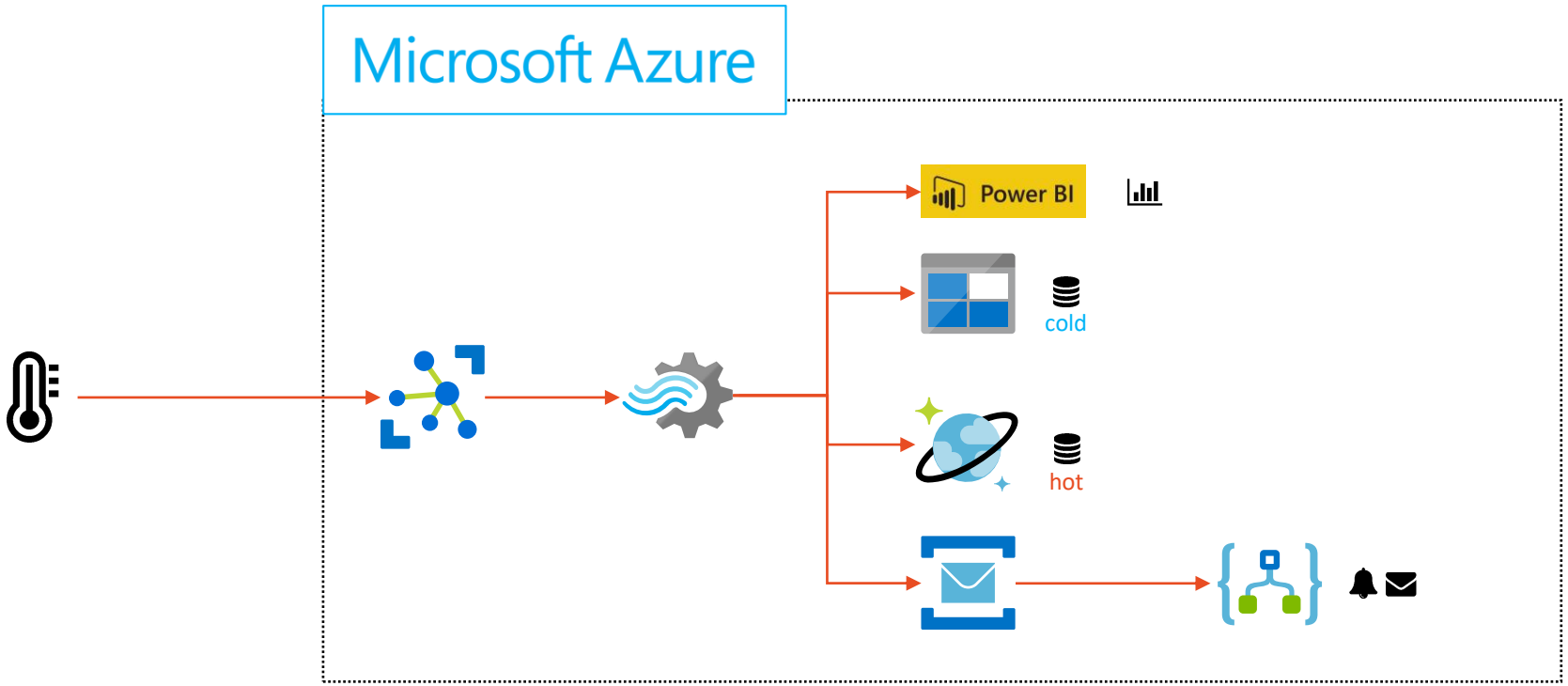
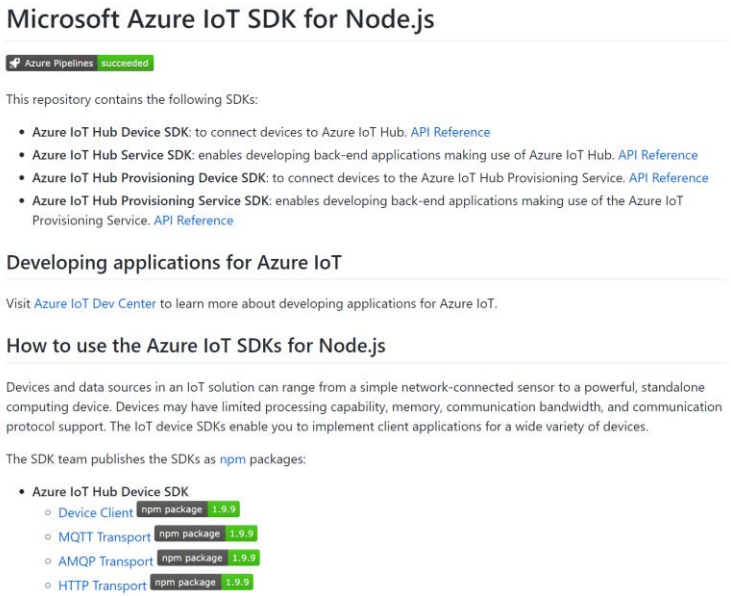

Microsoft
Azure


Azure

Telemetry Stream Analytics



Build and deploy solution



Microsoft Azure IoT SDK for Node.js

Azure Pipelines succeeded

This repository contains the following SDKs:

- **Azure IoT Hub Device SDK**: to connect devices to Azure IoT Hub. API Reference
- **Azure IoT Hub Service SDK**: enables developing back-end applications making use of Azure IoT Hub. API Reference
- **Azure IoT Hub Provisioning Device SDK**: to connect devices to the Azure IoT Hub Provisioning Service. API Reference
- **Azure IoT Hub Provisioning Service SDK**: enables developing back-end applications making use of the Azure IoT Provisioning Service. API Reference

Developing applications for Azure IoT

Visit Azure IoT Dev Center to learn more about developing applications for Azure IoT.

How to use the Azure IoT SDKs for Node.js

Devices and data sources in an IoT solution can range from a simple network-connected sensor to a powerful, standalone computing device. Devices may have limited processing capability, memory, communication bandwidth, and communication protocol support. The IoT device SDKs enable you to implement client applications for a wide variety of devices.

The SDK team publishes the SDKs as npm packages:

- **Azure IoT Hub Device SDK**
  - Device Client npm package 1.9.9
  - MQTT Transport npm package 1.9.9
  - AMQP Transport npm package 1.9.9
  - HTTP Transport npm package 1.9.9

azure-iot-sdk-node

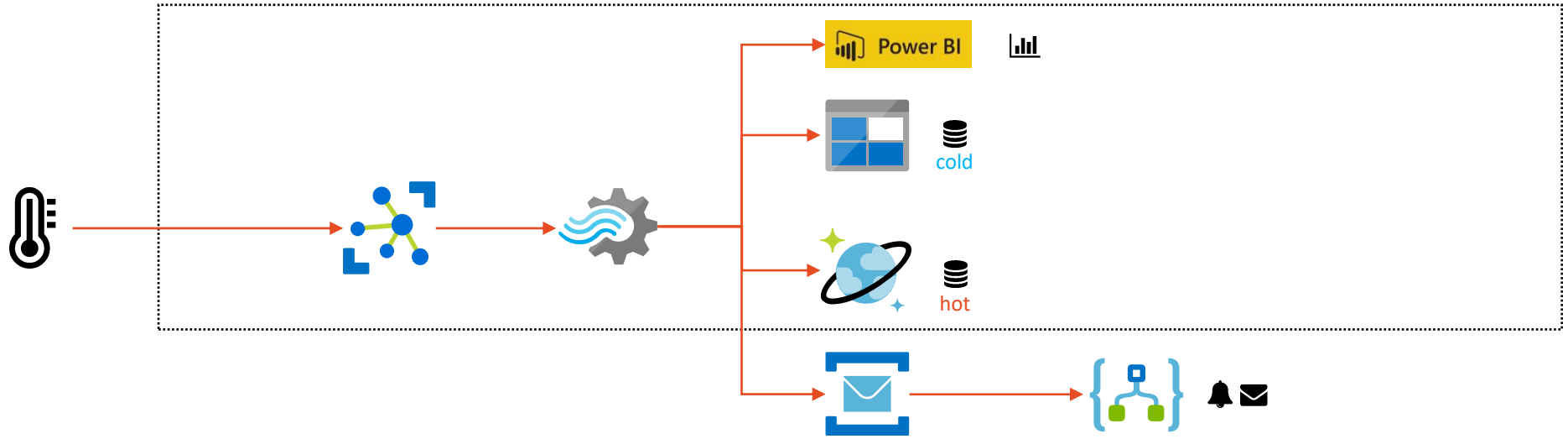
For the puporse of the article, we will simulate a device through the Microsoft Azure device SDK for Node.js. Our example will simulate a simple telemetry device, with the following structure:

```
{
        "id": "spaceneedle"
        ,temperature: 20 + (Math.random() * 10)
        ,localisation: {
                type: "Point"
                , coordinates: [2.3488, 48.8534]
        }
        ,Owner: {
                CompanyName: "Miscrosoft"
                ,id_companyName: "4309718283412"
        }
}
```

Create a Power BI account

https://signup.microsoft.com/signup?Sku=a403ebcc-fae0-4ca2-8c8c-7a907fd6c235&ru=https%3A%2F%25app.powerbi.com&pbi_source=ASA

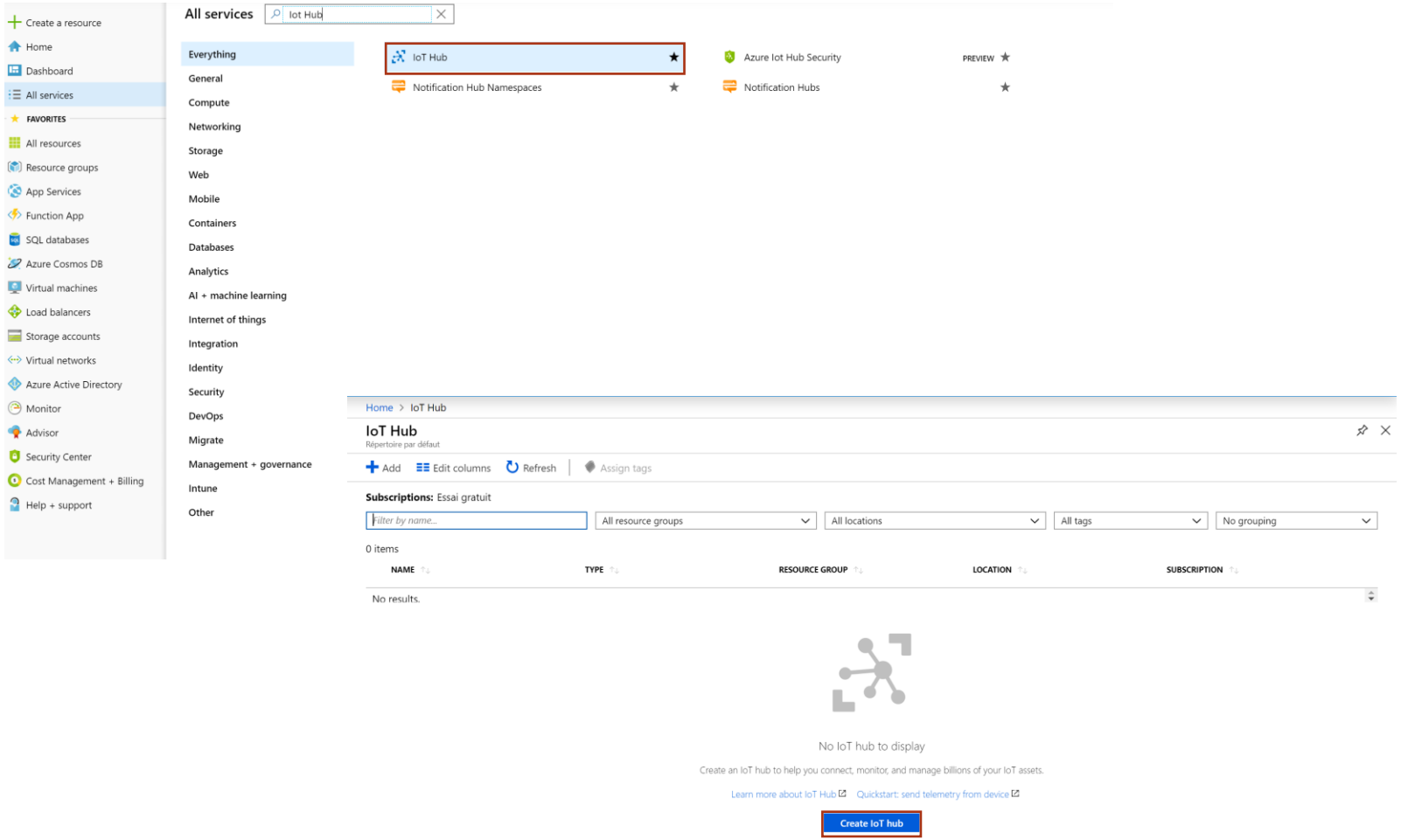
real-time reporting



Azure IoT Hub creation

step 1

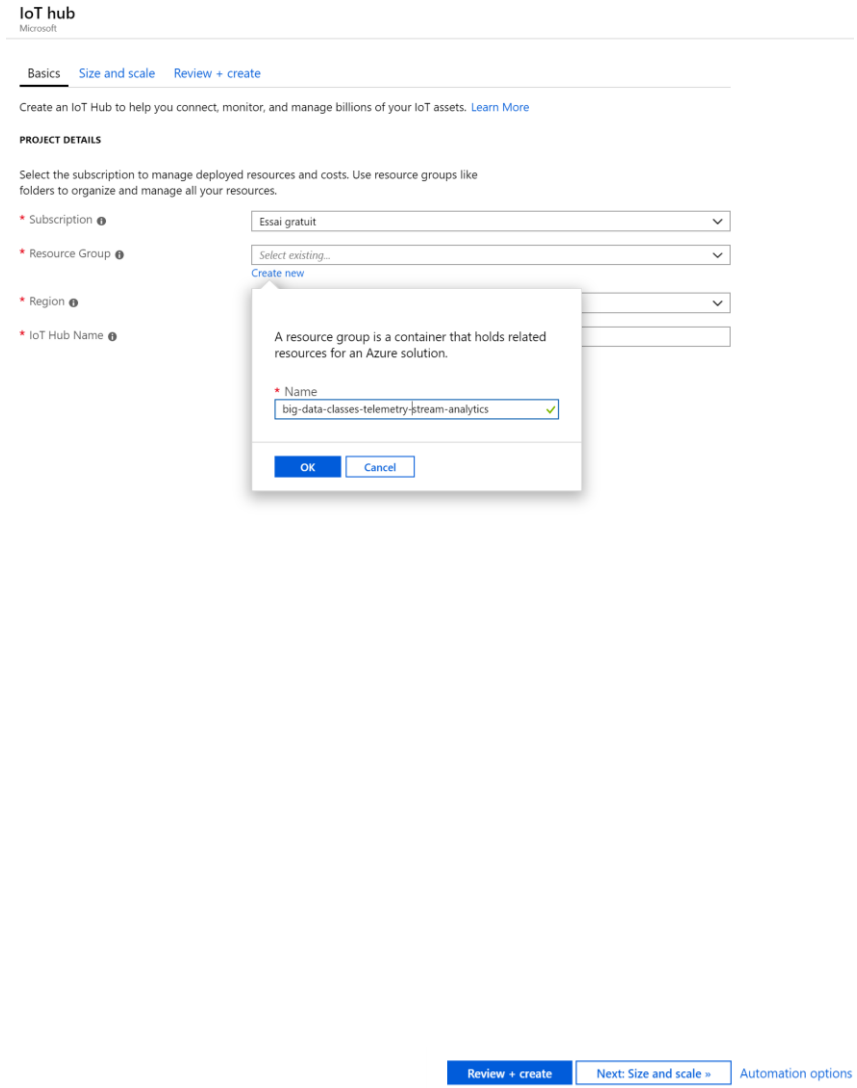
- Access to Azure portal : https://portal.azure.com/
- Launch the service “Iot hub”



Azure IoT Hub creation

step 1

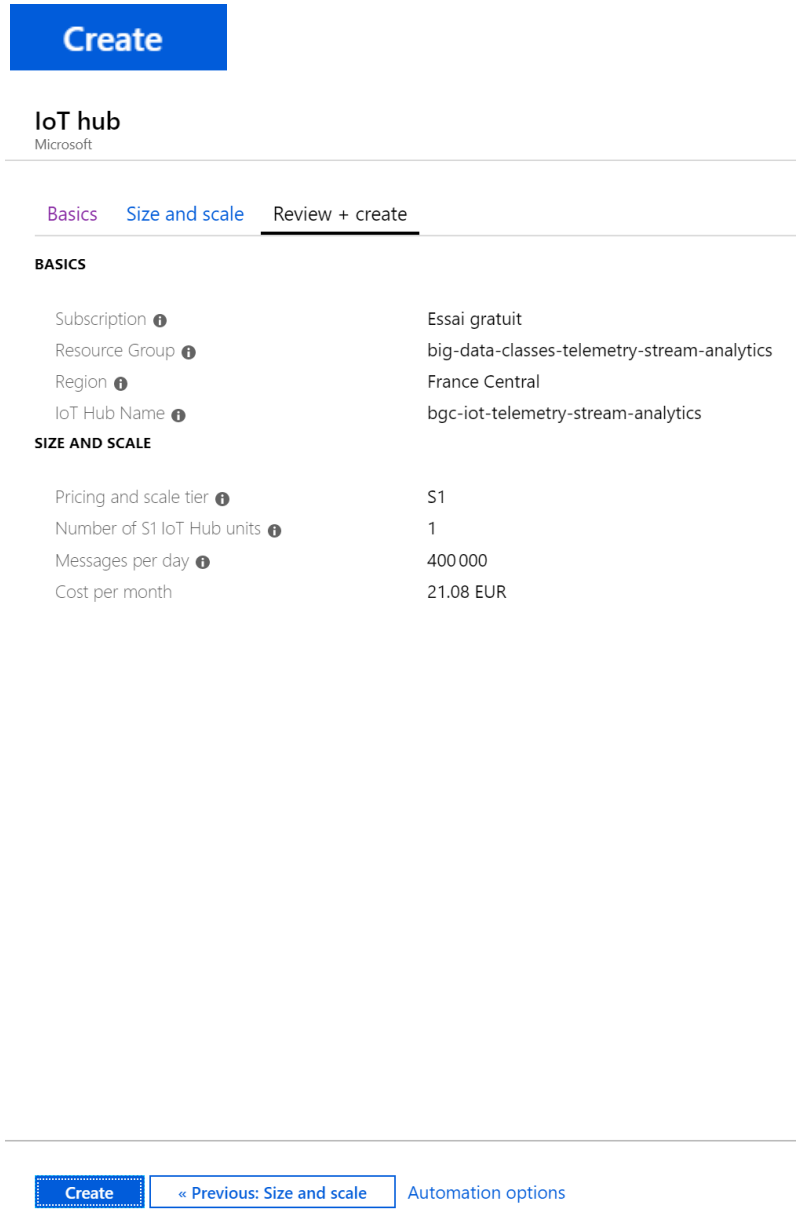
- Create a new ressource group, which will be using for all the exercice



- Review + create

Azure IoT Hub creation

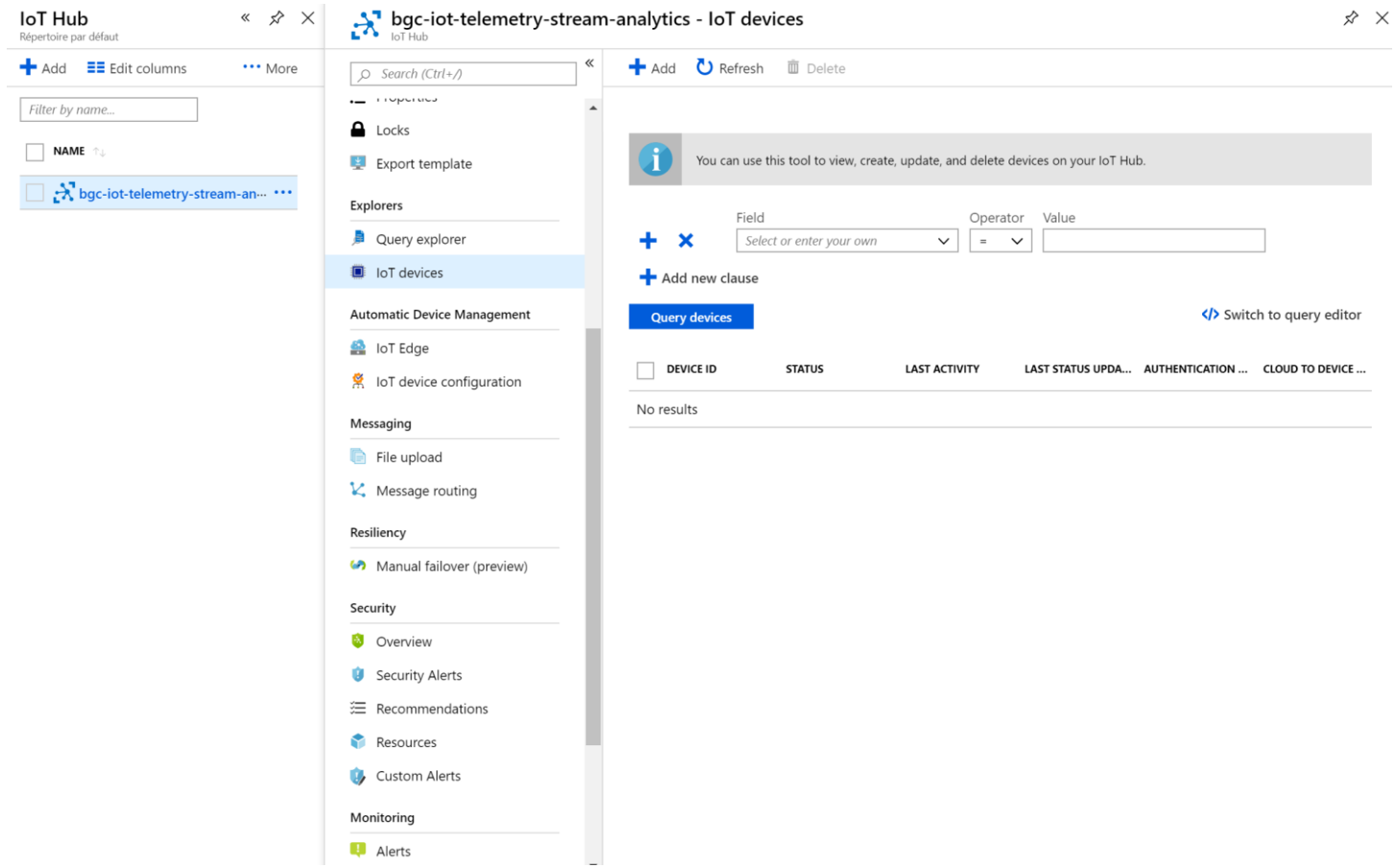
step 1



Azure IoT Hub device creation

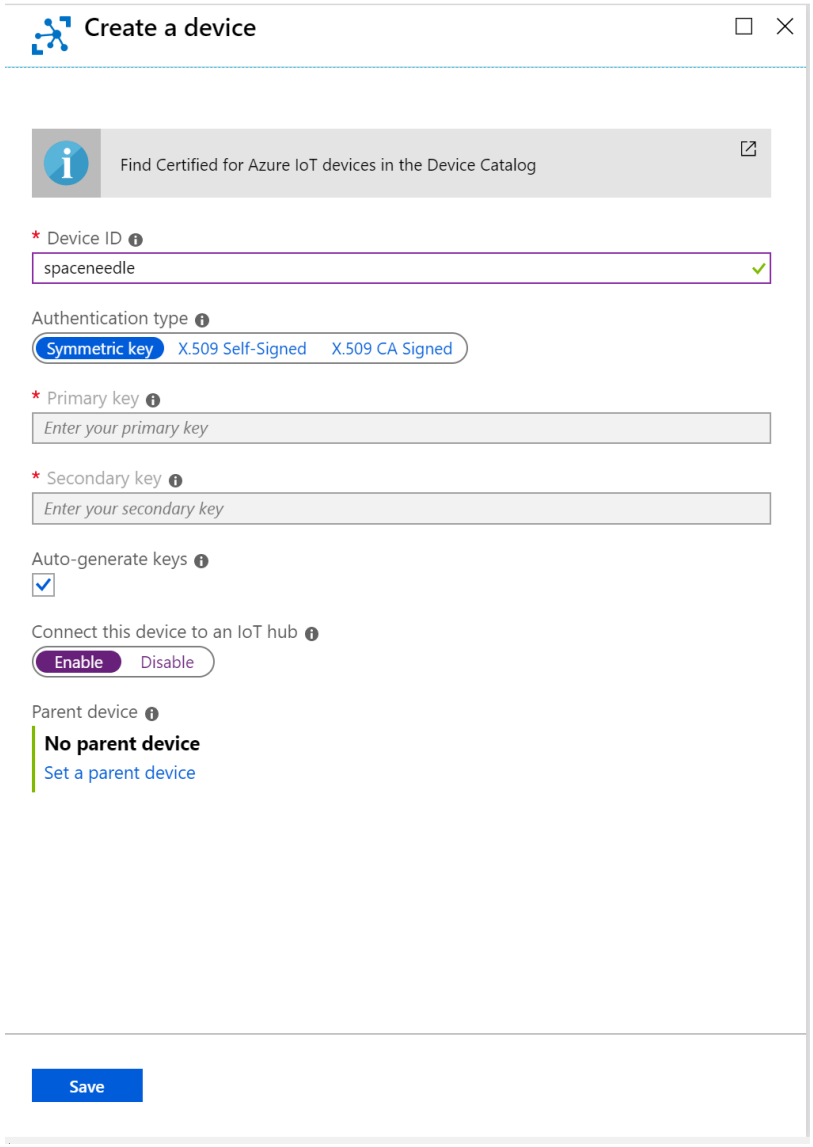
step 2

- Create a new device



Azure IoT Hub device creation

step 2



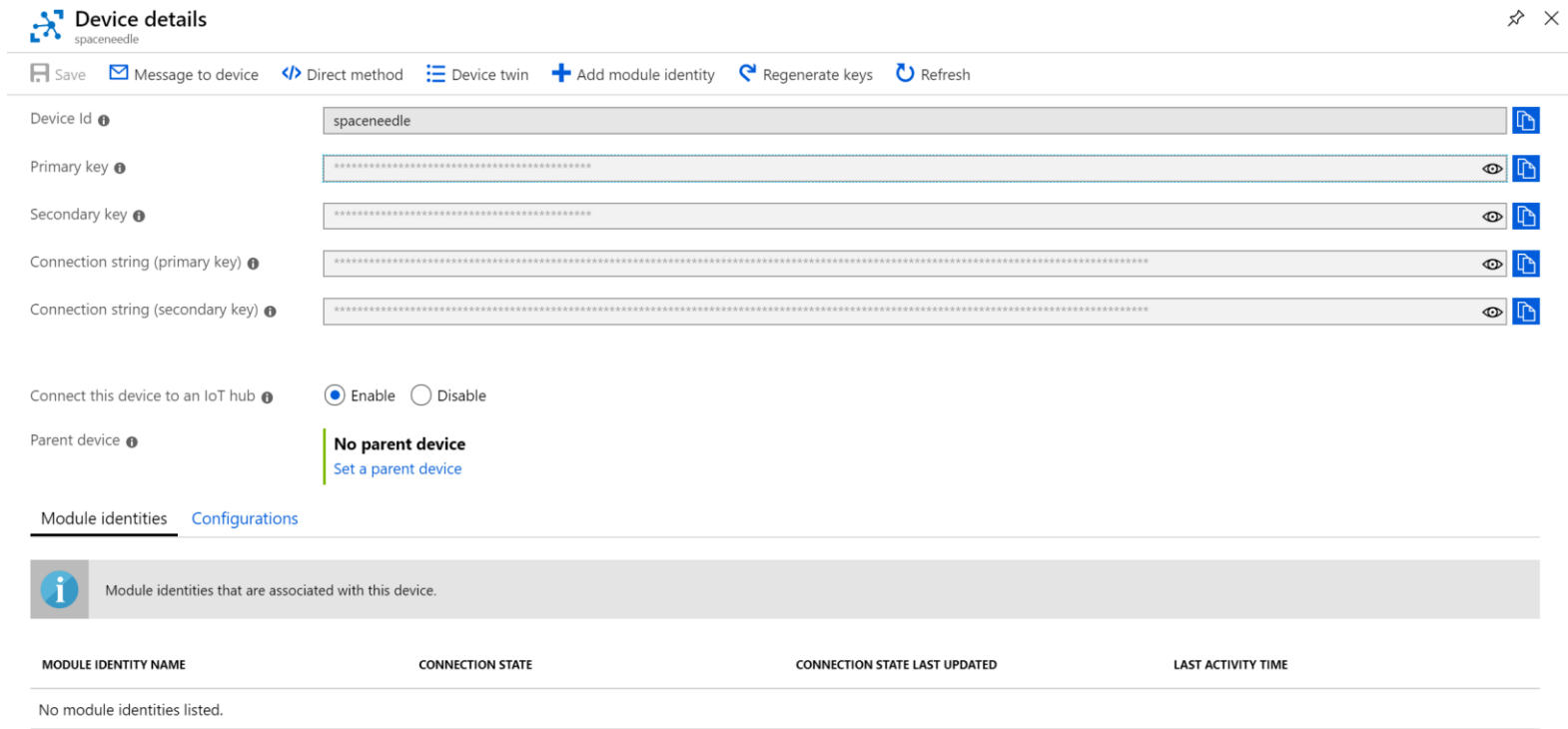
In «IoT Devices » pane:

- ID: enter a name keep it in mind for the other steps
- Authentication type: Symmetric Key
- Auto Generate keys: checked
- Connect device to IoT Hub : enable

Azure IoT Hub configuration

step 3

In your new created device, copy the “Connection string (primary key)”



Link solution to Azure IoT Hub

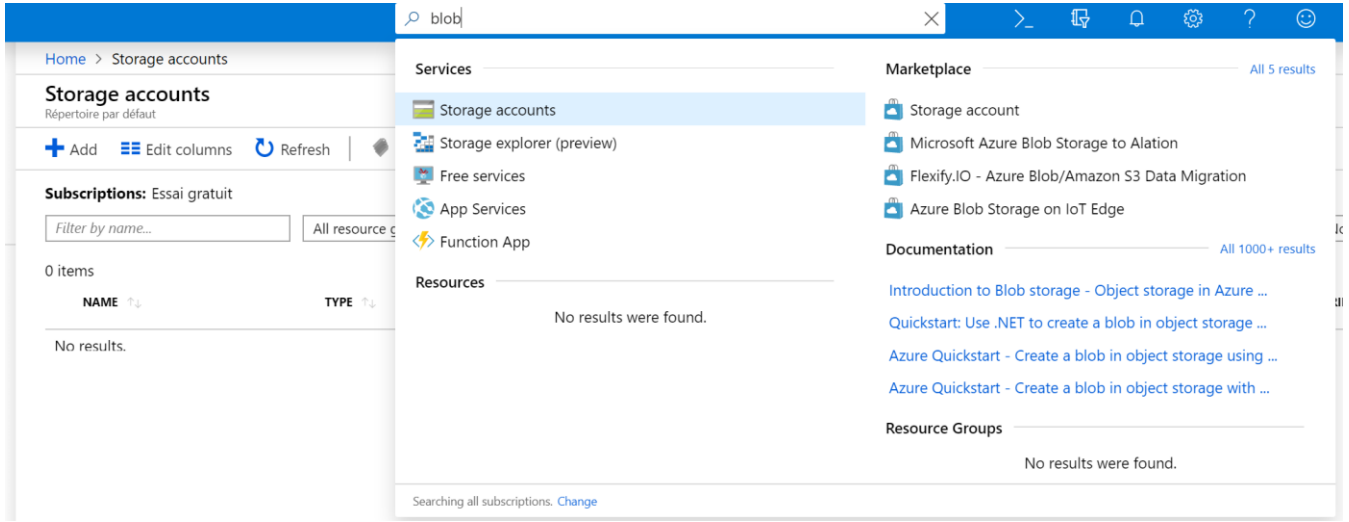
step 4

In your sdk code, paste the “Connection string (primary key)”

```
// Copyright (c) Microsoft. All rights reserved.
// Licensed under the MIT license. See LICENSE file in the project root for full license information.

'use strict';

var Protocol = require('azure-iot-device-mqtt').Mqtt;
// Uncomment one of these transports and then change it in fromConnectionString to test other transports
// var Protocol = require('azure-iot-device-amqp').AmqpWs;
// var Protocol = require('azure-iot-device-http').Http;
// var Protocol = require('azure-iot-device-amqp').Amqp;
// var Protocol = require('azure-iot-device-mqtt').MqttWs;
var Client = require('azure-iot-device').Client;
var Message = require('azure-iot-device').Message;

// String containing Hostname, Device Id & Device Key in the following formats:
//  "HostName=<iothub_host_name>;DeviceId=<device_id>;SharedAccessKey=<device_key>"
//var connectionString = "HostName=sqlbits-iothub-changefeed.azure-devices.net;DeviceId=devicetwo;SharedAccessKey=tYQr3V3l4C3fZdNU
var connectionString = 'HostName=bgc-telemetry-stream-analytics.azure-devices.net;DeviceId=spaceneedle;SharedAccessKey=+4/eHWHdbOW

// fromConnectionString must specify a transport constructor, coming from any transport package.
var client = Client.fromConnectionString(connectionString, Protocol);

var connectCallback = function (err) {
  if (err) {
    console.error('Could not connect: ' + err.message);
  } else {
    console.log('Client connected');
    client.on('message', function (msg) {
      console.log('Id: ' + msg.messageId + ' Body: ' + msg.data);
      // When using MQTT the following line is a no-op.
      client.complete(msg, printResultFor('completed'));
      // The AMQP and HTTP transports also have the notion of completing, rejecting or abandoning the message.
      // When completing a message, the service that sent the C2D message is notified that the message has been processed.
      // When rejecting a message, the service that sent the C2D message is notified that the message won't be processed by the de
      // When abandoning the message, IoT Hub will immediately try to resend it. The method to use is client.abandon(msg, callback
      // MQTT is simpler: it accepts the message by default, and doesn't support rejecting or abandoning a message.
    });

    // Create a message and send it to the IoT Hub every second
```

Blob Storage Creation

step 5

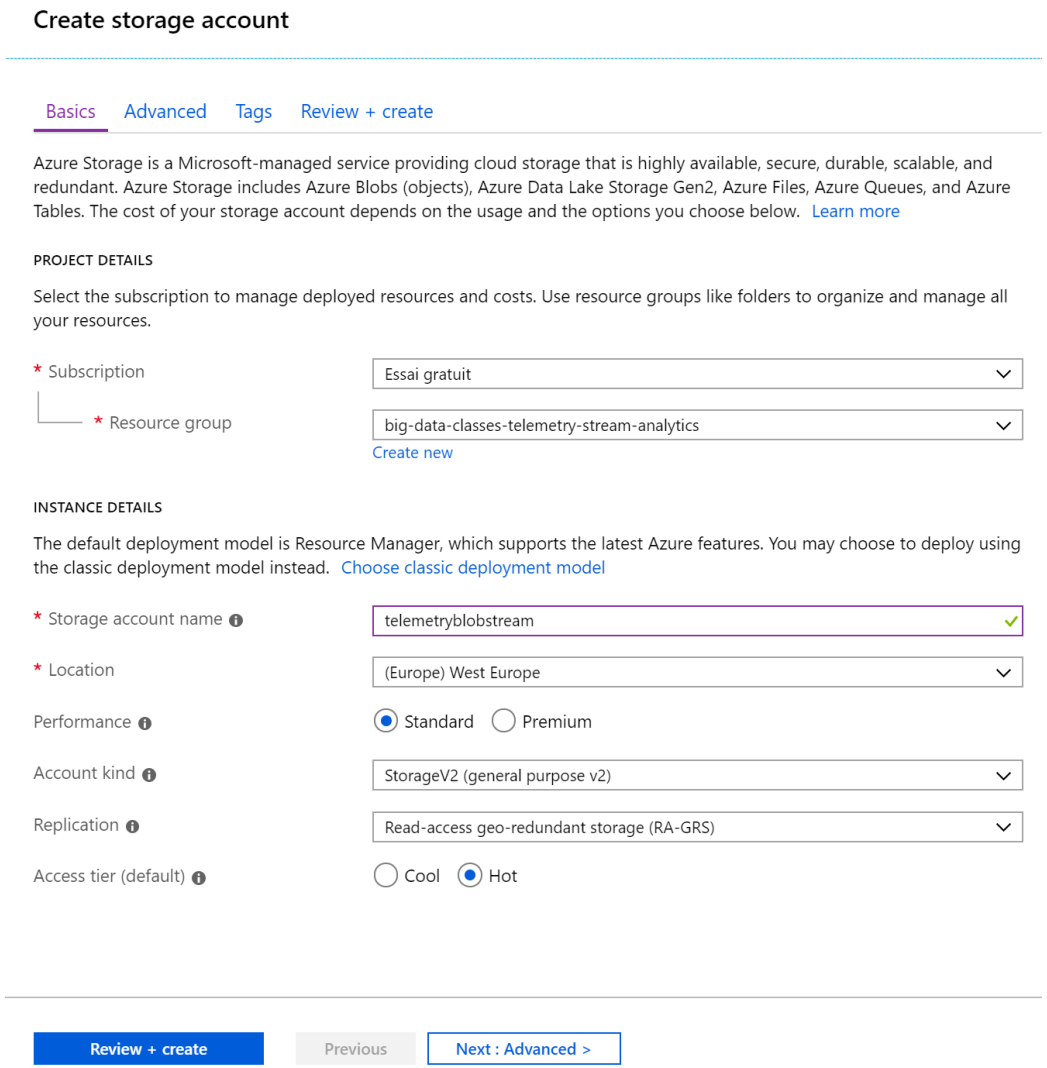


Blob Storage

REST-based object storage for unstructured data

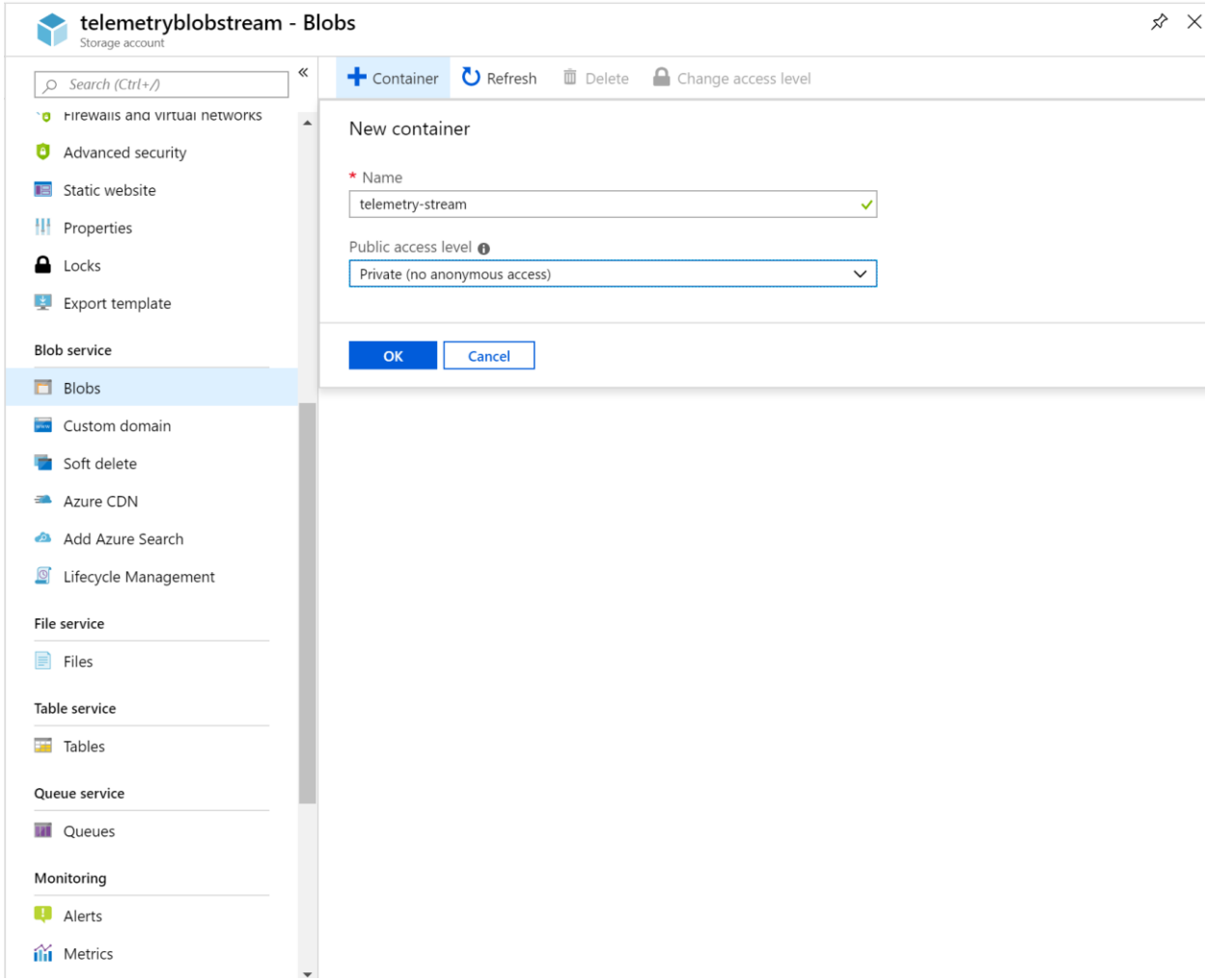
Blob Storage Creation

step 5



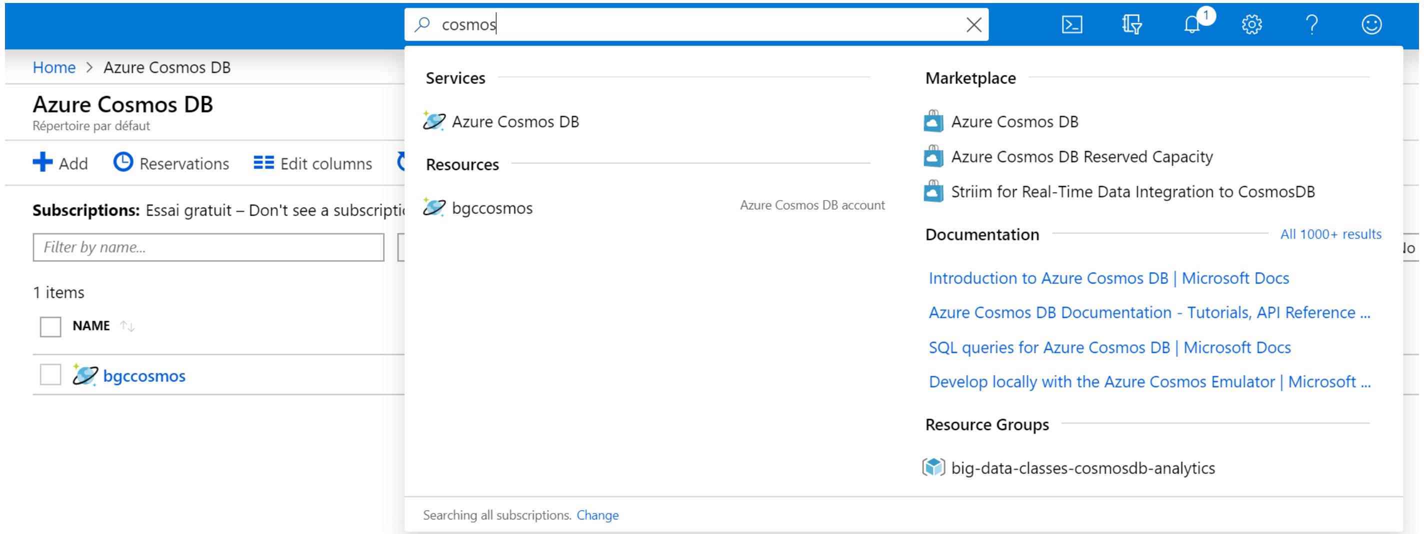
Create a blob storage container

step 6



Cosmos DB Creation

step 7

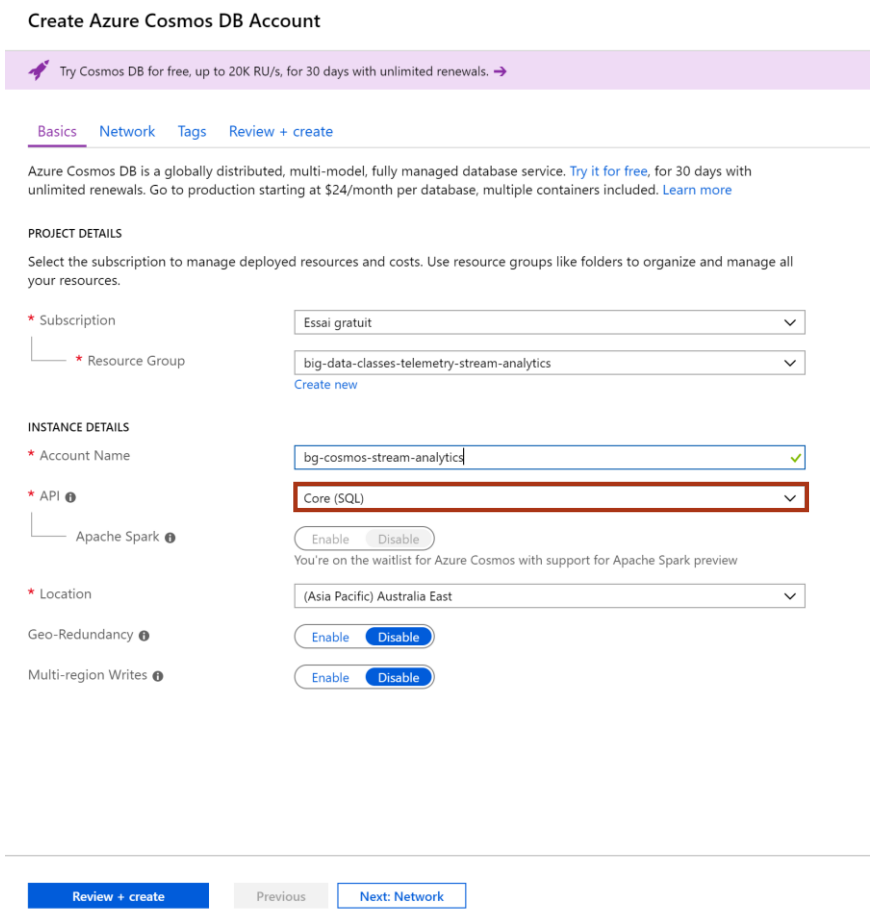


Azure Cosmos DB

Globally distributed, multi-model database for any scale

Cosmos DB Creation

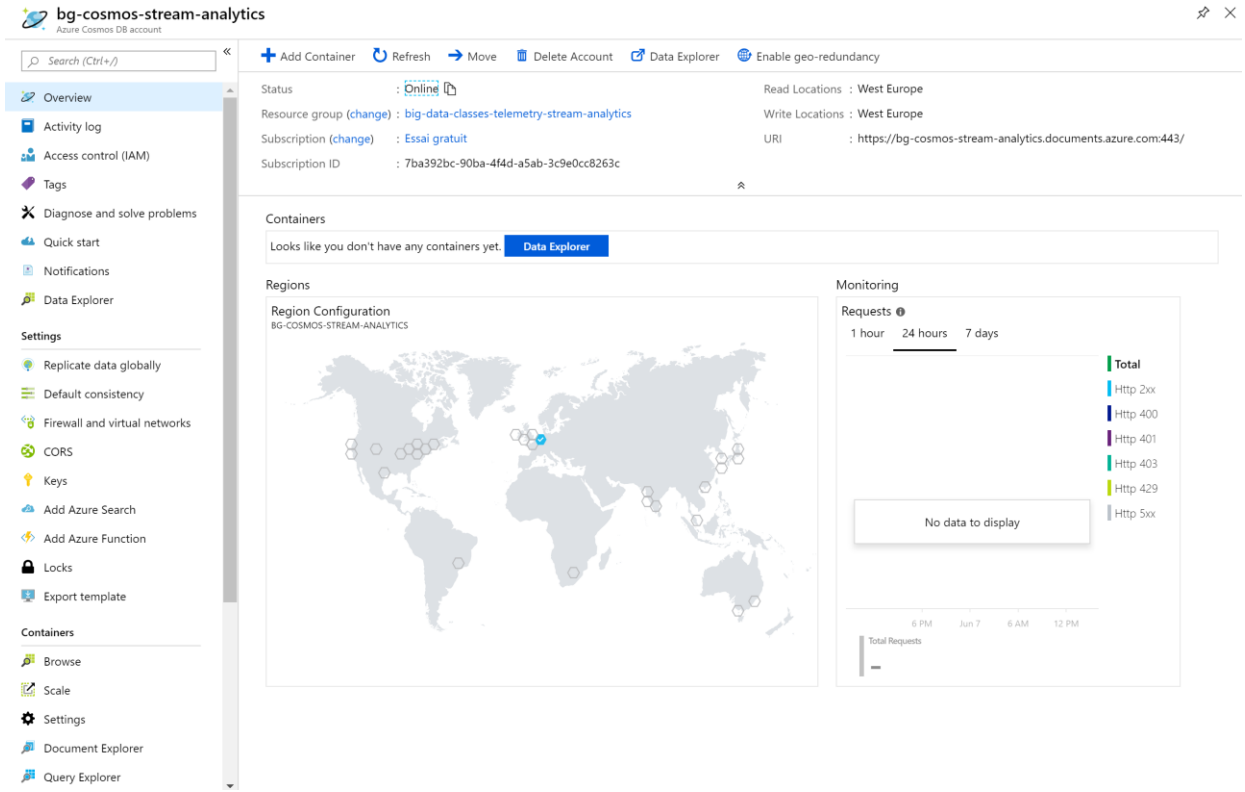
step 7



For the purpose of this architecture we will choose the Core(SQL) for the API.

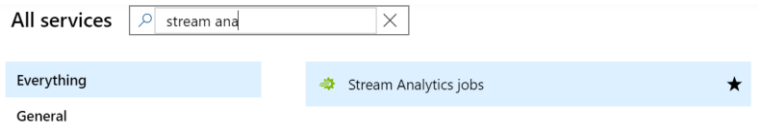
Cosmos DB Creation

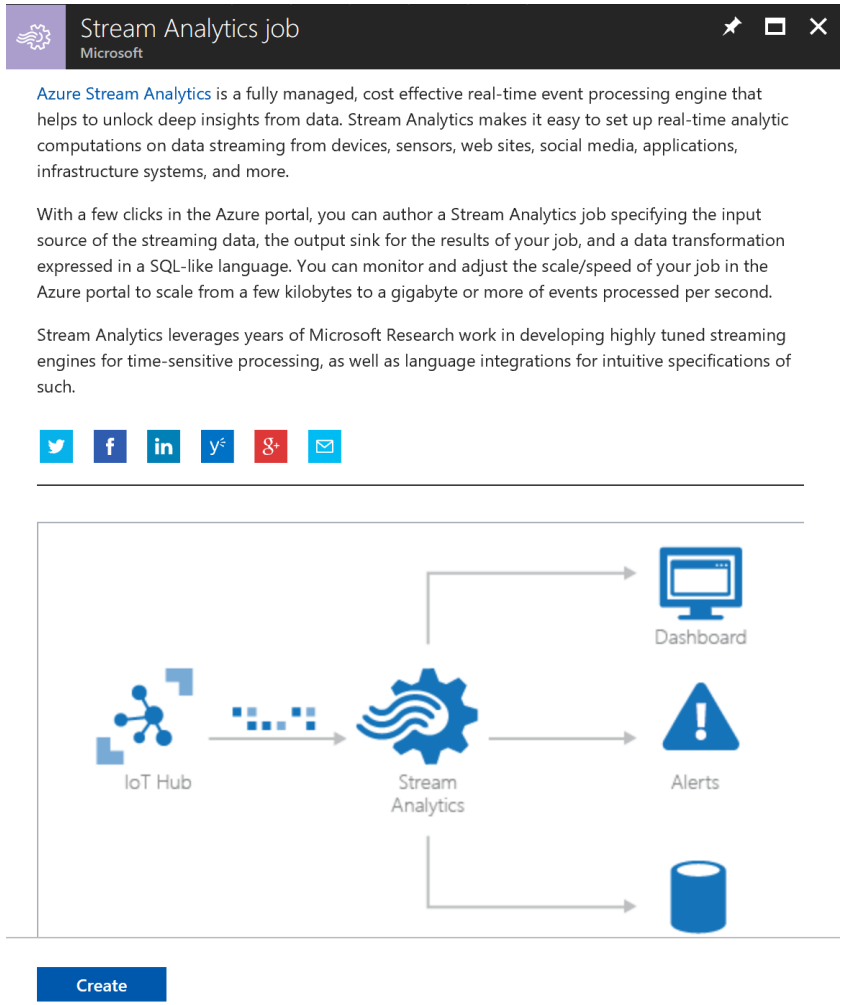
step 7



Stream Analytics job creation

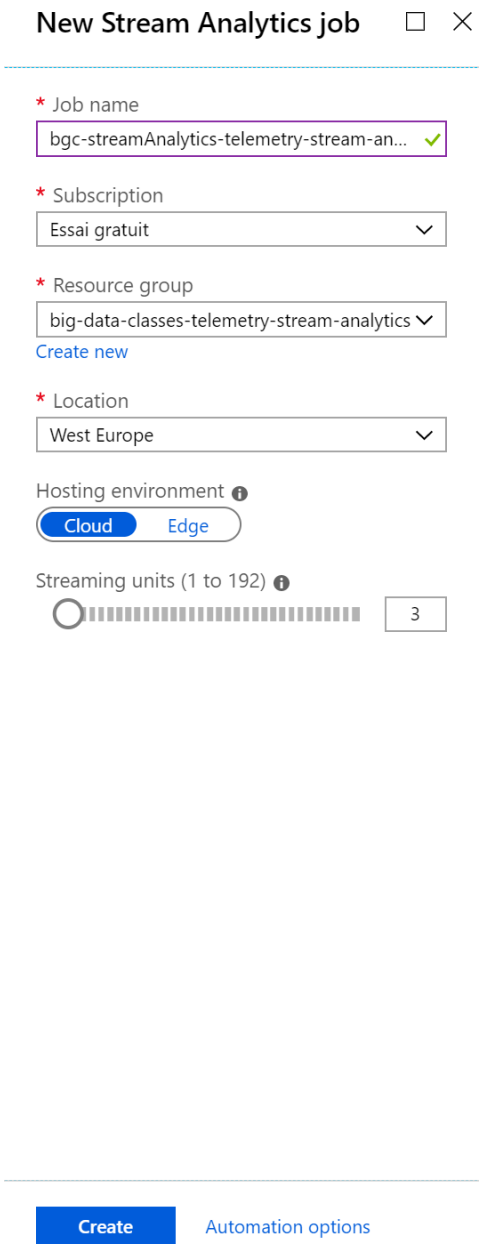
step 8





Stream Analytics job creation

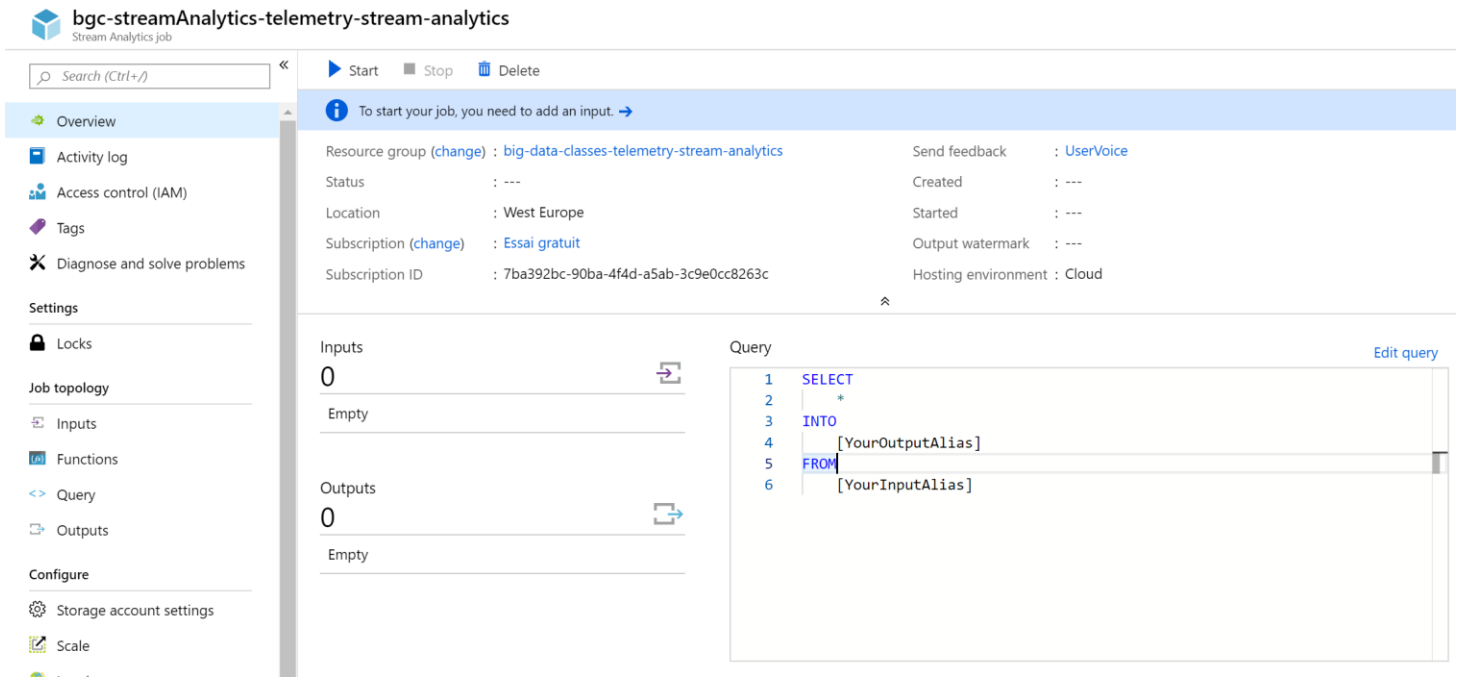
step 8



- Name: choose a different name
- Subscription : choose between available subscriptions
- Resource group: select the previously created group
- Hosting environment: cloud

Stream Analytics job configuration

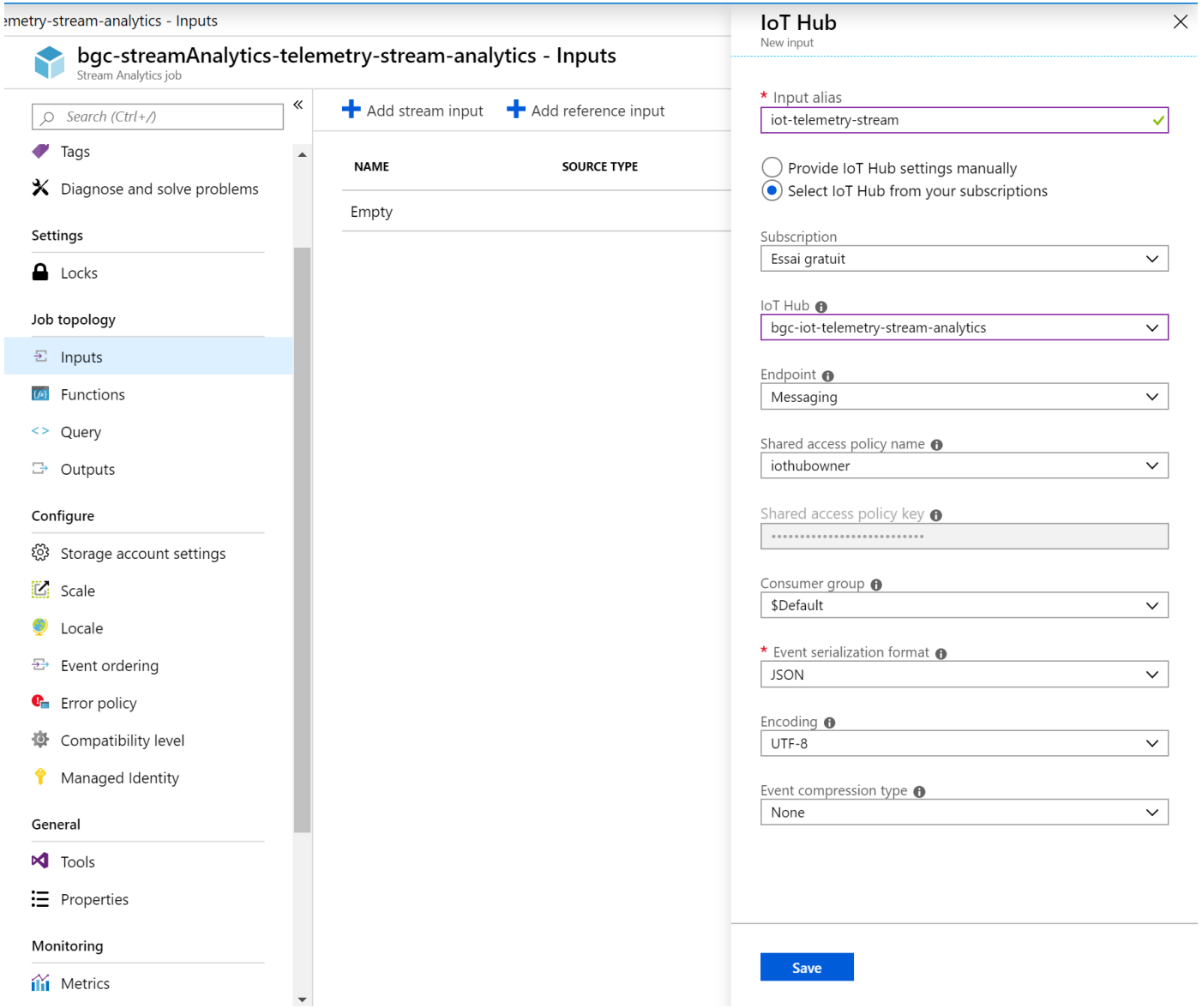
step 8



Stream Analytics job configuration

step 9

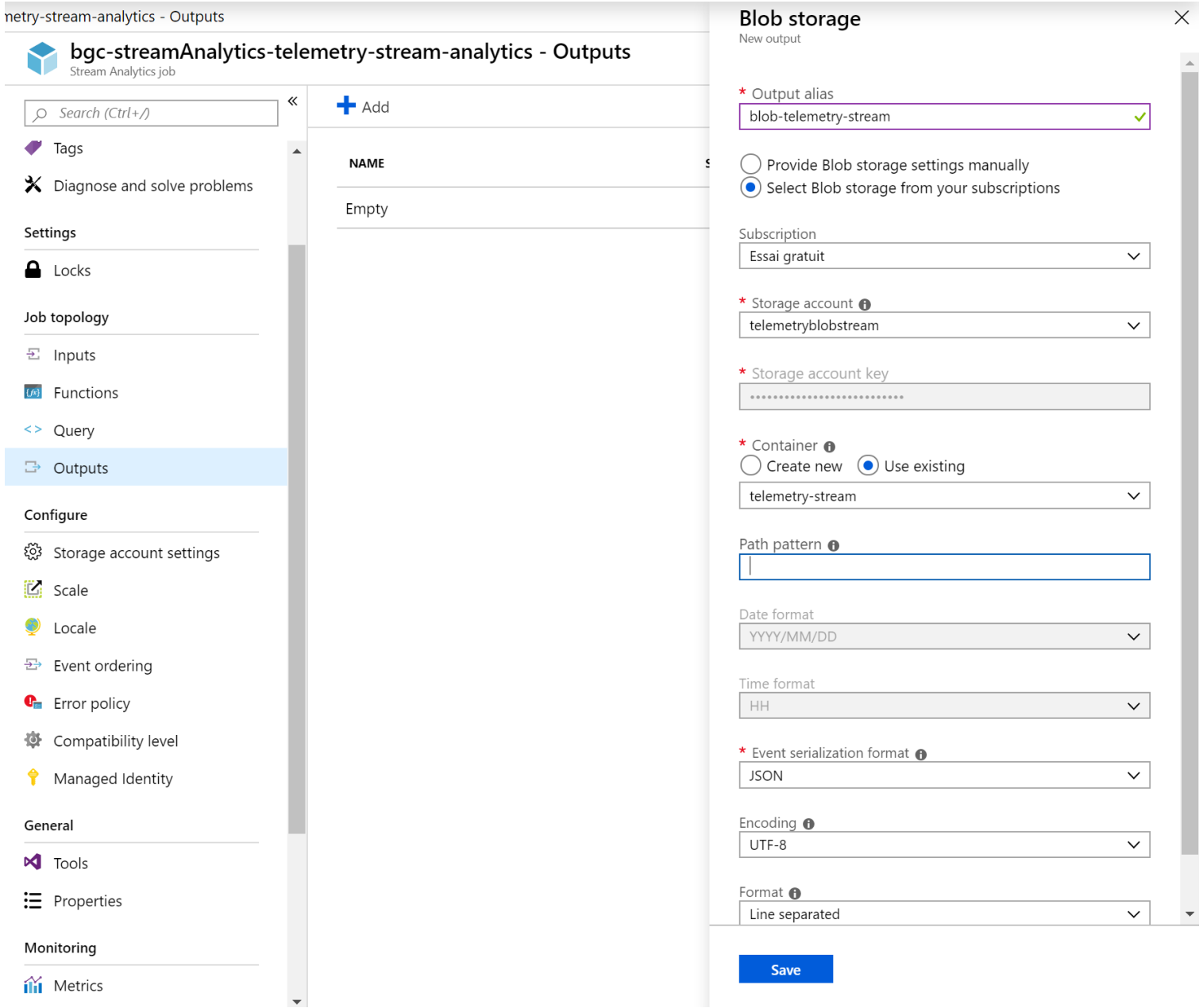
input : iot hub



Stream Analytics job configuration

step 9

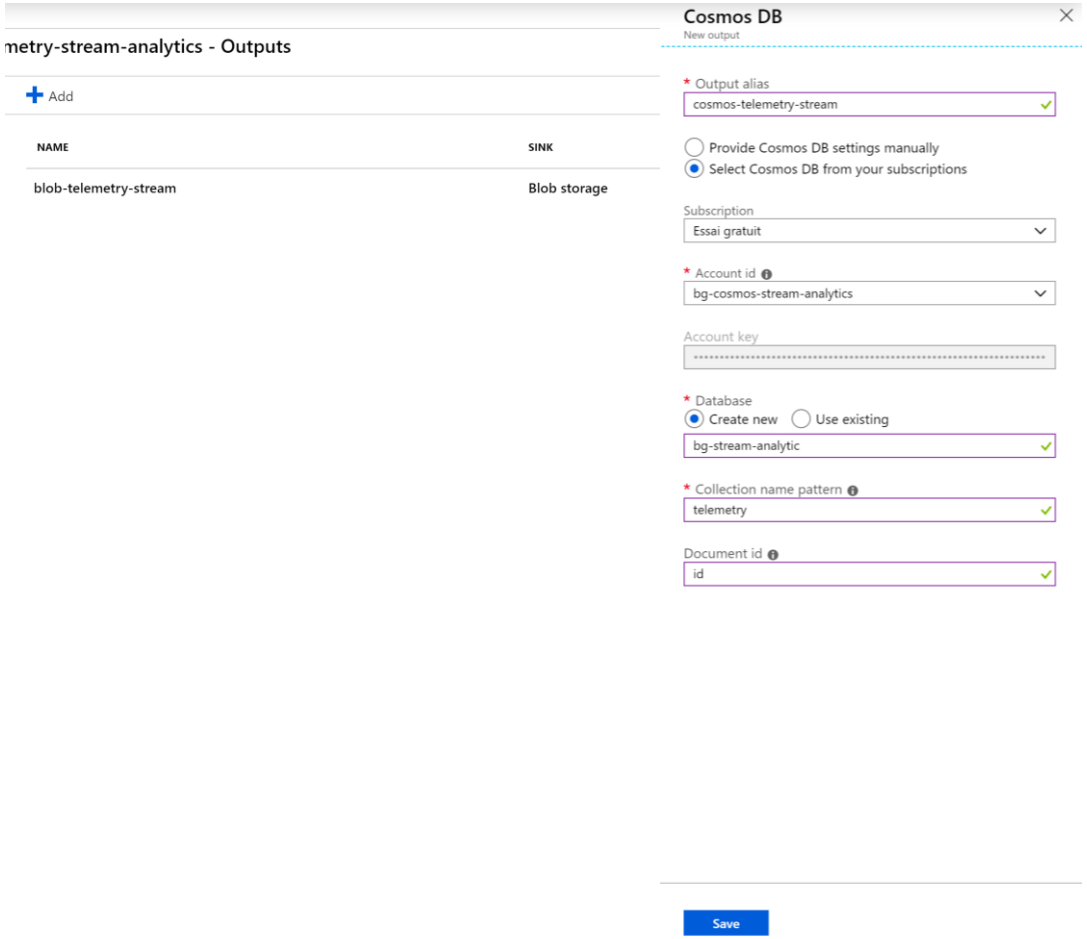
output : Blob storage



Stream Analytics job configuration

step 9

output : Cosmos DB



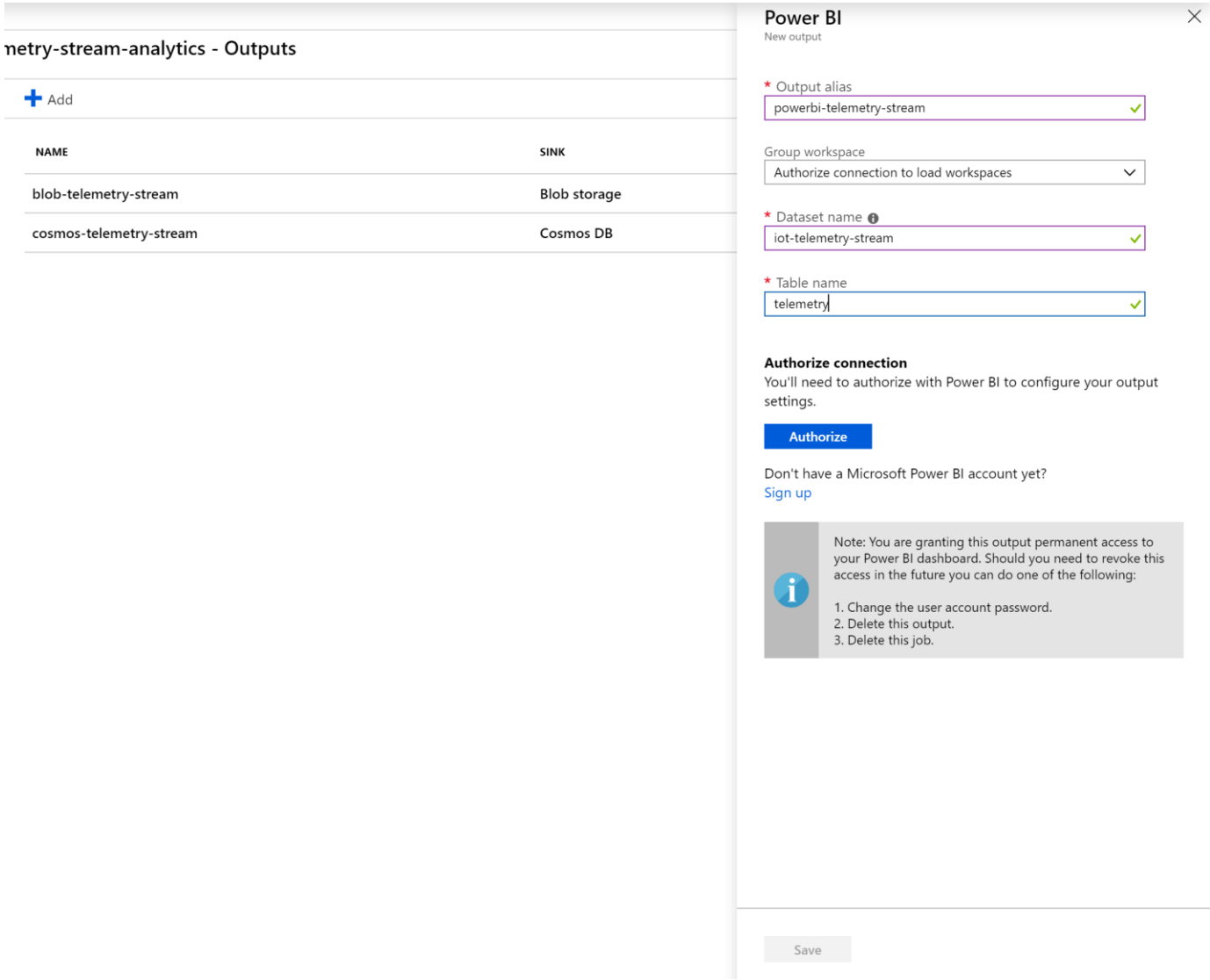
- Document id: The name of the field in output events used to specify the document id which insert or update operations are based on.

In our example, we are using the field “id” for that.

Stream Analytics job configuration

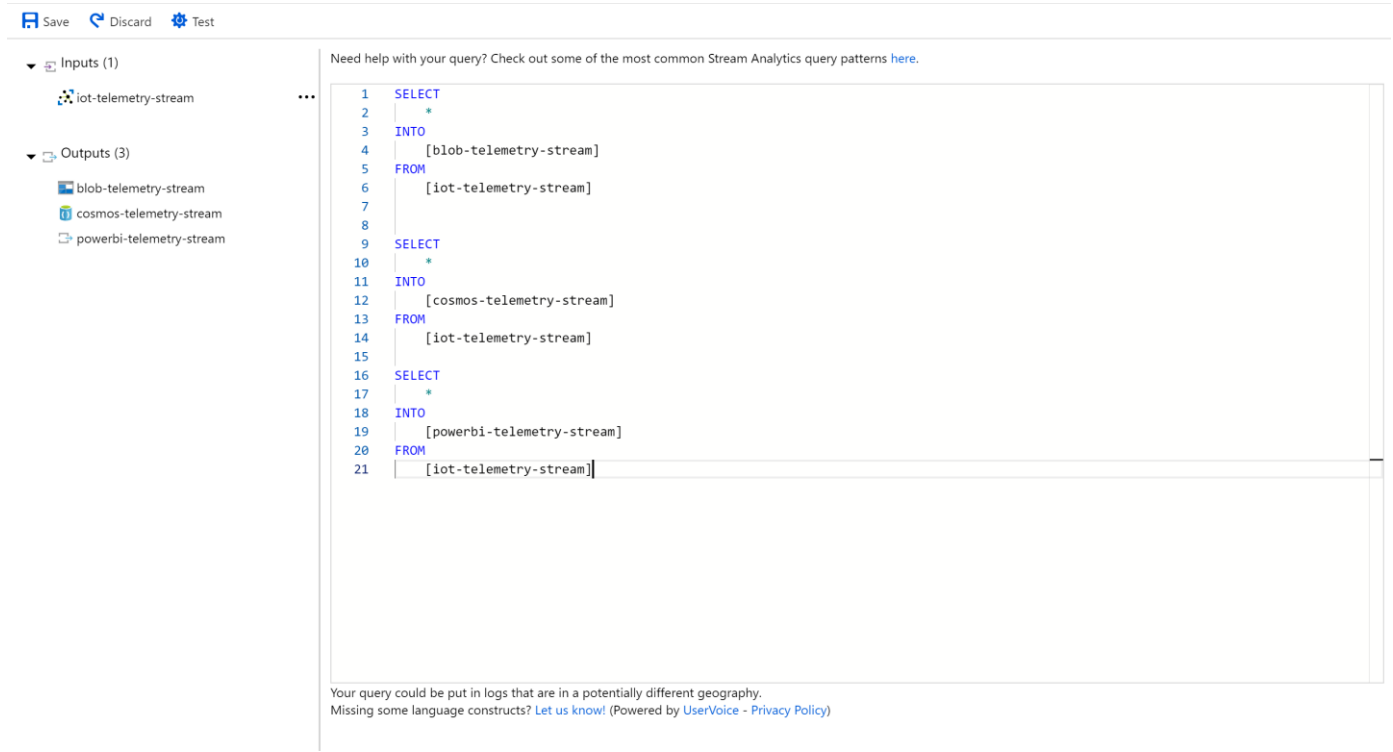
step 9

output : Power BI



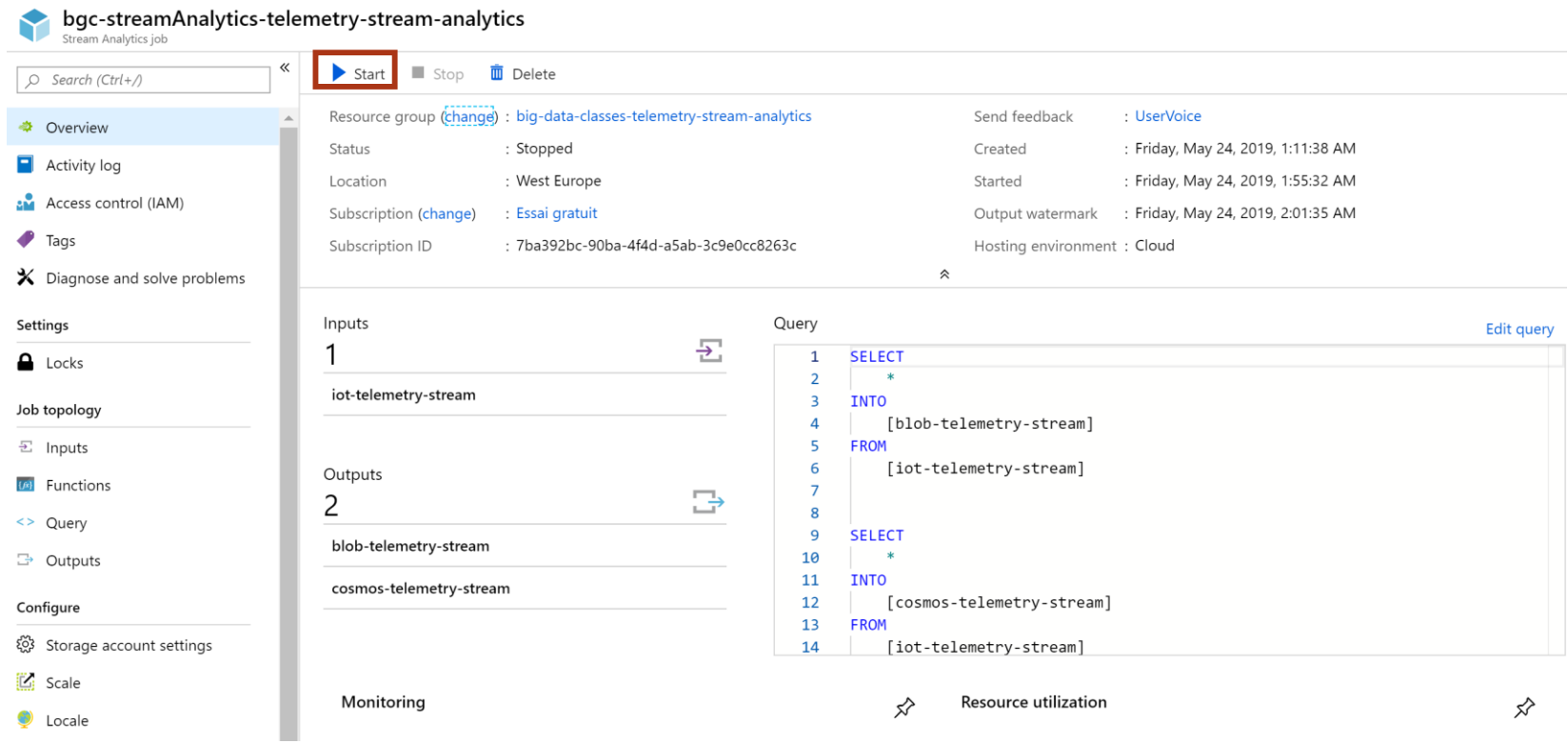
Stream Analytics job configuration: query

step 10



Start Stream Analytics job

step 11



Launch app

step 12

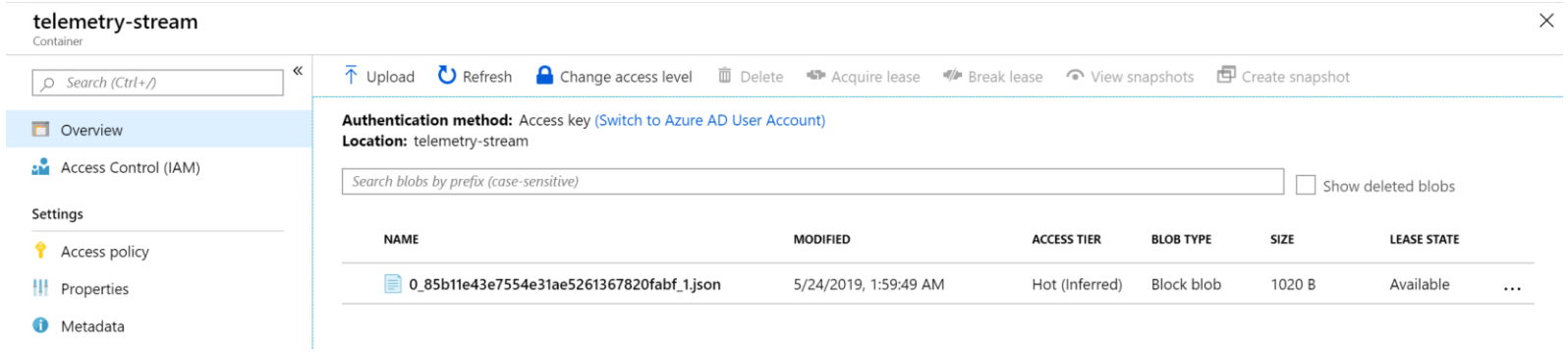
Launch big_data_classes_iot_stream.js:

```
C:\Users\alexa\Documents\Cours\Big Data\iot>node big_data_classes_iot_stream.js
Client connected
Sending message: {"id":"spaceneedle","temperature":22.80552350706524,"localisation":{"type":"Point","coordinates":[2.348
8,48.8534]},"Owner":{"CompanyName":"Miscrosoft","id_companyName":"4309718283412"}}
send status: MessageEnqueued
Sending message: {"id":"spaceneedle","temperature":24.085870065029816,"localisation":{"type":"Point","coordinates":[2.34
88,48.8534]},"Owner":{"CompanyName":"Miscrosoft","id_companyName":"4309718283412"}}
send status: MessageEnqueued
Sending message: {"id":"spaceneedle","temperature":29.415235594825035,"localisation":{"type":"Point","coordinates":[2.34
88,48.8534]},"Owner":{"CompanyName":"Miscrosoft","id_companyName":"4309718283412"}}
send status: MessageEnqueued
```

Verify

step 13

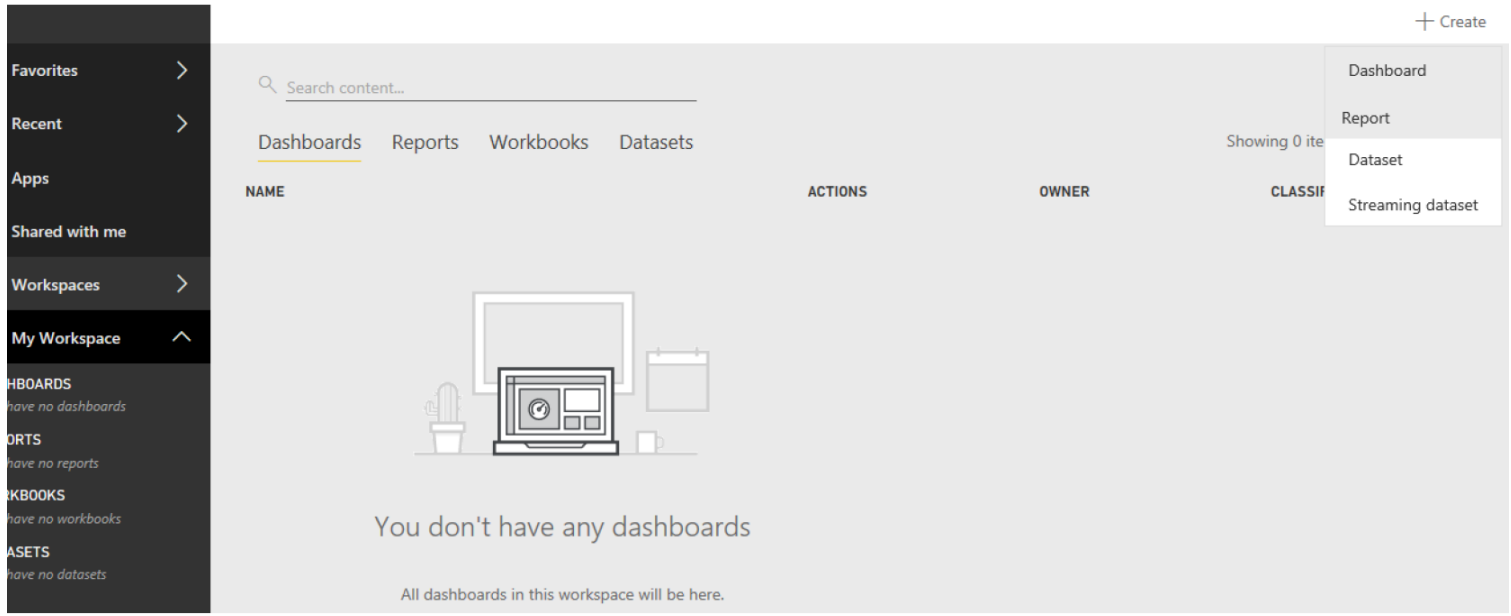
Check if the data is sending in blob storage account:



Blob should be appear in your container.

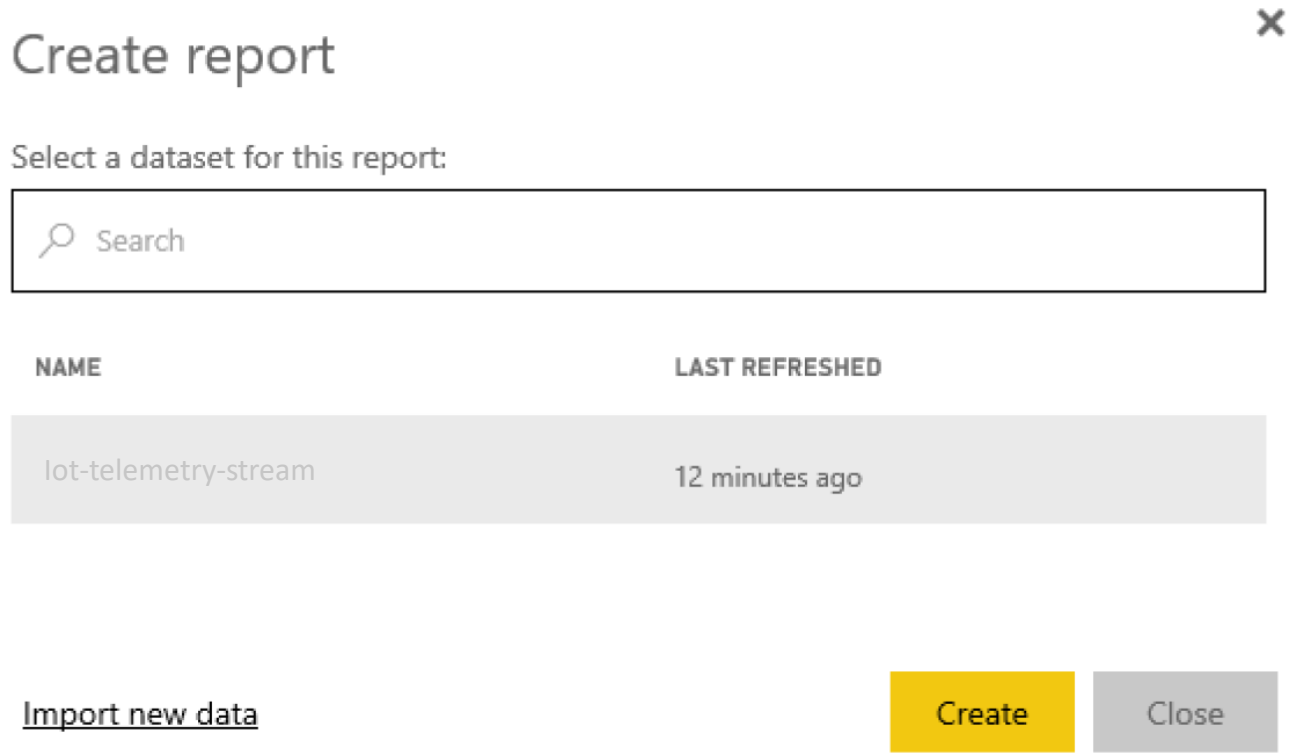
Power BI report: creation

step 14



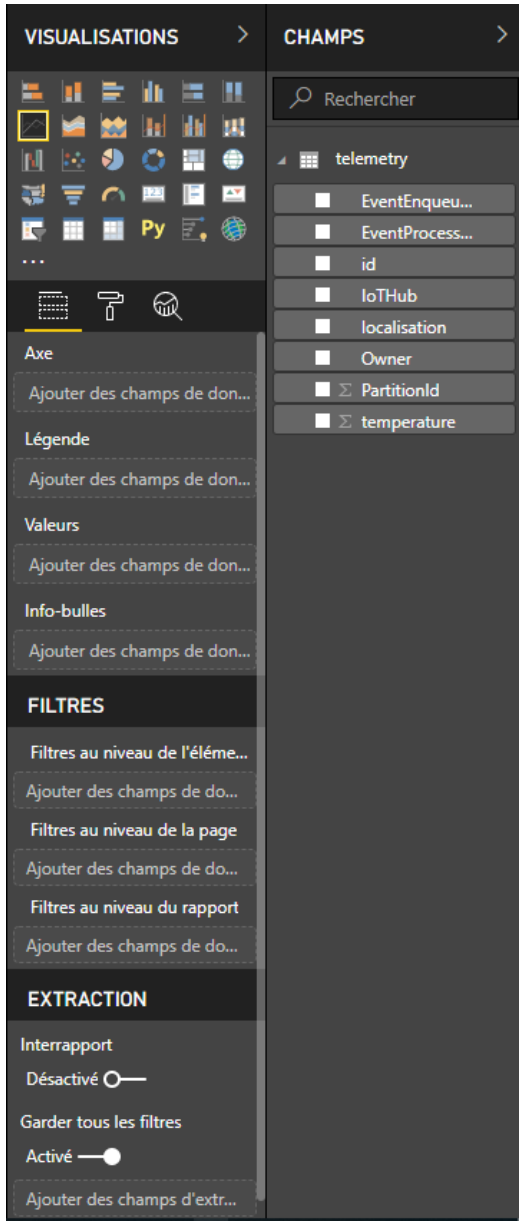
Power BI report: creation

step 14



Power BI report: creation

step 14



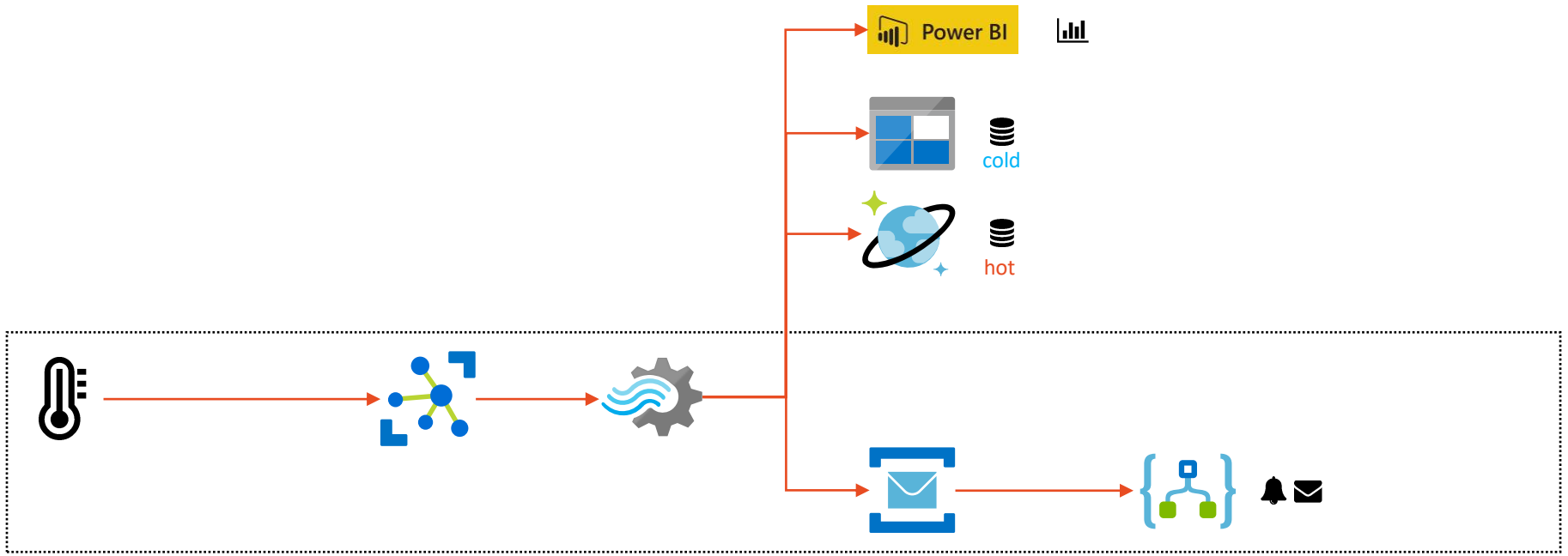
Add devices

step 15

Add 2 or 3 devices:

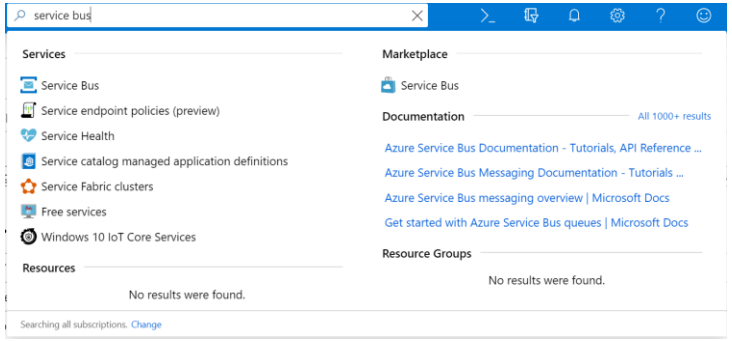
- Copy – paste the “big_data_classes_iot_stream.js” in multiple files (one for each future device)
- Change the coordinates point and device name.
- Add devices in iot hub
- Use the new connection string in your js file created earlier
- You could see your different device in Power BI

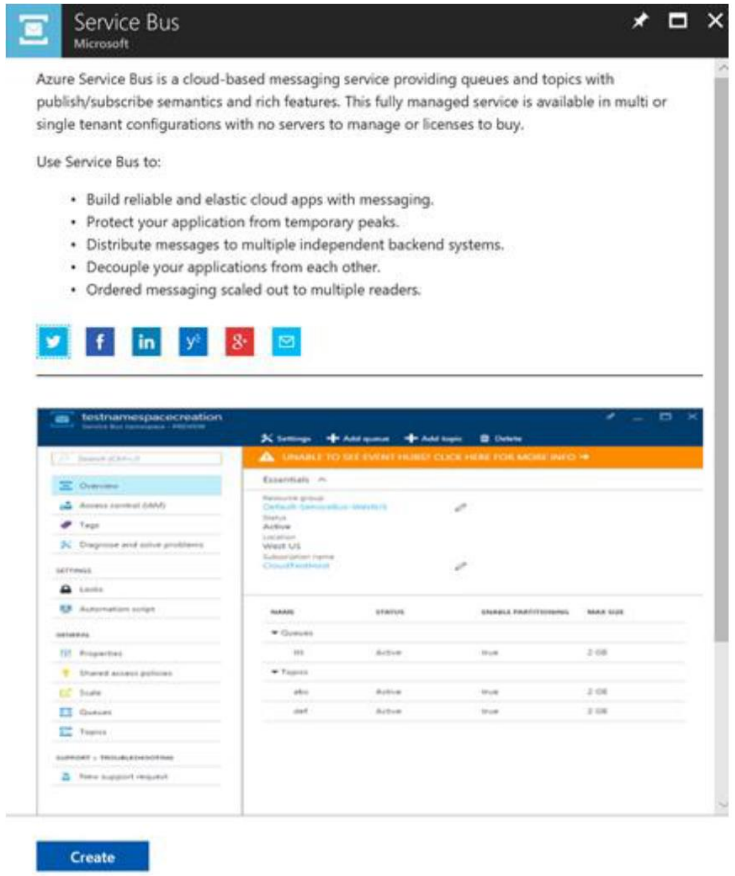
mail alert



Service bus creation

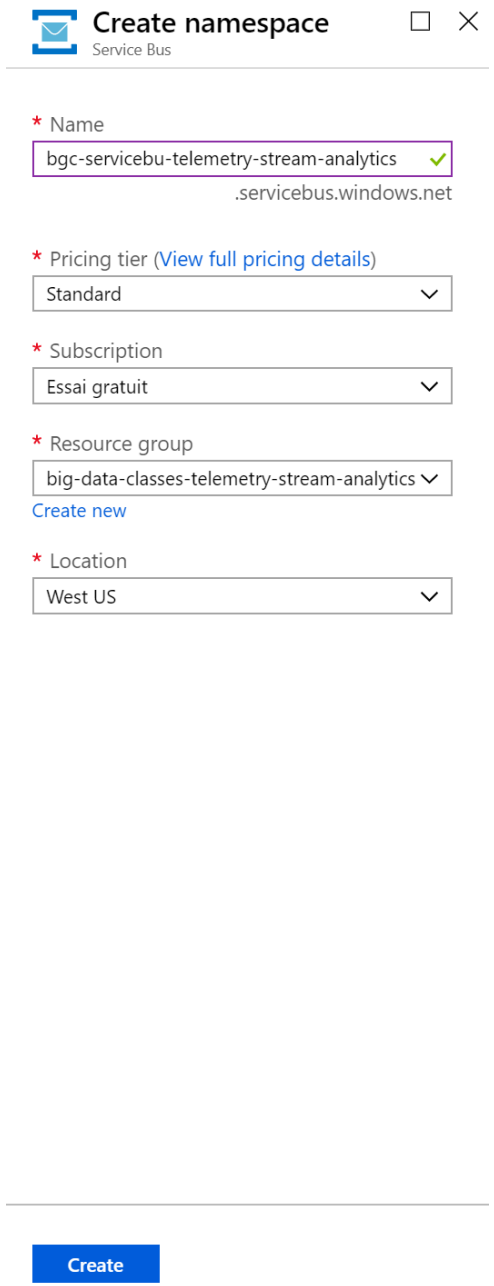
step 1





Service bus creation

step 1

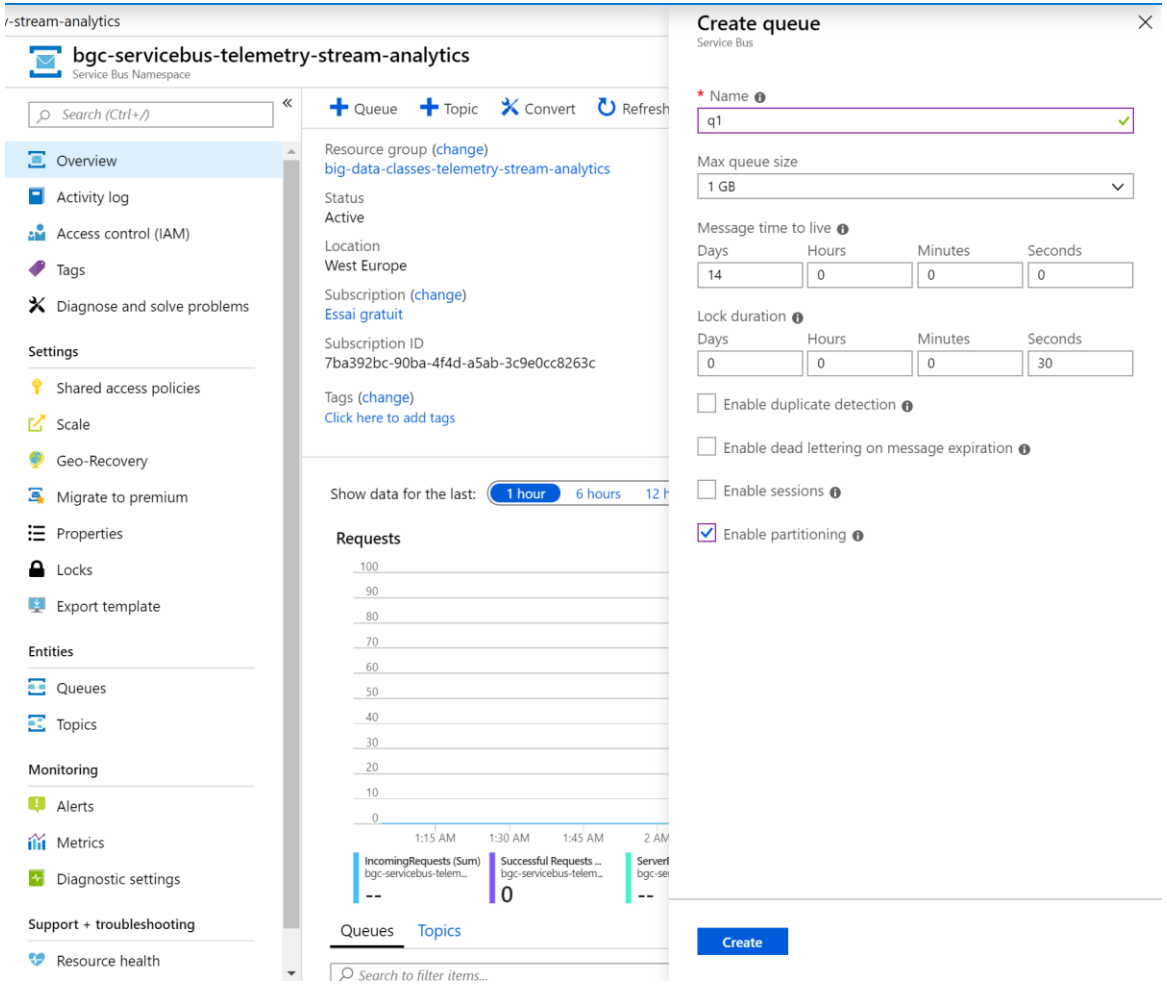


- Name: choose a different name
- Pricing tier: Standard
- Subscription : choose between available subscriptions
- Resource group: select the previously created group

Service bus configuration

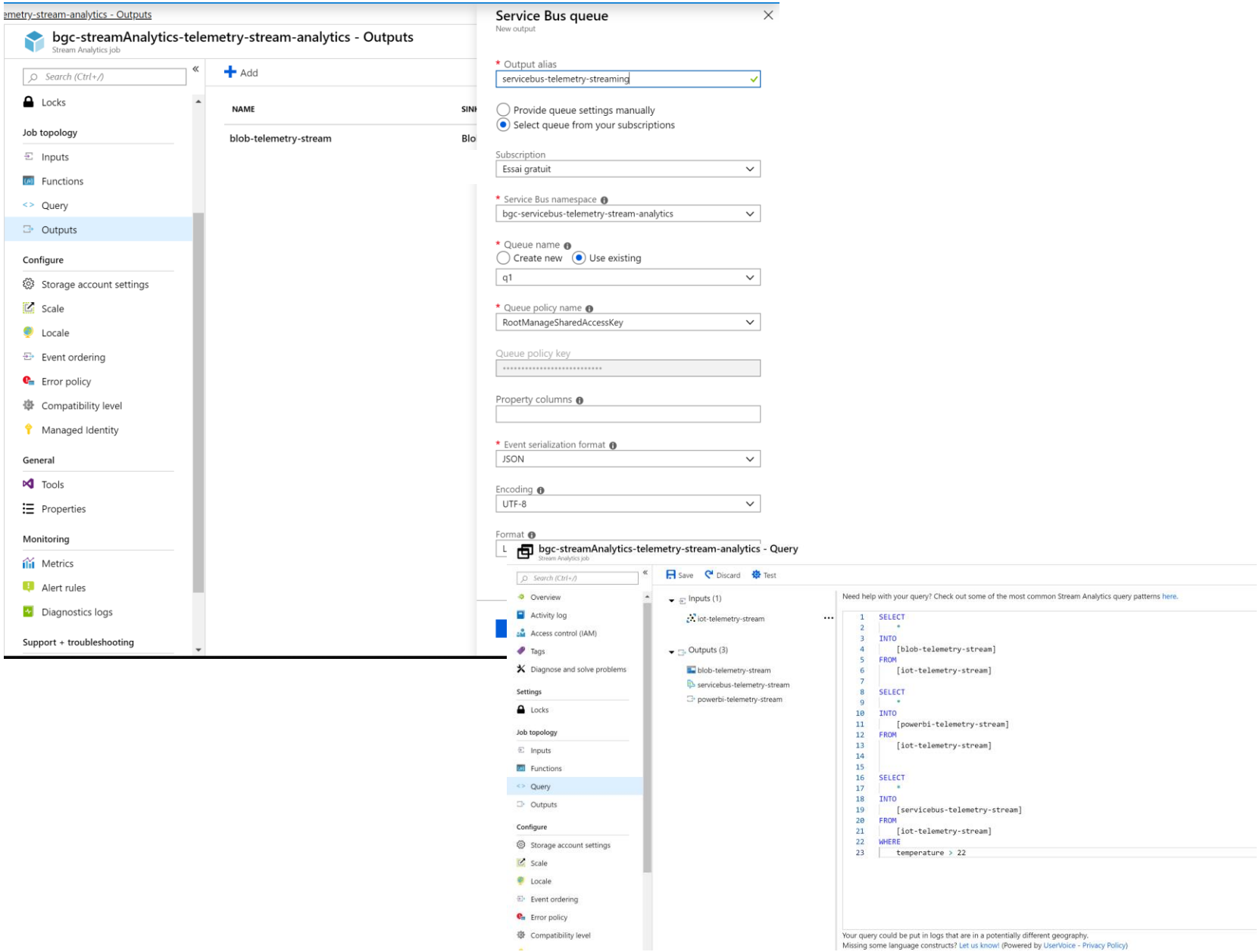
step 2

Create a queue:



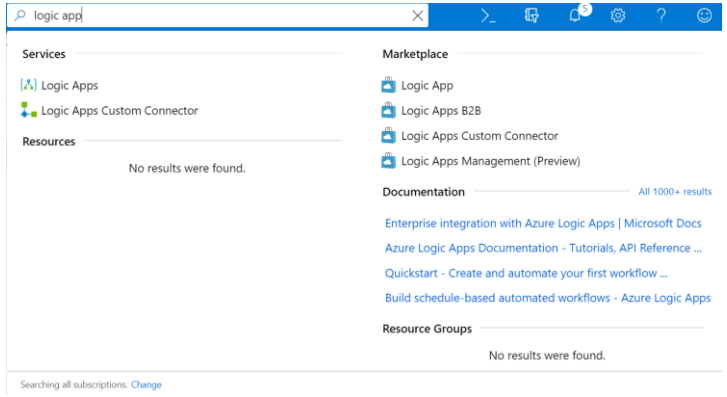
Stream Analytics configuration: output

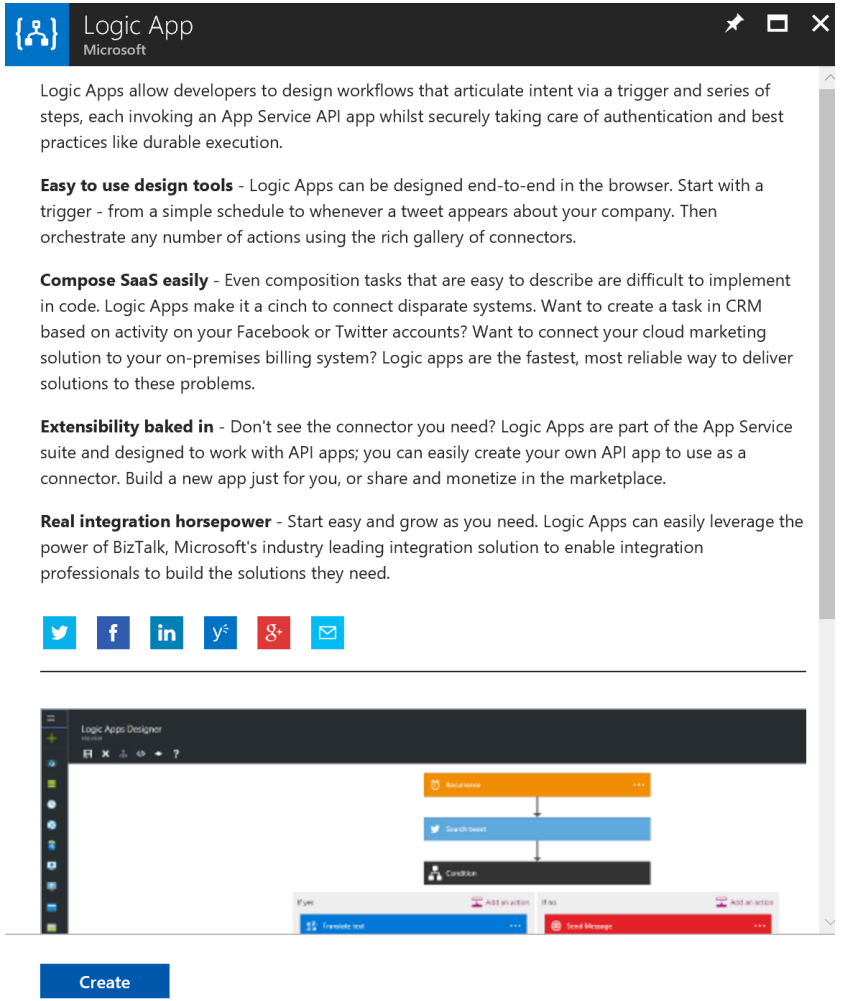
step 3



Logic app creation

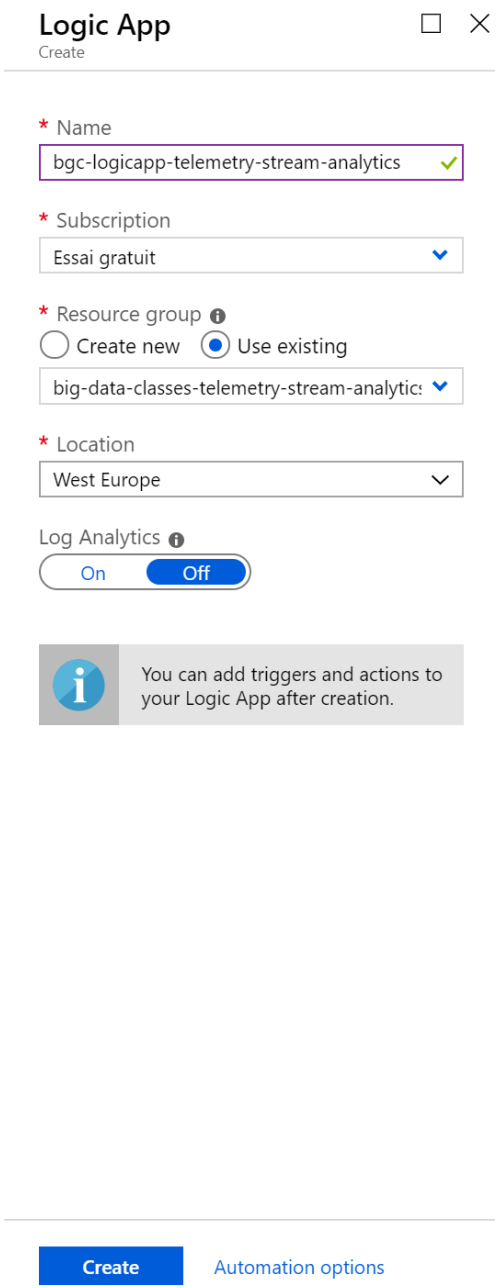
step 4





Logic app creation

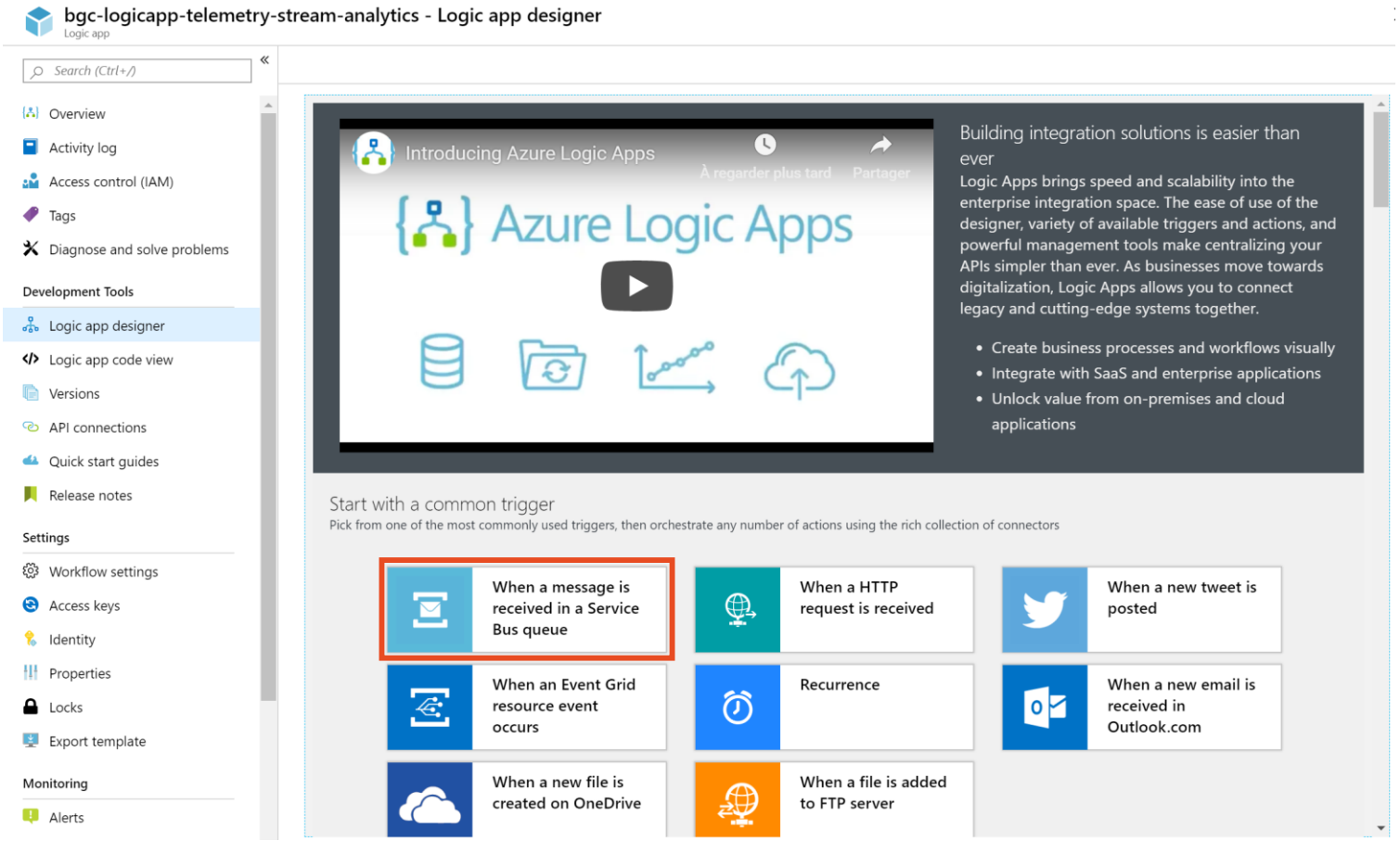
step 4



- Name: choose a different name
- Subscription : choose between available subscriptions
- Resource group: select the previously created group

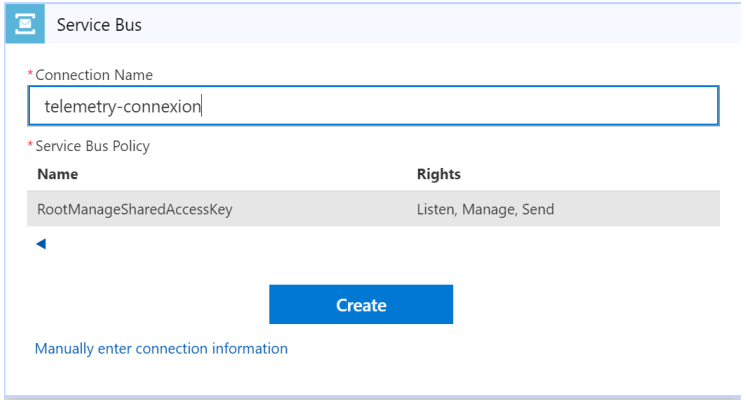
Logic app configuration

step 5

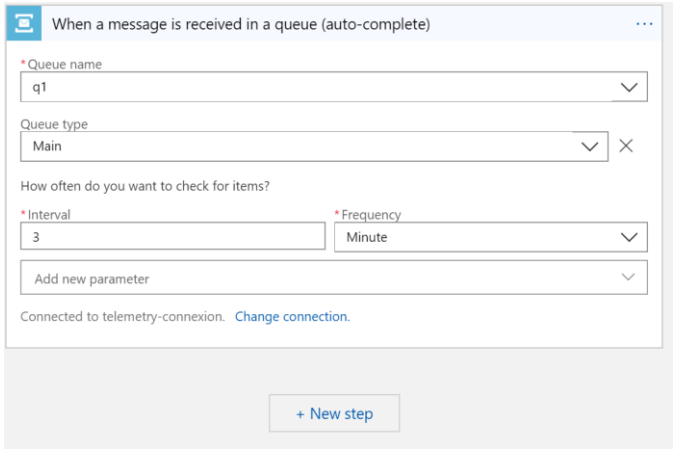


Logic app configuration

step 5



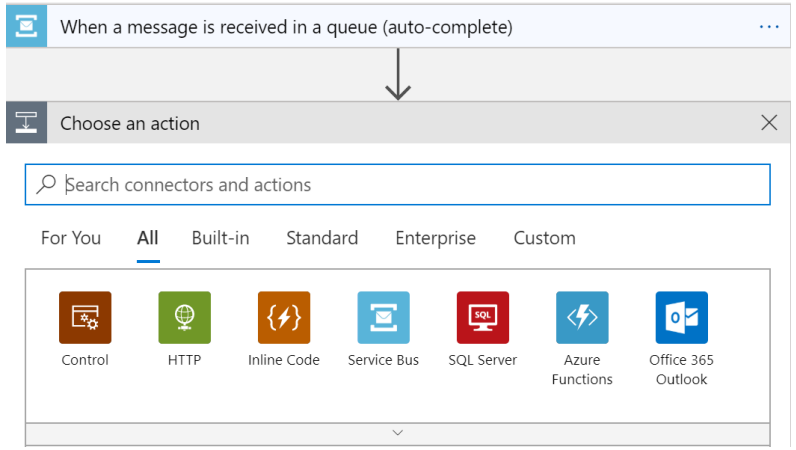
- Click on Create
- Select the service bus youcreatedpreviously
- Select the RootManageSharedAccessKeypolicy
- Give a name to the connection



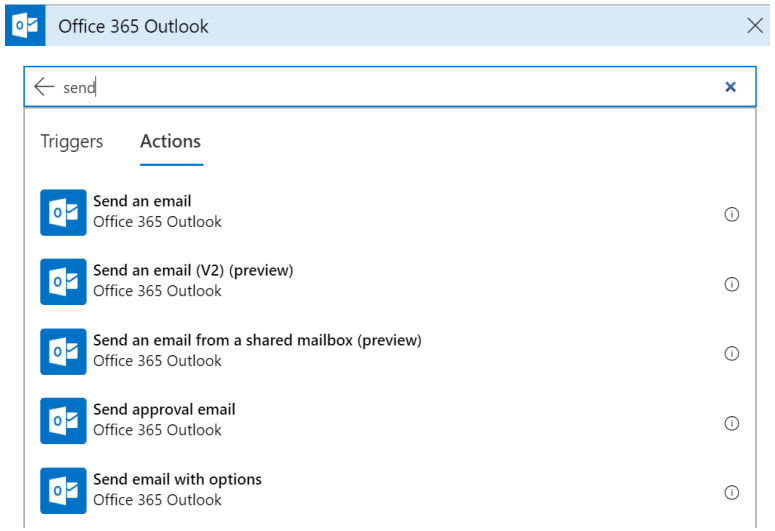
- Select the queue
- Click on Next step and then in Add an action

Logic app configuration

step 5



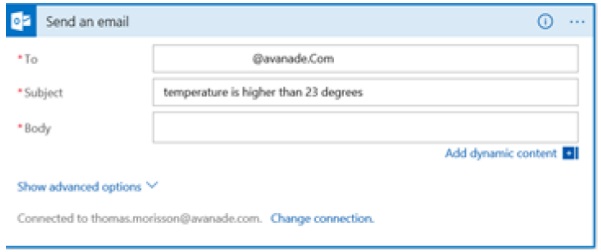
- Click on Next step and then in Add an action
- Look for Office 365 Outlook connector



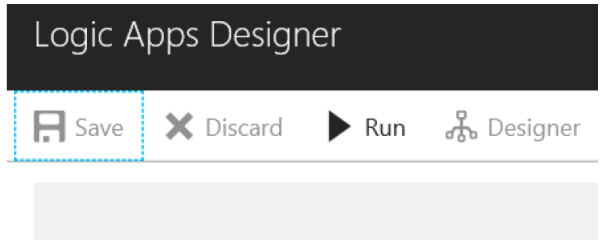
- Configure the « Send a email » box

Logic app configuration

step 5



- Configure the « Send a email » box



- Save the workflow, and run it

You can now follow the history of execution