Lab 1 - Partie 1

# Azure IoT Hub

L’IoT Hub va nous servir pour :

* Avoir La liste des « Devices » connectables de notre projet Eurek
* Recevoir les messages venants des « Devices »
* Récupérer ces messages cotés « Service » (Back-end) pour les traiter.

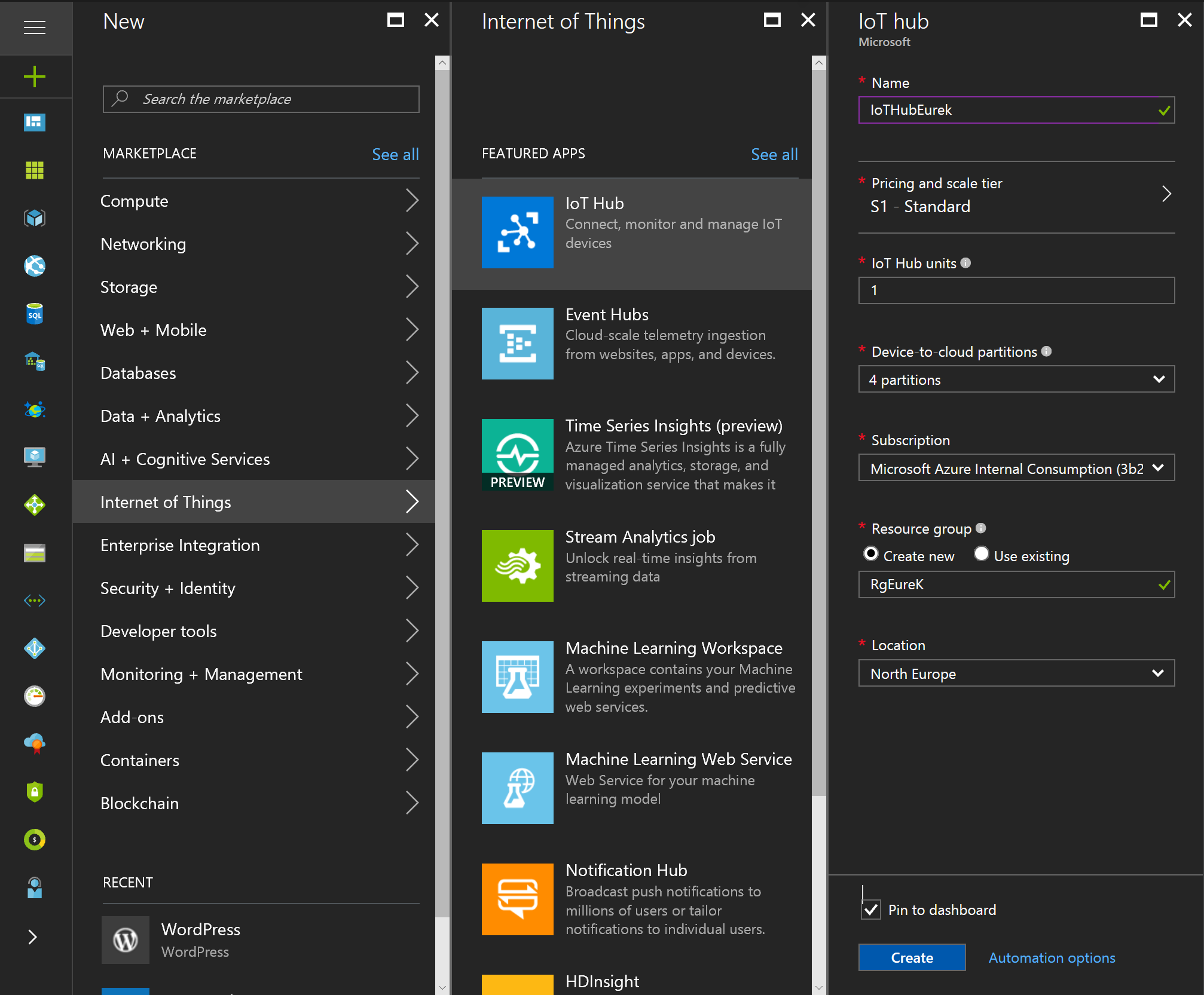
### Création de l’IoT Hub dans Azure

Dans le portail Azure :

***WARNING : Tous les noms de services et key utilisés dans les Lab (Lab1 – xxx) doivent être modifié pour votre propre usage.***

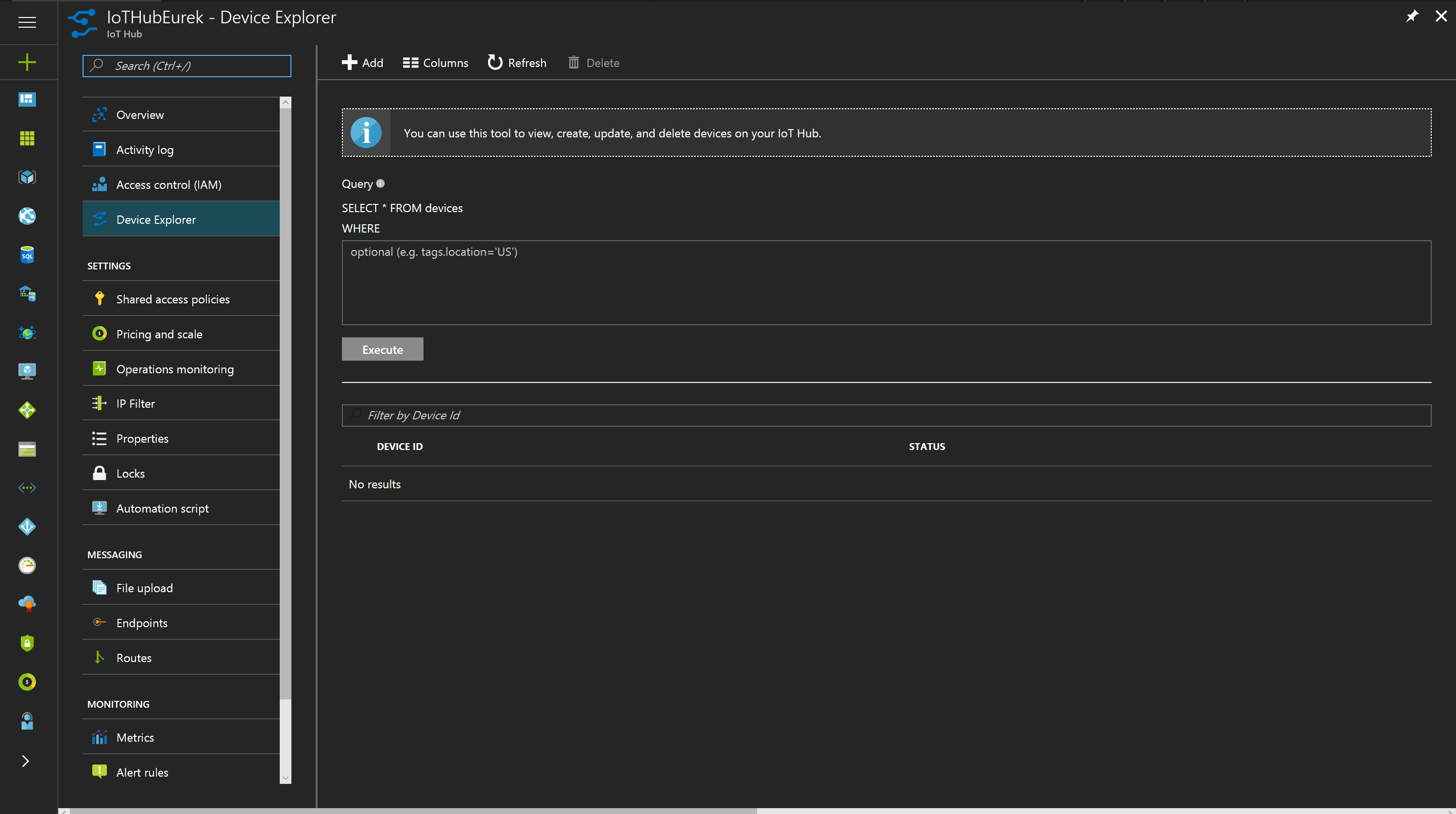
* Nom du Service : **IotHubEurek**
* Pricing and Scale Tier : **S1** (Free si vous en avez jamais utilisé)
* Ressource Group : **RgEurek** (ici je le crée puisque l’IoT Hub est ma première ressource) => *RG pour Ressource Group*
* Location : **North** **Europe** (mais à votre convenance).

Attendre 5 mn que le service soit provisionné.

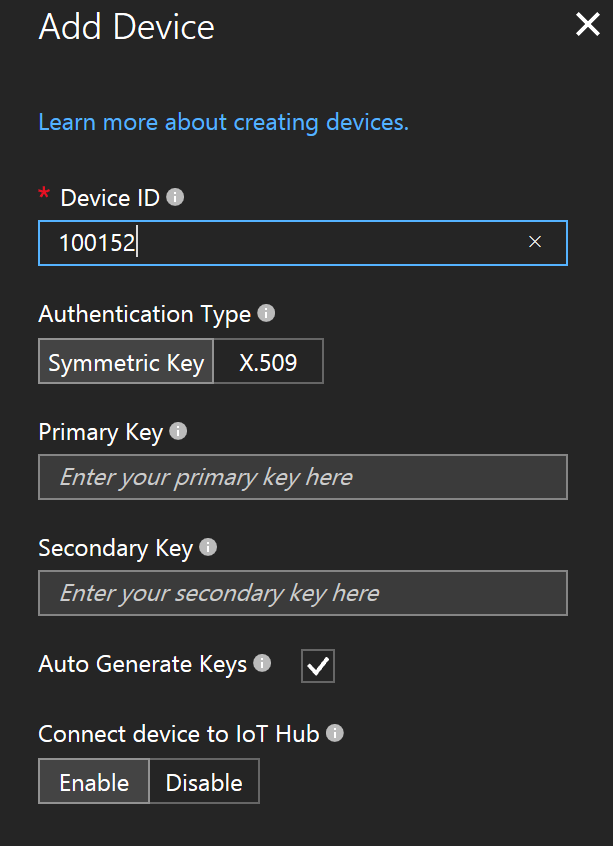


### Enregistrer un Device

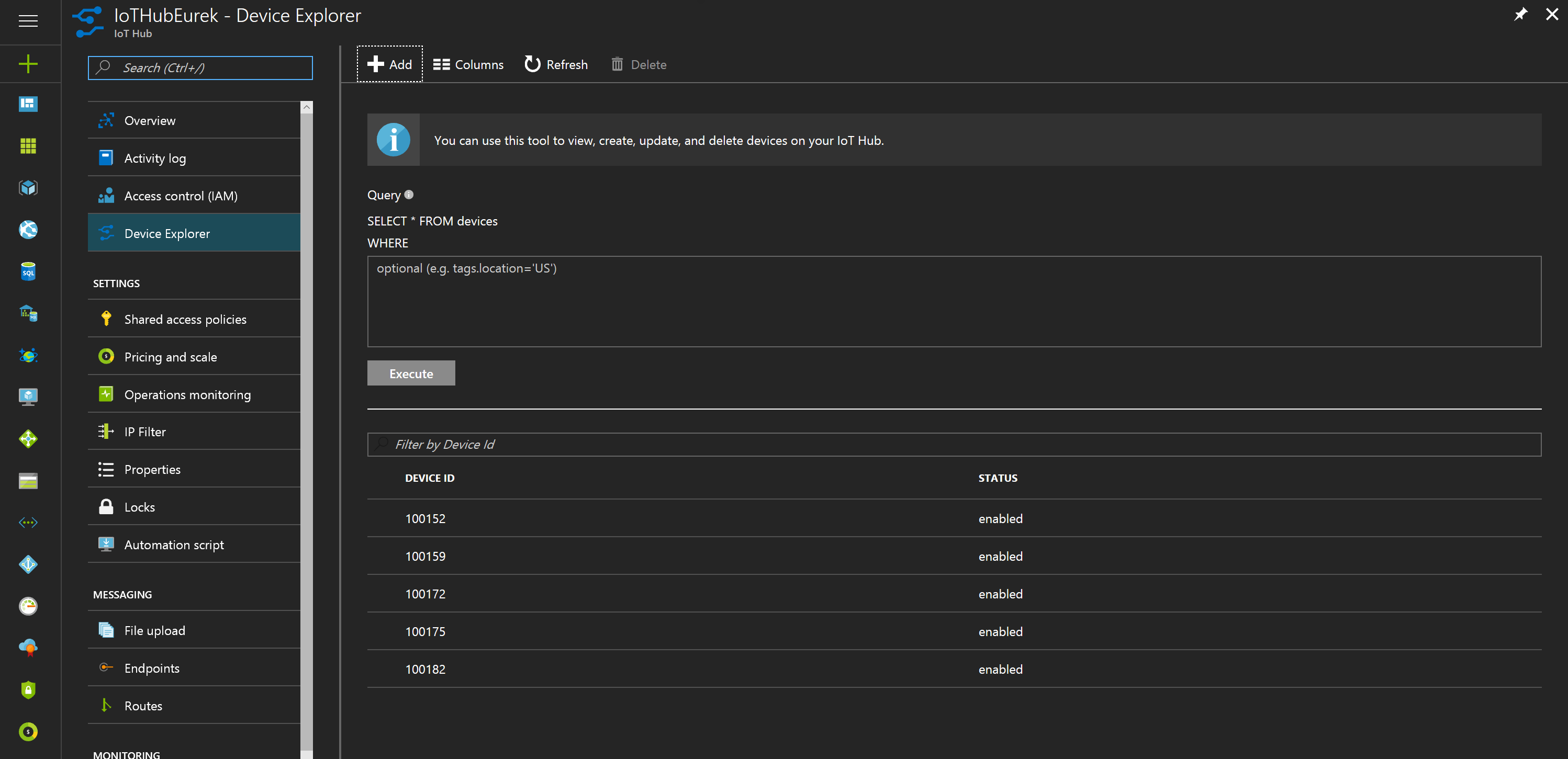
Dans Device Explorer, cliquez sur « + Add »



Device ID : 100152  
Nous allons laissez les autres options par défaut. Iot Hub va générer 2 clés que nous devrons utiliser dans le Device N° 100152 (Voir la partie Simulation et site Web)



Nous allons répéter l’opération pour 4 autres devices : **100159, 100172, 100175, 100182**



Ils seront utilisés dans le Site Web et dans l’application console.

Autres Tools :

* IoT Hub explorer : <https://github.com/Azure/iothub-explorer>
* DeviceExplorer : <https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer>
* IoTHub Diagnostics : <https://github.com/Azure/iothub-diagnostics>

Créer les devices par code : <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-csharp-csharp-getstarted>

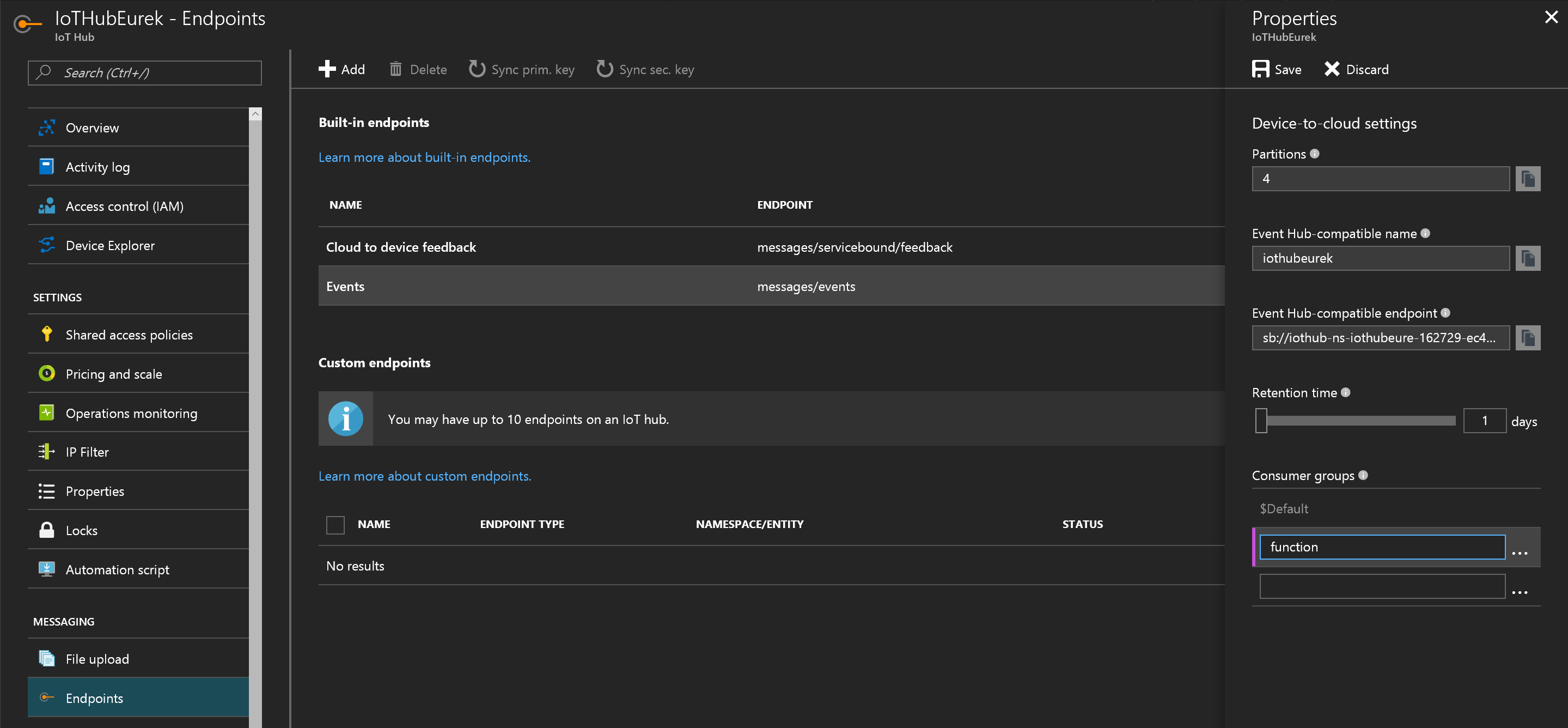
Extrait du code :

|  |
| --- |
| using Microsoft.Azure.Devices; using Microsoft.Azure.Devices.Common.Exceptions;  static RegistryManager registryManager;  static string connectionString = "{iot hub connection string}";  private static async Task AddDeviceAsync()  {  string deviceId = "myFirstDevice";  Device device;  try  {  device = await registryManager.AddDeviceAsync(new Device(deviceId));  }  catch (DeviceAlreadyExistsException)  {  device = await registryManager.GetDeviceAsync(deviceId);  }  Console.WriteLine("Generated device key: {0}", device.Authentication.SymmetricKey.PrimaryKey);  } |

### Création du Groupe de Consommation

Nous allons rajouter un groupe de consommation à notre « Endpoint » de messages venant des devices. Ceci permettra d’avoir plusieurs « Listeners » pour traiter les messages cotés « Service » (Backend).

Je vais le nommer «function »



Nous avons maintenant notre IoT Hub qui peut recevoir des messages de nos 5 devices et 2 groupe de consommation différents sur le même « endpoint ».

Ce lab est terminé, nous allons passer à la partie AppService Function pour héberger notre fonction pour simuler des appels à notre IotHub.

=> Fichier « Lab 1 – Partie 2 – FunctionApp.docx »