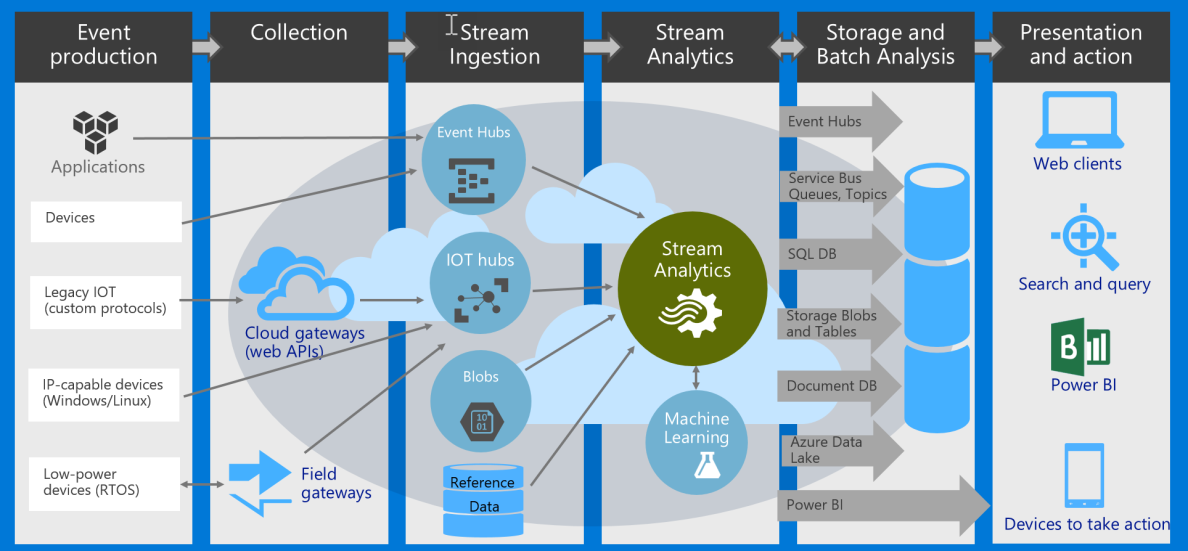
**Setup Guide for FT900 Microsoft Azure IoT demo**

**I. MICROSOFT AZURE IoT Architecture**

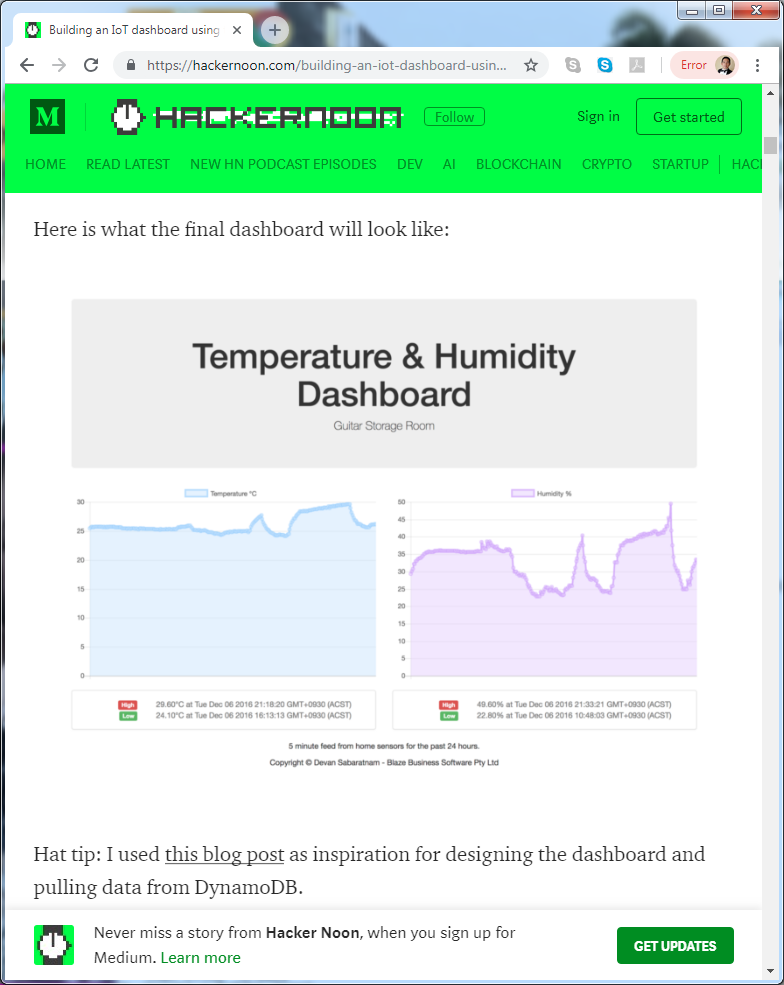
****

BACK-END**: FT900 ->** **IoT Hub -> Stream Analytics -> CosmosDB (aka DocumentDB)**

FRONT-END: **Browser -> (Dashboard webpage in) Storage -> CosmosDB 🡨 [NG: TODO]**

TODO: Integrate the provided Node.JS script to the sample AWS IoT Dashboard at

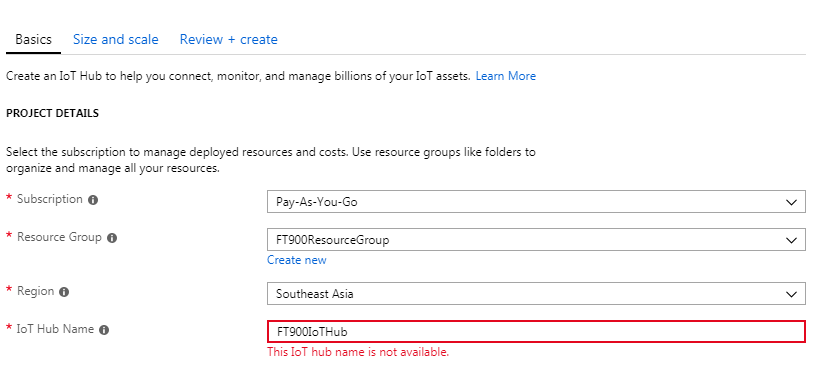
<https://hackernoon.com/building-an-iot-dashboard-using-the-onion-omega-and-amazon-aws-a3520f850c9>

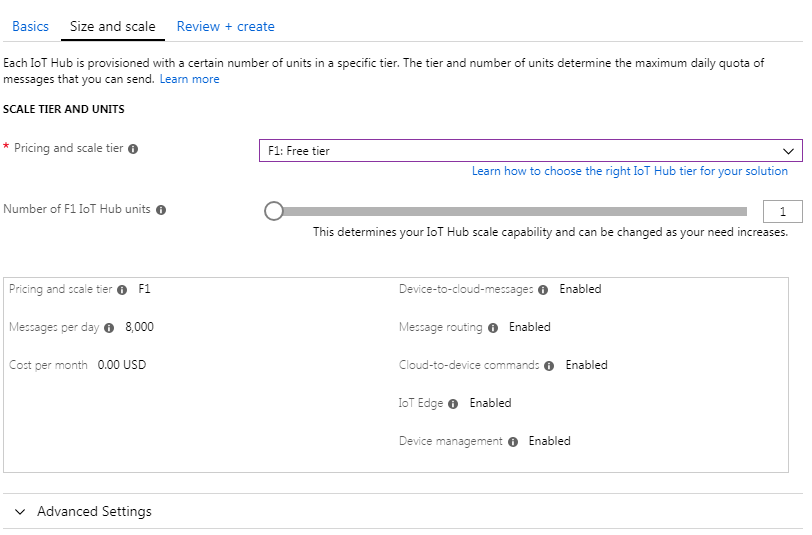


**II. BACKEND CLOUD Setup Guide:**

1. Create an IoT Hub resource ‘FT900IoTHub’

Create a resource > Internet of Things > IoT Hub





1. Create a Device (under IoT Hub) called ‘ft900device1’ – will use SAS token authentication

IoT Hub > IoT devices > Add

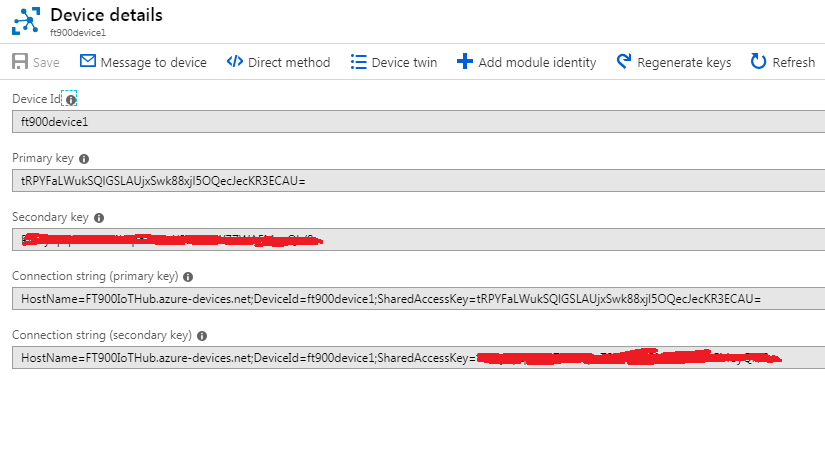
Create Device1 using the following details:

DeviceID: ft900device1

Authentication type: Symmetric Key

Auto-generate keys: YES

Click Save

Device 1 will use SAS token authentication. Copy the generated Primary Key and update ft900device1\_sas\_azure.pem 

1. Create a Device (under IoT Hub) called ‘ft900device2’ – X509 certificate authentication

Create Device2 using the following details:

DeviceID: ft900device2

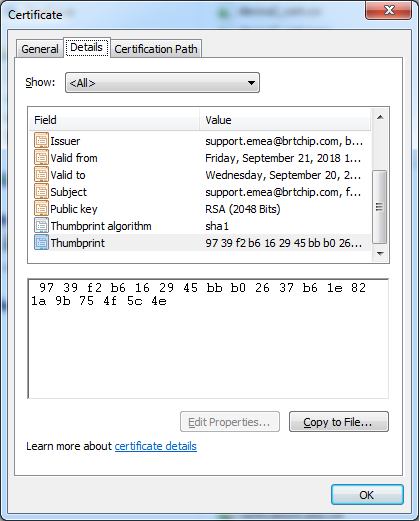
Authentication type: x.509 Self-Signed

Primary Thumbprint: 9739f2b6162945bbb02637b61e821a9b754f5c4e

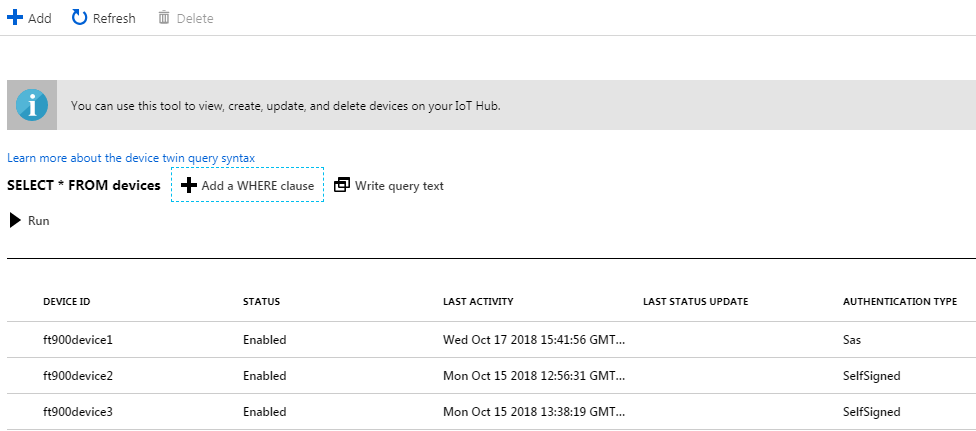
Secondary Thumbprint: 9739f2b6162945bbb02637b61e821a9b754f5c4e

Click Save

Device 2 will use X509 certificate authentication.The thumbprint above is retrieved using the certificate hash of ft900device2\_cert.pem.

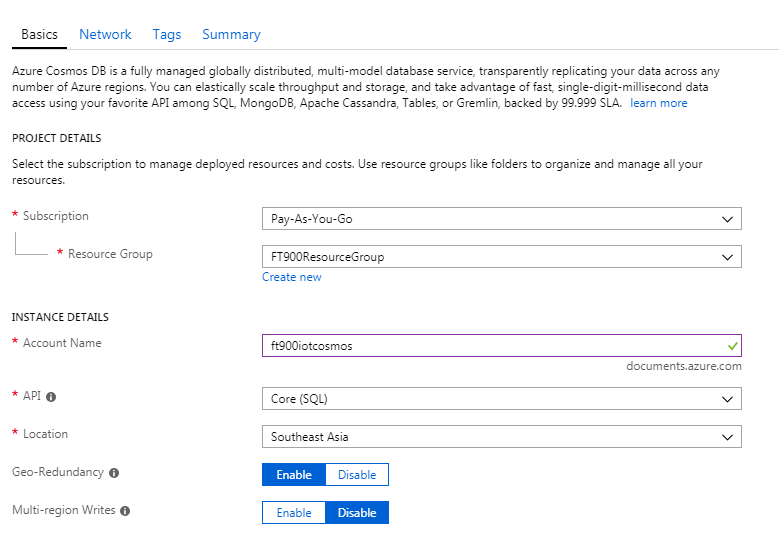


After creating device1 and device2, it should appear as below

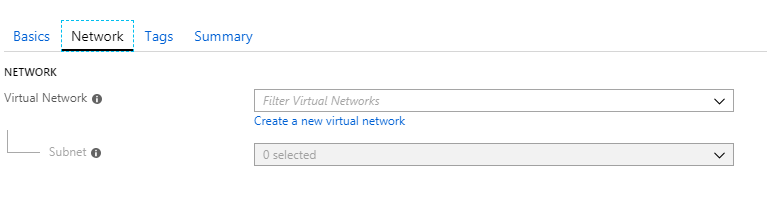


1. Create a CosmosDB resource ‘ft900iotcosmosdb’

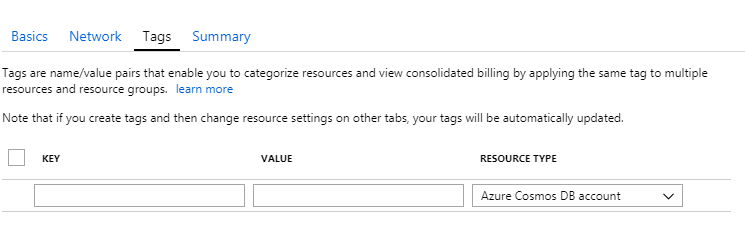
Create a resource > Databases > Azure Cosmos DB



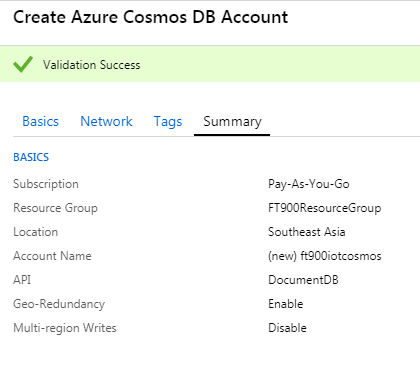
Click “Next: Network” button



Click “Next: Tags” button



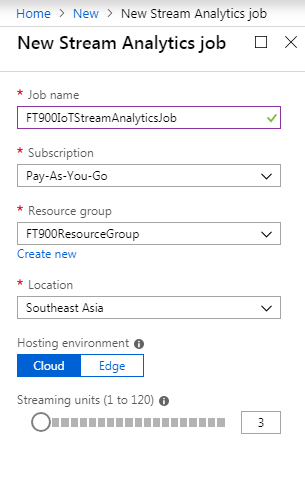
Click “Next: Summary” button



Click “Create” button

1. Create a Stream Analytics resource ‘FT900IoTStreamAnalytics’

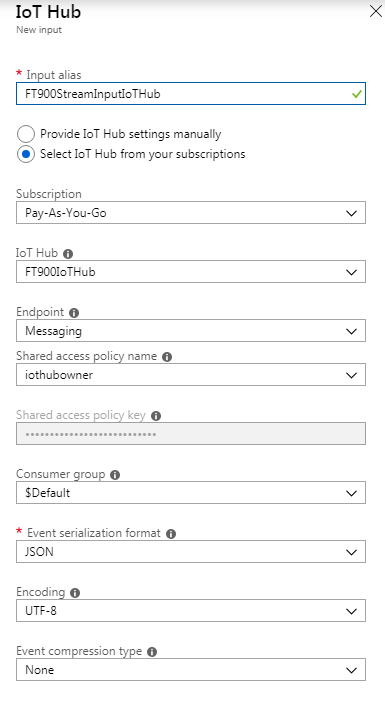
Create a resource >Analytics > Stream Analytics job



Click “Create” button.

Create the **Job Input**

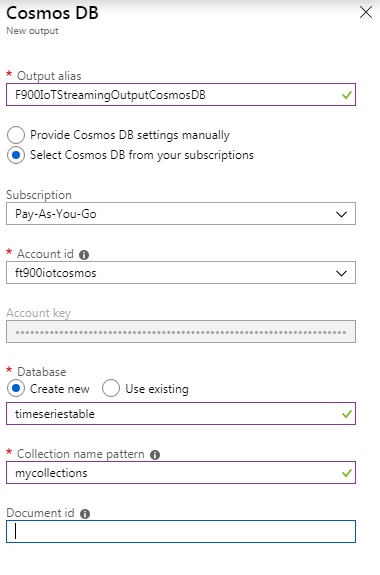
Go to Job topology > Inputs > Add stream input > IoT Hub



Click “Save” button

Create the **Job Output**

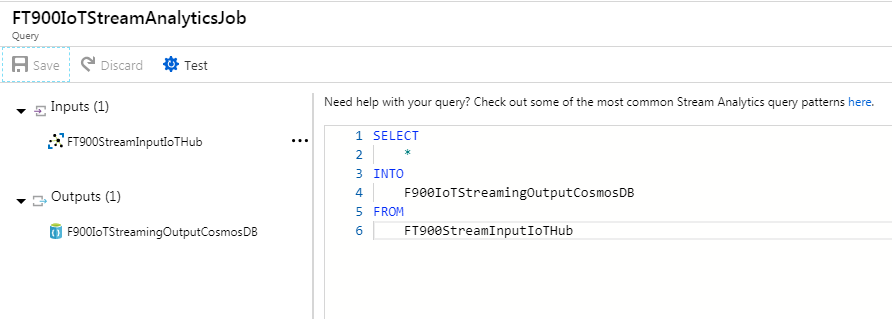
Go to Job topology > Outputs > Add > Cosmos DB



Click “Save” button

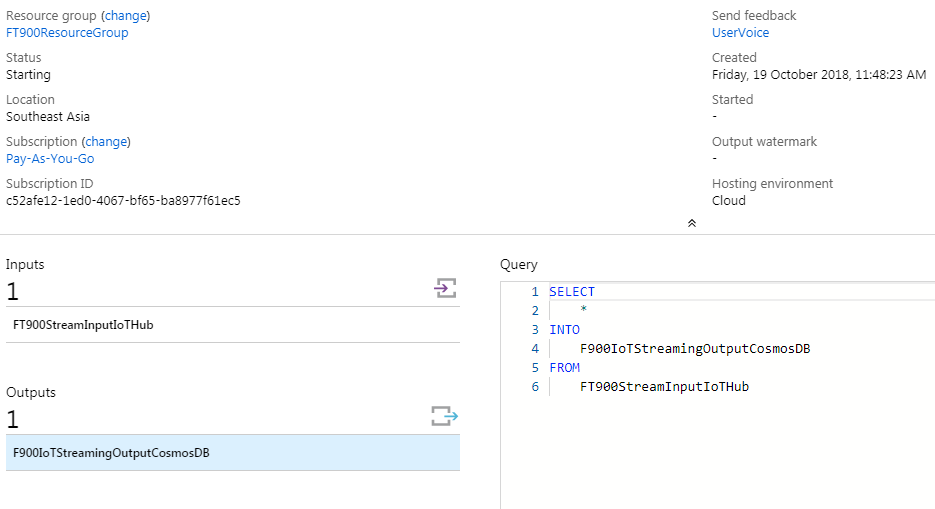
Update the Query

Go to Job topology > Query > Add > Cosmos DB



Click “Save” button

Check that it should look like this below



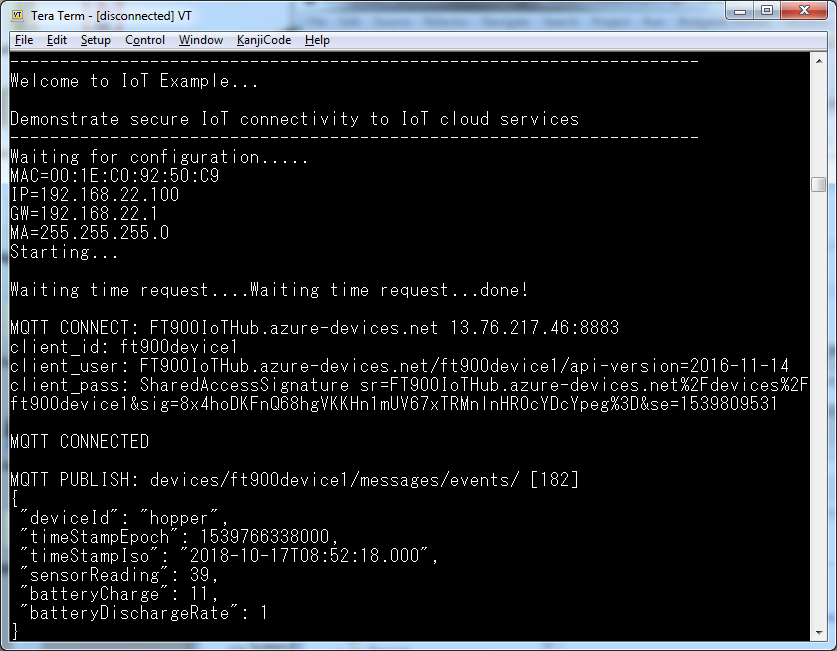
Click “Start” button > Start

Wait for the Streaming job to start successfully!

1. Verify FT900 MQTT messages are saved in Cosmos DB.

1. Run FT900 IoT Demo for 1 minute.

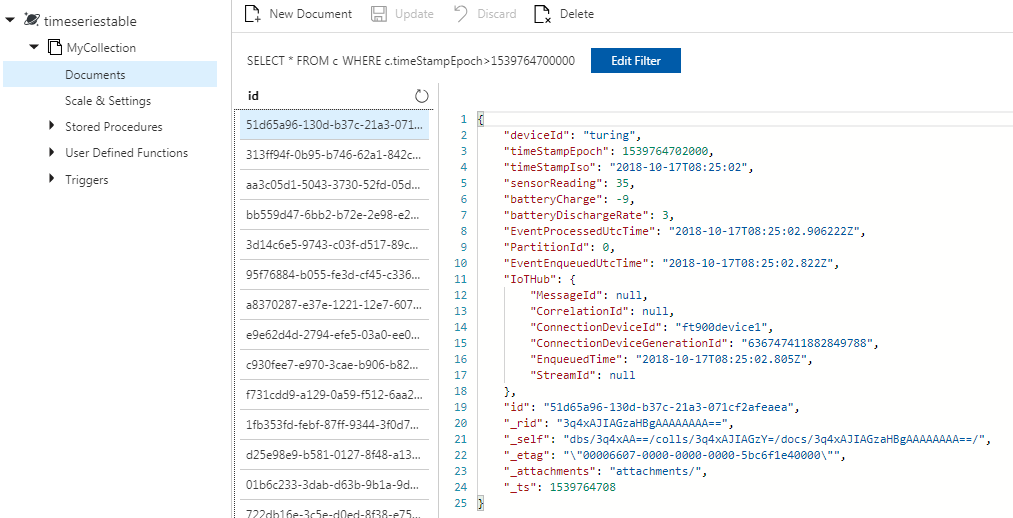
(make sure MQTT\_BROKER\_MAZ\_IOT and SAMPLE\_DEVICE\_1 is selected)



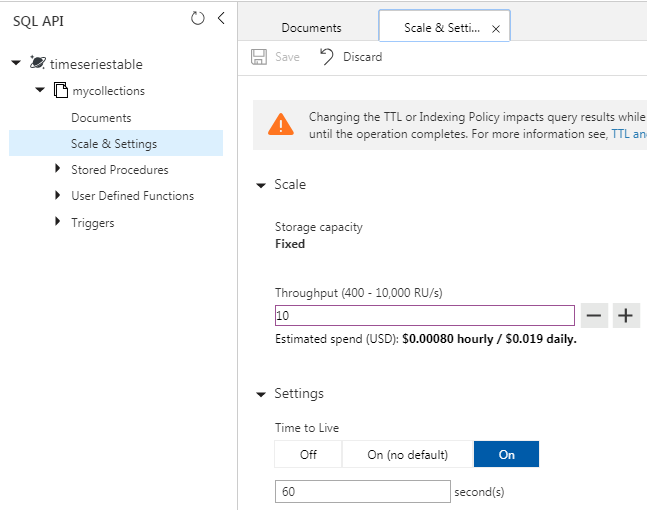
2. Go to CosmosDB > Document Explorer > Open Data Explorer

and run the following query “SELECT \* FROM c”

Verify that the data appears as screenshot below.



3. To save database memory, change the TTL Time to Live option to something like 60 seconds (1 minute), or depending on the dashboard use-case.



**III. FRONTEND CLOUD Setup Guide:**

A. Install the prerequisites

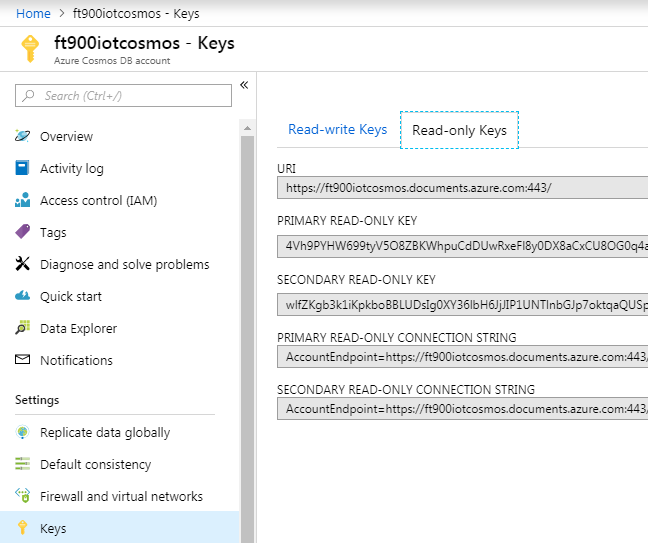
1. NodeJS

2. Azure Cosmos API

npm install async –save

npm install @azure/cosmos

B. Get the access in key CosmosDB



C. Update and run the Node.JS script “cosmosdbclient.js”

1. Update the values for

endpoint = "https://ft900iotcosmos.documents.azure.com:443/";

masterKey = "4Vh9PYHW699tyV5O8ZBKWhpuCdDUwRxeFl8y0DX8aCxCU8OG0q4aaAKDLB8FL5BzEfR4ZV2xe5eTwIIc5fRiqA==";

databaseId = "timeseriestable";

containerId = "mycollections";

2. Run FT90x for 1-2 minutes

3. Run cosmosdbclient.js using “node cosmosdbclient.js”

