



# **AI, Cloud & Modern Workplace Conference 2024**

15, 16 & 17 February, Online Conference

<https://aicmwc.azurewebsites.net>

---

## AI, Cloud & Modern Workplace Conference 2024

16, 17 & 18 February, Online Conference



**Michalis Michalos**

Senior Cyber Security Threat Management  
Specialist

## Mastering KQL: Empowering Threat Hunting and Incident Response for Defenders

17 February 2024, 3:00 P.M. (GMT+2)

# whoami

- Currently working as Senior Cyber Security Threat Management Specialist
- 11+ years experience in ICT
- 5 years experience in cybersecurity
- Electrical Engineer BSc, MSc, MBA



- Blog: [michalos.net](https://michalos.net)
- X Handle: [cyb3rmik3](https://twitter.com/cyb3rmik3)
- Github: [github.com/cyb3rmik3](https://github.com/cyb3rmik3)

# Agenda

- Fundamentals of KQL
  - Familiarize searching your Data
  - Presenting Data
  - Combining Data
- Perform Threat Hunting using KQL and Microsoft Security Technologies
- Perform Incident Response using KQL and Microsoft Security Technologies
- Closing remarks
- Resources



# Fundamentals of KQL

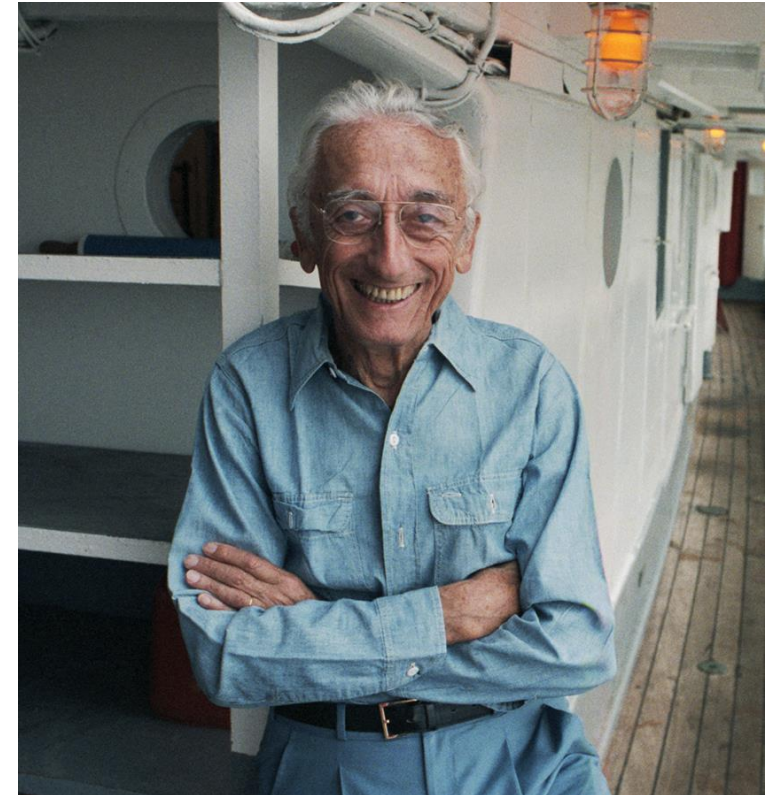


# What does “Kusto” stand for?

The name Kusto comes from **Jacques Cousteau**, a legendary oceanographer, explorer and adventurer.

Azure Application Insights was originally named Kusto and was later the foundation of Azure Monitor.

In recognition of KQL’s ability to deep dive into an ocean of data, Microsoft probably made the right decision to name its query language after Cousteau.



# So what is Kusto?

Kusto Query Language is a read-only request language with requests stated in plain text, using a data-flow model that is easy to read, author, and automate. KQL is a simple yet powerful language to query structured, semi-structured, and unstructured data.

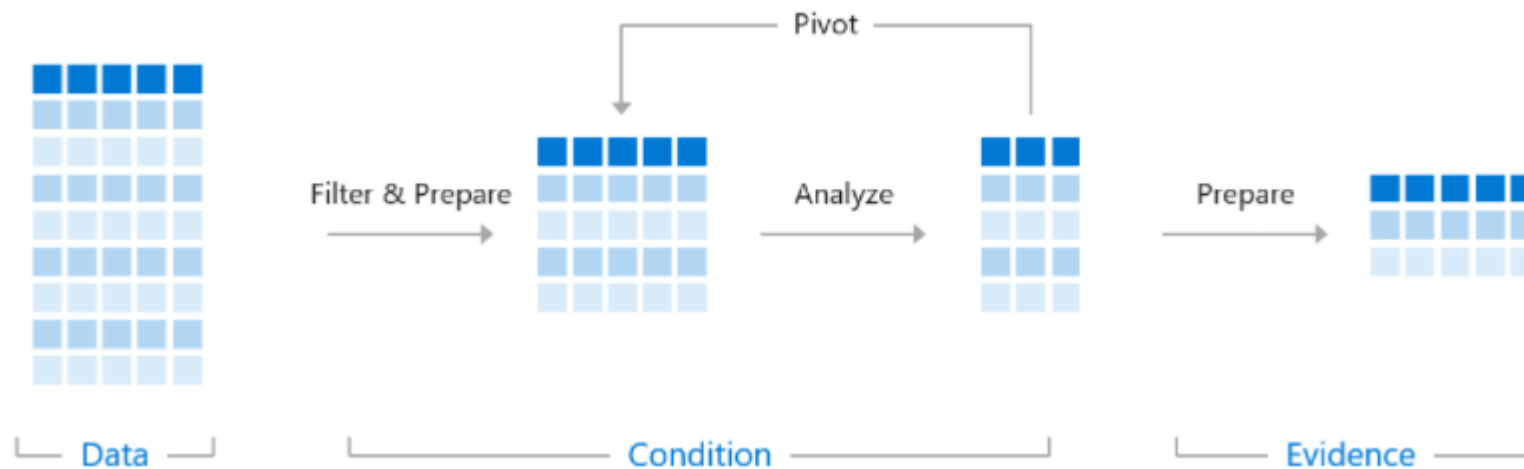
Where can I use KQL?

- Azure Data Explorer (ADX)
  - Azure Monitor Log Analytics
  - Microsoft Sentinel
  - Defender XDR Advanced Hunting
- ... across all of Azure!



# Query operators

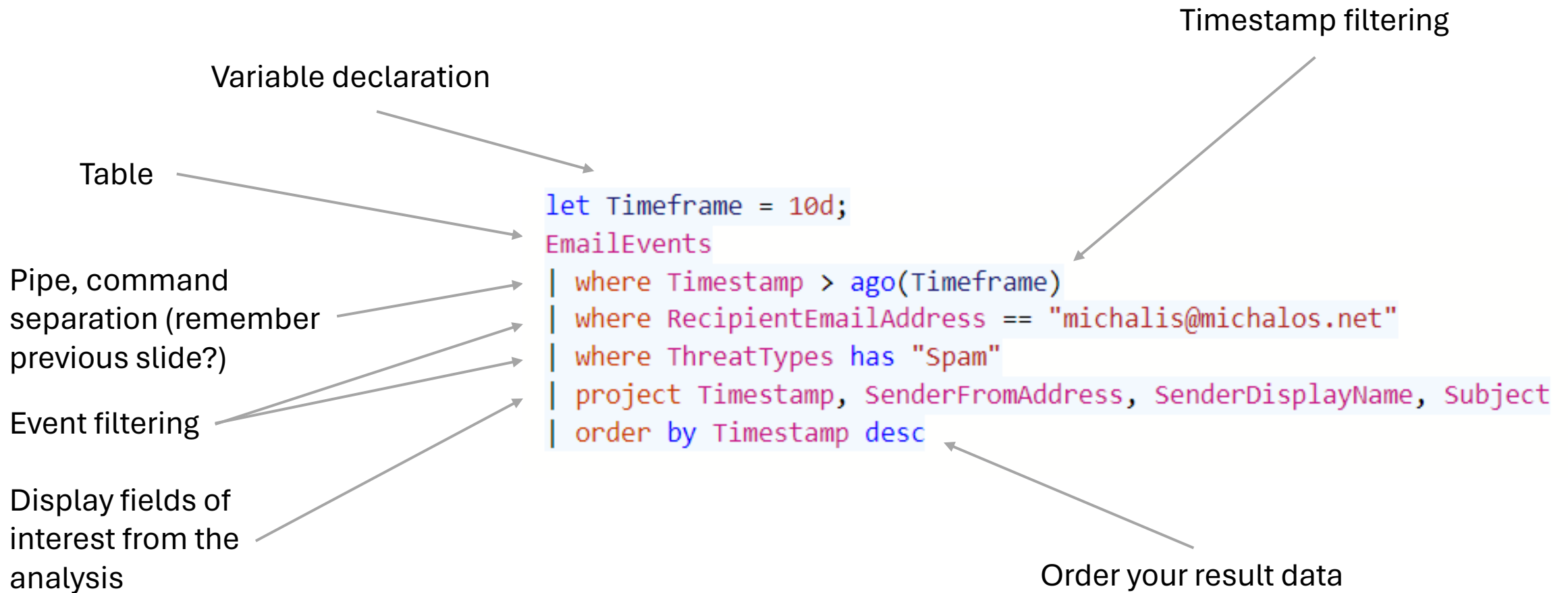
- language syntax is that of a data flow, where "data" means "tabular data" (data in one or more rows/columns rectangular shape)
- at a minimum, a query consists of source data references and one or more query operators applied in sequence
- a pipe character (|) is used to delimit operators



The goal is to filter down data and get the desired results.



# KQL query structure (a basic one...)



# Advanced Hunting blade

The screenshot displays the Microsoft Defender Advanced Hunting interface. On the left, a sidebar contains a 'Schema' section with 'Alerts & behaviors', 'Apps & identities', and 'Email & collaboration' categories, each listing various data sources like 'AlertEvidence' and 'EmailEvents'. A red box labeled 'Schema Community Saved Detections' highlights the top of this sidebar. Another red box labeled 'Tables' highlights the 'Email & collaboration' section. The main area features an 'Options Ribbon' at the top with buttons for 'Run query', 'Set in query', 'Save', 'Share link', and 'Create detection rule'. Below this is the 'Query' window, which contains a KQL script. A red box labeled 'Query window' highlights the script content. At the bottom, the 'Results pane' shows tabs for 'Getting started', 'Results', and 'Query history'. The 'Results' tab is active, displaying a table with columns: 'Timestamp', 'SenderFromAddress', 'SenderDisplayName', and 'Subject'. A red box labeled 'Results pane' highlights this table. The table currently shows '0 items'.

**Advanced hunting**

New query\* +

Schema Functions ...

Search

Alerts & behaviors

- AlertEvidence
- AlertInfo

Apps & identities

- IdentityInfo

Email & collaboration

- EmailAttachmentInfo
- EmailEvents
- EmailPostDeliveryEvents

Options Ribbon

Run query Set in query Save Share link Create detection rule

Query

```
1 let Timeframe = 1d;
2 EmailEvents
3 | where Timestamp > ago(Timeframe)
4 | where RecipientEmailAddress == "michalis@michalos.net"
5 | where ThreatTypes has "Spam"
6 | project Timestamp, SenderFromAddress, SenderDisplayName, Subject
7 | order by Timestamp desc
```

Query window

Getting started Results Query history Results and History

Export 0 items Search 00:00.261 Low Chart type Customize columns

Timestamp	SenderFromAddress	SenderDisplayName	Subject
-----------	-------------------	-------------------	---------

Results pane

# Before we begin...

Make sure, you use in-line commenting:

- It helps documenting
- It helps other people understand what your approach is
- Give instructions to define variables (for example time)
- You can use during experimenting with your operators/variables (quickly add/remove lines without completely losing them)

Use // to indicate a comment or exclusion.

```
let Timeframe = 1d; // Define your desired timeframe
EmailEvents
| where Timestamp > ago(Timeframe)
| where RecipientEmailAddress == "michalis@michalos.net"
//| where ThreatTypes has "Spam"
| project Timestamp, SenderFromAddress, SenderDisplayName, Subject
| order by Timestamp desc
```

# Familiarize searching your Data

# getschema operator

This operator will provide the data types for the table of interest.

Syntax:            Table | getschema

Example:           EmailEvents | getschema

Results:

	ColumnName	ColumnOrdinal	DataType	ColumnType
>	Timestamp	0	System.DateTime	datetime
>	NetworkMessageld	1	System.String	string

# search operator

Straightforward and interactive use.

Syntax:           Table | search

Example:          EmailEvents | search "keyword"

- If no table is defined, search will be performed in all tables.
- Better filter out, to save time.
- Helps answer: “Does it exist?”, “Where does it exist?”, “Why does it exist?”
- In general, it’s not recommended, as it is CPU intensive



# Data filtering

## **where** operator

Filters a table to the subset of rows that satisfy a predicate.

Syntax:                      Table | where Predicate  
Example:                      EmailEvents | where  
RecipientEmailAddress == "michalis@michalos.net"

Predicate satisfaction supported through searching string data types, using various query operators.

- has, contains, startswith, endswith...
- ==, !=, <, >, <= ...
- in, !in, has\_all, has\_any...
- isempty (), notempty (), isnull (), notnull

## **count** operator

Returns the number of records

Syntax:                      Table | count  
Example:                      EmailEvents | count

## **sort** operator

Sorts the rows of the input table into order by one or more columns.

Syntax:                      Table | sort by Column  
[asc | desc]  
Example:                      EmailEvents | sort by  
Timestamp desc

# Results filtering

**project** operator

## Selecting which Columns to present.

Syntax: `Table | project ColumnName`

Example: EmailEvents | project Timestamp, SenderDisplayName, Subject

## distinct operator

Produces a table with the distinct combination of the provided columns of the input table.

Syntax:                      Table | distinct

Example: EmailEvents | distinct SenderFromAddress, EmailAction

# Filtering searches

## **contains** filtering

Filters a record set for data containing a case-insensitive string. contains searches for arbitrary sub-strings rather than terms.

Syntax: Table | where Predicate contains "phrase"

Example: EmailEvents | where SenderFromAddress contains "microsoft"

Usually, a resource consuming operation.

## **has** operator

Filters a record set for data with a case-insensitive string. has searches for indexed terms, where an indexed term is three or more characters.

Syntax: Table | where Predicate has "phrase"

Example: EmailEvents | where SenderFromAddress has "microsoft"

```
1 EmailEvents
2 | where SenderFromAddress contains "microsoft"
3 | summarize count() by SenderFromAddress
```

Getting started	Results	Query history
↓ Export		5 ite
<input type="checkbox"/>	SenderFromAddress	count_
<input type="checkbox"/>	> o365mc@microsoft.com	8
<input type="checkbox"/>	> defender-noreply@microsoft.com	5
<input type="checkbox"/>	> microsoft-noreply@microsoft.com	11
<input type="checkbox"/>	> azure-noreply@microsoft.com	1
<input type="checkbox"/>	> [REDACTED]@[REDACTED]onmicrosoft.com	2

```
1 EmailEvents
2 | where SenderFromAddress has "microsoft"
3 | summarize count() by SenderFromAddress
```

Getting started	Results	Query history
↓ Export		
<input type="checkbox"/>	SenderFromAddress	count_
<input type="checkbox"/>	> o365mc@microsoft.com	8
<input type="checkbox"/>	> defender-noreply@microsoft.com	5
<input type="checkbox"/>	> microsoft-noreply@microsoft.com	11
<input type="checkbox"/>	> azure-noreply@microsoft.com	1

# Presenting Data

# extend operator





Creates calculated columns and append them to the result set.

Syntax:            Table | extend [ColumnName |  
                      (ColumnName[, ...]) =]  
                      Expression [, ...]

Example:           DeviceInfo | extend  
                      DeviceNameLength =  
                      strlen(DeviceName)

- The new column, is not indexed.
- To only change a column name, use project-rename operator

```
1 DeviceInfo
2 // Calculate the DeviceName length
3 | extend DeviceNameLength = strlen(DeviceName)
4 | project DeviceName, DeviceNameLength
```

Getting started <u>Results</u> Query history		
<a href="#">↓ Export</a> <span>226</span>		
<input type="checkbox"/>	DeviceName	DeviceNameLength
<input type="checkbox"/>	>  mm-gl-	14
<input type="checkbox"/>	>  mm-gl-	14
<input type="checkbox"/>	>  desktop	15
<input type="checkbox"/>	>  mm-gl-	14

# summarize operator

Produces a table that aggregates the content of the input table.

Syntax:            Table | summarize [  
                     SummarizeParameters ]  
                     [[Column =] Aggregation [, ...]]  
                     [by [Column =]

Example:           EmailEvents | summarize by  
                     SenderIPv4,  
                     SenderFromAddress

- Can be used with aggregation functions count(), sum(), min(), max() etc.

```
UrlClickEvents  
// Count how many times, a Url identified as phishing  
// has been clicked  
| where ThreatTypes has "Phish"  
| summarize count() by Url
```



# summarize operator (bin and time)

Summarize by creating a time series

- Change time values per specific needs, 1h, 1d etc.
- Could be useful to detect anomalies

```
1 EmailEvents
2 | summarize count() by bin(Timestamp, 1d)
3 | order by Timestamp asc
```

Getting started			Results	Query history	
			↓ Export		
<input type="checkbox"/>	Timestamp		count_		
<input type="checkbox"/>	>	Feb 3, 2024 2:00:0...	6		
<input type="checkbox"/>	>	Feb 4, 2024 2:00:0...	3		
<input type="checkbox"/>	>	Feb 5, 2024 2:00:0...	4		
<input type="checkbox"/>	>	Feb 6, 2024 2:00:0...	30		
<input type="checkbox"/>	>	Feb 7, 2024 2:00:0...	11		
<input type="checkbox"/>	>	Feb 8, 2024 2:00:0...	18		
<input type="checkbox"/>	>	Feb 9, 2024 2:00:0...	4		
<input type="checkbox"/>	>	Feb 10, 2024 2:00:...	5		

# render operator

Generates a visualization of the query results.

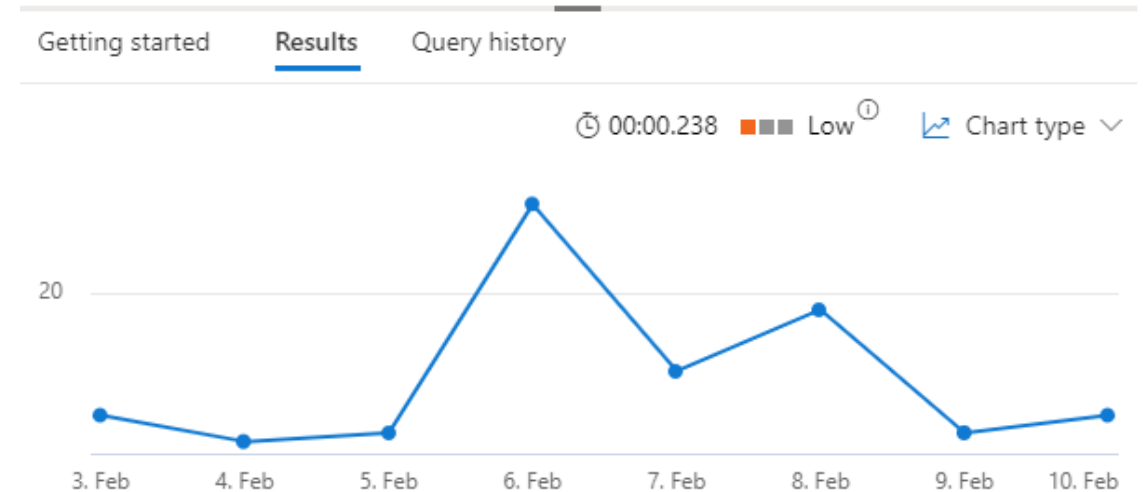
Syntax: `Table | render visualization [with ( propertyName = propertyValue [, ...])]`

Supported visualizations:

- anomalychart
- areachart
- barchart
- card
- columnchart
- ladderchart
- linechart
- piechart
- pivotchart
- scatterchart
- stackedareachart
- table
- timechart
- timepivot
- treemap

Not all visualizations are supported across platforms (log analytics, advanced hunting, data explorer)

```
1 EmailEvents
2 | summarize count() by bin(Timestamp, 1d)
3 | render timechart
```



# let statement

A let statement is used to set a variable name equal to an expression or a function, or to create views.

- Declaring variables to ease complexity
- Used for allow, deny, and exclusion lists
- Build a temporary container for a table
- Can be used multiple times in query
- Useful for targeted threat hunting

```
1 // Hunt for Remote Monitoring and Management (RMM)
2 // executables based on filenames
3 let SuspiciousRMMExecutables = datatable (rmmexecutable: string) [
4     @"anydesk.exe"
5     @"teamviewer.exe"
6 ];
7 DeviceProcessEvents
8 | where FileName in (SuspiciousRMMExecutables)
```

# Combining Data

# union operator

Takes two or more tables and returns the rows of all of them.

Syntax:            [ Table | ] union [ UnionParameters ]  
                     [kind=inner|outer] [withsource=  
                     ColumnName] [isfuzzy= true|false]  
                     Tables

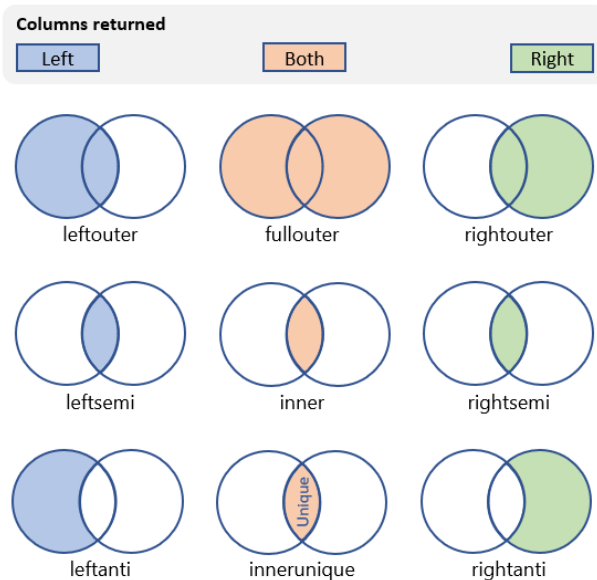
```
1 // Query will return any FileName from DeviceFileEvents and
2 // DeviceProcessEvents tables containing "anydesk" keyword
3 union DeviceFileEvents, DeviceProcessEvents
4 | where FileName contains @"anydesk"
```

- kind=inner(common columns), outer (all columns default)
- Supports wildcard to union multiple tables (union Table\*)
- Can union between tables from different clusters (or workspaces)

# join operator

Merge the rows of two tables to form a new table by matching values of the specified columns from each table.

Syntax: LeftTable | join [ kind = JoinFlavor ] [ Hints ] (RightTable) on Conditions



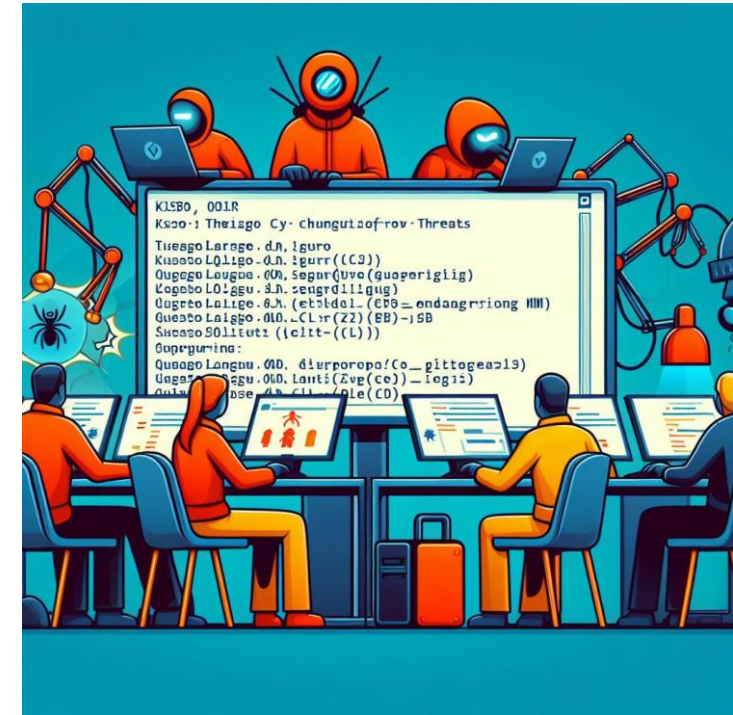
```
1 // Enrich URL click events from emails that don't belong to
2 // ClickAllowed, from the EmailEvents Table
3 EmailEvents
4 | join (UrlClickEvents
5 | where ActionType != "ClickAllowed") on NetworkMessageId
```



# wrapping up

- Operators, functions and methods presented, are the most common in KQL to begin with.
- By now, you should be able to do some basic filtering and present and combine data.
- You are ready to begin your journey in KQL!

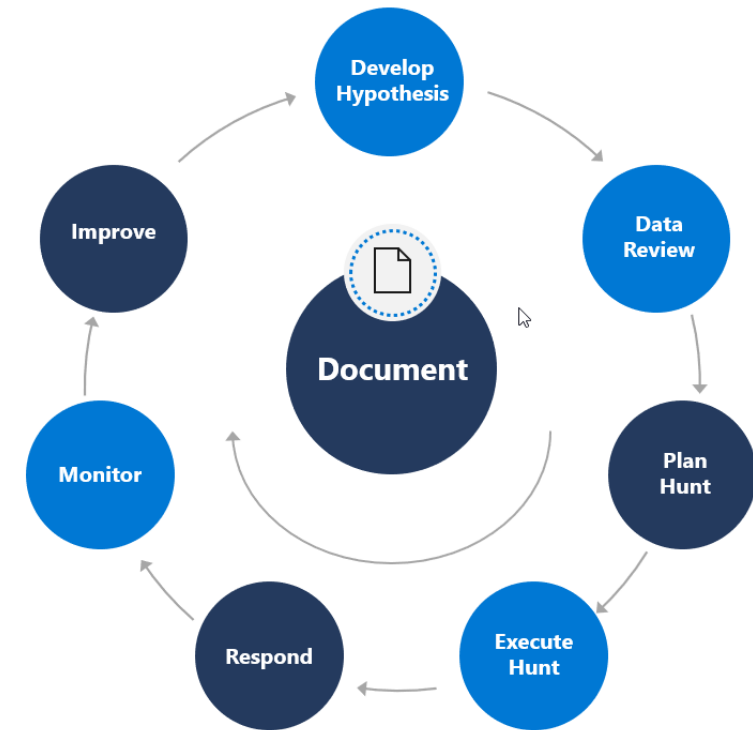
```
1 let Timeframe = 30d; // Choose the best timeframe for your investigation
2 let SuspiciousAnydeskFileCertificate = DeviceFileCertificateInfo
3     | where Timestamp > ago(Timeframe)
4     | where CertificateSerialNumber =~ "0dbf152deaf0b981a8a938d53f769db8" // Compromised Certificate Serial Number
5     | where Issuer == "DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1"
6     | project Timestamp, DeviceName, SHA1;
7 SuspiciousAnydeskFileCertificate
8     | join (DeviceProcessEvents
9         | where Timestamp > ago(Timeframe)
10        | where ProcessVersionInfoCompanyName !contains @"AnyDesk"
11        | project SHA1, ActionType, FileName, FolderPath, ProcessVersionInfoCompanyName, ProcessVersionInfoProductName,
12        ProcessCommandLine, AccountName, InitiatingProcessAccountName, InitiatingProcessFileName, InitiatingProcessCommandLine
13        )on SHA1
14     | sort by Timestamp desc
```



# Perform Threat Hunting using KQL and Microsoft Security Technologies

# Threat Hunting basics

- proactively hunting for a threat or a set of activities that haven't been previously detected
- a continual process
- documentation for the hunt should include:
  - What, How, and Why
  - Input and Output
  - How to replicate the hunt
  - Next Steps
- Hypothesis development:
  - Keep it achievable
  - Keep the scope narrow
  - Keep it time-bound
  - Keep it useful and efficient
  - Keep it related to the threat model that you are defending against



# Advanced Hunting in Defender XDR

- Perform quick and simple queries
- Can be saved under Queries > My queries for future use
- No reference, comments or any sort of contextualization

The screenshot displays the 'Advanced hunting' interface in Microsoft Defender XDR. On the left, a sidebar shows a tree view of query folders: 'Shared queries' (containing 'Suggested'), 'My queries' (with a note to save queries for quick access), and 'Community queries' (containing various threat categories like Campaigns, Collection, etc.). The main area shows a KQL query for finding recent attachments to a compromised email address. Below the query editor, there are tabs for 'Getting started', 'Results', and 'Query history'. The 'Results' tab is active, showing an 'Export' button and a table with columns: 'Timestamp', 'NetworkMessageId', 'SenderMailFromAddress', and 'SenderFrom'. A 'Save query' dialog is open on the right, allowing the user to name the query ('Review recent attachments'), choose a location ('My queries\BEC'), and preview the KQL code.

```
1 let CompromizedEmailAddress = ""; // Insert the email address of the compromised email
2 let Timeframe = 2d; // Choose the best timeframe for your investigation
3 let EmailInformation = EmailEvents
4   | where RecipientEmailAddress == CompromizedEmailAddress
5   | where Timestamp > ago(Timeframe)
6   | where DeliveryAction != "Blocked"
7   | where AttachmentCount != "0"
8   | project Timestamp, NetworkMessageId, SenderMailFromAddress, SenderFrom
9   | join (EmailAttachmentInfo
10    | project NetworkMessageId, FileName, FileType, FileSize
11    ) on NetworkMessageId
12   | sort by Timestamp desc
```

**Save query**

Name \*  
Review recent attachments

Location \*  
My queries\BEC

My queries  
New folder  
BEC

Shared queries  
where Timestamp > ago(Timeframe)  
where DeliveryAction != "Blocked"  
where AttachmentCount != "0"  
project Timestamp, NetworkMessageId, SenderMailFromAddress, SenderFrom  
join (EmailAttachmentInfo  
project NetworkMessageId, FileName, FileType, FileSize  
) on NetworkMessageId  
sort by Timestamp desc

# Hunting in Microsoft Sentinel

- Use build-in queries, or create new ones
- Operationalize MITRE ATT&CK framework
- Use livestream to actively hunt with your queries
- Contextualize your hunts, hypotheses and process

Microsoft Sentinel | Hunting

Selected workspace: 'mm-sentinel-prod'

Search Refresh Last 24 hours New query Run all queries Delete Hunt actions Columns Guides & Feedback

General

- Overview (Preview)
- Logs
- News & guides
- Search

Threat management

- Incidents
- Workbooks
- Hunting
- Notebooks
- Entity behavior
- Threat intelligence
- MITRE ATT&CK (Preview)

Content management

- Content hub
- Repositories (Preview)
- Community

Configuration

- Workspace manager (Preview)
- Data connectors
- Analytics
- Watchlist
- Automation

11 / 18 Active / total queries

0 / 0 Result count / queries run

0 Livestream Results

0 My bookmarks

More content at Content hub

Hunts (Preview) Queries Livestream Bookmarks

0 Reconnaissance 0 Resource Devel... 5 Initial Access 1 Execution 1 Persistence 1 Privilege Escala... 0 Defense Evasion 2 Credential Acce... 1 Discovery 0 Collection 1 Command And ... 0 Exfiltration 8 Impact 2 Lateral Movem...

Search queries Add filter

	Query	Results	Results delta	Results delta percen...	Content source	Data sources	Tactics	Techniques
<input type="checkbox"/>	Azure Network Security Group NSG Administrative Operations	--	--	--	Content hub	AzureActivity	Impact	T1496
<input type="checkbox"/>	Port opened for an Azure Resource	--	--	--	Content hub	AzureActivity		+3
<input type="checkbox"/>	Determine Successfully Delivered Phishing Emails by top IP A...	--	--	--	Content hub		Initial Access	T1566
<input type="checkbox"/>	Microsoft Sentinel Analytics Rules Administrative Operations	--	--	--	Content hub	AzureActivity	Impact	T1496
<input type="checkbox"/>	Anomalous Azure Operation Hunting Model	--	--	--	Content hub			+2
<input type="checkbox"/>	Rare Custom Script Extension	--	--	--	Content hub	AzureActivity	Execution	T1059
<input type="checkbox"/>	Azure Virtual Network Subnets Administrative Operations	--	--	--	Content hub	AzureActivity	Impact	T1496
<input type="checkbox"/>	Determine Successfully Delivered Phishing Emails to Inbox/Ju...	--	--	--	Content hub		Initial Access	T1566
<input type="checkbox"/>	Appspot Phishing Abuse	--	--	--	Content hub		Initial Access	T1566
<input type="checkbox"/>	Azure VM Run Command executed from Azure IP address	--	--	--	Content hub			+2
<input type="checkbox"/>	Granting permissions to account	--	--	--	Content hub	AzureActivity		T1098
<input type="checkbox"/>	Spoofing attempts from Specific Domains	--	--	--	Content hub		Initial Access	T1566

< Previous Page 1 of 1 Next > Showing 1 to 18 of 18 results.

# Hunting in Microsoft Sentinel

- Add description to your queries for better documentation
- Map your entities such as Users, Hosts and IPs
- Map MITRE ATT&CK TTPs

Home > Microsoft Sentinel > Microsoft Sentinel | Hunting >

### Create hunting query

Do not use fixed time ranges, either directly or in a function, in your query. Otherwise, we cannot show changes in query results over time.

Name \*

Potential Adversary in the middle Phishing

Description

List potential adversary in the middle phishing attempts that have been identified by the Office-Home application in combination with an empty deviceId.

Query \*

```
SignInLogs
| where AppDisplayName == "OfficeHome"
| where UserPrincipalName has "g"
| extend deviceId = tostring(DeviceDetail.deviceId), displayName = tostring(DeviceDetail.displayName)
| where isempty(deviceId)
| summarize RiskLevels = make_set(RiskLevelDuringSignIn), ResultTypes = make_set(ResultType), IPs = make_set(IPAddress) by CorrelationId, UserPrincipalName
// Optional to only filter on events with a RiskLevel during the sign-in
// | where RiskLevels has_any ("low", "medium", "high")
```

View query results >

Entity mapping

Account

DisplayName

UserPrincipalName

+ Add identifier

ip

Address

IPs

+ Add identifier

+ Add new entity

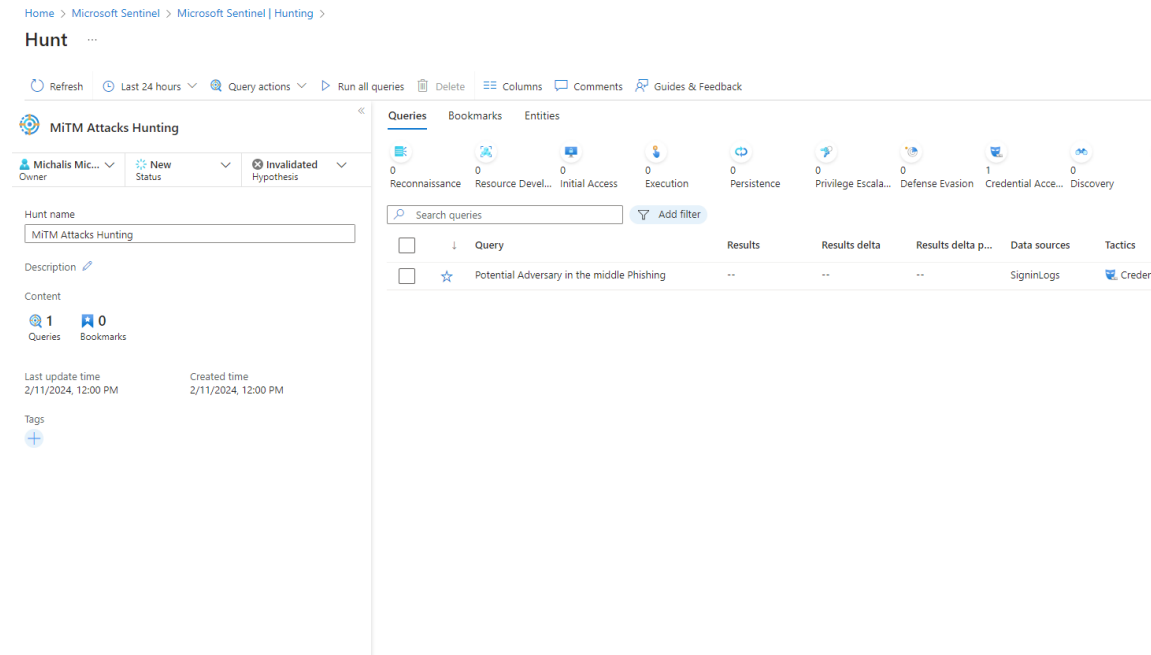
Tactics and techniques

2 selected



# Hunting in Microsoft Sentinel

- Group multiple queries into Hunts, for example if you have multiple queries related to a Threat Actor, or a Technique
- Contextualize based on Status (New, Active, Closed, Backlog and Approved), Hypothesis (Unknown, Invalidated and Validated) or Tags
- Documentation opportunities: Description and Comments



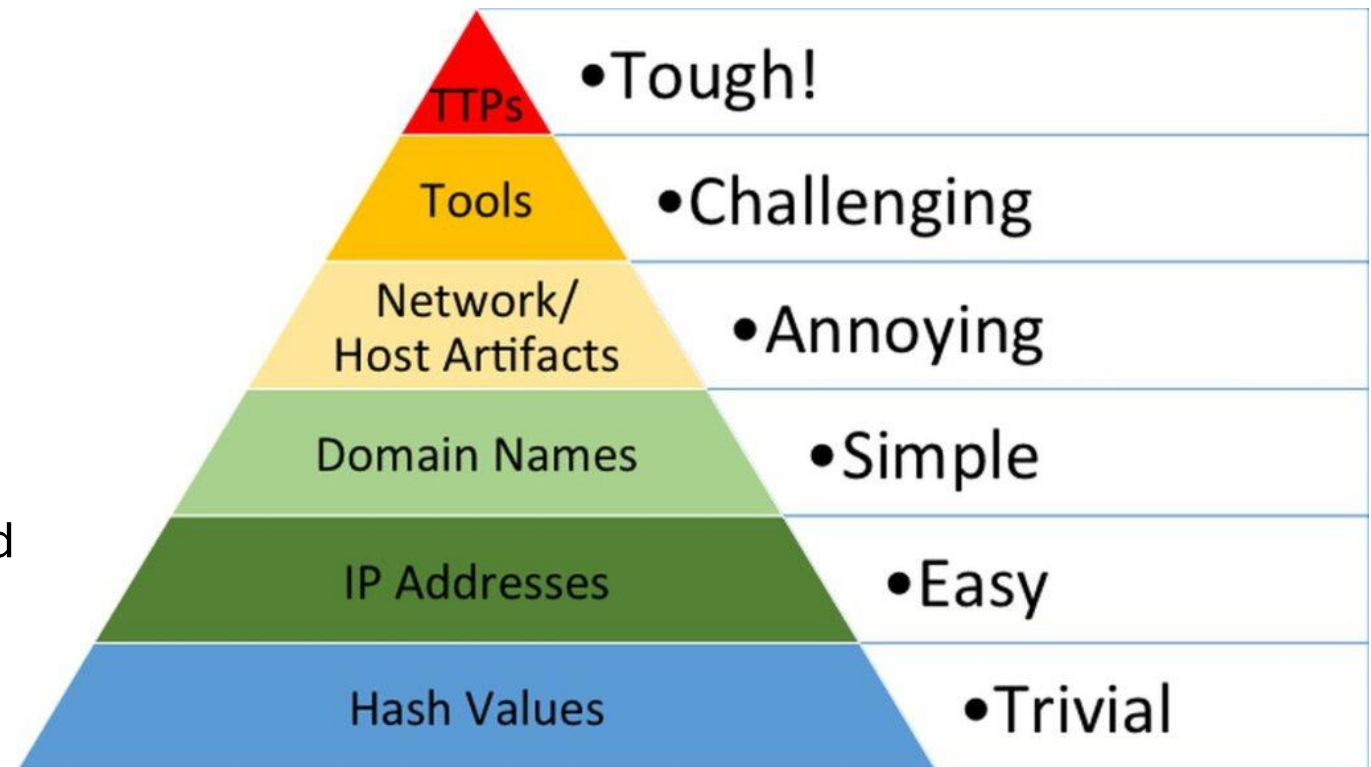
# Hunting in Microsoft Sentinel

```
1 // Groups MFA phone registration events into the number that was registered,
2 // can be useful to detect threat actors registering multiple accounts to the same numbers for persistence
3 // Data connector required for this query - Azure Active Directory - Audit Logs
4 // Source: https://github.com/reprise99/Sentinel-Queries/
5 AuditLogs
6 | where TimeGenerated > ago(90d)
7 | where TargetResources has "PhoneNumber"
8 | where OperationName has "Update user"
9 | where TargetResources has "StrongAuthenticationMethod"
10 | extend InitiatedBy = tostring(parse_json(tostring(InitiatedBy.user)).userPrincipalName)
11 | extend UserPrincipalName = tostring(TargetResources[0].userPrincipalName)
12 | extend targetResources=parse_json(TargetResources)
13 | mv-apply tr = targetResources on (
14 |     extend targetResource = tr.displayName
15 |     | mv-apply mp = tr.modifiedProperties on (
16 |         where mp.displayName == "StrongAuthenticationUserDetails"
17 |         | extend NewValue = tostring(mp.newValue)
18 |     )
19 | project TimeGenerated, NewValue, UserPrincipalName,InitiatedBy
20 | mv-expand todynamic(NewValue)
21 | mv-expand NewValue.[0]
22 | extend AlternativePhoneNumber = tostring(NewValue.AlternativePhoneNumber)
23 | extend Email = tostring(NewValue.Email)
24 | extend PhoneNumber = tostring(NewValue.PhoneNumber)
25 | extend VoiceOnlyPhoneNumber = tostring(NewValue.VoiceOnlyPhoneNumber)
26 | project TimeGenerated, UserPrincipalName, InitiatedBy,PhoneNumber, AlternativePhoneNumber, VoiceOnlyPhoneNumber, Email
27 | where isnotempty(PhoneNumber)
28 | summarize ['Count of Users']=dcount(UserPrincipalName), ['List of Users']=make_set(UserPrincipalName) by PhoneNumber
29 | sort by ['Count of Users'] desc
```

# don't forget...

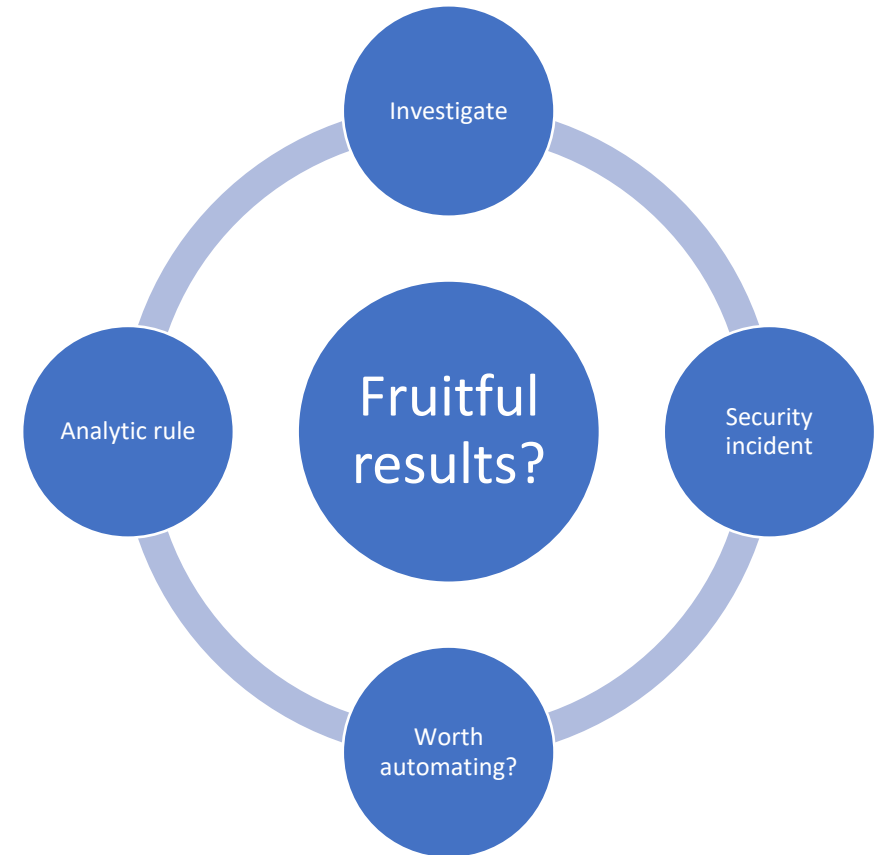
The Pyramid of Pain nurtures proactive Threat Hunting and hence, allowing organizations to stay one step ahead of adversaries.

KQL can help facilitate TTPs through suspicious behaviors and big data anomalies.



# wrapping up Threat Hunting

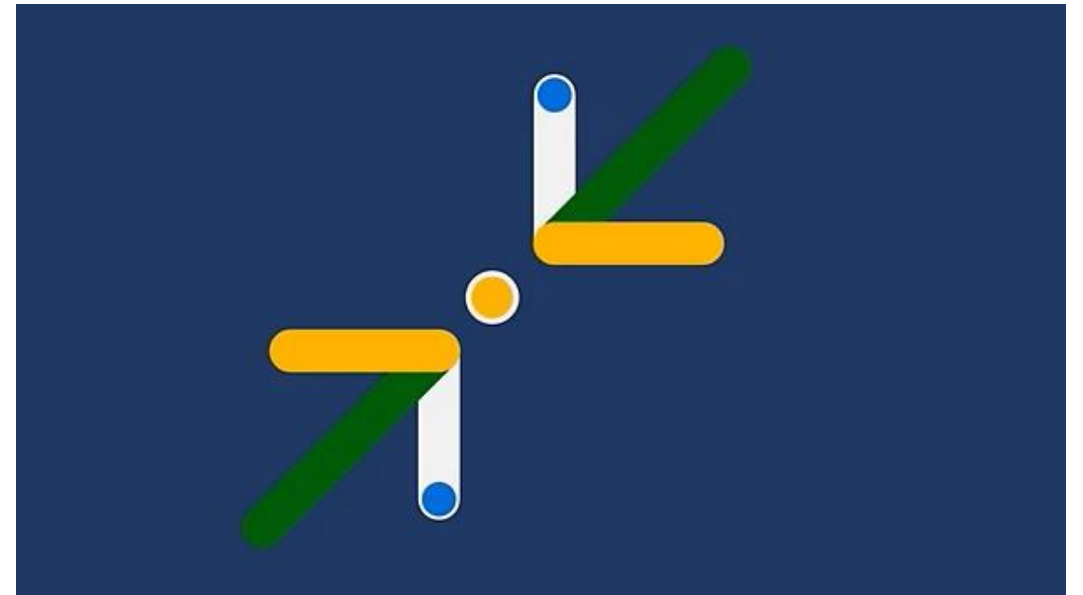
- A fruitful hunt would lead to an Investigation
- Investigation could lead to a Security incident
- Based on the process so far, is this hunt worth the automation?
- If yes, then build an Analytic rule



# Perform Incident Response using KQL and Microsoft Security Technologies

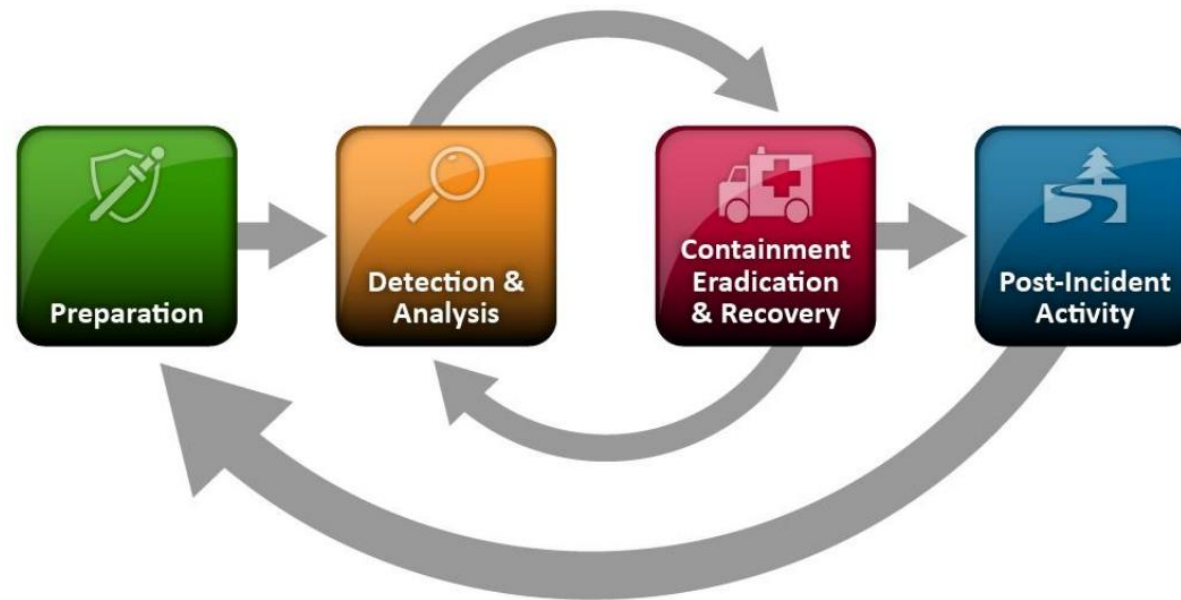
# Incident Response basics

Incident response is the practice of investigating and remediating active attack campaigns on your organization. Incident response is part of the security operations (SecOps) discipline and is primarily reactive in nature.



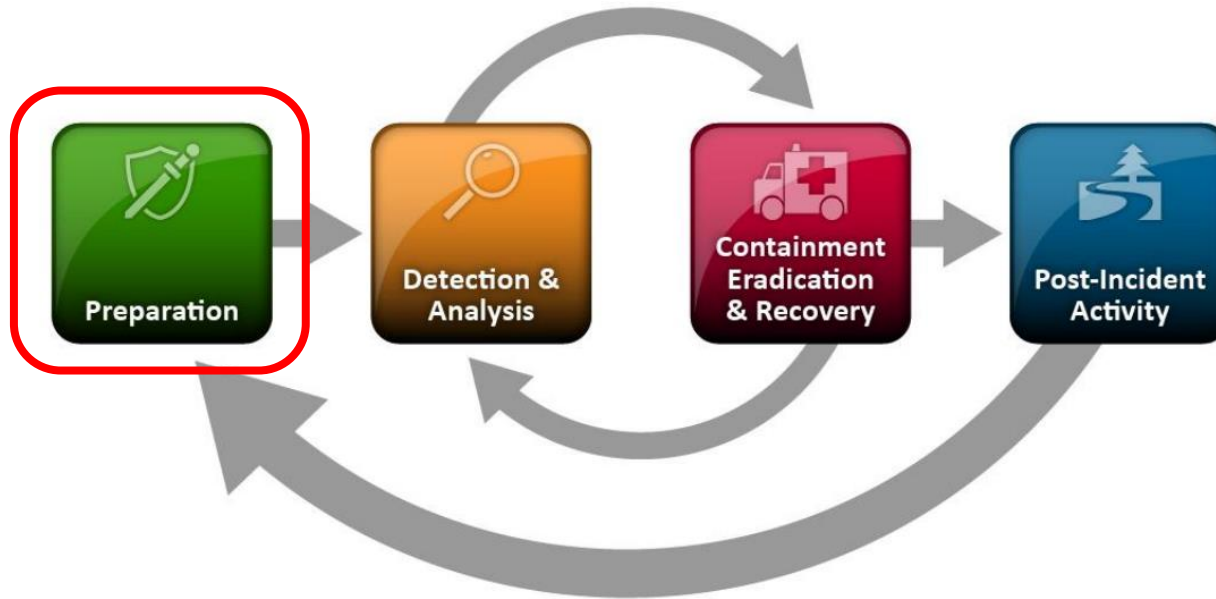
# Incident Response Life Cycle

- NIST.SP.800-61r2
- A four-step process for incident response, illustrated in the diagram below
- How can KQL relate to each step?





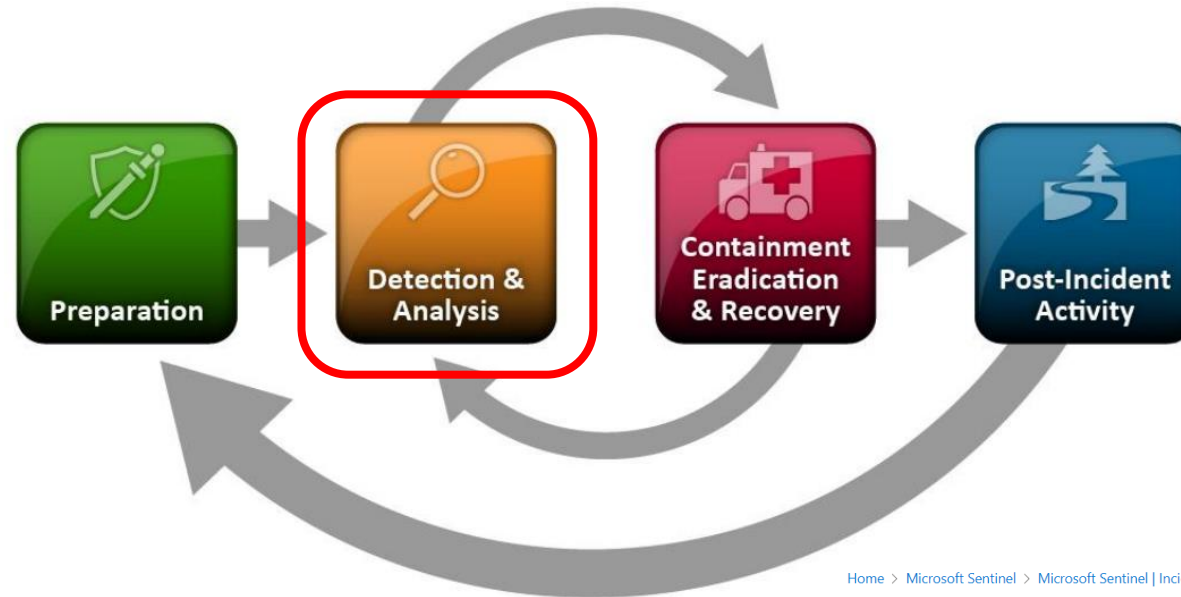
# Incident Response Life Cycle



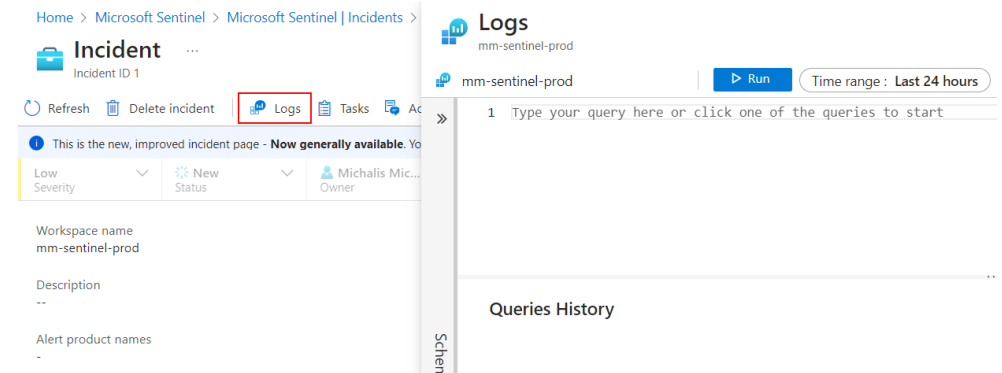
- Advanced Hunting, Custom Detection Rules in Defender XDR
- Logs, Threat Hunting and Analytics in Microsoft Sentinel
- Build Playbooks in Microsoft Sentinel
- Build Incident Response Playbooks and integrate steps through KQL opportunities



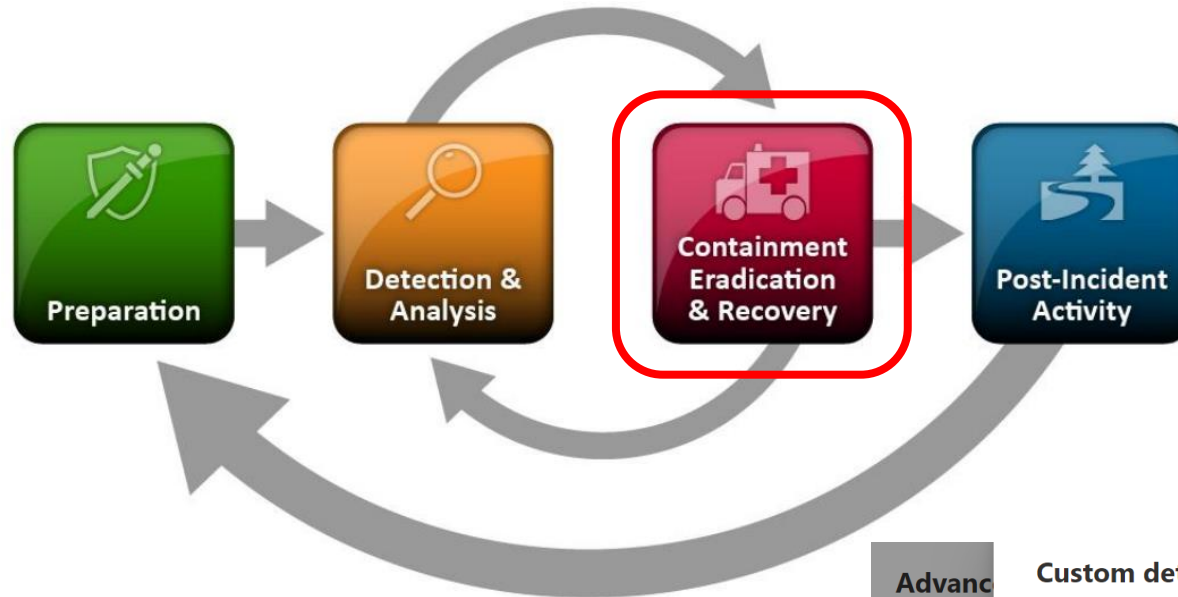
# Incident Response Life Cycle



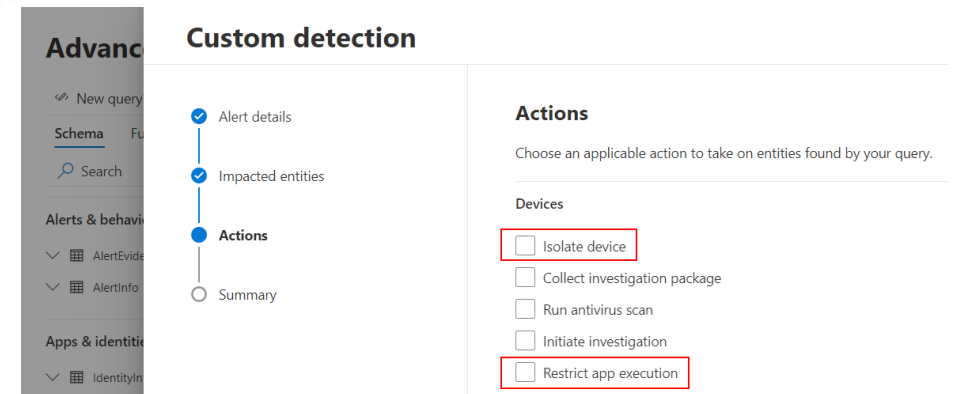
- Advanced Hunting in Defender XDR
- Logs in Microsoft Sentinel
- Opportunity from Incident in Microsoft Sentinel to Pivot to Logs and investigate



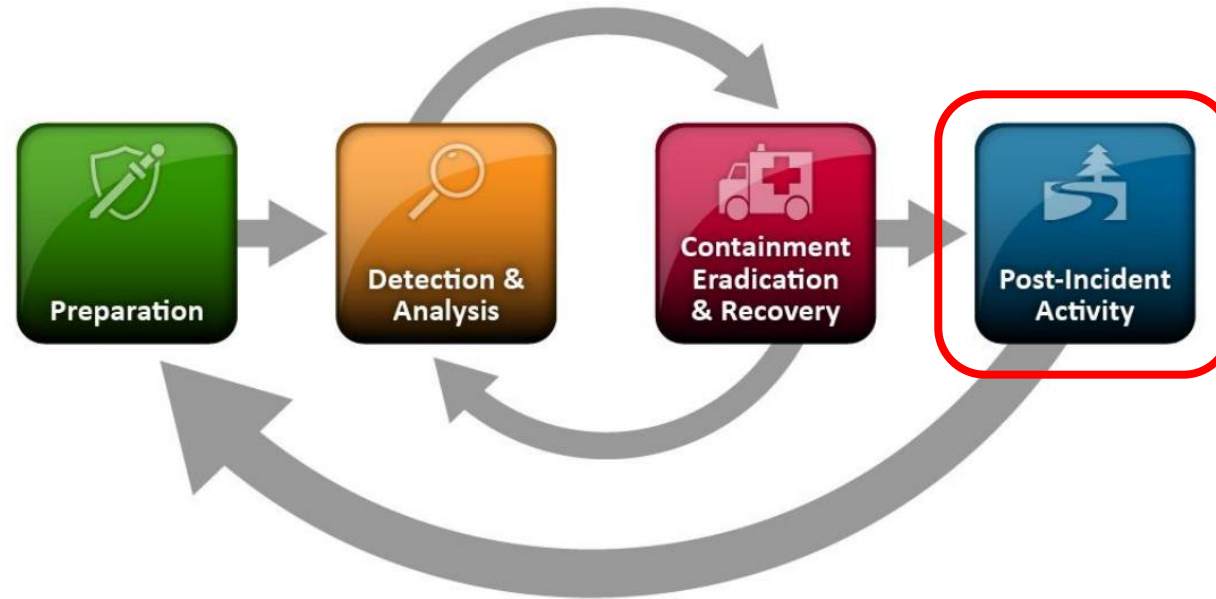
# Incident Response Life Cycle



- Custom Detection Rules from Defender XDR could lead to containment actions
- Playbooks in Microsoft Sentinel could also lead to containment actions



# Incident Response Life Cycle



- Review Custom Detection Rules, Analytics and Playbooks
- Review KQL Queries stored in Queries at Advanced Hunting and Logs
- Build new queries if required and use them for Threat Hunting or Detection
- Keep relevant documentation up to date

# Playbook step investigation

The following query could be a step from a Business Email Compromise (BEC) Incident Response Playbook and will present email details that have been identified as suspicious after delivery.

```
1 let CompromizedEmailAddress = ""; // Insert the email address of the compromised email address
2 let Timeframe = 2d; // Choose the best timeframe for your investigation
3 let EmailInformation = EmailEvents
4     | where RecipientEmailAddress == CompromizedEmailAddress
5     | where DeliveryAction != "Blocked"
6     | project Timestamp, NetworkMessageId, SenderMailFromAddress, SenderFromAddress, SenderDisplayName, ThreatNames;
7 EmailInformation
8     | join (EmailPostDeliveryEvents
9         | where ThreatTypes != ""
10        | project Timestamp, NetworkMessageId, Action, ActionType, ActionTrigger, ActionResult,
11            DeliveryLocation, ThreatTypes, DetectionMethods
12    ) on NetworkMessageId
13    | sort by Timestamp desc
```

# wrapping up incident response

## Further Incident Response Opportunities:

- Interact through API to build automations or conduct investigation actions.
- You may use Azure Data Explorer (ADX) to load your evidence files for analysis.

*“If you're proactive, you focus on preparing. If you're reactive, you end up focusing on repairing.”*

— John C. Maxwell



# Closing remarks

You can familiarize with KQL in many ways:

- Growing community of query contributions on GitHub, blogs, Microsoft Tech Community blogs, podcasts, newsletters etc. Don't forget, queries come as-is, review and test before formal roll-out. Make sure they fit your environment and requirements!
- Formal training providers apart from SC-200 and Ninja Training, already out there.
  - Also, book is on the way 😊
- CTF-a-like learning, Kusto Detective Agency and KC7 Cyber offer gamified learning experience.

There is only one way to learn KQL...

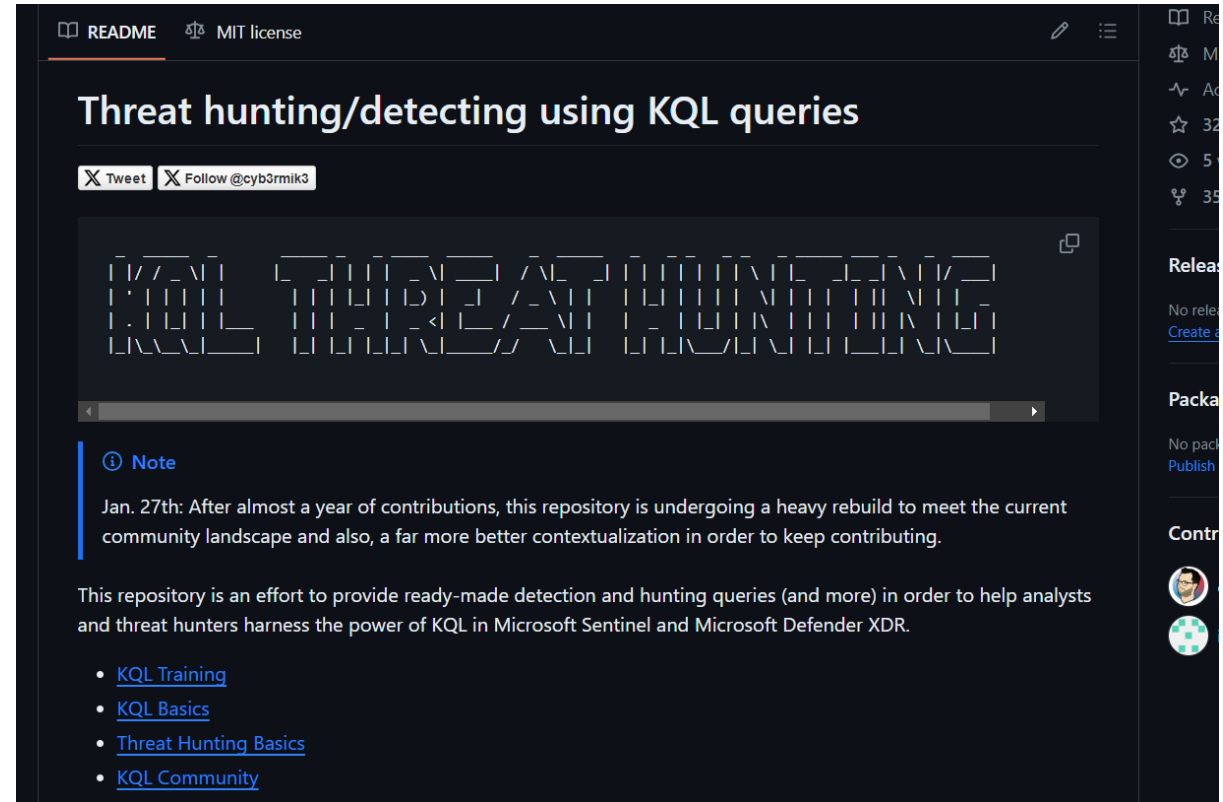
^ Query

```
1 LearnKQL
2 | where AnalystAction has "Practice"
```



# Resources

<https://github.com/cyb3rmik3>



Presentation notes & References:  
<https://github.com/cyb3rmik3/presentations>

# **AI, Cloud & Modern Workplace Conference 2024**

15, 16 & 17 February , Online Conference

**Thank You!**