



# Workshop

# Azure Data Explorer

# BASICS

December 1<sup>st</sup> 2022  
9 – 12 CET

**Thomas Pickl**

Cloud Solution Architect  
Advanced Analytics & AI - Automotive Sector  
Microsoft Azure Intelligent Cloud

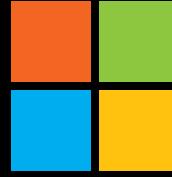


# Learning Objectives

- This is a Level 200 – Level 300 training.
- Understand basic concepts of Azure.
- Get to know Azure Data Explorer and get comfortable using it.
- Feel enabled to identify potential use cases in your daily work and apply your learned skills to prototype / implement them.
- Take away code snippets to work on your own use cases.
- Write your own code to make use of some of the services.
- Get to know people you can ask if you get stuck.

# Logistics | Code Repositories & MDR ADX Cluster

- Breaks
  - 10:15 – 10:30 CEST Coffee Break
- GitHub
  - <https://github.com/thpickl/adx-workshop-master>
- MDR ADX Cluster
  - <https://mdrdataadx.westeurope.kusto.windows.net/>
- ADX Query Best Practices
  - <https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/best-practices>
  - <https://learn.microsoft.com/en-us/azure/data-explorer/named-expressions>



# Microsoft Azure

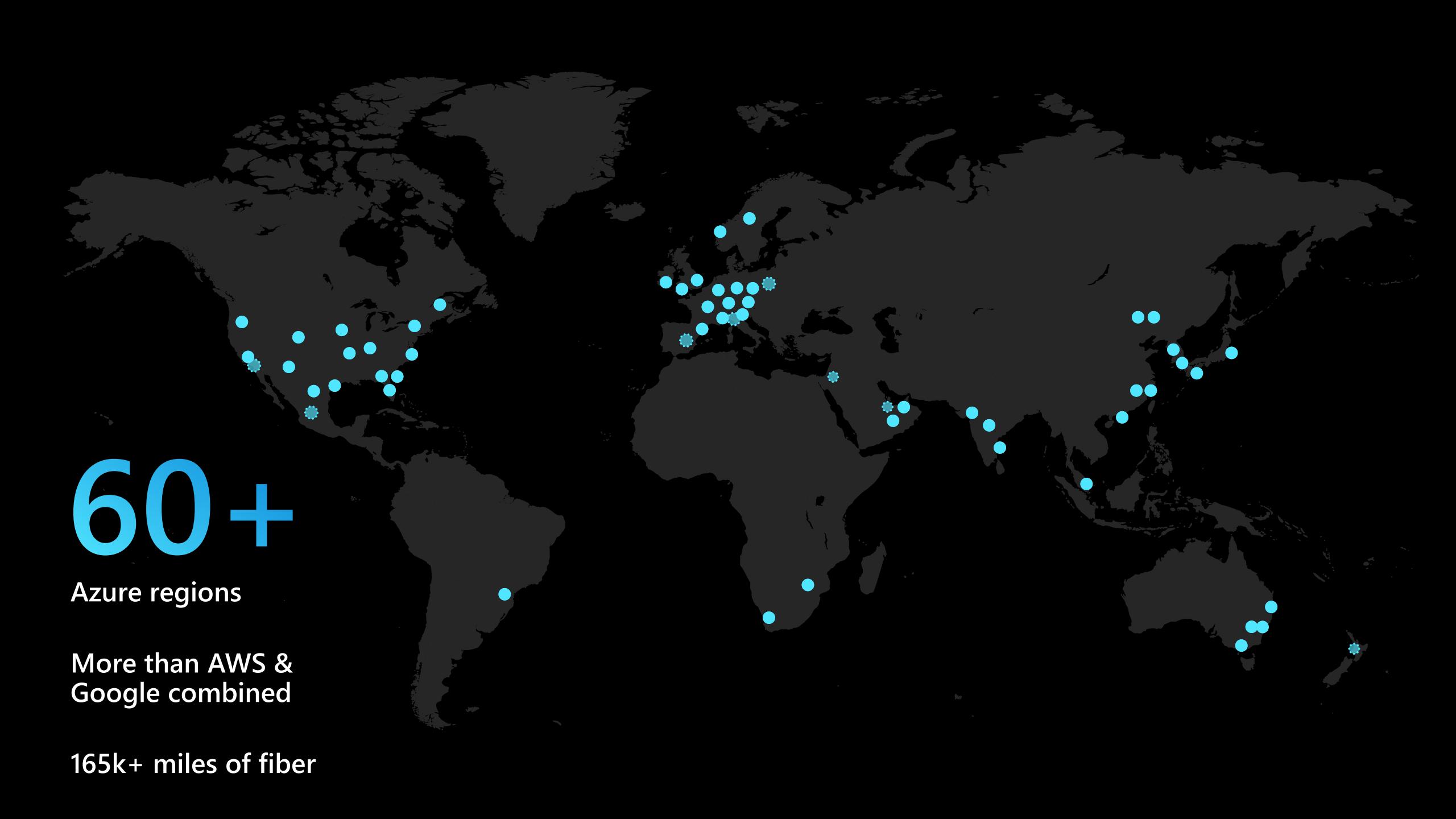
---

Be future  
ready

Build on  
your terms

Operate hybrid  
seamlessly

Trust  
your cloud



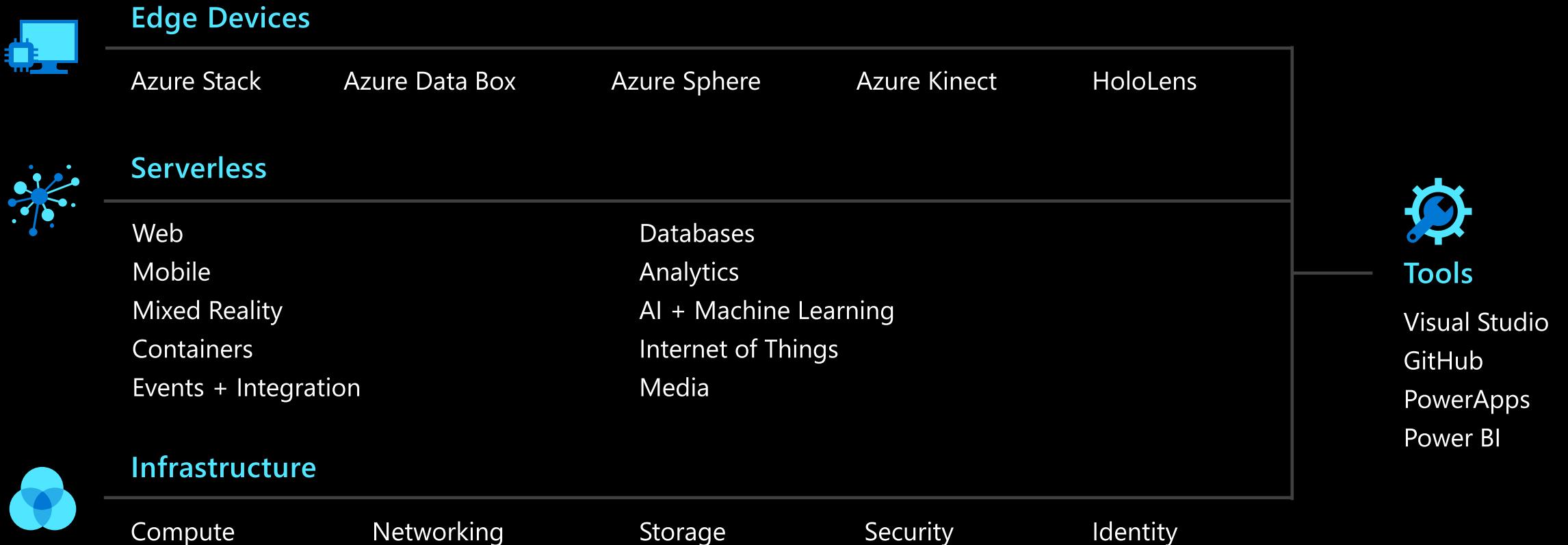
**60+**

Azure regions

More than AWS &  
Google combined

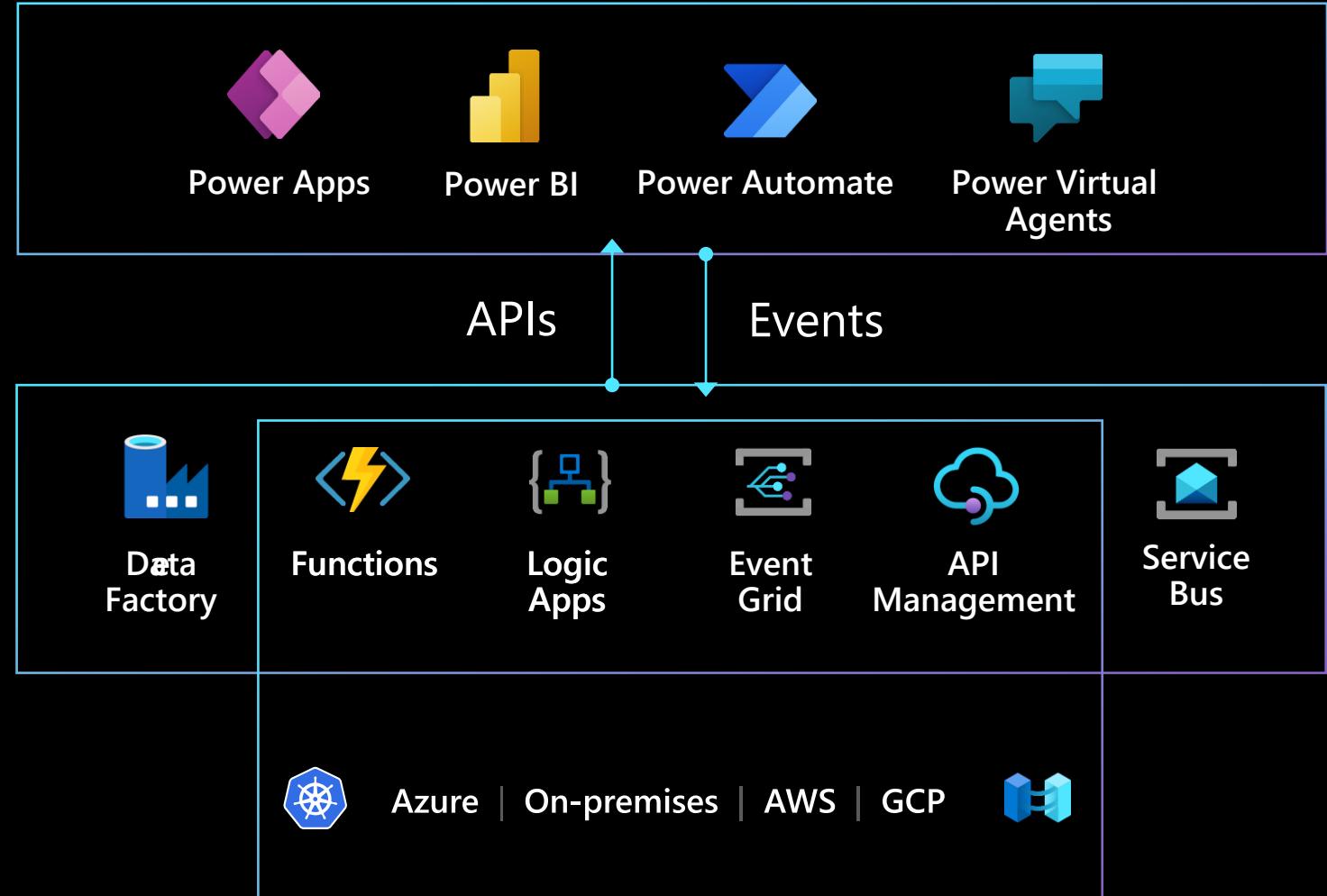
165k+ miles of fiber

# What is Azure?



# Azure Integration Services

Complete. Integrated. Citizen. Pro. Future ready.



Low-barrier to entry pricing model

Low-code business process automation and application integration

450+ out-of-the box connectors, AI for unstructured content, and UI automation (RPA)

Enterprise-grade governance & security

Citizen extensibility with Power Platform

Low-code ETL with Azure Data Factory



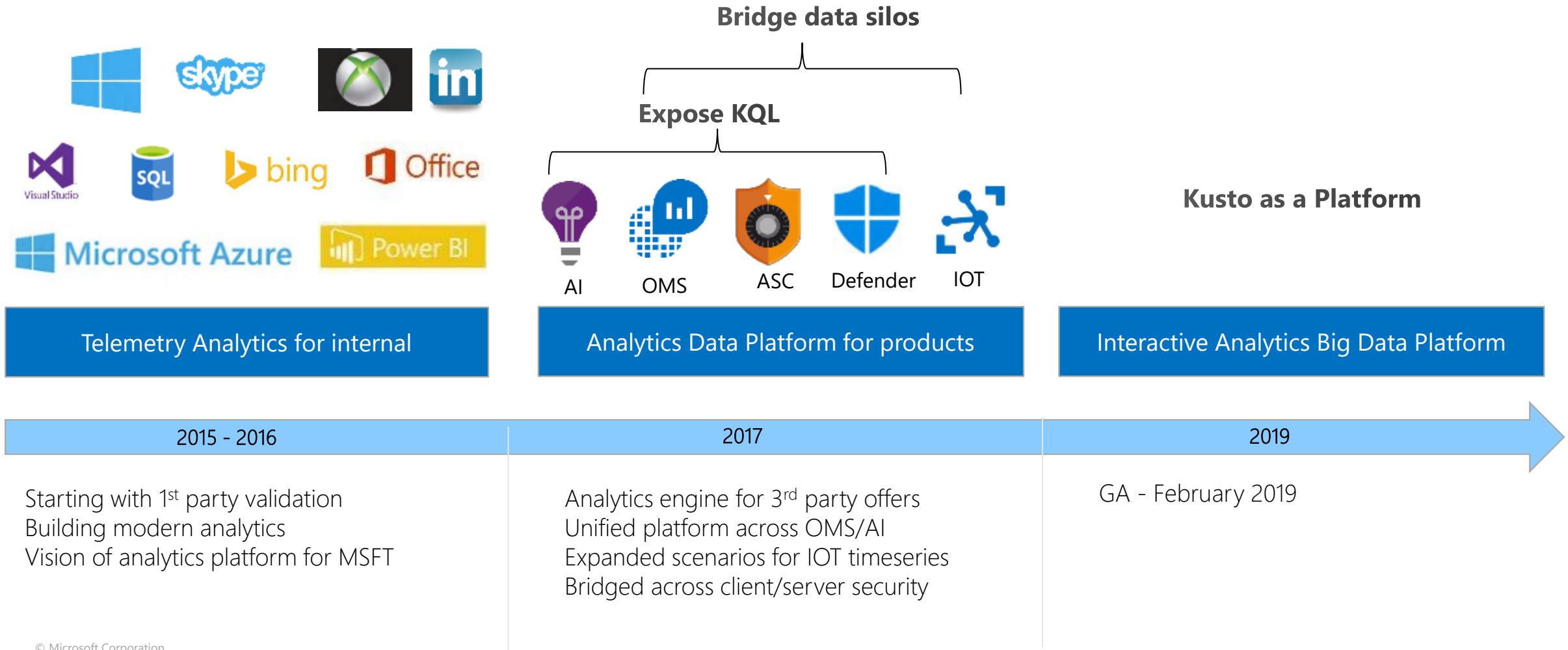
# Azure Data Explorer

## Fast and highly scalable interactive analytics

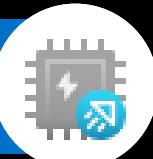
Start exploring your data

L100\_L200

# Azure Data Explorer - Evolution



# Technical Use Cases



Near-real time big data analytics solutions



Near-real time logs analytics solutions for your observability, monitoring, and security data

- Replace infrastructure-based log search solutions
- Complement with your SIEM product
- Accelerate your AI Ops journey



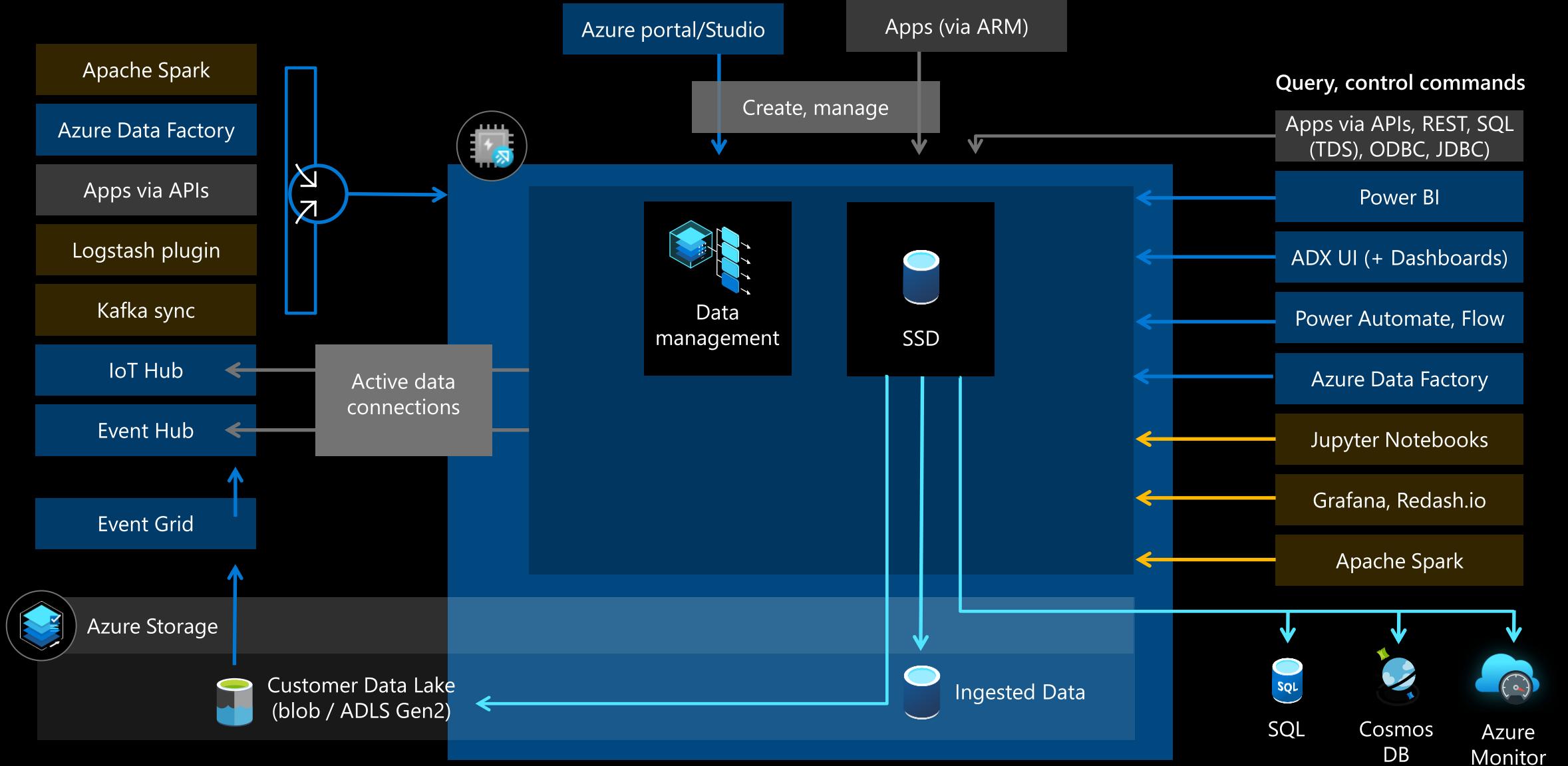
IoT Analytics solution

- Connected devices
- Build cloud base historian

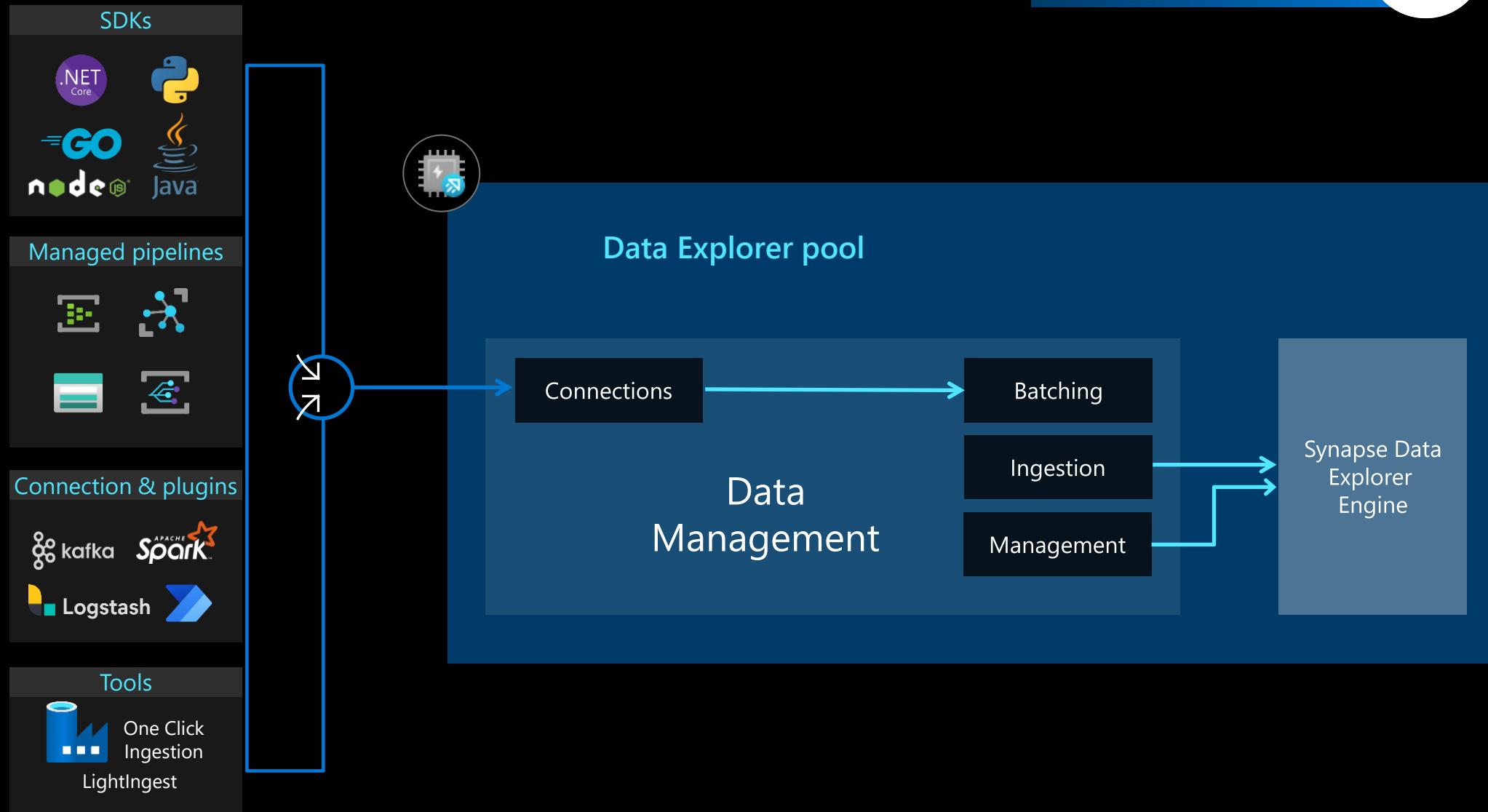
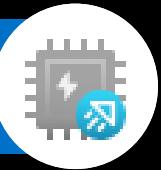


Analytical SaaS solutions

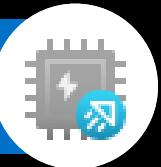
# Data Explorer pool architecture



# Data Management and Ingestion

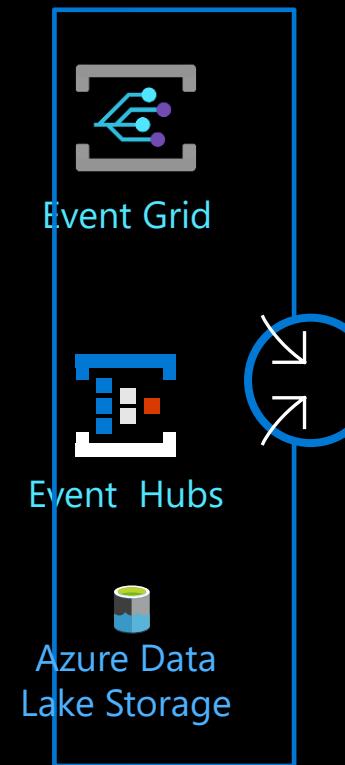


# ADX Web UI One-Click (wizard)



Friction free on-boarding of your Azure data sources to Data Explorer

1. Get data from Azure Data Lake, Blob, Event Hub, or local file
2. Infer the schema automatically from source
3. Create table and mapping automatically from source
4. Generate custom code to start ADX project with one of the supported SDKs
5. Manage data-policies (e.g. retention, batching policies, streaming)
6. Get insight on your data management
7. Ingest one-time data or create connection for continuous data ingestion
  - Continuous (EventHub/EventGrid), one time or backfill (LightIngest)

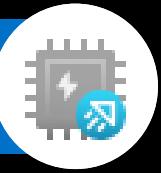


The screenshot shows the 'Data Management' section of the Data Explorer web UI. At the top, there's a brief description of one-click ingestion: "One-click ingestion allows you to quickly ingest data, create database tables, and map structures. Select data from different sources in different data formats to define the table schema. Explore your data as you like—you'll have the option to revert all changes later on. [Learn more](#)". Below this is a 'Quick actions' section with four buttons: 'Ingest new data', 'Create new table', 'Manage data', and 'Insight page'. The main area is titled 'All actions' and contains six cards for different ingestion types:

- Create new table**: Create a table and schema mapping with the Azure Data Explorer web app. (Create, Learn more)
- Create external table**: Create an external table and schema mapping with the Azure Data Explorer web app. (Create, Learn more)
- Ingest new data**: Ingest new data, create database tables, and map structures. (Ingest, Learn more)
- Ingest data From local file**: Ingest data from a local file and create database tables and map structures. (Ingest, Learn more)
- Ingest data From blob**: Ingest data from a local file and create database tables and map structures. (Ingest, Learn more)
- Ingest from blob container**: Ingest data as a one-time or continuous ingestion. (Ingest, Learn more)
- Ingest From ADLS Gen2**: Ingest data from ADLS Gen2 as a one-time or continuous ingestion process. (Ingest, Learn more)
- Ingest from Event Hub**: Set up a data connection to an existing Event Hub for an ongoing ingestion. (Ingest, Learn more)

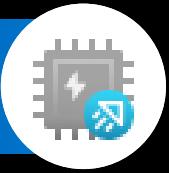
On the right side, there's a 'Documentation' sidebar with links to 'What is one-click ingestion?', 'Use one-click ingestion to create an Event Hub data connection for Azure Data Explorer', and 'Ingest data from a container/ADLS into Azure Data Explorer'. There's also a 'See all →' link.

# Distributed Query Engine



- Storage and Compute isolated
- Auto compression, indexing, optimization
- Semi-structure (JSON) and unstructured data (free text) indexing
- Data is sharded, distributed, cached across nodes on local SSDs and persisted on cloud storage
- Ingestion time-based partitioning + custom partitioning
- Hot and cold data management
- Table level caching + results set caching
- Workload management
- Partitioning
- Materialized views
- External tables for Blob, Data Lake, and SQL

# Kusto Query Language (KQL)



## Simple and powerful

- Rich rational query language (filter, aggregate, join, calculated columns, and more)
- Built-in full-text search, time series, user analytics, geospatial, and machine learning operators
- Out-of-the box visualization
- Easy-to-use syntax + Microsoft IntelliSense
- Highly recognizable hierarchical schema entities

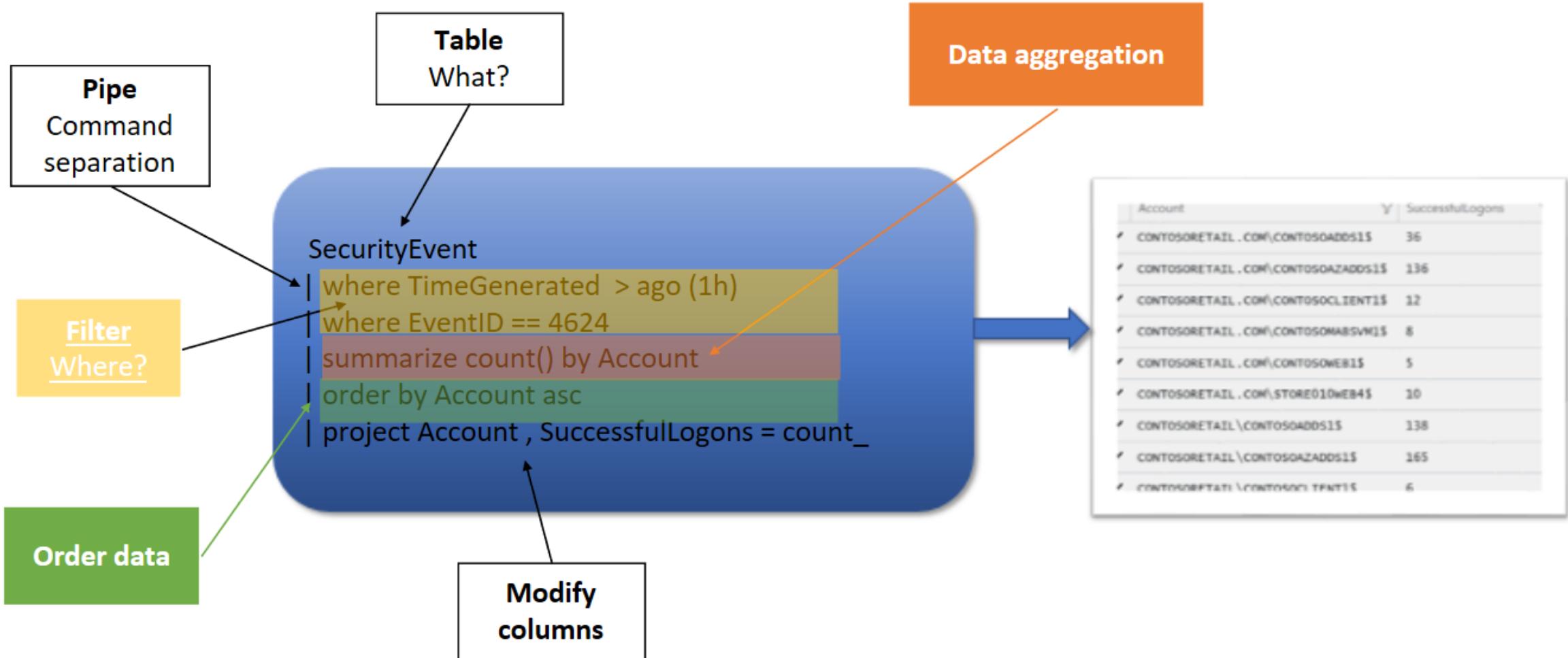
```
1 GithubEvent
2 | where CreatedAt between(now() .. ago(1d))
3 | summarize count() by
  Actor
 CreatedAt
  Id
  Payload
  Public
  Repo
  Type
  abs(number)
  acos(number)
  ago(timespan)
  array_concat(array, ...)
  array_ifc(condition_array, when_true, when_fals...
```

## Extensible

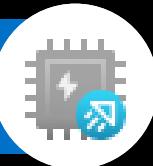
- In-line Python and R
- T-SQL

# Get started with Kusto Query Language (KQL)

## Search query



# Advanced Analytics – Built-in & Extendible



## Out of the box

- Auto Clustering for Diagnosis and RCA
- Anomaly Detection
- Regression
- Forecasting
- Time Series Analysis library



## Distributed Custom Code Execution

- Distributed Python and R execution
- Custom code is embedded in the KQL query



## Spark Integration

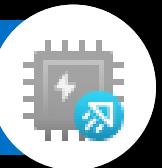
- Native Spark connector for heavy duty model training
- Operationalize model into DX pool for scoring



## Tools

- Jupyter Integration with KQL Magic
- Python, Java SDKs

# Geospatial query



## 1. Support for geospatial clustering

- Transformation from coordinates to geospatial clusters and back
- Support for Geohash, S2 and H3 cells

## 2. Distance

- Calculate the distance between two points or a point and a line

## 3. Contains

- Check whether a point is in a given circle or polygon

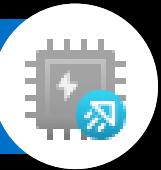
## 4. Advanced features

- Densification to control shape edges, geodesics or straight lines
- H3-Rings

Geospatial coordinates are interpreted as represented per the WGS-84 reference system.



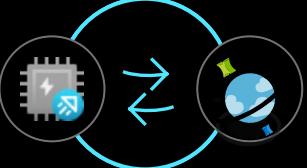
# Cross Queries



 Between Data Explorer databases and pools

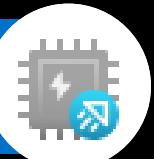
 Query SQL pool from Data Explorer pool

 Query Azure Monitor from Data Explorer pool

 Query CosmosDB from Data Explorer pool

 Query Data Lake from Data Explorer pool

# Data Sharing



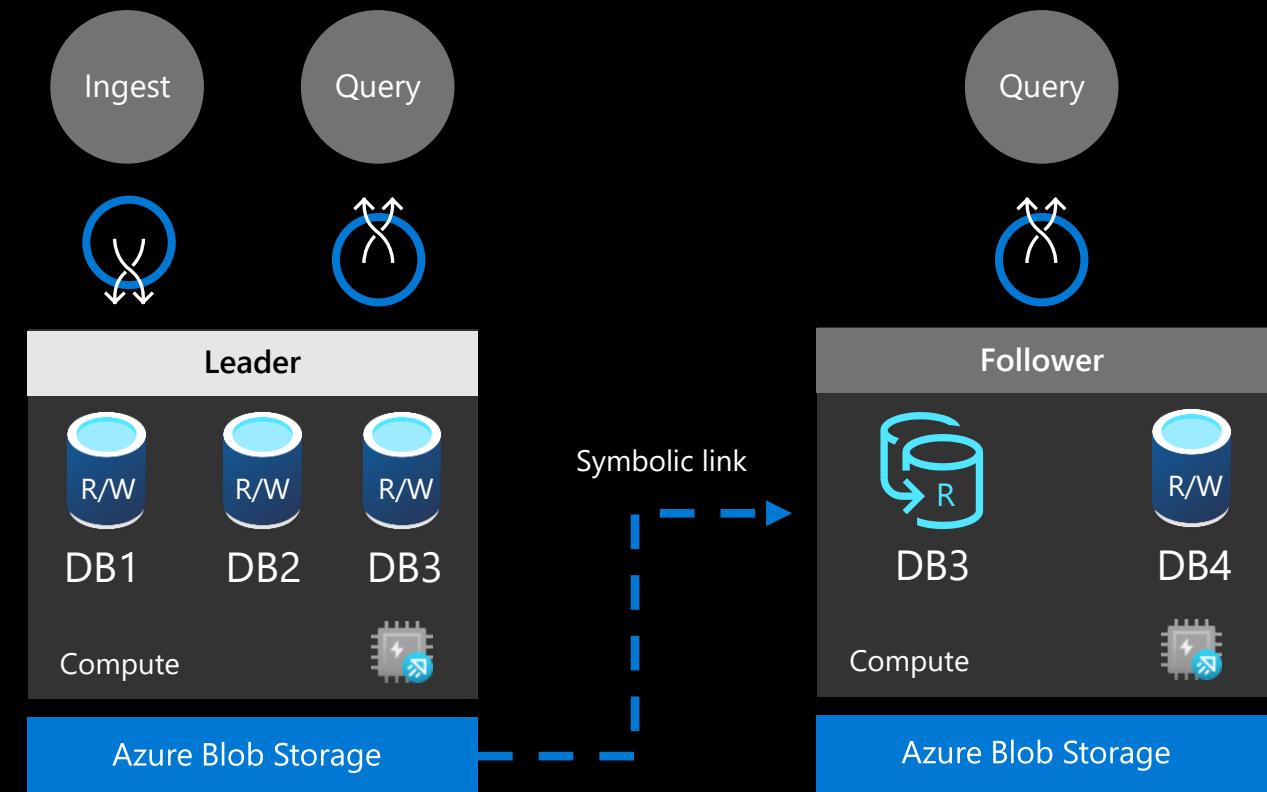
## ❖ Share data within the company

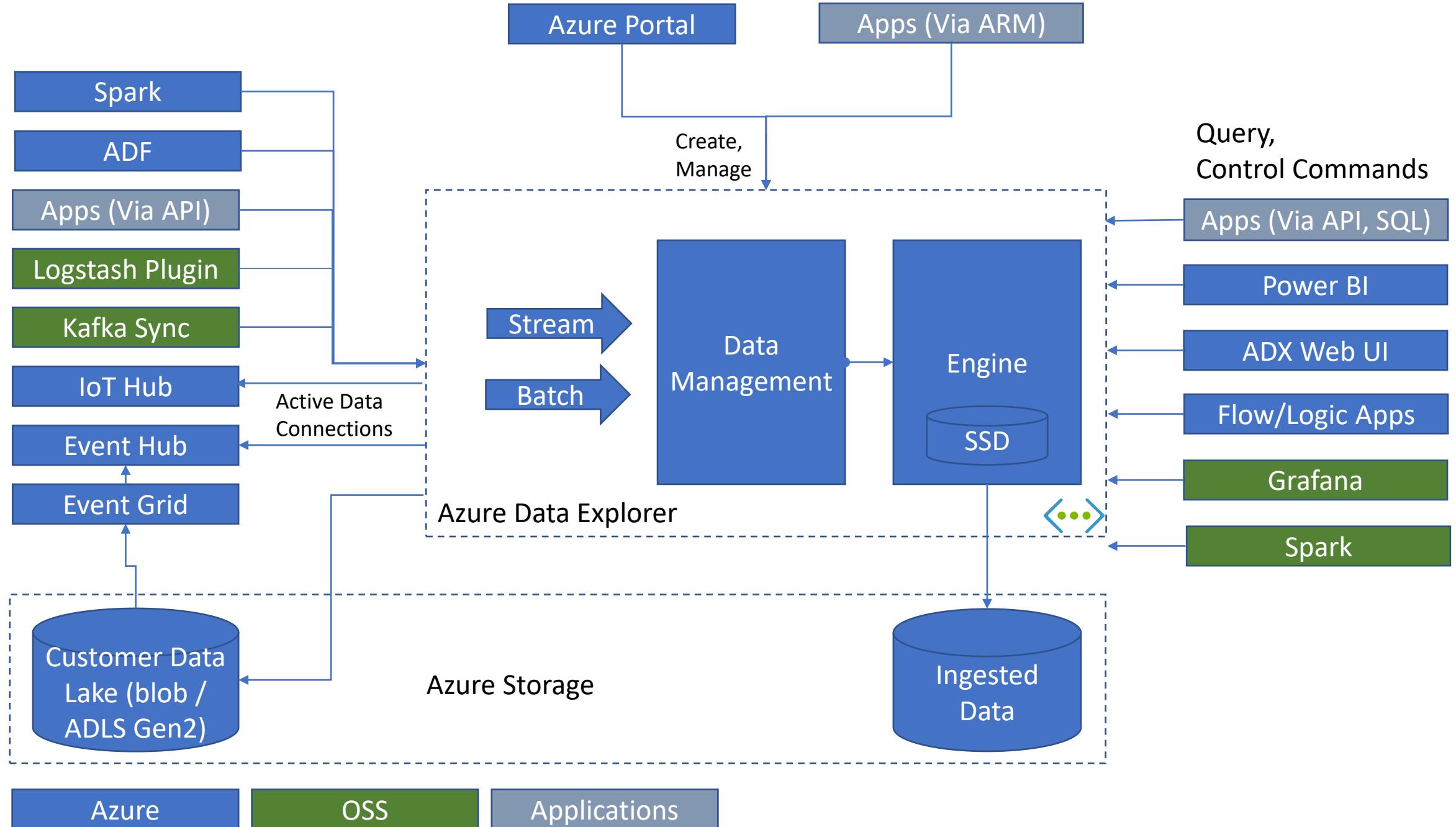
- Load balancing
- Data-as-a-service
- Hub and Spoke model
- Chargeback



## No Maintenance

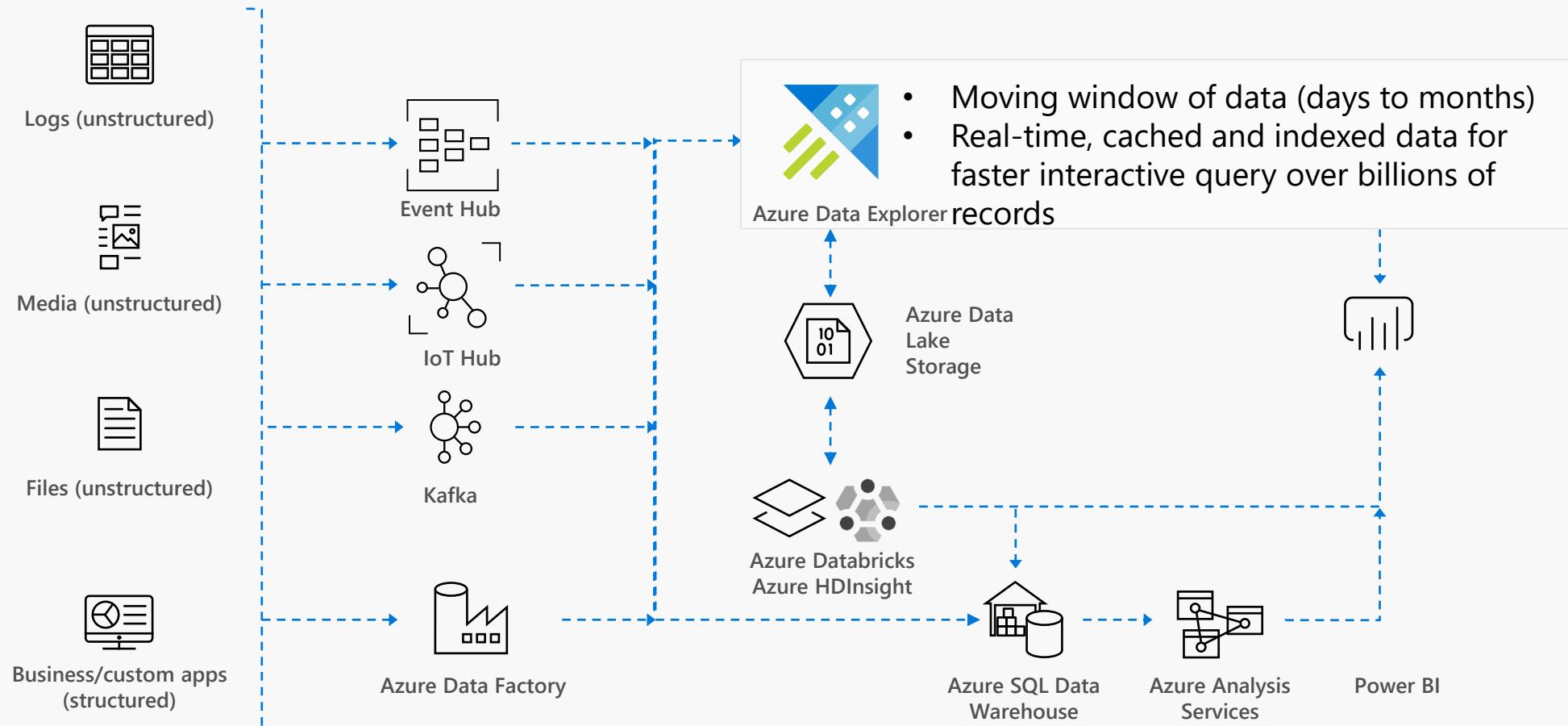
- In-Place Data Sharing
- No data pipeline to maintain
- Near real-time updates





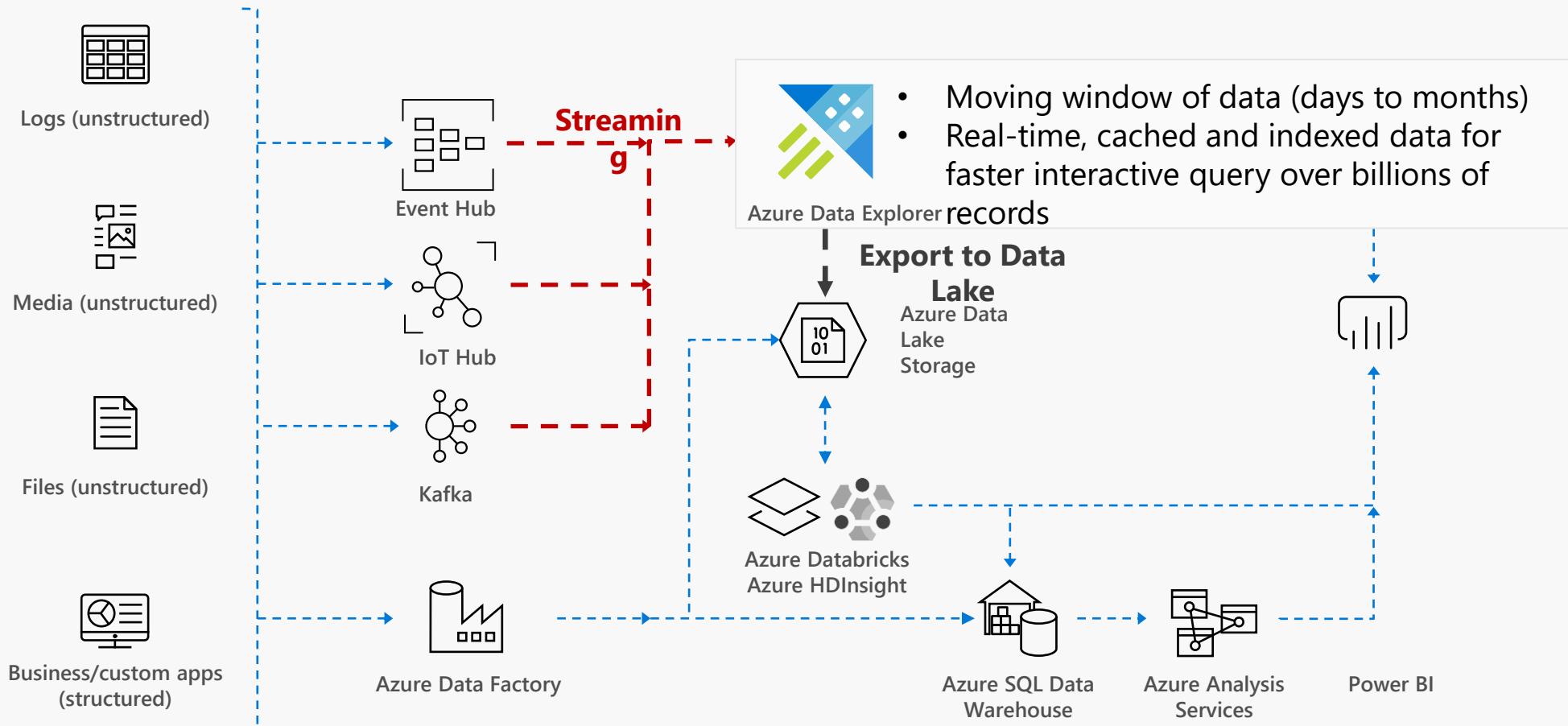
# Where will you fit ADX in Data Lake and Analytics?

Cached and Indexed data for fast interactive query



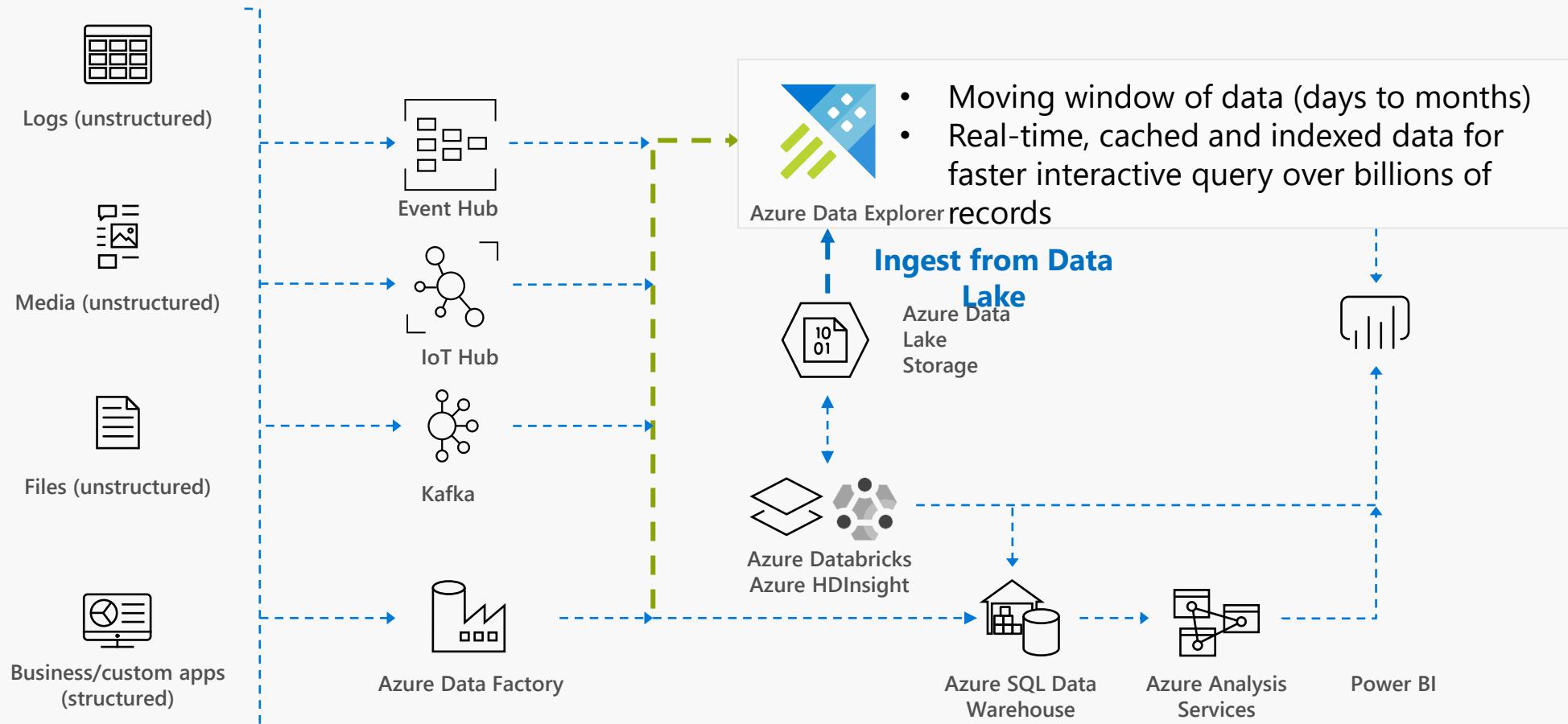
# Integration: Streaming Ingestion

Land your data in ADX and then export to the data lake



# Integration: Batch Ingestion

Ingest data from Data Lake or other sources in batch mode



# Azure Data Explorer overview

## 1. Capability for many data types, formats, and sources

Structured (numbers), semi-structured (JSON\XML), and free text

## 2. Batch or streaming ingestion

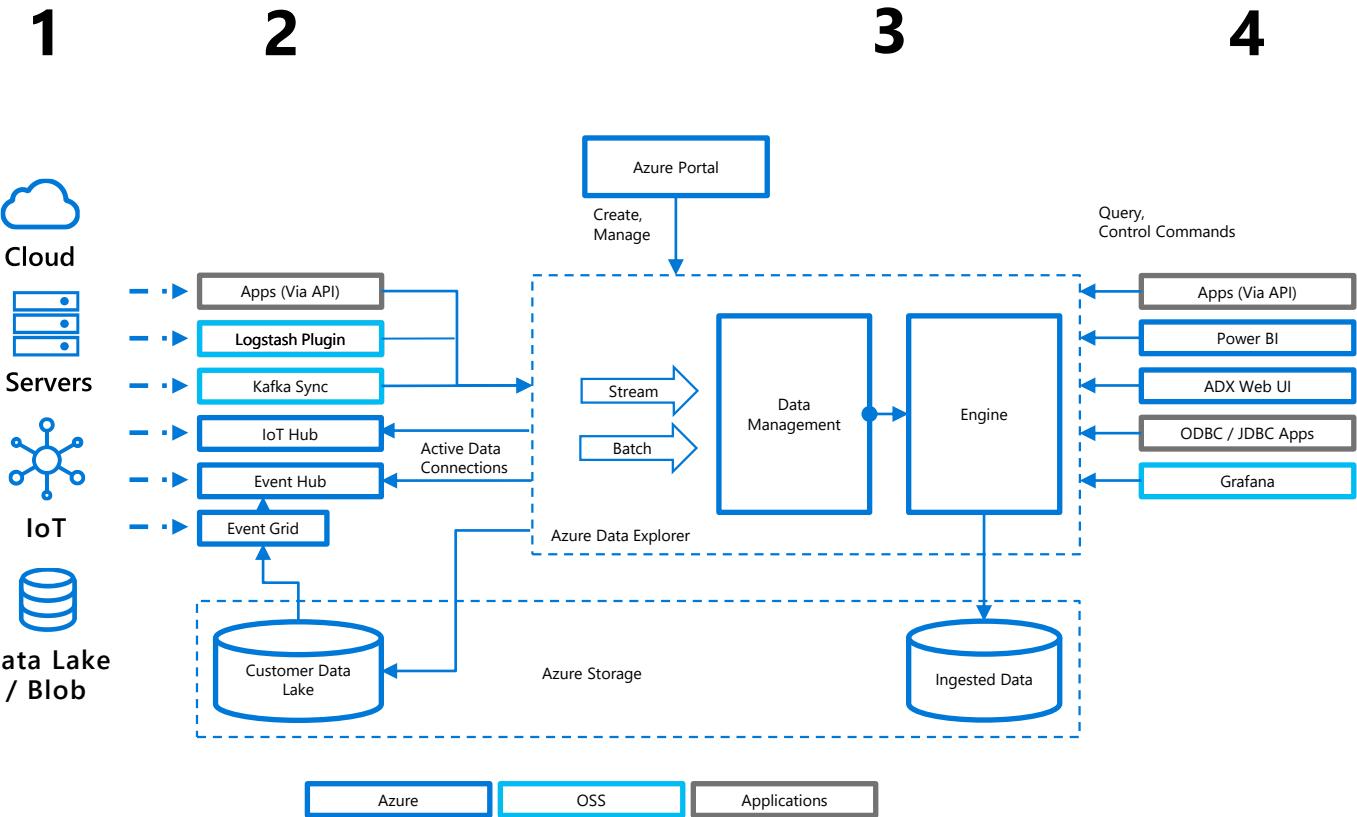
Use managed ingestion pipeline or queue a request for pull ingestion

## 3. Compute and storage isolation

- Independent scale out / scale in
- Persistent data in Azure Blob Storage
- Caching for low-latency on compute

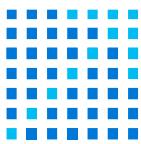
## 4. Multiple options to support data consumption

Use out-of-the box tools and connectors or use APIs/SDKs for custom solution



# Intuitive querying

Designed for data exploration



## Simple and powerful

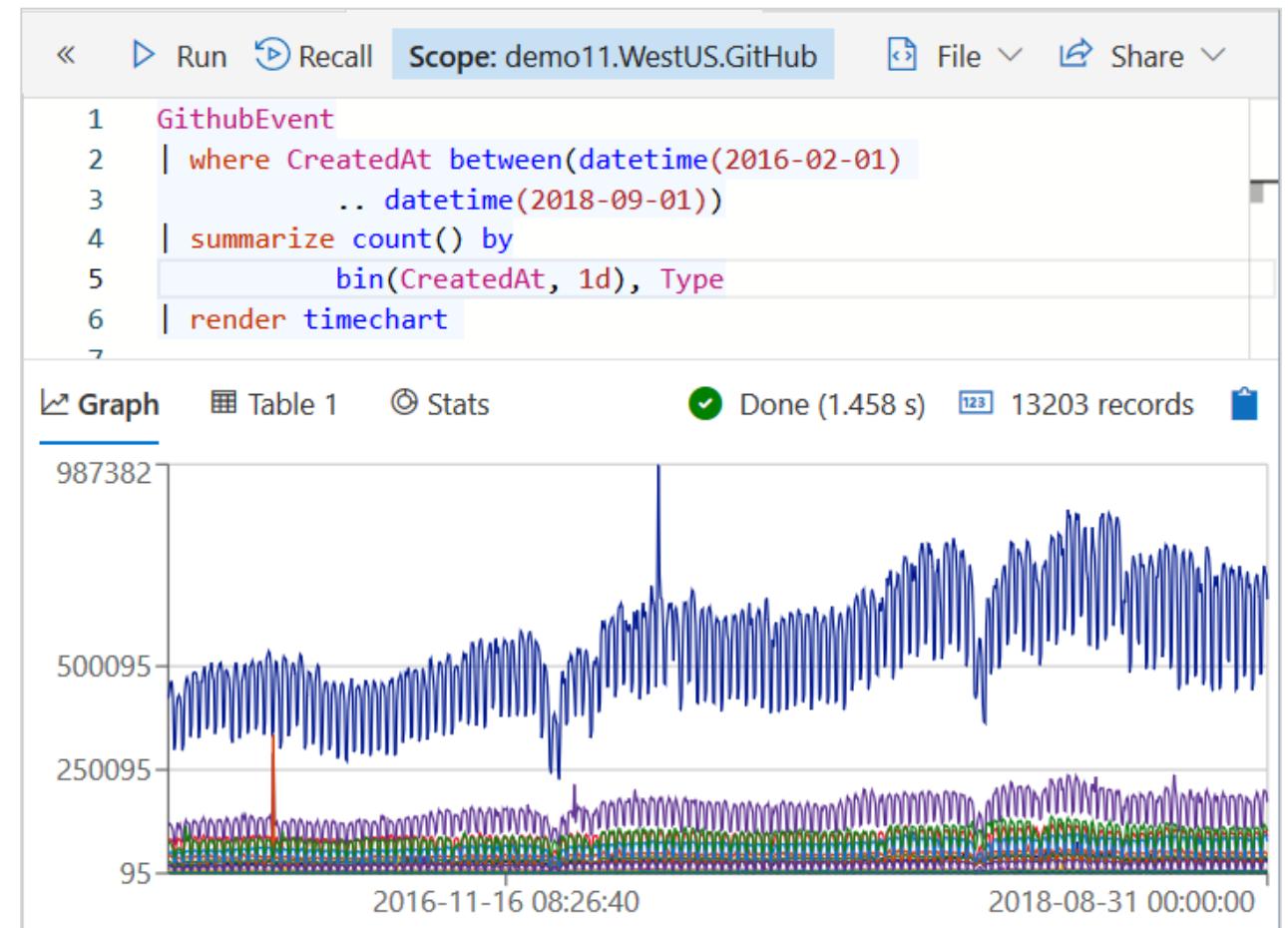
- Rich rational query language (filter, aggregate, join, calculated columns, and more)
- Built-in full-text search, time series, user analytics, and machine learning operators
- Out-of-the box visualization (render)
- Easy-to-use syntax + Microsoft IntelliSense
- Highly recognizable hierarchical schema entities

## Comprehensive

- Built for querying over structured, semi-structured and unstructured data simultaneously

## Extensible

- In-line Python
- SQL



# Azure Data Explorer customers



"In terms of usability it is beyond compare any other database query language we have used so far."

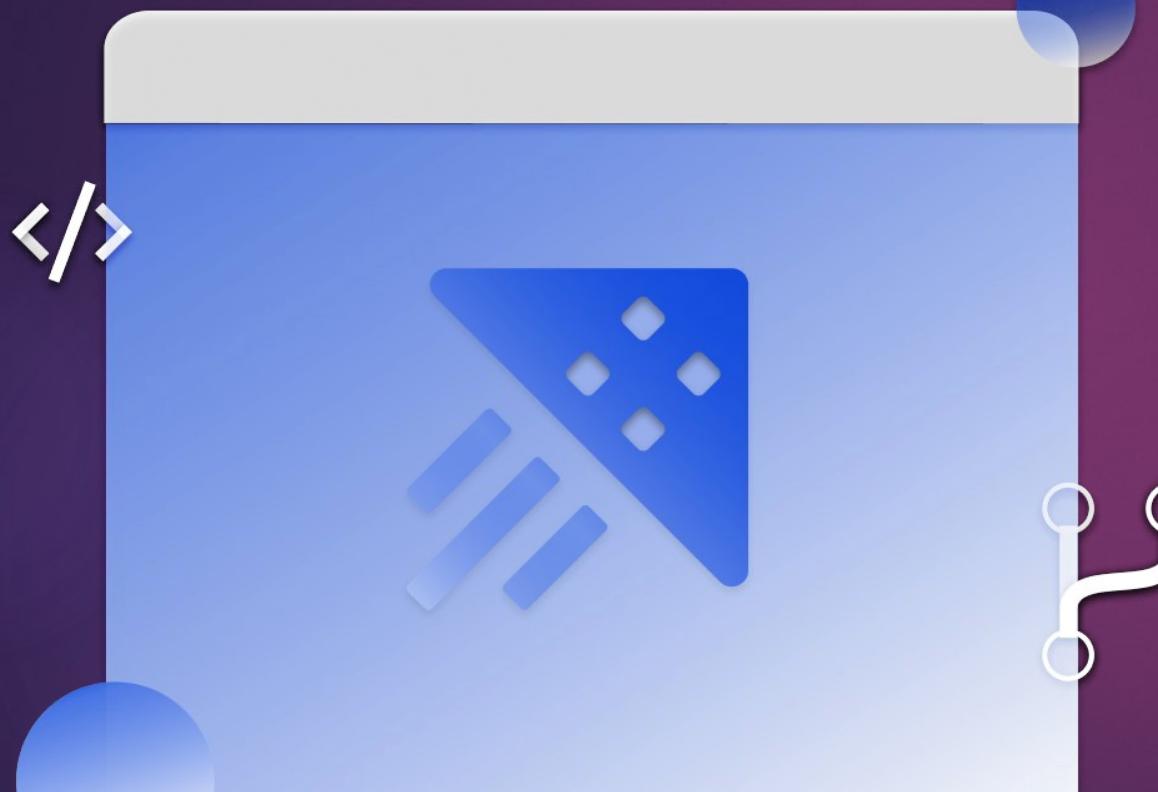
*Emilian Ertel  
Senior Key Expert*



"The solution was so simple that we were up and running in a week, ingesting and analyzing 17 TB of data per day."

*Ariel Pisetzky  
Vice President of IT*





# Visualizing data with **Azure Data Explorer**

-MITUL KACHERIA, RONY LIDERMAN

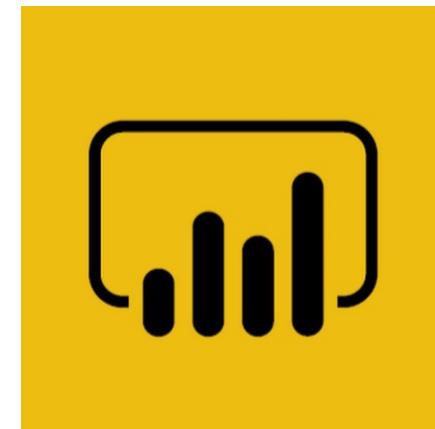
# Supported dashboarding solutions

## 1. Native connectors

- Power BI
- Grafana
- Kibana (K2)
- Redash
- ADX Dashboards (new!)

## 2. ODBC/JDBC connectors

- Power BI (ODBC)
- Tableau (ODBC)
- Qlik (ODBC)
- Sisense (JDBC)



kibana



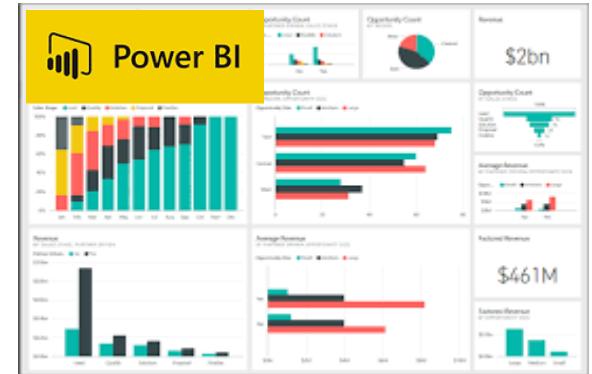
# Power BI

## Highlights

- ▶ Easy-to-use
- ▶ General-purpose BI platform
- ▶ Affordable pricing
- ▶ Highly interactive

## Kusto integration

- ▶ Data retrieval modes
  - ▶ Import mode
  - ▶ DirectQuery mode
- ▶ Composite model
  - ▶ Allows a combination of data from import and DirectQuery datasets
  - ▶ Triggers filter operations instead of costly joins
- ▶ Parameters – WIP
- ▶ Important to read [best practices](#) to get the right perf



# Grafana

## Highlights

- ▶ Open source
- ▶ Real-time
- ▶ Metric visualization service
- ▶ Commonly used for visualizing time series data

## Kusto integration

- ▶ Query Language Intellisense
- ▶ Parameters – parameters injection to queries enable performant query execution
- ▶ NRT – frequent dashboard refresh (down to 1 sec intervals) for up-to-date data
- ▶ Alerts – Integrated alert logic unlocks operational workflows based on the data collected



# Redash

## Highlights

Open source

- ▶ Data visualization tool with a good fit for **real-time analytics**
- ▶ Supports alerts but carries a limited set of visualization types

## Kusto integration

- ▶ **Integration by Dodo Pizza** – a Russian ADX user who's using both ADX and Redash
- ▶ Future improvements for supporting Intellisense and additional performance improvements are being discussed



# Kibana

## Highlights

Open source data visualization plugin for Elasticsearch

- ▶ Provides **real-time, time series** analytics visualizations
- ▶ Advanced **search** capabilities

## Kusto integration

- ▶ **Elastic migration enabler** – Kibana is a heavily used Elastic frontend in organizations that use Elastic
- ▶ **Search** – Powerful search capabilities allow semi ‘query like’ data exploration
- ▶ Dashboards not currently supported



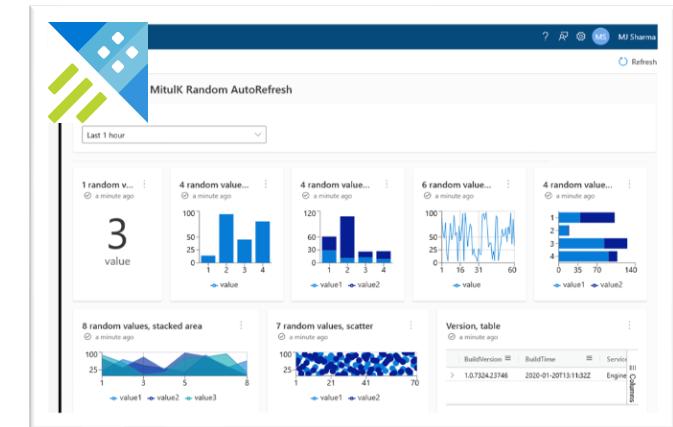
# ADX Dashboards

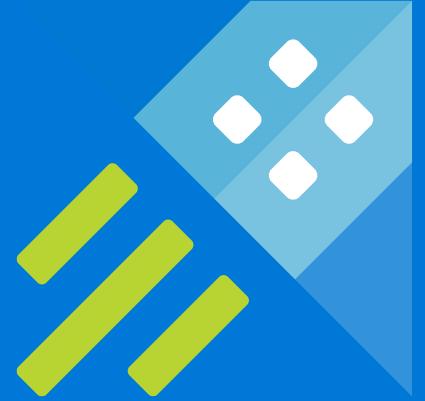
## Highlights

- ▶ Dashboarding integrated into the ADX web explorer
- ▶ Freely available for all ADX customers
- ▶ Performance focused
- ▶ Will support interactivity and real-time scenarios

## Kusto integration

- ▶ **Dashboards in ADX web explorer**
  - ▶ Import existing queries into a dashboard
  - ▶ Explore queries from a dashboard
- ▶ **Parameters** – parameter injection into queries enables performant query execution
- ▶ **Interactivity (WIP)** – support for cross filtering, drill throughs





# Real-Time Interactive Analytics with Azure Data Explorer

Focus: Geospatial capabilities

# Real-Time Analytics Use Cases

## Retail

Consumer Engagement



Pricing optimization, IoT

## Financial

Risk And Revenue Management



Risk and Fraud, Due diligence, Audit

## Oil/Gas & Energy

Grid Ops, Asset Optimization



Industrial IoT

## Security

Signal correlation



Security Intelligence, Threat detection

## Healthcare

Sensor Data



IoT device analytics

## Advertising

Recommendation Engine



Personalized offers, campaign management

## Media Entertainment

Consumer Engagement Analysis



Sentiment analysis, Content recommendation

## Automotive

Manufacturing, AI



Connected cars, Fleet management

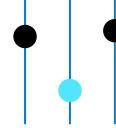
# Azure Data Explorer

Interactive analytics service for fast-flowing data



## Cuts down time-to-insight

Get real-time insights from fast-flowing data



## Fully managed

PaaS, Autoscale and maintenance-free indexing



## Enables data-driven culture

Proven intuitive and easy-to-use query language

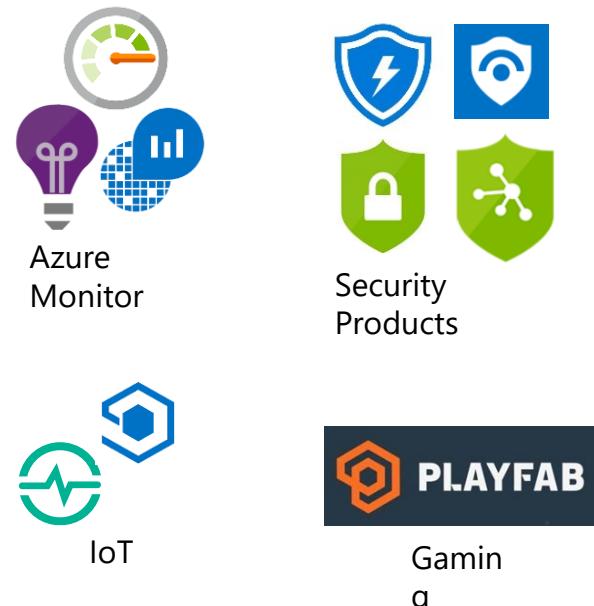
# Proven Technology

In production since 2015 for internal Microsoft workload, GA since Feb 2019.

Battle tested for Microsoft  
internal workload



The platform for analytical  
solutions (SaaS)



Available as PaaS



#PoweredByADX

## Common Use Cases

### Time Series, Logs, Events, Transactions Data analytics platform

- Consolidate and correlate your logs and events data across on-prem, cloud, 3<sup>rd</sup> party
- Replace legacy log search solutions - save cost, infra, and index management overhead
- Accelerate your AI Ops journey (pattern recognition, anomaly detection, forecasting, etc.)
- IoT Analytics on Time Series and Telemetry data
- Replace HBase and Time Series databases

### Build Analytical SaaS Solutions

- Build multi-tenant or single-tenant SaaS analytics solution for Time Series, Logs, Events, Transactions, and security data

### Analytic sandboxes

- Short term ad hoc exploratory analysis

# Geospatial Use Cases

## Public sector

- Calculate the impact of natural disasters
- Geofencing – raise alarms if individuals leave or enter a certain area
- Crime prediction

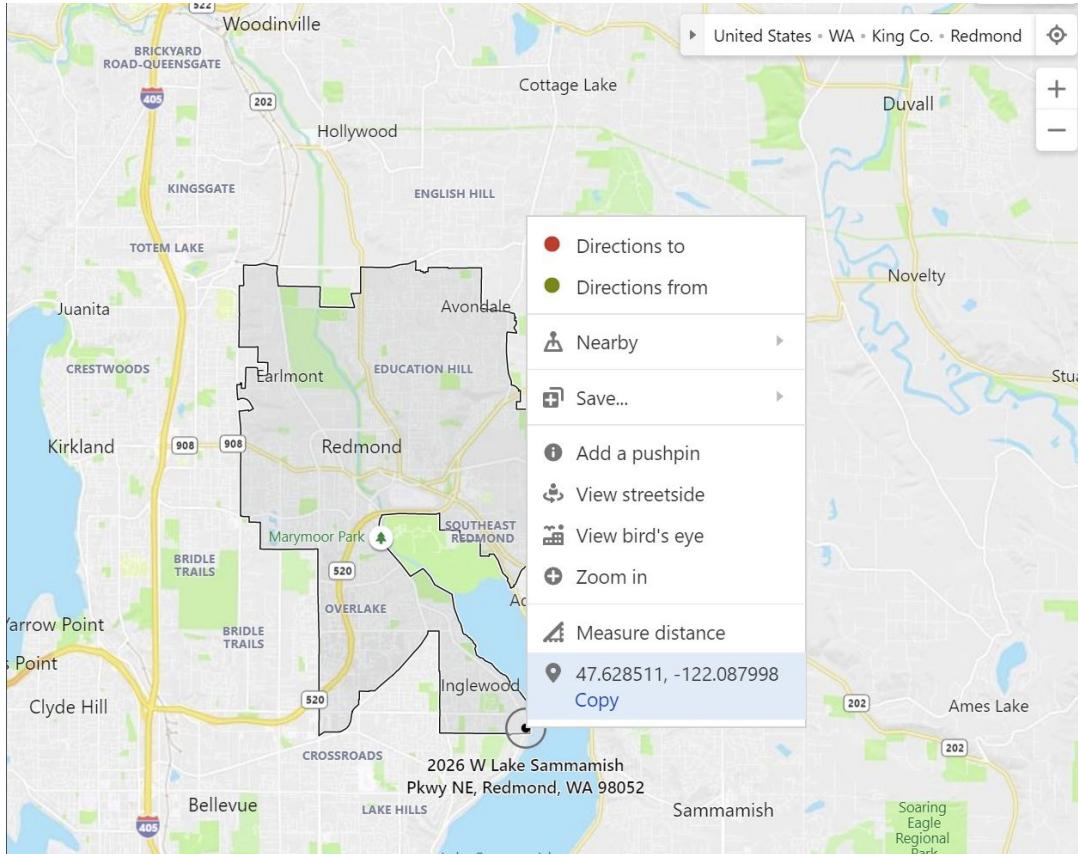
## Logistics & Transportation

- Add the geo-clustering secret sauce to timeseries analytics to create better predictions for the supply chain
- Predict road segments (line strings) conditions as a base component for autonomous driving
- Raise events if a vehicle is not on its route (automotive)

## Financial sector

- Tracking changes to real estate development over time
- Analyze demographic data in conjunction with geographical regions

# Customer cases – Azure Sentinel



Microsoft Azure    Search resources, services, and docs (G/)

Home > Azure Sentinel workspaces > Azure Sentinel - Logs

### Azure Sentinel - Logs

New Query 1\*

General

- Overview
- Logs
- News & guides

Threat management

- Incidents
- Workbooks
- Hunting
- Notebooks

Configuration

- Data connectors
- Analytics
- Playbooks
- Community
- Settings

Security

Time range : Last 24 hours

```
SigninLogs
| where TimeGenerated >= ago(1d)
| where ResultType == 0
| extend longitude = todouble(LocationDetails['geoCoordinates'][‘longitude’]),
      latitude = todouble(LocationDetails[‘geoCoordinates’][‘latitude’])
| where geo_point_in_polygon(longitude, latitude,
      dynamic([{"type": "Polygon", "coordinates": [
          [-122.164216, 47.711740],
          [-122.084565, 47.714050],
          [-122.077698, 47.627585],
          [-122.142930, 47.627585],
          [-122.144303, 47.660894],
          [-122.162499, 47.662743],
          [-122.164216, 47.711740]
        ]]}])
| project TimeGenerated, Identity, longitude, latitude
```

Completed

Table Chart Columns Add bookmark

Drag a column header and drop it here to group by that column

TimeGenerated [UTC]	longitude	latitude	Identity
12/16/2019, 9:44:53.082 PM	-122.121	47.681	Ajeet Prakash (MSTIC)
12/16/2019, 9:44:53.082 PM	-122.121	47.681	Ajeet Prakash (MSTIC)
12/16/2019, 9:44:53.082 PM	-122.121	47.681	Ajeet Prakash (MSTIC)
12/16/2019, 9:44:54.643 PM	-122.121	47.681	Ajeet Prakash (MSTIC)
12/16/2019, 10:44:57.701 PM	-122.121	47.681	Ajeet Prakash (MSTIC)

<https://gist.github.com/JohnLaTwC/cadc94bb8c9bd3d67dc3db56bfc829ae>

<https://twitter.com/JohnLaTwC/status/1206787684678524928>

# Basic functions – geo\_distance\_2points()

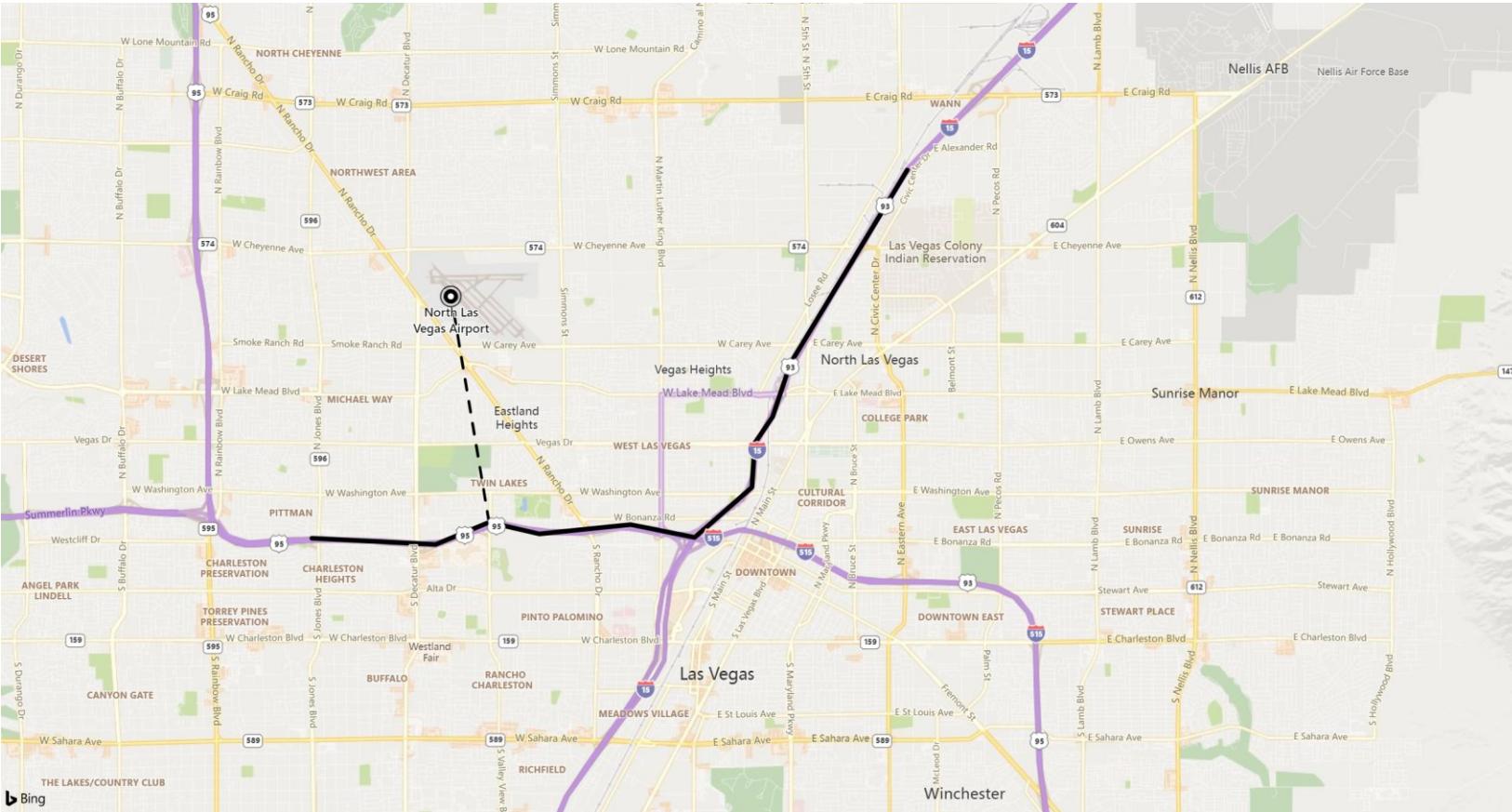
- Calculates distance between 2 coordinates on earth
- Earth is approximated as a perfect sphere and the coordinates should be in WGS-84
- The calculation is done using the [Haversine formula](#)
- Example: distance from Stuttgart to Munich in km:



```
let stuttgart = dynamic({ "longitude":9.182552, "latitude":48.771449});  
let munich = dynamic({ "longitude":11.582676, "latitude":48.130847});  
print strcat(toint(geo_distance_2points(  
    todouble(stuttgart.longitude), todouble(stuttgart.latitude),  
    todouble(munich.longitude), todouble(munich.latitude)) / 1000), ' km')
```

# Basic functions – geo\_distance\_point\_to\_line()

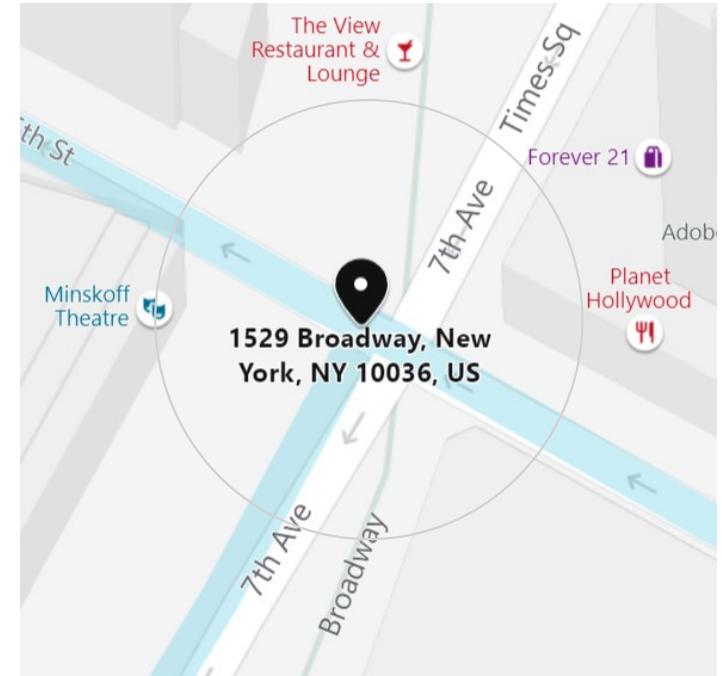
- Calculates the shortest distance between a coordinate and a line on Earth.
- Example: shortest distance between North Las Vegas Airport and nearby road:



```
print distance_in_meters = geo_distance_point_to_line(-115.199625, 36.210419,  
dynamic({ "type": "LineString", "coordinates": [[lon, lat]]}))
```

# Basic functions – geo\_point\_in\_circle()

- Use it for Nearby, Proximity queries
- Earth is approximated as a perfect sphere and the coordinates should be in WGS-84
- Circle is a spherical cap on earth
- The radius of the cap is measured along the surface of the sphere
- Example: Find all New York Taxi pick-ups near Times Square, within 50 meters:

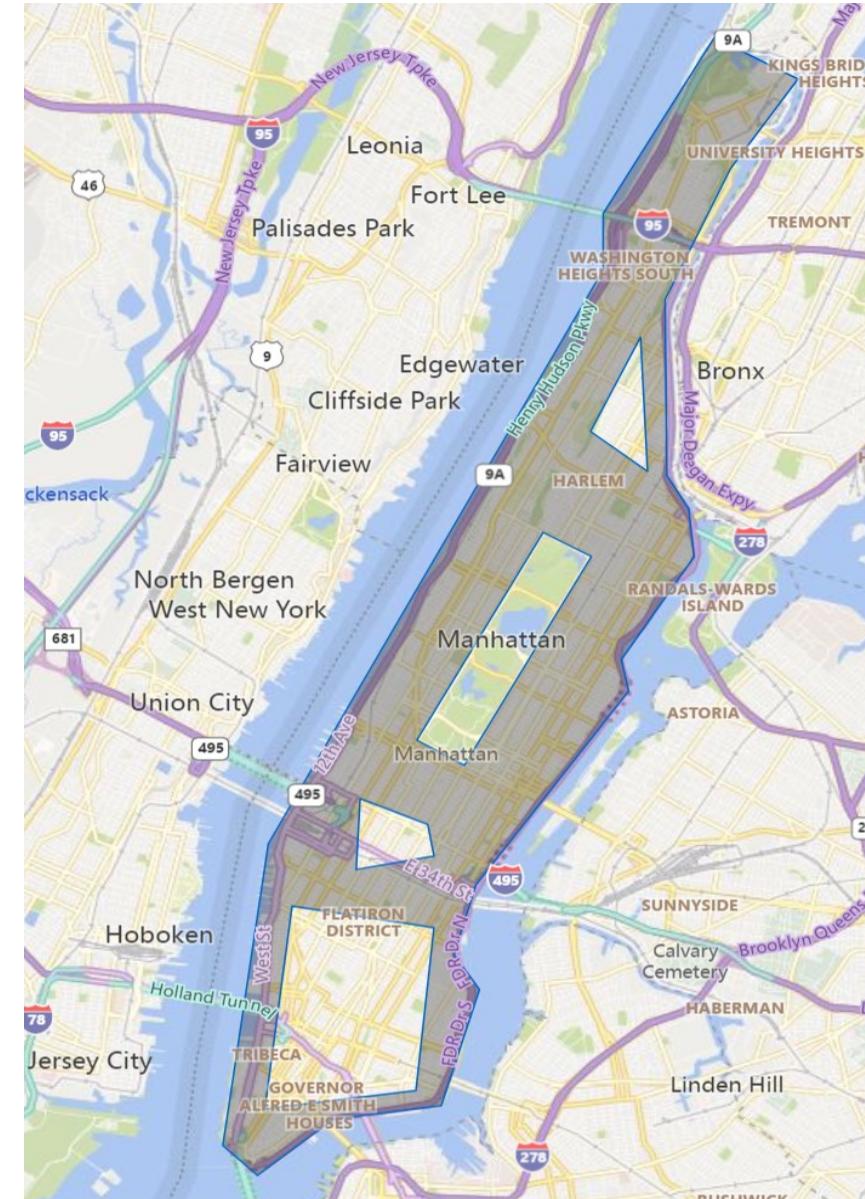


```
let times_square = dynamic({"longitude": -73.98552, "latitude": 40.757960});  
Trips  
| where geo_point_in_circle(pickup_longitude, pickup_latitude,  
    todouble(times_square.longitude), todouble(times_square.latitude), 50) > 0
```

# Basic functions – geo\_point\_in\_polygon()

- Use it for coordinates containment test within a complex shape

```
print in_polygon =  
    geo_point_in_polygon(-73, 40,  
    dynamic({ "type": "Polygon",  
        "coordinates": [[[0,0],[10,10],[10,1],[0,0]]]}))
```

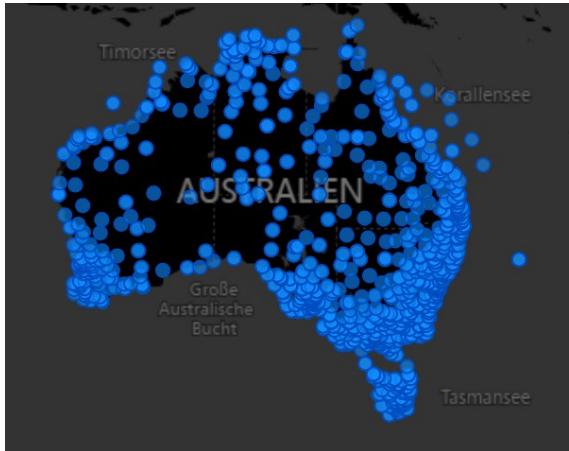


# Geospatial clustering

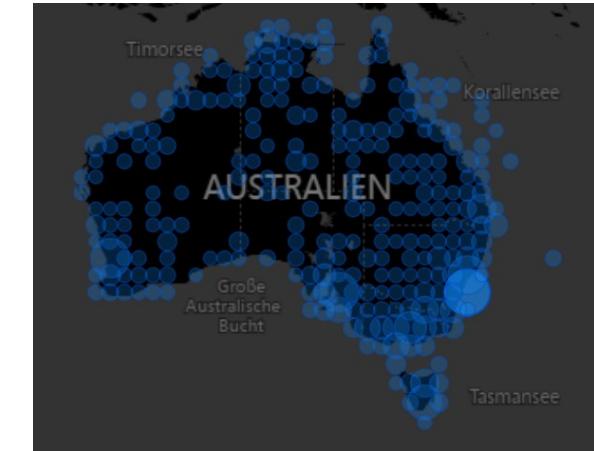


Weather  
| where ...

Extend with a cluster



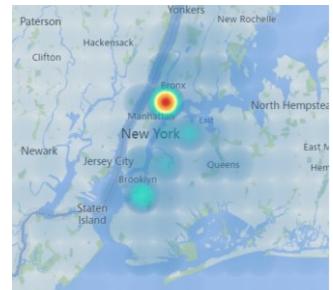
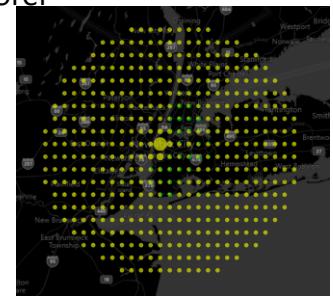
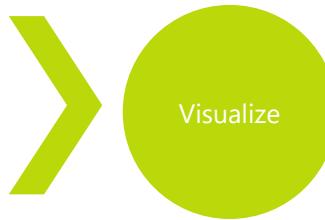
| summarize  
count() by  
cluster



Summarize

Get centroid

- PowerBI
- Kusto Explorer
- ...



All visuals have been implemented using the map visualization in PowerBI

# Geospatial clustering – Geohash vs S2 cells

## Geohash

18 accuracy levels

Rectangular area on a **plane** surface

**Common prefix** of Geohashes indicate on **proximity**

## S2 Cell

31 levels of hierarchy

Cell on a **sphere** surface and its edges are geodesics

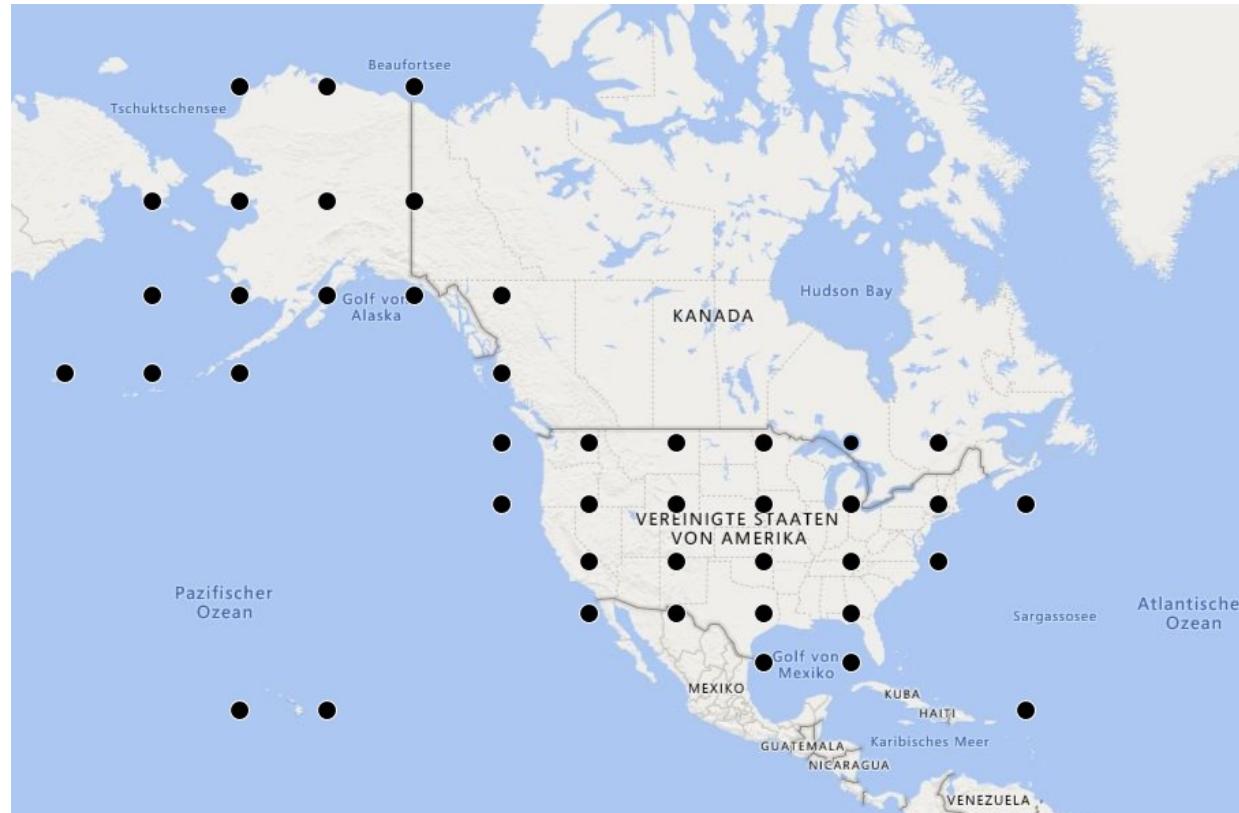
Good **preserveness** of the **cell center** during level increase

<https://en.wikipedia.org/wiki/Geohash>

[http://s2geometry.io/devguide/s2cell\\_hierarchy](http://s2geometry.io/devguide/s2cell_hierarchy)

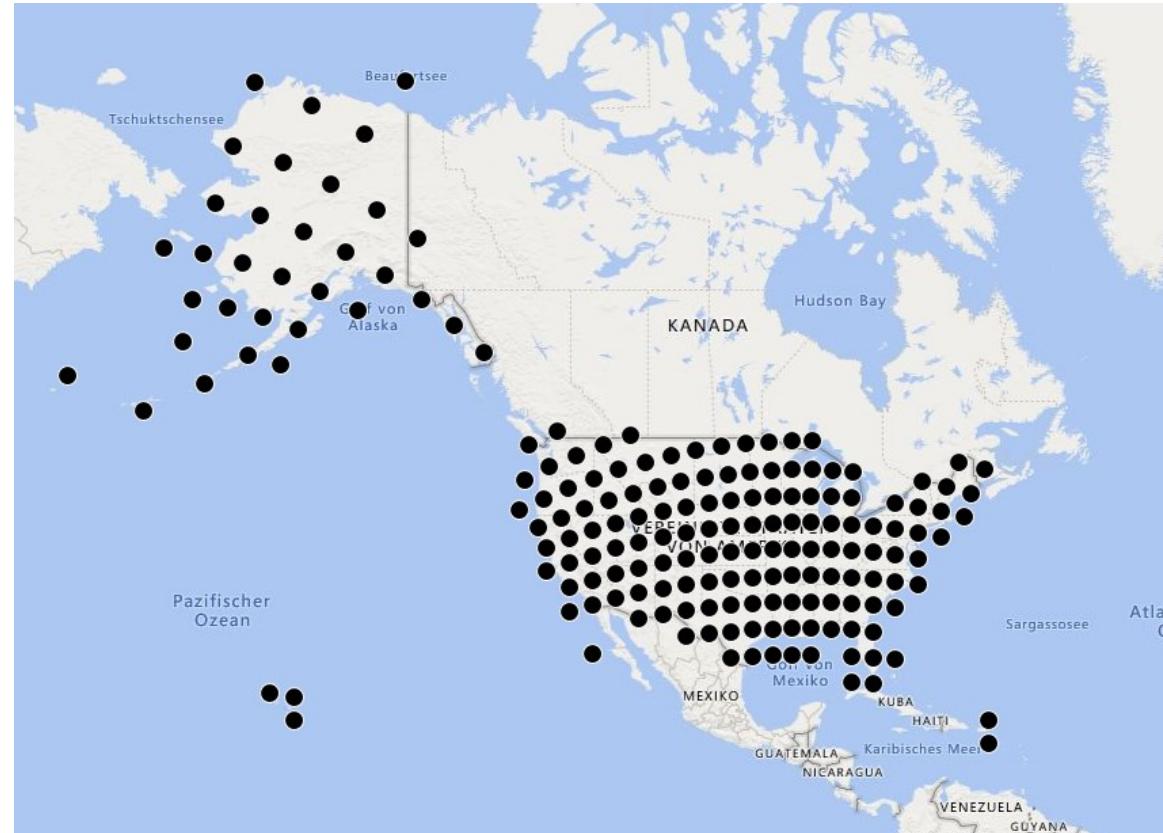
# Geospatial – clustered by geohash

```
GeospatialDemoData()  
| extend geohash = geo_point_to_geohash(lng, lat, 2)  
| summarize count() by geohash  
| extend point = geo_geohash_to_central_point(geohash)  
| project-reorder point, count_
```



# Geospatial – clustered by s2 cells

```
GeospatialDemoData()
| extend s2cell = geo_point_to_s2cell(lng, lat, 5)
| summarize count() by s2cell
| extend point = geo_s2cell_to_central_point(s2cell)
| project-reorder point, count_
```



# Geo-clustering – geo\_point\_to\_geohash()

- Calculates the Geohash string value for a geographic location
- Use it to find points in proximity to each other, based on a common prefix
- Up to level 18

```
print geohash = geo_point_to_geohash  
(-80.195829, 25.802215, 8)
```

geohash
dhwfz15h



Accuracy	Width	Height
1	5000 km	5000 km
2	1250 km	625 km
3	156.25 km	156.25 km
4	39.06 km	19.53 km
5	4.88 km	4.88 km
6	1.22 km	0.61 km
7	152.59 m	152.59 m
8	38.15 m	19.07 m
9	4.77 m	4.77 m
10	1.19 m	0.59 m
11	149.01 mm	149.01 mm
12	37.25 mm	18.63 mm
13	4.66 mm	4.66 mm
14	1.16 mm	0.58 mm
15	145.52 μ	145.52 μ
16	36.28 μ	18.19 μ
17	4.55 μ	4.55 μ
18	1.14 μ	0.57 μ

# Geo-clustering – geo\_geohash\_to\_central\_point()

- Use it to retrieve coordinates of the center of Geohash rectangular area
- Needed for visualization tools to draw a point on a map

```
print point = geo_geohash_to_central_point("sunny")
| extend coordinates = point.coordinates
| extend longitude = coordinates[0]
| extend latitude = coordinates[1]
```

point	coordinates	longitude	latitude
{ "type": "Point", "coordinates": [ 42.47314453125, 23.70849609375 ] }	[ 42.47314453125, 23.70849609375 ]	42,47314453125	23,70849609375



# Geo clustering – geo\_point\_to\_s2cell()

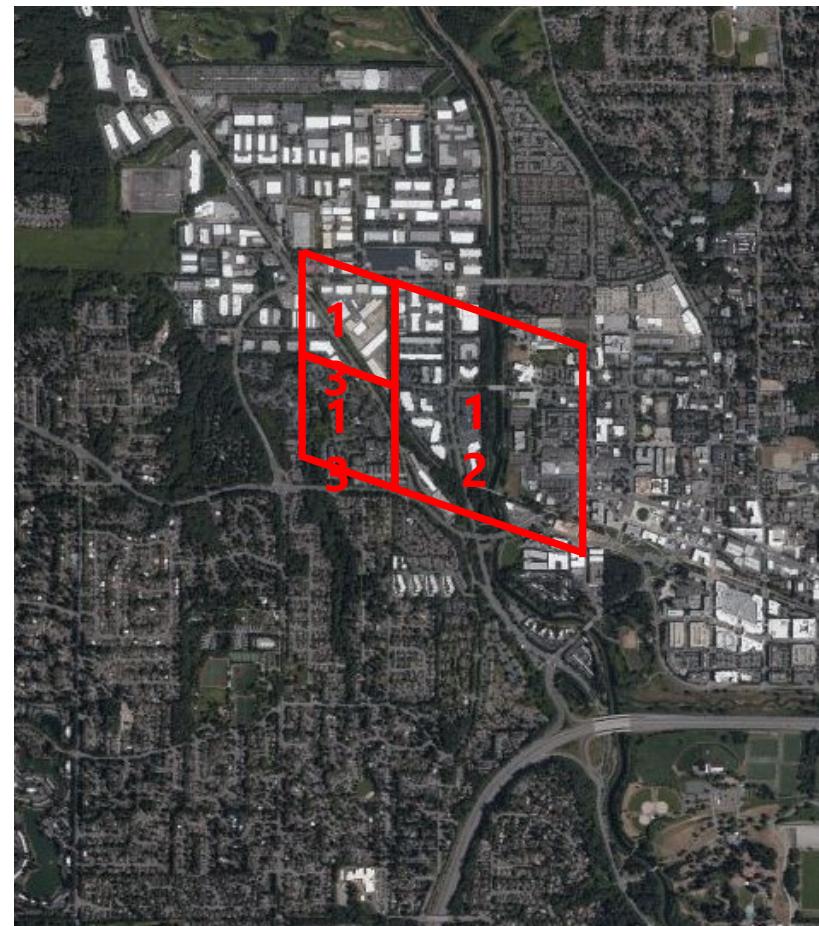
- Calculates the S2 cell token string value for a geographic location.
- Up to level 31

```
print s2token = geo_point_to_s2cell  
(-80.195829, 25.802215, 5)
```

s2token
88dc

## S2 Cell area coverage per level value:

Level	Min area	Max area	Average area	Units
0	85011012.19	85011012.19	85011012.19	km <sup>2</sup>
1	21252753.05	21252753.05	21252753.05	km <sup>2</sup>
2	4919708.23	6026521.16	5313188.26	km <sup>2</sup>
3	1055377.48	1646455.5	1328297.07	km <sup>2</sup>
4	231564.06	413918.15	332074.27	km <sup>2</sup>



# Geo-clustering – geo\_s2cell\_to\_central\_point()

- Calculates the geospatial coordinates that represent the center of S2 cell.
- Needed for visualization tools to draw a point on a map

```
print point = geo_s2cell_to_central_point("88dc")
| extend coordinates = point.coordinates
| extend longitude = coordinates[0]
| extend latitude = coordinates[1]
```

point	coordinates	longitude	latitude
{ "type": "Point", "coordinates": [ -80.810094787025363, 26.873943041714885 ] }	[ -80.810094787025363, 26.873943041714885 ]	-80,8100947870254	26,8739430417149



# Thank you!