

Triggering a Jenkins build from a push to Github



Marc Best

[Follow](#)

Apr 15, 2017 · 3 min read

In this article we will look at setting up the basis for Continuous Integration using Jenkins for orchestration and Github for source control. We will be configuring a Jenkins build to be initiated on a push to a repository.

1. Install Github Integration Plugin

First we need to install the GitHub Integration Plugin, this will give us the ability to configure Jenkins to use our Github repository.

The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search filter 'github integration' is applied. The table lists two plugins:

Install	Name	Version
<input checked="" type="checkbox"/>	GitHub Integration Plugin GitHub Integration Plugin for Jenkins	0.1.0-rc22
<input type="checkbox"/>	DotCi Github integration, build management through .ci.yml, environment management through docker.	2.39.9

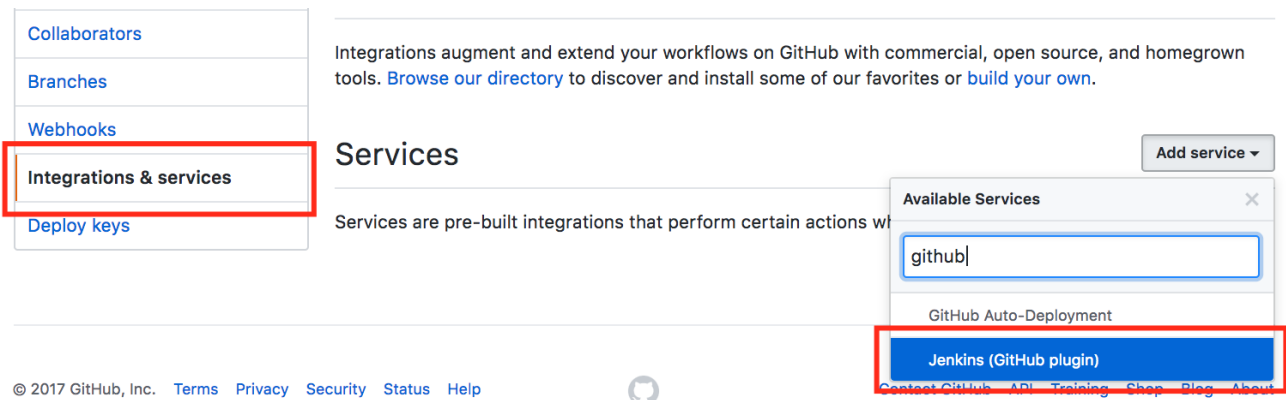
Buttons at the bottom: 'Install without restart', 'Download now and install after restart' (highlighted with a red box), and 'Check now'. A note indicates 'Update information obtained: 3 min 13 sec ago'.

Installing Github integration plugin

2. Prepare Github repository

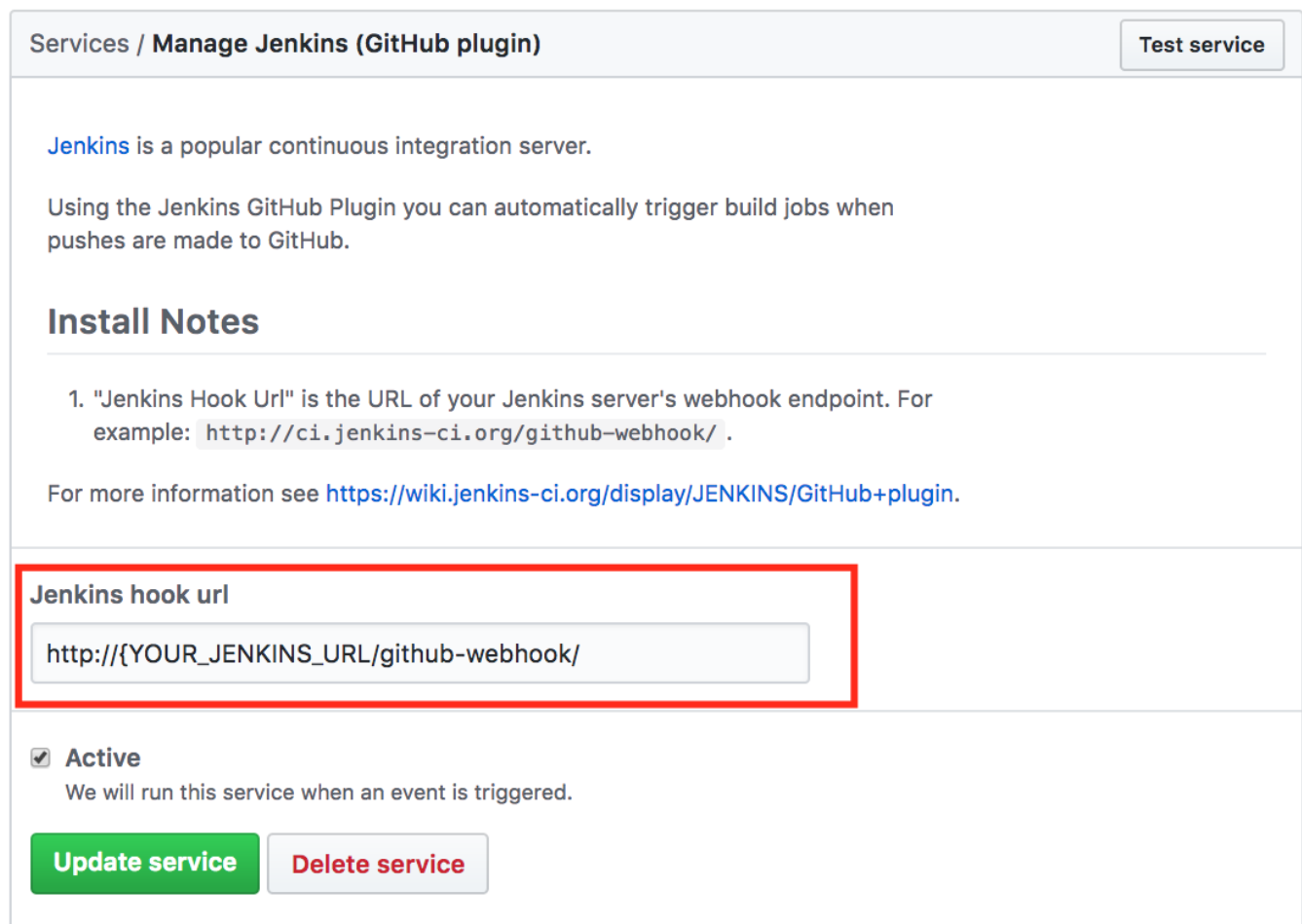
We need to add a service to call the Jenkins Github webhook on a push, to do this go to settings -> integrations & Services and add a new service. The Jenkins Github plugin service should be in the list of available services.

The screenshot shows the Github repository page for 'marcbest / medium-jenkins-git-tut'. The 'Settings' button is highlighted with a red box. Below the repository name, there are tabs for 'Options', 'Installed integrations', and 'Browse directory'.



The screenshot shows the GitHub 'Integrations & services' page. On the left sidebar, 'Integrations & services' is highlighted with a red box. The main content area shows 'Services' with a description: 'Integrations augment and extend your workflows on GitHub with commercial, open source, and homegrown tools. [Browse our directory](#) to discover and install some of our favorites or [build your own](#).' Below this, it says 'Services are pre-built integrations that perform certain actions w'. An 'Add service' button is in the top right. A modal window titled 'Available Services' is open, showing a search bar with 'github' entered. Below the search bar, 'GitHub Auto-Deployment' and 'Jenkins (GitHub plugin)' are listed. 'Jenkins (GitHub plugin)' is highlighted with a red box. At the bottom of the page, there is a footer with copyright information and links: '© 2017 GitHub, Inc. Terms Privacy Security Status Help' and a GitHub logo.

Enter the URL of your Jenkins instance followed by `/github-webhook/`



The screenshot shows the Jenkins 'Manage Jenkins (GitHub plugin)' page. The title bar says 'Services / Manage Jenkins (GitHub plugin)' and has a 'Test service' button. The main content area has a description: 'Jenkins is a popular continuous integration server. Using the Jenkins GitHub Plugin you can automatically trigger build jobs when pushes are made to GitHub.' Below this is a section titled 'Install Notes' with a list of instructions. The first instruction says: '1. "Jenkins Hook Url" is the URL of your Jenkins server's webhook endpoint. For example: `http://ci.jenkins-ci.org/github-webhook/`.' Below this, it says 'For more information see <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+plugin>.' Below the notes is a section titled 'Jenkins hook url' which is highlighted with a red box. It contains a text input field with the value `http://{YOUR_JENKINS_URL}/github-webhook/`. Below the input field is a checkbox labeled 'Active' which is checked. Below the checkbox is a text label: 'We will run this service when an event is triggered.' At the bottom of the section are two buttons: 'Update service' (green) and 'Delete service' (red).

3. Giving the Jenkins user access to the Github repository

We need to give the Jenkins user access to our repository by adding a deploy key in the Github settings.

The first step is generating SSH keys for the Jenkins user if they do not already exist.

```
jenkins@ip:/home/ubuntu$ ssh-keygen
```

Depending on where the key was created, you need to copy the public key so that it can be added to Github

```
jenkins@ip:/home/ubuntu$ cat /var/lib/jenkins/.ssh/id_rsa.pub
```

Add the key copied in the previous step to Github. To do this head to the repository settings -> Deploy keys

marcbest / medium-jenkins-git-tut

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Options Collaborators Branches Webhooks Integrations & services Deploy keys

Deploy keys

Add deploy key

There are no deploy keys for this repository

Title: Jenkins

Key: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCoD1zZ6nhwzXyQF1p7wiHM1dBDAS1IFycfFgCGuh0R4dHGb bObfvI3wU3ddPk3FNbY5Khzkwc+ZEquIBZyp0wQ/ZU7QtVPBJa0N22sJZ1ijzNcdKu6wS51+qQizKdc+4 w6nGoSf3L/q2kpg6Yb47YfaiwTPZsS0MOBCuel9IYw+O9+rYrd4U8D3cDr/5yghbJJuvzsrHtvx9VYmRT k8nofRFom2IDbALyk4cvOAtI7ofz7UVE0dkqcufowndDa8FrDqPr/iuM8EcnUEhCo6fnALelxdPegvr0OMw LnUkpxQIBDalqfaRL2DApH5fpqf0jXiZs2vCsYCTDJ71b0p/p jenkins@ip-172-31-22-109

☐ Allow write access
Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

The last step is to check that everything is working as expected, as the Jenkins user in your console enter the below to check the connection to Github.

```
jenkins@ip:~/.ssh$ ssh git@github.com
```

If successful you should see the following message

```
Warning: Permanently added the RSA host key for IP address  
'{YOUR_SERVER_IP}' to the list of known hosts.
```

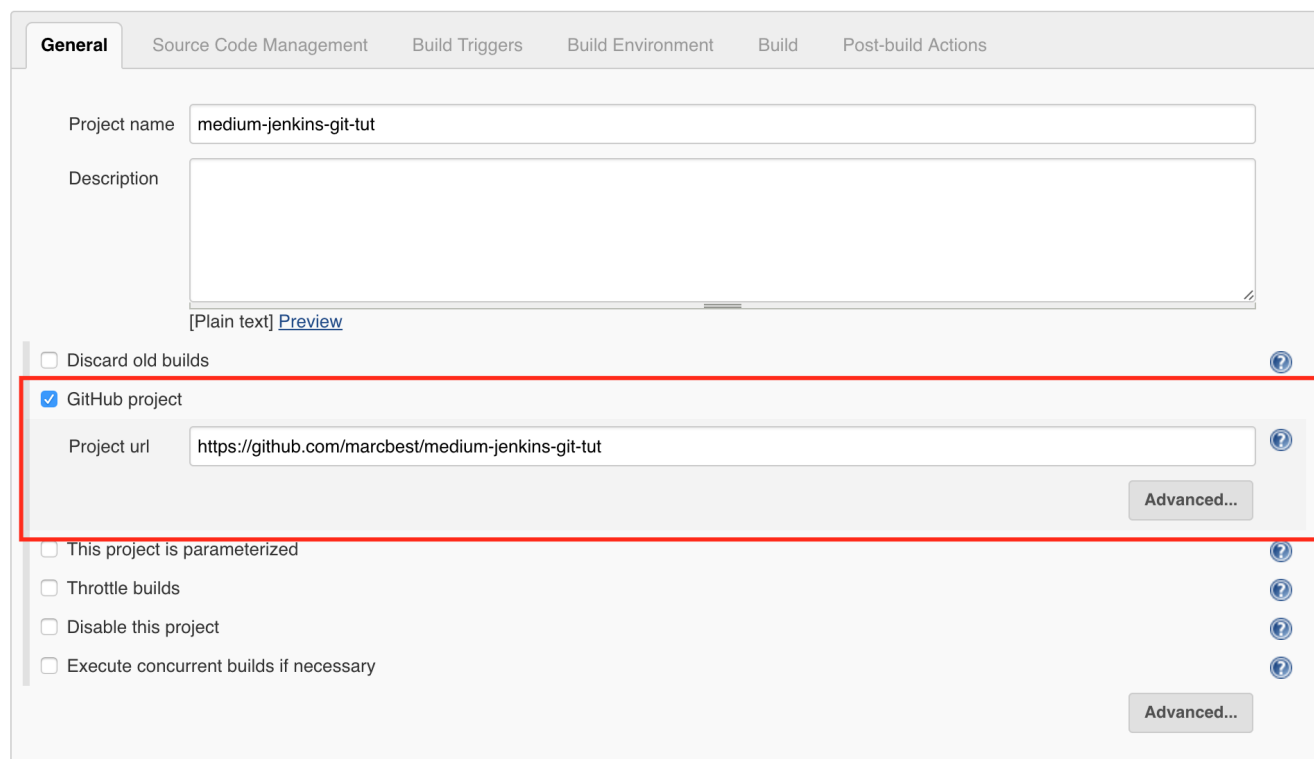
```
PTY allocation request failed on channel 0
```

```
Hi marcbest/medium-jenkins-git-tut! You've successfully  
authenticated, but GitHub does not provide shell access.
```

```
Connection to github.com closed.
```

4. Update Jenkins job with Github configuration

In the 'General' section of the job configuration check the Github project tick box and enter the URL to the repository that you configured in step 2.



The screenshot shows the Jenkins job configuration interface with the 'General' tab selected. The 'Project name' field is 'medium-jenkins-git-tut'. The 'Description' field is empty. Below the description, there are several checkboxes: 'Discard old builds' (unchecked), 'GitHub project' (checked), 'This project is parameterized' (unchecked), 'Throttle builds' (unchecked), 'Disable this project' (unchecked), and 'Execute concurrent builds if necessary' (unchecked). The 'GitHub project' section is highlighted with a red box. It contains the 'Project url' field with the value 'https://github.com/marcbest/medium-jenkins-git-tut' and an 'Advanced...' button. There are also help icons (question marks) next to the 'GitHub project' checkbox and the 'Project url' field.

Next update the Source Code Management section, first set the repository URL (note the format `git@github.com:{YOUR_REPO}`). You can also specify the branch you would like to use.

Source Code Management

☐ None
☒ Git

Repositories

Repository URL

Credentials [Add](#)

[Advanced...](#)

[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any')

[Add Branch](#)

The last step is to tell Jenkins to build when the Github hook is called, select the highlighted option below in the Build Triggers section.

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☐ GitHub Branches

☐ GitHub Pull Requests

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

That's it! Your Jenkins build should now be triggered whenever a push is made to your repository.

[Jenkins](#) [Continuous Integration](#) [Ci](#) [DevOps](#) [Github](#)

[About](#) [Help](#) [Legal](#)