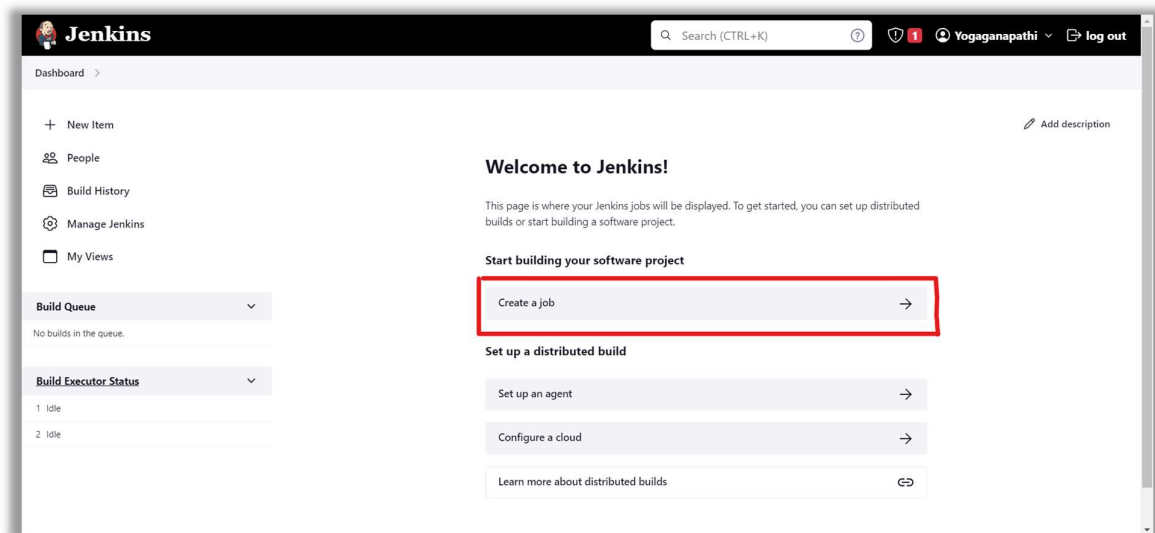
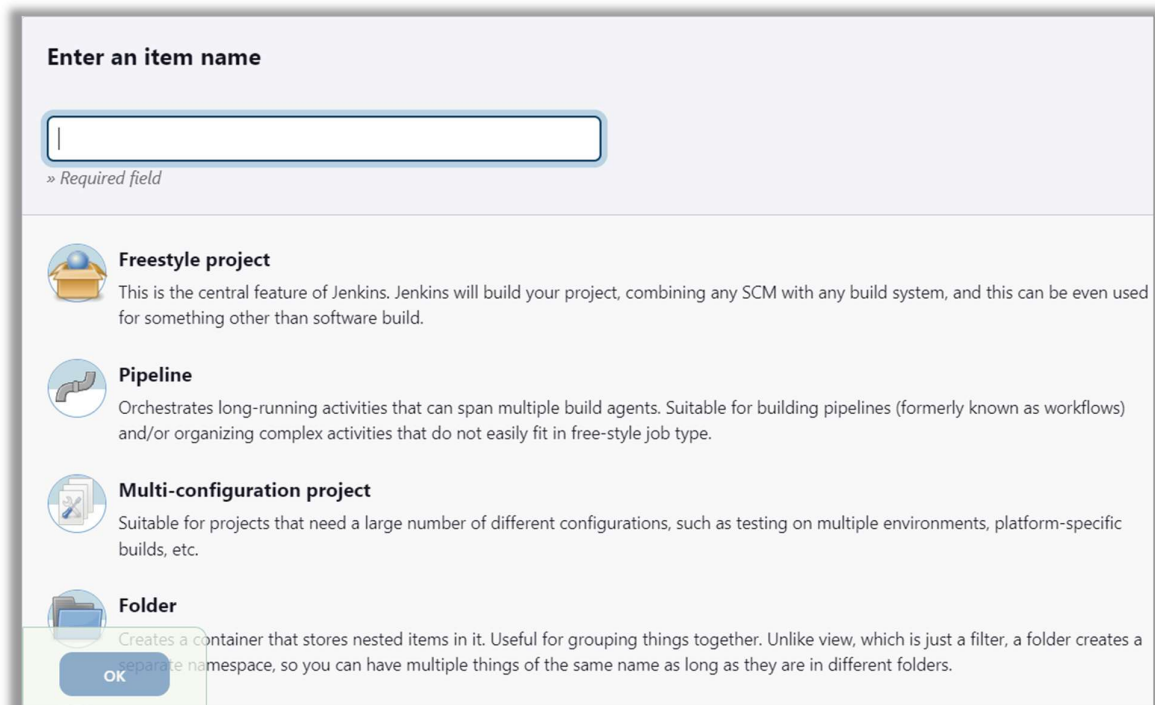


JENKINS CLASS -2

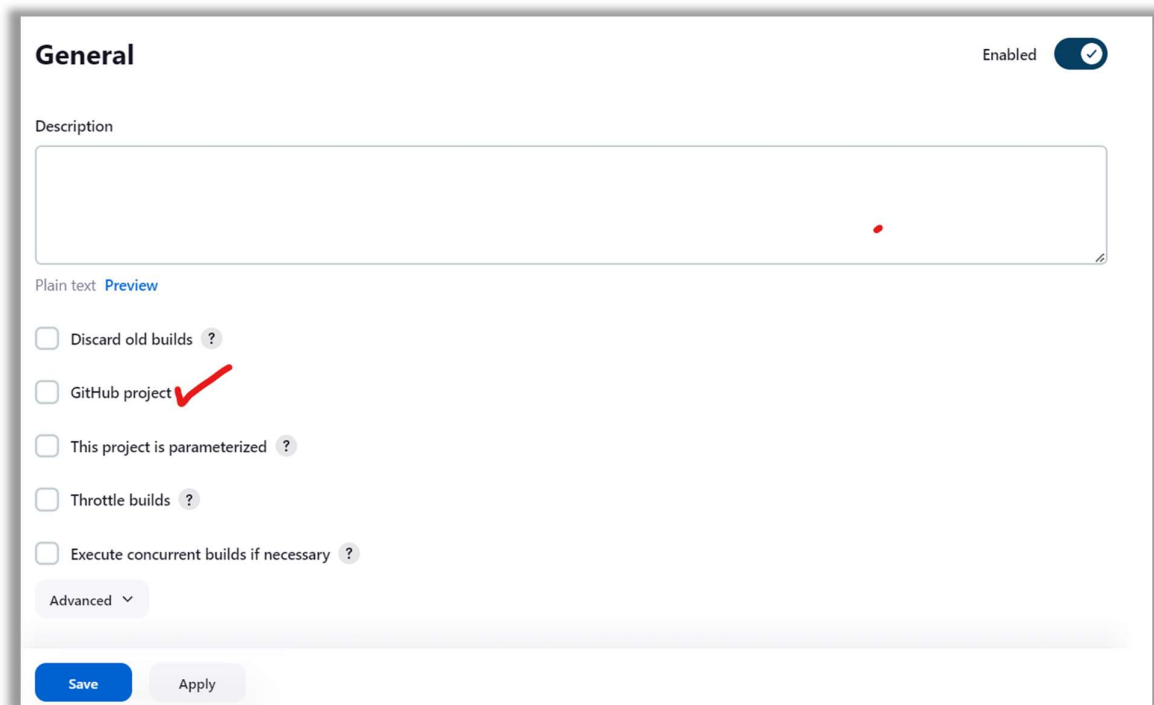
To create the job click on **create a job**. Job is nothing but what kind of action we have done by triggering functions which is call job in Jenkins.



Enter the job name as you wish. And choose **Freestyle project** for create the basic Github related automation as of now. Then click on ok.

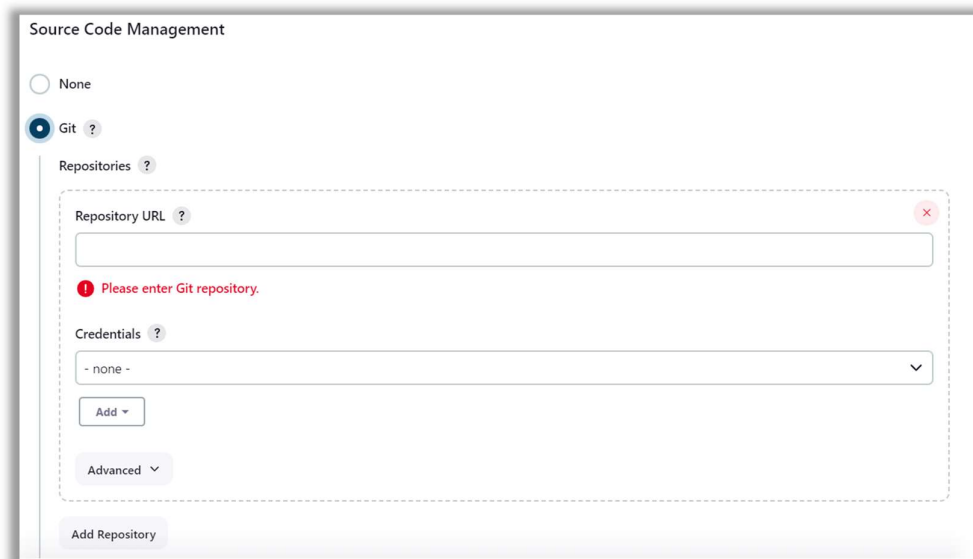


Choose the below options to start the automation along with github.



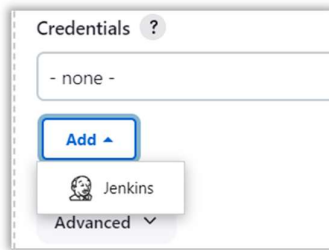
The image shows the 'General' configuration page in Jenkins. At the top right, there is a toggle switch labeled 'Enabled' which is turned on. Below this is a 'Description' text area. Underneath the description is a 'Plain text' label with a 'Preview' link. A list of checkboxes follows: 'Discard old builds', 'GitHub project' (which is checked and has a red checkmark next to it), 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. Below these is an 'Advanced' dropdown menu. At the bottom, there are 'Save' and 'Apply' buttons.

In this part we have to choose git for SCM (Source Code Management). Enter the GitHub repository link in repository URL. If you had been choosing the public repo then no need to set the credentials of your GitHub. If you have chosen the private repo, you'll get an error then you need to set the credentials for login purpose.



The image shows the 'Source Code Management' configuration page in Jenkins. At the top, there are two radio buttons: 'None' and 'Git'. The 'Git' radio button is selected. Below this is a 'Repositories' section with a dashed border. Inside this section, there is a 'Repository URL' text field with a red error message below it that says 'Please enter Git repository.' Below the URL field is a 'Credentials' dropdown menu with '- none -' selected. There is an 'Add' button next to the credentials dropdown. At the bottom of the 'Repositories' section is an 'Advanced' dropdown menu. Below the entire section is an 'Add Repository' button.

You can set credentials by following steps. Click on add then click on Jenkins. You will have a credential page.



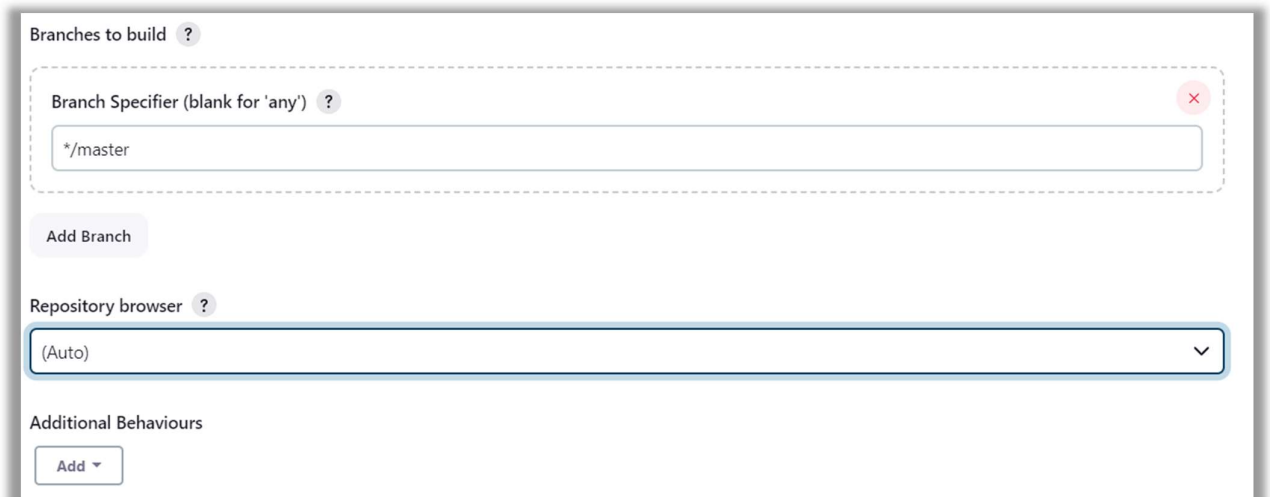
A screenshot of the Jenkins 'Credentials' dropdown menu. The menu is open, showing a list of credentials. The first option is '- none -'. Below it is an 'Add' button with a plus icon. Further down is a 'Jenkins' icon with the text 'Jenkins' next to it. At the bottom of the menu is an 'Advanced' dropdown arrow.

Now you have to fill the username (GitHub username), password (GitHub token). We already know about how to get the GitHub token we were discussed it in GitHub session.



A screenshot of the 'Jenkins Credentials Provider: Jenkins' form. The form is titled 'Add Credentials'. It has several fields: 'Domain' with a dropdown menu showing 'Global credentials (unrestricted)'; 'Kind' with a dropdown menu showing 'Username with password'; 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)'; and 'Username' with a text input field. There are also help icons (?) next to the 'Scope' and 'Username' labels.

After enter the GitHub repository credentials, now we have to enter the branch details. In this part we should choose the branch name for which branch we have to integrate with Jenkins. For example, If we want to integrate the master branch, then we need to put the value as ***/master** in the Branch Specifier field. If we want to integrate with all branch in the repo we can keep it as blank which mean erase all the data keep it as empty in Branch Specifier field.



A screenshot of the 'Branches to build' form. The form is titled 'Branches to build'. It has a 'Branch Specifier (blank for 'any')' field with a text input containing '*/master'. Below this field is an 'Add Branch' button. Further down is a 'Repository browser' dropdown menu showing '(Auto)'. At the bottom is an 'Additional Behaviours' section with an 'Add' button.

In build trigger part we have 3 option to automate the things as of now to learn.

Build periodically – we can set schedule when that project should build.

GitHub hook trigger for GITScm polling – whenever we commit a change it will create a build.

Poll SCM – We can set the schedule after commit the changes. This schedule only create the build when commit happen.

In this part we have cron 5-star rule to define the schedule.

1. Minute
2. Hour
3. Day of the month
4. Month of the year
5. Day of the week.

Cron syntax in Jenkins is a way of specifying when to run a job using a pattern of five fields that represent the minutes, hours, days of the month, months, and days of the week. For example, the cron expression `0 23 * * *` means to run the job at 11:00 PM every day. The cron syntax in Jenkins is similar to the Linux cron syntax, but with some differences and extensions. Here are some of the main features of the cron syntax in Jenkins:

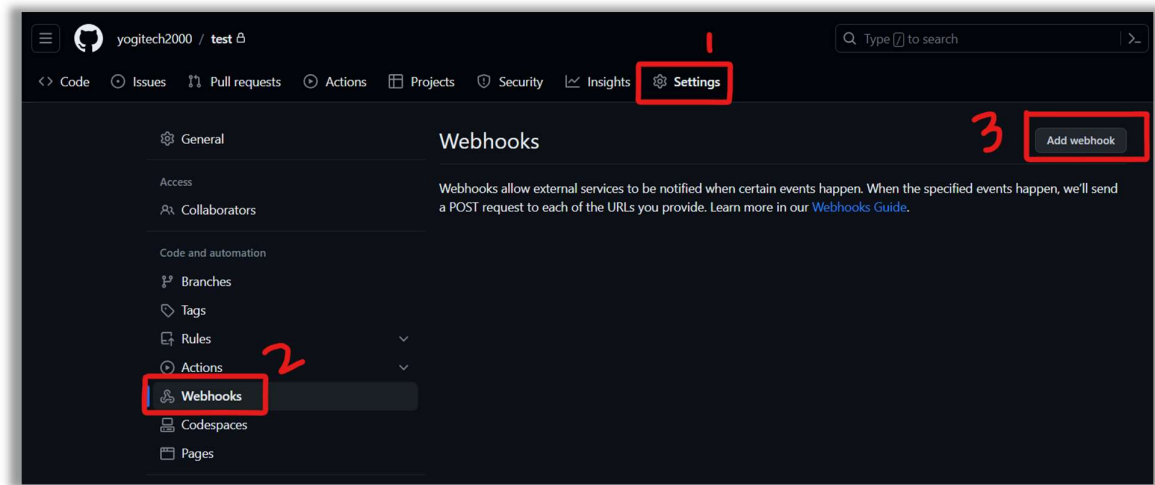
- You can use the symbol `H` (for hash) instead of a specific value to spread the load evenly across a time range. For example, `H/15 * * * *` means to run the job every 15 minutes, but not necessarily at :00, :15, :30, and :45. The actual time will be determined by hashing the job name.
- You can use ranges and intervals to specify multiple values. For example, `H(0-29)/10 * * * *` means to run the job every 10 minutes in the first half of every hour, three times per hour.
- You can use aliases for months and days of the week, such as `JAN`, `FEB`, `MON`, `TUE`, etc. For example, `0 0 1-7 * MON` means to run the job at midnight on the first Monday of every month.
- You can use special strings to specify common schedules, such as `@daily`, `@weekly`, `@monthly`, etc. For example, `@midnight` means to run the job at midnight every day.

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 ?

Then finally save the configs. Now we have to do some configs in GitHub.

Go to path : github->your repo-> setting -> webhooks ->add webhook.



Payload URL: <Jenkins-url till 8080>/github-webhook/.

Ex - <http://18.232.159.227:8080/>

Content type: application/json

Remaining set as default. Then add webhook configs.

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

Content type

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

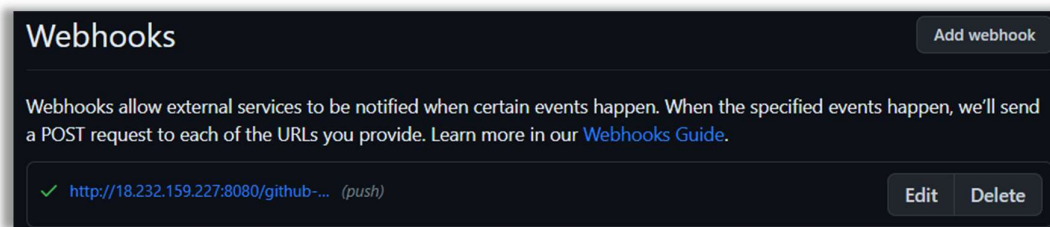
☐ Send me **everything**.

☐ Let me select individual events.

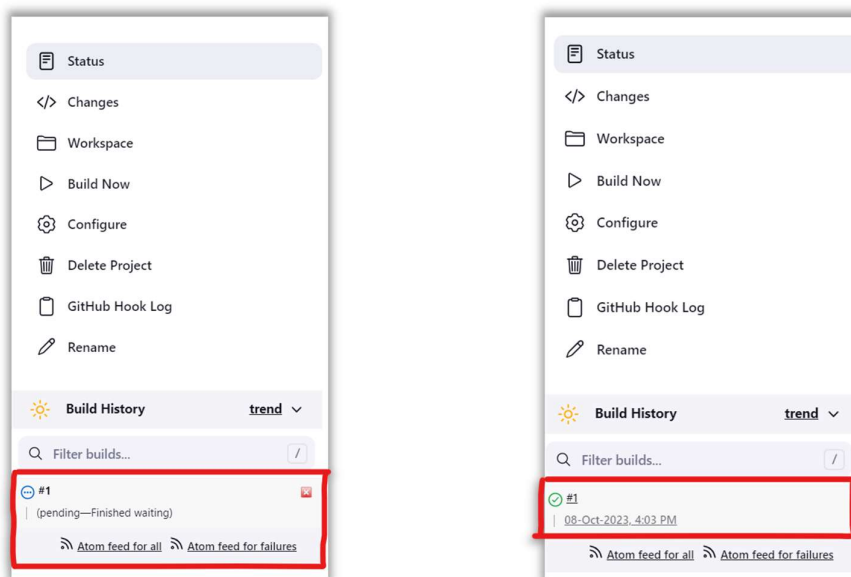
☒ **Active**

We will deliver event details when this hook is triggered.

Make sure the webhook as active by verify the tick mark.



Whenever you commit any changes in GitHub it will trigger the Jenkins it will make pull function in the instance. We can see the what are the changes are happened in public repo it will pull to our local repo automatically. Then it triggered successfully it will reflect as green tick ✓ . If that build get fail it will reflect as red cross mark ✗ .



Whenever we build the jenkins job, it will be store the job details in **/var/lib/jenkins/workspace/** - this path. We can check the jenkins build log in workspace which we have two methods.

1. We can check it in aws instance **/var/lib/jenkins/workspace/** in this path.
2. We have option in jenkins Dashboard left side to view the workspace shown in the below.

