Azure App Service



Vision Scope

Prepared for

Microsoft

24-Jan-16

Version 1.0 Draft

Prepared by

**Ryan Irujo**

Senior Technical Consultant

ryan.irujo@lumagate.com

Contributors

**Add Contributors to Doc Properties**

Table of Contents

[1 Before you Begin 3](#_Toc441443717)

[2 Deploy an Azure Website with a SQL Database 4](#_Toc441443718)

[3 Walkthrough of Web App Features in Azure 12](#_Toc441443719)

[3.1 Configuring Diagnostic and Streaming Logs 12](#_Toc441443720)

[3.2 Configure Authentication / Authorization to your Web App 17](#_Toc441443721)

[3.3 Configure Backups 25](#_Toc441443722)

[4 Azure App Service Support (Preview) 26](#_Toc441443723)

[5 Continous Deployment using GitHub 31](#_Toc441443724)

1. Before you Begin

The objective of this Guide is to expose the participants to how they can implement Azure Service Apps in their existing Azure Subscription deployment model. At the time of this writing the current requirements to get started are below.

* Admin access to an Azure Subscription (minimum Trial Subscription)
* Visual Studio 2015 Installed (minimum Community Edition)
* Microsoft Azure PowerShell 1.0.1 (November 2015)
* Azure SDK 2.8.1 or higher

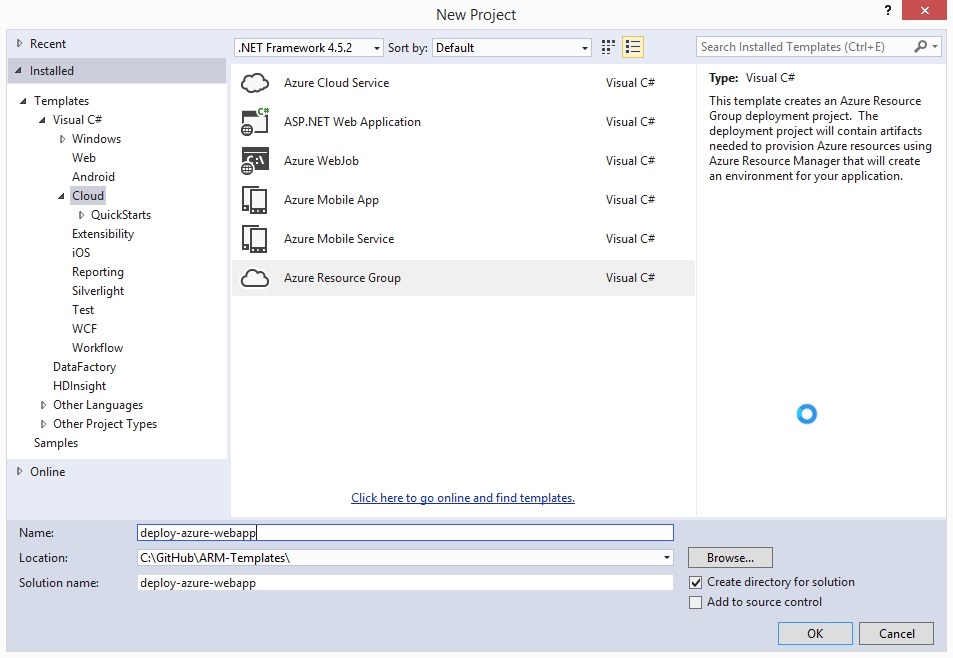
Finally, all Scripts, Templates and Applications mentioned within this Guide can be found in Lumagate’s GitHub Repository. If you require access, please contact a member of the Elite Incubation Team at Lumagate.

1. Deploy an Azure Website with a SQL Database

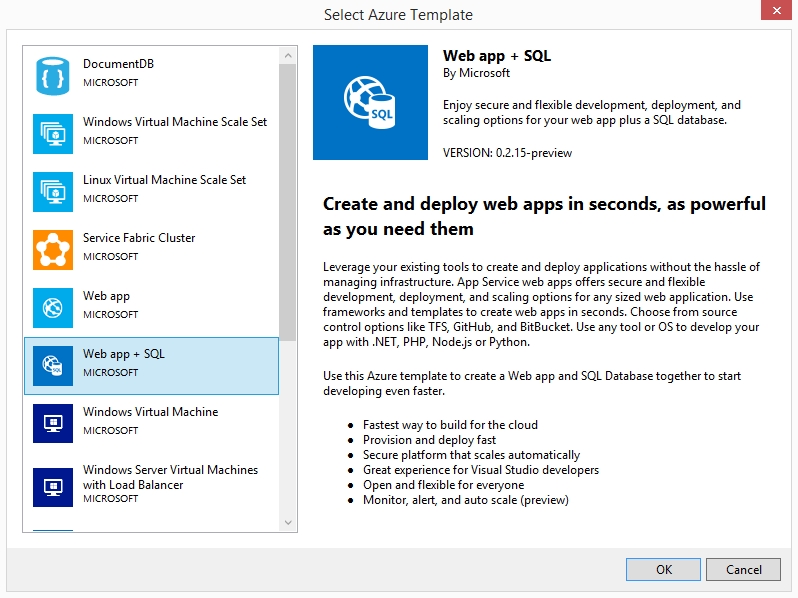
In this section, you will be deploying a Simple Web Application with a SQL Database attached using an Azure Resource Manager (ARM) Template.

Launch Visual Studio 2015 and click on **File 🡪 New 🡪 Project…**

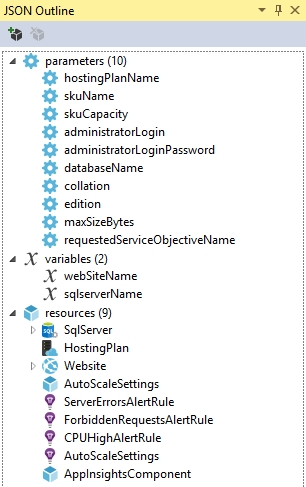
Next, in the New Project window, click on **Templates 🡪 Cloud 🡪 Azure Resource Group**. Change the Name and Solution Name fields to **deploy-azure-webapp** as shown below.



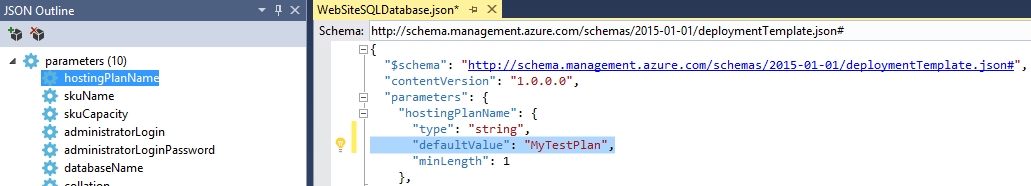
Next, look for the **Web app + SQL** Template and then click on OK.



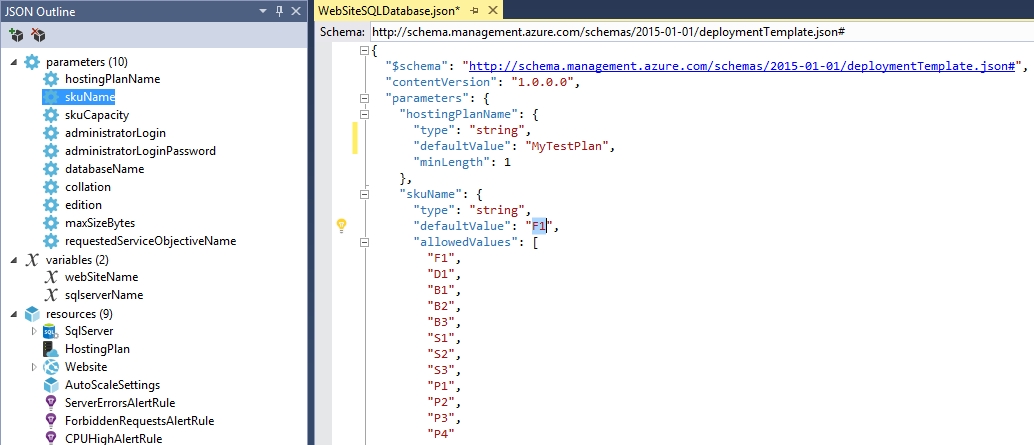
Once the Template has finished loading, expand out the **parameters**, **variables** and **resources** under the **JSON Outline**. Next, we are going to go over some of the options available to you when deploying an Azure Web App from an ARM Template and set some of their values before deploying the template.



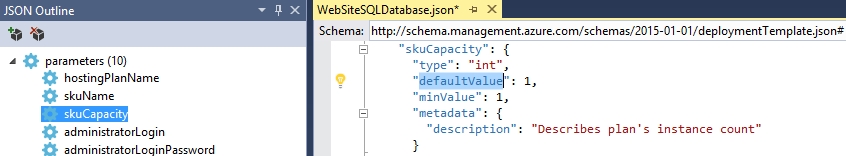
First, click on **hostingPlanName** and give it a default value, such as MyTestPlan, as shown in the image below.



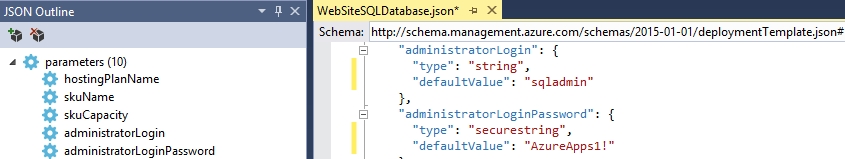
Next, click on **skuName**, change the defaultValue to S1. This will ensure that we have access to such capabilities.



Next, click on **skuCapacity** and change the defaultValue from 1 to 4.



Next, set the **administratorLogin** and **administratorPassword** parameters as shown in the image below. Use whatever defaultValues you want to here.



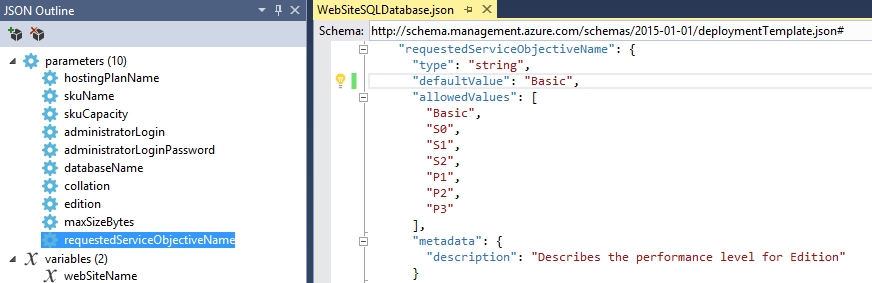
Next, set the **databaseName** parameter to **mytestsqldb**.



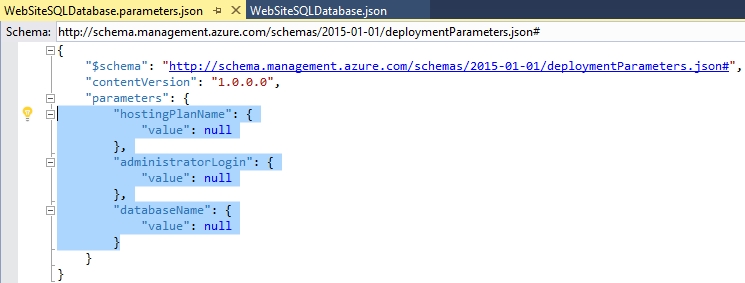
Click on the **edition** parameter, and change the defaultValue from **Basic** to **Standard**. By upgrading the value to Standard, you gain access to capabilities.



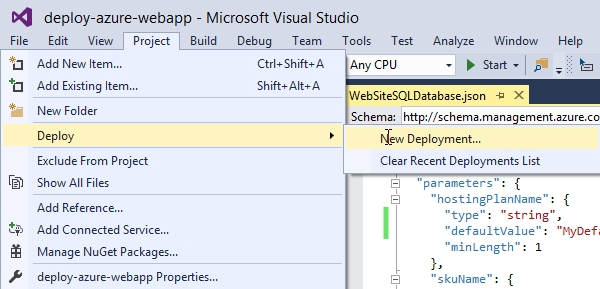
Next, click on the **requestedServiceObjectiveName** and change it from **Basic** to **S1**



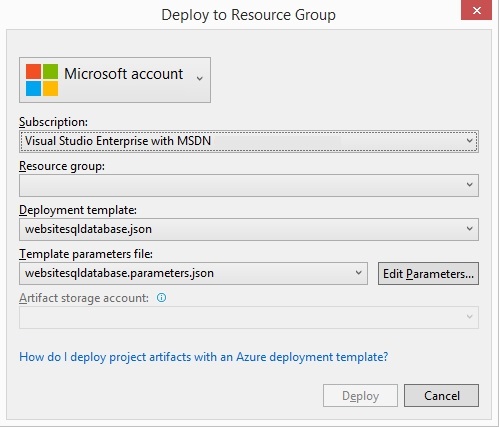
Next, open up the **WebsiteSQLDatabase.parameters.json** file and delete all the parameter entries highlighted below.



Next, click on **Project 🡪Deploy 🡪 New Deployment…**



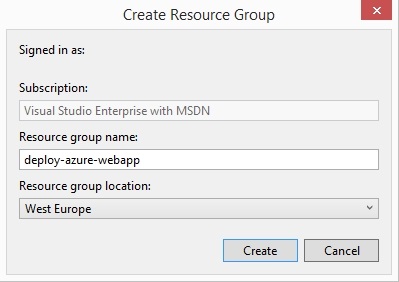
Put in your Microsoft Credentials if required and then click on the drop-down menu under **Subscription:** and choose which subscription you wish to deploy to.



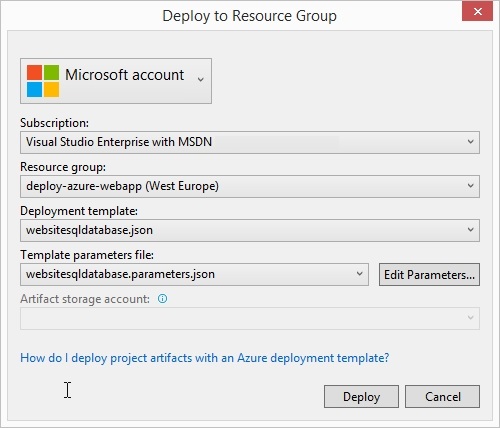
Next, click on the Resource Group drop-down Menu and click on Create New…



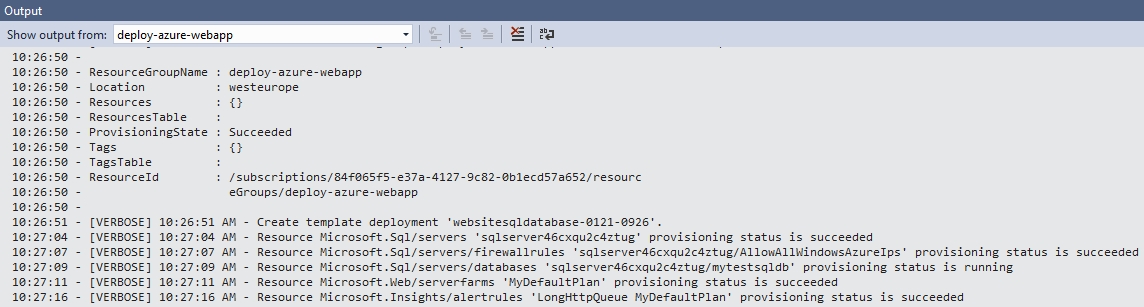
Take the default values for the Resource group name and choose a location to deploy to. When you are finished click Create.



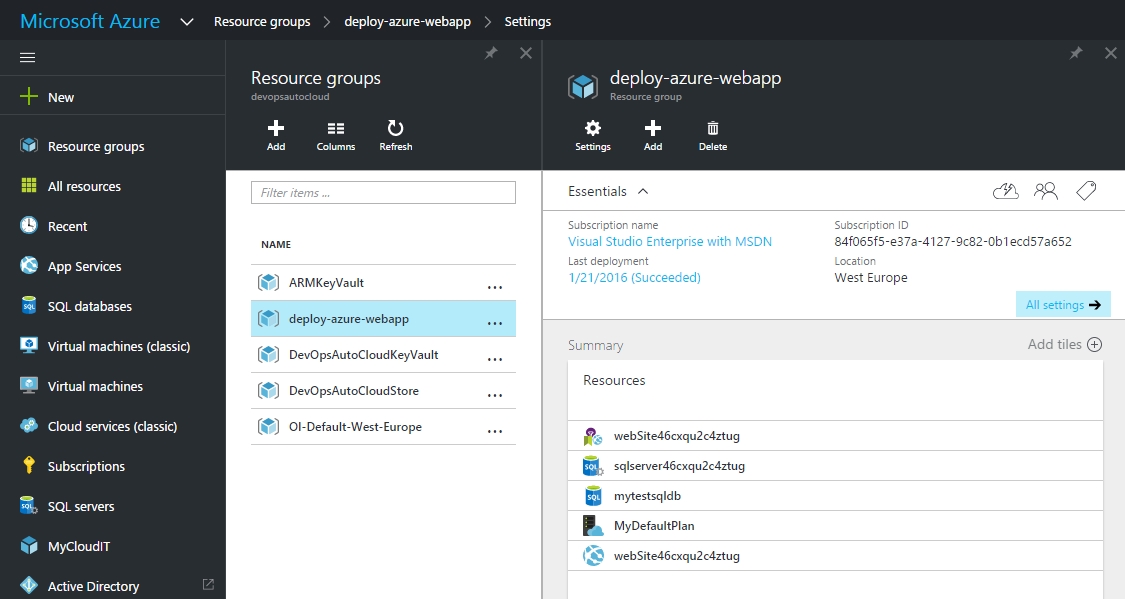
Lastly, you can double-check your parameter values and make changes to them if you so desire; afterwards click Deploy.



The Output window will appear allowing to follow the progress of the deployment.



When the deployment is complete, you can explore your deployed Web Application inside the Azure Portal.



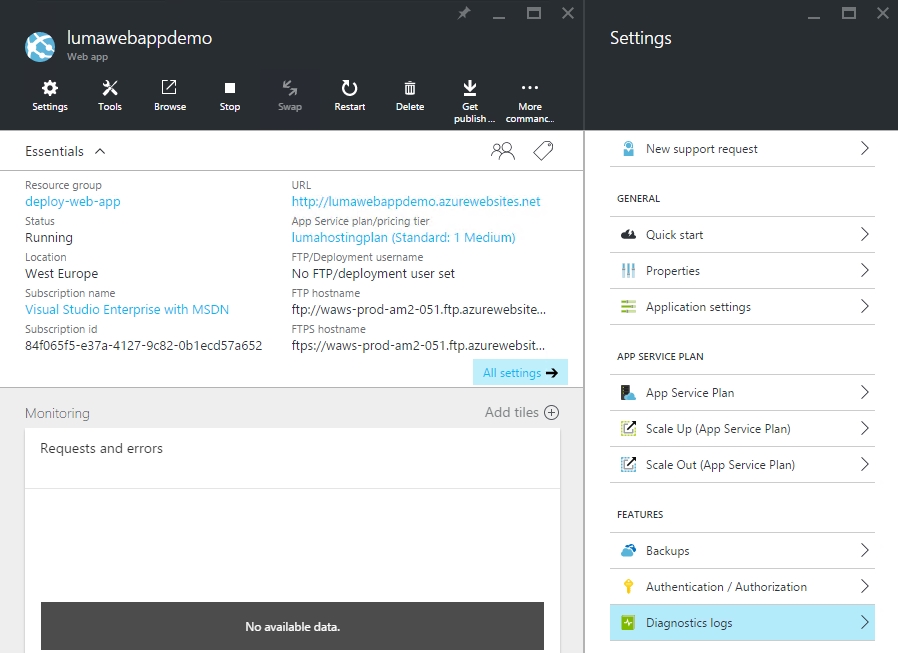
1. Walkthrough of Web App Features in Azure

In this section, we will go over some of the features you may want to take advantage of using the Web App Service. Before starting this section, make sure you have already deployed Azure Web Application using the **deploy-web-app** ARM Template included with this material using Visual Studio. Also, make sure you have access to the Azure Portal at <https://portal.azure.com>.

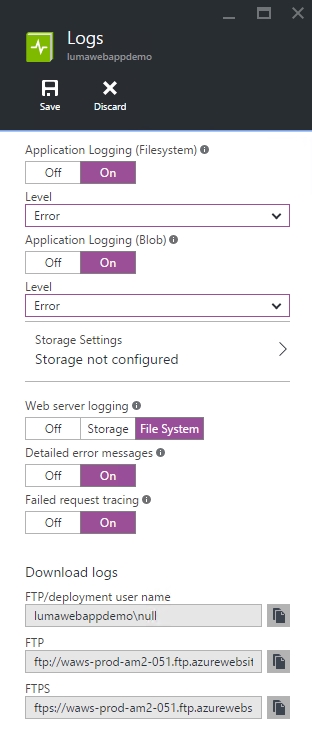
Finally, if you are deploying the deploy-web-app ARM Template in a classroom environment, make sure to append the names of the following parameter values with letters or numbers before attempting your deployment:

* hostingPlanName
* sqlServerName
* databaseName
* websiteName
* diagStorageName
* backupStorageName
  1. Configuring Diagnostic and Streaming Logs

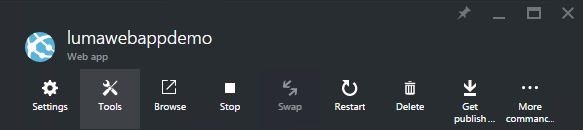
From the Resource Group where you deployed the **deploy-web-app** ARM Template, click on **lumawebappdemo 🡪 All Settings 🡪Features 🡪 Diagnostic Logs**



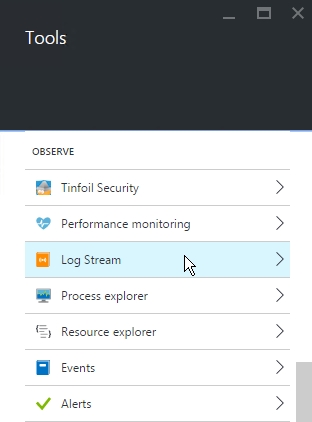
Next, turn on all of the features as shown below and make sure Web server logging is set to FileSystem. Once you are finished, click on the Save button. Within a minute, the settings will be updated and in place.



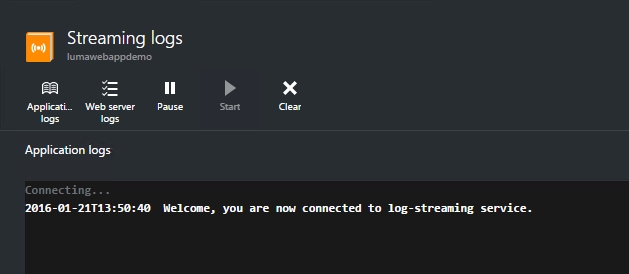
Next, go back to the lumawebappdemo Web app and click on Tools.



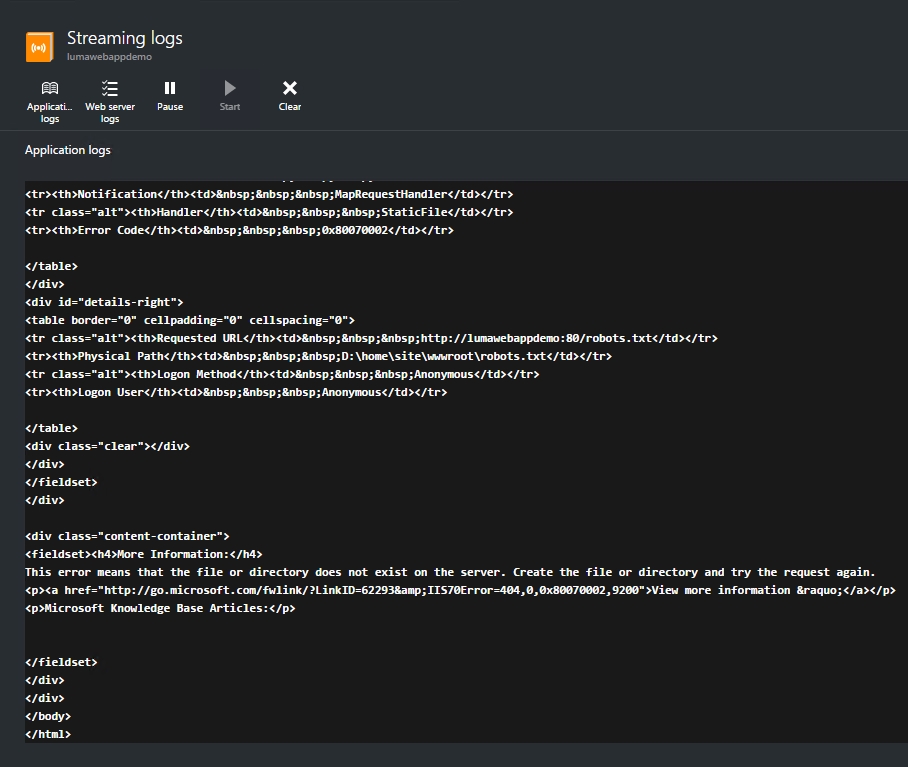
Next, click on Log Stream.



By default, the Log Stream will open up the Application Logs of the Web App.



Next, in a separate web browser (preferably a different monitor, go the following URL:  
<http://lumawebappdemo.azurewebsites.net/robots.txt>. You should see the following log information appear in the Streaming logs window stating that the robots.txt doesn’t exist.



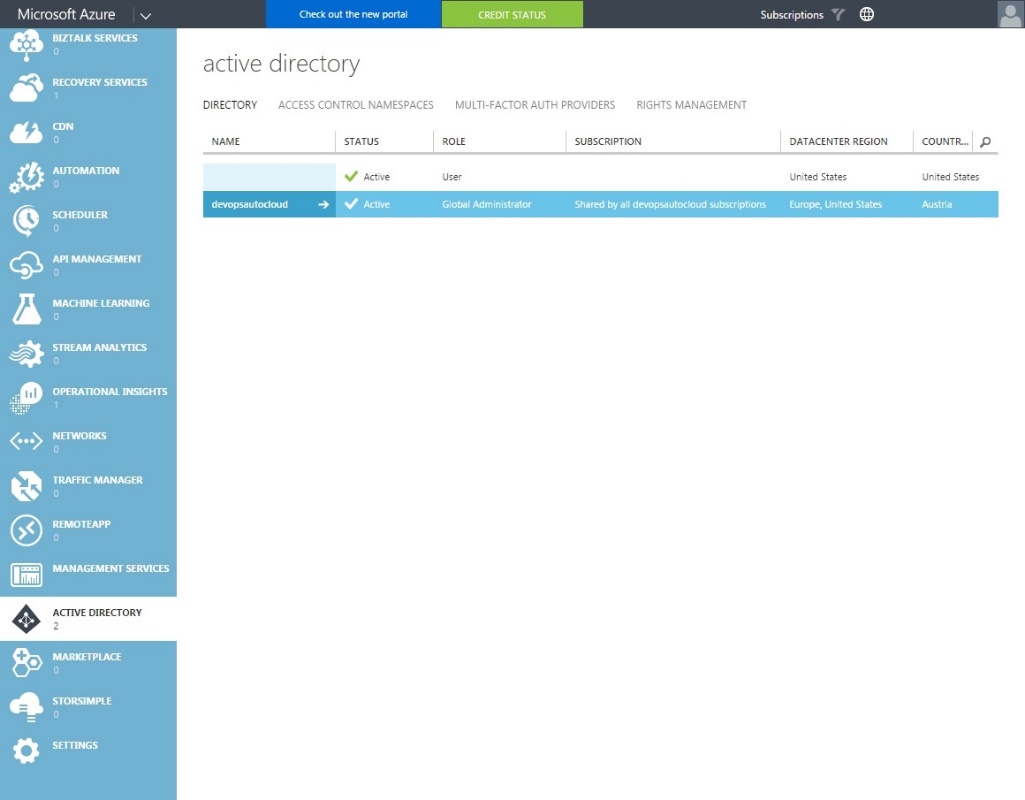
Feel free to explore streaming Web server logs as you see necessary.

* 1. Configure Authentication / Authorization to your Web App

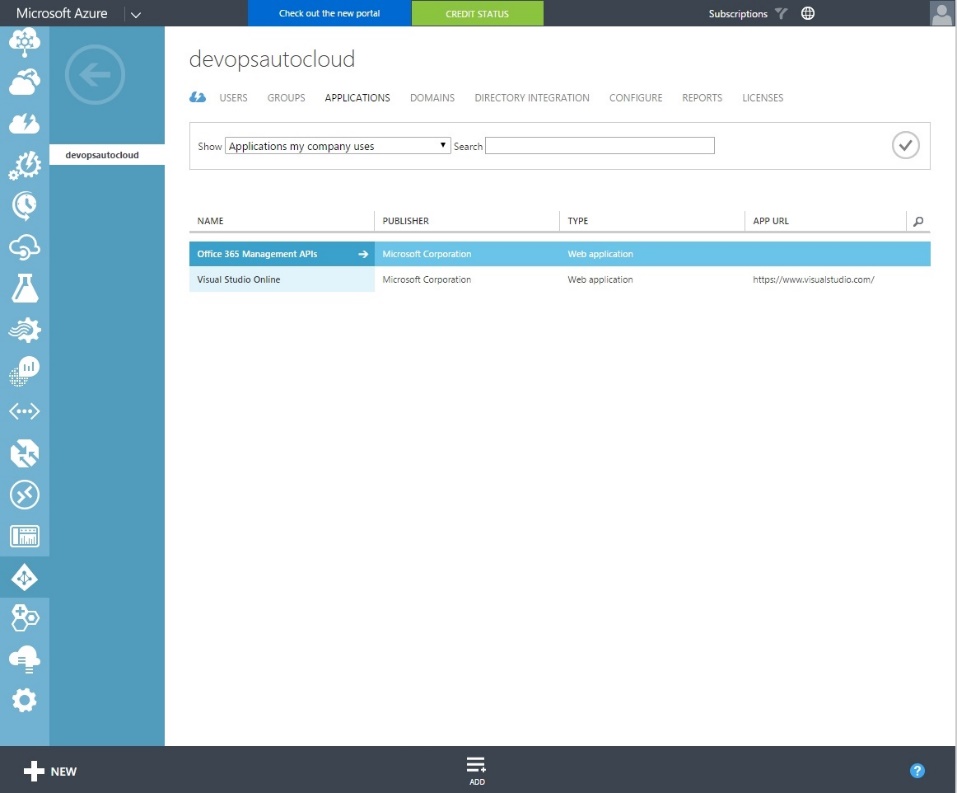
For this particular sub-module, you will be configuring authentication using Azure Active Directory. Before you begin, make sure you have the URL of the deployed Web Application from the **deploy-web-app** ARM template as you will need it later. By Default, the starting URL is <http://lumawebappdemo.azurewebsites.net>

First, login to the **Classic Azure Portal** at <https://manage.windowsazure.com>.

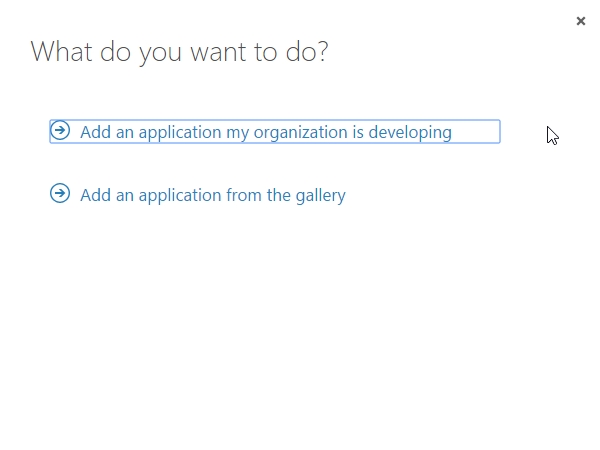
Click on **Active Directory** and the name of the Directory you are going to associate the Web App with.



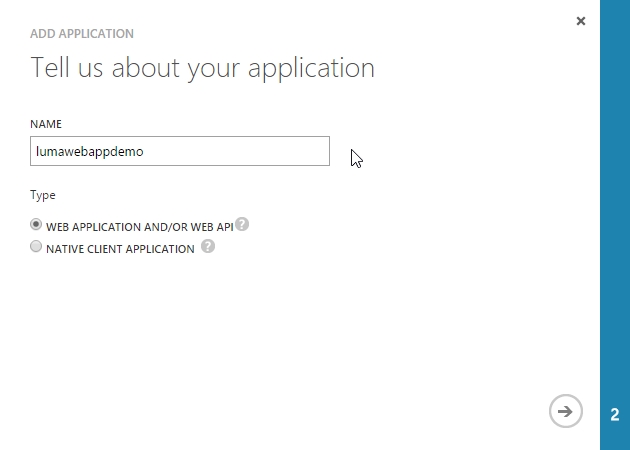
Next, click on **Applications** and then click on the **ADD** button at the bottom of the page



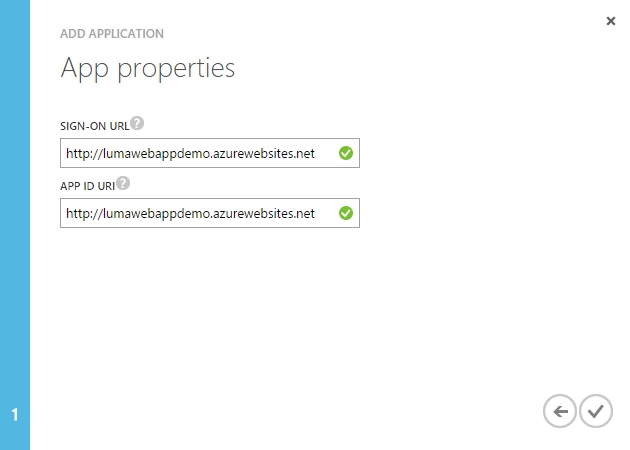
Next, click on **Add an application my organization is developing**.



Next, type in the **NAME** of the Web Application (lumawebappdemo by default) and choose the option **WEB APPLICATION AND/OR WEB API**. Afterwards, click on the Arrow button.



Next, copy and paste the URL of the Web App into the **SIGN-ON URL** and **APP ID URI** fields and then click on the Checkmark Button.



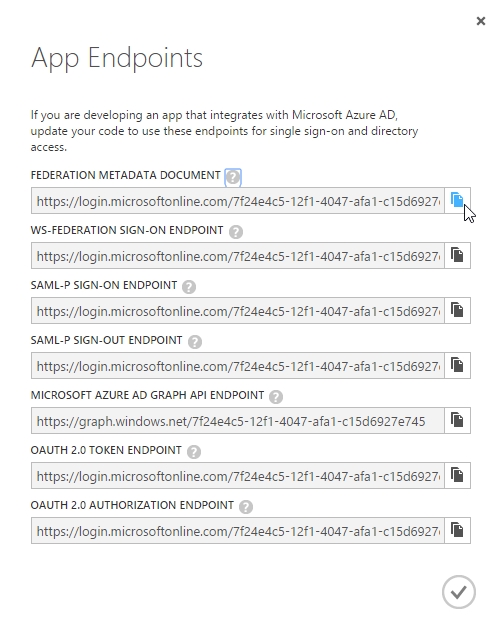
Once the new app has been added, click on the **CONFIGURE** tab under the Application and scroll down to the **single sign-on** section and change the **REPLY URL** to the following:

<https://lumawebappdemo.azurewebsites.net/.auth/login/aad/callback>

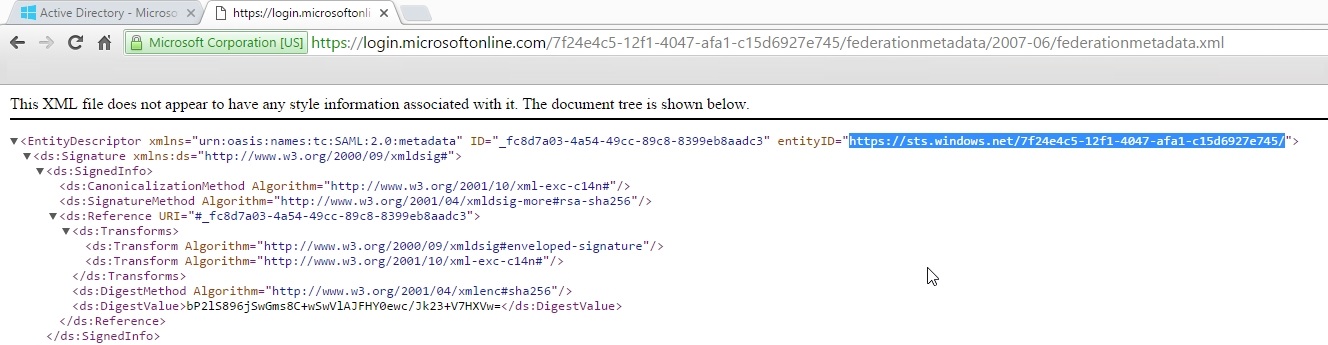
Make sure to click on the **SAVE** button at the bottom of the screen and copy the **CLIENT ID** for later use.



Next, at the bottom of the page, click on **VIEW ENDPOINTS** and then copy the **Federation Metadata Document URL** under App Endpoints.



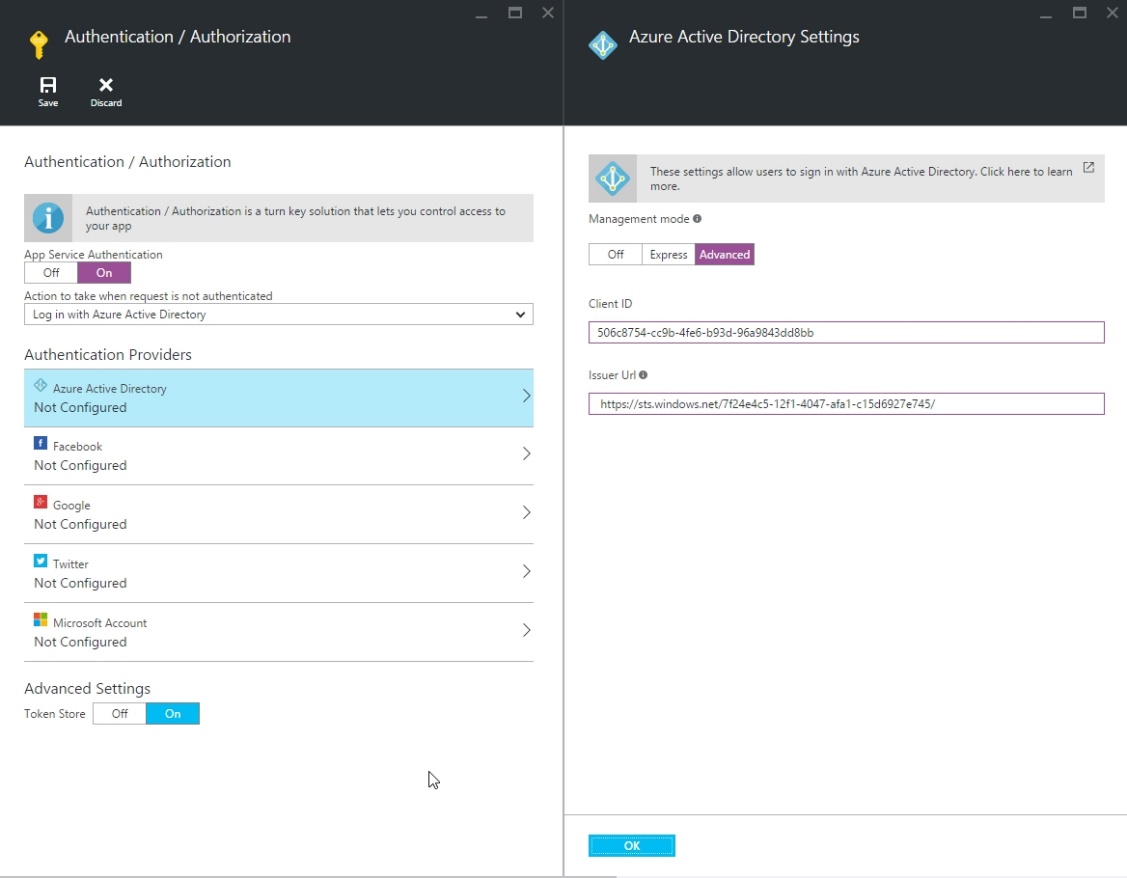
Next, browse to the **Federation Metadata Document URL** in a new Browser Tab and copy the **entityID** URL for your application as shown in the screenshot below.



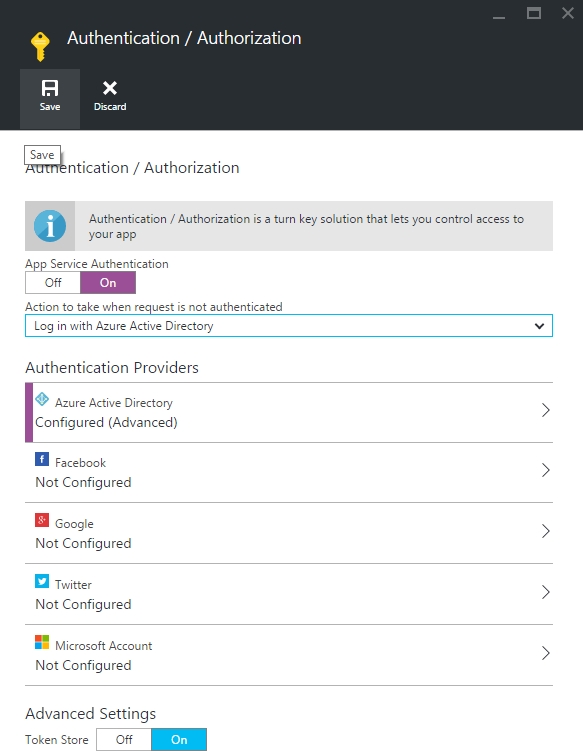
Next, browse back to the **Azure Portal** in a new tab and from the Resource Group where you deployed the **deploy-web-app** ARM Template, click on **lumawebappdemo 🡪 All Settings 🡪Features 🡪 Authentication / Authorization** and turnthe **App Service Authentication** to **On**.



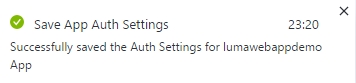
Next, click under Authentication Providers click on **Azure Active Directory** and under Azure Active Directory Settings, change the Management Mode to **Advanced**. Paste in the **Client ID** into the Client ID field and the entityID into the **Issuer Url** field. Afterwards click on **OK**.



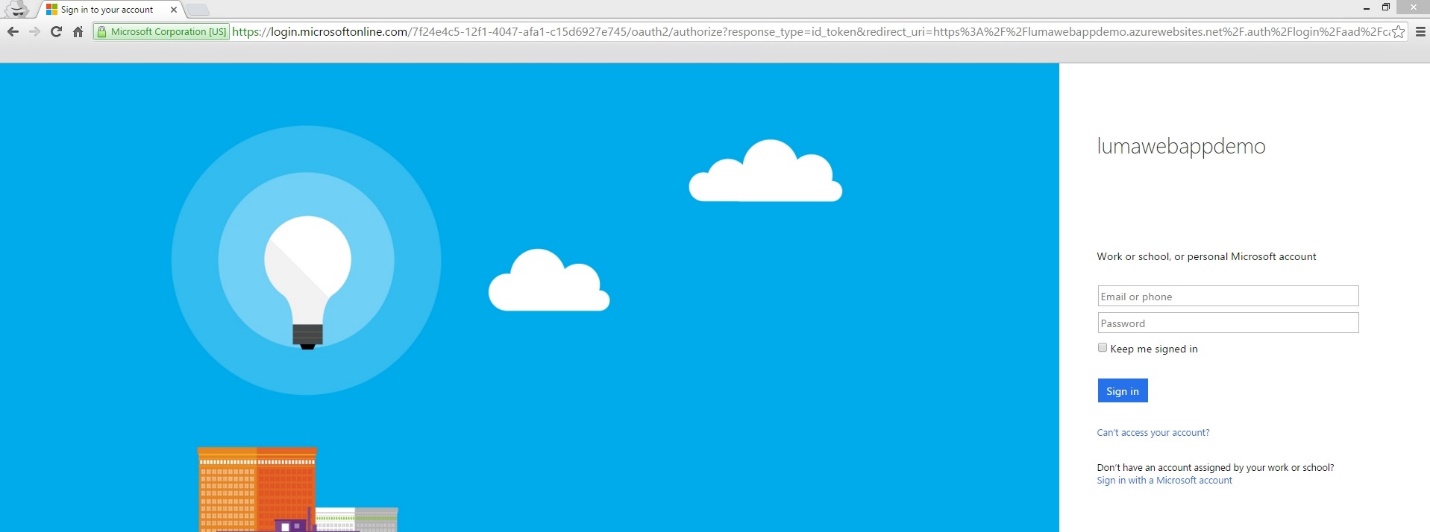
Under the **Authentication / Authorization** section, click on the **Save** button.



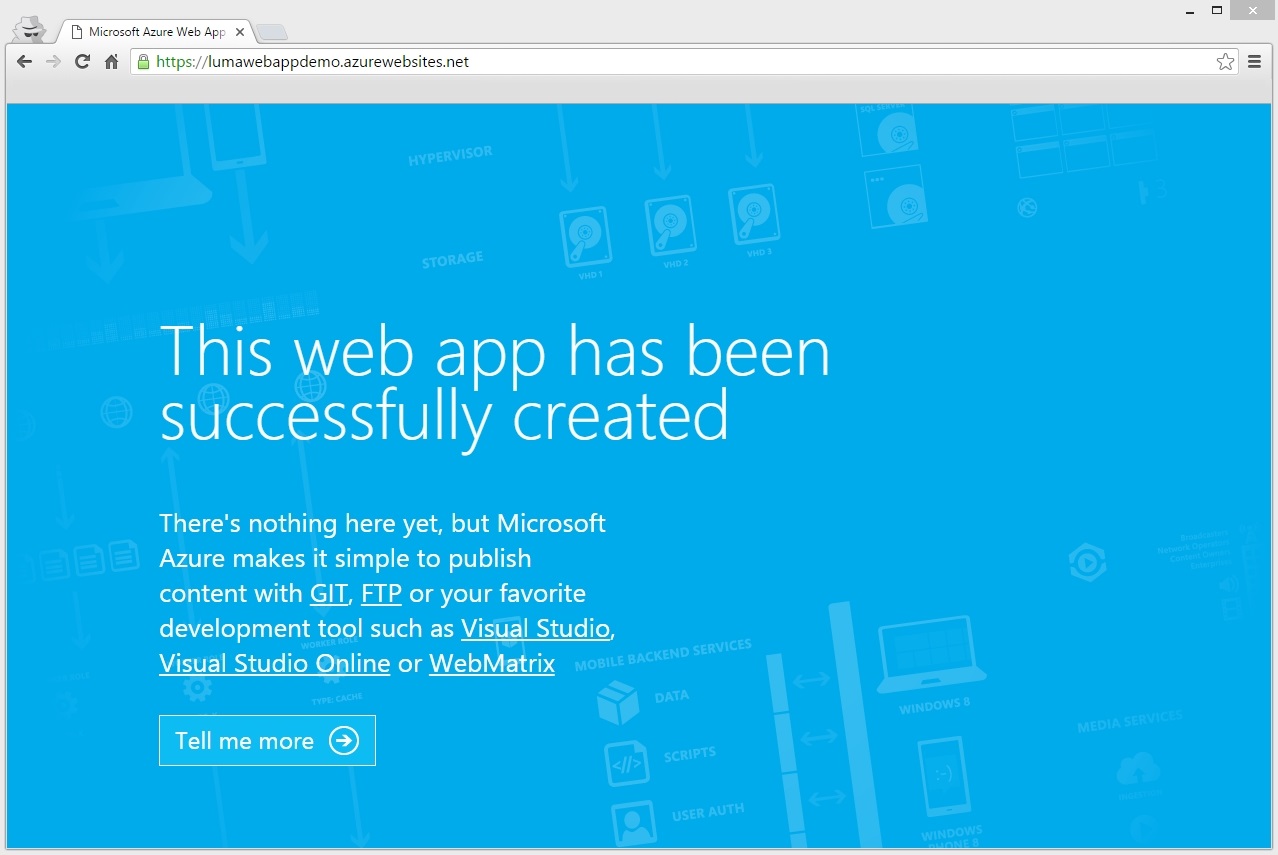
You should get back the following response.



Next, open up an incognito web browser tab and browse to the Web Apps URL (by default <http://lumawebappdemo.azurewebsites.net>). If you receive an Internal Error has occurred message, just refresh the page again. You should see a login page for **lumawebappdemo**.



Proceed to login using your Tenant Credentials; you should see the default web page of the Web App once you are logged in.



* 1. Configure Backups

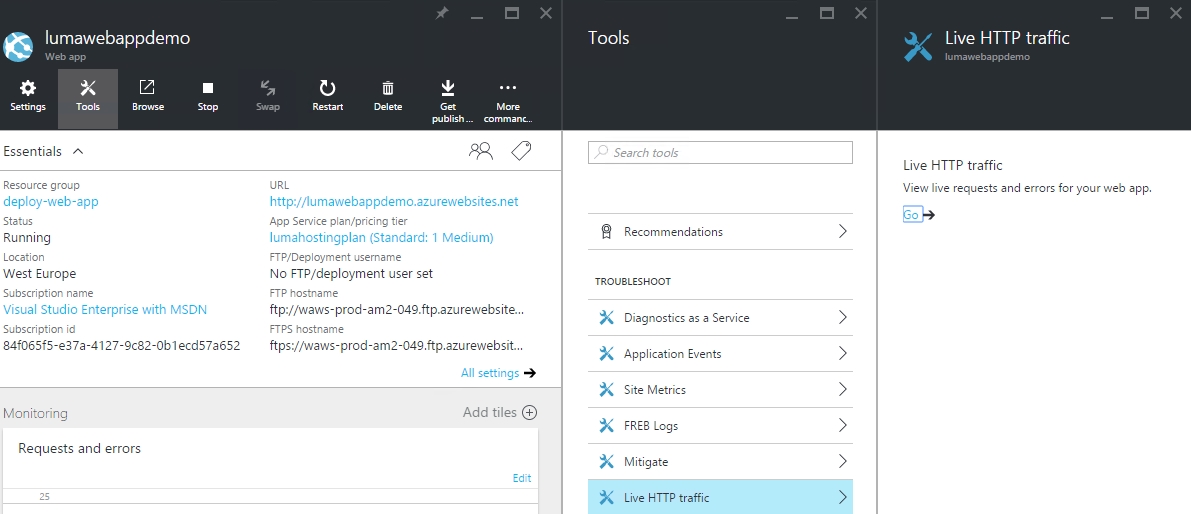
In this sub section, we will be going over how to configure Backups for the Web App.

1. Azure App Service Support (Preview)

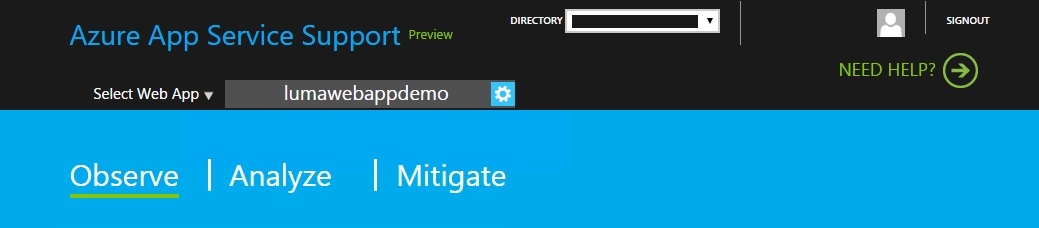
In this section, we will be going over how you can observe web traffic, check logs and mitigate downtime on your Azure Web App using the Azure App Service Support feature.

Start off by logging into the Azure Portal (<https://portal.azure.com>) and select the Subscription that you recently deployed the **deploy-web-app** ARM Template to.

From the Resource Group where you deployed the **deploy-web-app** ARM Template, click on **lumawebappdemo 🡪 All Settings 🡪 Tools 🡪 Troubleshoot 🡪 Live HTTP** Traffic and click on the **Go🡪** link.

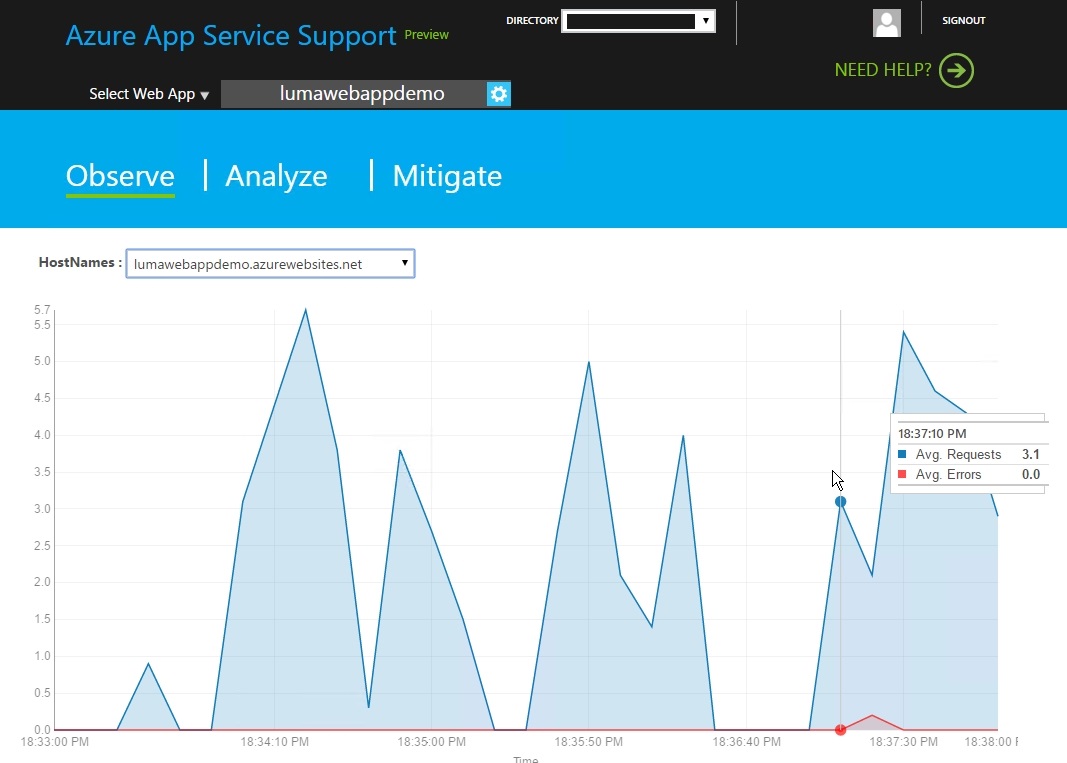


A new web brower tab will open up and you shoul see the **Azure App Service Support** page.

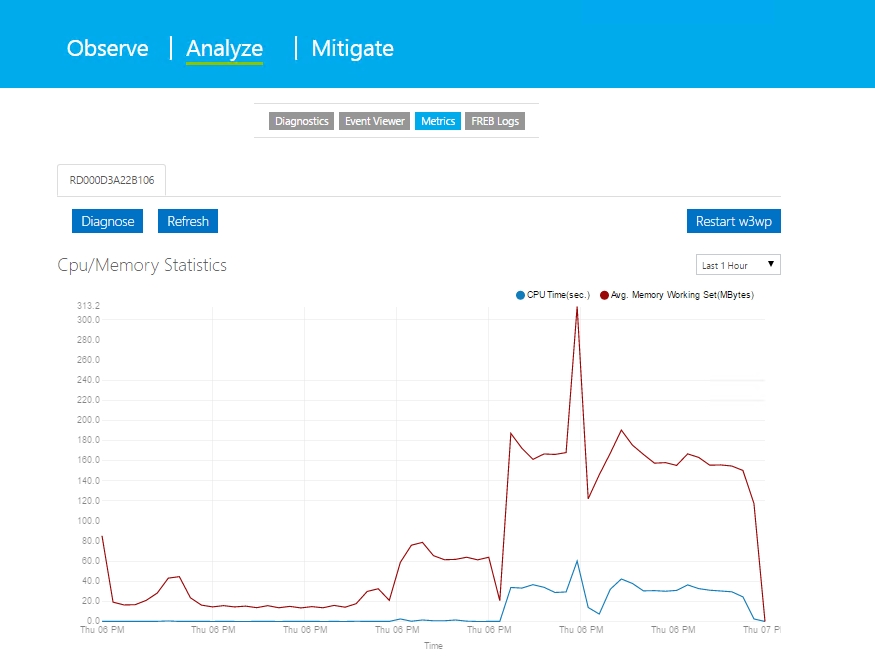


Next, from the **DIRECTORY** drop down menu, select the Subscription where the Web App was deployed and then select the **lumawebappdemo** Web App.

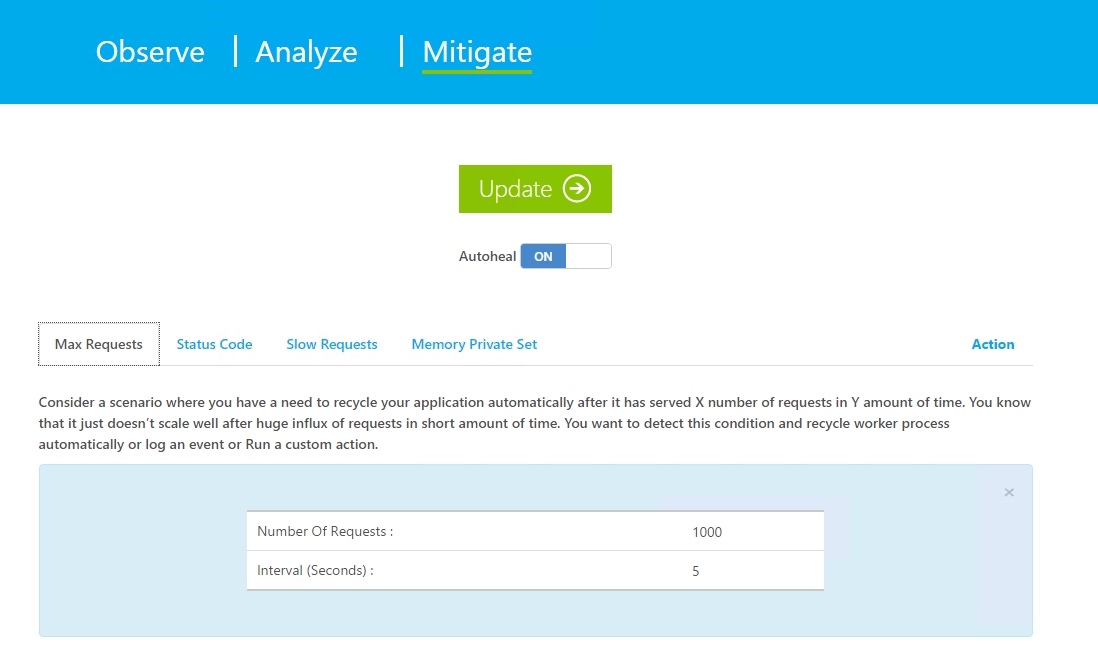
Under the Observe section of the Azure App Service Support page, you can click on the dropdown menu under **Select Web App** and view the Avg/Requests & Avg/Server Errors per sec. Additionally, you can change rather you want to see all Hostnames or only one Hostname.



Under the **Analyze** section, you have access to **Event Viewer Logs**, **Diagnostics**, **Performance Metrics** and **Failed Request Event Buffering (FREB) Logs**. An Example of **Cpu/Memory Statistics** is shown below.



Lastly, under the Mitigate section, you have the option to turn on the **Autoheal** option which allows you to configure the Web App to Log Events, Recyle an App Pool or perform custom events based upon criteria you set. For example, you can configure the number of Max Requests received before Recycling an App Pool as shown in the screenshot below.



Additionally, if the mitigation settings you set are not done completed with a certain period of time, you can configure custom Actions to occur as shown in the screenshot below.



1. Continous Deployment using GitHub

In this section we will be going over how you can perform continuous deployment to an existing Resource Group using Visual Studio. Before starting this section, make sure you have already deployed Azure Web Application using the **deploy-web-app** ARM Template included with this material using Visual Studio. Also, make sure you have access to the Azure Portal at <https://portal.azure.com>.

If the customer does not have a properly configured VPN or Azure Subscription in place, please have them consider the Azure IaaS Foundations IP at http://aka.ms/MCS\_EPG\_Azure\_Iaas-Foundation.

These are just examples; update the requirements adapting to the specific customer situation if necessary. Points to consider might include: tool usage (MAP versus customer provided), server locations, availability of test lab, and so on.