

# How-To: ASP.NET Web Deployment using Visual Studio

This tutorial series shows how to deploy (publish) an ASP.NET web application to Azure App Service Web Apps or to a third-party hosting provider using Visual Studio 2017. For information about the series, see [the first tutorial in the series](#).

For a current version of deploying to Azure, see [Create an ASP.NET Core web app in Azure](#).

## Overview

In this tutorial, you'll deploy an ASP.NET web application to Internet Information Server (IIS) on your local computer.

Generally when you develop an application, you run it and test it in Visual Studio. By default, web application projects in Visual Studio 2017 use IIS Express as the development web server. IIS Express behaves more like full IIS than the Visual Studio Development Server (also known as Cassini), which Visual Studio 2017 uses by default. But neither development web server works exactly like IIS. Consequently, an app could run and test correctly in Visual Studio but fail when it's deployed to IIS.

You can reliably test your application in two ways:

1. Deploy your application to IIS on your development computer using the same process that you'll use later to deploy it to your production environment.

You can configure Visual Studio to use IIS when you run a web project, but that wouldn't test your deployment process. This method validates your deployment process and that your application runs correctly under IIS.

2. Deploy your application to a test environment similar to your production environment.

The production environment for these tutorials is Web Apps in Azure App Service. The ideal test environment is an additional web app created in the Azure Service. Though it would be set up the same way as a production web app, you would only use it for testing.

Option 2 is the most reliable way to test. If you use option 2, you don't necessarily need to use option 1. However if you're deploying to a third-party hosting provider, option 2 might not be feasible or might be expensive, so this tutorial series shows both methods. Guidance for option 2 is provided in the [Deploying to the Production Environment](#) tutorial.

For more information about using web servers in Visual Studio, see [Web Servers in Visual Studio for ASP.NET Web Projects](#).

Reminder: If you receive an error message or something doesn't work as you go through the tutorial, be sure to check the [troubleshooting page](#).

## Download the Contoso University starter project

Download and install the Contoso University Visual Studio starter solution and project. This solution contains the completed tutorial.

[Download Starter Project](#)

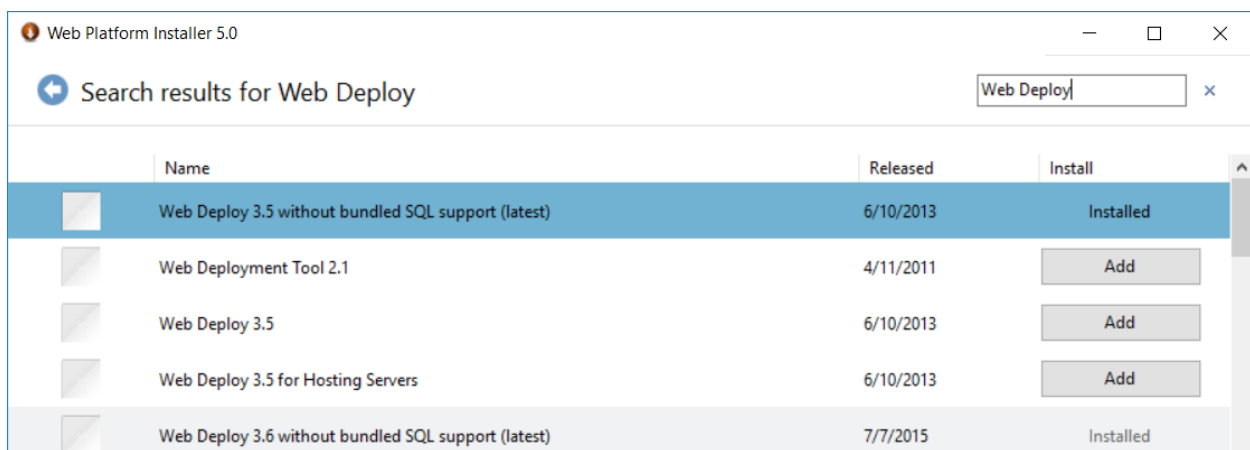
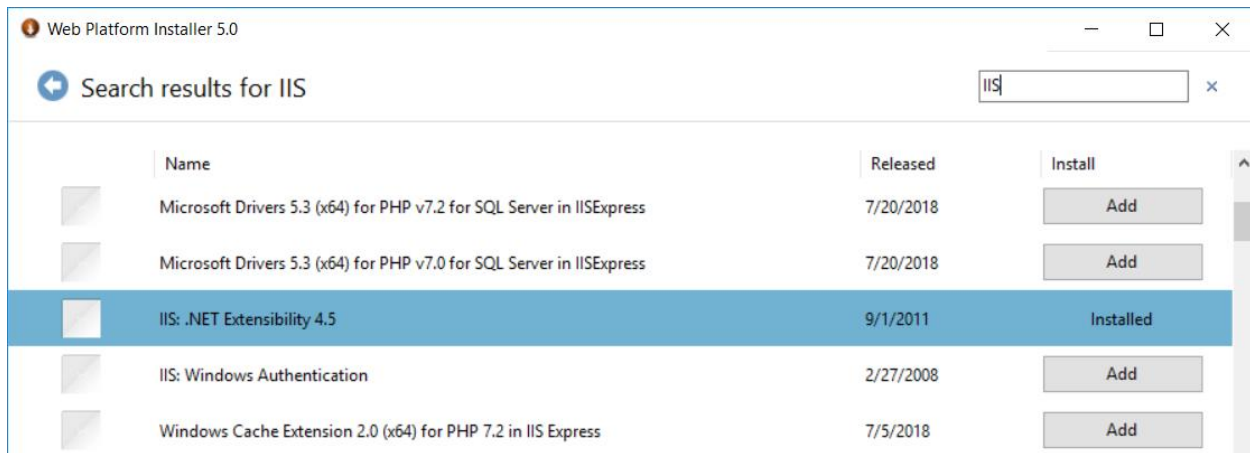
## Install IIS

To deploy to IIS on your development computer, confirm that IIS and Web Deploy are installed. By default, Visual Studio installs Web Deploy, but IIS isn't included in the default Windows 10, Windows 8, or Windows 7 configuration. If you've already installed IIS and the default application pool is already set to .NET 4, skip to [the next section](#).

1. It's recommended you use the [Web Platform Installer \(WPI\)](#) to install IIS and Web Deploy. WPI installs a recommended IIS configuration that includes IIS and Web Deploy prerequisites if necessary.

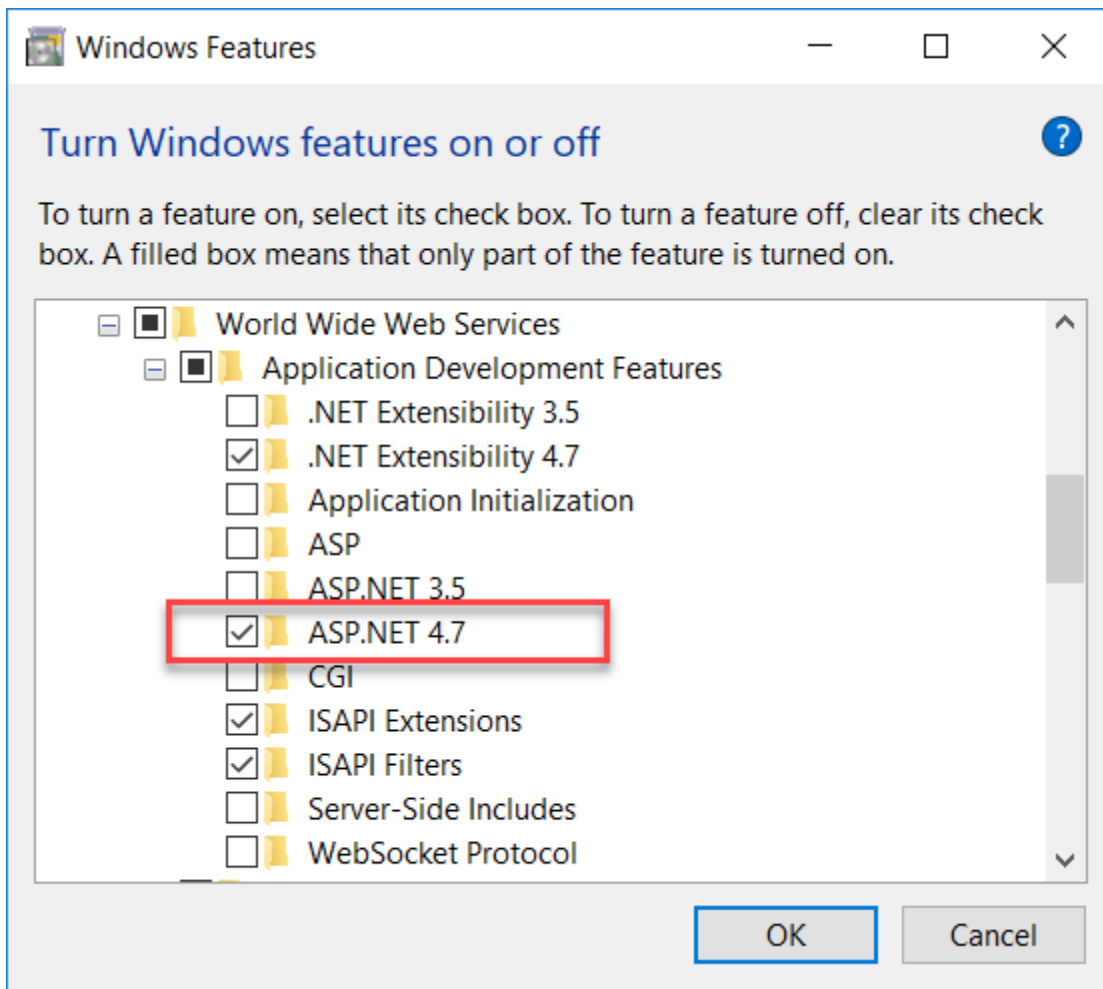
If you've already installed IIS, Web Deploy, or any of their required components, the WPI installs only what is missing.

- Use the Web Platform Installer to install IIS and Web Deploy:

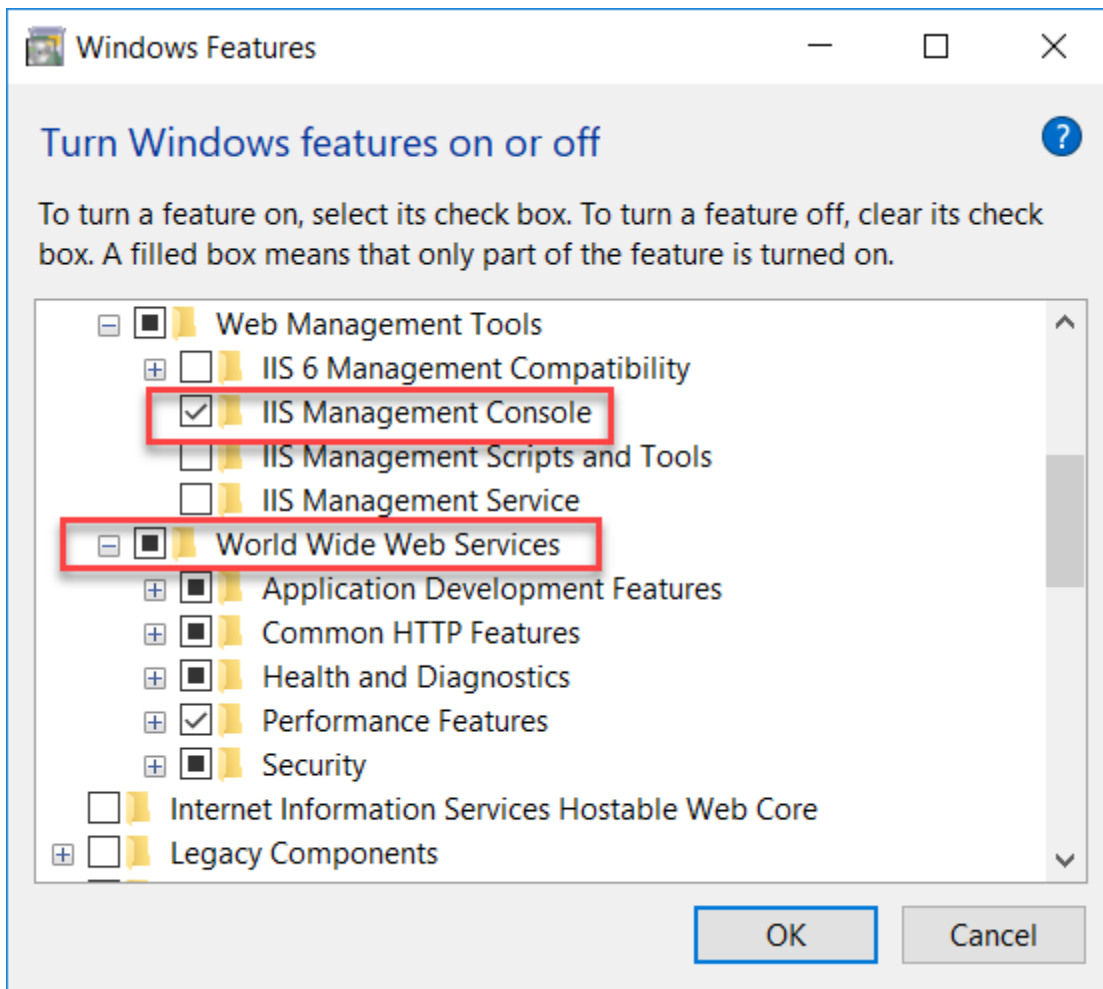


You'll see messages indicating that IIS 7 will be installed. The link works for IIS 8 in Windows 8; but for Windows 8 and later, go through the following steps to make sure that ASP.NET 4.7 is installed:

- Open **Control Panel > Programs > Programs and Features > Turn Windows features on or off**.
- Expand **Internet Information Services, World Wide Web Services, and Application Development Features**.
- Confirm that **ASP.NET 4.7** is selected.



2. Confirm that **World Wide Web Services** and **IIS Management Console** is selected. This installs IIS and IIS Manager.



3. Select **OK**. Dialog box messages indicating installation is taking place appear.

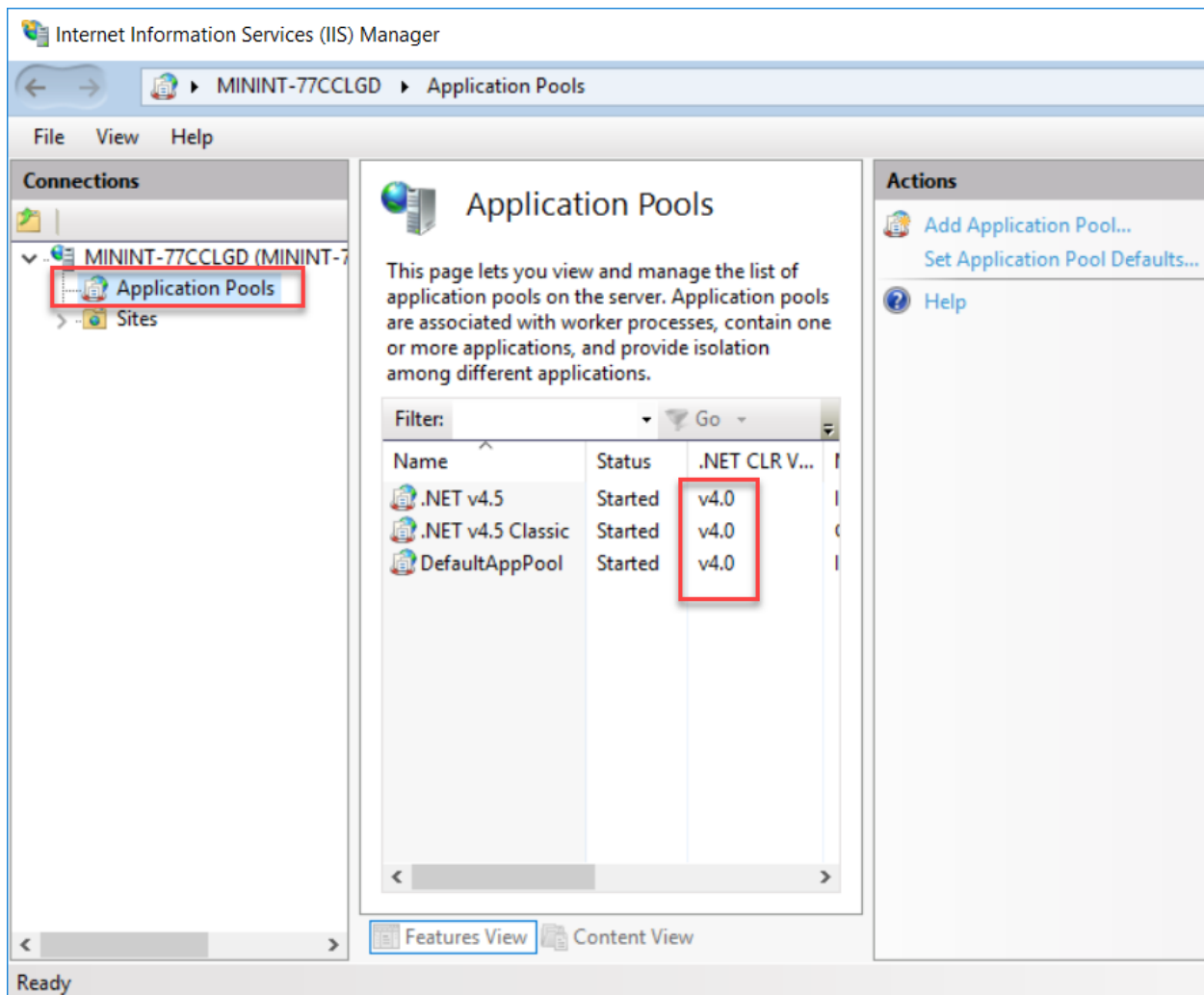
After installing IIS, run **IIS Manager** to make sure that the .NET Framework version 4 is assigned to the default application pool.

1. Press **WINDOWS+R** to open the **Run** dialog box.

(On Windows 8 or later, enter "run" on the **Start** page. In Windows 7, select **Run** from the **Start** menu. If **Run** isn't in the **Start** menu, right-click the taskbar, select **Properties**, select the **Start Menu** tab, select **Customize**, and select **Run command**.)

2. Enter "inetmgr" and select **OK**.

3. In the **Connections** pane, expand the server node and select **Application Pools**. In the **Application Pools** pane if **DefaultAppPool** is assigned to the .NET framework version 4 as in the following illustration, skip to the next section.



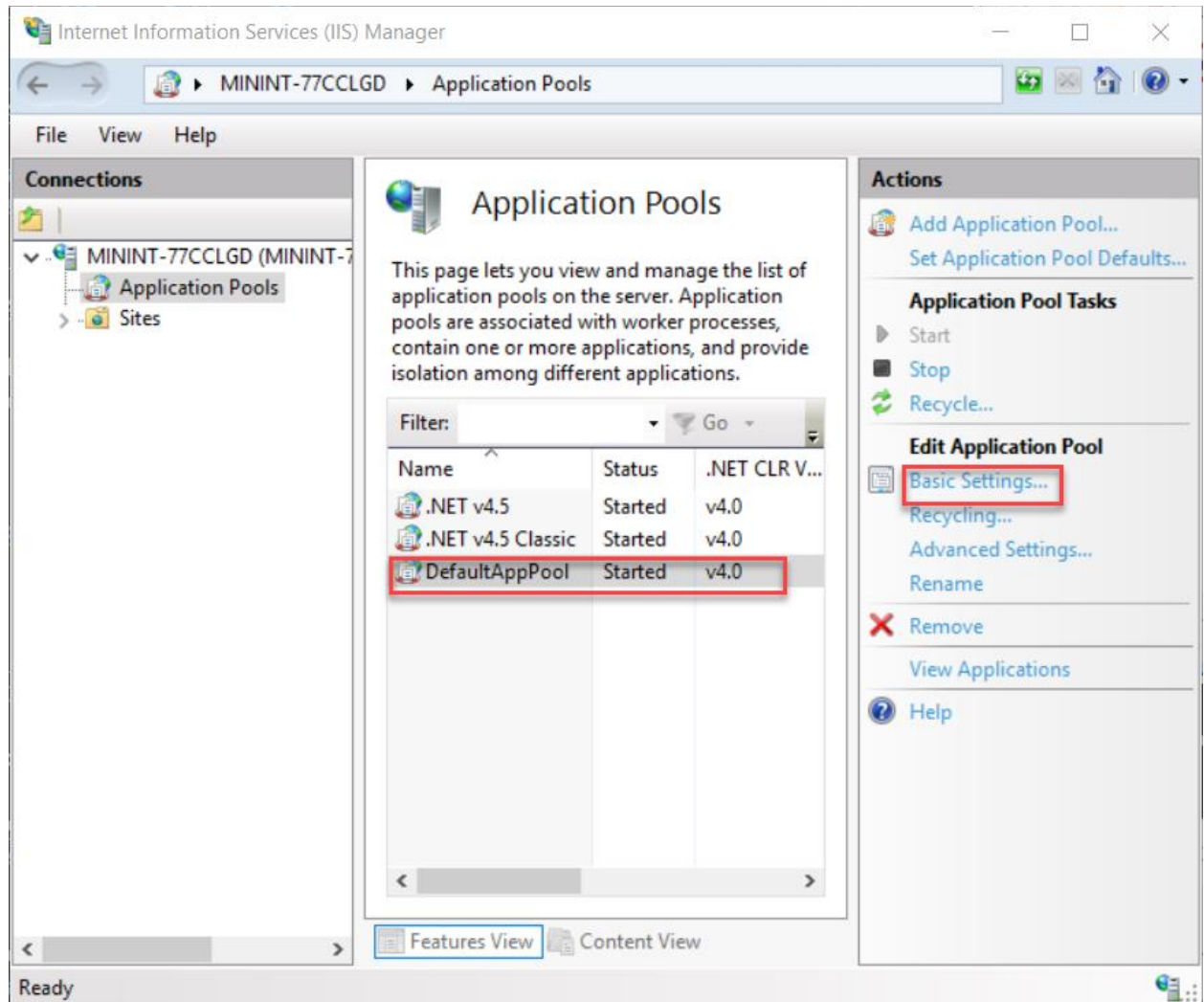
4. If you see only two application pools and both are set to .NET Framework 2.0, install ASP.NET 4 in IIS.

For Windows 8 or later, see the previous section's instructions for making sure that ASP.NET 4.7 is installed or see [How to install ASP.NET 4.5 on Windows 8 and Windows Server 2012](#). For Windows 7, open a command prompt window by right-clicking **Command Prompt** in the Windows **Start** menu and selecting **Run as Administrator**. Run [aspnet\\_regiis.exe](#) to install ASP.NET 4 in IIS using the following commands. (In 32-bit systems, replace "Framework64" with "Framework".)

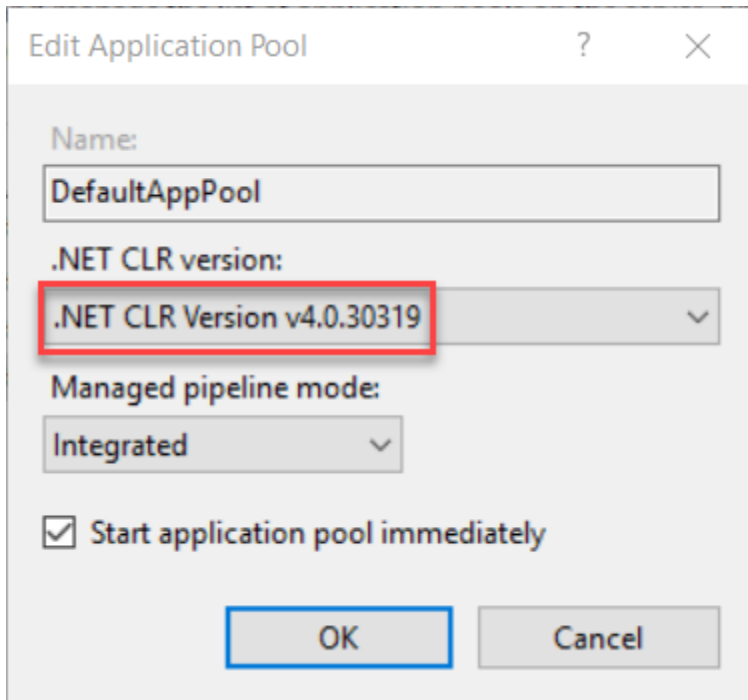
```
ConsoleCopy
cd %windir%\Microsoft.NET\Framework64\v4.0.30319
aspnet_regiis.exe -i
```

This command creates new application pools for the .NET Framework 4, but the default application pool will remain set to 2.0. You're deploying an application that targets .NET 4 to that application pool, so change the application pool to .NET 4.

5. If you closed **IIS Manager**, run it again, expand the server node, and select **Application Pools**.
6. In the **Application Pools** pane, select **DefaultAppPool**. In the **Actions** pane, select **Basic Settings**.



7. In the **Edit Application Pool** dialog box, change the **.NET CLR version** to **.NET CLR v4.0.30319**. Select **OK**.



You're now ready to publish a web application to IIS. First, however, create databases for testing.

## Install SQL Server Express

LocalDB isn't designed to work in IIS, so your test environment has to have SQL Server Express installed. If you're using Visual Studio 2010 SQL Server Express, it's already installed by default. If you're using Visual Studio 2012 or later, install SQL Server Express.

To install SQL Server Express, download and install it from [Download Center: Microsoft SQL Server 2017 Express edition](#).

On the first page of the SQL Server Installation Center, select **New SQL Server stand-alone installation or add features to an existing installation** and follow the instructions accepting the default choices. In the installation wizard, accept the default settings. For more information about installation options, see [Install SQL Server from the Installation Wizard \(Setup\)](#).

## Create SQL Server Express databases for the test environment

The Contoso University application has two databases:



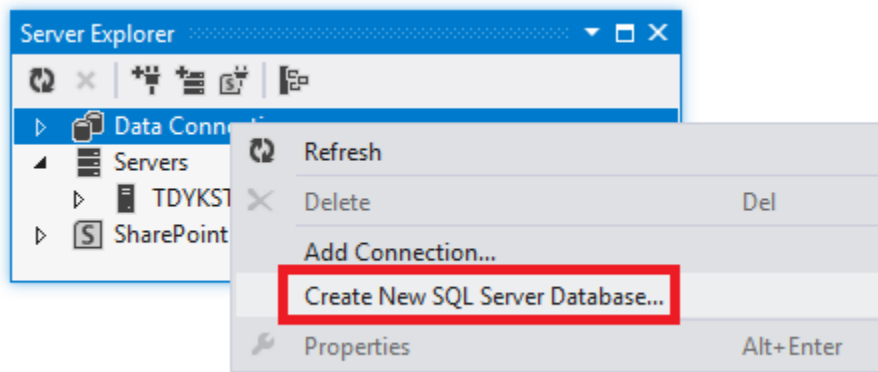
1. Membership database
2. Application database

You can deploy these databases to two separate databases or to a single database. Combining them makes database joins between them easier.

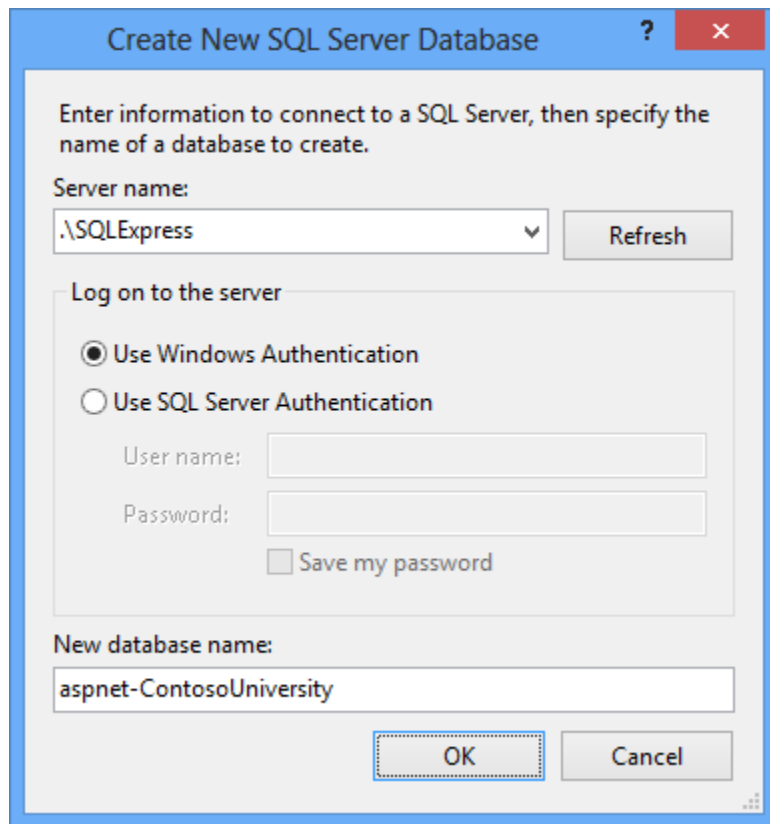
If you're deploying to a third-party hosting provider, your hosting plan might also give you a reason to combine them. For example, the provider might charge more for multiple databases or might not even allow more than one database.

In this tutorial, you'll deploy to two databases in the test environment and to one database in the staging and production environments.

From the **View** menu in Visual Studio, select **Server Explorer** (**Database Explorer** in Visual Web Developer). Right-click **Data Connections** and select **Create New SQL Server Database**.

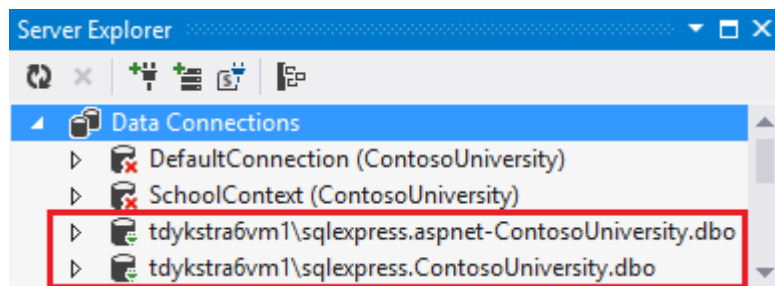


In the **Create New SQL Server Database** dialog box, enter ".\SQLExpress" in the **Server name** box and "aspnet-ContosoUniversity" in the **New database name** box. Select **OK**.



Follow the same procedure to create a new SQL Server Express School database named ContosoUniversity.

**Server Explorer** shows the two new databases.



## Create a grant script for the new databases

When the application runs in IIS on your development computer, the application uses the default application pool's credentials to access the database. However, by default, the application pool doesn't have permission to open the databases. This means you need to run a script to grant that permission. In this section, you'll create that script and run it later to make sure that the application can open the databases when it runs in IIS.

In a text editor, copy the following SQL commands into a new file and save it as *Grant.sql*.

#### SQLCopy

```
IF NOT EXISTS (SELECT name FROM sys.server_principals WHERE name = 'IIS
APPPool\DefaultAppPool')
BEGIN
    CREATE LOGIN [IIS APPPool\DefaultAppPool]
        FROM WINDOWS WITH DEFAULT_DATABASE=[master],
        DEFAULT_LANGUAGE=[us_english]
END
GO
CREATE USER [ContosoUniversityUser]
    FOR LOGIN [IIS APPPool\DefaultAppPool]
GO
EXEC sp_addrolemember 'db_owner', 'ContosoUniversityUser'
GO
```

In Visual Studio, open the Contoso University solution. Right-click the solution (not one of the projects), and select **Add**. Select **Existing Item**, browse to *Grant.sql*, and open it.

#### Note

This script is designed to work with SQL Server Express 2012 or later and with the IIS settings in Windows 10, Windows 8, or Windows 7 as they are specified in this tutorial. If you're using a different version of SQL Server or Windows, or if you set up IIS on your computer differently, changes to this script might be required. For more information about SQL Server scripts, see [SQL Server Books Online](#).

#### Note

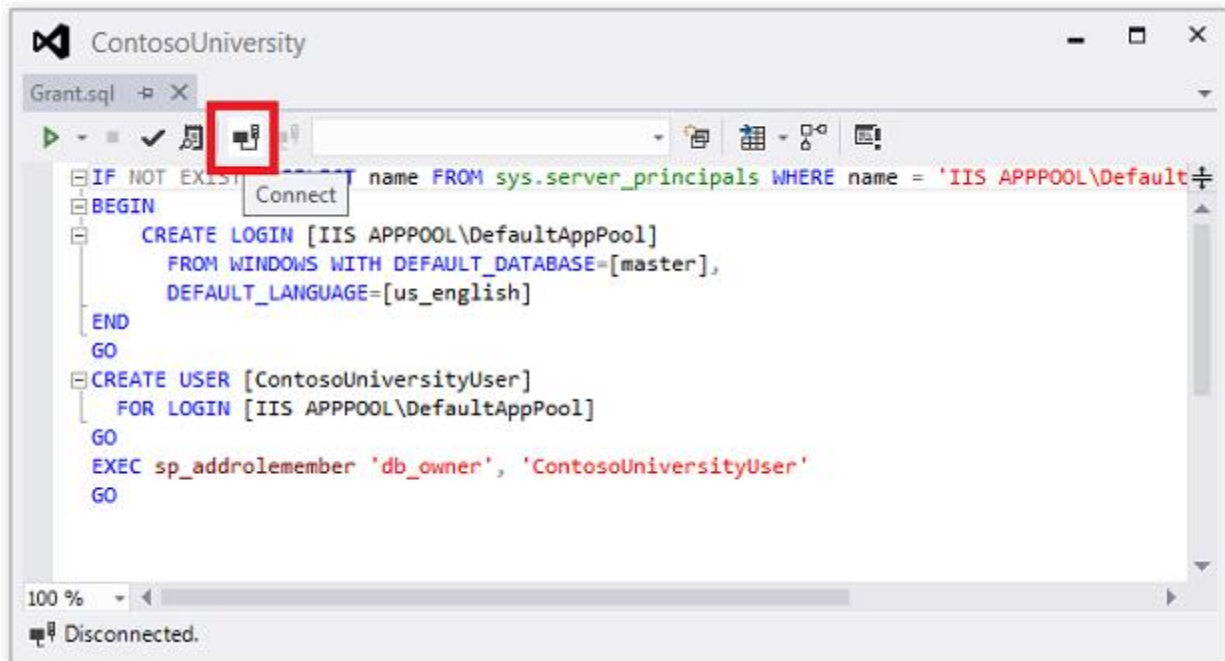
**Security Note** This script gives db\_owner permissions to the user that accesses the database at run time, which is what you'll have in the production environment. In some scenarios, you might want to specify a user that has full database schema update permissions only for deployment and specify for run time a different user that has permissions only to read and write data. For more information, see [Reviewing the Automatic Web.config Changes for Code First Migrations](#) later in this tutorial.

## Run the grant script in the application database

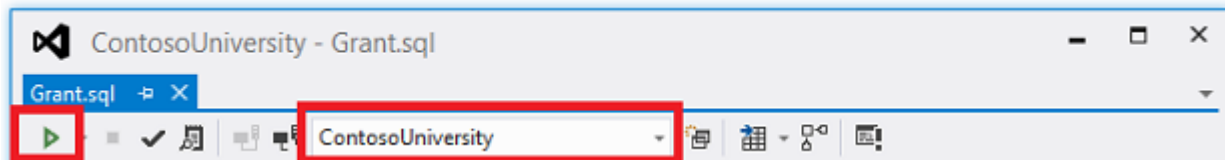
You can configure the publish profile to run the grant script in the membership database during deployment because that database deployment uses the [dbDacFx](#) provider. You can't run scripts during Code First Migrations deployment,

which is how you're deploying the application database. This means you have to manually run the script before deployment in the application database.

1. In Visual Studio, open the *Grant.sql* file that you created earlier.
2. Select **Connect**.



3. In the **Connect to Server** dialog box, enter `.\SQLExpress` as the **Server Name**. Select **Connect**.
4. In the database drop-down list select **ContosoUniversity**. Select **Execute**.



The default application pool identity now has sufficient permissions in the application database for Code First Migrations to create the database tables when the application runs.

## Publish to IIS

There are several ways you can deploy to IIS using Visual Studio and Web Deploy:

- Use Visual Studio one-click publish.

- Publish from the command line.
- Create a deployment package and install it using IIS Manager. The package has a .zip file with all the files and metadata required to install a site in IIS.
- Create a deployment package and install it using the command line.

The process you went through in the previous tutorials to set up Visual Studio to automate deployment tasks applies to all of these methods. In these tutorials, you'll use the first two methods. For information about using deployment packages, see [Deploying a web application by creating and installing a web deployment package](#) in the Web Deployment Content Map for Visual Studio and ASP.NET.

Before publishing, make sure that you're running Visual Studio in administrator mode. If you don't see **(Administrator)** in the title bar, close Visual Studio. In the Windows 8 (or later) **Start** page or the Windows 7 **Start** menu, right-click the Visual Studio icon and select **Run as Administrator**. Administrator mode is only required for publishing when you're publishing to IIS on the local computer.

## Create the publish profile

1. In **Solution Explorer**, right-click the **ContosoUniversity** project (not the **ContosoUniversity.DAL** project). Select **Publish**. The **Publish** page appears.
2. Select **New Profile**. The **Pick a publish target** dialog box appears.
3. Select **IIS, FTP, etc.** Select **Create Profile**. The **Publish** wizard appears.

The screenshot shows the 'Publish' dialog box in Visual Studio. The title bar says 'Publish' with a question mark and a close button. The main area has a left sidebar with 'Connection' (highlighted in blue) and 'Settings'. The main content area is titled 'CustomProfile \*'. It contains the following fields and controls:

- Publish method:** A drop-down menu with 'Web Deploy' selected.
- Server:** A text box with placeholder text 'e.g. server.contoso.com'.
- Site name:** A text box with placeholder text 'e.g. www.contoso.com or Default Web Site/MyApp'.
- User name:** A text box.
- Password:** A text box.
- Save password:** An unchecked checkbox.
- Destination URL:** A text box with placeholder text 'e.g. http://www.contoso.com'.
- Validate Connection:** A button.

At the bottom right, there are four buttons: '< Prev', 'Next >', 'Save', and 'Cancel'.

4. From the **Publish method** drop-down menu, select **Web Deploy**.
5. For **Server**, enter *localhost*.
6. For **Site name**, enter *Default Web Site/ContosoUniversity*.
7. For **Destination URL**, enter *http://localhost/ContosoUniversity*.

The **Destination URL** setting isn't required. When Visual Studio finishes deploying the application, it automatically opens your default browser to this URL. If you don't want the browser to open automatically after deployment, leave this box blank.

8. Select **Validate Connection** to verify that the settings are correct and you can connect to IIS on the local computer.

A green check mark verifies that the connection is successful.

The screenshot shows the 'Publish' dialog box in Visual Studio, specifically the 'CustomProfile \*' tab. The 'Publish method' is set to 'Web Deploy'. The 'Server' is 'localhost' and the 'Site name' is 'Default Web Site/ContosoUniversity'. The 'Destination URL' is 'http://localhost/ContosoUniversity'. A 'Validate Connection' button with a green checkmark is visible. The 'Connection' tab is selected on the left, and the 'Settings' sub-tab is active. The 'User name' and 'Password' fields are empty, and the 'Save password' checkbox is unchecked. Navigation buttons at the bottom include '< Prev', 'Next >', 'Save', and 'Cancel'.

Publish

Connection

Settings

CustomProfile \*

Publish method: Web Deploy

Server: localhost

Site name: Default Web Site/ContosoUniversity

User name:

Password:

☐ Save password

Destination URL: http://localhost/ContosoUniversity

Validate Connection ✓

< Prev Next > Save Cancel

9. Select **Next** to advance to the **Settings** tab.
10. The **Configuration** drop-down box specifies the build configuration to deploy. Leave it set to the default value of **Release**. You won't be deploying Debug builds in this tutorial.
11. Expand **File Publish Options**. Select **Exclude files from the App\_Data folder**.

In the test environment, the application accesses the databases that you created in the local SQL Server Express instance, not the .mdf files in the *App\_Data* folder.

12. Leave the **Precompile during publishing** and **Remove additional files at destination** check boxes cleared.

The screenshot shows the 'Publish' dialog box in Visual Studio, with the 'Settings' tab selected. The 'Configuration' dropdown is set to 'Release'. Under 'File Publish Options', the 'Databases' section is expanded. It contains two database entries: 'DefaultConnection' and 'SchoolContext'. Each entry has a text box for the connection string, a checkbox for 'Use this connection string at runtime (update destination web.config)', and a checkbox for 'Execute Code First Migrations (runs on application start)'. The 'DefaultConnection' entry has a checked runtime checkbox and an unchecked migrations checkbox. The 'SchoolContext' entry has both runtime and migrations checkboxes checked. At the bottom are buttons for '< Prev', 'Next >', 'Save', and 'Cancel'.

Publish

Connection

Settings

Configuration: Release

File Publish Options

Databases

DefaultConnection

Data Source=.\SQLExpress;Initial Catalog=aspnet-ContosoUniversity;Integr...

☒ Use this connection string at runtime (update destination web.config)

☐ Update database [Configure database updates](#)

SchoolContext

Data Source=.\SQLExpress;Initial Catalog=ContosoUniversity;Integrated Se...

☒ Use this connection string at runtime (update destination web.config)

☒ Execute Code First Migrations (runs on application start)

< Prev Next > Save Cancel

Precompiling is an option that is useful mainly for large sites. It can reduce startup time the first time a page is requested after the site is published.

You don't need to remove additional files since this is your first deployment and there won't be any files in the destination folder yet.

### Note

If you select **Remove additional files at destination** for a subsequent deployment to the same site, make sure that you use the preview feature so that you see in advance which files will be deleted before you deploy. The expected behavior is that Web Deploy will delete files on the destination server that you have deleted in your project. However, the entire folder structure under the source and destination folders is compared; and in some scenarios, Web Deploy might delete files you don't want to delete.



For example if you have a web application in a subfolder on the server when you deploy a project to the root folder, the subfolder will be deleted. You might have one project for the main site at contoso.com and another project for a blog at contoso.com/blog. The blog application is in a subfolder. If you select **Remove additional files at destination** when you deploy the main site, the blog application will be deleted.

For another example, your App\_Data folder might get deleted unexpectedly. Certain databases such as SQL Server Compact store database files in the App\_Data folder. After the initial deployment, you don't want to keep copying the database files in subsequent deployments, so you select **Exclude App\_Data** on the Package/Publish Web tab. After you do that if you have **Remove additional files at destination** selected, your database files and the App\_Data folder itself will be deleted the next time you publish.

## Configure deployment for the membership database

The following steps apply to the **DefaultConnection** database in the dialog box's **Databases** section.

1. In the **Remote connection string** box, enter the following connection string that points to the new SQL Server Express membership database.

```
ConsoleCopy
Data Source=.\SQLEXPRESS;Initial Catalog=aspnet-
ContosoUniversity;Integrated Security=True
```

The deployment process puts this connection string in the deployed Web.config file because **Use this connection string at runtime** is selected.

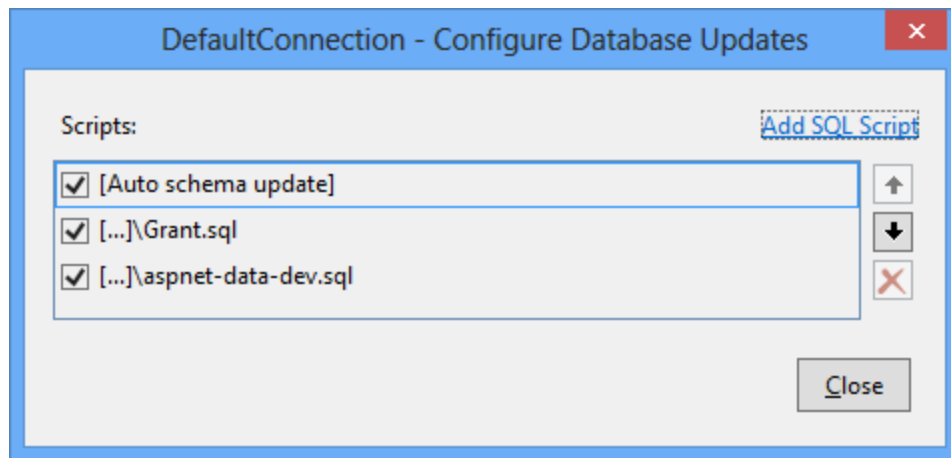
You can also get the connection string from **Server Explorer**. In **Server Explorer**, expand **Data Connections** and select the **<machinename>\sqlexpress.aspnet-ContosoUniversity** database, then from the **Properties** window copy the **Connection String** value. That connection string will have one additional setting that you can delete: **Pooling=False**.

2. Select **Update database**.

This causes the database schema to be created in the destination database during deployment. In next steps, you specify the additional scripts that you

need to run: one to grant database access to the default application pool and one to deploy data.

3. Select **Configure database updates**.
4. In the **Configure Database Updates** dialog box, select **Add SQL Script**. Navigate to the *Grant.sql* script that you saved earlier in the solution folder.
5. Repeat the process to add the *aspnet-data-dev.sql* script.



6. Select **Close**.

## Configure deployment for the application database

When Visual Studio detects an Entity Framework DbContext class, it creates an entry in the **Databases** section that has an **Execute Code First Migrations** check box instead of an **Update Database** check box. For this tutorial, you'll use that check box to specify Code First Migrations deployment.

In some scenarios, you might be using a DbContext database but you want to use the dbDacFx provider instead of Migrations to deploy the database. In that case, see [How do I deploy a Code First database without Migrations?](#) in the ASP.NET Web Deployment FAQ on MSDN.

The following steps apply to the **SchoolContext** database in the dialog box's **Databases** section.

1. In the **Remote connection string** box, enter the following connection string that points to the new SQL Server Express application database.

ConsoleCopy

```
Data Source=.\SQLEXPRESS;Initial Catalog=ContosoUniversity;Integrated Security=True
```

The deployment process puts this connection string in the deployed Web.config file because **Use this connection string at runtime** is selected.

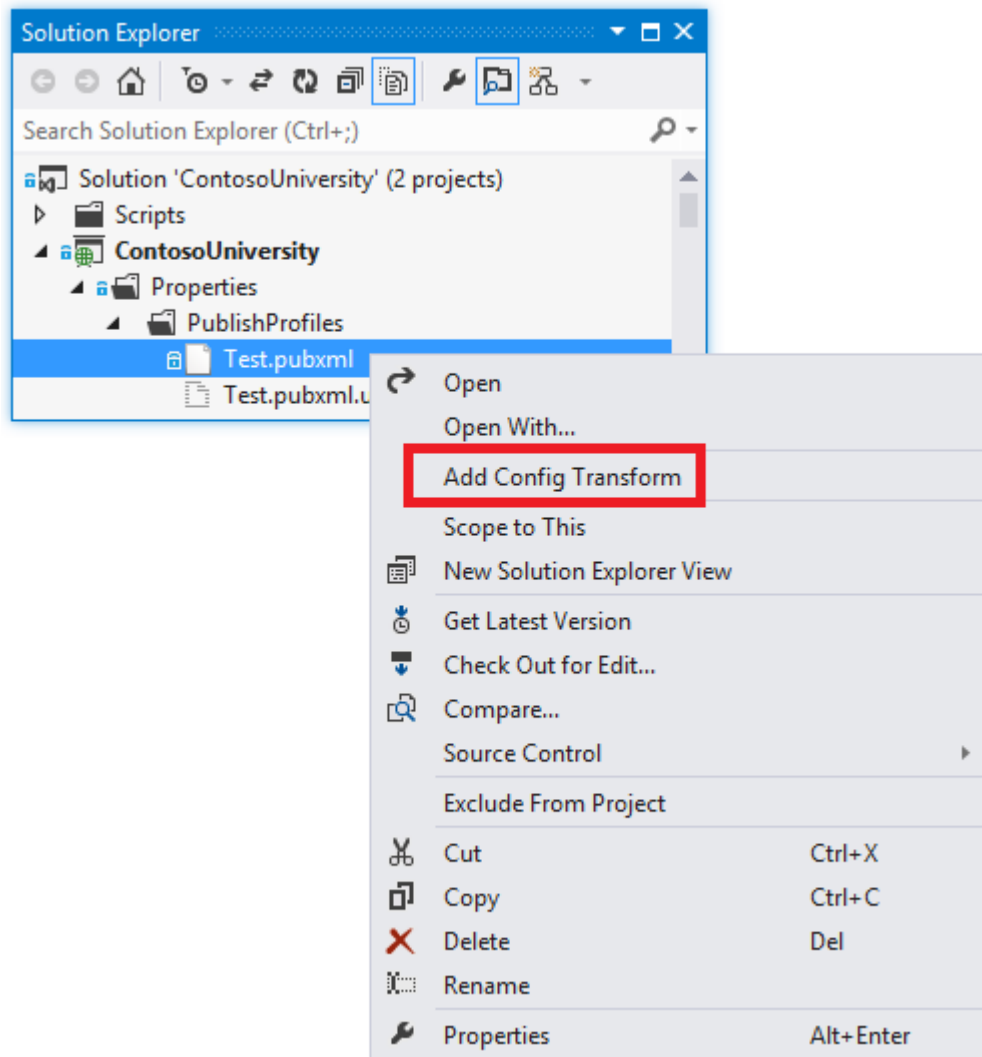
You can also get the application database connection string from **Server Explorer** in the same way you got the membership database connection string.

2. Select **Execute Code First Migrations (runs on application start)**.

This option causes the deployment process to configure the deployed Web.config file to specify the `MigrateDatabaseToLatestVersion` initializer. This initializer automatically updates the database to the latest version when the application accesses the database for the first time after deployment.

## Configure publish profile transforms

1. Select **Close**. Select **Yes** when you are asked if you want to save changes.
2. In **Solution Explorer**, expand **Properties**, expand **PublishProfiles**.
3. Right-click *CustomProfile.pubxml* and rename it *Test.pubxml*.
4. Right-click *Test.pubxml*. Select **Add Config Transform**.



Visual Studio creates the *Web.Test.config* transform file and opens it.

5. In the *Web.Test.config* transform file, insert the following code immediately after the opening configuration tag.

```
XMLCopy
<appSettings>
  <add key="Environment" value="Test" xdt:Transform="SetAttributes"
xdt:Locator="Match(key)"/>
</appSettings>
```

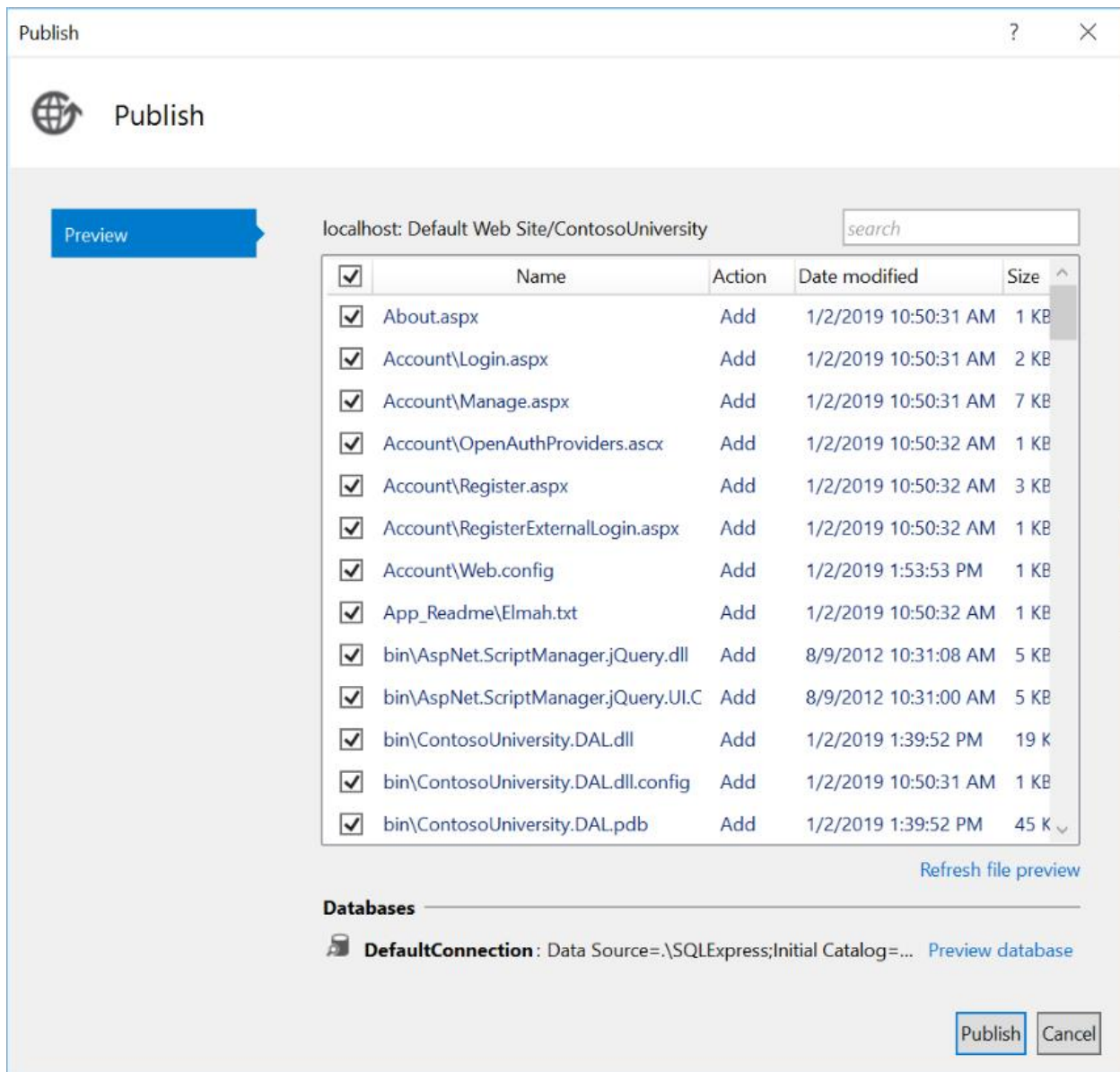
When you use the Test publish profile, this transform sets the environment indicator to "Test". In the deployed site, you'll see "(Test)" after the "Contoso University" H1 heading.

6. Save and close the file.
7. Right-click the *Web.Test.config* file and select **Preview Transform** to make sure that the transform you coded produces the expected changes.

The **Web.config Preview** window shows the result of applying both the *Web.Release.config* transforms and the *Web.Test.config* transforms.

## **Preview the deployment updates**

1. Open the **Publish Web** wizard again (right-click the ContosoUniversity project, select **Publish**, then **Preview**).
2. In the **Preview** dialog box, select **Start Preview** to see a list of the files that will be copied.

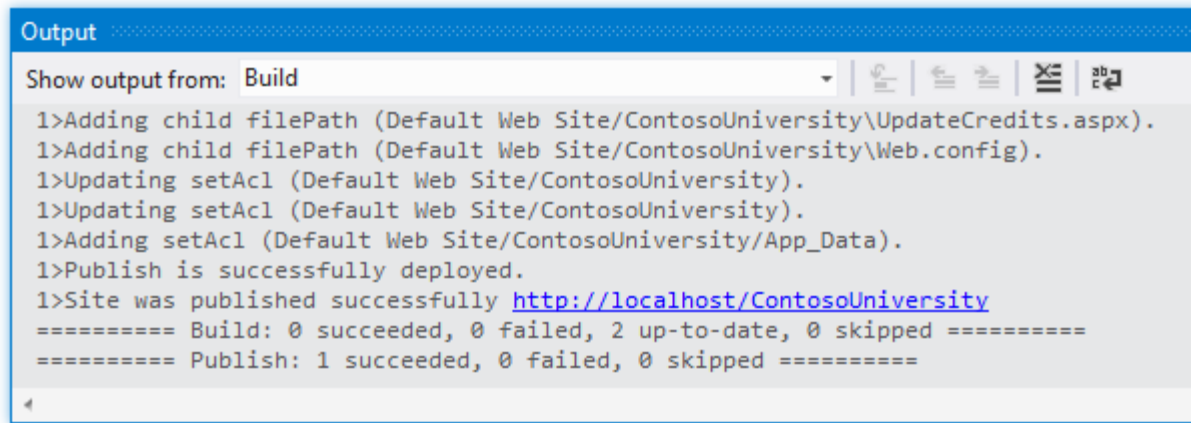


You can also select the **Preview database** link to see the scripts that will run in the membership database. (No scripts are run for Code First Migrations deployment, so there's nothing to preview for the application database.)

### 3. Select **Publish**.

If Visual Studio is not in administrator mode, you might get a permissions error message. In that case, close Visual Studio, open it in administrator mode, and try to publish again.

If Visual Studio is in administrator mode, the **Output** window reports successful build and publish.



```
Output
Show output from: Build
1>Adding child filePath (Default Web Site/ContosoUniversity\UpdateCredits.aspx).
1>Adding child filePath (Default Web Site/ContosoUniversity\Web.config).
1>Updating setAcl (Default Web Site/ContosoUniversity).
1>Updating setAcl (Default Web Site/ContosoUniversity).
1>Adding setAcl (Default Web Site/ContosoUniversity/App_Data).
1>Publish is successfully deployed.
1>Site was published successfully http://localhost/ContosoUniversity
===== Build: 0 succeeded, 0 failed, 2 up-to-date, 0 skipped =====
===== Publish: 1 succeeded, 0 failed, 0 skipped =====
```

If you entered the URL in the **Destination URL** box on the publish profile **Connection** tab, the browser automatically opens to the Contoso University Home page running in IIS on your computer.

## Test in the test environment

Notice that the environment indicator shows "(Test)" instead of "(Dev)," which shows that the *Web.config* transformation for the environment indicator was successful.

Run the **Instructors** page to verify that Code First seeded the database with instructor data. When you select this page, it may take a few minutes to load because Code First creates the database and then runs the seed method. (It didn't do that when you were on the home page because the application didn't try to access the database yet.)

Select the **Students** tab to verify that the deployed database has no students.

Select **Add Students** from the **Students** menu. Add a student, and then view the new student in the **Students** page. This verifies that you can successfully write to the database.


From the **Courses** menu, select **Update Credits**. The **Update Credits** page requires administrator permissions, so the **Log In** page is displayed. Enter the administrator account credentials that you created earlier ("admin" and "devpwd"). The **Update Credits** page is displayed. This verifies that the administrator account that you created in the previous tutorial was correctly deployed to the test environment.

Verify that an *ELMAH* folder exists in the *c:\inetpub\wwwroot\ContosoUniversity* folder with only the placeholder file in it.

# Review the automatic Web.config changes for Code First Migrations


Open the *Web.config* file in the deployed application at *C:\inetpub\wwwroot\ContosoUniversity* and you can see where the deployment process configured Code First Migrations to automatically update the database to the latest version.

```
<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlCeConnectionFactory
    <parameters>
      <parameter value="System.Data.SqlServerCe.4.0" />
    </parameters>
  </defaultConnectionFactory>
  <contexts>
    <context type="ContosoUniversity.DAL.SchoolContext, ContosoUniversity.DAL">
      <databaseInitializer type="System.Data.Entity.MigrateDatabaseToLatestVersion`2[
        <parameters>
          <parameter value="SchoolContext_DatabasePublish" />
        </parameters>
      </databaseInitializer>
    </context>
  </contexts>
</entityFramework>
```



The deployment process also created a new connection string for Code First Migrations to use exclusively for updating the database schema:

```
<connectionStrings>
  <add name="DefaultConnection" providerName="System.Data.SqlClient" connectionString="Data
  <add name="SchoolContext" providerName="System.Data.SqlClient" connectionString="Data Sou
  <add name="SchoolContext_DatabasePublish" connectionString="Data Source=.\SQLEXPRESS;Init
</connectionStrings>
```



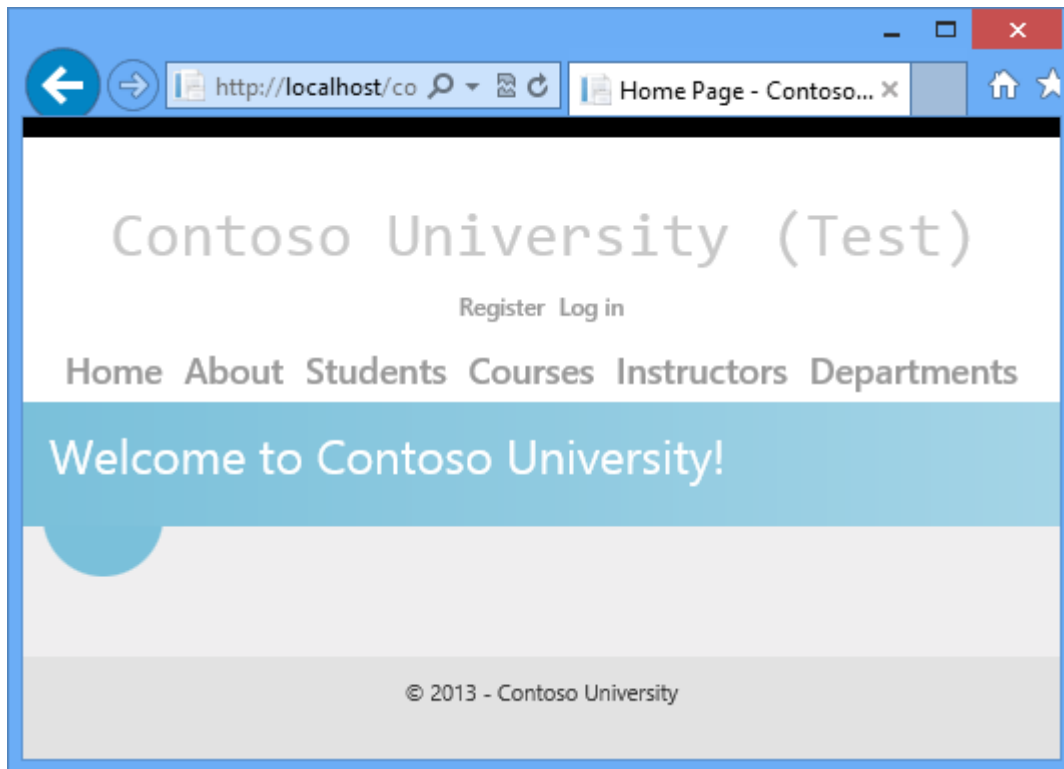
This additional connection string enables you to specify one user account for database schema updates and a different user account for application data access. For example, you could assign the **db\_owner** role to Code First Migrations and **db\_datareader** with **db\_datawriter** roles to the application. This is a common defense-in-depth pattern that prevents potentially malicious code in the application from changing the database schema. (For example, this might happen in a successful SQL injection attack.) These tutorials don't use this pattern. To implement this pattern in your scenario, take these steps:



1. In the **Publish Web** wizard under the **Settings** tab, enter the connection string that specifies a user with full database schema update permissions. Clear the **Use this connection string at runtime** check box. In the deployed Web.config file, this becomes the DatabasePublish connection string.
2. Create a Web.config file transformation for the connection string that you want the application to use at run time.

## Summary

You've now deployed your application to IIS on your development computer and tested it there.



This verifies that the deployment process copied the application's content to the right location (excluding the files that you did not want to deploy) and also that Web Deploy configured IIS correctly during deployment. In the next tutorial, you'll run one more test that finds a deployment task that has not yet been done: setting folder permissions on the *Elmah* folder.

Courtesy: <https://docs.microsoft.com/en-us/aspnet/web-forms/overview/deployment/visual-studio-web-deployment/deploying-to-iis>

Modified: 2021.10.04.7.41.AM

Dököll Solutions, Inc.