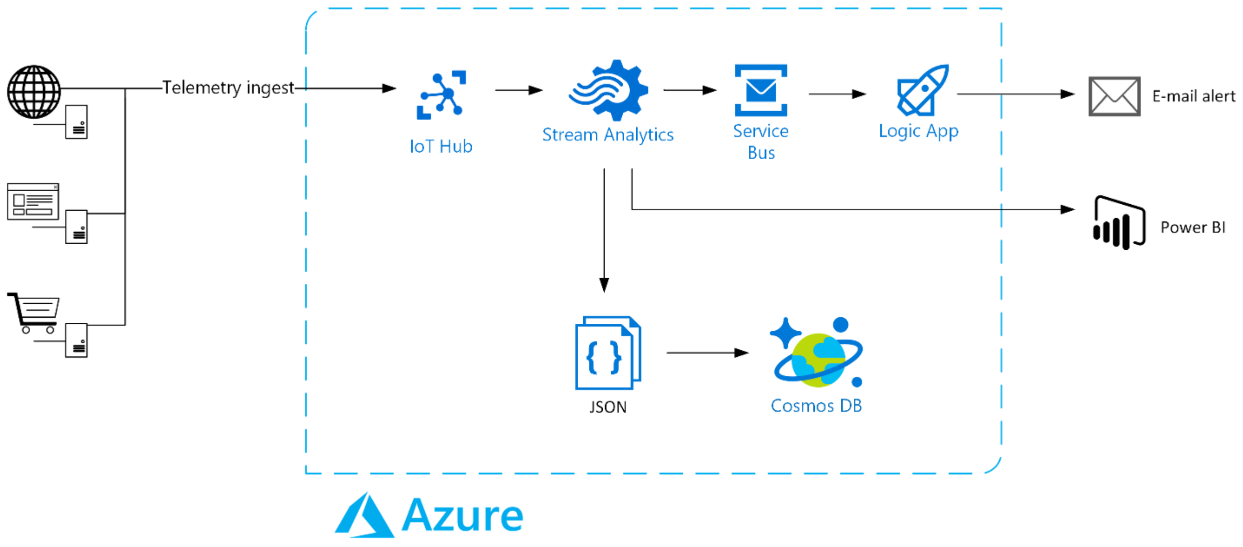
Azure IoT Workshop

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

- We will get CPU usage every 11 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



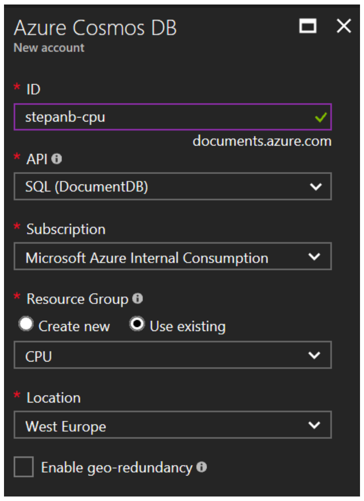
Prerequisites

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.



Azure Cosmos DB
New account

* ID
stepanb-cpu
documents.azure.com

* API
SQL (DocumentDB)

* Subscription
Microsoft Azure Internal Consumption

* Resource Group
Create new [✓ Use existing]
CPU

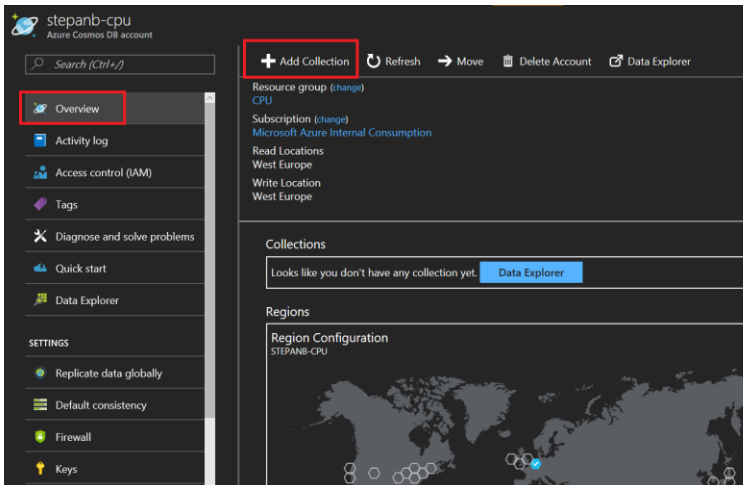
* Location
West Europe

Enable geo-redundancy

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

Configure Cosmos DB

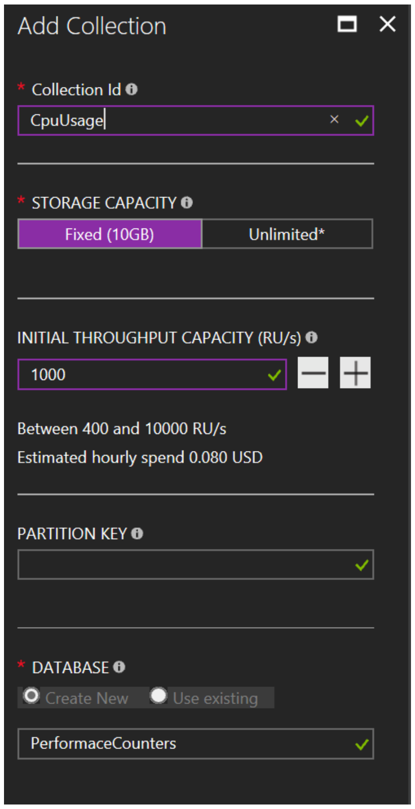
1. Open Cosmos DB from *CPU* Resource group.
2. Go to *Overview* and click *Add Collection*



Resource group (change)
CPU
Subscription (change)
Microsoft Azure Internal Consumption
Read Locations
West Europe
Write Location
West Europe

Looks like you don't have any collection yet.

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



Add Collection

* Collection Id
CpuUsage

* STORAGE CAPACITY
[✓ Fixed (10GB)] Unlimited*

INITIAL THROUGHPUT CAPACITY (RU/s)
1000

Between 400 and 10000 RU/s
Estimated hourly spend 0.080 USD

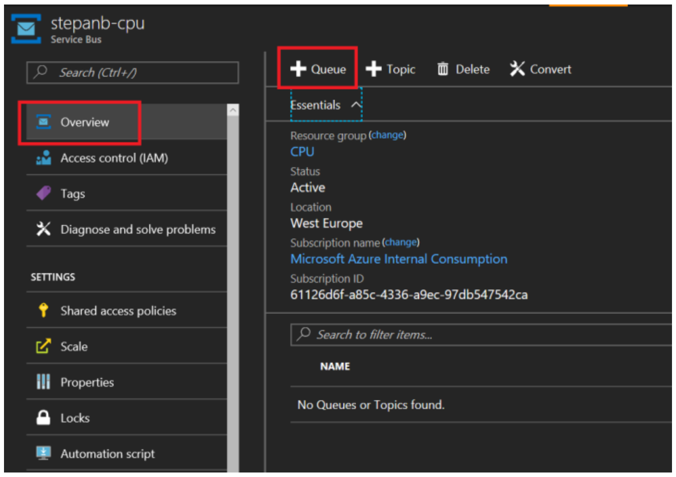
PARTITION KEY

* DATABASE
[✓ Create New] Use existing

PerformaceCounters

Configure Service Bus

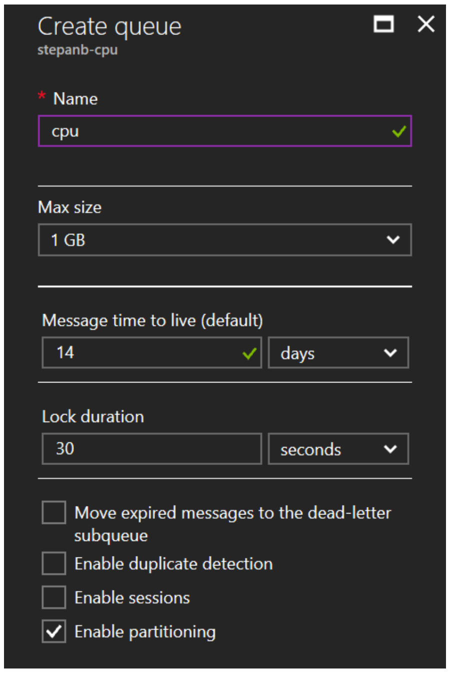
1. Open Service Bus from *CPU* Resource group.
2. Go to *Overview* and click + *Queue*


stepanb-cpu
Service Bus

Resource group (change)
CPU
Status
Active
Location
West Europe
Subscription name (change)
Microsoft Azure Internal Consumption
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca

No Queues or Topics found.

3. Use *cpu* as name.


Create queue
stepanb-cpu

* Name: cpu

Max size: 1 GB

Message time to live (default): 14 days

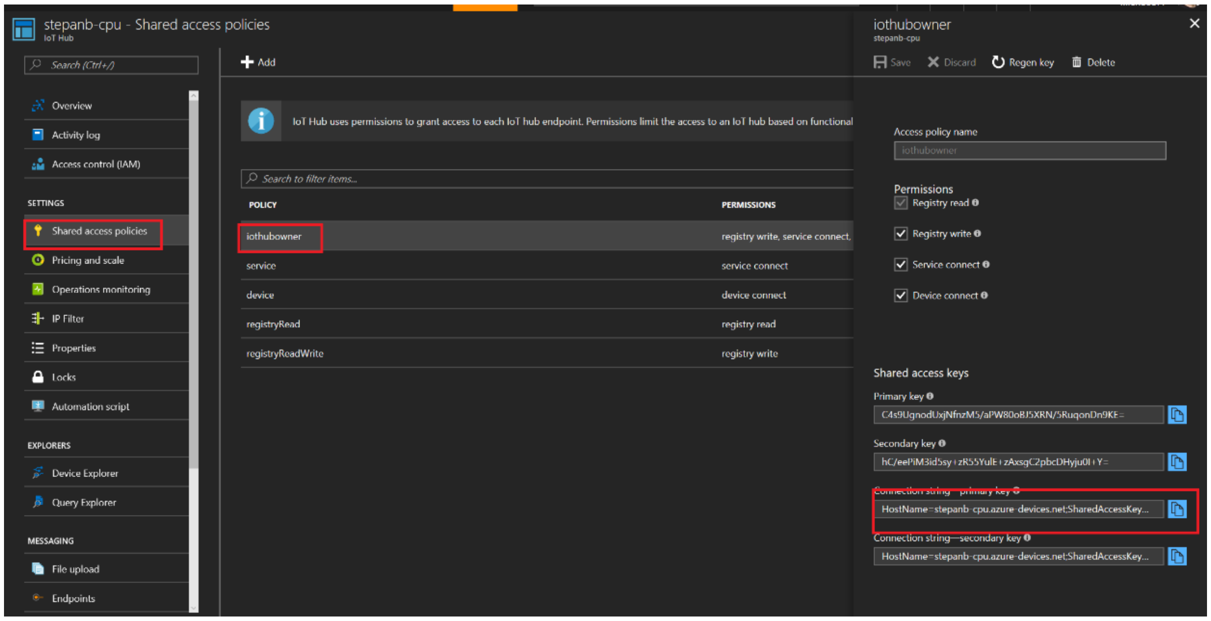
Lock duration: 30 seconds

- [ ] Move expired messages to the dead-letter subqueue
- [ ] Enable duplicate detection
- [ ] Enable sessions
- [x] Enable partitioning

Application

Application reads every 11 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.



IoT Hub uses permissions to grant access to each IoT hub endpoint. Permissions limit the access to an IoT hub based on functional

POLICY	PERMISSIONS
iothubowner	registry write, service connect,
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

Access policy name: iothubowner

Permissions
- ✓ Registry read
- ✓ Registry write
- ✓ Service connect
- ✓ Device connect

Shared access keys

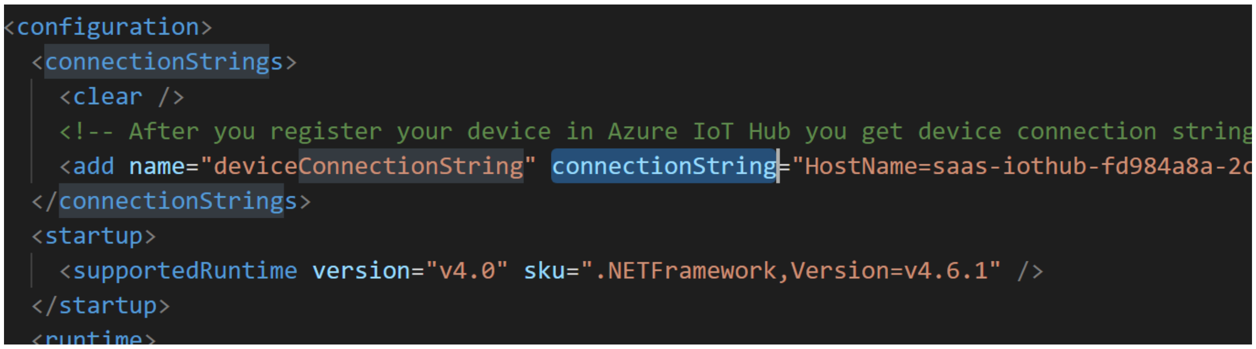
Primary key: C4s9UgnodUxjNfnzM5/aPW80oBJ5XRN/5RuqonDn9KE=

Secondary key: hC/eePiM3id5sy+zR55YulE+zAxsgC2pbcDHyju0I+Y=

Connection string—primary key: HostName=stepanb-cpu.azure-devices.net;SharedAccessKey...

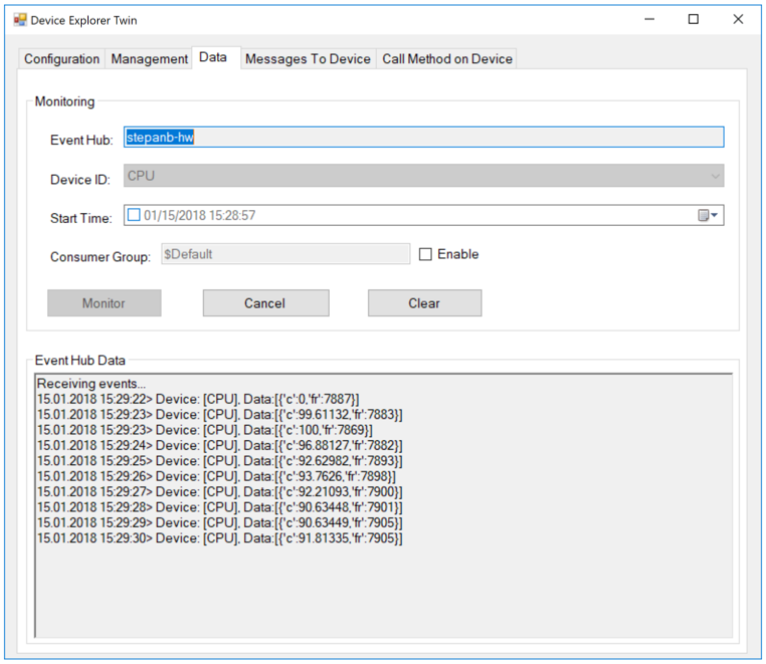
Connection string—secondary key: HostName=stepanb-cpu.azure-devices.net;SharedAccessKey...

4. Run Device Explorer and paste connection string to *IoT Hub Connection String* field and click *Update*.
5. 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.
6. Right click line with *your* device and click *Copy connection string for selected device*.
7. Open *CpuUsage2Azure* solution using Visual Studio
8. Open *App.config*.
9. Paste device connection string to *connectionString* setting and save file.



```
<configuration>
  <connectionStrings>
    <clear />
    <!-- After you register your device in Azure IoT Hub you get device connection string
    <add name="deviceConnectionString" connectionString="HostName=saas-iothub-fd984a8a-2c
  </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
  </startup>
  <runtime>
```

10. Open *Program.cs*. You can see how to use object *DeviceClient* to communicate with Azure IoT Hub. *DeviceClient* is from Device SDK.
11. Run application.
12. 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.



Monitoring

Event Hub: stepanb-hw

Device ID: CPU

Start Time: 01/15/2018 15:28:57

Consumer Group: $Default ☐ Enable

Event Hub Data

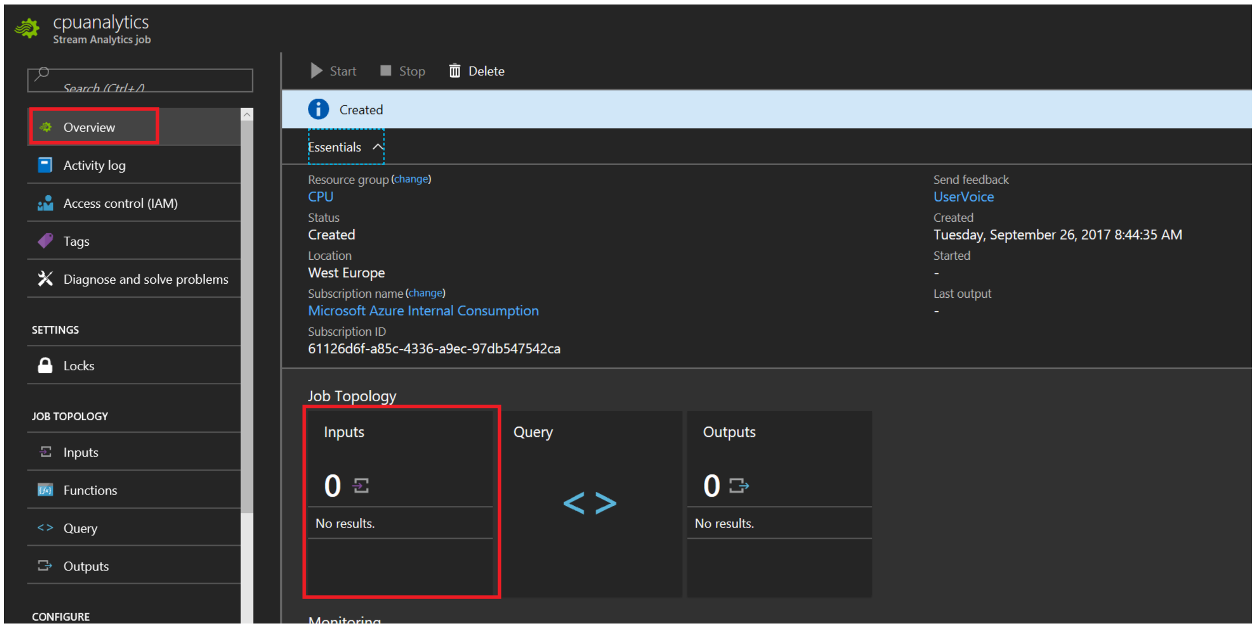
```
Receiving events...
15.01.2018 15:29:22> Device: [CPU], Data:[{'c':0,'fr':7887}]
15.01.2018 15:29:23> Device: [CPU], Data:[{'c':99.61132,'fr':7883}]
15.01.2018 15:29:23> Device: [CPU], Data:[{'c':100,'fr':7869}]
15.01.2018 15:29:24> Device: [CPU], Data:[{'c':96.88127,'fr':7882}]
15.01.2018 15:29:25> Device: [CPU], Data:[{'c':92.62982,'fr':7893}]
15.01.2018 15:29:26> Device: [CPU], Data:[{'c':93.7626,'fr':7898}]
15.01.2018 15:29:27> Device: [CPU], Data:[{'c':92.21093,'fr':7900}]
15.01.2018 15:29:28> Device: [CPU], Data:[{'c':90.63448,'fr':7901}]
15.01.2018 15:29:29> Device: [CPU], Data:[{'c':90.63449,'fr':7905}]
15.01.2018 15:29:30> Device: [CPU], Data:[{'c':91.81335,'fr':7905}]
```

13. 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*.



Created

Resource group (change)
CPU
Status
Created
Location
West Europe
Subscription name (change)
Microsoft Azure Internal Consumption
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca

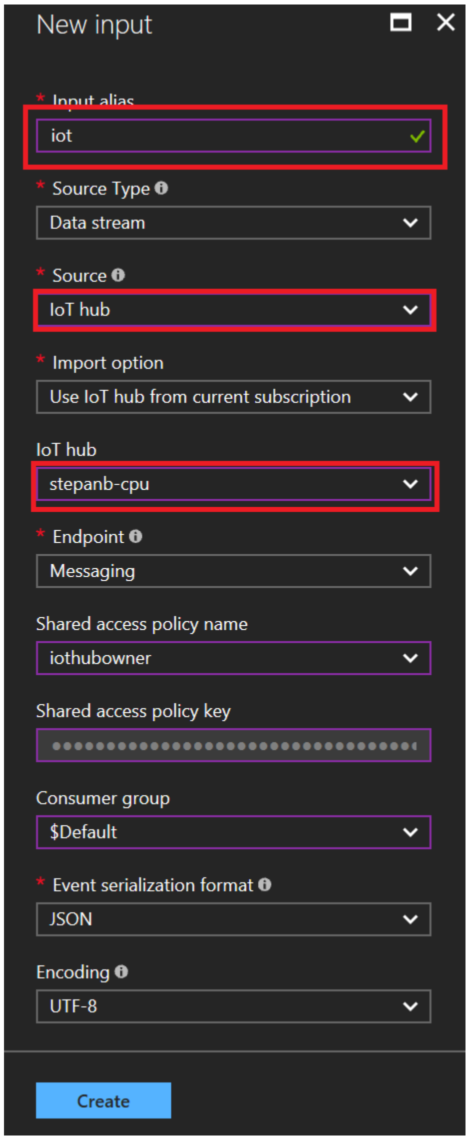
Send feedback
UserVoice
Created
Tuesday, September 26, 2017 8:44:35 AM
Started
-
Last output
-

Inputs 0 No results.

Query < >

Outputs 0 No results.

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



New input

* Input alias: iot

* Source Type: Data stream

* Source: IoT hub

* Import option: Use IoT hub from current subscription

IoT hub: stepanb-cpu

* Endpoint: Messaging

Shared access policy name: iothubowner

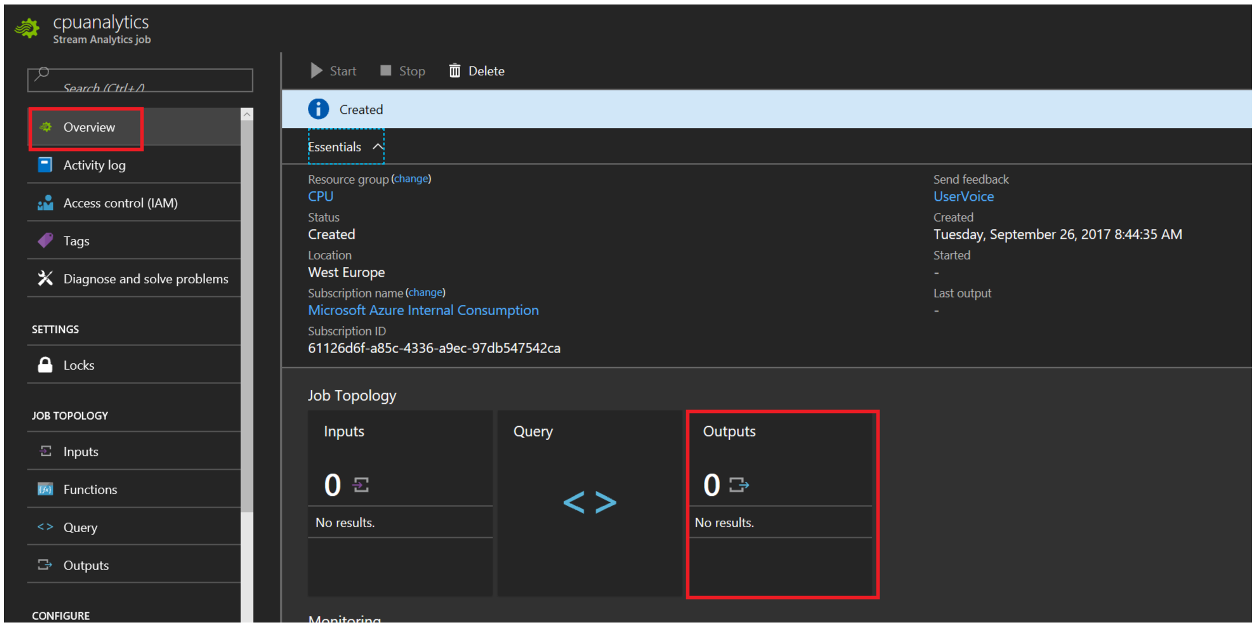
Shared access policy key: ●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

Consumer group: $Default

* Event serialization format: JSON

Encoding: UTF-8

4. Click *Create*.
5. Go to *Overview* and click *Outputs*.



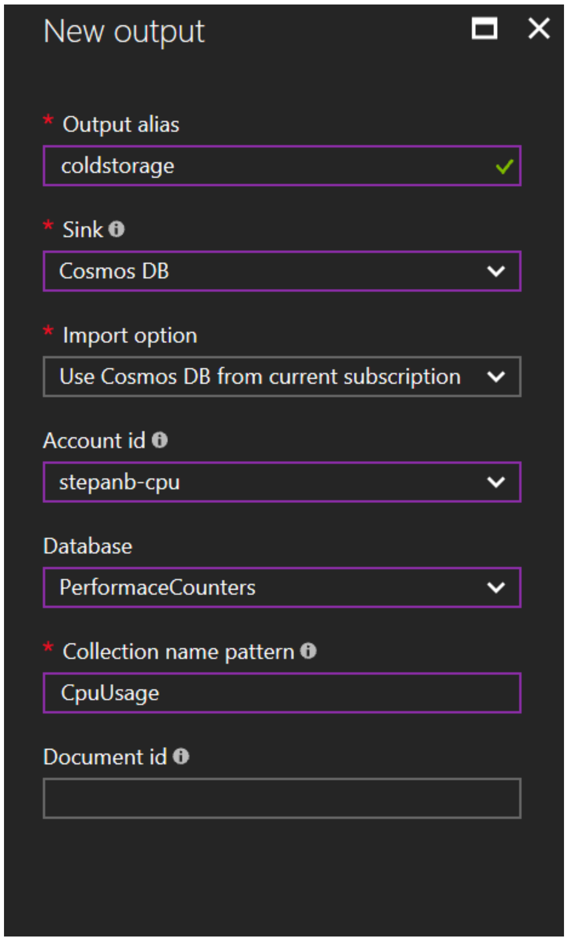
Created

Resource group (change)
CPU
Status
Created
Location
West Europe
Subscription name (change)
Microsoft Azure Internal Consumption
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca

Send feedback
UserVoice
Created
Tuesday, September 26, 2017 8:44:35 AM
Started
-
Last output
-

Inputs: 0 — No results.
Outputs: 0 — No results.

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



New output

* Output alias: coldstorage

* Sink: Cosmos DB

* Import option: Use Cosmos DB from current subscription

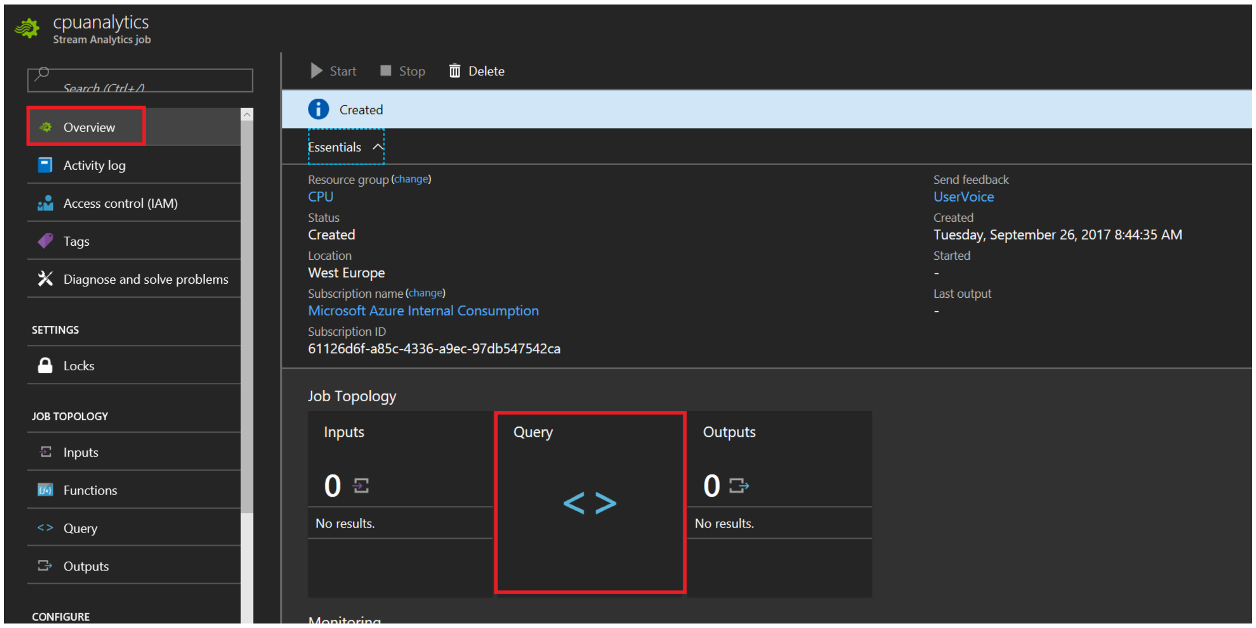
Account id: stepanb-cpu

Database: PerformaceCounters

* Collection name pattern: CpuUsage

Document id:

7. Click *Create*.
8. Go to *Overview* and click *Query*.



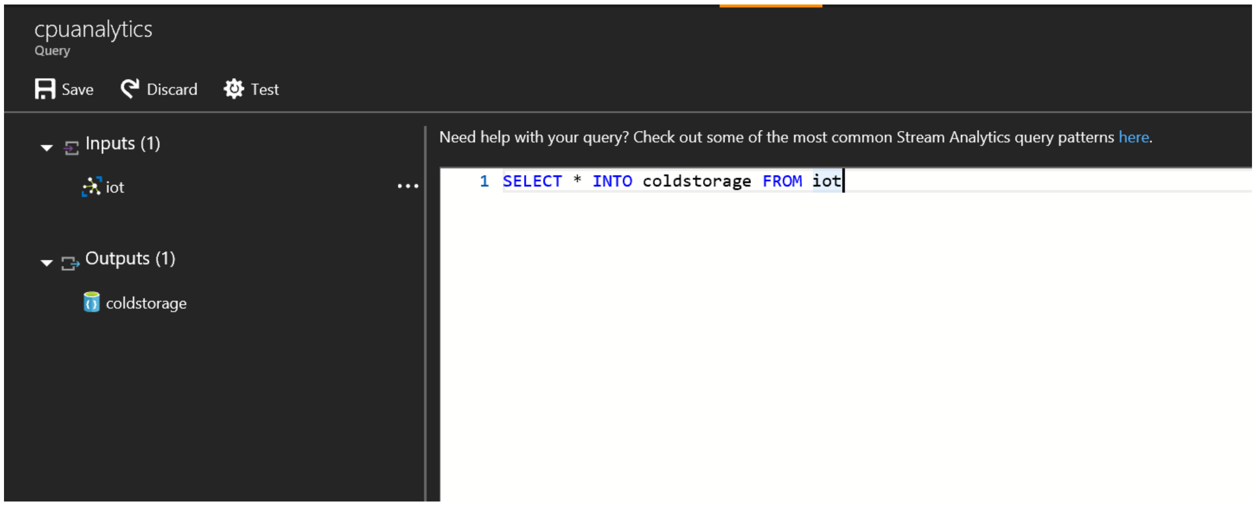
Resource group (change)
CPU
Status
Created
Location
West Europe
Subscription name (change)
Microsoft Azure Internal Consumption
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca

Send feedback
UserVoice
Created
Tuesday, September 26, 2017 8:44:35 AM
Started
-
Last output
-

Inputs 0 No results.
Outputs 0 No results.

9. We want to store all data for future use (for example, for Machine Learning). We will use basic query. All from input we send to output. Don't forget to click *Save* button.

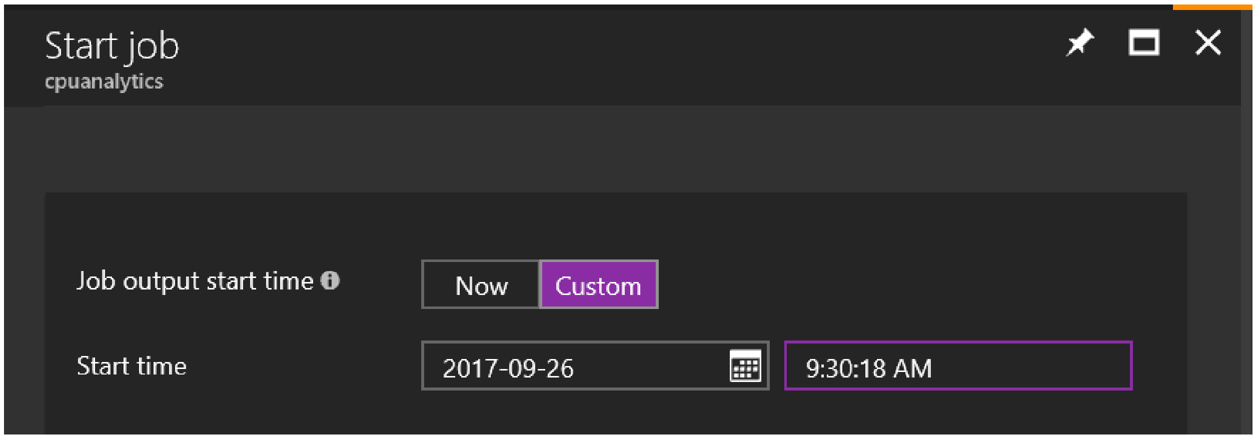
```
SELECT * INTO coldstorage FROM iot
```



Need help with your query? Check out some of the most common Stream Analytics query patterns here.

```
SELECT * INTO coldstorage FROM iot
```

10. 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.

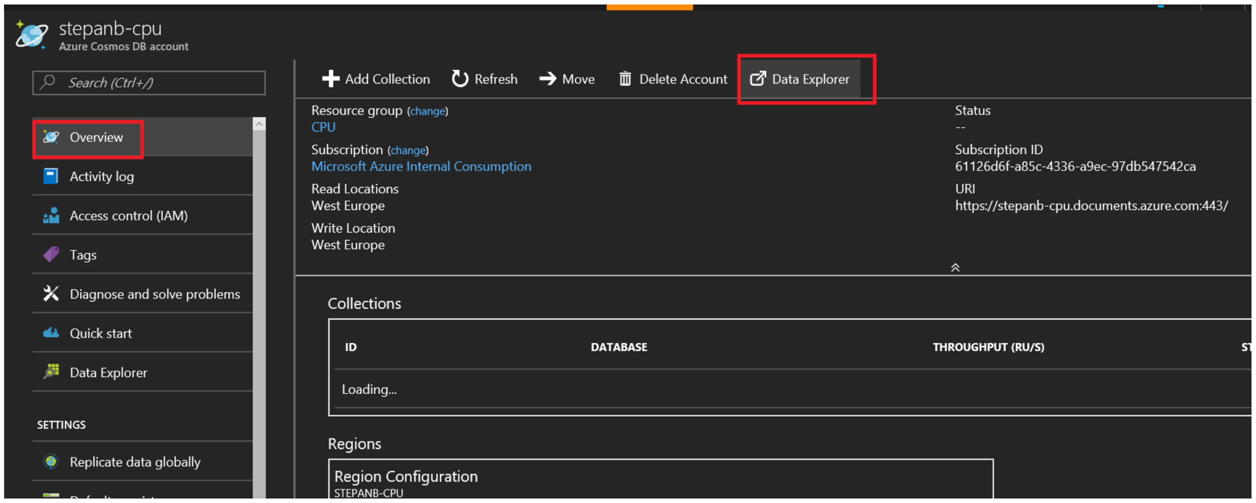


Start job
cpuanalytics

Job output start time: Now [✓ Custom]

Start time: 2017-09-26 9:30:18 AM

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



Resource group (change)
CPU
Subscription (change)
Microsoft Azure Internal Consumption
Read Locations
West Europe
Write Location
West Europe

Status
--
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca
URI
https://stepanb-cpu.documents.azure.com:443/

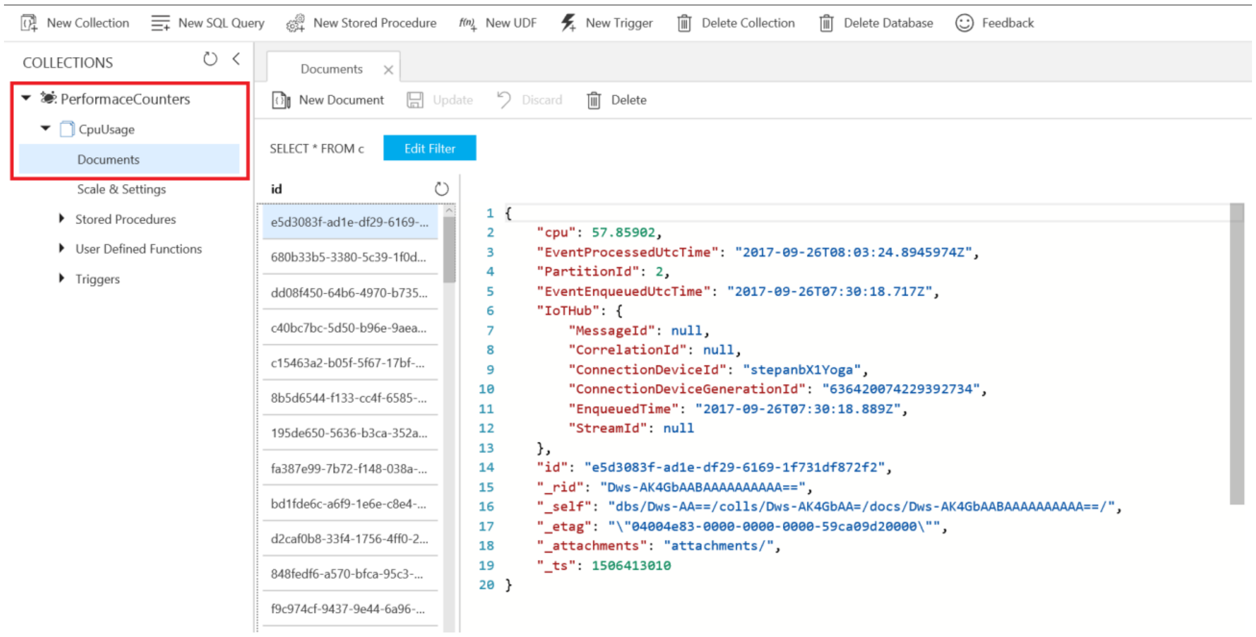
Collections

ID	DATABASE	THROUGHPUT (RU/S)
Loading...		

Regions

Region Configuration
STEPANB-CPU

12. Drill down to your collection.



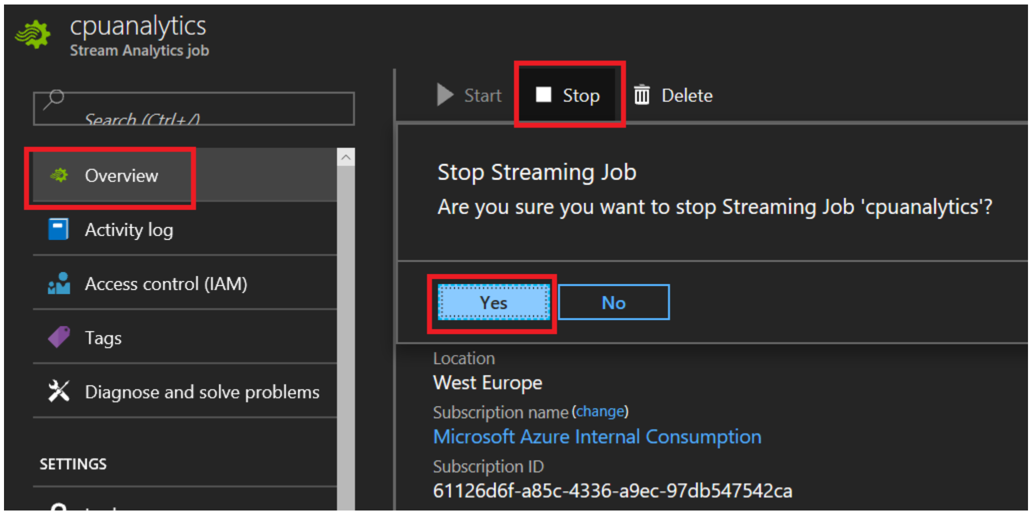
SELECT * FROM c

e5d3083f-ad1e-df29-6169-...
680b33b5-3380-5c39-1f0d...
dd08f450-64b6-4970-b735...
c40bc7bc-5d50-b96e-9aea...
c15463a2-b05f-5f67-17bf-...
8b5d6544-f133-cc4f-6585-...
195de650-5636-b3ca-352a...
fa387e99-7b72-f148-038a-...
bd1fde6c-a6f9-1e6e-c8e4-...
d2caf0b8-33f4-1756-4ff0-2...
848fedf6-a570-bfca-95c3-...
f9c974cf-9437-9e44-6a96-...

```
{
    "cpu": 57.85902,
    "EventProcessedUtcTime": "2017-09-26T08:03:24.8945974Z",
    "PartitionId": 2,
    "EventEnqueuedUtcTime": "2017-09-26T07:30:18.717Z",
    "IoTHub": {
        "MessageId": null,
        "CorrelationId": null,
        "ConnectionDeviceId": "stepanbX1Yoga",
        "ConnectionDeviceGenerationId": "636420074229392734",
        "EnqueuedTime": "2017-09-26T07:30:18.889Z",
        "StreamId": null
    },
    "id": "e5d3083f-ad1e-df29-6169-1f731df872f2",
    "_rid": "Dws-AK4GbAABAAAAAAAAAA==",
    "_self": "dbs/Dws-AA==/colls/Dws-AK4GbAA=/docs/Dws-AK4GbAABAAAAAAAAAA==/",
    "_etag": "\"04004e83-0000-0000-0000-59ca09d20000\"",
    "_attachments": "attachments/",
    "_ts": 1506413010
}
```

Near real-time data visualization in Power BI

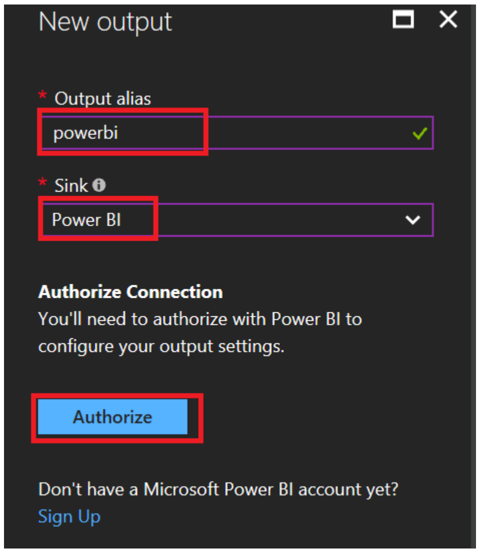
1. Go to Stream Analytics Job and stop it. Keep application reading CPU usage running.



Stop Streaming Job
Are you sure you want to stop Streaming Job 'cpuanalytics'?

Location
West Europe
Subscription name (change)
Microsoft Azure Internal Consumption
Subscription ID
61126d6f-a85c-4336-a9ec-97db547542ca

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



New output

* Output alias

powerbi

* Sink

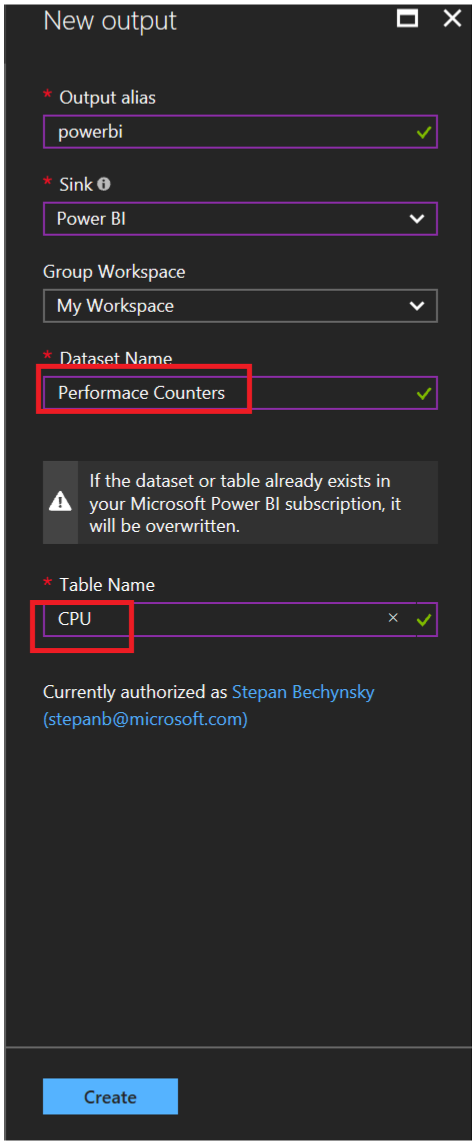
Power BI

Authorize Connection

You'll need to authorize with Power BI to configure your output settings.

Don't have a Microsoft Power BI account yet?
Sign Up

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


New output

* Output alias
powerbi

* Sink
Power BI

Group Workspace
My Workspace

* Dataset Name
Performace Counters

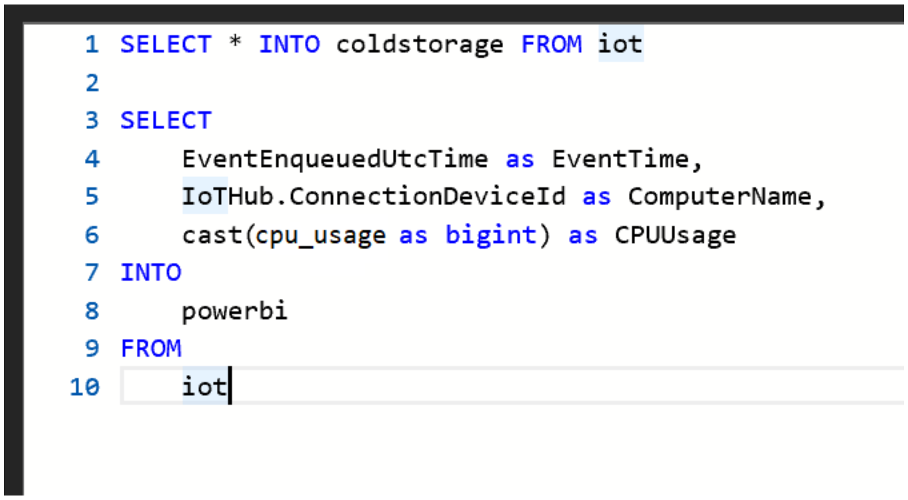
If the dataset or table already exists in your Microsoft Power BI subscription, it will be overwritten.

* Table Name
CPU

Currently authorized as Stepan Bechynsky (stepanb@microsoft.com)

6. Click create.
7. Go to *Query*.
8. 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.

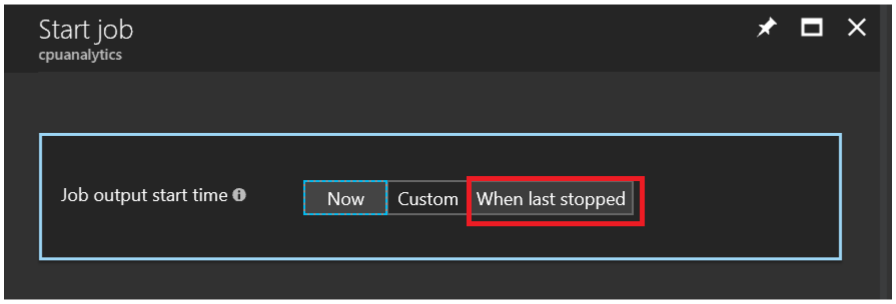
```
SELECT
    EventEnqueuedUtcTime as EventTime,
    IoTHub.ConnectionDeviceId as ComputerName,
    cast(cpu_usage as bigint) as CPUUsage
INTO
    powerbi
FROM
    iot
```



```
SELECT * INTO coldstorage FROM iot

SELECT
     EventEnqueuedUtcTime as EventTime,
     IoTHub.ConnectionDeviceId as ComputerName,
     cast(cpu_usage as bigint) as CPUUsage
INTO
     powerbi
FROM
     iot
```

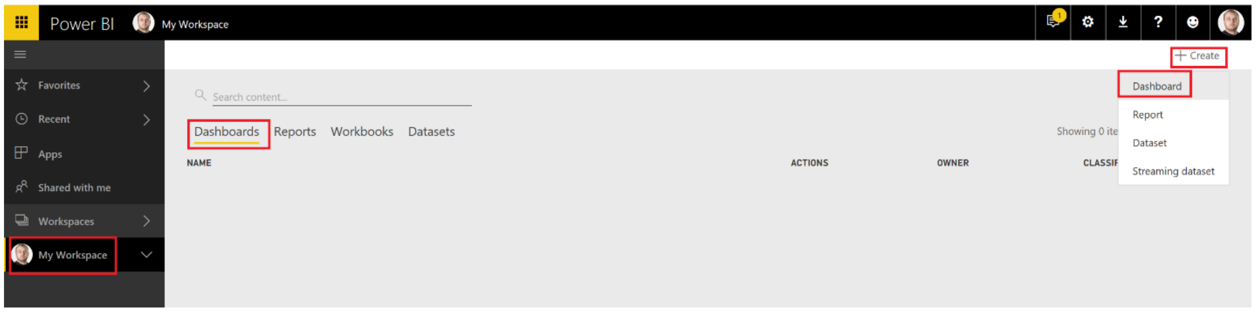
9. Save Query
10. Run Stream Analytics Jon. Now you have new option – *When Last Stopped*. Use this one. Remember. IoT Hub is buffer.



Start job
cpuanalytics

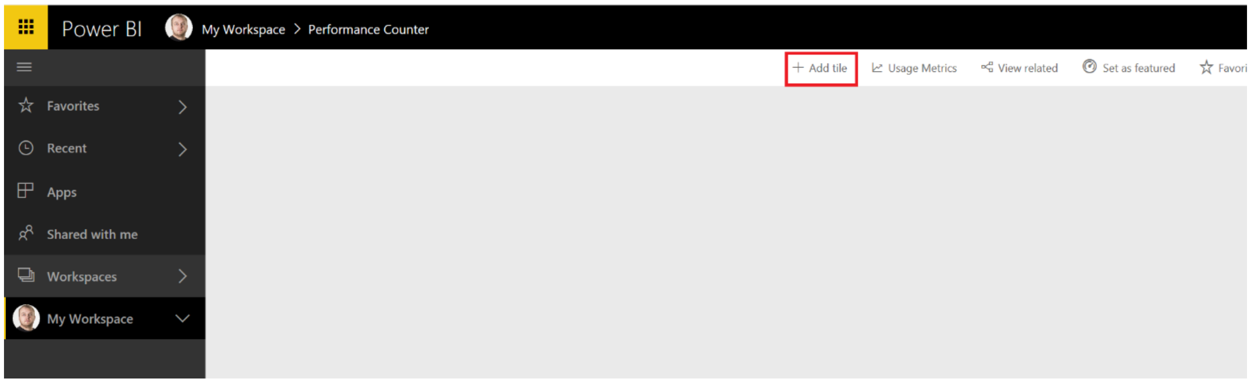
Job output start time: Now | Custom | [✓ When last stopped]

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

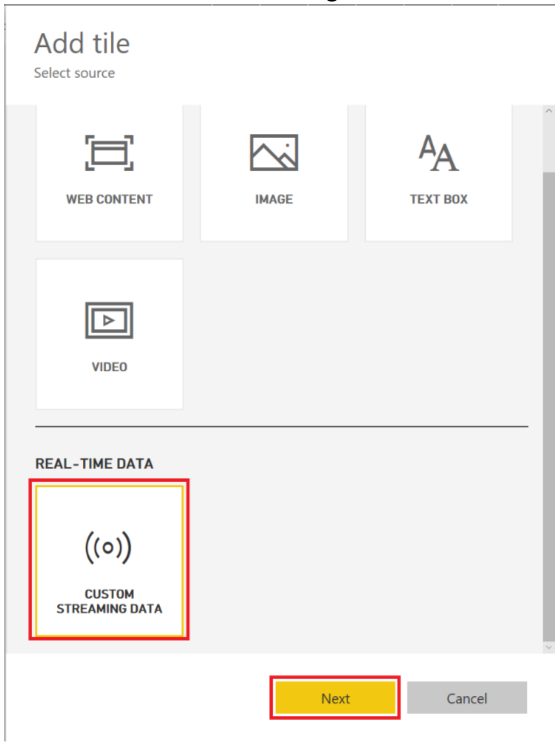


13. Name it *Performance Counter*
14. Click new Dashboard to open it.

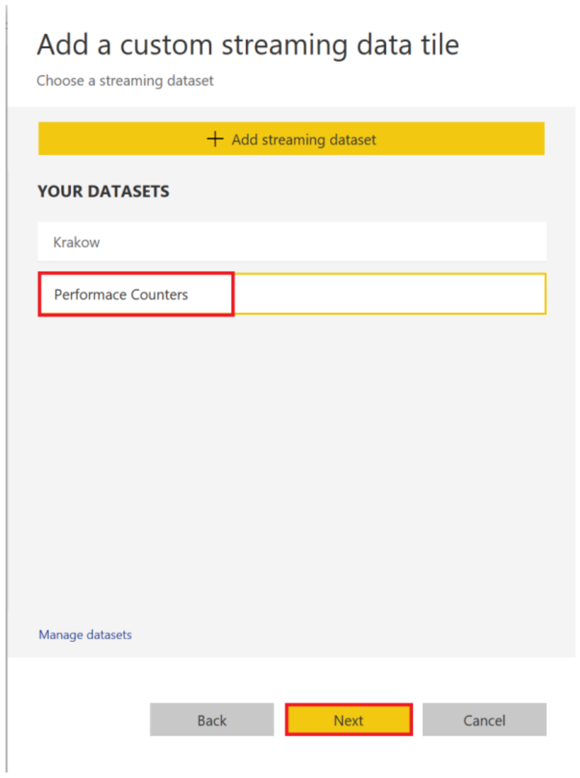
15. Click *Add Tile*



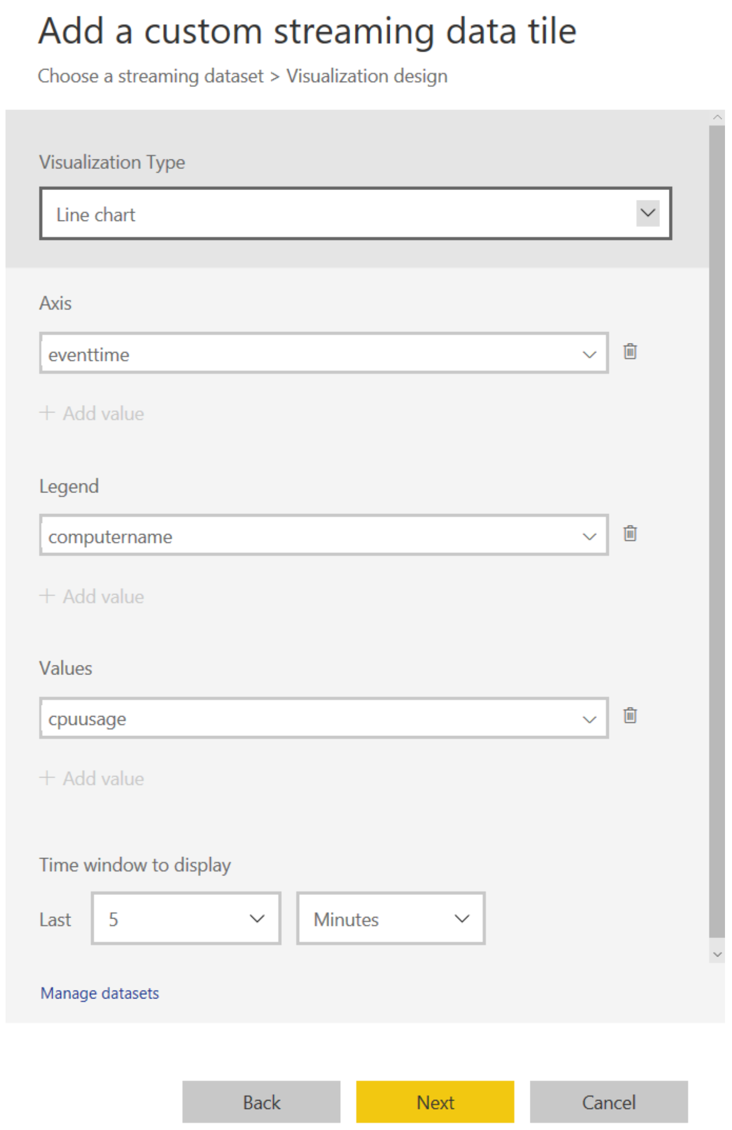
16. Choose *Custom Streaming Data* and click *Next*.



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



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


Add a custom streaming data tile

Choose a streaming dataset > Visualization design

Visualization Type: Line chart

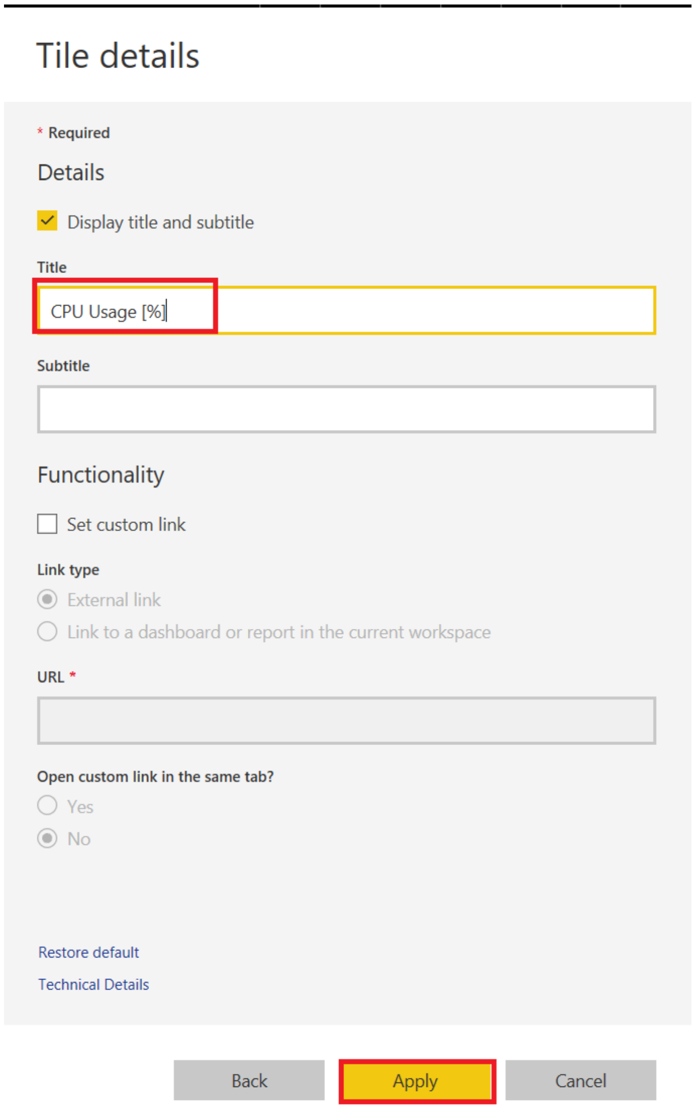
Axis: eventtime

Legend: computername

Values: cpuusage

Time window to display: Last 5 Minutes

19. Click *Next*.
20. Fill *Title* and click Applay.



Tile details

* Required

Details

[✓] Display title and subtitle

Title: CPU Usage [%]

Subtitle:

Functionality

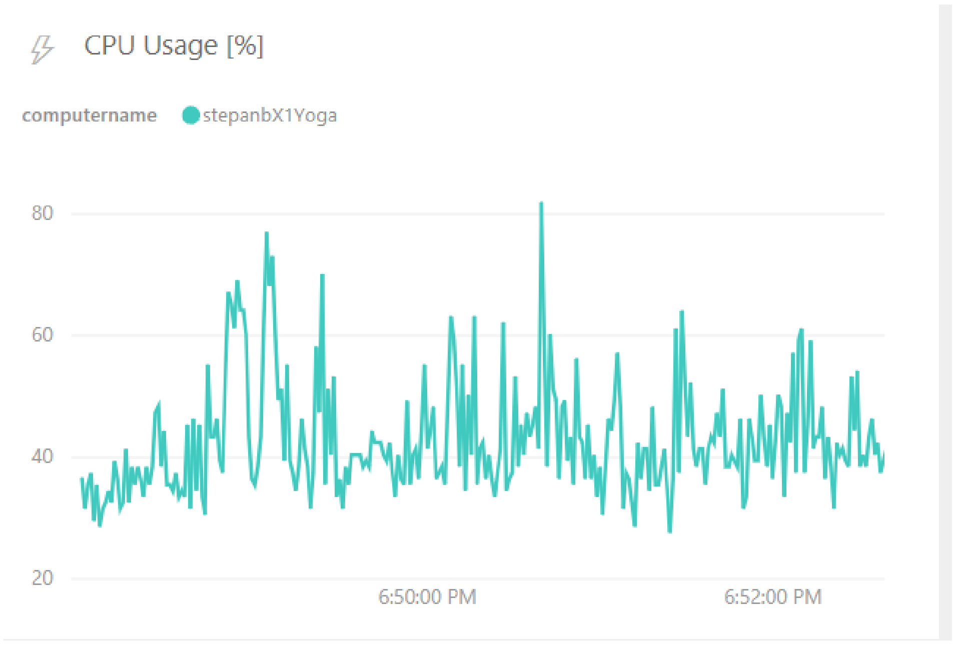
[ ] Set custom link

Link type
[✓ External link]
Link to a dashboard or report in the current workspace

URL *

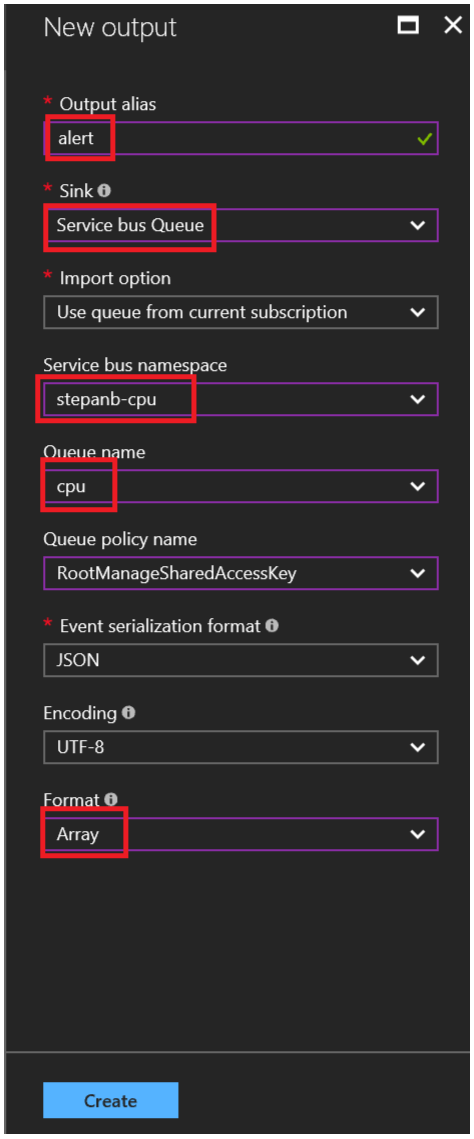
Open custom link in the same tab?
Yes [✓ No]

21. 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.


New output

* Output alias
alert

* Sink
Service bus Queue

* Import option
Use queue from current subscription

Service bus namespace
stepanb-cpu

Queue name
cpu

Queue policy name
RootManageSharedAccessKey

* Event serialization format
JSON

Encoding
UTF-8

Format
Array

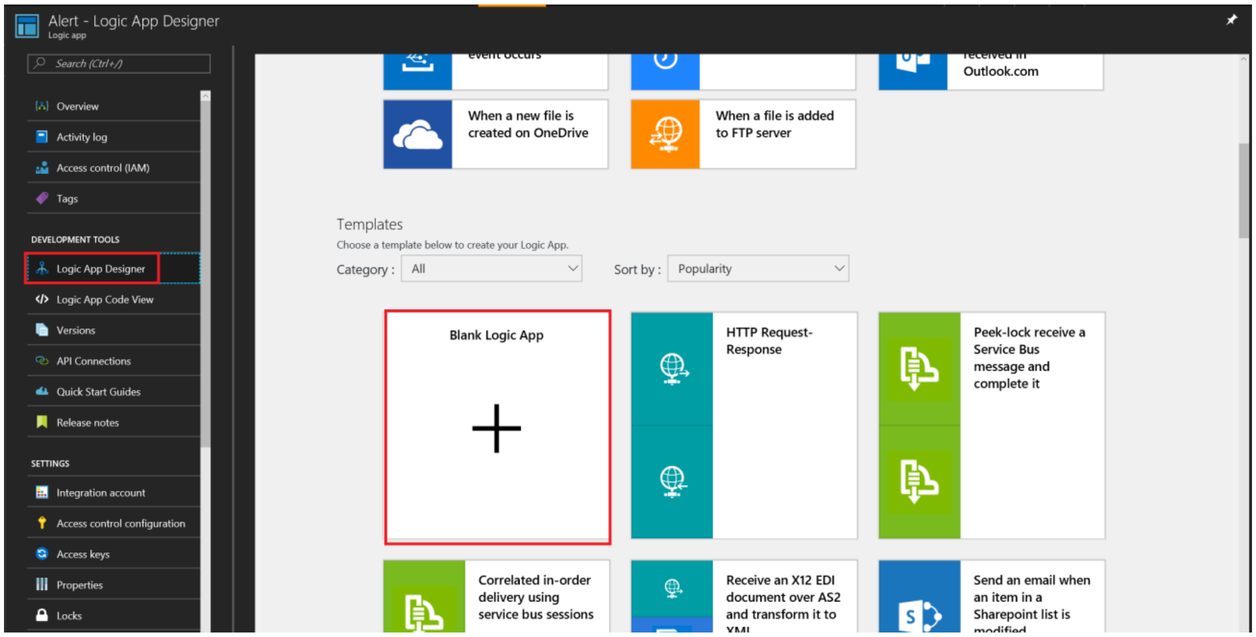
Create

4. Click create.
5. Go to *Query*.
6. Now we will use [Windowing]() 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.
7. 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
```

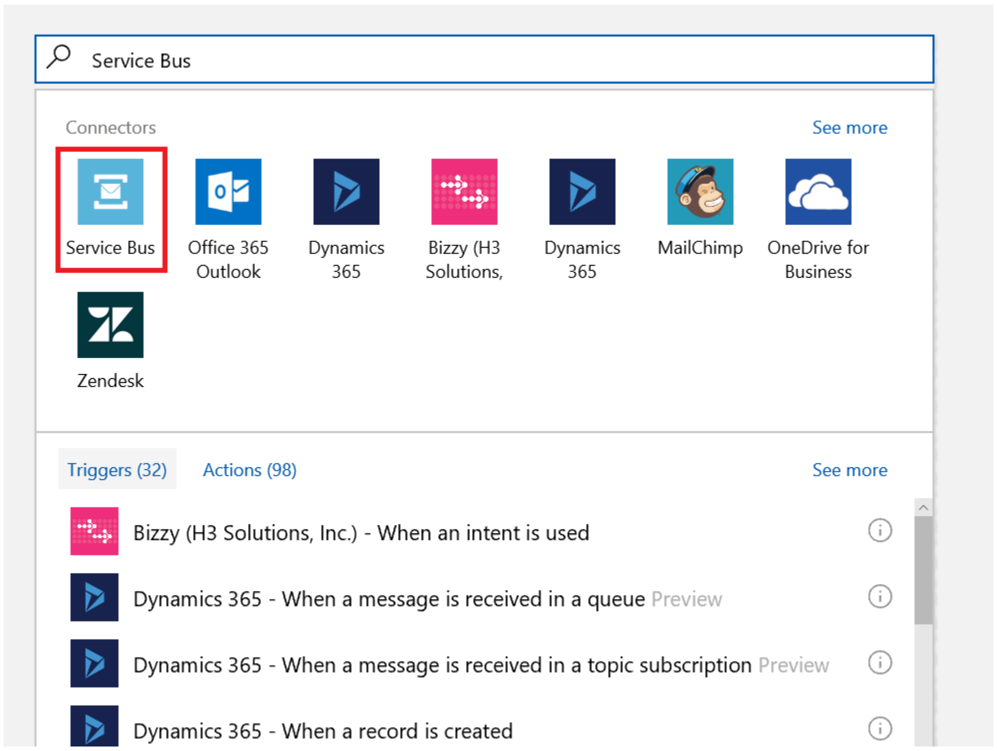
```
GROUP BY IoTHub.ConnectionDeviceId, HOPPINGWINDOW (second, 30, 15)
HAVING Count(*) < 20
```

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



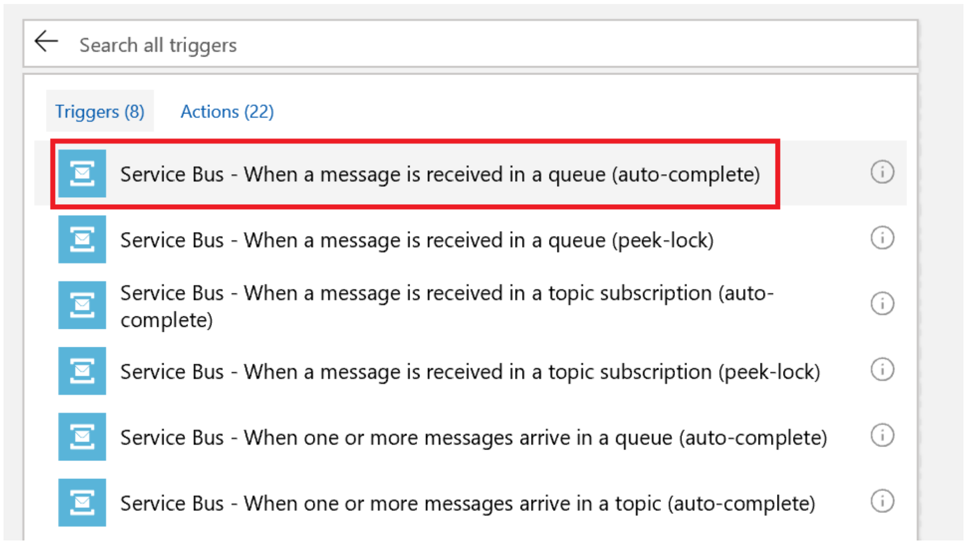
Alert - Logic App Designer
Logic app
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
DEVELOPMENT TOOLS
Logic App Designer
Logic App Code View
Versions
API Connections
Quick Start Guides
Release notes
SETTINGS
Integration account
Access control configuration
Access keys
Properties
Locks
Outlook.com
When a new file is created on OneDrive
When a file is added to FTP server
Templates
Choose a template below to create your Logic App.
Category : All
Sort by : Popularity
Blank Logic App
HTTP Request-Response
Peek-lock receive a Service Bus message and complete it
Correlated in-order delivery using service bus sessions
Receive an X12 EDI document over AS2 and transform it to
Send an email when an item in a Sharepoint list is

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



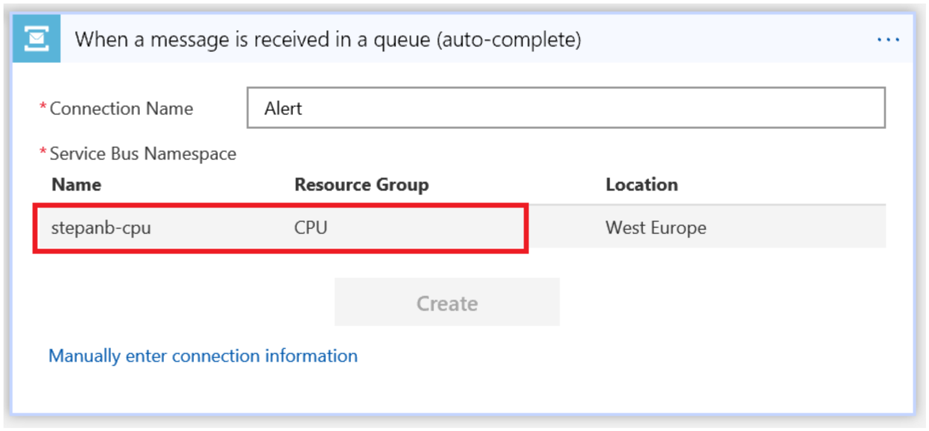
Bizzy (H3 Solutions, Inc.) - When an intent is used
Dynamics 365 - When a message is received in a queue Preview
Dynamics 365 - When a message is received in a topic subscription Preview
Dynamics 365 - When a record is created

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



Service Bus - When a message is received in a queue (auto-complete)
Service Bus - When a message is received in a queue (peek-lock)
Service Bus - When a message is received in a topic subscription (auto-complete)
Service Bus - When a message is received in a topic subscription (peek-lock)
Service Bus - When one or more messages arrive in a queue (auto-complete)
Service Bus - When one or more messages arrive in a topic (auto-complete)

16. Choose your Service Bus.

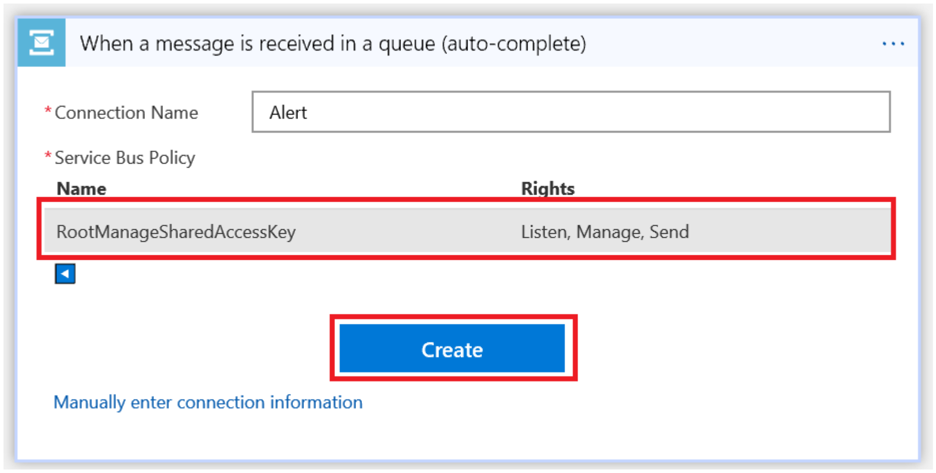

When a message is received in a queue (auto-complete)

*Connection Name: Alert

*Service Bus Namespace

Name	Resource Group	Location
stepanb-cpu	CPU	West Europe

17. Click suggested keys and Create.

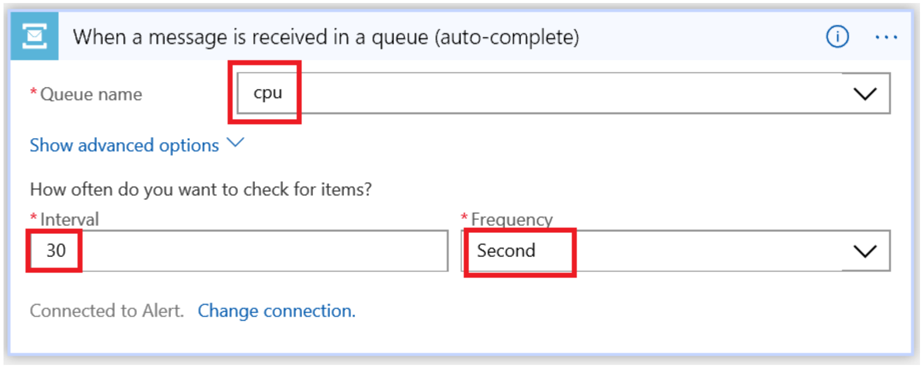

When a message is received in a queue (auto-complete)

*Connection Name: Alert

*Service Bus Policy

Name	Rights
RootManageSharedAccessKey	Listen, Manage, Send

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


When a message is received in a queue (auto-complete)

*Queue name: cpu

Show advanced options

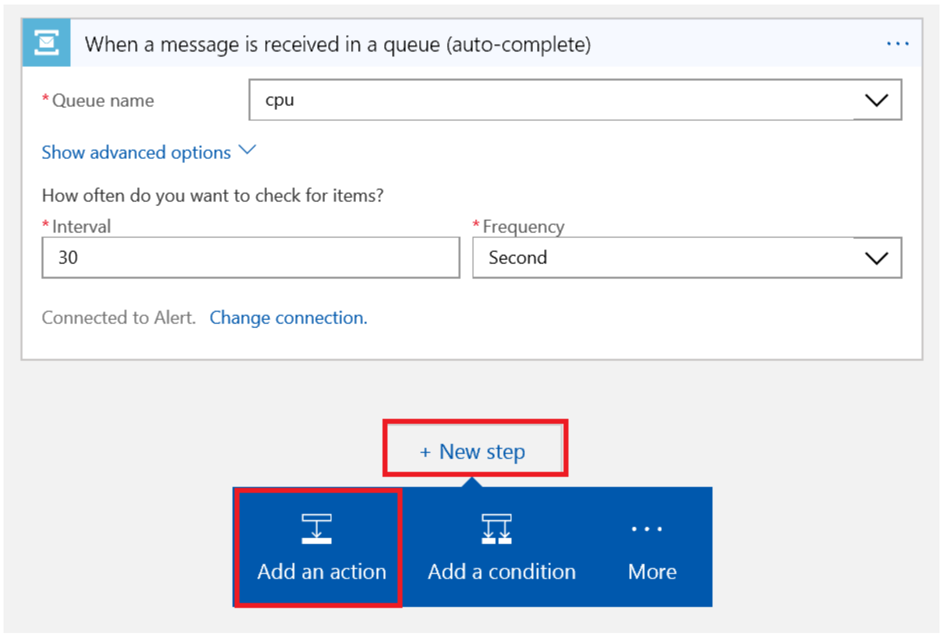
How often do you want to check for items?

*Interval: 30

*Frequency: Second

Connected to Alert. Change connection.

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



When a message is received in a queue (auto-complete)

*Queue name: cpu

Show advanced options

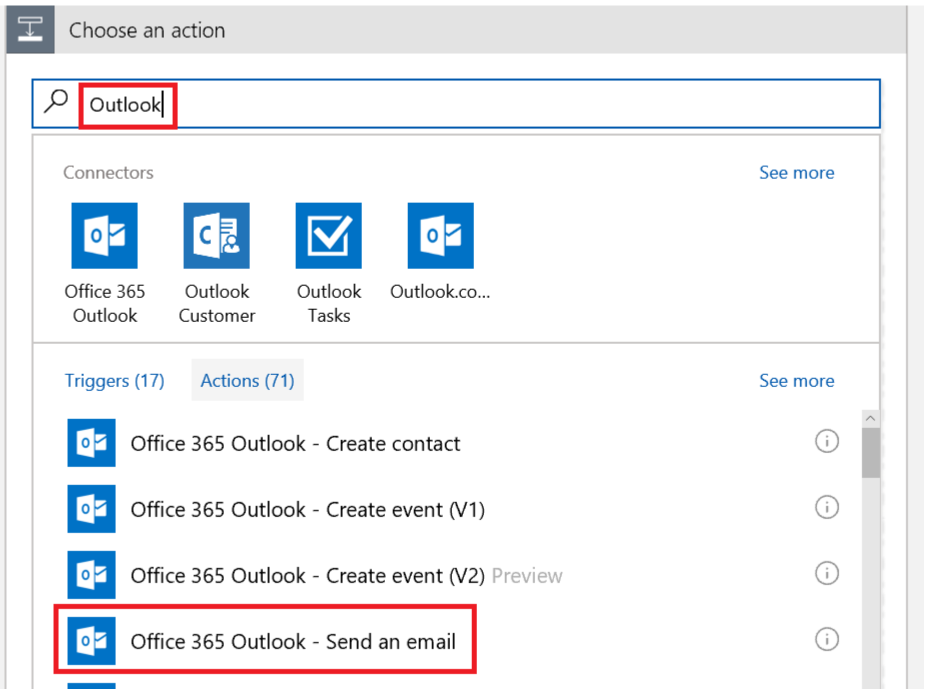
How often do you want to check for items?

*Interval: 30

*Frequency: Second

Connected to Alert. Change connection.

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



Choose an action

Outlook

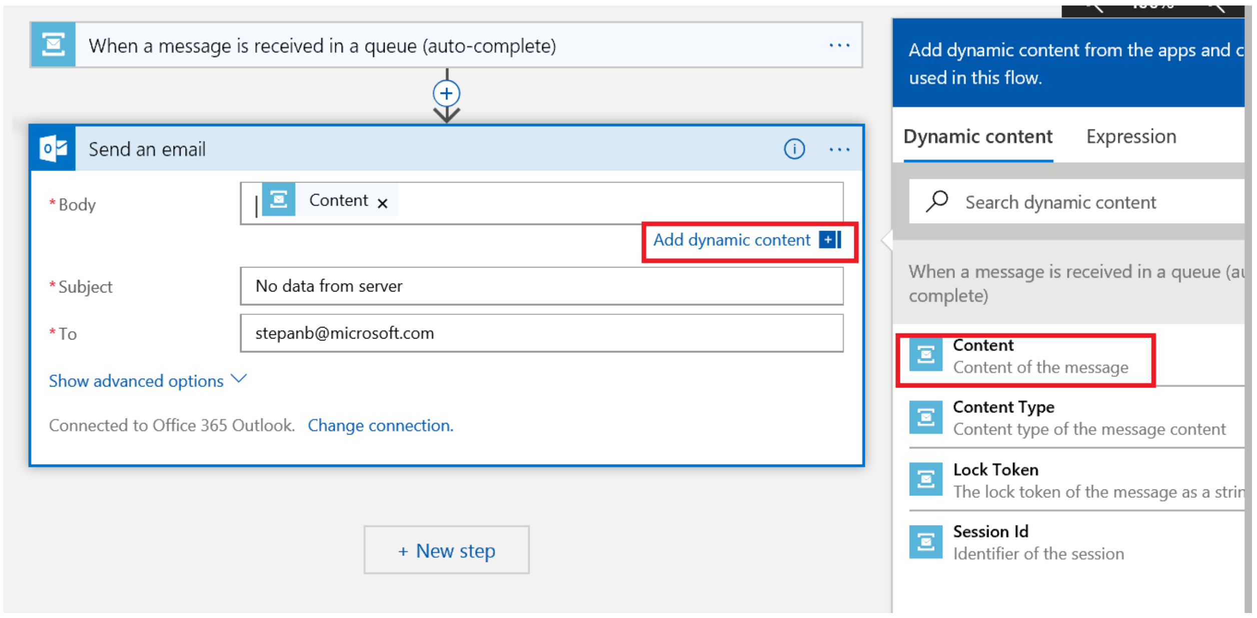
Office 365 Outlook - Create contact

Office 365 Outlook - Create event (V1)

Office 365 Outlook - Create event (V2) Preview

Office 365 Outlook - Send an email

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



When a message is received in a queue (auto-complete)

Send an email

*Body: Content ×

Add dynamic content

*Subject: No data from server

*To: stepanb@microsoft.com

Show advanced options

Connected to Office 365 Outlook. Change connection.

+ New step

Add dynamic content from the apps and c used in this flow.

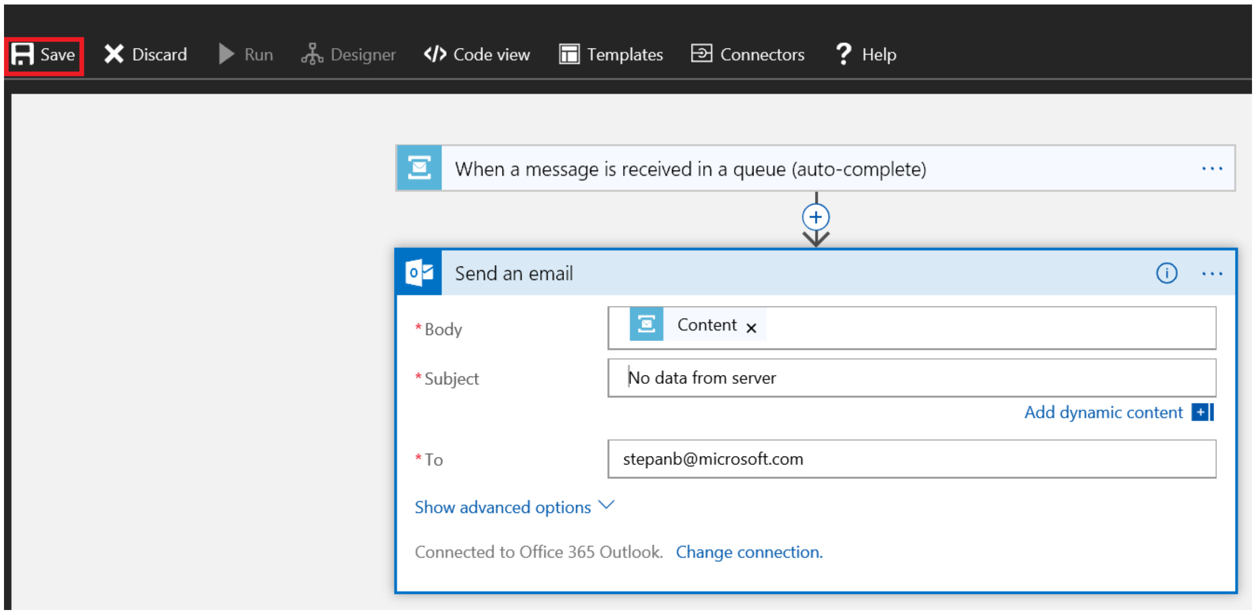
Dynamic content | Expression

Search dynamic content

When a message is received in a queue (au complete)

Content	Content of the message
Content Type	Content type of the message content
Lock Token	The lock token of the message as a strin
Session Id	Identifier of the session

23. Click *Save* and *Run*.



When a message is received in a queue (auto-complete)

Send an email

*Body: Content ×

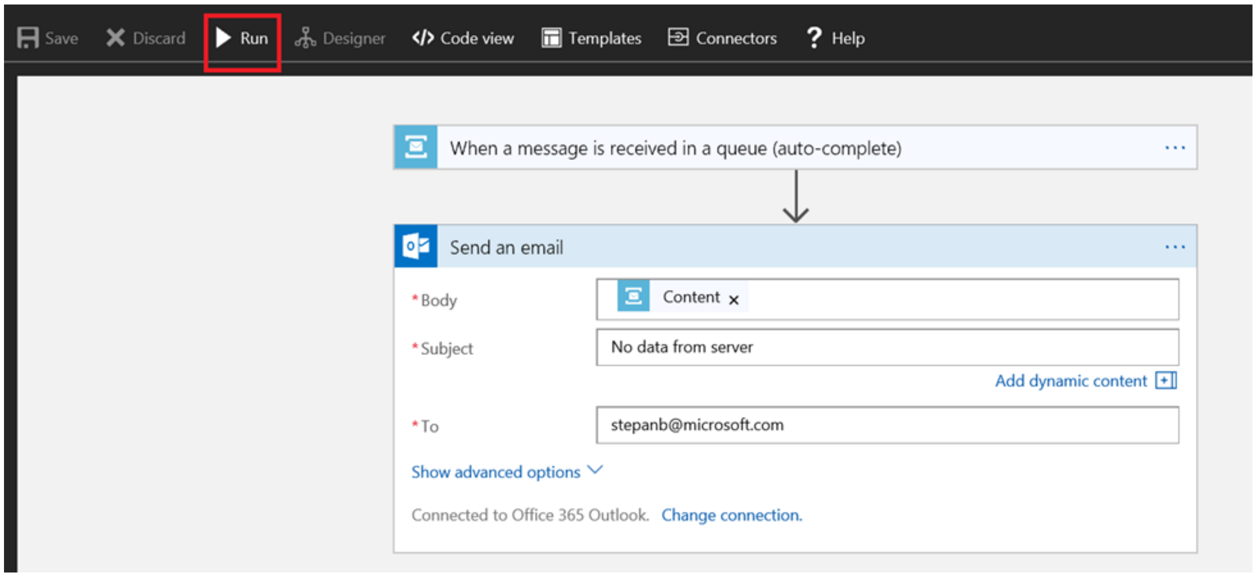
*Subject: No data from server

Add dynamic content

*To: stepanb@microsoft.com

Show advanced options

Connected to Office 365 Outlook. Change connection.



When a message is received in a queue (auto-complete)

Send an email

*Body: Content

*Subject: No data from server

Add dynamic content

*To: stepanb@microsoft.com

Show advanced options

Connected to Office 365 Outlook. Change connection.

24. If you stop application sending CPU usage you should get e-mail (after ~1 minute).