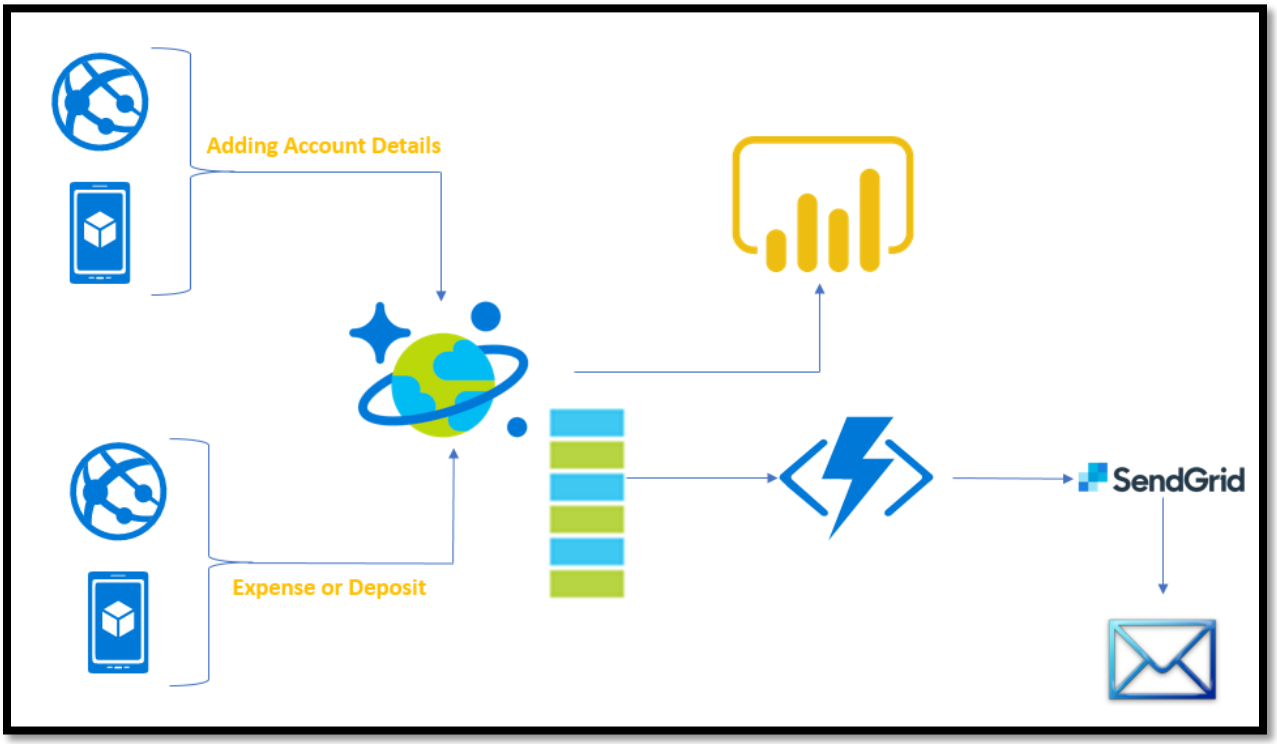
Azure CosmosDB in Banking Sector

Use case

In this article, we'll consider a use case where customer account details will be maintained and customer get notification of any transaction happen on their account. The balance will get adjusted based on the transaction. Below is the architecture of it.

Architecture

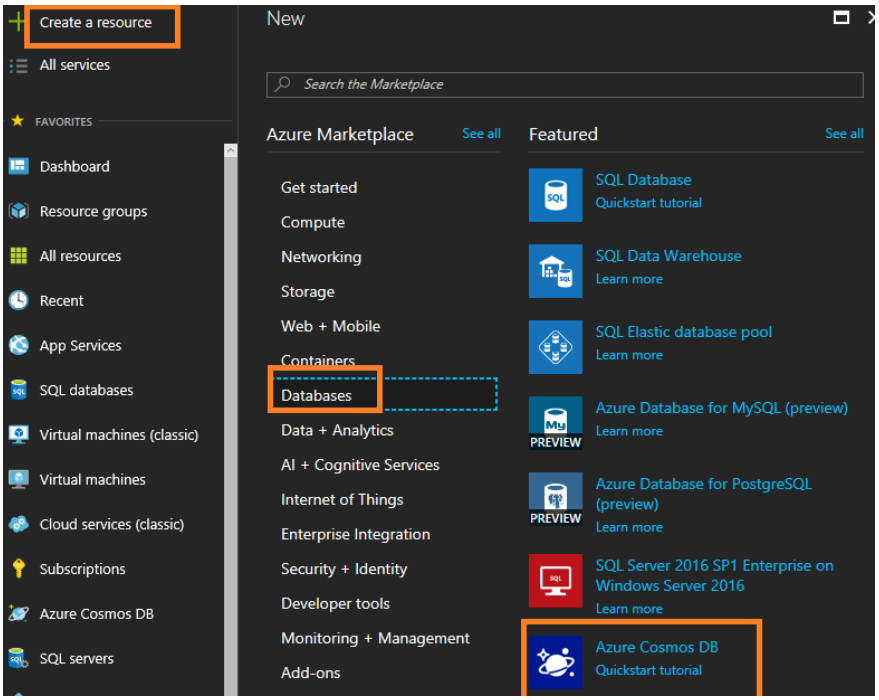


Prerequisites

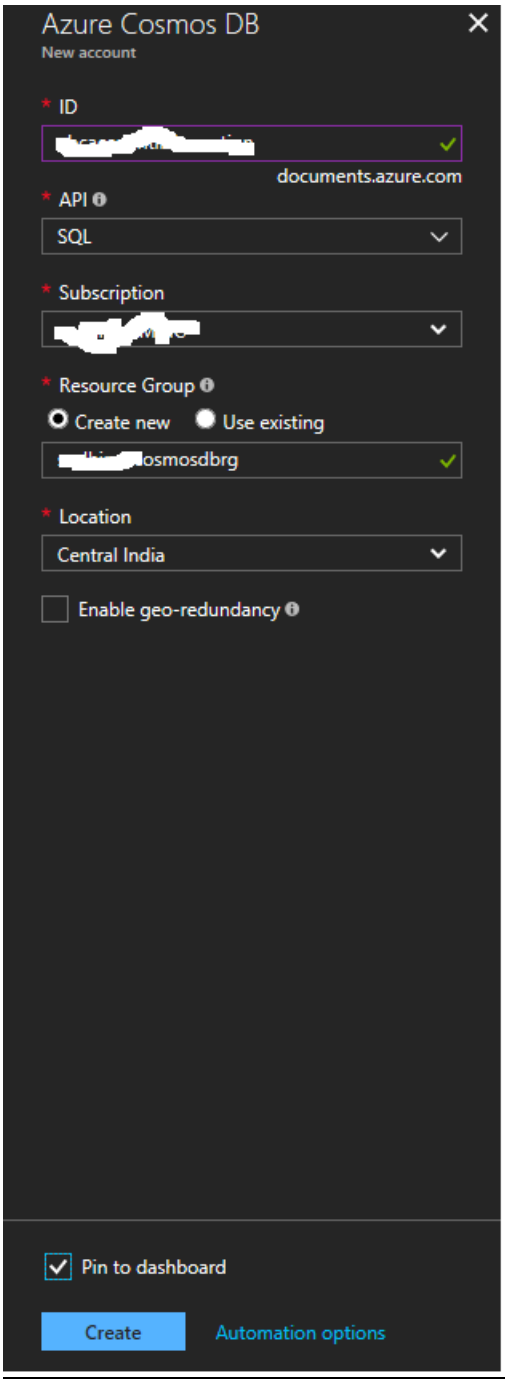
- Active Azure Subscription
- Visual Studio 2017

Setting up Azure CosmosDB account

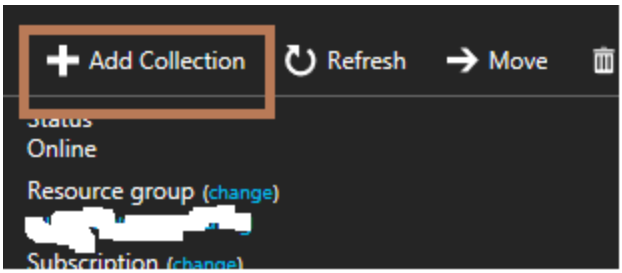
- Goto www.portal.azure.com
- Click **Create a resource** -> **Databases** -> **Azure Cosmos DB**



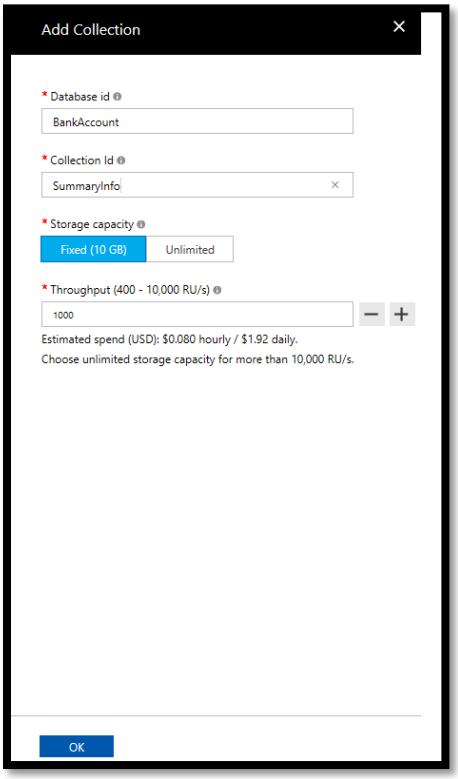
- Once page is open, Enter **ID**
- Select **API** as **SQL**
- Select **Subscription**
- Create/Select **Resource Group**
- Select **Location**
- Select **Pin to Dashboard**
- Click **Create**



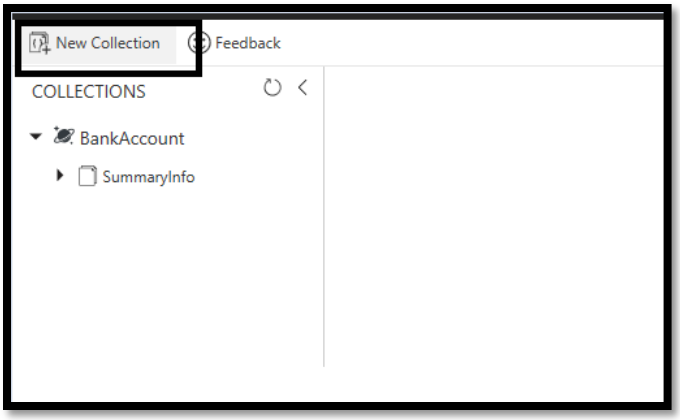
- Once page is open, click **Overview**
- Click **Add Collection**



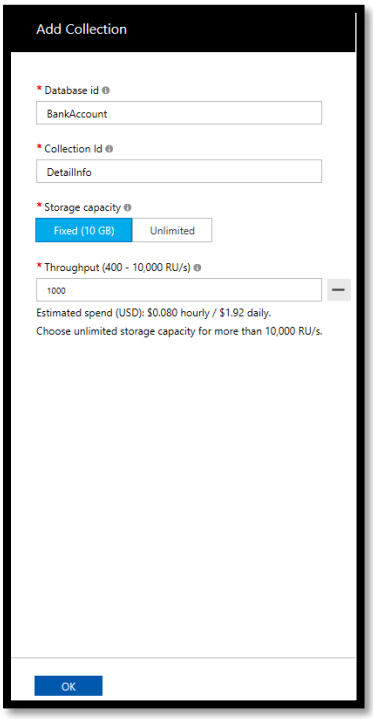
- Let's create Database and a collection to store account master information. Provide **Database id**
- Provide **Collection Id**
- Select **Storage Capacity** as **Fixed (10 GB)**
- Click **OK**



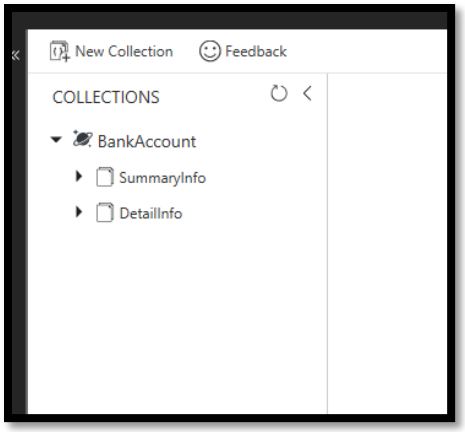
- Once Collection created successfully, Click **New Collection**



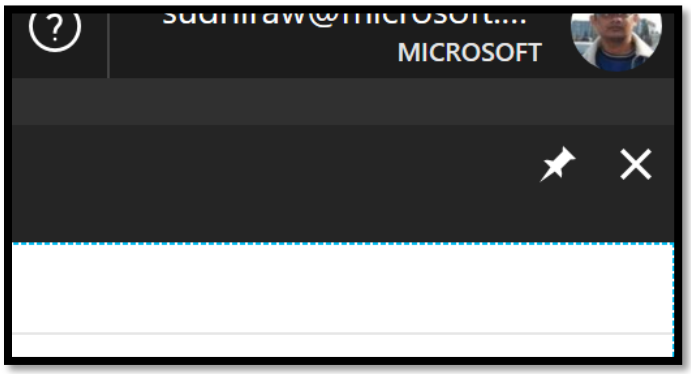
- Enter **Database id**



- Enter **Collection Id**
- Select **Storage capacity**
- Select **Throughput**
- Click **OK**
- Once Database and collection created, screen will look like below



- Click **Close** (**X**)



<u>Adding Account Master Details</u>

- Goto https://github.com/rawatsudhir1/AzureCosmosDBChangeFeedUseCase and either clone the repo or download it as zip
- In the repo under SupportingFiles, open CustomerAccountInfo.txt file. This file has some records or json document (contains customer account master information) which we will upload through portal. The other approach is to create a web-app or mobile app (with proper security enabled) to send this data.

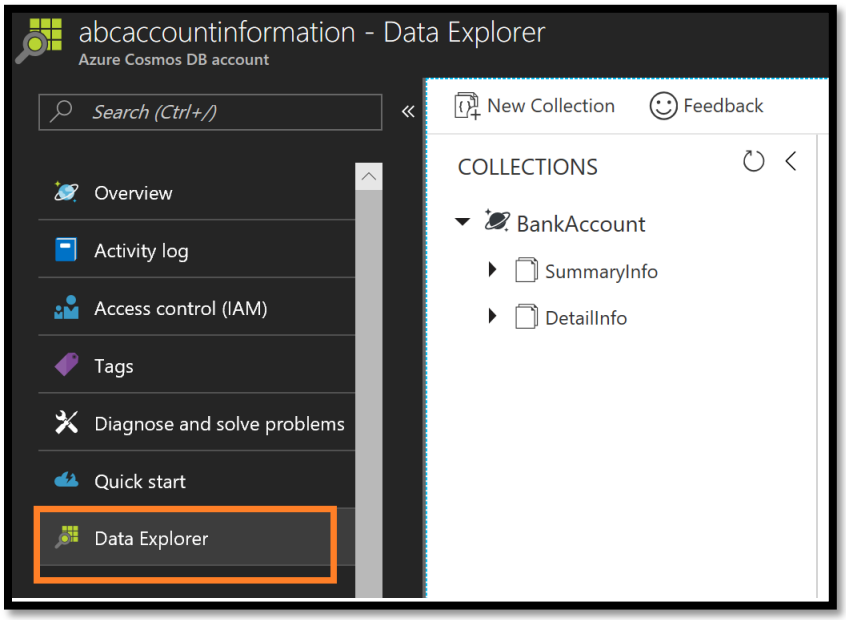
rawatsudhir1 Add files via upload

1 contributor

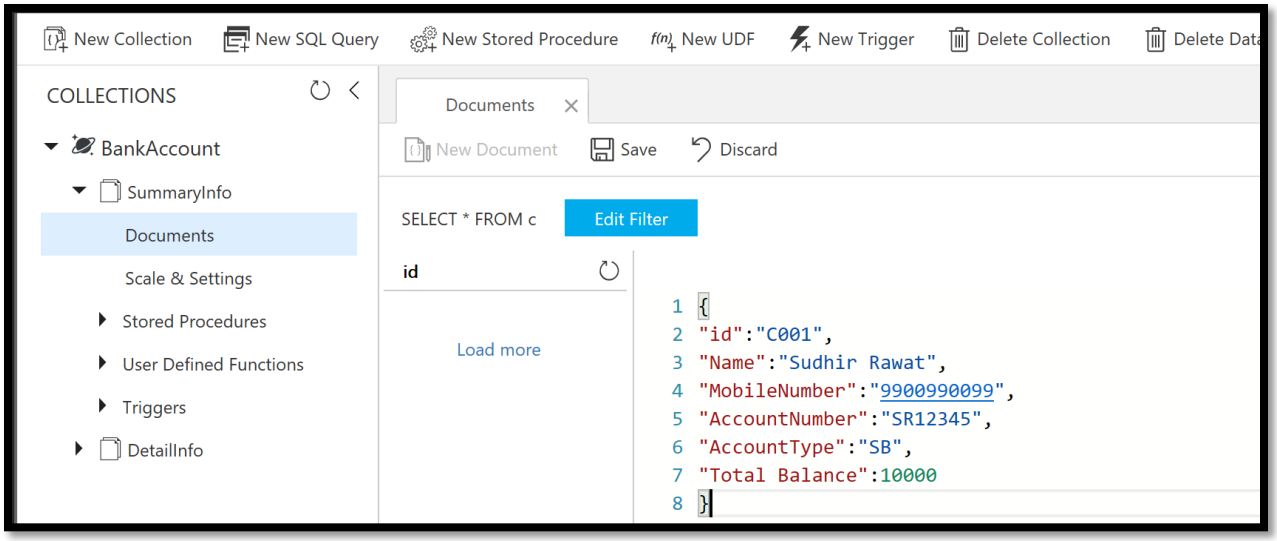
54 lines (48 sloc) | 890 Bytes

```
1  {
2  "id":"C001",
3  "Name":"Sudhir Rawat",
4  "MobileNumber":"9900990099",
5  "AccountNumber":"SR12345",
6  "AccountType":"SB",
7  "Total Balance":10000
8  }
9
10 {
11 "id":"C002",
12 "Name":"Bindu C",
13 "MobileNumber":"9911000022",
14 "AccountNumber":"BC67890",
15 "AccountType":"SB",
16 "Total Balance":50000
17 }
```

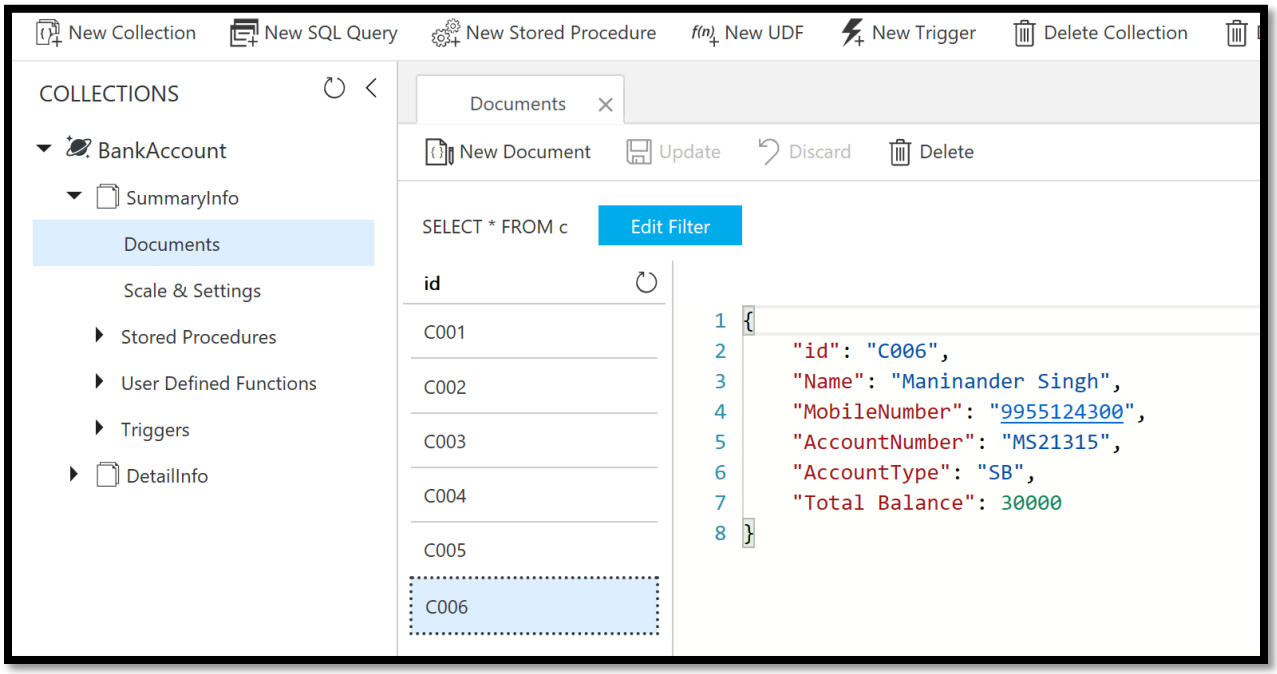
- Copy first record from the file (from line 1 to line 8 as per the above image)
- Goto **Azure CosmosDB account**, created in earlier step.
- Click **Data Explorer**



- Click **SummaryInfo**
- Click **Documents**
- Click **New Document**
- Paste the record copied from github in earlier step



- Click **Save**
- Repeat Steps to add more records (copy from github repo)
- After adding all six records, here is how screen look like

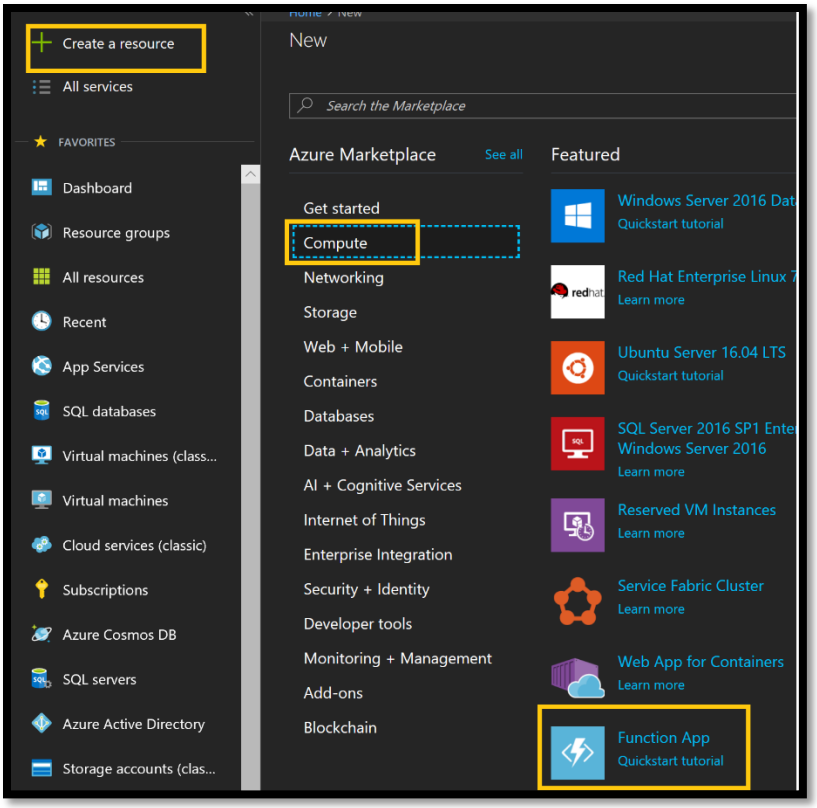


- Close (**X**) **Data Explorer**

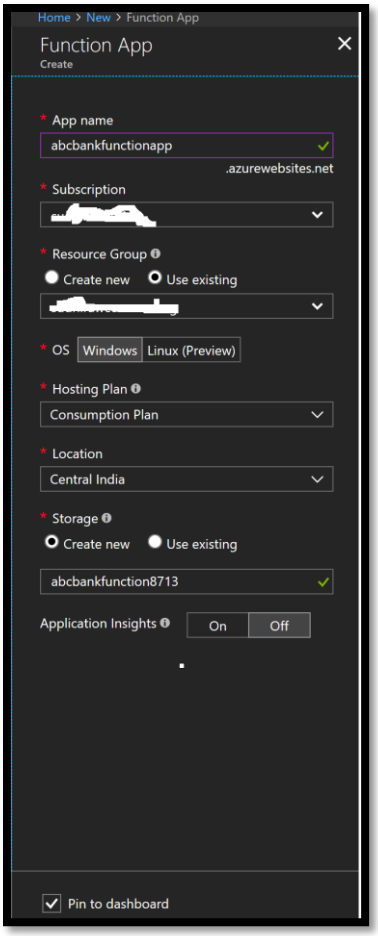
Before recording deposit or expense details in **DetailInfo** collection, let's first build out **Azure Function Logic**

<u>Building Azure function logic</u>

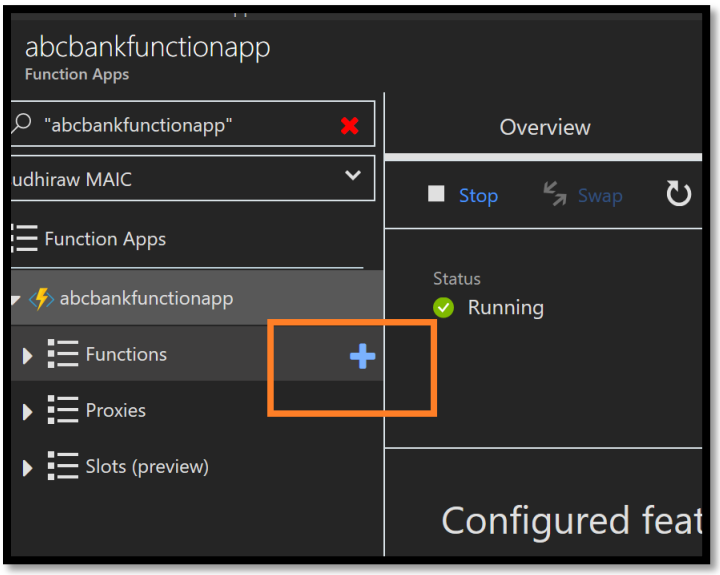
- Click **Create a Resource**
- Click **Compute**
- Click **Function App**



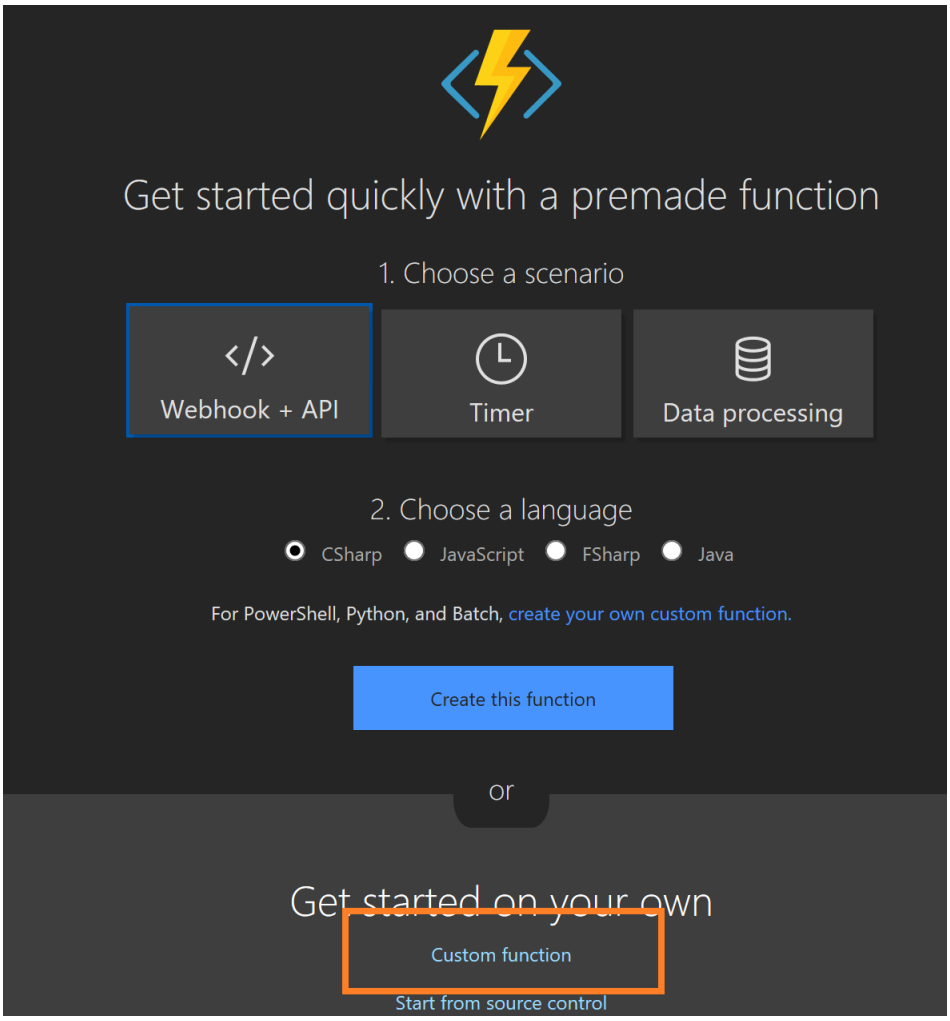
- Once blade is open, Enter **App name**
- Select **Subscription**
- **Create new** or **Use existing Resource Group**
- Select Windows as **OS**
- Select **Consumption Plan** as **Hosting Plan**
- Select **Location** (Select the same region where Azure CosmosDB is created)
- Let default values for **Storage**
- Click **Pin to dashboard**
- Click **Create**



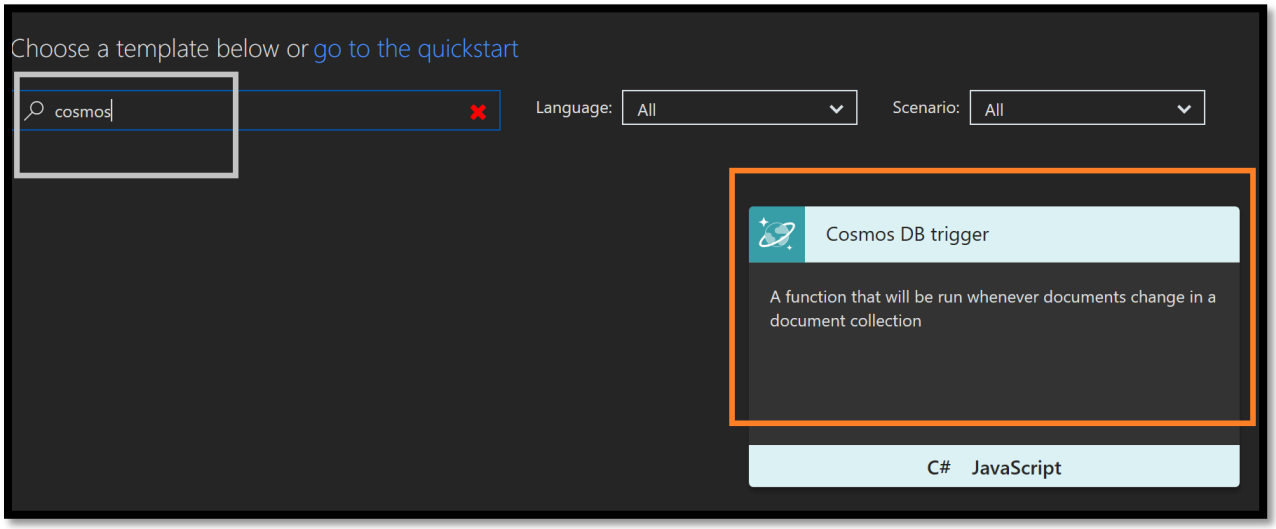
- Once Function App is created, Click on **New** (+)



- Click **Custom Function**

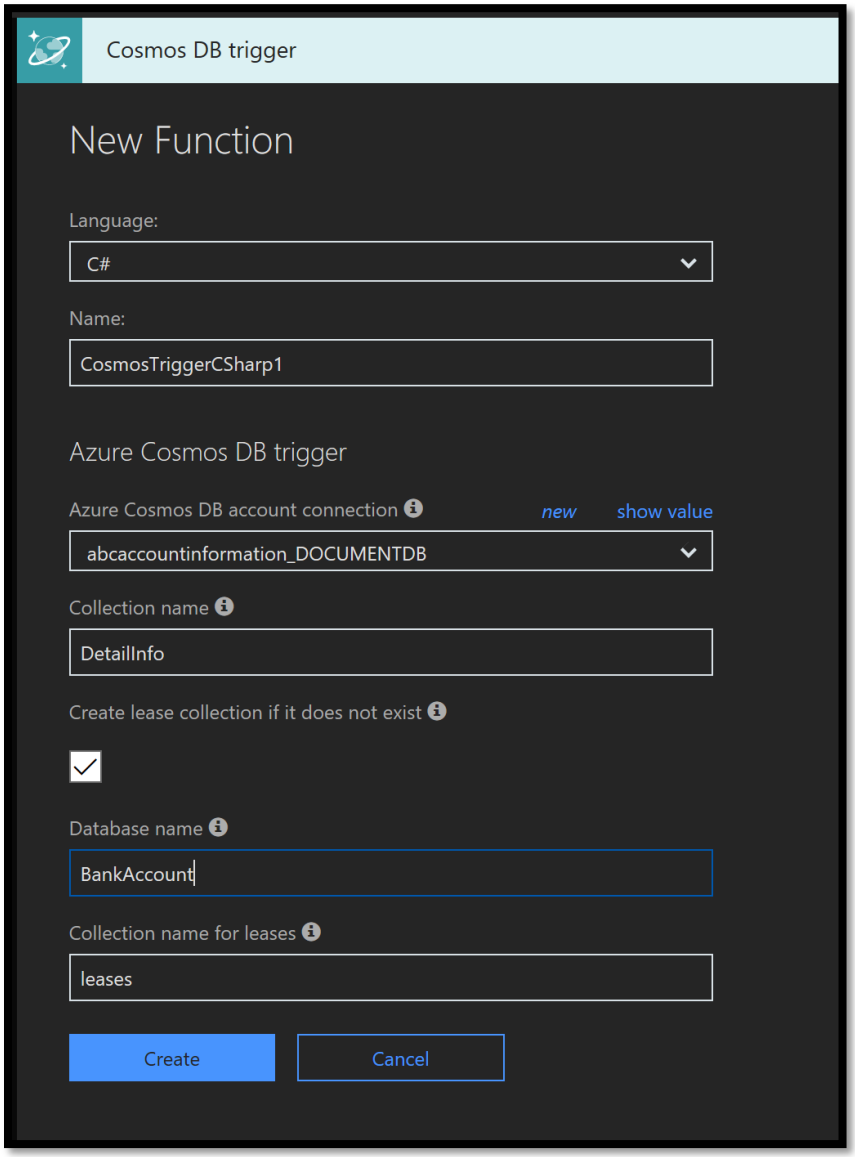


- Choose a template, enter cosmos in search

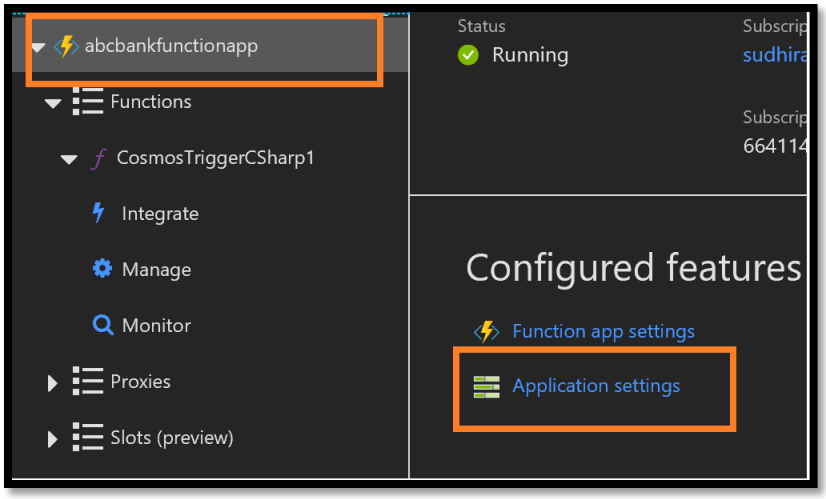


- Select **Cosmos DB trigger**

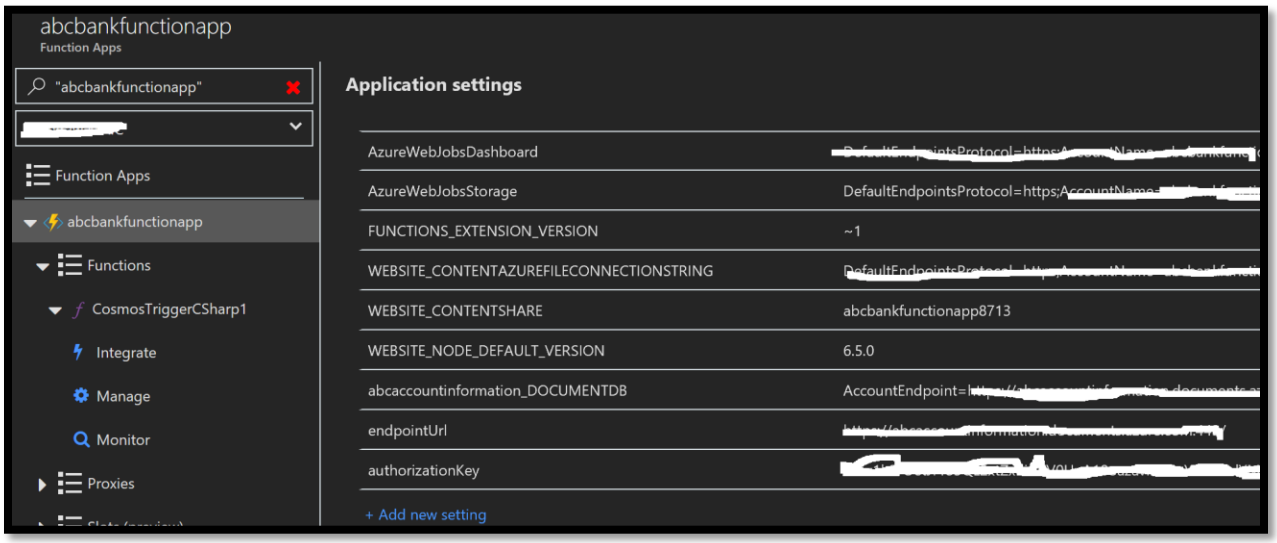
- On New Function, Select **C#** as **language**
- Provide **Name** of the function
- On **Azure Cosmos DB account connection** click **new** and select the cosmodb account created earlier
- Provide **Collection name**
- Provide **Database name**



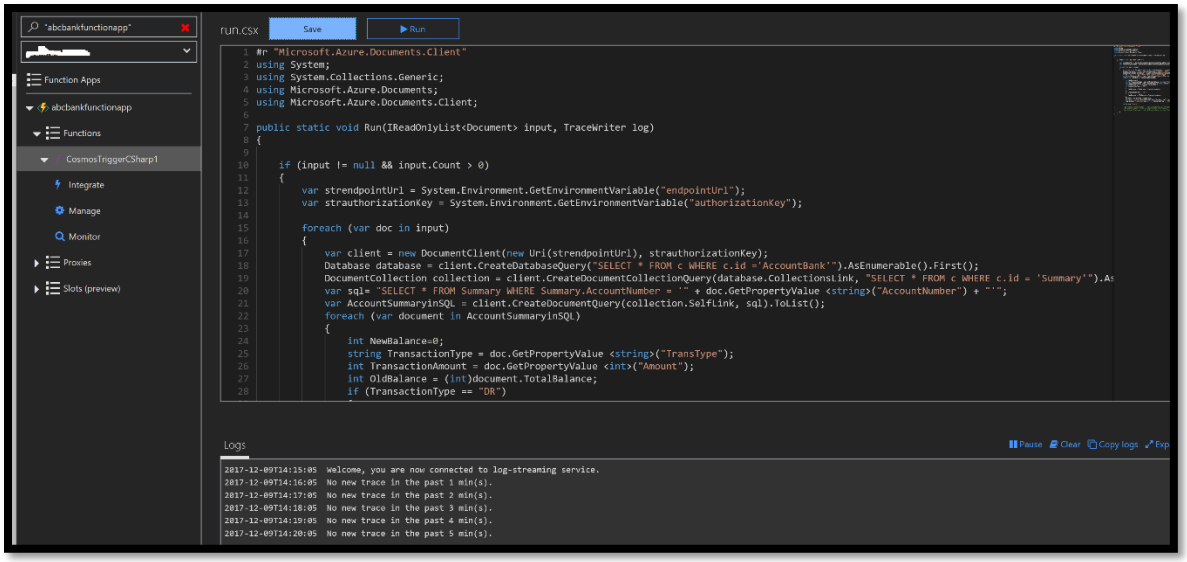
- Let **Collection name for leases** as it is
- Click **Create**
- Once function is created, Click on function app and then Application Setting



- Under Application settings, add two variables endpointUrl and authorizationKey. These variables hold values to connect CosmosDB to retrieve and update Bank Account Master table. Copy values for both the variable from CosmosDB account under **Keys** section.
- Add endpointUrl , authorizationKey with values and **Save** it. This is how it will look like.



- Click on function (if you choose default name then it will be **CosmosTriggerCSharp1**)
- Copy code from https://github.com/rawatsudhir1/AzureCosmosDBChangeFeedUseCase/tree/master/AzureFunctionCode and paste it in function. Click **Save**



- Azure function is setup. Let's move to next step

Post transaction

- Open Visual Studio 2017, create a console application
- Copy code from https://github.com/rawatsudhir1/AzureCosmosDBChangeFeedUseCase/tree/master/PostTransaction (**Program.cs**) and paste it.
- Add endpointurl and authorizationKey asshown below.

```
class Program
{
    private static readonly string endpointUrl = "XXXXX";
    private static readonly string authorizationKey = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    private static readonly string databaseId = "BankAccount";
    private static readonly string collectionId = "DetailInfo";

    0 references
    public static void Main(string[] args)
```

- Run/F5 to run console application. This action will post a transaction to Azure CosmosDB

Output at Azure Function

- Switch to Azure portal
- Look at **Logs** window in function

```
18          DocumentCollection collection = client.CreateDocumentCollectionQuery(database.CollectionsLink,
19          var sql= "SELECT * FROM SummaryInfo WHERE SummaryInfo.AccountNumber = '" + doc.GetPropertyVal
20          var AccountSummaryinSQL = client.CreateDocumentQuery(collection.SelfLink, sql).ToList();
21          foreach (var document in AccountSummaryinSQL)
```

Logs Pause Clear Copy logs

```
2017-12-10T06:29:01.106 Function started (Id=23e3324a-36a9-4fe3-9b19-e95e9f53c786)
2017-12-10T06:29:01.674 Transction type is DR Old balance was 50000. Transaction Amount is 200 New Balance is 49800
2017-12-10T06:29:01.674 Function completed (Success, Id=23e3324a-36a9-4fe3-9b19-e95e9f53c786, Duration=565ms)
```

Building PowerBI Report

- If time permits, try https://docs.microsoft.com/en-us/azure/cosmos-db/powerbi-visualize

Send Notification to user

- SMS :- https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-twilio
- Email :- https://docs.microsoft.com/en-us/azure/azure-functions/functions-how-to-use-sendgrid