Hands-on Lab

Lab 1: Creating the Trey Research Environment

Table of Contents

Objectives

In this lab you install the initial version of the Trey Research application and create the initial version of the Trey Research release pipeline. You can read about this version of the pipeline in Chapter 2 of Building a Release Pipeline with Team Foundation Server 2012.

During this hands-on lab, you will:

- Install the components that comprise the Trey Research application.
- Learn how the Windows Presentation Foundation (WPF) and Windows Communication Foundation (WCF) service work together.
- Install the test artifacts. Some of these artifacts will be used in later labs.
- Use the TFS default build template to construct a simple release pipeline that supports continuous integration.

The Trey Research application is located in the TreyResearch folder.

Prerequisites

The prerequisites for completing this lab is that you have already installed the components listed in the Introduction.

This lab assumes that you are using the Brian Keller virtual machine (VM).

Time

You should be able to complete this lab in approximately 40 minutes.

Exercise 1: Setting Up the Brian Keller VM

The following is what we learned working with the VM. The labs were done under the Administrator account.

> **Note:** Important – activate Windows per the Brian Keller document Working with the Visual Studio 2012 Update 2 RTM ALM Virtual Machine. This will eliminate issues dealing with configurations not resetting.

Task 1: Preloading the Software

Because the VM is network isolated, it's better to load all the software you'll use before you begin the labs. Here's the list of what you'll need and the links.

1. The Community TFS Build Extensions
2. Web Deploy v.3.0
3. Wix Toolset

Task 2: Changing the VM Time

The VM clock is set to early time. Set the VM to your current date and time.

> **Note: Working with Date & Time – from Brian Keller's "Working with VS 2012 RTM VM."**
>
> This virtual machine has been hard-coded to boot up with a system date of May 16, 2012. This is required in order to support its accompanying hands-on-labs and demo scripts. Synchronization with the host operating system is disabled, as is synchronization with Internet time servers. If you reboot this virtual machine after you begin working with data in Team Foundation Server, it may have

unintended consequences. Therefore it is recommended that you only reboot during the initial configuration.

Task 3: Understanding the TFS Team Collection Setup

TFS on the Brian Keller VM has no default collection. Instead, use the **FabrikamCollection**. The build controller and test controller are set to this collection.

Task 4: Overwrite the drivers\host File

Overwrite the c:\windows\system32\etc\drivers\hosts file with the blank file that is available at C:\util\blankhosts.

Task 5: Enable Time Synchronization

1. Use the Hyper-V manager to enable time synchronization for the VM.
2. Disable the scheduled task that sets the date on startup.

Task 6: Solving Issues with Version Mismatches

You may get errors in the event logs such as:

Unable to connect to the controller on 'vsalm:6901'. Cannot communicate with the Controller due to version mismatch.

If this occurs, update the Visual Studio agents at Agents for Visual Studio 2012 Update 3.

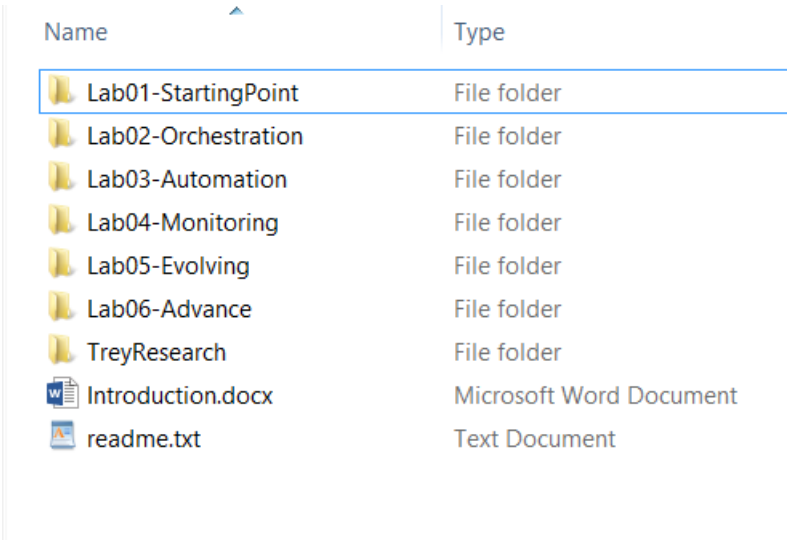
Exercise 2: Installing the Trey Research Application

In this exercise you install the Trey Research application, deploy the Windows Communication Foundation (WCF) service, and deploy the Windows Presentation Foundation (WPF) service.

Task 1: Install the Trey Research Solution

In this task you create a TFS team project, map it to a local directory, and populate the directory with the Trey Research solution files.

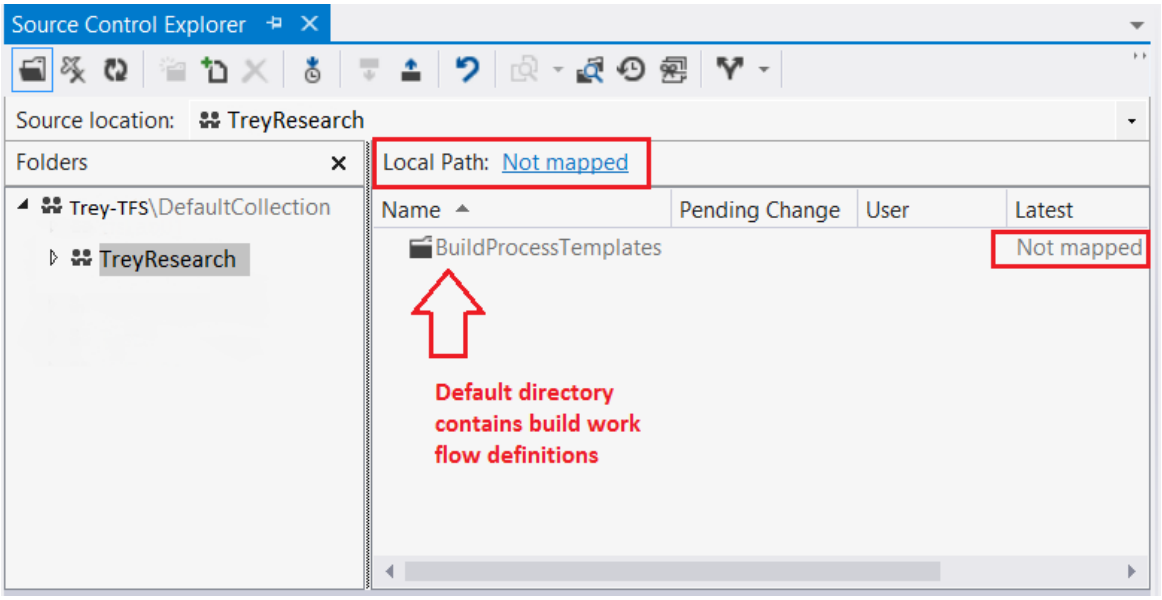
1. Create a working directory where all the lab work and projects will be placed. In these labs the default directory is **C:\HOL**.
2. Unzip the ReleasePipeline_HOL.zip file to a working directory. It should contain 6 lab folders and an Introduction.docx file. The following screenshot illustrates this.



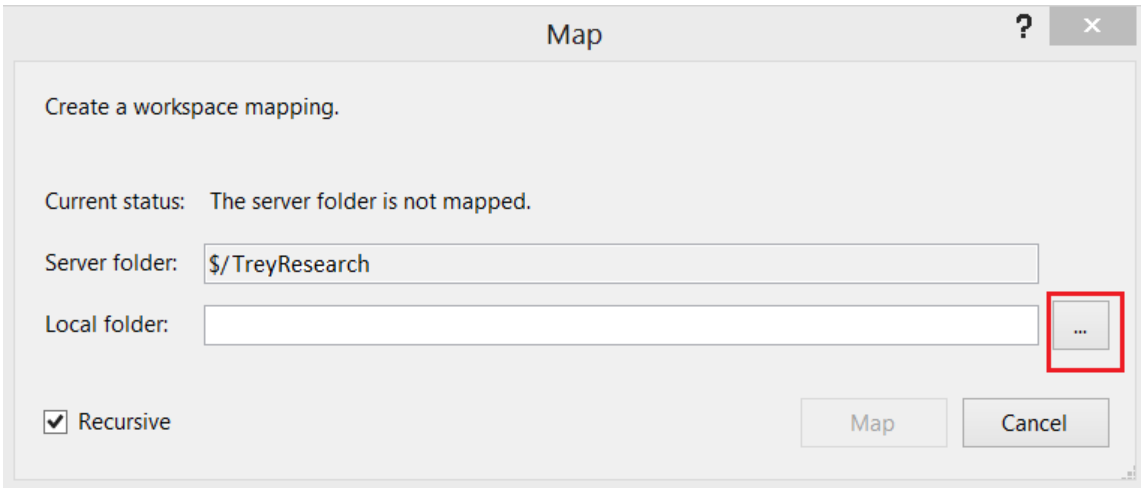
Task 2: Create the TreyResearch Team Project

In this task you create the TreyResearch team project in TFS. You'll add the code to this team project.

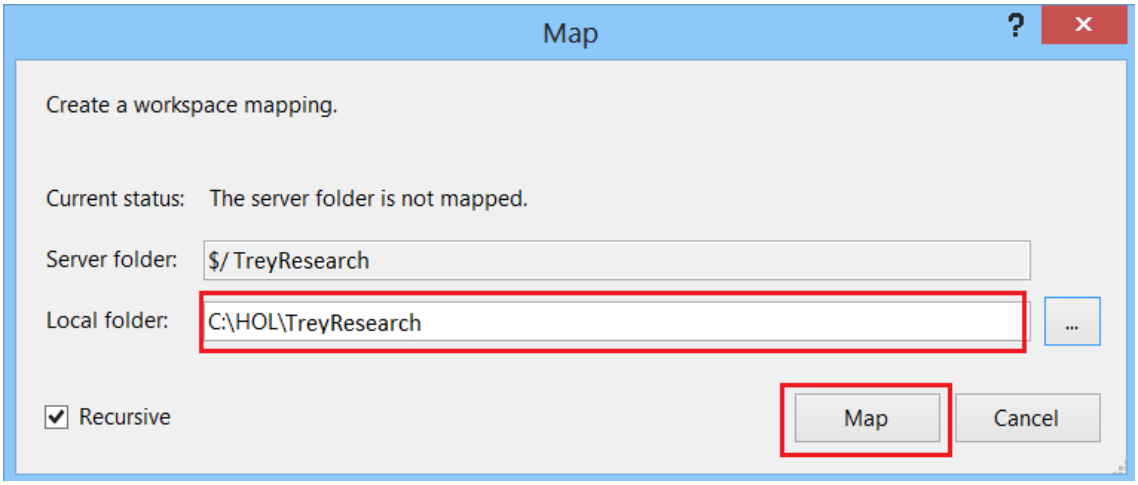
1. Open Microsoft Visual Studio.
2. Click **File**. Point to **New**. Select **Team Project**. Create a new team project and name it **TreyResearch**. Click **Next**.
3. Select the **MSF for Agile Software Development 6.2** process template.
4. Accept all the default option. Click **Finish**.
5. Select **Source Control Explorer**. Select the **TreyResearch** project. The default team project has a single **BuildProcessTemplates** folder. The Local Path is not mapped, as is shown in the following screenshot.



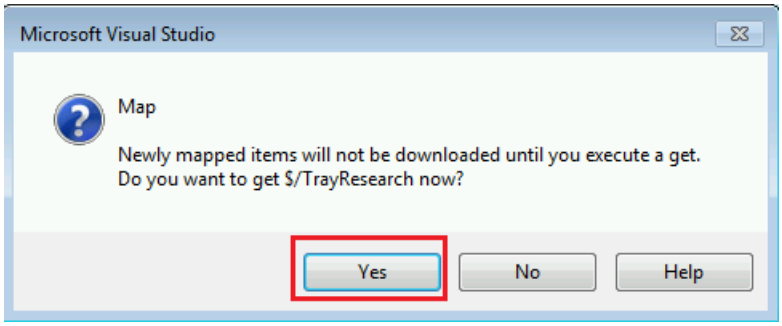
6. Click **Not mapped**. The **Map** dialog box appears.



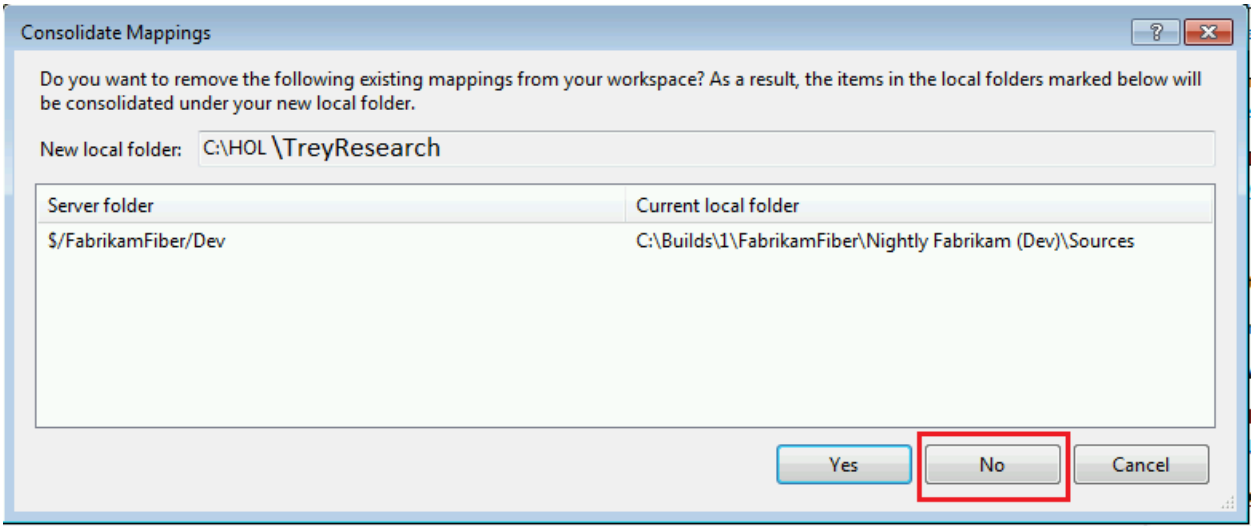
7. Click the **ellipses (…)** and navigate to the local directory where you want to map to the team project. Click **Map**.



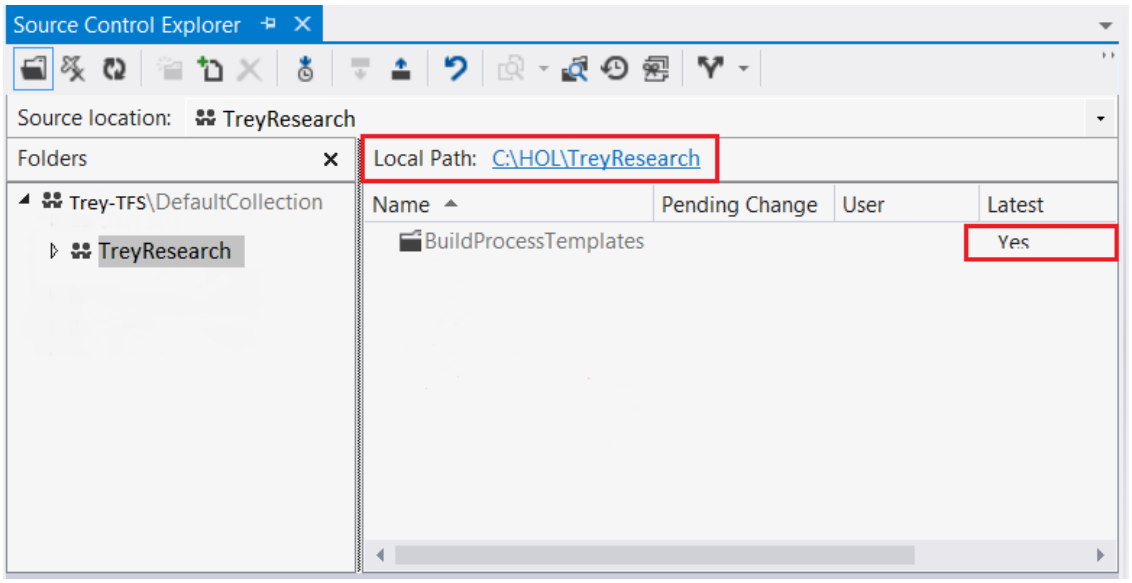
8. The Map dialog box appears. It asks if you want to get $/TreyResearch now? Click **Yes.**



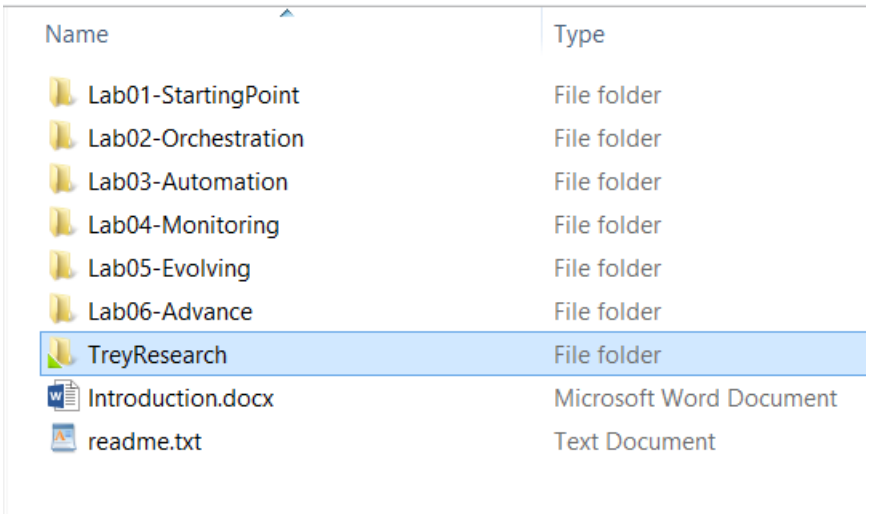
9. On the Brian Keller VM, a **Consolidate Mappings** dialog box appears. Select **No**.



10. The Solution Explorer view should now look like the following screenshot.



11. The local work space should look like the following screenshot. Note the green triangle on the TreyResearch folder. The green triangle means that the folder is now mapped to a TFS team project.



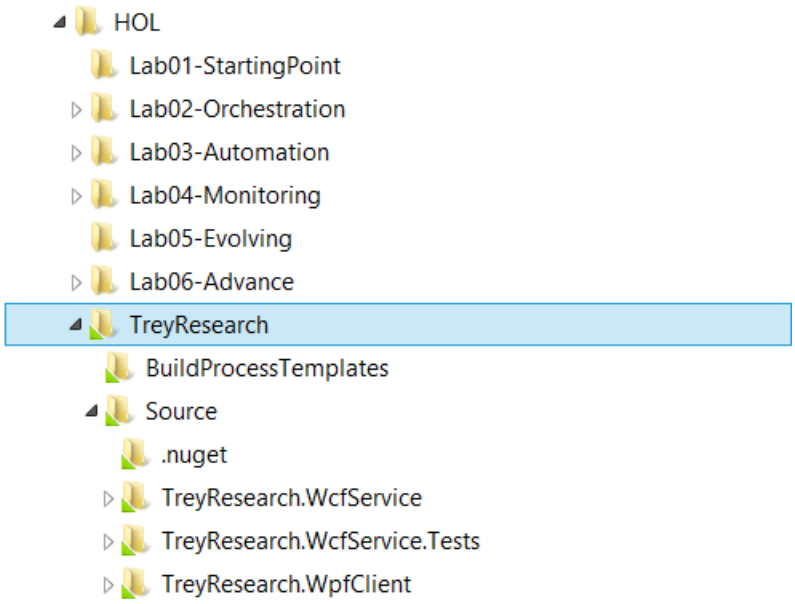
Task 3: Add the Trey Research Solution to the TreyResearch Team Project

In this task you add the Trey Research solution to the TreyResearch team project.

1. In Solution Control Explorer, navigate to the team project, **TreyResearch**.
2. Right-click on the **TreyResearch** folder and select **Add Items to Folder**. The **Add to Source Control** dialog appears.
3. Select the **Source** file folder in **TreyResearch** folder and click **Finish**. These files will be added to the project.
4. Check in the files. Right-click the **Source** file folder and select **Check In Pending Changes**.
5. In Team Explorer the **Pending Changes** dialog box appears. Click **Check In**. The check-in confirmation message appears. Click **Ok**.

> **Note:** The confirmation message doesn't always appear.

6. The TreyResearch solution is now a part of the team project. The following screenshot shows the file hierarchy. The green triangles mean that the files are under source control.



Task 4: Build the Trey Research Solution

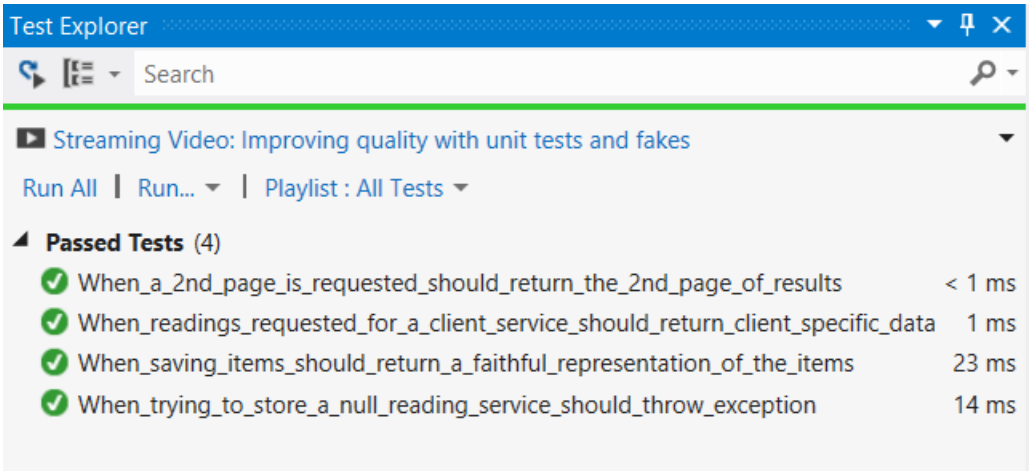
In this task you compile the Trey Research solution and test it. Because the lab use NuGet, make sure that it is enabled.

1. In Visual Studio open **TreyResearch.sln**.
2. Click the **Tools** menu. Click **Options**. Click the **Package Manager** tab.
3. Make sure that **Allow NuGet to download missing packages during build** is selected.
4. On the menu bar, click **Build**. Click **Build Solution**. The solution should build successfully.

Task 5: Run the Trey Research Unit Tests

In this task you run the unit tests that are associated with the Trey Research solution.

1. Open **Test Explorer**. On the menu bar, click **Test**. Point to **Windows**. Click **Test Explorer**.
2. Click **Run All**. Visual Studio builds the solution and runs the unit tests. After they complete, the unit tests should be listed under **Passed Tests**.



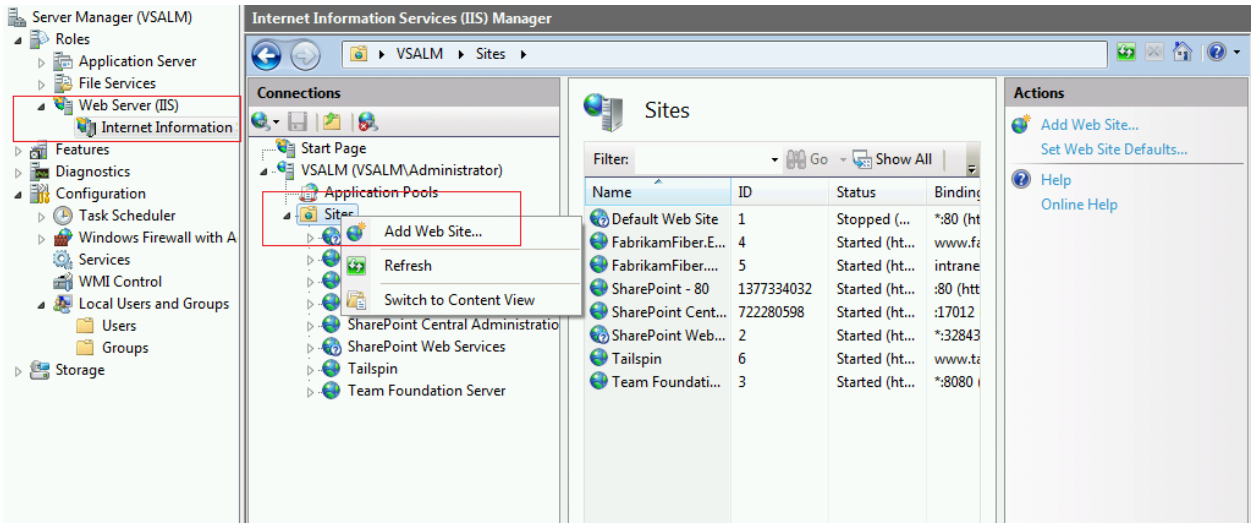
Exercise 3: Publishing the Trey Research Solution

In this exercise you publish the Trey Research WCF service. The Trey Research application uses WCF to pass sensor data to and from the mobile client

Task 1: Creating the TreyResearch Web Site

This exercise requires an open IIS port. On the Brian Keller VM, SharePoint uses port 80. This means that you must create a website on a different port. In this task you create a website on port 9100.

1. Open **Internet Information Services (IIS) Manager**. Right-click the **Sites** folder.

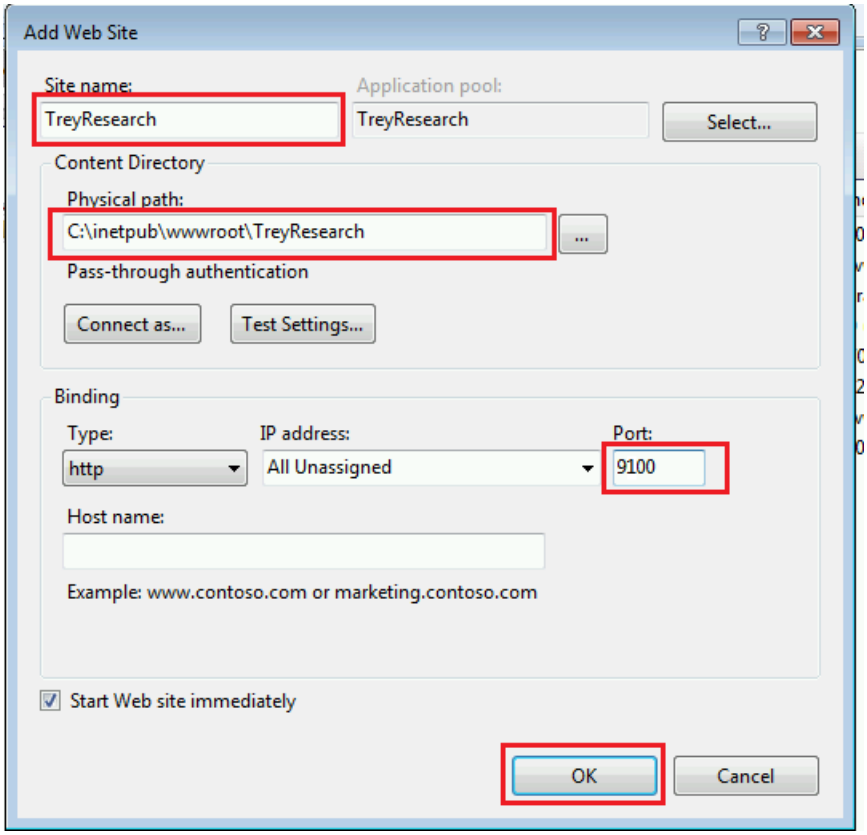


2. Click **Add Web Site**. The **Add Web Site** dialog box opens.
    a. In the **Site name** text box, enter **TreyResearch**. This will automatically create an application pool named TreyResearch.

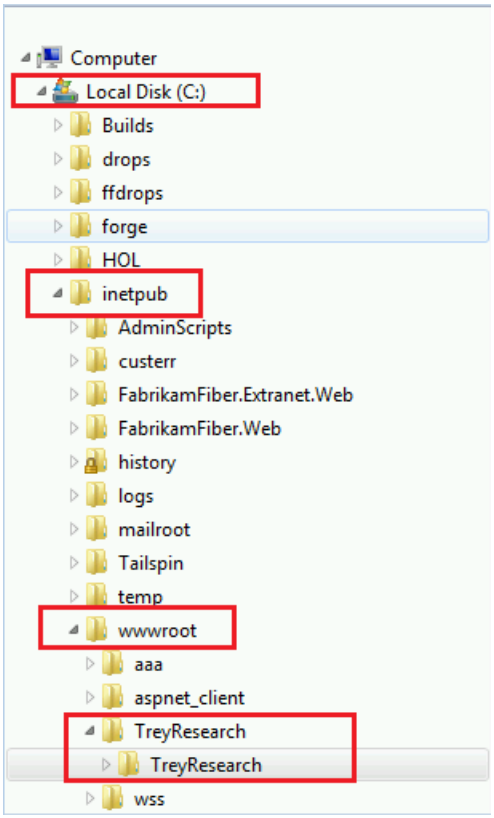
b. In the **Physical path** text box, enter **C:\inetpub\wwwroot\TreyResearch**.

c. Set **Port** to **9100**.

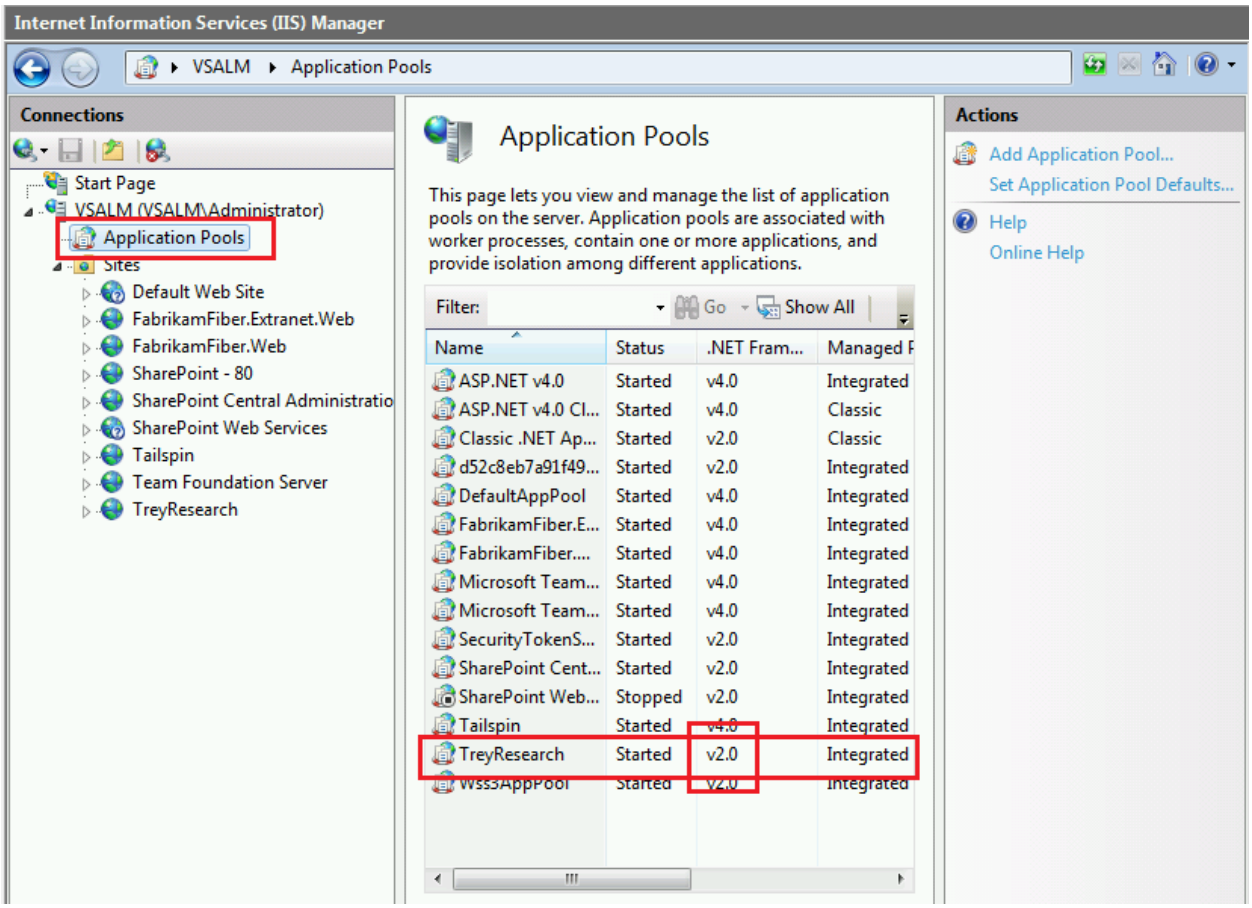
d. Click **Ok**. The following screenshot shows the completed dialog box.



3. To be compatible with the TreyResearch service URL, add another folder named **TreyResearch** under **C:\intentpub\wwwroot\TreyResearch**. The following screenshot shows with the file hierarchy should look like.

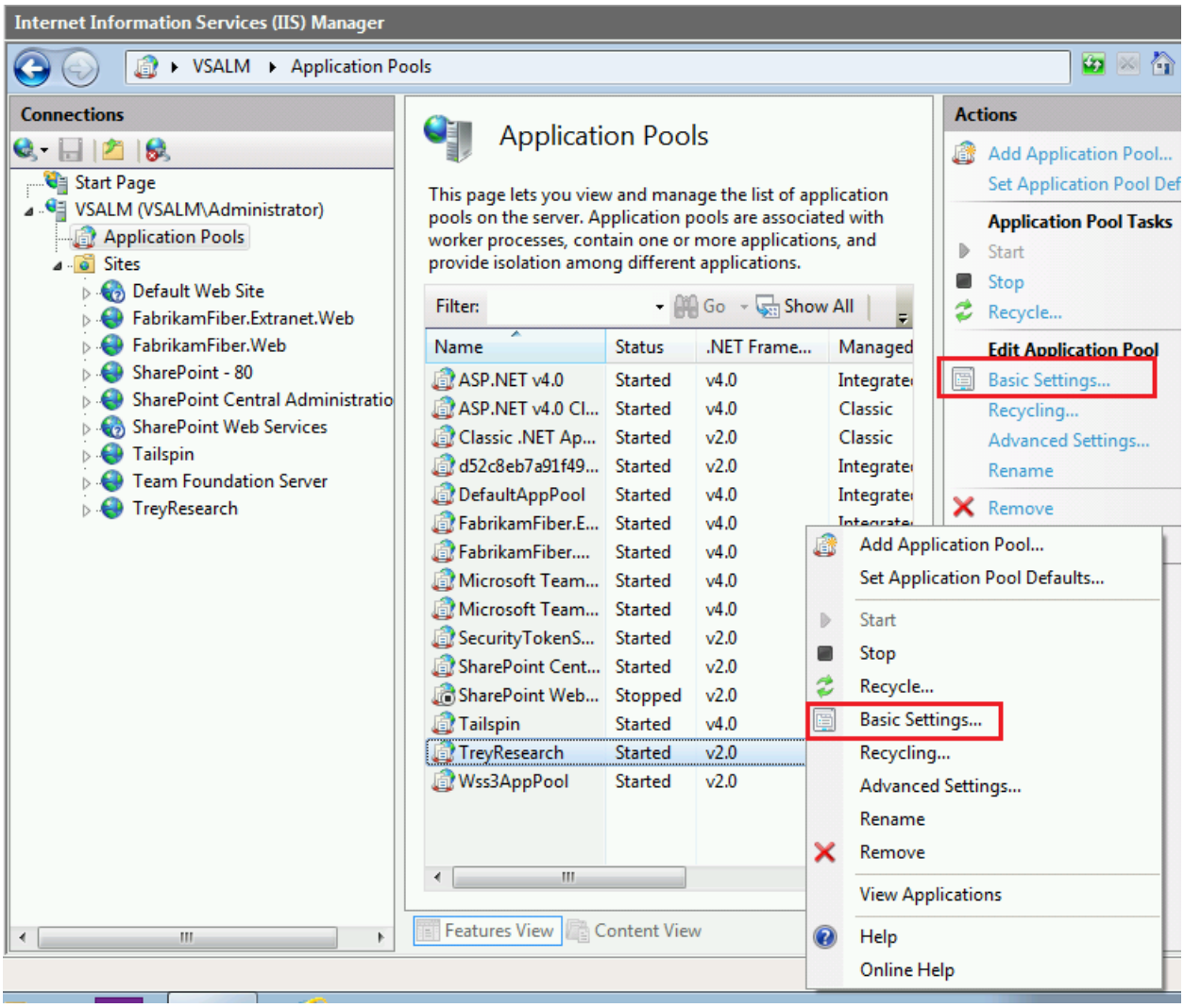


4. Check to see which version of the .NET Framework is running in the TreyResearch application pool. In IIS Manager, click the **Application Pools** node, which is above the **Sites** folder. Locate **Trey Research** in the Application Pools list.



5. If TreyResearch is using version 2.0 you must change it to v4.0. If it's already set to v4.0 you can go on to Task 2.

6. Select **TreyResearch** from the Application Pools list. Right-click on it. Select **Basic Settings**.

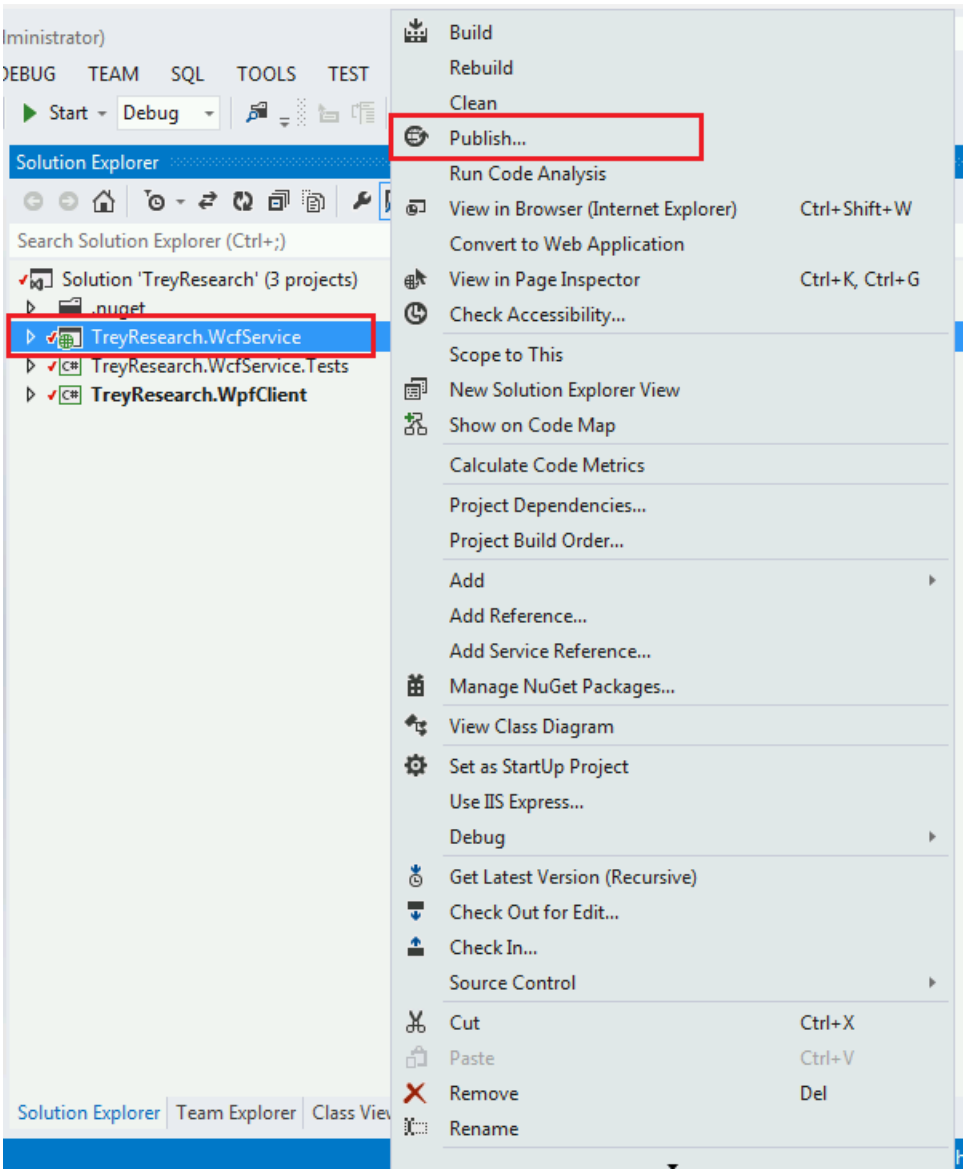


7. Select the .**NET Frame version** combo box. Select **.NET Framework v4.0.30319**. Click **OK**.
8. Close IIS Manager.

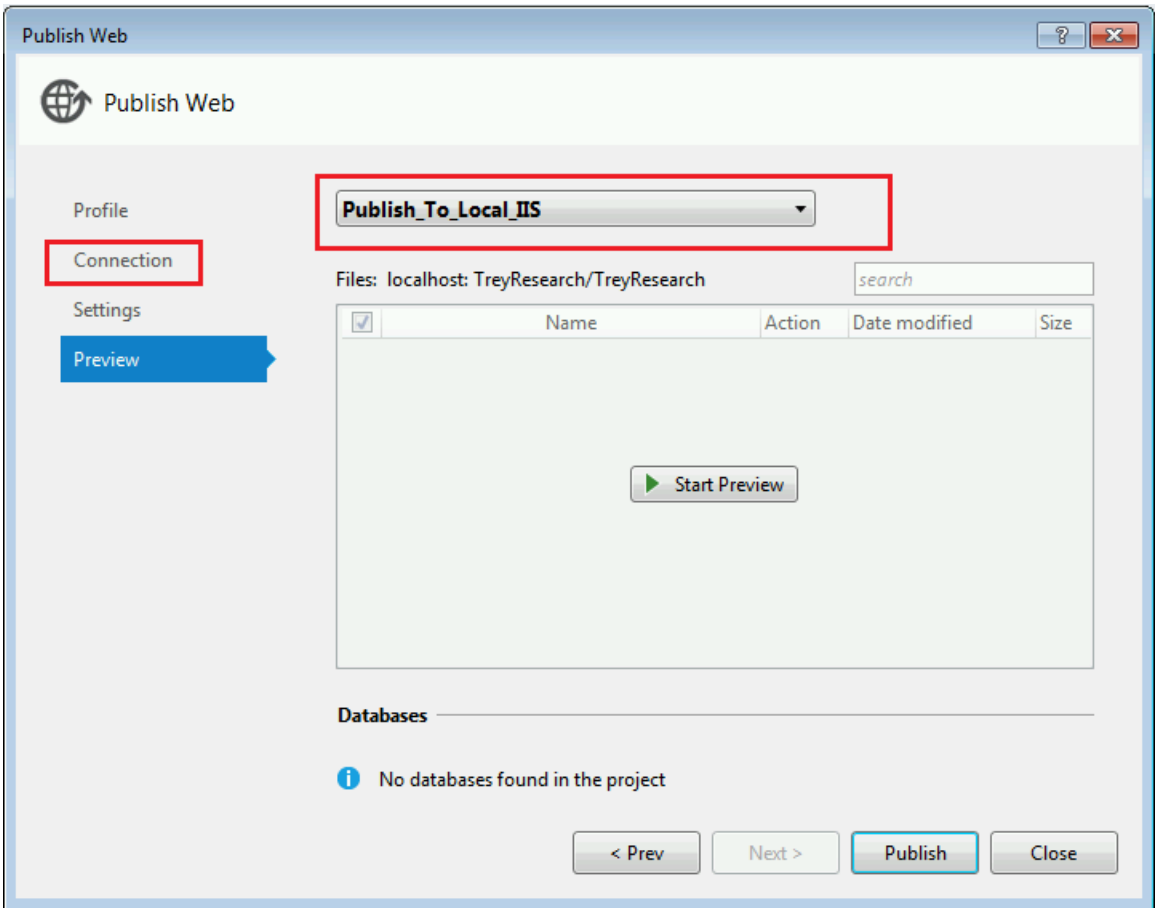
Task 2: Publishing WCF to IIS

In this task you publish the WCF web role to your local computer. You first create a profile, then publish the web role, and then validate that the WCF service was installed correctly.

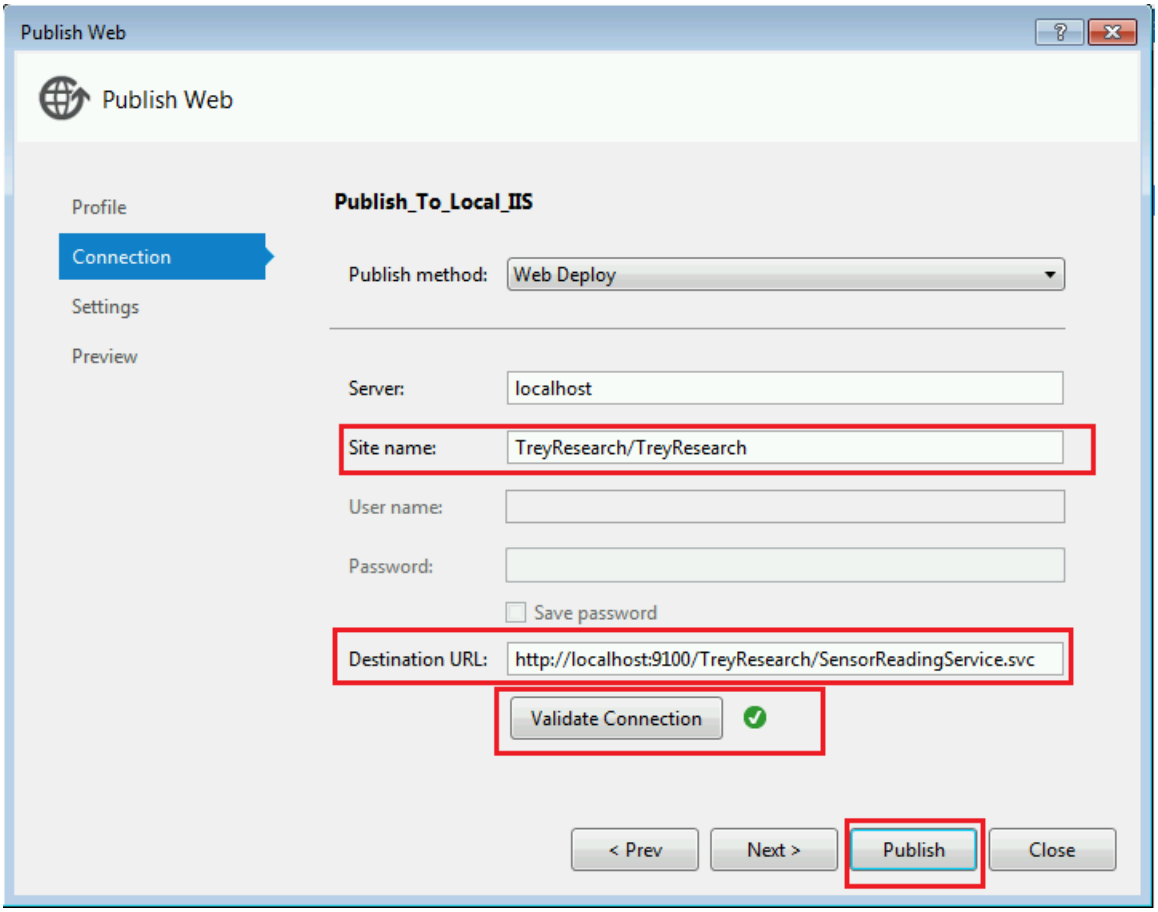
1. In Visual Studio, right-click **TreyResearch.WcfService.** In the popup menu, select **Publish**.



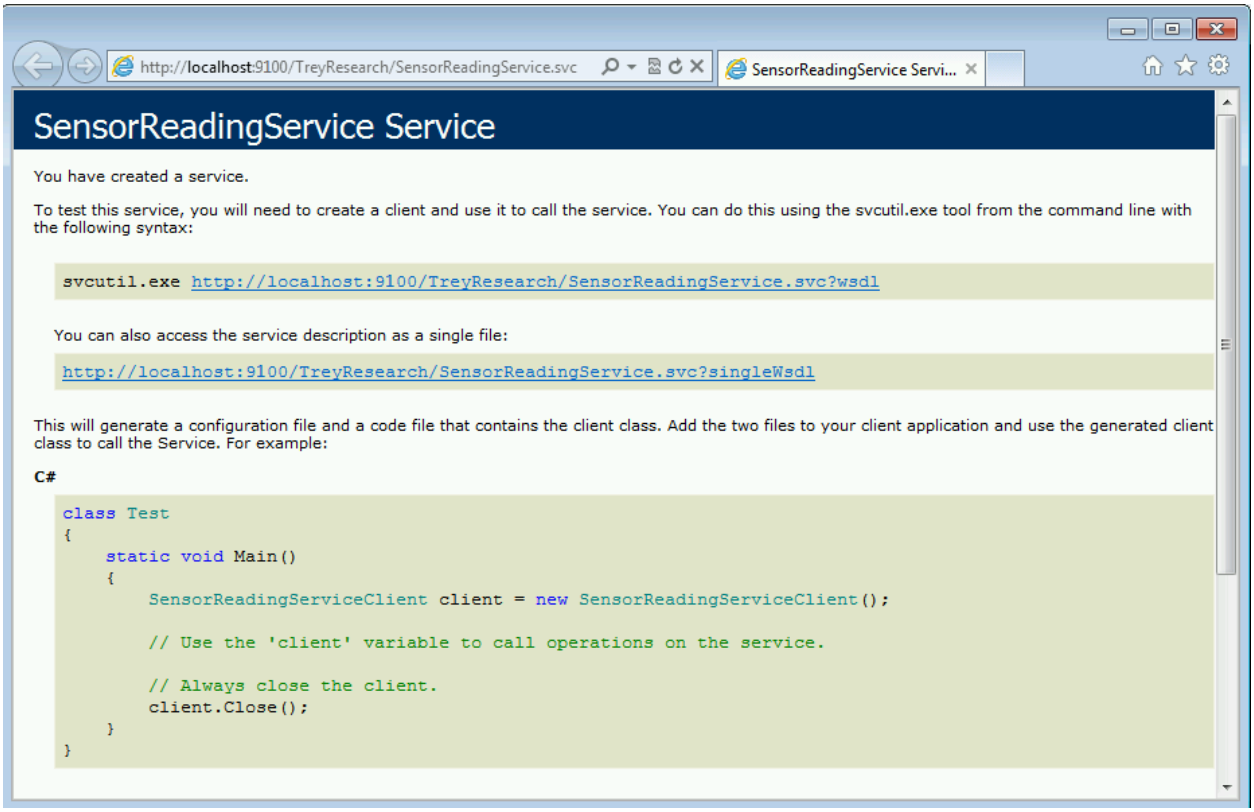
2. The **Publish Web** wizard appears. The profile **Publish_To_Local_IIS** profile should already be selected. If it isn't, select it from the drop-down list. Click the **Connection** tab.



3. In the **Connection** tab, the site name and destination URL are already set. Click **Validate Connection**. A green arrow should appear. Click **Publish**.



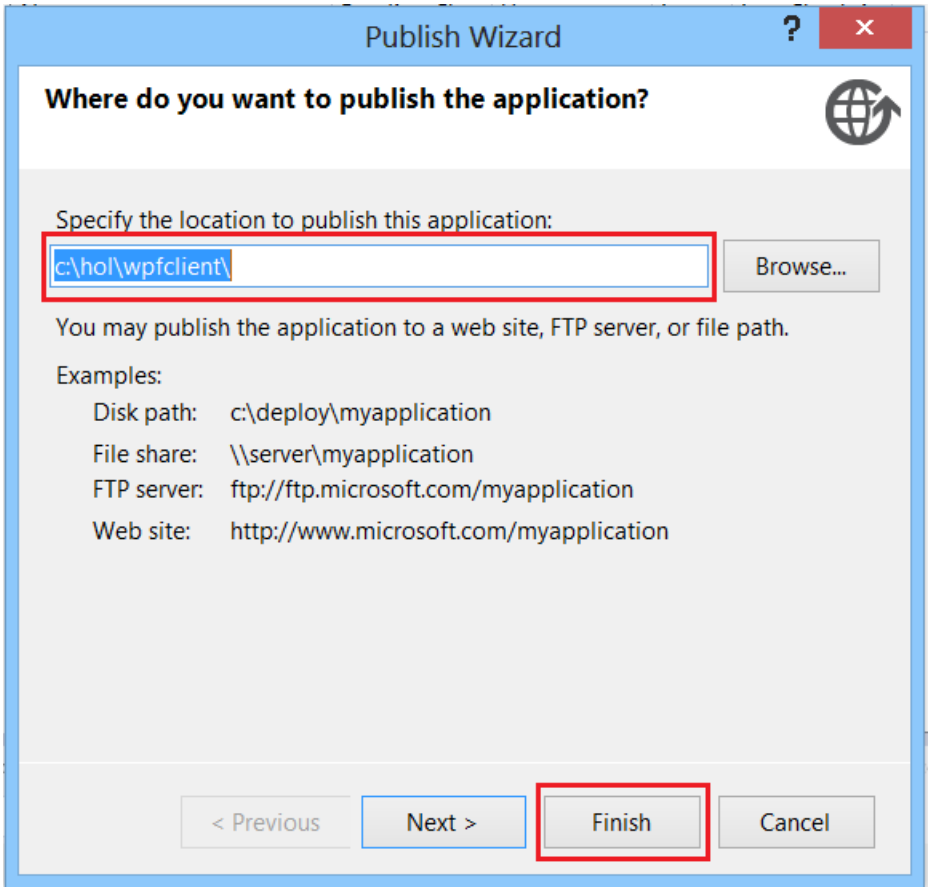
4. The browser launches and displays the Trey Research SensorReadingService.svc web page. The following screenshot shows an example.



Task 3: Deploying the WPF Application

In this task you use Visual Studio to deploy the WPF application to your local machine.

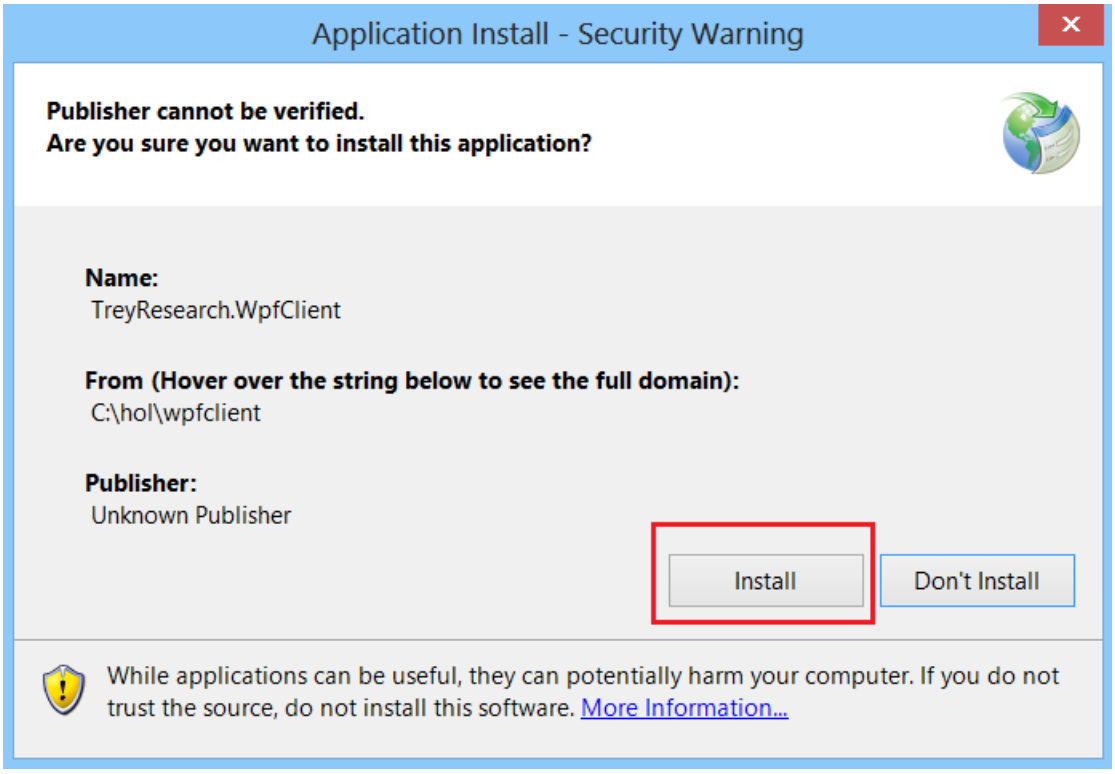
1. Right-click the **TreyResearch.WpfClient** project. Click **Set as StartUp Project**.
2. Right-click **TreyResearch.WpfClient** and select **Publish**. The **Publish Wizard** appears.



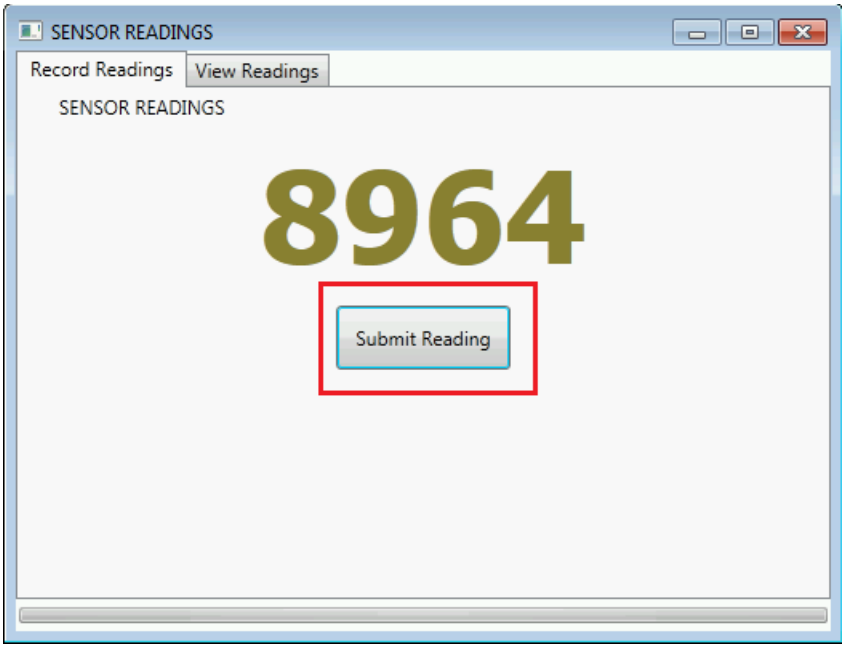
3. In the **Specify the location to publish this application** box, either enter or navigate to the location where the WPF application will be published. Click **Finish**. Visual Studio builds the project and publishes to that location.

4. The folder at the designated location opens. Click on either **Setup.exe** or **TreyResearch.WpfClient.application**. The **Application Install** dialog box opens. Click **Install**.

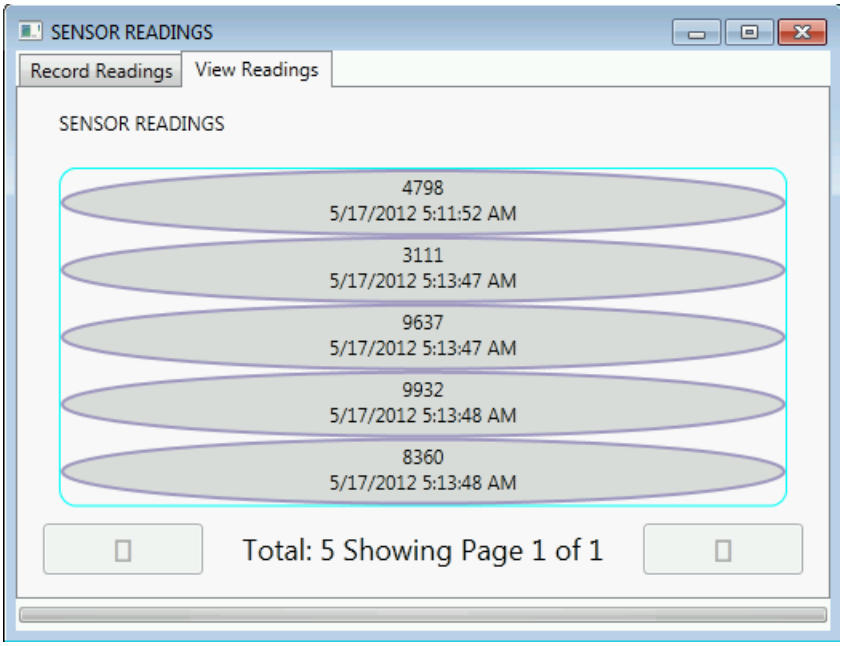
Note: The security warnings occur because there is no certificate for this application.



5. The application is installed and then launches. You should see the Trey Research application's **Sensor Readings** window. The following screenshot shows an example.



6. To test that the application installed correctly, click **Submit Reading** several times. You are submitting sensor data to the WCF service. Click the **View Reading** tab, which means you are reading data from the WCF service. The following screenshot shows an example.

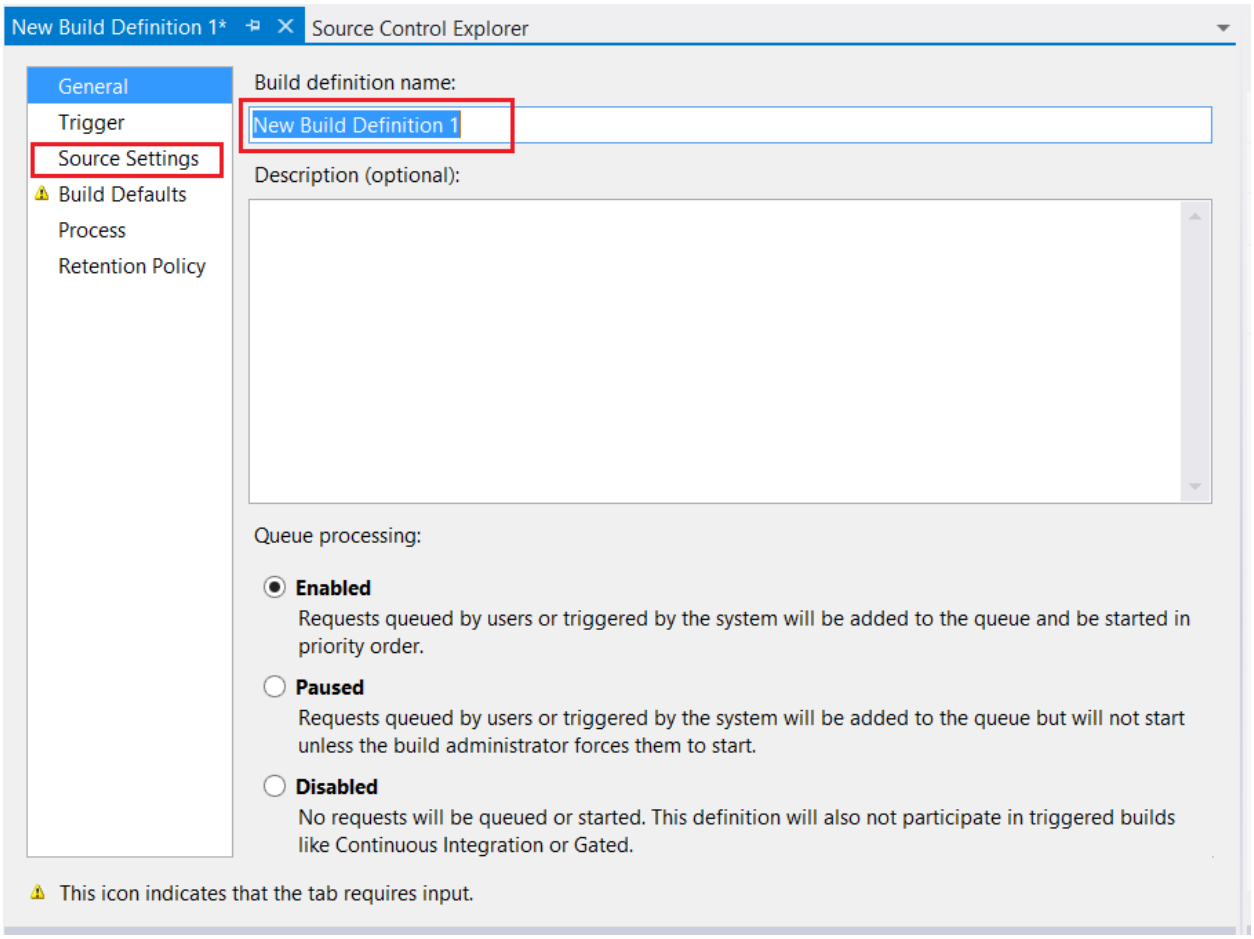


Exercise 4: Creating the Build definition for the Trey Research Pipeline

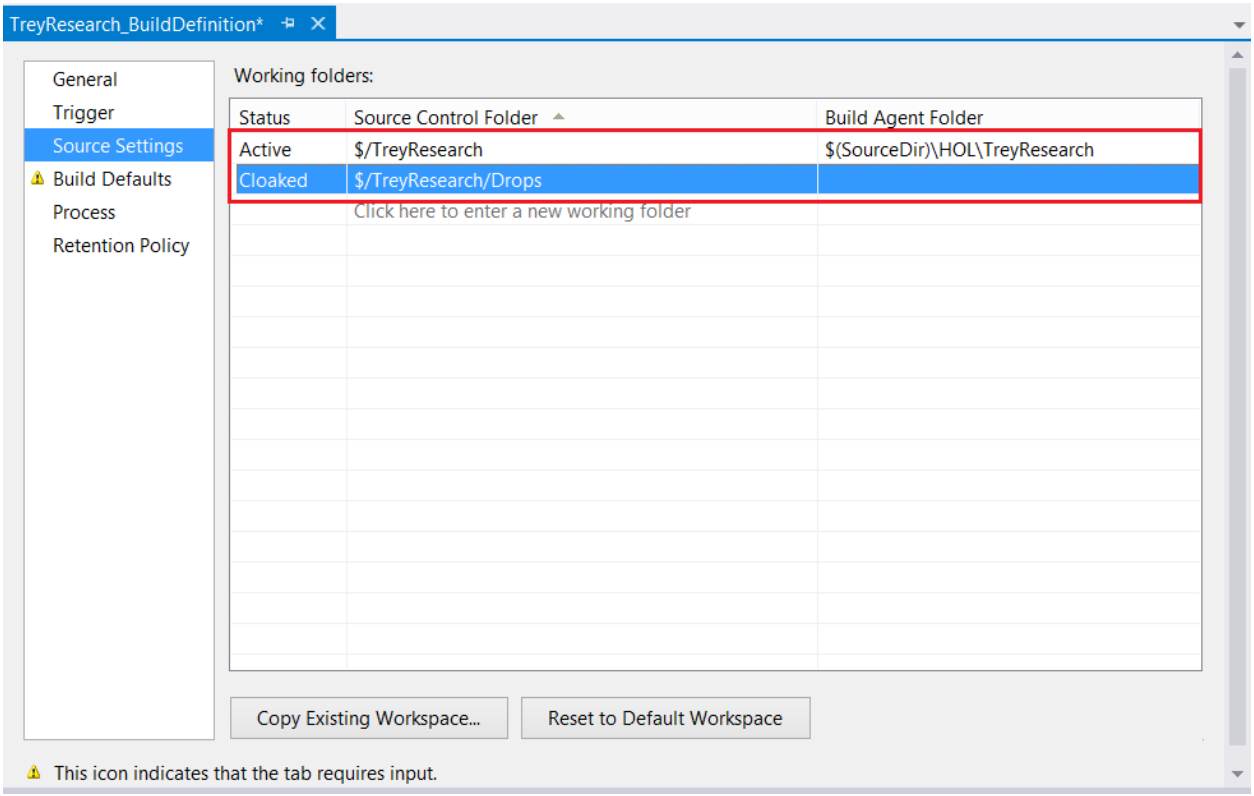
In the exercise you create a build definition for the Trey Research pipeline. For more information about build definitions, see How To: Create a Build Definition.

Task 1: Create the Basic Build Definition

1. In Team Explorer, click **Builds**.
2. Click **New Build Definition**.
3. In the General tab, in the **Build definition name** text box, enter a name for the build. Click **Source Settings**.

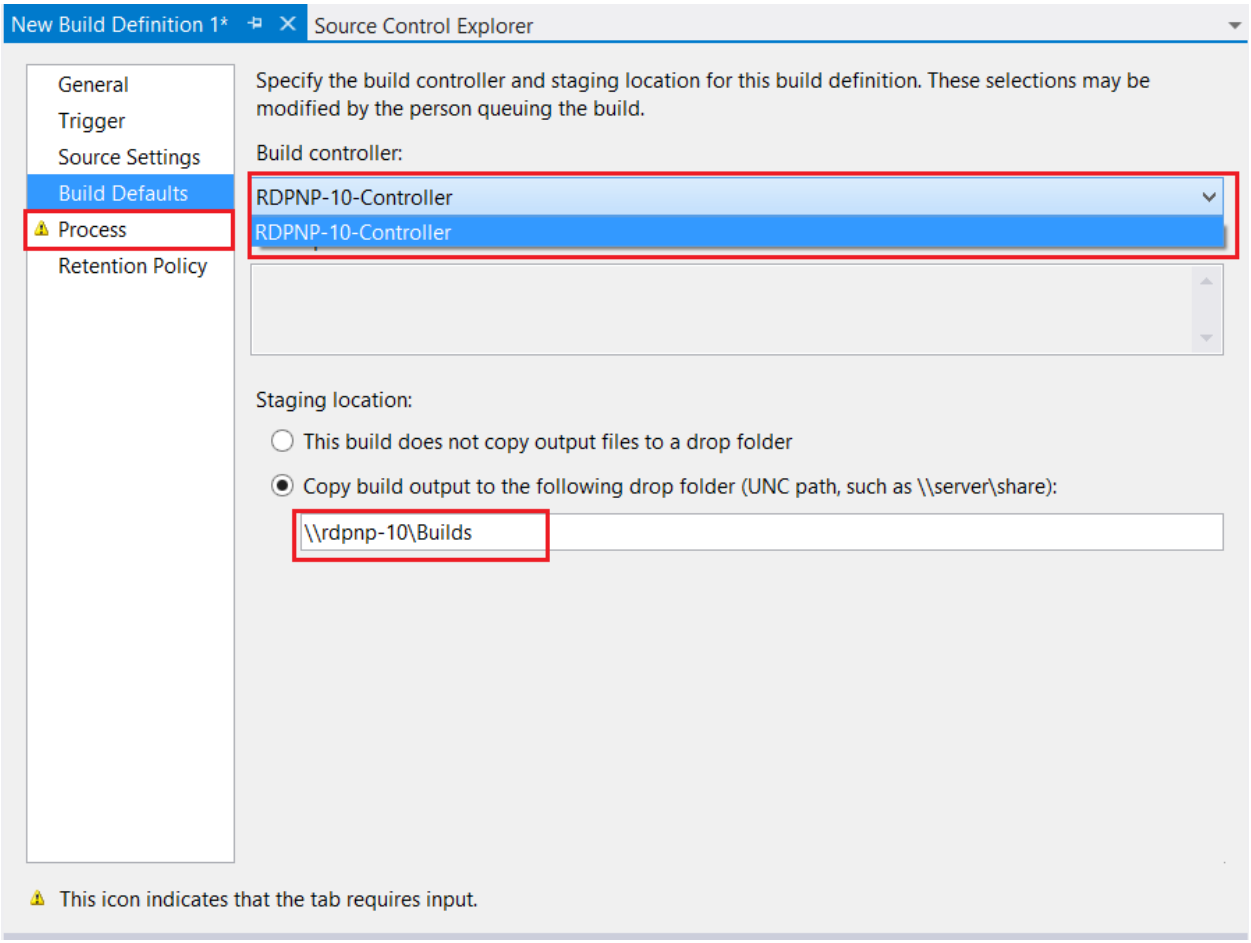


4. In **Source Settings**, check to see that the **Working folders** fields match the highlighted fields shown in the following screenshot. Click the **Build Defaults** tab.

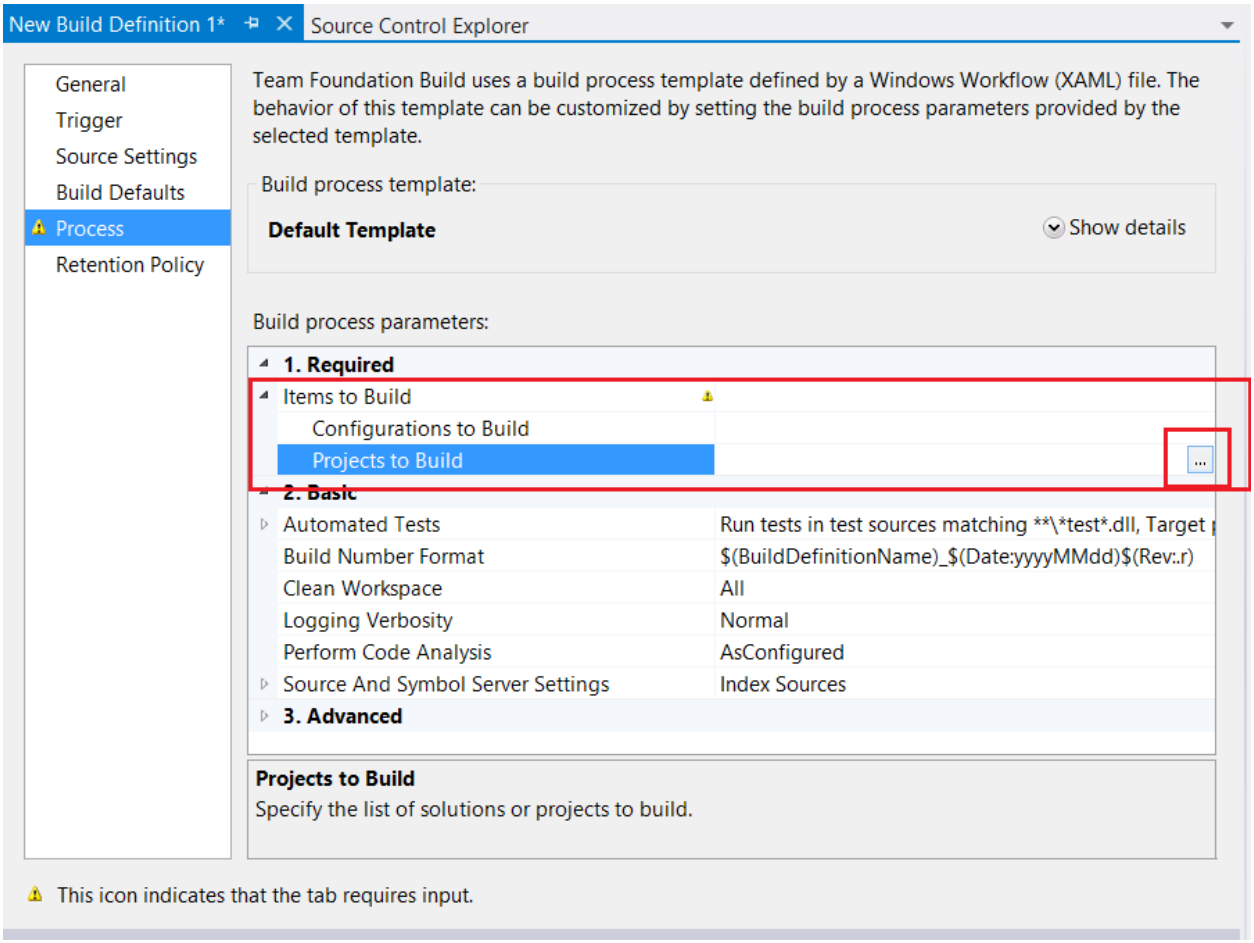


5. In **Build Defaults**, select the build controller used with TFS from the **Build controller** combo box. Under **Staging location**, enter the build directory in **the Copy build output to the following drop folder** text box. Select the **Process** tab.

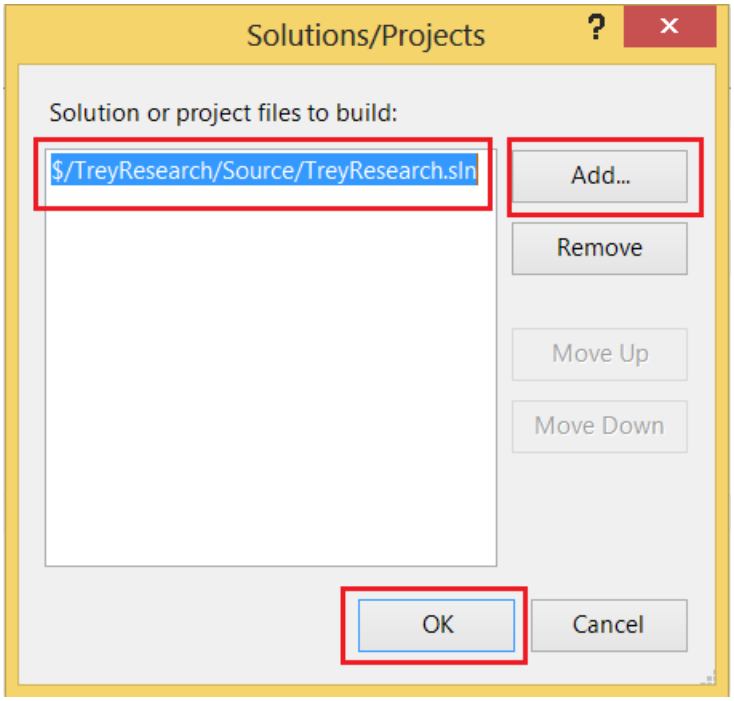
Note: In the Brian Keller VM the build directory is \\vsalm\\ffdrops.



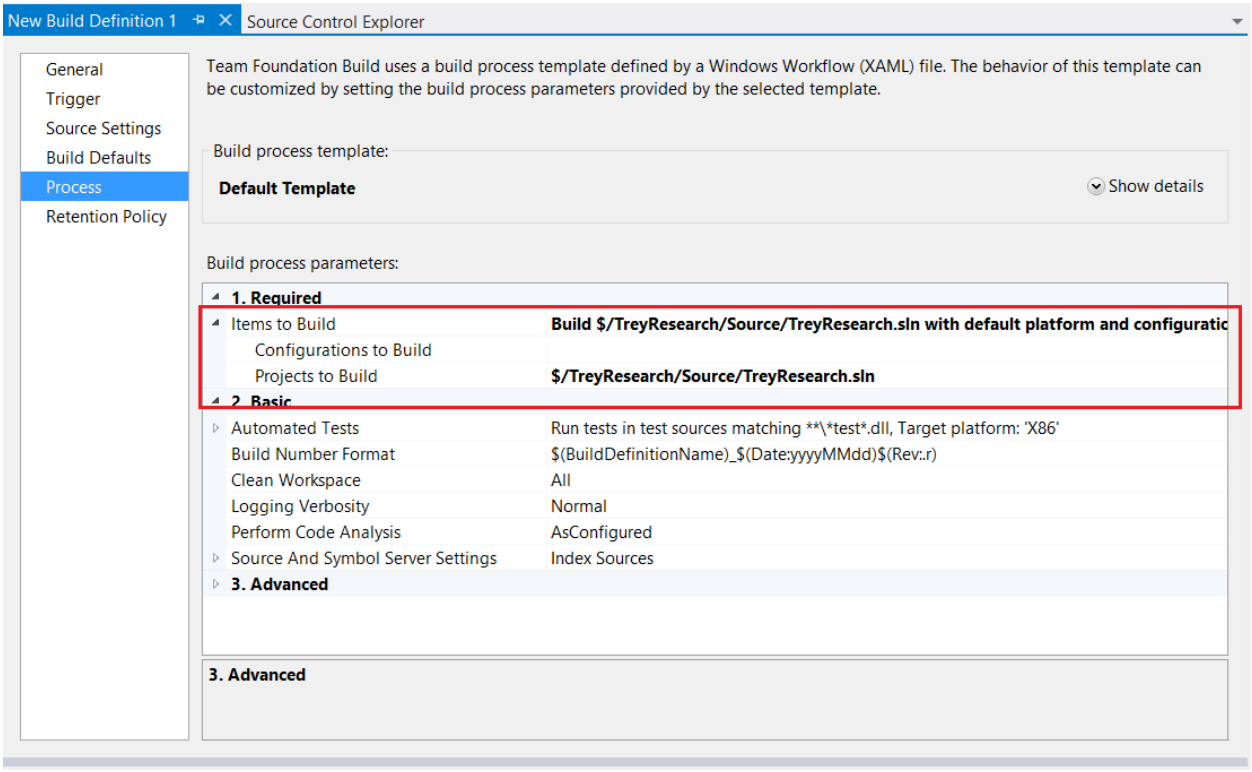
6. Under **Build process parameters**, add the solution to build under the **Required** heading. Expand the **Items to Build** node. In the **Projects to Build** field, click the ellipses **(…)**. The **Solutions/Projects** dialog box opens.



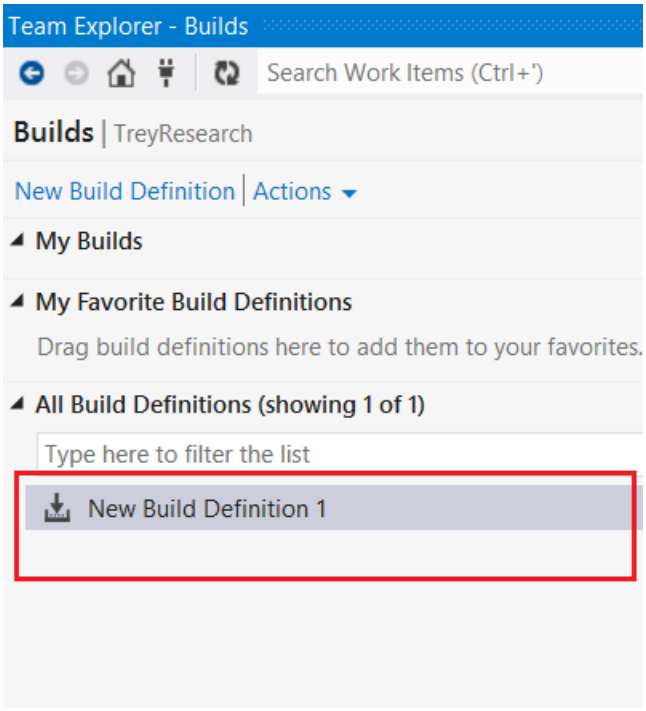
7. Click **Add** and browse to **TreyResearch.sln**. Click **OK**.



8. The **Process** tab should look like the following screenshot. Save the build.



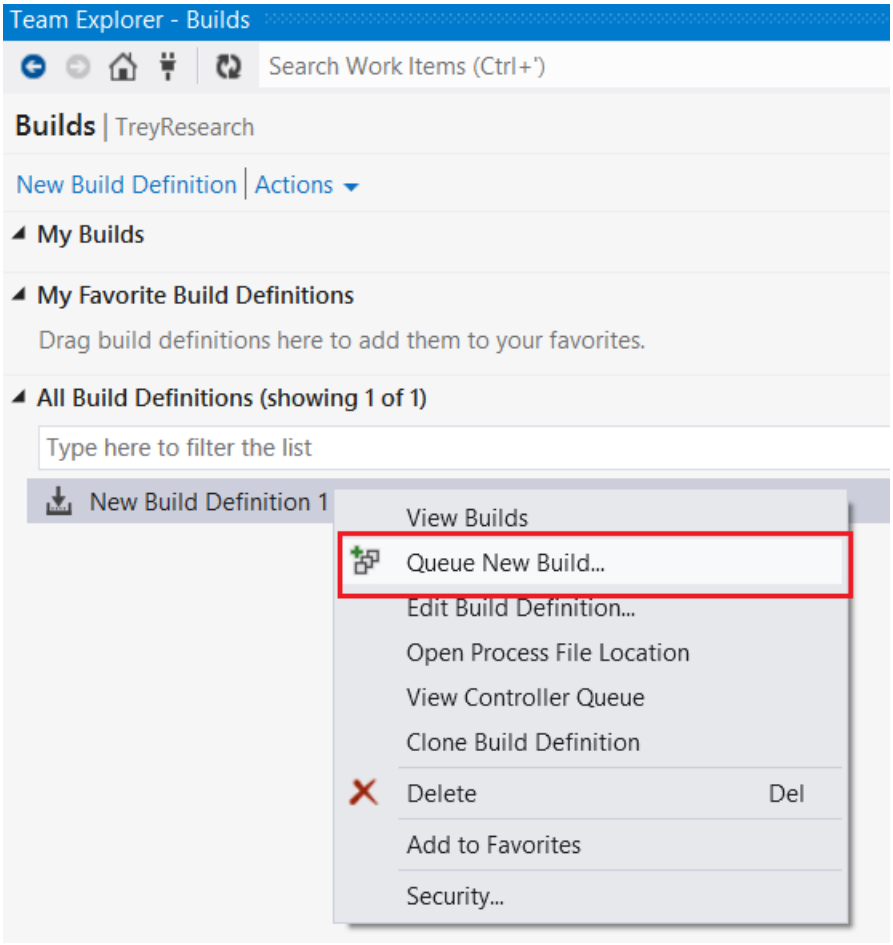
9. The new build definition will appear in the Team Explorer **Build** menu.



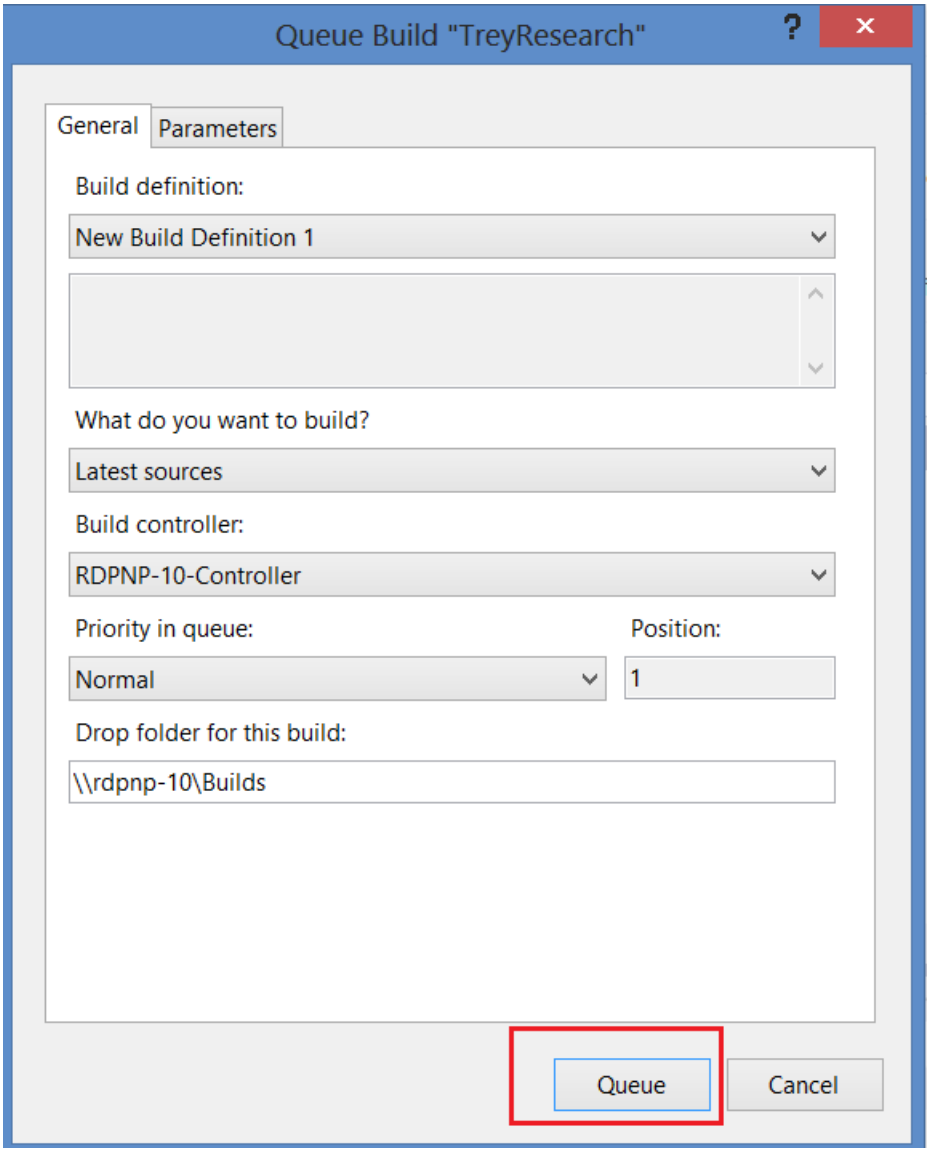
Task 2: Run the Basic Build Definition

In this task you run the build to make sure it works properly.

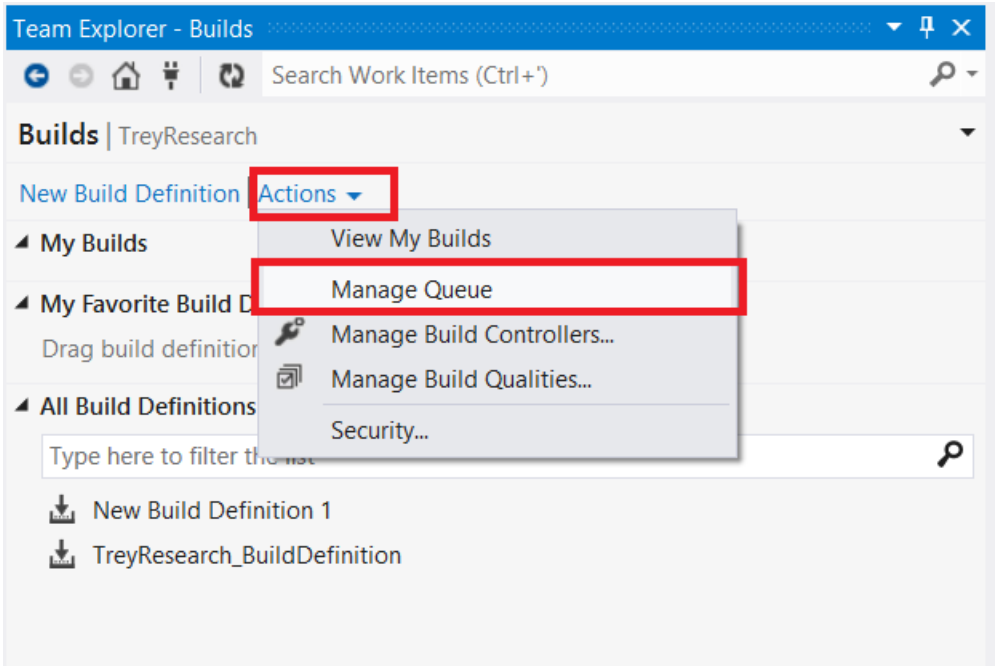
1. In Team Explorer, right-click on the new build definition to queue the build. Select **Queue New Build**. This launches the **Queue Build** dialog box.



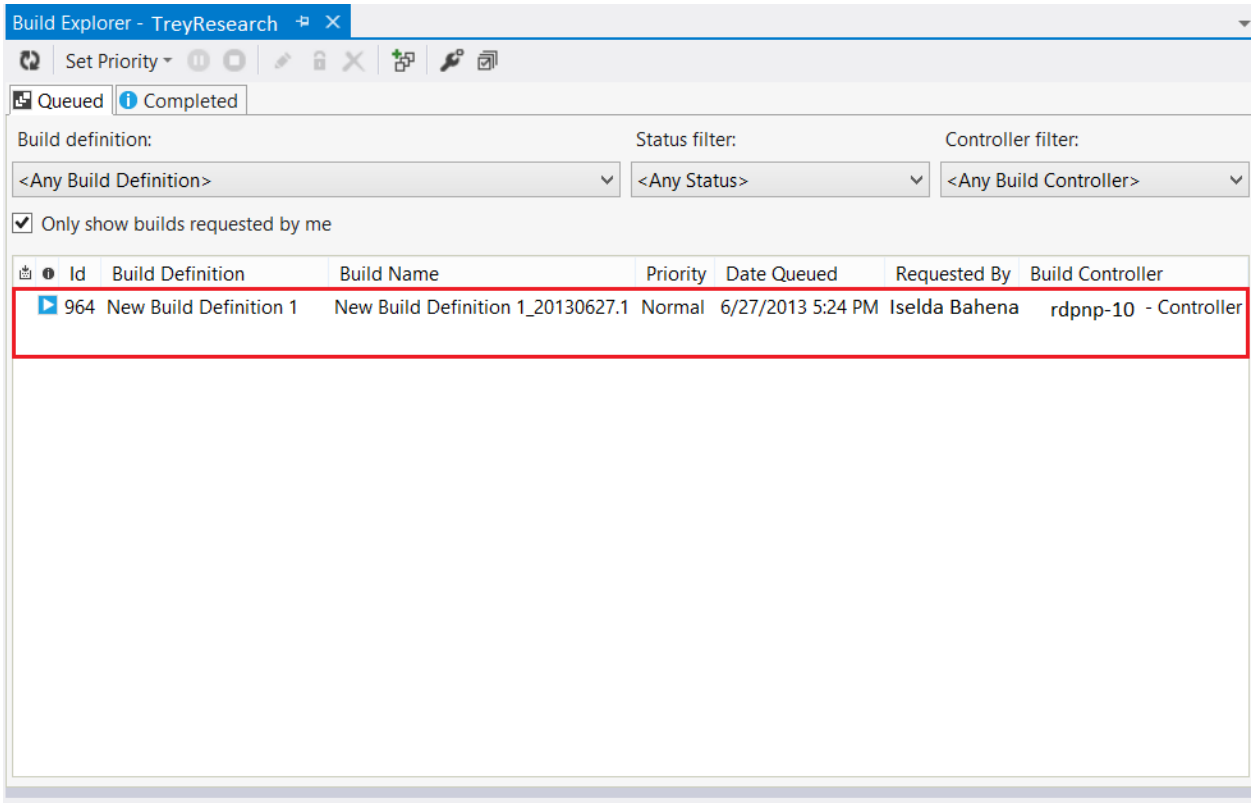
2. Leave the defaults as they are. Click **Queue**.



3. To watch as the build progresses, launch Build Explorer. Click **Actions**. Click **Manage Queue**.



4. In Build Explorer you can see both active and completed builds. The following screenshot is an example.



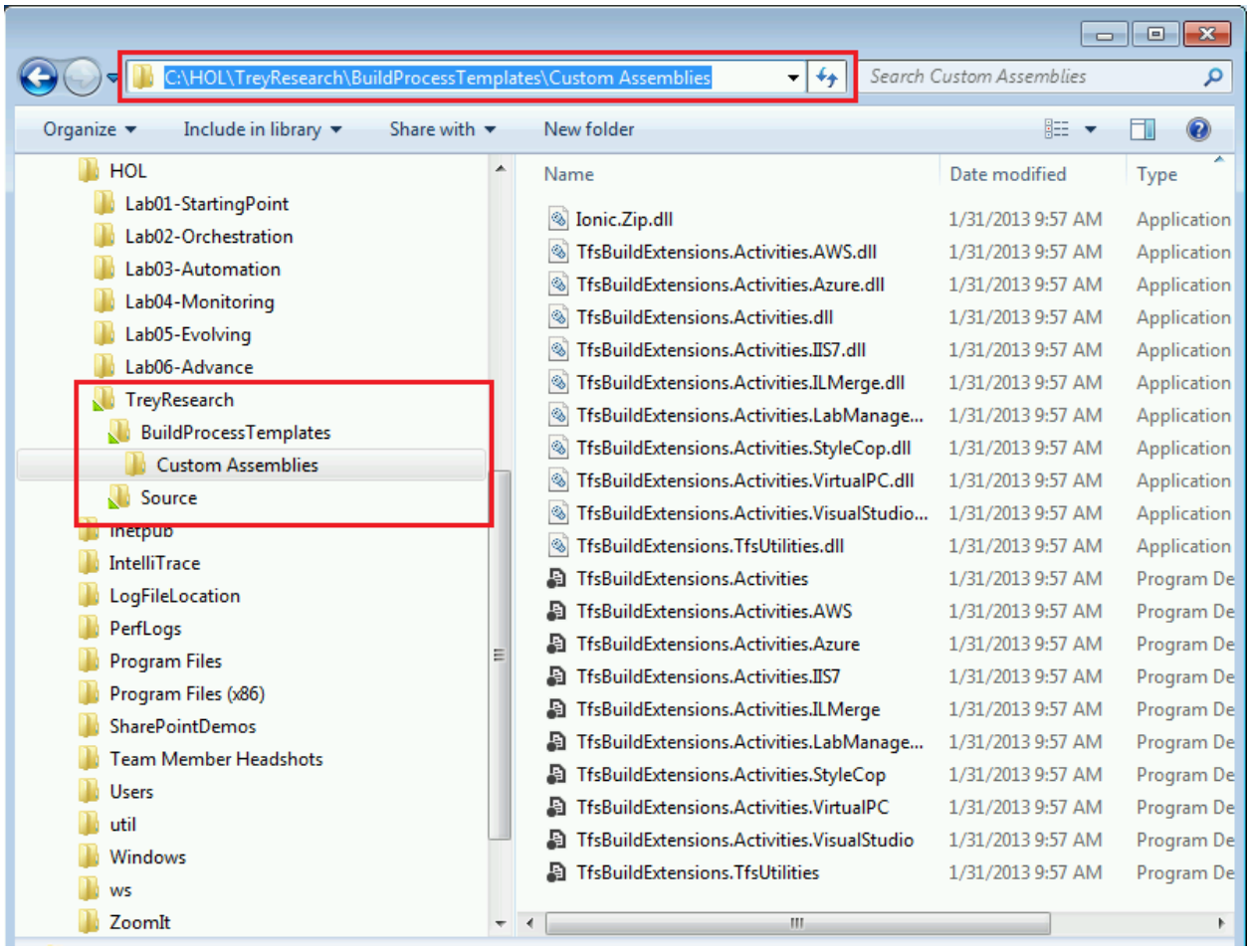
5. When the build is done it appears in both Build Explorer and the **Build** menu.

Exercise 5: Install the TFS Build Extensions

In this exercise you install the TFS Build Extensions. To do this, follow these steps.

Task 1: Copy Files to Custom Assemblies Folder.

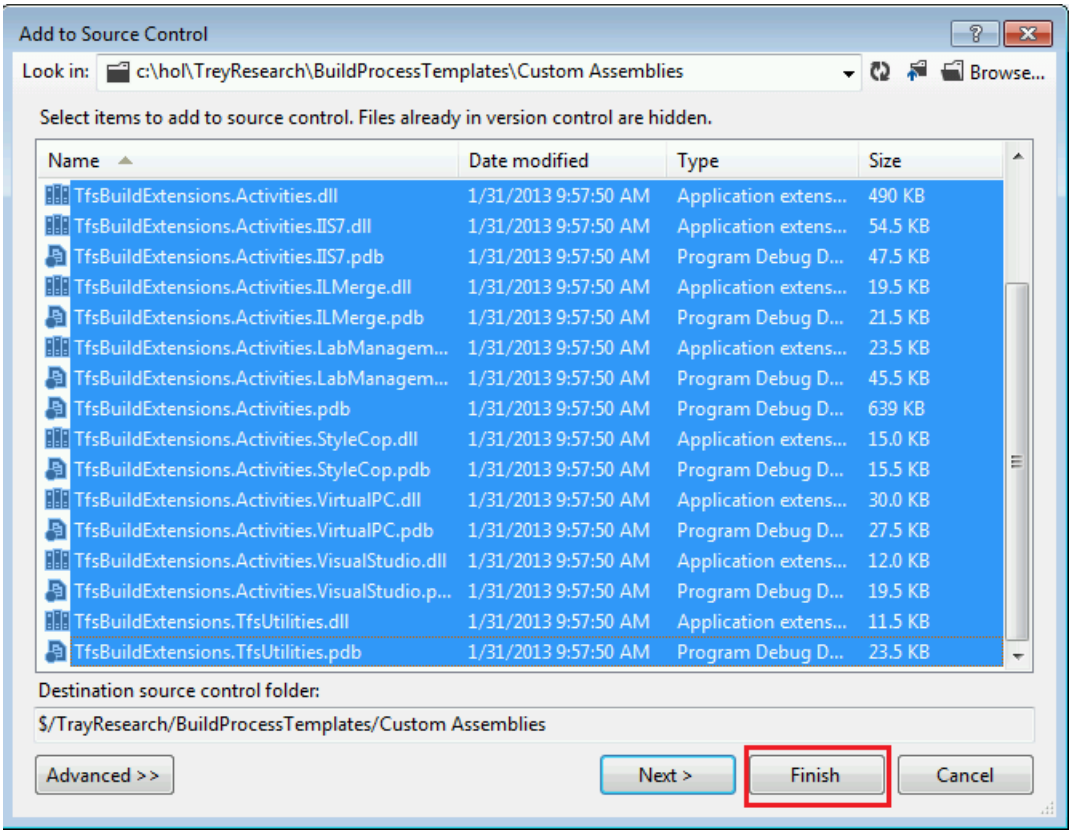
1. Unzip the file TfsBuildExtensions.zip that you downloaded in Exercise 1.
2. Go to your **TreyResearch** Folder and locate the **BuildProcessTemplates** folder. This was created when you created the Team Project.
3. Create a new folder under the **BuildProcessTemplates** called **Custom Assemblies**.
4. Copy all the assemblies located in subfolder **Code Activities\VS2012** of the unzipped package to the **Custom Assemblies** folder.
5. The following screenshot shows the contents of the Custom Assemblies folder.



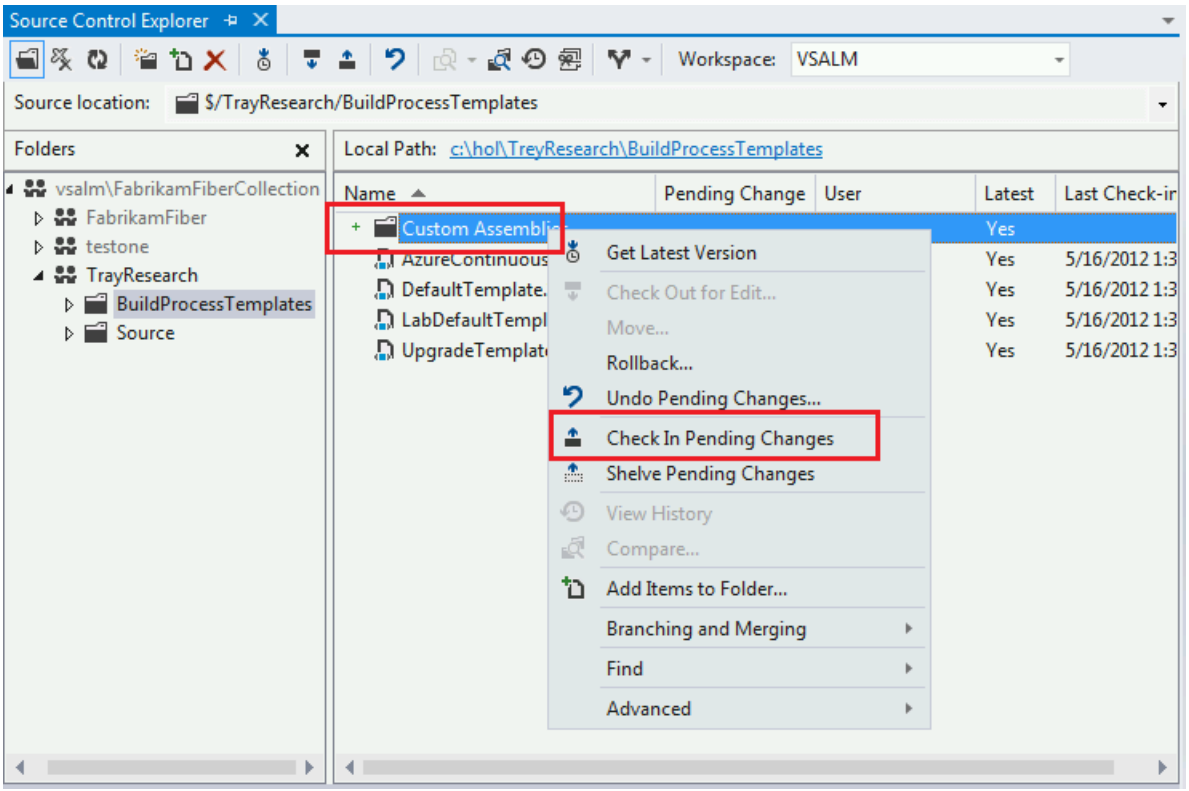
Task 2: Add Custom Assemblies files to TFS

Follow the same steps in Exercise 2 Task 3 to add the TreyResearch.SLN code to TFS.

1. In Solution Explorer, navigate to the team project, **TreyResearch**.
2. Right-click on the **BuildProcessTemplates** folder and select **Add Items to Folder**. The **Add to Source Control** dialog comes up.
3. Select the **Custom Assemblies** folder and double click on this selection. All the assemblies' files will be shown. Select them all, as shown. Click Finish, all files will now be pending to be added.



4. To check in the files, right-click on the **Custom Assemblies** folder and select **Check In Pending Changes**.

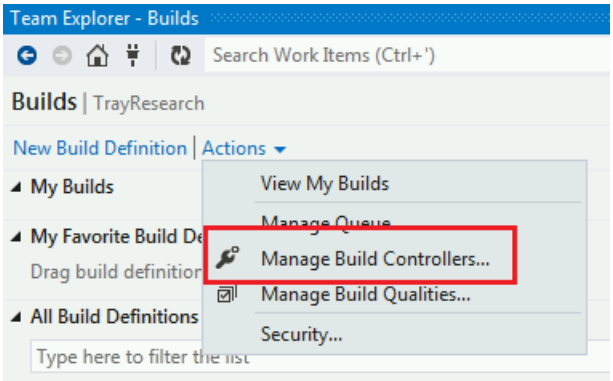


5. In Team Explorer, the **Pending Changes** dialog box appears. Click **Check In**. The check-in confirmation message appears. Click **OK**.

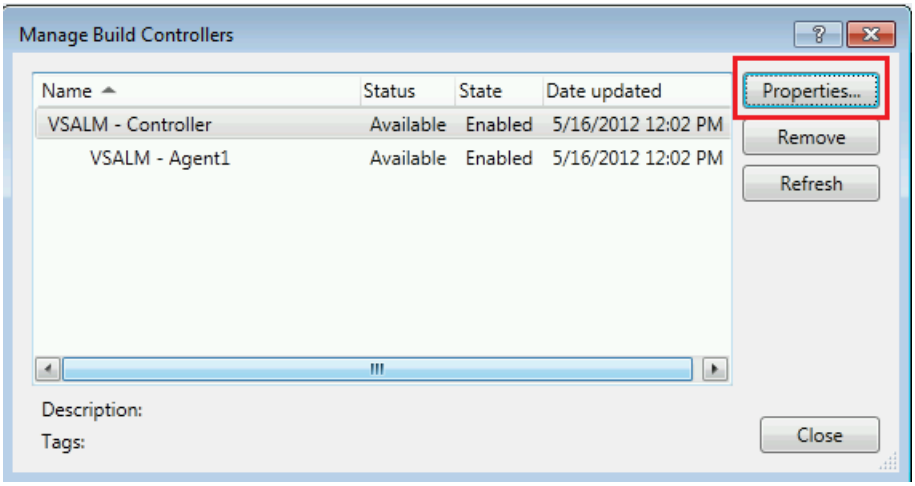
Task 3: Set the Version Control for the build controller to the Custom Assemblies

In this task you point the build controller to the Custom Assemblies folder.

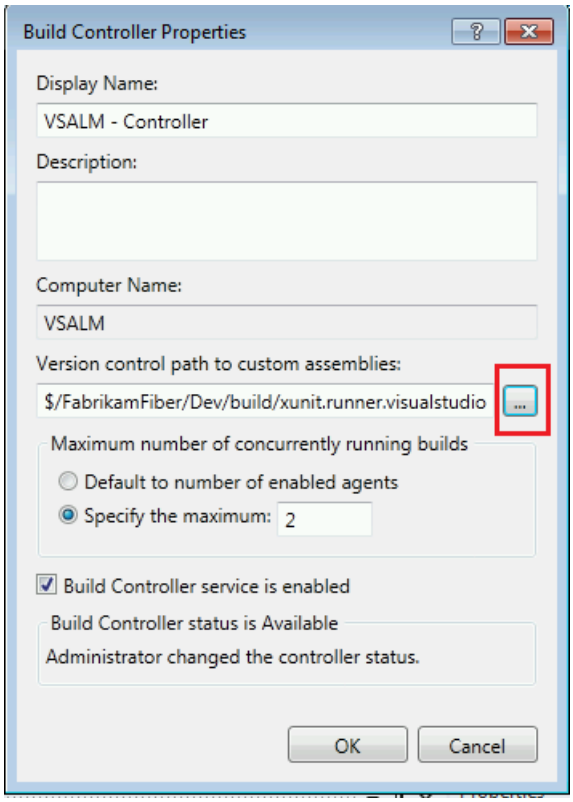
1. In Team Explorer, click **Builds**.
2. Select **Actions**. Select **Manage Build Controllers** from the drop-down menu. The Build Manager dialog box appears.



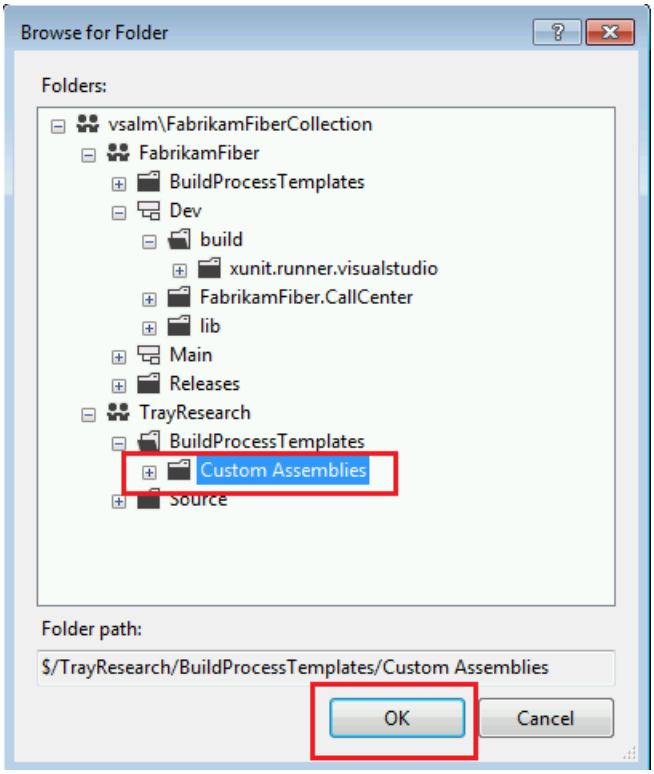
3. In the Manage Build Controllers dialog box, click **Properties**.



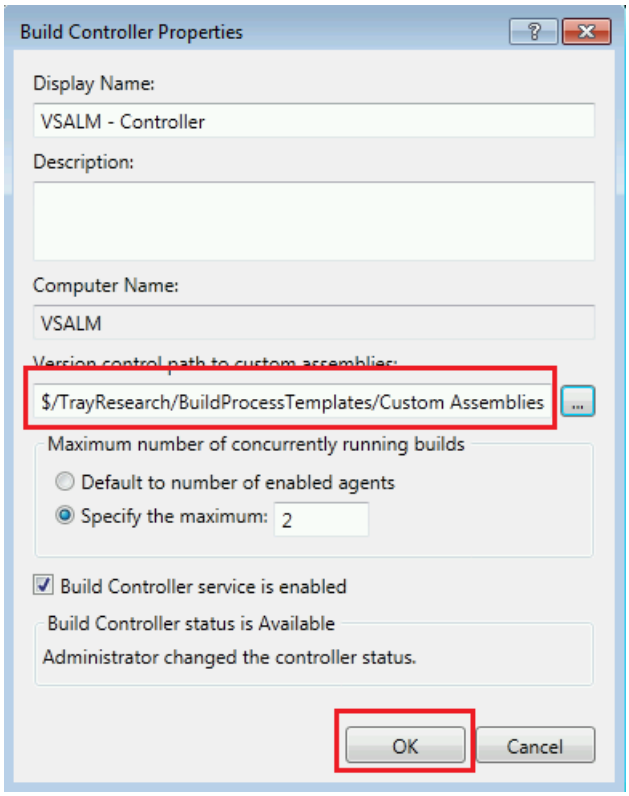
4. The **Build Controller Properties** dialog box appears. By default, the version control path is set to $/Fabrikam/Dev/build/xuint.runner.visualstudio. Click the ellipses **(…)**.



5. Navigate to the TreyResearch project and select the **Custom Assemblies**. Click **OK**.



6. The **Build Controller Properties** dialog box appears. The version control path should now point to the TreyResearch Custom Assemblies folder. Click **OK**.



7. Close the **Manage Build Controller** dialog.

Exercise 6: Setting Up the Target Environments

In this exercise you set up the target environments so that they can host the WCF services and so that they can be managed by the pipeline.

The pipeline has four environments: development, testing, staging, and production. Because the development environment is isolated and only exists on the development machines, it doesn't host any services. However, the three other environments host the WCF service, so each requires an IIS website.

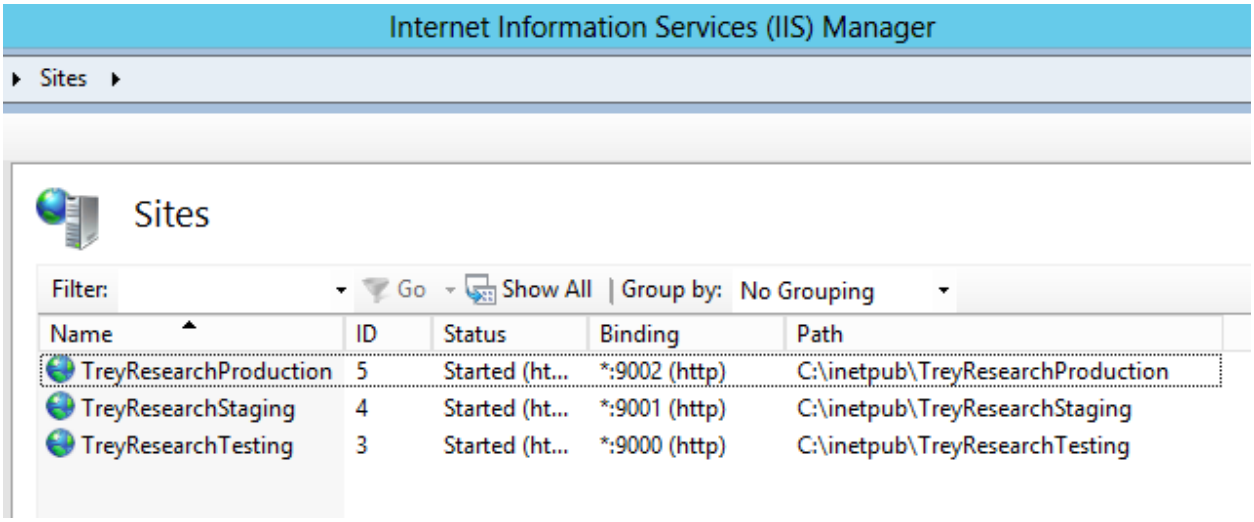
Typically, these three websites would reside on three different IIS servers across different computers. For simplicity, this lab creates three different websites on the same IIS server.

Task 1: Set Up the Web Server

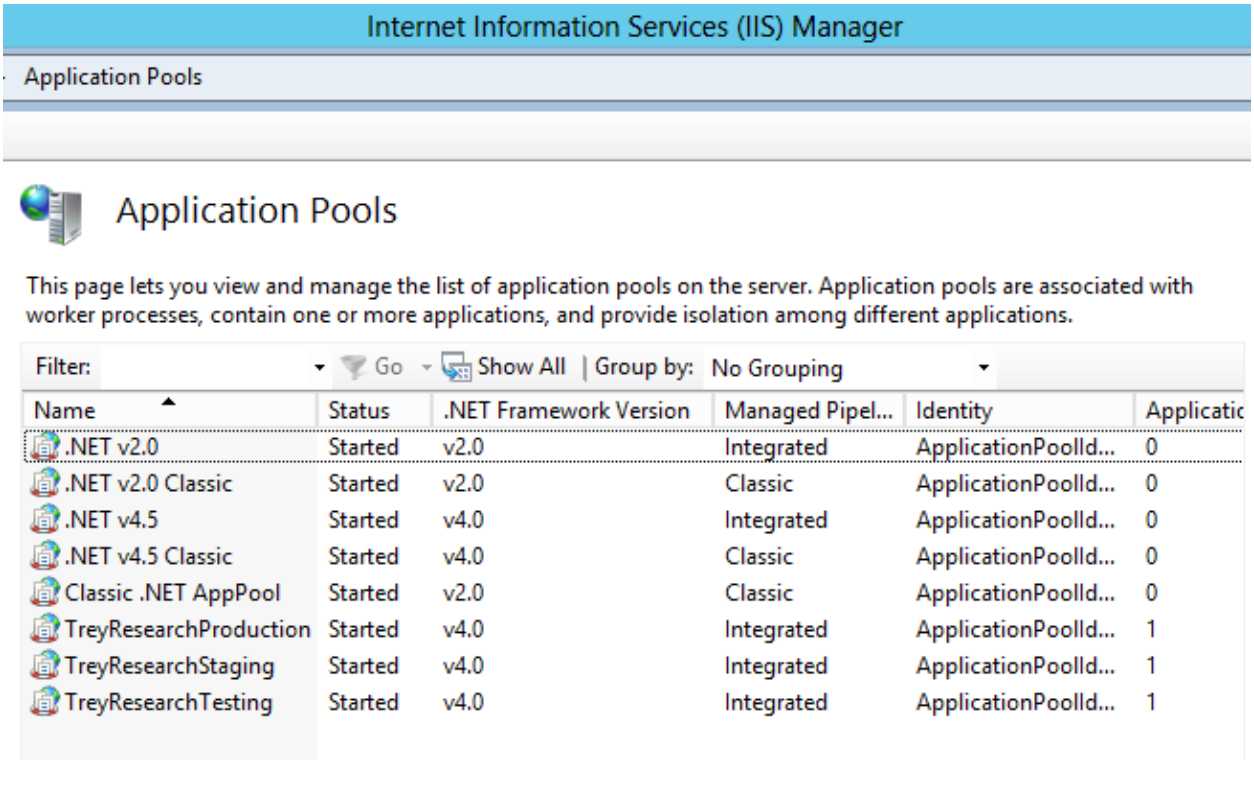
In this task you set up the computer that acts as the web server. This task assumes that IIS is already installed on the target machine.

1. Make sure that all the roles and features required to host WCF 4.5 services are installed. The minimum requirements are everything listed under the following sections is setup under the roles and features settings.
    - Application Server
    - Web Server (IIS)
    - .NET Framework 3.5 Features
    - .NET Framework 4.5 Features
    - Windows Process Activation Service
2. Install Web Deploy on all the development machines as well as the build machine(s) that host the build agent(s) that run the commit stage of the pipeline. Web Deploy contains the tools that transform the configuration files and package the files to be deployed. You can use either the WebPI installer or the standalone installer. Choose **Typical** when prompted by the installation wizard.
3. Create three IIS websites, one for each environment. Name them so that you can tell one environment from another. The following screenshot shows an example of how to name the three websites.

> **Note:** If you need more information about how to create IIS websites, see Create a Web Site.



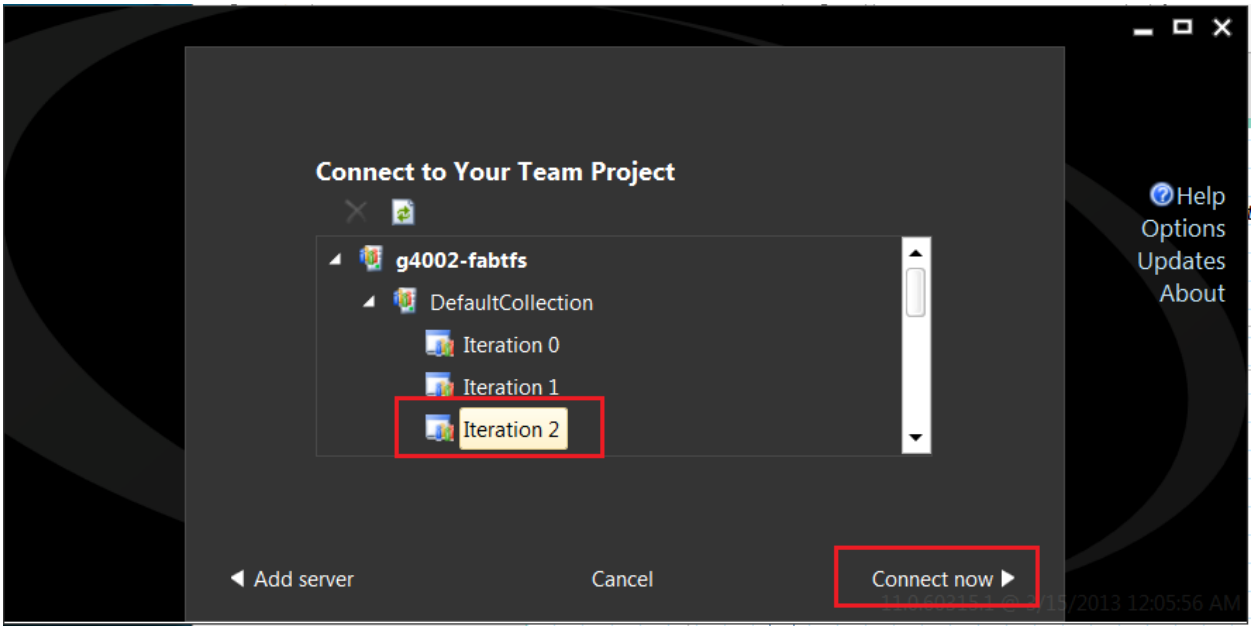
4. Make sure that each environment's associated application pool runs .NET Framework v4.0. The following screenshot shows an example. For further information about changing the .NET Framework for an application pool see, [Specify a .NET Framework Version for an Application Pool (IIS 7)].



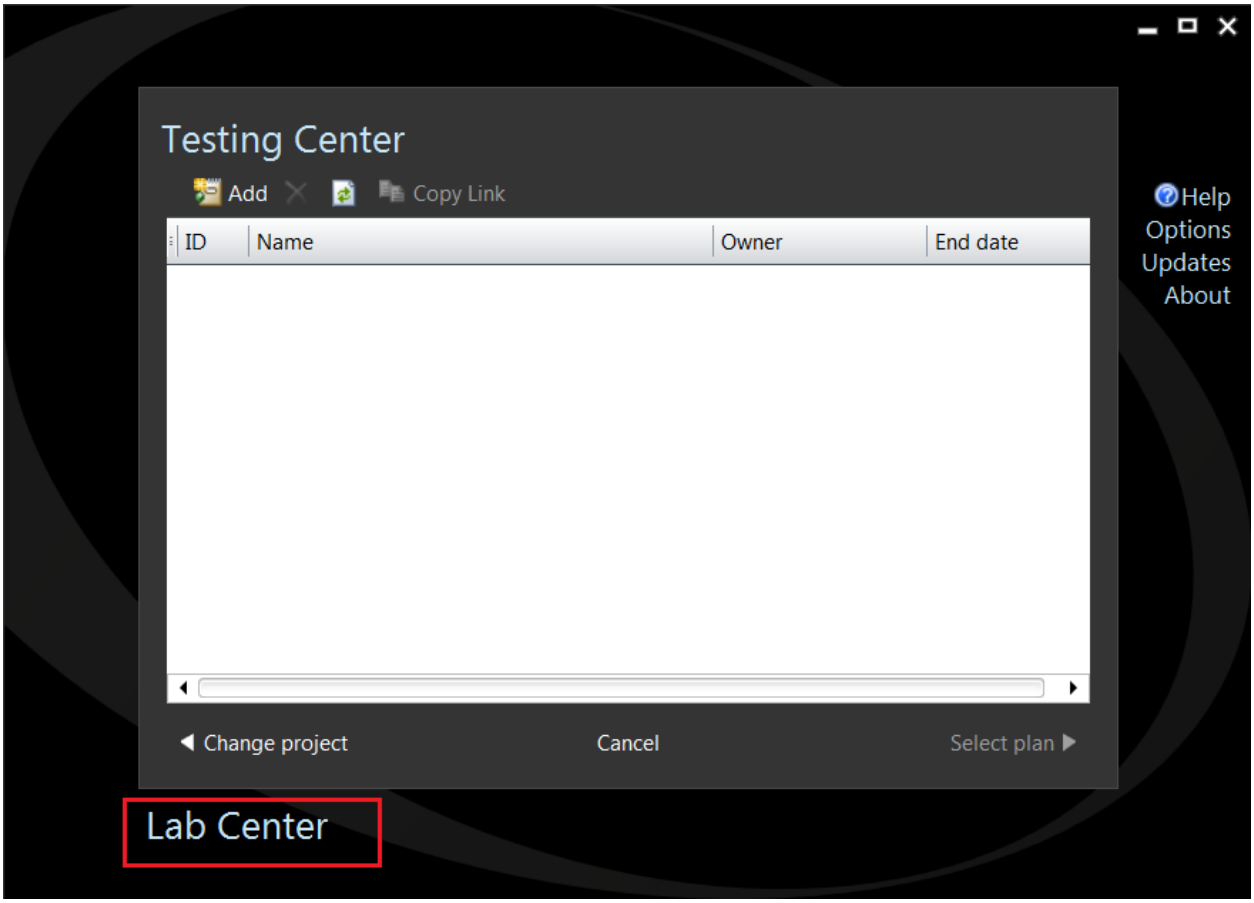
Task 2: Set Up the Environment in Lab Management

In this task you add the web server to a Lab Management environment. The pipeline uses Lab Management to manage the environments and for automation. Lab Management will automatically install the test agents on the target computers.

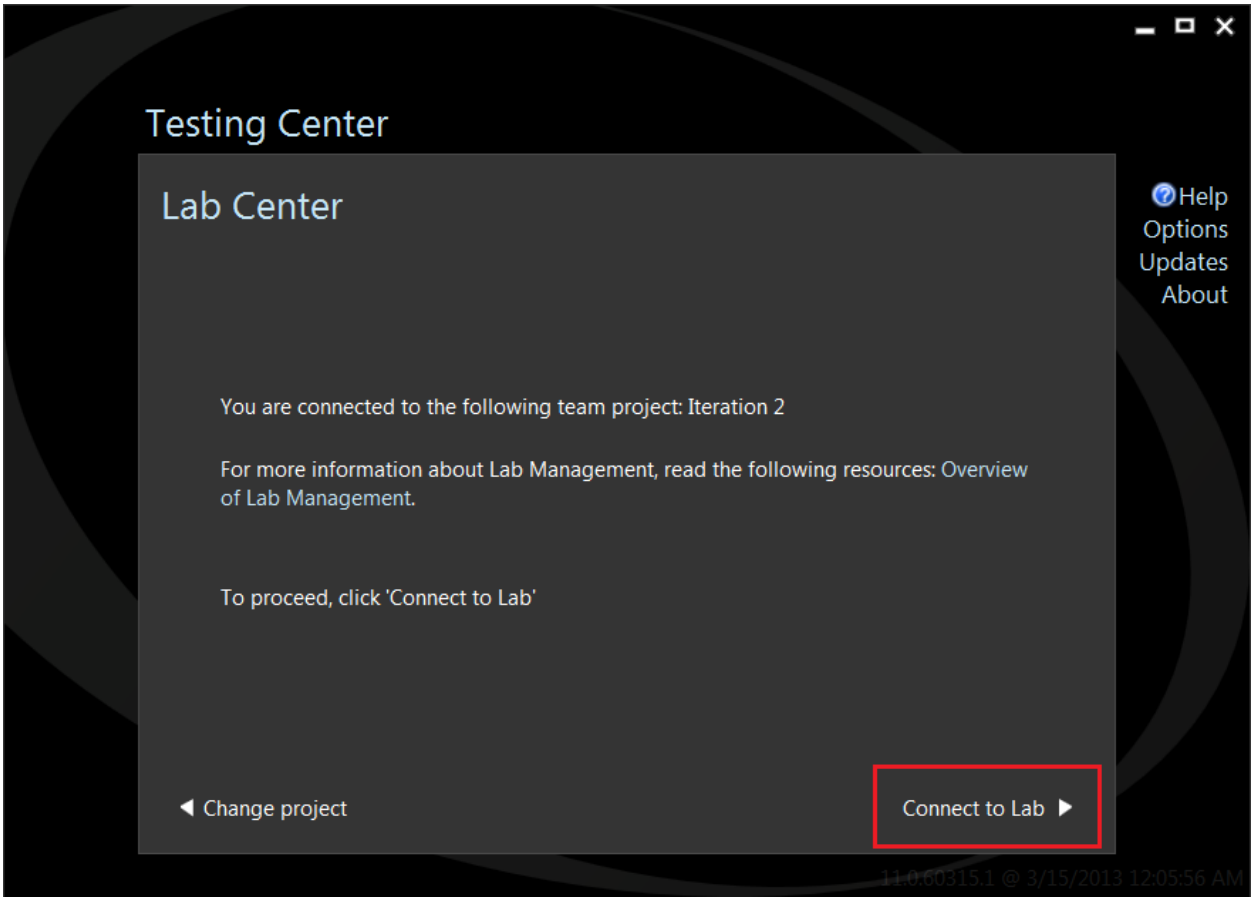
1. Open **Microsoft Test Manager**.
2. Select the TFS computer that stores the team project and that contains the build definitions for the pipeline orchestration. Select **Connect Now**.



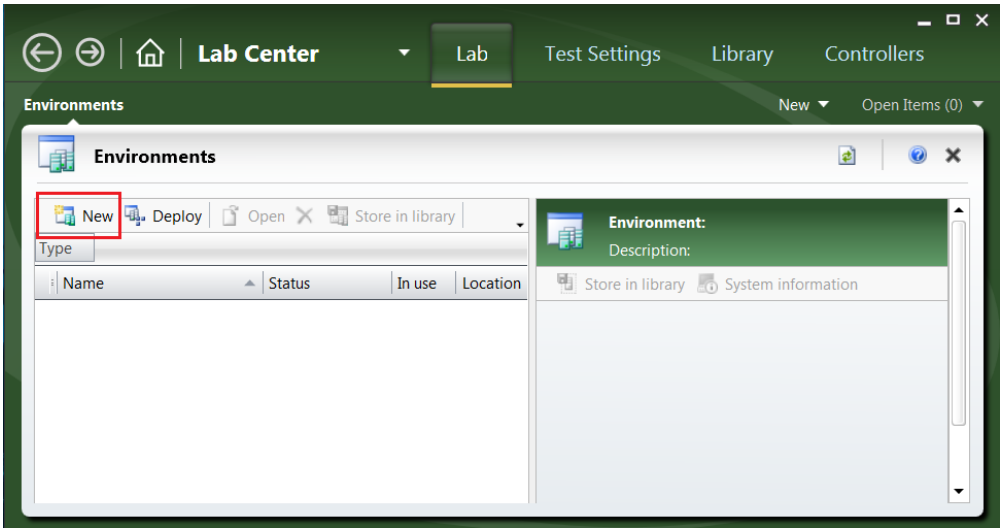
3. The Testing Center page opens. Click **Lab Center**, located near the bottom of the page.



4. The Lab Center page opens. Click **Connect to Lab**.

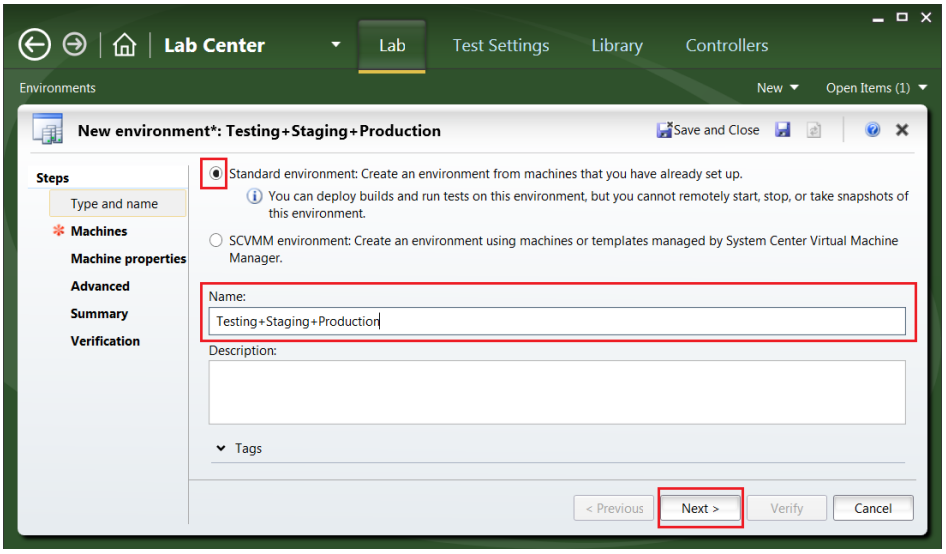


5. The Lab Center panel appears with the **Lab** tab selected. You should see the **Environments** pane. Click **New**.



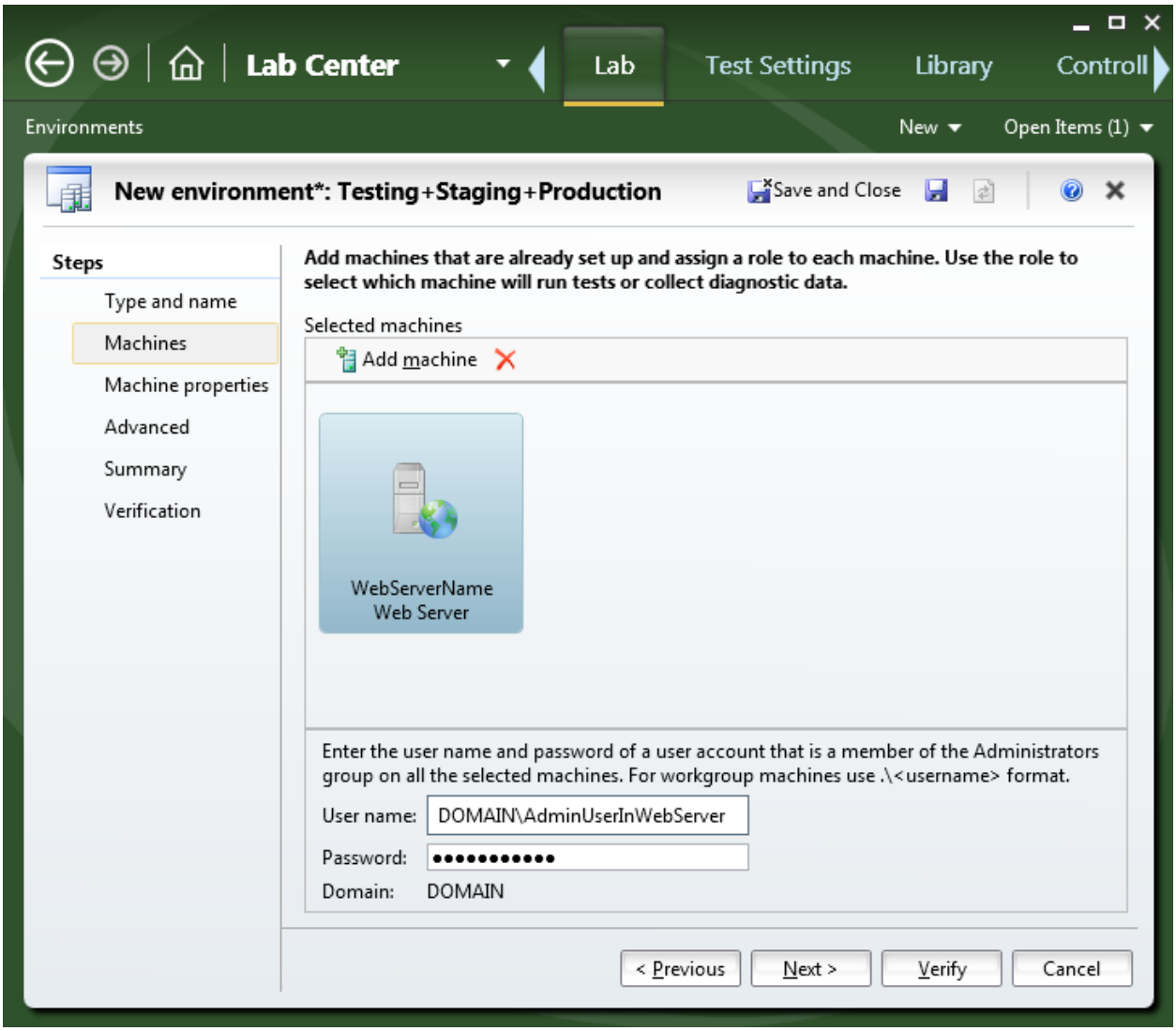
6. The **New Environment** wizard appears. The **Type and name** tab is selected. Select **Standard environment**.

7. In the **Name** text box, provide a meaningful name for the environment. The example shown in the screenshot is **Testing+Staging+Production** because the same machine is used for all three environments. Click **Next**.

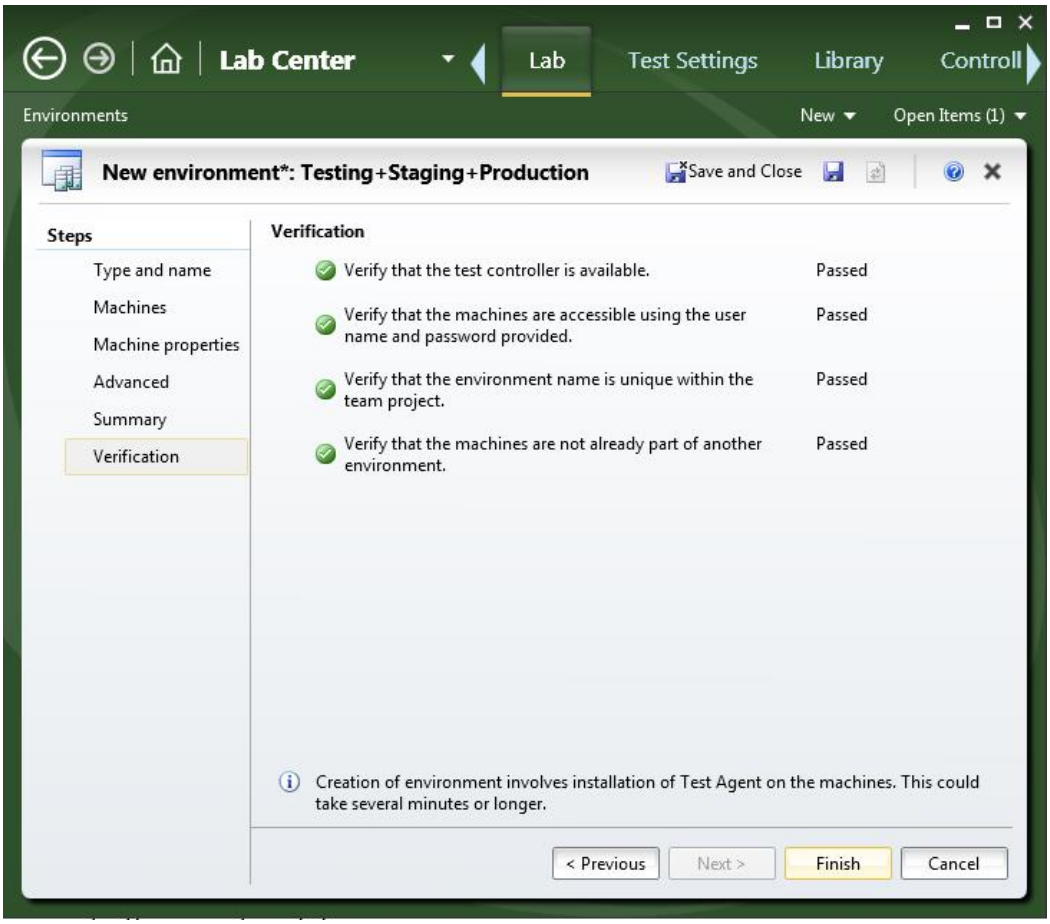


8. The **Machines** tab is selected. Click **Add Machine** and select the web server that you set up in Task 1.
9. In the **Computer name** field, provide either the NetBios or DNS name.
10. In the **Type Role** field enter **Web Server**.
11. In the **User name and Password** fields, enter the credentials of someone who is an administrator of the machine. Lab Management uses these credentials to install the agents that will run the automated deployments and tests.

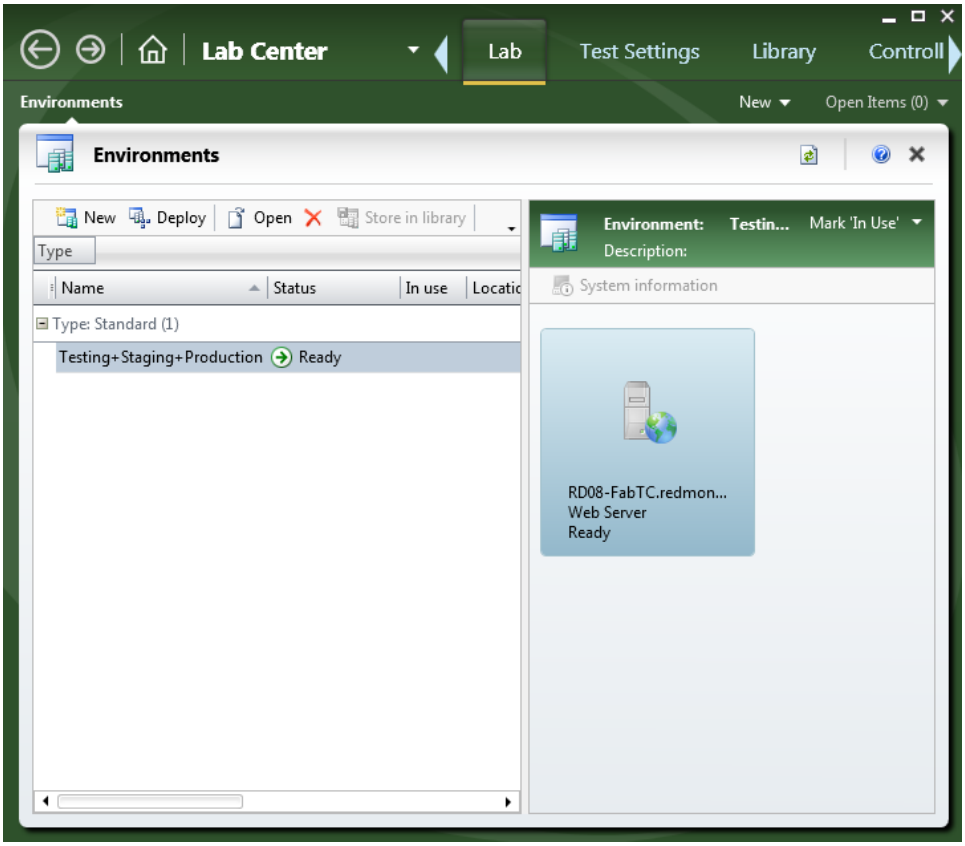
> **NOTE**: This is not the same person as the user who runs the agents. The user who runs the agents and performs the deployments and tests does not have to be an administrator, and is the user specified when you configured the test controller.



12. Click **Verify**.
13. If the verification succeeds, click **Finish**.



14. After a few moments, the agents are installed on the web server and the environment is available to the pipeline. The following screenshot shows an example of an environment.



Exercise 7: Installing the Test Artifacts (Optional)

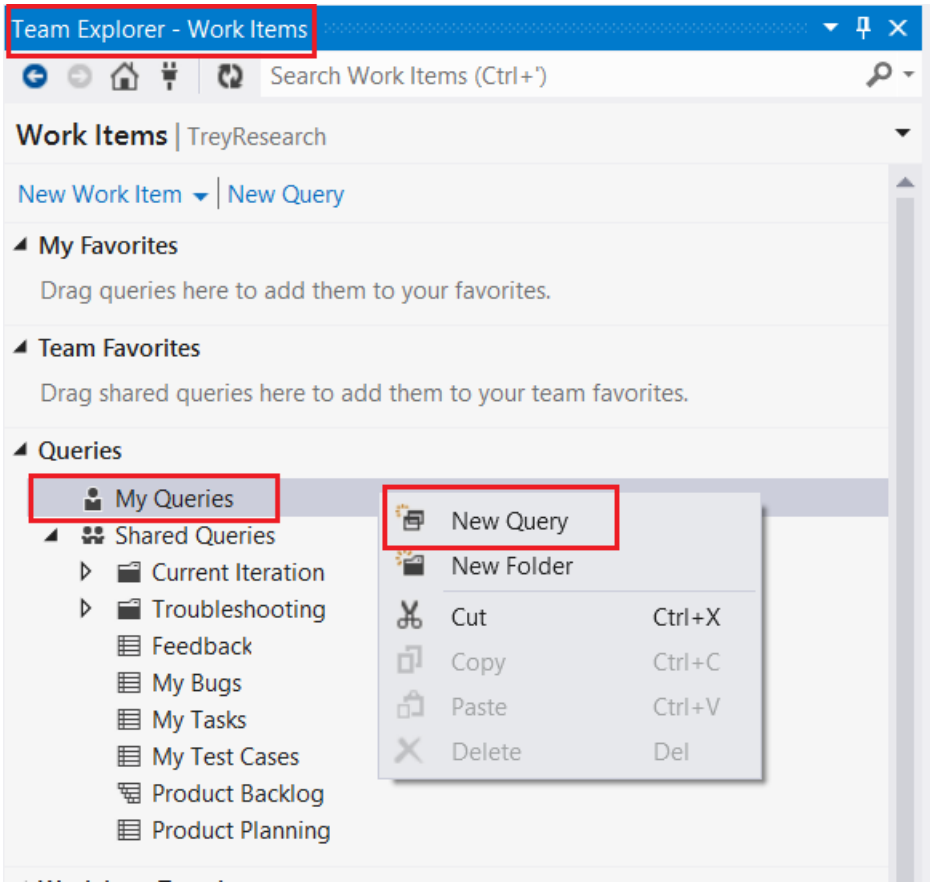
This exercise is optional, but it will help you to understand how the automation MTM works with the pipeline.

In this exercise you install the work item types (WIT) by copying items stored in a Microsoft Excel spreadsheet to a TFS Excel work form. If you're unfamiliar with TFS WITs, see How to: Create, Share, and Run Work Item Queries (Team System Web Access).

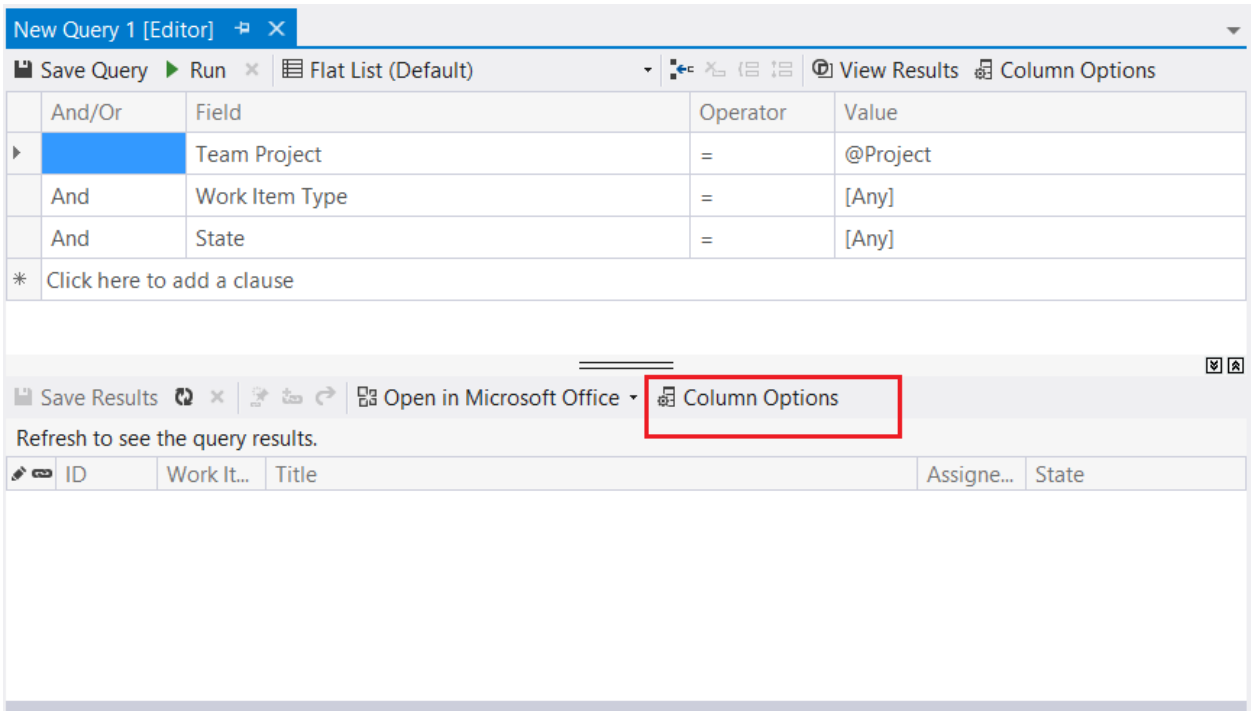
Task 1: Create an Excel TFS Work Items Types Excel Form

In this task you create a TFS Excel form that is used to organize the WITs. This is where you will store the items that are contained in the Excel spreadsheet Lab1_WIT.xlsl.

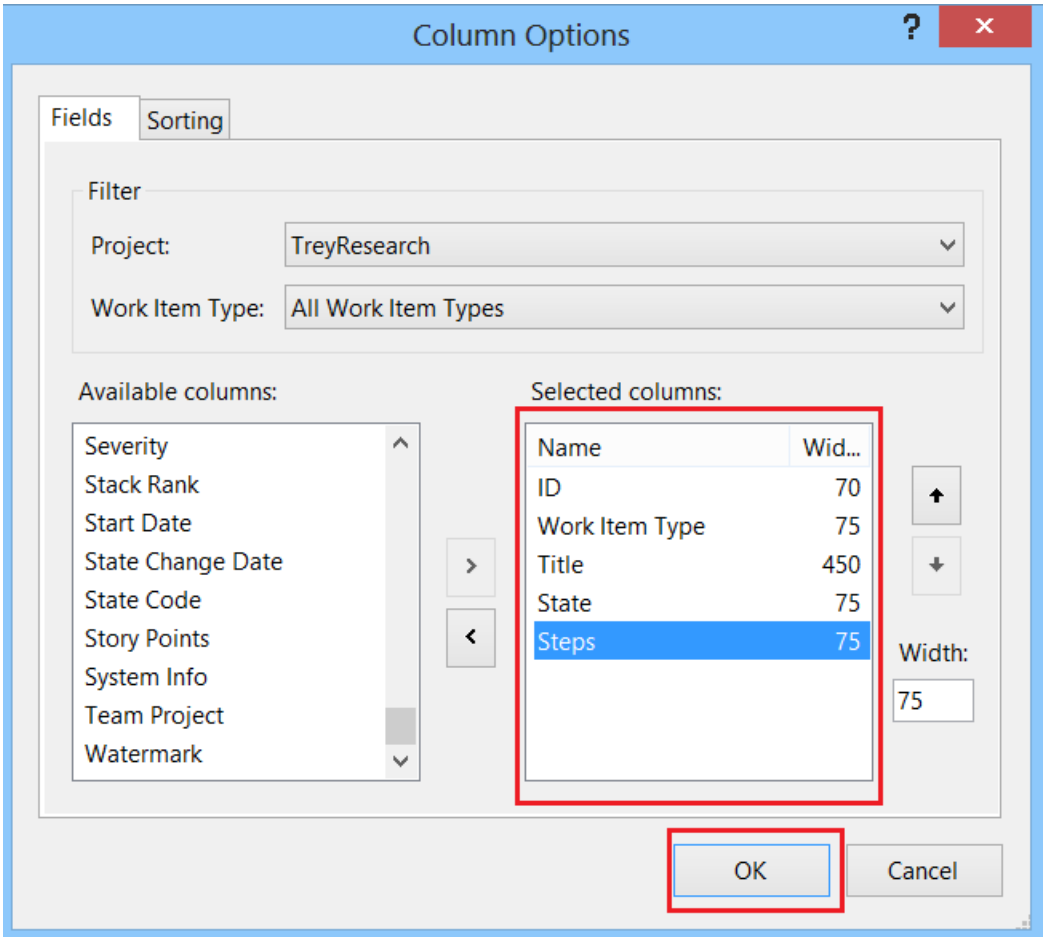
1. In Visual Studio Team Explorer, open the **Work Items** dialog. Right-click on **My Queries** and select **New Query**.



2. The **Query Editor** appears. Click **Column Option**s.

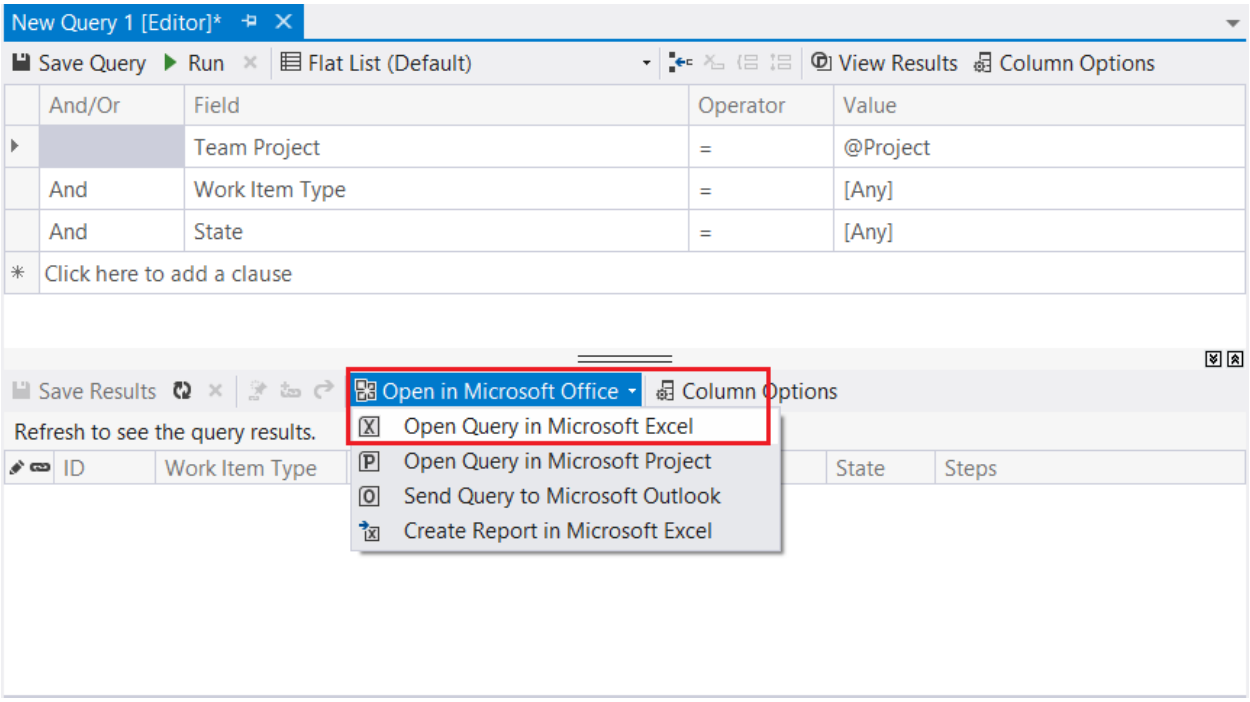


3. Select the following columns in the order listed below. When you are done, click **OK**.
    - ID
    - Work Item Type
    - Title
    - State
    - Steps

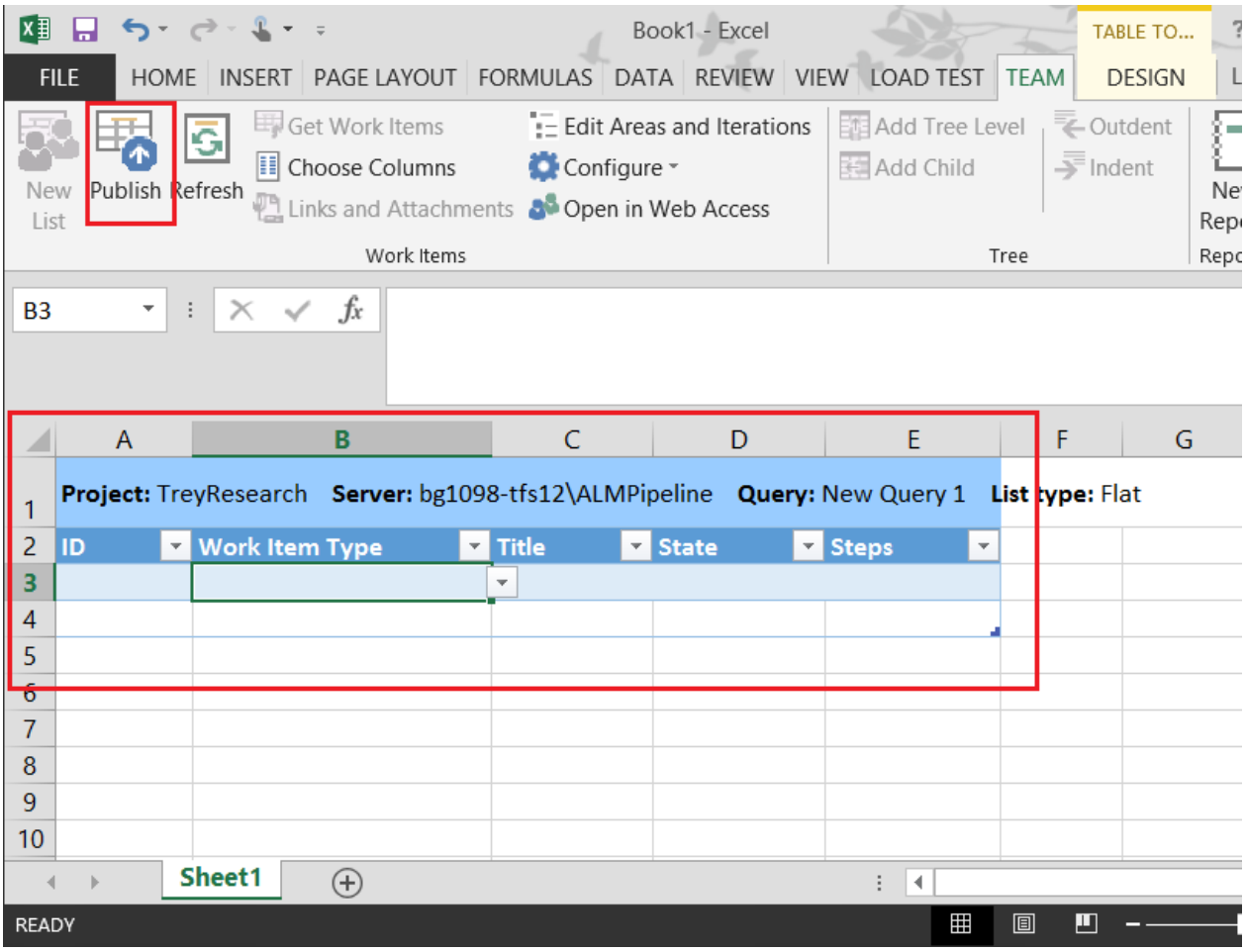


4. Click **Save Query**.

The Query Editor should now show the selected columns. Locate the **Open in Microsoft Office** combo box and select **Open Query in Microsoft Excel**.



5. Excel opens a form editor with the columns you selected. Note the **Publish** button. This will publish the data back to TFS once you add the data to the spreadsheet.



Task 2: Adding the WITs to the TFS Team Project

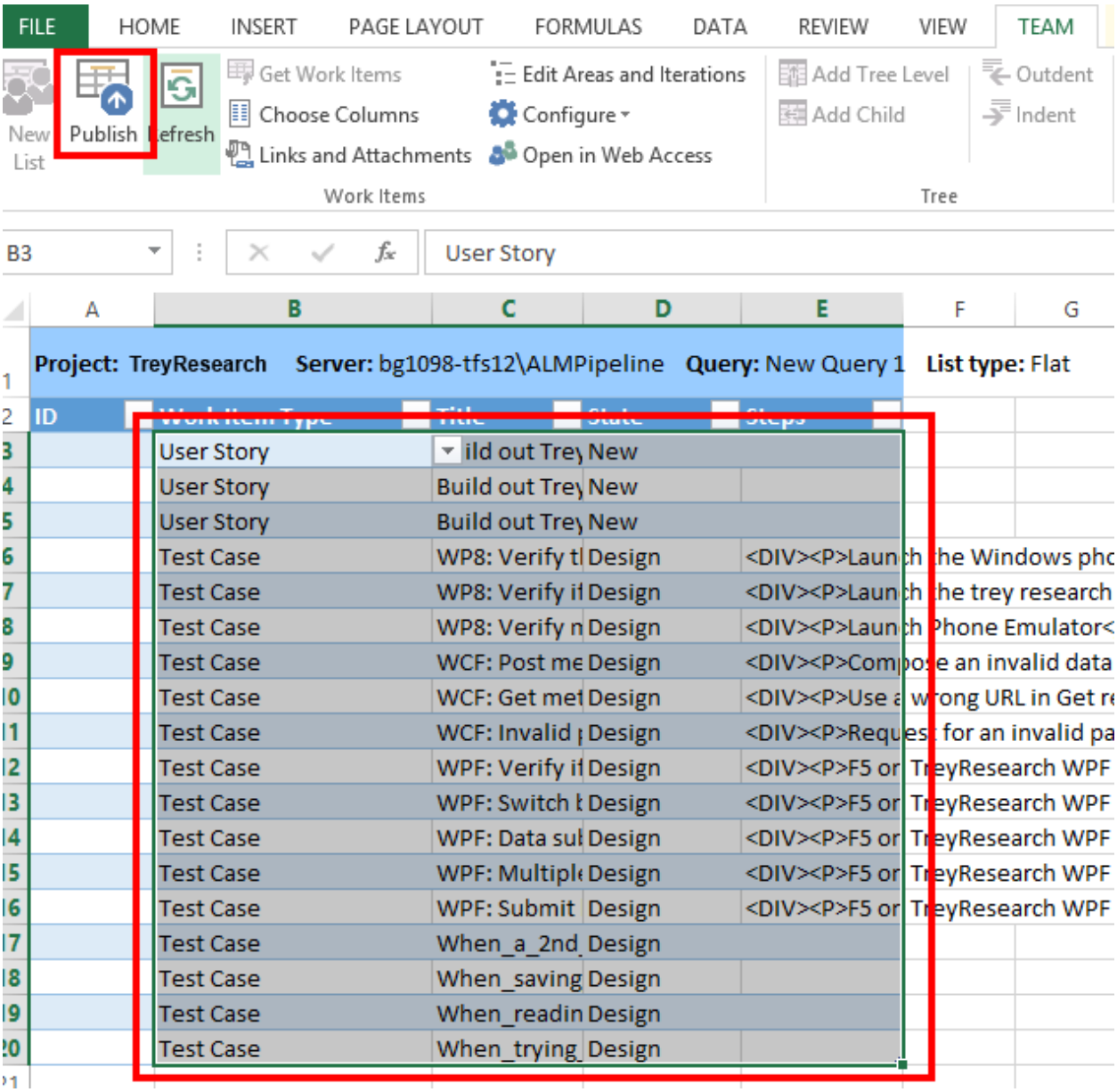
In this task you add the WITs from the Excel spreadsheet on your local machine to the TFS Excel form.

> **Note:** The action of importing steps into TFS has a bug that parses the steps incorrectly. The steps are imported as single step.

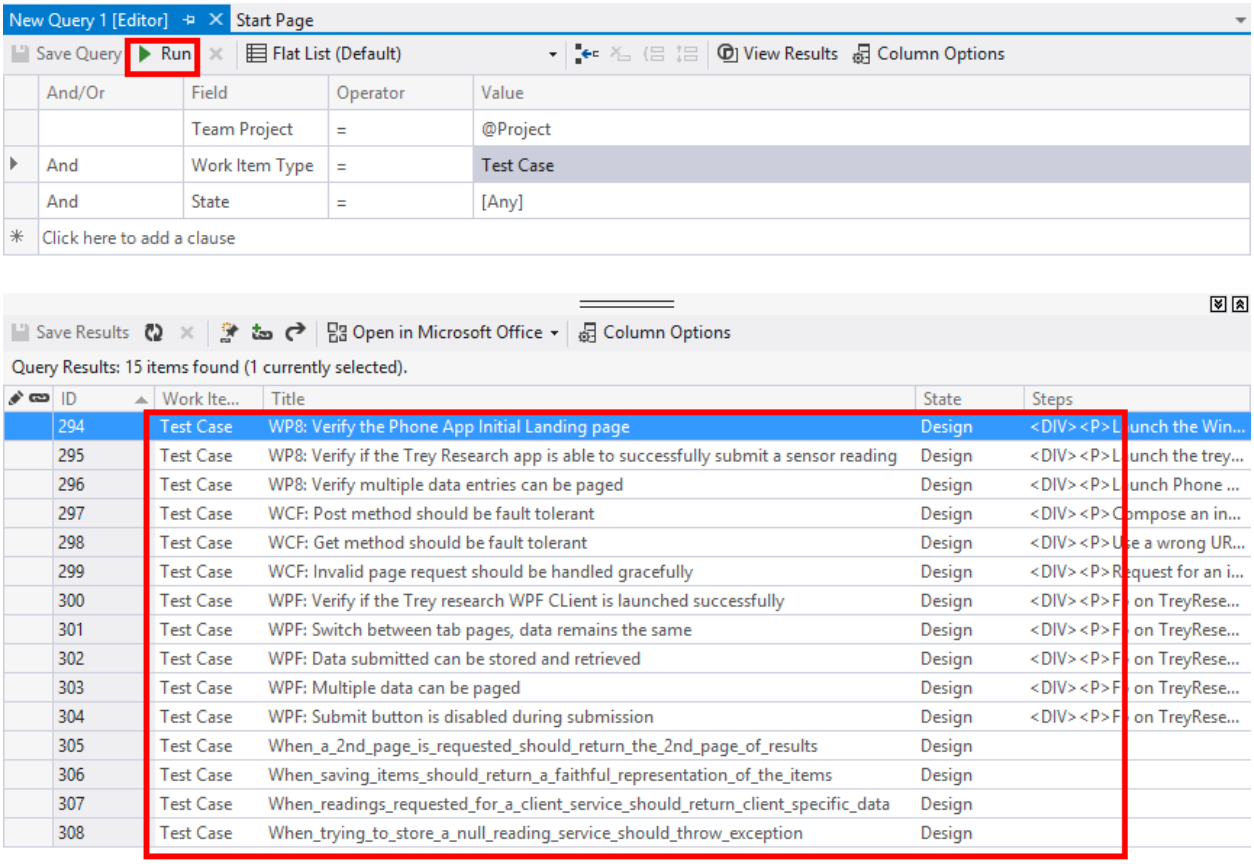
1. Open the Excel spreadsheet named Lab1_WIT.xlsx. It is located in the Lab01-StartingPoint directory. This spreadsheet contains the data for the user stories and the WITs for the test cases. Copy the data. Note that the order of the column values map to the ones in the TFS Excel Query Form worksheet. The following screenshot shows the data.

User Story	ild out Trey Phone App	New	
User Story	Build out Trey WCF Service	New	
User Story	Build out Trey WPF Client	New	
Test Case	WP8: Verify the Phone App Initial Landing page	Design	<DIV><P>
Test Case	WP8: Verify if the Trey Research app is able to successfully submit a sensor	Design	<DIV><P>
Test Case	WP8: Verify multiple data entries can be paged	Design	<DIV><P>
Test Case	WCF: Post method should be fault tolerant	Design	<DIV><P>
Test Case	WCF: Get method should be fault tolerant	Design	<DIV><P>
Test Case	WCF: Invalid page request should be handled gracefully	Design	<DIV><P>
Test Case	WPF: Verify if the Trey research WPF CLient is launched successfully	Design	<DIV><P>
Test Case	WPF: Switch between tab pages, data remains the same	Design	<DIV><P>
Test Case	WPF: Data submitted can be stored and retrieved	Design	<DIV><P>
Test Case	WPF: Multiple data can be paged	Design	<DIV><P>
Test Case	WPF: Submit button is disabled during submission	Design	<DIV><P>
Test Case	When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results	Design	
Test Case	When_saving_items_should_return_a_faithful_representation_of_the_iten	Design	
Test Case	When_readings_requested_for_a_client_service_should_return_client_spe	Design	
Test Case	When_trying_to_store_a_null_reading_service_should_throw_exception	Design	

2. Paste the copied values Into the TFS Excel form. Begin in the **Work Item Type** column, not the **ID** column.



3. Click **Publish**. The WITs are published to the TFS project.
4. In Visual Studio, refresh your query. The WITs you added appear.



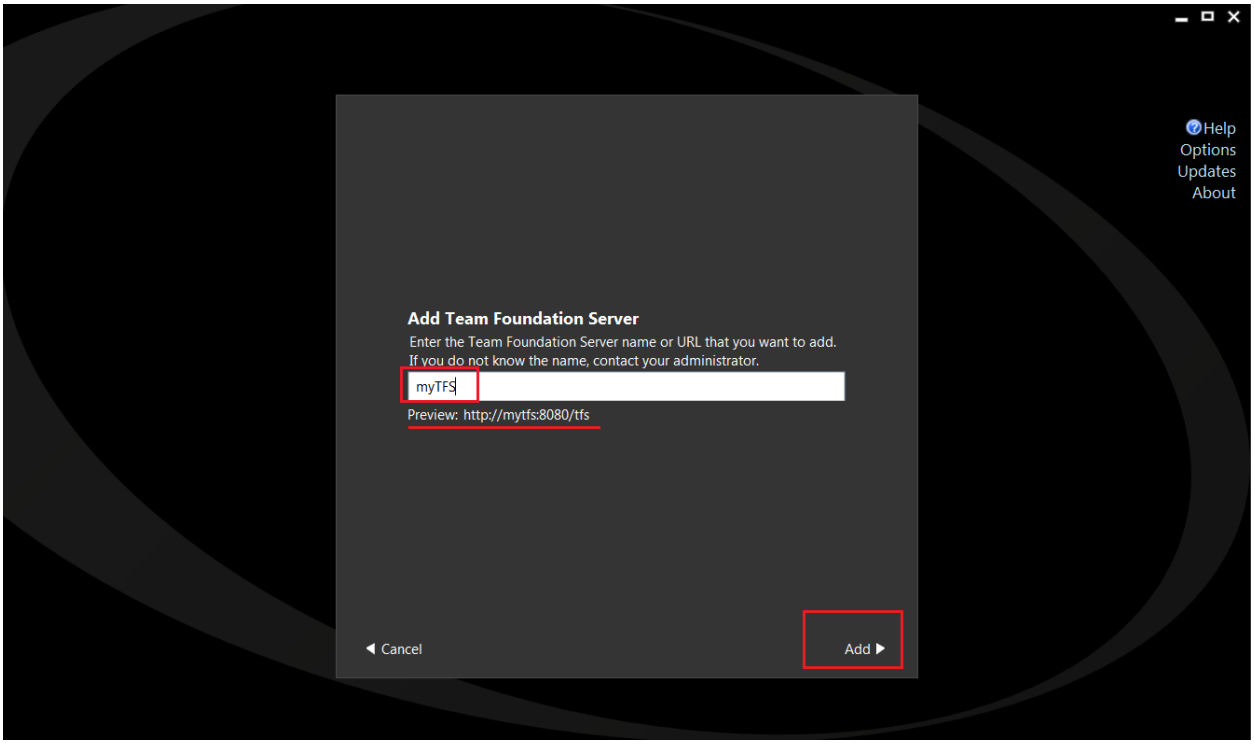
Exercise 8: Creating a Test Plan in Microsoft Test Manager (Optional)

This exercise is optional, but it will help you to understand how the automation and MTM work with the pipeline.

In this exercise, you create a Microsoft Test Manger (MTM) test plan and add test suites to it. MTM allows you to manage builds, test environments and test cases. If you aren't familiar with MTM, see Getting started with Lab Management. There is also some information about MTM in the Introduction document that is included with this guidance.

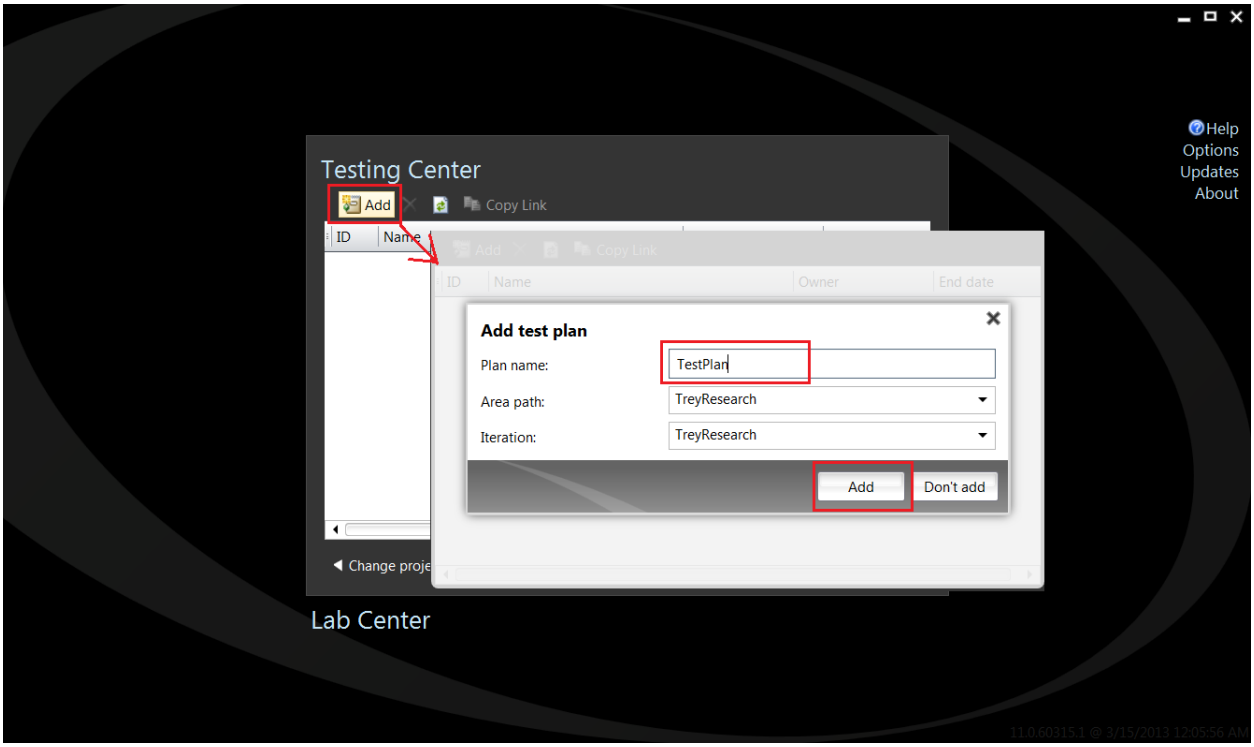
Task 1: Create the Test Plan

1. Open MTM and add the name of the Team Foundation Server that stores the TreyResearch team project. Click **Add**.

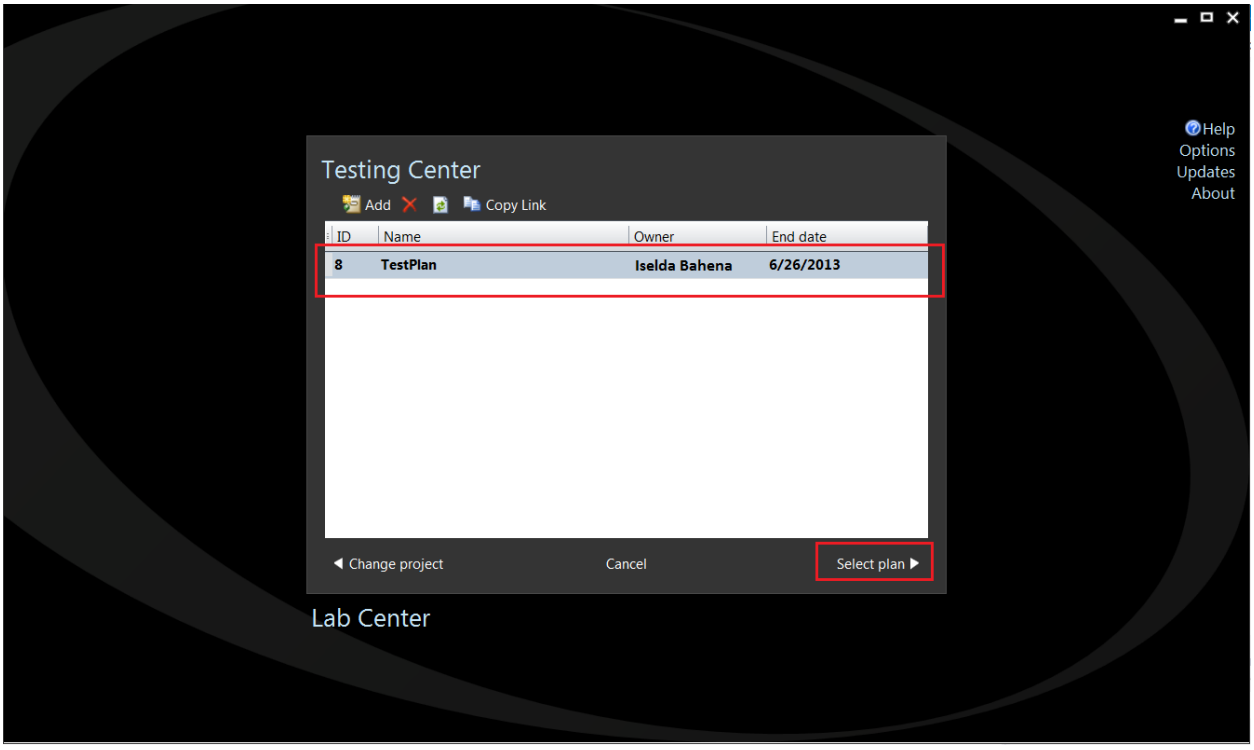


2. Select the **TreyResearch** team project. Click **Connect now**.

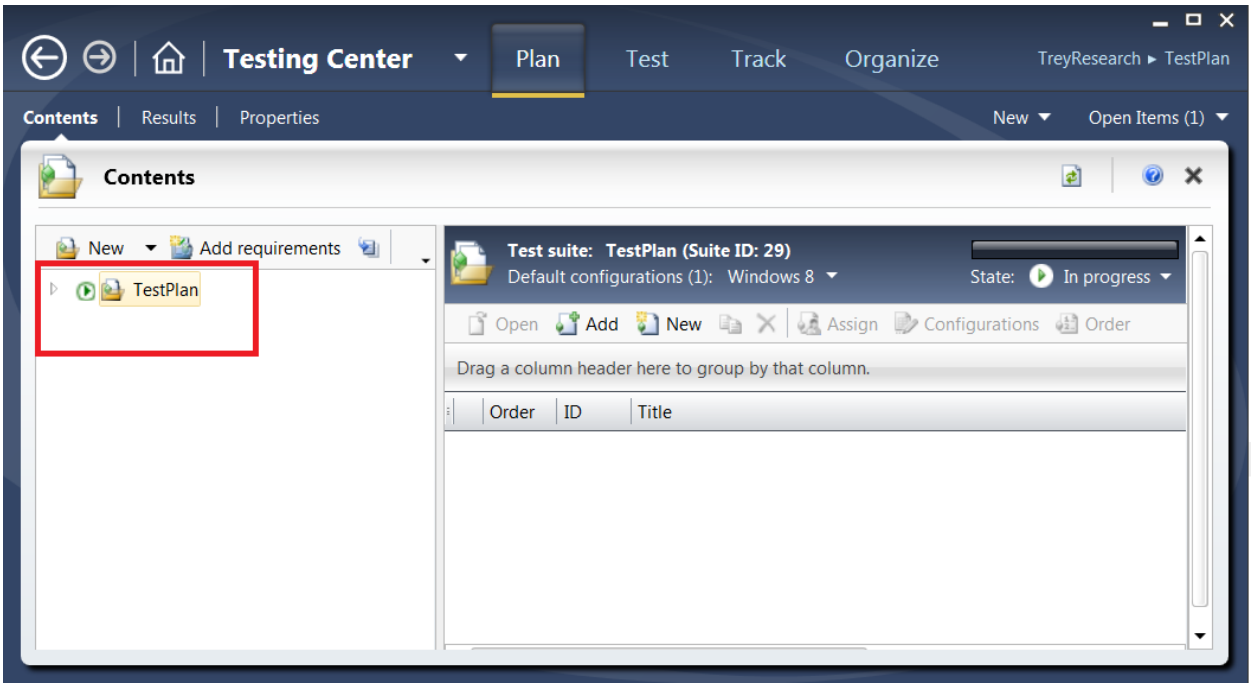
3. The **Testing Center** dialog box opens. Click **Add** to create a test plan. The **Add Test Plan** dialog box opens. In the **Plan name** box, enter **TestPlan**. Click **Add**.



4. Select **TestPlan**. Click **Select plan**.



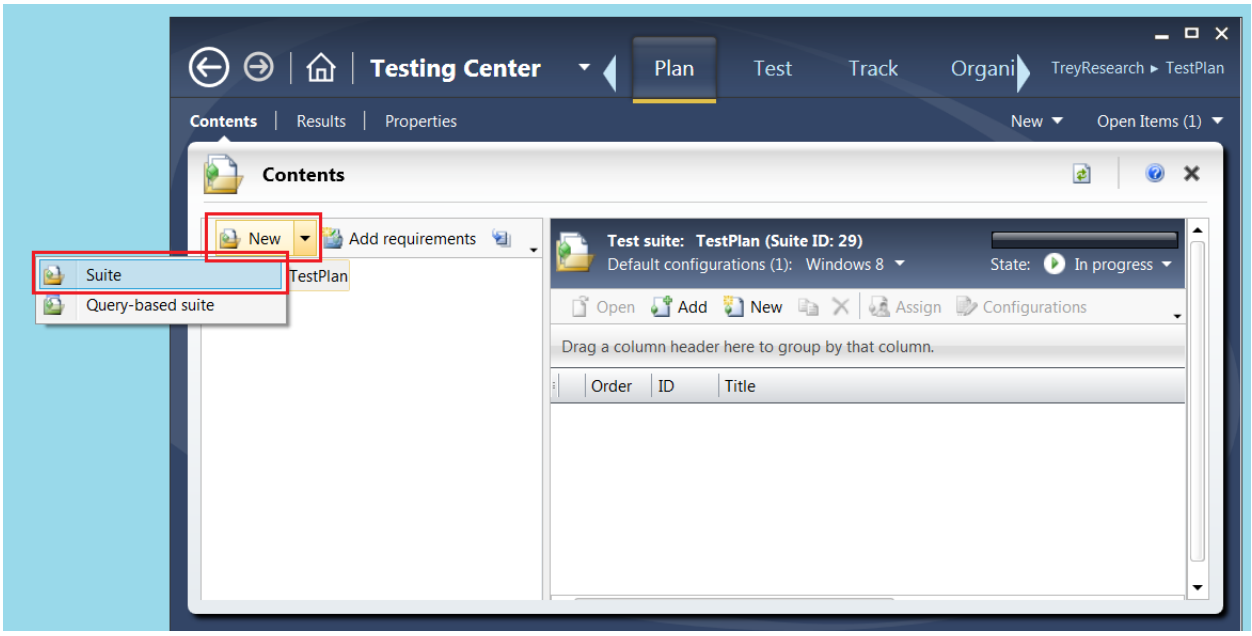
5. The Testing Center displays the new test plan. If you use the tree view, the top node is the **TestPlan** node.



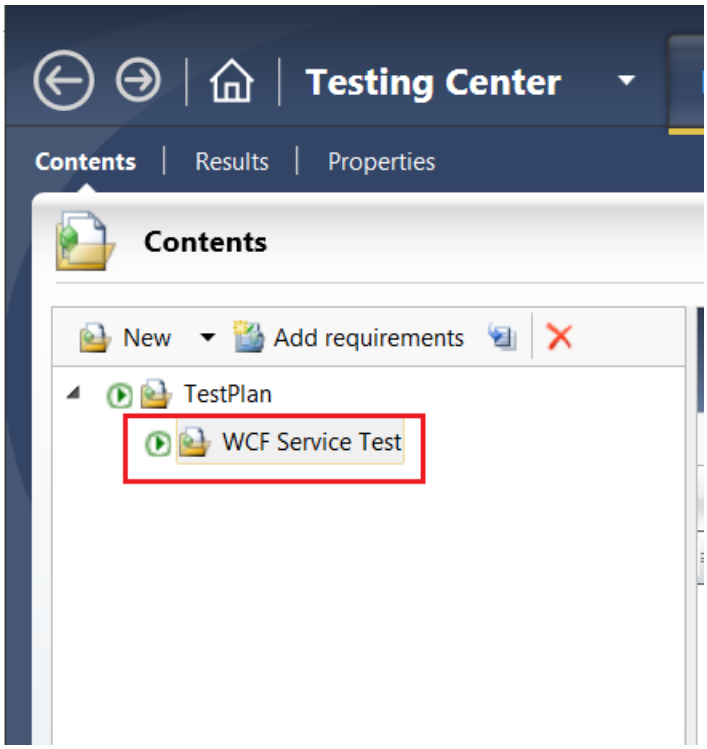
Task 2: Add Test Suites to the Test Plan

In this task, you add test suites to the test plan. Suites organize your test cases. They can be composed of both manual and automated test cases. For more information, see Quick Start Guide for Manual Testing using Microsoft Test Manager.

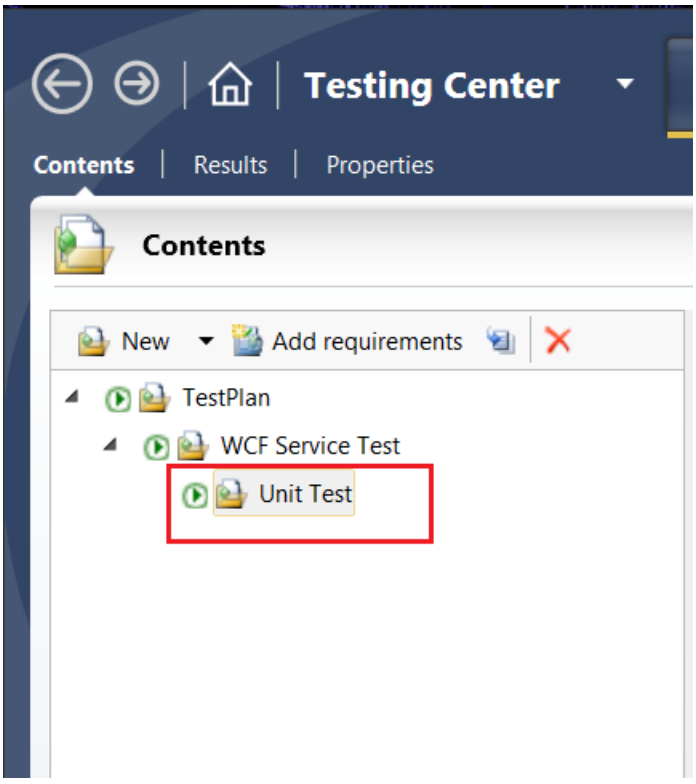
1. Select the **TestPlan** node. Click **New**. Select **Suite** from the drop-down list. A new suite node appears under **TestPlan**.



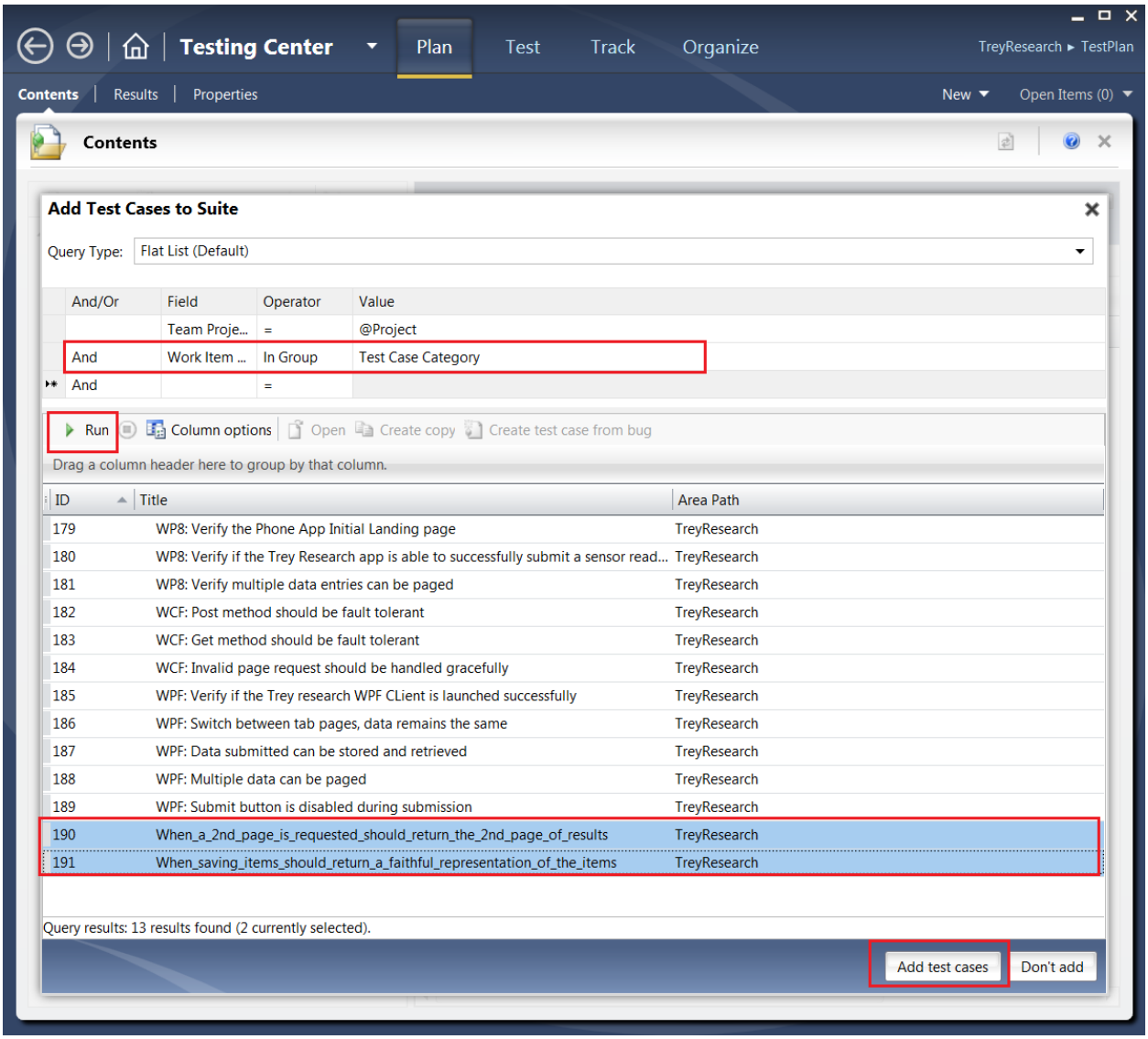
2. Name the suite **WCF Service Test**.



3. Select **WCF Service Test**. Right-click on it. In the context menu, click **New suite**. Name this suite **Unit Test**.

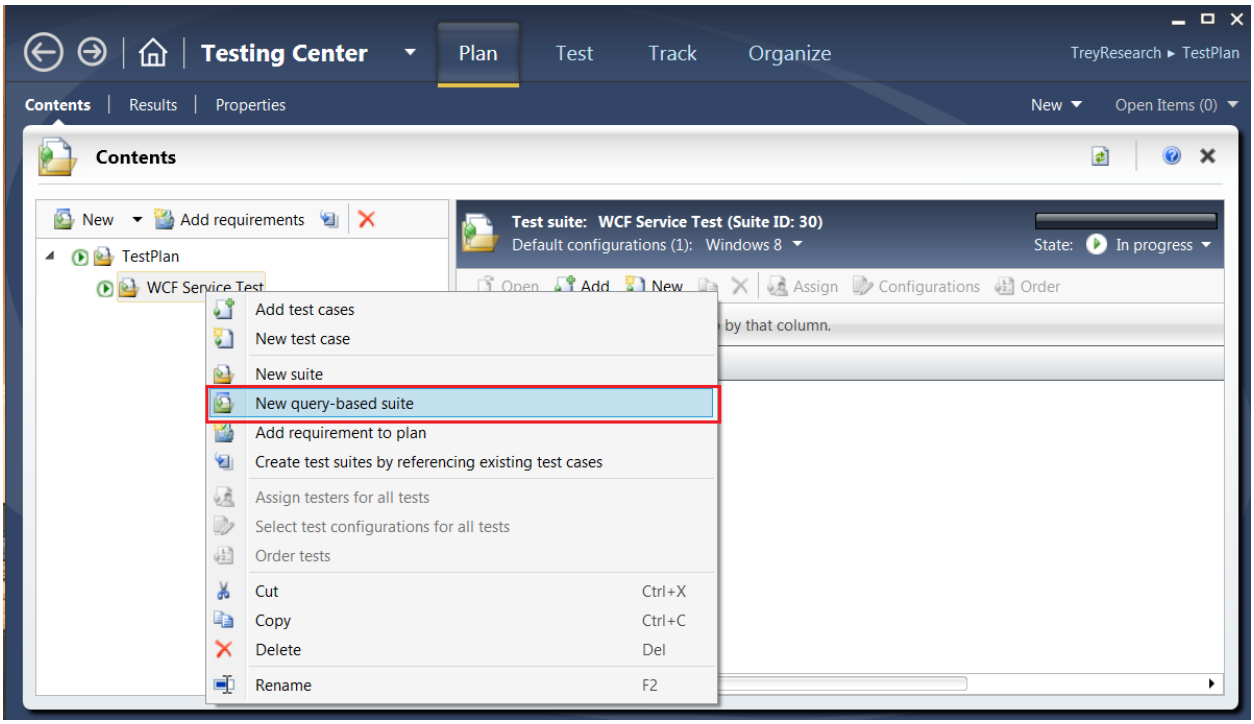


4. Right-click on **Unit Test**. Click **Add test cases**. The query window appears. Click **Run**. Because the filter is set to **Test Case Category**, the test cases you added earlier are visible. Select the two test cases without a WPF: a WP8: or a WCF: in front of the name. Click **Add test cases**. You return to the Testing Center page.

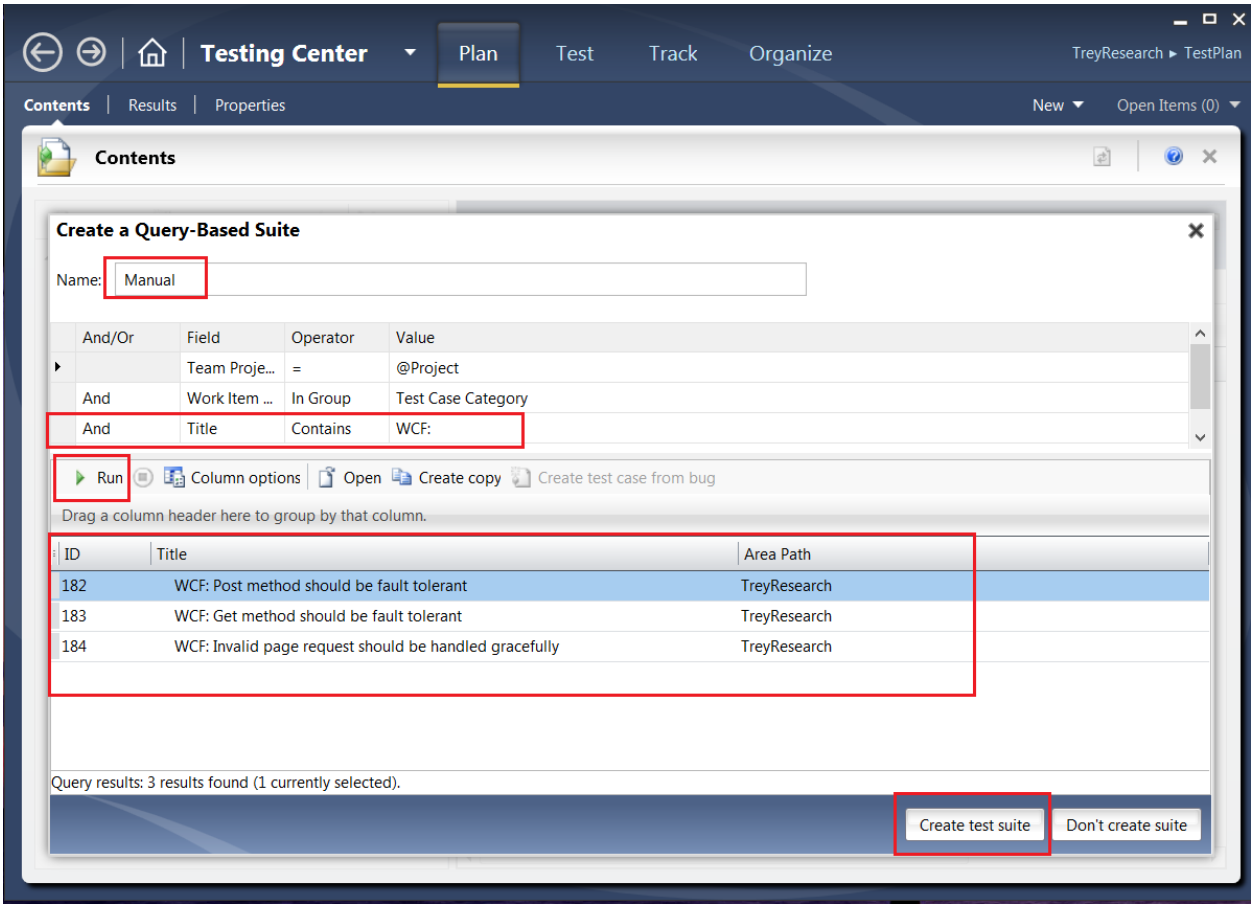


5. Select **WCF Service Test**. Right-click on it. In the context menu, select **New query-based suite**. The query edit form appears.

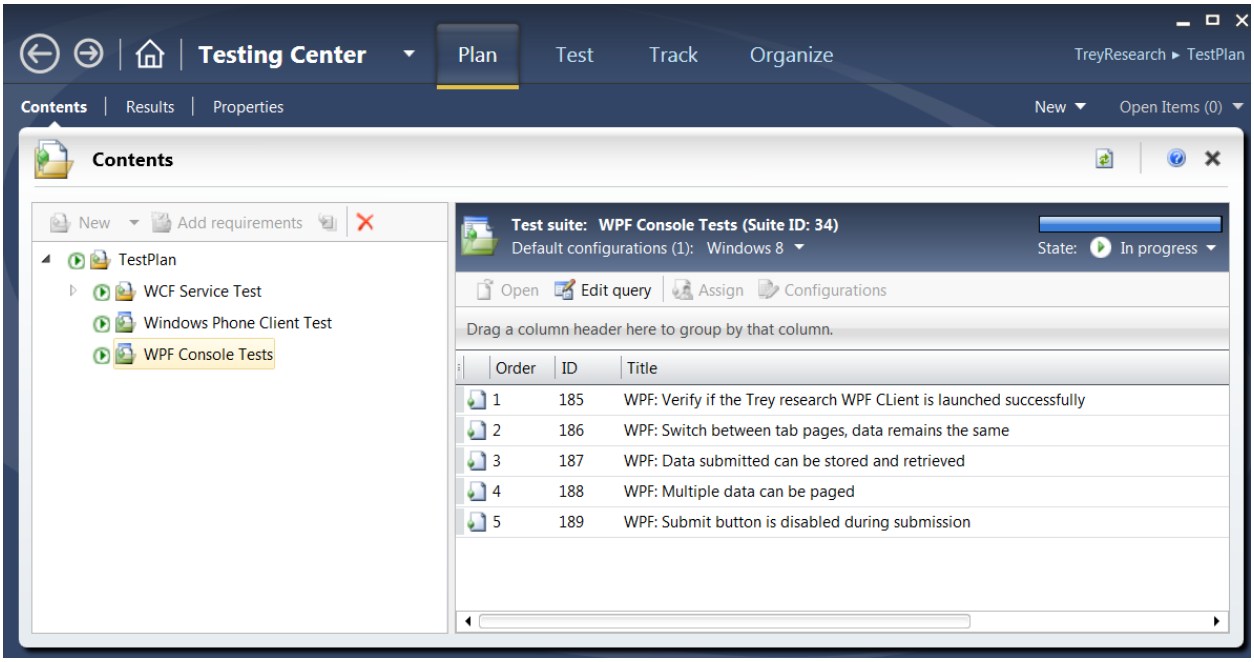
> **Note:** Query-based suites allow you to actively query TFS for new or changed test cases. Every time you open your test plan, it queries TFS for any changes.



6. In the query editor, in the **Name** text box, enter **Manual**. For the filter, use **And Title Contains WCF:**. Click **Run**. All the test cases with **WCF:** in the title appear. Click **Create test suite**.



7. Select the **TestPlan** again. Add a query-based suite named **WPF Console Tests**. Repeat step 6 but the filter is now **And Title Contains WPF:** Click **Create test suite**.
8. The test plan should now look like the following screenshot.



Summary

In this lab you set up everything you will need for the labs that follow. You first installed the Trey Research solution, created the TreyResearch team project, and then added the solution to the project. You then built the solution and ran the unit tests.

In the next exercise, you published and validated the WCF service web role to IIS. You then deployed the WPF application.

In the third exercise you installed the test artifacts. You added items in an Excel spreadsheet to a TFS Excel form.

In the fourth exercise you created a simple build definition for a pipeline that performs continuous integration and runs unit tests. In the fifth exercise, you created a test plan in MTM, added test cases to the test suites, and added the test suites to the test plan.

Copyright