# Continuous integration with Jenkins - Tutorial

Lars Vogel (c) 2009, 2017 vogella GmbHVersion 1.7,12.06.2017

Table of Contents

* [1. Using the Jenkins build server](http://www.vogella.com/tutorials/Jenkins/article.html#using-the-jenkins-build-server)
* [2. Installation and setup of Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#installation-and-setup-of-jenkins)
* [3. Configure Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration)
* [4. Jenkins management](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsmanagement)
* [5. Setting up a Jenkins job](http://www.vogella.com/tutorials/Jenkins/article.html#setting-up-a-jenkins-job)
* [6. Jenkins backup and copying files](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins-backup-and-copying-files)
* [7. About this website](http://www.vogella.com/tutorials/Jenkins/article.html#about-this-website)
* [8. Jenkins links](http://www.vogella.com/tutorials/Jenkins/article.html#hudsonlinks)
* [Appendix A: Copyright and License](http://www.vogella.com/tutorials/Jenkins/article.html#copyright-and-license)

*This article describes how to use the Jenkins continuous integration build server.*

## [1. Using the Jenkins build server](http://www.vogella.com/tutorials/Jenkins/article.html#using-the-jenkins-build-server)

Continuous integration is a process in which all development work is integrated as early as possible. The resulting artifacts are automatically created and tested. This process allows to identify errors as early as possible.

[Jenkins](https://jenkins.io/) is a popular open source tool to perform continuous integration and build automation. The basic functionality of Jenkins is to execute a predefined list of steps, e.g. to compile Java source code and build a JAR from the resulting classes. The trigger for this execution can be time or event based. For example, every 20 minutes or after a new commit in a Git repository.

Possible steps executed by Jenkins are for example:

* perform a software build using a build system like Apache Maven or Gradle
* execute a shell script
* archive a build result
* running software tests

Jenkins monitors the execution of the steps and allows to stop the process, if one of the steps fails. Jenkins can also send out notification in case of a build success or failure.

Jenkins can be extended by additional plug-ins. For example, you can install plug-ins to support building and testing Android applications.

## [2. Installation and setup of Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#installation-and-setup-of-jenkins)

For most platforms you have native packages, see the [Jenkins Homepage](https://jenkins.io/).

### [2.1. Installing of the Jenkins server on Ubuntu](http://www.vogella.com/tutorials/Jenkins/article.html#installing-of-the-jenkins-server-on-ubuntu)

Jenkins provides Debian/Ubuntu packages which install Jenkins and register Jenkins as start service. See the [Install Jenkins on Ubuntu description](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu) The Linux installation creates a /etc/init.d/jenkins script which starts Jenkins automatically at boot time.

Jenkins stores all the settings, logs and build artifacts in its home directory. The default installation directory is /var/lib/jenkins under Ubuntu.

#### [2.1.1. Using the .war file to start Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#using-the-war-file-to-start-jenkins)

Download the jenkins.war file from [Jenkins Homepage](https://jenkins.io/). From this file you can start Jenkins directly via the command line with java -jar jenkins\*.war.

If you start it locally, you find it running under the following URL: <http://localhost:8080/>

To run it in your Tomcat server, put the .war file into the webapps directory. If you start Tomcat, your Jenkins installation will be available under

http://localhost:8080/jenkins

|  |  |
| --- | --- |
|  | If the jenkins.war is deployed in your webapps directory, but cannot be started and the tomcat manager says ﻿FAIL - Application at context path /jenkins could not be started, you may need to grant the permissons for ﻿JENKINS\_HOME.  ﻿sudo mkdir .jenkins  ﻿sudo  chown tomcat7:nogroup .jenkins  This makes the .jenkins folder writable and Jenkins can use it. |

### [2.2. Connect to Jenkins for the initial configuration](http://www.vogella.com/tutorials/Jenkins/article.html#connect-to-jenkins-for-the-initial-configuration)

After installation, open a browser and connect to it. The default port of Jenkins is :8080, therefore on your local machine you find it under the following URL:

http://localhost:8080/

You will need to copy the initial password from the file system of the server.

Afterwards you can select to install plug-ins. Select the Install suggested plug-ins to get a typical configuration.

Create an admin user and press Save and Finish.

## [3. Configure Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration)

### [3.1. Secure Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#secure-jenkins)

It is recommended to secure Jenkins. Select Manage Jenkins and then Configure Global Security. Select the Enable security flag. The easiest way is to use Jenkins own user database. Create at least the user "Anonymous" with read access. Also create entries for the users you want to add in the next step.

On the login page, select Create an account to create the users you just gave access.

### [3.2. Create a new user](http://www.vogella.com/tutorials/Jenkins/article.html#create-a-new-user)

Go to Manage Jenkins, Manage and Assign Roles and then Assign Roles to grant the newly created user additional access rights.

Navigate to Manage Roles to define access restrictions in detail. Pattern is a regex value of the job name. The following grants unregistered users read-only access to your build jobs that start with the C-MASTER or M-MASTER prefix and only those.

### [3.3. Generate ssh key for Jenkins user](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsconfiguration_ssh)

If you want to access a private Git repo, for example at Github, you need to generate an ssh key-pair. Create a SSH key with the following command.

sudo -u jenkins ssh-keygen

The public key must be uploaded to the service you are using, e.g., Github.

## [4. Jenkins management](http://www.vogella.com/tutorials/Jenkins/article.html#jenkinsmanagement)

### [4.1. Plug-in management](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_pluginmanagement)

Jenkins can be extended via additional plug-ins with more functionality. You can configure your plug-ins via the Manage Jenkins ▸ Manager Plugins link.

To install plugins in Jenkins select use the Manage Jenkins ▸ Manager Plugins link and search for the plugin you want to install. Select it from the list and select to install it and restart Jenkins.

The following table is a summary of commonly used plug-ins.

| *Table 1. Jenkins plug-ins* | | |
| --- | --- | --- |
| **Plug-in name** | **Description** | **URL** |
| Git Plugin | This plugin allows use of Git as a build SCM. | <https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin> |
| Xvnc plugin | This plugin allows projects to run xvnc during a build. This allows for example to run tests which requires a display to run on a virtual display. To use this plug-in you need to connect once to your vncserver on the command line to provide a password. Use for example the following commands.  # install vncserver  apt-get install vnc4server  # **switch** to jenkins user  sudo su jenkins  # connect to vncserver which creates the password  vncserver :10 | <https://wiki.jenkins-ci.org/display/JENKINS/Xvnc+Plugin> |
| Gradle Plugin | This plugin allows to run Gradle builds, e.g., as required for Android, via Jenkins. | <https://wiki.jenkins-ci.org/display/JENKINS/Gradle+Plugin> |
| Maven Plugin | This plugin allows to run Maven builds. | <https://wiki.jenkins-ci.org/display/JENKINS/Maven+Project+Plugin> |
| GitHub plugin | This plugin integrates Jenkins with Github projects. | <https://wiki.jenkins-ci.org/display/JENKINS/Github+Plugin> |
| Publish Over SSH Plugin | This plugin allows to publish build artifacts via ssh | <https://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin> |
| Workspace Cleanup Plugin | This plugin allows to delete the workspace before the build or when a build is finished and artifacts saved. | <https://wiki.jenkins-ci.org/display/JENKINS/Workspace+Cleanup+Plugin> |
| Github Pull Request Builder | This plugin allows to build Github Pull Requests | <https://wiki.jenkins-ci.org/display/JENKINS/GitHub+pull+request+builder+plugin> |

### [4.2. Restart your Jenkins](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins_pluginmanagement_restart)

You can manually restart Jenkins by adding restart as URL parameter.

## [5. Setting up a Jenkins job](http://www.vogella.com/tutorials/Jenkins/article.html#setting-up-a-jenkins-job)

The build of a project is handled via jobs in Jenkins. Select New Item. Afterwards, enter a name for the job and select Freestyle Job and press OK.

Enter a description for the job and configure how many old jobs should be retained.

Configure how the source code can be retrieved. If you for example using Git, enter the URL to the Git repository. If the repository is not public, you may also need to configure the credentials.

Specify when and how your build should be triggered. The following example polls the Git repository every 15 min. It triggers a build, if something has changed in the repo.

I typically delete the workspace before a build to avoid any side-effect. In the Build section you can add a build step, e.g., a Maven build.

Press Save to finish the job definition. Press Build Now on the job page to validate the job works as expected.

After a while the job should go to green or blue (depending on your configuration), if successful. Click on the job and afterwards on Console Output to see the log file. Here you can analyze the build errors.

## [6. Jenkins backup and copying files](http://www.vogella.com/tutorials/Jenkins/article.html#jenkins-backup-and-copying-files)

Jenkins stores all the settings, logs and build artifacts in its home directory. For example, in /var/lib/jenkins under the default install location of Ubuntu.

To create a backup of your Jenkins setup, just copy this directory.

The jobs directory contains the individual jobs configured in the Jenkins install. You can move a job from one Jenkins installation to another by copying the corresponding job directory. You can also copy a job directory to clone a job or rename the directory.

You only need to copy the config.xml file. If you are using Git in the jobs directory you can use the following .gitignore file to exclude everything except this file.

# Ignore everything in /jobs

\*

# Reinclude all folders in /jobs

!\*/

# Ignore everything in the subfolders of /jobs

\*/\*

# Reinclude config.xml files in the first-level subfolders of /jobs

!\*/config.xml

Click Reload Configuration from Disk button in the Jenkins web user interface to force Jenkins to reload configuration from the disk.

See the following link for details: [https://wiki.jenkins-](https://wiki.jenkins-ci.org/display/JENKINS/Administering+Jenkins)