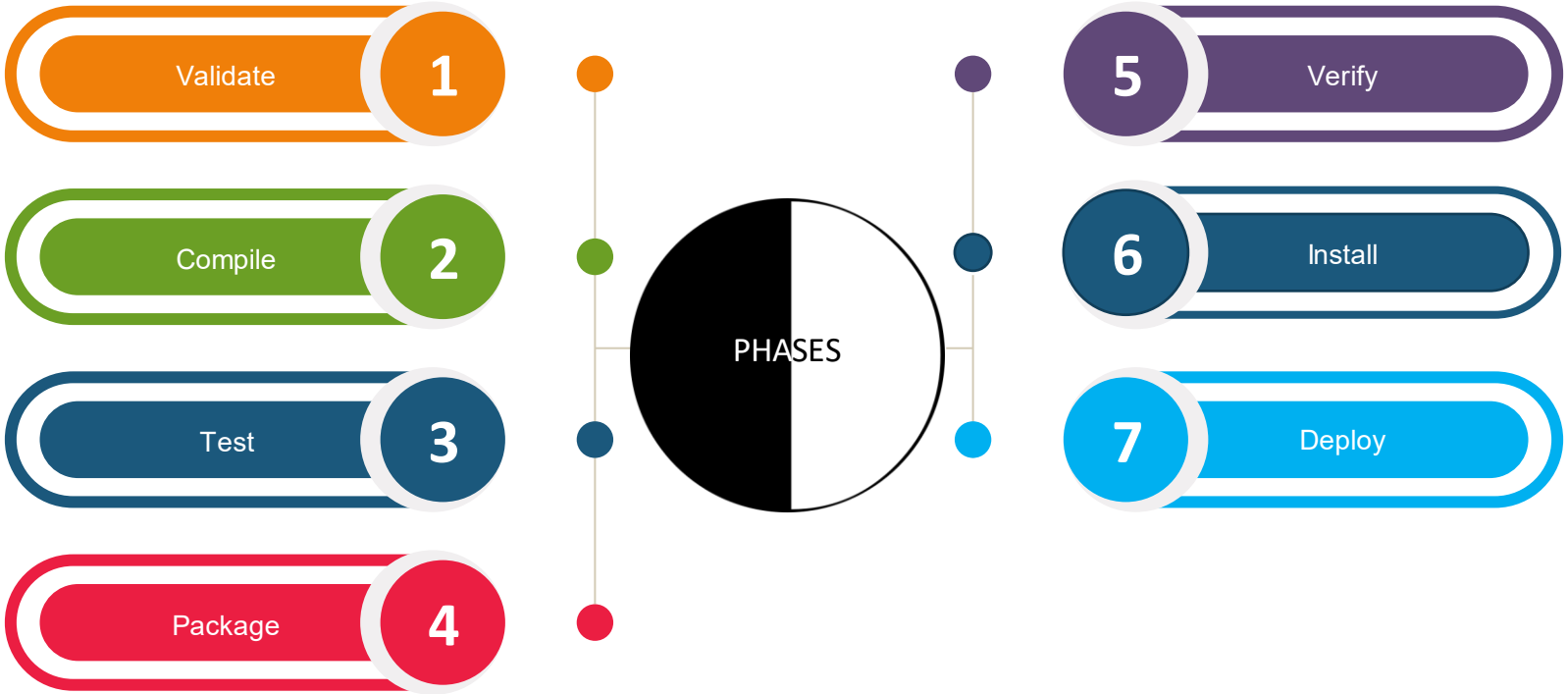
Every Maven project directory needs a **pom.xml** file. These files contain all details necessary for Maven to effectively execute a project. The POM file is stored in 'src' or in the root directory





Maven Build Life Cycle

Maven Build Life Cycle

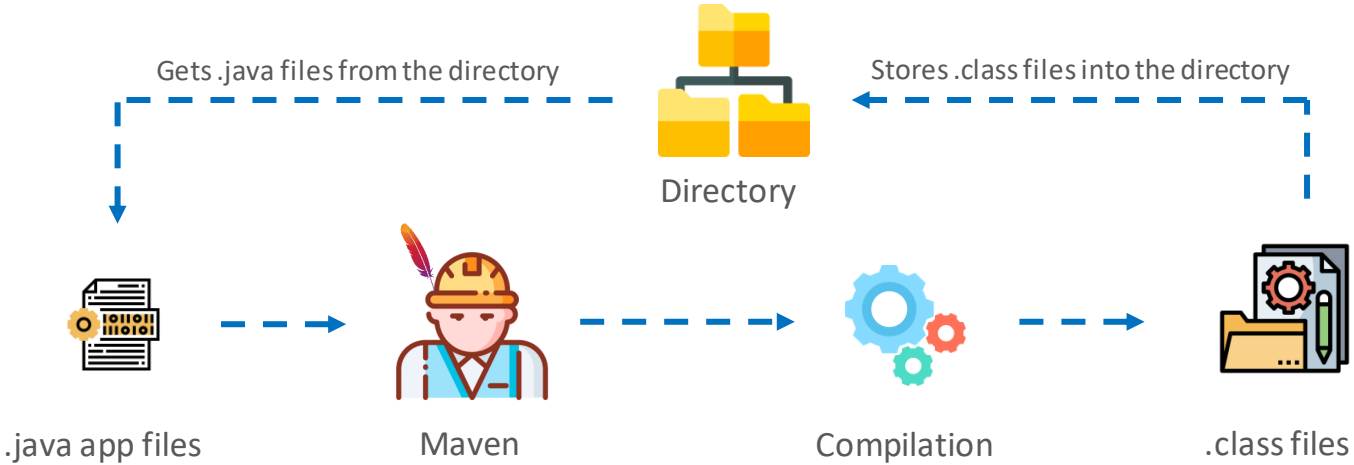


1. Validate Phase

validate the project is correct and all necessary information is available

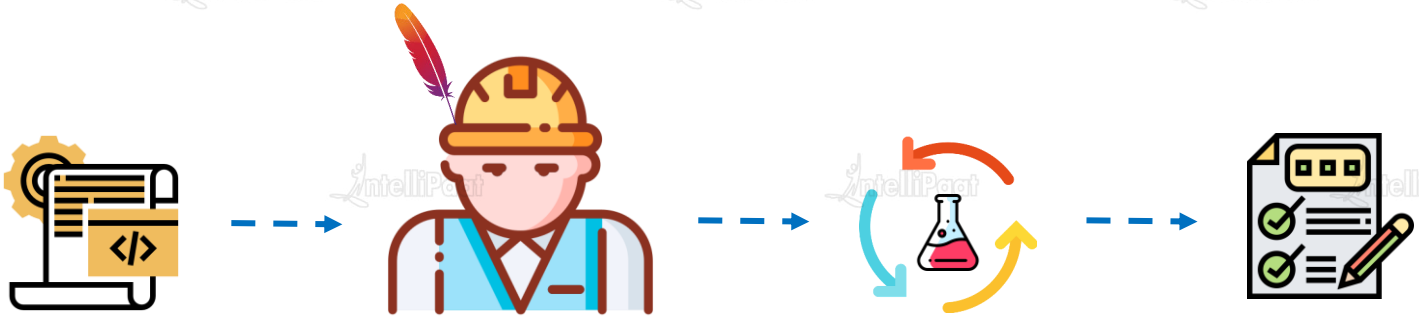
2. Compile Phase

During the compile phase, Maven will compile all **.java** files present in the main directory into **.class** files and put them back into the main directory in the dedicated folder



3. Test Phase

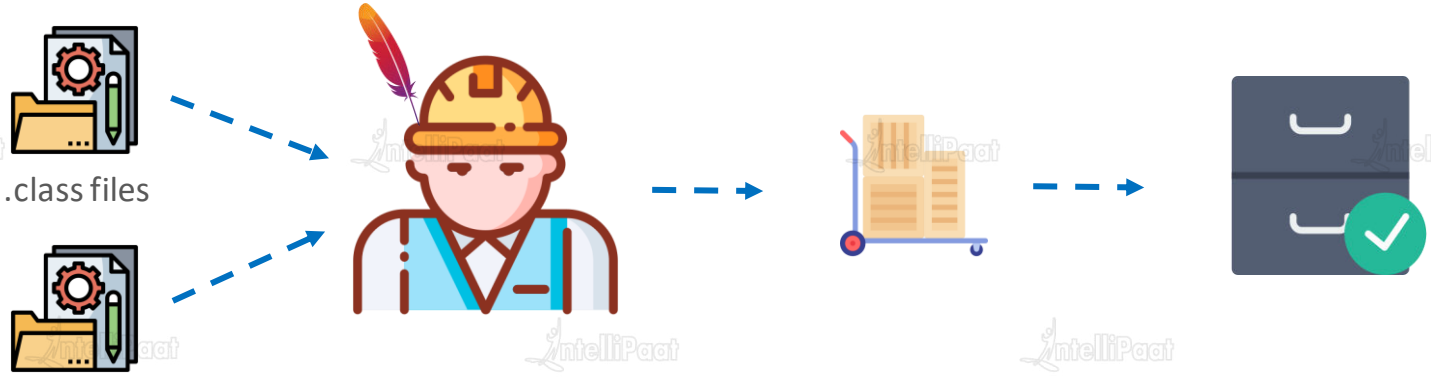
During the test phase, Maven will execute the specified test cases and create a summary log



4. Package Phase

During the package phase, Maven will package all **.class** files and resources into one file. This file will be formatted into one of the three types given below:

JAR Archive | WAR Archive | EAR Archive

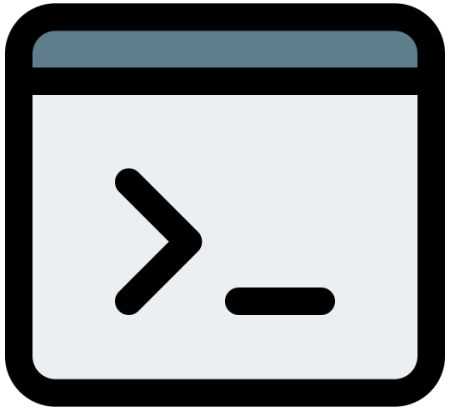


5. Verify Phase

It runs any checks on results of integration tests to ensure quality criteria are met, at the same time **all the previous lifecycle phases are also executed**

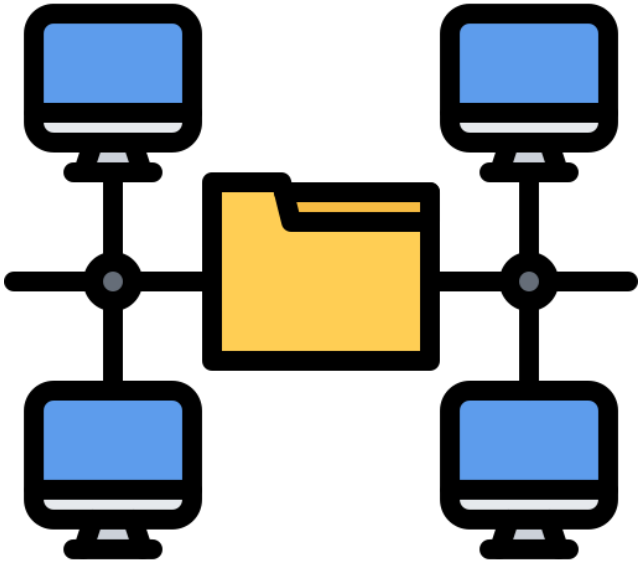
6. Install Phase

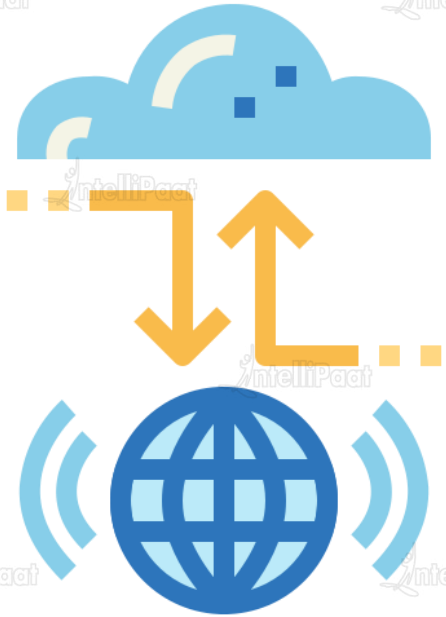
install the package into the local repository, for use as a dependency in other projects locally, all previous lifecycle phases are also executed



7. Deploy Phase

It copies the final package to the remote repository for sharing with other developers and projects.





Maven Repositories

Maven Repositories

After Maven understands which all dependencies are needed from the pom.xml file, it will download those dependencies from remote repositories and then store them in the local repository for current or future use

Installing Maven

Installing Maven

1. Download and Install Maven on Ubuntu 18.04 on AWS
2. Create a sample project using the maven generate command
3. Verify the directory structure by installing tree

Building a Project using Maven

Installing Maven

1. Create a sample Dynamic Website Project in Jenkins
2. Push the code to github
3. On the AWS Instance clone the code and Build the code
4. Deploy this code on tomcat to verify, if everything looks good

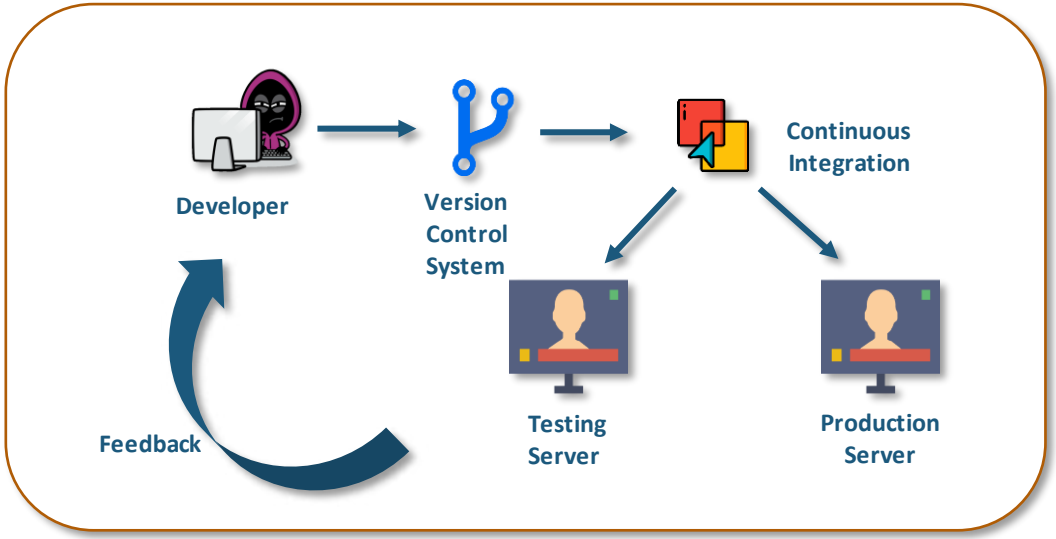
What is Continuous Integration?

Problems before Continuous Integration

- Infrequent Commits led to bigger releases in one go leading to complex integration.
- Manual testing took a lot of time.
- Feedback took a lot of time to reach the developer.
- It involved high risk and uncertainty.

What is Continuous Integration?

The process of having shorter release cycles (sometimes, several times a day), i.e., creating small features and integrating them to the source code and employing automated build and test processes for quicker feedback is called Continuous Integration.



Advantages of Continuous Integration

- Frequent Commits, hence small feature release
- Automated Build and Testing
- Instant feedback to the developer
- Low risk and faster delivery

What is Jenkins?

What is Jenkins?

Jenkins is an open-source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery.

Features of Jenkins

Adoption: Jenkins is extremely popular among the open-source community; hence, there are more than 147,000 active installations throughout the world and 1 million people are using it.

Plugins Support: With an extremely active open-source community, Jenkins has around 1000 plugins that allow it to integrate with most of the development, testing and deployment tools.

Advantages of Jenkins

Before Jenkins

- Locating and fixing bugs in the event of build and test failure was difficult and time consuming.
- Tests were triggered manually.
- No central place for triggering jobs on remote systems.

After Jenkins

- Smaller and automated continuous build and testing make the task accurate and faster.
- Developers have to just commit the code to the remote repository, build, test and deployment happen automatically.
- All builds or tests on multiple remote systems can be controlled from one place.

Installing Jenkins on AWS

Installing Jenkins on AWS

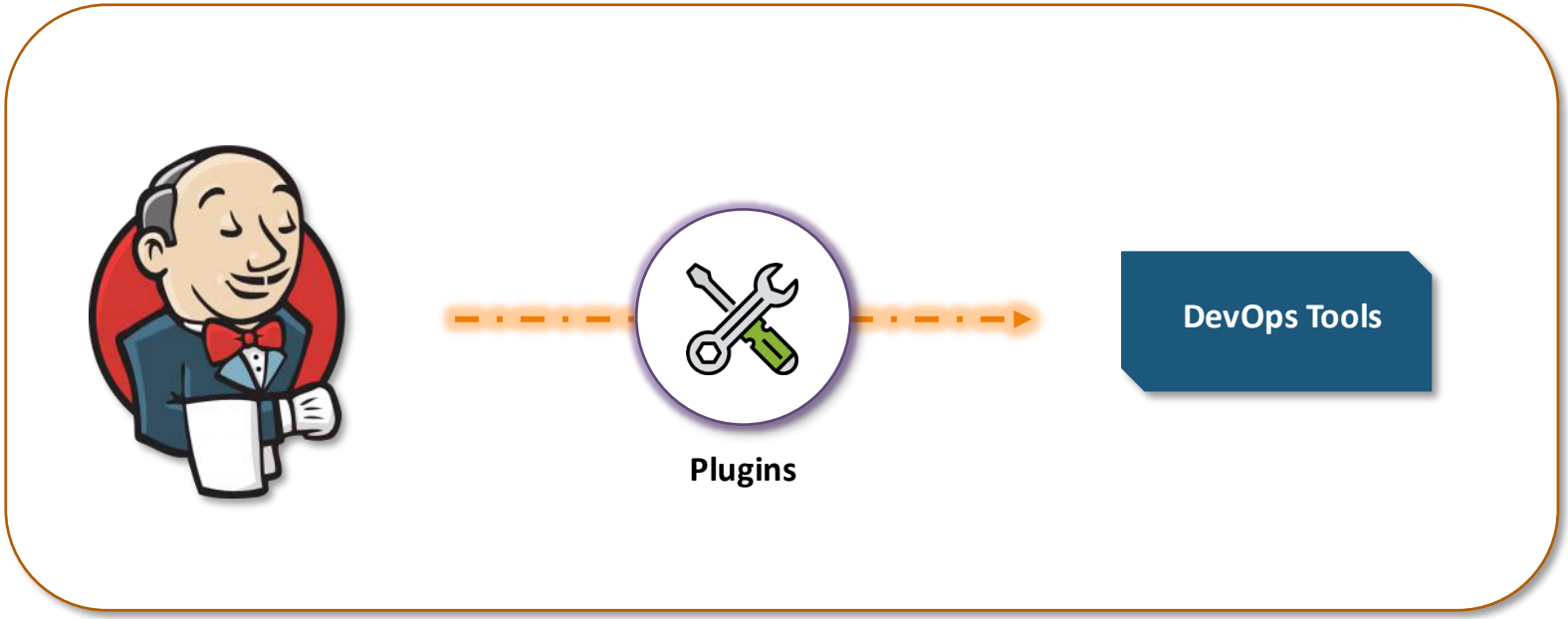
1. Launch an AWS Instance
2. Connect through SSH
3. Execute the following commands:

Jenkins Installation:

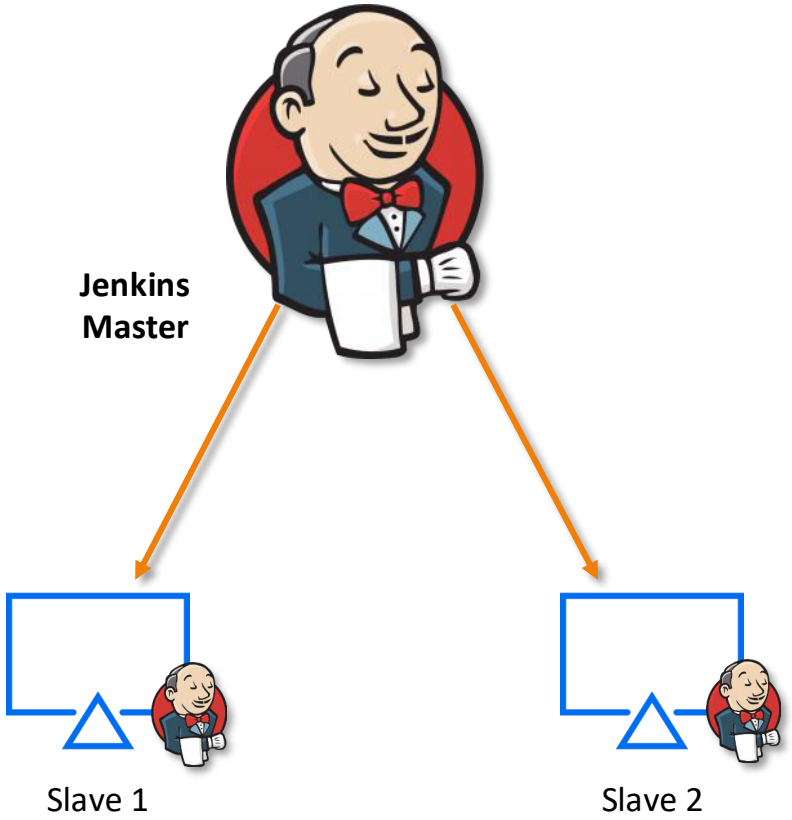
```
$> sudo apt-get update
$> sudo apt install openjdk-8-jdk
$> wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key  | sudo apt-key add -
$> sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable  binary/ >
/etc/apt/sources.list.d/jenkins.list'
$> sudo apt update
$> sudo apt install jenkins
```

Jenkins Architecture

Jenkins Architecture



Jenkins Architecture



Managing Nodes on Jenkins

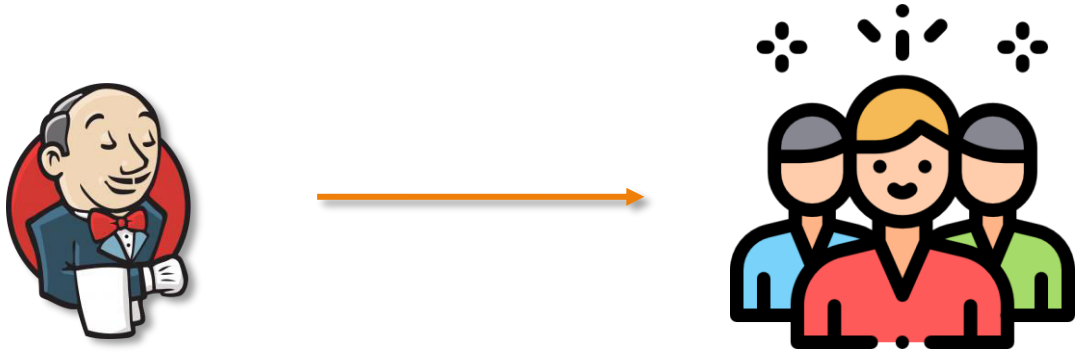
Managing Nodes on Jenkins

Add a slave node to Jenkins using JNLP connection

Managing users on Jenkins

Managing Users on Jenkins

We can provide access to multiple users in Jenkins and can provide them with roles based on project-based access.



Let’s start by adding some users in Jenkins

Hands-on: Adding users in Jenkins

Managing Users on Jenkins

Now obviously, you would not want each user to be an administrator in Jenkins, hence you can also control what permissions each user gets

Let's see, how we can manage permissions to a user using Matrix Based Authorization

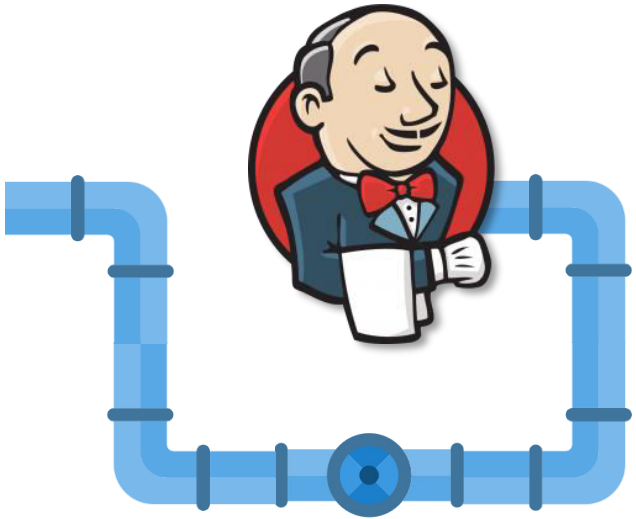
Hands-on: Giving restricted access to users

Understanding CI/CD Pipelines in Jenkins

What are CI/CD Pipelines?

CI/CD Pipelines, i.e., Continuous Integration, Continuous Delivery and Deployment pipelines, are a way of running Jenkins jobs in a sequence, which resembles a pipeline view.

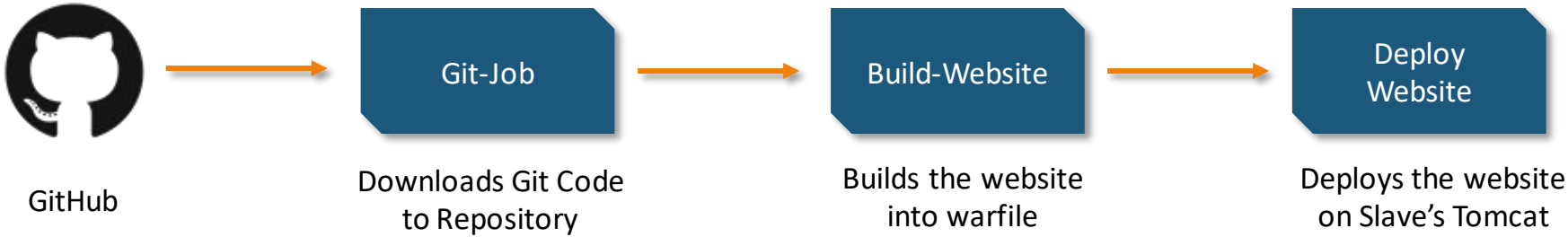
Finished Application



What are CI/CD Pipelines?

CI/CD Pipelines, i.e., Continuous Integration, Continuous Delivery and Deployment pipelines, are a way of running Jenkins jobs in a sequence, which resembles a pipeline view.

For Example:



Creating an automated CI/CD Pipeline in Jenkins

Creating an Automated CI/CD Pipeline

1. Initiate a Git Webhook for the Jenkin's git-job repository
2. Trigger the jobs after the completion of previous jobs with the following map: Git-Job → Build-Website → Deploy-Website
3. Install the plugin for the pipeline view
4. Make changes to the website and commit the job to see the changes

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor