# User Guide

**JWebSocket**

**Continuous Integration Environment for jWebSocket**

**V 1.0**

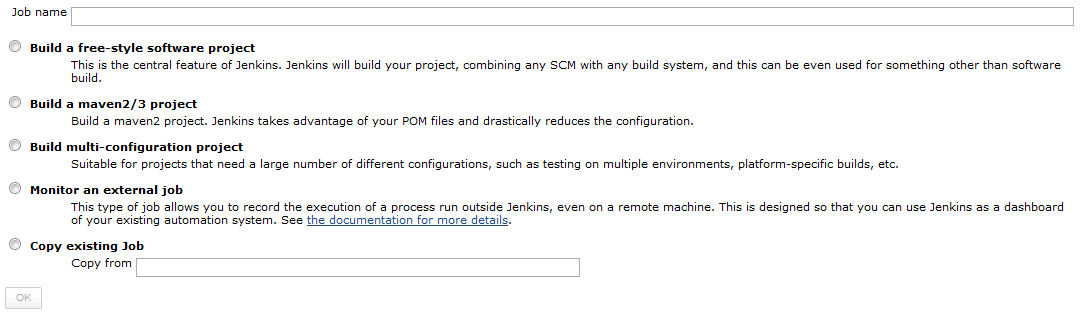
**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Autor** |
| 12/06/2012 | 1.0 | Creation of document | Carlos Alberto Feyt Salgueiro |

1. Creating a new Jenkins Build Job

Build jobs are at the heart of the Jenkins build process. Simply put, you can think of a Jenkins build job as a particular task or step in your build process. This may involve simply compiling your source code and running your unit tests. Or you might want a build job to do other related tasks, such as running your integration tests, measuring code coverage or code quality metrics, generating technical documentation, or even deploying your application to a web server. A real project usually requires many separate but related build jobs.

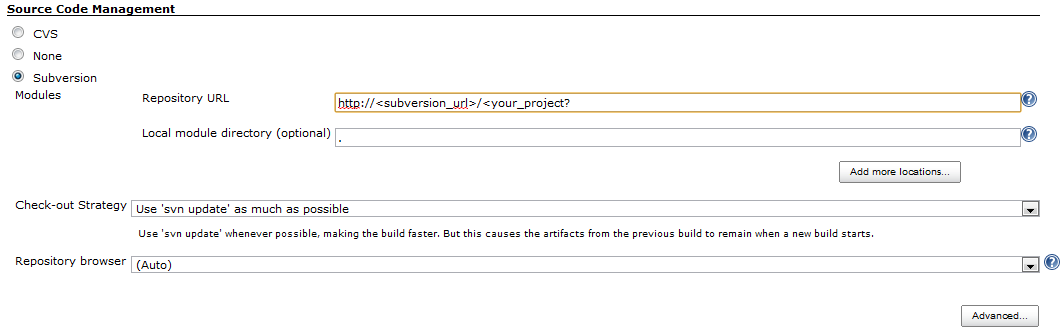
To create new build job click on the New Job link, you should be see a screen similar to this file:



Picture 1.1 Setting up your first build job in Jenkins.

Jenkins supports several different types of build jobs. The two most commonly-used are the freestyle builds and the Maven 2/3 builds. The Maven 2/3 builds understand the Maven project structure, and can use this to let you set up Maven build jobs with less effort. So choose “Build a maven2/3 project”, as shown in Figure 1.1, type the Job name and click the “Ok” button.

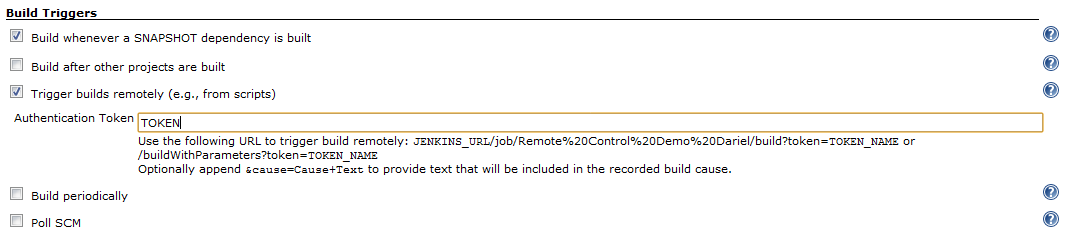
Then the first step is type the Project name. After you need to set “Source Code Management”, select the Subversion option and type your project URL in the textbox Repository URL. Following you can see an example for this configuration.



Picture 1.2 Source code management configuration.

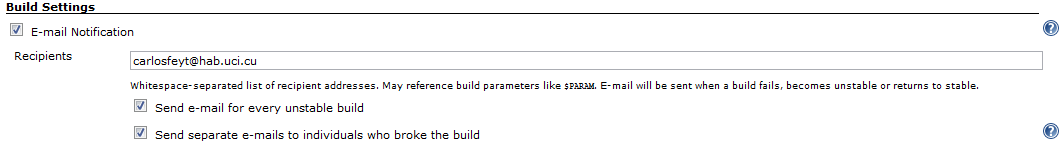
Other important step is active and configure the triggers in Jenkins Server. This is to allow the remotes executions for start the Jenkins project compilations. These executions will be launch for the Subversion server when some user makes a commit.

To do so go to the Build Triggers session and check the Trigger Builds remotely. Here you can see an example to this configuration.



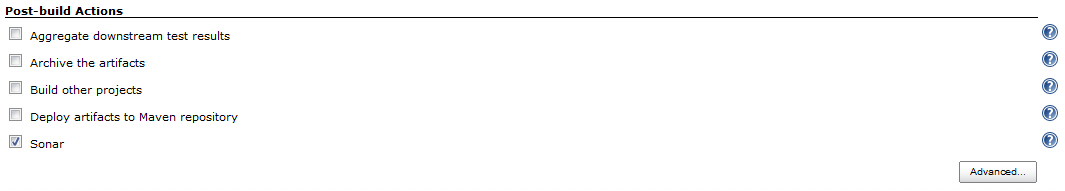
Picture 1.3 Example of trigger configuration in Jenkins projects.

The next step is configure the Email Notifications. This is to notificate when some build fails, Jenkins will send an email message to the developer who committed the changes, and optionally to other team members as well. To do so, you only need to check the “E-mail Notification” in the “Build Settings” session. There you can type an Email list to notificate some change at our project. Following you can see an example to Notification Settings.



Picture 1.4 Email notification settings

Finally you can to configure what do you want to happen when the build finished. In this case we only check the “Sonar” option to create a metric for this build. Following you can see an example for this setting.



Picture 1.5 Post-build setting

Now you can save this project.