Azure App Service



Vision Scope

Prepared for

Microsoft

21-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](#_Toc441152733)

[2 Deploy an Azure Website with a SQL Database 4](#_Toc441152734)

[3 Walkthrough of Web App Features in Azure 12](#_Toc441152735)

[3.1 Configuring and using Diagnostic Logs 12](#_Toc441152736)

[4 Continous Deployment using GitHub 16](#_Toc441152737)

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

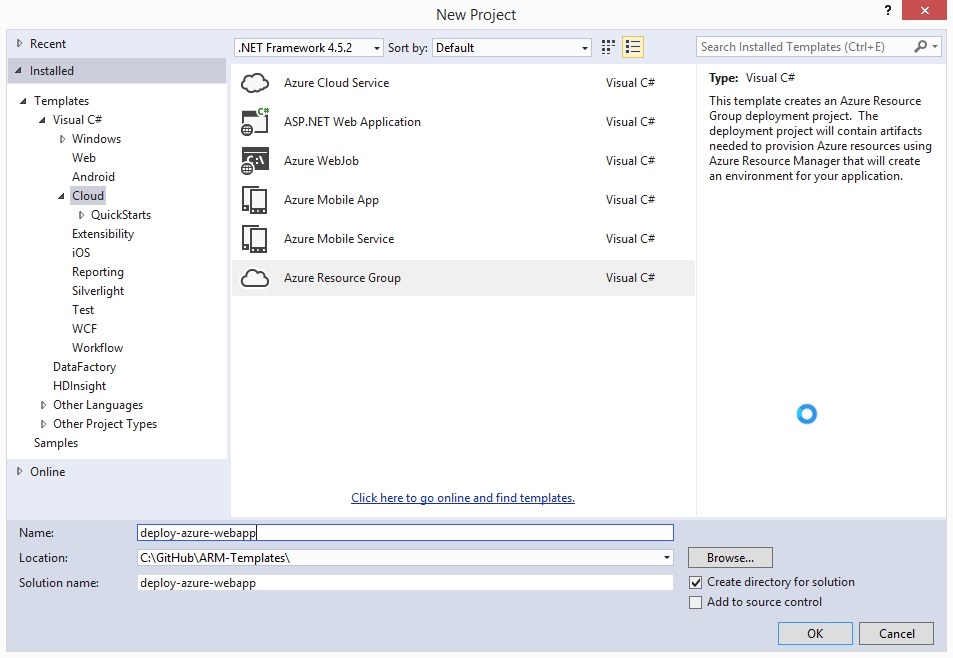
Finally, all Scripts, Templates and Applications mentioned within this Guide can be found in Lumagate’s GitHub Repository. If you require access, please contact a member of the Elite Incubation Team at Lumagate.

1. Deploy an Azure Website with a SQL Database

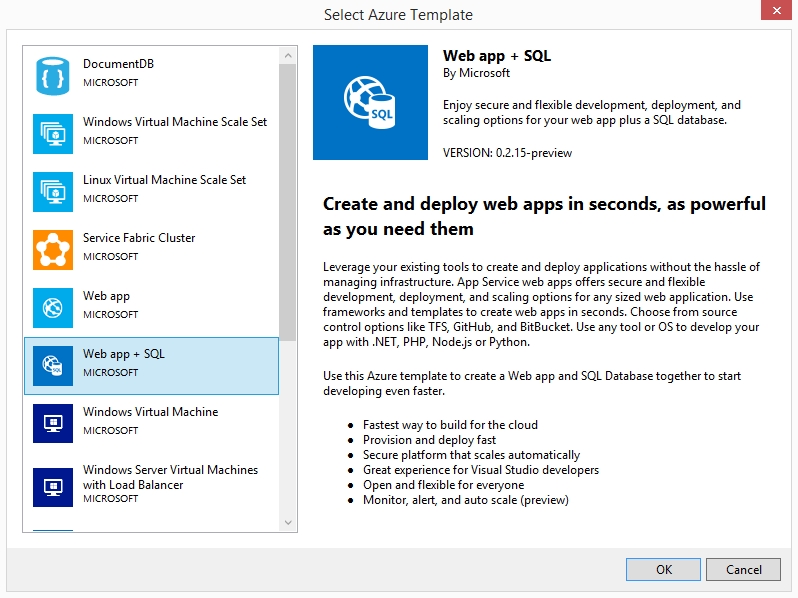
In this section, you will be deploying a Simple Web Application with a SQL Database attached using an Azure Resource Manager (ARM) Template.

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

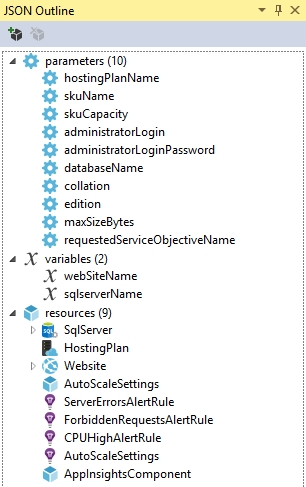
Next, in the New Project window, click on **Templates 🡪 Cloud 🡪 Azure Resource Group**. Change the Name and Solution Name fields to **deploy-azure-webapp** as shown below.



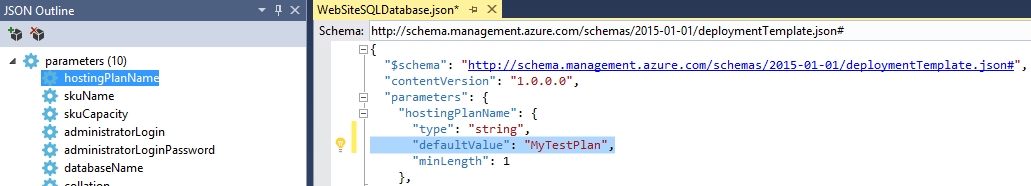
Next, look for the **Web app + SQL** Template and then click on OK.



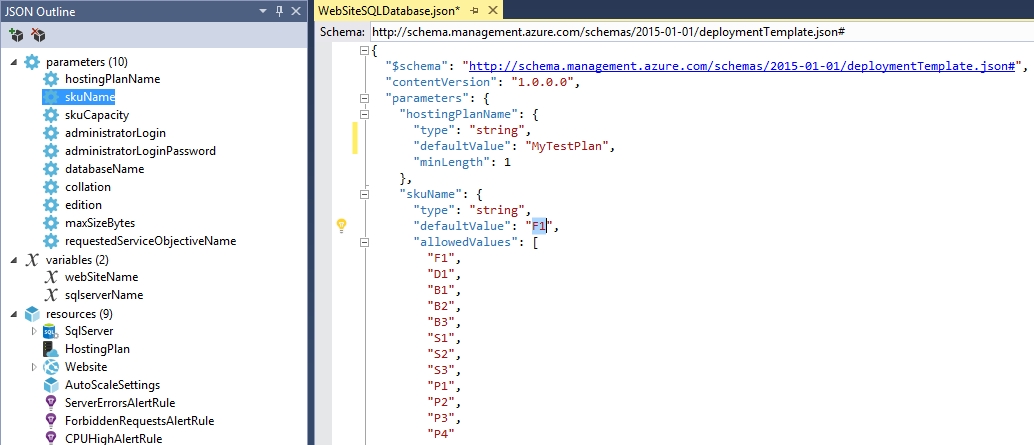
Once the Template has finished loading, expand out the **parameters**, **variables** and **resources** under the **JSON Outline**. Next, we are going to go over some of the options available to you when deploying an Azure Web App from an ARM Template and set some of their values before deploying the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



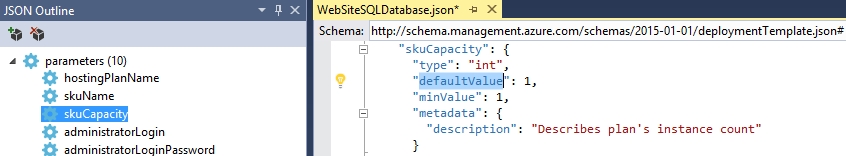
First, click on **hostingPlanName** and give it a default value, such as MyTestPlan, as shown in the image below.



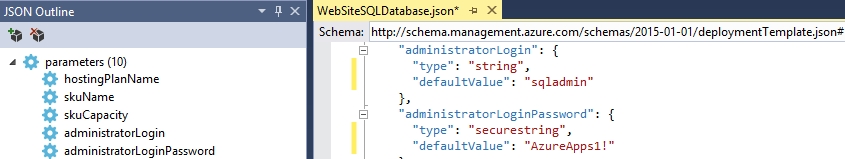
Next, click on **skuName**, change the defaultValue to S1. This will ensure that we have access to such capabilities.



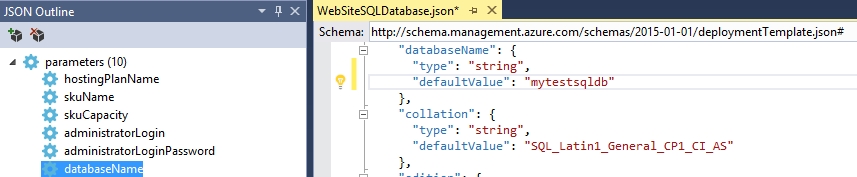
Next, click on **skuCapacity** and change the defaultValue from 1 to 4.



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



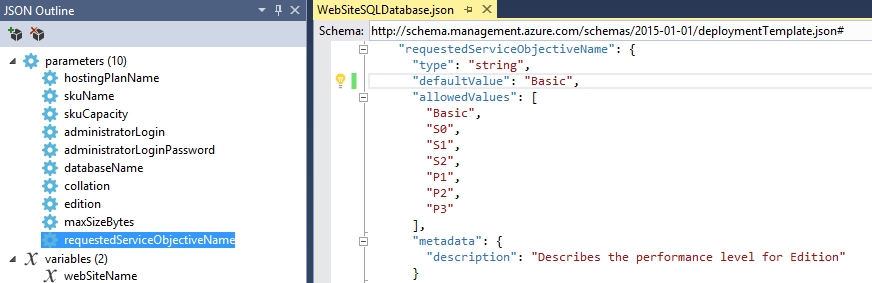
Next, set the **databaseName** parameter to **mytestsqldb**.



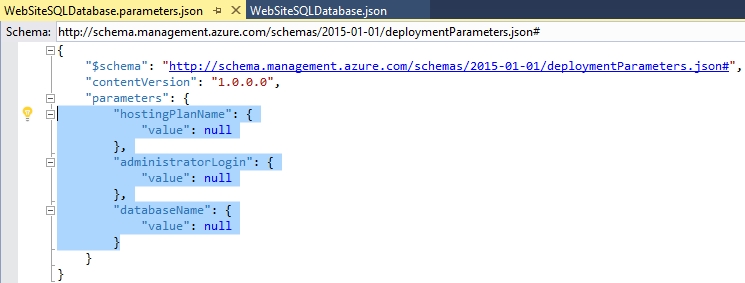
Click on the **edition** parameter, and change the defaultValue from **Basic** to **Standard**. By upgrading the value to Standard, you gain access to capabilities.



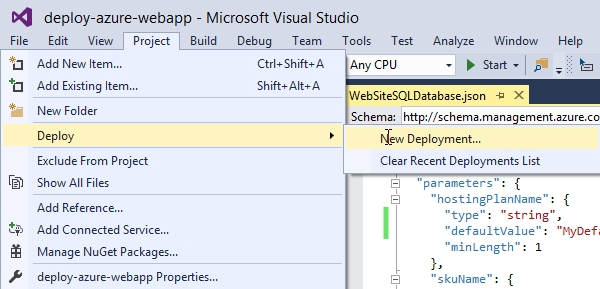
Next, click on the **requestedServiceObjectiveName** and change it from **Basic** to **S1**



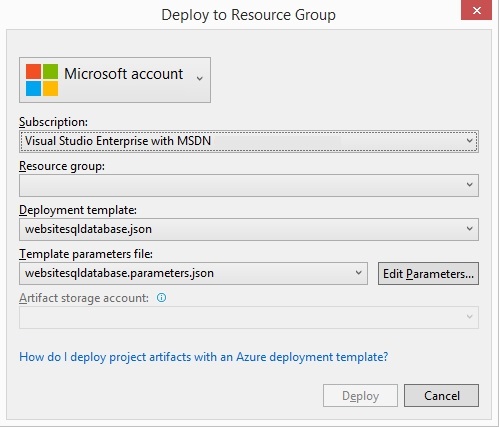
Next, open up the **WebsiteSQLDatabase.parameters.json** file and delete all the parameter entries highlighted below.



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



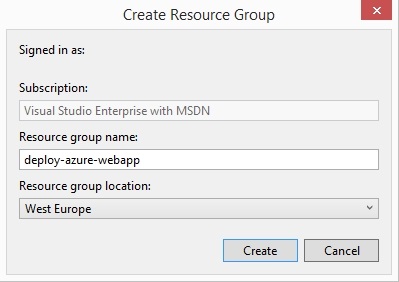
Put in your Microsoft Credentials if required and then click on the drop-down menu under **Subscription:** and choose which subscription you wish to deploy to.



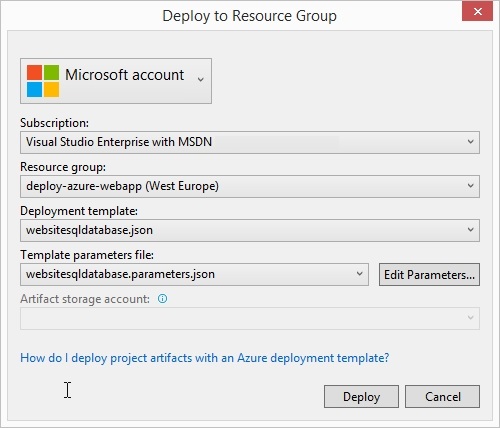
Next, click on the Resource Group drop-down Menu and click on Create New…



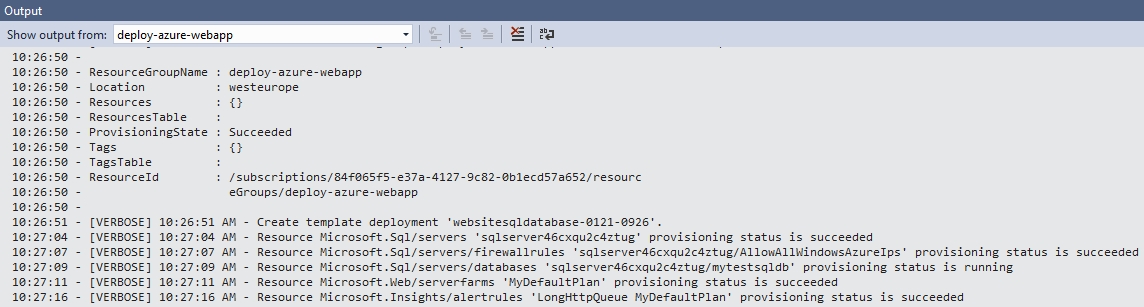
Take the default values for the Resource group name and choose a location to deploy to. When you are finished click Create.



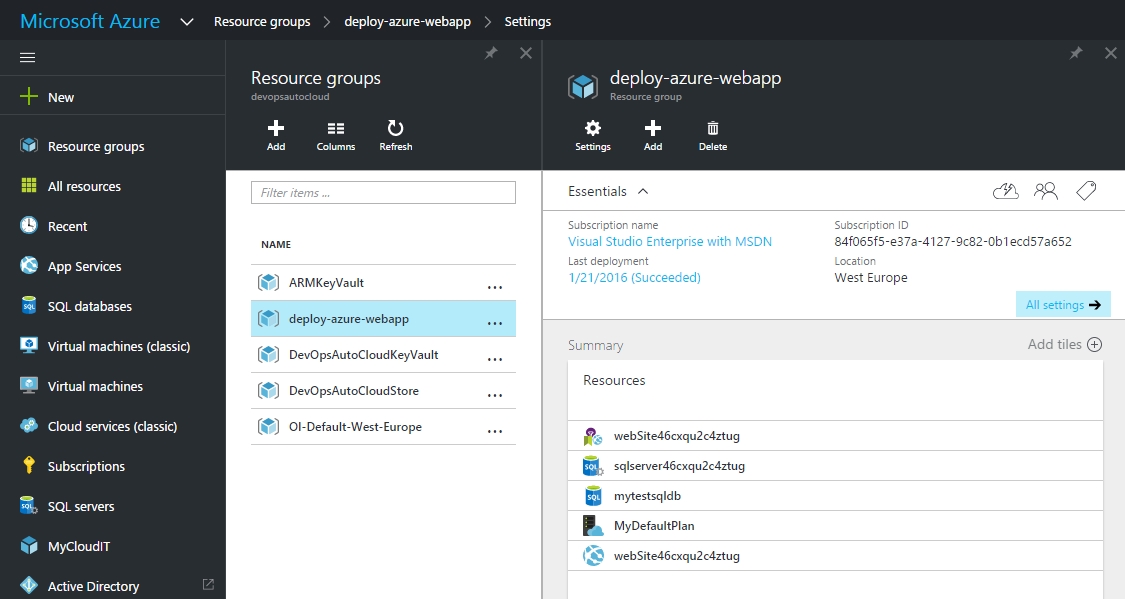
Lastly, you can double-check your parameter values and make changes to them if you so desire; afterwards click Deploy.



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.



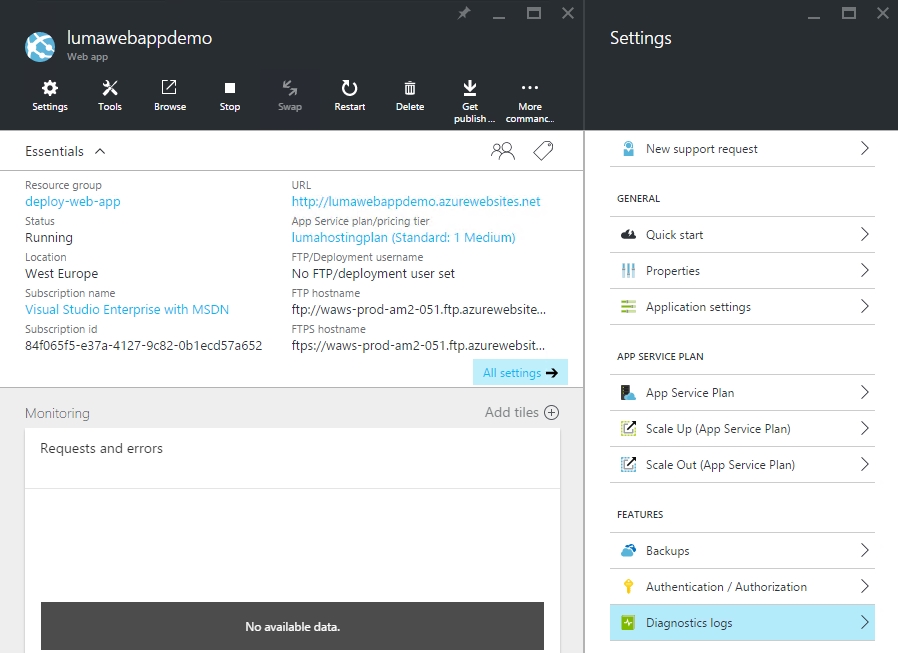
1. Walkthrough of Web App Features in Azure

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>.

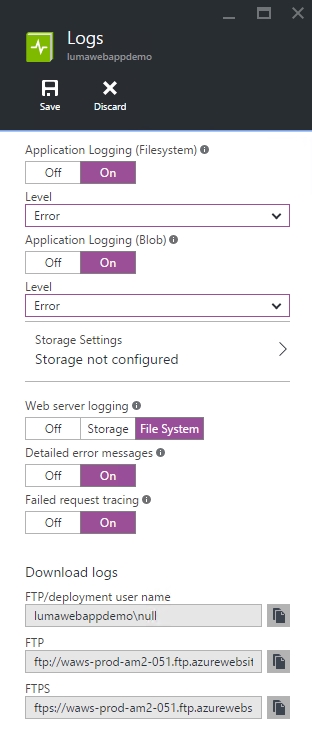
Finally, if you are deploying the deploy-web-app ARM Template in a classroom environment, make sure to append the names of the following parameter values with letters or numbers before attempting your deployment:

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

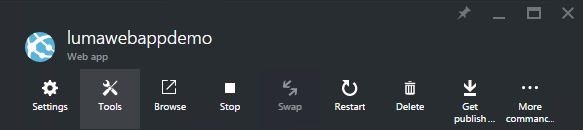
From the Resource Group where you deployed the deploy-web-app ARM Template, click on lumawebappdemo 🡪 All Settings 🡪Features 🡪 Diagnostic Logs



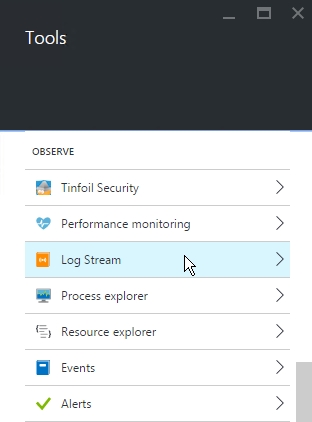
Next, turn on all of the features as shown below and make sure Web server logging is set to FileSystem. Once you are finished, click on the Save button. Within a minute, the settings will be updated and in place.



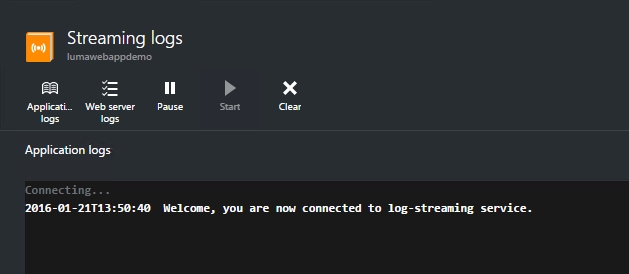
Next, go back to the lumawebappdemo Web app and click on Tools.



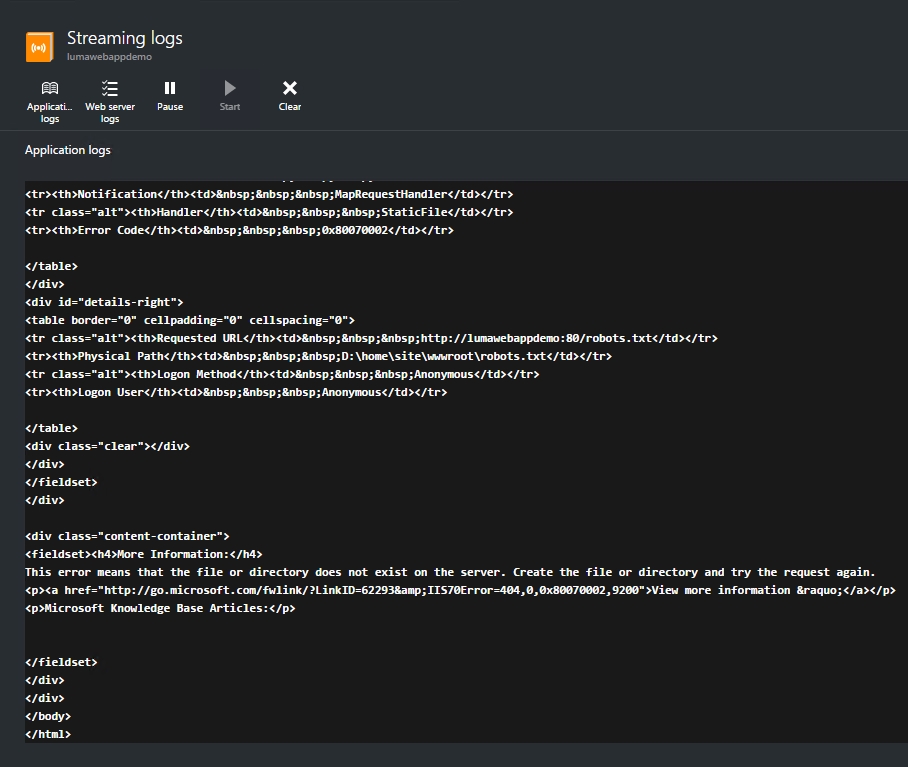
Next, click on Log Stream.



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. 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.

First,

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.