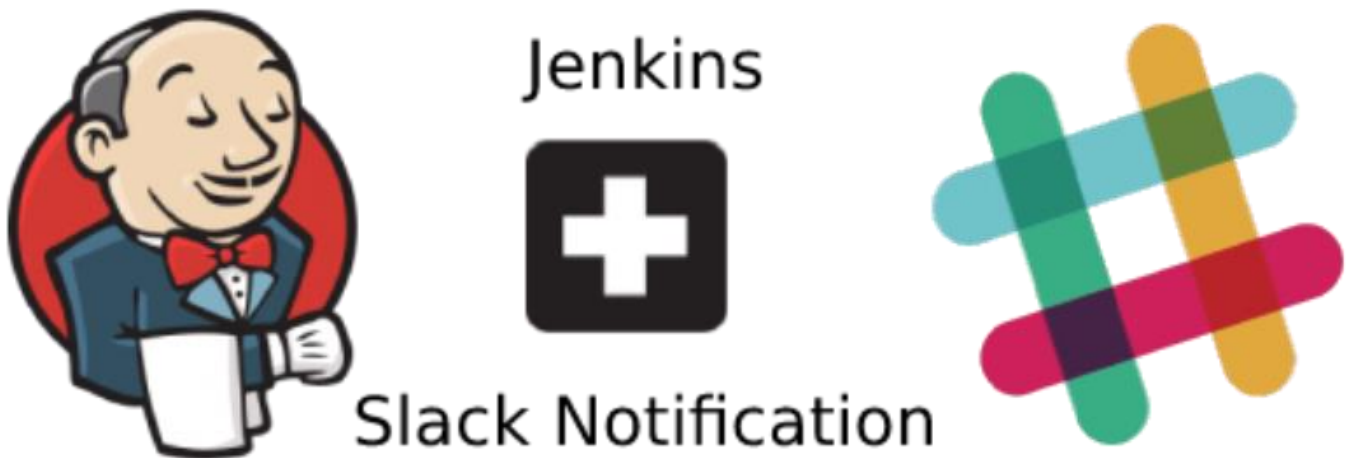


# Integrating Jenkins with Slack

## 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

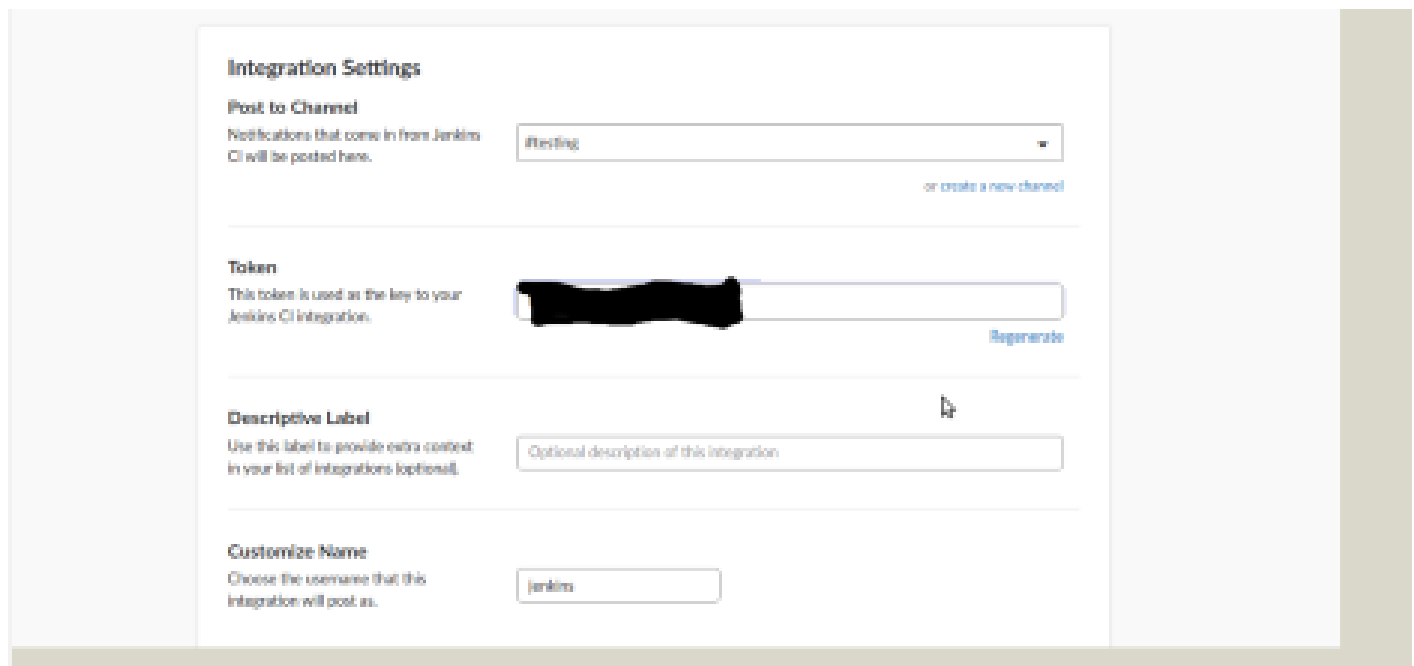
Integration Jenkins with slack

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>

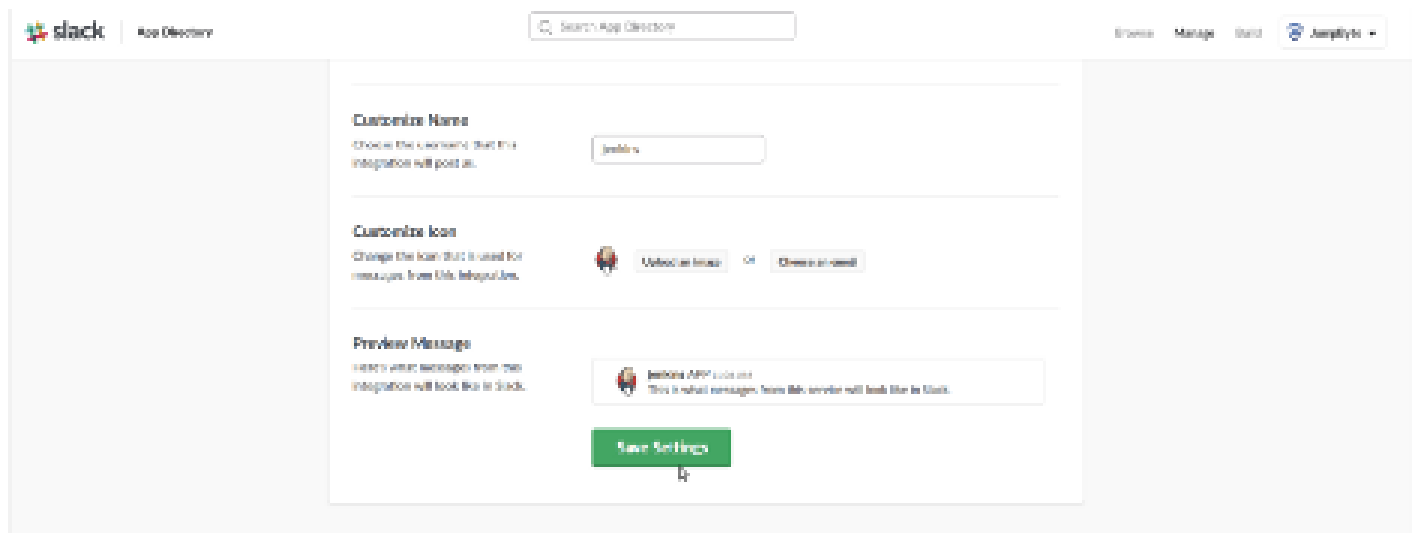


First, install 'Jenkins-ci' and then Add configuration and set channel and all thing like



The image shows the 'Integration Settings' form in Jenkins. It has four sections: 'Post to Channel' with a dropdown menu showing 'testing' and a link 'or create a new channel'; 'Token' with a masked token field and a 'Regenerate' link; 'Descriptive Label' with a text input field containing 'Optional description of this integration'; and 'Customize Name' with a text input field containing 'jenkins'.

Add channel name here and note Token.

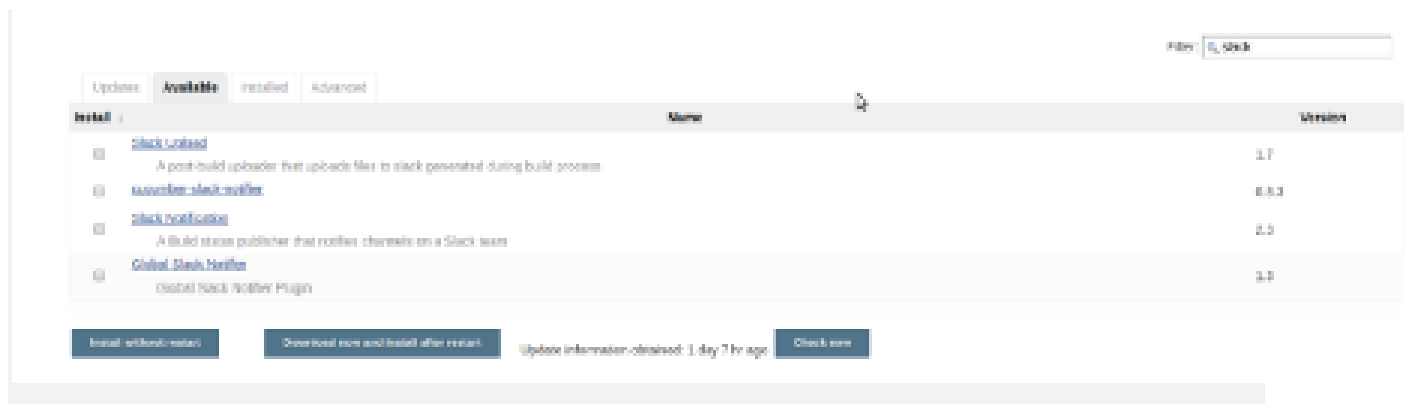


The image shows the 'Slack Integration Settings' form in Jenkins. It has three sections: 'Customize Name' with a text input field containing 'jenkins'; 'Customize Icon' with a dropdown menu showing 'Upload an image' and 'Choose an icon'; and 'Preview Message' with a text input field containing 'Jenkins APP build This is a build message from this service will look like in Slack.' Below the sections is a green 'Save Settings' button.

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.

The screenshot shows the Jenkins configuration page for a Slack integration. The 'Include Custom Message' checkbox is checked. The 'Custom Message' field contains the text: 'Please check. This is jenkins job result of [redacted] job.' The 'Notifier message includes' dropdown is set to 'commit list with authors only'. Below this, a label reads 'What commit information to include into notification message'. The 'Base URL' field is filled with 'https://[redacted].slack.com/services/hooks/jenkins-ci/'. The 'Team subdomain' field is filled with 'https://[redacted].slack.com'. The 'Integration Token' field is filled with a redacted token. A warning message below the token states: '⚠️ Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID'. The 'Integration Token Credential ID' dropdown is set to 'none', with an 'Add' button next to it. The 'Is bot user?' checkbox is checked. The 'Project Channel' field is filled with '#testing'. A 'Test Connection' button is located at the bottom right of the form.

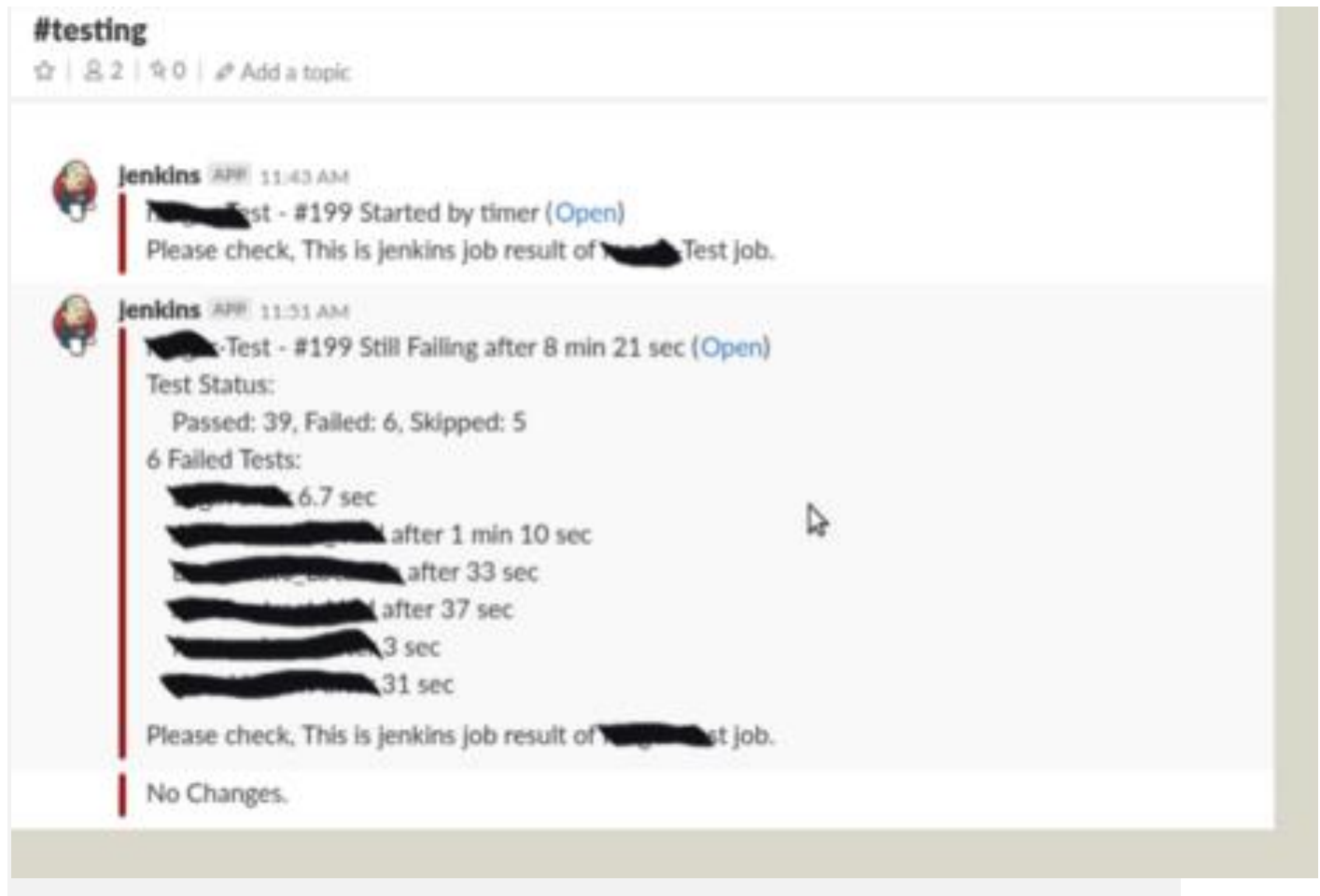
Enter Base URL, Team subdomain, Integration Token that created on you slack and Project Channel and click on apply and save button.

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.

The Jenkins Plugin Manager interface is shown. The top navigation bar includes the Jenkins logo, a red tab labeled '1', a search bar, and user links for 'srujan kumbha' and 'log out'. The breadcrumb trail shows 'Jenkins > Plugin Manager'. On the left sidebar, there are links for 'Back to Dashboard' and 'Manage Jenkins'. The main content area has a filter set to 'slack notification'. Below the filter are tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Available' tab is active, showing a table with columns 'Install', 'Name', and 'Version'. One plugin, 'Slack Notification', is listed with version 2.28. Below the table are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. A status message indicates 'Update information obtained: 3 min 30 sec ago'.

Install	Name	Version
<a href="#">Slack Notification</a>		2.28

This plugin is a Slack notifier that can publish build status to Slack channels.

Buttons: Install without restart, Download now and install after restart, Check now

Update information obtained: 3 min 30 sec ago

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’.

The Slack interface for a channel named '#sample' is shown. The channel header includes the name '#sample', a star icon, a count of 1 member and 0 topics, and an 'Add a topic' link. On the right, there are icons for a phone, info, settings, a search bar, and a menu with '@', a star, and a vertical ellipsis. A settings menu is open, displaying options: 'Jump to date...', 'Add people to #sample...', 'View channel details', 'Additional options...', 'Notification preferences...', 'Mute #sample', 'Add an app' (highlighted in blue), and 'Leave #sample'. Below the menu, the channel description reads: 'You created this channel today. This is the very beginning of the #sample channel.' At the bottom, there are links for 'Set a purpose', '+ Add an app', and 'Add people to this channel'.

#sample

☆ | 1 | 0 | Add a topic

Jump to date...  
Add people to #sample...  
View channel details  
Additional options...  
Notification preferences...  
Mute #sample  
**Add an app**  
Leave #sample

# sample


You created this channel today. This is the very beginning of the #sample channel.

Set a purpose + Add an app Add people to this channel

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

slack App Directory Search App Directory Learn Browse Manage Build

< Browse Apps



## Jenkins CI

App Info Settings

**This app was made by Slack.**  
This app was made by a member of the Slack team to help connect Slack with a third-party service; these apps may not be tested, documented, or supported by Slack in the way we support our core offerings, like Slack Enterprise Grid and Slack for Teams. You may provide feedback about these apps at [feedback@slack.com](mailto:feedback@slack.com).

It only uses data Slack already has access to (view our [Privacy Policy](#) to learn more). By enabling and/or using this app, you may be connecting with a service that is not part of Slack.


**Add Configuration**

App help

Terms

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.

slack App Directory Search App Directory Browse Manage Build



## Jenkins CI

An open source continuous integration server.

Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.

This integration will post build notifications to a channel in Slack.

**Post to Channel**  
Start by choosing a channel where Jenkins notifications will be posted.

# sample

or [create a new channel](#)

**Add 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).

**Global Slack Notifier Settings**

Slack compatible app URL (optional)	<input type="text"/>	?
Team Subdomain	<input type="text"/>	?
Integration Token Credential ID	<div>notification purpose <input type="button" value="Add"/></div>	?
Is Bot User?	<input type="checkbox"/>	?
Send as Text?	<input type="checkbox"/>	?
Icon Emoji	<input type="text"/>	?
Username	<input type="text"/>	?
Channel or Slack ID	<input type="text" value="sample"/>	?

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.

**Slack Notifications**

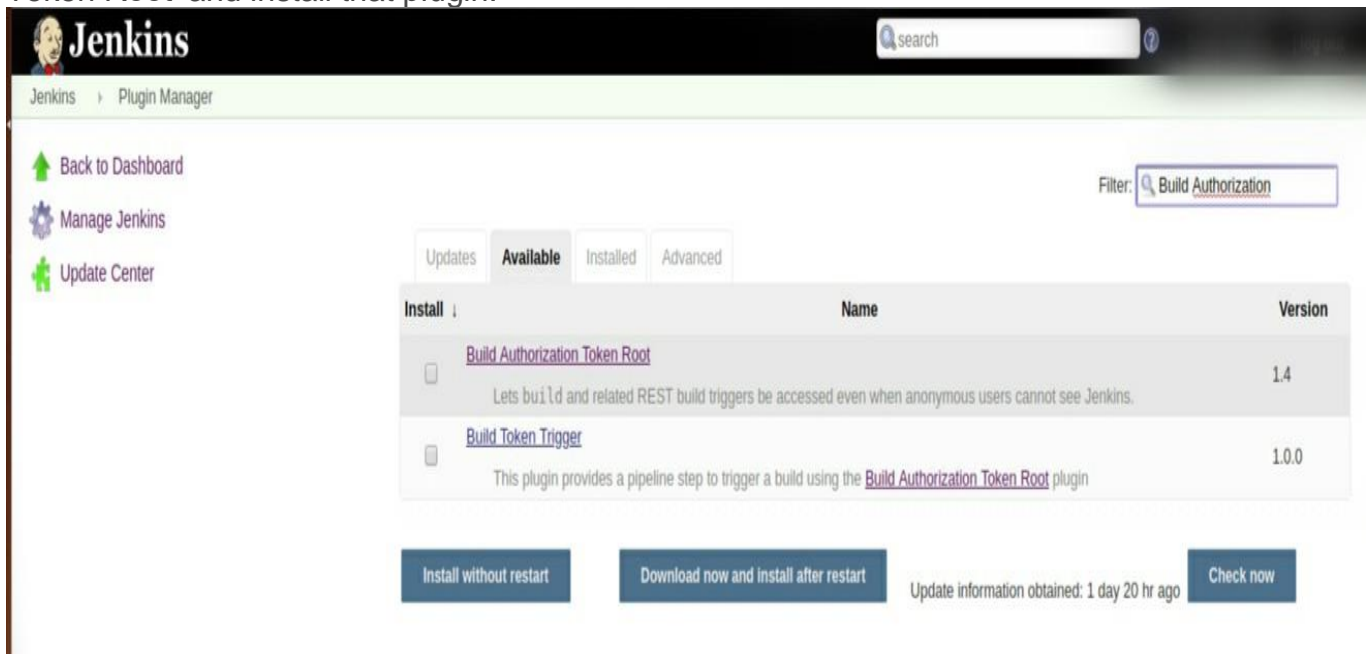
Notify Build Start	<input checked="" type="checkbox"/>
Notify Success	<input checked="" type="checkbox"/>
Notify Aborted	<input type="checkbox"/>
Notify Not Built	<input type="checkbox"/>
Notify Unstable	<input type="checkbox"/>
Notify Regression	<input type="checkbox"/>
Notify Every Failure	<input checked="" type="checkbox"/>
Notify Back To Normal	<input type="checkbox"/>

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

-  **Jenkins** APP 3:28 PM  
Centric - #165 Started by remote host 18.232.167.25 ([Open](#))
-  **Jenkins** APP 3:50 PM  
Centric - #166 Started by remote host 54.147.20.92 ([Open](#))
-  **Jenkins** APP 3:59 PM  
Centric - 1.0.41 centricsoftware/centric-cloud-saas-webservices-internal:ci-cd-02  
Success after 9 min 45 sec ([Open](#))

## 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.



The screenshot shows the Jenkins Plugin Manager interface. The left sidebar contains links: 'Back to Dashboard', 'Manage Jenkins', and 'Update Center'. The main area displays a table of available plugins. A search filter 'Build Authorization' is applied. The table lists two plugins: 'Build Authorization Token Root' (version 1.4) and 'Build Token Trigger' (version 1.0.0). Below the table are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. The update information is noted as '1 day 20 hr ago'.

Install	Name	Version
<input type="checkbox"/>	<a href="#">Build Authorization Token Root</a> Lets build and related REST build triggers be accessed even when anonymous users cannot see Jenkins.	1.4
<input type="checkbox"/>	<a href="#">Build Token Trigger</a> This plugin provides a pipeline step to trigger a build using the <a href="#">Build Authorization Token Root</a> plugin	1.0.0

Buttons: [Install without restart](#), [Download now and install after restart](#), [Check now](#)

Update information obtained: 1 day 20 hr ago

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".

## Build Triggers

☒ Trigger builds remotely (e.g., from scripts)

Authentication Token

Use the following URL to trigger build remotely: JENKINS\_URL/job/Centric%20Test%20Project/build?token=TOKEN\_NAME or /buildWithParameters?token=TOKEN\_NAME  
Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

Use the following URL to trigger the build:

JENKINS\_URL/job/SampleJob/build?token=TOKEN\_NAME

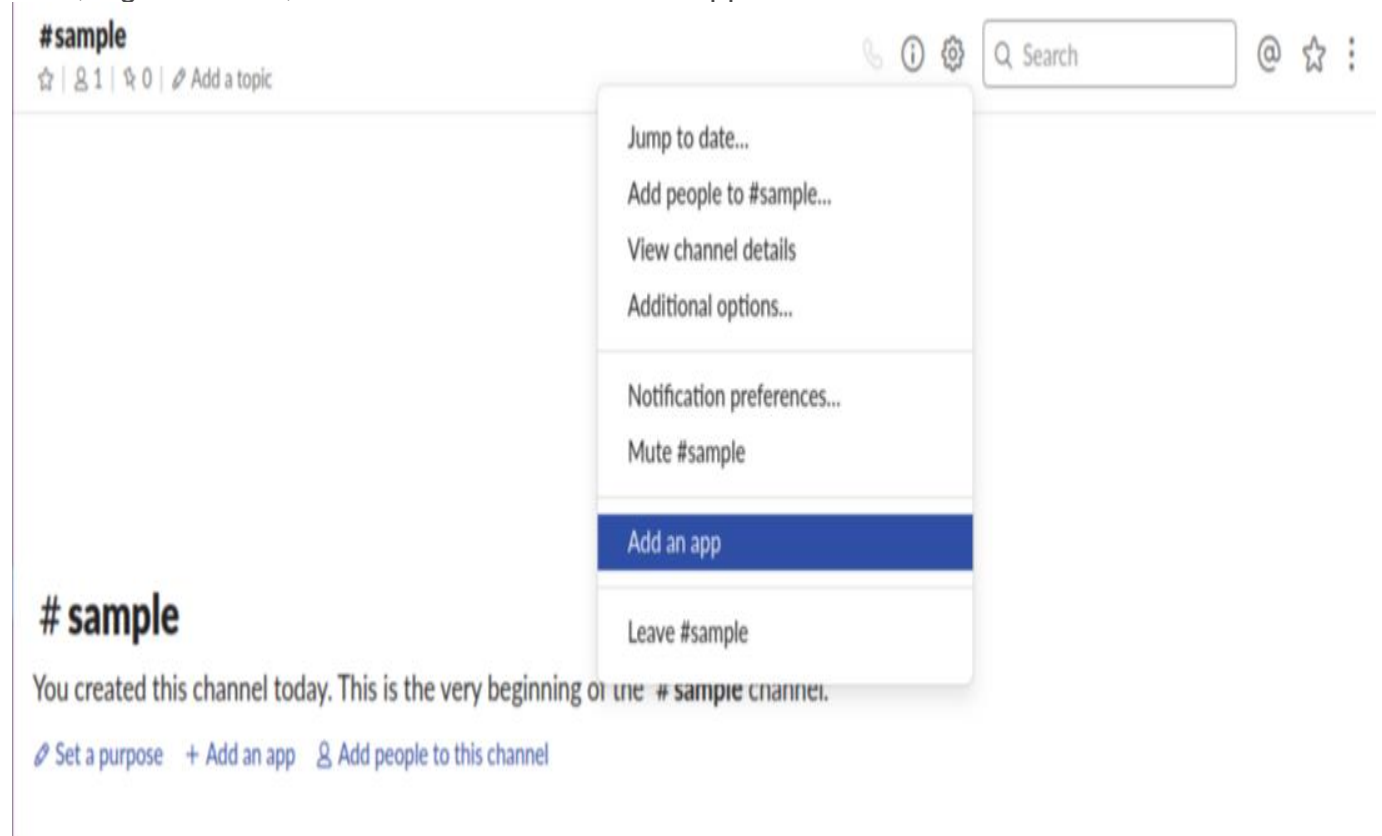
OR

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.

[< Browse Apps](#)[Add Configuration](#)[App help](#)[Terms](#)

## Slash Commands

[App Info](#) [Settings](#)

This app was made by Slack.

This app was made by a member of the Slack team to help connect Slack with a third-party service; these apps may not be tested, documented, or supported by Slack in the way we support our core offerings, like Slack Enterprise Grid and Slack for Teams. You may provide feedback about these apps at [feedback@slack.com](mailto:feedback@slack.com).

It only uses data Slack already has access to (view our [Privacy Policy](#) to learn more). By enabling and/or using this app, you may be connecting with a service that is not part of Slack.

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.



## Slash Commands

Customized Slack commands for your workspace.

Slash Commands allow you to listen for custom triggers in chat messages across all Slack channels. When a Slash Command is triggered, relevant data will be sent to an external URL in real-time.

For example, typing `/weather 94070` could send a message to an external URL that would look up the current weather forecast for San Francisco and post it back to Slack.

Choose a Command

Commands should be a single word (and can't be blank). Example: /deploy

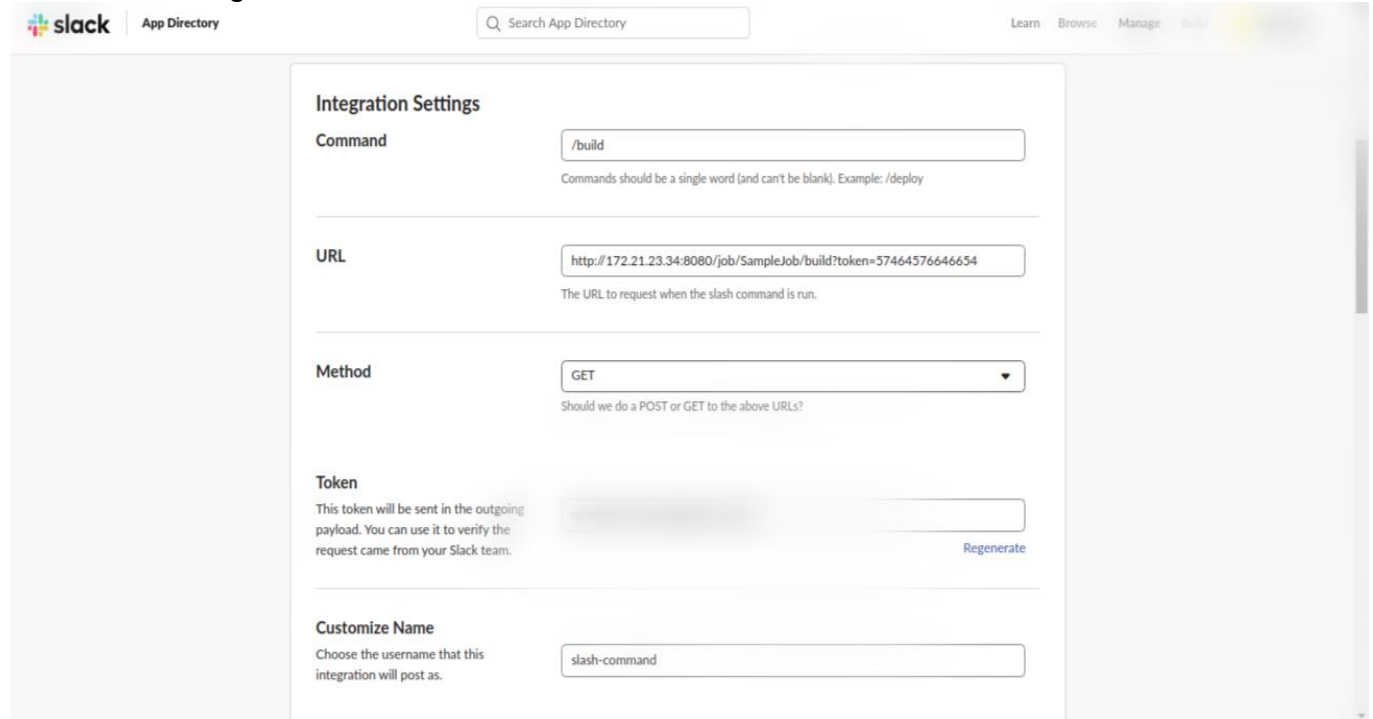
[Add Slash Command Integration](#)

By creating a slash command, you agree to the [Slack API Terms of Service](#).

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,



The screenshot shows the 'Integration Settings' form for a Slack app. The form is titled 'Integration Settings' and contains several sections:

- Command:** A text input field containing '/build'. Below it, a note states: 'Commands should be a single word (and can't be blank). Example: /deploy'.
- URL:** A text input field containing 'http://172.21.23.34:8080/job/SampleJob/build?token=57464576646654'. Below it, a note states: 'The URL to request when the slash command is run.'
- Method:** A dropdown menu currently set to 'GET'. Below it, a note states: 'Should we do a POST or GET to the above URLs?'.
- Token:** A text input field containing a blurred token. Below it, a note states: 'This token will be sent in the outgoing payload. You can use it to verify the request came from your Slack team.' To the right of the input field is a 'Regenerate' link.
- Customize Name:** A text input field containing 'slash-command'. Below it, a note states: 'Choose the username that this integration will post as.'

The form is part of the Slack App Directory interface, with the Slack logo and 'App Directory' text in the top left, and a search bar and navigation links ('Learn', 'Browse', 'Manage', 'Build') in the top right.

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

slack App Directory Search App Directory Learn Browse Manage Build Synerzip

integration will look like in Slack.

**slasn-command** API 4:32 PM  
This is what messages from this service will look like in Slack.

### Autocomplete help text

You can add this slash command to the autocomplete list and add some usage hints.

Commands matching `/feedb`

Slack

`/feedback` [your message] Send feedback to Slack

+ `/feedback`

☒ Show this command in the autocomplete list

**Description**

For Build command

A short description of what this slash command does.

**Usage hint**

List any parameters that can be passed.

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


#sample ☆ | 👤 1 | ✖ 0 | Add a topic 🔍 Search @ ☆ ⋮

#sample

You created this channel today. This is the very beginning of the #sample channel.

🔗 Set a purpose + Add an app 👤 Add people to this channel

Today

 build  
For Build command

🔗 /build

\*bold\* \_italics\_ ~strike~ `code` `preformatted` >quote

## 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

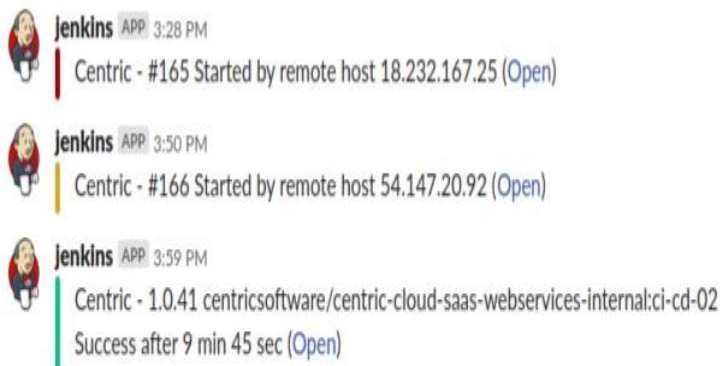
I have configured my Slack slash command to call:

```
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).



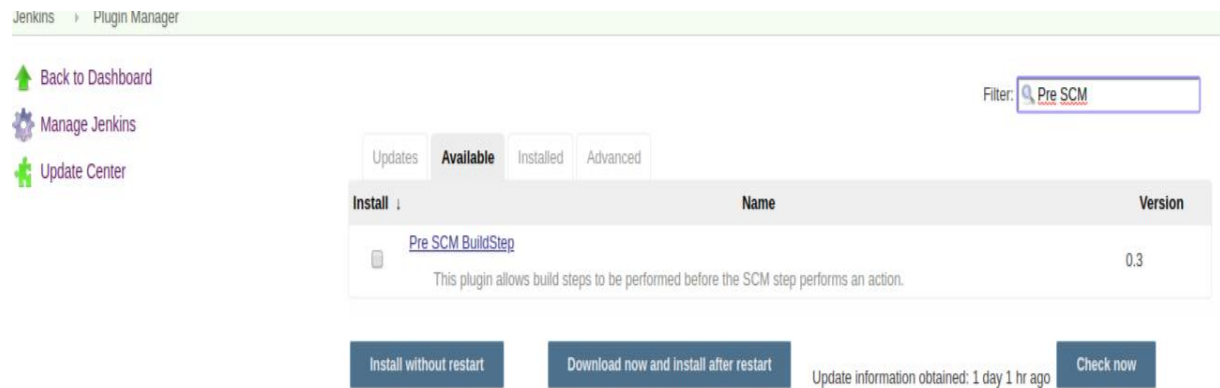
- /build
  - this is the command, the part that tells Slack to treat it as a Slash Command and has specific routing instructions.
- feature/2348757 ccd-01
  - this is the text portion, it includes everything after the first space following the command. It is treated as a single parameter that is passed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



[Back to Dashboard](#)

[Manage Jenkins](#)

Filter:

Updates Available Installed Advanced

Install	Name	Version
<a href="#">Environment Injector</a>	This plugin makes it possible to set an environment for the builds. <b>Warning: This plugin version may not be safe to use. Please review the following security notices:</b> <ul style="list-style-type: none"><li>• <a href="#">Exposure of sensitive build variables stored by EnvInject 1.90 and earlier</a></li></ul>	2.2.0

Install without restart

Download now and install after restart

Update information obtained: 28 sec ago

Check now

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.

**Inject environment variables**

Properties File Path:

Properties Content:

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`.

Repositories

Repository URL

Credentials

Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

\$BRANCH

Add Branch

Docker Build and Publish

Repository Name

Tag

Docker Host URI

Server credentials

- none -

Add

Docker registry URL

Registry credentials

Add

Advanced...

Save

Apply

The command

```
/build feature/2348757 cisd-01
```

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