



Azure Data Explorer - The New Star in your Data Platform

Johan Ludvig Brattås &
Frank Geisler

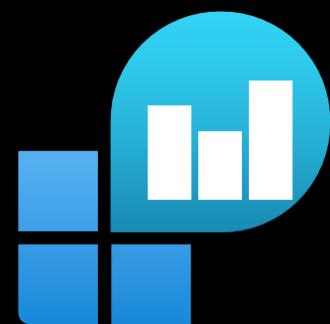
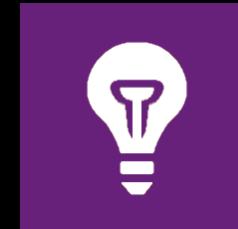
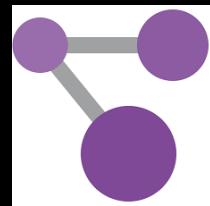


Agenda

- What exactly is Azure Data Explorer?
- A brief introduction to KQL
- Data Explorer – the hidden treasure of Azure analytic tools?
- Data Explorer in an IoT scenario
- Data Explorer with Synapse Link
- Graph Databases with Data Explorer

What exactly is Azure Data Explorer?

Azure Data Explorer



Azure Data Explorer

- Columnar
 - very good compression
- Append-Only
- Elastic scale
- Low latency ingestion (200 Mb/s per node)
- Fully managed
 - indexed by default



Azure Data Explorer

- Developed by Microsoft engineers in Israel to handle Azure's need for fast log and telemetry analytics
- Released as Application Insights in 2016
- Data Explorer went GA 2019



Azure Data Explorer

- RDBMS-like with databases, schemas & tables
- Powerful time series database
- Zero constraints
- Can be queried with SQL
- Better yet – the native language



Intro to KQL

- Kusto Query Language

No, Kusto – not Cousteau!



name of the table = SELECT *

« ▶ Ausführen | help/Samples

1 StormEvents

StormEvents ▾

StartTime	EndTime	Episodeld	EventId	State	EventType	InjuriesDirect
> 2007-01-01 00:00:00.0000	2007-01-27 14:00:00.0000	1.585	7.580	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-28 14:00:00.0000	1.585	7.586	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-28 21:00:00.0000	2.407	11.920	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-31 23:59:00.0000	2.407	11.923	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-30 10:34:00.0000	2.407	11.924	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-31 19:00:00.0000	1.575	7.499	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-31 10:00:00.0000	1.574	7.506	ILLINOIS	Flood	
> 2007-01-01 00:00:00.0000	2007-01-30 18:00:00.0000	1.574	7.505	ILLINOIS	Flood	
> 2007-01-01 00:00:00.0000	2007-01-20 10:24:00.0000	2.403	11.914	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-24 18:47:00.0000	2.408	11.930	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-27 10:27:00.0000	2.408	11.931	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-30 19:00:00.0000	1.575	7.498	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-22 18:49:00.0000	2.408	11.929	INDIANA	Flood	
> 2007-01-01 00:00:00.0000	2007-01-28 21:00:00.0000	2.407	11.921	INDIANA	Flood	

aggregation, sampling, ordering

The image displays three side-by-side screenshots of a data processing interface, likely Databricks or similar, illustrating various data manipulation techniques.

Left Screenshot: Shows a simple aggregation query.1 Trips
2 | count

Middle Screenshot: Shows an aggregation query with binning.<< ▶ Ausführen | help/Samples
1 FHV_Trips
2 | summarize count() by bin(pickup_datetime, 1d)

Right Screenshot: Shows an ordering and sampling query.<< ▶ Ausführen | help/Samples
1 FHV_Trips
2 | order by pickup_datetime desc
3 | take 100

Data Tables:

Left Table (Tabelle 1): A summary table showing the count of trips per day.

Count	Count
	1.547.471.776

Middle Table (Tabelle 1): A detailed table showing the count of trips per hour on January 1st.

	pickup_datetime	count_
>	2015-01-01 00:00:00.0000	77.789
>	2015-01-02 00:00:00.0000	61.832
>	2015-01-03 00:00:00.0000	81.955
>	2015-01-04 00:00:00.0000	62.691
>	2015-01-05 00:00:00.0000	71.063
>	2015-01-06 00:00:00.0000	81.722
>	2015-01-07 00:00:00.0000	93.090
>	2015-01-08 00:00:00.0000	103.978
>	2015-01-09 00:00:00.0000	100.710
>	2015-01-10 00:00:00.0000	103.838
>	2015-01-11 00:00:00.0000	80.717
>	2015-01-12 00:00:00.0000	87.703
>	2015-01-13 00:00:00.0000	94.221
>	2015-01-14 00:00:00.0000	94.558
>	2015-01-15 00:00:00.0000	100.055
>	2015-01-16 00:00:00.0000	105.375

Right Table (Tabelle 1): An ordered and sampled table of trip records.

Dispatching_bas...	pickup_datetime	dropoff_datetime	Pickup_location...	Dropoff_locatio...	Shared_Ride_Fl...
> B02510	2018-06-30 23:59:59...	2018-07-01 00:09:56...		89	133
> B02884	2018-06-30 23:59:59...	2018-07-01 00:25:38...		36	7
> B02871	2018-06-30 23:59:59...	2018-07-01 00:20:30...		255	97 true
> B01145	2018-06-30 23:59:59...	2018-07-01 01:10:12...			265
> B02864	2018-06-30 23:59:59...	2018-07-01 00:17:11...		65	79
> B00647	2018-06-30 23:59:59...	2018-07-01 00:13:59...			265
> B02887	2018-06-30 23:59:59...	2018-07-01 00:06:49...		129	82
> B02764	2018-06-30 23:59:59...	2018-07-01 00:18:04...		255	37
> B02395	2018-06-30 23:59:59...	2018-07-01 00:12:43...		244	166
> B02510	2018-06-30 23:59:59...	2018-07-01 00:19:35...		40	145
> B02764	2018-06-30 23:59:59...	2018-07-01 00:33:52...		41	218

filtering

```
1 FHV_Trips  
2 | where Dropoff_location_ID == 1  
3 | take 100
```

Tabelle 1 ▾

Dispatching_base_num	pickup_datetime	dropoff_datetime	Pickup_location_ID	Dropoff_location_ID	Shared_Ride_Fl...
B02884	2017-07-11 15:03:51...	2017-07-11 16:59:00...	162	1	
B02877	2017-07-11 15:18:15...	2017-07-11 16:00:10...	144	1	
B02877	2017-07-11 15:19:24...	2017-07-11 16:00:49...	114	1	
B02871	2017-07-11 16:05:43...	2017-07-11 16:58:43...	162	1	
B02871	2017-07-11 16:19:42...	2017-07-11 17:01:52...	158	1	
B02866	2017-07-11 16:35:23...	2017-07-11 17:44:18...	162	1	
B02764	2017-07-11 17:04:16...	2017-07-11 18:24:48...	75	1	
B02765	2017-07-11 17:08:49...	2017-07-11 18:49:24...	234	1	
B02884	2017-09-03 18:01:53...	2017-09-03 18:31:35...	13	1	
B02882	2017-09-03 18:02:42...	2017-09-03 18:48:35...	225	1	
B02884	2017-09-03 18:03:05...	2017-09-03 20:12:52...	248	1	
B02617	2017-09-03 18:05:54...	2017-09-03 18:29:18...	48	1	
B02879	2017-09-03 18:08:51...	2017-09-03 18:37:36...	230	1	

```
1 FHV_Trips  
2 | where Dropoff_location_ID == 1  
3 | take 100  
4 | project Dispatching_base_num, pickup_datetime, dropoff_datetime, Pickup_location_ID
```

Tabelle 1 ▾

Dispatching_base_num	pickup_datetime	dropoff_datetime	Pickup_location_ID
B02884	2017-07-11 15:03:51.0000	2017-07-11 16:59.0000	162
B02877	2017-07-11 15:18:15.0000	2017-07-11 16:00:10.0000	144
B02877	2017-07-11 15:19:24.0000	2017-07-11 16:00:49.0000	114
B02871	2017-07-11 16:05:43.0000	2017-07-11 16:58:43.0000	162
B02871	2017-07-11 16:19:42.0000	2017-07-11 17:01:52.0000	158
B02866	2017-07-11 16:35:23.0000	2017-07-11 17:44:18.0000	162
B02764	2017-07-11 17:04:16.0000	2017-07-11 18:24:48.0000	75
B02765	2017-07-11 17:08:49.0000	2017-07-11 18:49:24.0000	234
B02884	2017-09-03 18:01:53.0000	2017-09-03 18:31:35.0000	13
B02882	2017-09-03 18:02:42.0000	2017-09-03 18:48:35.0000	225

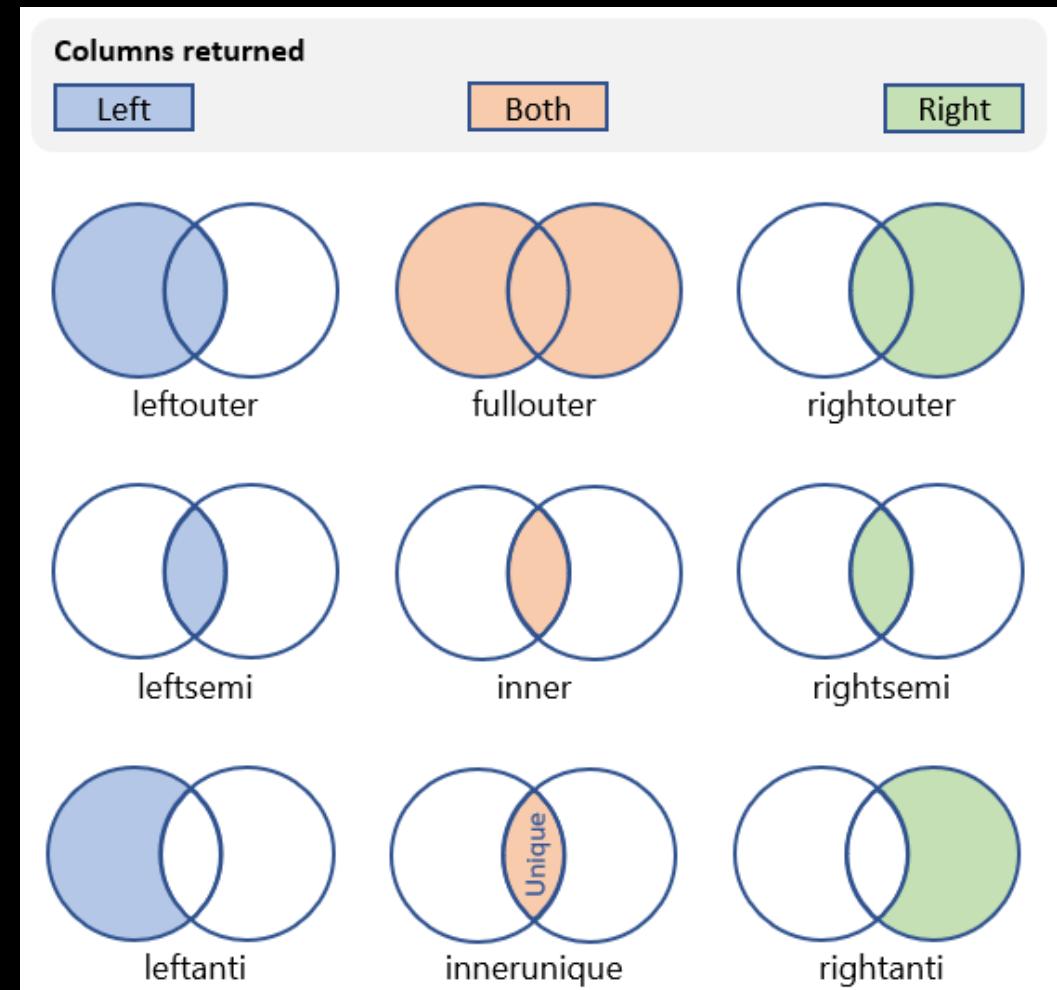
joins

```
1  StormEvents
2  | join kind=inner PopulationData on $left.State == $right.State
3  | project StartTime, EndTime, State, EventType, Population
4  | limit 15
```

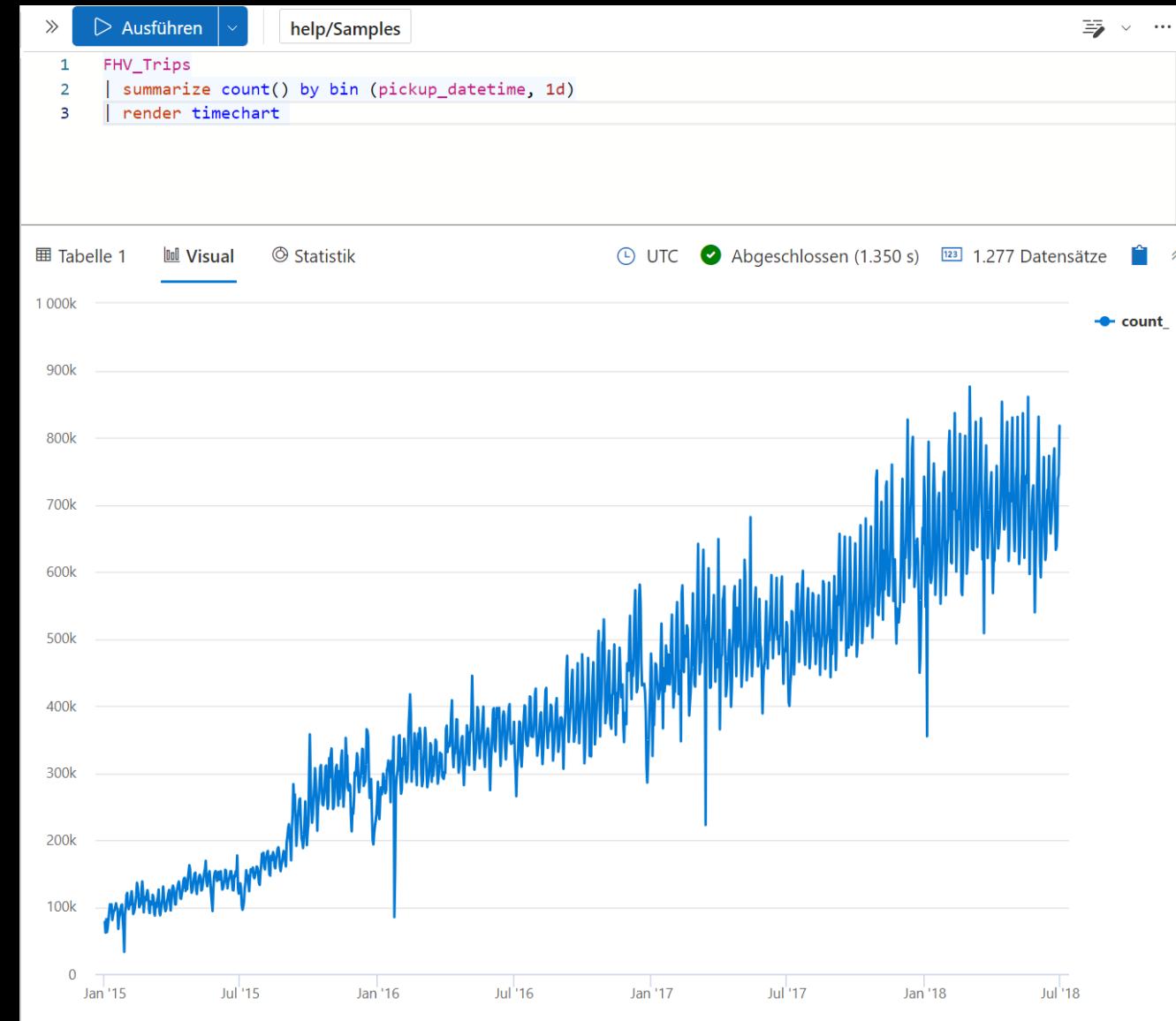
Tabelle 1					
	StartTime	EndTime	State	EventType	Population
>	2007-09-18 20:00:00.0000	2007-09-19 18:00:00.0000	FLORIDA	Heavy Rain	21.711.200
>	2007-09-20 21:57:00.0000	2007-09-20 22:05:00.0000	FLORIDA	Tornado	21.711.200
>	2007-12-11 21:45:00.0000	2007-12-12 16:45:00.0000	KANSAS	Flood	2.915.270
>	2007-12-13 09:02:00.0000	2007-12-13 10:30:00.0000	KENTUCKY	Flood	4.474.190
>	2007-12-20 07:50:00.0000	2007-12-20 07:53:00.0000	MISSISSIPPI	Thunderstorm Wind	2.971.280
>	2007-12-20 08:47:00.0000	2007-12-20 08:48:00.0000	MISSISSIPPI	Thunderstorm Wind	2.971.280
>	2007-12-20 10:32:00.0000	2007-12-20 10:36:00.0000	MISSISSIPPI	Tornado	2.971.280
>	2007-12-23 06:02:00.0000	2007-12-23 06:07:00.0000	OHIO	Thunderstorm Wind	11.701.900
>	2007-12-23 06:36:00.0000	2007-12-23 06:41:00.0000	OHIO	Thunderstorm Wind	11.701.900
>	2007-12-23 06:38:00.0000	2007-12-23 06:43:00.0000	OHIO	Thunderstorm Wind	11.701.900
>	2007-12-23 07:14:00.0000	2007-12-23 07:19:00.0000	OHIO	Thunderstorm Wind	11.701.900
>	2007-12-28 02:03:00.0000	2007-12-28 02:11:00.0000	MISSISSIPPI	Hail	2.971.280
>	2007-12-28 02:47:00.0000	2007-12-28 03:05:00.0000	MISSISSIPPI	Hail	2.971.280
>	2007-12-28 03:05:00.0000	2007-12-28 03:16:00.0000	MISSISSIPPI	Hail	2.971.280
>	2007-12-30 16:00:00.0000	2007-12-30 16:05:00.0000	GEORGIA	Thunderstorm Wind	10.723.700

joins

- different join types
 - innerunique (default) → only first match is included
(like the SSIS lookup component)
 - inner
 - leftouter, rightouter
 - fullouter
 - leftanti, rightanti
 - leftsemi, rightsemi



plotting



time series

```
1 FHV_Trips
2 | make-series trips=count() default=0 on pickup_datetime step 1d
```

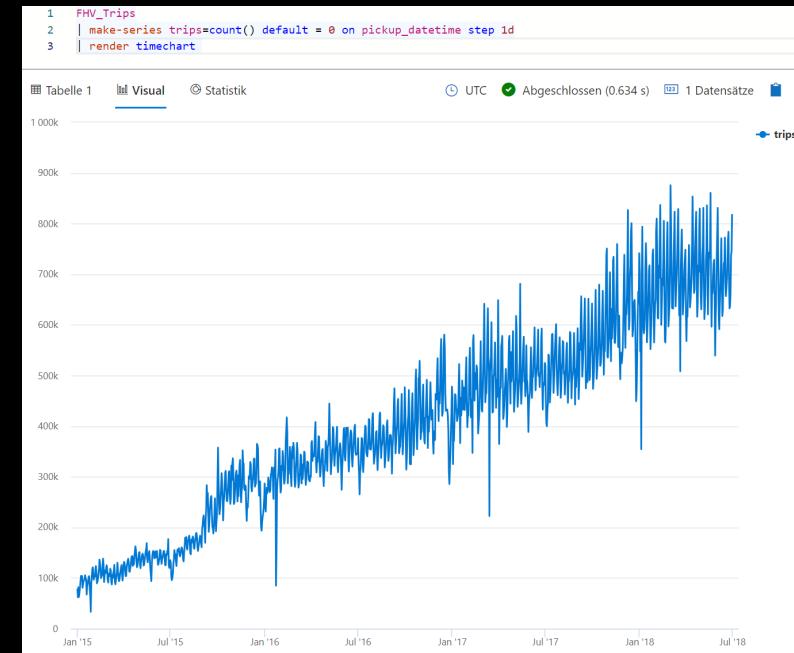
Tabelle 1 + Visuelles Element hinzufügen ⚡ Statistik 🔎 Suchen ⏱ UTC ✅ Abgeschlossen (0.784 s) 123 1 Datensätze 🌐 📁 📈

trips	pickup_datetime
[77789,61832,81955,62691,71063,81722,93090,103978,100710,103838,80717,87703,942...	"2015-01-01T00:00:00.000000Z","2015-01-02T00:00:00.000000Z","2015-01-03T00:0...

```
1 FHV_Trips
2 | make-series trips=count() default = 0 on pickup_datetime step 1d
3 | project series_stats(trips)
```

Tabelle 1 + Visuelles Element hinzufügen ⚡ Statistik 🔎 Suchen ⏱ UTC ✅ Abgeschlossen (2.121 s) 123 1 Datensätze 🌐 📁 📈

series_stats_trips_min	series_stats_trips_min_idx	series_stats_trips_max	series_stats_trips_max_idx	series_stats_trips_avg	series_stats_trip...
33.171	26	875.723	1.156	402.744,36256851995	191.777,52216295246



time series

- you can also
 - calculate moving averages
 - apply linear regression
 - detect patterns
 - forecast
 - ...

expand JSON

	AreaInfo	Location	Confirmed	Active
>	{"Alpha3CountryCode":null,"IsInternational":false,"CovidTrac...		{"ReportDate":null,"Change":1009572,"Source":null,"Value":1...	{"ReportDate":n...
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":42778,"Source":null,"Value":318...	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":3827,"Source":null,"Value":3718...	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":883,"Source":null,"Value":12296...	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":0,"Source":null,"Value":297215}	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":315,"Source":null,"Value":294779}	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":562,"Source":null,"Value":274875}	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":200,"Source":null,"Value":268962}	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":309,"Source":null,"Value":117081}	
>	{"Alpha3CountryCode":"USA","IsInternational":false,"CovidTr...		{"ReportDate":null,"Change":95,"Source":null,"Value":107859}	



```
1 Covid19_map2
2 | project - LastUpdated, AreaInfo
3 | mv-expand AreaInfo
4 | extend CountryName = tostring(AreaInfo.CountryName)
5 | project-away AreaInfo
6 | where CountryName <> ''
7 | take 100
```

Kusto is too hard?

```
1 Explain
2 SELECT · State, · count(*) · as · number_of_tornados · from · StormEvents
3 where · EventType · = · 'Tornado'
4 Group · by · State
5
6 StormEvents
7 | where (EventType == "Tornado")
8 | summarize number_of_tornados=toint(count()) by State
9 | project State, number_of_tornados
```

■ Tabelle 1 + Visuelles Element hinzufügen ◎ Statistik 🔎 Suchen ⏱ UTC ✓ A

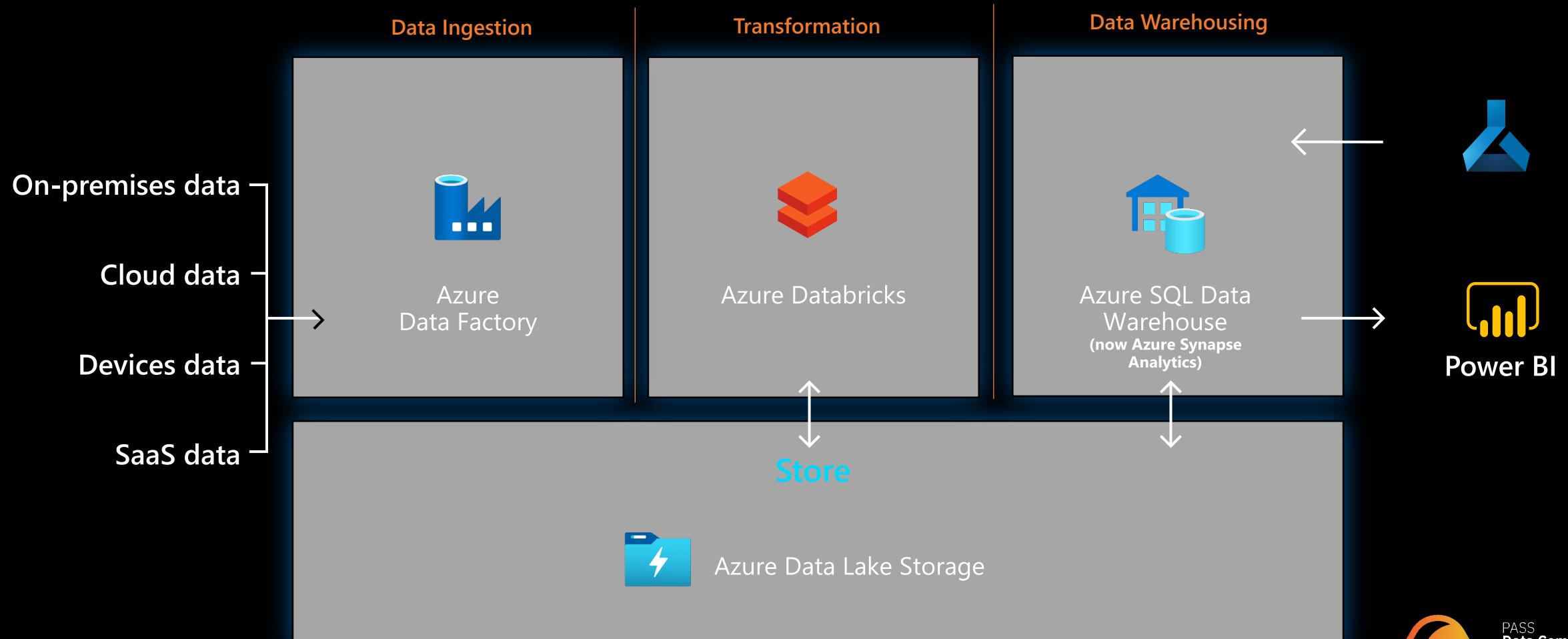
Query

> StormEvents | where (EventType == "Tornado") | summarize number_of_tornados=toint(count()) by State ...

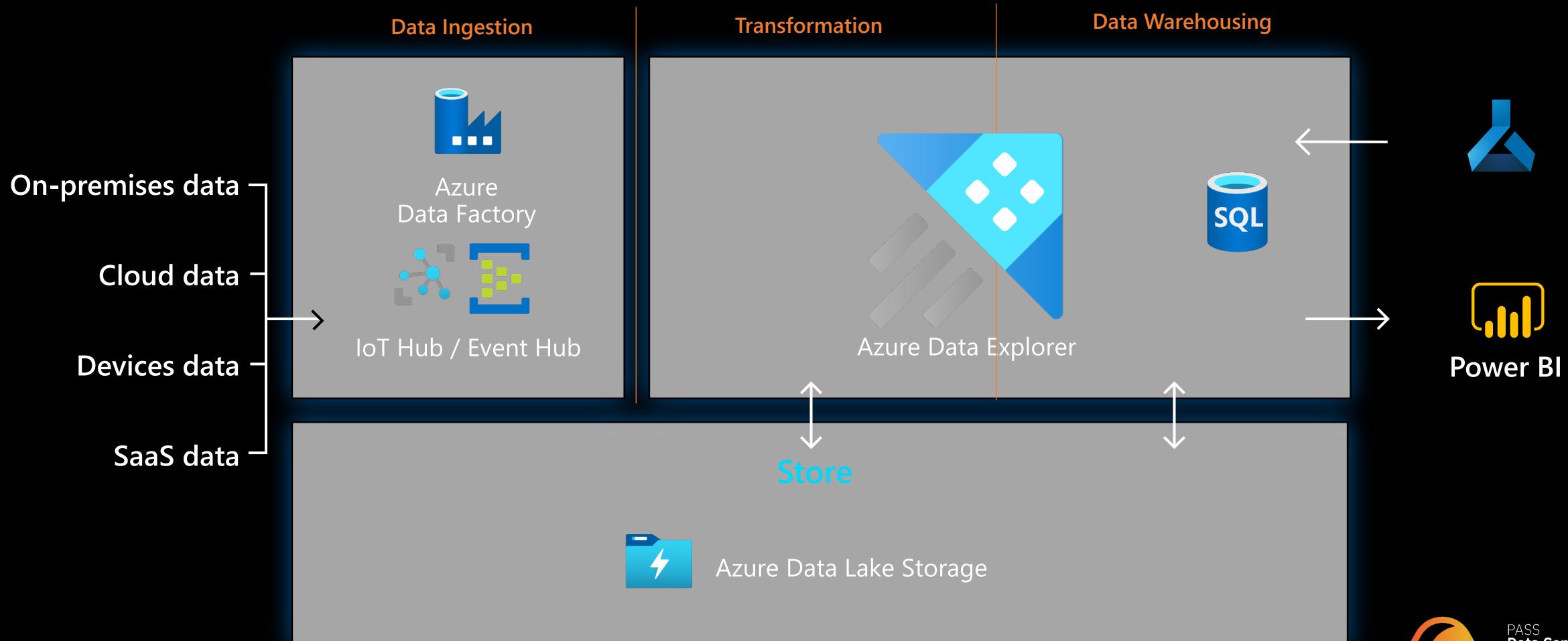
Data Explorer
– the new star in
your data platform?



Typical data platform in Azure

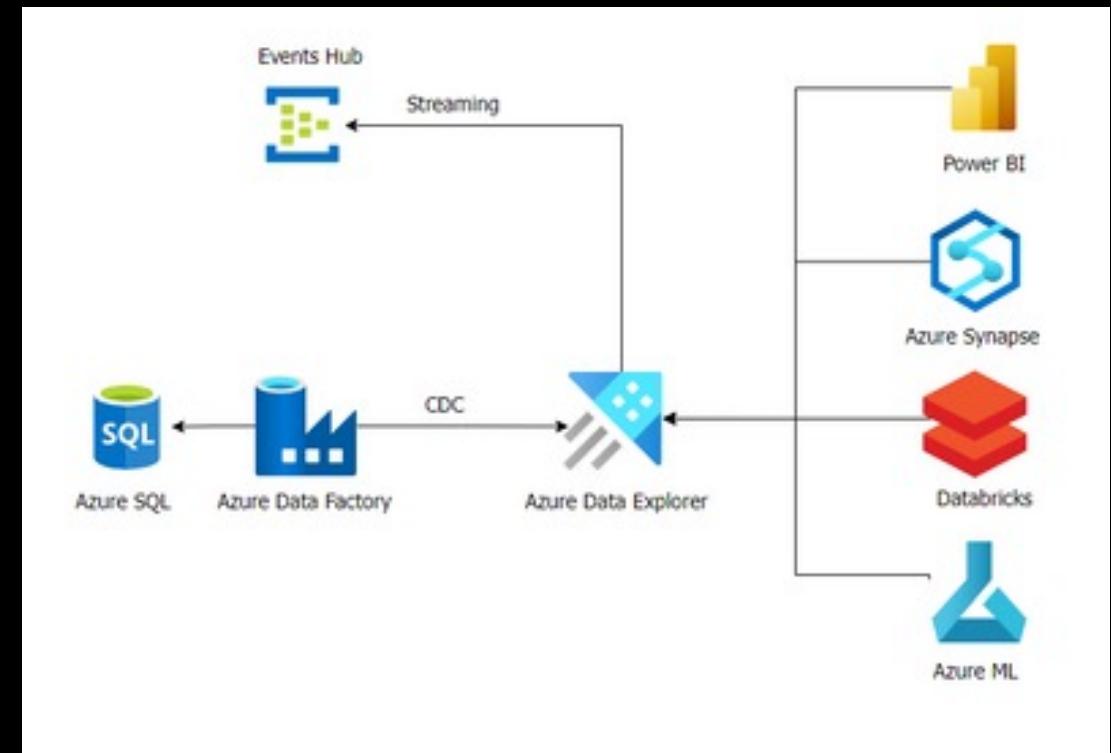


Azure Data Explorer



The benefits of Data Explorer

- Auto-ingest from event hubs
- Receive CDC & batch from Data Factory
- Continuous Export
- External Tables
- Materialized views
- Policies and retention to mimic SQL DB



Demo

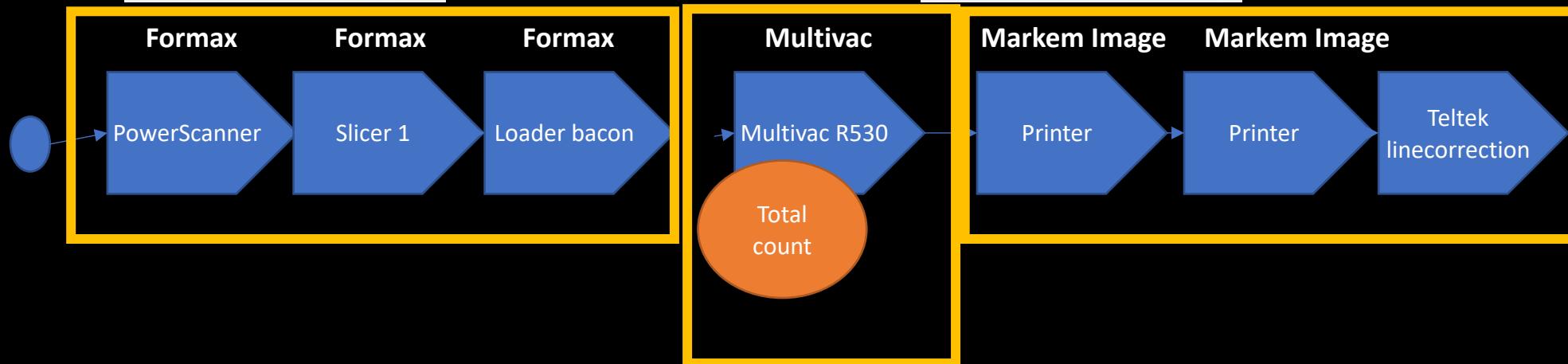


Azure Data Explorer in an IoT scenario

Overall Equipment Efficiency

OEE identifies the percentage of manufacturing time that is truly productive.

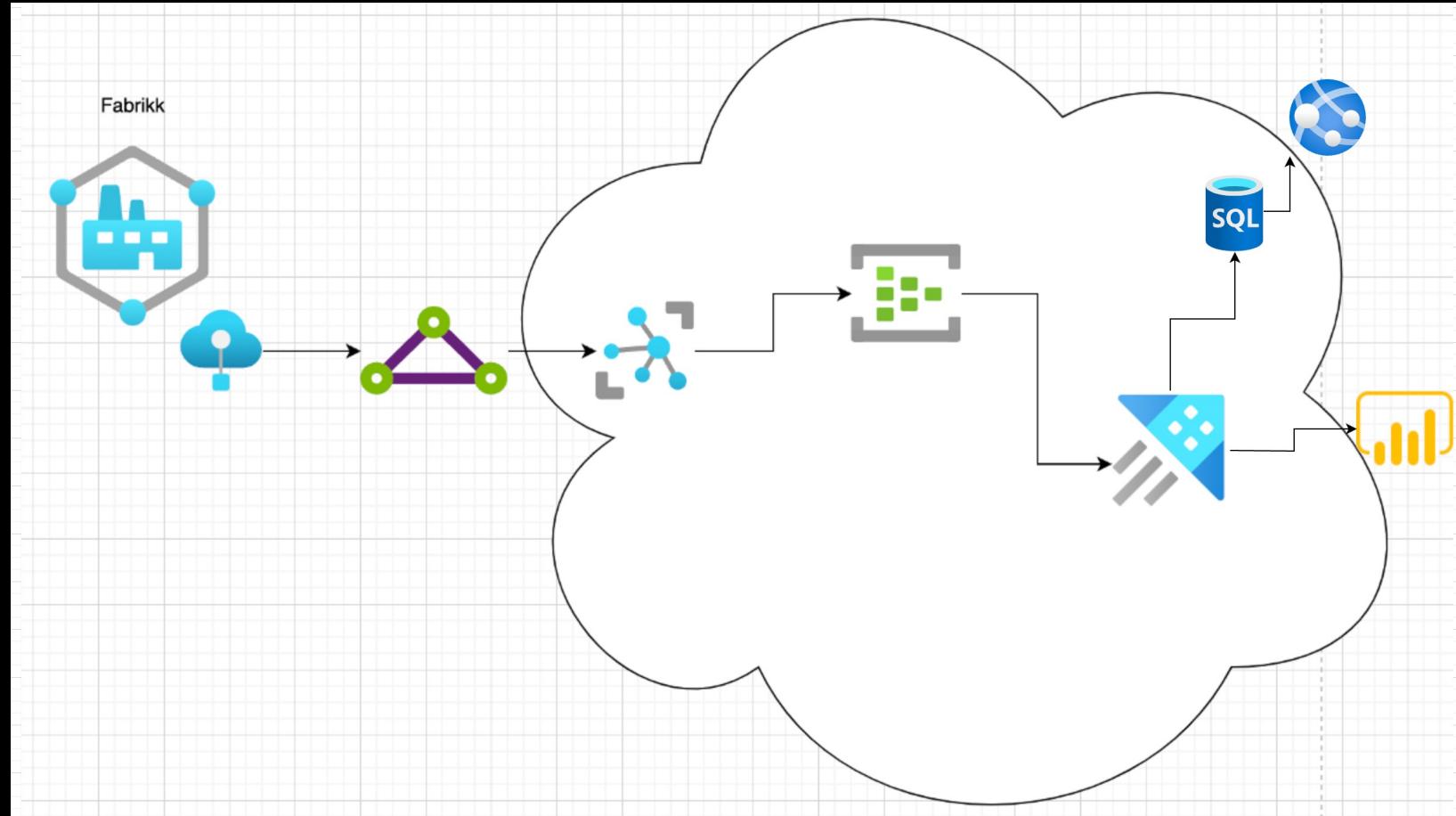
100% is theoretical potential – if no failures, no breaks, no downtime



Solution

- Counter connected
- Simple counter – 1,2,3,4...
- Stop if same number repeated
- > 1 minute defined as stop
- Query collating timeseries, identifying stops – however...
- Customer demand for export every 2 minutes to webapp
- What if stop lasts longer?

Overall Equipment Efficiency



Azure Synapse Link from Azure Cosmos DB to ADX

- (spoiler: it has nothing to do with Synapse)

Synapse Link

Create a cluster & database

Create an Azure Data Explorer Cluster

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Visual Studio Premium with MSDN

Resource group * ⓘ

(New) adxrg

[Create new](#)

CLUSTER DETAILS

Cluster name * ⓘ

adxcosmostest

Region * ⓘ

East US

COMPUTE SPECIFICATION

Select a VM size to support the workload you want to run. The VM size determines factors such as processing power, memory, and storage capacity. Azure charges an hourly price based on the VM's size and operating system. [Learn more about Compute specifications](#)

Workload * ⓘ

Dev/test

Size

Extra Small (2 cores)

Compute specifications

Dev(No SLA)_Standard_E2a_v4

[Select other](#)

Availability zones ⓘ

No zones

[Review + create](#)

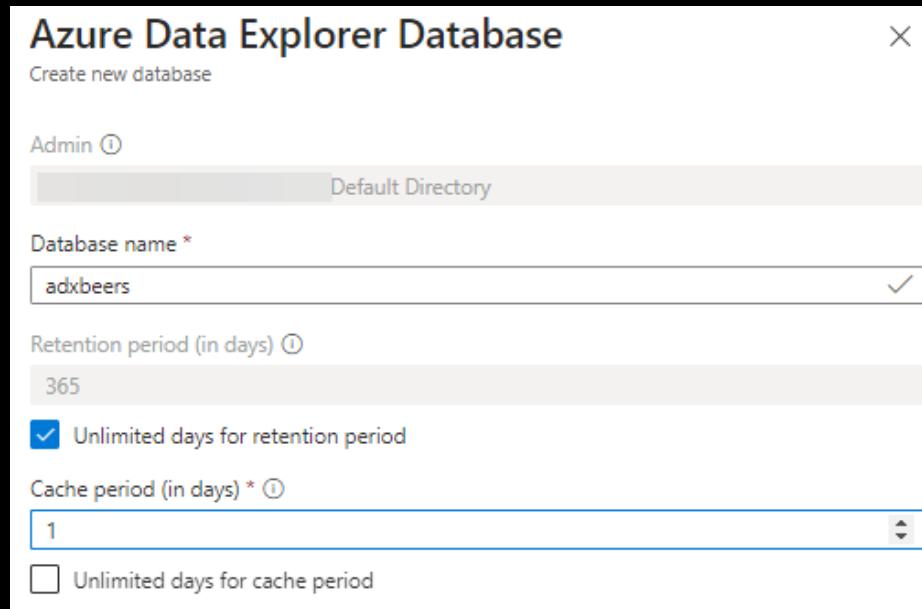
[Next : Scale >](#)

[Download a template for automation](#)

Waiting for the ADX
cluster to start...



Synapse Link



The screenshot shows the KQL editor with a table definition. The table is named 'BeerCheckins' and has the following schema:

```
.create table BeerCheckins(
    checkin_id:string
    ,created_date:string
    ,created_time:string
    ,beer_name:string
    ,beer_type:string
    ,beer_ibu:string
    ,beer_abv:string
    ,flavors:string
    ,venue_name:string
    ,venue_city:string
    ,venue_state:string
    ,venue_country:string
    ,venue_lat:string
    ,venue_lng:string
    ,brewery_name:string
    ,brewery_city:string
    ,brewery_state:string
    ,brewery_country:string
    ,purchase_venue:string
    ,score:string
    ,global_score:string
    ,global_weighted_score:string
    ,serving_type:string
    ,total_toasts:string
    ,total_comments:string
    ,comment:string
)
```

Synapse Link

Run Recall KQL tools adxcosmostest.eastus/adxbeers Open Copy Export

```
1 .create table BeerCheckins ingestion json mapping "DocumentMapping"
2 ...
3 [
4     {"column": "checkin_id", "path": "$.checkin_id"},
5     {"column": "created_at", "path": "$.created_at"},
6     {"column": "beer_name", "path": "$.beer.beer_name"},
7     {"column": "beer_type", "path": "$.beer.beer_type"},
8     {"column": "beer_ibu", "path": "$.beer.beer_ibu"},
9     {"column": "beer_abv", "path": "$.beer.beer_abv"},
10    {"column": "flavors", "path": "$.flavor_profiles"},
11    {"column": "venue_name", "path": "$.venue.venue_name"},
12    {"column": "venue_city", "path": "$.venue.venue_city"},
13    {"column": "venue_state", "path": "$.venue.venue_state"},
14    {"column": "venue_country", "path": "$.venue.venue_country"},
15    {"column": "venue_lat", "path": "$.venue.venue_lat"},
16    {"column": "venue_lng", "path": "$.venue.venue_lng"},
```

Table 1

Name	Kind	Mapping	LastUpdatedOn	Database	Table
DocumentMapping	Json	[{"column": "checkin_id", "path": "\$.checkin_id", "datatype": "", "transform": "null"}, {"column": "created_at", "path": "\$.created_at", "datatype": "", "transform": "null"}, {"column": "beer_name", "path": "\$.beer.beer_name", "datatype": "", "transform": "null"}, {"column": "beer_type", "path": "\$.beer.beer_type", "datatype": "", "transform": "null"}, {"column": "beer_ibu", "path": "\$.beer.beer_ibu", "datatype": "", "transform": "null"}, {"column": "beer_abv", "path": "\$.beer.beer_abv", "datatype": "", "transform": "null"}, {"column": "flavors", "path": "\$.flavor_profiles", "datatype": "", "transform": "null"}, {"column": "venue_name", "path": "\$.venue.venue_name", "datatype": "", "transform": "null"}, {"column": "venue_city", "path": "\$.venue.venue_city", "datatype": "", "transform": "null"}, {"column": "venue_state", "path": "\$.venue.venue_state", "datatype": "", "transform": "null"}, {"column": "venue_country", "path": "\$.venue.venue_country", "datatype": "", "transform": "null"}, {"column": "venue_lat", "path": "\$.venue.venue_lat", "datatype": "", "transform": "null"}, {"column": "venue_lng", "path": "\$.venue.venue_lng", "datatype": "", "transform": "null"}]	2023-04-14 18:46:44.0190	adxbeers	BeerCheckins

JPath: /Name

```
1 "Name": DocumentMapping,
2 "Kind": Json,
3 "Mapping": [{"column": "checkin_id", "path": "$.checkin_id", "datatype": "", "transform": "null"}, {"column": "created_at", "path": "$.created_at", "datatype": "", "transform": "null"}, {"column": "beer_name", "path": "$.beer.beer_name", "datatype": "", "transform": "null"}, {"column": "beer_type", "path": "$.beer.beer_type", "datatype": "", "transform": "null"}, {"column": "beer_ibu", "path": "$.beer.beer_ibu", "datatype": "", "transform": "null"}, {"column": "beer_abv", "path": "$.beer.beer_abv", "datatype": "", "transform": "null"}, {"column": "flavors", "path": "$.flavor_profiles", "datatype": "", "transform": "null"}, {"column": "venue_name", "path": "$.venue.venue_name", "datatype": "", "transform": "null"}, {"column": "venue_city", "path": "$.venue.venue_city", "datatype": "", "transform": "null"}, {"column": "venue_state", "path": "$.venue.venue_state", "datatype": "", "transform": "null"}, {"column": "venue_country", "path": "$.venue.venue_country", "datatype": "", "transform": "null"}, {"column": "venue_lat", "path": "$.venue.venue_lat", "datatype": "", "transform": "null"}, {"column": "venue_lng", "path": "$.venue.venue_lng", "datatype": "", "transform": "null"}, {"column": "brewery_name", "path": "$.brewery.brewery_name", "datatype": "", "transform": "null"}, {"column": "brewery_city", "path": "$.brewery.brewery_city", "datatype": "", "transform": "null"}, {"column": "brewery_state", "path": "$.brewery.brewery_state", "datatype": "", "transform": "null"}, {"column": "brewery_country", "path": "$.brewery.brewery_country", "datatype": "", "transform": "null"}, {"column": "purchase_venue", "path": "$.purchase_venue", "datatype": "", "transform": "null"}, {"column": "score", "path": "$.rating_score", "datatype": "", "transform": "null"}, {"column": "global_score", "path": "$.global_rating_score", "datatype": "", "transform": "null"}, {"column": "global_weighted_score", "path": "$.global_weighted_rating_score", "datatype": "", "transform": "null"}, {"column": "serving_type", "path": "$.serving_type", "datatype": "", "transform": "null"}, {"column": "total_toasts", "path": "$.total_toasts", "datatype": "", "transform": "null"}, {"column": "total_comments", "path": "$.total_comments", "datatype": "", "transform": "null"}, {"column": "comment", "path": "$.comment", "datatype": "", "transform": "null"}], "LastUpdatedOn": 2023-04-14T18:46:44.0191498Z, "Database": adxbeers, "Table": BeerCheckins}
```

Synapse Link

Get started with Azure Data Explorer

Use the Azure Data Explorer web app to manage your data easily. [Learn more](#)

Database creation
Create a database [Create](#)

Data ingestion
Ingest new data or go to the Azure Data Explorer web app to manage your data. [Ingest](#) [Create data connection](#)

- Cosmos DB (preview)
- Event Grid (Blob storage)
- Event Hub
- IoT Hub

Query
Write, run, and share Kusto Query Language commands and queries. [Explore](#)

Dashboards
Use Azure Data Explorer to create and share dashboards and visualize data. [Visualize](#)

Cosmos DB (preview)

Create data connection

Database Name *

adxbeers [\(View resource\)](#)

Data connection name * ⓘ

cosmosbeers

Subscription

Visual Studio Premium with MSDN

Cosmos DB account * ⓘ

mssqltips-cosmoslink [\(View resource\)](#)

SQL database * ⓘ Beers [\(View resource\)](#)

SQL container * ⓘ Checkins [\(View resource\)](#)

Table name *

BeerCheckins

Mapping name ⓘ DocumentMapping [X](#)

Advanced settings

Event retrieval start date ⓘ

Sat Jan 01 2022 [Calendar](#) 12:00 [X](#)

If you change the date, this might cause ingestion latency while older records are ingested.

Managed identity type

Select which identity type to use for this data connection. [Learn more about identity types](#)



System-assigned

Use an identity created in Azure AD that's tied to the cluster's lifecycle.



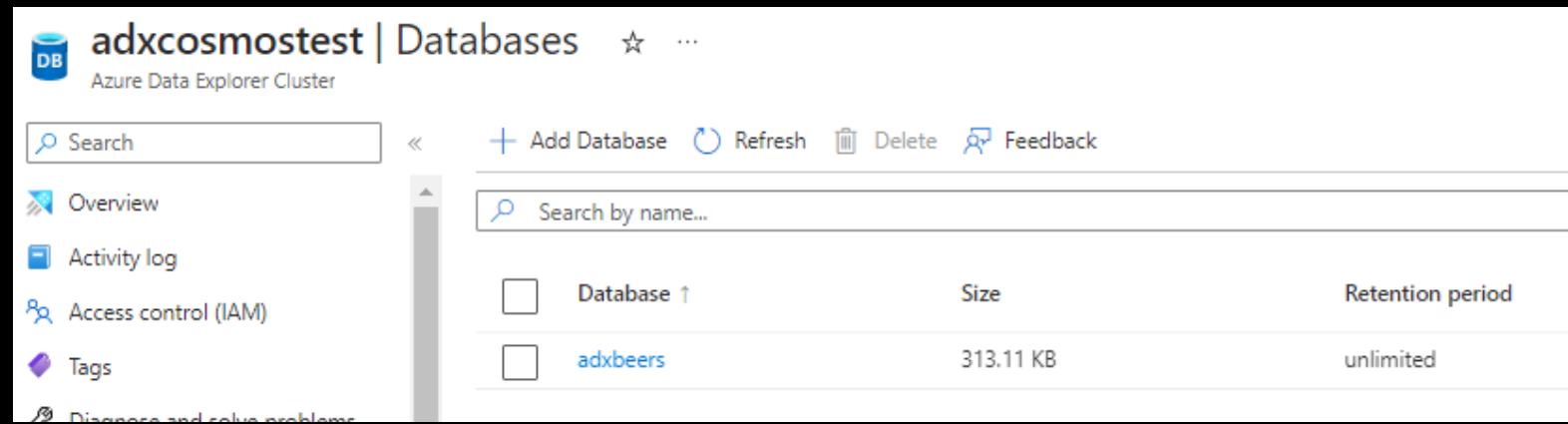
User-assigned

Use an identity that's a standalone Azure resource with a separate lifecycle.

ⓘ System-assigned identity is not authorized to access mssqltips-cosmoslink/Beers/Checkins as "Azure Cosmos DB Reader". Azure "Azure Cosmos DB Reader" role will be added

Synapse Link

data will be imported from the specified start date
using Azure Cosmos DB change feed



The screenshot shows the 'Databases' section of the Azure Data Explorer Cluster interface. The cluster name is 'adxcosmostest'. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, and Diagnosis and solve problems. The main area displays a table with two rows:

<input type="checkbox"/>	Database ↑	Size	Retention period
<input type="checkbox"/>	adxbeers	313.11 KB	unlimited

when it's imported, you can query it with KQL

profit!

Run Recall KQL tools adxcohostest.eastus/adxbeers

1 BeerCheckins

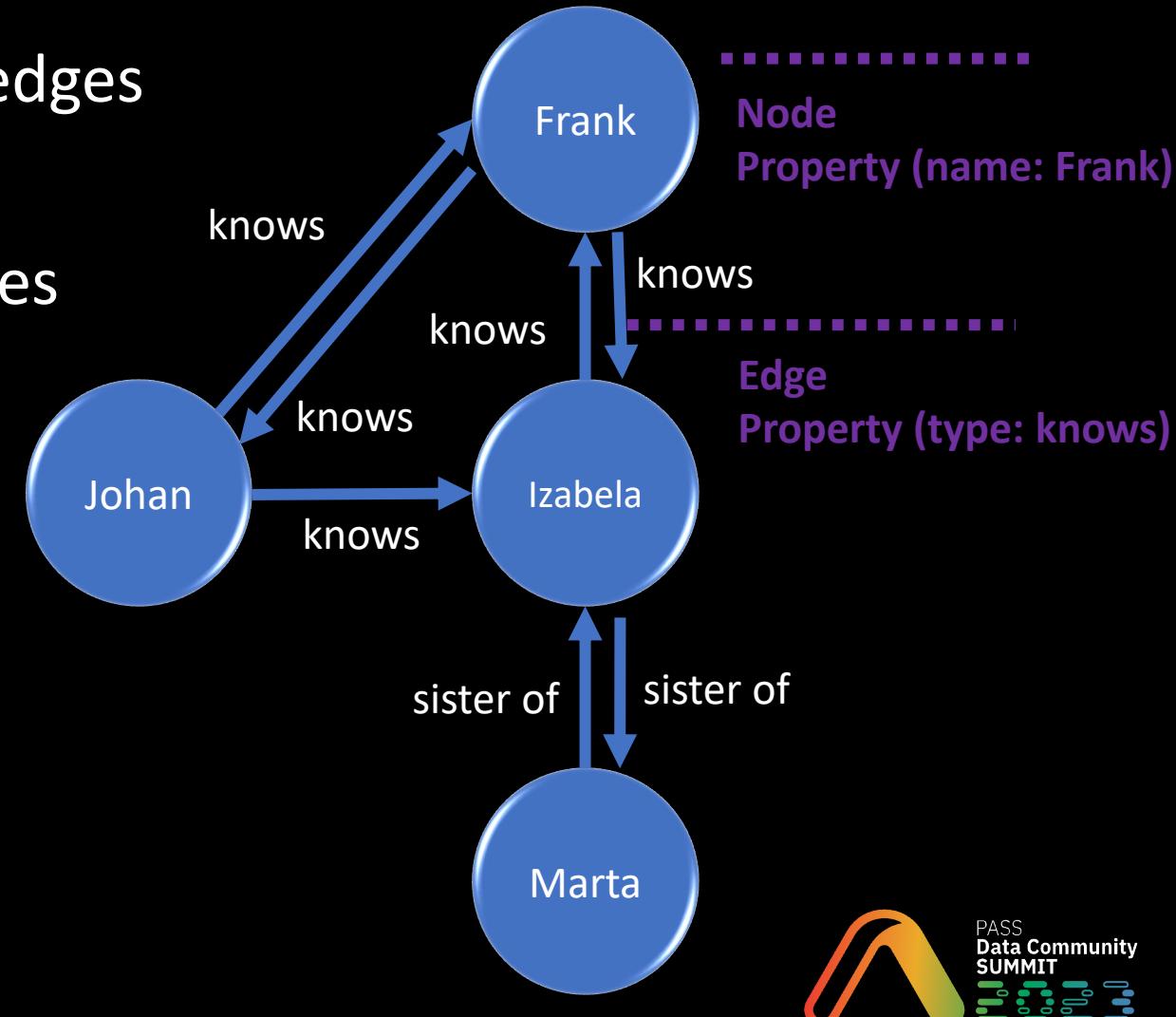
BeerCheckins Stats Search

checkin_id	created_at	beer_name	beer_type	beer_ibu	beer_abv	flavors
> 1218240683	2022-11-05T20:33:14	Zinne des Flandres	Pale Ale - B...	0	5.8	[{"flavor": "hoppy"}, {"flavor": "bitter"}]
> 1216549442	2022-10-30T18:42:35	HertenHaas	Belgian Tripel	35	7.5	[{"flavor": "belgiany"}, {"flavor": "sweet"}, {"flavor": "hoppy"}]
> 1214306625	2022-10-23T15:39:20	Chimay 150 (Green)	Belgian Stro...	31	10	[{"flavor": "floral"}, {"flavor": "strong"}, {"flavor": "spiced"}]
> 1210260383	2022-10-10T20:36:50	3 Fonteinen Oude Geuze	Lambic - Gu...	0	6	[{"flavor": "dry"}, {"flavor": "sour"}, {"flavor": "tart"}]
> 1210205708	2022-10-10T13:04:56	Gouden Carolus Tripel	Belgian Tripel	30	9	[{"flavor": "sweet"}, {"flavor": "strong"}, {"flavor": "smooth"}]
> 1209867243	2022-10-09T14:48:24	Vicaris Quinto	Belgian Blo...	25	5	[{"flavor": "soft"}, {"flavor": "light"}, {"flavor": "dry"}]
> 1206767994	2022-09-30T22:31:11	Pius X Tripel	Belgian Tripel	0	10.7	[{"flavor": "herbal"}, {"flavor": "strong"}]
> 1205300638	2022-09-25T01:38:57	Orval	Pale Ale - B...	36	6.2	[{"flavor": "fruity"}, {"flavor": "dry"}, {"flavor": "sour"}]
> 1203891699	2022-09-21T18:07:41	Julius 54BC	Belgian Stro...	25	8.5	[{"flavor": "sweet"}, {"flavor": "strong"}]
> 1198657148	2022-09-04T17:17:14	Vedett Extra Pilsner (Extra Blond)	Pilsner - Ot...	26	5.2	[{"flavor": "clean"}, {"flavor": "light"}, {"flavor": "smooth"}]
> 1193418465	2022-08-20T15:59:21	Poike	Blonde Ale	27	6.2	[{"flavor": "malty"}, {"flavor": "ginger"}, {"flavor": "spiced"}]
> 1190374989	2022-08-12T12:07:52	Mind Haze	IPA - New E...	40	6.2	
> 1186400983	2022-07-31T20:22:46	Kapittel Aged	Brown Ale - ...	0	7.7	[{"flavor": "fruity"}, {"flavor": "sweet"}, {"flavor": "sour"}, {"flavor": "strawb..."}]
> 1185164627	2022-07-29T16:56:32	Pink Flow	Pale Ale - O...	0	4.5	[{"flavor": "floral"}, {"flavor": "hoppy"}]

Graph Databases with Data Explorer

What is a graph database?

- Graph compromises of nodes and edges
- edges connect nodes
- edges and nodes can have properties
- data is differently stored than in a relational database:
 - Each node has a pointer to its direct neighbour
 - no need to index or join anything
 - easier and faster to traverse graph



Why graph database?

Useful for:

- complex and dynamic data
- network relationships
- hierarchical relationships
- many-to-many relationships

Examples:

- social networks
- recommendation systems
- connected assets
- knowledge graphs

queries can use graph structure and meaning => complex and powerful queries:

- finding paths
- finding patterns
- finding shortest distances
- finding communities
- finding centrality measures

Why graph semantics in KQL?

KQL does not support recursive joins:

- explicitly define traversals → use make-graph operator to define hops of variable length.
- Useful when relationship distance or depth is not fixed
- find all resources that are connected in a graph or find all places that you can reach from a source

Time-aware graphs

- unique feature of graph semantics in KQL → allow user to model graph data as series of graph manipulation events over time
- examine how graph evolves over time
 - changes of network or properties over time
 - how anomalies happen
 - use time series queries to discover
 - trends
 - patterns
 - outliers
 - analyze changes in
 - network density
 - centrality
 - modularity

Examples for use of Graph Queries

- Time-based queries:
 - analyze evolution of graph over time
(changes of network or property values)
- Geospatial Analysis:
 - spatial distribution or proximity of nodes and edges
(how distance or location effects relationships)
- Machine learning queries:
 - apply various algorithms to graph data
(clustering, classification, anomaly detection)

make-graph

- simple intuitive syntax
- works well with existing KQL features
- mix graph queries with other queries
 - time-based queries
 - machine learning queries
 - location-based queries
 - => more advanced and powerful data analytics
- KQL semantics → speed and scale of KQL mixed with flexibility and expressiveness of graph queries

Supported Graph Operators

Operator	Description
<u>make-graph</u>	Builds a graph from tabular data.
<u>graph-match</u>	Searches for patterns in a graph.
<u>graph-merge</u>	Merges two graphs into a single new graph.
<u>graph-to-table</u>	Builds nodes or edges tables from a graph.

- Graph is transient object
- Build when query is run and deceased when query is finished
- Persist a graph → transform back to tabular form and then stored as edges or nodes table

Limits and recommendations

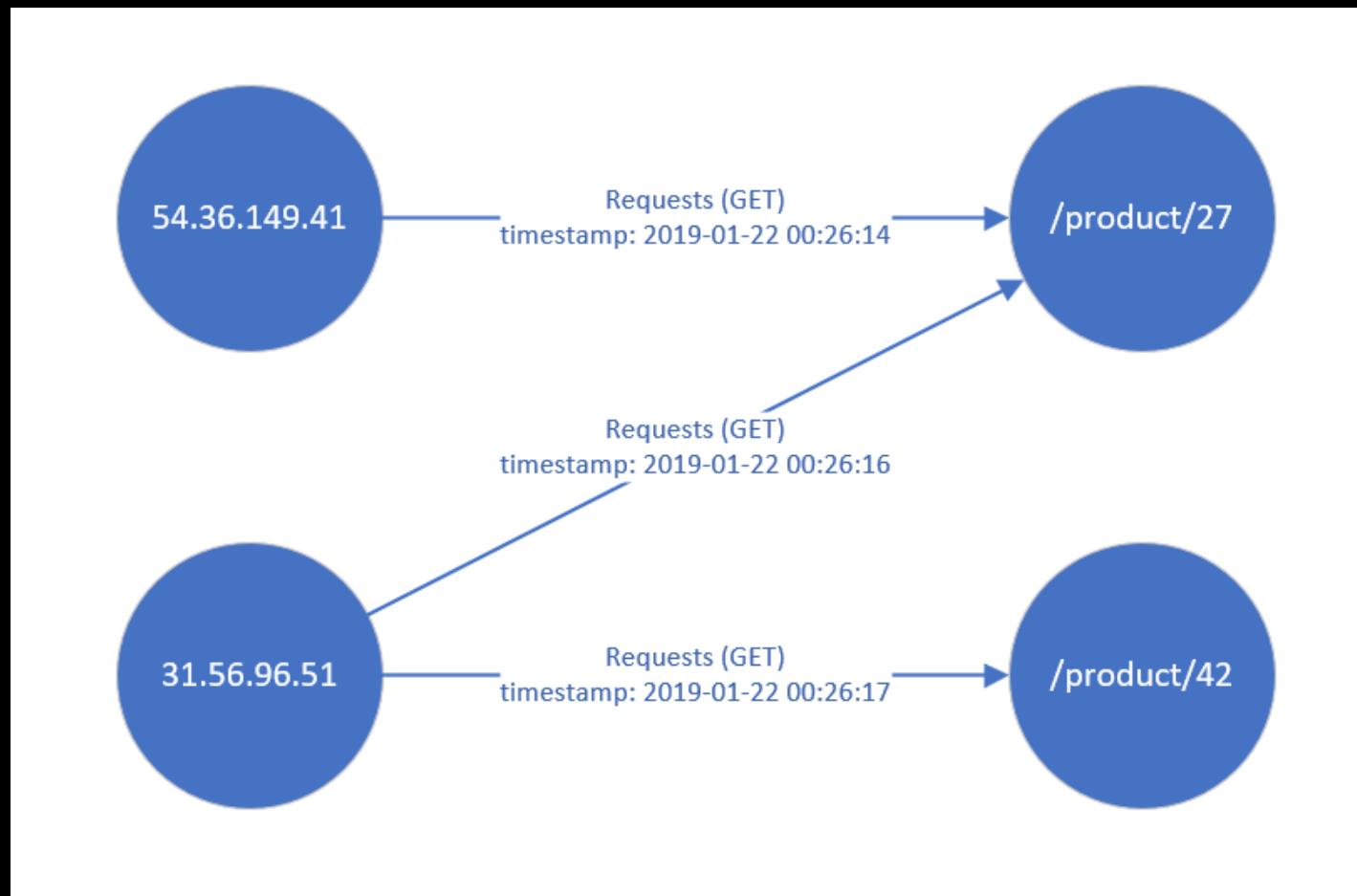
- graph is build in memory on the fly → performance cost for building the graph
- limit the size of the graph by only including necessary data
- building graphs with 10 Mio objects (edges and nodes) at max
- Query limitations:

[Query limits - Azure Data Explorer | Microsoft Learn](#)

Demo

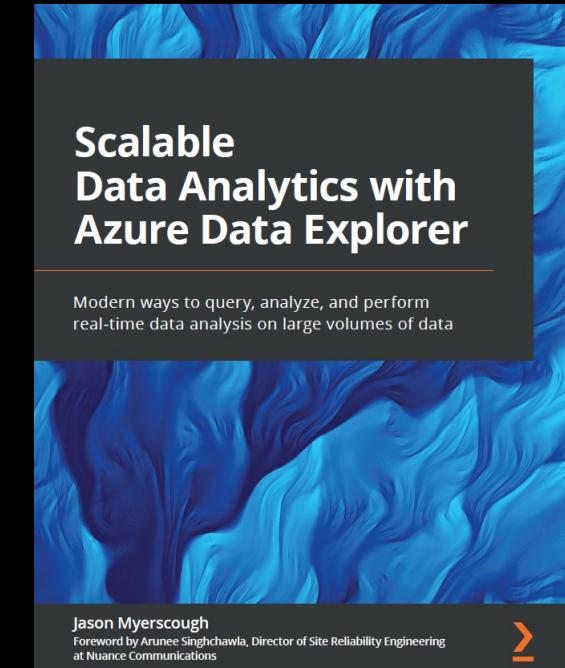


Scenario 2



Resources

- [ADX in a day – GitHub](#)
- [Kusto Detective Agency](#)
- [The Most Powerful Azure Service You've Never Heard Of – SQLBits](#)
- [Get started with Real-Time Analytics in Microsoft Fabric – Microsoft Learn](#)
- [Start for free with ADX](#)
- [Building a Data Lakehouse using Azure Data Explorer](#)





Frank Geisler

CEO, GDS Business Intelligence GmbH



/frank-geisler-6877541



@FrankGeisler



Frank_geisler@geislernet



GitHub

Community Person

Co-organizer – Data Moshpit
On the board of PASS Deutschland e.V.
Chapter Lead of PASS RG Münsterland

When not geeking out over new tech

Listening to Metal
Doing some decent BBQ
Going on "Adventures" with my 6y old daughter



Johan Ludvig Brattås

Director, Deloitte



/johanludvig



@intoleranse



jbrattas@deloitte.com



GitHub

Chronic volunteer

Co-organizer – DataSaturday Oslo
President – MDPUG Oslo
Frequent volunteer in general

When not geeking out over new tech

Teaching coeliacs how to bake gluten free
Baking
Hiking
Gardening

Session evaluation

Your feedback is important to us



Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation



