Create & deploy a simple Azure Application using Cloud Services

Contents

[Overview 2](#_Toc442197462)

[Contoso Ads application 2](#_Toc442197463)

[What you'll learn 3](#_Toc442197464)

[Prerequisites 4](#_Toc442197465)

[Application architecture 4](#_Toc442197466)

[Set up the development environment 5](#_Toc442197467)

[Download and run the completed solution 5](#_Toc442197468)

[Deploy the application to Azure 11](#_Toc442197469)

[**Create an Azure cloud service** 11](#_Toc442197470)

[**Create an Azure SQL database** 12](#_Toc442197471)

[**Create an Azure storage account** 13](#_Toc442197472)

[**Configure the solution to use your Azure SQL database when it runs in Azure** 15](#_Toc442197473)

[**Configure the solution to use your Azure storage account when it runs in Azure** 18](#_Toc442197474)

[**Deploy the project to Azure** 21](#_Toc442197475)

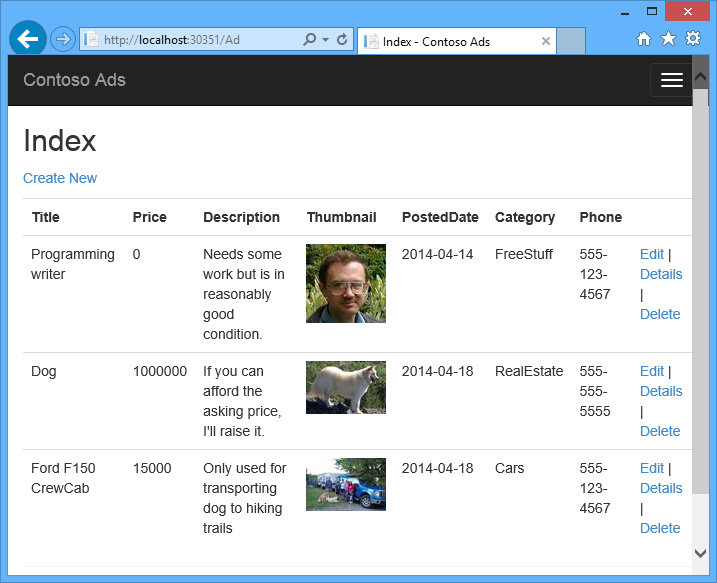
## Overview

This tutorial shows how to create a multi-tier .NET application with an ASP.NET MVC front-end, and deploy it to an [Azure cloud service](https://azure.microsoft.com/en-us/documentation/articles/fundamentals-application-models/#CloudServices). The application uses [Azure SQL Database](http://msdn.microsoft.com/library/azure/ee336279), the [Azure Blob service](http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/unstructured-blob-storage), and the [Azure Queue service](http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/queue-centric-work-pattern). You can [download the Visual Studio project](http://code.msdn.microsoft.com/Simple-Azure-Cloud-Service-e01df2e4) from the MSDN Code Gallery.

The tutorial shows you how to build and run the application locally, how to deploy it to Azure and run in the cloud, and finally how to build it from scratch. You can start by building from scratch and then do the test and deploy steps afterward if you prefer.

## Contoso Ads application

The application is an advertising bulletin board. Users create an ad by entering text and uploading an image. They can see a list of ads with thumbnail images, and they can see the full size image when they select an ad to see the details.



The application uses the [queue-centric work pattern](http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/queue-centric-work-pattern) to off-load the CPU-intensive work of creating thumbnails to a back-end process.

## What you'll learn

* How to enable your machine for Azure development by installing the Azure SDK.
* How to create a Visual Studio cloud service project with an ASP.NET MVC web role and a worker role.
* How to test the cloud service project locally, using the Azure storage emulator.
* How to publish the cloud project to an Azure cloud service and test using an Azure storage account.
* How to upload files and store them in the Azure Blob service.
* How to use the Azure Queue service for communication between tiers.

## Prerequisites

The tutorial assumes that you understand [basic concepts about Azure cloud services](https://azure.microsoft.com/en-us/documentation/articles/fundamentals-application-models/#CloudServices) such asweb role and worker role terminology. It also assumes that you know how to work with [ASP.NET MVC](http://www.asp.net/mvc/tutorials/mvc-5/introduction/getting-started) or [Web Forms](http://www.asp.net/web-forms/tutorials/aspnet-45/getting-started-with-aspnet-45-web-forms/introduction-and-overview) projects in Visual Studio. The sample application uses MVC, but most of the tutorial also applies to Web Forms.

You can run the app locally without an Azure subscription, but you'll need one in order to deploy the application to the cloud. If you don't have an account, you can [activate your MSDN subscriber benefits](https://azure.microsoft.com/pricing/member-offers/msdn-benefits-details/?WT.mc_id=A55E3C668) or [sign up for a free trial](https://azure.microsoft.com/pricing/free-trial/?WT.mc_id=A55E3C668).

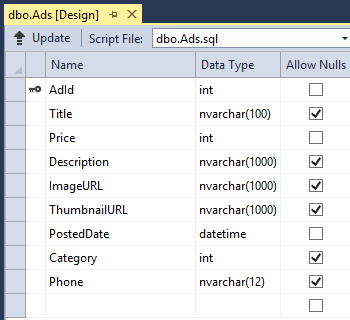
The tutorial instructions work with either of the following products:

* Visual Studio 2013
* Visual Studio 2015

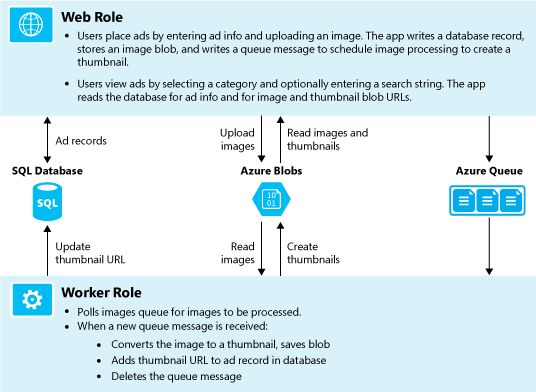
If you don't have one of these, Visual Studio 2015 will be installed automatically when you install the Azure SDK.

## Application architecture

The app stores ads in a SQL database, using Entity Framework Code First to create the tables and access the data. For each ad the database stores two URLs, one for the full-size image and one for the thumbnail.



When a user uploads an image, the front-end running in a web role stores the image in an[Azure blob](http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/unstructured-blob-storage), and it stores the ad information in the database with a URL that points to the blob. At the same time, it writes a message to an Azure queue. A back-end process running in a worker role periodically polls the queue for new messages. When a new message appears, the worker role creates a thumbnail for that image and updates the thumbnail URL database field for that ad. The following diagram shows how the parts of the application interact.



## Set up the development environment

To start, set up your development environment by installing the [Azure SDK for Visual Studio 2015](http://go.microsoft.com/fwlink/?linkid=518003&clcid=0x409) or the [Azure SDK for Visual Studio 2013](http://go.microsoft.com/fwlink/?linkid=324322&clcid=0x409).

If you don't have Visual Studio installed, use the link for Visual Studio 2015, and Visual Studio will be installed along with the SDK.

##### **NOTE:**

Depending on how many of the SDK dependencies you already have on your machine, installing the SDK could take a long time, from several minutes to a half hour or more.

## Download and run the completed solution

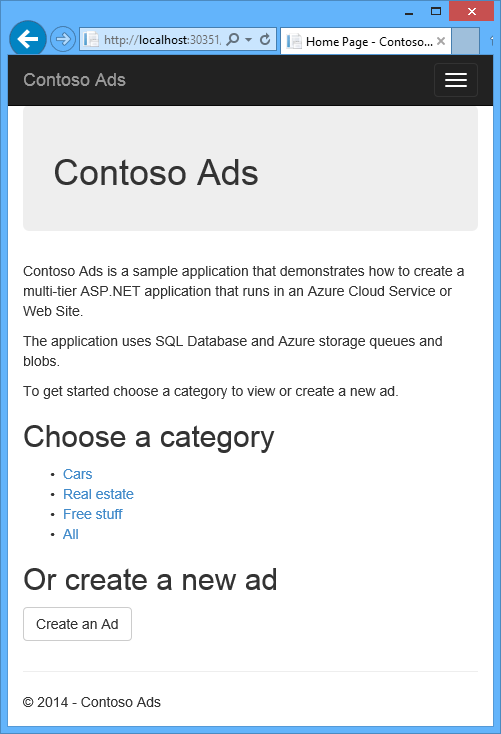
1. Download and unzip the [completed solution](http://code.msdn.microsoft.com/Simple-Azure-Cloud-Service-e01df2e4).
2. Start Visual Studio.
3. From the **File** menu choose **Open Project**, navigate to where you downloaded the solution, and then open the solution file.
4. Press CTRL+SHIFT+B to build the solution.

By default, Visual Studio automatically restores the NuGet package content, which was not included in the .zip file. If the packages don't restore, install them manually by going to the**Manage NuGet Packages for Solution** dialog box and clicking the **Restore** button at the top right.

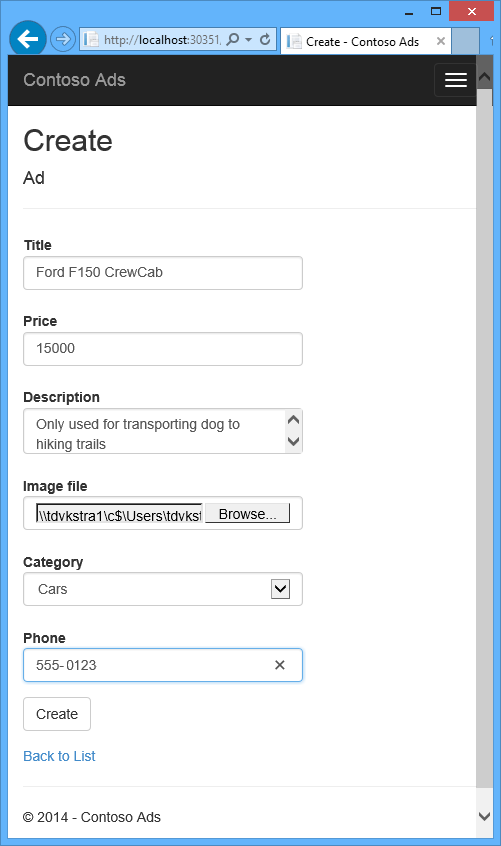
1. In **Solution Explorer**, make sure that **ContosoAdsCloudService** is selected as the startup project.
2. If you're using Visual Studio 2015, change the SQL Server connection string in the application Web.config file of the ContosoAdsWeb project and in theServiceConfiguration.Local.cscfg file of the ContosoAdsCloudService project. In each case, change "(localdb)\v11.0" to "(localdb)\MSSQLLocalDB".
3. Press CTRL+F5 to run the application.

When you run a cloud service project locally, Visual Studio automatically invokes the Azurecompute emulator and Azure storage emulator. The compute emulator uses your computer's resources to simulate the web role and worker role environments. The storage emulator uses a [SQL Server Express LocalDB](http://msdn.microsoft.com/library/hh510202.aspx) database to simulate Azure cloud storage.

The first time you run a cloud service project, it takes a minute or so for the emulators to start up. When emulator startup is finished, the default browser opens to the application home page.

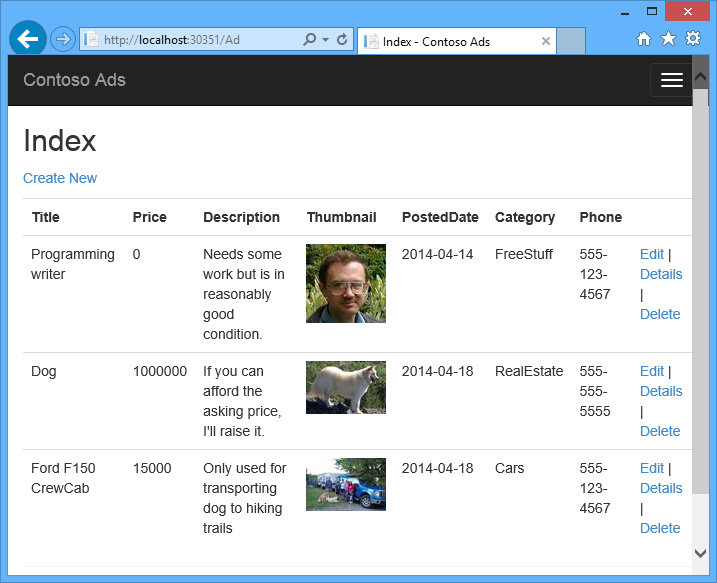


1. Click **Create an Ad**.
2. Enter some test data and select a .jpg image to upload, and then click **Create**.

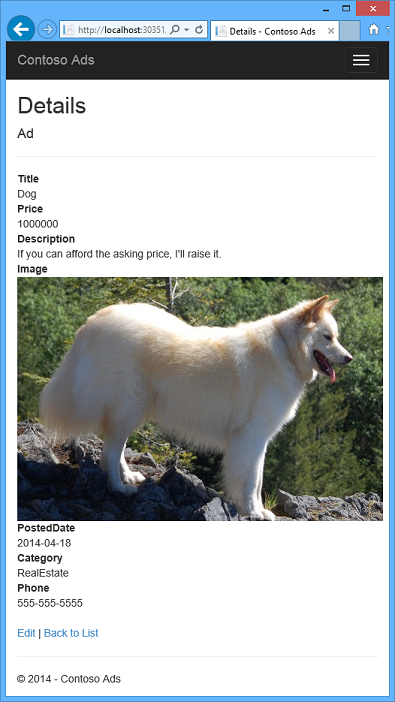


The app goes to the Index page, but it doesn't show a thumbnail for the new ad because that processing hasn't happened yet.

1. Wait a moment and then refresh the Index page to see the thumbnail.



1. Click **Details** for your ad to see the full-size image.



You've been running the application entirely on your local computer, with no connection to the cloud. The storage emulator stores the queue and blob data in a SQL Server Express LocalDB database, and the application stores the ad data in another LocalDB database. Entity Framework Code First automatically created the ad database the first time the web app tried to access it.

In the following section you'll configure the solution to use Azure cloud resources for queues, blobs, and the application database when it runs in the cloud. If you wanted to continue to run locally but use cloud storage and database resources, you could do that; it's just a matter of setting connection strings, which you'll see how to do.

## Deploy the application to Azure

You'll do the following steps to run the application in the cloud:

* Create an Azure cloud service.
* Create an Azure SQL database.
* Create an Azure storage account.
* Configure the solution to use your Azure SQL database when it runs in Azure.
* Configure the solution to use your Azure storage account when it runs in Azure.
* Deploy the project to your Azure cloud service.

### **Create an Azure cloud service**

An Azure cloud service is the environment the application will run in.

1. In your browser, open the [Azure classic portal](http://manage.windowsazure.com/).
2. Click **New > Compute > Cloud Service > Quick Create**.
3. In the URL input box, enter a URL prefix.

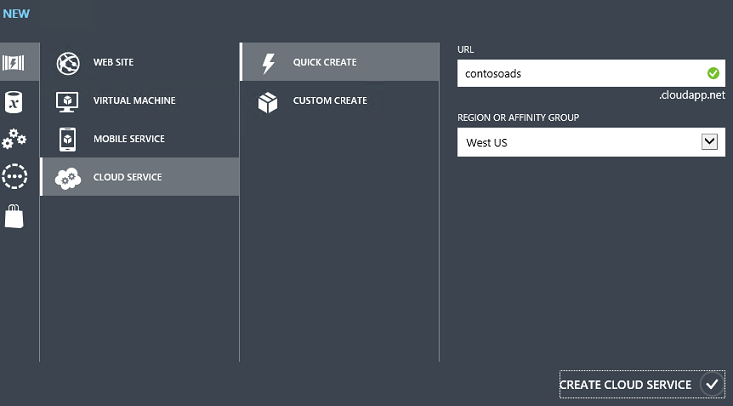
This URL has to be unique. You'll get an error message if the prefix you choose is already in use by someone else.

1. Choose the region where you want to deploy the application.

This field specifies which datacenter your cloud service will be hosted in. For a production application, you'd choose the region closest to your customers. For this tutorial, choose the region closest to you.

1. Click **Create Cloud Service**.

In the following image, a cloud service is created with the URL contosoads.cloudapp.net.



### **Create an Azure SQL database**

When the app runs in the cloud, it will use a cloud-based database.

1. In the [Azure classic portal](http://manage.windowsazure.com/), click **New > Data Services > SQL Database > Quick Create**.
2. In the **Database Name** box, enter contosoads.
3. From the **Server** drop-down list, choose **New SQL Database server**.

Alternatively, if your subscription already has a server, you can select that server from the drop-down list.

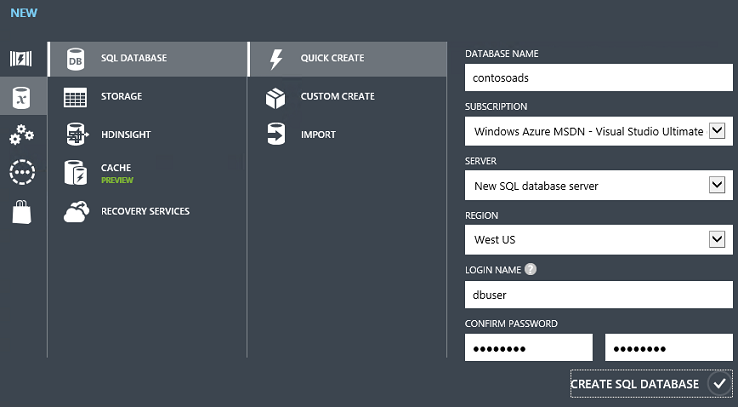
1. Choose the same **Region** that you chose for the cloud service.

When the cloud service and database are in different datacenters (different regions), latency will increase and you will be charged for bandwidth outside the data center. Bandwidth within a data center is free.

1. Enter an administrator **Login Name** and **Password**.

If you selected **New SQL Database server** you aren't entering an existing name and password here, you're entering a new name and password that you're defining now to use later when you access the database. If you selected a server that you created previously, you'll be prompted for the password to the administrative user account you already created.

1. Click **Create SQL Database**.



1. After Azure finishes creating the database, click the **SQL Databases** tab in the left pane of the portal, and then click the name of the new database.
2. Click the **Dashboard** tab.
3. Click **Manage allowed IP addresses**.
4. Under **Allowed Services**, change **Azure Services** to **Yes**.
5. Click **Save**.

### **Create an Azure storage account**

An Azure storage account provides resources for storing queue and blob data in the cloud.

In a real-world application, you would typically create separate accounts for application data versus logging data, and separate accounts for test data versus production data. For this tutorial you'll use just one account.

1. In the [Azure classic portal](http://manage.windowsazure.com/), click **New > Data Services > Storage > Quick Create**.
2. In the **URL** box, enter a URL prefix.

This prefix plus the text you see under the box will be the unique URL to your storage account. If the prefix you enter has already been used by someone else, you'll have to choose a different prefix.

1. Set the **Region** drop-down list to the same region you chose for the cloud service.

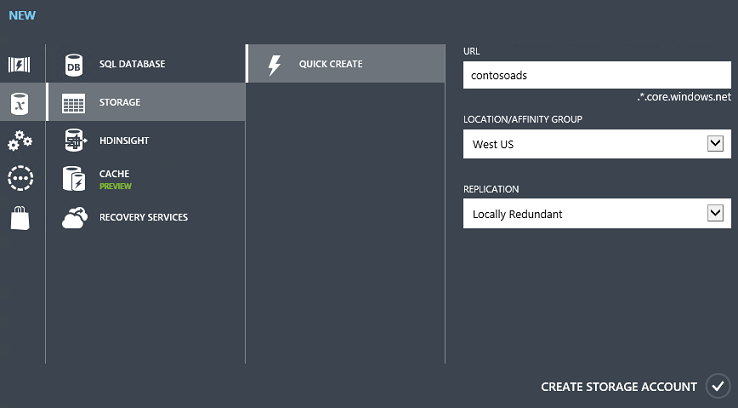
When the cloud service and storage account are in different datacenters (different regions), latency will increase and you will be charged for bandwidth outside the data center. Bandwidth within a data center is free.

Azure affinity groups provide a mechanism to minimize the distance between resources in a data center, which can reduce latency. This tutorial does not use affinity groups. For more information, see [How to Create an Affinity Group in Azure](http://msdn.microsoft.com/library/jj156209.aspx).

1. Set the **Replication** drop-down list to **Locally redundant**.

When geo-replication is enabled for a storage account, the stored content is replicated to a secondary datacenter to enable failover to that location in case of a major disaster in the primary location. Geo-replication can incur additional costs. For test and development accounts, you generally don't want to pay for geo-replication. For more information, see[Create, manage, or delete a storage account](https://azure.microsoft.com/en-us/documentation/articles/storage-create-storage-account/#replication-options).

1. Click **Create Storage Account**.



In the image, a storage account is created with the URL contosoads.core.windows.net.

### **Configure the solution to use your Azure SQL database when it runs in Azure**

The web project and the worker role project each has its own database connection string, and each needs to point to the Azure SQL database when the app runs in Azure.

You'll use a [Web.config transform](http://www.asp.net/mvc/tutorials/deployment/visual-studio-web-deployment/web-config-transformations) for the web role and a cloud service environment setting for the worker role.

##### **NOTE:**

In this section and the next section you store credentials in project files. [Don't store sensitive data in public source code repositories](http://www.asp.net/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/source-control#secrets).

1. In the ContosoAdsWeb project, open the Web.Release.config transform file for the application Web.config file, delete the comment block that contains a <connectionStrings>element, and paste the following code in its place.

Copy

<connectionStrings>

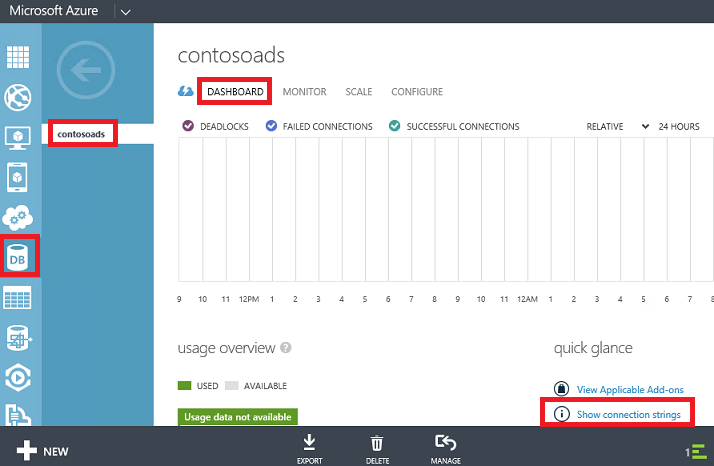
<add name="ContosoAdsContext" connectionString="{connectionstring}"

providerName="System.Data.SqlClient" xdt:Transform="SetAttributes" xdt:Locator="Match(name)"/>

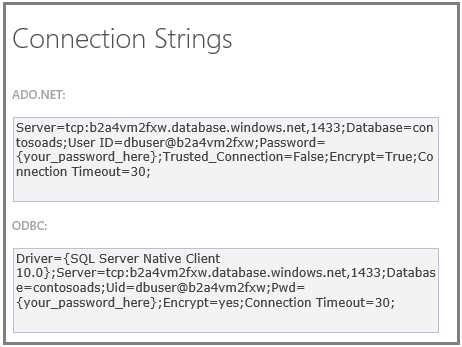
</connectionStrings>

Leave the file open for editing.

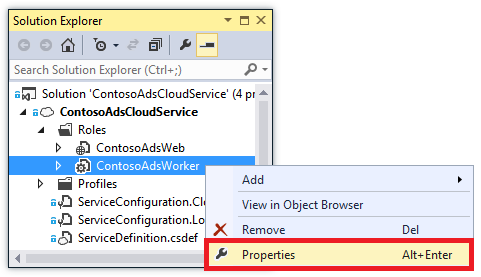
1. In the [Azure classic portal](http://manage.windowsazure.com/), click **SQL Databases** in the left pane, click the database you created for this tutorial, click the **Dashboard** tab, and then click **Show connection strings**.



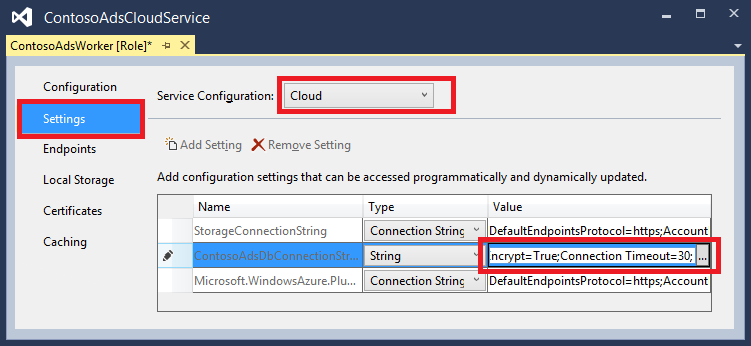
The portal displays connection strings, with a placeholder for the password.



1. In the Web.Release.config transform file, delete {connectionstring} and paste in its place the ADO.NET connection string from the Azure classic portal.
2. In the connection string that you pasted into the Web.Release.config transform file, replace{your\_password\_here} with the password you created for the new SQL database.
3. Save the file.
4. Select and copy the connection string (without the surrounding quotation marks) for use in the following steps for configuring the worker role project.
5. In **Solution Explorer**, under **Roles** in the cloud service project, right-click**ContosoAdsWorker** and then click **Properties**.



1. Click the **Settings** tab.
2. Change **Service Configuration** to **Cloud**.
3. Select the **Value** field for the ContosoAdsDbConnectionString setting, and then paste the connection string that you copied from the previous section of the tutorial.

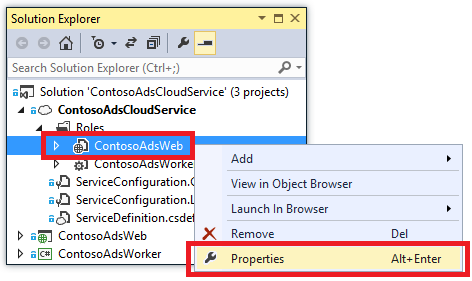


1. Save your changes.

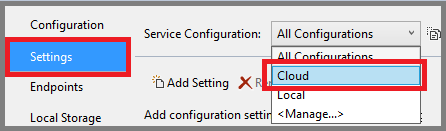
### **Configure the solution to use your Azure storage account when it runs in Azure**

Azure storage account connection strings for both the web role project and the worker role project are stored in environment settings in the cloud service project. For each project there is a separate set of settings to be used when the application runs locally and when it runs in the cloud. You'll update the cloud environment settings for both web and worker role projects.

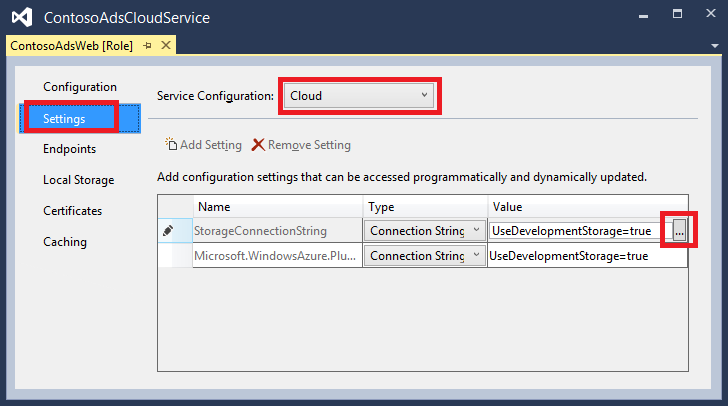
1. In **Solution Explorer**, right-click **ContosoAdsWeb** under **Roles** in the**ContosoAdsCloudService** project, and then click **Properties**.



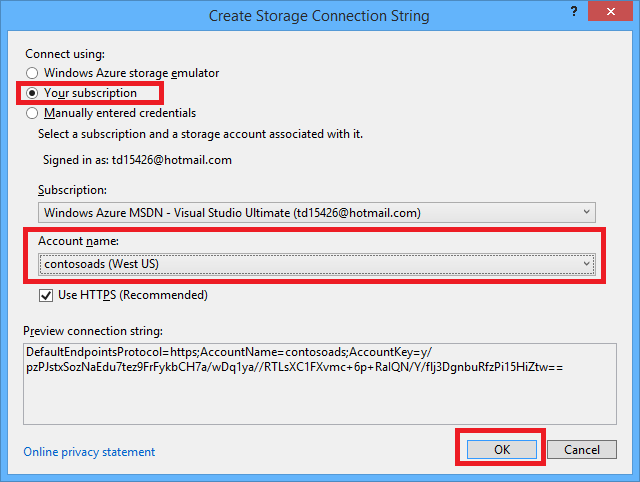
1. Click the **Settings** tab. In the **Service Configuration** drop-down box, choose **Cloud**.



1. Select the **StorageConnectionString** entry, and you'll see an ellipsis (**...**) button at the right end of the line. Click the ellipsis button to open the **Create Storage Account Connection String** dialog box.



1. In the **Create Storage Connection String** dialog box, click **Your subscription**, choose the storage account that you created earlier, and then click **OK**. If you're not already logged in, you'll be prompted for your Azure account credentials.



1. Save your changes.
2. Follow the same procedure that you used for the StorageConnectionString connection string to set the Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionStringconnection string.

This connection string is used for logging.

1. Follow the same procedure that you used for the **ContosoAdsWeb** role to set both connection strings for the **ContosoAdsWorker** role. Don't forget to set **Service Configuration** to **Cloud**.

The role environment settings that you have configured using the Visual Studio UI are stored in the following files in the ContosoAdsCloudService project:

* ServiceDefinition.csdef - Defines the setting names.
* ServiceConfiguration.Cloud.cscfg - Provides values for when the app runs in the cloud.
* ServiceConfiguration.Local.cscfg - Provides values for when the app runs locally.

For example, the ServiceDefinition.csdef includes the following definitions.

Copy

<ConfigurationSettings>

<Setting name="StorageConnectionString" />

<Setting name="ContosoAdsDbConnectionString" />

</ConfigurationSettings>

And the ServiceConfiguration.Cloud.cscfg file includes the values you entered for those settings in Visual Studio.

Copy

<Role name="ContosoAdsWorker">

<Instances count="1" />

<ConfigurationSettings>

<Setting name="StorageConnectionString" value="{yourconnectionstring}" />

<Setting name="ContosoAdsDbConnectionString" value="{yourconnectionstring}" />

<!-- other settings not shown -->

</ConfigurationSettings>

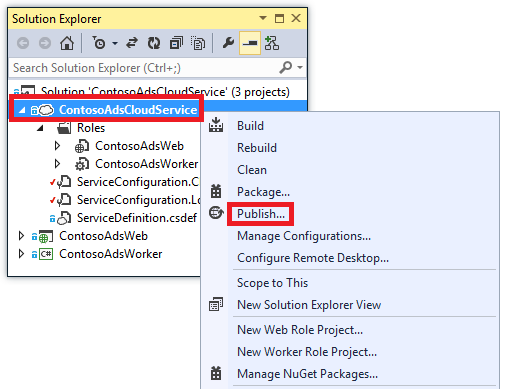
<!-- other settings not shown -->

</Role>

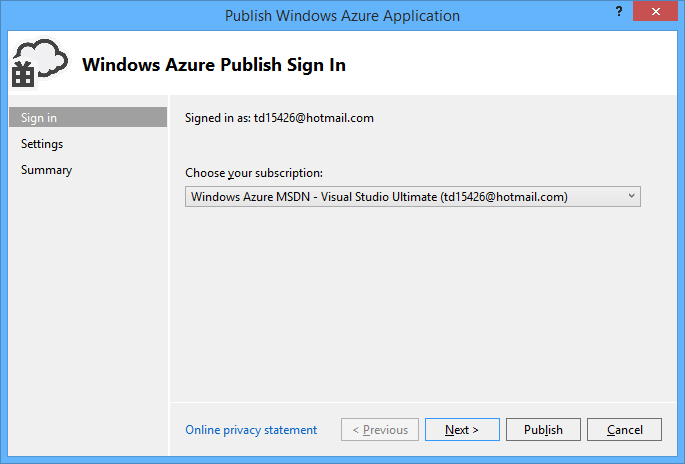
The <Instances> setting specifies the number of virtual machines that Azure will run the worker role code on. The [Next steps](https://azure.microsoft.com/en-us/documentation/articles/cloud-services-dotnet-get-started/#next-steps) section includes links to more information about scaling out a cloud service,

### **Deploy the project to Azure**

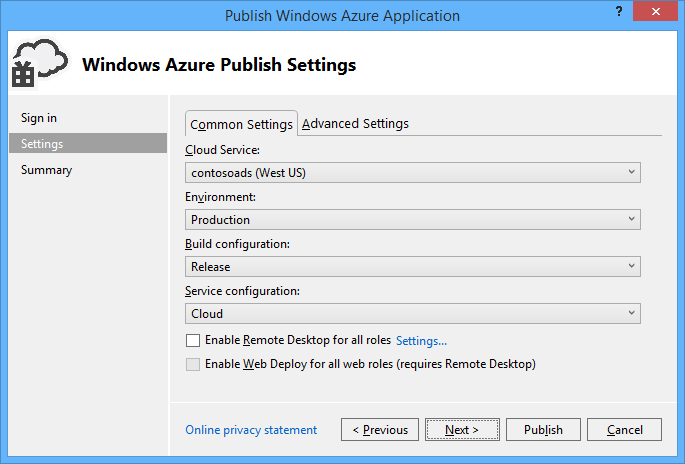
1. In **Solution Explorer**, right-click the **ContosoAdsCloudService** cloud project and then select**Publish**.



1. In the **Sign in** step of the **Publish Azure Application** wizard, click **Next**.

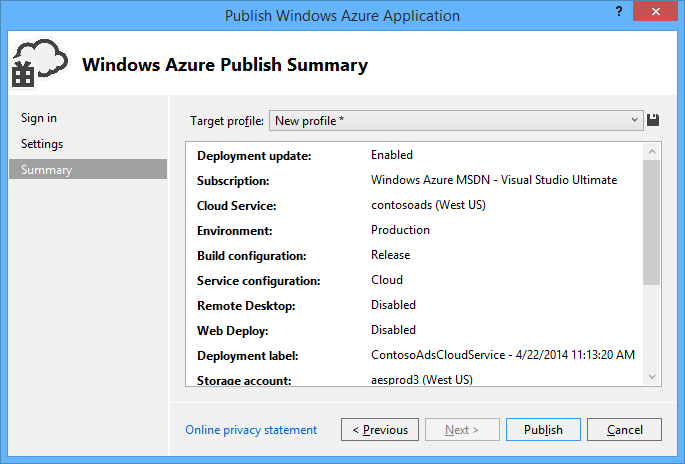


1. In the **Settings** step of the wizard, click **Next**.



The default settings in the **Advanced** tab are fine for this tutorial. For information about the advanced tab, see [Publish Azure Application Wizard](http://msdn.microsoft.com/library/hh535756.aspx).

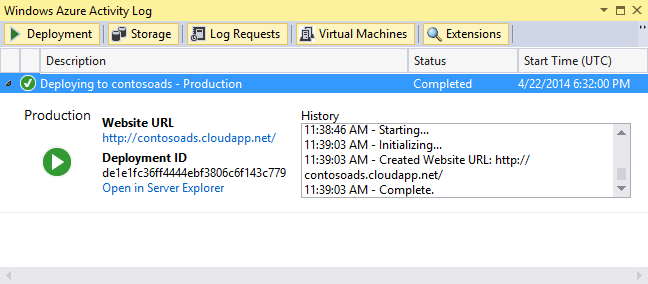
1. In the **Summary** step, click **Publish**.



The **Azure Activity Log** window opens in Visual Studio.

1. Click the right arrow icon to expand the deployment details.

The deployment can take up to 5 minutes or more to complete.



1. When the deployment status is complete, click the **Web app URL** to start the application.
2. You can now test the app by creating, viewing, and editing some ads, as you did when you ran the application locally.

##### **NOTE:**

When you're finished testing, delete or stop the cloud service. Even if you're not using the cloud service, it's accruing charges because virtual machine resources are reserved for it. And if you leave it running, anyone who finds your URL can create and view ads. In the [Azure classic portal](http://manage.windowsazure.com/), go to the **Dashboard** tab for your cloud service, and then click the **Delete** button at the bottom of the page. If you just want to temporarily prevent others from accessing the site, click **Stop** instead. In that case, charges will continue to accrue. You can follow a similar procedure to delete the SQL database and storage account when you no longer need them.

End of the document.