Make Intune reporting better and secure

Damien Van Robaeys
@syst_and_deploy

Sponsors

onedeploy
Windows Deployment Reimagined

ZERO TOUCH

Insight

control up

PATCH MY PC

robopack

numecent

Microsoft

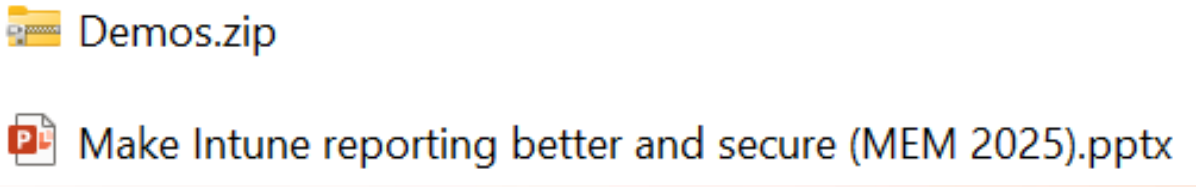
eido

Rimo3

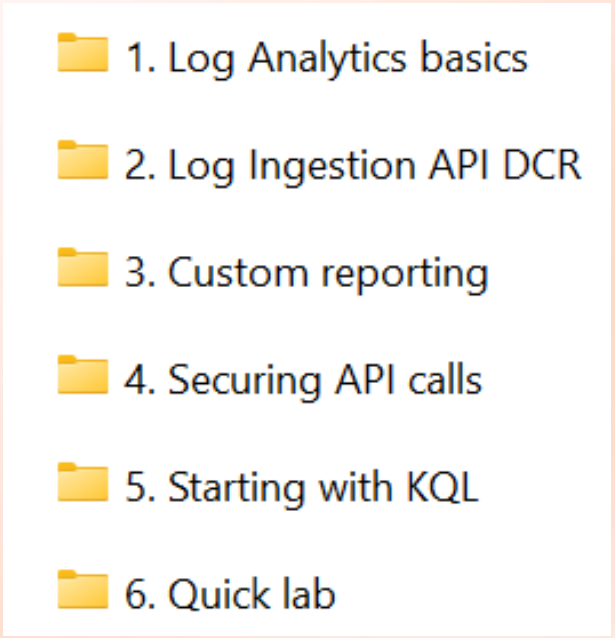
Slides & demos

Slides are available

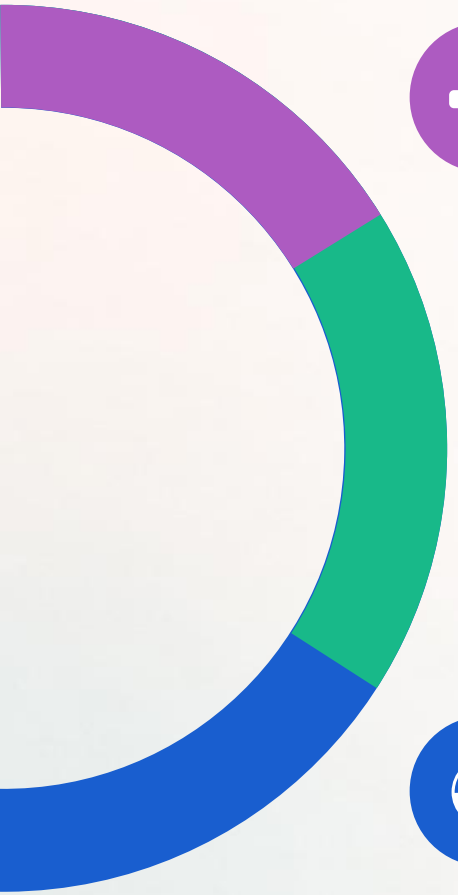
https://github.com/damienvanrobaeys/Events_Slides



SCHED



Custom reports



Monitor BSOD (blue screen)

A dashboard allowing you to troubleshoot BSOD on your devices

Available on my blog
Link here

Drivers inventory

A dashboard allowing you to see all drivers installed on all devices

Available on my blog
Link here

Windows Secure Boot certificate expiration dashboard

Monitor Secure boot certificate expiration on your Intune devices

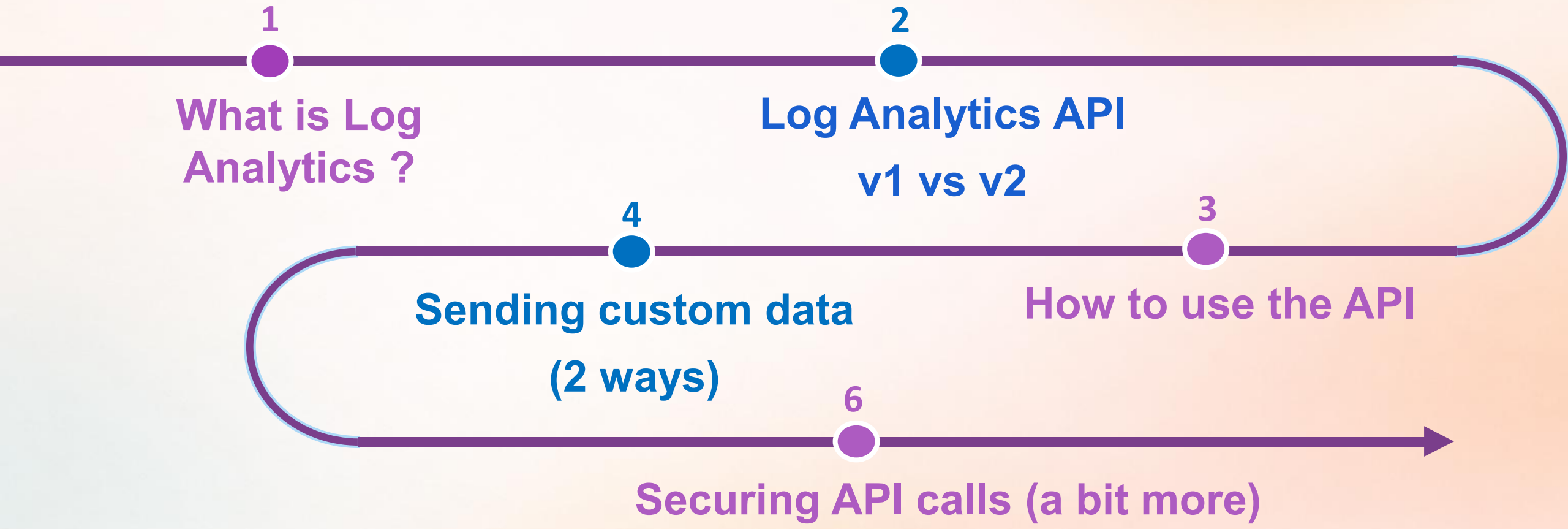
Available on my blog
Link here

Overview

Who use Log Analytics ?

Who use Azure Automation ?

What will you learn ?



About me

Modern Workplace consultant (Metsys)
Dual MVP Microsoft (9 years) & Official Contributor
Working with PowerShell, Intune, MS Graph, MECM, Log Analytics...

Damien Van Robaeys

systanddeploy.com

@syst_and_deploy

@systanddeploy

damien.vanrobaeys@gmail.com

Damien Van Robaeys

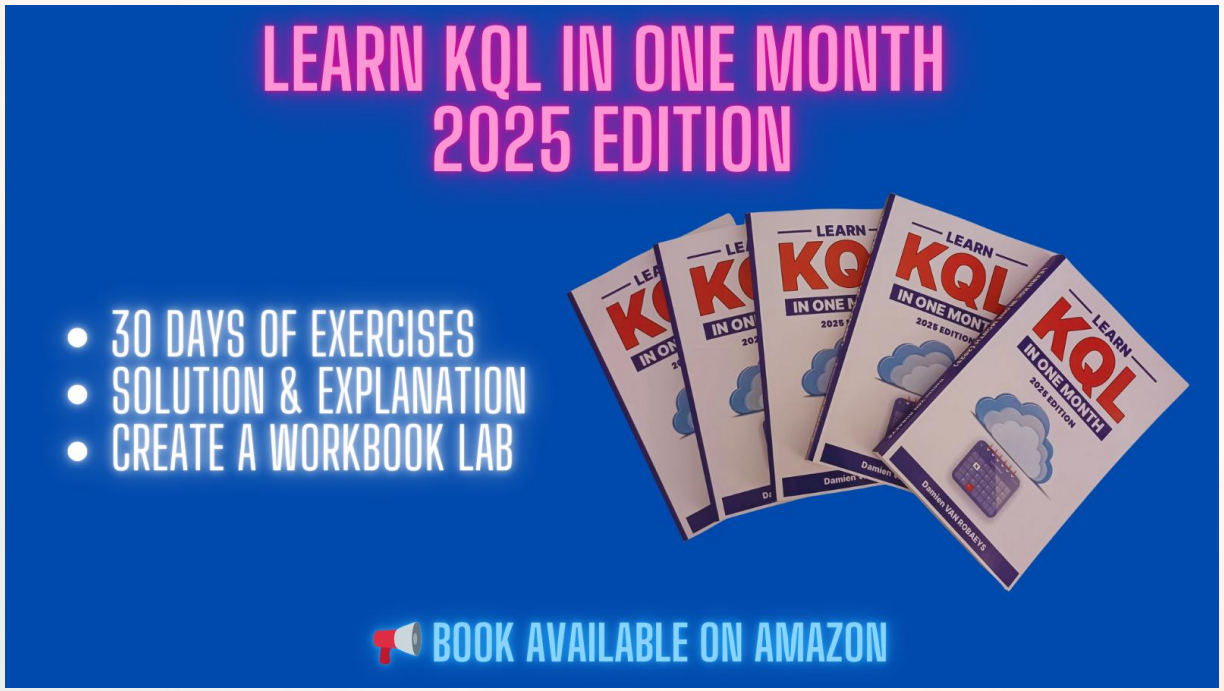
Microsoft Most Valuable Professional Award

METSYS

systanddeploy.com

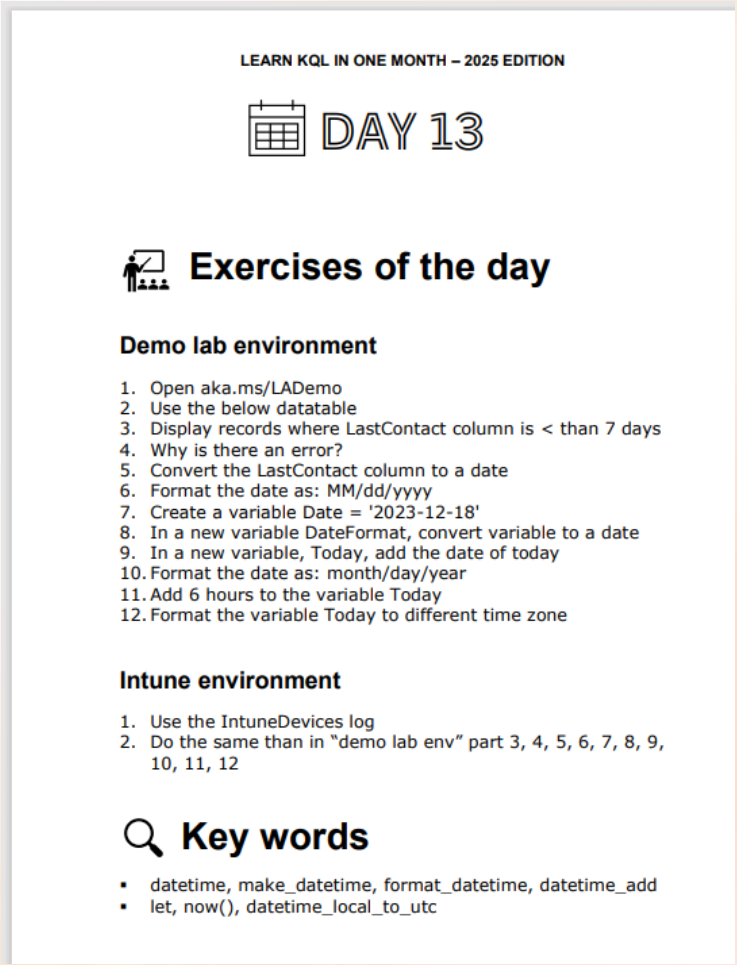
Learn KQL in one month

10 books to win



A yellow paper on your chair ?

If you see it, you win ☺



LEARN KQL IN ONE MONTH – 2025 EDITION

DAY 13

Exercises of the day

Demo lab environment

1. Open aka.ms/LADemo
2. Use the below datatable
3. Display records where LastContact column is < than 7 days
4. Why is there an error?
5. Convert the LastContact column to a date
6. Format the date as: MM/dd/yyyy
7. Create a variable Date = '2023-12-18'
8. In a new variable DateFormat, convert variable to a date
9. In a new variable, Today, add the date of today
10. Format the date as: month/day/year
11. Add 6 hours to the variable Today
12. Format the variable Today to different time zone

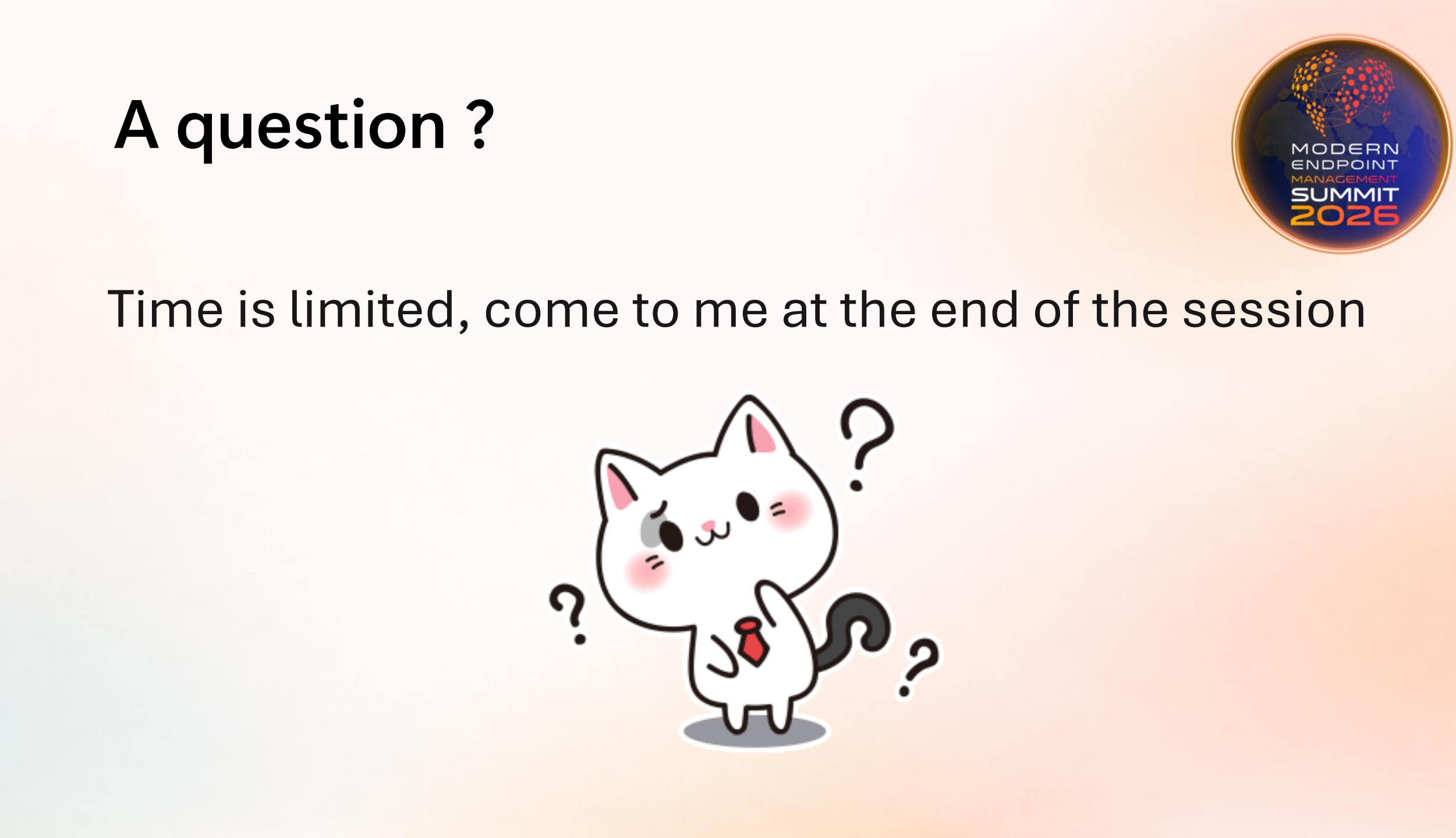
Intune environment

1. Use the IntuneDevices log
2. Do the same than in "demo lab env" part 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Key words

- datetime, make_datetime, format_datetime, datetime_add
- let, now(), datetime_local_to_utc

A question ?



Time is limited, come to me at the end of the session

Log Analytics, what is it ?

What is it ?

- Azure service available from Azure & Intune portals
- Goal: analyze data collected by Azure Monitor
- Centralize data from Azure services, resources, Intune devices…
- Run queries and play with data from your tenant, devices…
- Run KQL queries to investigate, monitor, create dashboard

- In Log Analytics data are located into Logs

By default, Log Analytics is empty you need to configure it

Starting with Log Analytics

Blog post series about starting with Log Analytics

https://www.systanddeploy.com/search/label/LogAnalytics_Start

1. Creating our first Log Analytics workspace
2. Importing your own data into the workspace
3. Creating our first workbook
4. Add Intune data into Log Analytics workspace
5. Running KQL queries in Log Analytics with PowerShell
6. Creating a lab by importing a CSV with fake data
7. Give your workbook a better look
8. Sending data to Log Analytics from Azure Automation
9. Running KQL queries on a workspace through Azure Automation
10. Securing Data transmission to Log Analytics v2 with Azure Automation
11. Securing Data transmission to Log Analytics v2 with Azure Function

Prerequisites

1. Azure subscription
2. Resource group (container that holds all your resources) *
3. Log Analytics workspace

* More info about resource group

https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal

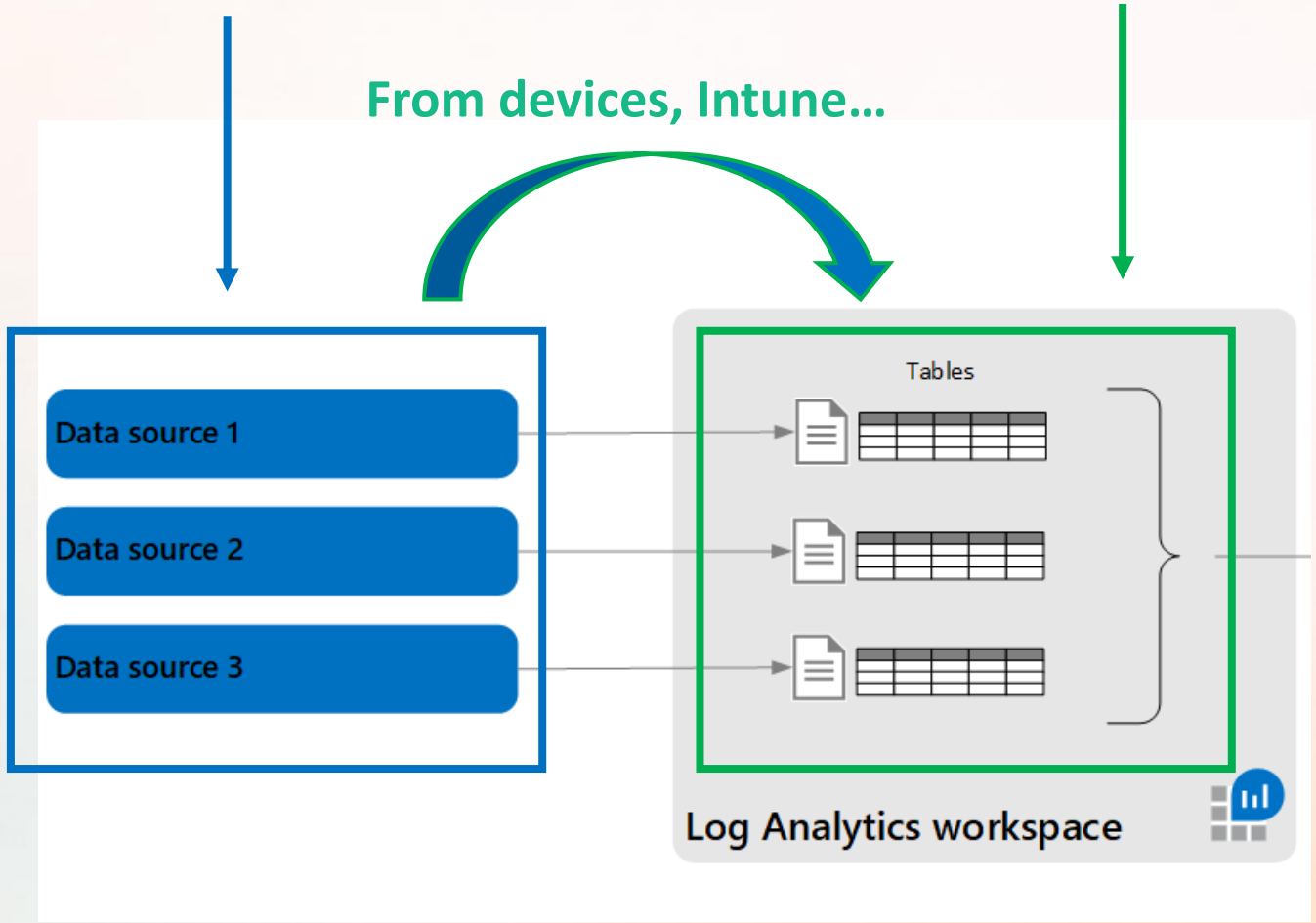
Log Analytics workspace

- When you collect logs, all data are stored in a workspace

- Contains all things relative to your logs and data:
  - ✓ Logs, Custom log
  - ✓ Microsoft Sentinel, Microsoft Defender portal

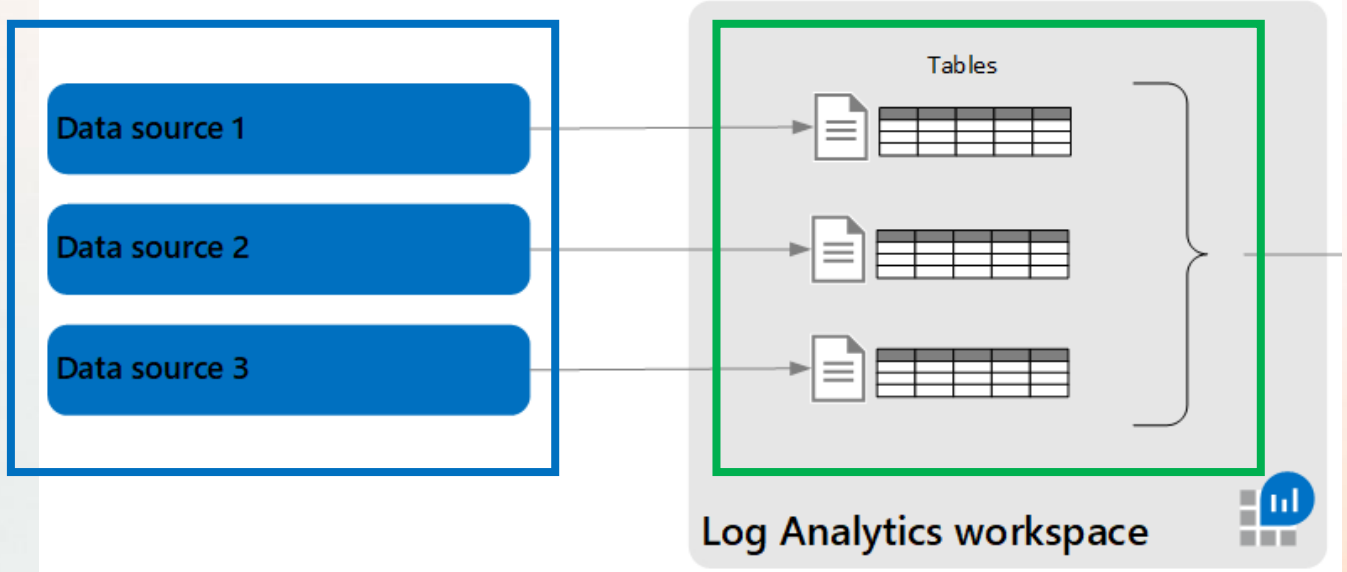
Data structure

- Data are sent from a source in Tables in a Workspace



Data structure

- Tables organized in columns containing rows of data
- A table = A Log

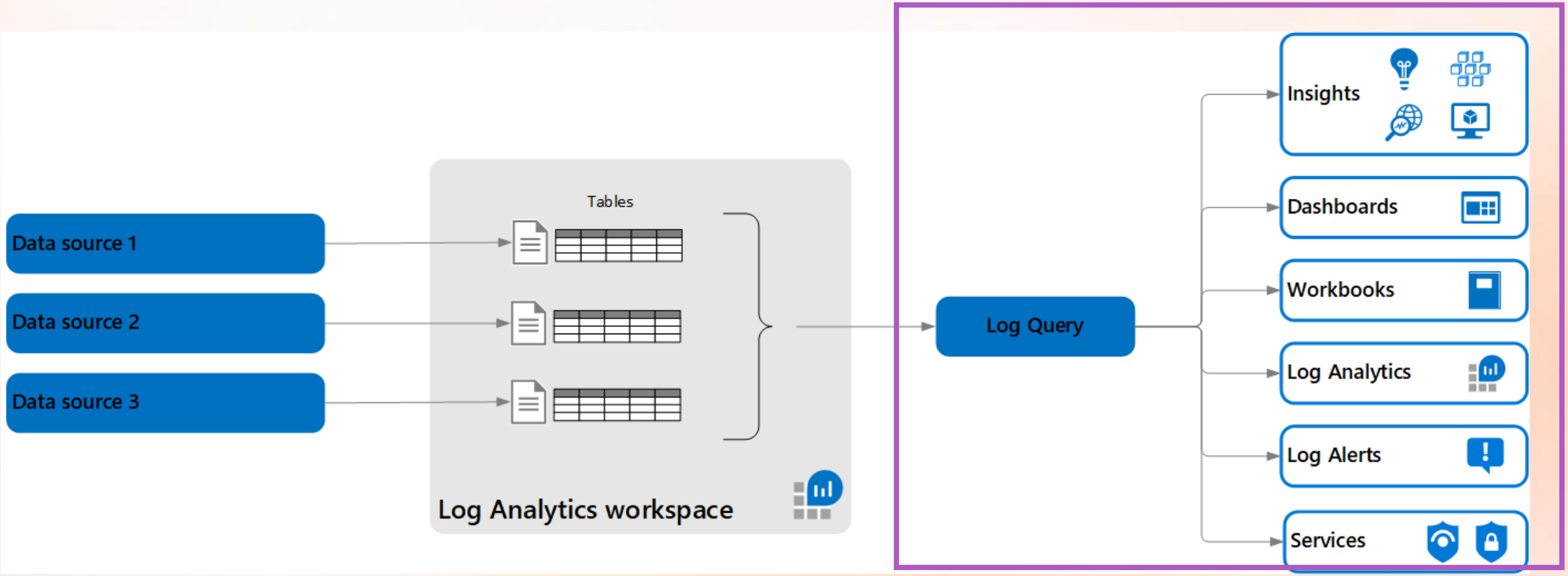


Column name ↓	Description	Type
CurrentBIOSDate		Datetime
CurrentBIOSDateFormat		Datetime
CurrentBIOSDaysOld		Int
CurrentBIOSDaysOldRange		String
CurrentBIOSVersion		String

Device	ModelFamilyname	IsUptoDate	CurrentBIOSVersion	LastBIOSVersion
[redacted]	ThinkPad T14s	No	1.32	1.34
[redacted]	ThinkPad T14s	No	1.32	1.34
[redacted]	ThinkPad T14s	No	1.32	1.34
LAPTOP-5...	ThinkPad T14s Gen 3	No	1.28	1.46

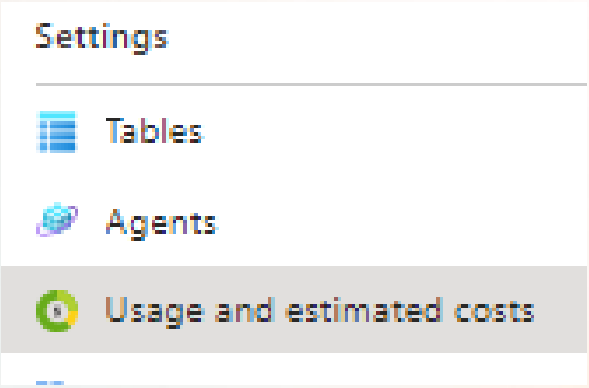
Data structure

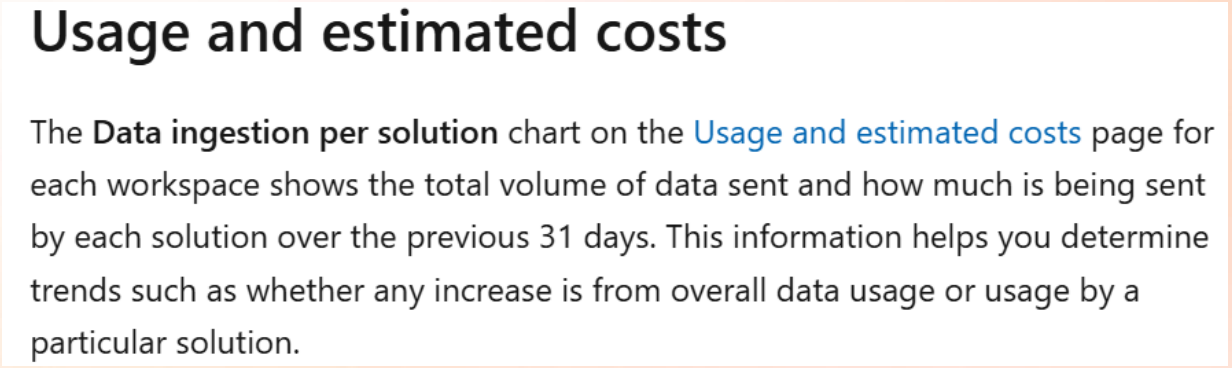
- Query log and play with data to create dashboard…



What about pricing ?

- Log Analytics is priced/billed by ingestion and retention
- The default pricing for Log Analytics is a pay-as-you-go
- Default pricing (get it through the portal):
  - ✓ Ingestion: €2,80/GB
  - ✓ Retention: €0,61/GB
- For more info go to: **Usage and estimated costs**





Explore your usage

- Useful table to analyze data & usage: **Usage** (in LogManagement)
- Use **IsBillable == true** (records for which there's an ingestion charge)
- Query to get billable Data ingested by table

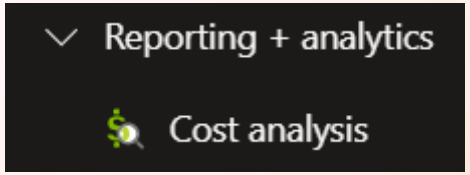
```
Usage
| where IsBillable
| summarize BillableDataGB=sum(Quantity) / 1.0E3 by DataType
| extend BillableDataGBRound = round(BillableDataGB, 2)
| sort by BillableDataGB desc
| project Log=DataType,['Billable Data']=BillableDataGBRound
```

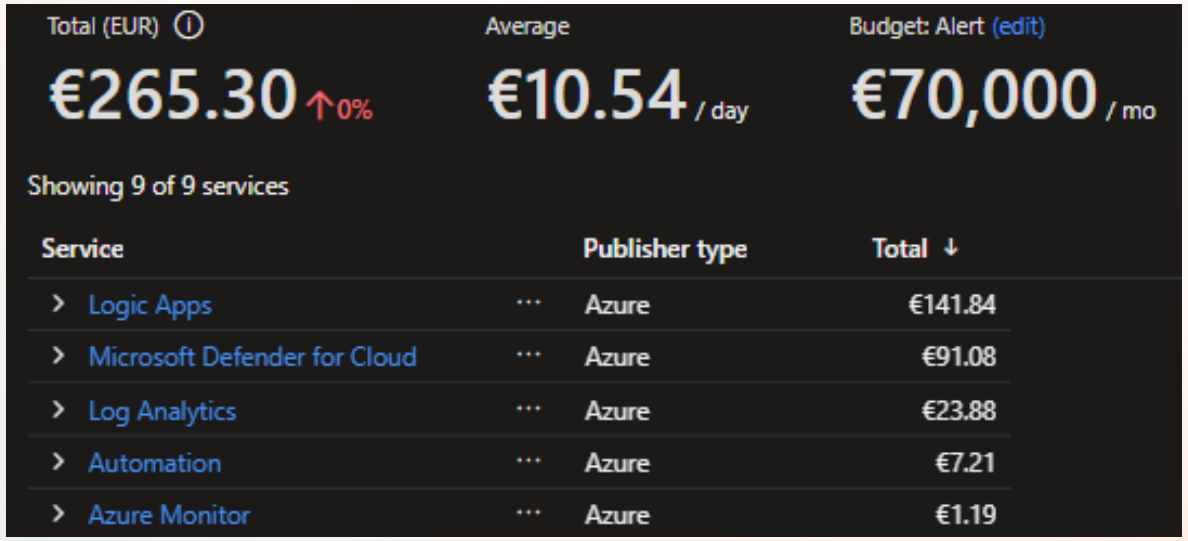
+ More info:

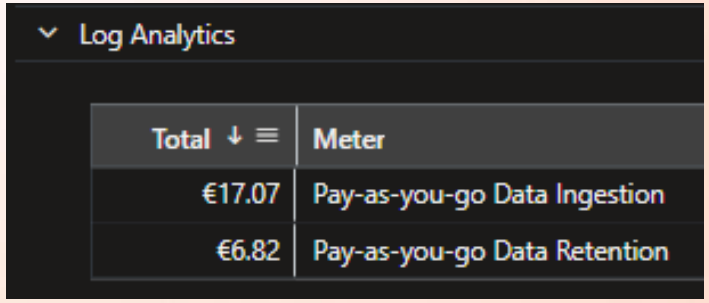
https://learn.microsoft.com/en-us/azure/azure-monitor/logs/analyze-usage

Cost analysis

- Cost analysis: **Cost Management** > **Cost analysis**
- Filter on resource group, workspace, month…



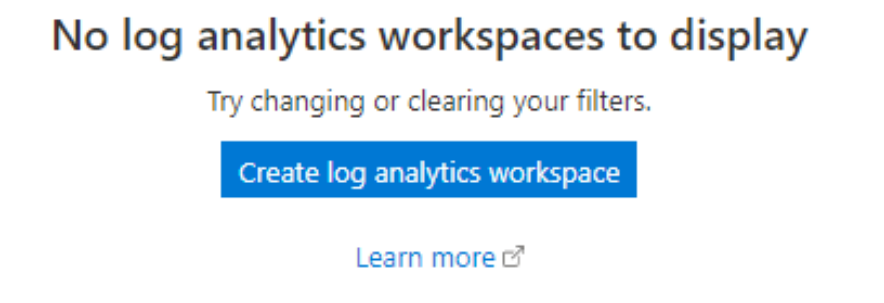


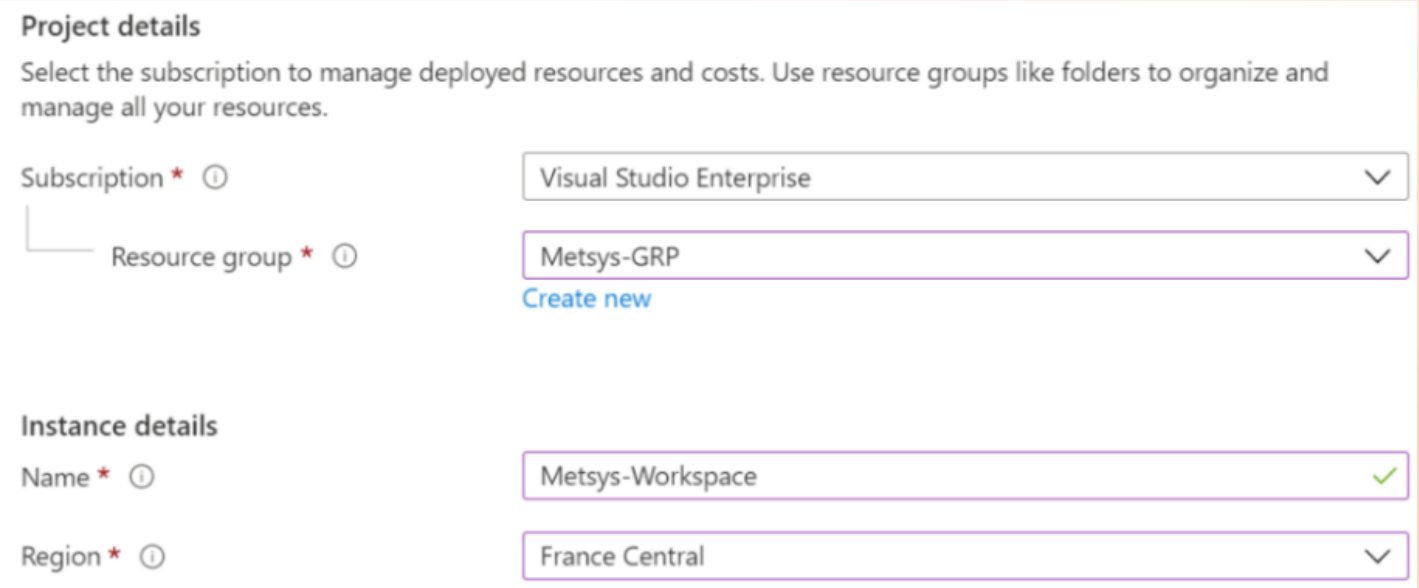


Creating a workspace

Manually

1. Go to **Log Analytics workspaces** > **Create**
2. Fill information

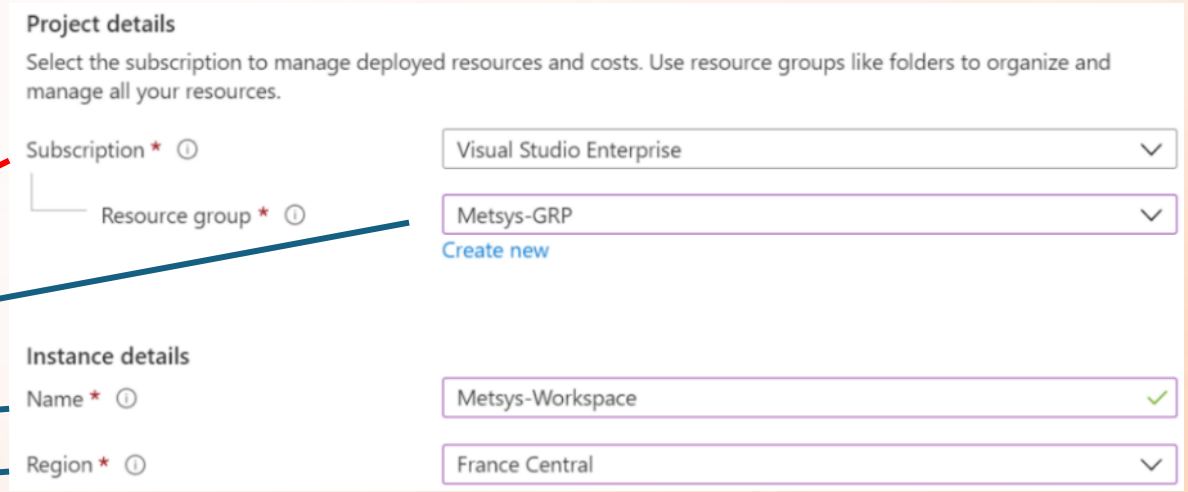




Creating a workspace

With PowerShell

1. Install module **Az**: Install-Module -Name Az -Force
2. Connect-AzAccount
3. New-AzOperationnalInsightsWorkspace

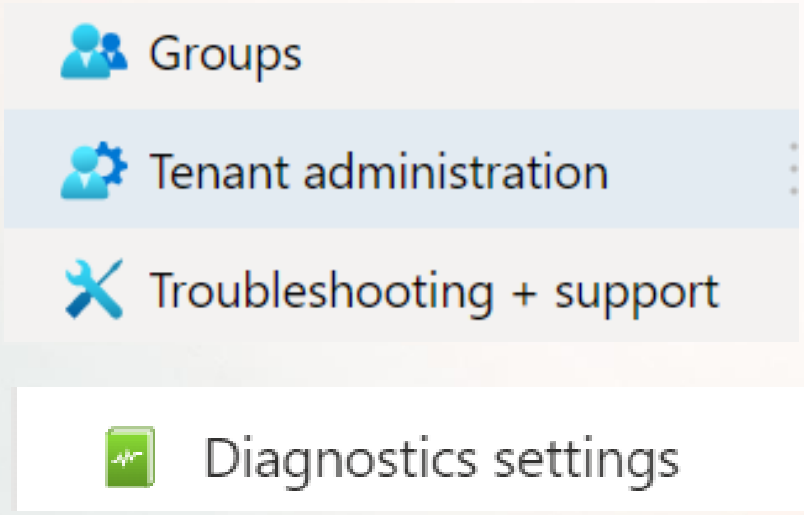


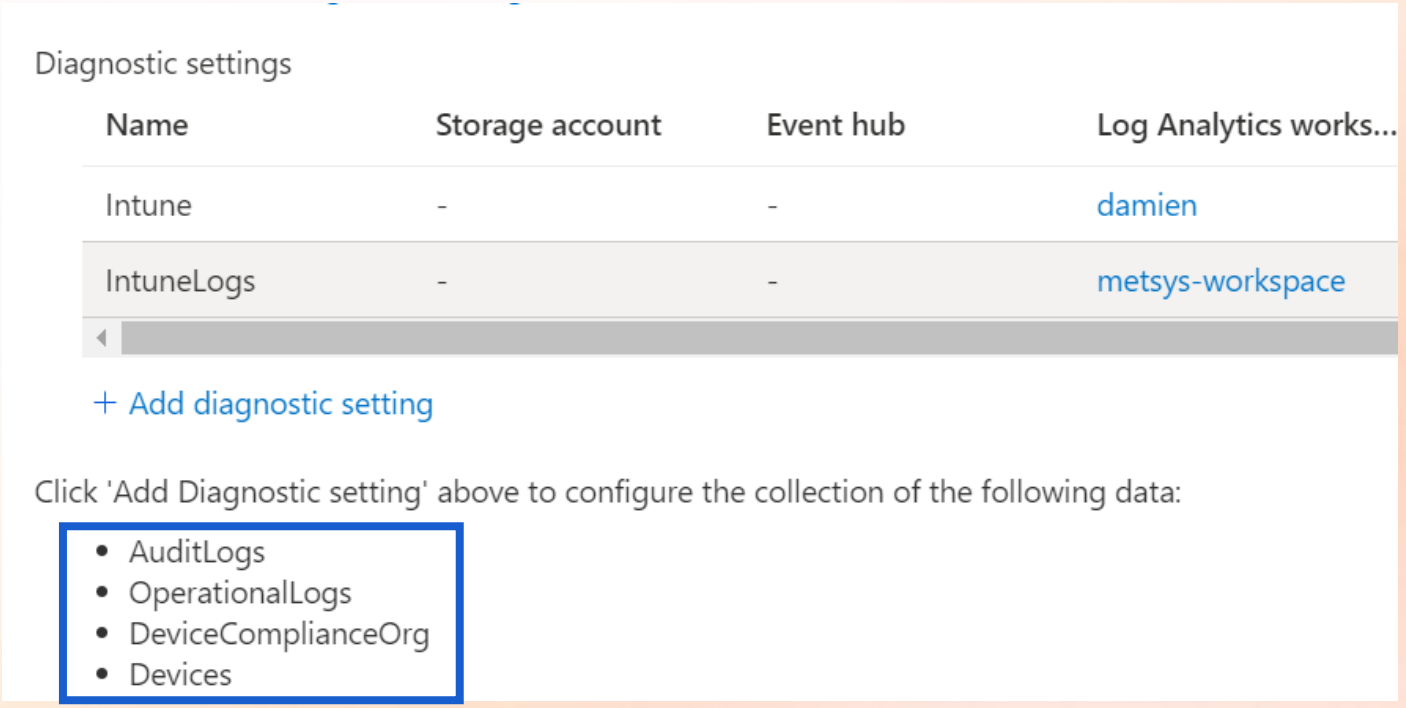

```
$SubscriptionID = "Your subscription"
$ResourceGroup = "Metsys-GRP"
$WorkspaceName = "Metsys-Workspace"
$Location = "francecentral"

Connect-AzAccount -Subscription $SubscriptionID
New-AzOperationalInsightsWorkspace -Location $Location -Name $WorkspaceName -ResourceGroupName $ResourceGroup
```

Adding Intune data

By default Logs part is empty (no tables)

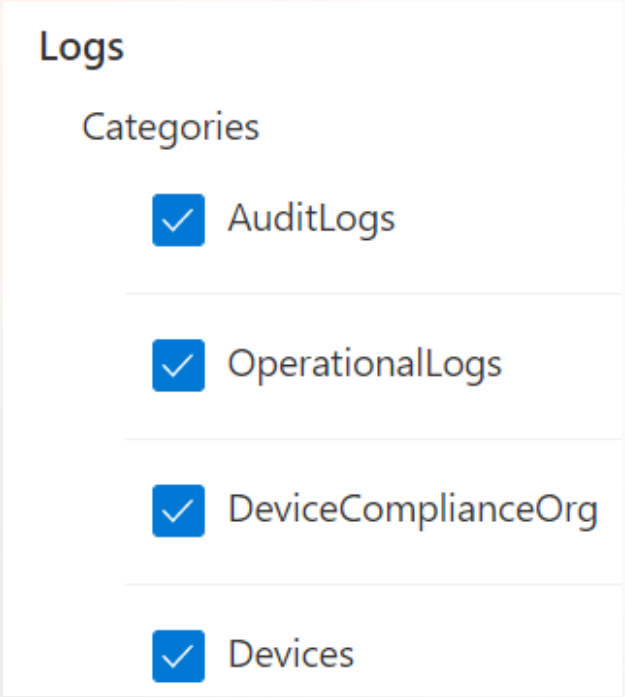
- Go to **Intune > Tenant administration > Diagnostic settings**
- Click on **Add diagnostic setting**





Intune logs

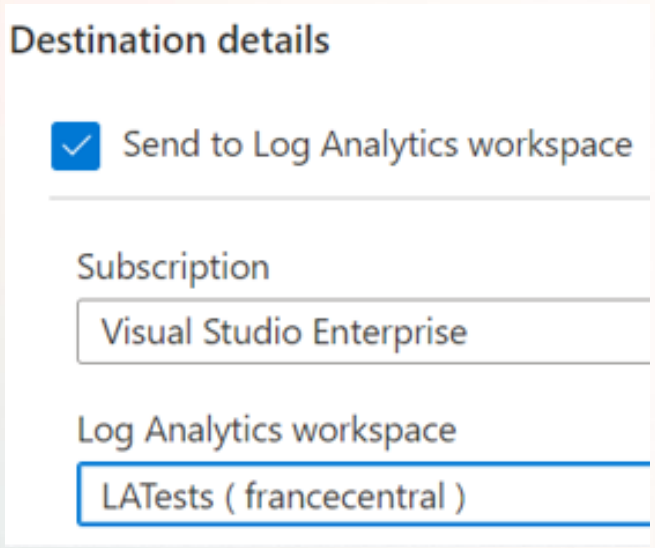
Select built-in logs to add



- **AuditLogs**: activities record that generate a change in Intune Intune (including create, update, delete, assign, and remote actions)
- **OperationalLogs**: details about users and devices that successfully enrolled (or failed) to enroll and details on non-compliant devices
- **DeviceComplianceOrg**: Contains report for device compliance in Intune and details on non-compliant devices
- **Devices**: device inventory and status information about Intune enrolled and managed devices

Intune logs

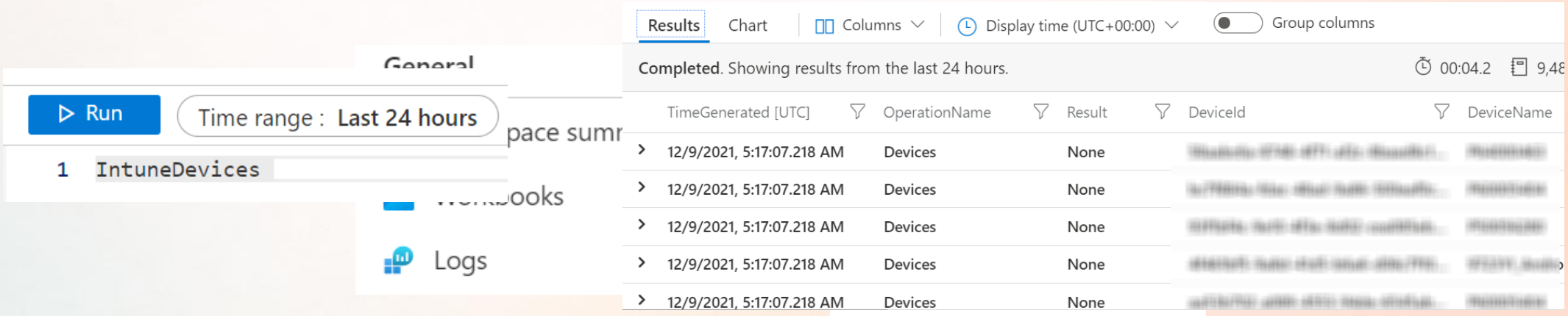
Select workspace where to send data



- **AuditLogs**: activities record that generate a change in Intune Intune (including create, update, delete, assign, and remote actions)
- **OperationalLogs**: details about users and devices that successfully enrolled (or failed) to enroll and details on non-compliant devices
- **DeviceComplianceOrg**: Contains report for device compliance in Intune and details on non-compliant devices
- **Devices**: device inventory and status information about Intune enrolled and managed devices

Log Analytics and Intune

- Go to **Logs**
- You have now Intune logs
  - Example **IntuneDevices** (list enrolled devices)
- You can now run query on your Intune datas
- Language for query is **KQL** (Kusto Query Language)



Log Analytics, Battle of the API

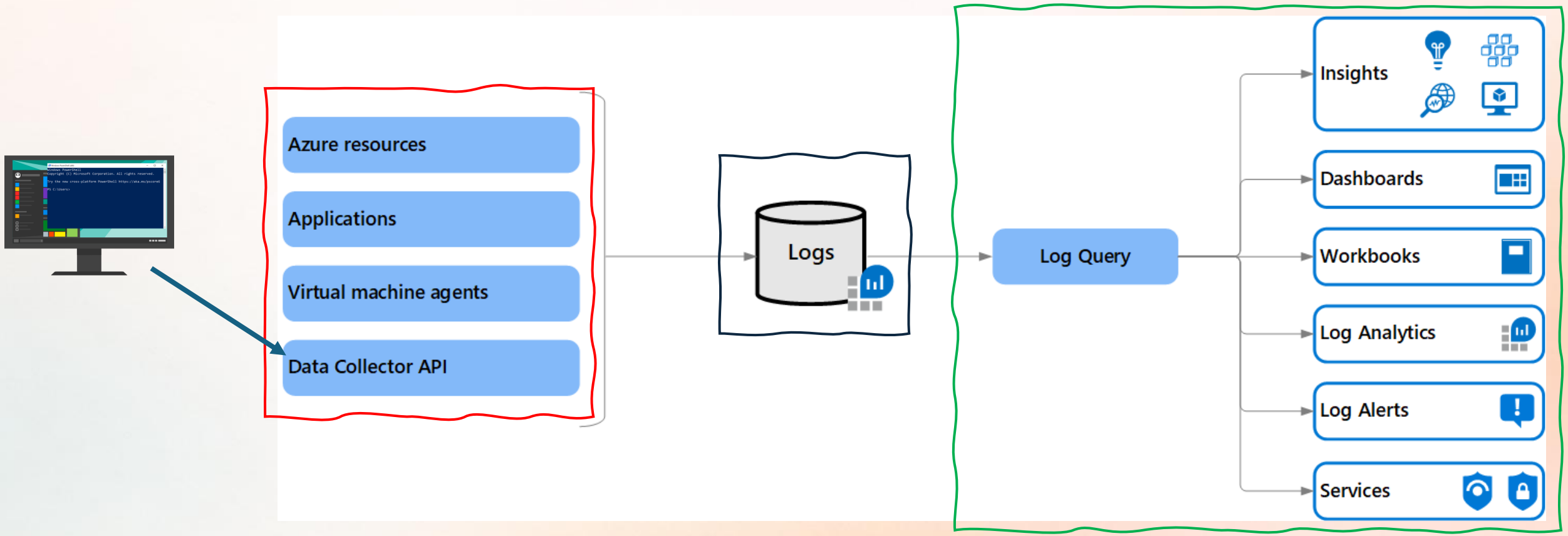
Log Analytics APIs

- Log Analytics V1: Data Collector API (**Old way and deprecated**)
- Log Analytics V2: Log Ingestion API (**Modern way**)

> **Note**
>
> The Azure Monitor HTTP Data Collector API has been deprecated and will no longer be functional as of 9/14/2026. It's been replaced by the **Logs ingestion API**.

Log Analytics v1: data collector API

- Script use Data Collector API to send collected data to a Custom Log
- Then you can query your logs and create dashboard, alerts…



Log Analytics v2: Log Ingestion API

Prerequisites

1. Custom table must exist before in Log Analytics workspace
2. App registration to authenticate against the API **OR** a managed identity*
3. Data collection endpoint (DCE) to receive data
4. Data Collection Rule (DCR) to direct the data to the target table

❖ **!! Data sent to the API must be formatted in JSON**

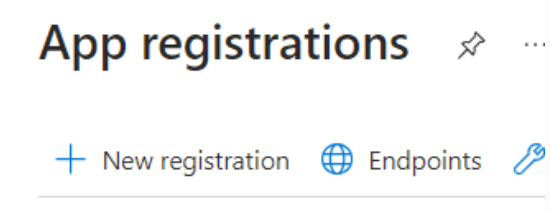
Nice script from **Morten Waltorp Knudsen**:
https://learn.microsoft.com/en-us/azure/azure-monitor/logs/set-up-logs-ingestion-api-prerequisites

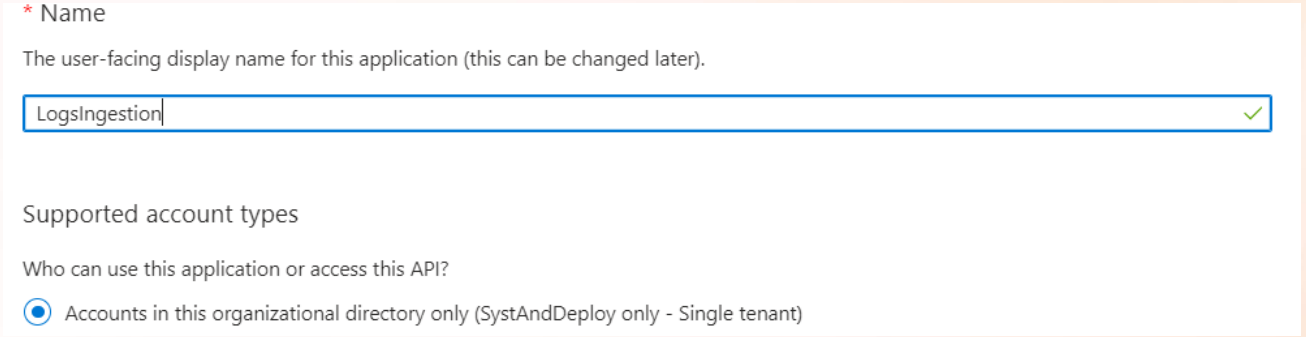
*** Depending of context ?**

- If running under Azure (automation/function app): use Managed identity
- If running outside of Azure (remediation script): use app registration

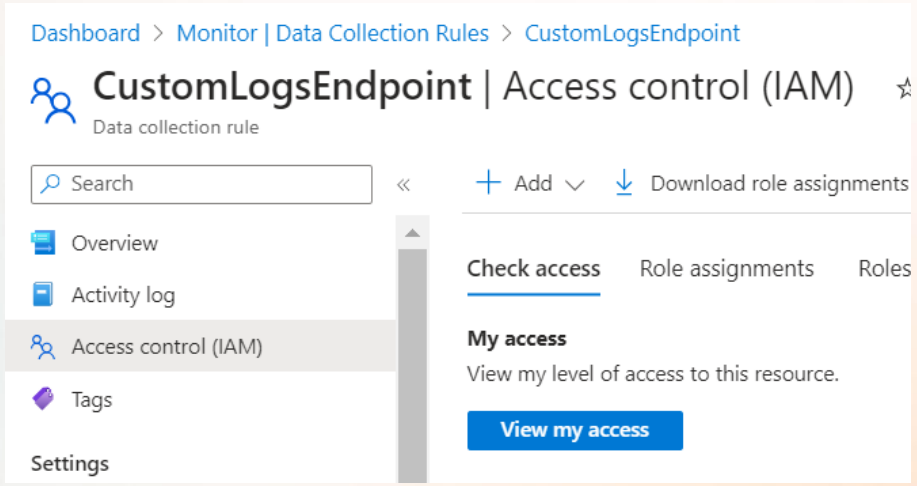
First step: app registration

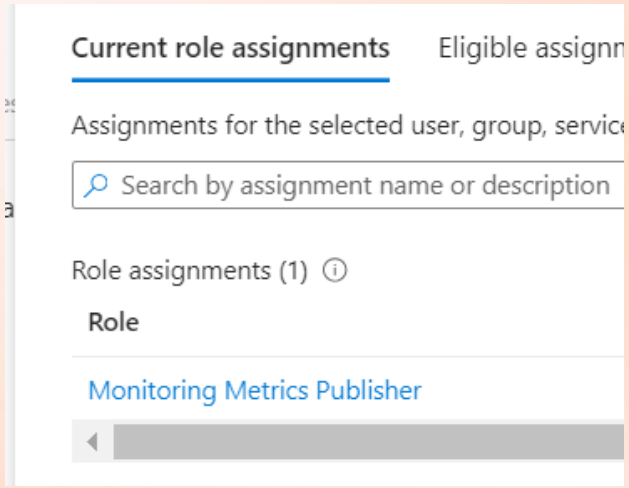
1. Create app registration to authenticate against the API





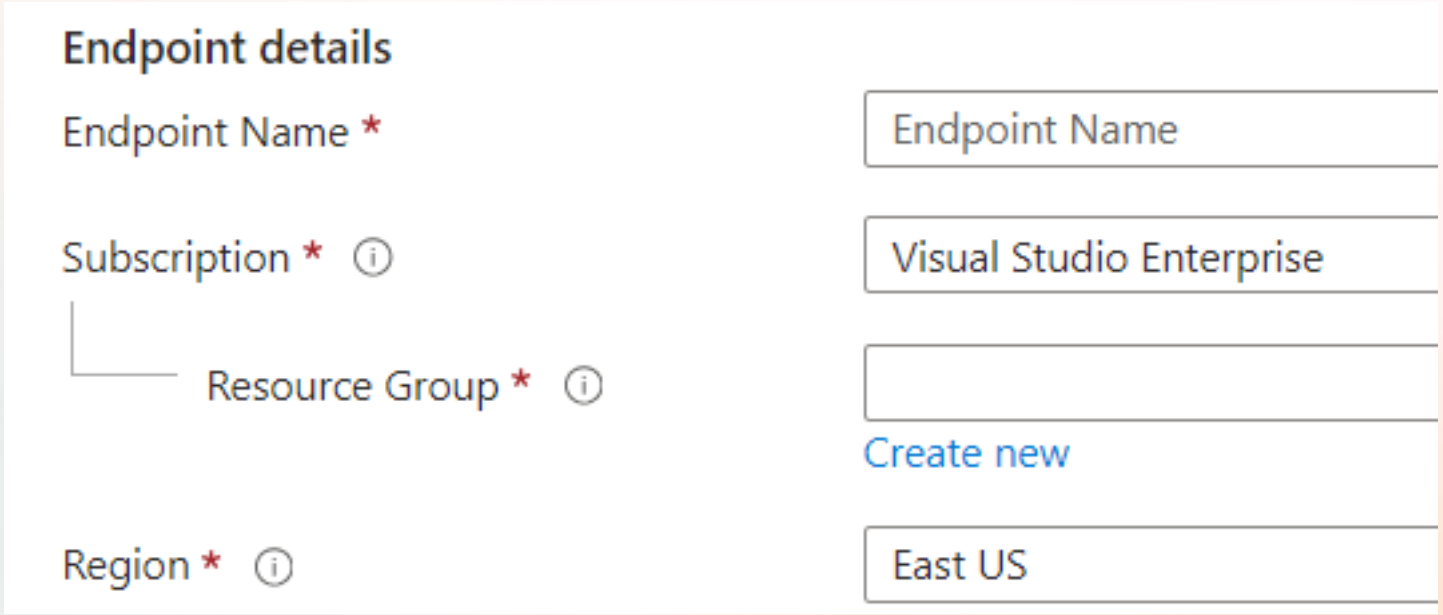
2. Give the app registration access to the DCR

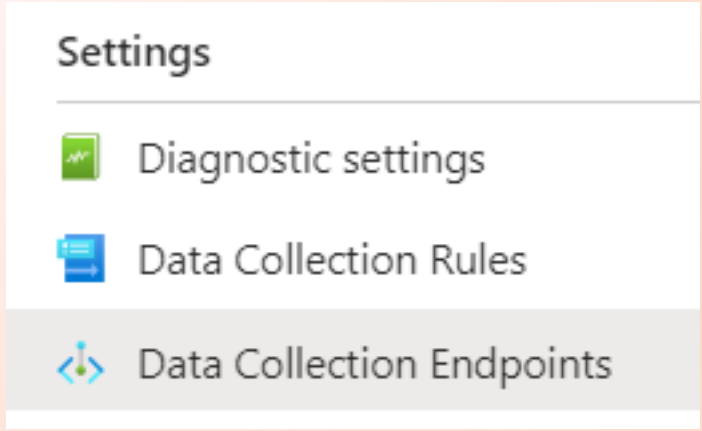




2nd step: Data Collection Endpoint

- Gateway between endpoint and Azure Log Ingestion Pipeline
- DCE: connection used by Logs ingestion API to send collected data
- A DCE **can support multiple DCRs**





Last step: Data Collection Rule

- DCR is here to define how and where the data should be stored

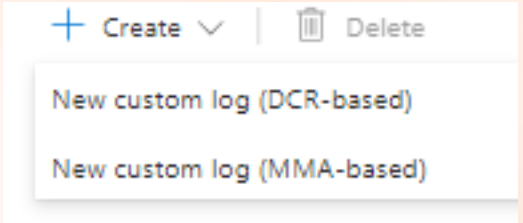
- Determines what to do with your data:
  - ❖ Which data should be collected
  - ❖ How to transform that data
  - ❖ Where to send data

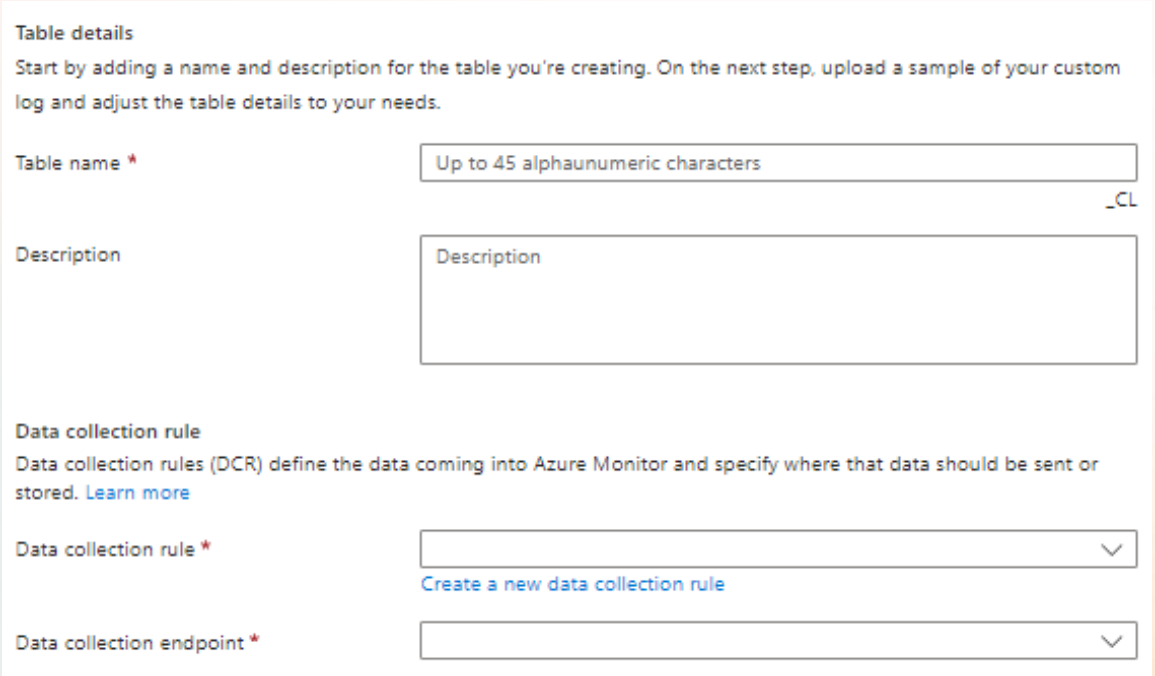
- MS recommends to have **one DCR by type of data**
- Give the app registration access to the DCR

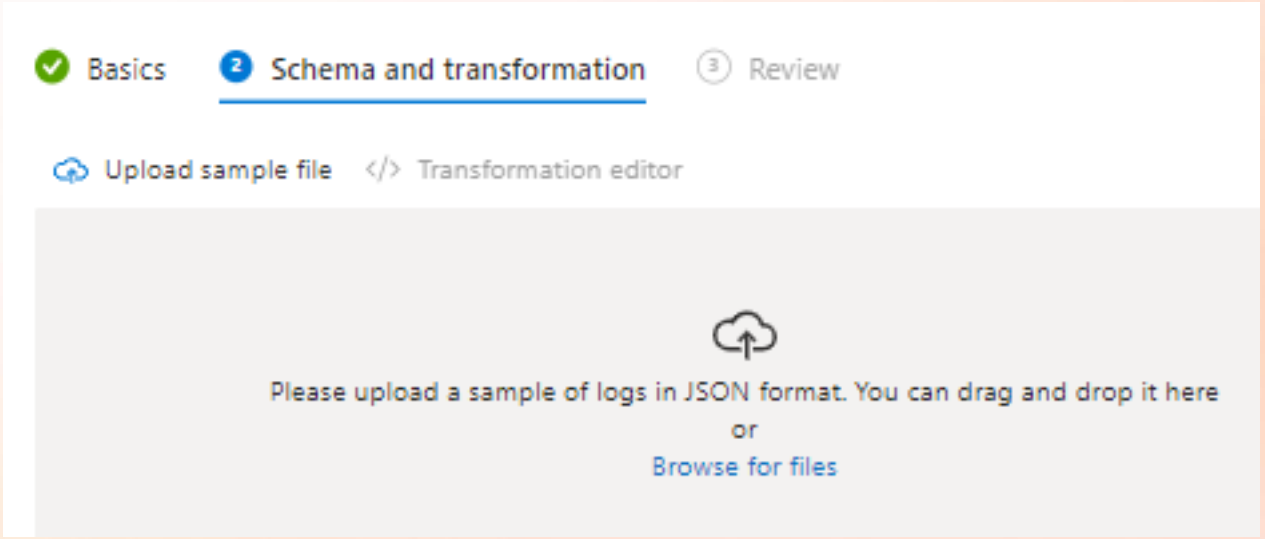
Creating the DCR

Creating a table (custom log)

- The table/custom log **should be created first**
- Go to **Tables** > **Create** > **Custom log (DCR-based)**
- Specify structure of the table by importing JSON



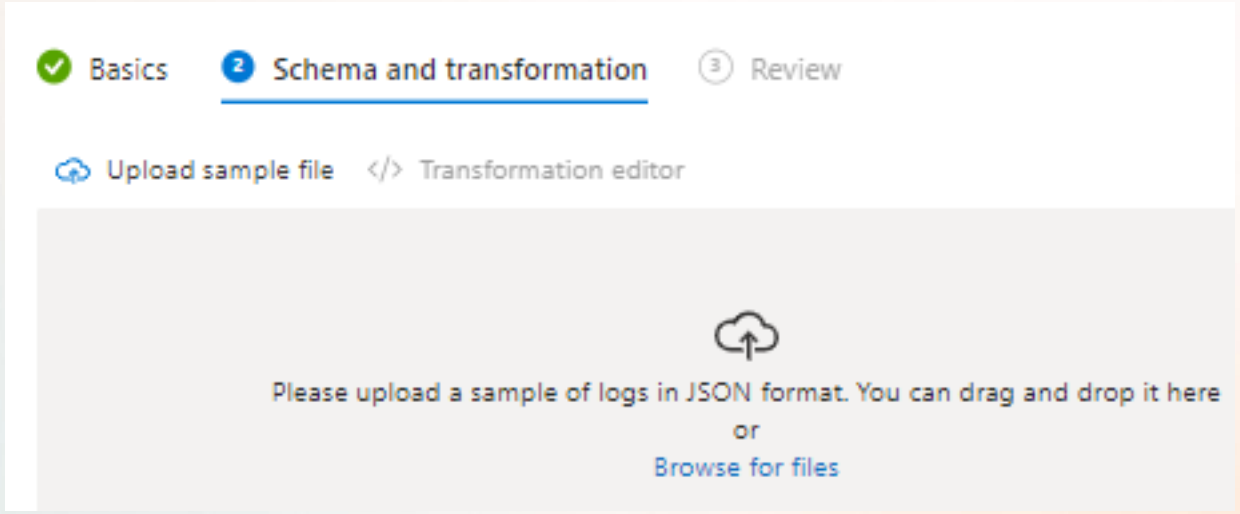




Creating the DCR

Creating a table (custom log)

- The table/custom log **should be created first**
- Go to **Tables** > **Create** > **Custom log (DCR-based)**
- Specify structure of the table by importing JSON



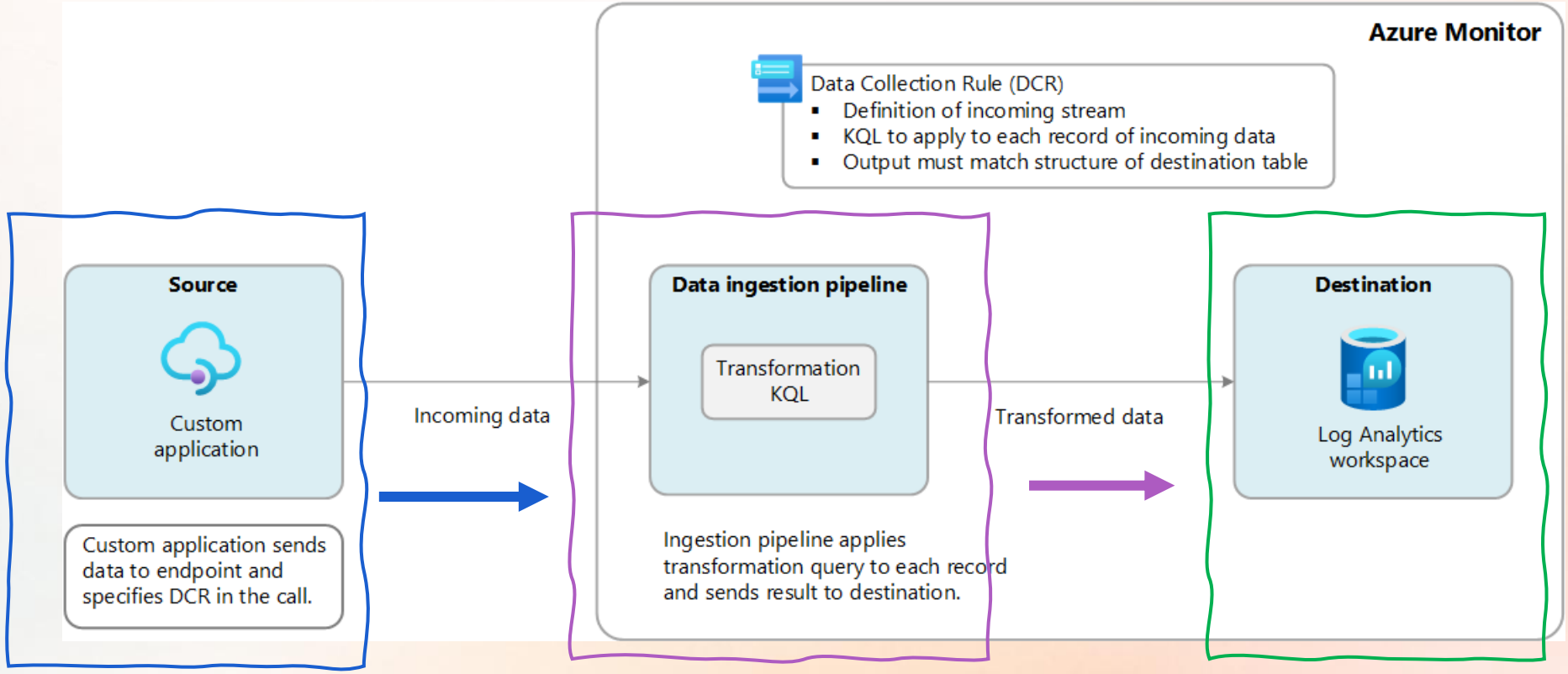

```
[
    {
        "TimeGenerated":  "vendredi 02/09/2024 16:53 +01:00",
        "DeviceName":  "P64000168",
        "ModelFriendlyName":  "ThinkPad T480s",
        "DeviceManufacturer":  "LENOVO",
        "Model":  "20L8SD1100",
        "DriverName":  "System Interface Foundation V2 Device",
        "DriverVersion":  "1.2.0.11",
        "DriverDate":  null,
        "DeviceClass":  "SYSTEM",
        "DeviceID":  "ROOT\\SYSTEM\\0002",
        "manufacturer":  "Lenovo",
        "InfName":  "oem187.inf",
        "Location":  null
    }
]
```

Log Ingestion process

- API sends data to DCE/DCR (through a script)

!! Data sent to the API must be formatted in JSON

- DCR transforms incoming data to match to the target table
- DCR sends transformed data to the custom log



Demo 1

What will we see ?

- Creating a DCE
- Creating a table (custom log) & DCR

KQL:
Kusto Query Language

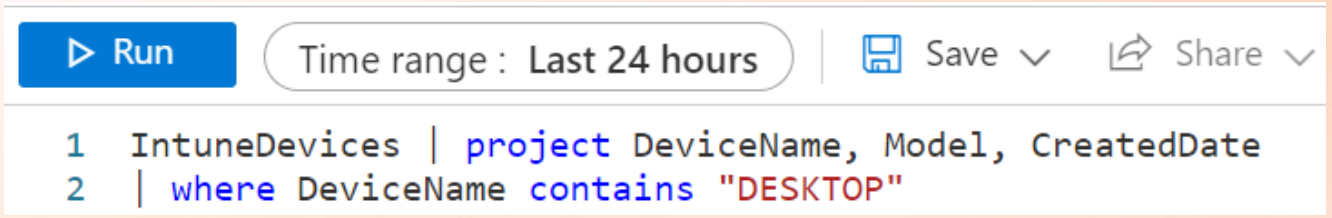
KQL: the heart of your analysis

- KQL for **Kusto Query Language**
- Language to request/explore data from your Logs
- Same language than in CMPivot
- Used in Log Analytics, Sentinel, MDE…
- Now in Intune device query
- You can use Copilot

Starting with KQL

1. Select a log like **IntuneDevices**
2. Select a **time range**
3. Use pipe | delimiter with **where** operator (there are essentials)

- | **where** : allows you to filter on a field (***contains***, ***==***, ***startswith…***)
- | **project** operator : allow you to select fields (columns) to display
- isnotempty(column), | join, | order, | count, | top, ago(delay)



KQL cheat sheets

An easy way to start with KQL

https://techcommunity.microsoft.com/t5/azure-data-explorer-blog/azure-data-explorer-kql-cheat-sheets/ba-p/1057404

Custom reporting
&
Log Analytics

How does it work ?

Scheduling the script

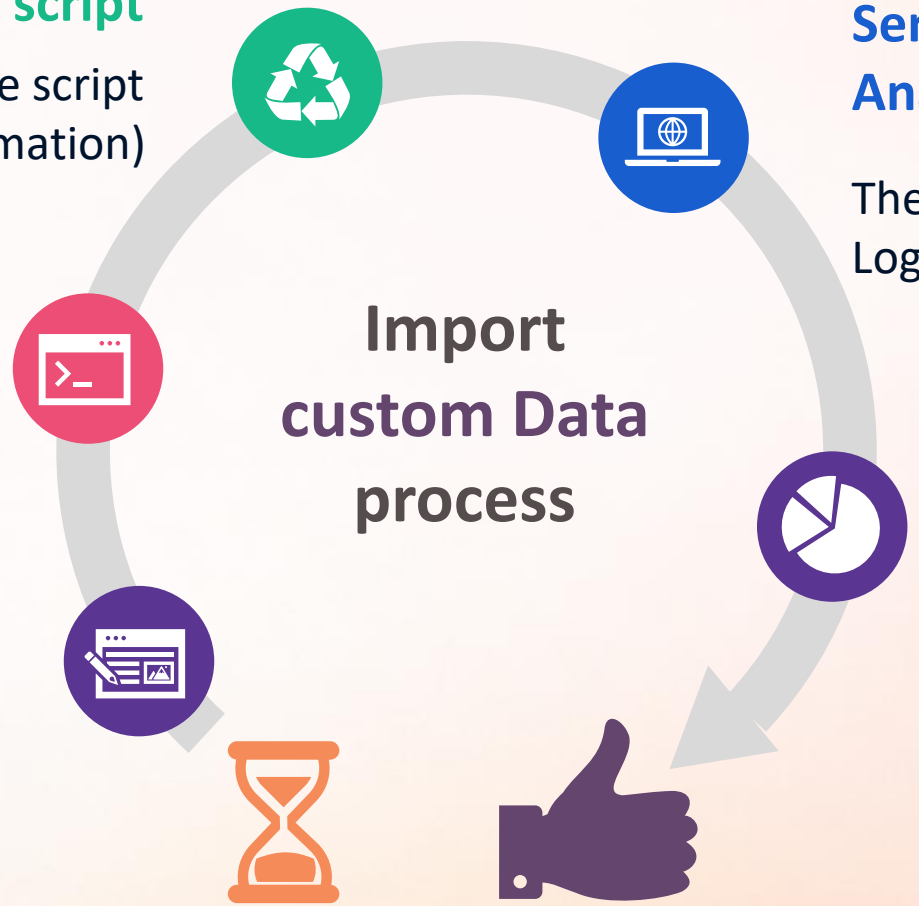
Add & schedule the script
(Remediation or Azure Automation)

Creating a script to get data

Create a script to get data you want from devices or tenant

Add prerequisites

App registration, DCE, table, DCR



Sending data to Log Analytics through API

The script will send them into a Custom Log in Log Analytics (***through the DCR***)

Gather data in a report

Once data are in the Custom Log, we can create a workbook with them

Sending data: from where ?

1. **Remediation script:** to import data from devices
   - Drivers inventory
   - Windows Secure boot cert expiration

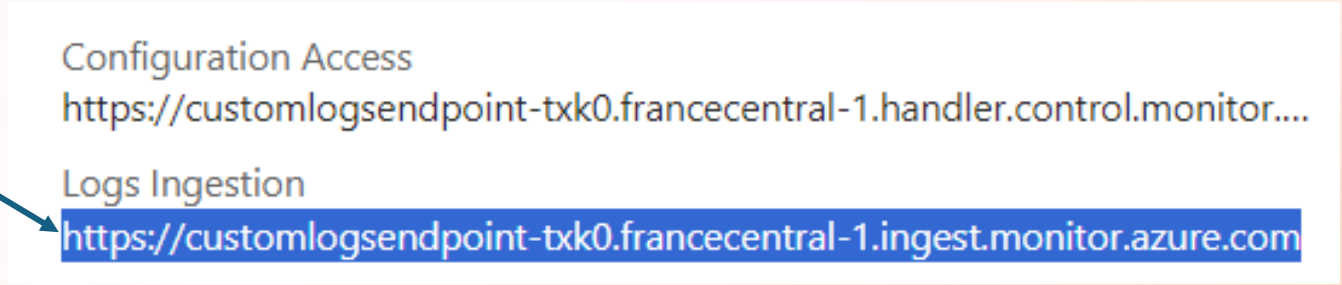
2. **Azure Automation runbook:** send data from your environment
   - BSOD report
   - Lenovo BIOS and warranty dashboards
   - Discovered apps dashboard

Prerequisites for sending data

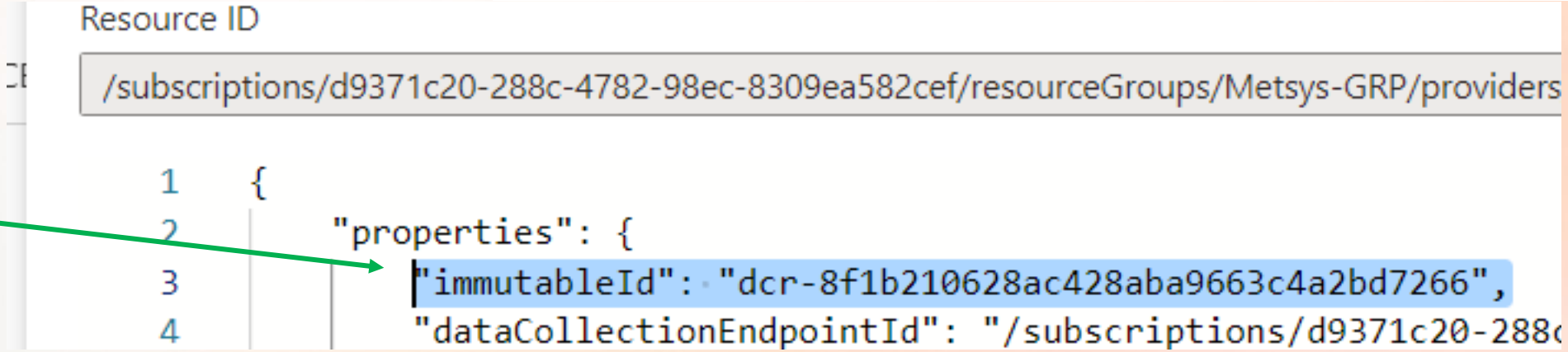
Prerequisites you need in your scripts:

1. Get the DCE URI (in Monitor > Data Collection Endpoint)
2. Get the DCR **immutableId** (in Monitor > Data Collection Rule)
3. Specify custom log where to send data

DCE



DCR



From remediation script

1. Add prerequisites in variables (DCE, DCR, custom log)
2. Add app registration info (tenant id, app id, secret)
3. Get data you want (drivers, custom info…)

```
$Properties = [Ordered] @{
    "TimeGenerated" = Get-Date ([datetime]::UtcNow) -Format O
    "DeviceName" = "LAPTOP1"
    "ModelFriendlyName" = "ThinkPad T480s"
    "DeviceManufacturer" = "Lenovo"
    "Model" = "20L8"
}
$Custom_Infos = New-Object -TypeName "PSObject" -Property $Properties
```

4. Convert data to JSON

```
$Body_JSON = $Custom_Infos | ConvertTo-Json -AsArray;
```

From remediation script

5. Authenticate and get a token through the app registration

```
# Getting token through the azure app from Azure monitor pipeline
$scope = [System.Web.HttpUtility]::UrlEncode("https://monitor.azure.com//.default")
$body = "client_id=$appId&scope=$scope&client_secret=$appSecret&grant_type=client_credentials";
$headers = @{"Content-Type" = "application/x-www-form-urlencoded" };
$uri = "https://login.microsoftonline.com/$tenantId/oauth2/v2.0/token"
$bearerToken = (Invoke-RestMethod -Uri $uri -Method "Post" -Body $body -Headers $headers).access_token
```

6. Send data to Log Analytics using Log Ingestion API

```
# Sending data to Log Analytics Custom Log
$headers = @{"Authorization" = "Bearer $bearerToken"; "Content-Type" = "application/json" };
$uri = "$DceURI/dataCollectionRules/$DcrImmutableId/streams/Custom-$Table"+"?api-version=2023-01-01";
$body = $Custom_Infos | ConvertTo-Json -AsArray;
$uploadResponse = Invoke-RestMethod -Uri $uri -Method "Post" -Body $body -Headers $headers;
```

From Azure Automation

1. Create Azure Automation with a managed identity
2. Grant permission on the DCR to the managed identity (MI)
3. Create a Runbook (script) in Azure Automation
4. Specify prerequisites in variables (DCE, DCR, custom log)
5. Authenticate against the Log Ingestion API with the MI and get a token

5

```
Connect-AzAccount -Identity | Out-Null
$bearerToken = (Get-AzAccessToken -ResourceUrl "https://monitor.azure.com//.default").Token
```

From Azure Automation

6. Collect data you want & convert data to JSON

```
$Properties = [Ordered] @{
    "TimeGenerated" = Get-Date ([datetime]::UtcNow) -Format O
    "DeviceName" = "LAPTOP2"
    "ModelFriendlyName" = "T14s Gen4"
    "DeviceManufacturer" = "Lenovo"
    "Model" = "21F7"
}
$Custom_Infos = New-Object -TypeName "PSObject" -Property $Properties
$body = $Custom_Infos | ConvertTo-Json -AsArray;
```

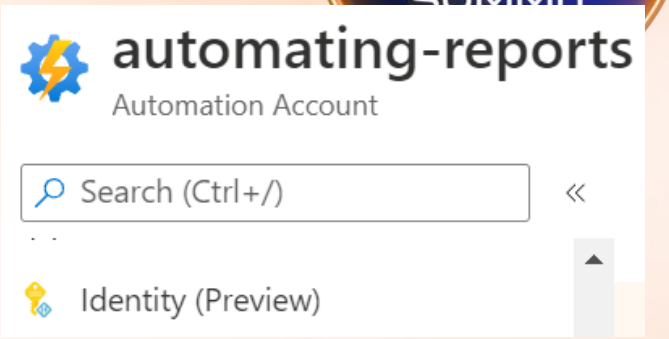
7. Send data using Log Ingestion API

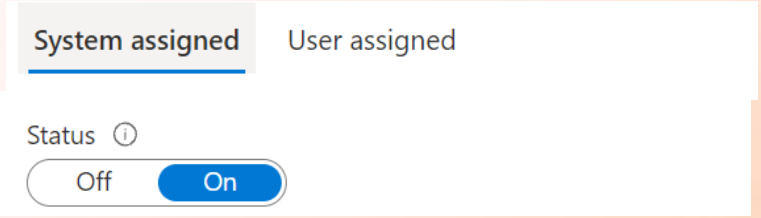
```
# Sending data to Log Analytics Custom Log
$headers = @{"Authorization" = "Bearer $bearerToken"; "Content-Type" = "application/json" };
$uri = "$DceURI/dataCollectionRules/$DcrImmutableId/streams/Custom-$Table"+"?api-version=2023-01-01";
$uploadResponse = Invoke-RestMethod -Uri $uri -Method "Post" -Body $body -Headers $headers;
```

Managed identity: what is it ?

What is a managed identity ?

- Accounts in Entra ID
- Goal: Access to Entra ID resources without credentials
- Runbook/Script use the MI to get token
- Two kinds of MI (System or User)





Pros ?

- You don’t need to manage credentials
- Credentials are never exposed in the code
- Can be used without any additional cost

Here are some of the benefits of using managed identities:

- You don't need to manage credentials. Credentials aren’t even accessible to you.
- You can use managed identities to authenticate to any resource that supports Azure AD authentication, including your own applications.
- Managed identities can be used without any additional cost.

How to ?

- In **Automation** > **Identities**, enable **System assigned**

System vs User

	System-assigned	User-assigned
Creation*	Created when MI is enabled	Separately and should be created first
Scope	Can be used only by one resource*	Can be used across multiple resources*
Lifecycle	Bind to the resource	Independant of the resource
Persistence	MI is deleted with the resource	MI persists if resource is deleted

MI = Managed Identity

***A corresponding service principal (enterprise application) is generated in Entra ID**
*** Resource can be Azure Automation, Logic App, Function app**

Managed identity authentication

Module name	Cmdlet to use
Az.Accounts	Connect-AzAccount -Identity
Microsoft.Graph.Authentication	Connect-MgGraph -Identity

What about permissions ?

- Permissions are **applied on the managed identity**
- Can not be added through the Azure portal, **only with PowerShell**

```
# Your tenant id (in Azure Portal, under Azure Active Directory -> Overview )
$TenantID=""
# Name of the manage identity or enterprise application
$DisplayNameOfMSI=""
# Permission to set to the managed identity
$Permissions = @('DeviceManagementManagedDevices.Read.All')
# Check if module is installed and if not install it
If(!(Get-Installedmodule Microsoft.Graph.Applications)){
Install-Module Microsoft.Graph.Applications}Else{Import-Module Microsoft.Graph.Applications}
# Authenticate
Connect-MgGraph -Scopes Application.Read.All, AppRoleAssignment.ReadWrite.All, RoleManagement.ReadWrite.Directory -TenantId $TenantID
# Get info about the managed identity
$MSI = Get-MgServicePrincipal -Filter "displayName eq '$DisplayNameOfMSI'"
$API = Get-MgServicePrincipal -Filter "displayName eq 'Microsoft Graph'"
# Check permissions to add
$AppRoles = $API.AppRoles | Where-Object {($_.Value -in $Permissions) -and ($_.AllowedMemberTypes -contains "Application")}
# Set permissions
ForEach($Role in $AppRoles){`
New-MgServicePrincipalAppRoleAssignment -ServicePrincipalId $MSI.Id -PrincipalId $MSI.Id -AppRoleId $Role.Id -ResourceId $API.Id`
}
```

Demo 1

What will we see ?

- Send data through remediation script
- Configure Azure Automation
- Adding permissions on the MI
- Send data through Azure Automation

Make it a bit more secure without the app registration

How to proceed ?

We will use an intermediate step between the device and Log Analytics.

Two ways for that:

1. **Azure Automation**

2. **Azure function app**

The new process

Scheduling the script

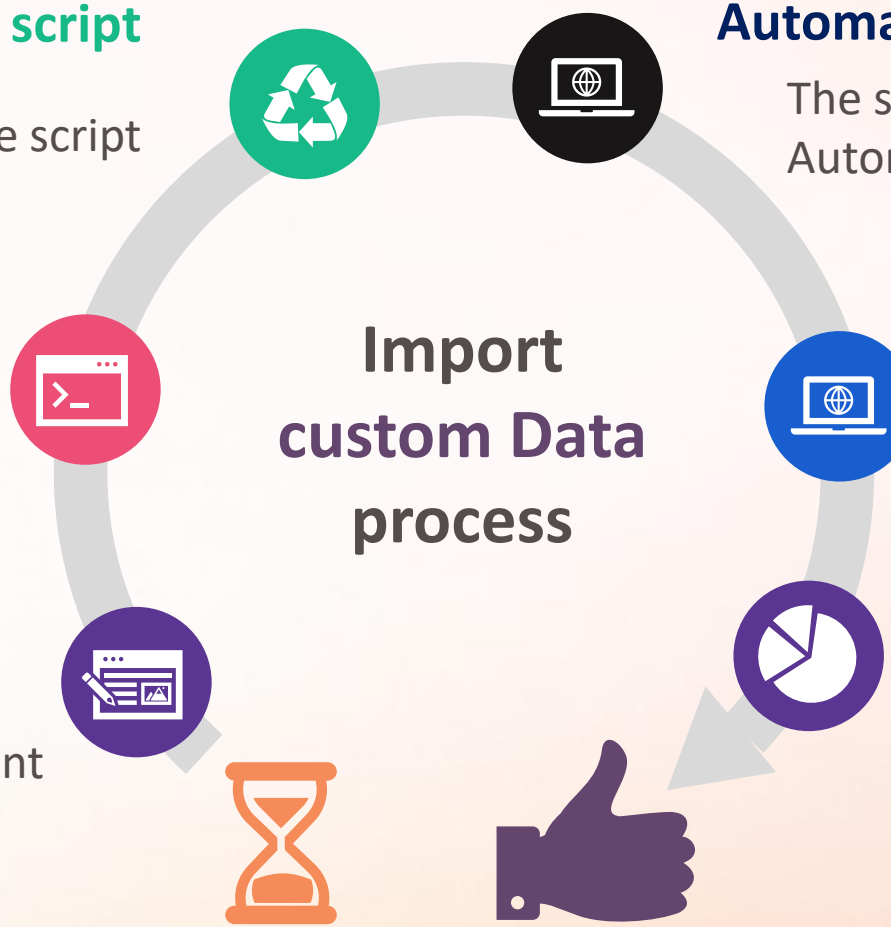
Add & schedule the script

Sending data to Azure Automation or function

The script will send data to Azure Automation or Azure function app

Creating a script to get data

Create a script to collect data



Azure Automation or function send data to LA

Azure Automation or function app will send data sent by devices into a Custom Log in Log Analytics

Add prerequisites

Use Azure Automation with MI Grant access on the DCR to the MI

Gather data in a report

Once data are in the Custom Log, we can create a workbook with them

With Azure Automation

From the device side

1. Add a webhook to your runbook first
2. Collect data to send & transform data to JSON
3. Call the runbook from device using the webhook:
   - **$header** contains password in **message**
   - **$webhookURI** contains URL of the webhook
   - Pass collected data (formated to JSON) through **$Body** parameter
   - Use **Invoke-WebRequest** cmdlet and **POST** method

```
$webhookURI = ""

$Properties = [Ordered] @{
"TimeGenerated" = Get-Date ([datetime]::UtcNow) -Format O
"DeviceName" = "LAPTOP3"
"ModelFriendlyName" = "ThinkPad T14s Gen3"
"DeviceManufacturer" = "Lenovo"
"Model" = "21BS"
}

$Custom_Infos = New-Object -TypeName "PSObject" -Property $Properties
$Body = $Custom_Infos | ConvertTo-Json -AsArray;
$Secure_header = @{message='Iam_a_bit_more_secure'}
Invoke-WebRequest -Method Post -Uri $webhookURI -Body $Body -UseBasicParsing -Headers $Secure_header
```

With Azure Automation

From the Azure Automation side

1. Check the header provided by the script & compare password provided in **message**

```
If($WebhookData.RequestHeader.message -ne 'Iam_a_bit_more_secure')
    {
        "RequestHeader not valid"
        EXIT
    }
```

2. Check if the device is managed with Graph API

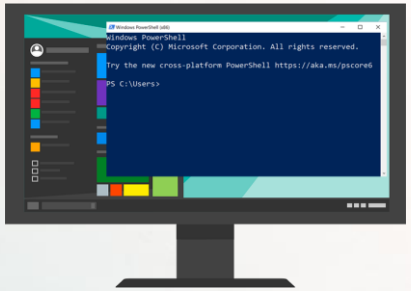
```
$URL = 'https://graph.microsoft.com/beta/deviceManagement/managedDevices?$filter' + "=contains(deviceName,'$DeviceName')"
$Get_Device = Invoke-WebRequest -Uri $URL -Method GET -Headers $Headers -UseBasicParsing
$Get_Device_json = ($Get_Device.Content | ConvertFrom-Json).value
If($Get_Device_json -ne $null) # "The device is allowed"
```

3. Send data to Log Analytics using Log Ingestion API and the managed identity

```
Add-Type -AssemblyName System.Web
$headers = @{"Authorization" = "Bearer $bearerToken"; "Content-Type" = "application/json" };
$uri = "$DceURI/dataCollectionRules/$DcrImmutableId/streams/Custom-$Table"+"?api-version=2023-01-01";
$uploadResponse = Invoke-RestMethod -Uri $uri -Method "Post" -Body $Inputs_JSON -Headers $headers;
```

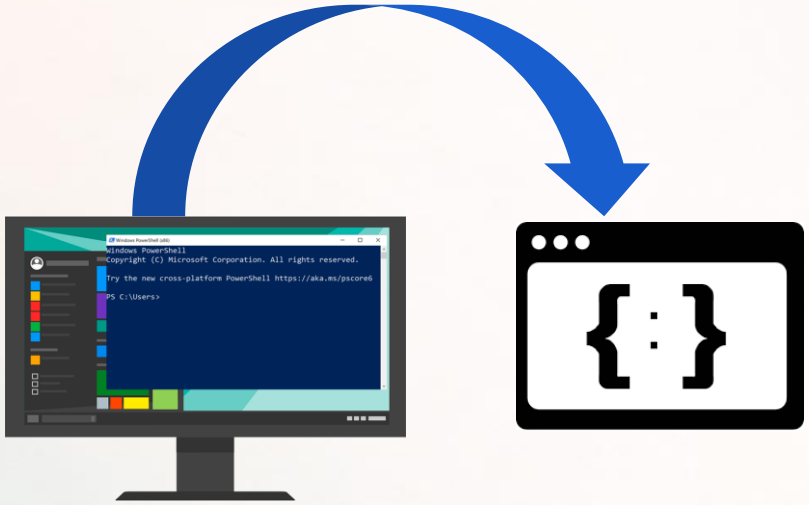
Main process

1. Script is executed on the device



Main process

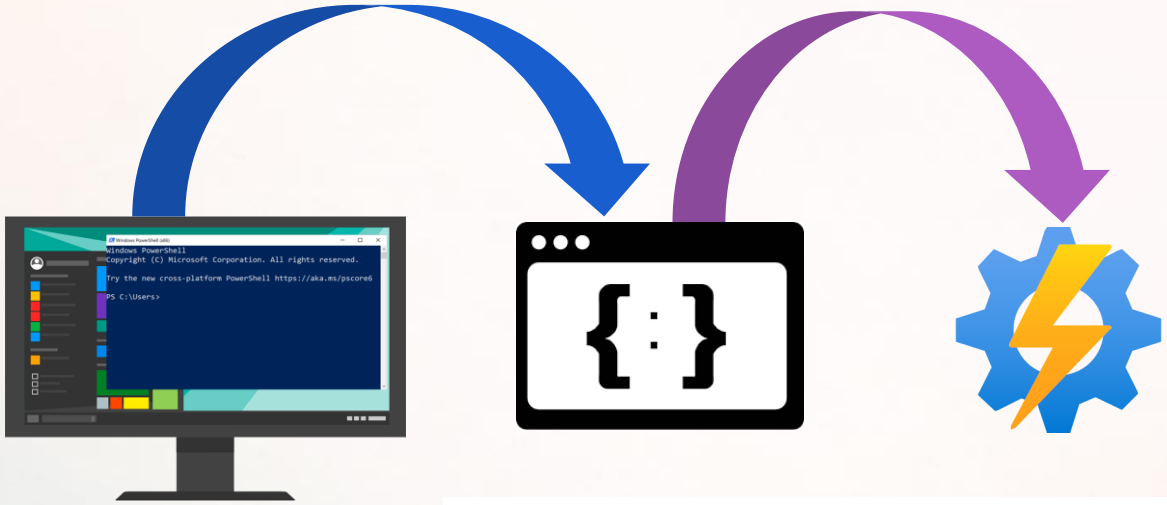
2. Password added in request header & device name sent as parameter



```
$params  = @{
DeviceName = "$env:COMPUTERNAME";
}
$Secure_header = @{message='Iam_a_bit_more_secure'}
$URI = ""
$body = ConvertTo-Json -InputObject $params
Invoke-WebRequest -Method Post -Uri $URI -Body $body -UseBasicParsing -Headers $Secure_header
```

Main process

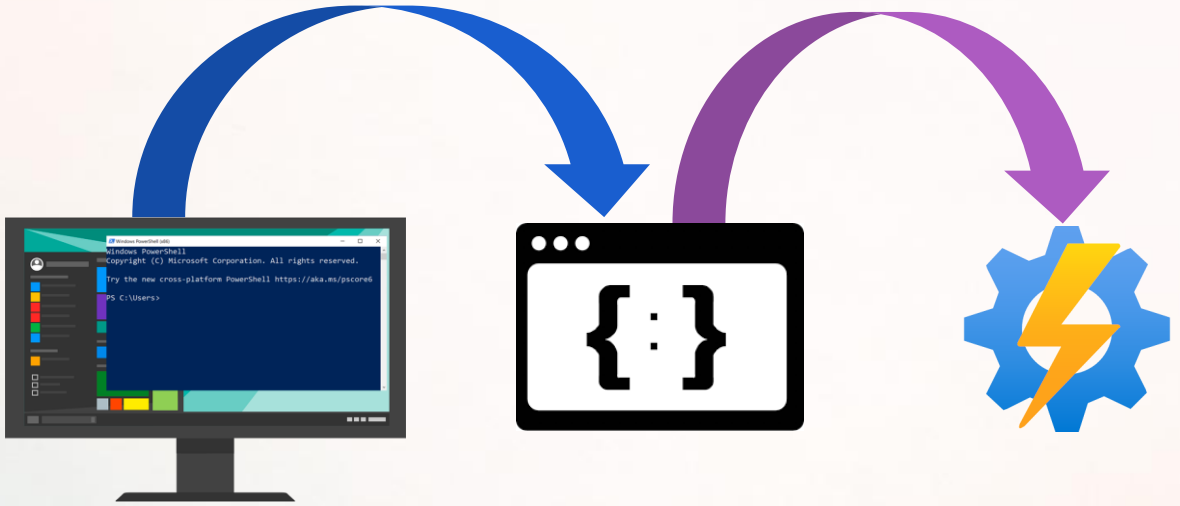
3. Runbook gets the script content

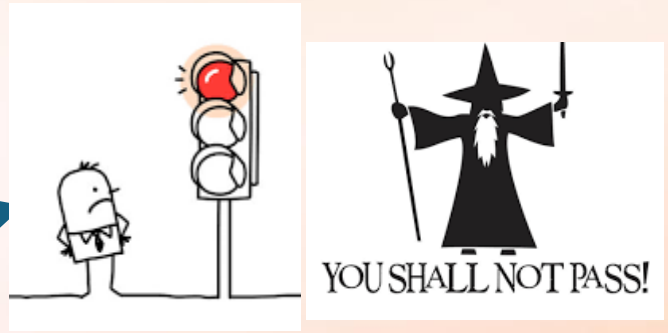


```powershell
param (
    [Parameter (Mandatory = $false)]
    [object] $WebHookData
)
$Inputs = ConvertFrom-Json $webhookdata.RequestBody
$DeviceName = $($Inputs[0].DeviceName)
$Inputs_JSON = $webhookdata.RequestBody
```

Main process

4. If header password is not OK ➔ **EXIT**

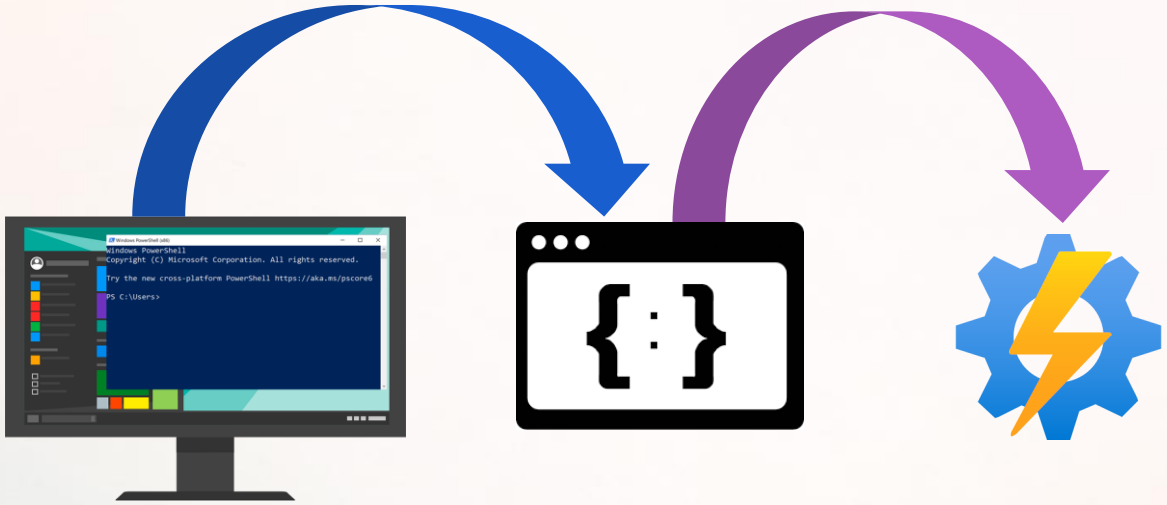


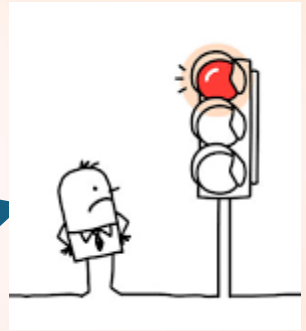



```
If($WebhookData.RequestHeader.message -ne 'Iam_a_bit_more_secure')
{
    "RequestHeader not valid"
    EXIT
}
```

Main process

4. If the device not found in Intune ➔ **EXIT**

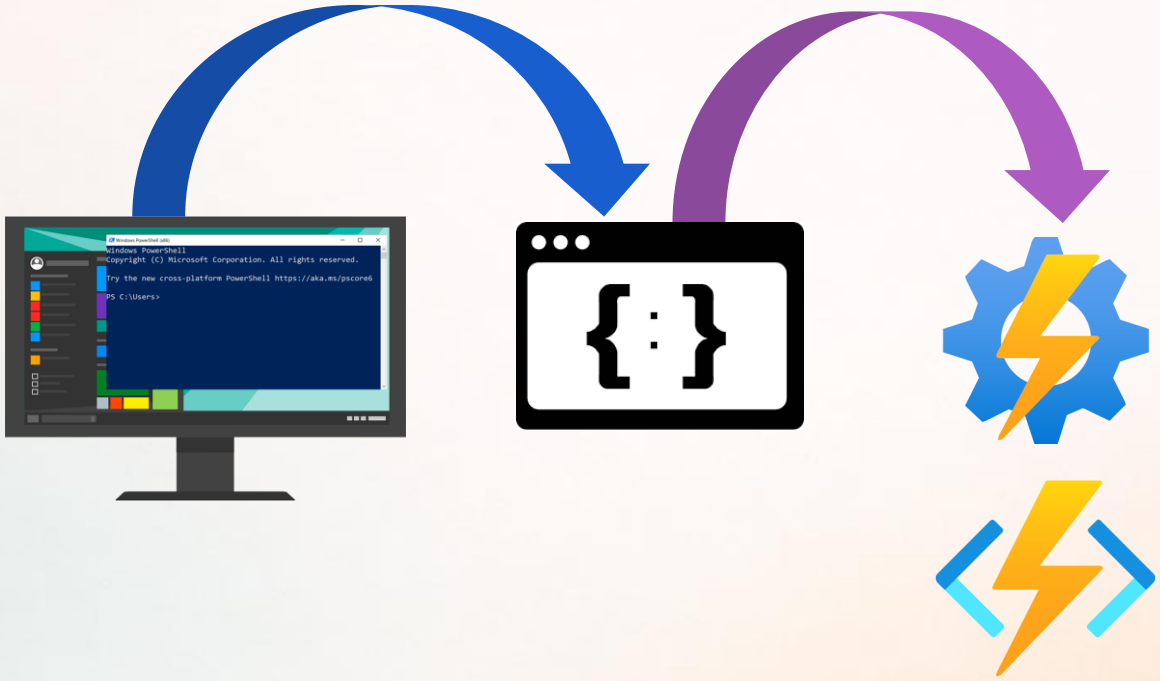


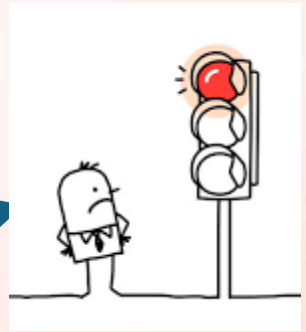



```powershell
$DeviceName = $($Inputs.DeviceName)
Connect-MgGraph -Identity
$Get_Device_Info = Get-MgDeviceManagementManagedDevice -Filter "contains(deviceName,'$DeviceName')"
if (-not $Get_Device_Info) {
    Write-Output "Device not found in managedDevices."
    EXIT
}
```

Main process

5. If everything is OK ➔ **Send data to Log Analytics**





Demo 1

What will we see ?

- Securing API calls
- Send data through script without secret

Please rate this session on Sched.com

We would love to hear what you liked and how we could improve!

Thanks!