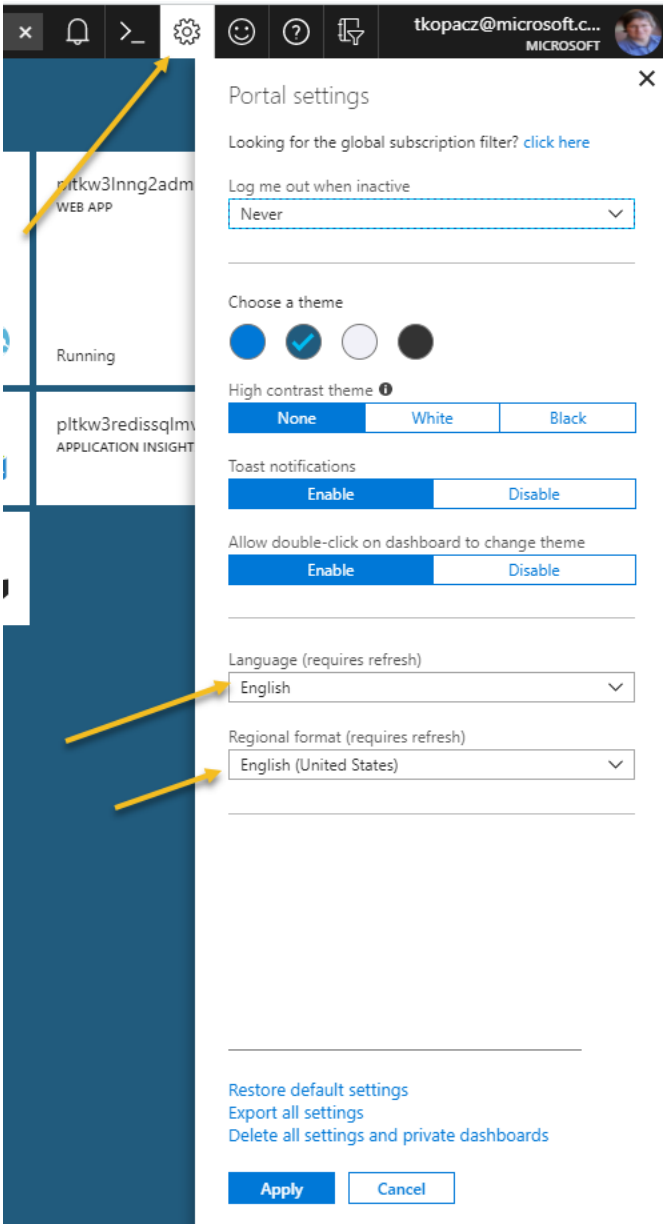
Workshop 1H

1. Uwaga! Do każdego stanowiska dołączone jest konto do logowania się do portalu azure
2. Portal do zarządzania: https://portal.azure.com/?feature.customportal=false
    a. Azure Cloud Shell (dla miłośników CLI w chmurze): https://shell.azure.com/
    b. Uwaga – ten skrypt zakłada używanie portalu!
    c. Zakładamy że w portalu wybrany jest język ANGIELSKI
3. Opcjonalne oprogramowanie (edytor kodu): https://code.visualstudio.com/
4. Ważne – jak portal nie działa – to **wyłączyć ad-block** w używanej przeglądarce.
5. Uwaga! Skrypt z założenia ma towarzyszyć w „*instructor-led lab*", na wypadek, gdyby uczestnik „się zgubił" lub musiał wrócić kilka kroków.

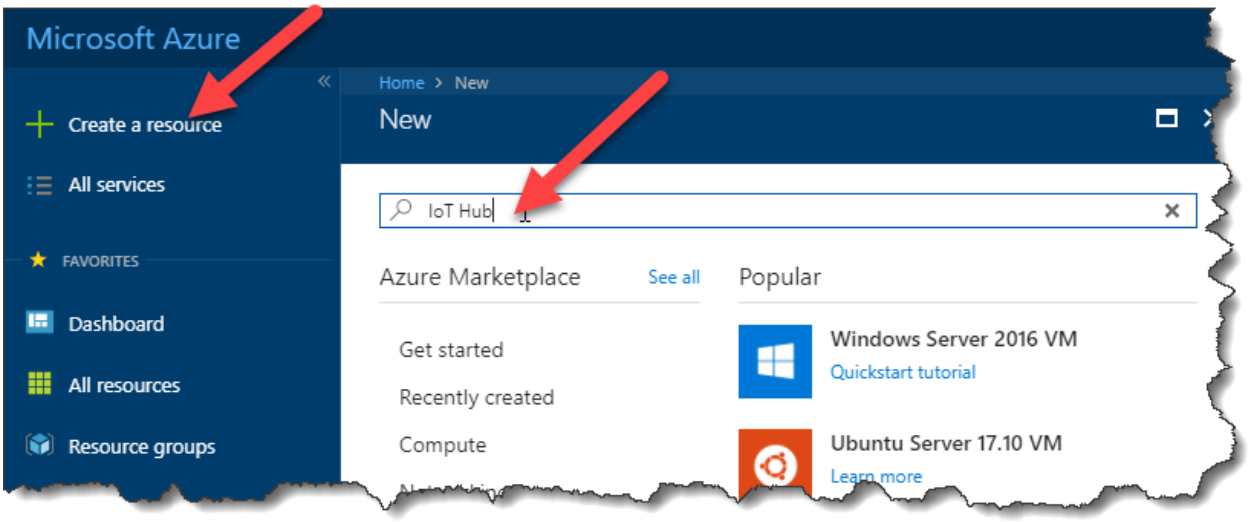
Tworzenie zasobów w Azure

Ustawienie języka angielskiego



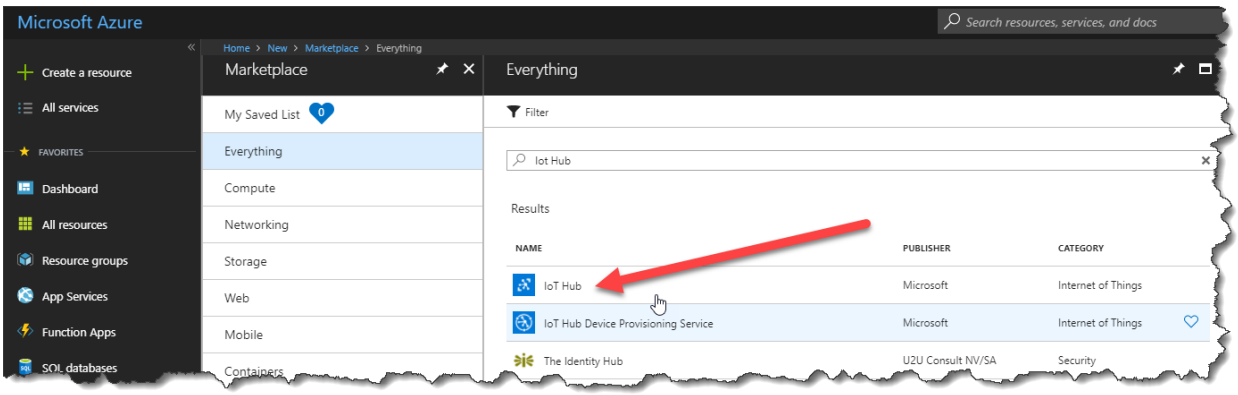
Rysunek 1 Ustawienie języka angielskiego na portalu

Założenie IoT Hub i Resource Group

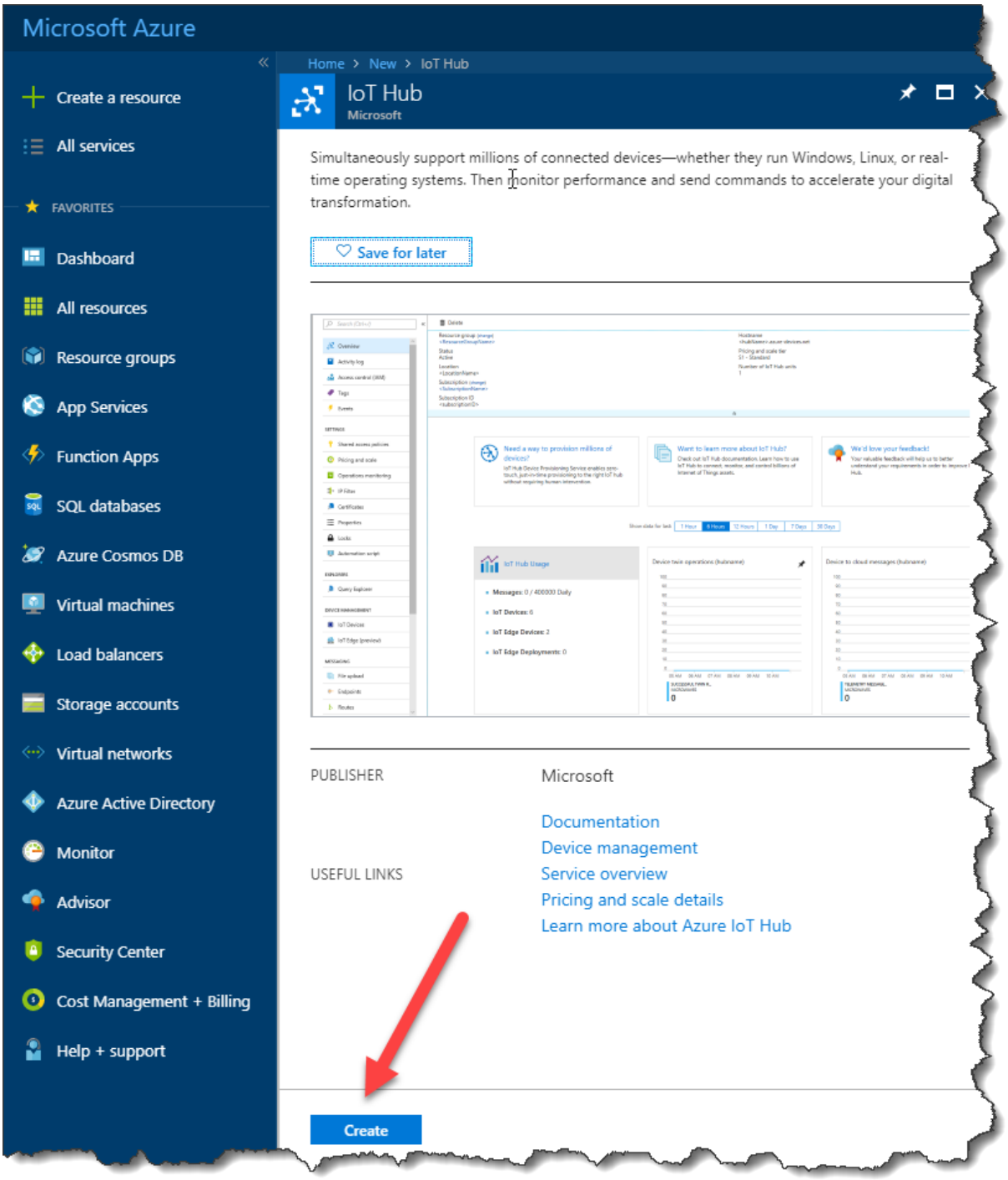


Rysunek 2 Wyszukanie IoT Hub

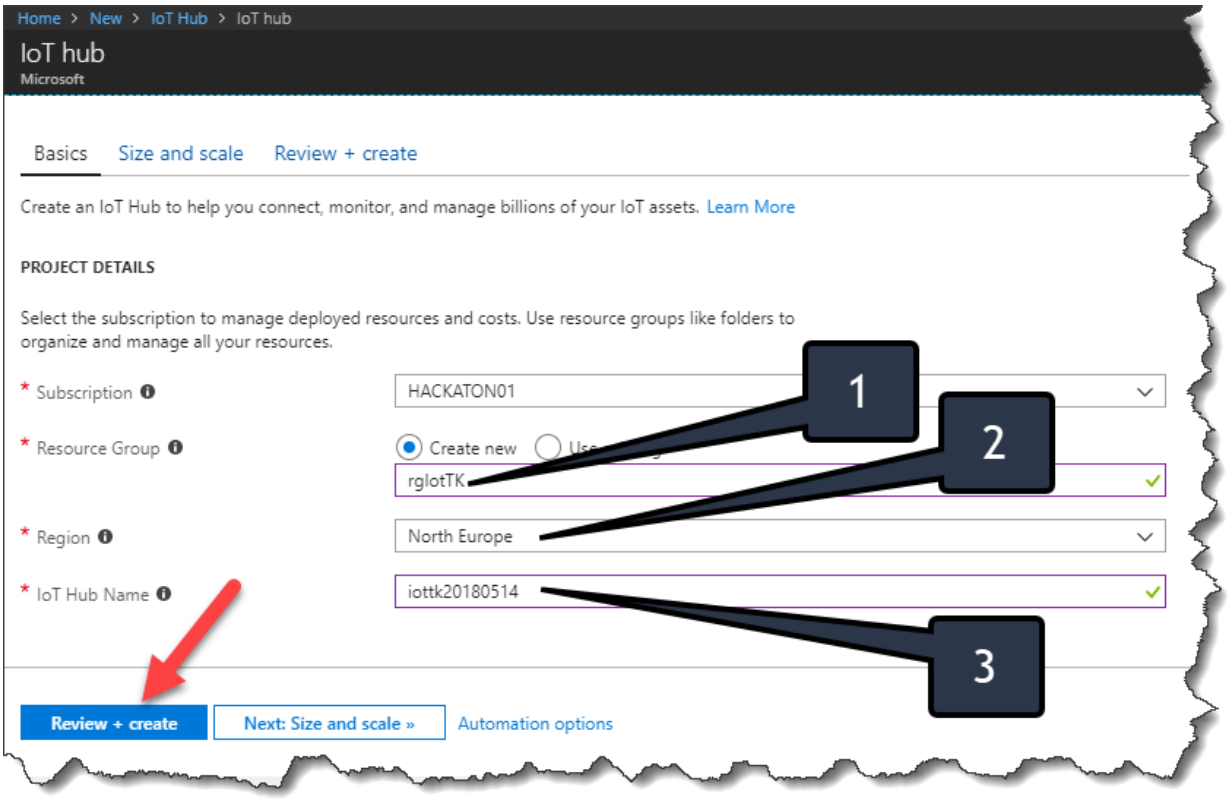
Po kliknięciu Enter pojawi się lista usług.



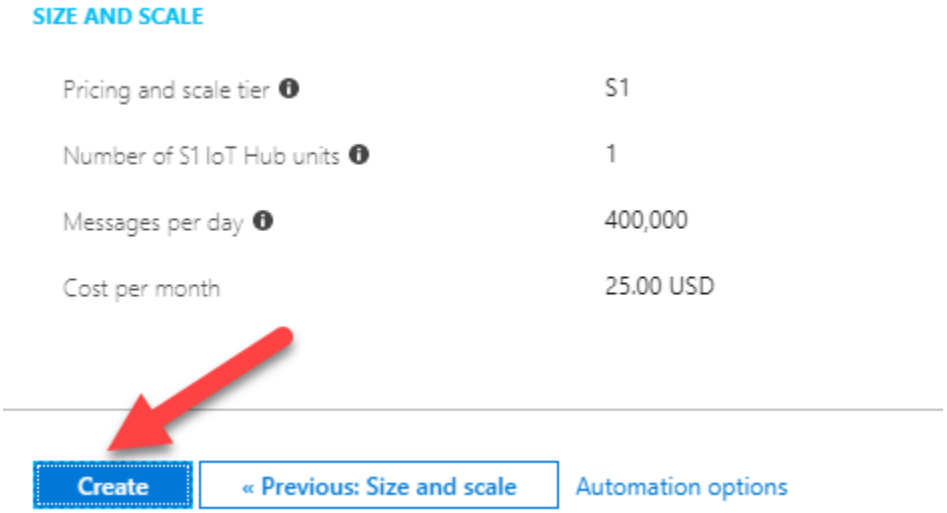
Rysunek 3 Wybór IoT Hub



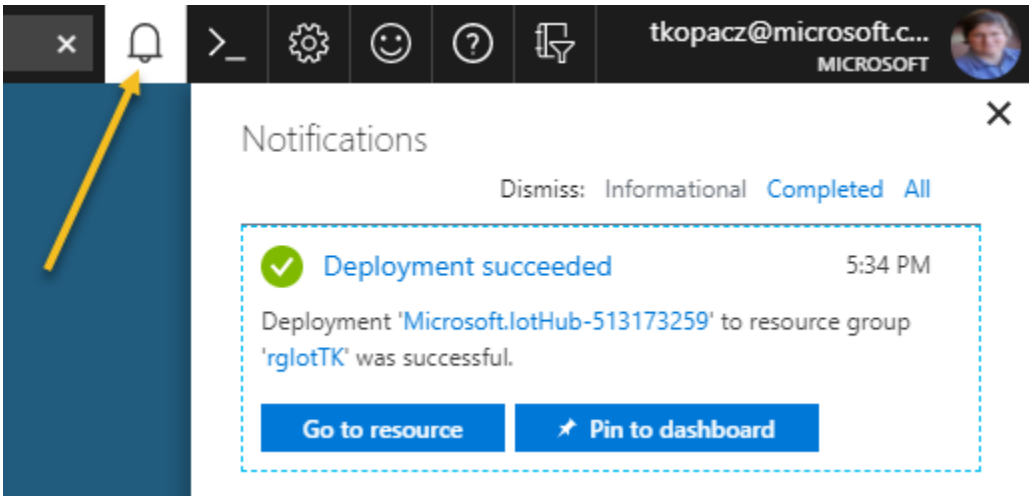
Rysunek 4 Tworzenie IoT Hub



Rysunek 5 Parametry IoT Hub. Należy podać nazwę Resource Group, region i nazwę (nazwa musi być unikalna, najlepiej – inicjały + dzisiejsza data)



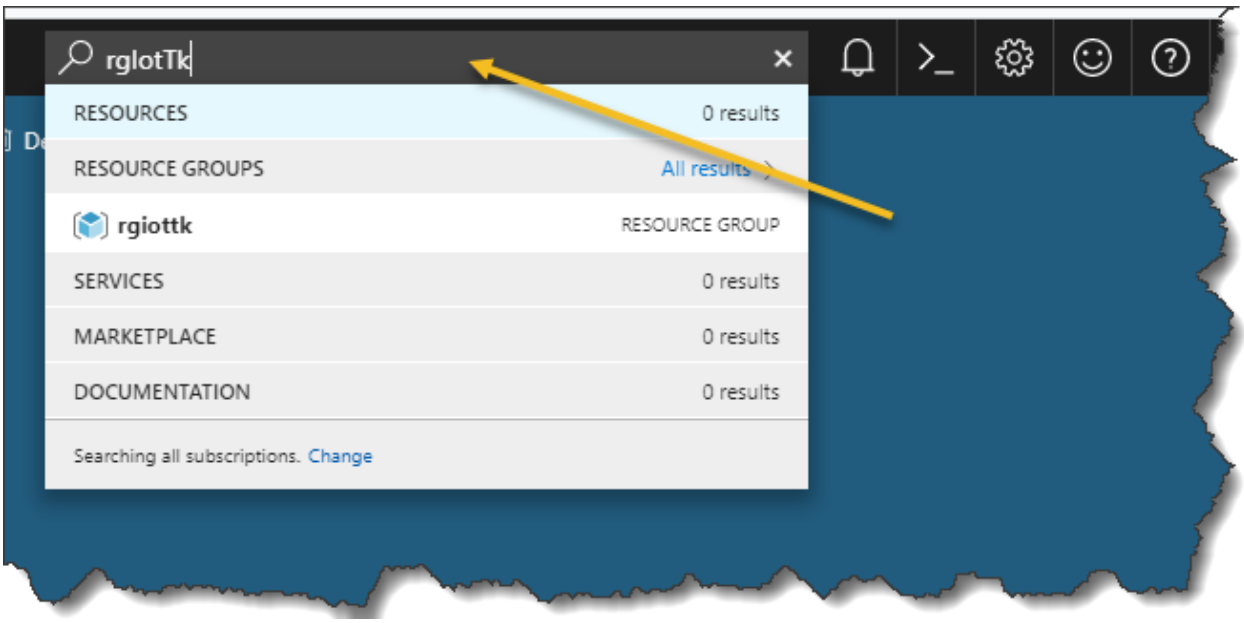
Rysunek 6 Koszty i podsumowanie



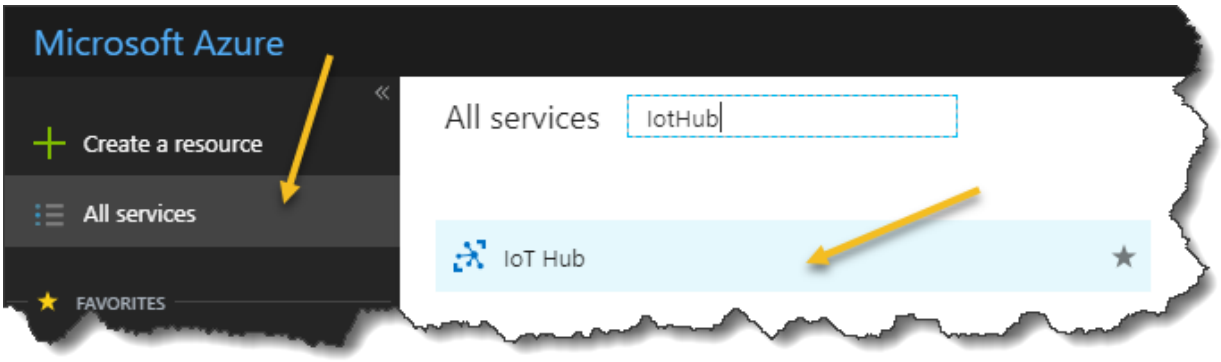
Rysunek 7 Po chwili pojawi się okno stanu z założonym zasobem

Znalezienie IoT Hub (i Resource Group)

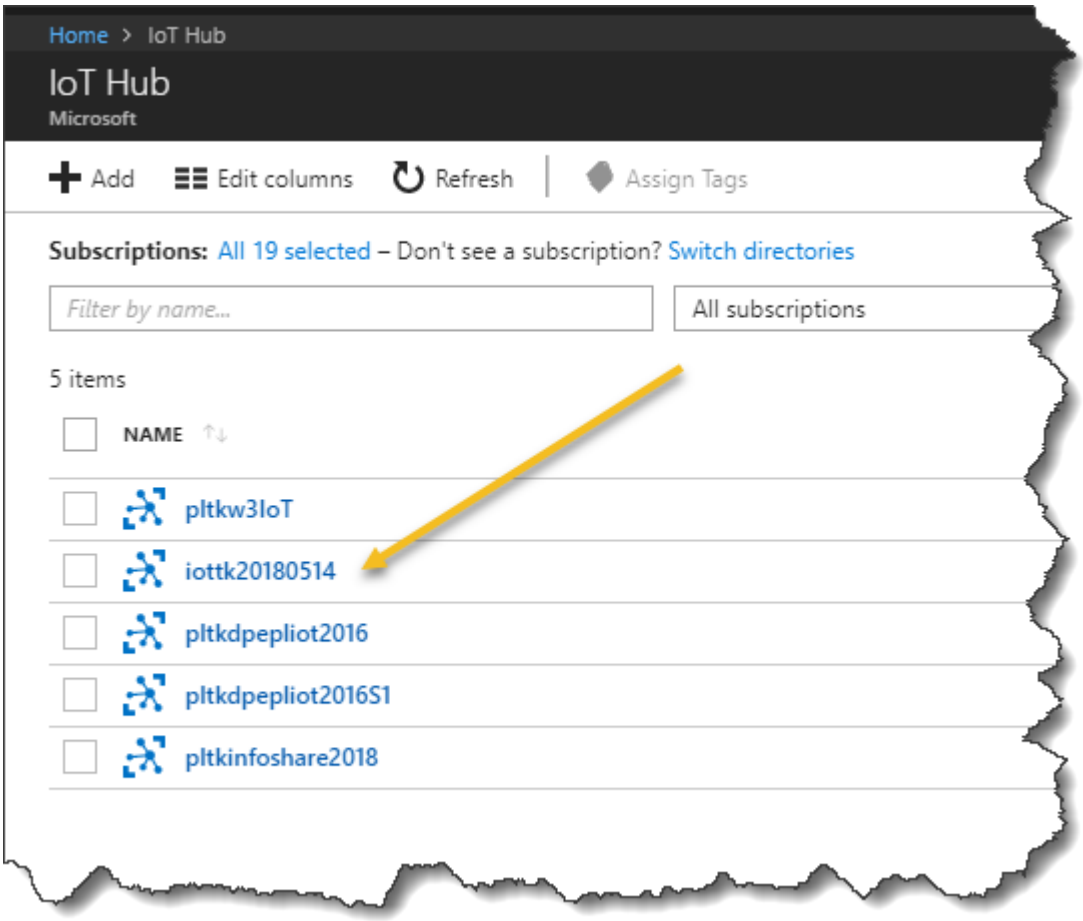
Jeżeli nie przypniemy zasobu do dashboard („pin” na Rysunek 7 Po chwili pojawi się okno stanu z założonym zasobem).



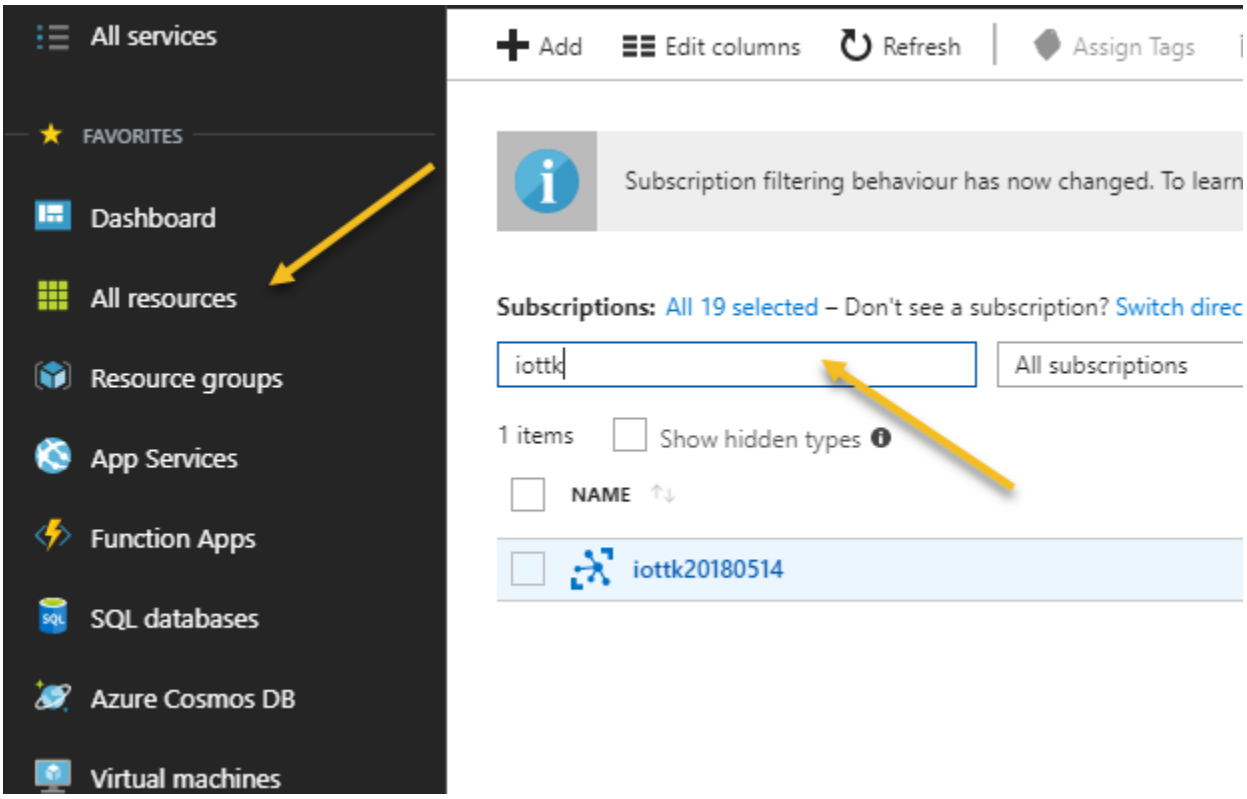
Rysunek 8 Wyszukiwanie Resource Group



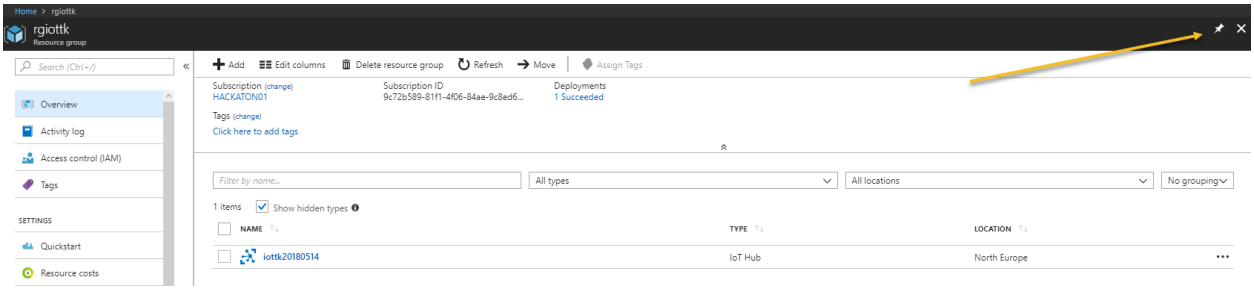
Rysunek 9 Wyszukiwanie IoT Hub



Rysunek 10 Lista IoT Hub we wszystkich subskrypcjach (na szkoleniu zwykle jest jedna)

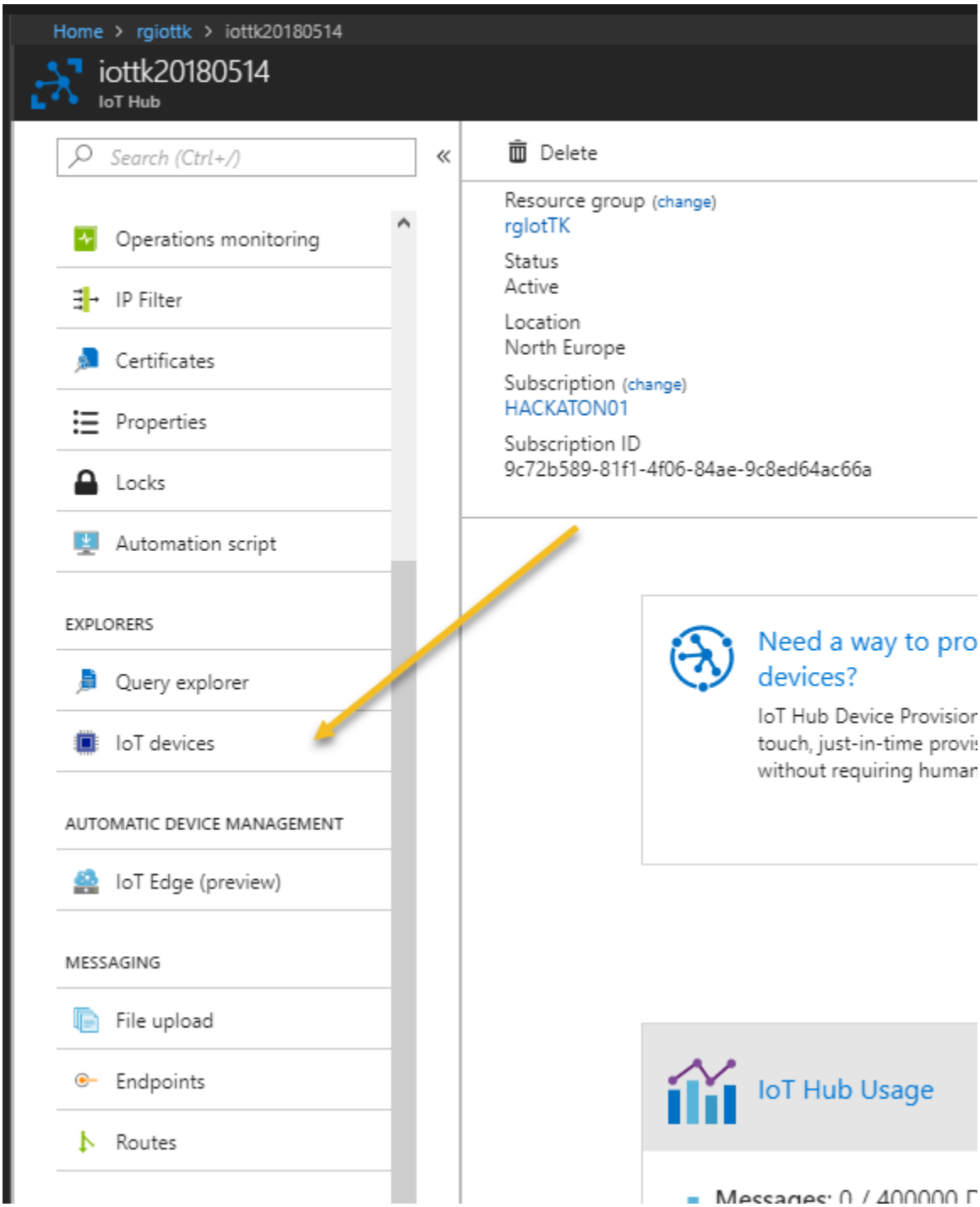


Rysunek 11 Filtrowanie listy wszystkich zasobów

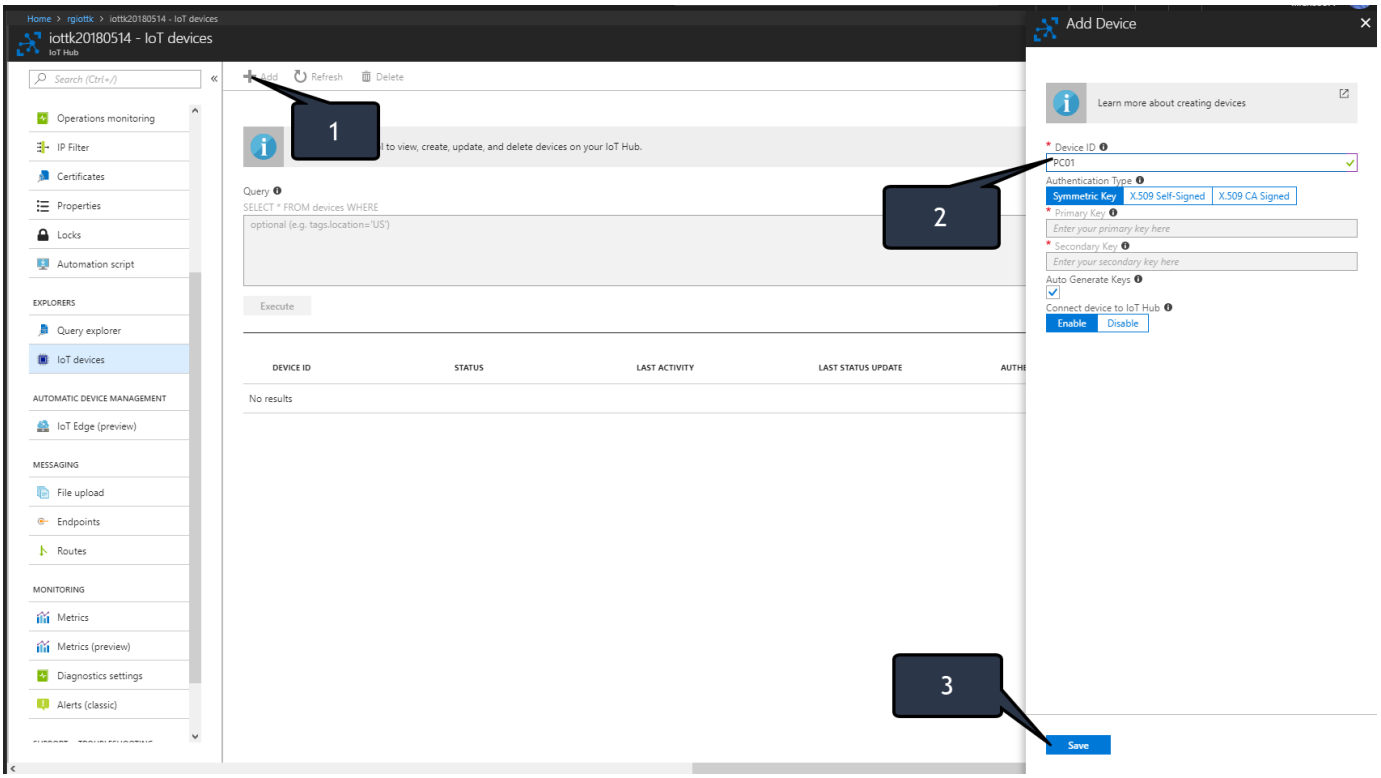


Rysunek 12 Wygodnie przypiąć do dashboard

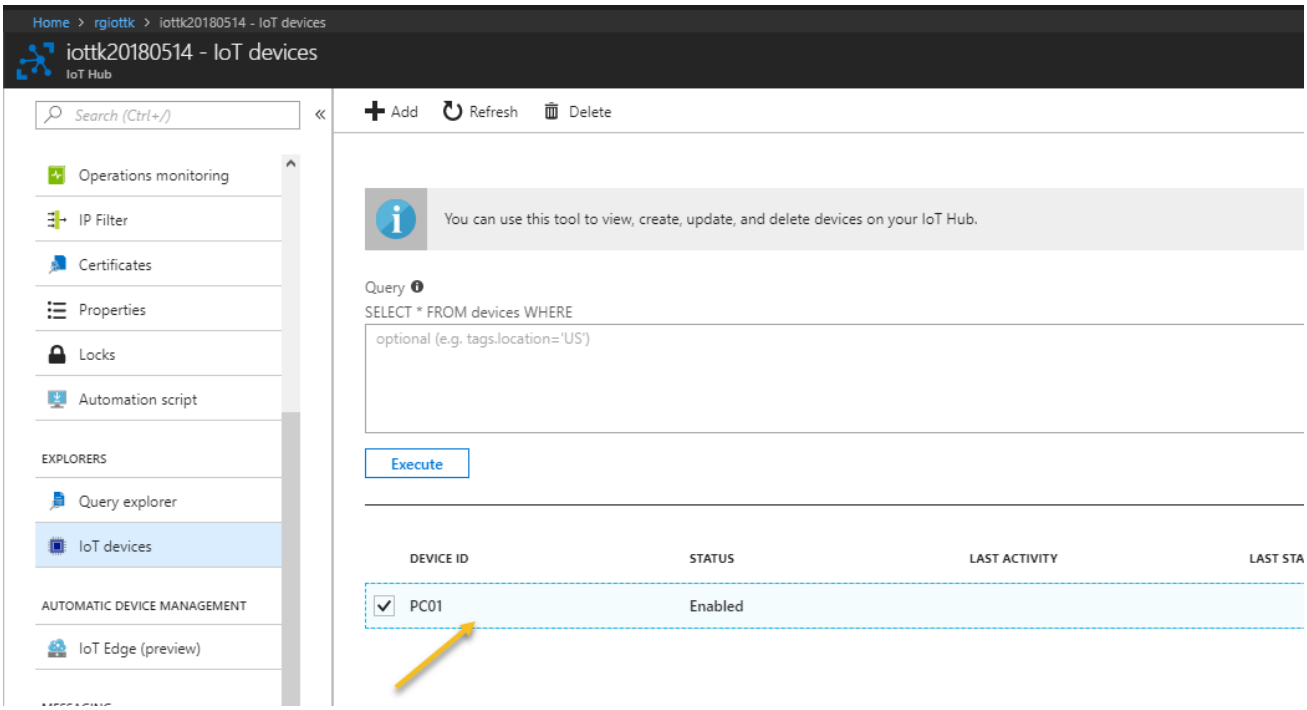
Rejestracja urządzenia



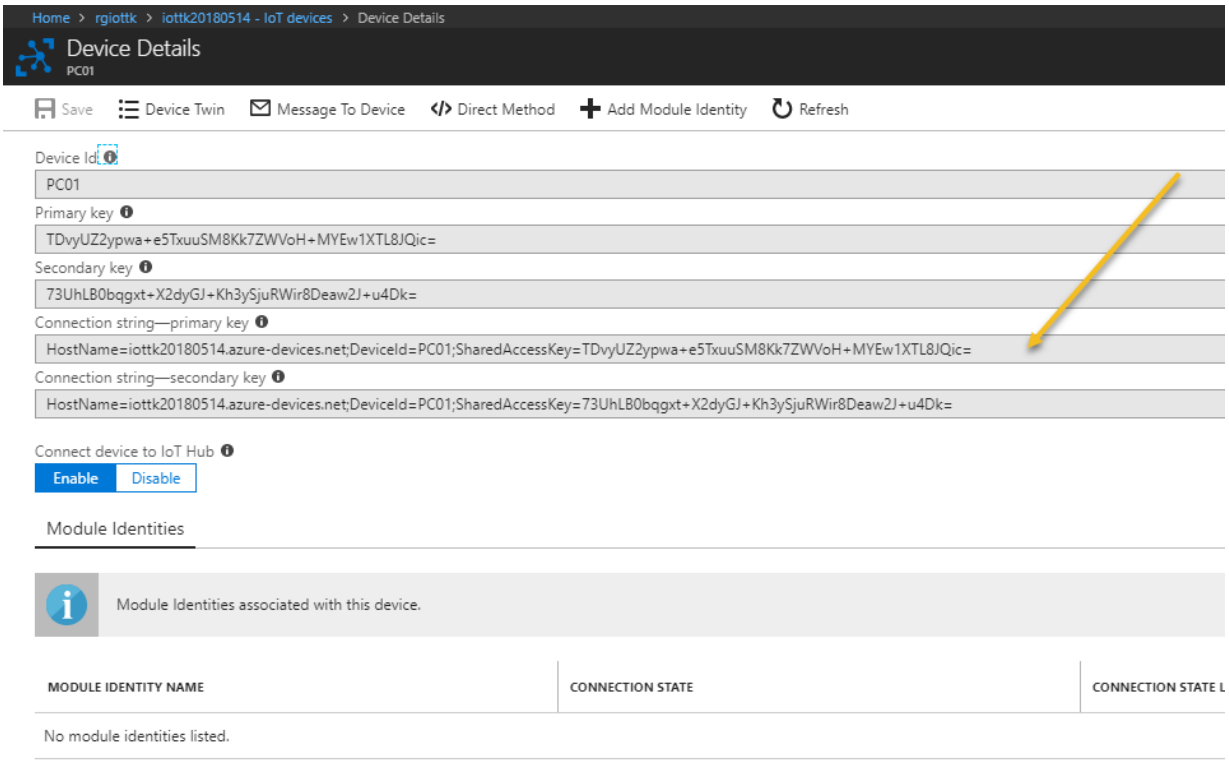
Rysunek 13 Zarządzanie urządzeniami



Rysunek 14 Dodanie nowego urządzenia (klucze się same wygenerują)



Rysunek 15 Szczegóły urządzenia

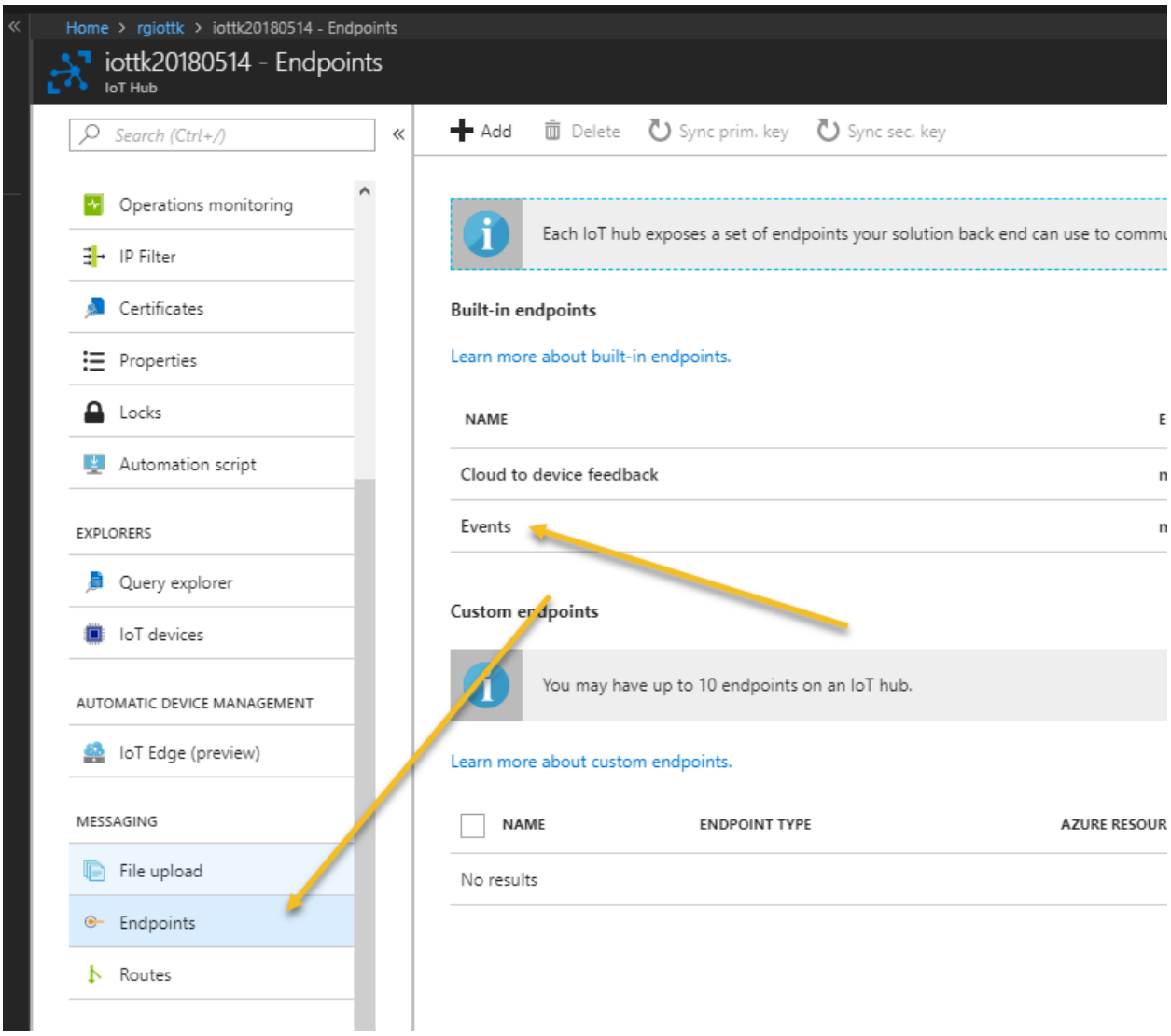


Rysunek 16 Connection String – skopiować

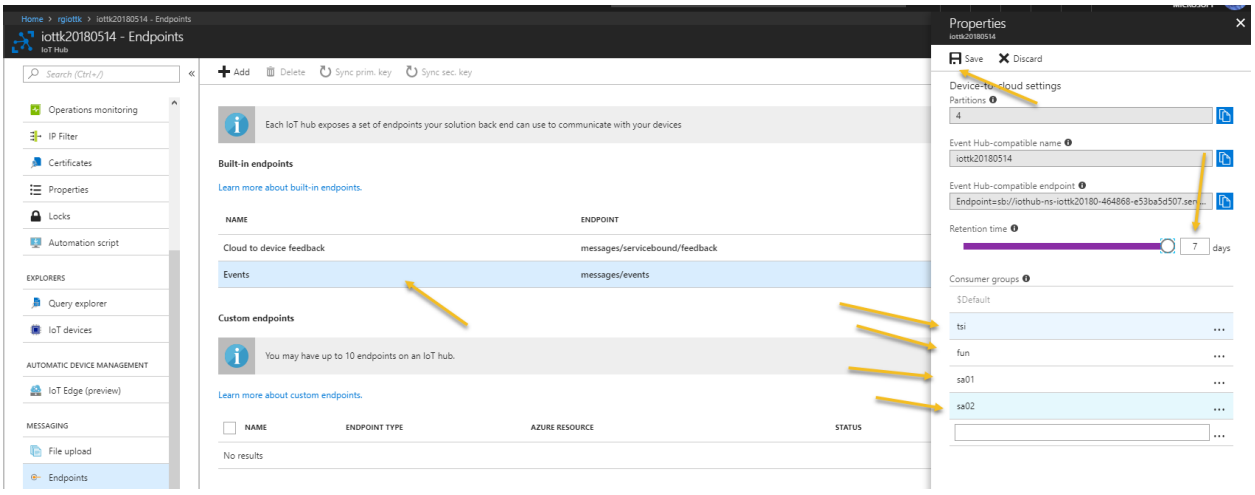
**** Przykład łańcucha:

HostName=iottk20180514.azure-devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA

Dodanie czterech consumer group

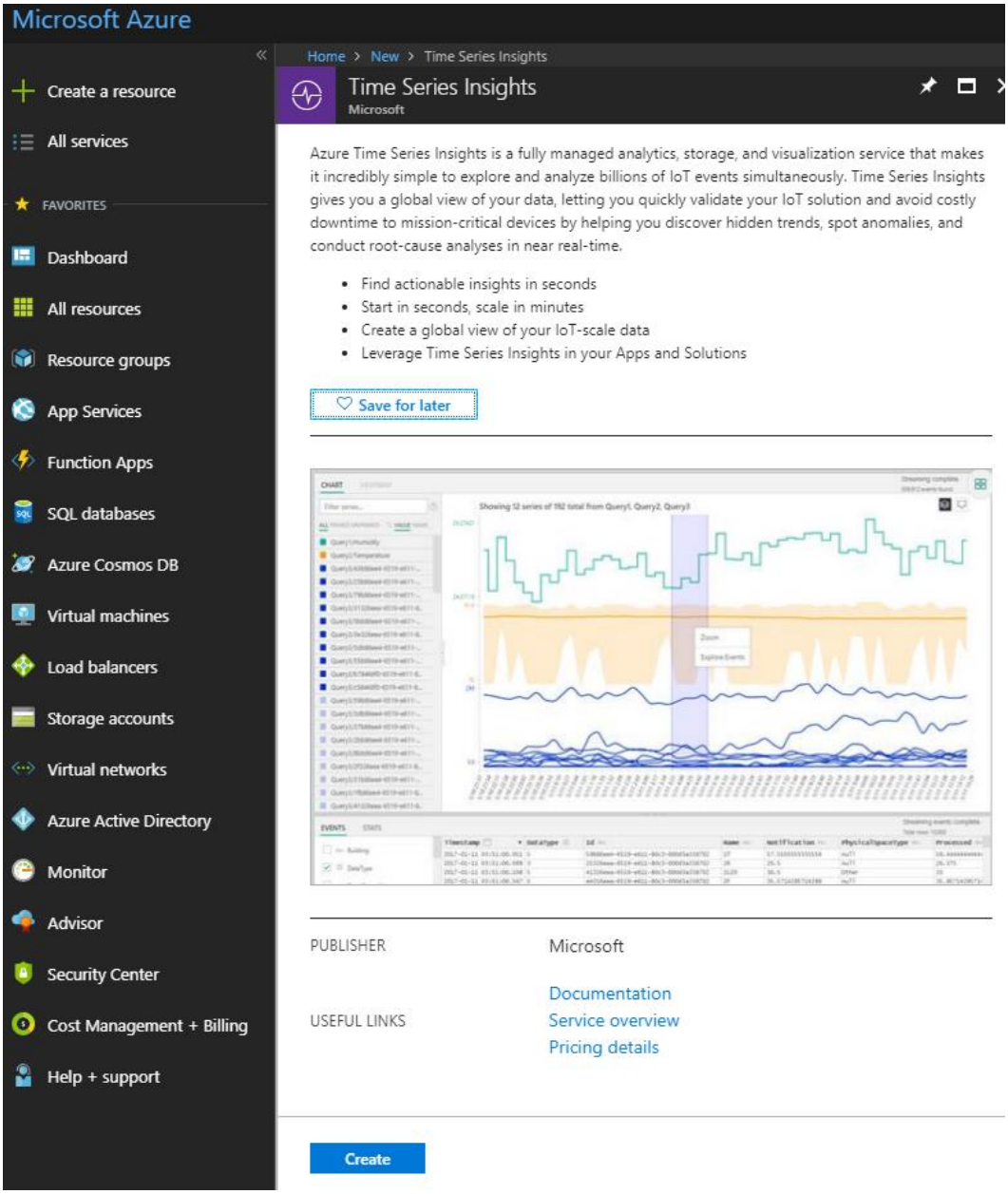


Rysunek 17 Dodawanie consumer group 1/2

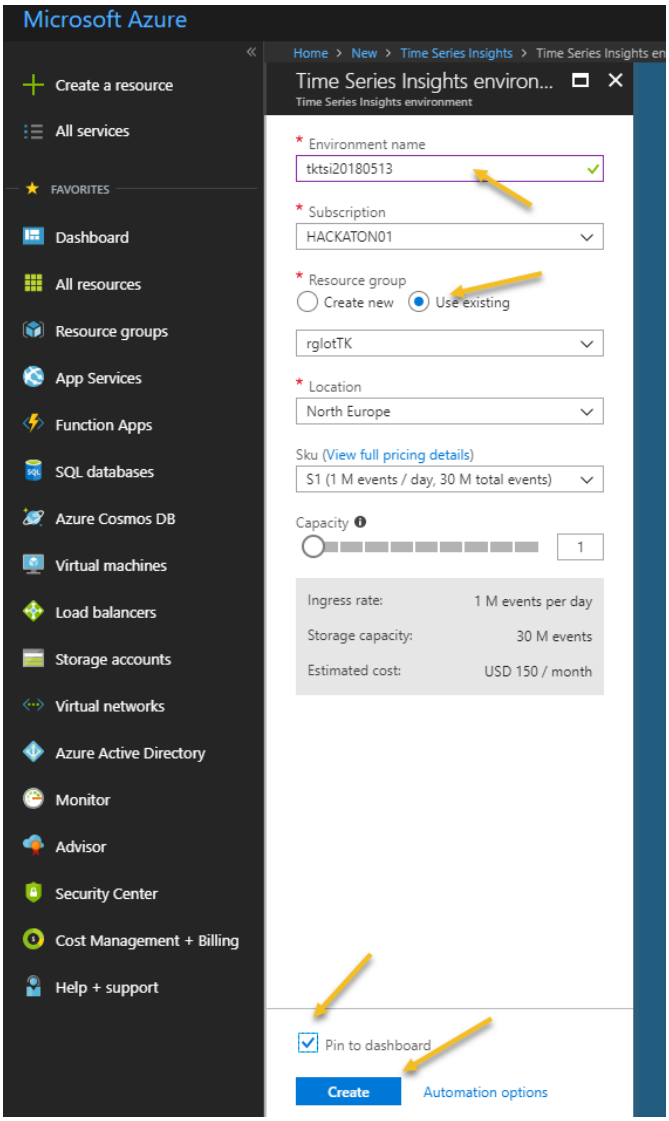


Rysunek 18 Dodawanie consumer group 2/2

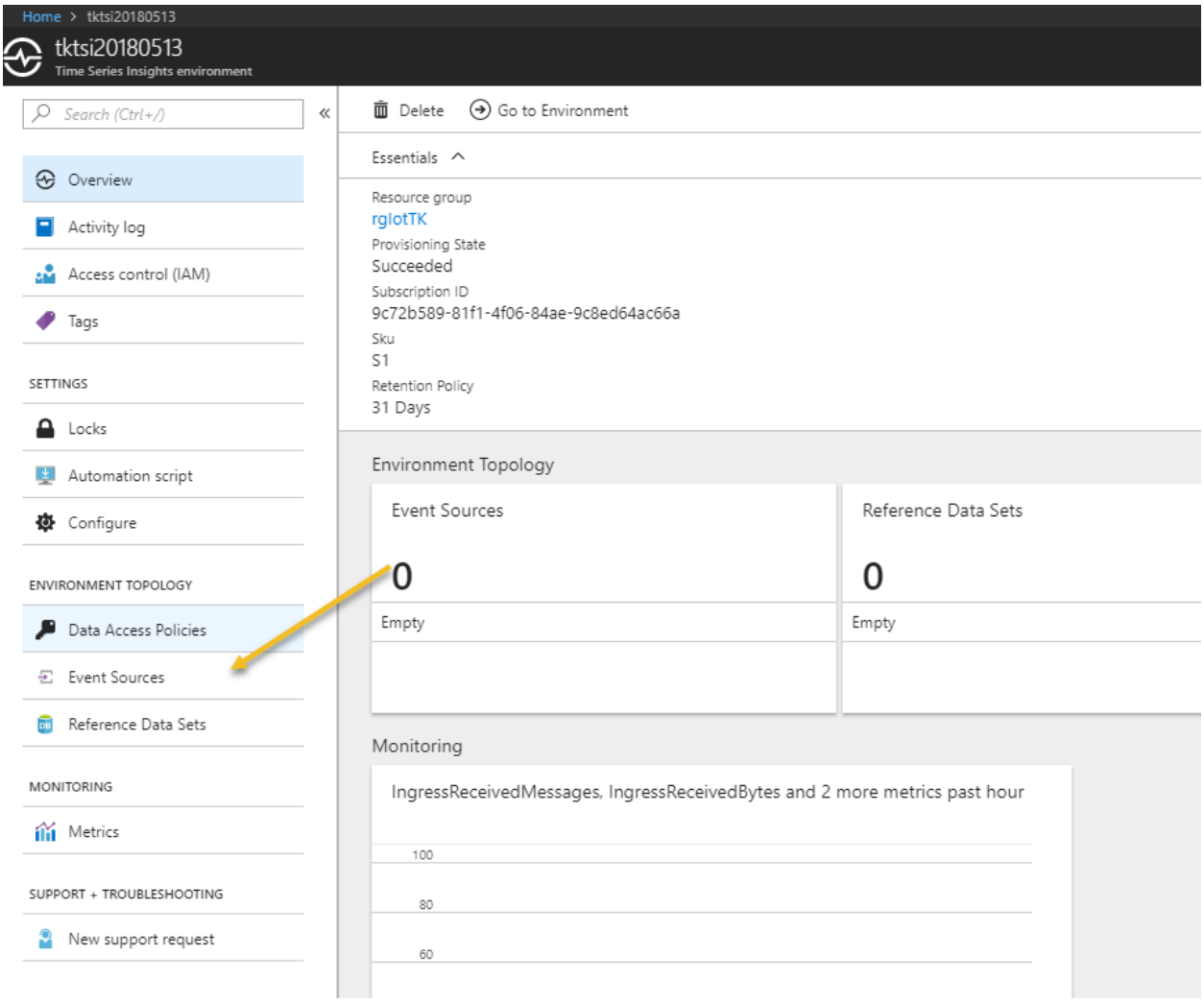
Tworzenie Time Series Insight (do podglądu wiadomości)



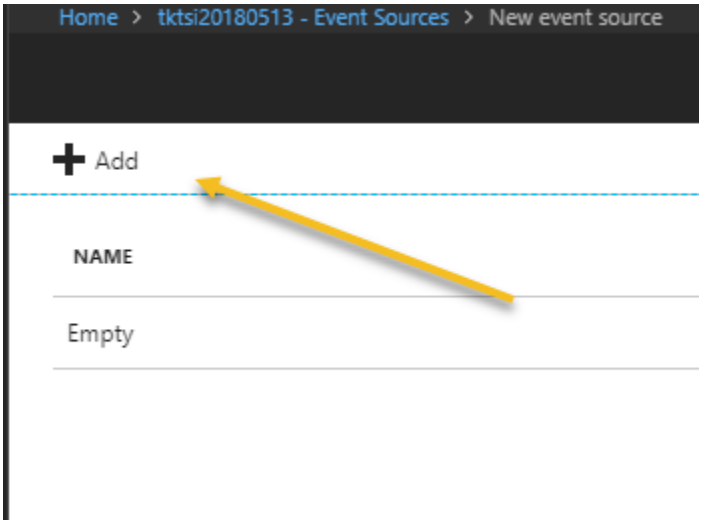
Rysunek 19 Tworzenie TSI



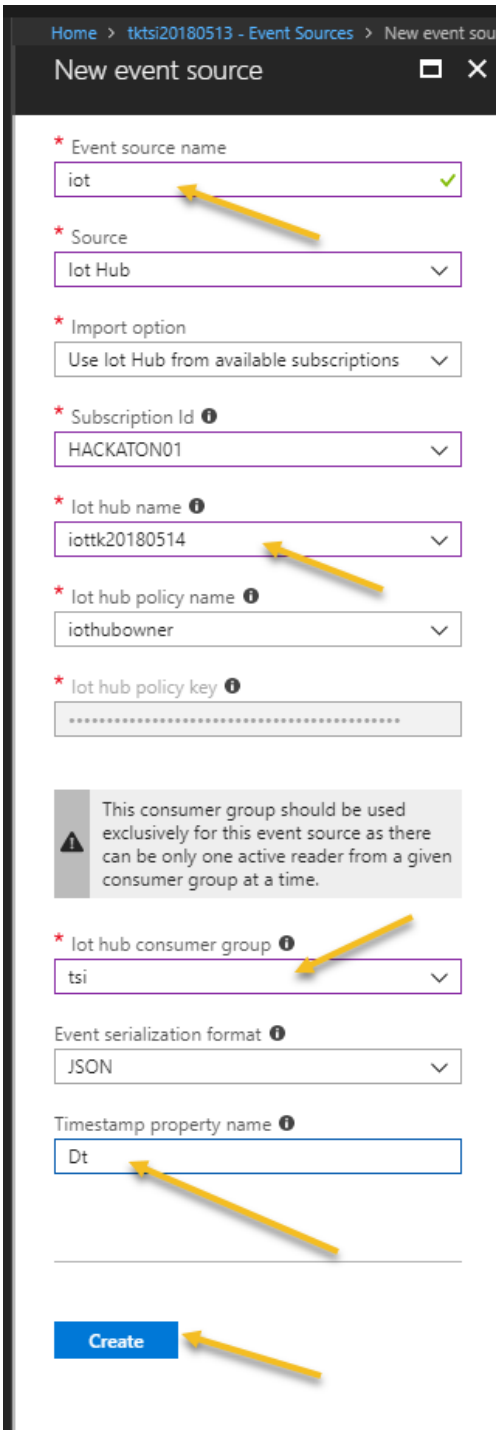
Rysunek 20 Parametry TSI

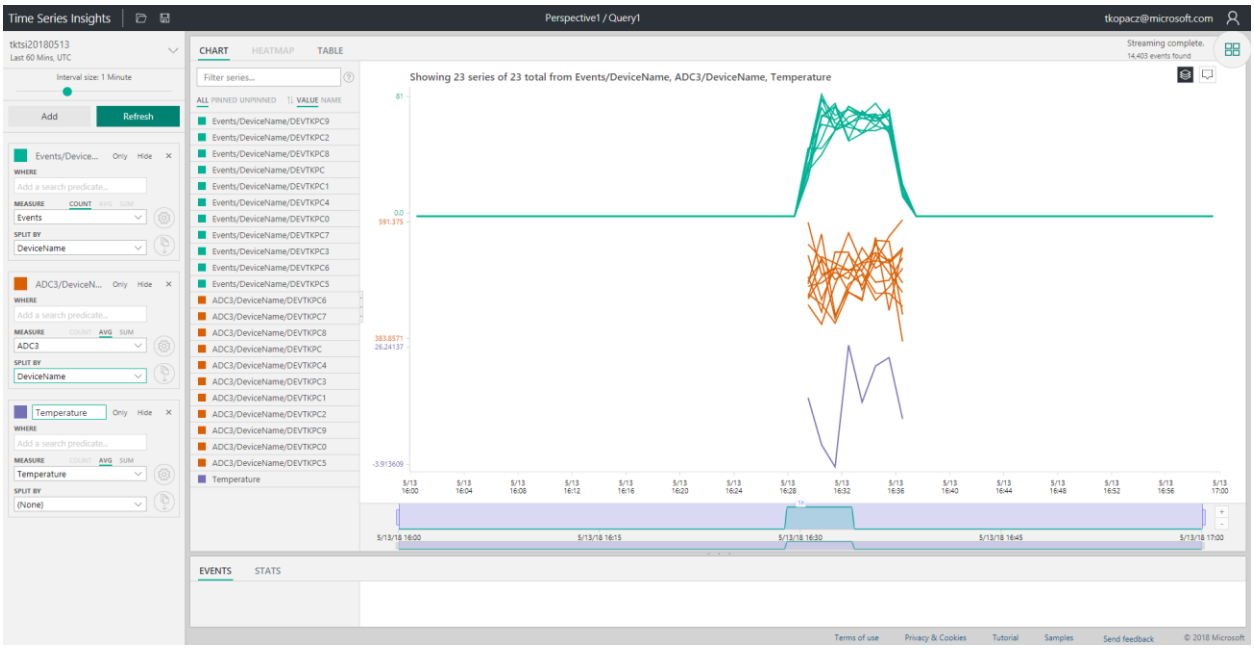


Rysunek 21 Dodanie źródła dla TSI



Rysunek 22 Wskazanie stworzonego IoT 1/2


Home > tktsi20180513 - Event Sources > New event sou
New event source
* Event source name
iot
* Source
Iot Hub
* Import option
Use Iot Hub from available subscriptions
* Subscription Id
HACKATON01
* Iot hub name
iottk20180514
* Iot hub policy name
iothubowner
* Iot hub policy key
This consumer group should be used exclusively for this event source as there can be only one active reader from a given consumer group at a time.
* Iot hub consumer group
tsi
Event serialization format
JSON
Timestamp property name
Dt
Create



Rysunek 23 Widok TSI

Wysyłanie telemetrii do IoT Hub

Prace z kodem zakłada że kod źródłowy jest w Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\

Jeżeli znajduje się w innym folderze – ścieżkę do plików należy odpowiednie zmodyfikować.

.NET Core

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step01SendNetCore\Step01SendNetCore\Program.cs

Zmienić:

```
            string cnn = "HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA";
```

Na:

```
            string cnn = "skopiowany łańcuch połączenia w poprzednim kroku, oznaczony **** ";
```

Uruchomić

Node JS

npm install

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step01SendNodeJS\SimulatedDevice\SimulatedDevice.js

```
var connectionString = 'HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA';
```

Na:

```
var connectionString = 'skopiowany łańcuch połączenia w poprzednim kroku, oznaczony **** ';
```

Python

pip install azure-iothub-device-client azure-iothub-service-client

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step01SendPython\SimulatedDevice.py

```
CONNECTION_STRING = "HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA"
```

Na

```
CONNECTION_STRING = " skopiowany łańcuch połączenia w poprzednim kroku, oznaczony **** "
```

Podgląd komunikatów

Po pewnym czasie (kilka minut) proszę zobaczyć co jest widoczne na Time Series Insight

Wysyłanie poleceń z chmury do urządzenia

Aby to zadziałało, urządzenie musi

1. Albo nasłuchiwać na protokole http (post + timeout)
2. Albo otworzyć sesję AMQP
3. Albo nasłuchiwać na odpowiednim topic-u MQTT (ten sposób tu użyjemy)

.NET Core – po stronie urządzenia

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step02ReceiveCommandsNetCore\Step02ReceiveCommandsNetCore\Program.cs

Zmienić:

```
            string cnn = "HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA";
```

NodeJS – po stronie urządzenia

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step02ReceiveCommandsNodeJS\receive.js

Zmienić:

```
var connectionString = 'HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA';
```

Python – po stronie urządzenia

Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step02ReceiveCommandsPython\Receive\receive.py

Zmienić:

```
CONNECTION_STRING = "HostName=iottk20180514.azure-
devices.net;DeviceId=PC01;SharedAccessKey=ABABABABA"
```

Python – do wysyłania poleceń

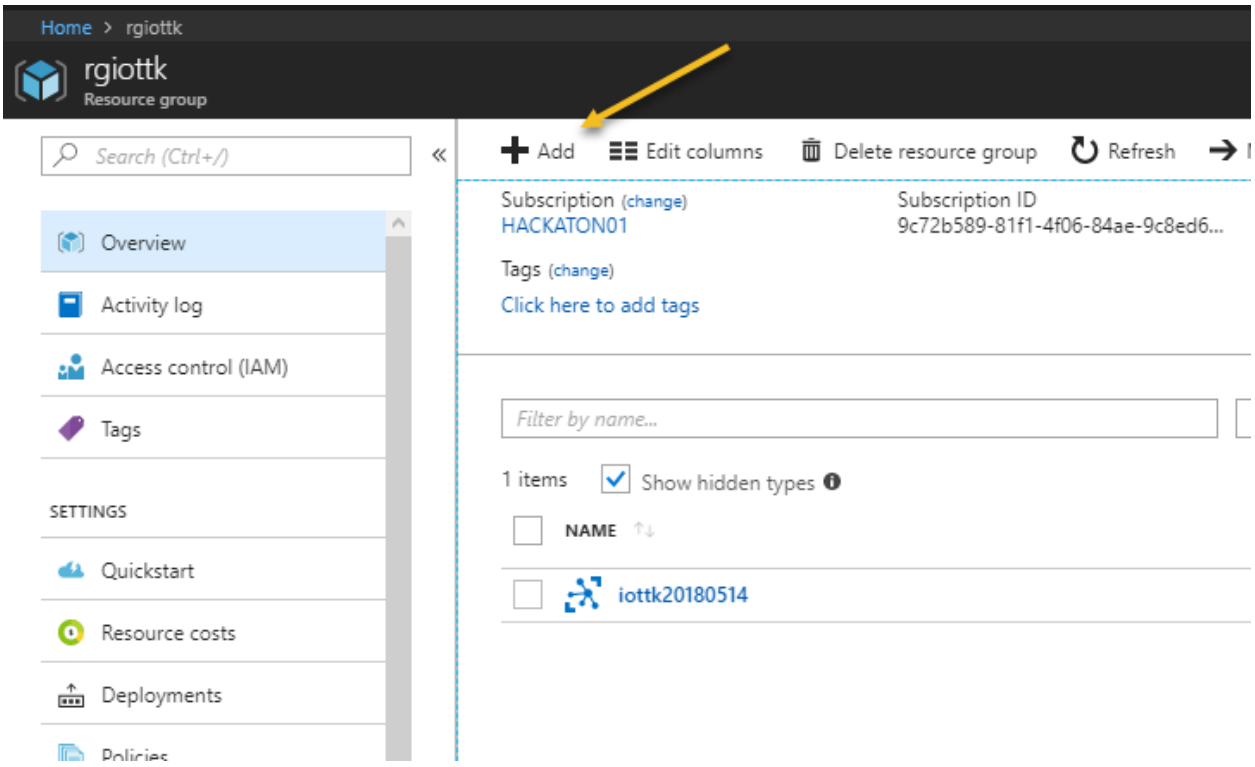
Z:\AzureFY18TK\20-IoT-Hub\2018_Workshop_1H\Step02ReceiveCommandsPython\Send\BackEndApplication.py

Uwaha! Używamy połączenia Service z Iot Hub – a nie konkretnego dla Device.

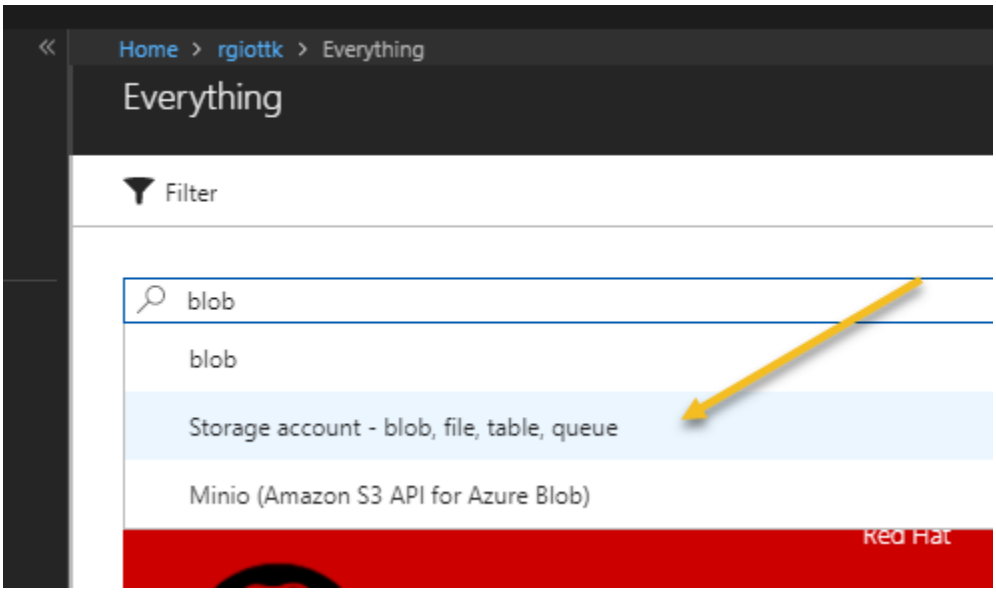
```
CONNECTION_STRING = "HostName=iottk20180514.azure-
devices.net;SharedAccessKeyName=service;SharedAccessKey=AAAAAA"
```

Analiza strumieniowa

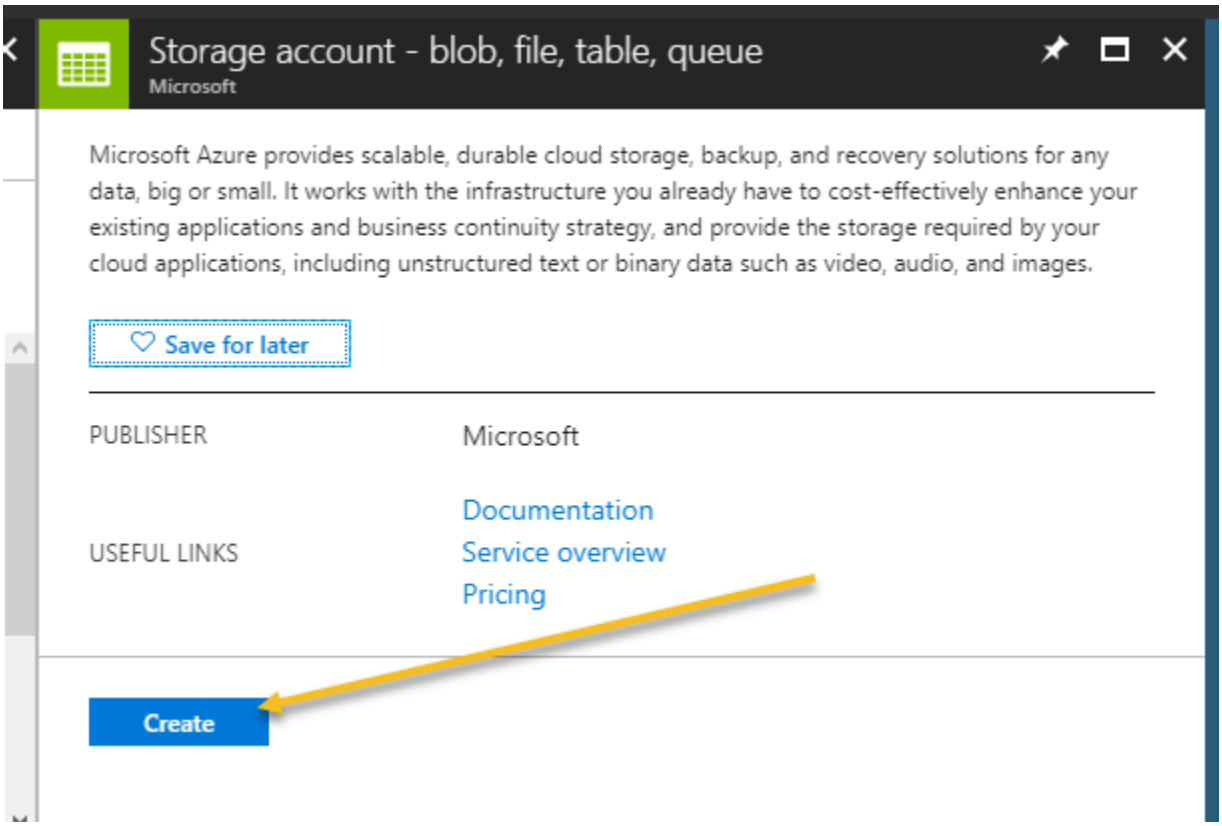
Tworzenie Azure Blob



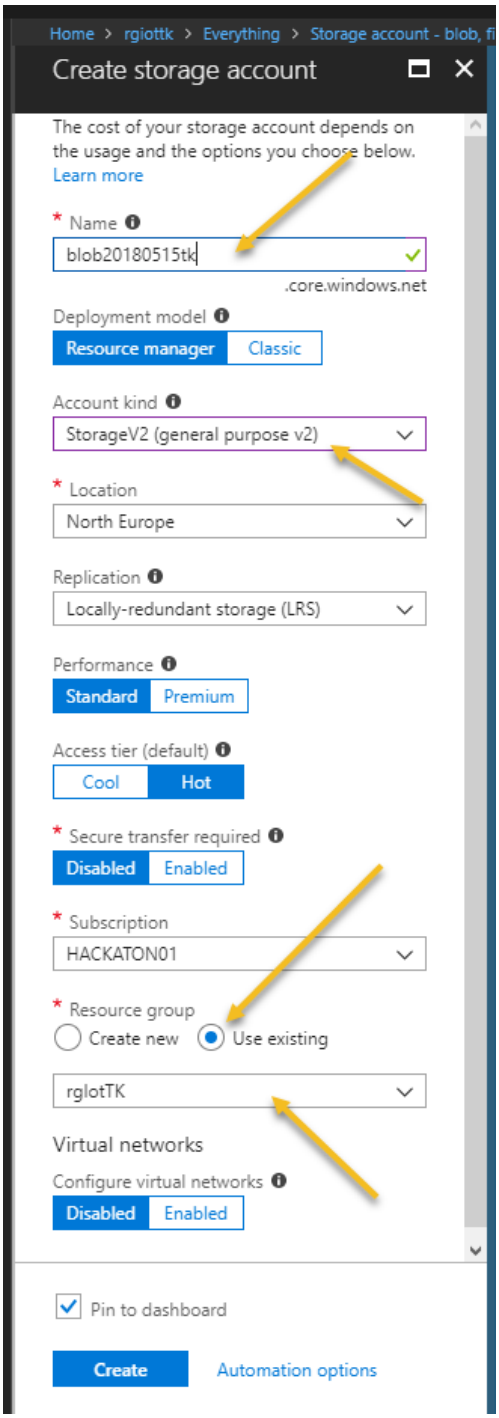
Rysunek 24 Tworzeie BLOB (do zapisu danych)



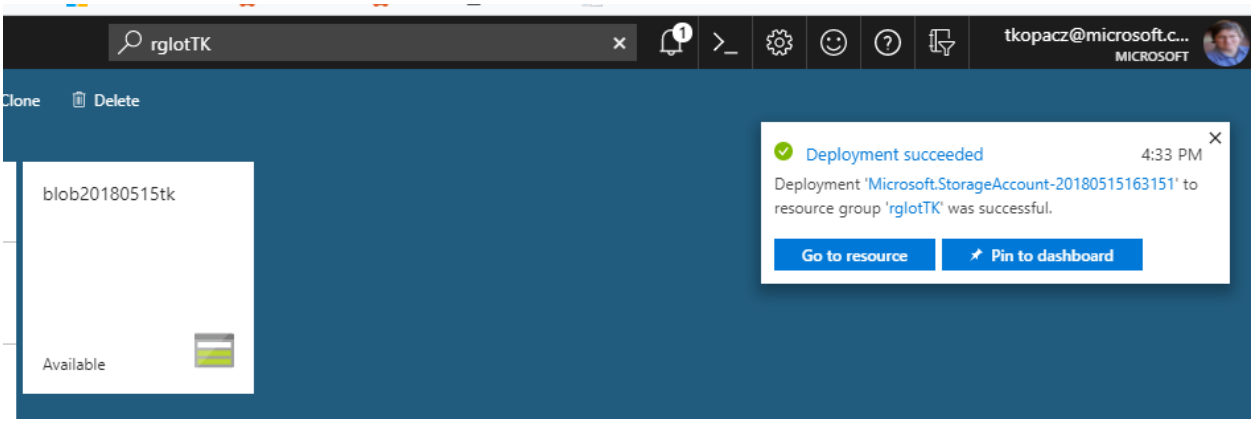
Rysunek 25 Wybór Storage



Rysunek 26 Create

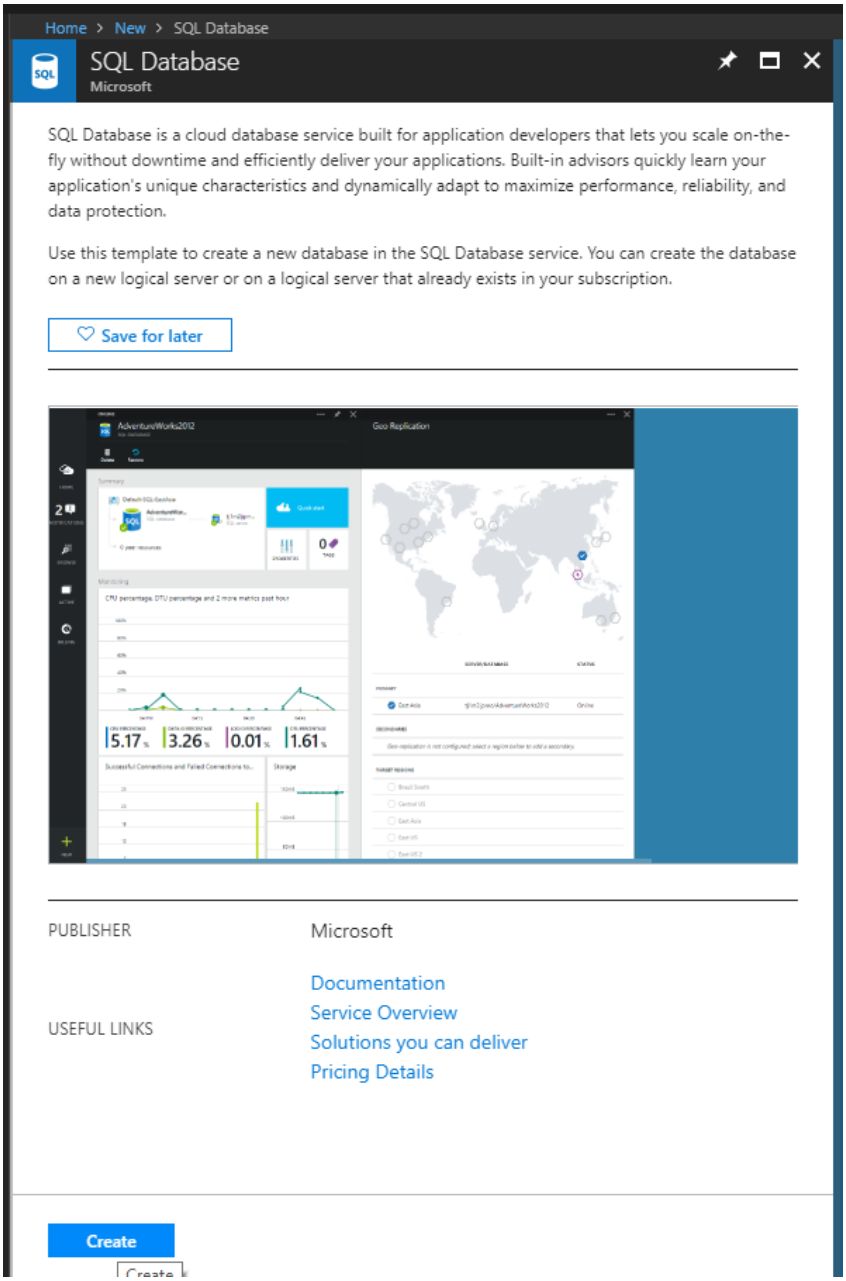


Rysunek 27 Parametry konta. Nazwa (data + inicjały), wersja V2, ta sama Resource Group co poprzednio

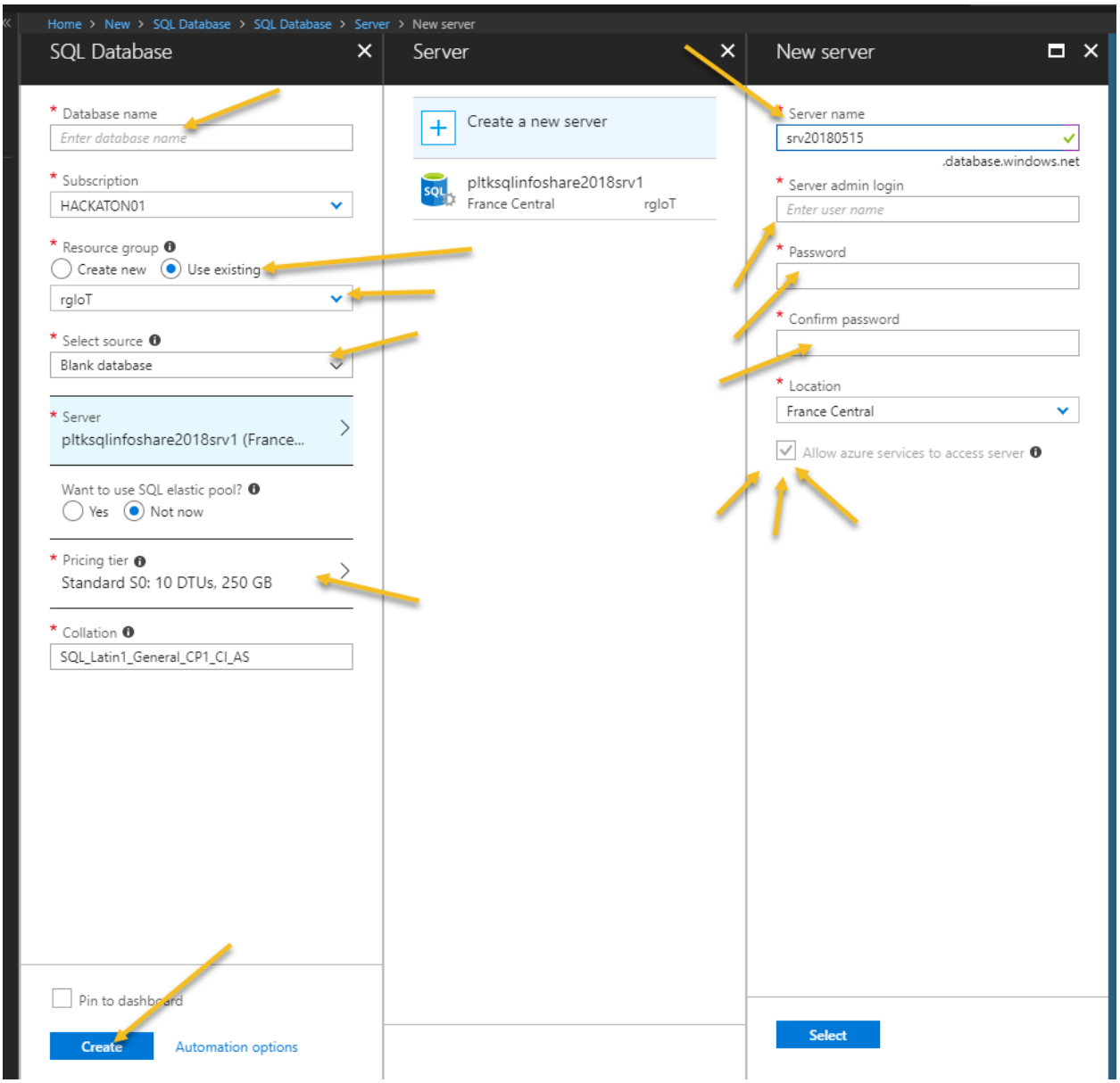


Rysunek 28 Warto konto przypiąć

Tworzenie SQL Database i SQL Server

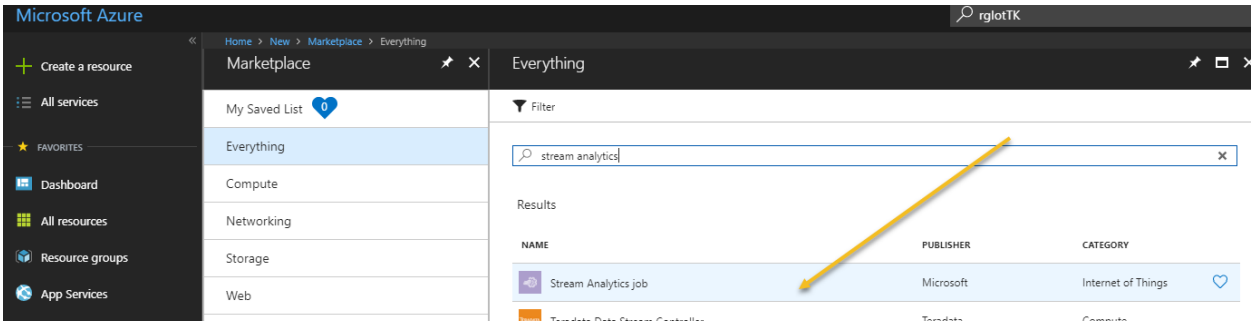


Rysunek 29 Tworzenie bazy danych

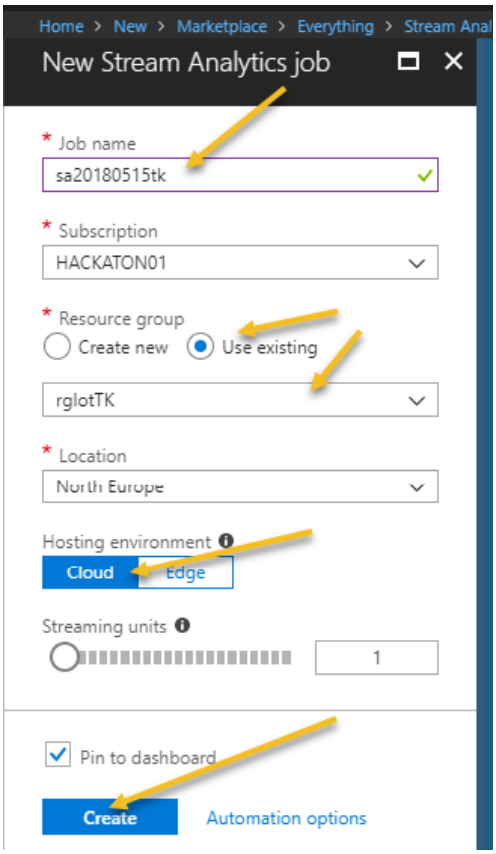


Rysunek 30 Parametry SQL

Tworzenie Stream Analytics

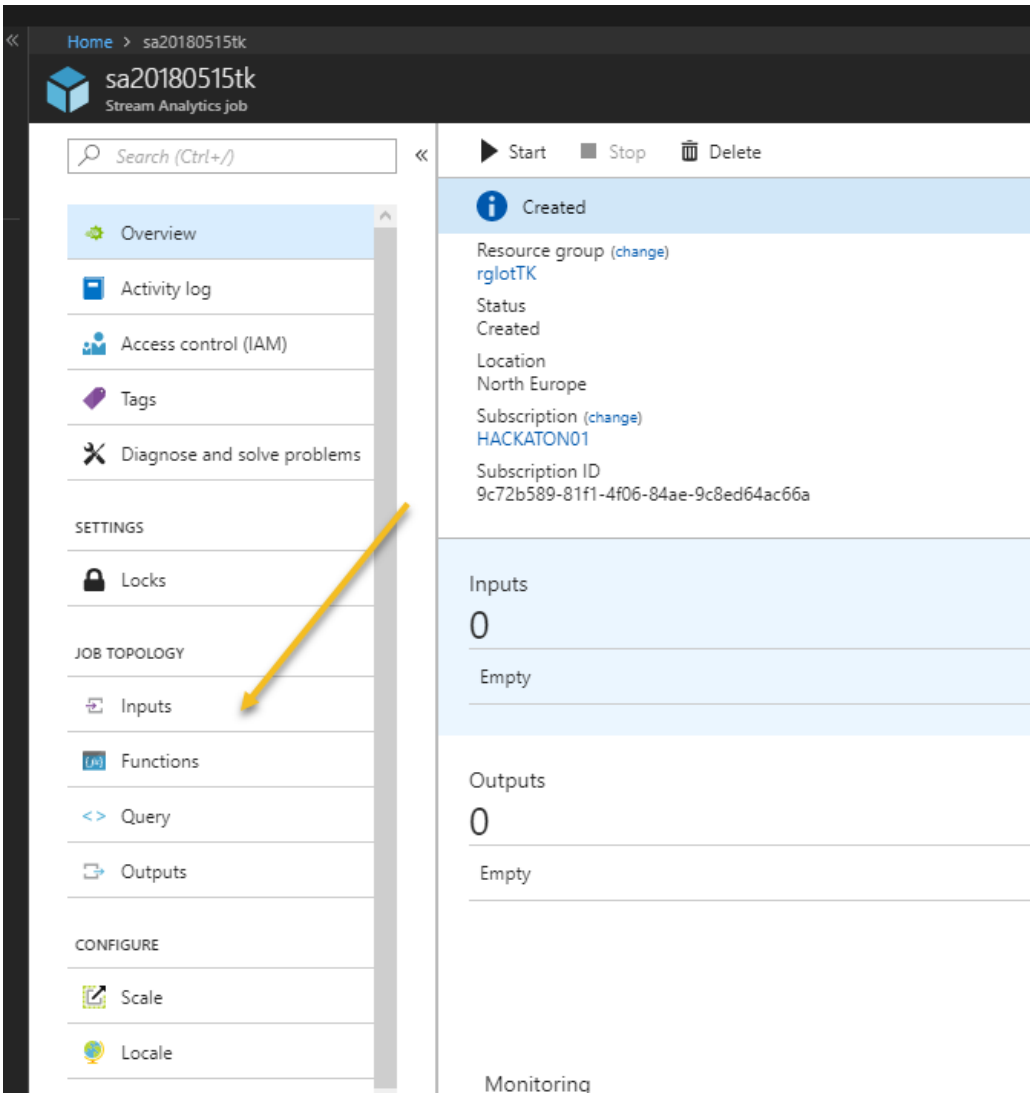


Rysunek 31 Tworzenie Stream Analytics

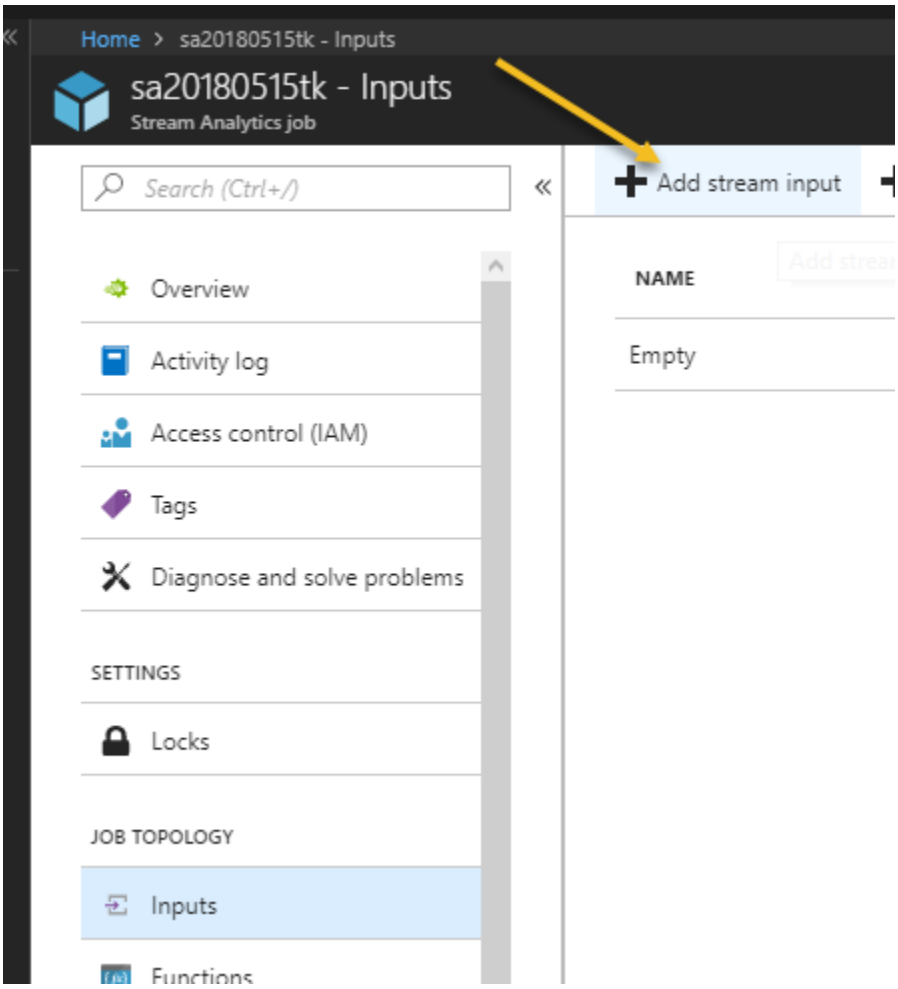


Rysunek 32 Stream Analytic Job - ta sama RG, nazwa unikalna

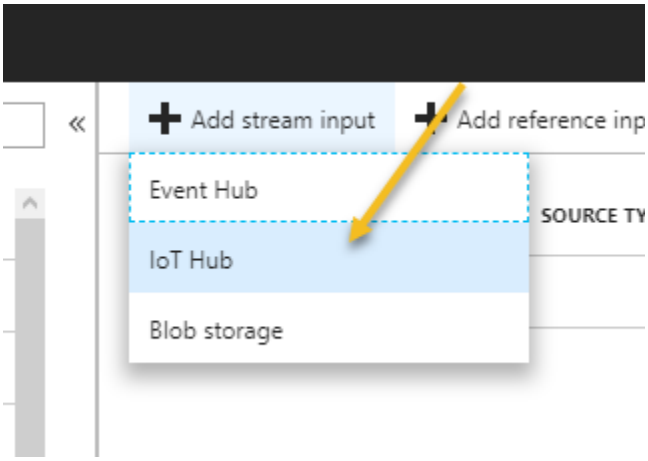
Rejestracja IoT Hub jako wejścia



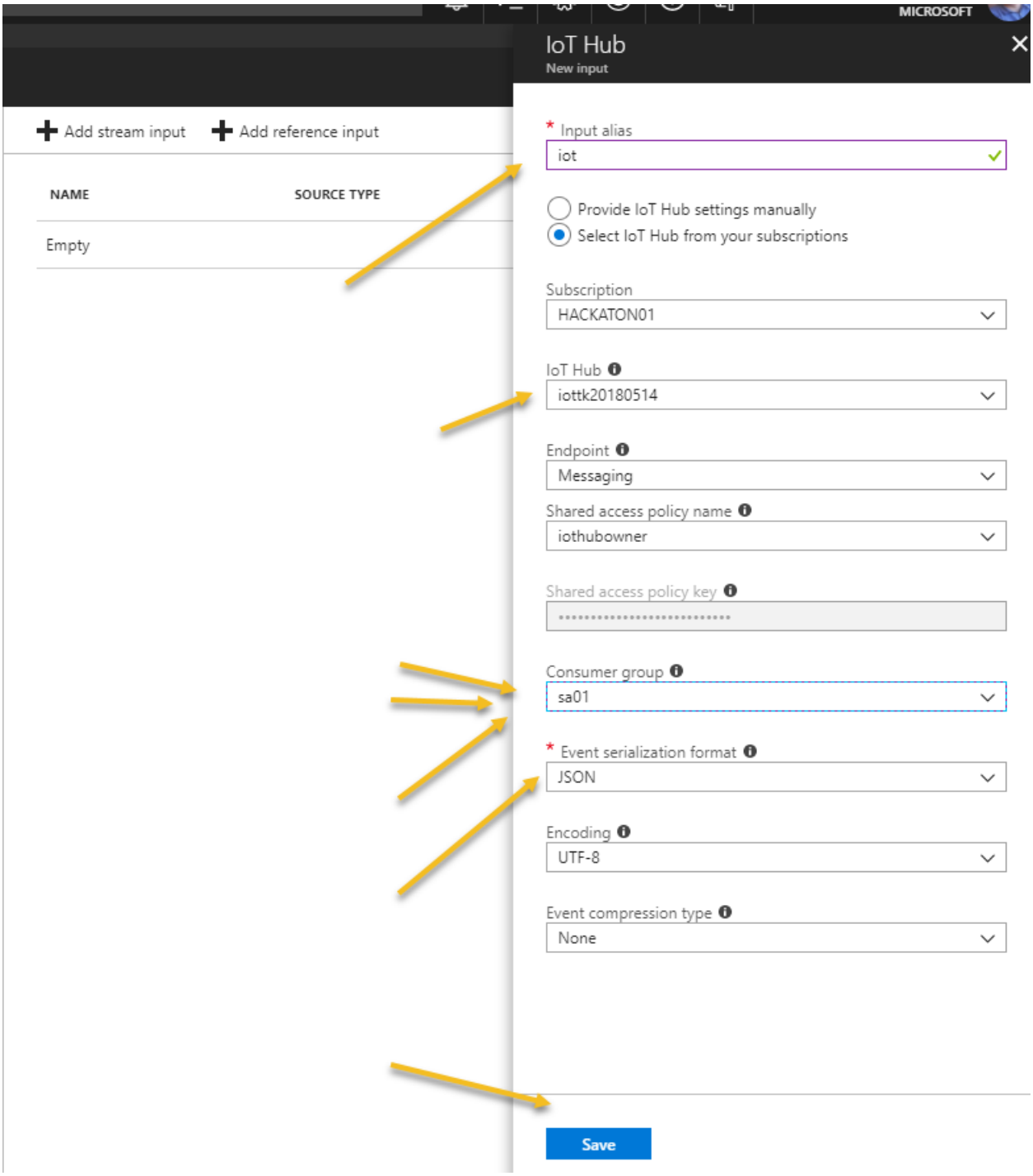
Rysunek 33 Rejestracja IoT hub jako wejścia



Rysunek 34 Rejestracja IoT Hub jako wejścia

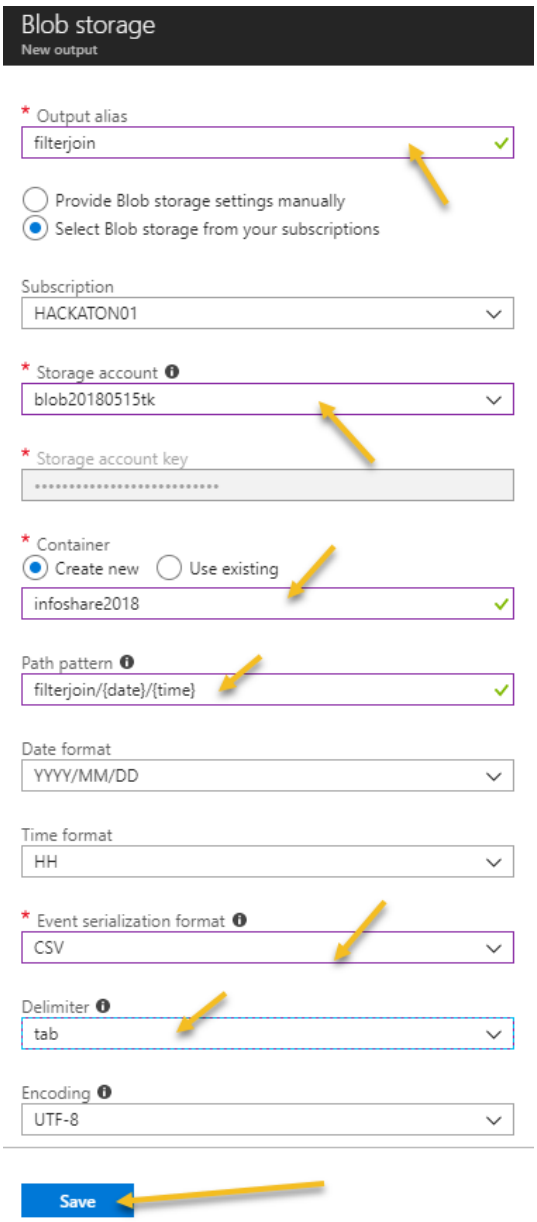


Rysunek 35 Rejestracja IoT Hub jako wejścia



Rysunek 36 Parametry wejścia - ważne - Consumer Group

Blob - filterjoin



Rysunek 37 Parametry wyjścia - blob (analogicznie inne)

Container: **infoshare2018**

Path pattern: **filterjoin/{date}/{time}**

Date format: **YYYY/MM/DD**

Time format: **HH**

Blob – rawdata

Podobnie dodajemy ujście o nazwie rawdata

Path pattern: **rawdata/{date}/{time}**

Blob – aggregate

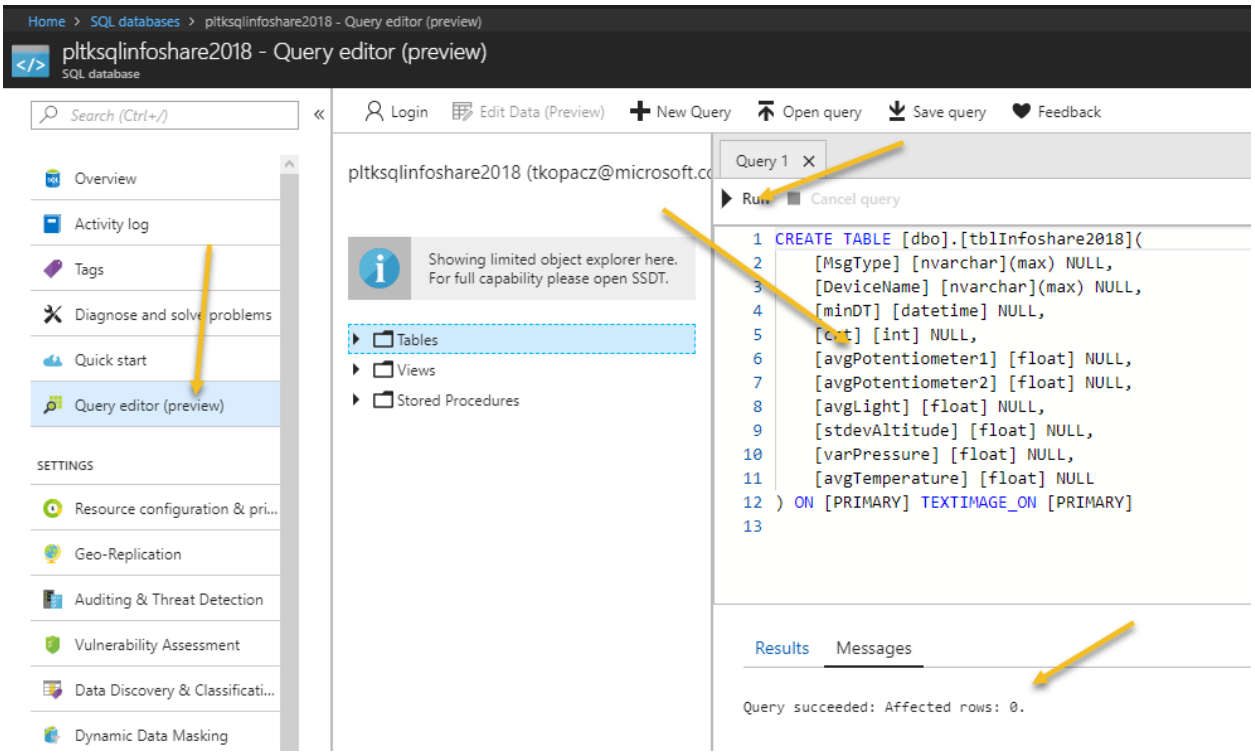
Podobnie dodajemy ujście o nazwie aggregate

Path pattern: **aggregate/{date}/{time}**

SQL – sql

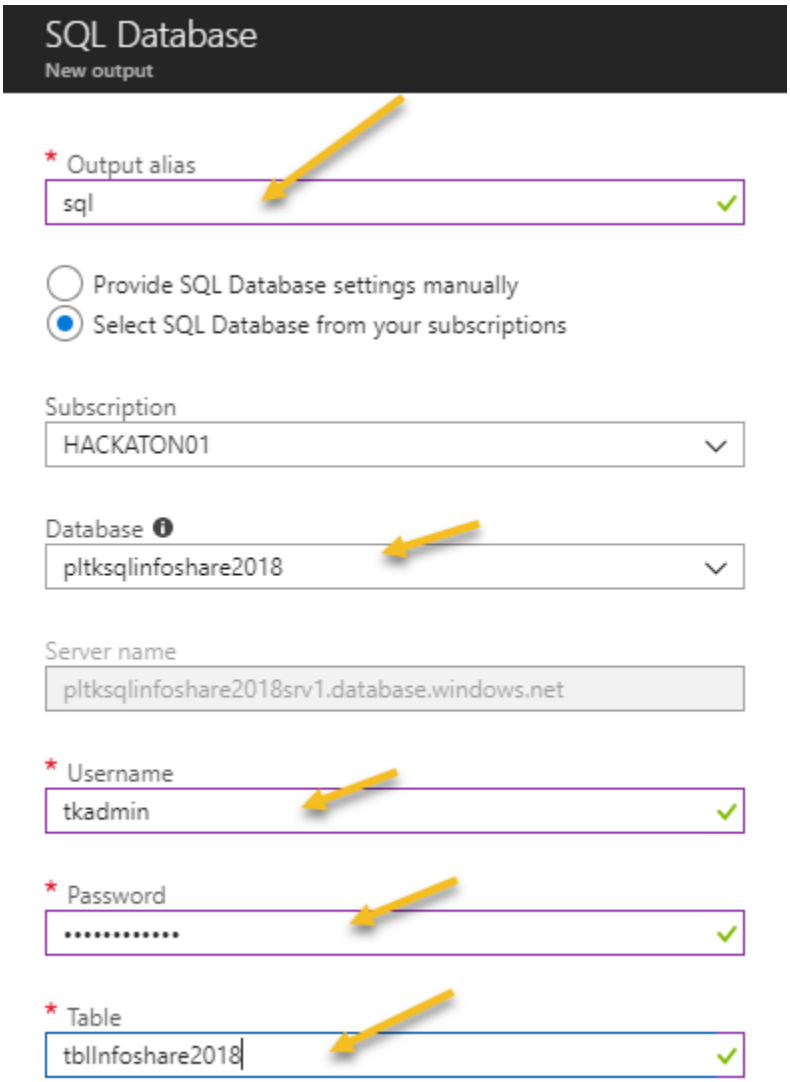
Najpierw należy w SQL Server utworzyć tabelę która będzie pasować do schematu kwerendy

```
CREATE TABLE [dbo].[tblInfoshare2018](
	[MsgType] [nvarchar](max) NULL,
	[DeviceName] [nvarchar](max) NULL,
	[minDT] [datetime] NULL,
	[cnt] [int] NULL,
	[avgPotentiometer1] [float] NULL,
	[avgPotentiometer2] [float] NULL,
	[avgLight] [float] NULL,
	[stdevAltitude] [float] NULL,
	[varPressure] [float] NULL,
	[avgTemperature] [float] NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```



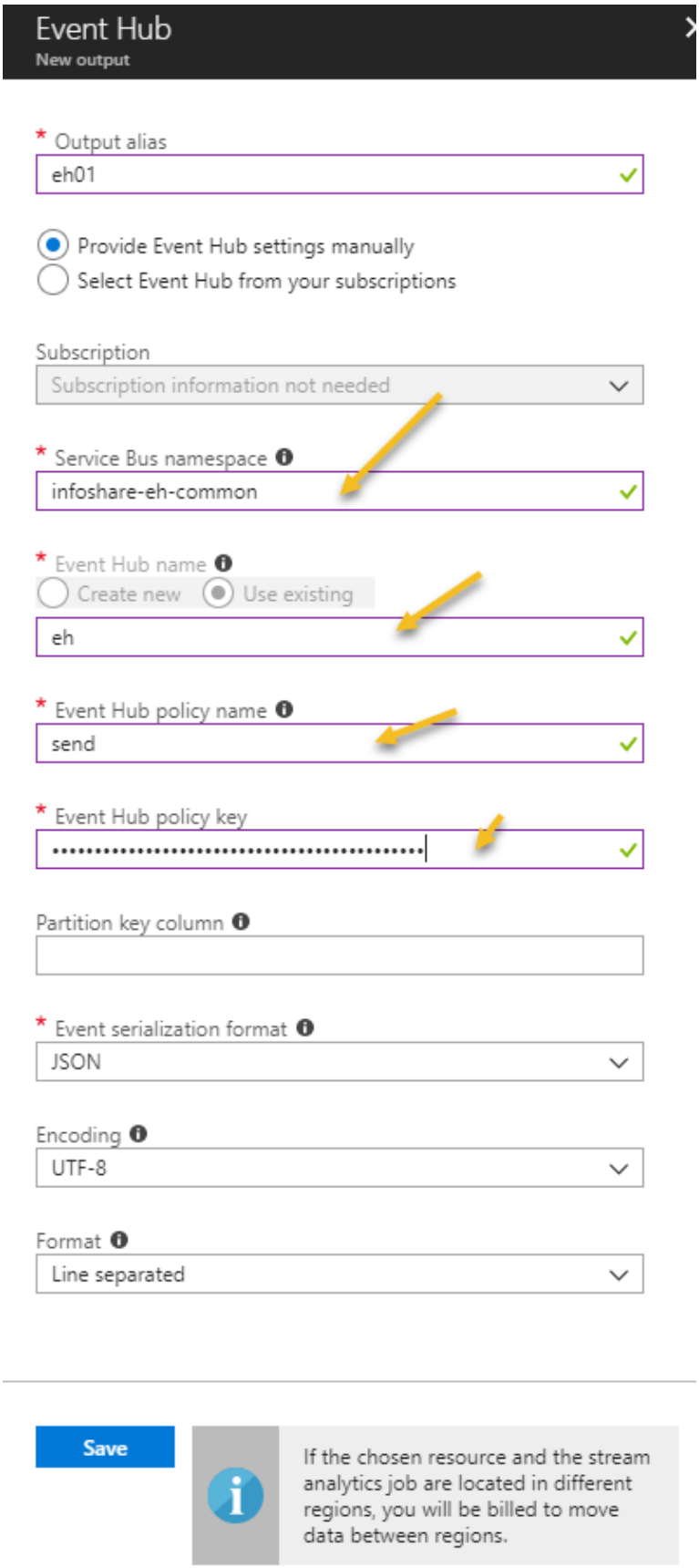
Rysunek 38 Tworzenie tabeli SQL przy użyciu Portalu

Rejestracja w Stream Analytics



Rysunek 39 Dodanie wyjścia do SQL Database

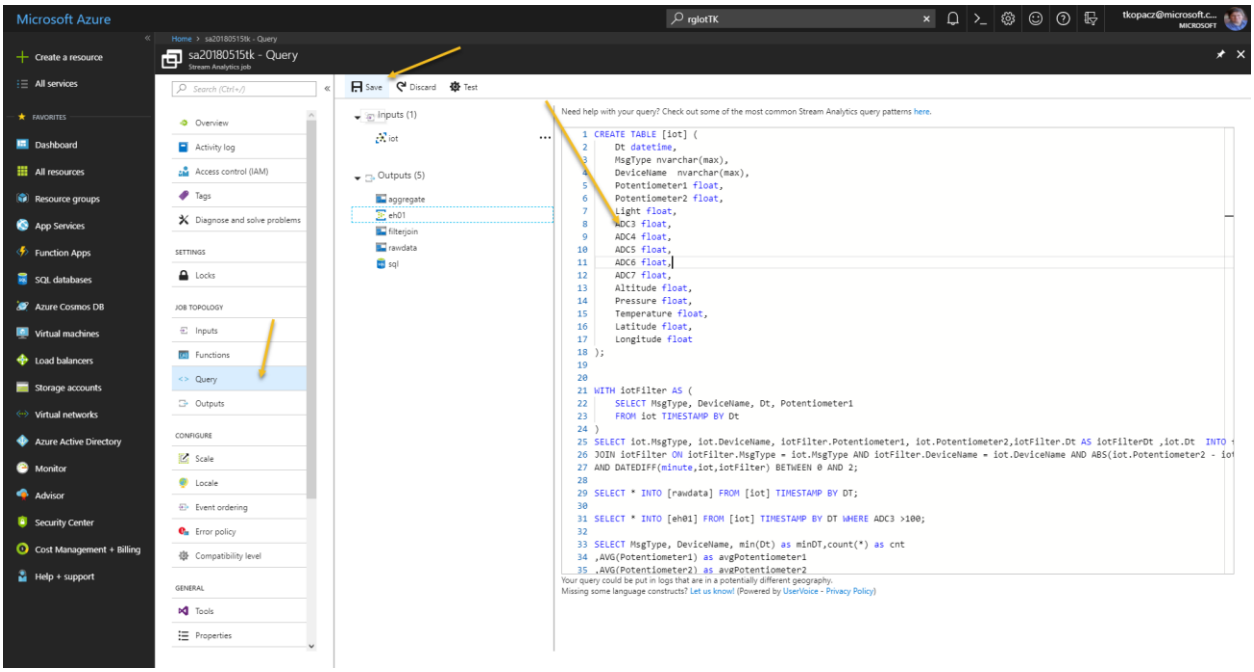
EventHub - Rejestracja wspólnego ujścia



Rysunek 40 Wspólne wyjście do Event Hub

Endpoint=sb://infoshare-eh-common.servicebus.windows.net/;SharedAccessKeyName=send;SharedAccessKey=ddyh8h1jz66PW///uRJJxUkLO/+3enq5LrXH5v71gtc=

Query:



Rysunek 41 Kwerenda w Stream Analytics

```
CREATE TABLE [iot] (
   Dt datetime,
   MsgType nvarchar(max),
   DeviceName  nvarchar(max),
   Potentiometer1 float,
   Potentiometer2 float,
   Light float,
   ADC3 float,
   ADC4 float,
   ADC5 float,
   ADC6 float,
   ADC7 float,
   Altitude float,
   Pressure float,
   Temperature float,
   Latitude float,
   Longitude float
);
```

```
WITH iotFilter AS (
   SELECT MsgType, DeviceName, Dt, Potentiometer1
   FROM iot TIMESTAMP BY Dt
)
SELECT iot.MsgType, iot.DeviceName, iotFilter.Potentiometer1, iot.Potentiometer2,iotFilter.Dt AS iotFilterDt ,iot.Dt
INTO filterjoin FROM iot TIMESTAMP BY Dt
JOIN iotFilter ON iotFilter.MsgType = iot.MsgType AND iotFilter.DeviceName = iot.DeviceName AND
ABS(iot.Potentiometer2 - iotFilter.Potentiometer1)<1
AND DATEDIFF(minute,iot,iotFilter) BETWEEN 0 AND 2;
```

*SELECT * INTO [rawdata] FROM [iot] TIMESTAMP BY DT;*
OR
*SELECT * INTO [eh01] FROM [iot] TIMESTAMP BY DT WHERE ADC3 >100;*

```
SELECT MsgType, DeviceName, min(Dt) as minDT,count(*) as cnt
,AVG(Potentiometer1) as avgPotentiometer1
,AVG(Potentiometer2) as avgPotentiometer2
,AVG(Light) as avgLight
,STDEV(Altitude) as stdevAltitude
,VAR(Pressure) as varPressure
,AVG(Temperature) as avgTemperature
INTO [aggregate] FROM [iot] TIMESTAMP BY DT GROUP BY MsgType, DeviceName, TumblingWindow(minute,10) ;

SELECT MsgType, DeviceName, min(Dt) as minDT,count(*) as cnt
,AVG(Potentiometer1) as avgPotentiometer1
,AVG(Potentiometer2) as avgPotentiometer2
,AVG(Light) as avgLight
,STDEV(Altitude) as stdevAltitude
,VAR(Pressure) as varPressure
,AVG(Temperature) as avgTemperature
INTO [sql] FROM [iot] TIMESTAMP BY DT GROUP BY MsgType, DeviceName, TumblingWindow(minute,10) ;
```

Uruchomić źródła i poczekać ;

Co dalej - rozważania?

Data processing in Event Grid and Web

Data processing in Azure Function

Edge processing