

How to use Webhooks in Jenkins

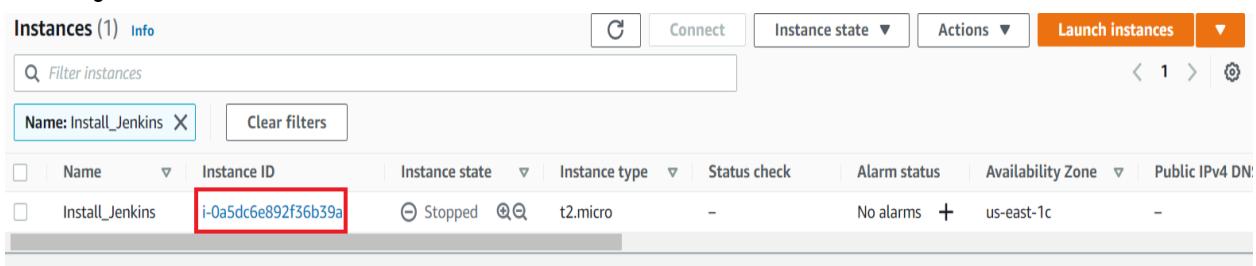
This document covers the instructions on how to integrate Jenkins with Git using Webhooks to trigger continuous downloading whenever any change is made in the code. Furthermore, this document is divided into the following three parts:

1. Opening port no 8080 for GitHub
2. Setting up WebHooks
3. Creating a freestyle job and triggering WebHooks

Opening port no 8080 for GitHub

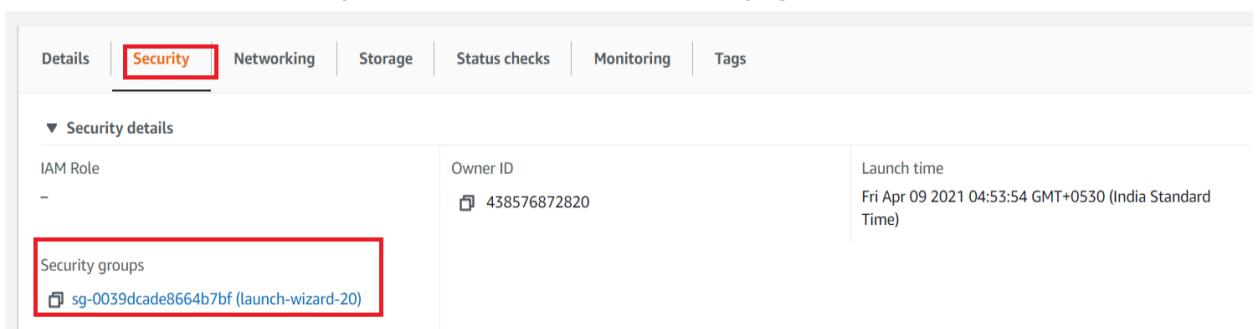
So far, we have been setting port 8080 to **Myip**, but for working with webhooks, we need to open port 8080 for all traffic in the security groups. So, let's take a look at the steps involved.

1. Go to your instance over the AWS dashboard as shown.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DN
Install_Jenkins	i-0a5dc6e892f36b39a	Stopped	t2.micro	-	No alarms	us-east-1c	-

2. Then, click on **Instance Id**.
3. Next, click on **Security** and then select **Security groups**.



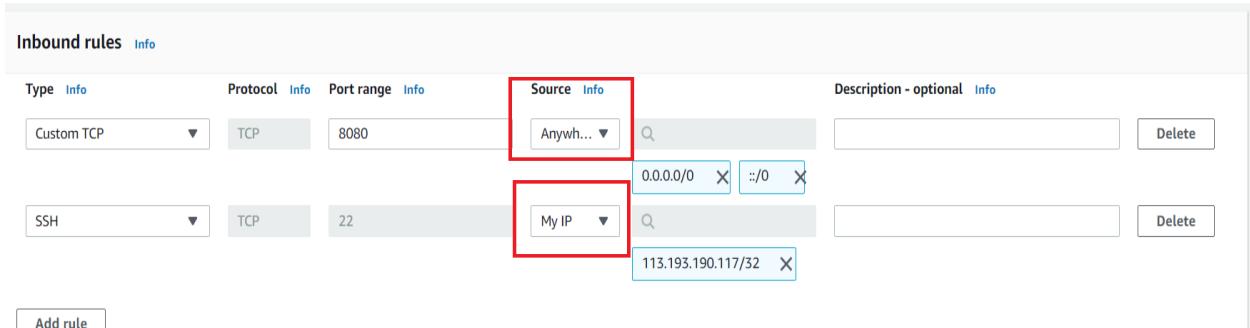
IAM Role	Owner ID	Launch time
-	438576872820	Fri Apr 09 2021 04:53:54 GMT+0530 (India Standard Time)

4. Click on **Edit inbound rules**.



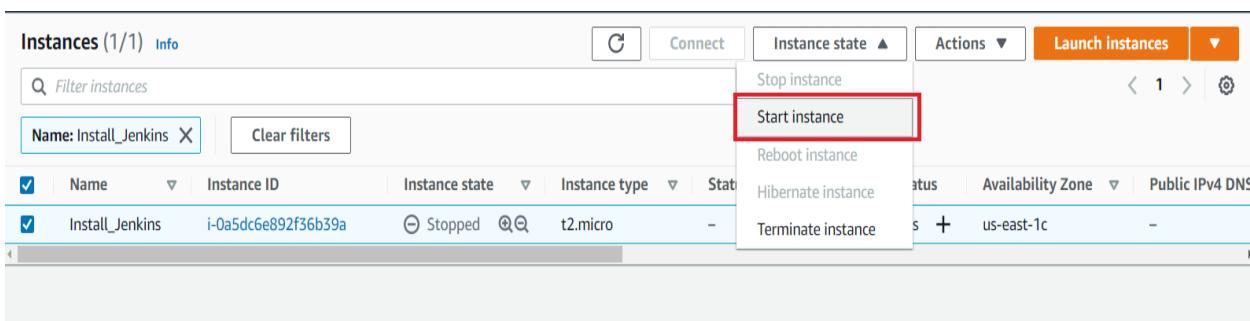
Inbound rules (3)				
Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8080	Anywhere	
SSH	TCP	22	My IP	113.193.190.117/32

5. Then, for port 8080, select **Anywhere** in the **Source**.
6. For port 22, it needs to be **My Ip**.



Inbound rules Info				
Type	Protocol	Port range	Source	Description - optional
Custom TCP	TCP	8080	Anywhere	
SSH	TCP	22	My IP	113.193.190.117/32

7. After that, click on **Save Rules**.
8. Then, come back to your instance dashboard from where you started.
9. Select **Your Instance**.
10. Click on **Instance State** and then select **Start Instance**.



Instances (1/1) Info						
Filter instances		Actions		Launch instances		
Name	Instance ID	Instance state	Instance type	Status	Availability Zone	Public IPv4 DNS
Install_Jenkins	i-0a5dc6e892f36b39a	Stopped	t2.micro	-	us-east-1c	-

11. Once your machine comes in the running condition, log in to your machine using putty and go to the terminal.
12. Then, type the command **sudo service jenkins start**.
13. Run the **sudo service jenkins status** to check whether Jenkins has started.

```

ubuntu@ip-172-31-27-198:~$ sudo service jenkins start
ubuntu@ip-172-31-27-198:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
  Loaded: loaded (/etc/init.d/jenkins; generated)
  Active: active (exited) since Fri 2021-04-09 00:32:43 UTC; 6min ago
    Docs: man:systemd-sysv-generator(8)
   Process: 424 ExecStart=/etc/init.d/jenkins start (code=exited, status=
              0)

Apr 09 00:32:40 ip-172-31-27-198 systemd[1]: Starting LSB: Start Jenkins a
Apr 09 00:32:41 ip-172-31-27-198 jenkins[424]: Correct java version found
Apr 09 00:32:41 ip-172-31-27-198 jenkins[424]: * Starting Jenkins Automat
Apr 09 00:32:42 ip-172-31-27-198 su[605]: (to jenkins) root on none
Apr 09 00:32:42 ip-172-31-27-198 su[605]: pam_unix(su-l:session): session
Apr 09 00:32:42 ip-172-31-27-198 su[605]: pam_unix(su-l:session): session
Apr 09 00:32:43 ip-172-31-27-198 jenkins[424]: ...done.
Apr 09 00:32:43 ip-172-31-27-198 systemd[1]: Started LSB: Start Jenkins at
ubuntu@ip-172-31-27-198:~$ 
  
```

14. Now, to access Jenkins, copy the public URL of your Ubuntu instance.

Instance summary for i-0a5dc6e892f36b39a (Install_Jenkins)		Info		Connect	Instance state ▾
Updated less than a minute ago					
Instance ID	i-0a5dc6e892f36b39a (Install_Jenkins)	Public IPv4 address	107.20.2.171 open address	Private IPv4 addresses	172.31.27.198
Instance state	Running	Public IPv4 DNS	ec2-107-20-2-171.compute-1.amazonaws.com open address	Private IPv4 DNS	ip-172-31-27-198.ec2.internal
Instance type	t2.micro	Elastic IP addresses	-	VPC ID	vpc-b142ecc
AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	IAM Role	-	Subnet ID	subnet-495f7204

15. Then, go to the browser and paste <url:8080> and press enter.



16. After that, provide the username and password.



Welcome to Jenkins!

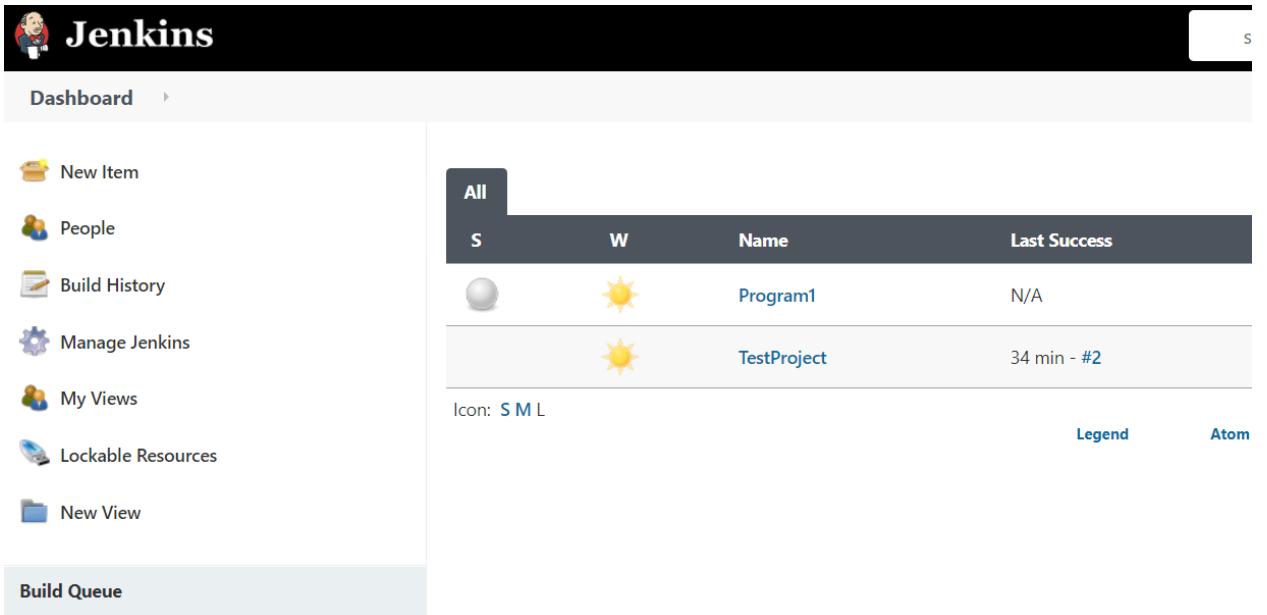
admin

.....

Sign in

Keep me signed in

17. Then, you will be directed to the Jenkins dashboard.



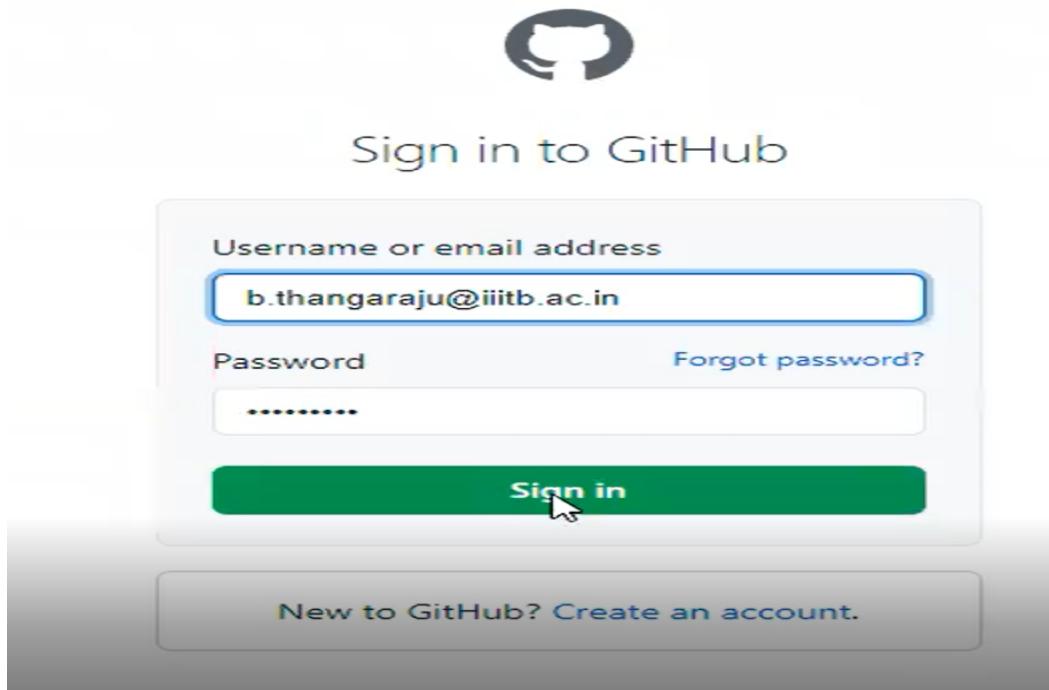
The screenshot shows the Jenkins dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. Below this is a 'Build Queue' section. The main area has a table titled 'All' showing build status for 'Program1' and 'TestProject'. The table columns are S (Status), W (Last Success), Name, and Last Success. Program1 has a grey circle icon, a yellow sun icon, and 'N/A' under Last Success. TestProject has a yellow sun icon, 'TestProject' under Name, and '34 min - #2' under Last Success. At the bottom right of the dashboard are 'Legend' and 'Atom' buttons.

S	W	Name	Last Success
grey circle	yellow sun	Program1	N/A
	yellow sun	TestProject	34 min - #2

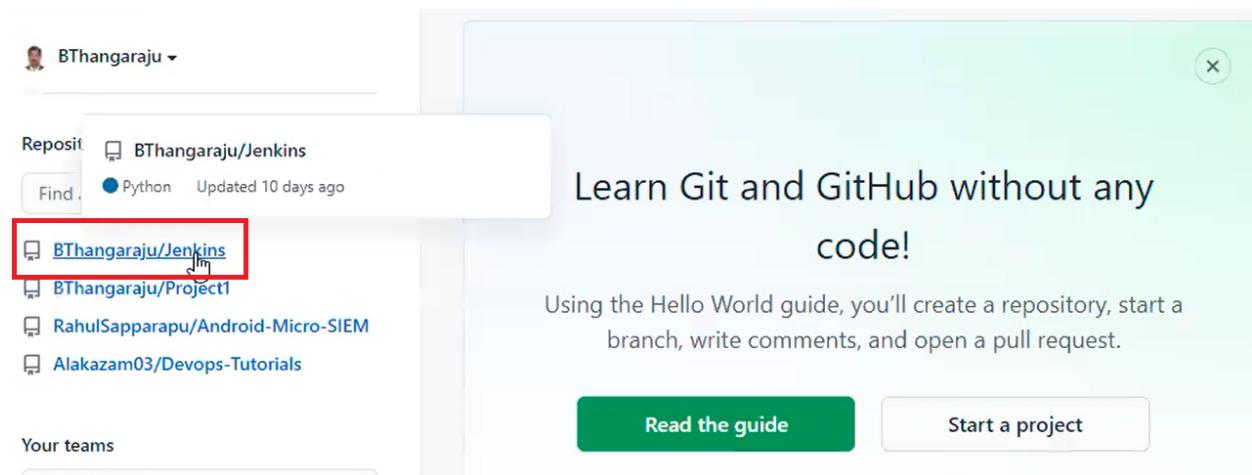
Setting up WebHooks

So, let's take a look at the steps involved in setting up WebHooks in Jenkins.

1. Sign in to the GitHub account.



2. Now, select the repository where you have kept your code.



BThangaraju

Repository BThangaraju/Jenkins

Find

BThangaraju/Jenkins

BThangaraju/Project1

RahulSapparapu/Android-Micro-SIEM

Alakazam03/Devops-Tutorials

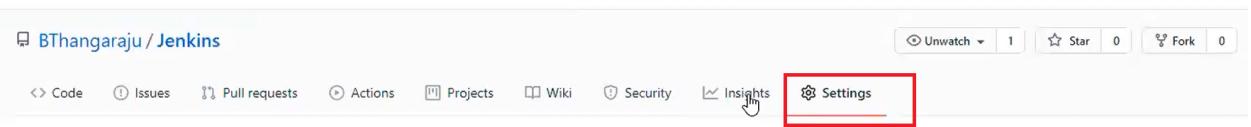
Your teams

Learn Git and GitHub without any code!

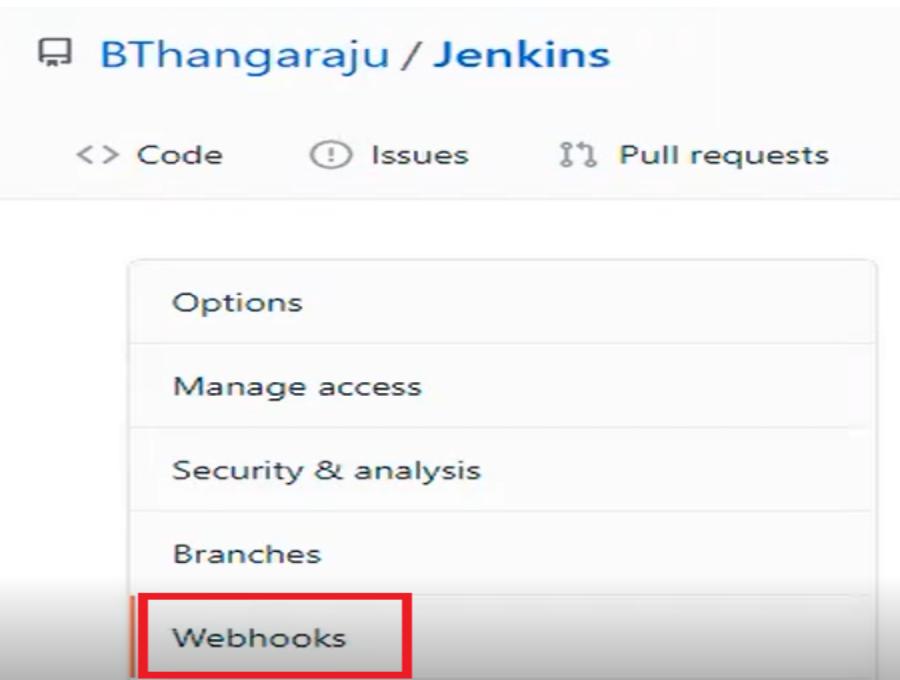
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide Start a project

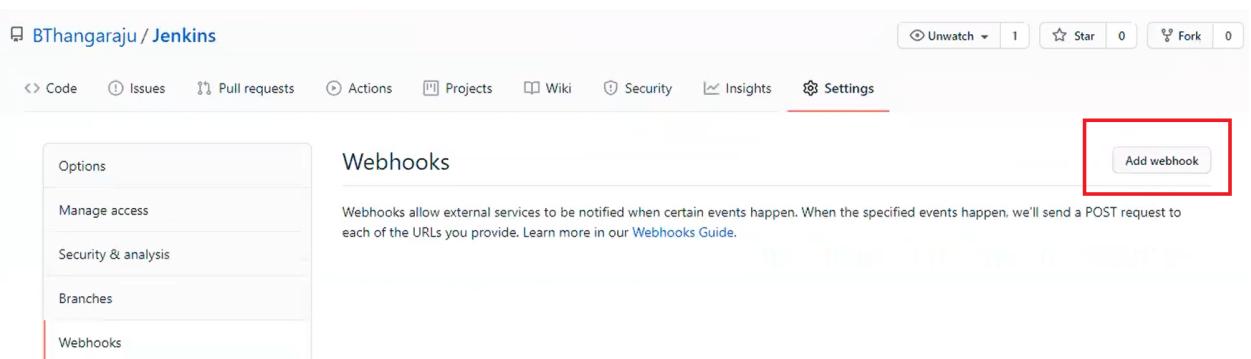
3. Then, click on **Settings**.



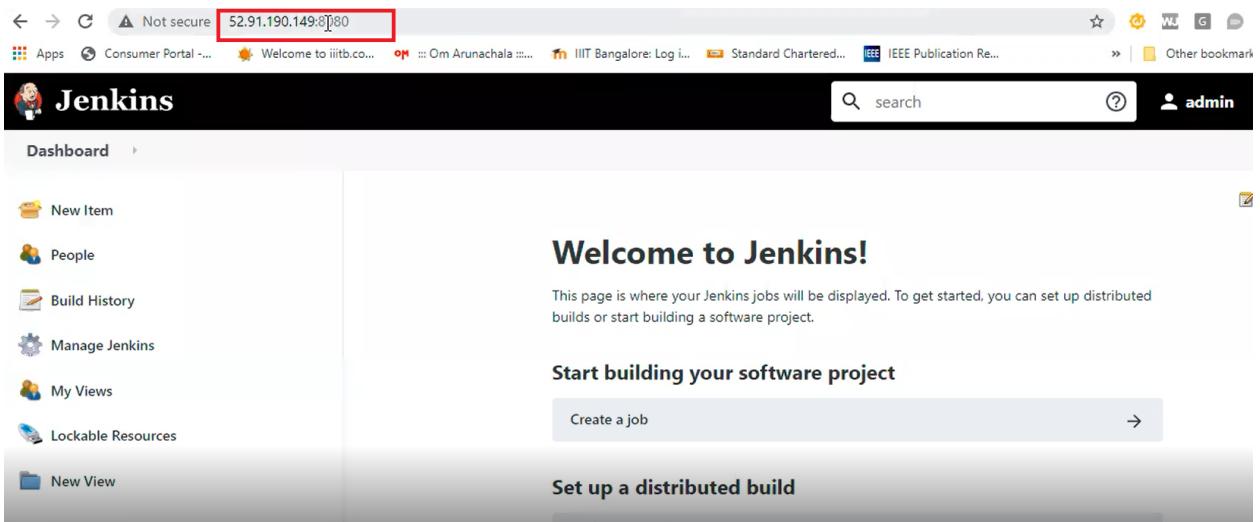
4. Then, select **Webhooks** from the menu on the left.



5. Then, click on **Add Webhooks** as shown in the screenshot given below.

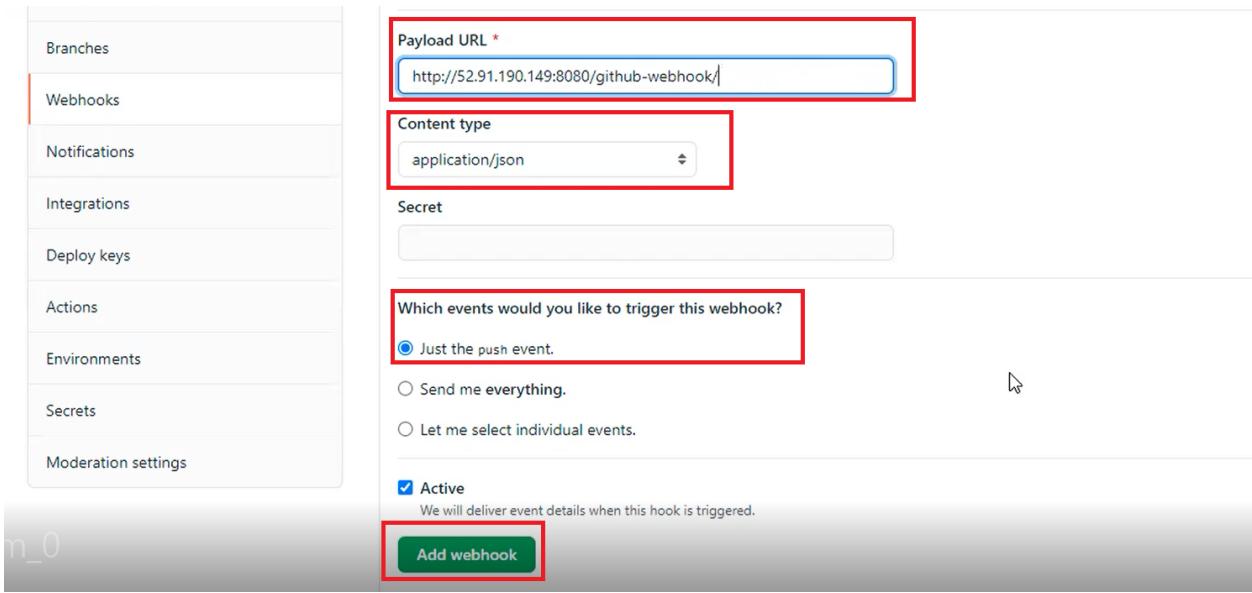


6. We need to provide a **Payload URL**. So, go to the Jenkins dashboard and copy the URL as shown in the screenshot given below.



The screenshot shows the Jenkins dashboard at the URL 52.91.190.149:8080. The dashboard features a sidebar with options like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main area displays a 'Welcome to Jenkins!' message, instructions for starting a build, and a 'Create a job' button. A 'Set up a distributed build' section is also present.

7. Next, come back to GitHub and paste the URL. Remember that you need to append **/github-webhook/** after the URL.
8. Then, select the **content type** and select **application/json**.
9. In the **Which events would you like to trigger the webhook?** option, select **Just the push event**.
10. After that, click on **Add Webhook** as shown in the screenshot given below.

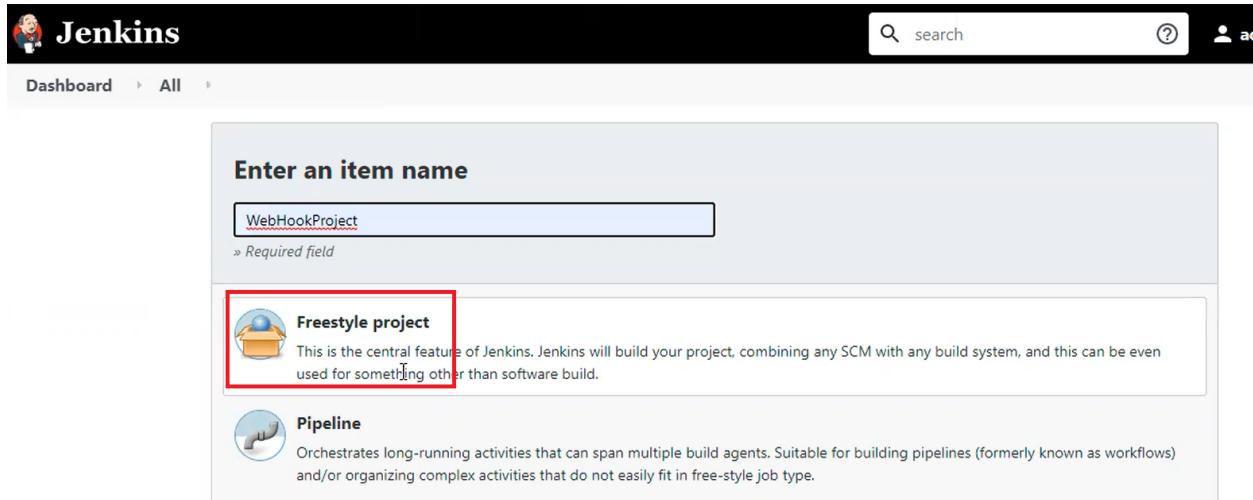


The screenshot shows the GitHub webhook configuration interface. On the left, a sidebar lists 'Branches', 'Webhooks' (which is selected), 'Notifications', 'Integrations', 'Deploy keys', 'Actions', 'Environments', 'Secrets', and 'Moderation settings'. The main form has fields for 'Payload URL *' containing 'http://52.91.190.149:8080/github-webhook/', 'Content type' set to 'application/json', and a 'Secret' input field. Below these, a section titled 'Which events would you like to trigger this webhook?' contains three radio buttons: 'Just the push event.' (selected), 'Send me everything.', and 'Let me select individual events.'. At the bottom, a checkbox 'Active' is checked with the note 'We will deliver event details when this hook is triggered.' A large green 'Add webhook' button is at the bottom right, which is highlighted with a red box.

Creating a freestyle project and trigger WebHooks

So far, we have set up webhooks. Now, you will learn about the steps that you need to follow to trigger webhooks.

- Come back to the **Jenkins Dashboard** and create a new freestyle job. Here, we created a job named WebhookProject.



The screenshot shows the Jenkins dashboard with a search bar and user account icon at the top right. Below it, a navigation bar has 'Dashboard' and 'All' selected. A central panel titled 'Enter an item name' contains a text input field with 'WebHookProject' typed in, followed by a note '» Required field'. Two options are listed: 'Freestyle project' (selected and highlighted with a red box) and 'Pipeline'. The 'Freestyle project' description states: 'This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.' The 'Pipeline' description states: 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.'

- In the **Source Code Management**, select **Git** and provide the **Repository URL** as shown in the screenshot given below.



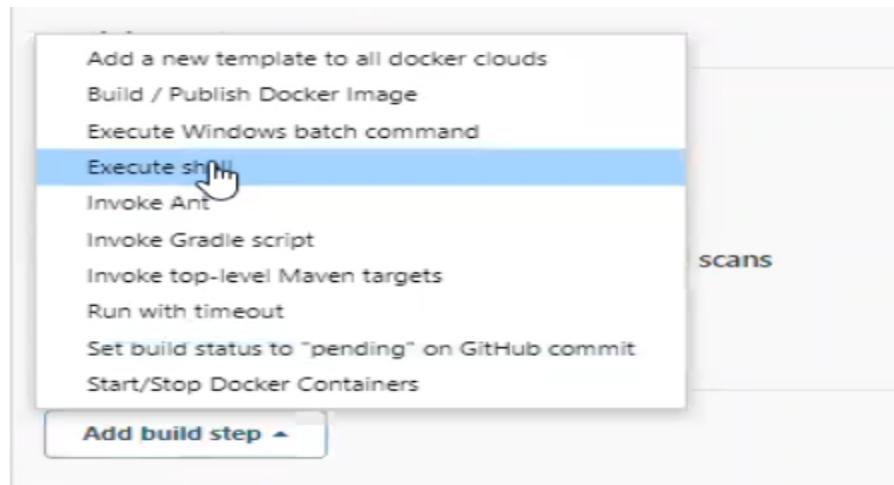
The screenshot shows the 'Source Code Management' section of a Jenkins job configuration. It includes a radio button for 'None' and one for 'Git' (which is selected and highlighted with a red box). Below this is a 'Repositories' section with a 'Repository URL' input field containing 'https://github.com/BThangaraju/Jenkins.git'. A red box highlights this field, and a red-bordered error message 'Please enter Git repository.' is displayed below it. Other sections like 'Credentials' and buttons for 'Advanced...' and 'Add Repository' are also visible.

- Scroll down, and in the **Build Triggers** option, select the **GitHub hook trigger** for the **GitScm polling** option.

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

4. Scroll down, and in the **Build** section, select the **Execute shell** option.



5. Then, write the script that you will execute.

Build

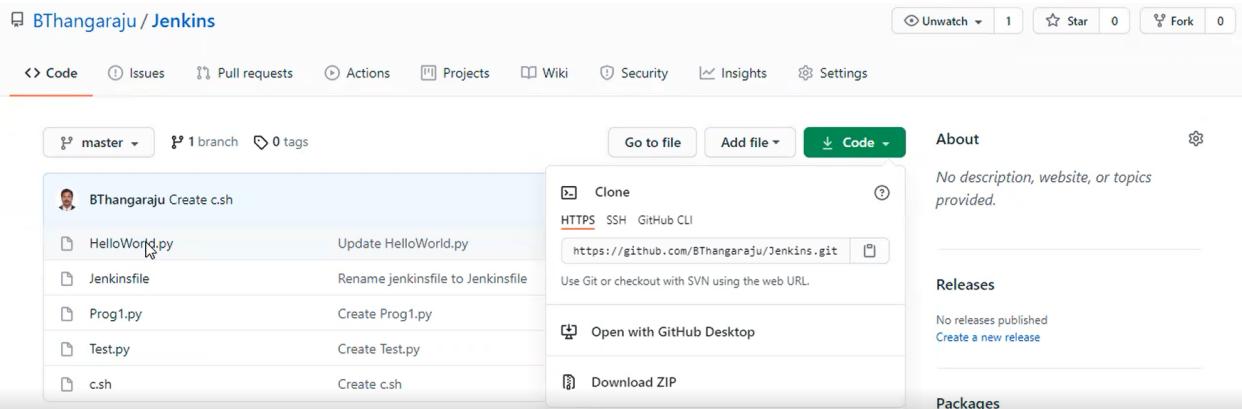
Execute shell

Command
`./HelloWorld`

See the list of [global environment variables](#)

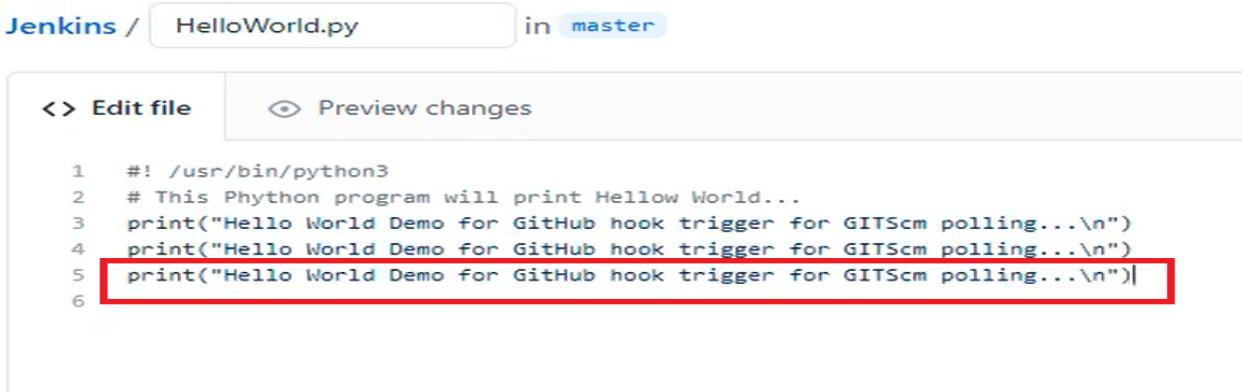
Save
Apply
[Advanced...](#)

6. Click on **Apply** and then **Save**.
 7. To trigger Webhooks, you need to make some changes. For this, go back to the GitHub repository.



The screenshot shows a GitHub repository page for 'BThangaraju/Jenkins'. The 'master' branch is selected, showing 1 branch and 0 tags. The repository contains several files: 'HelloWorld.py', 'Jenkinsfile', 'Prog1.py', 'Test.py', and 'c.sh'. A context menu is open over 'HelloWorld.py', with options like 'Clone', 'HTTPS', 'SSH', 'GitHub CLI', 'Use Git or checkout with SVN using the web URL', 'Open with GitHub Desktop', and 'Download ZIP'. To the right, there's an 'About' section with a note: 'No description, website, or topics provided.', a 'Releases' section with a note: 'No releases published. Create a new release.', and a 'Packages' section.

- Then, edit the file and make some changes. Here, we are adding one more line to the file **HelloWorld.py**.



The screenshot shows the GitHub code editor for 'HelloWorld.py' in the 'master' branch. The code is as follows:

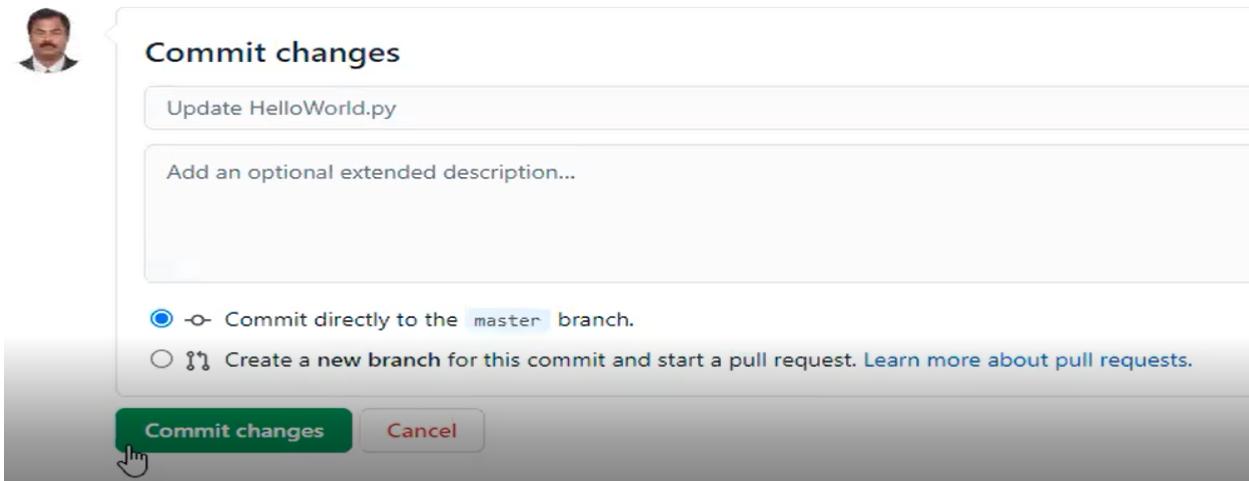
```

1  #!/usr/bin/python3
2  # This Python program will print Hello World...
3  print("Hello World Demo for GitHub hook trigger for GITScm polling...\n")
4  print("Hello World Demo for GitHub hook trigger for GITScm polling...\n")
5  print("Hello World Demo for GitHub hook trigger for GITScm polling...\n")
6

```

The last line of code ('print("Hello World Demo for GitHub hook trigger for GITScm polling...\n")') is highlighted with a red box.

- Now, scroll down and commit the changes.



The screenshot shows the 'Commit changes' dialog box. It includes fields for 'Update HelloWorld.py' and 'Add an optional extended description...', and two radio button options for committing:

- Commit directly to the `master` branch.
- Create a new branch for this commit and start a pull request. Learn more about pull requests.

At the bottom are 'Commit changes' and 'Cancel' buttons, with 'Commit changes' being the active button.

- Then, come back to the **Jenkins Dashboard**. You would see your job getting built as shown in the screenshot given below.

Dashboard > WebHookProject >

-  Workspace
-  Build Now
-  Configure
-  Delete Project
-  GitHub Hook Log
-  Rename

 **Build History** trend ^

find X

 #1	Mar 30, 2021, 11:51 AM
--	------------------------

11. Once the project is successfully built, click on the build number to view the log output as shown in the screenshot given below.

-  Workspace
-  Build Now
-  Configure
-  Delete Project
-  GitHub Hook Log
-  Rename

 **Build History** trend ^

find X

 #1	Mar 30, 2021, 11:51 AM
--	------------------------

11. Once you click on the build, you can view the console output as shown in the screenshot given below.

Console Output

```
Started by GitHub push by BThangaraju
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/WebHookProject
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/BThangaraju/Jenkins.git
> git init /var/lib/jenkins/workspace/WebHookProject # timeout=10
Fetching upstream changes from https://github.com/BThangaraju/Jenkins.git
> git --version # timeout=10
> git --version # 'git' version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/BThangaraju/Jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/BThangaraju/Jenkins.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision b652cc61c3c1fd3cd023e17c6d0f912759798ac4 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f b652cc61c3c1fd3cd023e17c6d0f912759798ac4 # timeout=10
Commit message: "Update HelloWorld.py"
First time build. Skipping changelog.
[WebHookProject] $ /bin/sh -xe /tmp/jenkins18193083689466439696.sh
```