



Move to the Cloud

Overview

Modern line-of-business applications have much to gain from Windows Azure. In addition to the increased scalability and reliability the cloud has to offer, cloud-based services are also easier to access from outside the corporate firewall. This new accessibility opens a whole class of opportunities for developers to enhance their functionality and extend their user experiences to new devices and tablets.

In our scenario, we have a variety of resources and services that exist behind a firewall. Now that we want to extend their reach and functionality, the time has come to move the experience into the cloud. The easiest way for us to migrate the project to the cloud is piece-by-piece. We'll start off with our functional system, and then move just the database. We'll be able to confirm success by wiring the existing on-premises WCF service to the new in-cloud database, and everything else should continue to work as expected. After that, we'll move the service itself out to the cloud. By changing the WCF service URL configured in the WPF application, it should also immediately work.

Important Note

Many of the screenshots in this lab use configuration fields from Windows Azure that are globally unique. As a result, your settings will likely be different. Please take care to use the fields for your configuration. For example, the domain **"expenseswcf.azurewebsites.net"** is in the screenshots, but you should use the domain names you generate during your session. Other places where screenshots will vary include the SQL Server name and administrator name, Windows Azure publish settings, and various GUIDs. Effort has been made to call these out in the steps, but please double-check copies & pastes.

Objectives

In this hands-on lab, you will learn how to:

- Migrate an on-premises database to SQL Azure
- Migrate an on-premises WCF service to Windows Azure Web Sites

Prerequisites

The following is required to complete this hands-on lab:

- Microsoft Visual Studio 2013
- The latest Windows Azure SDK for .NET

- A Windows Azure subscription

Setup

This lab picks up where module 2 left off.

Exercises

This hands-on lab includes the following exercises:

1. Deploying to SQL Azure
2. Deploying to Windows Azure Web Sites

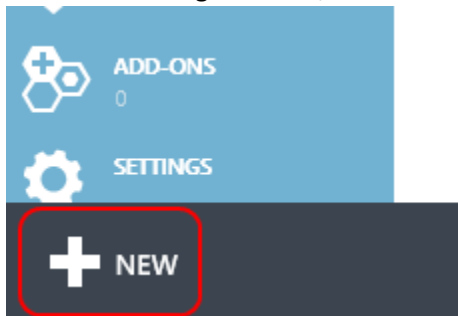
Exercise 1: Deploying to SQL Azure

In this exercise, we'll go through the process of setting up a SQL Azure database server. Then we'll deploy our existing database, including its data, out to that server.

Task 1: Creating a SQL Azure database

In this task, we'll create a SQL Azure database.

1. Log into your Azure administrative account at <https://manage.windowsazure.com>.
2. In the bottom right corner, click the **New** button.



3. Click **DATA SERVICES | SQL DATABASE | QUICK CREATE** and fill out the form to create your database. If you already have an existing SQL server configured in Azure you'll see the option to reuse it. For our exercise, please select the option to create a new one. Give the database a name, and choose an appropriate Region and administrative username and password for the new server. Click the **"CREATE SQL DATABASE"** button to create the database and new server.

NEW

SQL DATABASE QUICK CREATE

STORAGE CUSTOM CREATE

HDINSIGHT IMPORT

CACHE PREVIEW

RECOVERY SERVICES

SQL REPORTING

DATABASE NAME
Expenses

SERVER
New SQL database server

REGION
West US

LOGIN NAME
SQLAdminAccount

CONFIRM PASSWORD
.....

CREATE SQL DATABASE

- It will take a few moments for your database to get created.
- Once the empty database has been created, you'll see a message like this one.



- Click the **SQL Databases** tab along the left edge of the page and click on the newly created **expenses** database. Note that you'll need to click on the name specifically to navigate to its page.

Database Name	Status	Region	Server Name	SKU	Storage
praffle_db	Online	West US	Azdem169D875...	ytw6j1j8m	1 GB
Expenses	Online	West US	Azdem169D875...	dsqdg8tg54	1 GB






- Click the **Dashboard** link to see more options for the database.



- In the **Quick Glance** section along the right edge you can get some useful information about your server. Take note of the **Server Name**, which we'll need later on for deploying the database. By default, only IP addresses inside the Azure datacenter have access to the database server. Next, click

the **Manage allowed IP addresses** link to allow your current connection to access the database as well.

quick glance

-  [Show connection strings](#)
-  [Learn more about troubleshooting connections](#)
-  [Business Continuity in Windows Azure SQL Database](#)
-  [Learn more about backup and restore](#)
-  [Manage allowed IP addresses](#)

SERVER NAME
awrdjymzx4.database.windows.net

9. If the public IP you connect from is a static IP address, you can add it by clicking the **Add to the allowed IP addresses** button. If you're accessing via network with a range of public IPs, you can add a rule for that range. Note that you also have the option to disallow access to your server from services running inside Azure, which may be useful if all access comes from external sources, only. In our case, we will be ultimately connecting to this database via services running in Azure, so we'll leave the option enabled.

allowed ip addresses

CURRENT CLIENT IP ADDRESS

50.47.43.20

ADD TO THE ALLOWED IP ADDRESSES.



RULE NAME

START IP ADDRESS

END IP ADDRESS

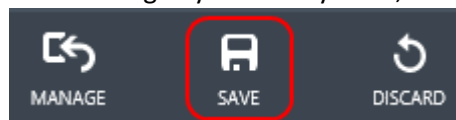
allowed services

WINDOWS AZURE SERVICES

YES


NO

10. After adding any necessary rules, click the **Save** button at the bottom of the page to commit them.



11. Click the **Databases** link to view the databases on this server.

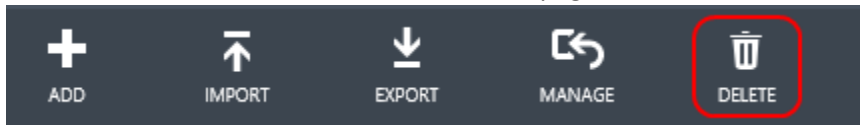
awrdjymzx4

 DASHBOARD **DATABASES** CONFIGURE HISTORY

- Click the row of the **expenses** database, but not on the name itself. We want to select the row, not navigate to the database's record.

NAME	STATUS	LOCATION	SUBSCRIPTION	S
expenses →	✓ Online	Southeast Asia	SharpLogic Azure	a

- Click the **Delete** button at the bottom of the page to delete this database.

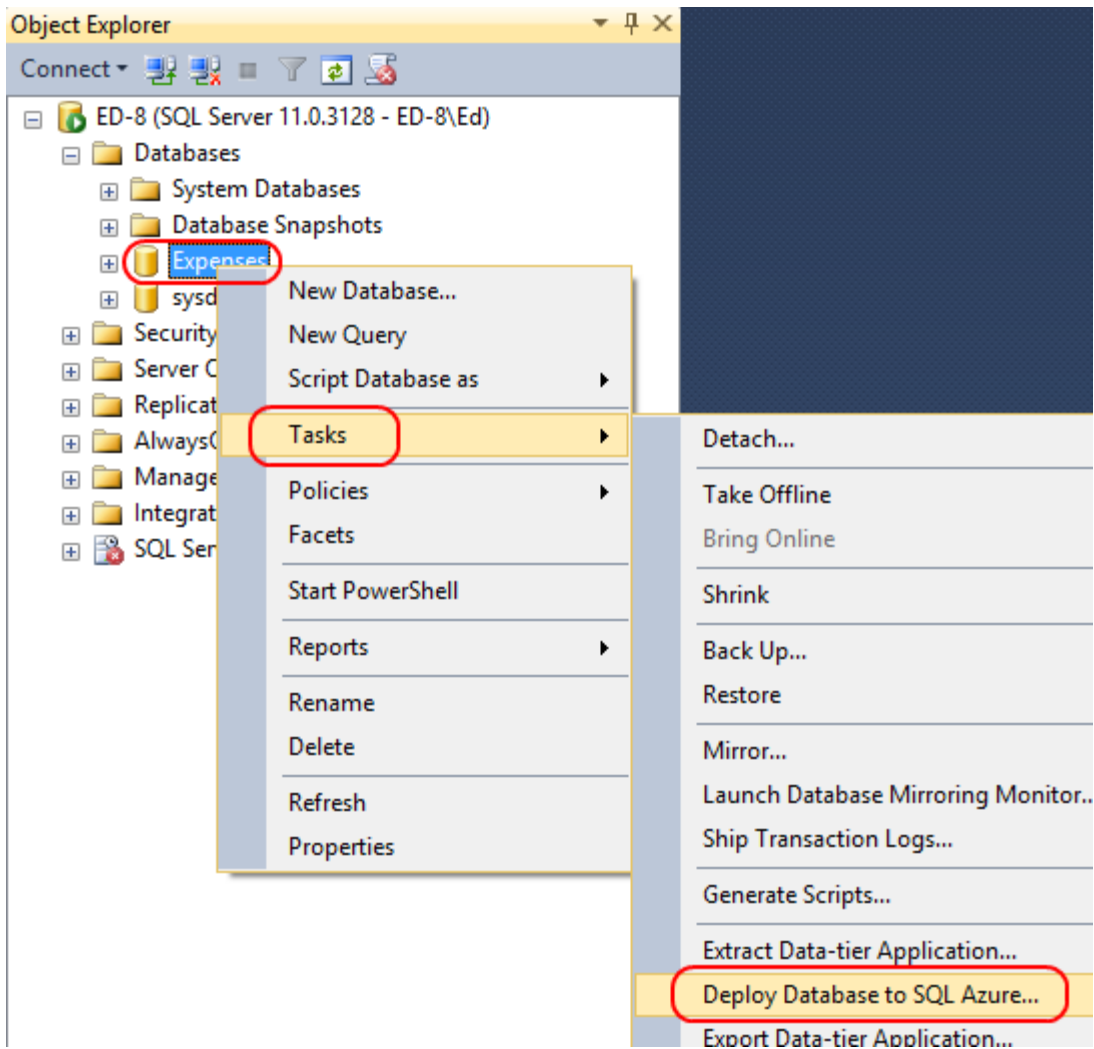


- The management tool will ask you to confirm that you want to delete the database. Accept the confirmation. However, it will then ask you if you would like to delete the server because it no longer has any databases. Deny this request since we'll be deploying a database in the next step.

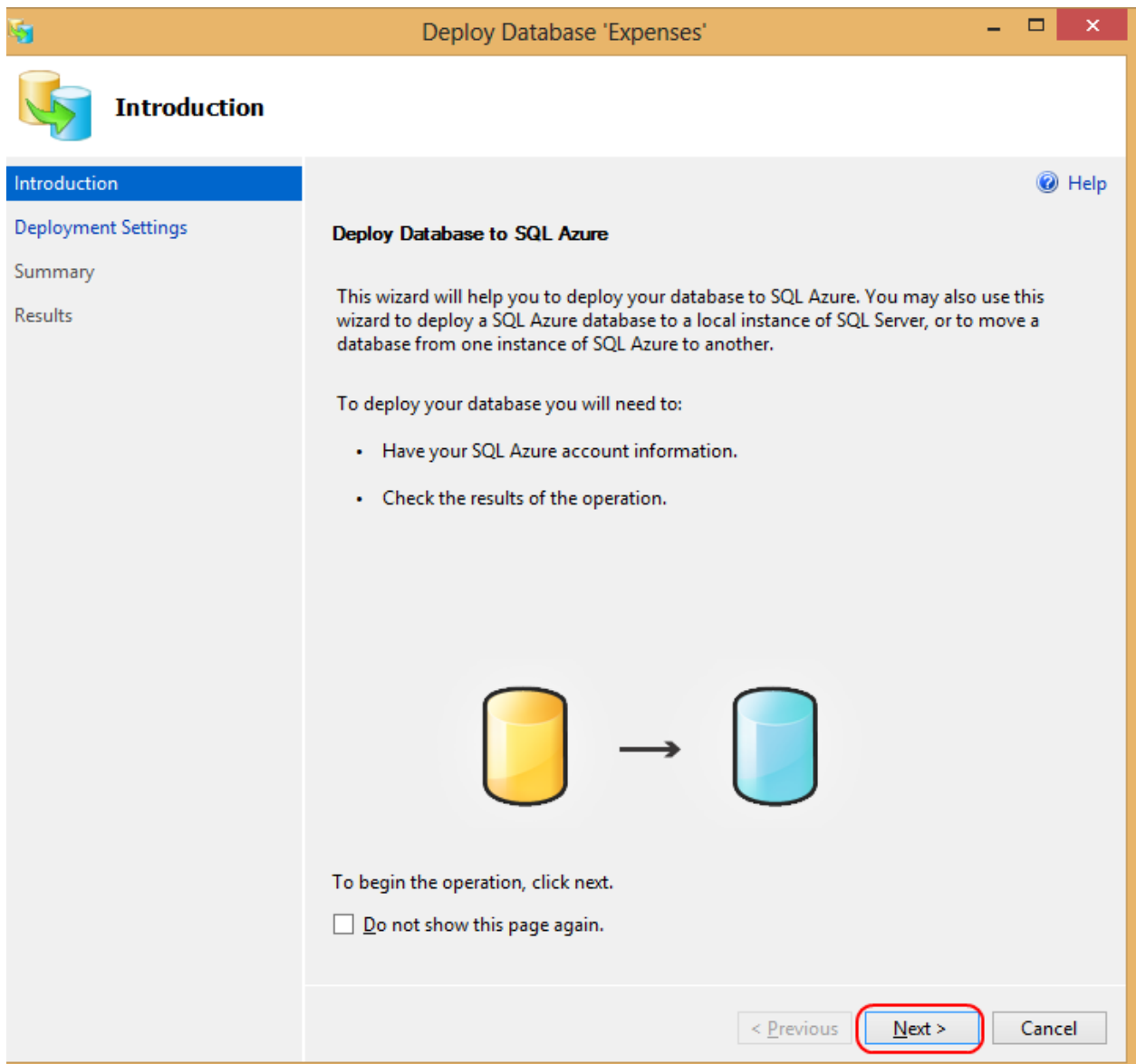
Task 2: [Deploy your existing database to SQL Azure](#)

In this task, we'll deploy your existing database to SQL Azure.

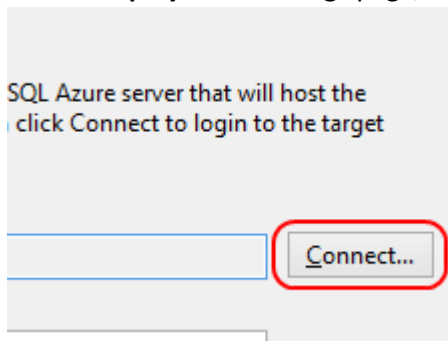
- Open **SQL Server Management Studio**.
- Connect to the server containing your local **Expenses** database.
- In the **Object Explorer**, right-click the **Expenses** database and select **Tasks | Deploy Database to SQL Azure....**



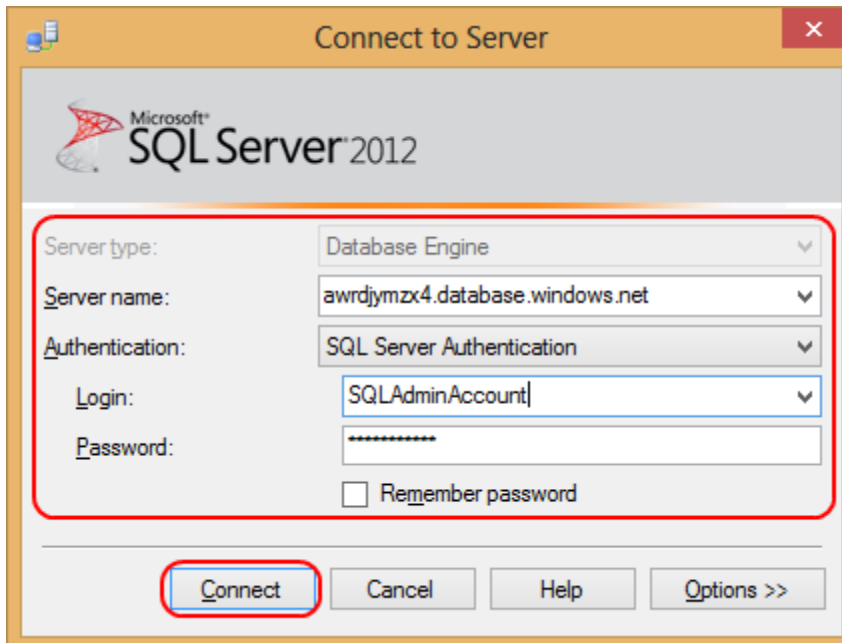
4. If presented with the **Introduction** step of the wizard, click **Next** to continue.



5. On the **Deployment Settings** page, click **Connect** to connect to the target server.

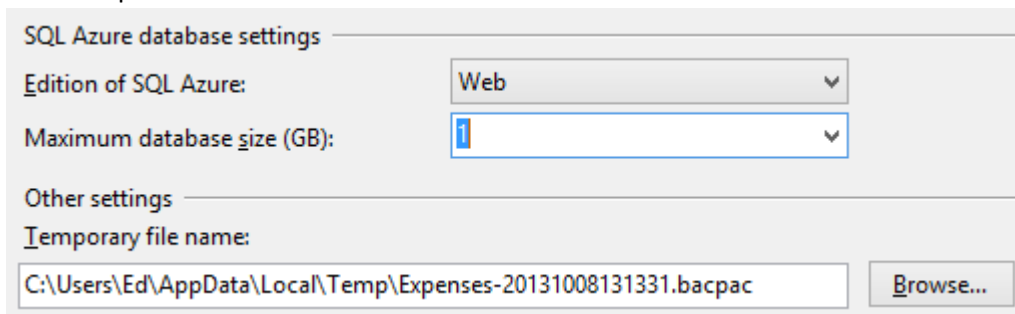


6. Fill out the connection details for **your SQL Azure server** using the settings configured during creation earlier and click **Connect**.

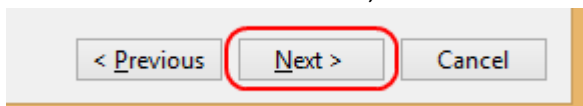


Note that there are some options available for configuration. In the **SQL azure database settings**, we can decide if we want to use the **Web** or **Business** editions of SQL Azure. At this time, the only difference between the two is the maximum database sizes they allow. Web is limited to up to 5GB, whereas Business is intended for databases up to 150GB. For the purposes of our lab, we can use the smallest option available.

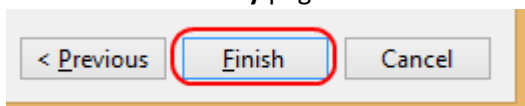
In the **Other settings**, you can decide where the temporary bacpac file is stored. If you have a very large database with limited space on your default drive, this gives you the option to select a different path.



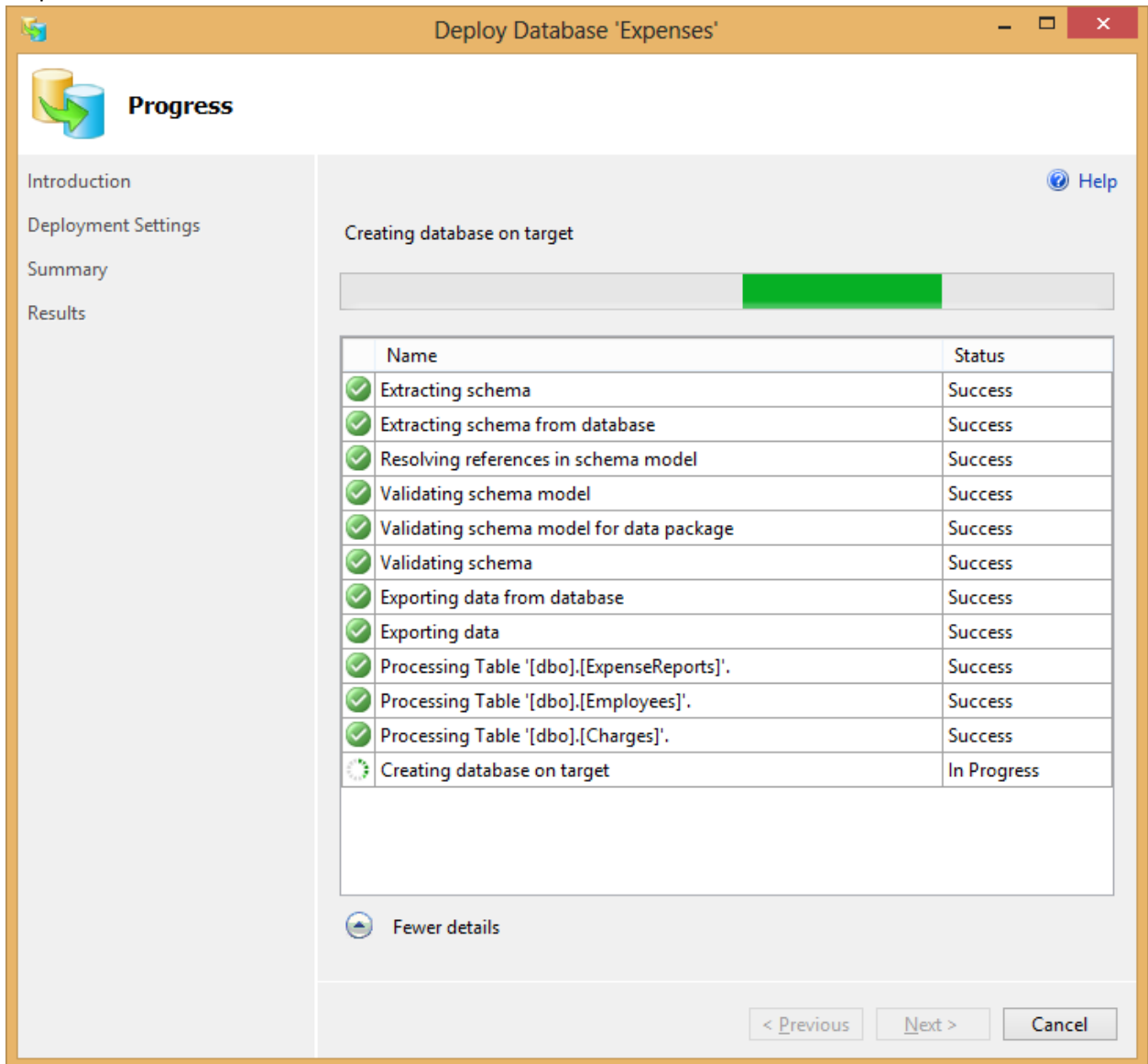
7. Once the connection succeeds, click **Next** to continue.



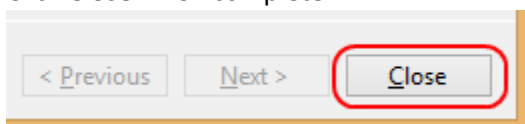
8. Review the **Summary** page. Click **Finish** to begin the deployment.



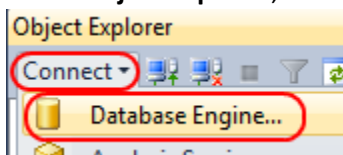
9. It'll take a minute or two for the database to fully deploy. During the operation you can see the steps it takes.



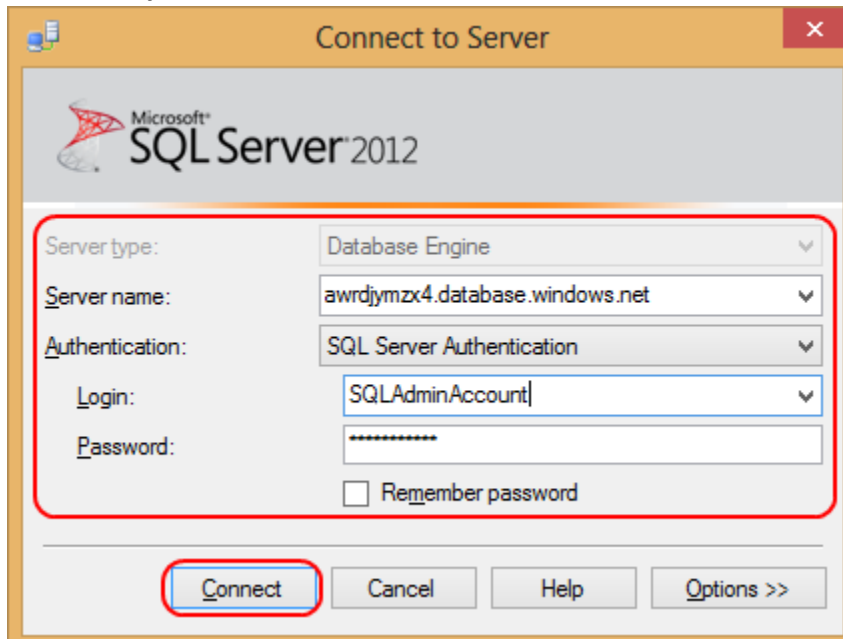
10. Click **Close** when complete.



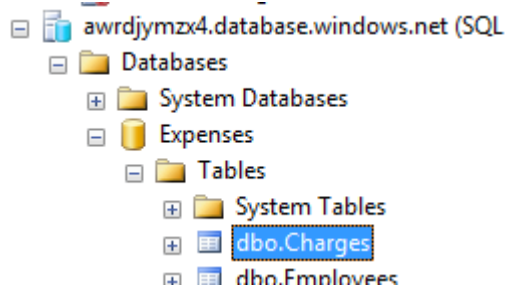
11. In the **Object Explorer**, click **Connect | Database Engine....**



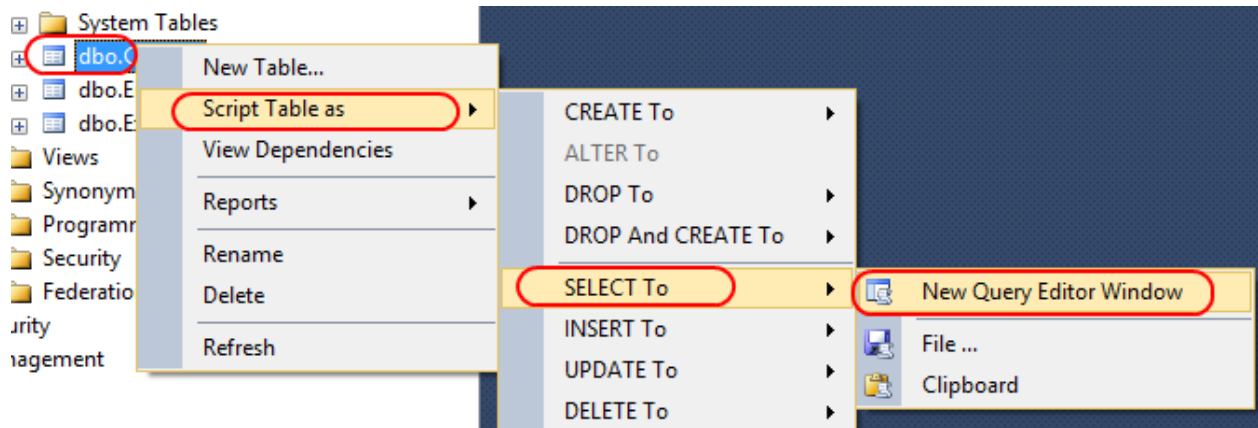
12. Connect to **your SQL Azure server**.



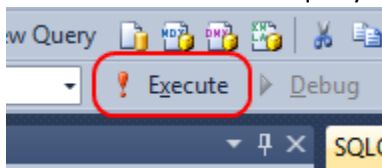
13. Expand **[your server] | Databases | Expenses | Tables**.



14. Right-click the **Charges** table. Note that the options when working with SQL Azure are somewhat limited versus those available for a local SQL instance. SQL Azure still supports almost everything local SQL does, except that some of the features are not exposed via the tool and require scripting. For example, there is no option to edit records inline. Select **Script Table as | SELECT To | New Query Editor Window**.



15. Click **Execute** to run the query.



16. We can now see that all of our data is available in our SQL Azure database.

	ChargeId	EmployeeId	ExpenseReportId	ExpenseDate	Merchant
1	295	16	NULL	2013-08-24	Northwind Inn
2	296	16	NULL	2013-09-18	Contoso Taxi
3	297	16	NULL	2013-09-30	Fourth Coffee
4	298	16	NULL	2013-10-04	Fourth Coffee
5	299	16	206	2013-08-09	Blue Yonder

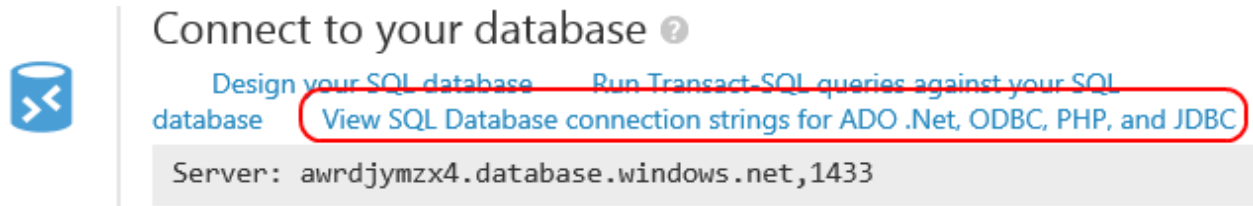
Task 3: Updating the Expenses WCF service to use the new database

In this task, we'll update our WCF service to use the newly created SQL Azure database.

1. Log into your Azure administrative account at <https://manage.windowsazure.com> if it's not already open.
2. Navigate to the newly deployed database by clicking the **SQL Databases** tab, followed by the **Expenses** database link.



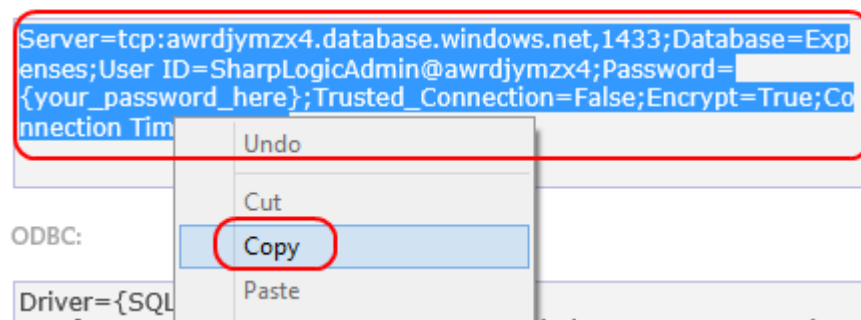
3. In the **Connect to your database** section, click the **View SQL Database connection strings for ADO.NET, ODBC, PHP, and JDBC** link.



4. In the **Connection Strings** dialog, highlight and copy the connection string from the **ADO.NET** box. Press **Esc** to close the dialog when done.

Connection Strings

ADO.NET:



5. In Visual Studio, open the **"Expenses WCF Service"** solution we were working with previously. If it is still running from the previous lab, stop the debug session (*if needed you can open a working copy from "\Source\Begin\Expenses WCF Service\Expenses WCF Service.sln"*).
6. Open the **Web.config** file.
7. In the **ConnectionStrings** section, change **Expenses.WcfService.ServiceCore.Properties.Settings.ExpensesConnectionString** to use the new connection string for your SQL Azure database. **Be sure to replace the "{your_password_here}" token in the connection string with the password you specified when creating the SQL Server previously.**
8. Save **Web.config**.
9. Debug the **"Expenses WCF Service"** to run it again with the updated connection string. Leave the browser with the service page open to keep the service running.
10. In a SEPARATE COPY OF VISUAL STUDIO, Open the **"Expenses"** solution with the WPF client in it. Again, if needed a working starter can be opened from the **"\Source\Begin\Expenses WPF"** folder.

Run the **Expenses** client application to confirm that it works as expected, but is now using data from SQL Azure.

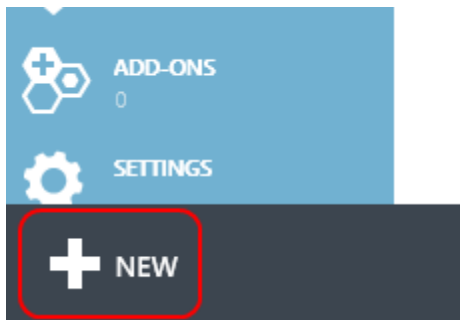
Exercise 2: Deploying to Windows Azure Web Sites

In this exercise, we'll go through the process of setting up a Windows Azure Web Site. Then we'll deploy our existing WCF service out to it.

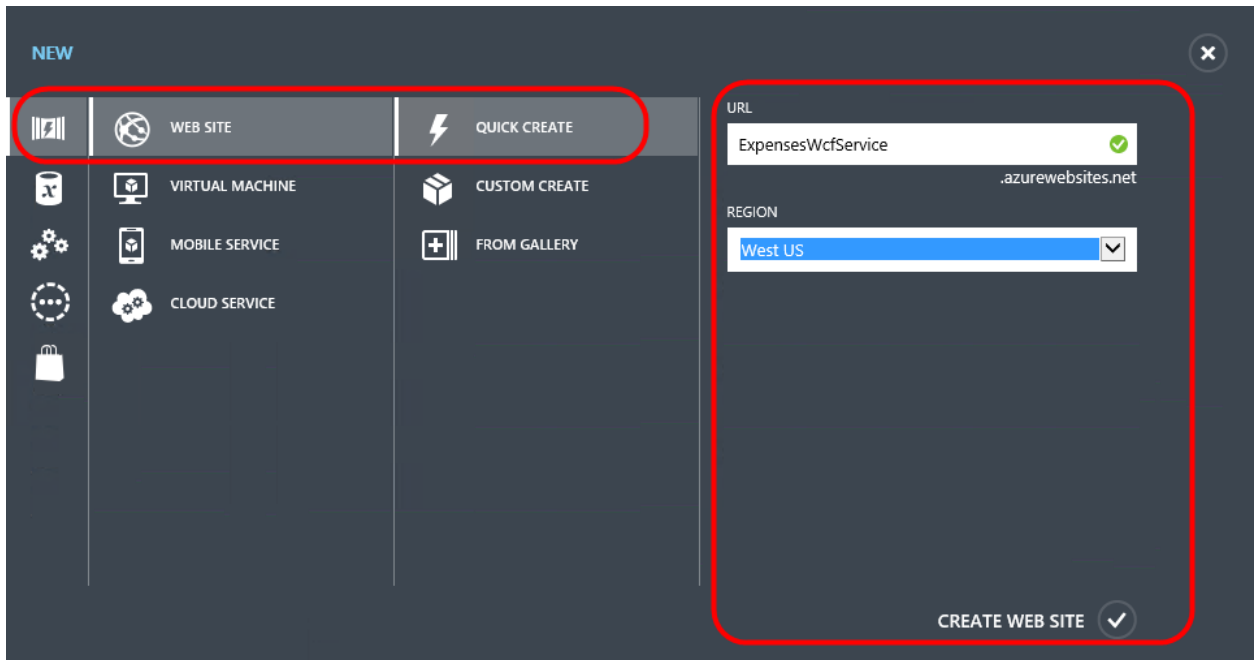
Task 1: Creating Windows Azure Web Site

In this task, we'll create a Windows Azure Web Site.

1. Log into your Azure administrative account at <https://manage.windowsazure.com> if not already open.
2. Click the **New** button in the bottom left corner.

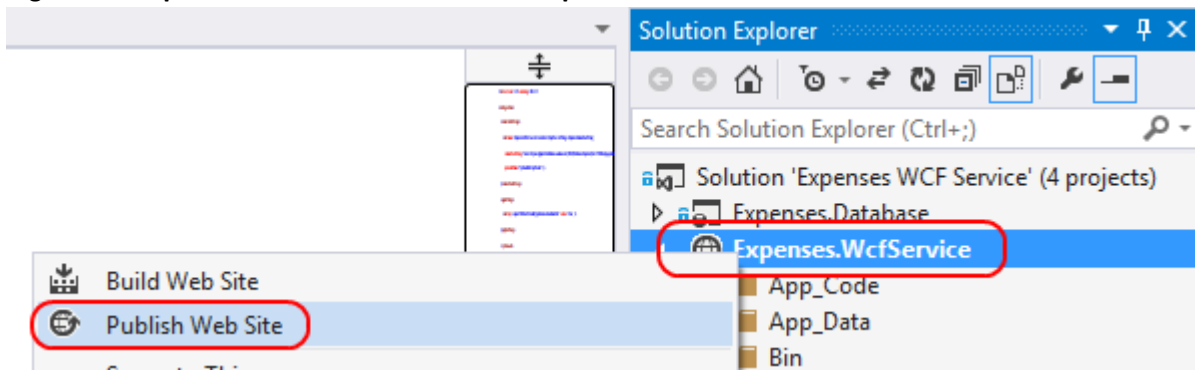


3. Select **Compute | Web Site | Quick Create** and fill out the settings. You'll need to come up with a unique name for the **URL**. Your eventual domain name will be **[your URL name].azurewebsites.net**. You should also place this web site in the same region (data center) as the SQL server you created previously. This will keep the latency between the service and the database down, as well as prevent charges for data moving between data centers. You can ignore the "SUBSCRIPTION" option in the screenshot below. Click **Create Web Site** to create the Web site.

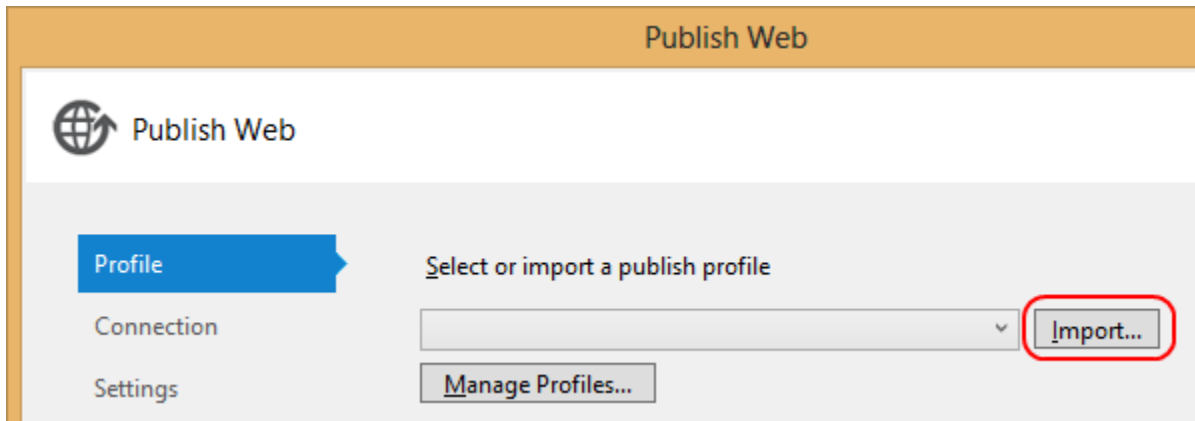


Task 2: Deploy the Expenses WCF service to our new Windows Azure Web Site
 In this task, we'll update our WCF service to use the newly created SQL Azure database.

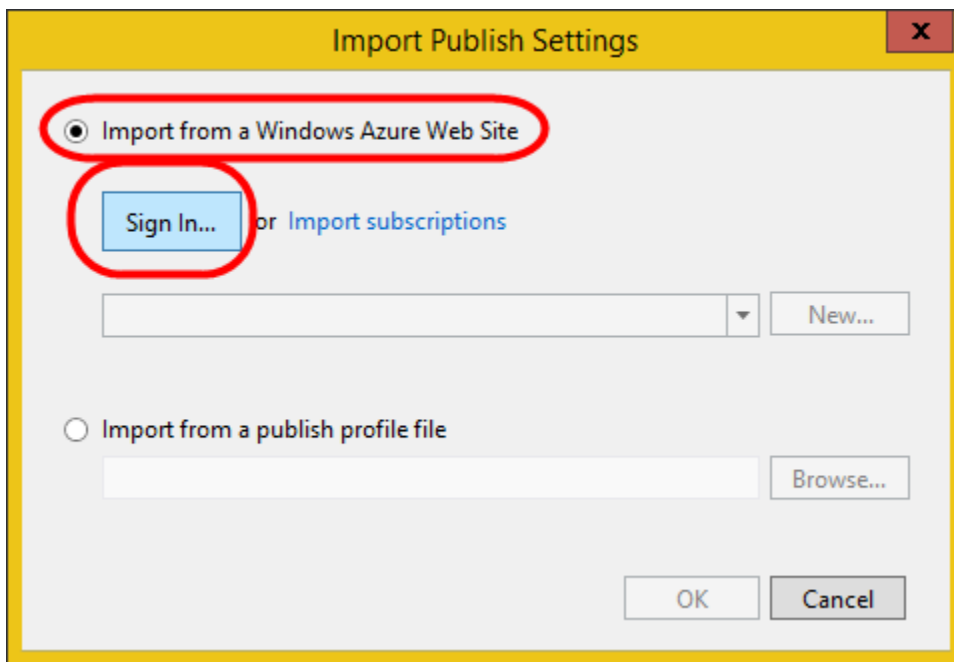
1. After the Web site has been created, return to the **\Expenses WCF Service\Expenses WCF Service.sln** in Visual Studio 2013. If it is running in debug from before, stop the debug session.
2. Right-click **Expenses.WcfService** in **Solution Explorer** and select **Publish Web Site**.



3. In the **Publish Web** dialog, click **Import....** Once we have this profile set up, we can use it in the future and jump right to the deployment step. For this lab, we'll go through the whole process of downloading and importing the publish profile for our Windows Azure subscription.



4. In the **Import Publish Profile** dialog, select the **Import from a Windows Azure web site** option and click the “**Sign In...**” button to sign into your Azure account.



5. Enter your Microsoft Account credentials that are associated with your Azure subscription and click “**Sign in**”

Sign in to your Microsoft account

Sign in

Microsoft account [What's this?](#)

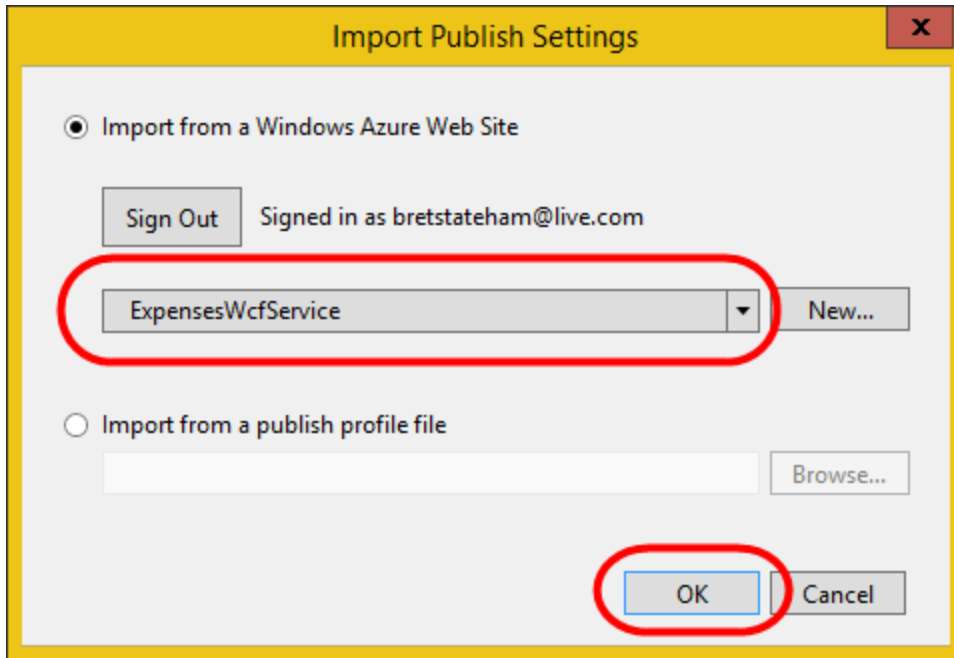
☐ Keep me signed in

[Sign in](#)

[Can't access your account?](#)

[Sign in with a single-use code](#)

6. Once you have signed in, you can select the web site name from the drop-down list, and click **OK**



7. In the **Connection** step of the **Publish Web** wizard, click **Next** to continue.

Publish Web

Publish Web

Profile

Connection

Settings

Preview

ExpensesWcfService *

Publish method: Web Deploy

Server: waws-prod-bay-001.publish.azurewebsites.windows.net:443

Site name: ExpensesWcfService

User name: \$ExpensesWcfService

Password: [Redacted]

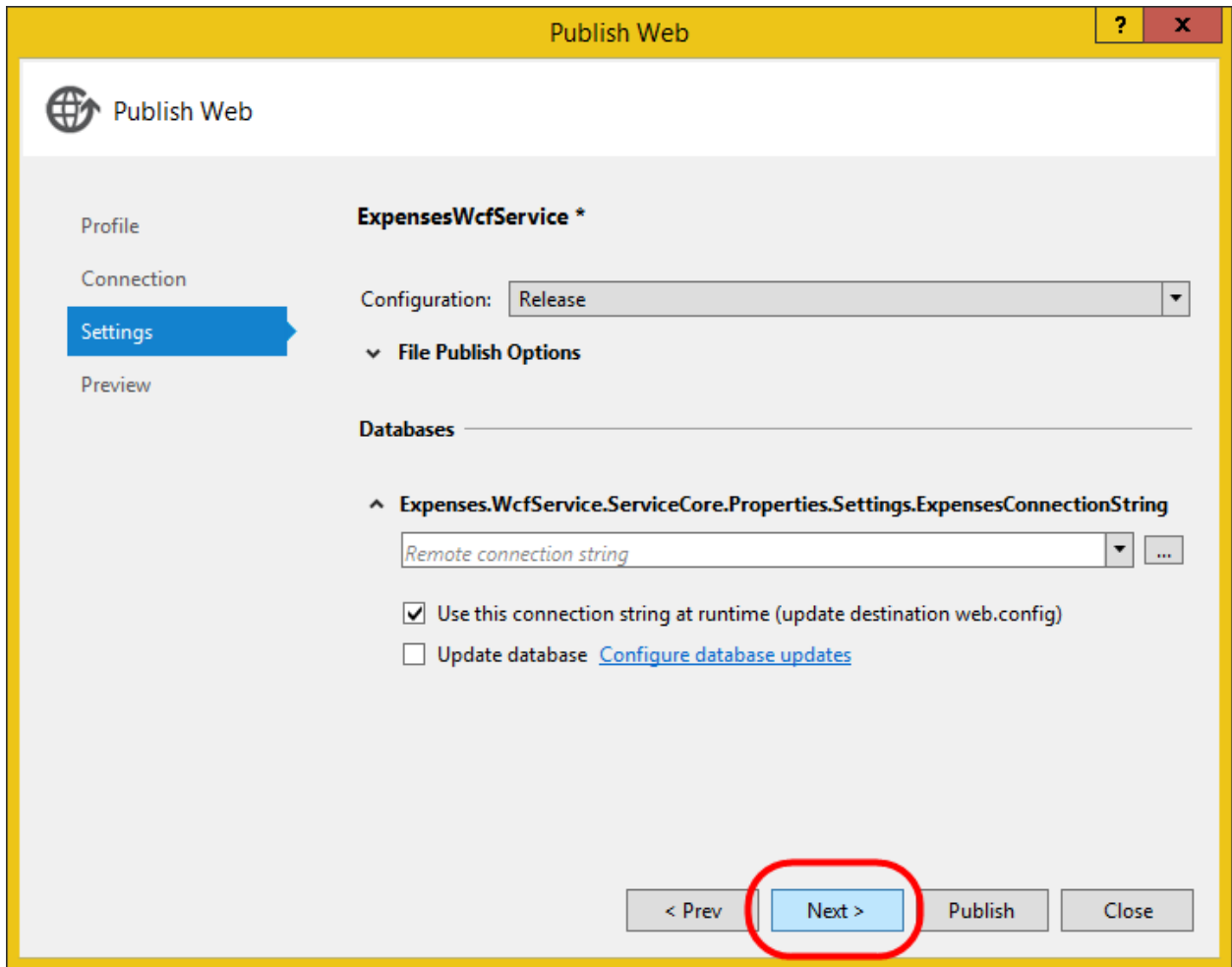
☒ Save password

Destination URL: http://expenseswcfservice.azurewebsites.net

Validate Connection

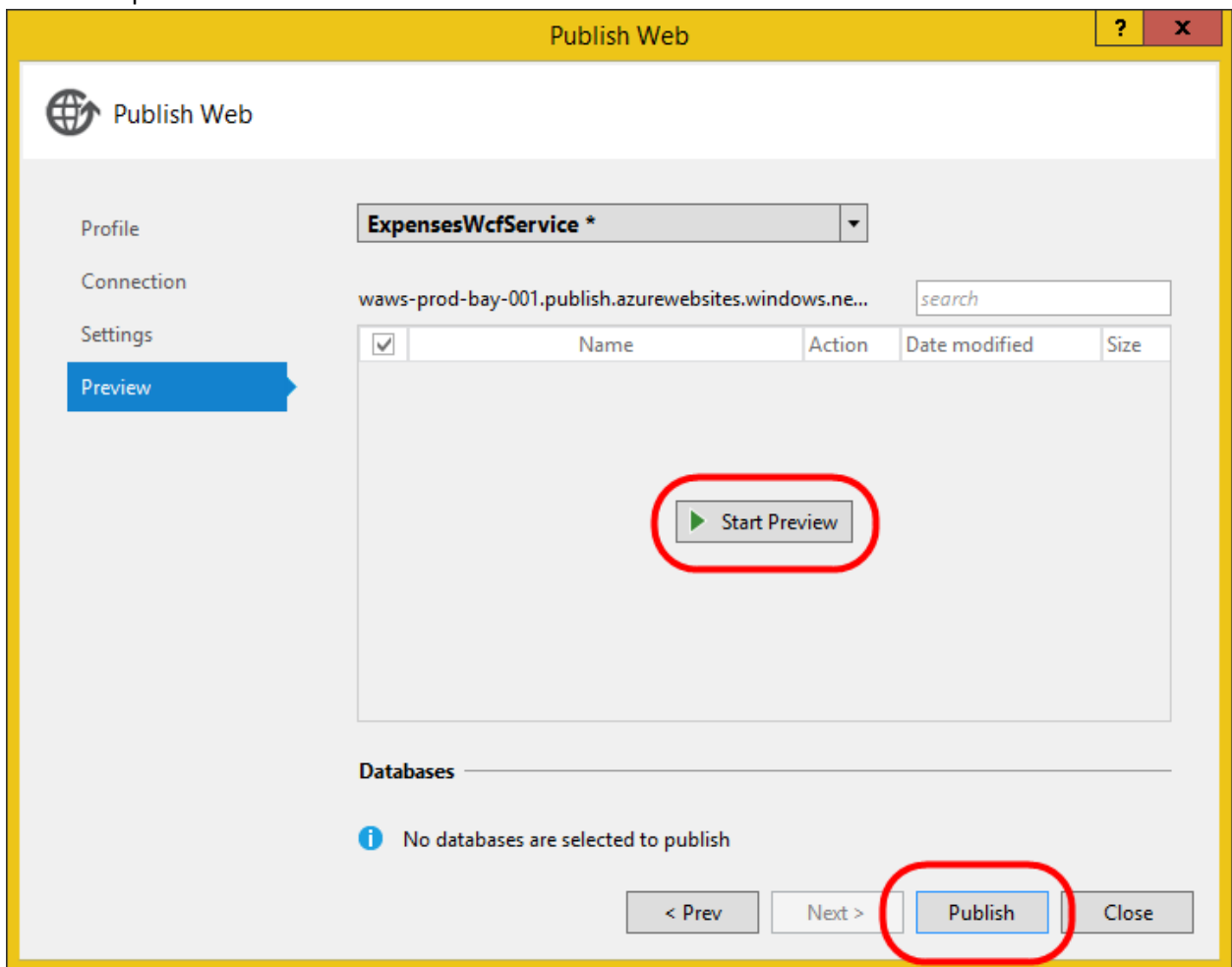
< Prev Next > Publish Close

8. In the **Settings** step of the **Publish Web** wizard, click **Next** to continue.

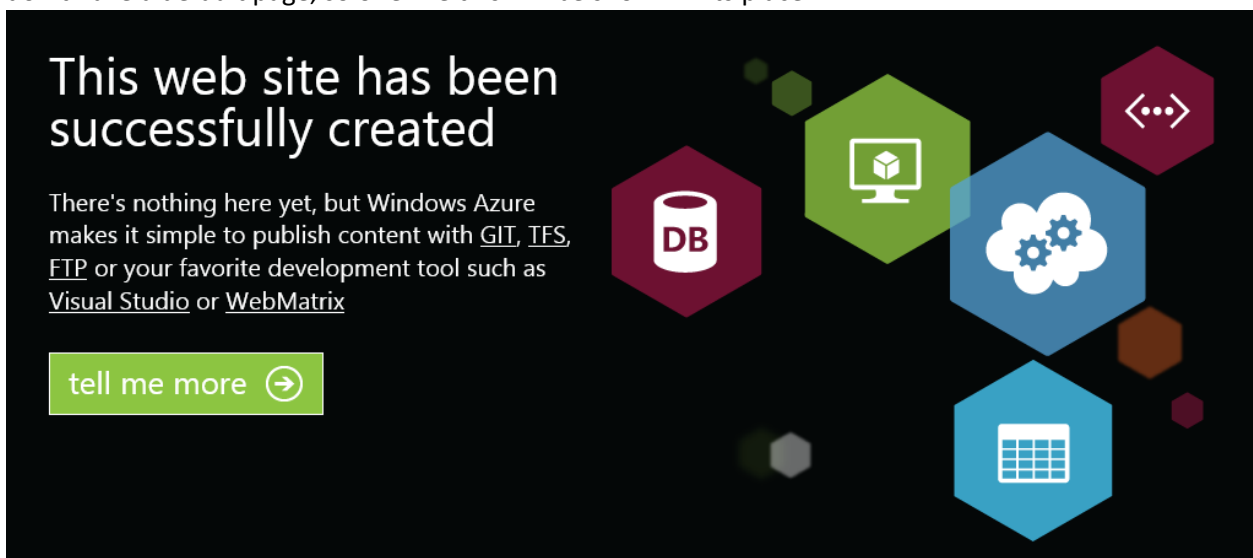


9. In the **Preview** step of the **Publish Web** wizard, click **Start Preview** to see what files have changed and need to be deployed. Since this is our first deployment, it will be all of them. Note that it will only show a few because only the built files that are necessary to run the service are published. Click

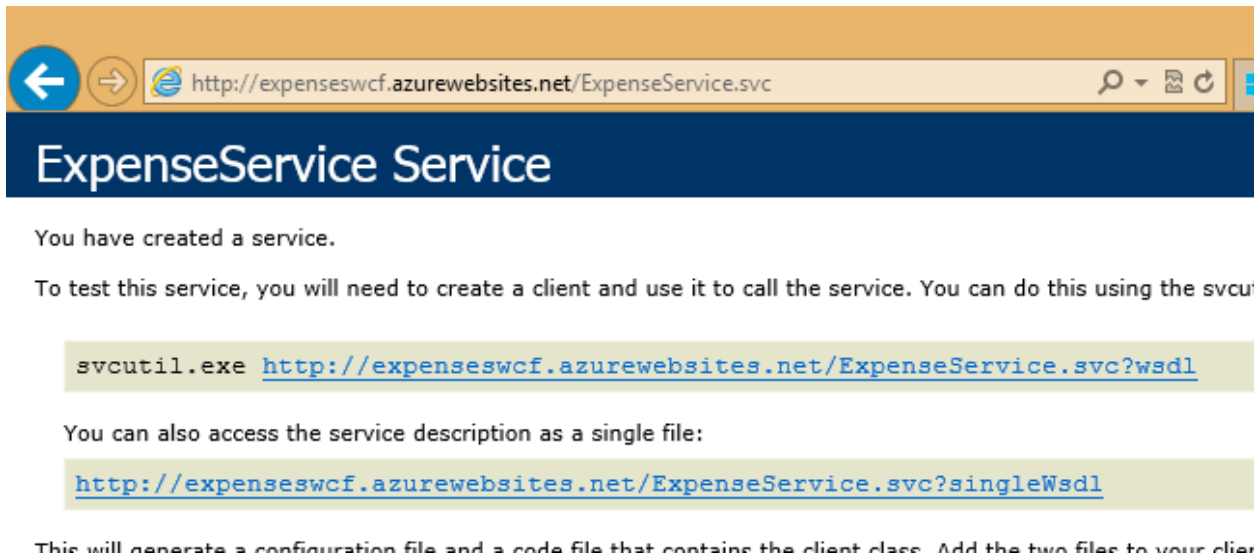
Publish to publish.



10. Once the deployment is complete, a browser window will open to the domain root. In our case, we don't have a default page, so one like this will be shown in its place.



11. To confirm our service is available, add “**ExpenseService.svc**” to the end of the “**http://<yoursitename>.azurewebsites.net/**” URL and press **Enter**. You should see the service default page. Copy the full URL for the service to the clipboard so you can update the client application with it in the next task.



Task 3: Updating the Expenses WPF client to use the new service

In this task, we'll update our WPF client to use the new cloud-hosted WCF service.

1. Open **\Expenses\Expenses.sln** in Visual Studio 2013. Use a new instance of Visual Studio so that both the service and client solutions are open.
2. From the **Expenses.Wpf** project, open **App.config**.
3. In the **configuration/appSettings** section, change the **expenseServiceUrl** to match the domain of your new hosted service. It should look like **http://[site_name_from_earlier].azurewebsites.net/ExpenseService.svc**. It may be easiest to copy the URL used in the previous task to confirm the service deployment.

Note that for any app already deployed across our enterprise, we could just update the config files they use locally to reach this new service endpoint.

4. Press **F5** to launch the application. It should connect to the cloud-hosted service and work as expected.