**Installation**

Let’s start by installing Jenkins. This installation is specific to systems operating on Ubuntu. Follow the below steps:

Step 1: Install Java  
$ sudo apt update  
$ sudo apt install openjdk-8-jdk  
  
Step 2: Add Jenkins Repository  
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add –  
  
Step 3: Add Jenkins repo to the system  
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'  
  
Step 4: Install Jenkins  
$ sudo apt update  
$ sudo apt install Jenkins  
  
Step 5: Verify installation  
$ systemctl status Jenkins  
  
Step 6: Once Jenkins is up and running, access it from the link:  
[http://localhost:8080](http://localhost:8080/)

**Most commonly used Jenkins Plugins**

Jenkins comes with over 2000 plugins and each plugin has unique functionality. But when it comes to software development most developers use a set of plugins, such as

* Maven
* Git
* Ant
* Docker
* Amazon EC2
* HTML publisher
* Copy artifact

Follow the below step to install the above plugins or any other Jenkins plugin.

Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available

In the filter text field enter the name of the plugin you want to install.

**Different Types of Jenkins Jobs**

Jenkins provides the option of choosing from different types of jobs to build your project.

Below are the types of jobs you can choose from:

* **Freestyle**

Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility. It can be used for any type of project.

* **Pipeline**

This project runs the entire software development workflow as code. Instead of creating several jobs for each stage of software development, you can now run the entire workflow as one code.

* **Multiconfiguration**

The multiconfiguration project allows you to run the same build job on different environments. It is used for testing an application in different environments.

* **Folder**

This project allows users to create folders to organize and categorize similar jobs in one folder or subfolder.

* **GitHub Organization**

This project scans your entire GitHub organization and creates Pipeline jobs for each repository containing a Jenkinsfile

* **Multibranch Pipeline**

This project type lets you implement different Jenkinsfiles for different branches of the same project.

**Jenkins Pipeline**

Jenkins pipeline is a single platform that runs the entire *pipeline as code.*Instead of building several jobs for each phase, you can now code the entire workflow and put it in a Jenkinsfile.

Jenkinsfile is a text file that stores the pipeline as code. It is written using the Groovy DSL*.*It can be written based on two syntaxes:

* **Scripted pipeline**

Code is written on the Jenkins UI instance and is enclosed within the node block

node {  
 scripted pipeline code  
}

* **Declarative pipeline**

Code is written locally in a file and is checked into a SCM and is enclosed within the pipeline block

pipeline {  
 declarative pipeline code  
}

**Build Pipeline**

Build pipeline can be used to chain several jobs together and run them in a sequence. Let’s see how to install Build Pipeline:

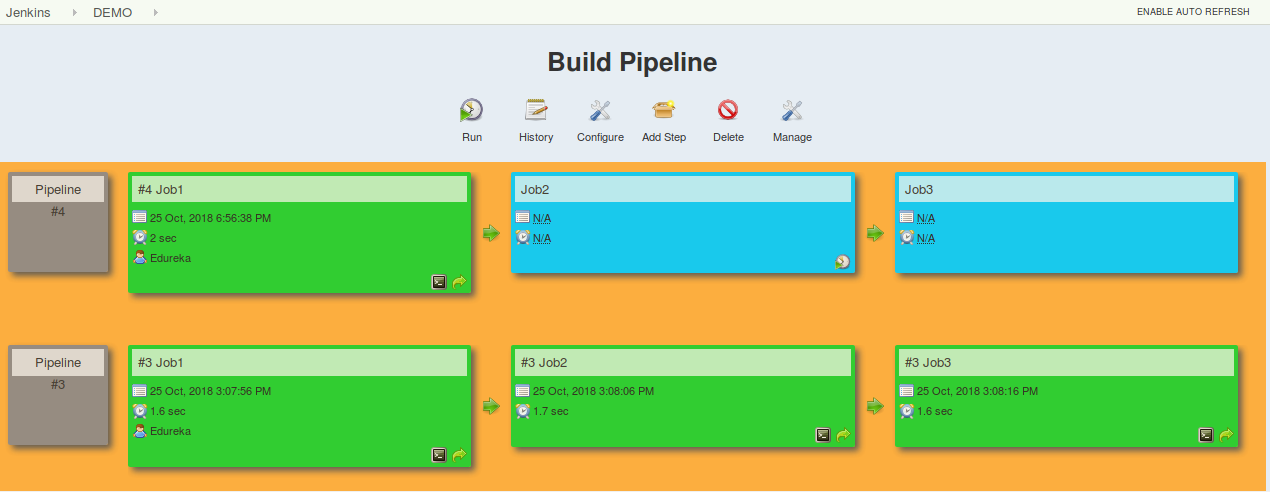
Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available

In the filter text field enter the name of the plugin you want to install.

**Build Pipeline Example**

Step 1: Create 3 freestyle Jobs (Job1, Job2, Job3)  
  
Step 2: Chain the 3 Jobs together  
Job1 -> configure -> Post Build -> Build other projects -> Job2  
Job2 -> configure -> Post Build -> Build other projects -> Job3  
  
Step 3: Create a build pipeline view  
Jenkins Dashboard -> Add view -> Enter a name -> Build pipeline view -> ok ->  
configure -> Pipeline flow -> Select Initial job -> Job1 -> ok  
  
Step 4: Run the Build Pipeline

Image for post



**Pipeline Concepts**

The below fundamentals are common to both, scripted and declarative pipeline:

1. **Pipeline:** A user-defined block which contains all the stages. It is a key part of declarative pipeline syntax.
2. **Node:** A node is a machine that executes an entire workflow. It is a key part of the scripted pipeline syntax.
3. **Agent:** instructs Jenkins to allocate an executor for the builds. It is defined for an entire pipeline or a specific stage.

**It has the following parameters:**

* *Any*: Runs pipeline/ stage on any available agent
* *None*: applied at the root of the pipeline, it indicates that there is no global agent for the entire pipeline & each stage must specify its own agent
* *Label*: Executes the pipeline/stage on the labelled agent.
* *Docker*: Uses docker container as an execution environment for the pipeline or a specific stage.

1. **Stages:** It contains all the work; each stage performs a specific task.
2. **Steps:** steps are carried out in sequence to execute a stage

**Jenkins Pipeline syntax example**

node {  
 stage(‘SCM checkout’) {  
 //Checkout from your SCM(Source Control Management)  
 //For eg: Git Checkout  
 }  
 stage(‘Build’) {  
 //Compile code  
 //Install dependencies  
 //Perform Unit Test, Integration Test  
 }  
 stage(‘Test’) {  
 //Resolve test server dependencies  
 //Perform UAT  
 }  
 stage(‘Deploy’) {  
 //Deploy code to prod server  
 //Solve dependency issues  
 }  
}

**Create your first Jenkins Pipeline**

After installing Jenkins, building jobs using the Build pipeline and briefly discussing pipeline concepts, let’s see how to create a Jenkins pipeline.

Follow the below steps to create both, a scripted pipeline and a declarative pipeline:

Step 1: Log into Jenkins and select ‘New Item from the Dashboard'  
  
Step 2: Next, enter a name for your pipeline and select ‘Pipeline project’. Click ‘ok’ to proceed  
  
Step 3: Scroll down to the pipeline and choose if you want a Declarative or Scripted pipeline  
  
Step 4a: If you want a Scripted pipeline, then choose ‘pipeline script’ and start typing your code  
  
Step 4b: If you want a Declarative Pipeline, select ‘Pipeline script from SCM’ and choose your SCM and enter your repository URL  
  
Step 5: Within the Script path is the name of the Jenkinsfile that is going to be accessed from your SCM to run. Finally click on ‘apply’ and ‘save’

**Jenkins Tips and Tricks**

**Start, stop and restart Jenkins**

Follow the below command to start, stop and restart Jenkins through the CLI.

$ sudo service jenkins restart  
$ sudo service jenkins stop  
$ sudo service jenkins start

**Deploy a custom build of a core plugin**

Step 1: Stop Jenkins.  
  
Step 2: Copy the custom HPI to **$Jenkins\_Home/plugins**.  
  
Step 3: Delete the previously expanded plugin directory.  
  
Step 4: Make an empty file called **<plugin>.hpi.pinned**.  
  
Step 5: Start Jenkins.

**Schedule a build periodically**

Jenkins uses Cron expressions to schedule a job. Each line consists of 5 fields separated by TAB or whitespace:

Syntax: (Minute Hour DOM Month DOW)  
  
MINUTE: Minutes in one hour (0-59)  
  
HOURS: Hours in one day (0-23)  
  
DAYMONTH: Day in a month (1-31)  
  
MONTH: Month in a year (1-12)  
  
DAYWEEK: Day of the week (0-7) where 0 and 7 are sunday  
  
Example: H/2 \* \* \* \* (schedule your build for every 2 minutes)

Try this example:

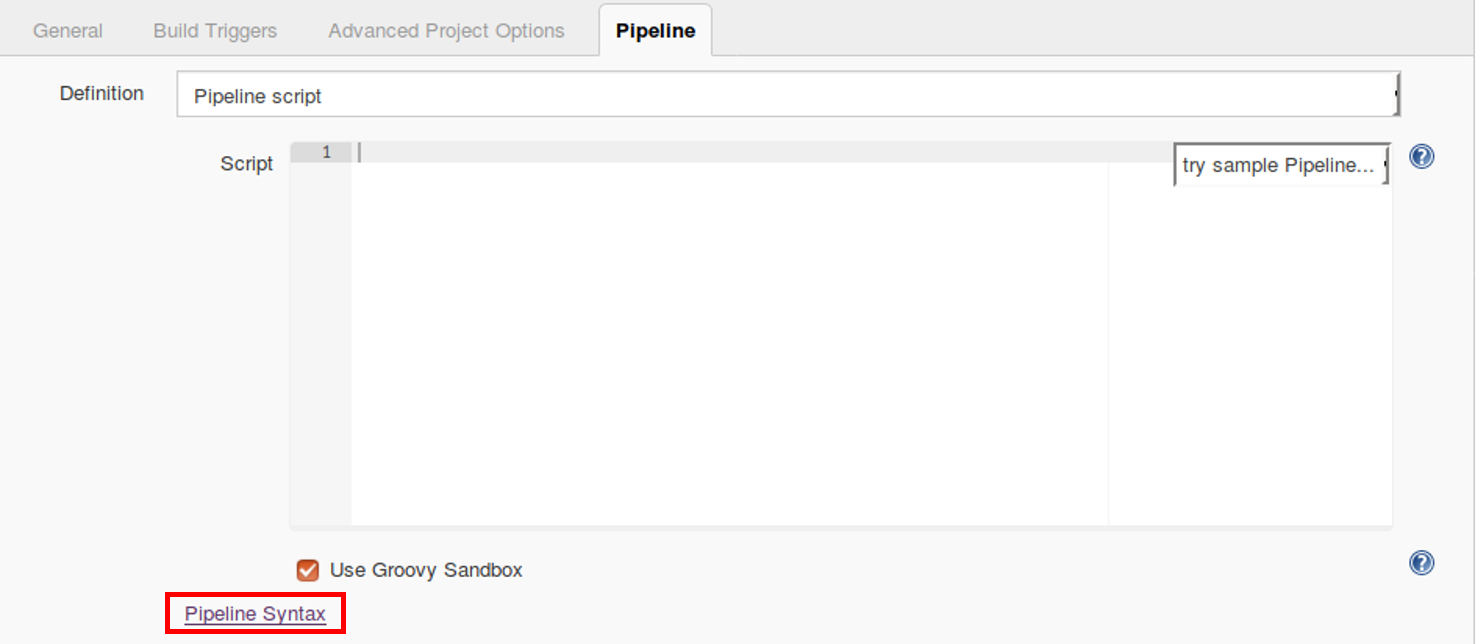
H/2 \* \* \* \* (schedules your build for every 2 minutes)

**Snippet Generator**

A tool that lets users generate code for individual steps in a scripted pipeline. Let’s look at an example:

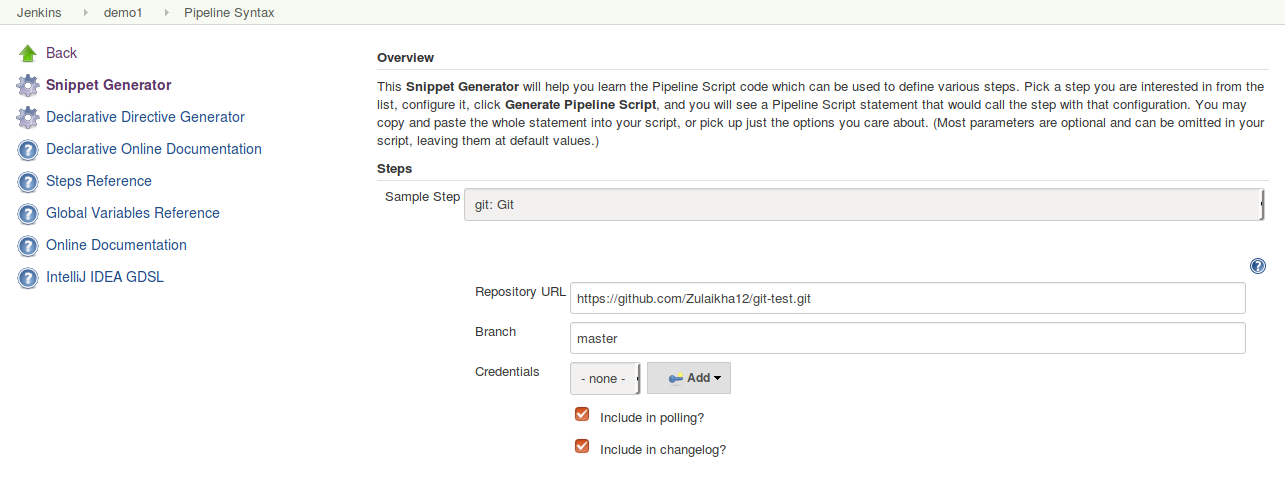
Step 1: Create a pipeline job > configure  
  
Step 2: Select pipeline script from pipeline definition  
  
Step 3: Click on Pipeline syntax > snippet generator  
  
Step 4: Step > select Git > enter repo URL  
  
Step 5: Scroll down > Generate pipeline script  
  
Step 6: Copy the script into your pipeline script UI

Image for post



Below is an image of the Snippet Generator. You can select from a variety of steps and generate a code for each step.

Image for post



Below is an image of the Scripted pipeline UI with the code generated from snippet generator

Image for post

