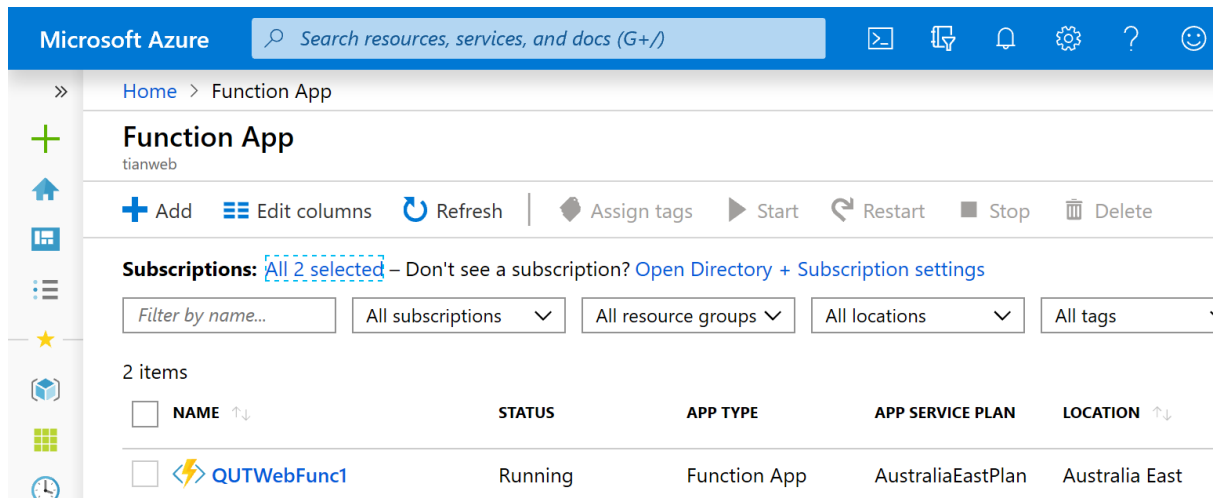
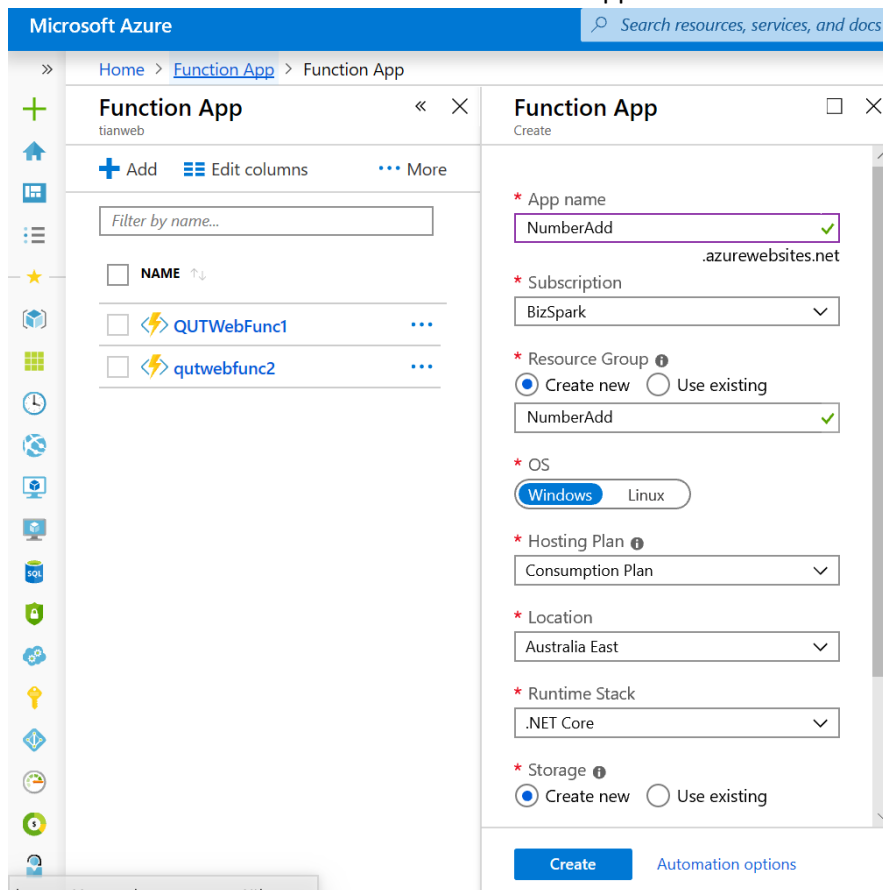


## Function App

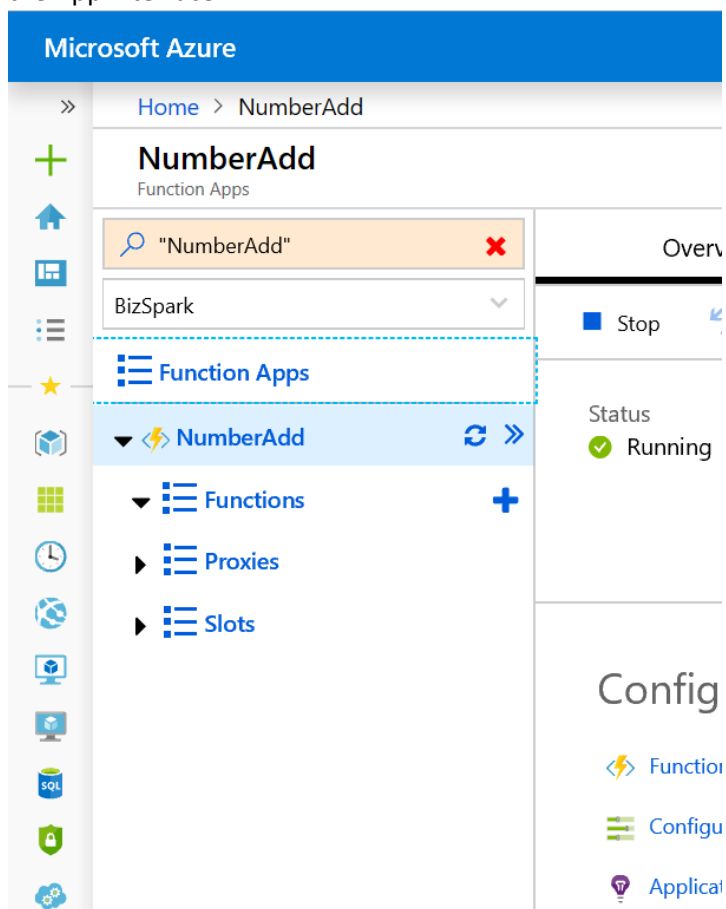
1. In the search field in the Azure portal , type Function App. This will bring up the Function Add command



2. Click on the + Add and fill the form to create the app

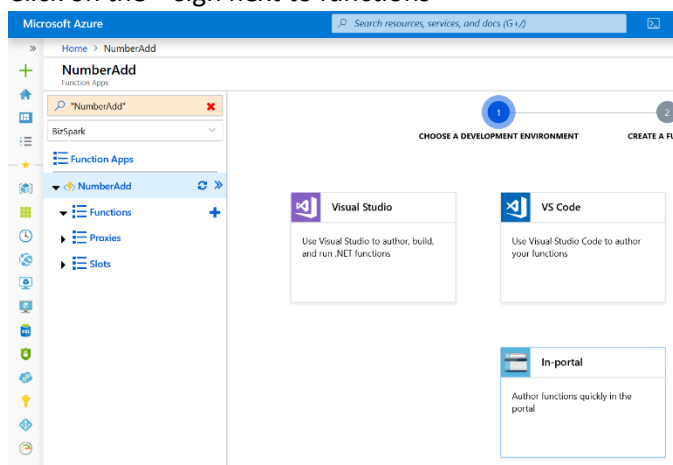


- It will take a few minutes to be created. Once done it will provide a message to navigate to the App interface.



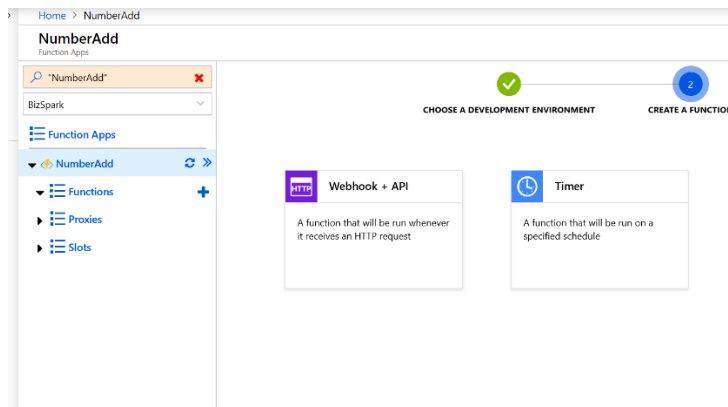
- Two functions need to be created.
  - DataReceiver :An http triggered function
  - QueueProcessor : A queue triggered function.
- DataReceiver

Click on the + sign next to functions

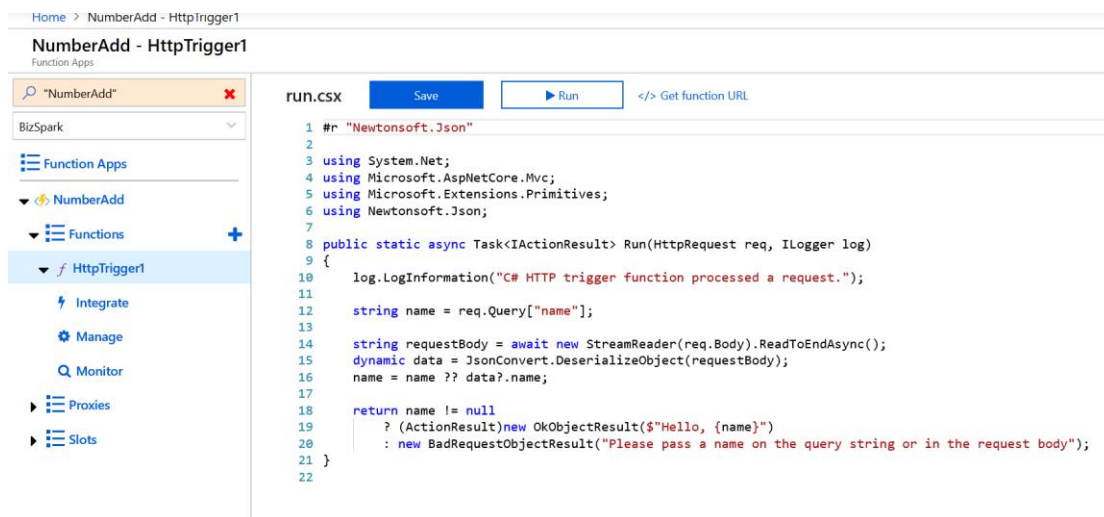


It will present different way to create the function. Select In Portal

And then web hook



The function will be created with simple code that will respond with hello. To try it out copy the url by clicking on get function url



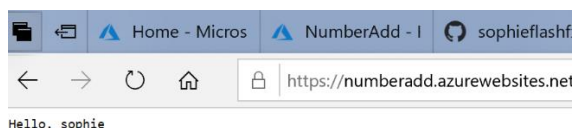
The url will be like the following with a unique value at the end as shown below

<https://numberadd.azurewebsites.net/api/HttpTrigger1?code=353ssddf32.....>

Add to it the query param as follows

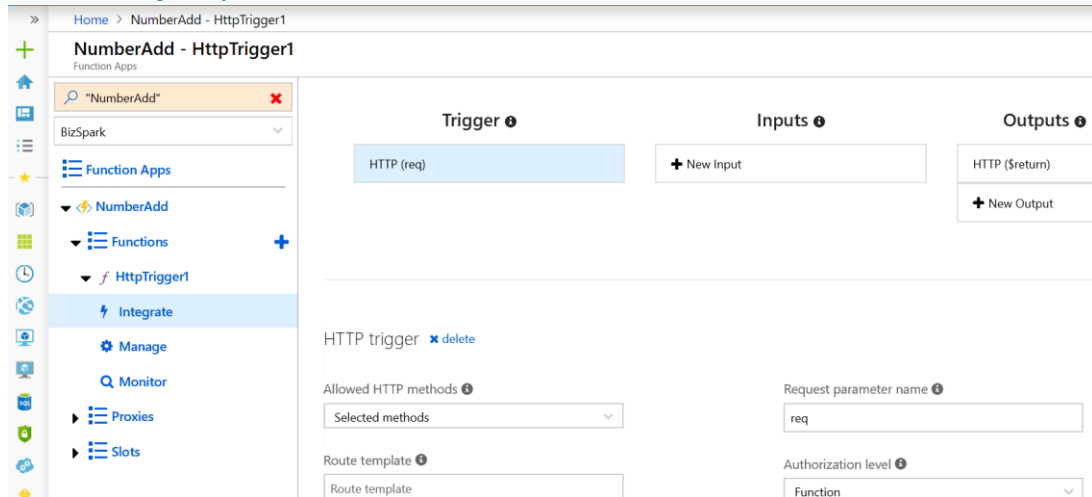
<https://numberadd.azurewebsites.net/api/HttpTrigger1?code=353ssddf32.....&name=sophie>

paste the url into a browser and call the function. The function will respond with Hello, sophie

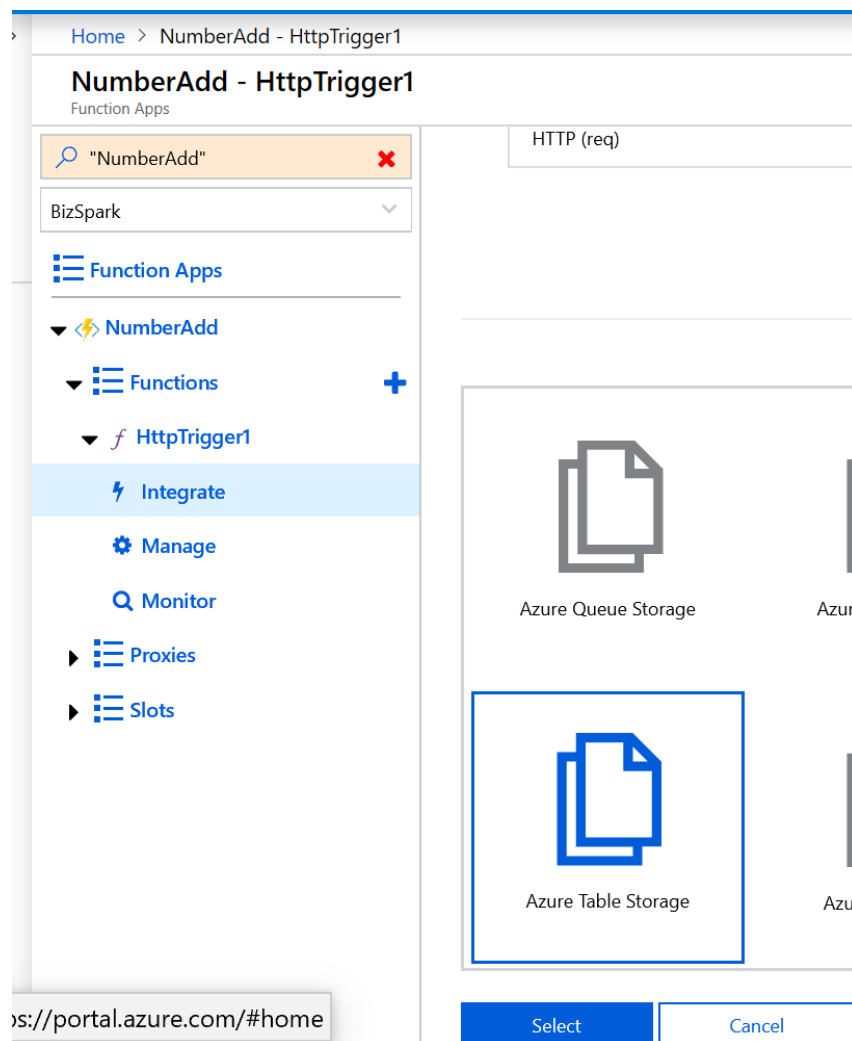


Next a Azure storage account needs to be created and integrated with the function. Do this as follows.

6. Click on integrate just below the function name



- 7 Click on +New Outputs and then Azure Table storage.



It will give a warning extension not installed. Click on install

Function Apps

- NumberAdd
- Functions
  - HttpTrigger1
- Integrate**
- Manage
- Monitor
- Proxies
- Slots

Azure table Storage output

Extensions not Installed

This integration requires the following extensions.

- Microsoft.Azure.WebJobs.Extensions.Storage

Install

Table parameter name ⓘ

outputTable

☐ Use function return value

Storage account connection ⓘ [show value](#)

AzureWebJobsStorage [new](#)

Save Cancel

The following screen results

BizSpark

Function Apps

- NumberAdd
- Functions
  - HttpTrigger1
- Integrate**
- Manage
- Monitor
- Proxies
- Slots

+ New Output

Azure Table Storage output

Installing template dependencies. Dependency installation happens in the background and can take up to 2 minutes. Your app will go into offline mode until the installation is complete. You can continue to use the portal during this time, but closing the portal before the installation is complete will leave your app in offline mode

Table parameter name ⓘ

outputTable

☐ Use function return value

Storage account connection ⓘ [show value](#)

AzureWebJobsStorage [new](#)

Table name ⓘ

outTable

Save Cancel

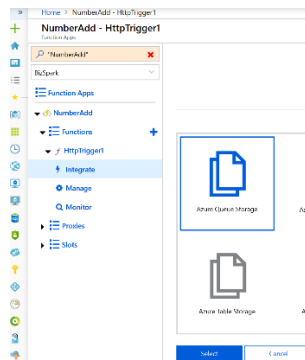
Once its done click save. A Azure table called outTable will be created and integrated with the app.

Next input integration to the same table is needed. Click integrate and this time choose + New Input

Select Azure table as before. For the table name it will have inTable. However we need to use the same table that was created for output. As such replace the inTable with outTable and click save.

Finally, a queue needs to be created.

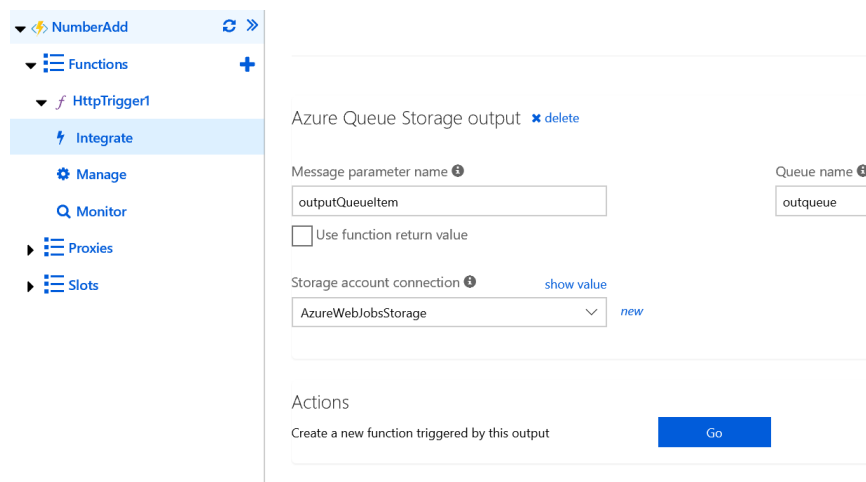
Click integrate and then + New output and select Azure Queue Storage



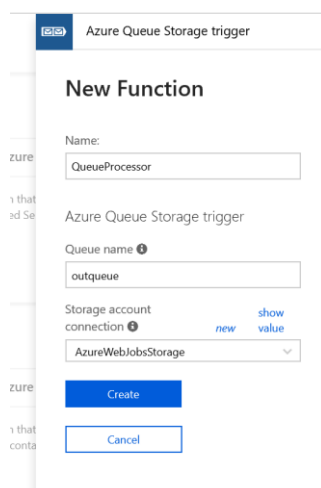
Keep the default setting and click save.

Now all the integrations with storage has been completed.

When an entry is made in the Queue we want it to trigger the QueueProcessor function. The interface Action provides the ability to create this function at this stage. Click on Go



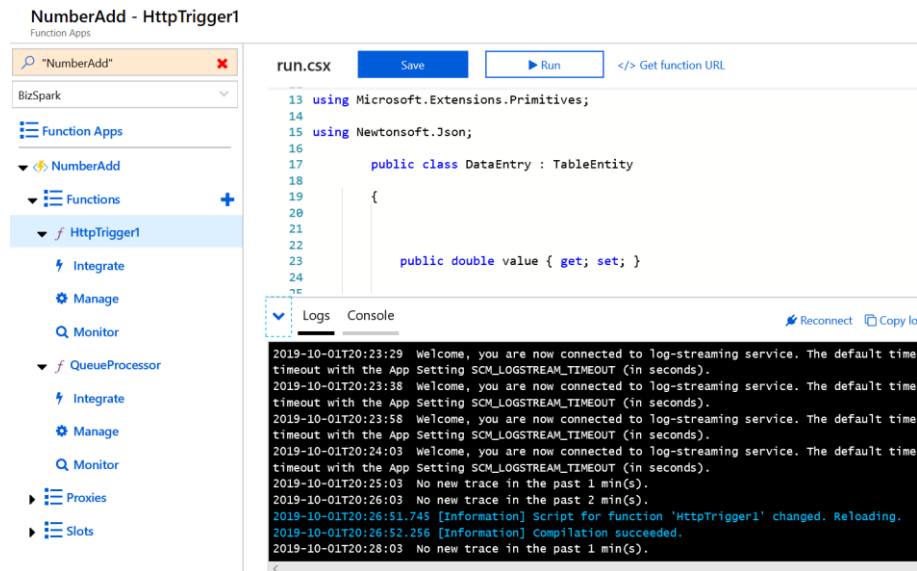
And then select Azure Queue Storage trigger. Set the name of the function to be QueueProcessor and the name of the queue to be outqueue



Open the code in the http trigger function and replace the code with the one on this page

<https://github.com/sophieflashfx/azureserverless/blob/master/datareceiver.csx>

Pressing save will compile and save the function. Clicking on log will show if there are any compilation errors



NumberAdd - HttpTrigger1

Function Apps

BizSpark

Function Apps

NumberAdd

Functions

HttpTrigger1

Integrate

Manage

Monitor

QueueProcessor

Integrate

Manage

Monitor

Proxies

Slots

run.csx

Save

Run

</> Get function URL

```
13 using Microsoft.Extensions.Primitives;
14
15 using Newtonsoft.Json;
16
17 public class DataEntry : TableEntity
18 {
19
20
21
22
23     public double value { get; set; }
24
25 }
```

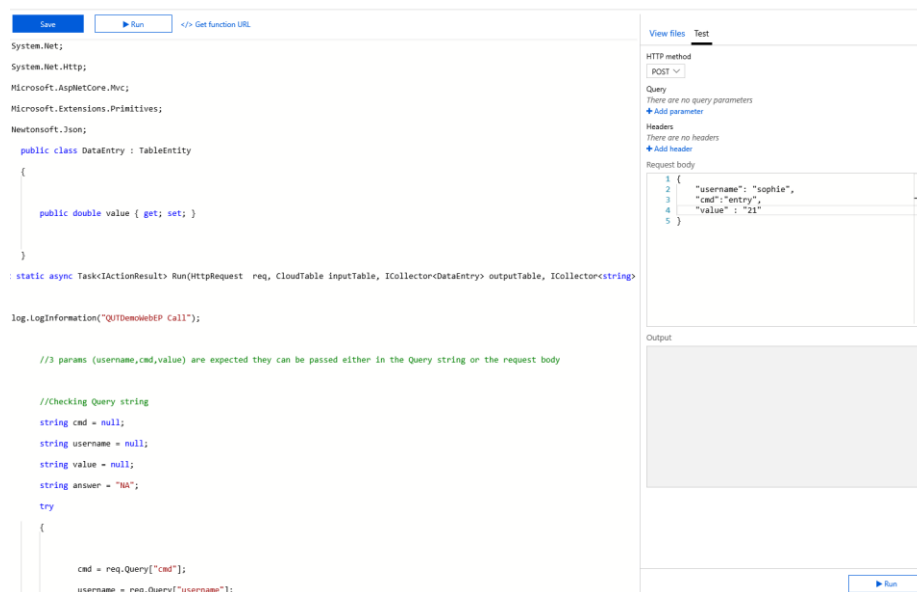
Logs Console

Reconnect Copy log

2019-10-01T20:23:29 Welcome, you are now connected to log-streaming service. The default timeo  
timeout with the App Setting SCM\_LOGSTREAM\_TIMEOUT (in seconds).  
2019-10-01T20:23:38 Welcome, you are now connected to log-streaming service. The default timeo  
timeout with the App Setting SCM\_LOGSTREAM\_TIMEOUT (in seconds).  
2019-10-01T20:23:58 Welcome, you are now connected to log-streaming service. The default timeo  
timeout with the App Setting SCM\_LOGSTREAM\_TIMEOUT (in seconds).  
2019-10-01T20:24:03 Welcome, you are now connected to log-streaming service. The default timeo  
timeout with the App Setting SCM\_LOGSTREAM\_TIMEOUT (in seconds).  
2019-10-01T20:25:03 No new trace in the past 1 min(s).  
2019-10-01T20:26:03 No new trace in the past 2 min(s).  
2019-10-01T20:26:51.745 [Information] Script for function 'HttpTrigger1' changed. Reloading.  
2019-10-01T20:26:52.256 [Information] compilation succeeded.  
2019-10-01T20:28:03 No new trace in the past 1 min(s).

The function can be tested by using the test values. Replace the default body with the following

```
{
  "username": "sophie",
  "cmd": "entry",
  "value": "21"
}
```



Save

Run

</> Get function URL

System.Net;

System.Net.Http;

Microsoft.AspNetCore.Mvc;

Microsoft.Extensions.Primitives;

Newtonsoft.Json;

```
public class DataEntry : TableEntity
{
    public double value { get; set; }
}

static async Task Run(HttpRequest req, CloudTable inputTable, ICollector<DataEntry> outputTable, ICollector<string>)
```

Log.LogInformation("QUTDemokabEP Call");

/\*\*/3 params (username,cmd,value) are expected they can be passed either in the Query string or the request body

/\*\*Checking Query string

```
string cmd = null;
string username = null;
string value = null;
string answer = "NA";
try
{
    cmd = req.Query["cmd"];
    username = req.Query["username"];
```

View files Test

HTTP method

POST

Query

There are no query parameters

Add parameter

Headers

There are no headers

Add header

Request body

```
1 {
2   "username": "sophie",
3   "cmd": "entry",
4   "value": "21"
5 }
```

Output

Run

If all is ok you will get the status 200 ok and the response

"Request Recieved username:sophie,cmd:entry,value:21 output answer =NA"

The screenshot displays a REST client interface with two tabs: "View files" and "Test". The "Test" tab is active, showing the configuration for an HTTP POST request. The "HTTP method" is set to "POST". The "Query" section indicates "There are no query parameters" with a link to "Add parameter". The "Headers" section indicates "There are no headers" with a link to "Add header". The "Request body" is a JSON object: 

```
{  "username": "sophie",  "cmd": "entry",  "value": "21"}
```

. Below the request configuration, the "Output" section shows the response status as "Status: 200 OK" and the response body as "Request Recieved username:sophie,cmd:entry,value:21 output answer =NA".

View files Test

HTTP method  
POST

Query  
There are no query parameters  
[+ Add parameter](#)

Headers  
There are no headers  
[+ Add header](#)

Request body

```
1 {  
2   "username": "sophie",  
3   "cmd": "entry",  
4   "value" : "21"  
5 }
```

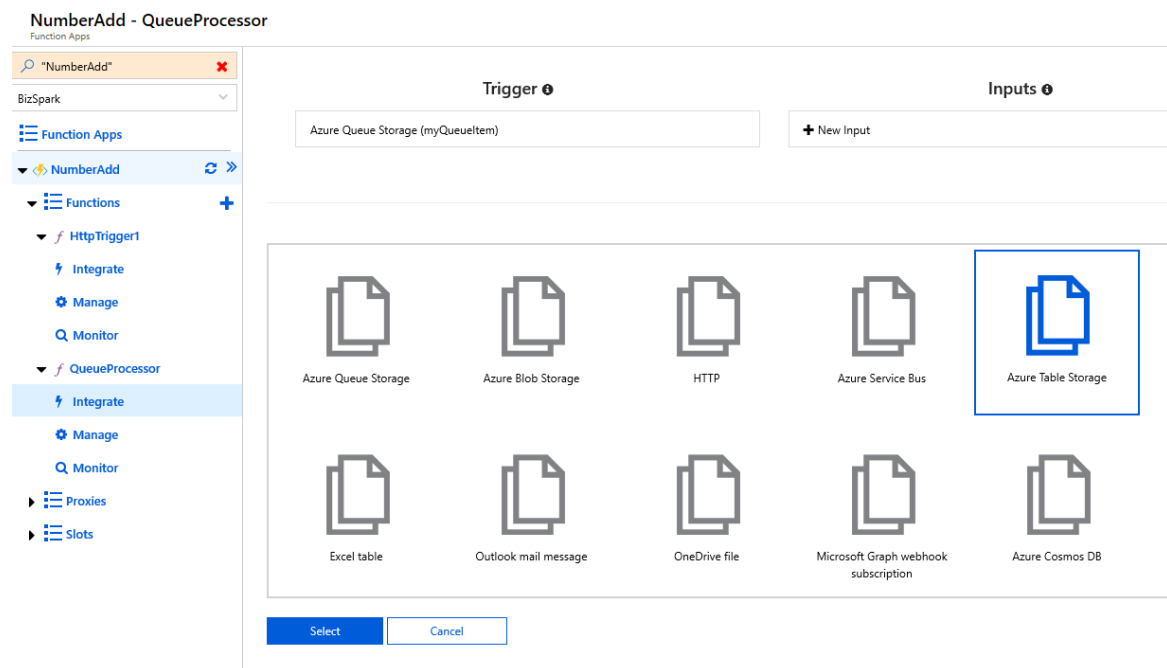
Output ✔ Status: 200 OK

Request Recieved username:sophie,cmd:entry,value:21 output answer =NA



## QueueProcessor

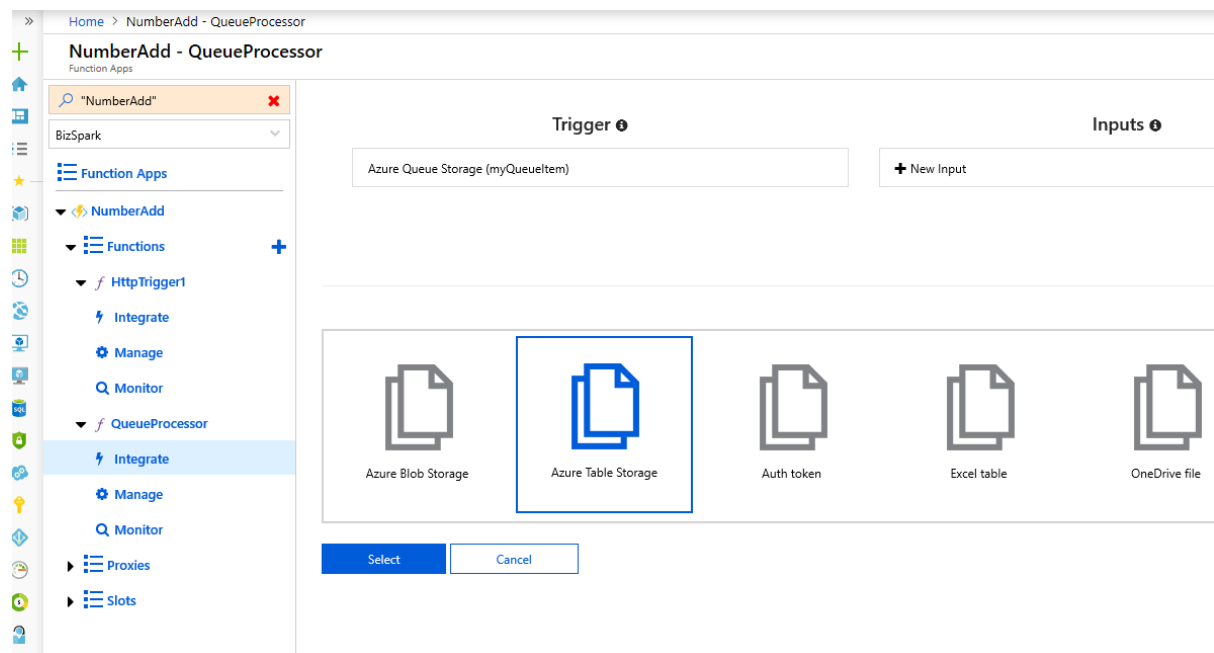
1. Next we need to integrate the QueueProcessor with the Azure Table as well. To do this click on integrate and then + New Output and select Azure Table storage.



Make sure the table name is outTable and click save.

3. The function needs to be also integrated for input from the table.

To do this click integrate and +New Input then Azure Table / select and change the table name to outTable and save.



Finally replace the code of the function with the code from this page

<https://github.com/sophieflashfx/azureserverless/blob/master/queueprocessor.csx>

The user Interface

The static html page for the app will be saved in an azue blob and served from it. To do this follow

The instruction on this page

<https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-static-website>.

The code for the index.html page is given on this link

<https://github.com/sophieflashfx/azureserverless/blob/master/index.html>