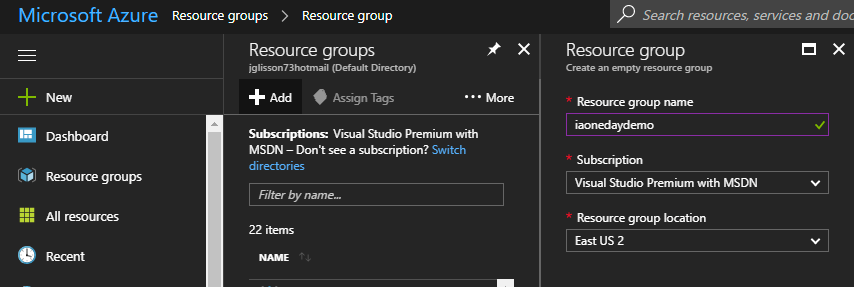
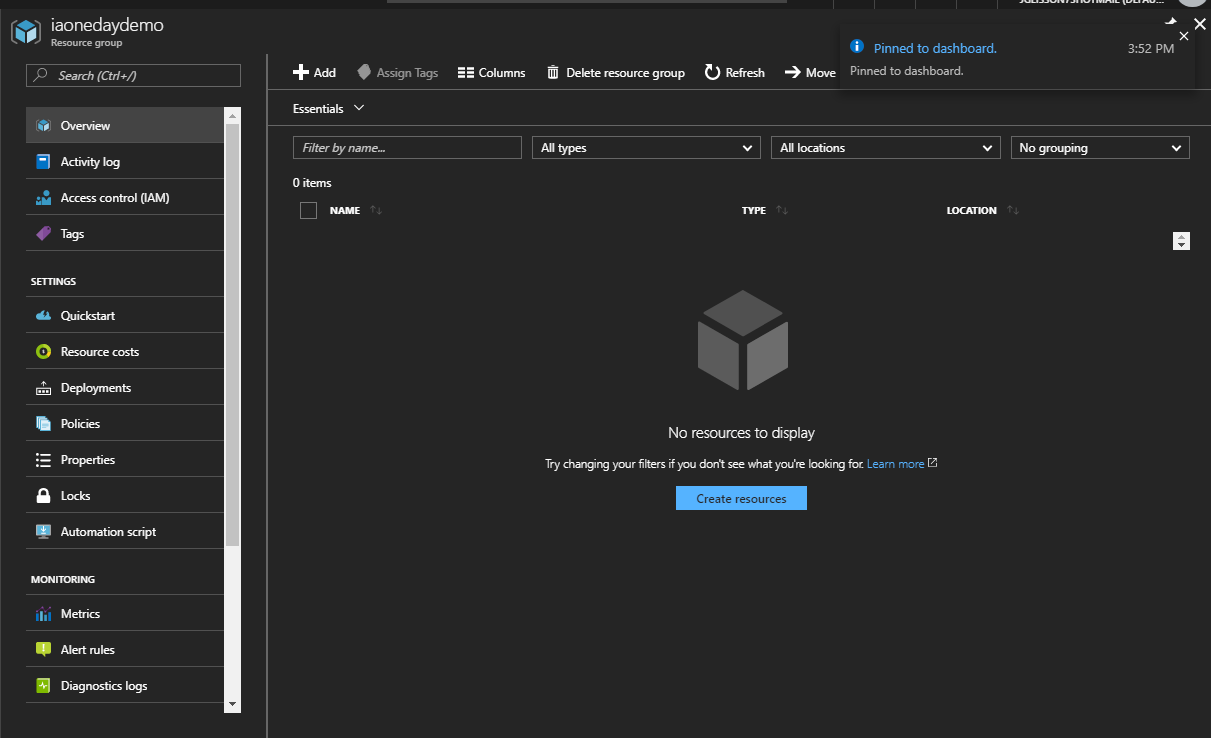
Lab Script

1. Log into <https://portal.azure.com> using the credentials for your Azure trial subscription.
2. Create an Azure Resource Group. The Resource Group is a logical container for your Azure resources for an application.



Once it is created, your empty Resource Group will look like the following. You will add all of your other resources from this screen.

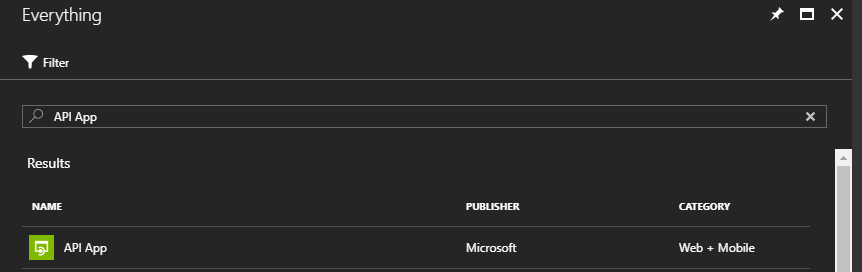


1. Add an API App within your Resource Group. The API App will be the container for your REST API.

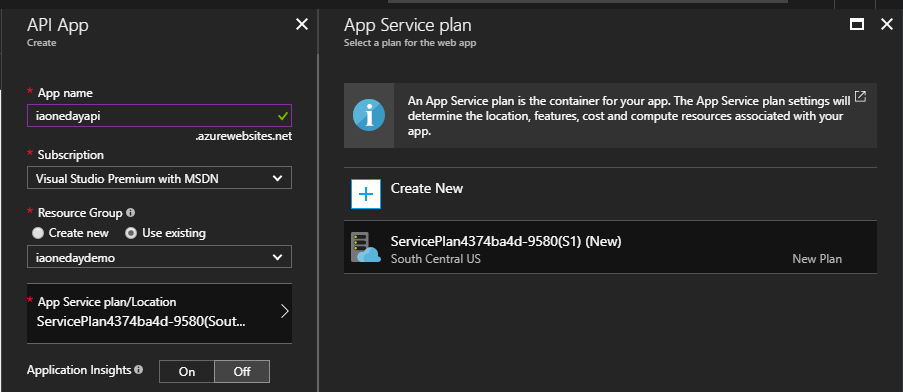
Click the Add button

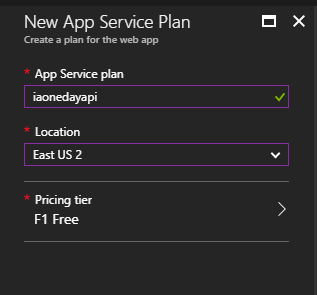


Search for API App



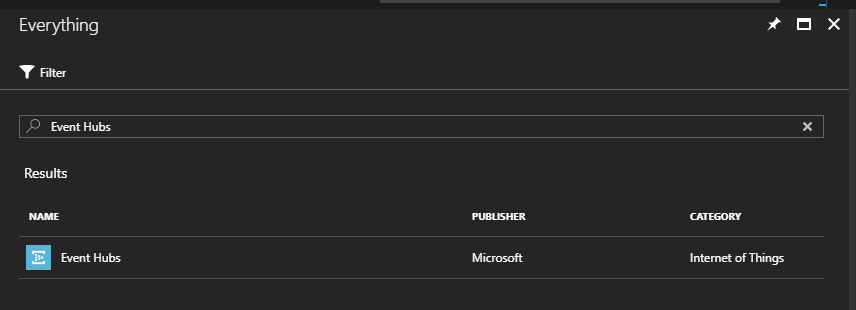
Create new App Service Plan in the F1 Free tier



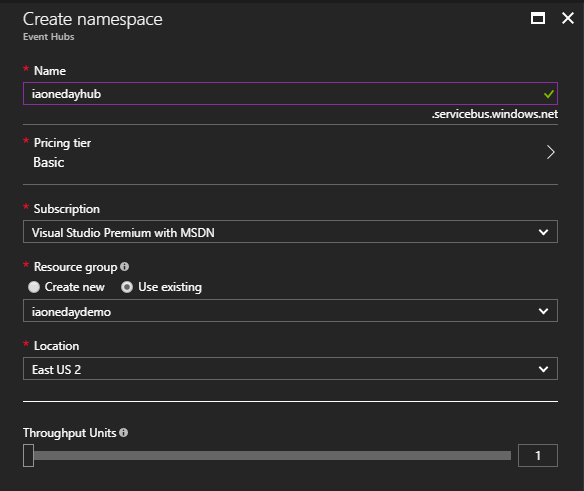


1. Create Event Hub Namespace

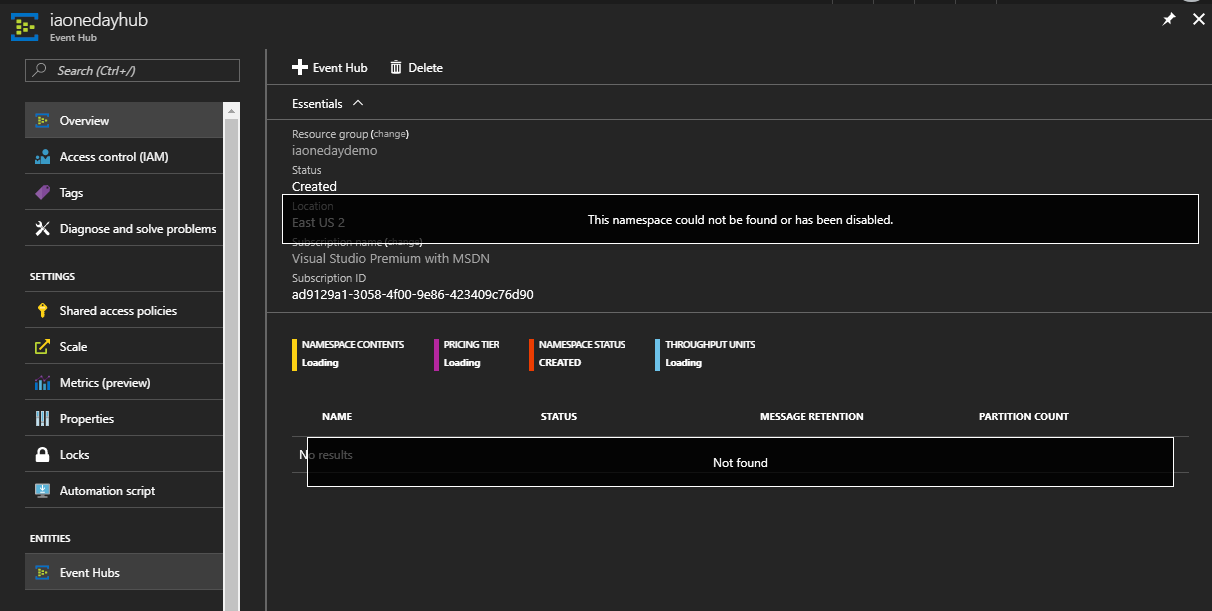
Click Add and Search for Event Hubs

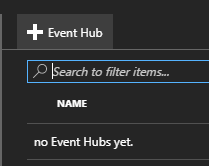


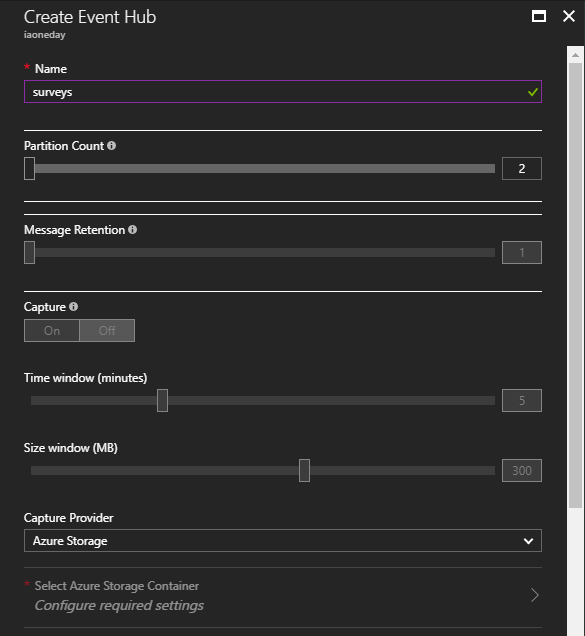
Select the Basic pricing tier



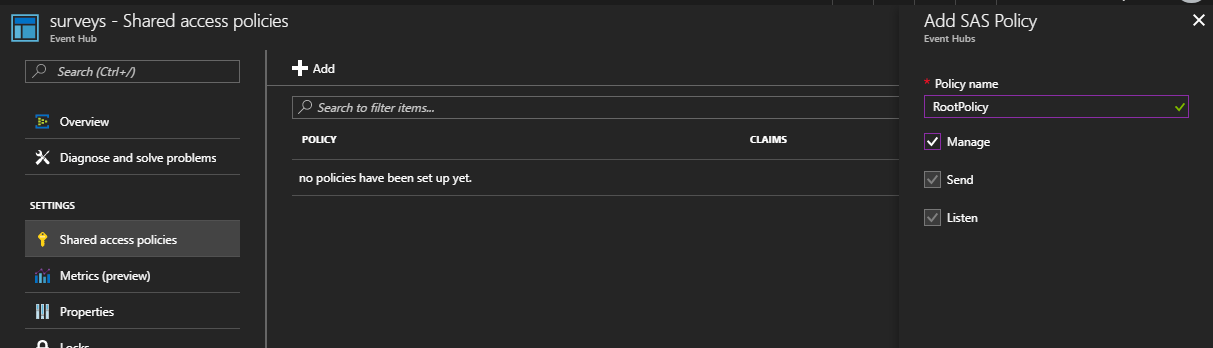
1. Create an Event Hub named “surveys” within the Event Hub namespace you just created



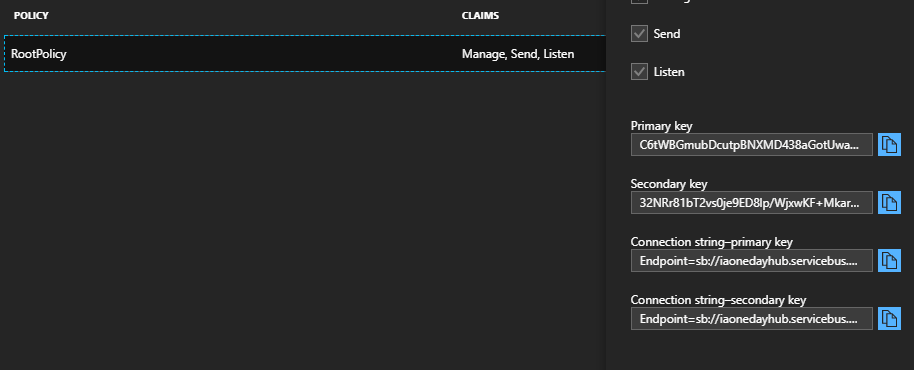




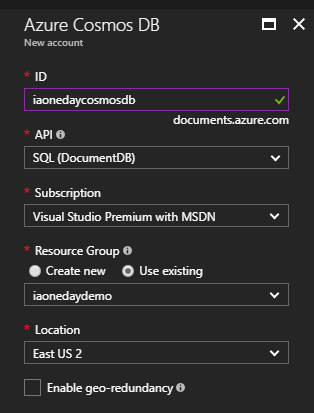
1. Add a SAS Policy named “RootPolicy” to the surveys Event Hub you just created. Give it Manage permissions. In a production app, you would create separate policies for Send and Listen. You may not even need one for Manage.



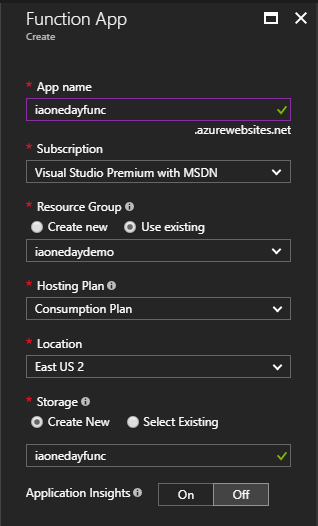
Once created, click on the RootPolicy and take note of the Connection string-primary value. Save this for later.



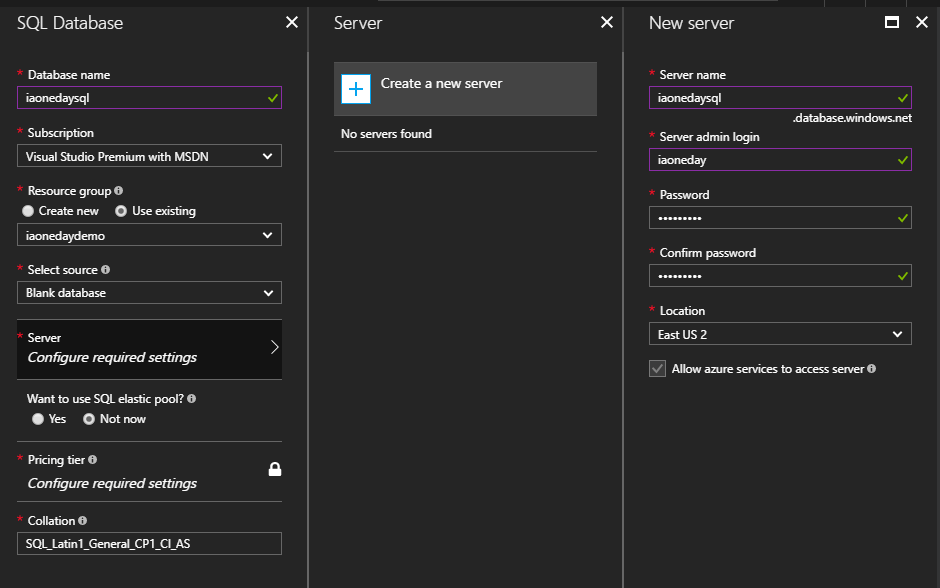
1. Add an Azure CosmosDB. It should be configured to use the SQL (DocumentDB) API. This will take a few minutes to deploy, so we will come back to get the connection information.



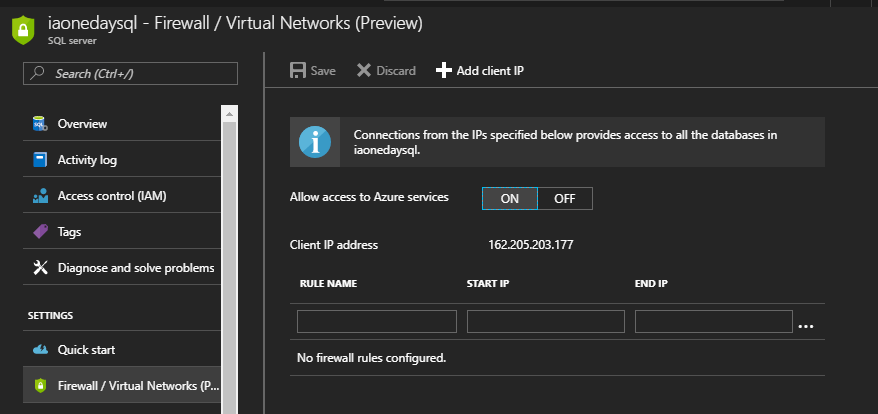
1. Add a Function App. Be sure it uses the Consumption Plan hosting. Also, let the Function App go ahead and create a new storage account for use.



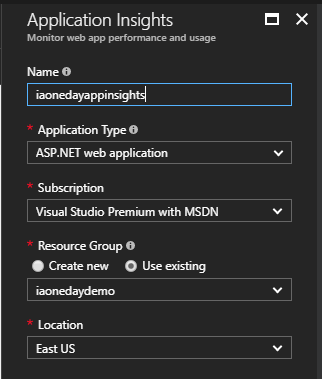
1. Add an Azure SQL Database and Azure SQL Server. Be sure to choose East US2 as the location for the Azure SQL Server. You also should choose the Basic pricing tier. Take note of the Server name, Server admin login and Password you use when creating your SQL Server. You will need them later.



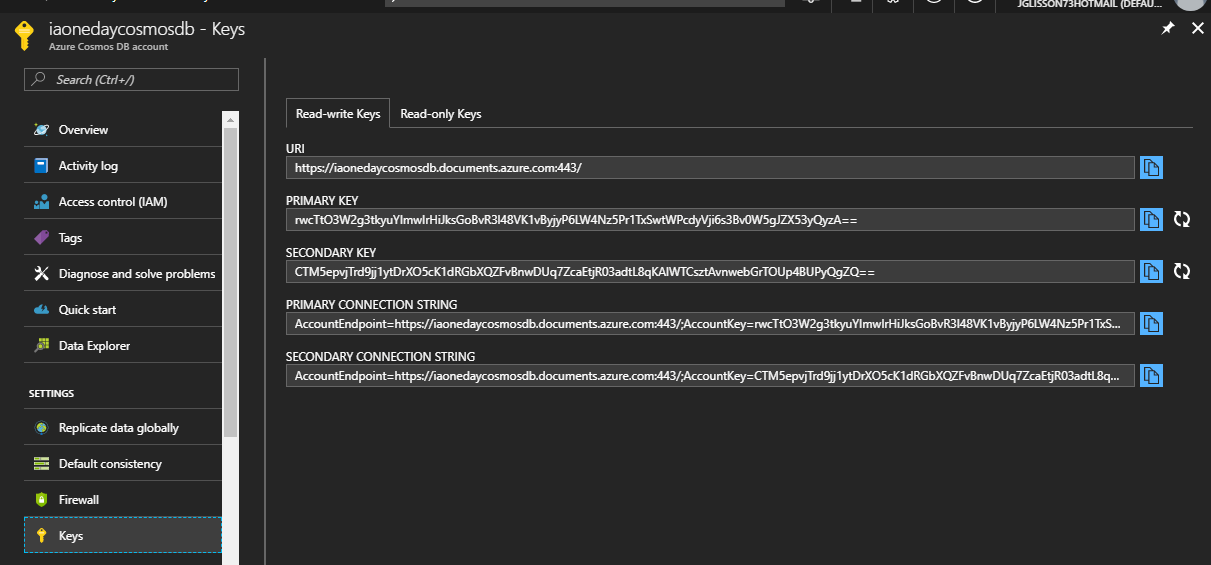
1. Once your Azure SQL Server has been created, navigate to it from the Resource group. Then click on the Firewall settings and Add client IP to add your present IP address to the Firewall and allow you to access the database remotely. Click Save to save this new setting.



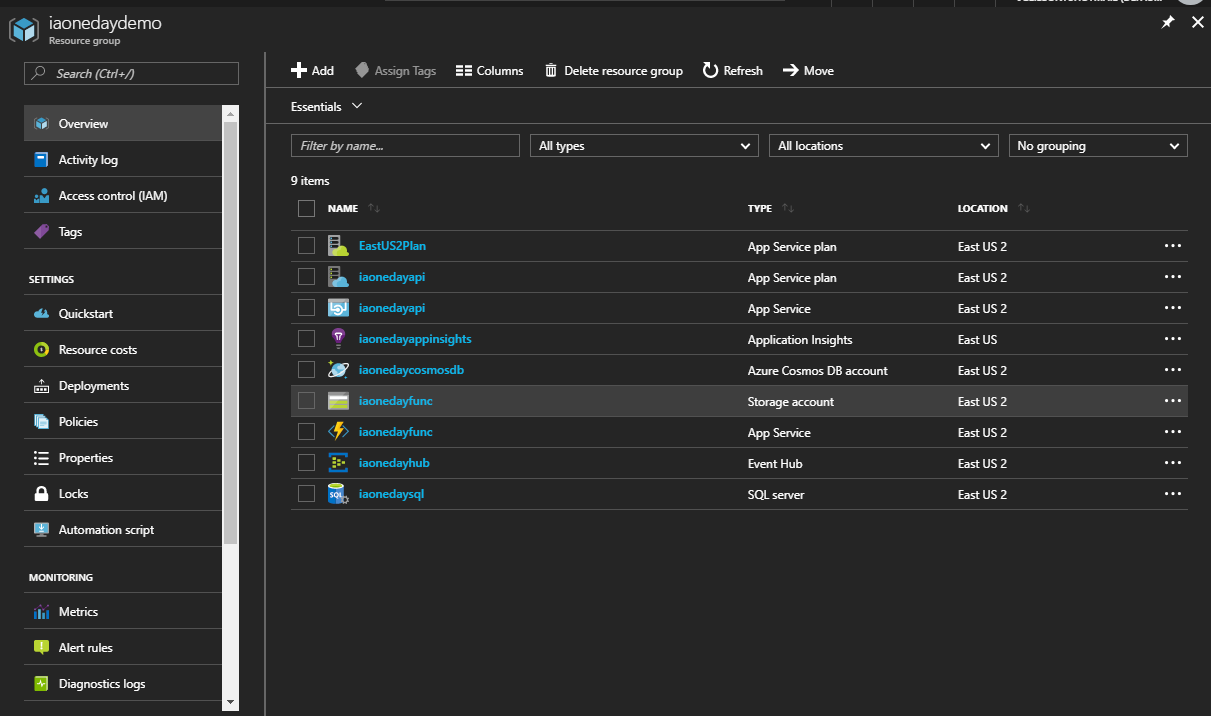
1. Add an Application Insights instance to the Resource Group. Keep the defaults.



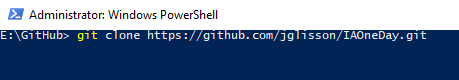
1. Retrieve the connection information for the Cosmos DB. Select the Cosmos DB from the Resource Group and then click on Keys. Take note of the URI and Primary Key values for later use.



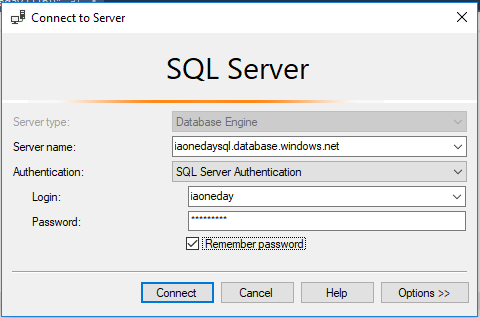
This is what the completed Resource Group should look like.



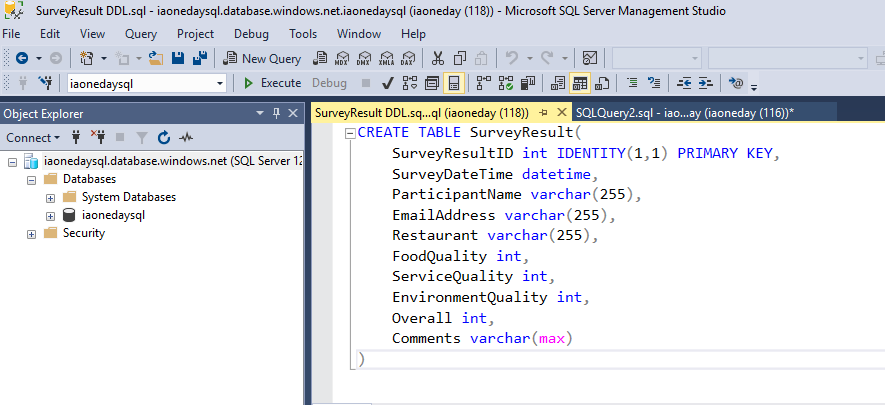
1. Retrieve the demo code from Github by cloning <https://github.com/jglisson/IAOneDay.git>



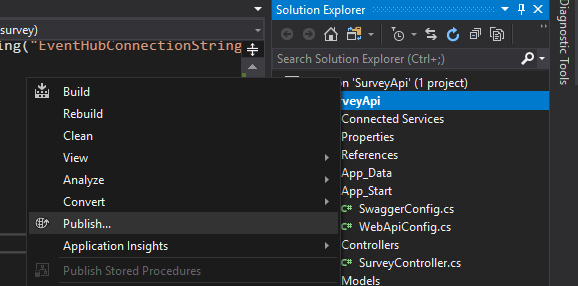
1. Create the table needed in the Azure SQL Database. First, connect to the Azure SQL Database in SQL Server Management Studio (SSMS). You will need to use the server name, Login and Password you used when creating your Azure SQL instance.

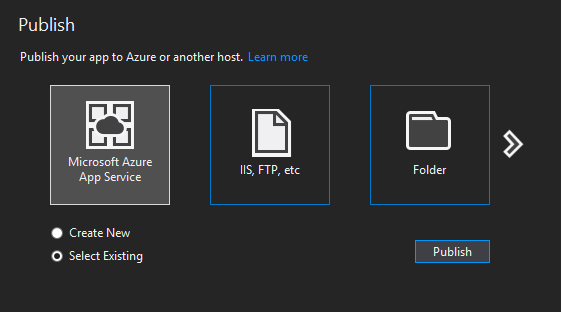


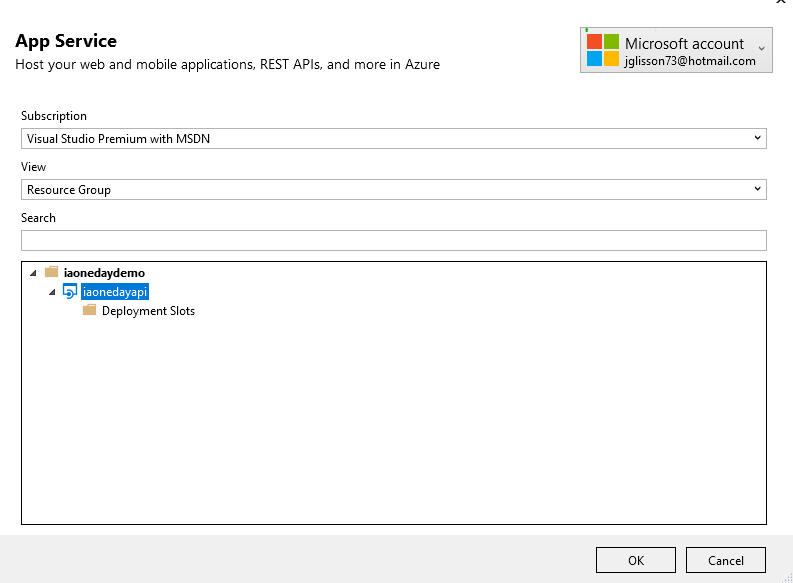
1. Once you are connected, Execute the SurveyResult DDL.sql file in the DB directory of your Git repository against the database you created. Be sure you have your database selected, and not the master database. You can leave SSMS open so you can query the table later and verify results.



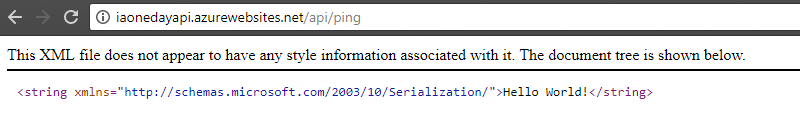
1. Open the SurveyApi application in Visual Studio. Then open the web.config and update the EventHubConnectionString value with the connection string you retrieved from the surveys Event Hub.
2. Publish the code to the API App container you provisioned in Azure.



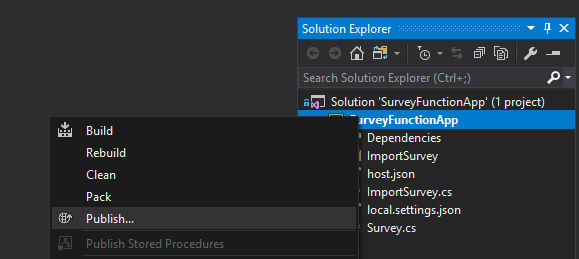




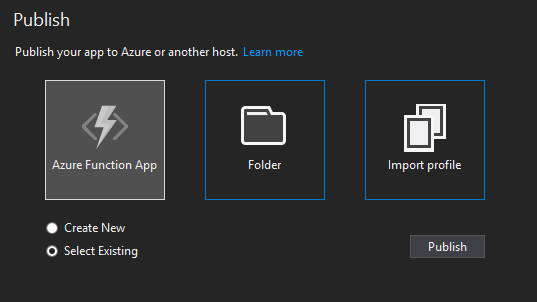
1. When the deployment is complete, your browser will pop up to the URL for your newly deployed API. Take note of the URL in your browser.
2. If you put /api/ping at the end of the URL in your browser, you should get a Hello World response showing you the API has been deployed properly.



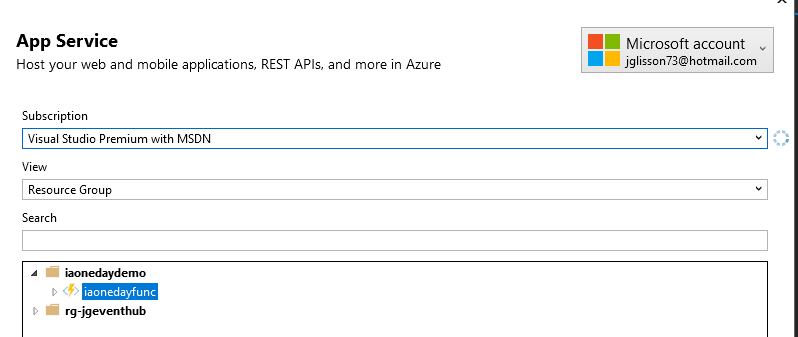
1. Open the SurveyFunctionApp solution from the Git repo in Visual Studio. Right-click on the project in Visual Studio and click Publish…



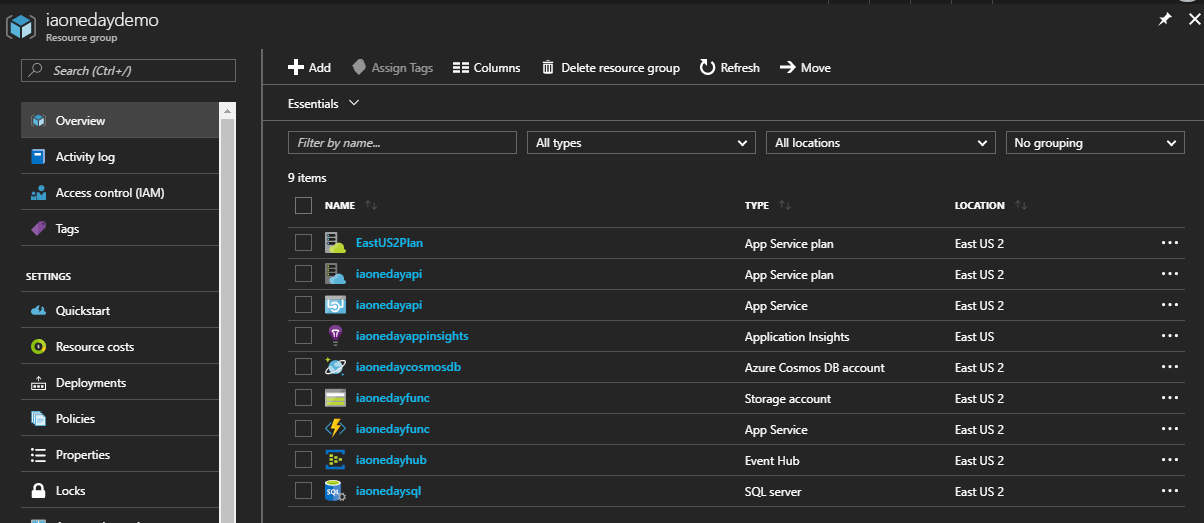
Select Existing Function App



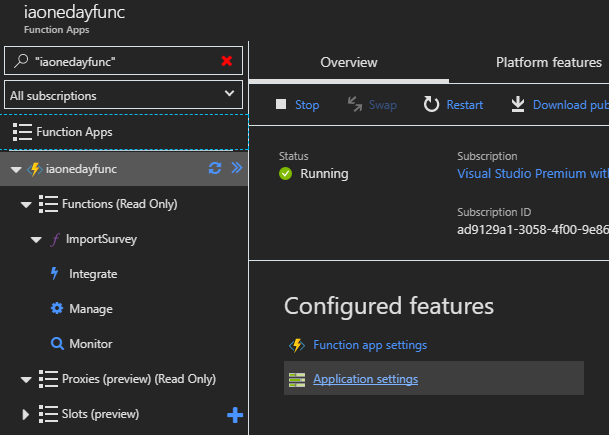
Choose the Azure Function App you provisioned.



1. Now we need to configure the Function App in the Azure Portal. Navigate to the Function App from the Resource Group in the Azure Portal.



Select Application Settings



You will need to add FIVE application settings and ONE connection string



**Application Settings**

eventHubConnectionString = the Event Hub connection string value you collected earler

cosmosDbUri = the Cosmos DB URI value you collected earlier

cosmosDbKey = the Cosmos DB key value you collected earler

cosmosDbCollection = Surveys

cosmosDbName = SurveyArchive

**Connection Strings**

sqlDbConnectionString = Server=tcp:**YOURSERVERNAME**.database.windows.net,1433;Initial Catalog=**YOURDATABASENAME**;Persist Security Info=False;User ID=**YOURUSERNAME**;Password=**YOURPASSWORD**;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;

Reference:

<http://www.c-sharpcorner.com/article/working-with-azure-api-apps/>

<https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-dotnet-standard-getstarted-send>