JENKINS

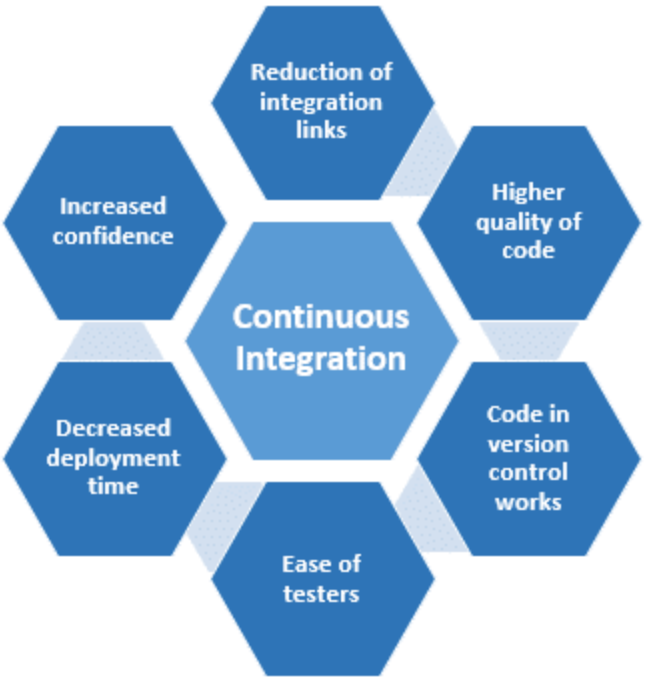
Jenkins is a powerful application that allows continuous integration and continuous delivery of projects, regardless of the platform you are working on. It is a free source that can handle any kind of build or continuous integration. You can integrate Jenkins with a number of testing and deployment technologies.

What is Continuous Integration?

Continuous integration (CI) happens to be one of the most vital parts of DevOps. It is primarily used to integrate various stages of DevOps together. In other words, it is a coding practice that essentially enables the development team to make and implement small changes in the code and version control methods quite frequently.
Continuous integration is usually done in the form where all developers push the code onto a shared repository mostly multiple times a day. It is pretty fit for a project that should be coded and developed on different platforms with multiple tools. Currently, it has become important to have such a mechanism in place to integrate and validate the changes made to the code in a parallel way.

Why use Continuous Integration?

What exactly are the benefits of continuous integration? Why do we adopt this practice? To help answer these questions, here is the list of some of the advantages of CI.



- **Reduction of integration links**: All projects employ more than one person to develop and it greatly increases the risk of errors during integration. Depending on the complexity of the code, it is possible that a lot of changes would have to be made. Here comes CI to the rescue and helps alleviate the issues as it allows for regular integration.
- **Higher quality of code**: As the risks drastically reduce, a lot of the time and manpower can be diverted to creating a much more functionality-oriented code.
- **Code in version control works**: Committing something that breaks the build immediately triggers a notification thereby preventing anyone from pulling a broken code.
- **Ease of testers**: Retaining the different versions and builds of the code eases the work of QAs to understand, locate, and trace bugs efficiently.
- **Decreased deployment time**: Automating the process of deployment eases and frees up a lot of time and manpower.
- **Increased confidence**: The absence of a possible failure or breakdown gives developers peace of mind and thereby helps in delivering greater productivity and higher quality products.

What is Jenkins? How is Jenkins used for Continuous Integration?

Jenkins is an automation tool written in Java with built-in plugins for continuous integration tasks. It is used to continuously build and test projects making it easier to integrate the changing codes to it.

Jenkins allows for faster delivery of software by working with a large number of deployment and testing technologies. It also accelerates the development phase via the automation of tasks. It is primarily a server-based app and requires a web server like Tomcat.

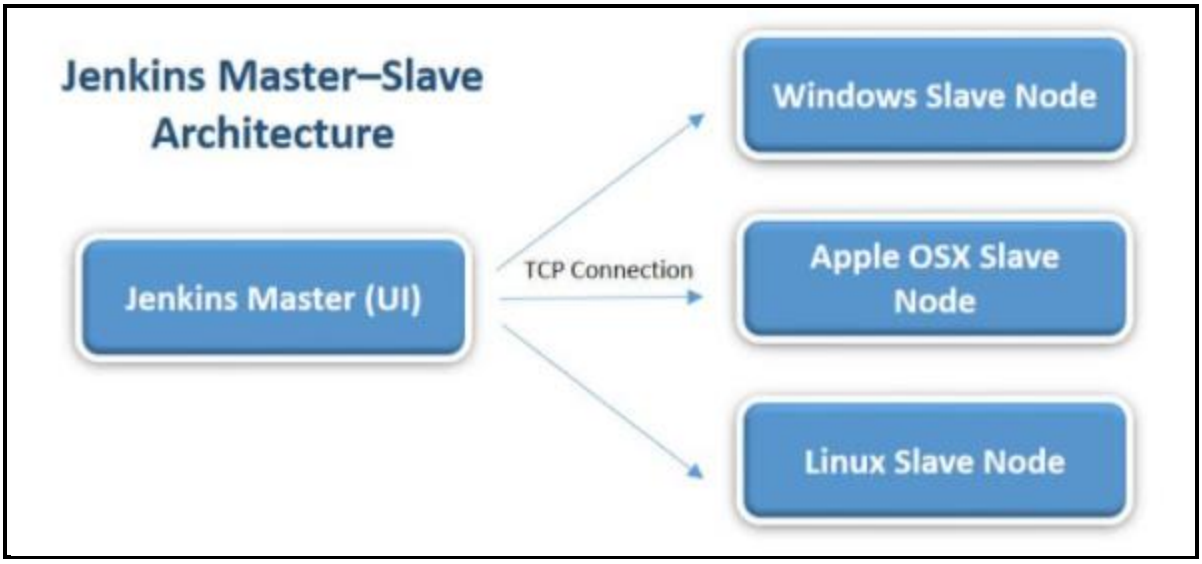
Jenkins rose to fame because of its monitoring of repeated tasks. If a team is developing a project, then Jenkins will constantly check and evaluate the code thereby returning any possible error/failure early in the development phase.

How does Jenkins work?

Standalone Jenkins instances can be an intensive disk and CPU resource-eating process. To avoid this, we can scale it by implementing slave nodes which essentially would help us offload a part of the master node's responsibilities.

A slave is just a device that is configured to act as an executor on behalf of the master. The master is the base installation of the Jenkins tool and does the basic operations and serves the user interface while the slaves do the actual work.

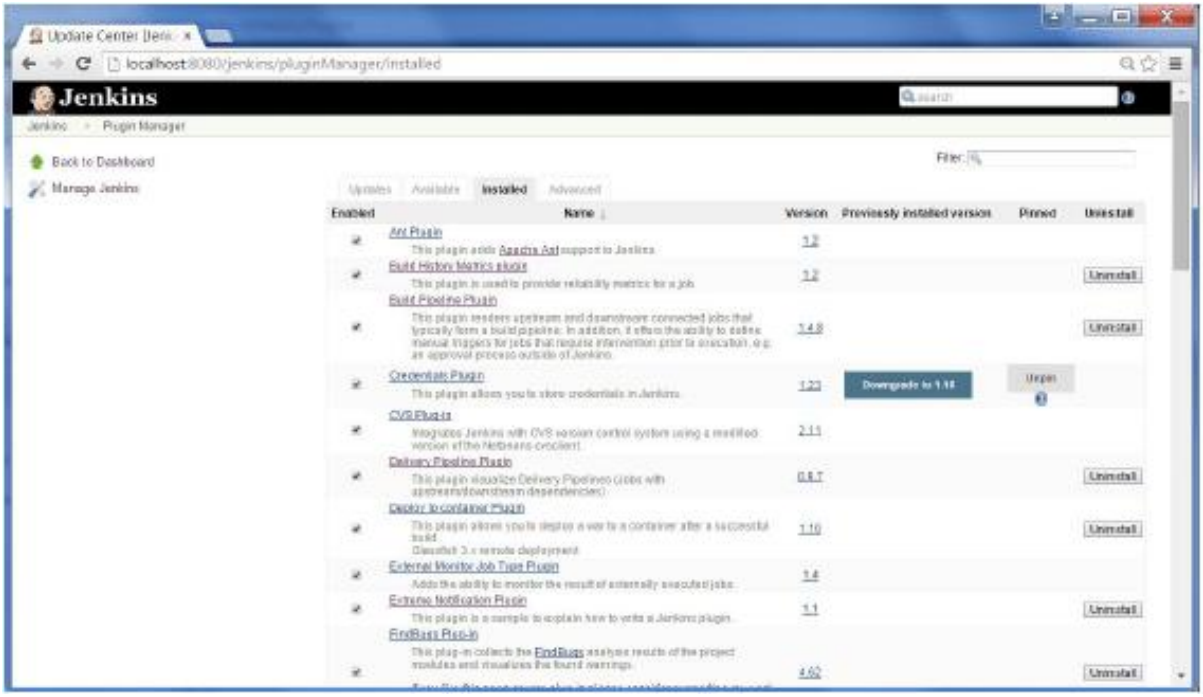
In the below image, the Jenkins master is in charge of the UI and the slave nodes are of different OS types.



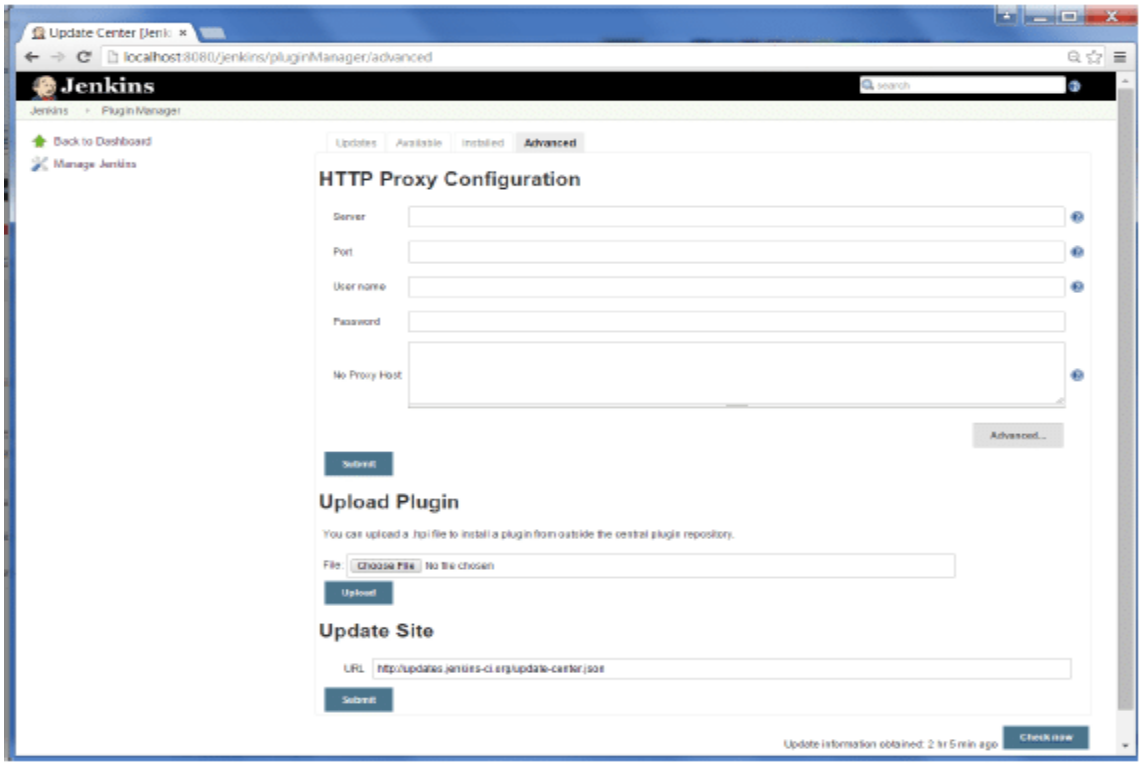
Installing Jenkins Plugins

One of the core features of Jenkins is the integration of Jenkins plugins. They help add functionalities over the core to give us more powerful tools with regard to the project. Now, let's look into how we can list, add, modify, update, and remove these plugins from Jenkins.

- To list all the plugins supported by Jenkins, go to https://wiki.jenkins-ci.org/display/JENKINS/Plugins.
- Once logged in, head over to the 'Manage Jenkins' tab on the left-hand side. This is where we would handle all the installed plugins, as well as add or remove new ones.
- Under the Manage Plugins tab, we can search for a plugin or see all the available plugins.
- By selecting a plugin and clicking on Install without restart, we can install the plugin and check its functionality sooner, rather than having to wait to restart Jenkins.
- When we need to uninstall a plugin, head over to the Installed tab, select the plugin that we would like to remove and click on Uninstall. However, we must make sure to restart Jenkins for the changes to reflect



- In some cases, we would like to use an older version of a certain plugin. In such a situation, we have to download the needed plugin from the desired site and then upload it onto Jenkins manually.
- If we have created our own plugins, we have to upload them to the site and help further grow the community base.



Creating Jenkins Builds

A build is often called when the source code is converted into a usable and runnable form. It allows compiling the code into an executable form. The process of building is typically handled by the build tool.
Builds are usually done when we reach a critical standpoint such as the integration of a feature or so on. As Jenkins is CI-based, we have a powerful feature where we can automate the build process to happen at a particular time or event. This is called as 'scheduled builds.'

Creating Scheduled Builds

1. Build Periodically

Now in this Jenkins tutorial, we will find out how we can schedule builds at certain times and triggers. To schedule a build, follow the below steps:

Step 1: In the 'Build Triggers' section, check the 'Build periodically' box

Step 2: In the text box, enter the scheduling parameters such as date, day, and time

The general syntax is MINUTE (0-59), HOUR (0-23), DAY (1-31), MONTH (1-12), DAY OF THE WEEK (0-7)

Now, let's look at some examples of how to schedule builds.

- If we would want to start a build every day at 8:30 from Monday to Friday, then we would have to give the parameters like **30 08 * * 1-6**.
- To start building daily in the afternoon from 4 to 4:59 pm depending on the projects hash, we would have to give the parameters like **H 16 * * 1-5**.
- If we would like it to start at midnight, then we would have to give the parameters like **@midnight** or **59 23 * * 6**.
- To build every hour, we would have to give it as **H * * * ***

2. Poll SCM

"Poll SCM" polls the SCM periodically for checking if any changes/ new commits were made and shall build the project if any new commits were pushed since the last build.

1. Click on the "Configure" of the job created in the Jenkins dashboard.
2. Click on build triggers in the configure settings and select the Poll SCM.
3. Enter the desired cron to poll the SCM. Here we have given * * * * which means the Jenkins polls the SCM every minute.

Jenkins Installation

Pre-Requisite for Jenkins installation:

Step 1. Install Java

01- Java Development Kit (JDK) is required to install Apache Maven. Use the following command to install the OpenJDK package:

```
# yum install java-1.8.0-openjdk
```

– To set path variable

```
vi  /etc/profile.d/java.sh
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
export PATH=${JAVA_HOME}/bin :${PATH}
```

```
source /etc/profile.d/java.sh
```

– To verify that Java was successfully installed, run the following command:

```
# java -version
openjdk version "1.8.0_161"
OpenJDK Runtime Environment (build 1.8.0_161-b14)
OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)
```

Step 2: Create Jenkins User and Provide root Privileges

##Create User jenkins
#useradd jenkins
#passwd jenkins
#visudo
add below under root ALL=(ALL:ALL) ALL
Jenkins ALL=(ALL:ALL) ALL

Step 3: Switch to Jenkins user and Verify login.

#sudo su – Jenkins

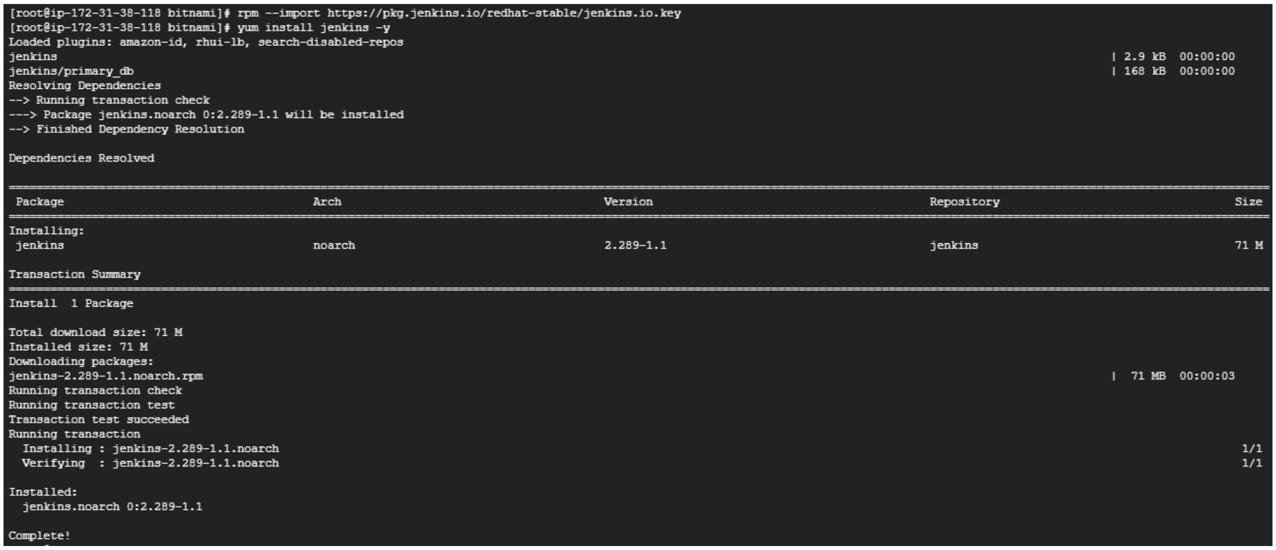
Step 4: Download the repo using below command:

```
wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
```

```
[root@ip-172-31-38-118 bitnami]# wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo
--2021-04-26 11:46:30--  http://pkg.jenkins-ci.org/redhat/jenkins.repo
Resolving pkg.jenkins-ci.org (pkg.jenkins-ci.org)... 52.202.51.185
Connecting to pkg.jenkins-ci.org (pkg.jenkins-ci.org)|52.202.51.185|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 71
Saving to: '/etc/yum.repos.d/jenkins.repo'

100%[===================================================================================================================>] 71

2021-04-26 11:46:30 (13.7 MB/s) - '/etc/yum.repos.d/jenkins.repo' saved [71/71]
```

Step 5: Import rpm key and install Jenkins using yum:

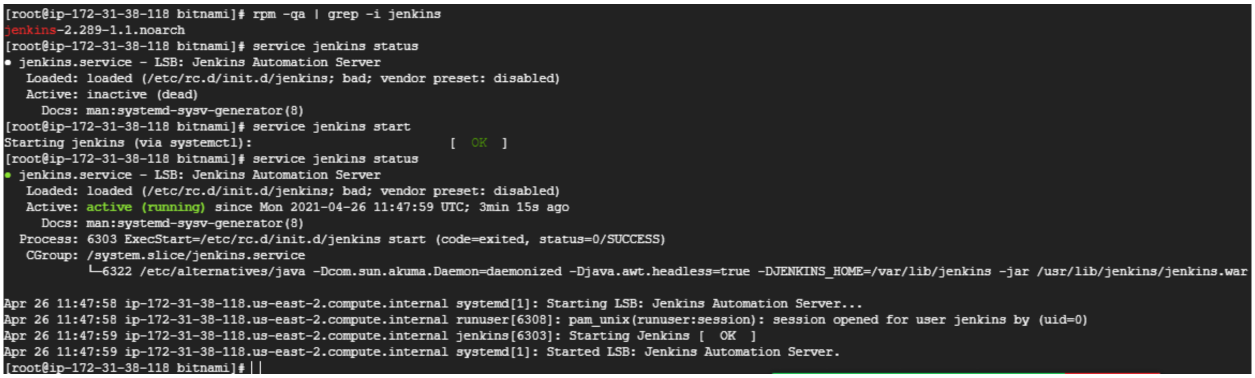
```
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum install jenkins -y
```



```
[root@ip-172-31-38-118 bitnami]# rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
[root@ip-172-31-38-118 bitnami]# yum install jenkins -y
Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
jenkins                                                                                                                              | 2.9 kB  00:00:00
jenkins/primary_db                                                                                                                   | 168 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package jenkins.noarch 0:2.289-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================================================================================
 Package                                 Arch                                    Version                                     Repository              Size
==========================================================================================================================================================
Installing:
 jenkins                                 noarch                                  2.289-1.1                                   jenkins                 71 M

Transaction Summary
==========================================================================================================================================================
Install  1 Package

Total download size: 71 M
Installed size: 71 M
Downloading packages:
jenkins-2.289-1.1.noarch.rpm                                                                                                         |  71 MB  00:00:03
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.289-1.1.noarch                                                                                                                1/1
  Verifying  : jenkins-2.289-1.1.noarch                                                                                                                1/1

Installed:
  jenkins.noarch 0:2.289-1.1

Complete!
```

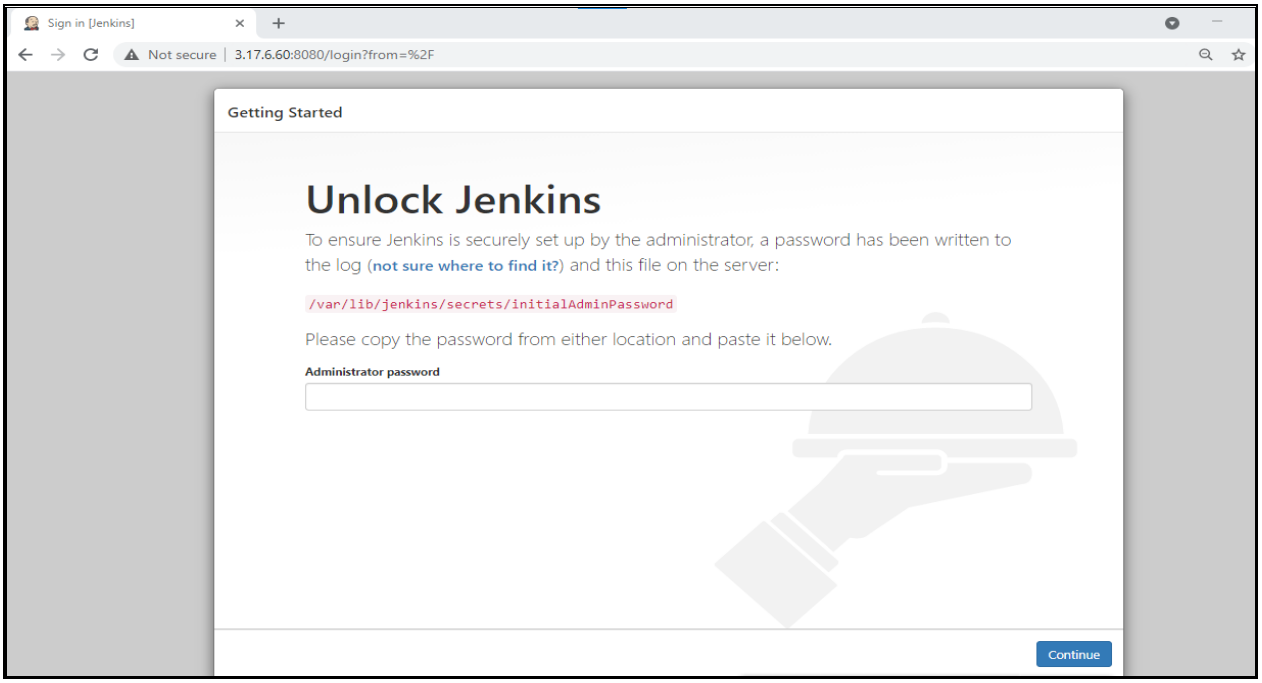
Step 6: Verify Jenkins Package and start Jenkins

```
sudo rpm -qa | grep -i jenkins
sudo service jenkins start
```



```
[root@ip-172-31-38-118 bitnami]# rpm -qa | grep -i jenkins
jenkins-2.289-1.1.noarch
[root@ip-172-31-38-118 bitnami]# service jenkins status
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:systemd-sysv-generator(8)
[root@ip-172-31-38-118 bitnami]# service jenkins start
Starting jenkins (via systemctl):                          [  OK  ]
[root@ip-172-31-38-118 bitnami]# service jenkins status
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: active (running) since Mon 2021-04-26 11:47:59 UTC; 3min 15s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 6303 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/jenkins.service
           └─6322 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemonized -Djava.awt.headless=true -DJENKINS_HOME=/var/lib/jenkins -jar /usr/lib/jenkins/jenkins.war

Apr 26 11:47:58 ip-172-31-38-118.us-east-2.compute.internal systemd[1]: Starting LSB: Jenkins Automation Server...
Apr 26 11:47:58 ip-172-31-38-118.us-east-2.compute.internal runuser[6308]: pam_unix(runuser:session): session opened for user jenkins by (uid=0)
Apr 26 11:47:59 ip-172-31-38-118.us-east-2.compute.internal jenkins[6303]: Starting Jenkins [  OK  ]
Apr 26 11:47:59 ip-172-31-38-118.us-east-2.compute.internal systemd[1]: Started LSB: Jenkins Automation Server.
[root@ip-172-31-38-118 bitnami]# ||
```

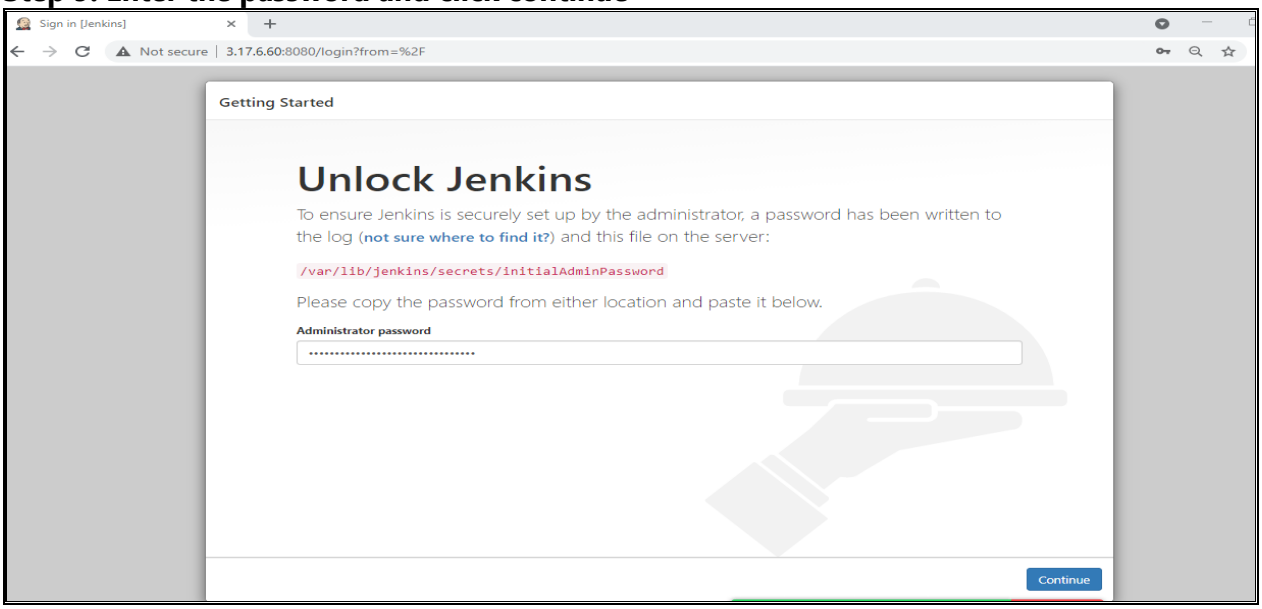
Step 7: Open URL http://IP_ADDRESS:8080



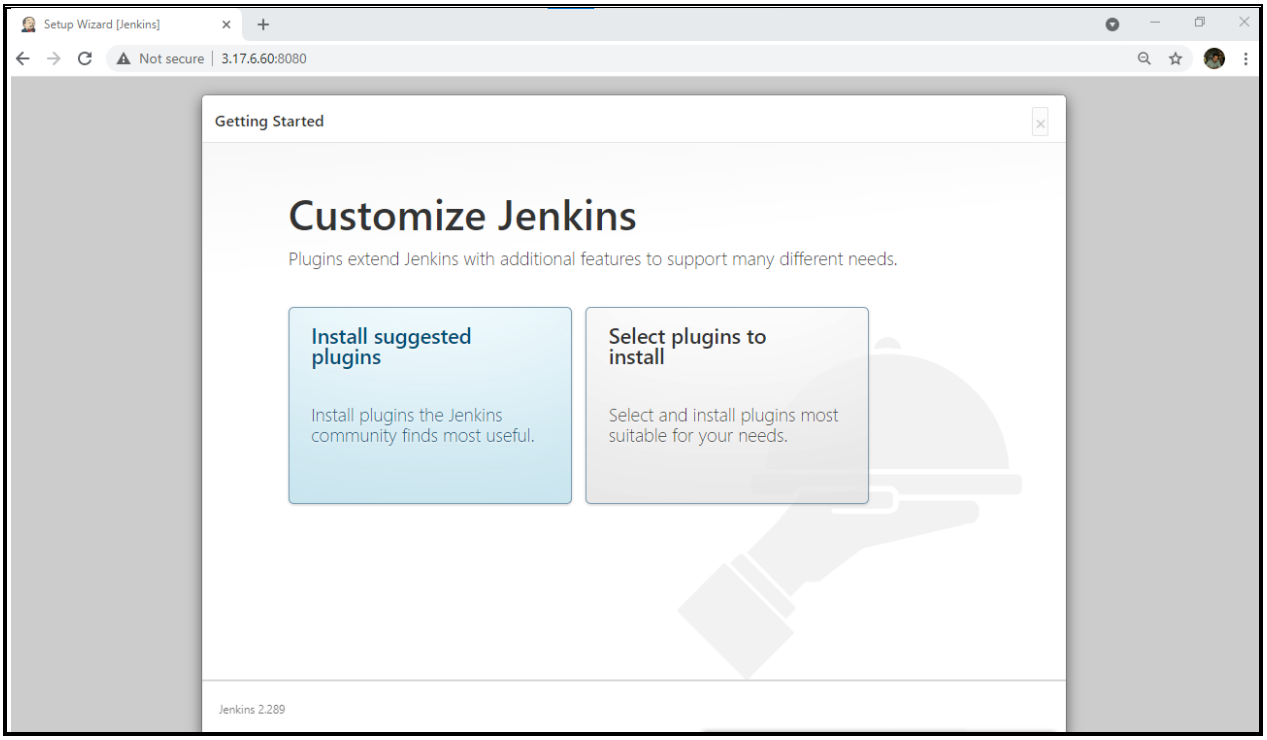
Step 8: Login into server and check for password:

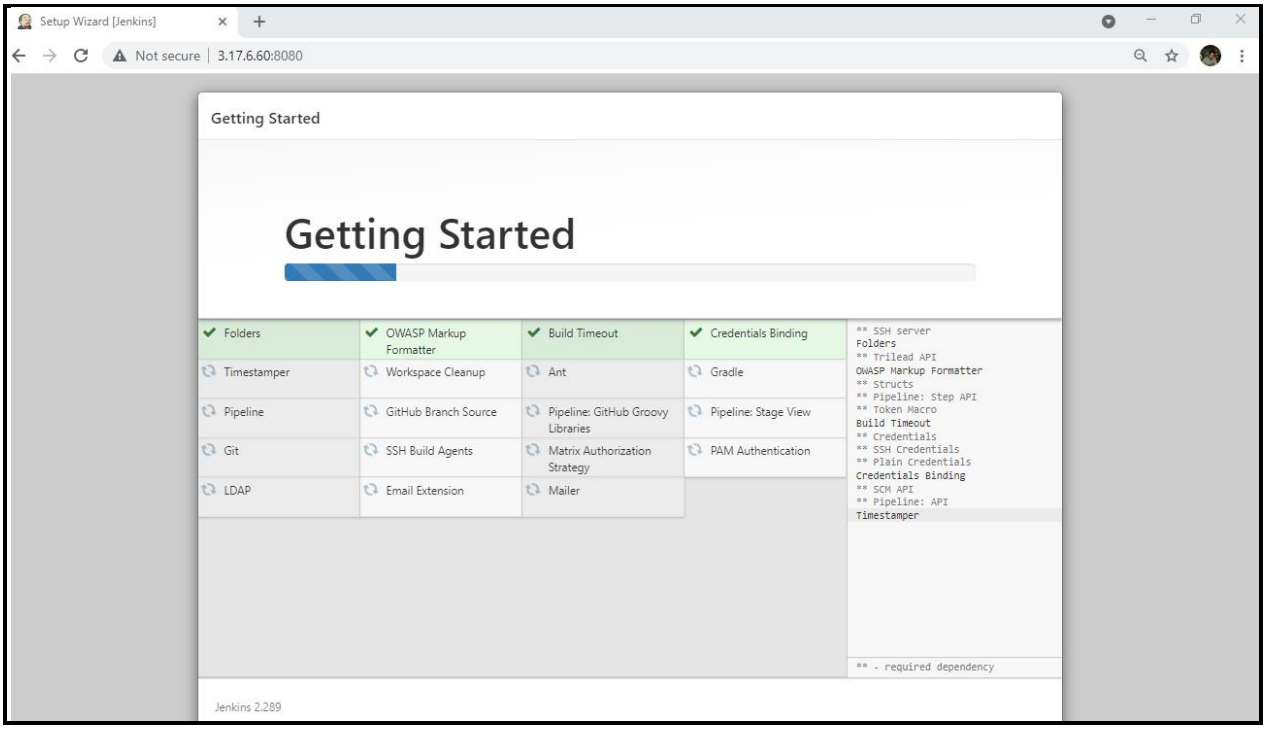
```
[root@ip-172-31-38-118 jenkins]# cat /var/lib/jenkins/secrets/initialAdminPassword
b704b37805964fd88ad2d43a78370ba4
[root@ip-172-31-38-118 jenkins]#
```

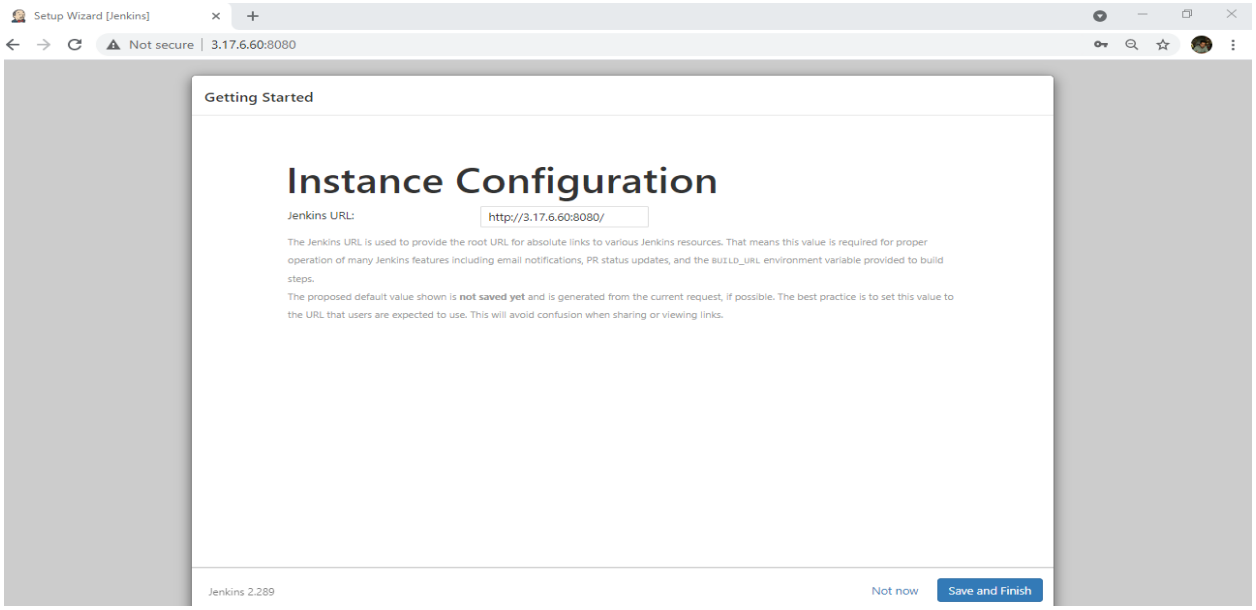
Step 9: Enter the password and click continue



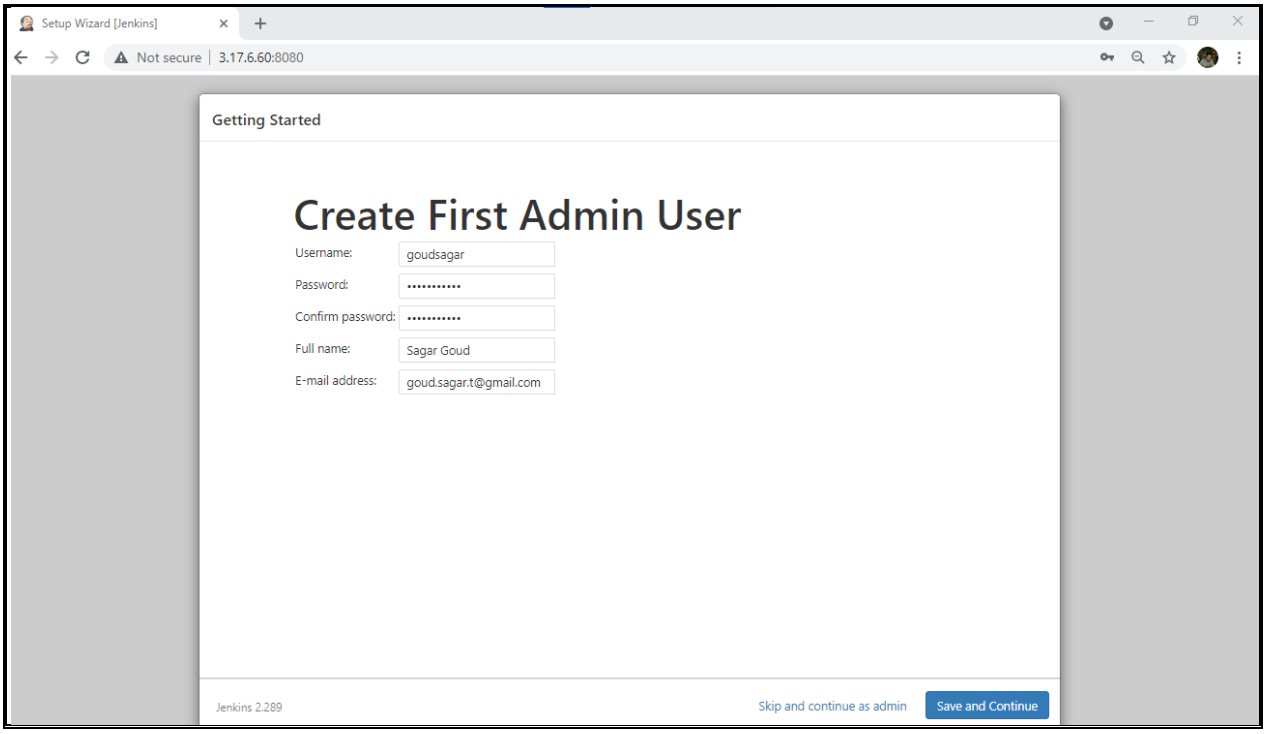
Step 10: Install Packages based on below selection

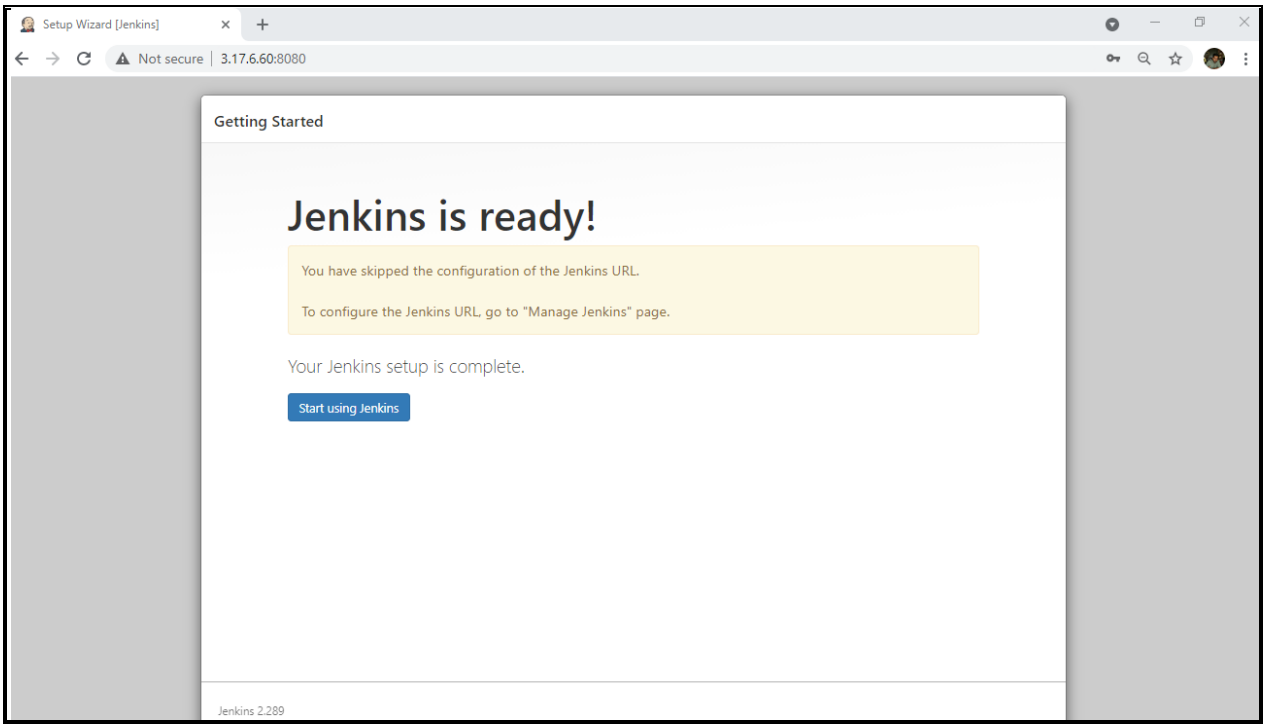




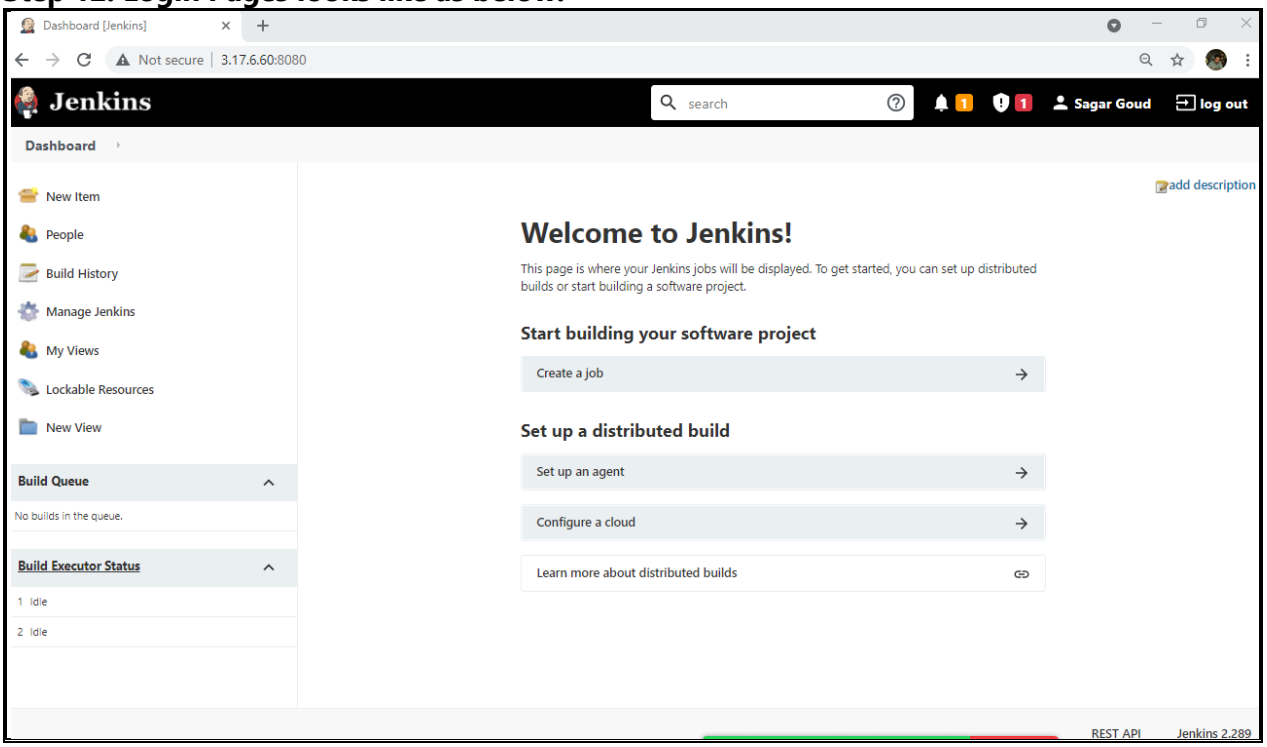


Step 11: Create an Admin User





Step 12: Login Pages looks like as below:



Step 13: For checking Jenkins logs /var/log/jenkins/jenkins.log

```
[root@ip-172-31-38-118 jenkins]# tail -100f jenkins.log
Running from: /usr/lib/jenkins/jenkins.war
2021-04-26 11:48:00.334+0000 [id=1]     WARNING winstone.Logger#logInternal: Parameter handlerCountMax is now deprecated
2021-04-26 11:48:00.365+0000 [id=1]     WARNING winstone.Logger#logInternal: Parameter handlerCountMaxIdle is now deprecated
2021-04-26 11:48:00.383+0000 [id=1]     INFO    org.eclipse.jetty.util.log.Log#initialized: Logging initialized @1025ms to org.eclipse.jetty.util.log.JavaUtilLog
2021-04-26 11:48:00.464+0000 [id=1]     INFO    winstone.Logger#logInternal: Beginning extraction from war file
2021-04-26 11:48:02.309+0000 [id=1]     WARNING o.e.j.s.handler.ContextHandler#setContextPath: Empty contextPath
2021-04-26 11:48:02.405+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: jetty-9.4.39.v20210325; built: 2021-03-25T14:42:11.471Z; git: 9fc7ca5a922f2a37b84ec9dbc26a516
8cee7e667; jvm 1.8.0_292-b10
2021-04-26 11:48:02.939+0000 [id=1]     INFO    o.e.j.w.StandardDescriptorProcessor#visitServlet: NO JSP Support for /, did not find org.eclipse.jetty.jsp.JettyJspServlet
2021-04-26 11:48:03.072+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: DefaultSessionIdManager workerName=node0
2021-04-26 11:48:03.072+0000 [id=1]     INFO    o.e.j.s.s.DefaultSessionIdManager#doStart: No SessionScavenger set, using defaults
2021-04-26 11:48:03.073+0000 [id=1]     INFO    o.e.j.server.session.HouseKeeper#startScavenging: node0 Scavenging every 600000ms
2021-04-26 11:48:03.939+0000 [id=1]     INFO    hudson.WebAppMain#contextInitialized: Jenkins home directory: /var/lib/jenkins found at: SystemProperties.getProperty("JENKINS_HOME")
2021-04-26 11:48:04.194+0000 [id=1]     INFO    o.e.j.s.handler.ContextHandler#doStart: Started w.@3153ddfc{Jenkins v2.289,/,file:///var/cache/jenkins/war/,AVAILABLE}{/var/cache/jenk
ins/war}
2021-04-26 11:48:04.229+0000 [id=1]     INFO    o.e.j.server.AbstractConnector#doStart: Started ServerConnector@776a6d9b{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
2021-04-26 11:48:04.229+0000 [id=1]     INFO    org.eclipse.jetty.server.Server#doStart: Started @4871ms
2021-04-26 11:48:04.267+0000 [id=22]    INFO    winstone.Logger#logInternal: Winstone Servlet Engine running: controlPort=disabled
2021-04-26 11:48:06.149+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: Started initialization
2021-04-26 11:48:06.196+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Listed all plugins
2021-04-26 11:48:08.570+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Prepared all plugins
2021-04-26 11:48:08.575+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Started all plugins
2021-04-26 11:48:08.624+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: Augmented all extensions
2021-04-26 11:48:10.568+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: System config loaded
2021-04-26 11:48:10.568+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2021-04-26 11:48:10.568+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2021-04-26 11:48:10.568+0000 [id=28]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updated
2021-04-26 11:48:10.653+0000 [id=41]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Started Download metadata
2021-04-26 11:48:10.676+0000 [id=41]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updates server
2021-04-26 11:48:11.659+0000 [id=27]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
```

```
b704b37805964fd88ad2d43a78370ba4

This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

2021-04-26 11:48:25.798+0000 [id=27]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2021-04-26 11:48:25.846+0000 [id=21]    INFO    hudson.WebAppMain$3#run: Jenkins is fully up and running
2021-04-26 11:48:25.954+0000 [id=41]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2021-04-26 11:48:25.954+0000 [id=41]    INFO    hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
2021-04-26 11:48:25.956+0000 [id=41]    INFO    hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Download metadata. 15,301 ms
```

Master Slave Integration

On the Agent machine:

1- Install the necessary packages

You will need to install some packages on the agent node, such as Java, use the below command to install the `openjdk`:

```
# sudo yum install java-1.8.0-openjdk
```

2- Create a user on the agent to be used by Jenkins

Now we need to create a user on the agent. The Jenkins master will log into the agent as this user, and all build jobs will execute as this user. The new user will be called `jenkins` with `/var/lib/jenkins` as home directory:

```
# sudo useradd -d /var/lib/jenkins jenkins

# passwd jenkins
```

3- Generate an ssh key

Next, we need to generate an ssh key. Jenkins will use this key to authenticate with the agent node and log in as the jenkins user. This key can be generated on practically any Linux machine, but you can also do it on the agent node itself and copy it to the new agents nodes:

```
# su - jenkins
# ssh-keygen -t rsa -C "Jenkins agent key"
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa.
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:5qJWiPnvv+Gozm8iP+Ered03HgLJW2eyW0tzA5r1YYU jenkins ssh slaves
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|              .  |
|            E .  |
|     . .     .   |
|   o .+ S = o    |
|  o o .* O + .   |
|   + +o.B = +    |
|  +.B.o+.*o= .   |
|   BBX=.=+o.     |
+----[SHA256]-----+
```

– Add the public SSH key id_rsa.pub to the list of authorized_keys file like below:

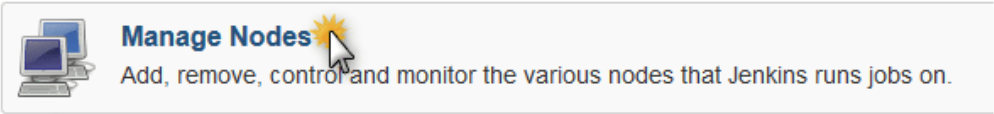
```
# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

# chmod 600 ~/.ssh/authorized_keys
```

– Copy the private SSH key ~/.ssh/id_rsa from the agent machine to your OS clipboard. The SSH private key should be similiar to this

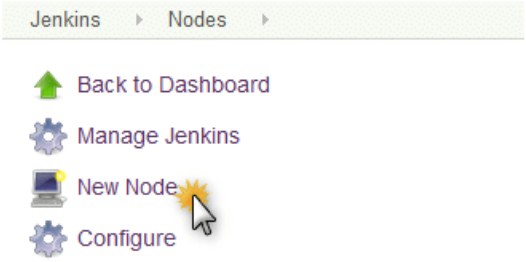
```
# cat ~/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
.................................
.................................
.................................
-----END RSA PRIVATE KEY-----
```

In Jenkins Server:

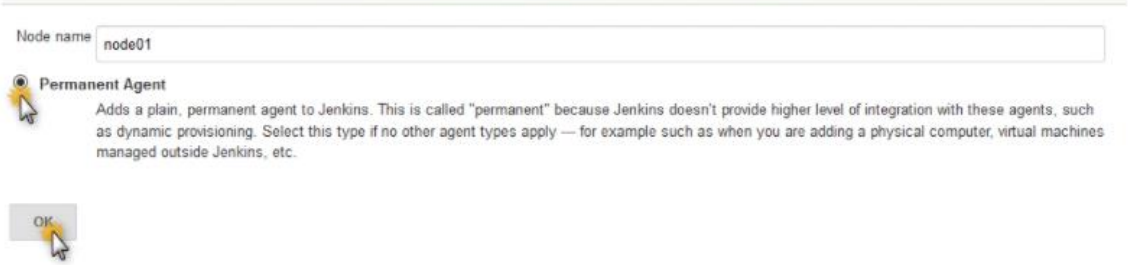
1- Go to `Manage Jenkins` -> `Manage Nodes`:



2- Then click on the `New Node` button:

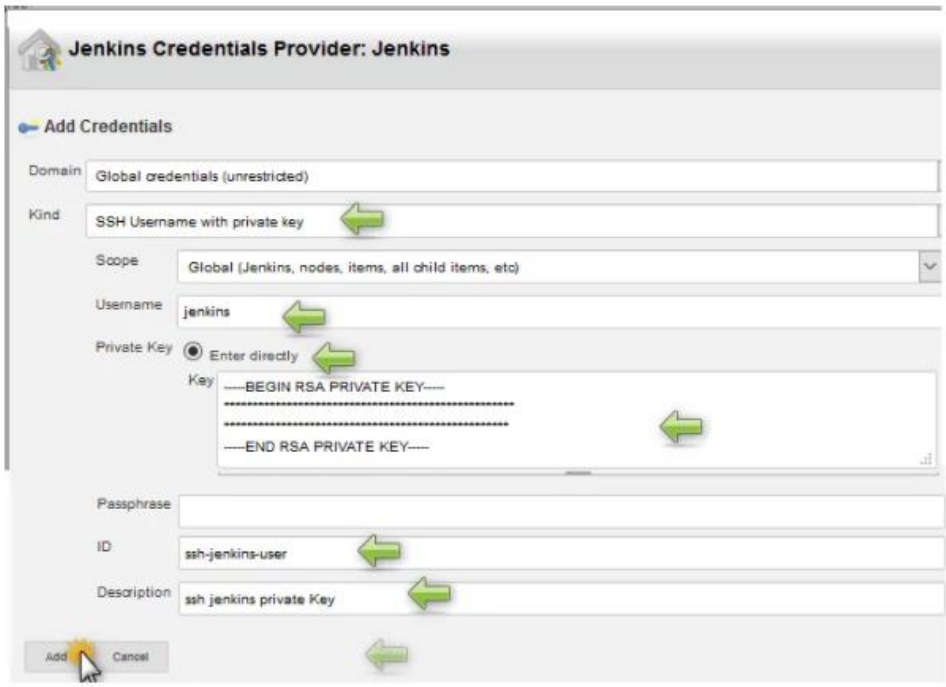


3- Configure the name of the agent, select `Permanent agent` and click on the `Ok` button:



4- After creating the new node, you have to configure the node settings. Fill in the `Remote root directory` with a path the user on the agent is allowed to write to, set the `Host` value with the hostname of the agent, and press the `Add` button for `Credentials` :

5- Choose `SSH Username with private key` option, fill the `Username` value with the user account on the agent machine, in our example is `jenkins` , and choose `Private Key` -> `Enter directly` and paste the key from your OS clipboard, and give an `ID` and a useful `Description` for this credential. Finally click the `add` button.



6- Select the `Manually trusted key Verification Strategy` value of the `Host Key Verification Strategy` menu and click `save` button.

7- Your new node should now appear in the list of nodes. You may notice a red X on the node's icon. This indicates that it is not connected yet. Wait a few seconds and refresh the page, and the red X will go away, indicating that the node is successfully connected.

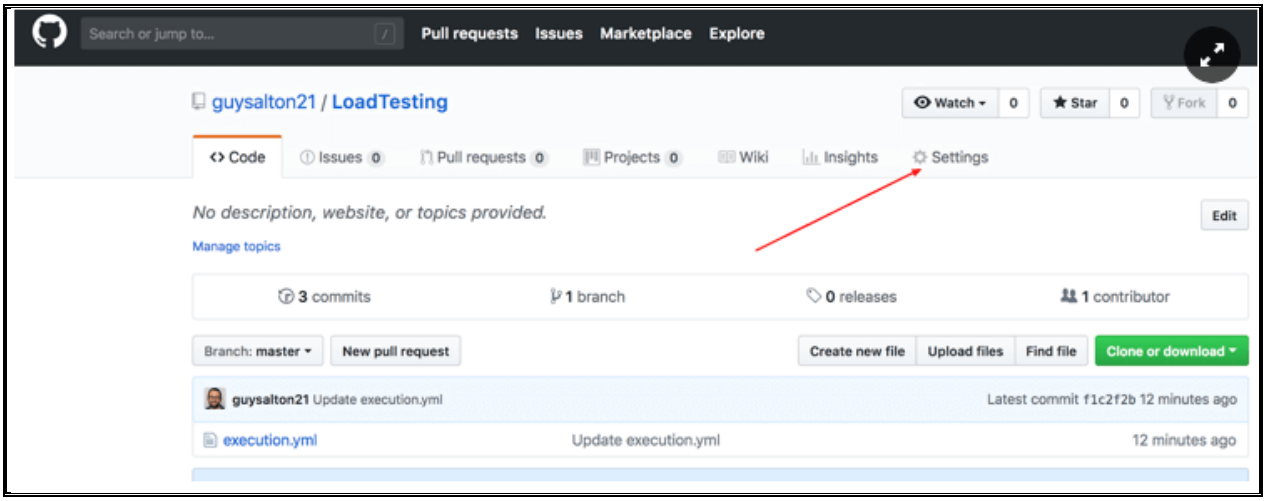
S	Name ↓	Architecture	Clock Difference
	master	Linux (amd64)	In sync
	node01	Linux (amd64)	In sync

Webhook Integration with Jenkins

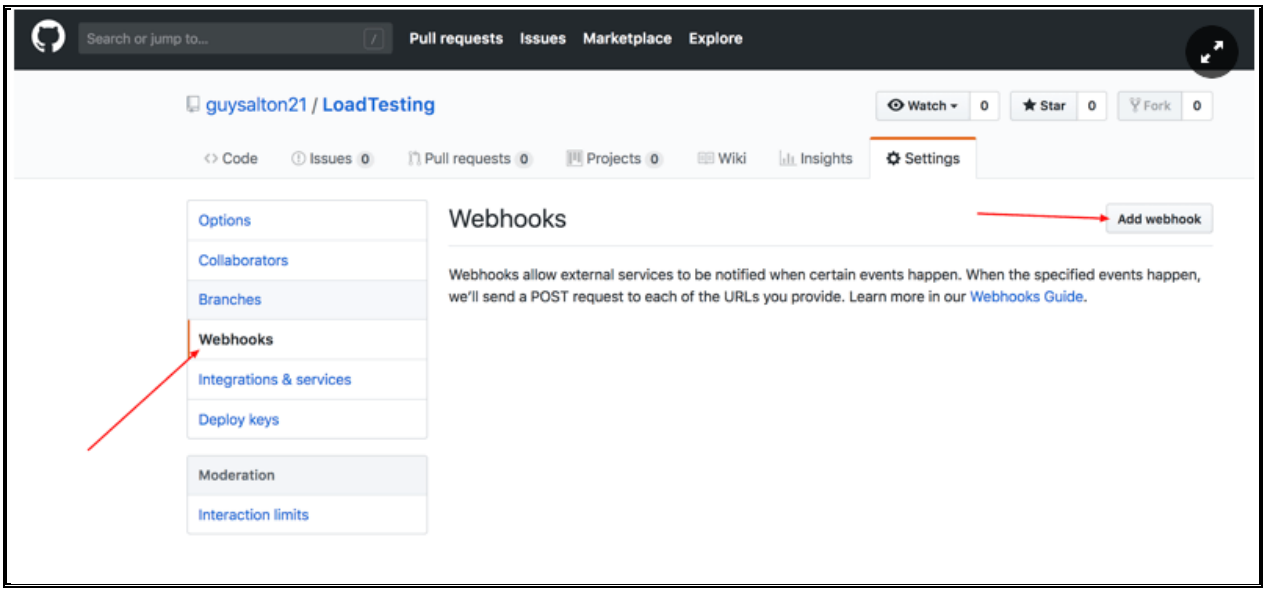
- Schedule your build
- Pull your code and data files from your GitHub repository to your Jenkins machine
- Automatically trigger each build on the Jenkins server, after each Commit on your Git repository

Configuring GitHub

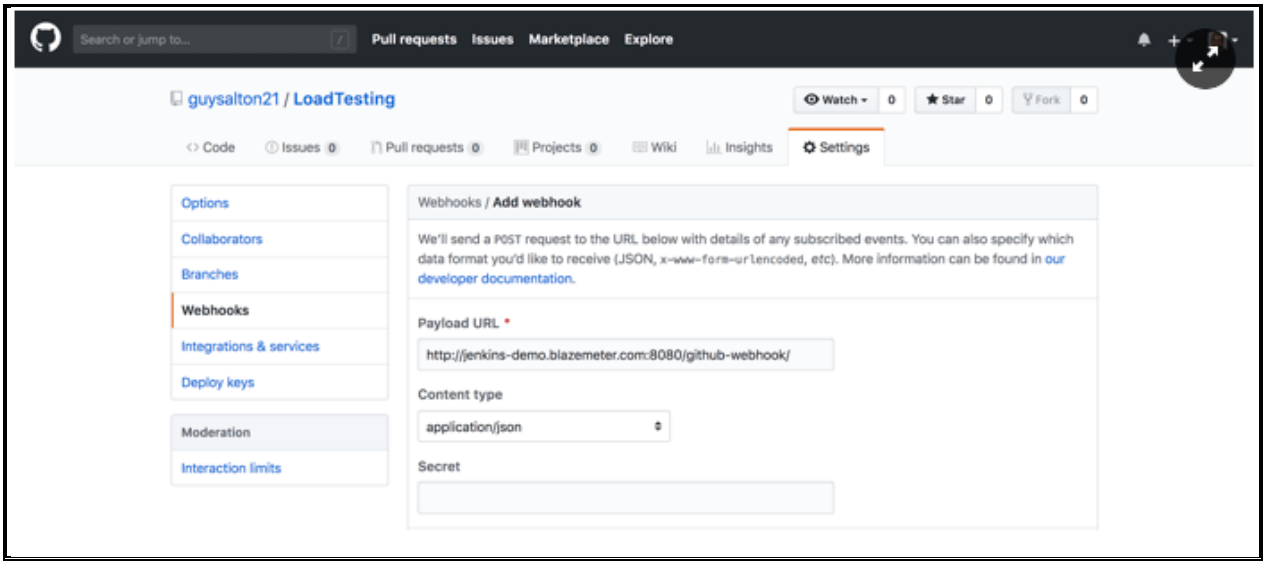
Step 1: go to your GitHub repository and click on **'Settings'.**



Step 2: Click on **Webhooks** and then click on **'Add webhook'.**



Step 3: In the **'Payload URL'** field, paste your Jenkins environment URL. At the end of this URL add */github-webhook/*. In the **'Content type'** select: *'application/json'* and leave the **'Secret'** field empty.



Step 4: In the page **'Which events would you like to trigger this webhook?'** choose *'Let me select individual events.'* Then, check *'Pull Requests'* and *'Pushes'*. At the end of this option, make sure that the *'Active'* option is checked and click on *'Add webhook'*.

Which events would you like to trigger this webhook?

- Just the push event.
- Send me **everything**.
- Let me select individual events.

- [ ] **Check runs**
  Check run is created, requested, rerequested, or completed.
- [ ] **Check suites**
  Check suite is requested, rerequested, or completed.
- [ ] **Commit comments**
  Commit or diff commented on.
- [ ] **Branch or tag creation**
  Branch or tag created.
- [ ] **Branch or tag deletion**
  Branch or tag deleted.
- [ ] **Deployments**
  Repository deployed.
- [ ] **Deployment statuses**
  Deployment status updated from the API.
- [ ] **Forks**
  Repository forked.
- [ ] **Wiki**
  Wiki page updated.
- [ ] **Issue comments**
  Issue comment created, edited, or deleted.
- [ ] **Issues**
  Issue opened, edited, deleted, transferred, closed, reopened, assigned, unassigned, labeled, unlabeled, milestoned, or demilestoned.
- [ ] **Labels**
  Label created, edited or deleted.
- [ ] **Collaborator add, remove, or changed**
  Collaborator added to, removed from, or has changed permissions for a repository.
- [ ] **Milestones**
  Milestone created, closed, opened, edited, or deleted.

- [ ] **Visibility changes**
  Repository changes from private to public.
- [x] **Pull requests**
  Pull request opened, closed, reopened, edited, assigned, unassigned, review requested, review request removed, labeled, unlabeled, or synchronized.
- [ ] **Pull request reviews**
  Pull request review submitted, edited, or dismissed.
- [ ] **Pull request review comments**
  Pull request diff comment created, edited, or deleted.
- [x] **Pushes**
  Git push to a repository.
- [ ] **Releases**
  Release published in a repository.
- [ ] **Repositories**
  Repository created, deleted, archived, unarchived, publicized, or privatized.
- [ ] **Repository imports**
  Repository import succeeded, failed, or cancelled.
- [ ] **Repository vulnerability alerts**
  Vulnerability alert created, resolved, or dismissed on a repository.
- [ ] **Statuses**
  Commit status updated from the API.
- [ ] **Team adds**
  Team added or modified on a repository.
- [ ] **Watches**
  User stars a repository.

- [x] **Active**
  We will deliver event details when this hook is triggered.

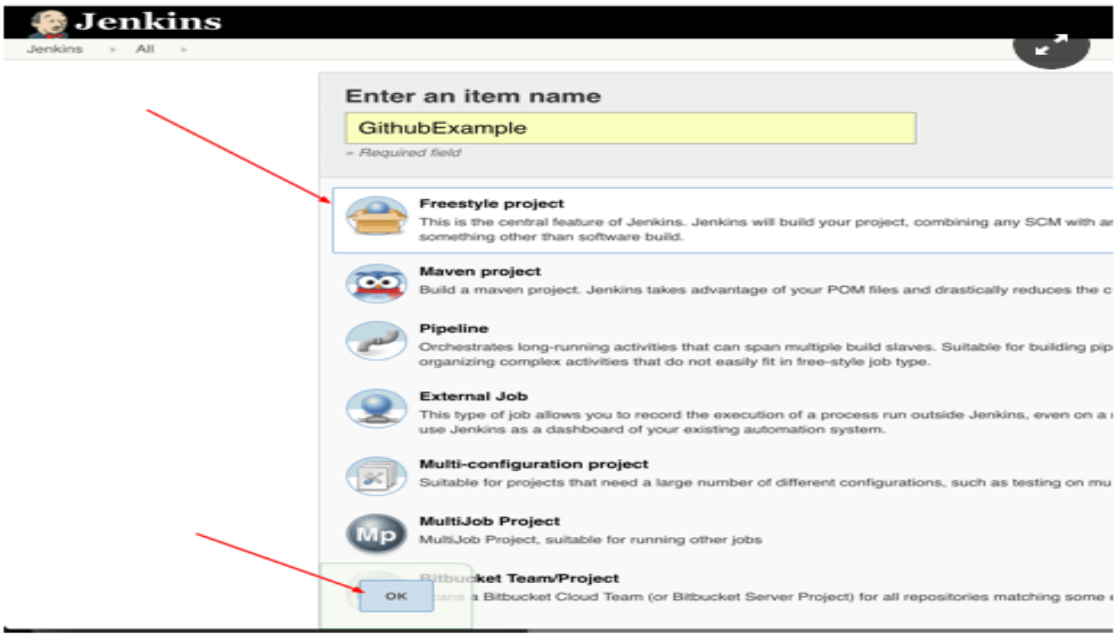
Add webhook

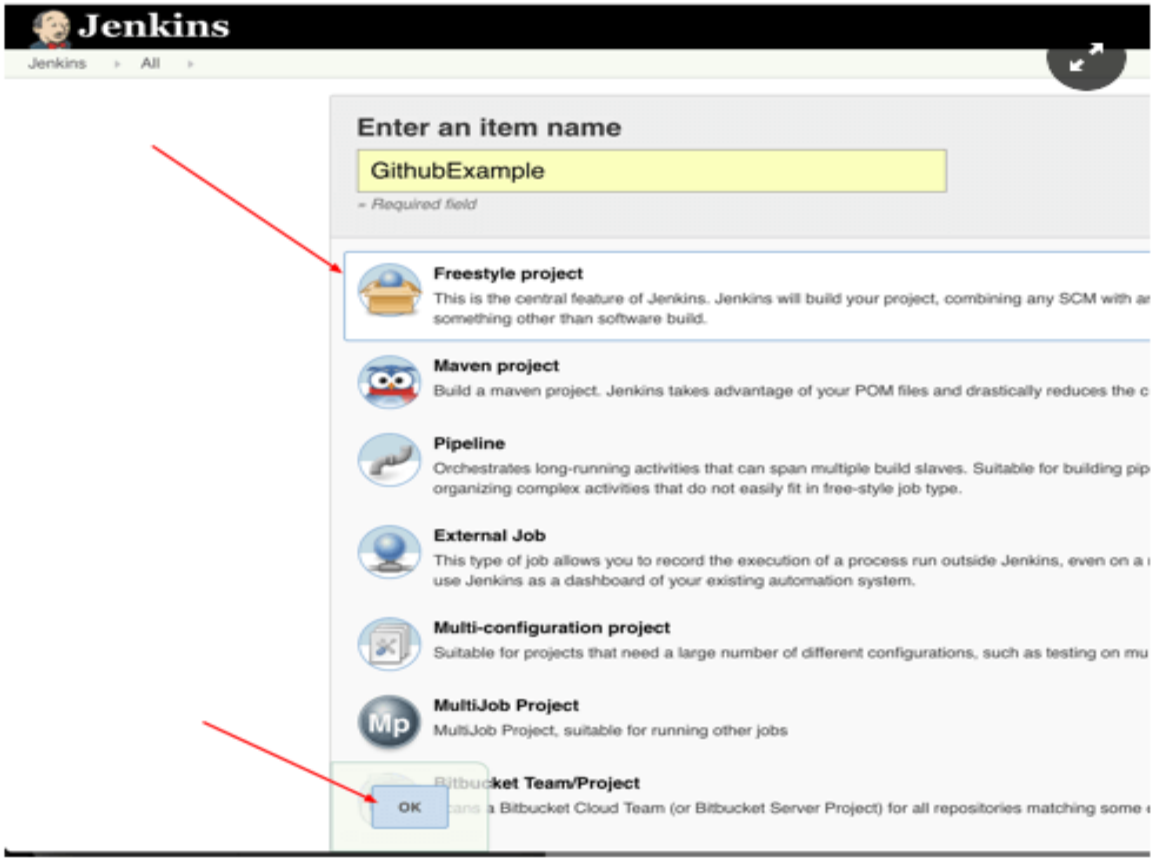
We're done with the configuration on GitHub's side! Now let's move on to Jenkins.

Configuring Jenkins

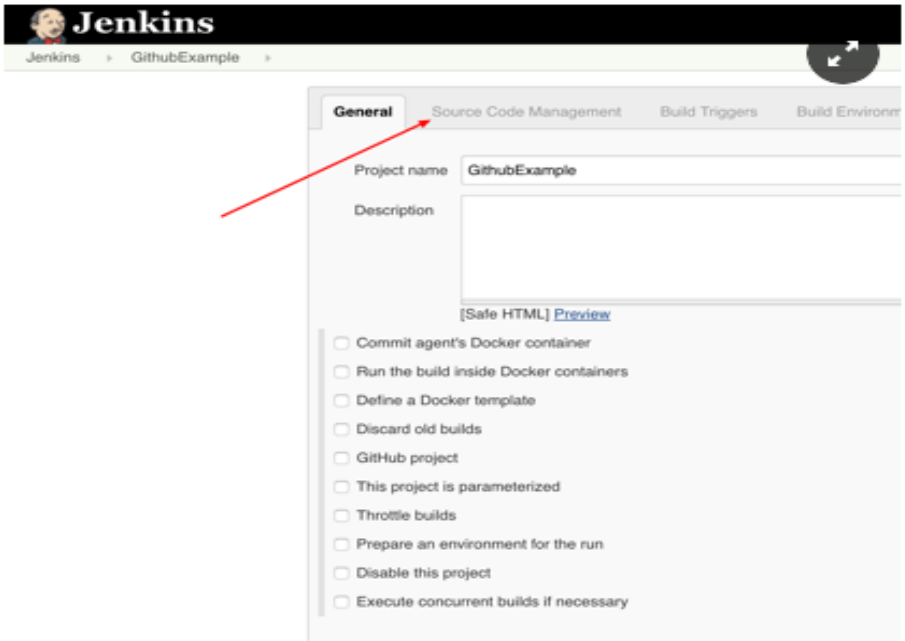
Step 5: In Jenkins, click on **'New Item'** to create a new project.

Step 6: Give your project a name, then choose *'Freestyle project'* and finally, click on *'OK'*.

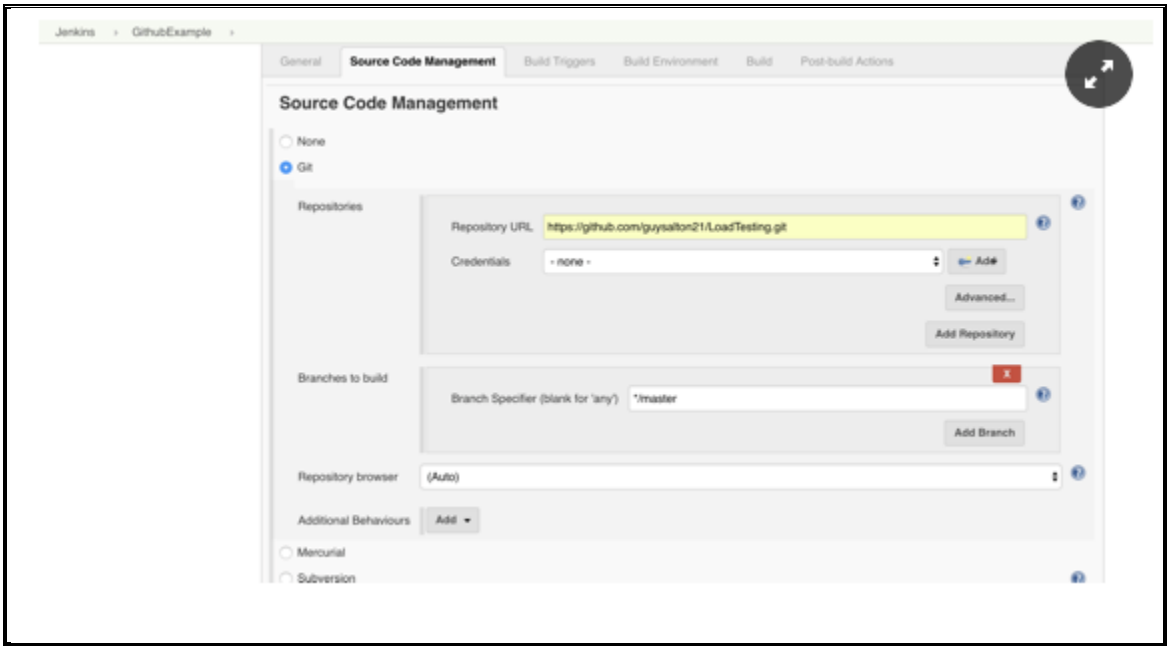




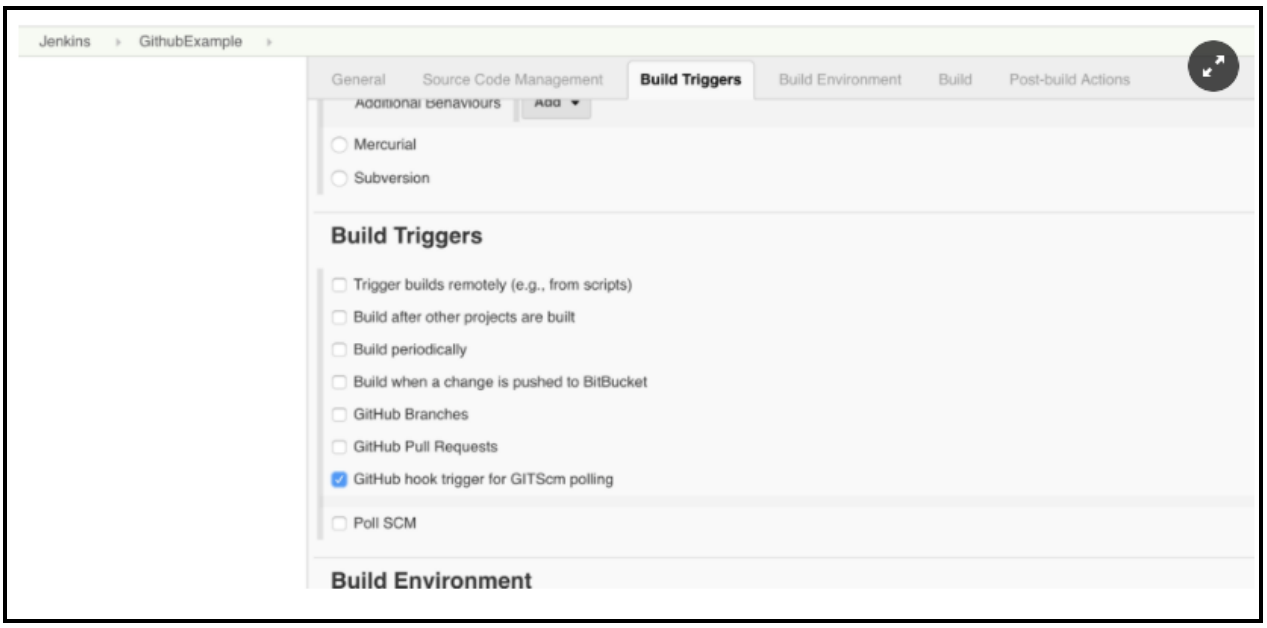
Step 7: Click on the **'Source Code Management'** tab.



Step 8: Click on Git and paste your GitHub repository URL in the **'Repository URL'** field.



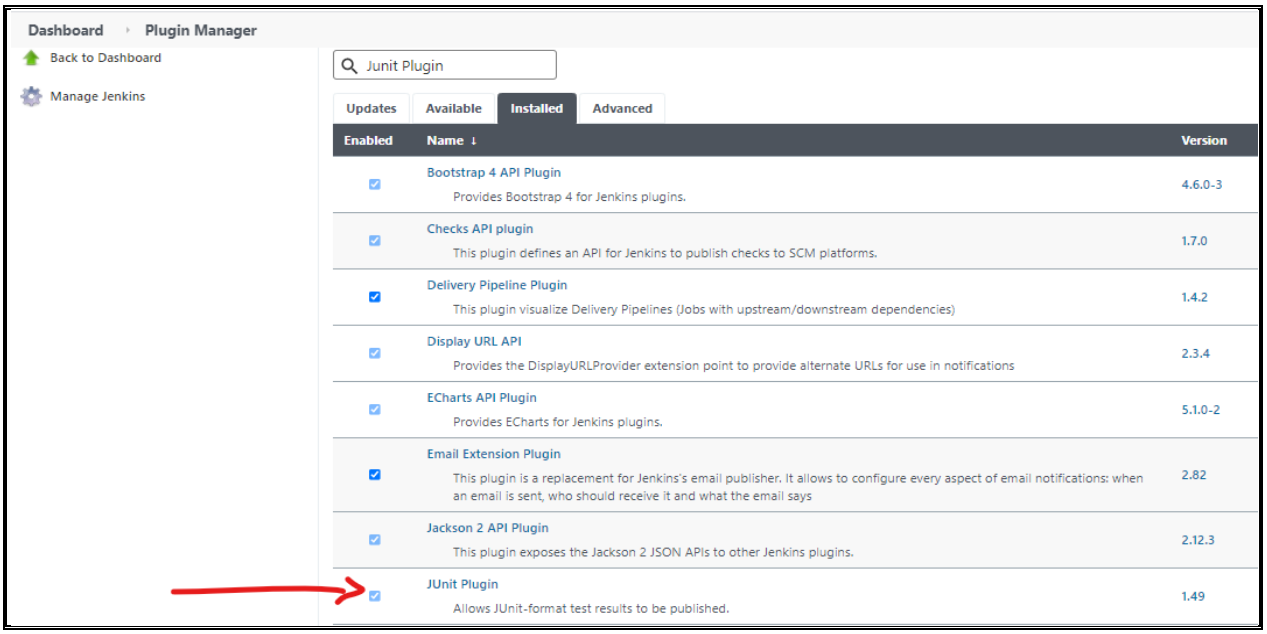
Step 9: Click on the **'Build Triggers'** tab and then on the *'GitHub hook trigger for GITScm polling'*. Or, choose the trigger of your choice.



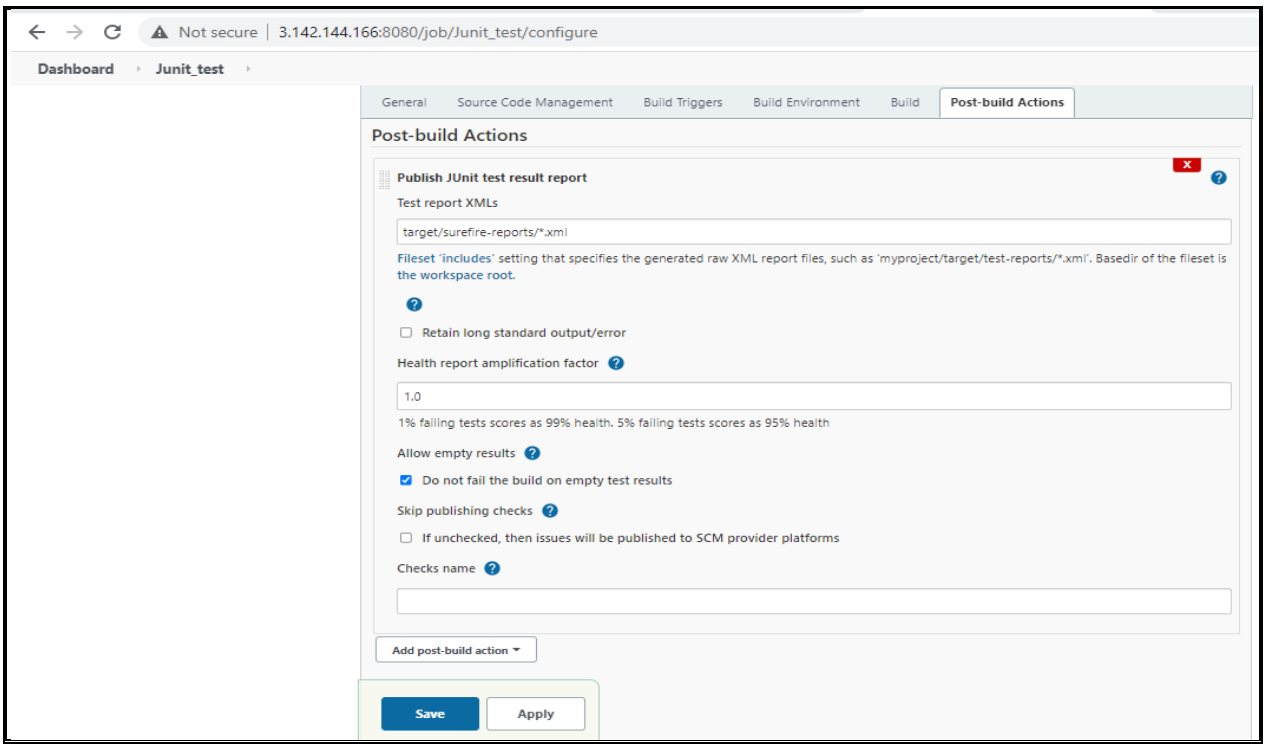
That's it! Your GitHub repository is integrated with your Jenkins project. You can now use any of the files found in the GitHub repository and trigger the Jenkins job to run with every code commit

How to run Unit Testing with Junit and Publish reports on Jenkins jobs page.

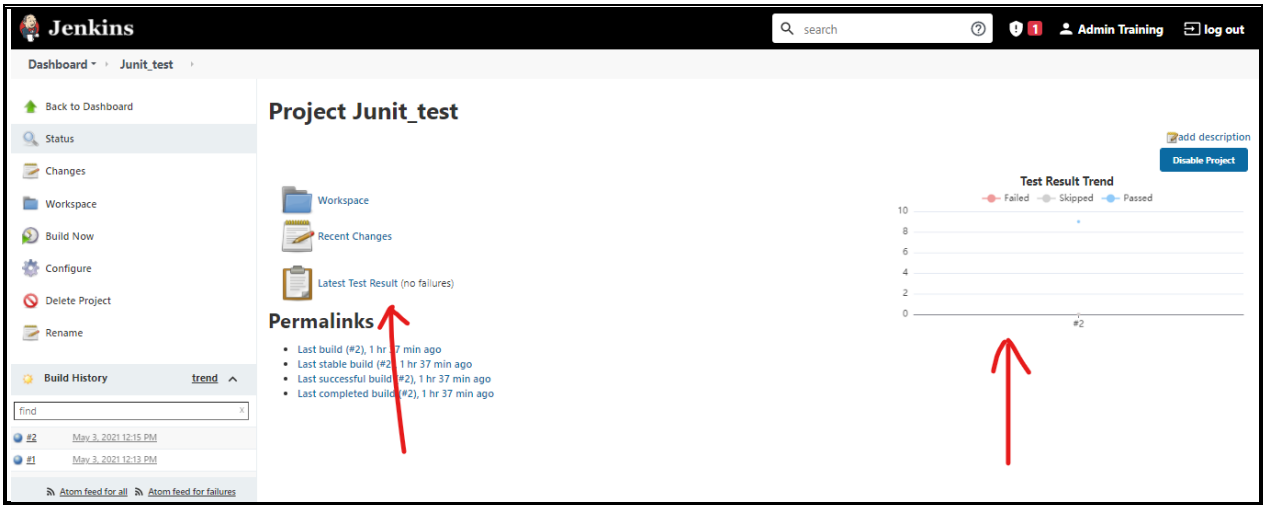
Step 1: Install Junit plugin from Manage Jenkins



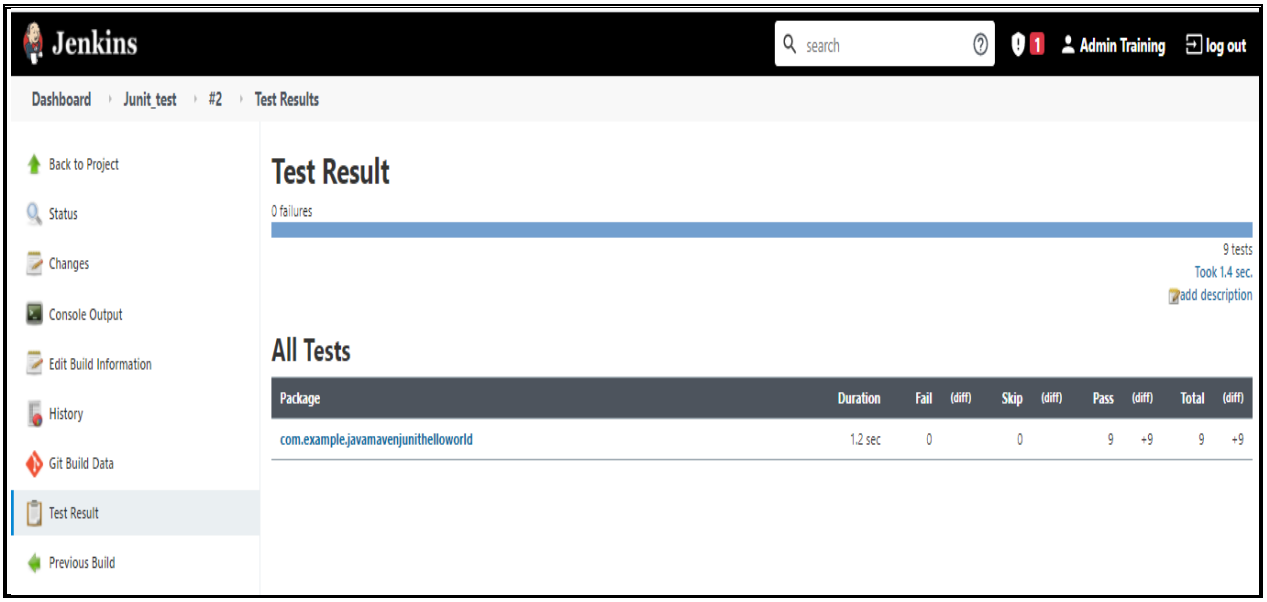
Step 2: In post build Actions, Configure the Publish Junit test result report



Step 3: Check on Jenkins Job page, for Junit Test results

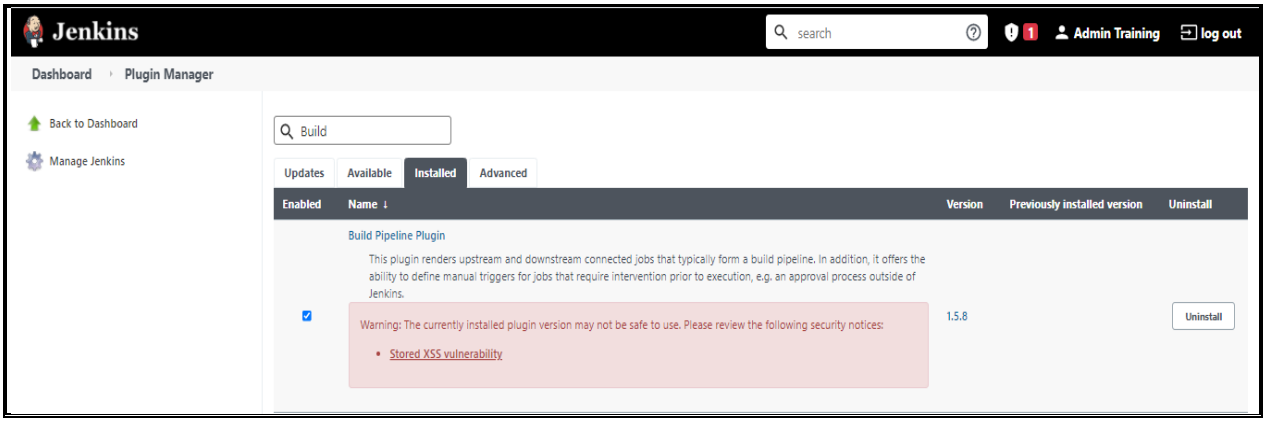


Step 4: Click Latest Test Result link for checking detailed reports.

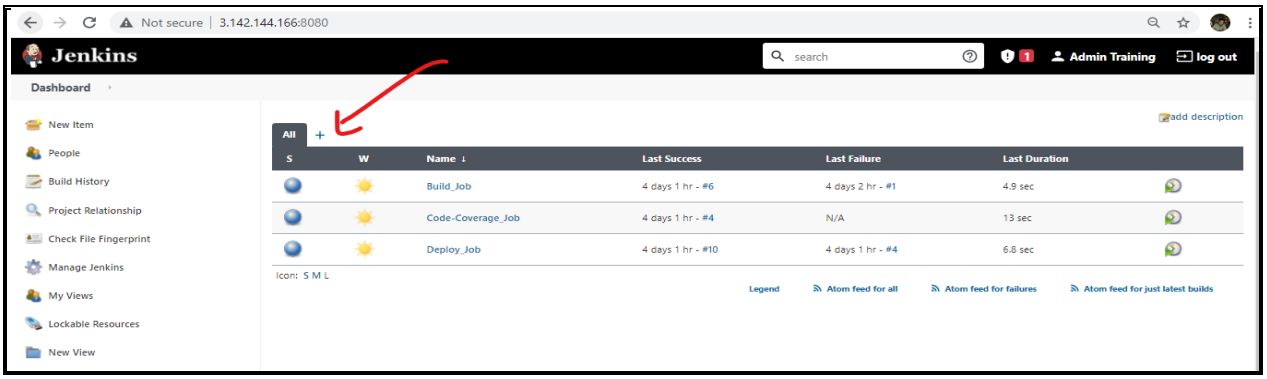


Build Pipeline

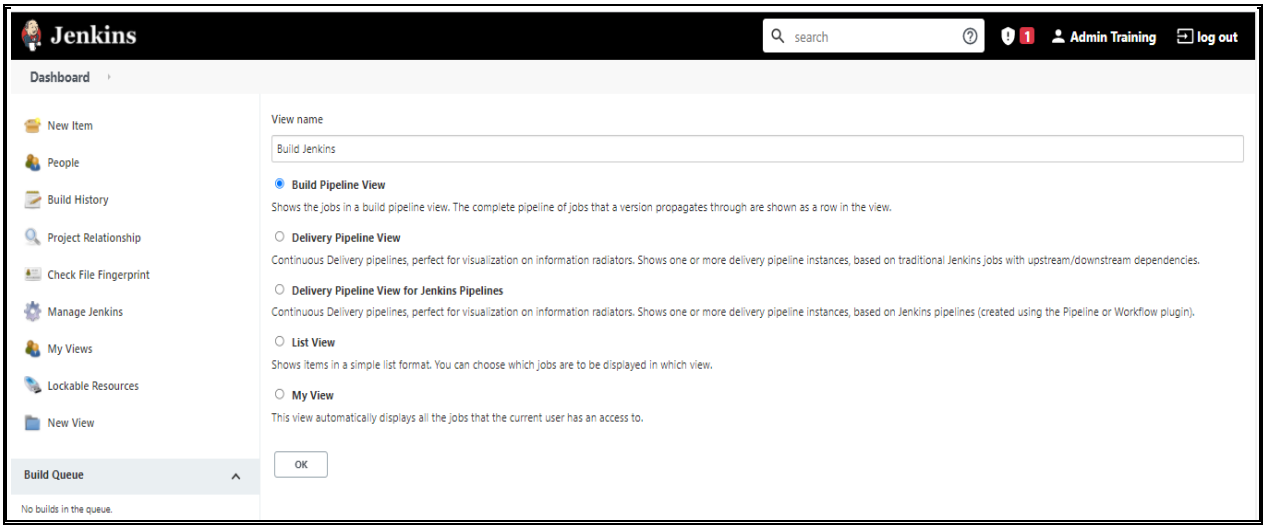
Step 1: Install Build Pipeline Plugin for Building Jenkins Pipeline.



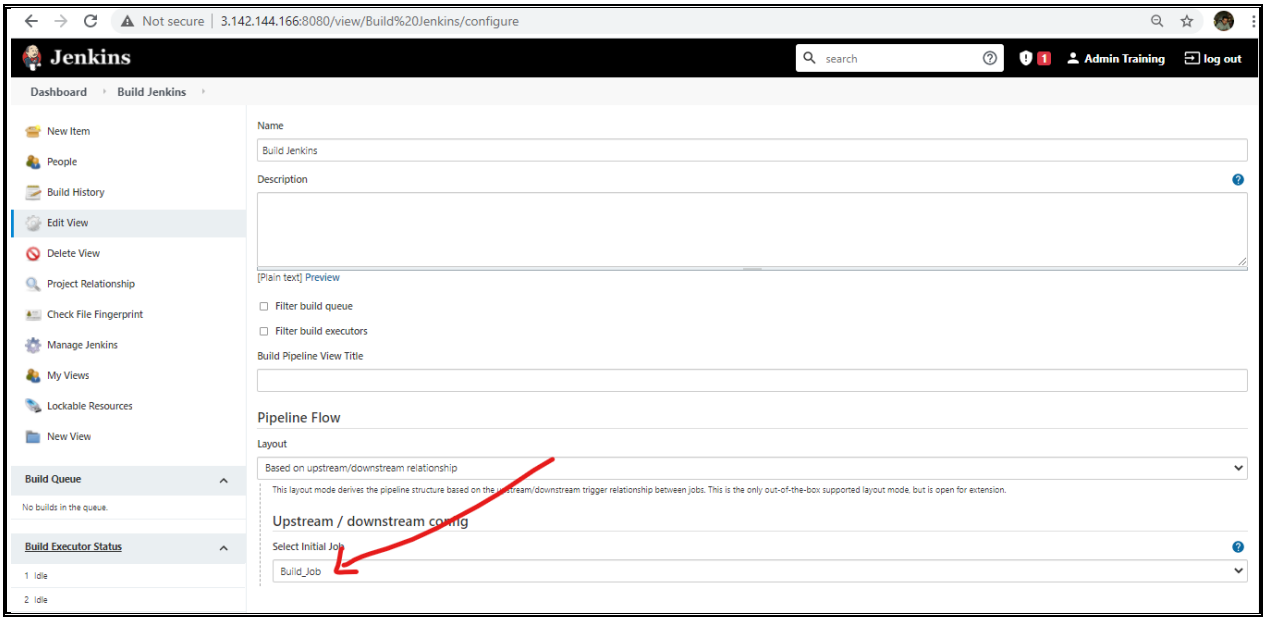
Step 2: Click on + symbol button

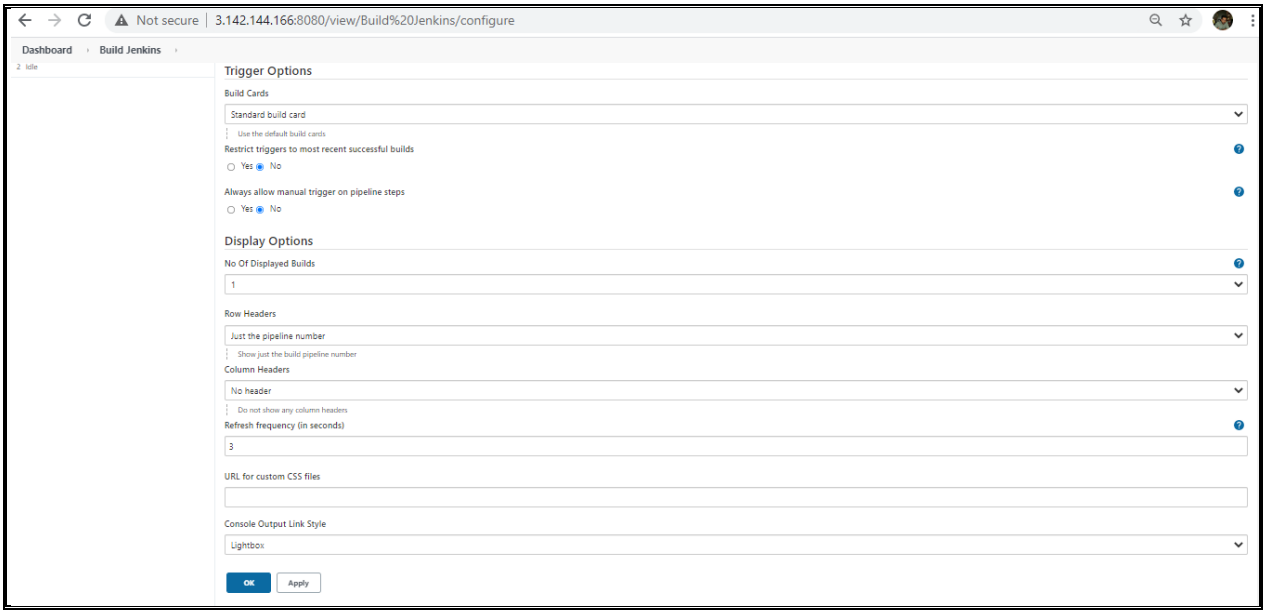


Step 3: Provide Name for Build Pipeline.

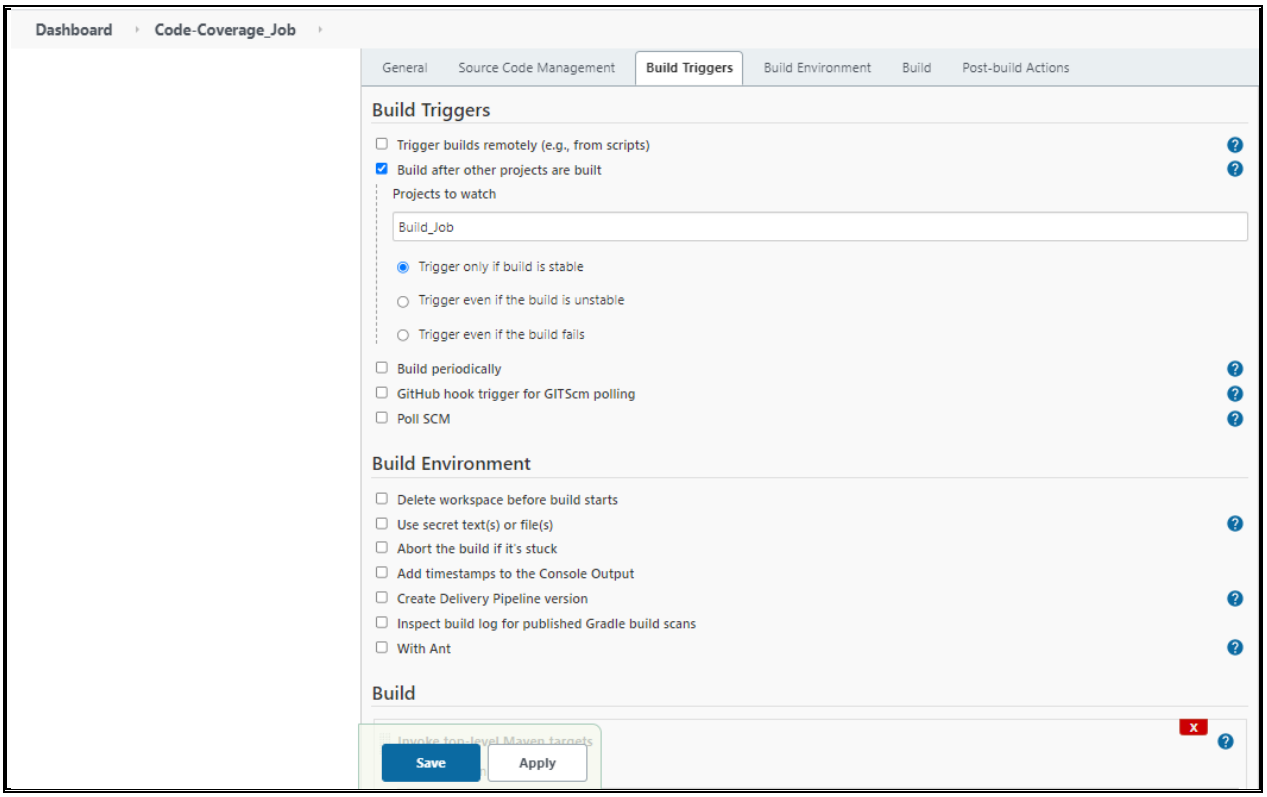


Step 4: Provide Initial Job details.

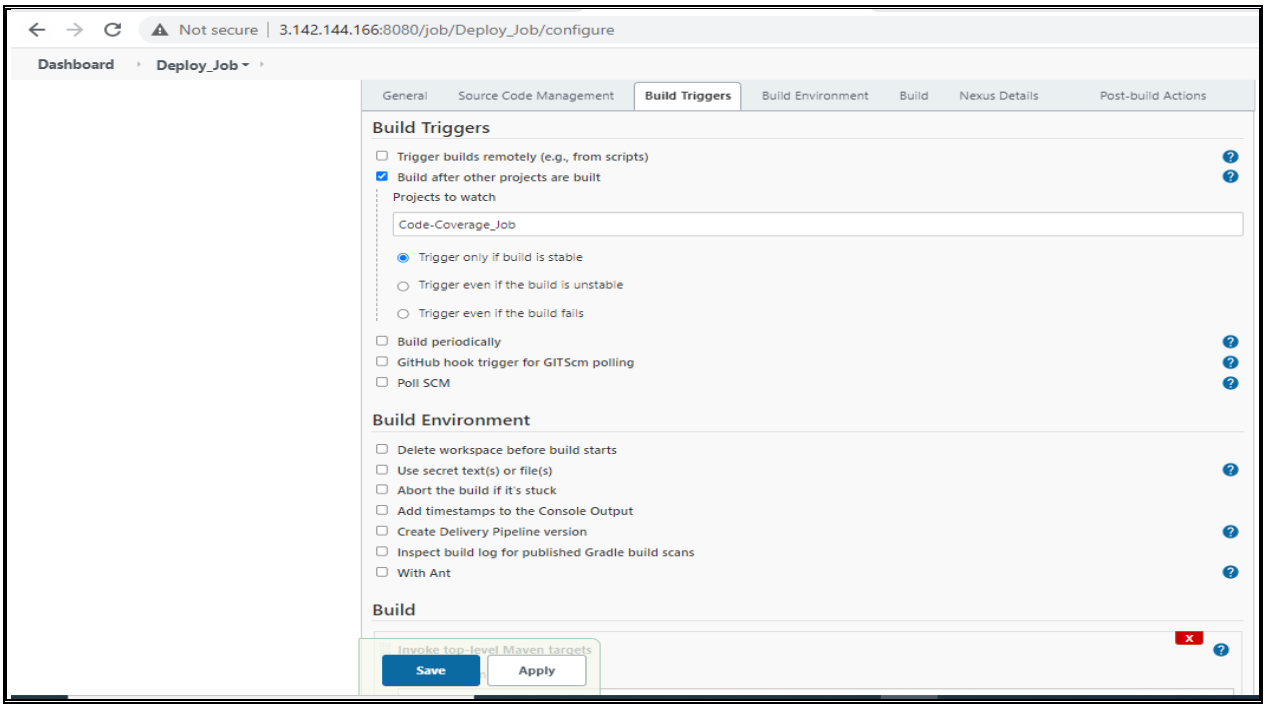




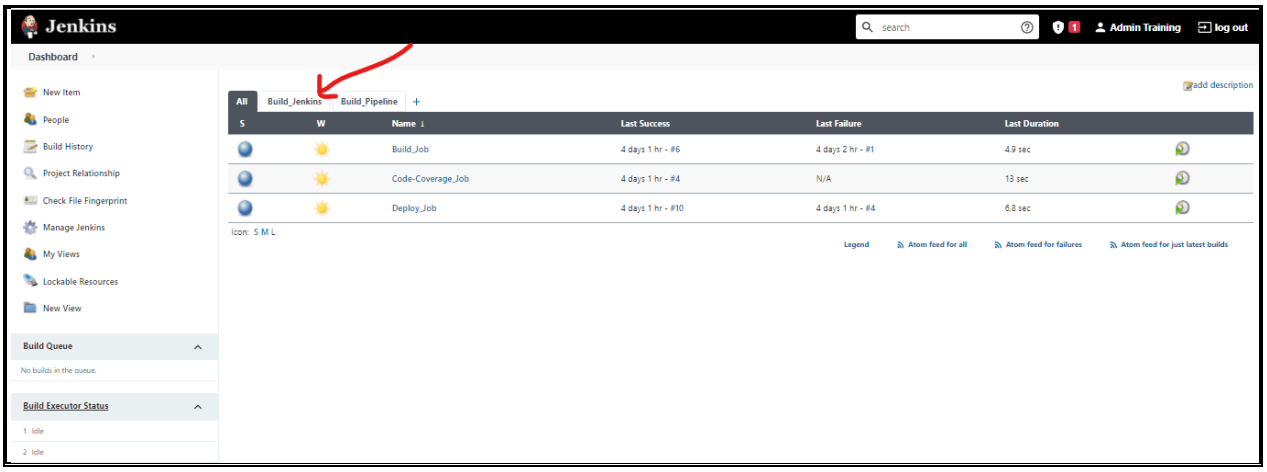
Step 5: Open Second Job and Add below configuration for triggering one job after another.



Step 6: Open Third Job and add same above configuration for triggering one job after another.



Step 7: Click on Build Jenkins View.



Step 8: After triggering the first Job, Build Pipeline looks like below.