Indoor real-time position tracking using LoRaWAN devices and Azure Maps

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

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

# Add Device (e.g., smart phone)

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

Graphical user interface, text, application, email

Description automatically generated

Select the device that was just created.

Select “Connect”.

Select “QR code”.

A picture containing graphical user interface

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

# Create Azure Web PubSub

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

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

# Create Storage Account

Graphical user interface, text, application, email

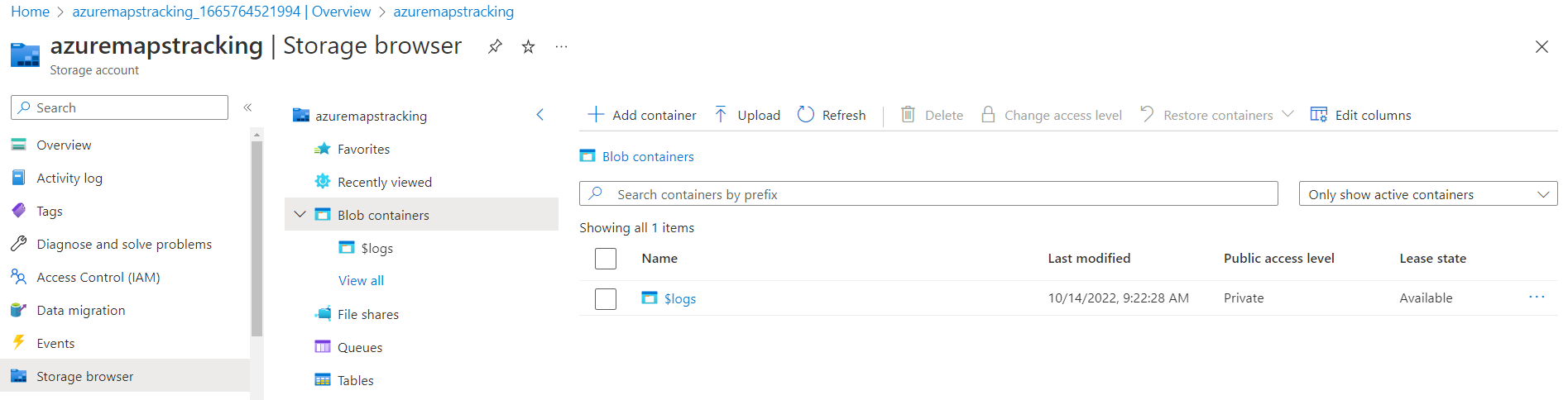
Description automatically generated

Graphical user interface, text, email

Description automatically generated

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):

Graphical user interface, application

Description automatically generated

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

Graphical user interface, application

Description automatically generated

Enable CORS:

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

# Create Azure Maps instance

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Enable CORS:

Graphical user interface, text, application, email

Description automatically generated

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

# Create Azure Function

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

Once the Function App is created, enable CORS:

Graphical user interface, text, application, email

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

Also, under “Configuration”, enable Web sockets:

Graphical user interface, application

Description automatically generated

# 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):

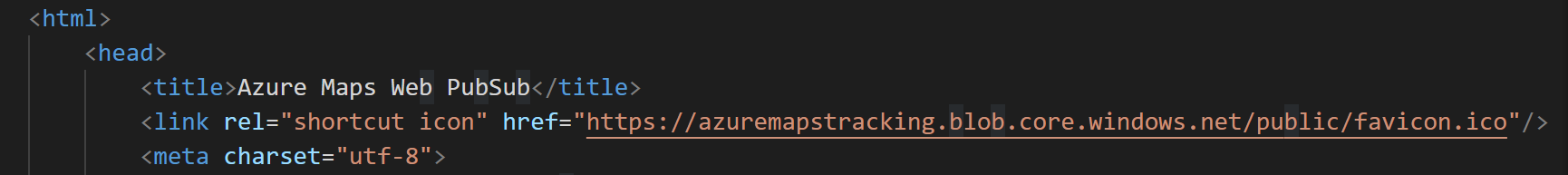
A screenshot of a computer

Description automatically generated with medium confidence

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:



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:

A screenshot of a computer

Description automatically generated with medium confidence

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

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# 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:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Add a new Destination:

Graphical user interface, text, application, email

Description automatically generated

And finally add a Transform using the following transformation query:

{

    deviceId: .device.id,

    enqueuedTime: .enqueuedTime,

    telemetry: .telemetry | map({ key: .name, value: .value }) | from\_entries,

}

Graphical user interface, text, application

Description automatically generated

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.

Map

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

# 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:

Graphical user interface, text, application, email

Description automatically generated

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.