Microsoft

MICROSOFT PURVIEW ADVANCED RICH REPORTS FOR POWER BI AND SENTINEL

BASED ON OFFICE 365 MANAGEMENT API, MICROSOFT GRAPH API, AIP SERVICE API AND OFFICE 365 EXCHANGE ONLINE API

(MARCH 2023)

GitHub

https://aka.ms/MPARR-GitHub

Sebastián Zamorano A.
ISD Senior Consultant

Gratitude

My special gratitude to all my colleagues that helped and supported me on this new release. Special thanks to Grzegorz Berdzik, our black belt in this script; Dominik Kot who share some ideas and present me to Grzegorz; Stephan Carsten that who put some order in this script. And others that contact me constantly to ask for this, they make a lot of pression and now is here.

Additional thanks to Florian Boigner who is guide me on the Power B.I path, as a good instructor.

We cannot forget to Walid Elmorsy who is the owner of the original script used for all of this.

And several other people that is believe on me and this crazy idea.

Really thanks to all

Sebastián Andrés Zamorano Andrade

AGENDA

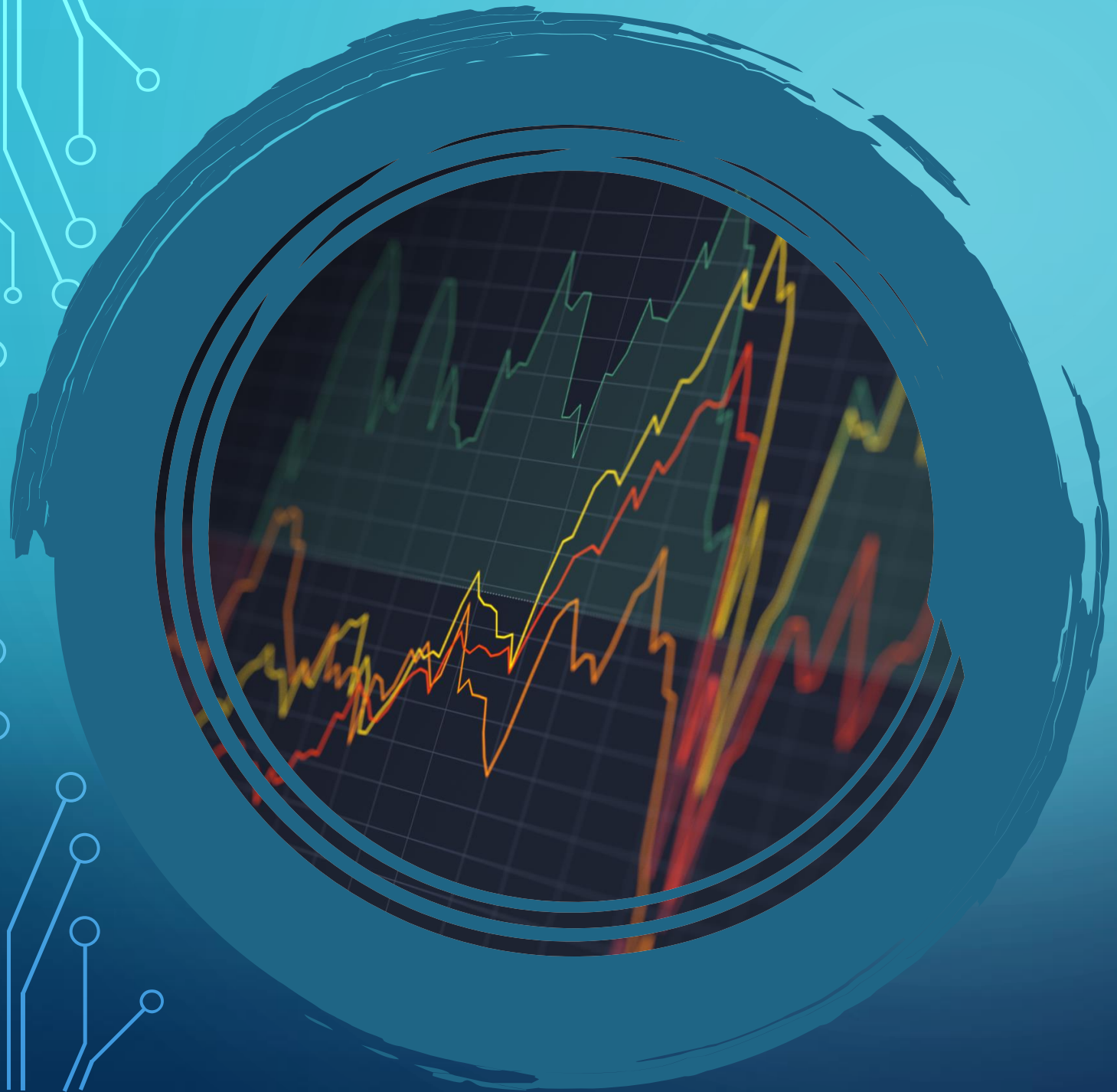
WHAT WILL WE HAVE FOUND HERE?

AGENDA

What will we have found here?

- Samples OF Queries and reports that can be created
- General concepts
- Prerequisites
- Baseline configuration
- Power BI configuration
- Advanced Settings

Microsoft



DASHBOARDS

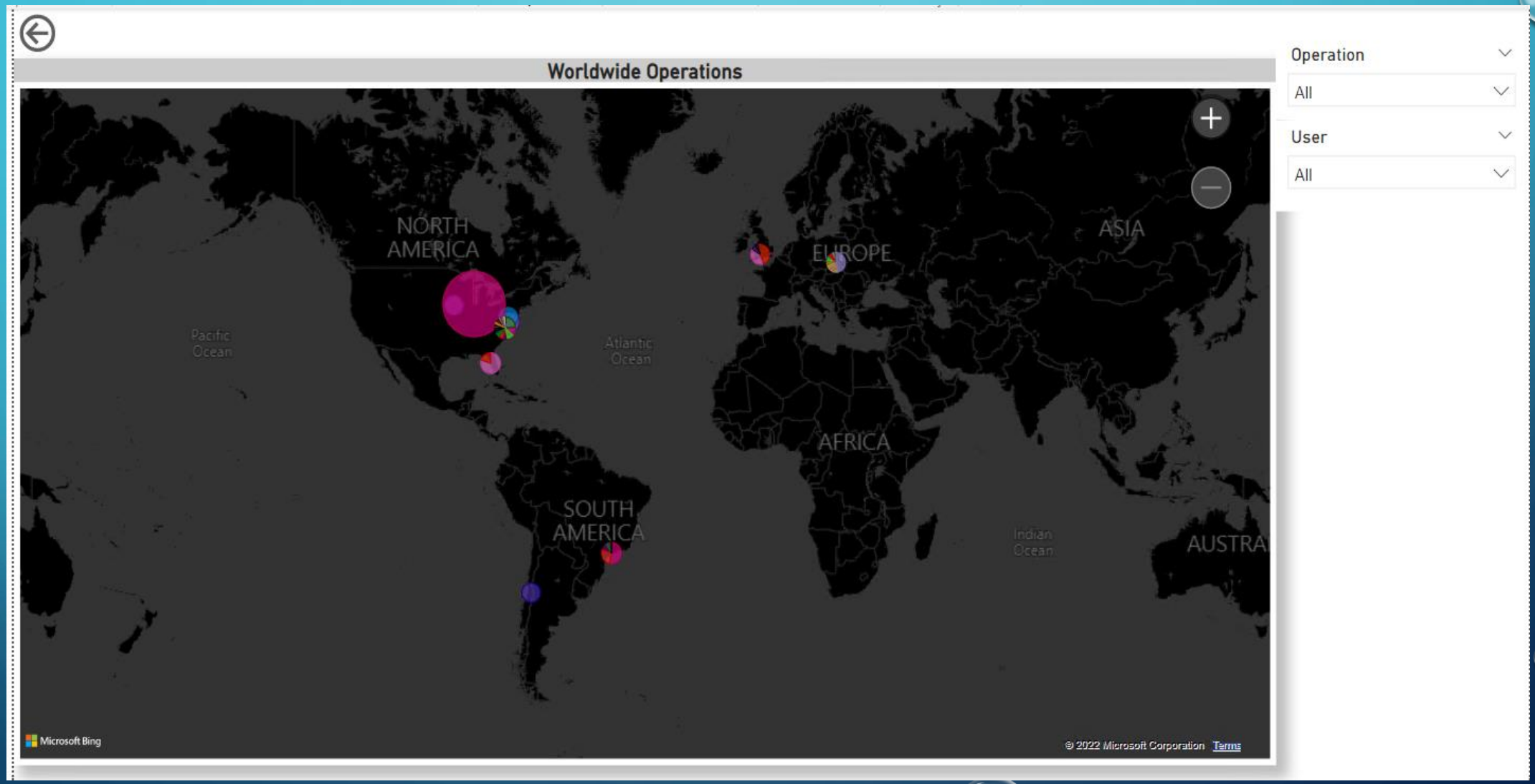
SOME SAMPLE REPORTS

DASHBOARD - OVERVIEW

Dashboard for Unified Labeling behavior

DASHBOARD – GEO LOCATION

Dashboard for audit activities



DASHBOARD – USER BEHAVIOR

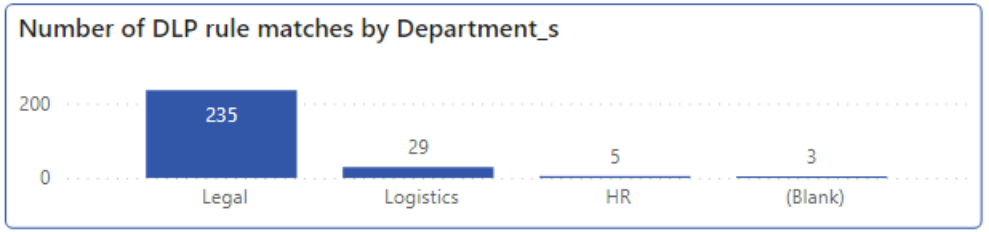
Dashboard for DLP activities

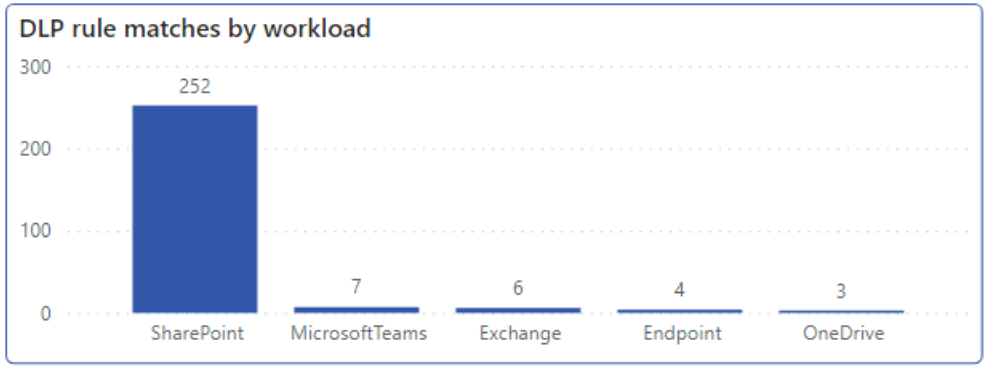
TimeGenerated

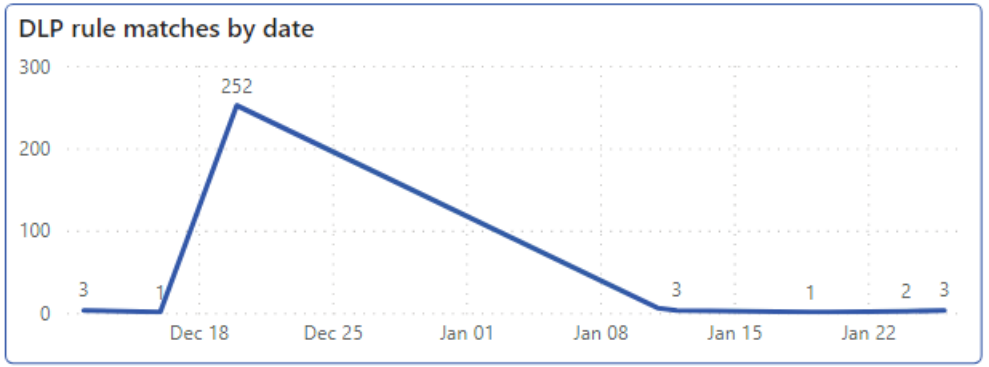
12/12/2022 1/26/2023

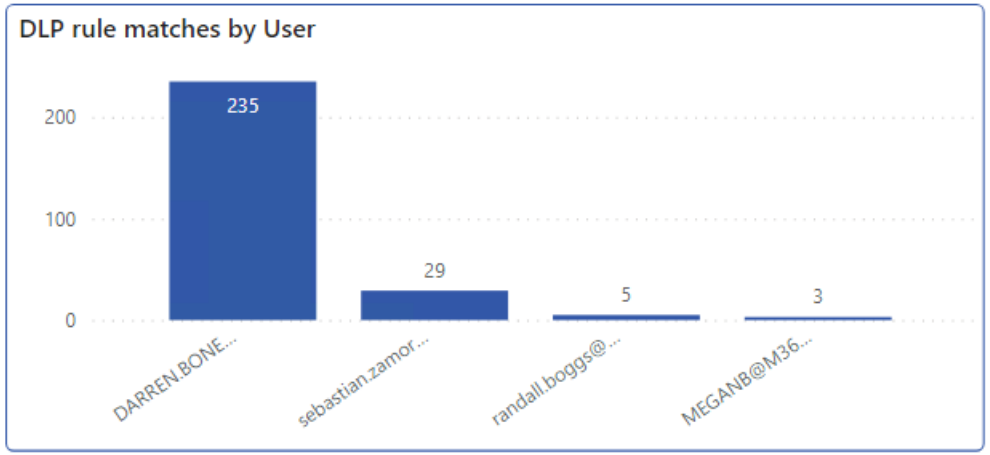
272

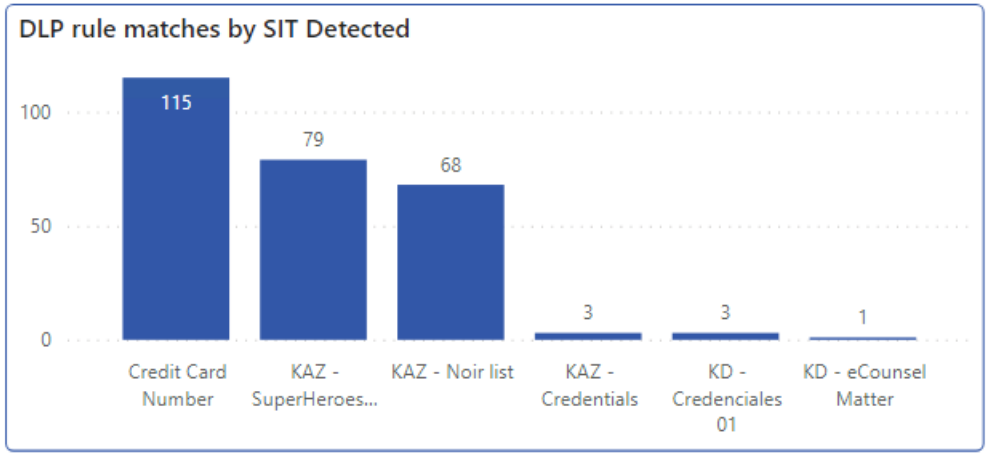
Number of DLP rule matches









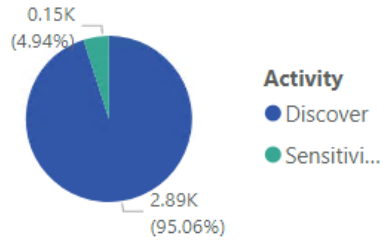


Right click on SIT Detected for drill-down to the details page

DASHBOARD – AIP SCANNER

Dashboard for MIP Scanner activities

MIP Scanner activities



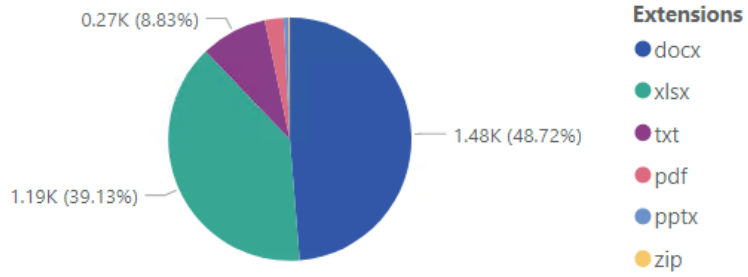
Total files identified

3036

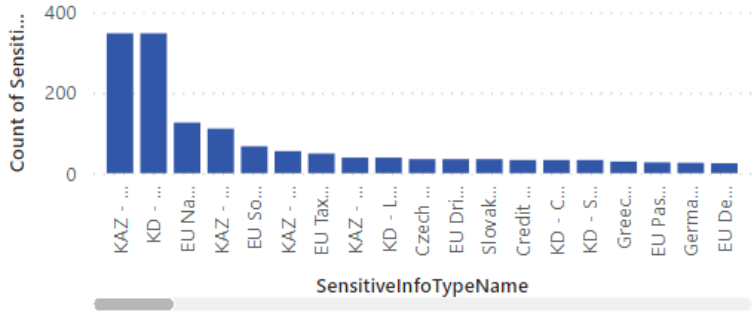
Date

12/14/2022 3/10/2023

Kind of files detected

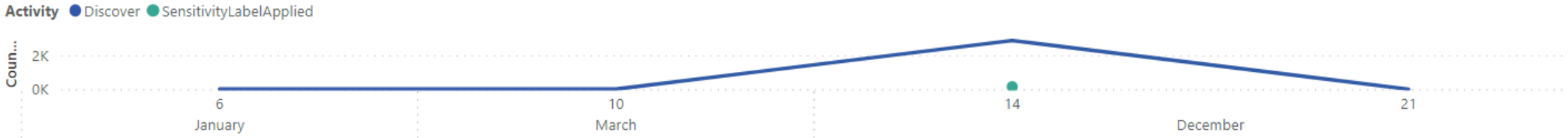


Sensitive Info Types detected



FileName	PATH	Label Applied	SensitiveInfoTypeName	Activity	Extensions	Protection Owner	Confidence
create samples.xlsx	c:\\Data\\MyNewFolder\\PII		Ukraine Passport Number (International)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII		Ukraine Passport Number (International)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII\\PII		Ukraine Passport Number (International)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII\\PII\\PII		Ukraine Passport Number (International)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII		Ukraine Passport Number (Domestic)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII		Ukraine Passport Number (Domestic)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII\\PII		Ukraine Passport Number (Domestic)	Discover	xlsx		75
create samples.xlsx	c:\\Data\\MyNewFolder\\PII\\PII\\PII		Ukraine Passport Number (Domestic)	Discover	xlsx		75

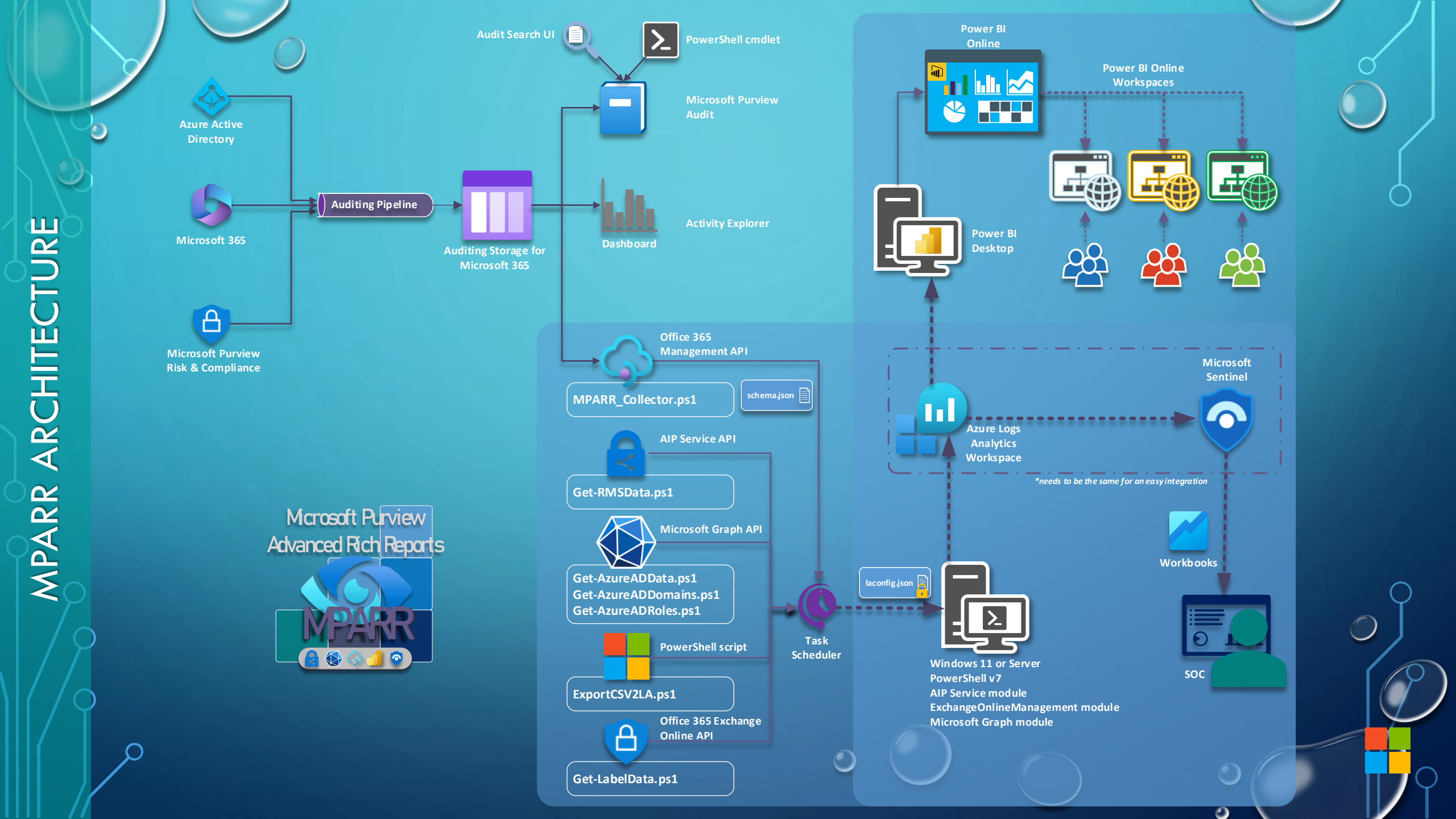
MIP Scanner per day



SOME STUFFS TO TAKE ON MIND

Capabilities to implement previously reports shown depends to implement MPARR as your solution to collect all the information available through Office 365 Management API, and supporting APIs, if some of the scripts are not implemented, the reports can be not implemented in the right way.

MPARR ARCHITECTURE



CHECKLIST

HAVE I ALL TO OBTAIN RESULTS?

Certs folder: To execute the **Get-xxxxx** scripts we need set some permissions based on certificates, inside of this folder you will find a **CreateCertificate.ps1** script, used for that function.

Logs Folder: Here the **timestamp.json** file is set (this file is used for **export_logs.ps1** script) and additional any error is record here (The default folder is located on C:\APILogs)

RMSLogs Folder: Used by Get-RMSData.ps1 script, this is used to process data collected from AIP Service API.

Support Folder: Contains the document used to create the table with service plan and friendly names, that can be downloaded from here

Laconfig.json: this file contains the keys and data require to execute the next scripts; the data related to keys can be encrypted.

Schemas.json: this file contains the names of the content blobs used by the Office 365 Management API, and the collector takes the information from these ones, if some information cannot be collected, the value needs to be change from True to False.

MPARR_Collector.ps1: Script used to obtain the data from Office 365 management API and send to Logs Analytics, to use with parameters through Task Scheduler a new file needs to be created called **run_me.ps1**

ExportCSV2LA.ps1: Script used to export the CSV file (located on Support folder) with service plan to Logs Analytics (required to execute on-demand)

Get-AzureADData.ps1: Script used to collect data from Azure AD using Microsoft Graph API, permissions are added to Azure AD App for unattended execution, can be modified to add or remove certain Azure AD attributes (Can be execute through Task Scheduler Monthly, depending on users' rotation)

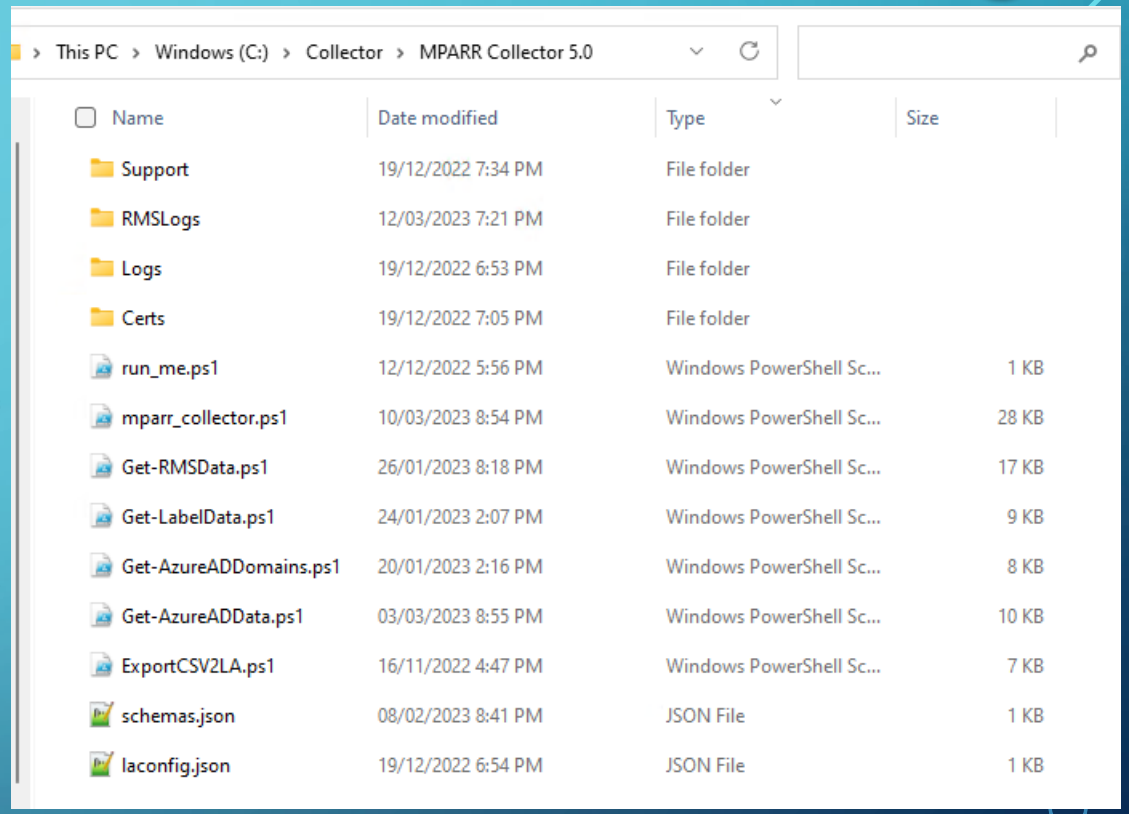
Get-AzureADDomains.ps1: Script used to collect data from Azure AD using Microsoft Graph API, permissions are added to Azure AD App for unattended execution, this script permit to collect all the domains registered at Tenant.

Get-AzureADRoles.ps1: Script used to collect data from Azure AD using Microsoft Graph API, permissions are added to Azure AD App for unattended execution, this script permit to collect all the administrator roles assigned to user accounts.

Get-LabelData.ps1: Script used to obtain Display Name and ID for labels, require ExchangeOnlineManagement PowerShell Module, permissions are added to Azure AD App for unattended execution (can be executed on-demand)

Get-RMSData.ps1: Script used to obtain information about external access to protected files, require AIPService PowerShell Module, permissions are added to Azure AD App for unattended execution , can be execute with the same timing as export_logs.ps1

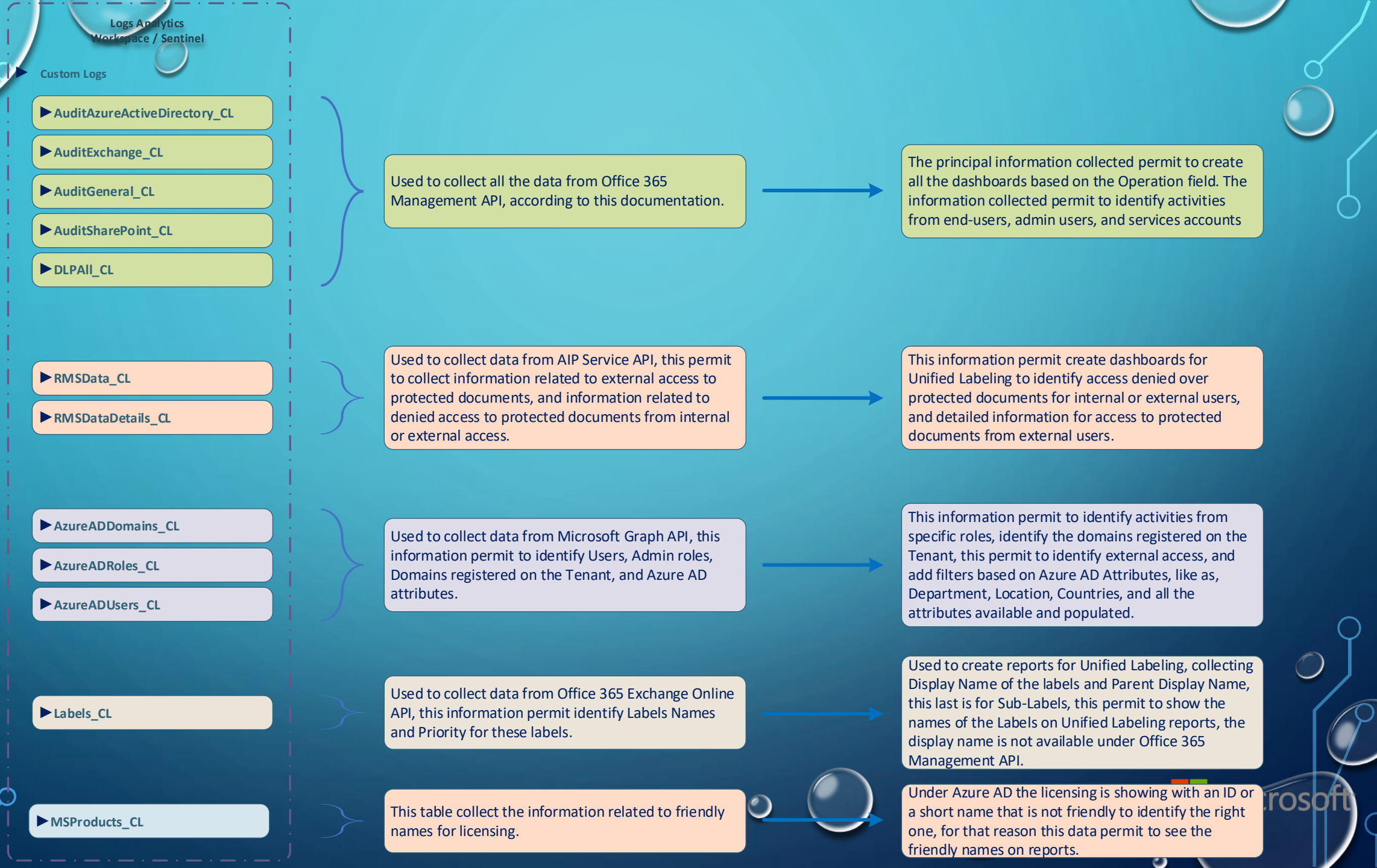
Run_me.ps1: script used to call export_logs.ps1 with parameters to use with task scheduler



Microsoft

DETAILED ARCHITECTURE

DETAILED ARCHITECTURE



PREREQUISITES

Prerequisites to implement

- Logs Analytics workspace (Azure subscription)
- Workstation or Server with Internet access
- PowerShell v7 installed on the previous machine
- ExchangeOnlineManagement version 3.0.0 module for PowerShell
- Microsoft.Graph PowerShell module installed
- AIPService Module for PowerShell
- At least Application administrator to create an App under Azure AD
- Compliance administrator role to obtain Sensitivity Labels list
- Power BI desktop
- Open URLs:
  - https://*.aadrm.com
  - https://*.protection.outlook.com
  - https://*.azure.com
  - Https://manage.office.com (can be different for GCC, GCCH or DoD tenants)
  - https://graph.microsoft.com
  - https://login.microsoftonline.com

Microsoft

INDEX

GO DIRECTLY FROM HERE TO THE SPECIFIC TOPICS

Microsoft

AZURE AD

Create App to recollect data (1/2)

- Open https://portal.azure.com
- Look for Azure Active Directory
- Go to "**App registrations**" menu
- Press **New registration**
  - Use any name here
  - Select Accounts in this organizational directory only
  - On Redirect URI, select Web and set https://localhost
  - Press Register
- On the new app
  - copy Application (client) ID and Directory (tenant) ID

To give unattended access

- Then go to Certificates & secrets, press +New Client Secret and set a Description and Expiration time
- After press Add button on the blade a new Key will be created, copy that key under Value column
- Under certificates tab, on the same menu, select Upload certificate
- Select the certificate created on this Step and add a description.

Microsoft

AZURE AD

Create App to recollect data (2/2)

- On the new app (coming from previous page)
  - Go to API permissions and select +Add a permission:

    To give access to Office 365 Management API

    - At the blade open looking for Office 365 Management APIs
    - After select this Application, click on Application permissions
    - Select the 3 options available, check all of them, and then press Add permissions button at the final of that blade

    To give access to Microsoft Graph

    - At the blade open looking for Microsoft Graph
    - After select this Application, click on Application permissions
    - Select AuditLog.Read.All, Directory.Read.All, Group.Read.All, Organization.Read.All, User.Read.All

    To give access to Compliance PowerShell console (Additional permission is required here)

    - At the blade open looking, at top select the tab "APIs my organization uses"
    - Search for "Office 365 Exchange online"
    - After select this Application, click on Application permissions
    - Under Exchange submenu select "Exchange.ManageAsApp"

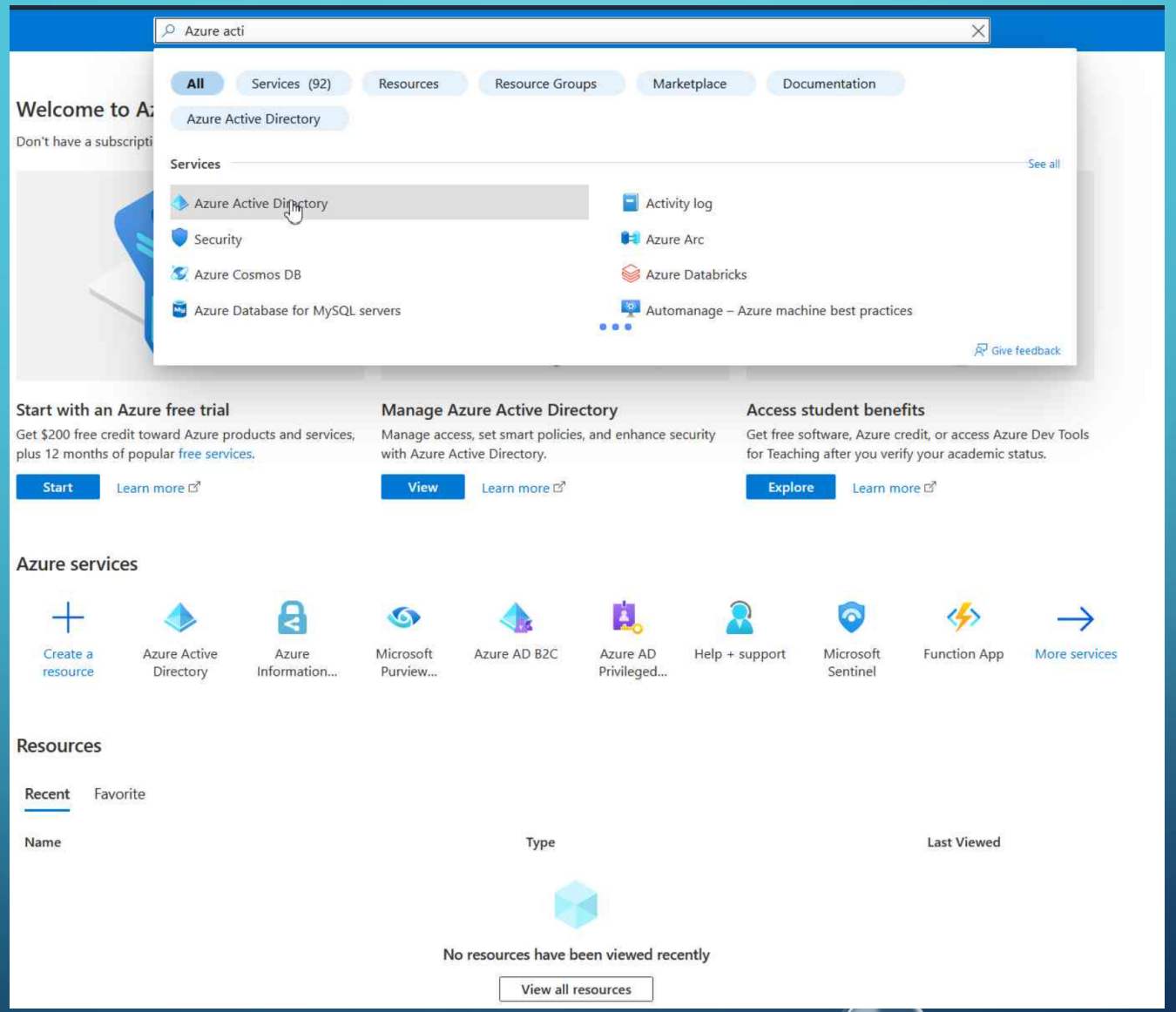
    To give access to AIP Service API (Azure Rights Management Services)

    - At the blade open looking for Azure Rights Management Services
    - After select this Application, click on Application permissions
    - Select Application.Read.All
  - Press "Add permissions" in all the cases

Microsoft

AZURE AD

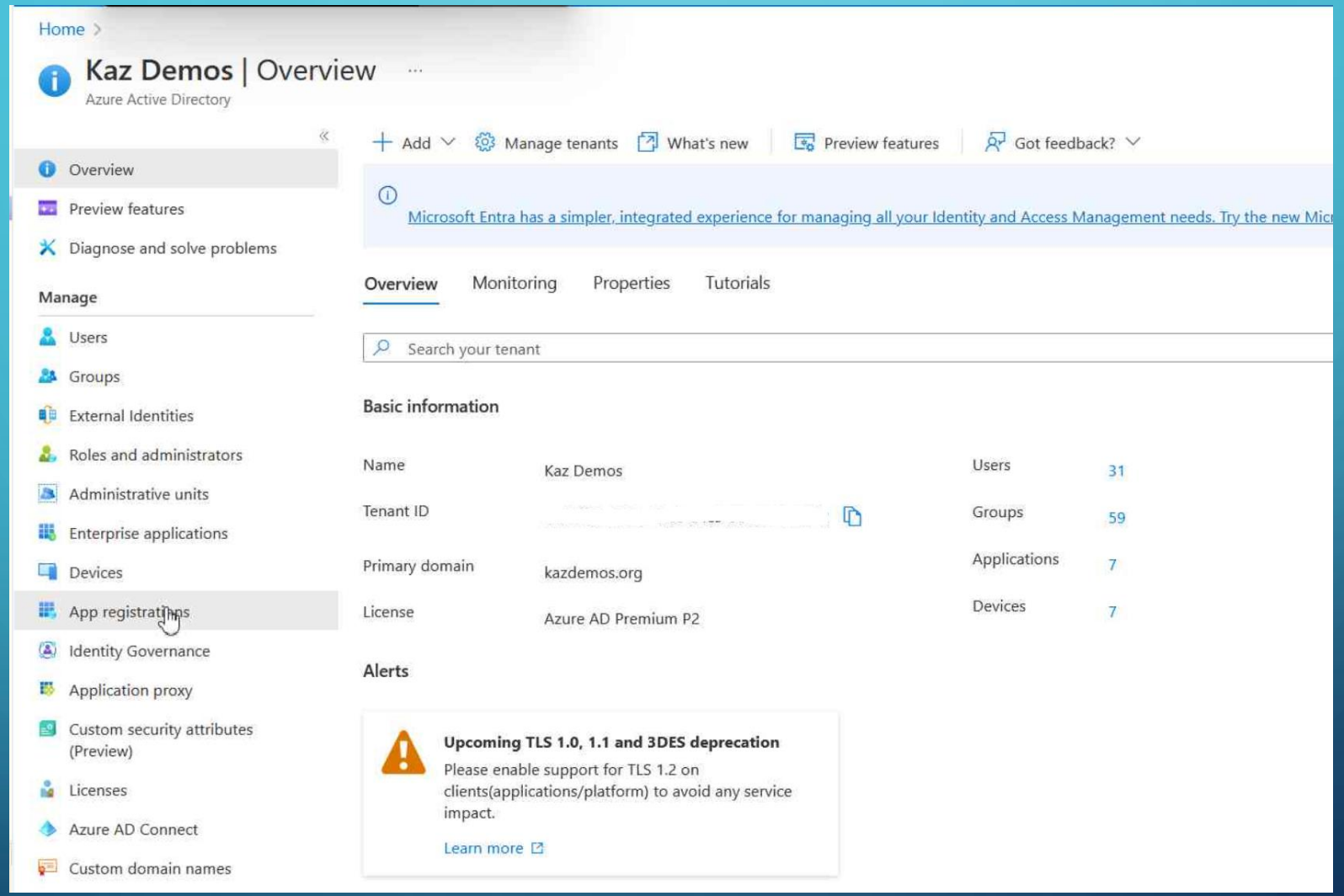
Steps



Microsoft

AZURE AD

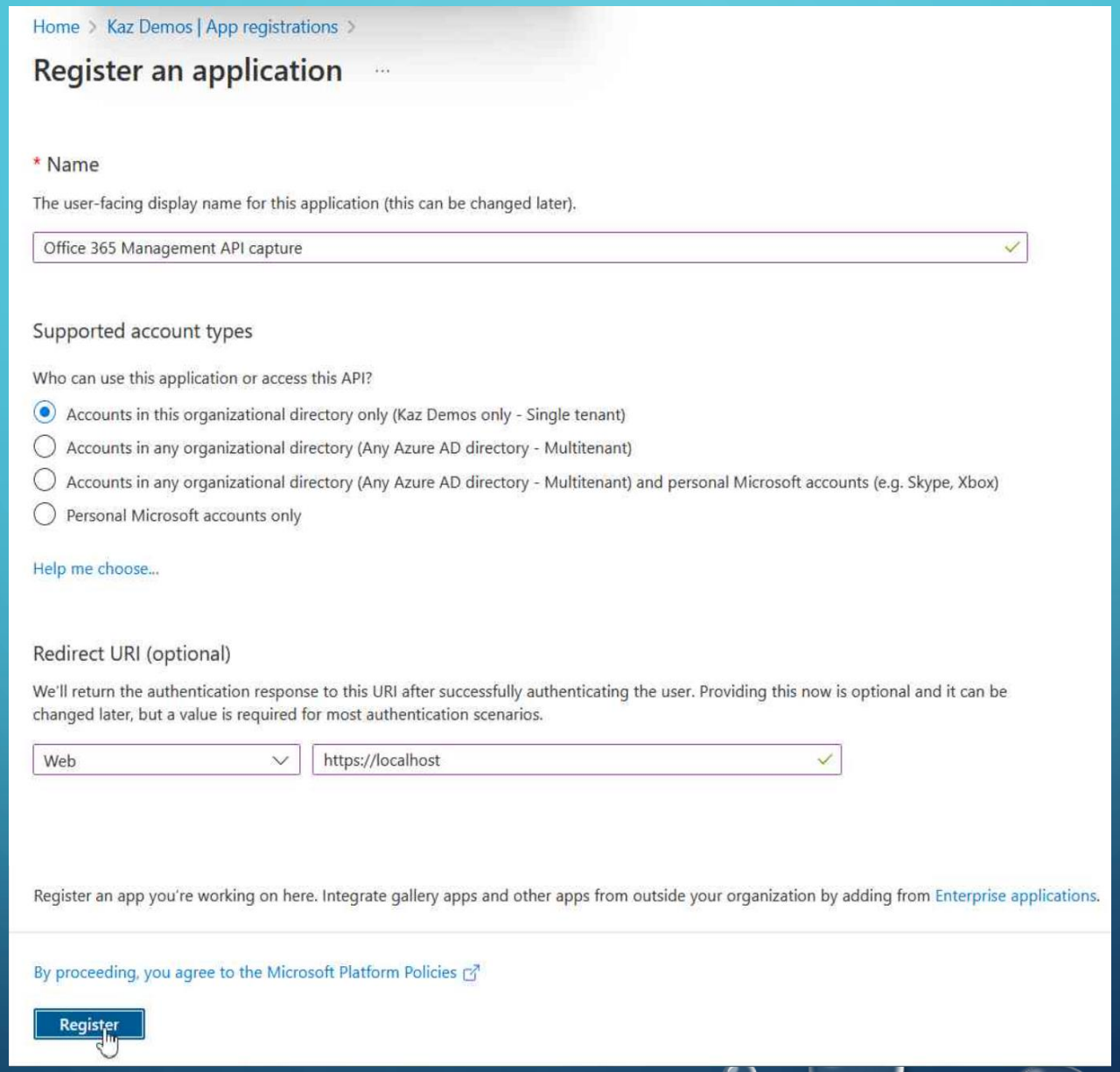
Steps



Microsoft

AZURE AD

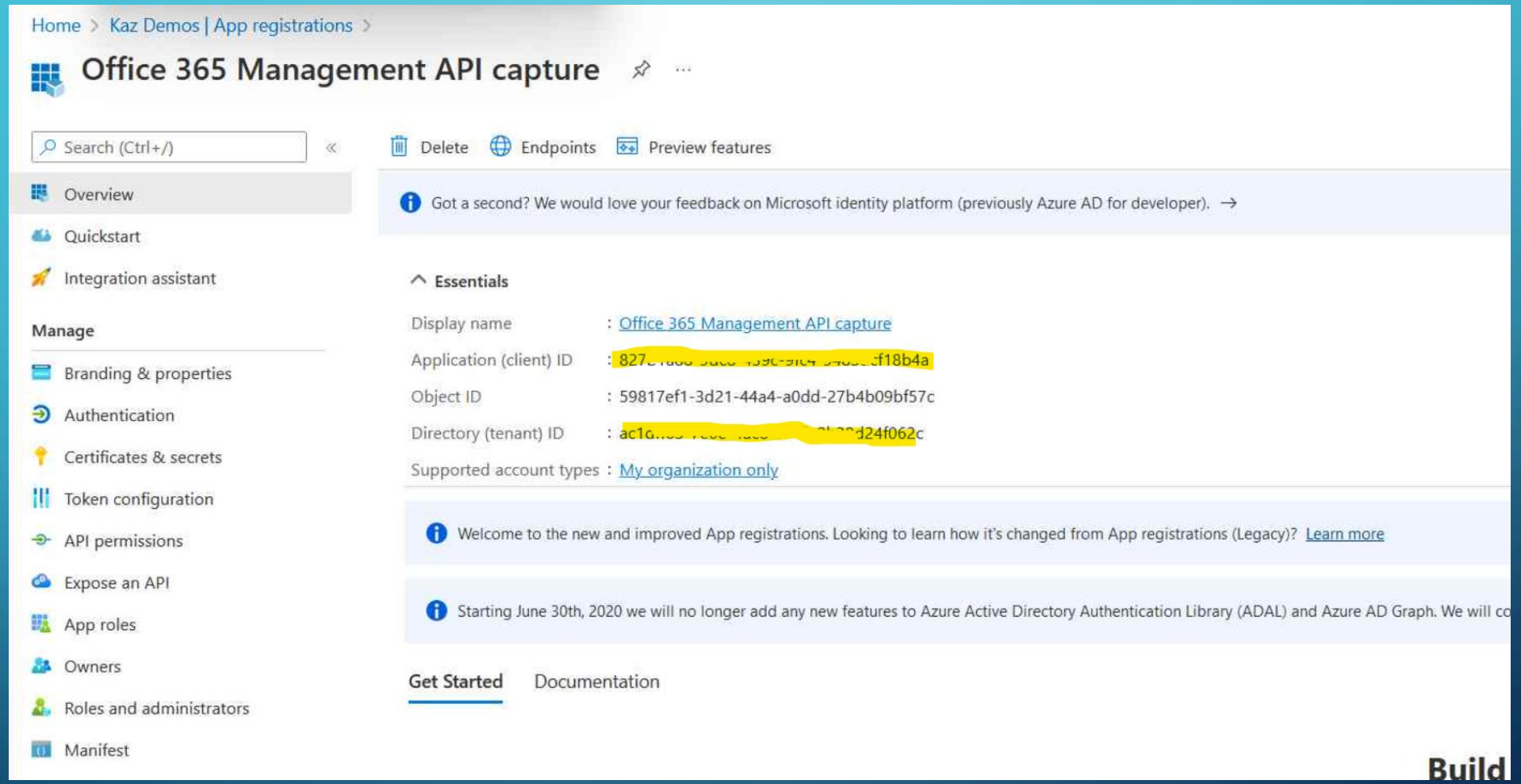
Steps



Microsoft

AZURE AD

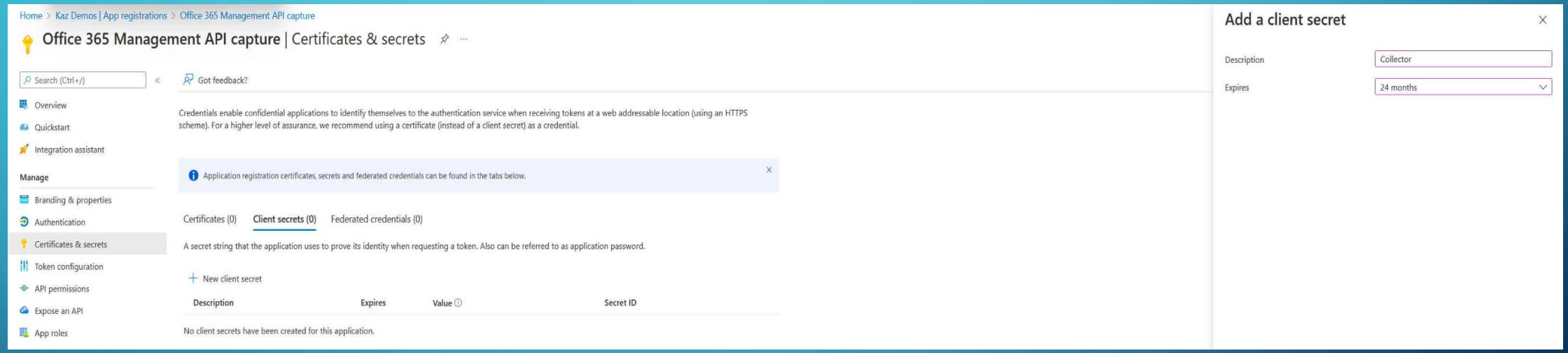
Steps



Microsoft

AZURE AD

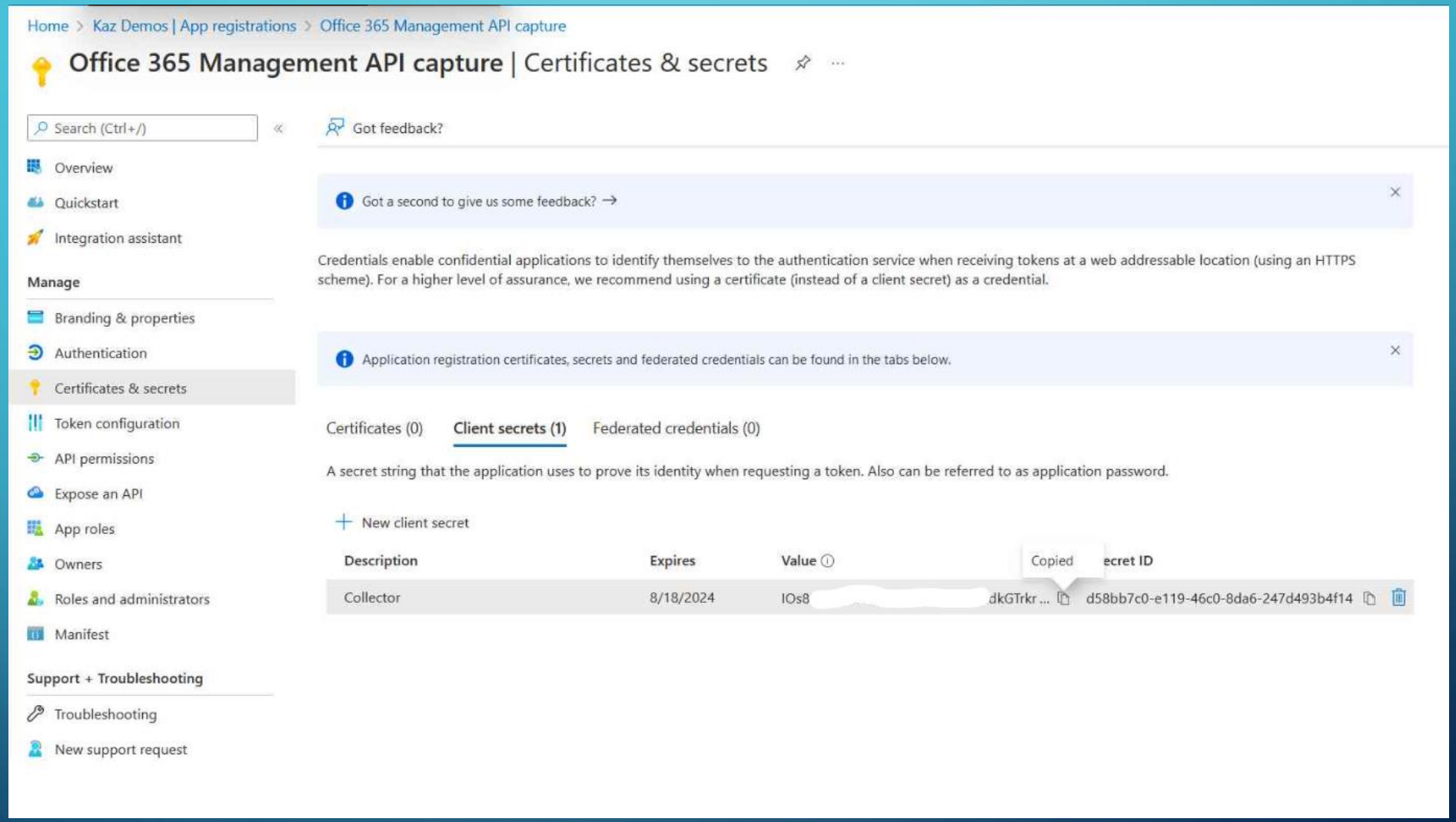
Steps



Microsoft

AZURE AD

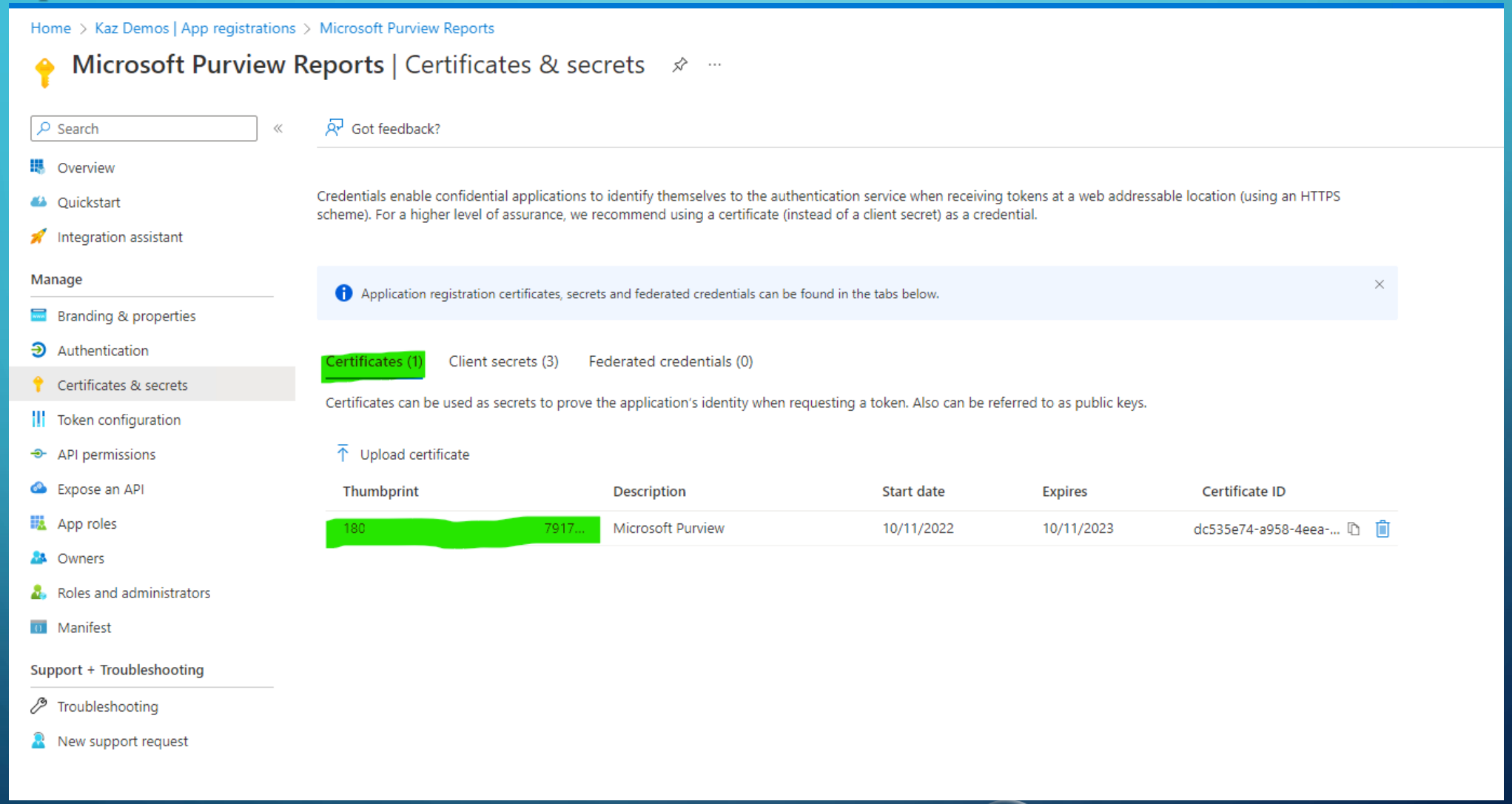
Steps



Microsoft

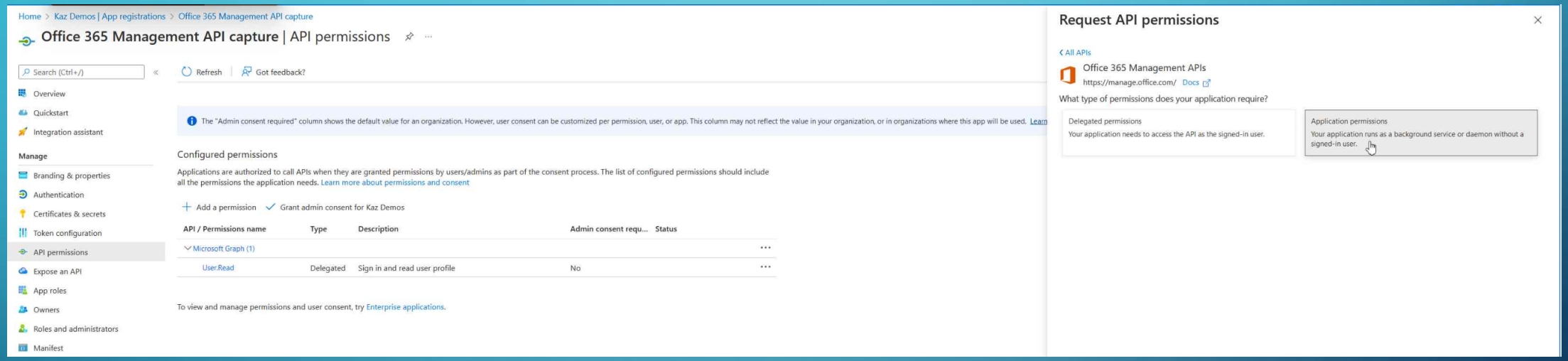
AZURE AD

Steps



AZURE AD

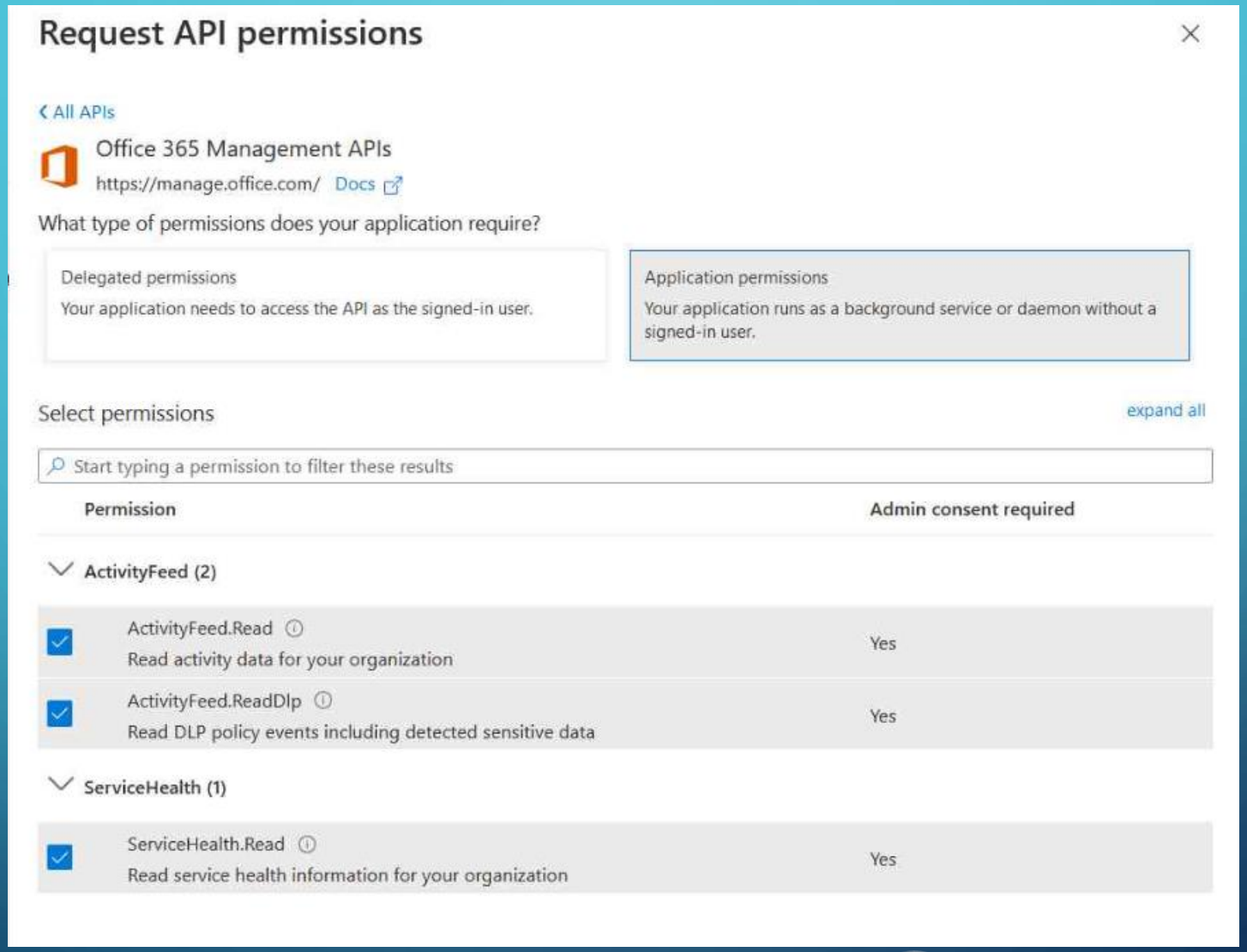
Steps



Microsoft

AZURE AD

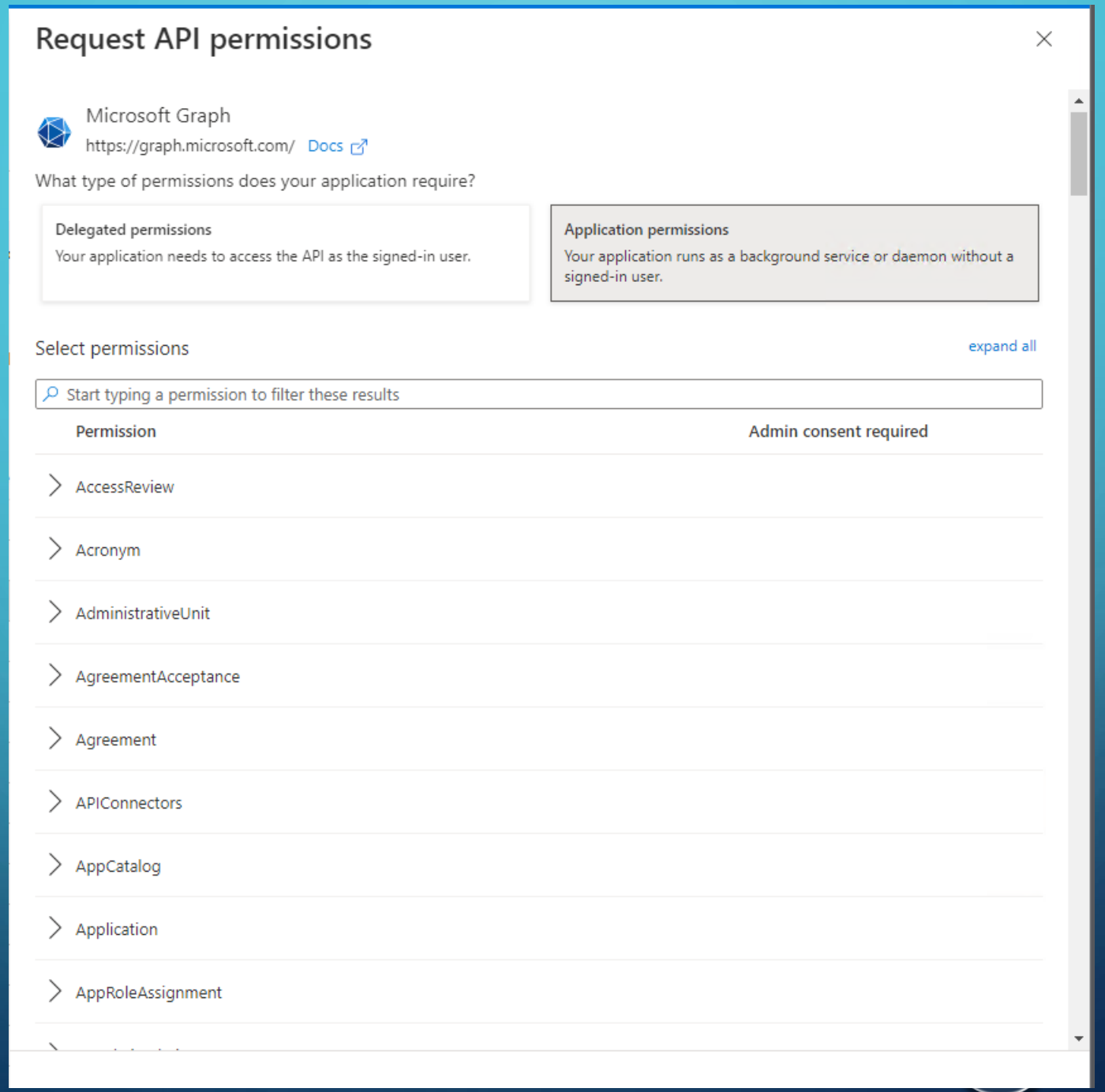
Steps



Microsoft

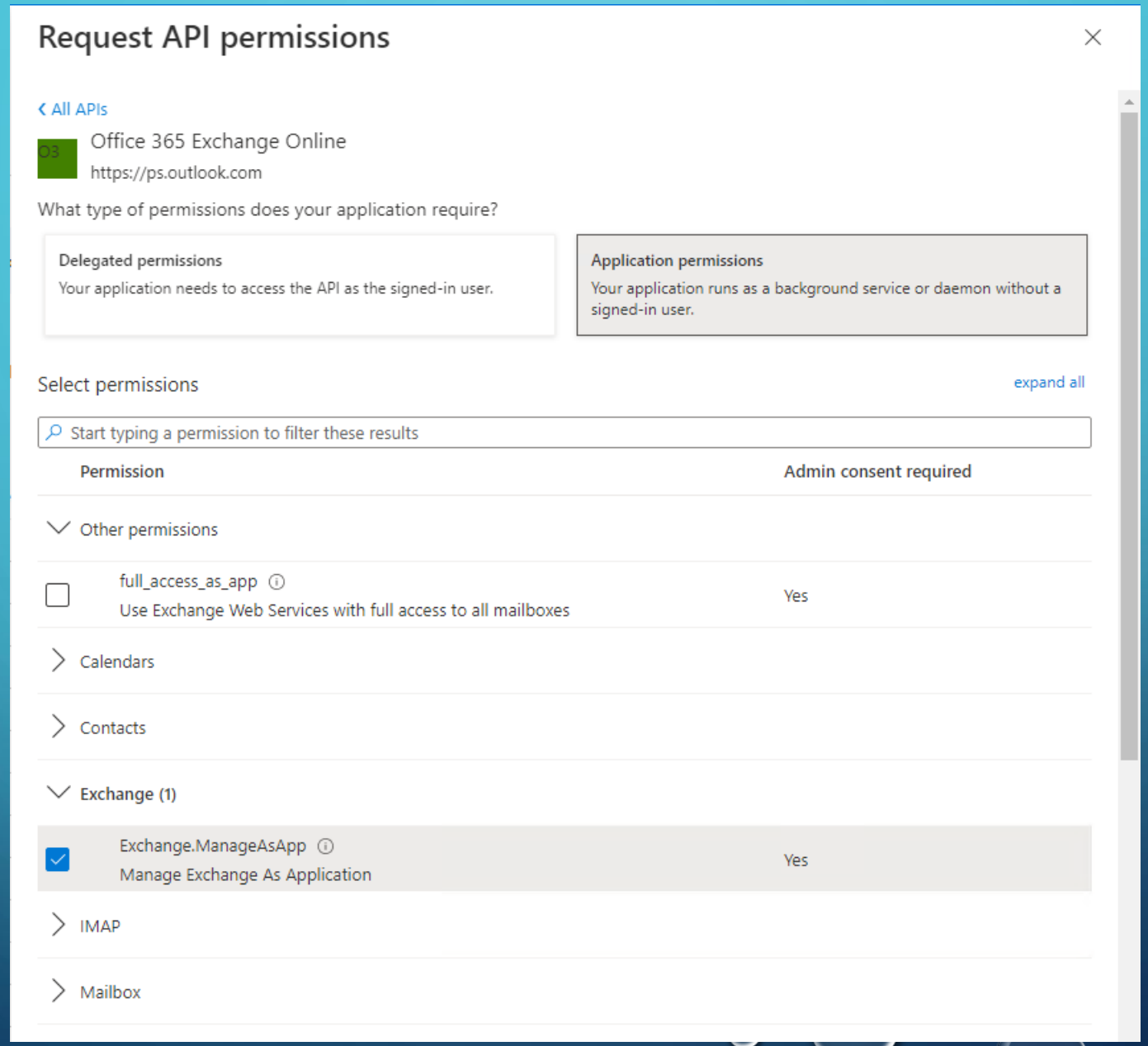
AZURE AD

Steps



AZURE AD

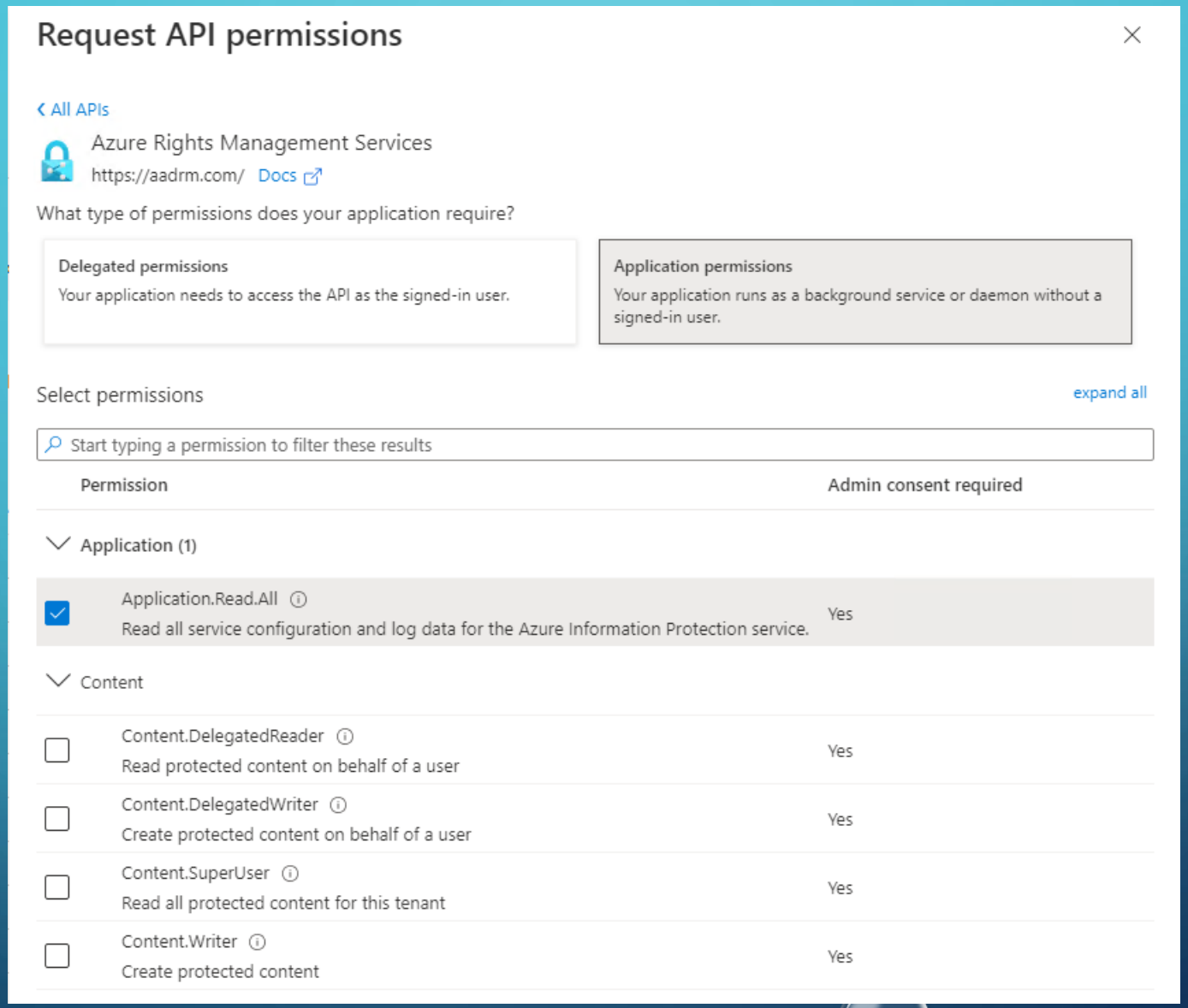
Steps



Microsoft

AZURE AD

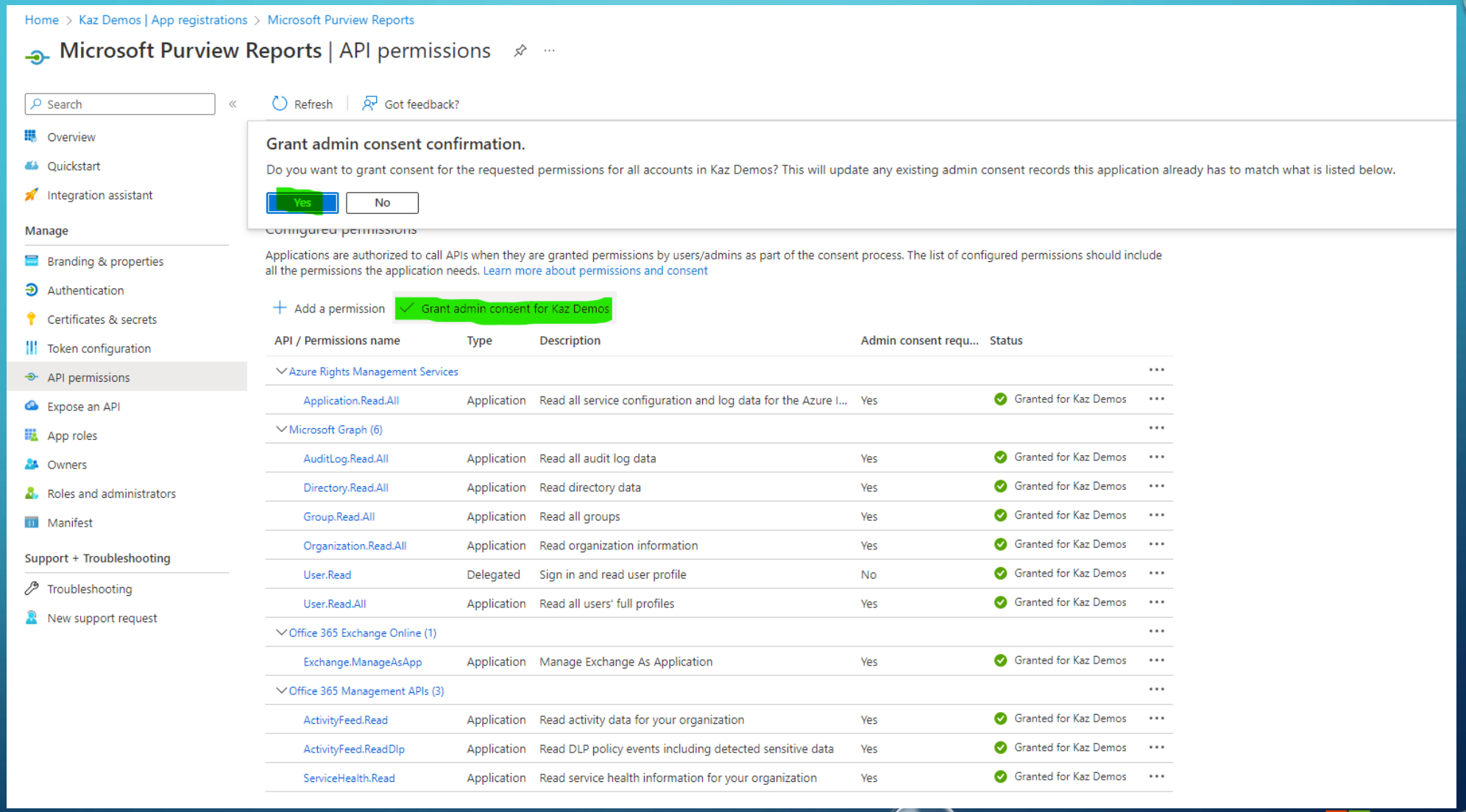
Steps



Microsoft

AZURE AD

Steps



Microsoft

LOGS ANALYTICS WORKSPACE

Create a workspace in Logs Analytics

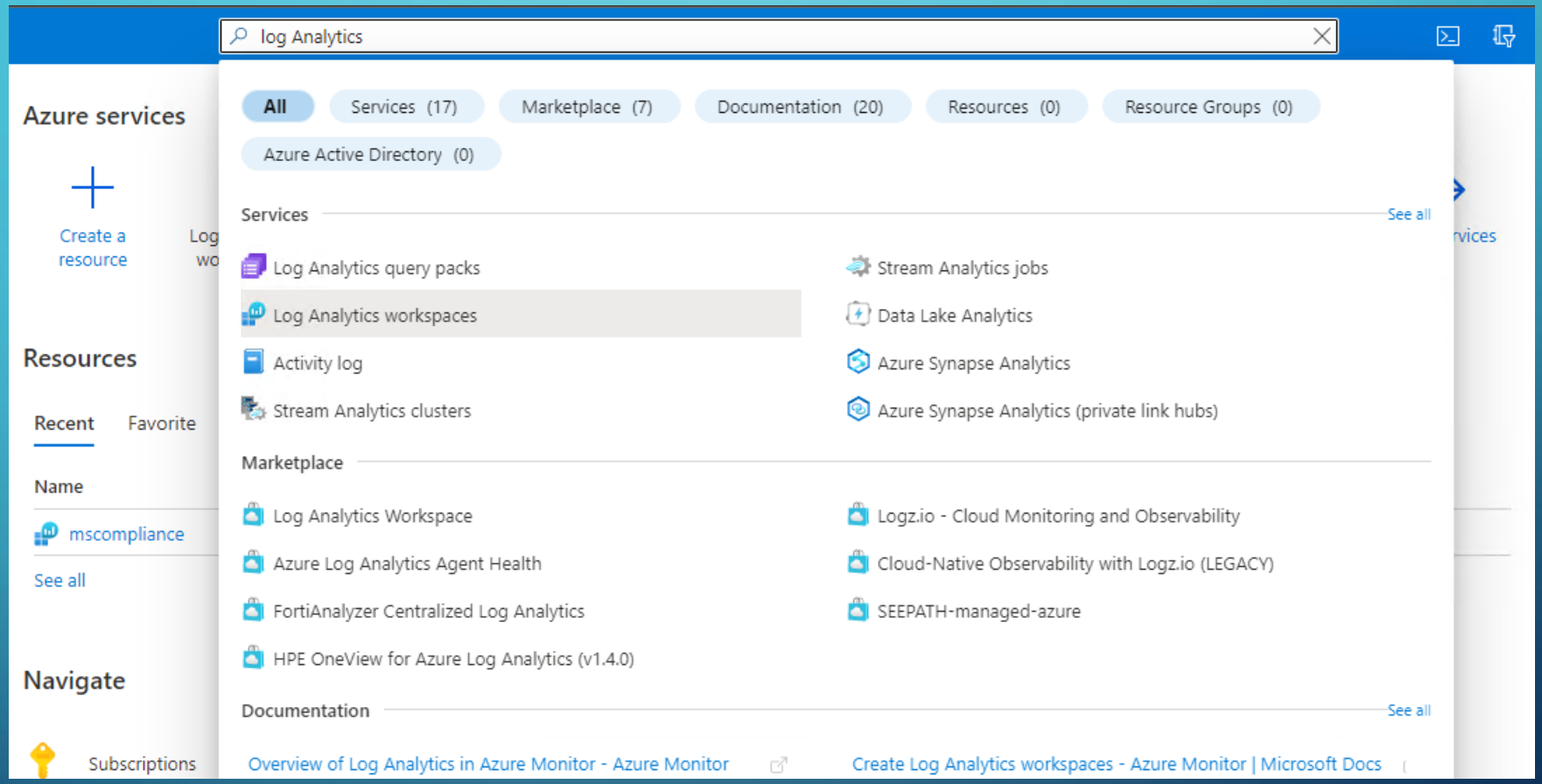
- Open https://portal.azure.com
- Look for Log Analytics workspaces
- Press +Create
  - Select your Azure Subscription
  - Select or create a Resource group
  - Set a Name related to this workspace
  - Select the Region that best match
  - Press Review + Create, wait until a resume is show
  - Press Create
- Wait until the workspace be success deployed
- Open the new workspace
- Go to Agents management menu, and open Log Analytics agent instructions
  - Copy Workspace ID
  - Copy Primary key

To **integrate** with **Sentinel** the same Logs Analytics workspace is needed to use.

Microsoft

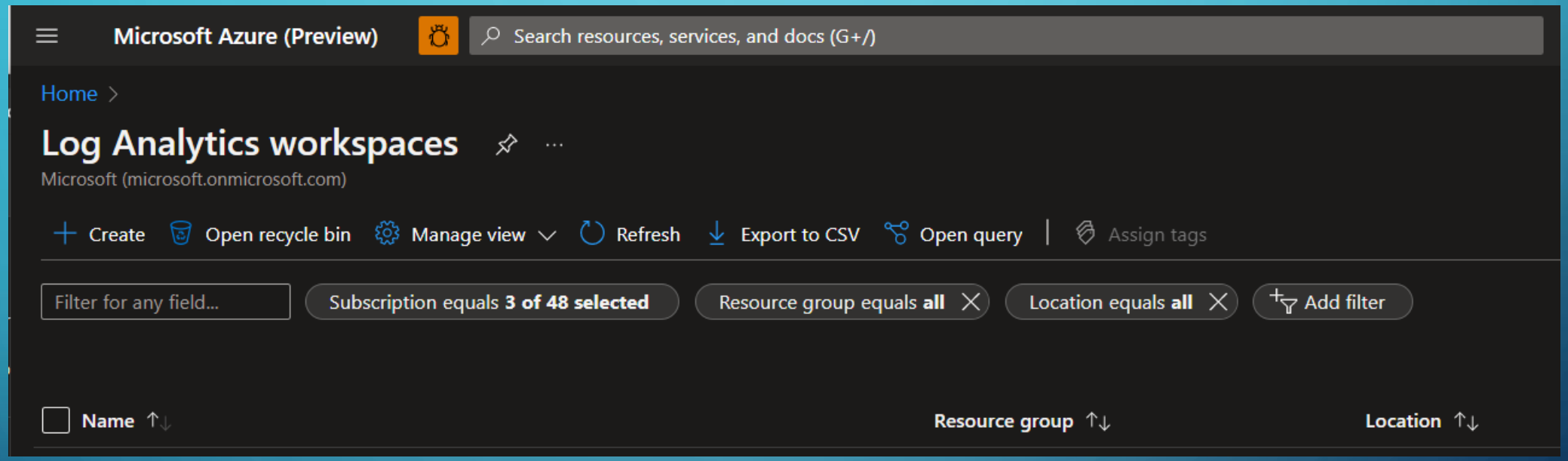
LOGS ANALYTICS WORKSPACE

Steps



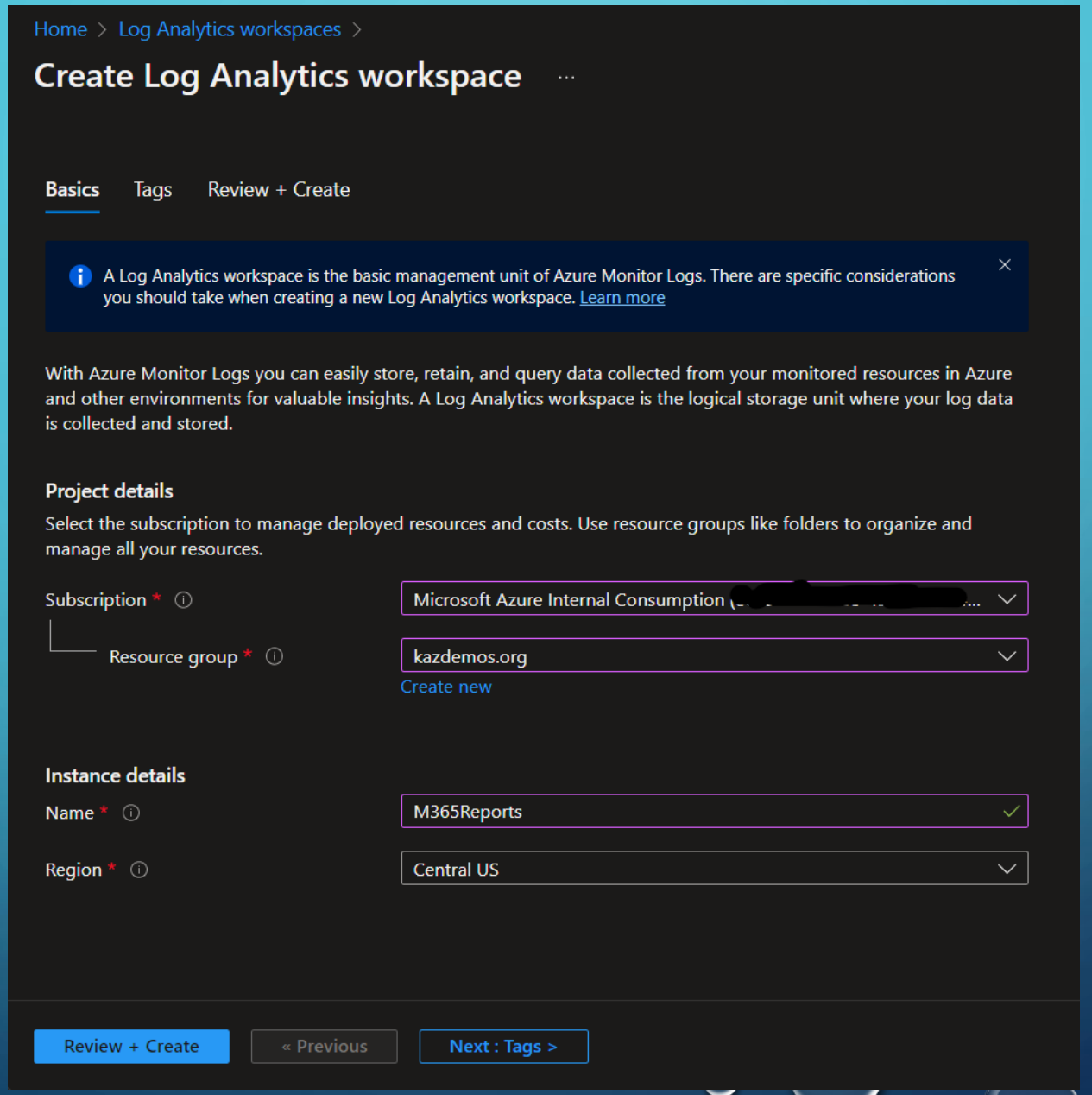
LOGS ANALYTICS WORKSPACE

Steps



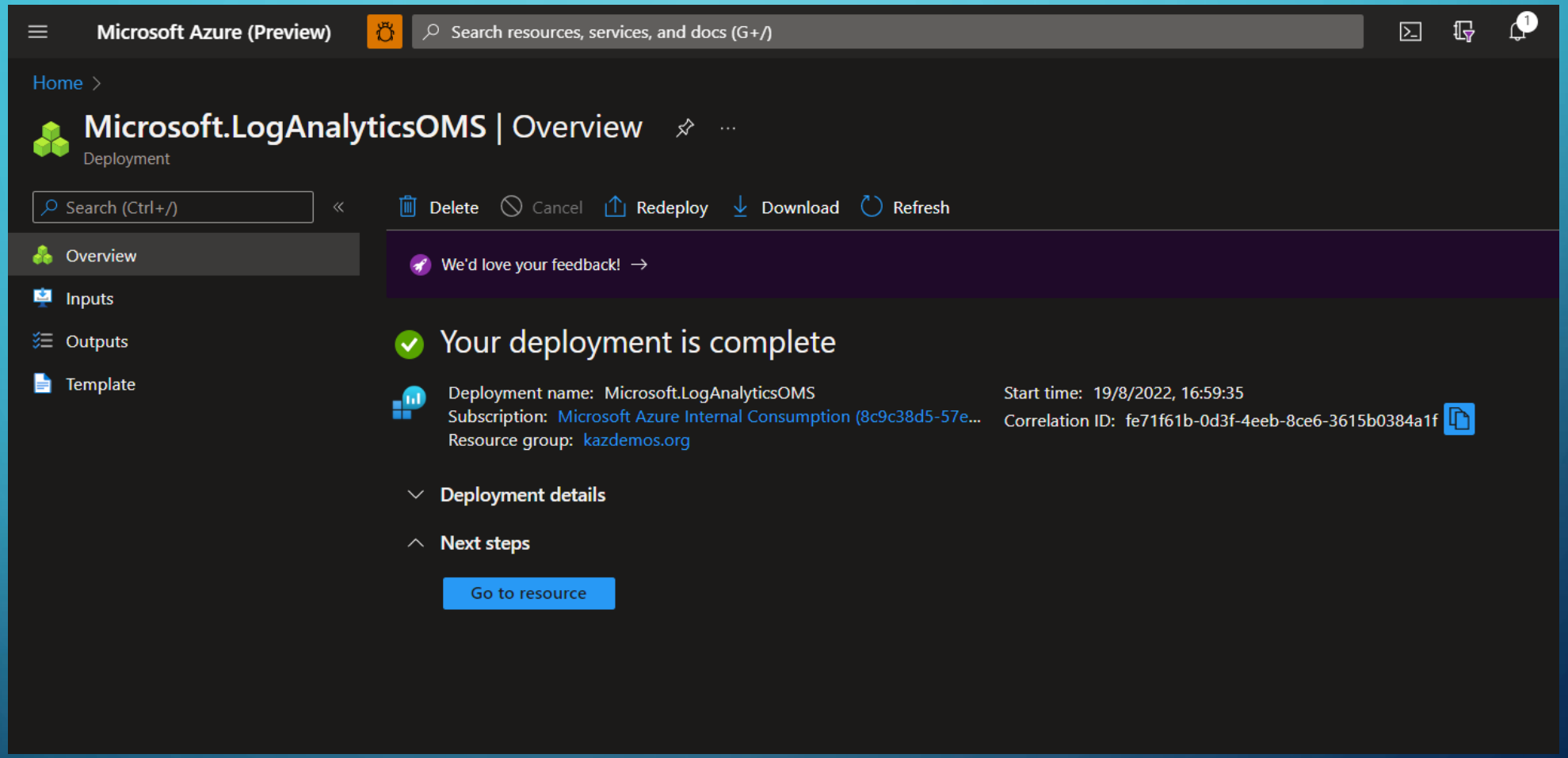
LOGS ANALYTICS WORKSPACE

Steps



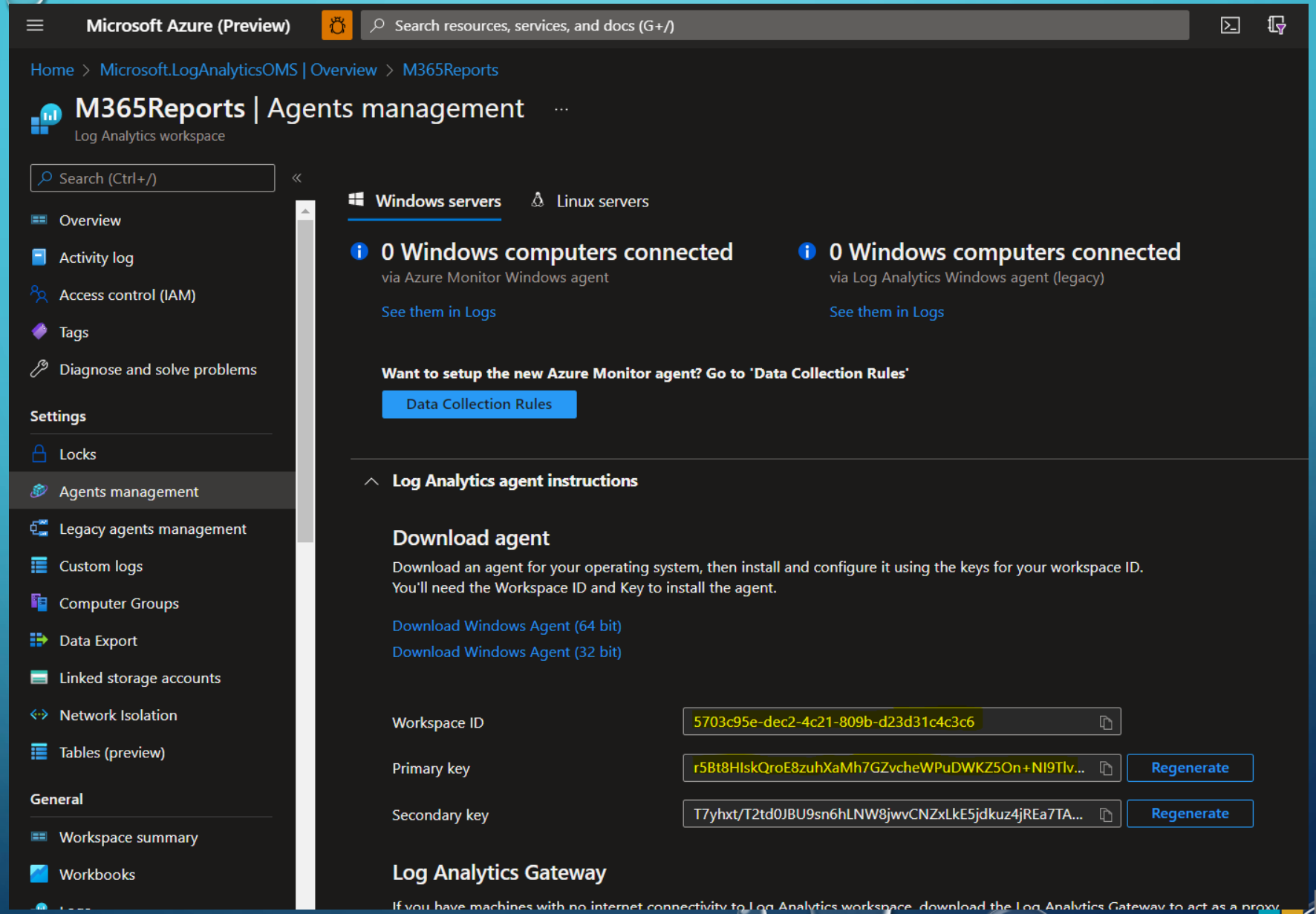
LOGS ANALYTICS WORKSPACE

Steps



LOGS ANALYTICS WORKSPACE

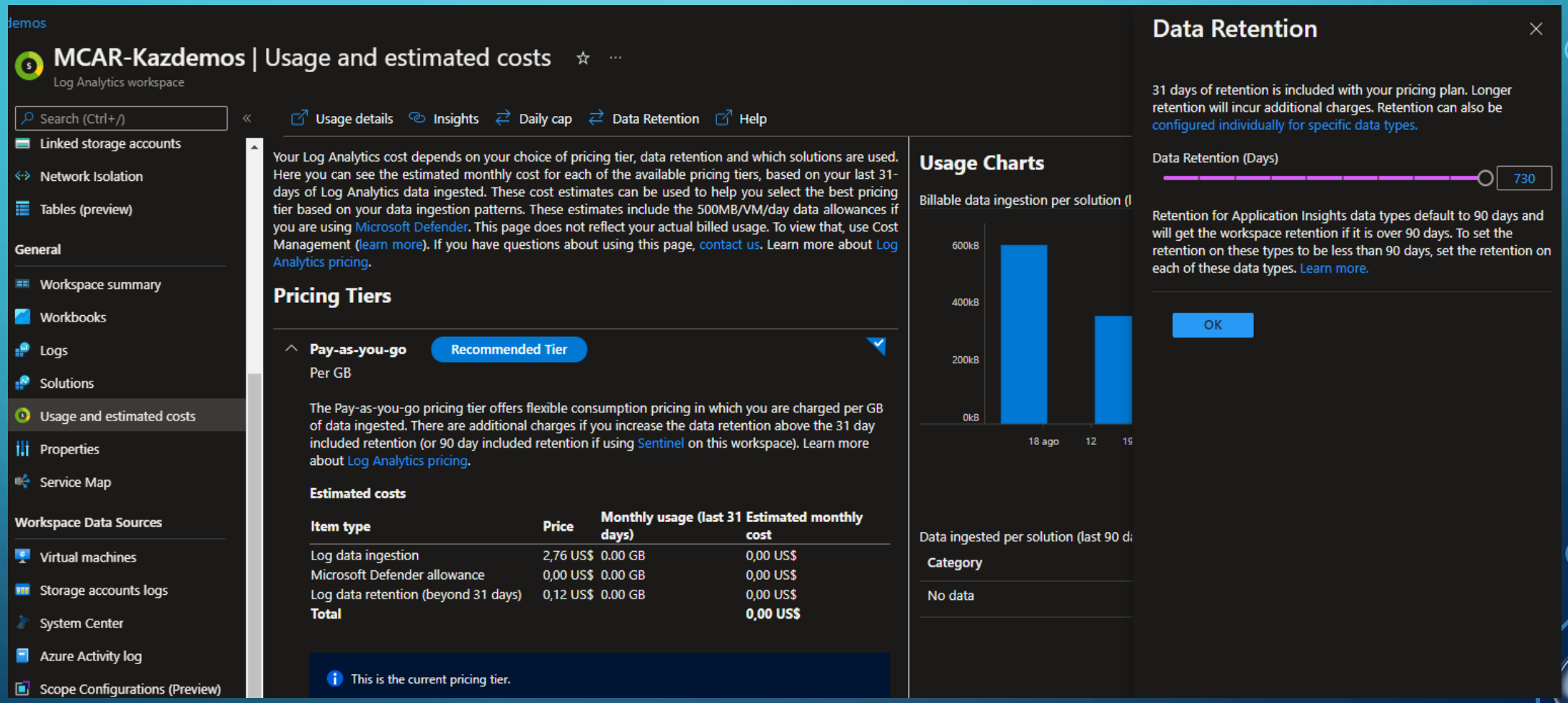
Steps



LOGS ANALYTICS WORKSPACE

Extend your data retention until 2 years

Under General select Usage and estimated costs and then Data Retention, set Data Retention period.



Microsoft

Generate a Certificate to Execute some PowerShell cmdlets unattended

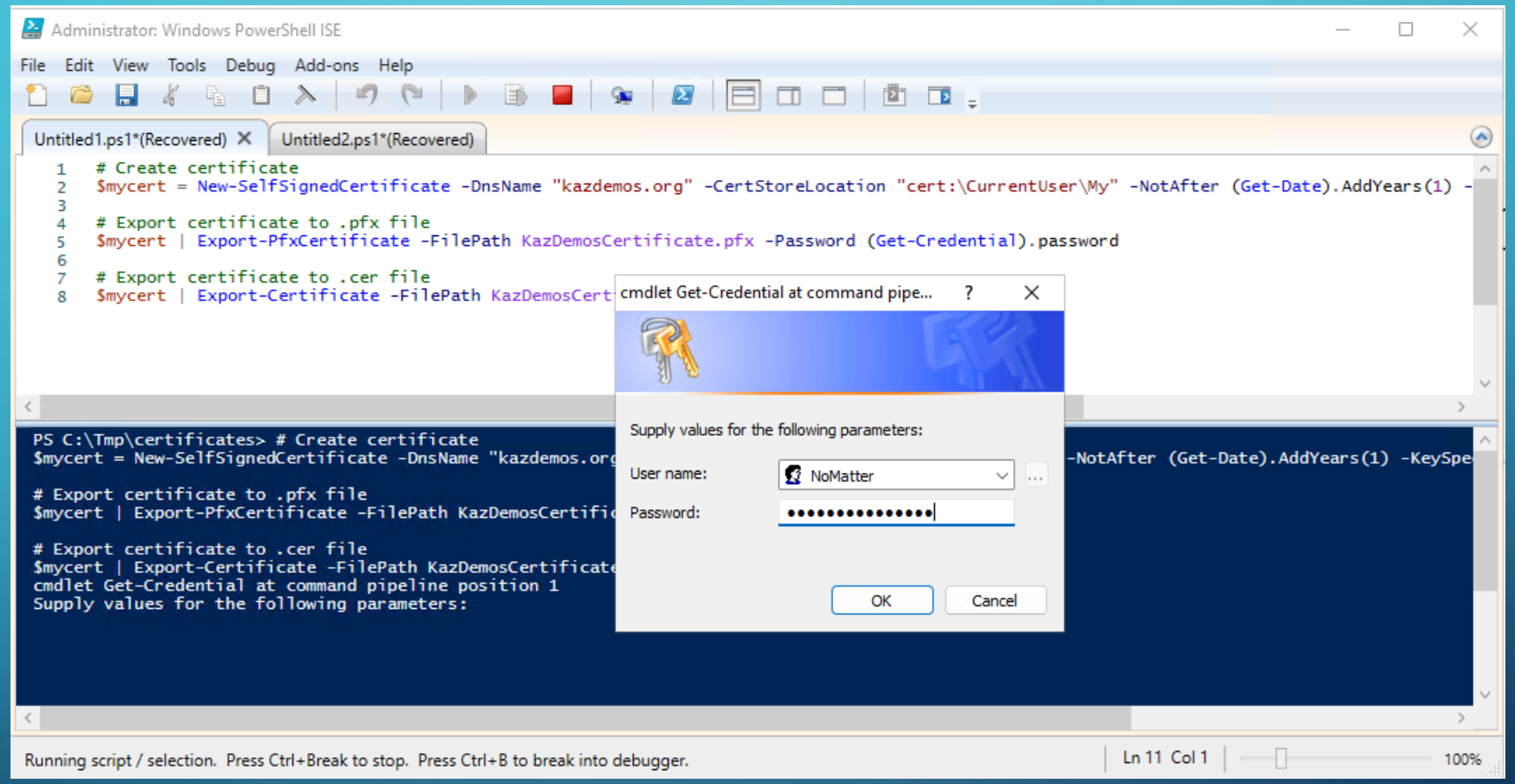
- We will need to **create a certificate** to connect to Information Protection services through Azure AD application, to do that the 1st step is create that certificate:
  - Open an ISE PowerShell console and execute (Select a folder 1st to save the 2 files created, example c:\Tmp):

```
# Create certificate
$mycert = New-SelfSignedCertificate -DnsName "kazdemos.org" -CertStoreLocation "cert:\CurrentUser\My" -NotAfter (Get-Date).AddYears(1) -KeySpec KeyExchange
# Export certificate to .pfx file
$mycert | Export-PfxCertificate -FilePath mycert.pfx -Password (Get-Credential).password
# Export certificate to .cer file
$mycert | Export-Certificate -FilePath mycert.cer
```

  - On the 2nd line a Password will be required for PFX file, PowerShell will ask for user and password, but finally only password will be used (add both)
  - We need to use the same Azure AD application explained here
  - Under the same Azure AD App we need go to “certificates & secrets” menu and select certificates, then press “Upload certificate” here is required the file .cer, a new blade appear, select the folder icon to looking for the file and open the file.
  - After Add copy the Thumbprint value, we will need to add to the laconfig.json file
- A **script to create the certificate is added with the rest of the scripts under the Certs folder**, and called **CreateCertificate.ps1**, this script contains the same previous cmdlets.

CERTIFICATES CREATION

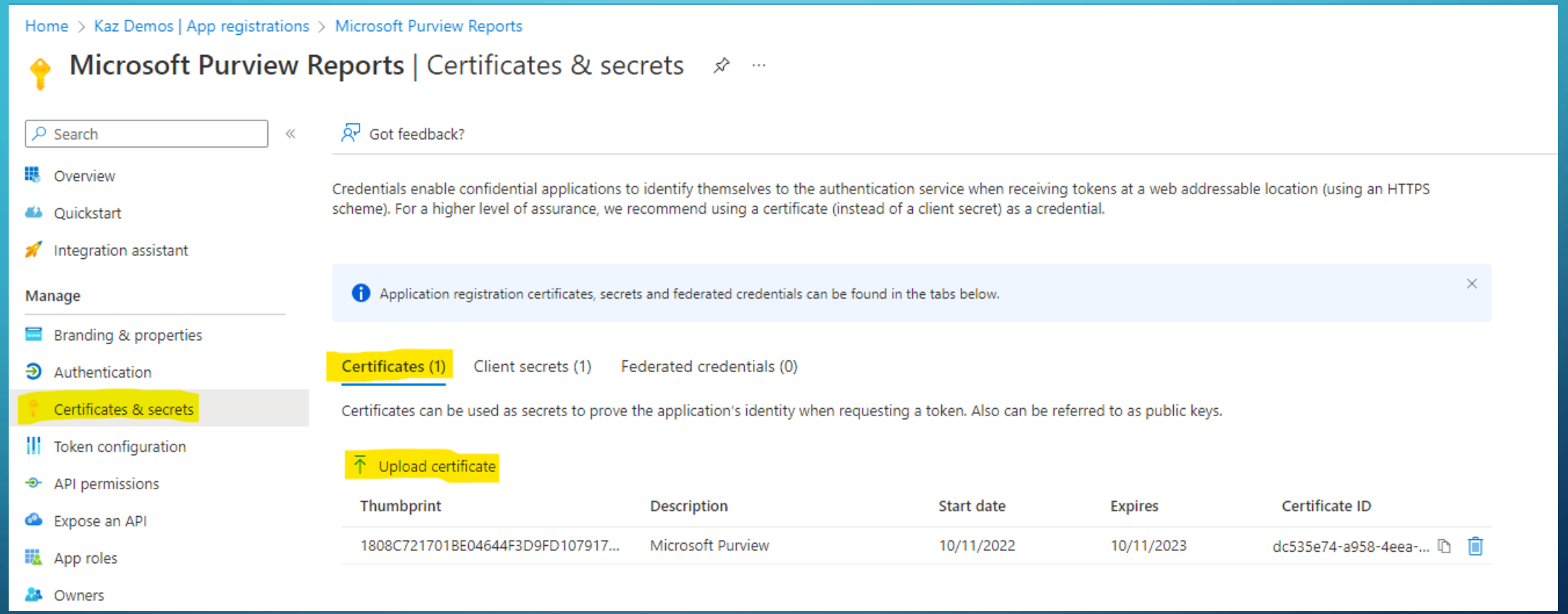
STEPS



Microsoft

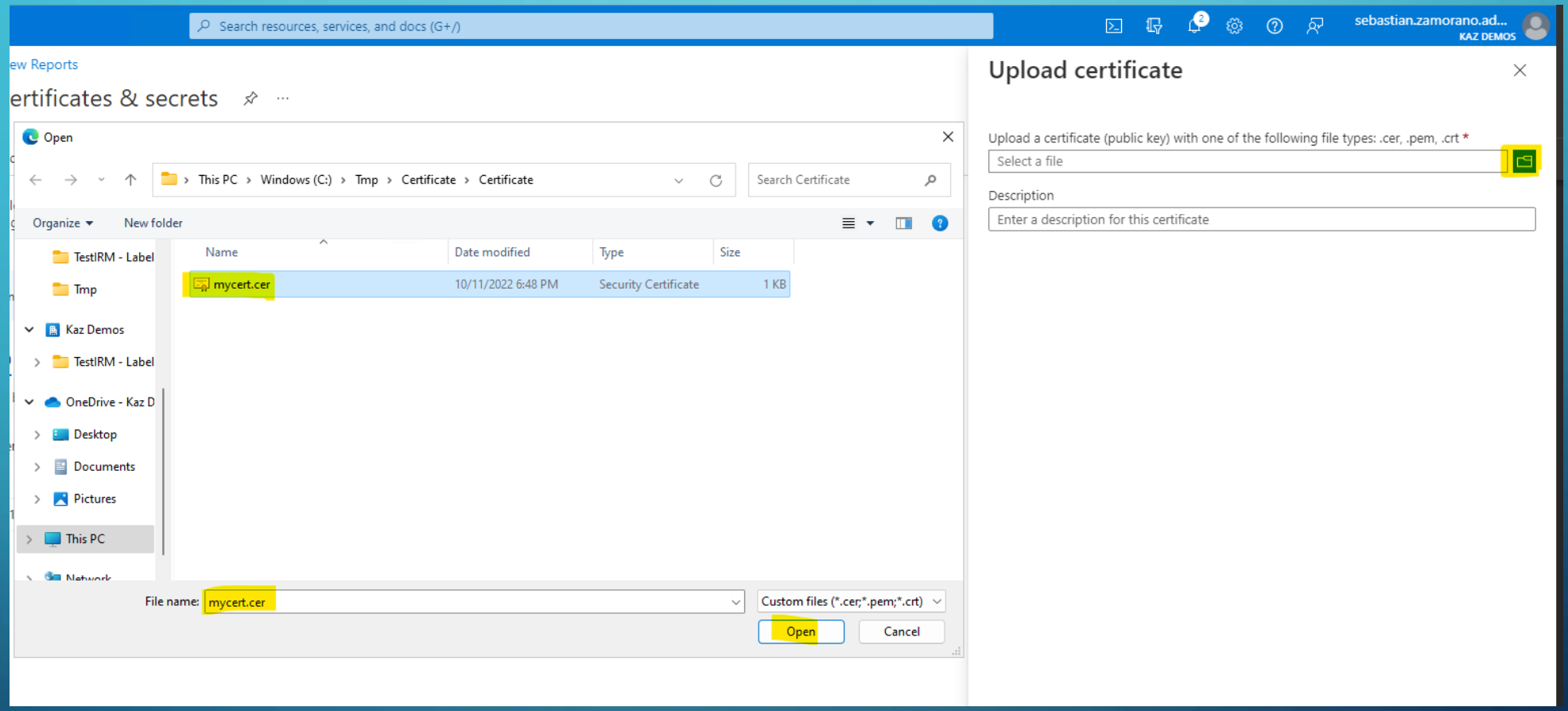
ADD CONFIGURATION TO AZURE AD APP

STEPS



ADD CONFIGURATION TO AZURE AD APP

STEPS



Microsoft

ADD CONFIGURATION TO AZURE AD APP

STEPS

Certificates (1) | Client secrets (1) | Federated credentials (0)

Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

Upload certificate

Thumbprint	Description	Start date	Expires	Certificate ID
1808C721701BE04644F3D9FD107917...	Microsoft Purview	10/11/2022	10/11/2023	dc535e74-a958-4eea-...

new 1 | new 2 | my_kazdemos_labels.csv | new 3 | new 4 | run_me.ps1 | new 5 | Get-auditLogs.ps1 | export_logs.ps1 | Get-LabelData.ps1 | laconfig.json | ExportCSV2LA.ps1

```
{
    "EncryptedKeys": "True",
    "AppClientID": "[redacted]",
    "ClientSecretValue": "01000000d08c9ddf0115d1118c7a00c04fc297eb010000009be602676a73ec489a1075b1c814e1e3000000
    "TenantGUID": "[redacted]",
    "TenantDomain": "kazdemos.org",
    "LA_CustomerID": "[redacted]",
    "LA_SharedKey": "01000000d08c9ddf0115d1118c7a00c04fc297eb010000009be602676a73ec489a1075b1c814e1e30000000002
    "CertificateThumb": "01000000d08c9ddf0115d1118c7a00c04fc297eb010000009be602676a73ec489a1075b1c814e1e30000000
    "OnmicrosoftURL": "[redacted]"
}
```

In this case the certificate thumbprint was encrypted using the steps explained in this slide

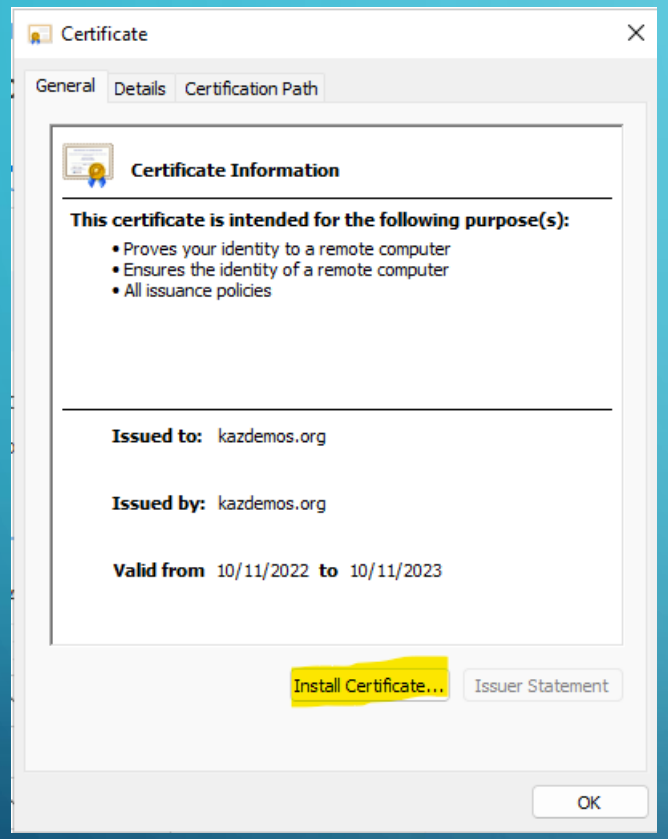
Microsoft

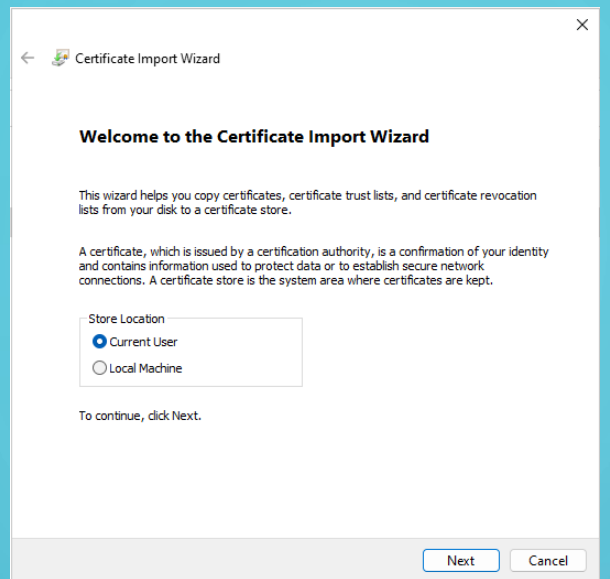
To execute the script using the certificate

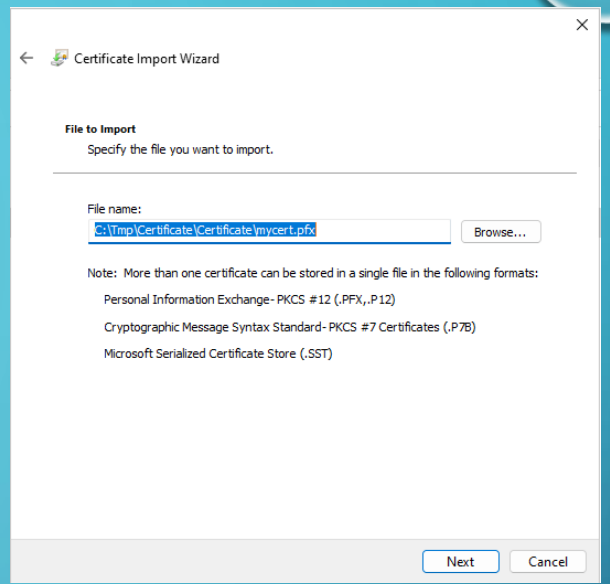
- On the same computer used to execute the scripts is needed to install both certificates previously created
- The certificates needs to be installed on the same account used to execute these scripts:
  - Get-AzureADData.ps1
  - Get-AzureADDomains.ps1
  - Get-AzureADroles.ps1
  - Get-LabelData.ps1
  - Get-RMSData.ps1(This one can be executed only using PowerShell 5)
- To install, is required put both certificates on the machine used to execute the scripts and execute both doing double click.
- For each one is required to install the certificate locally.

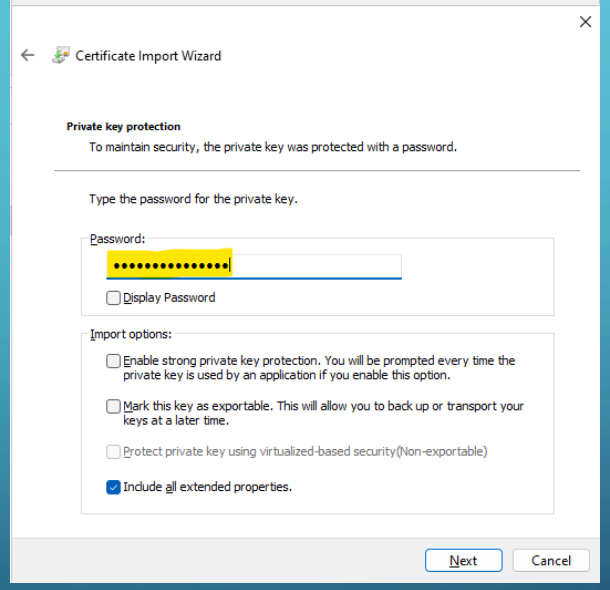
INSTALL CERTIFICATE LOCALLY

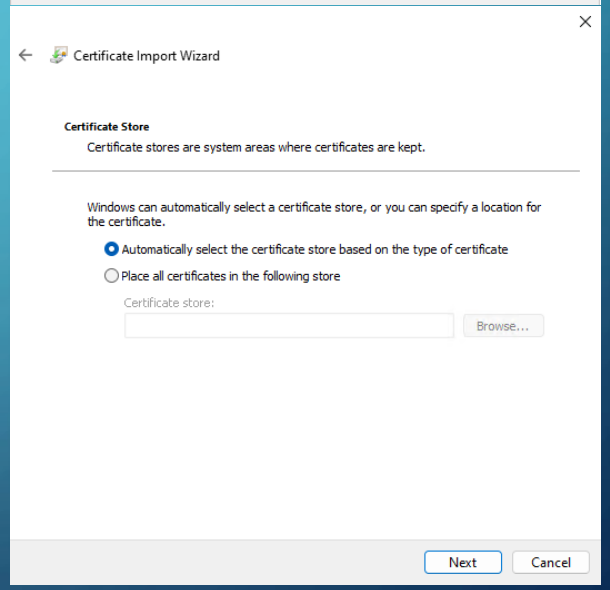
STEPS

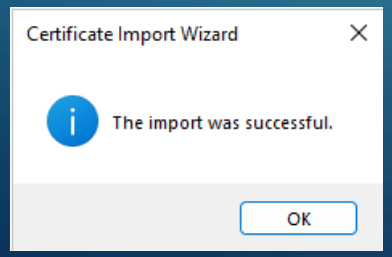












Microsoft

COMPLIANCE ADMIN ROLE FOR SCRIPT

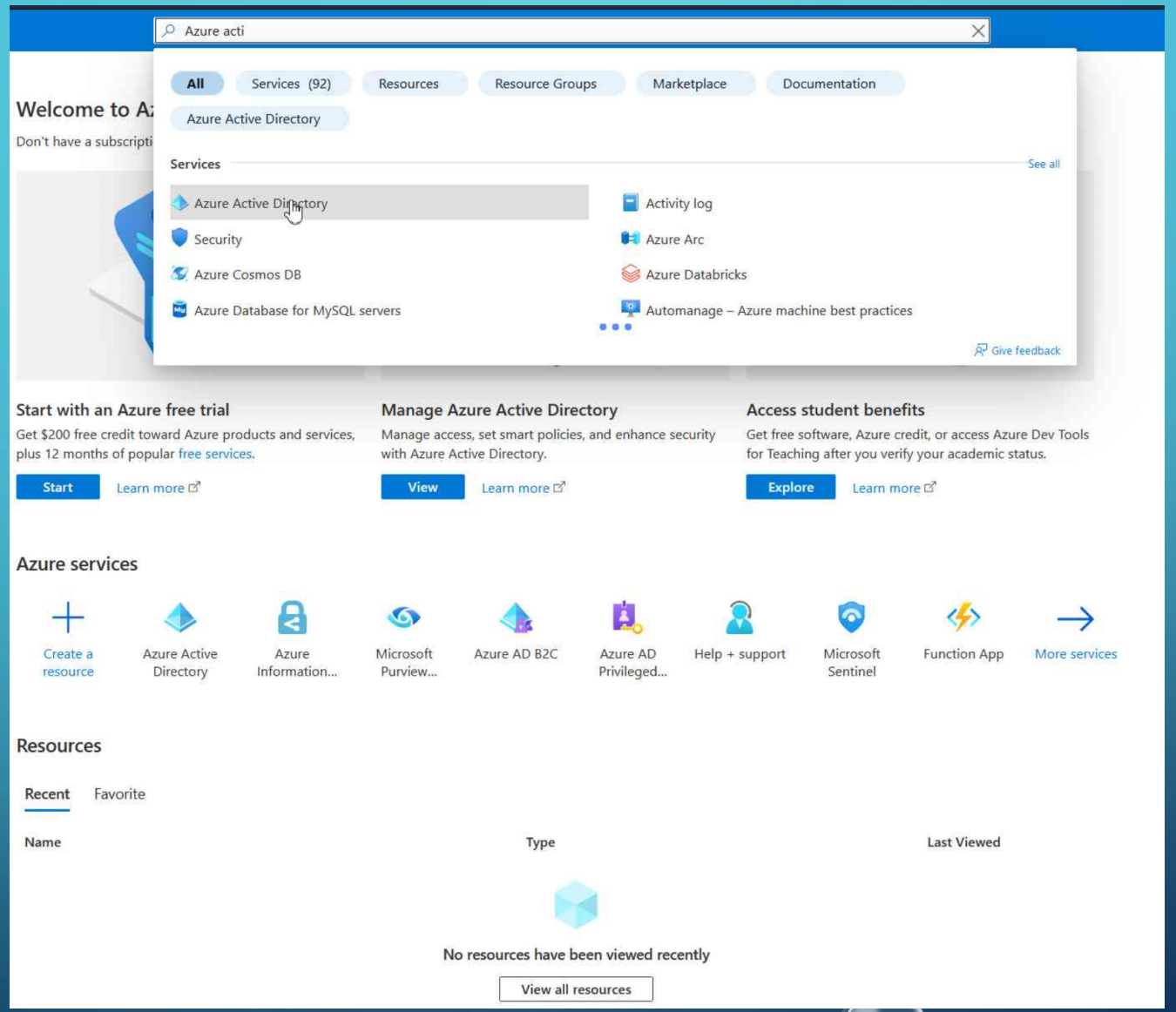
Compliance administrator role assigned to Azure AD App

- To execute Get-LabelData.ps1 script additional permissions are required:
  - To give the permissions is required open Azure Active Directory
  - Then go to “Roles and administrators” menu
  - Search for Compliance Administrator and press the name
  - In the new window click over “+ Add assignments”
  - Under Add assignments interface, press under “Select member(s)” and looking for the name of the Azure AD App previously created, click it over the name and press select
  - Press the Next button, and in the new interface select “Active” under Assignment type, maintain check the option “Permanently assigned” and provide a justification to enable the “Assign” button
- Understanding the complexity to give elevate privileges to an unattended script, a line can be modified on the script to a manual execution, in this case someone with Compliance Administrator role needs to execute this script on PowerShell, this execution is on-demand and is required only when you have modified or new labels, to up to date the reports with this information, or can be required depending the retention period in the Logs Analytics workspace.

```
154
155 Function Export-LabelData() {
156     # ---------------------------------------------------------------
157     #    Name           : Export-LabelData
158     #    Desc           : Extracts data from Get-Label into Log analytics workspace tables for reporting purposes
159     #    Return         : None
160     # ---------------------------------------------------------------
161         <#
162         .NOTES
163         If you cannot add the "Compliance Administrator" role to the Azure AD App, for security reasons, you can comment the line 167 and uncomment the line 166, in that case
164         Someone with "Compliance Administrator" role needs to execute this script, this script is executed on-demand to refresh the label names
165         #>
166         #Connect-IPPSSession
167         Connect-IPPSSession -CertificateThumbPrint $CertificateThumb -AppID $AppClientID -Organization $OnmicrosoftTenant
```

COMPLIANCE ADMIN ROLE FOR SCRIPT

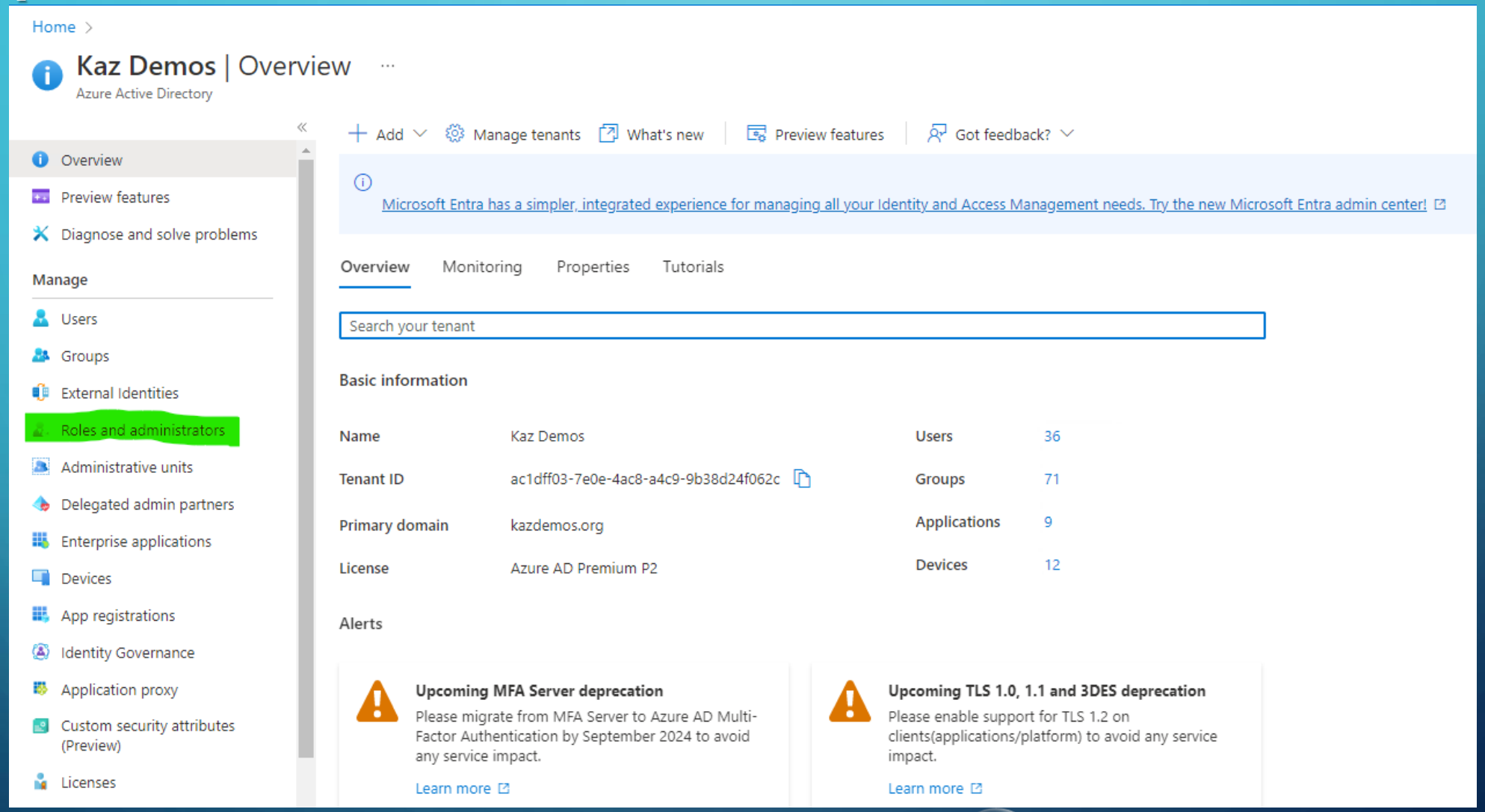
Steps



Microsoft

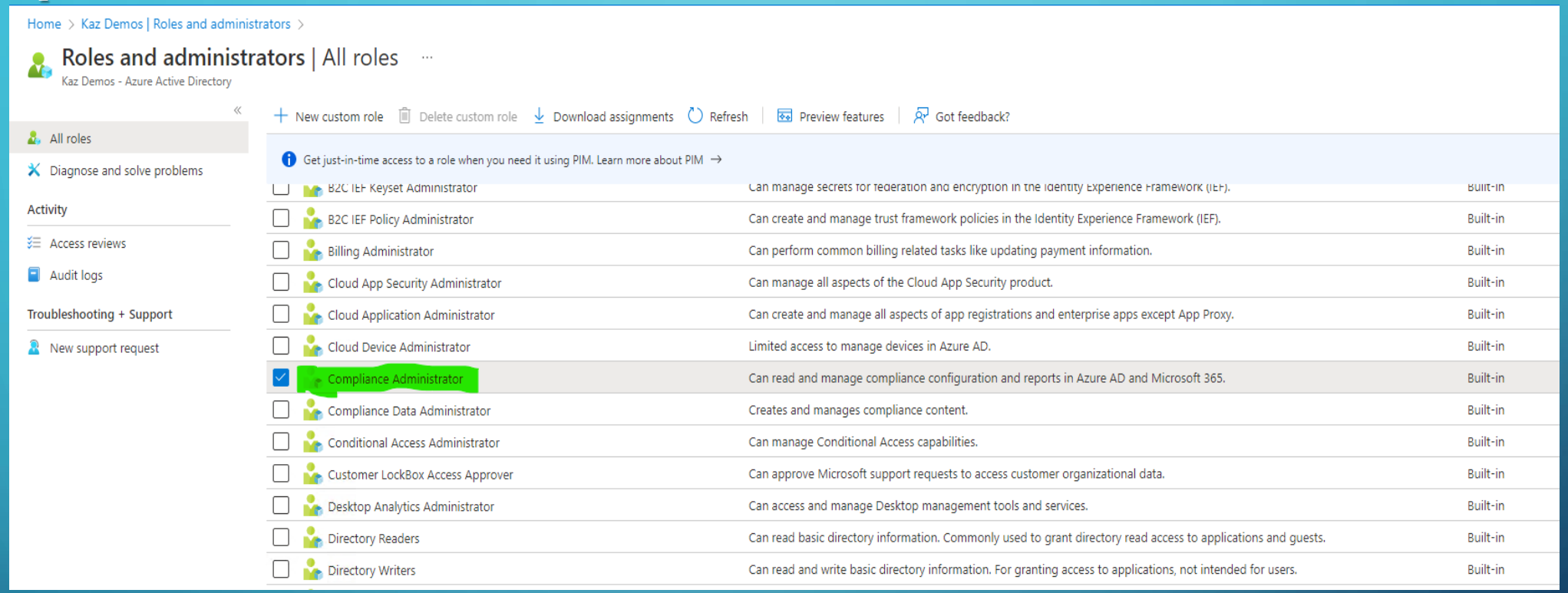
COMPLIANCE ADMIN ROLE FOR SCRIPT

Steps



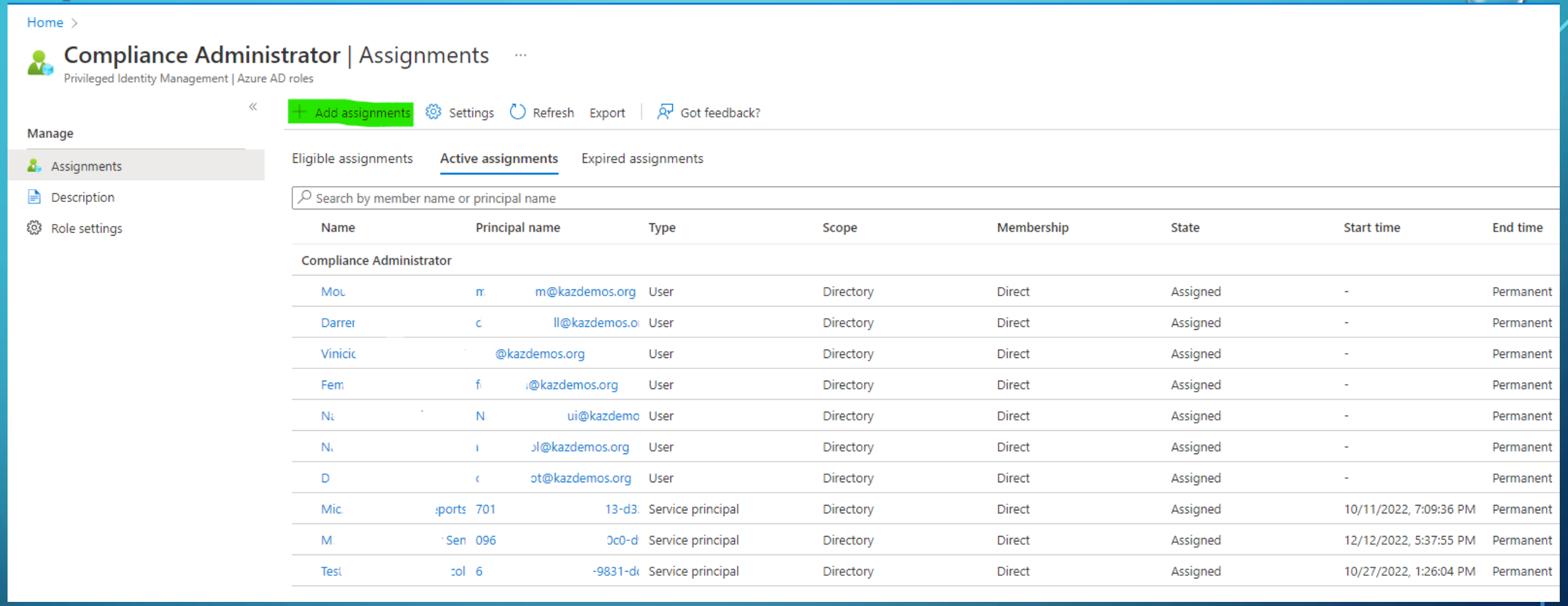
COMPLIANCE ADMIN ROLE FOR SCRIPT

Steps



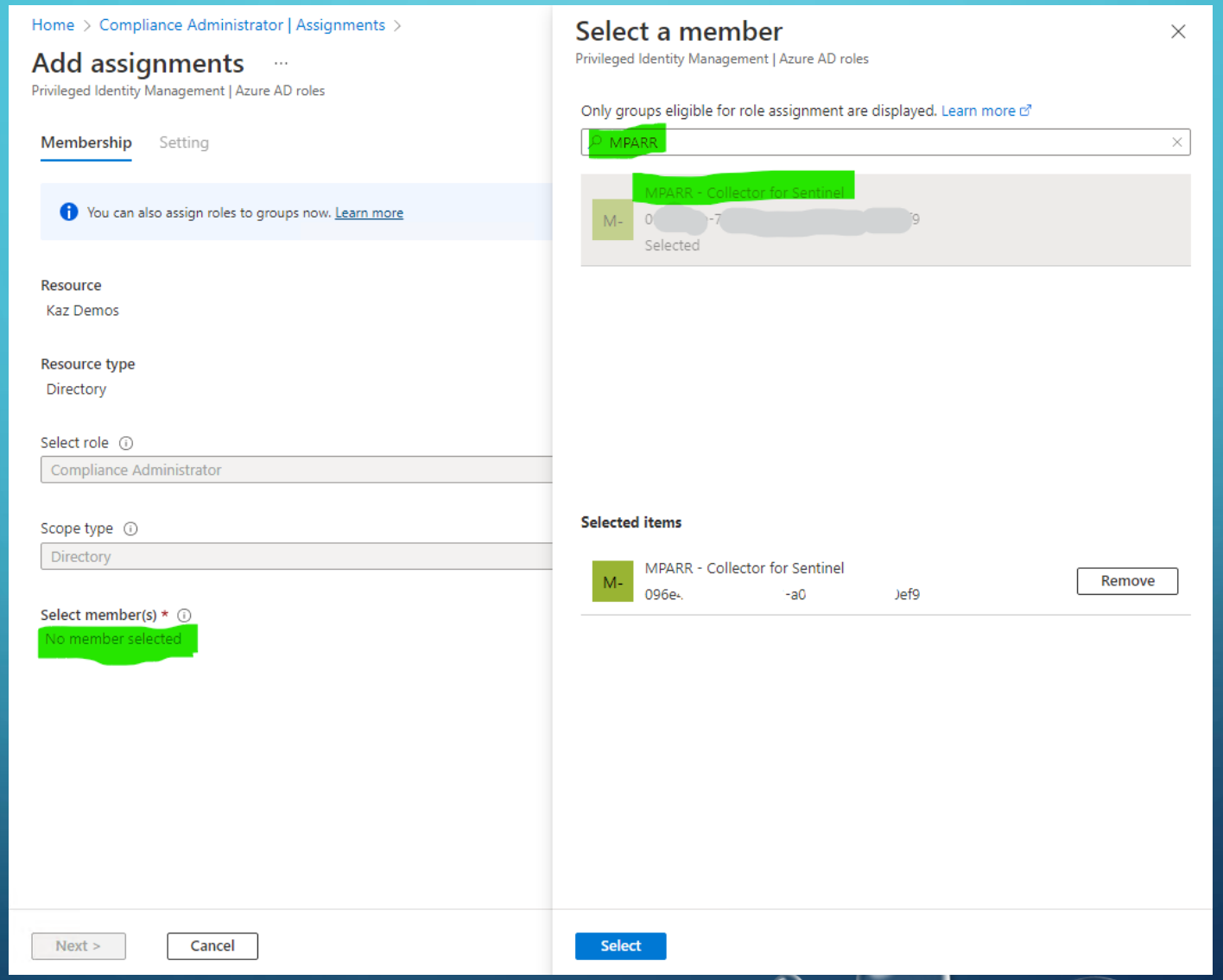
COMPLIANCE ADMIN ROLE FOR SCRIPT

Steps



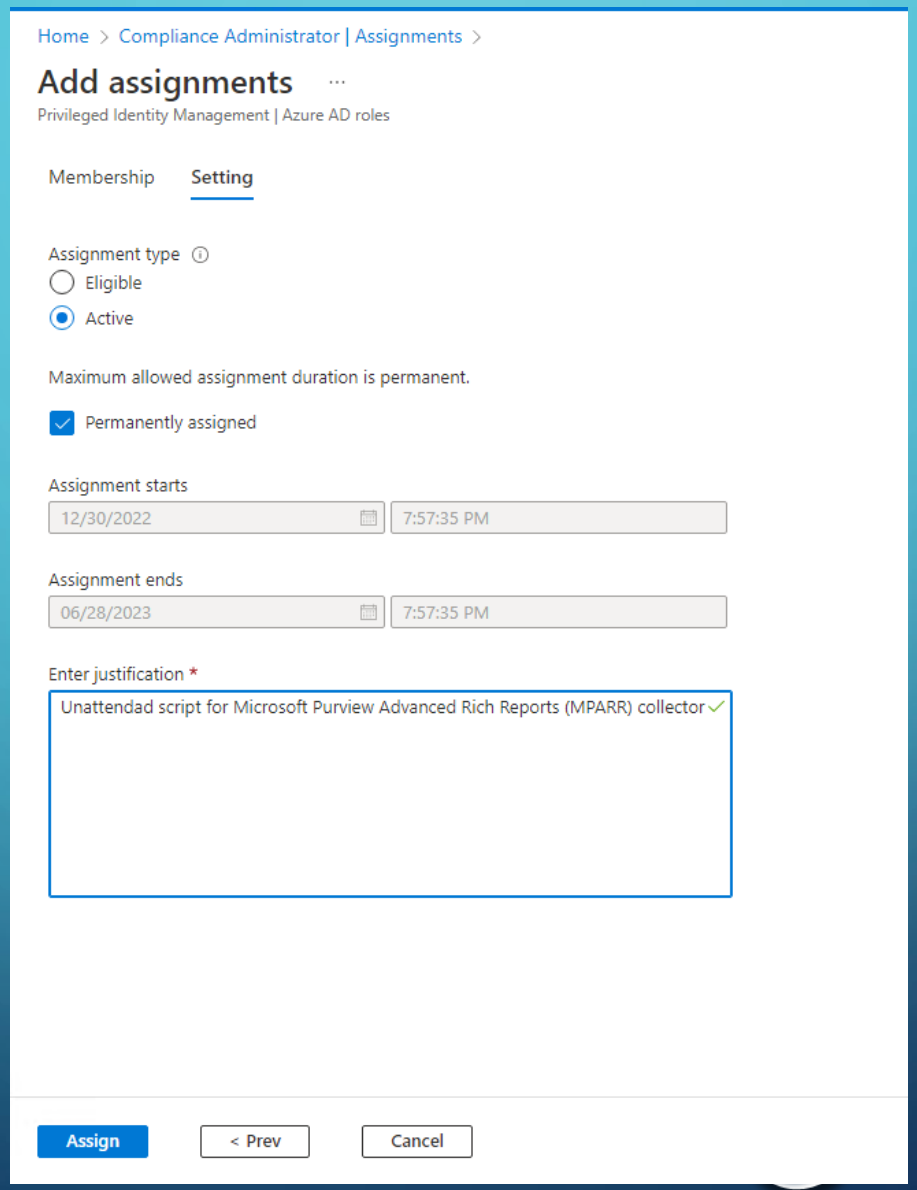
COMPLIANCE ADMIN ROLE FOR SCRIPT

Steps



COMPLIANCE ADMIN ROLE FOR SCRIPT

Steps



WORKSTATION OR SERVER

Configure the script

- Install **PowerShell v7**
- Create a local folder under C:\ like as "**Collector**"
- Unzip the file "**MPARR Collector.zip**" in that folder
- Open the file "**laconfig.json**" and complete with this information:
  - **EncryptedKeys:** "False" (To use check here, this can be used to encrypt the Keys stored in this file)
  - **AppClientID:** Application (client) ID from app registration step
  - **ClientSecretValue:** Secret key from Certificates and secrets under app registration step
  - **TenantGUID:** Tenant ID from app registration step
  - **TenantDomain:** Principal domain in the tenant
  - **LA_CustomerID:** Workspace ID from Logs Analytics (here can be used Sentinel Workspace)
  - **LA_SharedKey:** Primary key from Logs Analytics (here can be used Sentinel Workspace)
  - **Cloud:** used to select the kind of tenant, this permit uses the right URL for the collector (see notes)
  - **CertificateThumb:** Certificate thumbprint from a self signed certificate
  - **OnmicrosoftURL:** Tenant domain in format <tenant.onmicrosoft.com>
  - **RMSLogs:** This will be used for the Get-RMSdata script to capture information from AIP Service API
  - **OutPutLogs:** Select where your Logs will be recorded
- Save the changes

Microsoft

WORKSTATION OR SERVER

Configure the script

- On the same folder
- Open the file "**schemas.json**" and make changes if you want to avoid download certain kind of information:
  - According to this information, All the information collected on the different Office 365 Management API schemas are collected on 5 content blobs, these are:
    - Audit.AzureActiveDirectory
    - Audit.Exchange
    - Audit.SharePoint
    - Audit.General (includes all other workloads not included in the previous content types)
    - DLP.All (DLP events only for all workloads)
  - To make any changes to avoid certain information the value "True" need to changed to "False"
- In the new version of "schemas.json" new filtering capabilities was added to the file, for each Content Blob named previously can be set a filter for "Contains" or "NotContains", previously the filter only worked with "Contains", in this case if some operation is not wanted using "NotContains" this can be avoided. This is only on the case that you want to use any of these parameters with the MPARR Collector script:
  - FilterAuditAzureActiveDirectory
  - FilterAuditExchange
  - FilterAuditSharePoint
  - FilterAuditGeneral
  - FilterDLPAll
- Save the changes

Microsoft

WORKSTATION OR SERVER

Configure task scheduler

- Open **Task Scheduler**
- Go to Task Scheduler Library
- Under Actions click **Create Task***
  - On General tab
  - Set a Name
  - Under Security options change to Run whether users is logged on or not
  - On Triggers tab
  - On Settings select Daily
  - On Advanced settings set Repeat task every in this case 10 minutes, can be set every hour, and for a duration of Indefinitely
  - On Actions tab
  - Click New…
  - Under Program/script looking for PowerShell 7 application, normally is in this path "C:\Program Files\PowerShell\7\pwsh.exe“
  - Under Add arguments select the path to the script like as "C:\Collector\Sentinel\MPARR_Collector.ps1“**
  - Select OK and local credentials will be required
  - The script can be executed pressing Run option

*Using the same process can be created a new task under task scheduler to execute the PowerShell script ***Get-RMSData.ps1***, is required to execute on PowerShell v5*

** *Previously was used run_me.ps1 as principal script used to add the parameter “OutputPath” now is included on laconfig.json file*

Microsoft

WORKSTATION OR SERVER

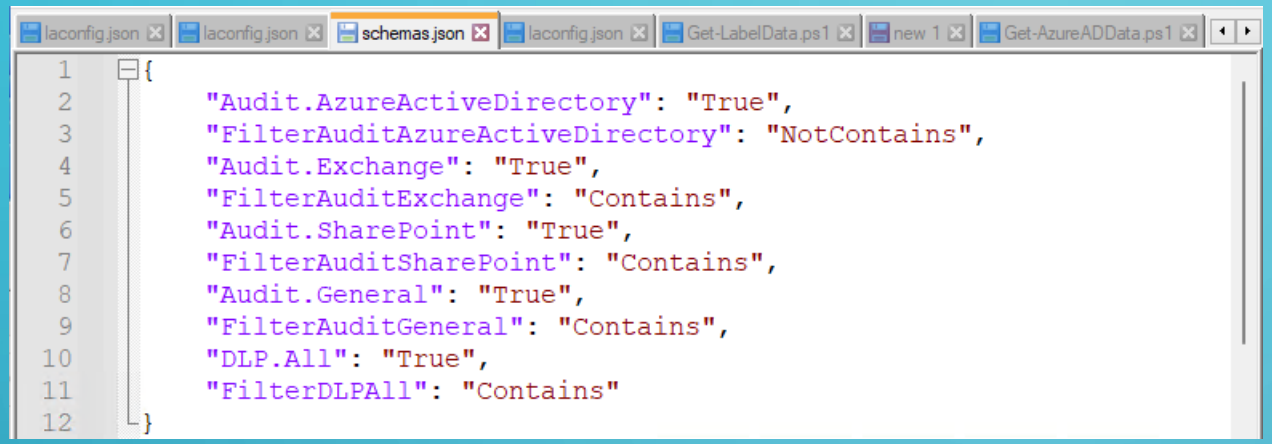
Steps

```
{
    "EncryptedKeys": "False",
    "AppClientID": "7(      :f-2. .5-4  f-8  .3-c          20",
    "ClientSecretValue": "i5o                                  jRZEVbKs",
    "TenantGUID": "a       3-7  e-4  8-a  9-9          :2c",
    "TenantDomain": "kazdemos.org",
    "LA_CustomerID":  "5     d4c-7]  :-4:  d-8  :f-(          )4",
    "LA_SharedKey":  "kN                                 HDKC                                                    )l4dw==",
    "Cloud": "Commercial",
    "CertificateThumb": "18                                                    0F4179BCD9",
    "OnmicrosoftURL": "M3          .onmicrosoft.com",
    "RMSLogs": "c:\\Collector\\Script2.0\\RMSLogs\\",
    "OutPutLogs": "Logs\\"
}
```

Microsoft

WORKSTATION OR SERVER

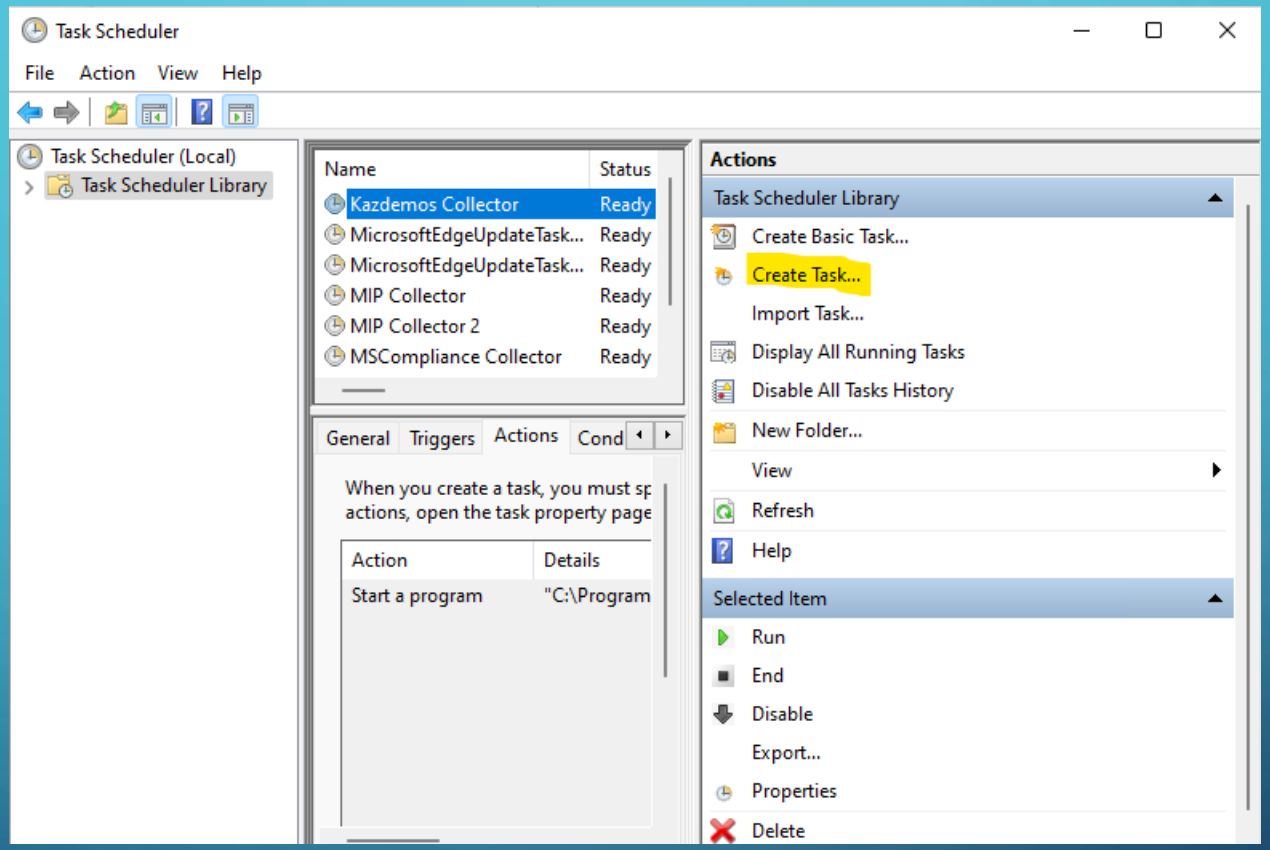
Steps



Microsoft

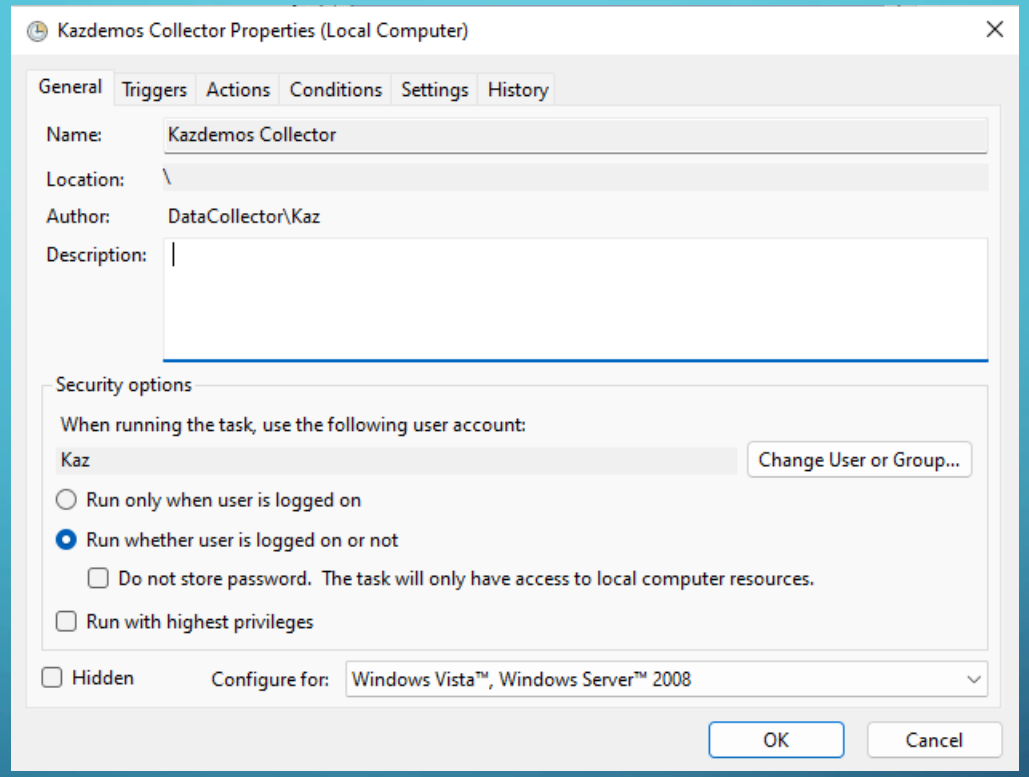
WORKSTATION OR SERVER

Steps



WORKSTATION OR SERVER

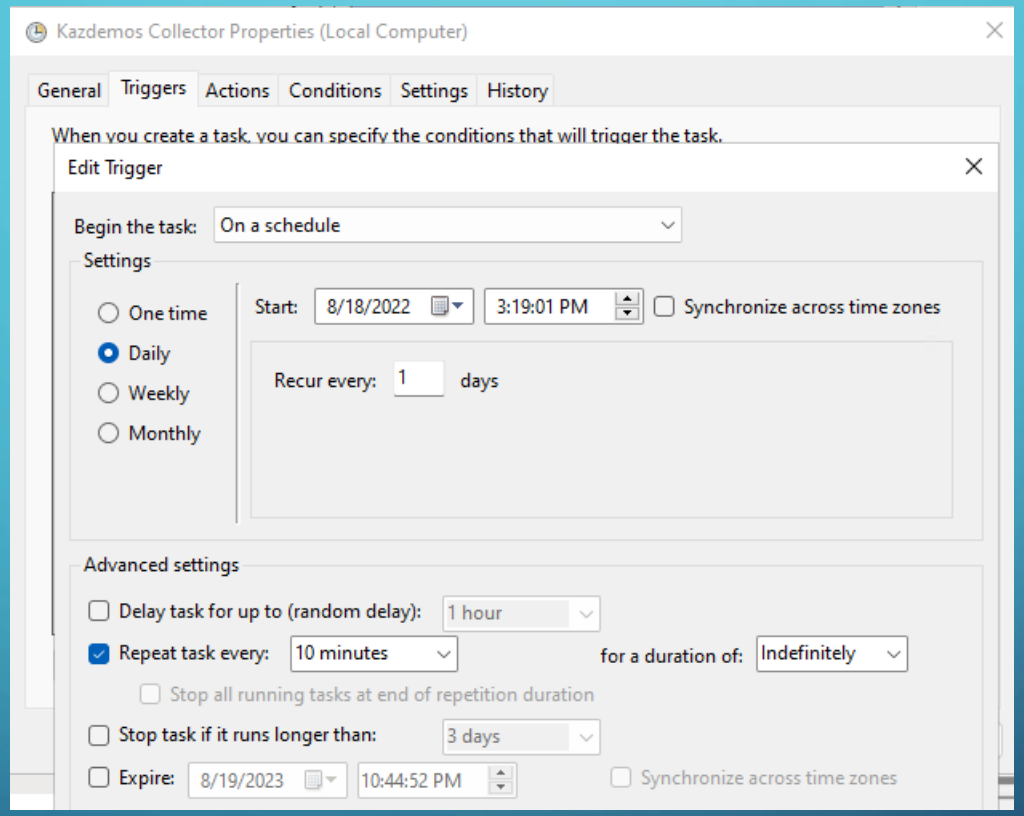
Steps



Microsoft

WORKSTATION OR SERVER

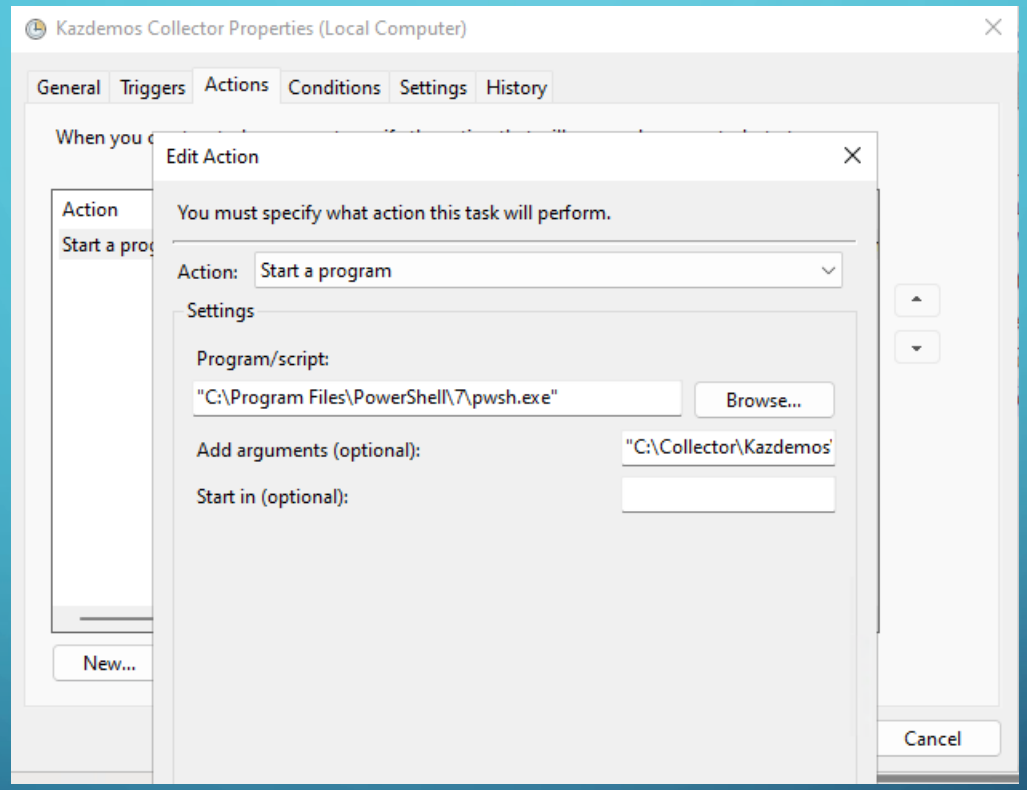
Steps



Microsoft

WORKSTATION OR SERVER

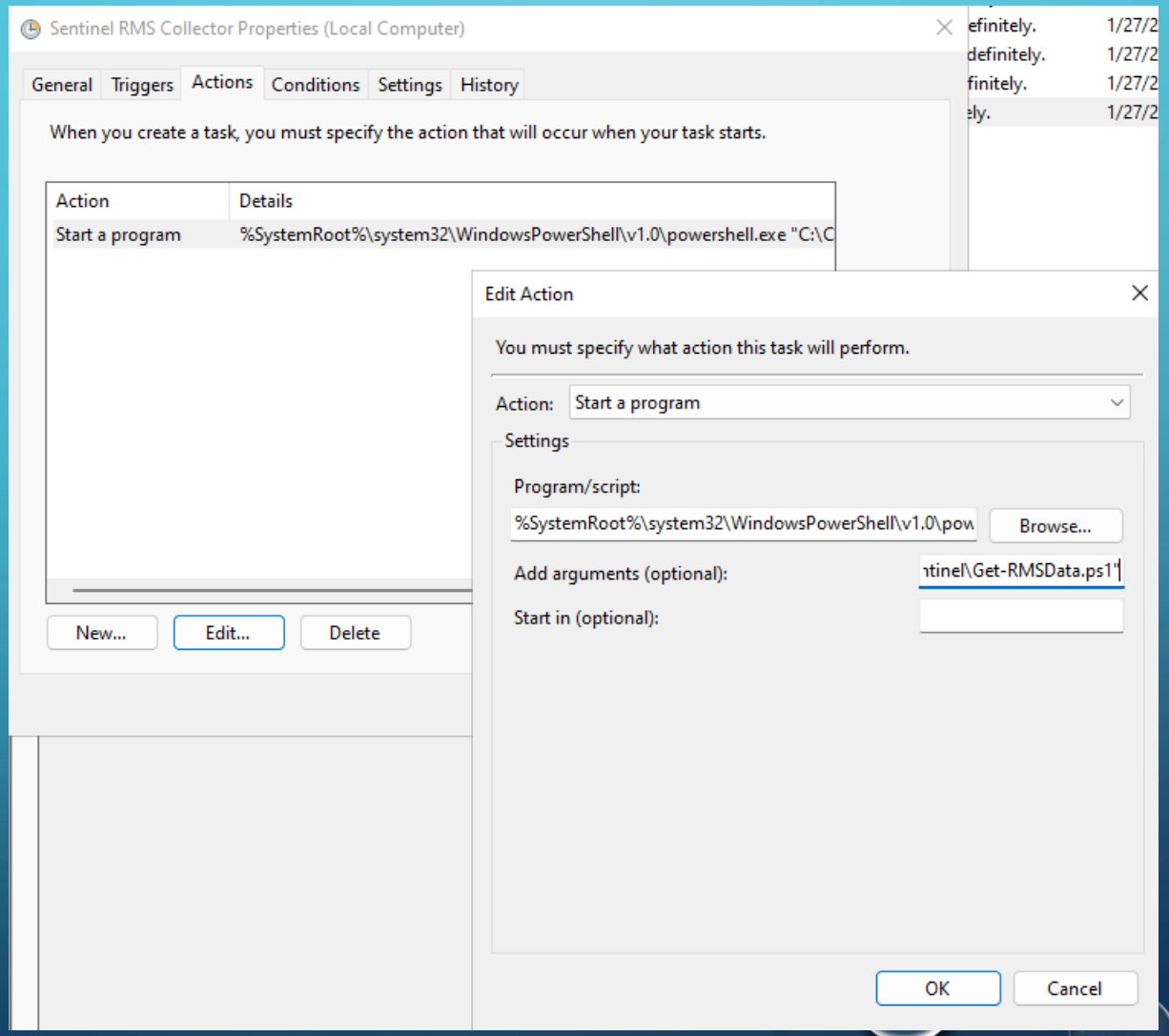
Steps



Microsoft

WORKSTATION OR SERVER

Steps (Get-RMSData.ps1 needs to be execute with PowerShell v5)

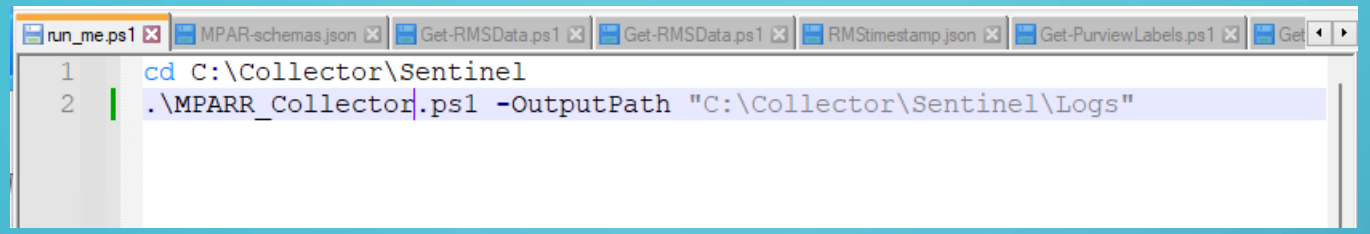


Microsoft

WORKSTATION OR SERVER

To execute the script with parameters

- Create a **run_me.ps1** file in this way, and then replace the script used under Task Scheduler



- The new MPARR_Collector.ps1 use the variable OutputPath inside of Iaconfig.json file, in this case run_me.ps1 can be still used to set other parameters.

- ADDITIONAL PARAMETERS ARE AVAILABLE AS A COMMENTS UNDER "EXPORT_LOGS.PS1", THAT OPTIONS CAN BE USED TO FILTER SOME **SPECIFIC OPERATIONS**, OR CHANGE LOG DIRECTORY, OR SET ONLY TO EXPORT TO LOCAL FILES. THESE FILTERS ARE:
  - FILTERAUDITAZUREACTIVEDIRECTORY
  - FILTERAUDITEXCHANGE
  - FILTERAUDITSHAREPOINT
  - FILTERAUDITGENERAL
  - FILTERDLPALL
- ALL THE FILTER AVAILABLE ARE EXPLAINED ON THE PRINCIPAL SCRIPT HEADER

Microsoft

WORKSTATION OR SERVER

To encrypt Azure AD app key, workspace key and Certificate Thumbprint do this:

- **Open laconfig.json file and do* this actions:**
  - **Under "EncryptedKeys" change "False" by "True"**

Copy the value for "ClientSecretValue" and execute on PowerShell this cmdlet:

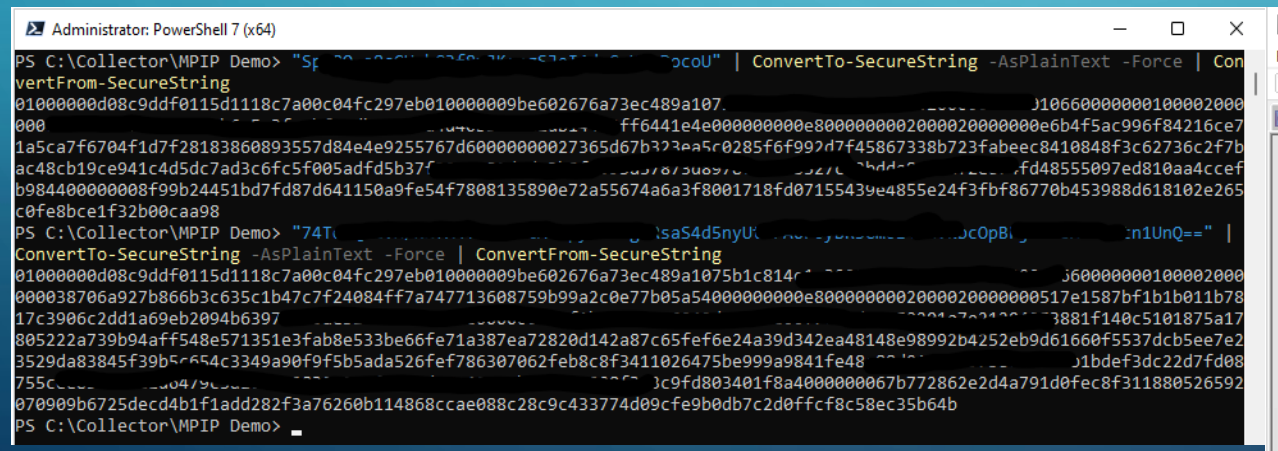
- `PS C:\> "ClientSecretValue**" | ConvertTo-SecureString -AsPlainText -Force | ConvertFrom-SecureString`
- **Replace the value on laconfig.json with the new string**

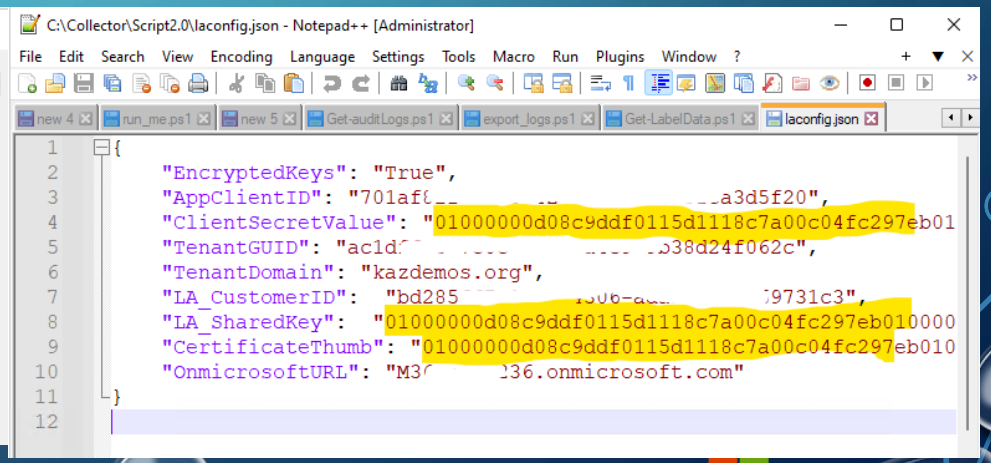
Copy the value for "LA_Sharedkey" and execute on PowerShell this cmdlet:

- `PS C:\> "LA_SharedKey**" | ConvertTo-SecureString -AsPlainText -Force | ConvertFrom-SecureString`
- **Replace the value on laconfig.json with the new string**

Copy the value for "CertificateThumb" and execute on PowerShell this cmdlet:

- `PS C:\> "CertificateThumb**" | ConvertTo-SecureString -AsPlainText -Force | ConvertFrom-SecureString`
- **Replace the value on laconfig.json with the new string**





**This action must be executed with the same account used to run the script*

***Use the key value*

Microsoft

SELECT TENANT TYPE

SELECT YOUR TENANT TYPE BETWEEN ENTERPRISE, GCC, GCCH OR DOD

- SELECT TENANT TYPE WAS ADDED TO **"LACONFIG.JSON"** FILE UNDER **"CLOUD"** ATTRIBUTE
- THE POSSIBLE VALUES ARE:
  - **COMMERCIAL** - COMMERCIAL CLOUD
  - **GCC** - GOVERNMENT COMMUNITY CLOUD
  - **GCCH** - GOVERNMENT COMMUNITY HIGH CLOUD
  - **DOD** - DEPARTMENT OF DEFENSE CLOUD

Microsoft

MS PURVIEW - POWERSHELL

Obtain your Label list with IDs and send to Logs Analytics

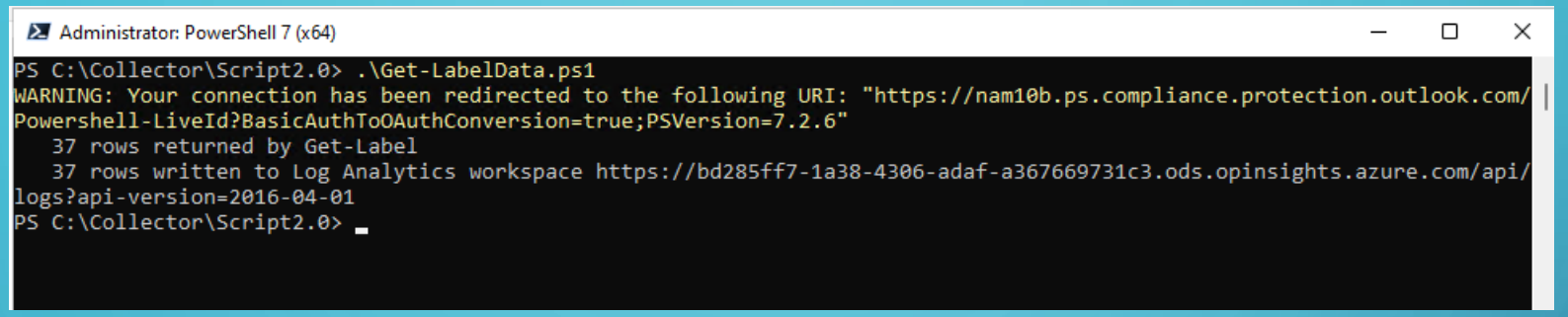
- Please complete first all this steps, **Office 365 Exchange Online API** is used to obtain the next information
- The data stored on API schemas contains only the ID of the labels, for that reason is required have a Matrix between IDs and Display Names, the script used here take the information from Information Protection Service and Import the data in Logs Analytics.
- To execute this script please complete these steps 1st.
- Copy the script Get-LabelData.ps1 to the same folder where the laconfig.json file is located, validate that you have the last version of this file containing Certificate Thumbprint and Onmicrosoft URL
- If you have an error to execute, validate that the certificate was imported in the machine used to execute this script
- How to consume through Power BI, click here.
- Understanding the complexity to give elevate privileges to an unattended script, a line can be modified on the script to a manual execution, in this case someone with Compliance Administrator role needs to execute this script on PowerShell, this execution is on-demand and is required only when you have modified or new labels, to up to date the reports with this information, or can be required depending the retention period in the Logs Analytics workspace.

**Depending on the Logs Analytics workspace retention this script will be required to execute continuously(monthly for a 30 days retention period) to don't lose that information*

Microsoft

WORKSTATION - POWERSHELL

STEPS




```
154
155 Function Export-LabelData() {
156     # ---------------------------------------------------------------
157     #    Name           : Export-LabelData
158     #    Desc           : Extracts data from Get-Label into Log analytics workspace tables for reporting purposes
159     #    Return         : None
160     # ---------------------------------------------------------------
161         <#
162         .NOTES
163         If you cannot add the "Compliance Administrator" role to the Azure AD App, for security reasons, you can comment the line 167 and uncomment the line 166, in that case
164         Someone with "Compliance Administrator" role needs to execute this script, this script is executed on-demand to refresh the label names
165         #>
166         #Connect-IPPSSession
167         Connect-IPPSSession -CertificateThumbPrint $CertificateThumb -AppID $AppClientID -Organization $OnmicrosoftTenant
```

SERVICE PLAN IDENTIFIERS FOR LICENSING

Create a Table in Logs Analytics with Product names and service plan identifiers for licensing

- When user licensing data is exported, a special names are used and doesn´t match with Known products, these steps helps to create a Matrix for that purpose.
- Download the latest updated product list from here and save the file in a known folder.
- Using the script ExportCSV2LA.ps1 you can import the data to Logs Analytics, to do that, execute the script in this way:

```
PS C:\Collector> .\ExportCSV2LA.ps1 -FileName ".\Product names and service plan identifiers for licensing.csv"-TableName "MSProducts"
```

- After executing the script can take between 5 to 10 minutes to display the New table under Logs Analytics
- How to consume through Power BI, click here.

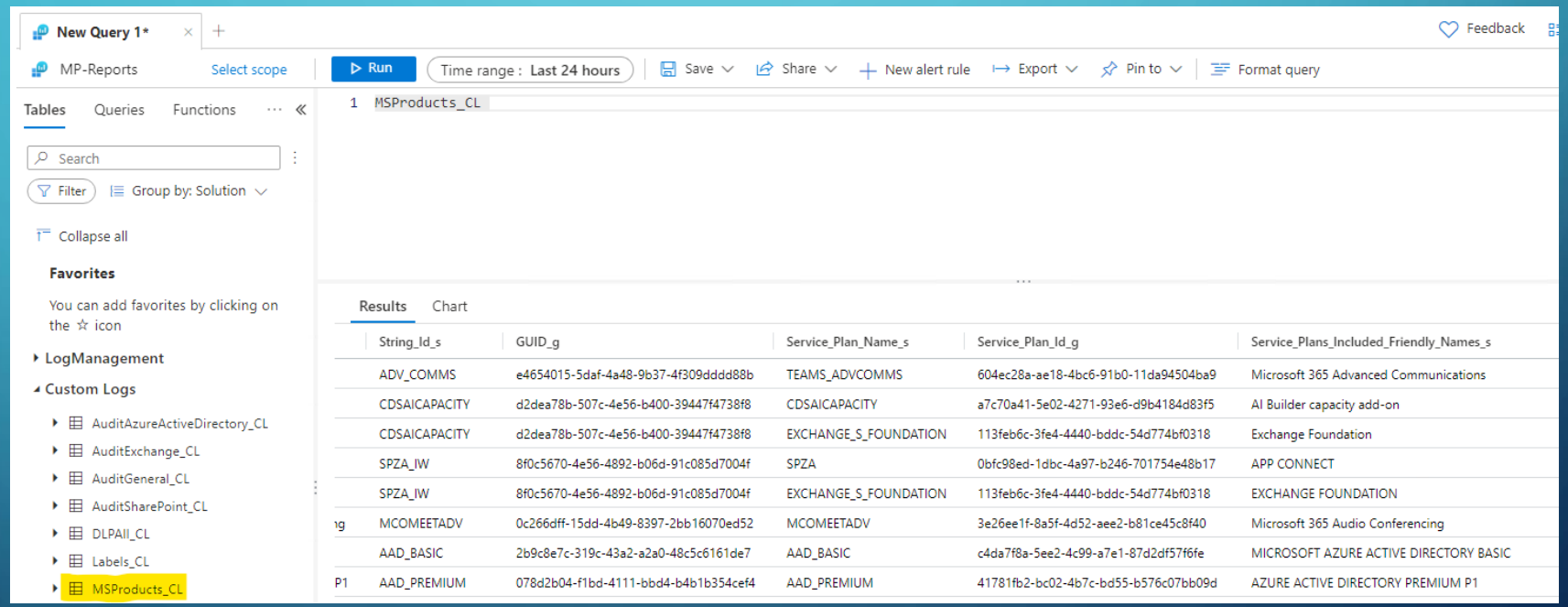
*https://learn.microsoft.com/en-us/azure/active-directory/enterprise-users/licensing-service-plan-reference
**Depending on the Logs Analytics workspace retention this script will be required to execute continuously(monthly for a 30 days retention period) to don't lose that information
*** Power BI templates use the table name called "MSProducts" any change to this name, need to make the change on Power BI queries

Microsoft

IMPORT DATA TO LOGS ANALYTICS

STEPS

```
Administrator: PowerShell 7 (x64)
PS C:\Collector\Script2.0> .\ExportCSV2LA.ps1 -FileName ".\Product names and service plan identifiers for licensing.csv"
-TableName "MSProducts"
Importing CSV file...
Calculating batch size...
Starting export to LA...
.
Export finished.
PS C:\Collector\Script2.0>
```



String_Id_s	GUID_g	Service_Plan_Name_s	Service_Plan_Id_g	Service_Plans_Included_Friendly_Names_s
ADV_COMMS	e4654015-5daf-4a48-9b37-4f309dddd88b	TEAMS_ADVCOMMS	604ec28a-ae18-4bc6-91b0-11da94504ba9	Microsoft 365 Advanced Communications
CDSAICAPACITY	d2dea78b-507c-4e56-b400-39447f4738f8	CDSAICAPACITY	a7c70a41-5e02-4271-93e6-d9b4184d83f5	AI Builder capacity add-on
CDSAICAPACITY	d2dea78b-507c-4e56-b400-39447f4738f8	EXCHANGE_S_FOUNDATION	113feb6c-3fe4-4440-bddc-54d774bf0318	Exchange Foundation
SPZA_IW	8f0c5670-4e56-4892-b06d-91c085d7004f	SPZA	0bfc98ed-1dbc-4a97-b246-701754e48b17	APP CONNECT
SPZA_IW	8f0c5670-4e56-4892-b06d-91c085d7004f	EXCHANGE_S_FOUNDATION	113feb6c-3fe4-4440-bddc-54d774bf0318	EXCHANGE FOUNDATION
MCOMEETADV	0c266dff-15dd-4b49-8397-2bb16070ed52	MCOMEETADV	3e26ee1f-8a5f-4d52-aee2-b81ce45c8f40	Microsoft 365 Audio Conferencing
AAD_BASIC	2b9c8e7c-319c-43a2-a2a0-48c5c6161de7	AAD_BASIC	c4da7f8a-5ee2-4c99-a7e1-87d2df57f6fe	MICROSOFT AZURE ACTIVE DIRECTORY BASIC
AAD_PREMIUM	078d2b04-f1bd-4111-bbd4-b4b1b354cef4	AAD_PREMIUM	41781fb2-bc02-4b7c-bd55-b576c07bb09d	AZURE ACTIVE DIRECTORY PREMIUM P1

LIST OF AZURE AD USERS WITH ATTRIBUTES AND LICENSING

- Please complete first all this steps, **Microsoft Graph API** is used to obtain the next information
- To generate a list of Azure AD users with some attributes like Country, City, Department, Job Title, or Office Location, you need execute the script **Get-AzureADData.ps1** previously check that you are using the latest **laconfig.json** file that contain the attribute **CertificateThumb**
- Execute the script on PowerShell 7 and that is.

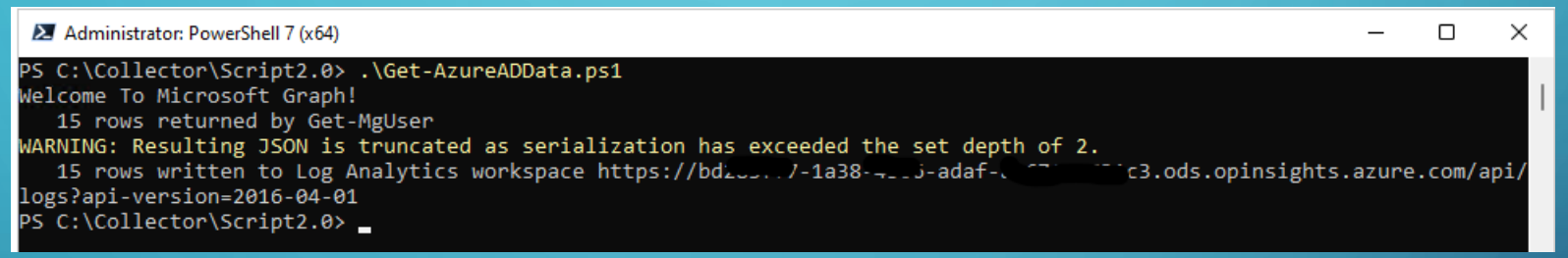
```
PS C:\Collector> .\Get-AzureADData.ps1
```

**Depending on the Logs Analytics workspace retention this script will be required to execute continuously(monthly for a 30 days retention period) to don't lose that information*

Microsoft

POWERSHELL - AZURE AD USERS

STEPS



LIST OF AZURE AD ROLES AND DOMAINS REGISTERED

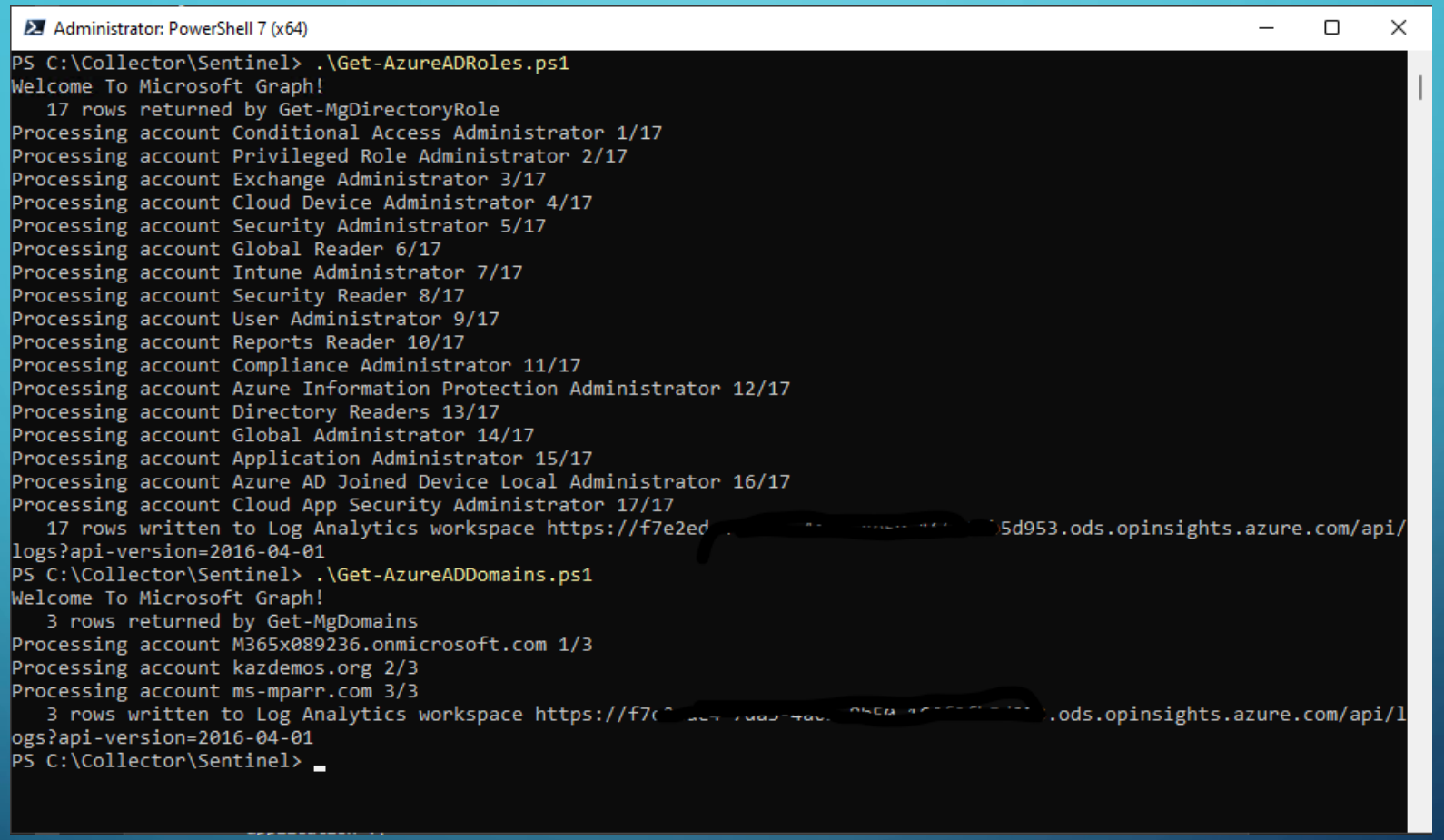
- Please complete first all this steps, **Microsoft Graph API** is used to obtain the next information
- To generate a list of Azure AD Roles, and Domains registered on the Tenant, you need execute the script **Get-AzureADRoles.ps1** and **Get-AzureADDomains.ps1,** previously check that you are using the latest **laconfig.json** file that contain the attribute **CertificateThumb**
- Execute the script on PowerShell 7 and that is.

```
PS C:\Collector> .\Get-AzureADDomains.ps1
PS C:\Collector> .\Get-AzureADRoles.ps1
```

**Depending on the Logs Analytics workspace retention this script will be required to execute continuously(monthly for a 30 days retention period) to don't lose that information*

POWERSHELL - AZURE AD USERS

STEPS



LOGS ANALYTICS WORKSPACE

Connect to logs stored in your workspace

- Open https://portal.azure.com
- Look for Log Analytics workspaces
  - Select your workspace pressing over the name
  - On the new blade under General category press Logs menu
  - Close the pop-up windows
  - Look for Custom Logs under Tables and press the arrow to show all the tables
  - If the script is running well and the Tenant have a normal use these 5 tables(based on the principal script) will be appear* under Custom Logs:
    - **AuditAzureActiveDirectory_CL**
    - **AuditExchange_CL**
    - **AuditGeneral_CL**
    - **AuditSharePoint_CL**
    - **DLPALL_CL**
  - Other tables are added from other scripts:
    - **AzureADUsers_CL**
    - **AzureADRoles_CL**
    - **AzureADDomains_CL**
    - **Labels_CL**
    - **MSProducts_CL**
    - **RMSData_CL**
    - **RMSDATADetails_CL**

**The first time that the script is running the tables can take between 5 to 15 minutes to appear under Custom Logs, if as an example no DLP rules are applied or Exchange is not used the tables will not appear until an operation is recorded.*

Microsoft

Export to use from Power BI

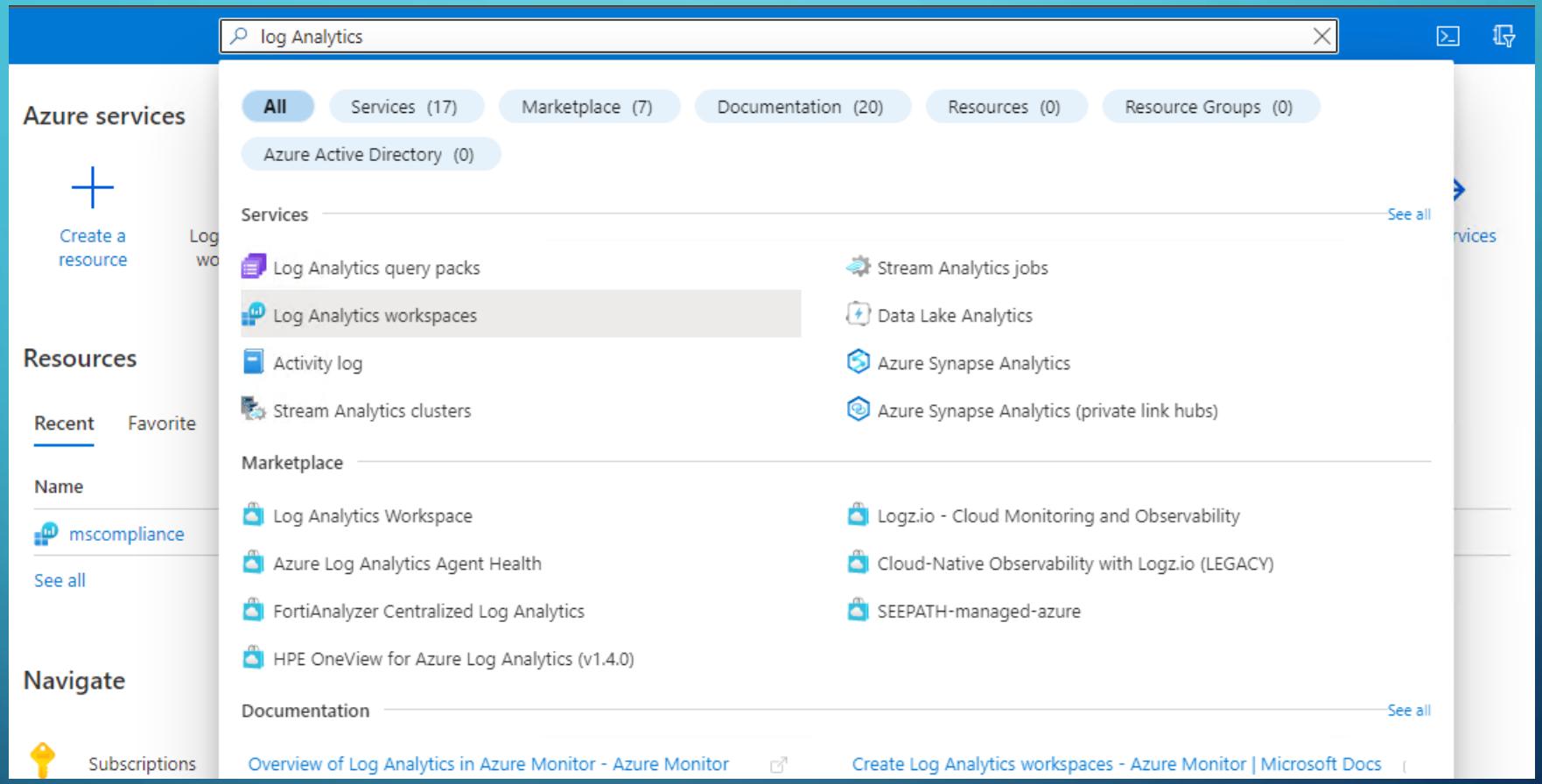
- The next steps are required to do for each table (5):
  - Double click in Table name, that action will be put the name of the table on query space
  - Press Run button (is not require make changes over Time range)
  - After pressing the query will be running and some results will be appeared at result area.
  - The Export option will be available, press and select Export to Power BI (M query)
  - The previous step will generate a txt file, save and rename the file in a know location using the table name as a filename.
  - Under the same query, below Table name add the next query:

```
| where TimeGenerated > now(-730d)
| summarize by
  Year = datetime_part('Year',TimeGenerated),
  Month = datetime_part('Month',TimeGenerated),
  Day = datetime_part('Day',TimeGenerated)
```

  - The previous will be used to resolve a download limit on Power BI, more information in this link
  - Press Run button and Export to Power BI (M query), same as previous step, and add the word “date” as a prefix for filename.
  - The results will be show years and months for the last 730 days of the data collected, in this case maybe only 1 year and one month.

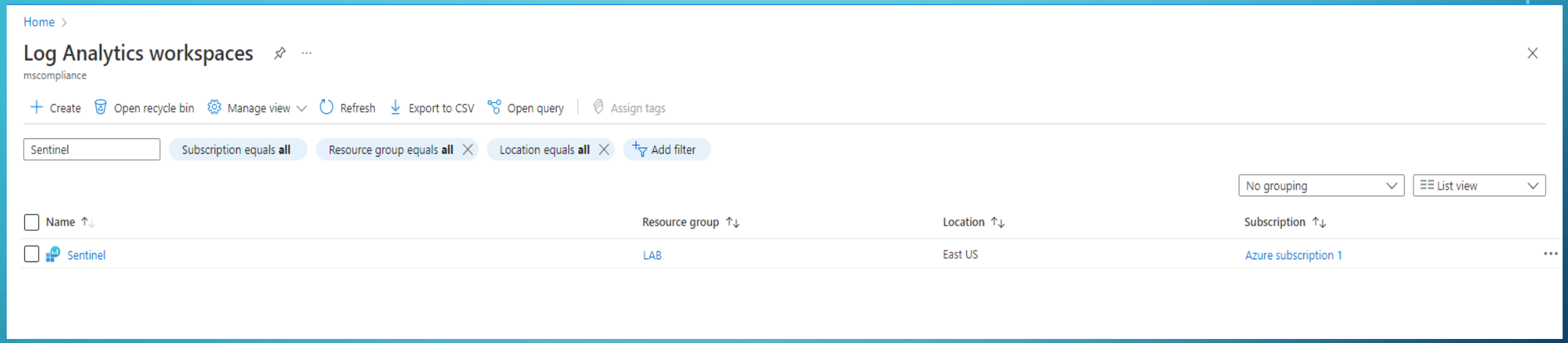
LOGS ANALYTICS WORKSPACE

Steps



LOGS ANALYTICS WORKSPACE

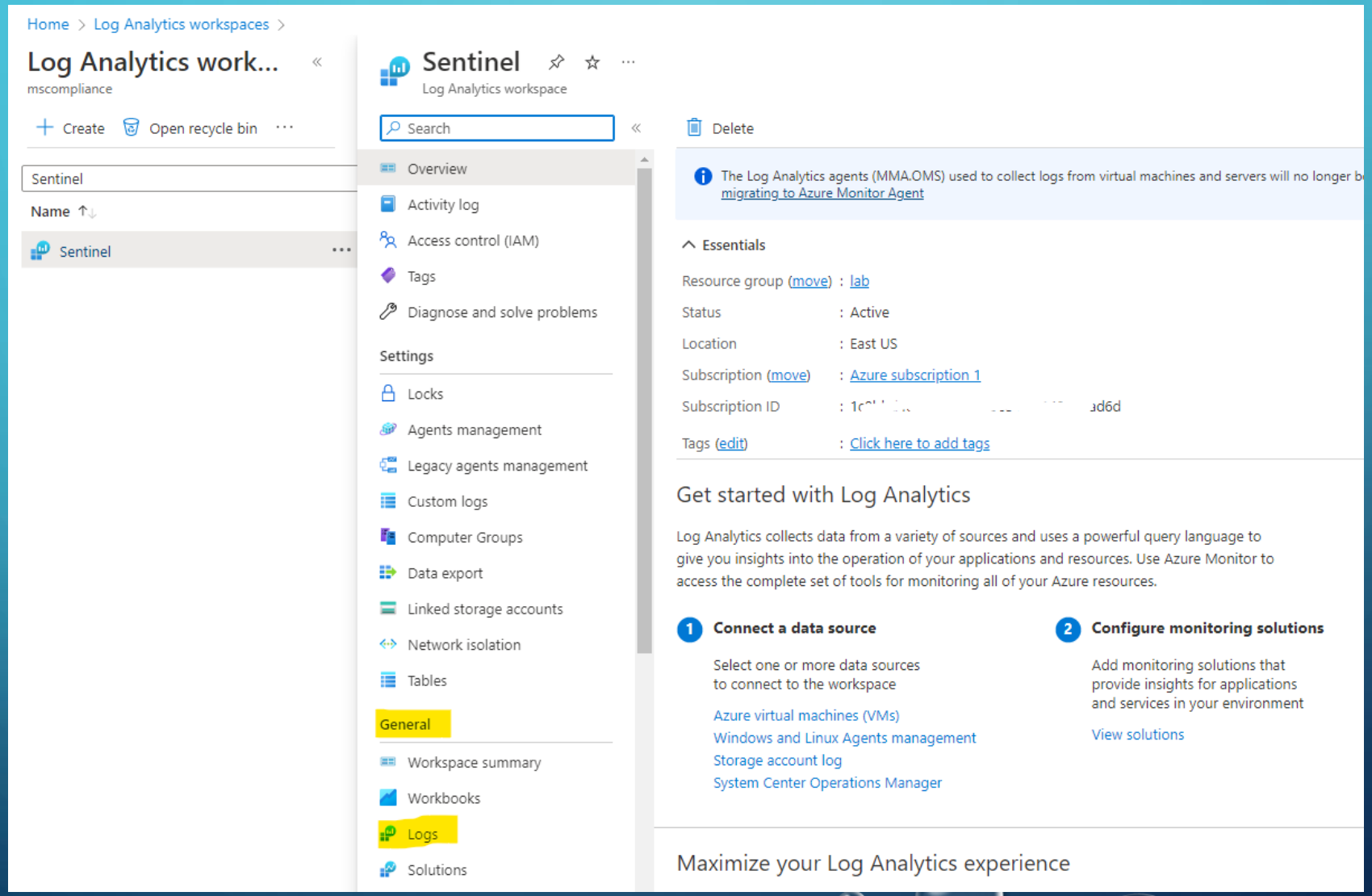
Steps



Microsoft

LOGS ANALYTICS WORKSPACE

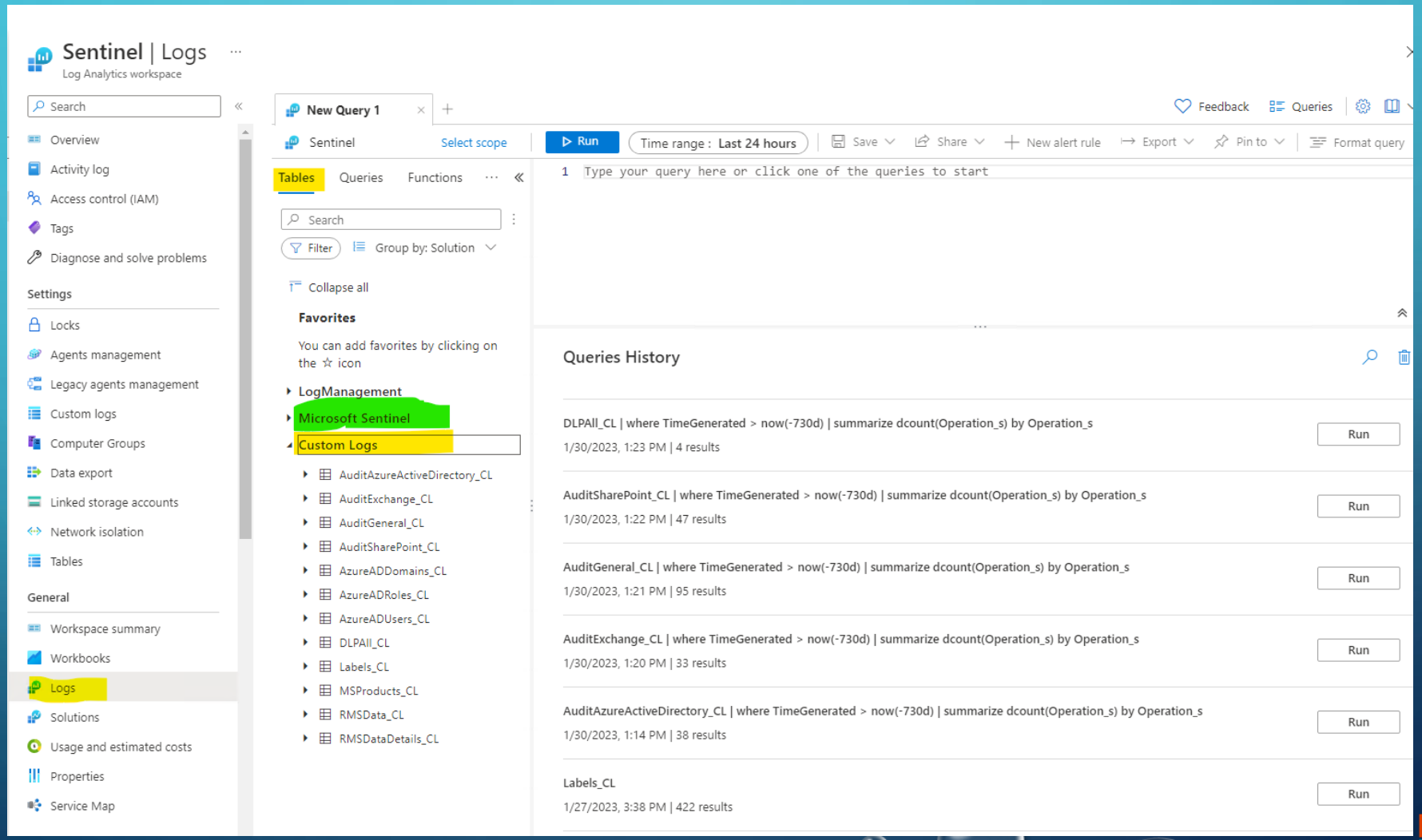
Steps



Microsoft

LOGS ANALYTICS WORKSPACE

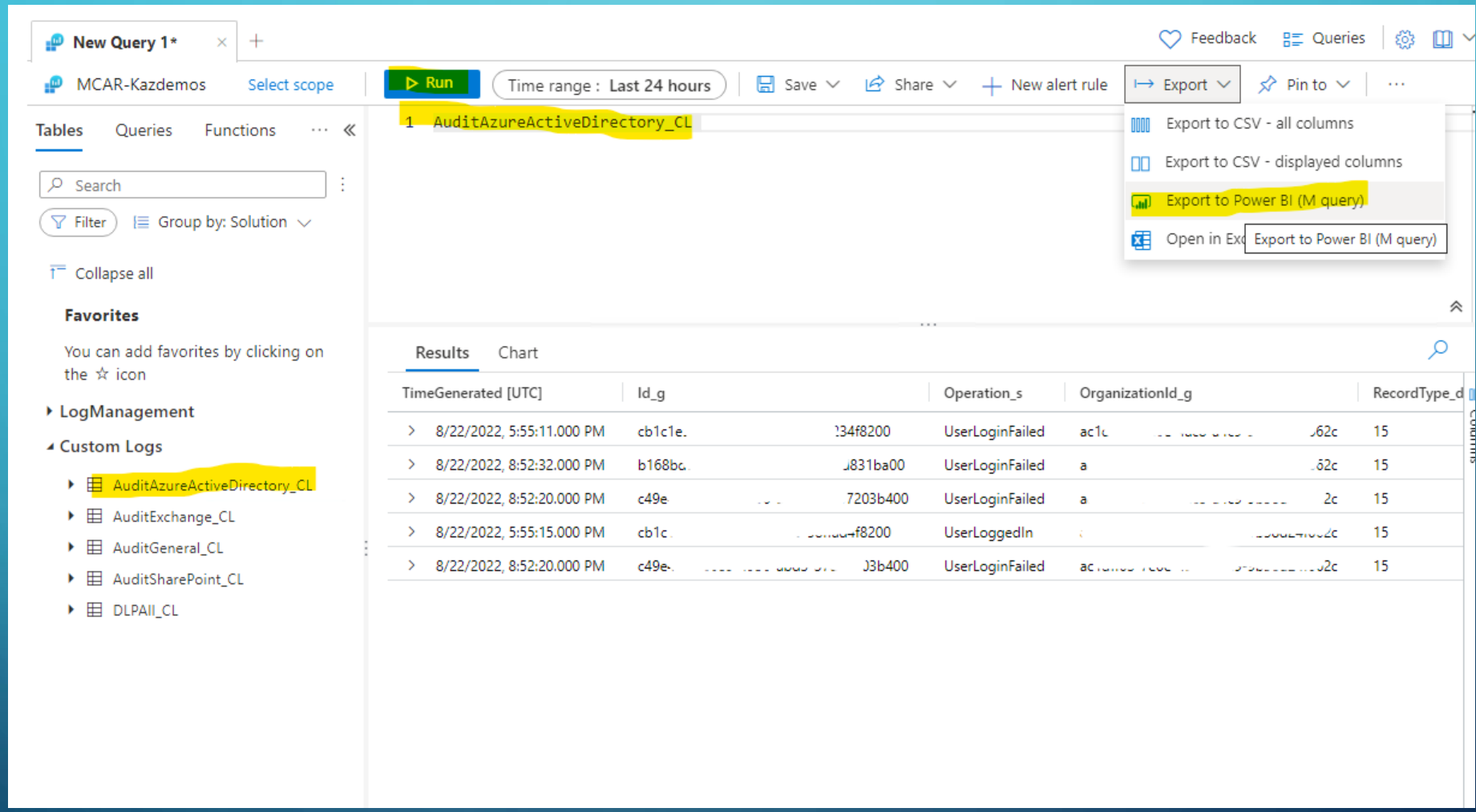
Steps



Microsoft

LOGS ANALYTICS WORKSPACE

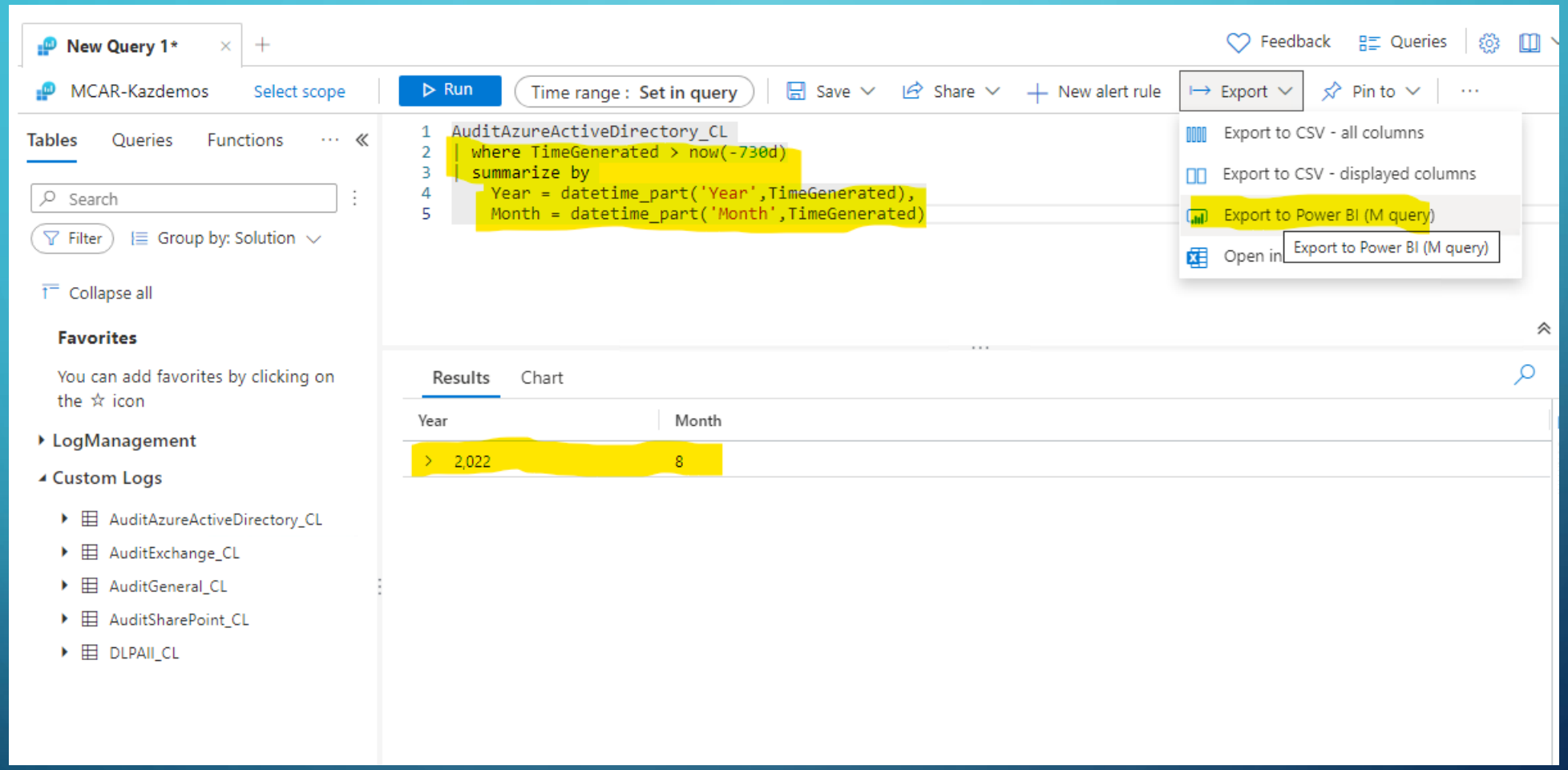
Steps



Microsoft

LOGS ANALYTICS WORKSPACE

Steps



LOGS ANALYTICS WORKSPACE

Export example

File Edit View

```
/*
The exported Power Query Formula Language (M Language ) can be used with Power Query in Excel
and Power BI Desktop.
For Power BI Desktop follow the instructions below:
1) Download Power BI Desktop from https://powerbi.microsoft.com/desktop/
2) In Power BI Desktop select: 'Get Data' -> 'Blank Query'->'Advanced Query Editor'
3) Paste the M Language script into the Advanced Query Editor and select 'Done'
*/


let AnalyticsQuery =
let Source = Json.Document(Web.Contents("https://api.loganalytics.io/v1/workspaces/5d642594-0883-4
[Query=[#"query"="AuditAzureActiveDirectory_CL
",#"x-ms-app"="OmsAnalyticsPBI",#"timespan"="P1D",#"prefer"="ai.response-thinning=true"],Timeout=#
TypeMap = #table(
{ "AnalyticsTypes", "Type" },
{
{ "string",   Text.Type },
{ "int",      Int32.Type },
{ "long",     Int64.Type },
{ "real",     Double.Type },
{ "timespan", Duration.Type },
{ "datetime", DateTimeZone.Type },
{ "bool",     Logical.Type },
{ "guid",     Text.Type },
{ "dynamic",  Text.Type }
}),
DataTable = Source[tables]{0},
Columns = Table.FromRecords(DataTable[columns]),
ColumnsWithType = Table.Join(Columns, {"type"}, TypeMap , {"AnalyticsTypes"}),
Rows = Table.FromRows(DataTable[rows], Columns[name]),
Table = Table.TransformColumnTypes(Rows, Table.ToList(ColumnsWithType, (c) => { c{0}, c{3}}))
in
Table
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Microsoft

KQL FOR LATEST LABEL INFORMATION

TO OBTAIN THE LATEST INFORMATION ABOUT LABELS

- Logs Analytics works appending data, for that reason every time that the script is executed to update the information, that data is added, in that order of ideas is important to download the most recently information, to do that execute this query:

```
Labels_CL
| where TimeGenerated > now(-730d)
| project
   DisplayName_s,
   Guid_g,
   Priority_d,
   ParentLabelDisplayName_s,
   TimeGenerated
|summarize max(TimeGenerated) by Guid_g, DisplayName_s, Priority_d, ParentLabelDisplayName_s
```

- Then export as Power BI(M query) and add to the rest of queries under Power BI Desktop

Microsoft

KQL FOR LATEST INFORMATION

TO OBTAIN THE LATEST INFORMATION ABOUT SERVICES PLAN AND PRODUCT NAMES

- Logs Analytics works appending data, for that reason every time that the script is executed to update the information, that data is added, in that order of ideas is important to download the most recently information, to do that execute this query:

```
MSProducts_CL
| where TimeGenerated > now(-730d)
| project
   Product_Display_Name_s,
   GUID_g,
   String_Id_s,
   TimeGenerated
|summarize max(TimeGenerated) by GUID_g, Product_Display_Name_s, String_Id_s
```

- Then export as Power BI(M query) and add to the rest of queries under Power BI Desktop

Microsoft

KQL FOR LATEST AZURE AD INFORMATION

TO OBTAIN THE LATEST INFORMATION ABOUT AZURE AD USERS AND LICENSING

Logs Analytics works appending data, for that reason every time that the script is executed to update the information, that data is added, in that order of ideas is important to download the most recently information, to do that execute this query:

```
AzureADInfo_CL
| where TimeGenerated > now(-730d) and UserPrincipalName_s != ""
| project
   UserPrincipalName_s,
   DisplayName_s,
   AssignedLicenses_s,
   Country_s,
   City_s,
   JobTitle_s,
   Department_s,
   Mail_s,
   OfficeLocation_s,
   TimeGenerated
|summarize max(TimeGenerated) by UserPrincipalName_s, AssignedLicenses_s, City_s, Country_s, Department_s, DisplayName_s, JobTitle_s, Mail_s, OfficeLocation_s
```

Then export as Power BI(M query) and add to the rest of queries under Power BI Desktop

Microsoft

POWER BI

Connect to logs stored in your workspace*

1. Open Power BI desktop
2. Close the initial welcome pop-up
3. Go to Get data and select Blank query
4. Under Power Query Editor window select Advanced Editor
5. Clear the example, open “your_table_name.txt” copy and paste all the information and press Done
6. Credentials can be request, open and select Organizational account, user with read permissions over Logs Analytics workspace are required
7. On the same window select New Source and press Blank Query again
8. Click on Advanced Editor and clear the sample
9. Open “date_your_table_name.txt” copy and paste all the information and press Done
10. At top right under PROPERTIES rename the query with your Table name

*Power BI templates are delivered with the script, these templates contains all the steps explained in this section, using the templates the next steps are not required

Microsoft

POWER BI

Power BI limit resolved

11. At left right click over first query and select Create Function
12. A pop-up appears, press Create
13. Assign a name, like as Data_YourTableName and press OK
14. Go to Advanced Editor and click it
15. A pop-up appears, press OK
16. Modify line 2 adding inside Source this additional data

    Source = (Month as text, Year as text) => let AnalyticsQuery =

17. Looking on the same editor the variable #"timespan"="P1D“, and replace P1D with this:

AuditAzureActiveDirectory_CL | where datetime_part('Month',TimeGenerated) == "& Month &" and datetime_part('Year',TimeGenerated) == "& Year &“

18. Take care on previous step to use the correct Table name and press Done
19. First query can be deleted, only the Date query and the function are required
20. Repeat from step 3 pressing New Source and Blank Query for each Table name, at the final you will have 5 queries and 5 functions.

Microsoft

POWER BI

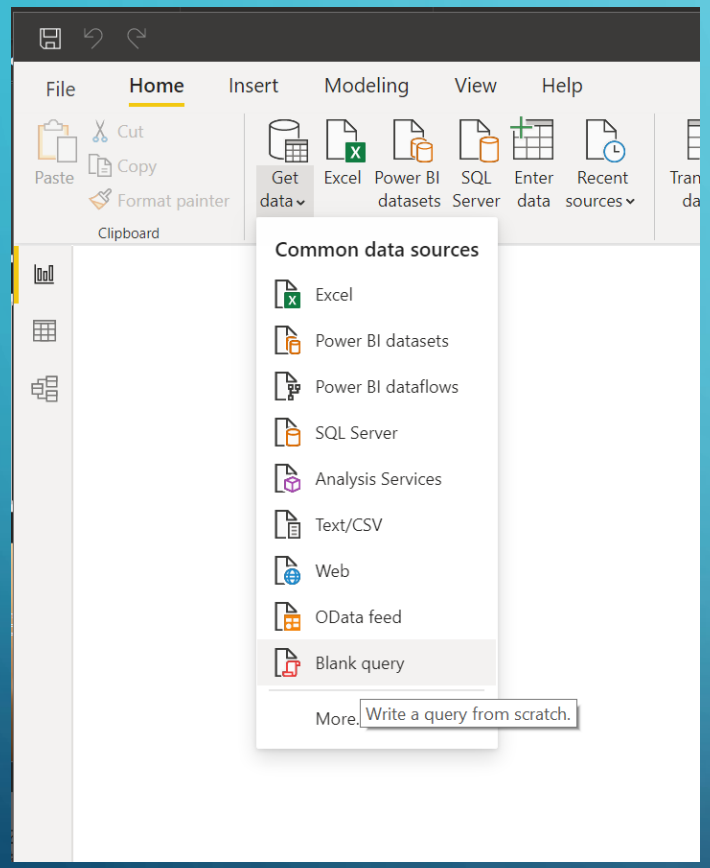
Power BI limit resolved

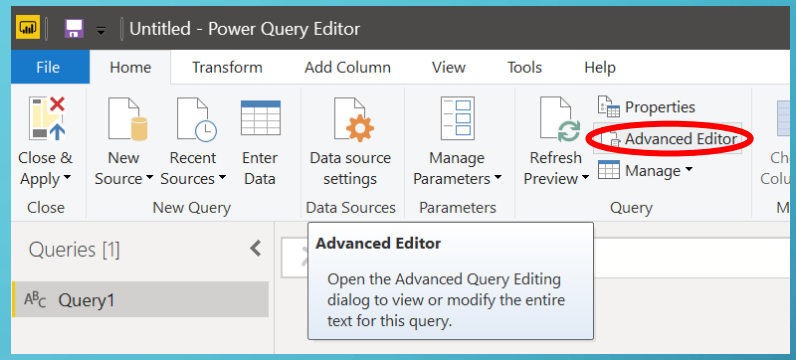
21. After finish the previous steps with the 5 tables is required merge the queries with the functions, to do that please go forward with the next steps taking care in each point
22. Select your query (the one with year and month as result), at the top select the tab Add Column and then press Invoke Custom Function
23. On the pop-up don't touch New column name under Function query select the function related to your Table name; after that 2 text areas will be appear fill Month with Month and Year with Year and press OK
24. A new column will be added with the capability to expand press the icon with 2 arrows one to right and the other to left located just right to columna name
25. In new window all columns related to the original tabl  ill be appear (sometimes appear at bottom an option to load more columns, in that case please press that option) uncheck Use original name as prefix
26. Now all the columns will be added based on the date, that step resolve the 500k limit on Power BI, if the data is too many a field Day can be added to the Date query.
27. Is important go to TimeGenerated column and change the Data Type to Date/Time/Timezone this step is relevant to apply filters based on time

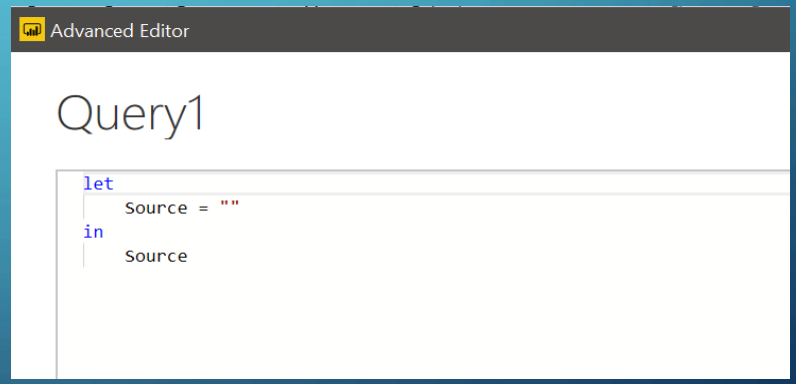
Microsoft

POWER BI

Steps







Microsoft

POWER BI

Steps

Advanced Editor

Query1

Display Options

```
/*
The exported Power Query Formula Language (M Language ) can be used with Power Query in Excel
and Power BI Desktop.
For Power BI Desktop follow the instructions below:
1) Download Power BI Desktop from https://powerbi.microsoft.com/desktop/
2) In Power BI Desktop select: 'Get Data' -> 'Blank Query'->'Advanced Query Editor'
3) Paste the M Language script into the Advanced Query Editor and select 'Done'
*/


let AnalyticsQuery =
let Source = Json.Document(Web.Contents("https://api.loganalytics.io/v1/workspaces/5d6425.                    .2d/query",
[Query=[#"query"="AuditAzureActiveDirectory_CL
",#"x-ms-app"="OmsAnalyticsPBI",#"timespan"="P1D",#"prefer"="ai.response-thinning=true"],Timeout=#duration(0,0,4,0)])),
TypeMap = #table(
{ "AnalyticsTypes", "Type" },
{
{ "string",   Text.Type },
{ "int",      Int32.Type },
{ "long",     Int64.Type },
{ "real",     Double.Type },
{ "timespan", Duration.Type },
{ "datetime", DateTimeZone.Type },
{ "bool",     Logical.Type },
{ "guid",     Text.Type },
{ "dynamic",  Text.Type }
}),
DataTable = Source[tables]{0},
Columns = Table.FromRecords(DataTable[columns]),
ColumnsWithType = Table.Join(Columns, {"type"}, TypeMap , {"AnalyticsTypes"}),
Rows = Table.FromRows(DataTable[rows], Columns[name]),
Table = Table.TransformColumnTypes(Rows, Table.ToList(ColumnsWithType, (c) => { c{0}, c{3}}))
in
Table
in AnalyticsQuery
```

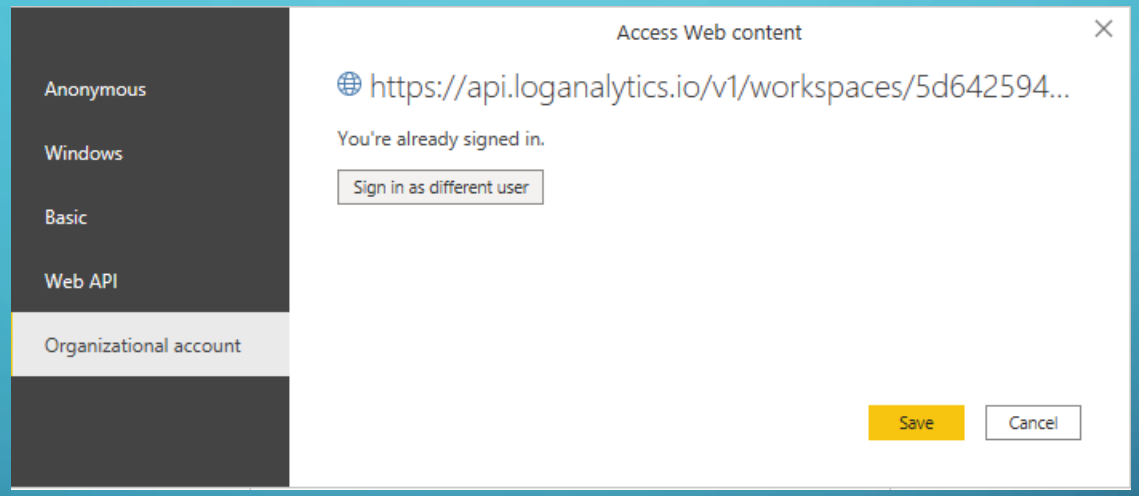
No syntax errors have been detected.

Done Cancel

Microsoft

POWER BI

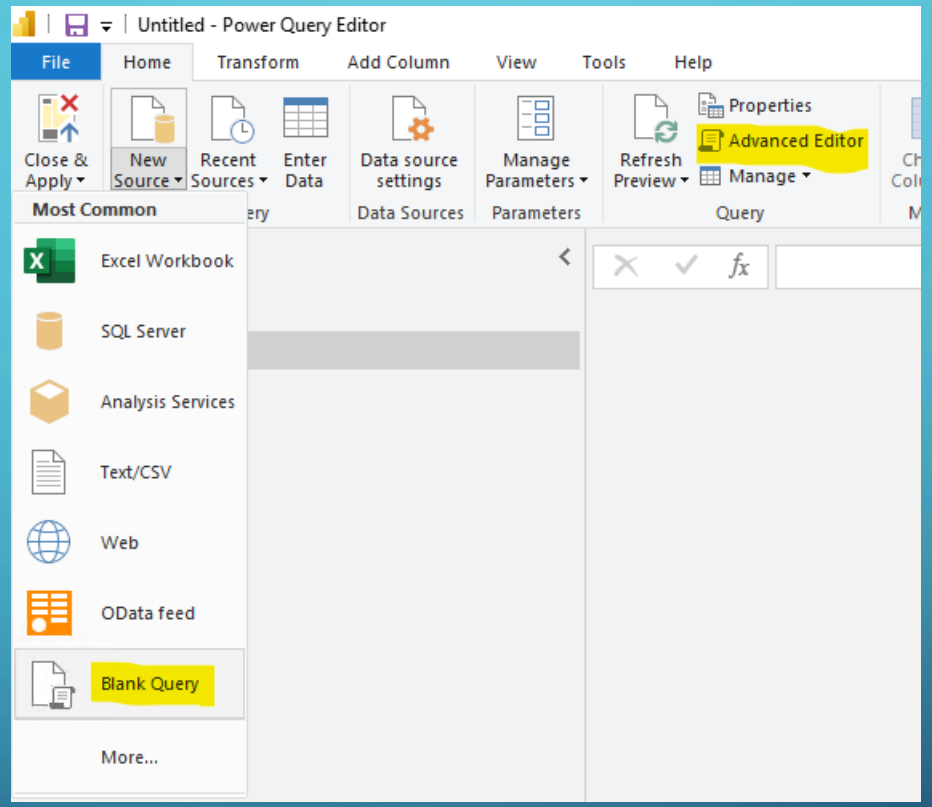
Steps



Microsoft

POWER BI

Steps



Microsoft

Steps

POWER BI

Advanced Editor

Query2

Display Options

```
/*
2) In Power BI Desktop select: 'Get Data' -> 'Blank Query'->'Advanced Query Editor'
3) Paste the M Language script into the Advanced Query Editor and select 'Done'
*/


let AnalyticsQuery =
let Source = Json.Document(Web.Contents("https://api.loganalytics.io/v1/workspaces/5d64                                    !d/query",
[Query=[#"query"="AuditAzureActiveDirectory_CL
| where TimeGenerated > now(-730d)
| summarize by
    Year = datetime_part('Year',TimeGenerated),
    Month = datetime_part('Month',TimeGenerated)
",#"x-ms-app"="OmsAnalyticsPBI",#"prefer"="ai.response-thinning=true"],Timeout=#duration(0,0,4,0)])),
TypeMap = #table(
{ "AnalyticsTypes", "Type" },
{
{ "string",   Text.Type },
{ "int",      Int32.Type },
{ "long",     Int64.Type },
{ "real",     Double.Type },
{ "timespan", Duration.Type },
{ "datetime", DateTimeZone.Type },
{ "bool",     Logical.Type },
{ "guid",     Text.Type },
{ "dynamic",  Text.Type }
}),
DataTable = Source[tables]{0},
Columns = Table.FromRecords(DataTable[columns]),
ColumnsWithType = Table.Join(Columns, {"type"}, TypeMap , {"AnalyticsTypes"}),
Rows = Table.FromRows(DataTable[rows], Columns[name]),
Table = Table.TransformColumnTypes(Rows, Table.ToList(ColumnsWithType, (c) => { c{0}, c{3}}))
in
Table
in AnalyticsQuery
```

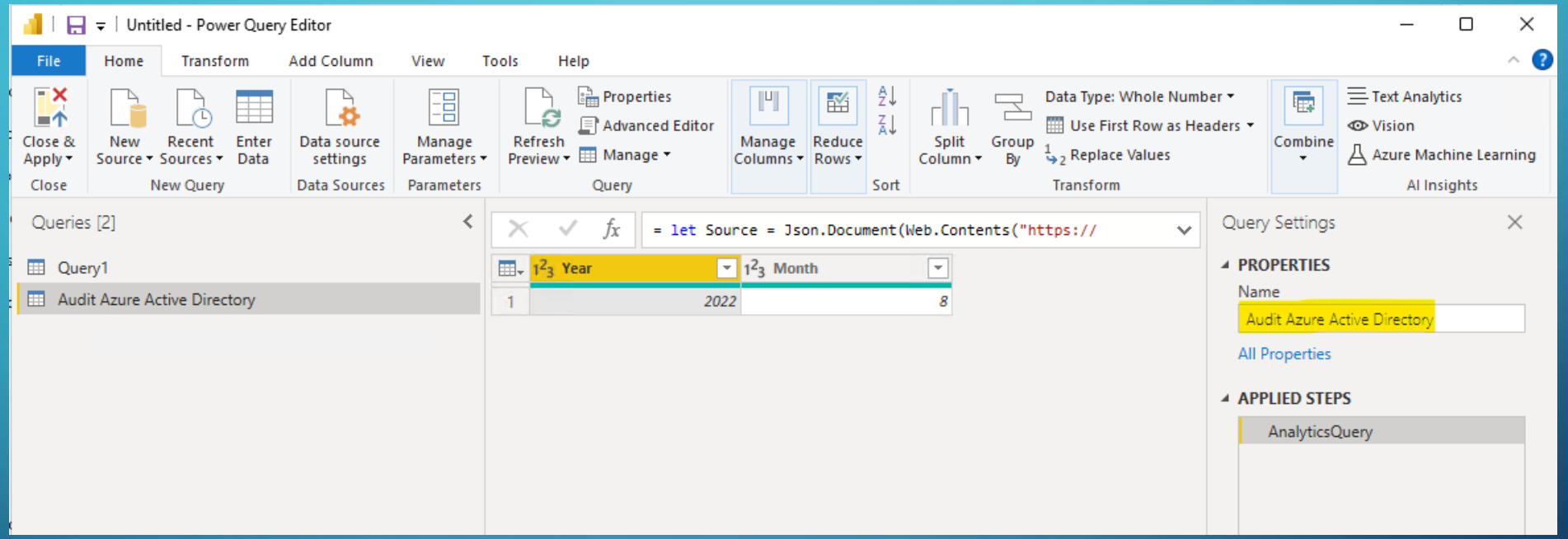
No syntax errors have been detected.

Done Cancel

Microsoft

POWER BI

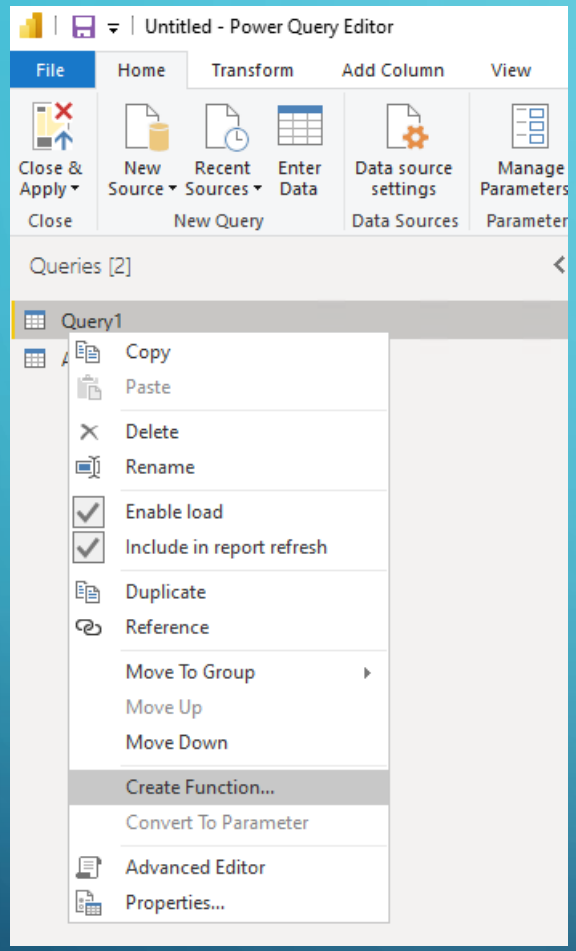
Steps

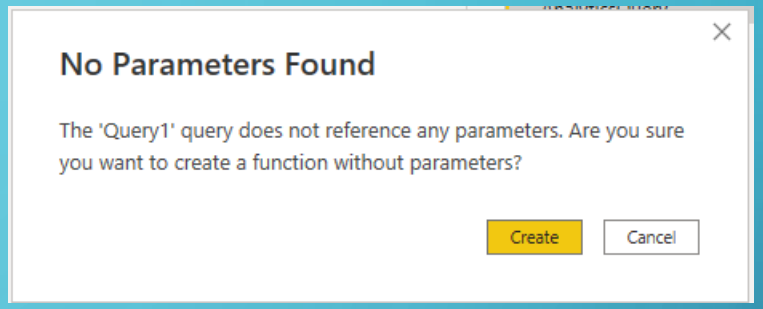


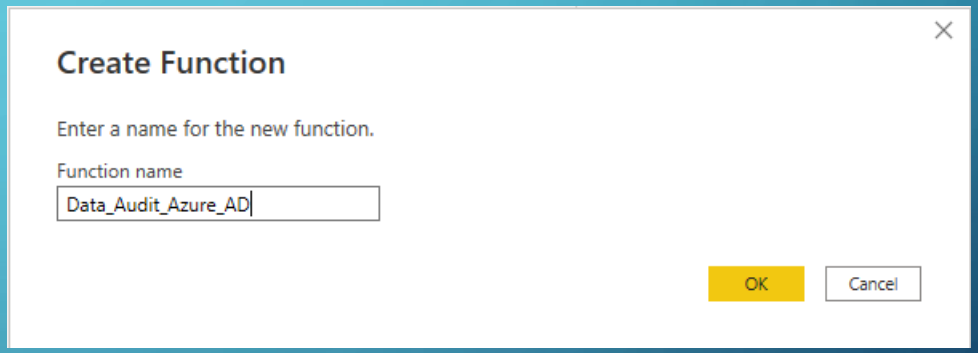
Microsoft

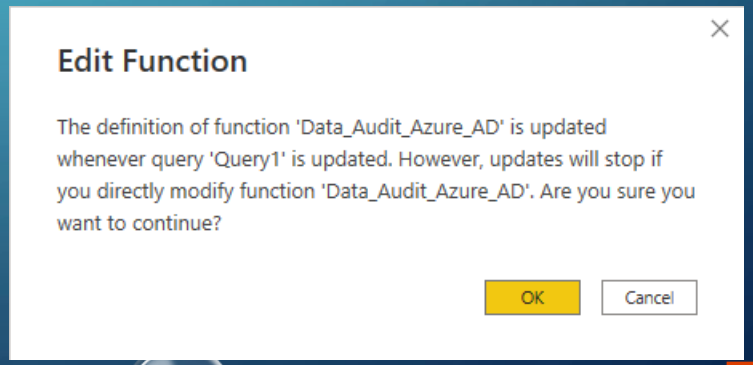
POWER BI

Steps









Microsoft

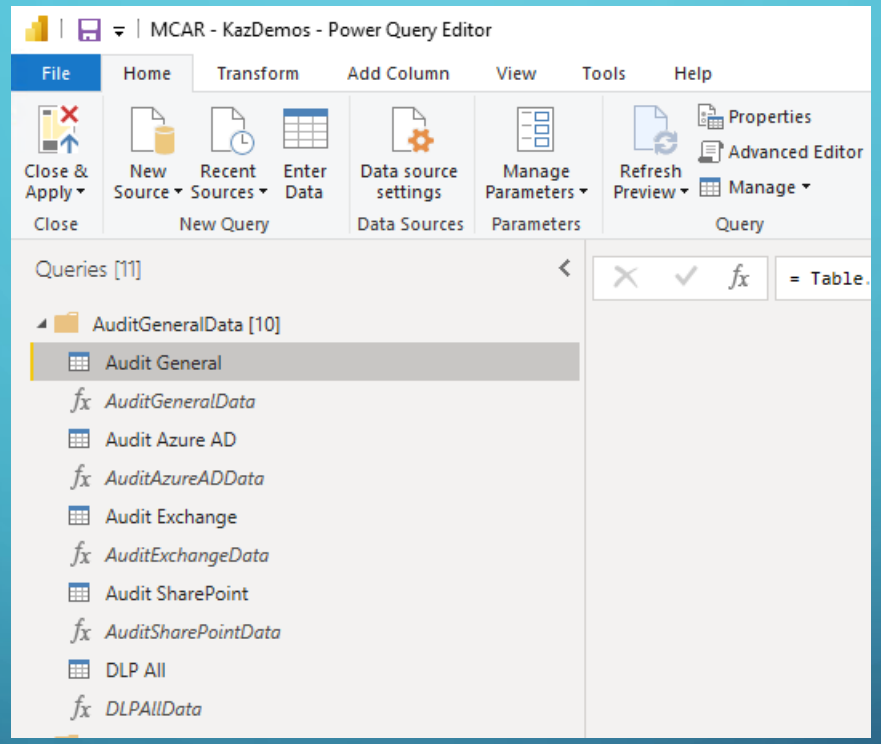
POWER BI

Steps

Microsoft

POWER BI

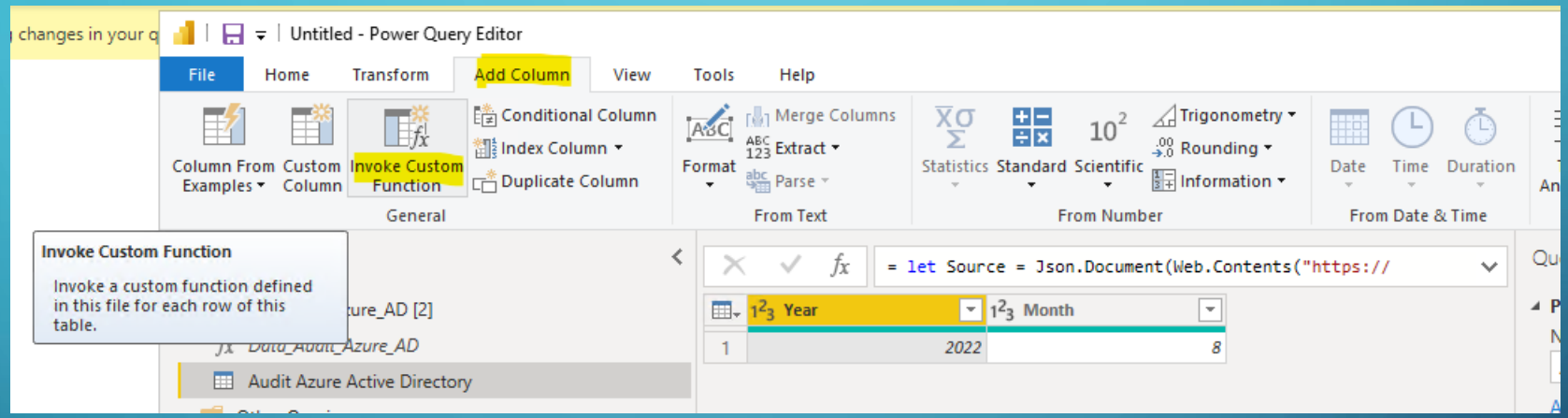
Steps



Microsoft

POWER BI

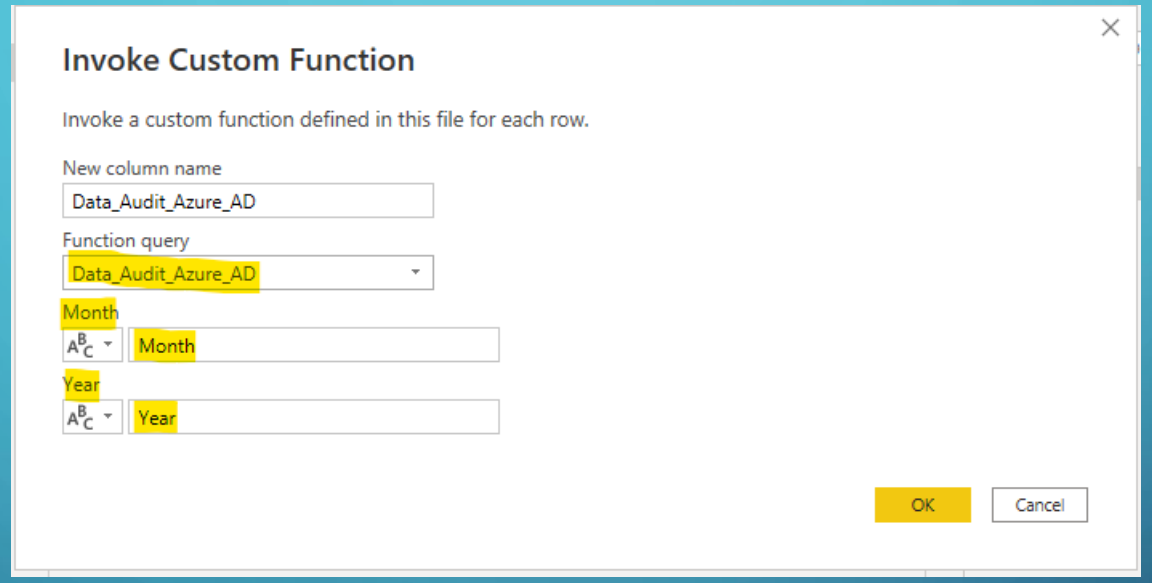
Steps



Microsoft

POWER BI

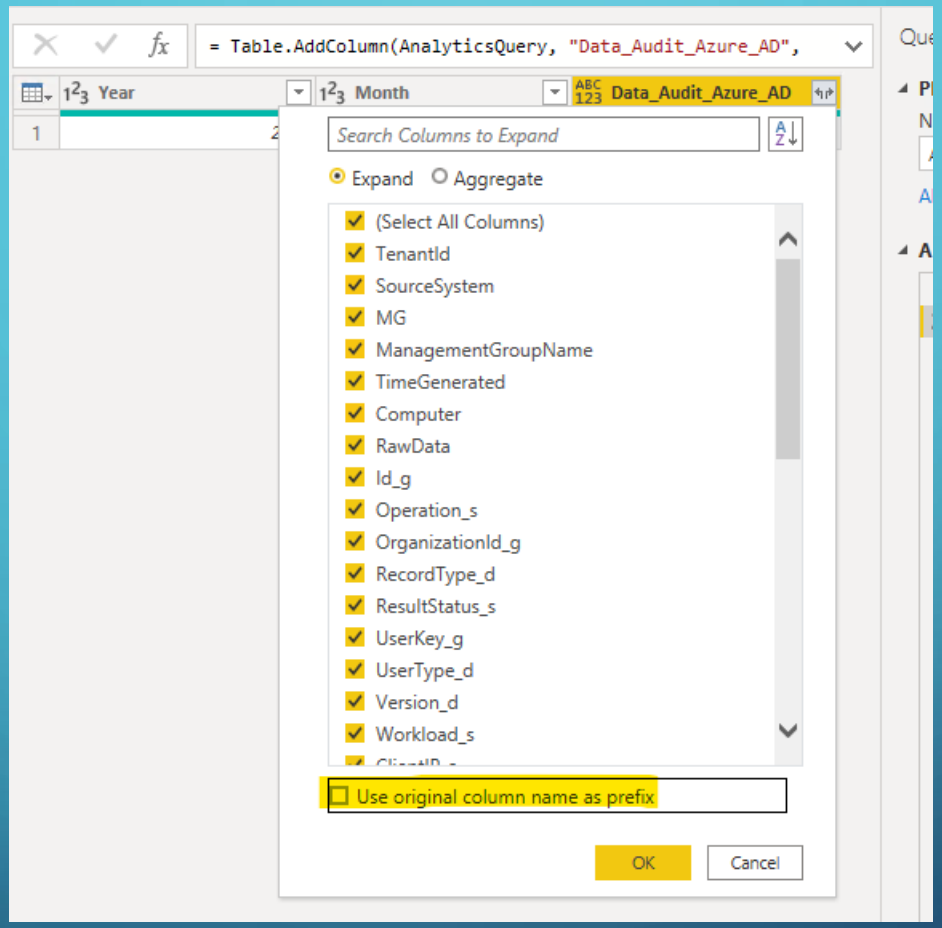
Steps



Microsoft

POWER BI

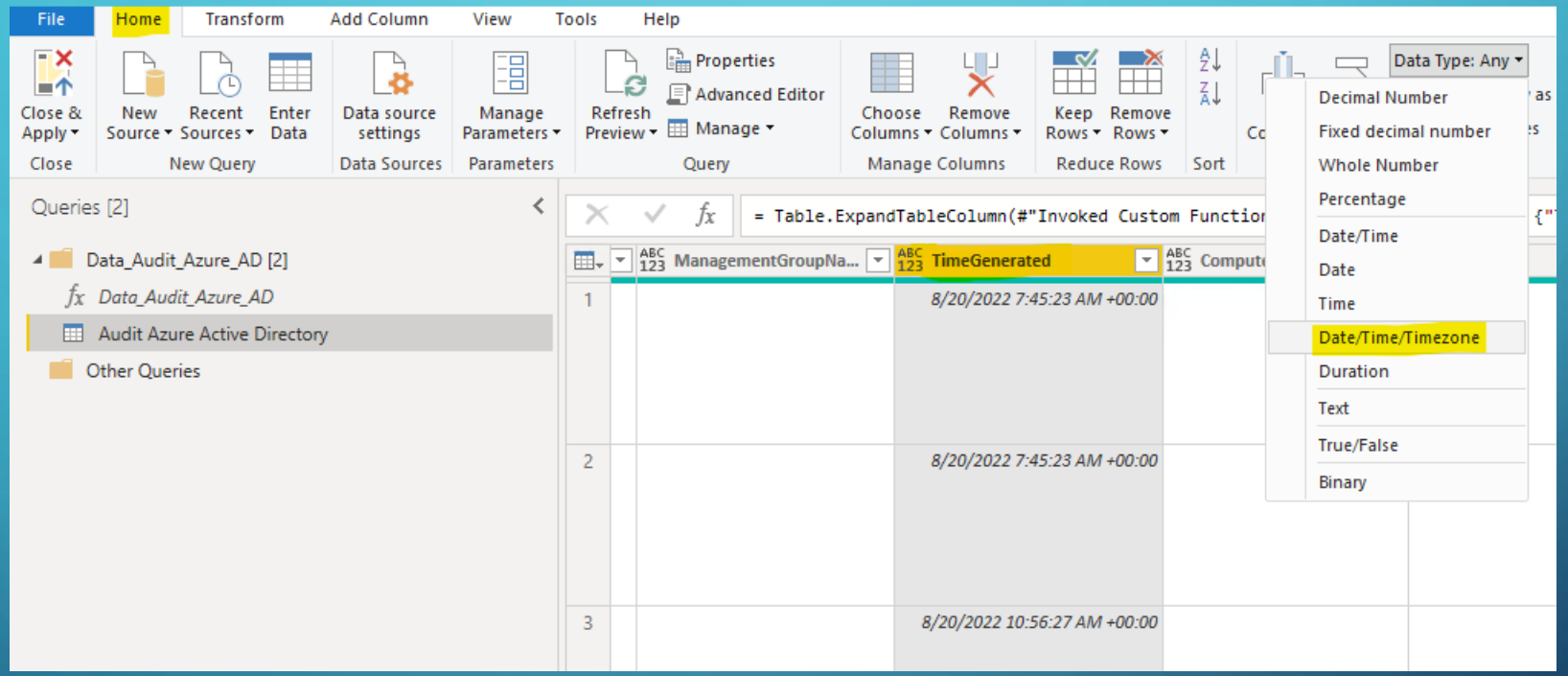
Steps



Microsoft

POWER BI

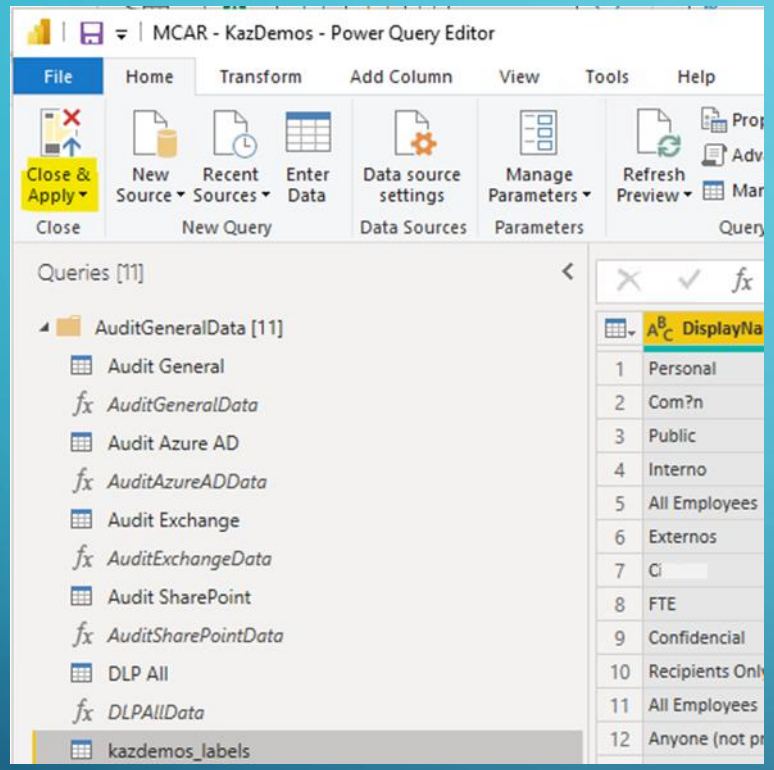
Steps

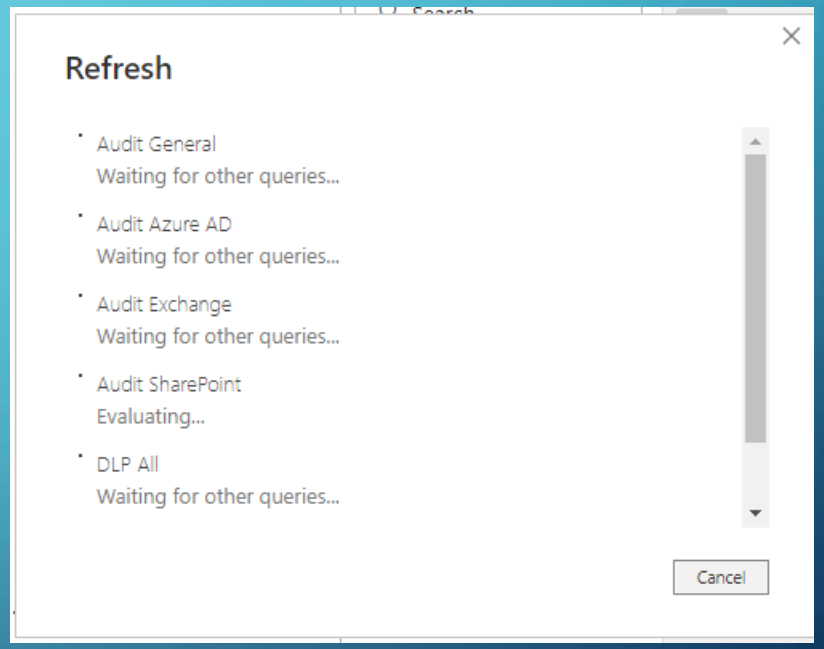


Microsoft

POWER BI

Steps

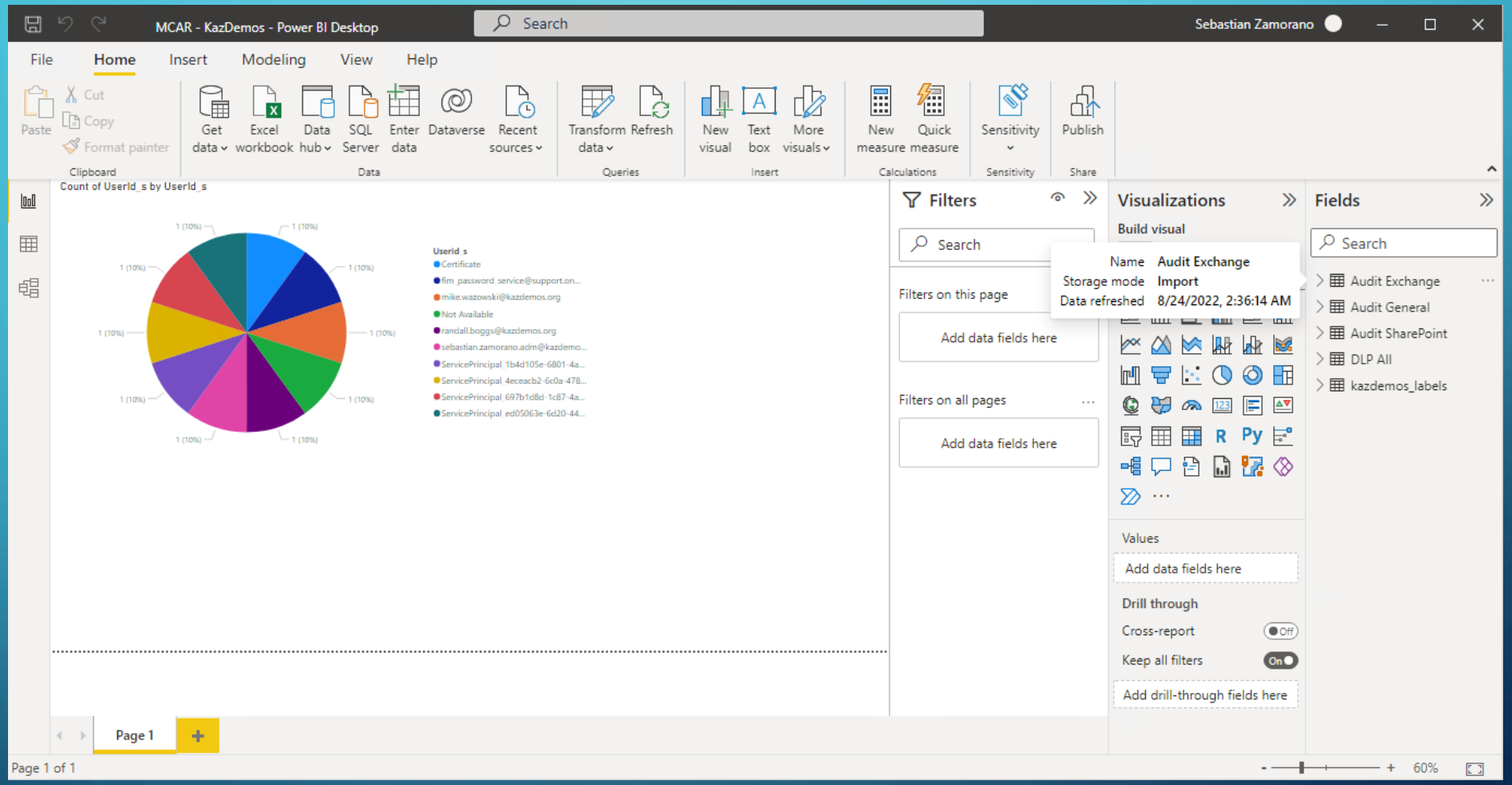




Microsoft

POWER BI

Steps



Microsoft

POWER BI - TEMPLATES

To simplify, we can create templates

- To easily export our Power BI effort, do the next steps:

1. Right click at left over any of the table names and select Edit query
2. Under Power Query Editor go to Manage Parameters and select New Parameter
3. On the new pop-up window Add these:
    - Name: **Workspace_ID**
    - Description: Workspace ID from Logs Analytics
    - Check Required
    - Type: Text
    - Suggested Values: Any value
    - Current Value: {*add your workspace id from Logs Analytics*}
4. After that, in each table(5) and function(5) is required change the Workspace ID with the new variable go through Advanced Editor

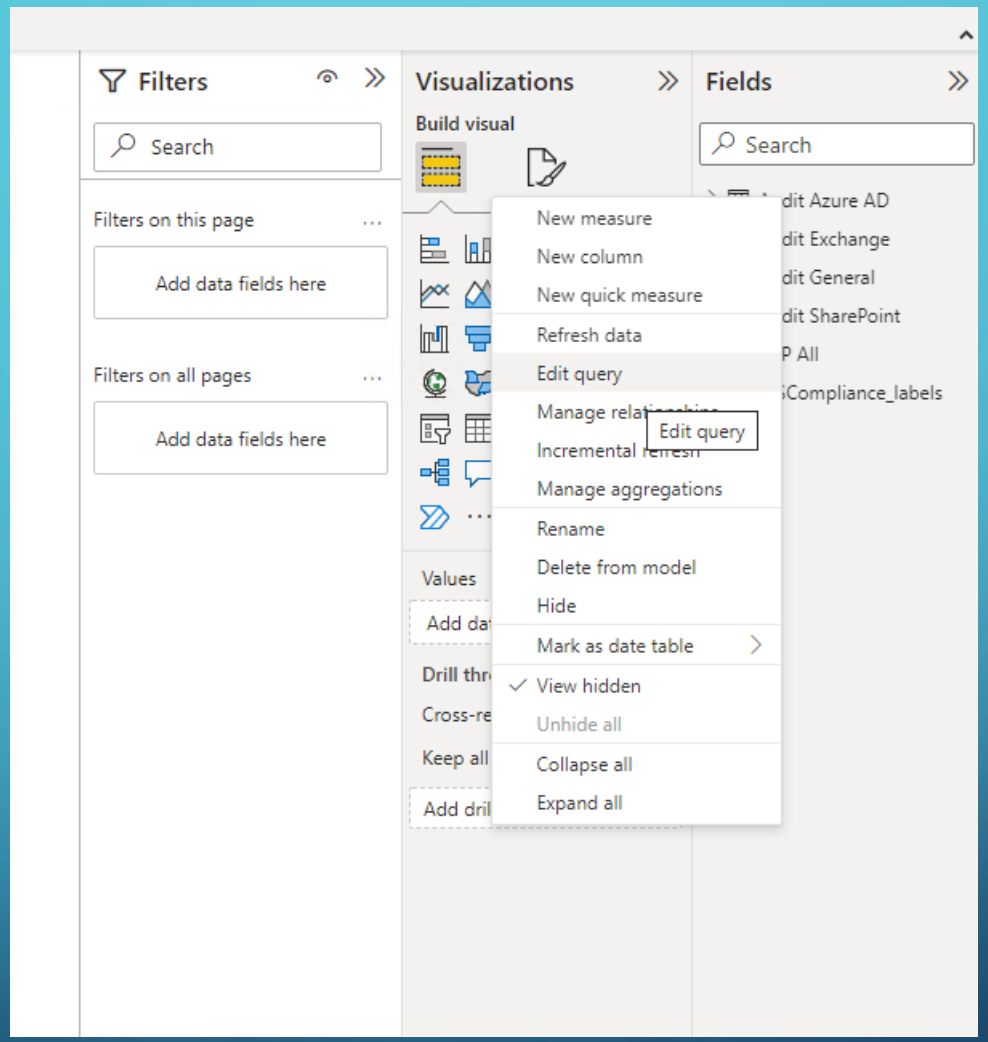
let Source = Json.Document(Web.Contents("https://api.loganalytics.io/v1/workspaces/"& Workspace_ID &"/query",

5. After finish, press Close & Apply
6. To export go to File and select Export and select Power BI template
7. Add a Template description
8. Set a File name and Location and Save

Microsoft

POWER BI - TEMPLATES

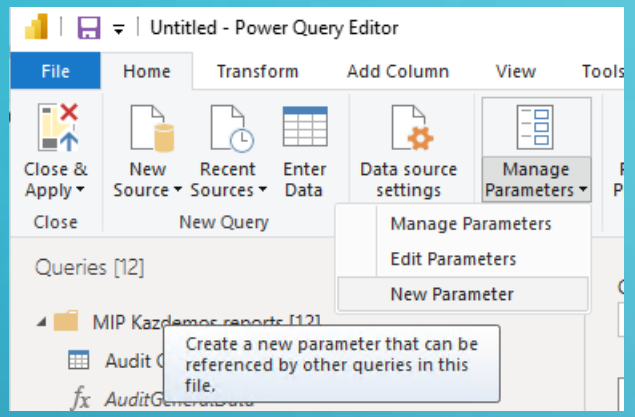
Steps

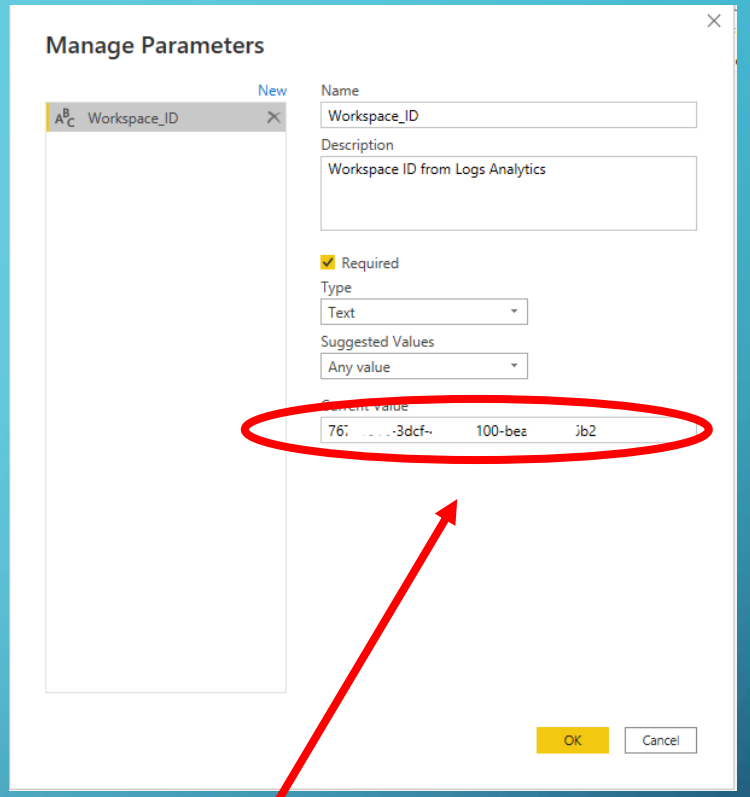


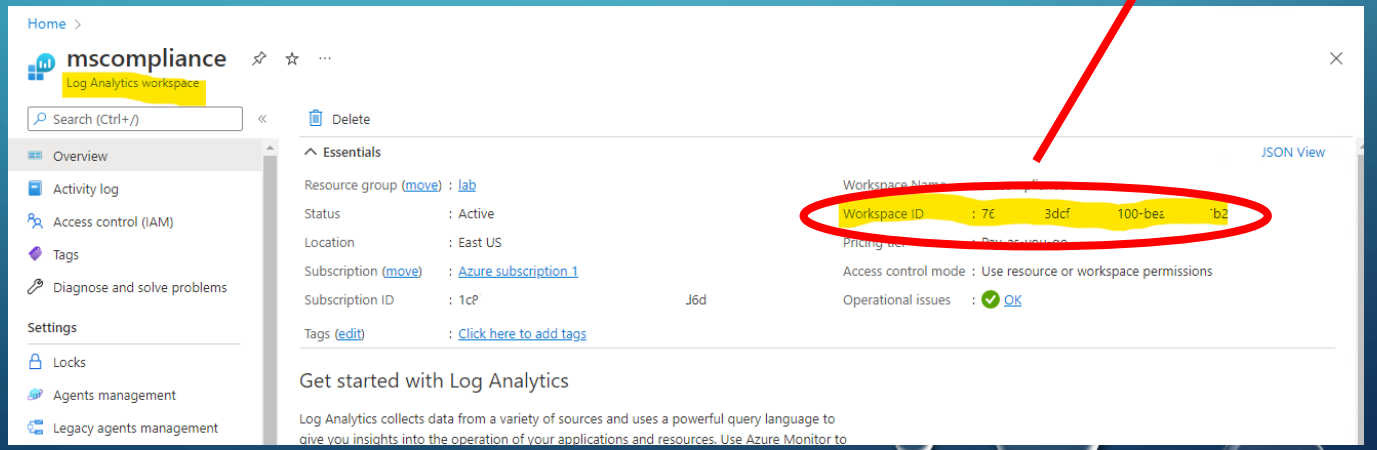
Microsoft

POWER BI - TEMPLATES

Steps



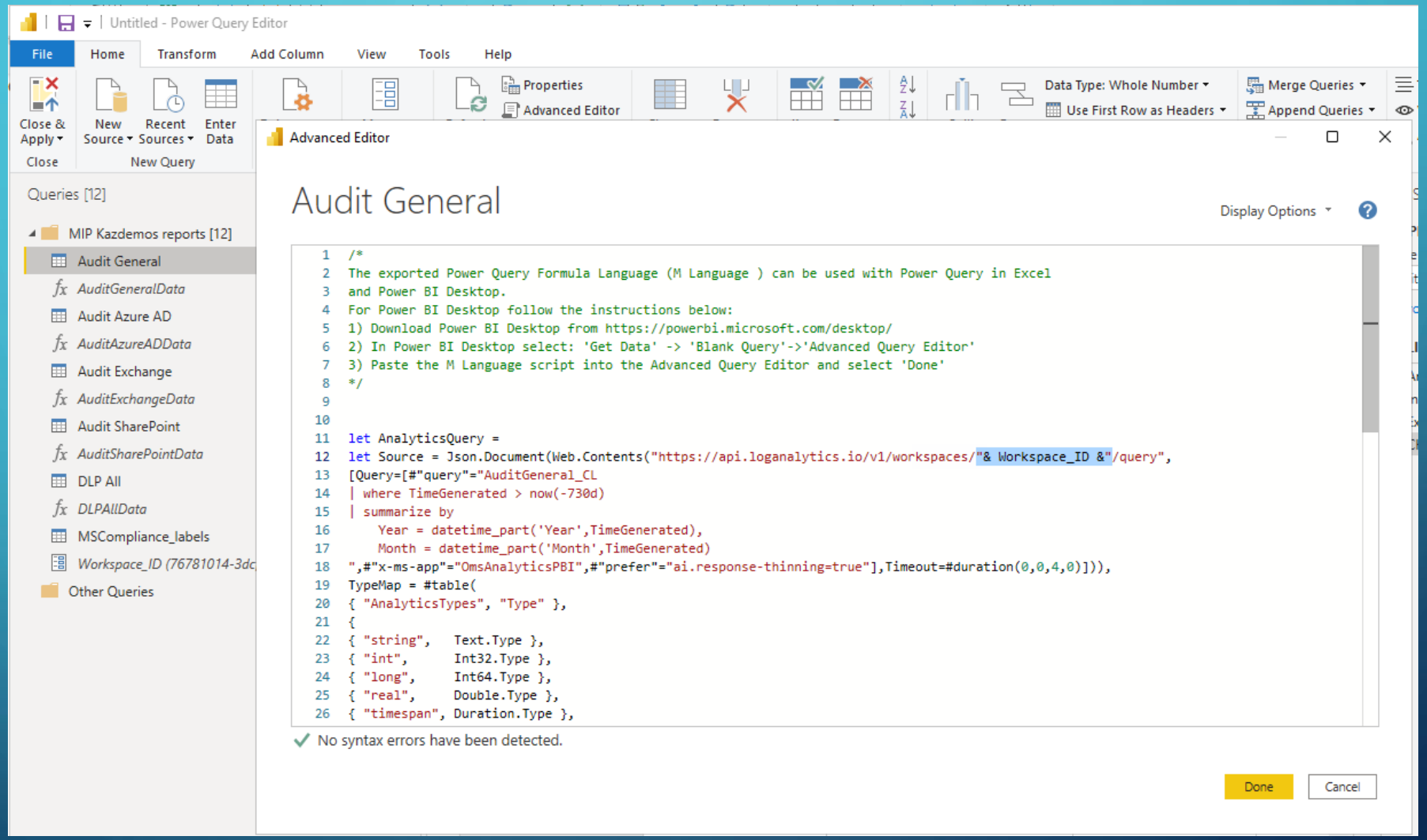




Microsoft

POWER BI - TEMPLATES

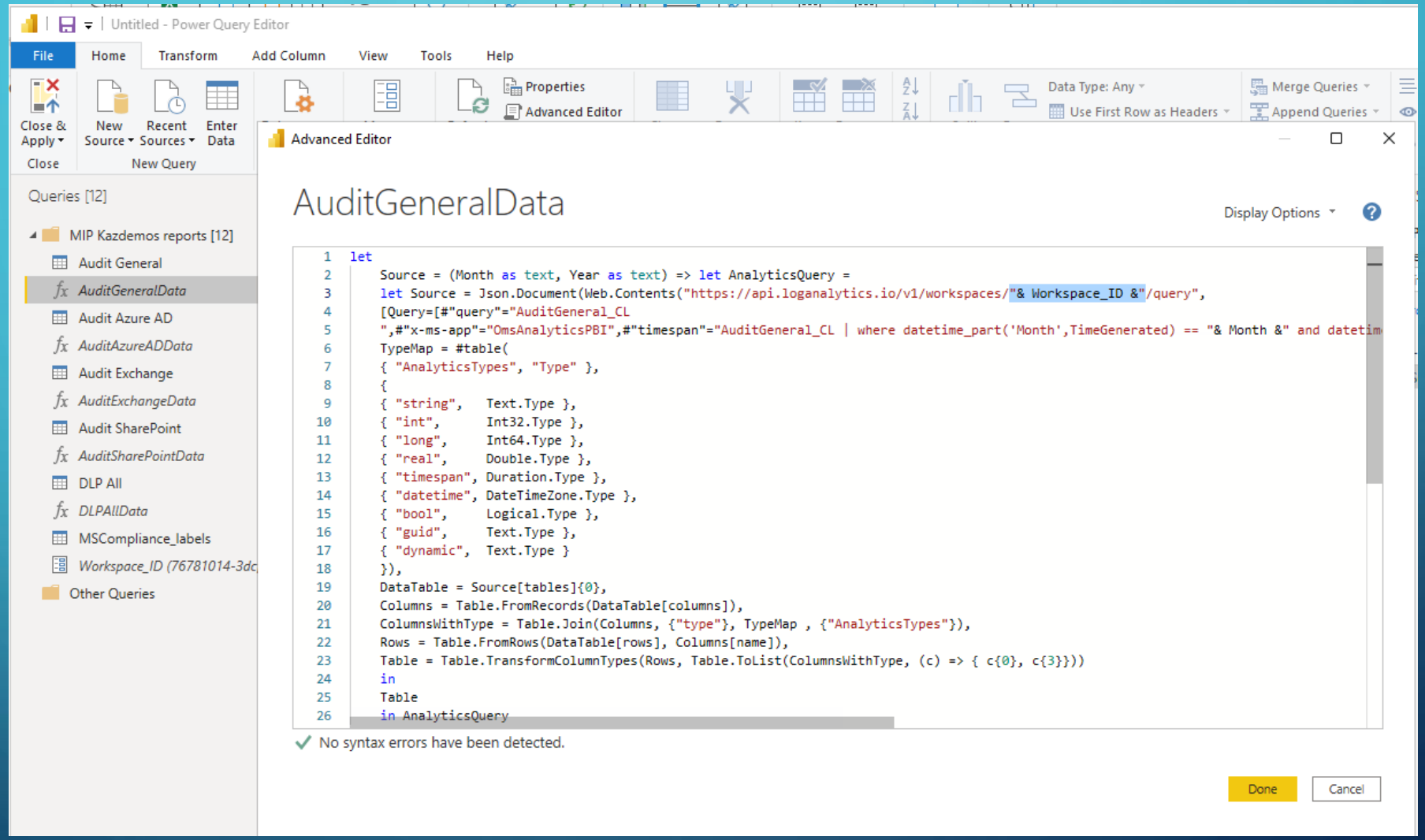
Steps



Microsoft

POWER BI - TEMPLATES

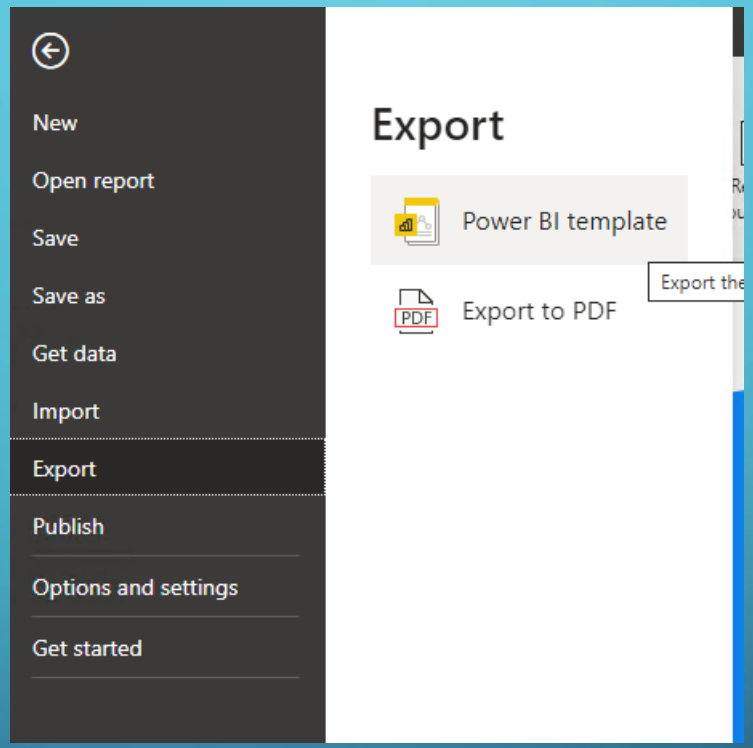
Steps



Microsoft

POWER BI - TEMPLATES

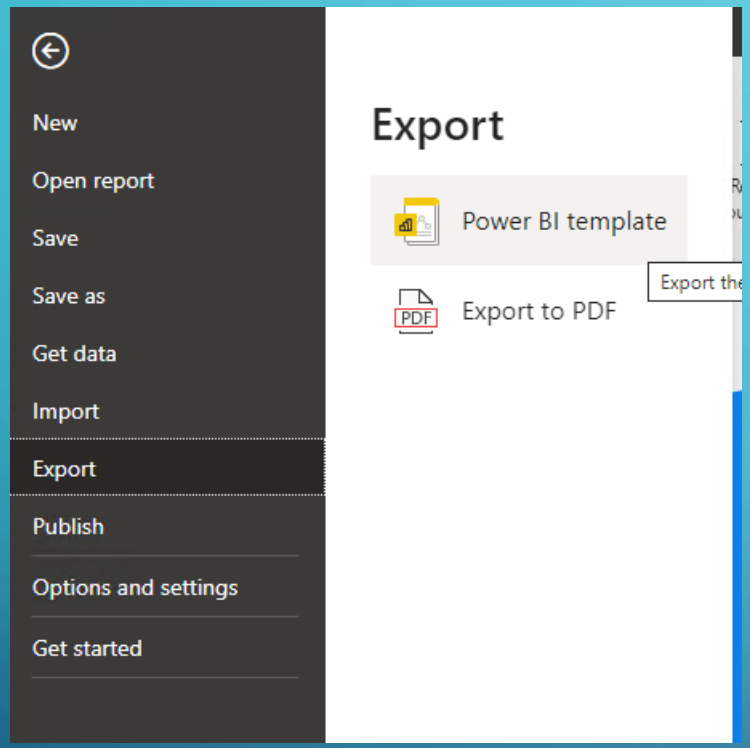
Steps

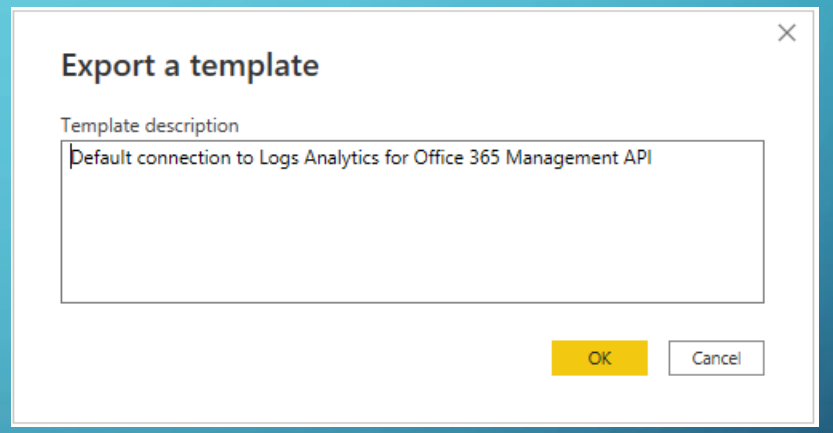


Microsoft

POWER BI - TEMPLATES

Steps

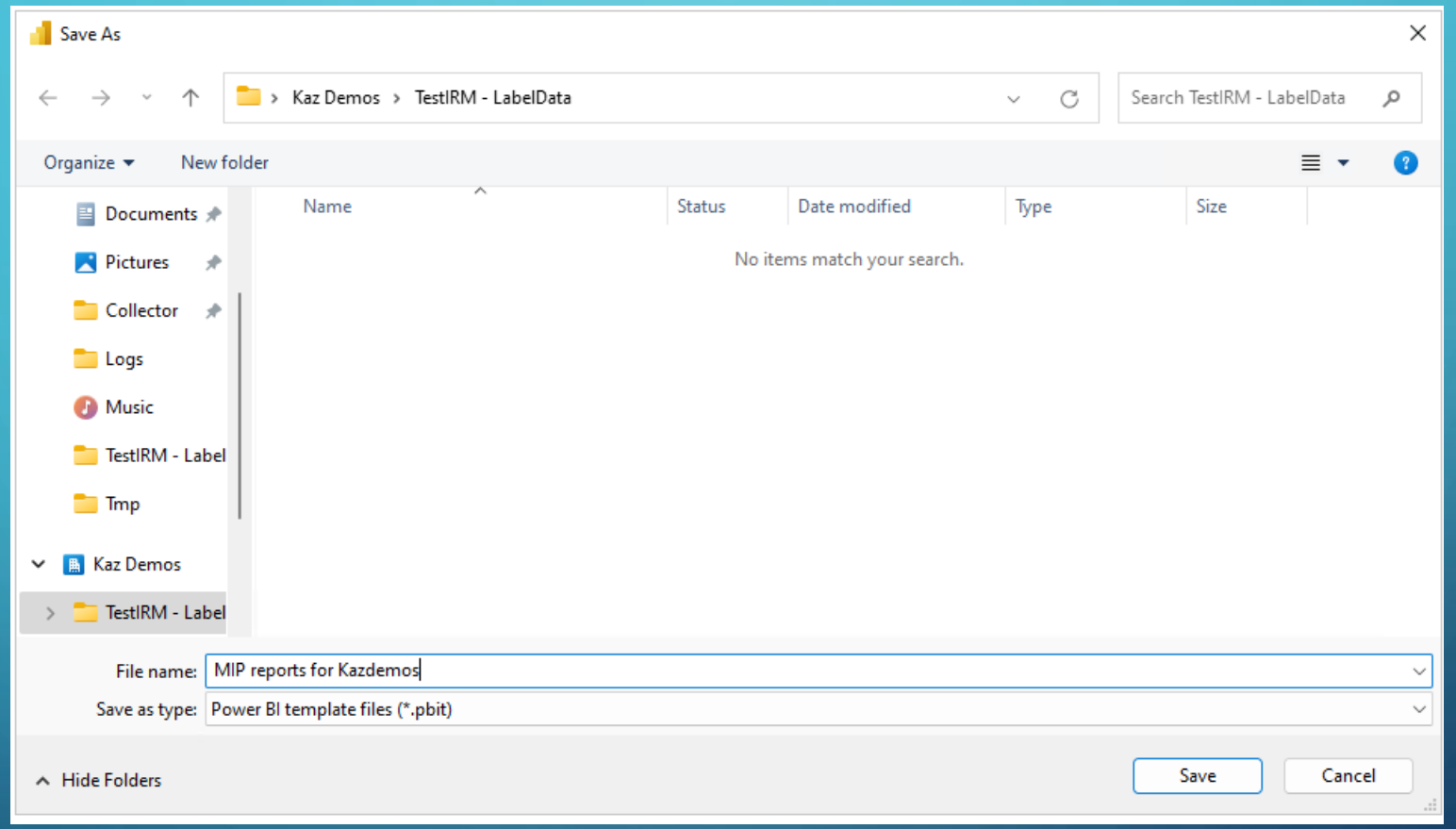




Microsoft

POWER BI - TEMPLATES

Steps



CHECKLIST TO IMPORT POWER BI TEMPLATE

TO IMPORT ADVANCED POWER BI TEMPLATE

- Validate **these scripts was executed first**:
  - Export_logs.ps1 and is running under task scheduler
  - .\ExportCSV2LA.ps1 -FileName '.\Support Data\Product names and service plan identifiers for licensing.csv' -TableName MSProducts (was executed and the Tablename is created with the same name shown)
  - .\Get-AzureADData.ps1 (run on demand)
  - .\Get-AzureADRoles.ps1 (run on demand)
  - .\Get-AzureADDomains.ps1 (run on demand)
  - .\Get-LabelData.ps1 (run on demand)
  - .\Get-RMSData.ps1 and is running under task scheduler
- If it's the first time wait around 15 minutes until the tables are populated and appear under Custom Logs on Logs menu in your Logs Analytics workspace

POWER BI – IMPORT TEMPLATE

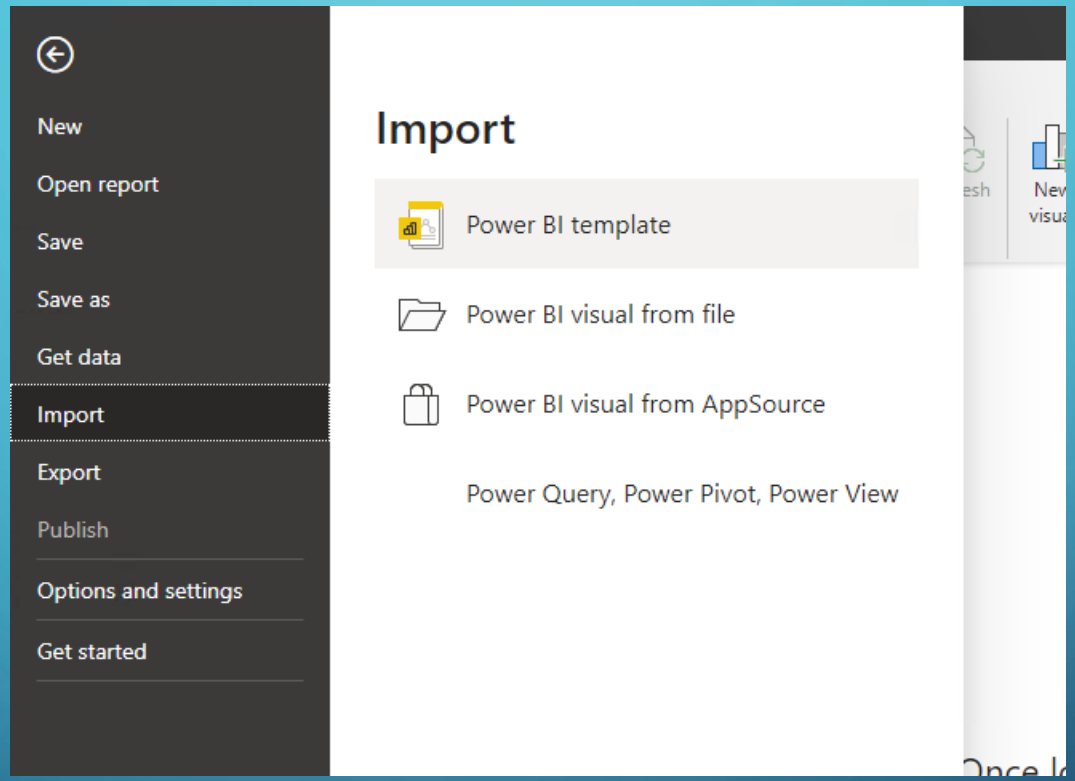
To import a Template

- With this steps we will reduce a lot the previous effort to connect each data source and all the previous configuration
- Open Power BI Desktop
- Close the pop-up window
- Go to File then Import and select Power BI template
- Locate your template file, select them and press Open
- Workspace_ID will be required, enter your Workspace ID and press Load
- Depending the account used as sign-in on Power BI, new credentials will be required
- To validate the right settings, open Power Query Editor doing a right click over any of the table names located at left under Fields and selecting Edit query
- Check Workspace_ID variable that match with your Workspace ID
- Validate that you are using the right matrix for Label names and GUIDs, in case that not match with the current workspace ID information, remove and create a new one based on previous steps explanation.
- Close & Apply

Microsoft

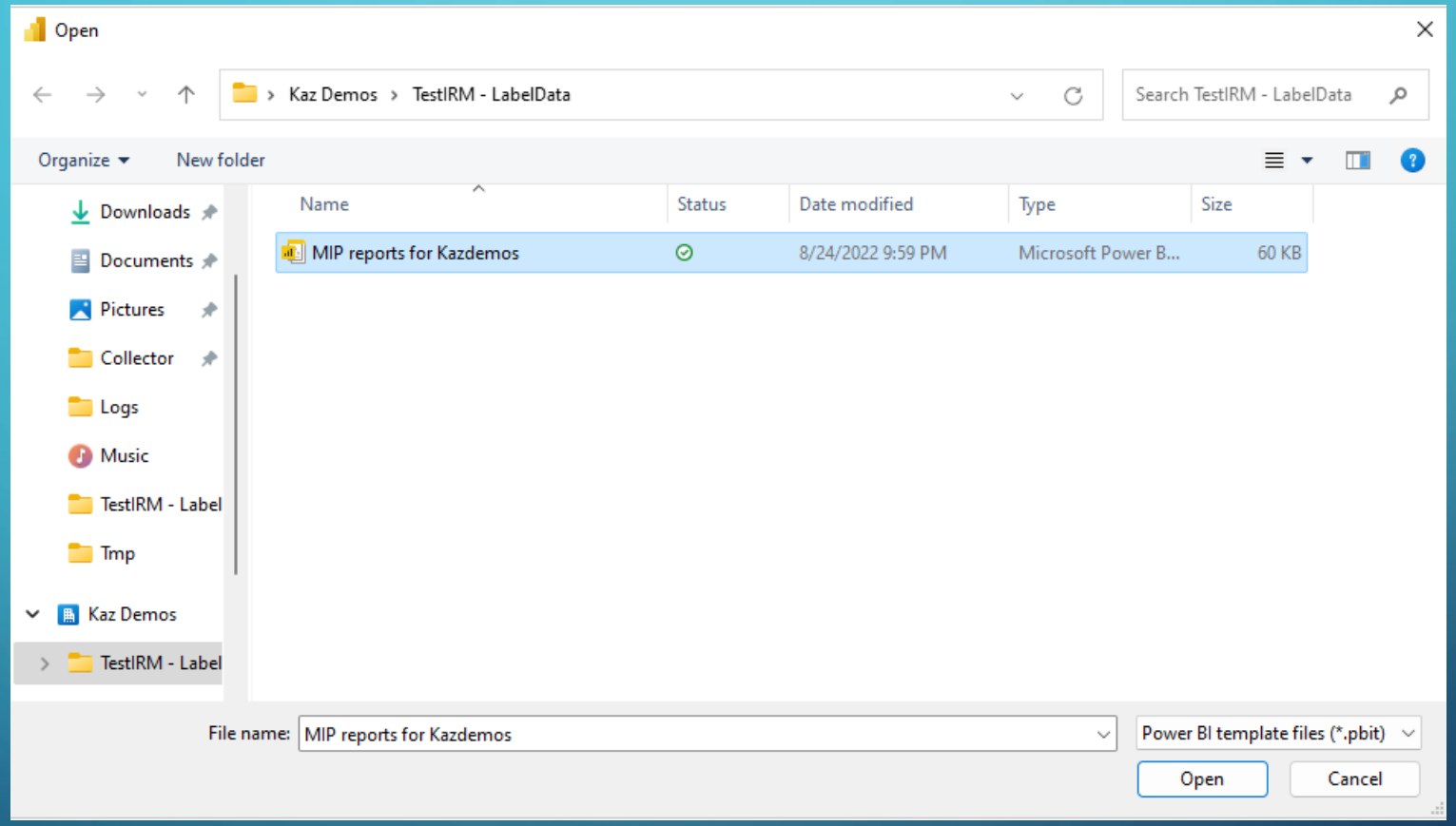
POWER BI - TEMPLATES

Steps



POWER BI - TEMPLATES

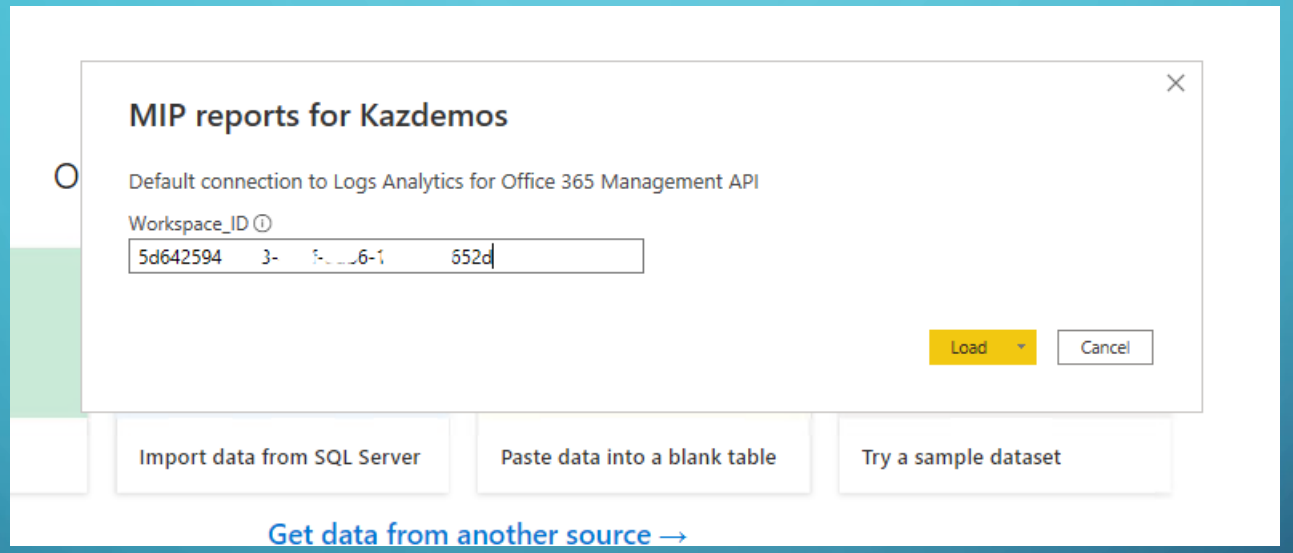
Steps



Microsoft

POWER BI - TEMPLATES

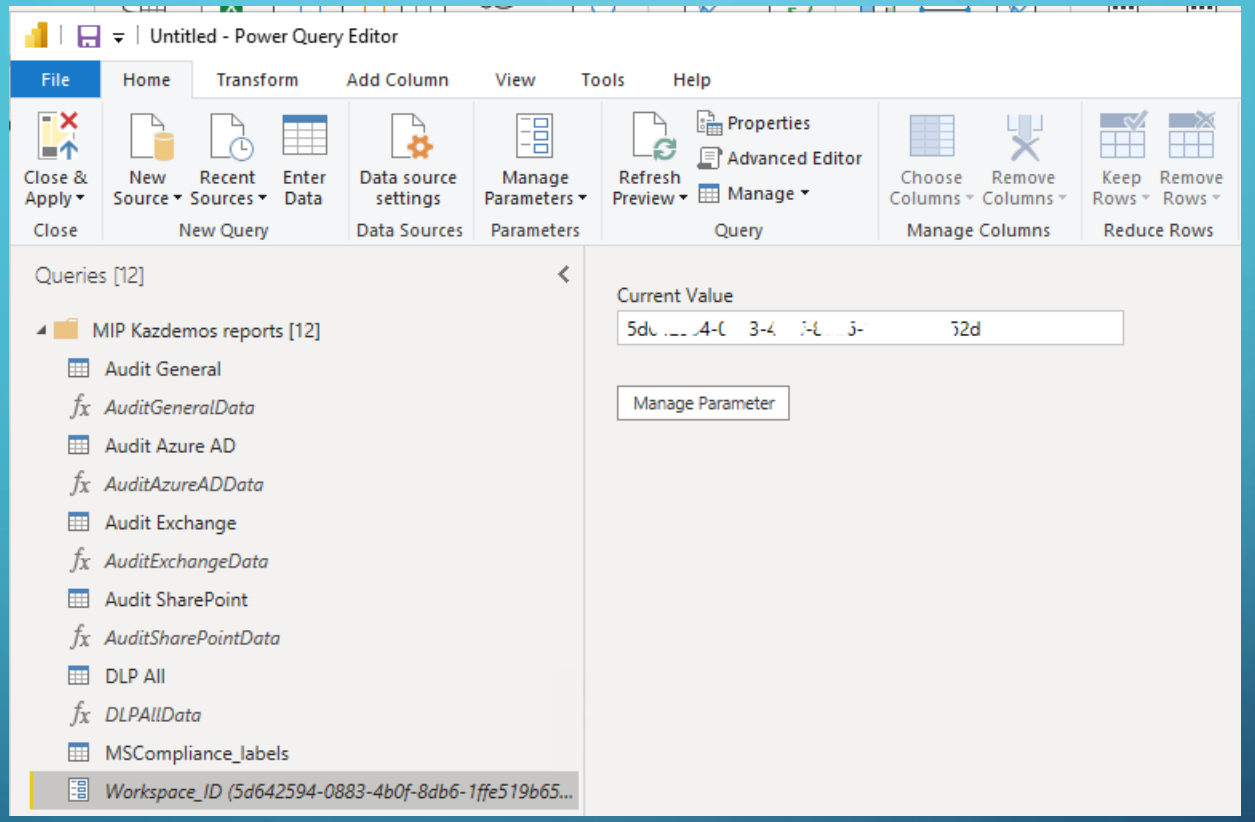
Steps



Microsoft

POWER BI - TEMPLATES

Steps



Microsoft

GEO LOCATION ON POWER BI

TO CREATE A FUNCTION ON POWER BI FOR GEO LOCATION BASED ON IP ADDRESSES

- Open Power BI, press "**Get Data**" and select "**Blank query**" in the new window open select "**Advanced Editor**" and replace the content with this Function.

```
let
Source = (#"IP Address" as text) => let
Source = Json.Document(Web.Contents("http://ipwhois.app/json/" & #"IP Address")),
#"Converted to Table" = Record.ToTable(Source),
#"Transposed Table" = Table.Transpose(#"Converted to Table"),
#"Promoted Headers" = Table.PromoteHeaders(#"Transposed Table")
in
#"Promoted Headers"
in
Source
```

- At left menu change the name to something like as "GeoLocationFunction"

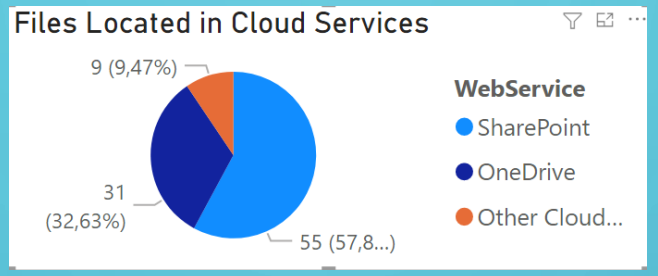
*This report uses an external service, https://ipwhois.io/, and can have additional costs

Microsoft

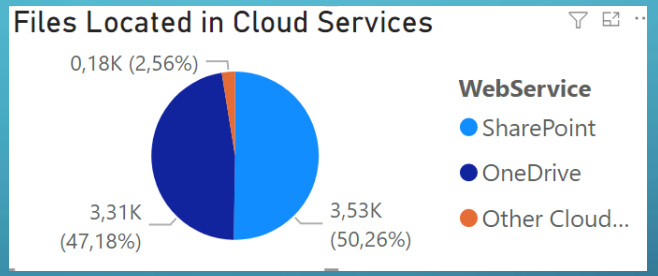
Tips

Avoid some bad reports

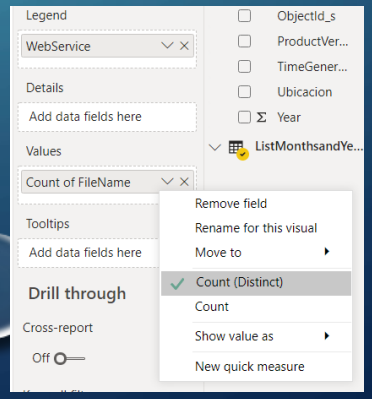
- Sometimes we need take care with the values and filters used when we create our reports to show wrong information, here an example:



- In the image shown we add a filter to count distinct FileName, but the Legend is related to Files located in Cloud Services, if we can not do that, we obtain this result:



- You can make the change in the same Power Bi on filters



Microsoft

Links

Links of Interest

- Incremental refresh in Power BI
- Sensitivity labels in Power BI
- Exceed the 500,000 row limit in Application Insights and Log Analytics with Power BI
- Kusto

URL Used

- Power BI Query Editor - Getting IP Address Details from IP Address - Reporting/Analytics Made easy with FourMoo and Power BI

Geo Location services by IP

- 14 Best IP Geolocation API to Offer Personalized Content (geekflare.com)

Geo location APIs tested

- IP Lookup API and IP Geolocation Documentation (ipwhois.io)
- IP-API.com - Geolocation API - Documentation - JSON (ip-api.com)
- Pricing | IP Geolocation API (ipify.org)
- IP Geolocation API with country information
- Free IP Geolocation API and Accurate IP Geolocation Database

Microsoft

THANK YOU

Sebastián Zamorano

+56 9 5207 3058

sebastian.zamorano@microsoft.com

https://aka.ms/MPARR-GitHub

myprofile.kazblog.me

Microsoft

TOPICS PENDING TO RELEASE

Some additional effort to release soon

- Kusto queries for the new schema, the idea is download only columns Required for this reports
- Some examples and templates for Power BI dashboards
- Geo location on Power BI based on IP address
- Add Azure AD schema to generate reports based on areAs or departments
- Add Azure RMS logs to show access from external users to protected Information
- Incremental refresh on Power BI to reduce Logs Analytics cost
- Publish different reports for separate audience on Power BI Online
- Fields dictionary
- Relationship map between tables

Microsoft