Installation and Setup Guide

**Sonar**

Table of Contents

[1 Introduction 3](#_Toc428876383)

[2 Apache Tomcat Installation 3](#_Toc428876384)

[2.1 Installing Apache Tomcat as a Windows Service 4](#_Toc428876385)

[2.2 Configure Apache Tomcat Ports and Users 8](#_Toc428876386)

[2.3 Verifying Installation 10](#_Toc428876387)

[3 DB Installation 10](#_Toc428876388)

[4 Sonar Installation 10](#_Toc428876389)

[5 Sonar Initial Setup 12](#_Toc428876390)

[5.1 Configuring the Default Quality Profile 13](#_Toc428876391)

[6 Sonar Update Centre 14](#_Toc428876392)

[7 Installing and Configuring the Jenkins Sonar Plugin 15](#_Toc428876393)

[8 Installing and configuring the Eclipse Sonar plugin 17](#_Toc428876394)

[8.1 Installing the Eclipse Sonar plugin 17](#_Toc428876395)

[8.1.1 Getting Sonar using Eclipse Marketplace 17](#_Toc428876396)

[8.1.2 Installing Sonar using the Eclipse update center 18](#_Toc428876397)

[8.1.3 Installation 18](#_Toc428876398)

[8.2 Configuring the Eclipse Sonar plugin 19](#_Toc428876399)

[9 References 23](#_Toc428876400)

[10 Document Control 23](#_Toc428876401)

[10.1 Change History 23](#_Toc428876402)

[10.2 Open Issues 23](#_Toc428876403)

# Introduction

Sonar is a Continuous Quality Assurance engine that consolidates static and dynamic quality metrics leveraging known tools as Checkstyle, PMD, FindBugs, JUnit, Cobertura, JDepend or Architecture Rules. Sonar integrates with Maven builds or with Jenkins jobs, automatically running all the tools, gathering all the relevant data and collating them all in a single dashboard.

Sonar is the recommended tool in Java Blueprint for adopting Automated Solution Quality practices in the server-side using Sonar for Build Quality Dashboard.

Also since Java Blueprint 3.2 also Sonar is the recommended tool for Automated Code / Configuration Review in developer workstations using the Eclipse plugin for Sonar.

This document contains detailed insights on how to install Sonar and setup it for first use. The companion document, ***Sonar Usage Guide***, includes information for end users on how to analyse sources and work with the Sonar dashboard.

# Apache Tomcat Installation

The preferred way of running Sonar, for best performance and user experience, is embedded inside a Java web application container. Apache Tomcat is a well-known, reference web application container from the Apache Foundation. It is Open Source and probably the most used web container.

NOTE: The installation process below assumes that you are installing only Sonar in this instance. If you plan for installing both Jenkins and Sonar, it is recommended to place them in different Tomcat instances. It is possible, however, to have them running in the same instance. In this case, please be aware that you should skip part of the process when the second tool is being installed.

Before installing Tomcat, we need to ensure that the Oracle Java Development Kit (JDK) is installed and that the environment variable JAVA\_HOME points to the folder where JDK is installed. The Java Blueprint 3.2 recommends using JDK 7 update 15 version. In the rest of the guide we will assume that JDK is installed in **c:\java\jdk-7** folder. Update all references to this folder in case you choose to install JDK in a different place. JDK can be downloaded from: [http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jdk-7u15-oth-JPR](http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html" \l "jdk-7u15-oth-JPR)

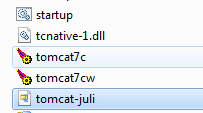
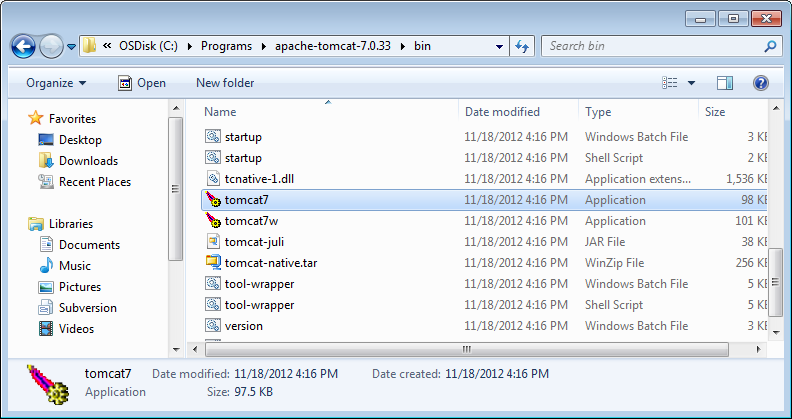
If you are installing Tomcat and Sonar in a 64-bit box, you may choose to install the 64-bit version of JDK 7. This will enable the Tomcat process to be able to manage more than 2 GB of RAM, useful when you have Jenkins and Sonar running in the same Tomcat instance or a high number of users or configured projects.

Once JDK 7 is installed, we decompress the binary distribution of Tomcat on any desired folder. In the rest of the guide we will assume that Tomcat is installed in **c:\java\apache-tomcat-7.0.22** folder. Update all references to this folder in case you choose to install Tomcat in a different place. Tomcat can be downloaded from: <http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.22/bin/apache-tomcat-7.0.22-windows-x86.zip>

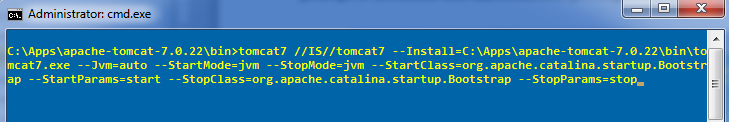
## Installing Apache Tomcat as a Windows Service

It is recommended to install Tomcat as a Windows Service. This way Tomcat (and Sonar with it) will be available automatically each time Windows is started, without the need for human intervention.

First thing to consider when installing Tomcat as a service is whether multiple Tomcat instances will be running in the same box. If this is the case, you may need to rename two files inside Tomcat **bin** folder: **tomcat7.exe** and **tomcat7w.exe**. The name of these files, **tomcat7**, is used to identify the service in Windows registry. Thus, to prevent name conflicts, you may rename both files in the different Tomcat instances, e.g. **tomcat7c.exe** and **tomcat7cw.exe**.



Once the names are correct, we proceed with the installation of the service. To do that, open a command prompt in Windows with administrator privileges and move to the Tomcat **bin** folder.



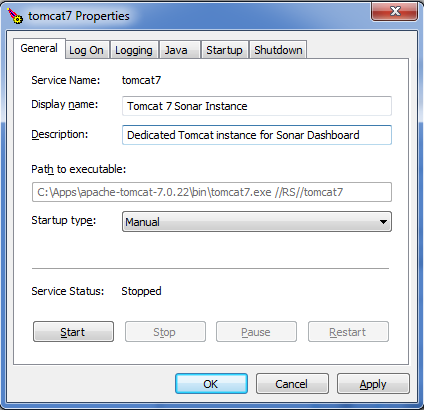
Under **bin** folder, issue the following command:

tomcat7 //IS//**tomcat7** --Install=C:\Apps\apache-tomcat-7.0.22\bin\tomcat7.exe --Jvm=auto --StartMode=jvm --StopMode=jvm --StartClass=org.apache.catalina.startup.Bootstrap --StartParams=start --StopClass=org.apache.catalina.startup.Bootstrap --StopParams=stop

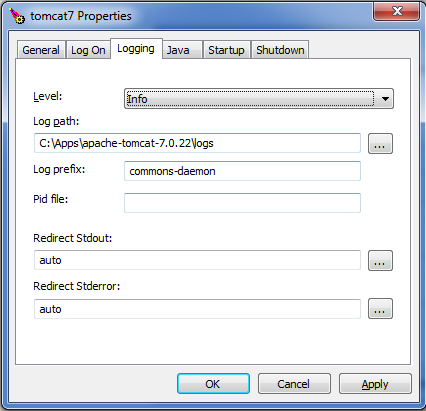
You may need to adapt the names and paths accordingly to the service name and Tomcat installation folder.

Once the service is installed, we need to configure it running the executable ending in ‘w’, e.g. **tomcat7w.exe**. A dialog with multiple tabs will appear. To complete Tomcat installation, follow these steps:

* Modify the service name and description (if needed and desired).
* Modify start-up type to Automatic (recommended).



* Modify logon account if needed, e.g. to access network resources.
* Modify logging settings:
  + Level: Error – Can be switched to info/debug if needed to troubleshoot problems
  + Path: C:\Apps\apache-tomcat-7.0.22logs
  + Redirect: auto – In both streams



* Modify Java options:
  + JVM: C:\java\jre\jre7u15\bin\server\jvm.dll
  + Java claspath: C:\Apps\apache-tomcat-7.0.22\bin\bootstrap.jar;C:\Programs\apache-tomcat-7.0.33\bin\tomcat-juli.jar
  + Java options:

-Dcatalina.base= C:\Apps\apache-tomcat-7.0.22

-Dcatalina.home= C:\Apps\apache-tomcat-7.0.22

-Djava.endorsed.dirs= C:\Apps\apache-tomcat-7.0.22\endorsed

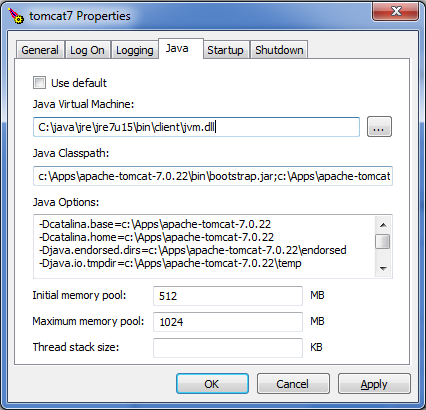
-Djava.io.tmpdir= C:\Apps\apache-tomcat-7.0.22\temp

-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager

-Djava.util.logging.config.file= C:\Apps\apache-tomcat-7.0.22\conf\logging.properties

-XX:MaxPermSize=512m

* + Initial memory: 512
  + Maximum memory: 2048



## Configure Apache Tomcat Ports and Users

Next step is to configure ports and users for Tomcat.

Usually default ports work fine just after installation, but in the case that other software is using the same ports (or you want to setup different Tomcat instances) you may need to update ports to be used in configuration.

To update the ports, the file **conf\server.xml** needs to be edited. You will need to search for 4 different ports. By default they are:

* HTTP transport: 8080
* HTTPS transport: 8443
* Shutdown port: 8005
* AJP connector: 8009

To enable new ports, just search in the configuration file all occurrences of the ports above and replace them for the chosen ones.

<Server port="**8005**" shutdown="SHUTDOWN">

…

<Connector port="**8080**" protocol="HTTP/1.1" URIEncoding="UTF-8"

connectionTimeout="20000" redirectPort="**8443**" />

…

<Connector port="**8009**" protocol="AJP/1.3" redirectPort="**8443**" />

If you like to run for example the Sonar instance on port 80 you should:

* HTTP/HTTPS ports - use 80 for HTTP and 443 for HTTPS:

<Connector port="**80**" protocol="HTTP/1.1" URIEncoding="UTF-8"

connectionTimeout="20000"

redirectPort="**443**" />

* Change instances of 8443 port (in redirect attributed) to 443 port.

To update the users, the file **conf\tomcat-users.xml** needs to be edited. This step is needed to enable an administrator user to access Tomcat console and manage the server. Once opened, edit the file ensuring that a proper administrator user is configured with manager, admin and manager-gui roles:

<tomcat-users>

<role rolename="tomcat"/>

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

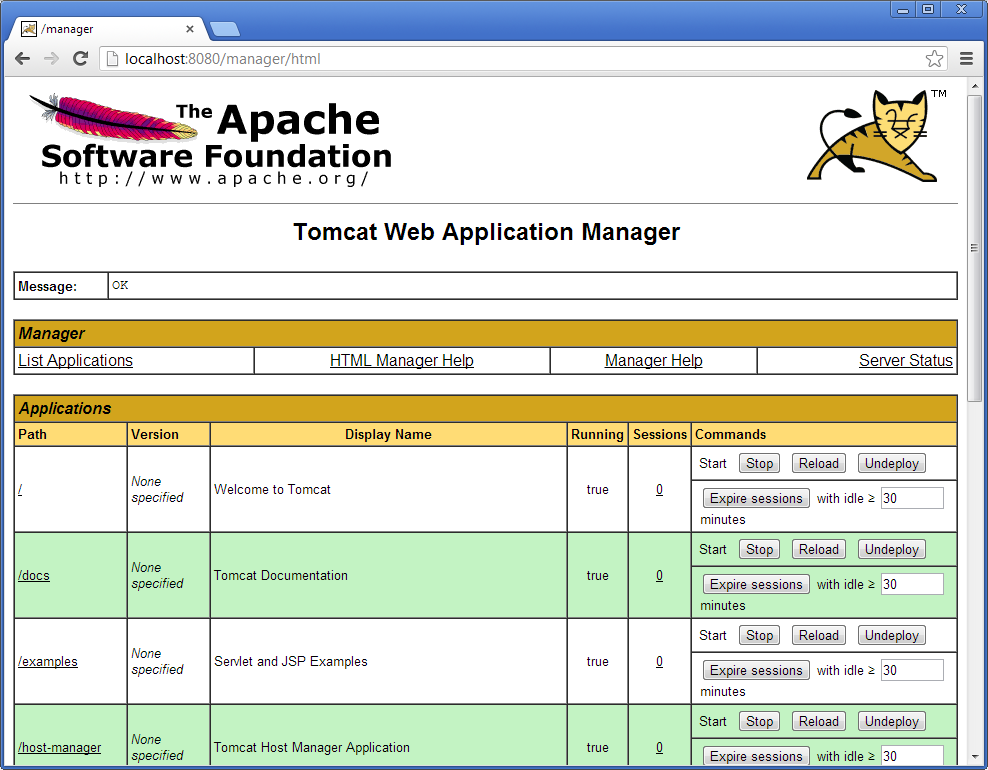
<role rolename="admin"/>

<user username="admin" password="adminadmin" roles="tomcat, manager-gui, manager-script, manager-jmx, manager-status,admin"/>

</tomcat-users>

## Verifying Installation

Once the previous steps are finished, you should verify that Tomcat is properly installed by starting the service and opening in your browser the administration console URL: <http://localhost:<port>/manager/html>



# DB Installation

Sonar uses a relational database to store the configuration and metrics for configured projects. Although Sonar ships with an embedded Apache Derby database, for the best performance and user experience it is recommended to use a more robust database engine.

# Sonar Installation

This section of the guide will explain how to install Sonar inside the Apache Tomcat instance and using the DB instance installed on previous sections.

The Java Blueprint 3.2 recommends the installation of Sonar 3.4.1 that can be downloaded from this URL: <http://dist.sonar.codehaus.org/sonar-3.4.1.zip>

Please get the latest version.

Before installing Sonar, please ensure that the Apache Tomcat service is not running.

Next, decompress the **sonar-3.4.1.zip**. We will assume that target folder is **C:\Apps\sonar-3.4.1**. Edit the file **C:\Apps\sonar-3.4.1\conf\sonar.properties** to refer to the Derby database created in previous section. The properties that should be updated are the following:

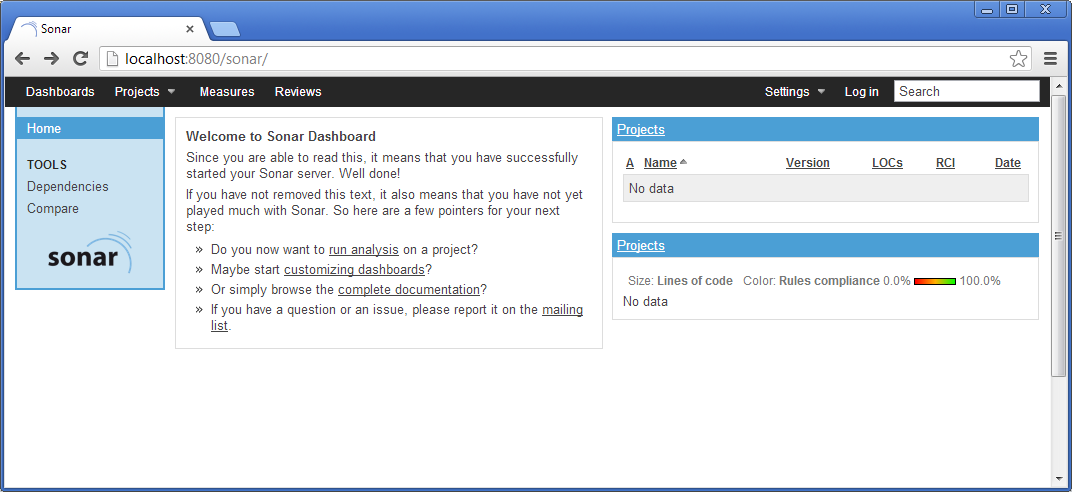
* **sonar.jdbc.url** = jdbc:derby://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true
* **sonar.jdbc.driverClassName** = <Apache Derby Driver name>
* **sonar.jdbc.validationQuery** = select 1
* **sonar.jdbc.username** = sonar
* **sonar.jdbc.password** = sonarsonar

Note that there are some provided templates for quick configuration of supported databases.

If Sonar dashboard is running behind a proxy, you should also configure its settings in the **sonar.properties** file.

Next step is to create the web application file by executing the command: **C:\Apps\sonar-3.4.1\war\build-war.bat**. After a few seconds, the web application file will be created. Copy the file **C:\Apps\sonar-3.4.1\war\sonar.war** to **C:\Apps\apache-tomcat-7.0.22\webapps** folder.

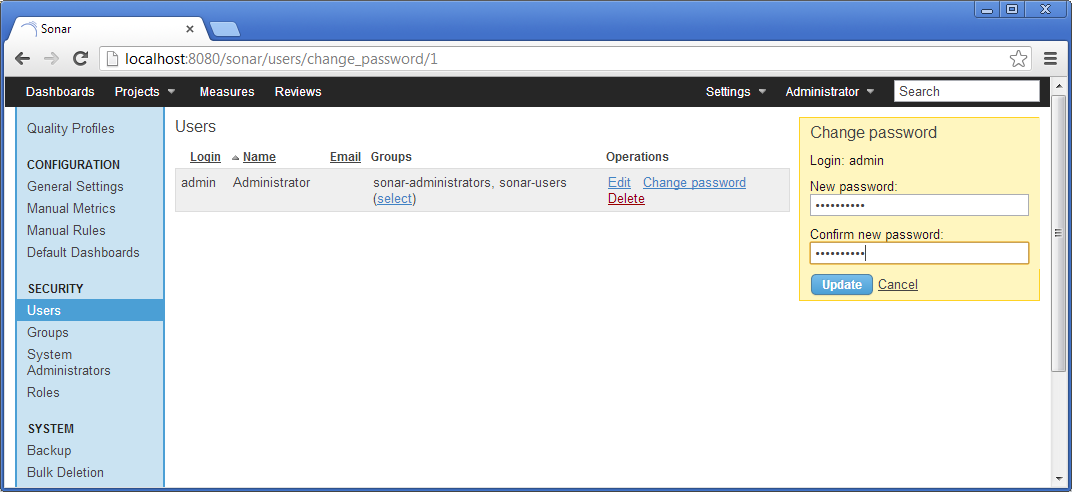
Now start the Tomcat service. After a few seconds, Tomcat should be up and running including the Sonar dashboard with it. Verify that Sonar is properly installed by opening in your browser the Sonar dashboard URL: <http://localhost:<port>/sonar>



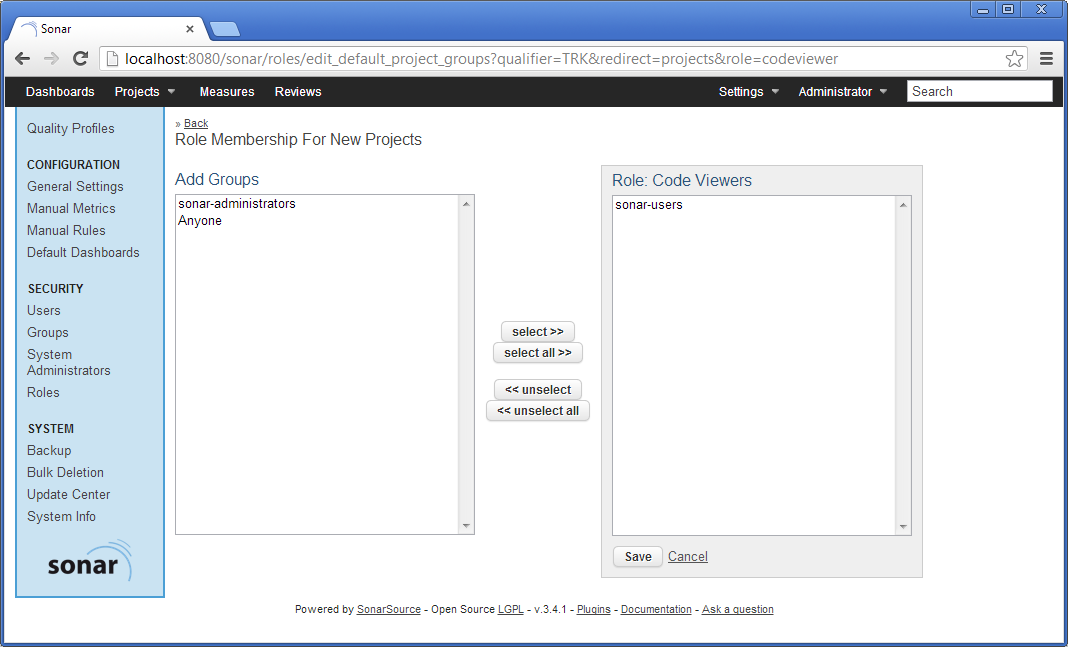
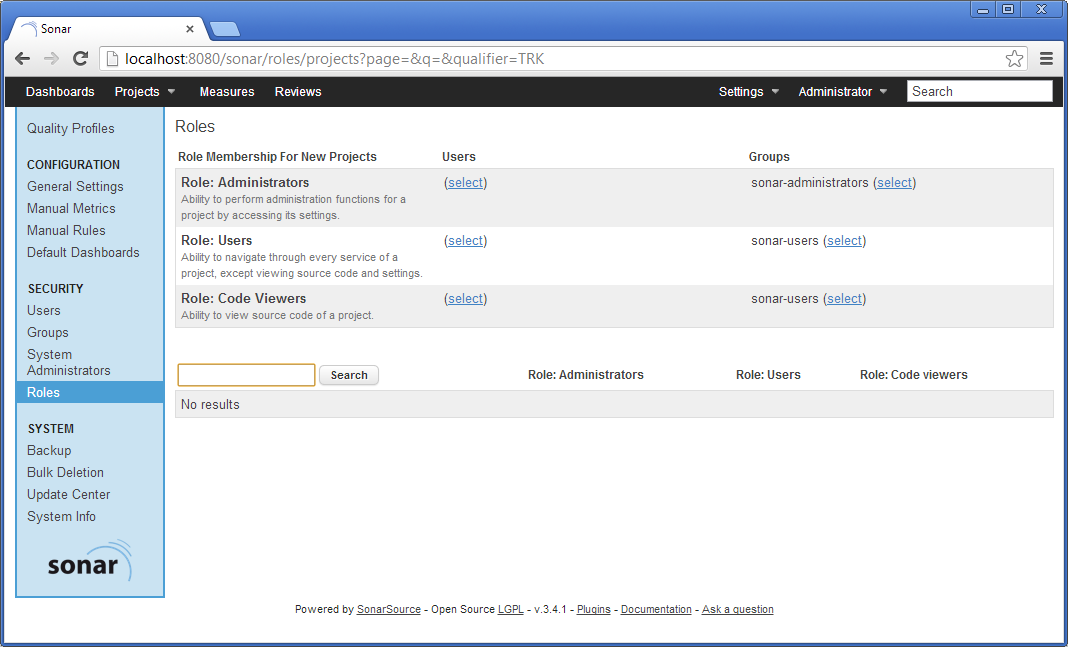
# Sonar Initial Setup

Now that Sonar is installed and we have access to Sonar dashboard, we will prepare it for first use.

In the top menu, select **Log In**. Enter default administrator credentials, user **admin** and password **admin**. In the top menu again, select **Settings -> Configuration**. In the left menu, select **Users** and set up a new administrator password, for example **adminadmin**:



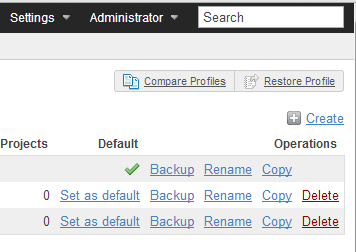
It is recommended to remove access to anonymous users. This will prevent unauthorized access to the Sonar dashboard information, which includes not only metrics and historic data, but sources and Javadoc documentation. Note that sources and documentation can be hidden in the dashboard if security policies or client agreement requires that. To remove anonymous access, in the left menu select **Project roles**. In the list of roles and groups, remove the group **Anyone** from **Users** and **Code viewers** roles:



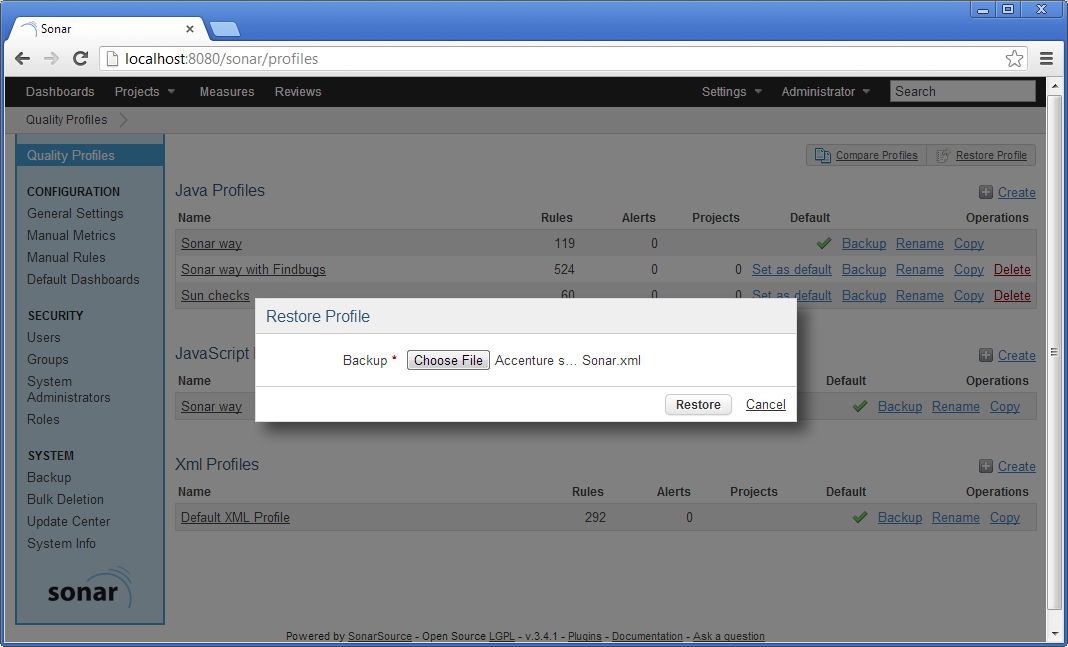
## Configuring the Default Quality Profile

A quality profile in Sonar is a set of quality rules (sometimes referred also as checkers, detectors or profilers) and their associated severity and configuration, plus definition of thresholds and alerts for multiple quality indicators. Quality rules come from the tools that Sonar uses to collect quality data: Checkstyle, PMD, FindBugs and Sonar Squid (based on JavaNCSS, JDepend and Architecture Rules).

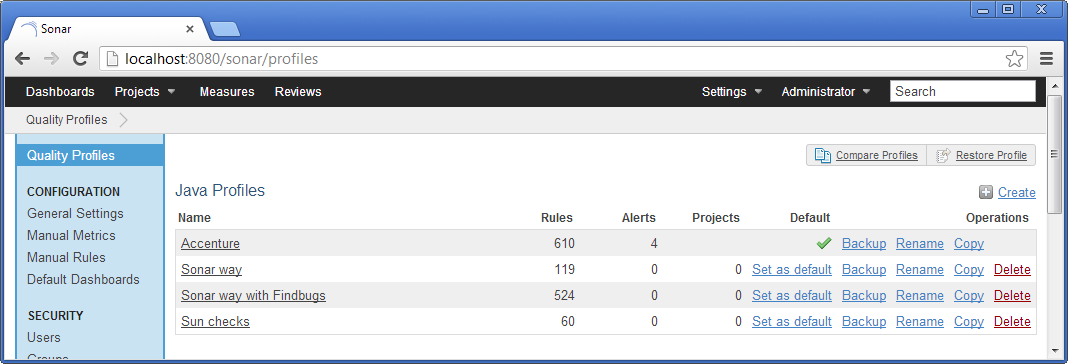
To import the profile into Sonar, on the top menu select **Settings ->** **Quality Profiles** and below the top menu select **Restore profile**:



On the next screen, browse for the downloaded quality profile file and click the **Restore profile** button:



It is recommended that the profile is set as the default for new projects:

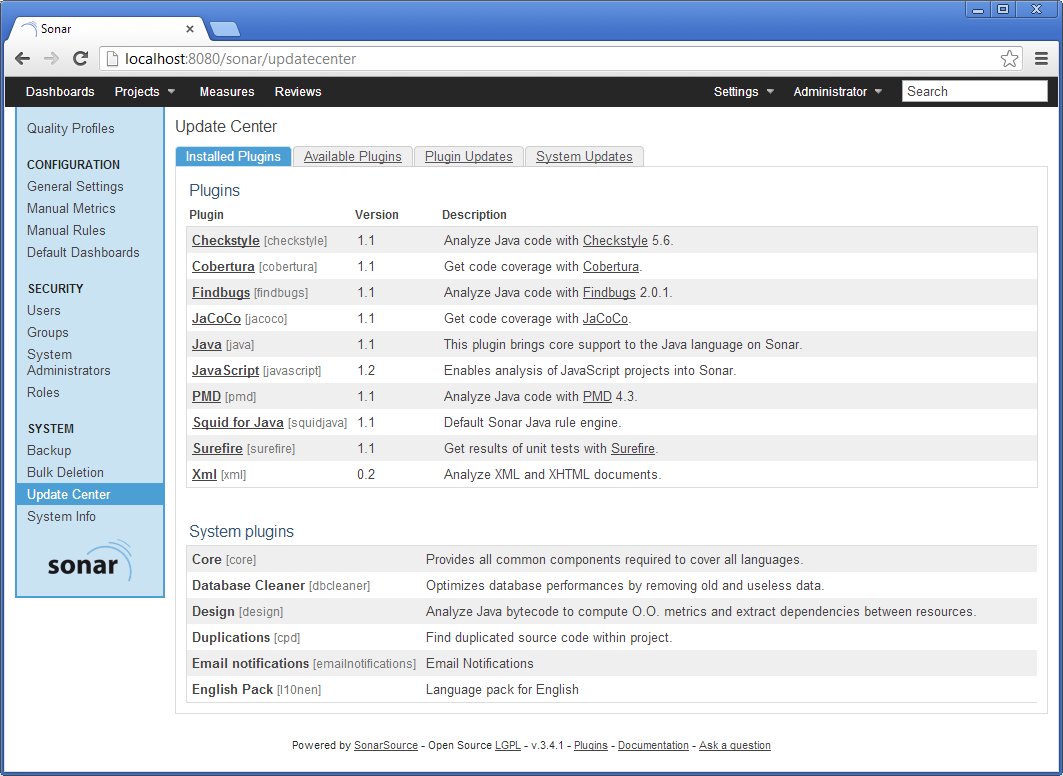


Once the quality profile is configured, Sonar is ready to analyse new Java projects.

# Sonar Update Centre

Sonar comes with an update centre that allows easy installation and upgrade of Sonar plugins. Although Sonar is bundled with some core plugins that enable typical use cases you can install new plugins any time that will extend Sonar functionality and match your needs. Note that there are free and commercial plugins, so you may need to review individual plugin licenses before installing them.

To access the update centre, in the top menu select **Settings** and then in the left menu select **Update Center**:



In the update centre screen there are four tabs: **Installed Plugins** shows installed plugins, **Available** **Plugins** shows the catalogue of available plugins, **Plugin Updates** shows the list of plugins that can be upgraded and allows upgrading them and **System Updates** shows whether Sonar can be upgraded and instructions to do the upgrade.

The plugin catalogue includes additional languages (C, Cobol, Flex, PL/SQL, Visual Basic…), additional analysers and metrics (JMeter, Jira, Trac, C-rules), governance models (Quality Index, SIG MM, SQALE), integration with other tools and extended visualization and reporting (PDF Report).

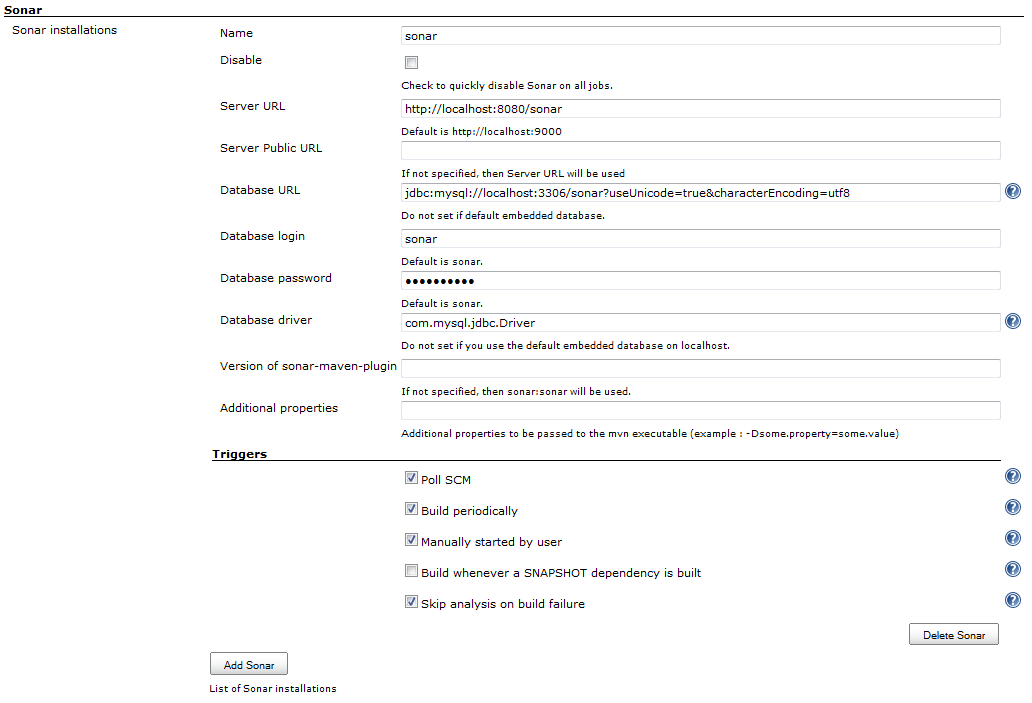
# Installing and Configuring the Jenkins Sonar Plugin

Jenkins and Sonar work closely together to provide the best of the Continuous Integration and Continuous Quality Assurance practices. Through the use of the Jenkins Sonar plugin, we can launch a Sonar analysis on selected jobs and link the Jenkins dashboard with Sonar dashboard to review analysis results.

To install the Jenkins Sonar plugin, just go to the Jenkins update centre as described in previous section and search for the Sonar plugin to install it. Restart Jenkins to finish off with the plugin installation.

Although in the rest of the section we are going to describe how to configure the plugin parameters, note that the Java Build Accelerator installer already includes Jenkins and Sonar configured to work together.

Once installed, log in to the Jenkins dashboard as administrator and in the left menu select **Manage Jenkins**. Then select **Configure System** in the list of management tasks. Now that the plugin is installed, near the end of the screen you will find a **Sonar** section. In this section we can add as many Sonar instances as needed, selecting for each one the connectivity parameters and details on how to trigger the Sonar analysis.



The connectivity parameters, and default values when using recommended setup and settings, are:

* **sonar.jdbc.url** = jdbc:derby://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8
* **sonar.jdbc.driver** = <Apache derby driver>
* **sonar.jdbc.username** = sonar
* **sonar.jdbc.password** = sonarsonar
* **sonar.host.url** = http://localhost:<port>/sonar

Don’t forget to update previous settings in the case that you are not following the recommended setup and settings, e.g. you are using a different database engine or have Jenkins and Sonar in different boxes.

The **Triggers** subsection can be used to globally enable or disable Sonar analysis from selected types of builds. For example, we may enable Sonar analysis in periodic builds but not in those triggered by an SCM poll. It is recommended, unless needed, to globally activate Sonar in all builds except when triggered by a dependency.

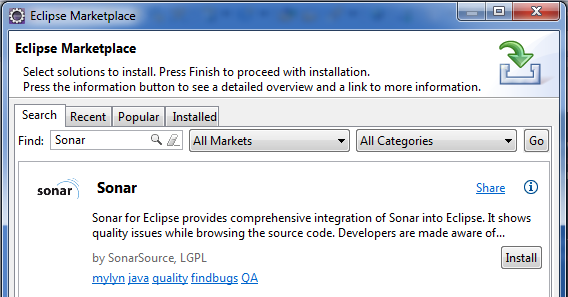
# Installing and configuring the Eclipse Sonar plugin

Eclipse can be configured to work with a remote Sonar instance. It is recommended that each developer need not install a copy of Sonar instead they can connect to the remotely setup instance as part of continuous integration using the Eclipse Sonar plugin. This way the developers can view the violations in Sonar within Eclipse without having to leave the IDE.

## Installing the Eclipse Sonar plugin

### Getting Sonar using Eclipse Marketplace

By searching for “Sonar” in the Eclipse Marketplace

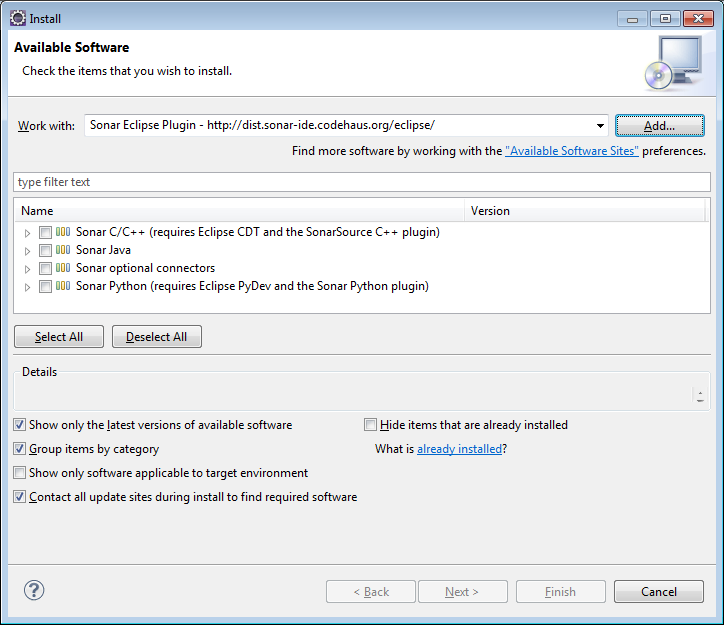


### Installing Sonar using the Eclipse update center

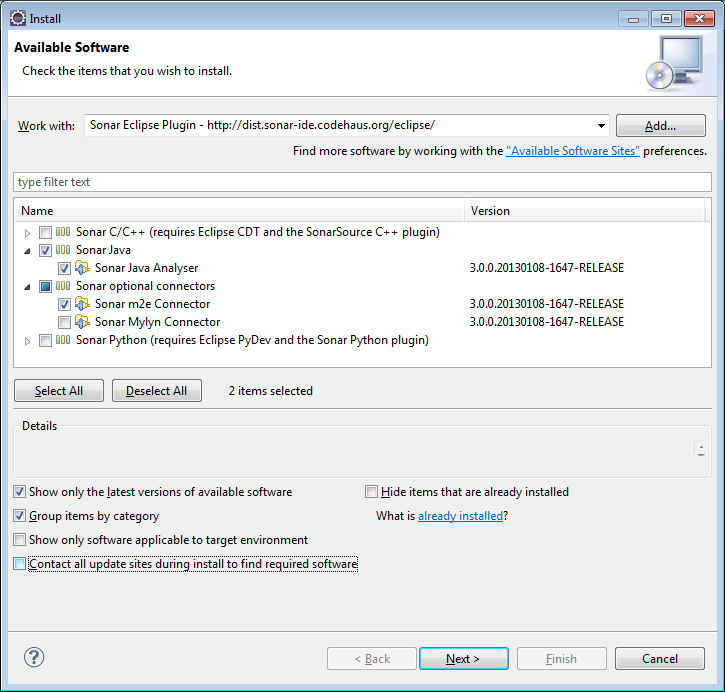
1. In Eclipse Go to **Help -> Install New Software.**
2. Add the following url as a update site: <http://dist.sonar-ide.codehaus.org/eclipse/>

### Installation

This will list available components to download and install:



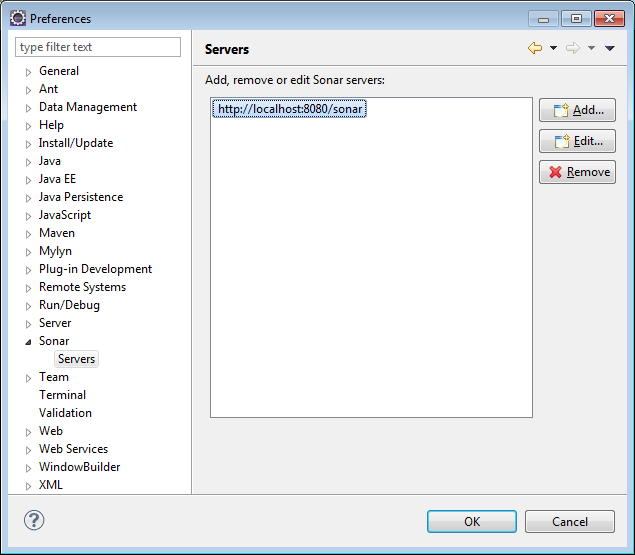
1. Select the Sonar Java and Sonar m2e Connector to install and complete the installation.



1. Once installation is complete restart Eclipse as required.

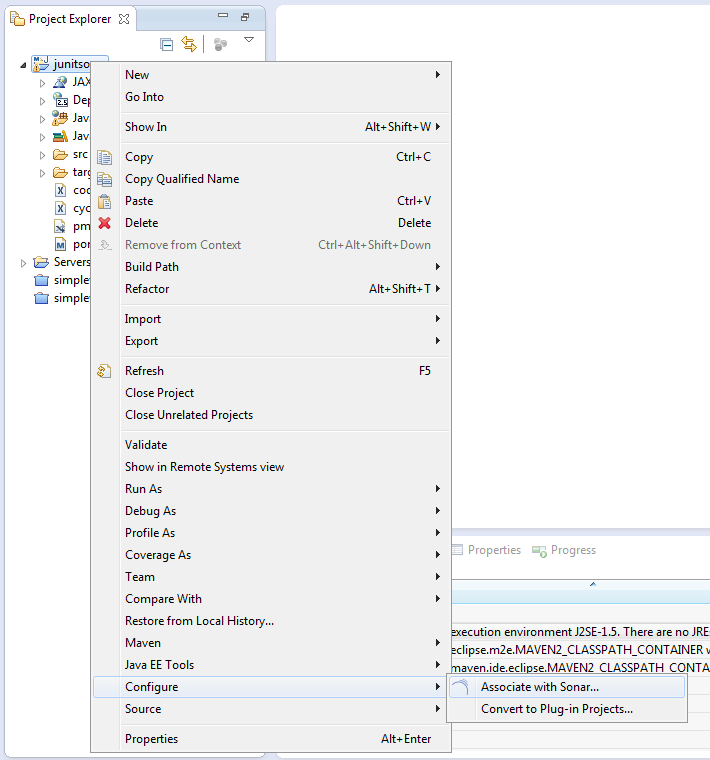
## Configuring the Eclipse Sonar plugin

Once installation is complete Eclipse needs to be configured to connect to a Sonar instance. Navigate to **Windows->Preferences** and under **Sonar->Servers** add entry for your remote Sonar server.

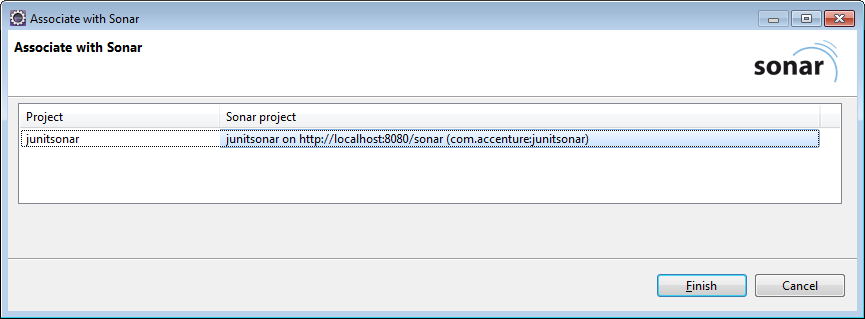


Test the connection to make sure Eclipse is able to connect to the remote Sonar instance.

To link the project for the first time with the Sonar instance right click on the project explorer and select **Configure->Associate with Sonar.**

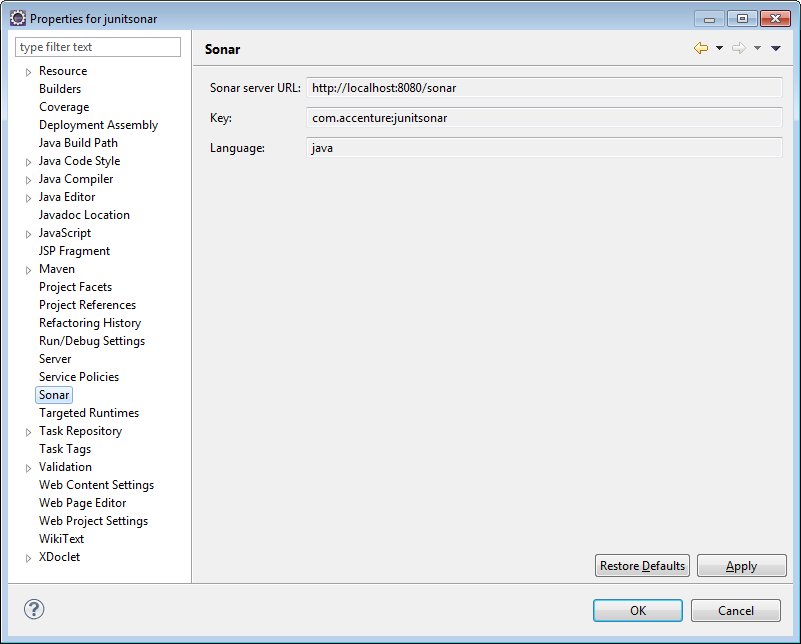


The Sonar instance configured should show up for association.



Maven projects are automatically associated with Sonar. By default the url used during this association is localhost. In case you have a remote instance configured you may want to remove the link to the default association and link to correct instance of Sonar.

Right click on the Project and under properties navigate the Sonar section to verify if the project has been associated to the right Sonar instance.



A project can be disassociated with Sonar by selecting **Sonar->Remove Sonar nature** at the menu options available on the right click of the project.

# References

Apache Tomcat: <http://tomcat.apache.org/>

Sonar: <http://www.sonarsource.org/>

Sonar on-line documentation: <http://docs.codehaus.org/display/SONAR/Documentation>

# Document Control

## Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Author | Approver | Comment |
| 0.1 | 01/09/2015 | Sandip Mante |  | First draft version |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |