Automate the Cloud: Azure automation & PowerShell in action

Damien Van Robaeys
@syst_and_deploy

Sponsors

onedeploy
Windows Deployment Reimagined

ZERO TOUCH

Insight

control up

PATCH MY PC

robopack

numecent

Microsoft

eido

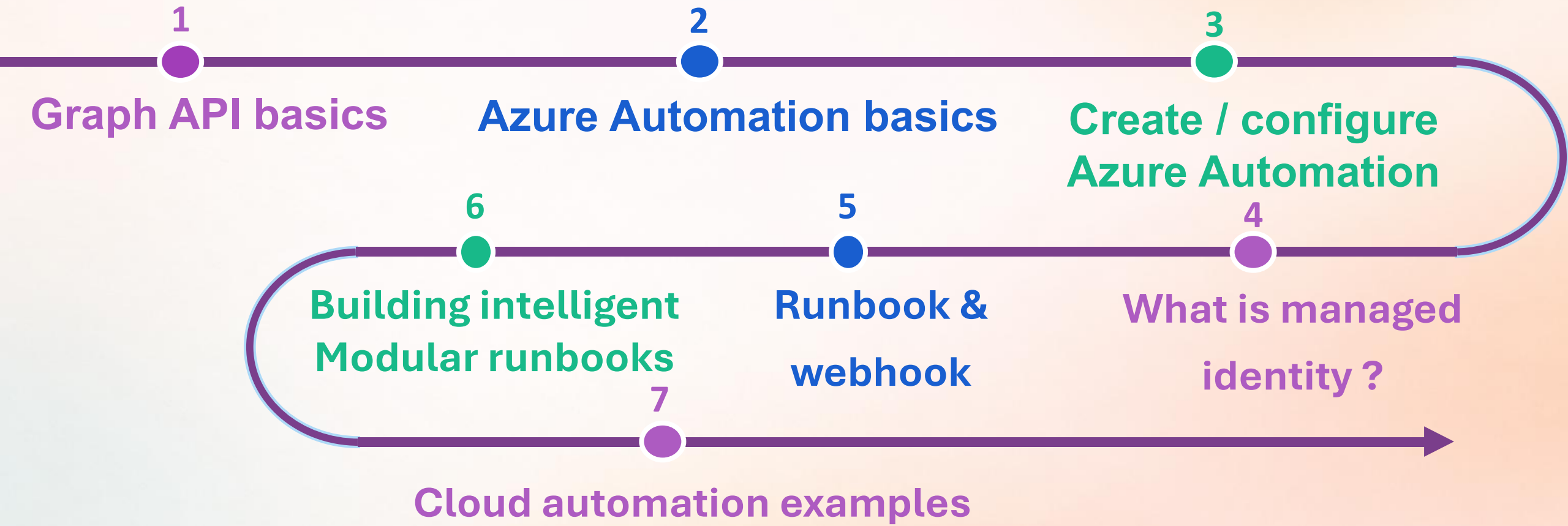
Rimo3

Overview

Who use Graph API ?

Who use Azure Automation ?

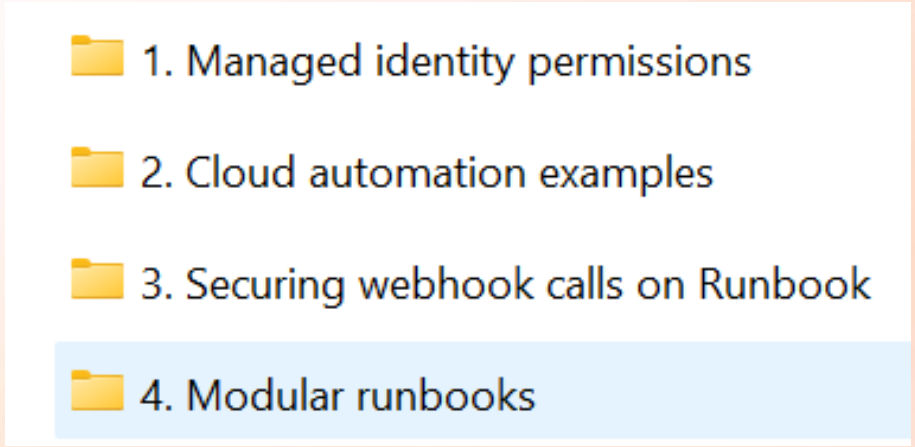
What will you learn ?



Slides & demos

Slides and demos are already available

https://github.com/damienvanrobaeys/Events_Slides



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

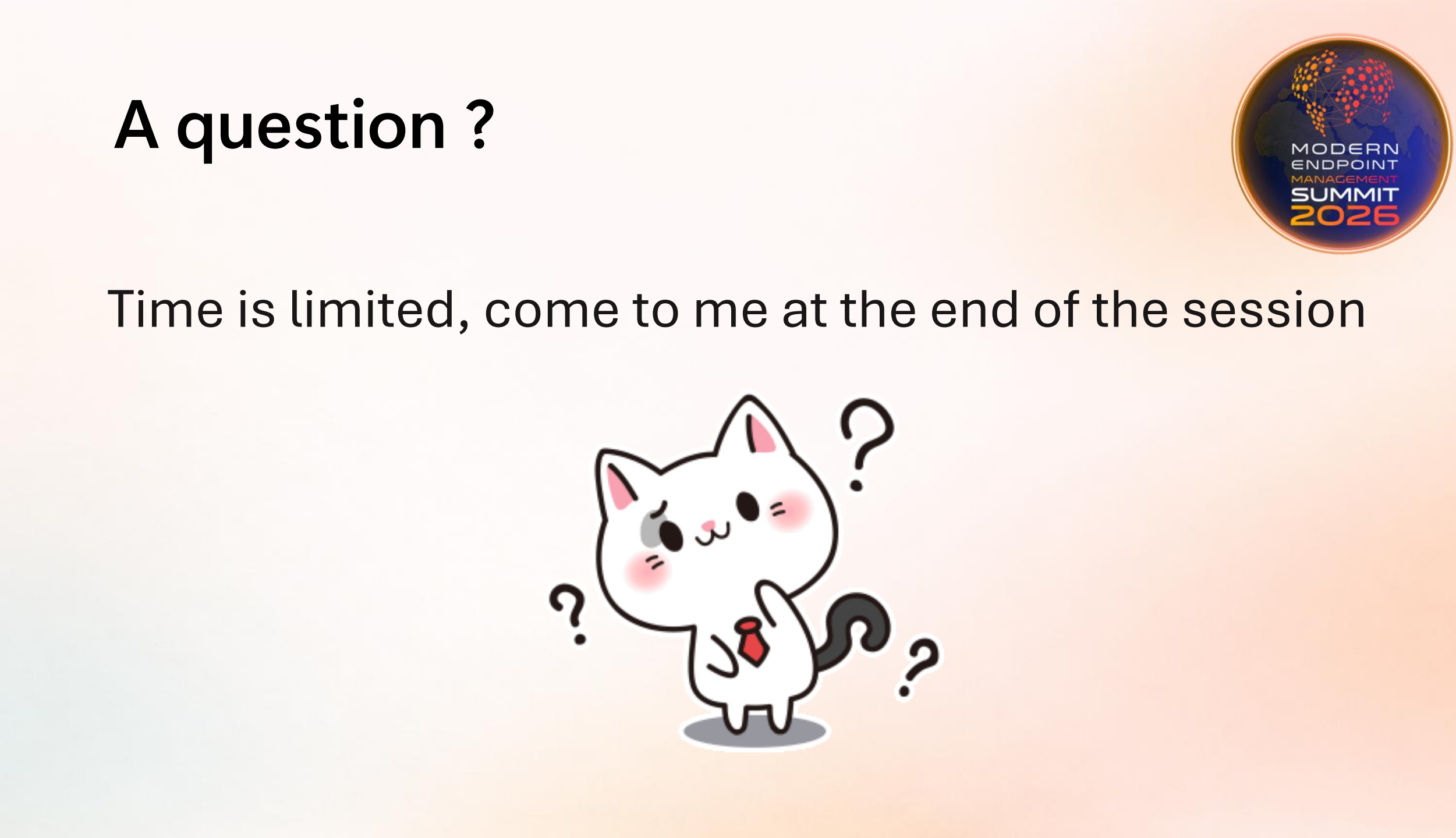
Damien Van Robaeys

Microsoft Most Valuable Professional Award

METSYS

systanddeploy.com

A question ?



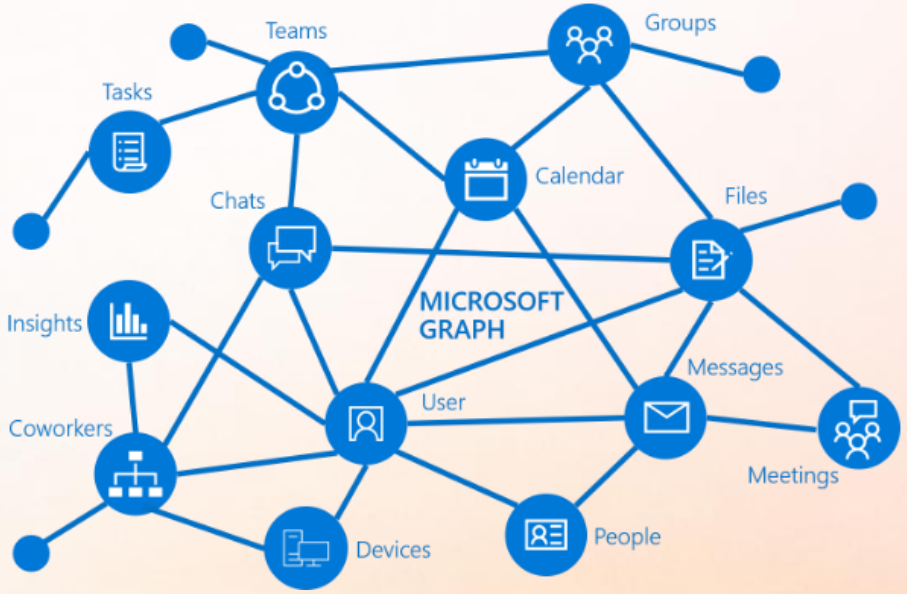
Time is limited, come to me at the end of the session

Graph API basics

What is it ?

- REST API allowing you to access to Microsoft cloud services

- All services are interconnected, and Graph is here to manage all of them
- Provides access to resources from the M365 ecosystem (Users, groups, devices…)
- Graph link: https://graph.microsoft.com



Graph API versions

2 versions of Graph API: Beta and 1.0

- 1.0: Generally Available and ready for production use
- Beta: for managing resources in Preview
- 1.O ➔ https://graph.microsoft.com/1.0/
- Beta ➔ https://graph.microsoft.com/beta/

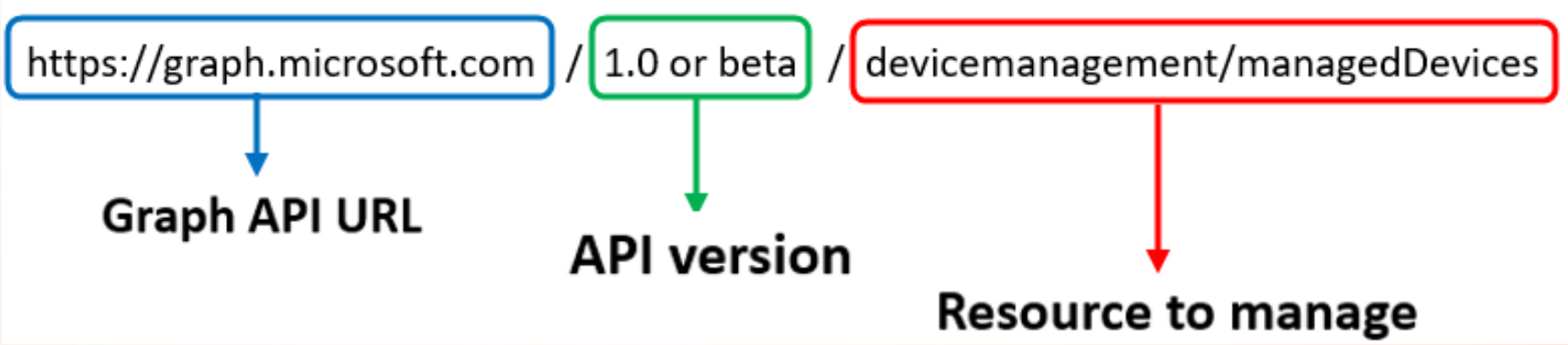
- Preview features are generally pushed first in Beta version

Beta API is subject to change

Graph query: structure

- Structure: **MS Graph link** + **API version** + **Resource to manage**

  https://graph.microsoft.com/API Version/Resource



- For a specific item (device, user…) ➔ add item ID to the query

  https://graph.microsoft.com/1.0/devicemanagement/managedDevices/**mydeviceID**

What is a resource ?

- Component in Intune, Entra ID…
- Can be managed trough the portal (and of course Graph API)
- Most known resources: devices, users, applications, remediation script

https://graph.microsoft.com/API Version/Resource URI

Resource	Resource URI
Devices	/deviceManagement/managedDevices
Applications	/deviceAppManagement/mobileApps
Scripts	/deviceManagement/deviceManagementScripts
Remediation scripts	/deviceManagement/deviceHealthScripts

What is a method ?

- Available methods : GET, POST, PATCH, DELETE

- **GET / DELETE**: Don’t require a body
- **POST**: create a resource or do an action
- **PATCH**: update a resource

- For **DELETE** and **PATCH**, add ID of the resource to manage

Method	Description
GET	Read data from a resource.
POST	Create a new resource, or perform an action.
PATCH	Update a resource with new values.
PUT	Replace a resource with a new one.
DELETE	Remove a resource.

Graph API & PowerShell

How to proceed ?

Microsoft Graph SDK module from Microsoft available on PowerShell Gallery

- **Microsoft.Graph.Authentication** : for authentication
- **Microsoft.Graph.DeviceManagement** : for managing Intune devices

How to install it ?

- Install-Module Microsoft.Graph

How to authenticate ?

- **Interactive**: Connect-MgGraph
- **Certificate**: Connect-MgGraph -Certificate $Cert -TenantId $TenantId -ClientId $ClientId
- **App registration**: Connect-MgGraph -TenantId $tenantID -ClientSecretCredential $Secret
- **Managed identity**: Connect-MgGraph -Identity

Permissions

- Allow you to do an action on a resource
- Each resource require specific permissions

Permission type	Permissions (from most to least privileged)
Delegated (work or school account)	DeviceManagementManagedDevices.ReadWrite.All, DeviceManagementManagedDevices.Read.All
Delegated (personal Microsoft account)	Not supported.
Application	DeviceManagementManagedDevices.ReadWrite.All, DeviceManagementManagedDevices.Read.All

- Find-MgGraphCommand -Uri "/deviceManagement/managedDevices"

```
Command                                 Permissions
-------                                 -----------
Get-MgDeviceManagementManagedDevice     {DeviceManagementManagedDevices.Read.All, Devi
New-MgDeviceManagementManagedDevice     {DeviceManagementManagedDevices.ReadWrite.All,
```

```
CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Find-MgGraphCommand                                2.34.0     Microsoft.Graph.Authentication
```

Authentication and token

- **Authentication methods available:**
  - Interactive: by typing user credentials
  - Providing a secret
  - Using a certificate
  - Managed identity (under Azure context)

App registration (Providing a secret, Using a certificate)

For automation (Providing a secret, Using a certificate, Managed identity (under Azure context))

- Authentication methods will help us to get a token

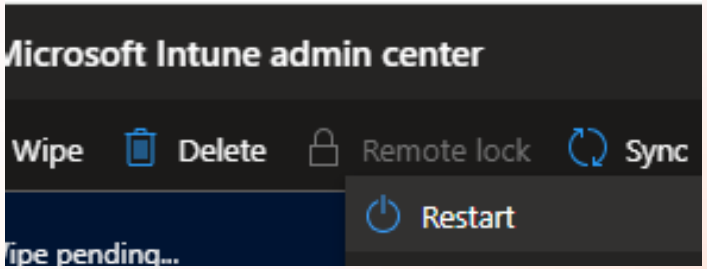
What is a token ?

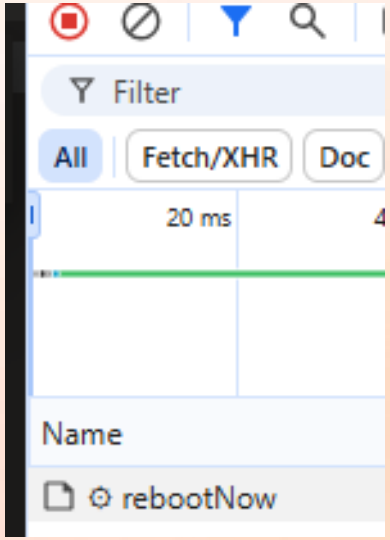
- **The token helps us to:**
  - Prove the user identity
  - Prove that the caller has the proper permissions to perform an operation

Finding the good query with a browser

A quick way to find the Graph equivalent from the Intune portal

1. Use Google Chrome, Edge Chromium…
2. Go to the **Intune portal**
3. Open the **Developer mode** with **F12**
4. Go to the **Network** tab
5. **Run your action** from the portal
6. Check the **Network** tab
7. Get the appropriate **URL** and **method**



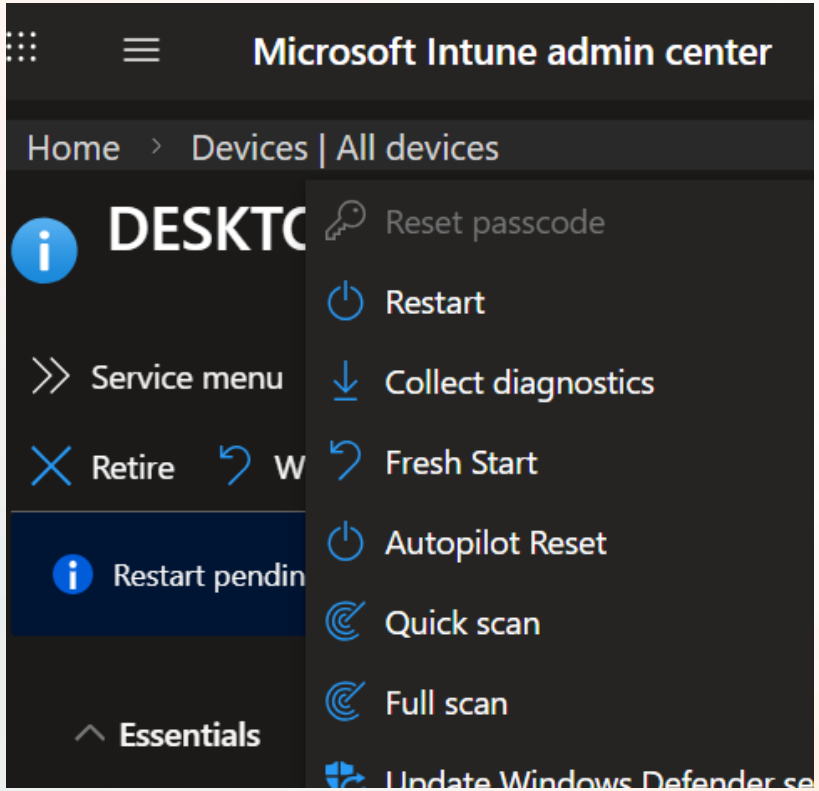


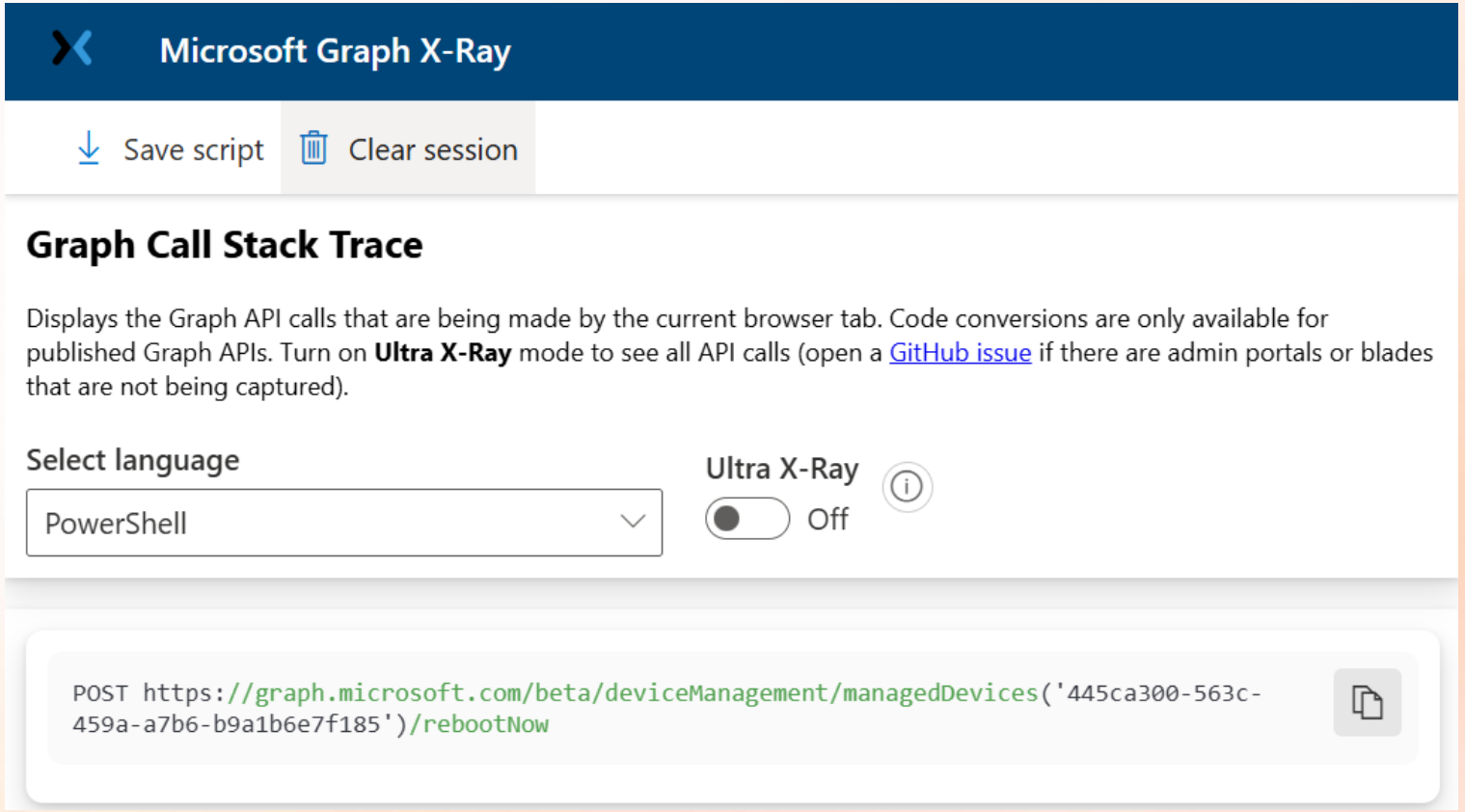
See my blog post there

Graph X-Ray

Browser add-on created by Merill Fernando: https://graphxray.merill.net

See Graph API calls behind the Clicks





Demo 1

What will we see ?

- Developer mode
- Graph Xray

What is Azure Automation ?

What is it ?

What is it ?

- A cloud-based automation service for automating your cloud tasks
- Designed to run and schedule scripts (runbooks)
- Script terminology = runbook
- A bit like a scheduled task on the cloud
- Manage both cloud or on-prem resources (hyrbrid worker)

For what ?

- For system management
- For repetitive administrative tasks
- For hybrid cloud/on-premises environments
- For long tasks

What can you interact with ?

Intune / Entra ID

- Automatically clean up stale or inactive devices
- App registration secrets/certificates expiration alert
- Add devices to a group based on user attributes or group
- Purge unwanted members from a group
- Export Intune platform scripts and remediation scripts to SharePoint

SharePoint

- Download files / Upload generated data (CSV, JSON, logs)
- Interact with lists and document libraries

Microsoft Defender for Endpoint (MDE)

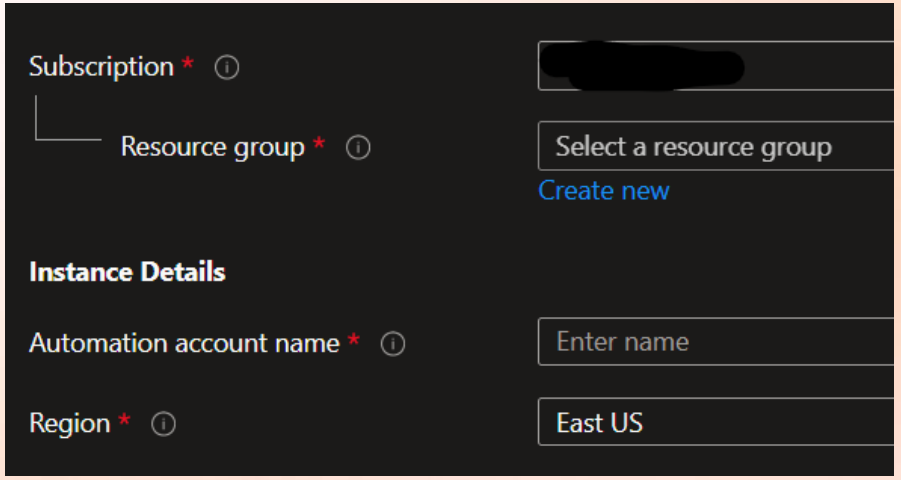
- Run KQL queries remotely

Log Analytics

- Export data to Log Analytics table for creating workbook

Prerequisites

- Azure account
- Resource group
- Automation account
- Permissions to manage automation account, runbooks
- Managed identity (for authentication)



Runbook triggers

Available triggers :

- **Manual**: Manually start from Azure automation
- **Scheduled**: runbook executed every x minutes, hours, days
- **On-demand**: runbook executed remotely through a webhook
- **Azure Monitor alert**: runbook is executed when alert is created

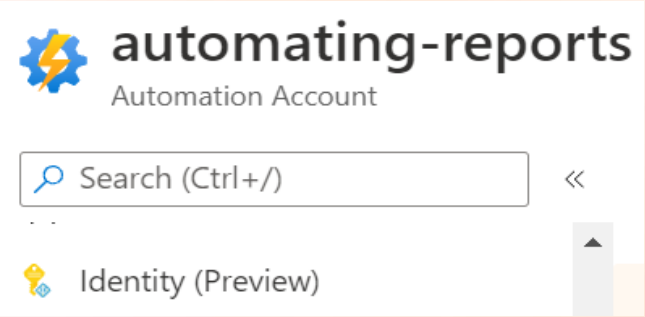
Managed identity, what is it ?

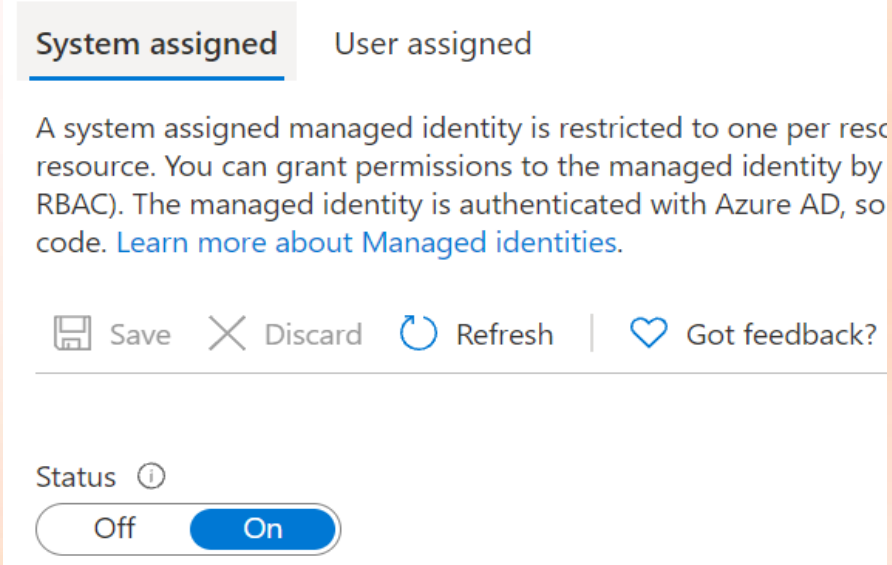
What is it and why to use it ?

Managed identity: the key for successful and secure authentication

What is a managed identity ?

- Account in Entra ID
- Goal: Access to Entra ID resources without credentials
- Runbook/Script use the MI to get token
- Two kinds of MI (see next slide)





Pros ?

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

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

Using modules in Azure Automation

For what ?

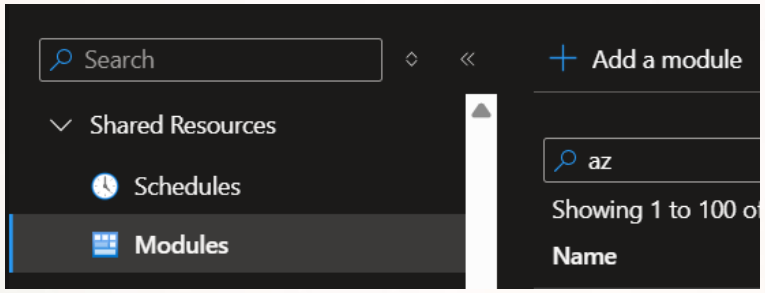
Makes runbooks more **powerful,** **reusable,** **easier to maintain**

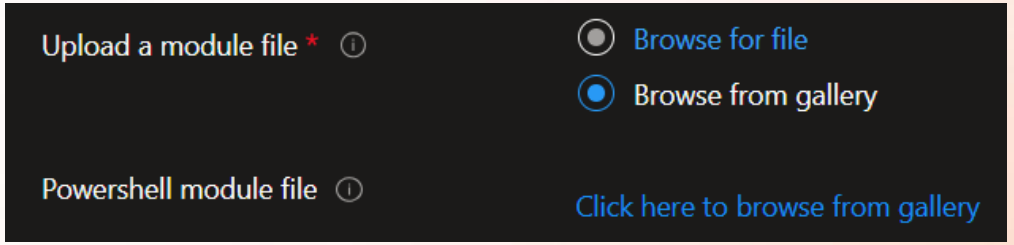
- **Functionality**: bring in additional cmdlets not available by default
- **Reusability**: Centralize functions, cmdlet and reuse code across multiple runbooks
- **Easier maintenance**: updating a module automatically updates all runbooks that depend on it

How to add modules ?

You can add/update PowerShell modules to use in your runbook

- In **Shared resources** > **Modules**
- Add modules > example **Az.Accounts**





- Managed identity authentication: more info here
- Run custom MgGraph query: more info here

Module rollback

Microsoft.Graph.DeviceManagement 2.33.0

Microsoft Graph PowerShell Cmdlets

Minimum PowerShell version
5.1

Installation Options

Install Module | Install PSResource | Azure Automation | Manual Download

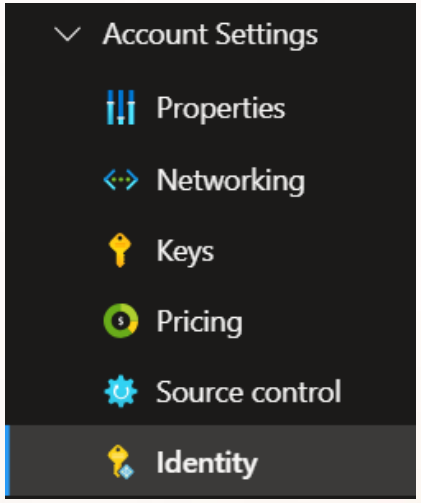
You can deploy this package directly to Azure Automation. Note that deploying packages with dependencies will deploy all the dependencies to Azure Automation. Learn More

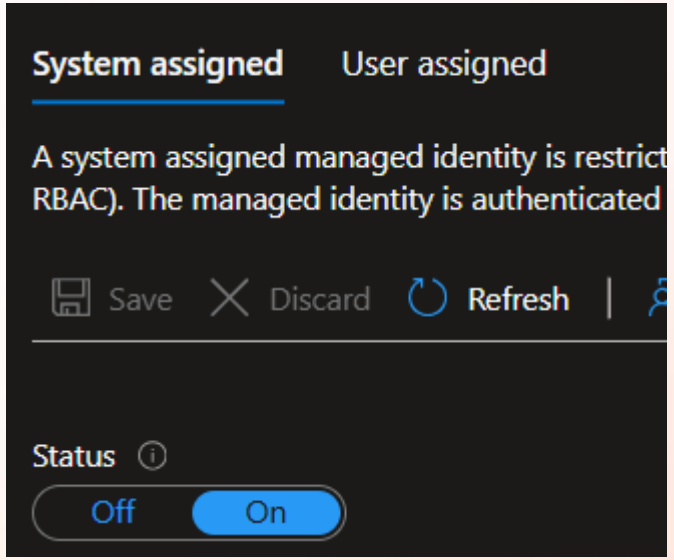
Deploy to Azure Automation

Configuring the managed identity (MI) on the automation account

Where ?

- In **Account settings** > **Identities**
- Enable **System assigned** (or User)

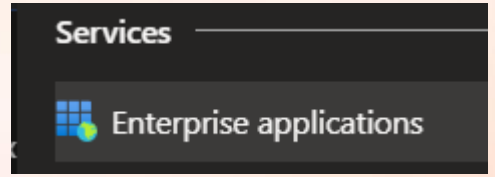


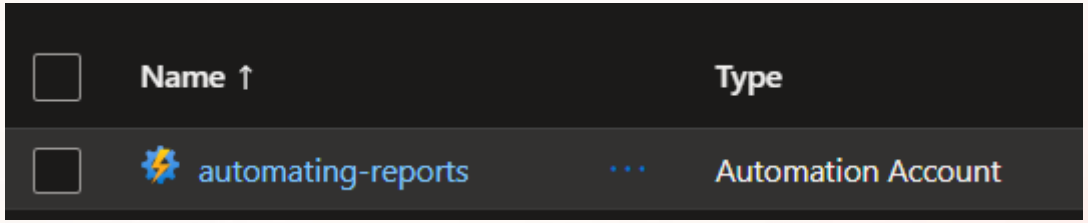


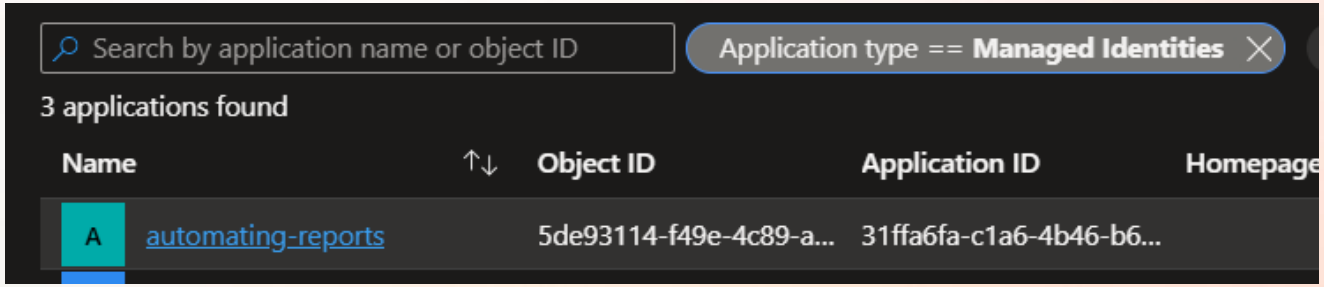
Enterprise application and MI

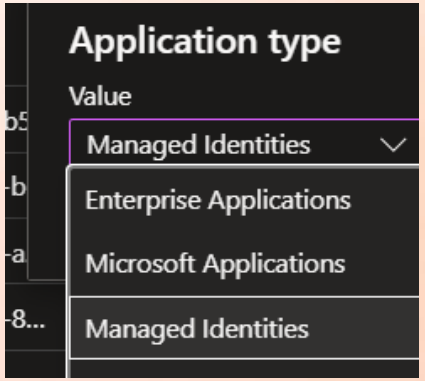
An app with the same name as the Automation Account is available

- Go to Azure > **Enterprise applications**
- Filter on **Managed identities**









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`
}
```

Some useful permissions

	Permission	Cmdlet
Read user device	User.Read.All	Get-MgUserOwnedDevice User.Read.All
Send mail	Mail.send	Send-MgUserMail
Interact with SP site	Sites.Selected	Invoke-MgGraphRequest
Run KQL on MDE	ThreatHunting.Read.All	Start-MgSecurityHuntingQuery

Demo 2

What will we see ?

- Create automation account
- Managed identity
- Adding modules
- Authenticating through the MI
- Adding permissions

Demo 3

What will we see ?

- App registrations expiration alert
- Dynamic devices group based on user
- Interact with MDE for local admin

Runbook and webhook

What is a webhook ?

Webhook

- Unique URL generated in Azure Automation & bind to a runbook

https://<GUID>/<region>/azure-automation.net/webhooks?token=<token>

- **GUID**: generated by Azure Automation
- **Region**: we (for West Europe, in my case)
- **Azure-automation.net**: domain used by Azure Automation
- **webhooks?token**: used for authentication

Uses cases

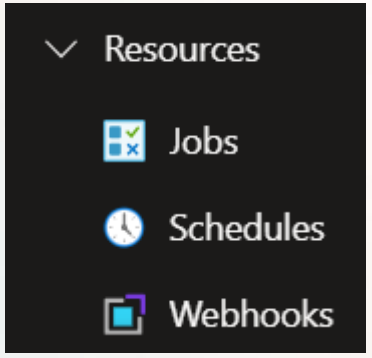
- Automatically add device to Entra ID group after installing an application or remediation
- Run on-demand remediation script from your device without authentication

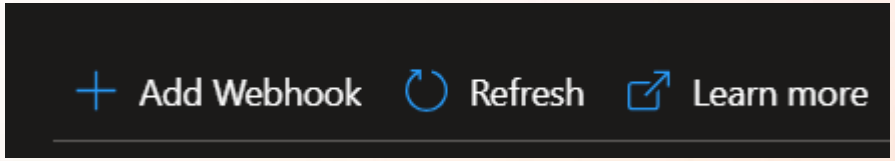
!!! Do not use for sensitive or high-risk operations !!!

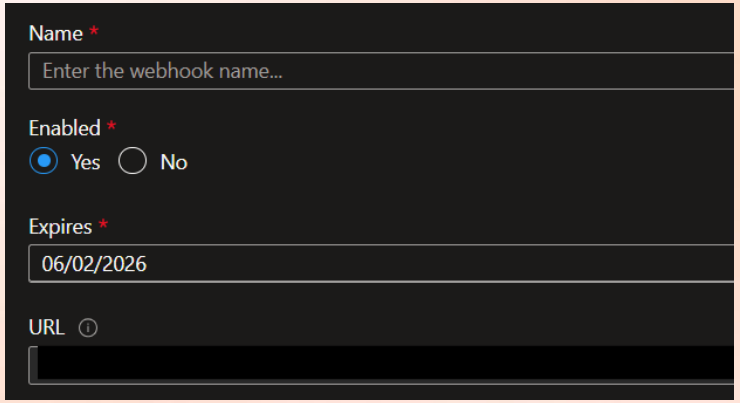
How to add it ?

- A webhook is applied on a Runbook

- Go to your runbook > **Resources**
- Go to **Webhooks** > **Add a webhook**
- Copy the URL to add it in your script







How to call it ?

- With **Invoke-WebRequest** cmdlet and **POST** method
- Add URL of your webhook in **URI** parameter
- Pass info through **$Body**
- More info here

```powershell
$SerialNumber = (Get-WmiObject -Class Win32_Bios).SerialNumber
$params  = @{
DeviceName ="$env:COMPUTERNAME";
SerialNumber ="$SerialNumber"
}

$webhookURI = ""
$body = ConvertTo-Json -InputObject $params
$response = Invoke-WebRequest -Method Post -Uri $webhookURI -Body $body -UseBasicParsing
$StatusCode = $response.StatusCode

$Code_Return = @{
    "202" = "Request accepted"
    "400" = "Bad Request : The webhook has expired or is disabled"
    "404" = "Not Found: webhook, runbook or account wasn't found"
    "500" = "Internal Server Error: The URL was valid, but an error occurred. Resubmit the request"
}
```

In the runbook

- Receives a parameter **WebhookData** containing info sent by the webhook
  - Request body: ConvertFrom-Json $webhookdata.RequestBody
  - Headers: $WebhookData.RequestHeader
  - Webhook name: $WebhookData.WebhookName

```
param (
    [Parameter (Mandatory = $false)]
    [object] $WebHookData
)

$WebhookName = $WebhookData.WebhookName
$Inputs = ConvertFrom-Json $webhookdata.RequestBody
```

```
# get devicenam and SN
$Inputs = ConvertFrom-Json $webhookdata.RequestBody
$DeviceName = $($Inputs.DeviceName)
$SerialNumber = $($Inputs.SerialNumber)
```

Securing webhooks execution

Goal ?

- Runbook with webhook can be executed easily from any devices
- We need to secure it a bit more

How to secure it a bit more (in 2 steps) ?

1. First step
   - Add a password in both the script (that calls the runbook) and runbook
   - Compares both password

2. Second step
   - Check if the device running the script is managed in Intune

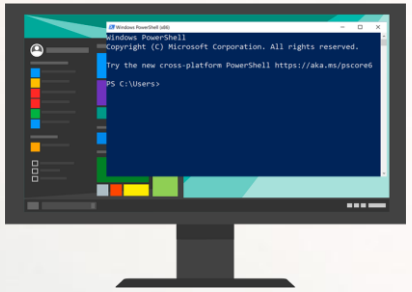
3rd step ? Auto-remove your script after execution

https://www.systanddeploy.com/2022/05/removing-automatically-proactive.html

!!! Do not use for sensitive or high-risk operations !!!

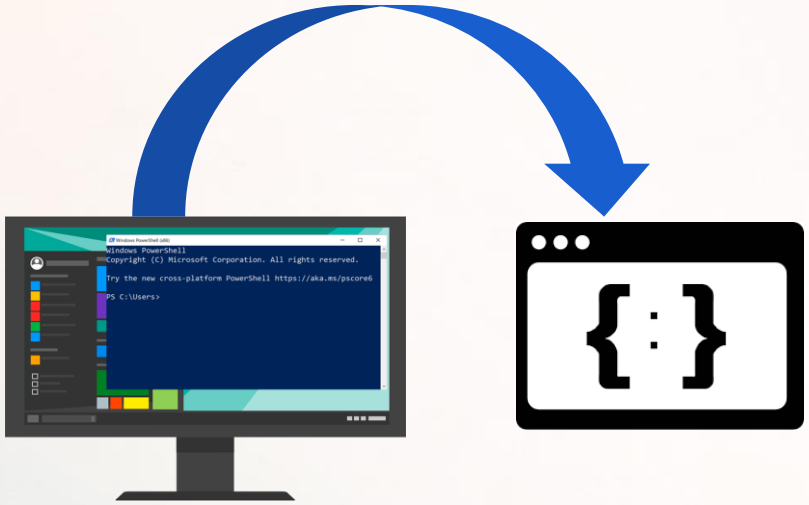
Main process

1. Script is executed on the device



Main process

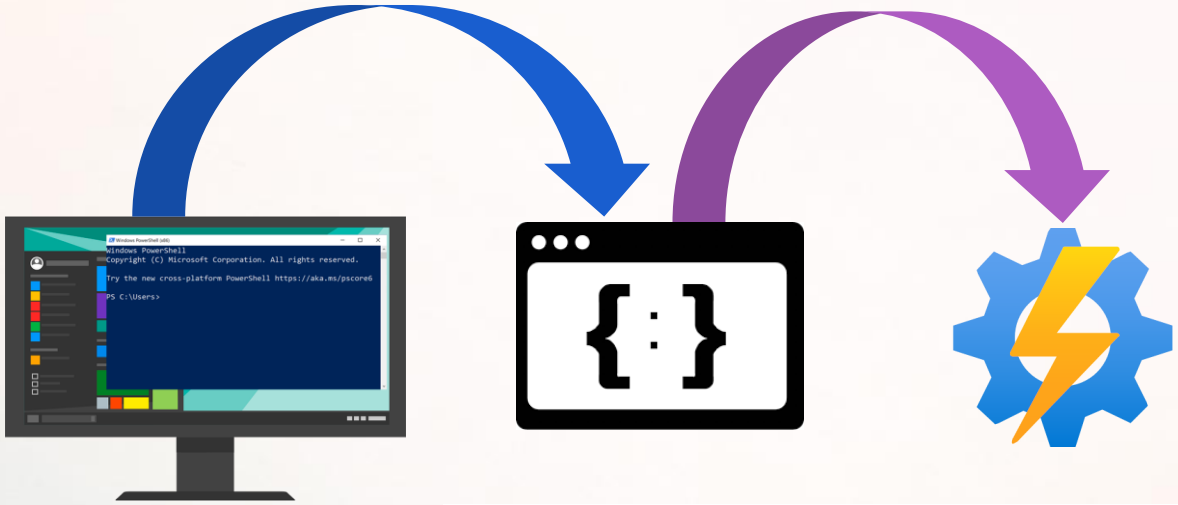
2. Password added in request header & device name sent as parameter



```
$params  = @{
DeviceName = "$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

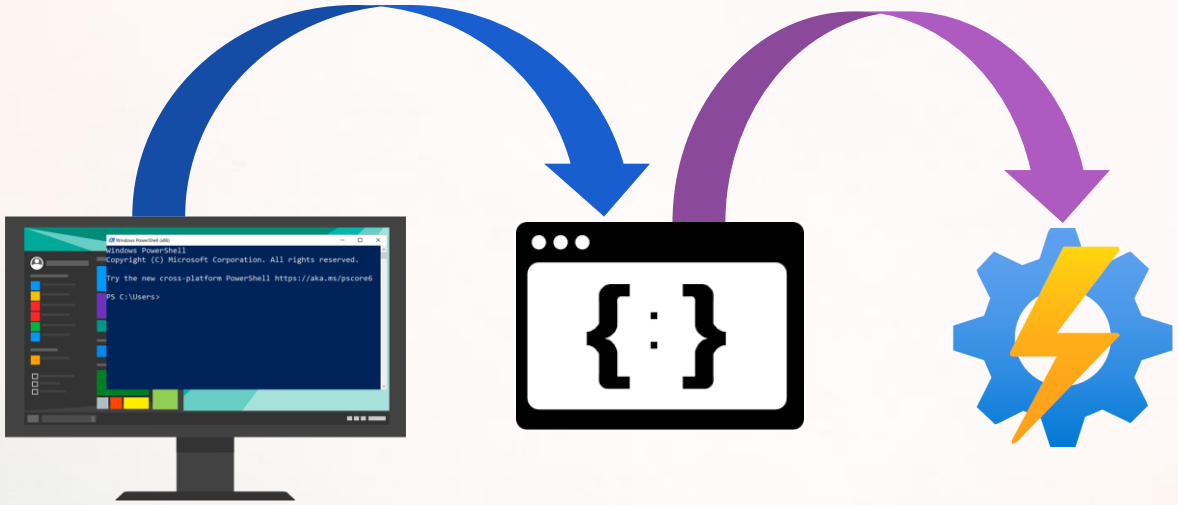
3. Runbook gets the script content

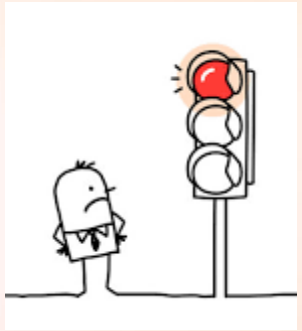


```powershell
param (
    [Parameter (Mandatory = $false)]
    [object] $WebHookData
)
$Inputs = ConvertFrom-Json $webhookdata.RequestBody
$DeviceName = $($Inputs[0].DeviceName)
$Inputs_JSON = $webhookdata.RequestBody
```

Main process

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

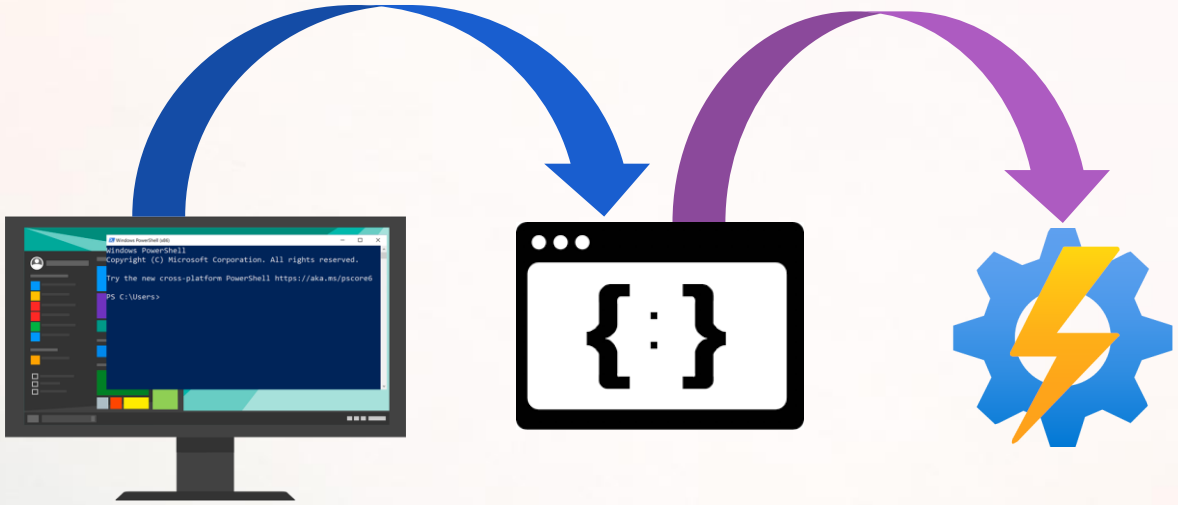


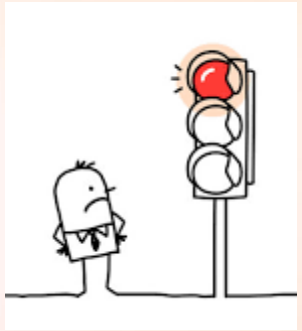



```powershell
If($WebhookData.RequestHeader.message -ne 'Iam_a_bit_more_secure')
{
    "RequestHeader not valid"
    EXIT
}
```

Main process

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

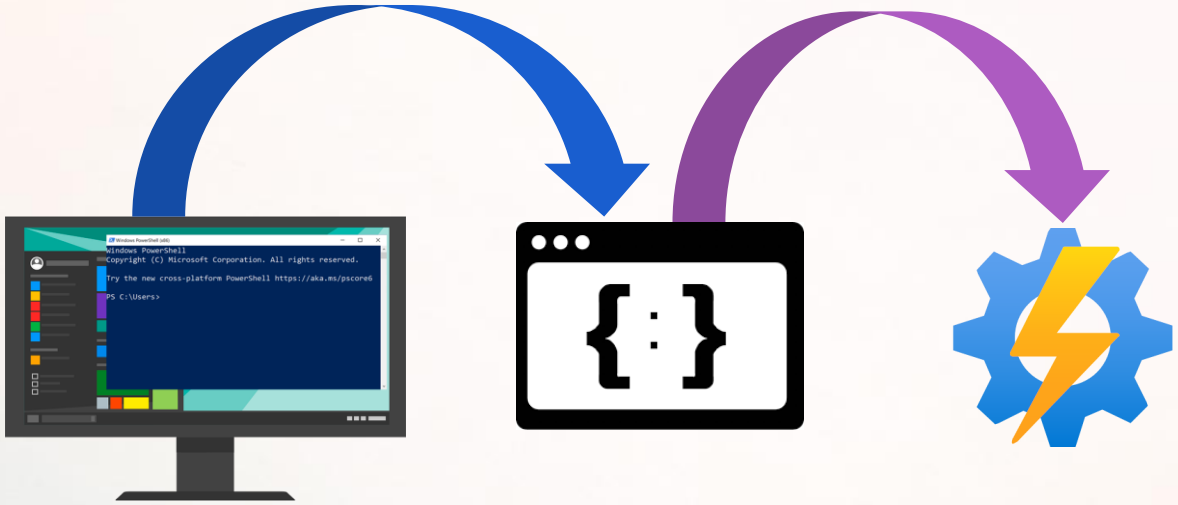


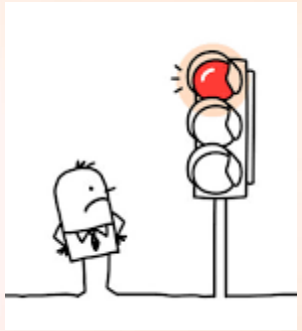



```powershell
$DeviceName = $($Inputs.DeviceName)
Connect-MgGraph -Identity
$Get_Device_Info = Get-MgDeviceManagementManagedDevice -Filter "contains(deviceName,'$DeviceName')"
if (-not $Get_Device_Info) {
    Write-Output "Device not found in managedDevices."
    EXIT
}
```

Main process

5. If everything is OK ➔ **Continue the runbook execution**





Demo 4

What will we see ?

- Adding a webhook
- Securing runbook execution
- Adding device in a group

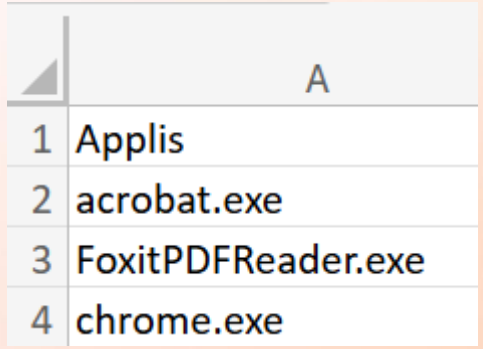
Building intelligent modular runbook

For what ?

Use case

A runbook is used to:

1. Import a list of apps in a CSV on SharePoint (or from SharePoint list)
2. For each app, run KQL query on MDE (**DeviceProcessEvents**) to get app usage
3. Export result to CSV for each app or export to a Log Analytics table



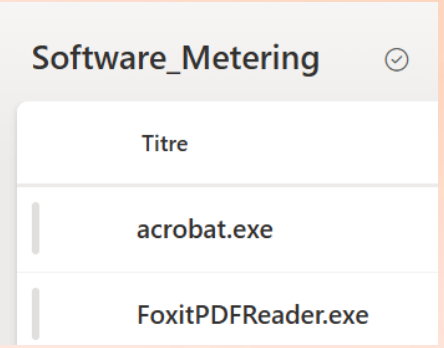
	A
1	Applis
2	acrobat.exe
3	FoxitPDFReader.exe
4	chrome.exe

It may consume a large amount of data and may fail with error:
System.OutOfMemoryException

Modular runbook

Instead of building one runbook that executes all actions, we can have two runbooks:

- Main runbook
- Child runbook



Software_Metering

Titre
acrobat.exe
FoxitPDFReader.exe

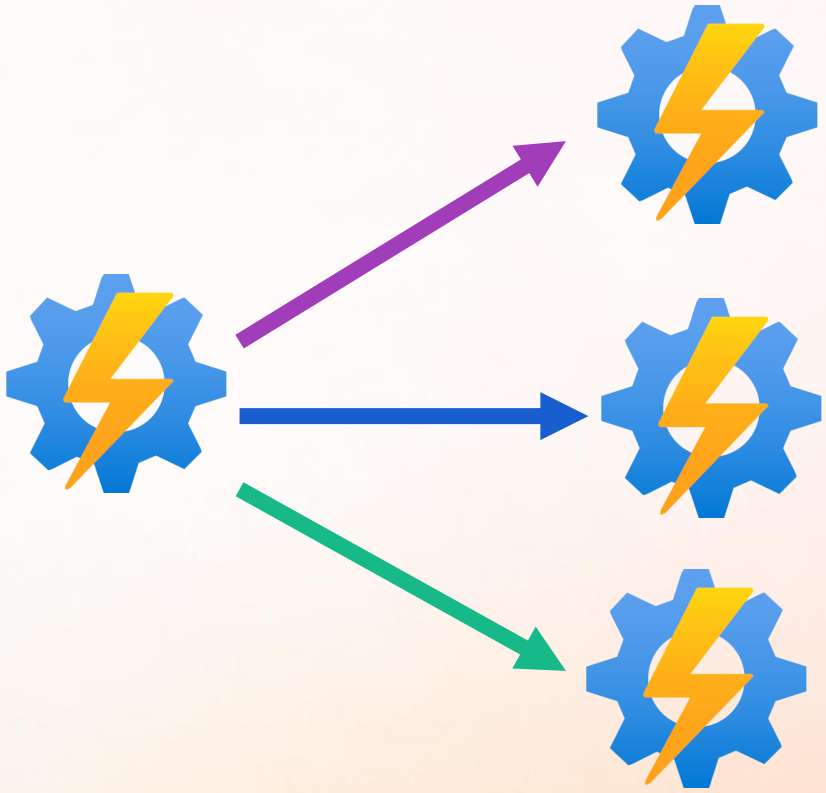
Main runbook runs each KQL query and export report in a child runbook instance

New process

A main runbook is used to call once or multiples instances of a child runbook

Main runbook

1. Load CSV and for each app
2. Call child runbook for the app

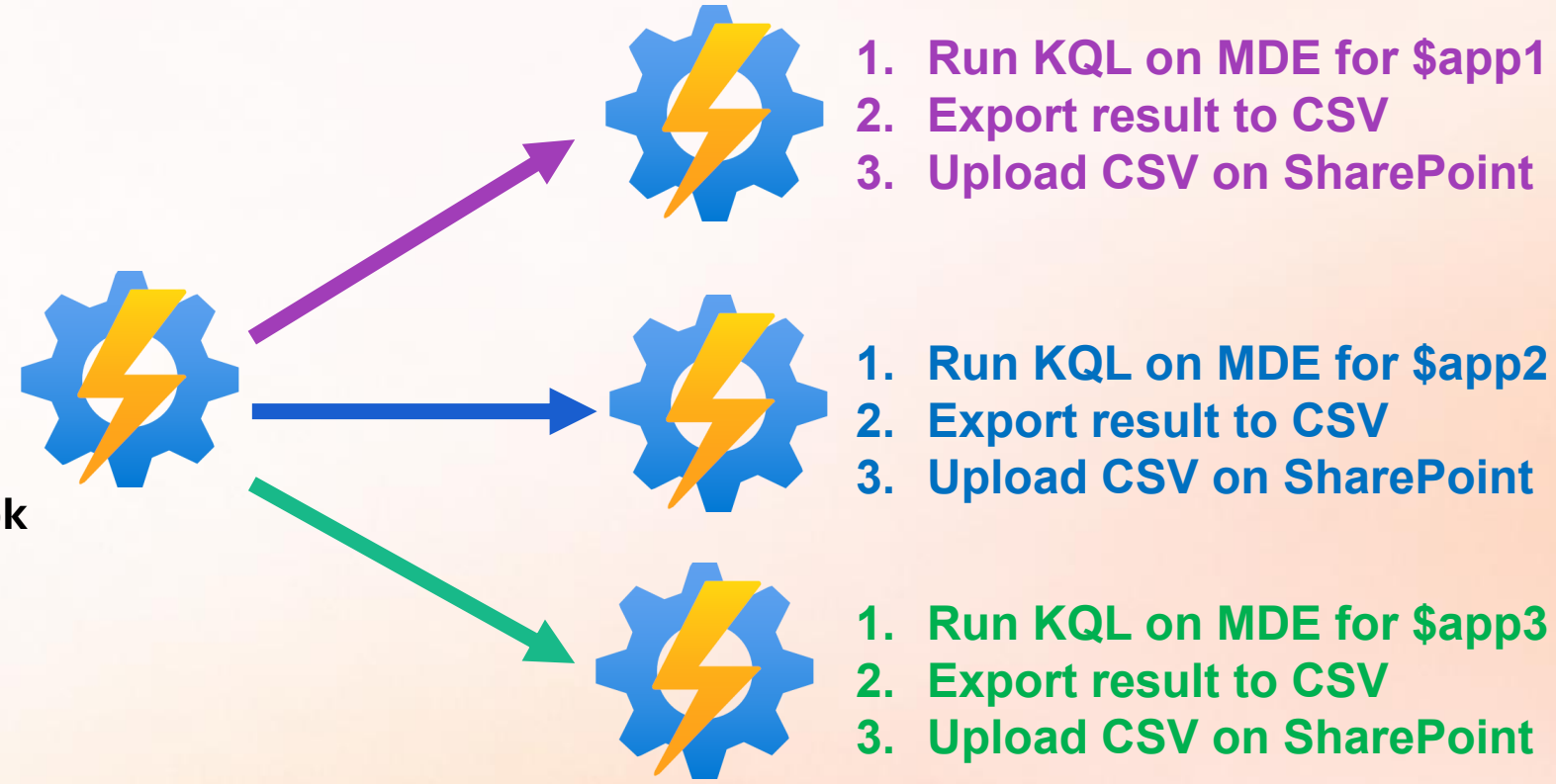


1. Run KQL on MDE for $app1
2. Export result to CSV
3. Upload CSV on SharePoint

1. Run KQL on MDE for $app2
2. Export result to CSV
3. Upload CSV on SharePoint

1. Run KQL on MDE for $app3
2. Export result to CSV
3. Upload CSV on SharePoint

Make it more intelligent



Calling child runbook

1. **Specify info about automation account and child runbook**

$AutomationAccountName = « <Your automation account name »
$ResourceGroupName = « <Your resource group> »
$RunbookName = « <Your child runbook> »

2. **Authenticate through the managed identity**

Connect-AzAccount -Identity

3. **Calling the child runbook**

Start-AzAutomationRunbook -AutomationAccountName $AutomationAccountName -ResourceGroupName $ResourceGroupName -Name $RunbookName

Demo 5

What will we see ?

- Calling child runbook
- Building intelligent modular runbook
- Software Metering example

Please rate this session on Sched.com

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

Thanks!