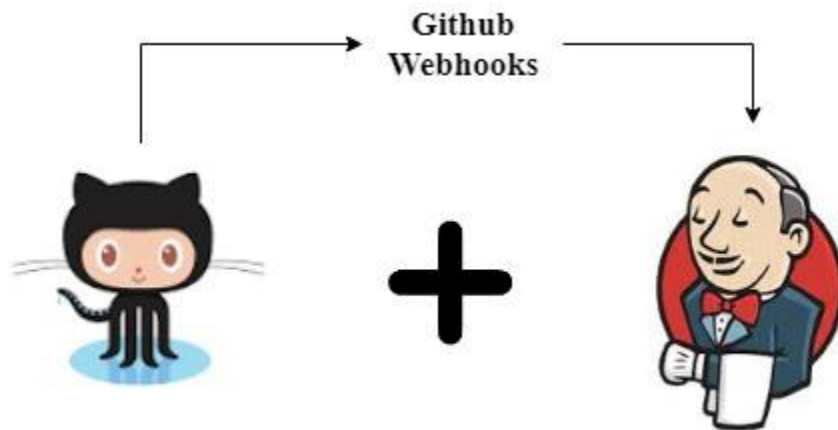


Triggering a Jenkins build on push using GitHub webhooks



Jenkins Github Integration

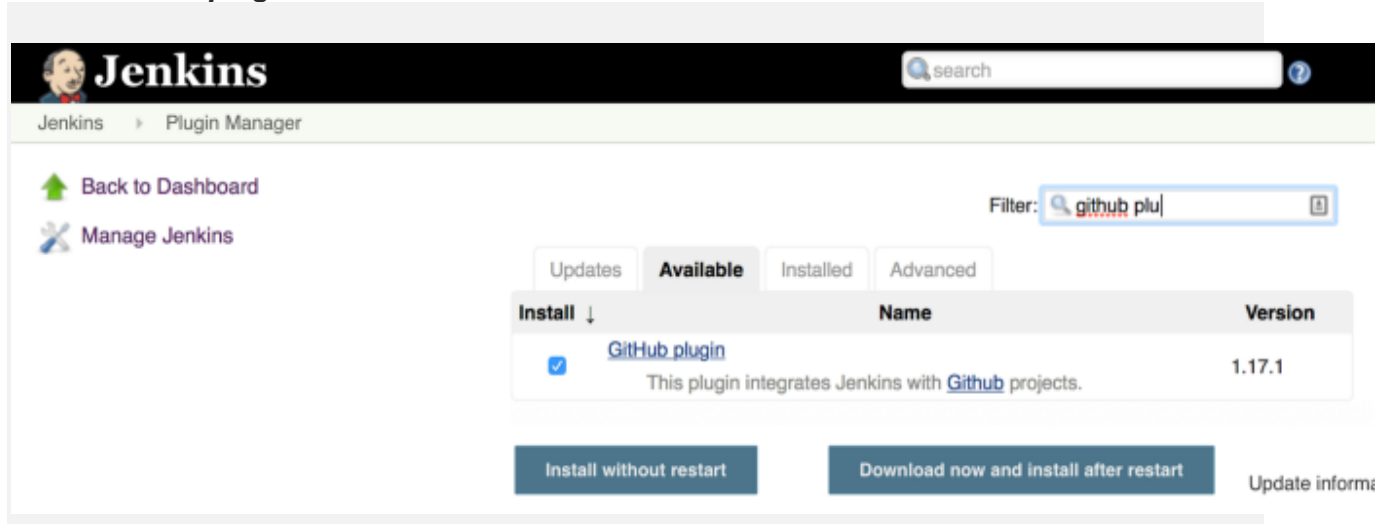
Jenkins is a popular open source tool to perform continuous integration and build automation. Let's take a look at how we can integrate **GitHub** with Jenkins for **source code management** and **trigger build on push using web-hooks**.

Prerequisites :

1. **Jenkins:** Download and install Jenkins as described [here](#).
2. **Git:** Install Git. To check whether you have git installed, open a terminal window and type
below command.
git --version
3. **Plugins:** Add Git and GitHub Plugins.

Go to Manage-Jenkins-> Manage Plugin

search **Github plugin** and install without restart



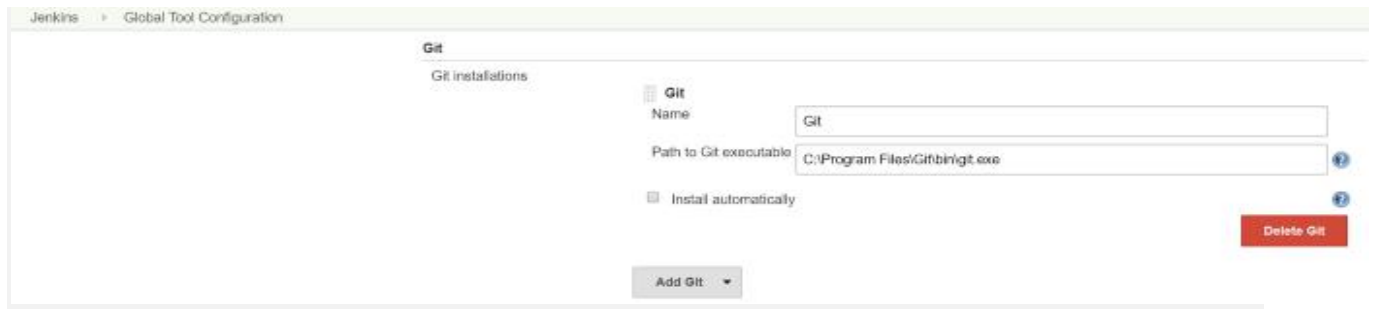
The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a search bar with the text "github plu" entered. Below the search bar, there are tabs for "Updates", "Available", "Installed", and "Advanced". The "Available" tab is selected. A table lists available plugins, with the "GitHub plugin" highlighted. The table has columns for "Name" and "Version". The "GitHub plugin" is listed with version "1.17.1" and a description: "This plugin integrates Jenkins with GitHub projects." Below the table, there are two buttons: "Install without restart" and "Download now and install after restart".

Install ↓	Name	Version
<input checked="" type="checkbox"/>	GitHub plugin This plugin integrates Jenkins with GitHub projects.	1.17.1

Plugin Manager

4. Go to Manage Jenkins -> Global Tool Configuration -> Git

Add git executable path to Global Tool Configuration.



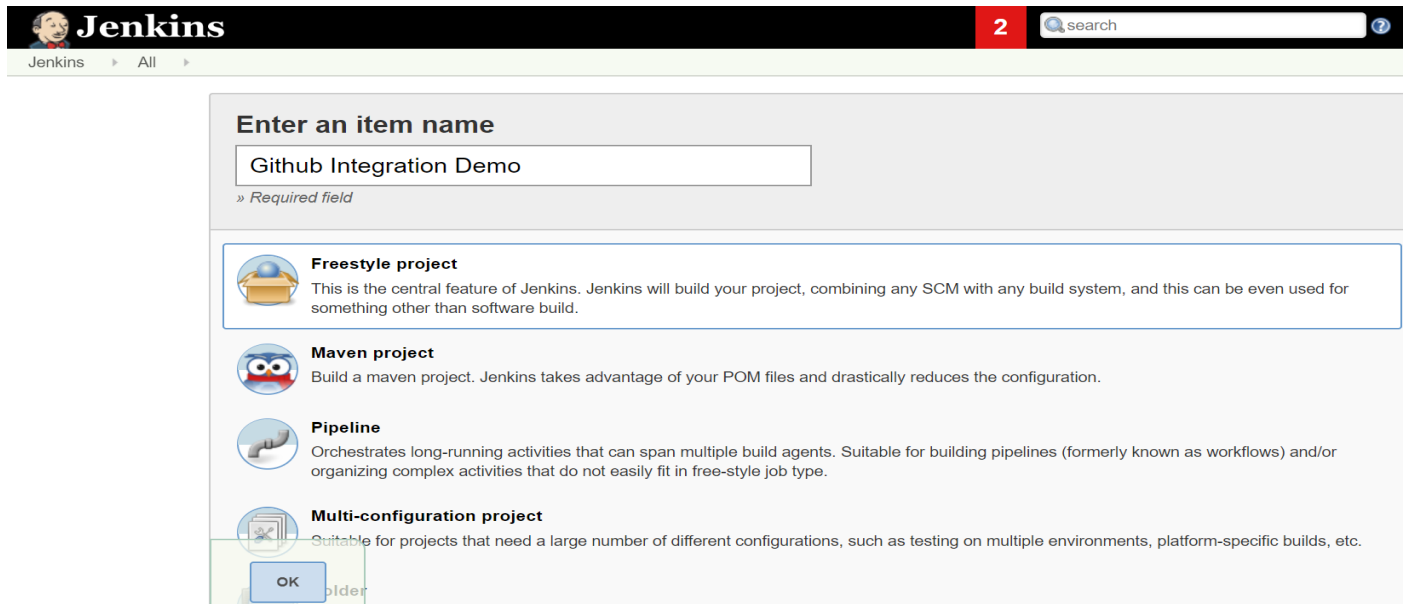
The screenshot shows the Jenkins Global Tool Configuration page for Git. It has a section titled "Git installations". There is a table with one entry for "Git". The "Name" field is "Git" and the "Path to Git executable" field is "C:/Program Files/Git/bin/git.exe". There is a checkbox for "Install automatically" which is unchecked. At the bottom, there is a button "Add Git" and a "Delete Git" button.

Git
<div>Git installations</div> <div><div>Git</div><div>Name</div><div>Git</div><div>Path to Git executable</div><div>C:/Program Files/Git/bin/git.exe</div><div><input type="checkbox"/> Install automatically</div></div>

Global tool configurations

Let us start with creating a Freestyle Project :

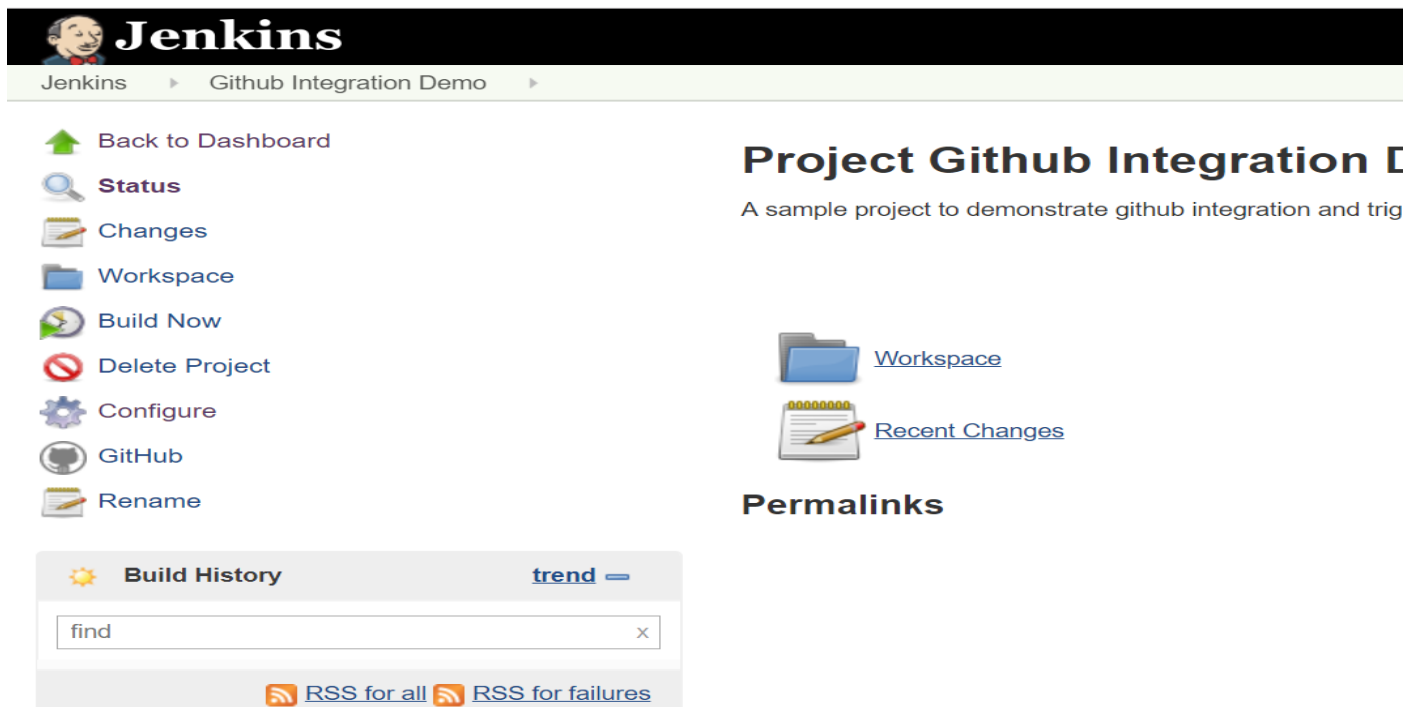
Step 1: Go to New Item -> create a freestyle project.



The screenshot shows the Jenkins 'New Item' page. At the top, the Jenkins logo and name are on the left, a red box with the number '2' is in the center, and a search bar is on the right. Below the header, the breadcrumb 'Jenkins > All' is visible. The main section is titled 'Enter an item name' and contains a text input field with the value 'Github Integration Demo'. Below the input field is a link that says '» Required field'. Underneath this, there are four project type options, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

At the bottom left of the project type list, there is a blue 'OK' button.



The screenshot shows the Jenkins 'Project View' for the 'Github Integration Demo' project. The header bar shows the Jenkins logo and name, and the breadcrumb 'Jenkins > Github Integration Demo'. On the left side, there is a vertical menu with the following items:

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- GitHub
- Rename

On the right side, the project title 'Project Github Integration [...]' is displayed, followed by a subtitle 'A sample project to demonstrate github integration and trig'. Below this, there are two links: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document and pencil icon). Further down, there is a section titled 'Permalinks'. At the bottom, there is a 'Build History' section with a 'trend' link and a search input field containing the text 'find'. Below the search field, there are two RSS feed links: 'RSS for all' and 'RSS for failures'.

Creating a freestyle project

Step 2: Go to **Configure**, add a project description, and Github project URL.

Step 3: In **Source Code Management** tab select on **Git**, add your Github repository URL and click Add button to save your GitHub credentials.

Save and click build to make sure everything is right till here and your project is successfully building.

step 4: In **Build Triggers** select **GitHub hook trigger for GITScm polling**. When Jenkins will receive PUSH GitHub hook, it will trigger Git SCM polling logic which will start a new Jenkins build, with the updated code.



Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitLab. GitLab webhook URL: <http://localhost:8080/project/Github%20Integration%20Demo>
- ☐ GitHub Branches
- ☐ GitHub Pull Requests
- ☒ GitHub hook trigger for GITScm polling

If Jenkins will receive PUSH GitHub hook from repo defined in Git SCM section it will trigger Git SCM polling logic. So polling logic in fact belongs to Git SCM.

(from [GitHub plugin](#))

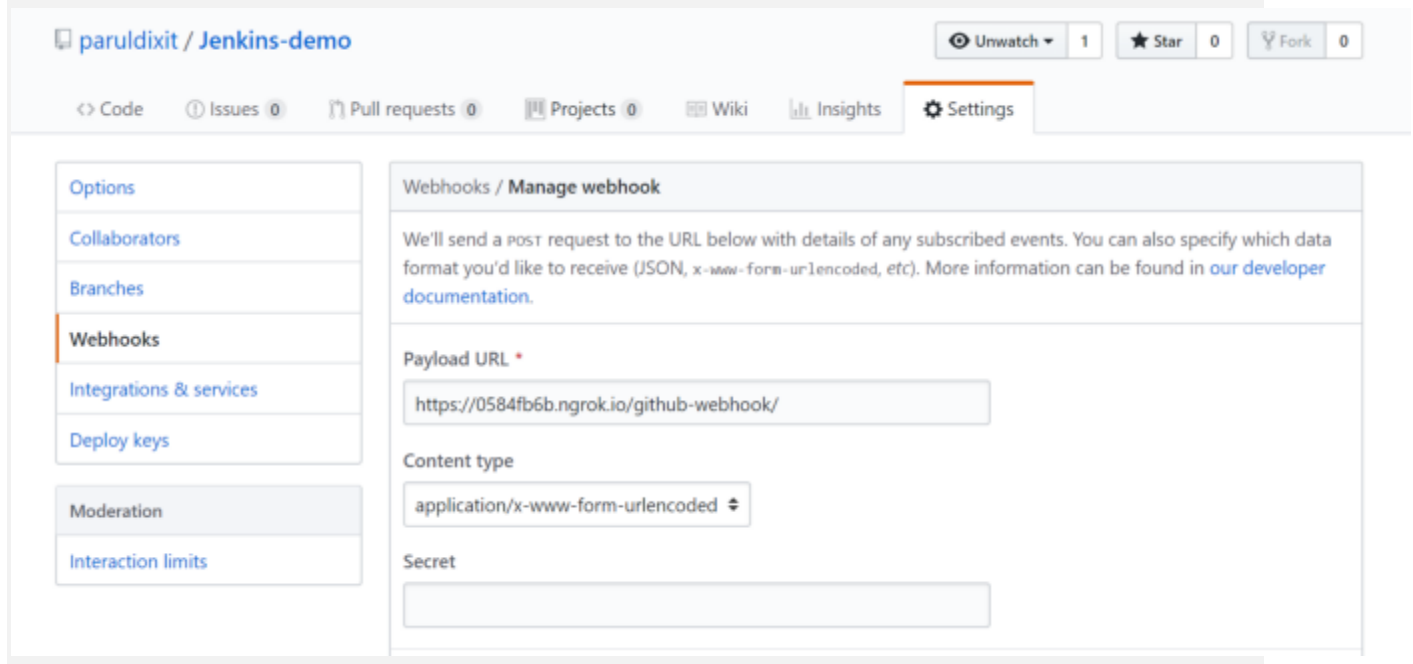
Build Triggers

We use [ngrok](#) to expose local jenkins to the internet, so that github can send the webhooks.

Step 5: Go to your Github repo -> settings -> webhooks

Add public URL of your tunnel as **Payload URL**, it will tell Github where to send the webhooks as below:

`https://0584fb6b.ngrok.io/github-webhook/`



The screenshot shows the GitHub repository settings for 'paruldixit / Jenkins-demo'. The 'Settings' tab is selected, and the 'Webhooks' section is active. The 'Payload URL' is set to 'https://0584fb6b.ngrok.io/github-webhook/'. The 'Content type' is set to 'application/x-www-form-urlencoded'. The 'Secret' field is empty. The left sidebar shows the repository navigation menu with 'Webhooks' highlighted.

paruldixit / Jenkins-demo

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Options

Collaborators

Branches

Webhooks

Integrations & services

Deploy keys

Moderation

Interaction limits

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, ~~x-www-form-urlencoded~~, etc). More information can be found in [our developer documentation](#).

Payload URL *

https://0584fb6b.ngrok.io/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

step 6: Finally add your build steps in the **Build** tab and save.

That's all! Now whenever any change is pushed, a new Jenkins build will be triggered.