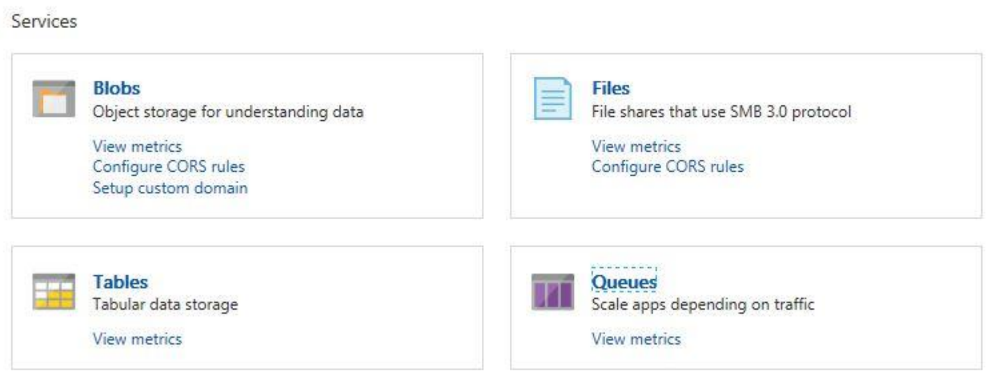
Exercise. Create a serverless function

In this exercise we will create a simple serverless function that reads messages from a queue and puts them in a table.   To do this first create a new storage account or use an existing one.   Go to the storage account page and you will see:



Services

Blobs
Object storage for understanding data
View metrics
Configure CORS rules
Setup custom domain

Files
File shares that use SMB 3.0 protocol
View metrics
Configure CORS rules

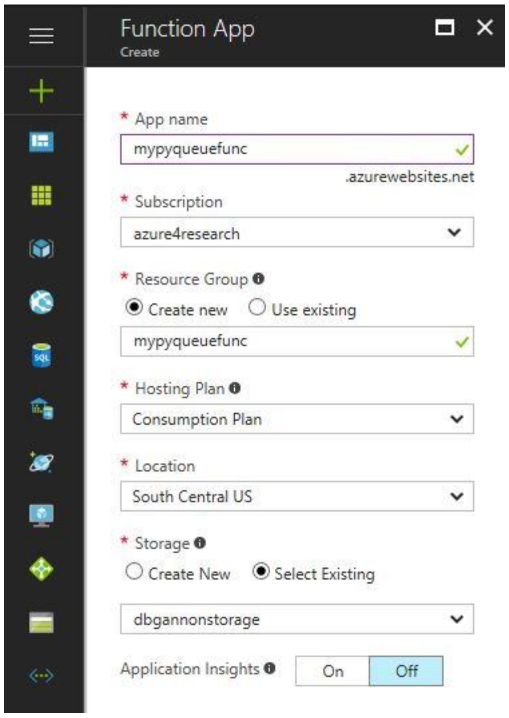
Tables
Tabular data storage
View metrics

Queues
Scale apps depending on traffic
View metrics

Click on Table and create a new table and remember its name. Then go to Queues and create a new one and remember its name. Open the storage explorer and look for these item. Clicking on the table name should give you a picture of an empty table.

Create the function.

Go back to the portal main page and click + and look for "Function App" and click create.   There is a table to fill in like the one on the right. Give it a name and allow it to create a new resource group.   For the storage you want to use the dropdown and look for your storage account name.   (If your storage account is not in "south central" you can change that location.   If it refuses to give you a function there, then you may need to create a new storage account in south central.   The important thing is to align this with your storage account.



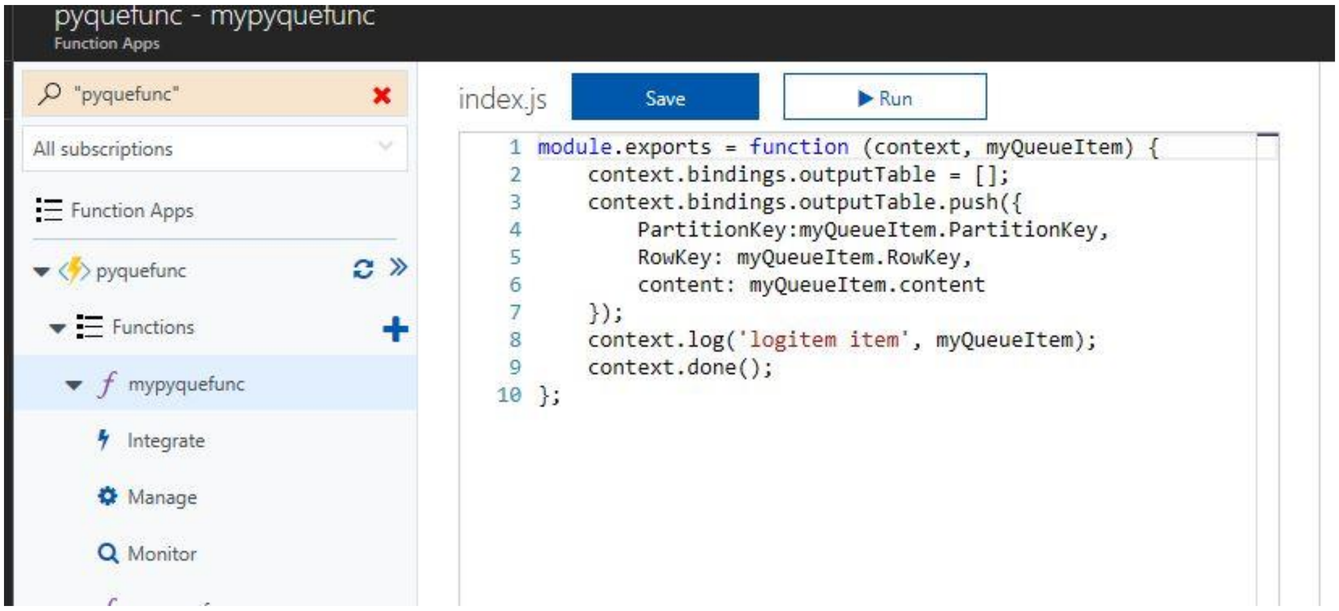
Function App
Create

* App name	mypyqueuefunc .azurewebsites.net
* Subscription	azure4research
* Resource Group	[✓ Create new] Use existing — mypyqueuefunc
* Hosting Plan	Consumption Plan
* Location	South Central US
* Storage	Create New [✓ Select Existing] — dbgannonstorage
Application Insights	On [✓ Off]

Click create and wait for the function to appear on your portal home.

You should see it in your resource groups. Follow that link and you will see that it is running.

Go to the functions tab and hit "+".   It will ask you to pick one of the templates.   At the top where it says "language" select Javascript and pick the one that is called QueueTrigger. This will load a basic template for the function.   Now edit the template so that it looks like the following example.



```
module.exports = function (context, myQueueItem) {
    context.bindings.outputTable = [];
    context.bindings.outputTable.push({
        PartitionKey:myQueueItem.PartitionKey,
        RowKey: myQueueItem.RowKey,
        content: myQueueItem.content
    });
    context.log('logitem item', myQueueItem);
    context.done();
};
```

The main difference is that we have added an output table and instructions to push three items into the table. The function is assuming that the items in the queue are of the form

{'PartitionKey': 'part1', 'RowKey': '73', 'content': 'some data '}

Next we need to tie the queue to our queue and the table to our table. So click on "Integrate" on the left. You need to fill in the form so that it ties it to your stuff as illustrated below

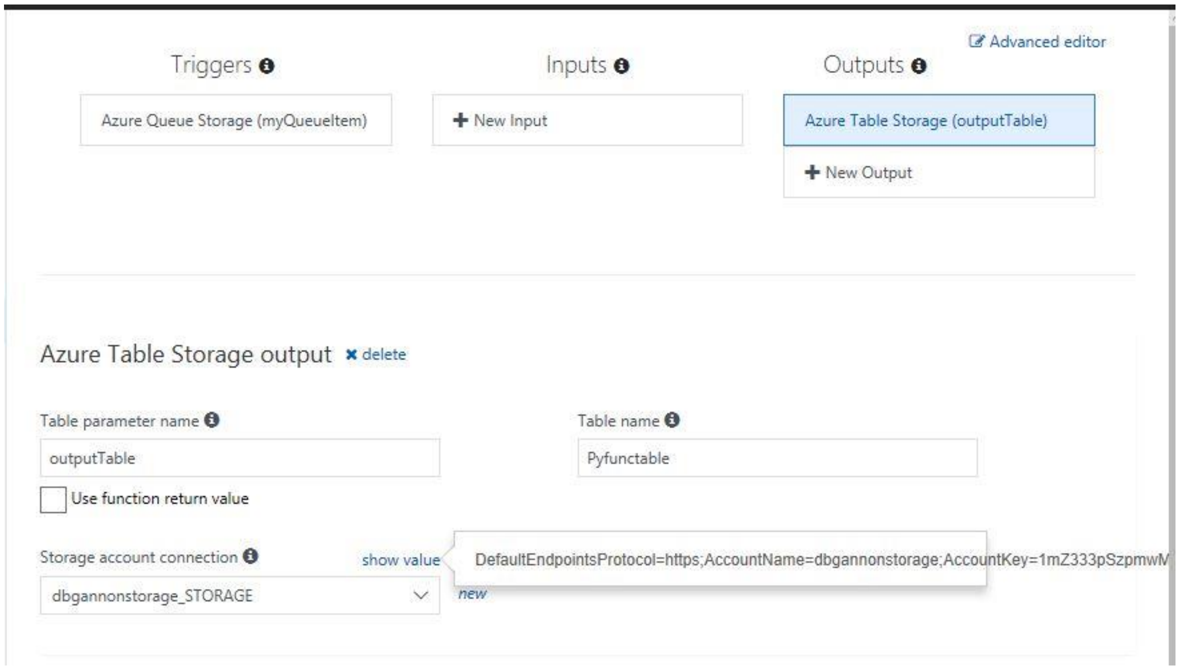

Advanced editor
Triggers
Inputs
Outputs
Azure Queue Storage (myQueueItem)
New Input
Azure Table Storage (outputTable)
New Output
Azure Table Storage output
delete
show value
new

Table parameter name: outputTable

Table name: Pyfunctable

Use function return value

Storage account connection: dbgannonstorage_STORAGE

DefaultEndpointsProtocol=https;AccountName=dbgannonstorage;AccountKey=1mZ333pSzpmwM

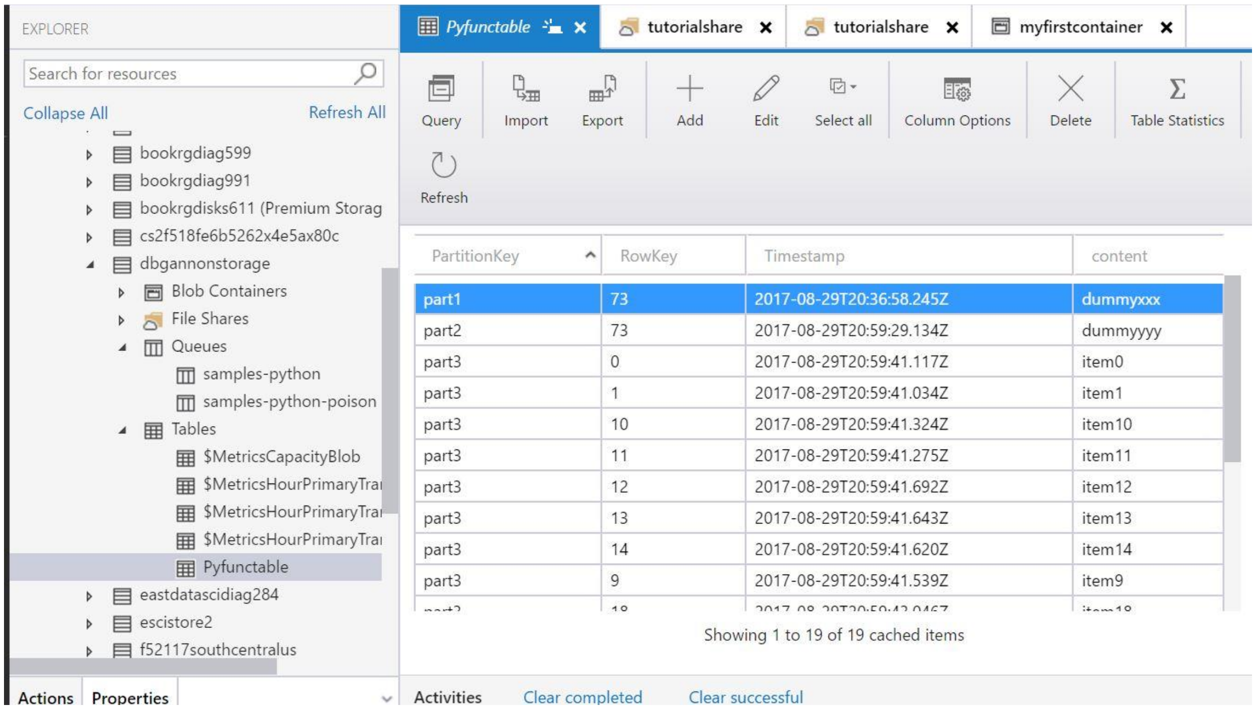
You should see your storage account in the dropdown menu. Select it. And they add the Table name. You need to do the same for the AzureQueueStorage.

Once this is done and your function is saved and the system is running your function should be instantiated and invoked as soon as you send it queue items. For that we have a simple python script in a Jupyter notebook.

Login to your datasci vm and do

```
$ cd notebooks
$ wget https://SciEngCloud.github.io/py-functions-queue-driver.ipynb
```

You will need to fill in your account key for the storage account, but then you should be able to step through the rest. Your storage account should look like this:



PartitionKey	RowKey	Timestamp	content
part1	73	2017-08-29T20:36:58.245Z	dummyxxx
part2	73	2017-08-29T20:59:29.134Z	dummyyyy
part3	0	2017-08-29T20:59:41.117Z	item0
part3	1	2017-08-29T20:59:41.034Z	item1
part3	10	2017-08-29T20:59:41.324Z	item10
part3	11	2017-08-29T20:59:41.275Z	item11
part3	12	2017-08-29T20:59:41.692Z	item12
part3	13	2017-08-29T20:59:41.643Z	item13
part3	14	2017-08-29T20:59:41.620Z	item14
part3	9	2017-08-29T20:59:41.539Z	item9

Showing 1 to 19 of 19 cached items