Process streaming data in Power BI with Azure Data Explorer

Nico Jacobs
@SQLWaldorf

Platinum partners

creates.

In Summa

Goud partners

Kimura

plainwater
de kracht van heldere data

KASPAROV FINANCE & BI

Zilver partners

rockfeather

Dynamic People

GET RESPONSIVE

Brons partners

Hso

macaw

iqbs

VICTA
BUSINESS INTELLIGENCE

Quanto
collective analytics

ilionx

valcon

VALID
STAY AHEAD

Community partners

broadwick
Data & development recruiters

THE DATA COOKS

Tabular Editor

Datamanzi

Power BI Connector
by DAVISTA

MINOVA

AZURROFINANCE

DATA KINGDOM

volda;
INFORMATIESPECIALISTEN

DashData.

VisionBI
Smart Data Experts

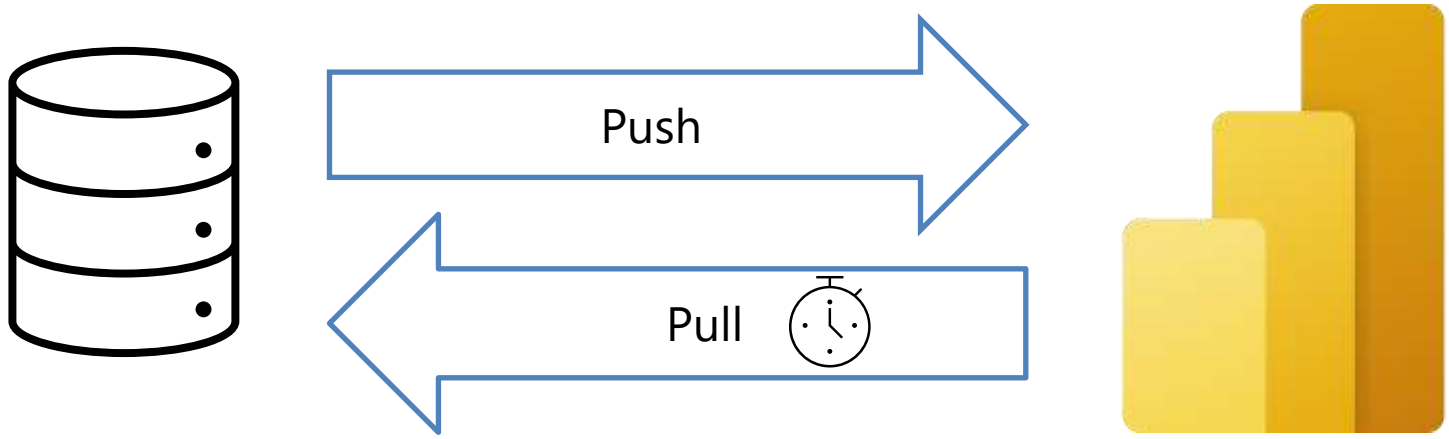
easydash

Why Streaming Data?

- Traditional Business Intelligence first collects data and analyzes it afterwards
  - Typically 1 day latency
- But we live in a fast paced world
  - Social media
  - Internet of Things
  - Just-in-time production
- We want to monitor and analyze streams of data in near real time
  - Typically a few seconds up to a few minutes latency

Power BI data access

- Power BI has different ways to interact with source data
  - Pull: Power BI queries the data when needed
  - Push: The data is sent to Power BI whenever the source wants this
- Pull is the most common technique
- Pull often doesn't know when the source data has changed
  - This can increase latency between data changed in the source and data changed in the report



Pull options for tables in semantic model

Import

- All data and meta-data is stored in Power BI
- Fast query performance
- Long refresh time needed -> increases latency
- Models can become too large

DirectQuery

- Meta-data is stored in Power BI
- Data is queried from the source for every DAX query → fast source needed
- No data refresh needed (unless automatic aggregations are used)
- No concerns on model size

Dual

- Combines Import and DirectQuery on a single table
- Dependent on query Power BI can decide to use on or the other

Live Connection

- Only possible to Power BI/Analysis Services models
- Both meta-data and data are stored in the source

Push options for streaming data in Power BI

- Power BI has a few built-in techniques for dealing with streaming data:

Push semantic model

Streaming semantic model

PubNub streaming semantic model

Push semantic model

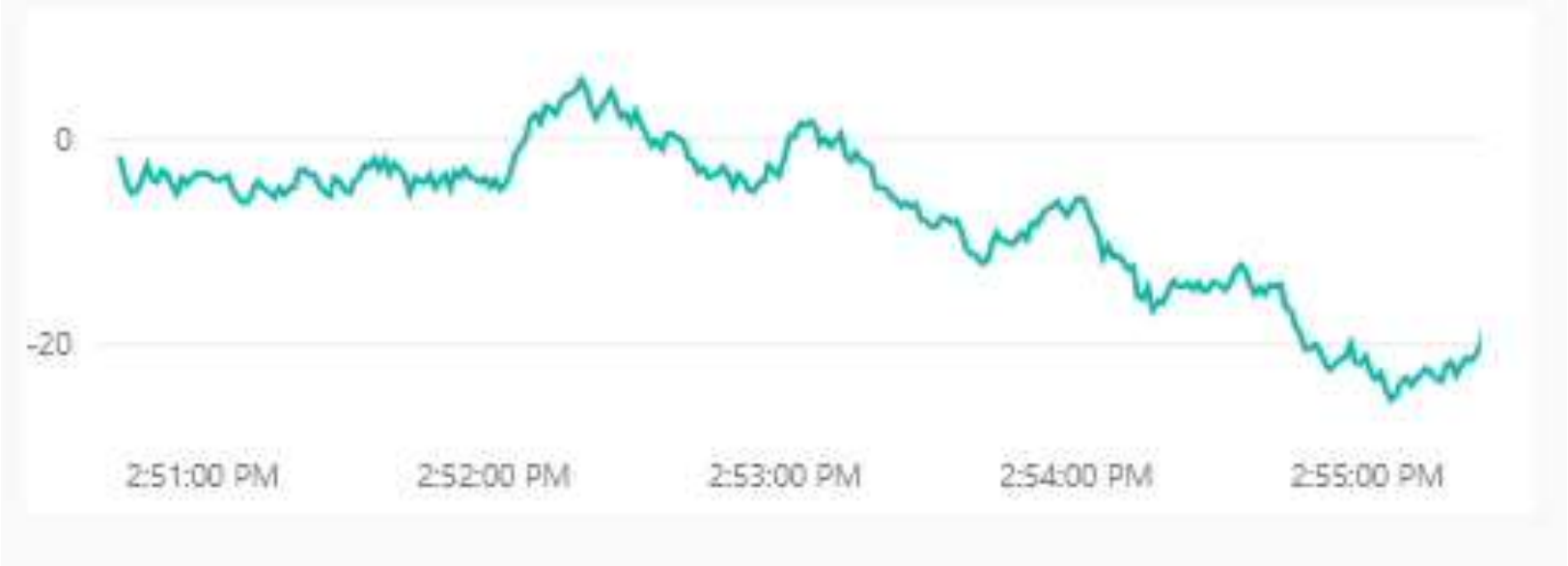
- Creates a special Power BI semantic model
- This model doesn't pull data upon model refresh
- Via a special web service software has to push data into the model
  - Or via Azure Stream Analytics
- Regular reports and dashboards can be created on top of it
- Dashboard visuals and Q&A visuals refresh in real time

Limitations Push semantic model

- Max 75 tables
- Max 200.000 rows per FIFO table
  - New rows overwrite the older rows
- Max 5 million rows per regular table
  - New rows are ignored once limit is reached
- Max 1 million new rows per hour
- More limitations apply, check documentation

Streaming semantic model

- Creates special Power BI semantic model as well (same REST API)
- But data pushed into it is only cached for 1 hour
- No regular reports, Q&A, filters, alerts etc possible
- Only special Dashboard tiles are supported
  - But the visuals have smooth updates
- Since no Power Query or DAX can be involved, this provides lowest latency



Streaming data with PubNub

- PubNub is an external service
- You require a subscription on PubNub to use it
- Data is not stored in Power BI
- Very similar to Streaming semantic model

Advantages of these approaches

- The push and streaming models do support dashboard visuals with very low latency
- For Push models also regular Power BI reports can be made, but these require the auto-page refresh for auto-updates
  - Push model is treated as DirectQuery for auto-refresh
- Easy to setup from within Azure portal when using Azure Stream Analytics

Disadvantages of these approaches

- Difficult to handle very large volumes of data ingestion and storage
- Difficult to handle varying sources
- The need for Azure Stream Analytics to transform and push the data
  - Unless you program it directly against REST service or PubNub

Wrap up

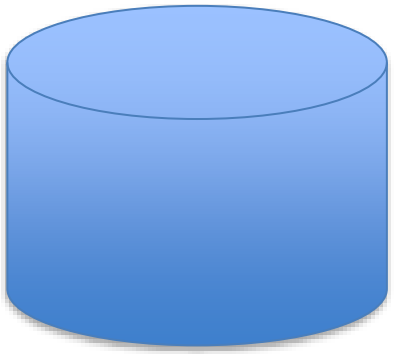
- When just showing detailed numbers, push datatsets can be used to build reports or dashboards showing streaming data
- But this approach has issues with
  - Large data sets
  - Large volumes of incoming data
  - Complex calculations
- A solution might be DirectQuery
  - This does not limit the volume or influx of data at all
  - It has minor limitations on what can be done in PowerQuery
    - Depends on source used and amount of query folding supported

Azure Data Explorer

Is it reporting?

Is it analytics?

No, it's a database!



What is Azure Data Explorer?

u2u

Cloud-based, scale-out, append-only database for data exploration

Cloud-based

- It runs in the Azure cloud
- It is part of other Azure offerings
  - Azure Monitor, log analytics, application insight, …
- Or it can be us
  - Azure Data Ex
- Recently a Doc

```
1  resources
2  | summarize count() by location
3  | order by count_ desc
```

abric Synapse Data Explorer

as been released

Scale-out

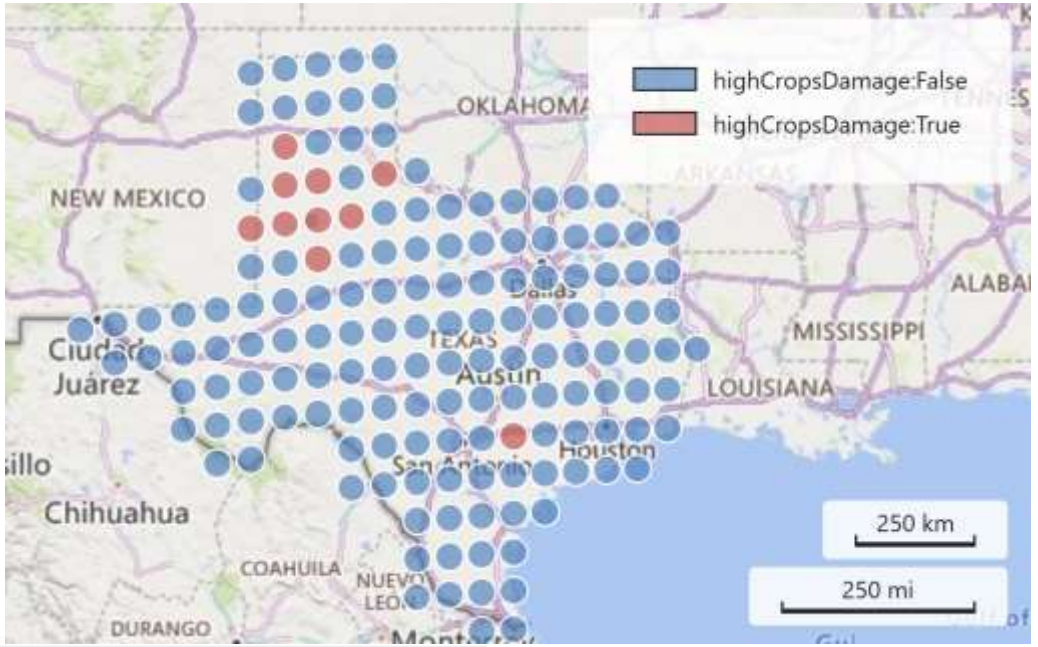
- ADX spreads the work over multiple machines
- We can control how powerful these machines are
  - Scale-up
- We can control how many machines we want
  - Scale-out
- The service can be paused to save cost
  - Restarting takes 5-10 minutes

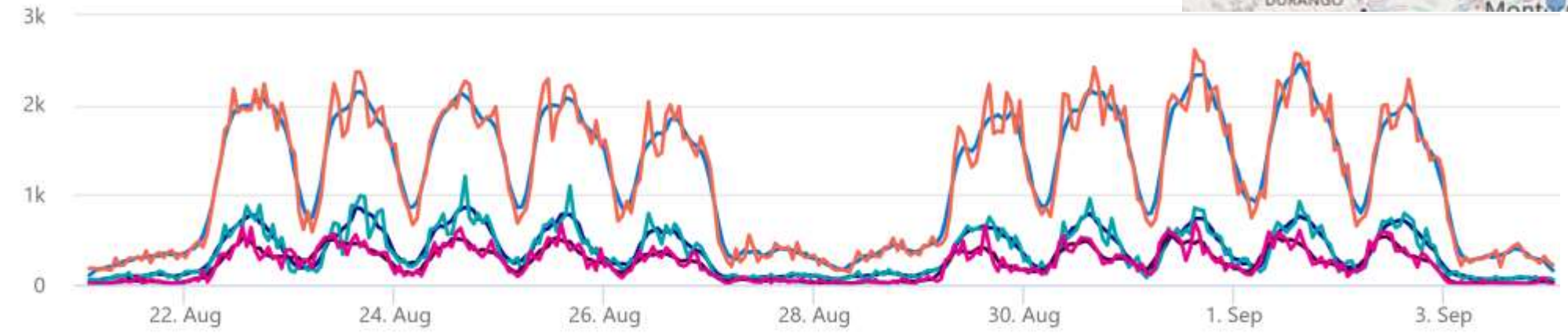
Append-only

- ADX is designed for processing streams of data
  - Telemetry data, logfiles, stream of business events, …
- The data can be inserted in batches
  - Usually, every couple of minutes a batch of data get processed
  - Streaming data is supported as well, but lower throughput (< 4Gb/h)
- Data is kept for a pre-defined time window
  - Days, weeks, months, …
  - Default set at the database level, can be controlled at the table level
- By default, no other row delete options
  - Unless purge is enabled
- No update option
  - But you can write query results to new table and drop table

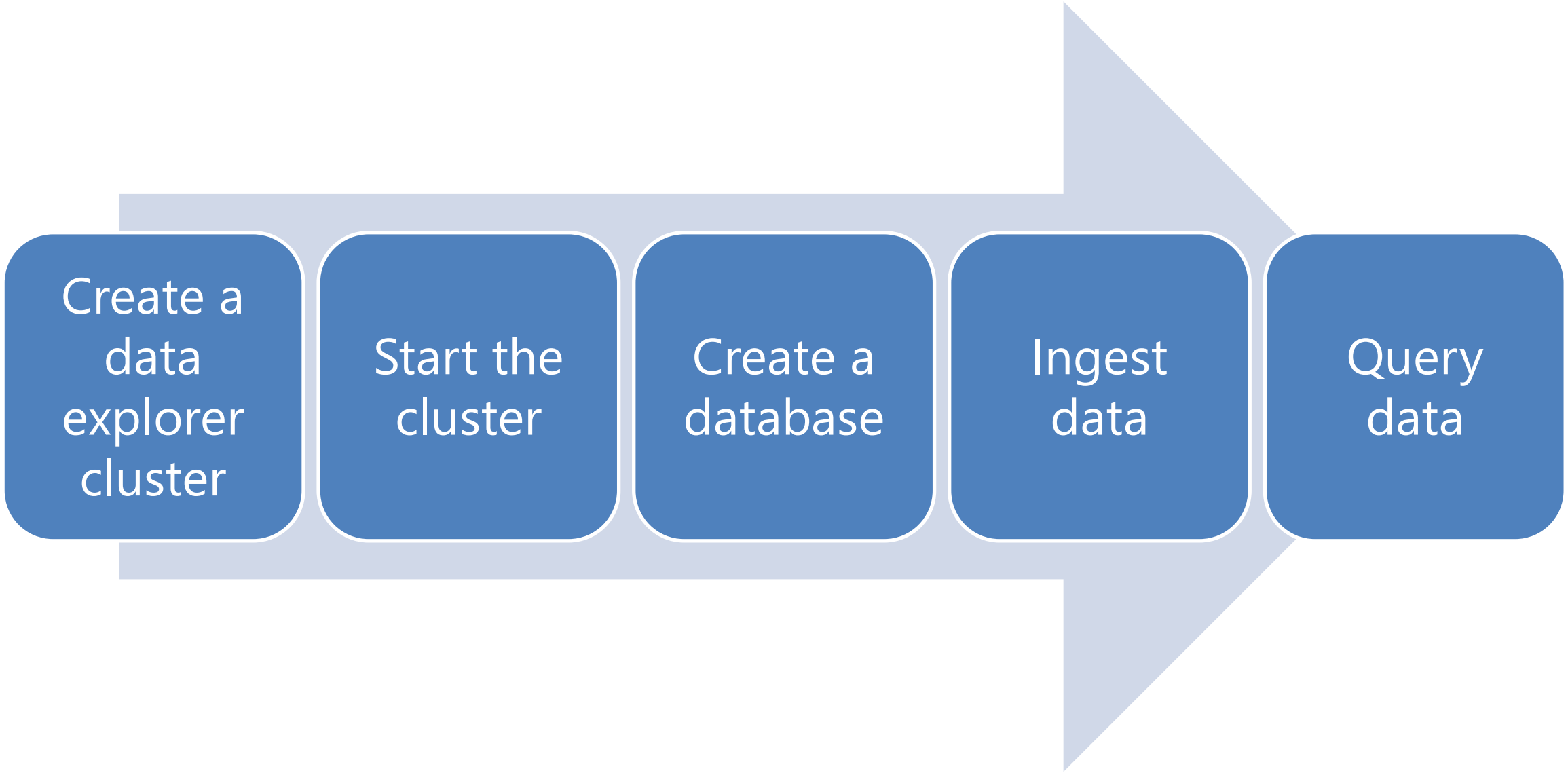
For data exploration

- Contains many features to make data analysis easier
  - Time series functions
  - Geolocation functions
  - Python and R integration
  - Data visualization





Start using Azure Data Explorer



Create the cluster in Azure

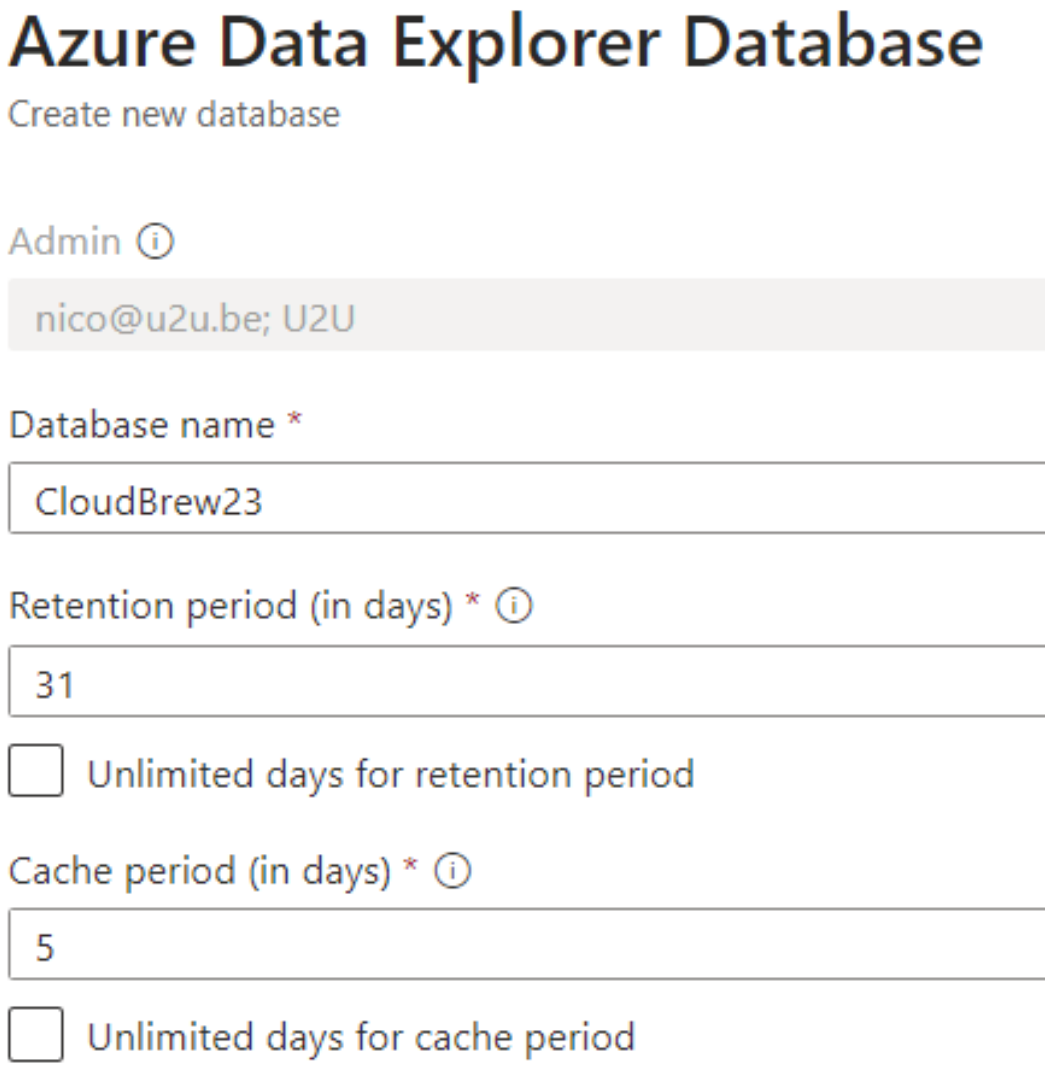
- Azure Data Explorer in Azure
- Data Explorer Database in Azure Synapse Analytics (preview)
- KQL Database in Microsoft Fabric Synapse Real-Time Analytics
- Azure Data Explorer docker container (on-prem, dev only)
- Free cluster (8 Gb ram, 100 Gb data) at https://aka.ms/kustofree

Cluster properties

- VM type (compute or storage optimized) and size
- Number of VMs and auto-scale options
- Streaming option
  - Streaming data is first loaded in row store before conversion to column store
  - Reduces throughput but data available in seconds instead of minutes
- Purge option
  - Transactional inconsistent delete option (GDPR)

Create a database

- Data is loaded in extends
- Once an extend is finished it becomes read-only
- Extends can be merged, copied to different machines, …
- An extend expires (and gets removed from blob storage) after the Retention period
- An extend can be kept on SSD (or even in memory) during the Cache period



Inspecting and changing database settings

- The Kusto query language has statements to inspect and modify meta-data
- These statements all start with a .
  - To distinguish them from regular queries
- With .show database and .show databases meta-data at the database level can be retrieved
- Using .alter database allows to change the meta-data if needed

```
1  .show databases
2
```

Table 1 + Add visual

DatabaseName	PersistentStorage	Version	IsCurrent
babs	https://xm9virtualengines.blob.core.windows.net/5caengine...	v42.0	false
kustodemo	https://xm9virtualengines.blob.core.windows.net/67rkustod...	v80.0	true

Retention Policy

- The database retention policy controls how long data is kept in the database

```
.alter database kustodemo policy retention
```
```
{
  "SoftDeletePeriod": "365:00:00",
  "Recoverability": "Disabled"
}
```

```
.show database kustodemo policy retention
```

PolicyName	EntityName	Policy
RetentionPolicy	[kustodemo]	{ "SoftDeletePeriod": "365.00:00:00", "Recoverability": "Disabled" }

Cache policy

- By setting the caching policy by code, more detailed caching can be controlled
  - E.g. cache data for the last 20 days, but also the Christmas sales

```
.alter database kustodemo policy caching
        hot = 20d,
        hot_window = datetime(2023-12-15) .. datetime(2024-01-01)
```

- Most policies can be set at the cluster level and overwritten at the database and/or table level

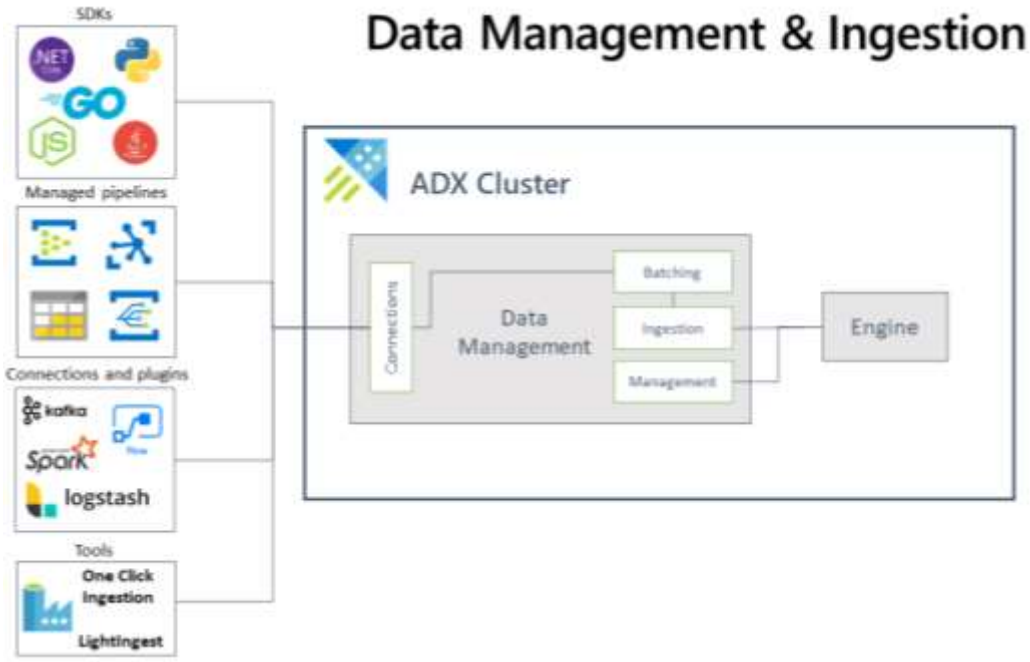
Create a table

- ADX is table-oriented
  - Table has a set of strongly typed columns
  - In contrast to ElasticSearch and other search engines
  - But dynamic columns can hold JSON or XML
- Tables have no primary nor foreign keys
  - But you can join tables while querying
- Both a GUI as well as Kusto code can be used to create tables

```
.create table ['station']  (['station_id']:long, ['name']:string, ['lat']:real,
['long']:real, ['dockcount']:long, ['landmark']:string, ['installation']:datetime)
```

Ingesting data

- Data is processed by the data management component
  - Schema checks
  - Data type conversions
  - Sharding
  - Compression
  - Indexing
- After persisting it in storage the data load is committed



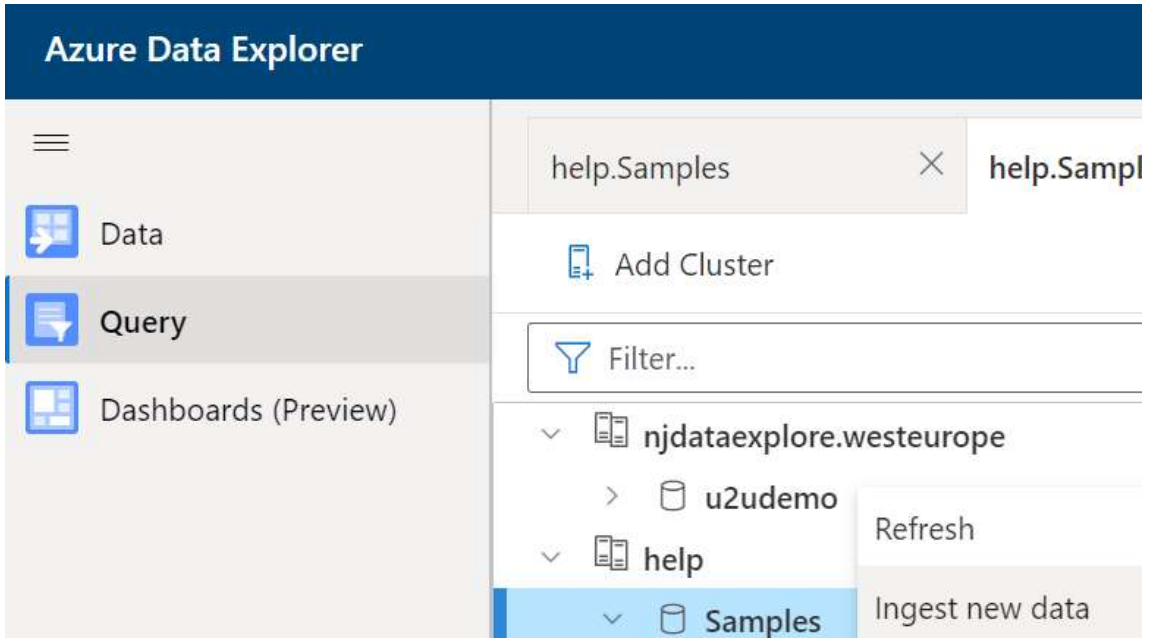
(https://docs.microsoft.com/en-us/azure/data-explorer/ingest-data-overview)

Data ingestion options

- Web portal
- Control statements
- LightIngest console application
- APIs
- Azure Data Factory, Synapse pipelines, Fabric pipelines and EventStreams

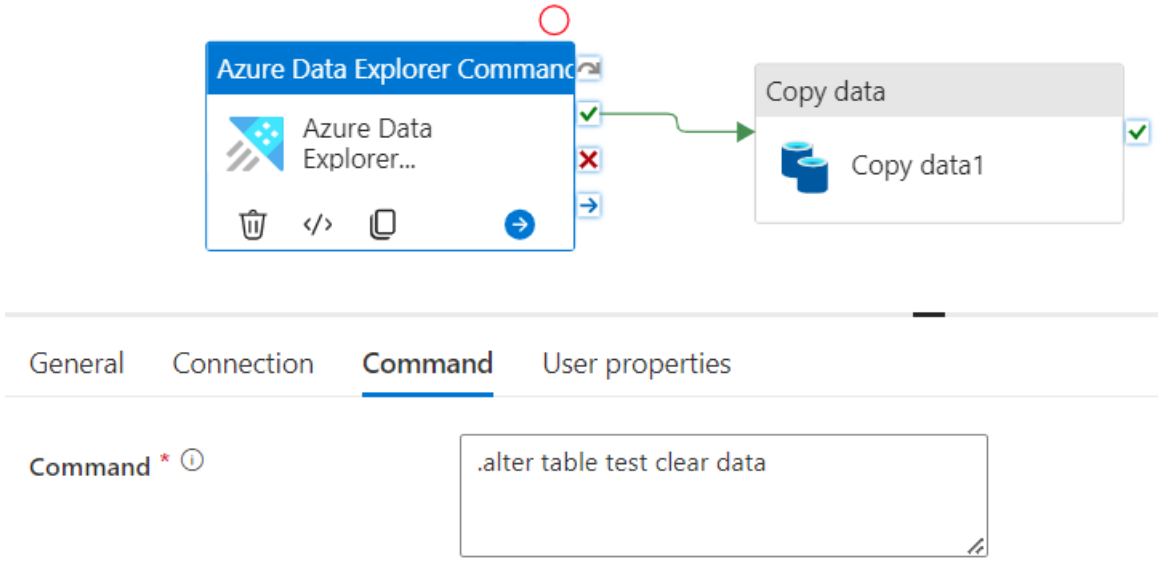
Data ingestion options

- One click ingestion
  - Start from Azure Data Explorer web portal → Ingest new data
  - For one time data loading from limited set of sources
    - Blob, Data Lake (1 & 2) or local file
  - For incremental loading from Event Hub
- LightIngest
  - Console application
  - Can insert data from local file or blob storage



Pipelines

- Azure Data Factory, Synapse Analytics and Fabric all support pipelines
- The Copy Data and Data Flow activity in there supports ADX tables as sources and destinations
- There is also an Azure Data Explorer Command to issue statements to truncate tables, drop or recreate tables, …



Ingest data from Kusto

- Using Kusto code, data can be ingested as well
- However, first a mapping from source files into destination columns needs to be defined:

```
.create table ['station'] ingestion csv mapping 'station_mapping'
'[{"column":"station_id", "Properties":{"Ordinal":"0"}},{"column":"name",
"Properties":{"Ordinal":"1"}},…,{"column":"installation",
"Properties":{"Ordinal":"6"}}]'
```

- Only then, the ingestion can happen (e.g., from an Azure Blob Storage Shared Access Signature (SAS)

```
.ingest async into table ['station']
('https://yoursa.blob.core.windows.net/container/file.csv?sv=2023-01-03&…') with
(format='csv',ignoreSizeLimit=true,ingestionMappingReference='station_mapping’,
ingestionMappingType='csv',tags="['90388f32-6a6d-417f-ae69-b527766418f8']")
```

Working with streaming data

- How “real” is “real time”?
- If acceptable latency is minutes: Queued ingestion
  - Don’t enable streaming ingestion at database or table
  - Ingest data from blob storage (add new files) or event hub
  - Large volumes of data supported
- If acceptable latency is just seconds
  - Enable streaming ingestion
  - Ingest data from event hub or IoT hub
  - Limited to 4Gb/hour incoming data
- Lowest latency is using custom code
  - APIs for multiple languages available
  - Lots of responsibilities (batch size, resend policy,…)

```
.alter database kustodemo policy
streamingingestion enable
```

Partitioning

- Azure Data Explorer always partitions data into extends
- By default, this happens based on creation time
- In some cases, explicit control over the partitioning can improve performance
- A Partitioning Policy can be setup to achieve this
  - Choose a Partition key and partition type (hash versus range)
- This increases processing time, so only needed in rare cases

Beyond tables

- Besides tables, regular relational databases contain views, functions, stored procedures, …
- In Azure Data Explorer a subset of these objects exist as well:
  - Functions allow to store and reuse logic that can be applied on the data
  - Materialized views are similar, except that they store the result of the view definition on disk (and in the cache), increasing storage and update cost, but decreasing query cost

Creating Functions

- Besides a wide list of built-in functions, users can also create user-defined functions
- These can be stored permanently in the database: Stored Functions
- Or they can be created using the let statement for use in just a single query: Query defined Functions

Let

- The let statement allows to define a temporary variable
- The scope of the variable is just one query
- It can hold a constant value as well as a function
- It reevaluates the expression every time it's used
  - Consider materialize() if you don't want that

```
let nrofsamples = 10;
station
| sample nrofsamples
```

Stored Functions

- Stored Functions are created using the .create function command
- Without parameters they act as a sort of view:

```
.create function smallstations() {station
| where dockcount < 12}

smallstations
```

Kusto querying options

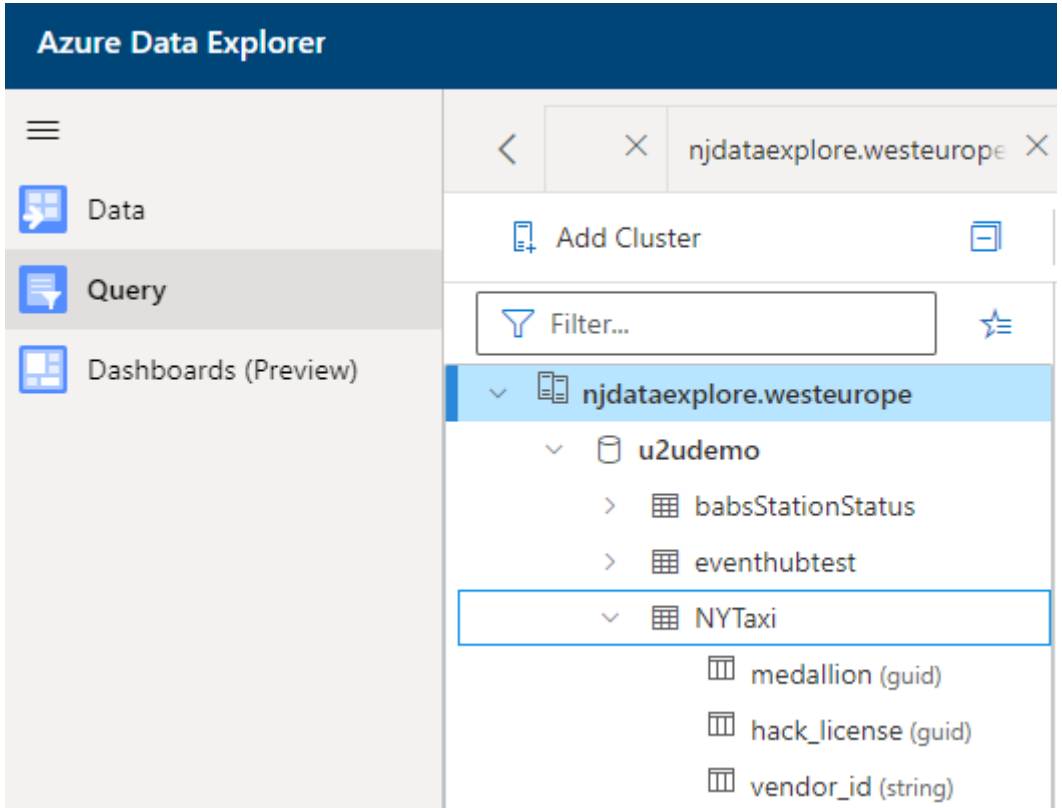
- Web portal
- Power BI
- Azure Data Studio
- Azure Data Factory (or similar pipelines)
- Kusto Explorer
- Azure Synapse Analytics
- Microsoft Fabric

Data Exploration

- Azure Data Explorer supports relational data
  - Similar to traditional sql tables
  - Also supports geographical data
- There is support for semi-structured data
  - JSON and XML support
- Text parsing functions support searching in unstructured data as well
- On top of the querying, there is also support for rendering the results graphically, which makes detecting patterns more easily
- Built-in machine learning helps with clustering, time series forecasting and anomaly detection

Inspecting databases

- You can browse to https://dataexplorer.azure.com
- On the query tab you can add your cluster
- Then in the object explorer you should see the database(s) you created within the server as well



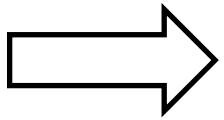
Kusto versus SQL

- ADX supports both SQL and Kusto query language
- The Kusto query language is natively build for Azure Data Explorer
- It supports much more advanced querying options
  - Geographical data, string and JSON parsing, time series, …
- It has more a more procedural nature which fits better the flow of data exploration
  - E.g. in SQL we can only have a single WHERE statement whereas in Kusto we can have multiple filter clauses

Getting started with Kusto

- By putting the explain keyword in front of a SQL query Azure Data Explorer returns the Kusto equivalent of your query

```
explain
SELECT count(*) as measures
, min(bikes_available) as min_bikes
, max(bikes_available) as max_bikes
, station_id
FROM stationdata
GROUP BY station_id
```



```
stationdata
| summarize measures=toint(count())
    , min_bikes=min(bikes_available)
    , max_bikes=max(bikes_available)
    by station_id
| project measures, min_bikes, max_bikes
    , station_id
```

Common queries

- Start with table of view name (best for completion as well)
- Every operation after a vertical bar (|)
- Line breaks wherever you want
- Most common functions:
  - project: select columns or expressions
  - where: WHERE clause (be sure to use == for comparison)
  - take: TOP clause
  - summarize by: GROUP BY and aggregations (also DISTINCT)
  - order by: ORDER BY (remember the nulls first/last option)
  - join: join (using e.g. kind = leftouter to specify type of join)

Demo

Kusto in Power BI

- Power BI supports both Import as well as DirectQuery access to Azure Data Explorer
- Power Query supports Query Folding on Azure Data Explorer
- Web interface for Kusto supports Export to Power BI
- Microsoft Fabric natively support Kusto

Demo

Conclusions

- ADX Cloud-based, scale-out, append-only database for data exploration
- Runs in Azure, Synapse and Fabric
- Supports different options for data ingestion
- Integrates with many client tools
  - Power BI
  - Fabric
  - Azure Data Studio
  - And more with ODBC connector

Platinum partners

creates.

In Summa

Goud partners

Kimura

plainwater
de kracht van heldere data

KASPAROV FINANCE & BI

Zilver partners

rockfeather

Dynamic People

GET RESPONSIVE

Brons partners

Hso

macaw

iqbs

VICTA
BUSINESS INTELLIGENCE

Quanto
collective analytics

ilionx

valcon

VALID
STAY AHEAD

Community partners

broadwick
Data & development recruiters

THE DATA COOKS

Tabular Editor

Datamanzi

Power BI Connector
by DAVISTA

MINOVA

AZURROFINANCE

DATA KINGDOM

volda;
INFORMATIESPECIALISTEN

DashData.

VisionBI
Smart Data Experts

easydash

PBIG
POWER BI GEBRUIKERSGROEP

Session evaluation

Event evaluation