Azure Tips and Tricks

See: Welcome | Azure Tips and Tricks (microsoft.github.io)

How to secure a Blazor application with Azure Active Directory

#Secure your applications with Azure Active Directory

You can use Azure Active Directory (AAD) (opens new window)to make users authenticate and authorize to use your app. AAD provides an intelligent identity-as-a-service that protects your application. And it is easy to use and implement.

In this post, we will create a new Blazor WebAssembly (opens new window)application and implement Azure Active Directory (opens new window)in it, so that users can authenticate themselves in the app.

#Prerequisites

If you want to follow along, you'll need the following:

- An Azure subscription (If you don't have an Azure subscription, create a free account (opens new window)before you begin)
- .NET Core latest version SDK(opens new window)

#Implementing Azure Active Directory in a Blazor WebAssembly application

We will secure a standalone, Blazor WebAssembly application with Azure Active Directory (AAD). To do this we'll start by registering an application in AAD in the Azure portal.

1. Go to the Azure portal(opens new window)
2. Select the **Menu in the top-left corner** and select **Azure Active Directory**
3. In AAD, select **App registrations**
4. Select **New registration**. This will bring up the **Register an application** blade. We'll use this to register the Blazor application
    1. Fill in a **Name** for the application
    2. Leave the **Supported account types** to **Accounts in this organizational directory only**
    3. In redirect URI, select **Web** and fill in https://localhost:5001/authentication/login-callback. We'll use the 5001

port for the app as that is the default port for it in IIS express, but it could be that we need to change this later when the application is created

4. Select **Register** to create the app registration

Dashboard > | App registrations >

Register an application

* Name

The user-facing display name for this application (this can be changed later).

BlazorStandAlone

Supported account types

Who can use this application or access this API?

- (●) Accounts in this organizational directory only (only - Single tenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ( ) Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Help me choose...

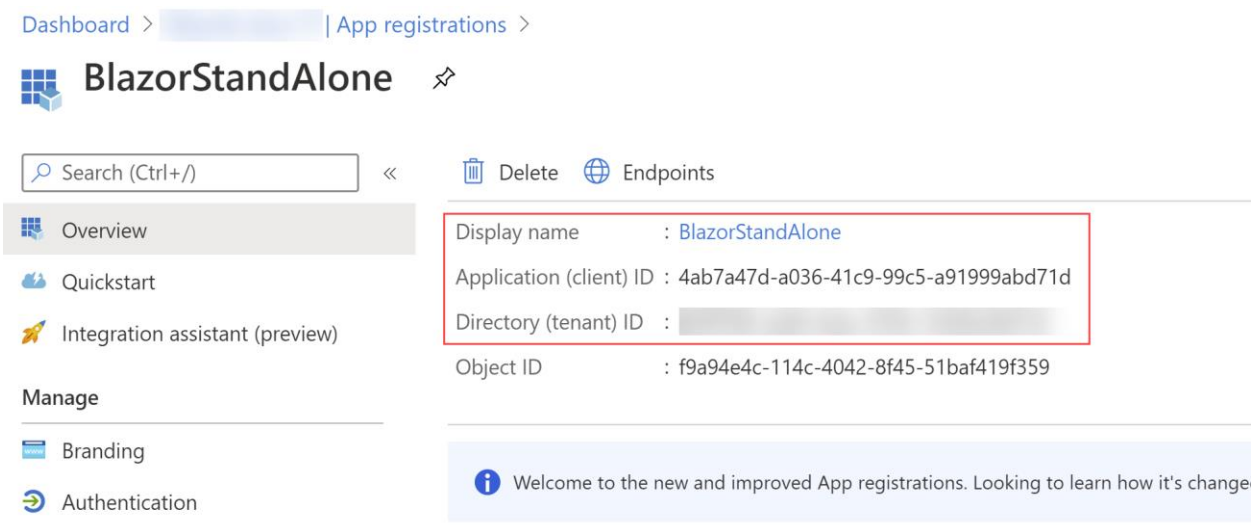
Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web | https://localhost:5001/authentication/login-callback

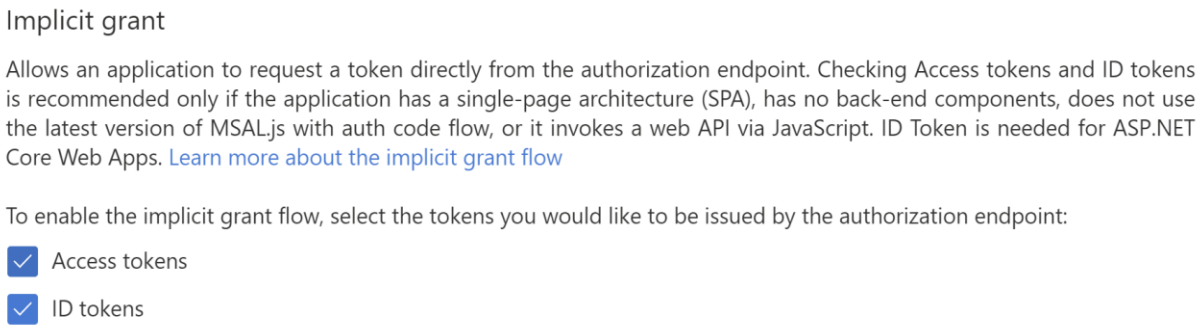
(Register an application in the Azure portal)

5. When the app has been registered, you'll see the **client id and tenant id**. Copy these as we'll need them later



(App details in the Azure portal)

6. Select the **Authentication** menu
7. Under **Implicit grant**, check the boxes for **Access tokens and ID tokens**



(Enable implicit grant in the Azure portal)

8. Select **Save**
9. That's it! Now, we can create the Blazor application. Open a command prompt and create the Blazor app with the following command, where you fill in the client and tenant id and the name of the app registration:

```
1
```

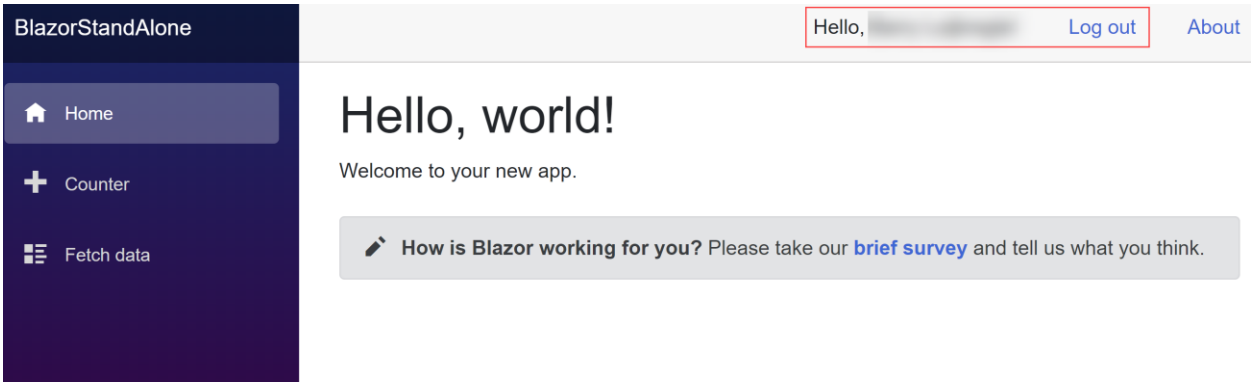
10. The previous command created a folder that contains the Blazor WebAssembly application. **Navigate to the folder** in the command prompt
11. Now run the following command to compile the app:

```
1
```

12. And run the app with:

```
1
```

13. The output will show you the URL on which the application is running. Check if the port of the URL is the same as we configured in the app registration in AAD (5001). If it isn't, change the app registration to match the port. **Open a browser and navigate to the URL** of the Blazor app. The application has a **login** menu item that you use to authenticate. Click on it and log in with your Azure account or another account that is present in your AAD tenant. You'll be logged in and see your name



(Authenticated in the Blazor WebAssembly app)

#**Conclusion**

Azure Active Directory (opens new window)enables you to secure your applications without worrying about complicated security setup. You can use it to secure all kinds of applications, including Blazor WebAssembly apps. (opens new window). Go and check it out!

Quickly Set Up Azure Active Directory with Azure App Services

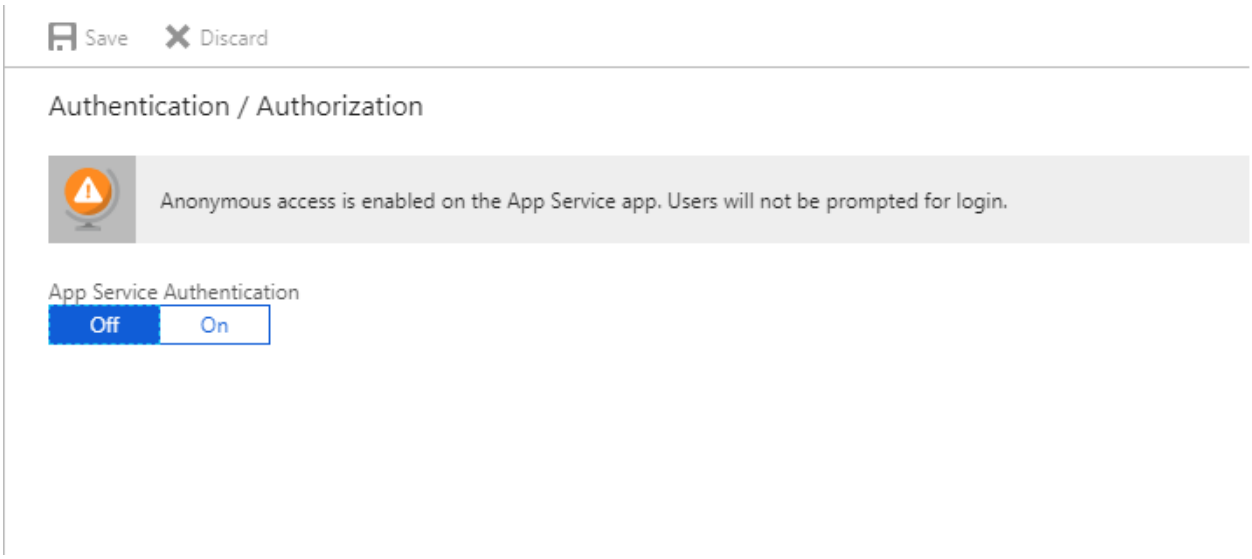
A while ago, I did a post on Quick and Dirty User Authentication with Azure Web Apps and MVC5 (opens new window), where I created a simple web app that used forms authentication. Since then, I've been asked if I could address how to use the **Settings -> Authentication / Authorization** feature to turn on AAD for an existing web app. In this post, we'll take a look at setting up Azure Active Directory with Azure App Services.

#My Requirements

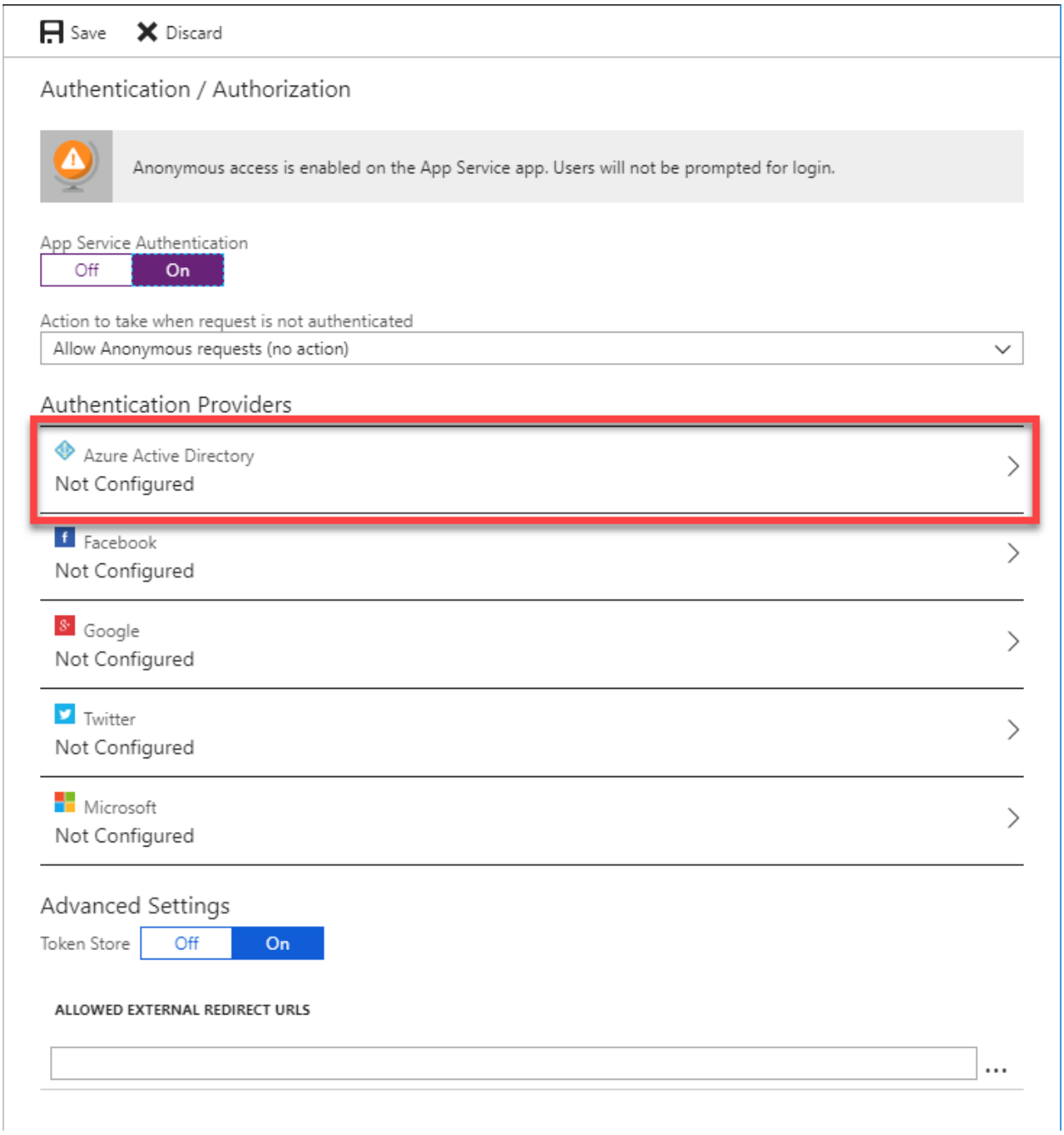
- Any user on my AAD will be able to log in.
- I won't write or add any code to my web app.
- I want to do this with the FREE Tier of Azure App Service Web Apps.

#How to Set Up Azure Active Directory with an App Service Web App

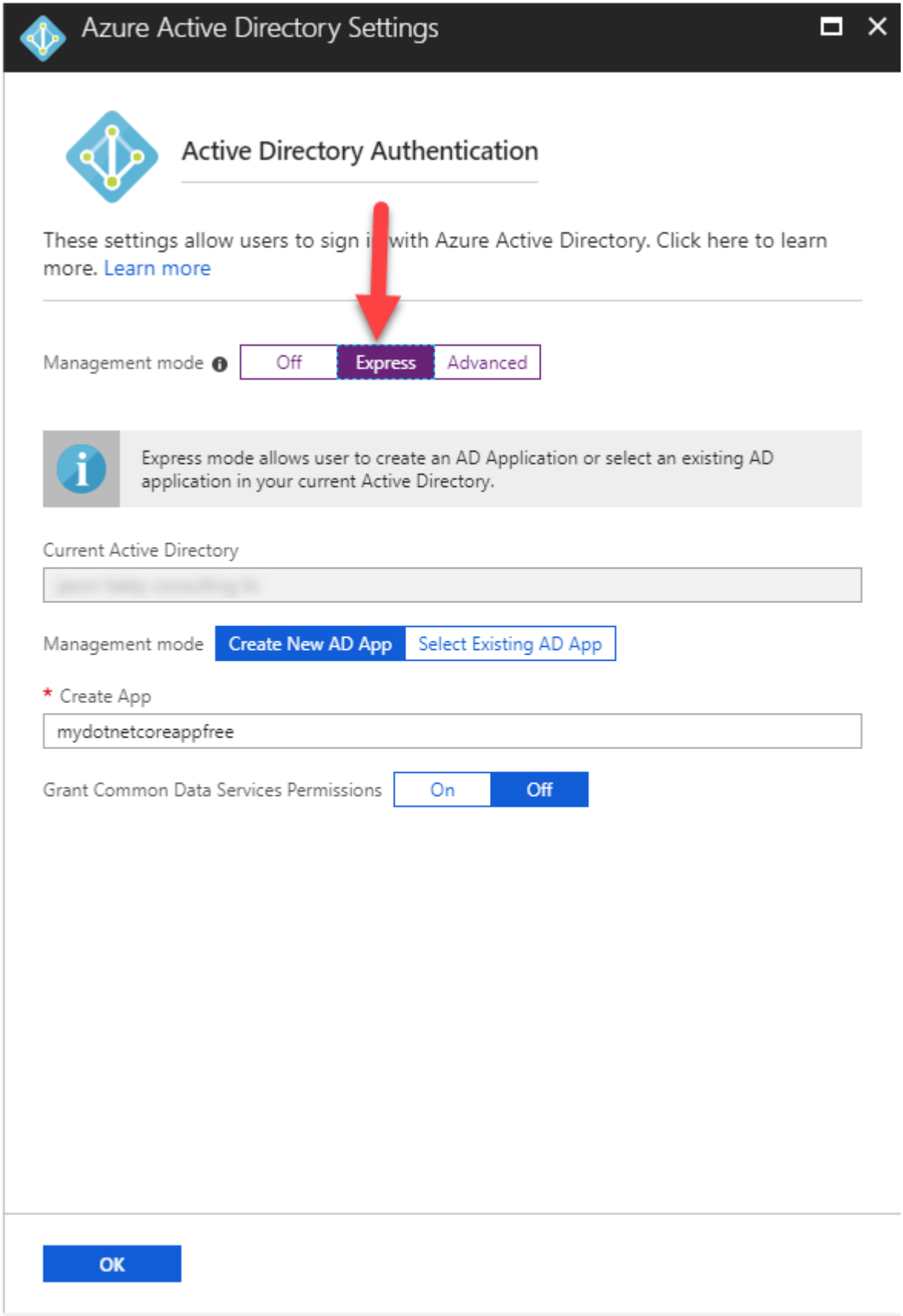
Go to the Azure portal and select my web app and click on **Authentication / Authorization** under **Settings** to get started.



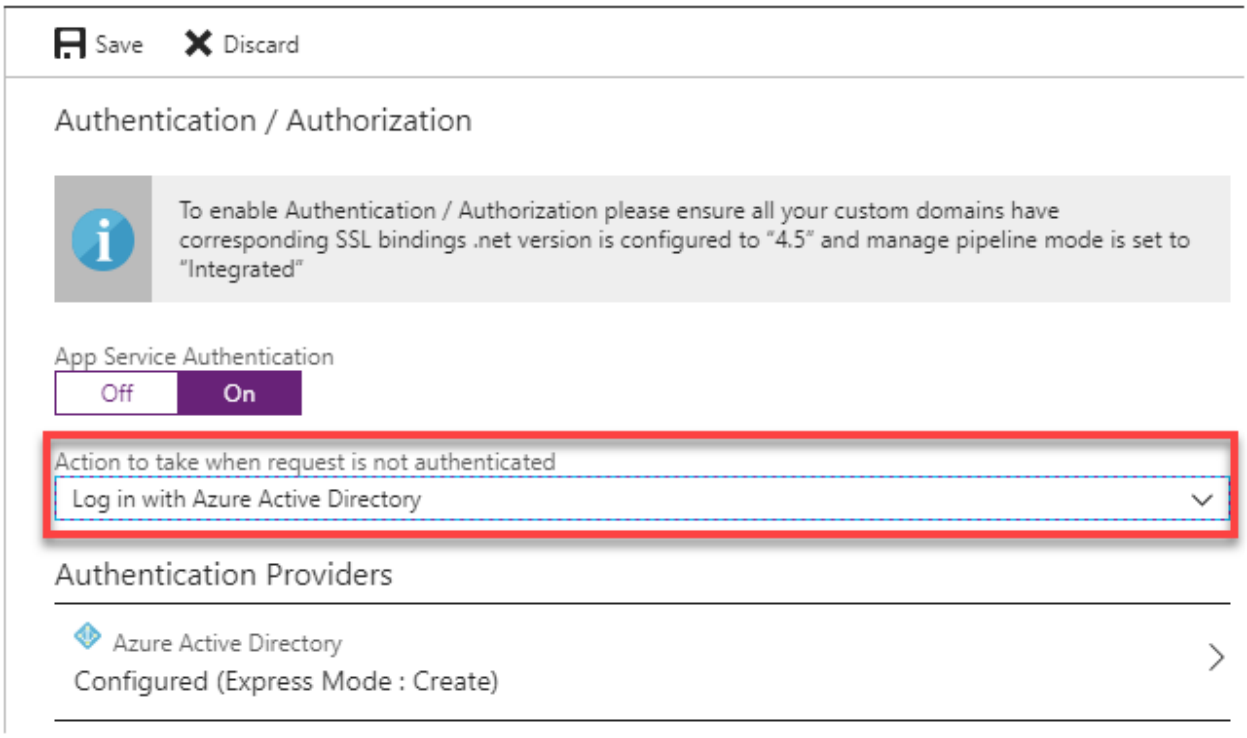
Click the **On** button to see the Authentication Provider list and then click **Azure Active Directory** in the list of providers.



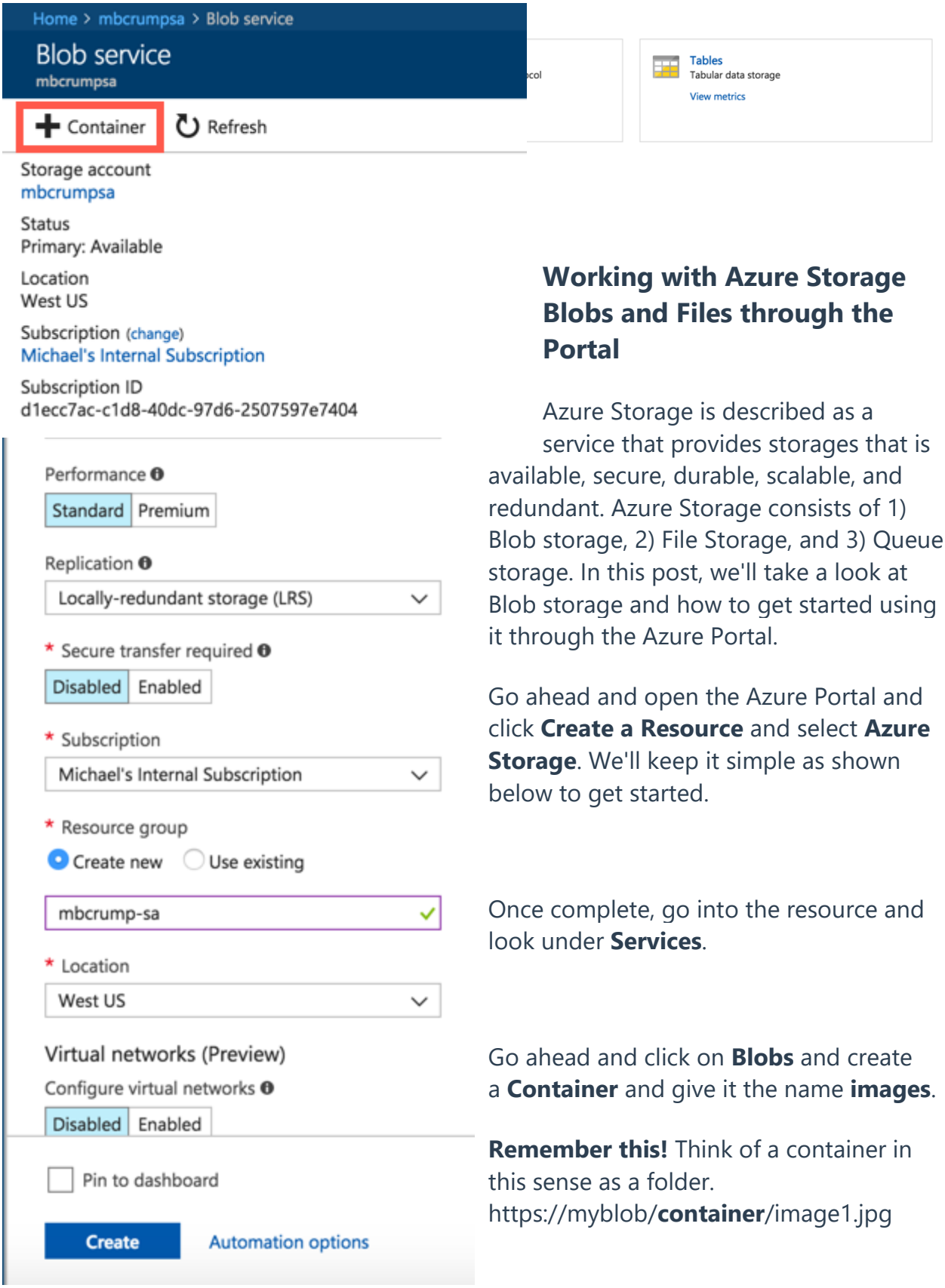
Great. Now click on the **Express** management mode button and click **OK**.


Azure Active Directory Settings
Active Directory Authentication
These settings allow users to sign i with Azure Active Directory. Click here to learn more. Learn more
Management mode
Off
Express
Advanced
Express mode allows user to create an AD Application or select an existing AD application in your current Active Directory.
Current Active Directory
Management mode
Create New AD App
Select Existing AD App
* Create App
mydotnetcoreappfree
Grant Common Data Services Permissions
On
Off
OK

Now you'll need to do one last thing before saving the Authentication / Authorization settings, which is to set the **Action to take when a request is not authenticated**. You'll want to make sure that it is set to **Log in with Azure Active Directory**. This makes sure anyone visiting your site has been authenticated by AAD first. If you are following along and find that you want to use a different AAD tenant (not the Azure account you usually sign into), you can find those steps here: Manually configure Azure Active Directory with advanced settings.



Now you can click the **Save** button to have AAD added as your Authentication Provider.



Working with Azure Storage Blobs and Files through the Portal

Azure Storage is described as a service that provides storages that is available, secure, durable, scalable, and redundant. Azure Storage consists of 1) Blob storage, 2) File Storage, and 3) Queue storage. In this post, we'll take a look at Blob storage and how to get started using it through the Azure Portal.

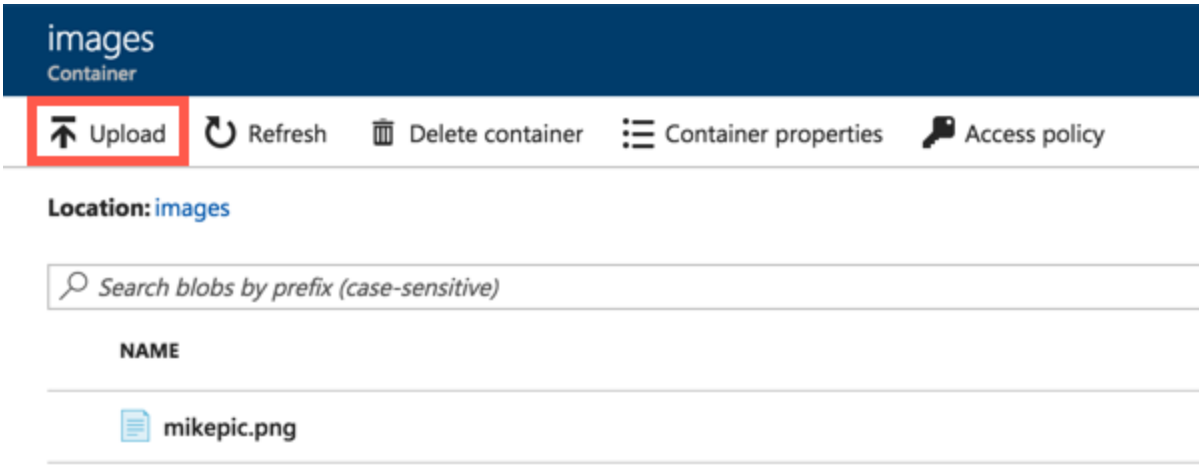
Go ahead and open the Azure Portal and click **Create a Resource** and select **Azure Storage**. We'll keep it simple as shown below to get started.

Once complete, go into the resource and look under **Services**.

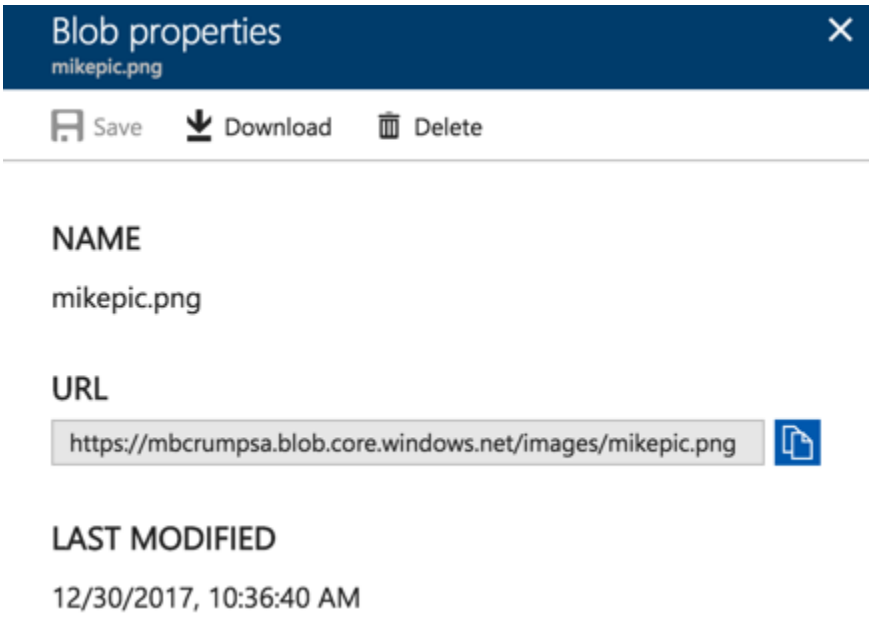
Go ahead and click on **Blobs** and create a **Container** and give it the name **images**.

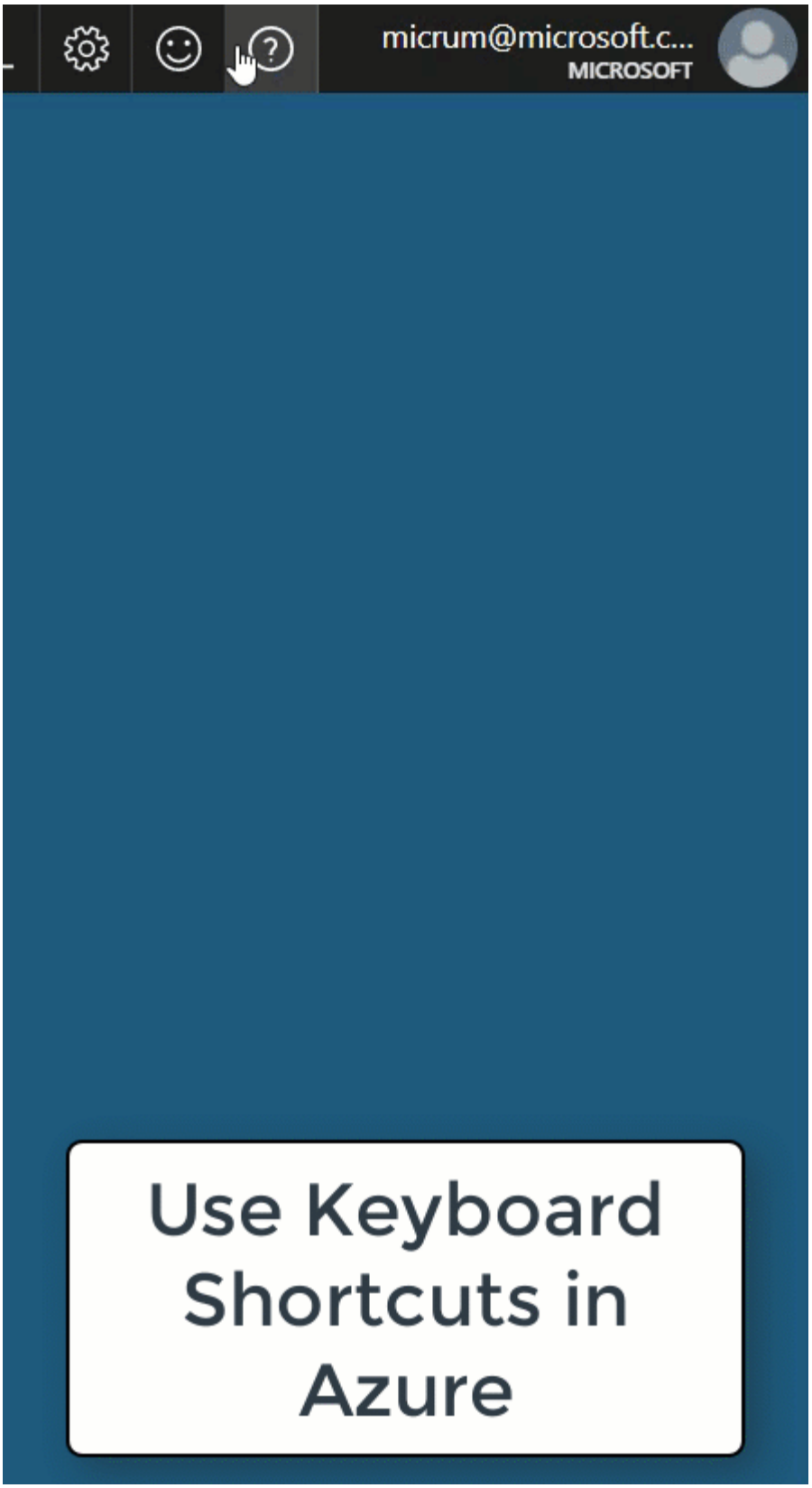
Remember this! Think of a container in this sense as a folder. https://myblob/**container**/image1.jpg

You'll now want to click **Upload** and select a file on your physical disk.



Now that your file is uploaded, select it, and you can click on the ellipsis and select **Blob properties** to see additional details.





Use Keyboard Shortcuts in the Azure Portal

#Azure Portal Keyboard Shortcuts

Developers love keyboard shortcuts and there are plenty keyboard shortcuts in the Azure platform. You can see a list by going to Help and then Keyboard Shortcuts in the portal as shown below.

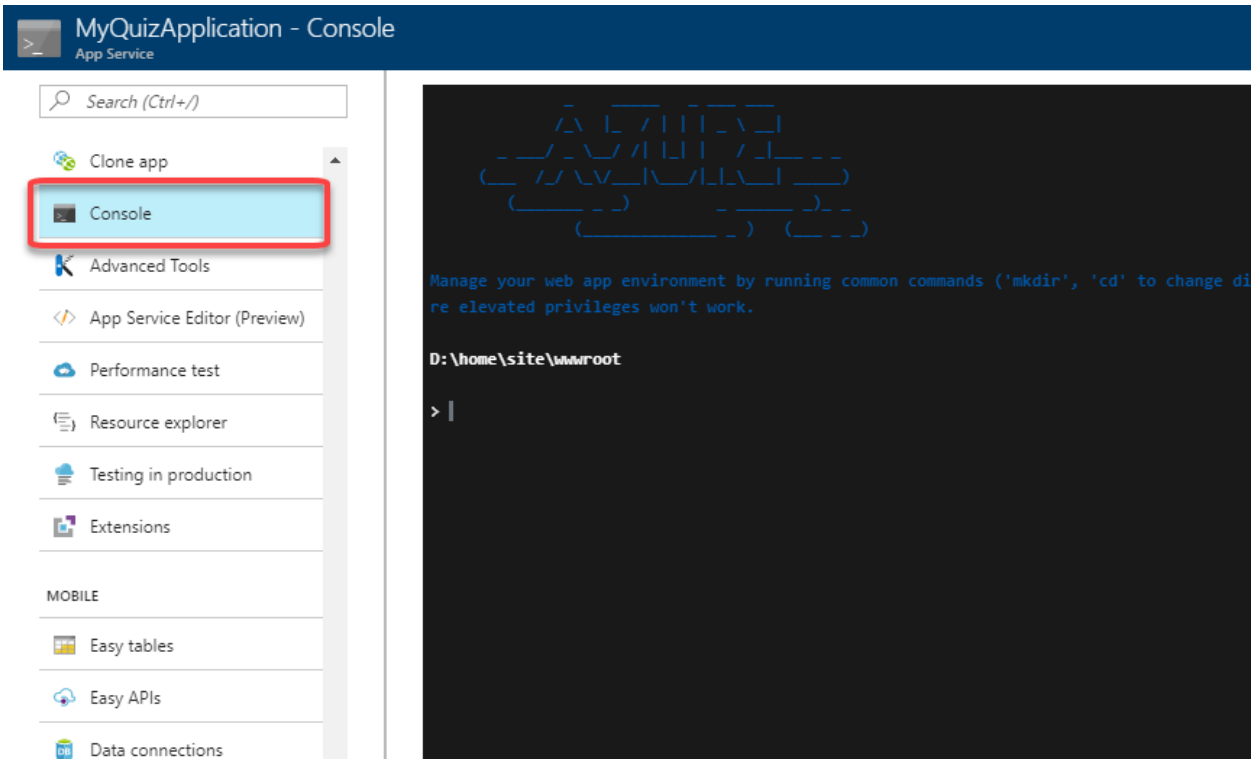
You will see that you have the following keyboard shortcuts available:

Working with Files in Azure App Service

In the Tip 19 - Deploy an Azure Web App using only the CLI, we created a web app and uploaded it to Azure App Service. In this post, we'll take a look at the files uploaded and three tools that I use to work with them

#Console Access to my App Service

I can go to the Azure Portal and select my App Service and click on **Console** under **Development Tools** to have a command prompt to quickly work with my Azure App Service.



As you can tell from the screen-shot, I start in `D:\home\site\wwwroot`. I can type `dir` to see a current directory listing.

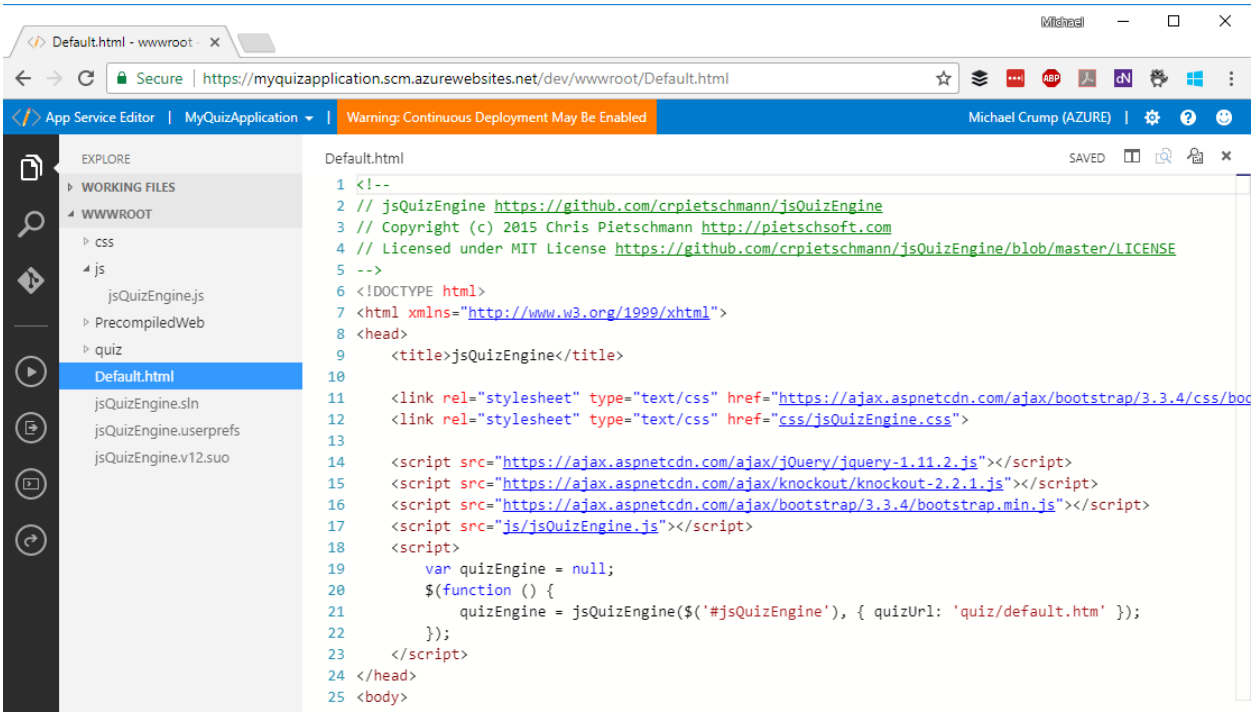
```
Volume in drive D is Windows
 Volume Serial Number is FE33-4717

 Directory of D:\home\site\wwwroot

09/21/2017  08:35 PM    <DIR>          .
09/21/2017  08:35 PM    <DIR>          ..
09/20/2017  09:03 PM    <DIR>          css
09/20/2017  09:03 PM             5,351 Default.html
09/20/2017  09:03 PM    <DIR>          js
09/20/2017  09:03 PM             1,950 jsQuizEngine.sln
09/20/2017  09:03 PM               304 jsQuizEngine.userprefs
09/20/2017  09:03 PM            31,744 jsQuizEngine.v12.suo
09/20/2017  09:03 PM    <DIR>          PrecompiledWeb
09/20/2017  09:03 PM    <DIR>          quiz
               4 File(s)         39,349 bytes
               7 Dir(s)   1,072,893,952 bytes free
```

I can do basic commands here and even use `type <filename>` to parse the output of a file to the screen. You can make directory and so forth, but keep in mind that this is a sandbox environment and some commands which require elevated permissions may not work.

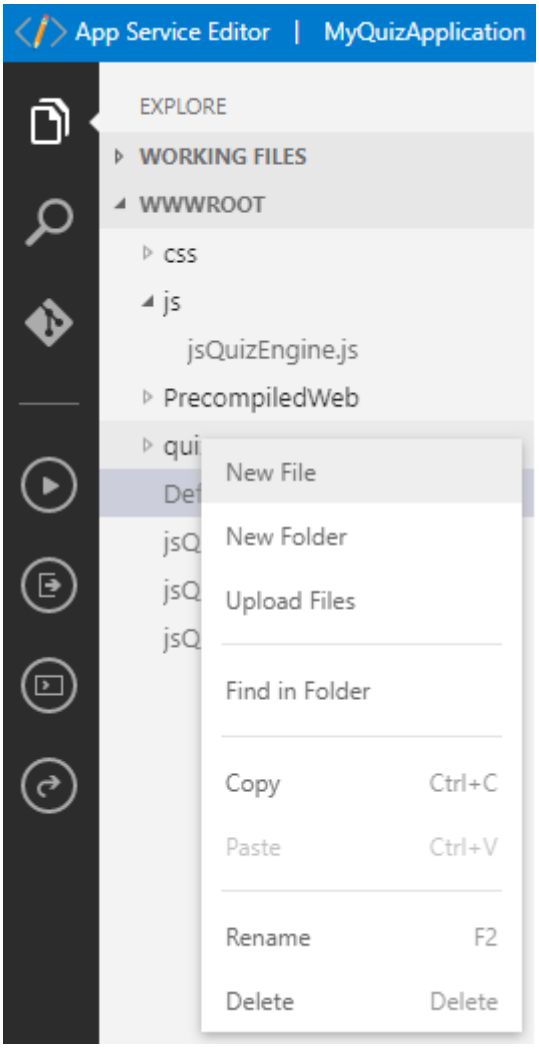
Quick Tip You can type `help` from the console window for a list of available commands.

#A VS Code experience to my App Service

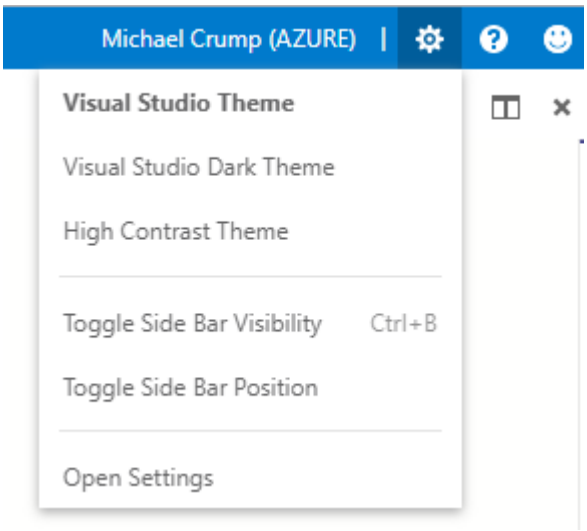
There is also another option that is called "**App Service Editor**" located just two items down from **"Console"** that you picked before.



If you're familiar with VS Code, then you'll be right at home as you can explore, search and add to Git. You can also manipulate files from within the window. This makes it easy to add, edit or delete files.

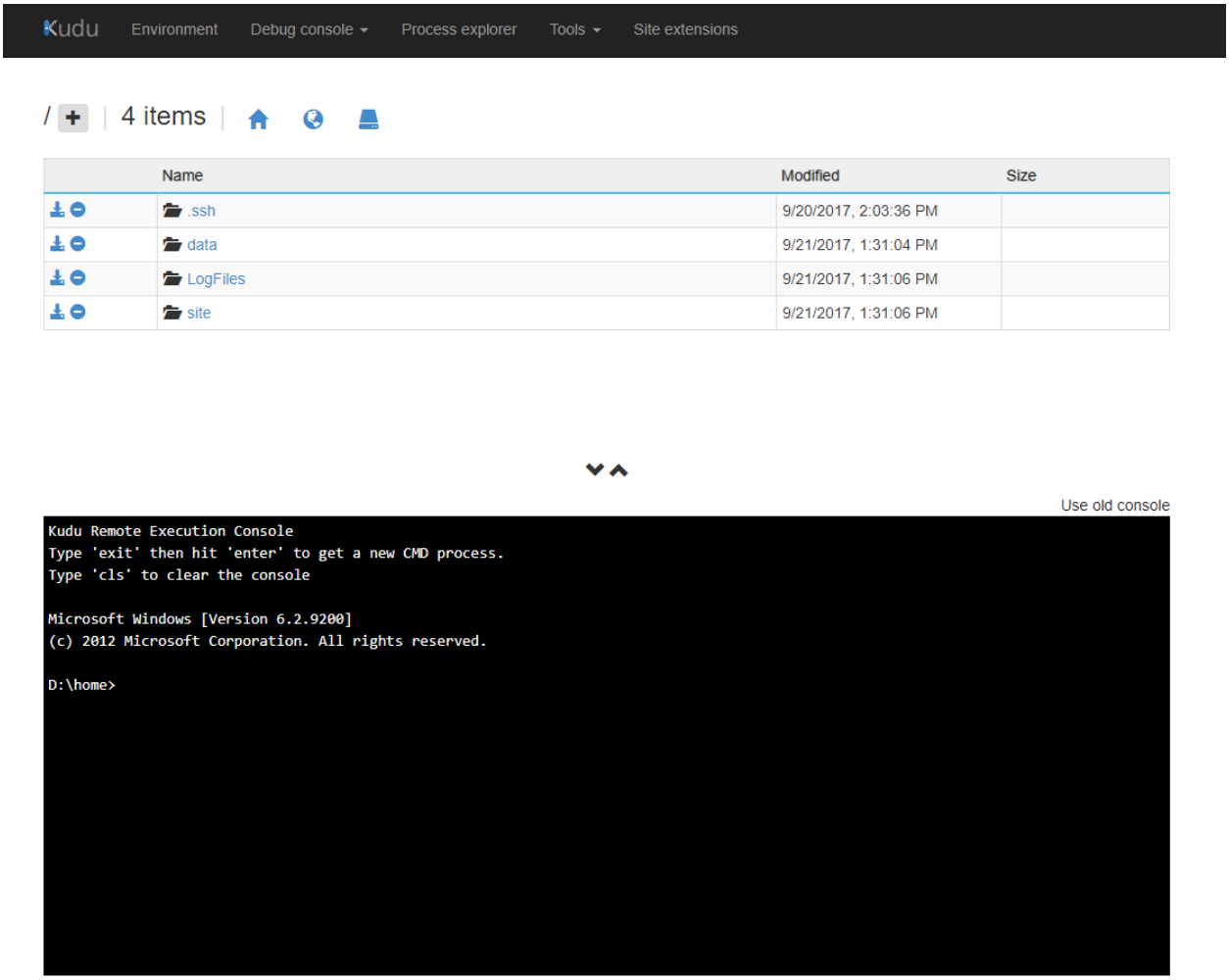


Just like in VS Code, you can modify your settings and even change your theme.



#Kudu Diagnostic Console

No App Service tutorial is complete without mentioning Kudu Diagnostic Console. You can access it from within the **App Service Editor** under **your app name** -> **Open Kudu Console** or through the portal under **Advanced Tools**.



You can just click on the folder name to navigate or type in the command. You can also easily manipulate the files, but I like the App Service Editor better for that functionality. The main reason that I typically come to the Kudu Diagnostic Console is to download files.