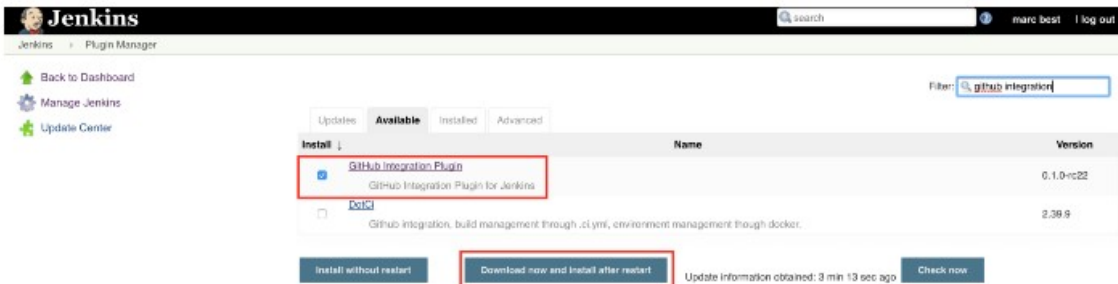


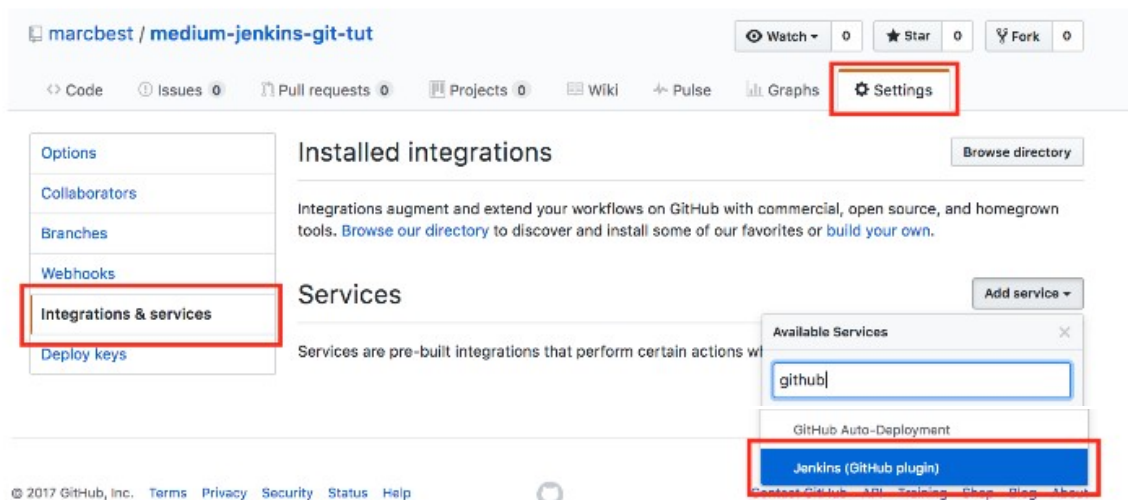
1. Install Github Integration Plugin

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



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.



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

Services / Manage Jenkins (GitHub plugin) Test service

Jenkins is a popular continuous integration server.

Using the Jenkins GitHub Plugin you can automatically trigger build jobs when pushes are made to GitHub.

Install Notes

- "Jenkins Hook Url" is the URL of your Jenkins server's webhook endpoint. For example: `http://ci.jenkins-ci.org/github-webhook/`.

For more information see <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+plugin>.

Jenkins hook url

`http://{YOUR_JENKINS_URL}/github-webhook/`

☒ **Active**
We will run this service when an event is triggered.

Update service

Delete service

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
AAAAAB3NzaC1yc2EAAAADAQABAAQCoD1zZ6nhwzXyQF1p7wiHM1dBDAS1IfycfFgCGuh0R4dHGb
bObfvI3wU3ddPk3FNbY5Khzkwc+ZEquIBZyp0wQ/ZU7Q1VPBjeQN22sLZ1ljzNcdKu6wS51+qQizKdc+4
w6nGoSf3L/q2kpg6Yb4ZY/aiwTPZsSOMOI8CueI9/Yw+O9+rYrd4U8D3cDr/5yghbJJuvzsrHtvx9VYmRT
k8nofRFom2IDbALyk4cvOAt7ofz7UVE0dkqcufoWndDa8FrDqPr/iuM8EcnUEhCo6fnALelxdPegvr0OMw
LnUkpxQIBDaiqlaRL2DApH5fpqf0jXiZs2vCsYctDJ71b0p/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.
```

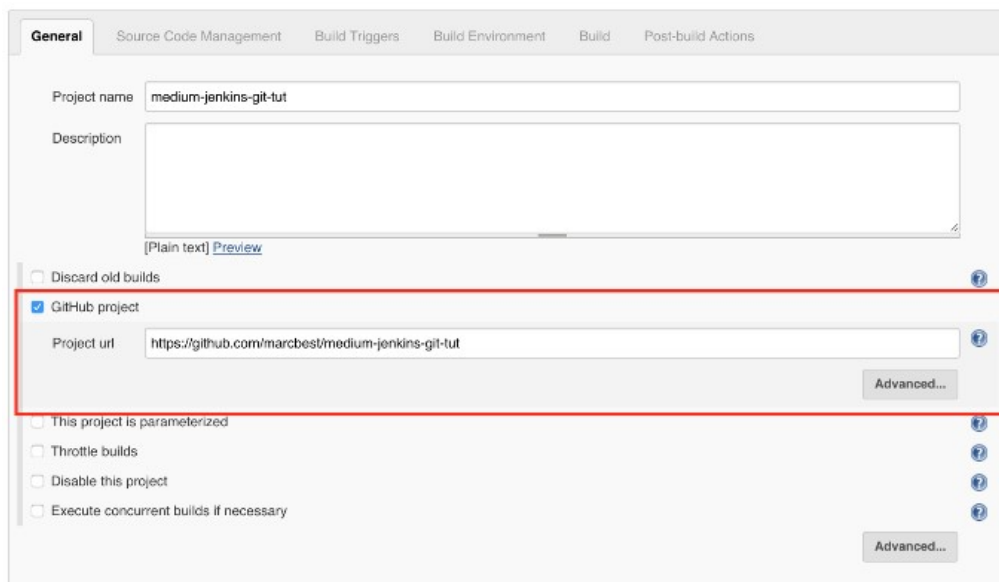
```
PTY allocation request failed on channel 0
```

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

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

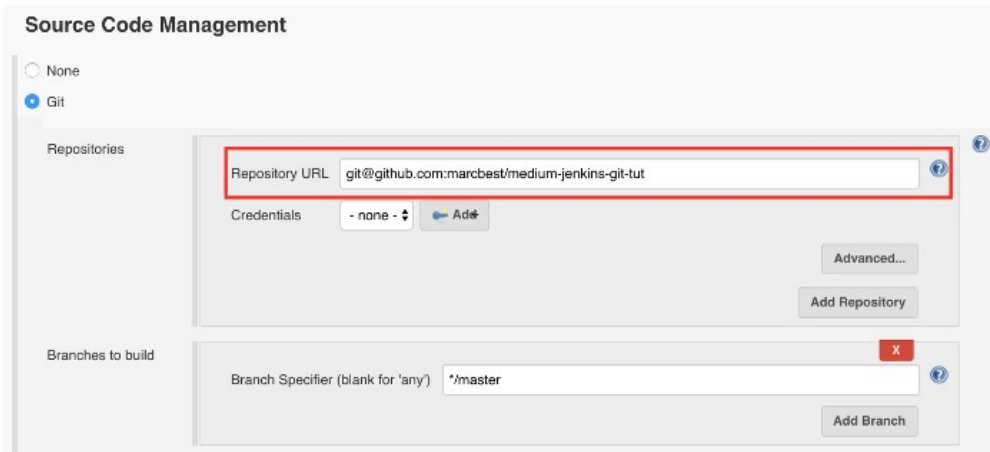
4. Update Jenkins job with Github configuration

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.



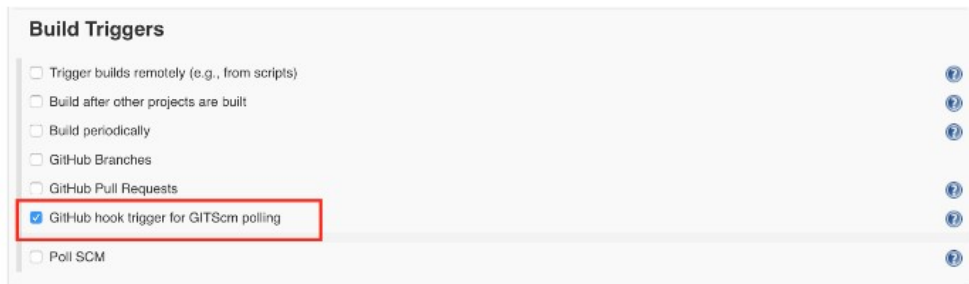
The screenshot shows the Jenkins job configuration interface with the 'General' tab selected. The 'Project name' field is 'medium-jenkins-git-tut'. The 'Description' field is empty. Below the description, there is a link to '[Plain text] Preview'. The 'Discard old builds' checkbox is unchecked. The 'GitHub project' checkbox is checked and highlighted with a red box. The 'Project url' field is 'https://github.com/marcbest/medium-jenkins-git-tut' and is also highlighted with a red box. There is an 'Advanced...' button next to the 'Project url' field. Below the 'GitHub project' section, there are several other checkboxes: 'This project is parameterized' (unchecked), 'Throttle builds' (unchecked), 'Disable this project' (unchecked), and 'Execute concurrent builds if necessary' (unchecked). There is another 'Advanced...' button at the bottom right of the configuration area.

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.



The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' radio button is selected. Under the 'Repositories' section, the 'Repository URL' field is highlighted with a red box and contains the text 'git@github.com:marcbest/medium-jenkins-git-tut'. Below it, the 'Credentials' dropdown is set to '- none -' with an 'Add' button. To the right of these fields are buttons for 'Advanced...', 'Add Repository', and 'Add Branch'. Under the 'Branches to build' section, the 'Branch Specifier (blank for \'any\')' field is highlighted with a red box and contains the text '*/master'. To the right of this field is an 'Add Branch' button. A red 'X' icon is visible in the top right corner of the 'Branches to build' section.

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.



The screenshot shows the 'Build Triggers' configuration page in Jenkins. A list of checkboxes is displayed, with 'Git-Hub hook trigger for GITScm polling' highlighted by a red box and checked. The other options are: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'Git-Hub Branches', 'Git-Hub Pull Requests', and 'Poll SCM'. Each option has a help icon (question mark) to its right.

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