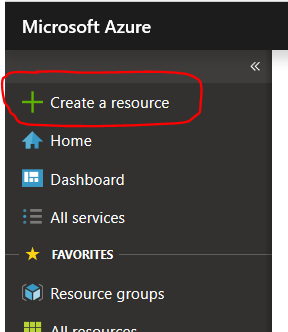
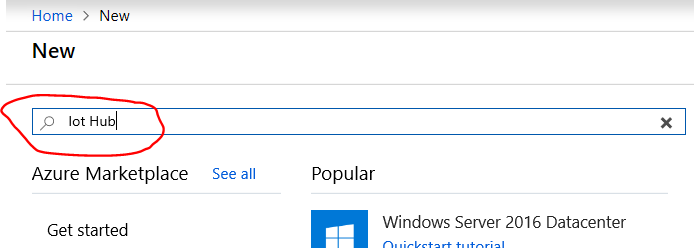


**IoT Hub:**

1. Log in to azure portal.
2. Click on create a resource from left pane as shown below.

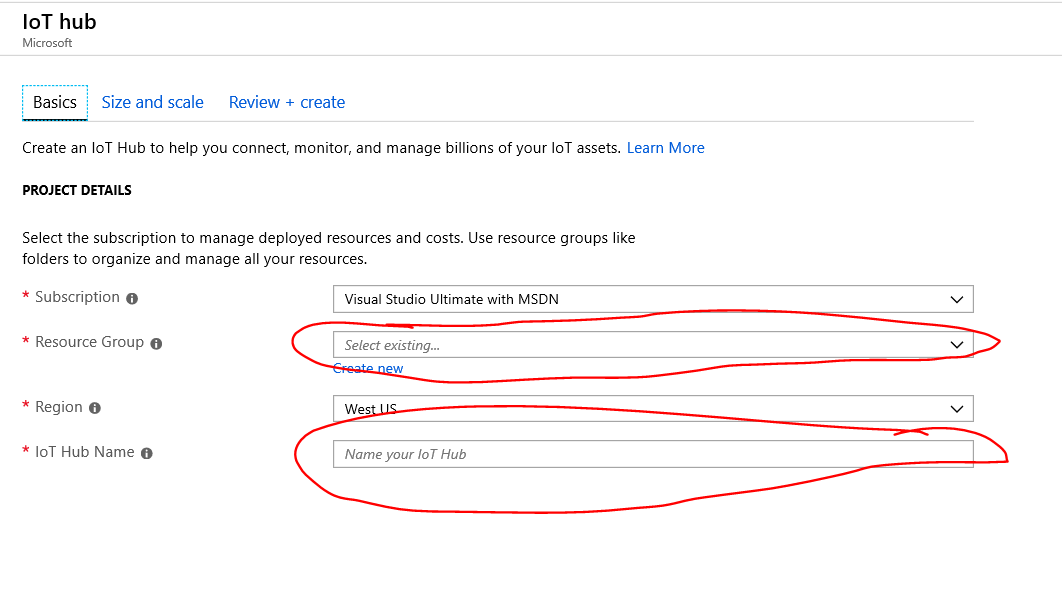


1. Type “IoT Hub” in the search box and click on the IoT Hub from the results.

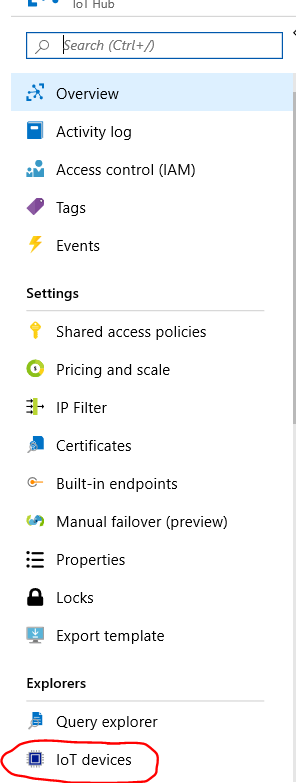


1. Click on create.
2. Provide the information about Iot Hub name and resource group, and create on “Review + Create” button. If you want to choose the pricing tier, click on “Size and Scale” Tab.

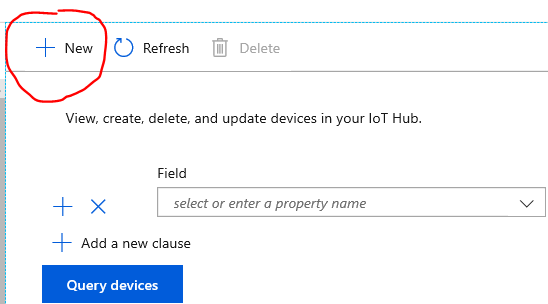
For tire information refer to [IoT Hub Pricing](https://azure.microsoft.com/en-us/pricing/details/iot-hub/).



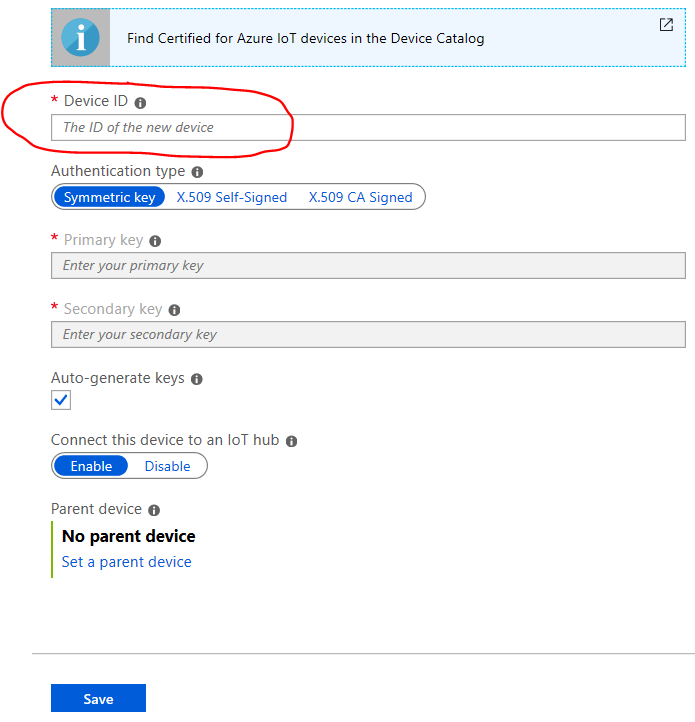
1. Once you click on Create it will start provisioning the IoT Hub. Wait for IoT hub to be provisioned.
2. Once IoT hub is deployed, go to IoT hub either from “all resources” option or by clicking on “Go To Resource” option in deployment notification.
3. To register a new device in IoT hub navigating to “IoT Devices” from left pane.



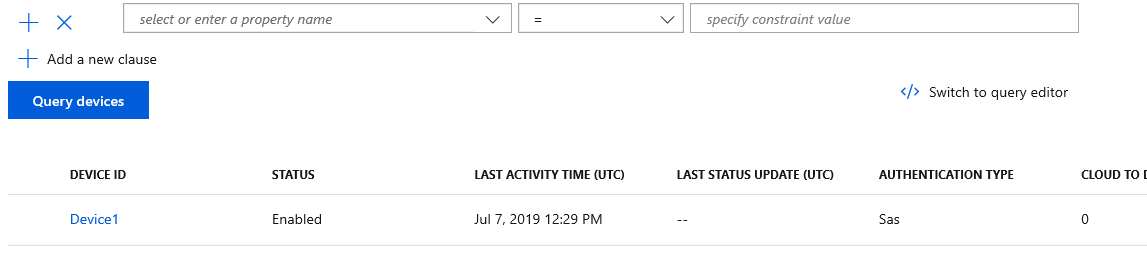
1. Click on New button.



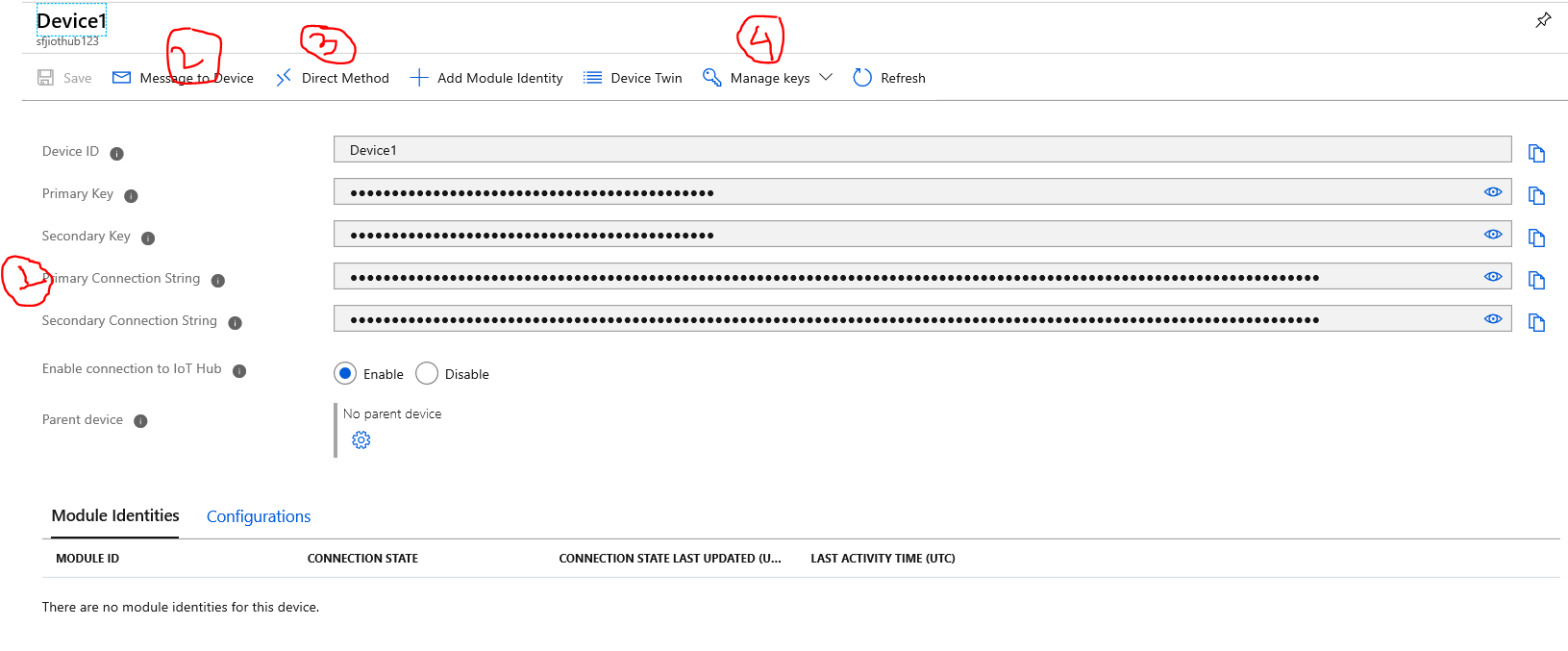
1. Provide the device Id, leave all other setting as is and click on “Save” for Symmetric key based authentication.



1. Once the add device operation is complete, you should be able to view your newly added device in the device list under “IoT Devices”.



1. Click on your device, it will take you to device details page.



On this page you have below details:

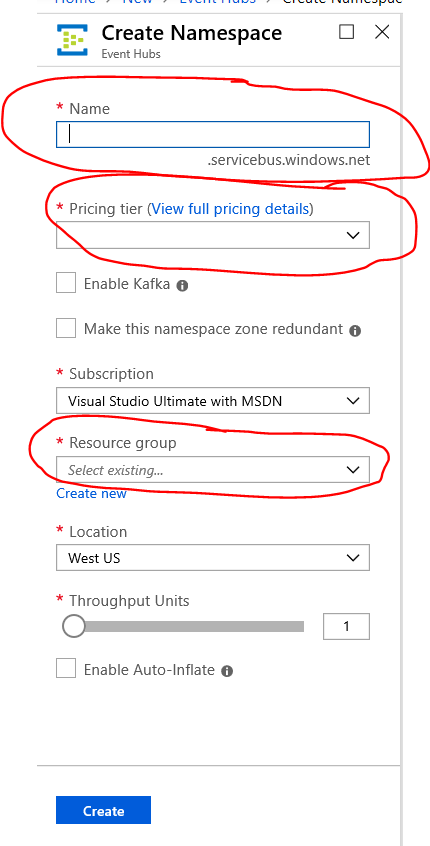
1. This is primary connection string that you will use in your device to connect to “IoT Hub”
2. When your device is connected, you can use this button to send messages to device.
3. When device is connected you can use this button to send messages to device through “Direct Method” option.
4. This button can be used to rotate or change the keys for device in case they are compromise or you want to rotate them. If you change the keys, the connection string will also change and you need to copy new connection string and place it in the device.

**Device**

1. After you have created a new device, copy the device connection string and place it in Device.py.
2. Run the Device.py, post which you should be able to see in the logs that device is connected to IoT hub and is sending messages.

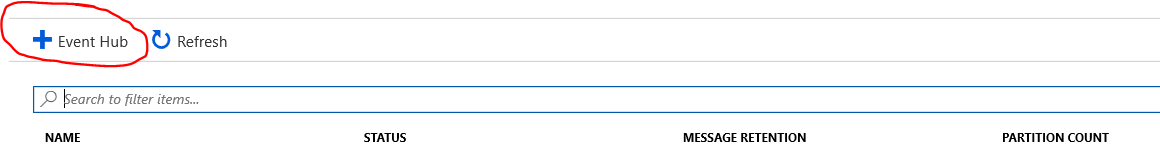
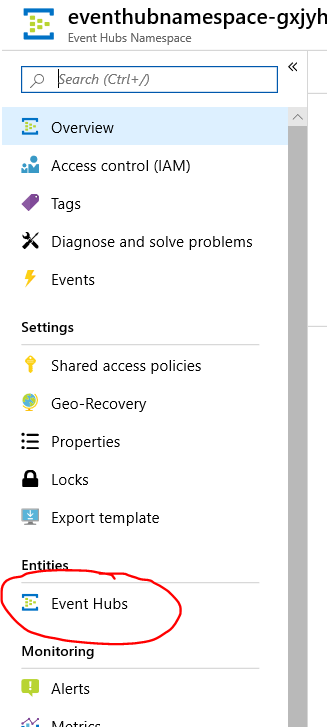
**Event Hub**

1. Click on “Create a Resource” from left pane as in step 2 of IoT Hub creation.
2. In the search box type “Event hubs” and select the first option.
3. Click on create.
4. Provide the details and click on Create.

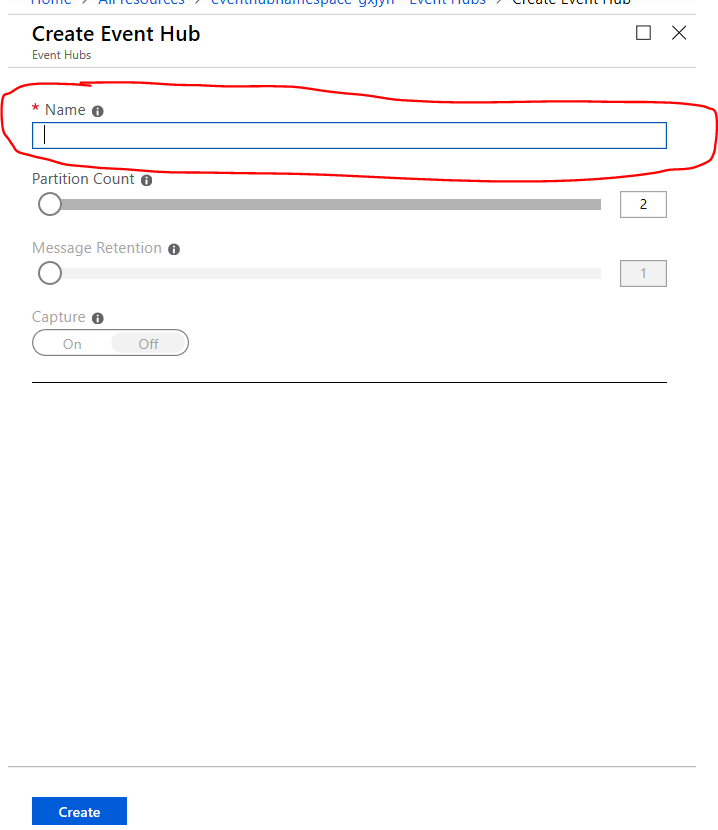


Wait for the resource to be deployed.

1. Once the resource is deployed, navigate to the resource.
2. Click on the “Event Hubs” from the left pane of the resource and click on “Add Event Hub”.



1. Provide the name of the event hub, leave the partition count as 2 only. The retention days setting lets you define for how long you want the message to be retained in the event hub.

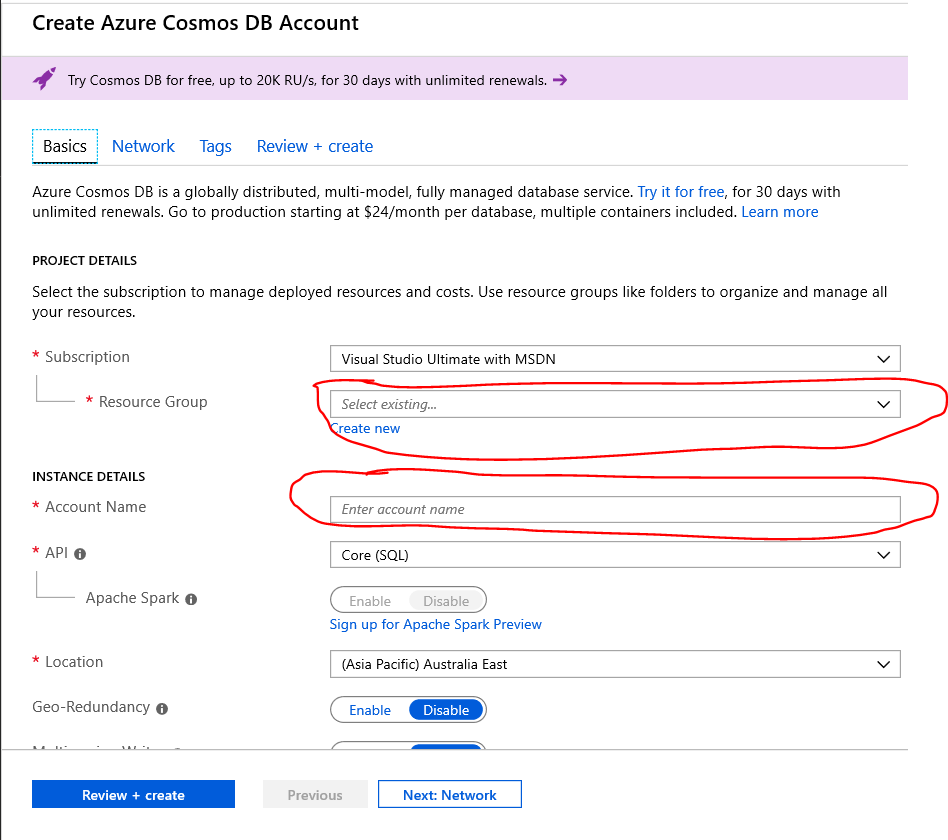


1. Wait for deployment of event hub to be completed. Once the deployment is done, you should see newly created in the event hubs list under the event hub namespace.

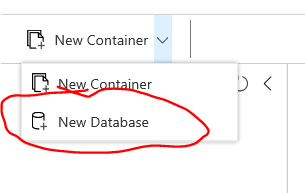
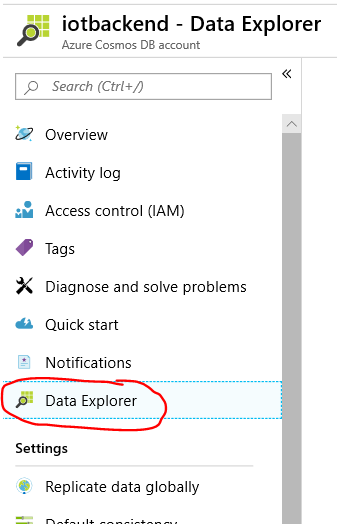


**Cosmos DB**

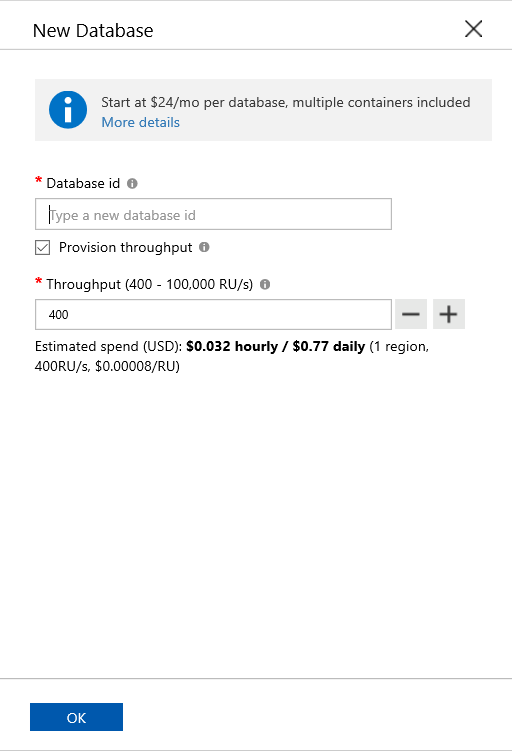
1. Click on “Create a Resource” from left pane.
2. Type “Azure Cosmos DB” in search bar and select the first option.
3. Click on Create and provide the details highlighted below.



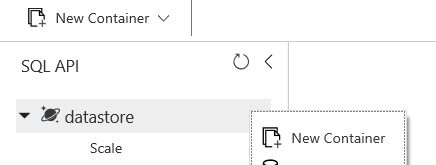
1. Click on “Review + Create” and then click on create.
2. Wait for resource to be deployed.
3. Once the resource is deployed navigate to “Cosmos DB” either from all resources or from “Go To Resource” button in the deployment completed notification.
4. Click on “Data Explorer” from the left navigation of Cosmos Db and click on “New Database” from the drop down.



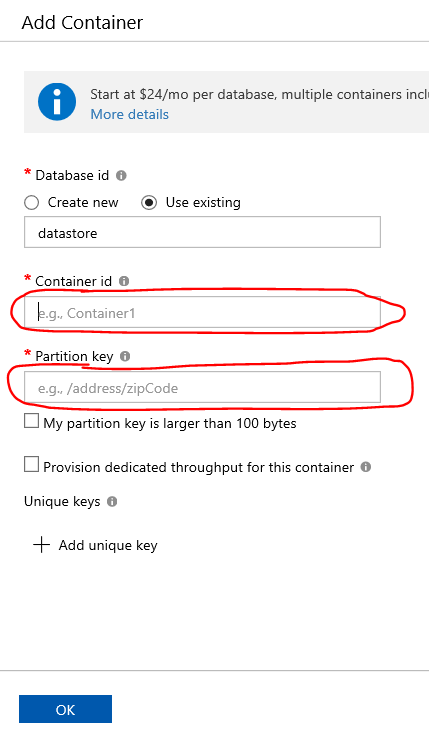
1. Provide a name for database and click on Ok.



1. Click on “…” from the database name and select “New Container” from the drop down.

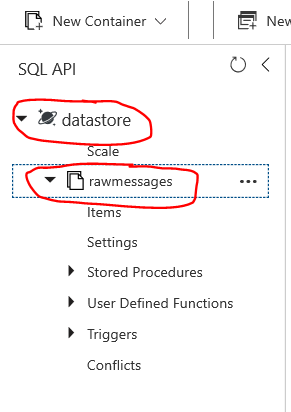


1. Provide the details for the container as below:



Give a container name and in Partition Key provide the value as “/DeviceId”. Click on Ok.

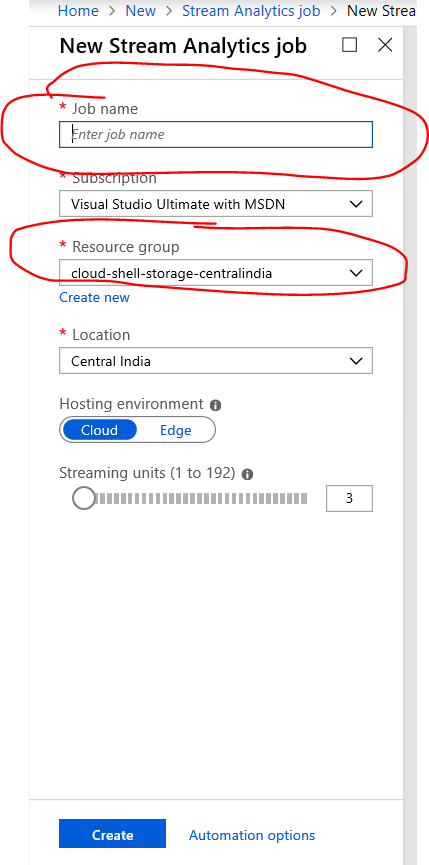
1. You should see the container under your database in the Data explorer.



Items option under the container will be used to view the data which is stored in this container.

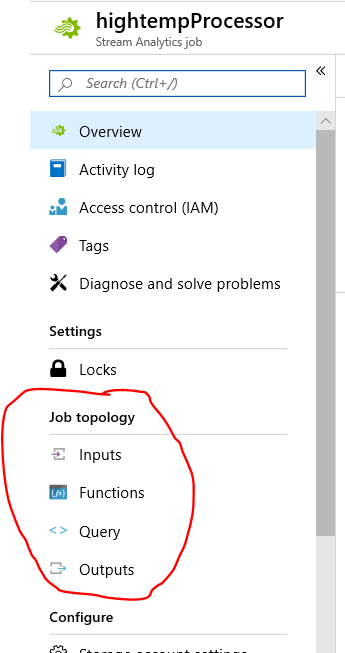
**Stream Analytics**

1. Click on “Create a Resource” from left pane.
2. Type “Stream Analytic Job” in search bar and select the first option.
3. Click on Create.
4. Provide the details and click on create.

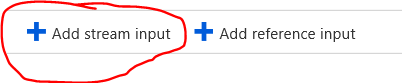
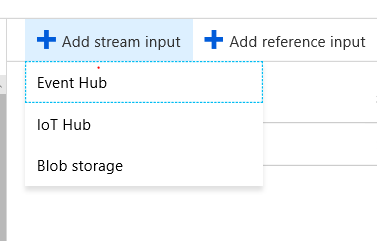


Leave the Hosting environment as “Cloud”. Streaming units are used to scale out, leave it to minimum for this workshop.

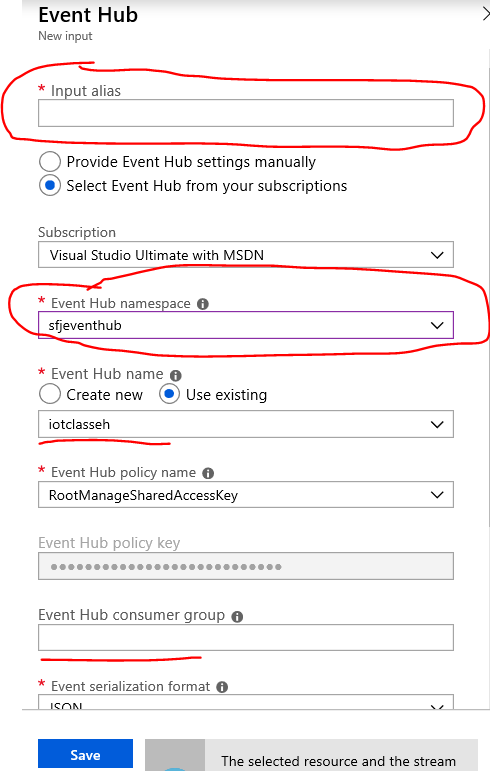
1. Let the deployment complete, and then navigate to the resource.
2. We need to set up input, output, query and function (if needed).



1. Click on inputs and click on Add Stream Input. From the Dropdown select eventhub.

1. Provide the details as below in the event hub window:



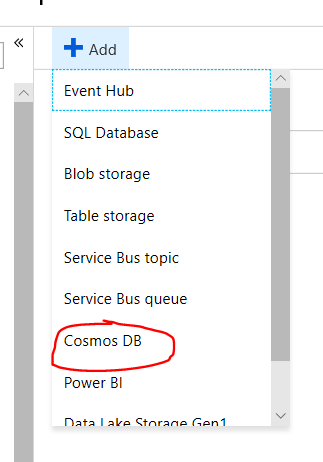
Provide a input alias, it will be used in stream analytic query to refer to this event hub.

From the Event hub namespaces drop down, select the event hub namespace you created. If you have only one event hub namespace in your subscription, it will be pre populated.

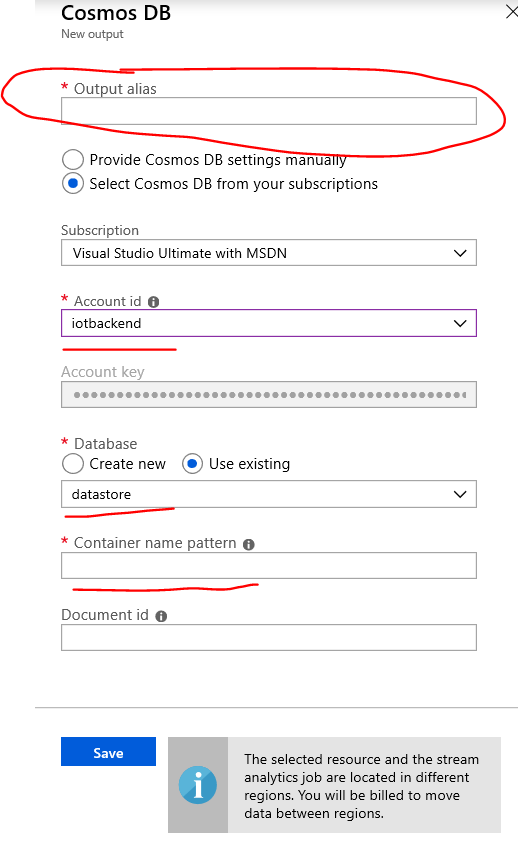
In the event hub name, you should select the event hub you created in this event hub namespace. If you have only one event hub in that namespace, it will be pre-populated.

Leave consumer group black. Blank means that it will be use “$Default” consumer group. If you have created another consumer group and you want to use that, provide the name of that consumer group here.

1. Click on save, it will save the details and test the connection to the event hub, the test connection should result in success. If there is a failure, look at the error and fix the details.
2. Now click on outputs from the left navigation, click on ADD and select Cosmos DB from the drop down.



1. Provide the details as below:



Provide a output alias, it will be used in stream analytic query to refer to this cosmos db.

Select your “cosmos db” resource name in the “Account Id” dropdown. If you have only one cosmos account in your subscription, it will be prepopulated.

Select the database name that you created in the cosmos db, if you have created only one database it will be pre-populated.

Give the name of the container that you created in Cosmos db step 10. You can find it out using Cosmos DB step 11.

Click on Save.

1. Click on save, it will save the details and test the connection to the cosmos db, the test connection should result in success. If there is a failure, look at the error and fix the details.
2. Click on Query from the left pane.
3. Replace the text in query as below:

SELECT

\*

INTO

[<<Type in the alias you have given for output>>]

FROM

[<<Type in the alias you have given for input>>]

For example if your input alias is eventhubinput and output alias is cosmosoutput, your query should look like below:

SELECT

\*

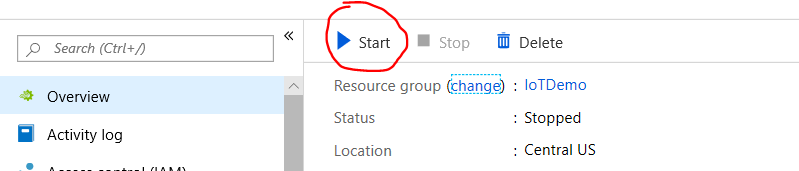
INTO

[eventhubinput]

FROM

[cosmosoutput]

1. Click on save at the top.
2. After query is saved, click on “Overview” from the left hand pane.
3. Click on start



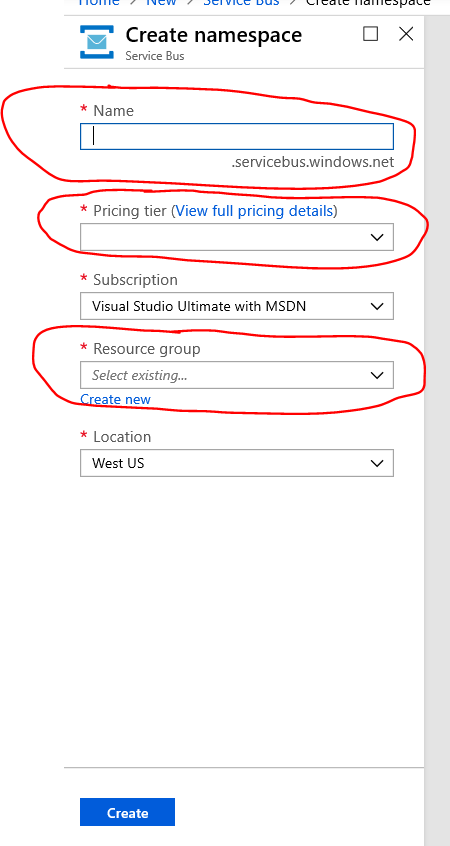
1. In the start job window, click on start at the bottom.



1. You can use custom option to run the job from past for future point in time. If you select Custom, if will ask you to select a date and time from where you want the job to start picking messages.
2. Wait for the message saying job started successfully.
3. Once you have this message, your Job is up and running and can process messages coming to event hub and insert them into the cosmos db.

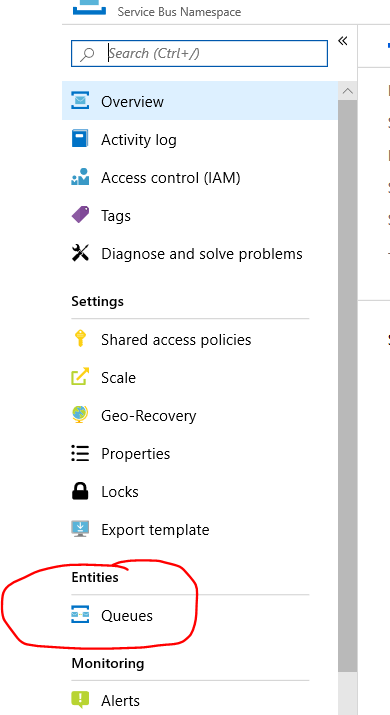
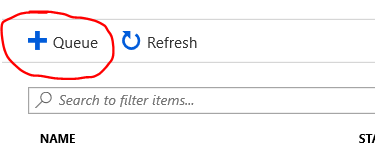
**Service Bus Queue**

1. Click on “Create a Resource” in the left side pane.
2. Type “service bus” in the search box, select first option and click Create.
3. Provide the details as below:

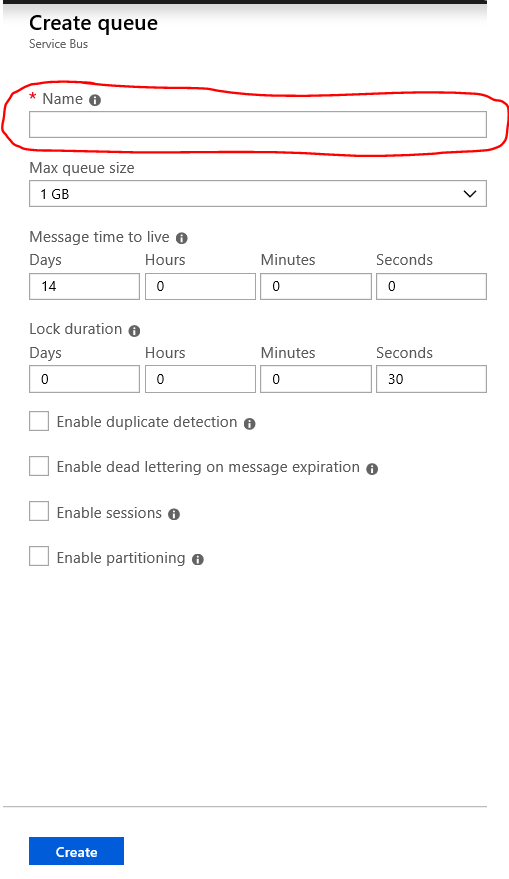


Provide a service bus name, select “Basic” from the pricing tier and select the resource group. Then click on “Create”

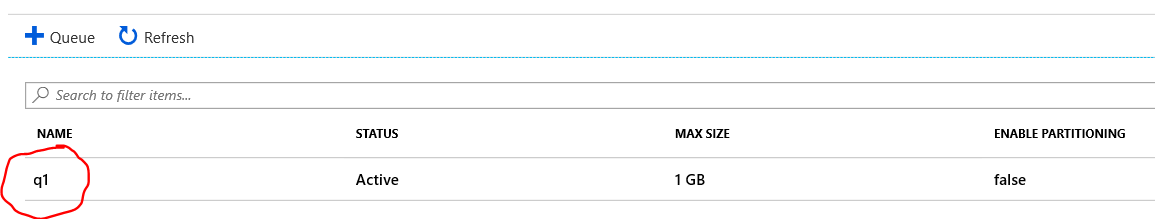
1. Wait for the deployment to complete. Once the deployment is complete navigate to the Service Bus.
2. Click on Queue from the left side navigation and click on Add Queue.

1. Provide a queue name and click on create.

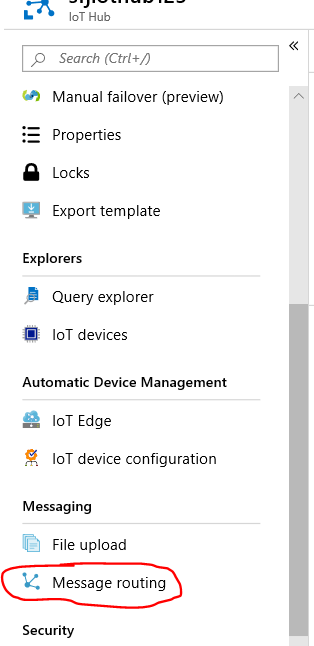


1. Once queue is created you should be able to see this in the list on the “Queues”.

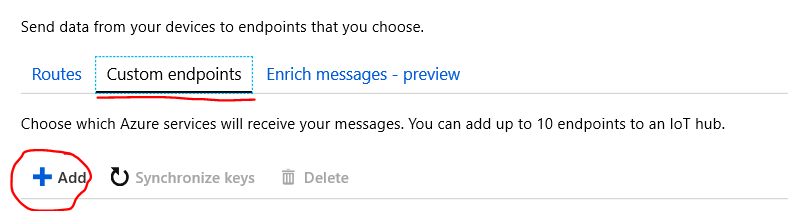


**IoT Hub Routing**

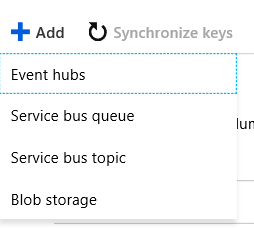
1. To set up IoT hub routes, navigate to your Iot hub.
2. Click on “Message Routing” under Messaging section on the left pane.



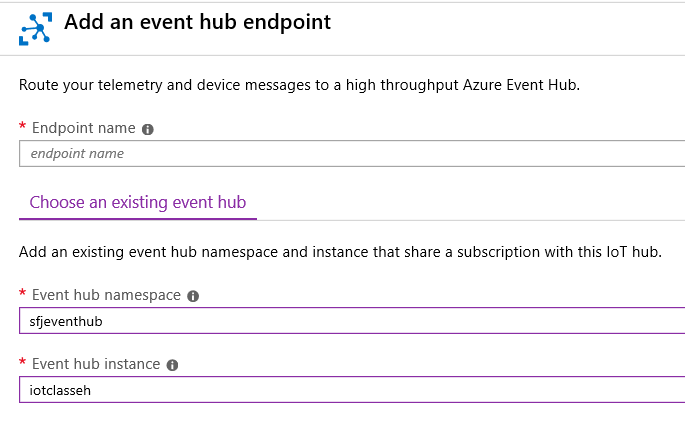
1. Select the second tab “Custom Endpoints”, and click on Add.



1. From the drop down, select event hubs.



1. Provide the details as below:

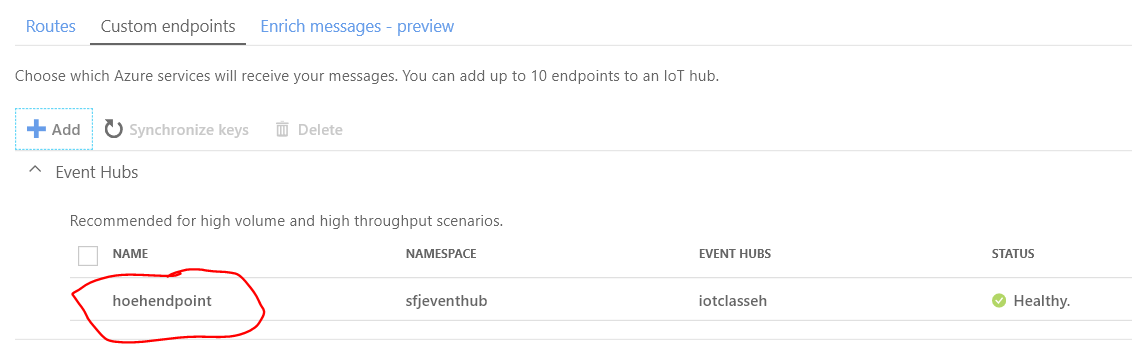


Provide an endpoint name which will be used in routing tab to create route.

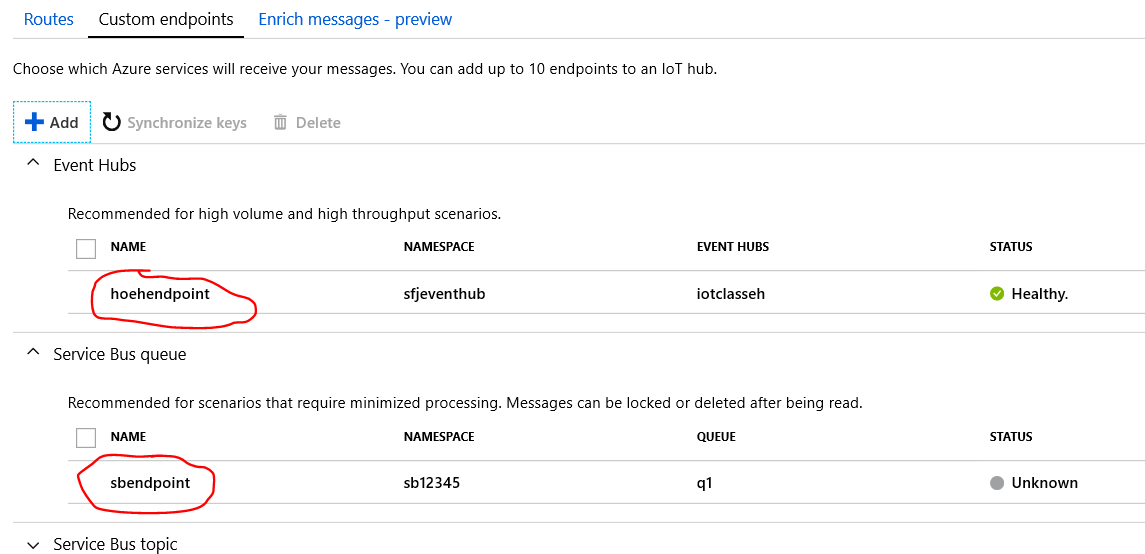
Select your event hub namespace and event hub that you have created earlier.

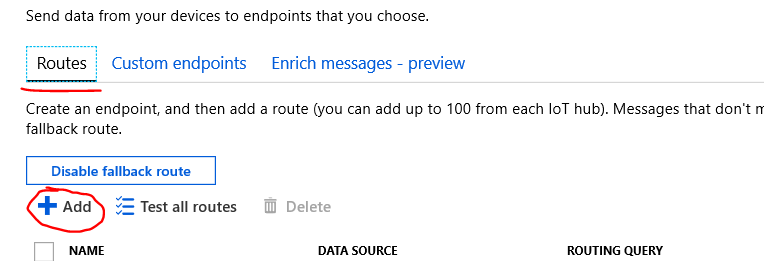
Click on create at the bottom.

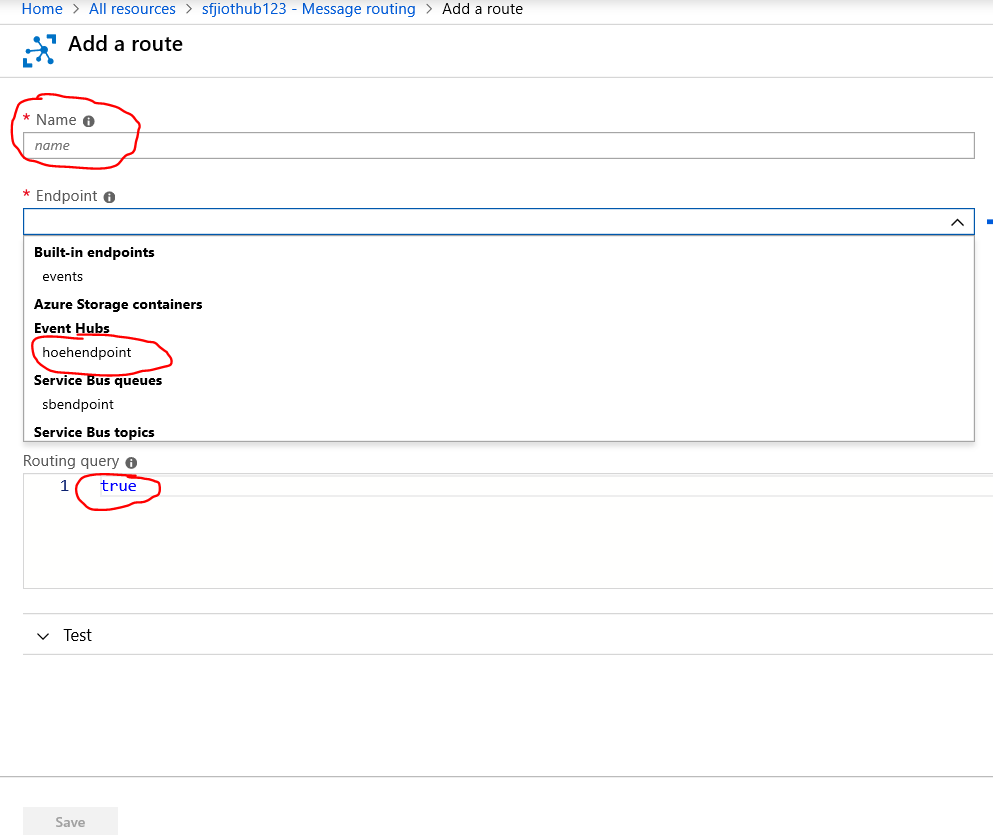
1. Once it is added, you should see this endpoint under eventhubs in the “Custom endpoints” tab as below:



1. Perform steps 4-6 for adding the “Service Bus queue” you have created.
2. After that you should see both the custom endpoints as below:



1. Click on the Routes tab, and click on Add. 
2. Provide details as below:



Provide a name for the route.

From endpoint drop down, select the “event hub” endpoint you created in previous step.

In the routing query, leave it as “true”, which means all the incoming messages will be send to this event hub.

Click on Save at the bottom.

1. You should see this route in the list under “Routes” tab.
2. Again click on Add for adding another route for service bus queue.
3. Provide a route name (ex. HighTempRoute), and select the service bus endpoint from the endpoint drop down.
4. In the routing query, replace the “true” with below text:

temperatureAlert = 'true'

This will route only messages coming in with temperature alert flag as true to service bus end point.

**Azure Function**

using System;

using System.IO;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Azure.WebJobs;

using Microsoft.Azure.WebJobs.Extensions.Http;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

using Microsoft.Azure.Devices;

using System.Text;

namespace Company.Function

{

public static class C2DMessage

{

static ServiceClient serviceClient;

static string connectionString = "IoT Hub Service connection string”;

[FunctionName("C2DMessage")]

public static async Task<IActionResult> Run(

[HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = null)] HttpRequest req,

ILogger log)

{

serviceClient = ServiceClient.CreateFromConnectionString(connectionString);

var commandMessage = new Message(Encoding.ASCII.GetBytes("Turn off heater."));

await serviceClient.SendAsync("Your device Id", commandMessage);

return (ActionResult)new OkObjectResult(string.Empty);

}

}

}

Please follow below link for creating and deploying azure function using VS Code:

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-function-vs-code>