Lab 1 - Partie 2

# Azure Function App

Dans le lab précèdent, nous avons créé notre IotHub, enregistré 5 devices qui ont comme DeviceID : **100152, 100159, 100172, 100175, 100182.**

Dans ce lab nous allons créer une Function App sour forme d’API pour simuler l’envoi de messages à l’IoT hub. Elle sera ensuite utilisée dans notre application Web.

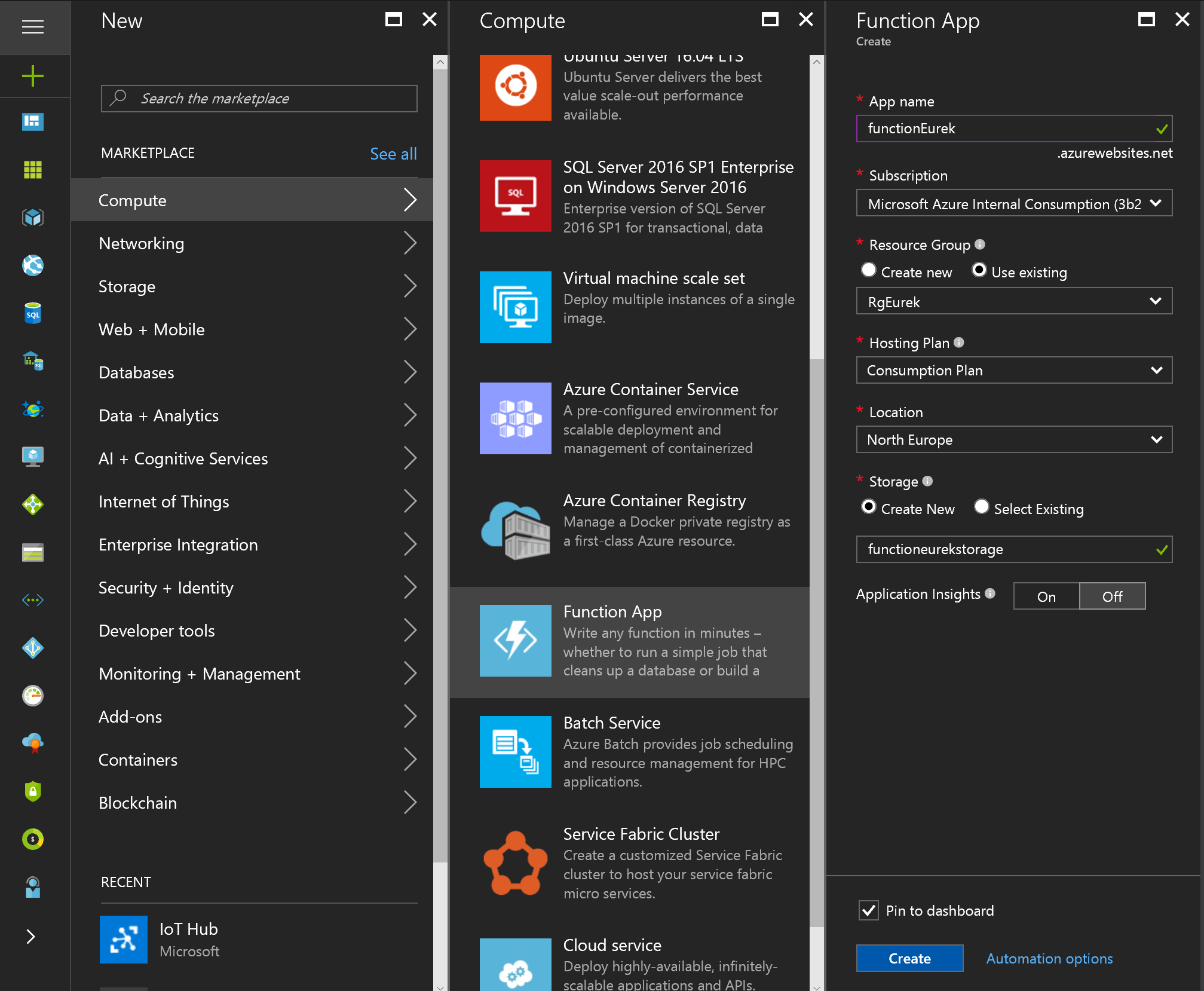
### Création de Function APP

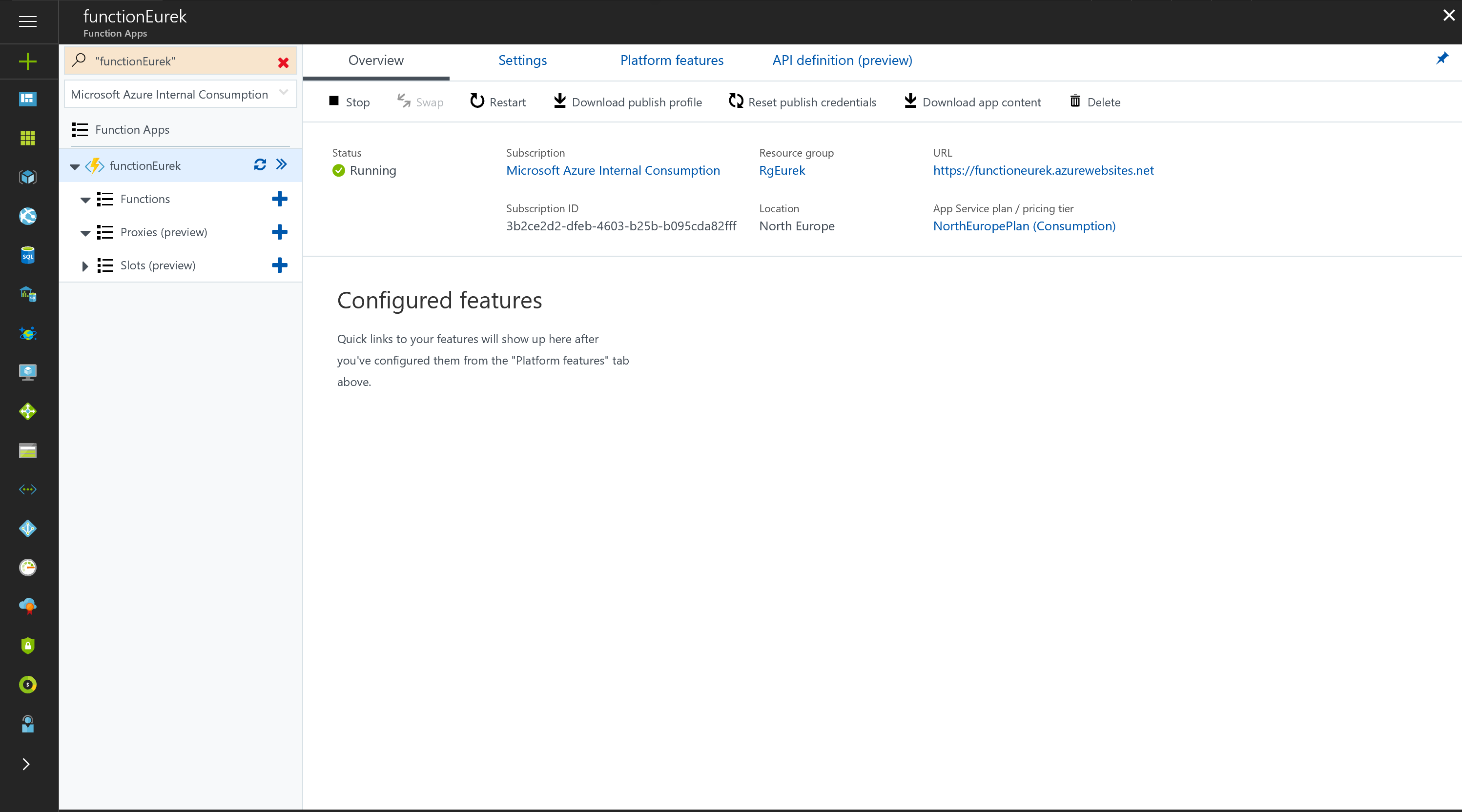
Dans le portail Azure

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

* Nom du service : functionEurek
* Ressource Group : “Use Existing » 🡪 RgEurek
* Hosting Plan : Consumption Plan (pour être facturé uniquement à l’usage)
* Location : North Europe (même que l’IoTHub)
* Storage : New 🡪 functioneurekstorage

Attendre 1 mn que le service soit provisionné.





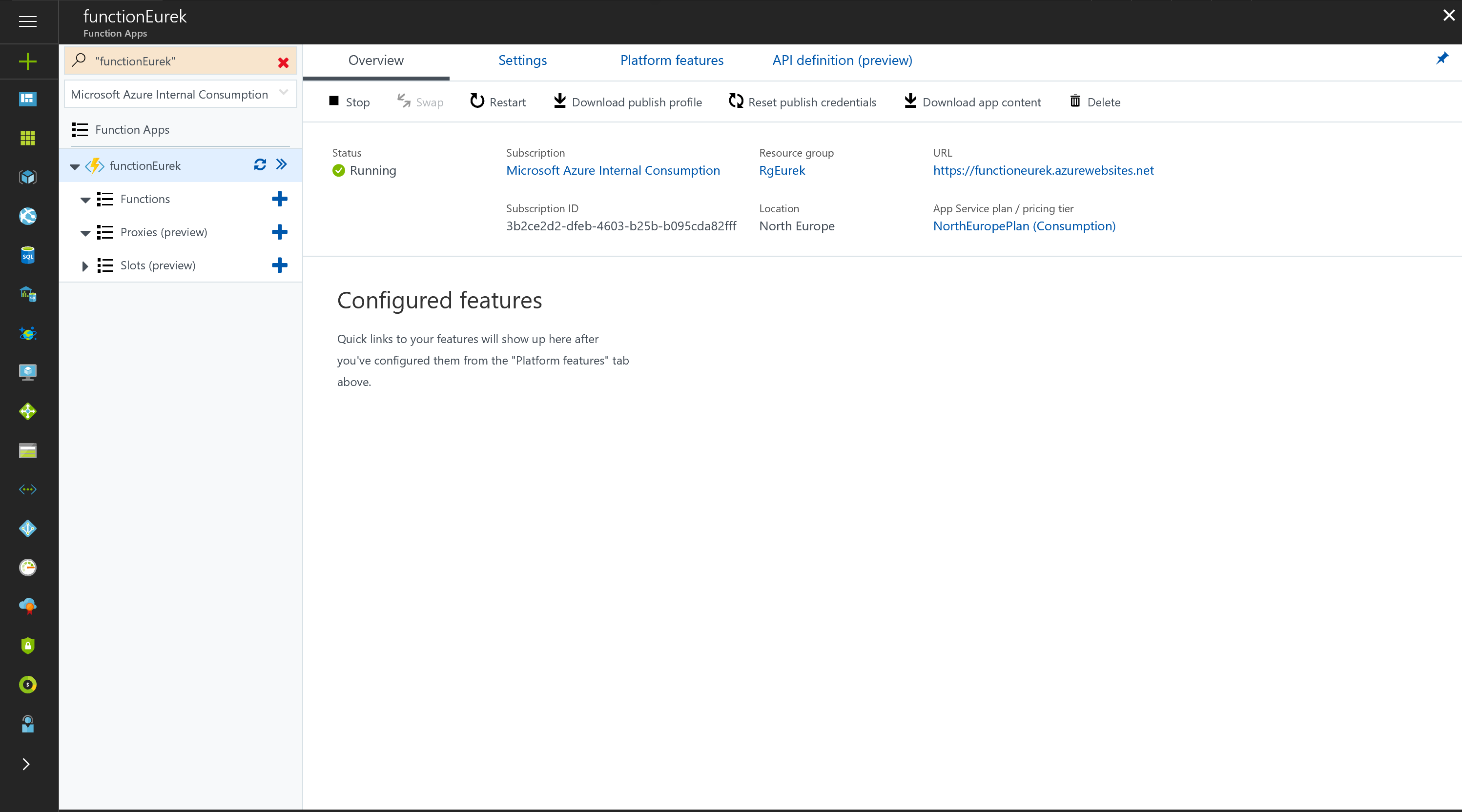
Donc on notera qu’ici ce service se nomme functionEurek et que pour l’instant aucune réelle fonction n’a été créé.

Dans ce lab, nous n’allons ni utiliser le tool pour Visual Studio 2017 (<https://blogs.msdn.microsoft.com/webdev/2017/05/10/azure-function-tools-for-visual-studio-2017/>), ni le déploiement continue.

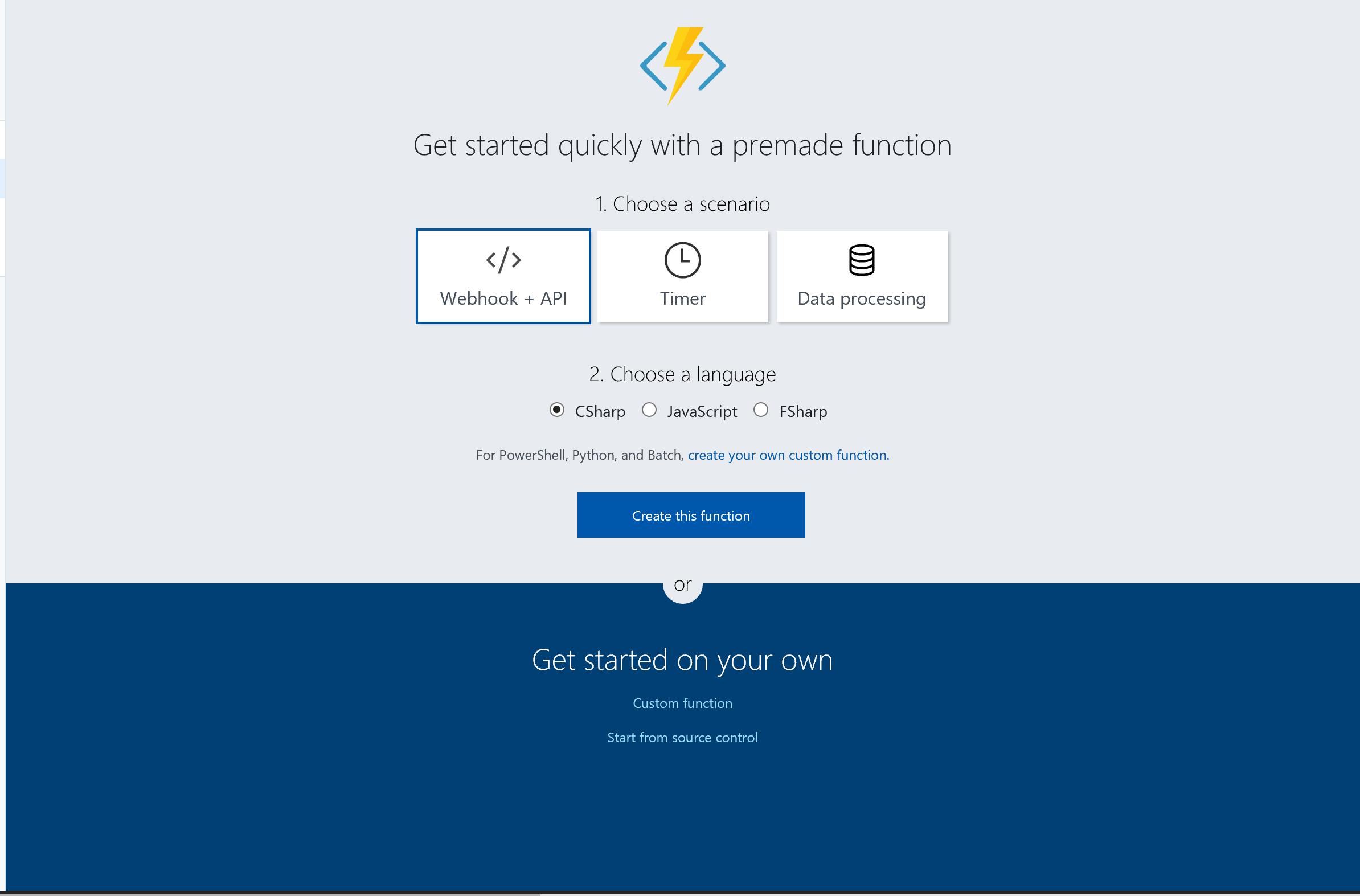
Nous allons créer une nouvelle fonction à l’intérieur de cette « enveloppe » functionEurek qui est en fait une AppService un peu particulière.

### Création de la fonction « API »

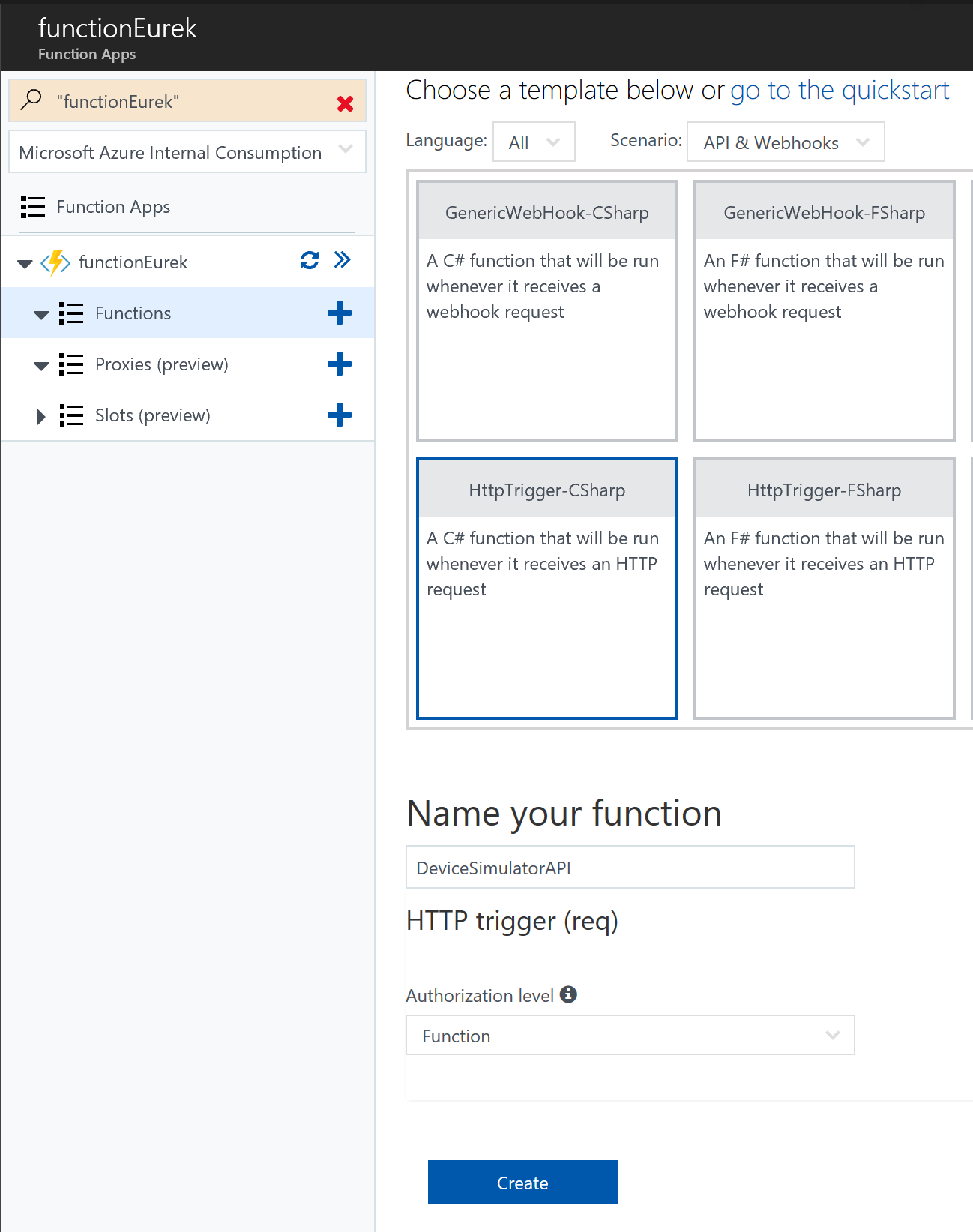
Chaque fonction se déclenche via un « Trigger » comme par exemple l’appel direct de la fonction via son URL, donc comme une API, en fait c’est une API.

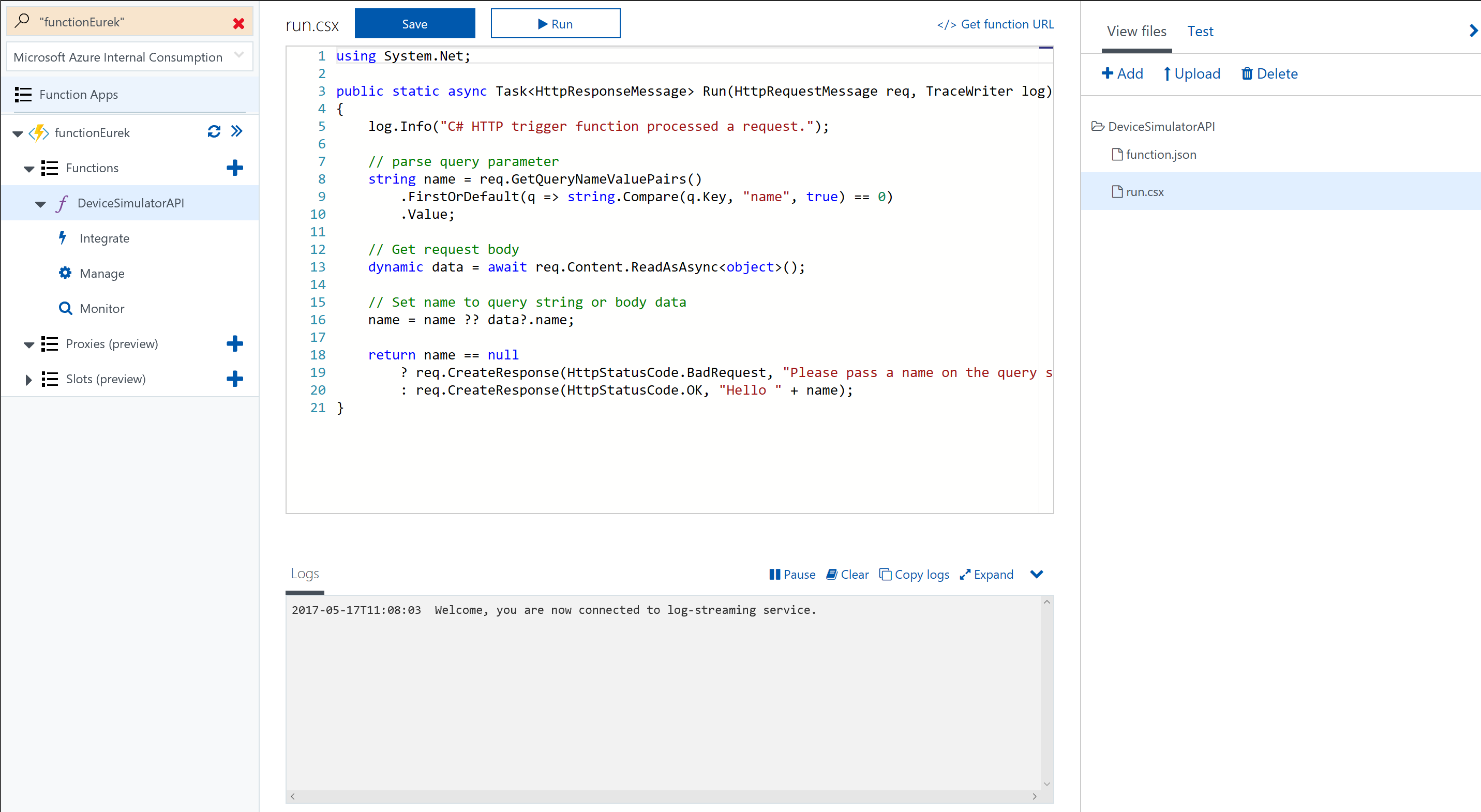


**Attention : Nous allons choisir « Custom function », sinon par defaut votre fonction aura un nom comme HttpTriggerCSharp1. On peut le changer mais cela demande quelques manipulations et donc pour simplifier on va choisir « Custom function »**



Donc notre fonction « réelle » se nomera « DeviceSimulatorAPI » et sera en C#.

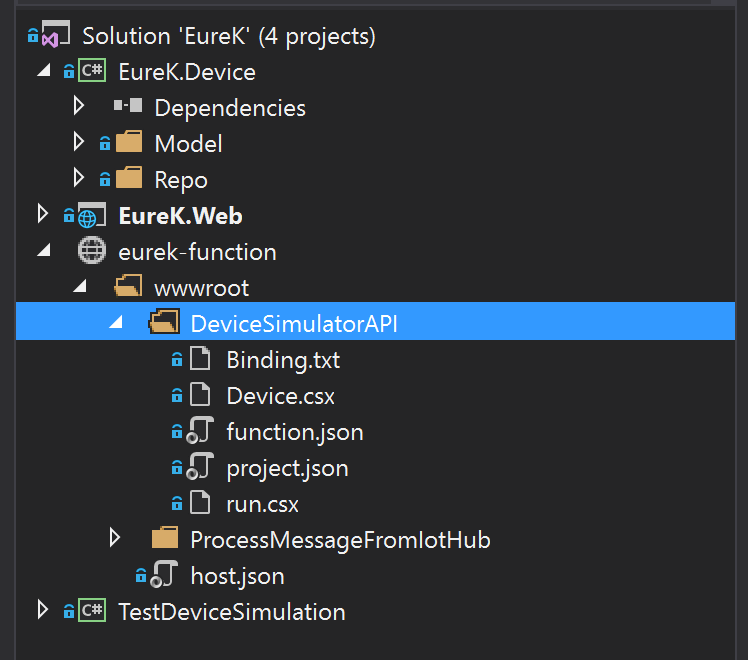




Nous allons maintenant créer de nouveaux fichiers, remplacer le code existant.

Vous trouverez les sources des fonctions sur les sources du Github et dans le dossier « [eurek-function/wwwroot](https://github.com/fwickert/Eurek/tree/master/Source/EureK/eurek-function/wwwroot" \o "This path skips through empty directories) ».

J’ai cloné le projet dans VS 2017 :

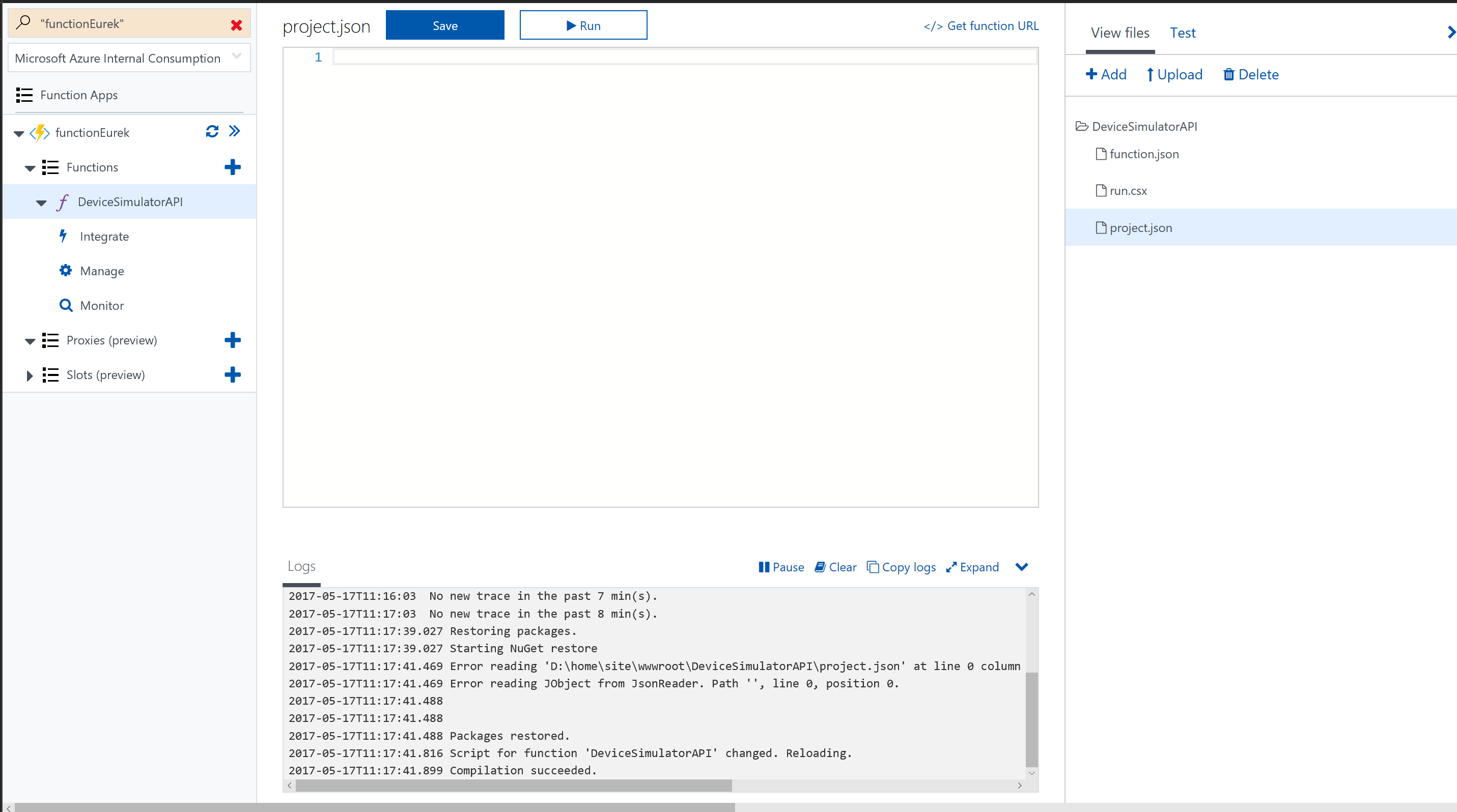


Pour les fonctions, voici la structure indispensable si vous souhaitez d’ailleurs faire du déploiement continue :

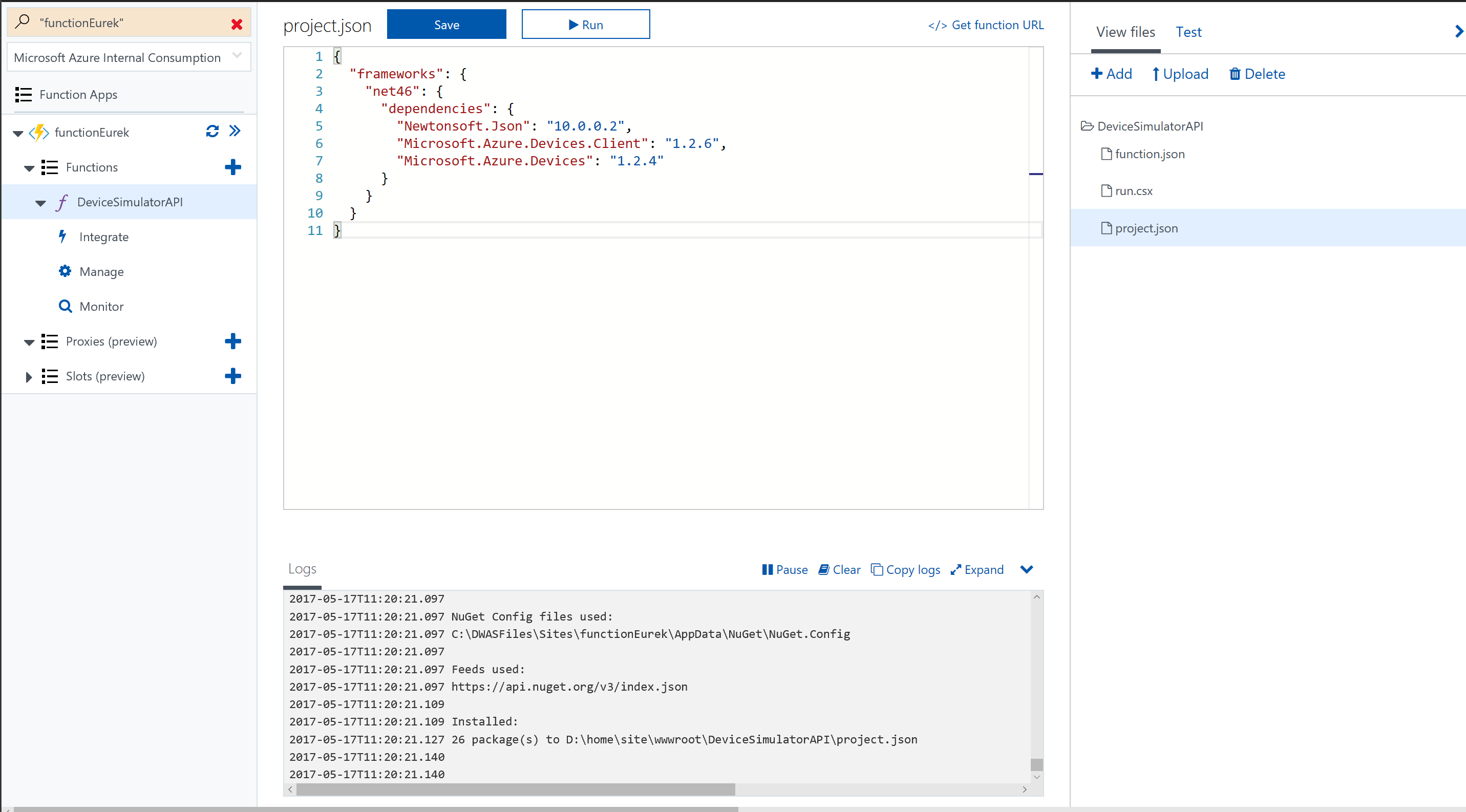
* Nom de l’enveloppe global : functionEurek (dans ce lab)
  + wwwroot
    - 1 Dossier par fonction (DeviceSimulator)
      * fichiers
    - ProcessMessageFromIotHub
  + host.json : Fichier obligatoire pour un déploiement continue mais vide (Nous ne l’utiliserons pas)

**Le fichier Binding.txt ne sert à rien ici (à l’origine pour un copier/coller)**

Commençons par créer le fichier « project.json » dans ma fonction dans Azure.

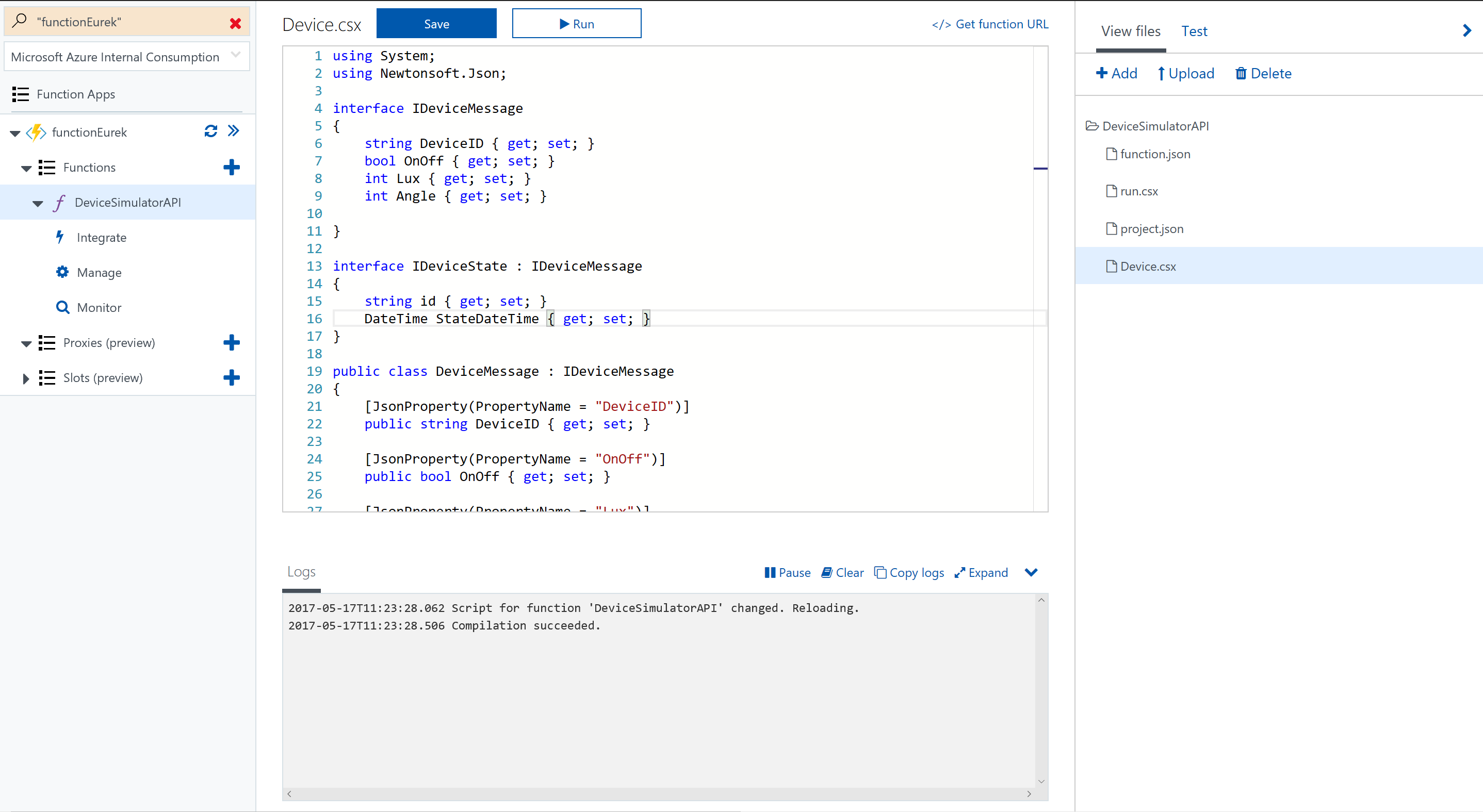


Et ensuite on va copier le contenue du fichier « project.json » du gitHub vers Azure.



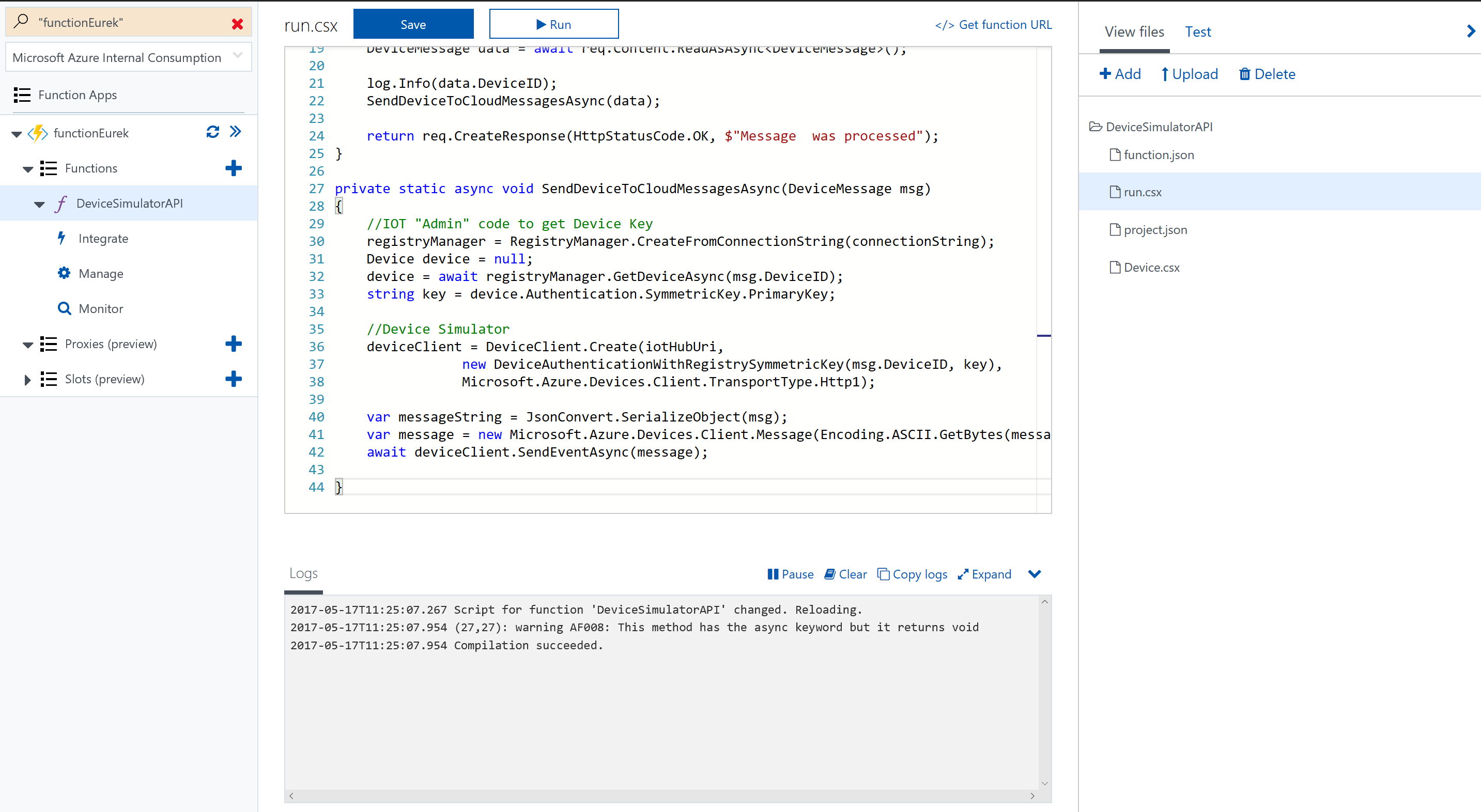
Apres un « save », vous allez voir dans les « logs » que les packages précisés dans ce fichier json vont étre installés.

Nous allons faire exactement la même chose avec le fichier « Device.csx »



Assurez-vous que votre fonction compile toujours.

Ensuite nous allons remplacer tout le code du fichier run.csx d’Azure avec celui du Github.



Assurez-vous que votre fonction compile toujours.

Il nous reste maintenant à modifier les clés pour accéder à l’IotHub que nous avons créé dans le lab précédent.

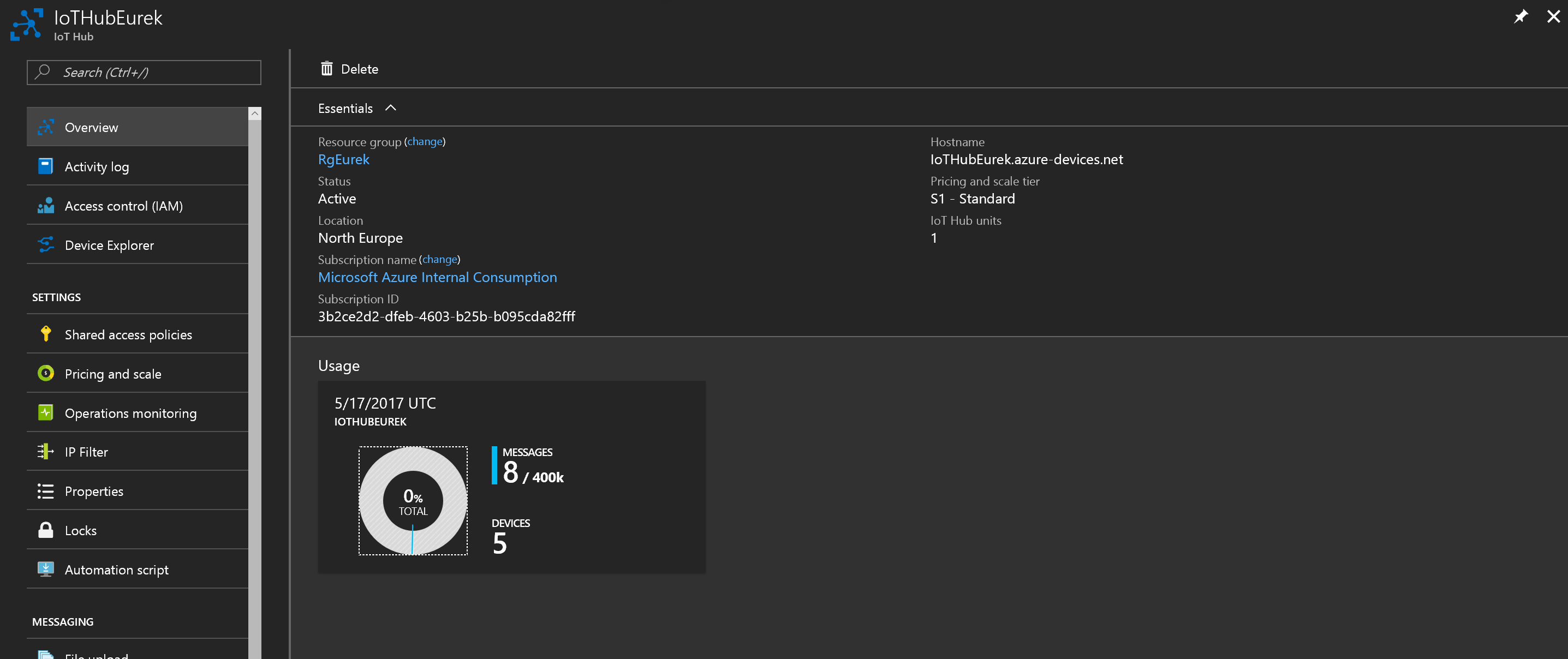
Il nous faut le nom de l’IotHub et la clé qui permettra de récupérer automatiquement la clé du device qui a appeler cette API.

|  |
| --- |
| private static string connectionString = "HostName=<Your IoTHub>.azure-devices.net;SharedAccessKeyName=registryReadWrite;SharedAccessKey=<Your SAK>"; private static RegistryManager registryManager; static DeviceClient deviceClient; static string iotHubUri = "<Your IoTHub>.azure-devices.net"; |

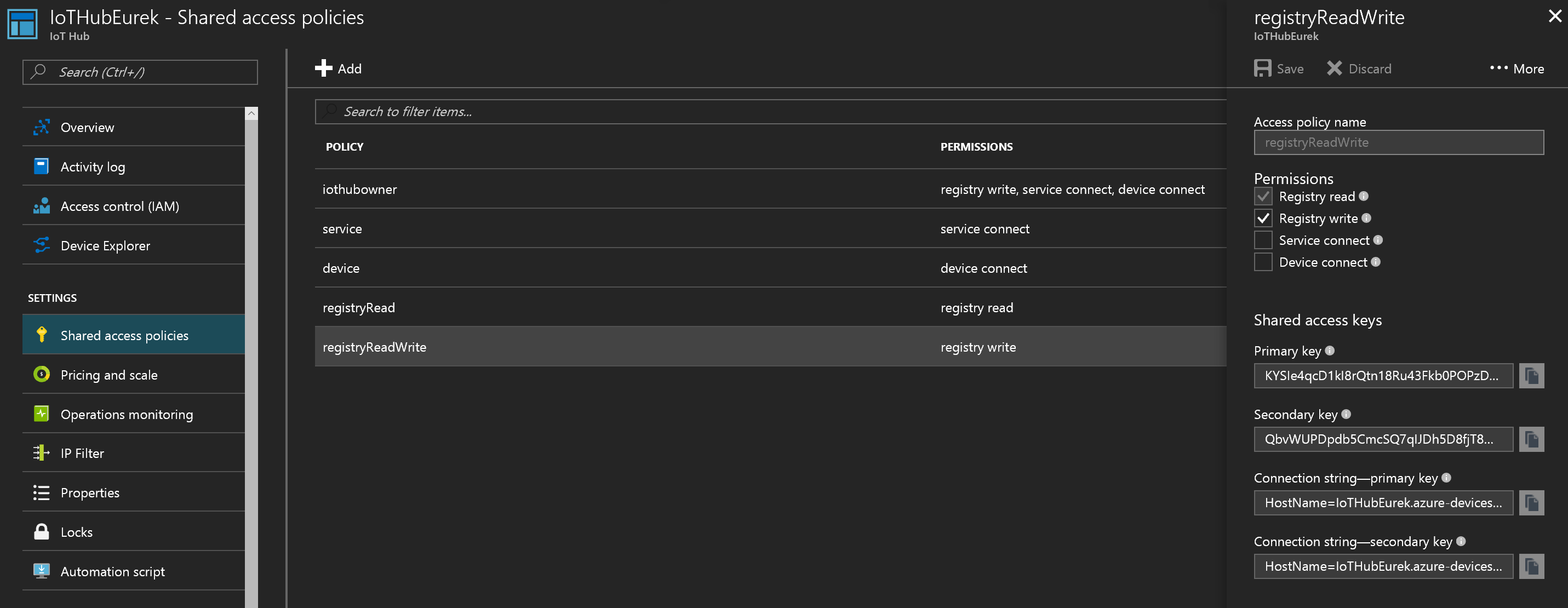
Où trouver <Your IotHub> et <Your SAK> ?

Retournez dans votre IotHub

Ici le Nom de votre IotHub



Ici la clé (et même la chaine de connexion complète)

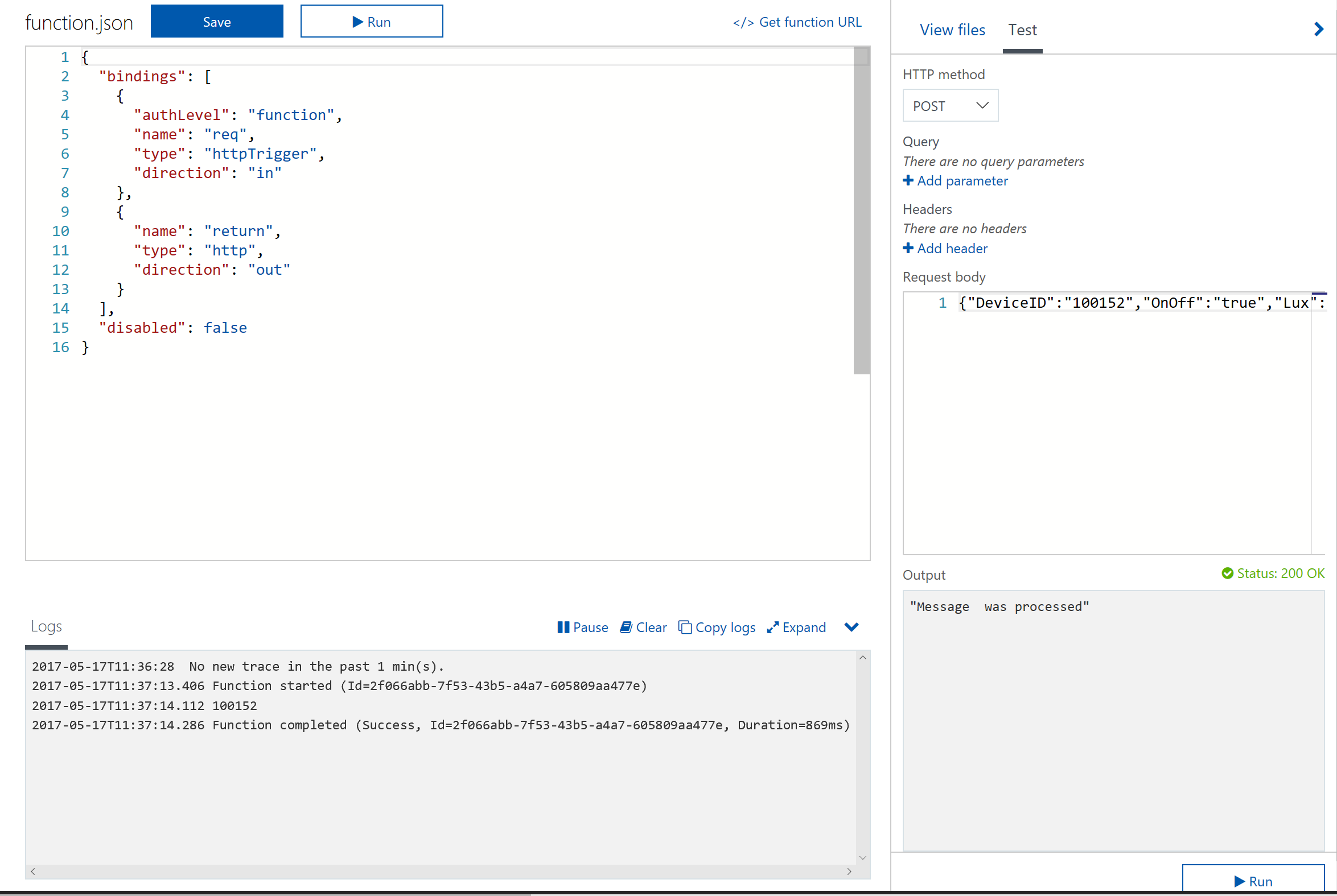


Code une fois modifié :

|  |
| --- |
| private static string connectionString = "HostName=IoTHubEurek.azure-devices.net;SharedAccessKeyName=registryReadWrite;SharedAccessKey=KYSIe4qcD1kI8rQtn18Ru43Fkb0POPzDV5KVx/qikPI=";  private static RegistryManager registryManager;  static DeviceClient deviceClient;  static string iotHubUri = "IoTHubEurek.azure-devices.net"; |

Pour tester votre fonction voici le corps de votre post : {"DeviceID":"100152","OnOff":"true","Lux":0,"Angle":"10"}

Vous pouvez l’utiliser dans la partie « Test » de votre fonction dans Azure :



Notre fonction est prête et nous pouvons maintenant utiliser le site Web pour simuler nos 5 devices.

Nous reviendrons sur les fonctions plus tard, pour en rajouter une sur le stockage de l’historique de l’état de nos devices.