

Windows Azure Deployment Tracker   
User Manual

*This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.*

*Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.*

*This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.*

*© 2012 Microsoft. All rights reserved.*

*Microsoft, SQL Server, Visual Studio, Windows, Windows Azure, Windows Live, and Windows Server are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.*

Contents

[Introduction 4](#_Toc332983766)

[Purpose 4](#_Toc332983767)

[System requirements 4](#_Toc332983768)

[Architecture 4](#_Toc332983769)

[Downloading and compiling the application 5](#_Toc332983770)

[Tool overview 5](#_Toc332983771)

[Scenario overview 6](#_Toc332983772)

[To label the application code in TFS that you want to deploy to the cloud 6](#_Toc332983773)

[To manage the TFS instance in Deployment Tracker where labeled code resides that you want to deploy 7](#_Toc332983774)

[To configure application-level settings, like logs and the TFS folder path 9](#_Toc332983775)

[To select labeled code in TFS for deployment 10](#_Toc332983776)

[To import the Windows Azure Publish Settings file from the Windows Azure Management Portal and to get the list of subscriptions associated with it 12](#_Toc332983777)

[To manage a Windows Azure storage account and hosted service components 14](#_Toc332983778)

[To make a new deployment to the cloud 18](#_Toc332983779)

[To deploy an existing package to Windows Azure 20](#_Toc332983780)

[To manage application-level logs that have error information encountered during the deployment process 22](#_Toc332983781)

[To view all of the successful or failed deployments by a particular person 23](#_Toc332983782)

[To roll back to a previous version of an application 24](#_Toc332983783)

[To upgrade an already-deployed application on a hosted service 25](#_Toc332983784)

## Introduction

The Windows Azure Deployment Tracker tool simplifies the deployment and management of cloud-based applications. Deployment Tracker fulfills your requirements by automating the process of compiling and deploying the application to cloud environments.

You can use Deployment Tracker to do the following:

Build and deploy your application to the cloud with a single click.

Audit deployments, including who performed deployment, what version was deployed, where the deployment was done, and when the application was deployed.

Not rely on directions from developers about what is needed and what is not supported during deployment.

Get alerts about the status of the deployment whether or not there are problems.

Perform multiple, simultaneous deployments.

Roll back to the previously deployed version with a single click.

View the results of a Windows Azure application deployment, and manage filters.

## Purpose

This document provides an overview of the Deployment Tracker sample application by describing the application and by taking you through different usage scenarios.

## System requirements

To run Deployment Tracker, you must have the following:

Windows 7 or Windows Server 2008, 64-bit operating system.

Visual Studio 2010 (any edition) and Team Explorer.

Windows Azure Software Deployment Kit (SDK) 1.6.

Access to TFS.

Access to MS Build Server.

A Windows Azure hosted service.

A Windows Live account, to log in to your Windows Azure Management Portal.

At least one Windows Azure subscription, to deploy the application to the cloud.

Administrator privileges to run Deployment Tracker.

## Architecture

## Downloading and compiling the application

## Tool overview

Deployment Tracker makes it easy for you to automatically deploy application code to the Windows Azure Management Portal.

You can use Deployment Tracker to do the following:

Specify the TFS instances that you want to use during the deployment process.

Download the labeled solution for a specific TFS instance.

Configure the Windows Azure settings, such as the subscriptions, hosted service, storage account, and environment that you want to deploy the application code to.

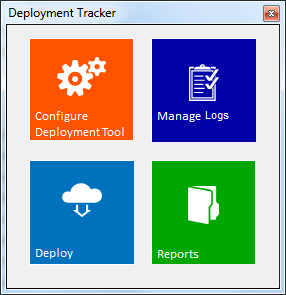
Build, publish, and deploy the downloaded application code into the Windows Azure Management Portal with a specific Windows Azure configuration.

Manage application-level error logs (if any) for each deployment that you attempt.

View reports that provide the status of each deployment.

Roll back currently running application code in the Windows Azure Management Portal to the last successfully deployed version (if any).

The following graphic shows the initial screen of Deployment Tracker:



Deployment Tracker has the following four sections:

**Configure Deployment Tool:** You can configure multiple TFS instances, and the Configure Deployment tool lists all TFS instances that you have saved in the database.

**Deploy:** The deployment tasks make up the core functionality of the application, and they automate the following:

Locate the labeled solution for a specific TFS configuration

Import the Publish Settings file from the Windows Azure Management Portal.

Manage the Windows Azure storage account and hosted service components.

Compile, build, and deploy the packages to the target environments.

**Manage Logs:** You can use Manage Logs to view any application-level errors, information, and warnings that arise during the deployment.

**Reports:** Reports tracks of all of the successful and failed application deployments that you make with Deployment Tracker. It records the status of the deployment (Failed, Deployed, Rollback, or Deploying), the target environment (Staging, Production), the timestamp, and the name of the person who deployed the application.

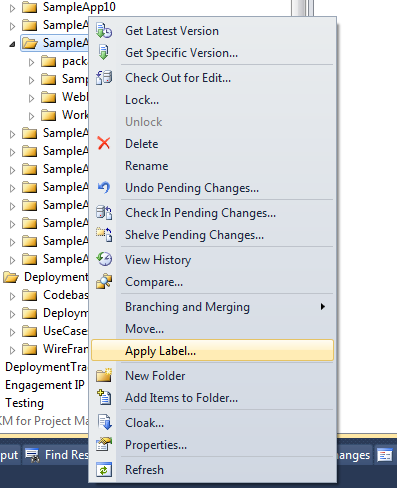
You can use Reports to filter the data by Date Range, Environment that the application is deployed to, Status (Deployed, Failed, Deploying, or Rollback), who the deployment was Performed By, and the Application Name.

Reports also includes a rollback mechanism, which you can use to roll back an application that is running in the cloud to the last successfully deployed version.

## Scenario overview

### To label the application code in TFS that you want to deploy to the cloud

You can connect to TFS, and then select the application solution that you want to label. Right-click the application solution, and then click **Apply Label**.

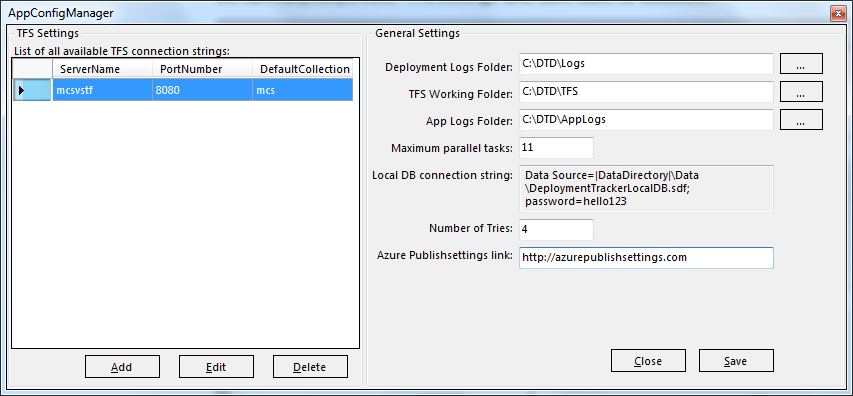


A window appears where you can creates a specific label for a selected solution. Type a label, and then click the **Create** button.

### To manage the TFS instance in Deployment Tracker where labeled code resides that you want to deploy

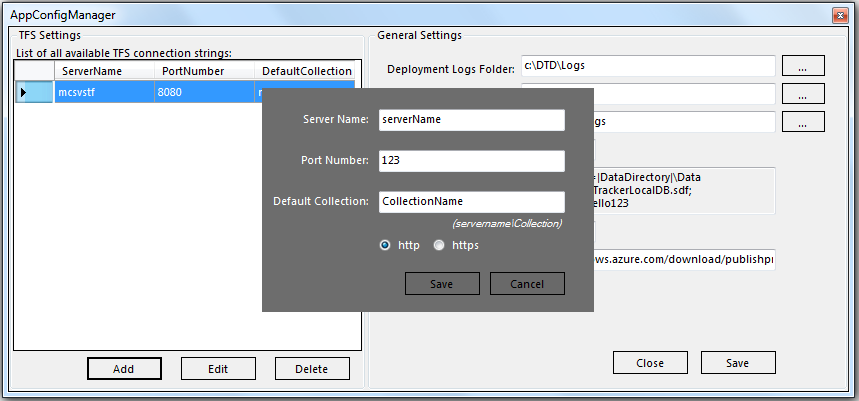
You can configure multiple TFS instances, and the Configure Deployment tool lists all TFS instances that you have saved in the database.

First, launch Deployment Tracker, and then click **Configure Deployment Tool**. The following window appears:

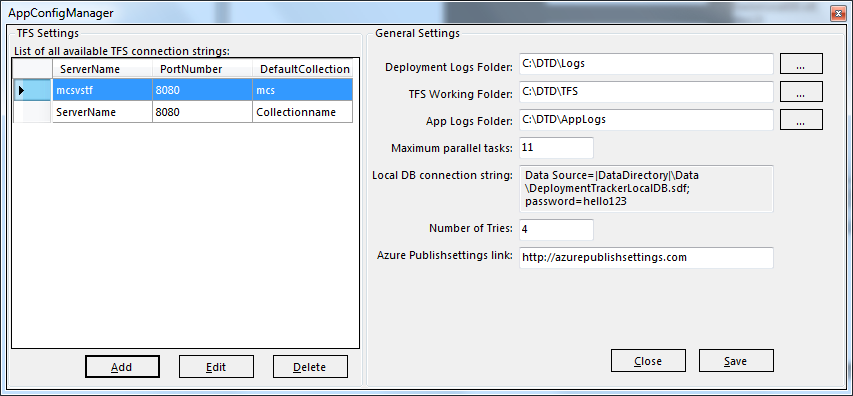


You can add, edit, or delete TFS settings and then save them to a local configuration database, as follows.

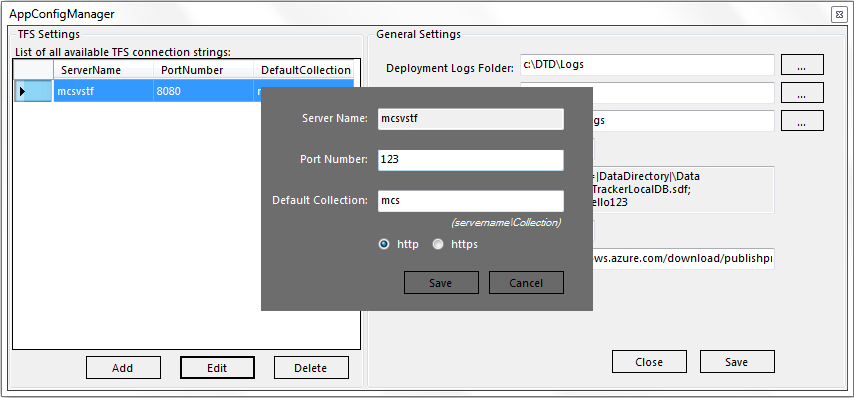
To add settings, click **Add**.TFS Editor appears. Type a **Server Name**, **Port Number**, **Default Collection** name, and choose a protocol (**http** or **https**). Click **Save**.



The following window appears:



To edit the TFS instance, either double-click the record or select the record and then click **Edit**. Type new values for **Port Number**, **Default Collection**, and protocol, and then click **Save** to save to the database.

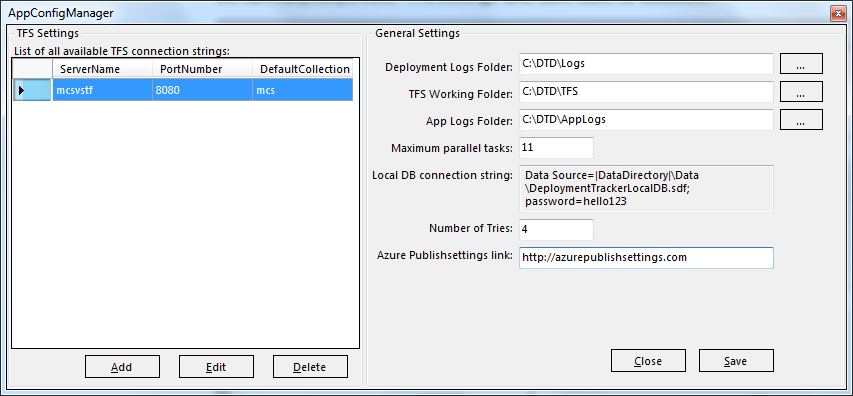


**Note:** To edit the **Server Name**, delete the record, and then add a new one.

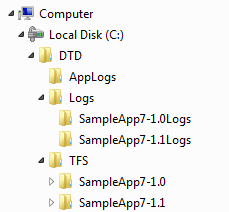
To delete a TFS instance, select the record, and then click **Delete**.

### To configure application-level settings, like logs and the TFS folder path

Launch Deployment Tracker, and then click **Configure Deployment Tool**. The following window appears:



To manage application-level settings, in **General Settings**, navigate to one of the following folders:



**Deployment Logs folder:** Configurable local folder where a log file is created for each day that the tool is accessed and which contains application-level error information (if any).

**TFS Working folder:** Folder where labeled code is downloaded from TFS.

**AppLogs folder:** Folder where TFS, MS Build Server, and Deploy logs exist for each deployment.

**Maximum parallel tasks:** Specifies how many parallel deployment processes to allow for the application (configurable).

**Local DB connection string:** Path of the .sdf file.

**Number of tries:** How many times the tool tries to communicate with Windows Azure.

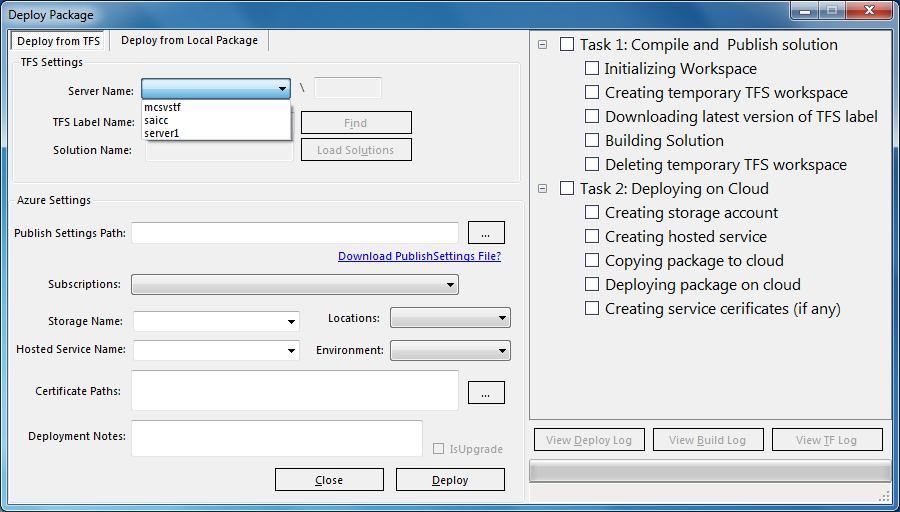
**Azure Publish Settings link:** Where you can download the Windows Azure Publish Settings file from.

To save changes to the settings, click **Save**. Then close and restart Deployment Tracker to reflect the new settings.

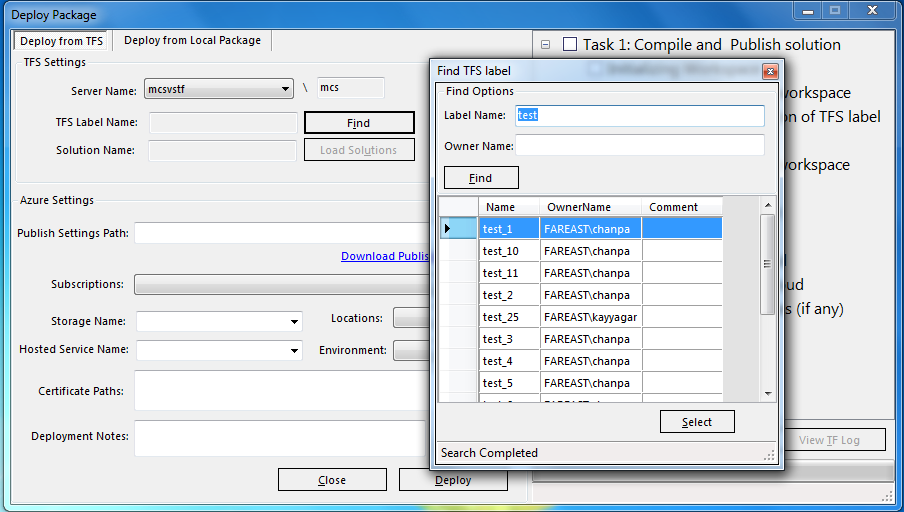
### To select labeled code in TFS for deployment

After you label the code in TFS and save the TFS server configuration to the database, the next step is to download the code from TFS into a local folder.

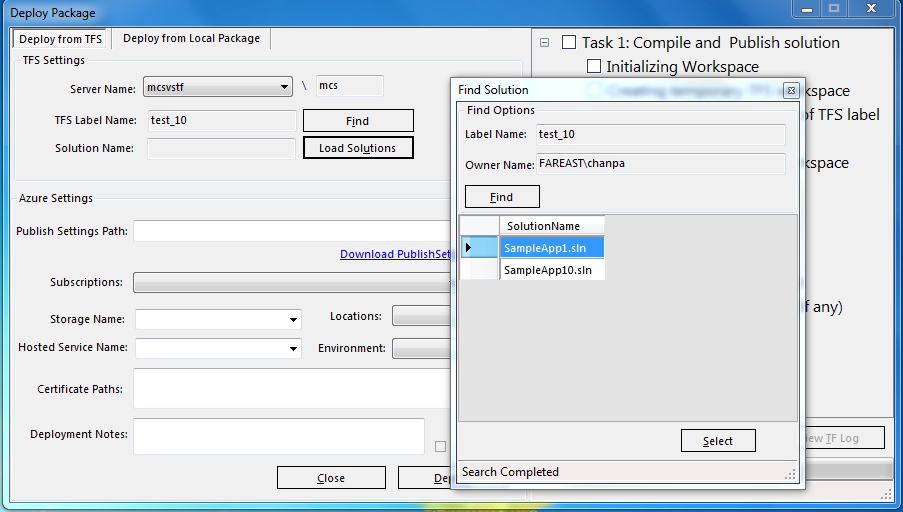
First, start Deployment Tracker, and then click **Deploy**. The following window appears:

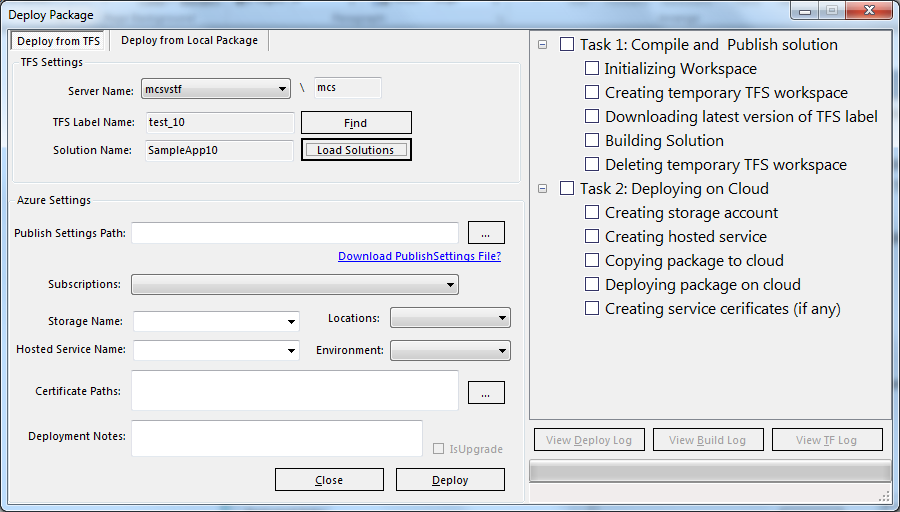


The **Deploy from TFS** tab lists the TFS instances. Choose a TFS instance, and then click the **Find** button. The **Find TFS Label** window appears.



Select a label, and then click **Load Solutions** to find solutions that are associated with that label. Select the solution that you want to build and deploy.



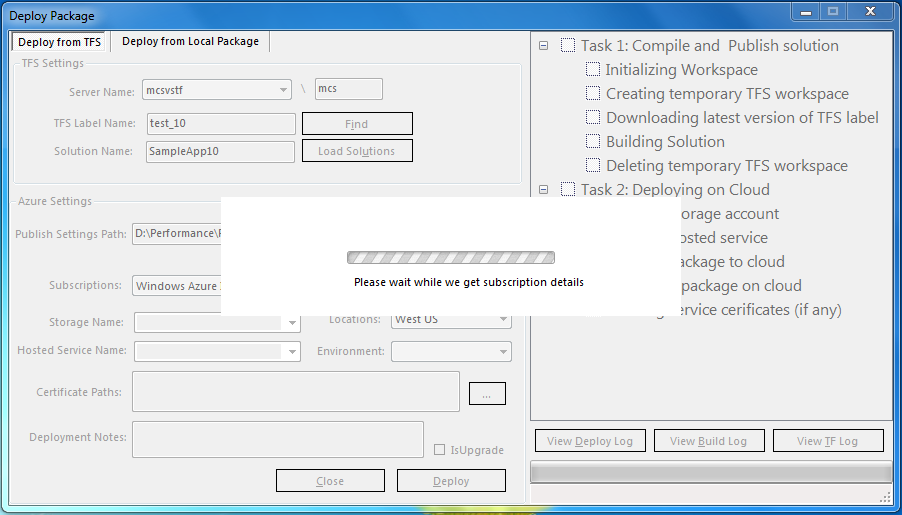


### To import the Windows Azure Publish Settings file from the Windows Azure Management Portal and to get the list of subscriptions associated with it

If you have already downloaded the Publish Settings file, you can browse to it from the **Deploy Package** window in order to import Windows Azure subscription settings. If you do not have the Publish Settings file already, you can download it with Deployment Tracker by using the **Download Publish Settings File?** link.

The Publish Settings file is an .xml file containing information about your Windows Azure subscriptions. The **Download Publish Settings File?** link takes you to the Windows Azure Management Portal. You need to log in first with your Windows Live ID, and then you are prompted to save the Publish Settings file. Remember the path where you save the file.

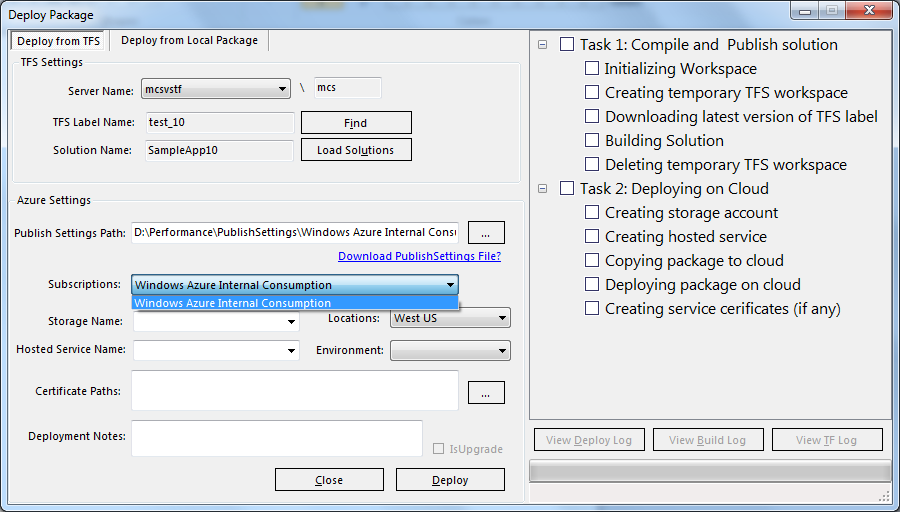
After you have the Publish Settings file, browse to it from Deployment Tracker in order to import the subscription information.

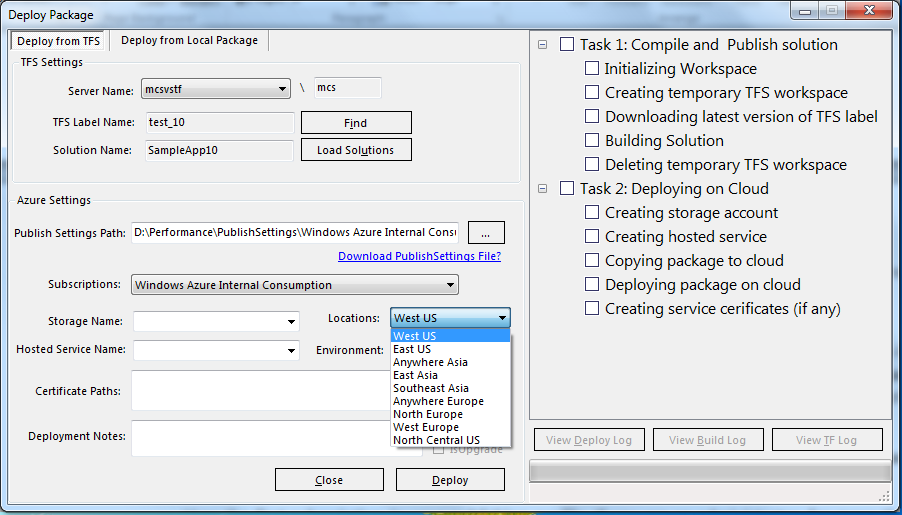


**Subscriptions and Locations:** A Windows Azure subscription grants you access to Windows Azure services and to the Windows Azure Management Portal.

After you import the Publish Settings file, its associated subscriptions (if any) are loaded automatically.

By default, Deployment Tracker uses the first subscription in your publishing profile. If you administer more than one subscription, you should specify a subscription to ensure that the intended subscription is used. After you specify it, the associated deployment locations are loaded automatically.





### To manage a Windows Azure storage account and hosted service components

A Windows Azure application requires both compute and storage services. The following steps describe how to create a new Windows Azure storage account to allow the application to persist its data. They also describe how to define a hosted service component to execute application code.

The naming conventions for Windows Azure storage accounts and hosted services are as follows:

Storage account names:

* Must be unique within an account.
* Must start with a letter or a number.
* Cannot contain any special characters.
* Must be 3 to 24 characters long and use lowercase letters only.

Hosted service names:

May use letters A-Z and numbers 0-9 and only the special character “-” (hyphen). The names are not case-sensitive.

May not use a space.

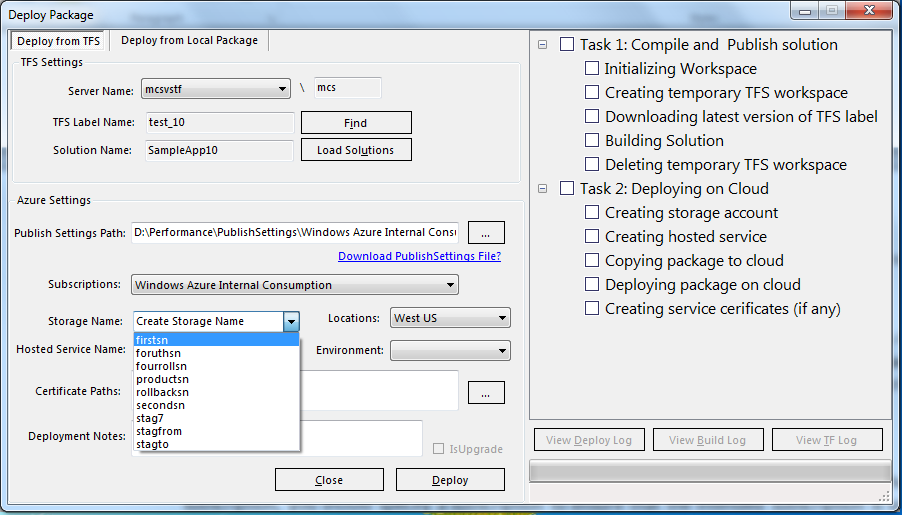
Can be 1 to 63 characters long.

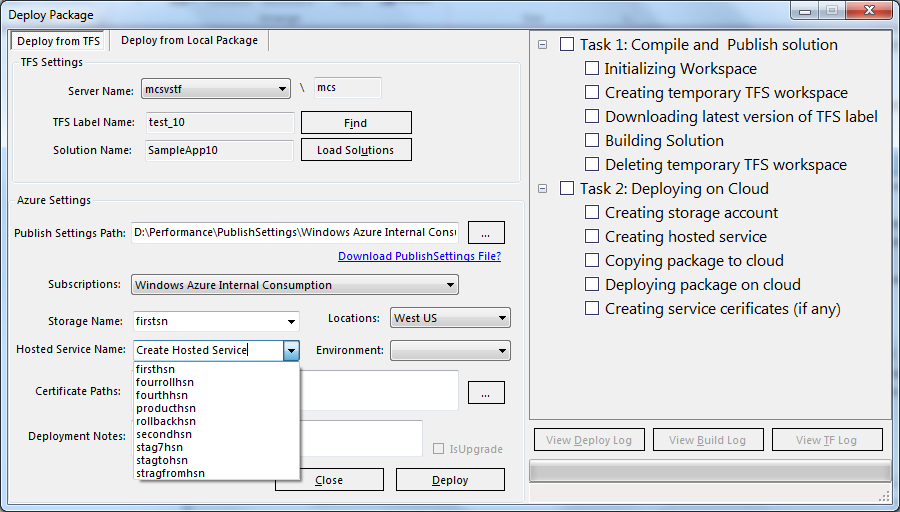
Cannot start with a hyphen.

To create storage account and hosted service components, set the following fields:

**Storage Account and Hosted Service Names:** After you import the Publish Settings file, Deployment Tracker automatically loads the subscription associated with it**.** The hosted services and storage account that are available for the selected subscription are also listed.

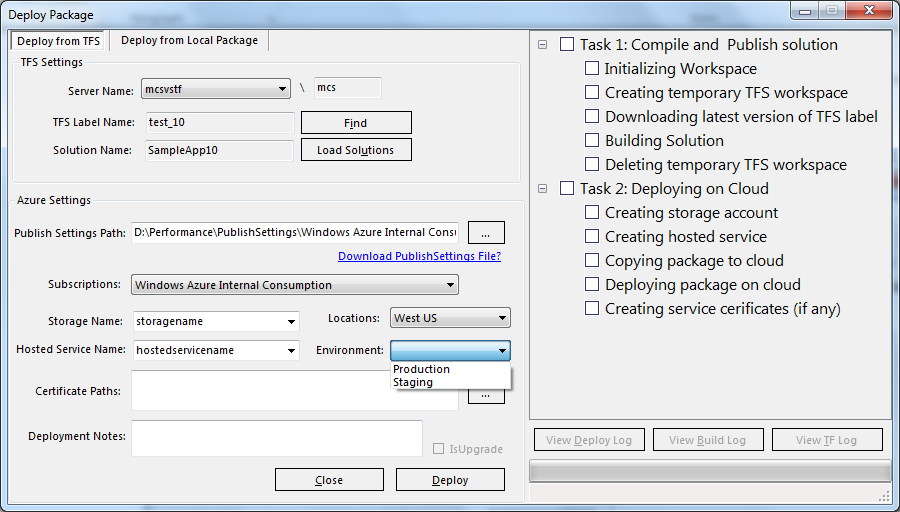
You can create new storage account and hosted service names in the current subscription, or you can choose existing ones.





**Note**: If the storage account and hosted service names are already in use for another subscription, the deployment fails.

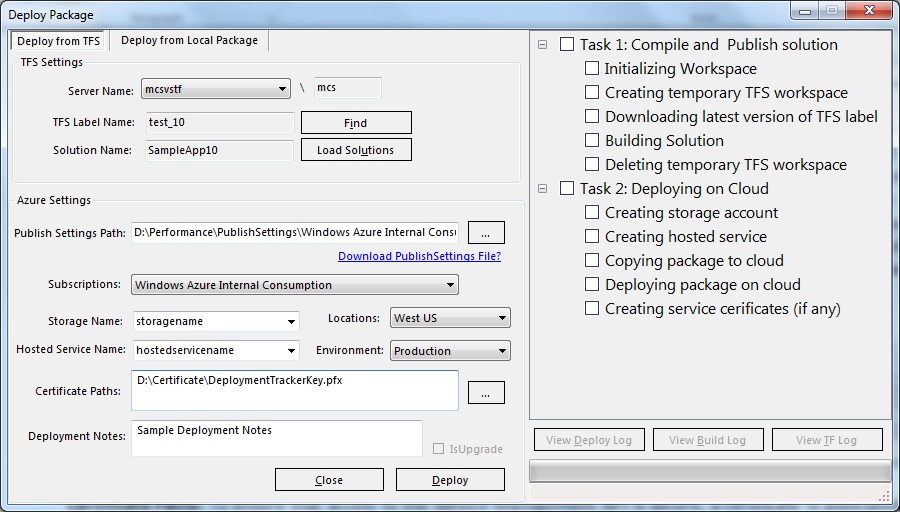
* **Environment:** You can deploy the application to either the staging or production environment.



**Certificate Paths:** To help secure access to the Service Management API, a certificate is associated with the subscription. The management service uses this certificate to authenticate requests. The certificate can be self-signed or signed by a certification authority. Any valid X.509 v3 certificate is suitable, as long as its key length is at least 2048 bits.

This is an optional field. If you use it, the certificate is uploaded to the **Certificates** folder for the particular hosted service in the **Hosted Services** node in the Window Azure Management Portal.

* **Deployment Notes:** You can use this field for your remarks about deployment.



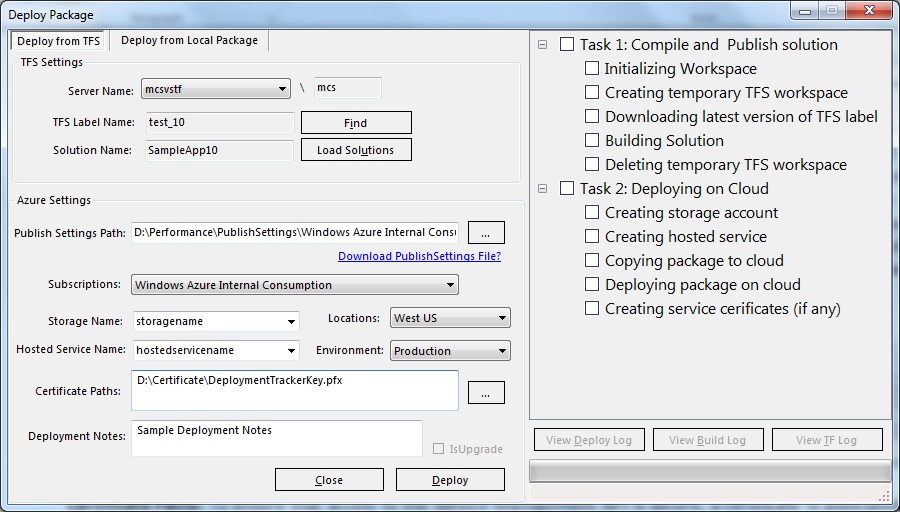
### To make a new deployment to the cloud

To deploy an application to Windows Azure, you need to build and publish it into a package file (.cspkg) and a configuration file (.cscfg).

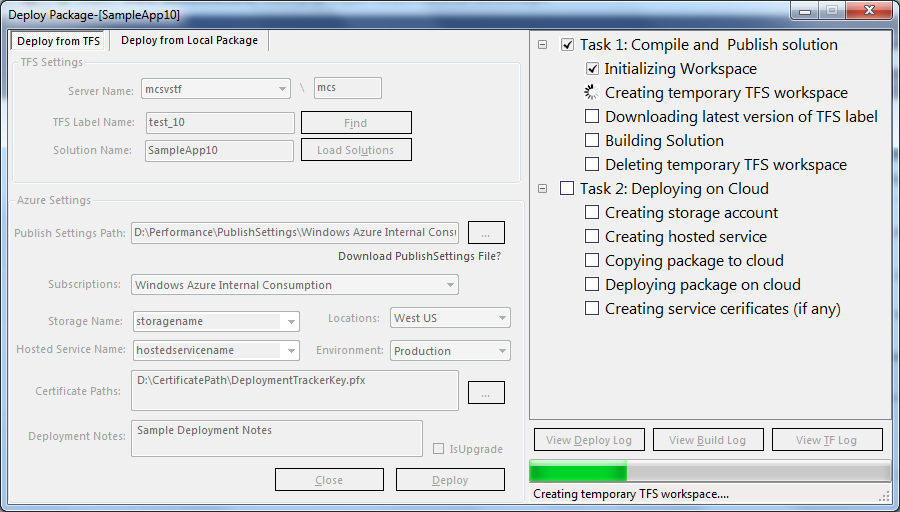
With Deployment Tracker, you can deploy an application to Windows Azure in two ways. You can either download, build, publish, and deploy labeled code from TFS, or you can deploy an existing package to Windows Azure.

To download, build, and publish labeled code from TFS:

1. Start Deployment Tracker, click **Deploy**, and then click **Deploy from TFS**.
2. The window that appears is used to prepare for and initiate the deployment. Identify the solution, identify the **Publish Settings Path** (download the Publish Settings file if necessary), and identify the environment (**Storage Name**, **Hosted Service Name**, **Locations**, and **Environment**).



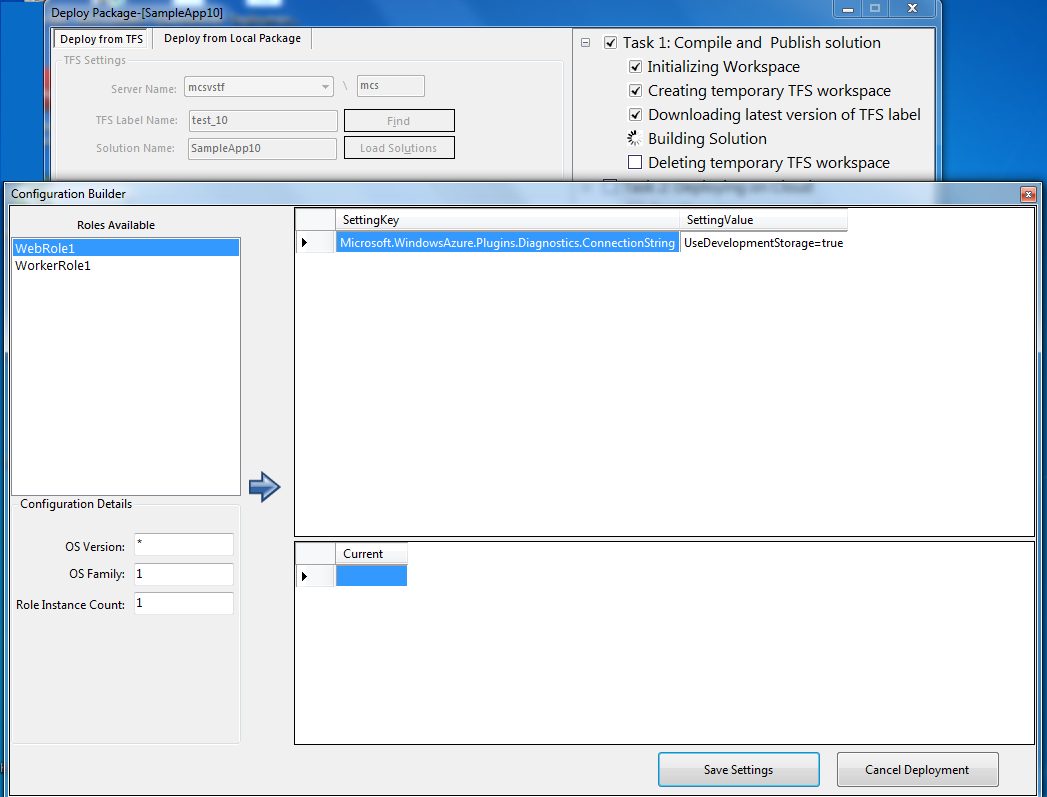
1. Click the **Deploy** button to start the deployment process.



This screenshot shows the deployment status and progress indicators that Deployment Tracker provides. The tasks to be completed are listed. Icons are used to indicate whether the task is in process (⭯), completed (🗸), or had an error (⮿). Buttons provide quick access to the various log files.

Building and publishing the downloaded code by using MS Build Server

After the code is downloaded locally, Deployment Tracker compiles and builds the solution by using MS Build Server to generate the package file and the configuration file (.cspkg and .cscfg) in the path that you specified. While the building is in progress, a window appears where you can make several configuration changes, such as the number of instances for each role.



Click **Save Settings** to save the changes and to continue the deployment process with the new configuration.

Click **Cancel Deployment** to stop the deployment process.

Deploying the packages to the target environment (staging/production)

After the package files are generated, they are ready to be deployed to the cloud. Before deploying the packages, a storage account and hosted service name need to be in place on the Windows Azure Management Portal.

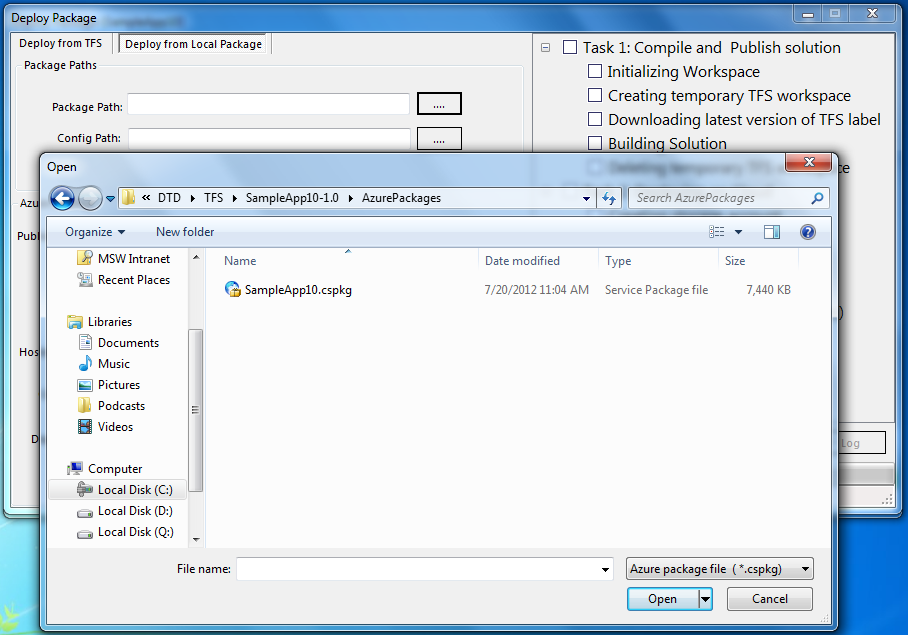
You can create new storage account and hosted service names, or you can use existing ones in the Deploy Package window of Deployment Tracker. After the storage account and hosted service names are in place, the package files are copied into the storage account, and then they are deployed from there to either the staging or production environment.

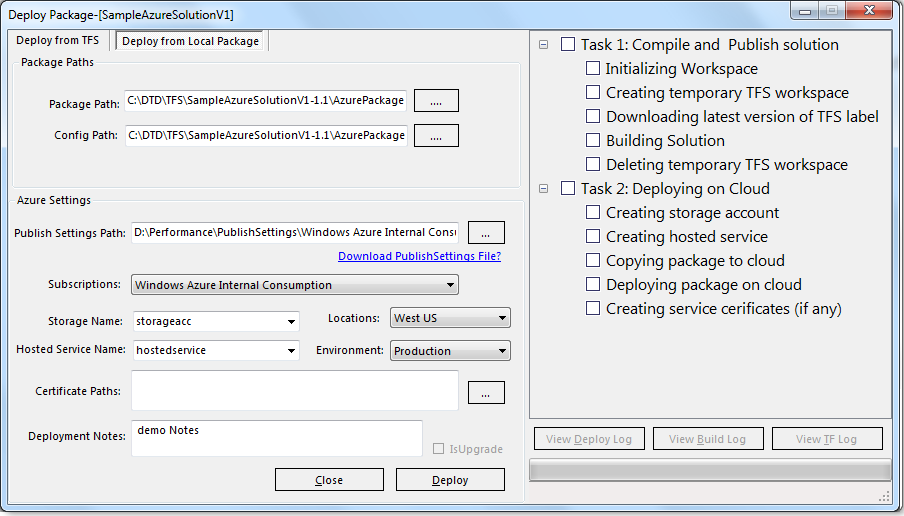
A certificate is added in the Windows Azure hosted service section if you include one when you configure Deployment Tracker.

### To deploy an existing package to Windows Azure

This involves deploying the local package (.cspkg and .cscfg) to Windows Azure. It does not include downloading the code from TFS, building the solution, or publishing it.

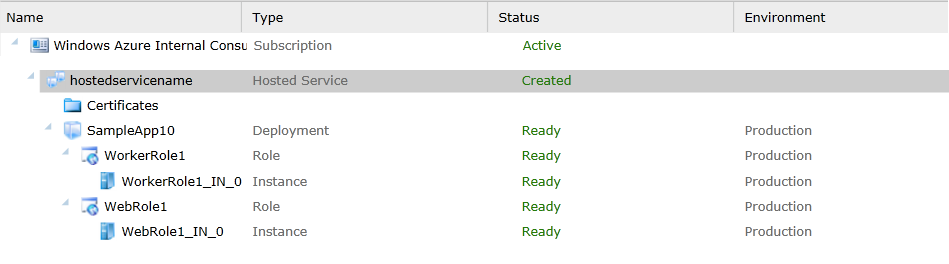
1. To create a service package, right-click the cloud service project, and then click **Package**.
2. In the **Package Windows Azure Application** window, click **Package**, and then wait until Visual Studio creates it. In the alternative, you can generate package files from Visual Studio and specify the file paths from **Deploy from Local Package**.
3. Start Deployment Tracker, click **Deploy**, and then click **Deploy from Local Package**.
4. After the package is ready, Visual Studio opens a window showing the folder that contains the generated files. Browse to the package and configuration files, as shown in the following figure.





**Note:** If you deploy an existing package, Deployment Tracker skips **Task 1: Compile and Publish solution** and begins from **Task 2: Deploying on Cloud**. It involves following step:

1. After the package files are in place, they are ready to be deployed to the cloud. Before deploying the packages, the storage account and hosted service name needs to be in place on the Windows Azure Management Portal. Either you can create new storage account and hosted service names or you can use existing ones, in the **Deploy Package** window of Deployment Tracker.
2. After the storage account and hosted service names are in place, the package files are copied into the storage account, and then, from there, they are deployed to either the staging or production environment.
3. After the deployment process is completed, you can log in to the Windows Azure Management Portal to verify that the application code is running in the hosted service, as shown in the following figure.



If the deployment process fails, you can view the Deploy Log, Build Log, and TF log by clicking a button in the **Deploy Package** window.

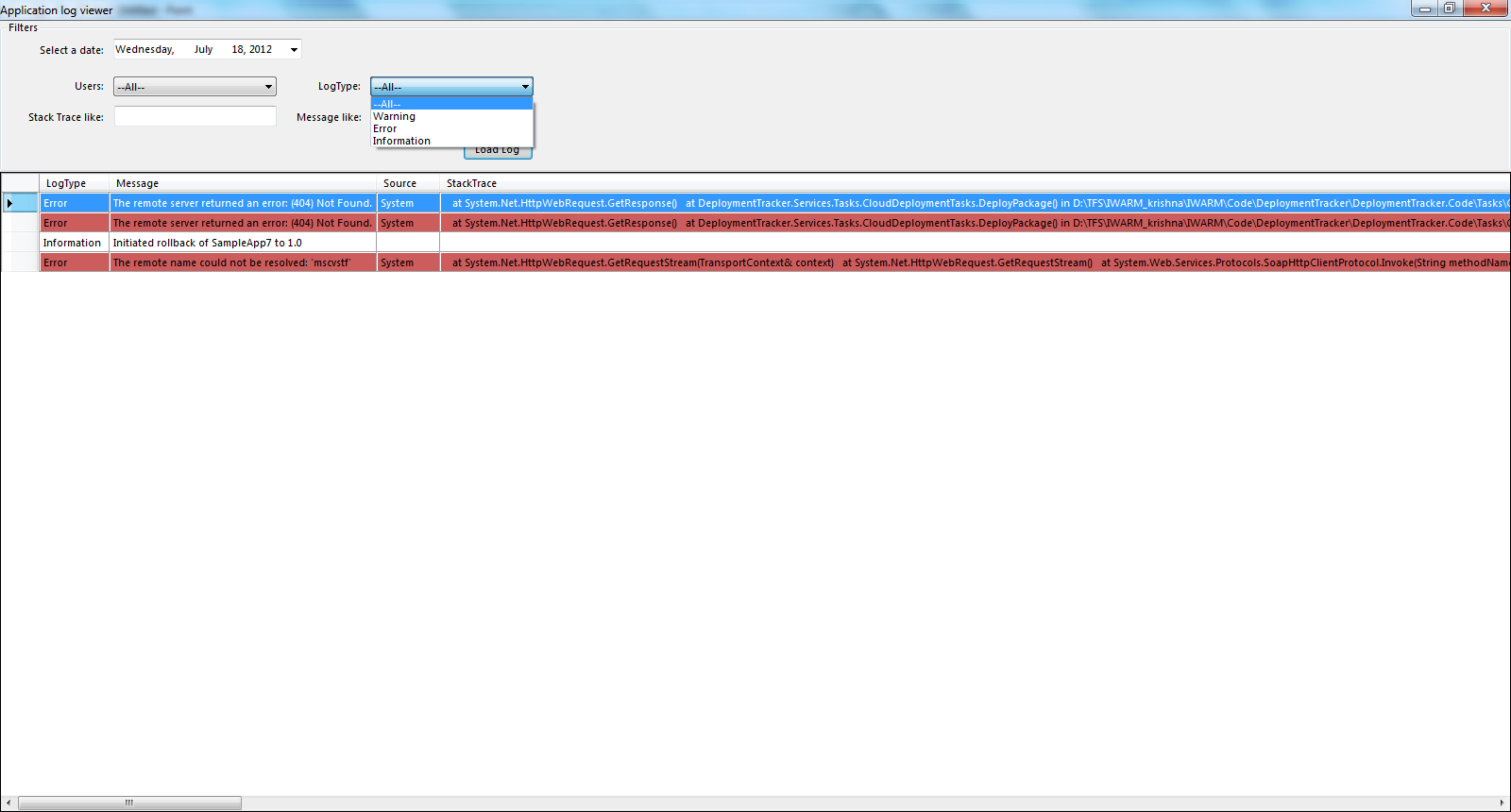
**Note:** Swapping the application code that is deployed to the hosted service between the staging and production environments is not supported by the tool out of the box. You need to log in to the Windows Azure Management Portal and carry out a virtual IP (VIP) swap.

### To manage application-level logs that have error information encountered during the deployment process

You can use **Manage Logs** to view application-level errors, information, and warnings encountered during deployment process.

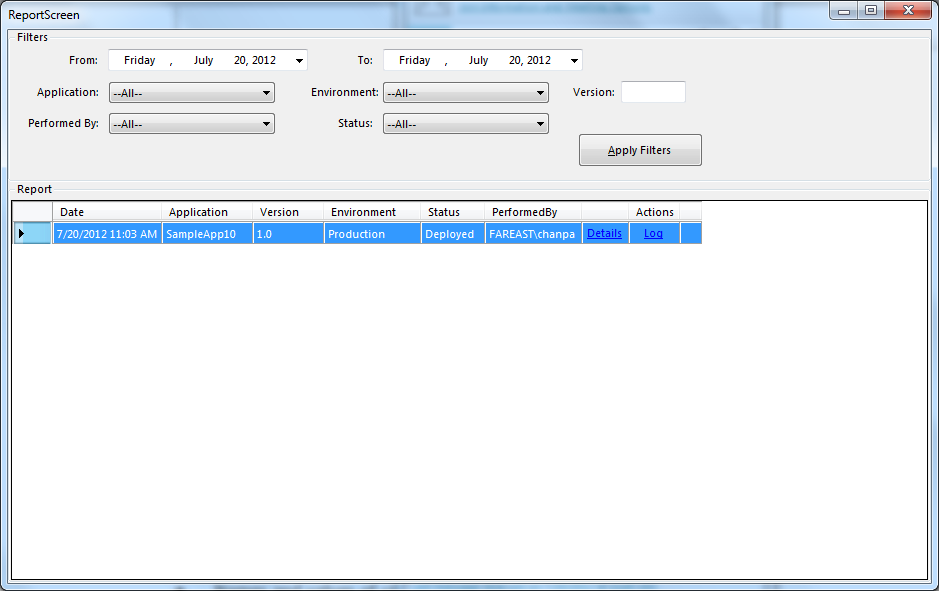
If the application log itself does not capture the reason for the failure, the exception is written to the system log under the Application category.

To manage the logs, start Deployment Tracker, and then click **Manage Logs**. The following window appears:



### To view all of the successful or failed deployments by a particular person

Reports tracks all of the successful and failed application deployments that a particular person makes, along with the deployment status (Failed, Deployed, Rollback, and Deploying), the target environment (Staging, Production), the timestamp, and the name of the person who tried to perform the deployment.

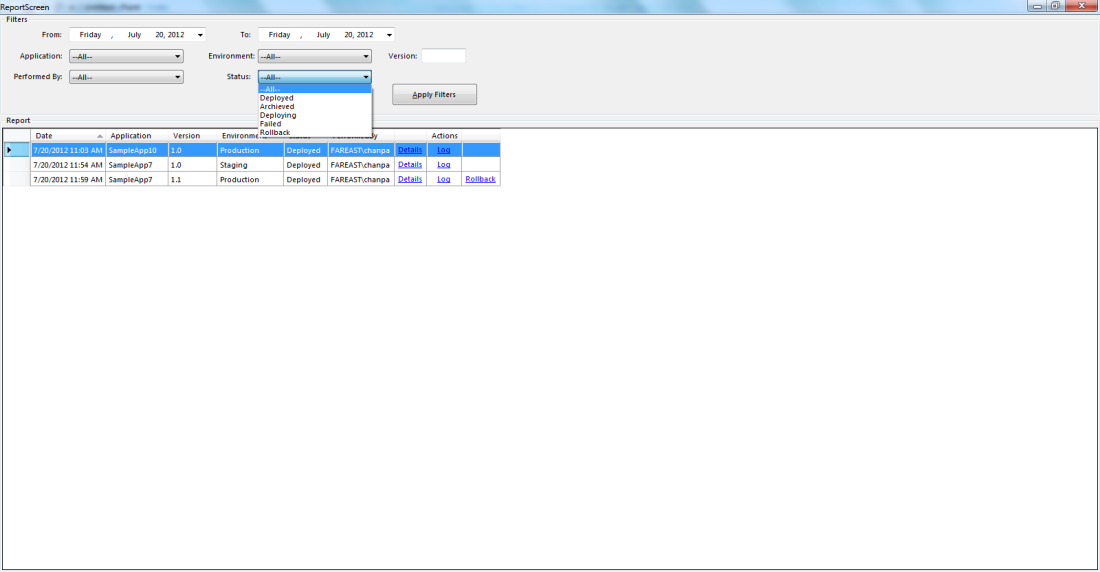


You can use the **Reports** window to filter the data in several ways, including by Date, Application name, Environment, Performed by, Status, and Version. (Deployment Tracker maintains the version internally for the rollback mechanism. The application version does not appear in the Windows Azure Management Portal.)

### To roll back to a previous version of an application

You can roll back from the **Report** window. A rollback replaces the currently deployed version of the application code with the last successfully deployed version.

Reports also provides a way to roll back. If you want to roll back the current version of a deployed application to the last successfully deployed version, go to the **Rollback** link on the report screen.



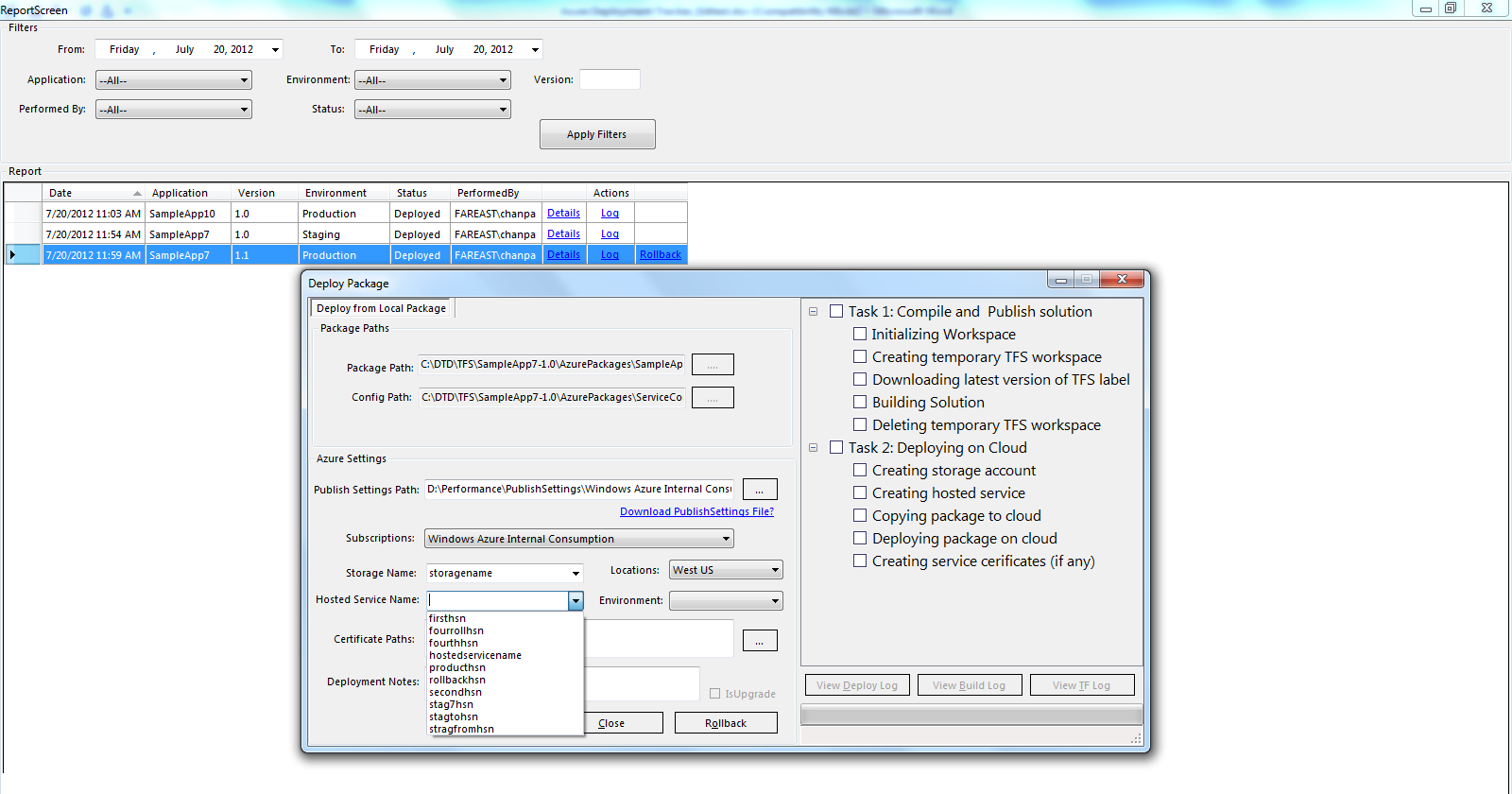
The rollback option is enabled if you have used Deployment Tracker to successfully deploy an earlier version of the application to Windows Azure.

For example, as shown in the preceding figure, there has been only one successful deployment of the SampleApp10 application. Because no previous version has been successfully deployed, the rollback option is not available.

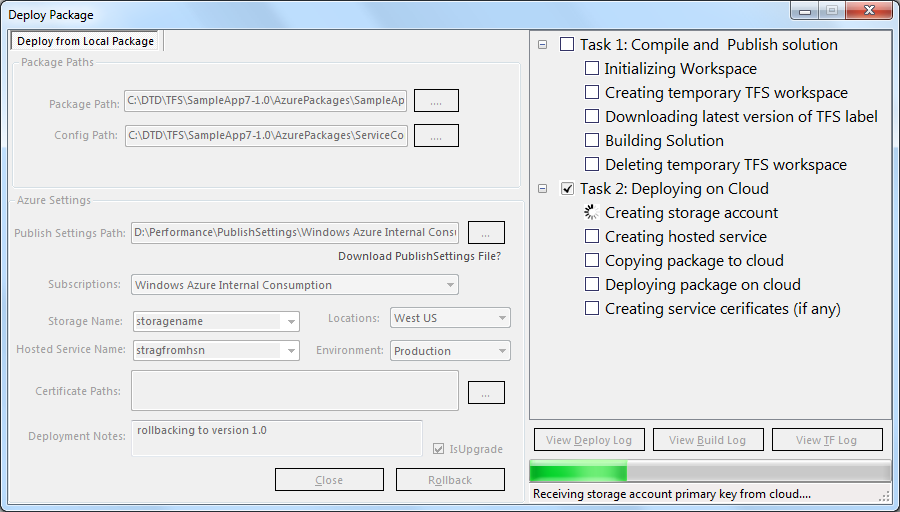
But there are two successful deployments of SampleApp7: version 1.0 and version 1.1. In this case, you can roll back version 1.1 to version 1.0.

If you click **Rollback**, a window appears with the paths to the .cspkg and cscfg files already populated. Deployment Tracker uses the reporting data to select the last successfully deployed package files. Make sure that the same subscription, storage account, and hosted service are selected for the rollback operation. You can verify the settings by using the Windows Azure Management Portal.

**Note:** If you choose a subscription, location, and hosted service other than that provided for SampleApp7 (version 1.1), it replaces the application that is running in the specified subscription, location, and hosted service**.**



If you click **Rollback**, SampleApp7 version 1.2 rolls back to SampleApp7 version 1.1.



### To upgrade an already-deployed application on a hosted service

You can use Deployment Tracker to replace an existing deployment if you update the application code according to the change-management process.

If the updated application code resides in TFS, you can select the appropriate solution from the **Deploy from TFS** tab, as documented previously.

Alternatively, you can generate package files from Visual Studio and then specify the file paths from the **Deploy from Local Package** tab.

In either case, you need to provide the same subscription, location, and hosted service names that the current version of the application uses. You can obtain this information by logging in to the Windows Azure Management Portal.