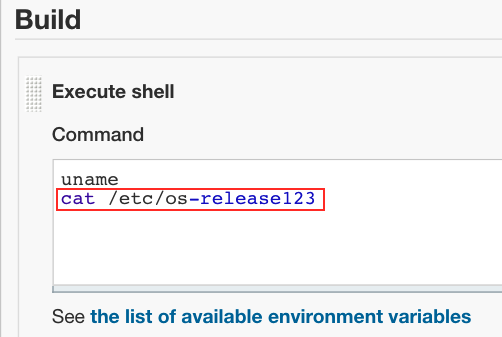
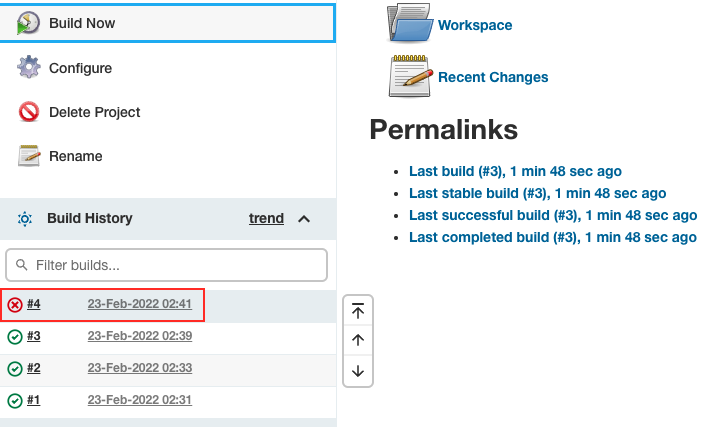
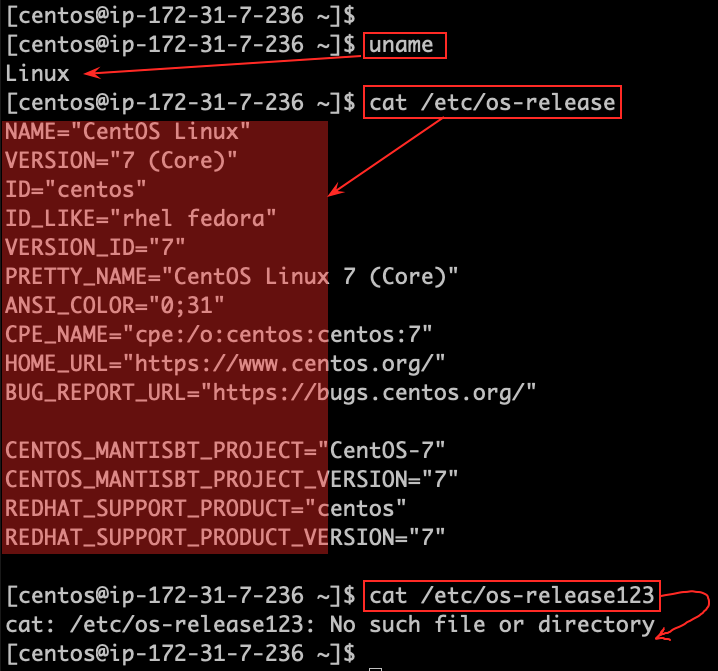


* When the job configuration is not updated properly, the jobs will fail.



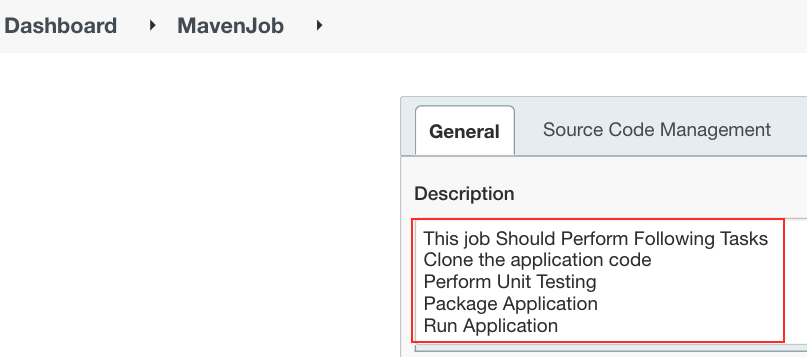






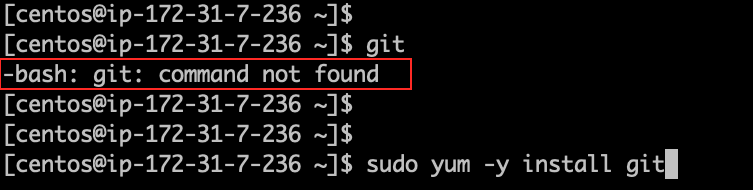
Maven - JOB

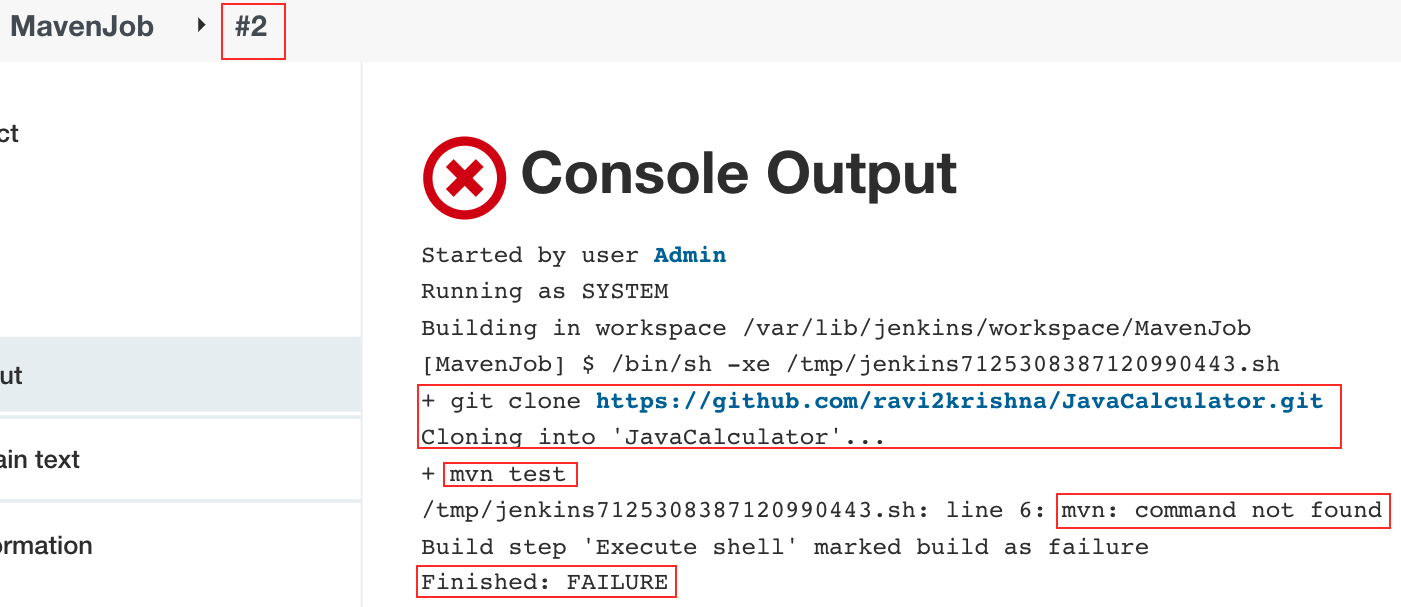
* Based on the job you are working with, make sure all the required softwares and tools are setup and ready
* Example, if we try to clone the software without installing git, it doesn’t work, as jenkins is not a vcs client, git is the client
* Same way we need to install maven



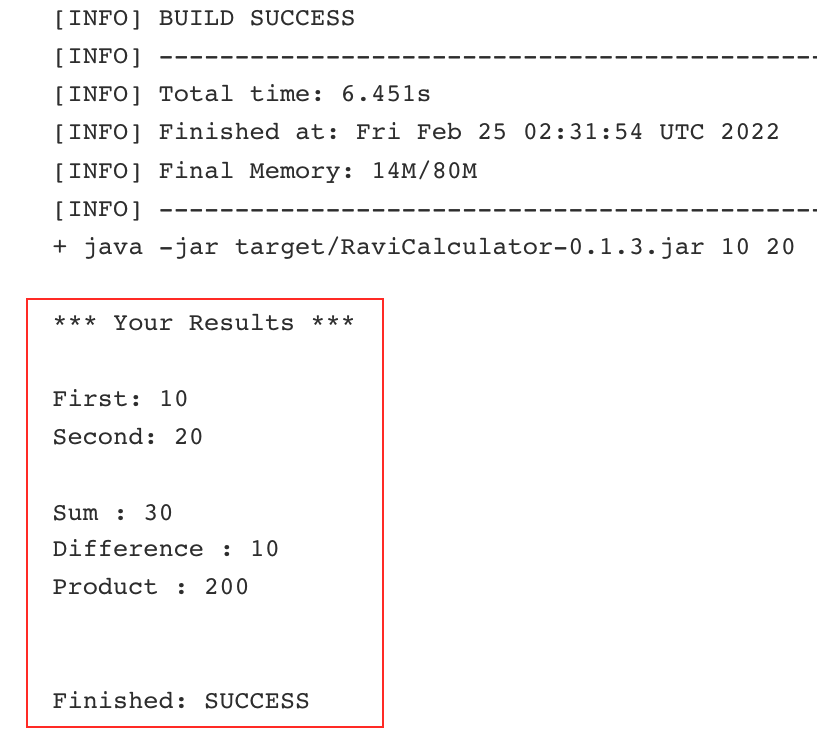








* Building Maven Job using jenkins, requires git and maven installations
  + sudo yum -y install git
  + sudo yum -y install maven

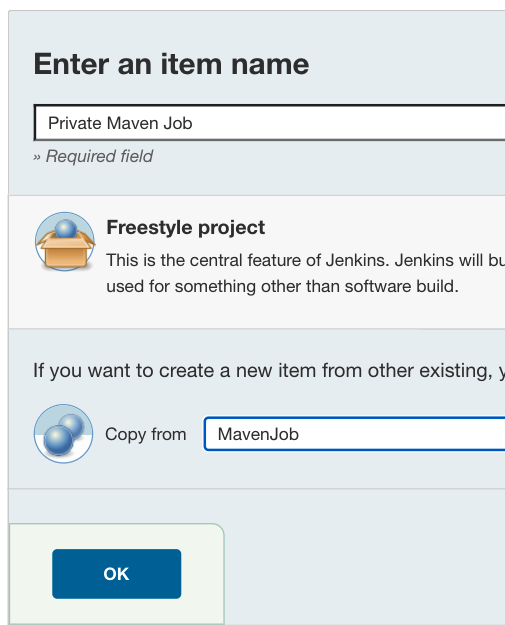


**Maven Setup For Jenkins**

<https://phoenixnap.com/kb/how-to-install-apache-maven-on-centos-7>

**Private Maven Job**

* If we use private github repository, then build will fail as, private repositories work only if we use credentials or keys
* fatal: could not read Username for 'https://github.com': No such device or address



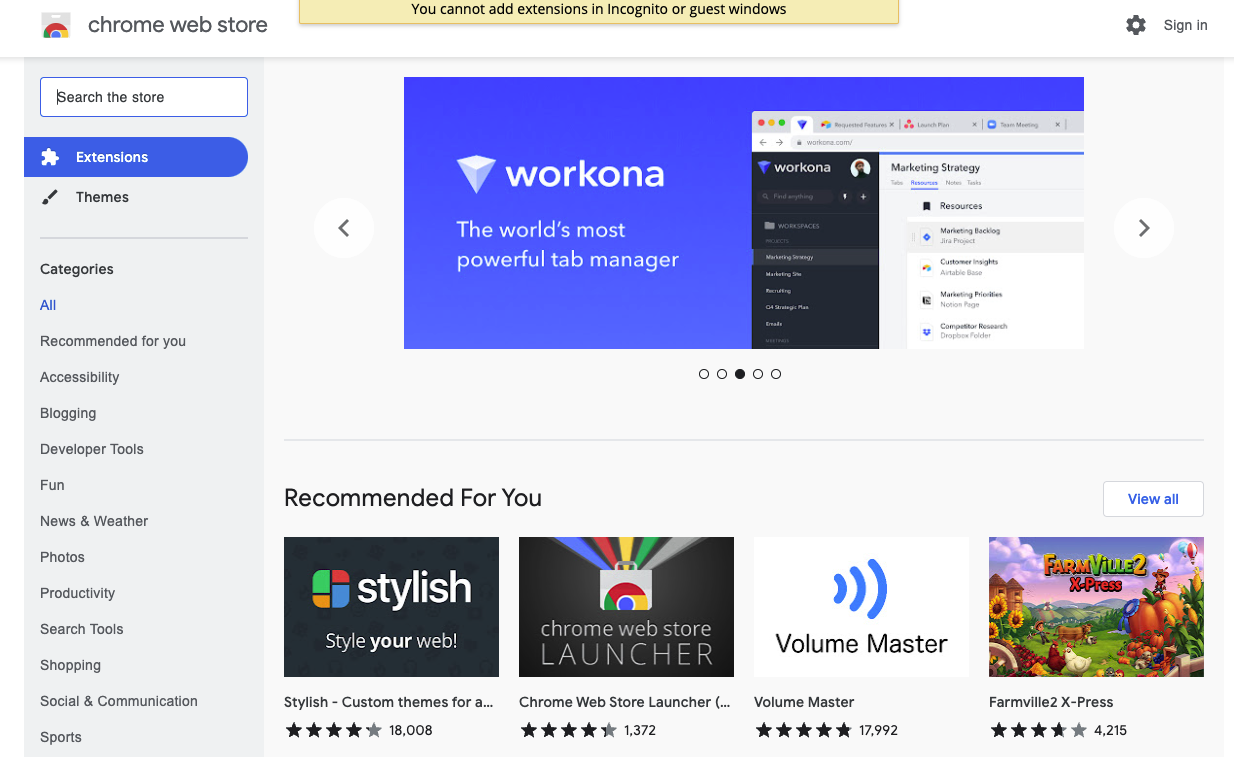
* If you have private repos, use the below syntax
  + git clone https://<github-username>:<github-token>@github.com/<your-repository>.git

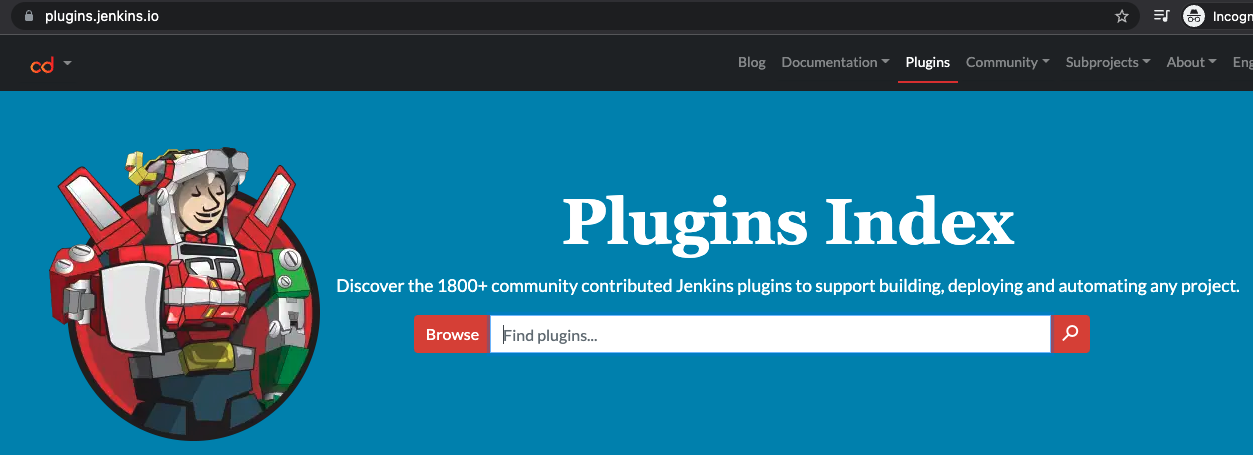
Problems

* See the results in human readable report formats
* Use some external scripts or develop external scripts to manage build process
* Working with private repositories -
  + > git clone https://username:token@github.com/username/repo\_name.git
  + Securely pass the credentials
* Manually pull the changes and manually perform builds
* Manually update developers about the build status, when builds fail

# **Plugins**

* Plugins are the primary means of enhancing the functionality of a Jenkins environment to suit organization or user-specific needs.
* Like Google Chrome Browser has addons to extend chrome functionality, we have jenkins plugins





* There are over a thousand different plugins which can be installed on Jenkins and to integrate various build tools, cloud providers, analysis tools, report generations and much more.
* Plugins can be automatically downloaded, with their dependencies, from the Update Center. The Update Center is a service operated by the Jenkins project which provides an inventory of open source plugins which have been developed and maintained by various members of the Jenkins community.

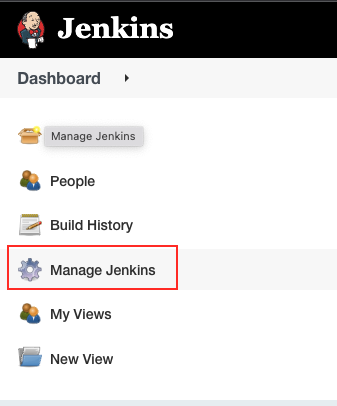
Fixes

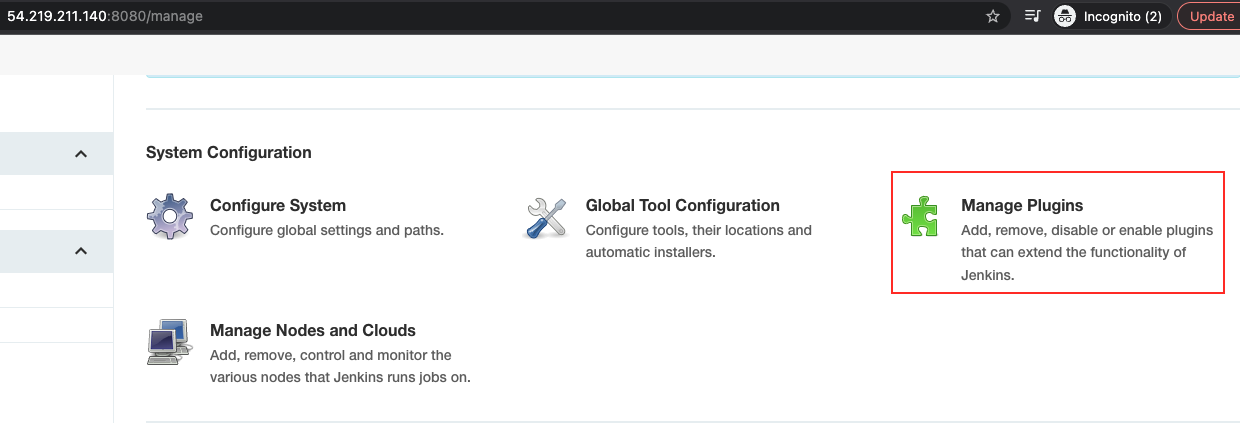
* Folders plugin allows users to create "folders" to organize jobs.
* Installing GitHub Plugin, will give access to Source Code Management, where we can add repositories along with credentials, as credentials plugin will be installed
* In Build Triggers, we can get new trigger option called GitHub Trigger, which will automatically pull the changes into jenkins, when developers update new changes
* Installing Junit Plugin will give the ability to publish junit test reports from Post-build Actions

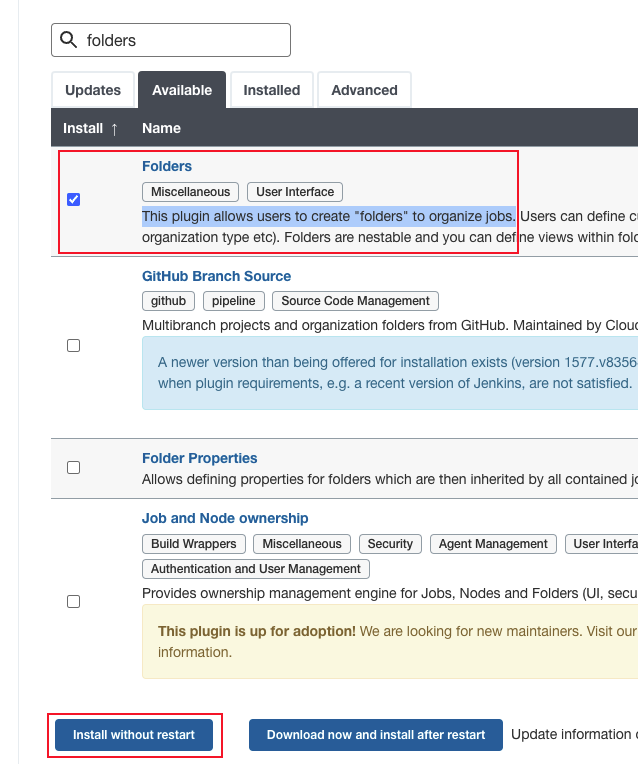
Installing Plugin

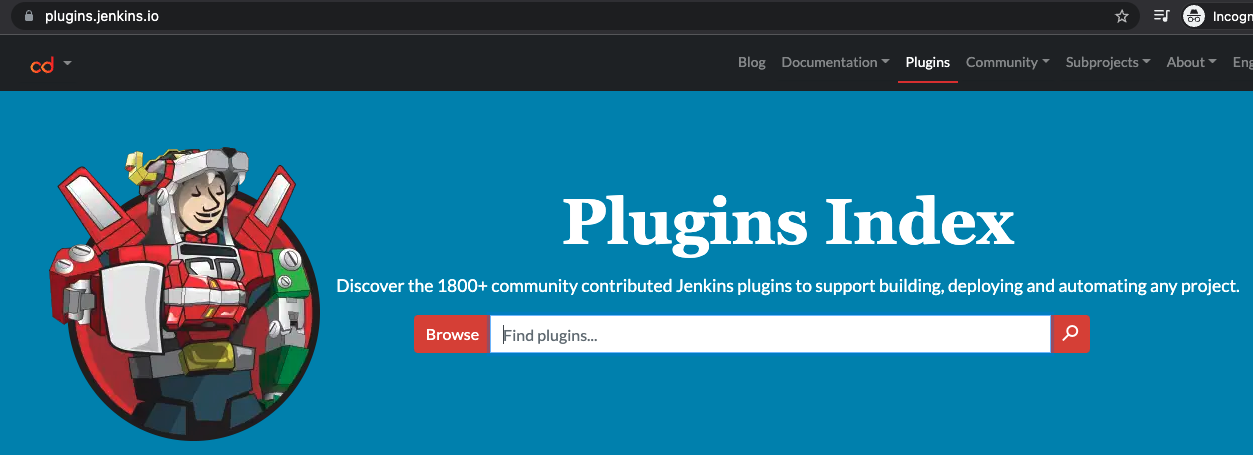
The simplest and most common way of installing plugins is through the Manage Jenkins > Manage Plugins

Check Plugins first, Available

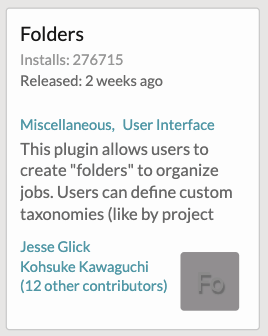


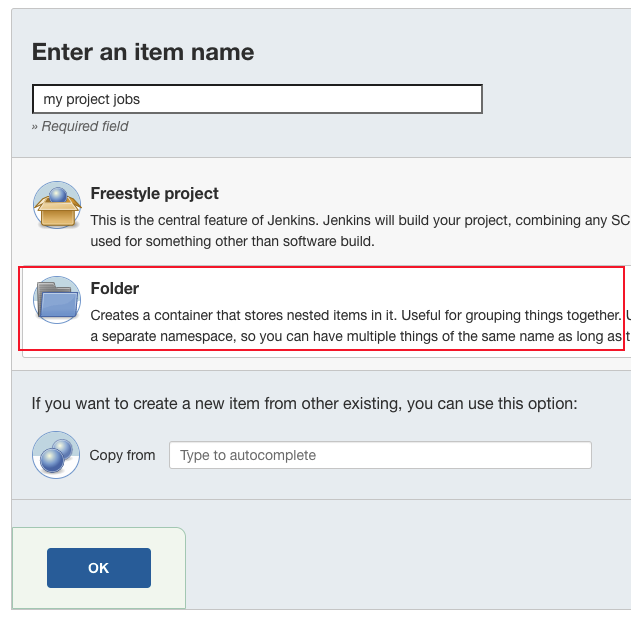


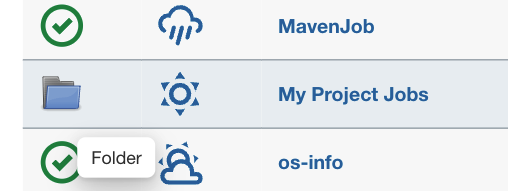


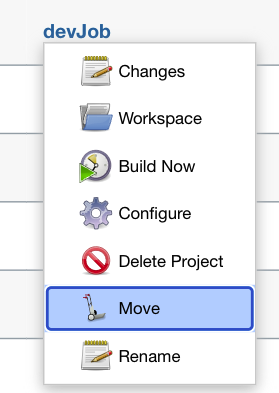


* Folders - plugin allows users to create "folders" to organize jobs.

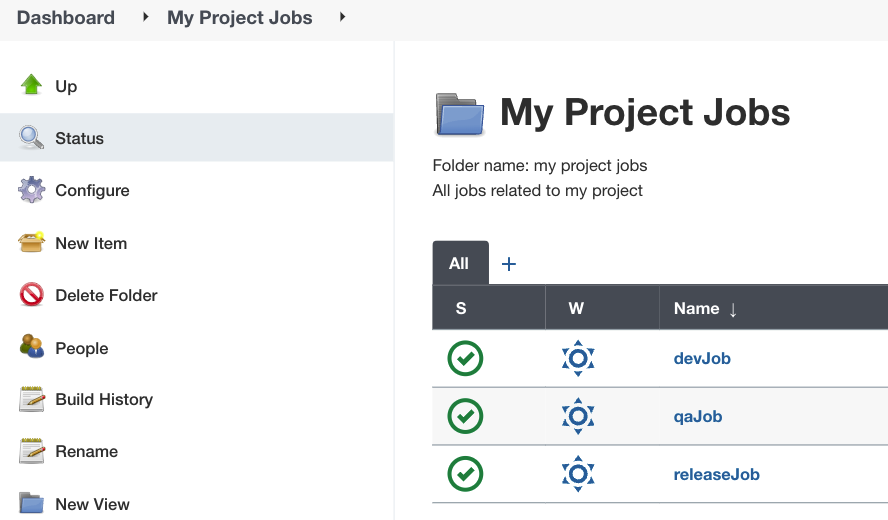




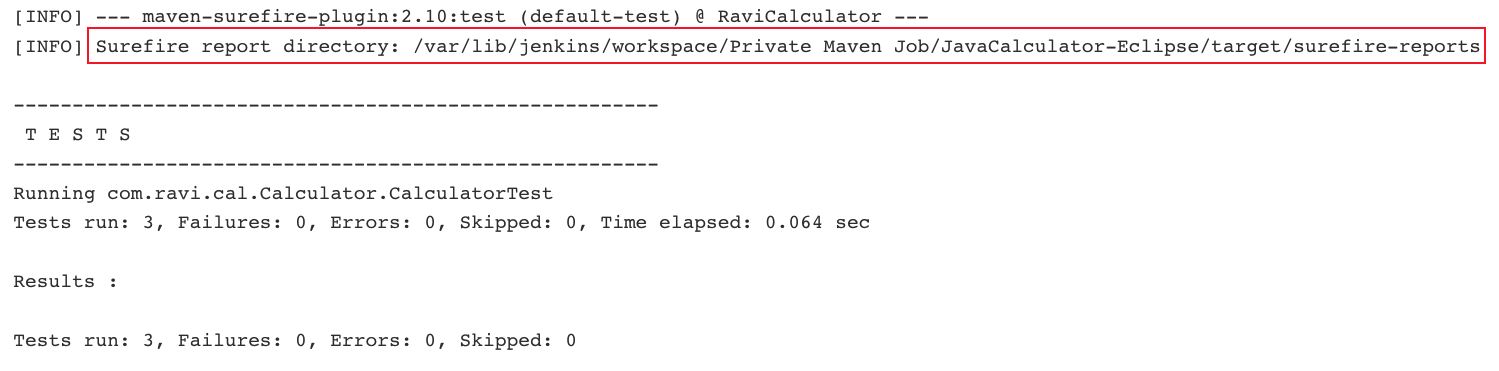


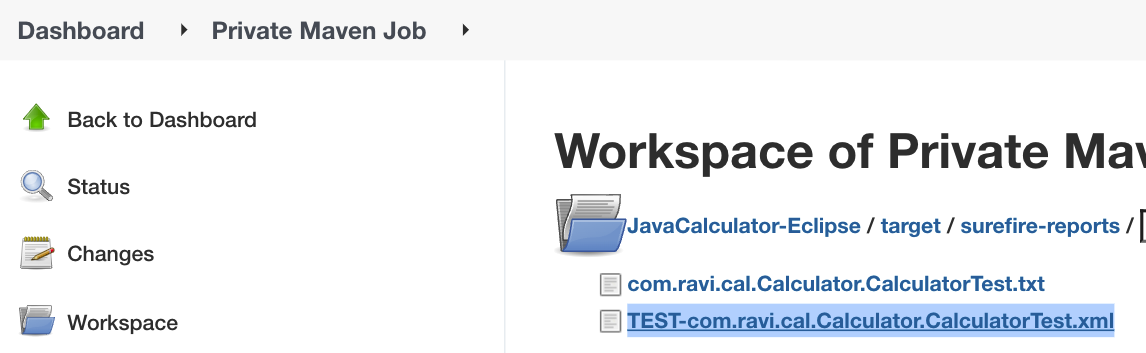


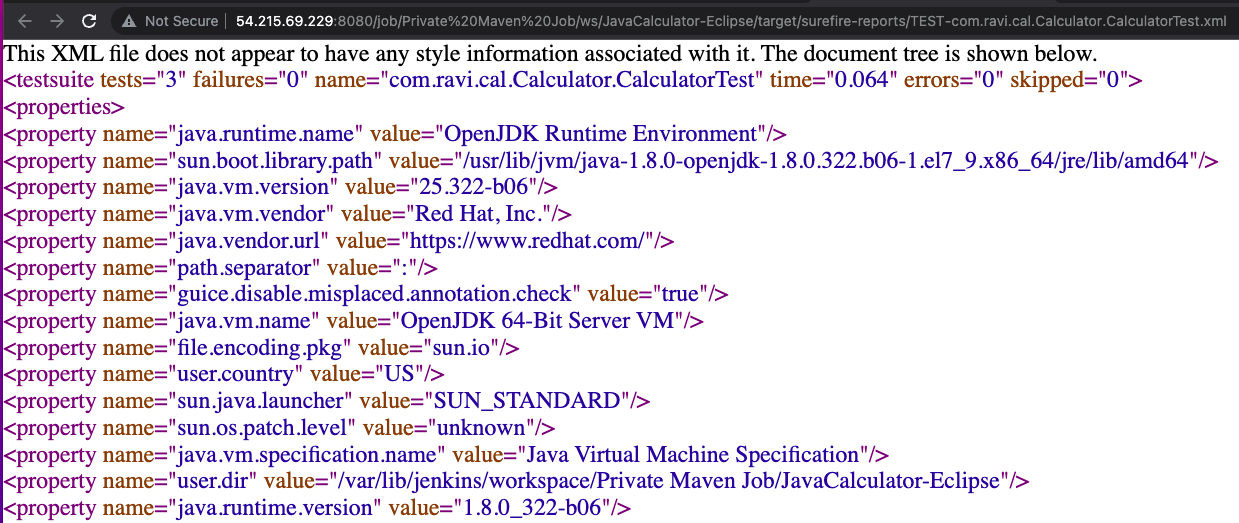




Junit Results



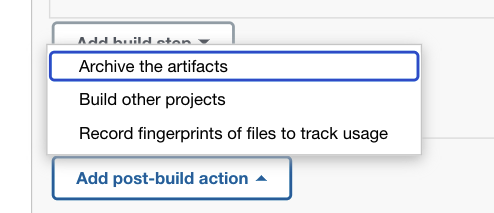




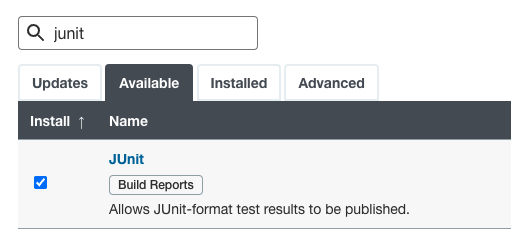
If you check at the unit test cases, you don't have any testing results stored.

Now you can actually use the junit plugin, to publish the test results.

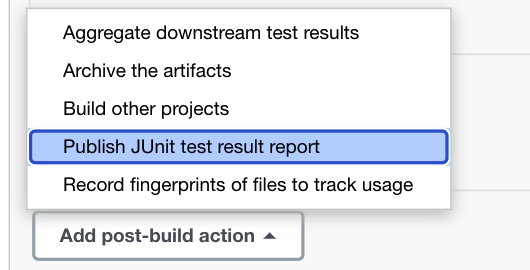
Look at the post build actions

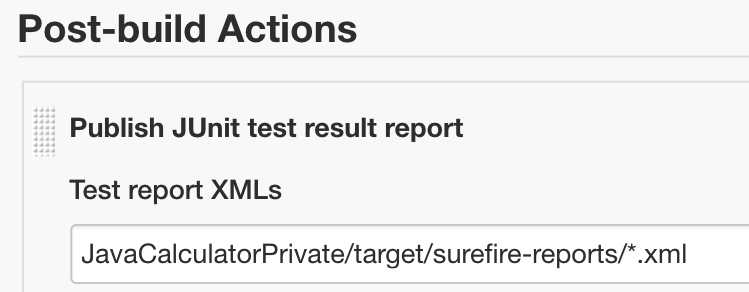


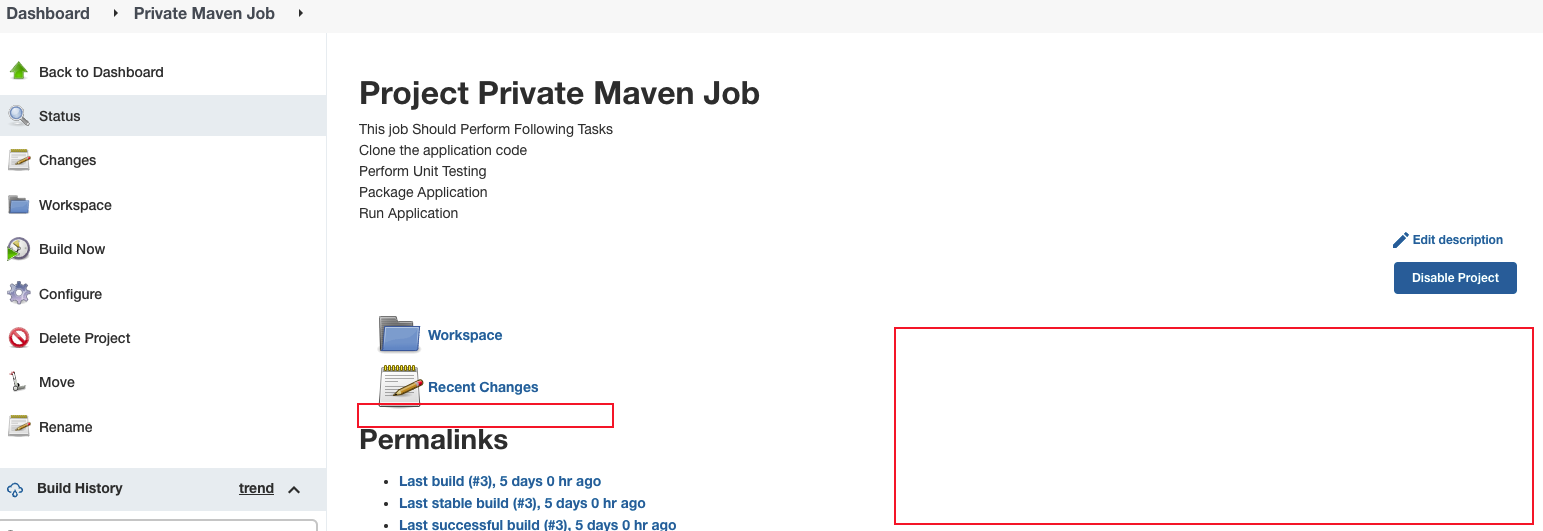
Junit Plugin



Now in Post build actions, we got publish junit results



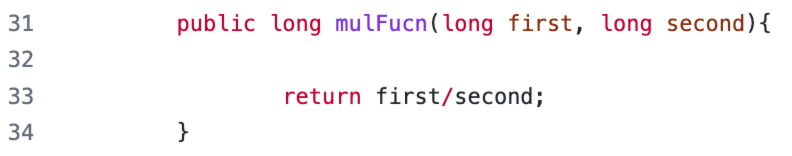


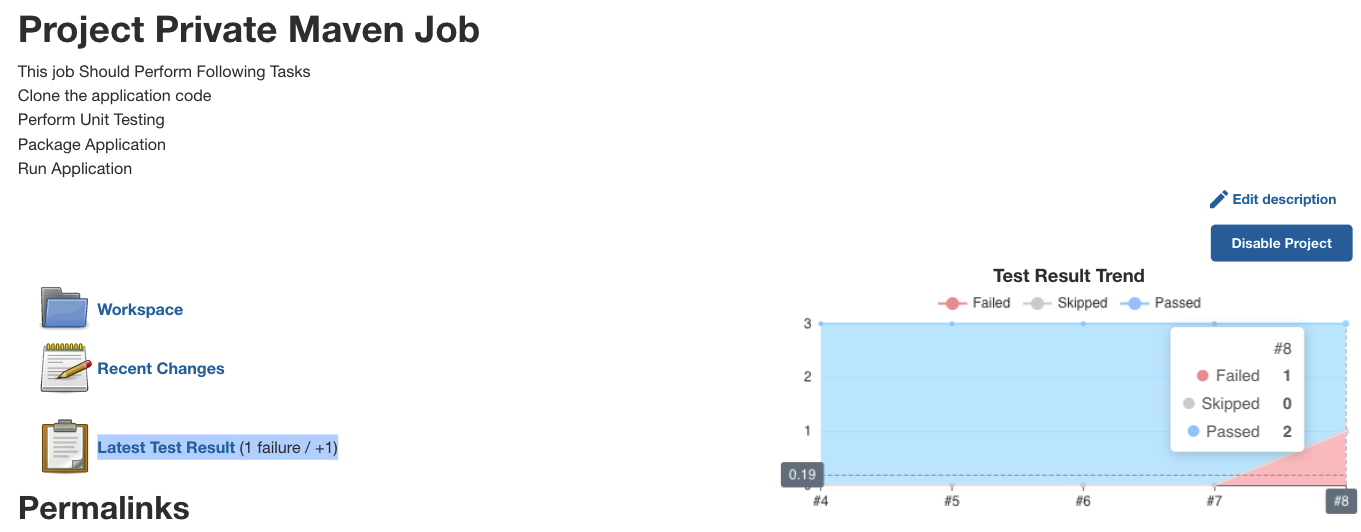


* Now after refreshing the job, you’ll new Build results being displayed



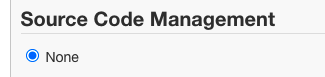
* Now let’s mess up the application by adding bugs which will result in a test failure

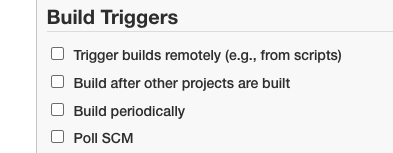


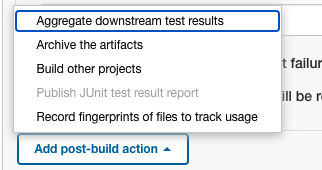


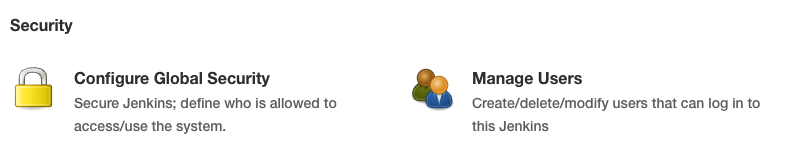
Additional Plugins

In Job Configuration > Source Code Management > We Don't have anything



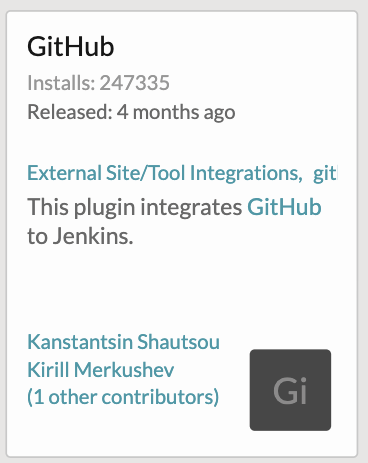


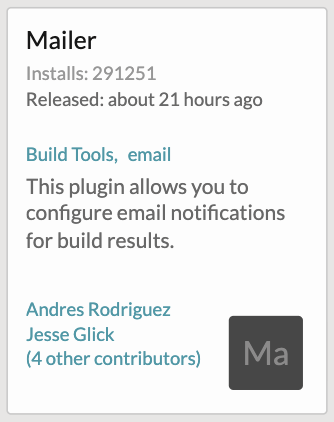


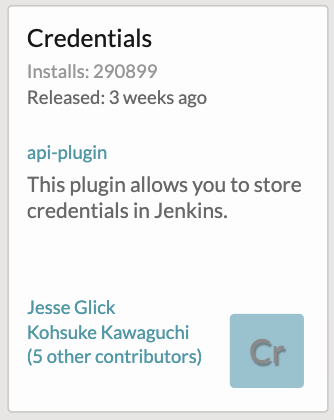


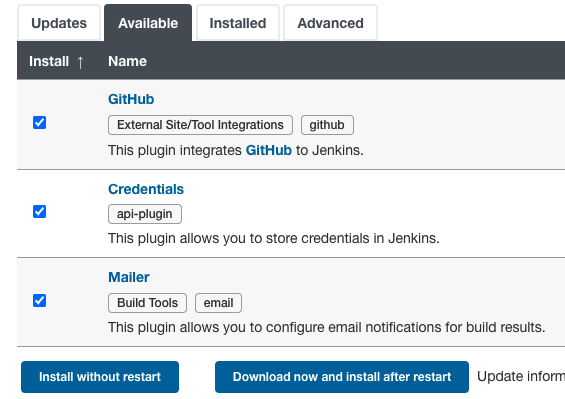
Github Plugin

* Let’s say we have a private repository and we need credentials to work with private repository
* Now if we use open credentials, it’s insecure and can be easily prone to misuse the credetains or easily prone to malicious attacks
* So to resolve this we can GitHub Plugin

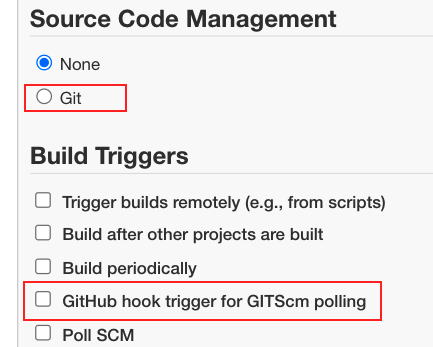


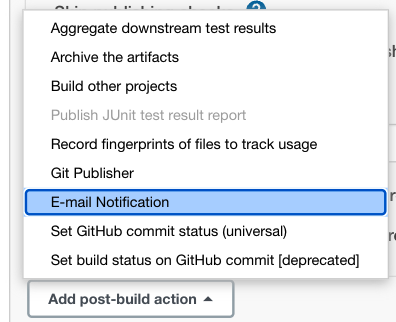


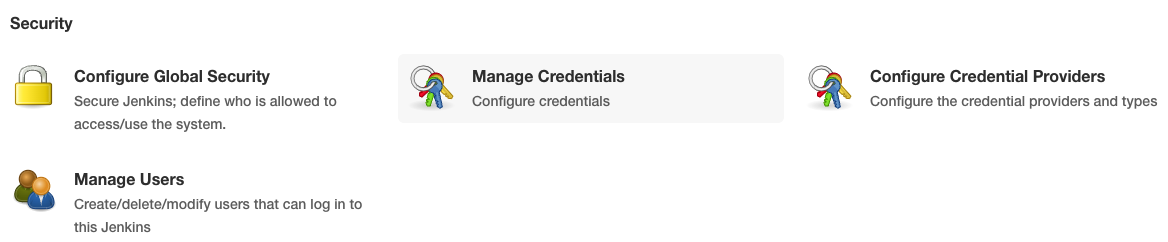




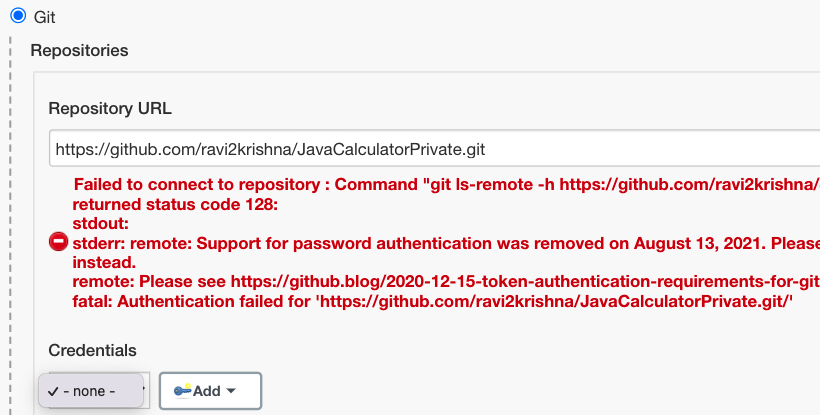
* After installing plugins, we got new options

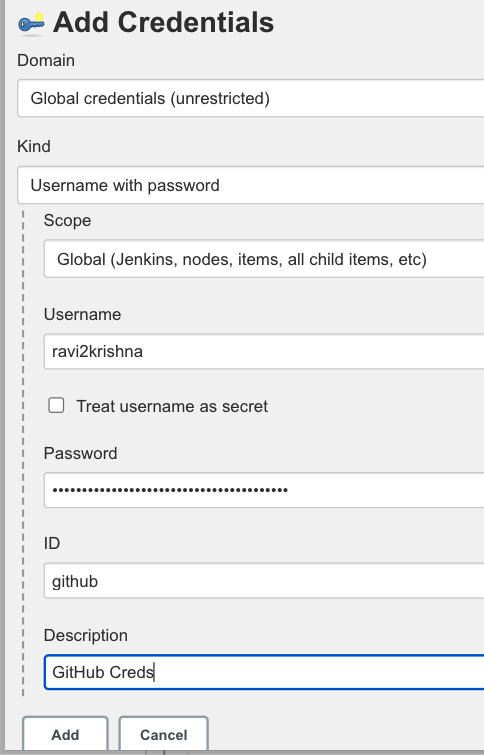


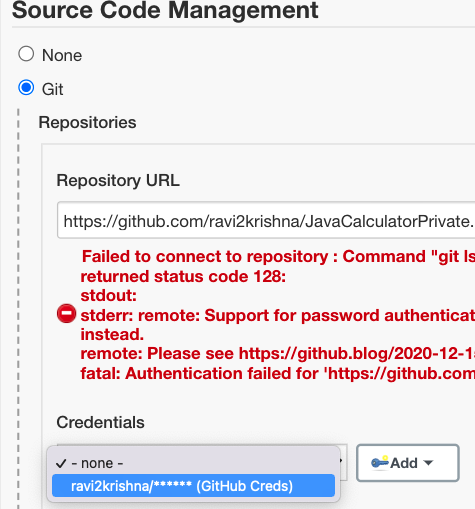


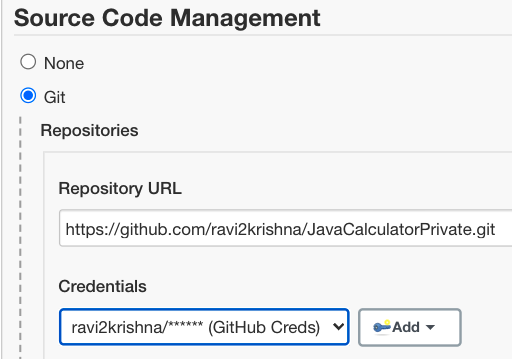


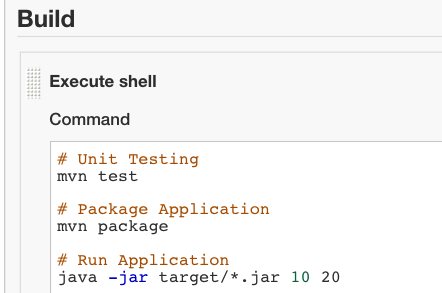
Private Repo - Job



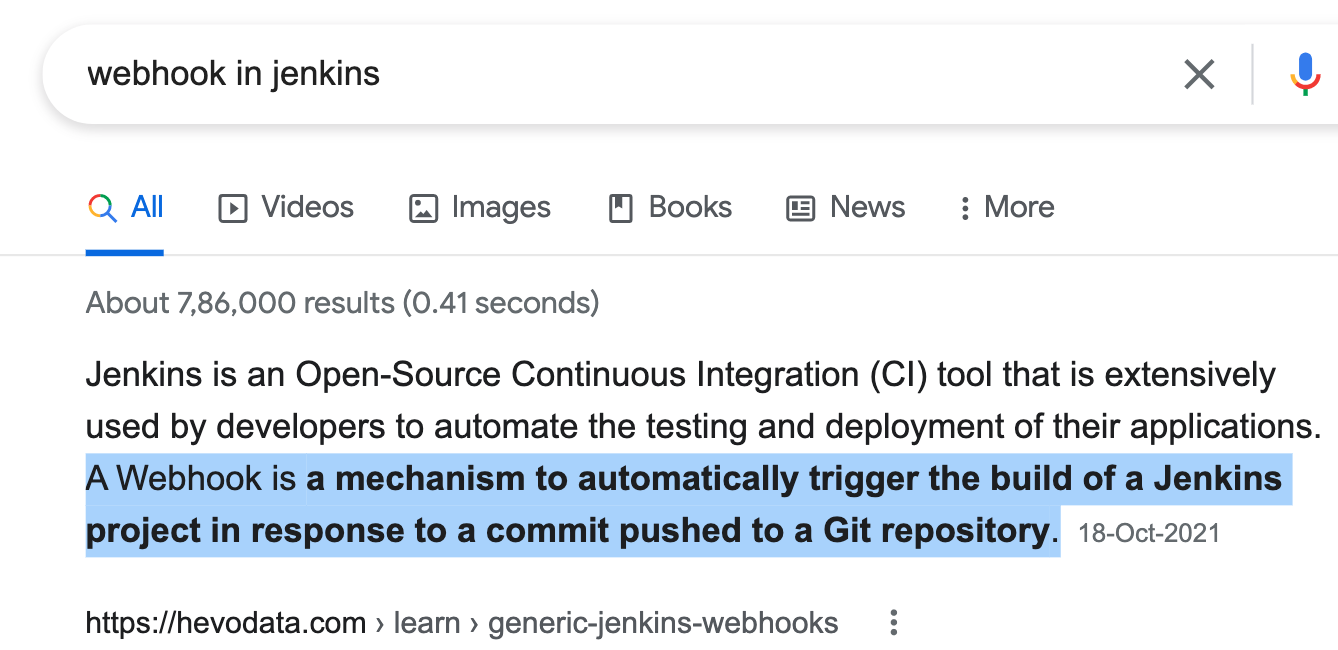




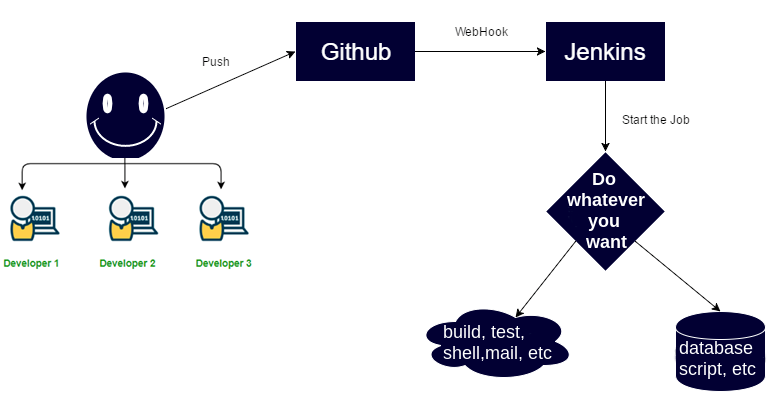




# **WebHooks**



* Webhooks in GitHub allow external services to be notified when certain events happen
  + { external service - jenkins }
* One of the basic steps of implementing CI/CD is integrating your Source Code Repository tools with your CI Tools.
* This saves you time and keeps your project updated all the time. One of the most popular and valuable SCM tools is GitHub. Will see how to integrate Jenkins with GitHub projects.
* The integration will help you to:
* Schedule your build
* Pull your code and data files from your GitHub repository to your Jenkins machine
* Automatically trigger each build on the Jenkins server, after each Commit on your Git repository

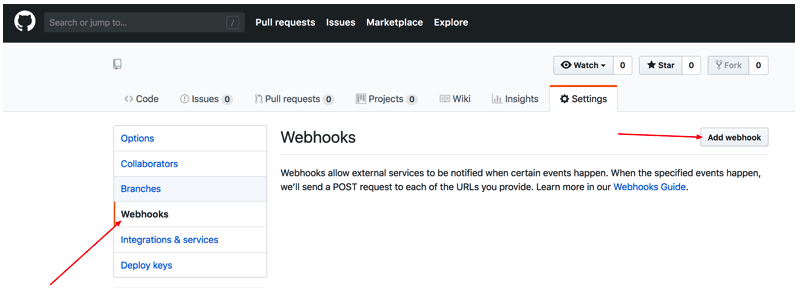


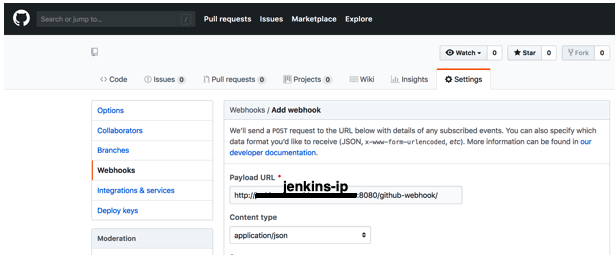
A diagram of a programming language

Description automatically generated

A screenshot of a computer

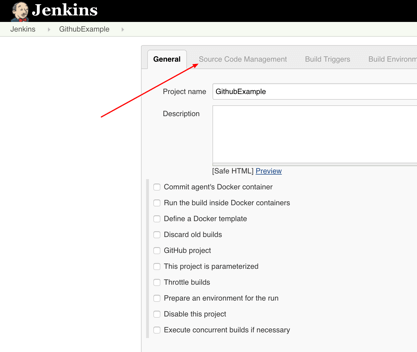
Description automatically generated

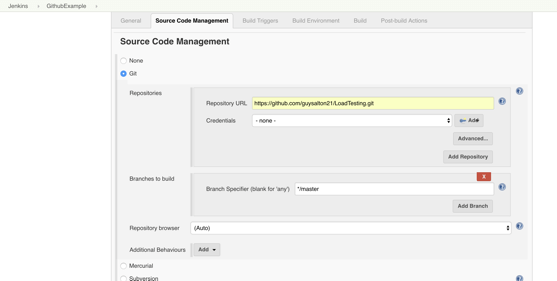


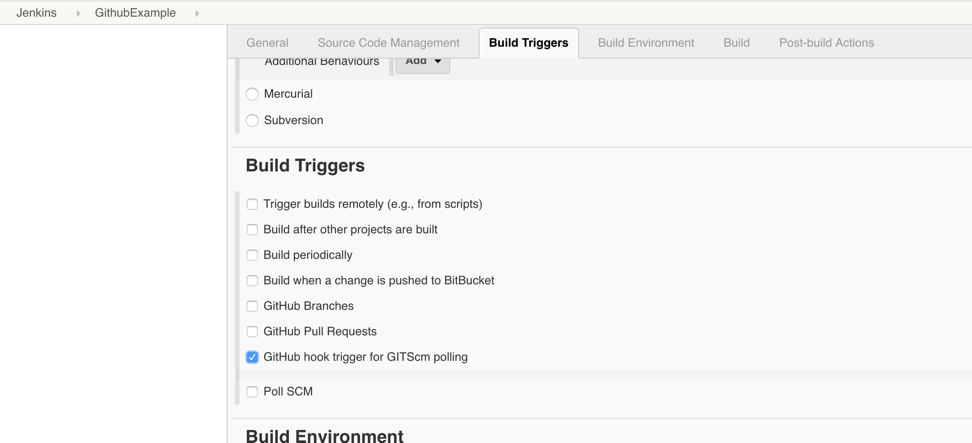


A screenshot of a computer

Description automatically generated







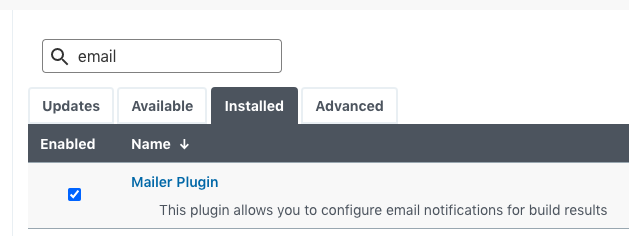
A black text on a white background

Description automatically generated

# **Notifications**

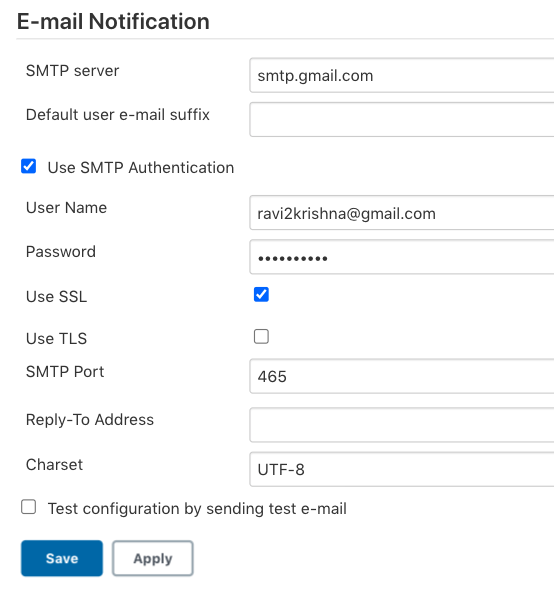
Email - Notify Build

* Jenkins provides you with an email notification service through which you can report the build status and testing results to the team.
* If the build is not successful then the team of developers are notified about the status of the build. This can be done with the help of an Mailer plugin in Jenkins
* Using the email plugin, you configure the email details of the concerned person who should be notified in case of build failure.
* Once the developer is notified about the error, he then fixes it and again pushes the code to GitHub. After this Jenkins again pulls the code from GitHub and prepares a fresh build.
* Use the following steps to setup the notifications in Jenkins
* Go to the Jenkins home page and click the ‘Manage Jenkins’ menu option. Then, select the ‘Configure System’ option > You don't see any Email Notification option in there
* Jenkins home page > Manage Jenkins > Manage Plugins > Search : email > Select : Mailer plugin > Install



## **GMAIL Setup**

* Jenkins home page > Manage Jenkins > Configure System > Email Notification >
* Enter the SMTP server name under ‘Email Notification’. Click the ‘Advanced’ button and then click the checkbox next to the ‘Use SMTP Authentication’ option. Now, set the following fields.
* SMTP server name : smtp.gmail.com
* User name: user\_email\_id@gmail.com
* Password: 123456
* Use SSL : Checked
* SMTP Port: 465



# **DISTRIBUTED BUILDS**

## Jenkins Master/Slave or Worker Configuration

* Sometimes many build machines are required if there are instances wherein there are larger and heavier projects which get built on a regular basis. And running all of these builds on a central machine may not be the best option.
* In such a scenario, one can configure other Jenkins machines to be slave or wroker machines to take the load off the master Jenkins server.
* Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.
* To address the above stated needs, Jenkins distributed architecture was introduced.

Jenkins Distributed Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds. In this architecture, Master and Slave communicate through TCP/IP protocol.

In master-slave architecture of Jenkins, master represents basic installation of Jenkins and it handles all tasks for build system.

Jenkins master is used to handle following things:

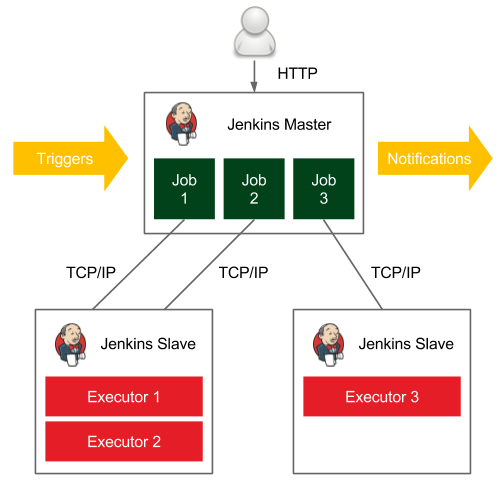
* Scheduling build jobs.
* Dispatching builds to the slaves for the actual execution.
* Recording and presenting the build results.
* A Master instance of Jenkins can also execute build jobs directly.

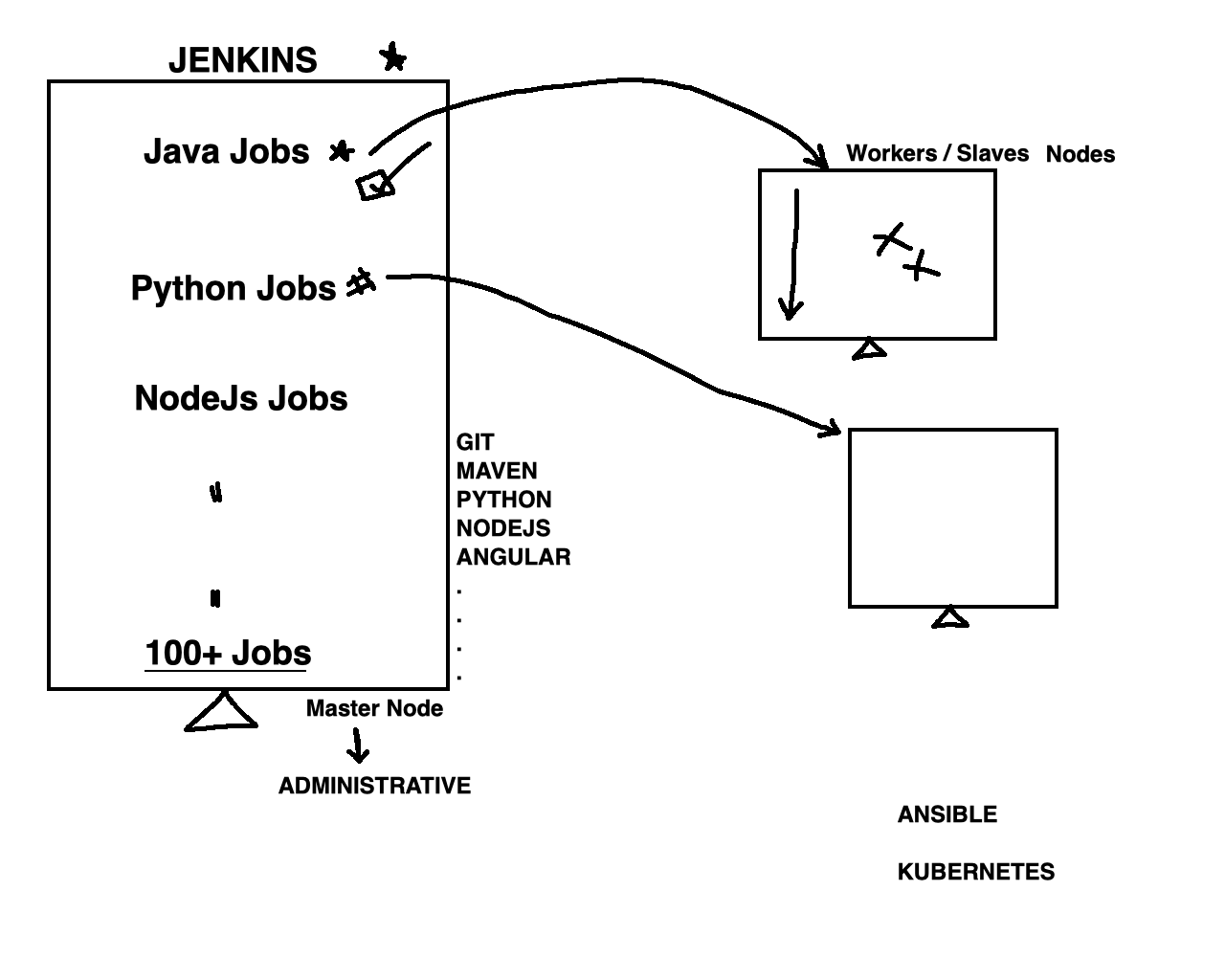
A slave is a computer that is set up to offload build projects from the master and once connection is established between master and slave , tasks distribution is done automatic. Each slave runs a separate program called a “slave agent” . There is no need to install the full Jenkins on a slave.

Following are the characteristics of Jenkins Slaves:

* It hears requests from the Jenkins Master instance.
* Slaves can run on a variety of operating systems.
* The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.

Below diagram shows the sample master slave connection. It consists of a Jenkins Master which is managing two Jenkins Slave.





Node

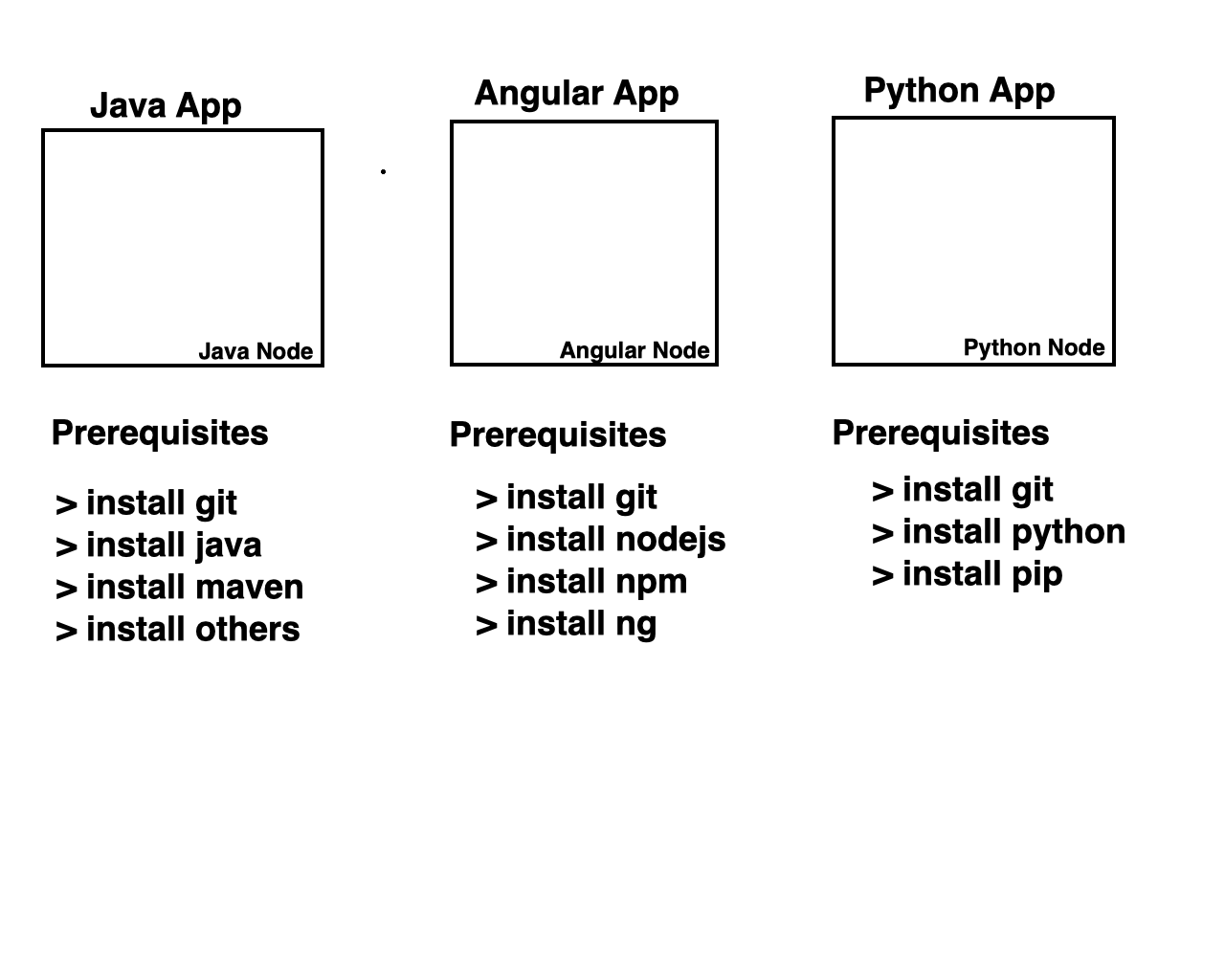
Now create a new centos server(machine) where you will be getting new ip {for example 172.31.30.45}

Node Prerequisites

* Create a Jenkins user in Slave, where access should be granted to master jenkins user
* JRE (Java Runtime Environment ) need to installed on every slave machine

NOTE: Slaves should have all the prerequisites in place first, let say you are building a maven job, where the source code is available in Github, then in the slave machine you should have set of following applications installed:

* Git
* Java
* Maven



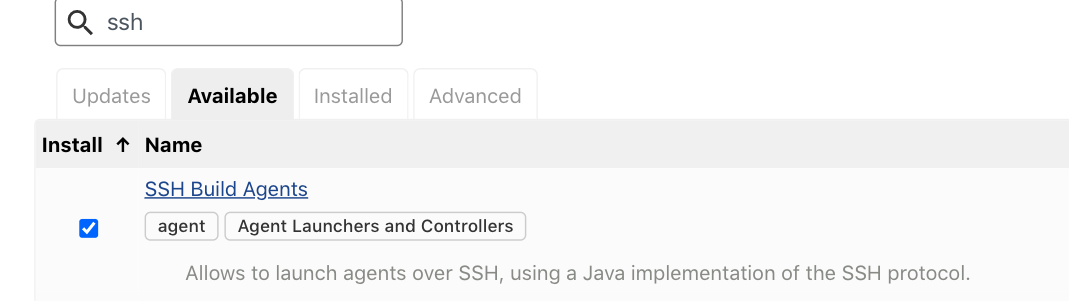
Configuring Node

So we talked about this a lot, that we don’t want to do builds much on master.

We are going to create slave nodes, slaves don't need to know anything about implementation, they should be able to just run jobs and be controlled by a master.

We are going to use these slave nodes in order to off-load the build processing to other machines so that the master server doesn’t get CPU, IO or N/W load etc in managing a large number of jobs across multiple servers multiple times a day.

On the Jenkins master install SSH Plugin as we need to establish the remote connection with the slave machine.



So we are going to create a slave node:

* > Manage Jenkins (scroll down) → Manage Nodes → Master (Now master is always going to be included by default )

# Manage Jenkins (scroll down) → Manage Nodes → New Node → Node name : Remote slave 1 → # of executors : 1 → Remote root Directory: (/home/centos) Labels: n1 → Usage: (as much as possible) → Launch method: Launch slave via SSH → Host: private ip of slave → Credentials: Service acc(Give settings of Kind: SSH username with private key && Private key: Copy the content of pem key used for slave server) → Availability: Keep online as much as possible → Save

Then click on node and see the log.

Install java and javac on all slaves.

Now if i goto Jenkins home you can see both master and slave.

Building JOB On Node

New job → Under General (✅ Restrict where this project can run)(Label expression: n1 or expression we gave while creating slave) → Build → Execute Shell → in command give # pwd # uname -a # hostname → Save → Run build.

Now you can see the job being executed on node1

Now execute the maven job on node1 too.

