

# 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 (BlazorStandAlone 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 ▼  ✓

(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



## BlazorStandAlone

Search (Ctrl+/)



Delete



Endpoints



Overview



Quickstart



Integration assistant (preview)

### Manage



Branding



Authentication

Display name : BlazorStandAlone  
Application (client) ID : 4ab7a47d-a036-41c9-99c5-a91999abd71d  
Directory (tenant) ID :   
Object ID : f9a94e4c-114c-4042-8f45-51baf419f359



Welcome to the new and improved App registrations. Looking to learn how it's change

(App details in the Azure portal)

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

### Implicit grant

Allows an application to request a token directly from the authorization endpoint. Checking Access tokens and ID tokens is recommended only if the application has a single-page architecture (SPA), has no back-end components, does not use the latest version of MSAL.js with auth code flow, or it invokes a web API via JavaScript. ID Token is needed for ASP.NET Core Web Apps. [Learn more about the implicit grant flow](#)

To enable the implicit grant flow, select the tokens you would like to be issued by the authorization endpoint:

- ☒ Access tokens
- ☒ 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

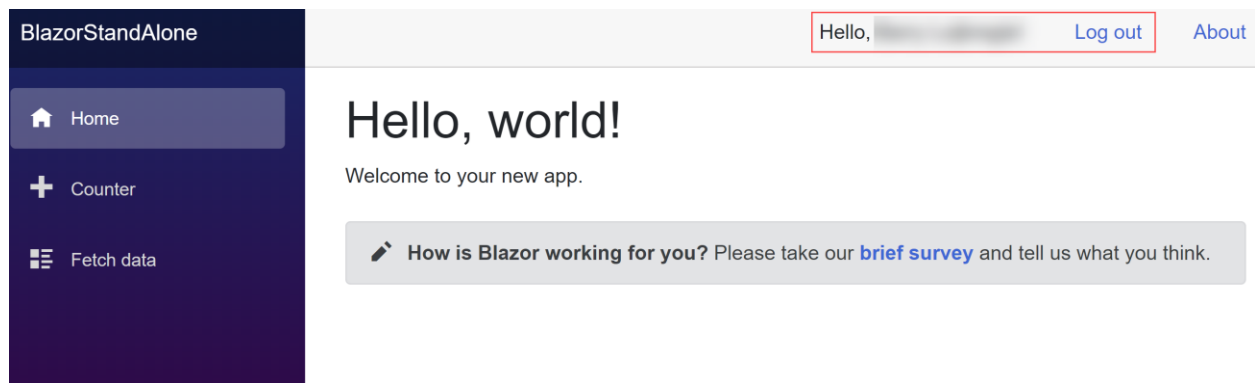
10. The previous command created a folder that contains the Blazor WebAssembly application. **Navigate to the folder** in the command prompt
11. Now run the following command to compile the app:

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

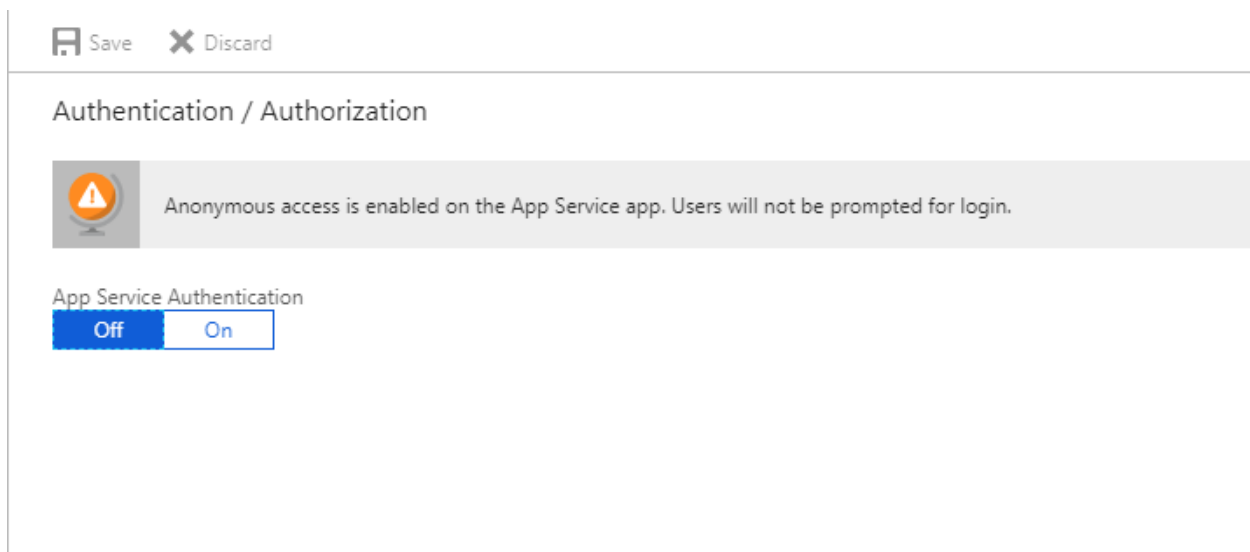
A while ago, I did a post on [Quick and Dirty User Authentication with Azure Web Apps and MVC5 \(opens new window\)](#), where I created a simple web app that used forms authentication. Since then, I've been asked if I could address how to use the **Settings -> Authentication / Authorization** feature 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.

 Save

 Discard

## Authentication / Authorization



Anonymous access is enabled on the App Service app. Users will not be prompted for login.

App Service Authentication


Off


On


Action to take when request is not authenticated


Allow Anonymous requests (no action) ▾


### Authentication Providers

 Azure Active Directory  
Not Configured >

 Facebook  
Not Configured >

 Google  
Not Configured >

 Twitter  
Not Configured >

 Microsoft  
Not Configured >

### Advanced Settings

Token Store 

Off

On

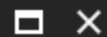
ALLOWED EXTERNAL REDIRECT URLS

...

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 in 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

mydotnetcoreappfree

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](#).

 Save  Discard

---

### Authentication / Authorization

 To enable Authentication / Authorization please ensure all your custom domains have corresponding SSL bindings .net version is configured to "4.5" and manage pipeline mode is set to "Integrated"



App Service Authentication

Off On

Action to take when request is not authenticated

Log in with Azure Active Directory

### Authentication Providers

 Azure Active Directory  
Configured (Express Mode : Create) 

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



[+ Container](#)

[Refresh](#)

Storage account  
[mbcumpsa](#)

Status  
Primary: Available

Location  
West US

Subscription [\(change\)](#)  
[Michael's Internal Subscription](#)

Subscription ID  
d1ecc7ac-c1d8-40dc-97d6-2507597e7404

### Performance ⓘ

[Standard](#) [Premium](#)

### Replication ⓘ

Locally-redundant storage (LRS) ▼

### \* Secure transfer required ⓘ

[Disabled](#) [Enabled](#)

### \* Subscription

Michael's Internal Subscription ▼

### \* Resource group

☒ Create new ☐ Use existing

mbcump-sa ✓

### \* Location

West US ▼

### Virtual networks (Preview)

Configure virtual networks ⓘ

[Disabled](#) [Enabled](#)

☐ Pin to dashboard

[Create](#)

[Automation options](#)



Tables  
Tabular data storage  
[View metrics](#)

## 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.


images  
Container

 Upload


 Refresh

 Delete container


 Container properties

 Access policy

Location: [images](#)


 Search blobs by prefix (case-sensitive)


NAME


 mikepic.png

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

Blob properties  
mikepic.png

 Save

 Download


 Delete

NAME

mikepic.png

URL

https://mbcrumps.blob.core.windows.net/images/mikepic.png



LAST MODIFIED

12/30/2017, 10:36:40 AM



## 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.

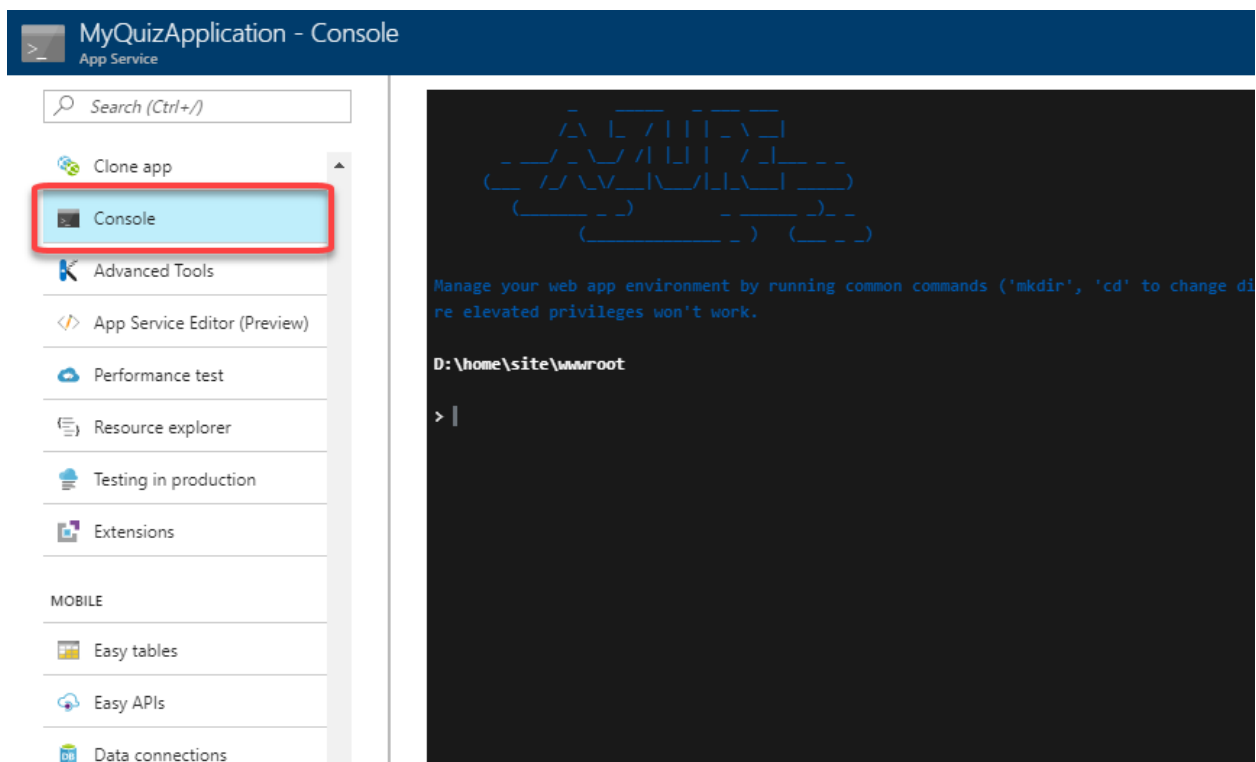
You will see that you have the following keyboard shortcuts available:

## 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.

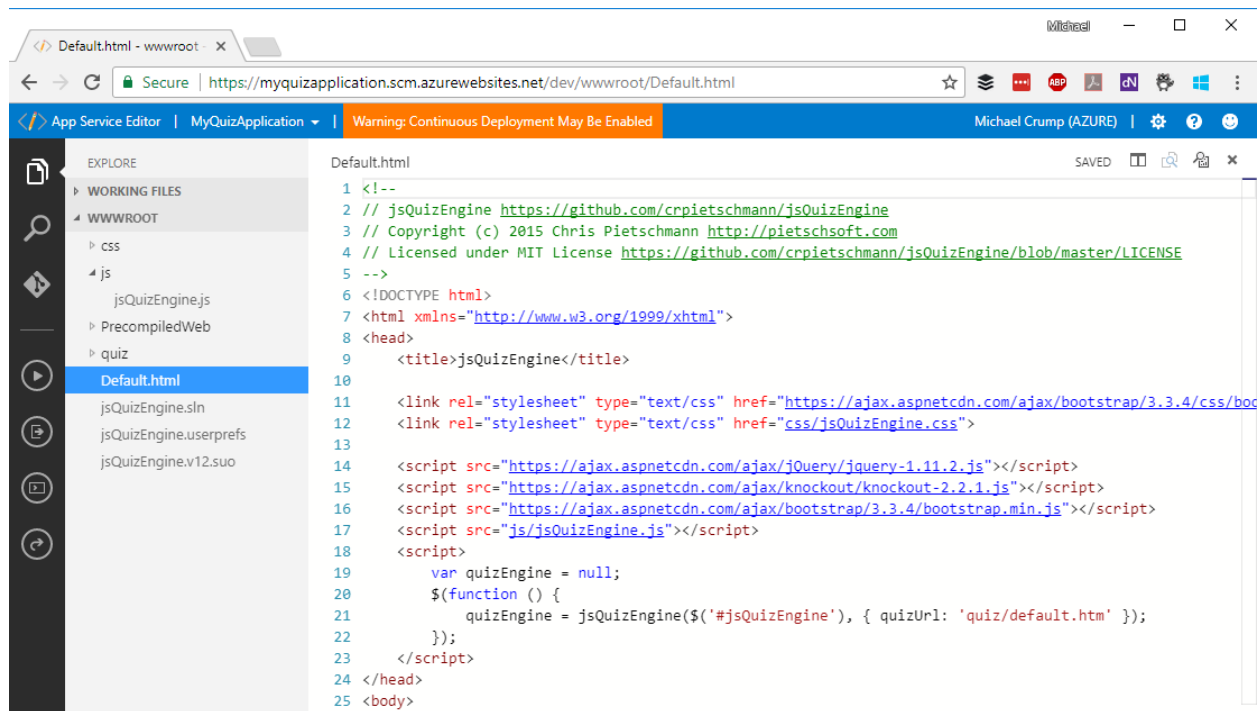
```
1
2 Volume in drive D is Windows
3 Volume Serial Number is FE33-4717
4
5 Directory of D:\home\site\wwwroot
6
7 09/21/2017 08:35 PM <DIR>      .
8 09/21/2017 08:35 PM <DIR>      ..
9 09/20/2017 09:03 PM <DIR>      css
10 09/20/2017 09:03 PM          5,351 Default.html
11 09/20/2017 09:03 PM <DIR>      js
12 09/20/2017 09:03 PM          1,950 jsQuizEngine.sln
13 09/20/2017 09:03 PM           304 jsQuizEngine.userprefs
14 09/20/2017 09:03 PM          31,744 jsQuizEngine.v12.suo
15 09/20/2017 09:03 PM <DIR>      PrecompiledWeb
16 09/20/2017 09:03 PM <DIR>      quiz
17          4 File(s)          39,349 bytes
18          7 Dir(s)  1,072,893,952 bytes free
19
```

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.

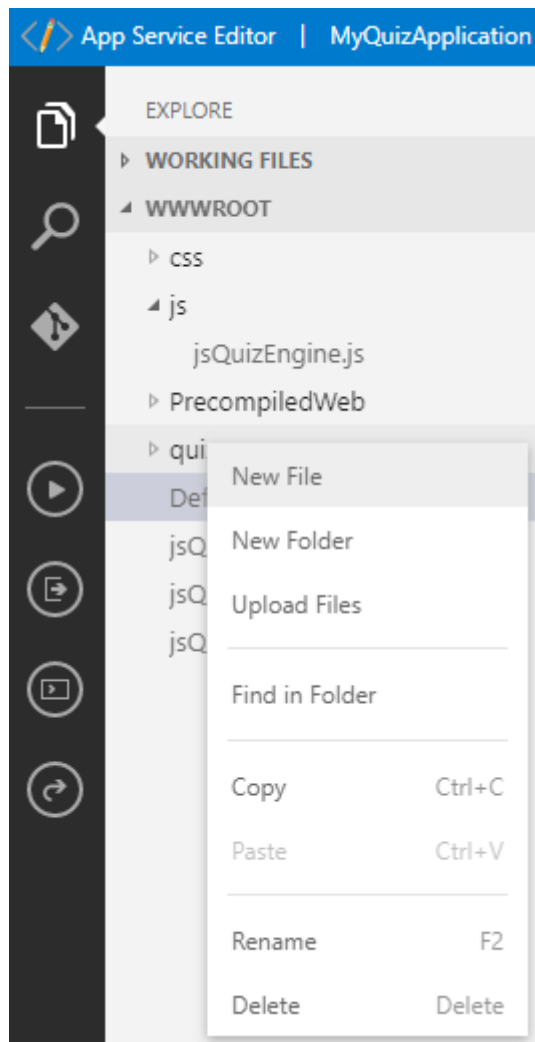
**Quick Tip** You can type `help` from the console window for a list of available commands.

[#A VS Code experience to my App Service](#)

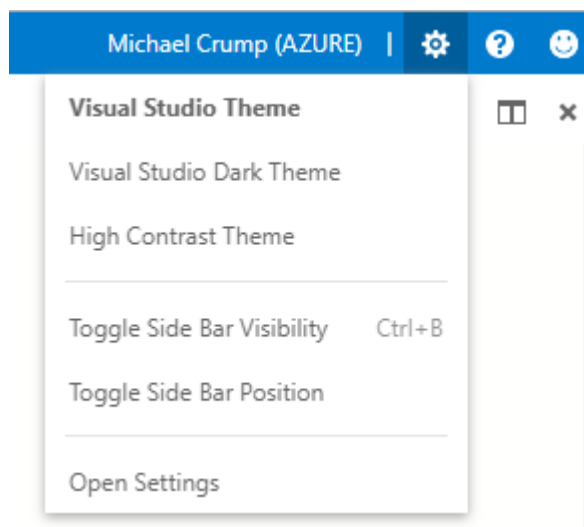
There is also another option that is called "**App Service Editor**" located just two items down from "**Console**" that you picked before.



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**.

The screenshot displays the Kudu Diagnostic Console interface. At the top, there is a navigation bar with the Kudu logo and links to Environment, Debug console, Process explorer, Tools, and Site extensions. Below this, a breadcrumb shows the current location: / + | 4 items. A file explorer table lists the following items:

	Name	Modified	Size
	ssh	9/20/2017, 2:03:36 PM	
	data	9/21/2017, 1:31:04 PM	
	LogFiles	9/21/2017, 1:31:06 PM	
	site	9/21/2017, 1:31:06 PM	

Below the file explorer is a toggle switch labeled "Use old console". The main area is the "Kudu Remote Execution Console", which contains the following text:

```
Kudu Remote Execution Console
Type 'exit' then hit 'enter' to get a new CMD process.
Type 'cls' to clear the console

Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

D:\home>
```

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.