

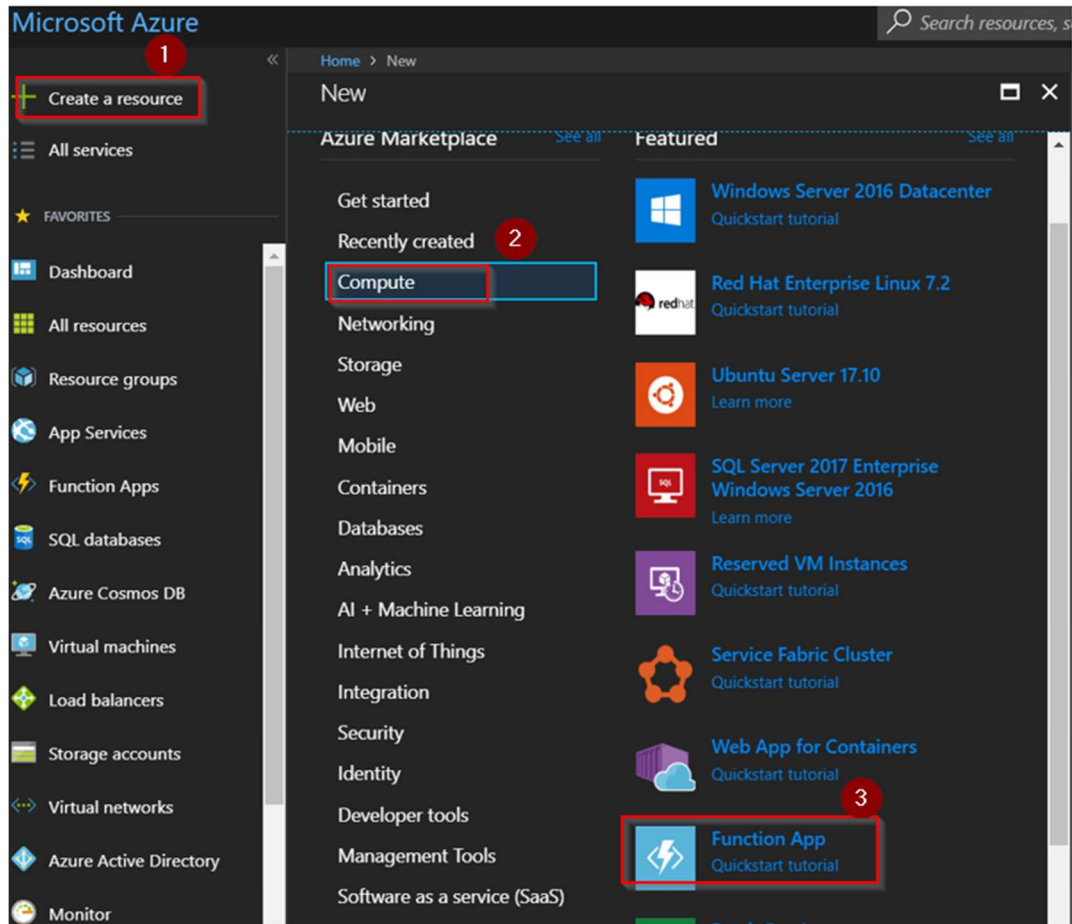
Azure Functions Demo- Part 1

Overview

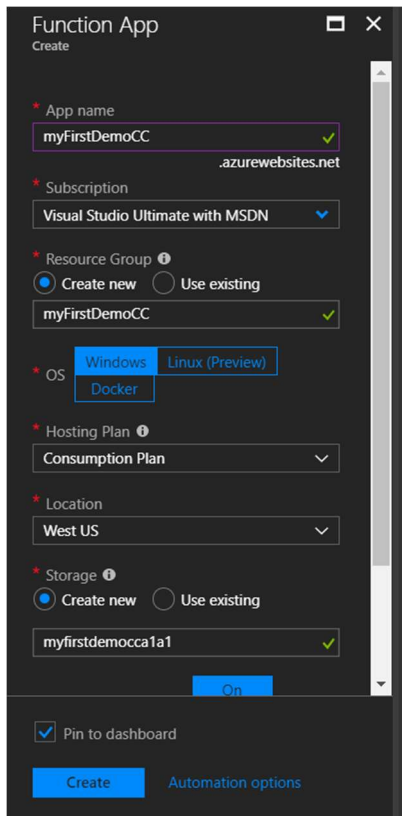
1. Create Function- using Azure Portal
2. Create Function - using Azure CLI
3. Create Function - using Visual studio IDE
4. Chaining of Microservices

1. Create Function- Azure Portal

1. Login to your Microsoft Azure Portal <https://portal.azure.com/>
Click > "Create a resource", then click "Compute" and choose "Function APP"



2. Enter the function APP name and choose one of the consumption plans. All other details are can be default.



Resource groups provide a way to monitor, control access, provision and manage billing for collections of assets that are required to run an application. Azure Resource Manager (ARM) is the technology that works behind the scenes so that you can administer assets using these logical containers.

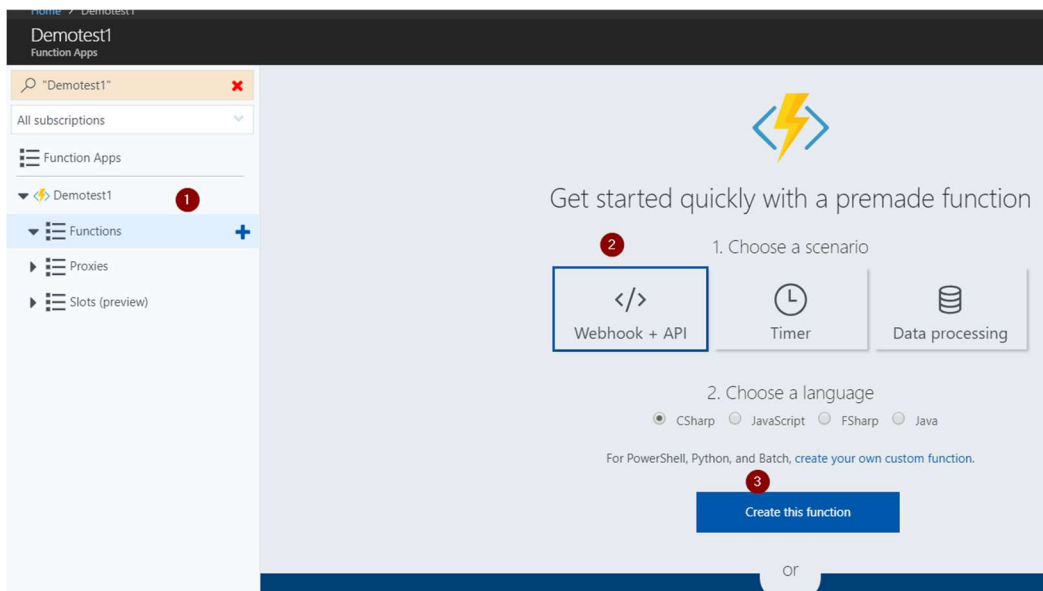
A resource group is simply **an identifier** that Azure Resource Manager applies to resources to group them together.

Hosting Plan- Pay per execution dynamically, predefined capacity allocation with predictable costs

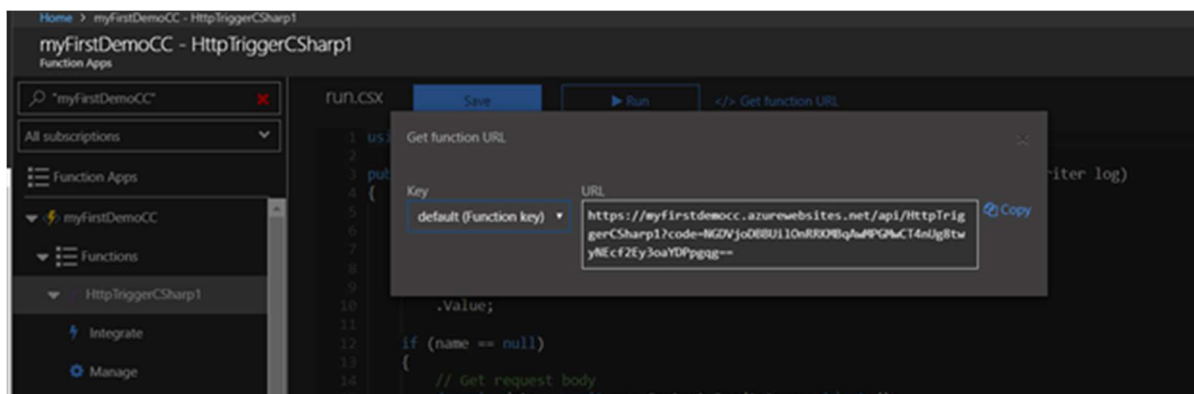
1. Consumption plan
2. AppService Plan

Check the option, pin to dashboard.

3. Once the function is created you can see the status of the function as “Running” and you set a custom trigger to test your application. Let’s add webhook+API,

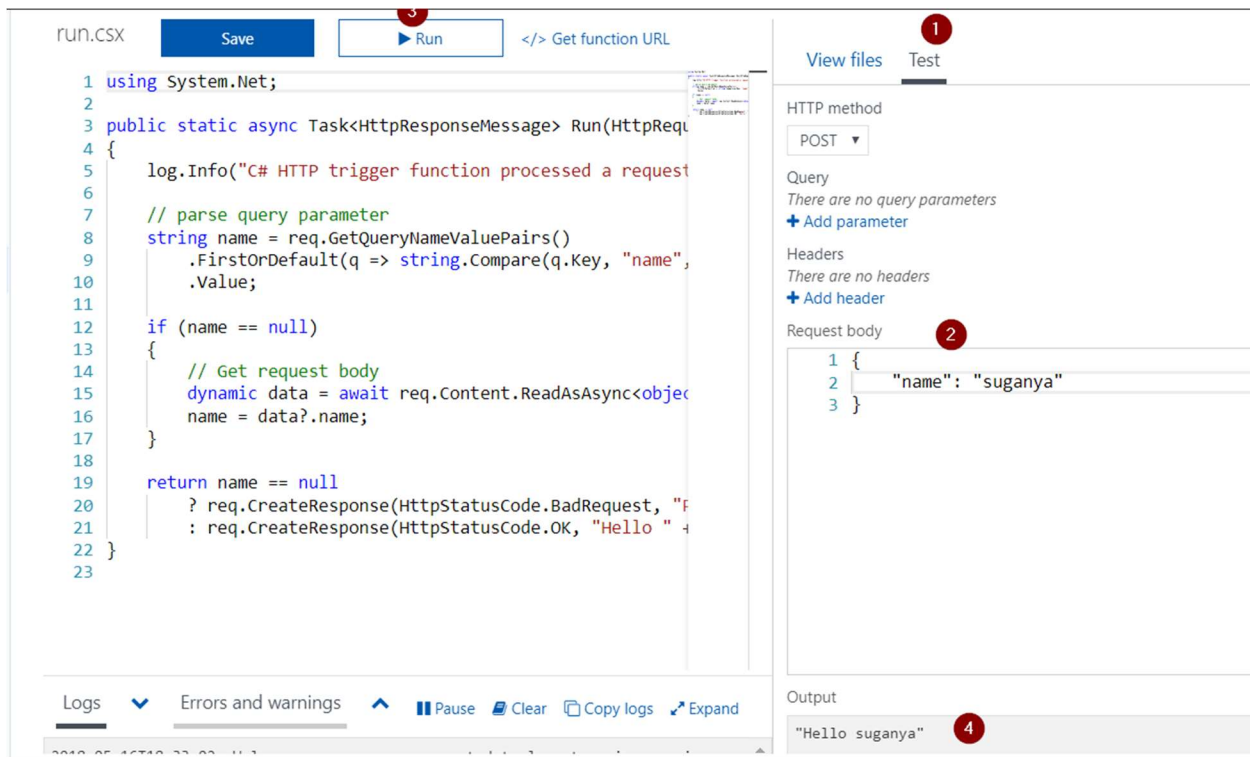


4. Click on the Function URL icon to view the access your function URL



Testing the function through HTTP call

1. Inbuilt testing tool



2. Testing from a web browser

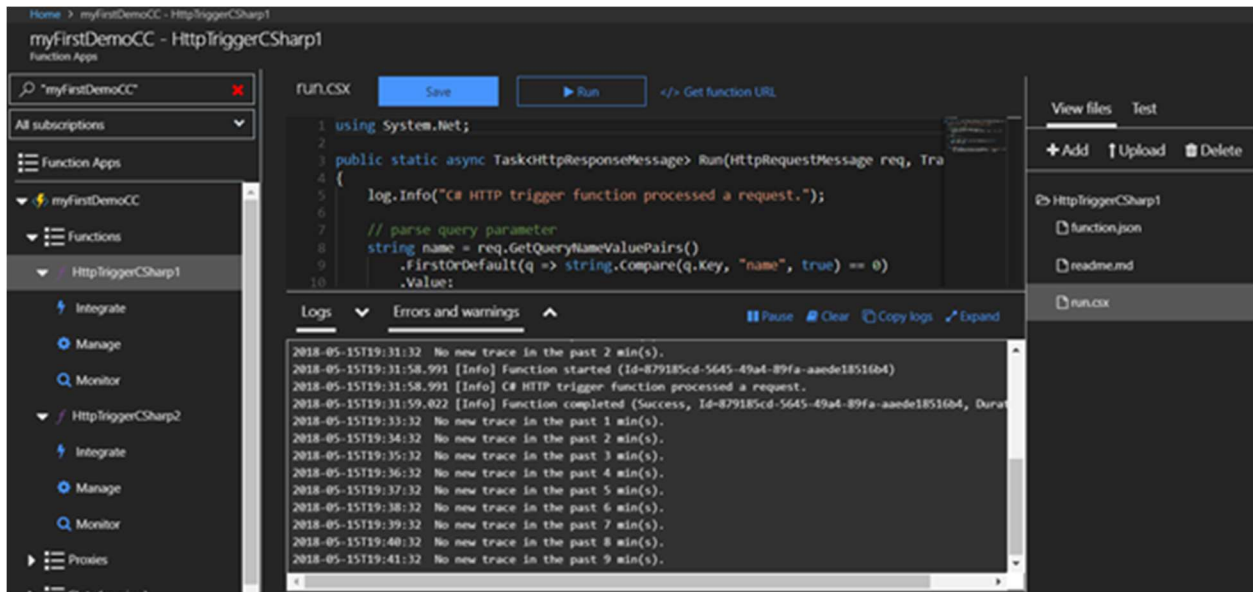
Function URL:

<https://myfirstdemocc.azurewebsites.net/api/HttpTriggerCSharp1?code=NGDVjoDBBUiOnRRKMBqAwMPGMwCT4nUg8twyNEcf2Ey3oaYDPpgg==&name=Friends>

Query string - "&name=<some_name>"



Click on the View files on the right corner, to check the log file.



2. Create Function - Azure CLI

Open the Azure CLI from the Azure Portal

1. Create a resource group

```
az group create --name <myFirstDemoCCTest> --location <westus>
```

```
Azure:\
PS Azure:\> az group create --name myFirstDemoCCTest --location westus
{
  "id": "/subscriptions/f8d97ed5-dc4c-46c3-81d9-b03649220a2c/resourceGroups/myFirstDemoCCTest",
  "location": "westus",
  "managedBy": null,
  "name": "myFirstDemoCCTest",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
Azure:\
```

2. Create an azure storage

```
az storage account create --name <myfirstdemostorage1> --location <westus> --resource-group <myFirstDemoCCTest> --sku Standard_LRS
```

Note: storage name should be within 3 to 24 characters of small letters and numbers.

```

Azure:~
PS Azure:\> az storage account create --name myfirstdemostorage1 --location westus --resource-group myFirstDemoCCTest --sku Standard_LRS
{
  "accessTier": null,
  "creationTime": "2018-05-15T18:30:20.568667+00:00",
  "customDomain": null,
  "enableHttpsTrafficOnly": false,
  "encryption": {
    "keySource": "Microsoft.Storage",
    "keyVaultProperties": null,
    "services": {
      "blob": {
        "enabled": true,
        "lastEnabledTime": "2018-05-15T18:30:20.693764+00:00"
      },
      "file": {
        "enabled": true,
        "lastEnabledTime": "2018-05-15T18:30:20.693764+00:00"
      },
      "queue": null,
      "table": null
    }
  }
}

```

3. Command to create a function

```

az functionapp create --deployment-source-url https://github.com/Azure-Samples/functions-quickstart
--resource-group myFirstDemoCCTest --consumption-plan-location westus --name myFirstDemoCCTest
--storage-account myfirstdemostorage1

```

```

Azure:~
PS Azure:\> az functionapp create --deployment-source-url https://github.com/Azure-Samples/functions-quickstart --resource-group myFirstDemoCCTest --consumption-plan
location westus --name myFirstDemoCCTest --storage-account myfirstdemostorage1
Linking to git repository 'https://github.com/Azure-Samples/functions-quickstart'
{
  "additionalProperties": {},
  "availabilityState": "Normal",
  "clientAffinityEnabled": true,
  "clientCertEnabled": false,
  "cloningInfo": null,
  "containersSize": 1536,
  "dailyMemoryTimeQuota": 0,
  "defaultHostName": "myfirstdemocctest.azurewebsites.net",
  "enabled": true,
  "enabledHostNames": [
    "myfirstdemocctest.azurewebsites.net",
    "myfirstdemocctest.scm.azurewebsites.net"
  ],
  "hostNameSslStates": [
    {
      "additionalProperties": {
        "ipBasedSslResult": null,
        "ipBasedSslState": "NotConfigured",
        "toUpdateIpBasedSsl": null
      },
      "hostType": "Standard",
      "name": "myfirstdemocctest.azurewebsites.net",
      "sslState": "Disabled",
      "thumbprint": null,
      "toUpdate": null,
      "virtualIp": null
    }
  ],
  "virtualIp": null
}

```

4.Delete a resource group

```

az group delete --name myResourceGroup

```

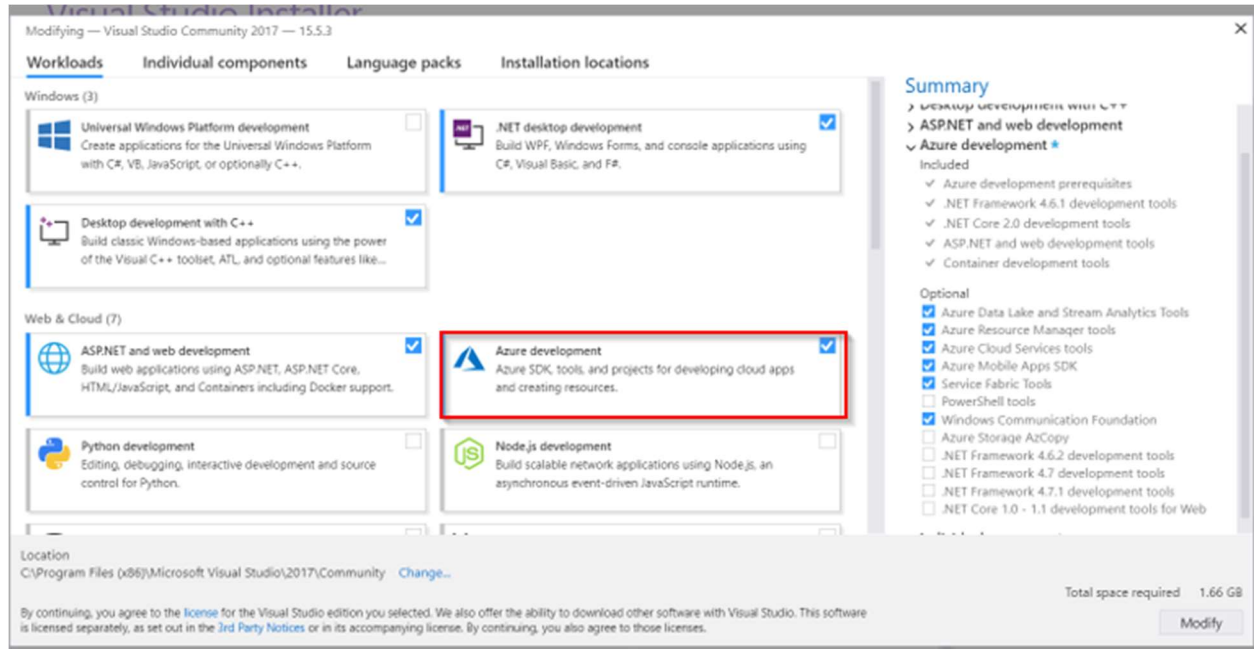
5. Follow the same steps given in section-1 to test your function.

Get your URI - <https://myfirstdemocctest.azurewebsites.net/api/HttpTriggerCSharp1?na>

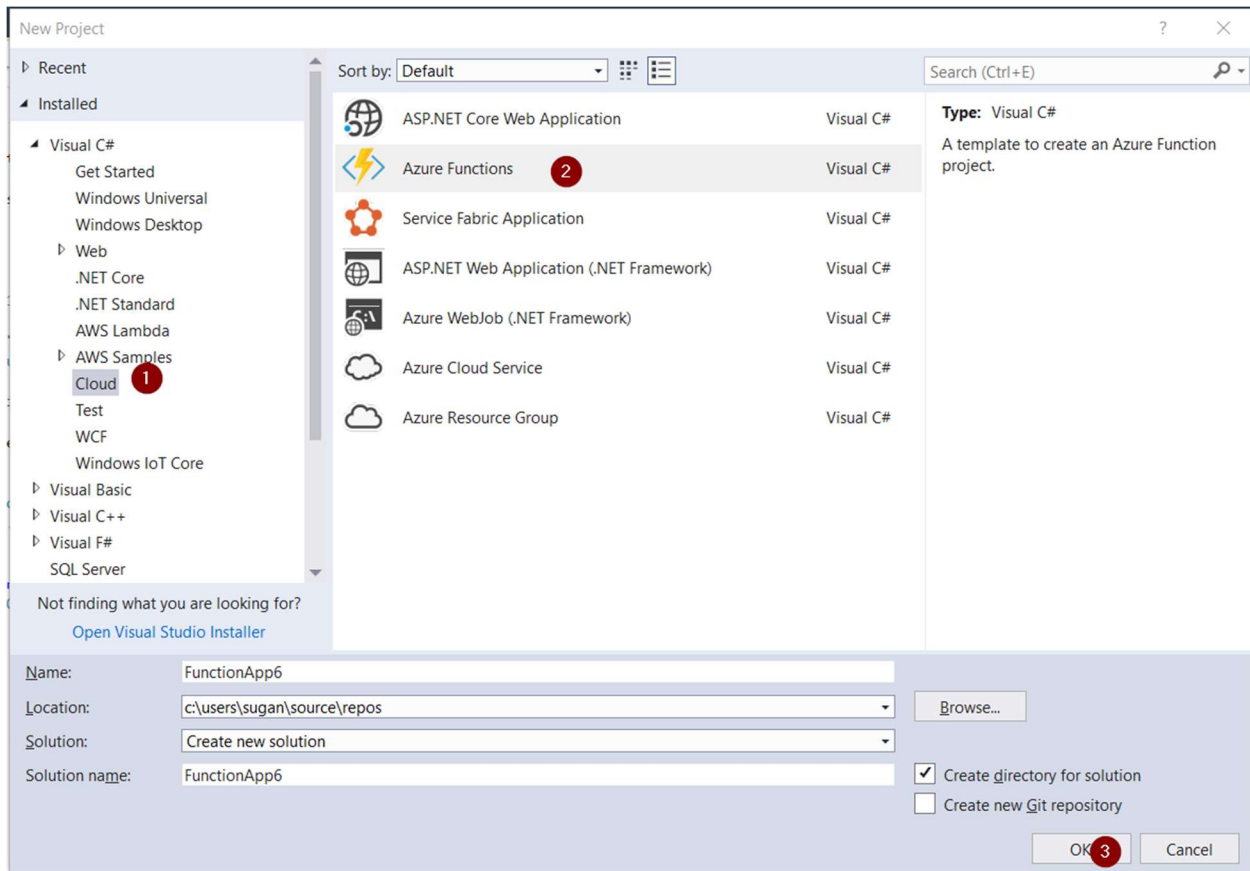
3.Create Function - Visual studio IDE

Prerequisite

Install the “Azure development” SDK in your Visual studio IDE

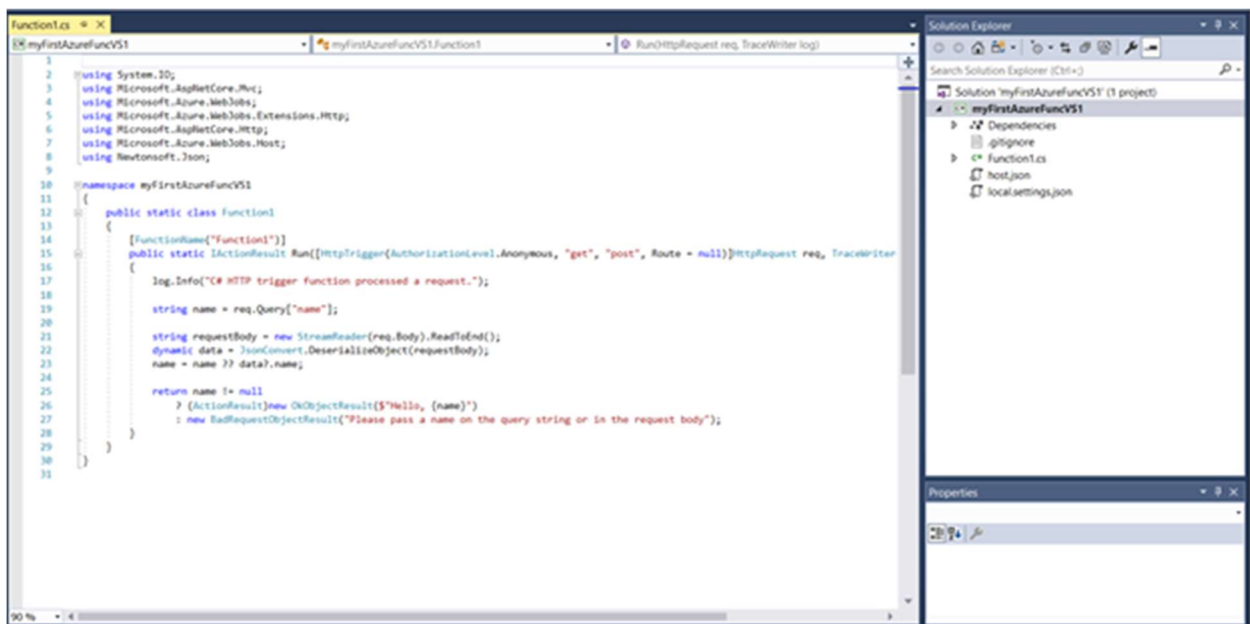


1.Create new project, Under C# choose Cloud, azure functions and name your function as you wish and then click OK. Then, on the next screen choose **HTTP Trigger**.

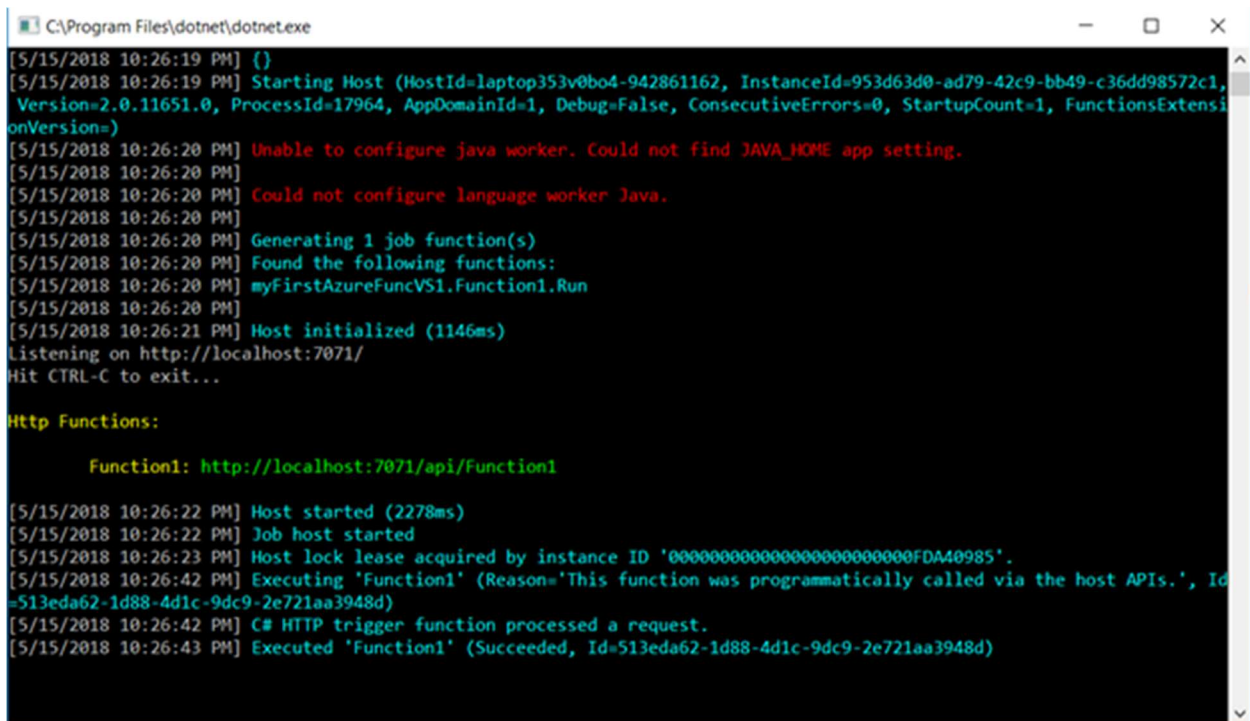


You will get a sample function open as below

All C# orchestration functions must have a parameter of type `DurableOrchestrationContext`, which exists in the `Microsoft.Azure.WebJobs.Extensions.DurableTask` assembly.



You can execute and test the function.



```
C:\Program Files\dotnet\dotnet.exe
[5/15/2018 10:26:19 PM] {}
[5/15/2018 10:26:19 PM] Starting Host (HostId=laptop353v0bo4-942861162, InstanceId=953d63d0-ad79-42c9-bb49-c36dd98572c1,
Version=2.0.11651.0, ProcessId=17964, AppDomainId=1, Debug=False, ConsecutiveErrors=0, StartupCount=1, FunctionsExtensi
onVersion=)
[5/15/2018 10:26:20 PM] Unable to configure java worker. Could not find JAVA_HOME app setting.
[5/15/2018 10:26:20 PM]
[5/15/2018 10:26:20 PM] Could not configure language worker Java.
[5/15/2018 10:26:20 PM]
[5/15/2018 10:26:20 PM] Generating 1 job function(s)
[5/15/2018 10:26:20 PM] Found the following functions:
[5/15/2018 10:26:20 PM] myFirstAzureFuncVS1.Function1.Run
[5/15/2018 10:26:20 PM]
[5/15/2018 10:26:21 PM] Host initialized (1146ms)
Listening on http://localhost:7071/
Hit CTRL-C to exit...

Http Functions:

    Function1: http://localhost:7071/api/Function1

[5/15/2018 10:26:22 PM] Host started (2278ms)
[5/15/2018 10:26:22 PM] Job host started
[5/15/2018 10:26:23 PM] Host lock lease acquired by instance ID '00000000000000000000000000000000FDM40985'.
[5/15/2018 10:26:42 PM] Executing 'Function1' (Reason='This function was programmatically called via the host APIs.', Id
=513eda62-1d88-4d1c-9dc9-2e721aa3948d)
[5/15/2018 10:26:42 PM] C# HTTP trigger function processed a request.
[5/15/2018 10:26:43 PM] Executed 'Function1' (Succeeded, Id=513eda62-1d88-4d1c-9dc9-2e721aa3948d)
```

Advantages

We can have checkpoint set for debugging.

Testing the function

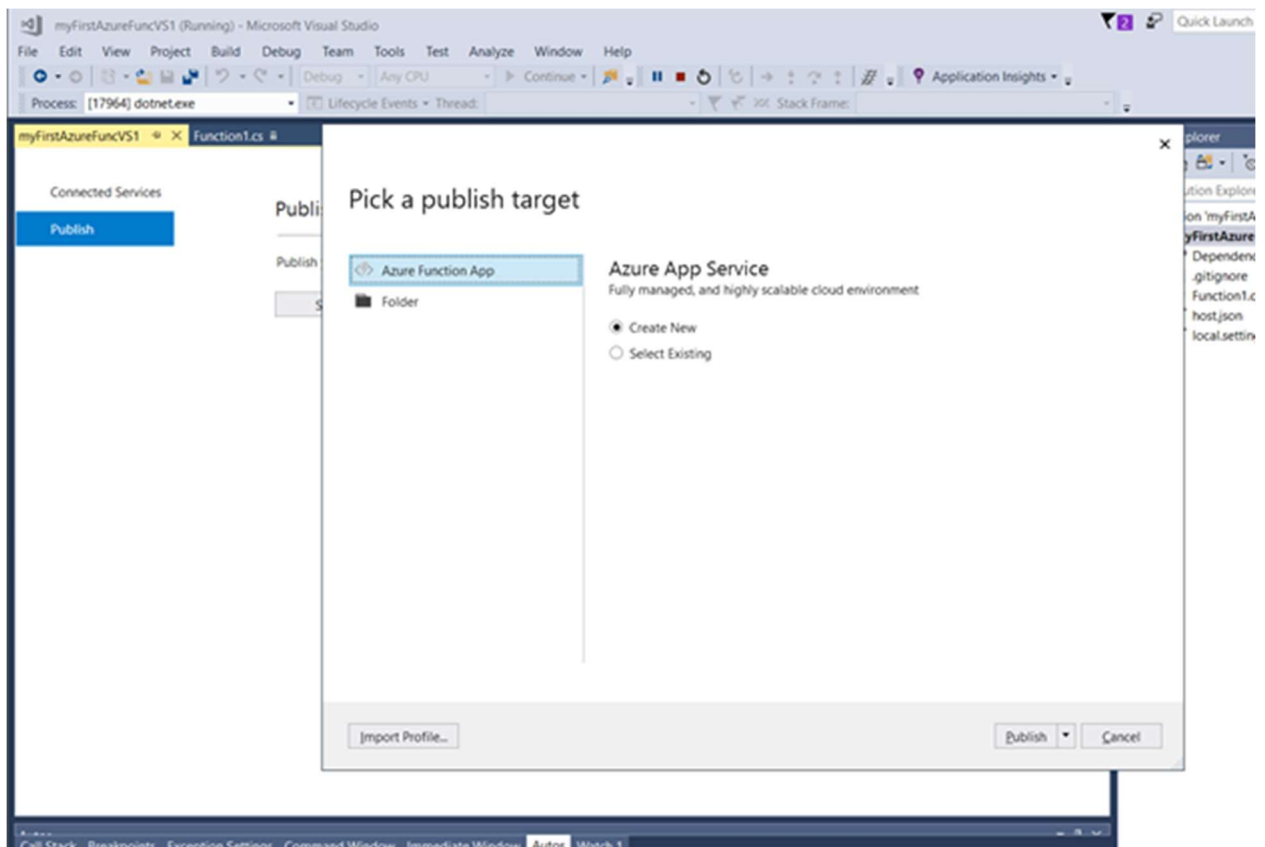
Add the query string along with the URI,

<http://localhost:7071/api/Function1>

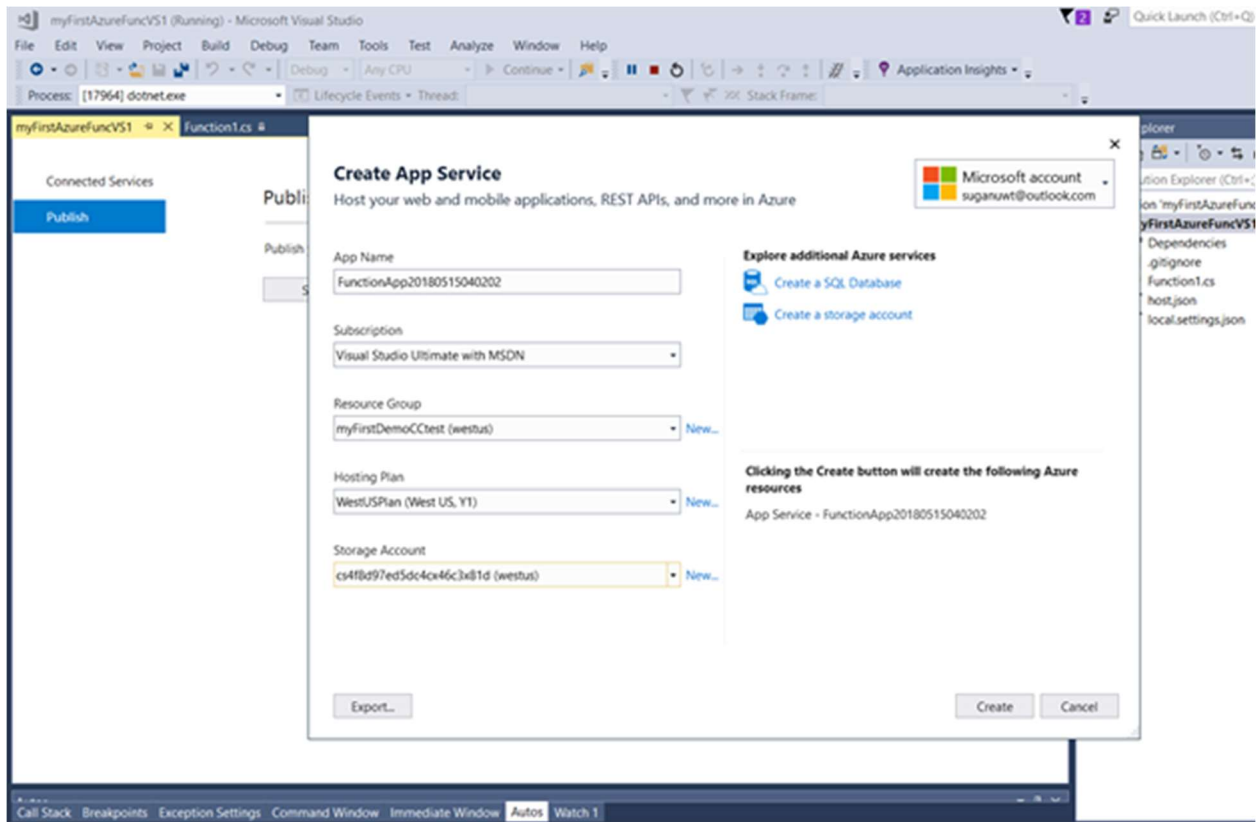
Query string - "&name=<some_name>"

Publish your function to Azure portal

1.Right click on your project and click publish, once you click publish, it will ask you to you login into your Azure account.



2.You can customize your app name at this point and click create



You will see a confirmation page, like below:

Publish

Azure successfully configured: [How was your experience?](#)

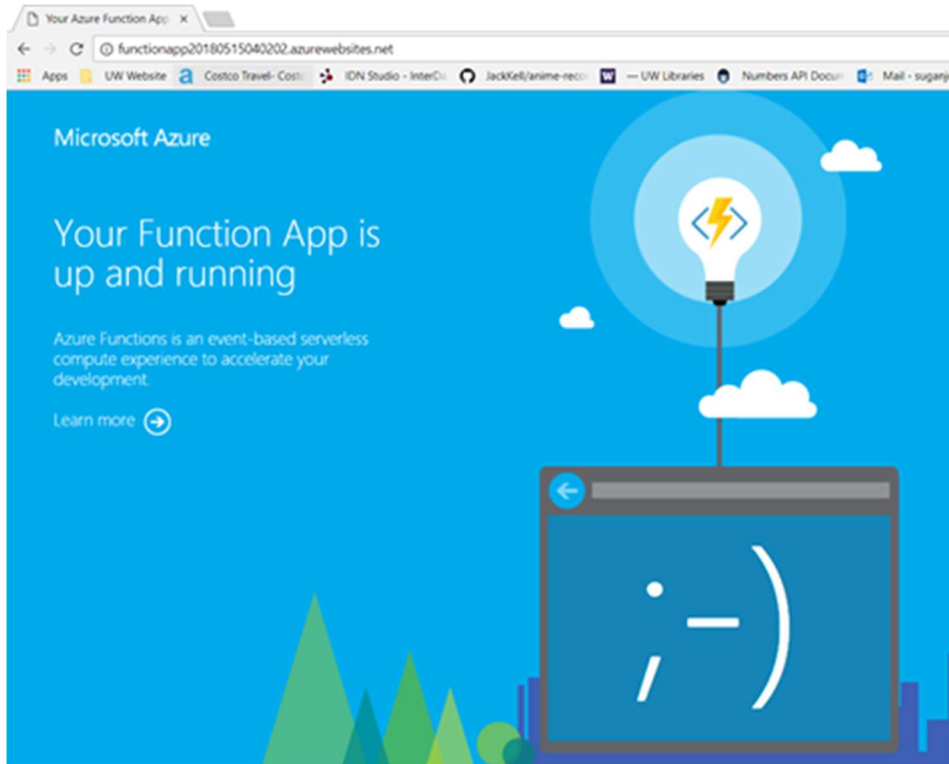
FunctionApp20180515040202 - Web Deploy Publish

New Profile...

Actions

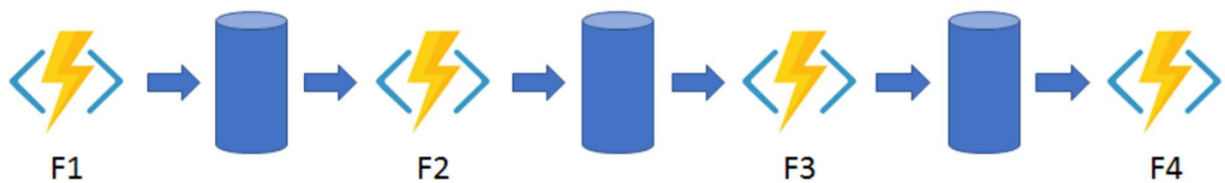
Site URL	http://functionapp201805...	Function App Settings...
Configuration	Release	Settings...
Delete existing files	False	
Username	\$FunctionApp20180515040202	
Password	*****	

You can test your function using a web browser or a fiddler tool



4. Chaining of Microservices

Function chaining refers to the pattern of executing a sequence of functions in a particular order. Often the output of one function needs to be applied to the input of another function.



ref-<https://docs.microsoft.com/en-us/azure/azure-functions/durable-functions-overview#pattern-1-function-chaining>

Prerequisites

1. Azure Development SDK in Visual Studio IDE
2. Install the Durable Functions extension and samples (Azure Functions)

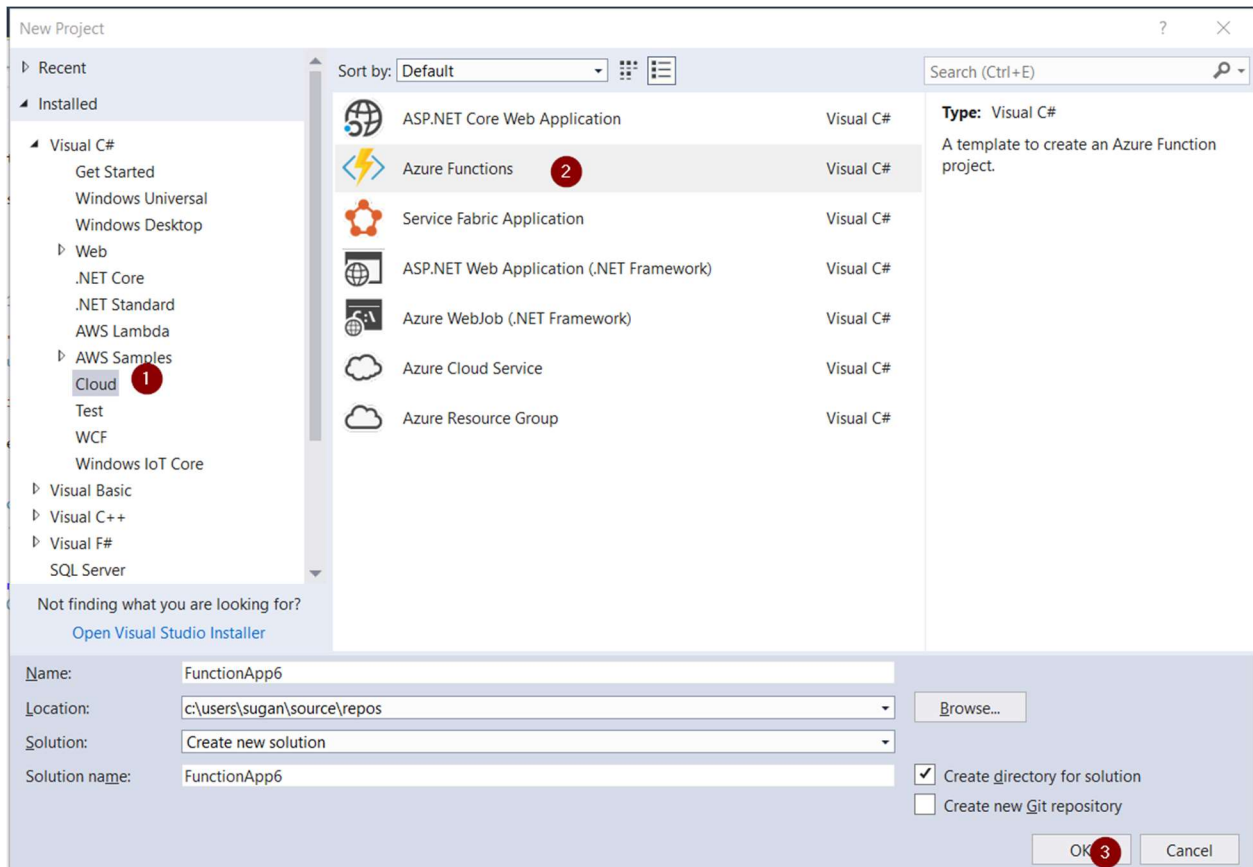
The Durable Functions extension for Azure Functions is provided in the NuGet package `Microsoft.Azure.WebJobs.Extensions.DurableTask`.

3. Install and run “Azure Storage Emulator”.

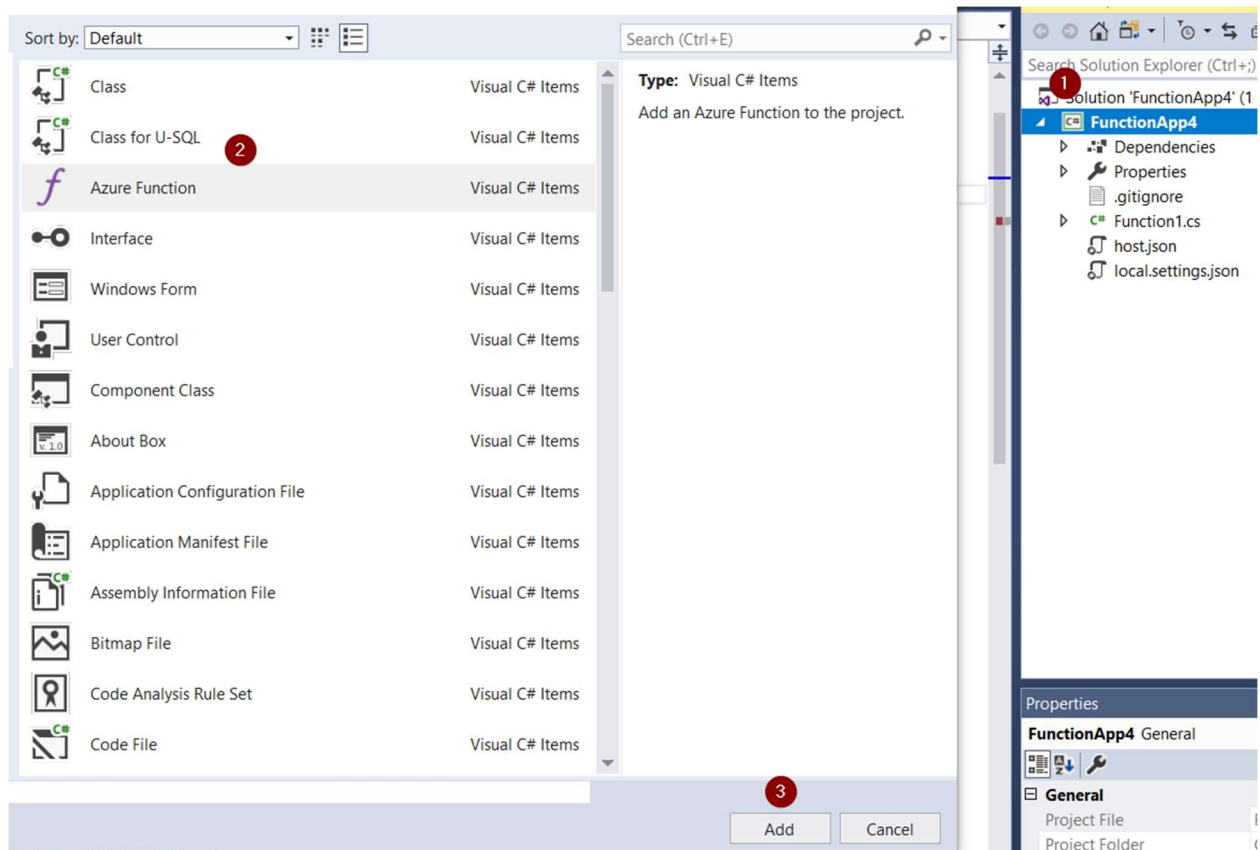
Steps for creating Chaining microservices

1. Create new project, Under C# choose Cloud, azure functions and name your function as you wish and then click OK.

Then choose an empty project.



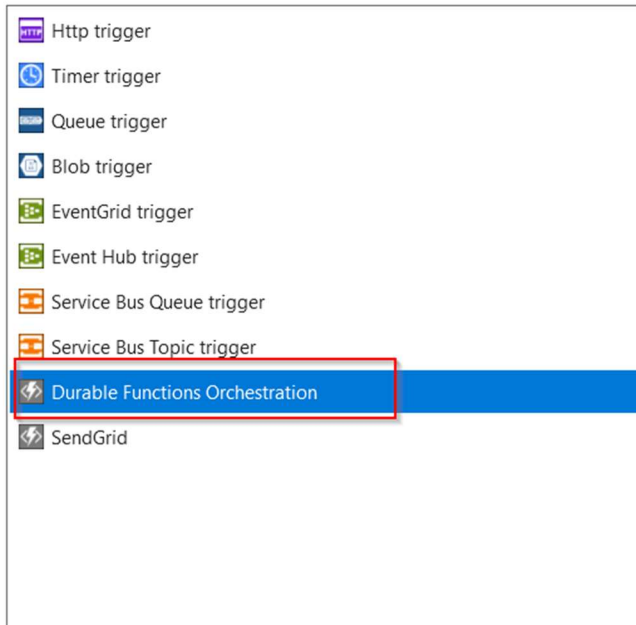
2. Right click on your solution file, click add and add a new Azure function



3. Then you will get options of triggers as below, choose "Durable Functions Orchestration"

New Azure Function - Function1

×



4. Once you click ok, Sample Durable function will appear. You can execute and test the same.

5. You will find the function URI in the execution window.


```
C:\Program Files\dotnet\dotnet.exe
Version=2.0.11651.0, ProcessId=5160, AppDomainId=1, Debug=False, ConsecutiveErrors=0, StartupCount=1, FunctionsExtension
Version=)
[5/16/2018 7:32:12 AM] Unable to configure java worker. Could not find JAVA_HOME app setting.
[5/16/2018 7:32:12 AM]
[5/16/2018 7:32:12 AM] Could not configure language worker Java.
[5/16/2018 7:32:12 AM]
[5/16/2018 7:32:12 AM] Loaded custom extension: DurableTaskExtension from 'referenced by: Method='FunctionApp4.Function1
.HttpStart', Parameter='starter'.'
[5/16/2018 7:32:13 AM] Generating 3 job function(s)
[5/16/2018 7:32:13 AM] Found the following functions:
[5/16/2018 7:32:13 AM] FunctionApp4.Function1.RunOrchestrator
[5/16/2018 7:32:13 AM] FunctionApp4.Function1.SayHello
[5/16/2018 7:32:13 AM] FunctionApp4.Function1.HttpStart
[5/16/2018 7:32:13 AM]
[5/16/2018 7:32:13 AM] Host initialized (1266ms)
Listening on http://localhost:7071/
Hit CTRL-C to exit...

Http Functions:

Function1.HttpStart: http://localhost:7071/api/Function1.HttpStart

[5/16/2018 7:32:14 AM] Host lock lease acquired by instance ID '00000000000000000000000000000000CD6D27D'.
[5/16/2018 7:32:16 AM] Host started (3983ms)
[5/16/2018 7:32:16 AM] Job host started
```

6. Test the function via HTTP call and once you access your function URI, you will get the statusQueryGetUri.

You can access the same to test your function.

```
localhost:7071/api/Function1.HttpStart
{"id": "388a8ed889d1404a924d9507994e1dcb", "statusQueryGetUri": "http://localhost:7071/runtime/webhooks/DurableTaskExtension/instances/388a8ed889d1404a924d9507994e1dcb?
taskHub=DurableFunctionsHub&connection=Storage&code=rbA7D10WvYhg7F8r6ZpSNn8/4ITJ1sJroURdrArR10i63DeskBFGmg=", "sendEventPostUri": "http://localhost:7071/runtime/webhooks/DurableTaskExtension/instances/388a8ed889d1
404a924d9507994e1dcb/raiseEvent/{eventName}?"
taskHub=DurableFunctionsHub&connection=Storage&code=rbA7D10WvYhg7F8r6ZpSNn8/4ITJ1sJroURdrArR10i63DeskBFGmg=", "terminatePostUri": "http://localhost:7071/runtime/webhooks/DurableTaskExtension/instances/388a8ed889d1
404a924d9507994e1dcb/terminate?reason={text}&taskHub=DurableFunctionsHub&connection=Storage&code=rbA7D10WvYhg7F8r6ZpSNn8/4ITJ1sJroURdrArR10i63DeskBFGmg="}
```

7. You can publish this function to Azure portal the same way we did it in Section 3- Creation Function through Visual Studio IDE.

8. After publishing your function to Azure portal, check for “Get function URI” option under Function1_HTTPStart.

Local Storage Emulator Log - You can find the status of your function here and it will be helpful for debugging.

Microsoft Azure Storage Explorer

One or more accounts requires reauthentication. Manage Accounts Close

Search for resources

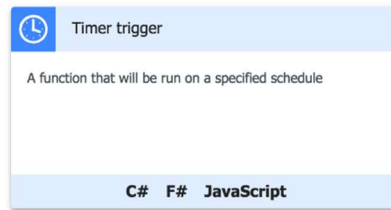
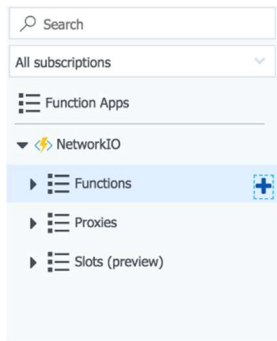
Refresh All

RowKey	Timestamp	EventId	EventType	ExecutionId
0000000000000008	2018-05-16T03:34:04.065Z	-1	OrchestratorStarted	efd489a8014144799dad80e4bcd9aa56
0000000000000000	2018-05-16T03:34:02.664Z	-1	OrchestratorStarted	efd489a8014144799dad80e4bcd9aa56
0000000000000002	2018-05-16T03:34:02.664Z	0	TaskScheduled	efd489a8014144799dad80e4bcd9aa56
0000000000000003	2018-05-16T03:34:02.664Z	-1	OrchestratorCompleted	efd489a8014144799dad80e4bcd9aa56
0000000000000004	2018-05-16T03:34:03.913Z	-1	OrchestratorStarted	efd489a8014144799dad80e4bcd9aa56
0000000000000005	2018-05-16T03:34:03.913Z	-1	TaskCompleted	efd489a8014144799dad80e4bcd9aa56
0000000000000006	2018-05-16T03:34:03.913Z	1	TaskScheduled	efd489a8014144799dad80e4bcd9aa56
0000000000000007	2018-05-16T03:34:03.914Z	-1	OrchestratorCompleted	efd489a8014144799dad80e4bcd9aa56
0000000000000001	2018-05-16T03:34:02.664Z	-1	ExecutionStarted	efd489a8014144799dad80e4bcd9aa56
0000000000000009	2018-05-16T03:34:04.065Z	-1	TaskCompleted	efd489a8014144799dad80e4bcd9aa56
000000000000000A	2018-05-16T03:34:04.065Z	2	TaskScheduled	efd489a8014144799dad80e4bcd9aa56
000000000000000B	2018-05-16T03:34:04.065Z	-1	OrchestratorCompleted	efd489a8014144799dad80e4bcd9aa56
000000000000000C	2018-05-16T03:34:04.564Z	-1	OrchestratorStarted	efd489a8014144799dad80e4bcd9aa56
0000000000000000	2018-05-16T03:34:04.564Z	-1	TaskCompleted	efd489a8014144799dad80e4bcd9aa56

Showing 1 to 17 of 17 cached items

Create a function triggered by timer

1. Go to function Apps. Click on +. Click on Timer trigger

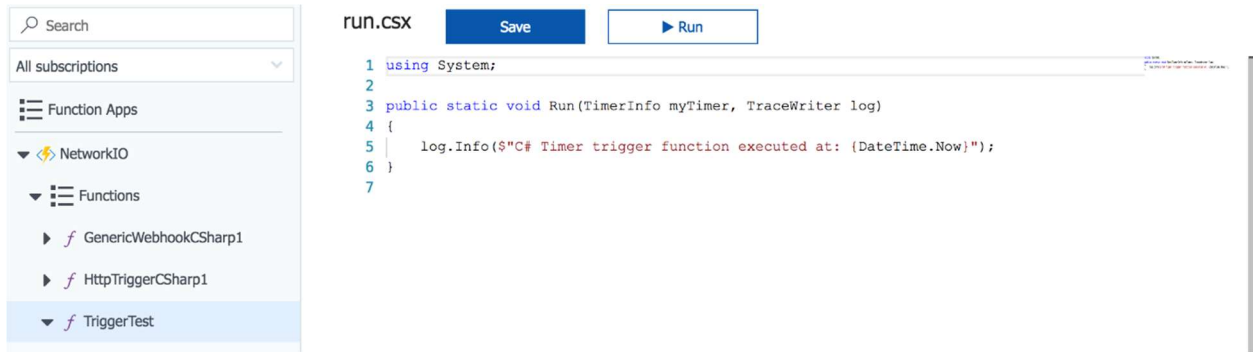


2. Choose the language (C#, JavaScript) and name for the trigger and click create.

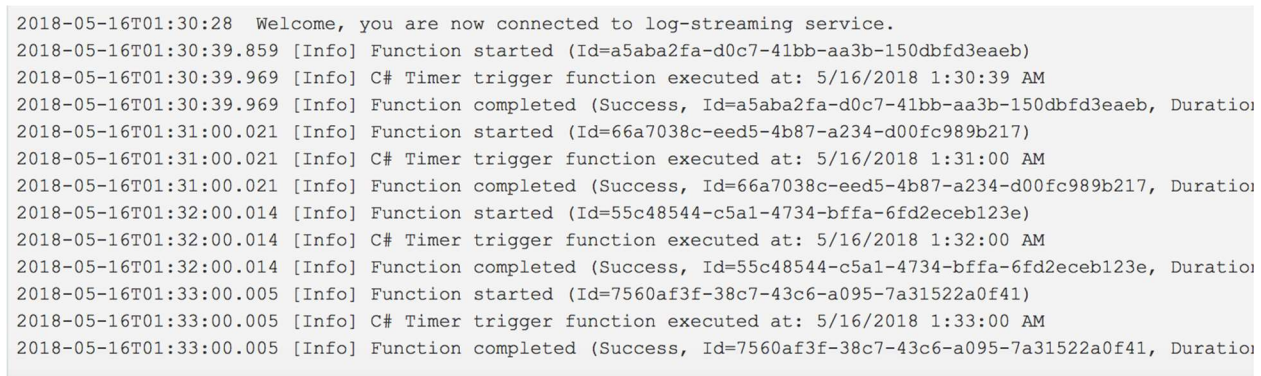
A screenshot of the 'New Function' form in the Azure portal. The form is titled 'New Function' and is for a 'Timer trigger'. It has a blue header with a clock icon and the text 'Timer trigger'. Below the header, there are three sections: 'Language:' with a dropdown menu set to 'C#'; 'Name:' with a text input field containing 'TriggerTest'; and 'Timer trigger' section with 'Schedule' and an input field containing '0 */1 * * * *'. At the bottom, there are two buttons: 'Create' (blue) and 'Cancel' (white with blue border).

The parameter in schedule `0 */1 * * *` will trigger the function for every 1 minutes. You can change the parameter as necessary (`0 */5 * * *` =for every 5 min, `0 0 */1 * * *` =for every 1 hour)

3. Choose the trigger created (TriggerTest) and click run.



You can see the results in log file. The function will be every 1 min.

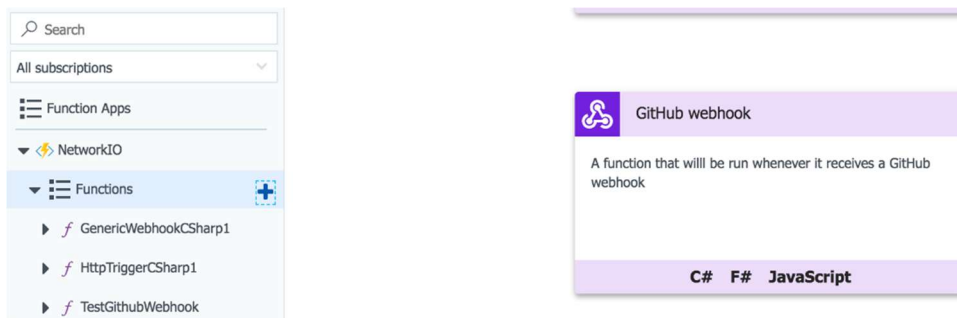


Create a function triggered by GitHub webhook

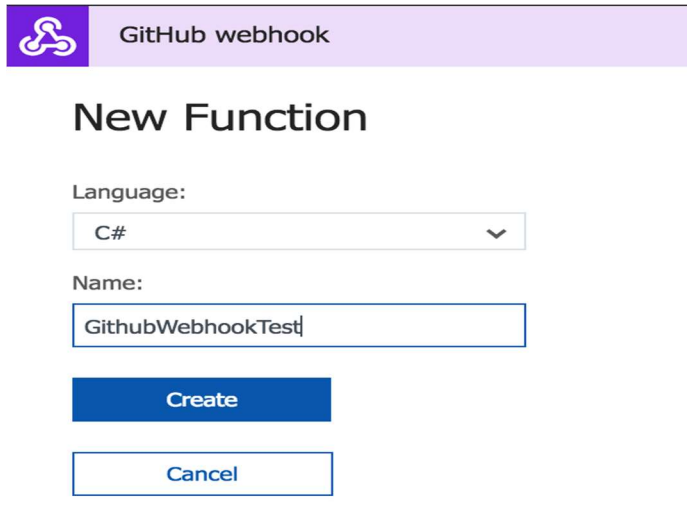
Prerequisite:

Git Hub account with one repository

1. Go to function Apps. Click on +. Click on GitHub webhook.



2. Choose the language (C#, JavaScript) and name for the function and click create.



Once the function is created copy the Get function URL and Get GitHub secret.



3. Log in to your GitHub account and click on a repository and click on settings and webhook and Add webhook

Payload URL *

https://networkio.azurewebsites.net/api/GithubWebhookTest?clier

Content type

application/json

Secret

.....

Payload URL: copy the Get function URL

content type: application/json

Secret: copy paste GitHub secret

• Let me select individual events.

- | | |
|---|---|
| <input type="checkbox"/> Commit comments
Commit or diff commented on. | <input type="checkbox"/> Branch or tag creation
Branch or tag created. |
| <input type="checkbox"/> Branch or tag deletion
Branch or tag deleted. | <input type="checkbox"/> Deployments
Repository deployed. |
| <input type="checkbox"/> Deployment statuses
Deployment status updated from the API. | <input type="checkbox"/> Forks
Repository forked. |
| <input type="checkbox"/> Wiki
Wiki page updated. | <input checked="" type="checkbox"/> Issue comments
Issue comment created, edited, or deleted. |
| <input checked="" type="checkbox"/> Issues
Issue opened, edited, closed, reopened, assigned, unassigned, labeled, unlabeled, milestone, or demilestone. | <input type="checkbox"/> Labels
Label created, edited or deleted. |

For events select Issue comments and Issues. Click Add Webhook. After adding go to Issues tab and click on New Issue.


The screenshot shows the GitHub repository navigation bar with tabs for Code, Issues (1), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below the navigation bar is a notification banner: "Label issues and pull requests for new contributors" with a "Dismiss" link. The notification text says: "Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue". Below the notification is a search bar with "is:issue is:open" and buttons for "Filters", "Labels", "Milestones", and a green "New issue" button.

Create Issue and give comments.

The screenshot shows the GitHub "New Issue" form. The title is "Test Web Hook". The form has "Write" and "Preview" tabs. The text area contains "Checking Azure GitHub WebHook". Below the text area is a dashed line and the text "Attach files by dragging & dropping, selecting them, or pasting from the clipboard." At the bottom left, it says "Styling with Markdown is supported". At the bottom right is a green "Submit new issue" button.

After submitting go to webhook in GitHub and click Recent Deliveries and check the response sent. You can see the comments written.

Recent Deliveries

✓  b2d56f10-58bf-11e8-90fd-dcd5b1a895c0 2018-05-15 21:14:59 ...

Request Response 200 ↻ Redeliver 🕒 Completed in 0 seconds.

Headers

```
Cache-Control: no-cache
Content-Length: 43
Content-Type: application/json; charset=utf-8
Date: Wed, 16 May 2018 04:15:00 GMT
Expires: -1
Pragma: no-cache
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
```

Body

```
"From Github:Checking Azure GitHub WebHook"
```

Go to Function and check logs and you can see that function is triggered and executed.

```
2018-05-16T04:11:07 No new trace in the past 3 min(s).
2018-05-16T04:11:53.190 [Info] Function started (Id=1c9170e9-fee4-48ba-860a-fd5cb210cf2f)
2018-05-16T04:11:53.190 [Info] C# HTTP trigger function processed a request.
2018-05-16T04:11:53.190 [Info] Function completed (Success, Id=1c9170e9-fee4-48ba-860a-fd5cb210cf2f, Duration: 0.0001788s)
2018-05-16T04:12:54.791 [Info] Function started (Id=ef2be21b-6e03-475e-8029-643143a9d94b)
2018-05-16T04:12:54.804 [Info] C# HTTP trigger function processed a request.
2018-05-16T04:12:54.804 [Info] Function completed (Success, Id=ef2be21b-6e03-475e-8029-643143a9d94b, Duration: 0.0001788s)
2018-05-16T04:14:07 No new trace in the past 1 min(s).
2018-05-16T04:15:00.449 [Info] Function started (Id=0b493343-9bc2-46d0-976a-19987da0c2ef)
2018-05-16T04:15:00.464 [Info] C# HTTP trigger function processed a request.
2018-05-16T04:15:00.464 [Info] Function completed (Success, Id=0b493343-9bc2-46d0-976a-19987da0c2ef, Duration: 0.0001788s)
```

Add messages to an Azure Storage queue using Functions

1. Create a sample HTTP trigger function.

HTTP HTTP trigger

New Function

Language:

Name:

HTTP trigger

Authorization level ⓘ

2. Click on the Integrate option in the created function and + New Output

Search

All subscriptions

Function Apps

- CDemo
 - Functions
 - HttpTriggerCSharp1
 - Sample
 - Integrate**
 - Manage
 - Monitor

Triggers ⓘ
HTTP (req)

Inputs ⓘ
+ New Input

Outputs ⓘ [Advanced editor](#)
HTTP (\$return)
+ New Output

HTTP trigger [delete](#)

Allowed HTTP methods ⓘ
Selected methods

Mode ⓘ
Standard

Select Azure Queue Storage and

Triggers ⓘ

Inputs ⓘ


Outputs ⓘ [Advanced editor](#)


HTTP (req)


+ New Input


HTTP (\$return)

+ New Output

 Azure Event Hubs

 Azure Queue Storage

 Azure Blob Storage

 External File (Preview)

Give all the inputs and save it

Azure Queue Storage output

Message parameter name ⓘ
outputQueueItem

Use function return value

Queue name ⓘ
outqueue

Storage account connection ⓘ [show value](#)
AzureWebJobsDashboard *new*





[Save](#) [Cancel](#)

Add the below code

outputQueueItem.Add("Name passed to the function: " + name); before return statement and add ICollector<string> outputQueueItem in the parameter list.

Run the function. After running go to Storage accounts click on the function App name and select queues.

Services

 Blobs REST-based object storage for unstructured data Configure CORS rules Setup custom domain View metrics	 Files File shares that use the standard SMB 3.0 protocol Configure CORS rules View metrics
 Tables Tabular data storage Configure CORS rules View metrics	 Queues Effectively scale apps according to traffic Configure CORS rules View metrics

and select the created queue

Search queues by prefix

QUEUE	URL
outqueue	https://cdemoba28.queue.core.windows.net/outqueue

You can see the message passed from the function.

ID	MESSAGE TEXT	INSERTION TIME	EXPIRATION TIME	DEQUEUE CO...
d139d3a8-b...	Name passed to the function: Azure	Wed, 16 May 2018 ...	Wed, 23 May 2018 ...	0

Reference:

[1] Microsoft Azure docs: <https://docs.microsoft.com/en-us/azure/azure-functions/durable-functions-install>

[2] Create function in Java/ Maven- <https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-java-maven>

[3]Add Messages to Queue- <https://docs.microsoft.com/en-us/azure/azure-functions/functions-integrate-storage-queue-output-binding>

[4] Timer Trigger- <https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-scheduled-function>

[5]GitHub Webhook- <https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-github-webhook-triggered-function>