

DevOps in Action

Jenkins with GitHub and Python

Github - Create an account

- Go to <https://github.com/join> in a web browser.

Create your personal account

Username *

✓

This will be your username. You can add the name of your organization later.

Email address *

✓

We'll occasionally send updates about your account to this inbox. We'll never share your email address with anyone.

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Verify account

✓

wikiHow to Create an Account on GitHub

Github - Create an account

- **Enter your personal details**
- **Click the "Create an account" button.**
- Complete the CAPTCHA puzzle.
- Click the "Verify email address" button in the message from GitHub.
- Select your preferences and click Submit.
- Open Inbox and search for email from - noreply@github.com
 - Click - "Verify Email Address"

GitHub Repositories

- Contain all the repositories on which the user is working.

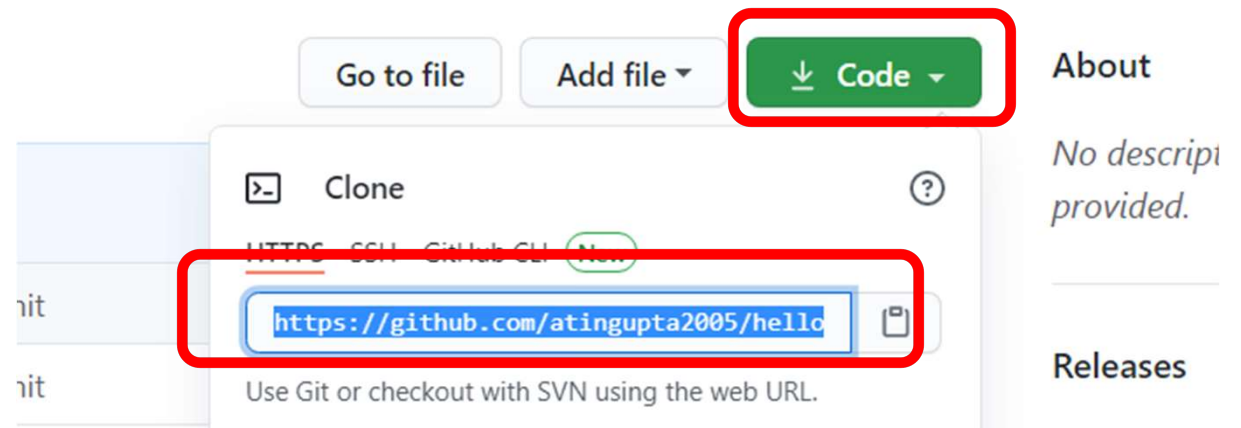
Github - Copy your application code

- Visit - <https://github.com/atingupta2005/flask-hello-world>



Git clone the github code

- Cloning a repository pulls down a full copy of all the repository data that GitHub has at that point in time
- The git clone command is used to create a copy of a specific repository or branch within a repository.



Continuous Integration Tool

Introduction

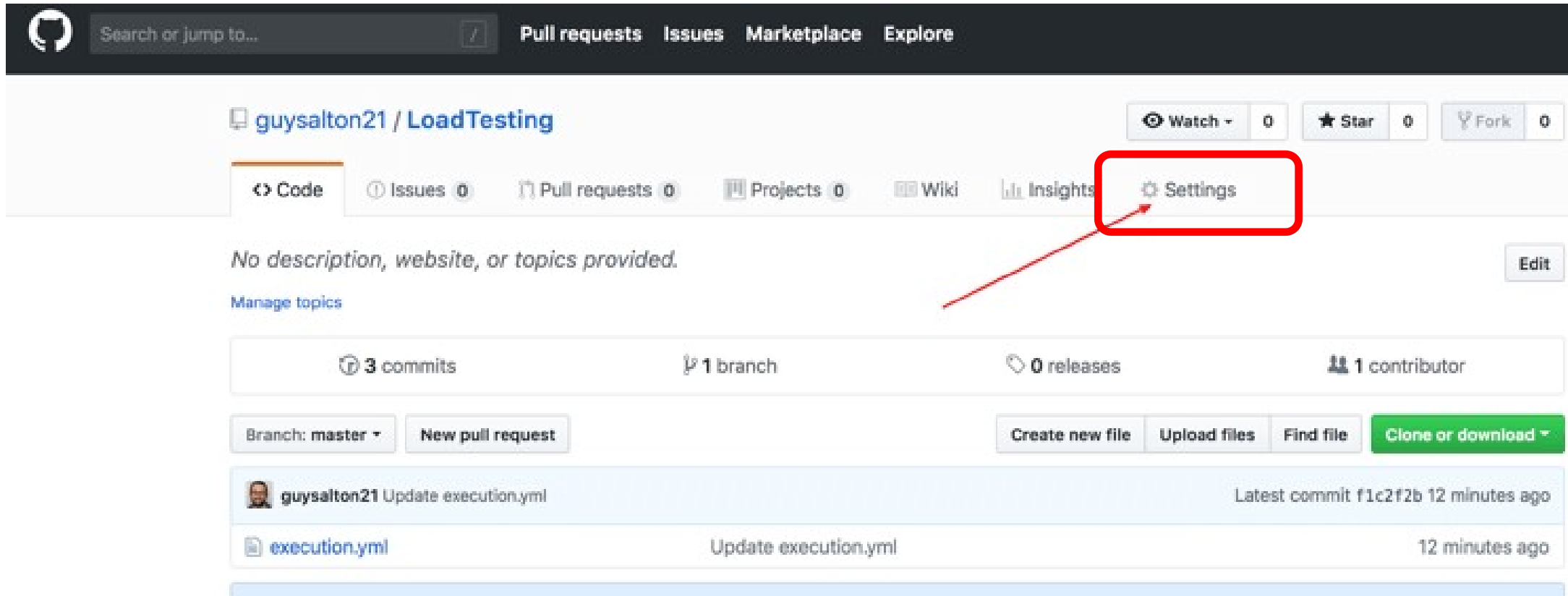
- One of the basic steps of implementing CI/CD is integrating your SCM (Source Control Management) tool with your CI tool.
- This saves you time and keeps your project updated all the time.
- One of the most popular and valuable SCM tools is GitHub.
- We will:
 - Schedule build
 - Pull code and data files from your GitHub repository to Jenkins machine
 - Automatically trigger each build on the Jenkins server, after each Commit on Git repository

Github Repo

- Fork Github repo:
 - <https://github.com/atingupta2005/flask-hello-world>

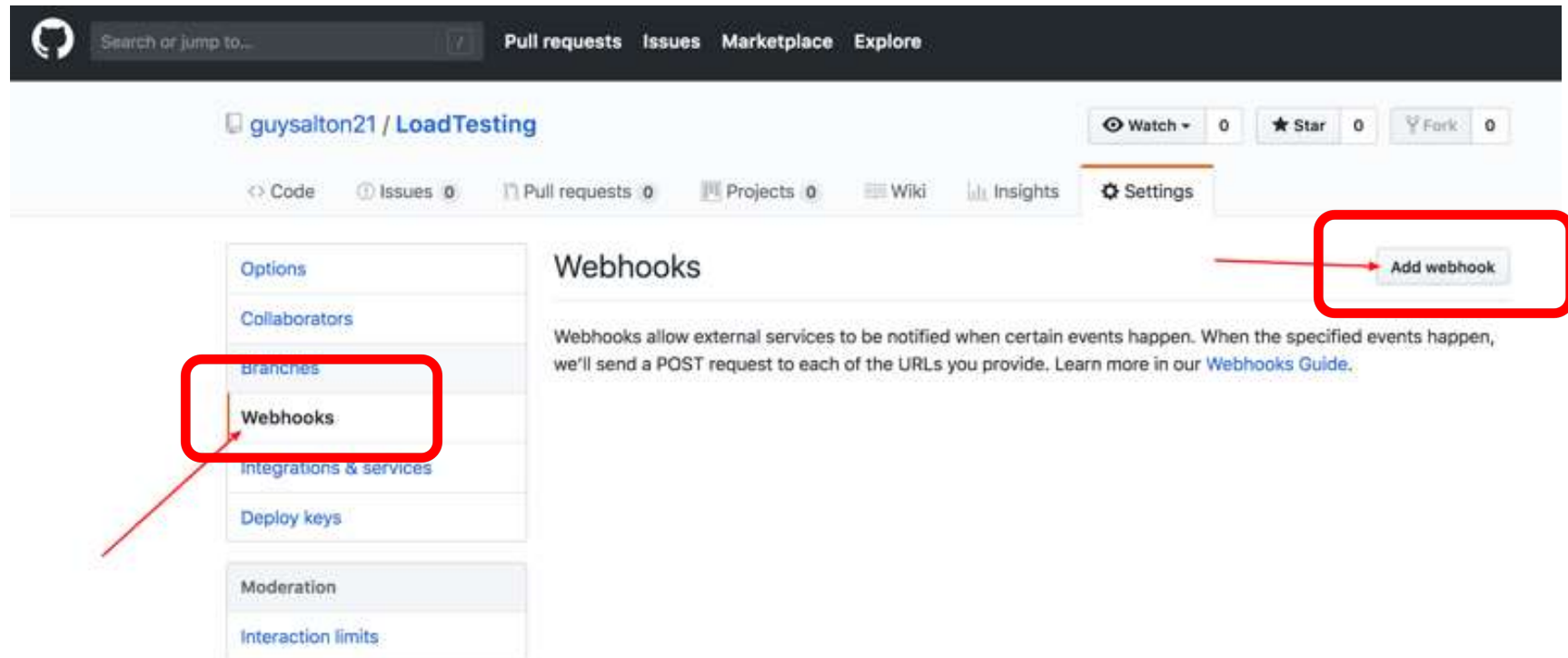
Configuring GitHub

- Go to your GitHub repository and click on 'Settings'.



Configuring GitHub

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



Configuring GitHub

- 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.

The screenshot shows the GitHub repository settings page for 'guysalton21 / LoadTesting'. The 'Settings' tab is selected, and the 'Webhooks' section is active. The 'Add webhook' form is displayed with the following fields:

- Payload URL ***: `http://jenkins-demo.blazemeter.com:8080/github-webhook/` (highlighted with a red rectangle)
- Content type**: `application/json` (highlighted with a red rectangle)
- Secret**: (empty field)

Configuring GitHub

- Step 4: in the 'Which events would you like to trigger this webhook?' 'Let me select individual events.'
- Then, check 'Pull Requests' and 'Push'
- At the end of this option, make sure 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, milestone, 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.

Configuring GitHub

The screenshot shows the GitHub webhook configuration page. It features a list of event types on the left and right, each with a checkbox and a description. At the bottom, there is an 'Active' checkbox and an 'Add webhook' button. Red boxes and arrows highlight the following elements:

- Pull requests**: A red box around the checkbox and label, with an arrow pointing to it from the top.
- Pushes**: A red box around the checkbox and label, with an arrow pointing to it from the left.
- Active**: A red box around the checkbox and label, with an arrow pointing to it from the right.
- Add webhook**: A red box around the green button, with an arrow pointing to it from the right.

Event Types and Descriptions:

- ☐ **Visibility changes**
Repository changes from private to public.
- ☒ **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.
- ☒ **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.

☒ **Active**
We will deliver event details when this hook is triggered.

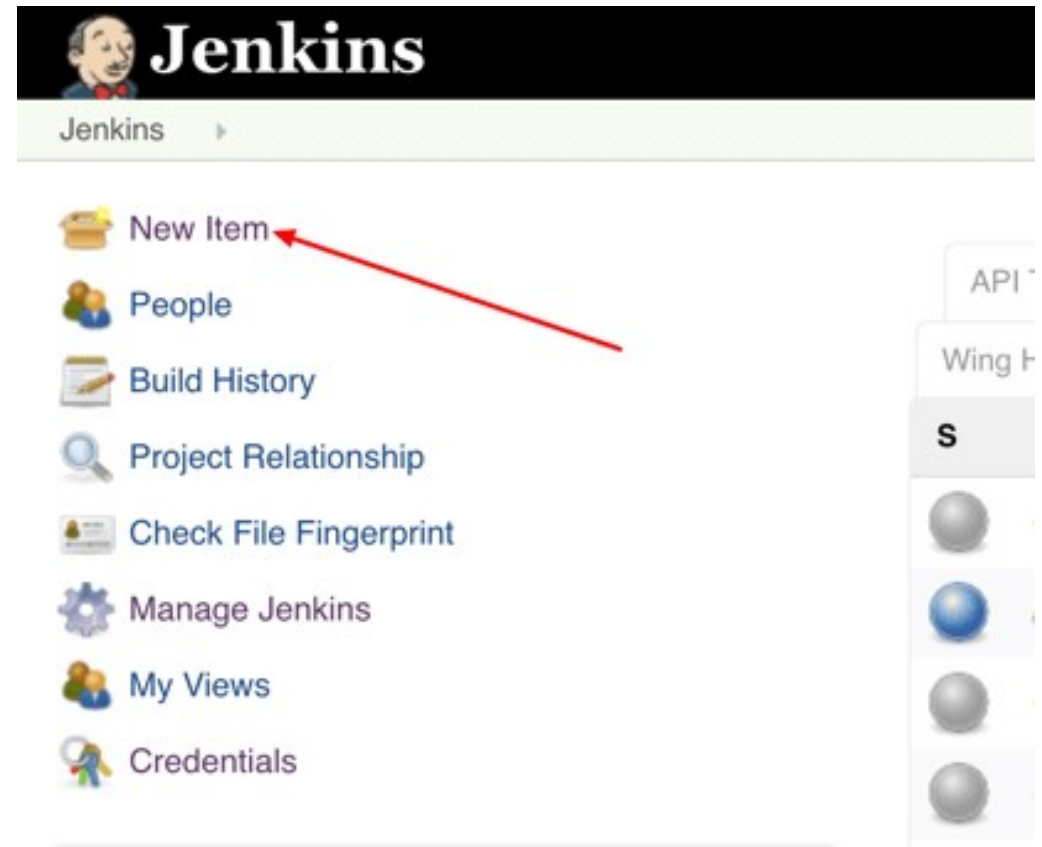
Add webhook

Configuring GitHub

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

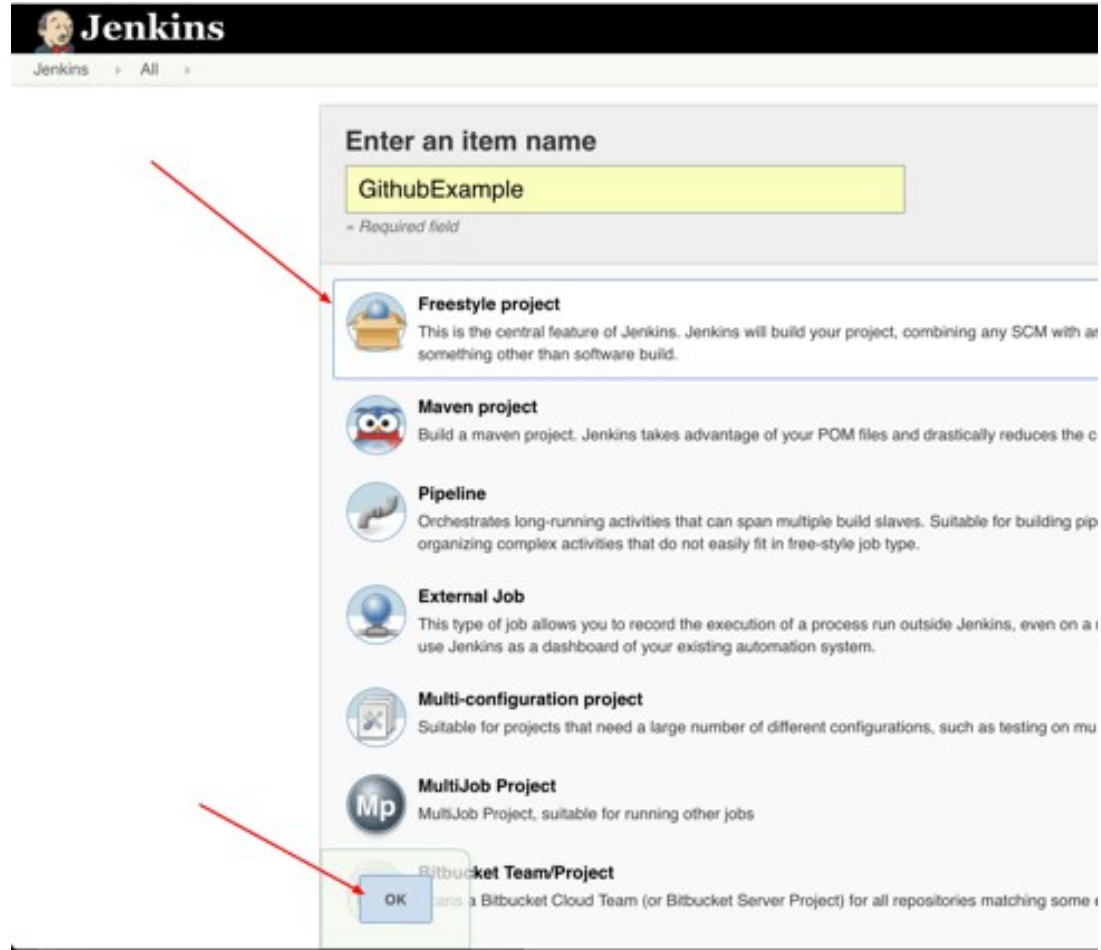
Configuring Jenkins

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



Configuring Jenkins

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



The screenshot shows the Jenkins 'New Item' configuration page. At the top, the Jenkins logo and navigation links are visible. Below the header, there's a section titled 'Enter an item name' with a text input field containing 'GithubExample'. A red arrow points from the 'Freestyle project' option in the list below to the 'OK' button at the bottom of the list. The list includes several project types: Freestyle project, Maven project, Pipeline, External Job, Multi-configuration project, MultiJob Project, and Bitbucket Team/Project. Each option has a brief description.

Enter an item name

GithubExample

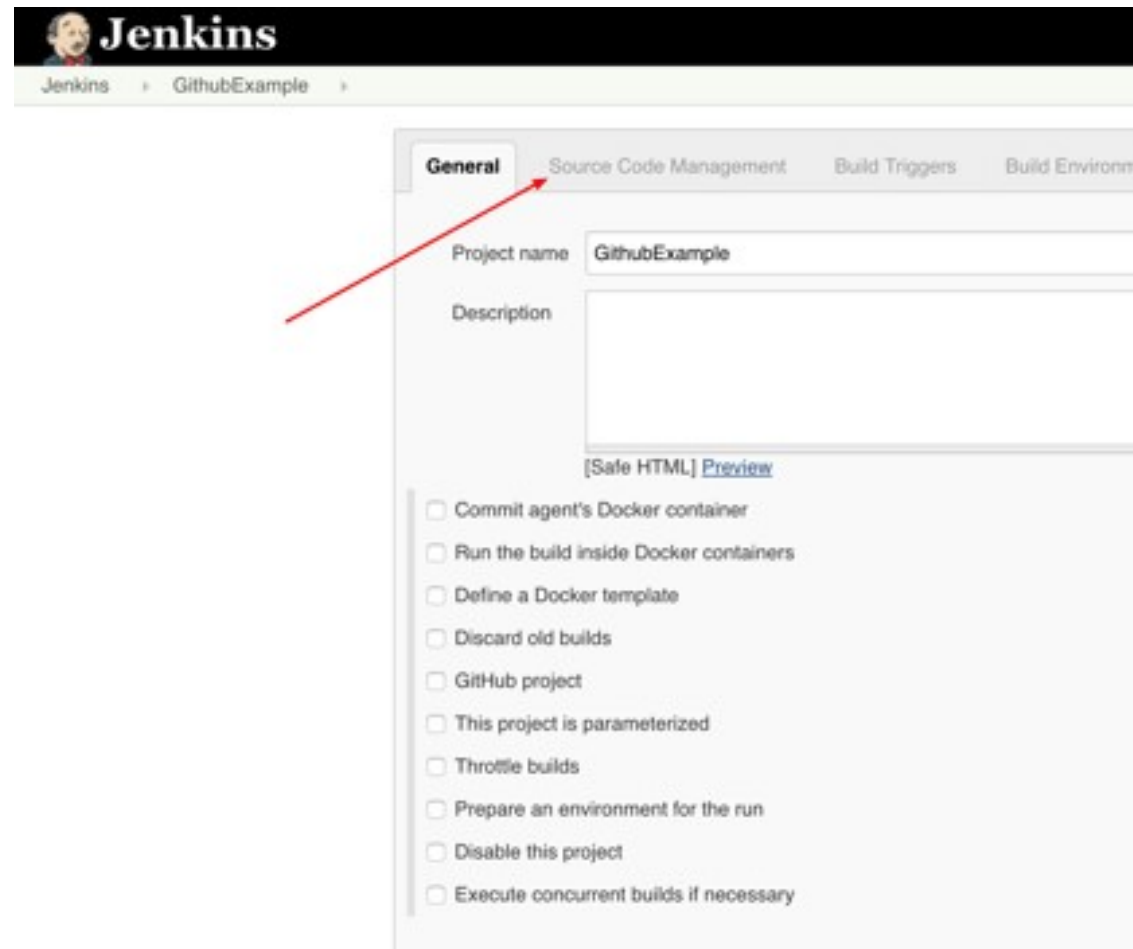
= Required field

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with anything other than software build.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the complexity of the build.
- Pipeline**
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines and organizing complex activities that do not easily fit in free-style job type.
- External Job**
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. Use Jenkins as a dashboard of your existing automation system.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple operating systems.
- MultiJob Project**
MultiJob Project, suitable for running other jobs
- Bitbucket Team/Project**
Create a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some pattern.

OK

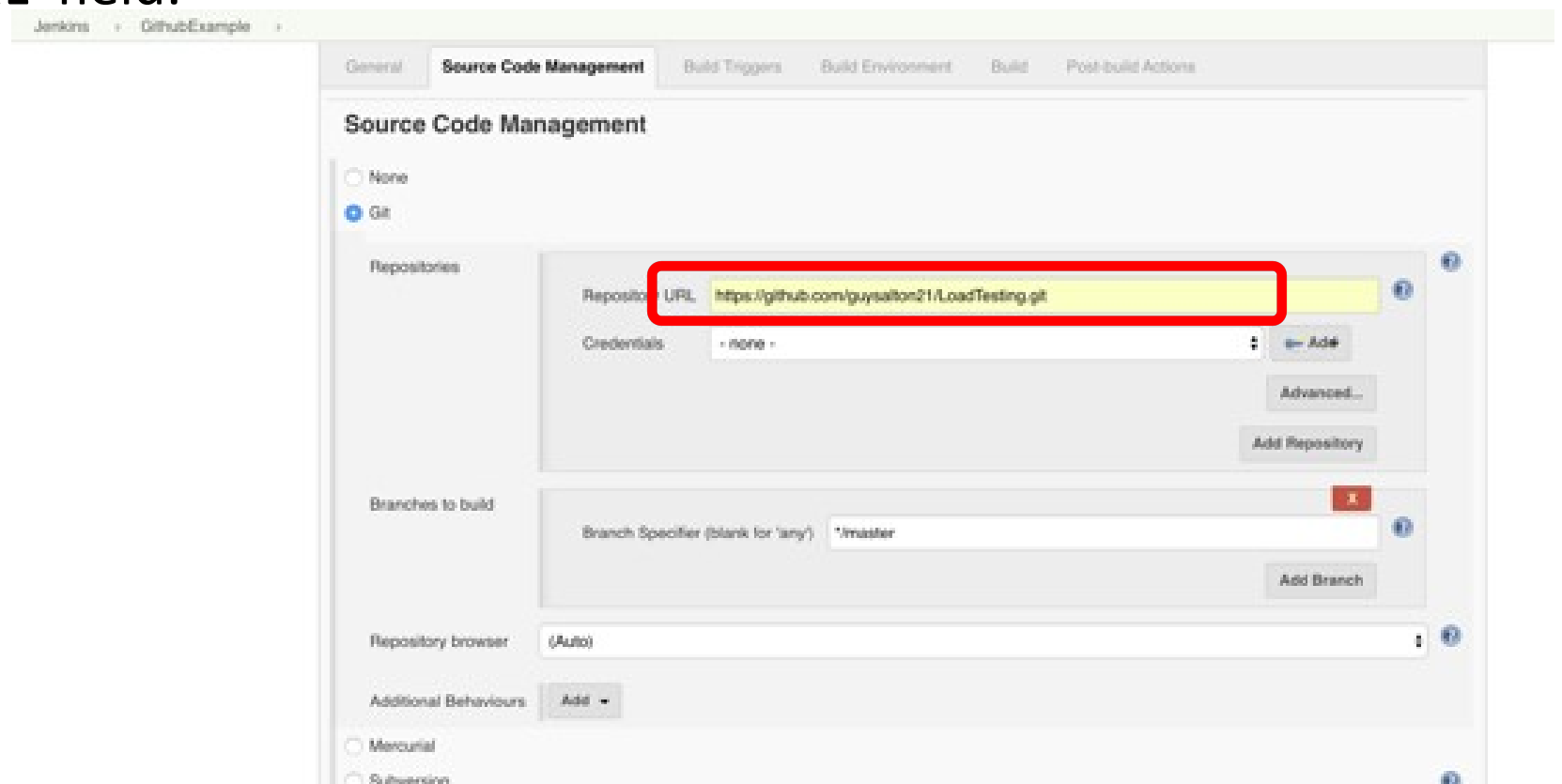
Configuring Jenkins

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



Configuring Jenkins

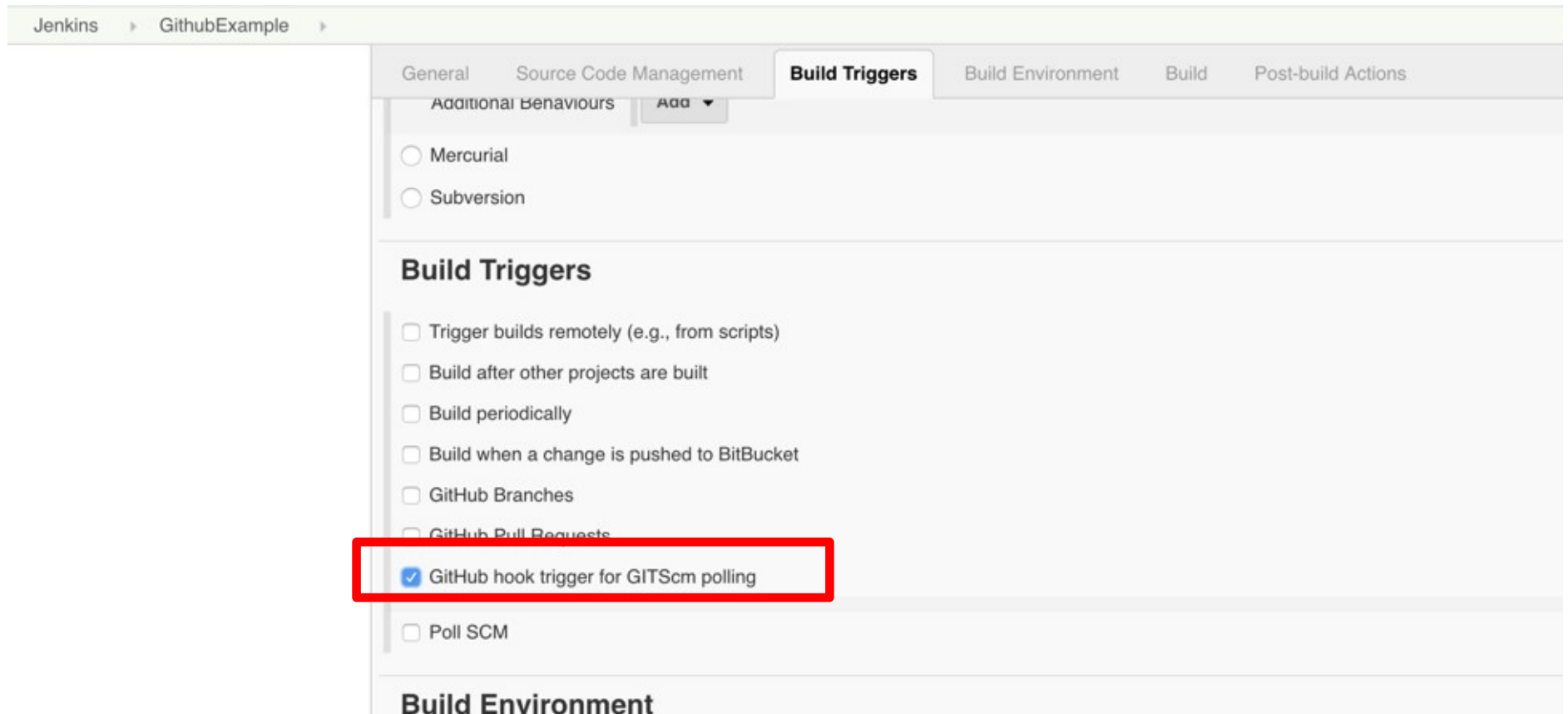
- Step 8: Click on Git and paste your forked GitHub repository URL in the 'Repository URL' field.



The screenshot shows the Jenkins configuration interface for a job named 'GithubExample'. The 'Source Code Management' tab is selected. Under the 'Source Code Management' section, the 'Git' option is chosen. In the 'Repositories' section, the 'Repository URL' field is highlighted with a red rectangle and contains the text 'https://github.com/guysaiton21/LoadTesting.git'. The 'Credentials' field is set to 'none'. Below this, the 'Branches to build' section shows the 'Branch Specifier (blank for \'any\')' set to '*/master'. The 'Repository browser' is set to '(Auto)'. At the bottom, there are options for 'Additional Behaviours' and 'Mercurial' or 'Subversion'.

Configuring Jenkins

- 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.



The screenshot shows the Jenkins configuration interface for a project named 'GithubExample'. The 'Build Triggers' tab is selected, displaying various options for triggering builds. The 'GitHub hook trigger for GITScm polling' option is checked and highlighted with a red rectangular box. Other visible options include 'Mercurial', 'Subversion', 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'Build when a change is pushed to BitBucket', 'GitHub Branches', 'GitHub Pull Requests', and 'Poll SCM'. The 'Build Environment' section is partially visible at the bottom.

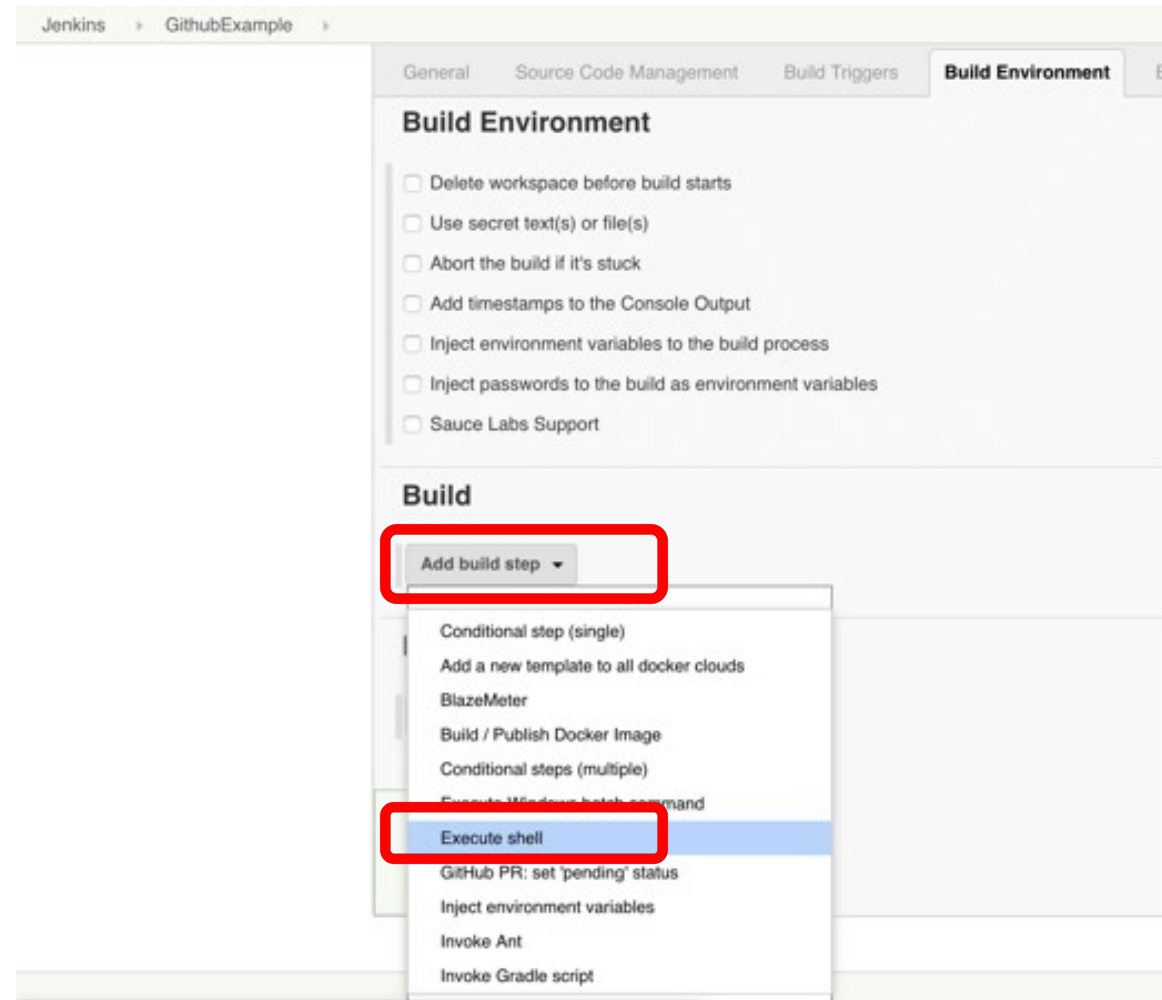
Configuring Jenkins

- 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.

Triggering the Jenkins Job to Run with Every Code Commit

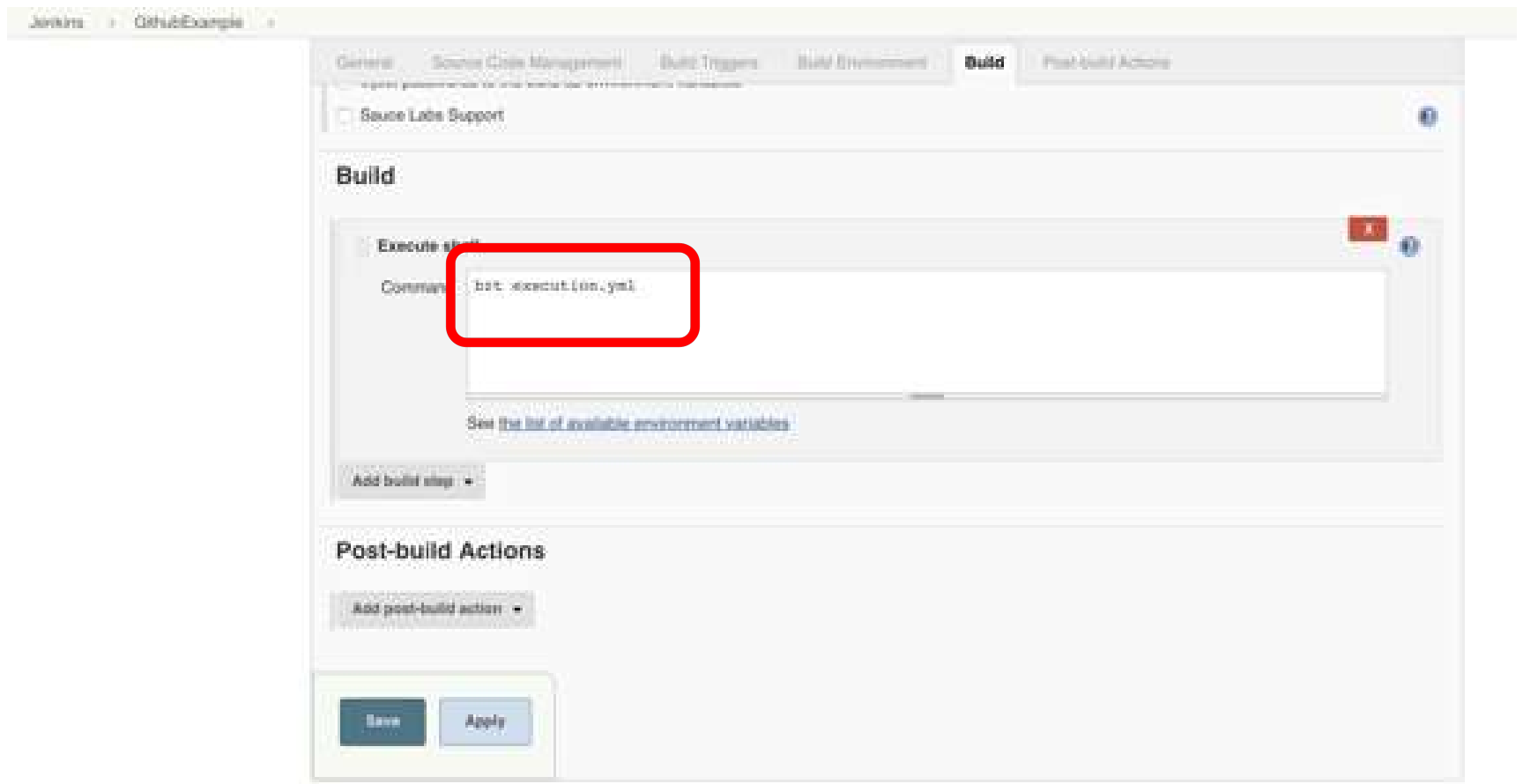
Triggering the Jenkins Job to Run

- Step 10: Click on the 'Build' tab,
- Then click on 'Add build step' and
- Choose 'Execute shell'.



Triggering the Jenkins Job to Run

- Step 11: To run sample commands - echo "Building Project"; echo "\$(pwd)"

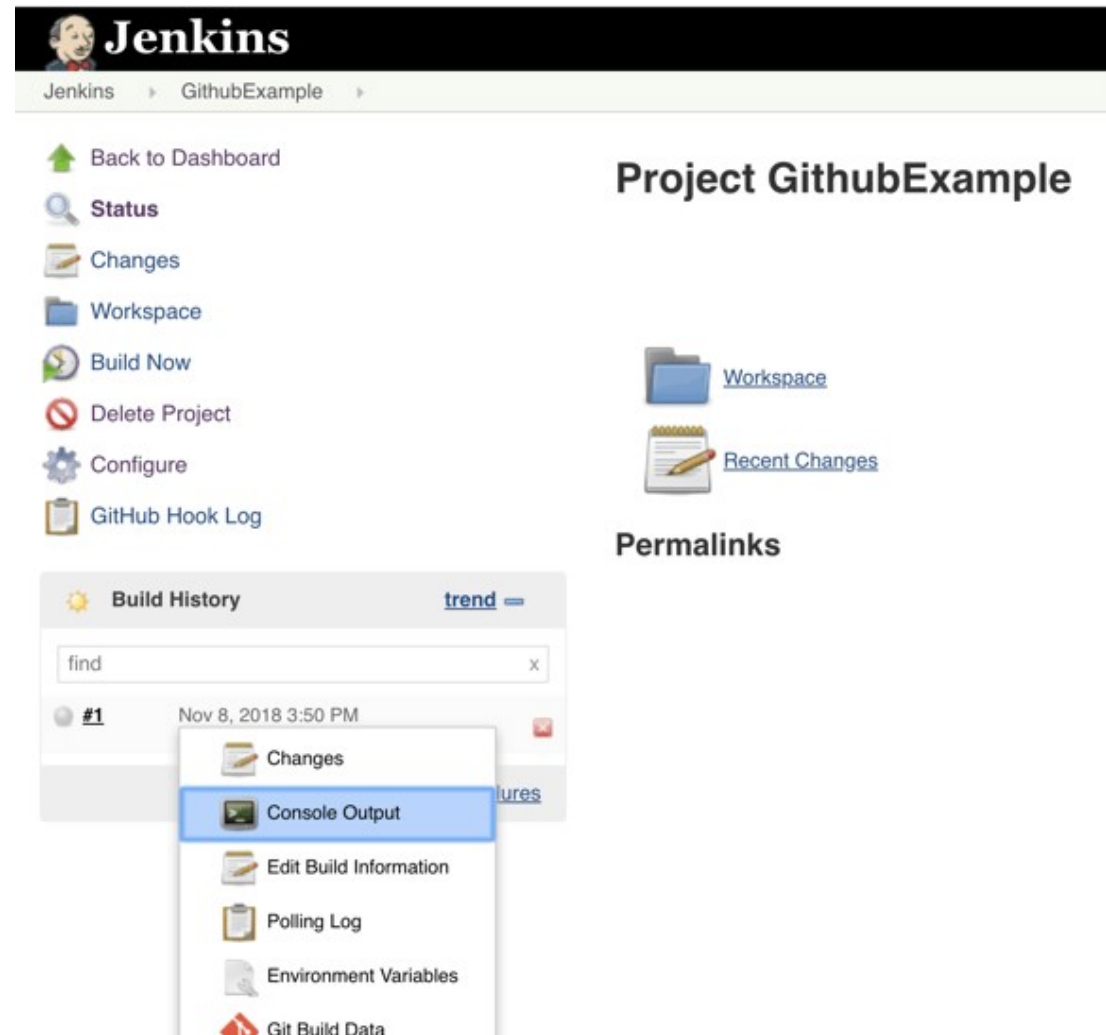


Triggering the Jenkins Job to Run

- Step 12: Go back to your GitHub repository, edit the code and commit the changes.
 - We will now see how Jenkins ran the script after the commit.

Triggering the Jenkins Job to Run

- Step 13: Go back to your Jenkins project and you'll see that a new job was triggered automatically from the commit we made at the previous step.
- Click on the little arrow next to the job and choose 'Console Output'.



Triggering the Jenkins Job to Run

- Step 14: You can see that Jenkins was able to pull the latest code and run it!
- Every time you publish your changes to Github, GitHub will trigger your new Jenkins job.

Thanks