Allied Solutions LLC.

Enterprise Architecture

Power BI Activity Architecture with Snowflake

2023-03-24

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No.** | **Date** | **Author** | **Revision Description** |
| 0.1 | 05-24-2023 | Steve Moran | Initial Draft |
| 0.2 | 05-24-2023 | Sam Bridegroom | Power BI Activity with Service Principal |
| 0.3 | 06-01-2023 | Steve Moran | Snowflake and AWS sections |

[Document History i](#_Toc136505188)

[1. Summary Overview 1](#_Toc136505189)

[2. Components 1](#_Toc136505190)

[2.1 AWS and Azure Configurations 1](#_Toc136505191)

[2.1.1 Service Principal Setup 1](#_Toc136505192)

[2.1.2 Lambda Function 2](#_Toc136505193)

[2.1.3 Secrets Manager 3](#_Toc136505194)

[2.1.4 S3 Bucket 3](#_Toc136505195)

[2.1.5 Simple Notification Service (SNS) 3](#_Toc136505196)

[2.2 Snowflake Components 4](#_Toc136505197)

[Appendix A: References 4](#_Toc136505198)

[Appendix B - Key Terms 5](#_Toc136505199)

# Summary Overview

There is currently a lack of capabilities that exist in the Power BI native administrator UI to do proper analysis of dashboard/report usage in the platform. PowerShell provides a cmdlet called Get-PowerBIActivityEvent to extract the information in JSON format. This data is then made available within an Amazon S3 bucket for ingestion into Snowflake via Snowpipe.

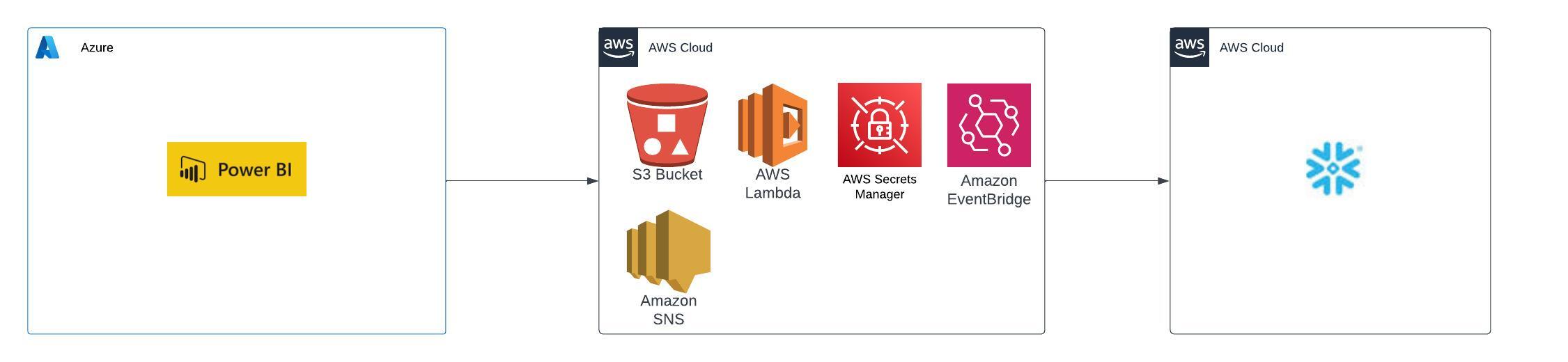


Figure 1 – Architecture Overview

# Components

There are several different components that are required for this implementation. These span Azure, AWS, and Snowflake.

|  |  |
| --- | --- |
| Service | Description |
| Power BI | Interactive data visualization tool by Microsoft |
| Snowflake | Data warehousing cloud |
| AWS | Amazon Web Services |
| Snowpipe | Snowflake service Integration tool |
| PowerShell | Command-line tool |
| Secrets Manager | AWS service for storing secrets |
| Lambda | AWS service for executing code (PowerShell) |
| EventBridge | AWS service for scheduling cron jobs |
| S3 | AWS service for file storage |
| SNS | Simple Notification Service |

## AWS and Azure Configurations

Several of the components have configurations that are required for the process to be fully automated. A service principal must be configured to allow the Lambda function to call the Get-PowerBIActivity cmdlet. Once the service principal is setup a PowerShell script will call the cmdlet and put the results into a S3 bucket for the Snowpipe integration to retrieve the data.

### Service Principal Setup

Step 1 – Azure Active Directory > App Registrations > New Registration:

• Name: PowerShell Power BI Activity

• Supported Account Types: Accounts in this organizational directory only

Step 2 – Redirect URI

Once the Registration is created – go to Redirect URI’s on the Overview screen and add a new URI:

• Platform: Web

• URI: https://login.microsoftonline.com/common/oauth2/nativeclient

Step 3 – Generate Client Secret

This secret is used as the password for the Account/Service Principal.

• Certificates and Secrets > New Client Secret

• Provide a description and set an expiration (Set at 1 year and will need to renewed)

When the secret is added, a screen containing the Secret ID and Value will be displayed. Copy the Value (via clipboard) to a secure location; upon leaving the screen, the value will be hashed and unavailable for extraction (requiring creation of a new secret).

Step 4 – Grant Permissions

Create the access for the account:

• API Permissions > Add a Permission > Power BI Service

• Choose Delegated Permissions (Service Principal won’t work with Application Permissions option)

• Add/Select the following:

o Tenant.Read. All

o Tenant.ReadWrite. All

o Report.Read.All

• Grant Admin Consent for these actions

Type of access must be Delegated.

Once the Service Principal is set up, add it to the AAD security group Admin-Power BI. This group is used in the Power BI Admin Portal to permit use of administrative APIs.

### Lambda Function

By default, a PowerShell script needs a custom runtime environment deployed to Lamdba. This allows the code to be written within the Lambda function using the code editor. Please see this [AWS blog](https://aws.amazon.com/blogs/compute/introducing-the-powershell-custom-runtime-for-aws-lambda/) for more information on how to deploy a custom runtime. The pre-requisites to deploy are AWS SAM, the Windows Subsystem for Linux (WSL) and PowerShell for Windows. The machine doing the deployment to AWS will require outbound internet access within PowerShell. Layers will also need to be added to allow the modules to be loaded. There are three that are required. The PowerShell runtime layer, the Power BI modules, and AWS tools layer. Here’s a list of the required modules that need to be uploaded as a zip file. The directory structure of the zip file must be /modules with the modules in each of their subfolders. This directory structure is necessary for the Lambda function to be able to use the modules at runtime.

|  |  |
| --- | --- |
| Power BI Administration Layer |  |
|  | MicrosoftPowerBIMgmt |
|  | MicrosoftPowerBIMgmt.Admin |
|  | MicrosoftPowerBIMgmt.Capacities |
|  | MicrosoftPowerBIMgmt.Data |
|  | MicrosoftPowerBIMgmt.Profile |
|  | MicrosoftPowerBIMgmt.Reports |
|  | MicrosoftPowerBIMgmt.Workspaces |
| PowerShell and AWS |  |
|  | AWS.Tools.All |
|  | AWS.Tools.Common |
|  | AWS.Tools.S3EventBridge |
|  | AWS.Tools.S3 |
|  | AWS.Tools.SecretsManager |

The tools can be saved to a local directory using the Save-Module cmdlet. The modules can then be zipped up in a /modules subdirectory for uploading to Lamdba.

### Secrets Manager

For the Power BI cmdlet to run the script must first login to the tenant using the Connect-PowerBIServiceAccount cmdlet. The required secrets are tenantID, appID, and the password. The Get-SECSecretValue cmdlet is used to retrieve the secrets. This is provided by using the AWS.Tools.SecretsManager module.

### S3 Bucket

S3 is utilized for the daily file before being ingested into Snowflake. The write-s3object cmdlet is required. The S3 bucket will store the output into a file for each day the cmdlet is ran.

### Simple Notification Service (SNS)

To trigger the Snowpipe a SNS topic needs to be in place on the S3 bucket. This allows for the integration of a file being placed in the S3 bucket fire the topic. The access policy for the Snowpipe must be configured so Snowflake can access the SNS topic.

## Snowflake Components

Snowflake needs the following components. There is a step-by-step guide within the Snowflake documentation. It can be found [here](https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration). The two components that are the most important are the SNS topic and the IAM role.

* A database and schema to hold the table.
* File Format for JSON.
* Storage Integration to S3 using an IAM role.
* A stage with the storage integration and file format created in previous steps.
* A Snowpipe using the SNS topic and stage created in previous steps.

# Appendix A: References

Table A.1 below summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Description** | **Location** |
| Get-PowerBIActivityEvent | Get-PowerBIActivityEvent | <https://learn.microsoft.com/en-us/powershell/module/microsoftpowerbimgmt.admin/get-powerbiactivityevent?view=powerbi-ps> |
| Custom Runtime for PowerShell in Lambda | Instructions on how to deploy custom runtime in Lambda. | <https://aws.amazon.com/blogs/compute/introducing-the-powershell-custom-runtime-for-aws-lambda/> |
| Save-Module | Save-Modlule | <https://learn.microsoft.com/en-us/powershell/module/powershellget/save-module?view=powershellget-2.x> |
| Write-S3Object | Write-S3Object | <https://docs.aws.amazon.com/powershell/latest/reference/items/Write-S3Object.html> |
| Snowpipe and S3 | Configuring a Snowflake Storage Integration to Access Amazon S3 | <https://docs.snowflake.com/en/user-guide/data-load-s3-config-storage-integration> |

Table A.1: References

# Appendix B - Key Terms

Table B.1 below provides definitions and explanations for terms and acronyms relevant to the content presented within this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AWS | Amazon Web Services |
| API | Application Programming Interface |
| ADR | Architectural Decision Record |

Table B.1 - Appendix B: Key Terms