

A BEGINNER'S GUIDE TO KICK-STARTING YOUR CI PIPELINE WITH JENKINS

Continuous integration (CI) servers coordinate a wide variety of activities, such as checking out and building new versions of code, running tests, and deploying software. These integrations are handled by automated build and testing systems, which means that your developers are alerted to issues such as code clashes or broken unit tests as early as possible.

The most popular CI tool today is Jenkins. Jenkins is an open-source, server-based system that provides a straightforward platform for automating build testing and integration and for supporting a wide selection of version control systems.

I'd like to share some practical guidance for building and deploying applications using Jenkins, including plugins, which support and extend Jenkins capabilities, making them a vital tool when building your integration and delivery processes.

Creating and scheduling jobs

Jobs are the runnable tasks that are controlled and monitored by Jenkins. Examples of jobs include compiling source code, running tests, provisioning a test environment, deploying, archiving, posting build jobs such as reporting, and executing arbitrary scripts.

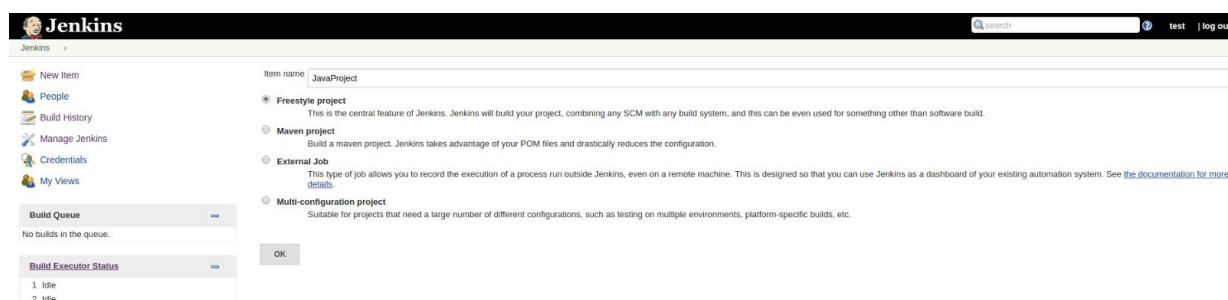
In a few steps, I will create a job that builds a Java project using the Maven build automation tool. Maven allows developers to automate the process of creating an initial folder structure for a Java application and performs the compilation, testing, and deployment of the final product.

For the purposes of this article, our project will include a job that pulls the latest code from the repository and builds it. You could extend this example to run some tests, and then deploy and start the application on a test server.

Executing the build

The first thing you need to do is install Jenkins. After clicking the *Create new jobs* link on the Jenkins dashboard (when there are no other jobs there) or clicking on *New item* in the left menu, you will see the form for a new job.

This form is relatively simple. You will need to enter a name for the job and select the purpose of the job. Jenkins has a preconfigured Maven project that we could use, but for the purposes of this article, we're going to use the *Freestyle project*, to illustrate the full range of options and flexibility available for building complex jobs.



The screenshot shows the Jenkins 'New Item' configuration page. The 'Item name' field is filled with 'JavaProject'. Under the 'Choose from the supported project types' section, 'Freestyle project' is selected with a radio button. Below it, a description states: '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.' Other options like 'Maven project', 'External Job', and 'Multi-configuration project' are listed but not selected. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', 'Credentials', and 'My Views'. At the bottom, there are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two idle executors). An 'OK' button is visible at the bottom right of the form area.

After selecting the job type and name, the next step is to configure the job. The first section I will configure is the source code management section—where my project is located and where the repository is—to pull the code and start the job. In this case, I selected Git as the source code management system, so I need to enter the Git URL in the Repository URL field. If your repository isn't public, you'll need to add your credentials in order to allow Jenkins to access your project/repository's source code. In the *Branches to build* field, enter the branch name that you want to use to build the project:

The screenshot shows the 'Source Code Management' section of the Jenkins job configuration. It includes radio buttons for 'None', 'CVS', 'CVS Projectset', and 'Git' (which is selected). Below this is a 'Repositories' section with a 'Repository URL' field containing 'git://github.com/<user>/<repo>', a 'Credentials' dropdown set to '- none -', and an 'Add' button. To the right are 'Add Repository' and 'Delete Repository' buttons, and an 'Advanced...' link. At the bottom, the 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field containing '*/master', with 'Add Branch' and 'Delete Branch' buttons.

Next is the *Build* section. Click on *Add build step* and select *Invoke top-level Maven targets*. Select the Maven version from the combo box and add Maven goals in the *Goals* field.

The screenshot shows the 'Build' section of the Jenkins job configuration. It features a 'Invoke top-level Maven targets' step. The 'Maven Version' dropdown is set to 'Maven-3.3.9'. The 'Goals' field contains 'verify install'. To the right are 'Advanced...' and 'Delete' buttons.

Schedule

With Jenkins, you can schedule jobs to run every so often to support the build automation process (e.g., nightly build and deploy).

The build can be triggered in several ways: whenever code is checked in, or at predefined times or time intervals, using Linux cron syntax, as shown in the screenshot below. The configuration setting for build triggers is located under the *Build Triggers* section of the job configuration page.

The screenshot shows the 'Build Triggers' section of the Jenkins job configuration. It includes checkboxes for 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', and 'Build periodically' (which is checked). Below the 'Build periodically' checkbox is a 'Schedule' field containing 'H/5 * * * *'. A tooltip below the field states: 'Would last have run at Sunday, April 24, 2016 10:14:02 PM CEST; would next run at Sunday, April 24, 2016 10:19:02 PM CEST.' At the bottom, there is a 'Poll SCM' checkbox.

One common way of triggering a job is to commit a change to a repository. This means that when a developer finishes a development task and pushes their changes onto the project's repository (e.g., the *Git push* command if you're using Git), the job will be automatically triggered. An easy way to do this is via the GitHub Jenkins plugin.

Jenkins plugin installation is done via the *Manage Jenkins* section, as shown below. The *Manage Plugins* option is responsible for adding, removing, enabling, and disabling the plugins. You can locate the plugin you want to install by typing its name in the "Filter" field. After selecting the plugin, you can click *Install without restart* to use the newly installed plugin immediately.

Jenkins

Search

?

Jenkins

ENABLE AUTO-REFRESH

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

⚠️ Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.

Setup Security

Dismiss

🔧 Configure System

Configure global settings and paths.

🔒 Configure Global Security

Secure Jenkins, define who is allowed to access/use the system.

🔄 Reload Configuration from Disk

Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.

🧩 Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**

💻 System Information

Displays various environmental information to assist trouble-shooting.

📄 System Log

System log captures output from java.util.logging output related to Jenkins.

📊 Load Statistics

Check your resource utilization and see if you need more computers for your builds.

🖥️ Jenkins CLI

Access/manage Jenkins from your shell, or from your script.

📝 Script Console

Executes arbitrary script for administration/trouble-shooting/diagnostics.

💻 Manage Nodes

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

👤 Manage Credentials

Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.

❓ About Jenkins

See the version and license information.

🗑️ Manage Old Data

Scrub configuration files to remove remnants from old plugins and earlier versions.

📝 In-process Script Approval

Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.

🛑 Prepare for Shutdown

Stops executing new builds, so that the system can be eventually shut down safely.

Jenkins

Plugin Manager

Back to Dashboard

Manage Jenkins

Filter:

Updates

Available

Installed

Advanced

Install

	Name	Version
<input checked="" type="checkbox"/>	GitHub plugin This plugin integrates Jenkins with GitHub projects.	1.19.1
<input type="checkbox"/>	Violation Comments to GitHub Plugin This plugin uses Violation Comments to GitHub Lib . It will find report files from static code analysis and comment GitHub pull requests with the content.	1.10

Install without restart

Download now and install after restart

Update information obtained: 10 min ago

Check now

Help us localize this page

Page generated: May 29, 2016 11:53:04 AM CEST

[REST API](#)

Jenkins ver. 1.651.2

Once you have the GitHub plugin installed, a new option, *Build when a change is pushed to GitHub*, will appear under the *Build Triggers* section we discussed above.

Build Triggers

☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☐ Build periodically

☒ Build when a change is pushed to GitHub

☐ Poll SCM

?

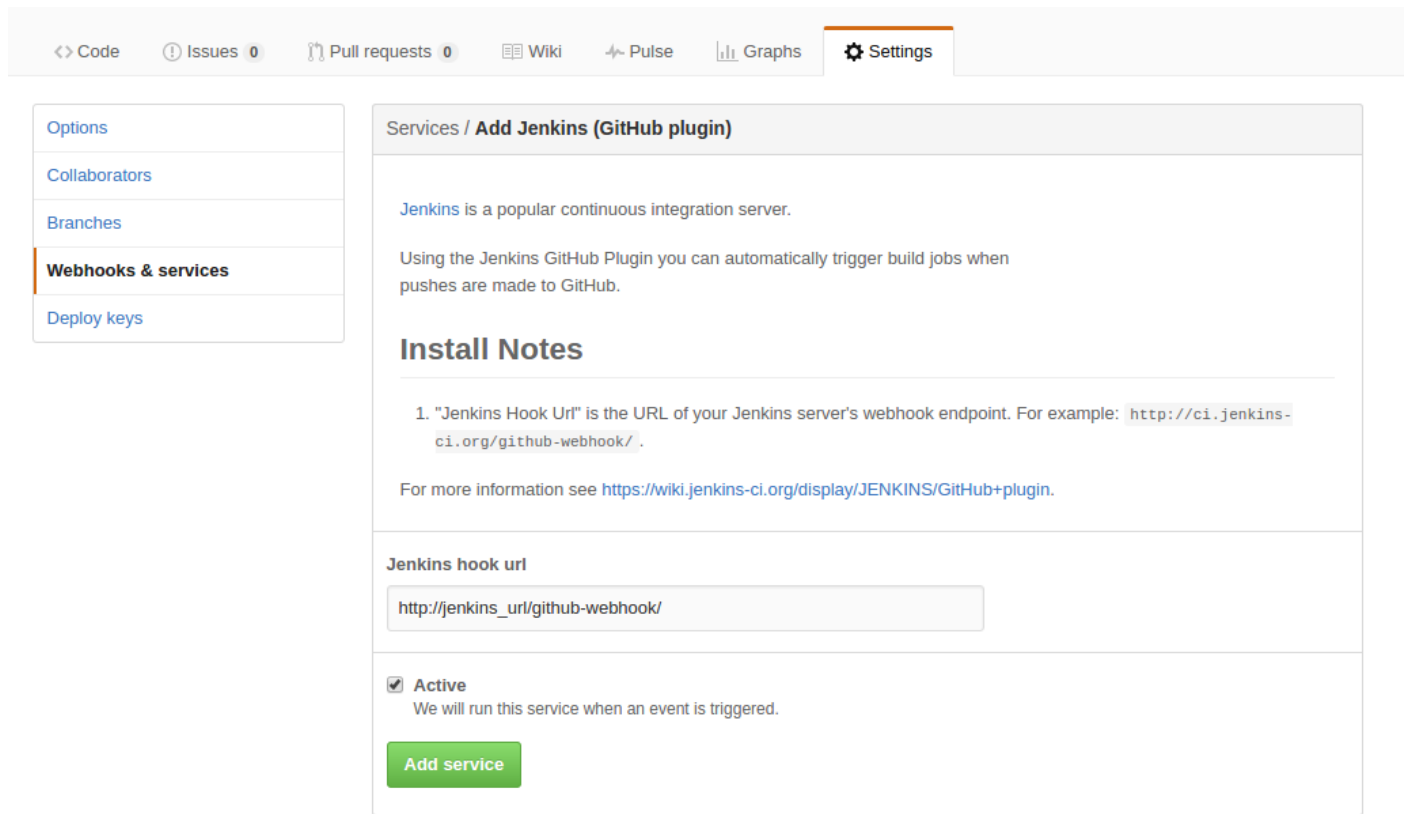
?

?

?

?

In order to complete the *Triggers* configuration, the webhook, which allows you to build or set up integrations that subscribe to certain events on GitHub.com, needs to be added. Go to your GitHub repository and click *Settings* (see the image below). Click *Add Service* and find *Jenkins (GitHub plugin)*. Enter your Jenkins URL in the *Jenkins hook url* field (see below).



The screenshot shows the GitHub repository settings page. The top navigation bar includes links for Code, Issues, Pull requests, Wiki, Pulse, Graphs, and Settings. The left sidebar lists various settings categories: Options, Collaborators, Branches, Webhooks & services (highlighted), and Deploy keys. The main content area is titled 'Services / Add Jenkins (GitHub plugin)'. It contains a description of Jenkins, an 'Install Notes' section with a list of instructions, a 'Jenkins hook url' input field, and an 'Active' checkbox. The 'Add service' button is green and located at the bottom.

Options

Collaborators

Branches

Webhooks & services

Deploy keys

Services / Add Jenkins (GitHub plugin)

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

1. "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://jenkins_url/github-webhook/`

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

Add service

Ensuring a robust Jenkins deployment

The following practices are important if you want to run a robust Jenkins integration server that won't break down.

Secure your Jenkins servers

Jenkins does not perform any security checks as part of its default configuration, so always ensure that you authenticate users and enforce access control on your Jenkins servers. Due to Jenkins' important role, a breach of your Jenkins servers can mean a loss of access credentials to your most valuable resources and other potential risks such as unauthorized users kicking off malicious builds and jobs.

Be cautious with the master(s)

Jenkins supports a master/slave mode where the workload for building projects is delegated to multiple *slave* nodes, allowing one Jenkins installation to host a large number of projects or provide different environments needed for builds/tests.

In a large, complex integration environment that includes multiple users that configure jobs, you should ensure that they are not running builds on the master with unrestricted access into the `JENKINS_HOME` directory. This is where Jenkins stores all of its important data.

Instead, you can use the Job Restrictions Plugin to limit which jobs can be executed on the master. You can also use multiple masters and restrict each to a specific team.

Backup configuration

In order to make sure all configurations and activity logs will be available when needed, you can use the thinBack plugin. This plugin backs up Jenkins' global and job-specific configurations and can be scheduled for automated backup of only the most critical configurations.

You must regularly test that you can restore your system from the backup; otherwise the backup serves no purpose.

Free up enough disk space

Jenkins needs disk space to perform builds, store data logs, and keep archives. To keep Jenkins up and running, make sure that you reserve 10 percent or more of the total disk space for Jenkins in order to prevent fragmentation (which may result in operating system performance degradation, for example). In most cases, mounting network drives and sharing the specific folders is sufficient to fix such issues.

You can use the Disk Usage Plugin to monitor your project usage trends and the remaining allocated disk space. Also, large and complex architectures always include monitoring tools in the integration and delivery systems. Installing lightweight tools such as Nagios or Zabbix can help with logging and monitoring Jenkins, and they will not harm its performance.

Version 2.0 will make CI even easier

In the new 2.0 version, Jenkins offers *pipeline as code*, a new setup experience, and several UI improvements. The Pipeline plugin introduces a domain-specific language (DSL) that helps users model their software delivery pipeline as code. Jenkins 2.0 will also help you choose the plugins that match your needs. Other improvements and redesign enhancements can be found on the official Jenkins 2.0 website.

Using Jenkins to implement your CI is simple, and it is the most common tool for providing CI servers in modern R&D environments. We encourage you to get involved and join the great Jenkins open-source community. You can contribute your own plugins, fix bugs, and share the fixes with the community, or stay tuned to Jenkins mailing lists and the IRC Channel.