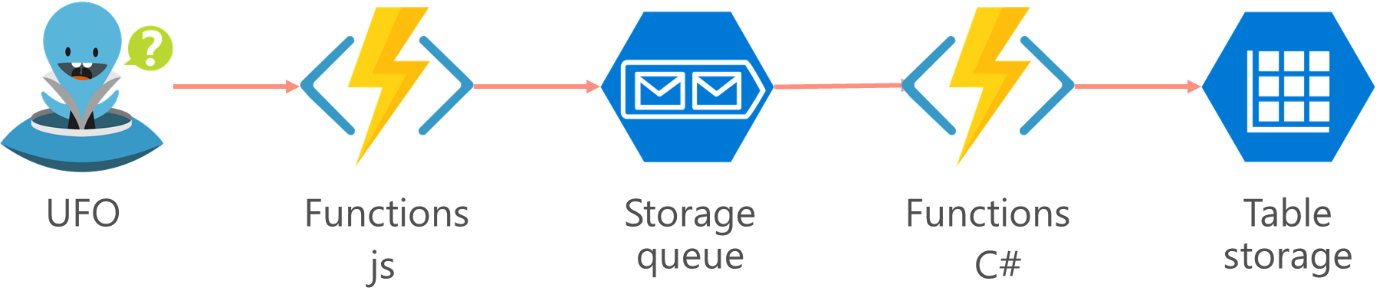
# Description

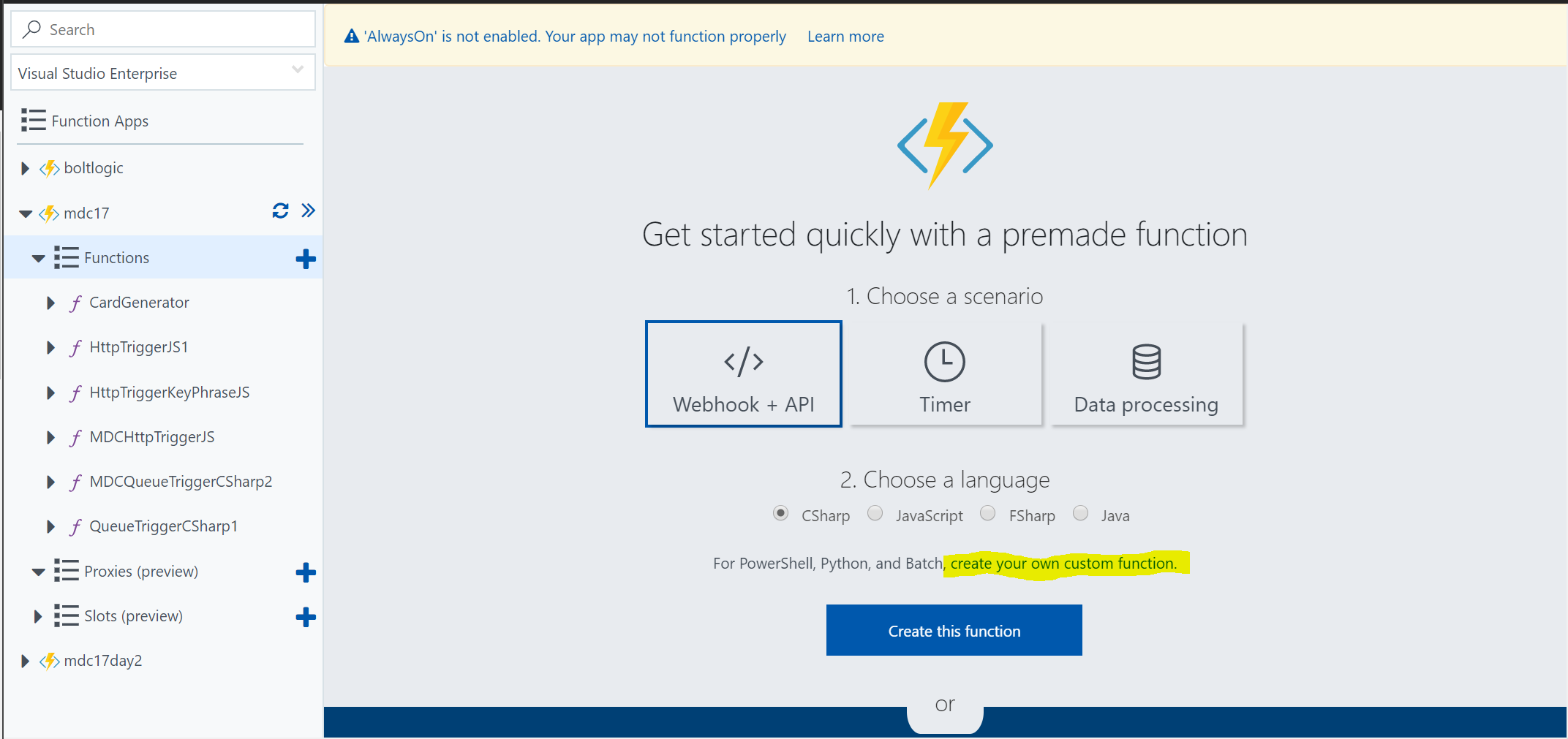
In this demo we will create to azure functions, the first one will be in node JS and second one will be in C#. the first function will be trigged using http request and will output messages to a queue, while the second function will get triggered using azure storage queue and will write the output in a table storage.

# Architecture

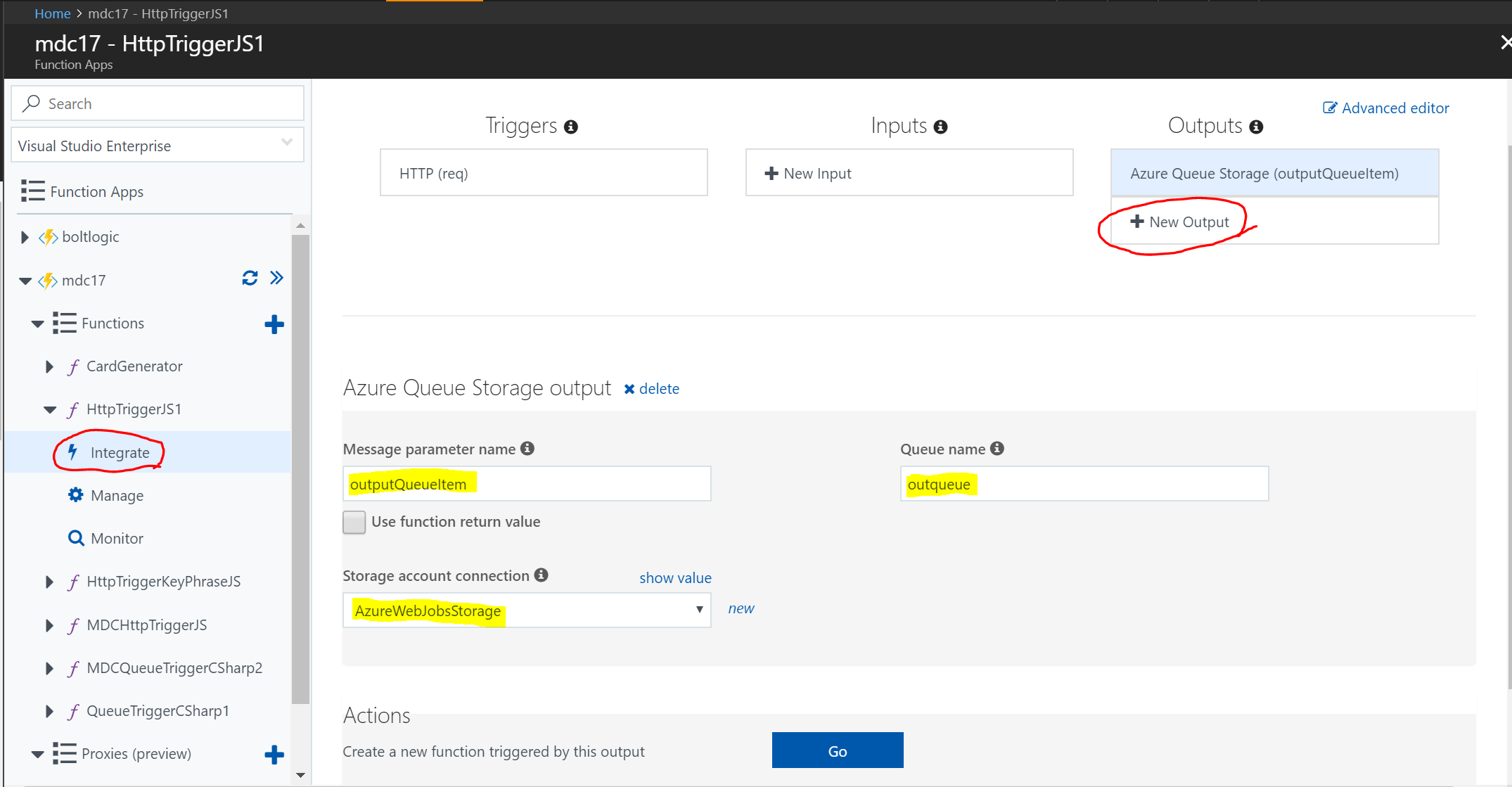


# Steps

1. Login to the azure portal
2. Click create a resource
3. Search for “Function App”
4. Click create
5. Fill in the information
6. Open the azure function app
7. Choose “create your own custom function” as per the below figure



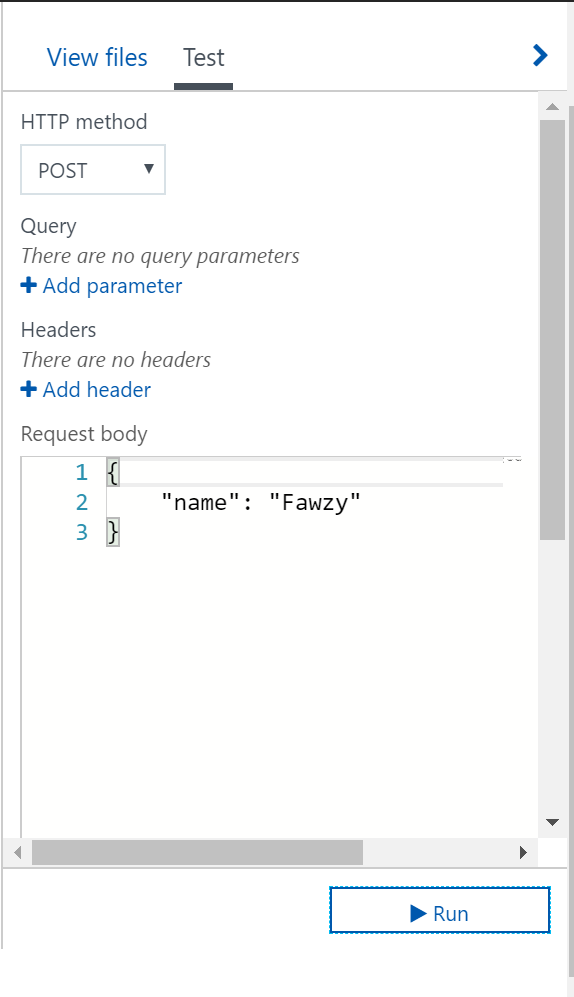
1. Choose the “HttpTrigger – JavaScript”
2. Click create at the bottom of the page
3. After the function is created click the integrate button as per the screen shot below



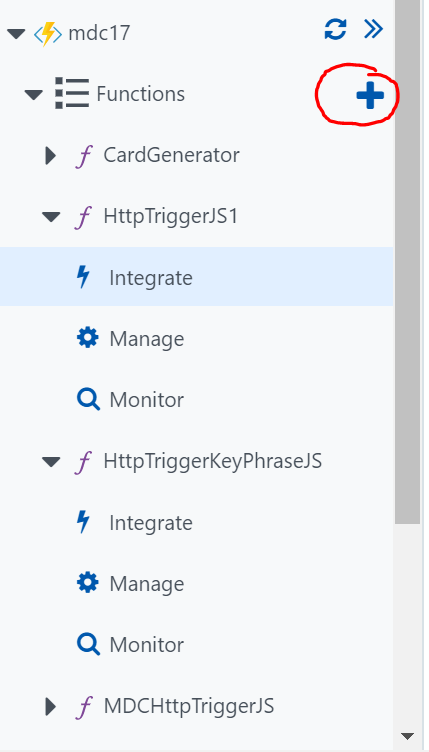
1. Then click the New output button and choose “Azure Queue storage”
2. You will find the fields populated as per the screen shot, but the **storage account connection** name would be different in your case.
3. Paste the below code or write it in your azure function

|  |
| --- |
| module.exports = function (context, req) {  context.log('JavaScript HTTP trigger function processed a request.');    if (req.query.name || (req.body && req.body.name)) {  context.log((req.query.name || req.body.name));  context.bindings.outputQueueItem = (req.query.name || req.body.name);  }  else  {  context.log('queue empty');  }  context.done();  }; |
| Code Description: the above code checks for a parameter called name in query string and in body, if found it adds it to the outputQueueItem |

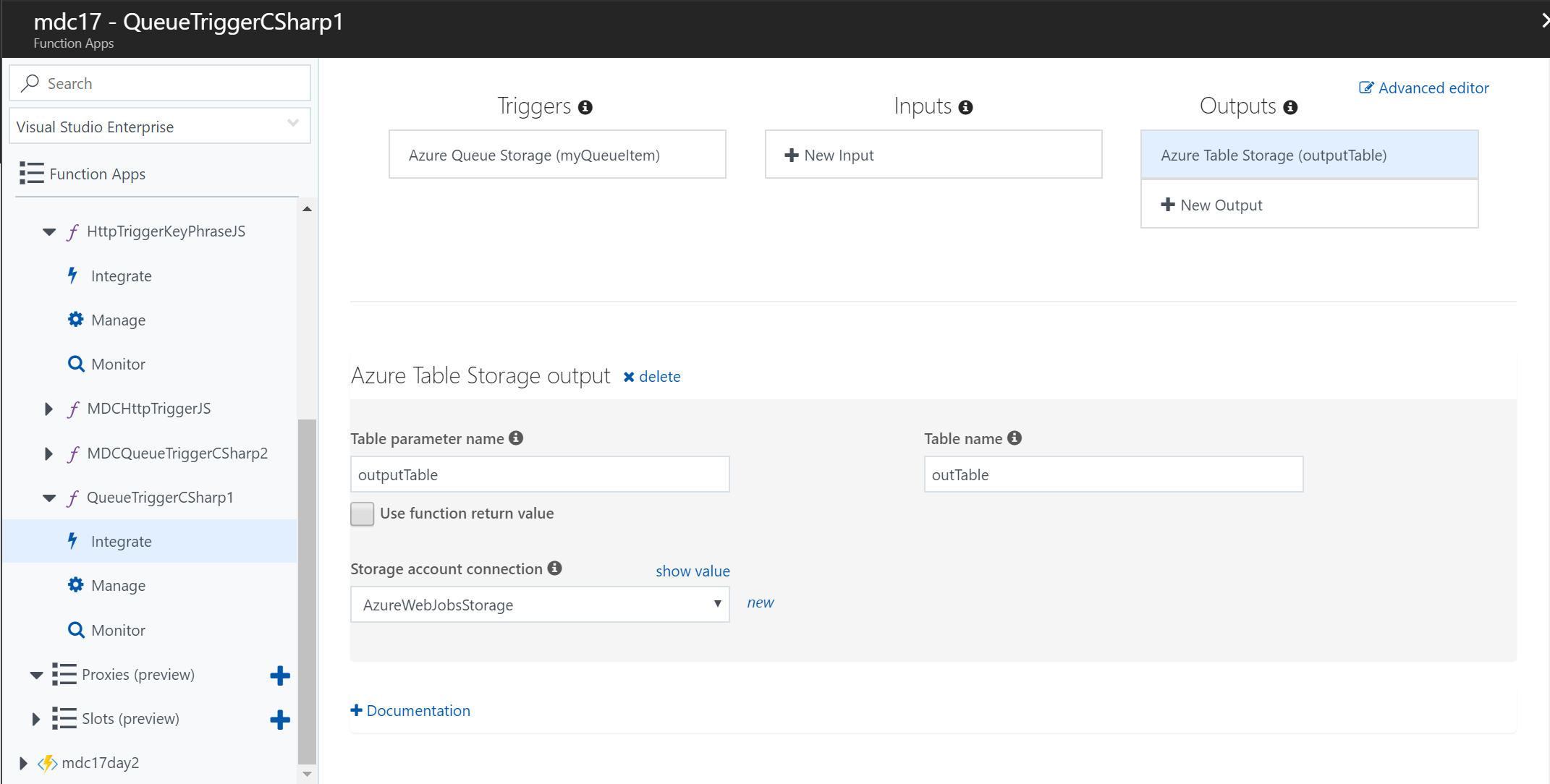
1. You can test your code from the test panel to the right side, this panel will simulate an http post request to your function



1. You can open the storage explorer app to show the messages that were added to the queue after you ran the tests. This concludes the first half of the demo
2. Click the add function button as per the below screen



1. Choose “QueueTrigger - C#”
2. Choose the correct storage account and type the correct queue name, click create
3. Click integrate



1. Under outputs click new output
2. Choose Azure table storage output
3. Make sure that the defaults are the same as the screen shot
4. Now go to the code view and paste the below code

|  |
| --- |
| using System;  public static void Run(string myQueueItem, ICollector<Order> outputTable, TraceWriter log)  {  log.Info($"C# Queue trigger function processed: {myQueueItem}");  outputTable.Add(  new Order{  PartitionKey = DateTime.Now.Month + "",  RowKey = DateTime.Now.Ticks + "",  Name = myQueueItem  }  );  }  public class Order  {  public string PartitionKey { get; set; }  public string RowKey { get; set; }  public string Name { get; set; }  } |
|  |

1. You can use the test pane to test you function, notice that test pane now looks different, it changes based on the input type
2. You can check the results using storage explorer
3. Now you can test the whole cycle by calling the first function through http and checking the output in table storage using the storage explorer app