



**Automate Selenium-Python**

**with**

**TeamCity Continuous Integration Server**

**Table of Content**

[I. Introduction 4](#_Toc466467822)

[II. Environments 4](#_Toc466467823)

[III. Basic Concepts 4](#_Toc466467824)

[IV. Setup New Project on TeamCity Server 5](#_Toc466467825)

[1. Create new project 5](#_Toc466467826)

[2. Project Configuration Settings 6](#_Toc466467827)

[a. Configuring Version Control System (VCS) Roots 6](#_Toc466467828)

[3. Build Configuration Settings 7](#_Toc466467829)

[a. Create new build configuration using a VCS URL 7](#_Toc466467830)

[b. Version control settings 8](#_Toc466467831)

[c. Config build steps manually 8](#_Toc466467832)

[V. Config Build Agents on localhost or other machines 10](#_Toc466467833)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Quang Tran | Nov 08 2016 | Initialize document | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Review and Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Role** | **Signature** | **Action** |
|  |  |  |  |
|  |  |  |  |

# Introduction

**Continuous Integration** is a software development practice in which developers commit code changes into a shared repository several times a day. Each commit is followed by an automated build to ensure that new changes integrate well into the existing code base and to detect problems early.

**JetBrains TeamCity** is a software suites that helps users to practice continuous integration on their project. TeamCity includes a user-friendly continuous integration (CI) server and an Agent component to perform automated building or executing source code.

This document provides the basic information to do the automated execution on TeamCity Server in a Selenium-Python project.

# Environments

1. Test Tools: Selenium 3.x for Python
2. Source version control: Git revision control systems. Here we use BitBucket.
3. Platform: Windows 7, Window 10, Mac OSX
4. Browser: Firefox, Chrome, Internet Explorer, Safari, Window Edge
5. CI Server: TeamCity 10.0.x

# Basic Concepts

The TeamCity build system comprises the **Web Server** and **Build Agents**.

* **TeamCity Server: This is a web server that monitors all the connected build agents, distribute queued builds to the agents based on compatibility requirements, and report the results. All information on the build results (build history and all the build-associated data except for artifacts and build logs), VCS changes, agents, build queue, user accounts and user permissions, etc. are stored in a database.**
* **Build Agent: A piece of software that actually executes a build process. It is installed and configured separately from the TeamCity server, i.e. the agent can be installed on a separate machine (physical or virtual, and it can run the same operating system (OS) as the server or a different OS. Build Agents in TeamCity can have different *platforms*, *operating systems*, and *pre-configured environments* that you may want to test your software on. Different types of tests can be run under different platforms simultaneously so the developers get faster feedback and more reliable testing results.**

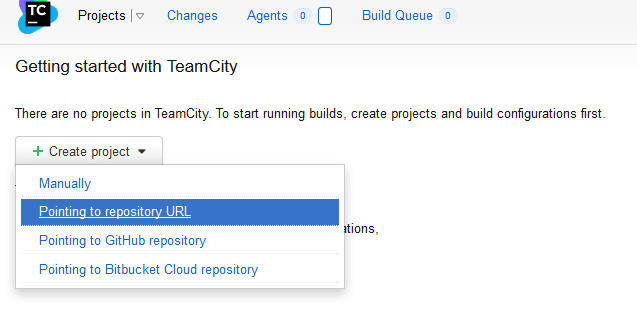
In TeamCity, there is a build life cycle:

* 1. The TeamCity server detects a change in your VCS Root and stores it in the database.
  2. The build trigger sees the change in the database and adds a build to the queue.
  3. The server finds an idle compatible build agent and assigns the queued build to this agent.
  4. The agent executes the Build Steps. While the build steps are being executed, the build agent reports the build progress to the TeamCity server sending all the log messages, test reports, code coverage results, etc. to the server on the fly, so you can monitor the build process in real time.
  5. After finishing the build, the build agent sends Build Artifacts to the server

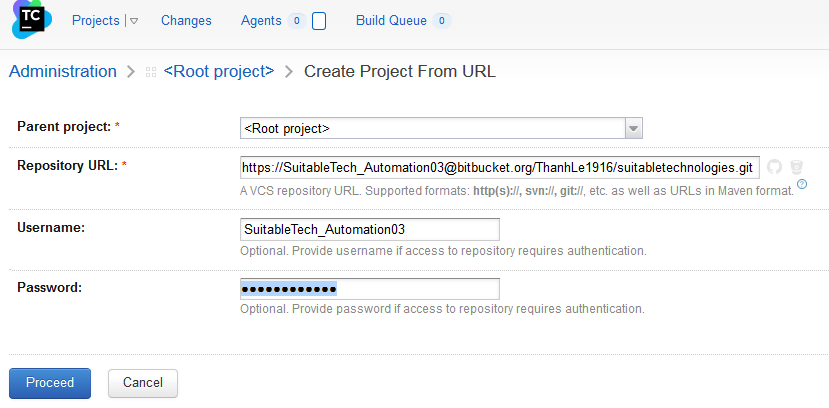
# Setup New Project on TeamCity Server

## Create new project

On the **Administration** | **Projects** page, select “Create project” menu and choose option “Pointing to repository URL”

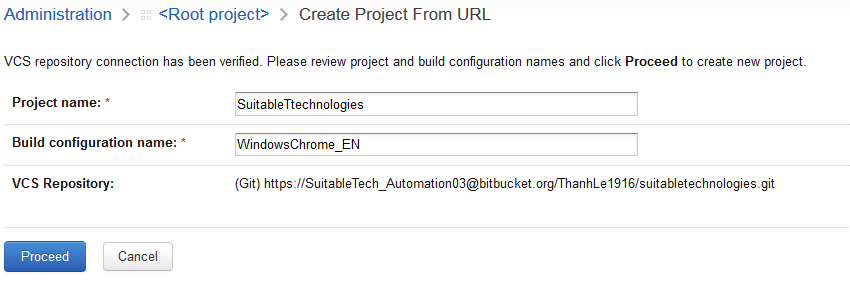


On the **Create New Project from URL** page, specify the project settings:



Click **Proceed** button.

On next page, specify the **Project** name and **Build congifuration name.**



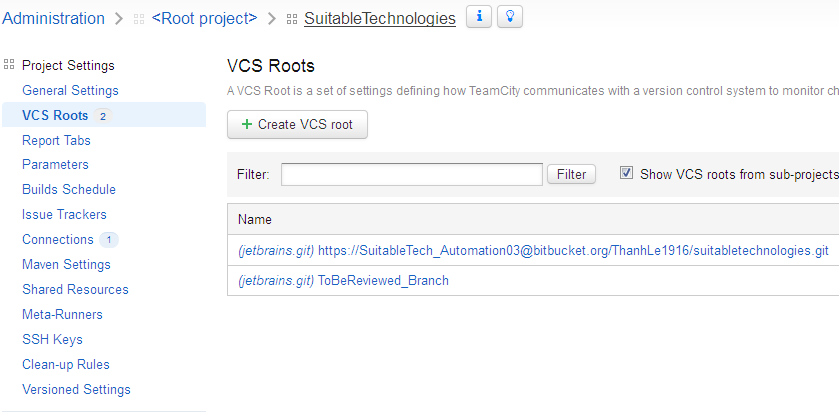
After click **Proceed** button, TeamCity will create a new project, a build configuration and VCS root.

## Project Configuration Settings

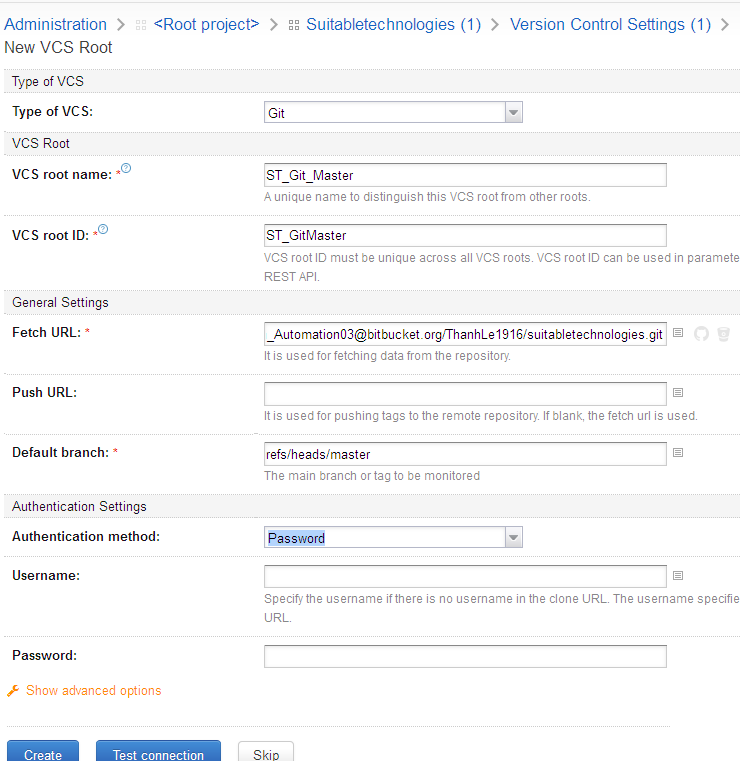
### Configuring Version Control System (VCS) Roots

A VCS root defines a connection to a version control system and consists of a set of settings (paths to sources, username, password, and other settings) that defines how TeamCity communicates with a version control (SCM) system to monitor changes and get sources for a build. A VCS root can be attached to a build configuration. You can add several VCS Roots to a build configuration or a template and specify portions of the repository to checkout and target paths.

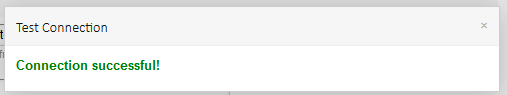
On the **Administrator | Project | Project Settings** page, click **Create VCS root** button

****

On the **New VCS Root** page, enter the Git server information:



Click the **Test connection** button to check the specified Git information. A notifcation will displays if the connection is checked successfully.



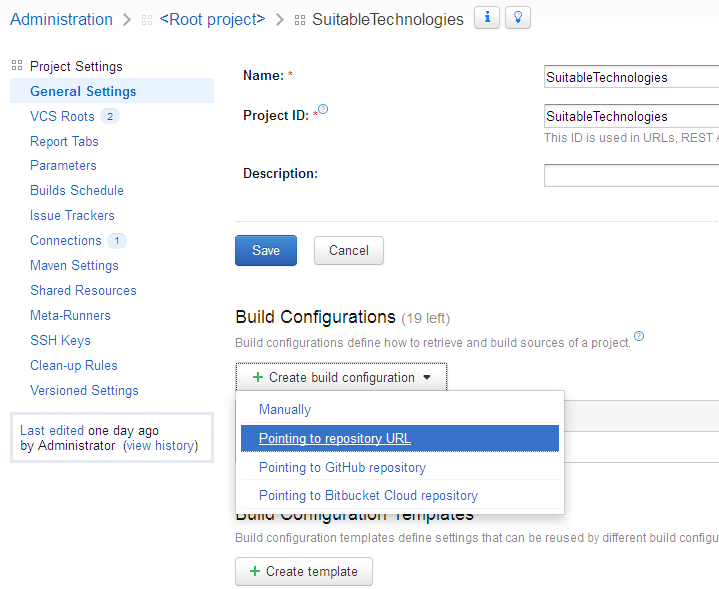
Then click **Create** button to finish VCS setting.

## Build Configuration Settings

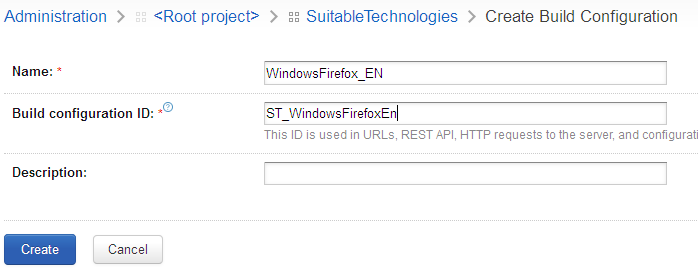
### Create new build configuration using a VCS URL

On the **Administration** > **Projects** page, select the desired project in the list. (Alternatively, open the project using the **Projects** popup, and click the **Edit Project Settings** link on the right). The **Project Settings** page opens.

On the **Project Settings | General Settings** page, **Build Configurations** section, click **Create build configuration** menu; then select **Manually** (or **Pointing to repository URL** if no VCS Root is configed before)

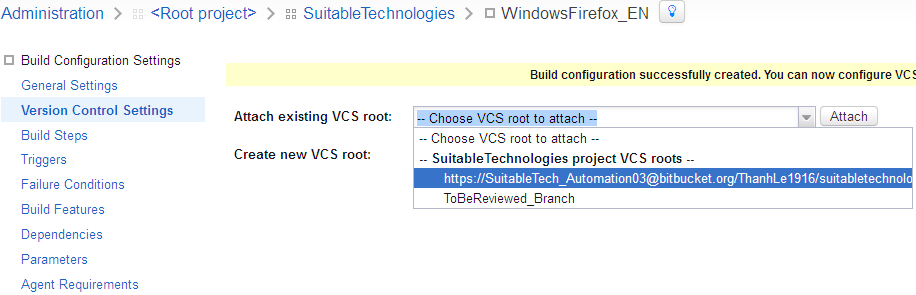


In the **Create Build Configuration** page, specify a name, Build configuration ID. Then click **Create** button.



### Version control settings

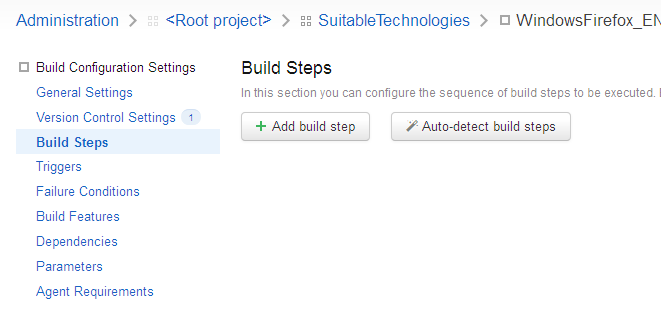
On the **Build Configuration Settings | Verion Control Settings** page, we can attach existing VCS root to the current build configuration, or create new VCS root. Here we use the pre-config VCS root.



Select an existing VCS root in the dropdown-list, and click **Attach** button.

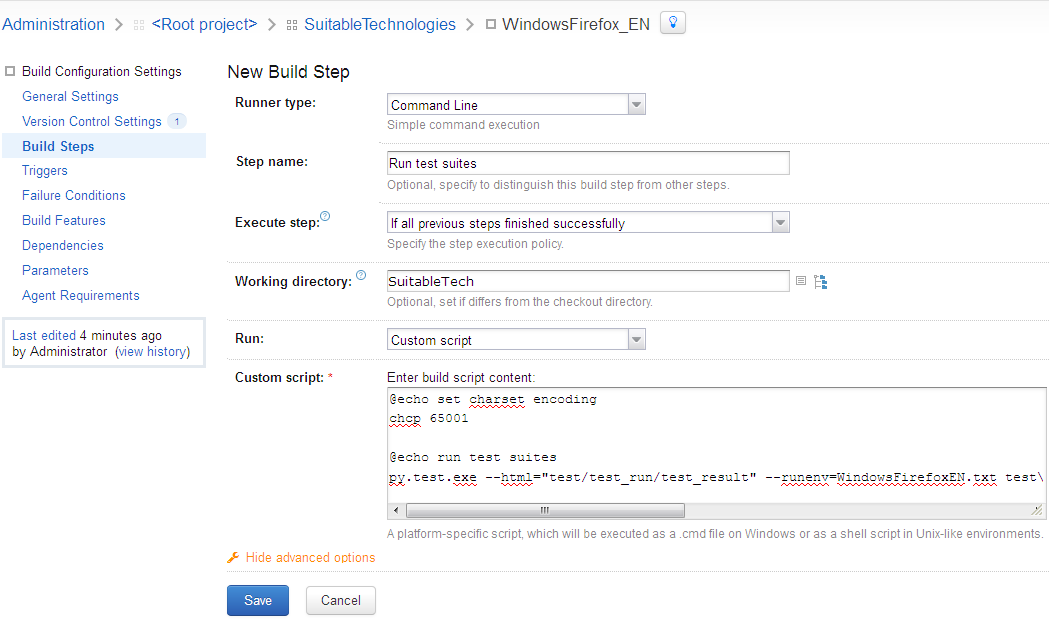
### Config build steps manually

On the **Build Configuration Settings | Build Steps** page, click **Add build step** button (or select the link **configure build steps manually**)

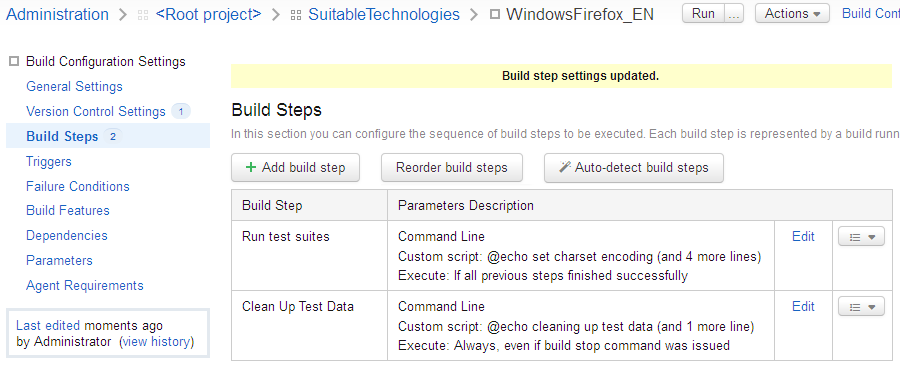


On the **New Build Step** page, select **Command Line** in the **Runner type** dropdown-list.

Specify *step name* <Run test suites>, *execute step*, *working directory*, *Run* <Custom script>, and *Custom script* as following image:



Click **Save** button to finish.



Now we can run the Build Configuration by doing click on the  **Run** button on the top page.

# Config Build Agents on localhost or other machines

We can setup a **Build Agents** on other machine by **using MS Windows installer** or **downloading a zip files and installing manually.**

Here we will install the agents via Zip file.

1. Make sure a JDK (JRE) 1.6+ (1.8 is recommended) is properly installed on the agent computer.
2. On the agent computer, make sure the **JRE\_HOME** or **JAVA\_HOME** environment variables are set (pointing to the installed JRE or JDK directory respectively).
3. In the TeamCity Web UI, navigate to the **Agents** tab.
4. Click the **Install Build Agents** link and select Zip file distribution to download the archive.
5. Unzip the downloaded file into the desired directory.
6. Navigate to the **<installation path>\conf** directory, locate the file called *buildAgent.dist.properties* and rename it to *buildAgent.properties*.
7. Edit the *buildAgent.properties* file to specify the TeamCity server URL and the name of the agent.
8. Under Linux, you may need to give execution permissions to the bin/agent.sh shell script.

To start the agent manually, run the following script:

1. **for Windows**: <installation path>\bin\agent.bat start
2. **for Linux and MacOS X**: <installation path>\bin\agent.sh start

Note that, we run test script by Selenium. So **do not configure the agent to be start automatically under Window services**b because the service run on **Session0** and cannot do test on GUI/Web.