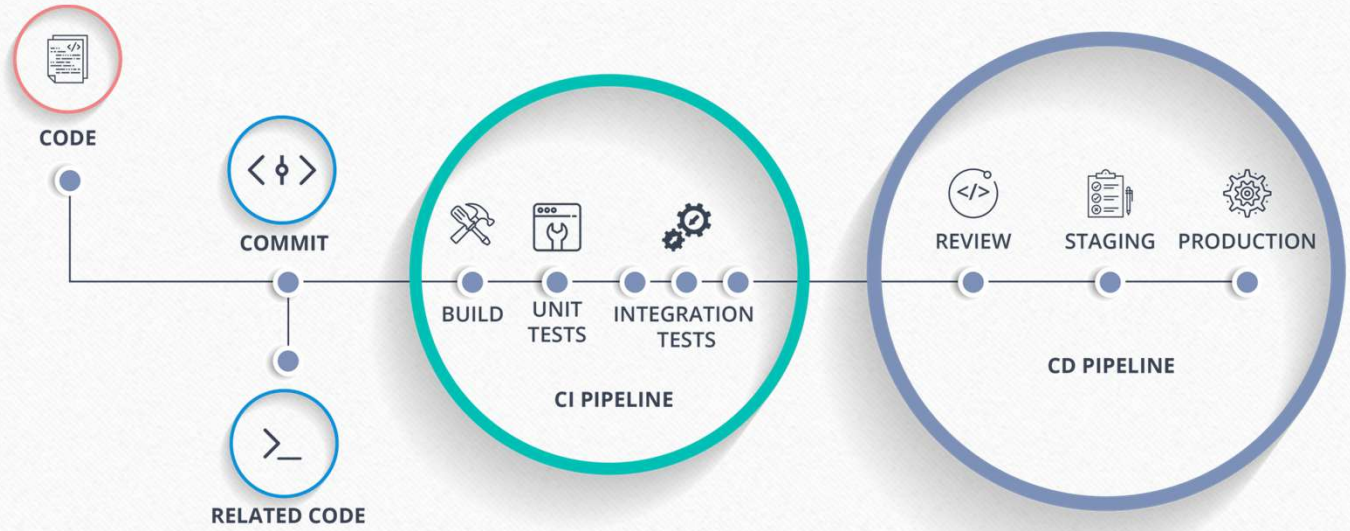
CI/CD Pipeline Using Jenkins

What is CI/CD pipeline

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. There is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

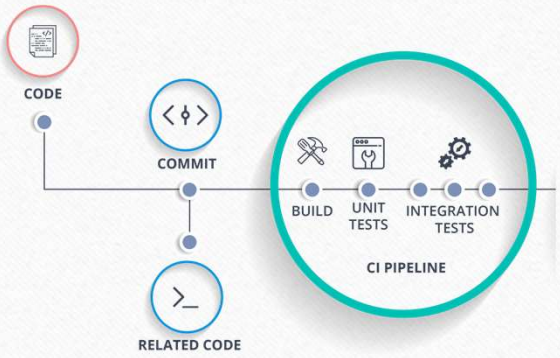
CI CD Structure

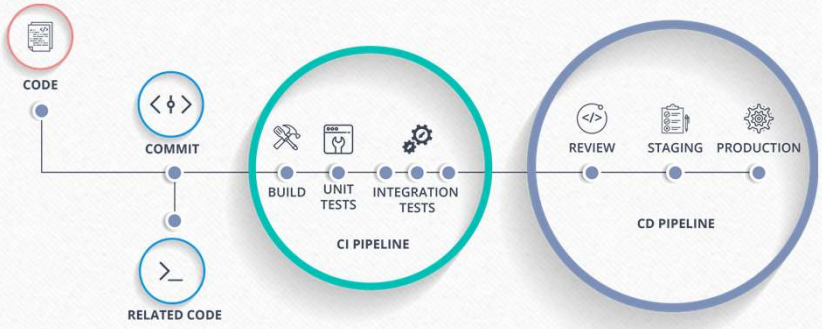


What is Continuous Integration (CI)/ Continuous Delivery (CD) ?

Continuous Integration is a practice that integrates code into a shared repository. It uses automated verifications for the early detection of problems. Continuous Integration doesn't eliminate bugs but helps in finding and removing them quickly.

Continuous Delivery is the phase in which the changes are made in the code before deploying. The team in this phase decides what is to be deployed to the customers and when. The final goal of the pipeline is to make deployments.





What is Jenkins?

Jenkins is an open-source automation tool used to build and test software projects. The tool makes it more convenient for developers to integrate changes to the project. Jenkins achieves Continuous Integration with the help of plugins. Jenkins turns out to be the best fit for building a CI/CD pipeline because of its flexibility, openness, plugin-capabilities, and simple to use nature.

Jenkins pipeline – declarative vs scripted pipelines

Jenkins allows users to use one of the two pipelines – declarative or scripted.

Jenkins pipeline –Scripted pipelines

Originally, Jenkins only allowed the user to write the pipeline as a code, hence a scripted pipeline. Such a pipeline is written in a JenkinsFile on the web UI of the Jenkins tool. You can write a scripted pipeline with DSL (domain specific language) and use Groovy based syntax. And what is Groovy? It is a JVM-based programming language. By using it in a scripted Jenkins pipeline, you gain access to the wide group of APIs packaged with the Java Development Kit (JDK).

Jenkins pipeline –Declarative pipeline script

Declarative pipelines are a relevantly new feature in Jenkins. Thanks to this option, the code of the pipeline is easier to write and read later. But it is quid pro quo – the code is simplified because declarative syntax is more limited. You won't be able to inject code into the pipelines whenever you want. Such a pipeline model is not really the best choice if you plan to build a pipeline with complex logi

Pipeline syntax differences

Declarative pipelines always begin with the word pipeline. Scripted pipelines, on the other hand, always begin with the word node. Declarative pipelines break down stages into individual stages that can contain multiple steps. Scripted pipelines use Groovy code and references to the Jenkins pipeline DSL within the stage elements without the need for steps.

These are the key differences that allow a developer to quickly differentiate between a scripted pipeline and a declarative pipeline.

Declarative

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {

            }
        }
        stage('Test') {
            steps {

            }
        }
        stage('Deploy') {
            steps {

            }
        }
    }
}
```

Scripted

```
node {
    stage('Build') {

    }
    stage('Test') {

    }
    stage('Deploy') {

    }
}
```

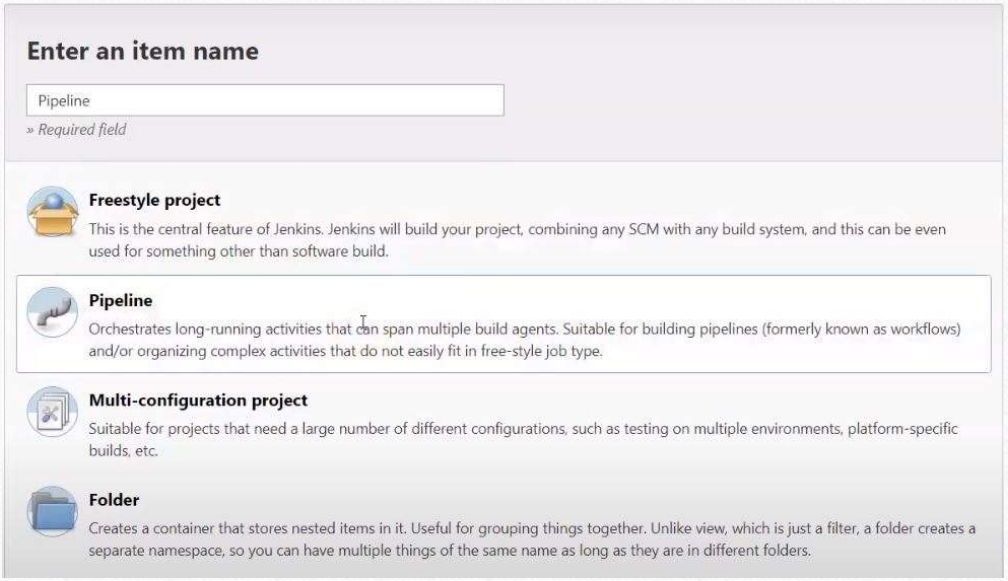
Execute Jenkins as a Java binary

1. Open the terminal window and enter cd <your-path>
2. Use the command java –jar ./Jenkins. war to run the WAR file.

Create A Jenkins Job

1. Open the web browser and open localhost:8080.
2. The Jenkins dashboard opens creates new jobs there.

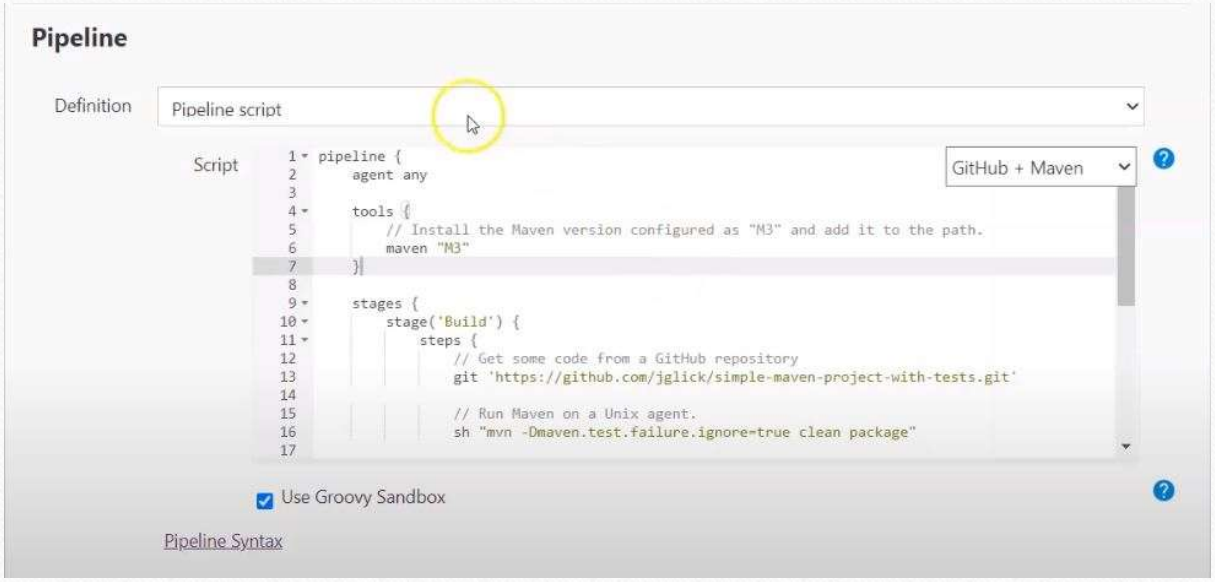
Create A Pipeline Job



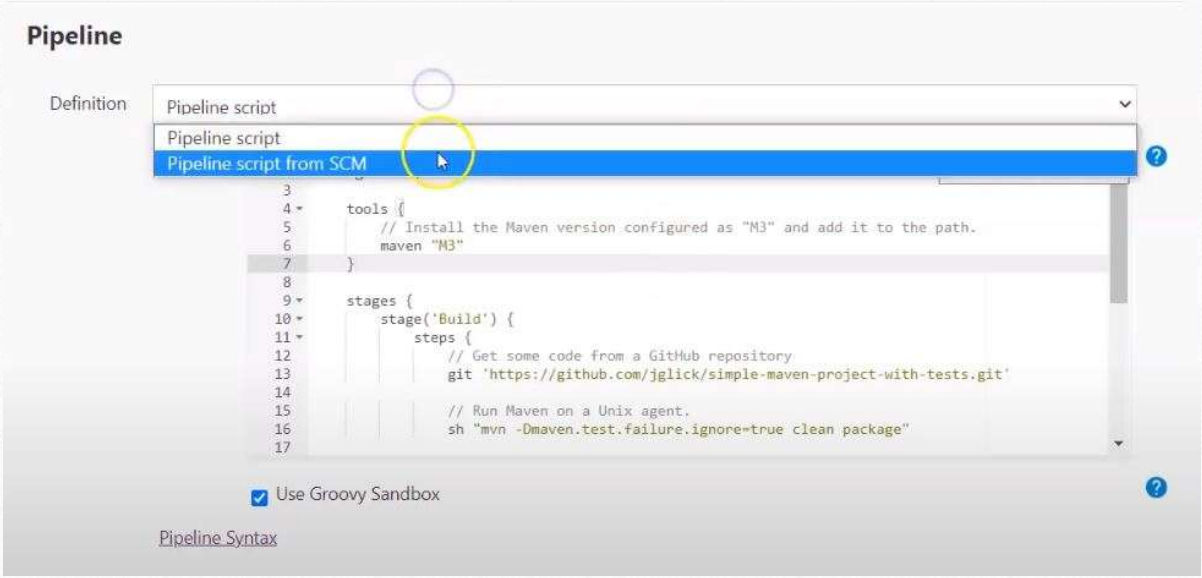
Create A Pipeline Job

1. Select and define what Jenkins job that is to be created.
2. Select Pipeline, give it a name and click OK.
3. Scroll down and find the pipeline section.
4. Either directly write a pipeline script or retrieve the Jenkins file from SCM (Source Code Management).

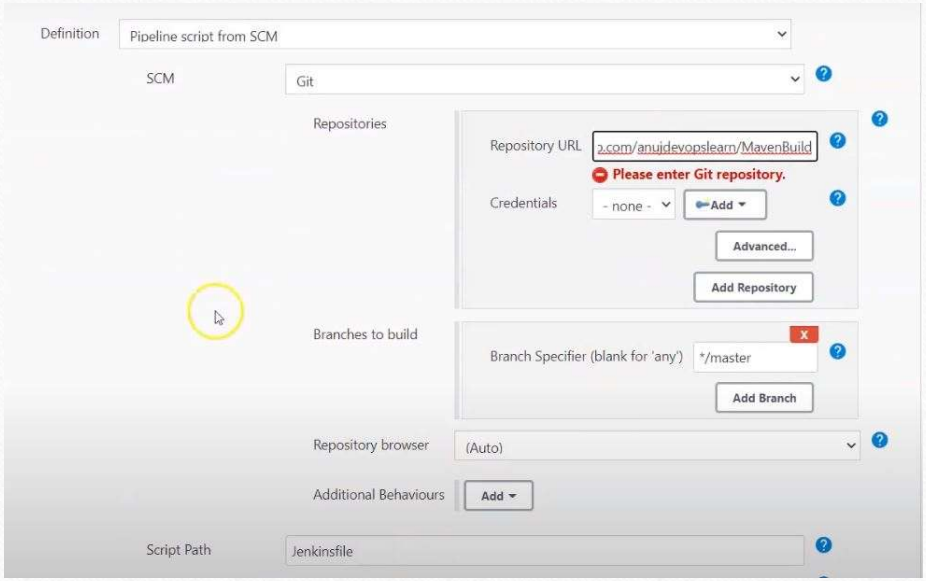
Create A Pipeline Job



Create A Pipeline Job



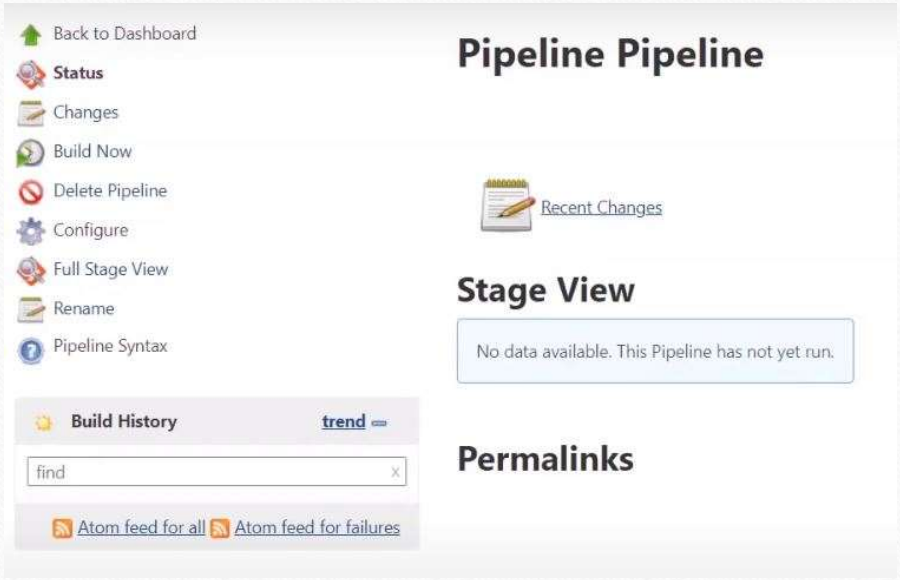
Create A Pipeline Job



Configure and Execute a Pipeline Job With a Direct Script

1. Choose Pipeline script as the Destination and paste the Jenkins file content in the Script from the GitHub.
2. Click on Save to keep the changes
3. Now click on the Build Now to process the build.
4. To check the output, click on any stage and click Log; a message will appear on the screen.

Configure and Execute a Pipeline Job With a Direct Script



Configure and Execute a Pipeline With SCM

1. Copy the GitHub repository URL by clicking on Clone or download.
2. Now, click on Configure to modify the existing job.
3. Scroll to the Advanced Project Options setting and select Pipeline script from the SCM option.
4. Paste the GitHub repository URL here.
5. Type Jenkinsfile in the Script, and then click on the Save button.
6. Next, click on Build Now to execute the job again.
7. There will be an additional stage, in this case, i.e., Declaration: Checkout SCM.
8. Click on any stage and click on Log.

After you have understood all the essential steps to build a CI/CD pipeline using Jenkins, a hands-on demonstration will serve as a cherry on the cake.

Configure and Execute a Pipeline With SCM