**Market Summary POC**

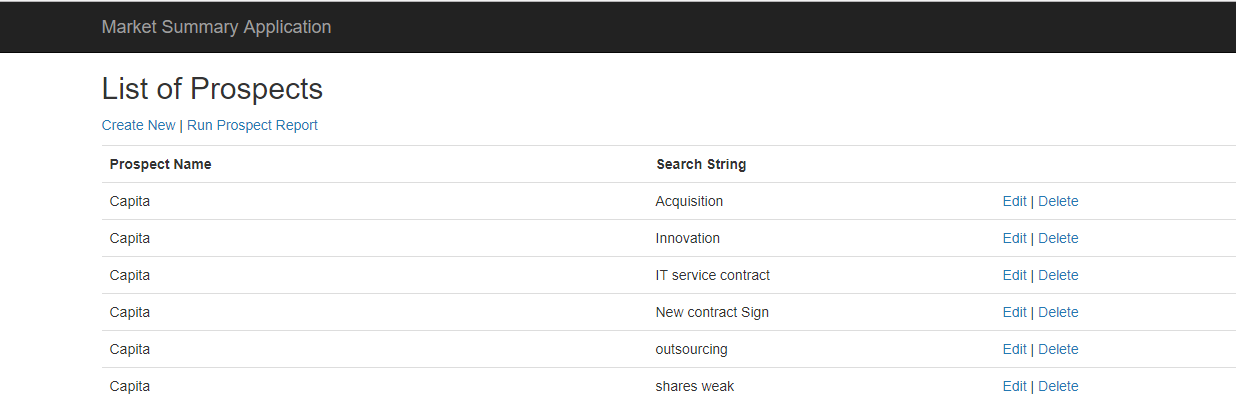
**Component Used/Explored**

1. Bing Search API
2. Azure SQL
3. Table Storage
4. Cosmos DB
5. Azure Functions
6. Logic App
7. Blob Storage

**Web Application (MarketSummaryWeb)**

This application is used to define the search criteria for prospects (master data). It allows you to perform CRUD operations on it. Based on this criteria, prospects data would get fetch using different azure components like Bing search API, Azure functions etc.

**GIT URL**: <https://github.com/EktaDate/MarketSummaryWeb.git>



This application is deployed on azure using visual studio.

URL: <https://marketsummary.azurewebsites.net/>

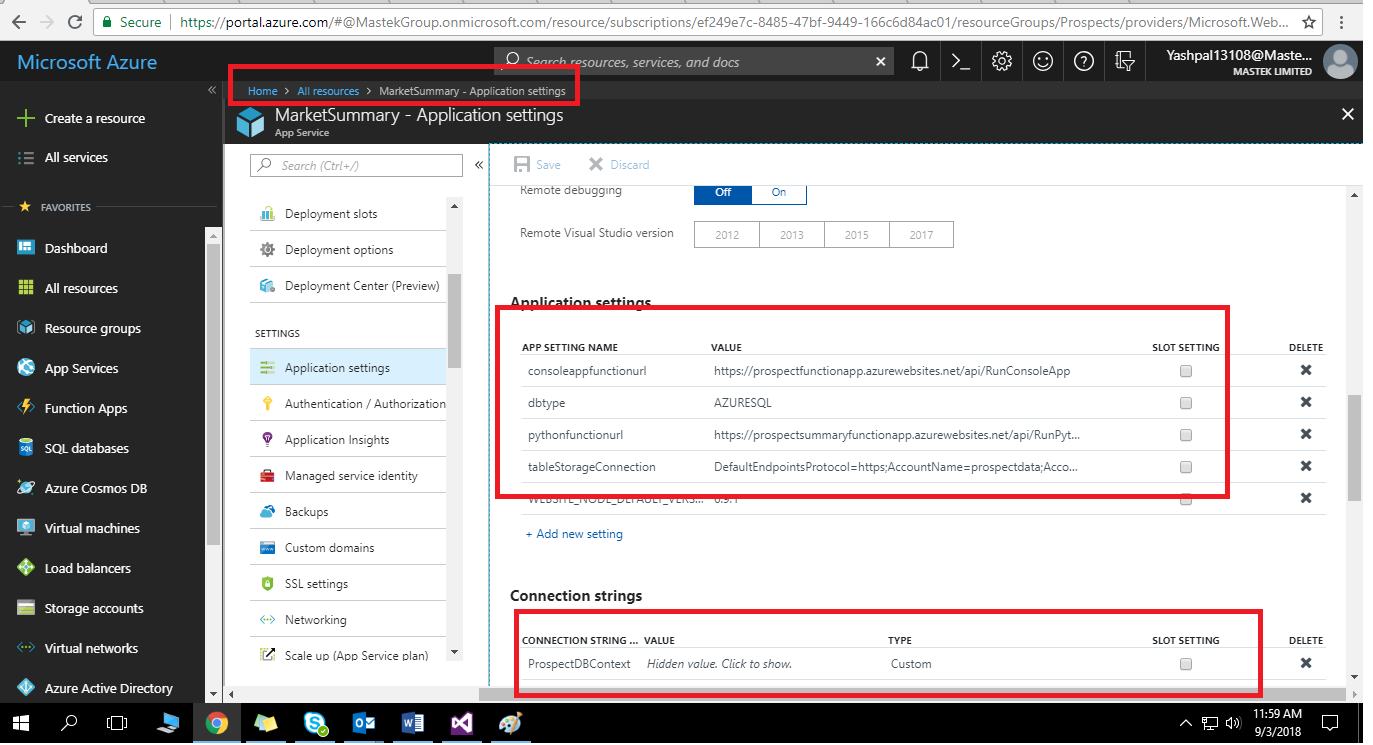
**Reference Link for deployment of web app on Azure Cloud.**

<https://docs.microsoft.com/en-us/aspnet/core/tutorials/publish-to-azure-webapp-using-vs?view=aspnetcore-2.1>

<https://www.c-sharpcorner.com/article/how-to-deploy-web-app-to-azure-using-visual-studio/>

Below application settings needs to be configure for the web app which we have deployed on Azure.

1. Go to Azure portal and select the Application Service
2. In Application service, go to Application settings
3. In Application settings define all the configuration. These configurations are same as configuration define in web.config e.g. Connection string.



**Console Application**

This application is used fetch prospect data based on the search criteria define in Web application. It will get called from Azure function and based on the parameters passed it would behave differently.

**Parameter: type**

If type is BINGSEARCH, it will run the process of Bing search updates or else run the process of Email updates.

GIT URL: <https://github.com/EktaDate/Market-Analytics-Tools.git>

**Flow for Bing Search**

1. Fetch search criteria for Bing search.
2. For each search criteria, call the Bing search API and fetch the URLs.
3. Store these URLs in Azure SQL or Table Storage in JSON format.

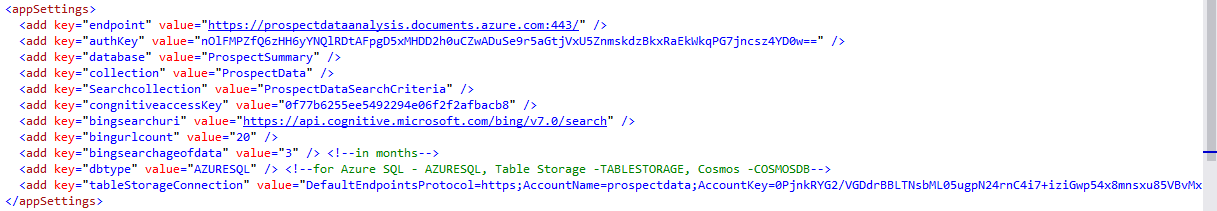
**Flow for Email Updates**

1. Get the Blob file name from parameters passed to the application.
2. Read Blob content which has the plain email text.
3. Store this plain email text in Azure SQL or Table storage.

Below keys are defined in app.config of console Application.

Here **dbtype** key is used to configure the database. E.g. For AzureSQL – AZURESQL,

Table Storage – TABLESTORAGE and Cosmos –COSMOSDB

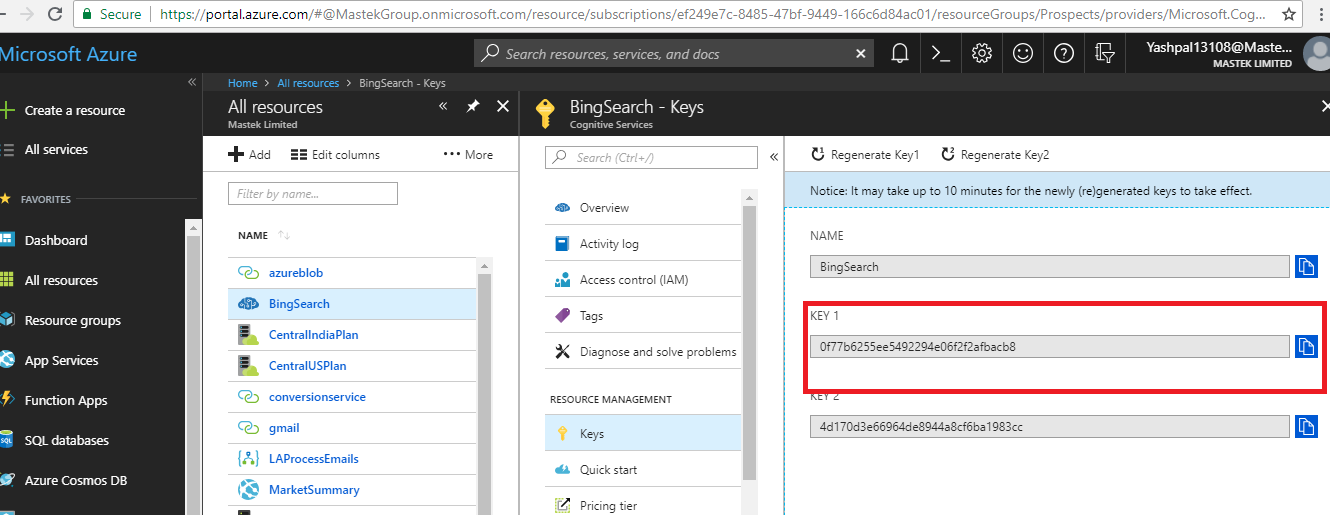


**BING Search Cognitive Service**

To use Bing search cognitive service we need to subscribe to the Bing search API. On subscription, we will get end point and access key which in turn used to call the Bing search API. You can refer to the below link to know how to call Bing search API from C#.

<https://docs.microsoft.com/en-us/azure/cognitive-services/bing-web-search/quickstarts/csharp>

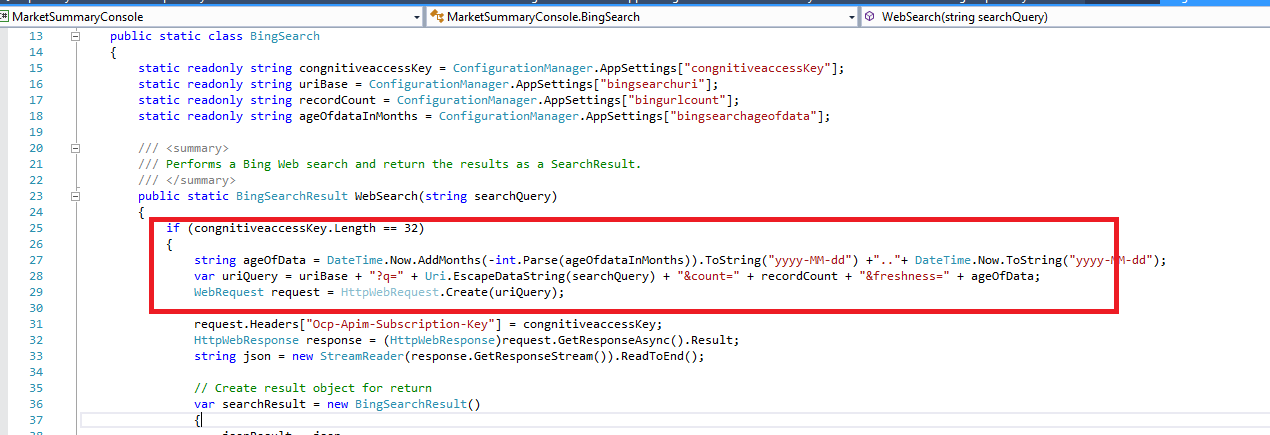
You will get the access keys from key section of the Bing Search as shown in screen shot.



We are passing below parameters to the BING Search API

1. Count – No of Urls to fetch
2. Freshness – age of the data i.e. Last 3 month/6 month /Year old data.

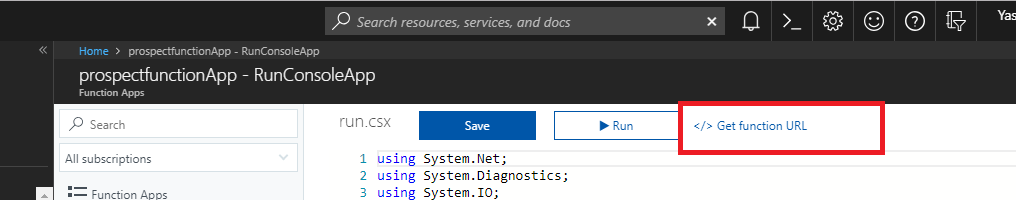
**Code Snippet**

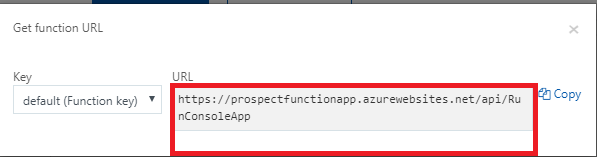


**Azure Function**

We have created the Azure function (RunConsoleApp) is of type HTTP trigger which is used to call the console application.

From the Azure portal we can get the function URL which will be used to call this function from our Web application.

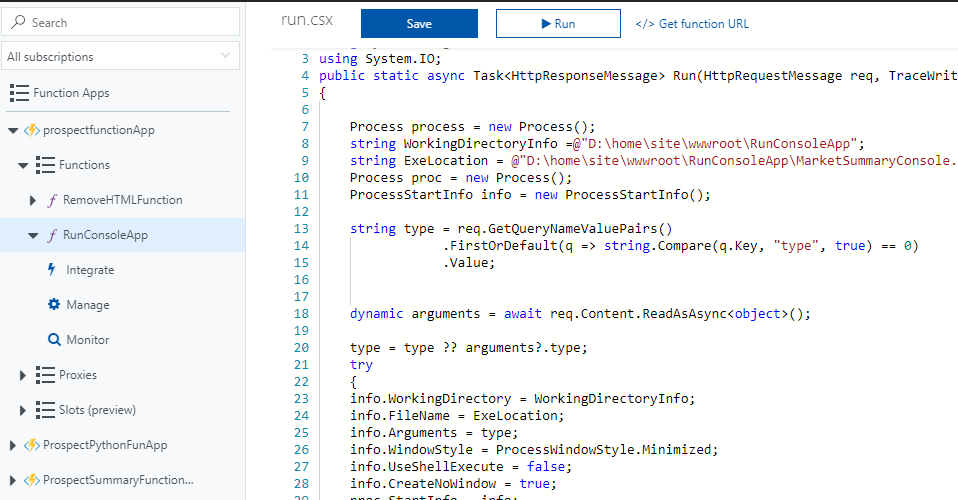


****

**Reference Link to Create Basic Azure Function**

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function>

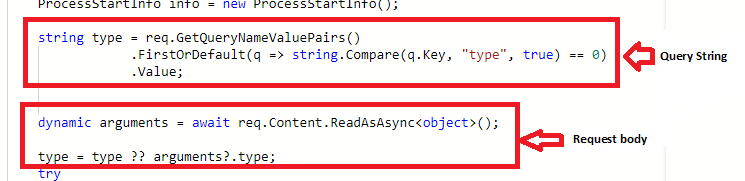
**Code Snippet of Azure Function**



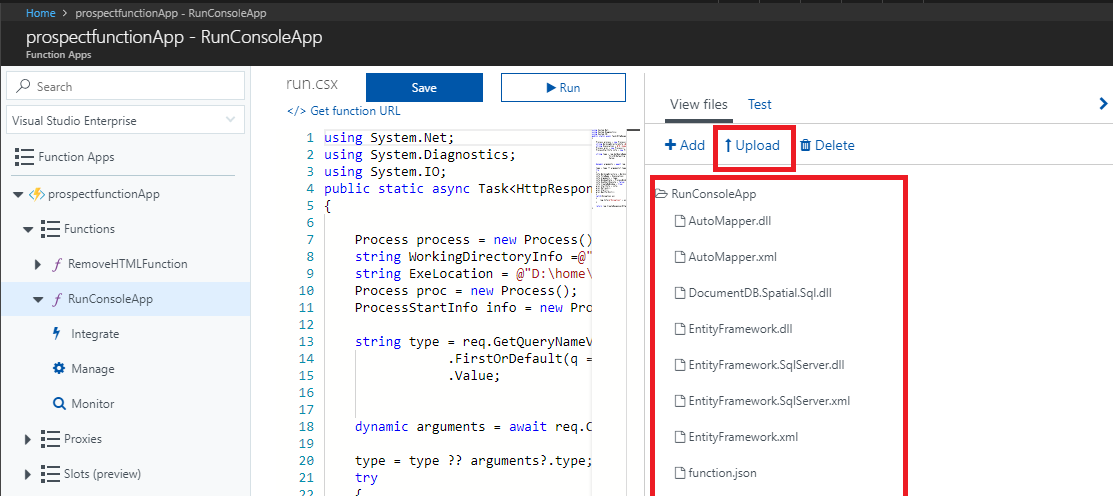
We are passing parameter to this function which in turn passed to the console application.

Here we are trying to read the parameter (type) in request url (querystring) or request body.

**Code Snippet to read parameter:**



As we are calling console application from Azure function, we need to upload the executable file and all the supporting dlls.

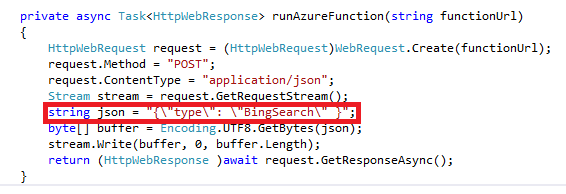


**How to call the Azure Function from C# Code**

By using HttpWebRequest and HttpWebResponse, we can call the Azure function. These objects use function URL for it.

**Code snippet.**

Here, we are passing parameter **type as “BingSearch”**

  
  
**AZURE SQL**

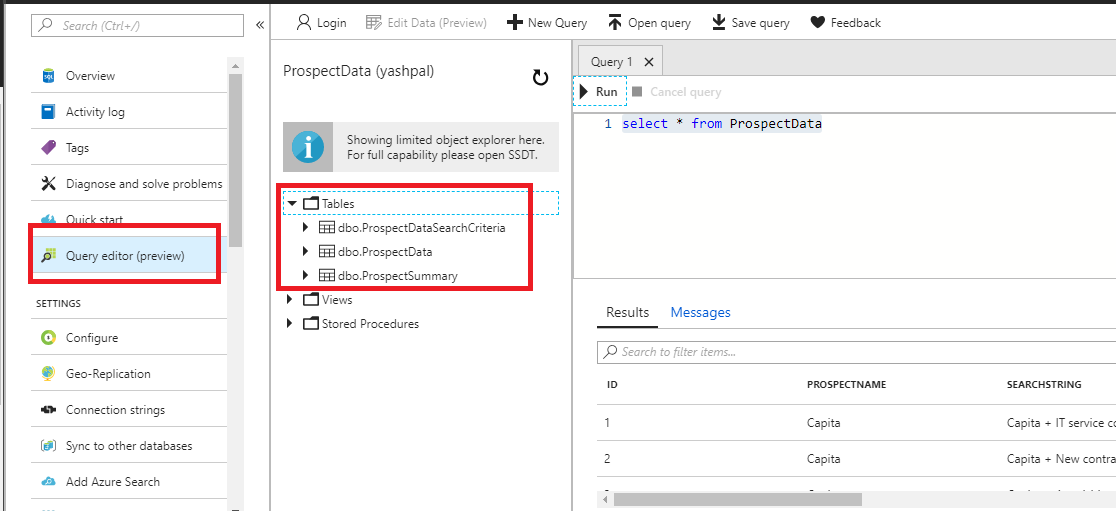
We have created SQL server and database. While creating this resources, you need to create credentials which in turn will be used to connect to the sql server.

Reference URL: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-get-started-portal>

Here we have created below tables in ProspectData Database

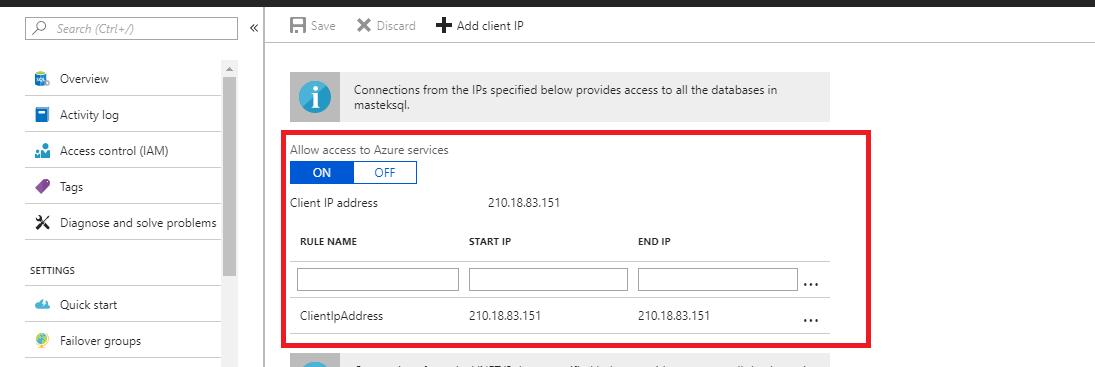
1. ProspectDataSearchCriteria – Stores the SearchCriterias for different prospects.
2. ProspectData - Stores the Prospect Data like Bing search URL and Email Body.
3. ProspectSummary – Stores the output of processed or analyzed data I.e. prospect summary.

There is query editor, wherein we can run the basic SQL commands and see the output.



Initially, all Transact-SQL access to your Azure SQL server is blocked by the firewall. To begin using your Azure SQL server, you must specify one or more server-level firewall rules that enable access to your Azure SQL server. Use the firewall rules to specify which IP address ranges from the Internet are allowed, and whether Azure applications can attempt to connect to your Azure SQL server.

For SQL server, we need configure firewall



**Reference URL:**

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-firewall-configure>

In Web Application and console application, we have created SQLRepository which is used to connect to the SQL using entity framework.



**Table Storage**

For table storage, we have created storage account named as “Prospectdata”. To connect to the storage account, we have used the access keys and connection string. You can find those keys in Azure portal under keys sections of the storage account.

Reference Links:

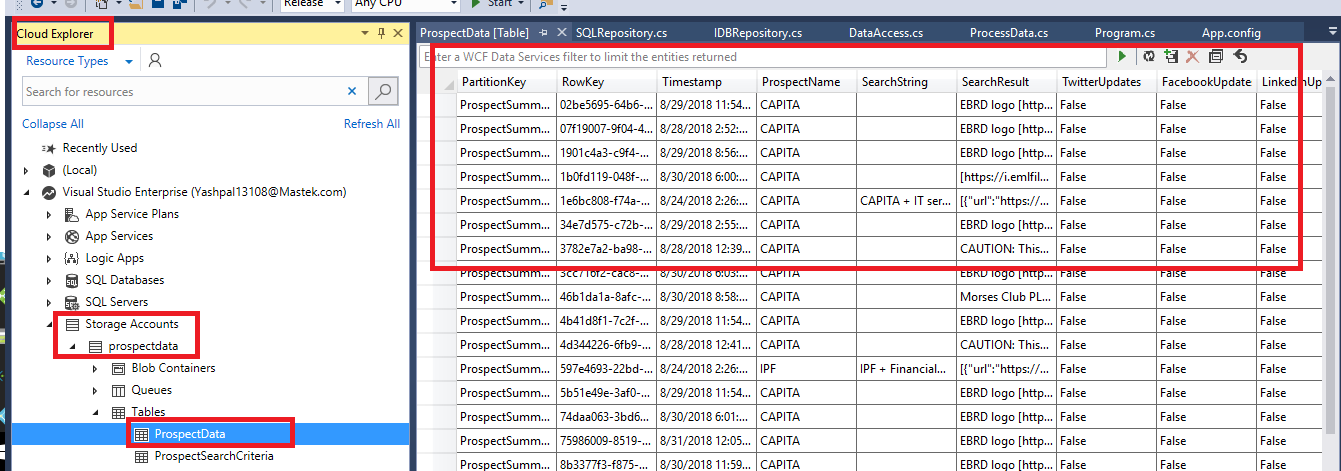
<https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction>

<https://docs.microsoft.com/en-us/azure/storage/common/storage-create-storage-account>

<https://docs.microsoft.com/en-us/azure/cosmos-db/table-storage-overview>

In table storage, each table (as in SQL) is referred as entity and data is stored in key value pair.

We can view table storage in cloud explorer of visual studio.



In Web Application and Console application, we have created TableStorageRepository which is used to connect to the Table Storage. Generally in config file, we store the connection string require to connect to the table storage.

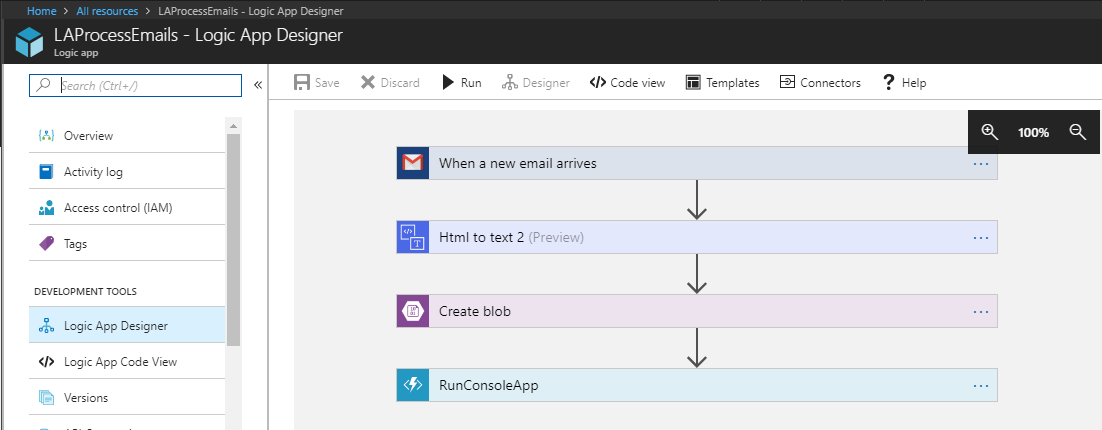


In tableStorageRepository.cs you will find the all methods require for CRUD operations.

**LOGIC APP**

[Azure Logic Apps](https://azure.microsoft.com/services/logic-apps) is a cloud service that helps you automate and orchestrate tasks, business processes, and [workflows](https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview#logic-app-concepts) when you need to integrate apps, data, systems, and services across enterprises or organizations. Logic Apps simplifies how you design and build scalable solutions for app integration, data integration, system integration, enterprise application integration (EAI), and business-to-business (B2B) communication, whether in the cloud, on premises, or both.

Here we have created logic app named as “LAProcessEmails.” This app is triggered when new mail is received in inbox (Here we have configure the Mastek Gmail id.). It has some configured actions, so whenever logic app is triggered, these actions get performed.



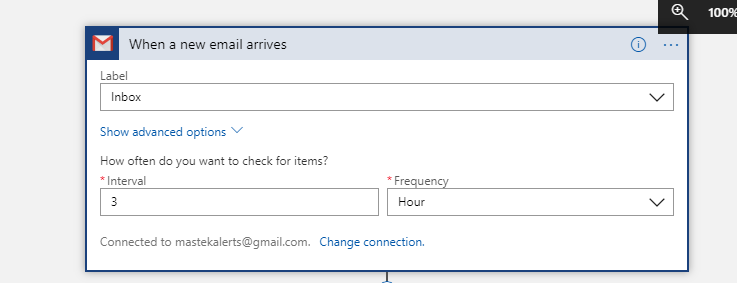
**Reference Link:** <https://docs.microsoft.com/en-us/azure/logic-apps/tutorial-process-email-attachments-workflow>

**Details Flow**

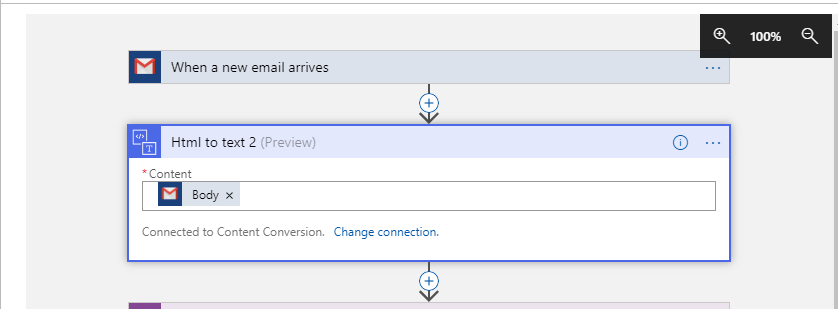
1. When new mail arrives. Logic app trigger is fired.

Here we can configure, how often you want to check for items.

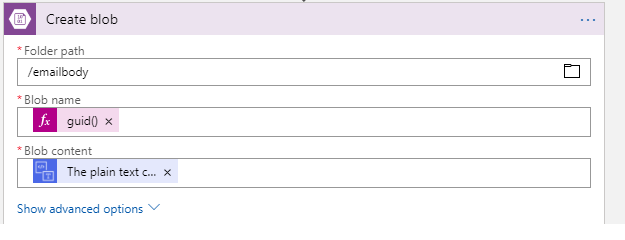
i.e. Interval and Frequency



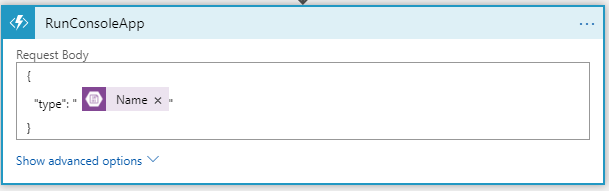
1. Read the body of Email and remove the HTML from it. We have used connector for it. (HTML to Text converter)



1. Create the Blob for the plain text generated in step 2.

 4) Call Azure function i.e. RunConsoleApp.

In this step, Azure function will call our console application which in turn will save this email body in the database.



Here, we are passing blob file name to the Azure function as a parameter. Azure function will read the content of the blob object and store it into the database.