model is refreshed. A balance must be found between catching all rows that have changed, and data volume, a range too big takes longer to load, and a range to small may miss late arriving data.

Detect data changes: Power BI can also be configured to only load partitions within the process window that has changed data changes. This is only possible if the data has a last changed date column, also note it that important to note that the timestamp column in SQL server does not work for this purpose.

Only refresh completed Days: The final configuration option is only to import completed periods; this is useful for reports then do not support partial periods.

Step 4: Deploy and publish report

Once the Incremental Refresh policy has been set up the next step is to publish the report. There are no particular actions required during this step; however, it should be noted that reports with an Incremental Refresh policy can only be published to a workspace backed by a premium capacity. There is little risk of saving to a non-premium workspace as the publish dialogue only lists Premium workspaces with non-premium workspaces lowlighted.

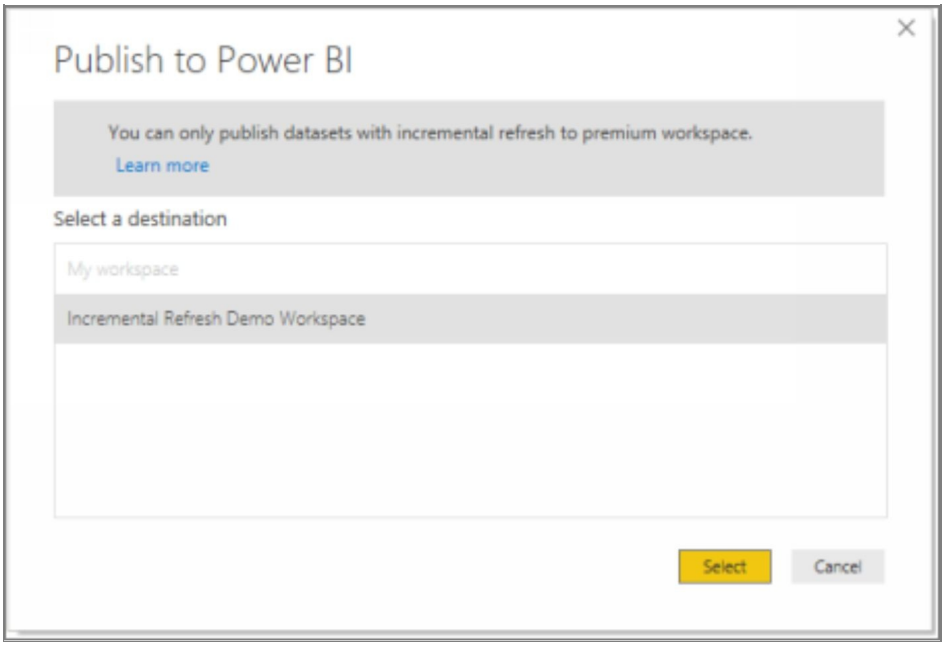


Figure 17-08: Publishing report with Incremental Refresh to a premium workspace

After the report has been published, you need to refresh the dataset as it still contains only the data from our filtering applied in the Power BI desktop, the Incremental Refresh policy has not yet been applied.

During the first data refresh Power BI removes the single partition of data created in Power BI desktop and replace it with a dynamic number of partitions required to support the Incremental Refresh policy. The first refresh takes approximately the same amount of time as a regular full load did as al the partitions need to be loaded. It is only on subsequent refreshes that we see the load time improve

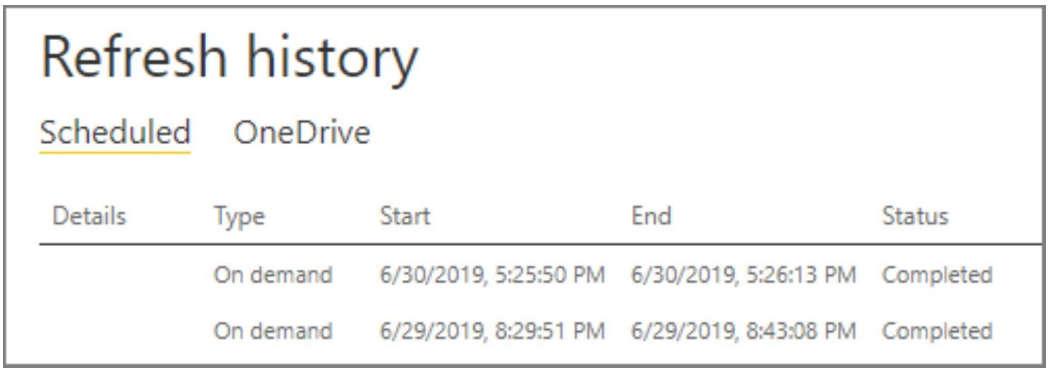


Figure 17-09: Difference in processing times of first and second refreshes.

Note: That first refresh took approximately 14 minutes while the second refresh took only 23 seconds

This is all that needs to be configured when setting up Incremental Refresh; the Power BI service manages all subsequent administration tasks such as creating new, merging existing and deleting old partitions. There is no further action required by the report creator. The remainder of this chapter explores the inner working of how Incremental Refresh is implemented.

Looking under the covers

Under the hood of Power BI lies Analysis Services and more specifically SSAS Tabular, which is the engine that is responsible for both storing data and executing queries against that data. For years Business Intelligence practitioners have used partitioning to manage data loads into and sometimes out of the model. Creating an effective partitioning strategy was time-consuming and occasionally error-prone process. In the Incremental Refresh, the Power BI implementation of partitioning all the work of creating, loading, merging and deleting partitions is handled for you keeping in line with the easy to start self-service goals of Power BI.

However, what do partitions look like under the hood, in this section we look at how Power BI implements its partitioning strategy. While it is possible to modify the policy to get different behaviours we will user the Incremental Refresh policy created in Figure 25.7 where the data retention period is set to 2 years, data refresh is set to 10 days and the other to option are not enabled.

To achieve this Power BI will create several partitions. Conceptually the simplest would be to create one partition per day. However, it becomes inefficient to have many smaller partitions. Power BI begins by creating daily partitions, when there are more than a calendar months' worth of daily partitions and those daily partitions are outside of the 10-day refresh window then those daily partitions will be merged into a single monthly partition. When there are three consecutive monthly partitions those are merged into a quarterly partition, which will then be merged into an annual or year partition when there are 4 of them all in the same year, that was certainly a mouthful so the graphic below should make this easier to understand.

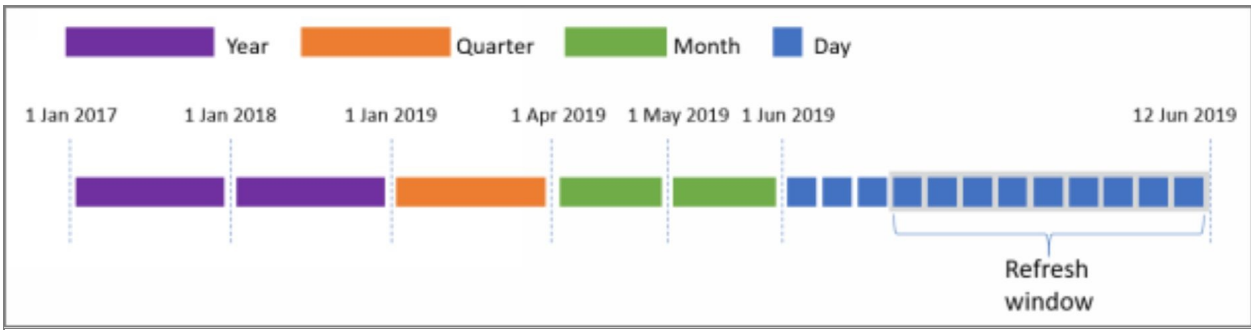


Figure 17-10: Conceptual view of the partitions created by incremental refresh

Assuming the date was 12 June 2019 then there would be 17 partitions (2 Year, 1

Quarter, 2 Month and 12 Day) this can be verified by using the new XMLA endpoint available in Power BI premium, XMLA is another set of tools in Analysis Services that allow us to query and eventually modify (on roadmap but not available) the structure of the Power BI dataset. The Path for the endpoint is found in the Workspace setting

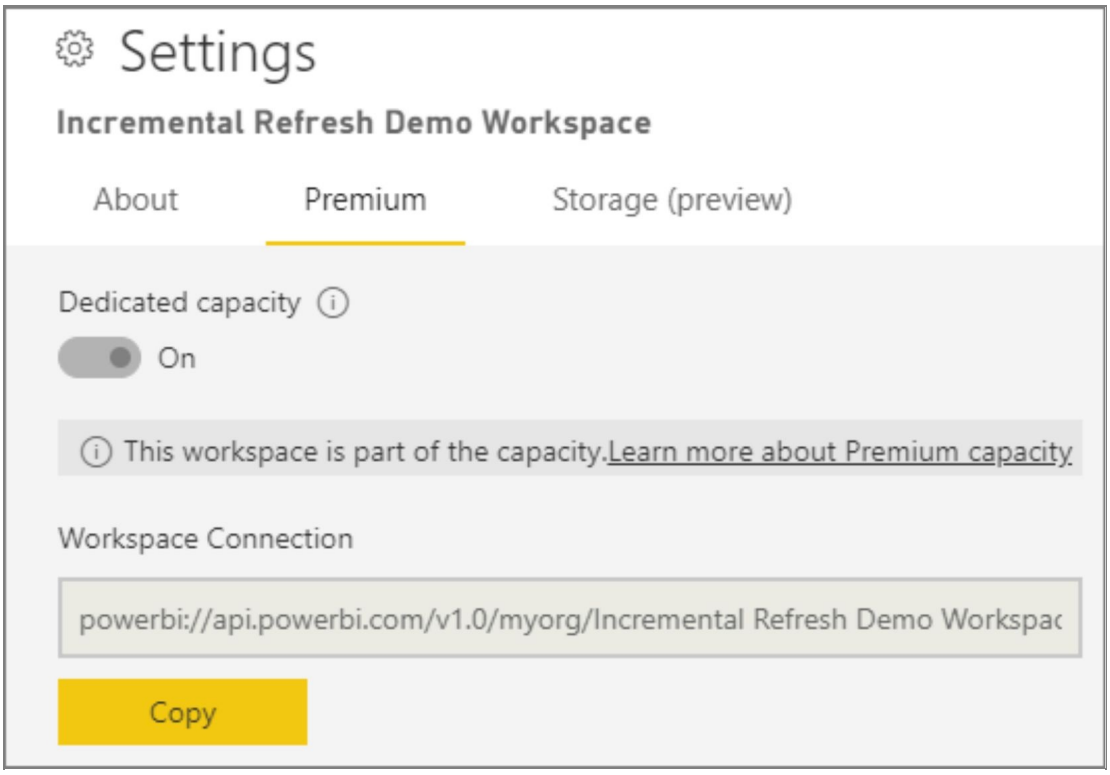


Figure 17-11: Physical view of partitions

Then using a tool such and SQL Management studio, a connection to the Power BI datasets can be made.

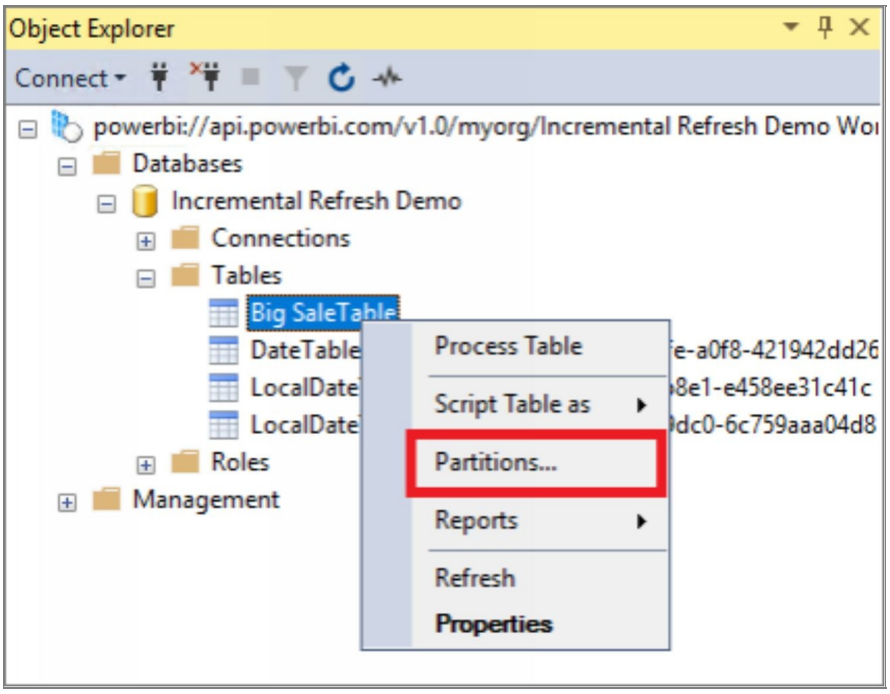
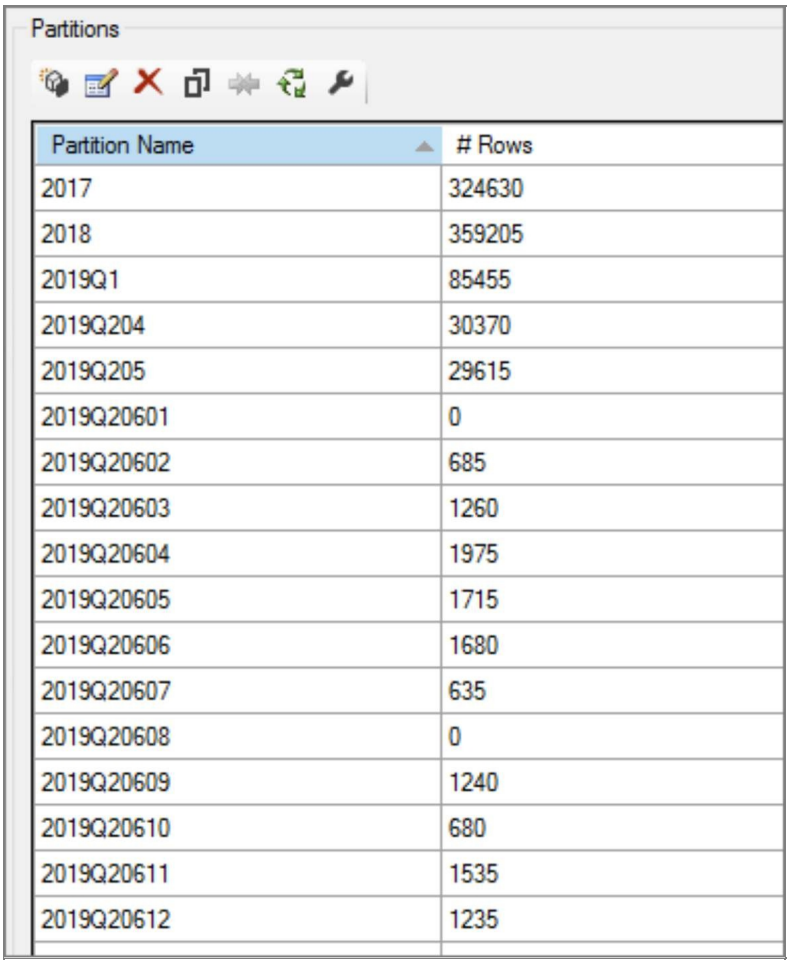


Figure 17-12: Connecting to Power BI datasets using the XMLA endpoints

When we query the partitions in the model we get, as we expected 17 partitions back



Partition Name	# Rows
2017	324630
2018	359205
2019Q1	85455
2019Q204	30370
2019Q205	29615
2019Q20601	0
2019Q20602	685
2019Q20603	1260
2019Q20604	1975
2019Q20605	1715
2019Q20606	1680
2019Q20607	635
2019Q20608	0
2019Q20609	1240
2019Q20610	680
2019Q20611	1535
2019Q20612	1235

Figure 17-13: Physical view of partitions

In this view, we see the 17 partitions we expected (2 Year, 1 Quarter, 2 Month and 12 Day) we also see some additional data such as the number of rows in each partition. The next thing to look at is the definition for the partitions themselves, we briefly mentioned XMLA however Power BI and SSAS Tabular (as of compatibility level 1200) now use Tabular Object Model (TOM), which is a JSON based format for representing the metadata of the data set. Scripting out the definition for the partition named ‘2019Q205’ which is May 2019 we get.

Figure 17-14: TOM for the May 2019 partition

Here we can see several useful bits of information starting with the names of the database, table and partition. We also see more detailed information about the partition itself

Type: This tells us what the Incremental refresh policy is being used, only Policy Range is available at this time, other options include M for tables that are derived from Power Query and Calculated for calculated tables created with DAX.

Start & End: The values for start and end correspond the first (Included) and last (not included) values in the partition, these map to the **StartRange** and **EndRange** parameters that we created in Step 1.

Granularity: This denotes the partition type and can be year, quarter, month or day.

The final task that we will look at is to examine the query that is sent to the data source, to get an idea of what the query will look like we can use the View Native Query option in the applied steps in Power Query, this shows us what the effect of query folding will look like

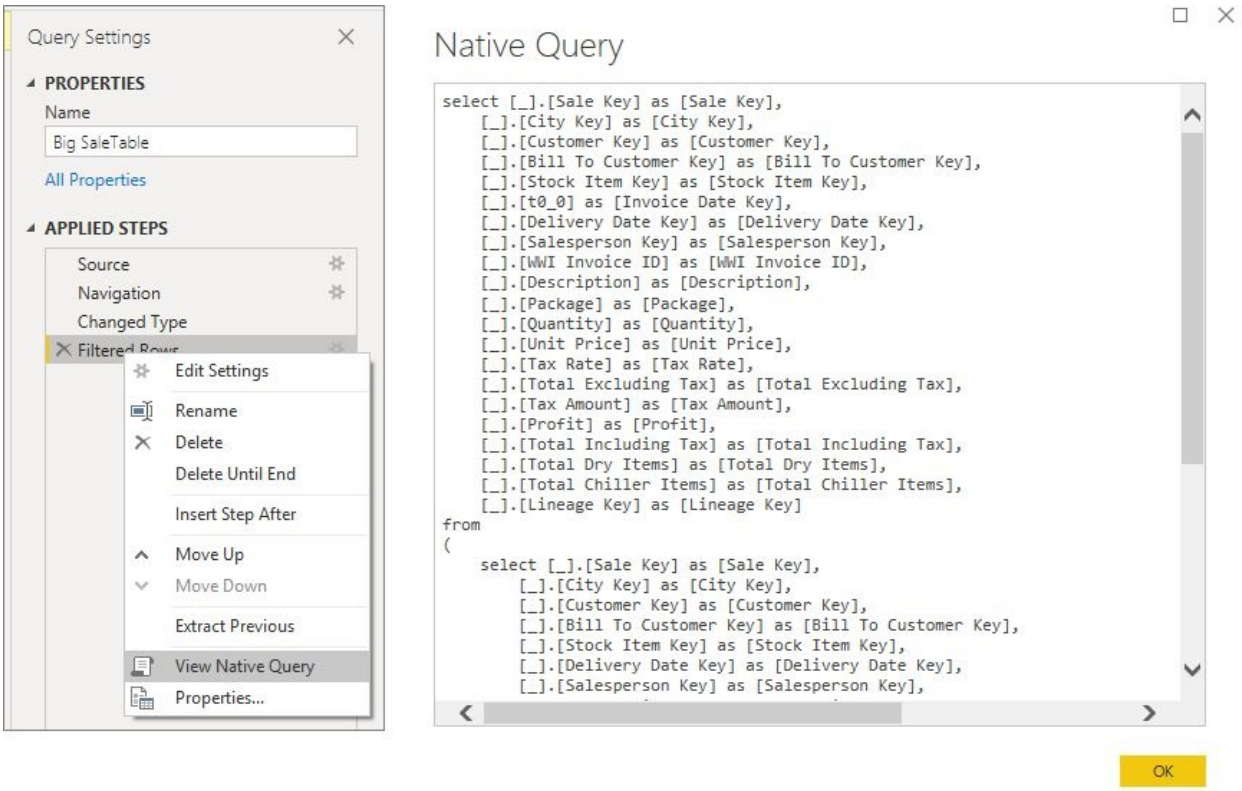


Figure 17-15: Query generated by power Query after Query folding

An alternative approach or one to validate this approach is to use extended events, or SQL profiler capture all incoming queries, below is the query generated.

```
execute sp_executesql N'
    select [_].[Sale Key] as [Sale Key],
        [_].[City Key] as [City Key],
        [_].[Customer Key] as [Customer Key],
        [_].[Bill To Customer Key] as [Bill To Customer Key],
        [_].[Stock Item Key] as [Stock Item Key],
        [_].[t0_0] as [Invoice Date Key],
        [_].[Delivery Date Key] as [Delivery Date Key],
        [_].[Salesperson Key] as [Salesperson Key],
        [_].[WWI Invoice ID] as [WWI Invoice ID],
        [_].[Description] as [Description],
        [_].[Package] as [Package],
        [_].[Quantity] as [Quantity],
        [_].[Unit Price] as [Unit Price],
        [_].[Tax Rate] as [Tax Rate],
        [_].[Total Excluding Tax] as [Total Excluding Tax],
        [_].[Tax Amount] as [Tax Amount],
        [_].[Profit] as [Profit],
        [_].[Total Including Tax] as [Total Including Tax],
        [_].[Total Dry Items] as [Total Dry Items],
        [_].[Total Chiller Items] as [Total Chiller Items],
        [_].[Lineage Key] as [Lineage Key]
    from
    (
        select [_].[Sale Key] as [Sale Key],
            [_].[City Key] as [City Key],
            [_].[Customer Key] as [Customer Key],
            [_].[Bill To Customer Key] as [Bill To Customer Key],
            [_].[Stock Item Key] as [Stock Item Key],
            [_].[Delivery Date Key] as [Delivery Date Key],
            [_].[Salesperson Key] as [Salesperson Key],
            [_].[WWI Invoice ID] as [WWI Invoice ID],
            [_].[Description] as [Description],
            [_].[Package] as [Package],
            [_].[Quantity] as [Quantity],
            [_].[Unit Price] as [Unit Price],
            [_].[Tax Rate] as [Tax Rate],
            [_].[Total Excluding Tax] as [Total Excluding Tax],
            [_].[Tax Amount] as [Tax Amount],
            [_].[Profit] as [Profit],
            [_].[Total Including Tax] as [Total Including Tax],
            [_].[Total Dry Items] as [Total Dry Items],
            [_].[Total Chiller Items] as [Total Chiller Items],
            [_].[Lineage Key] as [Lineage Key],
            convert(datetime2, [_].[Invoice Date Key]) as [t0_0]
        from [Fact].[Sale_big] as [_]
    ) as [_]
    where [_].[t0_0] >= Convert(datetime2, ''2019-05-01 00:00:00'')) and [_].[t0_0] < convert(datetime2, ''2019-06-01 00:00:00'')
'
```

Figure 17-16: Query generated by power Query after Query folding

Looking at the T-SQL generated in the statement above it can be seen how the power query have used query folding to pass the filter expression to SQL server where it is more effectively implemented, and the partitions have been used to dynamically replaced the date values in each query to only return the data required.

Limitations and things to look out for

While Incremental Refresh has made this process amazingly simple there are a few things to be aware of.

Incremental refresh does not support future dated transactions and there does not seem to be an option for this. This is usually not a problem for most systems but if you do have future dated transactions then this may be an issue.

There does not seem to be a way to force a full load after the initial publish load, e.g. if we wanted to reload as a once off exercise possible due to some correction in the business process, this is not currently possible, you will need to re-import the PIDB file to the service (if changes have been made to the report through the online report editor then these changes will be lost). Remember that the PIBX cannot be downloaded to the desktop once it has been published as the desktop does not know how to handle partitions.

If you are using a rational database as a source, make sure that you have indexes that support incremental refresh, the optimal would be to have the partition date used as the cluster index.

Summary

Incremental Refresh has been well implemented and will meet the needs of most Power BI users, and its use should be encouraged in all reports that import a large number of rows. The limitation of requiring a premium workspace impedes this, and we would love to see this removed as more efficient report loads are good for both the report creator and the Power BI service as a whole

To learn more about incremental refresh you can view the official Microsoft documentation at https://docs.microsoft.com/en-us/power-bi/service-premium-incremental-refresh

About the Author

Michael Johnson is a Business Intelligence architect and Microsoft data platform MVP living in Johannesburg, South Africa. Michael has been working with data for the last 15 years and has run the local SQL Server User Group and SQL Saturday conference event for the last few years.

He enjoys showing people new tools and technologies that allow them to work more effectively with their data.

Chapter 18: Report Server Administration

Author: Shree P Khanal

This Chapter will help to bridge the gap of understandings required for those organizations that are not using cloud infrastructure for Power BI reporting. A non-cloud Power BI architecture can be planned using a Power BI feature called Power BI Report Server. Though this feature gives an added flexibility, it has to manage and administer a new service.

Power BI Report Server

Today, Power BI is much more than a cloud-based reporting service. Along with the enhancement in its services and the tremendous increase of users, many businesses now demand of having data and reporting solutions in on-premises. Therefore, Microsoft has introduced an option to fully deploy Power BI on-premises. This version of the Power BI on-premise is called the Power BI Report Server.

Power BI is much more than a cloud-based reporting technology. With the increasing demand of having data and reporting solutions on-premises by many businesses, Microsoft introduced an option to fully deploy Power BI on-premises. The version of Power BI on-premise is called Power BI Report Server.

In this chapter, you will learn everything you need to know about Power BI Report Server, a complete on-premise solution. Including, how to install and configure it as well as will highlight the pros and cons as we get along it. At the end of this chapter, you will be able to decide whether Power BI on-premise is the right choice for you or not and how can you implement and configure this feature.

What Is Power BI Report Server?

Power BI Report Server is an edition of SQL Server Reporting Services that can host Power BI reports. To install the Power BI Report Server, you don't need to have a SQL Server installation media; it comes with its setup files. Power BI Report Server can host Power BI reports as well as Reporting Services like SSRS Reports.

Along with Power BI Report Server, there will be an instance of Power BI Desktop installation. The Power BI Desktop edition which comes along with the report server should be used to create the Power BI reports else reports cannot be hosted on the report server. Power BI Report Server also regularly gets updates like Power BI Desktop giving an essence as using a Power BI desktop vversion

Power BI Report Server Can be installed on the following Microsoft operating systems

Server - Operating System: Windows Server 2012 or higher versions
Workstation - Operating System: Windows 8 or higher versions

Web browser:

- Microsoft Edge,
- Microsoft Internet Explorer 11,
- Google Chrome,
- Mozilla Firefox

***Note*:** The Power BI Report Server installation is only supported on 64-bit OS and requires .Net framework 4.6 installed in it.

For more refer to this link:

Hardware and software requirements for installing Power BI Report Server: https://docs.microsoft.com/en-us/power-bi/report-server/system-requirements

Download Power BI Report Server

Before you begin installation, download the latest edition of Power BI Report Server from following link:
https://powerbi.microsoft.com/en-us/report-server/

Download following items on the server as well as client/desktop respectively.

- Power BI Report Server and
- Power BI Desktop Report Server edition (available in X86 and X64 versions).

Installing Power BI Report Server

Installation of Power BI Report Server is simple. The user just needs to run the setup file and follow the instructions. Figure 18-01 shows the very first screen of the installation process.

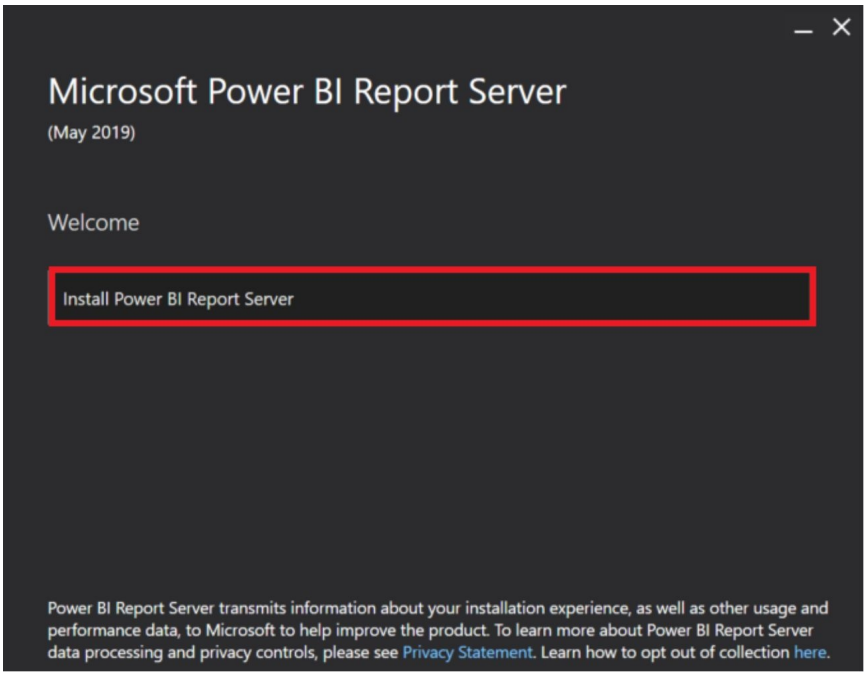


Figure 18-01: Installing Power BI Report Server

You can choose between the evaluation edition (180 days) and the licensed version of Power BI Report Server. Besides, the development edition is also available free of cost.

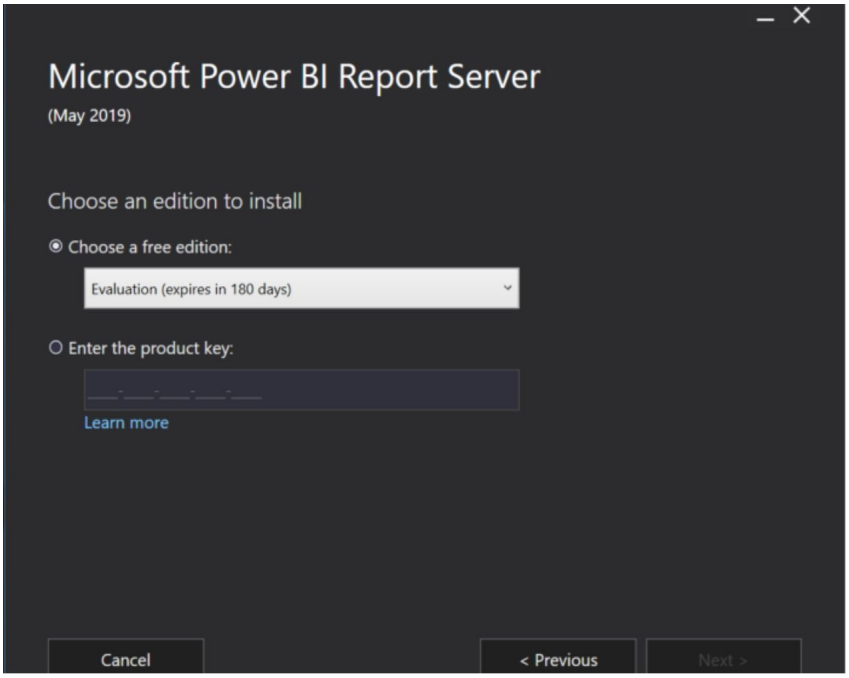


Figure 18-02: Choose the edition

Accept the license terms before moving to the next screen.

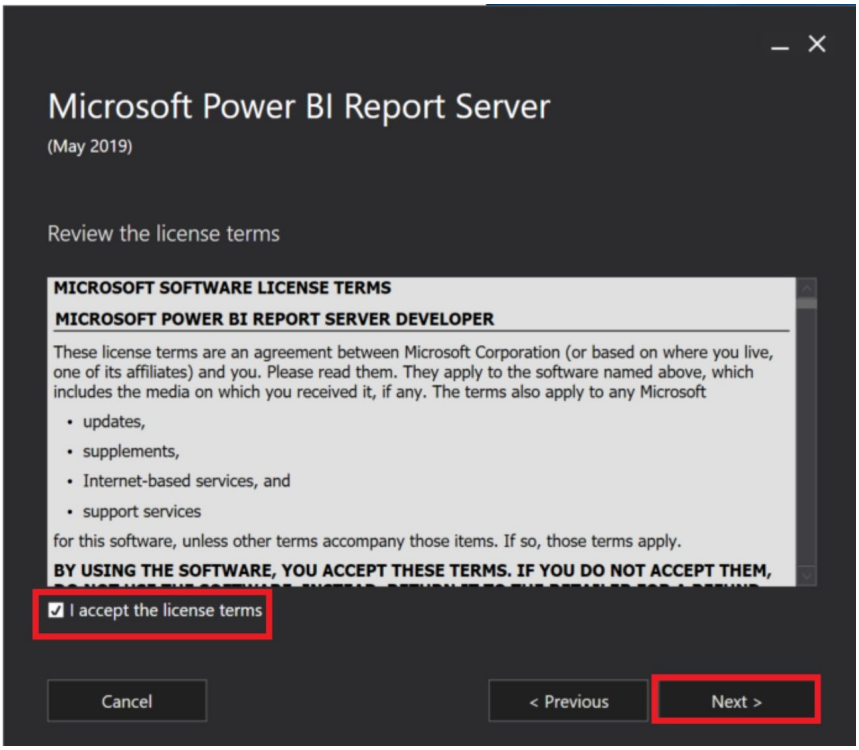


Figure 18-03: License terms for BI Report Server

Earlier, it was mentioned that you don't need to have SQL Server installed to get the Power BI Report Server. However, the SQL Server database engine is a

prerequisite for the report server to run. If you do not already have SQL Server installed, the Report Server setup process will install the database engine for you, as shown in Figure 18-04.

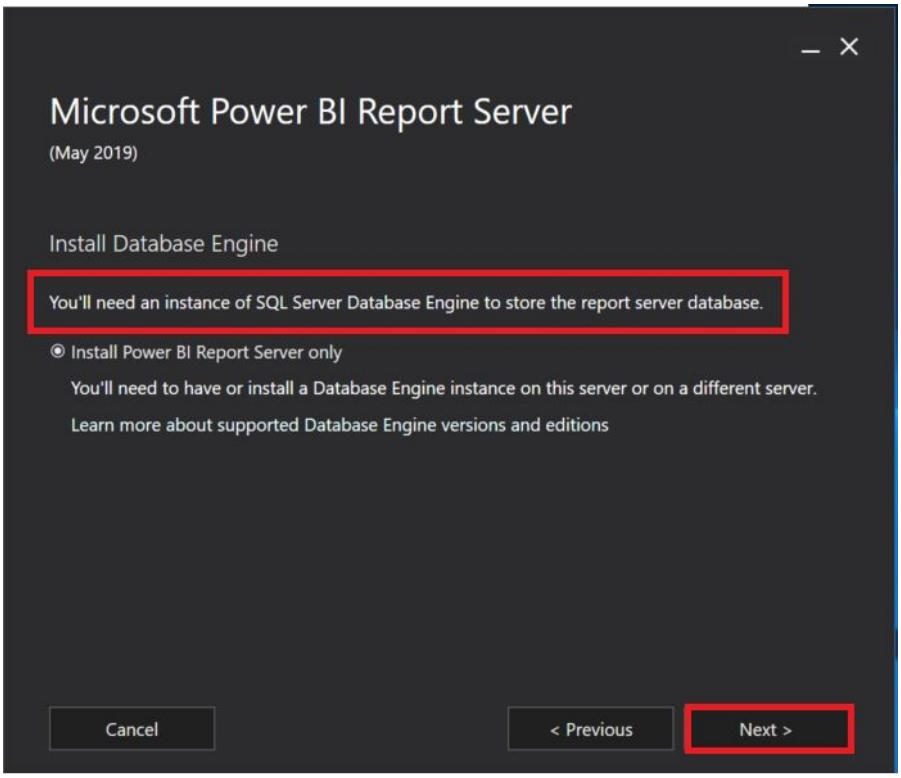


Figure 18-04: SQL Server database engine is required for the Power BI Report Server

Follow the on-screen instructions for the next steps.

Finally, restart the Report Server when the installation is successful. Upon restart, you will need to configure it.

Configuring the Power BI Report Server

After the installation is successful, open the configuration section by clicking on "Configure Report Server" button as shown in Figure 18-05. The instructions may ask you to restart the server. Once you restart, access the **Report Server Configuration Manager** from **Start | Microsoft Power BI Report Server.**

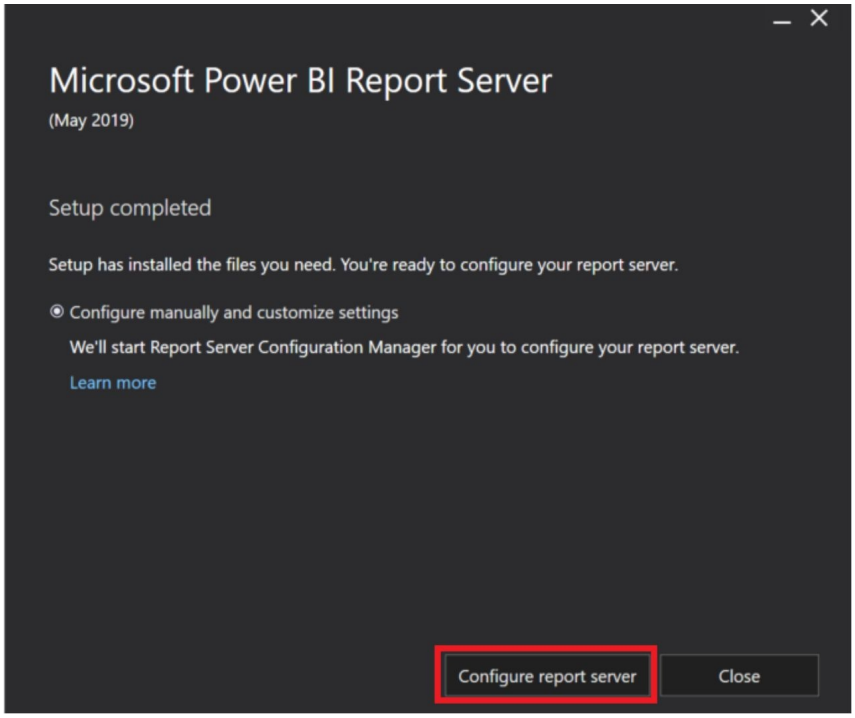


Figure 18-05: Configure report server

In the "**Connect** to a Power BI report server instance" dialogue box, make sure that your local report server instance (for example **PBIR**) is selected and click **Connect**.

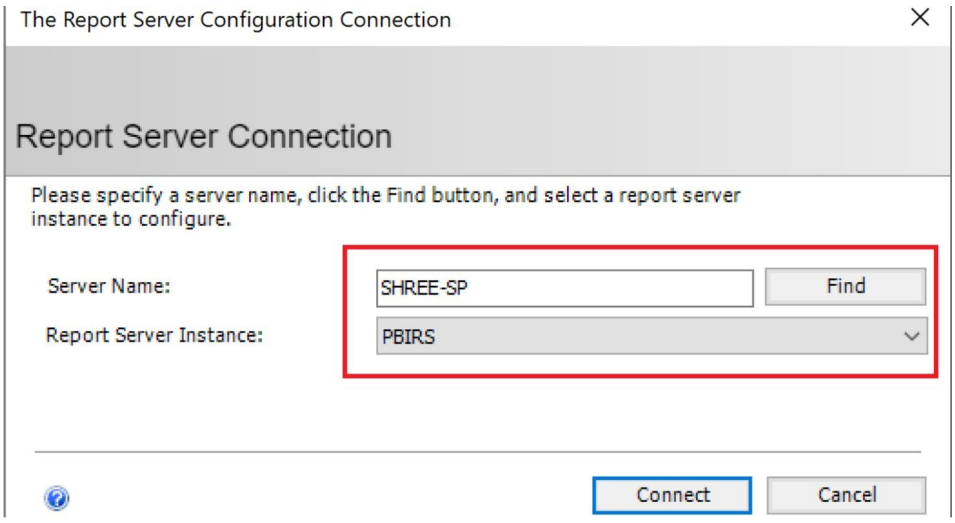


Figure 18-06: Report Server Configuration Manager connection

› setup a service account for the Report Server, access the 'Service Account' tab.

You can leave it to default or setup a new account.

Web Setup

eport Server needs Web setup for it to work. There are two URLs that you must configure. The first is for the web service and the other is for the web portal. These are accessed from their own tabs.

Web Service Setup:

To configure the web service, click on the **Web Service URL** ( ) tab, make sure that the **Virtual Directory** is set to **ReportServer** by default and the **TCP Port** is set to **80.** Alternatively, you can use your Report Server Name and a different port number.

Note: The virtual directory name identifies which application receives the request. Because an IP address and port can be shared by multiple applications, the virtual directory name specifies which application receives the request

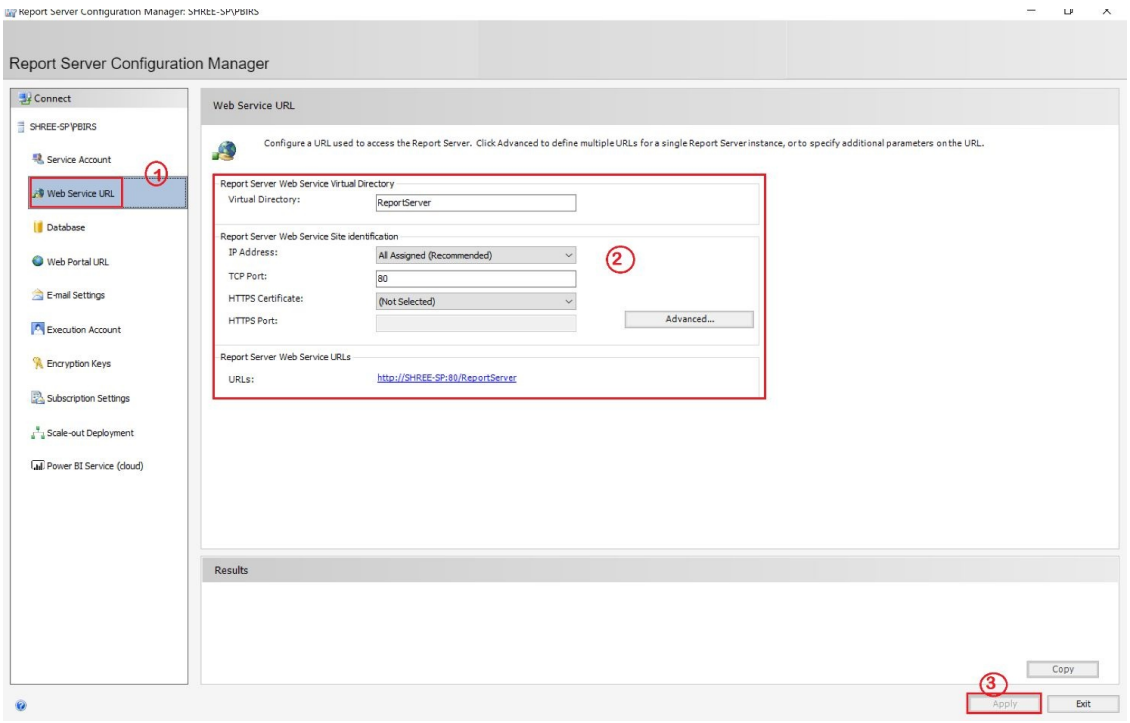


Figure 18-07: Web Service URL Configuration

Click on **'Apply'** after you have provided the necessary details. You will see success messages and a URL that you can click to open the report server's web service. For example, notice the highlighted URL in Figure 18-08.

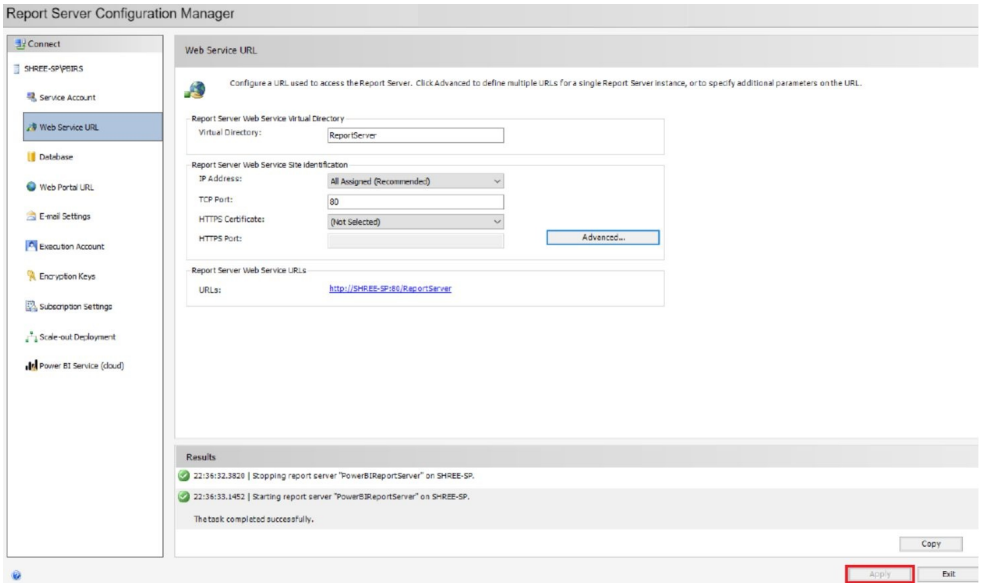


Figure 18-08: Web Service URL configuration

When you click on the URL, you should be able to open the web service's page without any issues.

Figure 18-9 shows an example of what that service page will look like.

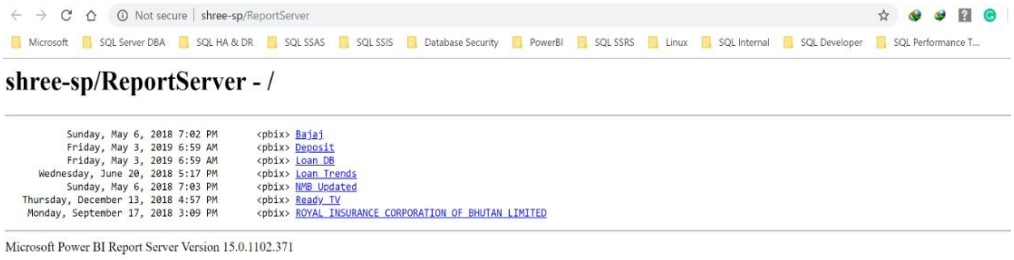


Figure 18-09: Web Service URL browsed in a web browser

In the Report Server page, initially, you will only see the version of the Report Server along with its name. However, later when you upload Power BI files, you'll be able to see the contents.

Database Setup:

On the **Database** tab( ), make sure that **the SQL Server Name** points to your local SQL Server instance & **Report Server Mode** is set to **Native**.

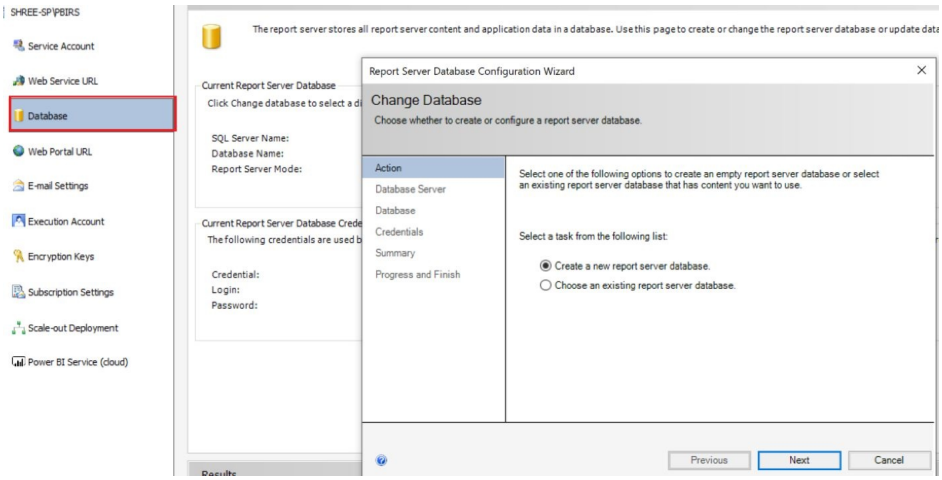


Figure 18-10: Database Configuration

To setup, follow these steps:

Click **Create Database** to open the Report Server Database Configuration Wizard.

Later, you can update the database setting by clicking on the **Change Database** button. On the screen that follows, provide new configuration values for Server Name, Authentication Type, Username and Password as shown in Figure 18-11.

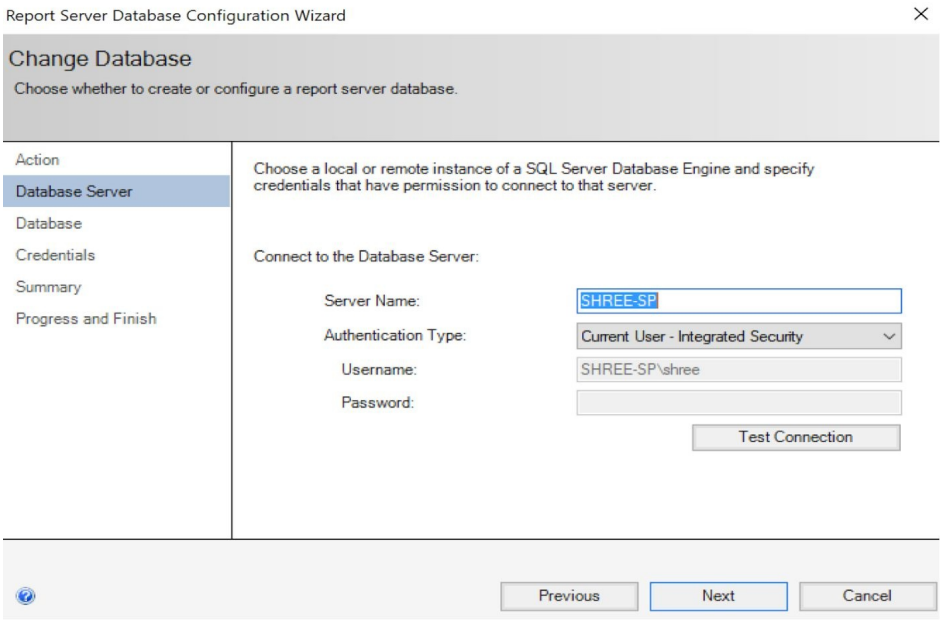


Figure 18-11: Database Server Configuration

In the **Database** page, type **ReportServer** as the default **Database Name,** select **Native Mode** for the **Report Server Mode,** then click **Next to continue.**

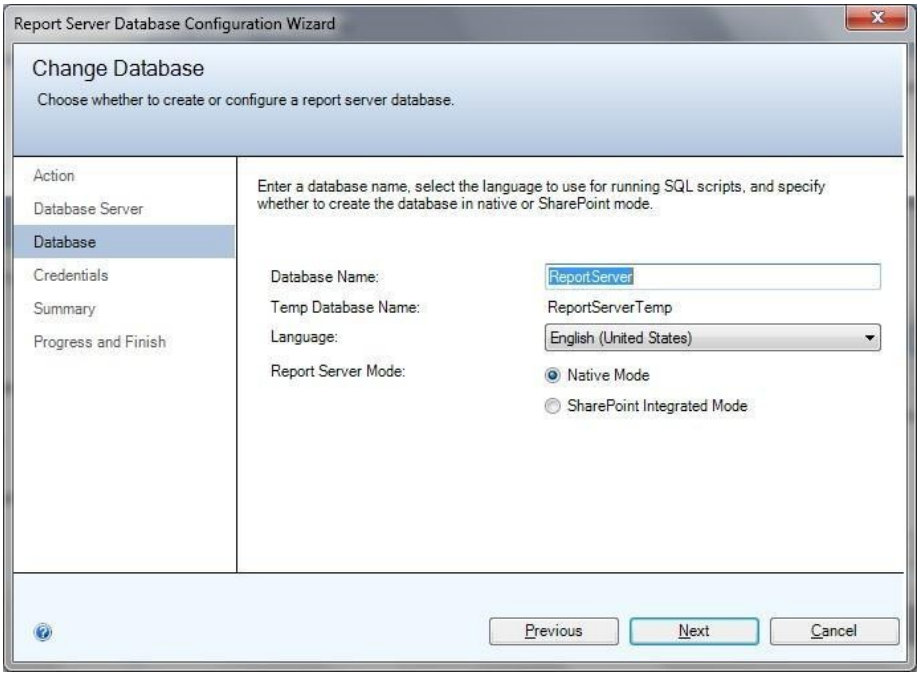


Figure 18-12: Database Configuration

In the **Credentials** page, select **Service Credentials** for the **Authentication Type,** then Click **Next** and finally **Finish** to complete the wizard.

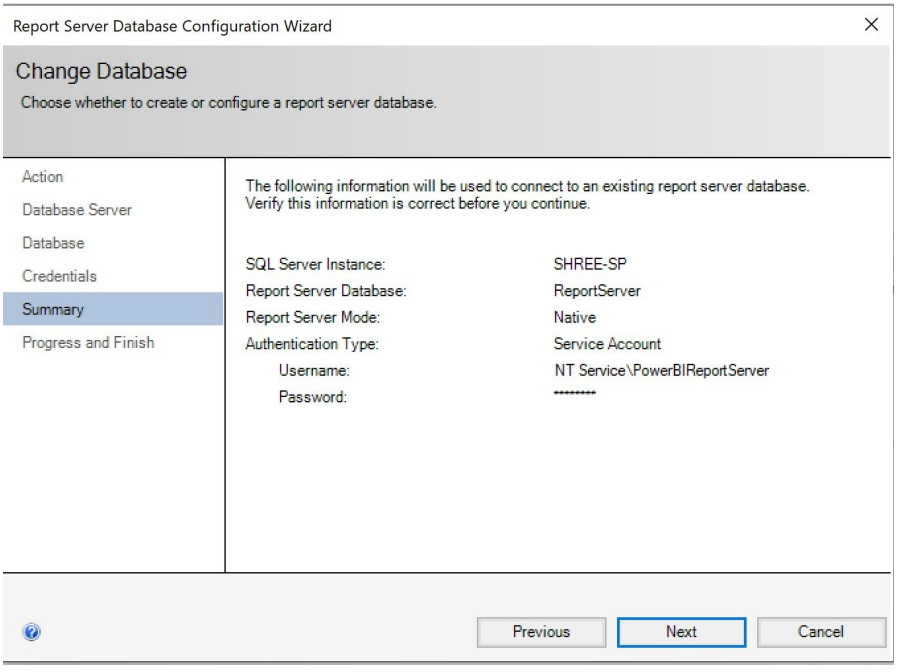


Figure 18-13: Database Configuration steps

Web Portal Setup

To set up the web portal, click on Web Portal URL tab. You can then configure the service as needed using the interface in Figure 18-14. When you're done, click on **Apply**. If the configuration process completes successfully, you will see a success message as shown in Figure 18-15. You can then click on the Web Portal URL to open it in a browser window, as in Figure 18-14.

Figure 18-14: Web Portal URL configuration

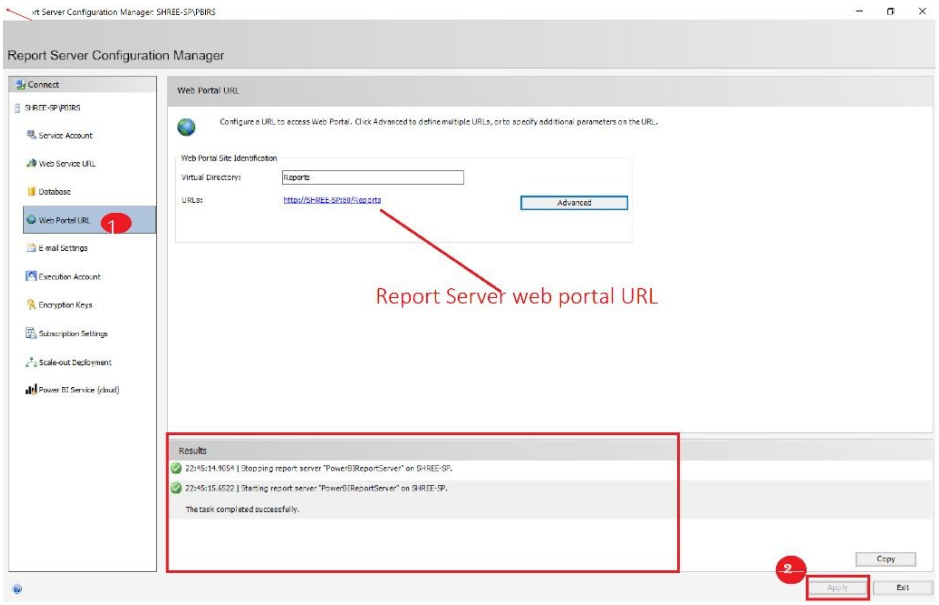


Figure 18-15: Web portal created

The web portal should show you the environment of Power BI Report Server's admin view.

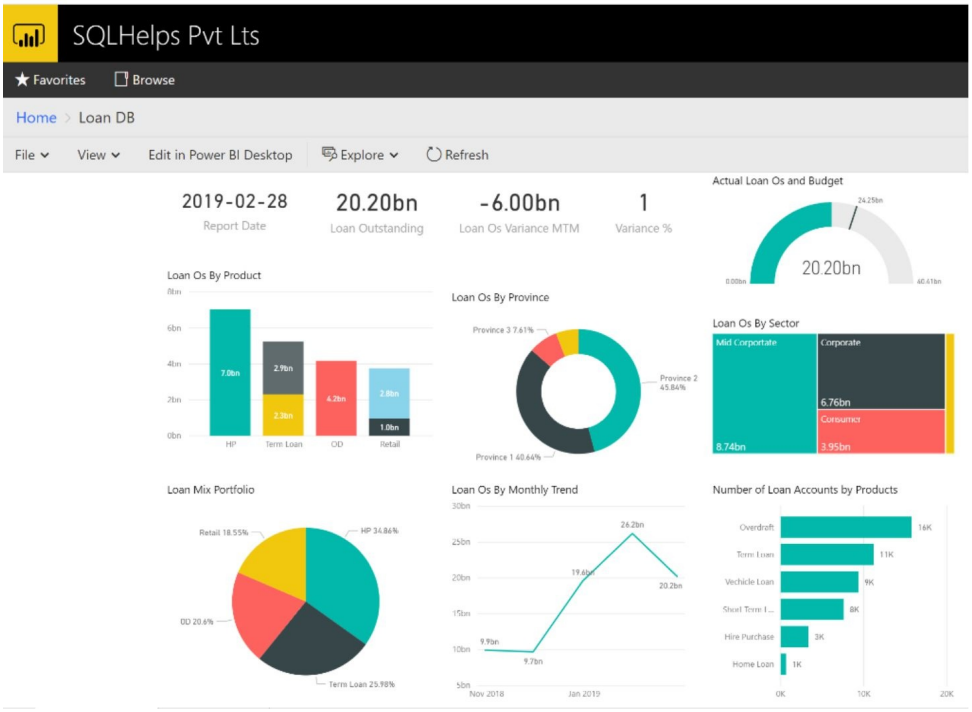


Figure 18-16: Power BI Report Server Web Portal

When installation and configuration of the Power BI Report Server are completed, you can close the Report Server Configuration Manager.

Installing Power BI Desktop Report Server

Power BI reports are developed with a specific tool called Power BI Desktop Report Server.

Download it from https://www.microsoft.com/en-us/download/details.aspx?id=56722 then Start the installation wizard and complete the installation.

Figure 18-17: Installing Power BI Desktop for the Report Server

After a successful installation, you can open the Power BI Desktop Report Server which appears as in Figure 18-18. This interface looks similar to the regular Power BI Desktop interface.

Figure 18-18: Power BI Desktop Report Server

Developing Reports with Power BI Report Server

You can create a report in the Power BI Desktop Report Server by following the same steps as you would follow when creating a report in normal Power BI Desktop. You can even open a report developed with normal Power BI Desktop in the Power BI Desktop Report Server. Figure 18-19 shows a report in Power BI Desktop Report Server.

Figure 18-19: A report from Power BI Desktop opened in Power BI Report Server

To deploy reports, you need to connect to a report server as shown in Figure 18-20.

Figure 18-20: Connecting to Power BI Report Server from Power BI Desktop

Then, enter the Web Portal URL. Use the same URL that you had used when configuring the Report Server. Figure 18-21 shows such an address at the bottom of the page.

Figure 18-21: Report Server URL is needed to connect to the Power BI Report Server

After successful deployment, you will see a message with a link to the report. Figure 18-22 shows an example.

Figure 18-22: Publishing to the Power BI Report Server

Figure 18-23 shows an example of a report hosted on Power BI Report Server. This report is fully interactive and is similar to the report hosted in Microsoft's Power BI web service.

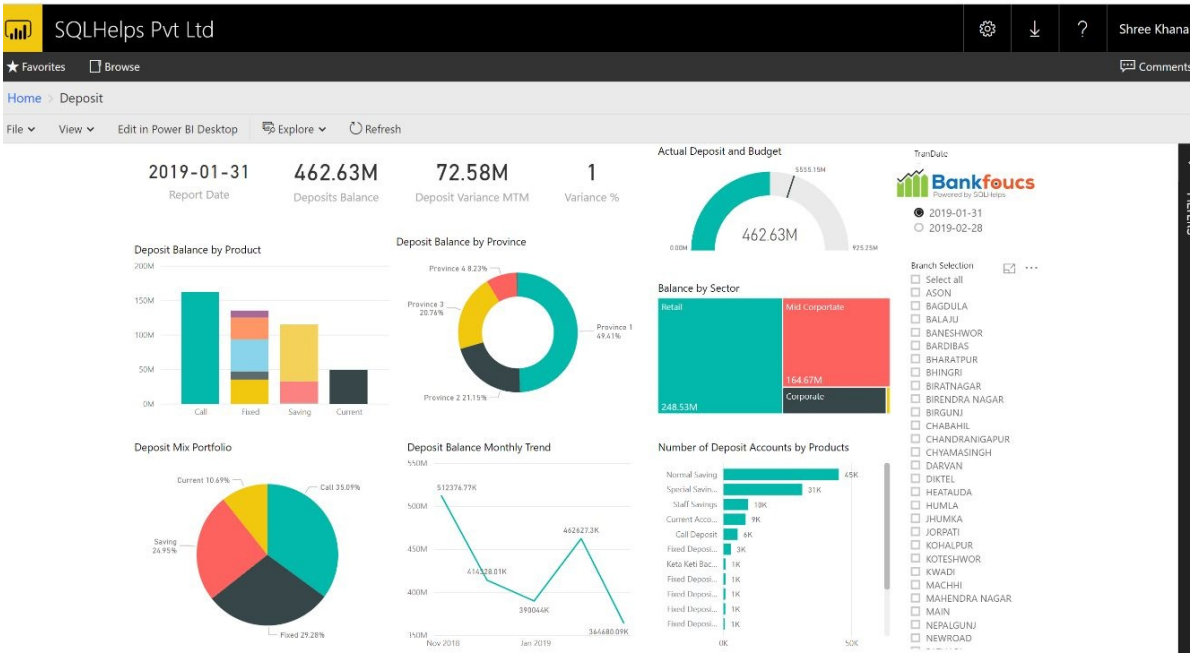


Figure 18-23: Reports are fully interactive in the Power BI Report Server

Figure 18-24 shows another way to publish a Power BI report to the report server.

You can do so by choosing the Upload option from the web portal's toolbar.

Figure 18-24: Uploading Power BI reports to the Report Server

Managing Datasets on the Report Server

A Power BI report published to the report server can be configured for scheduled data refresh. To configure, open the report server web portal, and click on the more options of the Power BI report. Figure 18-25 shows an example.

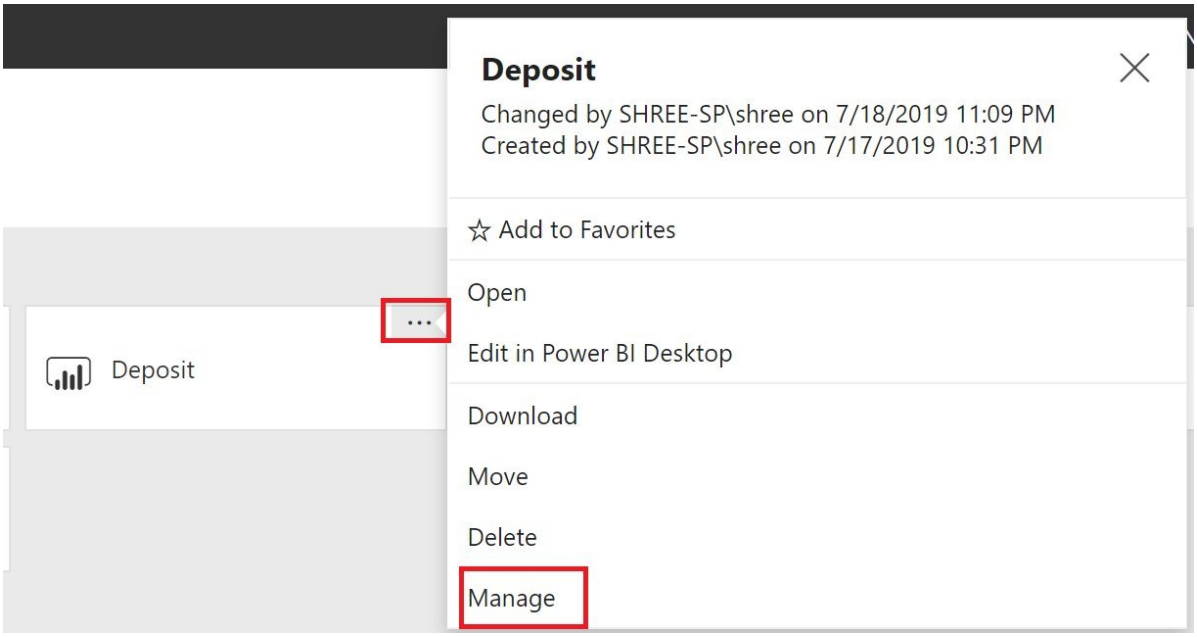


Figure 18-25: Manage the dataset of the report on the Report Server

Click on 'Manage' for data source configuration, connection to the data source, and to schedule a data refresh as required. Figure 18-26 shows the Manage tab for a report named Daily **deposit**.

Figure 18-26: Dataset properties configuration in Power BI Report Server

Schedule Refresh Requirement

If your report is sourced from a file, then you may need to schedule the report to refresh. At first, you need to specify file location from a network, as shown in the Figure 18-27.

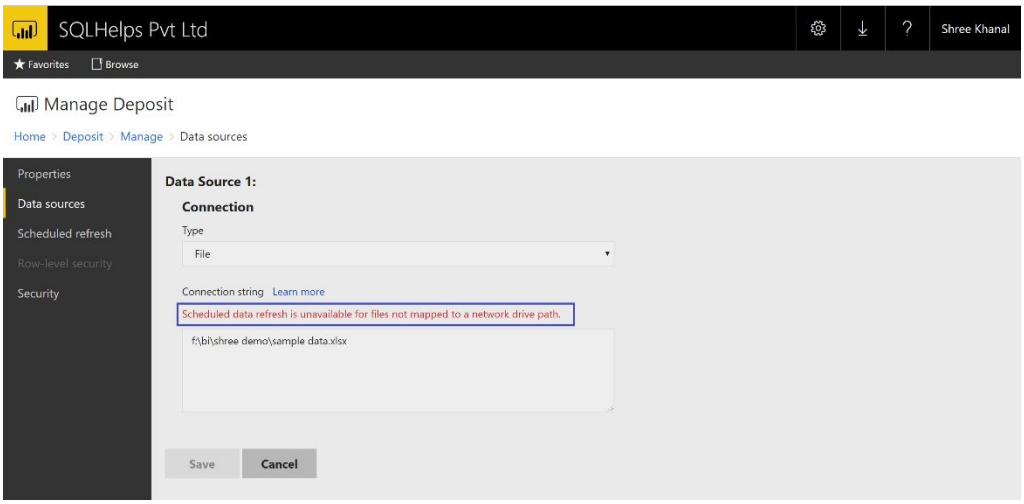


Figure 18-27: Scheduled data refresh is only available for files with a shared network drive path

If you use a network shared path to access the source file, you can set up the connection to the file as shown in Figure 18-28.

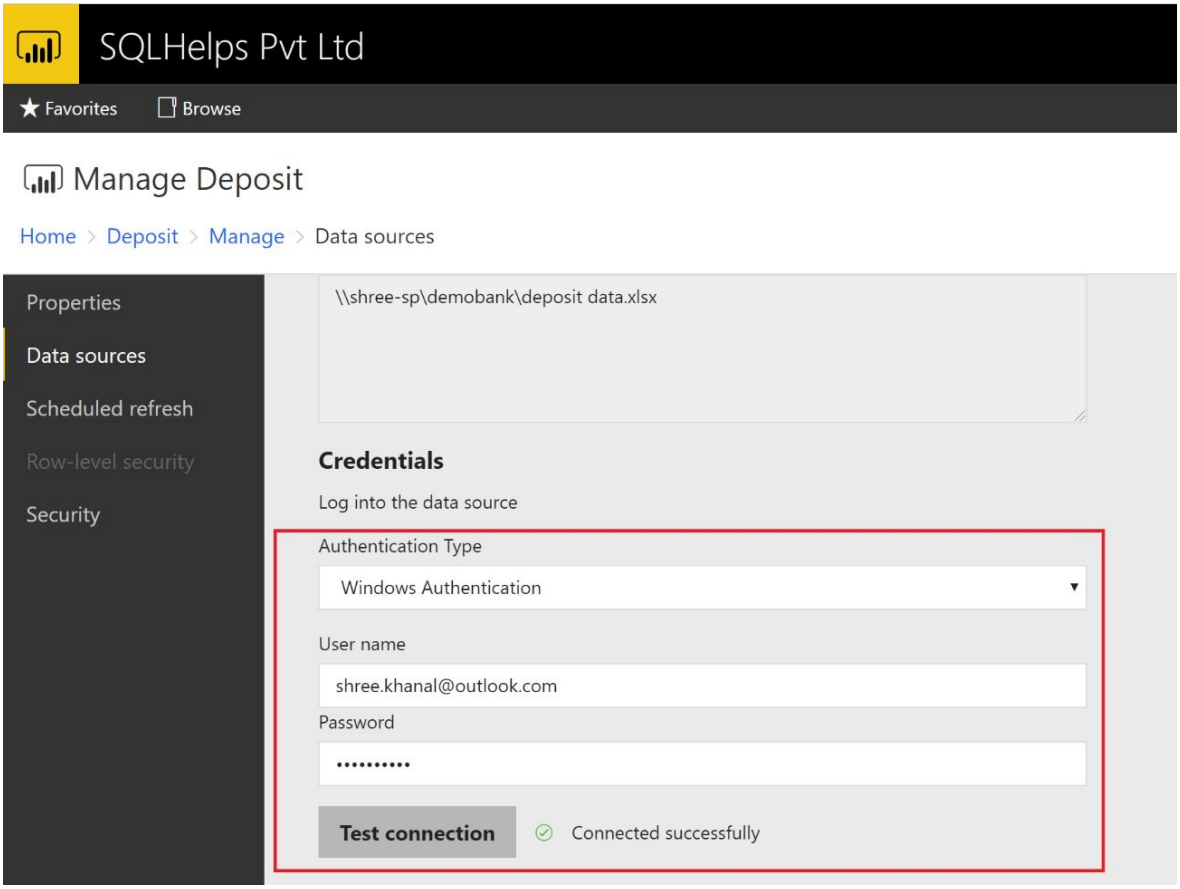


Figure 18-28: Setting up credentials for the data source connection and testing the connection

Make sure to save after this step. Otherwise, you won't be able to schedule the refresh process. Then click on the Scheduled refresh option on the left sidebar to create a refresh plan.

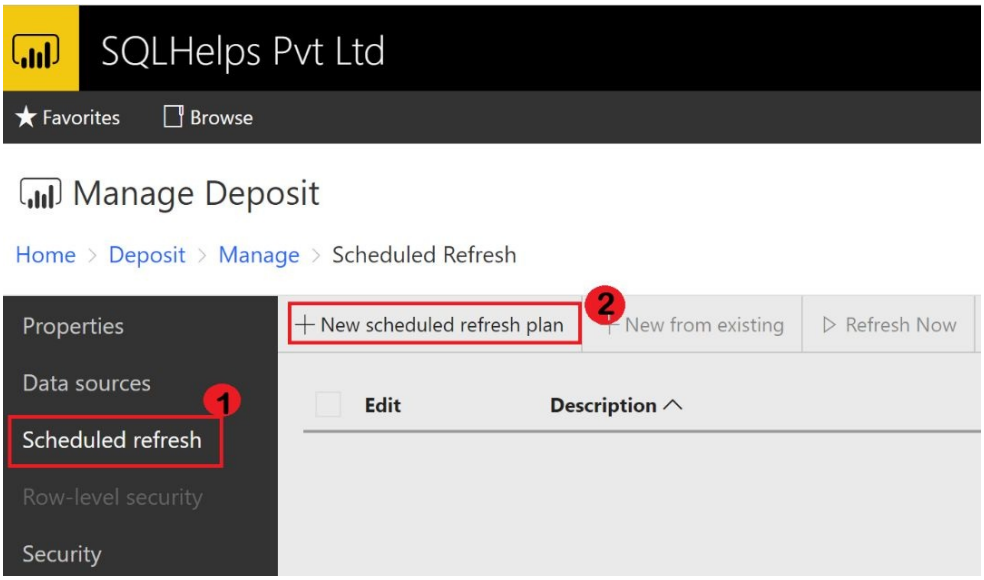


Figure 18-29: Creating a scheduled refresh plan for the dataset

The scheduled refresh configuration of the report server has more options than the Power BI Service. You can choose to schedule hourly, daily, weekly, monthly, or any custom period. Additionally, you can choose the start and end dates and many other configurations. Figure 18-30 shows some of the scheduling options that are available to you.

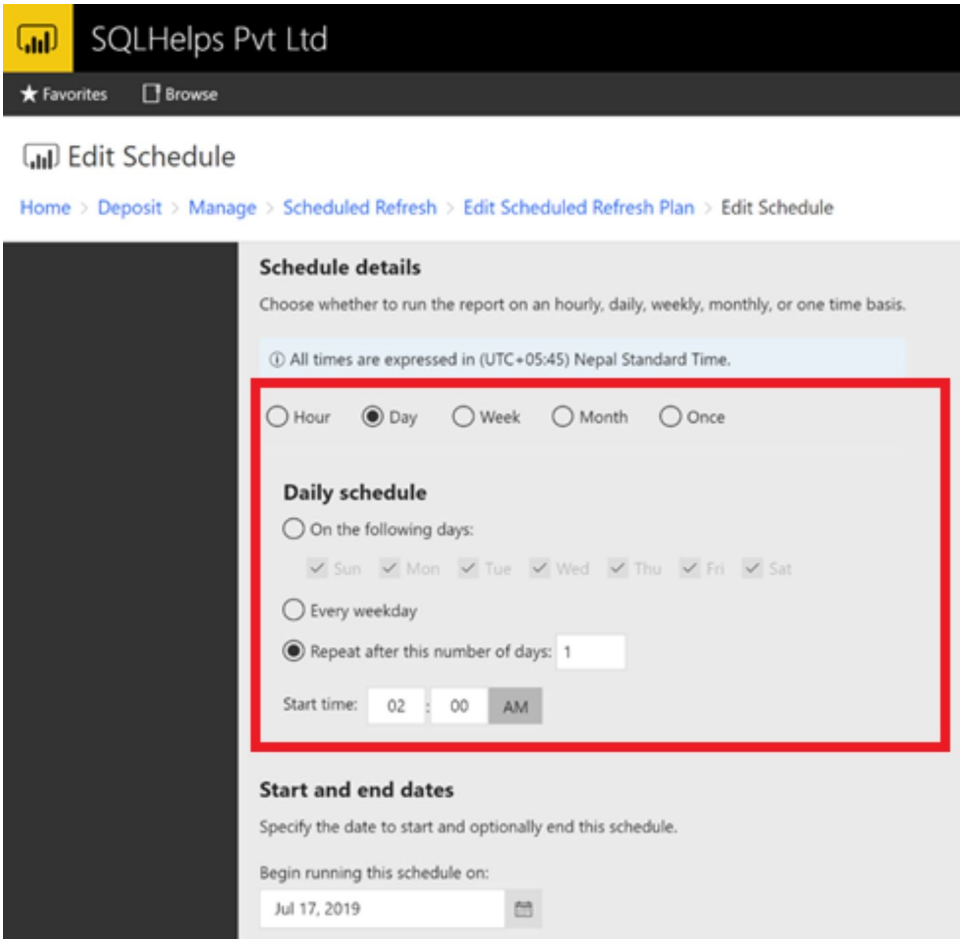


Figure 18-30: More detailed scheduling options are available with Power BI Report Server

In the Scheduled Refresh section, you can see a list of configurations along with their status.
Figure 18-31 shows such a list, with just one entry named Daily Refresh.

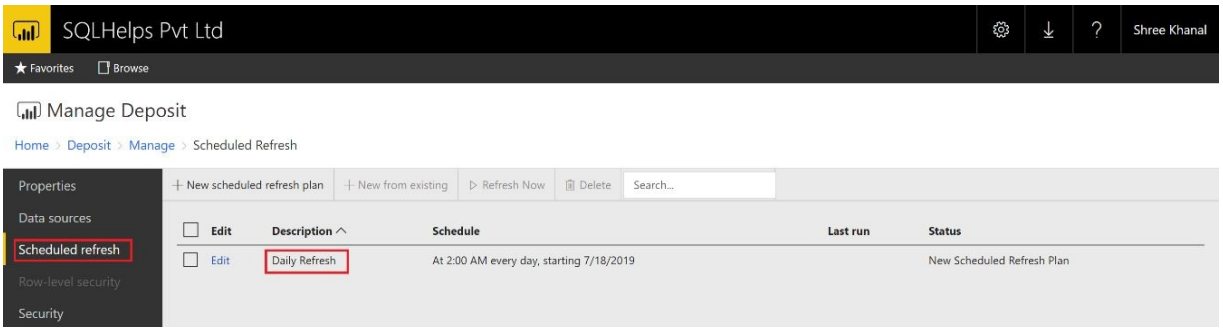


Figure 18-31: Monitoring scheduled refresh plans

Resources/ References

https://powerbi.microsoft.com/en-us/documentation/powerbi-desktop-use-directquery/

https://docs.microsoft.com/en-us/power-bi/report-server/admin-handbook-overview

https://radacad.com/power-bi-report-server-power-bi-in-on-premises-world

Summary

The Power BI Report Server is an on-premises reporting solution covering all aspects of report development, deployment and hosting. With the Power BI Report Server, interactive reports are now available on on-premises servers, not just on Power BI service.

Power BI Report Server needs a specific licensing, which comes either with the Power BI Premium or through the SQL Server Enterprise License with Software Assurance.

Power BI Report Server is a viable option for companies who are not yet ready to move to cloud-based technologies.

About the Author

Shree Khanal is a Database Architect, Speaker and Trainer having 15+ years of professional experience in database solution development and optimization. Mr. Khanal is a Data Platform MVP & Microsoft Certified Database Professional since 2000 to till date. Starting with SQL Server 7, he has been working with all the version of SQL Server up to 2017 in the production environment. In the context of OLAP cube development, he started working on it from SQL Server 2000 to latest SSAS 2017. He has worked several years providing SQL BI solutions to various enterprise clients most of them from the BI sector. Being a founder, he has been leading the Himalayan SQL Server User Group (www.sqlpassnepal.org) based in Kathmandu, Nepal since 2010.

Part VII: Architecture

Chapter 19: Governance

Author: Ásgeir Gunnarsson

The Business Dictionary defines governance as “Establishment of policies, and continuous monitoring of their proper implementation, by the members of the governing body of an organization.” [1] With this in mind, this chapter, in the context of Power BI, is going to focus on 4 pillars of governance, processes, training, monitoring and roles.

Introduction

Power BI, like many other self-service BI tools suffers for its dual purpose of being self-service but also used as an enterprise BI tool. Power BI started out as a pure self-service tool but has increasingly been moving to be more of an enterprise tool and can rightly be called a hybrid BI tool. No matter if you use Power BI as a self-service tool, as an enterprise BI tool or both, it's important to include governance into your implementation. Far too many organizations start using Power BI without thinking about governance and then have the problem of trying to get their users to stop doing things as they are used to and to start using process they are not used to and often feel will hinder their progress. No matter if your organization is starting its Power BI journey or has already ventured in the Power BI "Wild West", governance is an important and necessary part of any Power BI implementation.

This chapter will focus on the 4 pillars of a good governance strategy:

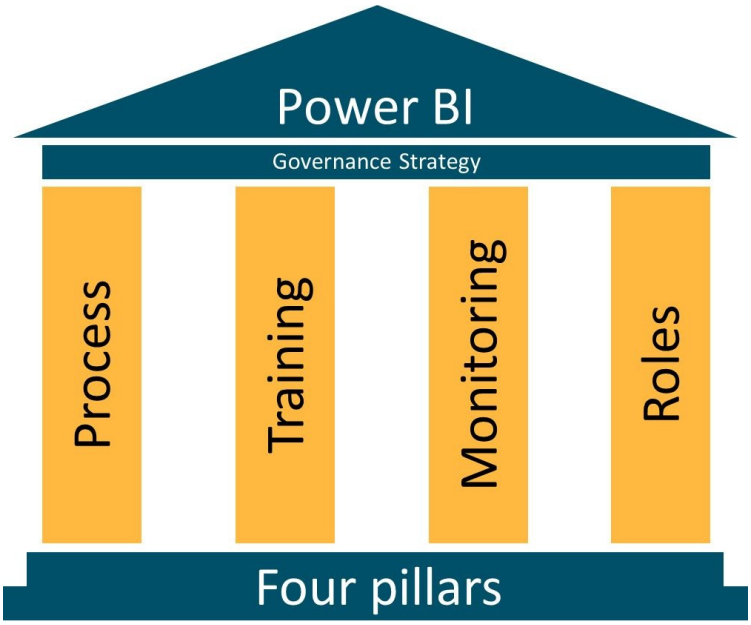


Figure 19-01: The Four Pillars of Power BI Governance Strategy

One of the following sections is devoted to each pillar.

Process

It´s important to have a formal governance process in place. This process is often broken down into smaller processes and usually contains processes for Development, Publishing, Sharing, Security, Naming standards, Support and Tenant Settings. In this section we will see examples of each process and how they can be tied together and exposed to end users.

At the heart of a governance plan are processes. There can be many smaller processes or few bigger ones but without them there is not much governance.

Most often these processes describe how to work with Power BI and sometimes they describe how to support Power BI. It´s vital that the processes are easily discoverable and are setup as a part of a whole so that users will know how each process ties into the whole governance strategy. One way is to have one master process document with links to all the process documents. Another way is to store all the process documents in the same library and categorize them so it´s easy to navigate between them and they are logically grouped.

We will look at the seven most common (in the authors opinion) processes and see examples on what they might contain.

1-Development process

A development process most often describes how a report, datasets or both are developed. They describe where you develop the Power BI content and how you store and version your files.

Developing Power BI content

It is recommended to always develop Power BI content in Power BI Desktop rather than in the Power BI Service. The main reasons for developing in Power BI Desktop are the following:

Power BI Desktop has full functionality while the Power BI Service does not have full parity. The Power BI Service has very limited data manipulation although Dataflows for Power BI do have some. It´s not possible to version the content in the Power BI Service (see next section)

Storing and versioning Power BI content

When you develop in Power BI Desktop you can store the original file in a secure place and version it. When creating reports in the Power BI Service you don´t have a source file and cannot store a master copy of it. If someone changes the report or even deletes it, you cannot go back to the original. Another

advantage of using Power BI Desktop is that you can put the file into version or source control. This enables you to revert to older versions without storing multiple copies of the report under different names. One such place that many organizations use is OneDrive for Business which has built in version control. Many BI teams have source control systems they use but they are often cumbersome for business users that don´t normally use such systems whereas OneDrive for Business is merely a folder on the computer if synchronized. Working on reports in Power BI Desktop also allows for the developer to have reports as work in progress without interfering with reports in the Power BI Service.

A development process will describe the above in detail, fitting to the needs of the organization

2-Publishing Process

The publishing process usually describes how to set up multiple environments and how to promote Power BI content between them.

Since the Power BI Service does not have built in functionality for multiple environments it´s important to describe how that should be done (if relevant to your organization). The most common way this can be achieved is through the use of multiple workspaces. You would create one workspace for each environment and move the datasets and reports between them as they go between development stages such as development, test, pre-production and production. A Publishing process normally describes how this should be achieved and how you go about promoting datasets between environments. This can be done using simple publish in Power BI Desktop or more elaborately using the Power BI REST API. The process will normally reference the Naming Standards and Security processes for the workspace naming and access control.

3-Sharing Process

Generally speaking, there are four ways of sharing Power BI report or a dashboard with an end-user:

- Share the Power BI Desktop file
- Direct sharing of reports and dashboards
- Give access to a Power BI Workspace where the report and/or dashboard reside
- Share a Power BI Workspace App

Sharing a Power BI Desktop file is not recommended as the user will have full control over the report and can change whatever they want and can access all the data if the data is imported. The user can also make their own copy of the report and change things the original author does not have any control over.

Directly sharing a Power BI report or dashboard is a method that can be used in certain circumstances. It´s not recommended unless the receivers are few or if you need to share one or few reports/dashboards out of a workspace with many reports/dashboards. The main drawbacks of direct sharing is that it is difficult to keep track of who has access and which reports/dashboards have been shared and which have not.

Giving access to a Power BI Workspace can be dangerous as the user can in most cases change or delete the report as well as access all the data in the report. With a recent change it is now possible to add users in a Viewer role in the new type of workspaces. I would still argue that only contributors should have access to a Power BI Workspace and the viewer role used for users that don´t need editing rights but will still contribute such as tester. See next section 4-Security Process for more details.

The best way to share Power BI reports and dashboards is through Power BI Workspace Apps. The Workspace Apps are made to share to a larger audience and include functionalities such as custom navigation, multiple reports/dashboards at once and the ability to share the finished report in the app while the report is being developed in the workspace. Microsoft recommends using workspace apps to share Power BI content with end users.

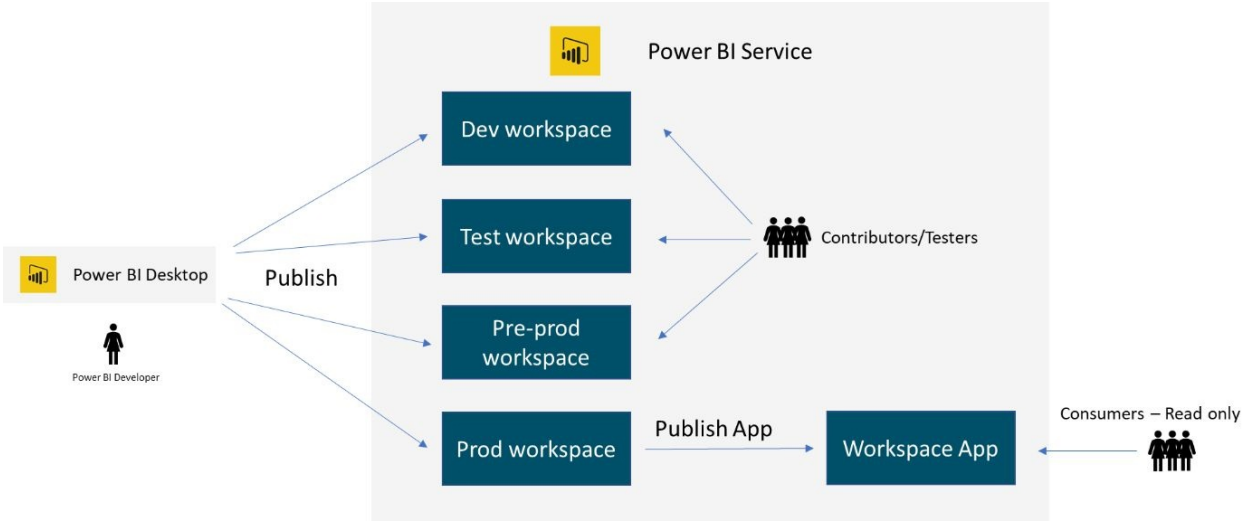


Figure 19-02: Publishing and Sharing in Power BI

The Sharing process describes how to share reports, dashboards and datasets and links to the security process for more details.

4-Security Process

A security process describes how to secure Power BI content. This is usually split into two categories: Object level security and data security.

Object level security

There are several types of objects that get created in the Power BI ecosystem and need to be secured. These objects are: Power BI Desktop files, Workspaces, Workspace apps, datasets, Power BI reports and dashboards in the service. Besides that, who can publish against certain data sources in the Power BI On-Premise Gateway needs to be decided.

It can be important not to store Power BI Desktop files where unauthorized people can get access to them. If the dataset uses import mode all the data is available to the person who has access to the Power BI Desktop file. The security process will normally point this out and advice on how to store the Power BI Desktop file.

As of when this is written everyone who has access to a workspace can see all the data, can change all reports and download a Power BI Desktop file of all reports in a Power BI Workspace. You therefore need to be careful of who get access to Power BI Workspaces. The golden rule is that only people that need to contribute to a report or dataset should have access to a workspace. In other words, if you need to WORK on the report or dataset you have access to a WORKspace. A security process will also describe if individuals, Active Directory Security groups or Active Directory Distribution List should be used to give access to workspaces.

The recommended way of sharing Power BI objects is via a Power BI Workspace App (see previous section 3-Sharing Process). Workspace apps can be considered a read only version of a workspace and as such will never give the user any modification rights. That said it´s important to give only authorized people access to workspace apps. A security process will normally describe how to give access to workspace apps as well as if individuals, Active Directory Security groups or Active Directory Distribution List should be used to give access.

Sharing reports/dashboards directly with end users is not recommended in most

cases. If you do, you can keep track of who the report/dashboard has been shared with using the Share dialog window. A security process will describe how and when you should use direct sharing and how to (manually) monitor who has access.

Data security

Sometimes it´s not enough to just control access on an object level. An example of this is when a single cost report serves the whole organization, but department managers can only see the data for their own departments. If that kind of security should be done using object level security the author would need to create as many copies of the report as there are departments. Instead it´s more efficient to secure the data instead of the objects. Securing data in that way is often referred to as row-level security. Power BI allows for row-level security where the report author can do static or dynamic row level security allowing or denying access to whole tables or the content of certain columns. When to do data security and how to implement it is often dependent on other security and privacy processes as well as rules and regulation such as GDPR.

Another aspect of data security is tightly tied to development processes as it´s best practice to connect to development versions of the source system until the report is ready and has been properly secured. The main reason for this is that development versions of source system don´t usually have real up to date data in them thus preventing the real data from getting into the wrong hands. After the author has secured the data and objects as prescribed in the security process the data source is switched to other environments ending with the production environment when ready to be released to users.

Deciding how users get added to the Power BI On-Premise Gateway is very important. The tendency is to just add all developers to the needed data source in the gateway but from a governance perspective it´s important to only add users that have been approved. Usually the process is that the data or data source owner needs to approve who can publish against the data source. This is often an excellent opportunity to engage the developer and make sure their reports/dashboards are following best practices and are approved.

A security process will describe how to secure access to objects as well as to data and will often refer to other security and process documents that exist within the organization.

5-Naming standard process

One of the most undervalued process is the Naming Standard process. Having this process early in the Power BI implementation will greatly improve the usability of the Power BI environment. Finding workspaces, reports, dashboards and datasets can be very tricky when you have hundreds of workspaces with no clear naming convention. When each project has development, test, pre-production and production workspaces (see 1-Development process) the number of workspaces can increase fast. Having a clear naming convention describing how to name workspaces as well as how to identify different environment is very important. Another common issue is that the user only sees certain amount of characters in the workspace name. Many users don´t realize this and put the environment name at the end but when browsing the workspaces in the Power BI Service you sometimes don´t see the end of the name and therefore need to hover over the workspace name to see which one is the dev or production workspace. A Naming Standard process will often be linked to a more general naming standard process within an organization.

6-Support process

Many organizations neglect to create a proper support organization when implementing Power BI (see Roles section). Having a good support process will enable your current support organization or dedicated Power BI support people to more easily assist users when needed. A support process will help non-Power BI supporters to know when to dig in and try to solve a problem and when to refer the problem to the report owner or Power BI Support people.

Typically, a support process will describe common scenarios like access requests, no data incidents, wrong data incidents and change requests to Power BI content and guide the supporter on how to react. Depending on how established your support process is, your support organization might do none, some or all of the support on Power BI content. If you have a support organization, they will get support requests on Power BI because users are used to getting support there. Therefore it´s important that you integrate your Power BI support into your current support organization even though it´s only for them to pass it on to the “real” Power BI supporters.

If your access is generally done via AD security groups or AD distribution lists a non-Power BI support organization will be able to support Power BI access requests as long as there is a separate support document for each Power BI Workspace App. This will greatly reduce the manual work by the Power BI developers or administrators and often reduce the time to get the request

resolved.

Your supports should also have a process section on how to handle dataset refresh failures and a way to redirect support requests to the Power BI Application owner or data owner if there is a data issue.

7-Tenant Settings process

There are several settings in the Power BI admin portal that are important when it comes to governance. Publish to web, Sharing outside of organization, Export data, Internal support page to name few are all very important for different reasons. It´s very important that the organization has a process in place defining how each setting in the Power BI Admin portal should be set, who the setting should apply to and describe why it´s important. Unfortunately, there is no way of automatically monitoring changes in the Admin portal which makes it even more important to have the settings well documented.

Training

If you want to have a successful Power BI implementation training is very important. You want to train everyone who touches Power BI but in a different way depending on their role. You want to make sure you get to everyone and deliver the right training based on their needs. In this section we will see examples of what categories of training you can use, the content of each training and the most effective delivery method. It´s not only governance training that is important. Training users in properly using Power BI and using best practices will deliver value faster and will make report and dataset developers more compliant.

One of the things we will explore is how to automate the training offer to users by using Microsoft Flow in combination with Office 365 (who has license) and the Power BI audit log (what are they doing)

Training categories

The most common training categories are Consumer, Report Developer and Report and Dataset Developer. For each category there is a definition of who belongs to it as well as what training content is appropriate and how it should be delivered.

Consumer training

Consumers are normally defined as Power BI users than will only consume reports or dashboards created by others. They will never create their own or modify any content. This is normally the biggest group of Power BI users and is often overlooked when it comes to training. At the same time this group is the easiest to train.

The content of the consumer training is usually general training in navigating Power BI. How to log in and find content, how to open a report or dashboard, how to use the “toolbar” above a Power BI report/dashboard and the most common ways to interact with a Power BI report/dashboard such as navigating between pages, using buttons and bookmarks and using slicers and filters. The appropriate delivery method for the consumer training is videos or training manuals. Live training is not needed and normally too expensive, due to the number of people. The exception from this is if you want to train a subset of consumers on a particular report or dashboard. That is often best done with live classroom training as then training will also involve how to analyze the data.

Report Developer

A report developer is a person that will create Power BI reports on top of readymade datasets created by others. They don´t create their own datasets but will use datasets such as Power BI datasets or SQL Server Analysis Services models.

The training for this group of people will often contain topics such as Visualization, Storytelling and Publishing and sharing. Below is an example from a real training course for report developers.

1. Introduction to Power BI
    a. What is Power BI
    b. Components of Power BI
2. Data visualization
    a. Introduction to visualization best practices
    b. The canvas and properties
    c. Theming reports
    d. Basic Charts and Visuals in Power BI
    e. Working with visuals in Power BI Desktop
    f. Custom Visuals in Power BI Desktop
    g. Practice: Create report pages and theming them
3. Power BI Service
    a. Introducing workspaces
    b. Sharing options in Power BI Service
    c. Dashboard vs. Report
    d. Practice: Publish reports to My workspace and share it with a colleague
4. Advanced scenarios
    a. Bookmarks and selection pane
    b. Report page tooltip
    c. Designing for Mobile devices
5. Introduction to Power BI development process
    a. Where is the document
    b. Content of the document
6. Governance
    a. Where is the documentation
    b. How do I makes sure I´m compliant
    c. Best practices and common sense

The report developer training is best done in a live classroom training but can be

done via live online if required. It´s also possible to create an on-demand video course but it´s the authors opinion that the users will get the fastest start and biggest benefit from a in person live classroom training.

Report and Dataset developers

Report and Dataset developers are the most advanced group of users that will both create their own datasets and reports.

The training for this group of users will often contain all the topics from the report developer course plus data topics such as Getting and cleaning data, Data modelling, Security, refreshing data and DAX. Below is an example from a real training course for report and dataset developers.

1. Introduction to Power BI
    a. What is Power BI
    b. Components of Power BI
2. Getting and cleaning data
    a. Connecting to data
    b. Direct/Live query vs. import
    c. Query Editor
    d. Transformation GUI
    e. Practice: Getting data from a database, Excel file, text file and a webpage
3. Data Modelling
    a. Why is data modelling important?
    b. Relationships in Power BI
    c. Hierarchies, formatting and hiding columns
    d. Sorting by other columns
    e. Date Table
    f. Practice: Create relationships and data modelling
4. DAX
    a. Introduction to DAX
    b. Calculated Columns and Measures
    c. Practice: Creating calculated columns and measures
5. Data visualization
    a. Introduction to visualization best practices
    b. The canvas and properties
    c. Theming reports
    d. Basic Charts and Visuals in Power BI

    e. Working with visuals in Power BI Desktop
    f. Custom Visuals in Power BI Desktop
    g. Practice: Create report pages and theming them
6. Power BI Service
    a. Introducing workspaces
    b. Sharing options in Power BI Service
    c. Dashboard vs. Report
    d. Row Level Security
    e. Schedule Refresh vs. Other types of connections
    f. Gateway's Role in the Service
    g. Practice: Publish reports to app workspace and share it as a workspace app with a colleague
7. Advanced scenarios
    a. Bookmarks and selection pane
    b. Report page tooltip
    c. Designing for Mobile devices
8. Governance
    a. Where is the documentation
    b. How do I makes sure I´m compliant
    c. Best practices and common sense

The report and dataset developer training is best done in a live classroom training It´s also possible to create an on-demand video course but it´s the authors opinion that the users will get the fastest start and biggest benefit from a in person live classroom training. In the experience of the author there is usually need for a person in the room to help people out as the dataset part of the training is often quite tricky.

Who to train and how to prioritize

One of the things the author has seen with many organizations is that they struggle to know who to train and how to prioritize when to train them. Here is a suggested method of discovering who to train and how to prioritize their training.

The first issue, who to train, can be automated in part. First of all, you can discover who has gotten a Power BI license (Pro or free) and send them an invitation to go through the consumer training, which is in the form of online videos or training manuals, so the users can do it in their own time. To discover who has a license one could build a call to the Office 365 API that has a method

which will return everyone with the type of license (Power BI) the query filters on. This needs to be stored somewhere and then the next time the call is made a comparison is made to the existing dataset and those that are not in the existing dataset are new licenses and should be invited to the training. The whole thing can be automated using Microsoft Flow or Azure Logic Apps. To decide if people need report developer or report and dataset developer training it is possible, in a similar way, to gather data from the Power BI Audit log in Office 365. The Audit log contains information about all Power BI interactions including creation of content. By looking at who is already creating content it can be used to prioritize training as those who are already creating content should get training straight away, especially on the governance.

Monitoring

One of the cornerstones of governance is monitoring. Monitoring what users are doing and monitoring what users are creating. From a governance perspective monitoring creation, access, usage, changes, deletion and data exports are the most important.

Artefact inventory

The Power BI Rest API can tell you what artefacts exists and who has access to what. Besides that, the Rest API has powerful administration endpoints that allow you to get information about various administration objects as well as allow you to perform admin tasks. To access the Power BI Rest API, you can either create your own web application and call the API or you can use PowerShell to call it. Microsoft has put some effort into wrapping many of the endpoint in the API into PowerShell cmdlets and they have also created a cmdlet to wrap the call to the Rest API. You can read more about the PowerShell cmdlets at https://docs.microsoft.com/en-us/powershell/power-bi/overview?view=powerbi-ps and you can read more about the Power BI Rest API at https://docs.microsoft.com/en-us/rest/api/power-bi/. The user will have access to some of the endpoints through normal Power BI workspace access, but a lot of the endpoints require the user to be a Power BI administrator, at least if they want tenant level information.

Monitoring usage

The Power BI Audit log can tell you who accessed what and who changed or deleted what. The audit log is part of the Office 365 Security and Compliance Centre. The Power BI audit log is turned off by default but can be turned on in the Power BI Admin Portal. To get access to the audit log in the Office 365 Security and Compliance Centre you need to have the View-Only Audit Logs or Audit Logs role in Exchange Online or be an Office 365 admin. It is possible to fetch data from the audit log in two ways. One is to log into the Office 365 audit Security and Compliance Centre, run the log query and either view the results on the screen or download the results as a CSV file. Another way is to use the Office 365 Rest API which is the preferred way if you want the automate the collection of the log information. Note that the log is only stored in the Office 365 Security and Compliance Centre for 90 days so if you want to keep it for a longer time you will need to collect it and store it in a different place such as data warehouse. More information about the audit log and how to collect the data can be found here: https://docs.microsoft.com/en-us/power-bi/service-

admin-auditing

The author recommends that both the audit log and artefact inventory is collected and stored in a database. Partly because of governance issues as described before but partly because there is valuable information in there about adoption, development over time and user behavior which could be beneficial for the organization at a later time.

Figure 19-03: Monitoring Power BI and reporting the results

At the time of this writing there are 91 events that are monitored in the Power BI audit log. If your organization does not want to store all that data you should consider taking all events that are about viewing, editing (including deleting) and exporting. When you have started the collection of the data you might want to join it to further information from the artefact inventory discussed in the previous section as well as information about the organization employees and the organization structure.

Monitoring the Power BI On-Premise Gateway

The Power BI On-Premise Gateway is a Windows service running on an on-premise server. The gateway needs to be monitored as other Windows services. The main things to monitor are the service uptime and server performance. Normally monitoring is in the hands of an infrastructure team (if one exists).

Roles

To be successful with a Power BI implantation in the long run it´s important to have well defined roles. This is most likely different from organization to organization and in some cases the same person might have more than one role. The most common roles are Power BI Administrator, Power BI Gateway Administrator, Data steward, Power BI Auditor and Power BI Supporter(s). In this section we will look at each role, it´s responsibilities and what type of person should have this role.

Power BI Administrator

The Power BI Administrator has two main responsibilities, Power BI tenant settings and capacity administration if the organization has Power BI Premium.

Power BI tenant settings, set in the Power BI Admin portal, are very important when it comes to governance. The Power BI Administrator can change all settings within the Power BI Admin portal and therefore has the power to allow for all, allow for certain groups or deny users access to certain functionality.

A Power BI administrator can also administer Power BI Premium capacities and assign workspaces to those capacities.

Far too often the Power BI administrator is the first developer that used Power BI within an organization or the “best” developer. We often refer to these administrators as accidental administrators. In some instances, this is fine, but the preferred way is to look for the best suited person that can fulfil the role within the organization. Typically, this person knows Power BI and it´s terms but has strong understanding of governance principles and the importance of the Power BI tenant settings. This is usually not a fulltime role, but some small allocation is needed if the person is to be able to do a good job.

Power BI Gateway Administrator

The Power BI Gateway Administrator is responsible for ensuring the Power BI On-Premise Gateway is running, updated and has its performance monitored. The administrator also has the encryption keys to the gateway.

The Power BI On-premise Gateway is a central component in the Power BI ecosystem and requires someone to administer it. The person might be part of an infrastructure team as the tasks are mostly about monitoring and updating which is often an integrated part of infrastructure teams.

Data steward

The data steward role is not Power BI specific but usually focused on master data. The reason it mentioned here is that it´s often a central part of a governance strategy as master data is a very important part of it. Due to it not being Power BI specific I won't get into any specifics about the role but instead leave it up to the reader to gather information about it.

Power BI Auditor

The Power BI Auditor is the one responsible for monitoring the Power BI Audit log in Office 365. The main responsibilities of the role are to make sure the audit data is gathered and stored and most importantly that Power BI users are acting in accordance with the organization's governance processes. This means reporting on top of the audit data and flagging non-compliance actions. They will often make parts of the audit log or reports on top of the audit log available to other people within the organization so they can fulfil their role. The data will often be anonymized when made available to others.

This role is often in the hands of an internal auditing function, governance team or security team.

Power BI Supporter

For a successful Power BI implementation there is a great need for someone to support Power BI developers. The main reason for this is that as Power BI is a self-service tool, developers are often business people with different backgrounds that are not necessarily used to developing integrations, data models or visualizations. Sadly, the author finds this role is often overlooked or the ones that are appointed to the role are supposed to do that AND their normal full-time job. This diminishes the success of a Power BI implementation.

The main responsibilities of the Power BI Supporter vary but often are to assist users with Power BI related problems, assure best practices are followed, champion new functionalities and generally be the go-to person for the business.

The person(s) selected for this role are often the internal super users or part of the business intelligence function in the organization. The most important thing, in the opinion of the author, is that with the role comes allocated time because without it, the quality of the organizations Power BI work might suffer.

Summary

This chapter suggest that a good Power BI governance strategy has 4 pillars, Processes, Training, Monitoring and Roles. Organizations need to define processes so that their users do Power BI right, train them to follow the processes as well as best practices when it comes to Power BI, Monitor the Power BI environment and have defined roles and responsibilities. Each pillar has equal importance and for a successful Power BI implantation you want to make sure you think about them all.

About the Author

Ásgeir is a Data Platform MVP and Chief Consultant at Datheos. He works on Business Intelligence solutions using the whole of the MS BI stack. Ásgeir has been working in BI since 2007 both as a consultant and internal employee. Before turning to BI Ásgeir worked as a technical trainer and currently teaches BI courses at the Continuing Education Department of the University of Iceland.

Ásgeir speaks regularly at events both domestic and internationally and is the group leader of the Icelandic PASS Group as well as the Icelandic Power BI user group.

Ásgeir is passionate about data and loves solving problem with BI

Chapter 20: Architecture of a Power BI Solution in an Enterprise Environment

Author: Dr Greg Low, SQL Down Under

Most of my consulting work is in large enterprises, particularly large financial organizations like superannuation (retirement fund) companies and banks. These organizations often ask me for a “big picture” of how Power BI should integrate with an enterprise architecture. Time and again, what they really are asking me for is how to do the following:

- Keep core organizational data on-premises
- Provide secure access to reports and dashboards in Power BI
- Control who sees which parts of the data

In this chapter, I describe how to do this.

The Big Picture

They say that a picture paints a thousand words, so let's start with a picture of the typical outcome that I'm looking to achieve:

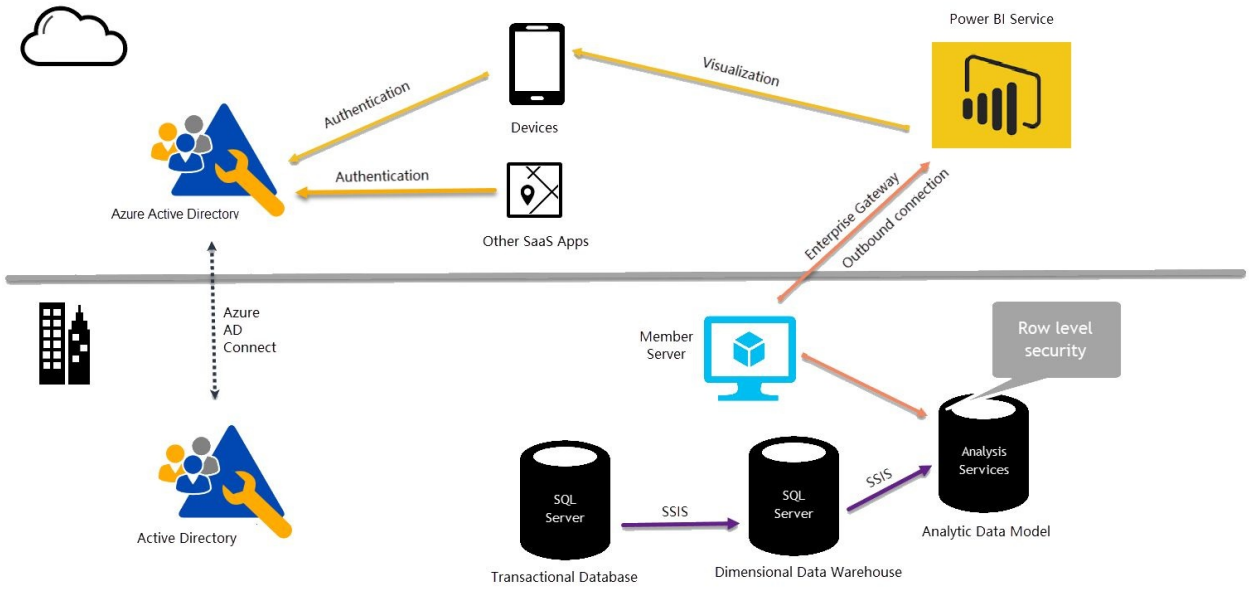


Figure 20-01: Big picture of an enterprise architecture involving Power BI

The diagram in Figure 20-01 shows the cloud-based infrastructure above the dividing gray line, and the on-premises infrastructure below that line. Let's now look at each of the key sections of this diagram separately.

Identity Management

I always tell customers that identity management is the most important aspect of planning this type of architecture. Figure 20-02 shows the core components that are involved.

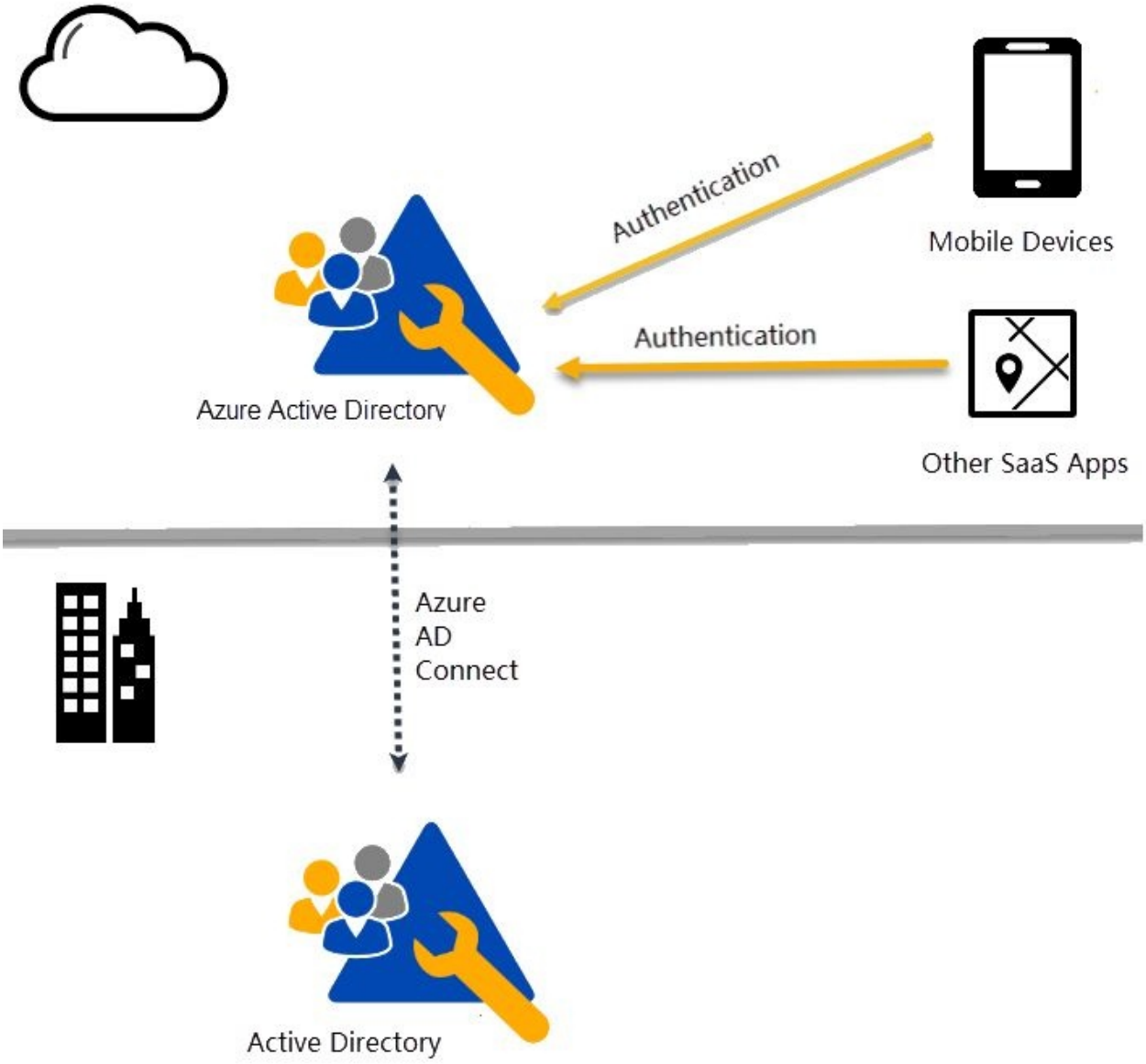


Figure 20-02: Core components for Hybrid Identity

On-Premises Active Directory

Organizations that use Windows-based PCs today, generally use Windows

Active Directory to provide identity management.

Windows Active Directory (AD) provides several services:

- ADDS - Active Directory Domain Services
- ADLDS - Active Directory Lightweight Directory Services
- ADFS - Active Directory Federation Services
- ADCS - Active Directory Certificate Services
- ADRMS - Active Directory Rights Management Services

Credential Spread Issues

Most enterprises will no doubt move into scenarios that involve cloud-based systems and require cloud-based authentication. If this process is not managed well, it quickly leads to a problem often referred to as YAUP (yet another username and password) where overall identity management becomes a significant mess.

Even if tooling like Office 365 is not on the immediate horizon, early investment in hybrid identity provision via AAD is important, so that this identity spread can be avoided right from the start. Later, Office 365 and other tools can then utilize the same hybrid identity configurations.

Hybrid Identity

Azure Active Directory (AAD) provides a cloud-based identity service that can help in avoiding these issues. Importantly, AAD can be directly integrated with the existing on-premises AD, to avoid users needing to have separate identities and passwords for the cloud services that are being used.

It is important to note that this hybrid identity can also be used in a wide variety

of other applications. I checked the marketplace gallery today and over 2800 applications are advertised as directly supporting AAD for authentication. These can be seen here:

https://azuremarketplace.microsoft.com/en-us/marketplace/apps/category/azure-active-directory-apps

As well as the standard Microsoft offerings like Office 365, SharePoint Online, Exchange Online, Lync Online, etc. all of which use AAD for identity, many organizations are adopting best-of-breed applications for different aspects of their businesses. They might use Zoom for meetings, ZenDesk for a helpdesk, or Drip for an email marketing engine, DropBox for cloud-based storage of shared documents, etc. These applications can also use this same form of hybrid identity that is provided by AAD.

Integration Applications

One case where this form of identity management is crucial is where integration applications are used. For example, a retirement fund might decide that when a new member is added to their core systems, that a new entry is made in SalesForce, and when that happens, a new email sequence is enabled in Drip, and so on. Often, applications like Zapier or Microsoft Flow are used to provide the logic behind these integrations. Once again, having a single form of identity across all these applications is highly desirable.

Azure Active Directory Offerings

AAD currently provides three main services:

- AAD (Azure Active Directory) – this provides identity services - exposing a graph API along with OAuth, SAML-P, WS-FEDERATION, and more
- AACS (Azure Access Control Services) – this federates identities from

external providers – generally on-premises AD but could also potentially include Google, Yahoo, Facebook, which could be important if member access is ever provided to resources

- AADDS (Azure Active Directory Domain Services) – this is basically a domain controller as a service. It is a scalable, high-performance, managed domain service for domain-join, LDAP, Kerberos, Windows Integrated authentication, and group policy

A major advantage of AAD is the ease of enabling options like multi-factor authentication (MFA), and cloud-based password reset. The implementation of these options alone often brings significant savings to IT support teams.

An added advantage is that member/customer/partner identities can also be stored and managed in the directory tenant.

Enhanced Security on Terminations

In terms of overall security, it is worth noting that this approach to identity also provides a single point where access can then be removed when required. If an employee leaves, and their identity is disabled in the local AD, the federation and hybrid identity means they will also no longer have access to any of these associated applications.

If separate identities had been used, disabling even a single employee can become an onerous task, and one that potentially could leave gaps open after the employee has been terminated, until they are all cleaned up.

Implementing Hybrid Identity

Azure AD Connect is the synchronization mechanism for on-premises AD and cloud-based AAD. It provides a choice of:

- Password hash synchronization (PHS)
- Pass-through authentication (PTA)
- Federation (AD FS)

My usual recommendation is to implement Federation as it provides the highest level of interoperability and ensures that not even hashed passwords are sent to the cloud.

Application Integration

It is likely that many new applications will be cloud-based. Integrating these with AAD is now straightforward.

For .NET development, developers can include the WIF (Windows Identity Framework) in their projects. The application is registered with AAD to create a Service Principal (which includes an identifier called an AppPrincipalID, and a secret).

When logon to an application starts, control is transferred to AAD for logon. AAD eventually calls the application back and passes a signed token that is then used to verify the user.

Authentication for External Users

Azure AD has two forms of authentication for external users:

- Azure AD B2B (business to business) is used to assign permissions on internal resources to external users. It is generally used with partner organizations and allows collaboration with internal users. The pricing structure for Azure AD currently allows a 1:5 ratio for internal users to B2B partner users.
- Azure AD B2C (business to consumer/customer) is used to manage authentication to corporate applications by external users. It is often used for customers or members who need access to corporate websites. These users do not need to be part of the internal domain's user list. There is no practical limit to the number of these external users that can be attached. The pricing is based upon the number of authentications performed each month. The first 50,000 authentications are currently free.

B2B allows the use of MFA (multi-factor authentication) if the Azure AD tier is one that supports it.

B2C allows the use of MFA at a very low additional cost.

Password-less Implementation

I have mentioned this topic for completeness although it might be a component of a longer-term strategy. Passwords are fast becoming considered as inappropriate for authentication. Many industry leaders consider password authentication as fundamentally broken.

The costs associated with using passwords are fast outweighing the benefits. Although many organizations have strict password policies, even the strongest passwords are easily phishable.

For enterprise IT departments, password and identity related support and maintenance is expensive. And as it becomes common for IT teams to require ever stronger password complexity and demanding more frequent password changes, the costs rise ever more quickly.

Details on approaches to password-less implementations can be found here:

https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE2KEup

On-Premises Source Data

The data that Power BI operates can come from on-premises data sources. My preference is for the following structure, as shown in Figure 20-03.

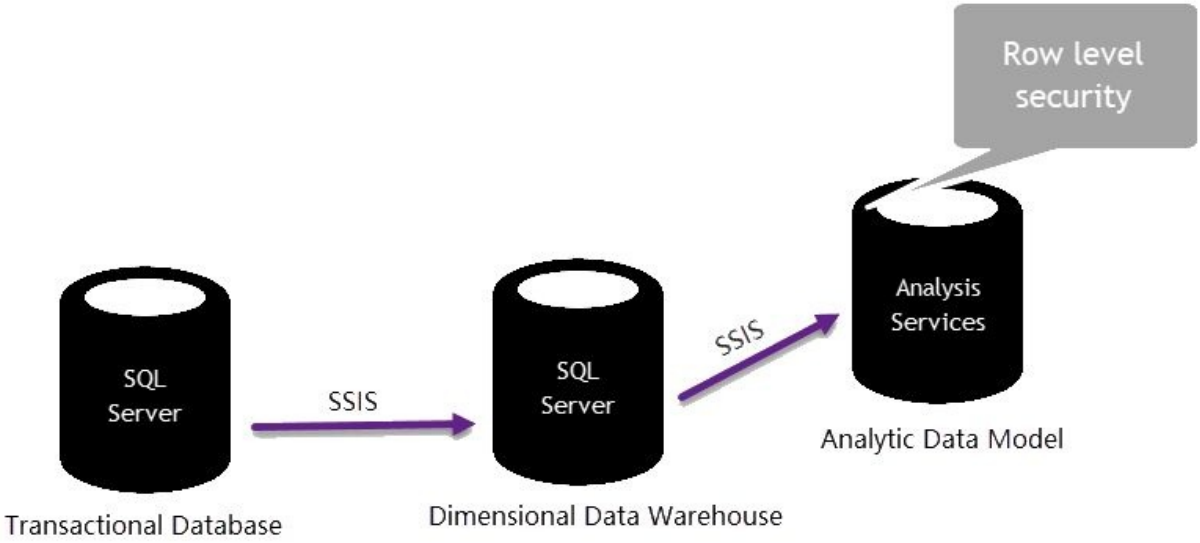


Figure 20-03: On-premises databases

Dimensional Data Models

Reporting directly from source systems (particularly for strategic reporting) is generally undesirable for several reasons such as:

- Transactional source systems are designed to support their applications, and are generally highly normalised
- The relationships between the tables are often complex and complex joins are required in queries
- Names of columns and tables are not directly usable in reports and charts
- Most source systems do not support versioning – they only show a single current state of the data
- Reporting can place a significant processing load on the source systems

Instead, we recommend the creation of a dimensional data warehouse that, while it is still a relational database in SQL Server, is structured to support analysis and reporting rather than transactional processing.

Some key advantages of this type of model are:

- Provides a single cleaned up model of the data
- Avoids the need for data cleansing in all reports and analytics by centralising it
- Has simplified (star schema based) relationships
- Directly supports both reporting and analytics
- Easy for end users to navigate when required
- Names are directly reusable in reports, charts, and dashboards
- Can support versioning if needed
- Can provide a single view of data from multiple source systems

Customers often want to build all reporting against the source system data, but this is not appropriate. It generally impacts the system too greatly; the source system data is usually not in a form that is designed for reporting and analytics; it is common to need to combine data from multiple source systems; and it is also often useful to add versioning of the data that is not provided by the source systems.

For example, if I have customers allocated to sales territories, and I rearrange the territories, what happens when I view last year's sales report? Does all the data automatically jump into the new territories? I might want that, but more commonly, I want the data to still look the same as it did last year. Versioning can help to fix that.

One of the challenges with any BI implementation though, is that if a separate dimensional data warehouse is created, latency will be involved in the transfer of source system data across to the data warehouse.

It's important to separate strategic reporting from operational reporting.

I recommend using SQL Server Reporting Services (SSRS), connected directly to the transactional database, for operational reporting that must be up to date. For reporting that is more strategic in nature, and can generally tolerate some small latency, I recommend using Power BI, Excel, and SSRS, with all of these supported by SQL Server Analysis Services (SSAS).

Strategic reporting is best delivered from an analytic data model (using SSAS), that was constructed over a dimensional data model in a separate SQL Server database. I know that you'll read information that tells you that you can just connect Power BI or other dashboarding tools, directly to your transactional databases, without worrying about creating a dimensional data model first. And while that's strictly true, it's not how you get a great outcome.

Transactional systems make poor direct data sources for analytic systems and for most dashboards. They also rarely hold versioned data. A dimensional data model can help with that.

Staging and Cleansing Data

Your reporting and analytics will need data from source systems. I recommend implementing staging databases to be used for import and cleansing of source system data that is needed for reporting and analytics. Generally, I try to achieve this while meeting the following aims:

- Minimising the impact on the source system
- Not missing any data
- Not duplicating any data
- Cleansing the data before reuse

Wherever possible, I try to import data incrementally. This is possible for most systems, but I do come across tables where this is not possible, and I need to retrieve entire tables. But I aim to do the data retrieval in a way that avoids excessive load on the source system.

With incremental loading, it is critical to avoid missing data, and duplication of

data.

Source system data generally also needs to be cleansed i.e. correcting/aligning data values, supplying missing values, correcting invalid values, fixing relationships between data elements.

I use staging databases in the dimensional data warehouse to hold the data where this work is being performed.

Analytic Data Models

SSAS allows you to use advanced mashup and modelling features to combine data from multiple data sources, define metrics, and secure your data in a single, trusted tabular semantic data model.

The data model provides an easy and fast way for users to browse large amounts of data for ad hoc analysis. Queries on the data model will often be dramatically faster than the equivalent queries run directly on the dimensional data warehouse.

Power BI, Excel, and other applications can be used as the visualisation layer for the data in SSAS.

SSAS provides the ability to create a semantic layer above your dimensional data model, by creating an analytic data model. There are two flavors of SSAS: Multi-dimensional and Tabular. At this point, I would almost always recommend that you use a tabular data model. There are a handful of exceptions, but you should just start with a tabular data model in SSAS.

Power BI can support multi-dimensional models somewhat, but it works cleanly and efficiently with tabular data models because that’s basically the same data model that it uses internally within Power BI.

As well as loading tables from your underlying dimensional database, the analytic model allows you to create:

- Relationships between the tables
- Control over query filtering directions
- Computed columns to ease navigation and simplify the data for the user
- Measures such as totals or averages over time

Importantly, an analytic data model in SSAS can also implement RLS (row level security) based on the users and groups in Active Directory. RLS lets you make sure, for example, that only New Zealand users see New Zealand data, Queensland users see Queensland data, and so on.

While it is possible to create row level security within the Power BI service, it will be tied to identities within Azure Active Directory.

Moving the data between databases

If I've convinced you that you need a dimensional database, and an analytic data model between your source systems and Power BI, you also need to be able to move data around. The tool that I typically use for this is SSIS (SQL Server Integration Services).

SQL Server Integration Services (SSIS) provides graphical tooling for the creation of enterprise-level data integration and data transformations solutions that involve SQL Server. In fact, it can be used for even broader purposes than SQL Server-related integration. It can copy or download files, load data warehouses, cleanse and mine data, and manage SQL Server objects and data.

SSIS can work with a wide variety of data sources such as XML data files, flat files, and relational data sources, and then load the data into one or more destinations. It includes a rich set of built-in tasks and transformations, graphical tools for building packages, and its own Catalog database, where you store, run, and manage packages.

Most SQL Server installations utilize SSIS for a wide variety of scheduled data movement tasks. The packages that are created with it can be scheduled by either the inbuilt SQL Server Agent, or by external schedulers such as Control-M.

Enterprises can use SSIS for controlling the cleansing of data in the staging databases, for loading data into the dimensional data model, and for processing (i.e. reloading) the analytic data model.

Paginated Reporting

Apart from Excel, SSRS is the most popular Microsoft BI tool. Since 2016, the server aspects of SSRS are shipped in two forms:

- Report Server (RS)
- Power BI Report Server (PBIRS)

Paginated reports can also now be deployed to the Power BI service.

I recommend using Reporting Services reports for both operational and strategic paginated reporting.

SSRS provides the ability to create, publish, and manage reports, then deliver them in several ways such as in a web browser, embedded within an internal application, on a mobile device, or emailed to the user. Reports can be viewed on-demand, or they can be scheduled.

The web portal can be branded to provide an organizational look and feel.

PBIRS is a superset of RS but also allows deploying some Power BI reports directly to an on-premises server. PBIRS is licensed by either SQL Server Enterprise with SA, or via Power BI Premium. Most organizations have SQL Server Enterprise Edition with SA, so it is possible to take advantage of the PBIRS based version. Generally, though, I recommend that Power BI reports are delivered from the Power BI service instead.

Providing Power BI Access to the On-Premises Data

Power BI needs access to the on-premises data sources. It does that by using the Enterprise Gateway. I recommend that enterprises implement the Microsoft Enterprise Gateway to support development of applications such as Power BI, Power Apps, Microsoft Flow, and others. The gateway is shown in Figure 20-04.

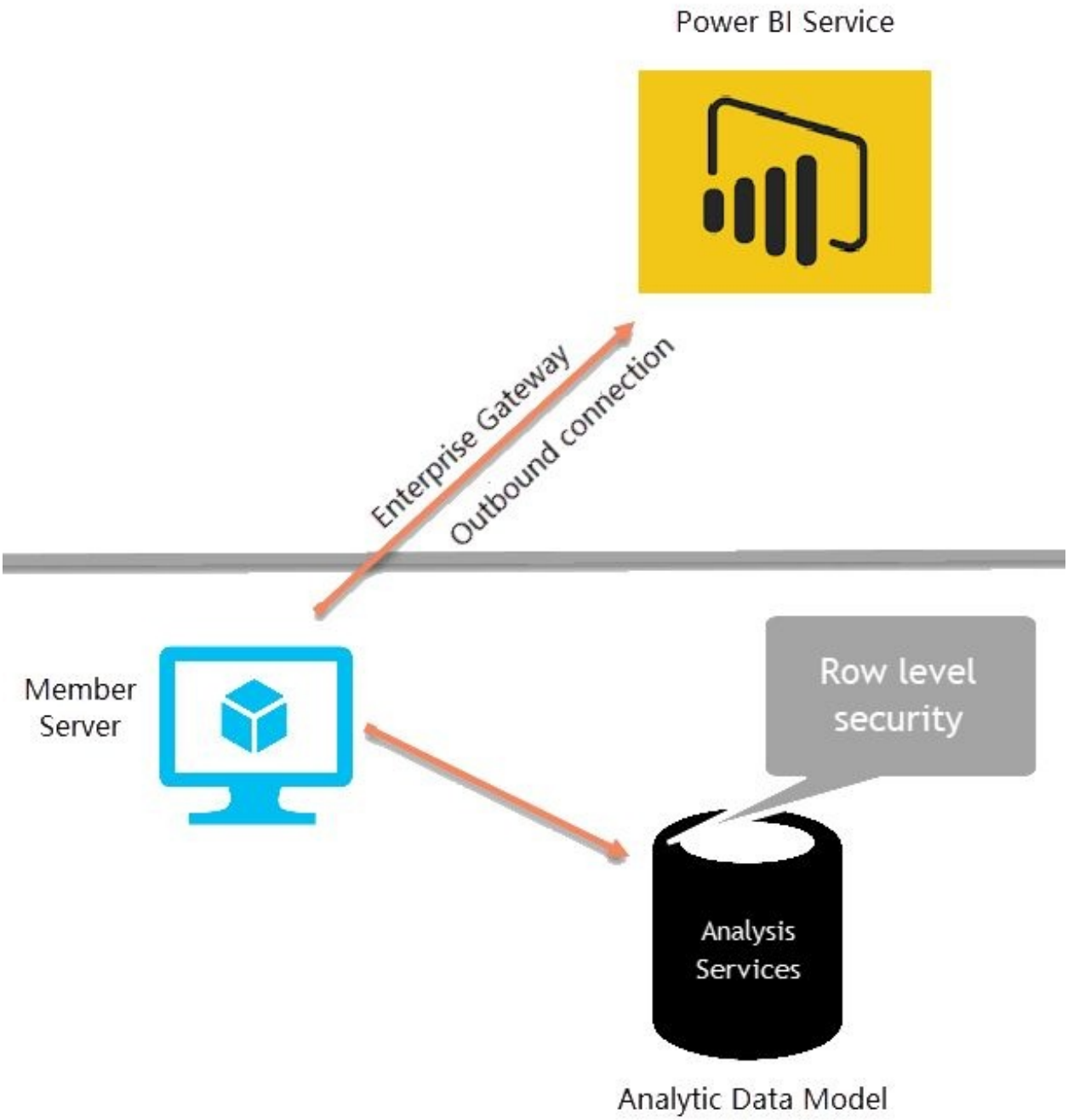


Figure 20-04: Enterprise Gateway providing access to on-premises data

Enterprise Gateway

Tools such as Flow, Power BI, Power Apps, etc. need the Microsoft Enterprise Gateway to be deployed before they can access any on-premises source data. It should be deployed before any of those applications or related applications

require it. This will be needed when starting to test or use Power BI based reports and dashboards, and if Microsoft Flow becomes part of the integration solution.

The Enterprise Gateway provides a secure communication service. It avoids the need for applications to connect into the on-premises systems, and for incoming ports in the corporate firewall. The gateway connects outbound to Microsoft infrastructure, and performs certificate exchanges for authentication. Secrets held by the Microsoft service are encrypted by a key that is supplied by the local gateway via the certificate exchange.

The gateway can be installed on almost any server in the network. Although it can be installed (and commonly is installed) on the same server as SQL Server Analysis Services or the SQL Server database engine, it does not need to be. If you were using import mode instead of Live Query mode, it would be better supported on a dedicated server.

In this case though, when it connects to SSAS using Live Query mode, it impersonates the user logged into Power BI.

If, for some reason, the domain in Azure (and Power BI) is different to the on-premises domain, and directory sync isn't being used, the gateway also provides the ability to map the UPNs (user principal names) so that users in the Azure AD domain can be mapped to users in the on-premises domain.

You can read more about the Enterprise Gateway in the following article:

https://docs.microsoft.com/en-us/power-bi/service-gateway-onprem

Power BI Service for Dashboards

I recommend implementing Power BI for the deployment of dashboards and analytic reports. It can support both users within the corporate network and mobile users, as shown in Figure 20-05.

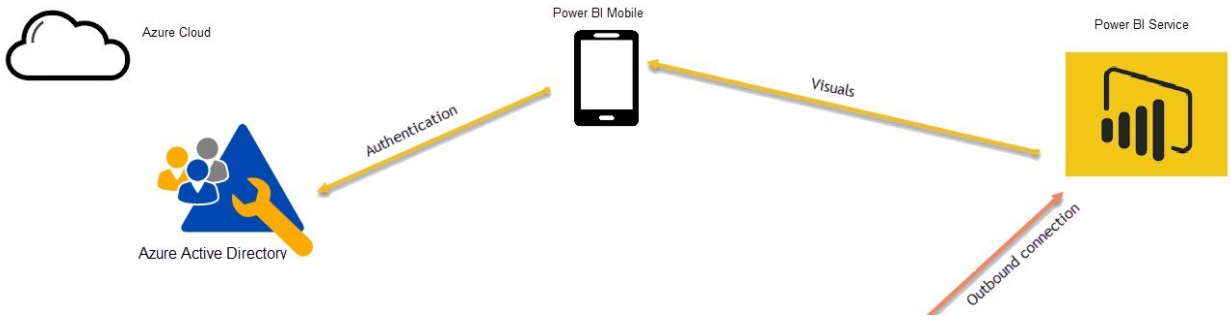


Figure 20-05: Authenticating mobile Power BI to Azure AD

Power BI Service

Power BI is one of the most popular applications that Microsoft has ever released. It is made up of several components:

- Power BI Service is an online (cloud-based) service that is used to provide dashboards and analytic reports to users. Data for the reports can be sourced from a very wide range of services, applications, and locations.
- While users could connect directly to the Power BI website, many will choose to use native applications to access this information. There are downloadable native Power BI applications (sometimes called Power BI Mobile) for Windows 8 and later (from the Microsoft Store), for iOS (from the Apple Store), and for Android (from the Google Play Store). It is also possible to embed Power BI directly into other applications with minimal code changes.
- Power BI Desktop is a free tool that can be used to create datasets, and reports based on those datasets. It can be used as a standalone "thick" client, or more commonly, these datasets and reports are published to the Power BI Service for consumption.
- Power BI Report Builder is a tool for building paginated reports to be deployed to the Power BI Service. This capability only works on Premium editions of the Power BI Service.

Power BI uses the hybrid AD (that I have also recommended) for authentication.

Power BI Workspaces

The Power BI Service has a concept of workspaces. These are collections of related Power BI objects. Workspaces are a security boundary. I generally use them for two purposes:

- To separate items being worked on by development teams
- As separate environments for deployment (i.e. UAT, Staging, Production)

Connecting Other Applications

Tableau

Many organizations already have an existing investment in Tableau reporting. Generally, I find that companies migrate from Tableau to Power BI when they have the opportunity, based on licensing. However, while it would be possible to continue to use Tableau to connect to its current data sources (often direct connections to source systems), if use of Tableau is continued, it is likely that it should move to using the recommended dimensional or analytic data models as a data source, or once the XMLA endpoints for the Power BI service are fully released, Tableau could connect to those interfaces.

Excel

Most organisations have many people who spend a great deal of time using Excel and are very familiar with it. Excel is a very capable tool and it could be used for further reporting and analytics via:

- Connection to the dimensional data model
- Direct connection to the analytic data model in SSAS (Excel has very capable options for doing this)

Summary

Power BI can be a key part of an enterprise BI strategy that keeps sensitive data on-premises while still providing secure access to dashboards.

About the Author

Greg is one of the better-known data consultants in the world. He is a long-term Microsoft Data Platform MVP, and a member of the Microsoft RD (Regional Director) program. Microsoft describe the RD program as consisting of “150 of the world's top technology visionaries chosen specifically for their proven cross-platform expertise, community leadership, and commitment to business results”.

Greg heads up a boutique data consultancy called SQL Down Under in Australia. He is best known for his SQL Down Under podcast, SDU Tools, and his online training school. Greg regularly presents at conferences world-wide.

Chapter 21: A Power BI-only Solution for Small Organizations

Author: Author: Gogula Aryalingam

Power BI plays many roles when it comes to analytics: A visualization and dashboarding tool in an enterprise solution, a prototyping tool to evaluate a solution to a business problem, a data profiling tool before undertaking an ETL implementation, and more. Now, there is this universe where organizational budget overrules many things; mainly information technology spending, including analytics. There are also other reasons, such as end users not seeing the entire picture of how analytics will help the organization. Hence, there are many cases where business intelligence is required to be implemented on a tight budget but showcasing quick value. This is where Power BI shines. In this chapter, I will talk about how, using just Power BI, you could design a complete business intelligence solution. Of course, there will be limitations and workarounds, but that is expected.

Background

Traditionally, a data warehouse has been the central component around which business intelligence is built. It worked on the principle of *the more data you bring, the more analytics you get*. Of course, data had to be of quality and formulated and structured to suit the business before any analysis was done. At present, however, analytics and the way analytics is done has changed immensely. There are various types of analytics that can be performed, which is enabled by the various types of data and numerous ways that these data are processed. Yet the old principle of a data warehouse remains, but with bells and whistles, and is implemented at a much grander scale where the traditional (relational) data warehouse becomes just part of the entire (enterprise) analytics platform or *modern data warehouse*.

A business intelligence tool such as Power BI fits in very nicely with a modern data warehouse. It also fits in very nicely in an enterprise data warehouse environment that is built along the traditional lines. However, there is also another arena that Power BI works well and thrives. In this arena, you have small organizations, departments of medium to large organizations, and in some case, specific large organizations themselves.

These institutions have one or more of the following in short supply; foremost is budget. Then comes other restrictions: the workforce to build a business intelligence solution (with the right skills, the right size, and the recommendation of the right technology), stakeholder buy-in (with the right conviction), and of course a champion who sees the value of analytics.

When such things are in short supply, the justification to invest in a business intelligence solution is naturally bleak, unless those in these institutions understand the value that analytics brings, and initiate a program that starts small, showcases value, then expands across departments or verticals of the organization, and finally up the ranks for the entire organization. The keyword, of course, is *budget* and to help come around this hurdle we have Power BI.

So, the premise of this chapter, and what it will build upon throughout its course will be the following: how with Power BI, small organizations, departments of medium to large organizations, or in some cases even large organizations themselves can start, evolve and build a complete business intelligence solution with the least investment. Power BI goes beyond being a cost-effective tool by providing the features and functionality to develop such a solution without the need for any other technologies, hence this chapter will deal with how all of this

can be achieved. Of course, there will be some exceptions and limitations, but that will be dealt with as we progress.

Putting a Solution Together with Power BI

A cautious approach to start a business intelligence solution is to start small with something tangible that provides value to a stakeholder, then get feedback and incorporate changes. This, in most cases, would be a process of 3-4 cycles. A tangible deliverable ideally would revolve around a business process or activity, such as sales performance monitoring, or an executive dashboard.

Once the value is proven, you move on to the next piece, and so forth. When the time comes, of course, an enterprise solution will be the natural course. While by that time much value will be seen by stakeholders, where budget may not be a question anymore.

Let us first look at a conceptual solution to determine who the actors are, and the processes that will be put forth for these actors to enable analytics, and to consume analytics for business intelligence. All of what you will see in this solution needs to be implemented at the beginning but can be included as the solution gains acceptance, and new requirements come in.

The Actors

The most important actor is the business user, also called a business analyst. Why the most important? Because this actor, performs a variety of activities in a business intelligence initiative such as this. From understanding the business needs, then going back to the various business systems to identify the right data sources, connecting to them, extracting, transforming and integrating the data, building business models on top of the data, and finally crafting the reports against the business models to answer specific business questions.

Now, it does not mean that such an individual, nor that the business analyst role should be carrying out all these tasks. Depending on the complexity and workforce that is available, the data wrangling and modeling task can be separated out to an individual with technical skills.

The end user becomes the ultimate consumer. They are the folk who would make or break the initiative. If the value is delivered to them, and that too in good time, they are the ones who would eventually become sponsors and stakeholders for the initiative.

Power users are actors who would come into the picture later in the initiative. These are users who go beyond what a business analyst would do to provide analytics to end users. Data scientists also make up this group.

The Solution

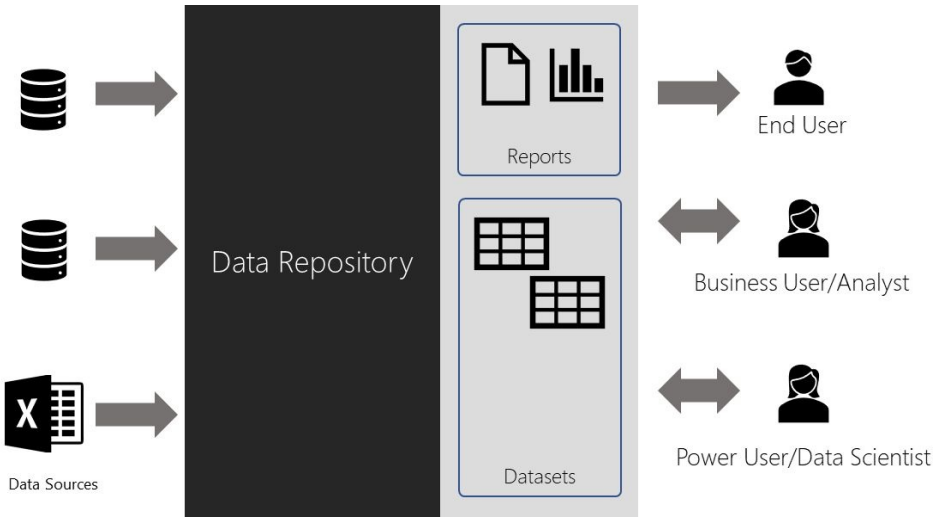


Figure 21-01: The Solution

In principle, the solution can be depicted as what you see in Figure 21-01. Information that is present in multiple data sources needs to be brought into the system on to a data repository. The three types of users would want to access and manipulate information in multiple ways. The end user needs to access and read report and dashboards to act upon the insights. To facilitate that, the business users will create perform the data cleansing, structuring, and formulating of data and load them to datasets, and then create reports and dashboards. The power users will go beyond what the business users do to create complex datasets and analyses to provide to the end users. In many cases, the business user will play the same role as the power user.

As simple as the above narrative explains the solution, the process of doing this can quickly become a nightmare. Power BI has evolved over the years and months into a tool that does wonders with data. Many an insightful report and dashboard complete with beautiful layouts have been built, and those who use its insights to make informed decisions have seen value in these “solutions”.

The issue, however, occurs when many such overzealous solutions start cluttering the environment that it becomes quite hard to maintain. Moreover, you would not know if all these solutions indeed possessed the single version of the truth. How one set of users define a metric, or a KPI differs from how others define them. When these two sets of users come together at a quarterly sales summit and start presenting their insights, everything but fisticuffs break loose. It becomes evident that when self-service business intelligence starts hitting the

brinks of madness, a method needs to be put into place to contain it

The solution that we are going to focus on in this chapter will be limited to Power BI Pro functionality. Power BI Premium features are deliberately left out since, as suggested in the Background, we will be focusing on small organizations.

Data Movement and Processing

Data Movement and Processing in Typical Cases

The movement of data from its sources, all the way to the reports follows a specific path. The following diagram outlines such a path that is similar to what traditional, enterprise, or even modern big data solutions make use of:

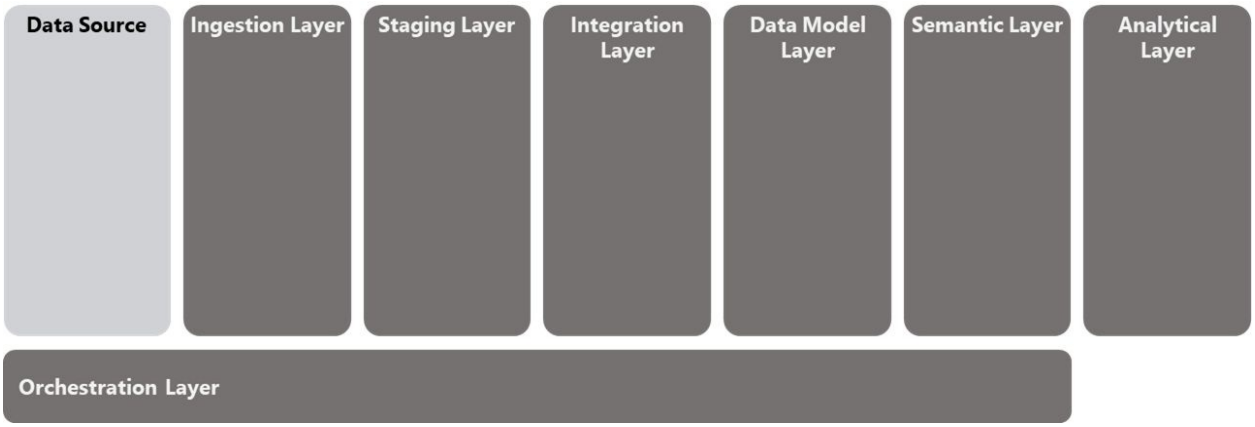


Figure 21-02: Typical Data Integration and Modelling Architecture

In a typical data integration architecture, the **Ingestion** layer pulls in data from the source systems and dumps them as-is on the **Staging** layer. Of course, the staging layer will be structured for effective dumping. This is the extract and load mechanism that we talk about in big data scenarios where data is ingested to a data lake, or in traditional cases when data is extracted and staged from source systems, to avoid performance issues at the source.

Then, when the need for analysis and reporting arises, relevant data will be cleansed, transformed and integrated via the **Integration** layer into the **Data Model** layer (which essentially will be transforming data onto a relational data warehouse or data mart, for example). The data model will evolve as more and more requirements are fulfilled but will remain the base organization-wide.

The **Semantic** layer provides data modelled for the business at a business function-specific level, be it descriptive, diagnostic, or even prescriptive. All that the users must do is use it in the **Analytical** layer in various ways; an example is

a self-service analytical report.

The complete data movement pipeline is facilitated and coordinated by the **Orchestration** layer.

Data Movement and Integration in Power BI

Now, in the Power BI-based scenario, not all the above layers can be segregated as they are, and they need not be either. For one, the above is ideal when analytics is designed and implemented for an enterprise since multiple technologies are used for specialized tasks. Hence, in our case, let's try looking to replicate the above as best as we could, but at the same time combine layers to overcome technical hurdles imposed by the single technology that is in focus here.

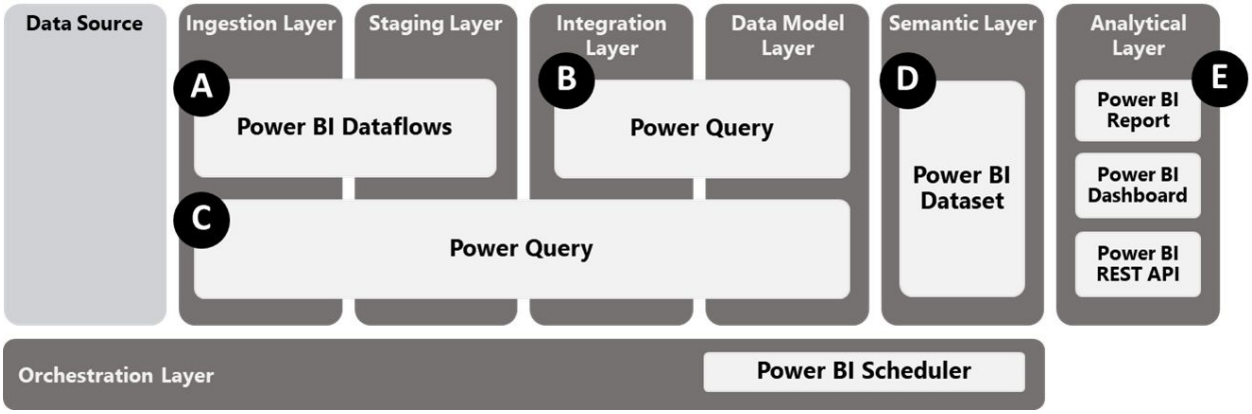


Figure 21:03: Data Integration and Modelling on Power BI

In the Power BI-only universe, one of the disadvantages that we have is the lack of a staging ground. However, this need not be a hindrance. We will utilize Power BI Dataflows to extract and prep entities that will be used as the basis for dimensions across multiple business units. Think of Dataflows entities as the catchall dimension table of a data warehouse. It will possess all the possible attributes of every dimension.

Take a look at Figure 21-04, which is the solution architecture; it shows three layers, the **curated staging layer**, the **business function layer**, and the **end user layer**. To understand how the *data integration and modelling architecture* (Figure 21-03) correlates with the *solution architecture*, you will need to refer to both diagrams when you read the rationalization below.

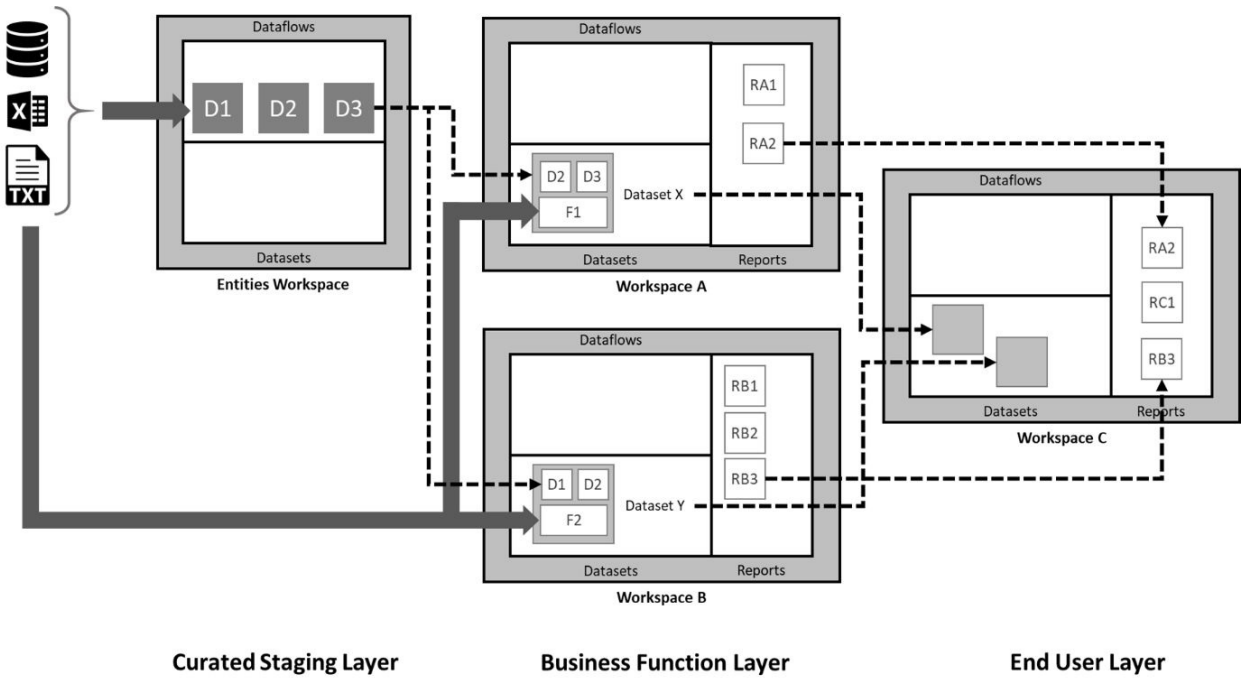


Figure 21-04: Solution Architecture

Curated Staging Layer

First up, to stage data, we do not have the luxury of a dedicated storage area. Hence, staging data as-is will not make much sense. Hence, we will build a set of entities in the **curated staging layer**. This layer will serve as a stage, but with cleansed data, with each entity containing all the information that will be required across business functions.

Now, the type of data that is used for analysis is categorized as dimensions and facts. Dimensions are used across the organization, some more than others. Hence, if a sales analyst from the sales department creates and uses the Product dimension for their sales analysis reports and dashboards, it would only make sense that the same dimension is used for marketing campaign analysis. Of course, the attributes that make-up product in sales would defer from those that make-up product in marketing. Hence, it is important that all possible attributes are available when creating a dimension so that it would cater to various types of analyses and business units.

Therefore, one of the first things that will have to be defined are entities. Entities are objects that are singular and identifiable within an organization. It is the entities that morph into dimensions based on need. Information that makes up an entity may come from multiple sources, and they all need to be combined and structured appropriately before being used across the business.

Therefore, the first step would involve setting up a workspace that will only house entities. These entities will be created on Power BI using Dataflows. This is what will make up the curated staging layer: Data extracted, cleansed, and stored as Dataflows. The layer will consist of a workspace dedicated to maintaining Dataflows of entities for the organization (*Entities* workspace). Think of this as the entity vault, and would only contain dataflows of entities, and no datasets, reports nor dashboards. Dataflows, once created, can be accessed from other workspaces. This will allow users from across multiple disciplines to access the Dataflows and build dimensions that are unique to their business context. Typically, it would be the business analyst role that performs the task of identifying, designing, and creating dataflows for each required entity.

This is **Process A** depicted in Figure 21-03.

Business Function Layer

The next step will be the exercise of creating packages of business process or business function-specific content, for example, sales or finance. These packages will be workspaces that belong to the *business function layer*, and one workspace per business function will be built. Hence, you will have a *Sales* workspace, a *Finance* workspace, and more.

Here, the analyst will create dimensions based on the entities customized for the current business function. If it is the sales business unit that we are looking at for example, and the Product dimension is one of those being built, then this dimension will be customized to suit the needs of sales personnel, for instance, including a product hierarchy, skipping specific columns, and modifying other columns. If the marketing business unit also needs the product dimension for their analysis, they will be provided with their own one customized to their specific needs. Each of these dimensions will be derived from one place: The Entities dataflow from the *curated staging layer* and built separately within a workspace of their own context, inside a dataset in the form of a Query, using Power BI Desktop.

This is **Process B** depicted in Figure 21-03.

Next up, we have facts (or the transactions) that will be pulled out of one or more systems. The data for facts will naturally be more significant in record size, and sometimes more extensive in terms of columns than dimensions. Transactional data are usually business function-specific, hence staging them as

entities will not make sense. The fact data will follow the long path of being ingested, transformed and integrated straight to the data model layer within a Power BI dataset. The fact tables will meet the dimension tables within this dataset as queries on Power BI Desktop. So, unlike on a data warehouse, where data can be staged, and stored temporarily, in the world of Power BI data preparation is mostly dealt with dynamically.

This is **Process C** from Figure 21-03.

The datasets are finalized when semantic modelling is performed on top of the queries. This is done by creating measures, formatting them, creating hierarchies on the dimensions to further enhance them, hiding columns that have no business purpose, creating relationships among the dimensions and facts, and more. This is when you will have the entire business function-specific semantic model.

This is **Process D** from Figure 21-03.

The next step, within the business function layer, is to build the reports and then dashboards off them (if required). These reports will be built off the dataset. However, when building the reports, the best practice would be to first publish the dataset to the Power BI service onto the business function-specific workspace and then build reports off a fresh instance of Power BI Desktop by connecting to the published dataset. Reports can also be created off the Power BI portal via a web browser, if necessary.

This is **Process E** depicted in Figure 21-03.

The business function-specific workspaces will be owned and maintained by a set of business analysts who belong to that business function, i.e. sales analysts will be responsible for the creation, maintenance and governance of content of the sales workspace, while market analysts will take care of the marketing workspace.

What you will ultimately have are one or more datasets relating to the business function, reports built off these datasets, and dashboards if necessary. All of these can be packages if necessary and published as an app within the organization (Workspaces A & B). Each of these workspaces will be managed by analysts from the related business function, i.e., finance will put out a finance workspace/app, sales would do the same, and so forth.

So now that we have departmental analytics packaged into workspaces. These will become standard across the organization, and users and analysts from across

the organization can tap into these for their analytical needs, apart from having their own analytics. However, it is a good idea that these business function-specific datasets be promoted for use by others, thus indicating that the dataset is indeed something that has been relatively used and tested. This would give confidence to users of the organization that the dataset can be used to answer crucial business questions. These datasets can be further endorsed by going through a certification process that will allow a specified select set of users, ideally, those who are subject matter experts of the business function domain to endorse the dataset, thus enabling wider adoption and trust across the organization.

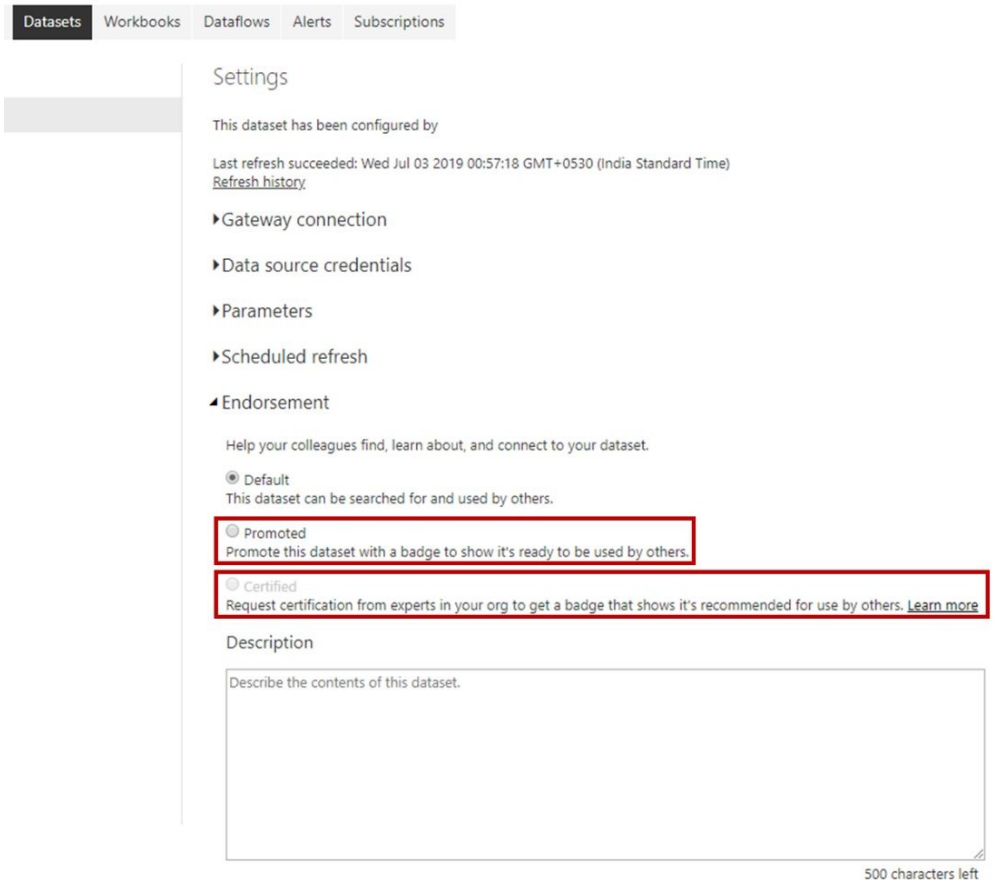


Figure 21-05: Options to promote and certify a dataset

End User Layer

Business users from across the organization can build upon the packaged business function-specific datasets and reports by bringing them in, by building reports on top of these datasets and creating their own datasets and reports within their workspace (Workspace C). Management dashboards that showcase aspects of the entire organization, drilling through to individuals reports from

various business functions is an excellent use case to illustrate this.

Security

The containers that we work within this architecture are workspaces, and as such, the Entities workspace needs one or more contributors who will perform the task of creating entities using dataflows. All users who will need access to consume the entities can be assigned the viewer role, so that they may not modify the contents of this workspace. A good idea will be to create AD security groups for the viewers of the entities, add the viewers to the group, and then set up the group as a viewer on the Entities workspace.

The viewers will be added as contributors on their respective business function workspaces, and here too these contributors can be bundled up inside an AD security group. Users from outside of the business functions may be included as viewers, and this time via another AD security group.

Development life-cycle

Source control

Unlike a solution where a development team completes building the solution, setting it up, and leaving the users just to use the system, a solution such as these involves the users' active participation. Hence when source-controlling the code that was crafted for the dataflows, datasets and reports need to be versioned and stored.

The easiest and most straightforward method for source control will be to use OneDrive or SharePoint. These technologies have a Check in/Check out function that allows you to keep track of a file's history of changes, which makes it ideal to double up for source control. Azure DevOps can also be used with a five-user-free deal, is purpose-built and is loaded with functionality. However, it can be an overkill.

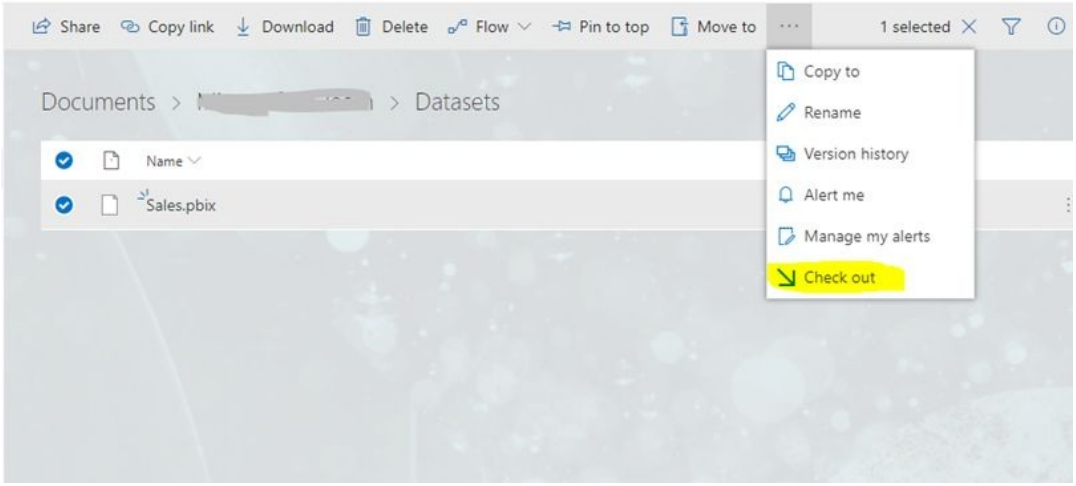


Figure 21-06: Check out function on SharePoint

Regardless of the approach that we utilize, we need to be cognizant to the fact that not all components of the solution can be source controlled in a straightforward manner. When listed down, this is what it looks like:

Component	Is source-control straight forward?	Method
Dataflows	No	Copy Power Query code behind from the Advanced editor, paste to a text editor, before being saved.
Datasets	Yes	Save the PBI desktop

		file used to create the dataset.
Reports	Yes	Save the PBI desktop file used to create the report
Dashboards	No	No source control mechanism available

Figure 21-07: Source control likelihood of components

Deployment

Like any solutions development practice, a solution such as this will have to follow a specific deployment process in conjunction with the source control process. An easy approach to take will be through the utilization of a sandbox workspace, where business users can build and test their data structures and reports first.

The process of development will start within the sandbox workspaces where the business analysts will start by building the dataflows to create the entities within a dataflow called Entities. As they are completed the Power Query source of each of the entities will be copied to a text file, named with the name of the entity and put into source control. The next step of creating the datasets will take place on the desktops of the business analysts, where they would create datasets on Power BI Desktop and then published to the business function-specific workspaces. Once published, these *.pbix files will be source controlled in appropriately-named folders. Reports, however, will not be created on the same file that houses the datasets. Instead, reports will be created in separate *.pbix files connecting to the published datasets on the Power BI service. These files, too, will be published to the appropriate workspaces before being source controlled. One thing that cannot be source controlled, however, are the dashboards that you build off report parts.

Once all the development is completed and tested on your sandbox, you will have to “push” your development to the “live” or “production” environment. This is where you will have to “replicate” your development by pulling out the code from source control and applying it on the new environment. If each of the components that made up the solution were able to be saved as files, then the application would be quite straightforward. However, in the case components such as dataflows, you will have to re-create the steps, with the only “easy part” being copying the code over from source control.

One thing that you would need to keep in mind while pushing your development to the live environment is that data source connectivity and credentials may have to be altered at the dataflow, dataset, and report levels.

Going through this process of sandbox-first, and then going live will increase your implementation effort, and will be too much of an overkill for specific organizations. In cases like that the sandbox may be skipped as long as a similar process of source control is set up as a process for developing the primary solution.

Summing it up

To conclude, Power BI is a versatile tool, which can work on the sidelines as a prototyping tool, and at the same time measure up to provide a complete business intelligence solution. Solutioning, however, is not just about building data models and reports to solve business problems. It is to provide a streamlined and structured process to build data models and reports to solve business problems, so that business users can be efficient, data is ensured to be relevant and reliable, while end users will know where to go for which piece of information. This can be achieved quite well with Power BI by architecting and designing the solution along with the ideas presented in this chapter.

About the Author

Gogula is a 12-time Data Platform MVP hailing from Sri Lanka. He currently provides technical leadership for Intellint, the business intelligence and analytics arm of Fortude. 14 of his 19 years in technology has been on data and analytics on the Microsoft space, now mainly focused on the cloud. His experience is complemented with a passion for data and analytics through his involvement in technical communities, writing, speaking, mentoring, speaking, and serving as subject matter expert for Microsoft certifications. He is the community leader of the Sri Lankan Data Community and is also PASS regional mentor for South Asia.

---

[1] http://www.businessdictionary.com/definition/governance.html