IoT Workshop

Goal of this workshop is to prepare CPU and Free RAM usage monitoring system for servers.

* We will get CPU usage every 1 second.
* All information will be stored in Cosmos DB for future analytics.
* We will have near real-time CPU usage visualization in Power BI.
* If we will not get data from computer longer than 30 seconds we will send alert using e-mail.

# Architecture

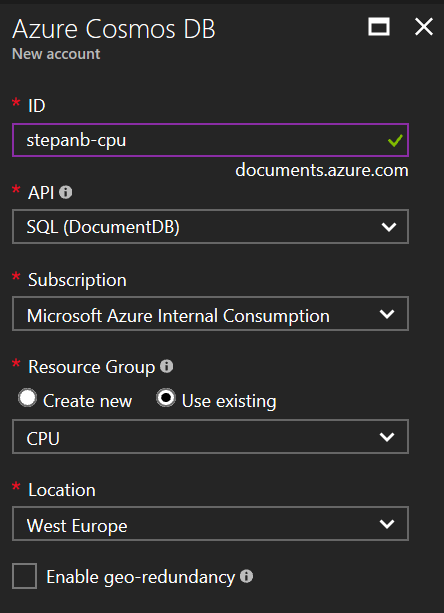


# Prerequisites

1. Azure subscription
2. Office 365 account
3. Visual Studio 2017 Community or better

# Prepare Azure infrastructure

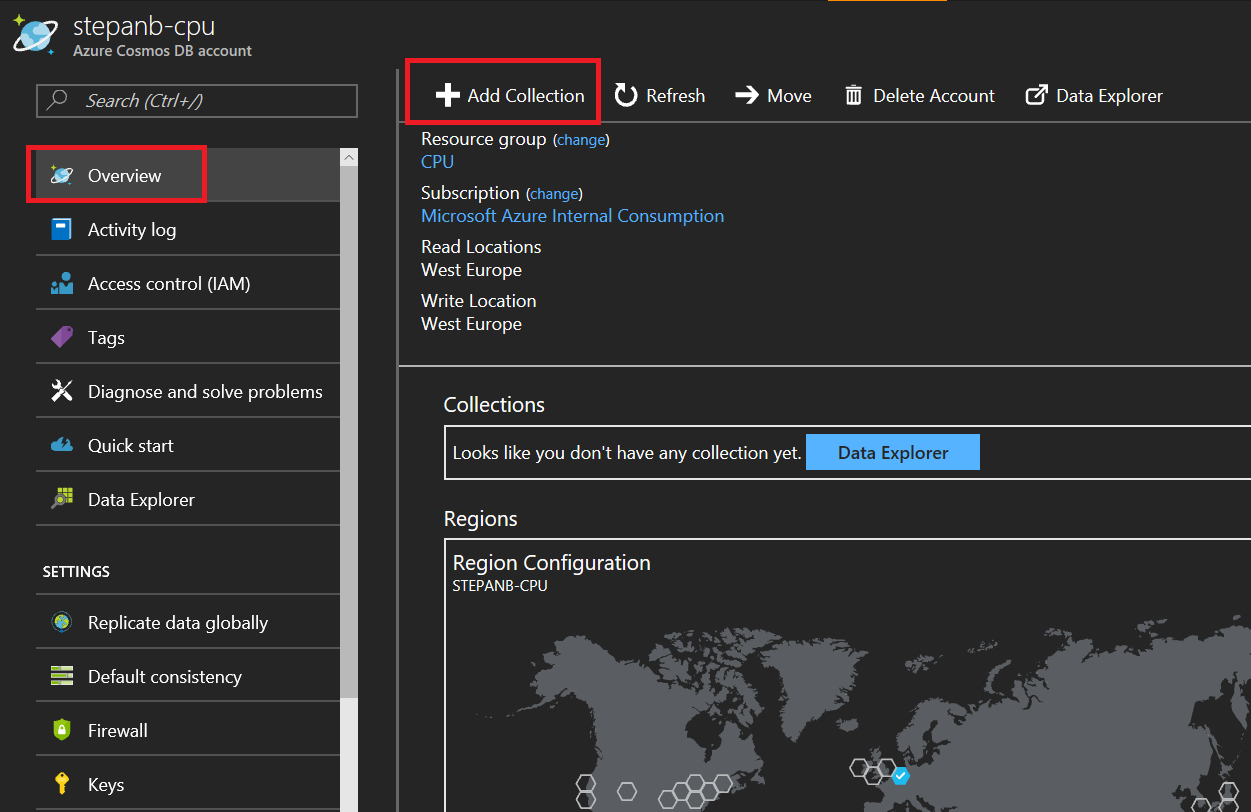
1. Go to Azure Portal.
2. Create new Resource group named *CPU* in West Europe region.
3. Put all new services to *CPU* Resource group.
4. Create new S1 IoT Hub. As name use your *alias*. Name must be worldwide unique. It is 3rd level domain name.
5. Create new Stream Analytics Job named *cpuanalytics*.
6. Create new Cosmos DB. As name use your *alias*. Name must be worldwide unique. It is 4th level domain name. Choose *SQL (DocumentDB)* as API.



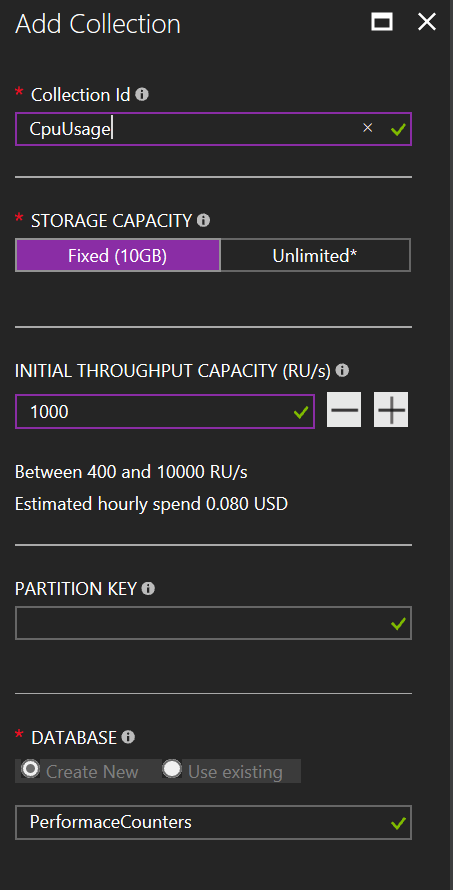
1. Create new Logic App named *Alert*.
2. Create new Service Bus. As name use your *alias*. Name must be worldwide unique. It is 4th level domain name.
3. Check if is your Power BI account working. Go to <https://powerbi.microsoft.com/> and Sign In using your Office365 account.

# Configure Cosmos DB

1. Open Cosmos DB from *CPU* Resource group.
2. Go to *Overview* and click *Add Collection*

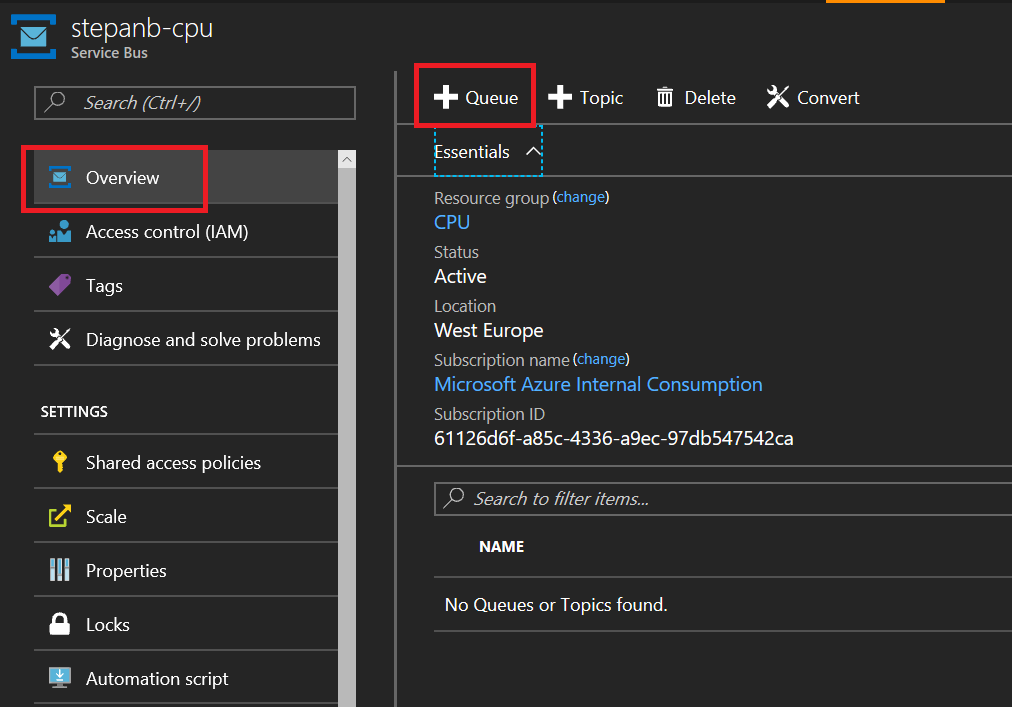


1. Create new collection. Collection Id: *CpuUsage*, DATABASE (New): *PerformaceCounters*.

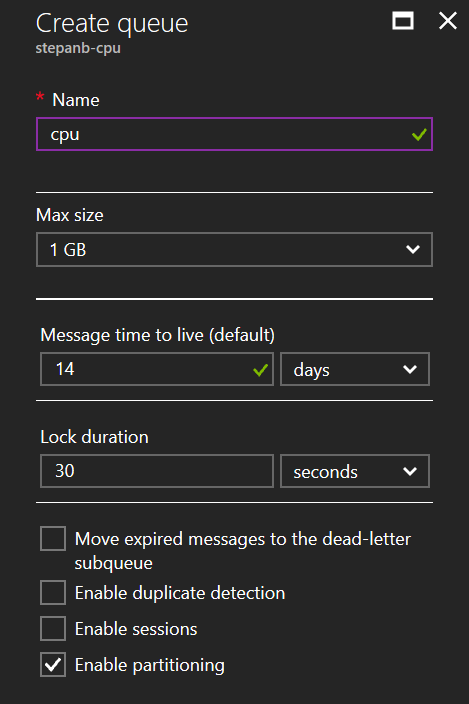


# Configure Service Bus

1. Open Service Bus from *CPU* Resource group.
2. Go to *Overview* and click *+ Queue*



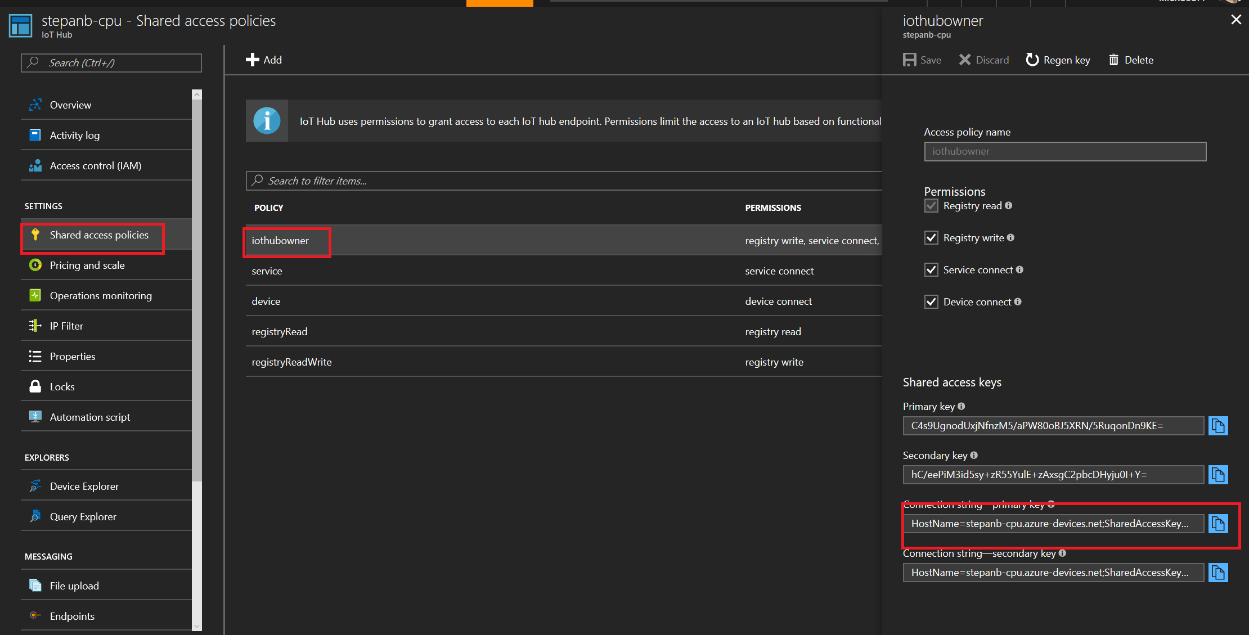
1. Use *cpu* as name.



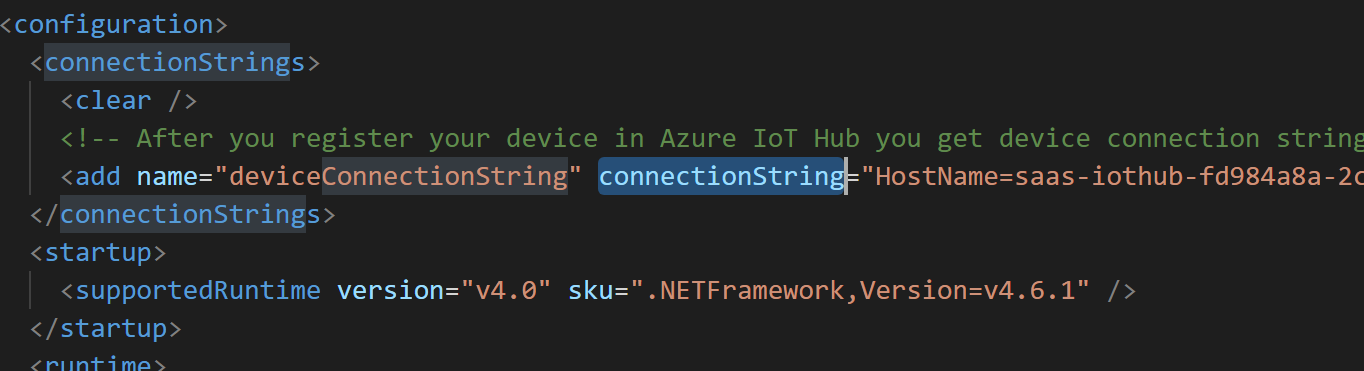
# Application

Application reads every 1 second CPU usage and sends it to Azure IoT Hub.

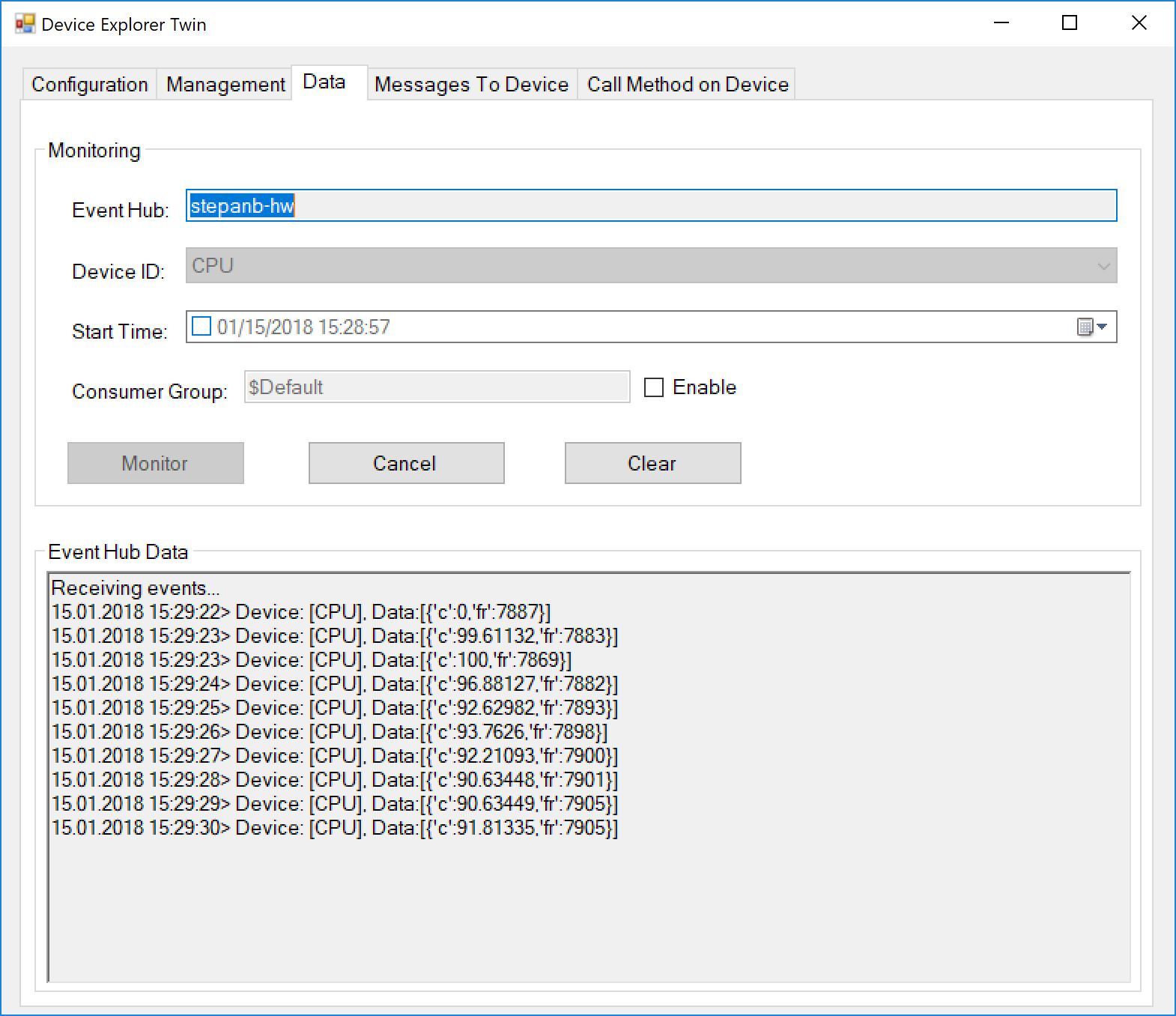
1. Install *Device Explorer* tool. It is debugging and device management tool for IoT Hub. <https://github.com/Azure/azure-iot-sdks/releases/download/2016-11-17/SetupDeviceExplorer.msi>
2. Open your IoT Hub from *CPU* Resource group.
3. Go to *Shared access policies* and click *iothubowner* policy. Copy connection string to clipboard.



1. Run Device Explorer and paste connection string to *IoT Hub Connection String* field and click *Update*.
2. Go to *Management* tab and click *Create* button. Use your computer name as *Device Id* name and click Add. You will see your device in list.
3. Right click line with *your* device and click *Copy connection string for selected device*.
4. Open *CpuUsage2Azure* solution using Visual Studio
5. Open *App.config*.
6. Paste device connection string to *connectionString* setting and save file.



1. Open *Program.cs*. You can see how to use object *DeviceClient* to communicate with Azure IoT Hub. *DeviceClient* is from [Device SDK](https://github.com/Azure/azure-iot-sdks).
2. Run application.
3. Go back to *Device Explorer* application a go to *Data* tab. Choose your device in *Device ID* list and click *Monitor*. You should see messages from your device. Those messages are from cloud now.

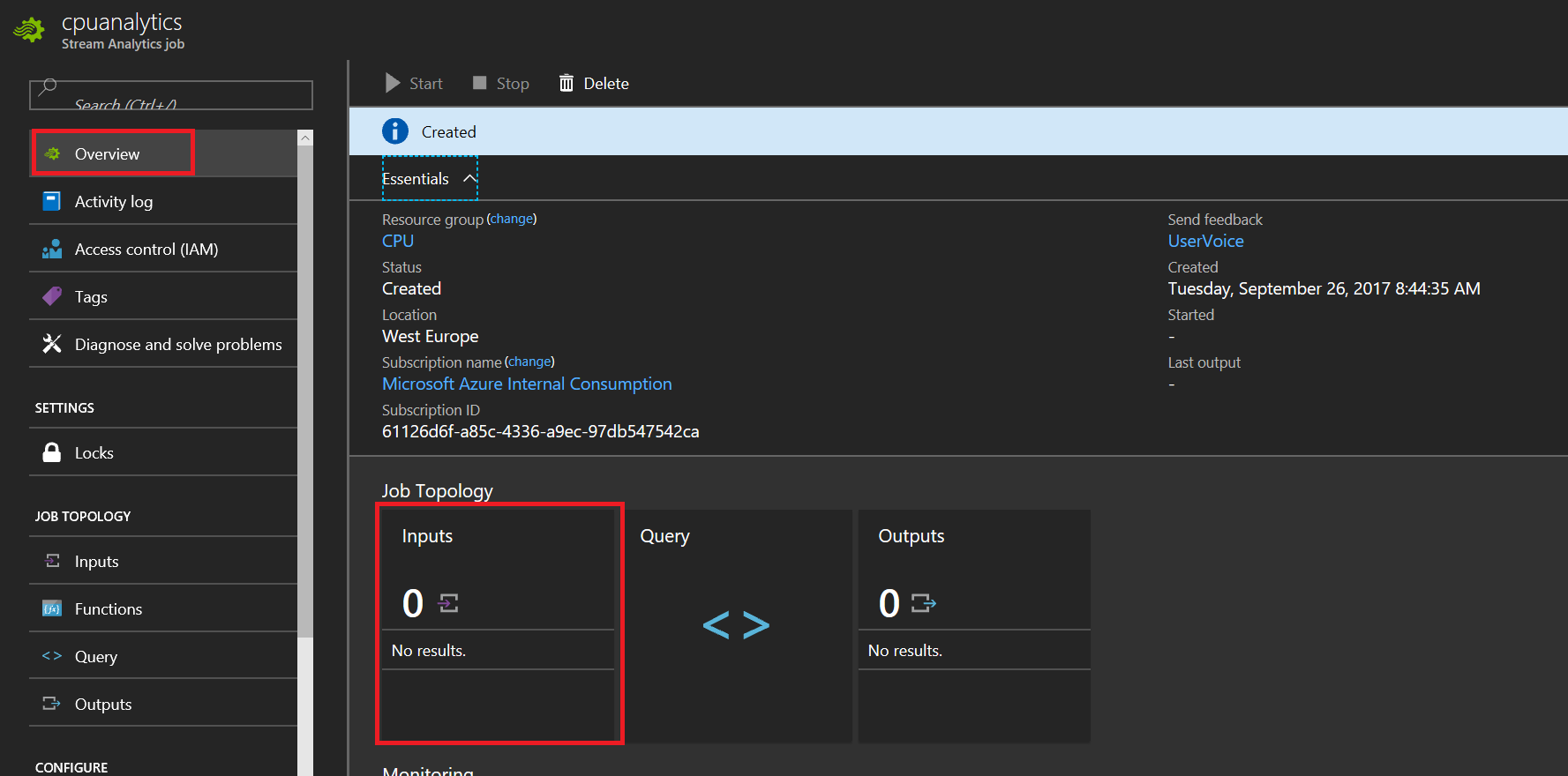


1. Keep it running.

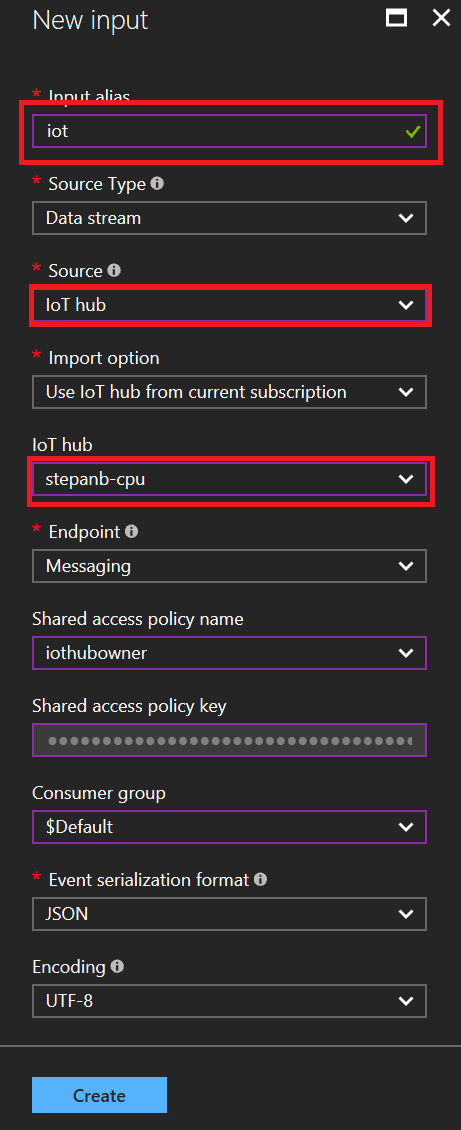
# Put services together

## Store raw data to Cosmos DB

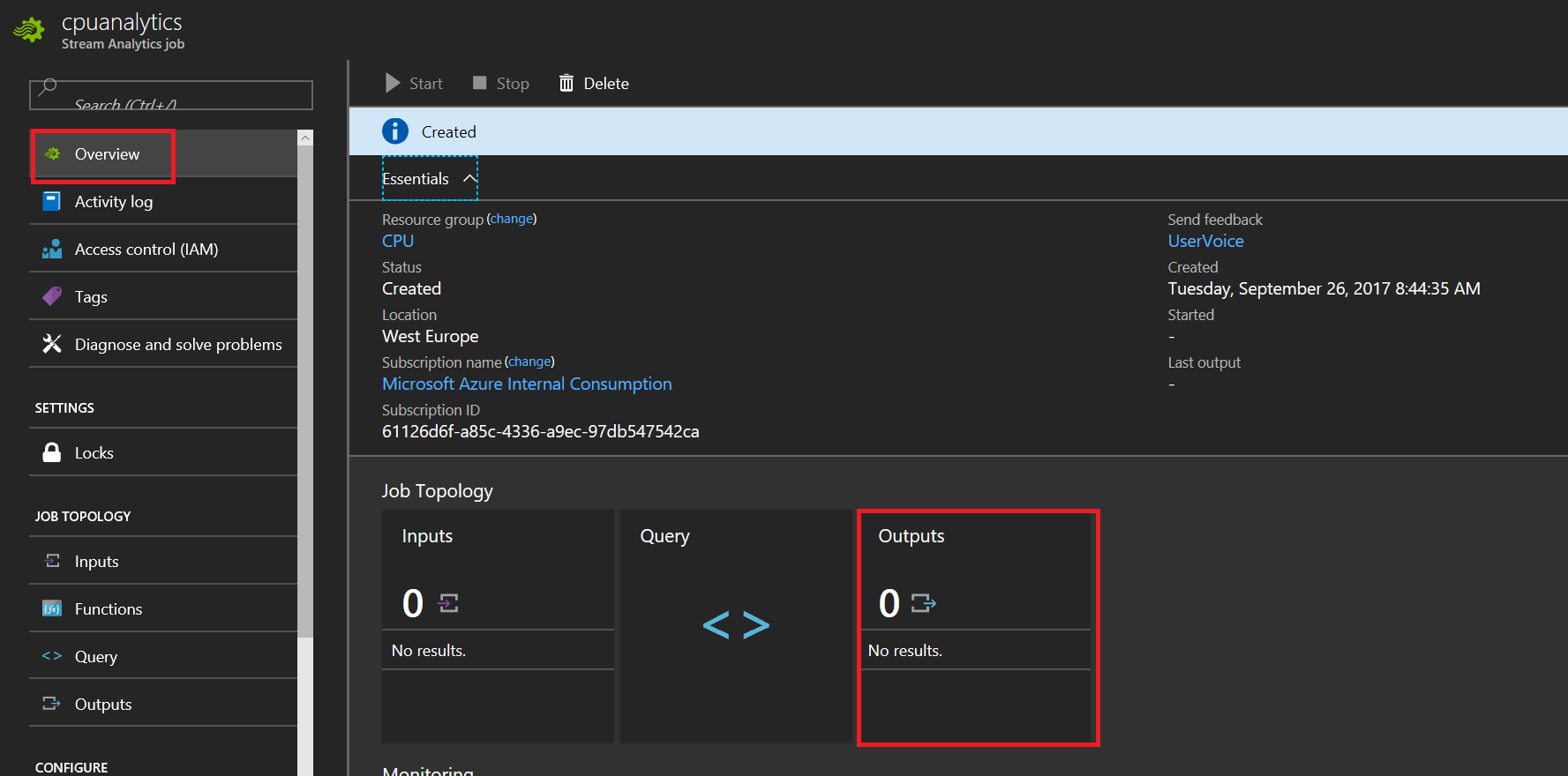
1. Go to Stream Analytics Job in *CPU* resource group.
2. Go to *Overview* and click *Inputs*.



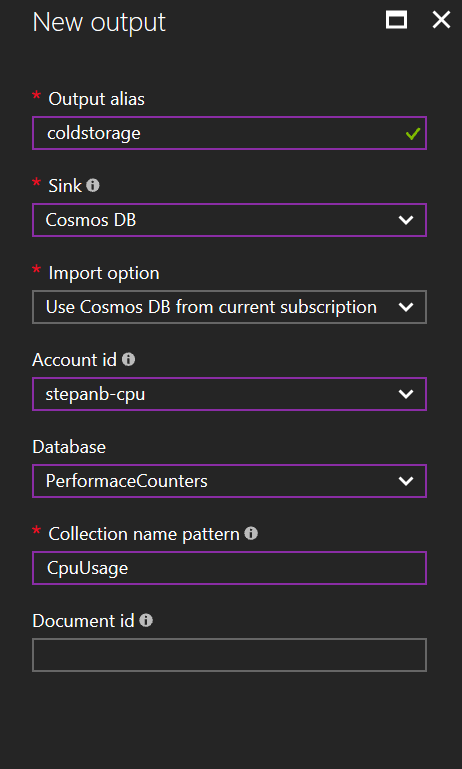
1. Click *Add*. Choose *Source* as IoT Hub and find your *IoT Hub* in list. Named it *IoT*.



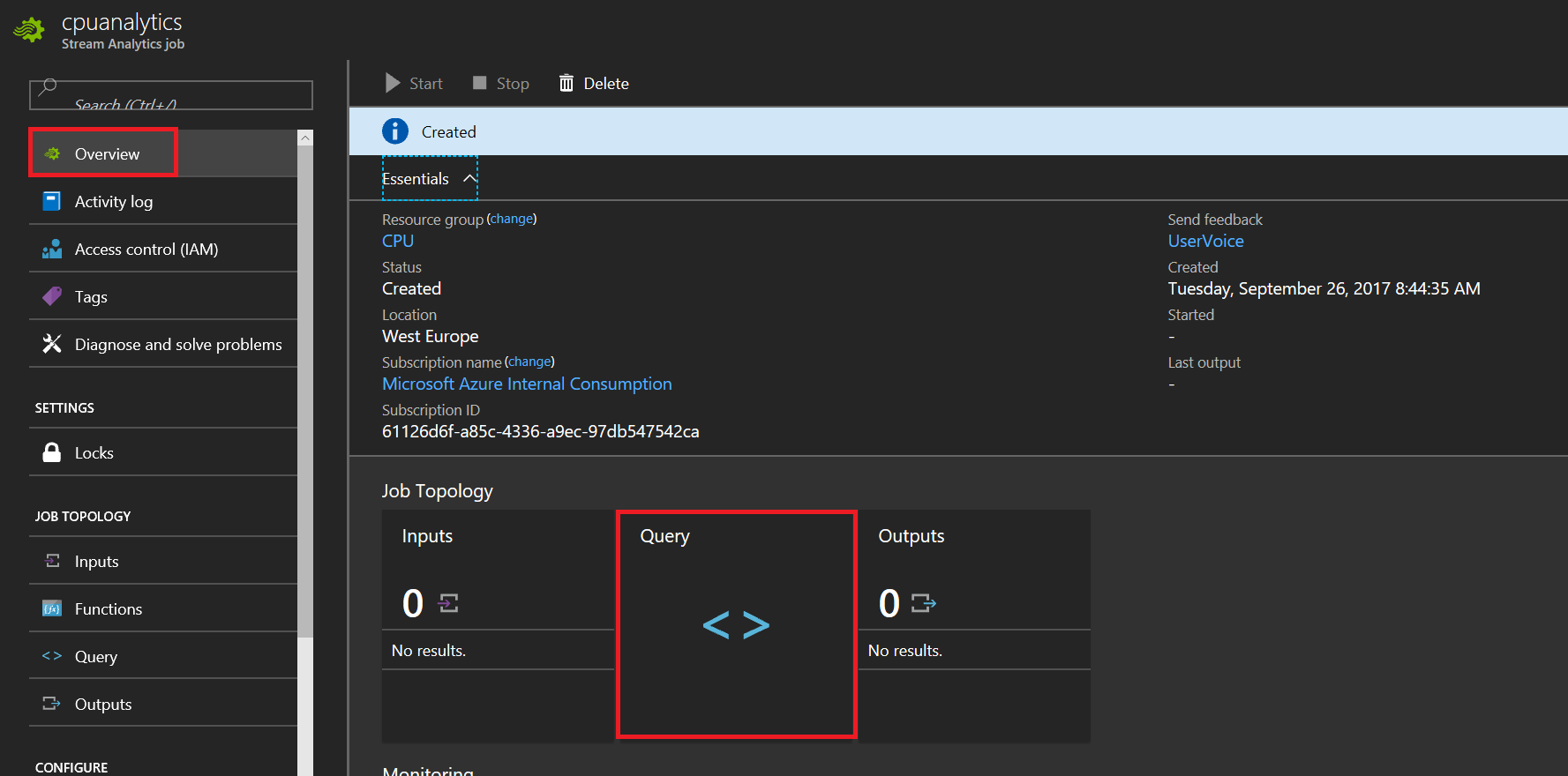
1. Click *Create*.
2. Go to *Overview* and click *Outputs*.



1. Click *Add*. Named new input as *coldstorage*. Use Cosmos DB as *Sink* and choose your Cosmos DB database and collection we created before.

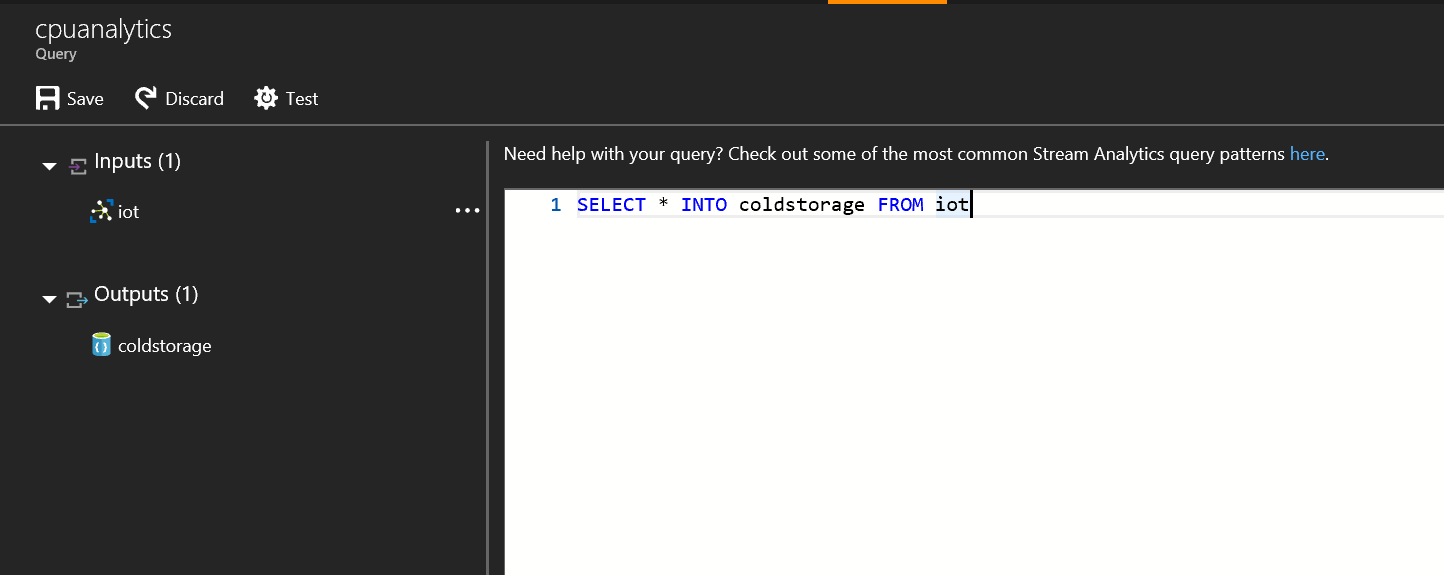


1. Click *Create*.
2. Go to *Overview* and click *Query*.

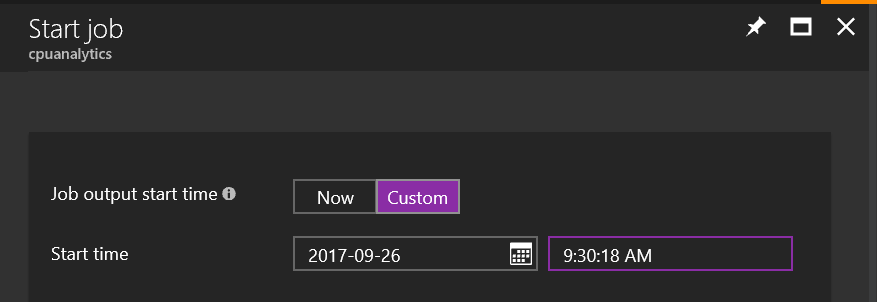


1. We want to store all data for future use. For example for Machine Learning. We will use basic quesry. All from input we send to output. Don’t forget to click *Save* button.

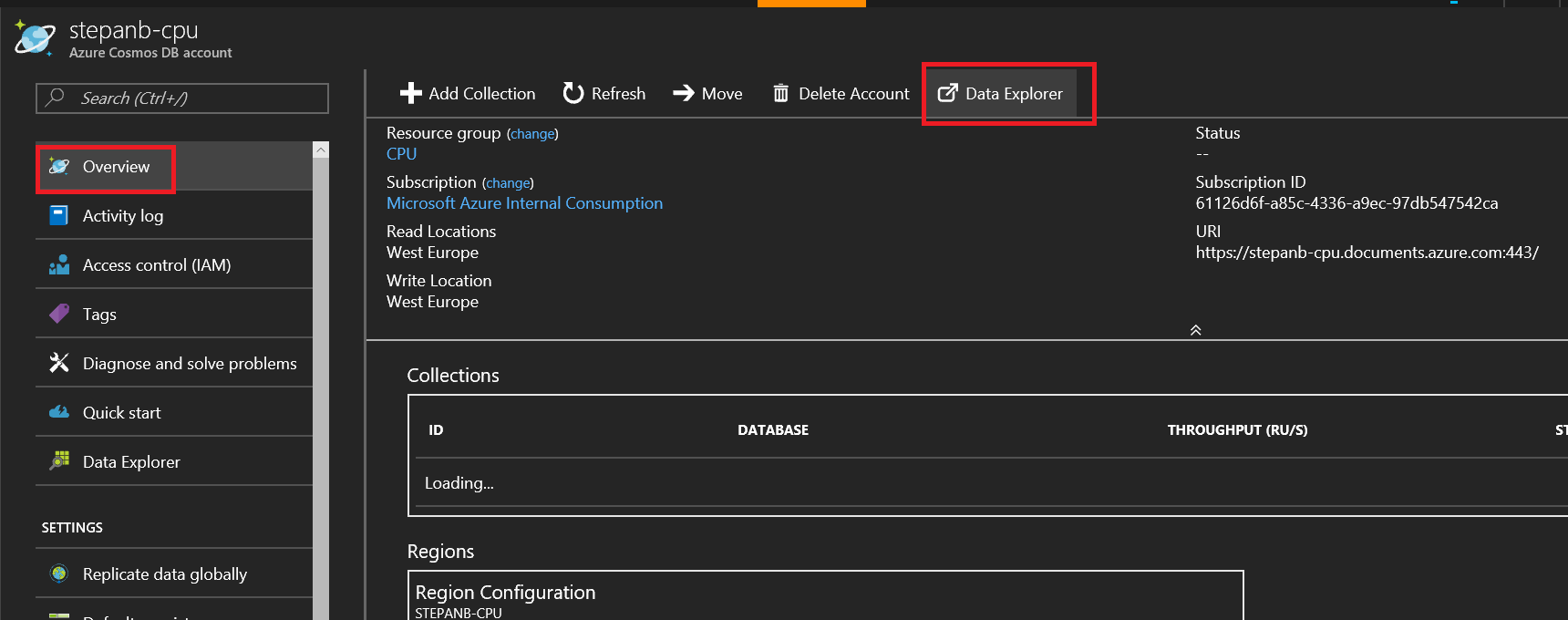
SELECT \* INTO coldstorage FROM iot



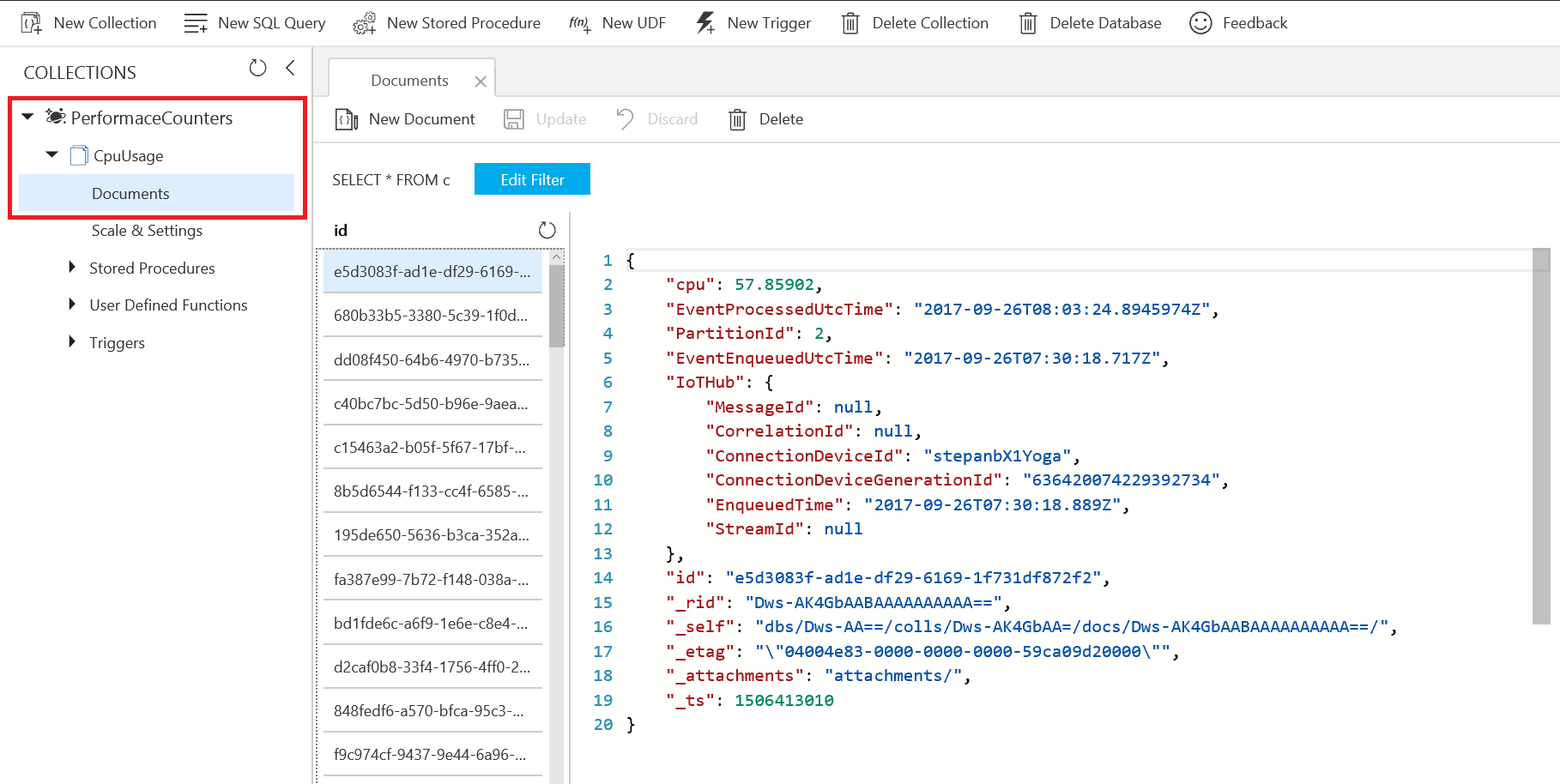
1. Go back to *Overview* and click *Start*. You can start job Now or in given time. IoT Hub works like buffer, so you can get data from past. It takes few minutes to run it.



1. If job started successfully go to Cosmos DB *Overview* and open *Data Explorer*.

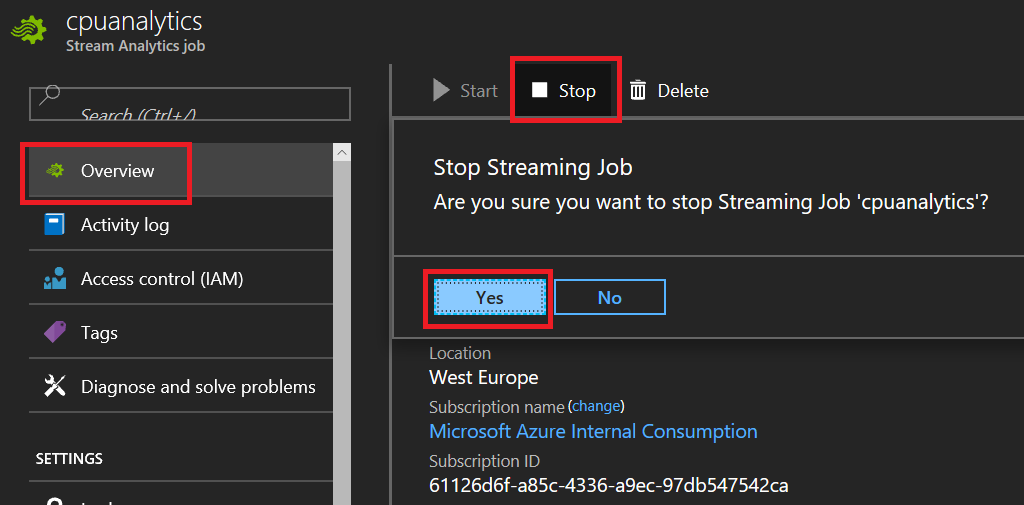


1. Drill down to your collection.

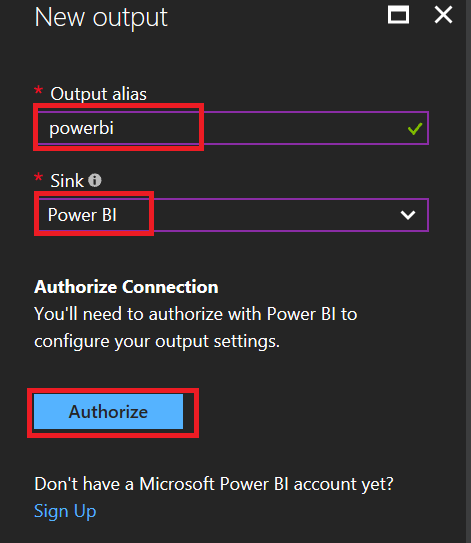


## Near real-time data visualization in Power BI

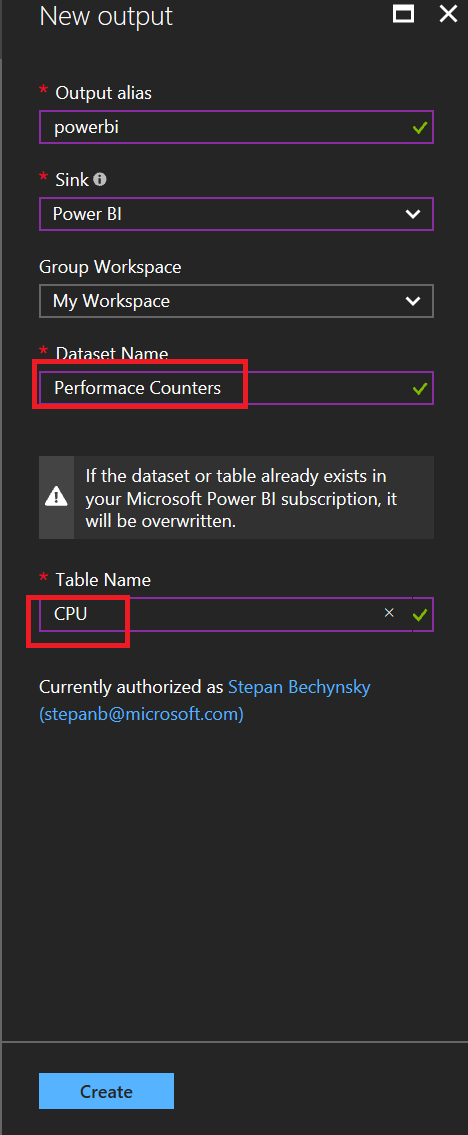
1. Go to Stream Analytics Job and stop it. Keep application reading CPU usage running.



1. Wait for Stream Analytics Job stopped.
2. Add new *Output*. *Name*: powerbi, *Sink*: Power BI.



1. Click *Autorize* button and long in using your Power BI credentials.
2. After you authorize Power BI you need to fill *Dataset Name* and *Table Name*.



1. Click create.
2. Go to *Query*.
3. Add new query. Do not delete existing one. To Power BI we will send just computer name (Device ID), time stamp (added by IoT Hub) and CPU usage without decimal points.

SELECT

EventEnqueuedUtcTime as EventTime,

IoTHub.ConnectionDeviceId as ComputerName,

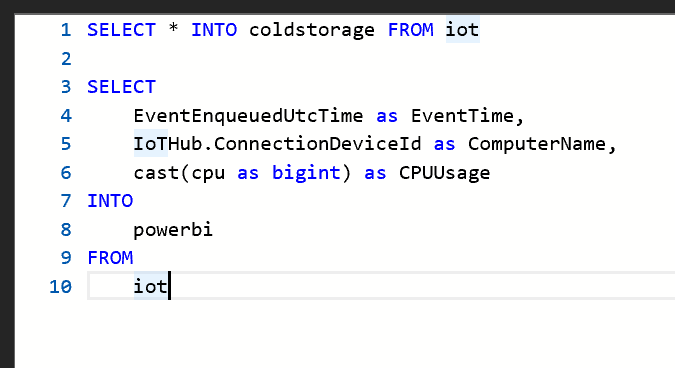
cast(c as bigint) as CPUUsage

INTO

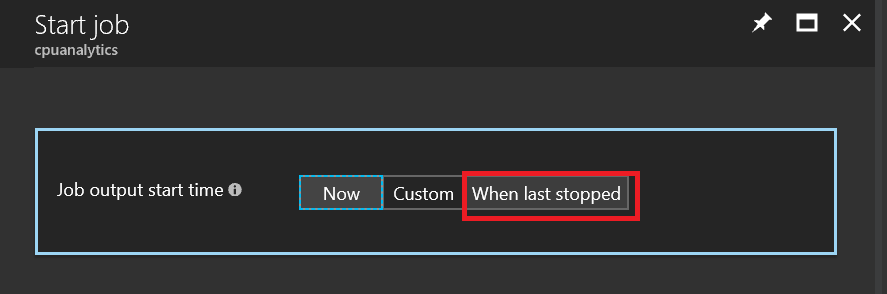
powerbi

FROM

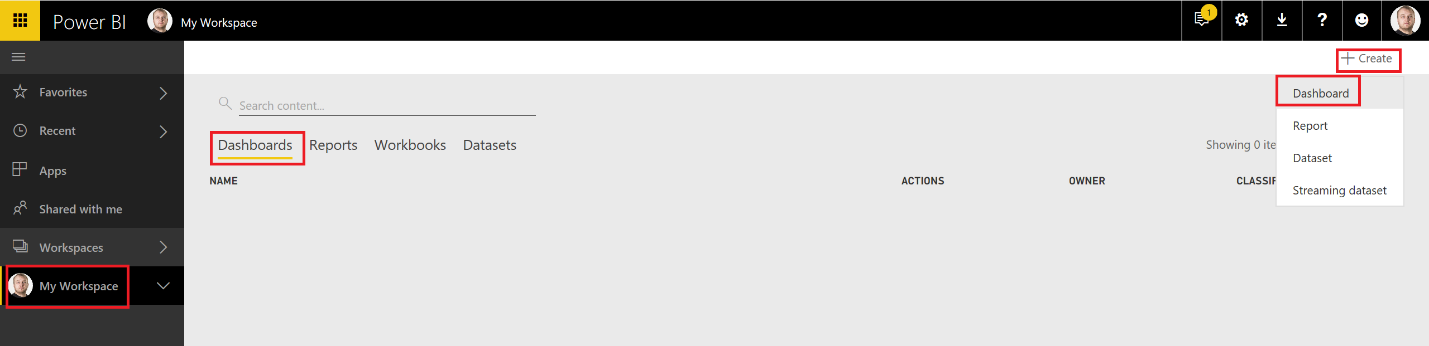
Iot



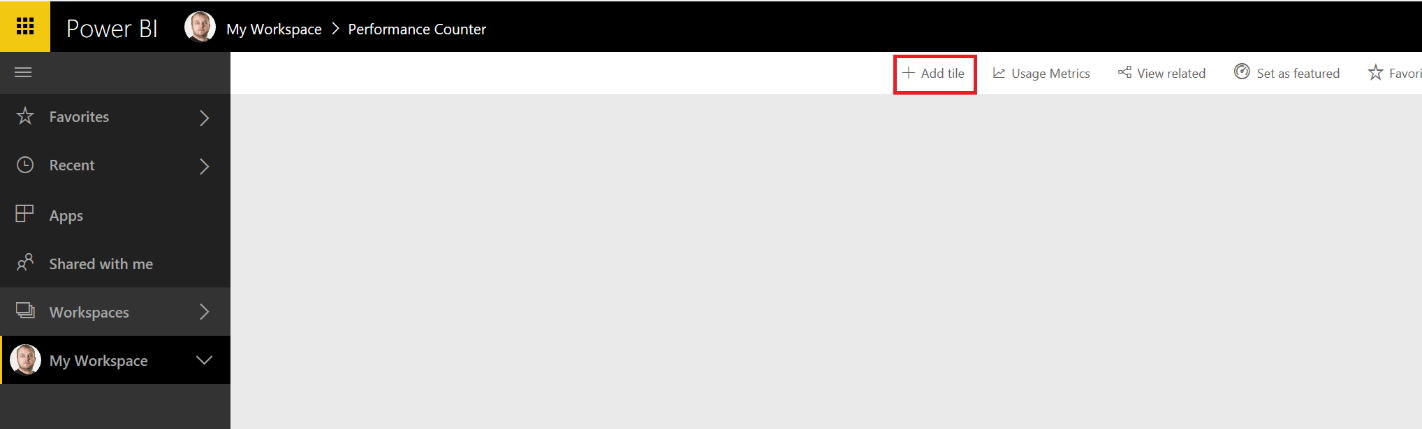
1. Save Query
2. Run Stream Analytics Jon. Now you have new option – *When Last Stopped*. Use this one. Remember. IoT Hub is buffer.



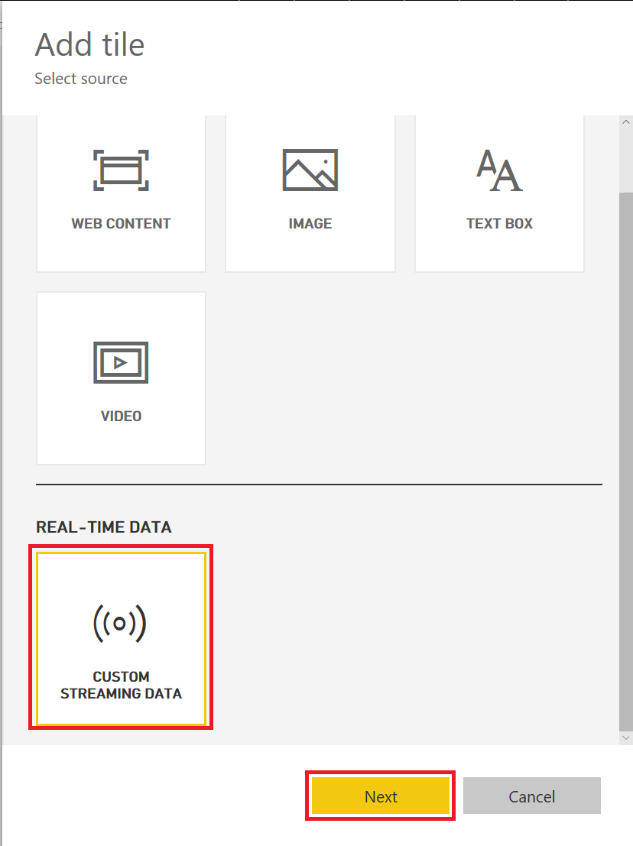
1. After Stream Analytics Job starts go to Power BI.
2. Open My Workspace – Dashboards and click Create – Dashboard.



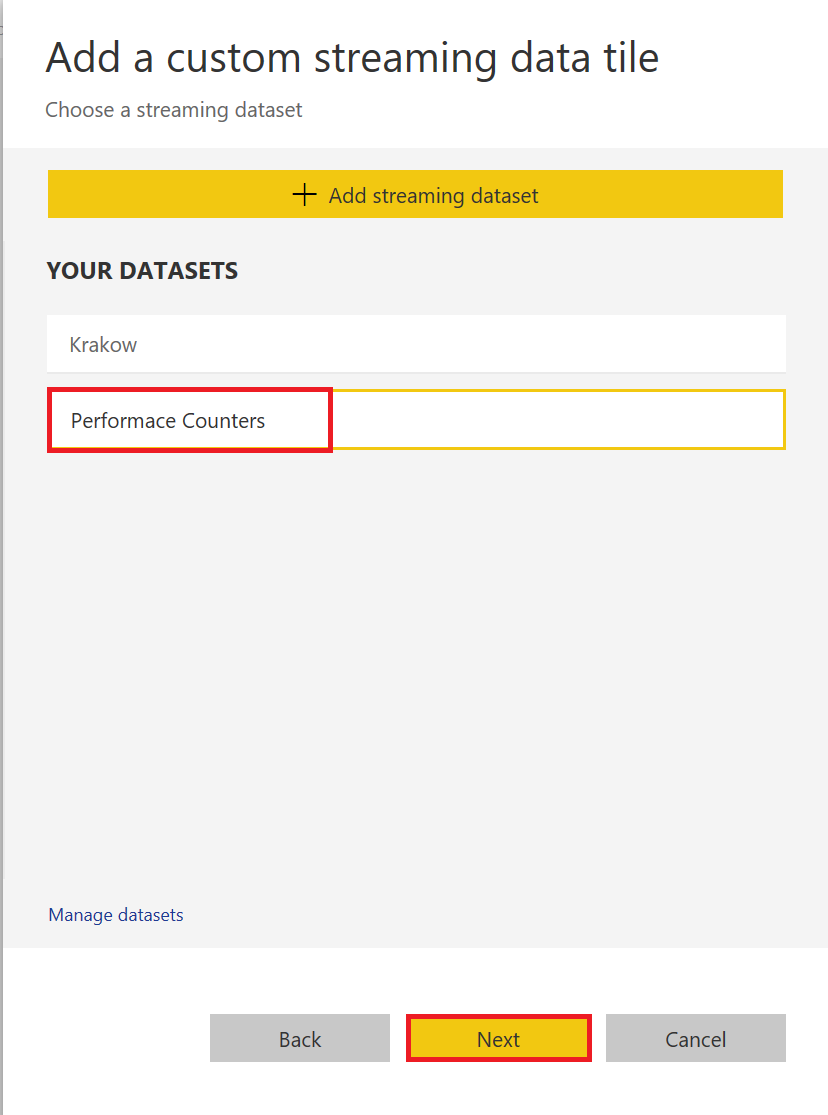
1. Name it *Performance Counter*
2. Click new Dashboard to open it.
3. Click *Add Tile*



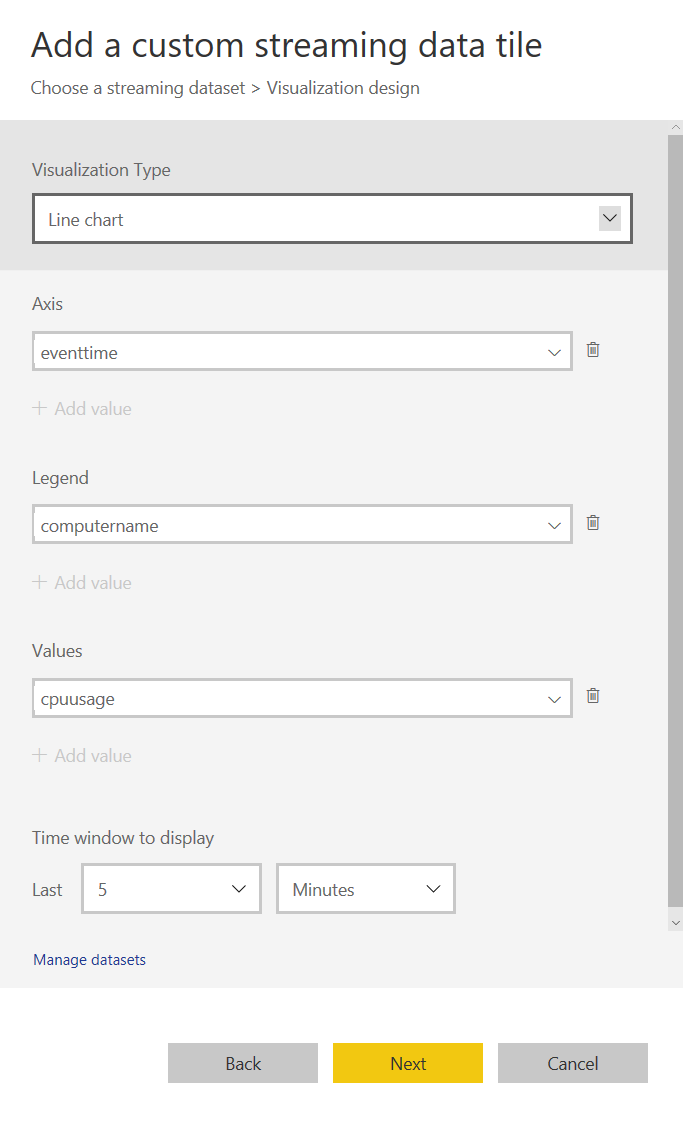
1. Choose *Custom Streaming Data* and click *Next*.



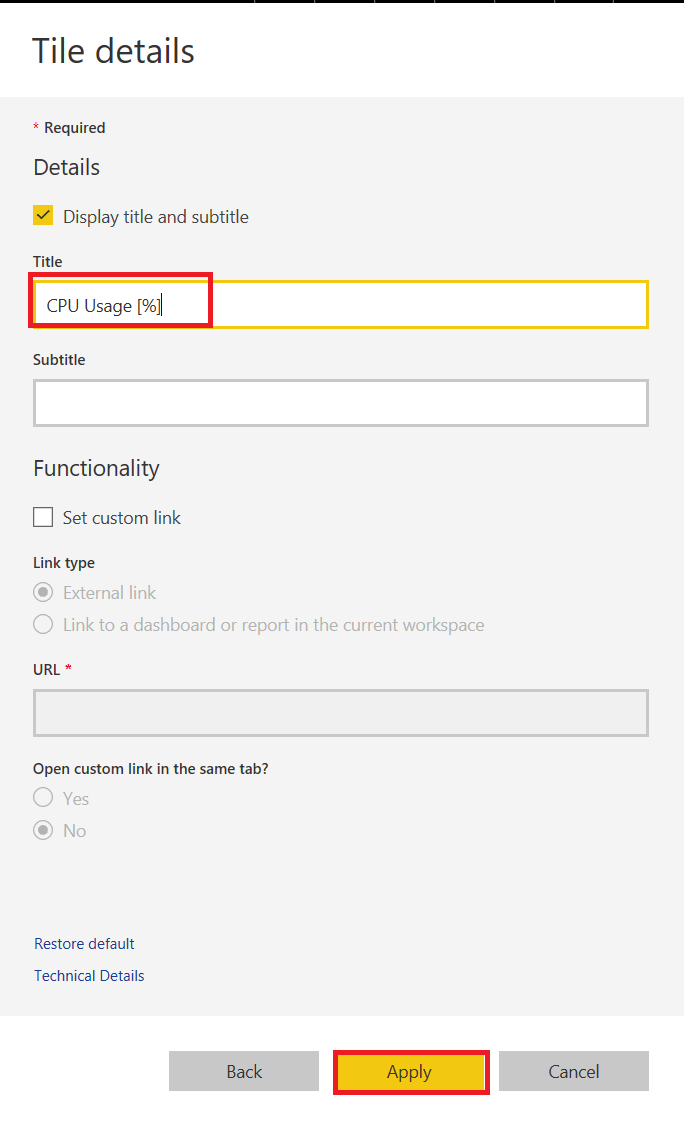
1. Choose your dataset from Stream Analytics Job and click *Next*.



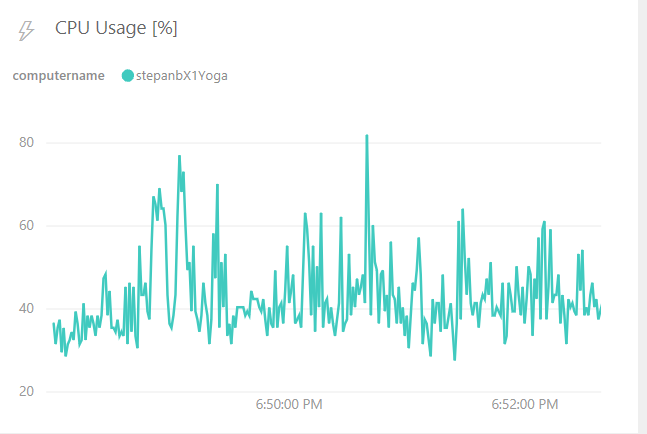
1. Crete chart for visualization. For example, *Line chart* with following settings.



1. Click *Next*.
2. Fill *Title* and click Applay.

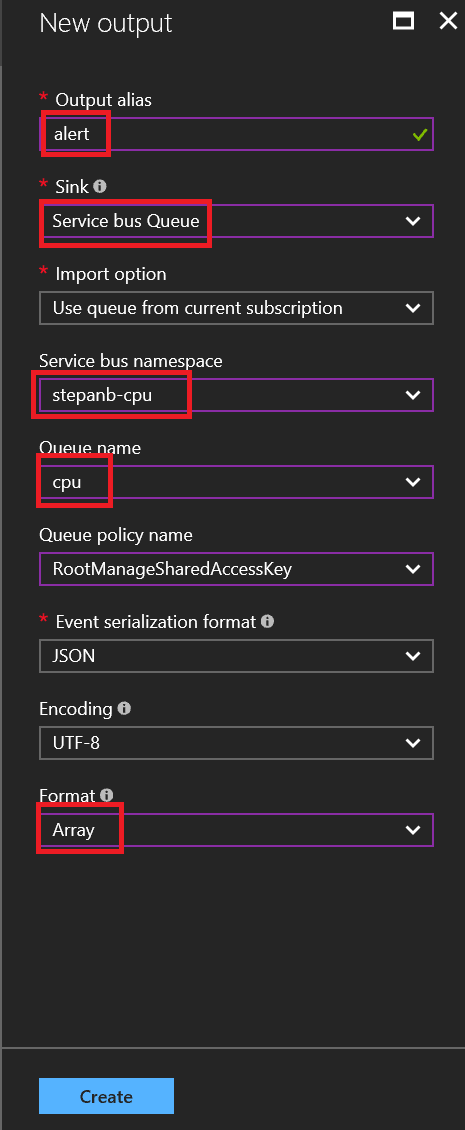


1. Now you should see almost real-time CPU usage of your computer. There is delay ~2-3 seconds thanks to Stream Analytics and Power BI.



## Send alert if we have no data from computer

1. Go to Stream Analytics Job and stop it. Keep application reading CPU usage running.
2. Wait for Stream Analytics Job stopped.
3. Add new *Output*. *Name*: powerbi, *Sink*:Service bus Queue. Choose queue we created before.



1. Click create.
2. Go to *Query*.
3. Now we will use [Windowing](https://msdn.microsoft.com/en-us/library/azure/dn835019.aspx) functionality of Stream Analytics Job. We will open time window for 30 seconds every 15 seconds and count number of messages. If everything is OK we should get 30 messages. We will fire alert if we get less than 20 messages. It is easy to simulate. We just shut down our application. In real usage we will probably check average usage of CPU to predict protentional problems.
4. Add new query. Do not delete existing one.

SELECT System.TimeStamp AS WindowEnd, 'alert' as Message, IoTHub.ConnectionDeviceId as ComputerName, Count(\*) as NumberOfMessages

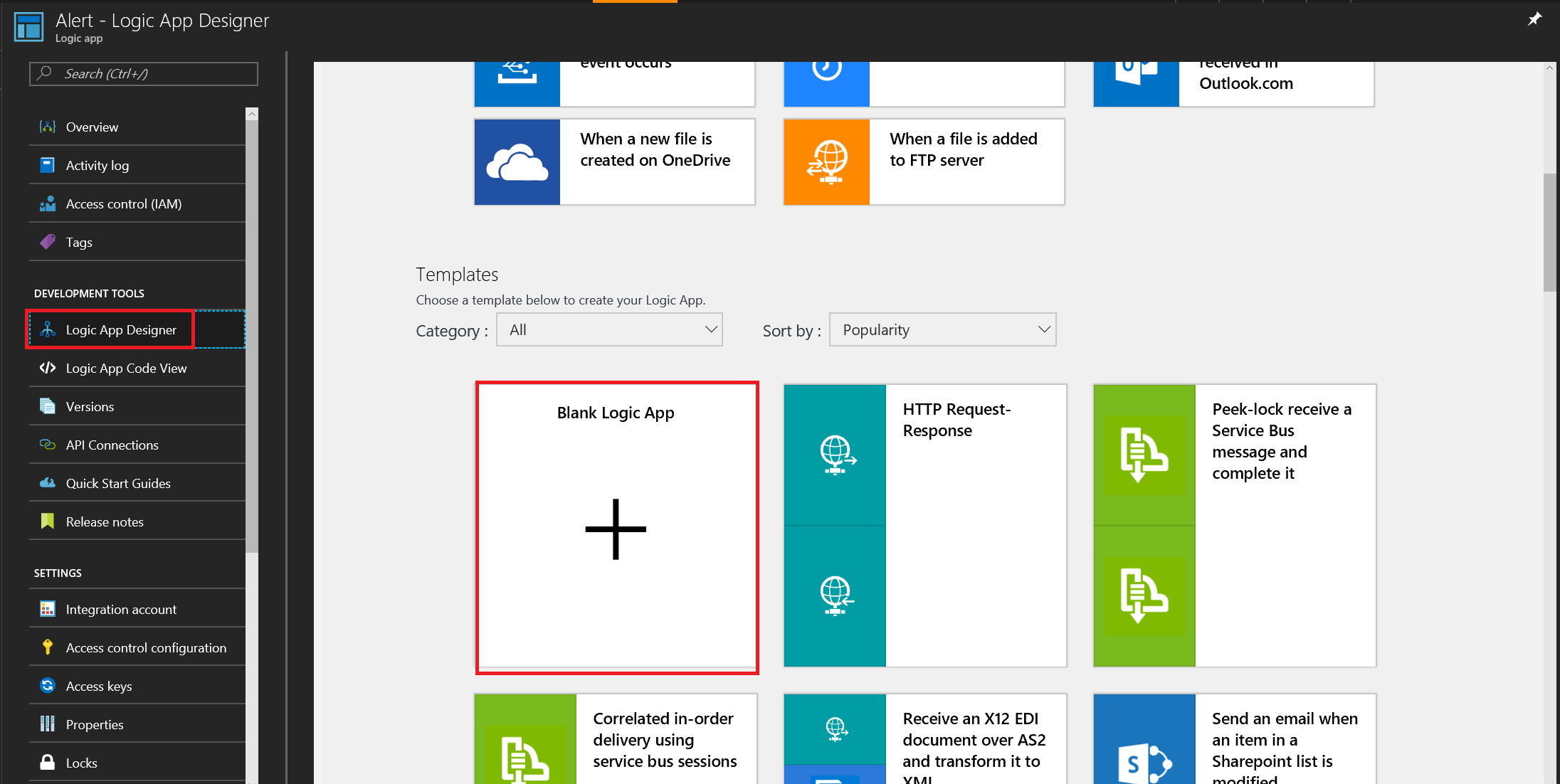
INTO alert

FROM iot

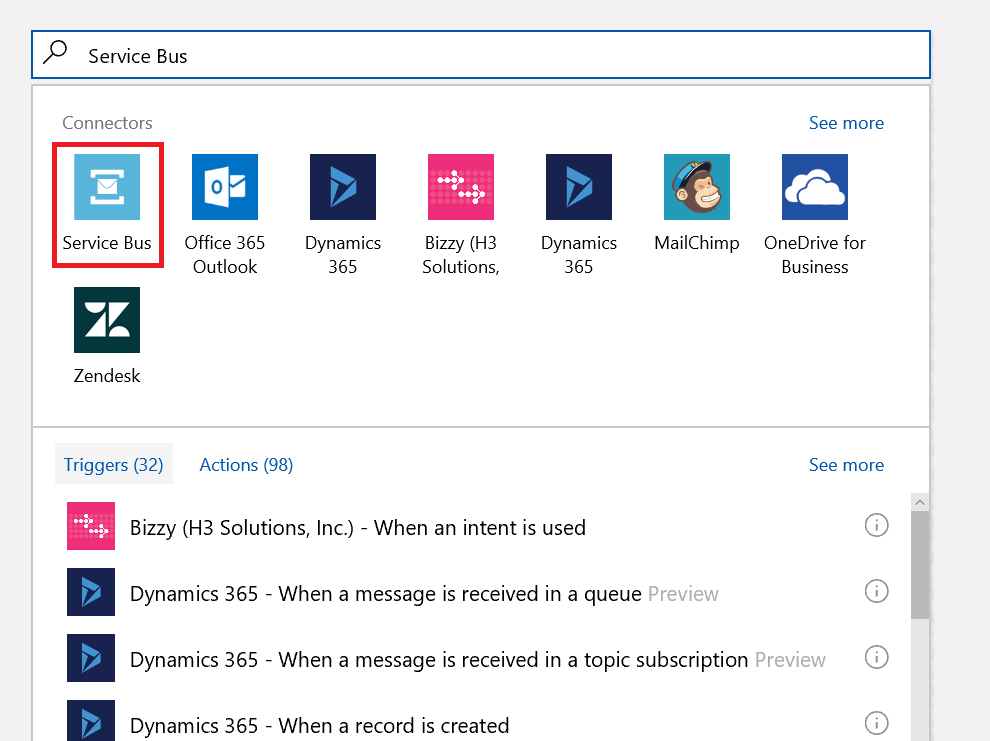
GROUP BY IoTHub.ConnectionDeviceId, HOPPINGWINDOW (second, 30, 15)

HAVING Count(\*) < 20

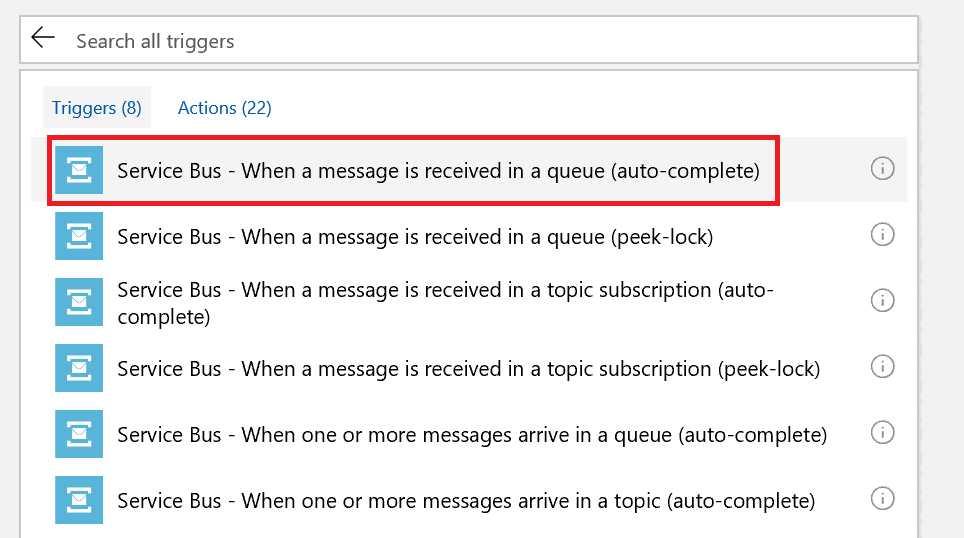
1. Save Query
2. Run Stream Analytics Jon. Now you have new option – *When Last Stopped*. Use this one. Remember. IoT Hub is buffer.
3. Go to Logic Apps service.
4. Open Logic Apps Designer and scroll down. Click *Blank Logic App*.



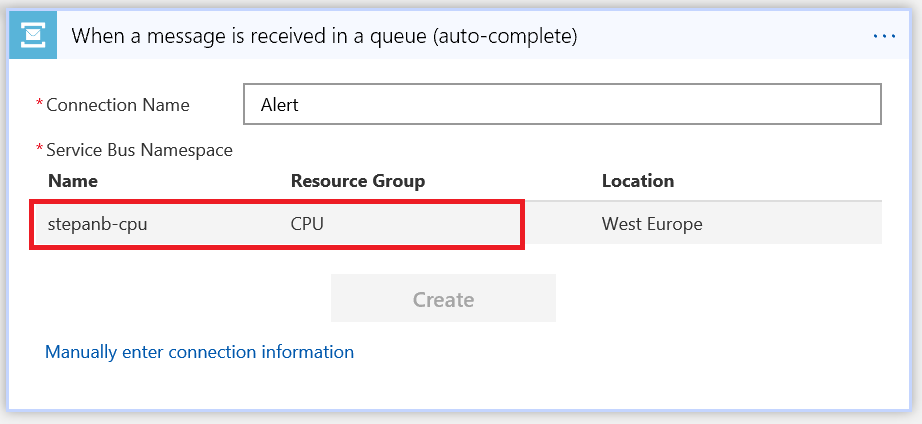
1. We will connect Servis Bus Queue containing error message from Stream Analytics with Outlook.
2. Search for Service Bus connector.



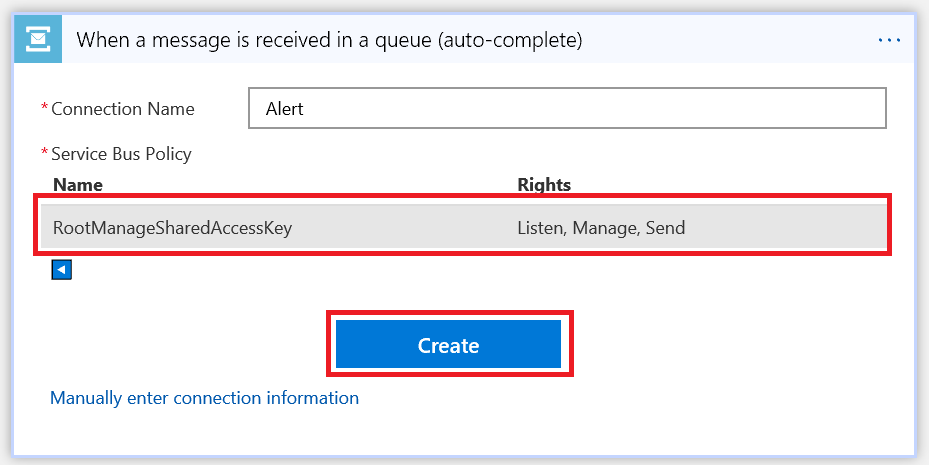
1. Click Service Bus connector.
2. Choose Trigger *When message is received in e queue (auto-complete)*.



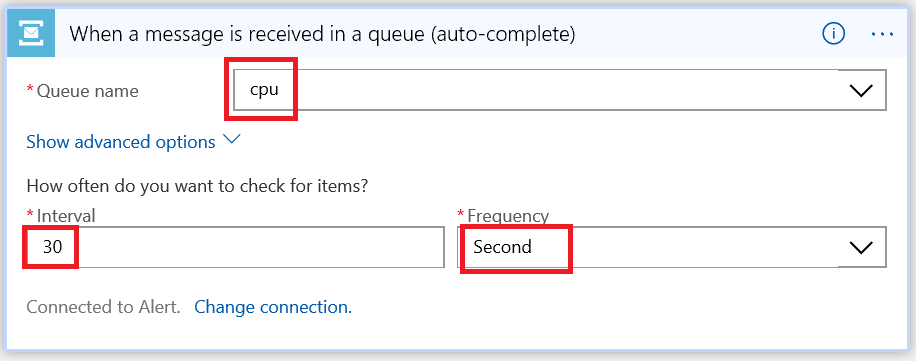
1. Choose your Service Bus.



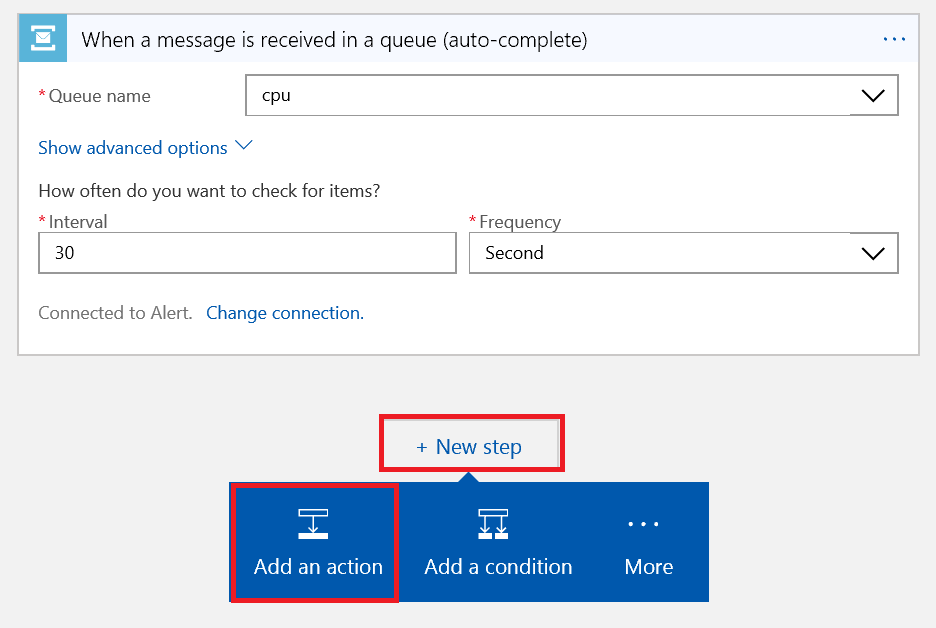
1. Click suggested keys and Create.



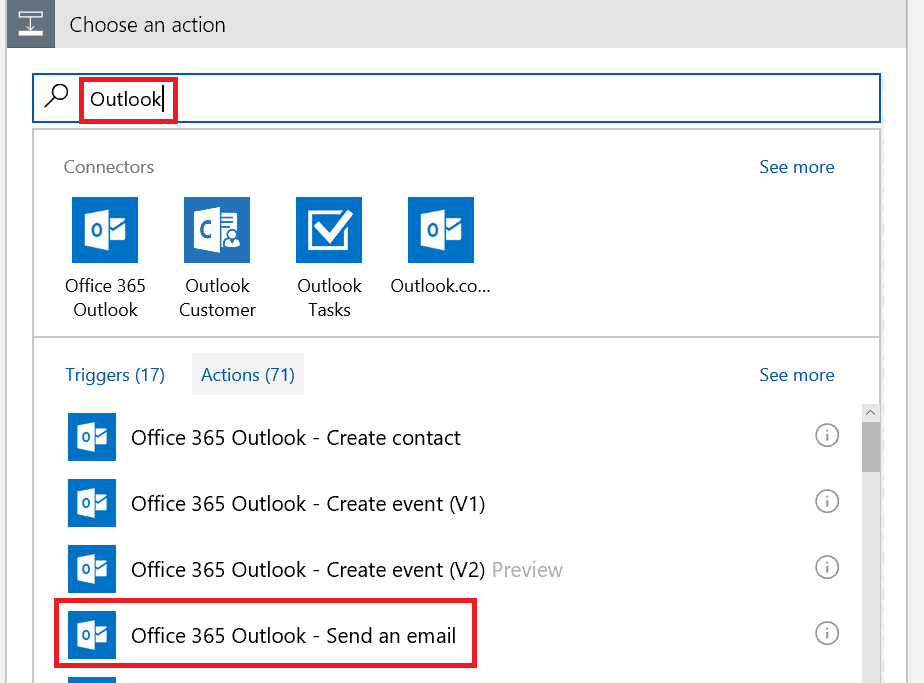
1. Choose your Queue and set check interval to 30 seconds.



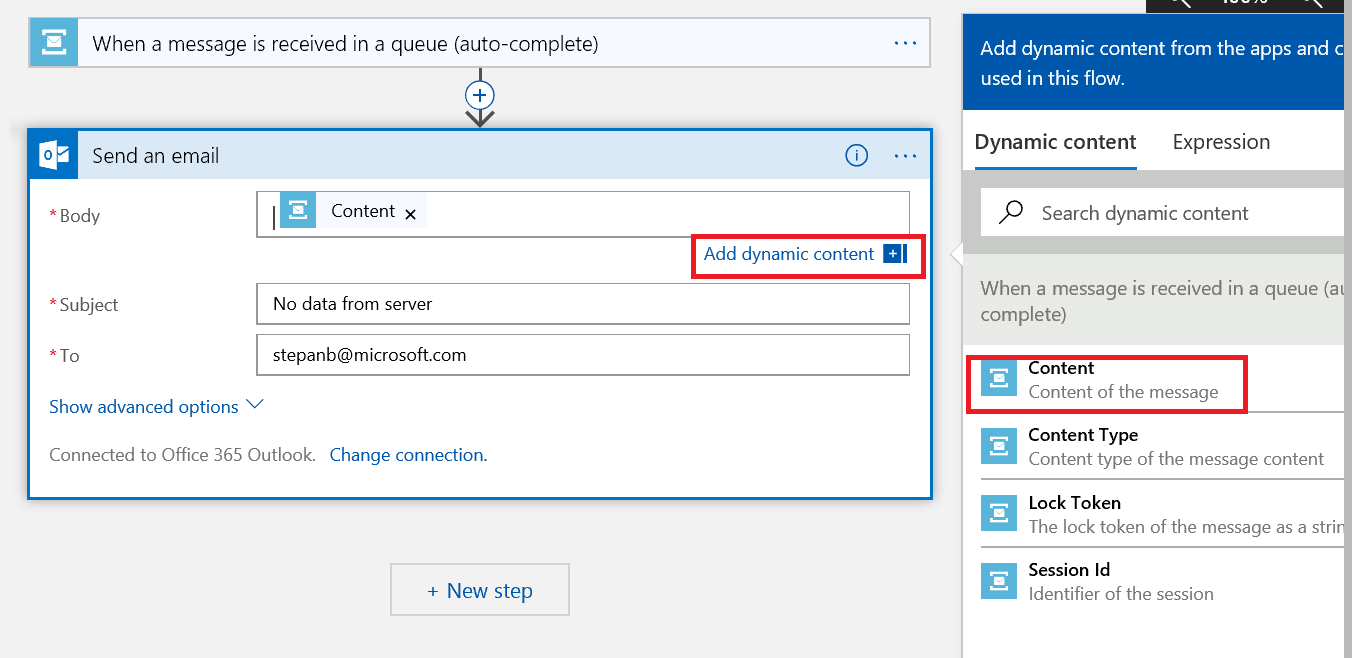
1. Click *Next Step* and *Add an Action*.



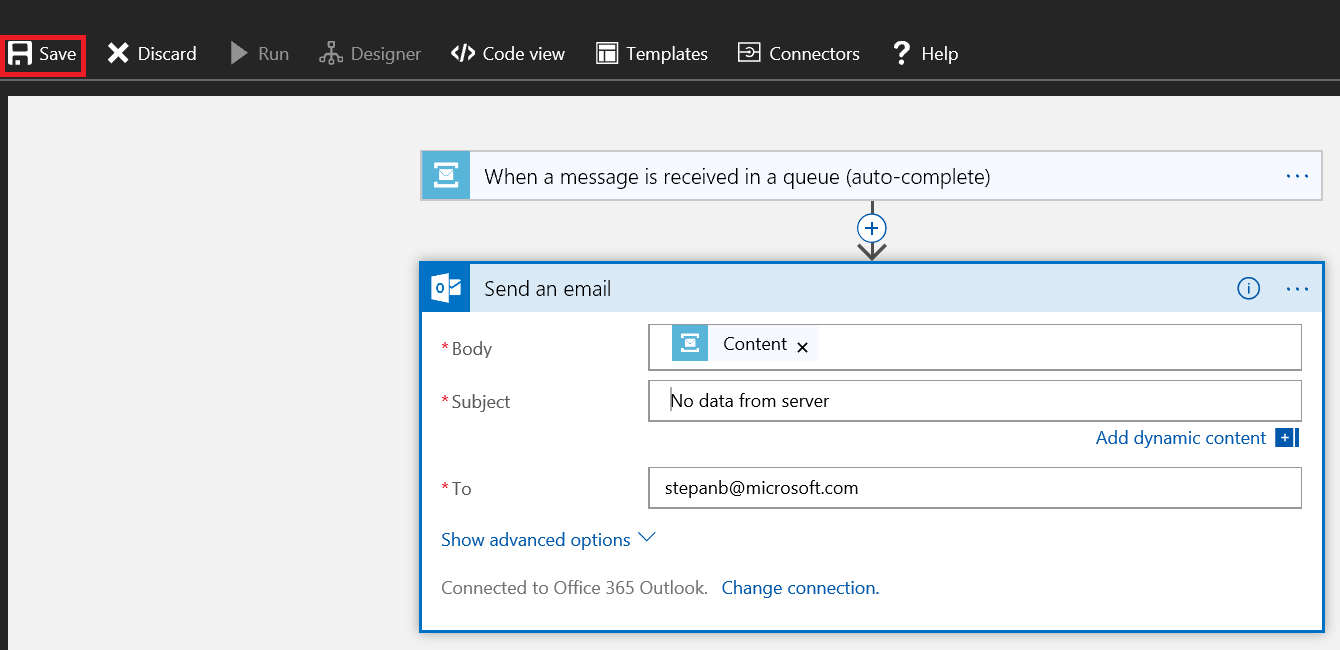
1. Search for *Office 365 Outlook*. Find *Office 365 Outlook – Send an email* action and click it.

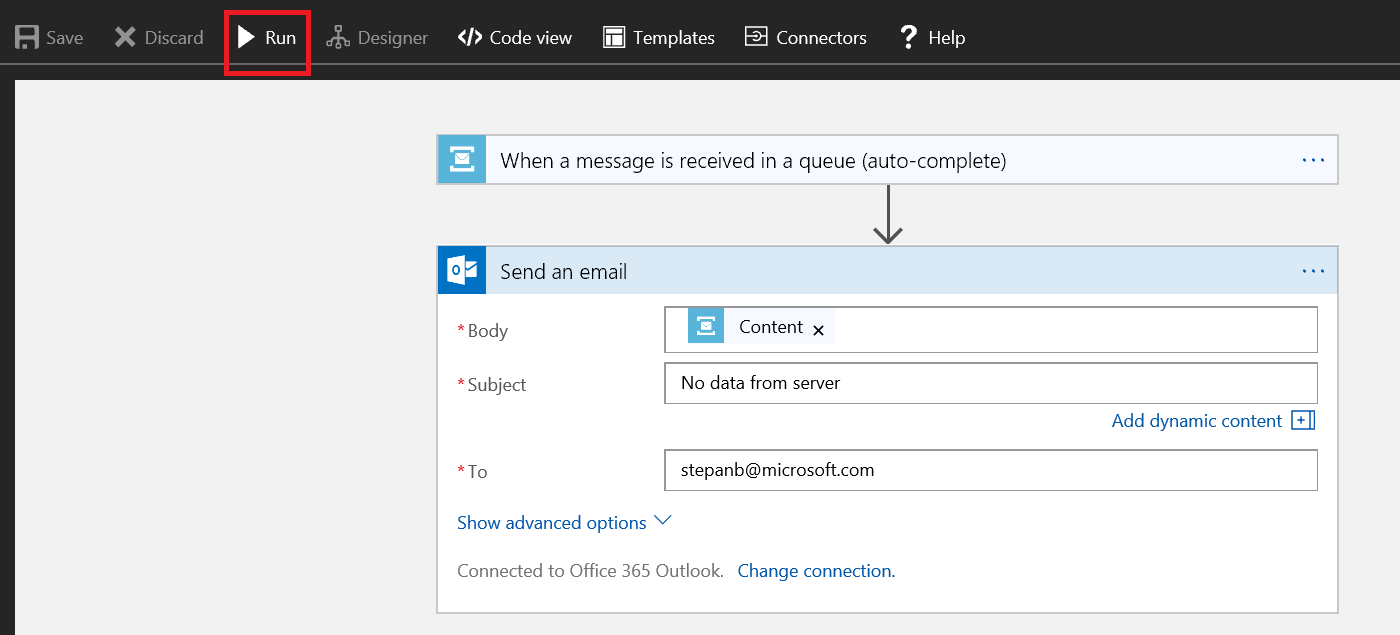


1. Click *Sign in*. Use your Office 365 credentials.
2. Create email. Use *Dynamic content* for *Body*.



1. Click *Save* and *Run*.





1. If you stop application sending CPU usage you should get e-mail.