# Configuring PR based trigger in Jenkins

# **Pre-requisite:**

* Jenkins server and github repository

## **Setup**

## we need to install plugin which are mentioned in below table

| **Plugin** | **Version** |
| --- | --- |
| GitHub Pull Request Builder | 1.42.1 |
| Git | 4.2.2 |
| GitHub Plugin | 1.30.0 |
| GitHub API | 1.114.1 |

## **GitHub Pull Request Builder**

## This is the plugin that handles everything related to your pull request. For me, all traditional approaches didn't work and this was the only plugin that actually did what I wanted it to do. Keep in mind: There might be other ways to achieve this!

## This plugin will expose a webhook which GitHub can later use to send meta data to. The webhook is exposed at: <yourJenkins/>/ghprbhook/ A request arriving at the hook is then used to identify a Jenkins build which is actually run then. The plugin will (when configured):

## add a comment to your PR and ask for a review

## add a merge check which you can add to the branch protection requirements for a review

## start the build and send the result back to GitHub using its API

## **Adding Credentials For Authentication With GitHub**

## You will need credentials stored in Jenkins for two things:

## Pulling your source code from GitHub Using the GitHub API to make comments and push the result of the merge check First you have to add some credentials to your Jenkins so that it can later authenticate requests to GitHub. github1

## In the main menu of Jenkins, click on "Credentials":Choose a scope. The global scope is okay. If you have another one you want to use for it, feel free to do so! When you have chosen, click on the scope.

## github2

## Click on "Add Credentials", now.

## github3

## On the following page, choose the kind "Username with password" and fill out all the necessary information. After this, click on the "OK" button and you are finished here.

## github4

## You could also work with an API token here, but for the sake of simplicity, username and password is just more straight forward.

## **Setting The GitHub Project URL**

## Before doing anything more, enter your GitHub project URL. Many posts or articles do not mention this enough, but the url is part of the routing mechanism, that decides what project to actually build when your webhook receives a message. If this field is left blank, no build will ever run!

## github5

## **Adding The Repository**

## In order for your Jenkins to actually build your project, you have to add the git coordinates. Go to your project and just copy the HTTPS clone link. After that, choose the credentials you created earlier from the dropdown. In your case, that red error message should, of course, not pop up. If it does, there is something wrong with your credentials (check your password again!) or your Jenkins can't reach GitHub.

## github6

## The goal is to build individual Pull Requests before they are merged into your main branch. As they originate from another branch as your main one, Jenkins somehow needs to get told which specific branch to build. By adding a refspec and a branch specifier, including variables that are set by the plugin, the job will always build what is specified by the webhook trigger, that comes from GitHub itself.

## github10

## The refspec used is: +refs/pull/${ghprbPullId}/\*:refs/remotes/origin/pr/${ghprbPullId}/\*

## Branch Specifier (blank for 'any') is : ${ghprbActualCommit}

## **Enabling GitHub Pull Request Builder And Setting It Up**

## You are nearly finished with setting up the Jenkins job now. It's time to activate the plugin within your job and make all settings necessary for it to work properly.

## Enable "GitHub Pull Request Builder" and tick the checkbox "Use github hooks for build triggering". Those two are the basic settings you need to get everything working. However, for security reasons, not everyone that creates a PR in your repository can automatically also trigger a build. That's a great feature to prevent misuse, spam, ddos, etc.

## github8

## .\*\[skip\W+ci\].\*

## You could, however, just tick the option 'Build every pull request automatically without asking (Dangerous!).' and whitelist basically everybody, but I would advice against it. It's better to add an organization or add people individually, and only yourself as the Admin.

## The last things to do are setting up the triggers. You can find them here, below all the other plugin settings.

## github9

## Give your build some context. This is a string that will later be shown on your merge checks. Click on "Add" and choose "Update commit status during build"

## github11

## And lastly, click on "Add" again and choose "Build Status Messages". Then add 3 messages for each possible build result.

## github12

## Save your build, and you are done with Jenkins, for now. Whenever Jenkins receives the correct payload at /ghprbhook/, your build will get triggered.

## What you now have to do is setting up your GitHub project to actually use the webhook.

## **Setting Up Your GitHub Repository Adding the Webhook**

## Navigate to your GitHub repository and choose "Settings" from the repository nav bar and choose "Webhooks" from the new menu appearing.

## github13

## 

## Add a new Webhook there by entering the full url to your Jenkins instance, including the path of the webhook, and choose application/json.

## github14

## http://Jenkins\_machine\_url:8080/ghprbhook/

## Finally, under "Which events would you like to trigger this webhook?" choose "Let me select individual events." and select:

## Commit comments

## Pull requests

## github15

## Interestingly, if you have given the user you use for your Jenkins job admin rights (on GitHub, to the repo), the GitHub Pull Request Builder will add those hook settings itself.

## **Create A Test Pull Request**

## Make any commit to your project on any feature branch and just create a Pull Request for testing purposes. You should now see your Jenkins instance starting the job you created. Let it run through and ensure that it runs successfully.

## **Adding The PR Check**

## When you now want to have those checks on your pull requests, there is one last thing left to do. Navigate to your repository's "Settings" once again and choose "Branches" this time.

## github16

## If you don't have a protection rule, yet, create a new one. Target your main/master branch or whatever is your base branch. The important setting you want to enable is "Require status checks to pass before merging". Under that, you should see your "Jenkins" context, if the pipe has run at least once. Check the box and you are finally finished.

## github17

## After that, your next Pull Request should have checks enabled like this:

## github18

## Congratulations, you've made it!