

|  |
| --- |
|  |

**Submitted For**

**TeamCity Demo**

**Submitted By:**

**Bhargav Reddy K**

Cybage Software Pvt. Ltd.

Cybage Towers, Pune - 411014

# Table of Contents

[Table of Contents 2](#_Toc478652439)

[1. INSTALLATION OF TEAMCITY 3](#_Toc478652440)

[2. BUILD AGENT 3](#_Toc478652441)

[3. TEAMCITY PLUGINS 4](#_Toc478652442)

[4. INTEGRATION OF SONARQUBE 6](#_Toc478652443)

[5. SETUP FOR MAVEN PROJECT 6](#_Toc478652444)

[6. RUN THE FIRST BUILD 6](#_Toc478652443)

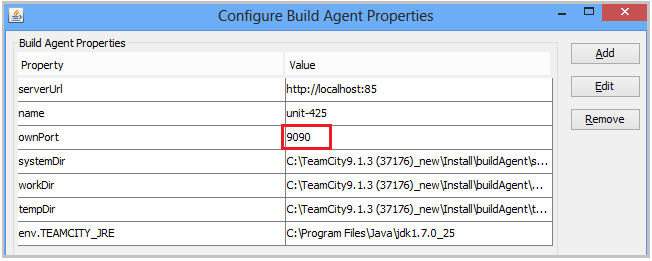
# Installation of teamcity

**On Windows**

Run the .exe file and follow the instructions of  the TeamCity Setup wizard. The TeamCity web server and one build agent will be installed on the same machine.

During installation, you can configure the following:

1. The [TeamCity home](https://confluence.jetbrains.com/display/TCD9/TeamCity+Home+Directory) where TeamCity will be installed.
2. Whether the TeamCity server and agent will run as Windows services.
3. The server port: **80** is the default port, which can be already used by other applications (Change the server port if it is already in use.
4. The agent port: **9090** is the default for incoming connections from the server. Change the own Port property if the default port is already in use.



# Build Agent

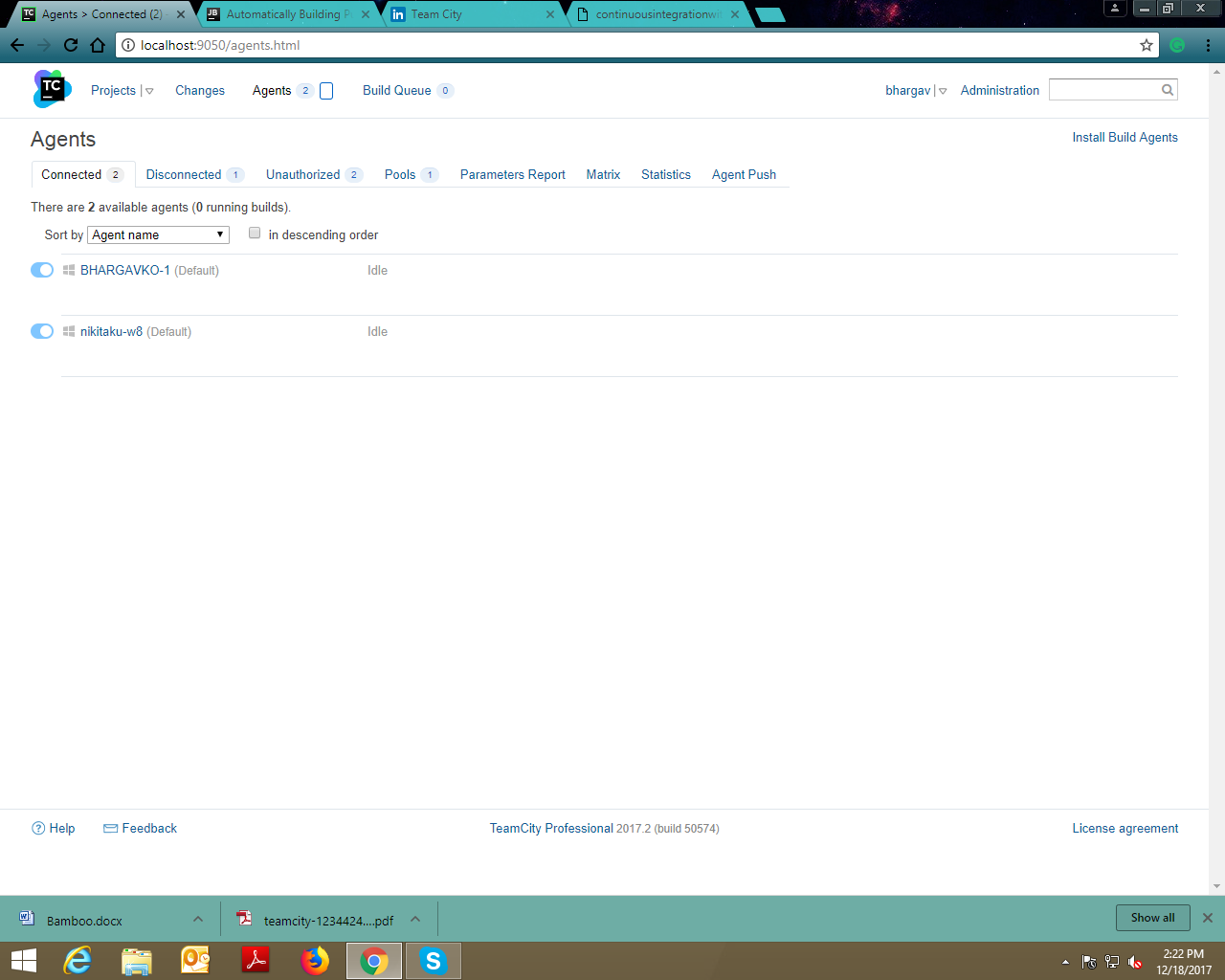
A TeamCity Build Agent is a piece of software which listens for the commands from the TeamCity server and starts the actual build processes. It is [installed and configured](https://confluence.jetbrains.com/display/TCD10/Setting+up+and+Running+Additional+Build+Agents) separately from the TeamCity server. An agent can be installed on the same computer as the server or on a different machine (the latter is a preferred setup for server performance reasons); an agent can run the same operating system (OS) as the TeamCity server or a different OS.

An Agent typically checks out the source code, downloads artifacts of other builds and runs the build process. An agent can run a single build at a time. The number of agents basically limits the number of parallel builds and environments in which your build processes are run.  
An Agent can run builds of any compatible build configuration.

The TeamCity server monitors all the connected agents and assigns queued builds to the agents based on [compatibility requirements](https://confluence.jetbrains.com/display/TCD10/Agent+Requirements), [Agent Pools](https://confluence.jetbrains.com/display/TCD10/Agent+Pools), Build Configuration restrictions configured for an agent and the selection algorithm described.

**Build Agent Status**

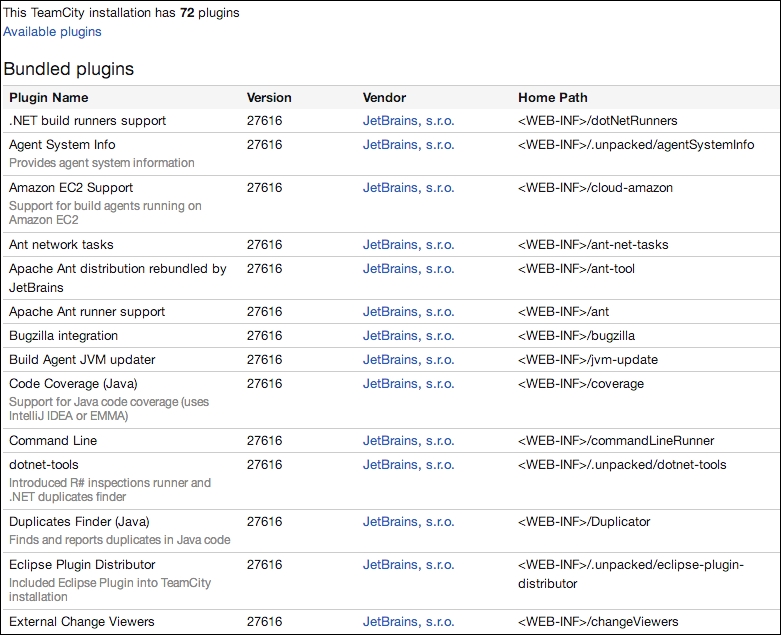
1. **Connected/ Disconnected -** An agent is connected if it is registered on the TeamCity server and responds to server commands, otherwise it is **disconnected**. This status is determined automatically.
2. **Authorized/ Unauthorized -** Agents are manually authorized via the web UI on the **Agents** page Only authorized build agents can run builds. The number of authorized agents at any given time cannot exceed the number of [agent licenses](https://confluence.jetbrains.com/display/TCD10/Licensing+Policy#LicensingPolicy-NumberofAgents) entered on the server.
3. **Enabled/ Disabled -** Agents are manually enabled/disabled via the [web UI](https://confluence.jetbrains.com/display/TCD10/Build+Agents+Configuration+and+Maintenance#BuildAgentsConfigurationandMaintenance-Enabling/DisablingAgentsviaUI). The TeamCity server only distributes builds to agents that are enabled.

****

# Teamcity Plugins

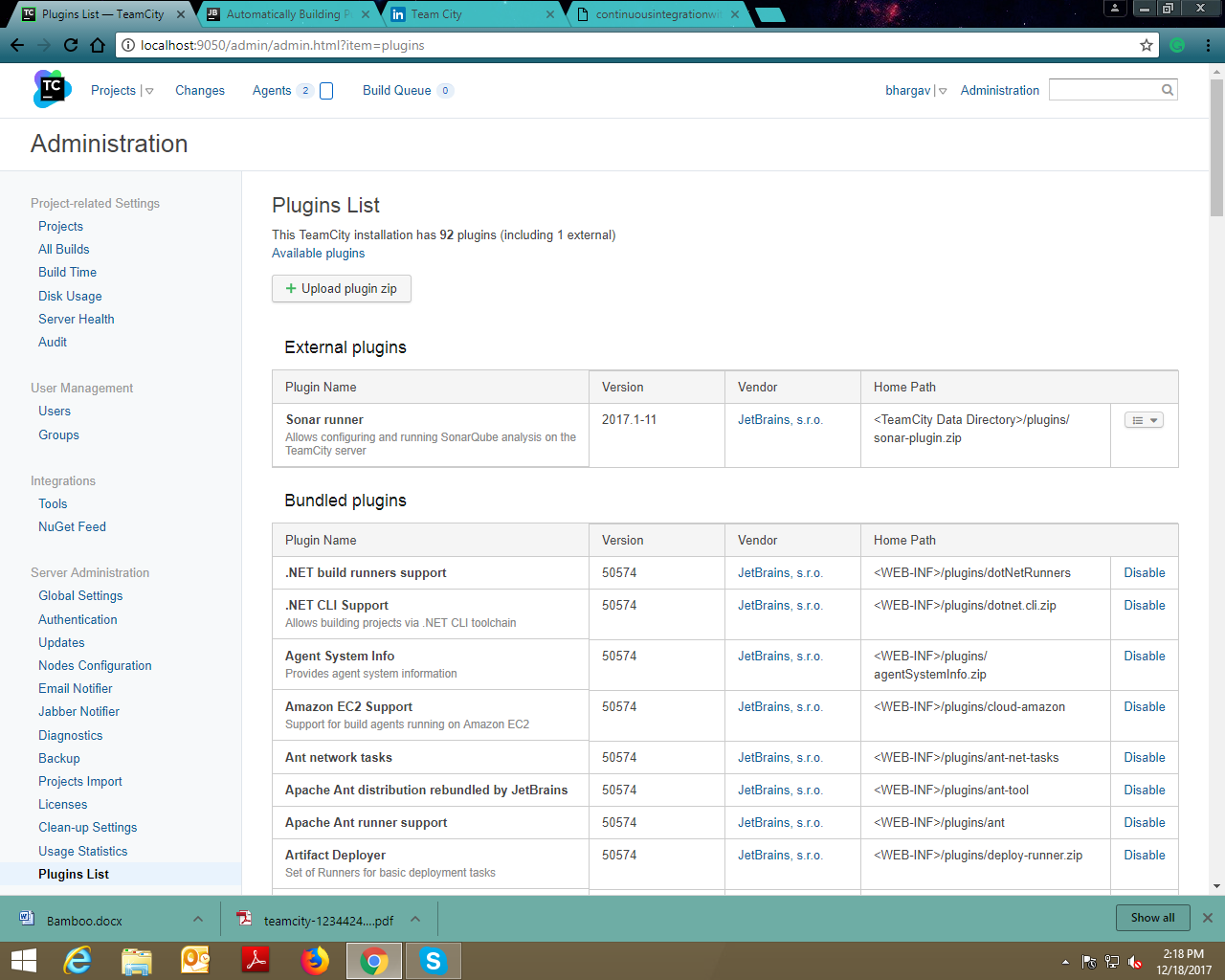
TeamCity has a plethora of plugins that can be used to extend and simplify the way TeamCity works. Many of the functionalities/features of TeamCity that come out of the box are in fact plugins that are bundled with TeamCity. These bundled plugins begin their life as external plugins that have to be installed, and then became bundled due to their usage and usefulness.

The list of bundled plugins in a TeamCity server can be seen from **Administration** | **Plugins List** as shown in the following screenshot:



***The general steps involved in installing a plugin to TeamCity are :***

1. Stop the TeamCity server.
2. Copy the zipped plugin to <TeamCity Data Directory>/plugins (TeamCity Data Directory is the directory where the TeamCity server's data is installed.)
3. Start the TeamCity server. TeamCity will decompress the plugin and start using it if everything is fine with the plugin.

******

SonarQube plugin is added in the External plugins.

**The plugins can also be uploaded directly in the plugins List as a zip file.**

# Integration of sonarqube

Download the plugin from the public [TeamCity server](http://teamcity.jetbrains.com/project.html?projectId=TeamCityPluginsByJetBrains_TeamCitySonarQubePlugin&tab=projectOverview) and install it as described [here](https://confluence.jetbrains.com/display/TCD9/Installing+Additional+Plugins).

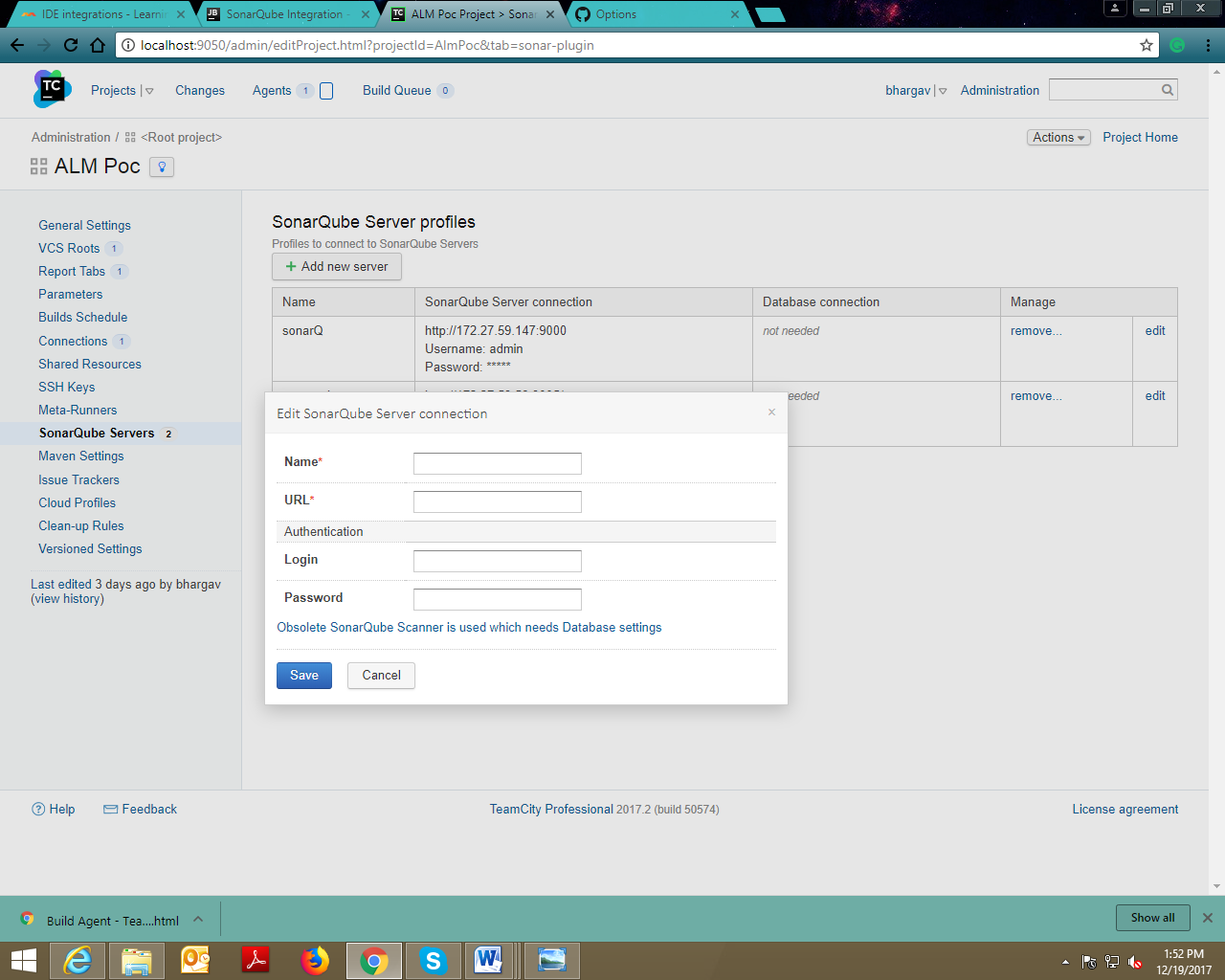
After the SonarQube Runner plugin is installed, the SonarQube Servers page appears in the project settings and the SonarQube Runner is added to the list of the available [runners](https://confluence.jetbrains.com/display/TCD9/Build+Runner) for a Build Step.

TeamCity 8.1 or greater is supported

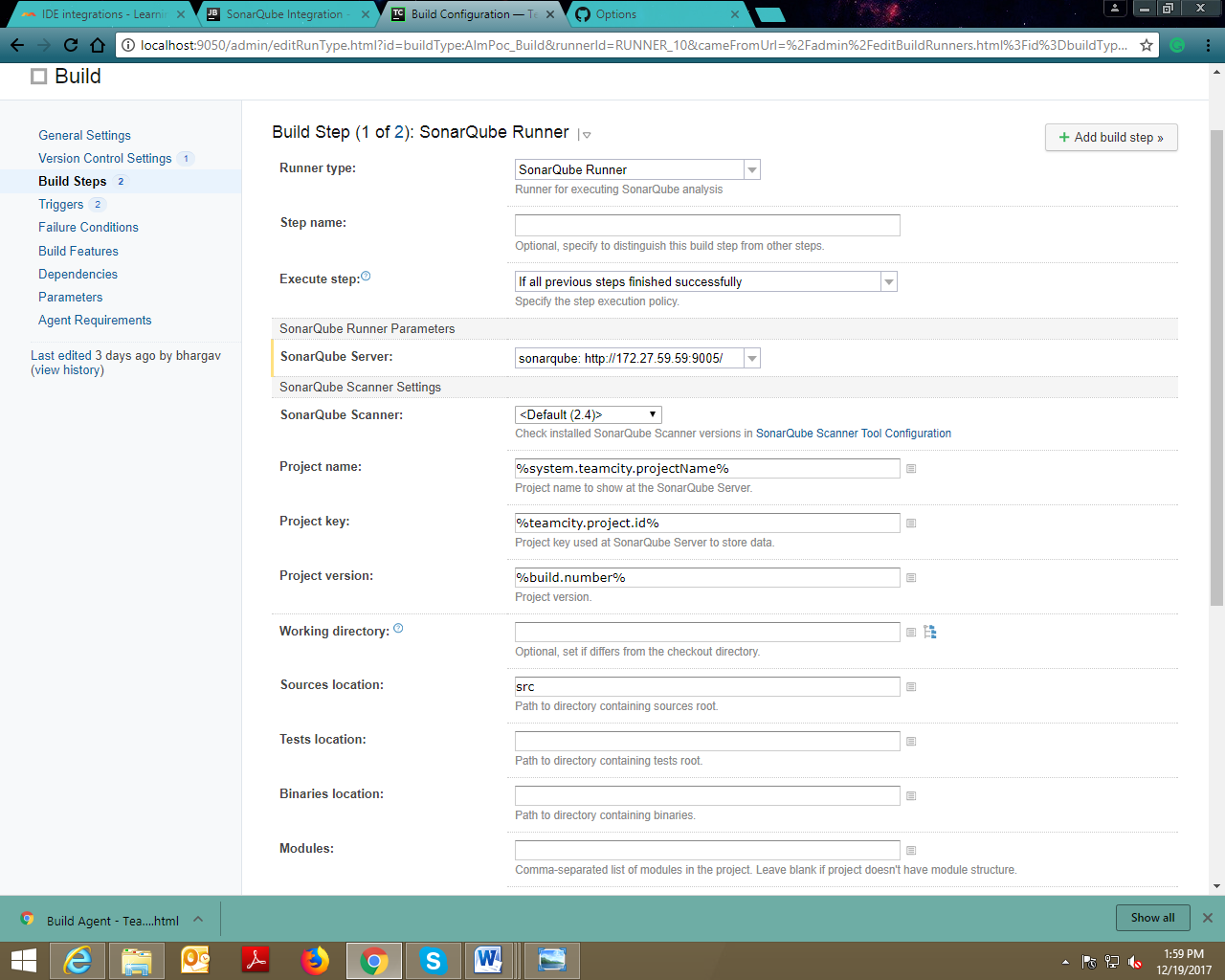
Configuring SonarQube Server Connection

To enable the integration, you need to create at least one connection to a SonarQube Server. The connection will be used by TeamCity to send data to the SonarQube Server.

SonarQube Servers connections are managed in the **SonarQube Servers** page of the Project Settings, where the SonarQube server name and Authentication.



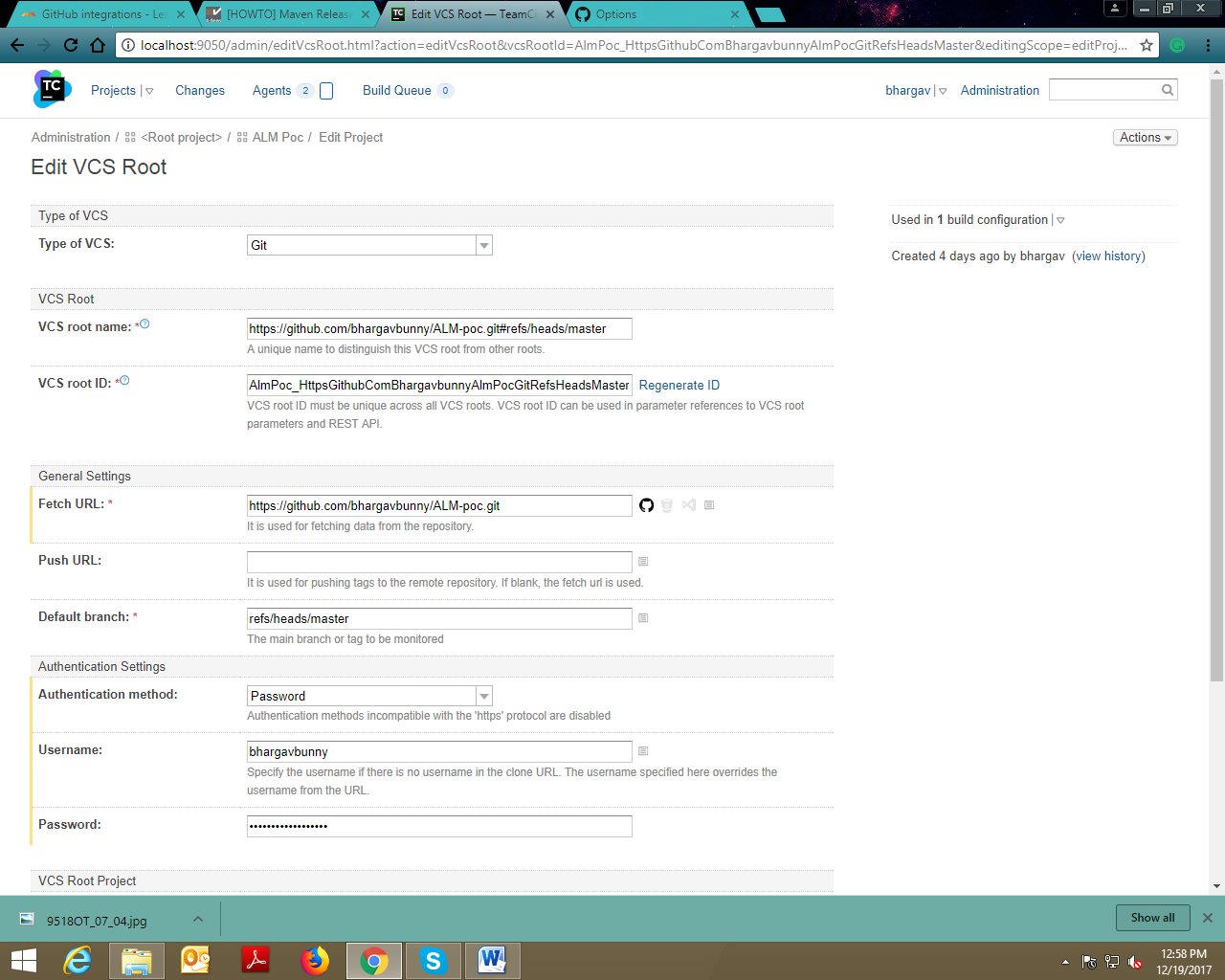
* Add SonarQube as a Build step in respective project.



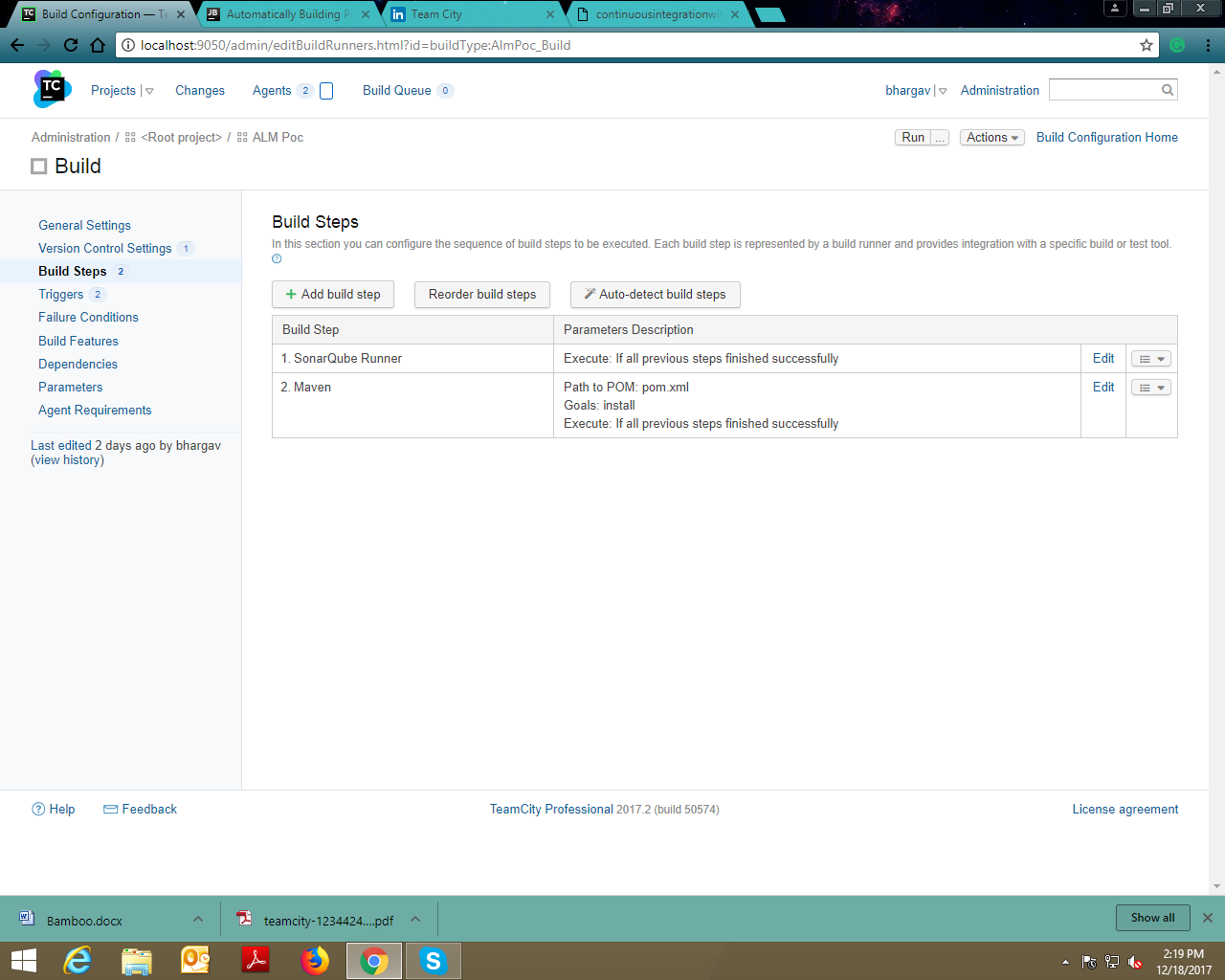
# setup for maven project

**Go to your TeamCity server and perform the following steps**

* Create a new project from URL
* Enter the repository URL, the username and the password as demanded
* Navigate to the settings of the newly created project
* Under VCS Roots you can edit the existing VCS root to define the branch to check out.

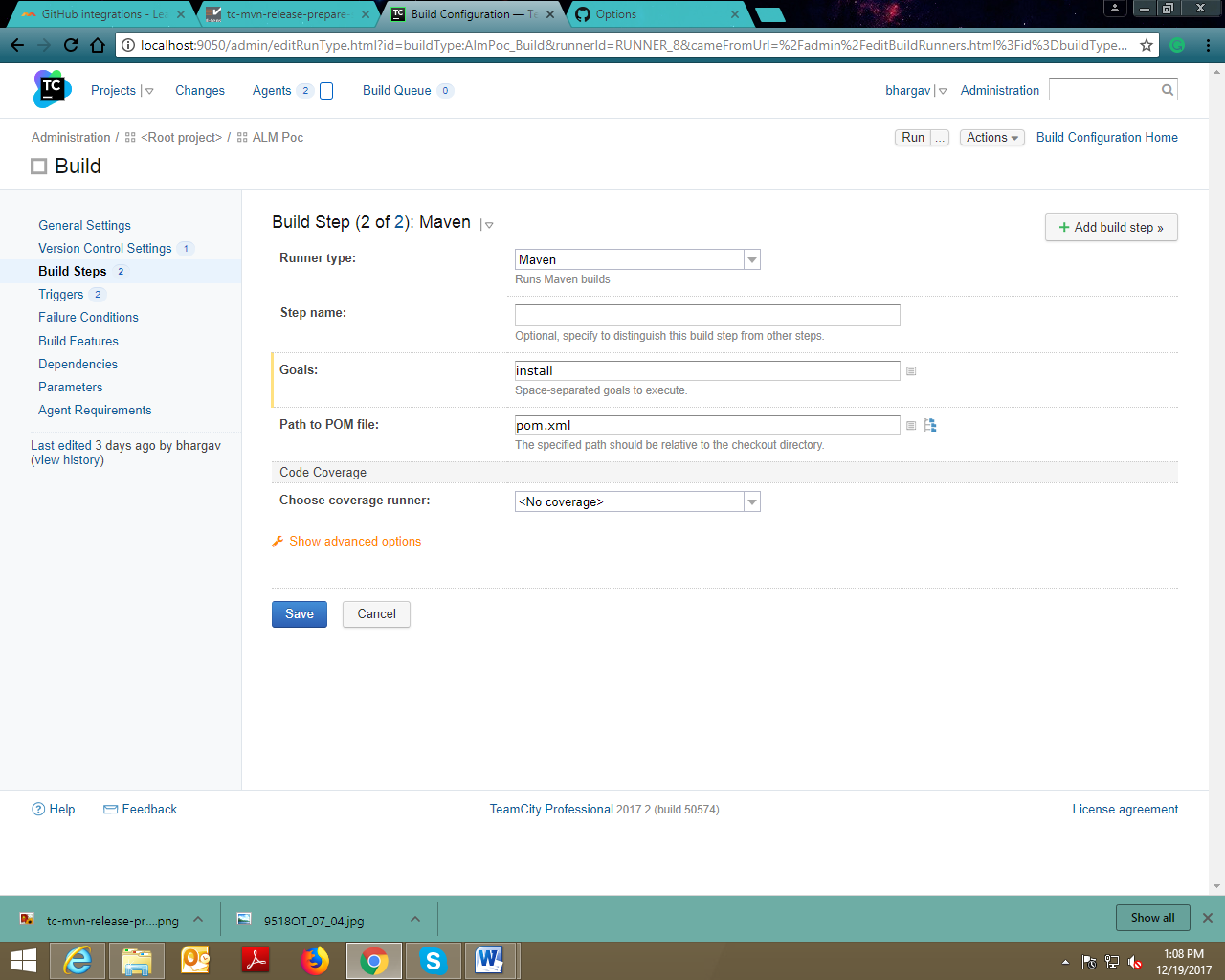


* Go to General Settings, choose the auto detected build configuration and click on edit



\

* Go to the Build Steps menu, delete the existing build step and create a new build steps
* Build step for install



# RUN THE FIRST BUILD

Your build currently has one build step, but you can add as many steps as you like and reorder them if required. You can add steps manually (1) or ask TeamCity to detect them automatically.

TeamCity will also suggest settings (2), such as triggers, failure conditions, and build features. Depending on the build configuration settings, it may prompt some additional options. You can follow the suggestions and add the settings to configure your build. You can always tweak the settings after running your first build.

Now you can launch your first build by clicking **Run**(3) in the upper right corner of the TeamCity web UI:

