HOW TO AUTOMATICALLY RUN A JENKINS BUILD AFTER PUSHING CODE CHANGES TO GIT

By Jordan Johnson

Jenkins is an open source tool that lets you automatically build, test and deploy software. It helps you leverage the power of continuous integration and delivery.

This guide will teach you how to trigger a Jenkins build whenever you push changes to your Git repository.

Install a Local Instance of Jenkins on Windows

Download the Windows Jenkins installer here:
https://www.jenkins.io/download/#downloading-jenkins

See the following page for more information on installing Jenkins on various environments:
https://www.jenkins.io/doc/book/installing/

Trigger a Jenkins Build

There are two ways to automatically trigger a build in Jenkins:

1. **Push notifications** – The version control system notifies Jenkins of each new commit.
    a. **Note:** Push notifications are a more efficient method of running automated builds. With push notifications, a build only runs when there is a relevant change.
2. **Polling** – Jenkins polls the repository in regular intervals to see if there have been any changes.

Configure Push Notifications

To use push notifications in Jenkins, you need to configure both Jenkins and your version control system (e.g. Githib, GitLab) so they can communicate with each other.

Note: Each version control system must be configured differently to connect with Jenkins.

Process Overview:

To configure push notifications in Jenkins follow the following process:

1. Install a Jenkins plugin based on your version control system.
2. Configure a repository server hostname.
3. Add an access token or credential.

Note: Jenkins needs to be able to access the host of your Git repository, not just the repository itself. Jenkins can not poll repositories hosted on your local machine by default. Also, Jenkins needs the access token of your version control management system user (e.g. Your Gitlab user).

In Jenkins navigate to **System Configuration > Manage Jenkins**.

You can allow Jenkins to connect to your repository host depending on your version control management system by doing the following:

- Under **Gitlab Web Hook** configuring a Webhook to use with Jenkins.
- Under **GitHub** adding a GitHub server to use with Jenkins.

Note: See the official Jenkins documentation for information on connecting Jenkins with other version control systems.

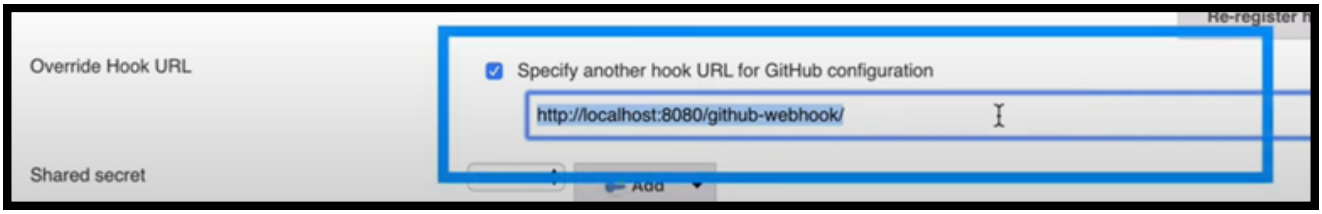
In Gitlab navigate to **Settings > Integrations**.

Under URL enter the Jenkins Webhook URL.

To get the Webhook URL:

- In Jenkins navigate to **System Configuration > Manage Jenkins > GitHub**.
- Click the GitHub Server dropdown and select **GitHub Server** to add a GitHub Server entry.
- Click Advanced.

Click **Specify another hook URL for GitHub configuration**. This is the URL where Jenkins will poll for code changes.



- Under **Trigger** you can select what actions should trigger a build. You can select **Push Events** and enter the repository name Jenkins should monitor.
- If you have Jenkins running locally, you can't run builds automatically because Jenkins can't access localhost. It needs a proper domain.
- Click **Add Web Hook**.
- Now when a developer or tech writer pushes a commit to the repository, GitHub (or the Version Control System) will push a notification to the Jenkins web hook URL which will trigger a Jenkins Build.

Note: Web hook notifications can become unreliable. For example, if the Jenkins server is down, Jenkins likely won't get notified of changes to a repository.

Configure Polling

To configure polling in Jenkins do the following:

1. In Jenkins navigate to **Jenkins Job > Configure**.
2. In your pipeline under **Scan Multibranch Pipeline Triggers** select **Periodically**.
3. Select the interval that Jenkins should poll the changes (e.g. every hour).