

GitHub Plugin

Plugin Information

View GitHub [on the plugin site](#) for more information.



Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- [Server-side request forgery](#)
- [CSRF vulnerability and lack of permission checks allows capturing credentials](#)
- [CSRF vulnerability and insufficient permission checks allow capturing credentials](#)

Github Plugin

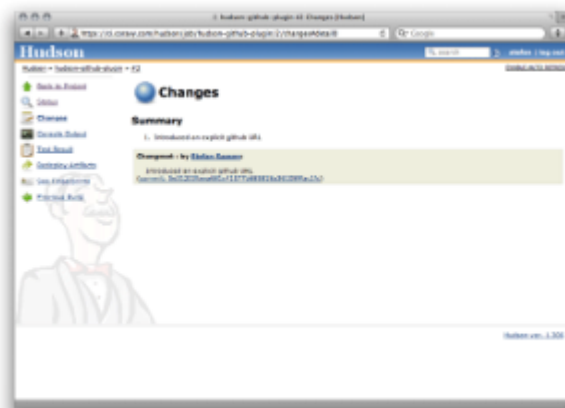
This plugin integrates Jenkins with [Github](#) projects. The plugin currently has three major functionalities:

- Create hyperlinks between your Jenkins projects and GitHub
- Trigger a job when you push to the repository by groking HTTP POSTs from post-receive hook and optionally auto-managing the hook setup.
- Report build status result back to github as [Commit Status \(documented on SO\)](#)
- Base features for other plugins

- [Github Plugin](#)
- [Hyperlinks between changes](#)
- [GitHub hook trigger for GITScm polling](#)
 - [Manual Mode](#)
 - [Automatic Mode \(Jenkins manages hooks for jobs by itself\)](#)
 - [Security Implications](#)
 - [Jenkins inside a firewall](#)
 - [Trouble-shooting hooks](#)
 - [Using cache to GitHub requests](#)
- [Possible Issues between Jenkins and GitHub](#)
 - [Windows:](#)
 - [Pipeline examples](#)
 - [Setting commit status](#)
 - [Change Log](#)
 - [Open Issues](#)

Hyperlinks between changes

The Github plugin decorates Jenkins "Changes" pages to create links to your Github commit and issue pages. It adds a sidebar link that links back to the Github project page.



When creating a job, specify that it connects to git. Under "Github project", put in: `git@github.com:Person/Project.git` Under "Source Code Management" select Git, and put in `git@github.com:Person/Project.git`

GitHub hook trigger for GITScm polling

This feature enables builds after [post-receive hooks in your GitHub repositories](#). This trigger only kicks git-plugin internal polling algo for every incoming event against matched repo.



Old name

Previously named as "Build when a change is pushed to GitHub"

To be able to use this feature:

Manual Mode

In this mode, you'll be responsible for registering the hook URLs to GitHub. Click the icon (under Manage Jenkins > Configure System > GitHub) to see the URL in Jenkins that receives the post-commit POSTs — but in general the URL is of the form `$JENKINS_BASE_URL/github-webhook/` — for example: `https://ci.example.com/jenkins/github-webhook/`.

Once you have the URL, and have added it as a webhook to the relevant GitHub repositories, continue to **Step 3**.

Automatic Mode (Jenkins manages hooks for jobs by itself)

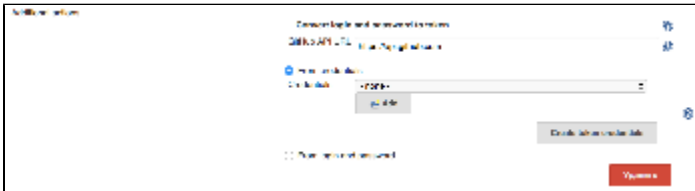
In this mode, Jenkins will automatically add/remove hook URLs to GitHub based on the project configuration in the background. You'll specify GitHub OAuth token so that Jenkins can login as you to do this.

Step 1. Go to the global configuration and add GitHub Server Config.



Step 2.1. Create your personal access token in GitHub.

Plugin can help you to do it with all required scopes. Go to **Advanced** -> **Manage Additional GitHub Actions** -> **Convert Login and Password to token**



Two-Factor Authentication

Auto-creating token doesn't work with [GitHub 2FA](#)

You can create "**Secret text**" credentials with token in corresponding domain with login and password directly, or from username and password credentials.

Step 2.2. Select previously created "Secret Text" credentials with GitHub OAuth token.



Required scopes for token

To be able manage hooks your token should have **admin:org_hook** scope.



GitHub Enterprise

You can also redefine GitHub url by clicking on **Custom GitHub API URL** checkbox.

Note that credentials are filtered by entered GH url with help of domain requirements. So you can create credentials in different domains and see only credentials that matched by predefined domains.



Step 3. Once that configuration is done, go to the project config of each job you want triggered automatically and simply check "Build when a change is pushed to GitHub" under "Build Triggers". With this, every new push to the repository automatically triggers a new build.

Note that there's only one URL and it receives all post-receive POSTs for all your repositories. The server side of this URL is smart enough to figure out which projects need to be triggered, based on the submission.

Security Implications

This plugin requires that you have an HTTP URL reachable from GitHub, which means it's reachable from the whole internet. So it is implemented carefully with the possible malicious fake post-receive POSTs in mind. To cope with this, upon receiving a POST, Jenkins will talk to GitHub to ensure the push was actually made.

Jenkins inside a firewall

In case your Jenkins run inside the firewall and not directly reachable from the internet, this plugin lets you specify an arbitrary endpoint URL as an override in the automatic mode. The plugin will assume that you've set up reverse proxy or some other means so that the POST from GitHub will be routed to the Jenkins.

Trouble-shooting hooks

If you set this up but build aren't triggered, check the following things:

- Click the "admin" button on the GitHub repository in question and make sure post-receive hooks are there.
 - If it's not there, make sure you have proper credential set in the Jenkins system config page.
- Also, [enable logging](#) for the class names
 - `com.cloudbees.jenkins.GitHubPushTrigger`
 - `org.jenkinsci.plugins.github.webhook.WebhookManager`
 - `com.cloudbees.jenkins.GitHubWebHook`and you'll see the log of Jenkins trying to install a post-receive hook.
- Click "Test hook" button from the GitHub UI and see if Jenkins receive a payload.

Using cache to GitHub requests

Each **GitHub Server Config** creates own GitHub client to interact with api. By default it uses cache (with **20MB** limit) to speedup process of fetching data and reduce rate-limit consuming. You can change cache limit value in "Advanced" section of this config item. If you set 0, then this feature will be disabled for this (and only this) config.



Additional info:

- This plugin now serves only hooks from github as main feature. Then it starts using git-plugin to fetch sources.
- It works both public and Enterprise GitHub
- Plugin have some [limitations](#)

Possible Issues between Jenkins and GitHub

Windows:

- In windows, Jenkins will use the the SSH key of the user it is running as, which is located in the %USERPROFILE%\ssh folder (on XP, that would be C:\Documents and Settings\USERNAME\ssh, and on 7 it would be C:\Users\USERNAME\ssh). Therefore, you need to force Jenkins to run as the user that has the SSH key configured. To do that, right click on My Computer, and hit "Manage". Click on "Services". Go to Jenkins, right click, and select "Properties". Under the "Log On" tab, choose the user Jenkins will run as, and put in the username and password (it requires one). Then restart the Jenkins service by right clicking on Jenkins (in the services window), and hit "Restart".
- Jenkins does not support passphrases for SSH keys. Therefore, if you set one while running the initial Github configuration, rerun it and don't set one.

Pipeline examples

Setting commit status

This code will set commit status for custom repo with configured context and message (you can also define same way backref)

```
void setBuildStatus(String message, String state) {
    step([
        $class: "GitHubCommitStatusSetter",
        reposSource: [$class: "ManuallyEnteredRepositorySource", url: "https://github.com/my-org/my-repo"],
        contextSource: [$class: "ManuallyEnteredCommitContextSource", context: "ci/jenkins/build-status"],
        errorHandlers: [[$class: "ChangingBuildStatusErrorHandler", result: "UNSTABLE"]],
        statusResultSource: [ $class: "ConditionalStatusResultSource", results: [[$class: "AnyBuildResult",
message: message, state: state]] ]
    ]);
}
```

```
setBuildStatus("Build complete", "SUCCESS");
```

More complex exa~~m~~e (can be used with multiply scm sources in pipeline)

```
def getRepoURL() {
    sh "git config --get remote.origin.url > .git/remote-url"
    return readFile(".git/remote-url").trim()
}

def getCommitSha() {
    sh "git rev-parse HEAD > .git/current-commit"
    return readFile(".git/current-commit").trim()
}



def updateGithubCommitStatus(build) {
    // workaround https://issues.jenkins-ci.org/browse/JENKINS-38674
    repoUrl = getRepoURL()
    commitSha = getCommitSha()

    step([
        $class: 'GitHubCommitStatusSetter',
        reposSource: [$class: "ManuallyEnteredRepositorySource", url: repoUrl],
        commitShaSource: [$class: "ManuallyEnteredShaSource", sha: commitSha],
        errorHandlers: [[$class: 'ShallowAnyErrorHandler']],
        statusResultSource: [
            $class: 'ConditionalStatusResultSource',
            results: [
                [$class: 'BetterThanOrEqualBuildResult', result: 'SUCCESS', state: 'SUCCESS', message: build.
description],
                [$class: 'BetterThanOrEqualBuildResult', result: 'FAILURE', state: 'FAILURE', message: build.
description],
                [$class: 'AnyBuildResult', state: 'FAILURE', message: 'Loophole']
            ]
        ]
    ])
}
```

Change Log

[GitHub Releases](#)

Open Issues

| T | Key | Summary | Status | Updated | Created |
|---|---------------|--|--------|--------------|--------------|
|  | JENKINS-60901 | GitHub manages hooks even when it has not been configured to do it | OPEN | Feb 28, 2020 | Jan 29, 2020 |
|  | JENKINS-47597 | Jenkins UI wording for removing a GitHub-based multibranch Pipeline project is very misleading | OPEN | Feb 24, 2020 | Oct 20, 2017 |

| | | | | | |
|---|---------------|---|-------------|--------------|--------------|
|  | JENKINS-61133 | Github webhook override breaks CSRF exclusion | OPEN | Feb 18, 2020 | Feb 18, 2020 |
|  | JENKINS-55787 | Switch labels from entry to checkbox | OPEN | Feb 18, 2020 | Jan 25, 2019 |
|  | JENKINS-49332 | Jenkins unable to manage webhooks of Github organization | OPEN | Feb 10, 2020 | Feb 02, 2018 |
|  | JENKINS-60480 | github is deprecating basic authentication using password | OPEN | Feb 04, 2020 | Dec 13, 2019 |
|  | JENKINS-44426 | GitHub still triggering on SCM changes after polling option set to false | OPEN | Jan 16, 2020 | May 22, 2017 |
|  | JENKINS-60738 | Override github hook url is not being saved | OPEN | Jan 16, 2020 | Jan 11, 2020 |
|  | JENKINS-50154 | Webhook delivery failed as X-Hub-Signature did not match calculated | OPEN | Jan 06, 2020 | Mar 13, 2018 |
|  | JENKINS-60612 | Could not match URL git@github.company.com:/GAEBLER/imsartifacts github webhook does not work | OPEN | Dec 30, 2019 | Dec 30, 2019 |
|  | JENKINS-60585 | Jenkins triggers hundreds builds for one commit | OPEN | Dec 25, 2019 | Dec 25, 2019 |
|  | JENKINS-60534 | Git Status Notification Fails With Folders | OPEN | Dec 18, 2019 | Dec 18, 2019 |
|  | JENKINS-60470 | Github Plugin sends build status for skipped scm builds resulting in build failed status | OPEN | Dec 13, 2019 | Dec 13, 2019 |
|  | JENKINS-41377 | Polling has not run yet, although received push event | OPEN | Dec 12, 2019 | Jan 24, 2017 |
|  | JENKINS-45280 | duplicate build triggered by same github push notification when there is long scm polling | OPEN | Nov 30, 2019 | Jul 04, 2017 |
|  | JENKINS-59604 | When saving the Configure System if there is no elements the modification is not saved | OPEN | Oct 01, 2019 | Oct 01, 2019 |
|  | JENKINS-59242 | github webhook triggers the job which is defined ONLY SHOULD be triggered on TAGS | OPEN | Sep 05, 2019 | Sep 05, 2019 |
|  | JENKINS-59088 | Github's pending does not wait until all concurrent builds are finished | OPEN | Aug 26, 2019 | Aug 26, 2019 |
|  | JENKINS-52382 | Can't stop / cancel private repository builds | OPEN | Aug 06, 2019 | Jul 05, 2018 |
|  | JENKINS-51344 | Jackson-Databind needs to be upgraded to 2.9.4+ to address CVE-2018-5968 | IN PROGRESS | Jul 30, 2019 | May 15, 2018 |

Showing 20 out of 106 issues