Enhancing Cyber Security with Graph Semantics and KQL

YellowHat 2026

-Henning Rauch-

YELLOWHAT

HARD HATS FOR CYBER THREATS

Elite

VirtualMetric

Gold

blu raven

glueckkanja

nedscaper

FalconForce

Enterprise Security Wortell

onevinn

baseVISION

water
IT Security & Defense

PINK

Rubicon

Bronze

KustoWorks SmartSecurity

SiTS | TRAXION

MOSADEX E-HEALTH

valid

INSPARK

yellow arrow

DELTA-N Waarmakers in IT.

ENDOR LABS

yellowhat.live

About Me

Current status: Under the radar

Background:

- Former Principal Product Manager at Microsoft
- Part of Kusto Detective Agency (Prof. Smoke)

- LinkedIn: https://www.linkedin.com/in/henning-rauch-adx/

Graphs at Microsoft with a strong bias towards Kusto

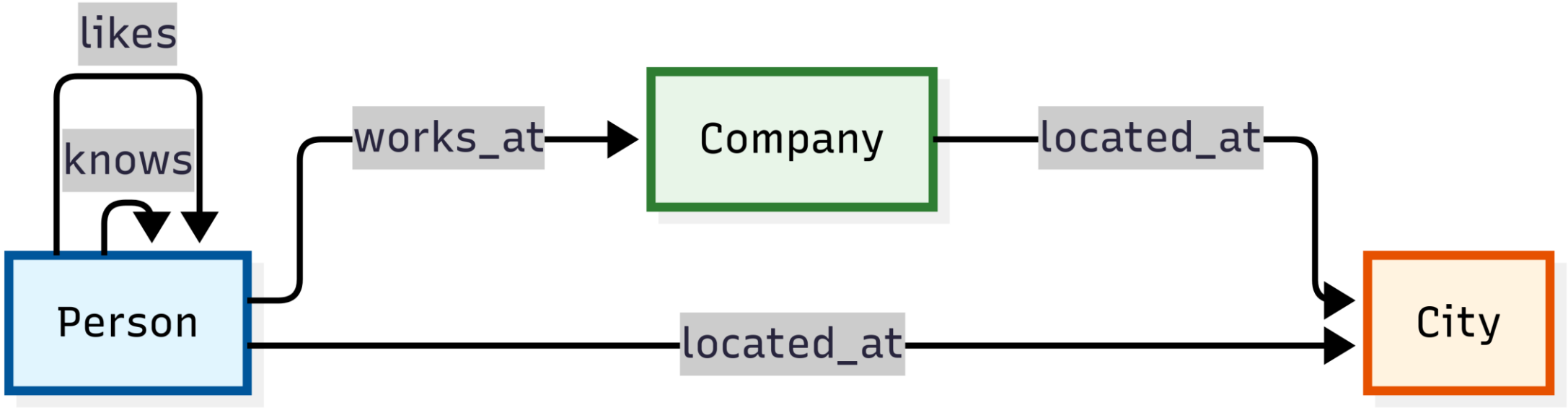
YellowHat 2026

-Henning Rauch-

Agenda

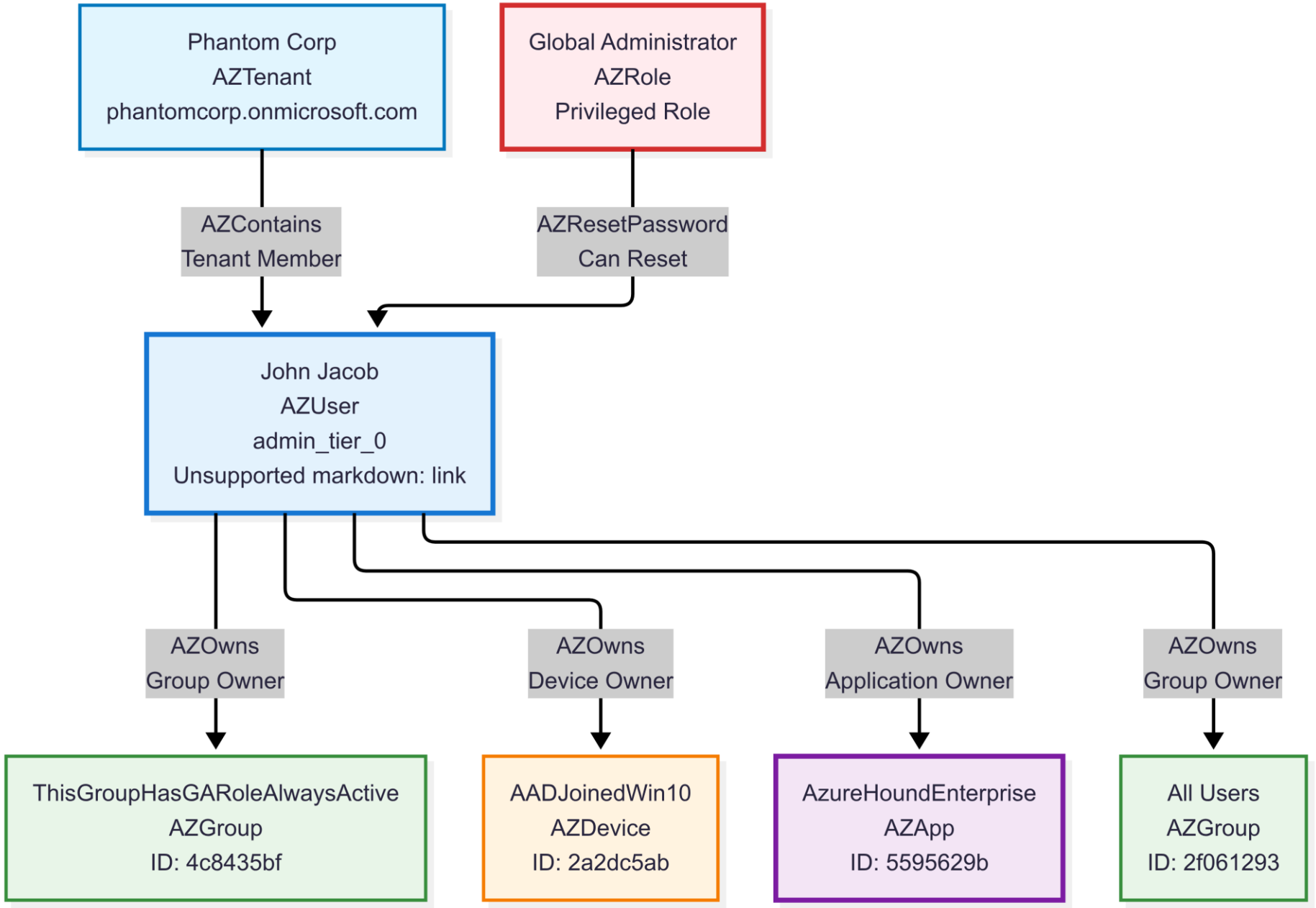
- Intro to graphs
- Graph services from Microsoft
- Deep dive into Kusto
- Demo

Graphs?



https://learn.microsoft.com/en-us/kusto/query/graph-sample-data

Graphs?



https://learn.microsoft.com/en-us/kusto/query/graph-sample-data

Kusto?

Size of Kusto

From Microsoft Digital Defense Report 2025

Our unique vantage point

Microsoft's global presence—spanning billions of users, millions of organizations, and a vast network of partners—provides us with an unparalleled perspective on the cybersecurity threat landscape.

100 trillion	**4.5 million**	**38 million**
100 trillion security signals processed daily	4.5 million net new malware file blocks every day	38 million identity risk detections analyzed in an average day
15,000+	**34,000**	**5 billion**
15,000 partners in our security ecosystem, making it one of the largest in the world	34,000 full-time equivalent security engineers employed worldwide	5 billion emails screened daily on average to protect users from malware and phishing

Microsoft Digital Defense Report 2025 | Microsoft

KustoCon
Learn | Share | Practice

kustocon/KustoCon 2025/Accelerating Innovation with KQL - Henning Rauch at main · KQLCafe/kustocon

Size of Kusto

From Fabric Realtime Intelligence slides…

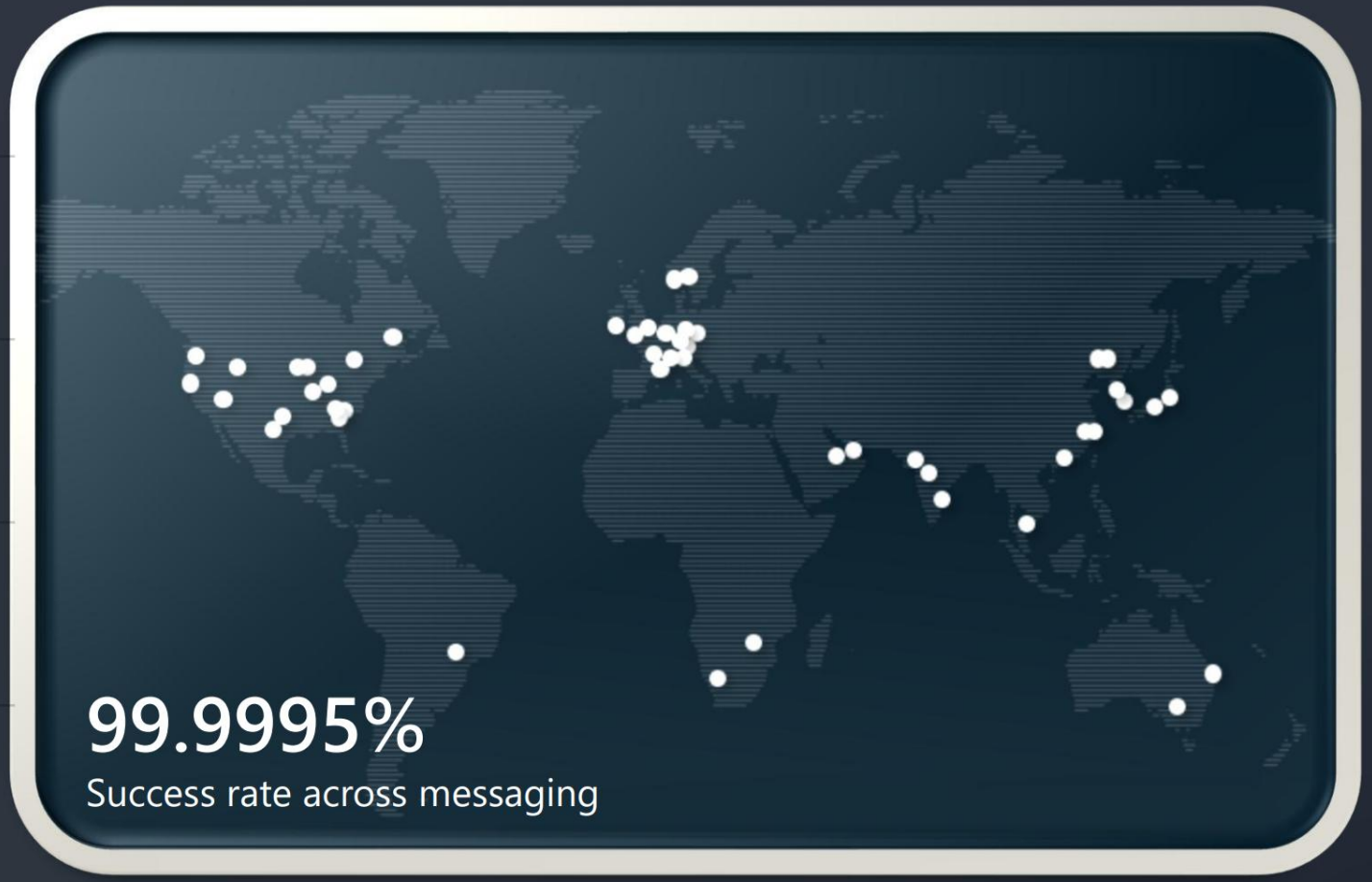
502 T
Total messages per month

33 T
Streaming events processed monthly

12.5 EB
Events and logs per month

470 PB
Ingested daily

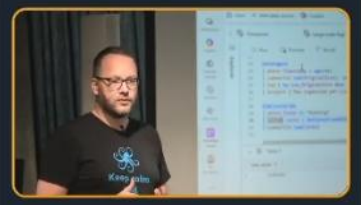
7 B
Real-time queries per day

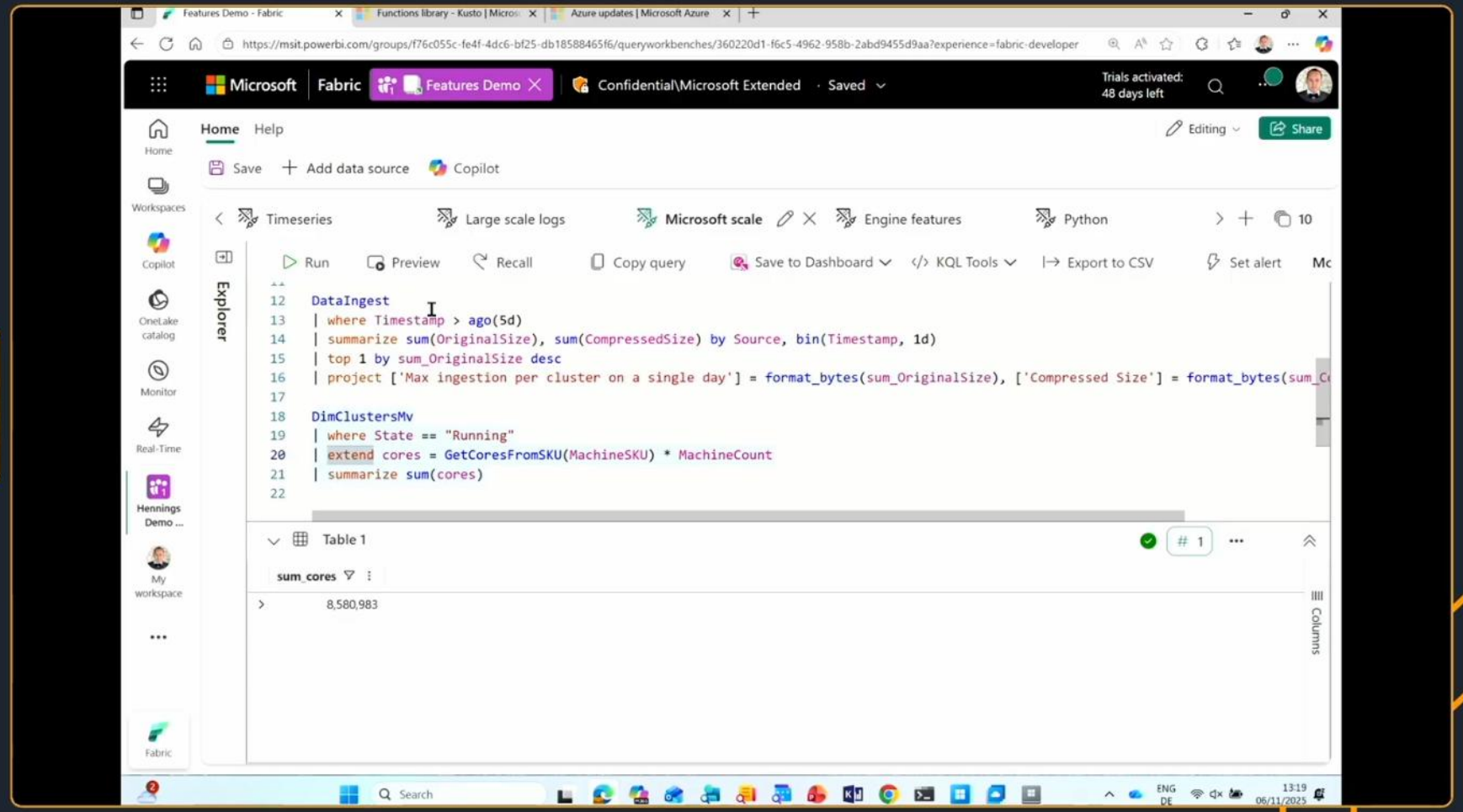


KustoCon
Learn | Share | Practice

kustocon/KustoCon 2025/Accelerating Innovation with KQL - Henning Rauch at main · KQLCafe/kustocon

KustoCon
Learn | Share | Practice





MORE VIDEOS

https://youtu.be/Lc-TgF9Oklg?si=VLBt5TbBrE1E3Z_i

Why Graphs for Cybersecurity?

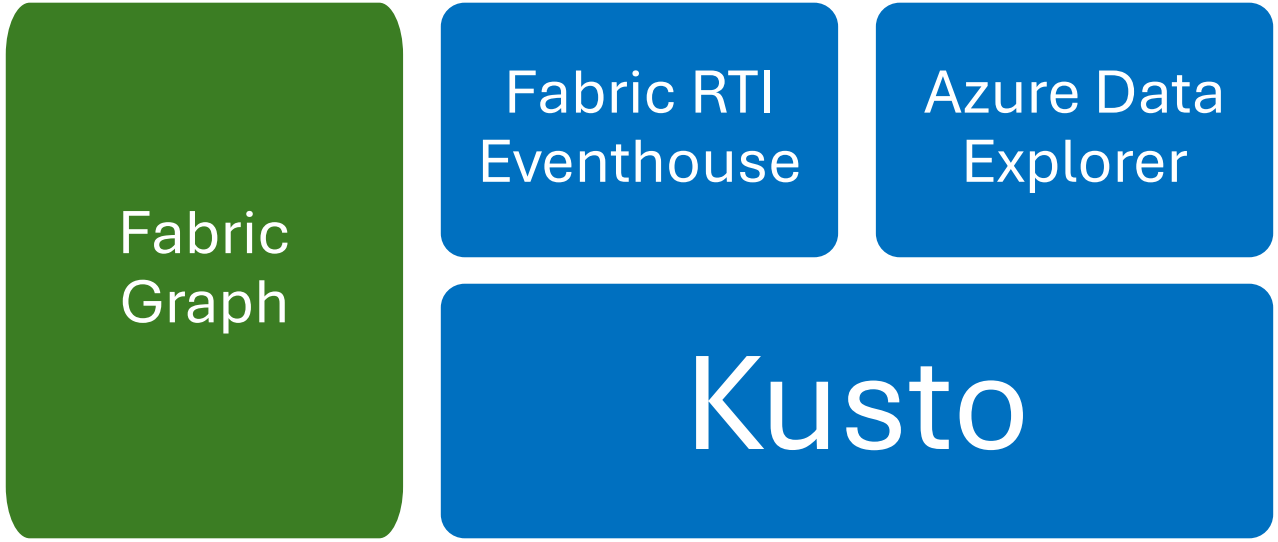
- **The Problem with Tabular-Only Analysis**
  - Tables show events in isolation
  - Joins become complex and expensive at scale
  - Hard to visualize attack narratives

- **What Graphs Reveal**
  - Attack paths across multiple hops
  - Lateral movement chains
  - Blast radius of compromised accounts
  - Hidden dependencies between assets

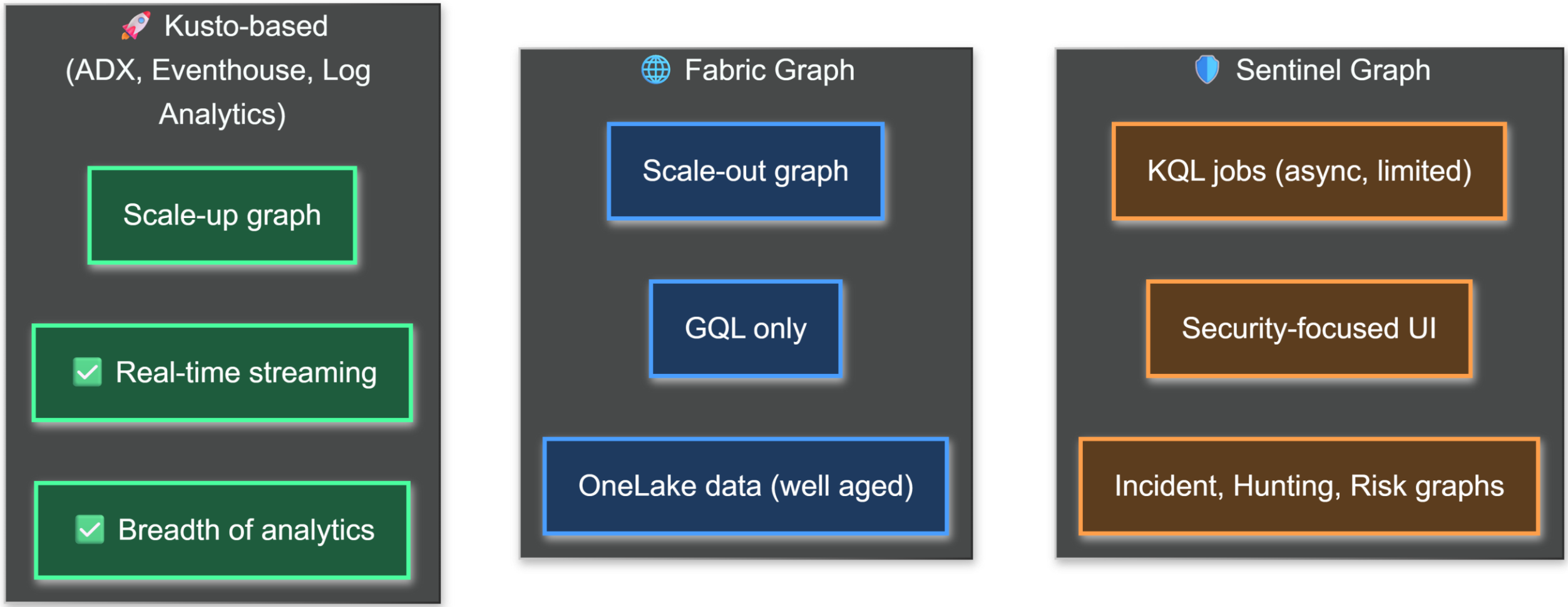
John Lambert | Building Attack Graphs & the Algebras of Defense | Graph the Planet 2025

The Microsoft Graph Technology Landscape

- CosmosDB (Gremlin)
- SQL databases
- Spark
- **Kusto**
  - **Azure Data Explorer**
  - **Eventhouse**
  - **(Log Analytics / Sentinel)**
- Fabric Graph (in preview)
- Microsoft Sentinel Graph

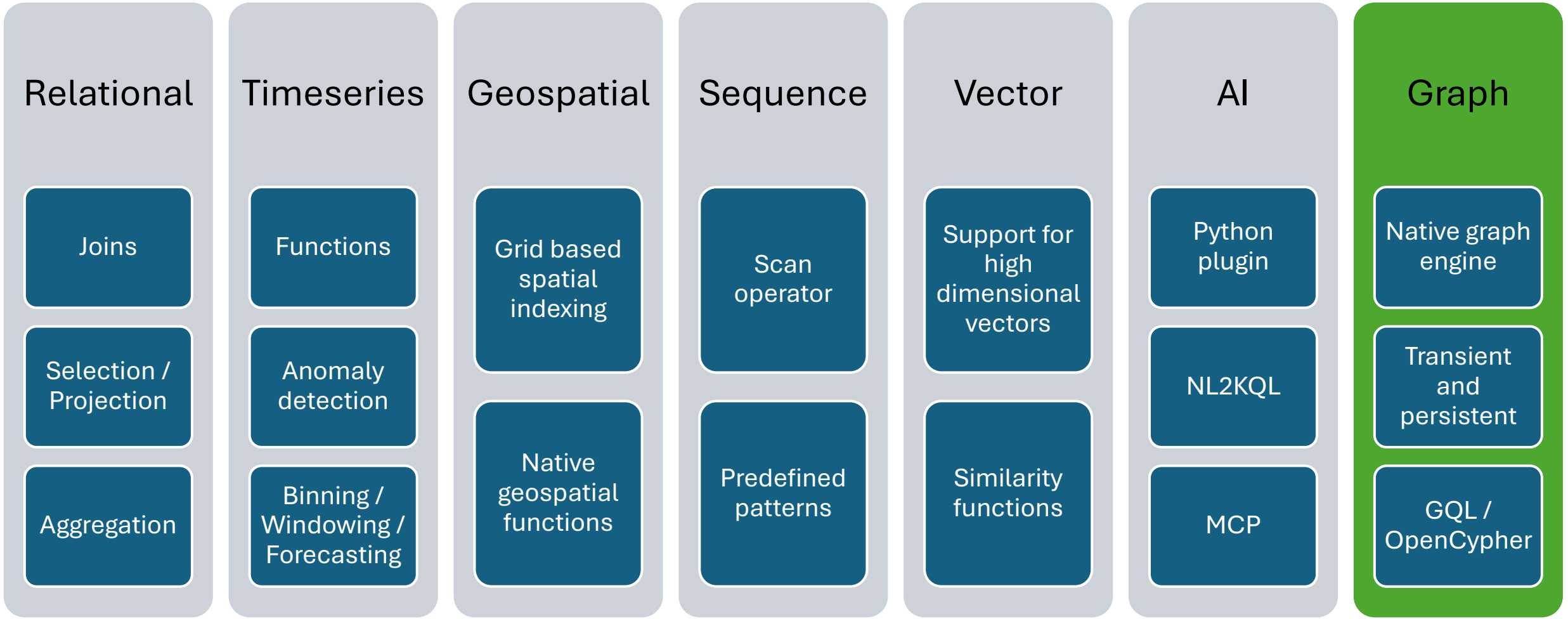


Microsoft's Graph Technologies: Choose Your Path



Details on Kusto

The cybersecurity world runs on KQL - and graph is just one capability among many.



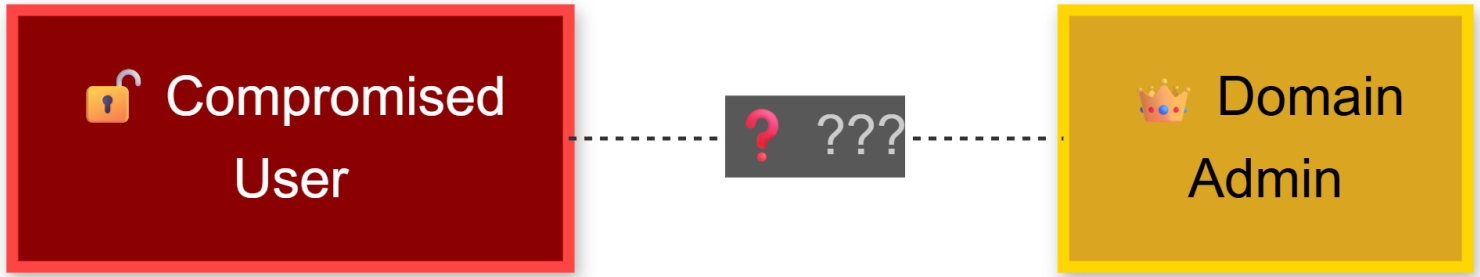
One language. One platform. All your security analytics.

Transient vs persistent graphs

Aspect	Transient	Persistent
Engine	Kusto	Kusto, Fabric Graph, Sentinel Graph
Storage	In-memory	Snapshot based storage
Lifespan	Query duration	Indefinite or until they are retained
Setup	None, Graph model optional	Graph model required
Best for	Exploration, ad-hoc	Repeated queries, dashboards, investigation on a fixed dataset
Scale limit	~10-100M elements (it depends)	Kusto: ~6B elements Fabric Graph: ? Sentinel Graph: ?
Refresh	Every query	snapshot refresh

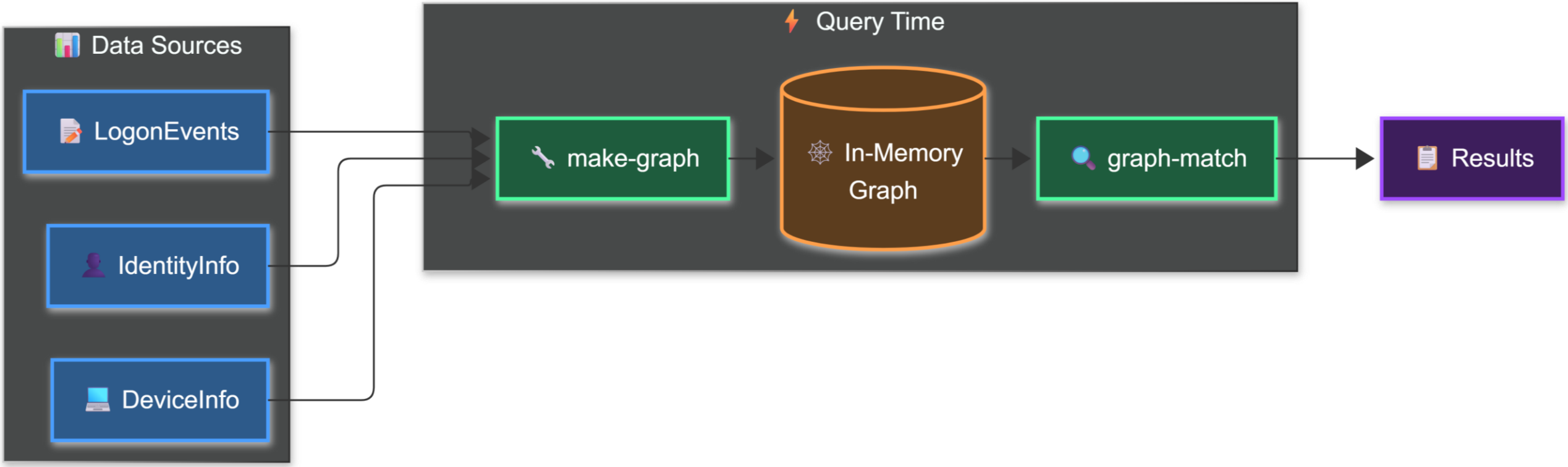
The Lateral Movement Scenario

- **Same problem, multiple solutions**
  - We want to detect attack paths from compromised users to Domain Controllers.
- **Data sources:**
  - IdentityInfo - User accounts and privileges
  - DeviceInfo - Machines and their roles
  - DeviceLogonEvents - Who logged in where, with what rights
- **Graph structure:**
  - Nodes: Identities and Devices
  - Edges: AdminTo, HasSession relationships



Approach 1: Transient Graph (make-graph)

- Graph built inline within the query
- No persistent storage
- Maximum flexibility
- Must rebuild for every execution

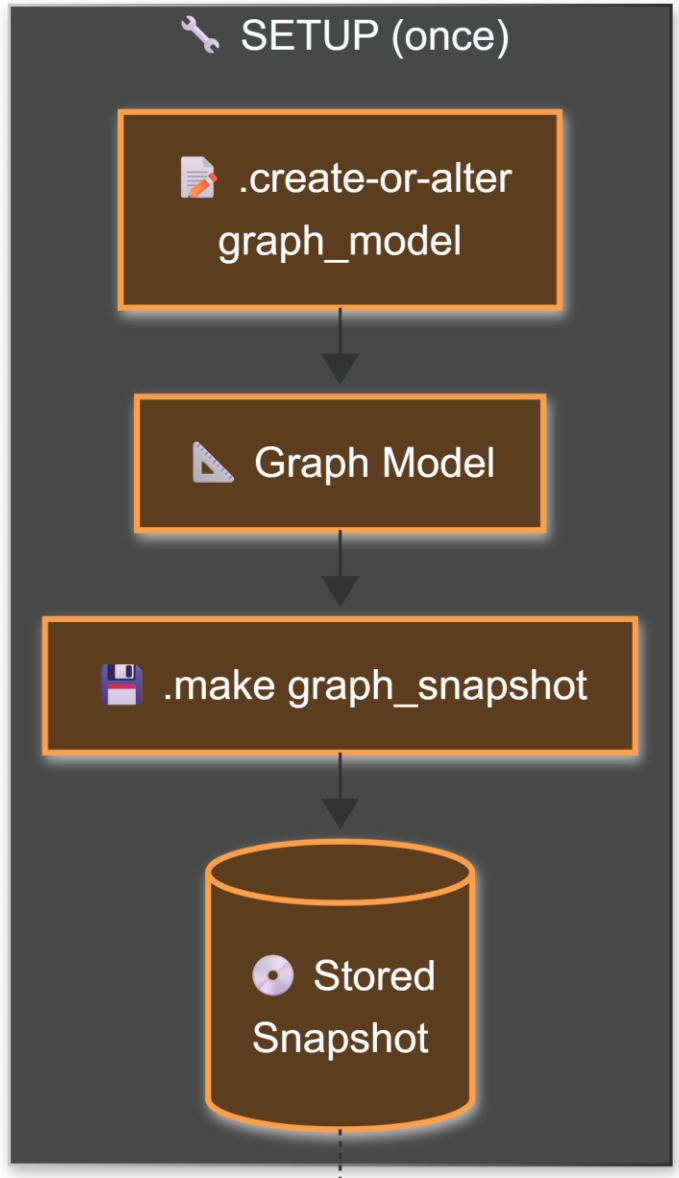


Approach 1: Transient Graph (make-graph)

```
// Transient lateral movement detection
let identities =
    IdentityInfo
    | summarize arg_max(Timestamp, *) by AccountSid
    | project
        NodeId = AccountSid,
        NodeLabel = "Identity",
        AccountName,
        IsAdmin = IsDomainAdmin or IsLocalAdmin;
let devices =
    DeviceInfo
    | summarize arg_max(Timestamp, *) by DeviceId
    | project
        NodeId = DeviceId,
        NodeLabel = "Device",
        DeviceName,
        IsDomainController;
let nodes = union identities, devices;
```

Approach 2: Persistent Graph Model

- Schema and structure defined once
- Snapshots materialize the graph
- Query against pre-built snapshots
- Separates definition from execution



Approach 2: Create the Graph Model

```
.create-or-alter graph_model LateralMovement ```
{
    "Schema": {
        "Nodes": {
            "Identity": {
                "AccountSid": "string",
                "AccountName": "string",
                "IsAdmin": "bool"
            },
            "Device": {
                "DeviceId": "string",
                "DeviceName": "string",
                "IsDomainController": "bool"
            }
```

Approach 2: (continued) Definition Steps

```
"Definition": {
    "Steps": [
        {
            "Kind": "AddNodes",
            "Query": "IdentityInfo | summarize arg_max(Timestamp, *) by AccountSid | project NodeId=AccountSid, AccountName, IsAdmin=IsDomainAdmin or IsLocalAdmin",
            "NodeIdColumn": "NodeId",
            "Labels": ["Identity"]
        },
        {
            "Kind": "AddNodes",
            "Query": "DeviceInfo | summarize arg_max(Timestamp, *) by DeviceId | project NodeId=DeviceId, DeviceName, IsDomainController",
            "NodeIdColumn": "NodeId",
            "Labels": ["Device"]
        },
        {
            "Kind": "AddEdges",
            "Query": "DeviceLogonEvents | where IsLocalAdmin == true | project SourceNode=AccountSid, TargetNode=DeviceId, Timestamp",
            "SourceColumn": "SourceNode",
            "TargetColumn": "TargetNode",
            "Labels": ["AdminTo"]
        },
        {
            "Kind": "AddEdges",
            "Query": "DeviceLogonEvents | where IsLocalAdmin == false | project SourceNode=AccountSid, TargetNode=DeviceId, Timestamp",
            "SourceColumn": "SourceNode",
            "TargetColumn": "TargetNode",
            "Labels": ["HasSession"]
        }
    ]
}
```

Approach 2: Create a Graph Snapshot

```
// Create a snapshot (synchronous - waits for completion)
.make graph_snapshot DailySnapshot_20260109 from LateralMovement
```

Returns:

Name	SnapshotTime	NodesCount	EdgesCount
DailySnapshot_20260109	2026-01-09T06:00:00Z	45,823	1,247,891

```
// For large graphs, use async
.make async graph_snapshot WeeklySnapshot from LateralMovement
```

Approach 2: Query the Persistent Graph

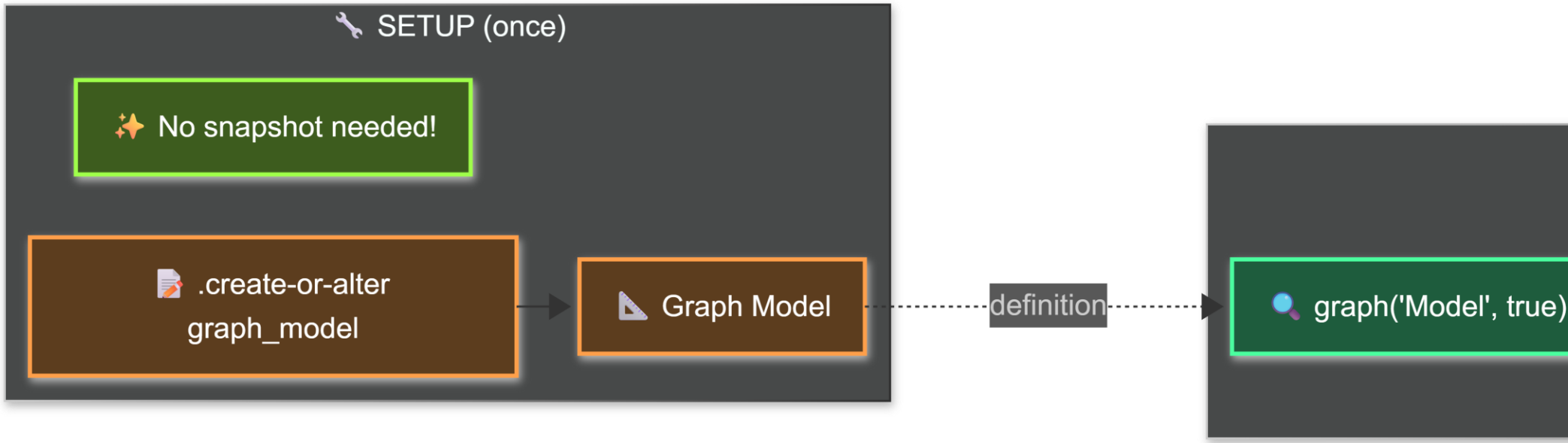
```
// Query the latest snapshot
graph("LateralMovement")
| graph-match (user)-[path*1..5]->(target)
    where labels(user) has "Identity"
      and labels(target) has "Device"
      and user.IsAdmin == false
      and target.IsDomainController == true
    project
        StartUser = user.AccountName,
        PathLength = array_length(path),
        TargetDC = target.DeviceName
| order by PathLength asc
```

Approach 2: Query the Persistent Graph

```
// Query a specific snapshot (for historical comparison)
graph("LateralMovement", "DailySnapshot_20260108")
| graph-match (user)-[path*1..5]->(target)
    where labels(user) has "Identity"
      and labels(target) has "Device"
      and user.IsAdmin == false
      and target.IsDomainController == true
    project StartUser = user.AccountName, PathLength = array_length(path)
```

Approach 3: Transient Graph from Model

- Uses the model's schema and definition
- Builds the graph fresh at query time (like make-graph)
- No snapshot required
- Always uses current data



Approach 3: Transient Graph from Model

```
// Transient graph from model - always uses latest data
graph("LateralMovement", true)
| graph-match (user)-[path*1..5]->(target)
    where labels(user) has "Identity"
      and labels(target) has "Device"
      and user.IsAdmin == false
      and target.IsDomainController == true
    project
        StartUser = user.AccountName,
        PathLength = array_length(path),
        TargetDC = target.DeviceName
| order by PathLength asc
```

Comparing the query syntax

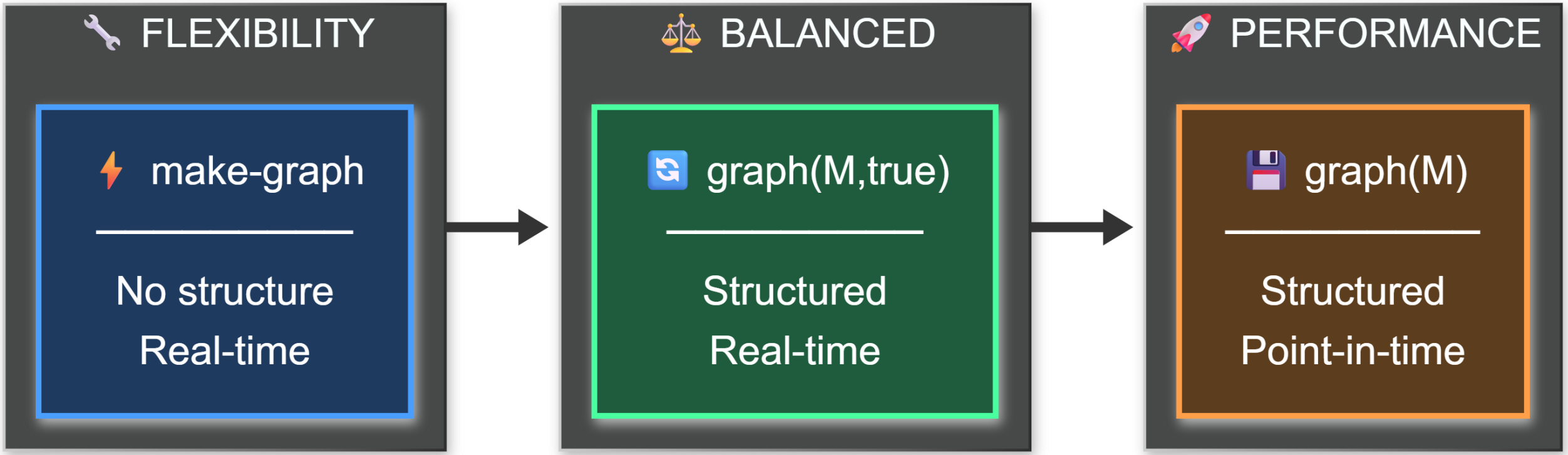
Approach	**Code**
Transient from make-graph	edges | make-graph Src --> Tgt with nodes on Id
Snapshot	graph("Model") or graph("Model", "SnapshotName")
Transient from model	graph("Model", true)

You can run all graph operators (graph-match, graph-to-table) on all type of Kusto graphs regardless of how they have been created.

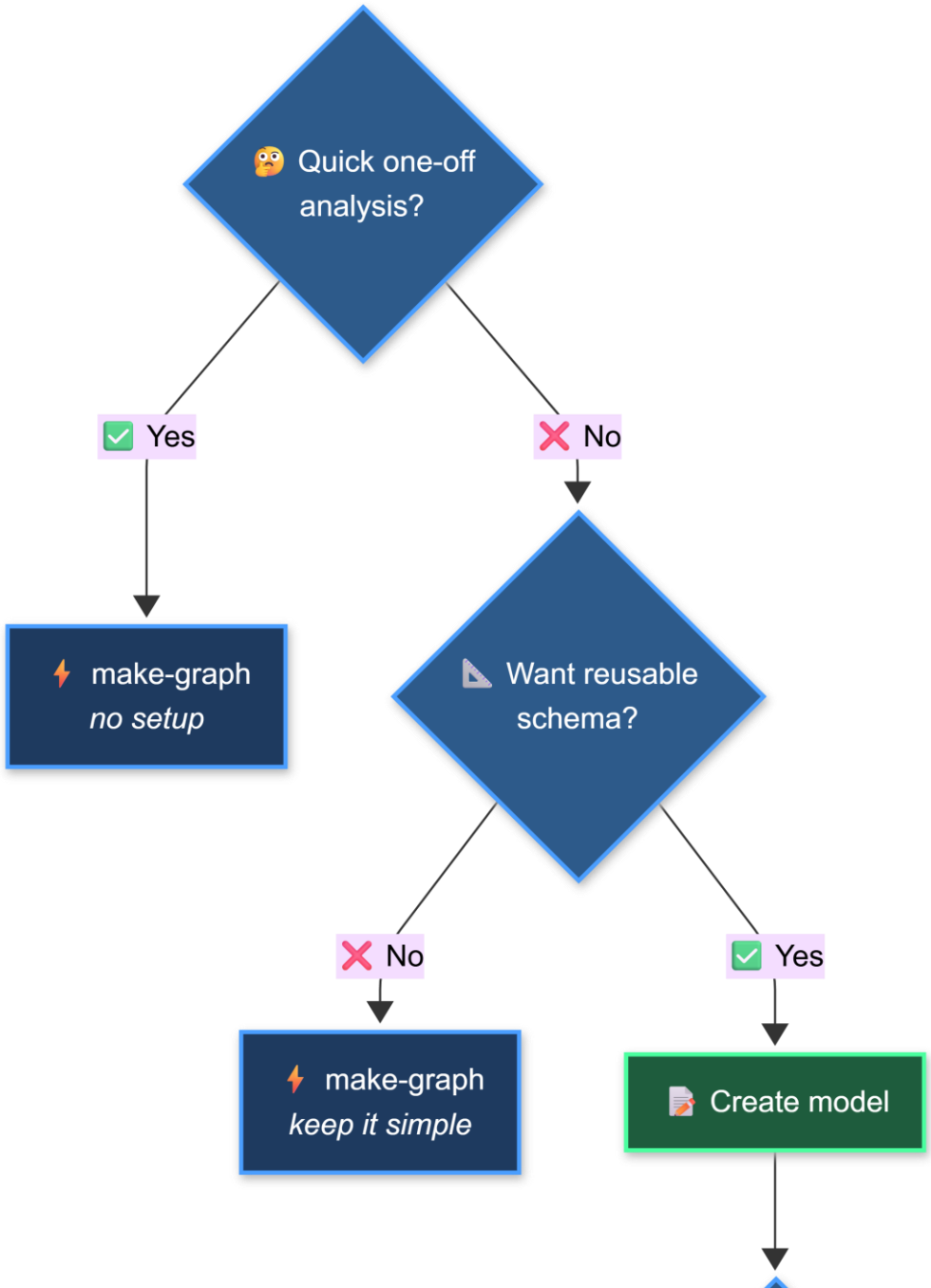
Three approaches compared

Aspect	make-graph	graph(Model, true)	graph(Model)
Setup	None	Create model	Model + snapshots
Schema	Inline	Defined once	Defined once
Data freshness	Real-time	Real-time	Snapshot time
Query performance	Rebuild each time	Rebuild each time	Pre-built
Historical	No	No	Yes
Maintenance	None	Model-only	Model + snapshots
Best for	Ad-hoc exploration	Structured real-time	Production dashboards, Fix investigations

Three approaches compared



Choosing Your Approach: Decision Tree



Cybersecurity Functions

Cybersecurity Functions: Multiple Algebras Required

- Graph functions

Function	Purpose	Question answered
graph_blast_radius_fl()	Downstream impact	"If this is compromised, what's at risk?"
graph_exposure_perimeter_fl()	Upstream exposure	"What could compromise this asset?"
graph_node_centrality_fl()	Critical nodes	"Which nodes are chokepoints?"
graph_path_discovery_fl()	Path enumeration	"How can attacker reach target?"

- Anomaly detection function

Function	Technique	Use Case
detect_anomalous_access_cf_fl()	Collaborative filtering	Unusual user-resource access
detect_anomalous_new_entity_fl()	Poisson distribution	New suspicious entities
detect_anomalous_spike_fl()	Z-score / Q-score	Data exfiltration spikes

Author: **Andrey Karpovsky**

Functions library - Kusto | Microsoft Learn

Demo time

1) **Find attack paths using exposure graph tables**
2) **Combining Algebras: Spike + Blast Radius**
3) **Collaborative filtering anomaly detection + Graph exposure perimeter = Prioritized resource protection**
4) **Graph-RAG for Threat Intelligence**

Key takeaways

- **Multiple Microsoft graph technologies exist**
  - Kusto, Fabric Graph, Sentinel Graph - each with pros and cons
  - Choose based on your scale, real-time needs, and workflow
- **One algebra is not enough for cybersecurity**
  - Graph alone won't protect you
  - Combine graph with statistics, ML, time series analysis
- **KQL: One language, all algebras, one query**
- **Use what Microsoft already built**
  - Cybersecurity functions ready to use

Next Steps

- MCP Registry | Fabric Real-Time Intelligence
  - Go check out MCP server for KQL

- langgraph-kusto · PyPI
  - Kusto-backed storage and checkpointing for LangGraph
  - Enables multi-agent systems with human in the loop

Thank you!

YELLOWHAT

HARD HATS FOR CYBER THREATS

Elite

VirtualMetric

Gold

blu raven

glueckkanja

nedscaper

FalconForce

Enterprise Security Wortell

onevinn

baseVISION

water
IT Security & Defense

PINK

Rubicon

Bronze

KustoWorks SmartSecurity

SiTS | TRAXION

MOSADEX E-HEALTH

valid

INSPARK

yellow arrow

DELTA-N Waarmakers in IT.

ENDOR LABS

yellowhat.live