

# **Configuration Manual**

MSc Research Project MSc Cyber Security

Amiket Kumar Srivastava Student ID: X22119451

School of Computing National College of Ireland

Supervisor: Dr Vanessa Ayala-Rivera

### National College of Ireland



#### **MSc Project Submission Sheet**

#### **School of Computing**

Student Name:	Amiket Kumar Srivastava	
Student ID:	22119451	
Programme:	MSCCYB1	<b>Year:</b> 1
Module:	MSc Research Project	
Supervisor:	Dr Vanessa Ayala-Rivera	
Date:	14 <sup>th</sup> December, 2023	
Project Title:	Designing the Architecture of an Efficient Cl Security Posture Management System	loud-based Data

 Word Count:
 2330
 Page Count:
 14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Amiket Kumar Srivastava

**Date:** 14/12/2023

#### PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	
Attach a Moodle submission receipt of the online project	
submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project,	
both for your own reference and in case a project is lost or mislaid. It is	
not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

# **Configuration Manual**

## Amiket Kumar Srivastava Student ID: X22119451

# **1** Introduction

This configuration manual provides detailed instructions on setting up the architecture for the data security posture management system (DSPM). It allows us to implement the various components that we have researched for our DSPM system. This research showcases the combined use of Microsoft Purview Data Catalogue Services with Python and PowerShell Scripts and Azure DevOps Services to form a cost-effective and efficient DSPM system. Azure Cloud Services have been used to create a sample environment for testing the different components of the DSPM system. This manual is divided into 4 major sections -1) Azure Cloud Services Configuration, 2) Microsoft Purview Data Catalogue Configuration, 3) Azure DevOps Services Configuration and 4) Execution.

# 2 System Configuration

The required system configuration is given in below subsections.

## 2.1 Hardware Configuration

Operating System: - Windows 11 Home Single Language version 22H2 Processor: 11th Gen Intel(R) Core (TM) i9-11900H @ 2.50GHz 2.50 GHz System: 64-bit operating system, x64-based processor Hard drive: 1 TB Memory (RAM): 16 GB

## 2.2 Software Configuration

Tools used:

Tool	Version	Description
Azure DevOps Self-hosted	3.230.2	Agent was used to run Azure DevOps
Agent		pipeline jobs.
Python 3	3.12.1	Python3 was used to run the automation
		scripts on the self-hosted Azure DevOps
		agent.
PowerShell	5.1.22621.2506	PowerShell was used to run the
		automation scripts on the self-hosted
		Azure DevOps agent.
Microsoft Purview Account	NA	Purview was used for data catalogue
SaaS		services.
Azure Service Principal SaaS	NA	Service Principal was used to access

Azure Managed Identity SaaS	NA	Managed Identity was used to integrate
		Purview with other cloud services.
Azure Storage Account SaaS	Standard	Blob storage was used to store dynamic
		data backups of sensitive data sources.
Azure SQL Server SaaS	12.0.2000.8	SQL DB was used to test dynamic data
		masking and dynamic data backups
		policies.
Azure Key Vault SaaS	Standard	Key Vault was used to store secrets
		which were pulled later in pipelines and
		used to run scripts securely.
Tableau Desktop	23.3.345	Tableau was used to create visualisations
		from the data and publish reports.

Python libraries used:

Library	Version	Description
requests	2.31.0	Agent was used to run Azure DevOps
		pipeline jobs.
pandas	2.1.1	Python3 was used to run the automation scripts on the self-hosted Azure DevOps agent.
pyapacheatlas	0.15.0	PowerShell was used to run the automation scripts on the self-hosted Azure DevOps agent.
openpyxl	3.1.2	Purview was used for data catalogue services.

# **3** Execution

This section describes the installation, working and execution of the DSPM system.

## **3.1** Software Installation

• Azure DevOps self-hosted agent version 3.230.2 was used as it was the latest stable version. This tool allowed us to run pipeline jobs for free. Agent was downloaded from the Azure DevOps official site and installation was performed by following the instructions as shown in Figure 1.

indows	macOS	Linux
Indows	macos	LINUX
	System prerequisit	tes
Configure you Configure your ac	r account count by following the steps outlined <u>he</u>	<u>re</u> .
Download the	agent	
Download		
Create the age	ent	
PS C:\> mkdin PS C:\agent> [System.IO.Co agent-win-x64	• agent ; cd agent Add-Type -AssemblyName System.IC mpression.ZipFile]::ExtractToDir  -3.230.2.zip", "\$PWD")	).Compression.FileSystem ; ectory("\$HOME\Downloads\vsts-
Configure the	agent Detailed instructions 🖪	
PS C:\agent>	.\config.cmd	
Optionally run	the agent interactively	
lf vou didn't run a	a service above:	
	indows Configure you Configure your acc Download the Download the Download Create the age PS C:\> mkdir PS C:\agent> Configure the PS C:\agent> Optionally run	indows       macOS         System prerequisit         Configure your account         Configure your account by following the steps outlined here         Download       Download         Download       Download         Create the agent       Create the agent         PS C:\> mkdir agent ; cd agent PS C:\agent> Add-Type -AssemblyName System.10 [System.10.Compression.ZipFile]::ExtractToDir agent-win-x64-3.230.2.zip", "\$PWD")         Configure the agent Detailed instructions L*         PS C:\agent> .\config.cmd         Optionally run the agent interactively

 Python 3.12.1 was used as it was the latest stable version with proper security updates as shown in Figure 2. Link to download Python – <u>https://www.python.org/downloads/release/python-396/</u>



Figure 2 - Download Python

• PowerShell 5.1.22621.2506 was used as it was the latest stable version with proper security updates that was already available on the local windows machine that hosted the Azure DevOps agent as shown in Figure 3.

Name	Value 
PSVersion	5.1.22621.2506
PSEdition	Desktop
PSCompatibleVersions	{1.0, 2.0, 3.0, 4.0}
BuildVersion	10.0.22621.2506
CLRVersion	4.0.30319.42000
WSManStackVersion	3.0
PSRemotingProtocolVersion	2.3
SerializationVersion	1.1.0.1

**Figure 3 - PowerShell Details** 

• Microsoft Purview Account was created in Azure and used for the data catalogue services as shown in Figure 4.

Microsoft Purview + dspmcatalogue	崎 🖘 琵 ಧ ⑳ ? 🖓 amiket.cad@gmail.com 🚰
» »	
📕	
•	dspmcatalogue
•	
<u></u>	G 3 sources IIII 69 assets III 1 glossary term
	O Search catalog
8	
	Browse assets         Manage glossary           Explore assets by source type and collection.         Create and edit terms with custom term templates.         Siccover learning materials and tutorials.
Recently accessed	My items Links
	Microsoft Purview OP
	Discover the capabilities offered by Microsoft Purview and learn how to make the most of them.
	Getting started 🖸
	Figure 4 Migrosoft Purview Date Catalogue

Figure 4 - Microsoft Purview Data Catalogue

• A Service Principal was created in Microsoft Entra ID and given IAM permissions to all the resources in Azure including API permissions as shown in Figure 5. This was used in scripts to call API endpoints.

	rmissions 🖈 …					
♀ Search «	🕐 Refresh 🕴 🗖 Got feedbac	k?				
Overview	The "Admin consent required" in organizations where this appresent the second secon	column shows th o will be used. Le	e default value for an organization. However, user const earn more	ent can be customized per permission,	user, or app. This column may no	t reflect 1
📣 Quickstart		_				
💅 Integration assistant	Configured permissions					
Manage	Applications are authorized to call A all the permissions the application r	APIs when they needs. Learn mo	are granted permissions by users/admins as part of ore about permissions and consent	the consent process. The list of conf	figured permissions should incl	lude
Branding & properties						
Authentication	Add a permission V Grant	t admin consent	for Default Directory			
📍 Certificates & secrets	API / Permissions name	Туре	Description	Admin consent requ	Status	
Token configuration	→ Azure Storage (1)					
API permissions	user_impersonation	Delegated	Access Azure Storage	No	♂ Granted for Default Dire	•••
Expose an API	✓ Microsoft Graph (4)					
App roles	Directory.Read.All	Application	Read directory data	Yes	Granted for Default Dire	
A Owners	Directory.ReadWrite.All	Application	Read and write directory data	Yes	♂ Granted for Default Dire	
👃 Roles and administrators	Directory.Write.Restricted	Application	Manage restricted resources in the directory	Yes	♂ Granted for Default Dire,	
0 Manifest	User.Read	Delegated	Sign in and read user profile	No	Sranted for Default Dire	•••

Figure 5 - API Permissions Assigned to Service Principal

• A Managed Identity was created in Azure and used for integrating Purview with other cloud resources so that catalogue scans could be run. It was given IAM permissions to all other resources and then integrated with the data catalogue as shown in Figure 6.

<b>dspmcatalogue</b>   M Microsoft Purview account	lanaged identities (previe	ew) ☆ …	
✓ Search «	User assigned		
S Overview			
Activity log	User assigned managed identities enabling resources, and have their own lifecycle.	e Azure resources to authenticate A single resource (e.g. Virtual Mac	to cloud services (e.g. Azure Key Vault) without stor hine) can utilize multiple user assigned managed ide
Access control (IAM)	resources (e.g. Virtual Machine).		
🗳 Tags	🕂 Add 📋 Remove 💍 Refresh	Got feedback?	
🗙 Diagnose and solve problems			
<sup>8</sup> ∞ Root collection permission	Name	$\uparrow_{\downarrow}$	Resource group
Settings	researchidentity		rg_purview

Figure 6 - Managed Identity Integration with Data Catalogue

• Storage Account was created, and Blobs were configured so that dynamic data backup policy could be implemented as shown in Figure 7. Logs were switched to 2.0 in Diagnostic Settings (classic) so that data access could be monitored as shown in Figure 8.

Name	Last modified	Anonymous access level	Lease state
\$logs	27/11/2023, 9:02:16 am	Private	Available
sample	27/11/2023, 9:22:17 am	Container	Available
sqlbackups	30/11/2023, 1:33:30 am	Container	Available

Figure 7 - Blobs Set Up for Dynamic Data Backup Policy



Figure 8 - Storage Account Diagnostic Settings

- Azure SQL Server was deployed, and a database was configured so that dynamic data backup and dynamic data masking policies could be implemented. The server was then connected to using SQL Server Management Studio as shown in Figure 9. The following queries were then run on master and target database:
  - O "CREATE LOGIN researchadmin WITH password='\*\*\*\*\*';"
  - o "CREATE USER researchadmin WITH password='\*\*\*\*\*';"
  - o "EXEC sp\_addrolemember 'db\_owner', [researchadmin]".

The password was stored in the Key Vault.



Figure 9 - Azure SQL Server Connected via SSMS

• A Key Vault was created to store secrets such as service principal details, SQL account password and storage keys which were retrieved later in pipelines to run scripts. Vault Permission model was switched to Vault Access policy and the service principal and Azure DevOps account were given permissions as shown in Figure 10.

Name	Туре	Status	Expiration date				
appid		✓ Enabled		Permission model			
appsecret		✓ Enabled		Grant data plane access by using a Azure RBAC or Key Vault access policy			
sqlkey		✓ Enabled					
storagekey	rragekey 🗸 Enabled			Azure role-based access control (recommended)			
storagetoken		✓ Enabled		Valit access poincy ()			
tenantid		✓ Enabled		Go to access policies			



• Tableau Desktop 23.3.345 was used as it was available for a free 1-year license and it was the latest stable version with proper security updates as shown in Figure 11.



Figure 11 - Tableau Desktop

## 3.2 Set up Data Scanning, Classification and Glossary Terms

• Register All the Data Sources in Purview from Data Map as shown in Figure 12.



Figure 12 - Registering a Data Source in Purview

• Create Custom Classification Rules from Data Map as shown in Figure 13

Classi	Classification rules					
+ N6	ew 🖉 Edit 📋 Delete 💍 Refresh 🗌 Disabl	e ▷ Enable				
Syster	m Custom					
∀ Fil	iter by name					
	Name	Data pattern	Column pattern	Match %	Last applied	State
	cc_expiredate		cc_expiredate		11/29/2023, 8:4	🕑 Enabled
	cc_cvc		cc_cvc		11/29/2023, 8:4	🕑 Enabled
	cc_number_restricted		cc_number		11/29/2023, 8:4	🕑 Enabled
	phone_restricted		phone		11/29/2023, 8:4	🕑 Enabled
	address_restricted		address		11/29/2023, 8:4	🕑 Enabled
	email_restricted		email		11/29/2023, 8:4	🕑 Enabled
	DOB_Restricted		birthdate		11/29/2023, 8:4	🕑 Enabled

**Figure 13 - Creation of Classification Rules** 

• Schedule Scans for all the Data Sources as shown in Figure 14

Scan "AzureSqlDatabase-gGg"

Name *	
Scan-zN7	
Connect with integration runtime * $\odot$	
Azure AutoResolveIntegrationRuntime	~ 0
erver endpoint *	
dspmresearch.database.windows.net	
ielect database	
From Azure subscription O Enter mani	ually
Database name *	
Select	O
Credential *	
Microsoft Purview MSI (system)	~ 0
P Before you set up your scan you must giv account permissions to connect to your A	re the managed identity of the Microsoft Purview Azure SQL Database. Show more $\checkmark$
ineage extraction (preview) ① On	
	STest connection Cancel

**Figure 14 - Scheduling Scans for Data Sources** 

• Create Custom Data Source Schema in a JSON file as shown in Figure 15. An example of this file is published in the artifacts as typedef.json.

```
"entityDefs": [
                           {
                                                      "category": "ENTITY",
"name": "customer",
"description": "Customer Information",
"serviceType": "customer_info",
"options": {
    "schemaElementsAttribute": "columns"
                                                         },
"attributeDefs": [
                                                                                     {
                                                                                                             "name": "first_name",
"typeName": "string",
"isOptional": true,
"cardinality": "SINGLE",
"valuesMinCount": 0,
"valuesMaxCount": 1,
"idUnion": 6,
"addes": 6,

                                                                                                               "isUnique": false,
"isIndexable": true,
"includeInNotification": false
                                                                                     },
                                                                                     {
                                                                                                               "name": "last_name",
"typeName": "string",
"isOptional": true,
"cardinality": "SINGLE",
"valuesMinCount": 0,
                                                                                                                  "valuesMaxCount": 1,
                                                                                                                  "isUnique": false,
"isIndexable": true,
                                                                                                                  "includeInNotification": false
                                                                                     },
                                                                                       {
                                                                                                                  "name": "maiden name",
                                                                                                                "typeName": "string",
"isOptional": true,
"cardinality": "SINGLE",
                                                                                                                  "valuesMinCount": 0,
"valuesMaxCount": 1,
                                                                                                                  "isUnique": false,
"isIndexable": true,
                                                                                                                     "includeInNotification": false
                                                                                     },
                                                                                       {
                                                                                                                  "name": "gender",
"typeName": "string",
"isOptional": true,
```

Figure 15 - Custom Schema File

• Upload this schema to Purview using the upload\_typdef.py python script. The script takes three parameters as input. The first one is used to supply tenant id of the azure environment. Second parameter is used to supply the service principal application id and the third parameter is used to supply the service principal application secret to the script. A fourth parameter is used to supply the path to the custom schema JSON file which is then uploaded to Purview using the pyapacheatlas library as shown in Figure 16.

1	import os
2	import json, sys
3	
4	from pyapacheatlas.auth import ServicePrincipalAuthentication
5	<pre>from pyapacheatlas.core import AtlasAttributeDef, AtlasEntity, PurviewClient</pre>
6	<pre>from pyapacheatlas.core.typedef import EntityTypeDef, TypeCategory</pre>
7	
8	ifname == "main":
9	
10	This sample provides shows how to create custom type definitions and
11	how to creates entities using a custom type.
12	
13	
14	oauth = ServicePrincipalAuthentication(
15	<pre>tenant_id=sys.argv[1],</pre>
16	<pre>client_id=sys.argv[2],</pre>
17	client_secret=sys.argv[3]
18	)
19	<pre>client = PurviewClient(</pre>
20	account_name="dspmcatalogue",
21	authentication=oauth
22	)
23	
24	input_path = sys.argv[4]
25	<pre>client.upload_typedefs(json.load(open(input_path, 'r')), force_update=True)</pre>
26	

Figure 16 - Python Script to upload Custom Schema

- Next, classify the non-standard dataset using the classify.py script. This script reads the dataset and applies classification to its values based on a dictionary file. The dictionary file is available in artifacts as dictionary.csv.
- Now ingest the custom data source into Purview by running the upload\_entities.py script. This script parses the classified dataset and uploads it as entities to Purview. Figure 17 shows a custom type of entity and its classification value in Purview.

Customer + Add Tag
🖉 Edit 🕀 Select for bulk edit 🖓 Request access 🖒 Refresh 📋 Delete
Overview Properties Schema Lineage Contacts Related
Asset description
No description for this asset.
Managed attributes           Image: Tribute stribute
No Attributes for this asset.
Classifications $^{\bigcirc}$
Restricted
Schema classifications (1) $^{\odot}$

Figure 17 - Custom Type Entity and its Classification Value

• Create Data Residency Terms in Glossary from data catalog menu. Next, assign these terms to entities that have such legal requirements as shown in Figure 18.



Figure 18 - Data Residency Term Assignment

### **3.3** Setup Pipeline for Automation

• Modify the upload-typedef-pipeline.yml file from the repository as per your environment requirements. The trigger is set to none as we only need to run this schema once unless any modification is needed for the dataset. The first task is to connect to Azure Key Vault and retrieve secrets for the pipeline. Next task is for the python script to upload the custom schema and 4 parameters are supplied to the script – tenant id, app id, app secret and path to the JSON file. The pythonInterpreter path should be the path to the python library which was installed inside the Azure DevOps Agent directory.

```
1
2 trigger:
3 - none
4
5 pool: researchpool
6
7 steps:
8 - task: AzureKeyVault@2
9
   inputs:
10
      azureSubscription: 'Free Trial(cc2563bf-8958-4bca-8747-fceb708b8330)'
      KeyVaultName: 'dspmresearchvault'
11
     SecretsFilter: '*
12
13 RunAsPreJob: false
14
15 - task: PythonScript@0
16 displayName: 'Upload custom schema to data catalogue for special data types'
17
    inputs:
18
      scriptSource: 'filePath'
      scriptPath: '$(System.DefaultWorkingDirectory)\upload_typedef.py'
19
20
     arguments: '$(tenantid) $(appid) $(appsecret) $(System.DefaultWorkingDirectory)\typedef.json'
21
     pythonInterpreter: 'C:\agent\_work\_tool\python.exe'
22 condition: succeeded()
23
```

#### Figure 19 - Pipeline to Upload Custom Schema

• Next, modify the upload-entities-pipeline.yml file from the repository as per your environment needs. This pipeline again has the key vault task in the beginning. The second task is to upload the dataset to purview as entities using Purview API endpoints

which are used in the script (Yenamandra, Yunair, VladR, & Taojunshen, 2023). It accepts 4 parameters - tenant id, app id, app secret and path to the dataset that needs to be ingested. This dataset is available in the artifacts as sample-data.csv.

1

```
2 trigger:
3 - none
Δ
5 pool: researchpool
6
7 steps:
8 - task: AzureKeyVault@2
9
    inputs:
10
      azureSubscription: 'Free Trial(cc2563bf-8958-4bca-8747-fceb708b8330)'
      KeyVaultName: 'dspmresearchvault'
11
      SecretsFilter: '*
12
     RunAsPreJob: false
13
14
15 - task: PvthonScript@0
16 displayName: 'Upload special custom data to data catalogue'
17
    inputs:
      scriptSource: 'filePath'
18
19
      scriptPath: '$(System.DefaultWorkingDirectory)\upload_entities.py'
20
      arguments: '$(tenantid) $(appid) $(appsecret) $(System.DefaultWorkingDirectory)\sample-data.csv'
     pythonInterpreter: 'C:\agent\_work\_tool\python.exe'
21
22 condition: succeeded()
```

## Figure 20 - Pipeline to Upload Custom Entities

• Next, modify the access-log-pipeline.yml file from the repository as per your environment requirements. The pipeline is being run on a daily schedule by default but can be adjusted by the user as needed. This pipeline again has the key vault task in the beginning. The second task runs the access.py file which reads the access logs of azure storage account and produces a report of requestor IP addresses and their location of origin. This file accepts 2 parameters – storage token and path for output report file. The report is then published as an artifact by the pipeline as shown in Figure 21.

```
3 - none
4
5 schedules:
6 - cron: '0 1 * * *'
    displayName: Daily midnight build
7
8
    branches:
9
     include:
10 - main
11
12 pool: researchpool
13
14 steps:
15 - task: AzureKeyVault@2
16
    inputs:
      azureSubscription: 'Free Trial(cc2563bf-8958-4bca-8747-fceb708b8330)'
17
      KevVaultName: 'dspmresearchvault
18
19
      SecretsFilter: '*
20 RunAsPreJob: false
21
22 - task: PythonScript@0
23 displayName: 'Analyse Access Logs to implement Geo-Fencing
24
    inputs:
25
      scriptSource: 'filePath'
      scriptPath: '$(System.DefaultWorkingDirectory)\access.py'
26
      arguments: '$(storagetoken) $(System.DefaultWorkingDirectory)\researchdocs\AzBlobLogs.csv'
27
28
     pythonInterpreter: 'C:\agent\_work\_tool\python.exe'
29
    condition: succeeded()
30
31 - publish: $(System.DefaultWorkingDirectory)\researchdocs\AzBlobLogs.csv
32 artifact: AzBlobLogs.c
```

**Figure 21 - Geo Fencing Pipeline** 

• Next, modify the research-pipeline file from the repository as per your environment needs. The pipeline is also being run on a daily schedule by default but can be adjusted by the user as per their requirements. This pipeline again has the key vault task in the beginning. The second task fetches a list of all the resources in the cloud environment and adds it to the artifact. The third task uses this resource list and grabs the IAM permissions for each resource one by one and concatenates the information into a single IAM.csv file which is added to the artifacts as shown in Figure 22.

```
14 steps:
15

    task: AzureKeyVault@2

     inputs:
        azureSubscription: 'Free Trial(cc2563bf-8958-4bca-8747-fceb708b8330)'
17
18
       KeyVaultName: 'dspmresearchvault
        SecretsFilter: '*'
19
20
       RunAsPreJob: false
21
22
23
   - task: PowerShell@2
     displayName: 'Get List of Resources'
24
     inputs:
        targetType: filePath
filePath: '$(System.DefaultWorkingDirectory)\resourcelist.ps1'
25
26
27
       arguments: '-resourcelist "$(System_DefaultWorkingDirectory)\researchdocs\resourcelist.csv" -appid "$(appid)" -appsecret "$(appsecret)" -tenantid "$(tenantid)"
28
   - task: PowerShell@2
     displayName: 'Get IAM of Resources
31
     inputs:
targetType: filePath
32
33
        filePath: '$(System.DefaultWorkingDirectory)\IAM.ps1
       arguments: '.iam "$(system.DefaultWorkingDirectory)\researchdocs\IAM.csv" -appid "$(appid)" -appsecret "$(appsecret)" -tenantid "$(tenantid)"'
34
```

Figure 22 - Tasks to get IAM Permissions Report

• The fourth task in this pipeline gets the sensitivity label values for columns inside SQL tables from the data catalogue and adds that information inside a file using the name of the table in the artifacts which in our case was account\_info.csv. The fifth task reads these sensitivity values and creates dynamic data masking rules for columns with sensitive data if they aren't already created as shown in Figure 23.

```
- task: PvthonScript@0
 displayName: 'Get SQL Table Column Sensitivity values from Data Catalogue
 inputs:
    scriptSource: 'filePath'
   scriptPath: '$(System.DefaultWorkingDirectory)\ddm.py'
   arguments: '$(tenantid) $(appid) $(appsecret) $(System.DefaultWorkingDirectory)\researchdocs'
   pythonInterpreter: 'C:\agent\ work\ tool\python.exe
 condition: succeeded()
 task: PowerShell@2
 displayName: 'Use sensitivity values from previous step to create Dynamic Data Masking Rules'
 inputs:
               filePath
   targetType:
   filePath: '$(System.DefaultWorkingDirectory)\DDM.ps1'
   arguments: '-file "$(System.DefaultWorkingDirectory)\researchdocs\account_info.csv" -appid "$(appid)" -appsecret "$(appsecret)" -tenantid "$(tenantid)"'
    . . . . . . .
```

### Figure 23 - Tasks to Set Up Dynamic Data Masking Policy

• The sixth task in this pipeline gets the sensitivity information about all the cloud resources from the data catalogue and adds that information inside a file called resource\_risk.csv in the artifacts. The seventh task reads these sensitivity values and creates dynamic data backups for databases with sensitive data inside Azure blob storage as shown in Figure 24.

```
2 - task: PythonScript@@
3 displayMame: 'Get Sensitivity information about resources from Data Catalogue'
1 inputs:
5 scriptSource: 'filePath'
5 scriptSource: 'filePath'
5 scriptSource: 'filePath'
7 arguments: 'S(tenantid) S(appld) S(appsecret) S(System.DefaultWorkingDirectory)\researchdocs'
1 pythonInterpreter: 'C:\agent\_work\_tool\python.exe'
2 condition: succeeded()
3 to task: PowerShall@2
1 otask: PowerShall@2
1 otask: PowerShall@2
1 argetType: filePath
1 filePath: 'S(System.DefaultWorkingDirectory)\backup.p1'
1 argetType: filePath
1 filePath: 'S(System.DefaultWorkingDirectory)\backup.p1'
1 argetType: filePath
1 filePath: 'S(System.DefaultWorkingDirectory)\backup.p1'
1 arguments: '-file "$(System.DefaultWorkingDirectory)\backup.p1'
1 arguments: '-file "$(system.DefaultWorkingDirectory)\backup.p1'
1 arguments: '-file "$(system.DefaultWorkingDirectory)\backup.p1')
1 arguments: '-file "$(system.DefaultWorkingDirectory)\backup.p2')
1 arguments: '-file "$
```

Figure 24 - Tasks to Set Up Dynamic Data Backups

• The eighth task in this pipeline gets the data residency information about all the cloud resources from the data catalogue and adds that information inside a file called residencyaudit.csv in the artifacts. The ninth task reads these residency requirements

and checks each resource's deployment location against it. It gives a pass rating to resources whose values match and fails those resources whose values don't match. It exports this information inside a file called residencycompliance.csv and adds it to the artifacts.



### Figure 25 - Tasks to Audit for Residency Compliance

• The last or tenth task of the pipeline publishes the artifacts so that they can be downloaded and used by the users for their result analysis.

## 3.4 Read Artifacts and Create Visual Reports

• Download the Geo-Fencing.twb file from the repository and open it in Tableau Desktop. Next, go to the pipeline run for the access-log-pipeline.yml. Download the AzBlobLogs.csv file from the published artifact as shown in Figure 26. Update this file as the data source for the Geo-Gencing.twb file. Now we can use the visual reports to filter out and block IP addresses form unauthorised locations.

<ul><li>← Artifacts</li></ul>			
Published			
Name			
✓  ☐ AzBlobLogs.csv			
AzBlobLogs.csv			

Figure 26 - Published Artifact for Geo-Fencing Pipeline

• Download the IAM.twb file from the repository and open it in Tableau Desktop. Next, go to the pipeline run for the research-pipeline.yml. Download the IAM.csv file from the published artifact as shown in Figure 27. Update this file as the data source for the IAM.twb file. Now we can send these visual reports to the owners of the resources for access review and audit.

Name
〜 回 researchdocs
1 IAM.csv

Figure 27 - Published Artifact for Access Monitoring Pipeline

- Download the entire published artifact from the research-pipeline.yaml pipeline run. This artifact contains the following files as shown in Figure 28:
  - IAM.csv already used for creating visual reports for access review.
  - **account\_info.csv** contains classification sensitivity information about SQL table columns.

- **residencyaudit.csv** contains residency requirement information about all cloud resources.
- **residencycompliance.csv** Audit report of all cloud resources and their Residency Requirement Compliance Status.
- **resource\_risk.csv** contains classification sensitivity information about all cloud resources.
- **resource\_list.csv** is a list of all the cloud resources.



Figure 28 - Published Artifacts for DSPM main pipeline

## **3.5 DSPM Execution**

Now that all the pipelines have been setup, the regular use of this DSPM system can be done by following these steps:

- Register all new data sources into Purview.
- Verify their classifications in Purview and update the rules as needed.
- Periodically download the IAM.csv file and update the IAM.twb visual report and send it to data owners for review.
- Download the RiskAssessmentMatrix.xlsx from the repository and conduct manual assessments. Send reports of these assessments to the data owners for review.
- Periodically download the AzBlobLogs.csv file and update the Geo-Fencing.twb visual report and share it with data owners for further action.
- Download the residency compliance.csv and share any failures to the respective data owners regularly.
- Take regular feedback from different teams and incorporate the feedback in existing policies and procedures. Promote innovation and use ideas to create new policies and procedures.

## References

Yenamandra, N., Yunair, VladR, & Taojunshen. (2023, October 31). *Microsoft Purview Resource Provider Rest API*. Retrieved from Microsoft: https://learn.microsoft.com/en-us/rest/api/purview/