Skytap Automation Pack for Visual Studio Team Foundation Server

Skytap's Automation Pack for Visual Studio Team Foundation Server (or TFS Automation Pack) enables easy integration of your Skytap environments with TFS to run Continuous Integration and Automated Testing workflows using Build-Deploy-Test (BDT) concepts in TFS. BDT is the workflow engine that ships with TFS that enables a developer or tester to:

- Build their application code
- Deploy components of their application
- Define and choose the test environment the application will be deployed in
- Define and deploy automated tests for their application
- Collect results and other diagnostics data during test runs

Once defined, this BDT workflow can be repeated multiple times by members of the development team.

> The automation pack is available for download here (http://www.skytap.com/downloads/SkytapTFSv2.zip). For instructions on installing and using the pack, see the instructions below.

How to Set up and Use the Skytap Automation Pack for Visual Studio Team Foundation Server

This article will:

- Guide you through the prerequisites for using the TFS Automation Pack
- Show you how to install the TFS automation pack
- Walk you through setting up a test environment in Skytap that is ready for use in TFS
- Give an example BDT workflow with TFS Automation Pack

Prerequisites

There are a few prerequisites you must have in order to run a Skytap-integrated BDT workflow, as listed below:

Team Foundation Server 2012

You need to have TFS 2012 installed. This can either be on-prem or in Skytap itself. For the purposes of this article, we will assume TFS is installed on-prem. As part of TFS install you should have the following components of TFS also installed:

- **Build Controllers:** This is a distributed component of TFS that helps run a build as defined in your build definition. You can have it installed on a machine in the same location as your TFS server. At a minimum, you will need it configured with 2 agents. For more information on installing the TFS Build Service, see http://msdn.microsoft.com/en-us/library/ee259687.aspx (http://msdn.microsoft.com/en-us/library/ee259687.aspx)
- **Test Controllers:** This is a distributed component of TFS that helps deploy and execute tests as defined in your BDT. You can have it installed on a machine in the same location as your TFS server. For information on setting up test controllers in lab environments, see http://msdn.microsoft.com/en-us/library/hh546460.aspx (http://msdn.microsoft.com/en-us/library/hh546460.aspx). For setting up test controllers and test agents to manage tests with Visual Studio, see http://msdn.microsoft.com/en-us/library/hh546459.aspx (http://msdn.microsoft.com/en-us/library/hh546459.aspx).

VPN Connection to Skytap

In order for your TFS server to communicate with test environments in Skytap, you should have a VPN connection to Skytap configured. For more information, see Creating a VPN Connection to an Outside Network (Vpns.html).

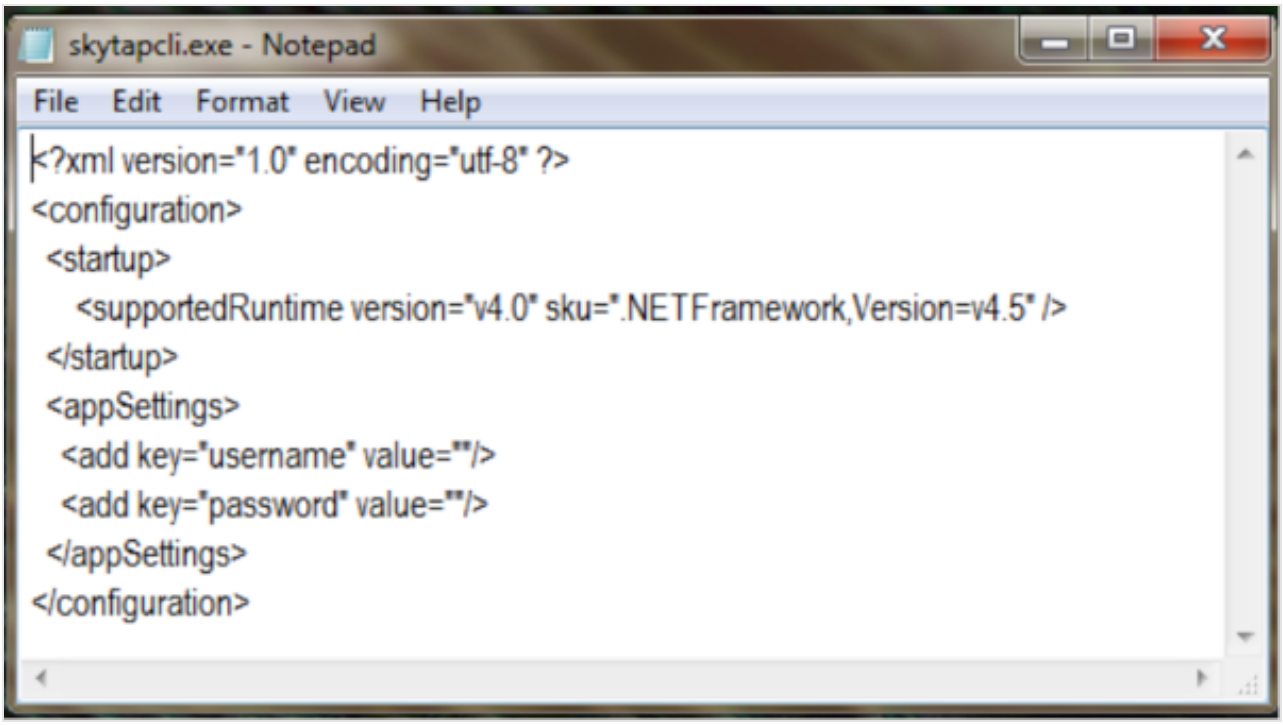
> If you have TFS installed in Skytap itself, then you don't need a VPN. Connection between TFS and test environments in this case is created using Skytap ICNR. For more information, see Networking Between Environments (Networking_Between_Environments.html).

Skytap Automation Pack for TFS

This is the automation pack provided by Skytap that enables you to employ Skytap in your BDT scenarios. You can download the pack here (http://www.skytap.com/downloads/SkytapTFS.zip). The pack has 2 components:

- **SkytapLabDefaultTemplate.11.xaml:** A template workflow for TFS BDTs that lets you define the build, deploy and test tasks. It also integrates with Skytap to create, start, stop, suspend, save, and delete the test environments. It also integrates with Skytap's auto networking features to dynamically connect your test environments to your TFS environment, whether it's on-prem or hosted in Skytap. This template workflow needs to be checked-in to the BuildProcessTemplates folder in your project.

- **SkytapCLI.exe:** This is the workhorse that drives actions requested by the BDT workflow in Skytap, using Skytap's REST APIs. SkytapCLI.exe needs to be installed on the TFS build controllers in the "C:skytap" directory. You can provide a default set of Skytap credentials that can be used for all user requested BDT runs using the Automation Pack. You can do this by adding those credentials to the SkytapCLI.exe.config file, as shown in the picture below:



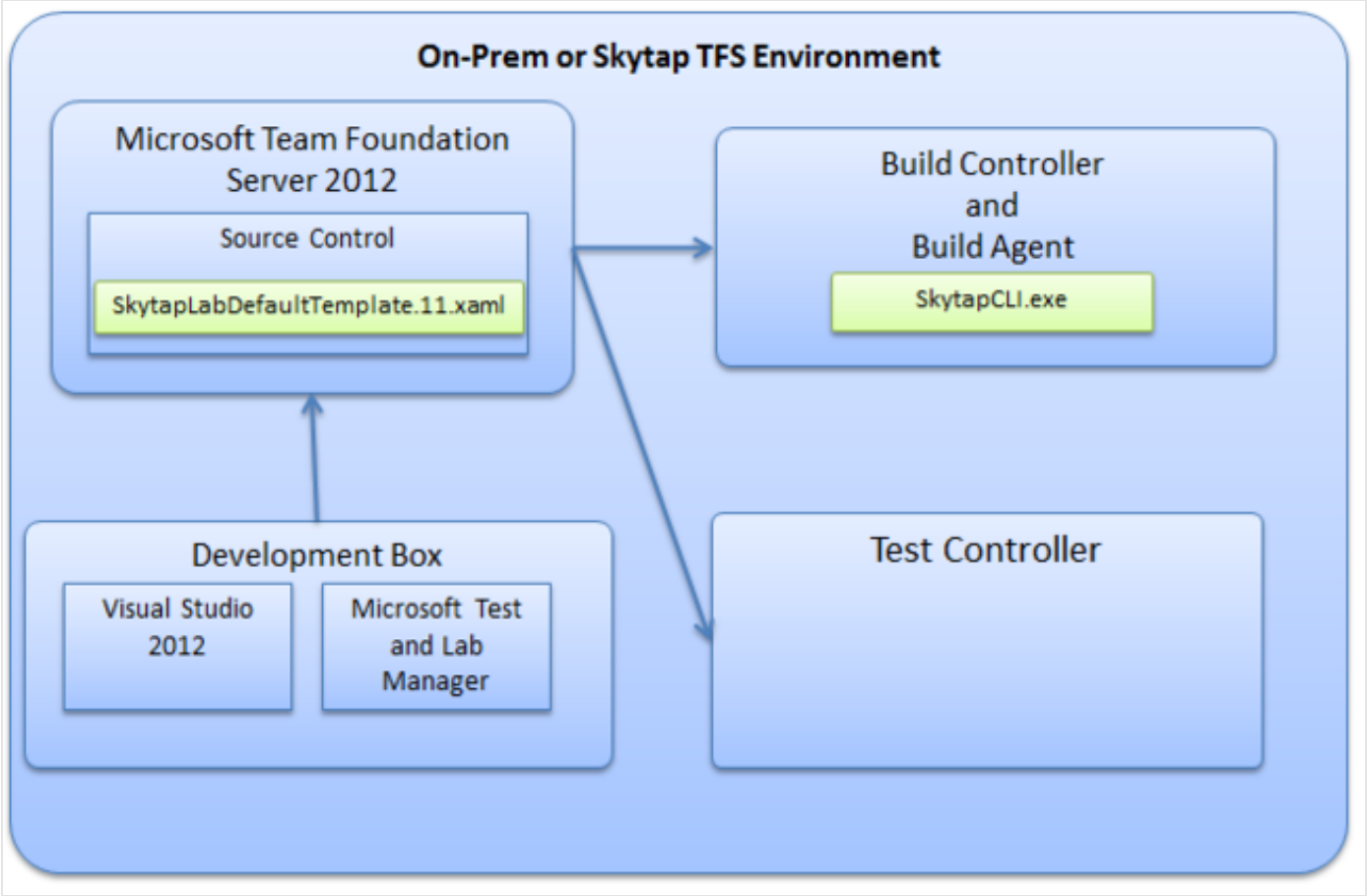
These credentials can be overridden when defining the BDT.

Target test template in Skytap

This is the Skytap template that you've designated as the test environment for your application. Each run of a BDT will create a fresh environment out of this template and perform tests against it. To prep this environment, you must perform these two steps:

1. Define the topology of your test environment in Skytap. This includes the hardware specifications of the VMs (such as CPU, RAM, and disk size), any software that needs installation, and any networking needed by the application. For a guide on making these changes to a VM, see Customizing Your Environment.
2. Convert your Skytap environment into a 'TFS Environment'. You'll need to use Lab Manager in Microsoft Test Manager 2012 to define your Skytap environment as a TFS test lab. This will be covered in detail in the 'How to set up your test environment in Skytap' section, below.

Once the prerequisites are in place, your on-prem infrastructure look something like the figure below:



How to set up your test environment

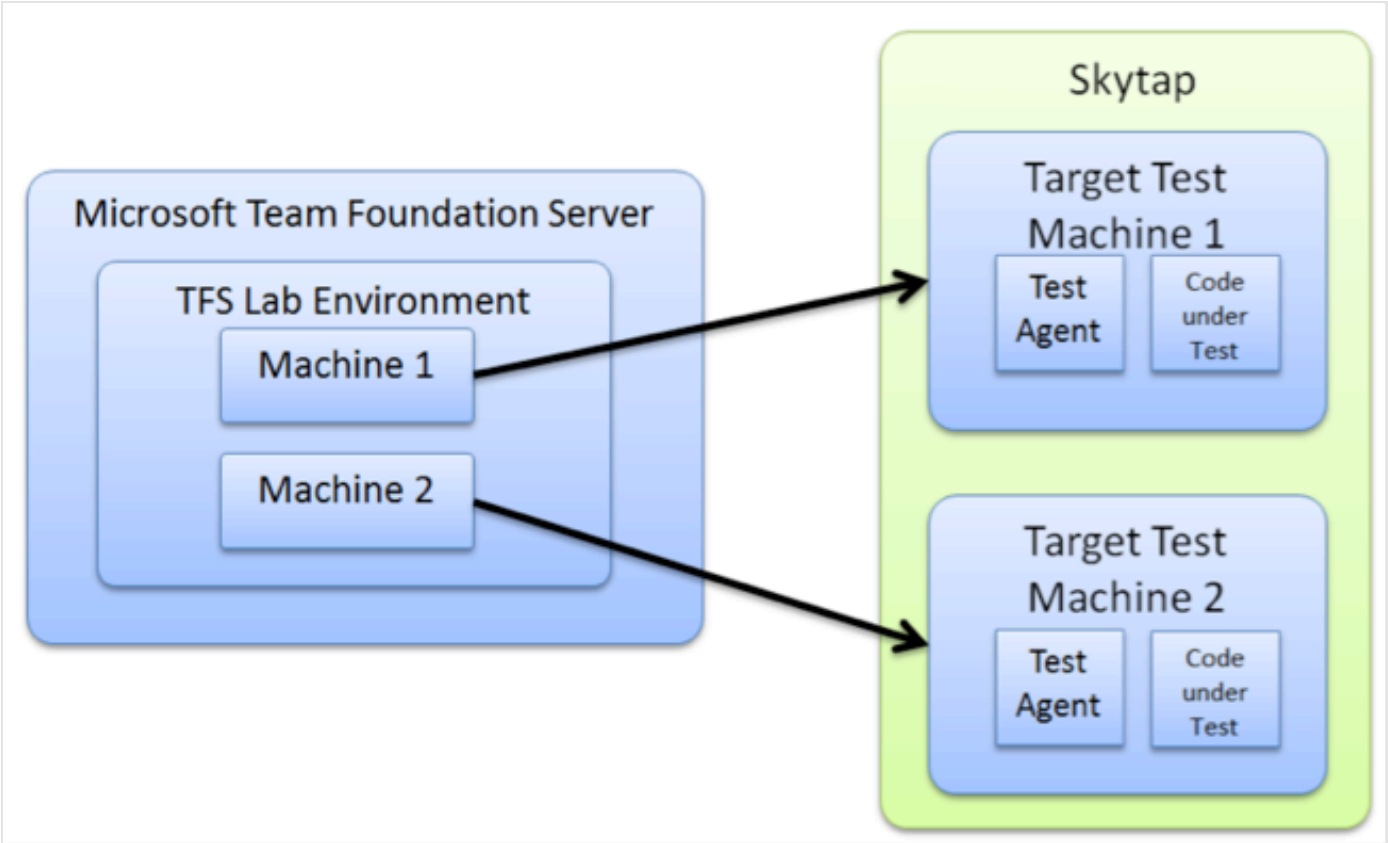
This section details how to create a Skytap environment that represents the topology of your test environment and designate it as a TFS test environment. You can use Skytap's self service features to:

- Create an environment (Creating_an_Environment.html)
- Add the desired VMs to it (Adding_and_Deleting_VMs.html), either starting from public templates or by importing your own VMs.
- Configure the hardware associated with each VM (Hardware.html), e.g. RAM, CPU, disk size, etc, as needed for your application
- Provision the networks and connect them to the VPN connection (Vpns.html) back to your on-prem network.

Once the environment has been modified to your satisfaction, you are ready for the next step to convert it into an environment that TFS can recognize.

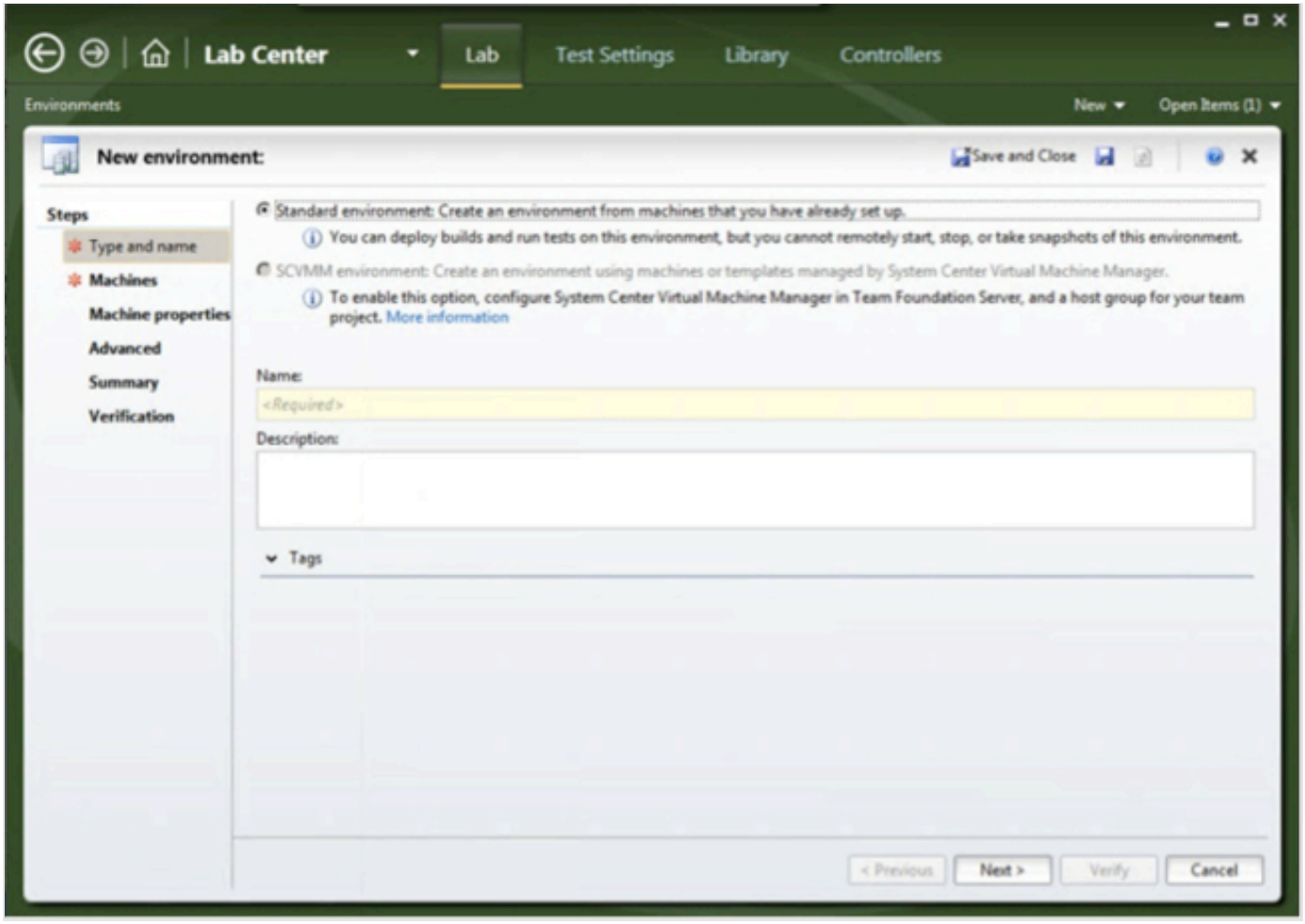
Convert the Skytap Environment into a TFS Environment

This step is needed so that the required agents can be installed and other tweaks can be made to the test VMs, such that TFS can recognize them as VMs when BDTs are run. The relationship is depicted in the conceptual diagram below:

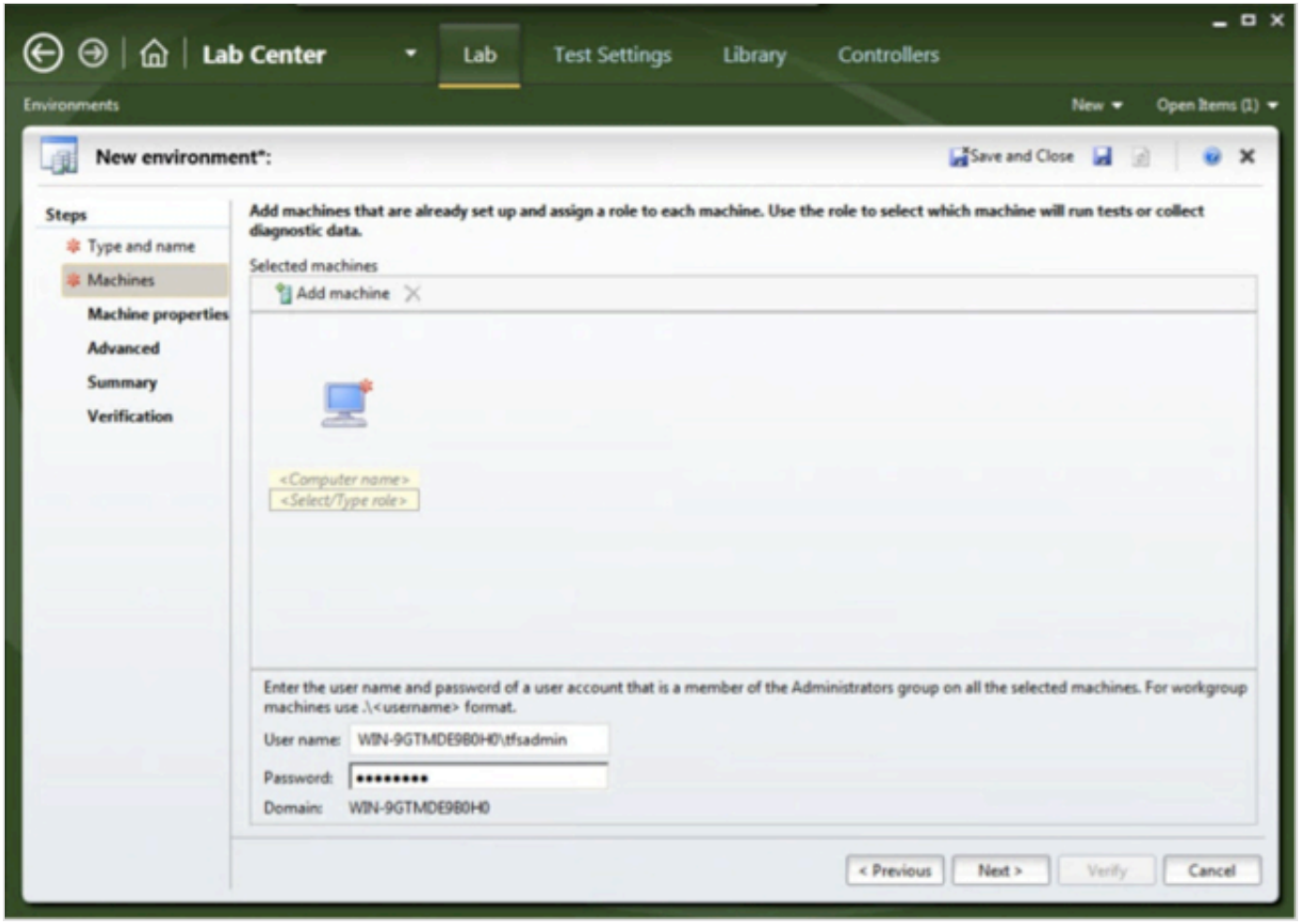


The following steps will guide you in creating this relationship using Lab Center components of Microsoft Test Manager 2012.

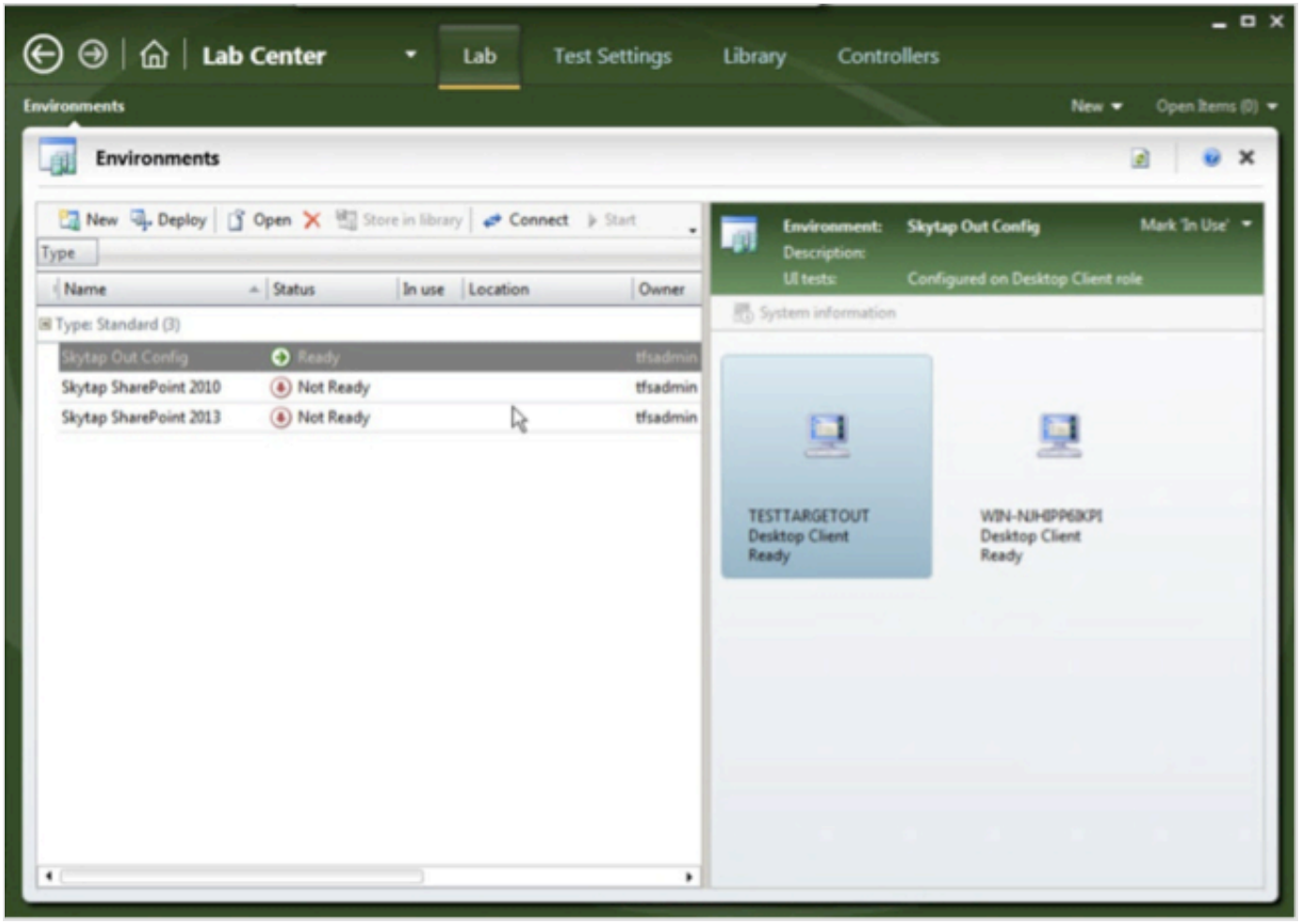
1. Ensure that your test environment is currently running in Skytap, as it must be running in order to create a TFS environment from it.
2. Open up Lab Center and create a new environment. Standard environments in Lab Center enable connection and deployment of the TFS Test Agent to any addressable machine. We use this to preconfigure the test target machines in Skytap.



3. In the ‘Machines’ section, start adding the VMs that are in your test environment. Use the IP addresses automatically provided by Skytap for each of your machines to add them in to this UI. This address is available from the Environment page containing the VM.



4. TFS requires that both the target test machines and the test controllers are able to address each other by name. Since the controller and test machines could potentially be in different domains, we will implement a workaround to achieve this; add the following entries into the host files (%systemroot%system32driversetchosts)
    1. The client that is running Microsoft Test Manager needs a hosts file to see all of the test target machines.
    2. Entries in the hosts file of test controller for all the target test machines.
    3. Entries in the hosts file of the target test machines for all the test controllers.
5. TFS also requires a shared account on all of the test controllers, build controllers and test agents. This should be a local account on all machines with the same username/password combination. The account should also be a member of the local administrators group on all of those machines.
6. Microsoft Test Mananger will go through its setup process and convert the Skytap environment into one that can be managed by TFS.
7. Once the TFS lab environment is in the ready state, which means that all machines are configured correctly, save the Skytap environment as a Skytap template (see Saving an Environment as a Template (Templates.html)). This template will be used by the Automation Pack during the BDT workflow to create the TFS Environments in Skytap.



At this point you are ready to create run BDTs.

Define a BDT in TFS

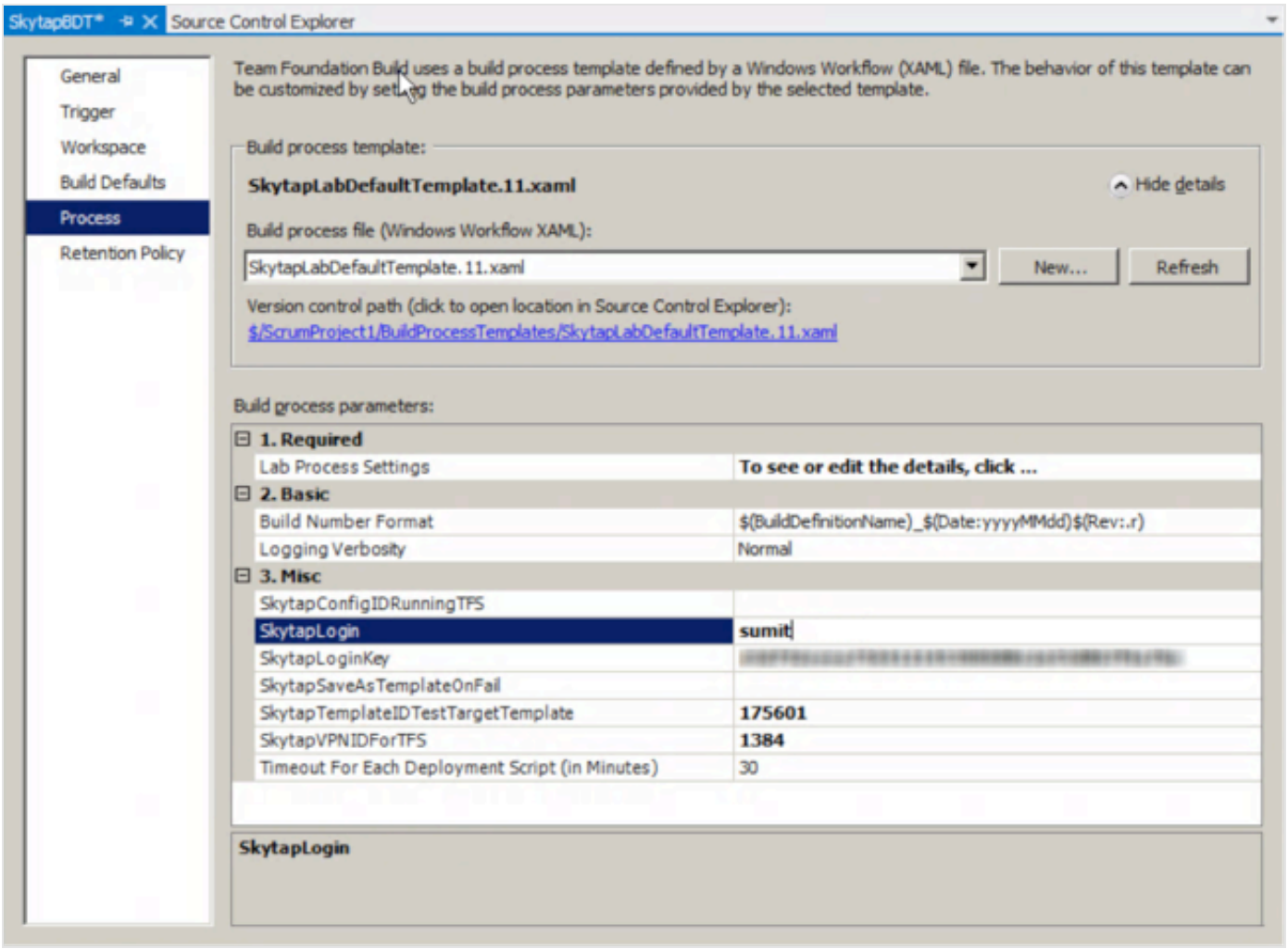
You may be familiar with the steps to create and deploy BDTs in TFS using Visual Studio and Microsoft Test Manager. We will use the same steps to illustrate how to create and run BDTs using the Automation Pack under the hood.

A developer or tester can create a BDT definition based on the template provided as part of the Automation Pack. The only difference is that instead of using default lab template you will use SkytapLabDefaultTemplate.11.xaml as the 'Build process file' . This template preserves the same UI and workflow as existing BDT templates. It also allows the user to enter the Skytap-specific information needed for the integration. Here's a brief description of Skytap specific fields that you can enter in this UI. Only two of these values is required.

- SkytapLogin: Your login id for Skytap. If you choose to use a centralized Skytap user for all your automation on Skytap, you can define that in SkytapCLI.exe.config on the build controller. SkytapLoginKey: Your password or API key, if token based API Authentication is turned on for your account.
- SkytapSaveASTemplateOnFail: True directs the Automation Pack to save the test environment as a

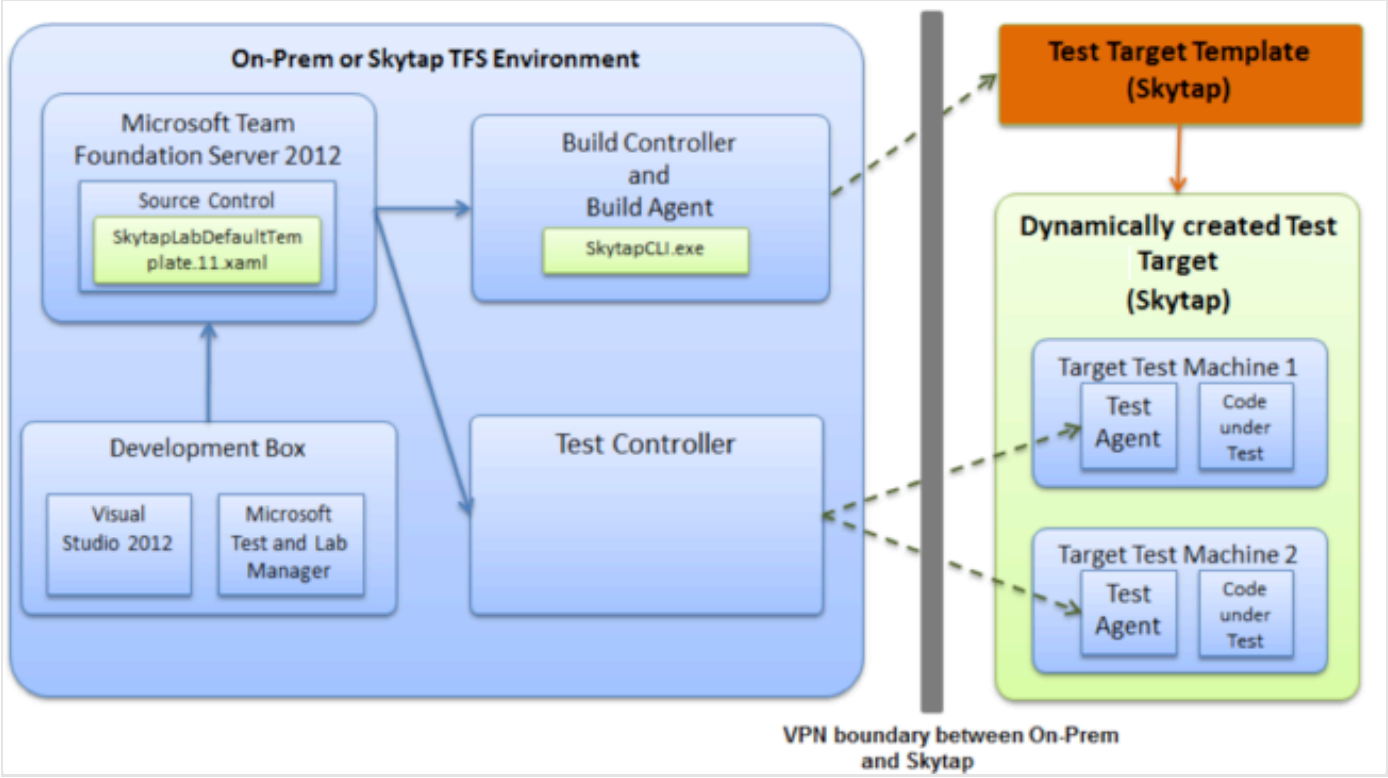
template in Skytap, in case there are failures encountered in the BDT run.
SkytapTemplateIDTestTargetTemplate: (Required) Template ID of the of the 'golden' test environment that was created in Skytap and registered as a TFS Environment.

- SkytapVPNIDForTFS (required): VPN ID of the VPN connection between Skytap and your on-prem network. You can get this ID from your Skytap Administrator.
- SkytapConfigIDRunningTFS: This field is useful if your TFS server is running in another environment in Skytap as opposed to running on-prem. In this case you need to provide the ID of the environment that the TFS server resides in. In this scenario, this field is required and SkytapVPNIDForTFS is not required.



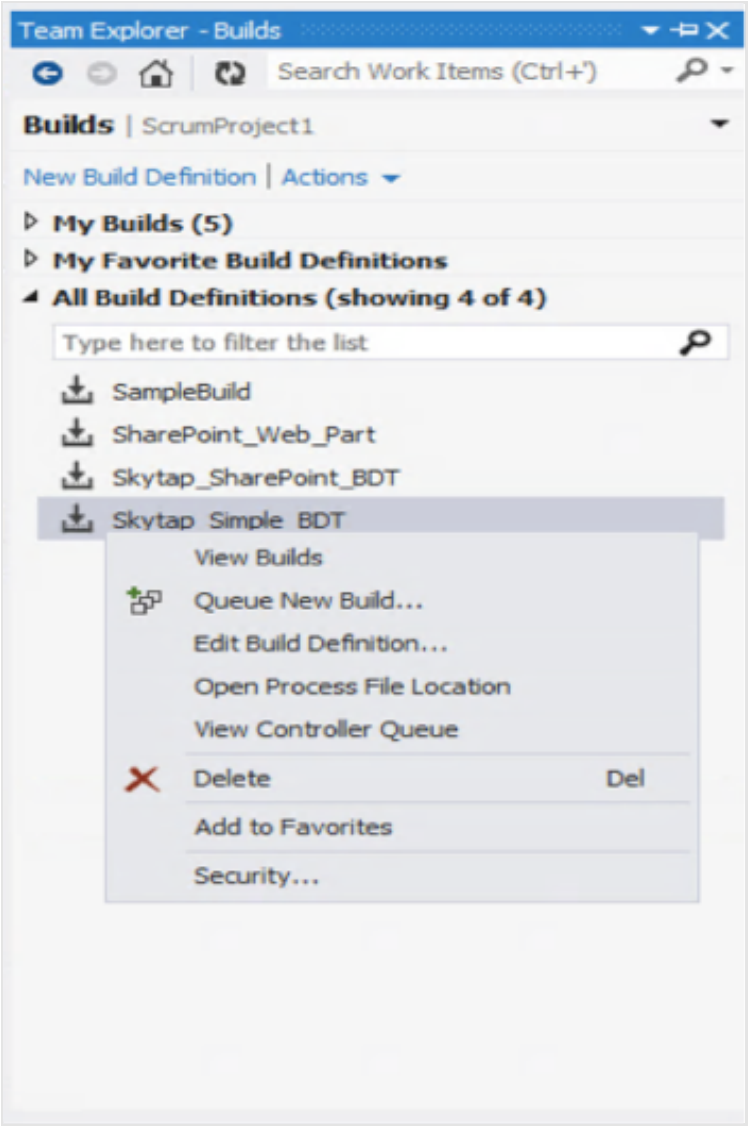
Run the BDT

In this section, you will see how an Automation Pack enabled BDT will run. In terms of workflow, you will see that the steps are no different from a vanilla BDT supported by TFS. However, during the workflow, the Automation Pack will kick in and take necessary actions to enable the BDT to use environments in Skytap. For the purposes of these illustrations we have used a sample application and some sample test cases for the application. The figure below gives a schematic view of the entire BDT process using the Automation Pack.



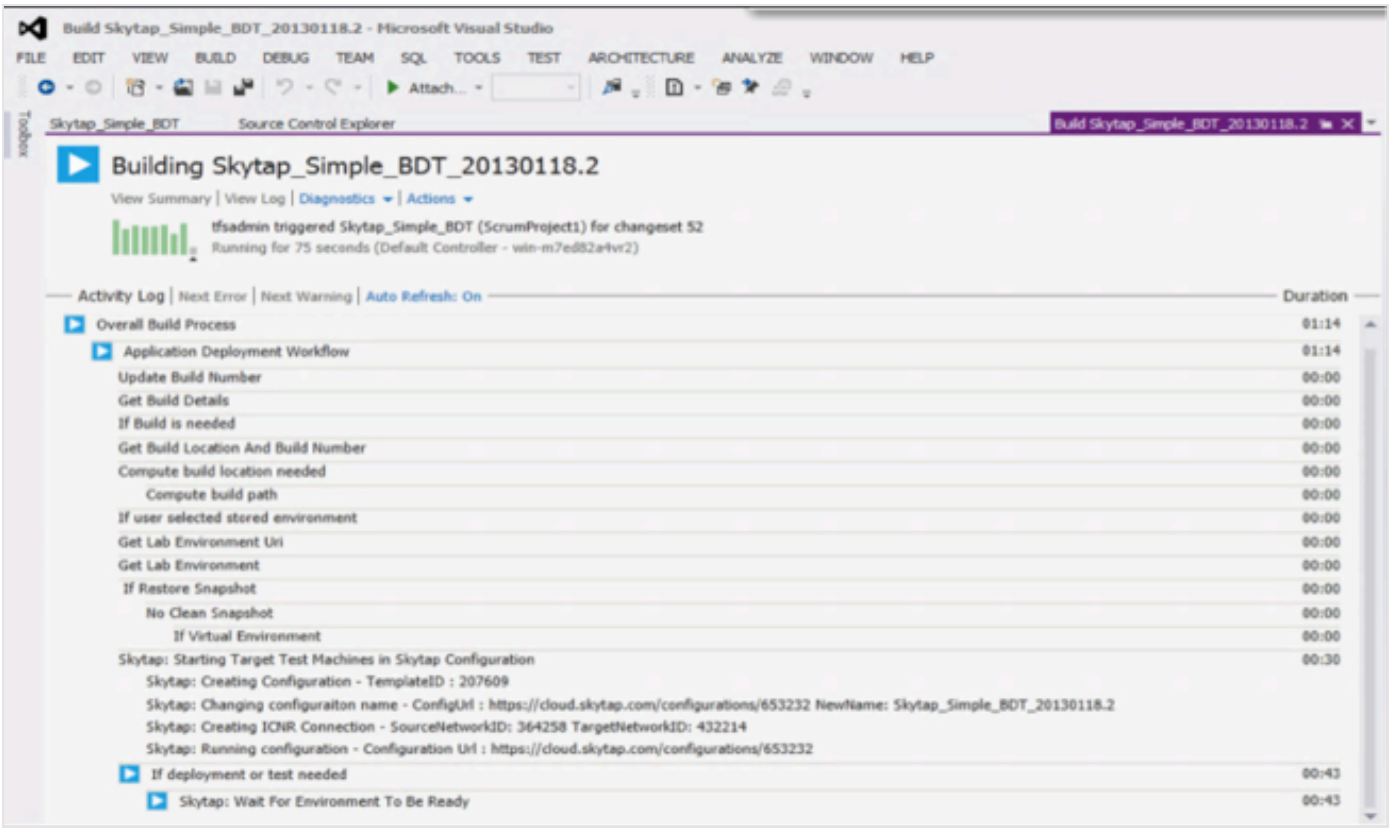
Start a BDT

You can set up the BDT to be executed as part of any check-in, or trigger a build manually. In this illustration we have triggered a manual build that will start the associated BDT that we have defined.

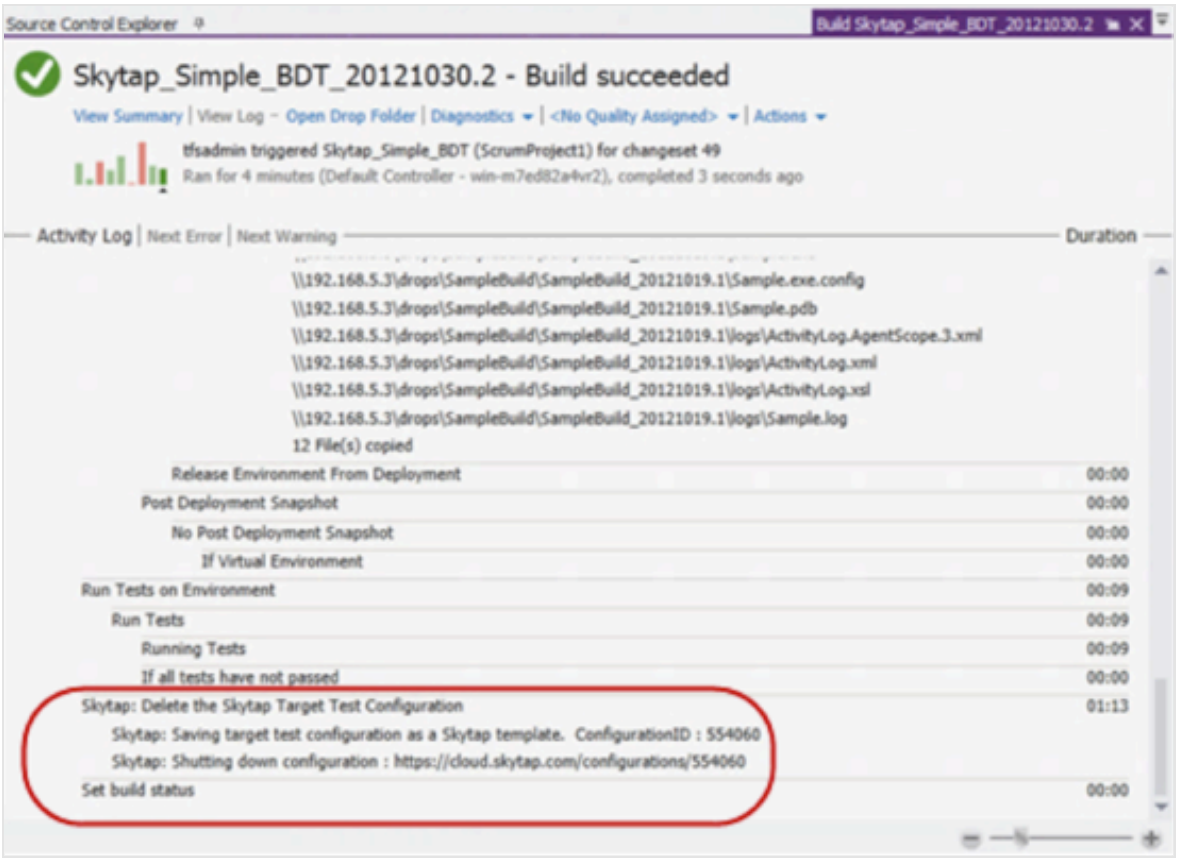


Automation Pack in Action

The build logs show the Automation Pack in action, where it is dynamically creating the test environment in Skytap and using AutoNetworking capabilities that connect back to a TFS over a VPN connection. All of this can be accomplished within a minute. The screenshot of Visual Studio on the developers machine shows the BDT in action.

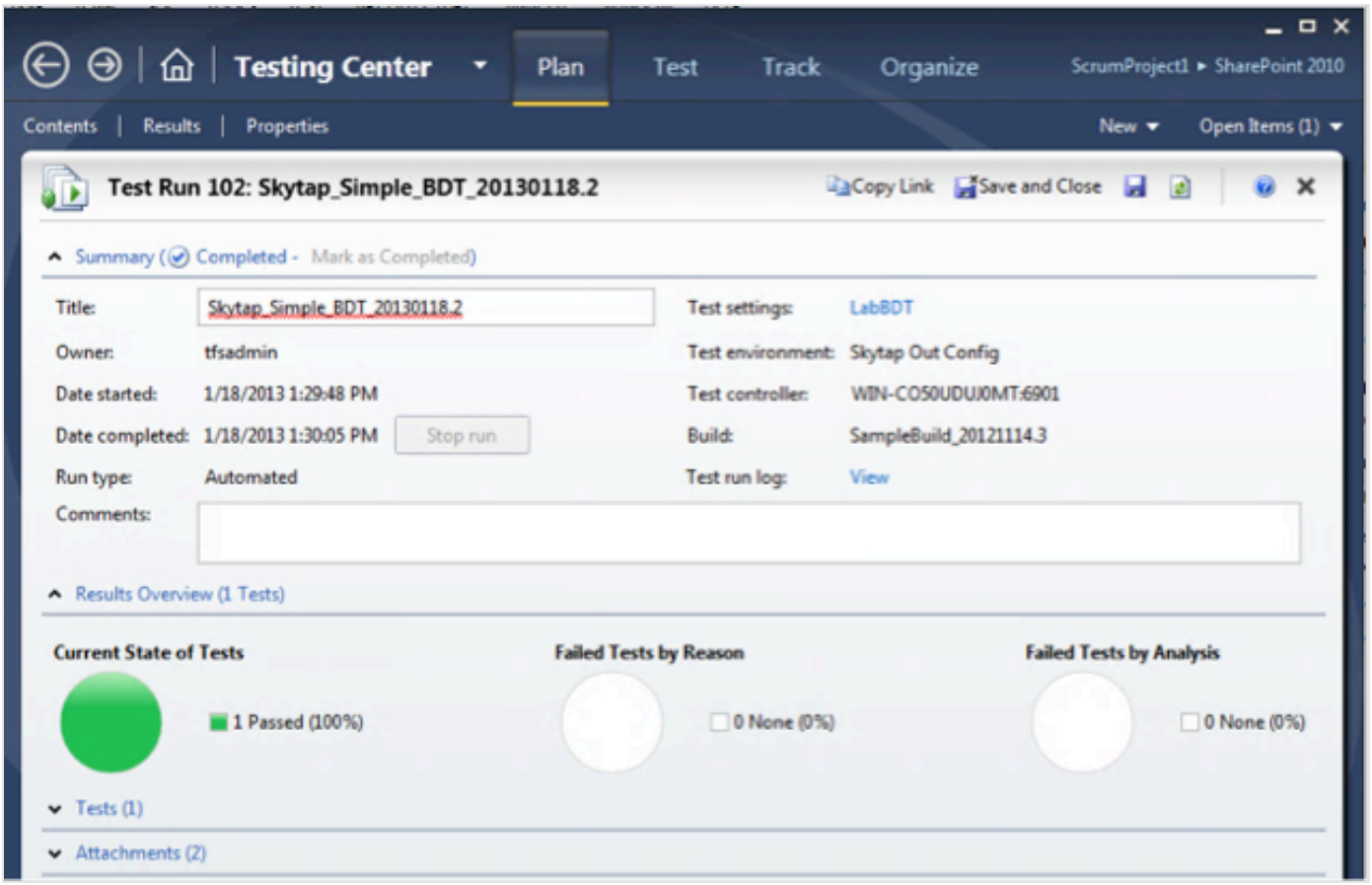


As a result of BDT, the test environment is dynamically created in Skytap from the template we had defined previously. The test environment is renamed with the build number of the build that triggered its creation. This makes it easy to identify the environments corresponding to particular BDT runs. The deployment of the application and tests taking place can be seen in the figure below. When the tests are complete, the test environment is automatically deleted. If there are failures, the test can be saved as a template for further investigation.



Results

Once the BDT run is done, you can see the logs right in Visual Studio on your developer box and also see the results of tests and other diagnostic data you collected, in Microsoft Test Manager.



If you have any questions about these scenarios, contact support@skytap.com.

Was this page helpful? ○ Yes ○ No

Tell us more...

Submit Feedback