How to create a .NET 8 C# console application to send messages from D2C (from Device to Azure IoTHub)

In this example we are going to create an application to simulate the Device.

From our application(Device) we will send a message to the Azure IoTHub

For additional info about Azure IoTHub Getting Start tutorials see the references:

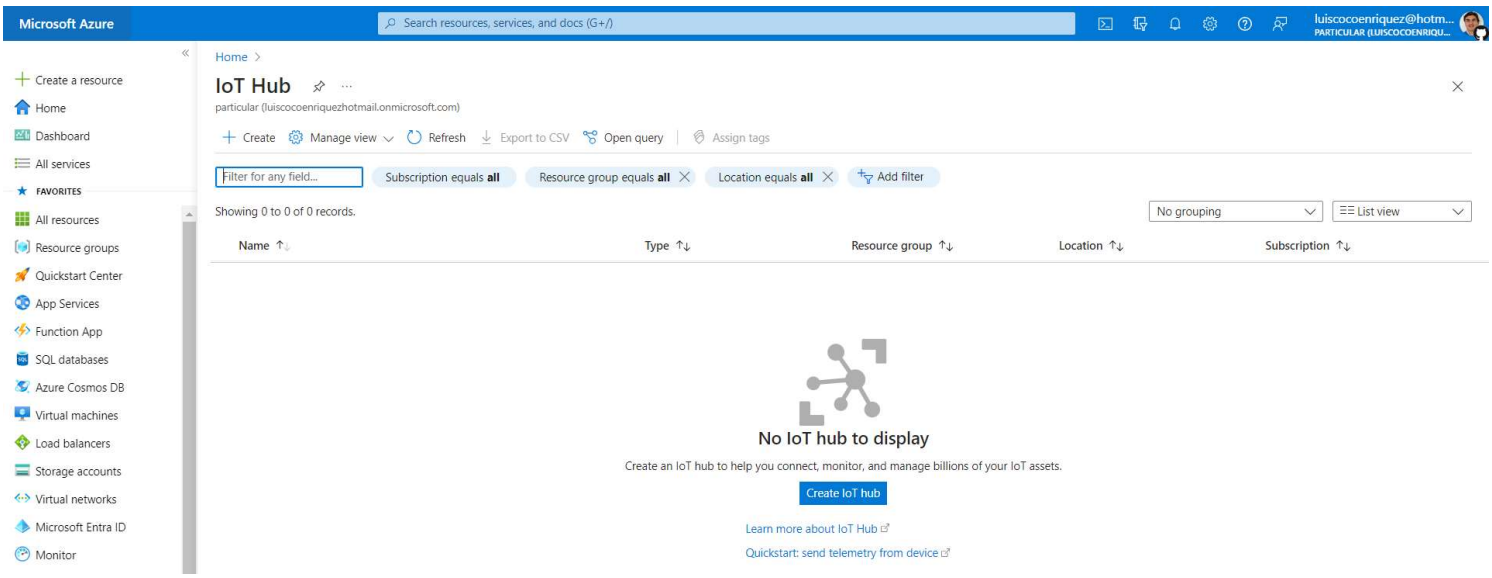
https://github.com/Azure/azure-iot-explorer

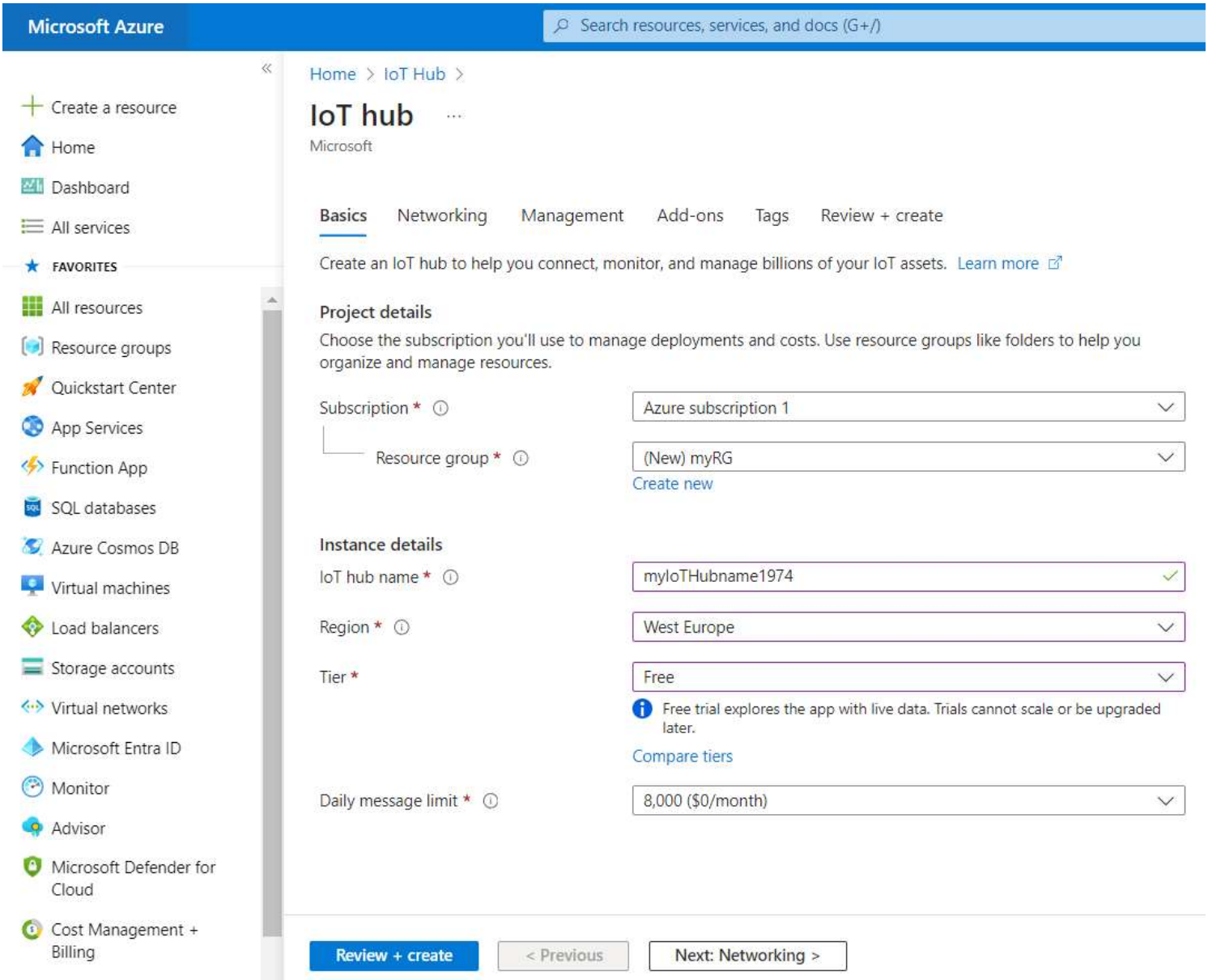
https://github.com/Azure/azure-iot-sdk-csharp

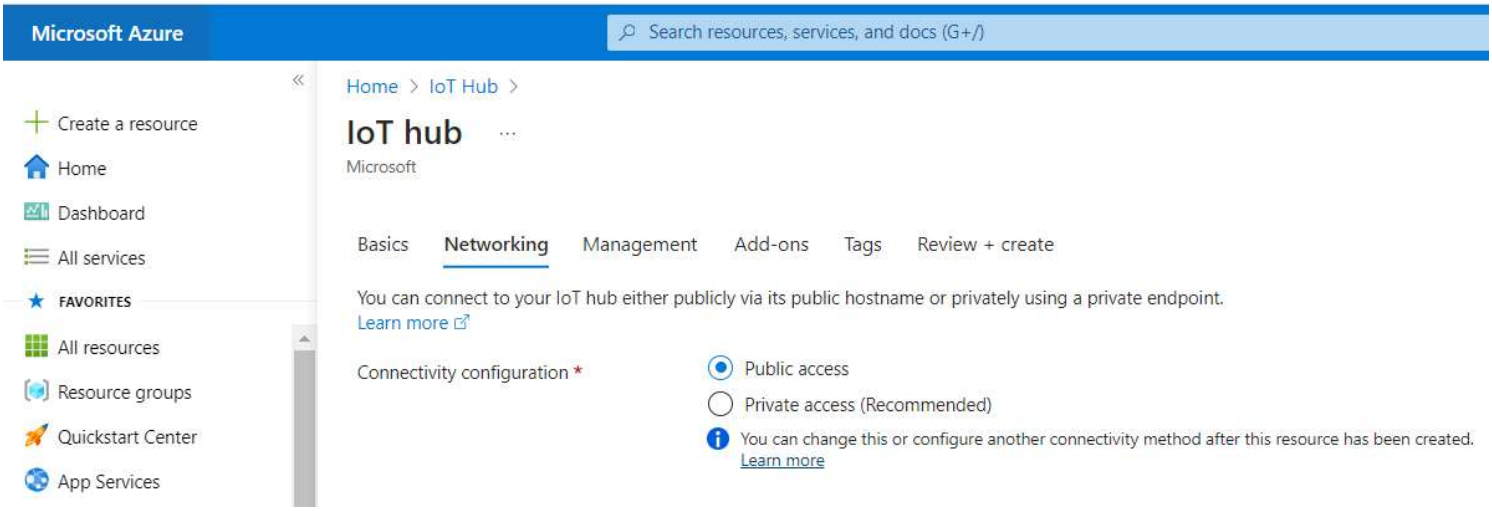
https://learn.microsoft.com/en-us/azure/iot-develop/quickstart-send-telemetry-iot-hub?source=recommendations&pivots=programming-language-csharp

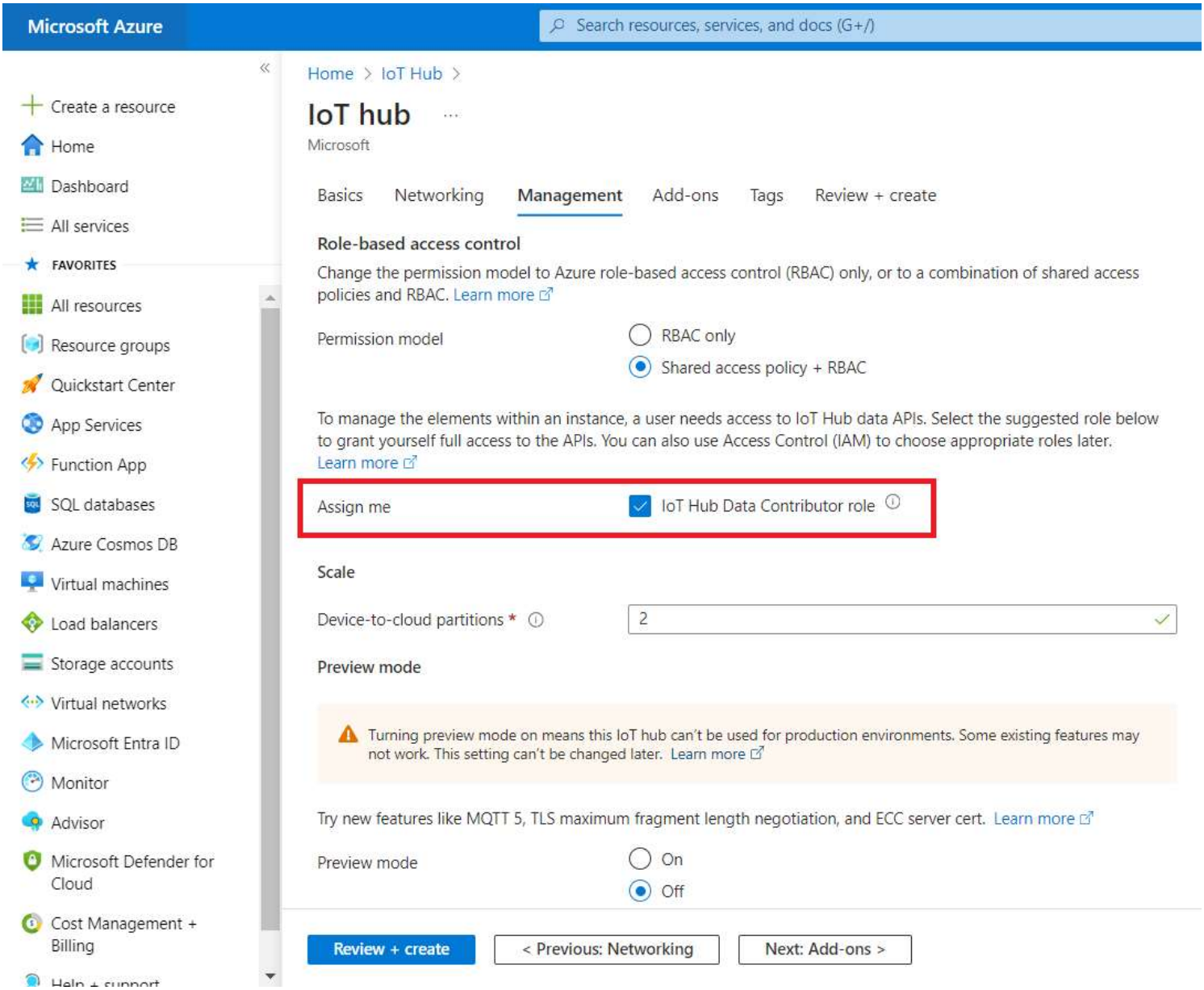
0. Prerequisites

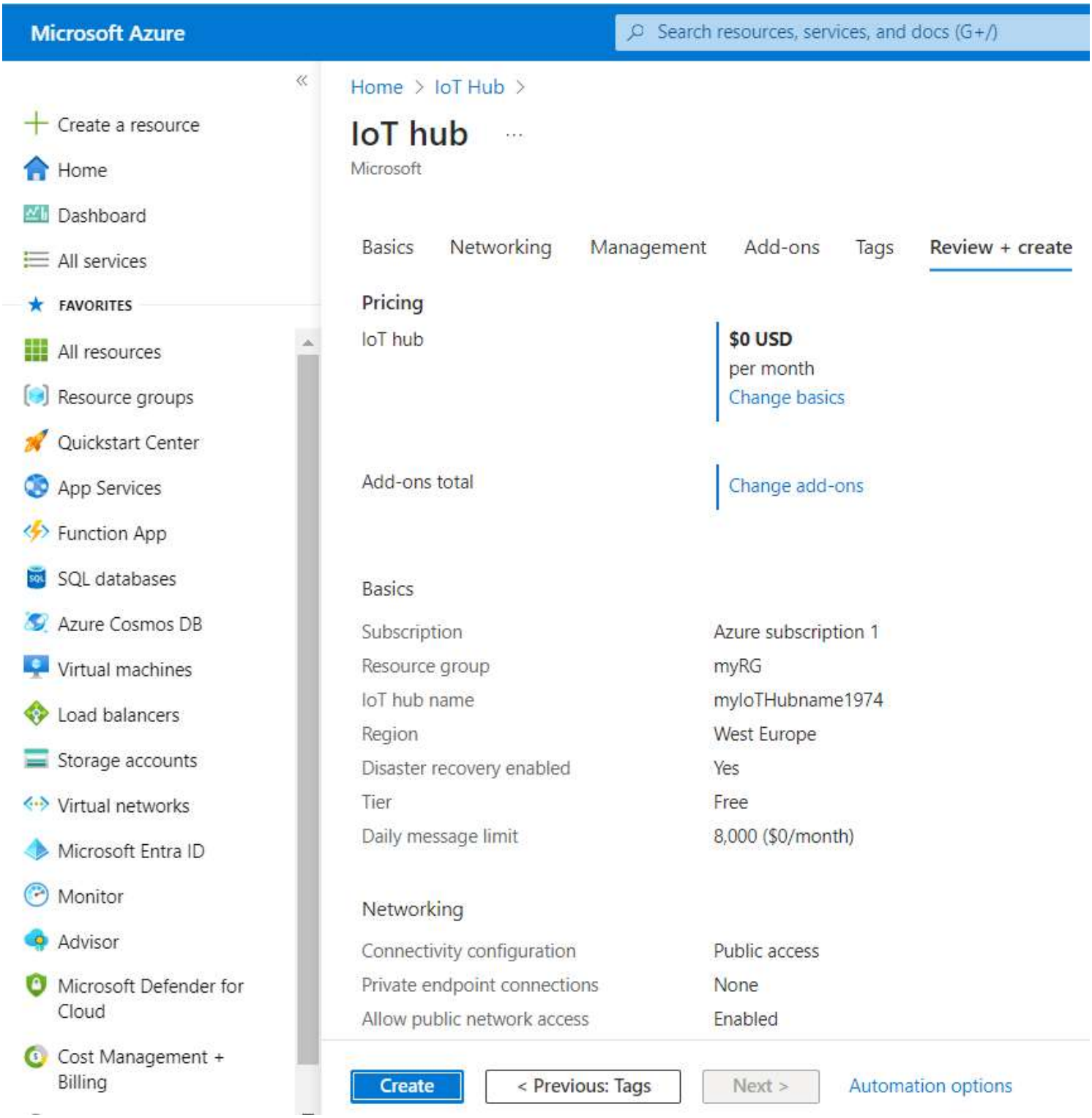
Create an Azure IoTHub

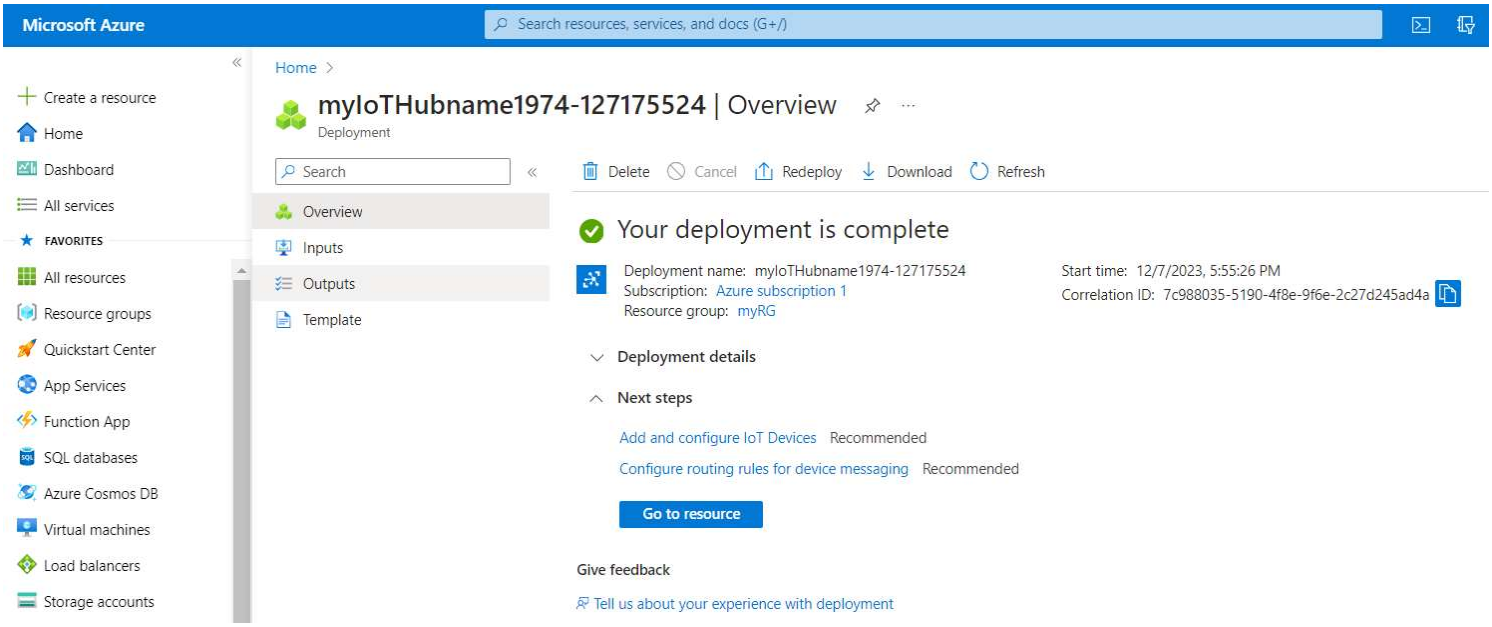


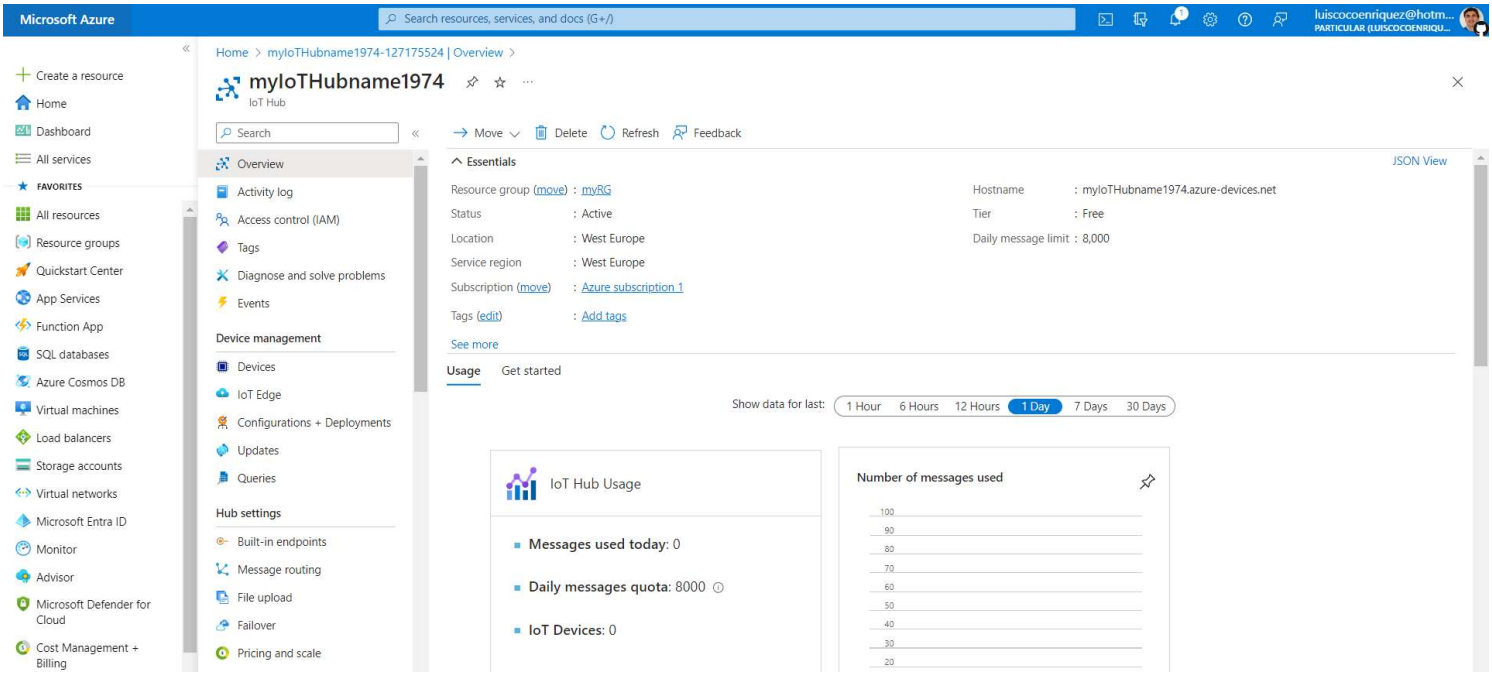

Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Microsoft Entra ID
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management + Billing
Home > IoT Hub >
IoT hub
Microsoft
Basics
Networking
Management
Add-ons
Tags
Review + create
Create an IoT hub to help you connect, monitor, and manage billions of your IoT assets. Learn more
Project details
Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.
Subscription *
Azure subscription 1
Resource group *
(New) myRG
Create new
Instance details
IoT hub name *
myIoTHubname1974
Region *
West Europe
Tier *
Free
Free trial explores the app with live data. Trials cannot scale or be upgraded later.
Compare tiers
Daily message limit *
8,000 ($0/month)
Review + create
< Previous
Next: Networking >


Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Home > IoT Hub >
IoT hub
Microsoft
Basics
Networking
Management
Add-ons
Tags
Review + create
You can connect to your IoT hub either publicly via its public hostname or privately using a private endpoint.
Learn more
Connectivity configuration *
Public access
Private access (Recommended)
You can change this or configure another connectivity method after this resource has been created.
Learn more

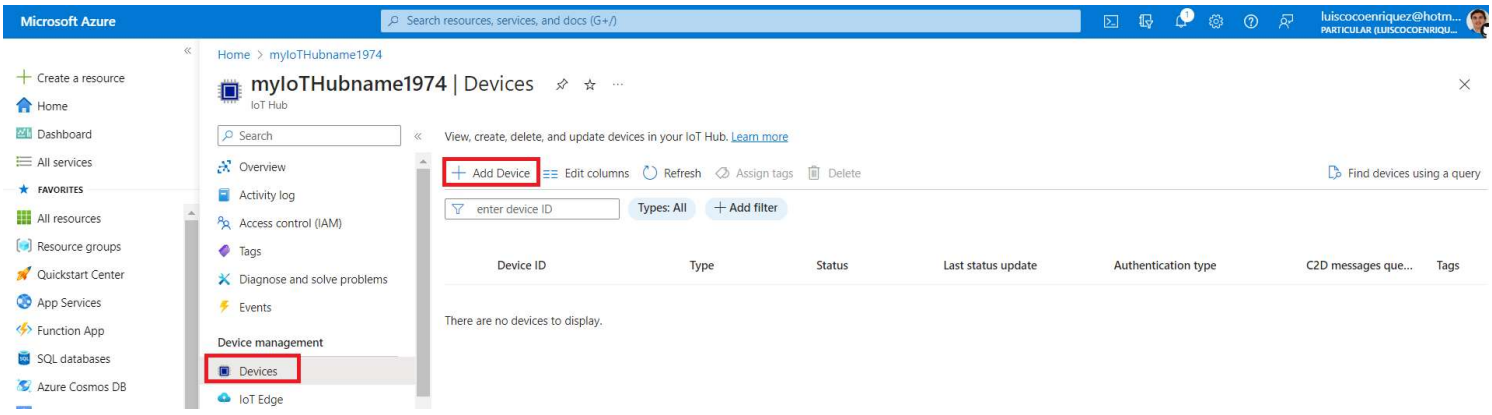

Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Microsoft Entra ID
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management + Billing
Home > IoT Hub >
IoT hub
Microsoft
Basics
Networking
Management
Add-ons
Tags
Review + create
Role-based access control
Change the permission model to Azure role-based access control (RBAC) only, or to a combination of shared access policies and RBAC. Learn more
Permission model
RBAC only
Shared access policy + RBAC
To manage the elements within an instance, a user needs access to IoT Hub data APIs. Select the suggested role below to grant yourself full access to the APIs. You can also use Access Control (IAM) to choose appropriate roles later.
Learn more
Assign me
IoT Hub Data Contributor role
Scale
Device-to-cloud partitions *
2
Preview mode
Turning preview mode on means this IoT hub can't be used for production environments. Some existing features may not work. This setting can't be changed later. Learn more
Try new features like MQTT 5, TLS maximum fragment length negotiation, and ECC server cert. Learn more
Preview mode
On
Off
Review + create
< Previous: Networking
Next: Add-ons >

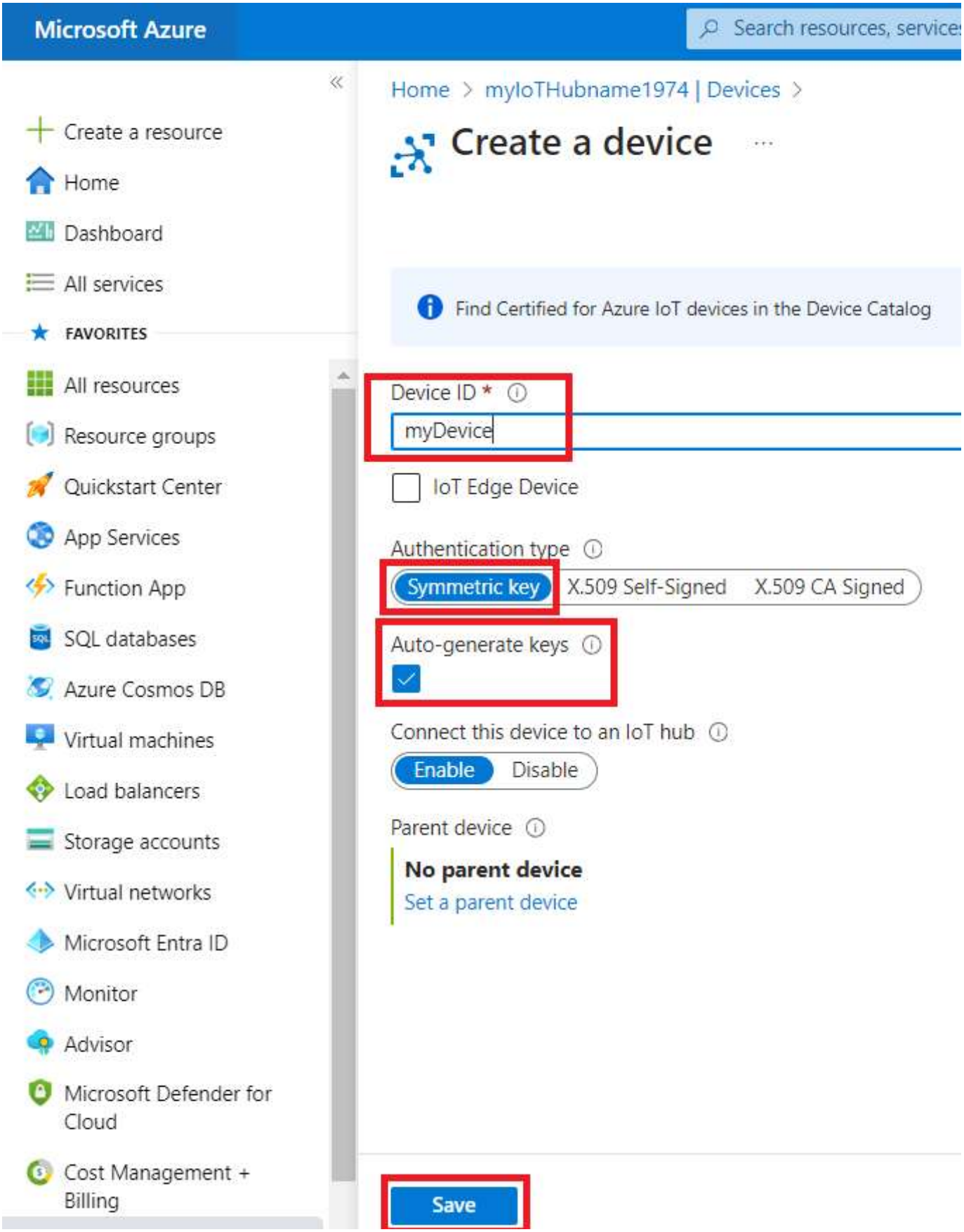

Microsoft Azure
Search resources, services, and docs (G+/)
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Microsoft Entra ID
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management + Billing
Home > IoT Hub >
IoT hub
Microsoft
Basics
Networking
Management
Add-ons
Tags
Review + create
Pricing
IoT hub
$0 USD
per month
Change basics
Add-ons total
Change add-ons
Basics
Subscription
Azure subscription 1
Resource group
myRG
IoT hub name
myIoTHubname1974
Region
West Europe
Disaster recovery enabled
Yes
Tier
Free
Daily message limit
8,000 ($0/month)
Networking
Connectivity configuration
Public access
Private endpoint connections
None
Allow public network access
Enabled
Create
< Previous: Tags
Next >
Automation options

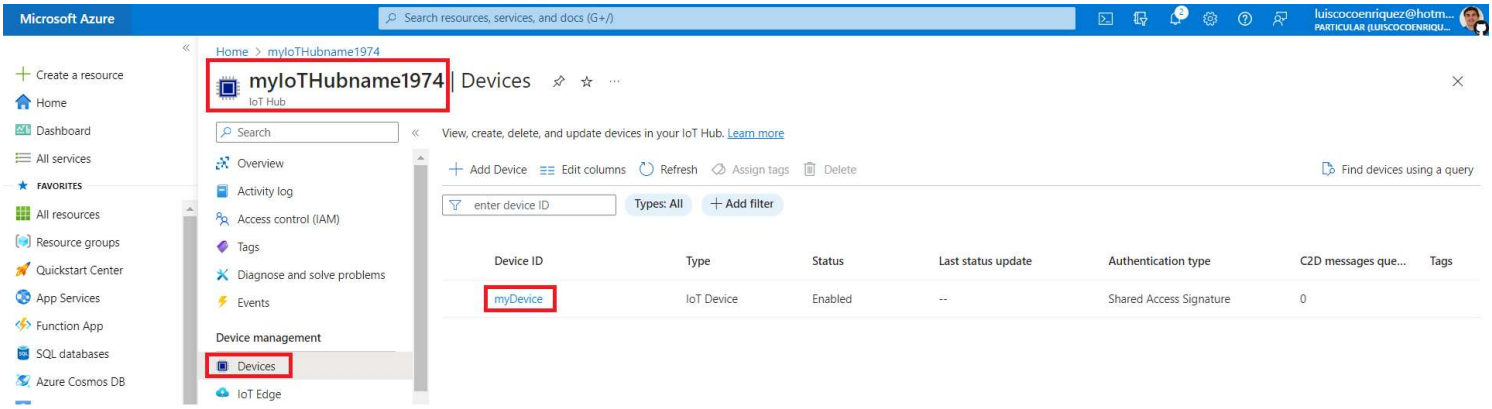




Create a Device inside the previouly created Azure IoTHub

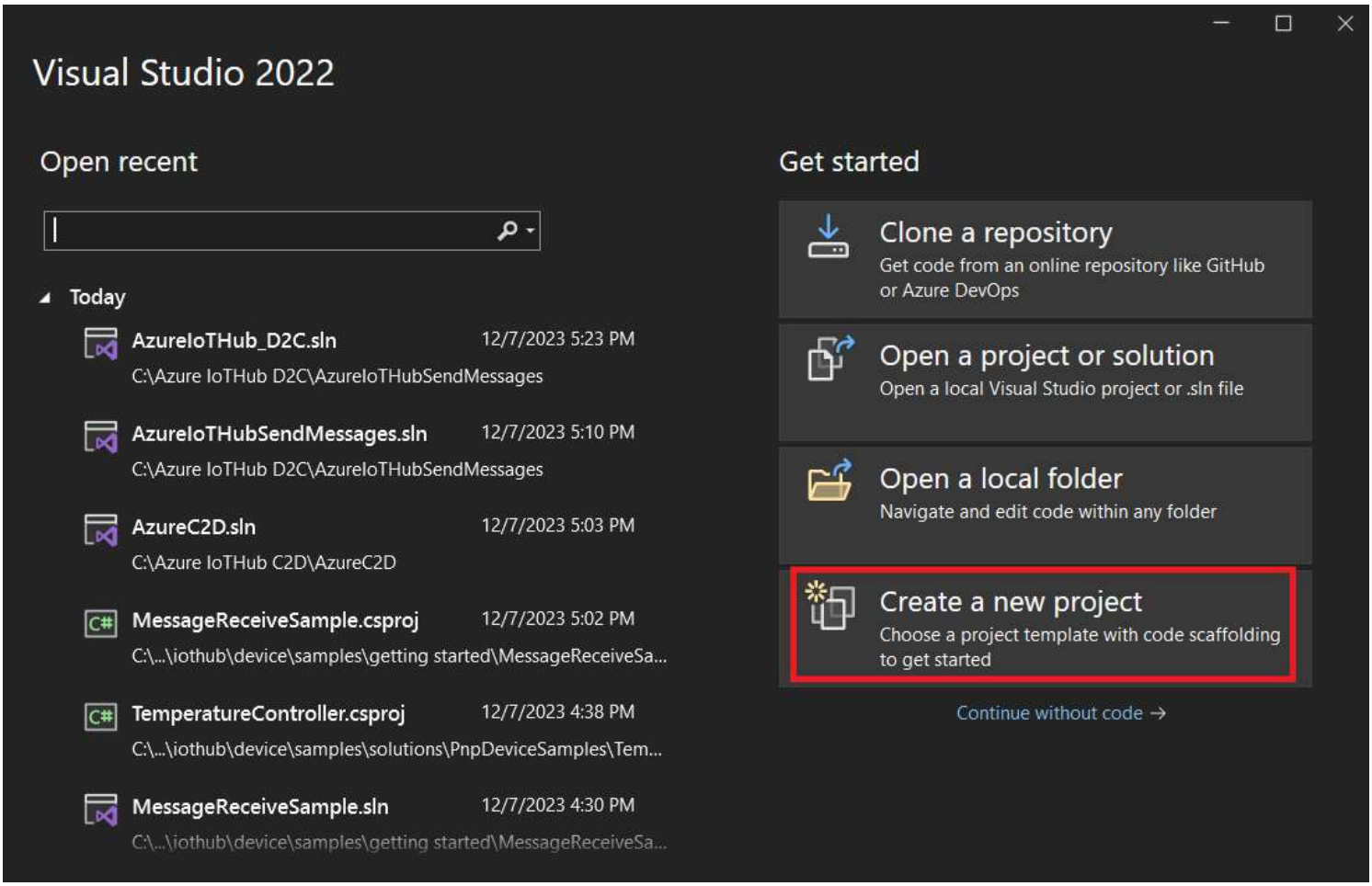


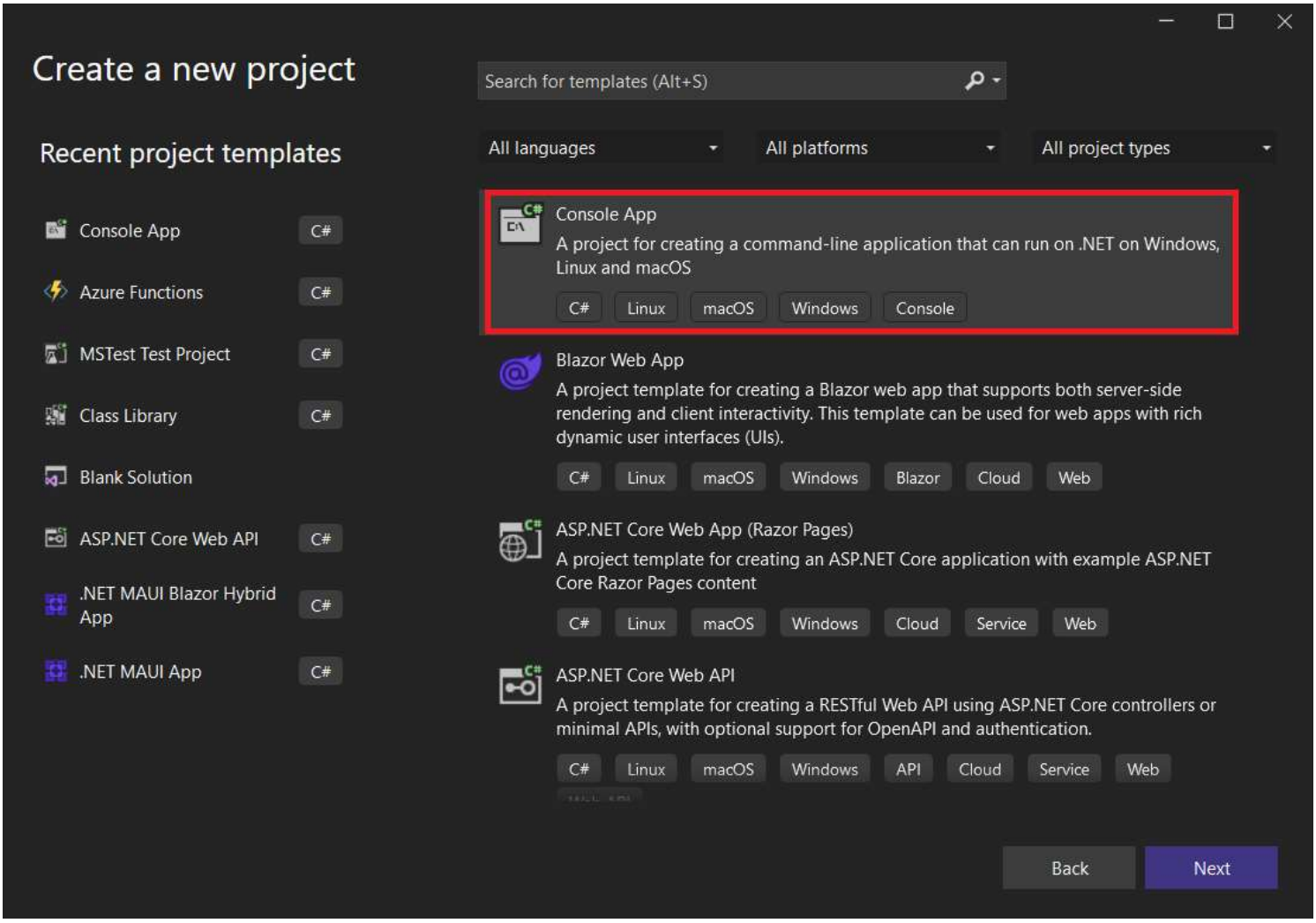

Microsoft Azure
Search resources, service
Create a resource
Home
Dashboard
All services
FAVORITES
All resources
Resource groups
Quickstart Center
App Services
Function App
SQL databases
Azure Cosmos DB
Virtual machines
Load balancers
Storage accounts
Virtual networks
Microsoft Entra ID
Monitor
Advisor
Microsoft Defender for Cloud
Cost Management + Billing
Home > myIoTHubname1974 | Devices >
Create a device
Find Certified for Azure IoT devices in the Device Catalog
Device ID *
myDevice
IoT Edge Device
Authentication type
Symmetric key
X.509 Self-Signed
X.509 CA Signed
Auto-generate keys
Connect this device to an IoT hub
Enable
Disable
Parent device
No parent device
Set a parent device
Save

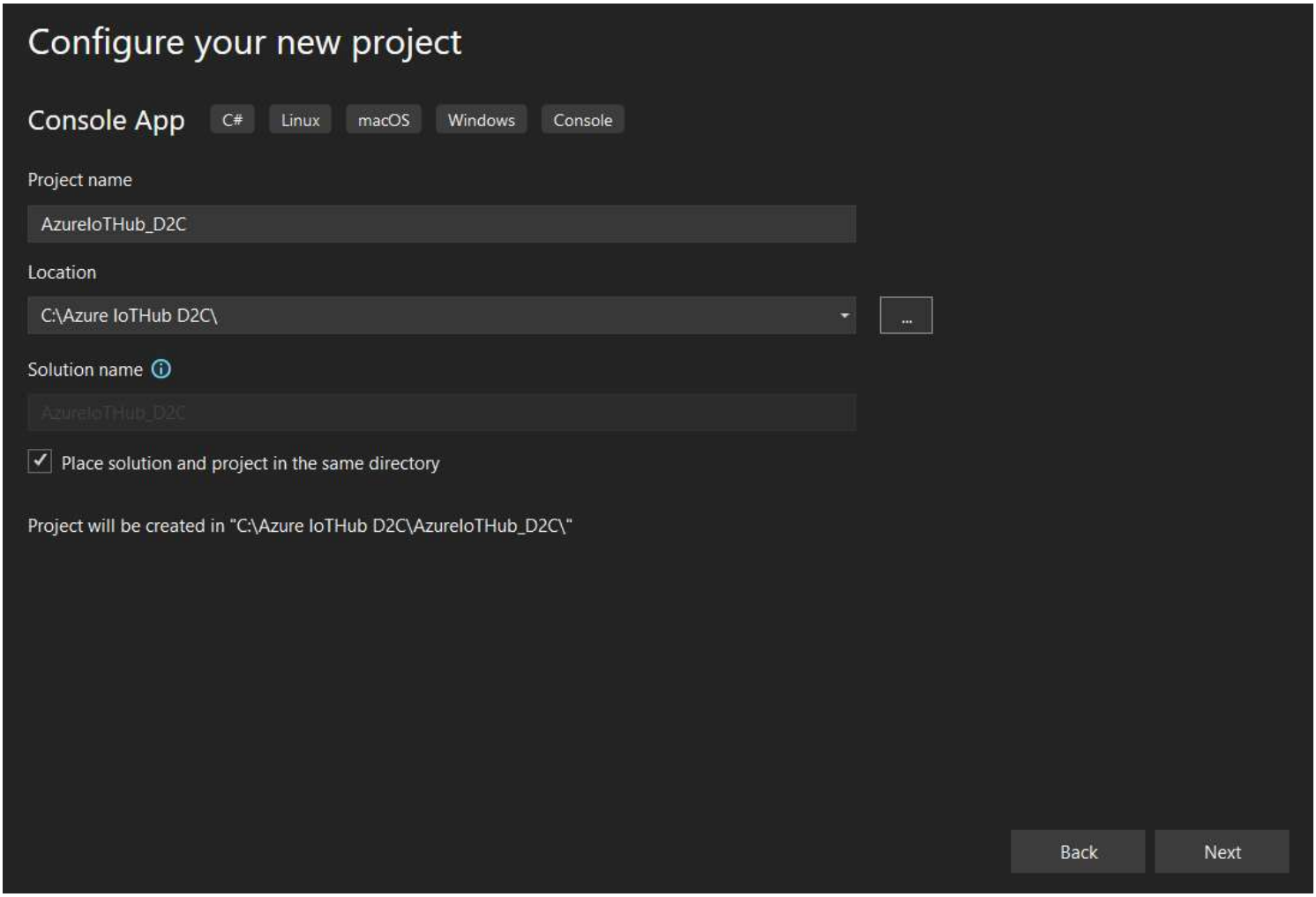


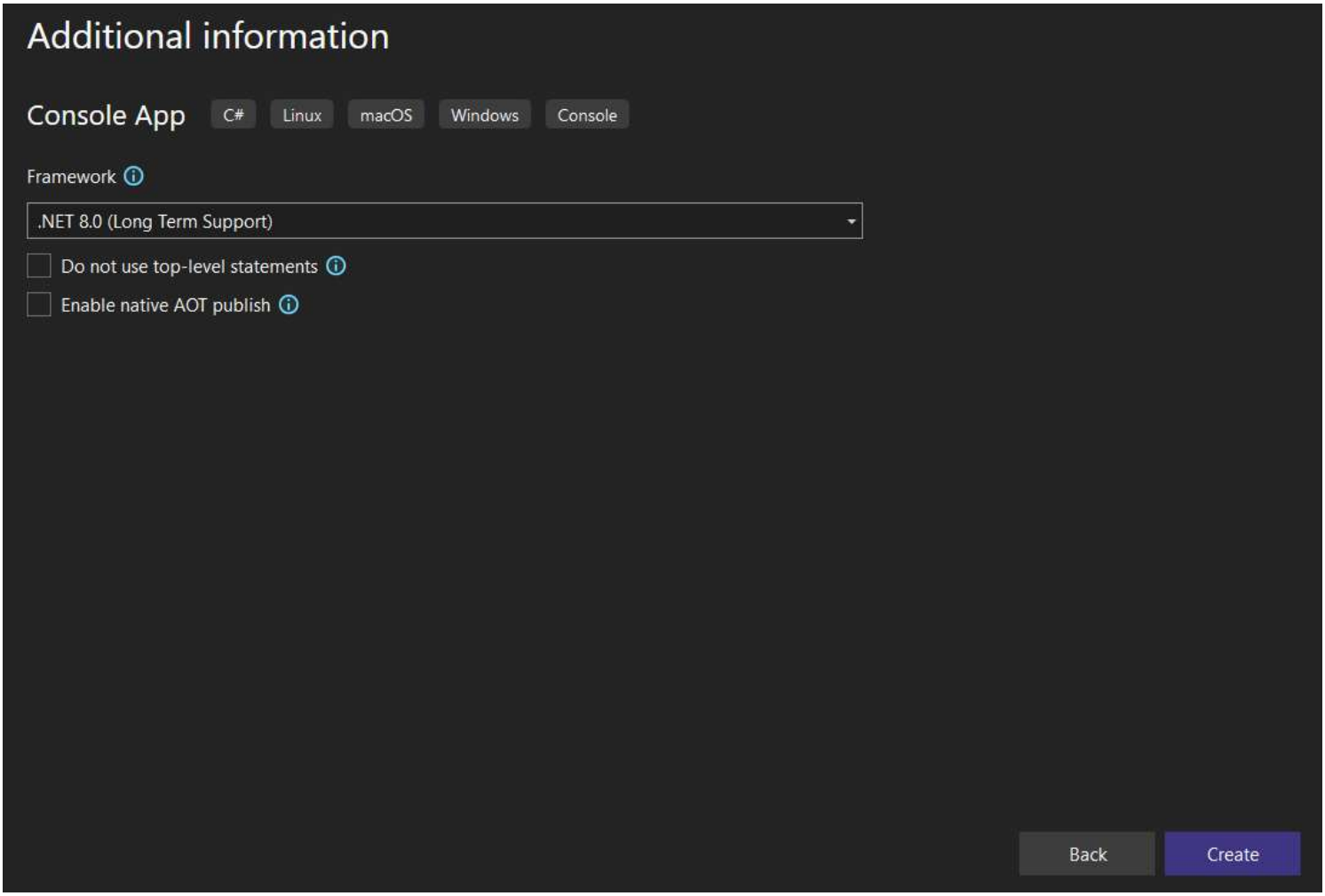
1. Create a .NET 8 C# console application in Visual Studio 2022 Community Edition

Run Visual Studio 2022 and follow these steps

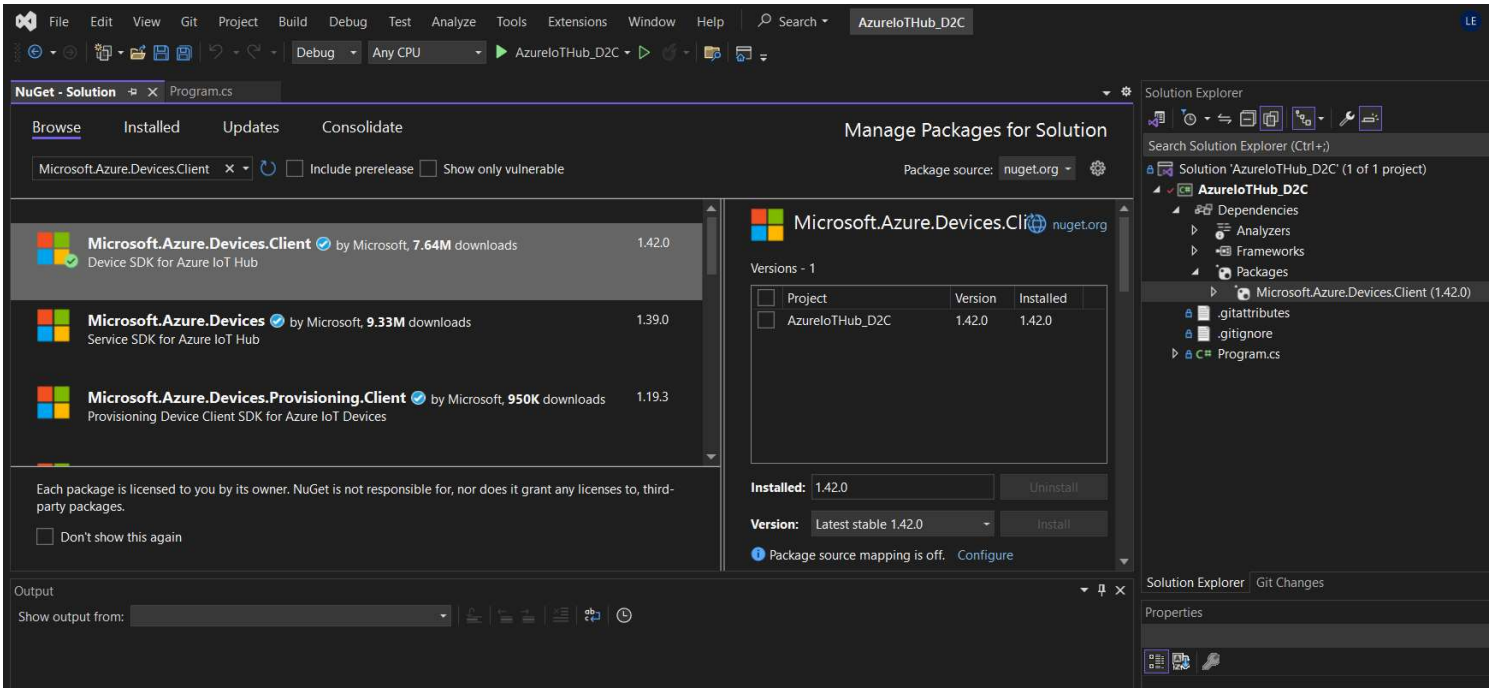



Create a new project
Search for templates (Alt+S)
Recent project templates
All languages
All platforms
All project types
Console App
C#
Azure Functions
C#
MSTest Test Project
C#
Class Library
C#
Blank Solution
ASP.NET Core Web API
C#
.NET MAUI Blazor Hybrid App
C#
.NET MAUI App
C#
Console App
A project for creating a command-line application that can run on .NET on Windows, Linux and macOS
C# Linux macOS Windows Console
Blazor Web App
A project template for creating a Blazor web app that supports both server-side rendering and client interactivity. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Blazor Cloud Web
ASP.NET Core Web App (Razor Pages)
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content
C# Linux macOS Windows Cloud Service Web
ASP.NET Core Web API
A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.
C# Linux macOS Windows API Cloud Service Web
Back
Next


Configure your new project
Console App
C#
Linux
macOS
Windows
Console
Project name
AzureIoTHub_D2C
Location
C:\Azure IoTHub D2C\
...
Solution name
AzureIoTHub_D2C
Place solution and project in the same directory
Project will be created in "C:\Azure IoTHub D2C\AzureIoTHub_D2C\"
Back
Next



Load with Nuget the library: **Microsoft.Azure.Devices.Client**

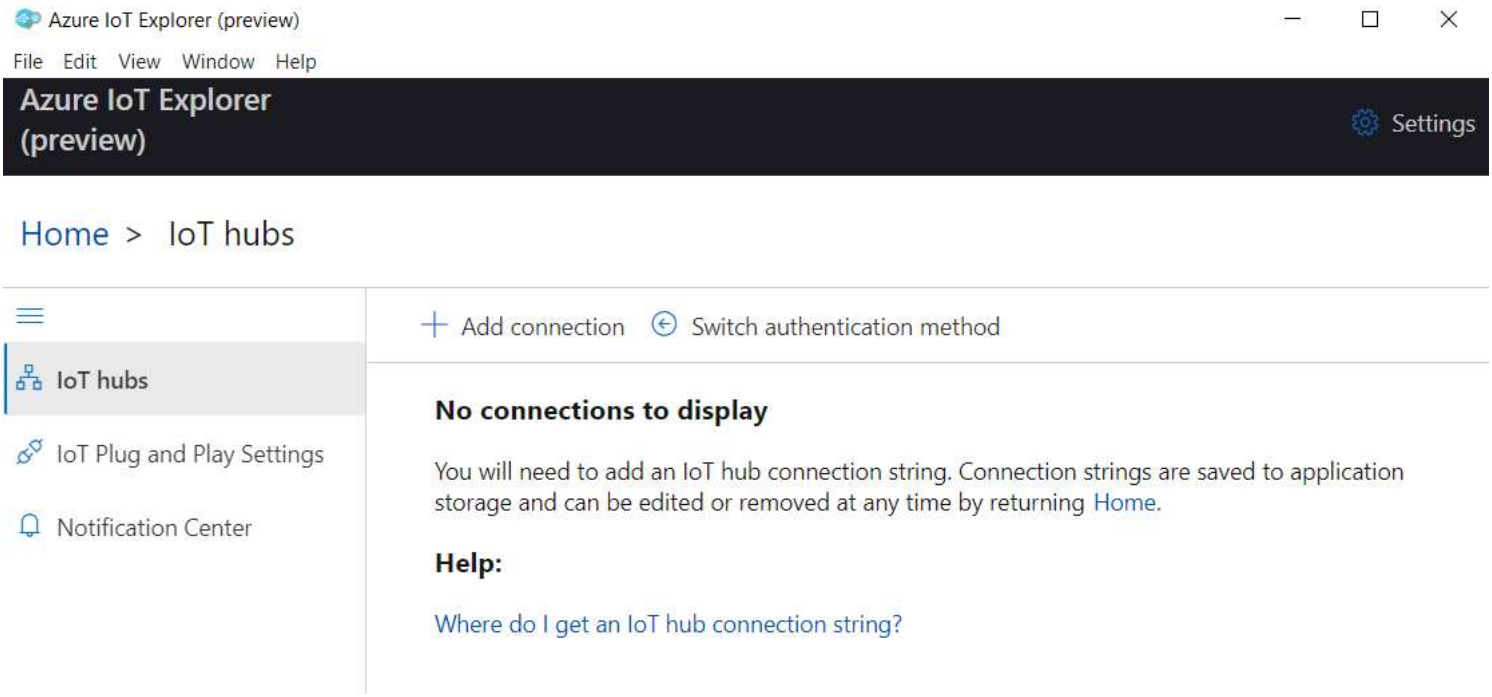


2. Input the application source code

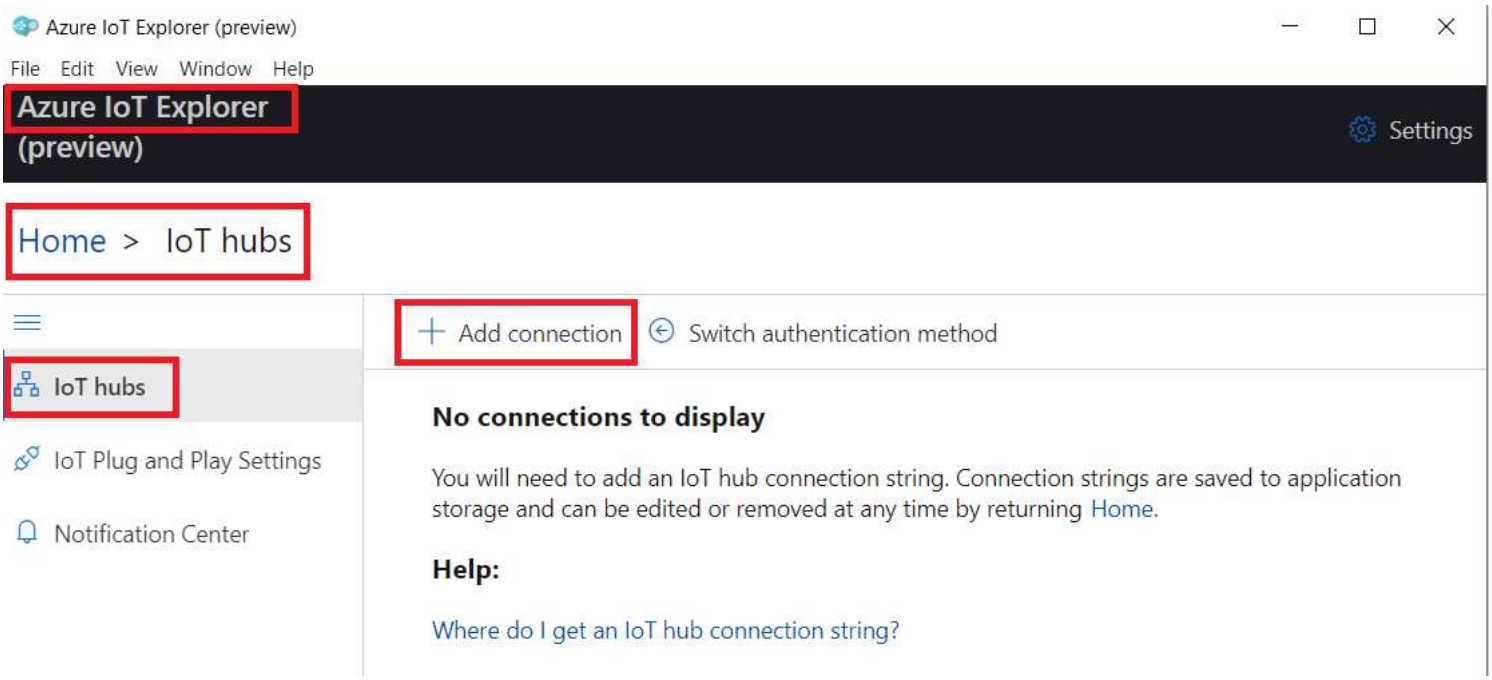
```
using Microsoft.Azure.Devices.Client;
using System.Text;

string ConnectionString = "HostName=myIoTHubname1974.azure-devices.net;DeviceId=myDevice;SharedAc

var message = "Hello from Azure IoT Hub!";
await SendMessageToIoTHubAsync(message);

async Task SendMessageToIoTHubAsync(string message)
{
    var deviceClient = DeviceClient.CreateFromConnectionString(ConnectionString, TransportType.Mq

    var payload = new Message(Encoding.UTF8.GetBytes(message));

    try
    {
        await deviceClient.SendEventAsync(payload);
        Console.WriteLine($"Message sent: {message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error sending message: {ex.Message}");
    }
    finally
    {
        await deviceClient.CloseAsync();
    }
}
```

3. Build and Run the application and Verify it with Azure.IoT.Explorer

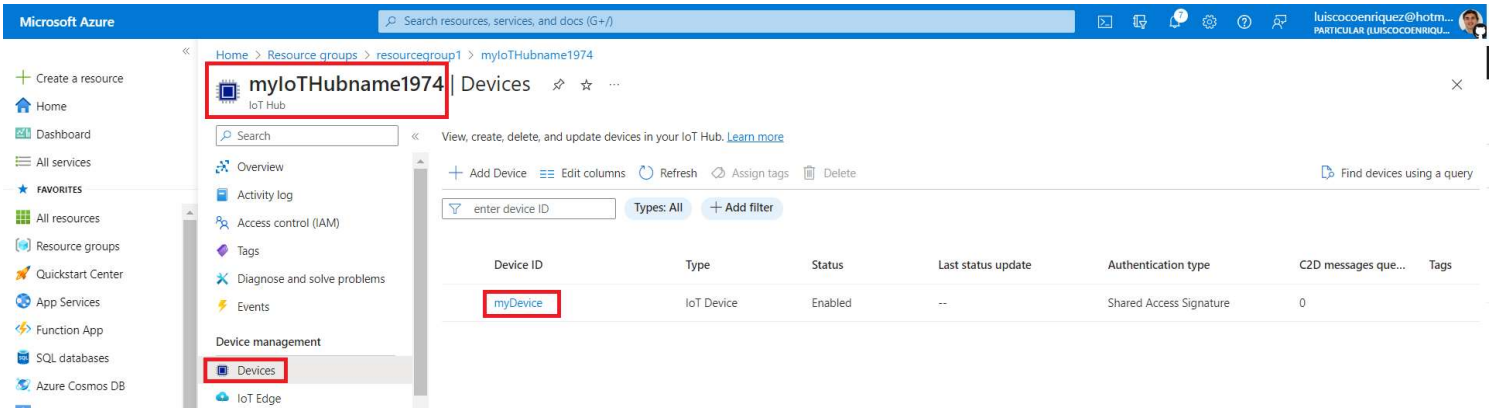
Donwload and install the Azure IoT Explorer: https://github.com/Azure/azure-iot-explorer

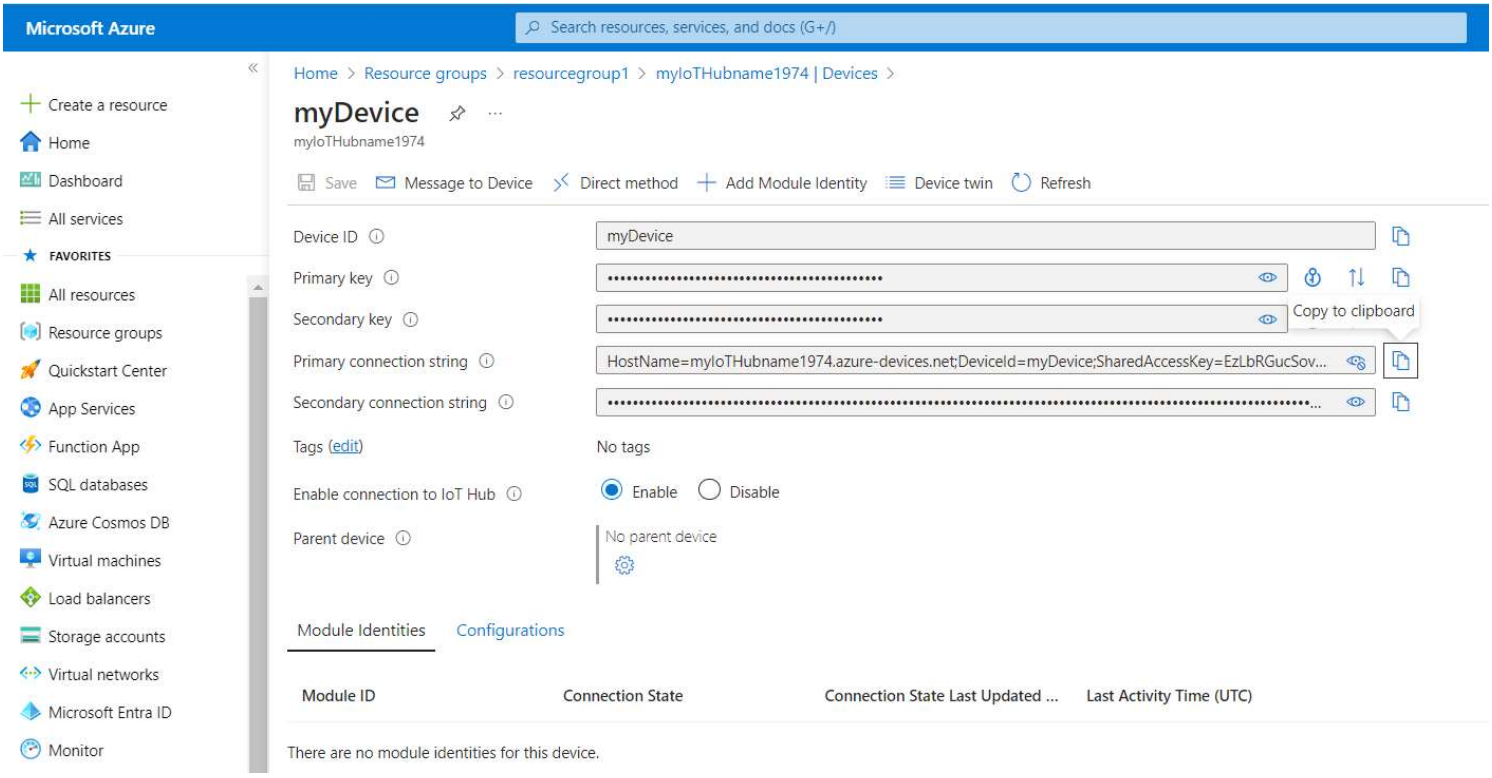


Add a new Azure IoTHub connection in the Azure IoT Explorer



Navigate to the Azure IoTHub and copy the connection string

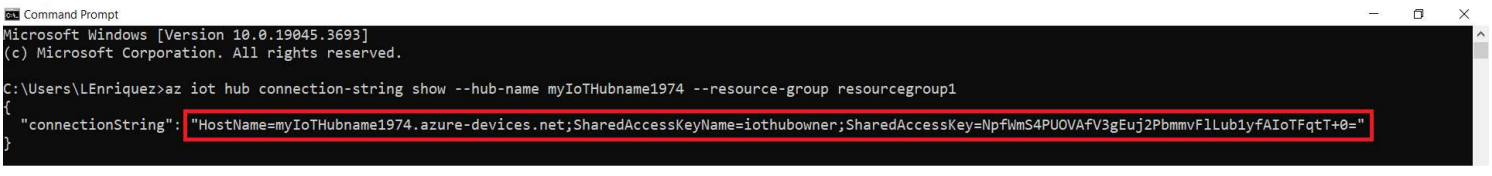




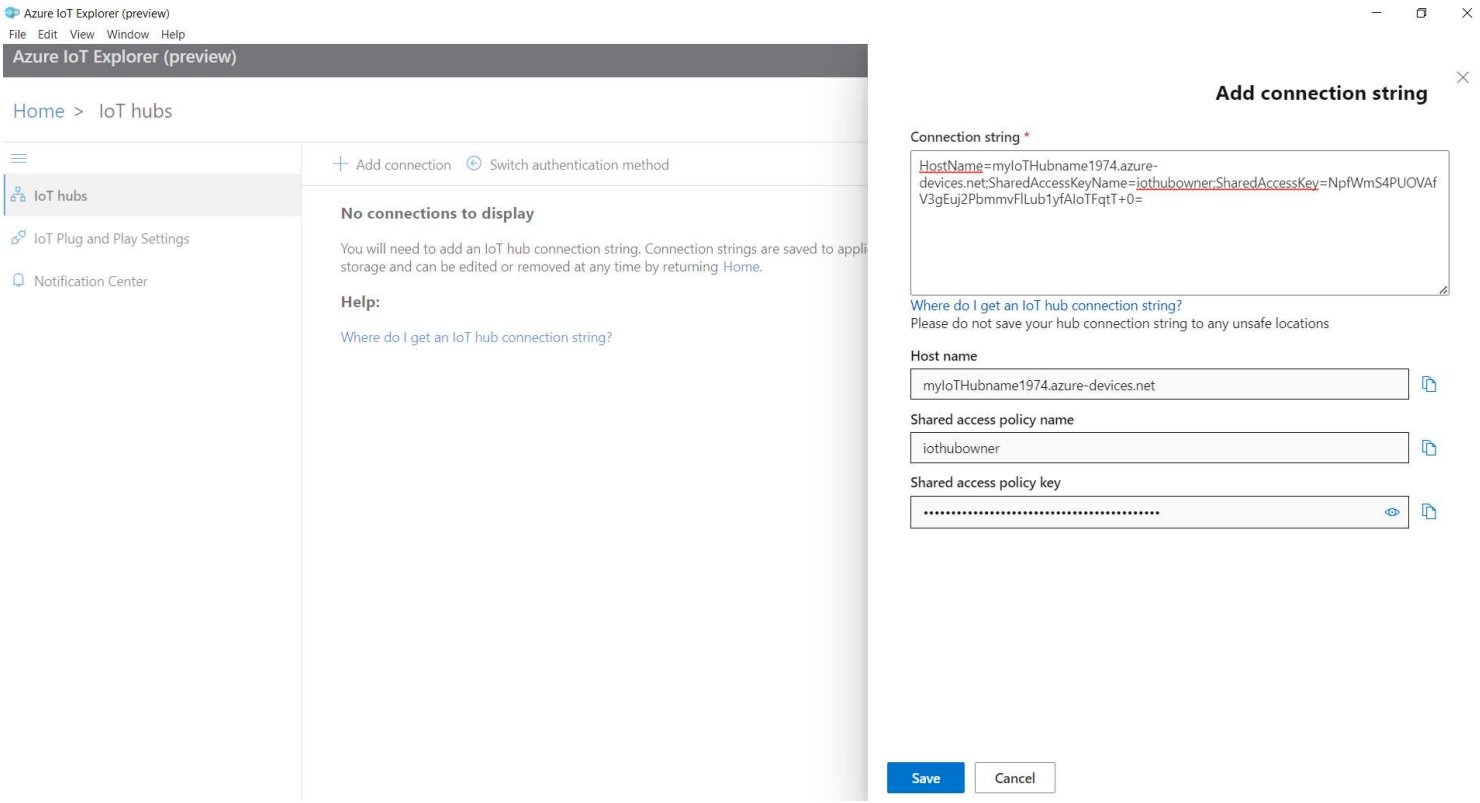
Or get the connection string running this command

```
az iot hub connection-string show --hub-name myIoTHubname1974 --resource-group resourcegroup1
```

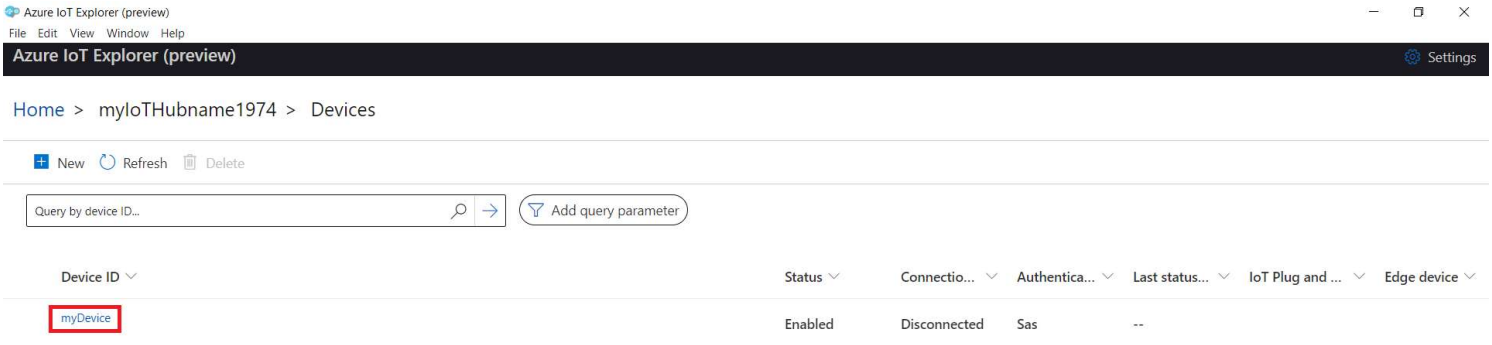
Do not forget to remove the quotes "" from the connection string



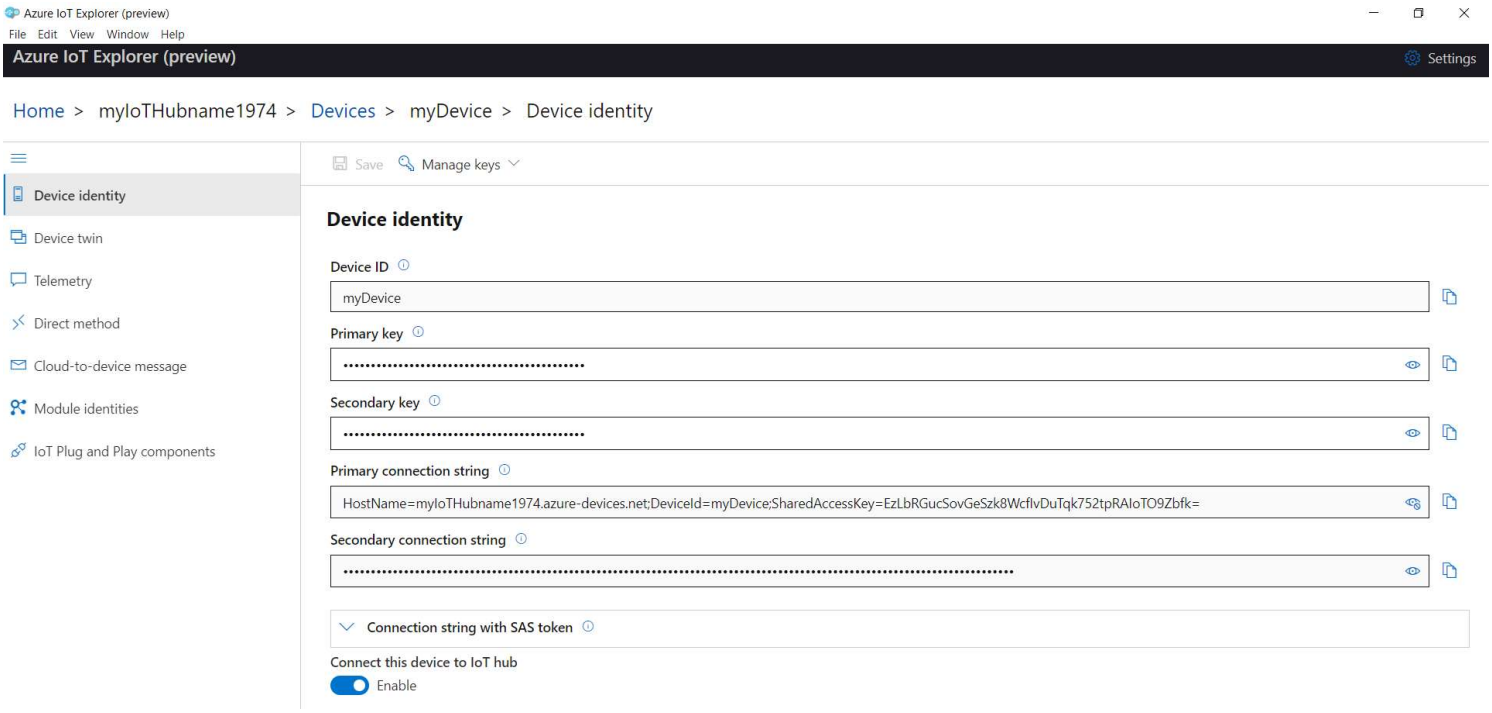
Add and configure a new connection to the Azure IoTHub



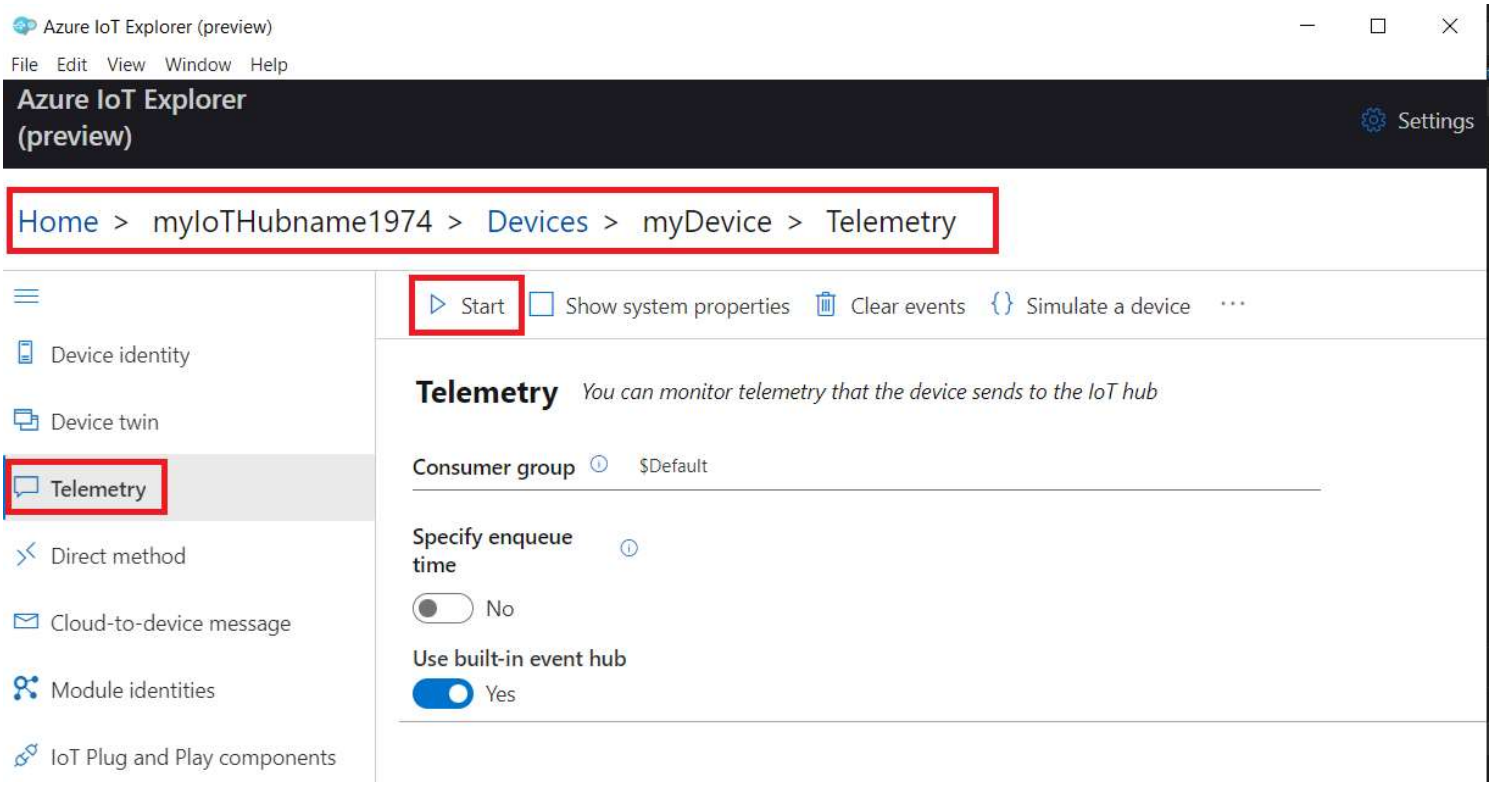
Save the connection and we can see out device inside out Azure IoTHub. Click on the Device Id

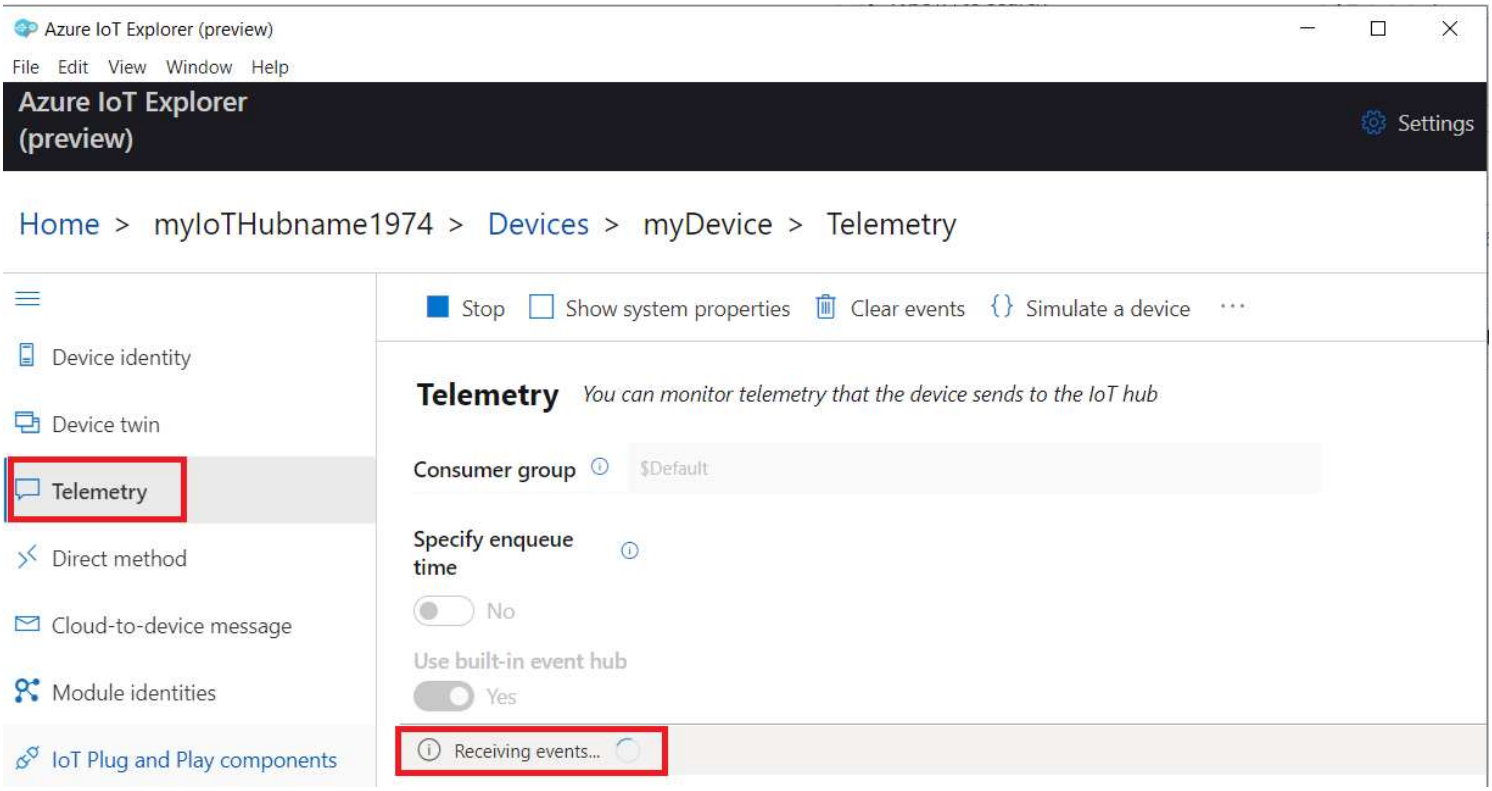


Now we can see the device option in the left menu



We are going to test sending a message from the device to the cloud (D2C), for this purpose we select the **Telemetry** option in the left menu.





It is the time to start the C# console application to send a message from our application(our simulated device) to the Azure IoTHub