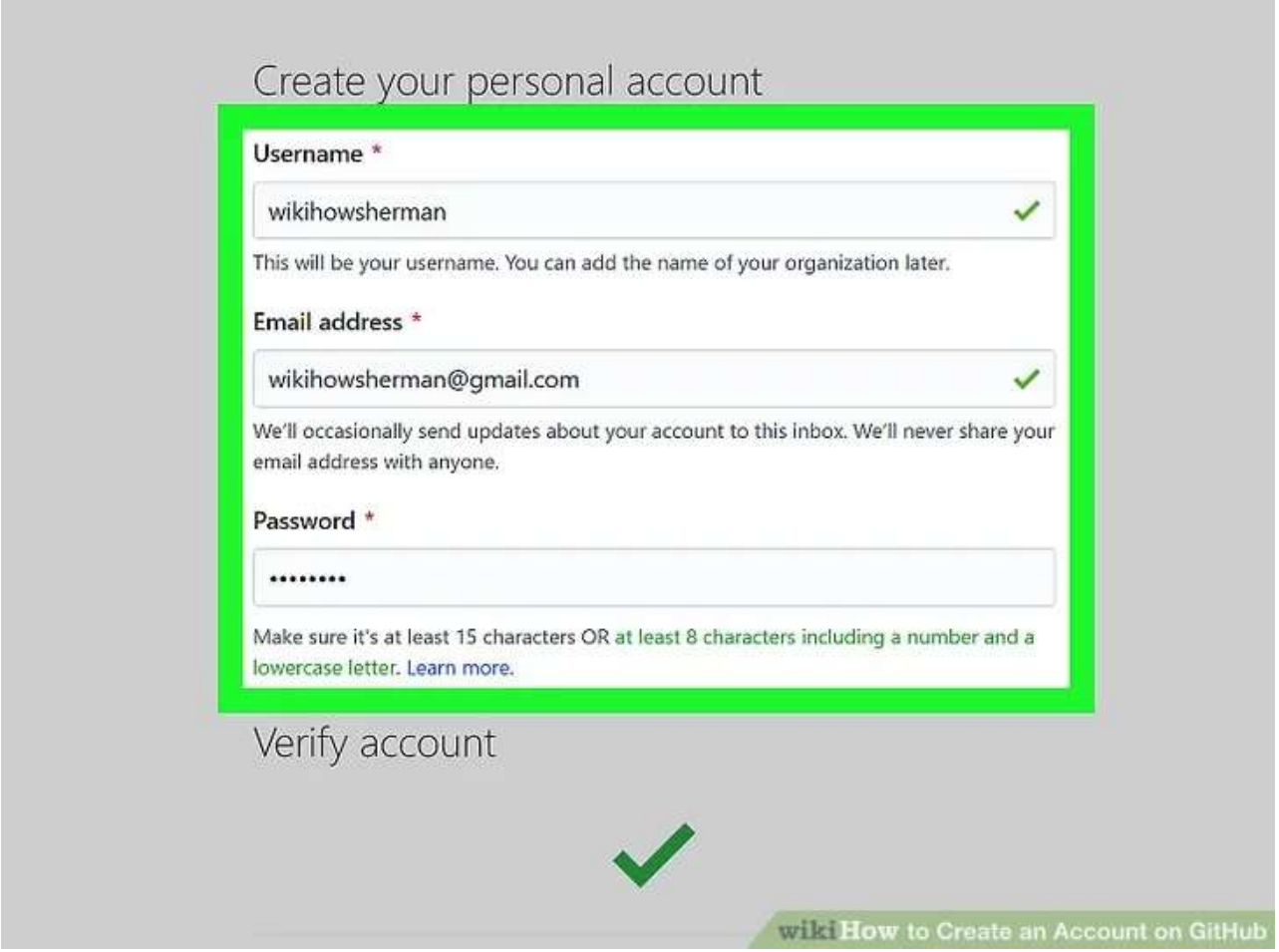
DevOps in Action

Jenkins with GitHub and Python

Github - Create an account

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



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

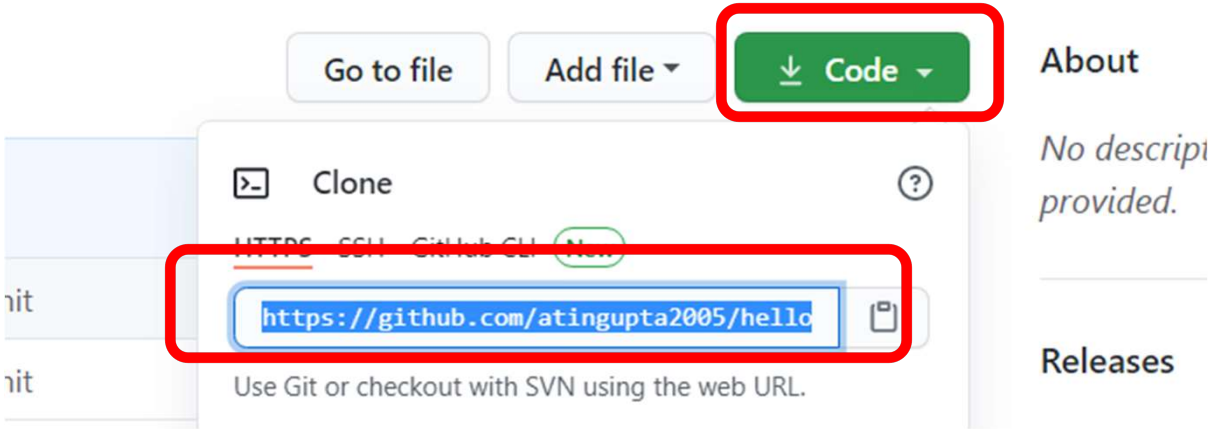
- Contain all the repositories on which the user is working.

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

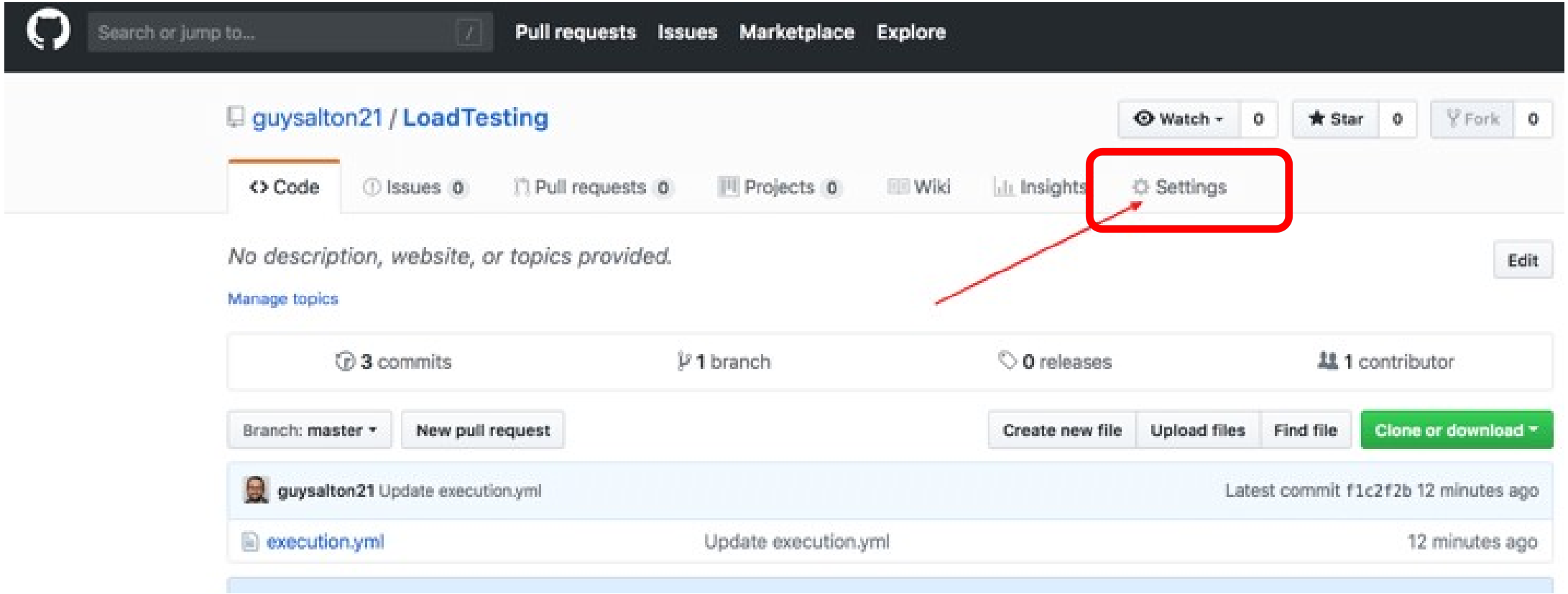
- One of the basic steps of implementing CI/CD is integrating your SCM (Source Control Management) tool with your CI tool.
- This saves you time and keeps your project updated all the time.
- One of the most popular and valuable SCM tools is GitHub.
- We will:
  - Schedule build
  - Pull code and data 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

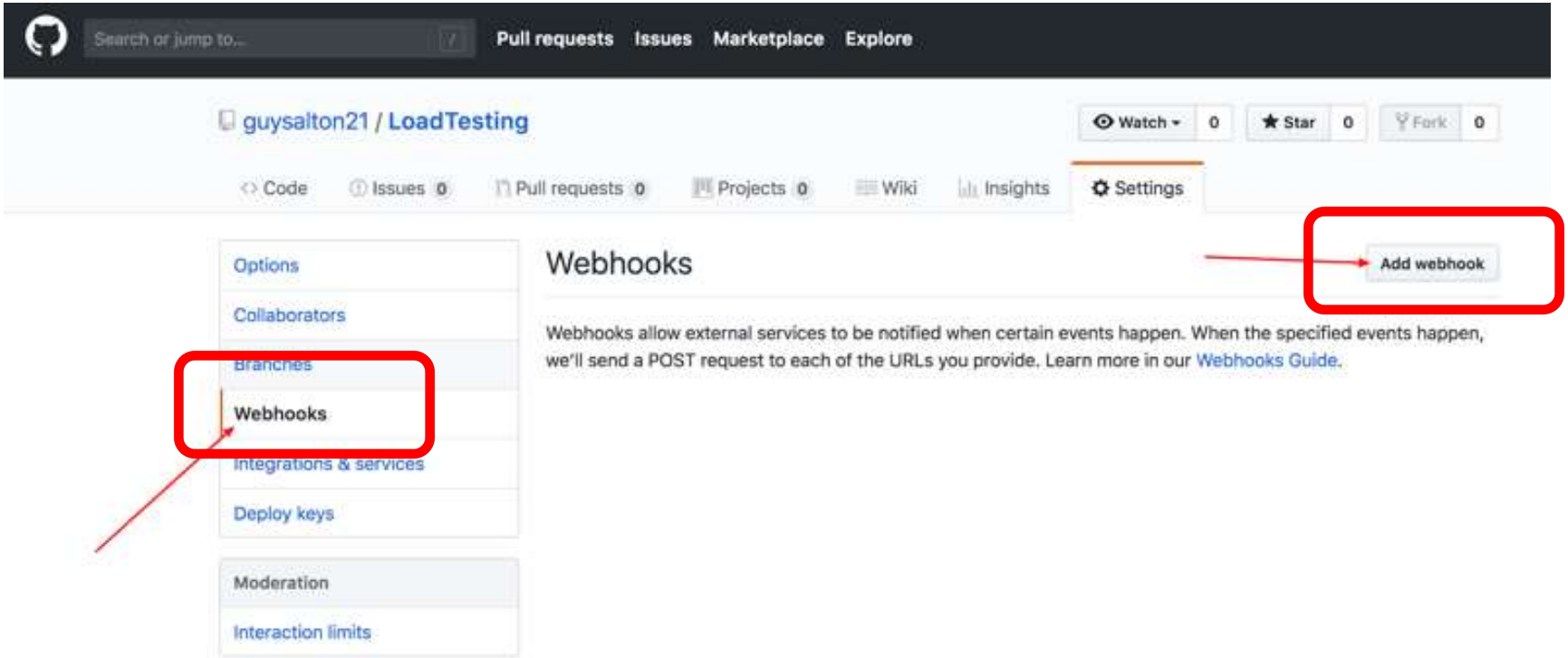
Configuring GitHub

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



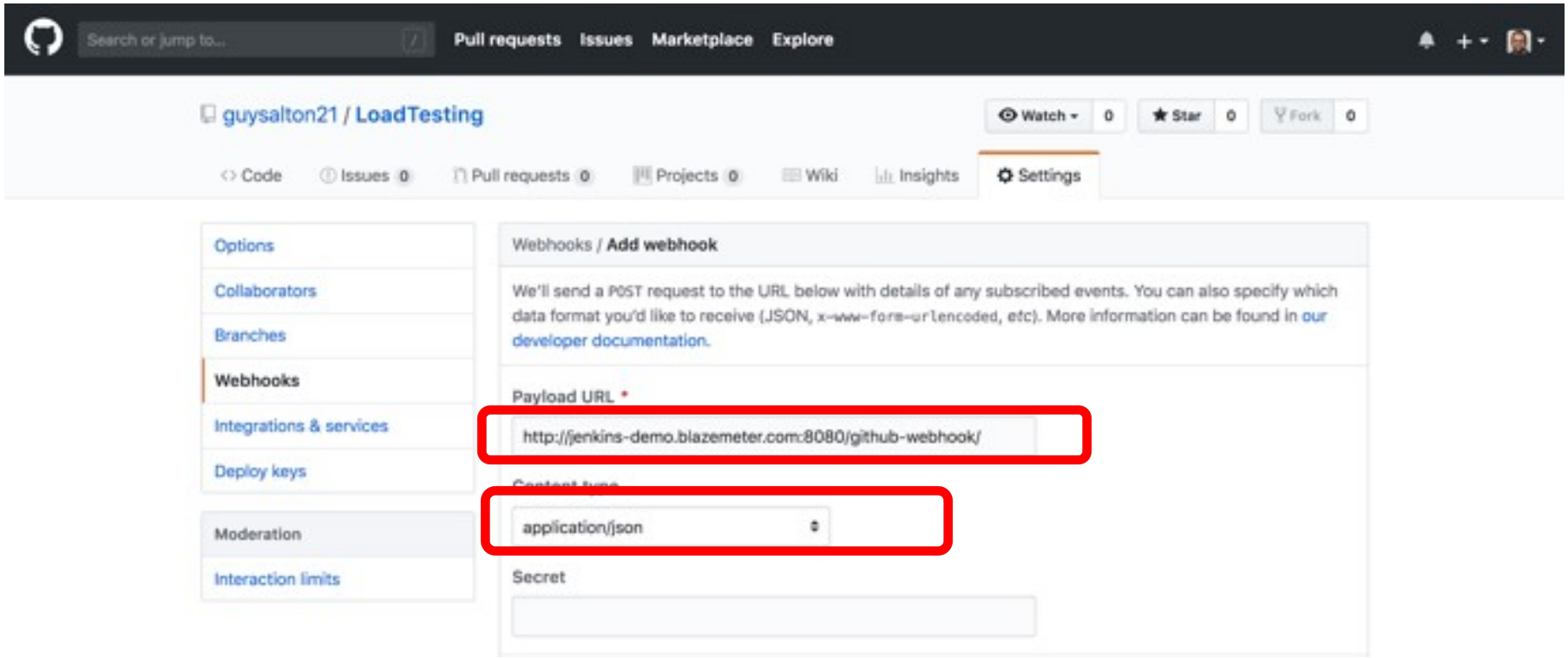
Configuring GitHub

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



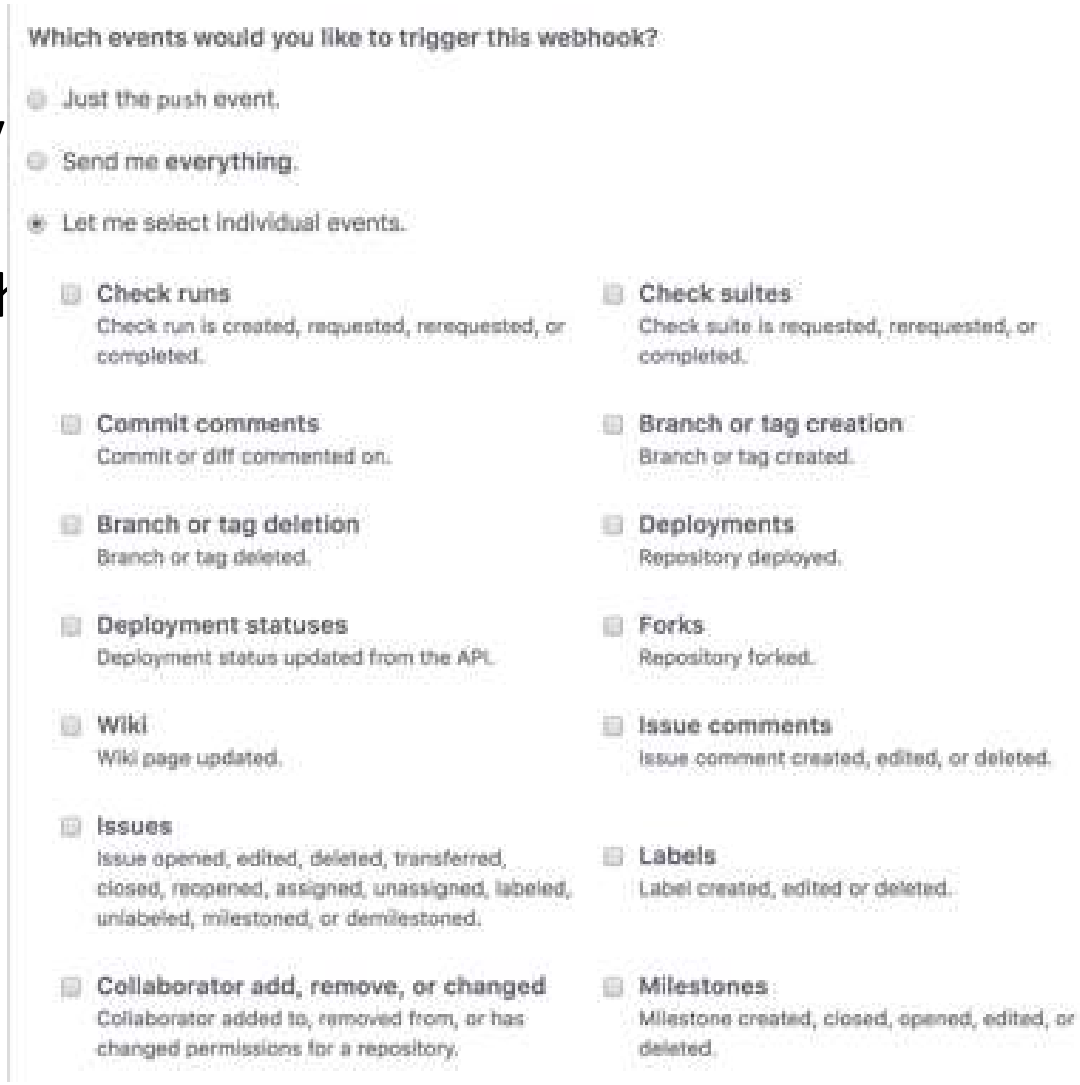
Configuring GitHub

- 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.

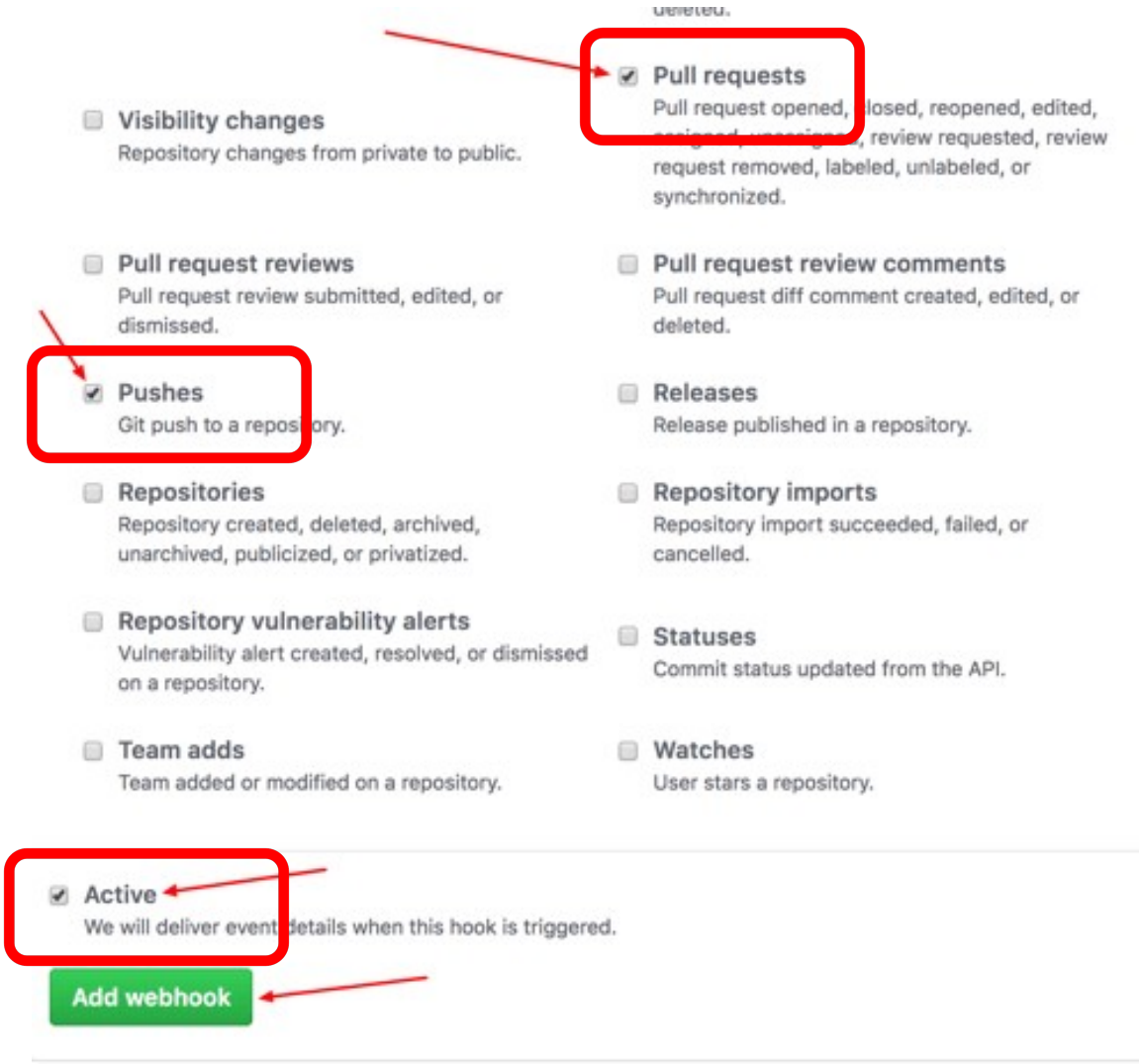


Configuring GitHub

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



Configuring GitHub

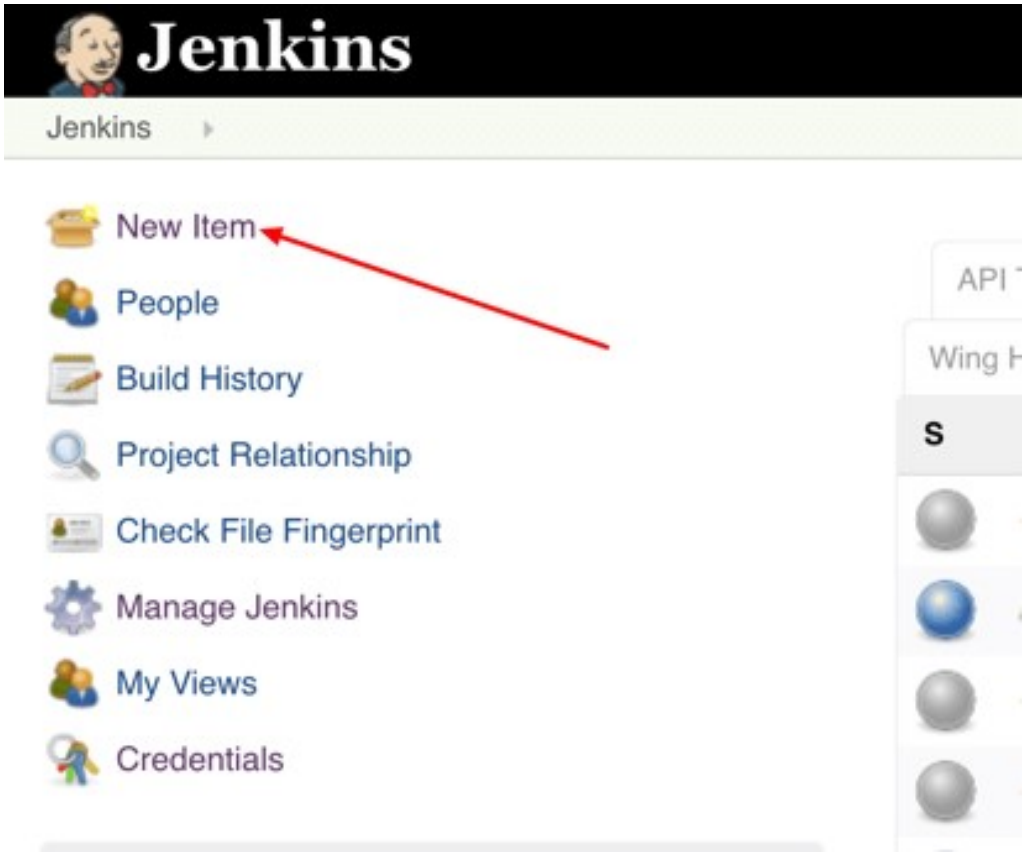


Configuring GitHub

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

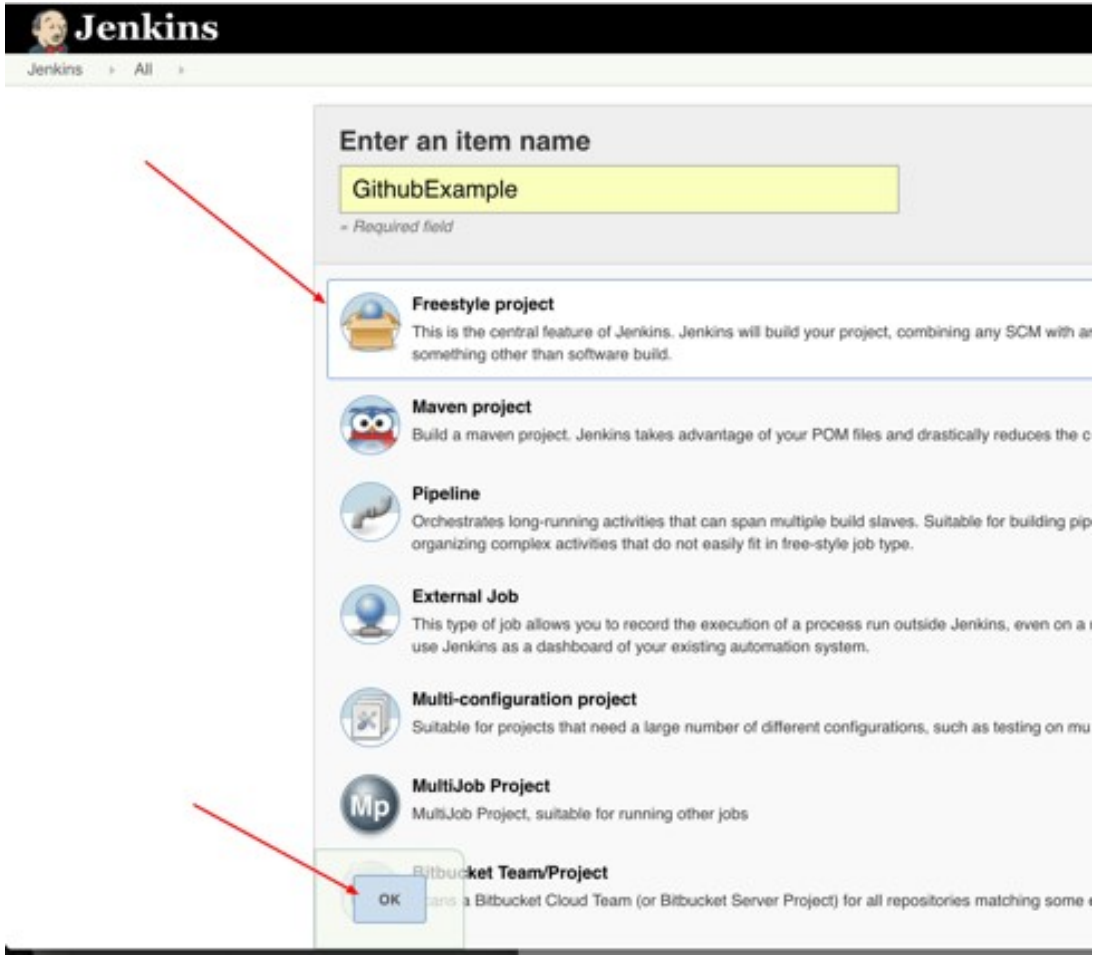
Configuring Jenkins

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



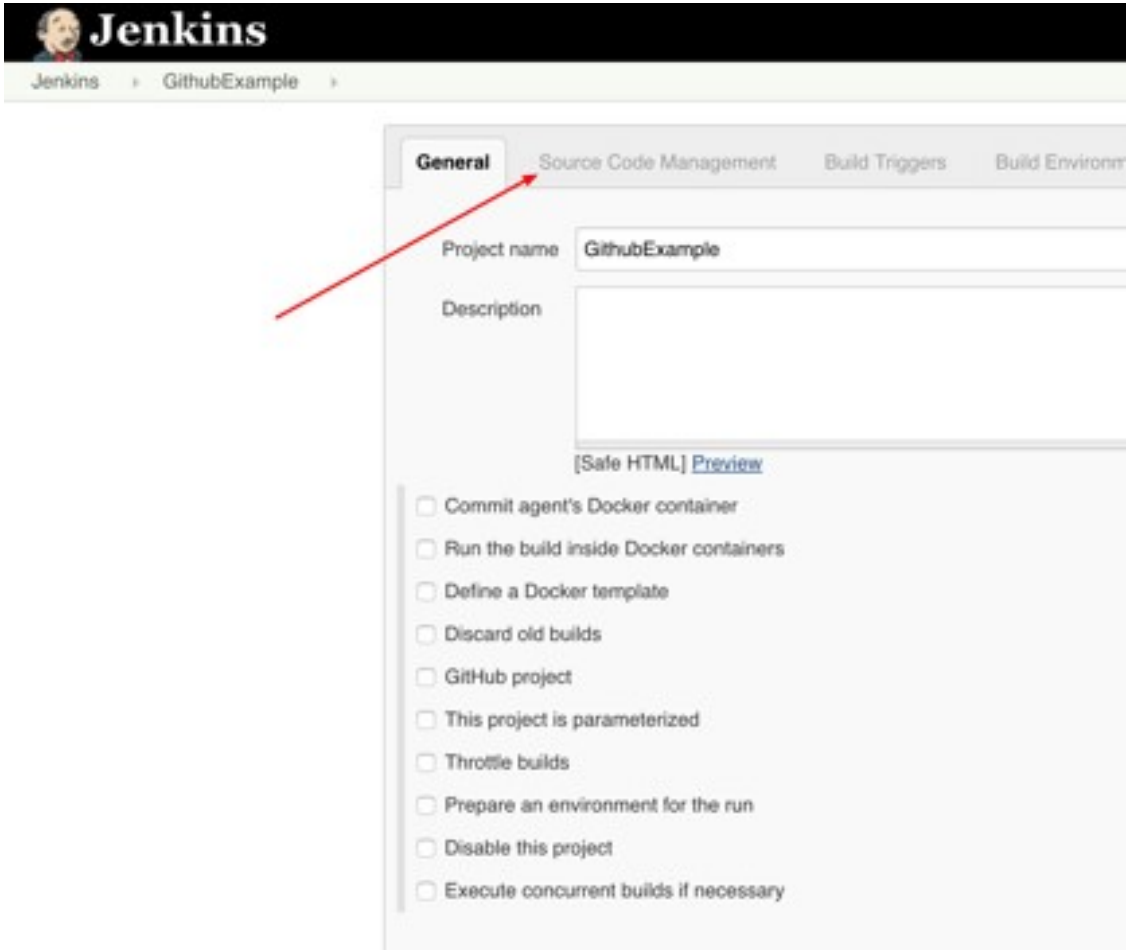
Configuring Jenkins

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



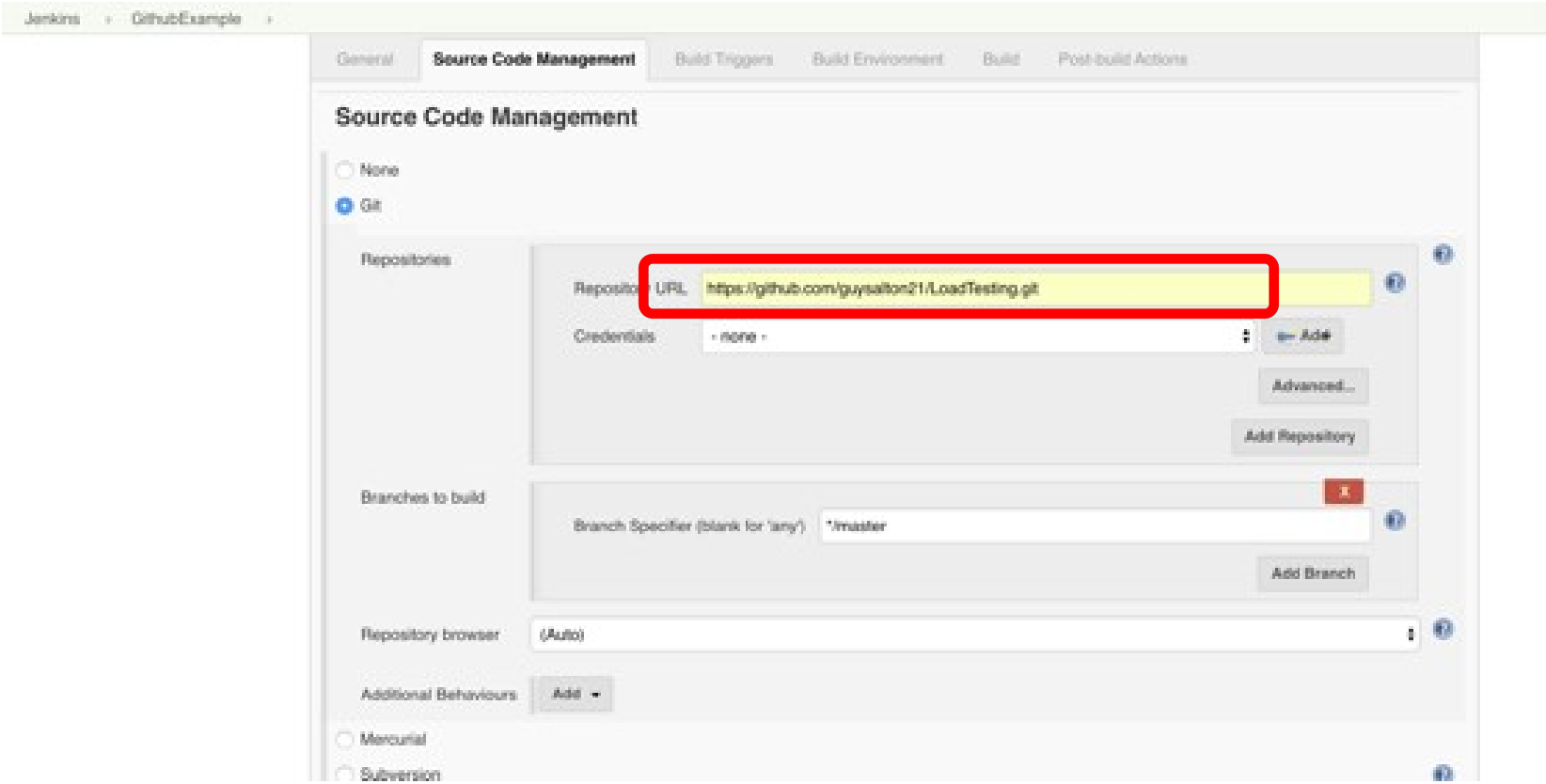
Configuring Jenkins

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



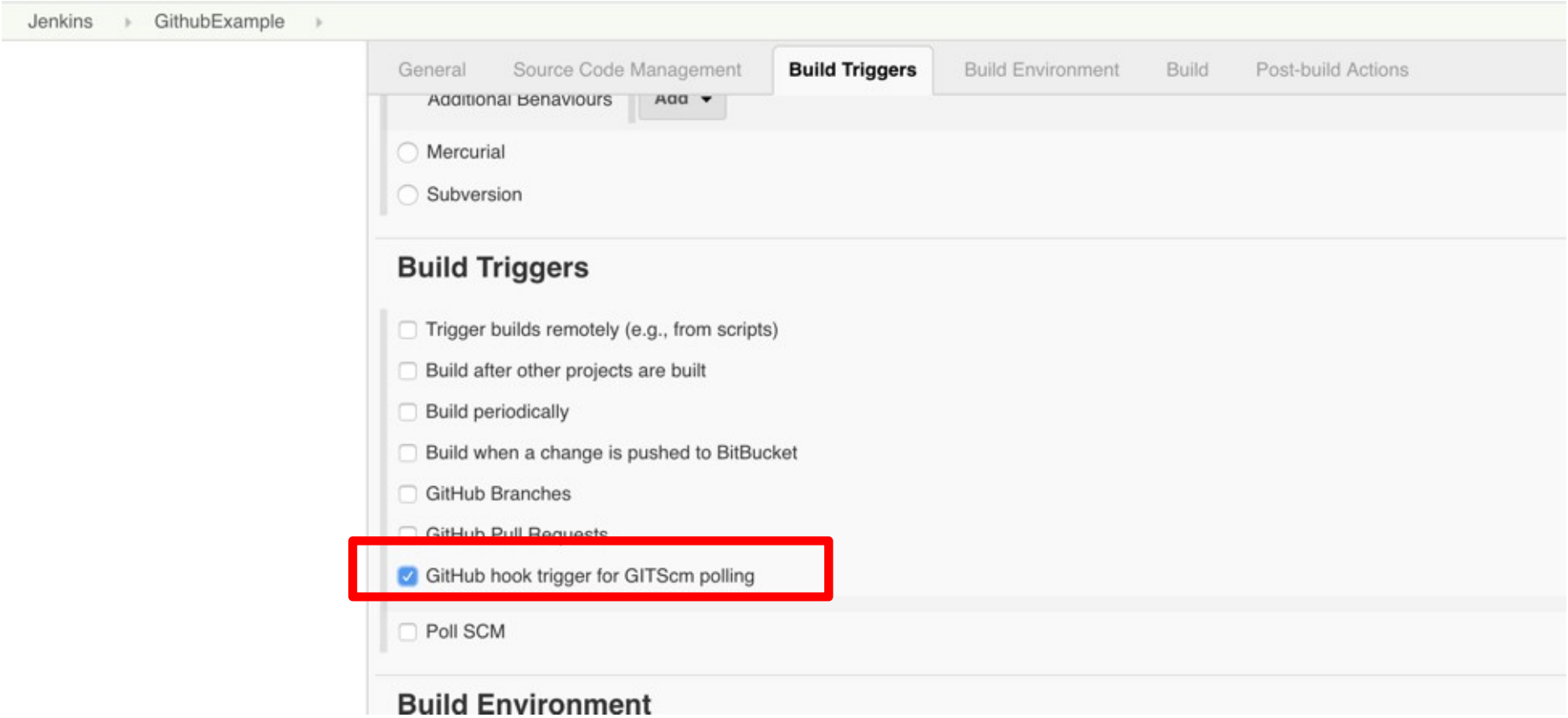
Configuring Jenkins

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



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.



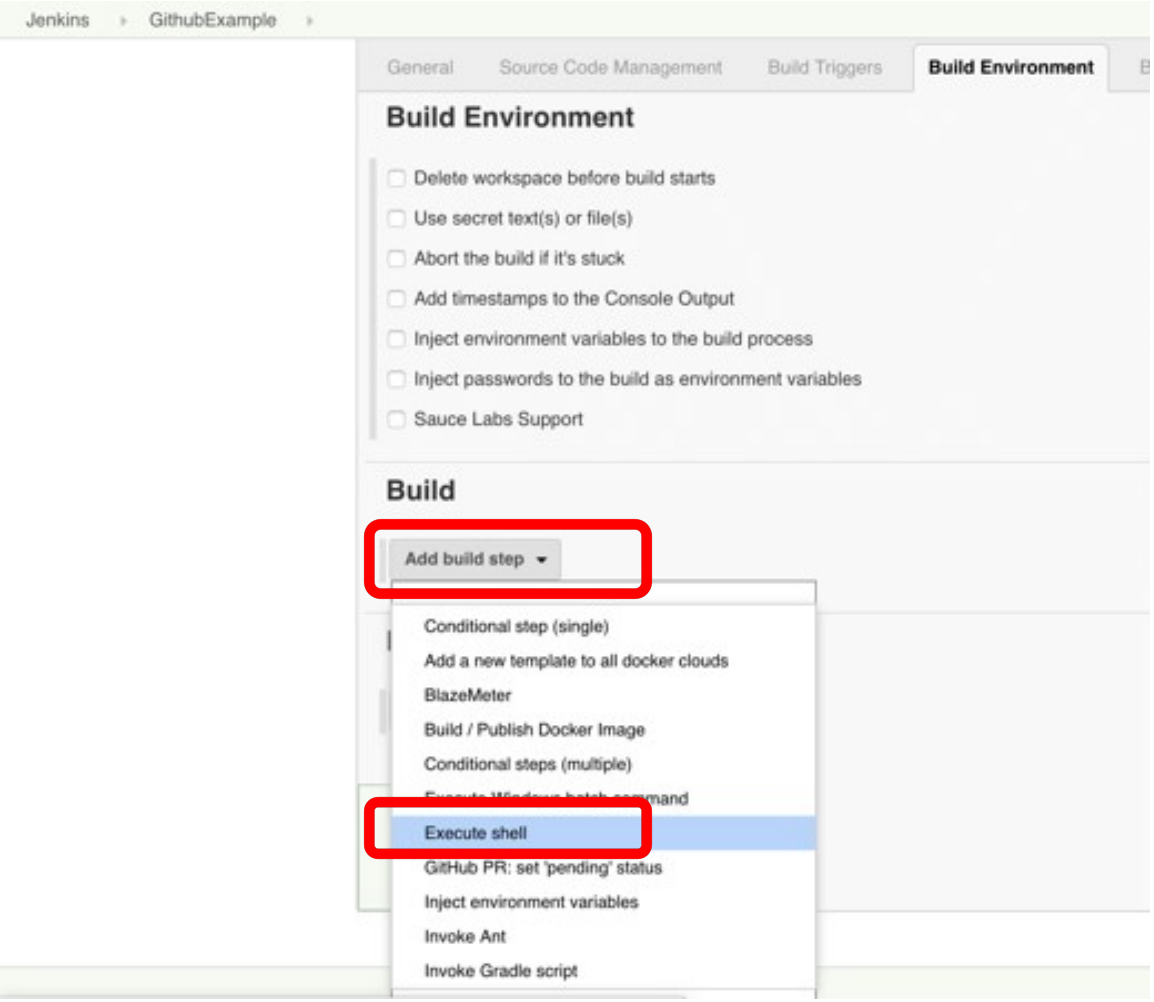
Configuring Jenkins

- 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

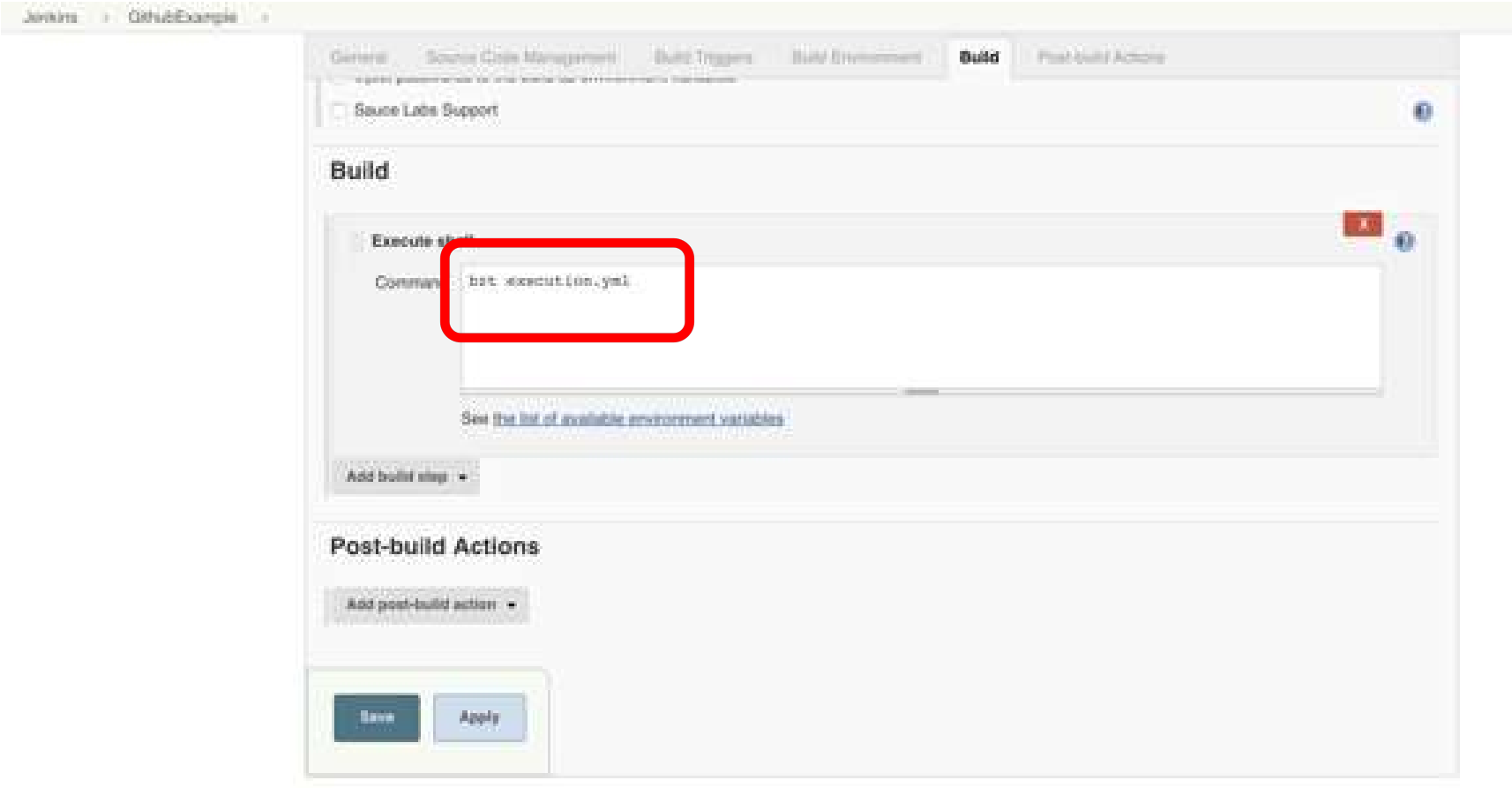
Triggering the Jenkins Job to Run

- 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)"

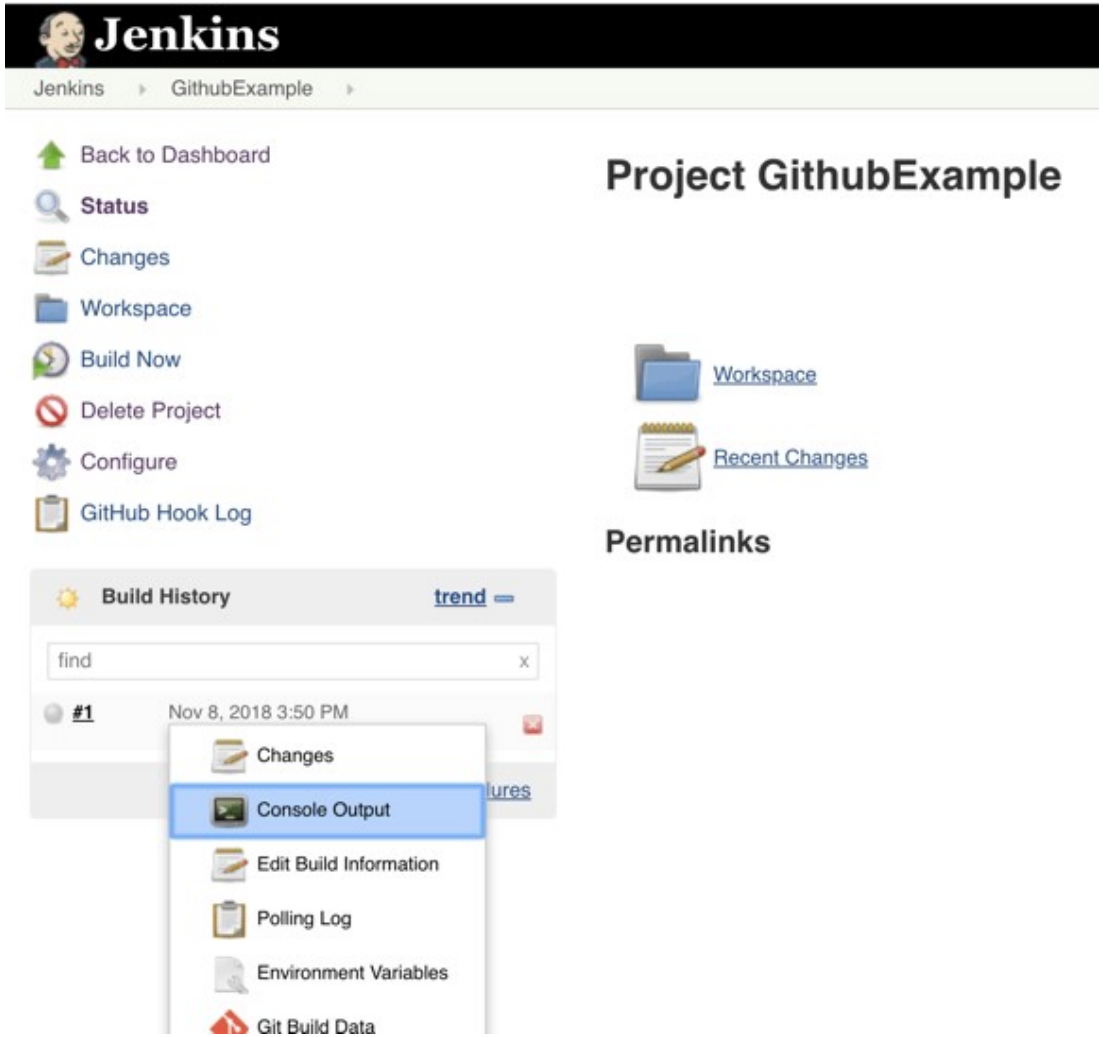


Triggering the Jenkins Job to Run

- 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