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

One of the basic steps of implementing [CI/CD](http://info.blazemeter.com/automated-performance-tests-in-jenkins-ci-environments?utm_source=BM&utm_medium=resources&utm_campaign=webinar-automated-performance-tests-jenkins-ci-environments?utm_source=blog&utm_medium=BM_blog&utm_campaign=how-to-integrate-your-github-repository-to-your-jenkins-project) 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. In this blog we will explain how to integrate [Jenkins](http://info.blazemeter.com/automated-performance-tests-in-jenkins-ci-environments?utm_source=BM&utm_medium=resources&utm_campaign=webinar-automated-performance-tests-jenkins-ci-environments?utm_source=blog&utm_medium=BM_blog&utm_campaign=how-to-integrate-your-github-repository-to-your-jenkins-project) with GitHub projects.

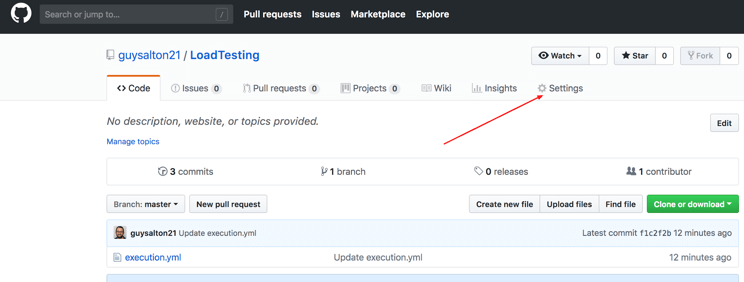
The integration presented in this blog post will teach you to:

* Schedule your build
* Pull your code and data files from your GitHub repository to your [Jenkins](https://jenkins.io/) machine
* Automatically trigger each build on the Jenkins server, after each Commit on your Git repository

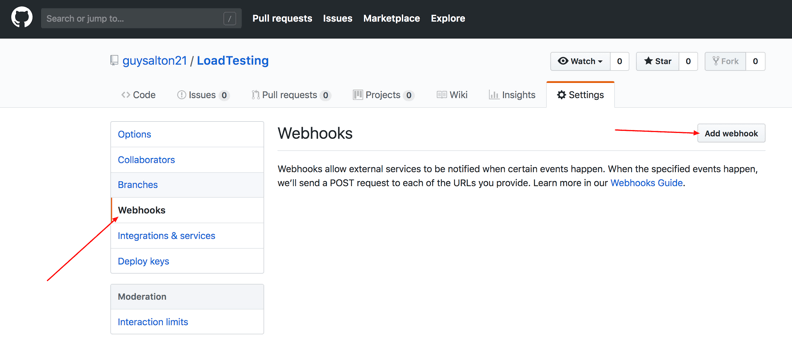
But first, let’s configure the GitHUb and Jenkins integration. Let's begin with the GitHub side!

**Configuring GitHub**

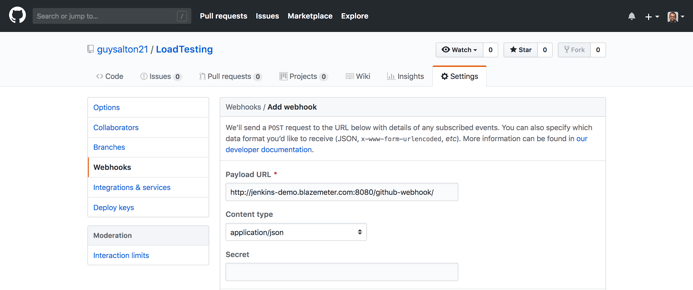
Step 1: go to your GitHub repository and click on ‘Settings’.



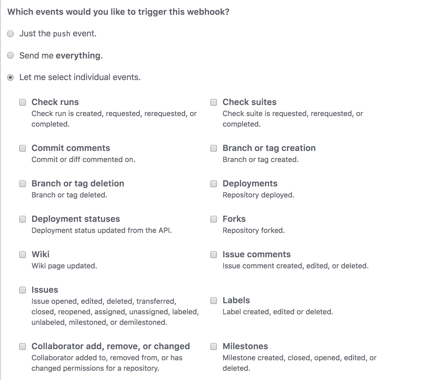
Step 2: Click on Webhooks and then click on ‘Add webhook’.

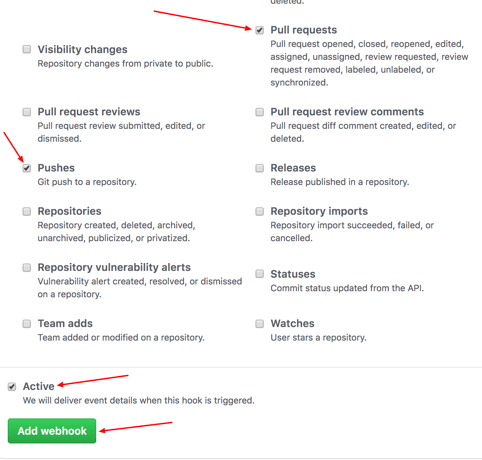


Step 3: in the ‘Payload URL’ field, paste your Jenkins environment URL. At the end of this URL add /github-webhook/. In the ‘Content type’ select ‘application/json’ and leave the ‘Secret’ field empty.



Step 4: in the ‘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 ‘Active’ is checked and click on ‘Add webhook’.

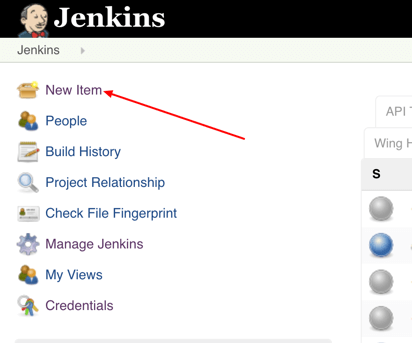




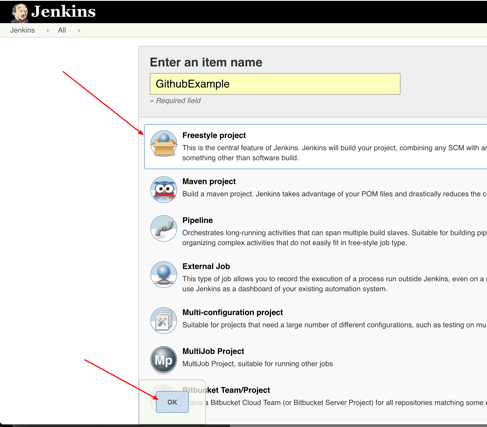
We're done with the configuration on GitHub’s side! Now let's move on to Jenkins.

**Configuring Jenkins**

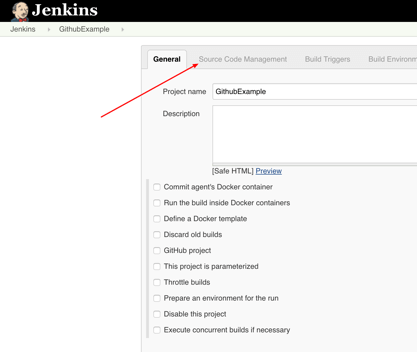
Step 5: In Jenkins, click on ‘New Item’ to create a new project.



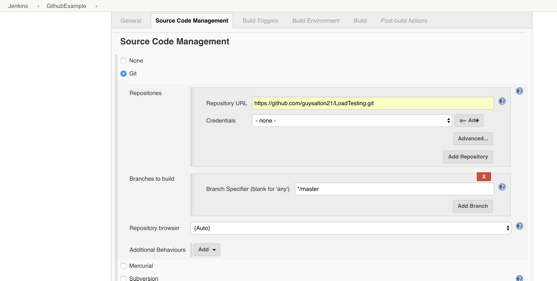
Step 6: Give your project a name, them choose ‘Freestyle project’ and finally click on ‘OK’.



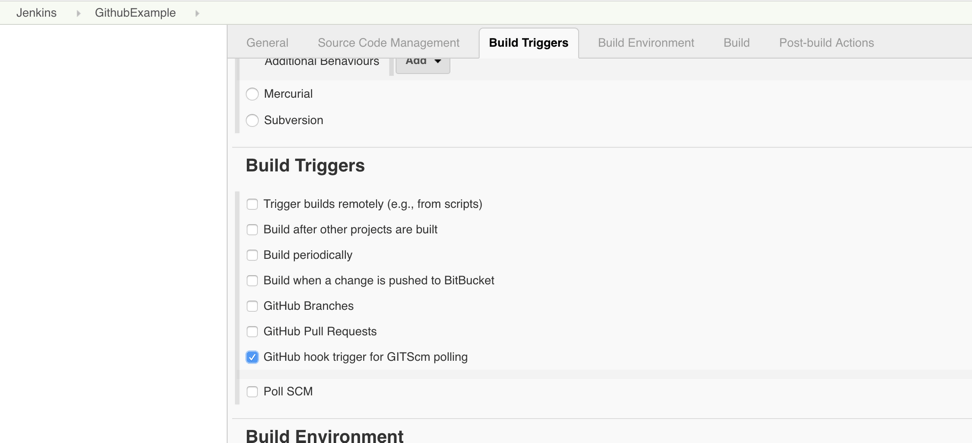
Step 7: Click on the ‘Source Code Management’ tab.



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.

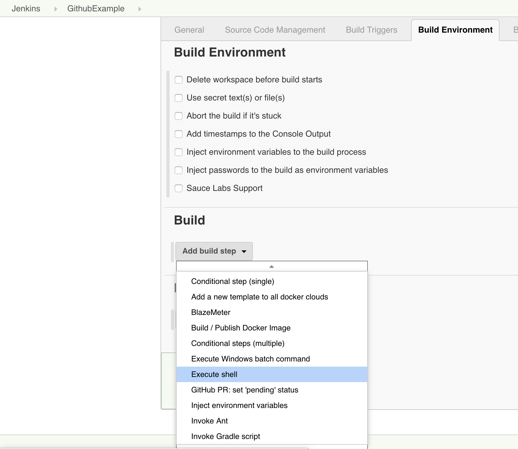


That's it! Your GitHub repository is integrated with your Jenkins project. You can now use any of the files found in the GitHub repository and trigger the Jenkins job to run with every code commit.

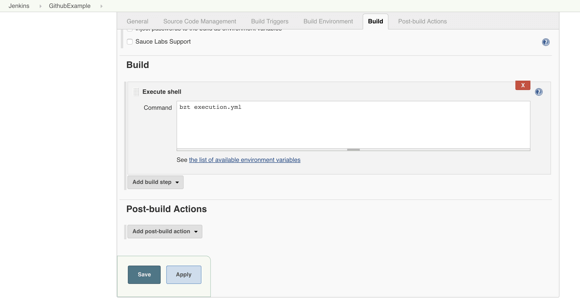
For example, I will show you how to run a Taurus script that I uploaded to my GitHub repository from my Jenkins project.

**Triggering the Jenkins Job to Run with Every Code Commit**

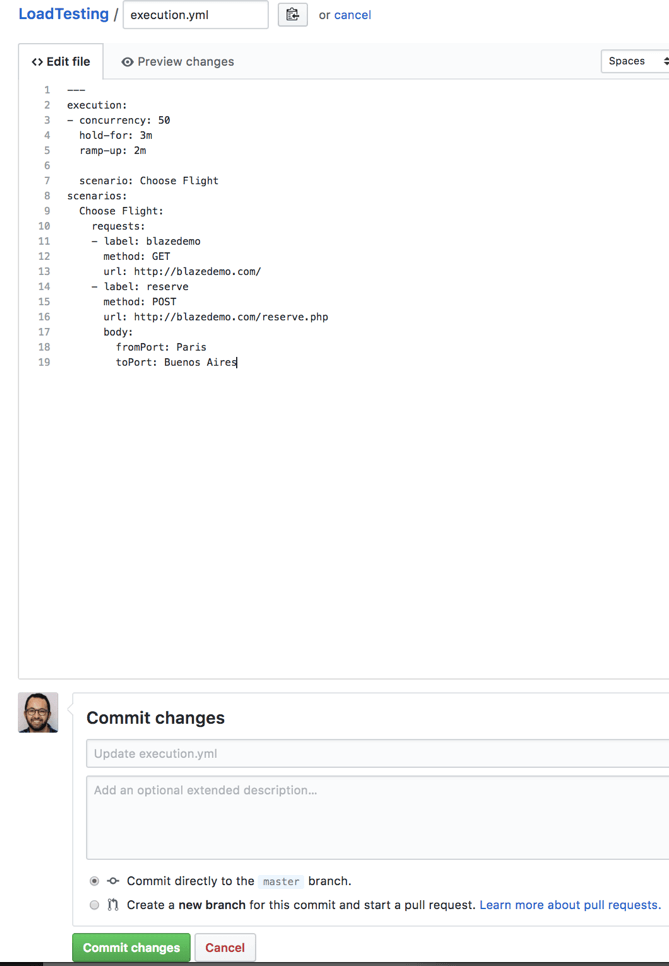
Step 10: Click on the ‘Build’ tab, then click on ‘Add build step’ and choose ‘Execute shell’.



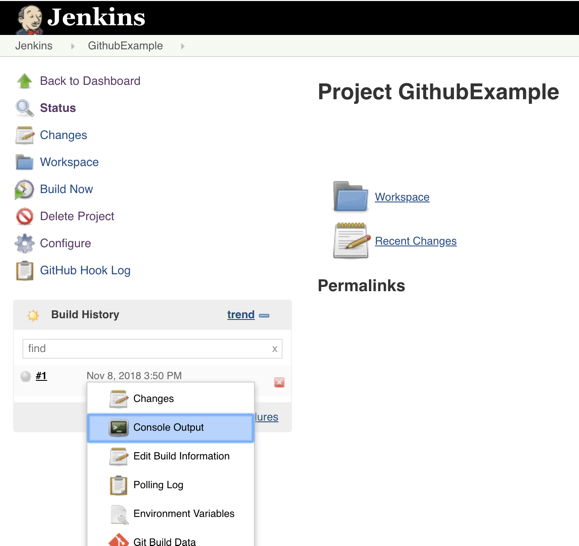
Step 11: To run a Taurus test, simply use the ‘bzt’ command, followed by the name of your YML file and click on ‘Save’.



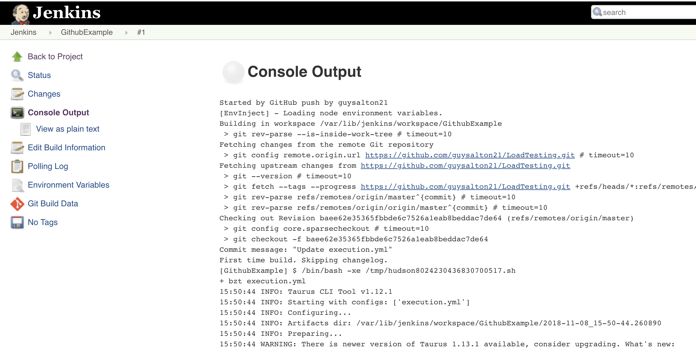
Step 12: Go back to your GitHub repository, edit the Taurus script and commit the changes. We will now see how Jenkins ran the script after the commit.



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!



Congratulations! 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](http://https/info.blazemeter.com/testing-landing-page2?utm_source=blog&utm_medium=BM_blog&utm_campaign=how-to-integrate-your-github-repository-to-your-jenkins-project) into your CI tool. For more details about how to do this with Taurus, an open source automation tool, [see here.](https://www.blazemeter.com/blog/are-you-using-jenkins-right-way?utm_source=blog&utm_medium=BM_blog&utm_campaign=how-to-integrate-your-github-repository-to-your-jenkins-project)