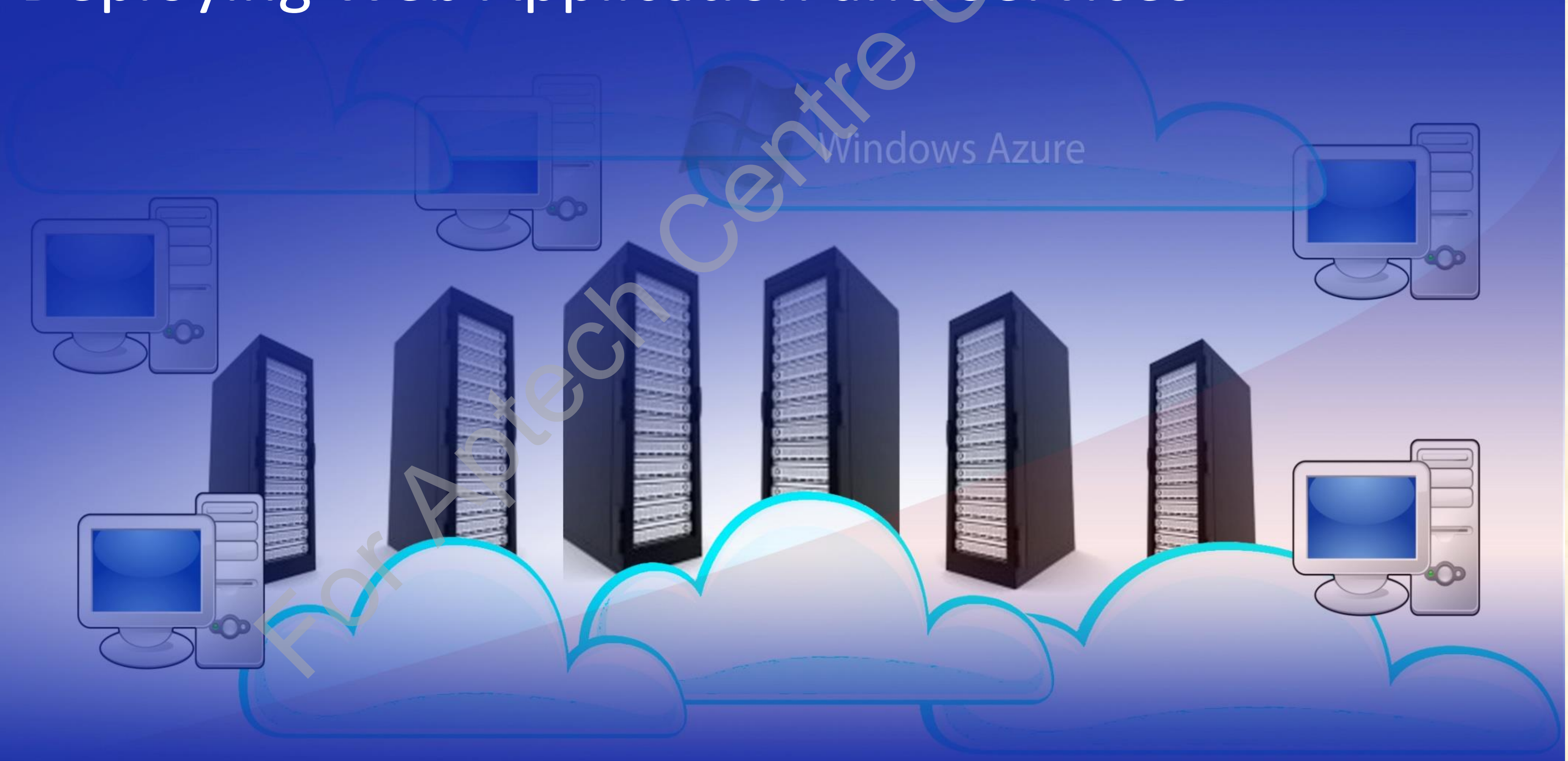
Enterprise Application Development Using Windows Azure and Web Services

Session 14

Deploying Web Application and Services



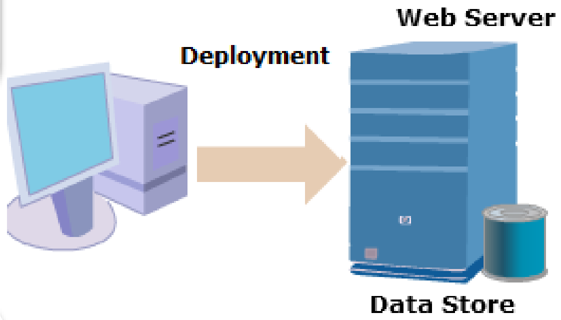
Learning Objectives

- Explain planning and designing a deployment strategy for Web applications and Azure
- Describe configuring and implementing deployment
- Describe using NuGet for managing packages

Planning and Designing a Deployment Strategy 1-2

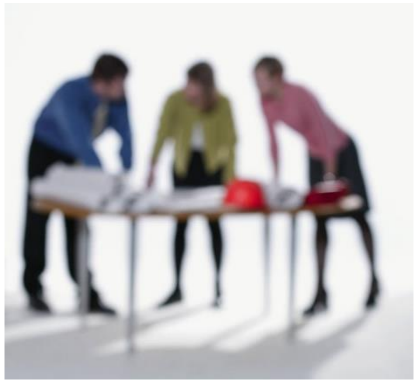
- After creating a Web application project or Website project:

You need to deploy or publish the project to a Web server, so that others can access and use the application.



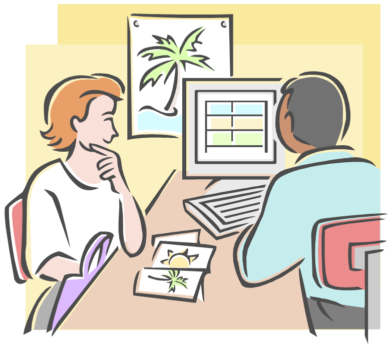
- **Deployment:**

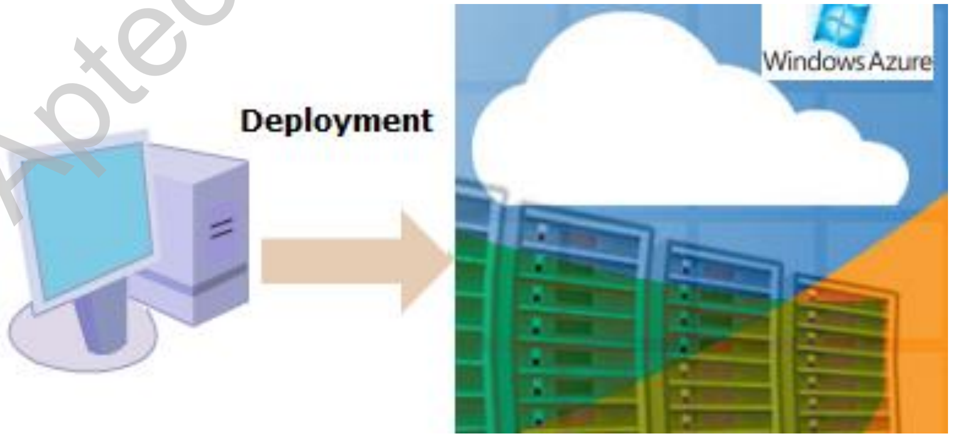
  - Is not just copying the application's files from one server to another.
  - May involve many other tasks involved in this process.
  - Has to be planned by developers with a strategy for Web applications before publishing.



Planning and Designing a Deployment Strategy 2-2

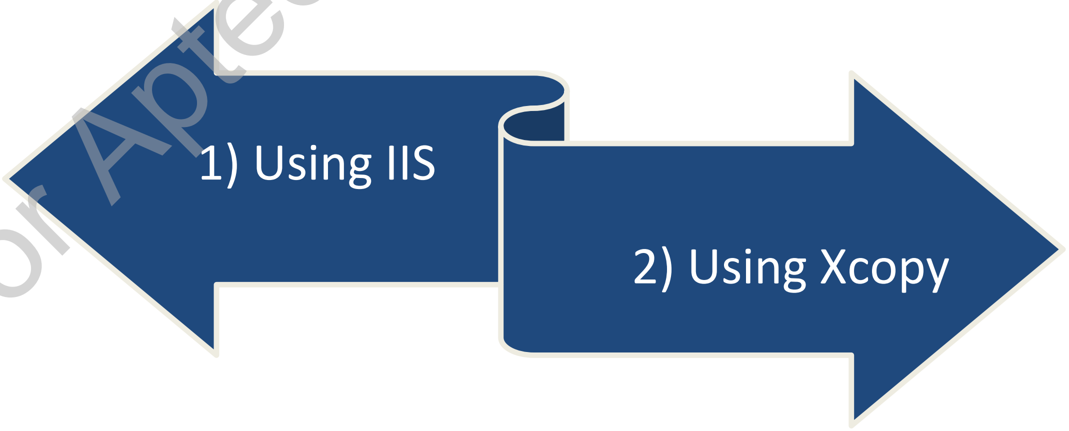
- **Developers:**
  - Select the suitable process based on the type of deployment.
  - Understand deployment options for Web applications in general before choosing the appropriate deployment strategy for a Windows Azure application.





Packaging and Deploying Using IIS and XCopy

- Visual Studio 2013 provides powerful support for packaging and deploying Web applications and services.
- In addition to Visual Studio 2013, you can also use several other approaches to package and deploy.
- Some of them are:



Using IIS 1-3

- Following are the steps to create Web deployment package and install it using IIS:

Step 1
- Run the command `%windir%\system32\inetsrv\appcmdadd backup "PreMsDeploy"` for backing up the IIS 7.0 server.

Step 2
- Launch the IIS Manager using **inetmgr** in the Run command window.

Step 3
- Expand **Sites** node and **Server** node and then, choose **Default Web Site/<MyApplication>**.

Step 4
- Next, in the **Manage Packages** section in the **Actions** pane, you need to select **Export Application**. This application will comprise the application chosen and the corresponding folders.

Step 5
- Select **ManageComponents**. In this, you will notice two prominent rows. While the first row is the application and the second, the provider name.

Using IIS 2-3

Step 6

- In the **Path** column, you need to add your script or database file path. The database will now be reflected in the package contents tree view.

Step 7

- To continue, click **Next**. Now, based on the providers that have been added, the parameters will be shown on screen.

Step 8

- Click **AddParameter**. Specify the parameter details for the connection string present in the Web.config file of the application.

Step 9

- Select a location for saving the package. Once these steps are done, you will notice that the wizard will complete the packaging process. It will then save the package to disk. With this, you will finish creating the package. This will now serve helpful while installing package.

Step 10

- Repeat steps 1 to 3. Then, select **Import Application** in the **Actions** pane.

Using IIS 3-3

Step 11	• Select the package that was created in step 9. In the **Install an Application Package** dialog box that is displayed, you will see the application, and the database.
Step 12	• On the **Parameters** page, type in a new application name (if desired), and a SQL connection string.
Step 13	• Click **Next** to install the package.

In 2009, Microsoft introduced a tool Web Deploy tool (MSdeploy.exe) that makes deployment of Web applications and Websites to IIS servers simpler and easier. Today, developers often prefer to use this tool.

Using Xcopy

The **Xcopy** command can be used for deploying an ASP.NET Web application.

Xcopy deployment is the process of installing a software application into the Microsoft Windows system by copying the files.

Advantages of Using Xcopy Deployment 1-3

- Following are some advantages of using Xcopy deployment:

You can use the drag-and-drop feature that is available in Microsoft Windows Explorer.

You can use the File Transfer Protocol (FTP) command for copying files.

The ASP.NET application does not require any registry related modifications.

The file transfer feature of Xcopy helps in simplifying the maintenance as well as deployment of ASP.NET Websites due to the absence of registry entries.

Advantages of Using Xcopy Deployment 2-3

- Xcopy style of deployment is not suitable in all cases.
- In case of line-of-business applications and large Websites, you should make the Website offline for some time.
- This can be done when the application assemblies and new content are going to be deployed.
- You need to perform maintenance tasks at a predefined time.

Advantages of Using Xcopy Deployment 3-3

- Following are the tips to minimize downtime for end users:

Tips

Make a new physical directory for the application you plan to update. Then, make a copy of all the new content to this directory.

Make changes to the configuration of the virtual directory for the application, to connect it to the new physical directory.
This will enable access to the new content in the new physical directory.

Choosing a Deployment Strategy for Windows Azure Applications 1-5

- Steps to perform to deploy a service to a Windows Azure Worker Role or Web Role are:

Step 1

Create a new Cloud Service and enable communication for role instances

Step 2

Create a package with the service files

Choosing a Deployment Strategy for Windows Azure Applications 2-5

Step 1 | **Create a new Cloud Service and enable communication for role instances**

- ❑ In Windows Azure, role instances are designed to run in a cloud service.
- ❑ An instance can use either internal or external connections depending on the type of communication that is required.

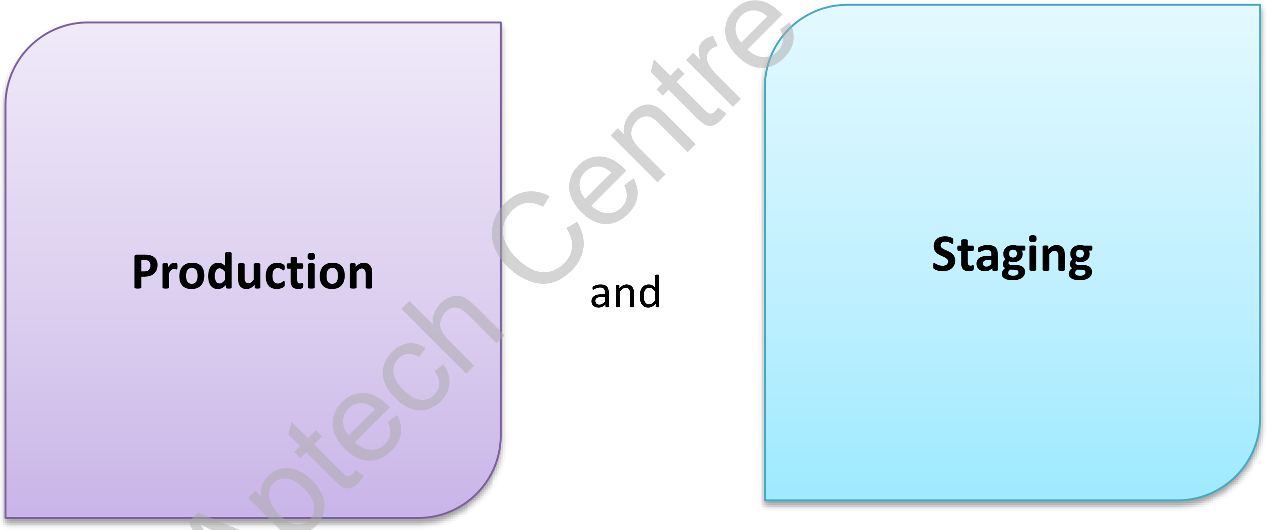
Choosing a Deployment Strategy for Windows Azure Applications 3-5

- Differences between the external and internal connections are:

External connections	Internal connections
• Mainly used for communicating with the clients that are external to your network.	• Mainly used for communicating with other role instances.
• Are also known as **input endpoints**. Input endpoints use the TCP, HTTPS, or HTTP protocols for connectivity.	• Are also known as **internal endpoints**.
• You need to associate a port to the input endpoint.	• Internal endpoints use the HTTP and TCP protocols for connectivity.
	• You can assign a dynamic IP address to the internal endpoint.

Choosing a Deployment Strategy for Windows Azure Applications 4-5

- You have two different deployments when working in cloud environments:



Choosing a Deployment Strategy for Windows Azure Applications 5-5

An application is deployed on the staging environment in the development stage.

After its production is complete, it is moved to the production environment after a thorough round of testing.

You can continue to develop the application or its components even after the application is deployed in the production environment.

Each of these two environments need an IP address and a DNS name for it to be accessible.

In the production environment, the DNS name is assigned for lifetime and is assigned during runtime.

In the staging environment, the DNS name is dynamically generated when the service is deployed.

The IP address is associated with the service until the time it is deployed.

Adding an Input Endpoint (External Connection) 1-2

- Following are the steps to add an input endpoint:

Step 1	Browse the Solution Explorer and locate the **ServiceDefinition.csdef** file.
Step 2	Open the file using a text editor, such as Notepad or even Visual Studio IDE.
Step 3	Locate the `Endpoints` element, which has an `InputEndpoint` element under the specific role element.
Step 4	Rename the endpoint to the name of your choice.
Step 5	Set the protocol to one of the following: HTTP, HTTPS, or TCP.
Step 6	Enter the port number for the role communication.
Step 7	Save changes in the file.

Adding an Input Endpoint (External Connection) 2-2

- Following code snippet shows how to add an input endpoint named `HttpEndPt` for HTTP protocol:

```
<ServiceDefinition name="TestService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008
/10/ServiceDefinition">
<WebRole name="TestWebRole">
<Endpoints>
<InputEndpoint name="HttpEndPt" protocol="http"
port="80" localPort="80" />
</Endpoints>
</WebRole>
</ServiceDefinition>
```

Adding an Internal Endpoint (Internal Connection) 1-2

- Following are the steps to add an internal endpoint:

Step 1: Edit the **ServiceDefinition.csdef** file and add the endpoints element for VM role, worker role, or a Web role. The element `InternalEndpoint` is under the specific element.

Step 2: Enter the name of the endpoint.

Step 3: Set the protocol for communication.

Step 4: Save changes in the file.

Adding an Internal Endpoint (Internal Connection) 2-2

- Following code snippet shows how to add an internal endpoint named `TestNewEndPoint` for HTTP protocol:

```
<ServiceDefinition name="TestService"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008
/10/ServiceDefinition">
<WebRole name="TestNewWebRole">
<Endpoints>
<InternalEndpoint name=" TestNewEndPoint"
protocol="http" port="1400"/>
</Endpoints>
</WebRole>
</ServiceDefinition>
```

Choosing a Deployment Strategy for Windows Azure Applications 1-2

Step 2 | **Create a package with the service files**

- Following table outlines the three key components that are essential to deploy an application as a cloud service in Azure:

File Type	Description
Service Definition	The cloud service definition file with extension **.csdef**, which defines the service model, including the number of roles.
Service Configuration	The cloud service configuration file with extension **.cscfg**, which provides configuration settings for the cloud service and individual roles, including the number of role instances.
Service Package	The service package with extension **.cspkg**, which contains the application code and the service definition file.

Choosing a Deployment Strategy for Windows Azure Applications 2-2

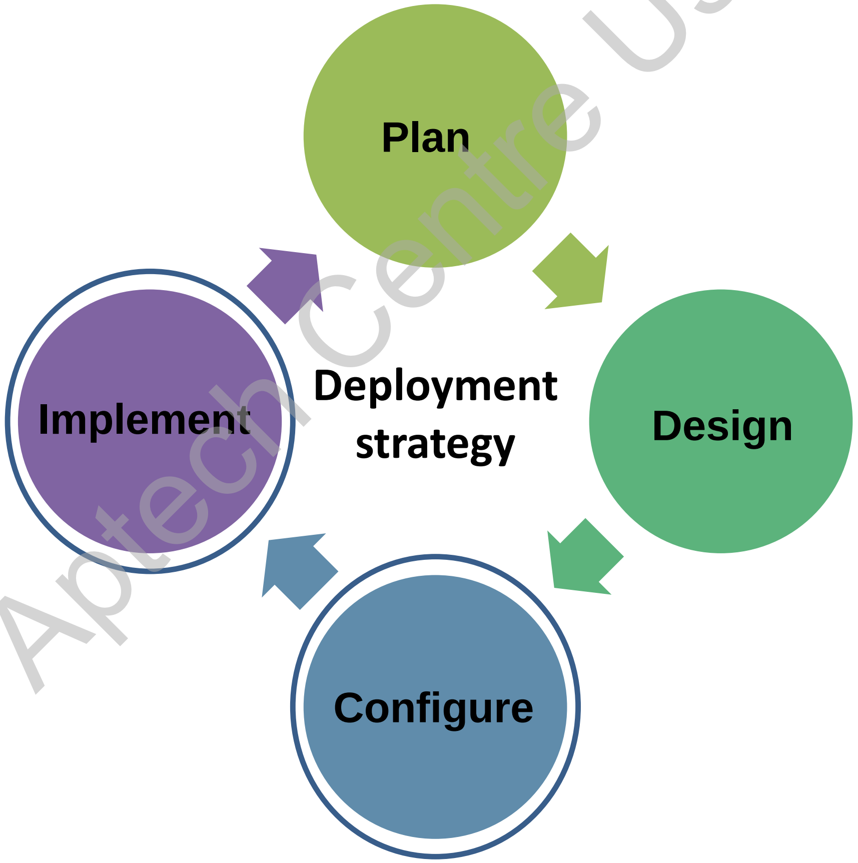
- The Azure SDK provides a command-line tool named **CSPack** for creating the application package.
- Following command shows how to package a sample application named **sampledemoweb** for deployment in Azure:

```
F:\Source Codes\Sampledemoweb> cspack
sampledemoweb\ServiceDefinition.csdef /out:sampledemoweb.cspkg
/role: sampledemoweb_WebRole;
sampledemoweb_WebRole /sites: sampledemoweb_WebRole;Web;
d:\sampledemoweb_WebRole\sampledemoweb_WebRole
```

- The outcome of this is **sampledemoweb.cspkg**, which is the service package to be deployed.

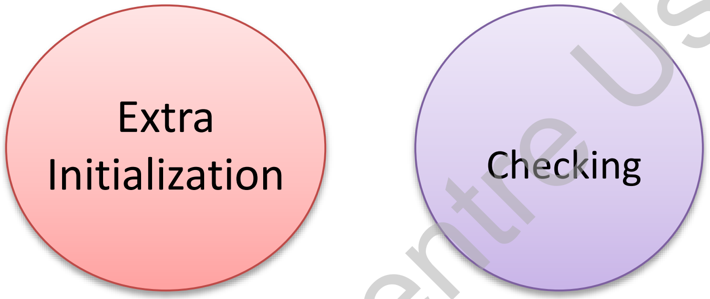
Configuring and Implementing Deployment

- Following are important phases in the deployment:



Switching from Production/Release Mode to Debug Mode 1-3

- The Debug mode has features such as:



- These features help you find and correct the issues that may occur in your application in Windows Azure environment.

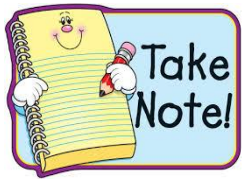
- Debug mode:
  - Slows the environment as issues are being tracked in different modes, such as checking syntaxes and also verifying the code is correct.
  - Runs slower than the released mode, in which no error check takes place.

Switching from Production/Release Mode to Debug Mode 2-3

- Following are the steps to switch from Debug or Release configuration:

Step 1
- Go to the Standard toolbar in the IDE.

Step 2
- Select **DebugMode** or **ReleaseMode** from the **Solution Configurations** list box.



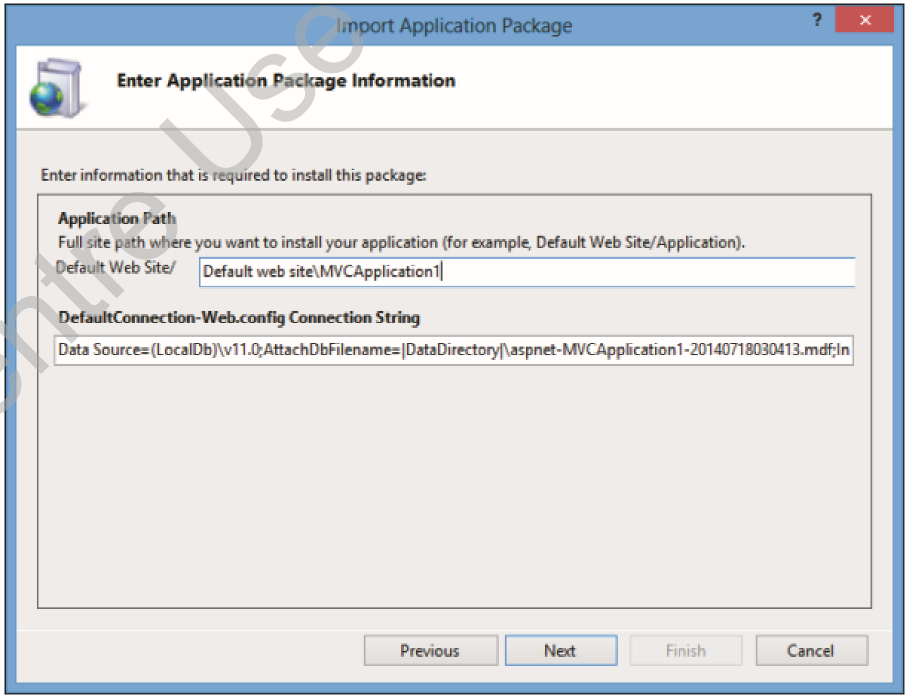
The Standard toolbar is not present in the Visual C# Express or Visual Basic Express. Instead, you can use the **Start Debugging** and press **F5** to Build Solution and press **F6** menu items for choosing the configuration.

Switching from Production/Release Mode to Debug Mode 3-3

- Regardless of which mode you use (Debug or Release):
  - You can use custom deployment parameters for configuration values that you may want to modify when the package is installed.
  - It can be done using the **SetParameters.xml** file.
  - This file is dynamically generated based on your Web application project file and other configuration files.

Use SetParameters to Set Up an IIS App Pool 1-3

- The **SetParameters.xml** file offers certain parameter values to the Web Deploy tool (MSDeploy.exe).
- You can make changes to the values in this file and forward it to Web Deploy for deploying the Web package.
- If you use IIS Manager for installing the package:
  - You must enter a value for the parameter in the **Enter Application Package Information** dialog box when you are prompted as shown in the figure.


Import Application Package
Enter Application Package Information
Previous
Next
Finish
Cancel

Enter information that is required to install this package:

Application Path

Full site path where you want to install your application (for example, Default Web Site/Application).

Default Web Site/ Default web site\MVCApplication1

DefaultConnection-Web.config Connection String

Data Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-MVCApplication1-20140718030413.mdf;In

Use SetParameters to Set Up an IIS App Pool 2-3

When you click **Next**, you will be prompted with the **Import Application Package** dialog box.

It confirms whether you have ASP.NET 4.0 installed on your system.

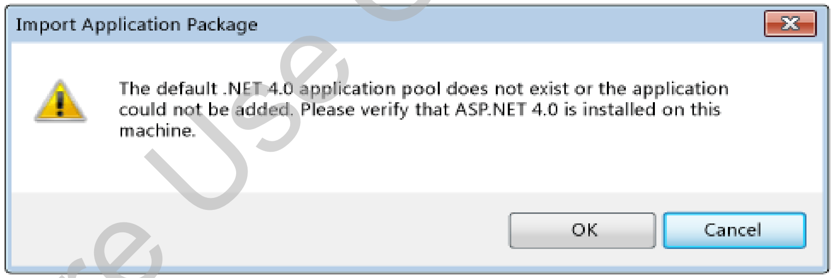
If you have it installed, click **OK**.

If you do not have it installed, you will need to click **Cancel** and then install ASP.NET 4.0.

After the package is installed, you will need to change the Application Pool to the ASP.NET v4.0 application pool in the IIS Manager.

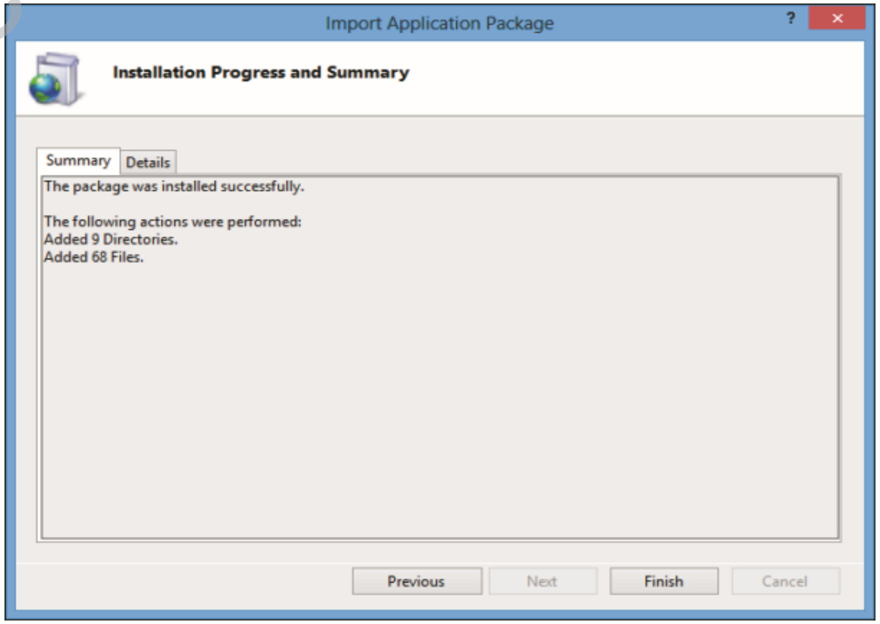
Use SetParameters to Set Up an IIS App Pool 3-3

- Figure shows the **Import Application Package** dialog box with the error:


Import Application Package

The default .NET 4.0 application pool does not exist or the application could not be added. Please verify that ASP.NET 4.0 is installed on this machine.

- After the package is deployed, the **Summary screen** is displayed as shown in figure:


Import Application Package

Installation Progress and Summary

The package was installed successfully.

The following actions were performed:
Added 9 Directories.
Added 68 Files.

- You will need to review the results and click **Finish**.

Configuring WCF Endpoints, Bindings, and Behaviors 1-2

- In addition to configuring Web applications, you can also configure Web services for deployment.
- **The behavior:**
  - The behavior element comprises a set of settings for the behavior of an endpoint.
  - The behavior is catalogued based on their name.
  - An endpoint can be linked to a behavior based on this name.
- From .NET Framework 4 onwards, behaviors and bindings are not required to have a name.

Configuring WCF Endpoints, Bindings, and Behaviors 2-2

- **In the deployment phase:**
  - Users get the flexibility of providing endpoint and service behavior data after configuring a WCF service with the help of configuration file.

- **The Binding:**

Transports	Protocol
• HTTP • Pipes • TCP • Message Queuing	• Safety • Dependability • Transaction flows

  - The binding is made of elements of binding where each element explains the fact or way of communicating an endpoint with the world.

Transforming Web.Config Using XSLT 1-2

- When you deploy an application, the settings are stored in the `Web.config` file.
- However, you may be making continues changes to the application in the staging environment and want to deploy these changes.
- The `Web.config` file in the staging environment will be different from the one in the production environment.
- To automatically deploy these changes in the `Web.config` file, you can set up a `Web.config` transform file.
- This file contains a number of settings that you can use to automate the deployment process.
- The transform file is associated with a build configuration of an application.

Transforming Web.Config Using XSLT 2-2

- Following are the steps for transforming the `Web.Config` using XSLT:

Step 1	• To create a transform file, you will need to create a build configuration first. You can create a build configuration using Configuration Manager.
Step 2	• Verify that you have an existing transform file. If it exists, it will be indicated with the name.
Step 3	• If the file does not exist, then you need to create one. Right-click `Web.config` in the Solution Explorer and select **Add Config Transforms**.
Step 4	• Open and edit the transform file with the required changes.
Step 5	• Save the transform file.

Configuring Azure Configuration Settings 1-3

- The Azure Configuration Settings are defined in the service configuration file.

- You can specify the following settings:

Azure Configuration Settings			
Number of role instances for deploying the values of any configuration settings	Every role in the service	Thumbprints for all certificates that are associated with the given role	The Virtual Network configuration information, in case the service is part of a virtual network

Configuring Azure Configuration Settings 2-3

- The service configuration file has the `.cscfg` default extension.
- By default, the configuration schema file is found in the following directory:

C:\Program Files\Microsoft SDKs\Windows Azure\.NET SDK\<version>\schemas

- The service configuration file has the following default format:

```
<ServiceConfigurationserviceName="<service-name>" osFamily="<osfamily-
number>" osVersion="<os-version>" schemaVersion="<schema-version>">
<Role …>
     …
</Role>
<NetworkConfiguration>
     …
</NetworkConfiguration>
</ServiceConfiguration>
```

Configuring Azure Configuration Settings 3-3

- Following table lists the attributes defined in the service configuration file:

Attribute	Description	Requirement
`serviceName`	Defines the name of the cloud service. This name must be same as the name specified in the service definition file.	Mandatory
`osFamily`	Defines which Guest OS will run on role instances in the cloud service.	Optional
`osVersion`	Defines the version of the Guest OS.	Optional
`schemaVersion`	Defines the version of the Service Configuration Schema.	Optional

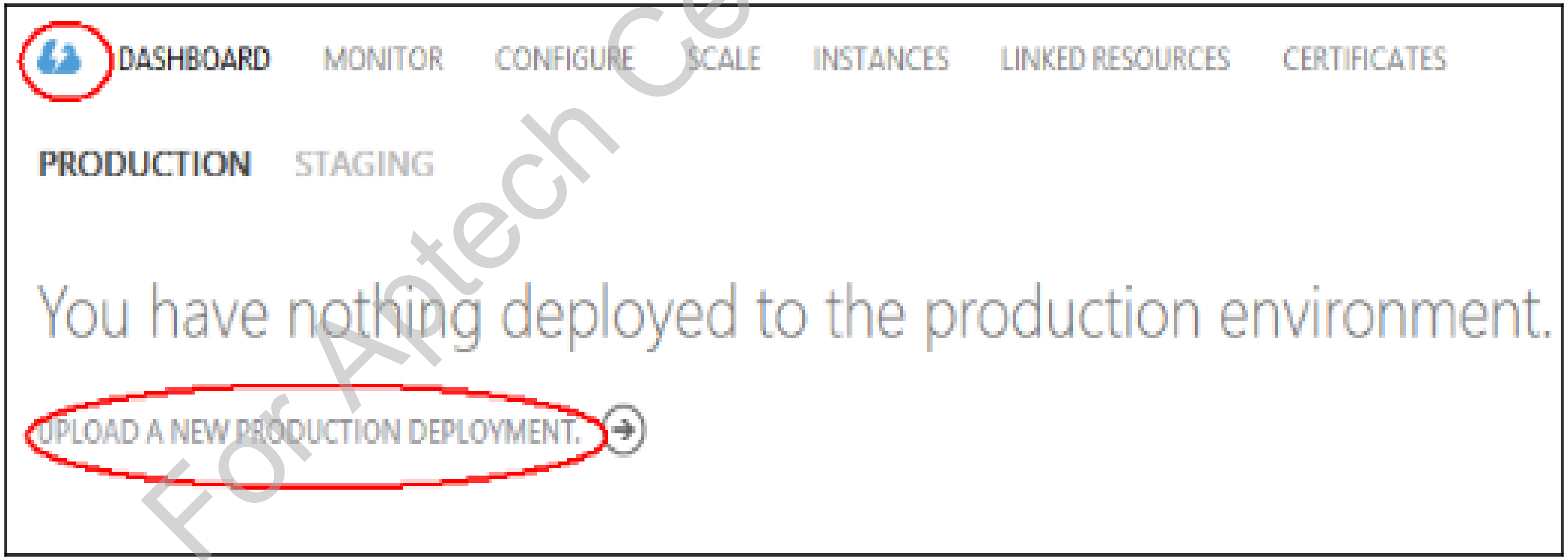
Implementing Deployment 1-4

- Deployment of a cloud service or application involves uploading and deploying the package and configuring to Windows Azure.

Implementing Deployment 2-4

- Following are the steps to upload the package in the Windows Azure management portal:

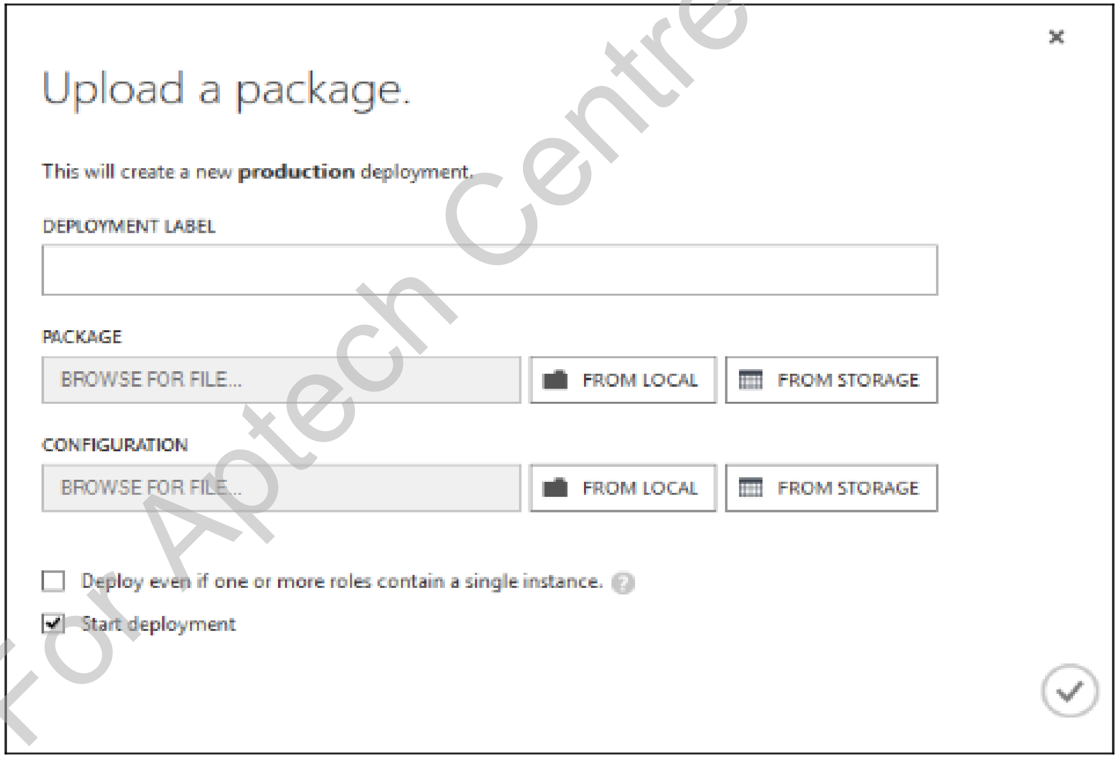
1 Select your application, or cloud service and click either the **Upload icon** or the **UPLOAD A NEW PRODUCTION DEPLOYMENT** link as shown in figure.



You have nothing deployed to the production environment.

Implementing Deployment 3-4

2 The **Upload a package** dialog box is displayed as shown in the following figure.


Upload a package.

This will create a new **production** deployment.

DEPLOYMENT LABEL

PACKAGE

BROWSE FOR FILE... FROM LOCAL FROM STORAGE

CONFIGURATION

BROWSE FOR FILE... FROM LOCAL FROM STORAGE

☐ Deploy even if one or more roles contain a single instance.

☑ Start deployment

Implementing Deployment 4-4

3	Specify a name in the **DEPLOYMENT LABEL** box.
4	Click **FROM LOCAL** or **FROM STORAGE** next to the **BROWSE FOR FILE** box under **Package** to select the service package file (.cspkg).
5	Similarly, under **Configuration** sections, select the service configure file (.cscfg) to be used.
6	If your cloud service includes any roles with only one instance, select the **Deploy even if one or more roles contain a single instance** check box. Then, click the **OK** checkmark to begin deployment.

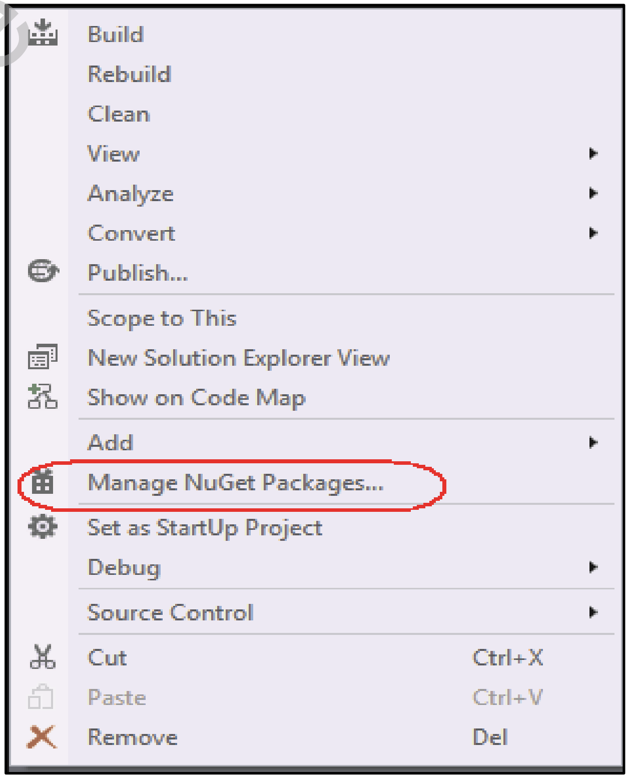
Using NuGet to Manage Packages

- **NuGet:**
  - NuGet is a package manager for .NET as well as other Microsoft development platforms.
  - Using NuGet client tools, you can produce and consume packages.
  - The NuGet Gallery, which is **nuget.org**, acts as the central package repository for all package authors and consumers who use NuGet.
  - The changes are automatically made when you add or remove a package.
  - When you add a package, the changes are made to the required files, such as `app.config` or `Web.config`.
  - If you remove a package, the changes are immediately made to ensure that there are no leftover files.

Creating and Configuring a NuGet Package 1-4

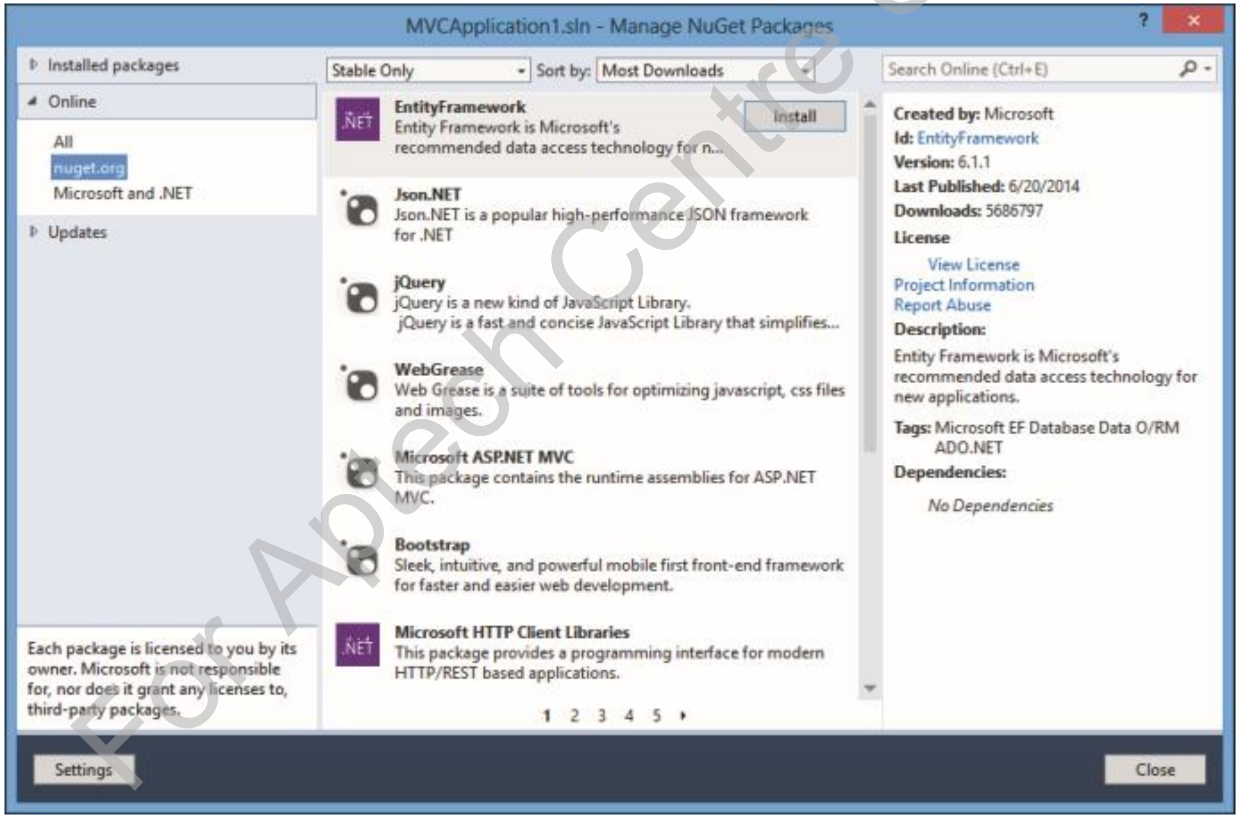
- NuGet runs as an extension in all versions of Visual Studio IDE starting from Visual Studio 2010 onwards.
- In the Solution Explorer pane:

Right-click the application name and select **Manage NuGet Packages** as shown in figure.



Creating and Configuring a NuGet Package 2-4

It displays the **Manage NuGet Packages** window as shown in the following figure.



EntityFramework
Entity Framework is Microsoft's recommended data access technology for n...

Json.NET
Json.NET is a popular high-performance JSON framework for .NET

jQuery
jQuery is a new kind of JavaScript Library.
jQuery is a fast and concise JavaScript Library that simplifies...

WebGrease
Web Grease is a suite of tools for optimizing javascript, css files and images.

Microsoft ASP.NET MVC
This package contains the runtime assemblies for ASP.NET MVC.

Bootstrap
Sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.

Microsoft HTTP Client Libraries
This package provides a programming interface for modern HTTP/REST based applications.

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Created by: Microsoft
Id: EntityFramework
Version: 6.1.1
Last Published: 6/20/2014
Downloads: 5686797
License
View License
Project Information
Report Abuse
Description:
Entity Framework is Microsoft's recommended data access technology for new applications.
Tags: Microsoft EF Database Data O/RM ADO.NET
Dependencies:
No Dependencies

Creating and Configuring a NuGet Package 3-4

- **Creating package:**
  - You can download and use the **NuGet Package Explorer** for creating packages.
  - Following are the basic steps for creating a package using Package Explorer:

Step 1
- Click **File** → **New** in the Package Explorer.

Step 2
- Click **File** → **Edit Package Metadata**.

Step 3
- Drag the contents of the package into the Package contents pane.

Step 4
- Click **OK** to place the file in the respective folder.

Step 5
- Save the package. Click **File** → **Save**.

Creating and Configuring a NuGet Package 4-4

- **Publish package:**
  - You need to publish the package after creating and saving the package.
  - Following are the steps to publish the package:

Step 1	• Click **File** → **Publish.** The **PublishPackage** dialog box will appear.
Step 2	• Enter the API key and hit **Publish**.

Connecting to a Local Repository Cache for NuGet 1-2

- In order to avoid making unnecessary downloads, NuGet stores a cache of packages that have been downloaded earlier.
- It helps NuGet:
  - To confirm before making any new downloads.
  - To download versions that are recent and not old versions, which are already stored in the cache.
- The cache is stored in the same format as the NuGet repository, so that it is easily accessible and the code can be reused.
- When you need the code, the NuGet Package Manager will access it from the cache location and pick up the packages in the cache.

Connecting to a Local Repository Cache for NuGet 2-2

- Following are the steps to add the cache as a NuGet repository:

Step 1

- Select **Tools → Options**.

Step 2

- In options, select **PackageManager** and then choose **PackageSources**.
- If no additional NuGet repositories have been set up, it will only display one entry.

Setting Up Your Own Package Repository

- NuGet displays packages that may be gathered from various package sources.

- Following are the steps for adding a package source:

Step 1	• Click the **Settings** button in the dialog box for launching the **Options** dialog box. Ensure that the **Package Sources** node is selected in the dialog box.
Step 2	• Next, type the source name and the folder path and click **Add**. You will now see the package source listed under the **All** node.
Step 3	• Select **Install** to install any package. NuGet will install the selected package as well as other packages that are depended on the given package.

Summary

- A Web application or Website project needs to be deployed or published to a Web server so that others can access and use the application.
- Visual Studio 2013 provides powerful support for packaging and deploying Web applications and services.
- In addition to Visual Studio 2013, you can also use IIS or XCopy for deployment.
- When you use Xcopy deployment, you need not install any special software on the Web server.
- Microsoft's Web Deploy tool (MSdeploy.exe) makes deployment of Web applications and Websites to IIS servers simpler and easier.
- In Windows Azure, role instances run in a cloud service.
- Cloud environments that are formed in Windows Azure can be deployed into various environments, staging environment and production environment.
- The Debug has features such as extra initialization and checking.
- The SetParameters.xml file is generated from the configuration files within the project and the Web application project file.