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

Deployment Instructions

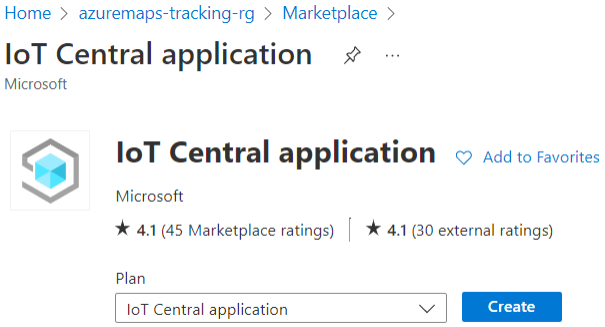
Clone repo

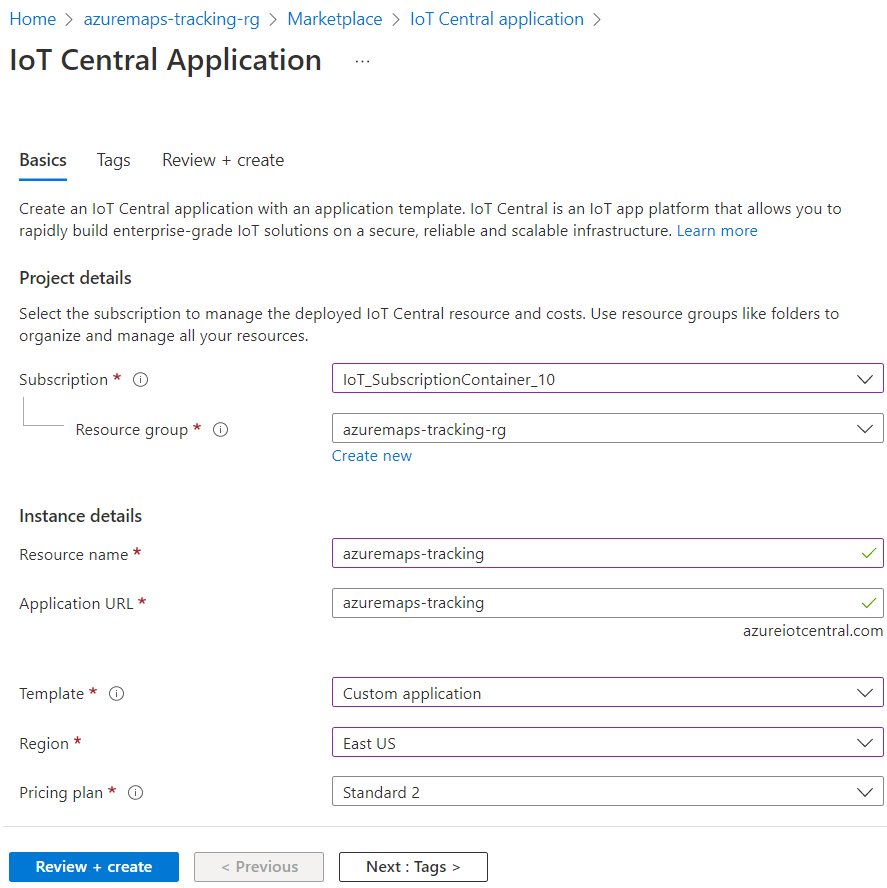
https://github.com/Azure-Samples/azuremaps-indoor-realtime-position-tracking

Create resource group in Azure Subscription

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

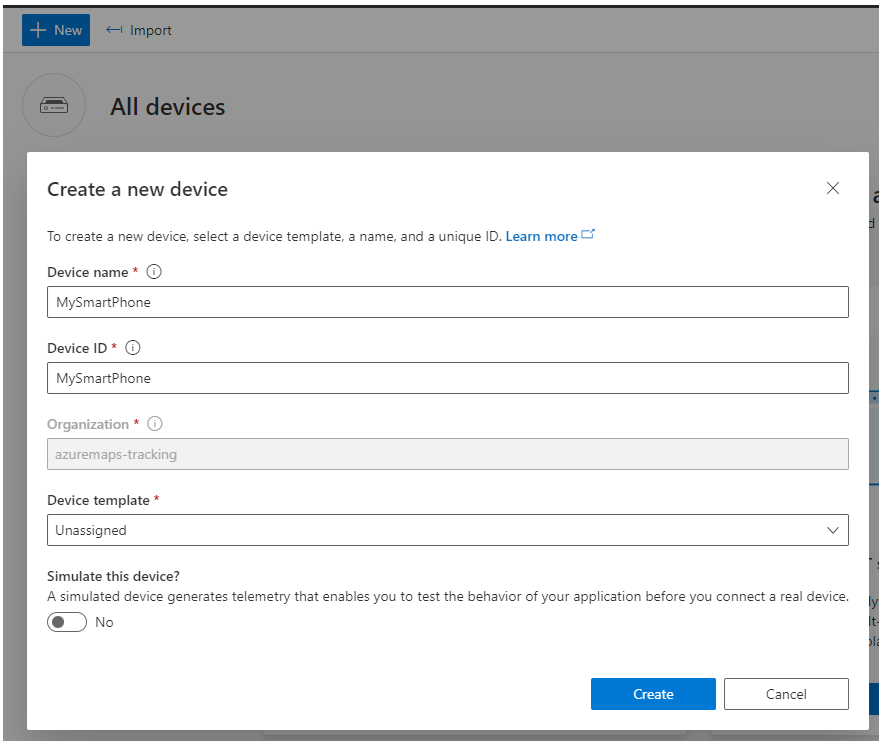
Create IoT Central Application





Add Device (e.g., smart phone)

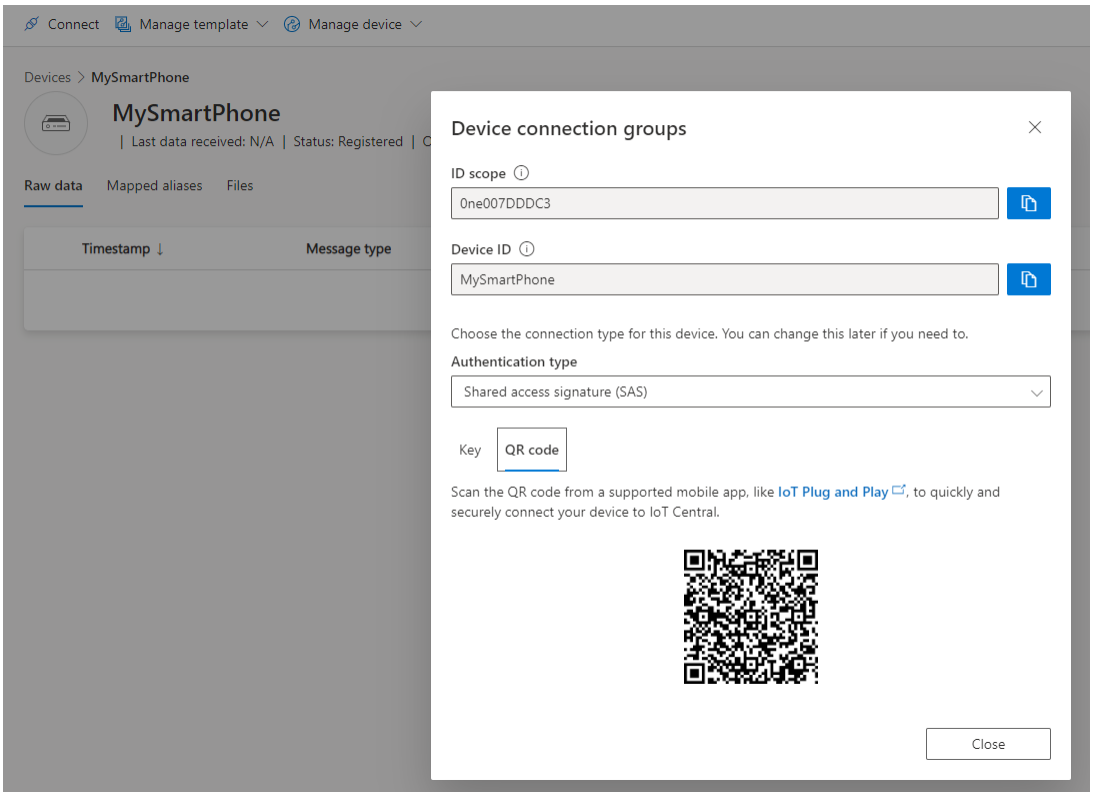
Go to the IoT Central application URL and add a new device.



Select the device that was just created.

Select “Connect”.

Select “QR code”.

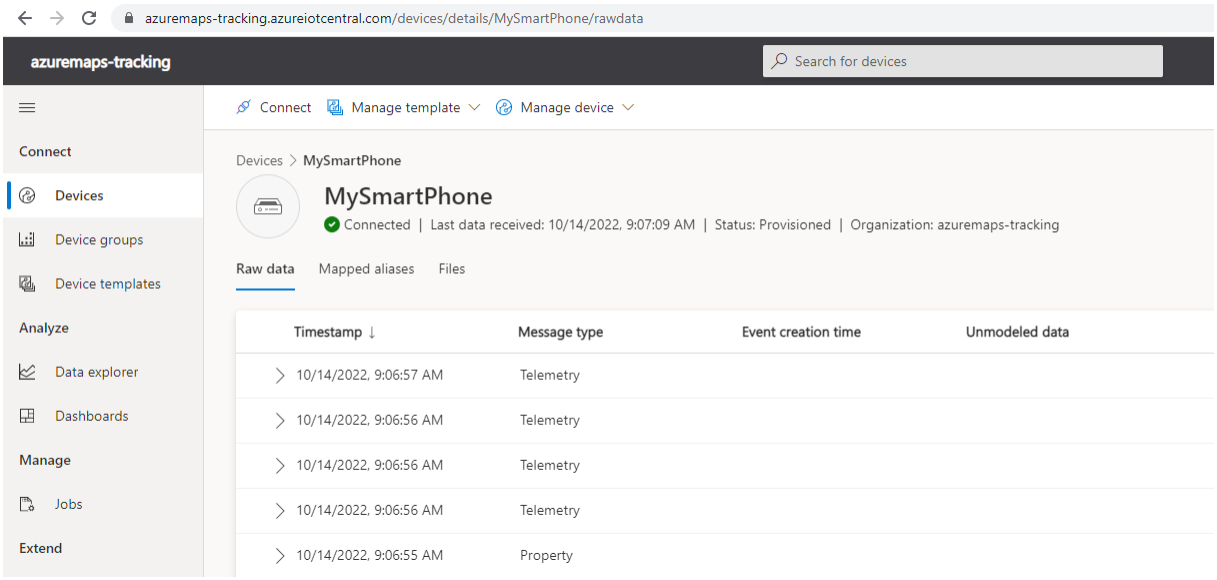


On your smartphone, install the “IoT Plug and Play” app.

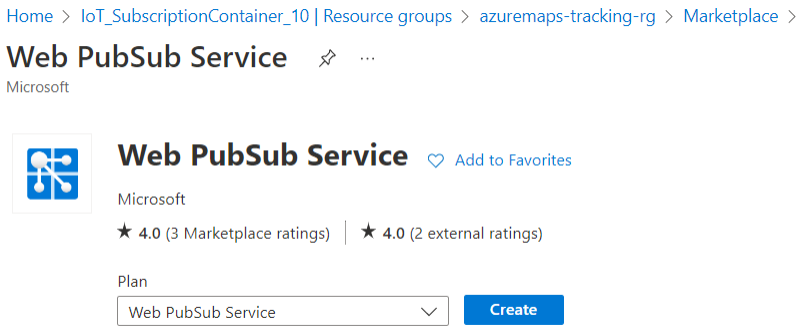
Go to Settings and Registration and select “Scan QR code”.

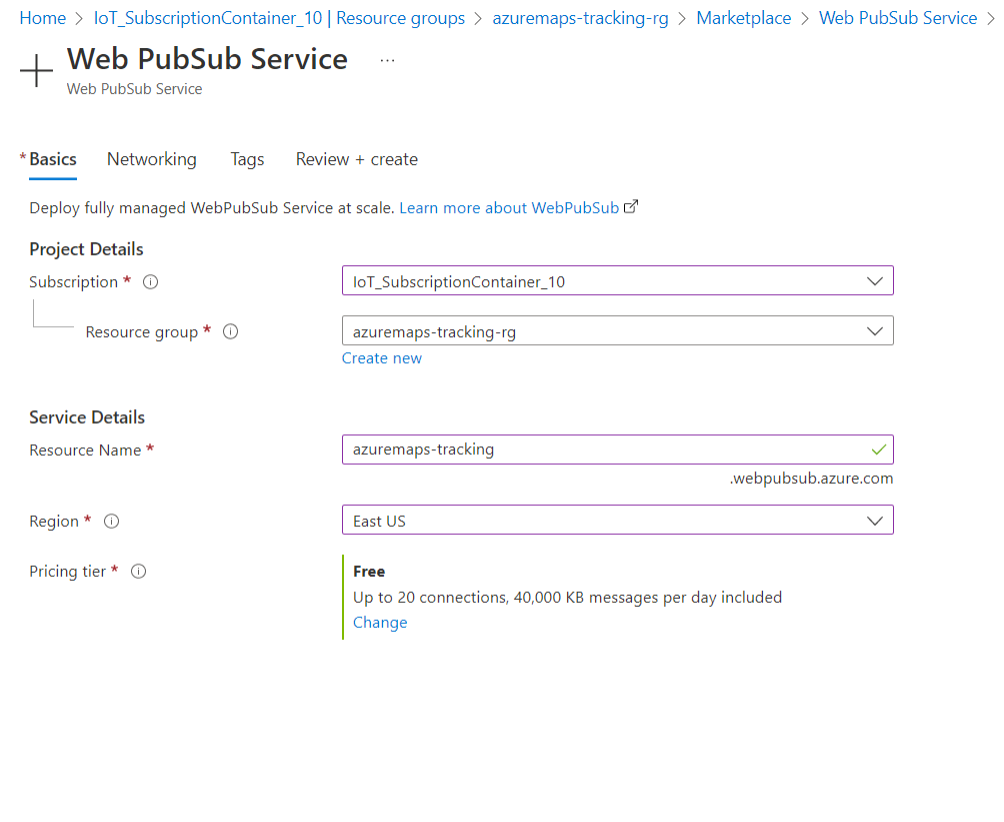
Then, scan the QR code from the IoT Central application so that your smart phone gets linked to this IoT Central application.

You should be seeing telemetry now in your IoT Central application.



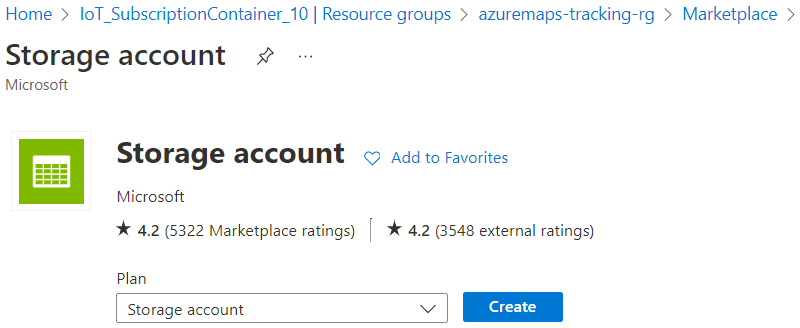
Create Azure Web PubSub

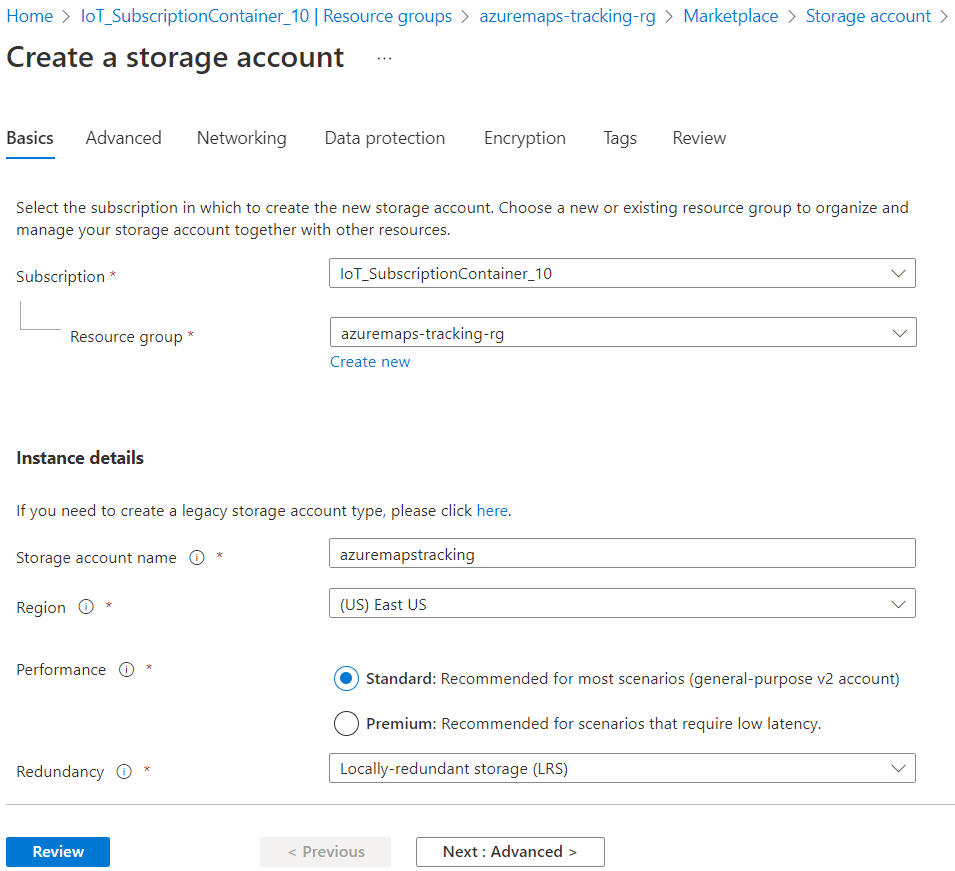




Under “Keys”, take a note of the connection string for this service.

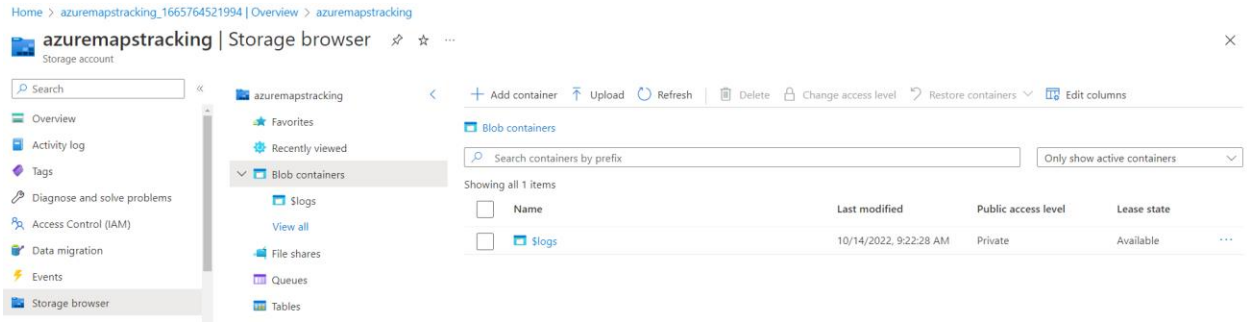
Create Storage Account



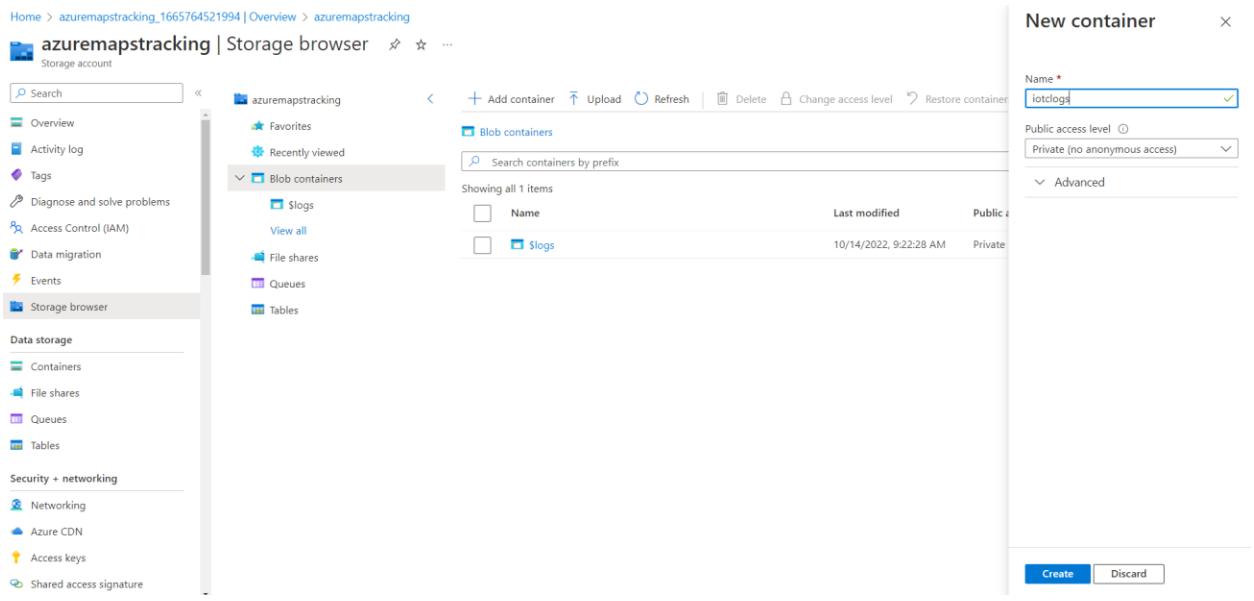


Save the value for the connection string under “Access keys” since you will need it later.

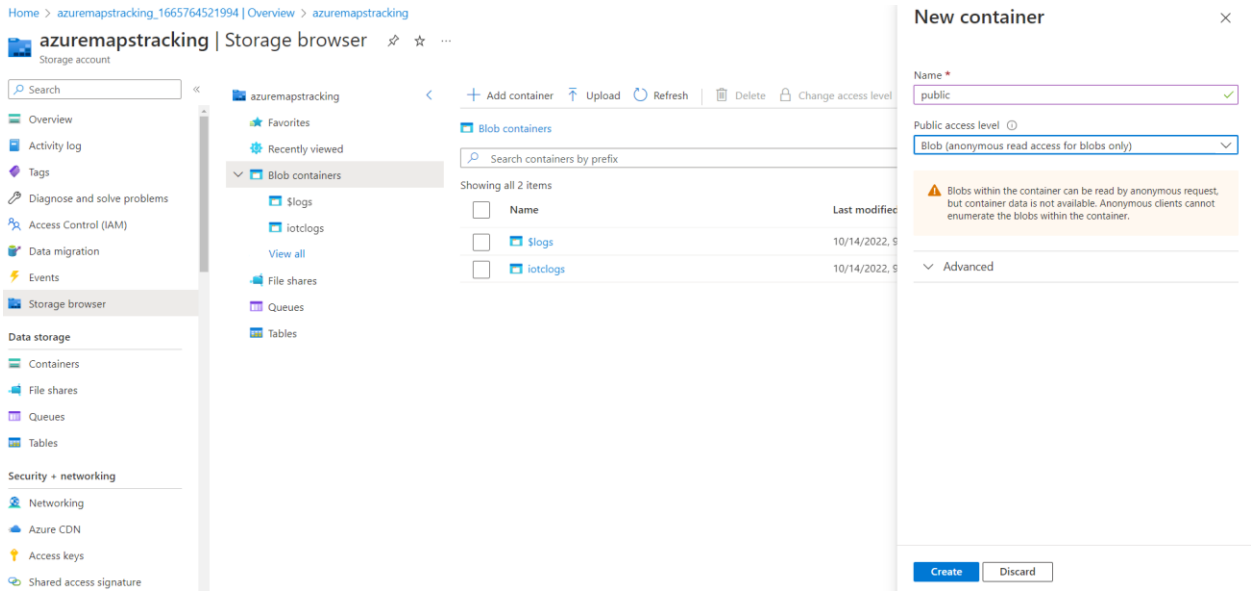
Under “Storage browser”, click on “Blob containers”:



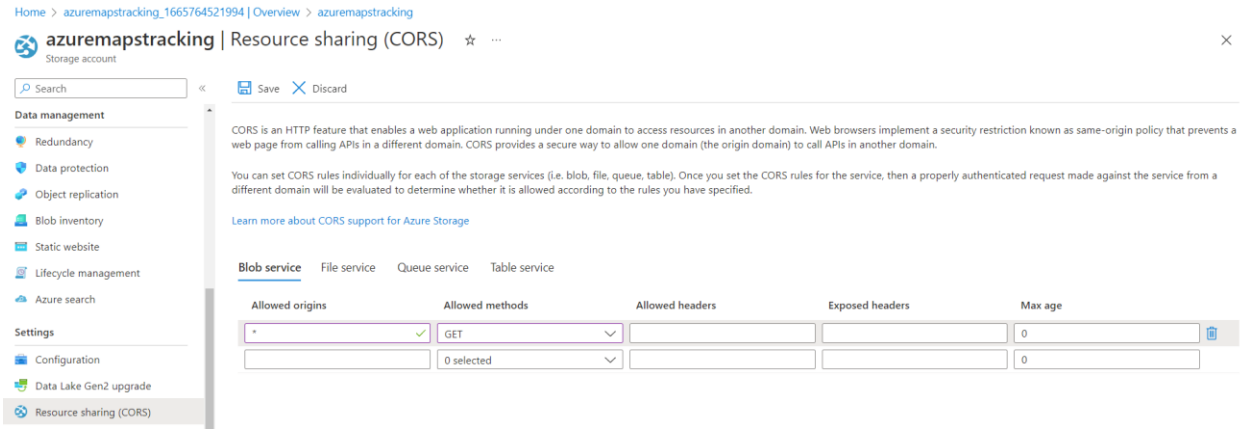
Create a new blob container called “iotclogs” (private):



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

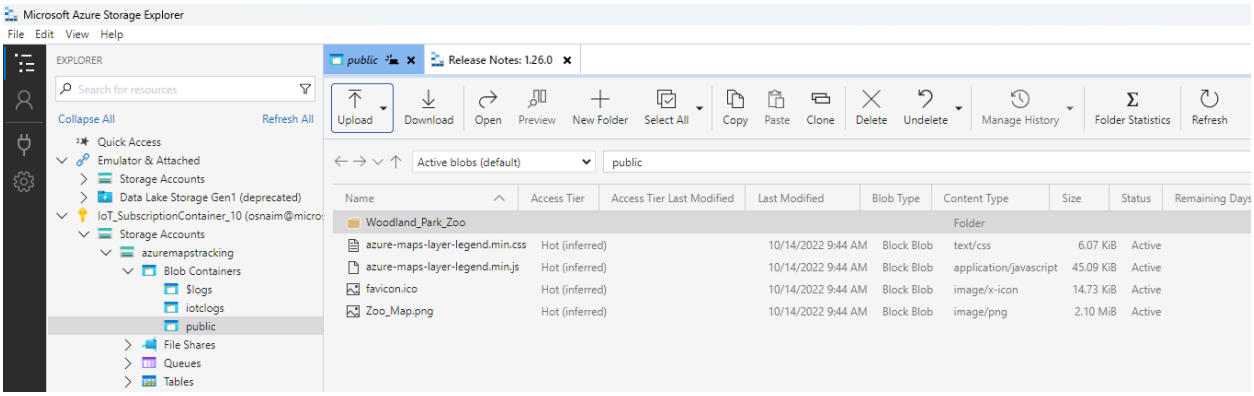


Enable CORS:

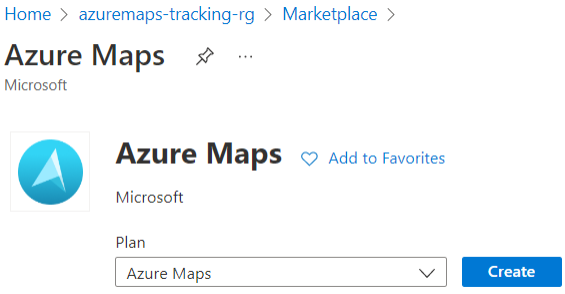


Install “Azure Storage Explorer”: https://azure.microsoft.com/en-us/features/storage-explorer/

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)

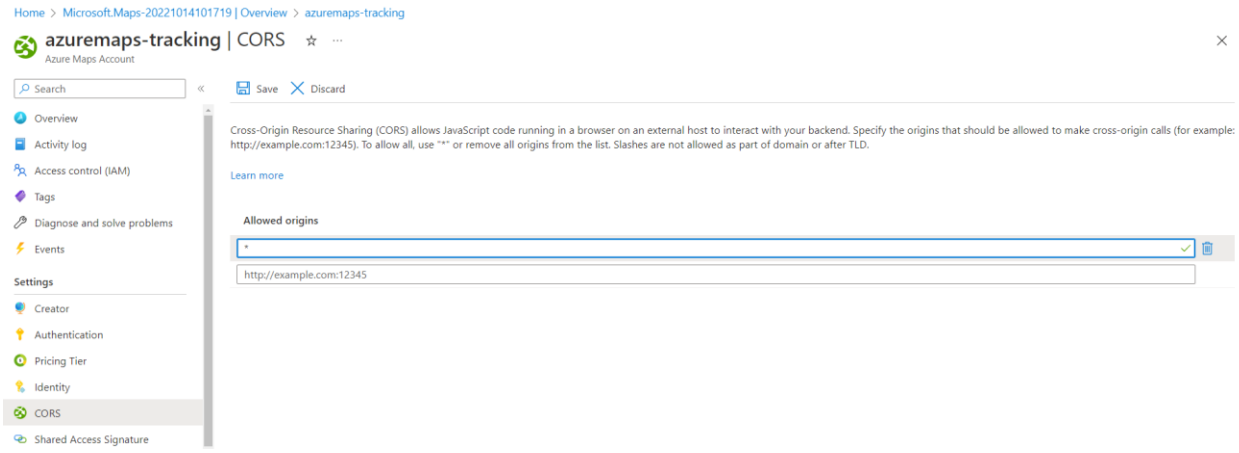
View full pricing details

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

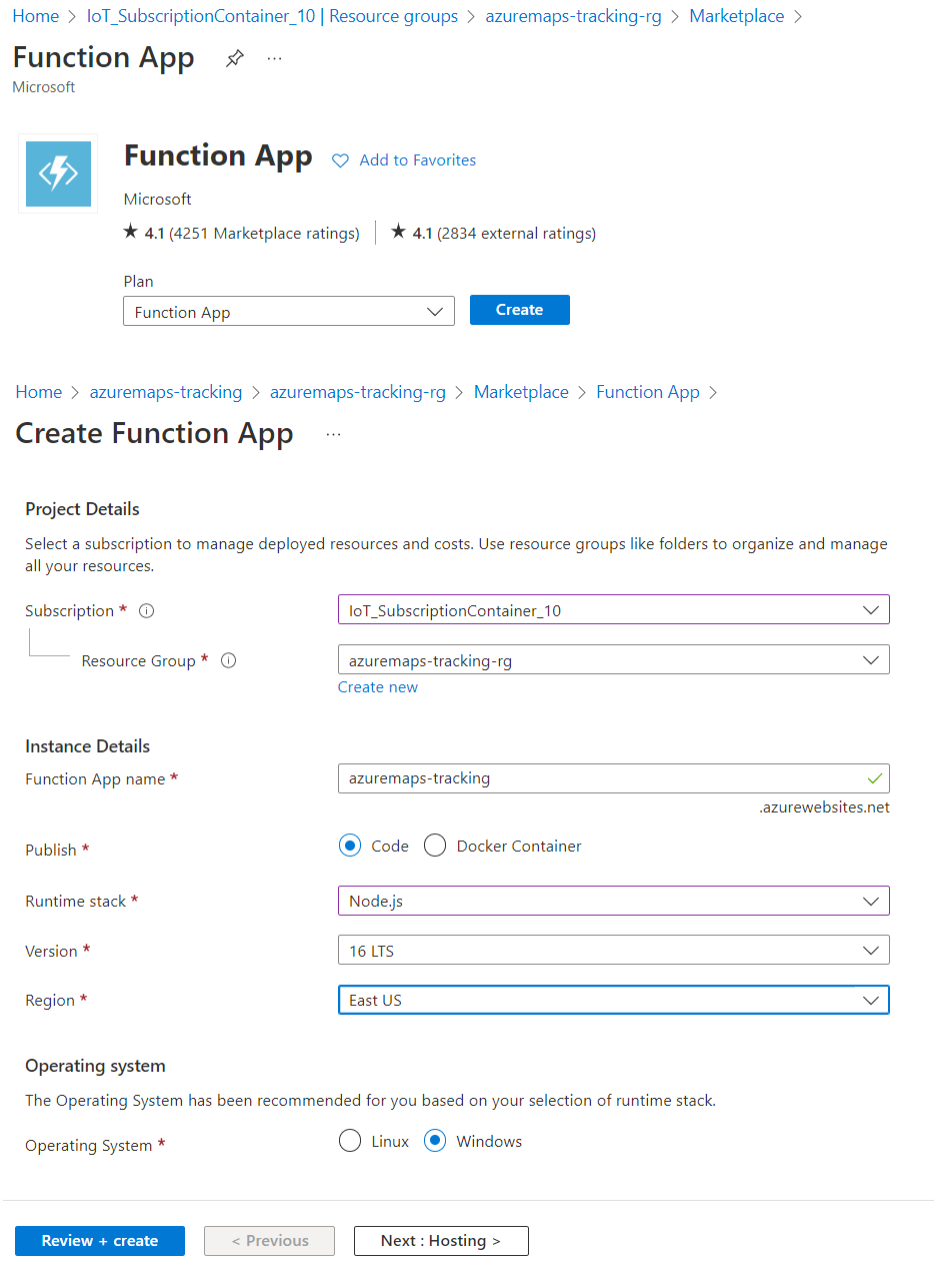
Review + create | < Previous | Next : Advanced >

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:

Home > azuremaps-tracking-rg > Marketplace > Function App >

Create Function App ...

Basics | Hosting | Networking | Monitoring | Deployment | Tags | Review + create

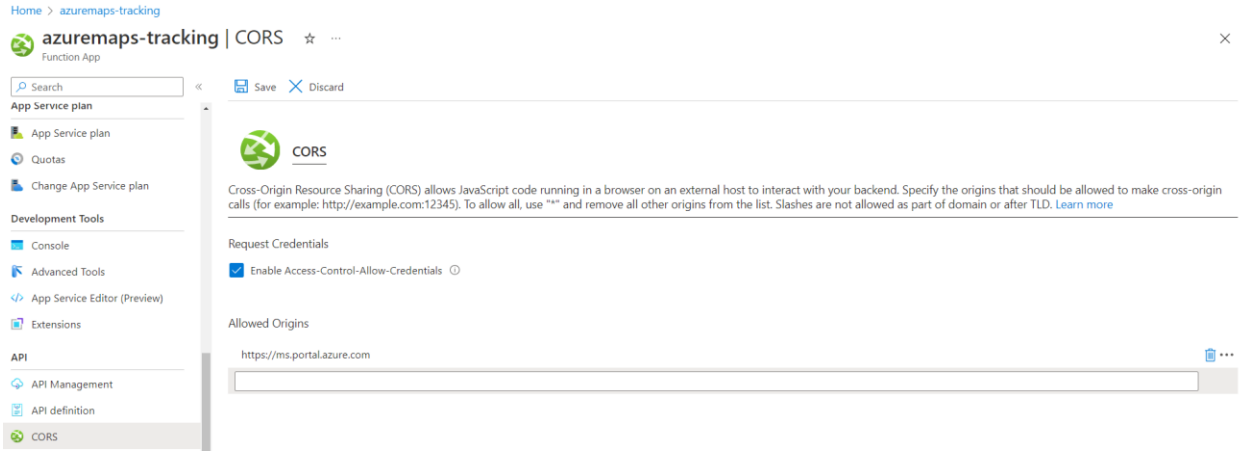
Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

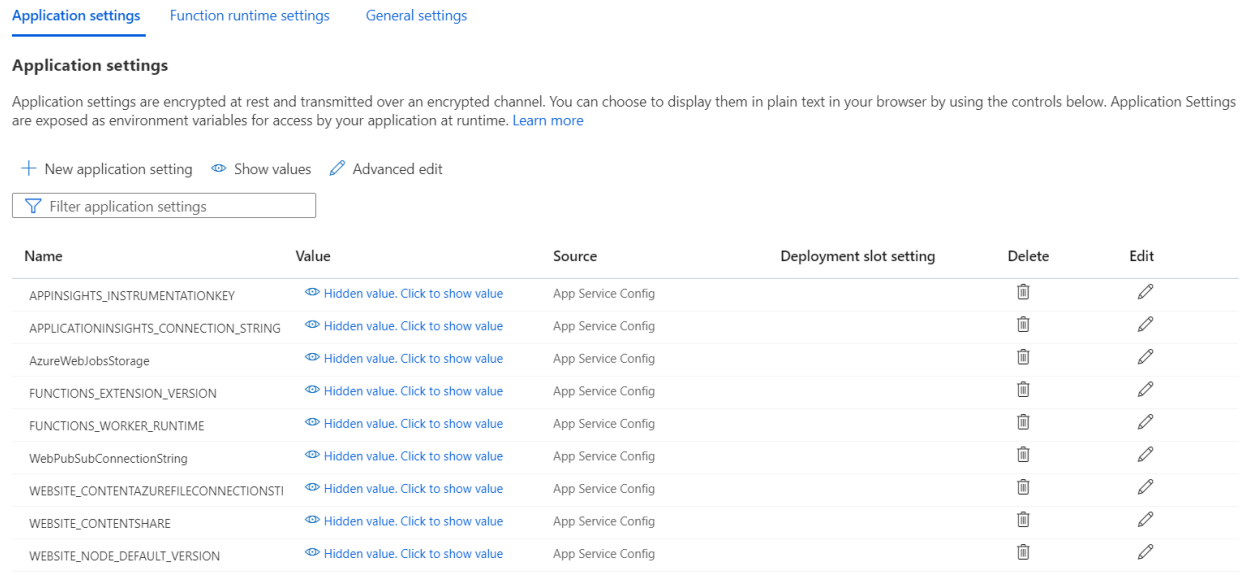
Storage account * azuremapstracking (v2)

Create new

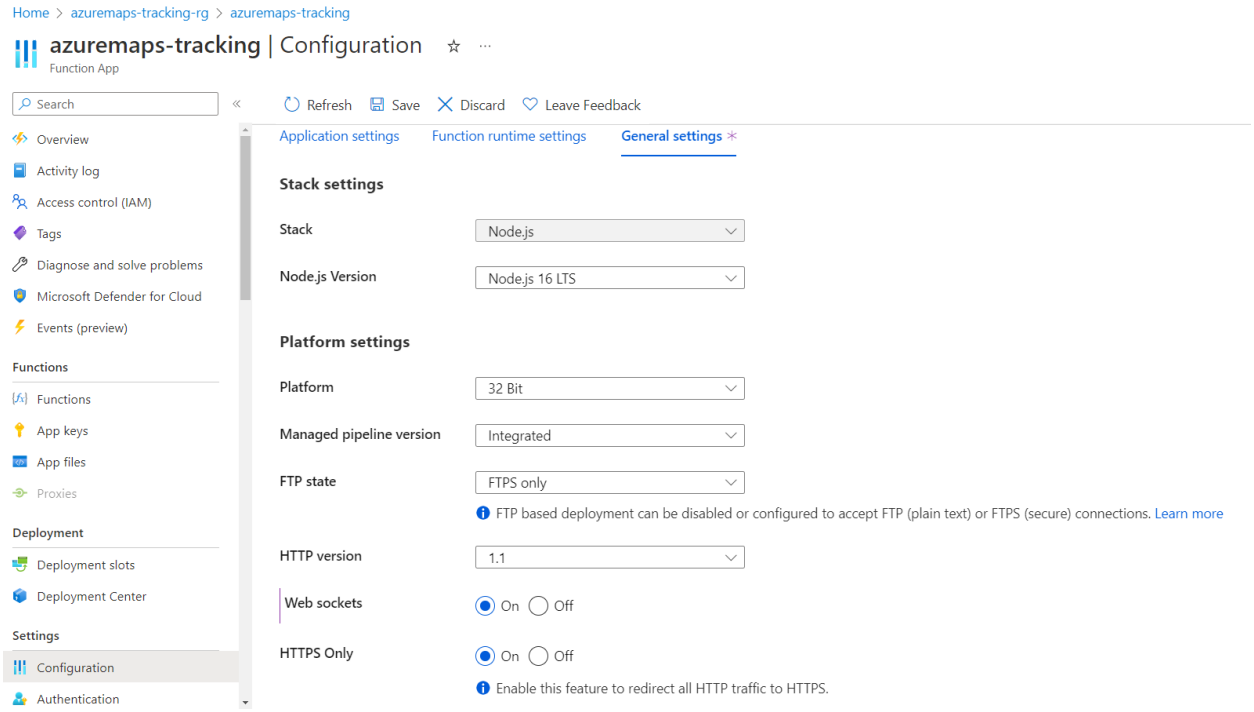
Once the Function App is created, enable CORS:



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



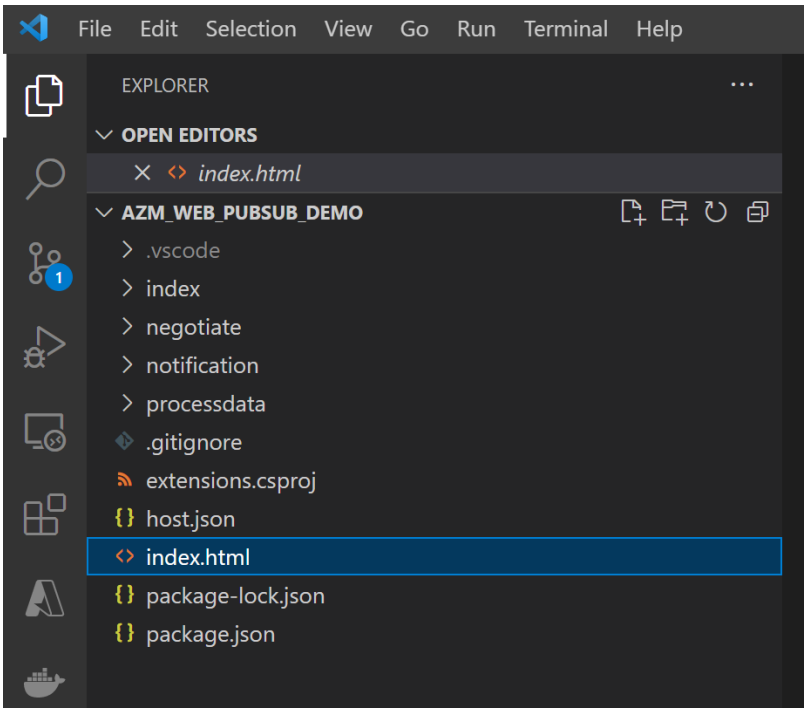
Also, under “Configuration”, enable Web sockets:



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-iotcentraldemo\AzM_Web_PubSub_Demo-v01):



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.

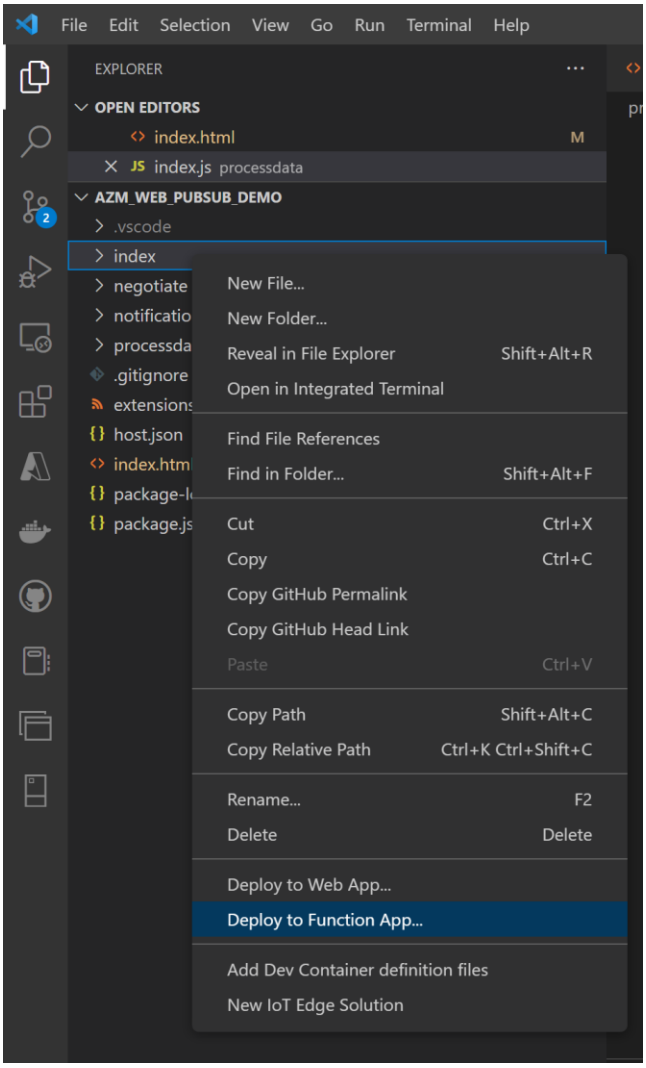
For example:

```
<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">
```

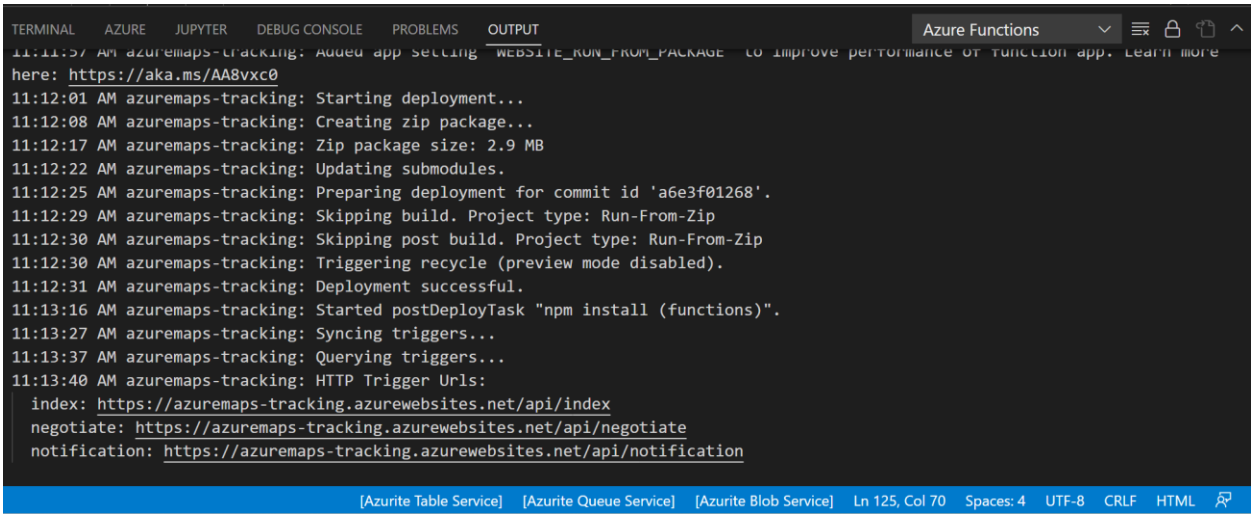
Then, replace “<YOUR-AZURE-MAPS-KEY>” with the map key you had saved previously.

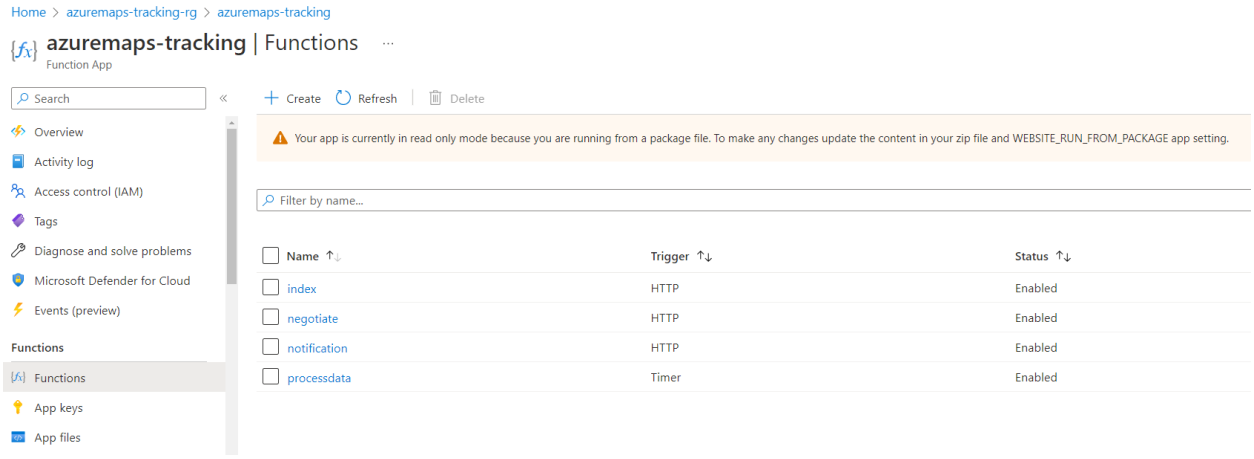
Deploy Function App to Azure

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.

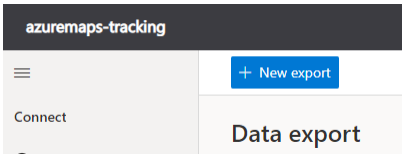


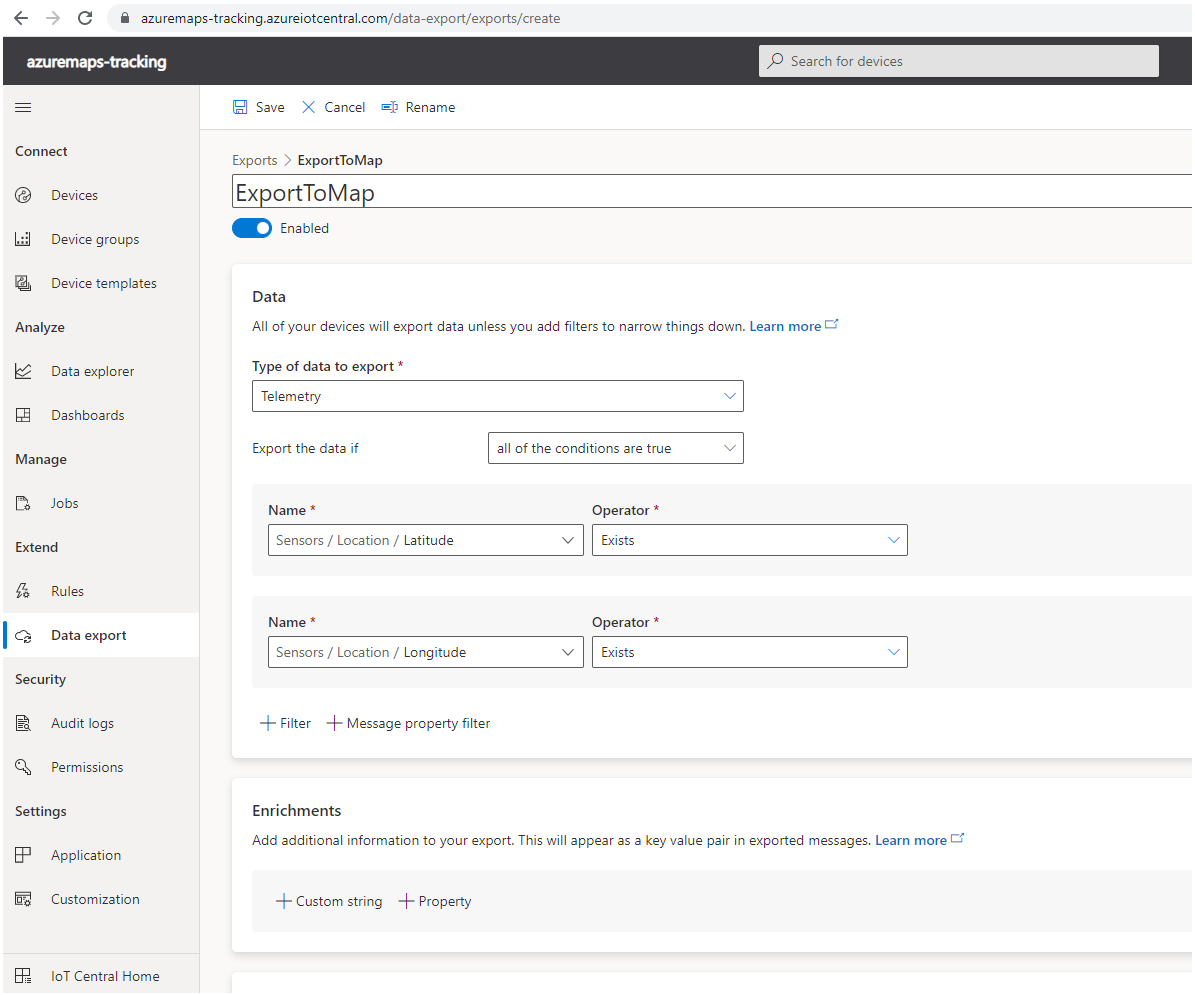


Export from IoT Central to Webhook

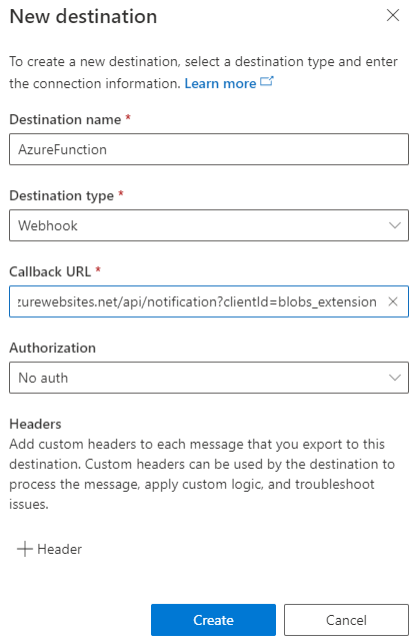
Create a new Data Export component using the following steps:

Click on “New export” and fill out the fields as follows:



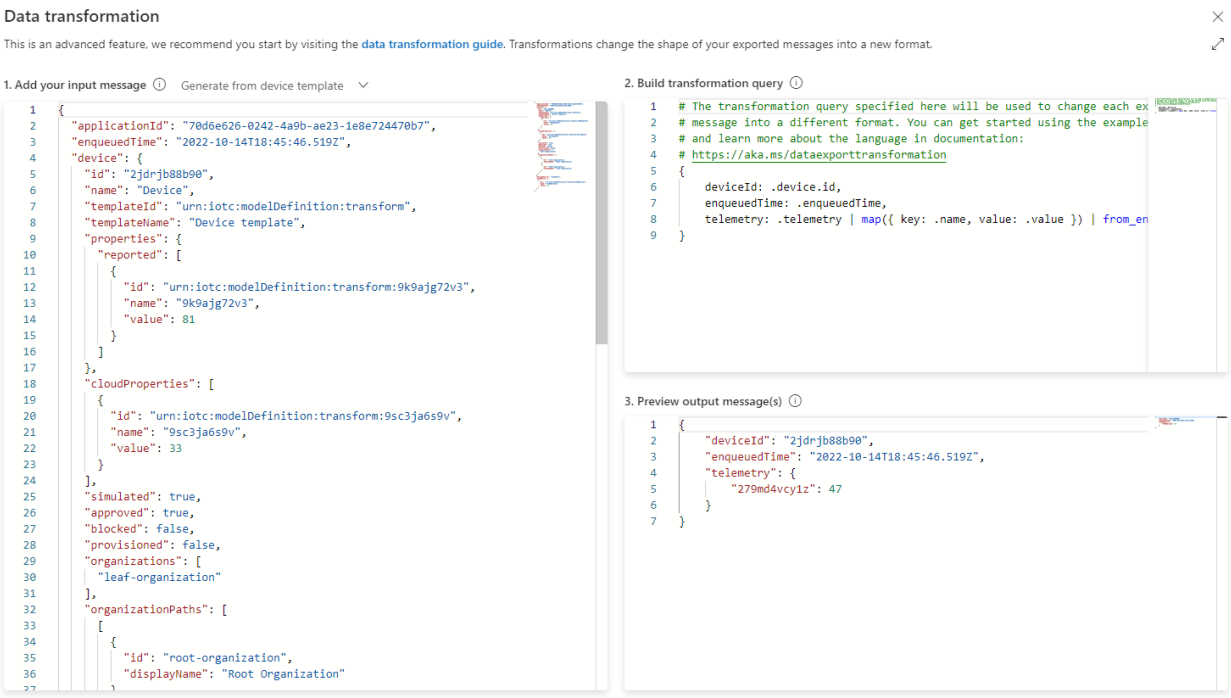


Add a new Destination:



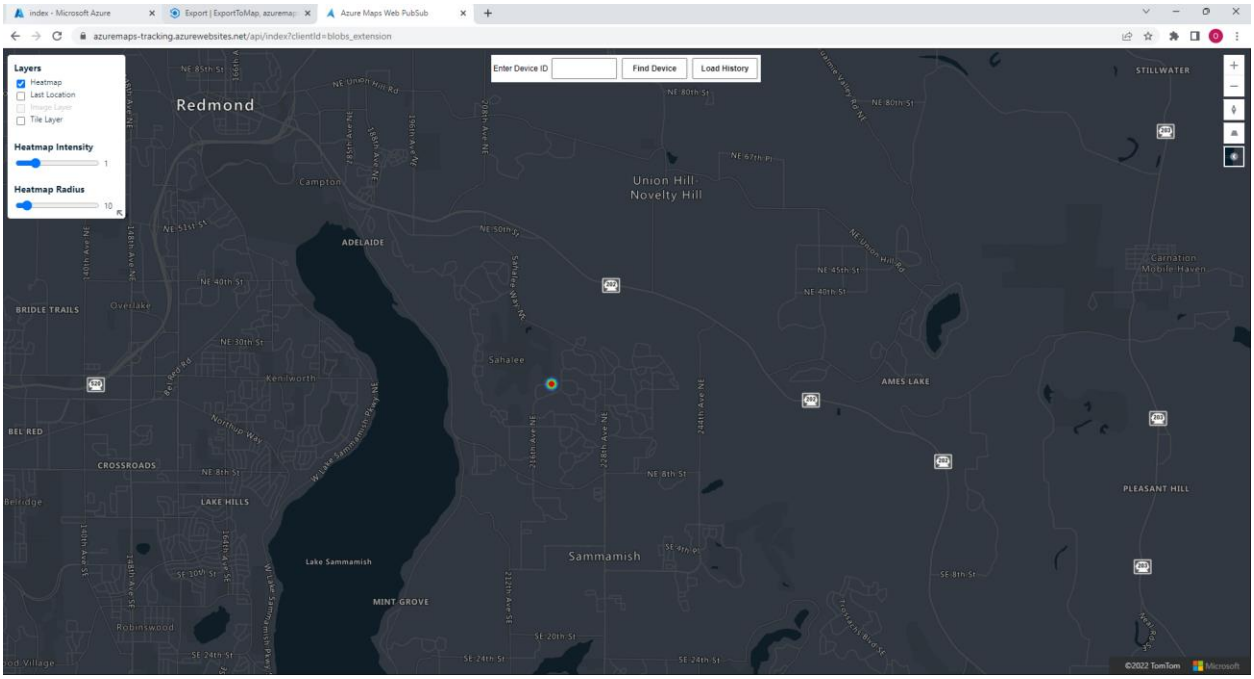
And finally add a Transform using the following transformation query:

```
{
    deviceId: .device.id,
    enqueuedTime: .enqueuedTime,
    telemetry: .telemetry | map({ key: .name, value: .value }) | from_entries,
}
```

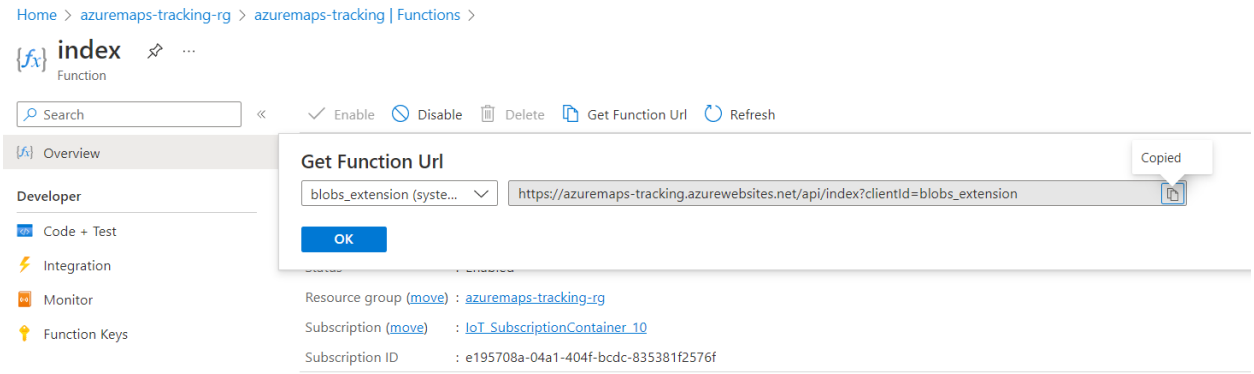


Now you can save this Data Export component.

Once this service is showing a “healthy” state, you should 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.



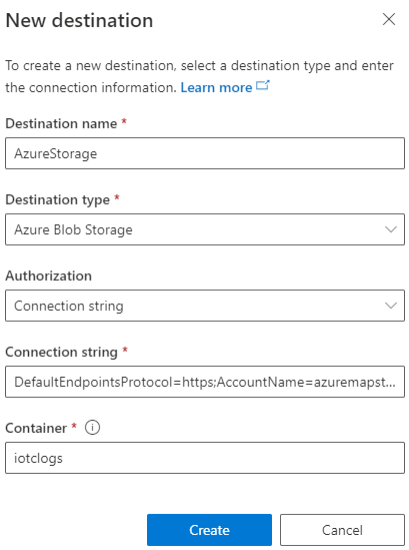
Just go to the URL that corresponds to the “index” function that you deployed to the Function App:



Export from IoT Central to Storage

Let’s create a new Data Export destination for processing the location history.

Using the same Data Export component that we created previously, click on create a new destination with the following details:



We'll use the connection string for the storage account that we saved previously. This will push location history data to iotclogs in blob storage.