



# Ungoogle yourself or how to build your own digital (customer data) analytics platform with Tealium and Microsoft Azure

## Hands-on Guide

Maciek Stanasiuk

# Table of Contents

Introduction.....	3
Initial TMS setup.....	3
Tealium iQ implementation testing .....	6
Tealium as a CDP .....	7
Creating a new Azure Event Hub.....	8
Setting up Azure Event Hubs connector in Tealium EventStream .....	10
Audience processing via Tealium AudienceStream.....	12
Getting insights with Microsoft Azure.....	15
Real-time event data analysis using Azure Stream Analytics and Microsoft PowerBI.....	15
Azure Stream Analytics setup.....	15
Power BI data visualization .....	18
Big data analytics on Microsoft Databricks .....	21
Creating a Databricks cluster.....	21
Extra references .....	24

# Introduction

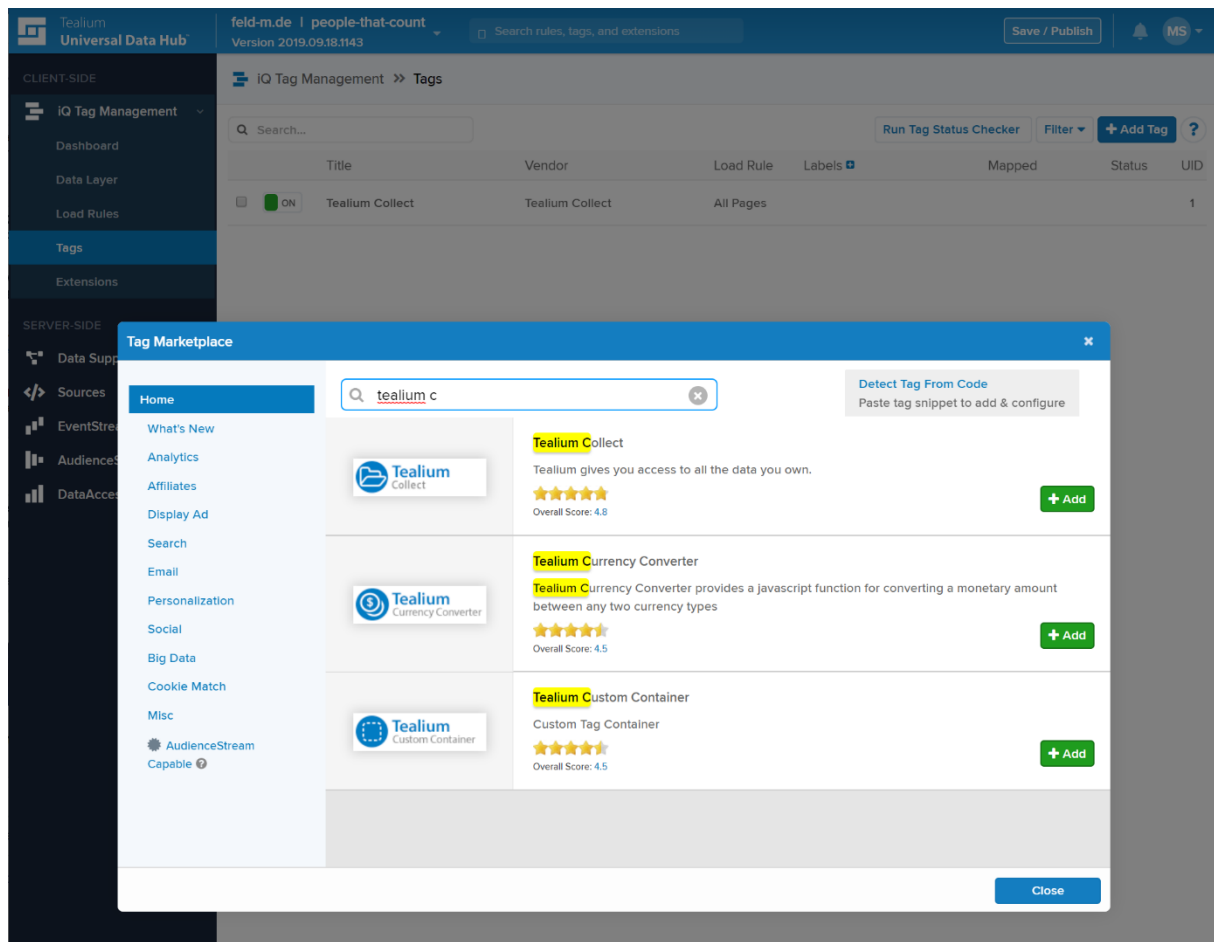
Welcome to the first ever workshop introducing Tealium iQ and Universal Data Hub along with Microsoft Azure as a full-fledged digital marketing/web/mobile/product data analysis and activation platform. I'm super happy you're here. Let's get started!

Maciek Stanasiuk (@stanasiukcom)

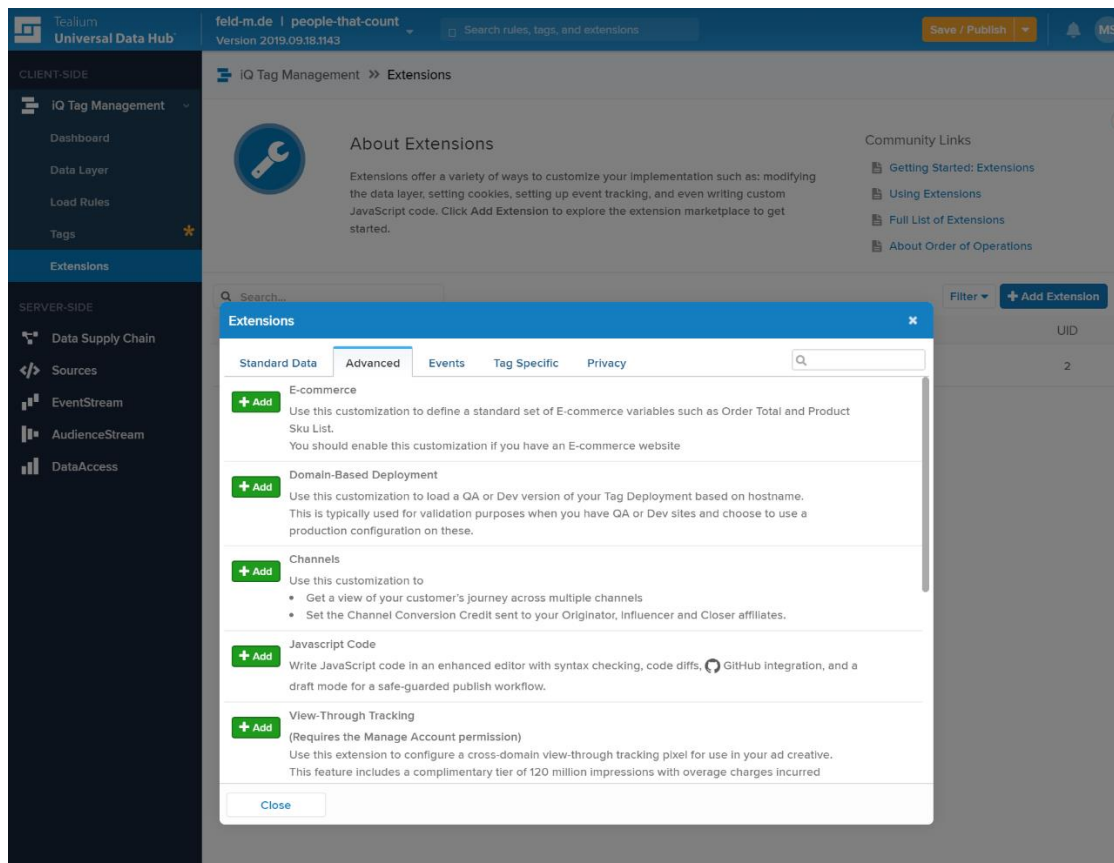
## Initial TMS setup

Login to Tealium iQ at [my.tealiumiq.com](https://my.tealiumiq.com) with your credentials. There, we'll need one tag for our client-side data collection needs and one extension to make our tracking setup more robust.

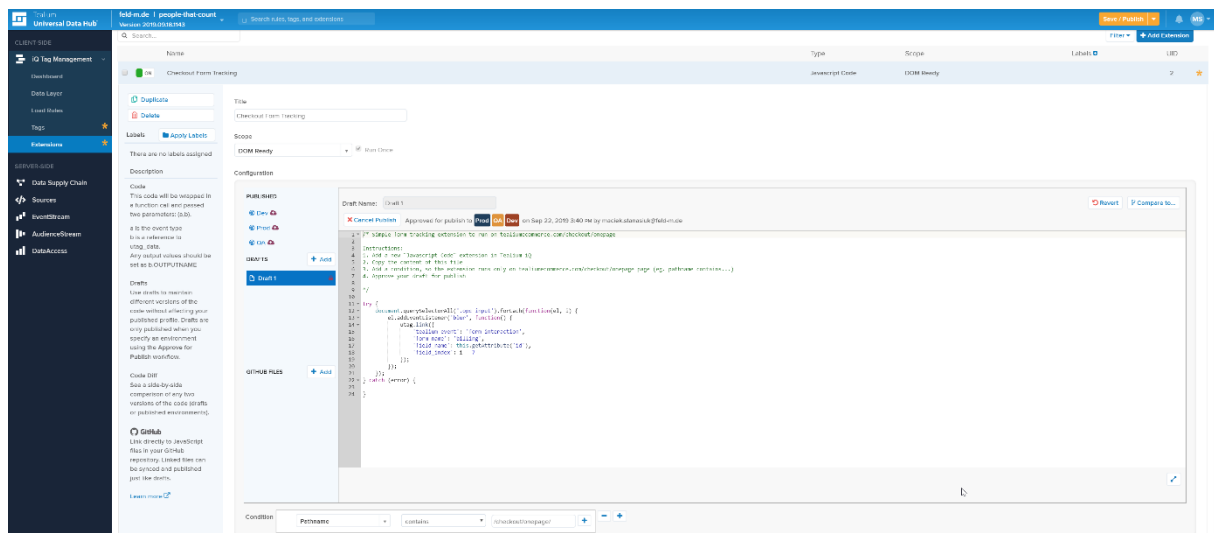
Go to **iQ Tag Management > Tags** and add the **Tealium Collect** tag. No configuration is needed, so simply go ahead and click **Finish**.



Then, go to **iQ Tag Management > Extensions**, click **Add Extension**, go to **Advanced** tab, and add a new **Javascript Code** extension where we will add our form tracking script.



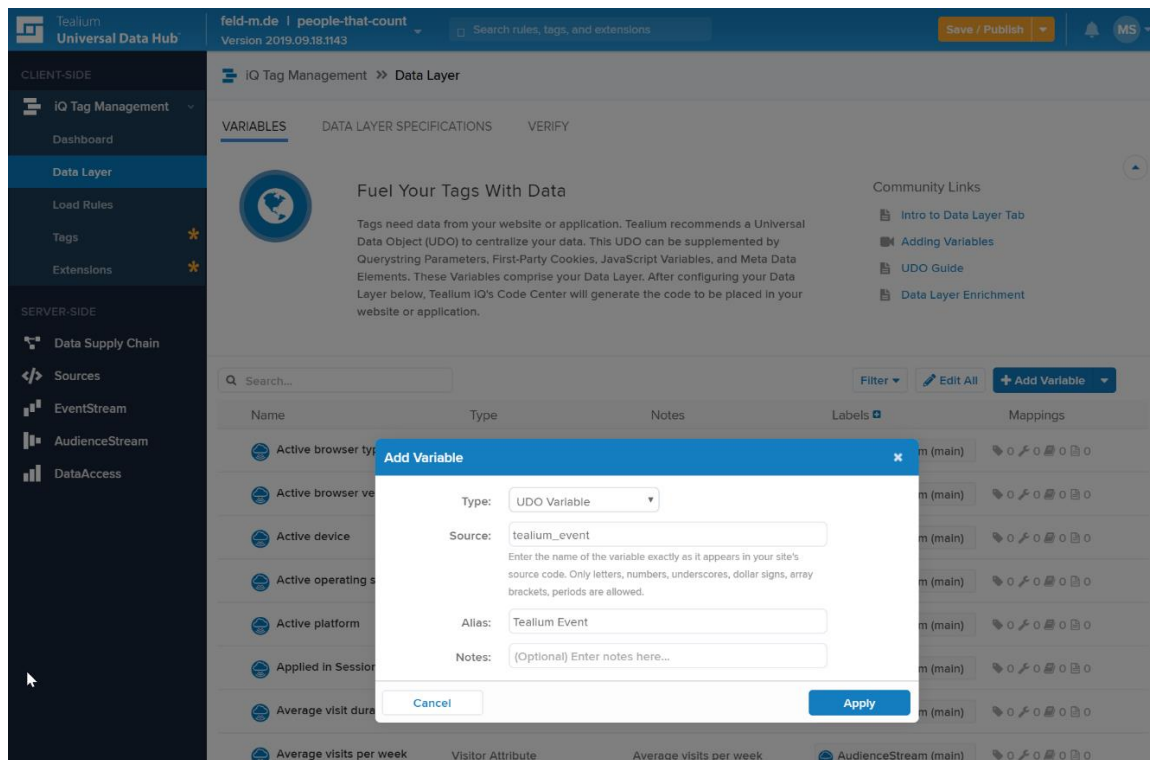
Name it “Checkout Form Tracking” and paste the contents of the **tealium-ecommerce-form-tracker.js** file there. Add appropriate conditions (eg. “**Pathname** contains /checkout/onepage/”) and approve the extension for publish.



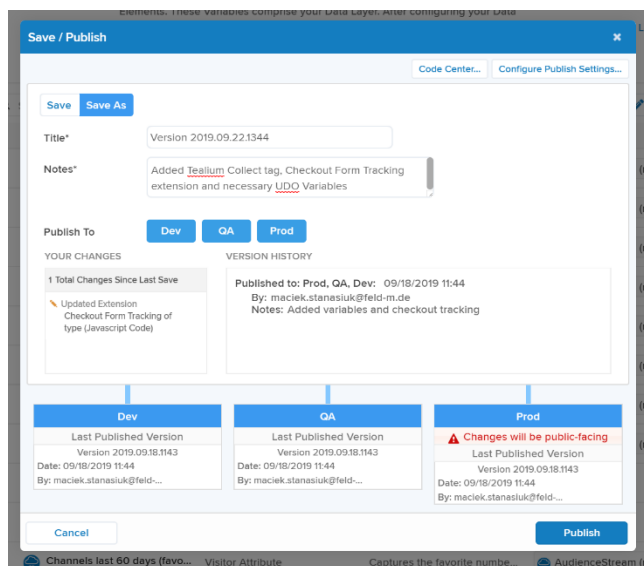
Finally, go to **iQ Tag Management > Data Layer** and add your new variables as **UDO variables**:

- tealium\_event,
- form\_name,

- field\_name,
- field\_index.



Once added, **Save & Publish** your profile to all the environments.



## Tealium iQ implementation testing

Go to [tealiumecommerce.com/training](https://tealiumecommerce.com/training) and enter your data:

- account = “feld-m.de”
- profile = your profile (eg. “ptc-1”)
- environment = “prod” (as long as you’ve published your changes to “Prod” environment)

Then, open the **Network** tab of your browser. Click around the store, try buying something and go into the checkout funnel. You should see the following Tealium Collect (eg. <https://collect.tealiumiq.com/feld-m.de/people-that-count/2/i.gif>) requests going out including our new data:

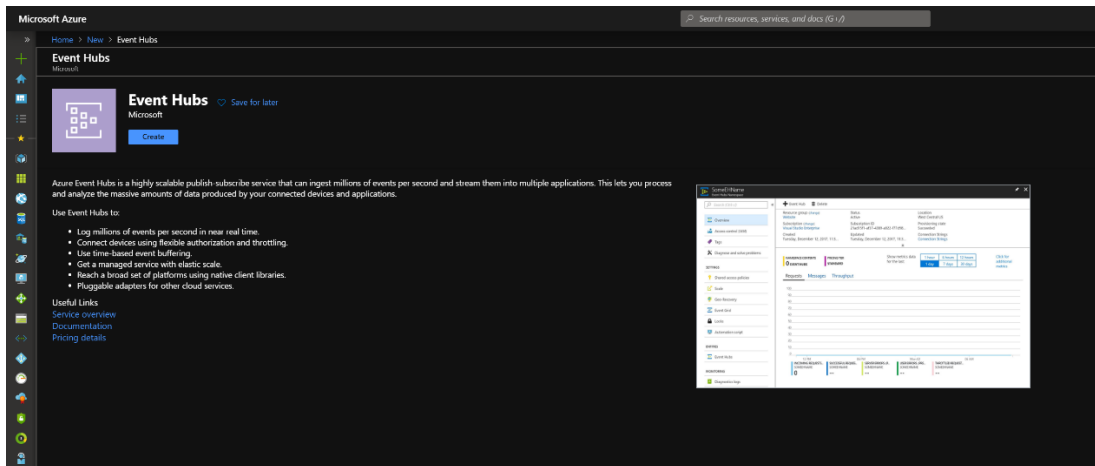
Voila! Now we can get into setting up our Azure Event Hub setup.

## Tealium as a CDP

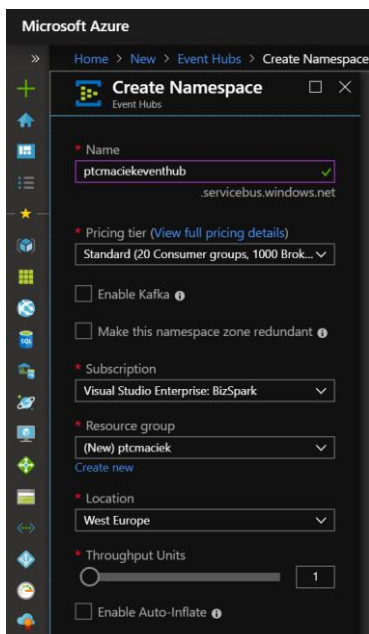
Over the next couple of steps we'll set up Tealium to send the event data to Azure and see how it can be used for audience definition and activation.

# Creating a new Azure Event Hub

Go to [portal.azure.com](https://portal.azure.com) and login with your Microsoft account. Then, click **Create a resource** and search for **Event Hubs**. Go ahead and create one!

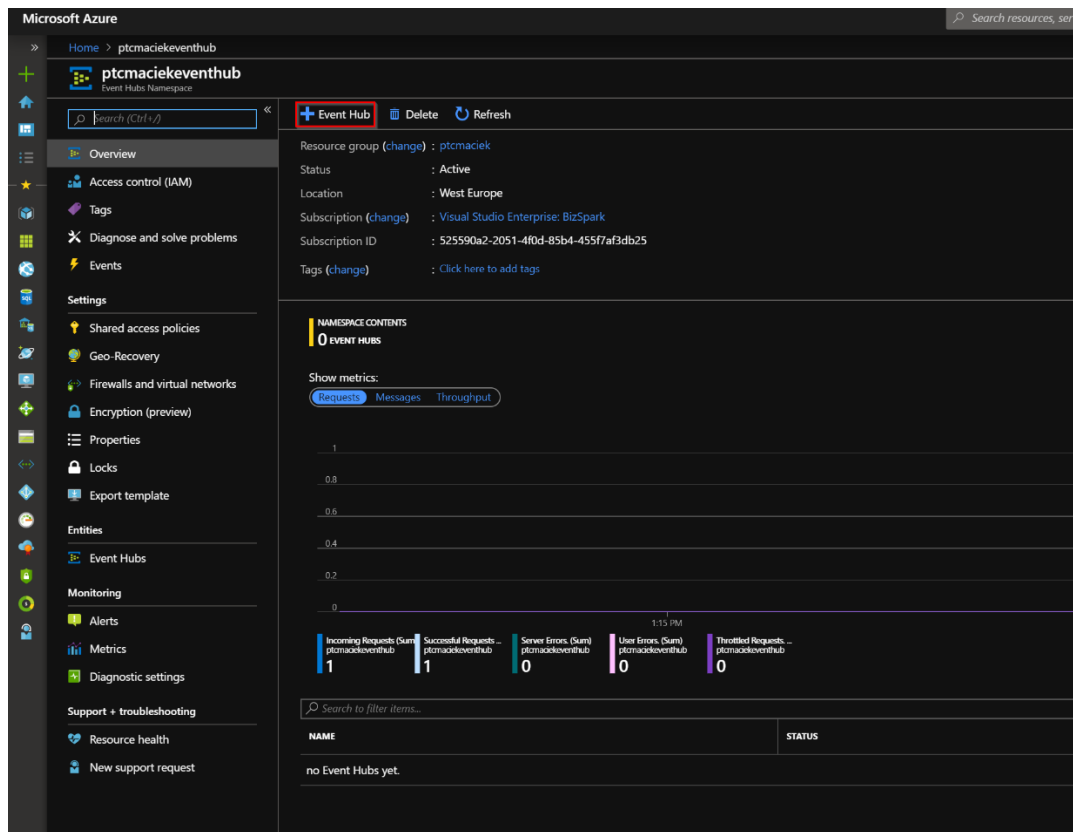


Name it within "ptcyournameeventhub" (eg. "ptcmaciekeventhub") schema, select **Standard** pricing tier, create a new resource group (ptcyourname), select **West Europe** location and **1 Throughput Unit**, click **Create** and wait for the deployment to finish.

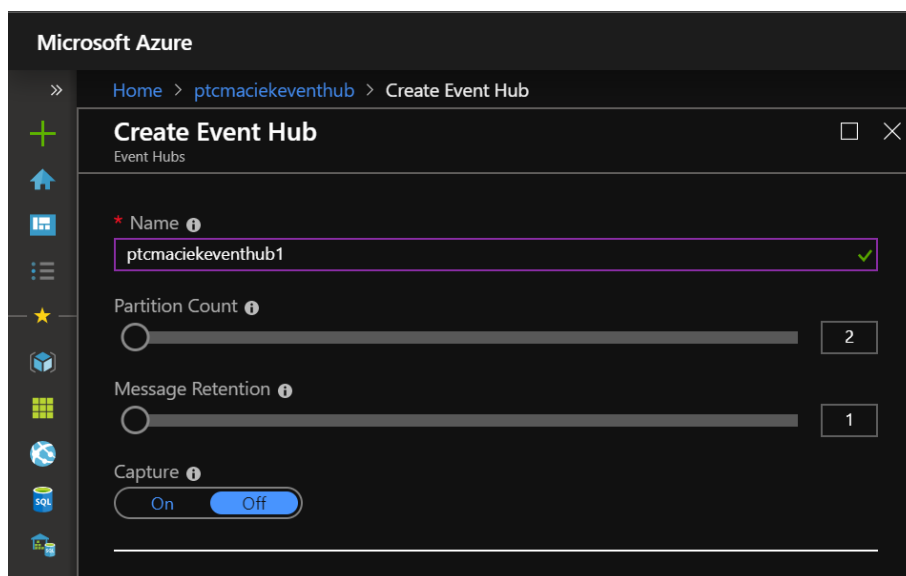


Afterwards, go to your new Event Hubs Namespace (just search for your event hub name) and click to **Add a new Event Hub** inside of it.

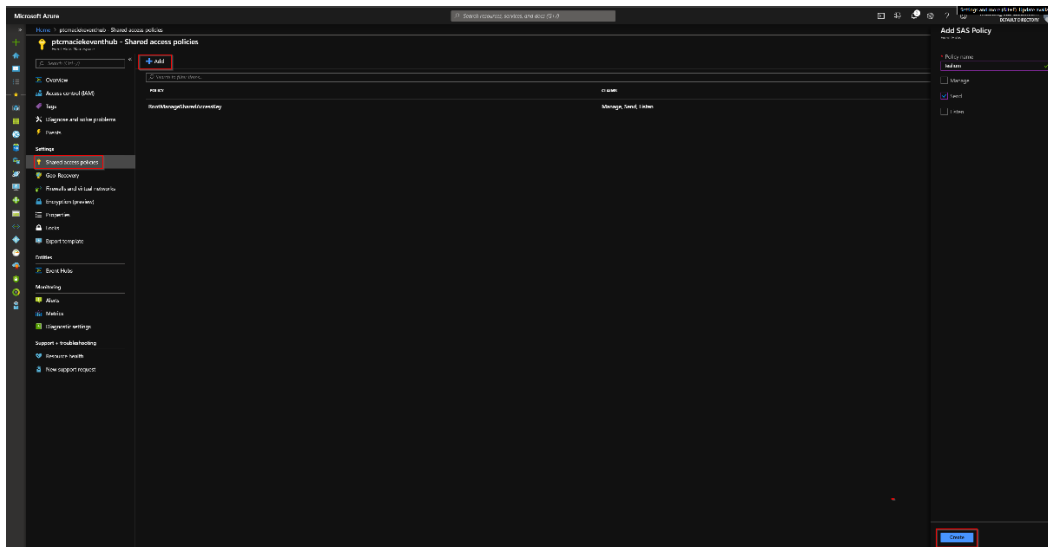




Name it with `ptcyournameeventhub1`, leave Partition Count and Message Retention settings at the lowest positions and keep Capture off – we’ll let Tealium handle data collection. Finally, create the hub.



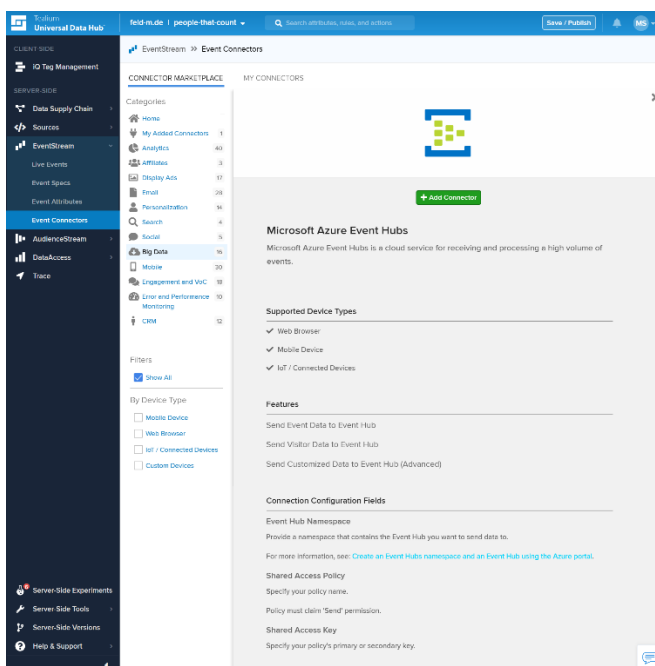
When you’re done, go back a level up to your Event Hubs Namespace and go to Shared access policies. There, **Add** a new policy called “Tealium”, let it **Send & Manage** and click **Create**.



Once created, copy its **Primary key** – you’ll need it at the next step.

## Setting up Azure Event Hubs connector in Tealium EventStream

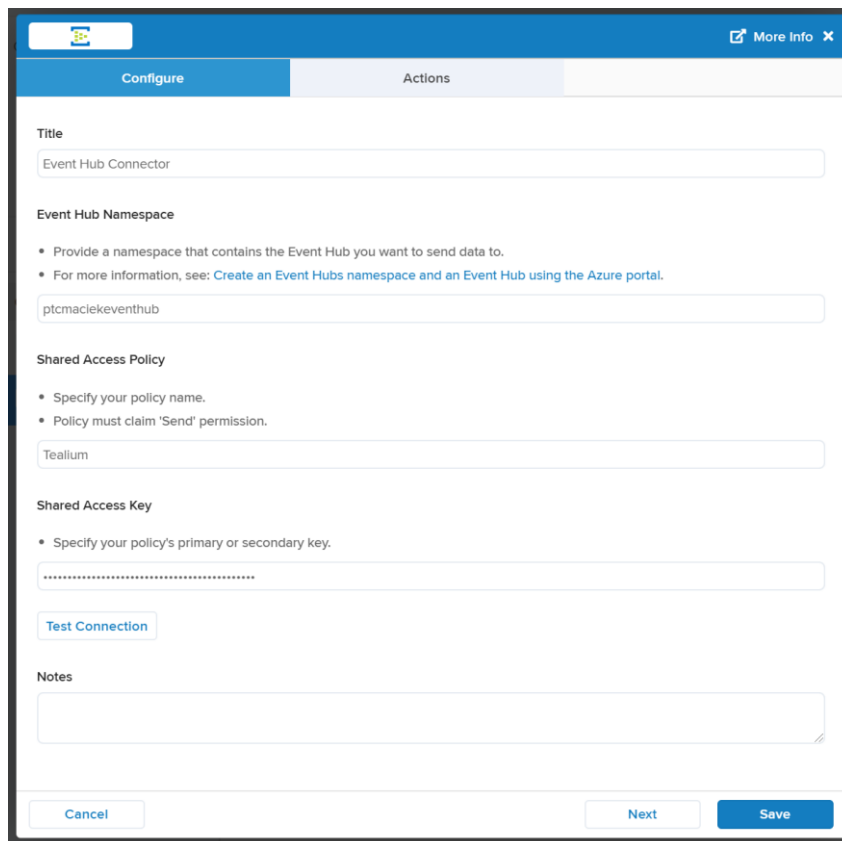
To have Tealium send all the event data to Azure we need to add a new connector inside EventStream.



To do so, go to **EventStream > Event Connectors > Big Data** and select **Microsoft Azure Event Hubs**. Click **Add Connector** and configure it as below:

- Title = “Event Hub Connector”,
- Event Hub Namespace = your namespace name,
- Shared Access Policy = “Tealium”,
- Shared Access Key = the key you’ve copied.

Click **Test Connection** and if everything's fine proceed to the **Actions** setup.



The screenshot shows the 'Configure' tab of the 'Event Hub Connector' configuration window. The window has a blue header with the Azure logo and a 'More Info' link. Below the header, there are two tabs: 'Configure' (active) and 'Actions'. The 'Configure' tab contains several sections with input fields and instructions:

- Title:** A text box containing 'Event Hub Connector'.
- Event Hub Namespace:** A section with two bullet points: 'Provide a namespace that contains the Event Hub you want to send data to.' and 'For more information, see: [Create an Event Hubs namespace and an Event Hub using the Azure portal.](#)'. Below the instructions is a text box containing 'ptcmaclekeventhub'.
- Shared Access Policy:** A section with two bullet points: 'Specify your policy name.' and 'Policy must claim 'Send' permission.'. Below the instructions is a text box containing 'Tealium'.
- Shared Access Key:** A section with one bullet point: 'Specify your policy's primary or secondary key.'. Below the instruction is a text box containing a series of asterisks '\*\*\*\*\*'.
- Test Connection:** A blue button labeled 'Test Connection'.
- Notes:** A large text area for notes.
- Buttons:** At the bottom of the window, there are three buttons: 'Cancel', 'Next', and 'Save'.

There, simply name your action, set Source to **All Events** (we want to keep it simple for now and just send all the data possible) and select your Event Hub. Check **Print Attribute Names** and **Save**.

The screenshot shows the 'Actions' tab in the Tealium Universal Data Hub interface. At the top, there's a 'More Info' link. Below it, the 'Actions' section shows 'Send Event Data to Event Hub' as the selected action, with a '+ Create Action' button. A list of actions follows, with 'Send Event Data to Event Hub' selected and a 'Remove' button. The configuration details for this action are shown below:

- Action Name:** Send Event Data
- Action Categories:** Big Data
- Source:** All Events
- Event Hub:**
  - Required.
  - Select an Event Hub to send data to.
  - If your configured policy only claims 'Send' permission, you will need to manually enter a custom value.

ptceventhubhubnew
- User Properties:** (expandable section)
- Broker Properties:** (expandable section)
- Print Attribute Names:** ☒

At the bottom, there are 'Cancel' and 'Save' buttons.

Now, **Save & Publish** your new setup. From now on, all the events being received by Universal Data Hub will be immediately passed to our Event Hub.

## Audience processing via Tealium AudienceStream

As a real CDP should Tealium Universal Data Hub also enables near real-time audience identification and activation via AudienceStream. This can be used in a multitude of ways, but some of the most common ones include, eg. cart abandoners retargeting. Let's take a look at how we could do it at tealiumcommerce.com.

Firstly, go to **AudienceStream > Visitor / Visit Attributes**. There, **Add Attribute** of a **Visitor** and select its type to be a **Badge**. Now name your badge "Cart Abandoner", select its icon and color and add the **Enrichments** – the logic for when we want to assign it. There are multiple ways of how it could be achieved, but let's go with the simplest one. We want to assign this badge to visitor when **visit ended** and they've added something to cart but didn't finish their transaction.

Select **VISIT ENDED** and Create a new Rule. Name it “Added to Cart” and set so that is being processed for visitors who have attribute “tealium\_event” that contains “shopping\_cart\_view”.

The screenshot shows the 'Edit Rule' window with a blue header. On the left is a circular icon with a curly brace and equals sign. The 'Title' field contains 'Added to Cart' with an 'Add Label' button below it. To the right is a 'Notes' section with a text area labeled 'Write your notes here...'. Below the title is the text 'This rule processes all visitors who...'. A condition bar shows '...have attribute' followed by a tealium\_event icon, a dropdown menu set to 'tealium\_event', a dropdown menu set to 'contains', and a text field with 'shopping\_cart\_view'. Below this bar is a '+ Attribute Condition' button. At the bottom left is a '+ OR' button, and at the bottom right are 'Cancel' and 'Save' buttons.

Then, create another rule, name it “Didn’t buy” and set it for “tealium\_event” attribute not containing “order\_confirmation\_view”.

The screenshot shows the 'Edit Rule' window with a blue header. On the left is a circular icon with a curly brace and equals sign. The 'Title' field contains 'Didn't buy' with an 'Add Label' button below it. To the right is a 'Notes' section with a text area labeled 'Write your notes here...'. Below the title is the text 'This rule processes all visitors who...'. A condition bar shows '...have attribute' followed by a tealium\_event icon, a dropdown menu set to 'tealium\_event', a dropdown menu set to 'does not contain', and a text field with 'order\_confirmation\_view'. Below this bar is a '+ Attribute Condition' button. At the bottom left is a '+ OR' button, and at the bottom right are 'Cancel' and 'Save' buttons.

In the end, your badge setup should be looking similarly to as follows:

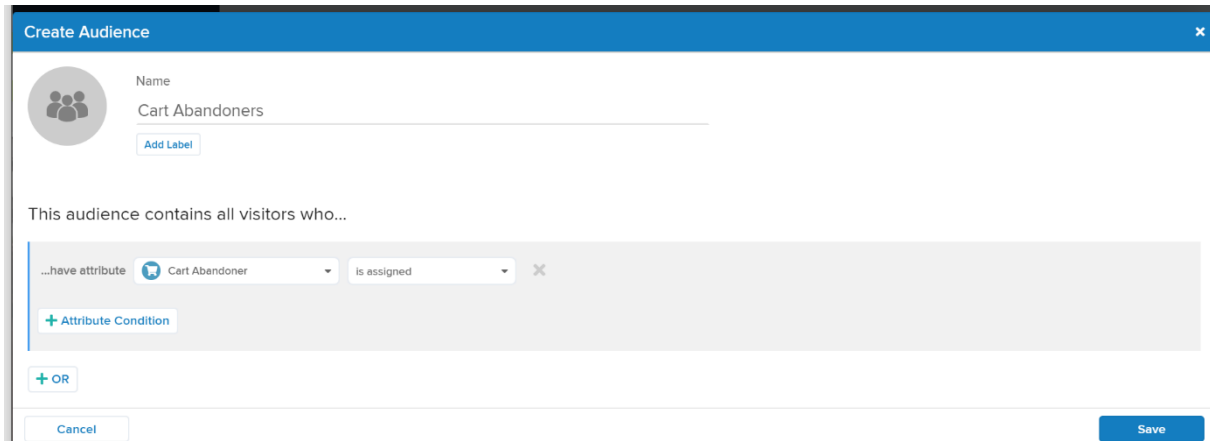
The screenshot shows the 'Add Attribute' interface for a 'Cart Abandoner' attribute. The interface is divided into several sections:

- Header:** 'Add Attribute' with a close button (X) in the top right corner.
- Attribute Info:** A shopping cart icon, the title 'Cart Abandoner', and tabs for 'Visitor' and 'Badge'.
- Properties:**
  - Title:** A text input field containing 'Cart Abandoner'.
  - Notes:** A text area with the placeholder 'Write your notes here...'.
  - Restricted Data:** A checkbox labeled 'Check the box if this Attribute includes a visitor's personal information. [Learn more about Restricted Data.](#)'.
  - AudienceDB:** A checkbox labeled 'Check the box to make this Attribute available to AudienceDB. [What is AudienceDB?](#)'.
  - Badge:**
    - Label:** A text input field.
    - Image:** A dropdown menu showing a shopping cart icon.
    - Theme:** Two buttons, 'Light' (selected) and 'Dark'.
    - Color:** A row of seven colored circles: brown, red, orange, yellow, green, blue (selected), and purple.
- Enrichments:**
  - Assign badge:** A modal window titled 'Assign badge' with a close button (X) in the top right corner.
  - Assign this badge to visitor:**
    - WHEN:** A dropdown menu showing 'VISIT ENDED'.
    - AND Added to Cart:** A rule entry: 'String `tealium_event` contains `shopping_cart_view`'.
    - AND Didn't buy:** A rule entry: 'String `tealium_event` does not contain `order_confirmation_view`'.
    - Choose a rule (optional):** A dropdown menu and a 'Create Rule' button.

At the bottom of the 'Add Attribute' interface are 'Cancel' and 'Finish' buttons.

You're now able to identify all the cart abandoners! Now, we only need to build a list (or a feed) of them – an audience – that we could, eg. pass to our email marketing tool or an advertising network for retargeting.

Go to **Audiences** and click **Add Audience**. Name it “Cart Abandoners” and define so that it includes all visitors who have the Cart Abandoner badge assigned.



Voila, our small Proof of Concept is done and we could now use the audience with whatever connector Tealium provides. Do you see any downsides of the current setup? Is there anything that could go wrong?

## Getting insights with Microsoft Azure

As we’ve already got our data collection setup ready (was pretty quick, wasn’t it?) it’s finally time to get some insights! We’ll firstly come up with a way to quickly analyze and visualize real-time data and then move on to starting up a platform for all our advanced analytics use cases.

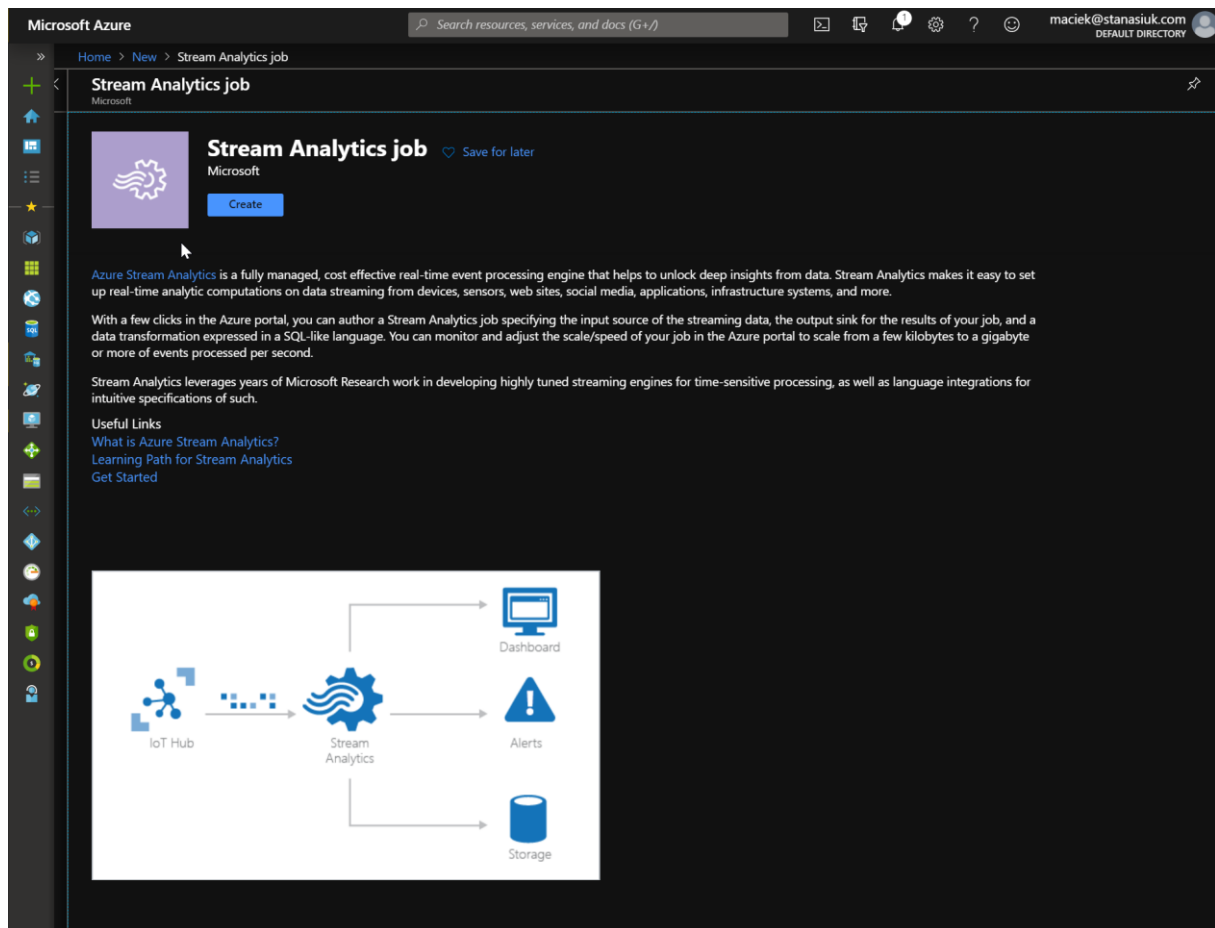
### Real-time event data analysis using Azure Stream Analytics and Microsoft PowerBI

Finally, let’s visualize our new, real-time data! To do this, we need to have a visualization tool (Power BI) and a way to provide it with the data (Azure Stream Analytics).

To start, go to **powerbi.com** and sign up. Once you are in, upgrade to a Pro free trial account at the top of the screen.

#### Azure Stream Analytics setup

Once you have your Power BI account set we can move on to feeding the events there. To do this, go to **portal.azure.com** and add a **Stream Analytics job** resource.



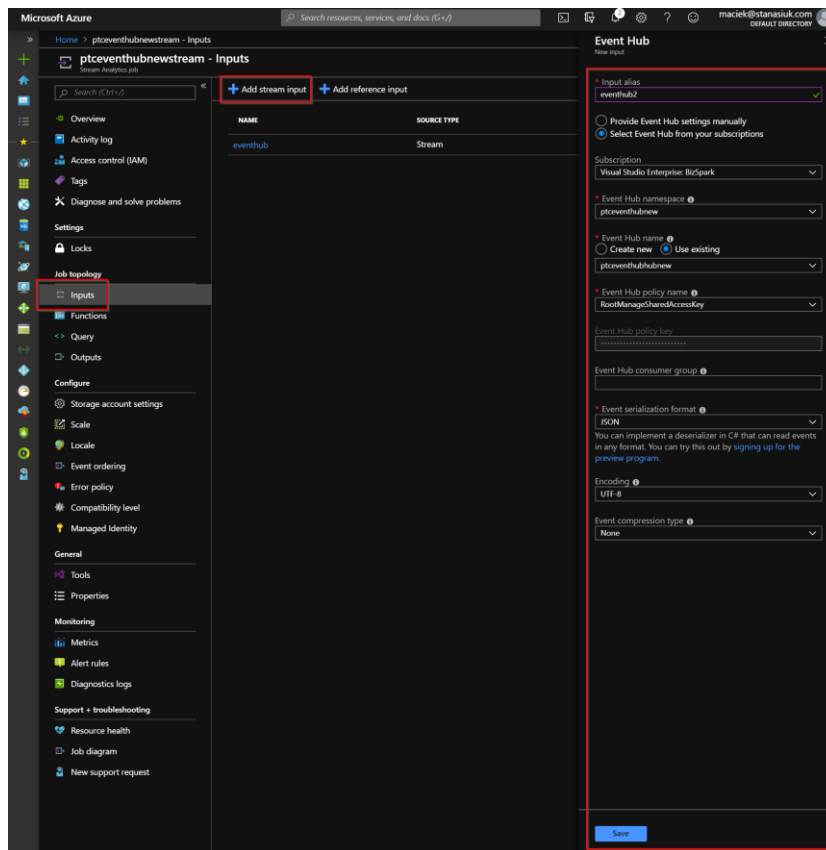
Name it, as usual with “ptc`yourname`eventhub”, select your resource group, set location to “West US” and choose only 1 Streaming Unit – it should be more than enough for our current needs.

The screenshot shows the 'New Stream Analytics job' form. The fields are filled as follows: Job name: ptcmaciekeventhub, Subscription: Visual Studio Enterprise: BizSpark, Resource group: cloud-shell-storage-west-europe, Location: West US, Hosting environment: Cloud, and Streaming units: 1.

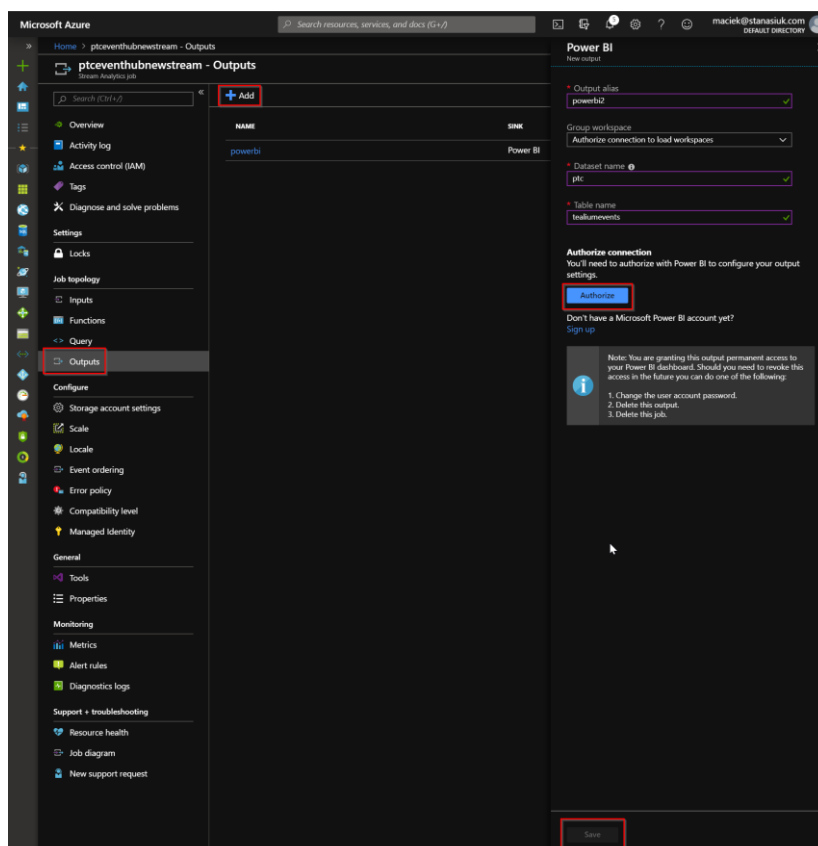
Once deployed, go to the resource. There we'll set up our super basic Stream Analytics logic – taking all the data from our Event Hub and push it to Power BI.

Go to **Inputs**, click **Add stream input** and select **Event Hub**. Name it “eventhub” and **Select Event Hub from your subscriptions** to use one you've previously created. Don't change any of the other settings.





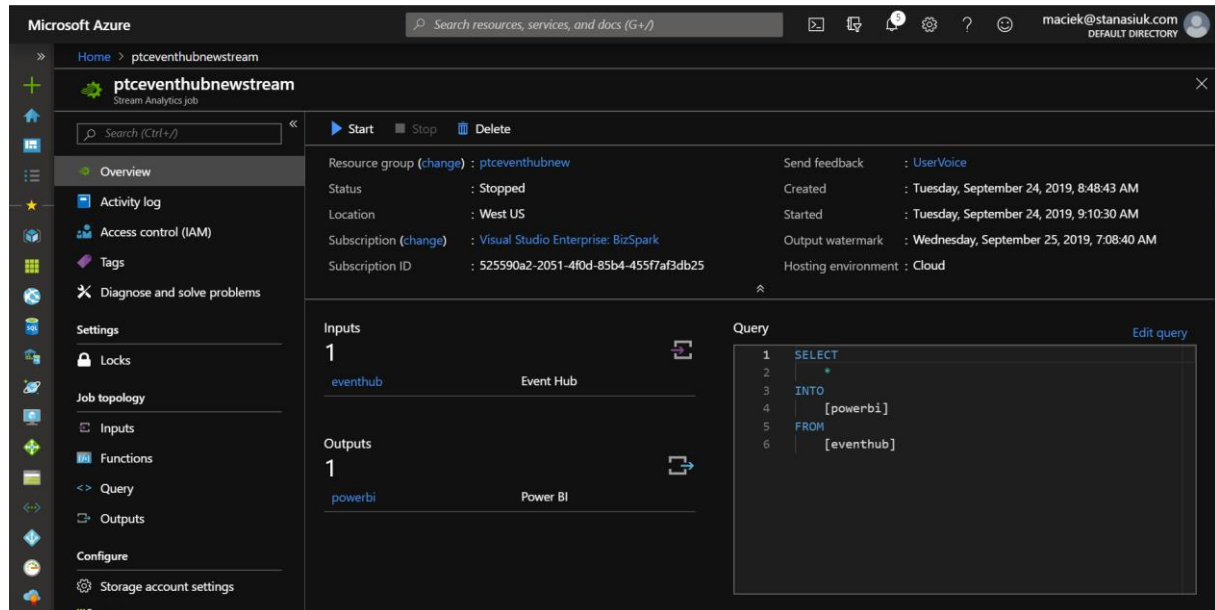
When the input passes its test, go to **Outputs**, click **Add** and select **Power BI**. Name the output “powerbi”, your dataset “ptc” and your table “tealiumevents”. Then, click **Authorize**, log in to your powerbi.com account and once authorized save your output.



Finally, go to **Query** and set your query to simply be:

```
SELECT
    *
INTO
    [powerbi]
FROM
    [eventhub]
```

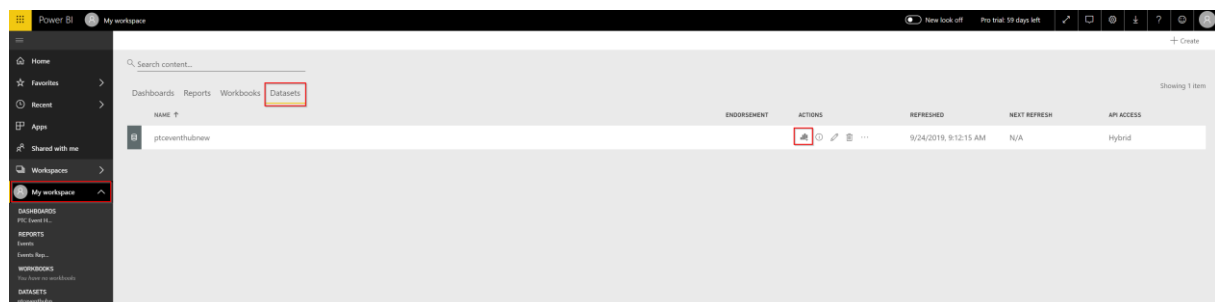
Save it, go back to **Overview** and **Start** your job.



Now, go back to **tealiumcommerce.com** and play around it to generate more events. Remember to make sure that your /training setup is still valid, though, so that events are sent to the correct place!

## Power BI data visualization

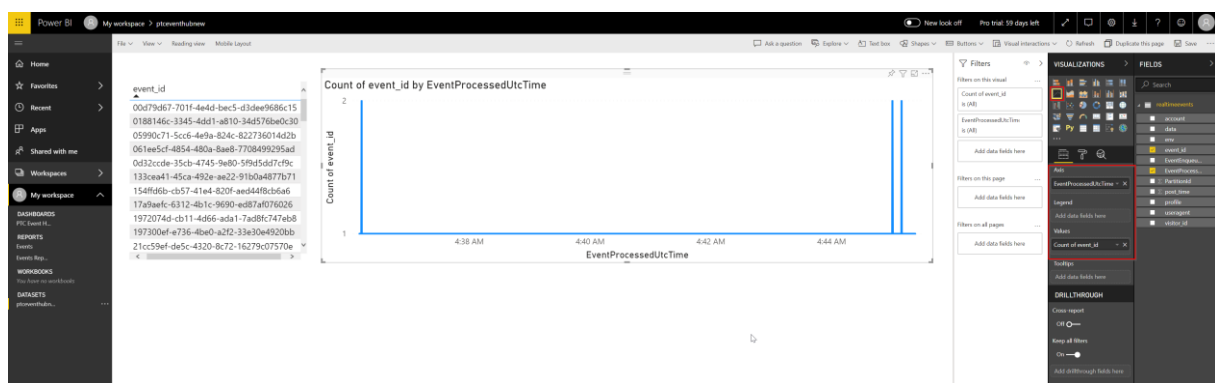
After a couple of minutes, go to **powerbi.com**, open **My workspace** and click on **Datasets**. There, you should already see your new dataset created by Stream Analytics. In **Actions** click on **Create report** to start working with its data.



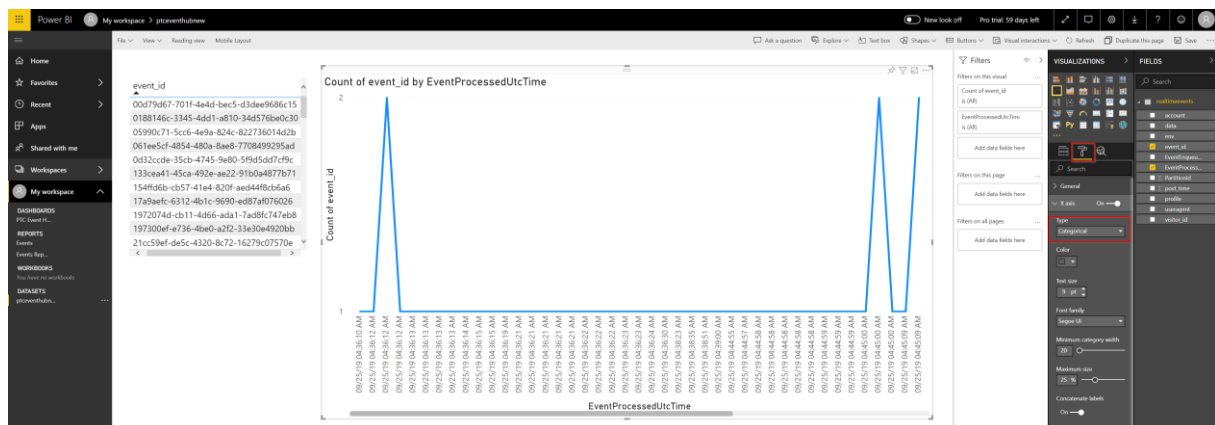
Let's start with validating if our data looks fine. Simply add a new **Table** visualization to your report and add **event\_id** as its values. You should get a list of all the incoming event ids.



Now, another very basic thing – incoming events timeline. Create a new **Line chart** visualization, add **event\_id** to the values (so it becomes **Count of event\_id**) and **EventProcessedUtcTime** to the axis.



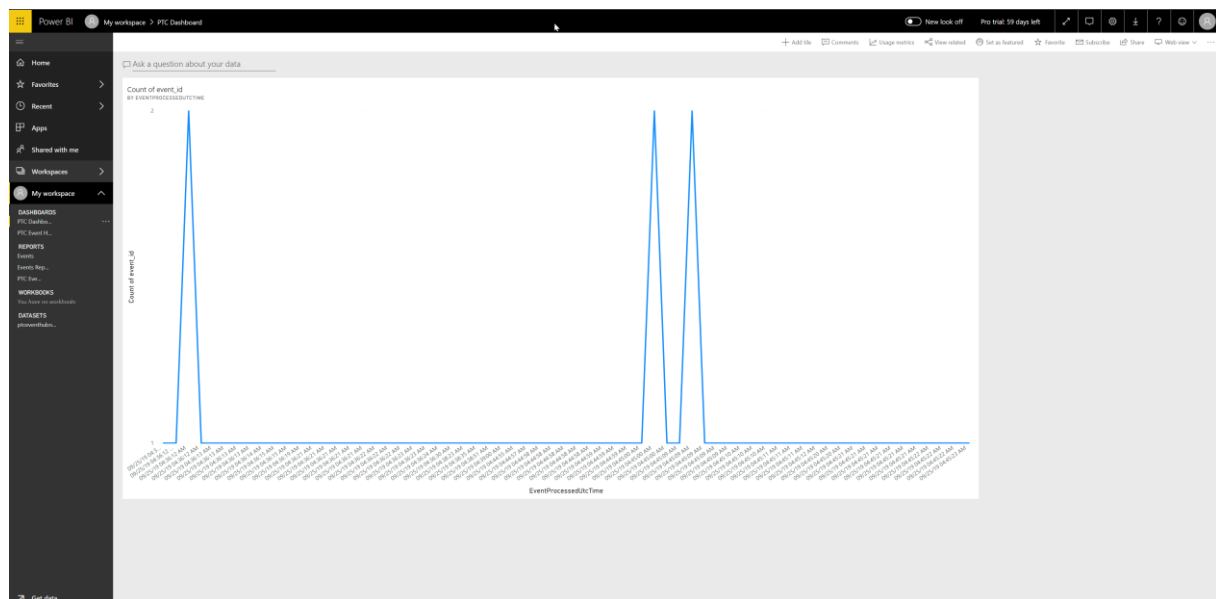
Then, to make it look nicer, go to **Format** and change the **X axis' Type** from Continuous to **Categorical**.



09/25/19 04:36:13 AM



Done! Your custom events real-time dashboard is live. Whenever you send new events to Tealium you should be able to see them here.

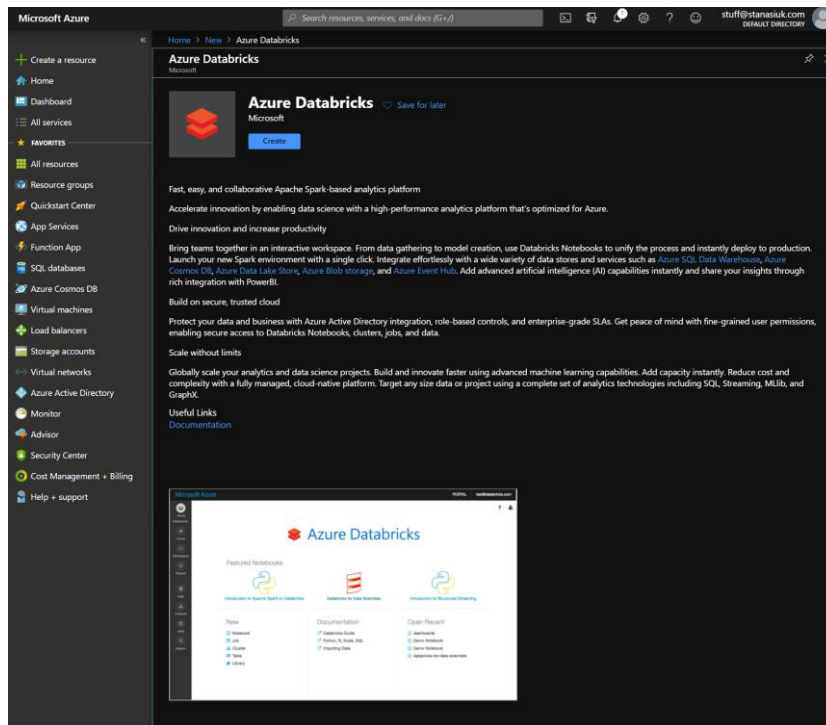


As you can see, as all data there is reported per second it doesn't look super spectacular (but it's still something!) Don't worry – it could be quite easily improved by grouping timestamps on the Power BI's side or by batching events in your Stream Analytics job. Also, as you can see, only a handful of parameters is available right out of the box – it could be adjusted as well. You could also run a proper real-time analytics process (eg. slicing, enriching and batching data) and visualize it exactly like that. Pretty cool, huh?

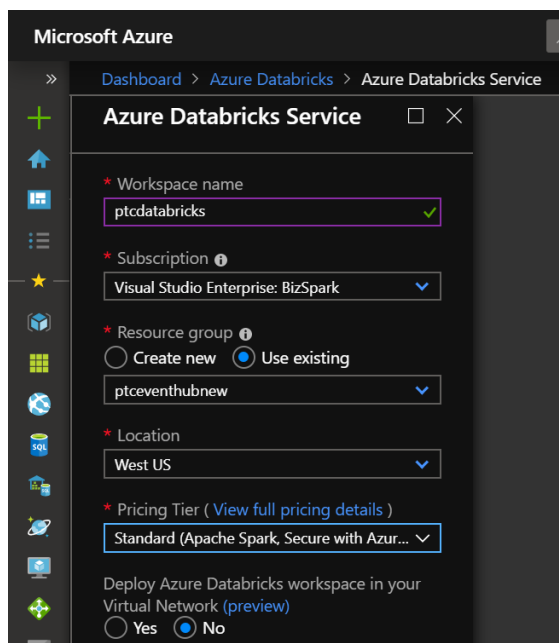
# Big data analytics on Microsoft Databricks

## Creating a Databricks cluster

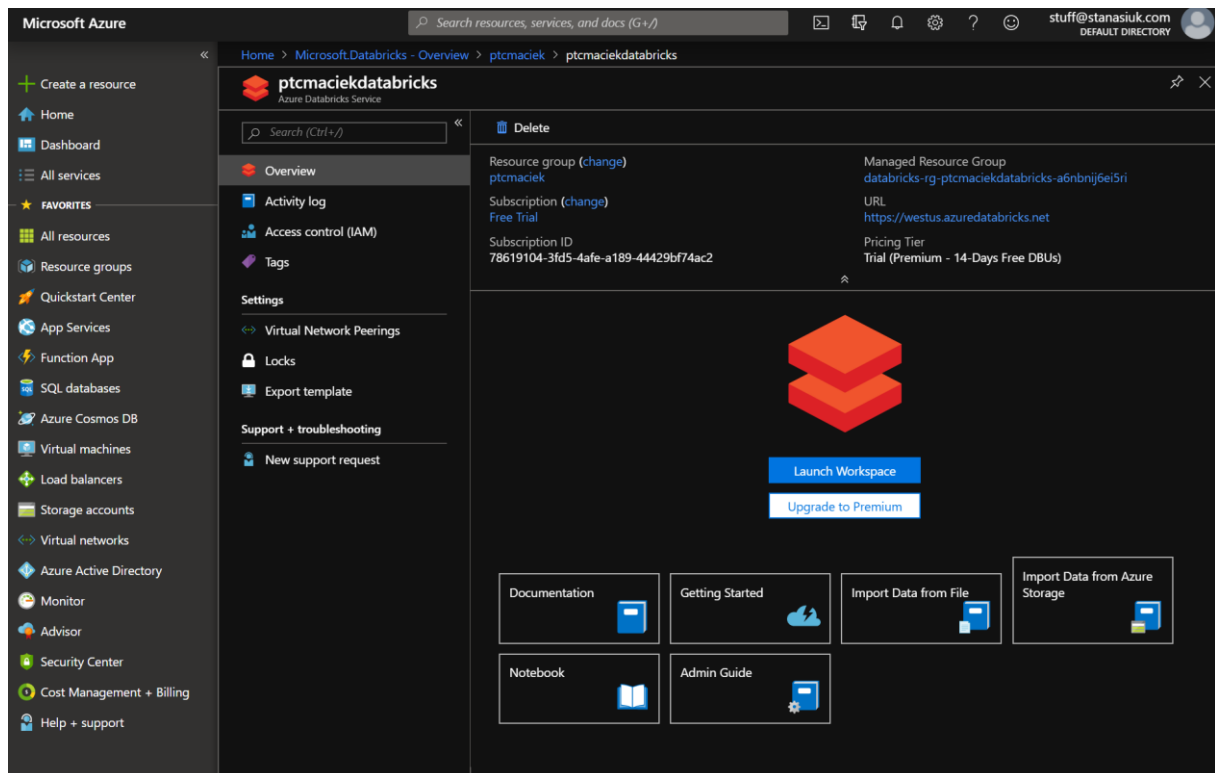
Go to Azure, search for **Azure Databricks** and **Create** a resource.



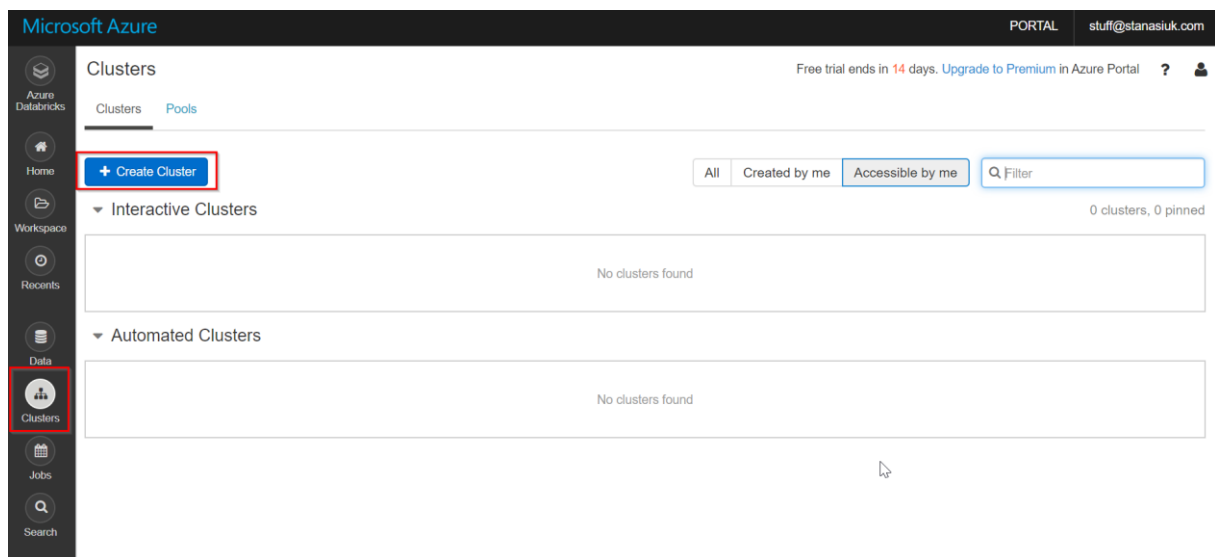
Name your workspace with “*ptcyournamedatabricks*”, select a **Visual Studio Enterprise: BizSpark** subscription, add it to your existing resource group, leave location as is and select a **Standard** pricing tier.



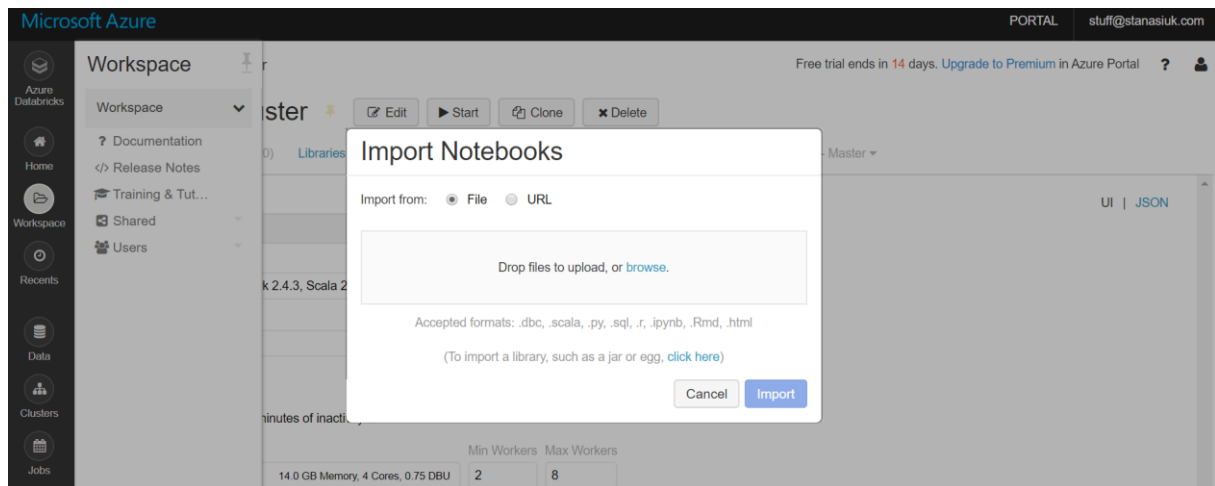
Once deployment finishes, go to your new resource and **Launch Workspace**. Now, there are a couple of things we want to do: create a new Apache Spark cluster and continue working with our Jupyter notebooks.



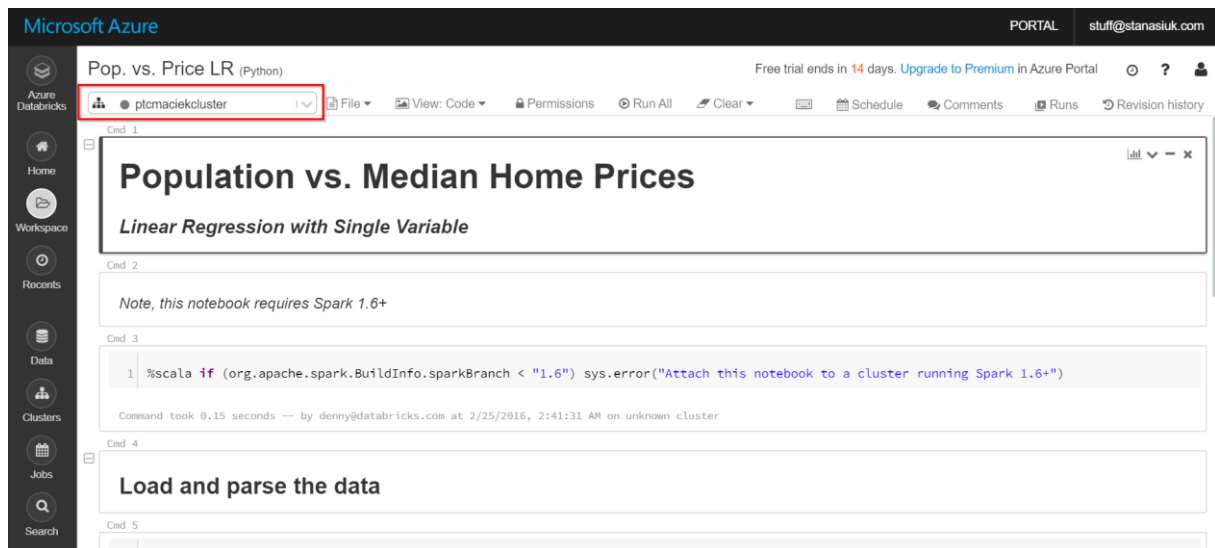
To create a cluster, go to **Clusters** and click **Create Cluster**.



There, name your cluster with "ptcyournamecluster", leave the settings as-is (we should be fine with them for now) and once again click **Create Cluster**. Once it's done, go to **Workspace**, click on the arrow and **Import**.



Import both attached notebooks, open the first one, attach your cluster to your notebook and continue with your work there.



## Finishing up

You're done! Congratulations! For extra questions or if you're craving more please see the links below. I hope you enjoyed it and now see how easy it is these days to ungoogle yourself!

## Extra references

- <https://tealium.com/docs-overview/>
- <https://docs.tealium.com/>
- <https://community.tealiumiq.com/t5/Universal-Data-Hub/Tealium-Collect-Tag-Setup-Guide/ta-p/11923>
- <https://community.tealiumiq.com/t5/iQ-Tag-Management/Extensions/ta-p/13649>
- <https://community.tealiumiq.com/t5/Universal-Data-Hub/Introduction-to-AudienceStream/ta-p/16087>
- <https://community.tealiumiq.com/t5/Universal-Data-Hub/Microsoft-Azure-Event-Hubs-Connector-Setup-Guide/ta-p/20936>
- <https://docs.microsoft.com/en-us/power-bi/service-real-time-streaming#using-the-streaming-dataset-ui-to-push-data>
- <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-power-bi-dashboard>
- <https://docs.databricks.com/spark/latest/data-sources/aws/amazon-redshift.html>
- <https://docs.databricks.com/user-guide/visualizations/index.html>