Indoor real-time position tracking using Azure Maps and IoT Hub

Deployment Instructions

Clone repo

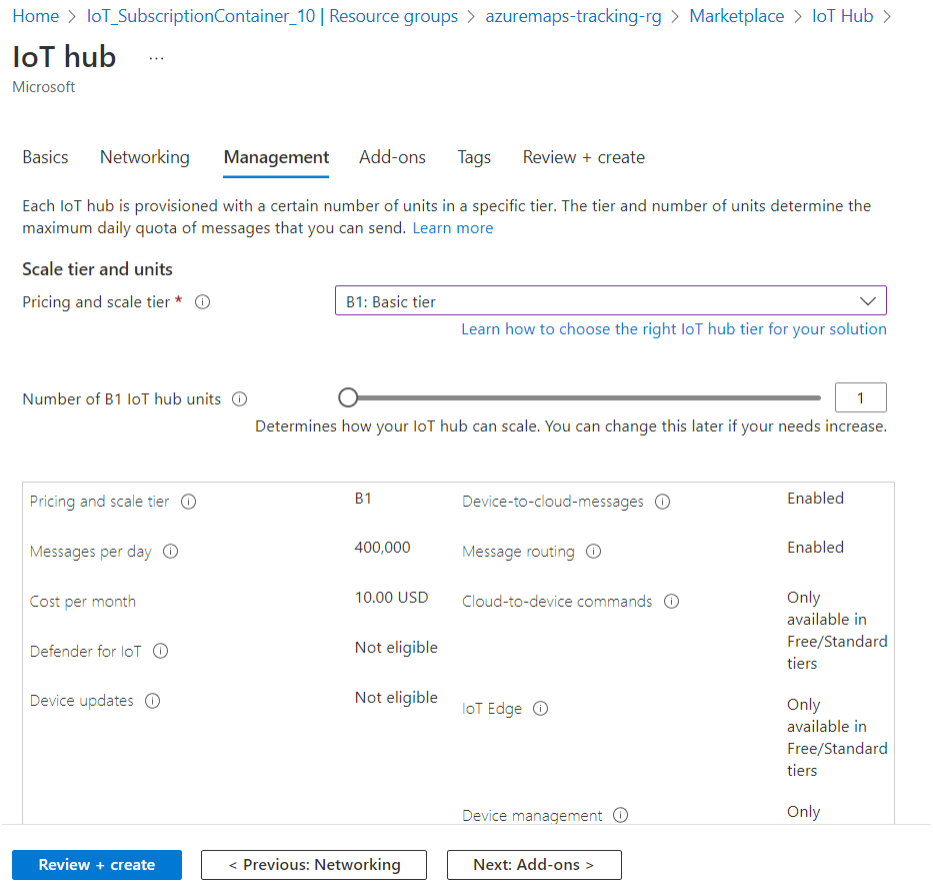
https://github.com/Azure-Samples/azuremaps-indoor-realtime-position-tracking

Create resource group in Azure Subscription

To isolate all resources associated with this deployment, create a new resource group in the Azure portal.

Create IoT Hub instance

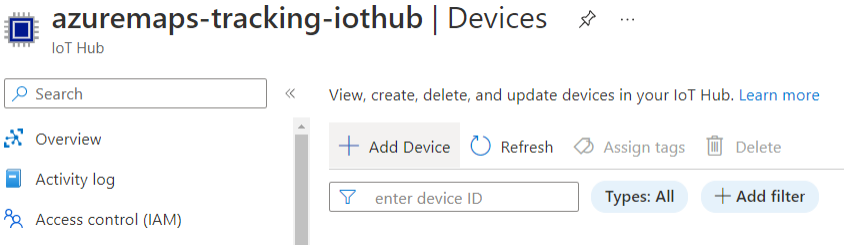
Create an IoT Hub instance in the Azure portal and make sure to select either the basic tier or higher:



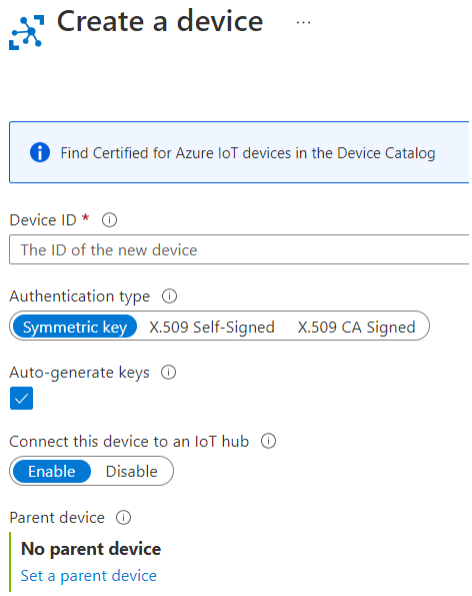
This is important because we are going to be using routing and the free tier only allows 1 route. Plus, the basic tier is also upgradable if needed.

Add Device (e.g., smartphone)

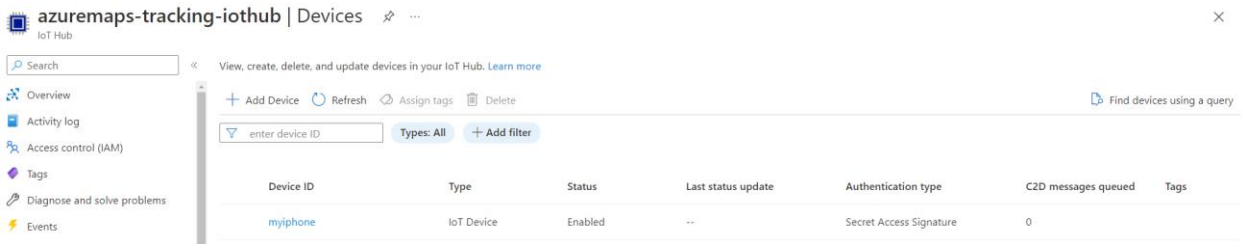
In IoT Hub, select add a new device:



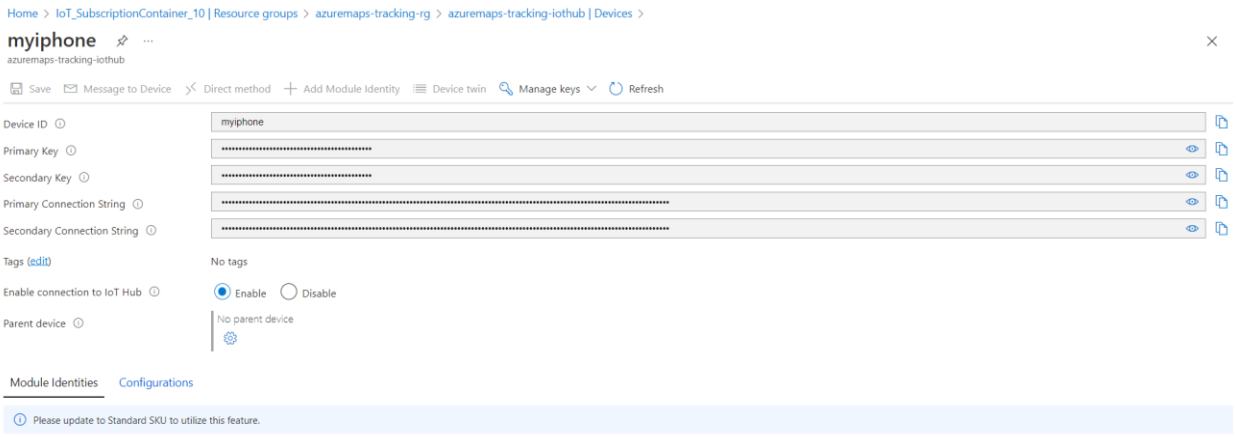
And create your device. You can call it “myiphone” device, for example, since we are going to be using the IoT Plug and Play Application from the Apple (or Google) store.



Once the device is created, click on that device:



and copy the corresponding primary connection string:

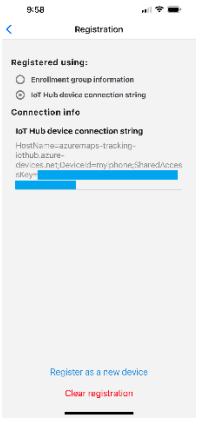


You are going to need this connection string for linking the smart phone application with this device in IoT Hub.

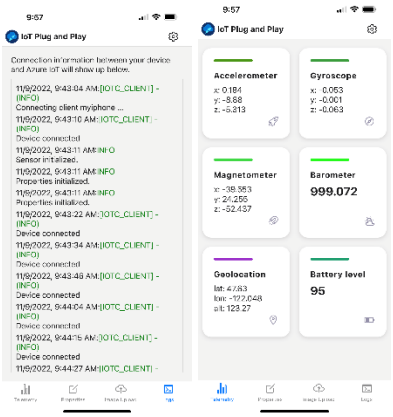
Install IoT Plug and Play Application

On your smartphone, install the “IoT Plug and Play” app. You can find it in the Apple or Google store.

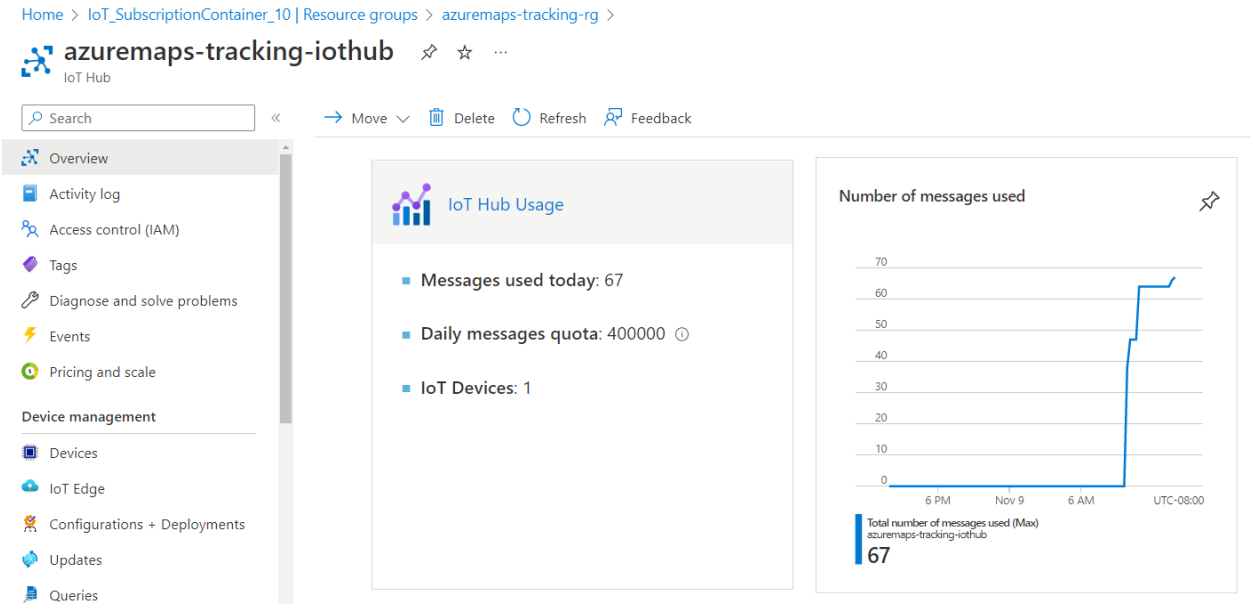
Go to Settings and Registration and paste the connection string from the previous step:



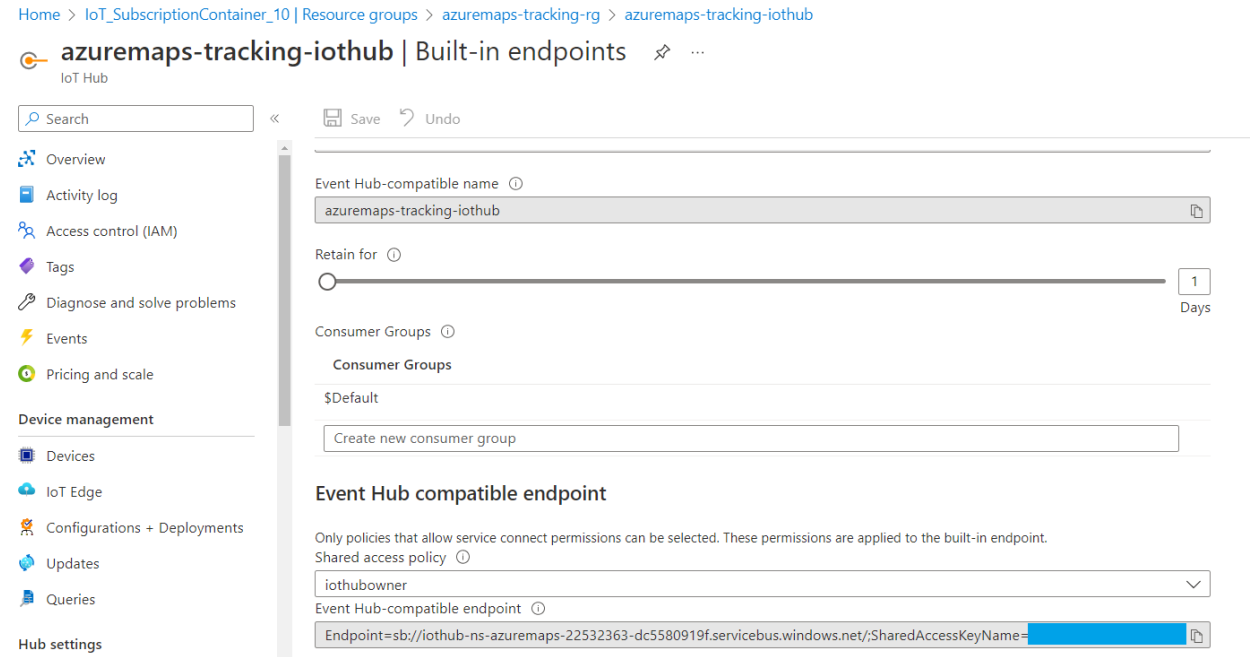
And make sure that you can confirm that it is connected to Azure by checking the “Logs” tab in the application:



You should now be receiving messages from this device in IoT Hub. You can confirm this by checking the “Overview” page and looking at “Number of messages used” and “Connected Devices”.

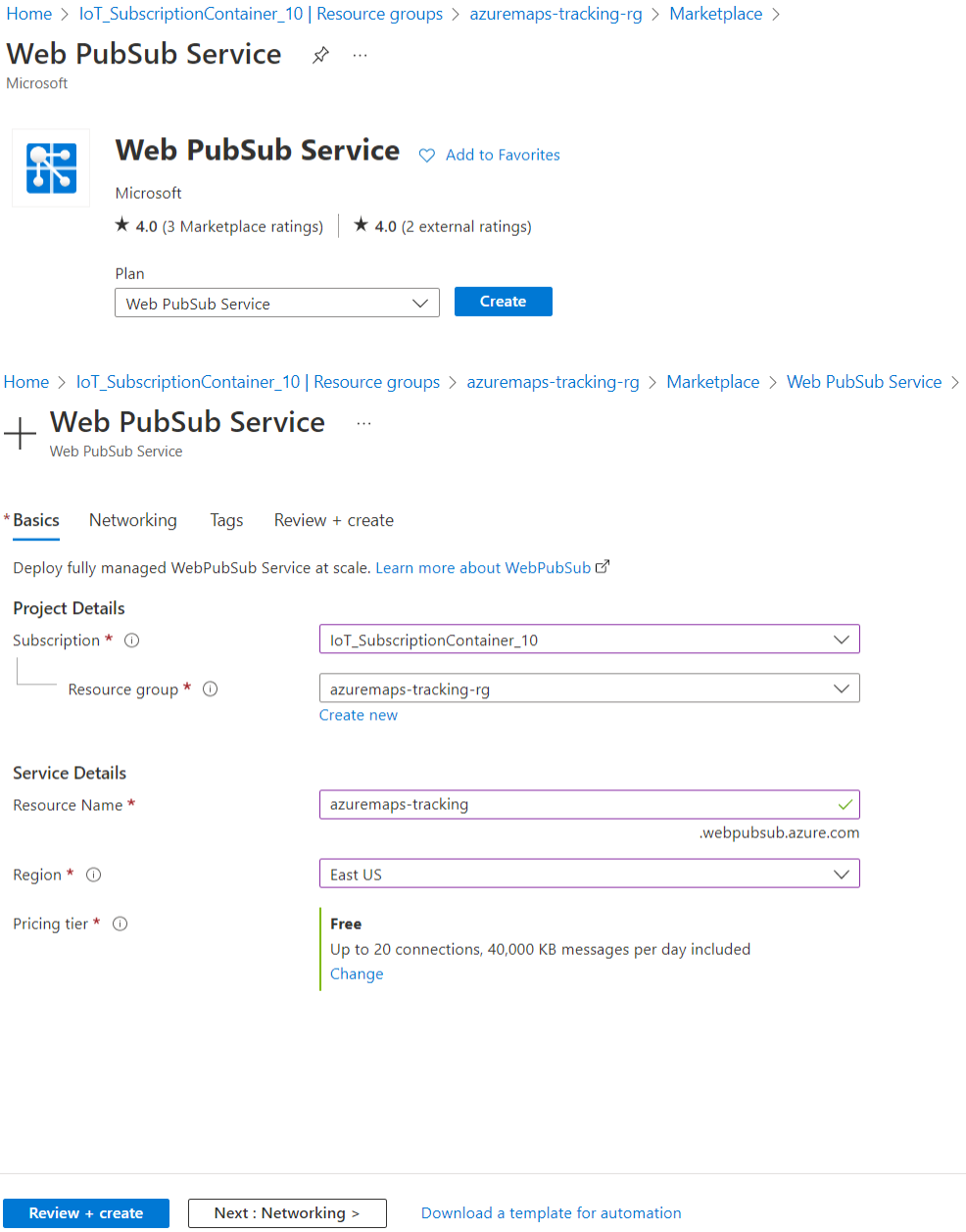


Next, go to the “Built-in endpoints” under “Hub settings”, scroll down, and copy the “Event Hub-compatible endpoint” connection string. We will use this value later to trigger the Function App whenever new messages arrive to IoT Hub via the built-in Event Hub endpoint.



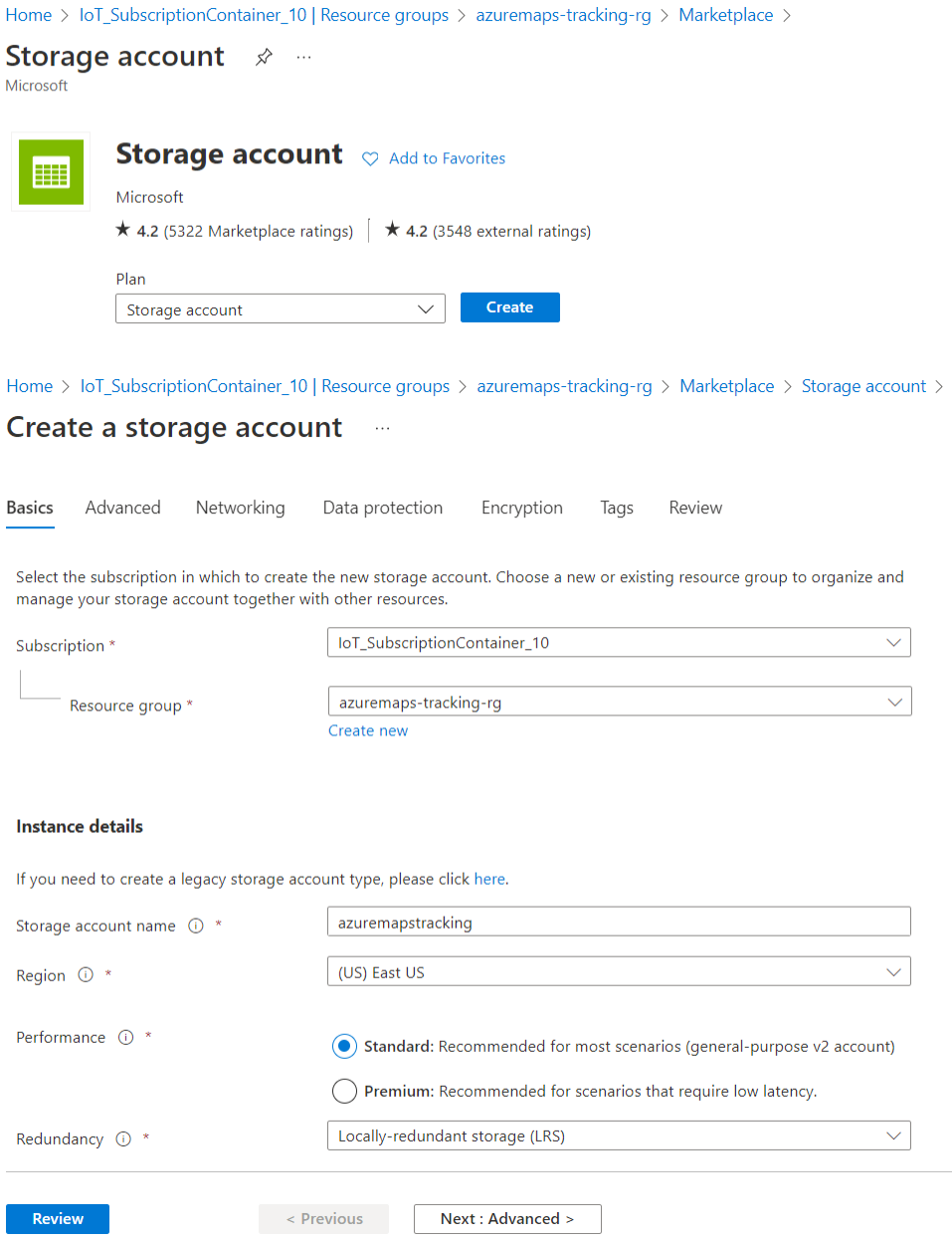
Finally, take note of the “Event hub-compatible name” as well.

Create Azure Web PubSub



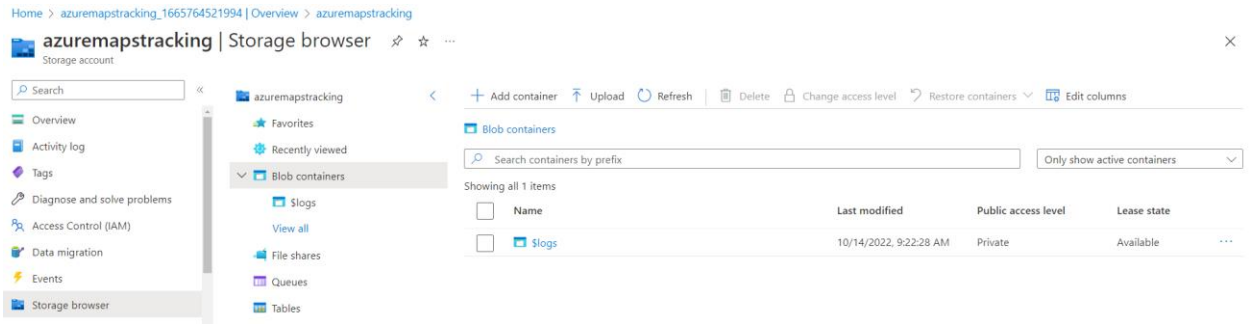
Under “Keys”, take a note of the connection string for this service.

Create Storage Account

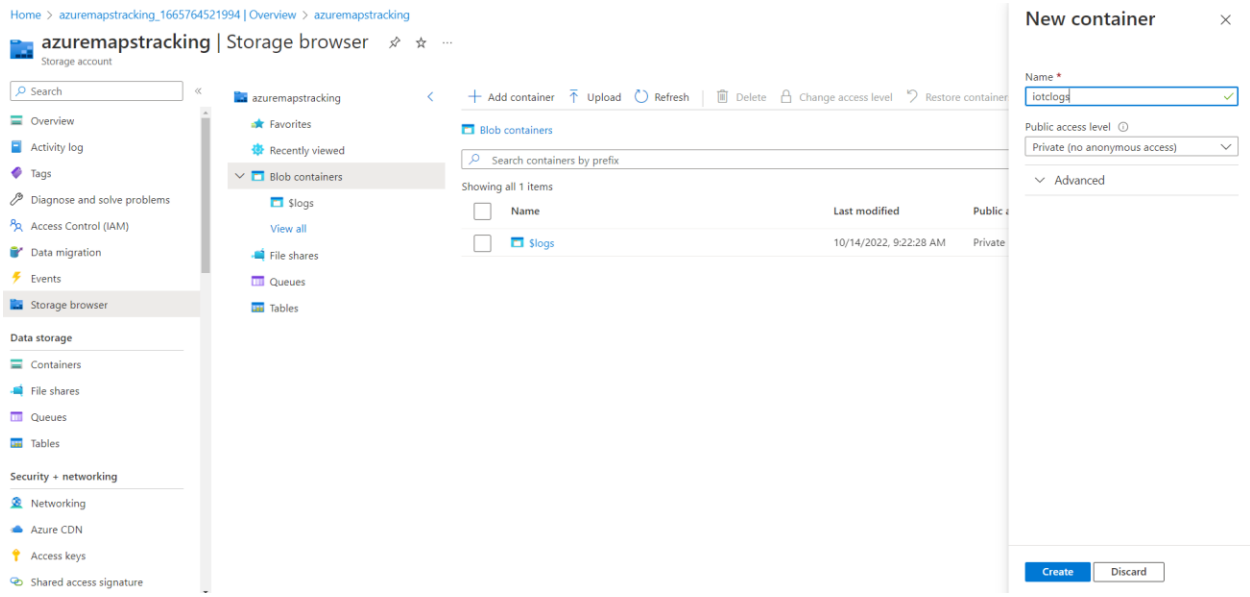


Save the value for the connection string under “Access keys” since you will need it later.

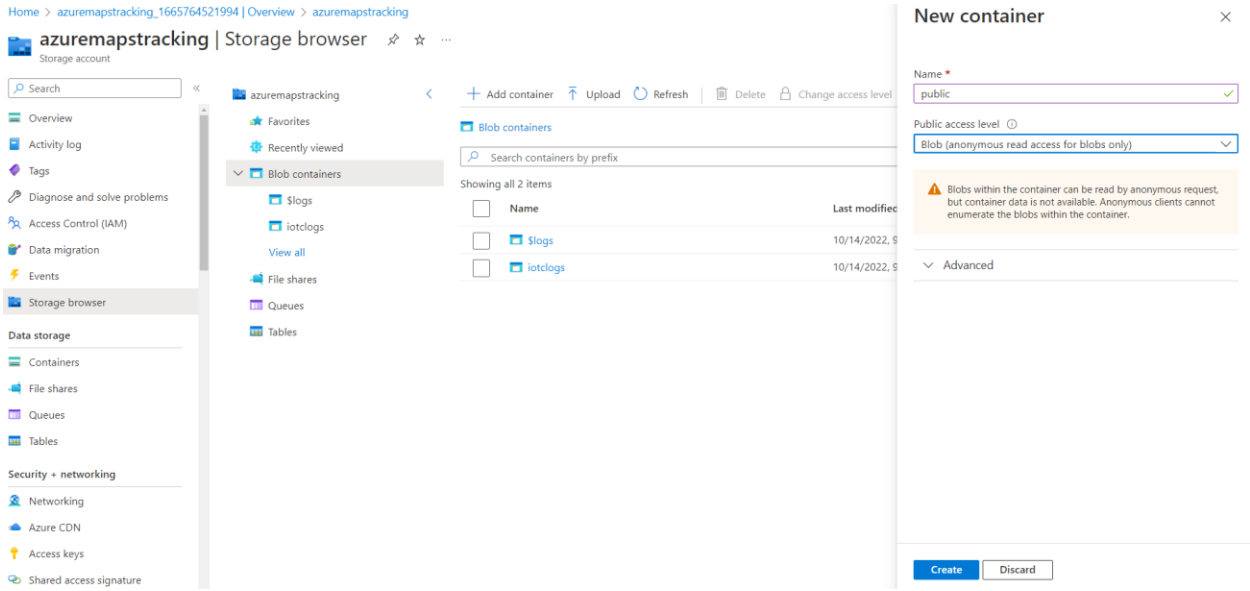
Under “Storage browser”, click on “Blob containers”:



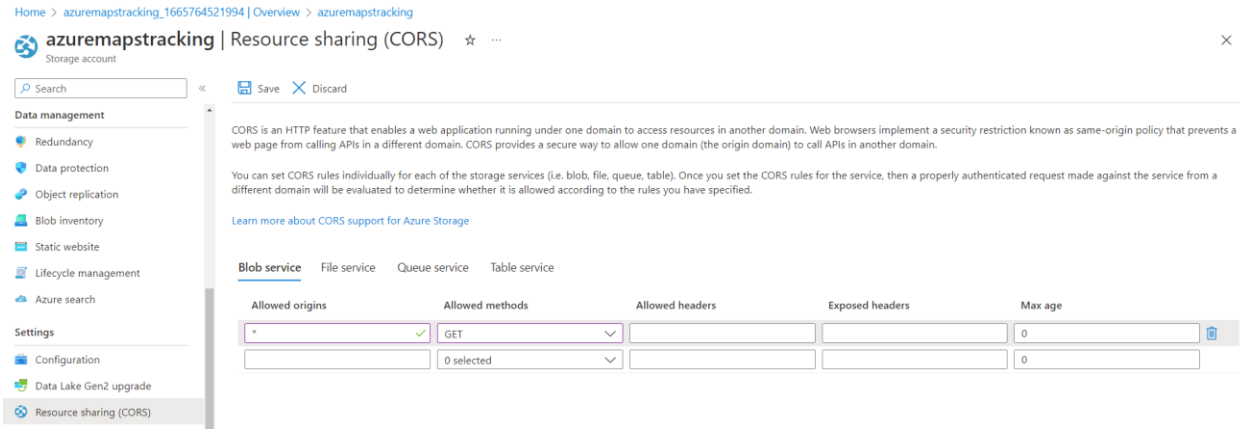
Create a new blob container called “iotclogs” (private):



Create another blob container called “public” with anonymous read access:

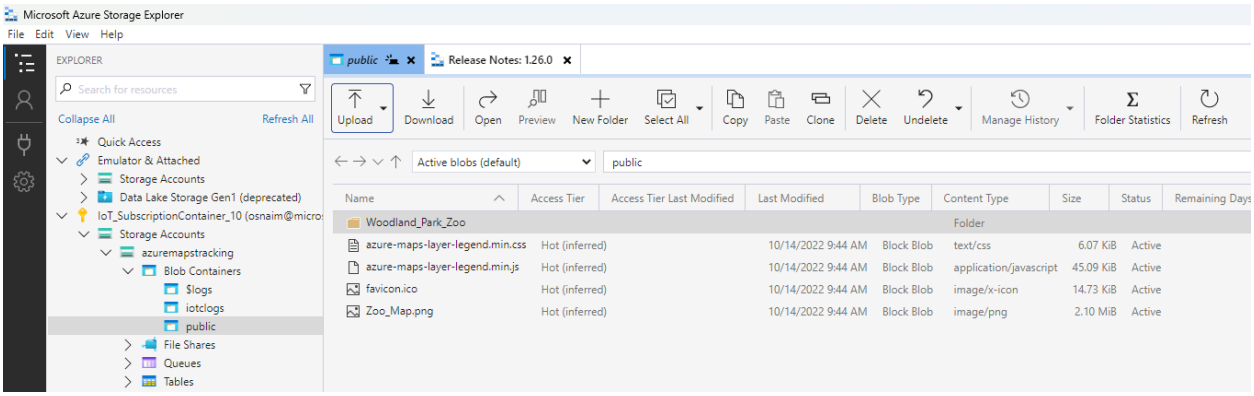


Enable CORS:

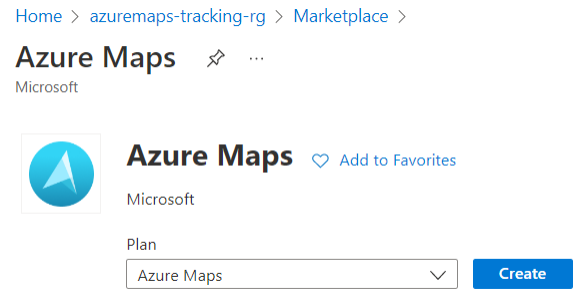


Install “Azure Storage Explorer”: https://azure.microsoft.com/en-us/features/storage-explorer/

Using Azure Storage Explorer, upload the content of the “public” folder in your repo to the corresponding blob container.



Create Azure Maps instance



Home > azuremaps-tracking-rg > Marketplace > Azure Maps >

Create an Azure Maps Account resource ...

Basics | Advanced | Identities | Tags | Review + create

Azure Maps is a collection of geospatial services and SDKs that use fresh mapping data to provide geographic context to web and mobile applications. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : IoT_SubscriptionContainer_10

Resource group * : azuremaps-tracking-rg

Create new

Instance details

Name * : azuremaps-tracking

Region * : East US

Pricing tier * : Gen2 (Maps and Location Insights)

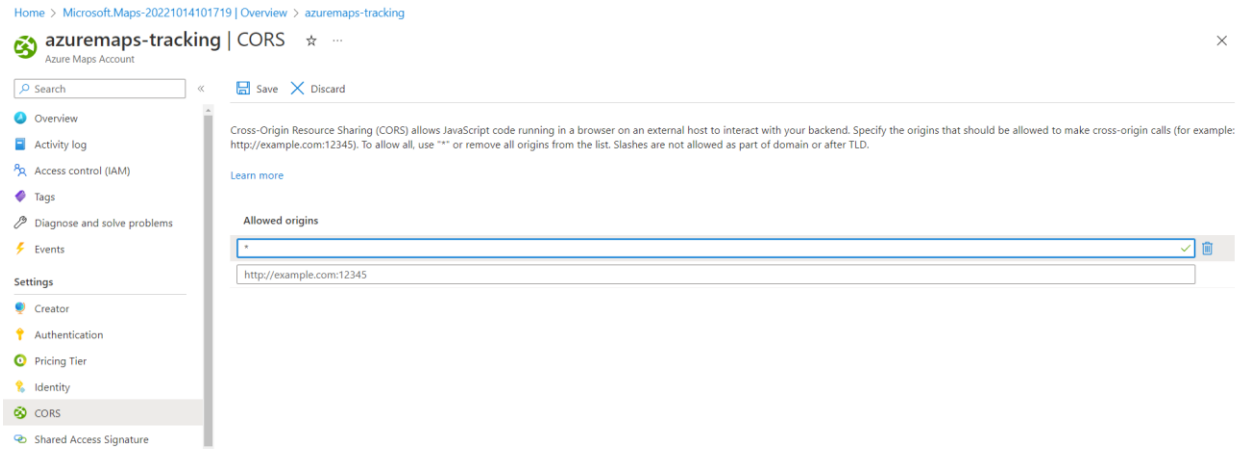
View full pricing details

Terms

Azure Maps shares customer-provided address/location queries ("Queries") with third party TomTom for mapping functionality purposes. Queries are not linked to any customer or end-user when shared with TomTom and cannot be

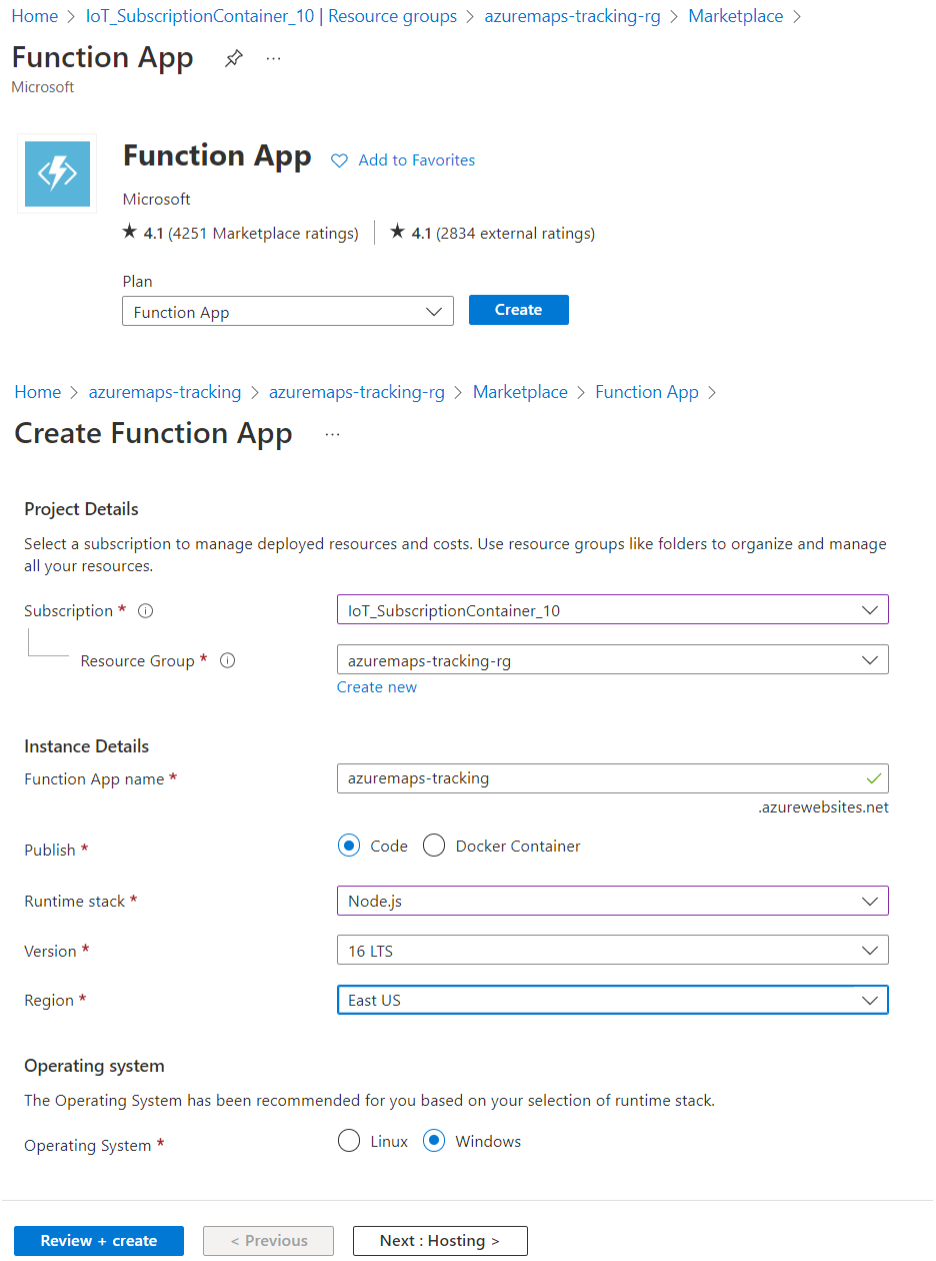
Review + create | < Previous | Next : Advanced >

Enable CORS:

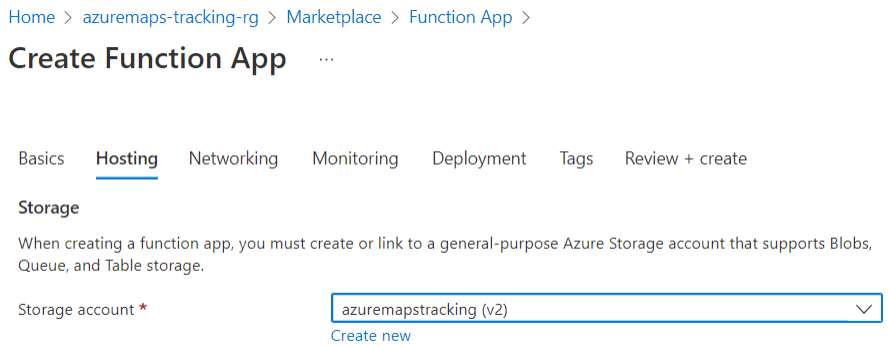


Save the primary key Azure Maps, which you can find under "Authentication".

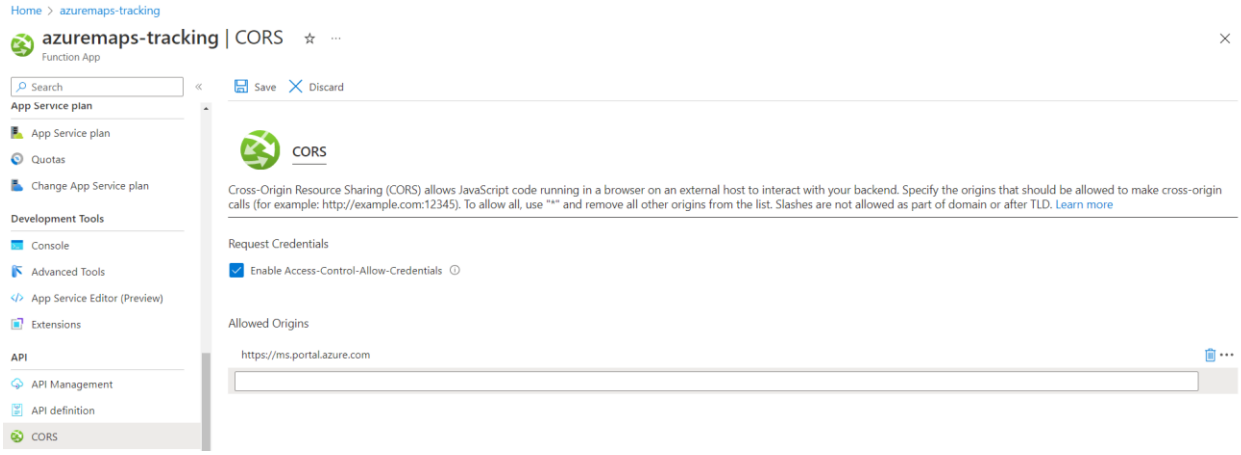
Create Azure Function



And make sure you use the same storage account we created previously:

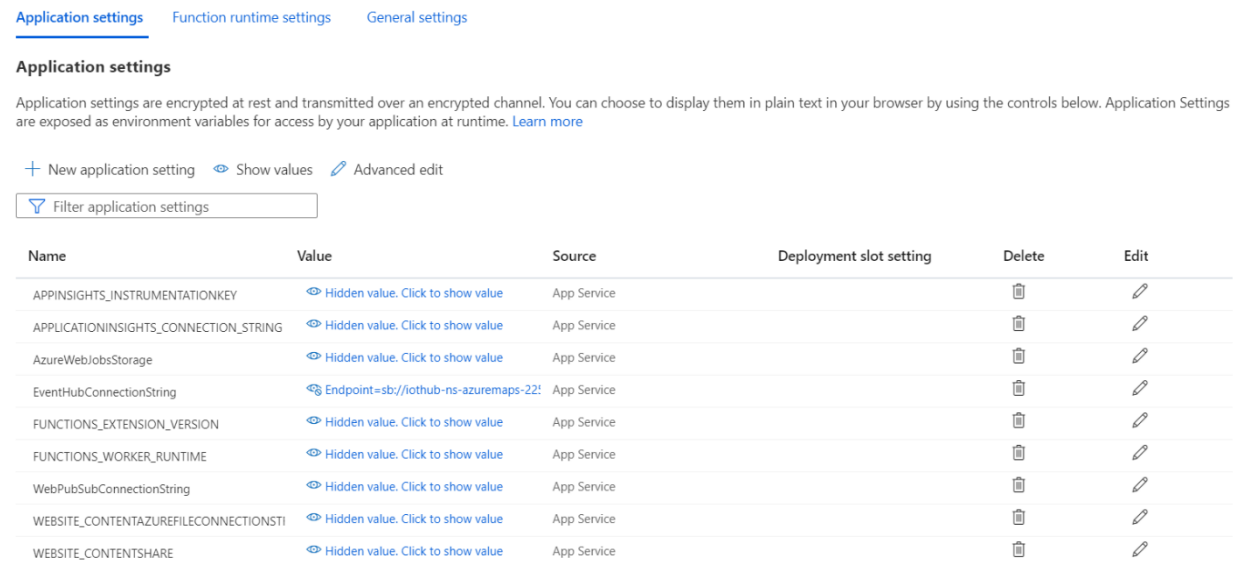


Once the Function App is created, enable CORS:

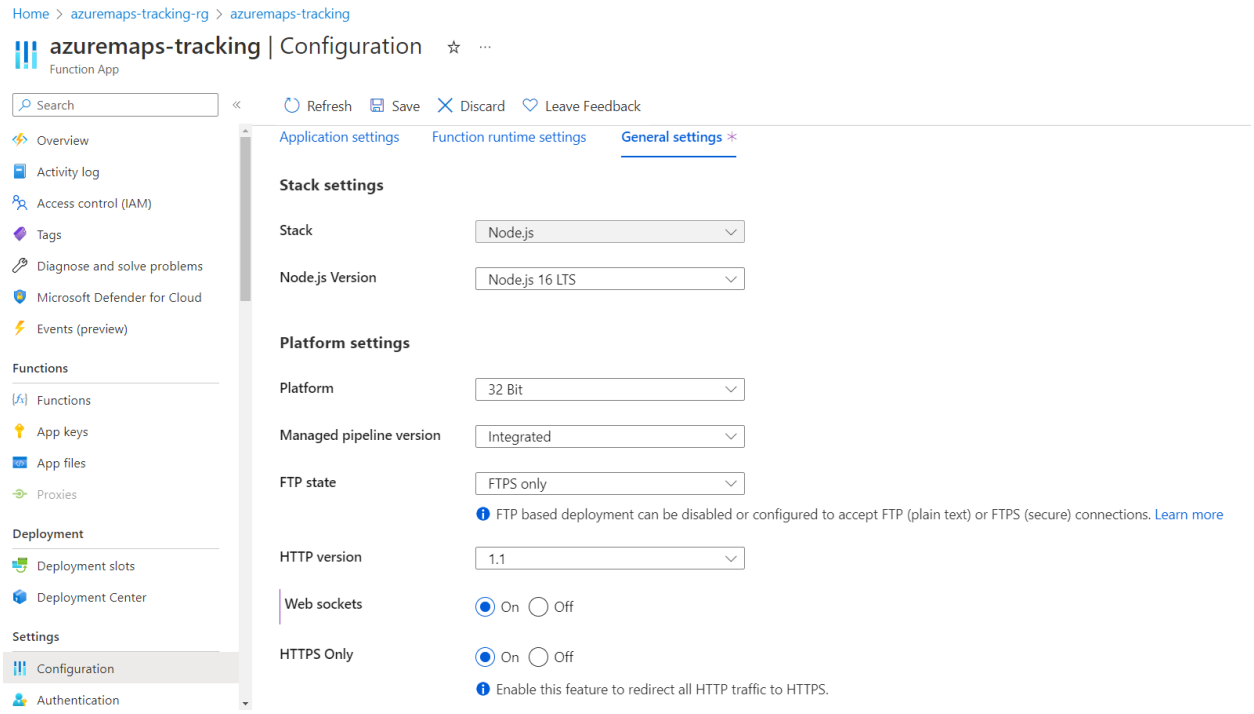


Then, under “Configuration”, add a new entry for “WebPubSubConnectionString” with the corresponding value that we saved previously.

Also, add another entry for “EventHubConnectionString” and use the value we saved when we created the IoT Hub instance.



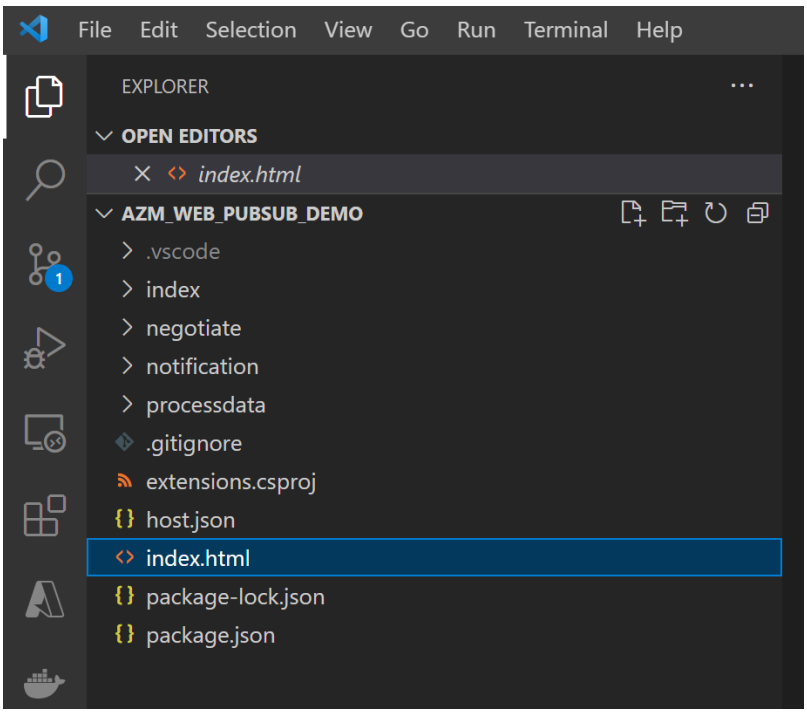
Also, under “Configuration”, enable Web sockets:



Modify Function App

We will now modify the Azure Function by deploying the code we cloned from the repo.

Open the following folder using VS Code (**AZM_WEB_PUBSUB_DEMO**, which you can find under **realtime-azuremaps-update-iothubdemo\AzM_Web_PubSub_Demo-v02**):



Let's start with "index.html" in the main folder.

First, replace "<YOUR-BLOB-STORAGE-URL>" with the corresponding value, which you can get from Settings/Endpoints/Blob service in the Azure portal. There should be 6 instances of it that you need to replace in this file.

For example:

```
<html>
    <head>
        <title>Azure Maps Web PubSub</title>
        <link rel="shortcut icon" href="https://azuremapstracking.blob.core.windows.net/public/favicon.ico"/>
        <meta charset="utf-8">
```

Then, replace "<YOUR-AZURE-MAPS-KEY>" with the map key you had saved previously.

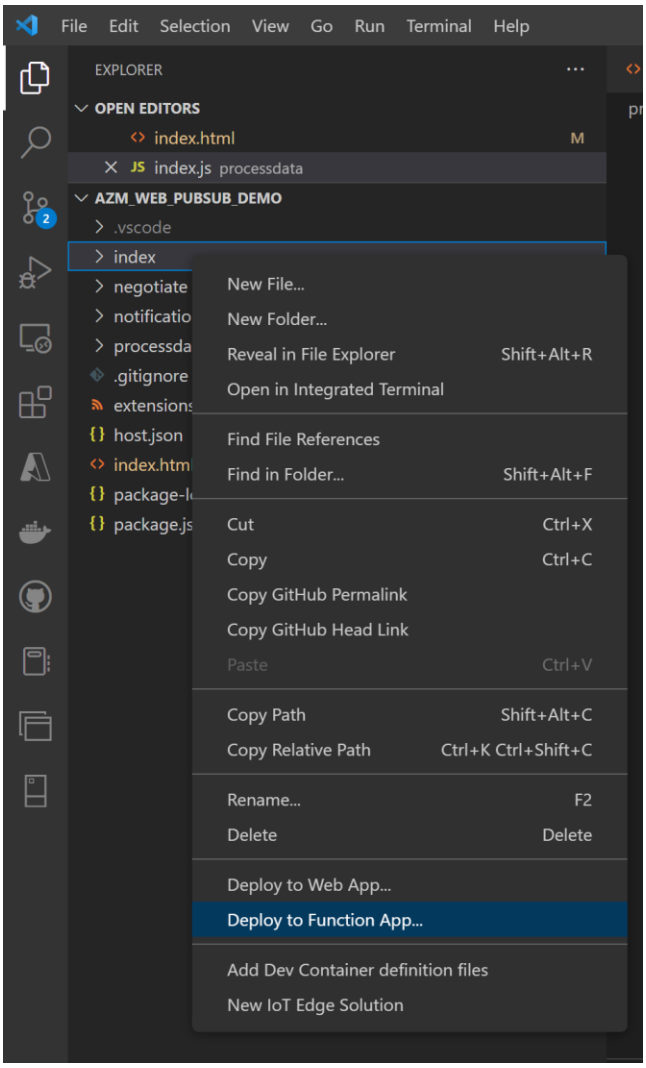
Finally, edit the **function.json** file under the "notification" folder and add the corresponding Event Hub name that we saved when we created the IoT Hub instance. For example:

src > realtime-azuremaps-update-iothubdemo > AzM_Web_PubSub_Demo-v02 > AzM_Web_PubSub_Demo > notification > {} function.json >

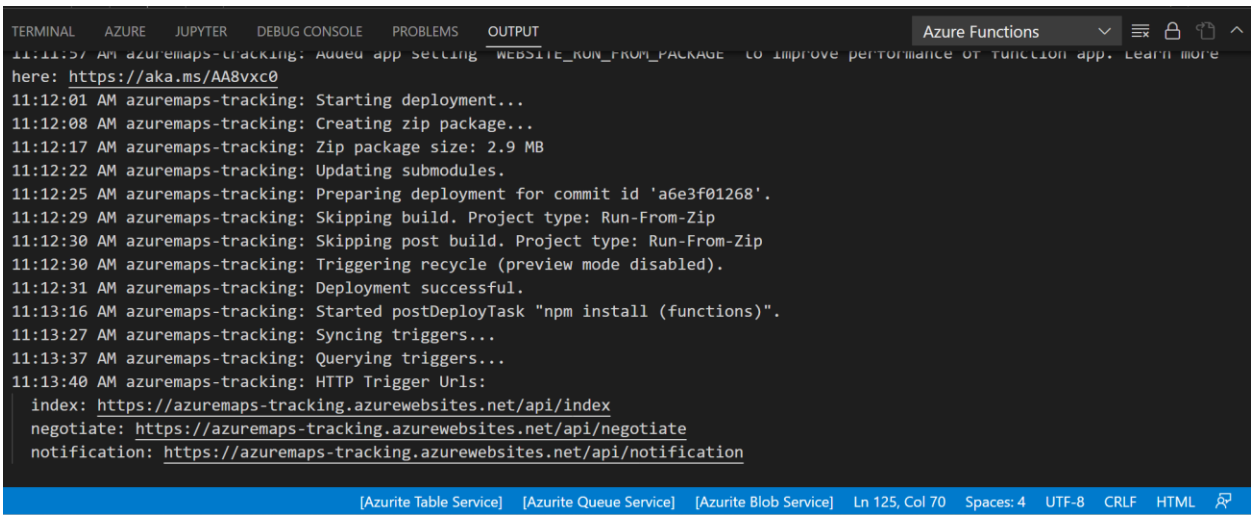
```
{
  "bindings": [
    {
      "type": "eventHubTrigger",
      "name": "myEventHubMessage",
      "direction": "in",
      "eventHubName": "azuremaps-tracking-iothub",
      "connection": "EventHubConnectionString",
      "cardinality": "many",
      "consumerGroup": "$Default",
      "dataType": "string"
    },
    {
    "type": "webPubSub",
    "name": "actions",
    "hub": "notification",
    "direction": "out"
    }
  ]
}
```

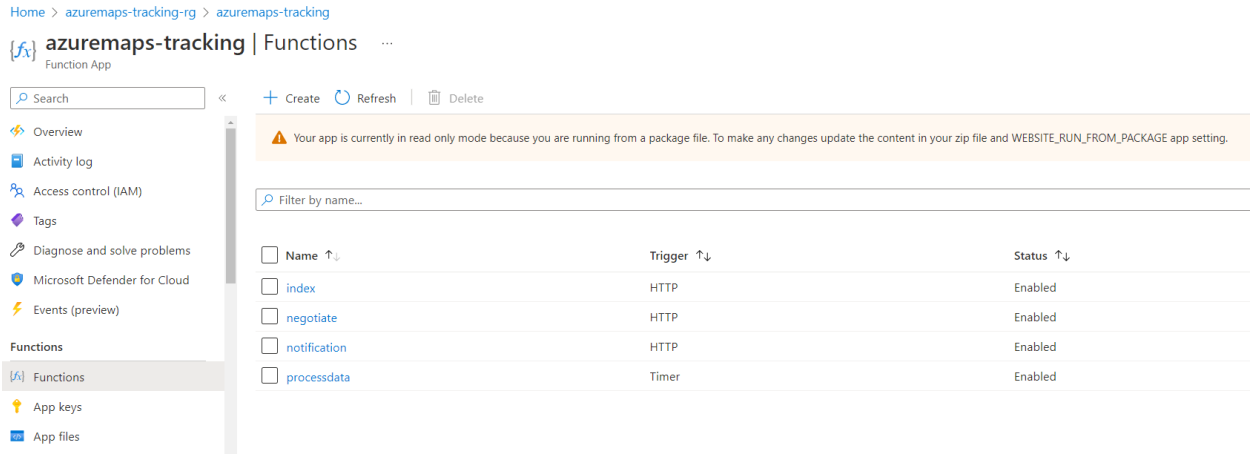
Deploy Function App to Azure

We now need to deploy these functions to the Azure Function we created previously. We will do this by right clicking on the corresponding folder and selecting deploy to Function App:



This step will deploy 4 functions to the Function App: **index**, **negotiate**, **notification**, and **processdata**.

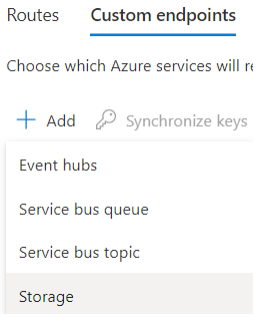




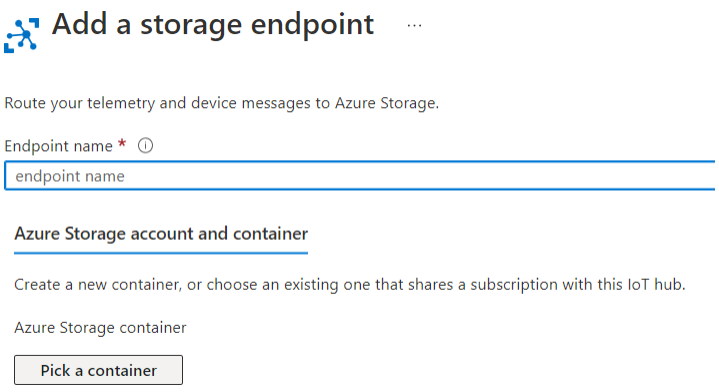
Update IoT Hub message routing endpoints

Now, let's go back to our IoT Hub instance and configure message routing so that messages can flow to the corresponding consumers.

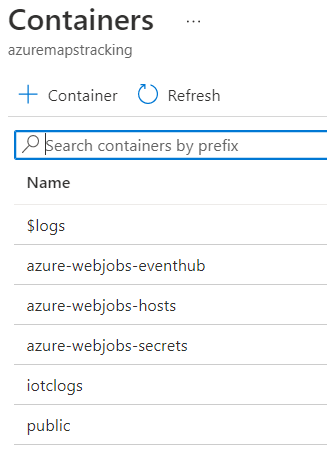
First, create a new endpoint called "iotclogs" that is going to be pointing to the blob container that we had created previously:



Pick a container by navigating to your storage account:

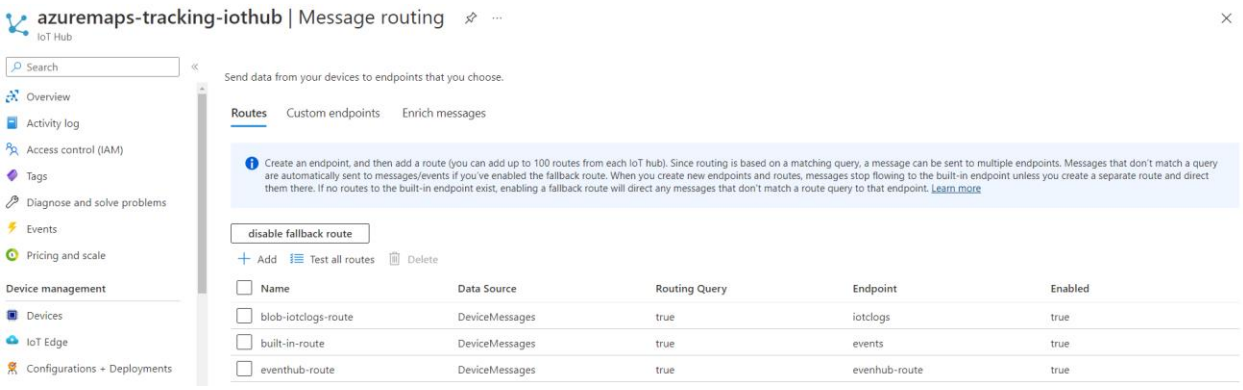


and selecting "iotclogs":



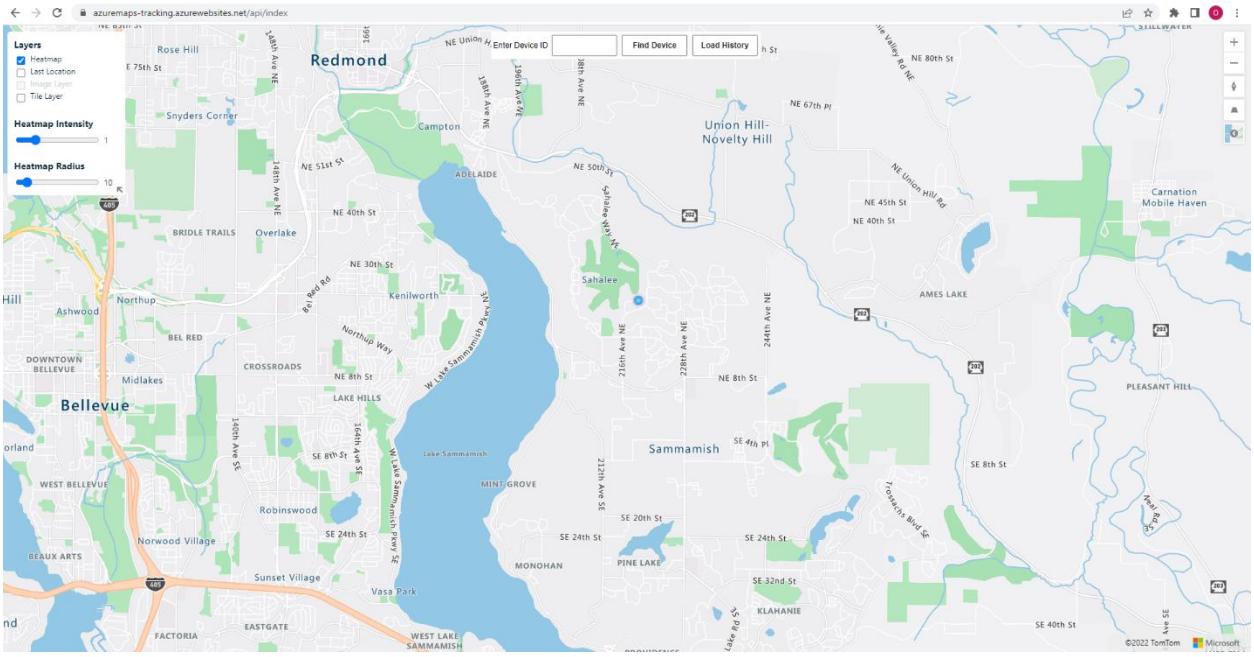
Repeat this process if you would like to create another route for pushing messages to an Event Hub (that we could then use to connect to Azure Data Explorer if so desired).

Make sure that there is a built-in route enabled for events, so that telemetry can keep flowing to the Function App listening to the built-in Event Hub.

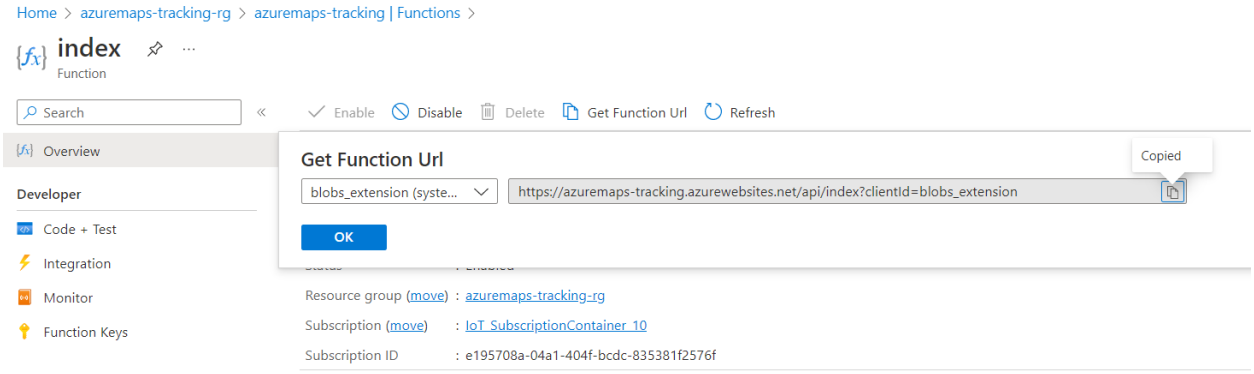


View data in your map

You should now be ready to start seeing data points in your map! Just make sure that the IoT Plug and Play application is open in your smartphone.



Just go to the URL that corresponds to the “index” function that you deployed to the Function App:



Congratulations! You have completed this tutorial 😊