

**TEAMCITY**

**Continuous Integration TOOL**

**What is Continuous Integration?**

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

We have many CI tools available in the market. But here, we will use Team city as our CI tool.

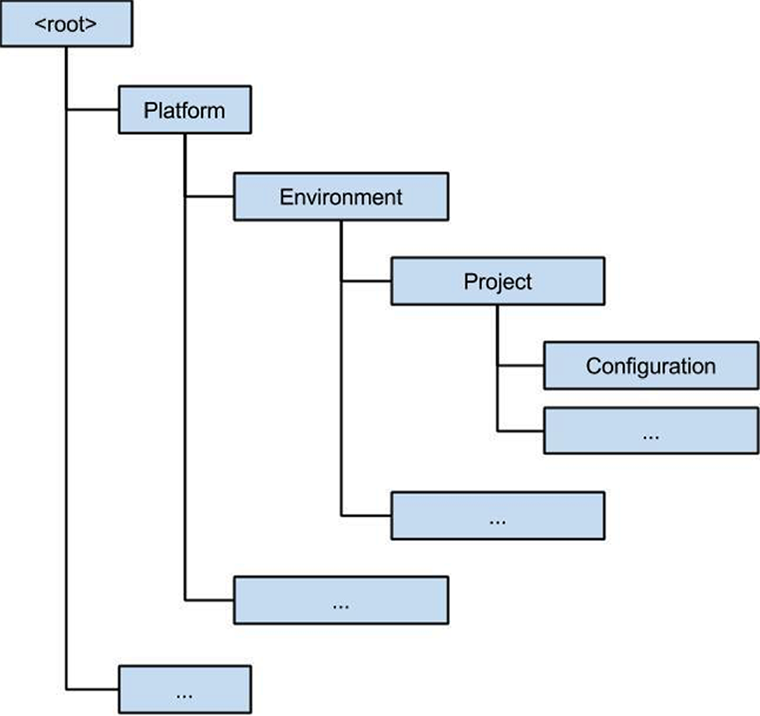
**TEAM CITY:**

Team city is a continuous integration tool we use in our project for automated builds.

**PROJECT STRUCTURE**

TeamCity allows the creation of hierarchical project trees that allow you to group projects by platform, environment, name, etc. So first we create the grouping by platform (Windows), and inside the platform group we create additional “environment” groups (Development, Staging, Production). Only then do we add the “project” for the required environment.

The final project tree looks like this:



**What can you do with TeamCity?**

* Run parallel builds simultaneously on different platforms and environments
* Optimize the code integration cycle and be sure you never get broken code in the repository
* Review on-the-fly test results reporting with intelligent tests re-ordering
* Run code coverage and duplicates finder for Java and .NET
* Customize statistics on build duration, success rate, code quality, and custom metrics and much more.

**Basic CI Workflow in TeamCity**

To understand the data flow between the server and the agents, what is passed to the agents, how and when TeamCity gets the results, let's take a look at a simple build lifecycle.

* 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](https://confluence.jetbrains.com/display/TCD10/Build+Artifact) to the server.

**Distributing Builds over Agents:**

TeamCity allows you to distribute builds over different build servers, called Agents. Every time you run a build, it searches for an available agent from the “Agent pool”. If the build requirements for that agent match, then the build is executed on it.

**Installing Build Agents:**

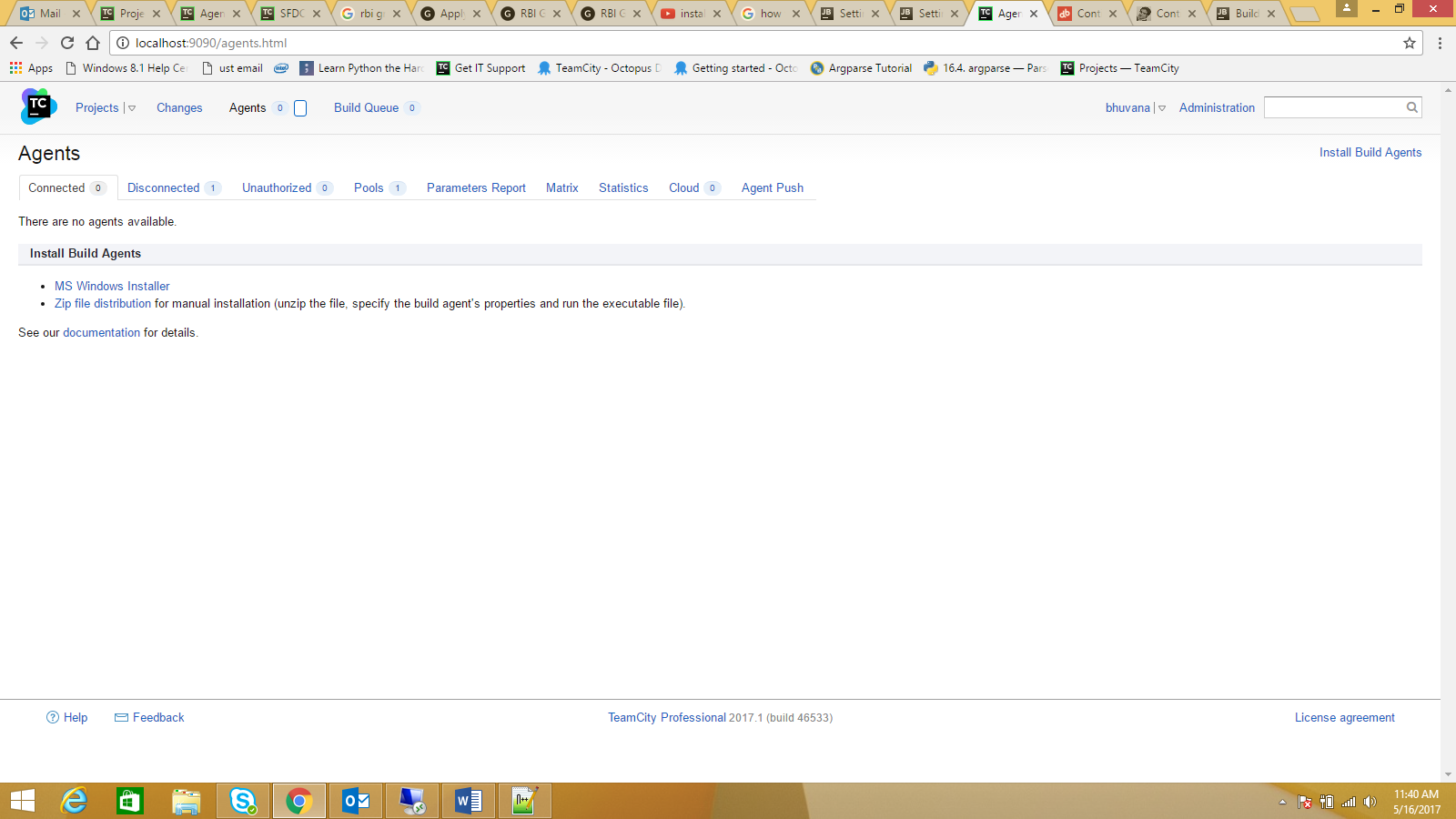
To build your code or deploy your software you need at least one agent. As you add more code you'll eventually need more.

An agent is installable software that runs one build or deployment job at a time. When your build or deployment runs, the system begins one or more jobs.

A TeamCity Build Agent is a piece of software which listens to the commands from the TeamCity server and starts the actual build processes. It is [installed and configured](https://confluence.jetbrains.com/display/TCD9/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. An agent can run the same operating system (OS) as the TeamCity server or a different OS.

To install a Build Agent in Team city, go to 🡪 Agents

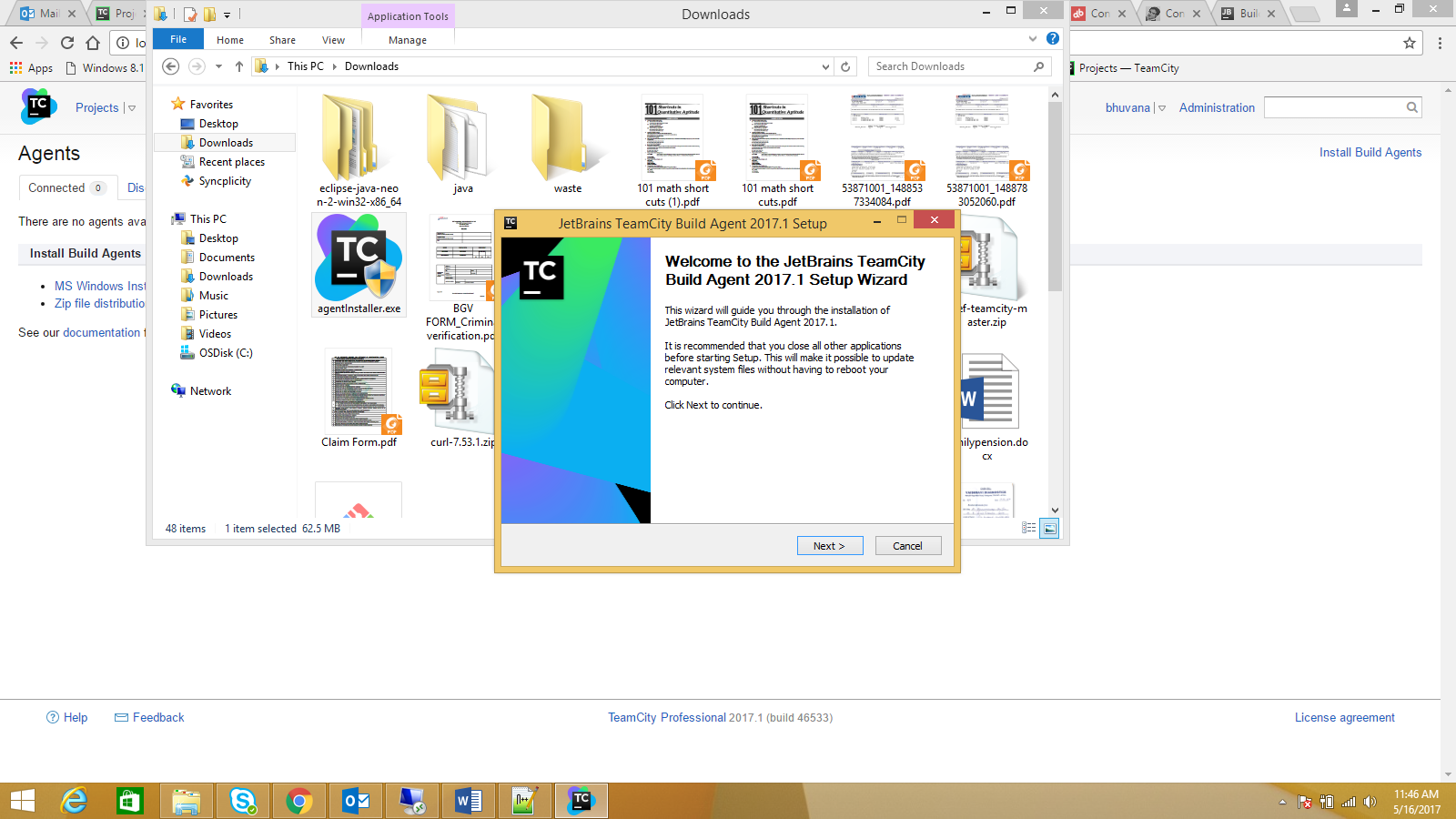


Install a build agent using any of the following options:

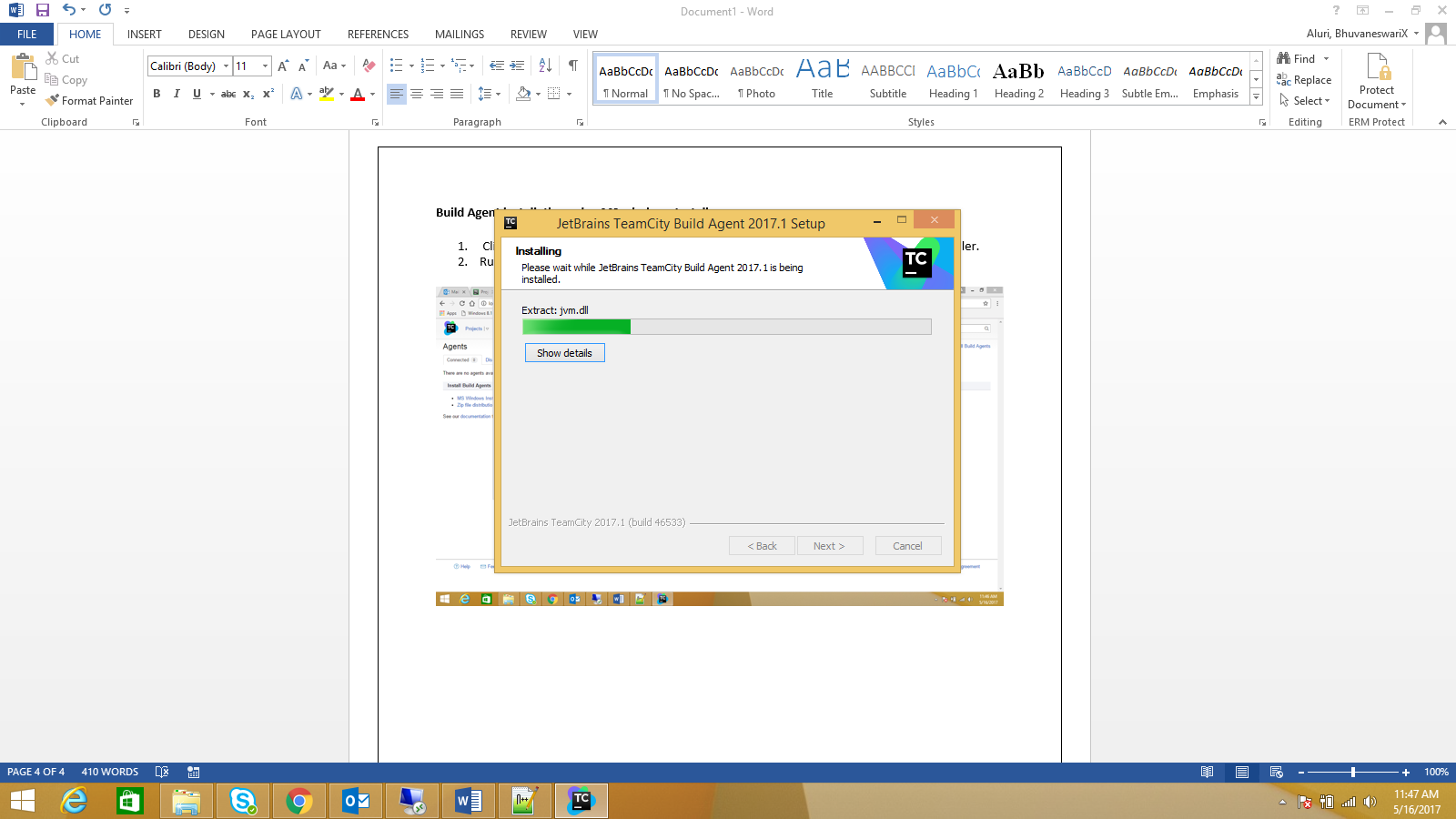
* Using MS Windows installer
* By downloading a zip file and installing manually.

**Build Agent installation using MS windows Installer:**

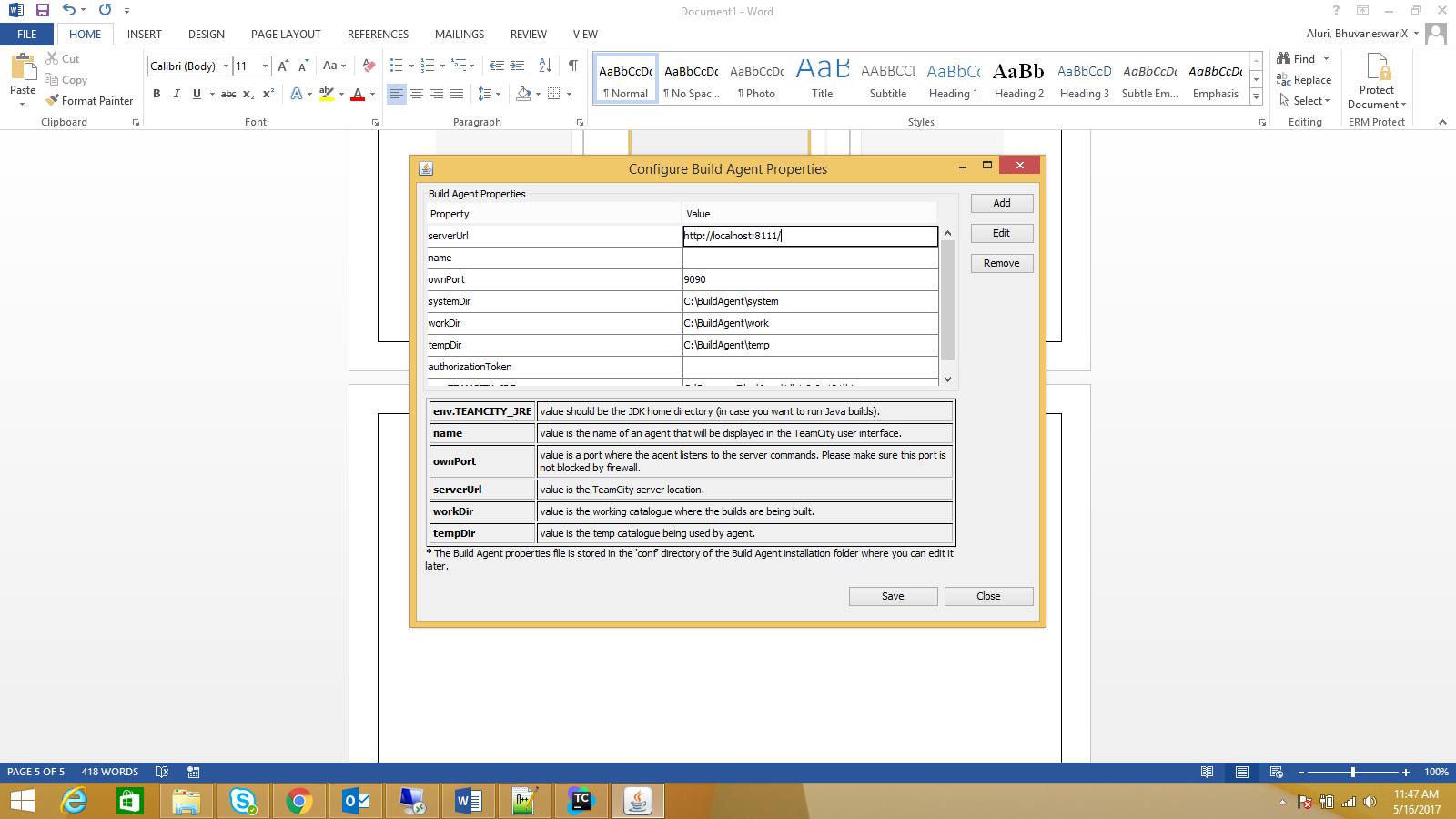
1. Click the Install Build Agents link and select MS Windows Installer to download the installer.
2. Run the agentInstaller.exe Windows Installer and follow the installation instructions.



Follow the instructions as per the next-next installation.



In this page, we need to provide the Build Agent URL and the port.



* After installation, configure the agent specifying its name and the address of the TeamCity server in the [conf/buildAgent.properties](https://confluence.jetbrains.com/display/TCD10/Build+Agent+Configuration) file.
* [Start](https://confluence.jetbrains.com/display/TCD10/Setting+up+and+Running+Additional+Build+Agents#SettingupandRunningAdditionalBuildAgents-StartingtheBuildAgent) the agent. If the agent does not seem to run correctly, please check the [agent logs](https://confluence.jetbrains.com/display/TCD10/Viewing+Build+Agent+Logs).

When the newly installed agent connects to the server for the first time, it appears on the Agents page, unauthorized agents tab visible to administrators/users with the permissions to authorize it.

Agents will not run builds until they are authorized in the TeamCity web UI. The agent running on the same computer as the server is authorized by default.

**Setting up Projects in Team city:**

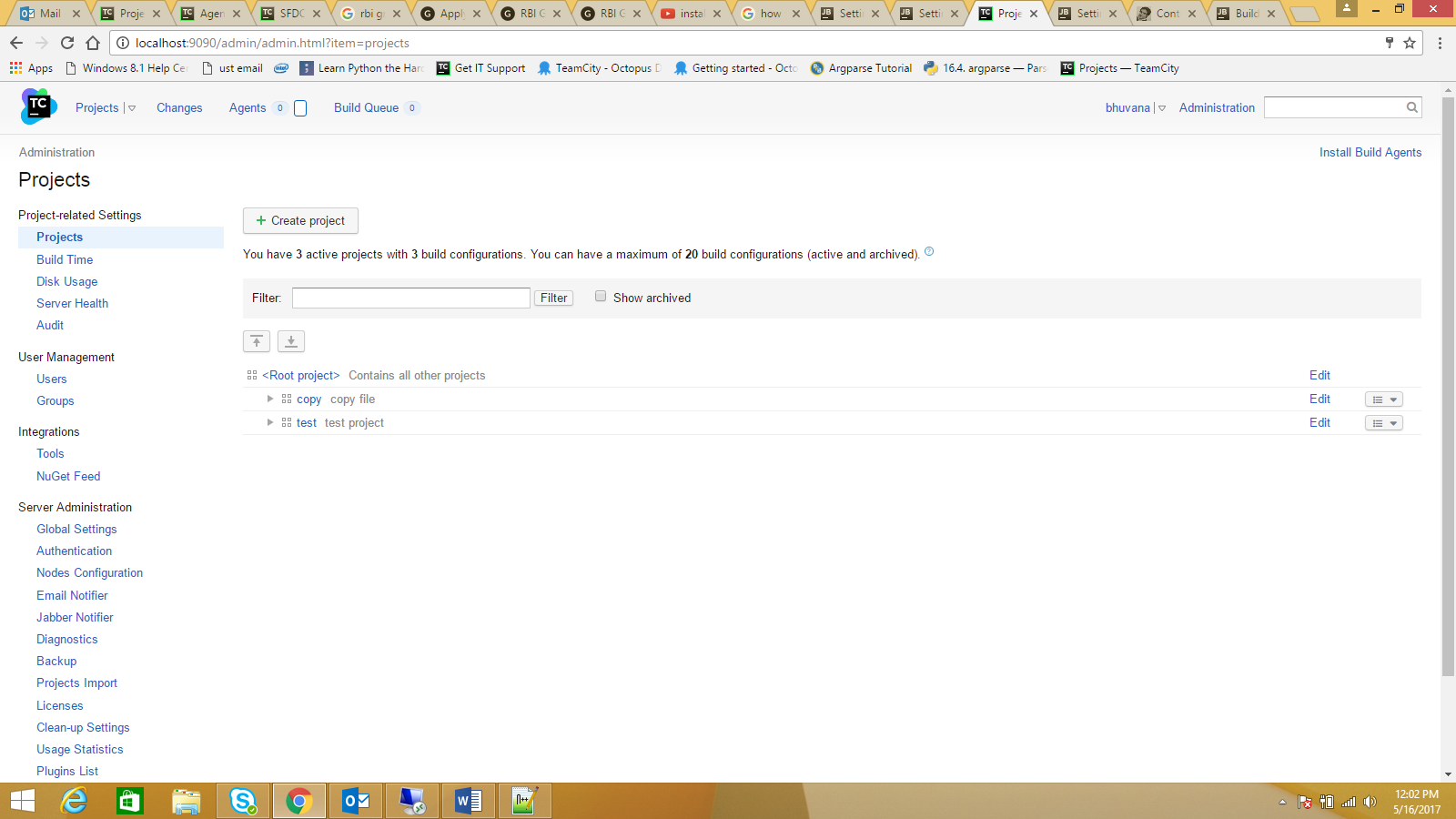
Now, we have a running TeamCity server and a user with System administrator rights and Build agents to run jobs and you’re logged into the admin console.

So before anything else we need to create a TeamCity project which is a simple grouping of build configurations.

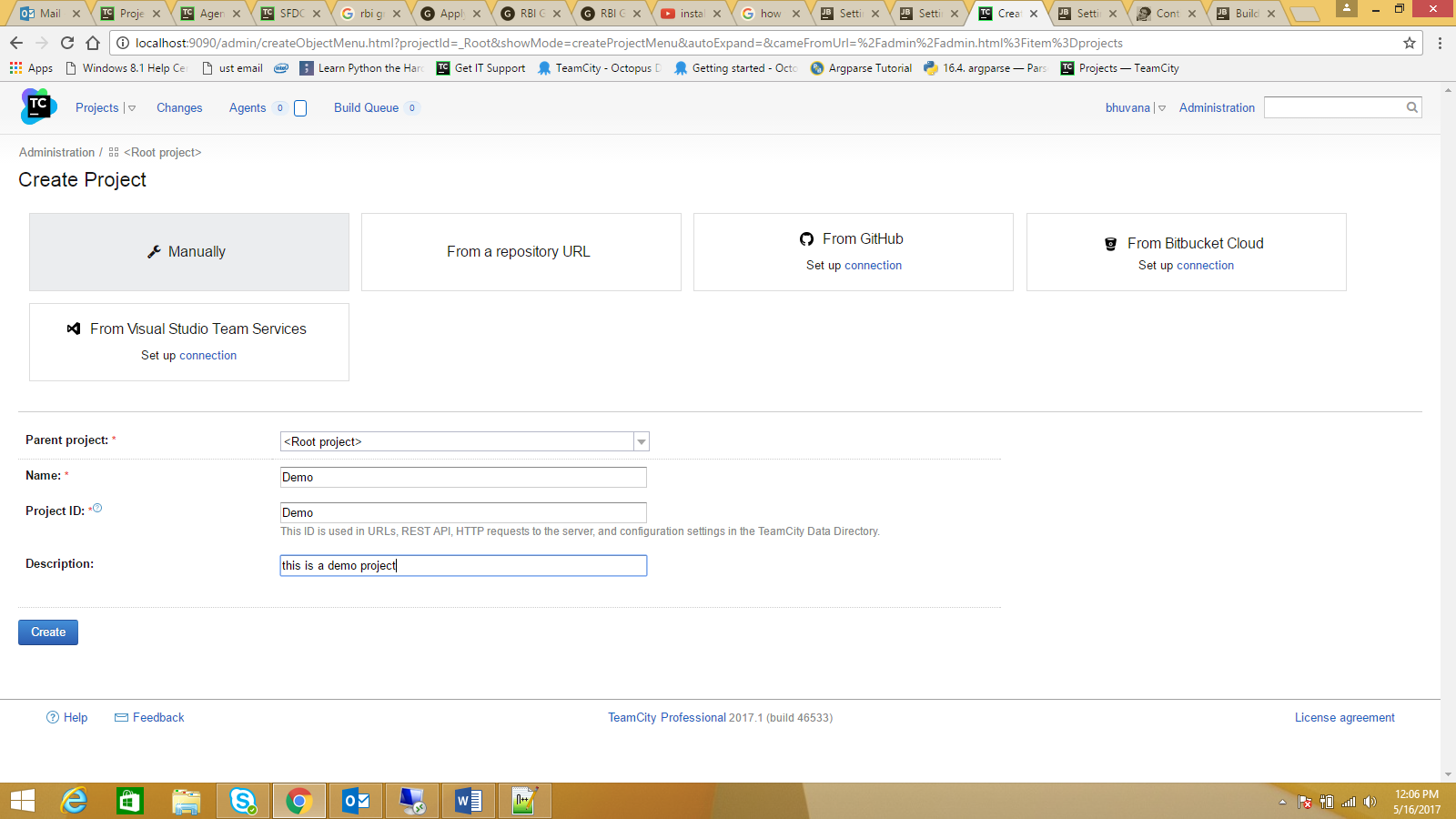
**Creating a TeamCity project:**

On the TeamCity admin console go to Administration (and click on the Projects from the left navigation bar). That takes you to a page showing a list of projects.

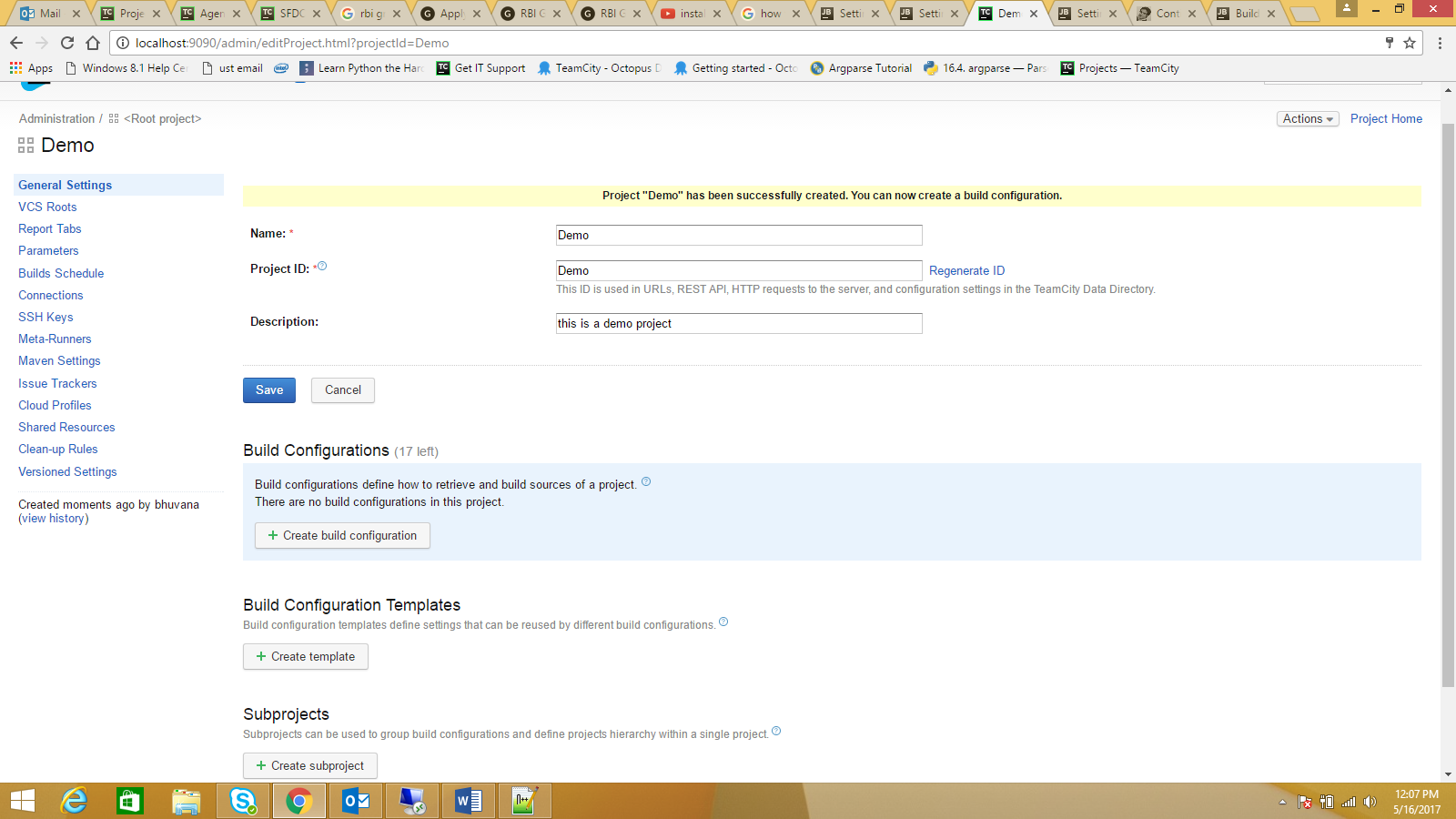
There is also a ‘Create Project’ button (if you are System Administrator) to create new projects: Please refer screenshot below.



After clicking on the ‘Create Project’ button you see the following page where you can enter your project details:

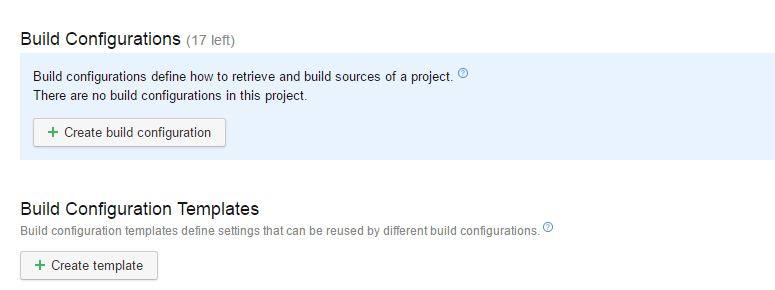


After you create the project, you are taken to the project home page where you can setup the build configurations.

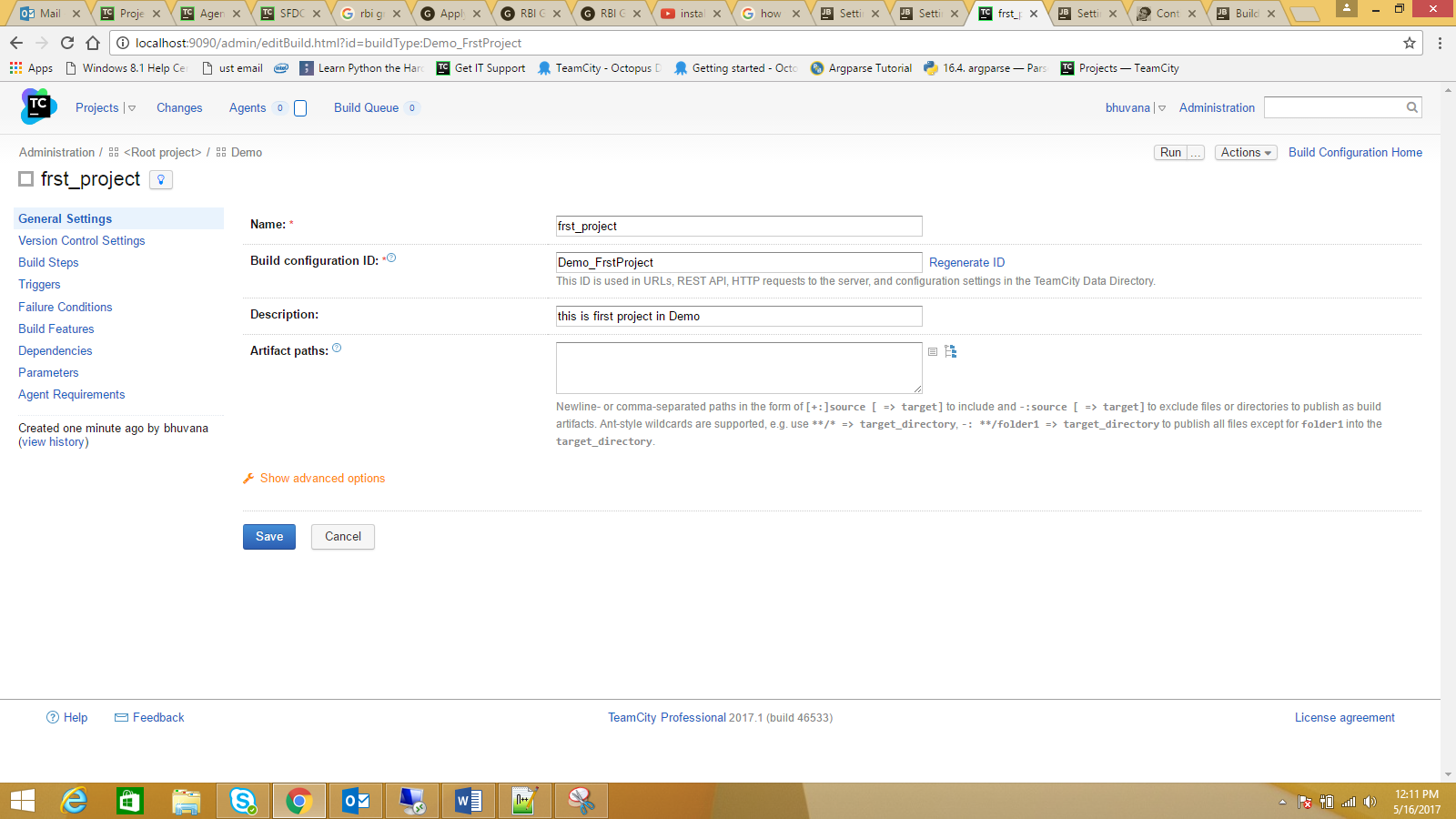


So you now have a TeamCity project and want to setup **Continuous Integration** for your project:

The first step is to create a build configuration for Continuous Integration. You can do so by clicking on the ‘Create build configuration’ button on the project home page:



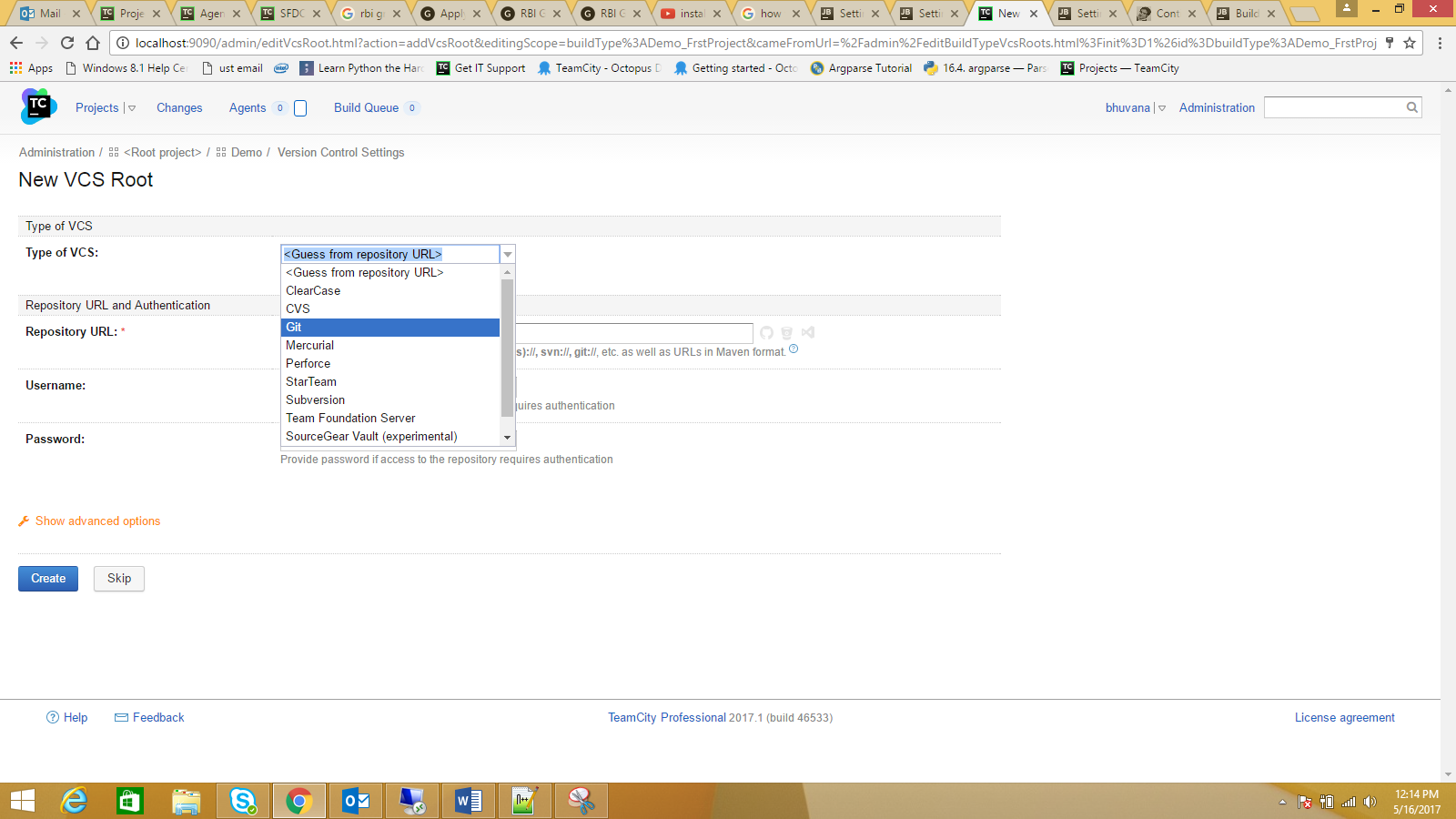
After clicking on the button you will be taken to the ‘Create build configuration’ page:



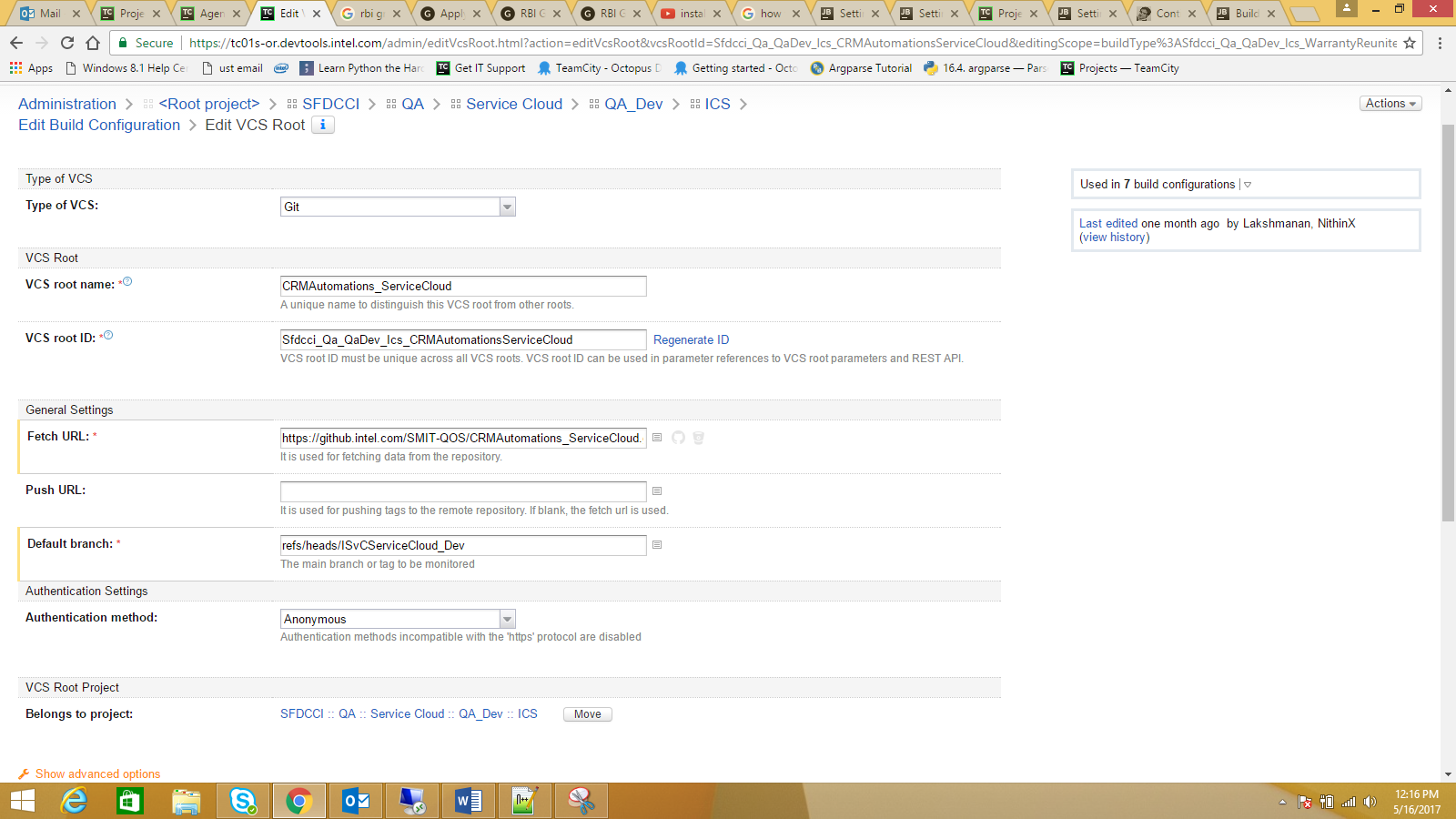
**Source control settings**

Once your build configuration is created you should set the ‘VCS settings’ so TeamCity knows how to get the code needed for the build.

Different VCS engines have different settings so the first step is to pick the VCS you want to use - in our case Git (for GitHub):



Once you choose Git you are provided with a page to setup your Git repository connection:



* **VCS root name and ID**: use a unique name preferably related to your project so it’s easy to spot it amongst other VCS roots you might have in your TeamCity.
* **Fetch URL**: this tells TeamCity where it should look for the source code. If you’re using GitHub you can grab this from your project’s GitHub home.
* **Default branch:** should be set to your master branch which is where your CI should focus more. So we set it to **refs/heads/master** which is the Git master branch.

Once you create a VCS root you can reuse it across build configurations

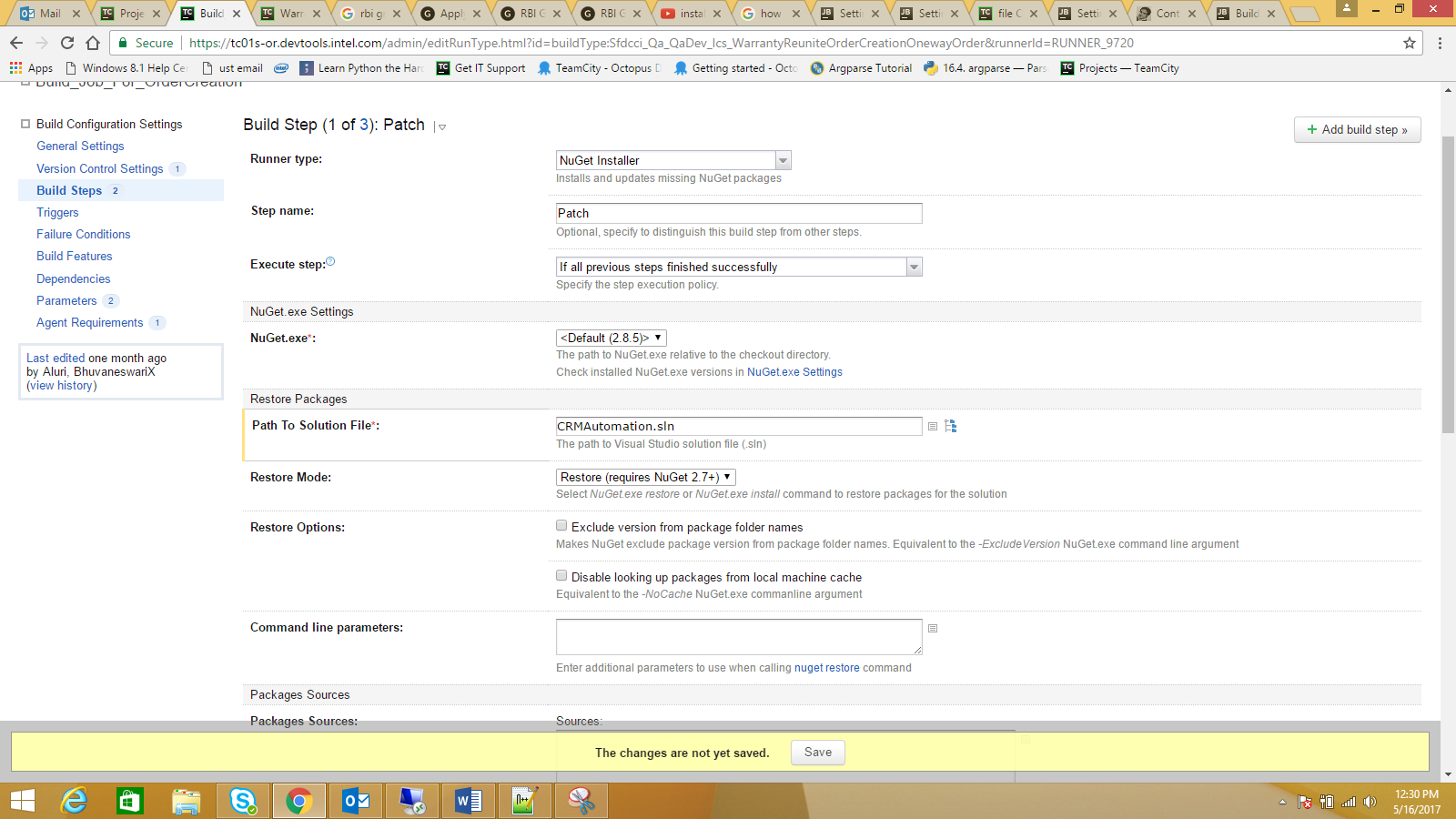
You have a build configuration attached to a source control. We can now create the build steps. I am going to create two steps:

1. NuGet installer
2. Build the code

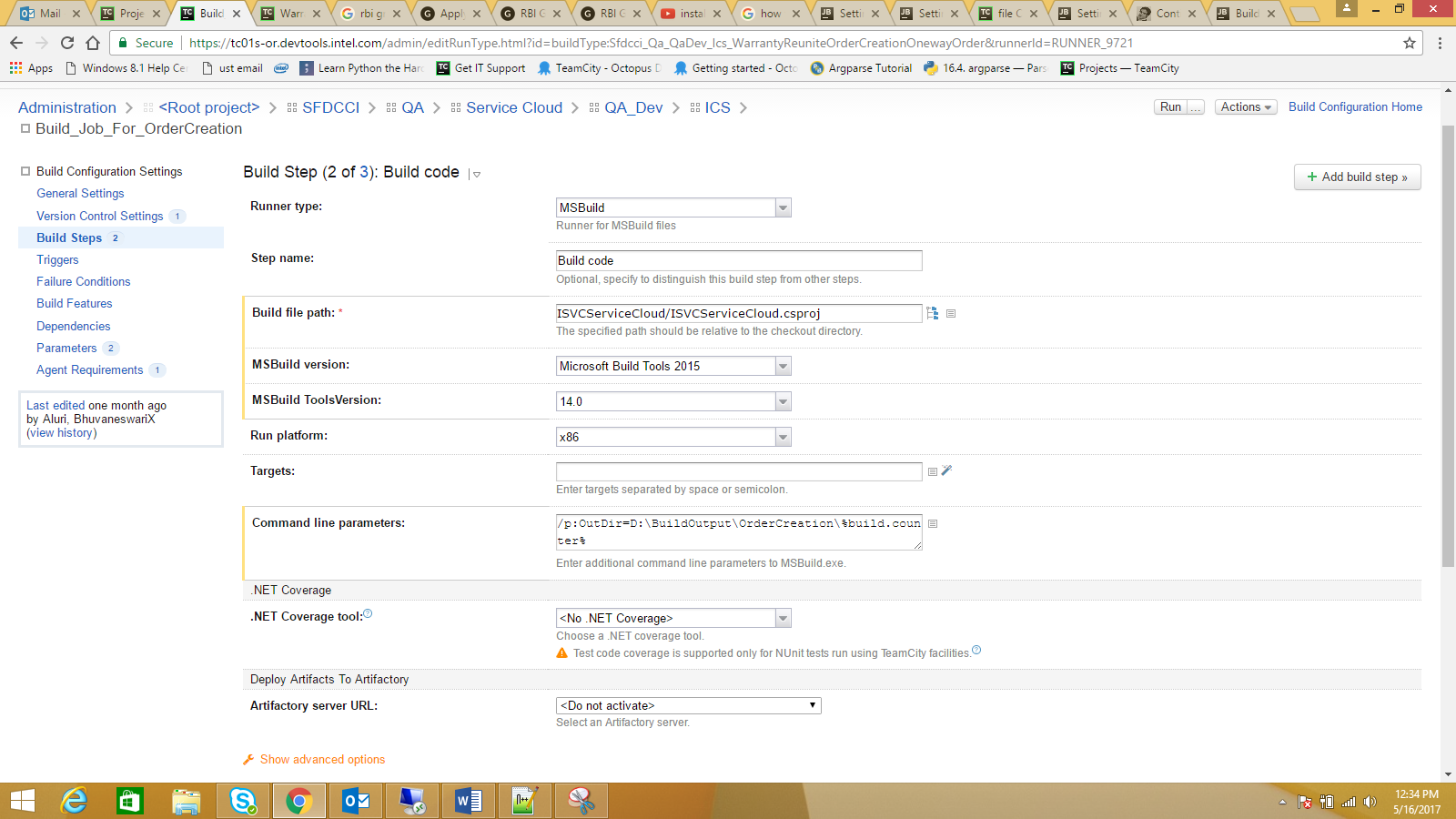
To create a build step you should click on the ‘Add build step’ button.

To install and updates the missing Nuget Packages, so here I will update NuGet.exe packages via project Soution file.

Hence, Runner type would be NuGet Installer. Please refer the screen shots below for reference.



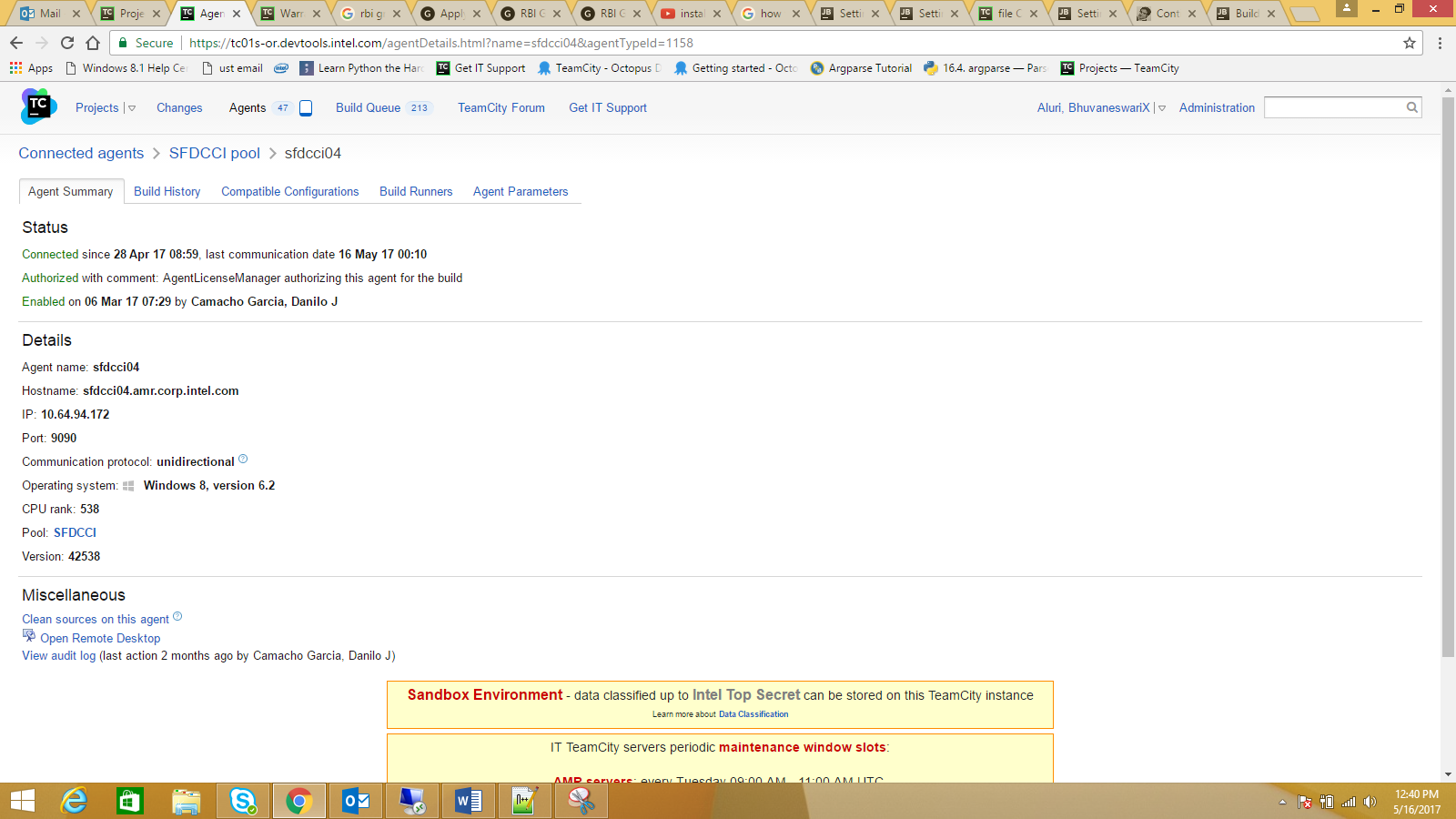
**Build Solution**: To build code, the easiest way is to build the project file. So I pick ‘MSBUILD’ for the ‘Runner type’:



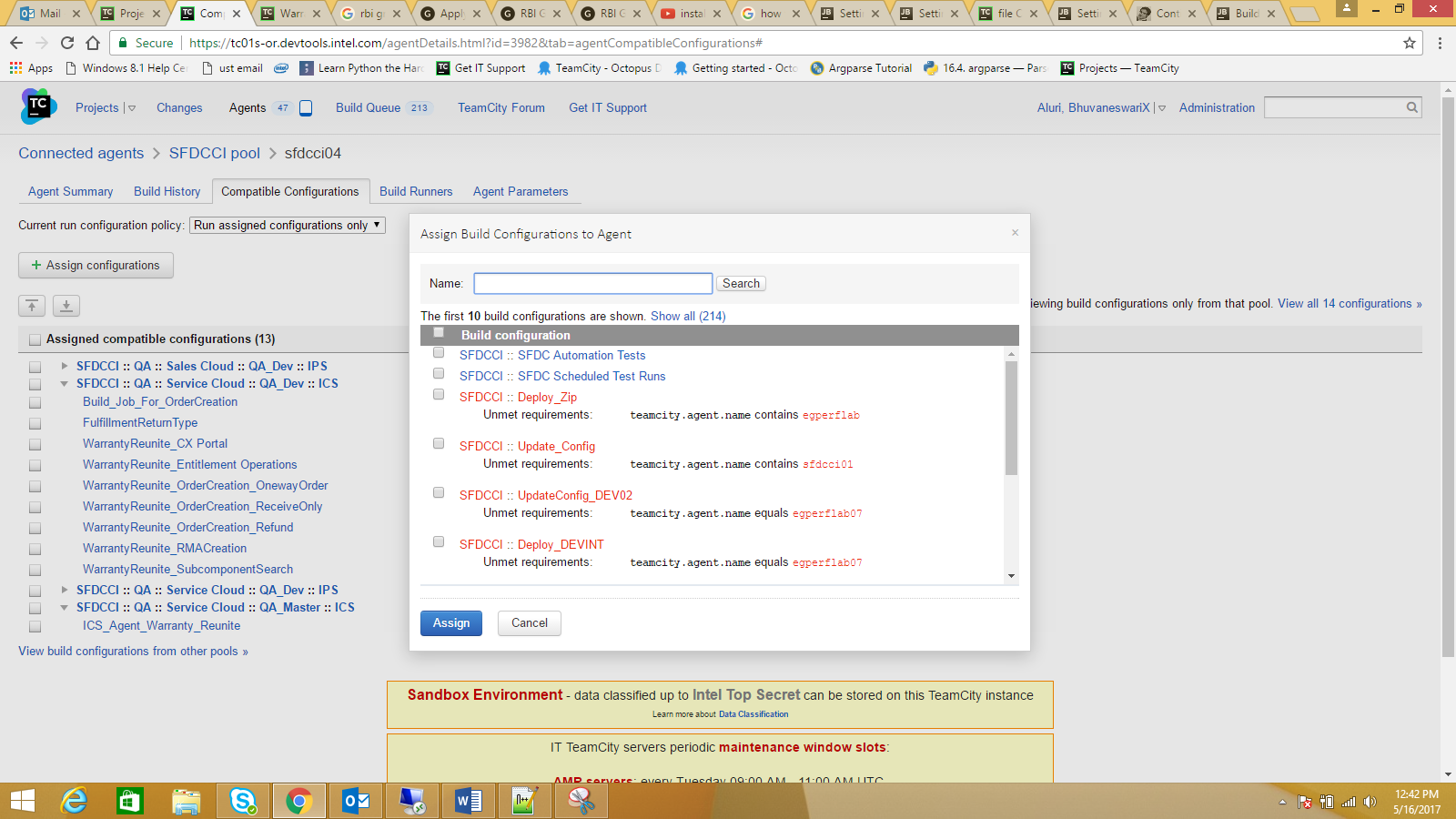
**Assign your build configuration to an agent:**

Once you have a build step to build your source code, it’s a good idea to run your build to see if it works. For that to work though you have to assign a build agent to your build. So I am going to jump ahead a bit and explain how you can assign your build configuration to a build agent.

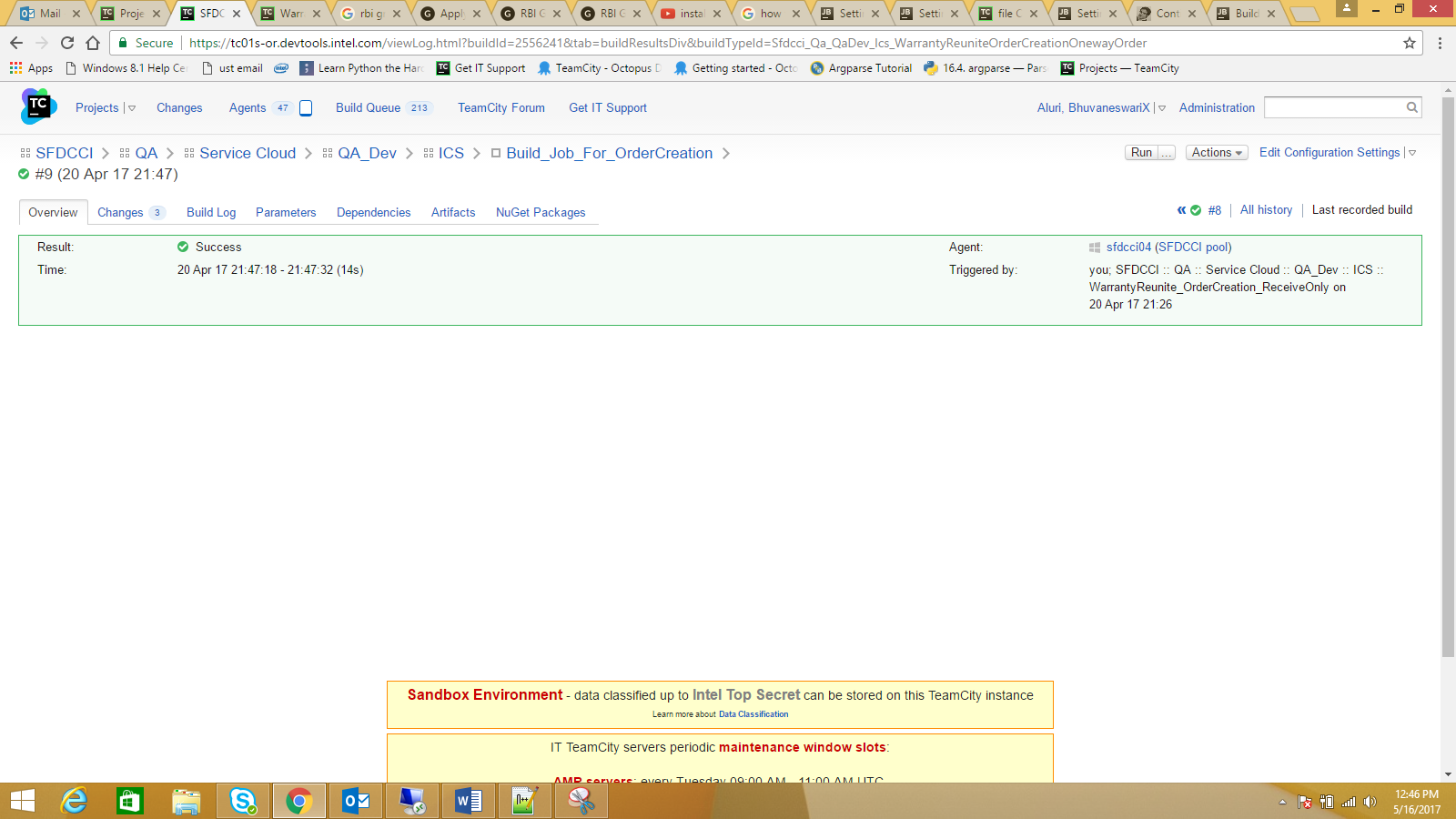
On the top navigation bar click on the ‘Agents’ link and then navigate to the build agent you want to use for your CI build configuration:



From there you go to the ‘Compatible Configurations’ pane where you can assign your new build configuration to your agent.



So now you have your build configuration with one build step and an agent assigned to it. Go forth and click run on your build. It should get the source from your repository and build it:



Make sure your build is green and it does what it should do: getting the latest code and building your solution. Check the ‘Changes’ and ‘Build Log’ panes.

**Build Trigger**:

 We now have a complete CI build configuration. There is still one problem though.

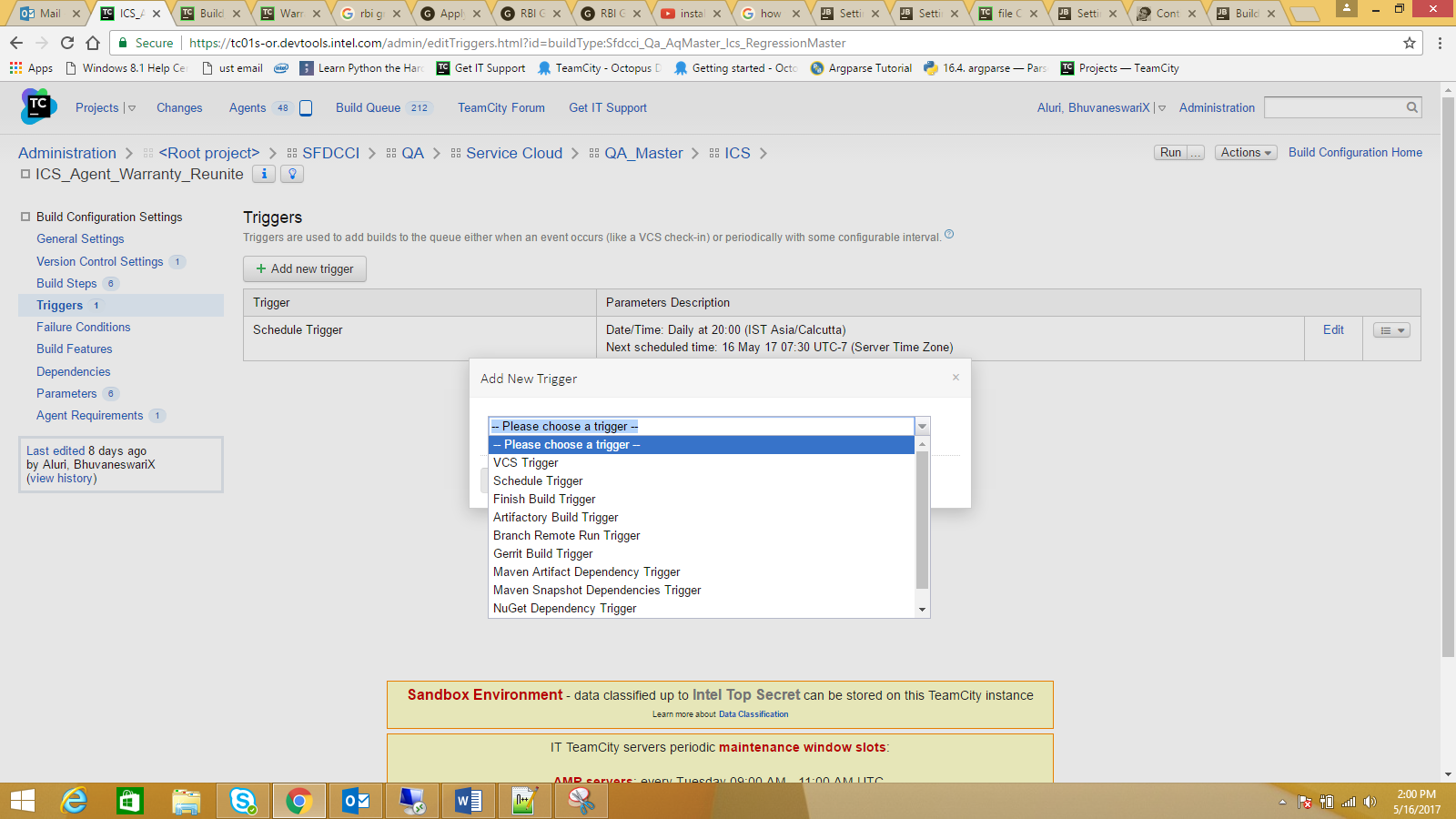
We have been running this build configuration manually! A CI setup should be able to detect changes on the source code and build them automatically.

You have many Triggers available in ‘Triggers Tab’. Here we will brief two triggers.

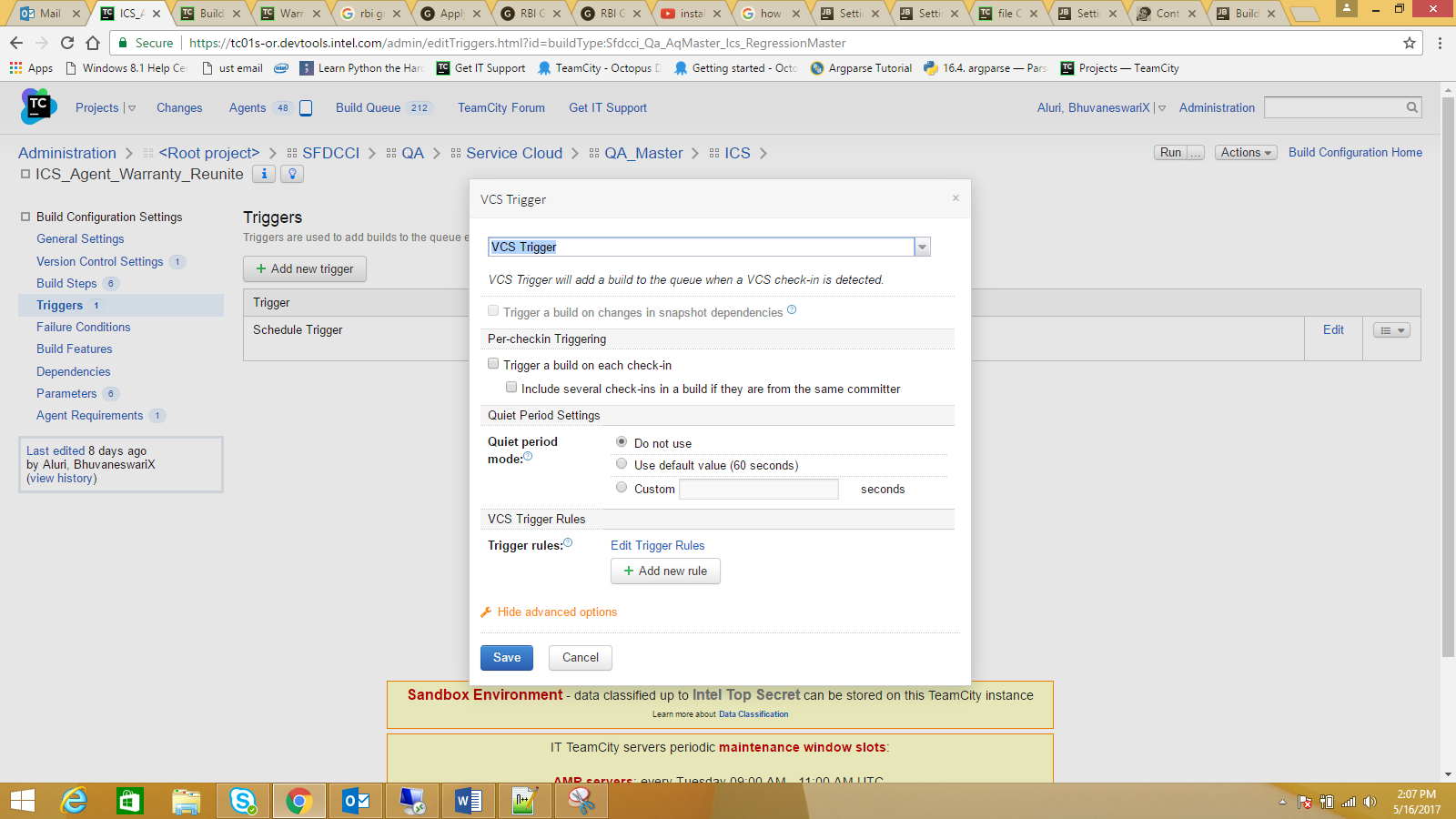
1. VCS trigger, which triggers a job automatically if it detects a VCS check in.
2. Schedule Trigger, which triggers a job during the scheduled time.

We can achieve that using ‘Build Triggers’. Again from the left navigation bar on your CI Build Configuration select ‘Build Trigger’ and click on ‘Add new trigger’ button to add a build trigger.

Below are the screenshot references on how to add and configure triggers for the build job.



VCS Trigger:



**Dependencies:**

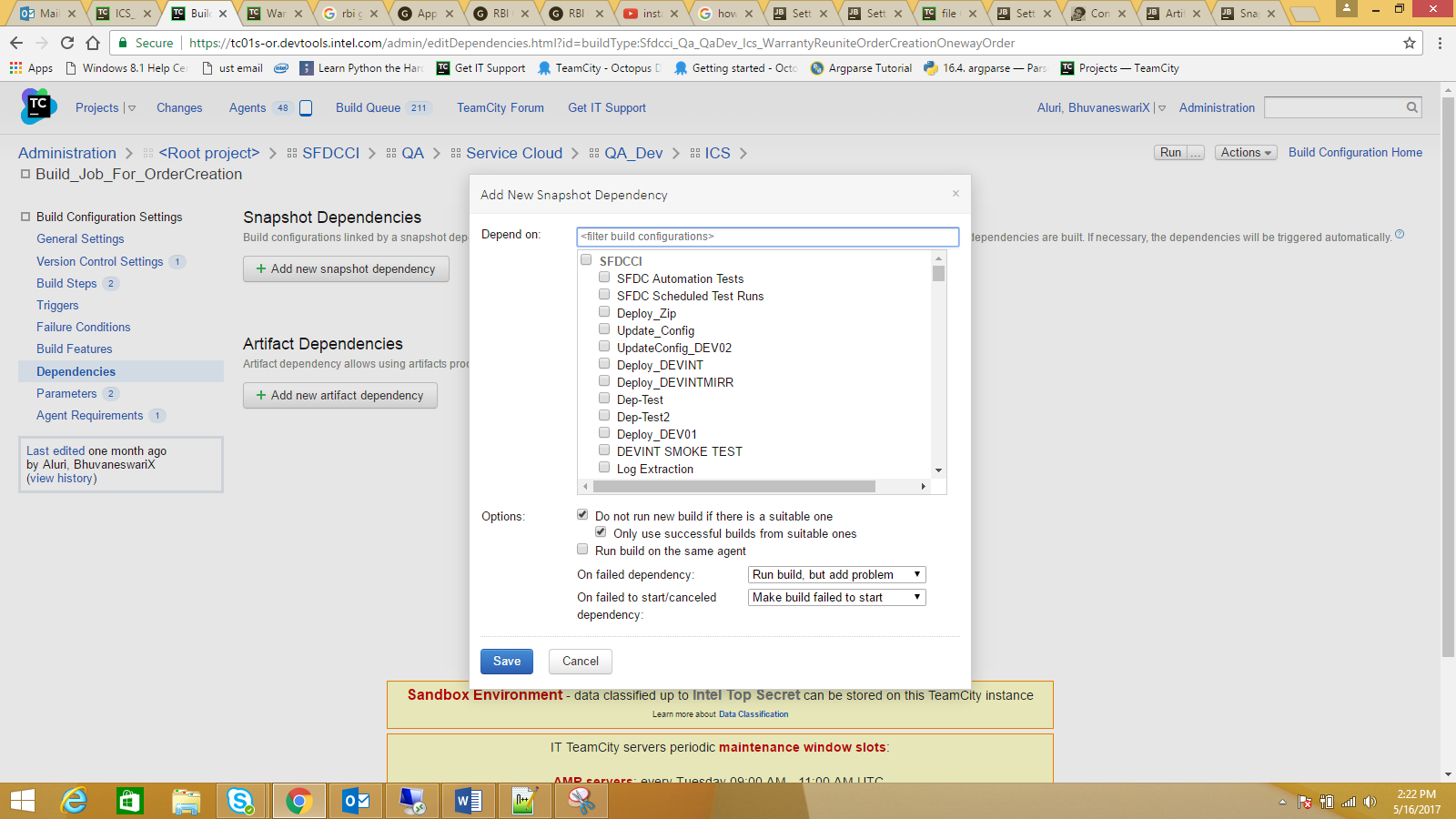
The idea of artifact dependencies in TeamCity is very simple: download the artifacts produced by another build before the current one begins. After the artifacts are downloaded to the folder specified (checkout directory by default), your build script can use them to achieve its goals.

In other words, setting the dependency by other build's sources you can ensure that a build will start only after the one it depends from is run and finished.

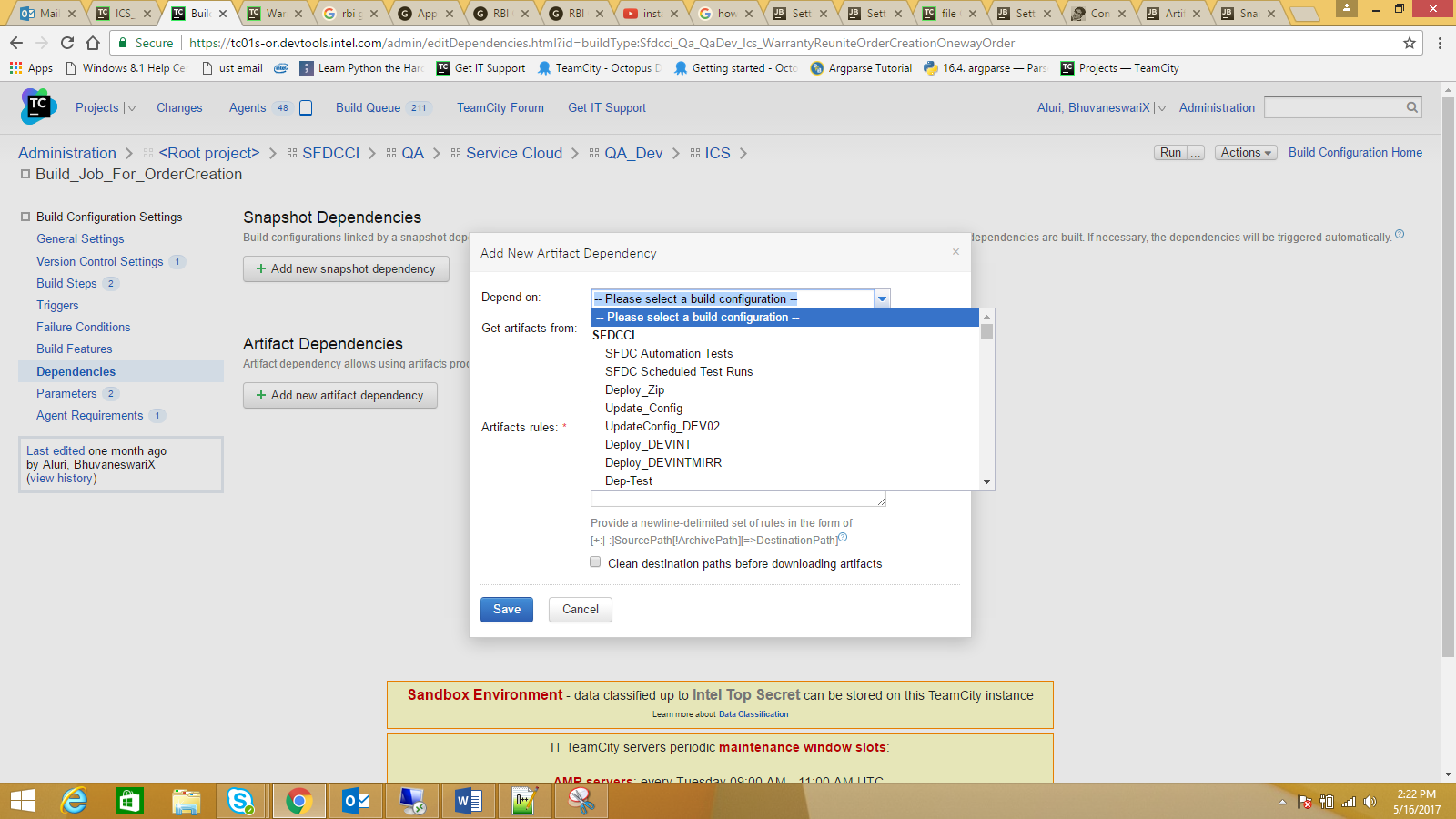
Below are the two dependencies available:

1. Snapshot Dependency, Build configurations linked by a snapshot dependency will use the same snapshot of the sources. The build of this configuration will run after all the dependencies are built. If necessary, the dependencies will be triggered automatically
2. Artifact Dependency, Artifact dependency allows using artifacts produced by another build.

Snapshot Dependency:



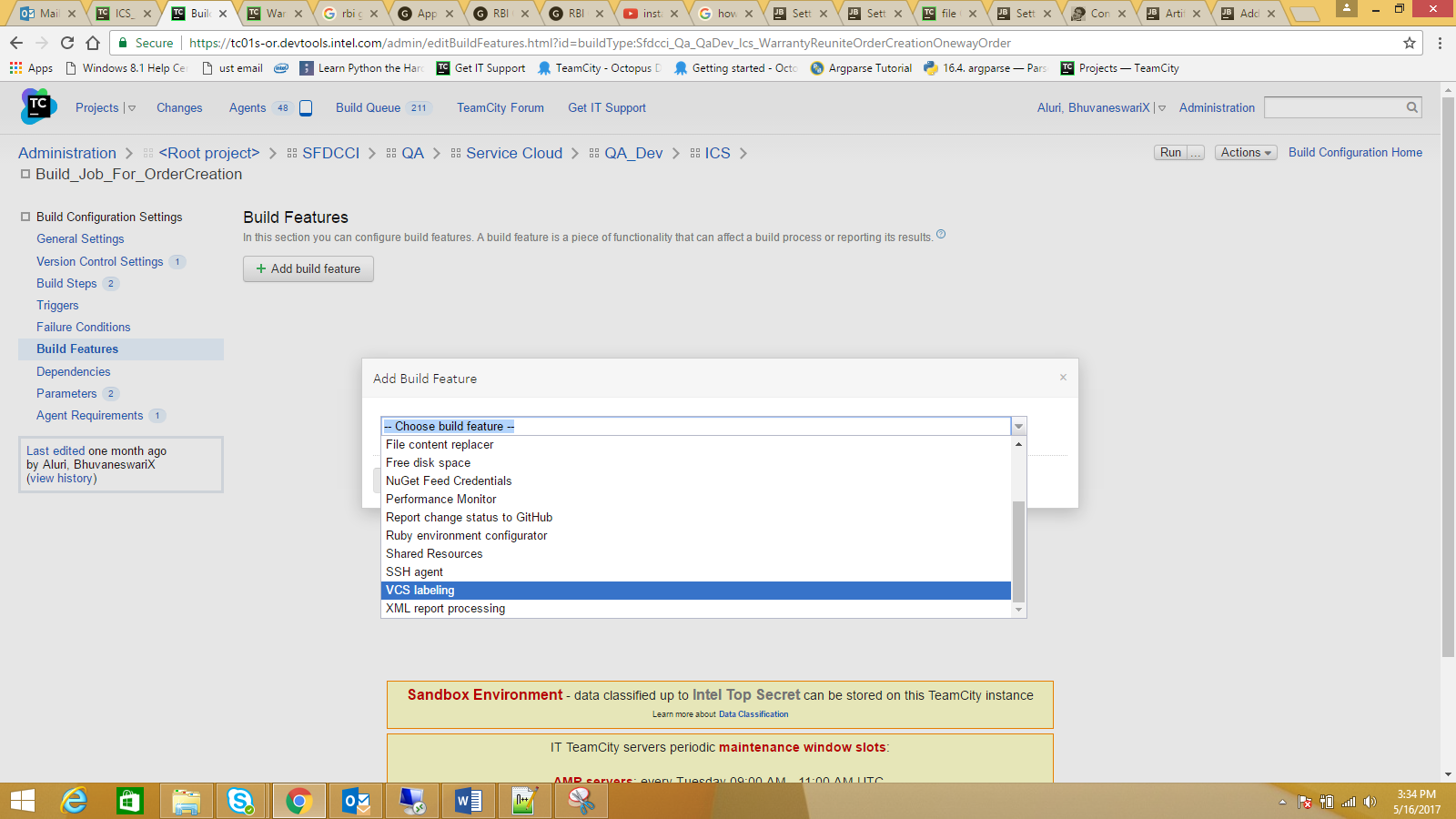
Artifact Dependency:



**Build Feature:**

A "build feature" is a piece of functionality that can be added to a build configuration to affect running builds or reporting build results.

Build features are configured on the dedicated page of the **Build Configuration Settings** available from the list on the left.



**Email Notification for Builds:**

TeamCity provides a wide range of notification possibilities to keep developers informed about the status of their projects. Notifications can be sent by e-mail.

Notifications in TeamCity are sent per-user according to the user's configured notification rules.

We can configure notifications in two ways.

1. Using, Team city internal notification.
2. Customized way (using script).

**Subscribing to Notifications in Team city:**

TeamCity allows you to flexibly adjust the notification rules, so that you receive notifications only on the events you are interested in. To subscribe to notifications:

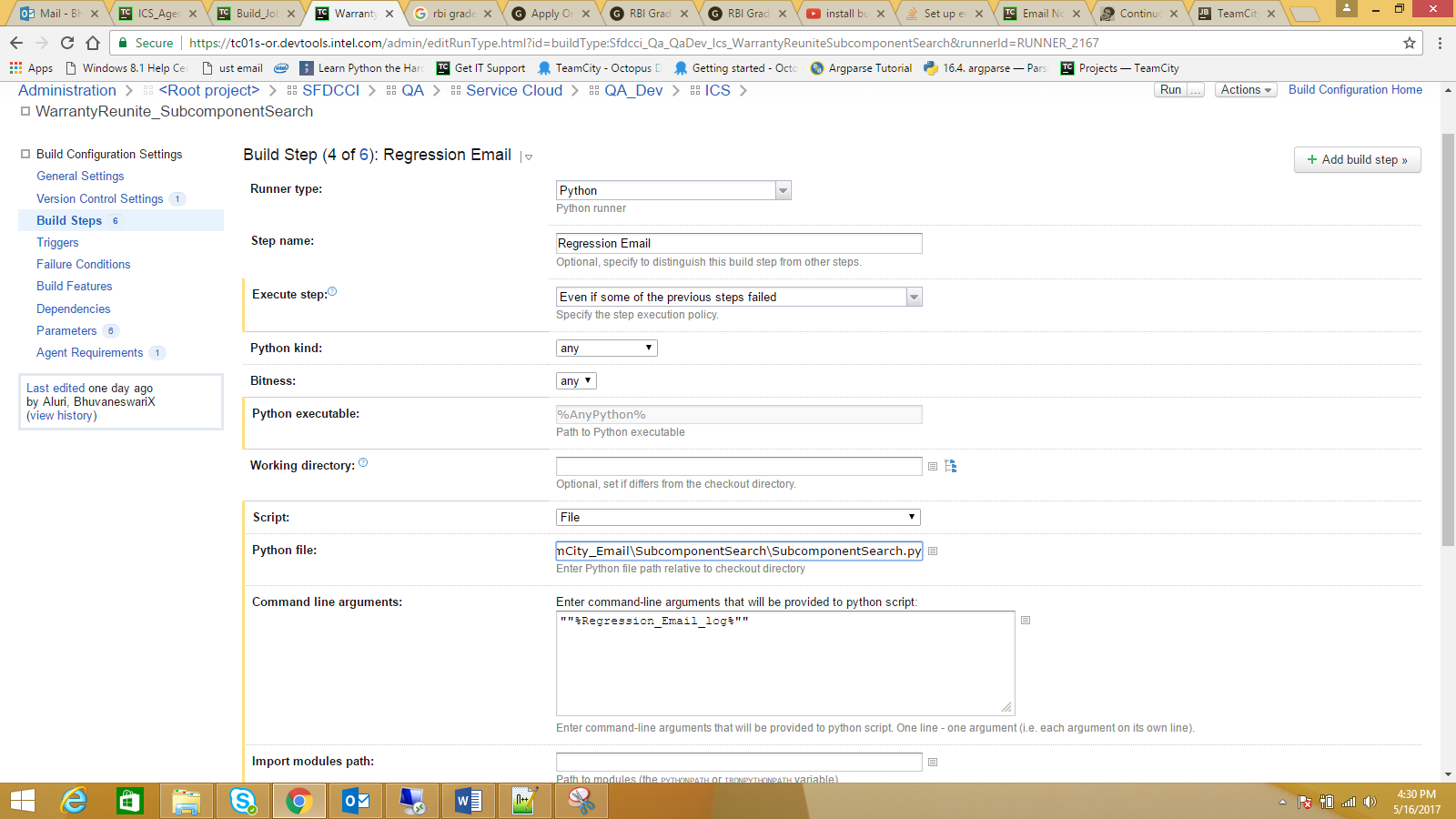
In the top right corner of the screen, click the arrow next to your username, and select **My Settings & Tools** from the drop-down list. Open the **Notification Rules** tab.

Click the required notifications type:

**Email Notifier**: to be able to receive email notifications, your email address must be specified in the **General** area on the **My Settings & Tools** page.

**Customized Way:**

Here, we are using python script to configure mail notifications and calling that python file in the build configurations of job in Team city.



**Build Parameters:**

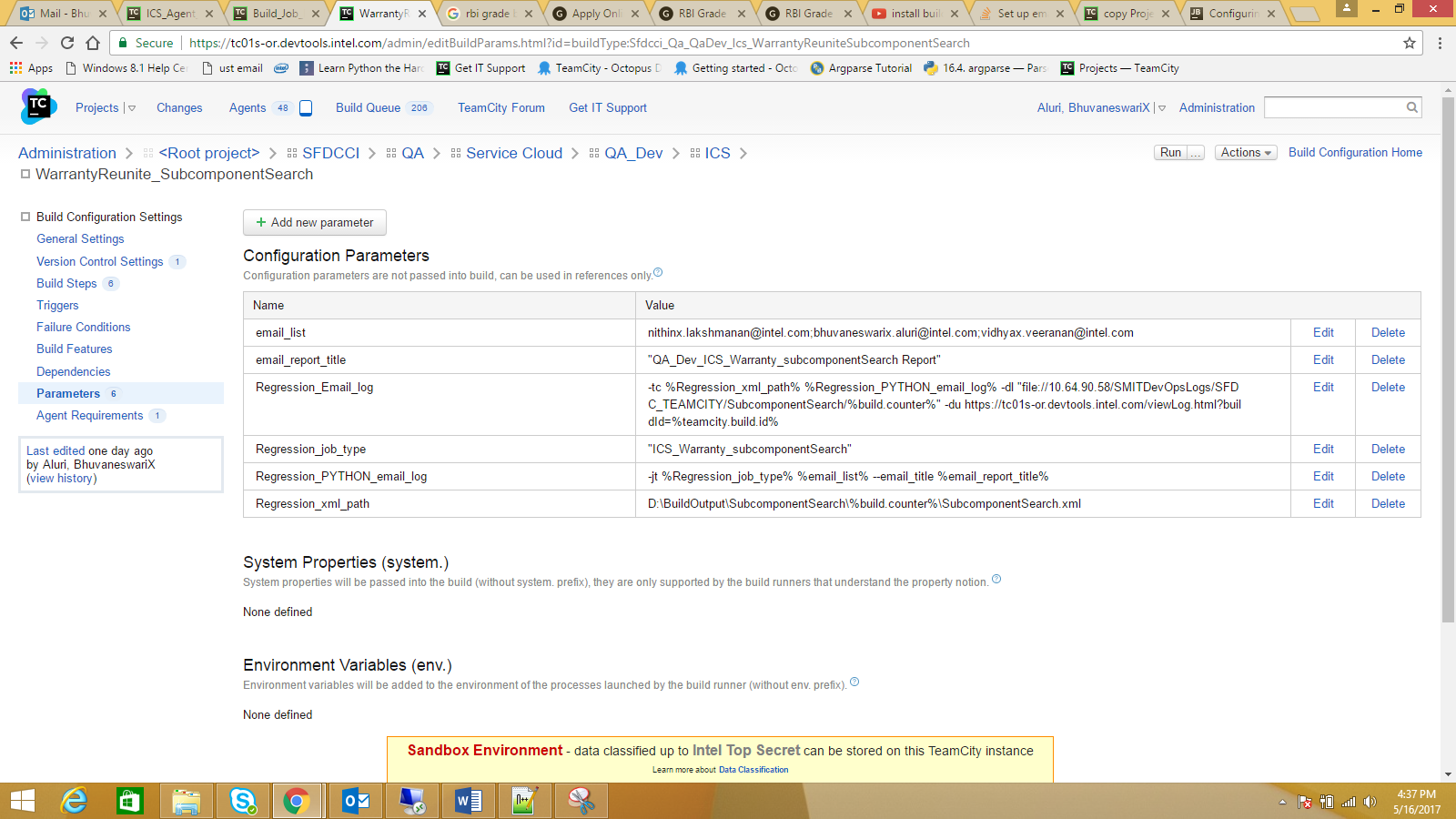
Build Parameters provide you with flexible means of sharing settings and a convenient way of passing settings into the build.

Build parameters are name-value pairs, defined by a user or provided by TeamCity, which can be used in a build.

There are three types of build parameters:

* **Environment properties** (defined using "env." prefix) are passed into build process as environment
* **System properties** (defined using "system." prefix) are passed into the build scripts of the supported runners (e.g. Ant, MSBuild) as build-tool specific variables
* **Configuration parameters** (no prefix) are not passed into the build and are only meant to share settings within a build configuration (can be used as a references only).

Configuration Parameter:



**System and Environment parameters:**

