**How to Integrate Your GitHub Repository to Your Jenkins Project**

Jenkins and GitHub are two powerful tools on their own, but what about using them together? In this blog, learn about the benefits of a Jenkins GitHub integration and how to set up the integration on your own.

**Table of Contents:**

* [Can You Use Jenkins With GitHub?](https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project#can)
* [Why Integrate GitHub + Jenkins?](https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project#why)
* [How to Set Up the Jenkins + GitHub Integration](https://www.blazemeter.com/blog/how-to-integrate-your-github-repository-to-your-jenkins-project#how)

**Can You Use Jenkins With GitHub?**

You can and should use Jenkins with GitHub to save time and keep your project up-to-date.

One of the basic steps of implementing CI/CD is integrating your [SCM (Source Control Management)](https://www.blazemeter.com/blog/scm-version-control) 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.

**What is GitHub?**

GitHub is a Git-based repository host, commonly used for open source projects. GitHub enables code collaboration, hosting, and versioning.

📕 Related Resource: Learn more about [Selenium + GitHub: A Step-by-Step Testing Guide](https://www.blazemeter.com/blog/selenium-github)

**What is Jenkins?**

Jenkins is an open-source Continuous Integration and Continuous Deployment (CI/CD) tool for automating the software development life cycle (SDLC). With [Jenkins testing](https://www.blazemeter.com/solutions/jenkins), teams can automate the building, testing, and deploying of code.

**Why Integrate GitHub + Jenkins?**

You should integrate Jenkins and GitHub to improve the efficiency of building, testing, and deploying your code.

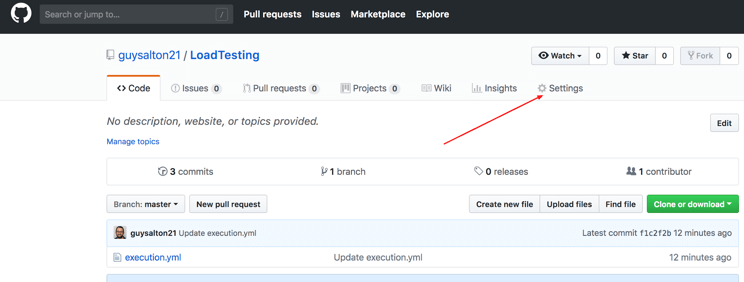
The integration presented in this blog post will teach you how to schedule your build, pull your code and data files from your GitHub repository to your Jenkins machine, and automatically trigger each build on the Jenkins server after each Commit on your Git repository.

But first, let’s configure the Jenkins and GitHub integration. Let's begin with the GitHub side!

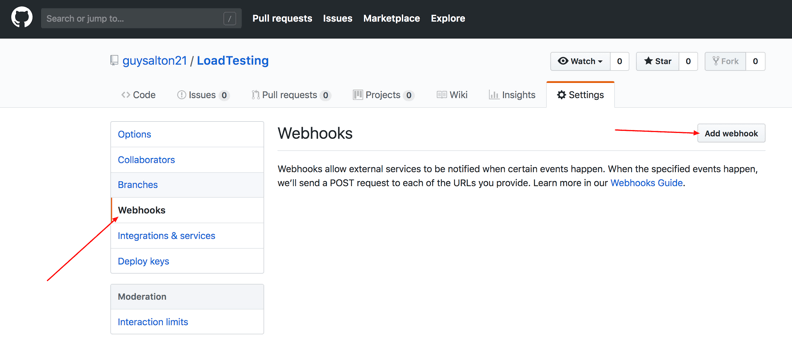
**How to Set Up the Jenkins + GitHub Integration**

**Configuring GitHub**

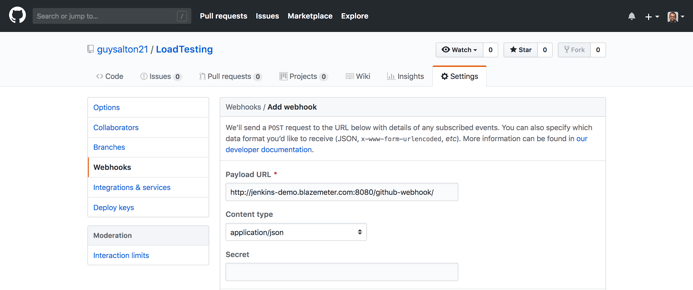
**Step 1:** go to your GitHub repository and click on **‘Settings’.**



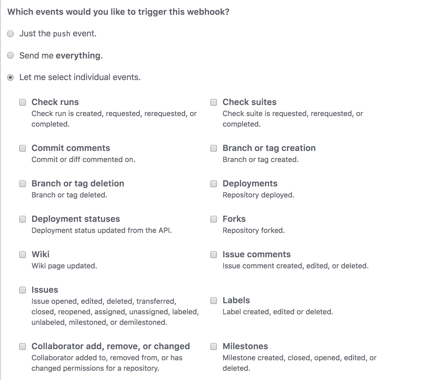
**Step 2:** Click on **Webhooks** and then click on **‘Add webhook’.**

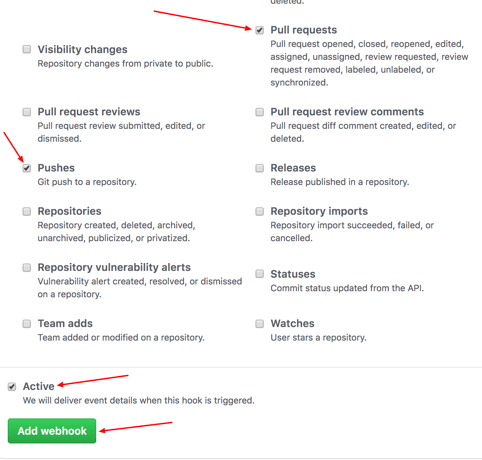


**Step 3:** In the **‘Payload URL’** field, paste your Jenkins environment URL. At the end of this URL add */github-webhook/*. In the**‘Content type’** select: *‘application/json’*and leave the **‘Secret’**field empty.



**Step 4:** In the page **‘Which events would you like to trigger this webhook?’** choose *‘Let me select individual events.’* Then, check *‘Pull Requests’* and *‘Pushes’*. At the end of this option, make sure that the *‘Active’* option is checked and click on *‘Add webhook’*.

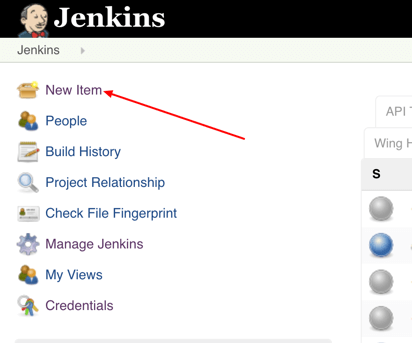




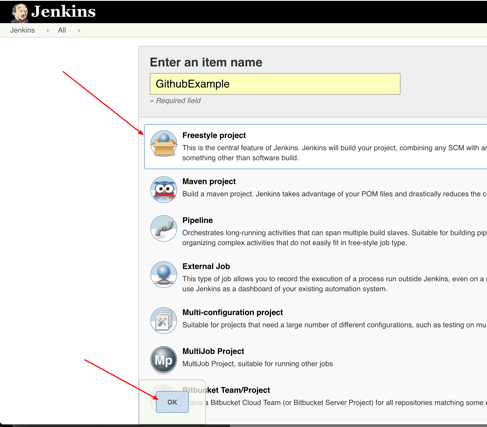
We're done with the configuration on GitHub’s side! Now let's move on to Jenkins.

**Configuring Jenkins**

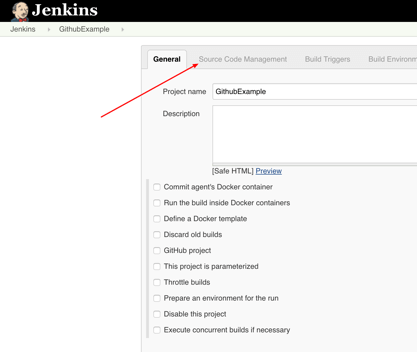
**Step 5:** In Jenkins, click on**‘New Item’** to create a new project.



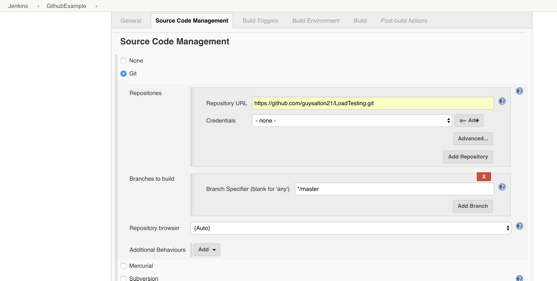
**Step 6:**Give your project a name, then choose *‘Freestyle project’*and finally, click on*‘OK’.*



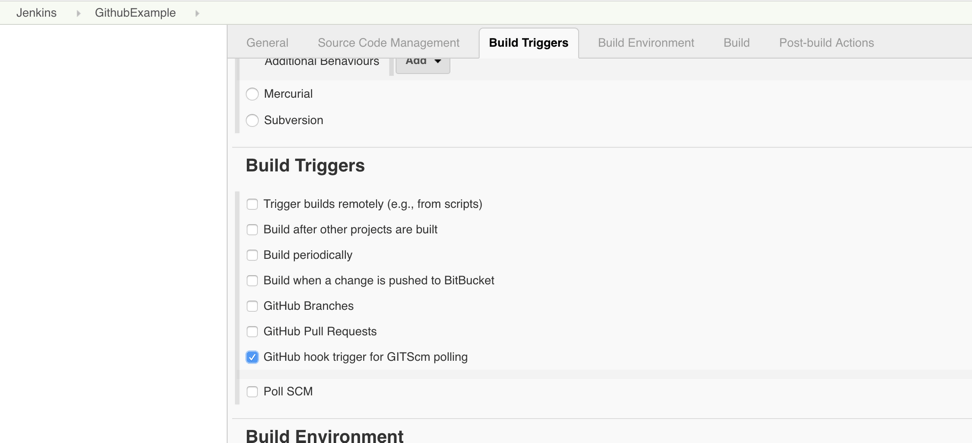
**Step 7:** Click on the **‘Source Code Management’** tab.



**Step 8:** Click on Git and paste your GitHub repository URL in the**‘Repository URL’** field.



**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.

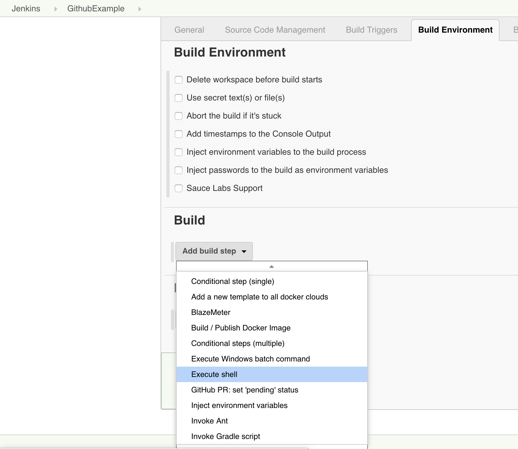


That's it! Your GitHub repository is integrated with your Jenkins project. With this Jenkins GitHub integration, you can now use any file found in the GitHub repository and trigger the Jenkins job to run with every code commit.

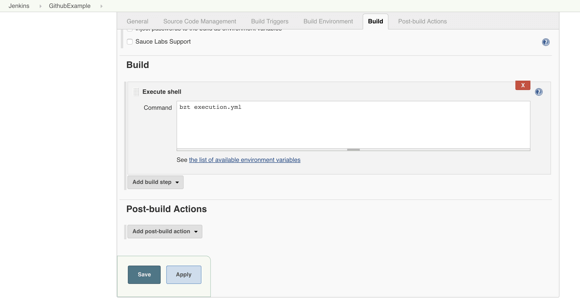
For example, I will show you how to run a Taurus script that I uploaded to my GitHub repository from my Jenkins project. [Taurus](http://gettaurus.org/) is an open source load testing solution, enabling developers to run load testing scripts from sophisticated platforms like JMeter and Selenium, but with a simple YAML code.

**Triggering the Jenkins GitHub Integration With Every Code Commit**

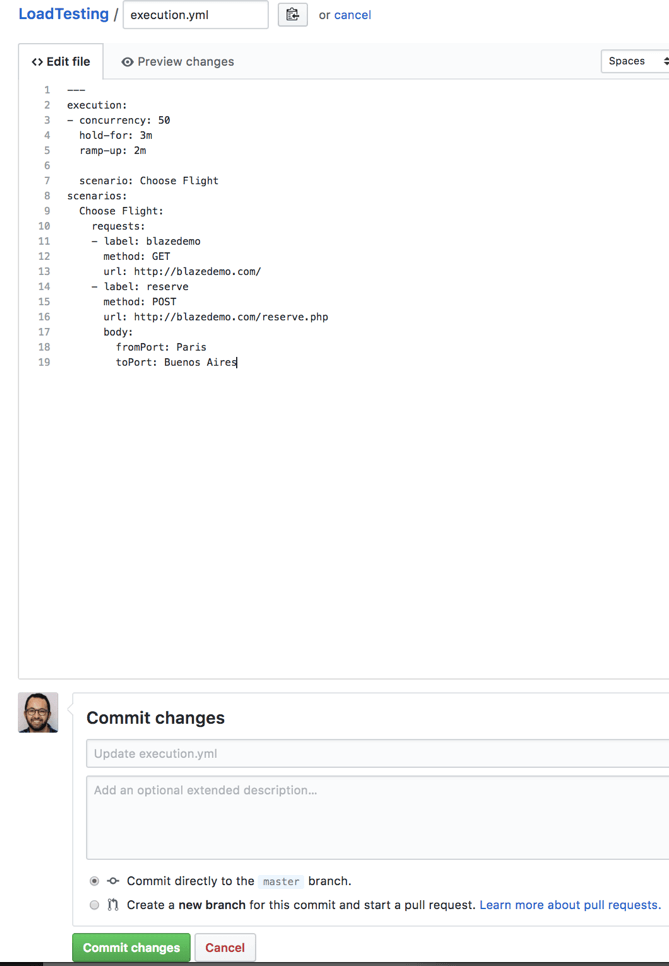
**Step 10:**Click on the **‘Build’** tab, then click on *‘Add build step’*and choose *‘Execute shell’.*



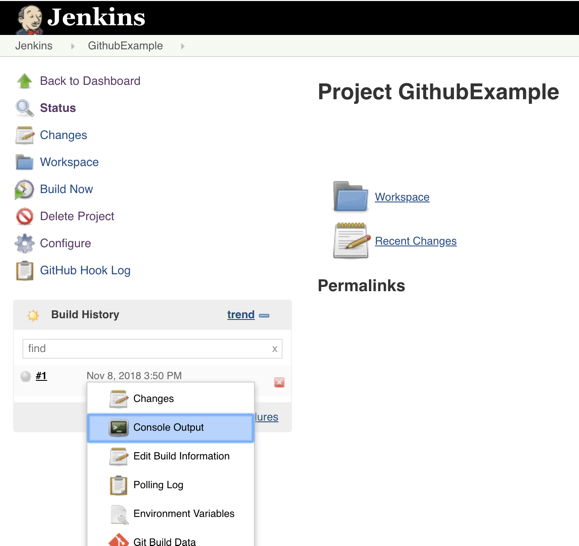
**Step 11:** To run a Taurus test, simply use the **‘bzt’** command, followed by the name of your YML file and click on**‘Save’.**



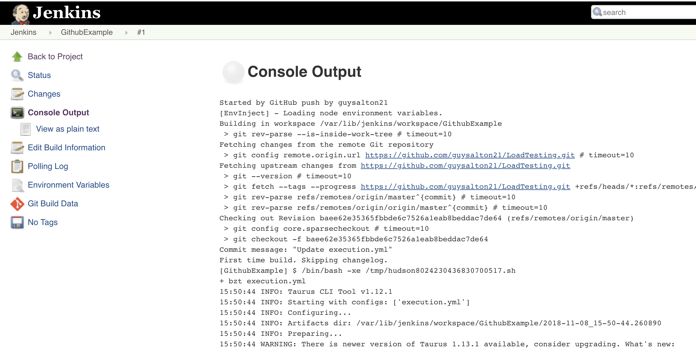
**Step 12:**Go back to your GitHub repository, edit the Taurus script and commit the changes. We will now see how Jenkins ran the script after the commit.



**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’.**



**Step 14:**You can see that Jenkins was able to pull the Taurus script and run it!



**Bottom Line**

Congratulations! You have now set up a fully functioning Jenkins GitHub integration. Every time you publish your changes to Github, GitHub will trigger your new Jenkins job.

Once the GitHub plugin is fully installed and integrated into your Jenkins project, you have completed a very crucial step towards the full CI process. Now, you can proceed to the testing phase. To complete the full CI process, integrate your load testing into your CI tool. Using the BlazeMeter integration with Jenkins, incorporating load testing into your CI/CD pipeline has never been simpler.