Integrating Jenkins with Slack Notifications

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real-time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

So How automation build environment to run build every single push happens on the source repository we need to broadcast the status of the build to the team members will ensure code base sanity.

This blog will help you to set up a continuous integration environment using Jenkins with slack for notifications.

Here Are some steps you need to follow to set up Jenkins that we discuss in our previous blog.

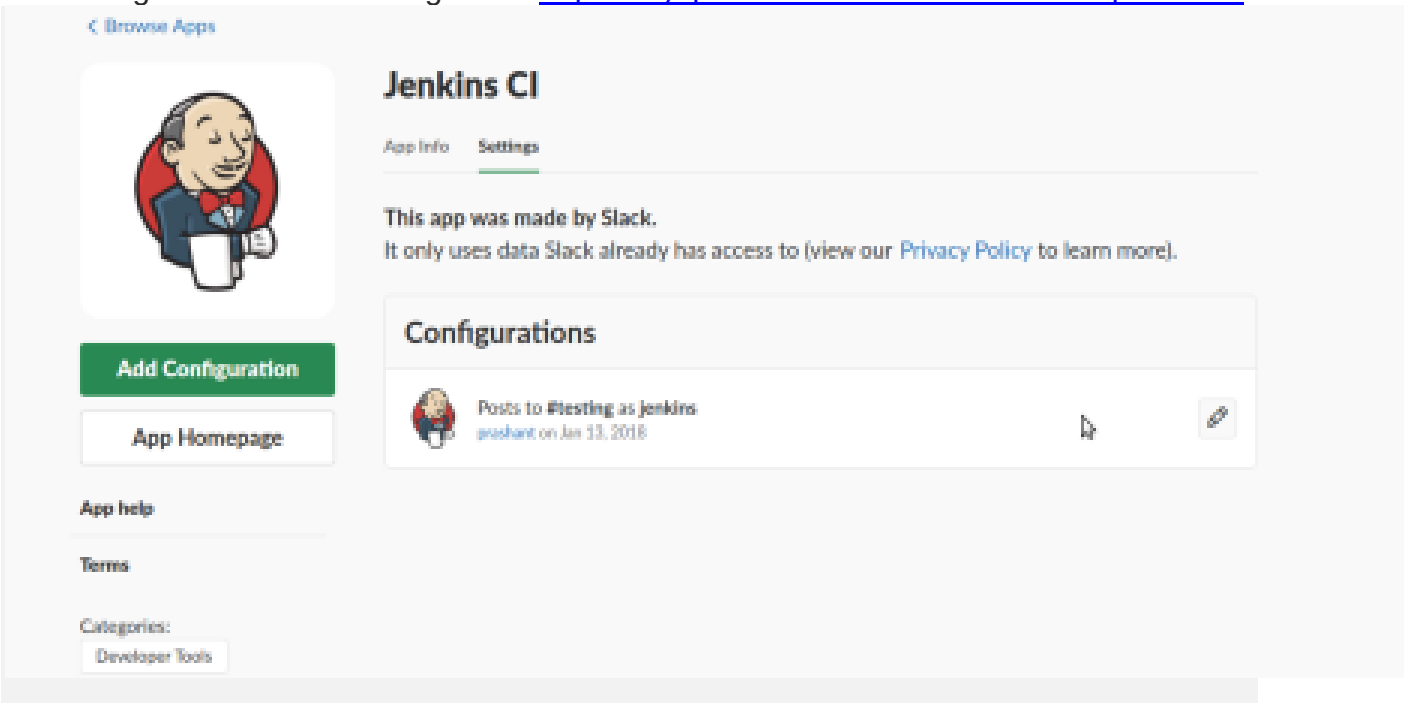
Setup Jenkins on CentOS with Docker for Selenium
I haven't found any walk-through about setting up Jenkins on CentOS with Docker for Selenium, and since I got to do it…
medium.com

Integration Jenkins with slack

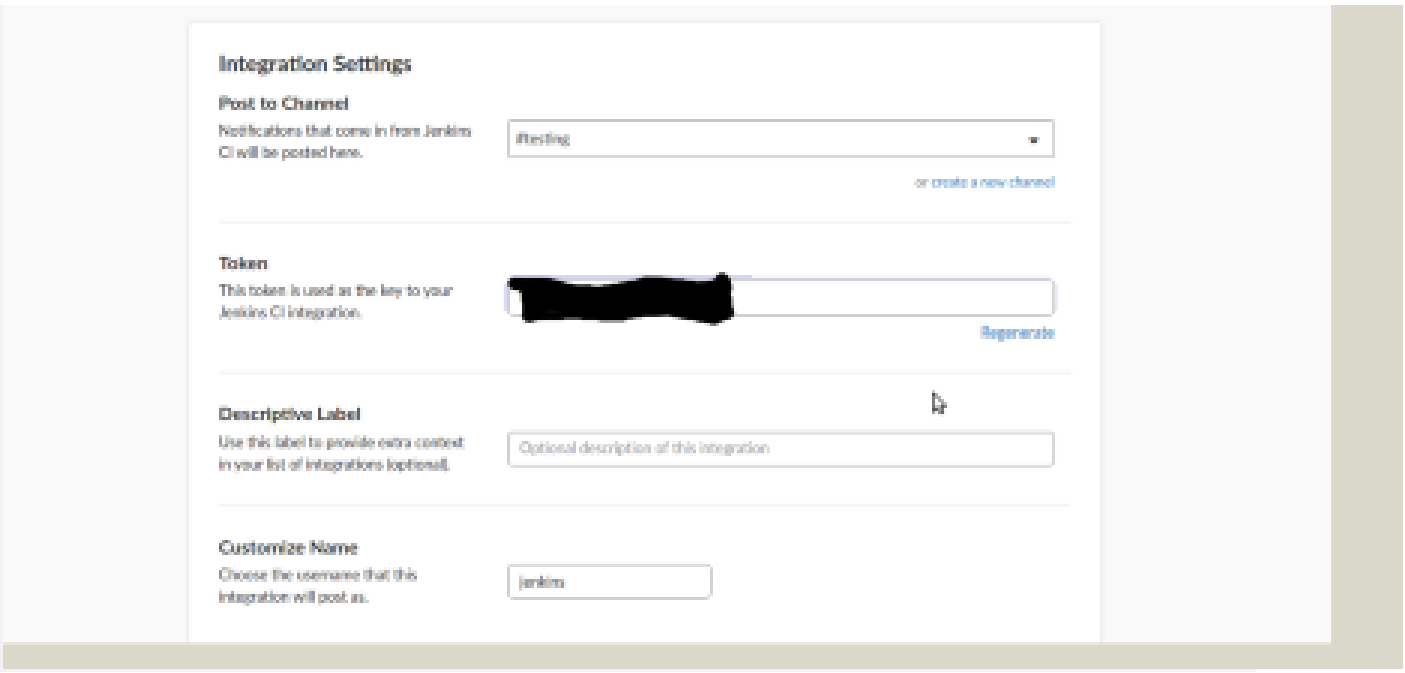
First, we need to configure slack on our machine.

1. Create a slack account: https://slack.com/

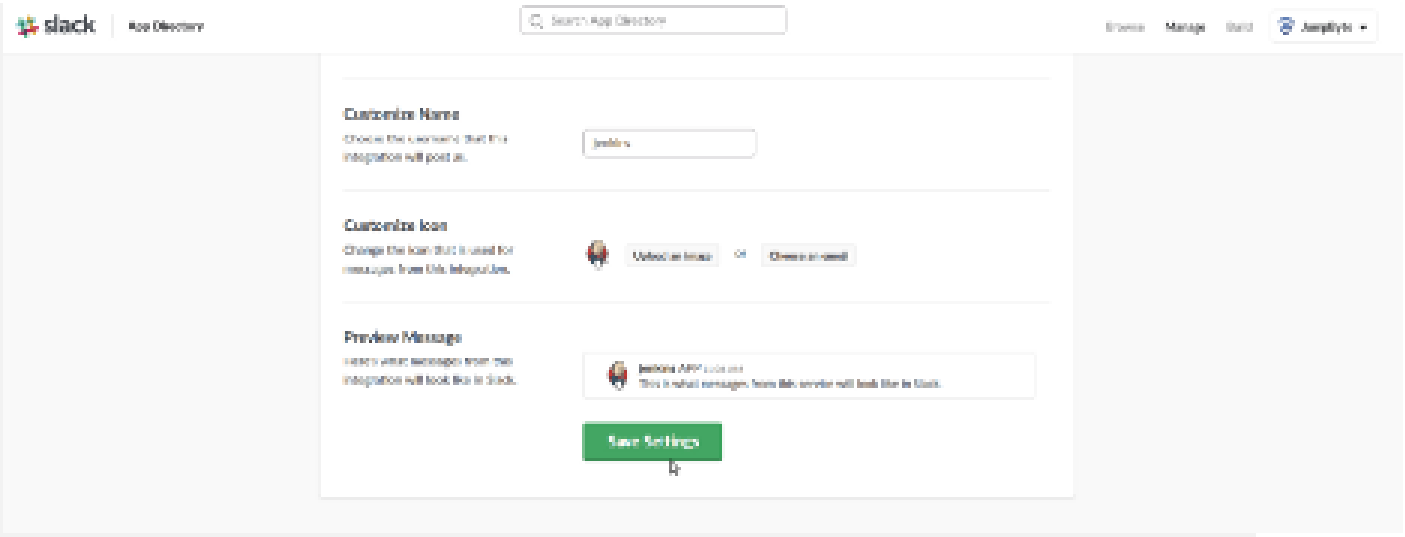
2. configure the Jenkins integration: https://myspace.slack.com/services/new/jenkins-ci



First, install 'Jenkins-ci' and then Add configuration and set channel and all thing like



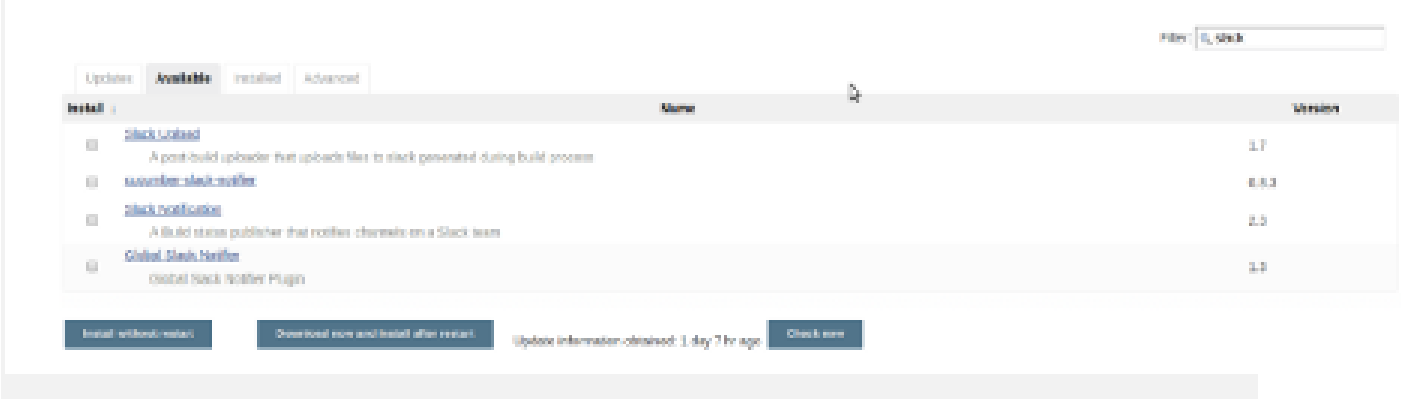
Add channel name here and note Token.



Then scroll down and click on save setting button.

After that, we need to set configuration on Jenkins [Slack Notifications plugin](#).

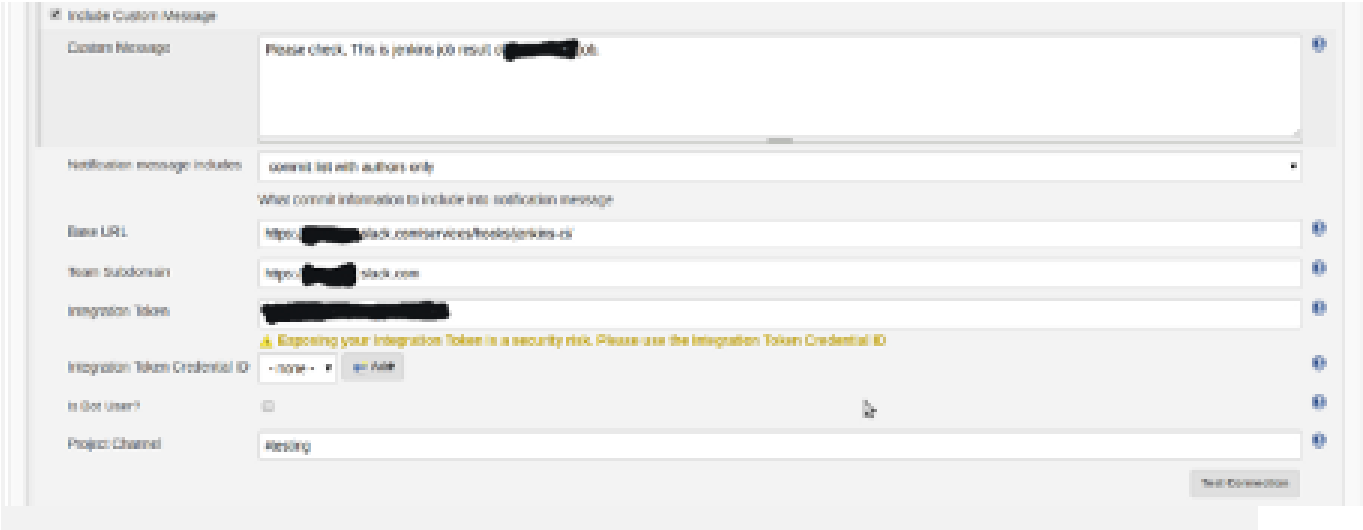
For Jenkins to notify slack, we need to install in Jenkins. By now, you must know how to do this, so go ahead and install the plugin.



Select Slack Notifications plugin and click on Install without restart button.

It displays success with the plugin install successfully.

Then go to Jenkins job if you have no job then you need to create one job and go to the post-build section. Select Slack Notification` and it's display Slack Notification` wizard.



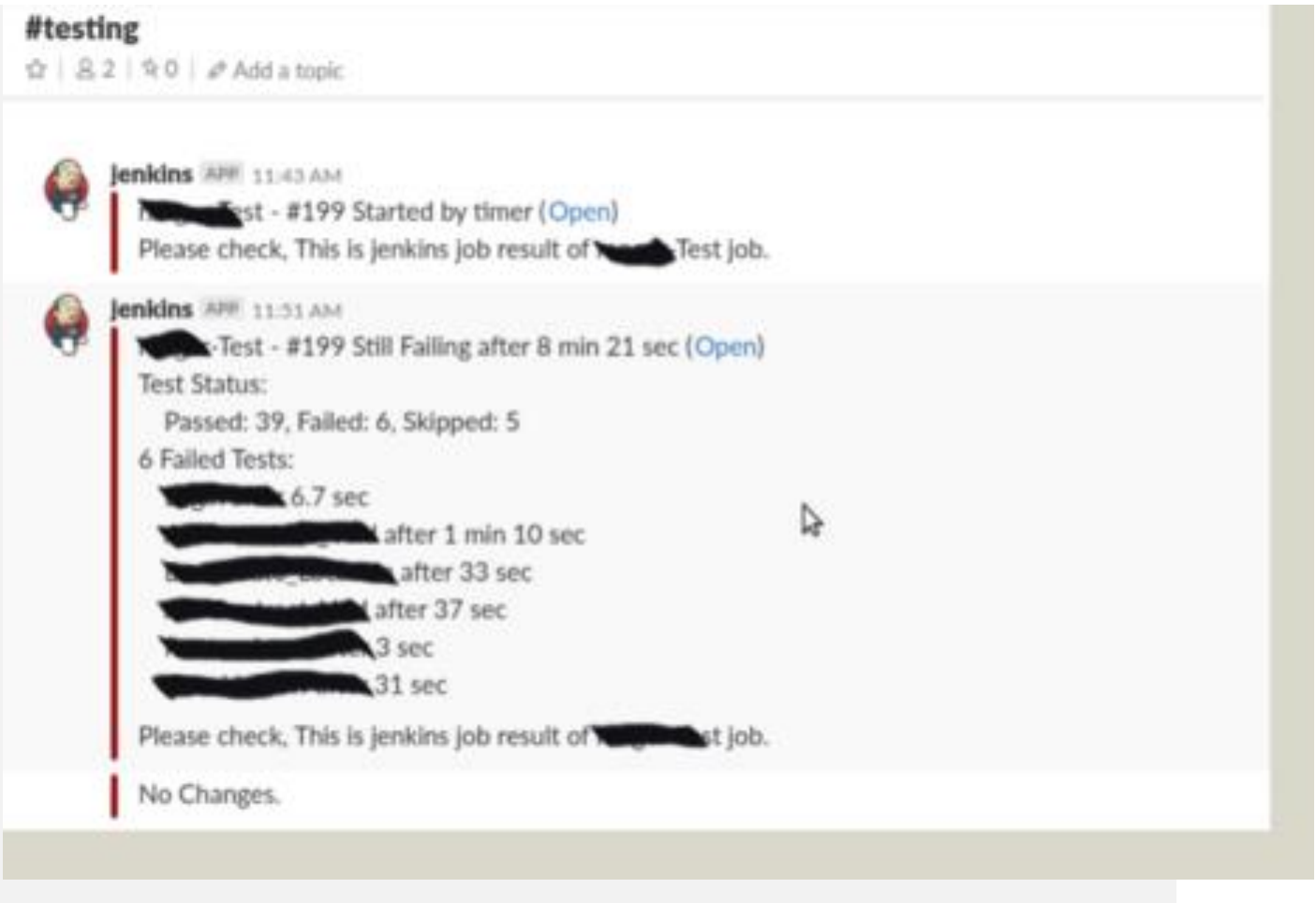
Enter Base URL, Team subdomain, Integration Token that created on you slack and Project Channel and click on apply and save button.

And also for Integration Token, It’s recommended is to configure credentials for your Integration Token. Just add credentials (Secret text) and define your token.

Test setup

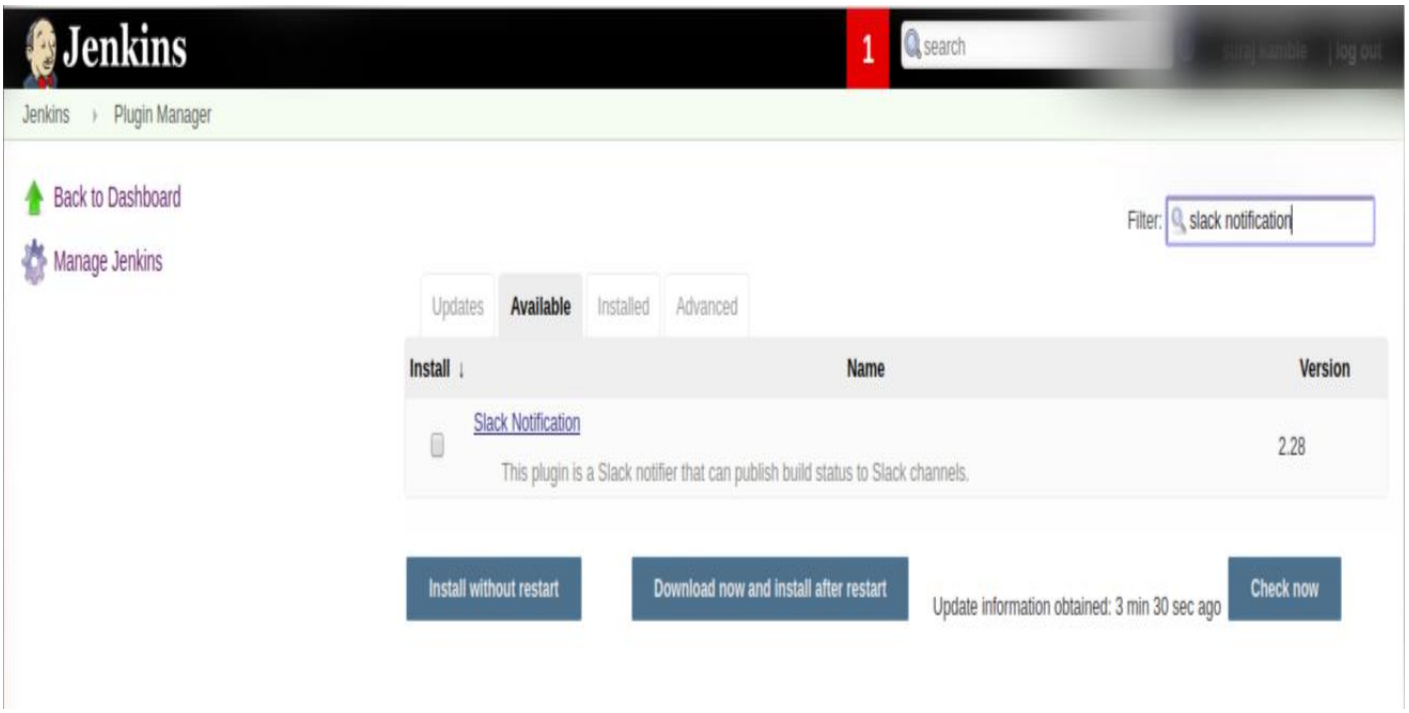
Now we can finally test our current setup. We can run the Jenkins job.

based on your build result it sends slack notifications like here.

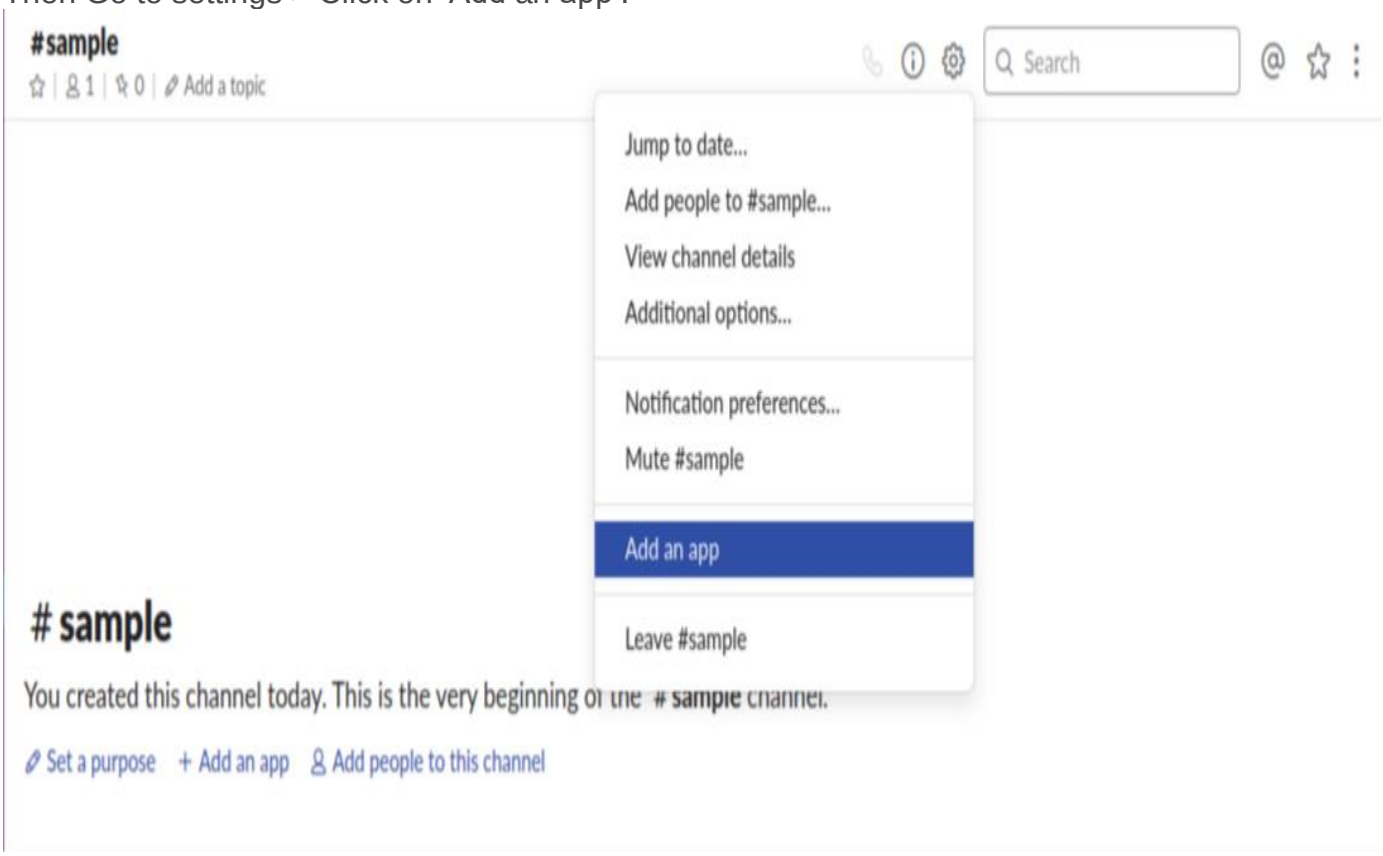


Section 1 – Integrate Jenkins tools With Slack Notifications

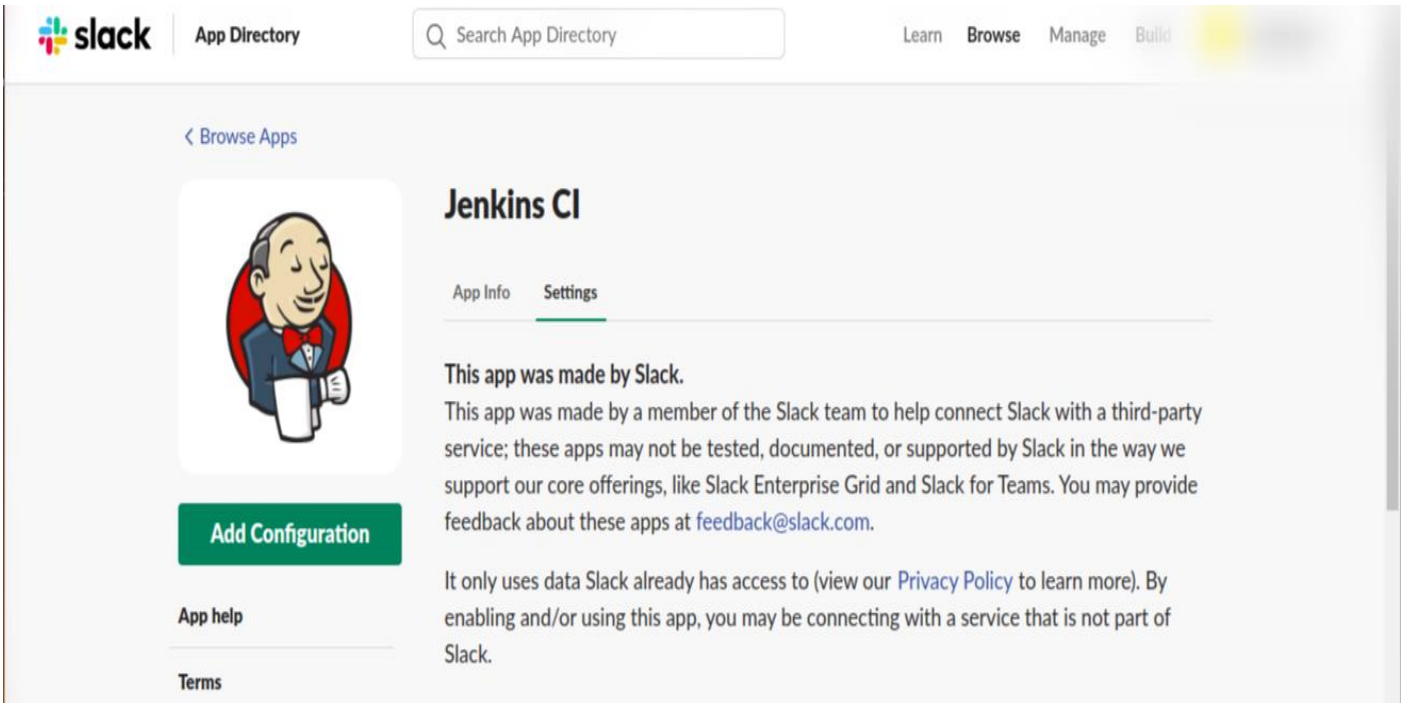
Go to Jenkins > Manage Jenkins > Plugin Manager > Search for the “Slack Notification Plugin” and install it.



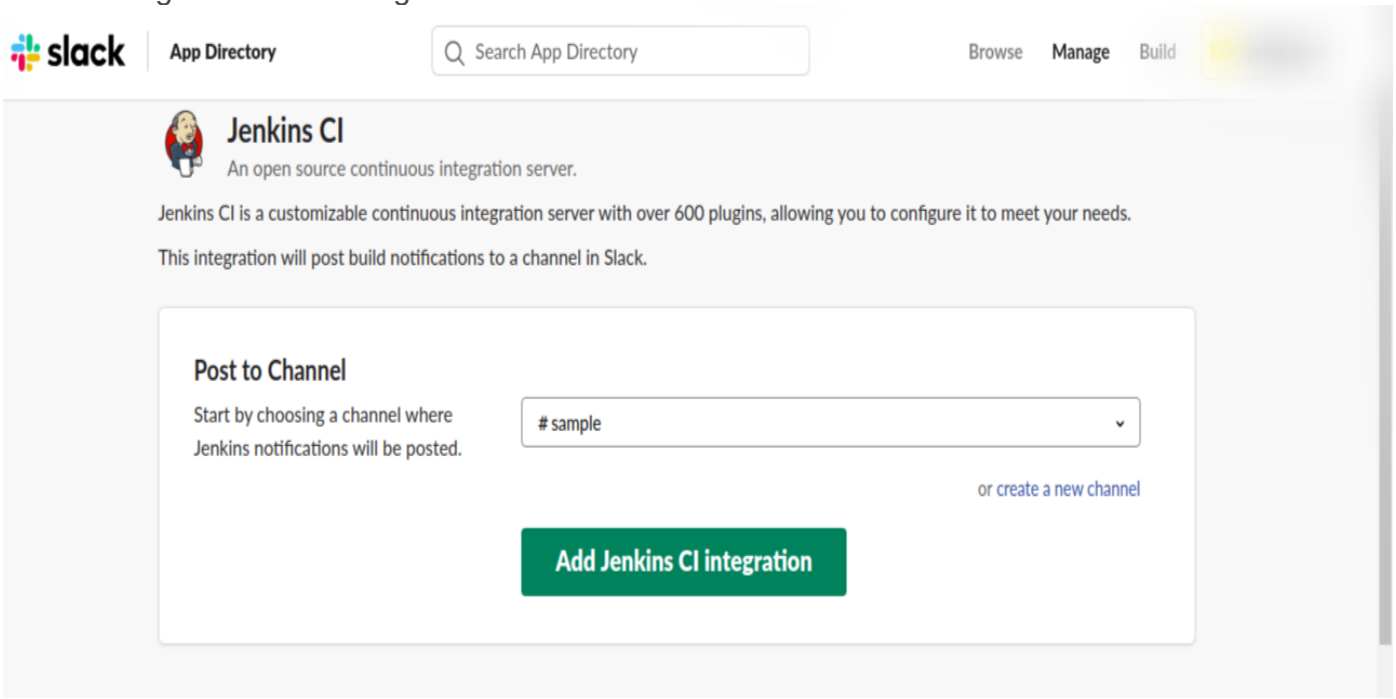
Create a Slack account (if you do not have one) and then create a private/public channel. In my Scenario, I have created a channel named “sample”.
Then Go to settings > Click on ‘Add an app’.



Click on View App Directory. Search for “Jenkins CI” and click add configuration.



Set the channel name that you wish to integrate with Jenkins and click on “Add Jenkins CI Integration”.You will get a Team Domain name and Integration Token Credential ID after adding Jenkins CI Integration.



After a successful installation, go to Jenkins > Click on ‘Manage Jenkins’ > Click on ‘Configure System’.
Find the ‘Global Slack Notifier Settings’ section and add the following values:

- Team Subdomain:

DomainName

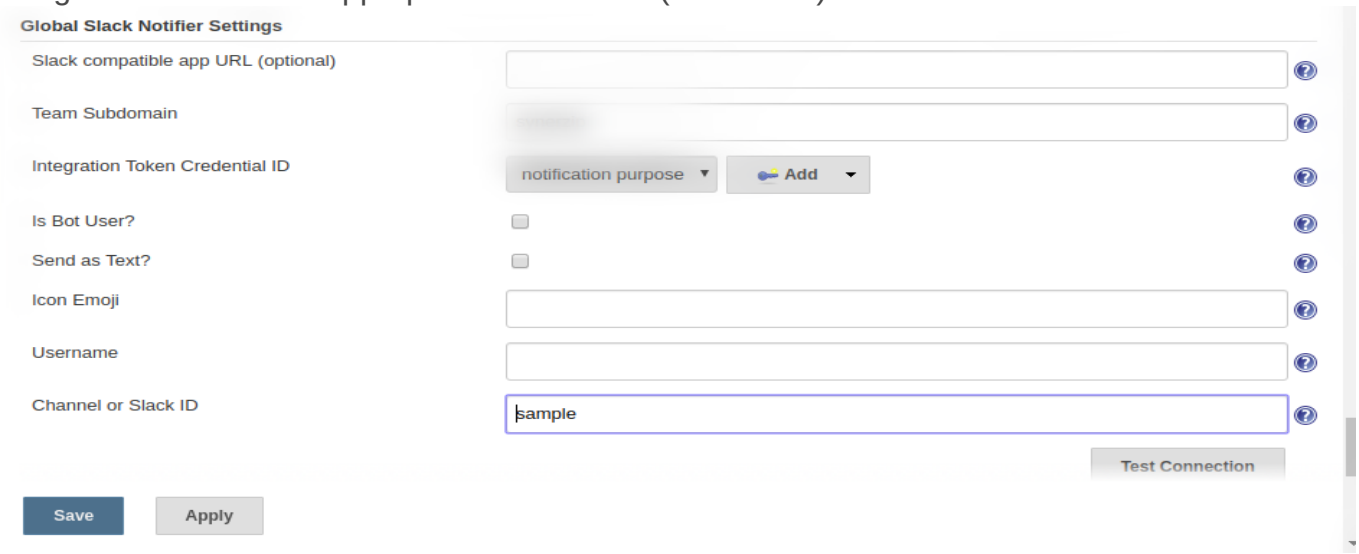
- Integration Token Credential ID:

secret text

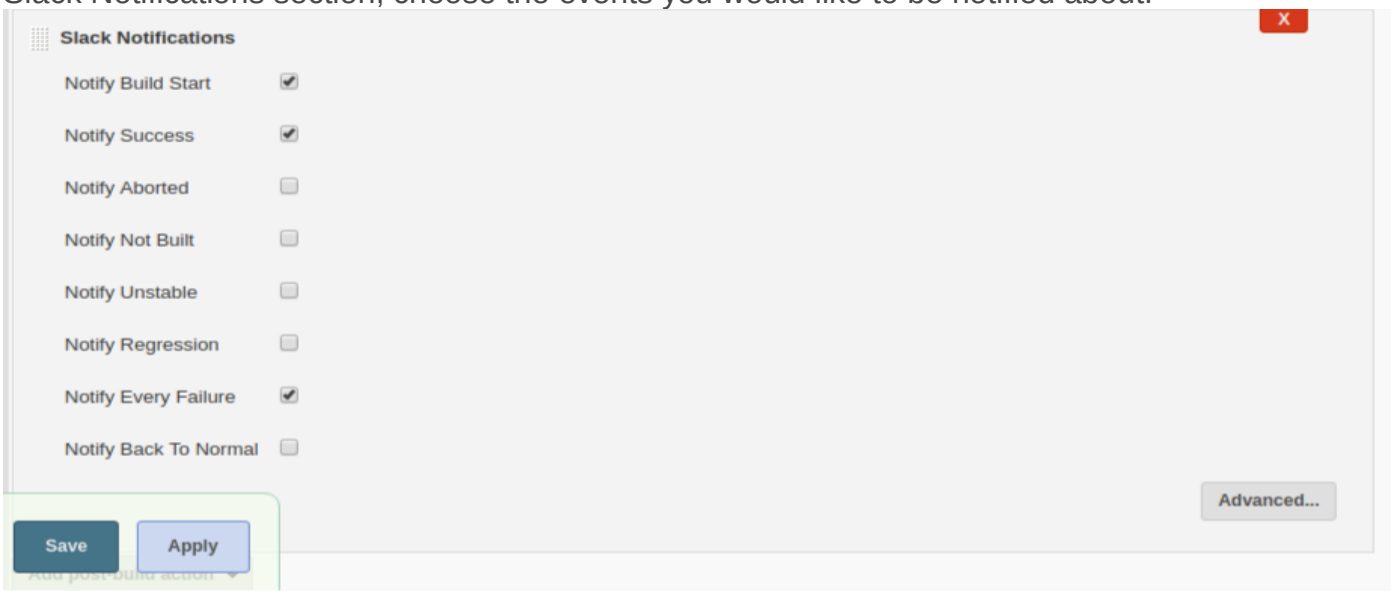
- Channel or Slack Id and other fields are optional

Then save these settings

Note: Exposing Integration Token is not secure. Please remember to replace the Integration Token with appropriate credentials (secret text).



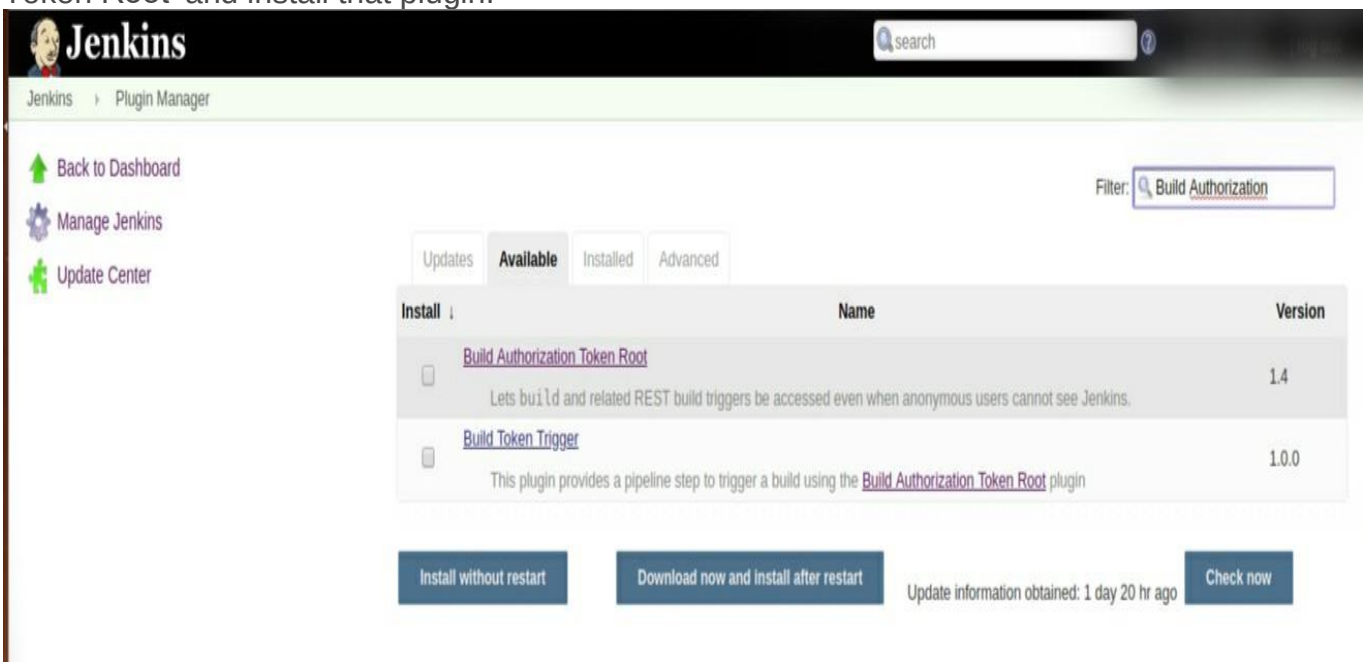
Go to Jenkins job > Go to the post-build section > Select Slack Notifications. In the Slack Notifications section, choose the events you would like to be notified about.



Now Build the Jenkins job. Based on the build result you will receive a Slack notification on the Slack channel as shown below.

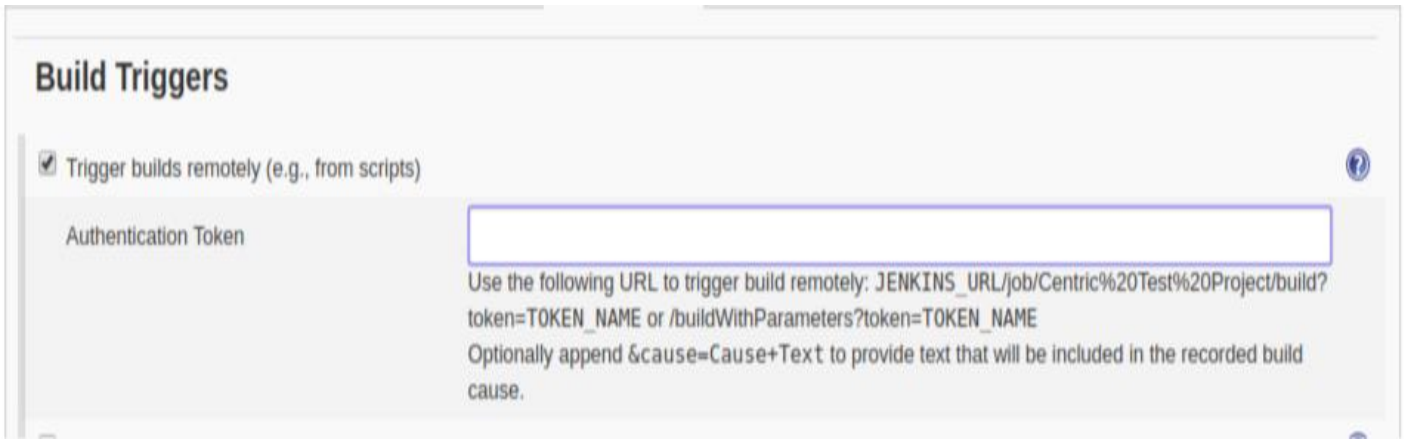
Section 2:- Trigger Jenkins Job From Slack

Go to Jenkins > Manage Jenkins > Plugin Manager and search for ‘Build Authorization Token Root’ and install that plugin.



After installing the plugin, go to Dashboard > Click on Job name > Click on Job configure > On Build Triggers Click

On ‘Trigger builds remotely’ (e.g., from scripts) add Authentication token (you can generate a random token from here) for eg. “57464576646654”.



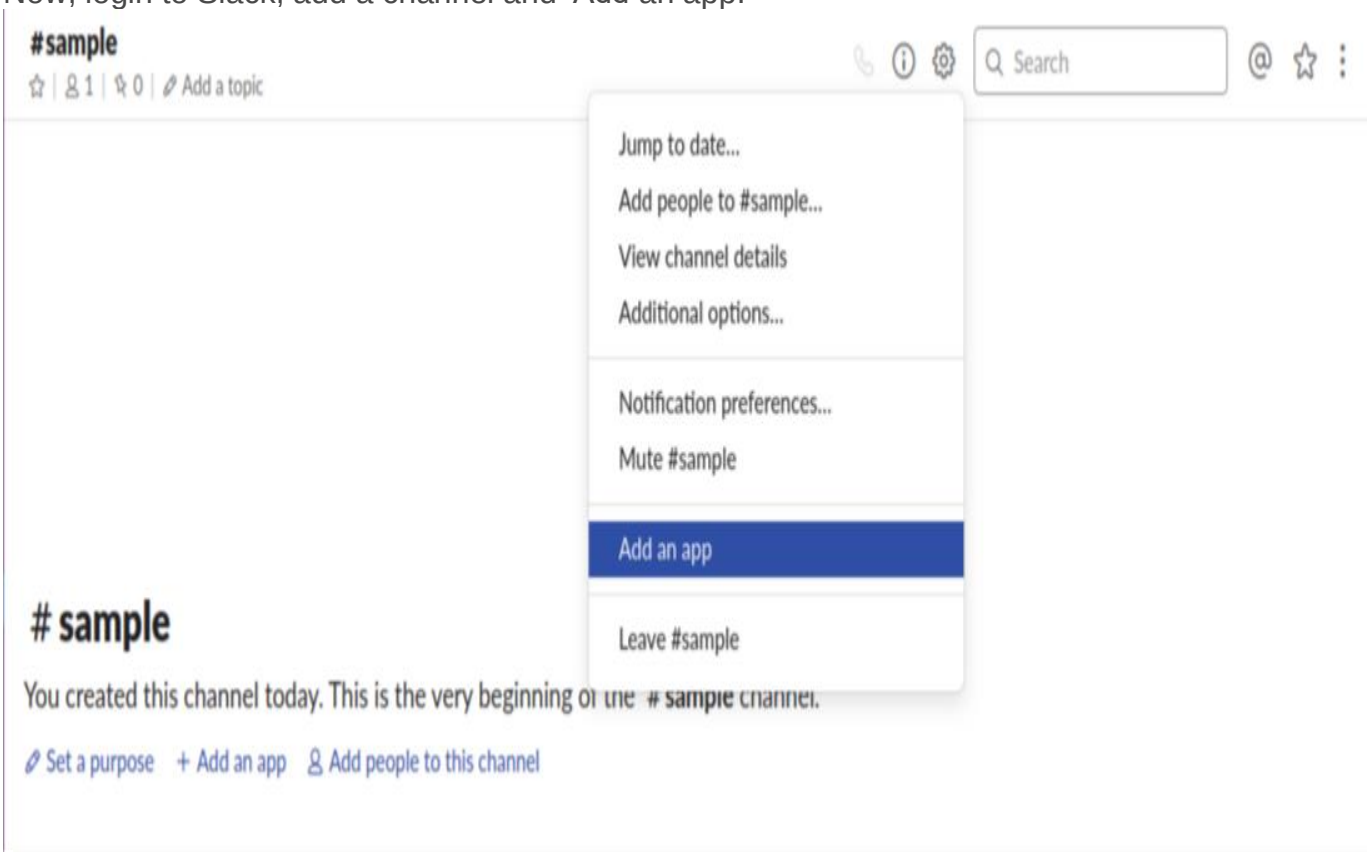
Use the following URL to trigger the build:

```
JENKINS_URL/job/SampleJob/build?token=TOKEN_NAME
```

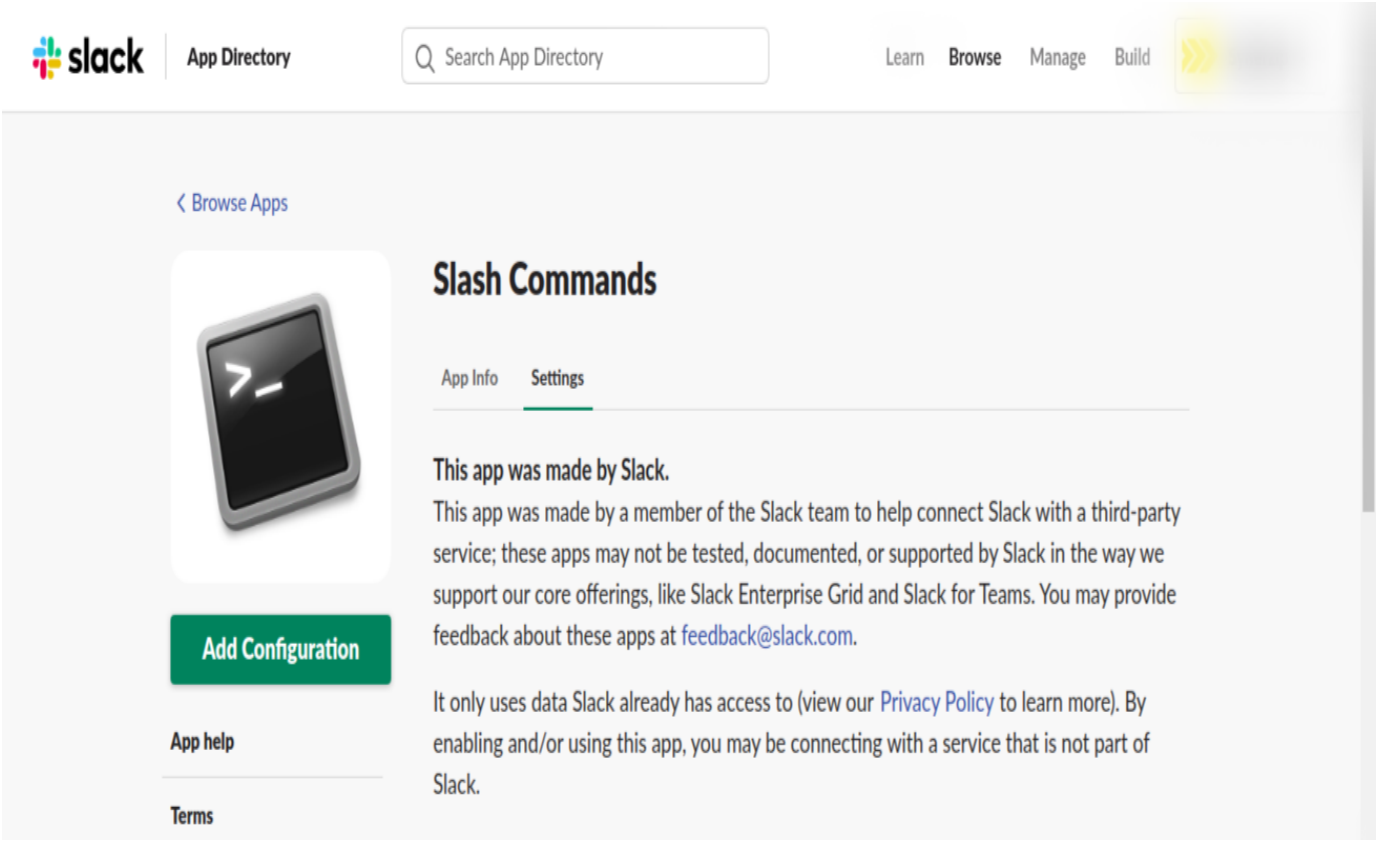
OR

```
JENKINS_URL/job/SampleJob/buildWithParameters?token=TOKEN_NAME
```

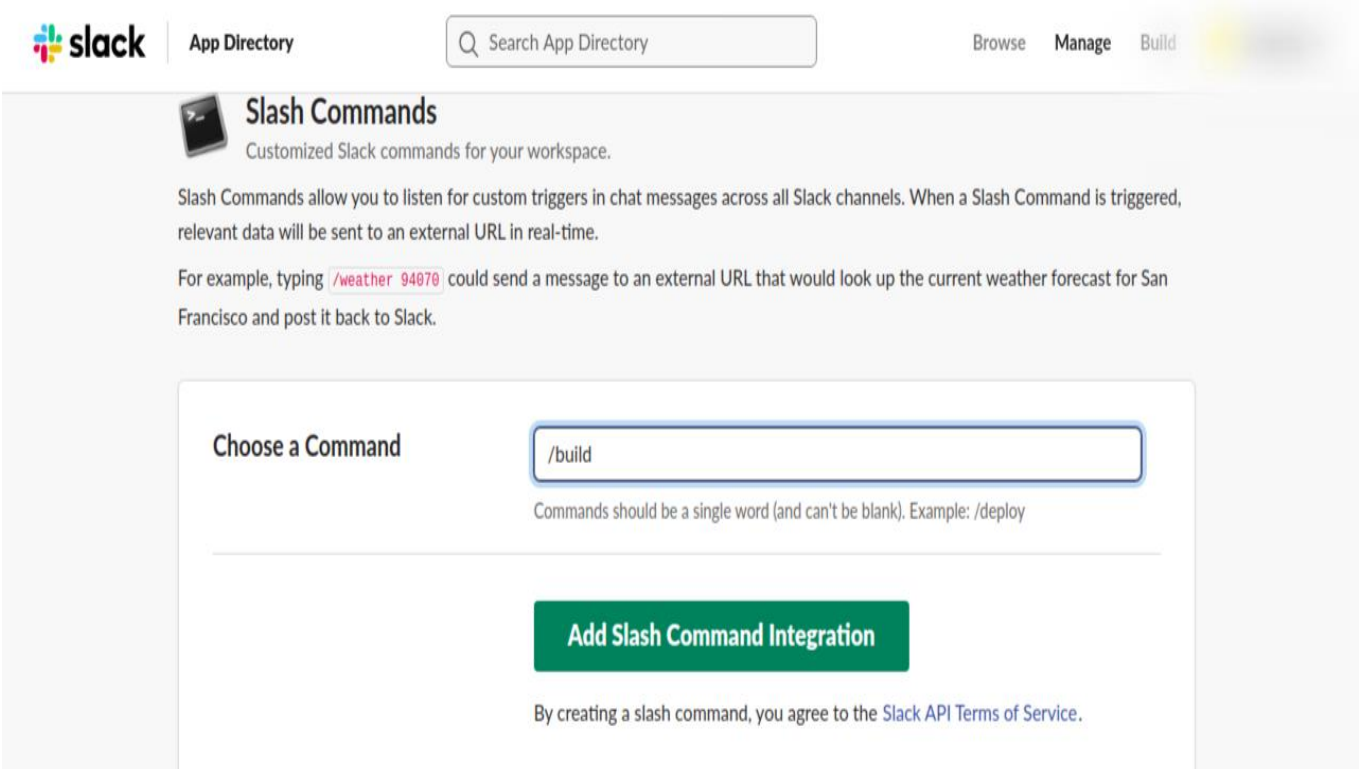
Optionally append '&cause=Cause+Text' to provide text that will be included in the recorded build cause.
For e.g:
Now, login to Slack, add a channel and 'Add an app.



'Click on 'View App Directory', search for "Slash Commands" and click add configuration.

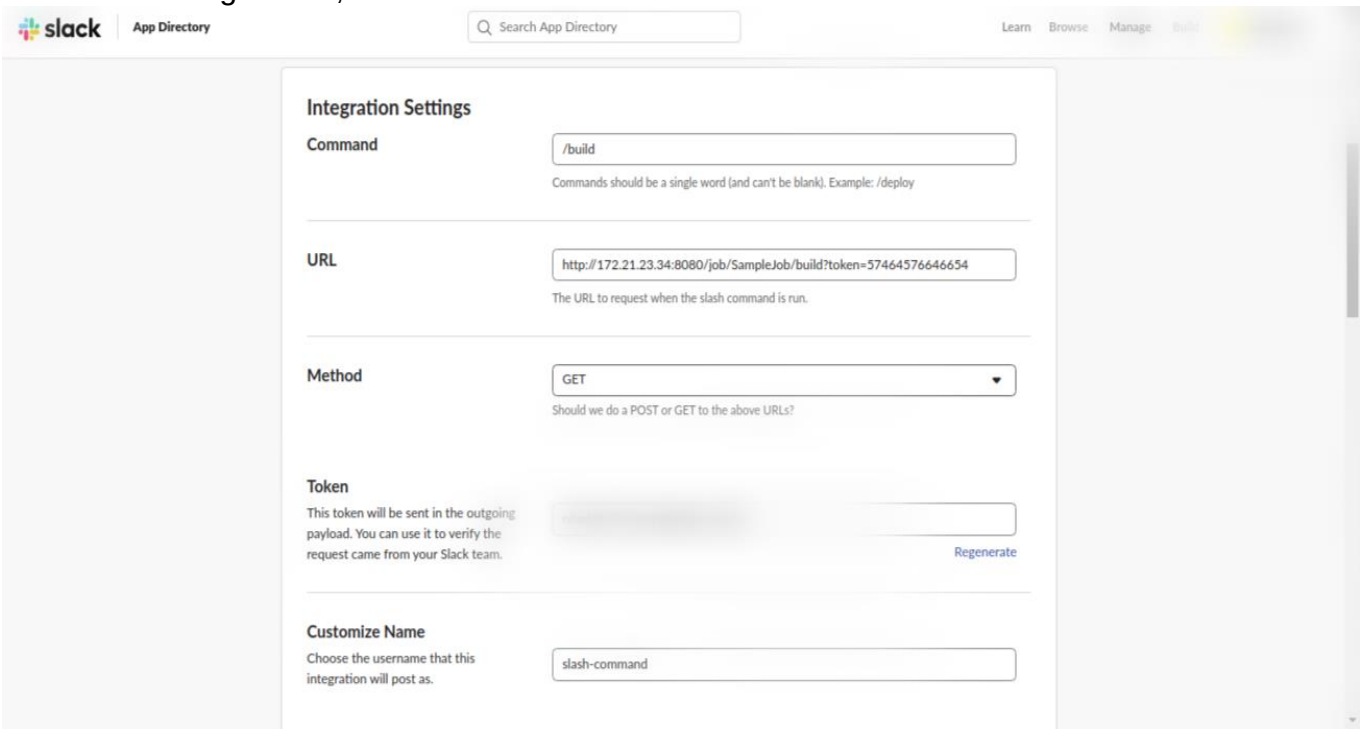


Click on Add Configuration and, in my scenario, I have set my command name as "/build". However, you can give custom name to your command. Click on the "Add Slash Command Integration" button.

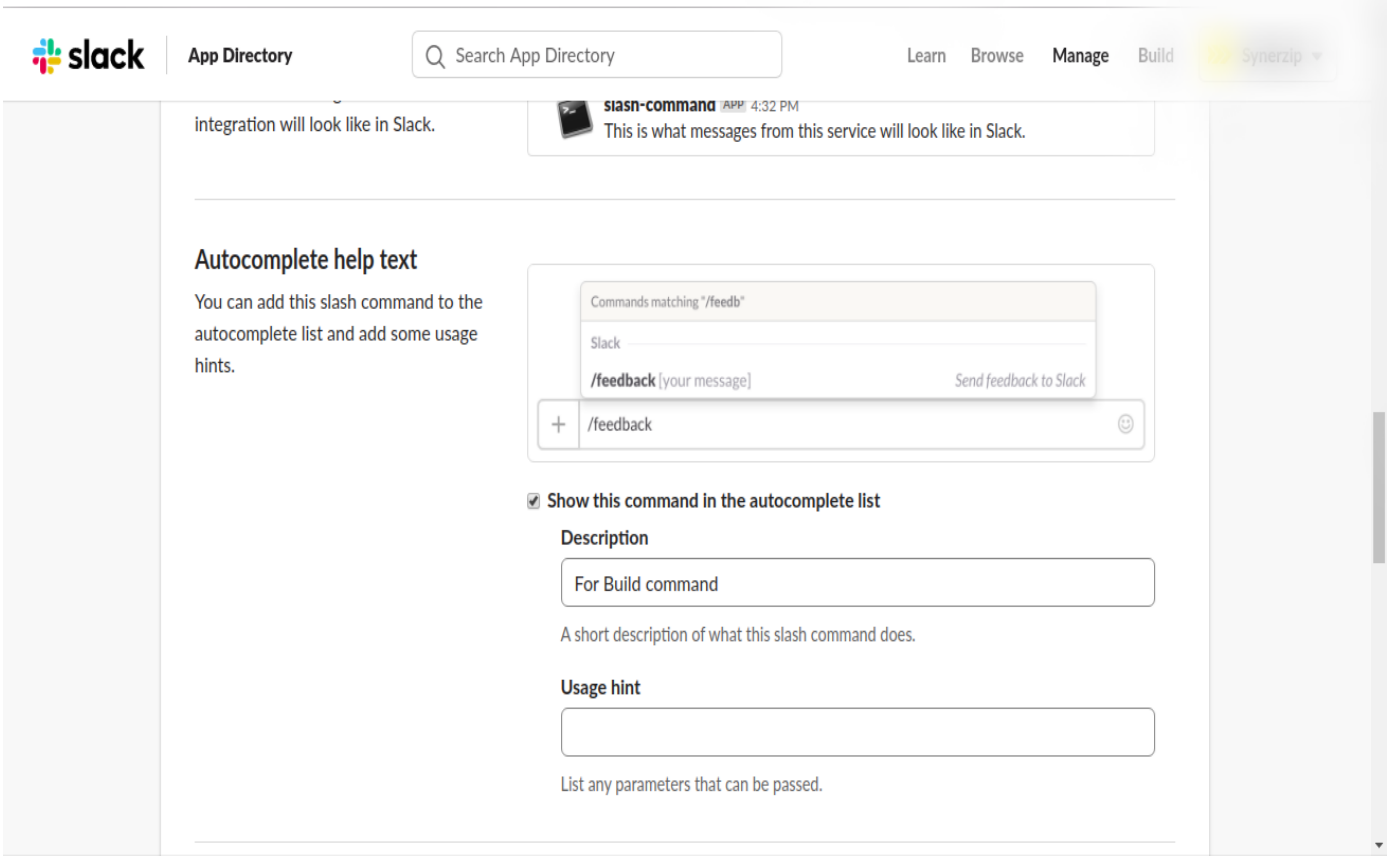


Provide a request URL that is needed when the slash command runs and performs a Post and Get method to the URL.
Shown below is a sample

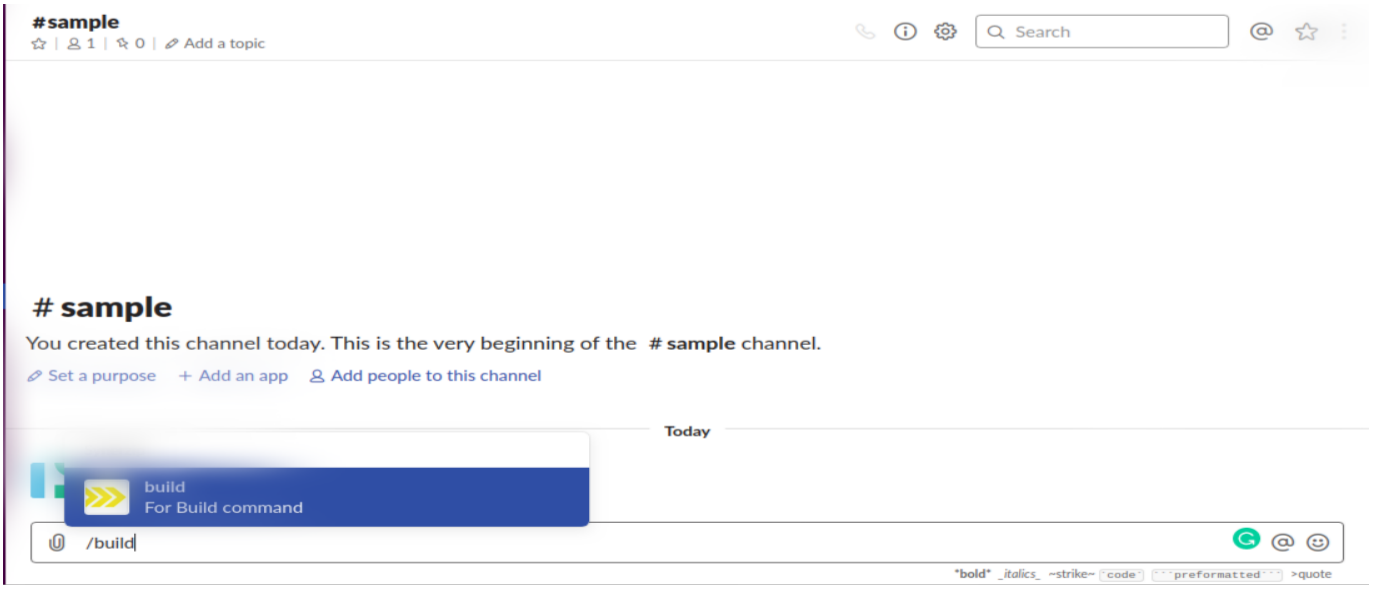
slash
command configuration,



Hint – For your understanding select Autocomplete help text. This would help the user see suggestions for your command.



Save your Integration. Now run the **/build** command from your Slack channel. This will start your Jenkins job from Slack.



Section 3 – Trigger a build for a specific branch or a parameterized Jenkins Job From Slack

In the Above section, I showed how we can trigger a Jenkins Job from within Slack. In this section, I will show how we can trigger a build for a specific branch or pass a parameter through Slack for a Jenkins Job.

For this, We need three plugins:

1. Pre SCM BuildStep Plugin
2. EnvInject Plugin
3. Build Authorization Token Root Plugin

In the above section, we installed and used the Build Authorization Token Root Plugin which will be useful now.

Go to Slack > Slash command configuration

I have configured my Slack slash command to call:

```
https://Jenkins_URL/job/Job_name/buildWithParameters?token=Your_Authentication_token
```

When I type

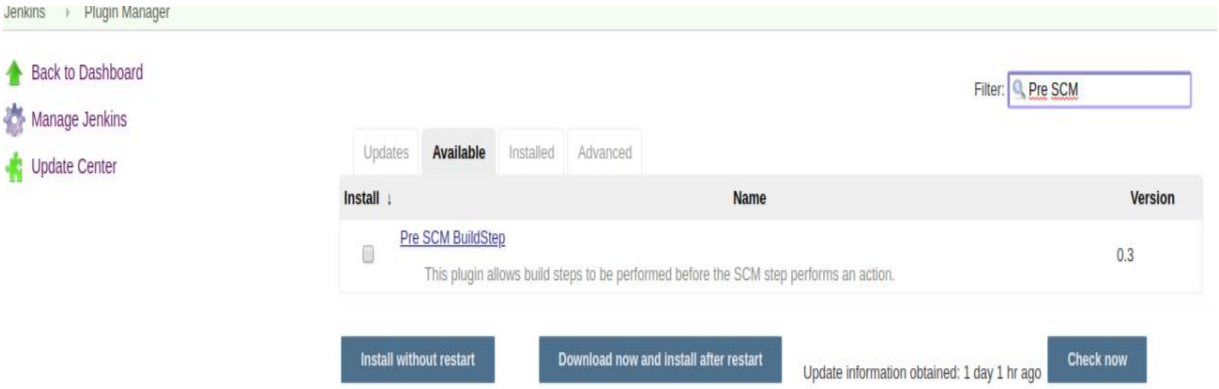
```
/build parameter1 parameter2
```

, the text I type after the slash command gets sent with the URL as a parameter (branch_name, dockertag in this case).

- `/build`
  – this is the command, the part that tells Slack to treat it as a Slash Command and has specific routing instructions.
- `feature/2348757 cicd-01`
  – this is the text portion, it includes everything after the first space following the command. It is treated as a single parameter that is passed to the app that owns the command.

The *text* contains a whole string (i.e *text* = *branch_name dockertag*).

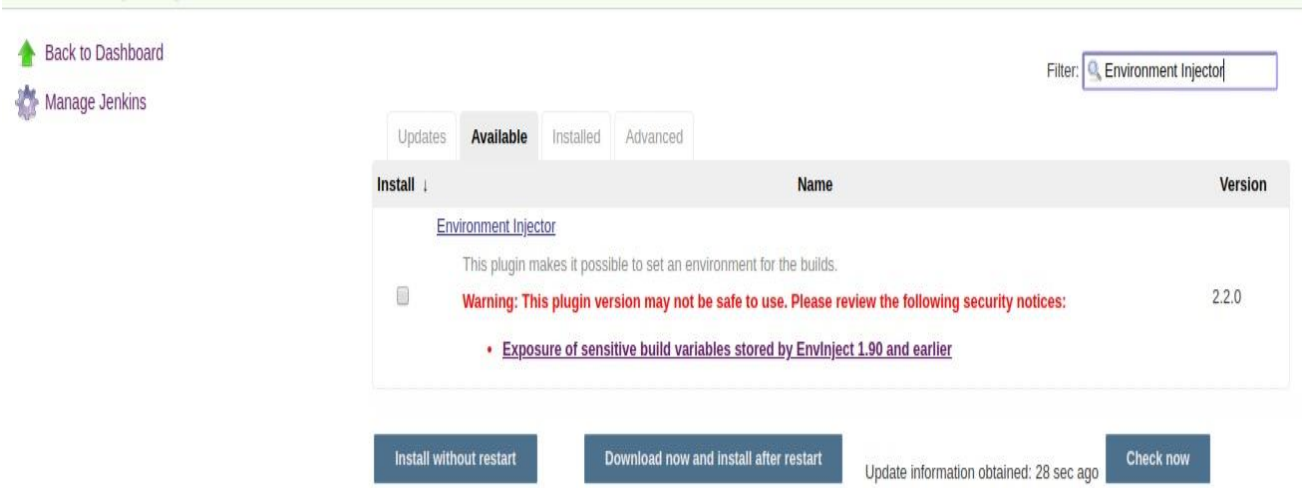
Hence we need to split this *'text'* parameter into multiple environment variables that Jenkins can use for the build process.
To achieve this we need to add two pre-SCM build steps and install the Pre-SCM Build Step plugin.
To do this, go to Jenkins > Manage Jenkins > Plugin Manager and search for the "Pre SCM BuildStep" plugin and install it.



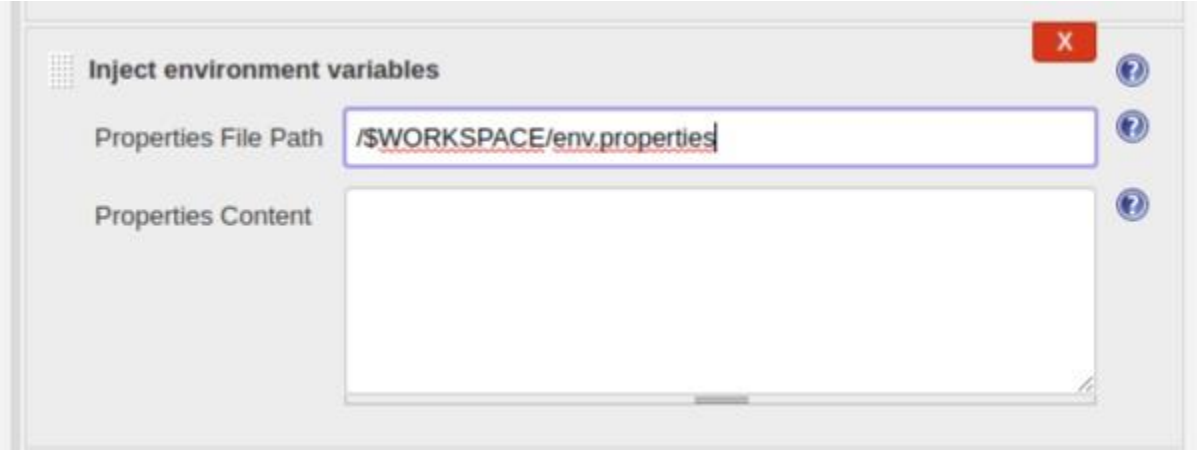
After installation,
Go to the Jenkins job and click on run *BuildStep before SCM runs* under 'On Build Environment'. Now click on Add Build Step > Execute shell command which consists of the following script:

```
build_parameters=($text)
echo BRANCH=${build_parameters[0]} > env.properties
echo param2=${build_parameters[1]} >> env.properties
```

'env.properties' is a file used to store the parsed strings. This script stores the environment variables in a key-value pair. This script stores a parameter in a string array where the parameters are split by the 'space' character. Parameters are read and stored in a variable.
Pre-SCM build step 2 is 'Inject Environment Variables'. For this, we need to install the "Inject Environment Variables" plugin.
Go to Jenkins > Manage Jenkins > Plugin Manager and search for "Environment Injector" plugin and install it.

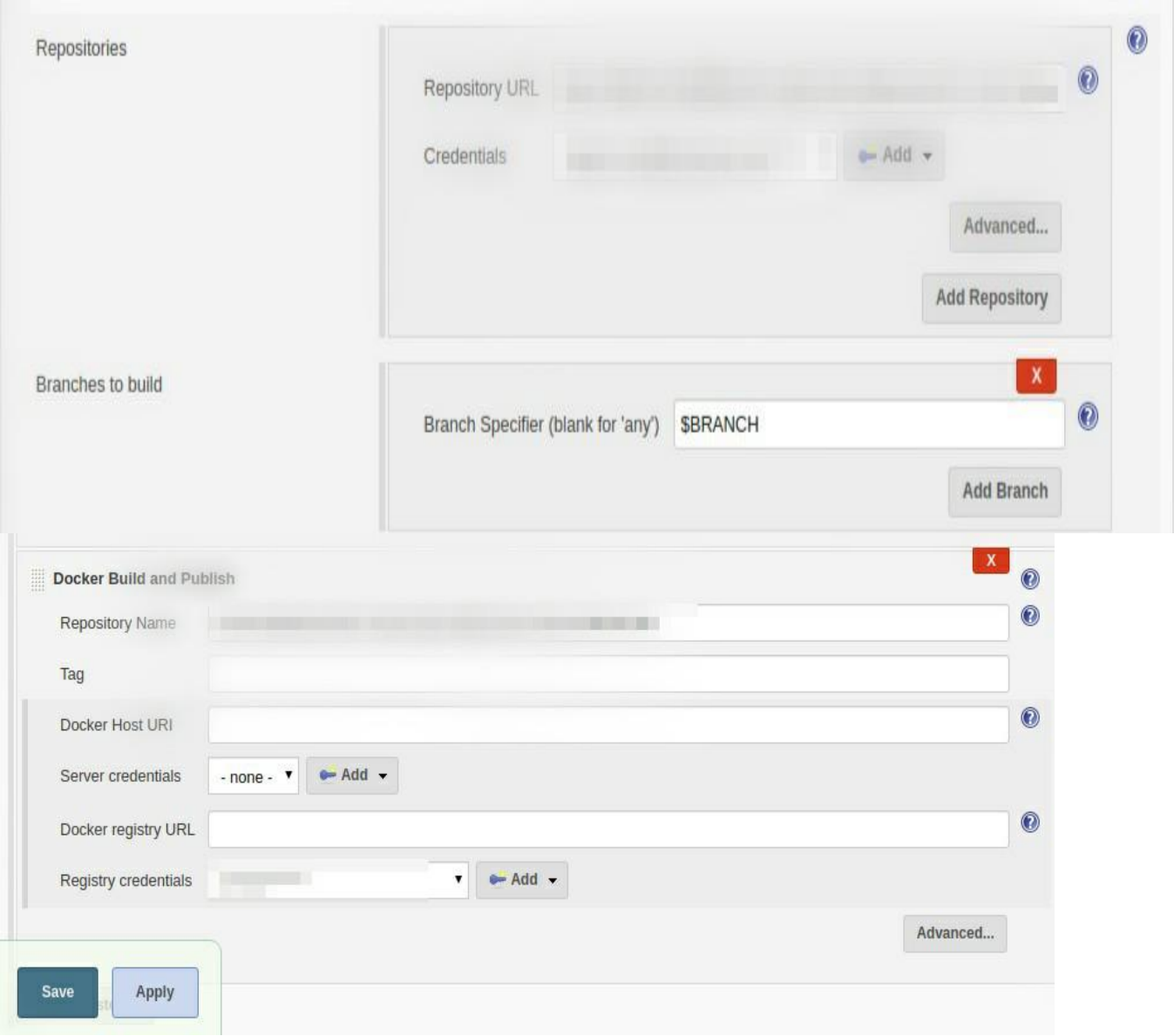


In the *Run build step before SCM runs* section, click on *Add Build Step* then on *Inject Environment Variables* and in the *Properties File Path* > Specify path as *$WORKSPACE/env.properties*.
I have created the properties file in a Workspace to read the file with my parsed environment variables and inject them in the job environment.



Set the Flag as parameterized > Create a String parameter.

Now, Specify the git repository (note that we're specifying the branch as $BRANCH, hence it will read the properties file and return a value of Branch (in this case, develop, master or feature). Also specify the docker tag as $PARAM2, to return the value of PARAM2.



The command

```
/build feature/2348757 cicd-01
```

in Slack starts the Jenkins job and triggers the build for specific branch and sets docker image tag