Demo - Moving Application to Azure

Lab version: 1.0

Last updated: 1/14/2015

Estimated demo delivery time: **20 to 30 minutes**.



Contents

[Overview 3](#_Toc408989056)

[Prerequisites 3](#_Toc408989057)

[Setup 3](#_Toc408989058)

[Exercise 1: Creating new Azure Website 4](#_Toc408989059)

[Task 1: Creating new Azure Website 4](#_Toc408989060)

[Exercise 2: Deploying the Expenses Application to Azure 9](#_Toc408989061)

[Task 1: Creating a Staging Slot for the Service 9](#_Toc408989062)

[Task 2: Publishing Service to Staging Slot 13](#_Toc408989063)

[Task 3: Introducing Azure Integration with Visual Studio Server Explorer 19](#_Toc408989064)

[Task 4: Connecting to Websites using FTP 24](#_Toc408989065)

[Task 5: Using Visual Studio Online Extension to Edit Website using Browser 27](#_Toc408989066)

[Task 6: Utilizing Diagnostic Data and Utilities in the Azure Portal 32](#_Toc408989067)

[Task 7: Configuring and Testing the Expenses Service in the Staging Slot 36](#_Toc408989068)

[Exercise 3: Creating Operational Alerts and Swapping to Production 40](#_Toc408989069)

[Task 1: Creating Web Test Alerts for Production Website 40](#_Toc408989070)

[Task 2: Creating Operational Alerts for Production Website 46](#_Toc408989071)

[Task 3: Swapping the Staging Website with Production 48](#_Toc408989072)

# Overview

In this demo, we show how to create a new Azure Website resource, deploy the Expenses ASP.NET application, and configure the Azure Website. At the end of this demo, the Expenses server-side components will be completely moved over and running in Azure.

# Prerequisites

The following are required to complete this demo:

* [Microsoft Visual Studio 2013](http://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx) (tested with Update 4)
* Internet connection
* [Microsoft Azure](http://azure.microsoft.com/en-us/pricing/free-trial/) subscription
* Expenses.Mvc codebase

# Setup

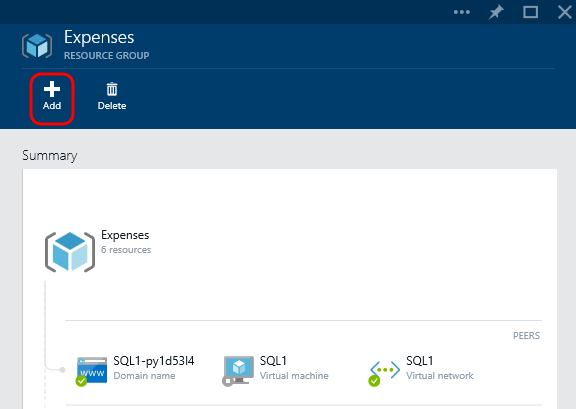
* Load and build the Expenses.Mvc solution to ensure that it builds correctly (**use the solution as it was left at the end of the previous demo “Moving Database to Azure”**).
* Log into the Microsoft Azure subscription that you will be using for demonstration.

# Exercise 1: Creating new Azure Website

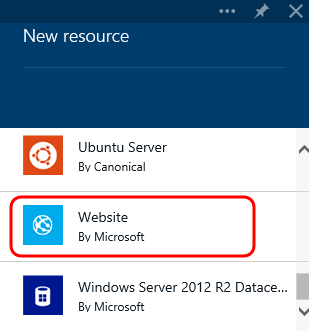
In this exercise, we will demonstrate how to create a new Azure Website resource using the portal.

## Task 1: Creating new Azure Website

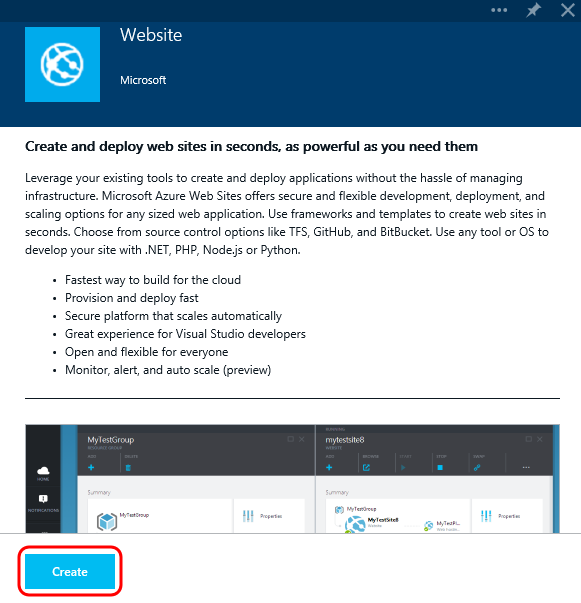
1. Log into the Microsoft Azure [portal](https://portal.azure.com/).
2. Navigate to the Expenses resource group blade.
3. Click the Add button.



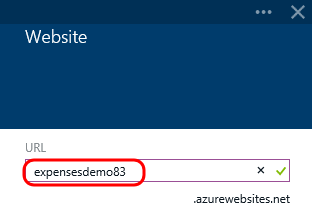
1. Select the Website resource.



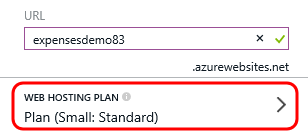
1. Click the Create button.



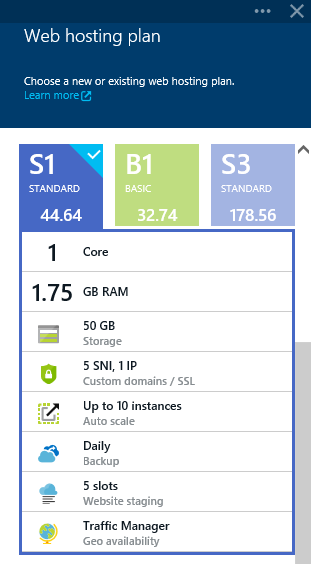
1. Provide a globally unique URL for the new website.



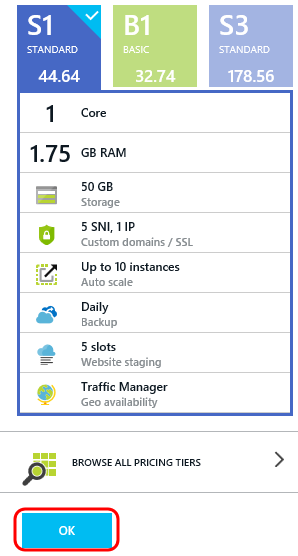
1. Click the Web Hosting Plan option.



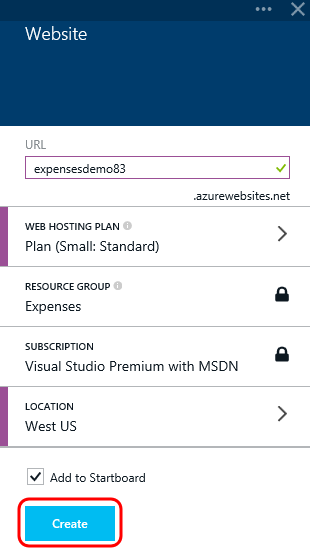
1. In the Web Hosting Plan blade, browse the available pricing tiers that are shown. The Standard S1 tier is selected by default, and you can see here a summary of the processing, memory, storage, and other features provided by the plan including SSL, auto scale, traffic manager support, and so on. Feel free to view the other plans shown on this summary blade before moving on.



1. Press OK to select the default Standard S1 pricing tier option.



1. Since we started the creation process from within the Expenses resource group, the Resource Group option is already set in place.
2. Preferably select the same Location as you did for the Azure SQL Database, so that the Expenses web application and the database are located within the same datacenter.
3. Click Create.



# Exercise 2: Deploying the Expenses Application to Azure

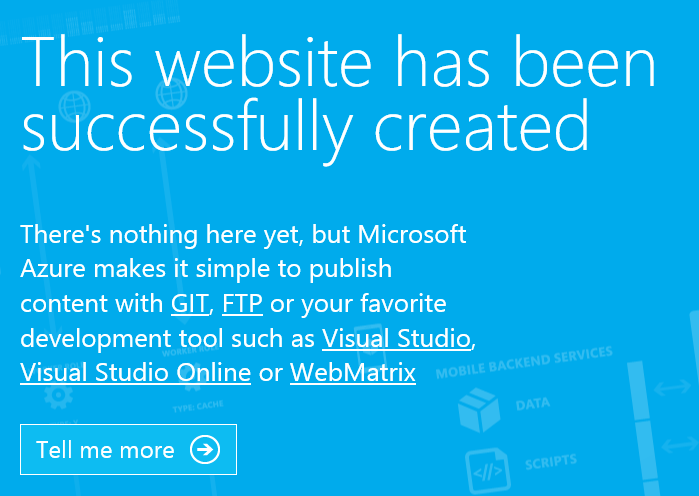
In this exercise, we will demonstrate the steps necessary to deploy the Expenses ASP.NET application to the Azure Website that we just created. This journey will include stops to showcase some of the functionality provided by Azure Websites such as publication options, slots, Visual Studio tooling integration, and diagnostics.

## Task 1: Creating a Staging Slot for the Service

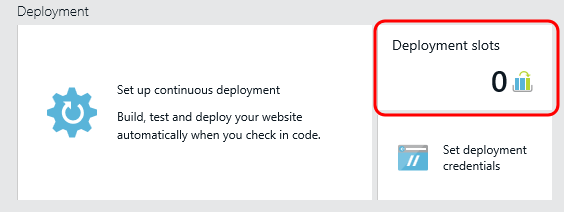
1. Navigate to the new Azure Website blade.
2. Click the Browse button at the top of the Azure Website blade.



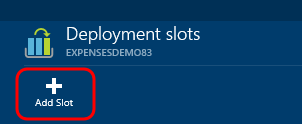
1. Note that a new browser tab is opened to the website, which currently shows a static page indicating that the website has been successfully created. It also has some links that show how to get started publishing your content from Git source control, using FTP, and so on.



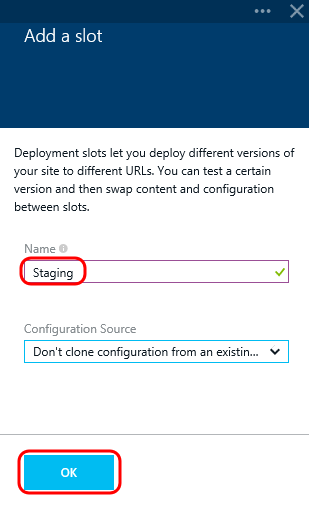
1. Let’s say that we would like to keep this website as a placeholder until we have fully deployed and tested the move to Azure, and in the future allow us to publish changes to the application and test them before fully releasing them to our production site. Close the browser tab that previously opened and return to the Azure portal.
2. Scroll down and click on the Deployment Slots tile.



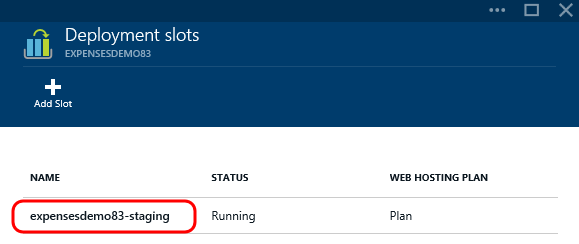
1. Click the Add Slot button.



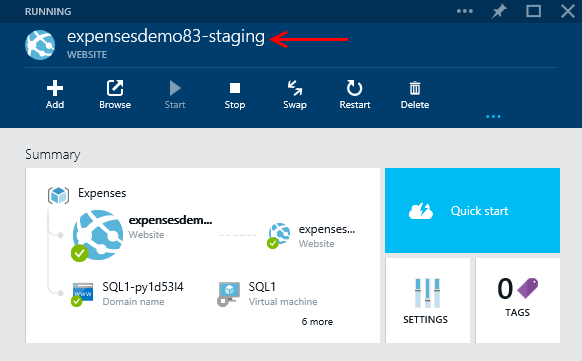
1. Provide a Name of “Staging” for the slot.
2. For Configuration Source, use the default option to not clone the configuration from an existing slot.
3. Click OK.



1. After a few seconds you should see the new slot show up in the Deployment Slots blade. Note that the full name of the deployment slot site consists of the production version of the website, followed by a dash, and finally the name of the slot appended at the end.
2. Click the new deployment slot.

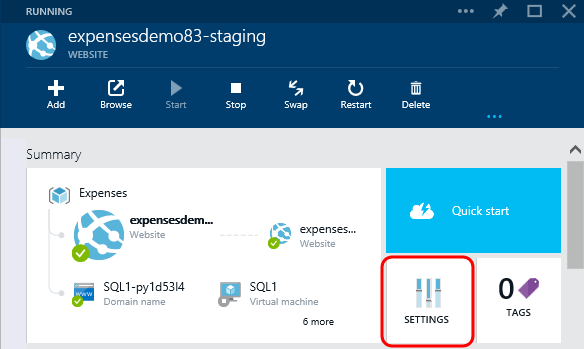


1. Note that the staging slot is a Website resource, just like the production version. It provides you the same level of control and configuration, so for example you may need to setup a different database connection string to differentiate production and staging.

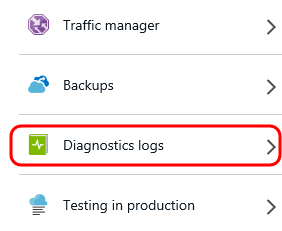


## Task 2: Publishing Service to Staging Slot

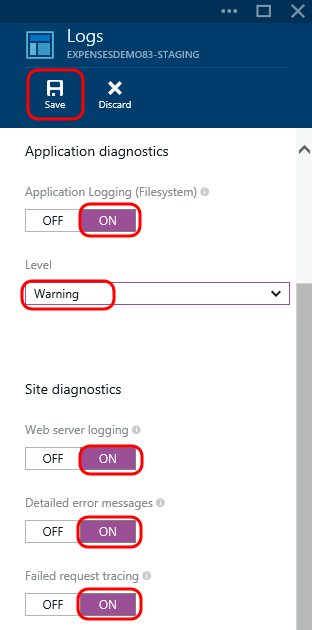
1. Click the Settings tile from within the staging slot blade.



1. Scroll down if necessary and then click on the Diagnostics Logs option.



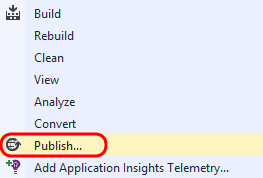
1. In the Logs blade, turn on all logging options including Application Logging at the Warning level, web server logging, detailed error messages, and failed request tracing. Click Save.



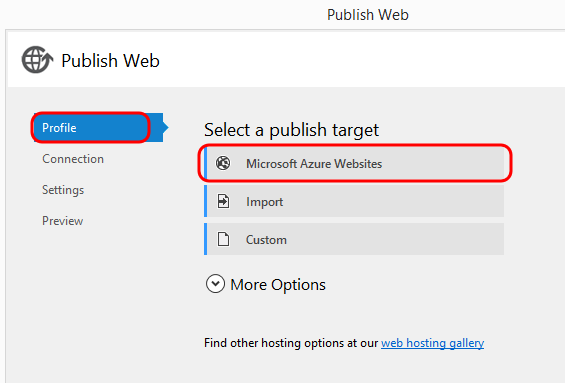
1. Open the Expenses.Mvc solution in Visual Studio and open Web.config from the Expenses.Web project. In a previous demo, we changed the connection string to the Azure SQL database that was created. This isn’t something that we would want to check into source control, so comment out the current connection string and then un-comment the one that is set to LocalDb (do a swap).



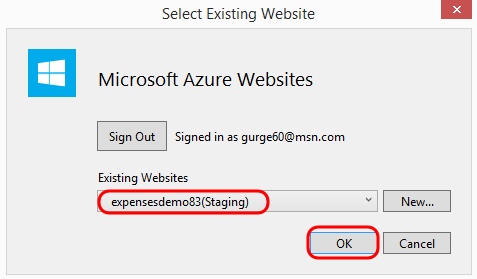
1. In Solution Explorer, right-click on the Expenses.Wweb project node and select the Publish option.



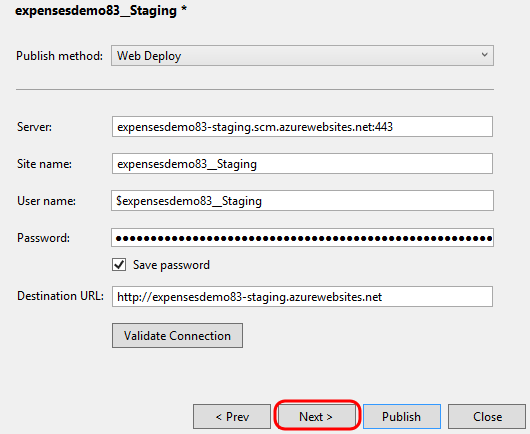
1. Click the Profile step in the wizard to ensure that you are currently on that step, and then click the Microsoft Azure Websites button.



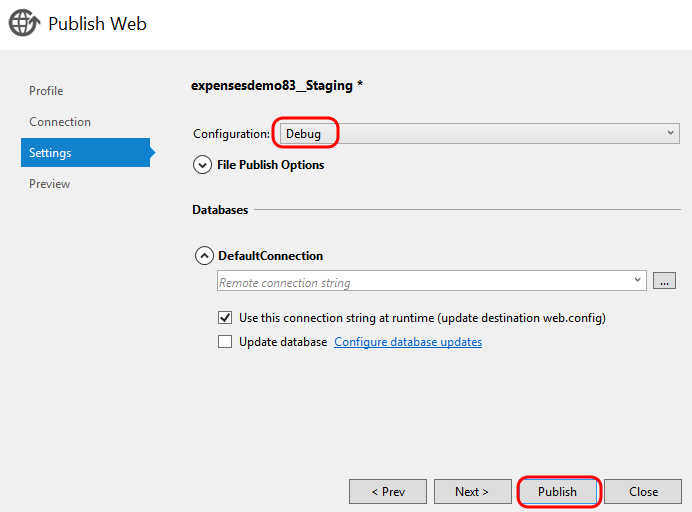
1. If necessary, sign into Azure using your credentials.
2. Select the Existing Websites drop-down and then select the staging website that you just created.
3. Click OK.



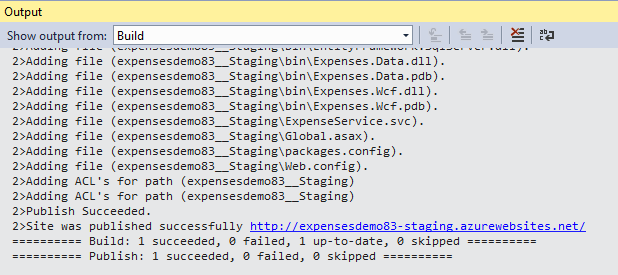
1. In the Connection step of the wizard, note that the details needed to publish to the website using Web Deploy were automatically populated for you.
2. Click Next.



1. In the Settings step, select the Debug configuration since we are publishing to the staging slot.
2. Click Publish.



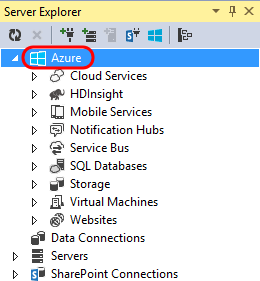
1. You can watch the Output window in Visual Studio to see the publication process in action.



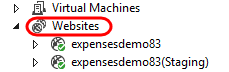
1. Once the publication has finished, a browser tab will open to the URL of the staging slot. We’ll do some additional configuration and testing of the deployed site later on, so go ahead and close the browser window that is opened for now.

## Task 3: Introducing Azure Integration with Visual Studio Server Explorer

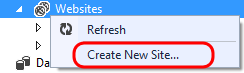
1. In Visual Studio, open the Server Explorer window and expand the Azure node.



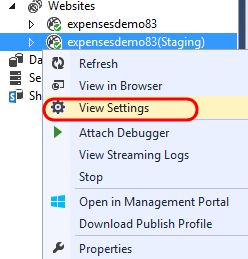
1. The Azure node shows a number of the different types of resources available, including Websites. Expand the Websites node to view the available websites, which should include the demonstration sites that you recently created.



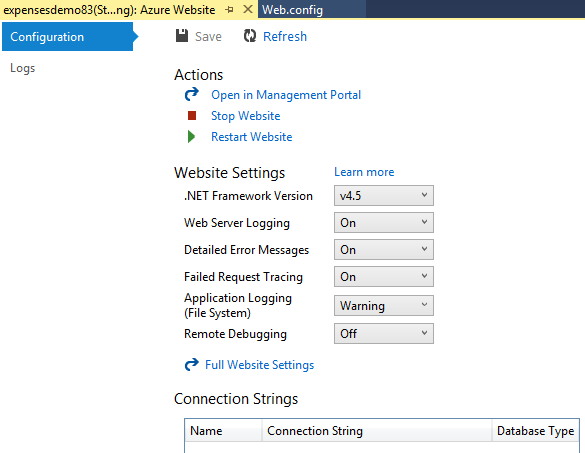
1. Right-click on the Websites node and note that we could also create a website from within Visual Studio if desired. However, we will not create another website at this time.



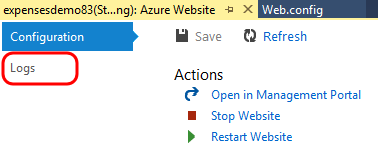
1. Right-click on the node for the staging version of our site and note the available options. Here we can load a browser window to the site, view logs, attach a debugger, and so on. Select the View Settings option.



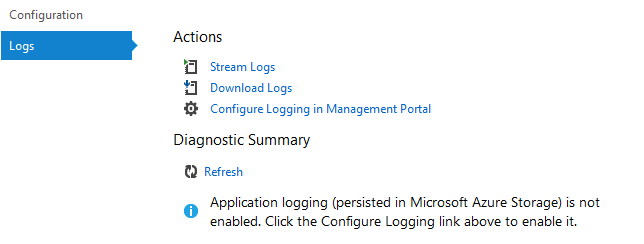
1. In the Settings window, note that we can take a number of actions including stop or restart the site, configure common settings such as .NET Framework version, or configure connections strings and application settings.



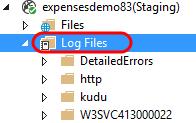
1. Click the Logs tab.



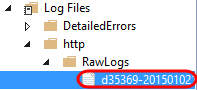
1. This view provides links to view data from a streaming log service, download all logs, or to open and configure logging in the portal. Rather than demonstrate these here, let’s do so directly from Server Explorer. Close the Settings window and return to Server Explorer.



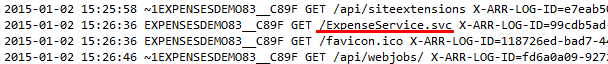
1. Back in Server Explorer, expand the staging website node followed by the Log Files node.



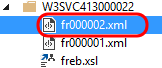
1. Expand the http | RawLogs node and then open one of the logs.



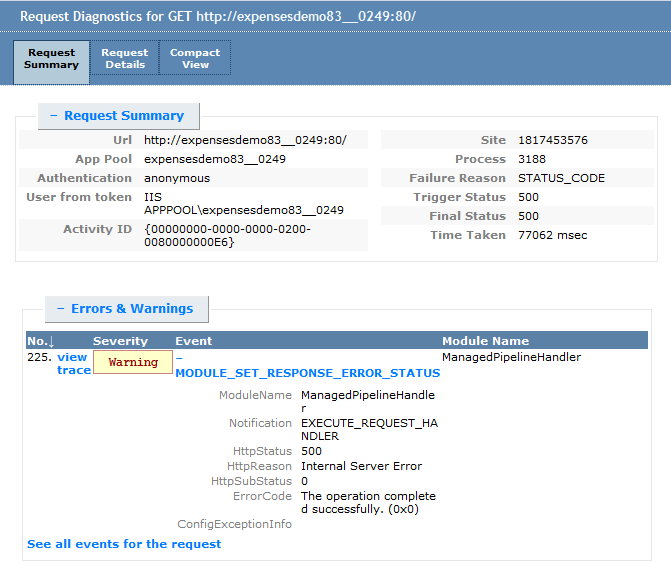
1. Note that this is a raw IIS log file, and that it shows all of the HTTP requests that have been handled so far.



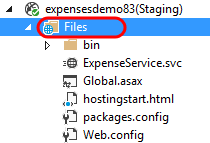
1. Close the IIS log file.
2. Expand the log file node that begins with “W3SVC” and open one of the logs.



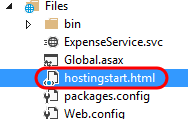
1. This log file represents a failed request. The failed request that you open may be different from the screenshot below.



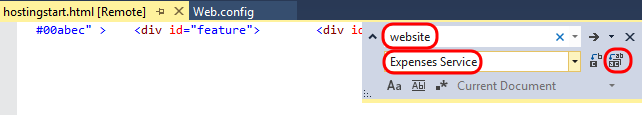
1. Close the failed request log file.
2. Expand the Files node for the staging slot of the website. Note that we can now see the website files that we published earlier.



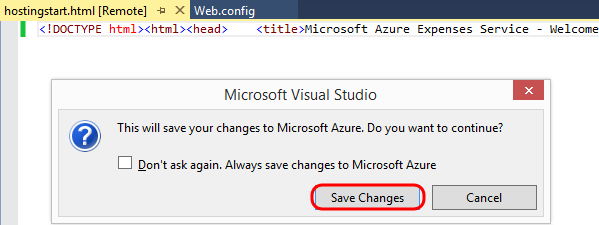
1. Double-click on hostingstart.html to download a copy of the file.



1. Let’s make a minor change to the file to show that we can easily save this back to the server. Select Edit | Find and Replace | Quick Replace from the main menu.
2. In the Find box, enter “website”.
3. In the Replace box, enter “Expenses App”.
4. Click the Replace All button to make the change.

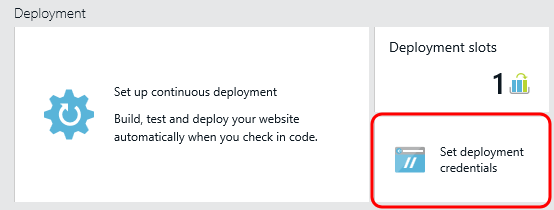


1. Close the Find and Replace dialog.
2. Press Ctrl + S to save the changes back to the server. If you are prompted to confirm your choice, click Save Changes.

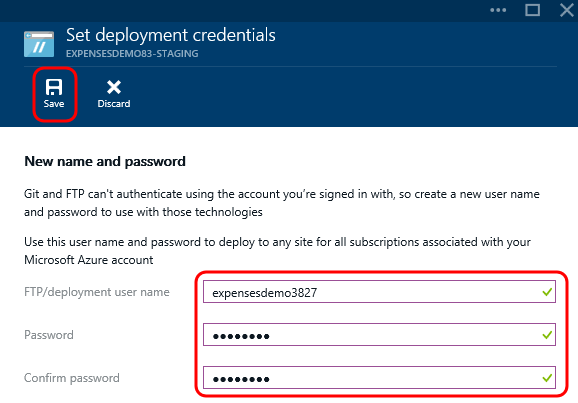


## Task 4: Connecting to Websites using FTP

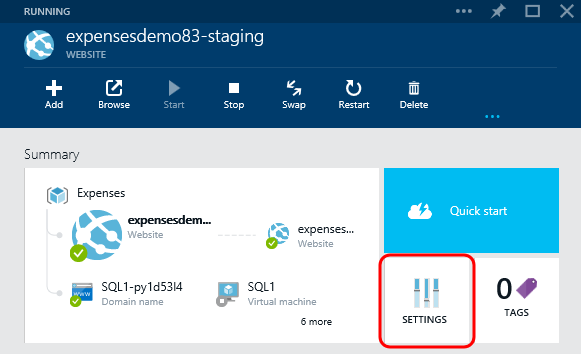
1. Return to the Azure portal and navigate to the production website blade once again.
2. Scroll down to the Deployment section and click the “Set Deployment Credentials” tile.



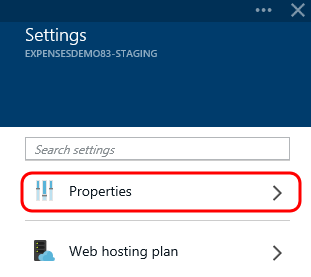
1. If you have already setup deployment credentials, please skip this step. Otherwise, provide a globally unique (create your own, don’t try to use the one shown in this screenshot) user name and credentials to use and then click Save.



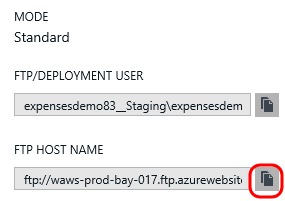
1. Close the Set Deployment Credentials blade and return to the staging website blade.
2. Click the Settings tile.



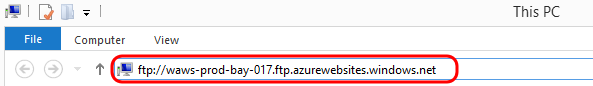
1. Click the Properties option.



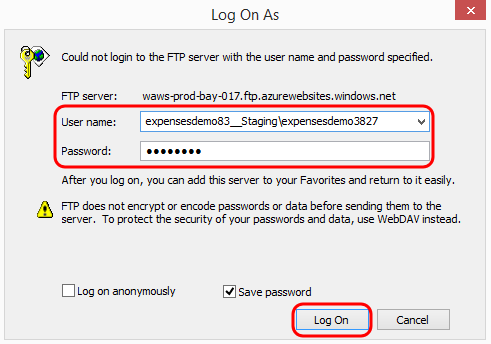
1. Click the Click to Copy button to the right of the FTP Host Name to copy the host name to the clipboard.



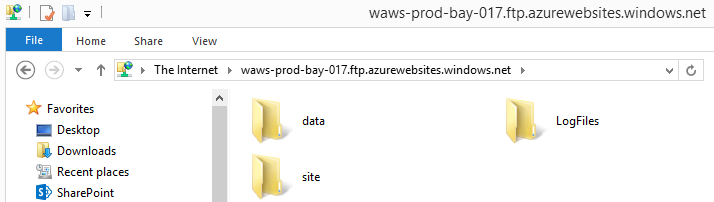
1. Launch an instance of Explorer (Windows + E) and paste in the FTP host name that you copied to the clipboard into the address bar.



1. Press the Enter key.
2. Provide the User Name and Password to log on as. You can copy the User Name to the clipboard from the portal (see FTP/DEPLOYMENT USER property) and then click the Log On button.



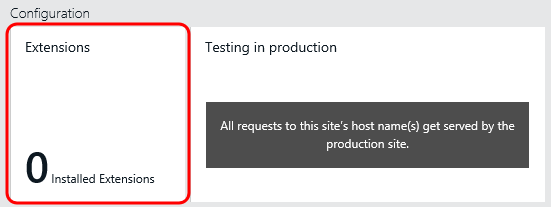
1. After connecting via FTP, take a moment to browse the directory structure shown to note that you have access to the website folder, log files folder, and so on.



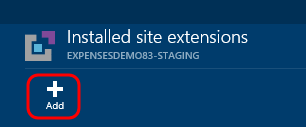
1. Close the Explorer window.

## Task 5: Using Visual Studio Online Extension to Edit Website using Browser

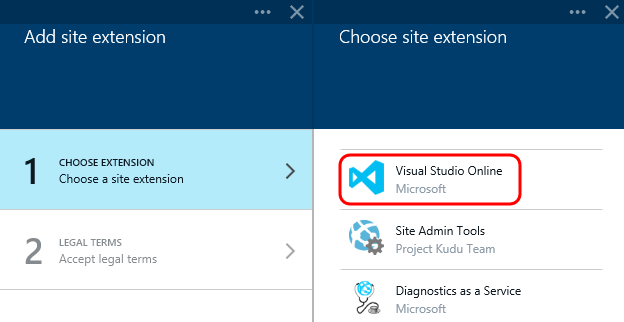
1. Return to the staging website blade in the Azure portal.
2. Scroll down to the Configuration section and click on the Extensions tile.



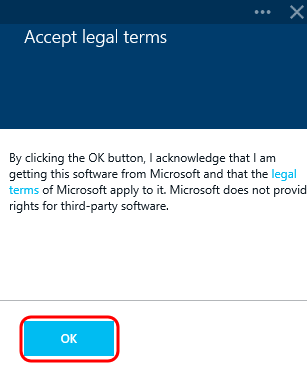
1. Click Add.



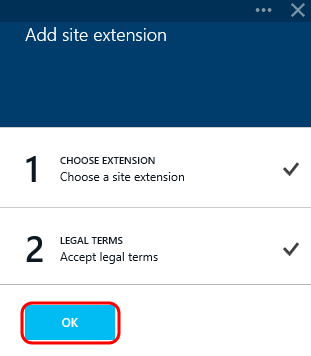
1. Click the Visual Studio Online option.



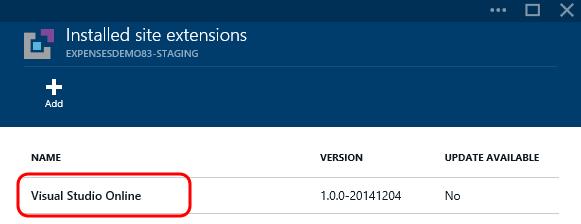
1. Accept the legal terms by clicking OK.



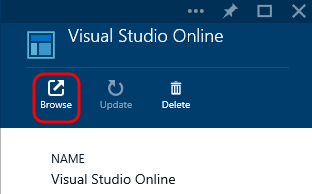
1. Click OK in the Add Site Extension blade to add the extension.



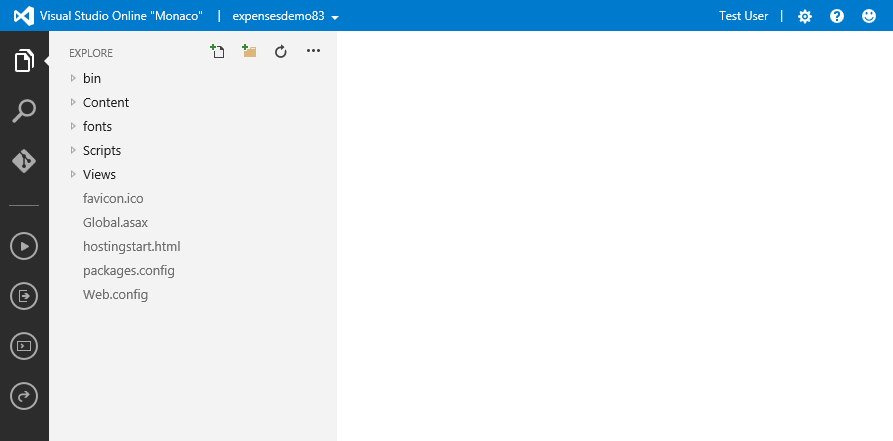
1. Click the Visual Studio Online extension that was installed.



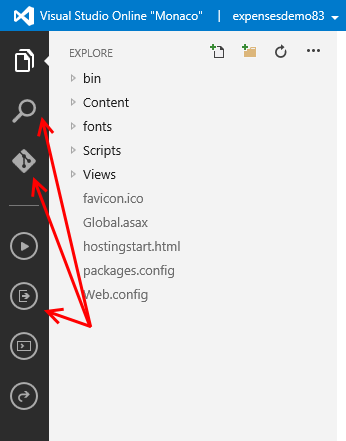
1. Click the Browse button.



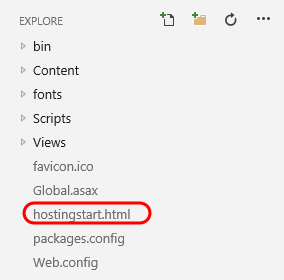
1. In the new browser tab that opens, wait for the Visual Studio Online app to load. The Explore view is selected by default, which allows you to browse and edit your website files directly, create new files and folders, and so on.



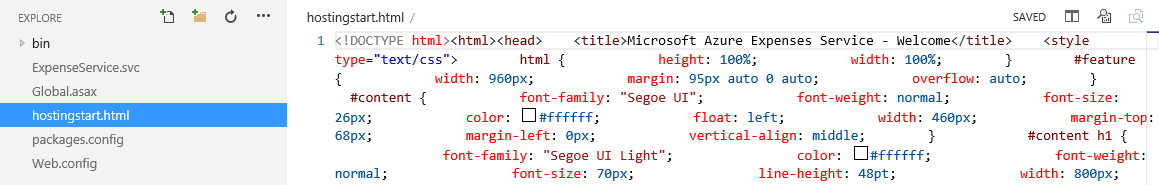
1. Other features (shown on the left-hand side) include code search, integration with Git repository, and more.



1. Click on the hostingstart.html file to load it in the editor.



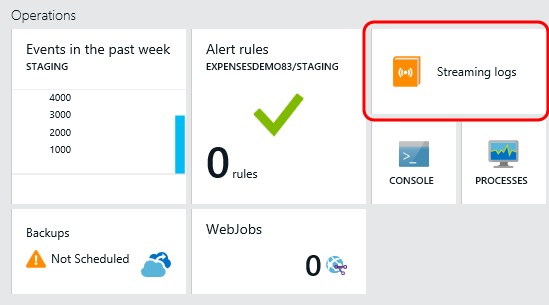
1. At this point we could easily make quick changes in the editor and then save the changes to the server.



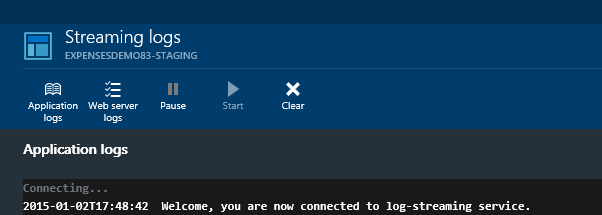
1. Close the browser tab that was opened.

## Task 6: Utilizing Diagnostic Data and Utilities in the Azure Portal

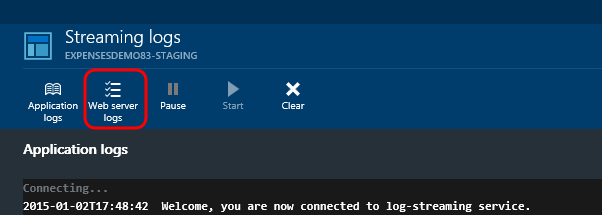
1. Return to the staging website blade in the Azure portal.
2. Locate the Operations section of the website blade and click the Streaming Logs tile.



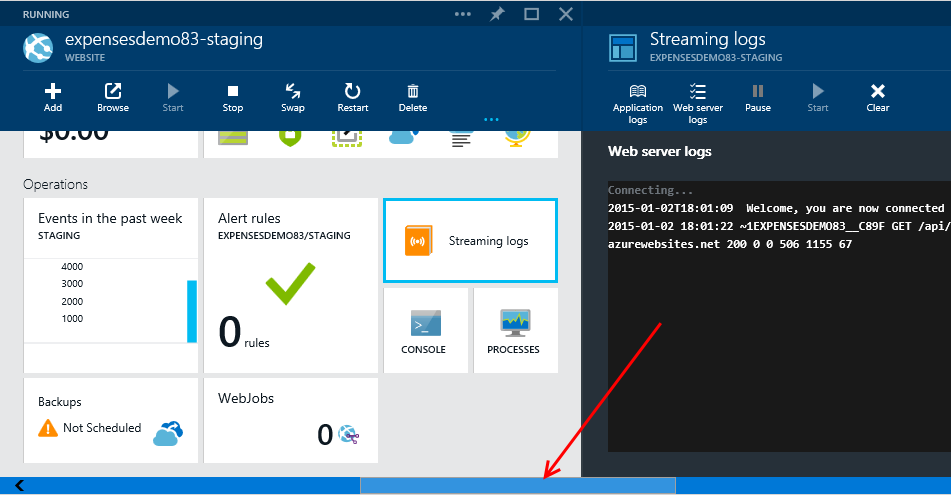
1. Once connected to the streaming log service, this blade will show application logs by default.



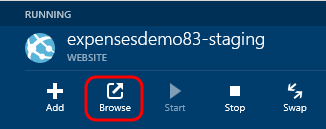
1. Click the Web Server Logs button.



1. Use the scrollbar for the Portal to scroll left until you can see the staging website blade once again (do not close the Streaming Log blade).



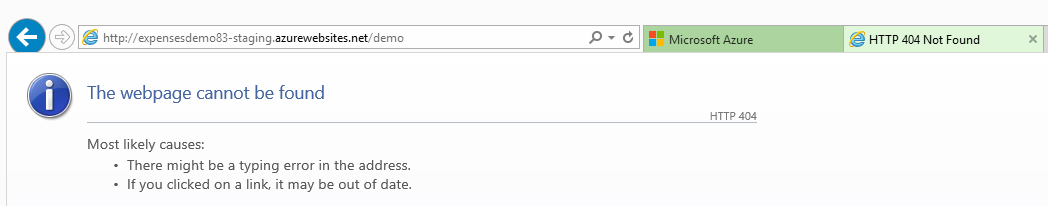
1. Click the Browse button.



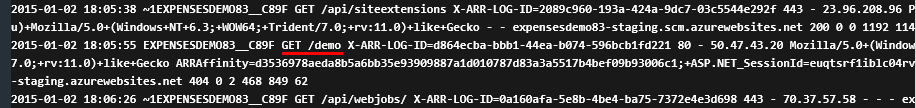
1. In the browser tab that opens to the staging site, modify the URL in the address bar to append “demo” or something else of your choosing. Press Enter to send the request to the server.



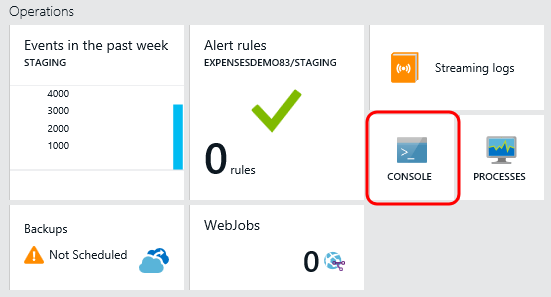
1. After the browser notifies you that “the webpage cannot be found”, close the browser tab to return to the Azure portal.



1. Take a look at the Streaming Logs blade once again and search for the bad request that we just made. Note that it may take a few seconds for the updated logs entries to show.

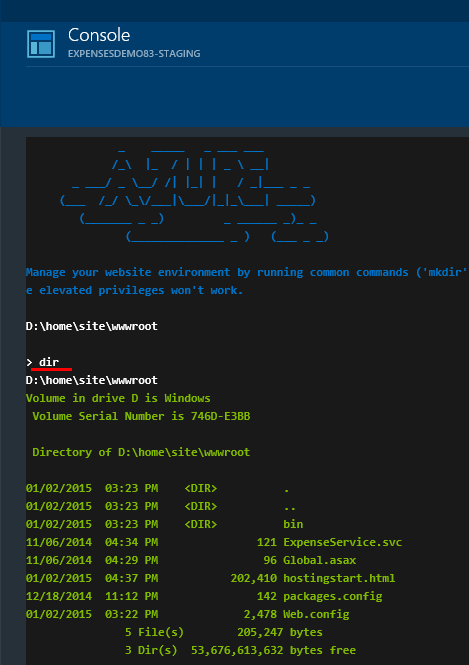


1. Close the Streaming Logs blade to return to the staging website blade.
2. Click the Console tile.

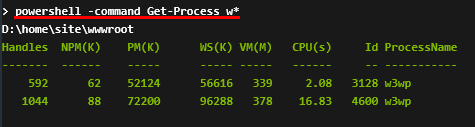


1. In the Console blade, we can run common shell commands as long as they don’t require elevated privileges. Type ‘dir’ and then press Enter.

**Note:** Some common networking utilities (such as ping and nslookup) will not work since they require more privileges than are available in the console. In this case, you can use two separate [tools](http://azure.microsoft.com/blog/2014/10/30/using-vnet-or-hybrid-conn-with-websites/) that have been added to help debug issues, nameresolver.exe and tcpping.exe.



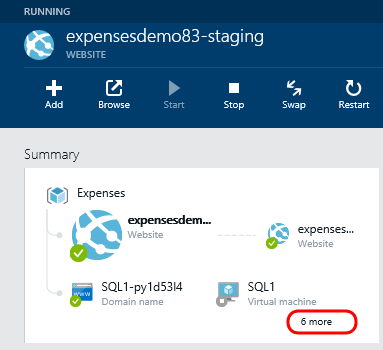
1. It is also possible to take advantage of PowerShell. Type “powershell -command Get-Process w\*” and then press Enter. This example shows how one could list all the processes that start with the letter ‘w’. This shows us information some key metrics about the running IIS worker processes.



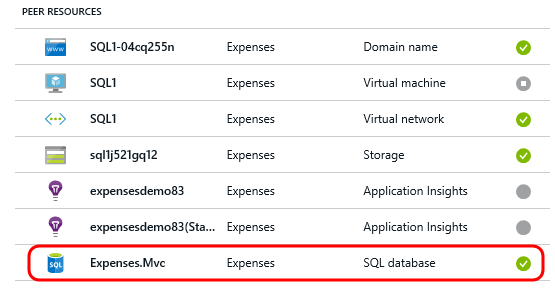
1. Close the Console blade.

## Task 7: Configuring and Testing the Expenses Service in the Staging Slot

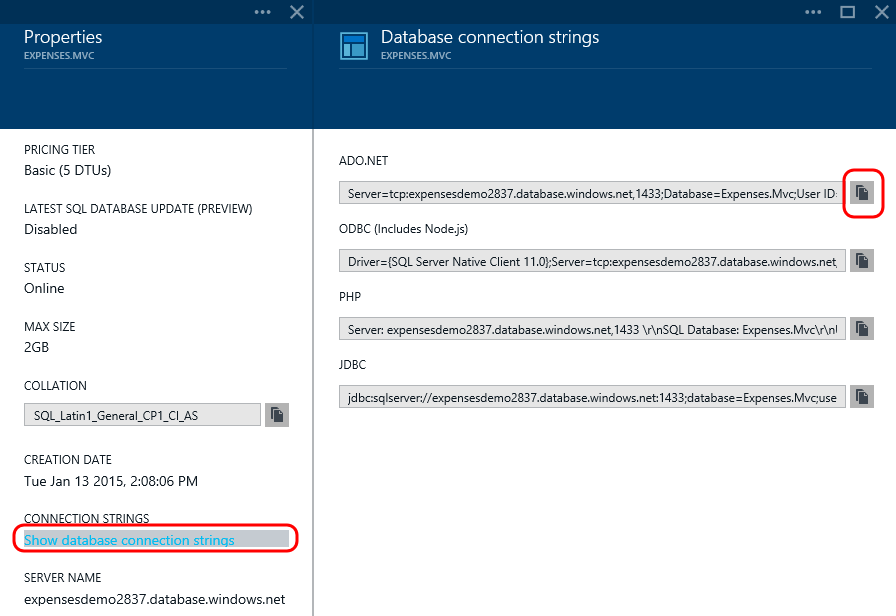
1. In the Azure portal, return to the staging slot of the website.
2. Click on the ‘X more’ link on the Summary graph shown near the top of the staging website blade to view all resources in the Expenses group.



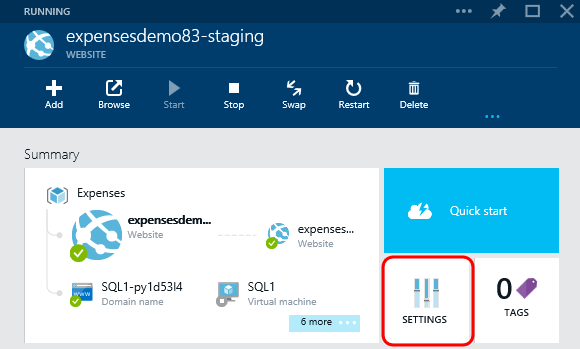
1. Click on the Expenses.Mvc SQL database resource (not the SQL virtual machine).



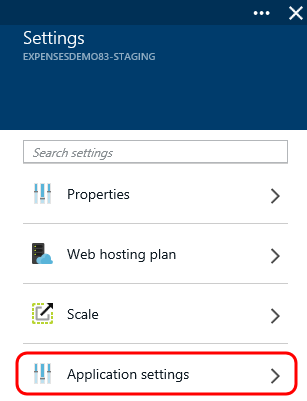
1. Copy the ADO.NET connection string as you learned how to earlier, that is in the Properties blade.



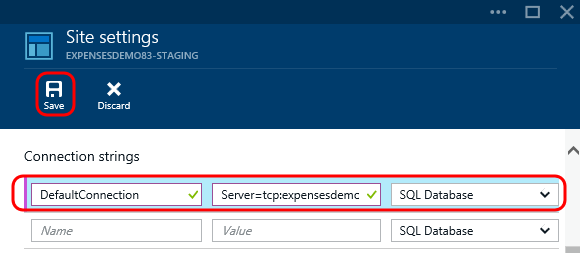
1. Scroll left to return to the staging website blade and then click Settings.



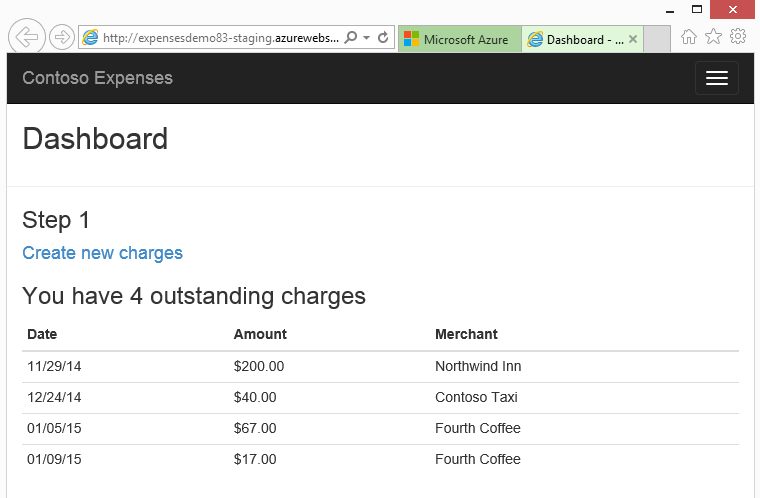
1. Click Application Settings.



1. Locate the Connection String section and paste in the connection string to the Azure SQL database into the Value field.
2. Update the password placeholder as appropriate (just as you did before).
3. Provide a Name of “DefaultConnection”. This will override the connection string in Web.config, which was published with a LocalDb configuration.
4. Click Save.



1. Return to the staging website blade and click the Browse button to load the staging website in a browser tab. The web application should now work as expected running in an Azure Website and served by an Azure SQL database.



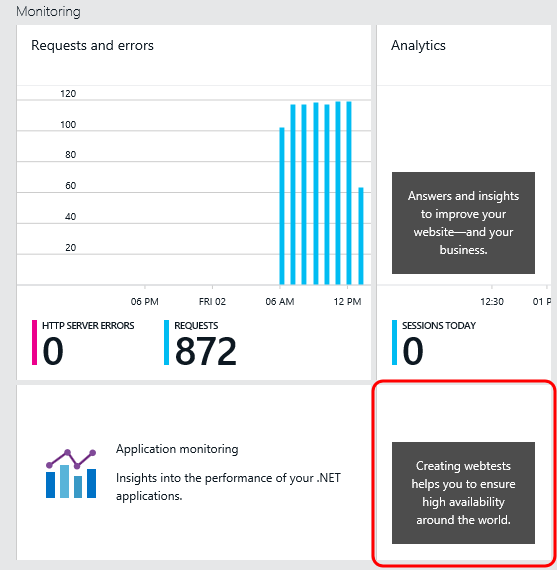
1. Close the browser tab window that was opened to the Expenses application.

# Exercise 3: Creating Operational Alerts and Swapping to Production

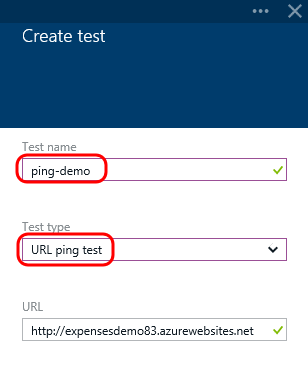
In this exercise, we will close the loop by setting up some alerts and then swapping the staging slot with the production slot, thereby finishing up with our release process.

## Task 1: Creating Web Test Alerts for Production Website

1. In the Azure portal, navigate to the production version of the website hosting the Expenses service, scroll down to the Monitoring section, and finally click on the tile that currently reads “Creating webtests helps you ensure high availability…”



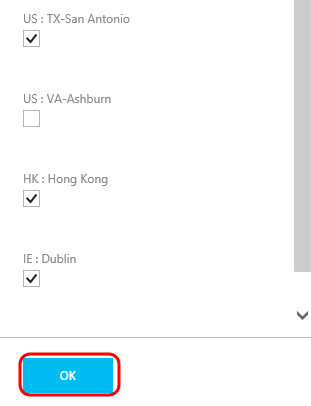
1. Provide a name for this availability test of “ping-demo” and use the default Test Type of “URL ping test”. Note that the other test type is a “multi-step test”, which is basically just a .webtest file that can be authored using Visual Studio.



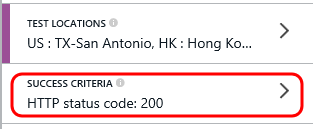
1. Click the Test Locations option.



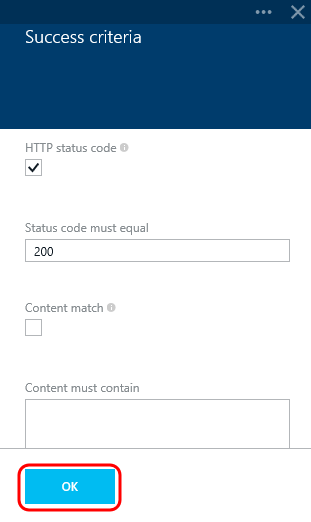
1. Select up to three locations of your choosing and then click OK.



1. Click the Success Criteria option.



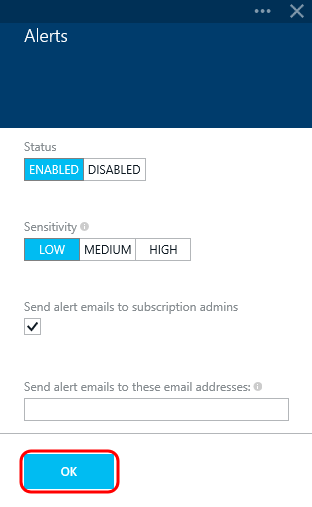
1. By default, this availability test will report success as long as the request to the specified URL returns a HTTP code of 200. Note that we could also specify that the returned content must match or contain specified text. Click OK.



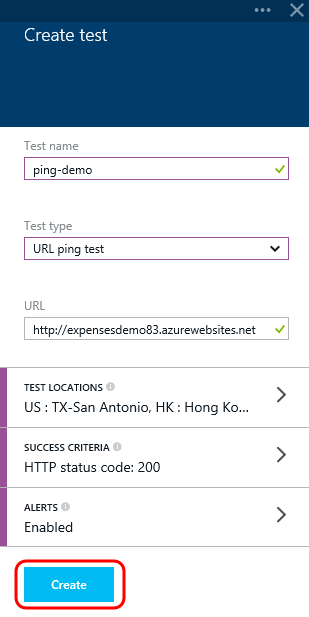
1. Click the Alerts option.



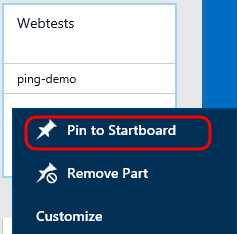
1. By default, the alert is enabled and set to a low sensitivity threshold, although this can be adjusted here if desired. In addition, it is also possible to specify additional email addresses that should be notified about the configured alert. Click OK.



1. Click Create.

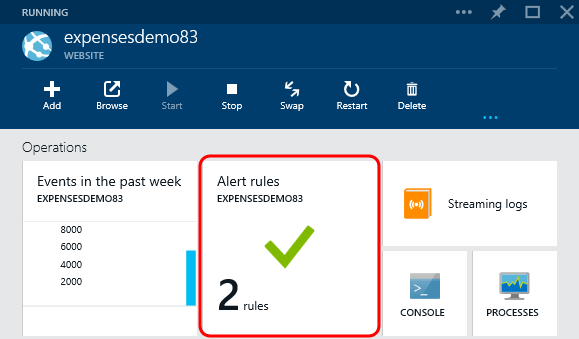


1. After the new test has been saved, right-click on the Webtests tile and select the Pin to Startboard option. This will allow us to quickly drill into the tests straight from our customized dashboard.

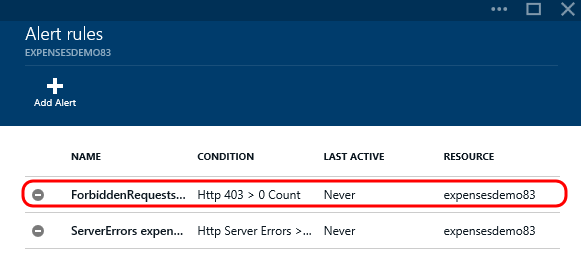


## Task 2: Creating Operational Alerts for Production Website

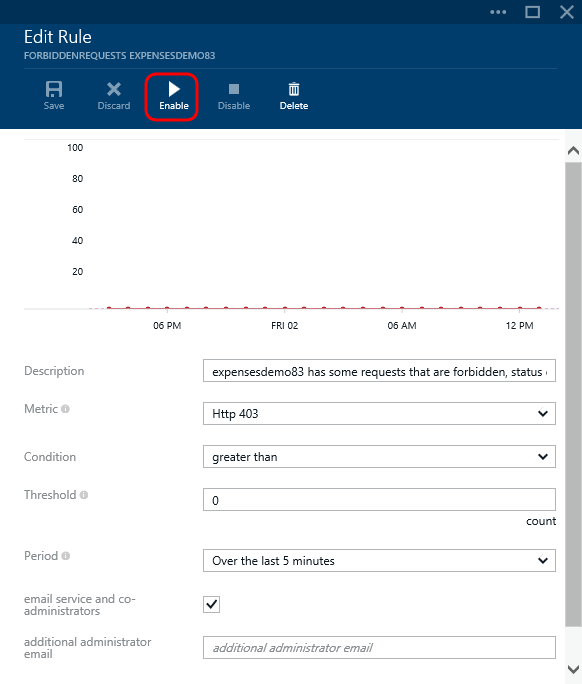
1. Scroll down to the Operations section and then select the Alert Rules tile.



1. Note that by default there are already two rules that have been added, but that they are currently inactive. Click the first alert rule named ForbiddenRequests to open its definition.

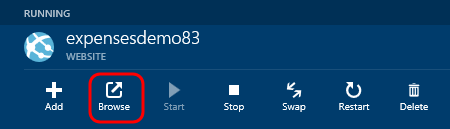


1. Although there may not be any forbidden requests logged at this point in time, we can see that the alert definition is set to be when there are HTTP 403 error codes returned to clients over a five minute period of time. Note that additional email addresses can be configured here as well.
2. Click the Enable button.

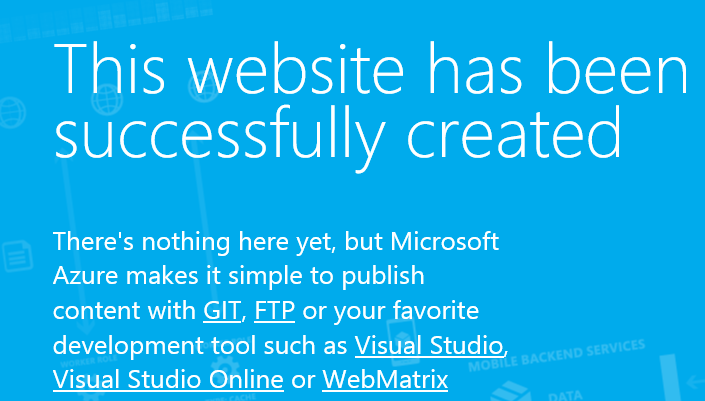


## Task 3: Swapping the Staging Website with Production

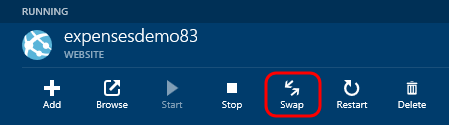
1. Navigate to the production website blade and click the Browse button.



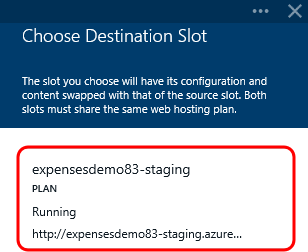
1. In the browser tab that opens, note that the production website still reads “This website has been successfully created”. This is expected, as we published the web application to the staging site earlier.



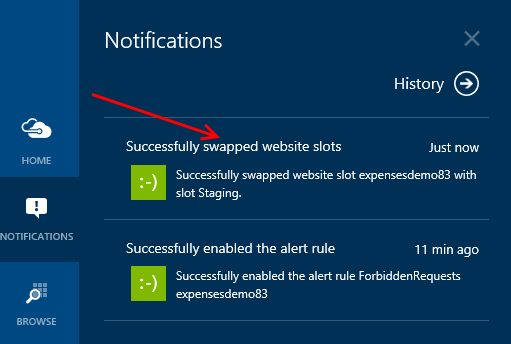
1. Navigate to the production website blade once again and then click the Swap button.



1. In the Choose Destination Slot blade, select the staging slot.



1. Note that after just a few seconds, the Notifications hub indicates that the swap was a success.



1. Browse to the production website once again, refresh the page if necessary, and note that it now shows the Expenses web application as expected.

**Note:** This swap does a virtual IP swap behind the scenes, meaning that any new traffic to the production website will be routed to the website slot that we setup and configured as a staging slot. Therefore, it is important to note that other settings such as database connection strings and the fact that we deployed a debug build to the staging slot need to be taken into account when doing this.