# Deploy Power BI Solutions with Semantic Models and Reports

Power BI Desktop supports Power BI Desktop Developer Mode. Power BI Desktop has traditionally allowed saving projects in PBIX file format. Power BI Desktop now allows saving projects in Power BI Project (PBIP) format.

* Power BI Desktop saves semantic model definition using model.bim and definition.pbism
* Power BI Desktop saves report definition using report.json and definition.pbir

A screenshot of a computer

Description automatically generated

.pbi folder used to store local resources that should not be saved into source control system.

To use PBIP formats, you must enable **Power BI Project (.pbip) save option** in Power BI Desktop Options dialog.

A screenshot of a computer program

Description automatically generated

Power BI Desktop offers a feature to save the item definitions for semantic models and report in more-modern and advanced formats. Optionally configure semantic models and reports to use newest definition formats.

A white background with black text

Description automatically generated

## TMSL versus TMDL - Semantic Model Formats

Semantic Model Definitions can be saved in TMSL format or TMDL format

* TMSL is simple - a single model.bim file contains entire semantic model definition
* TMDL is more granular – better for source control when multiple developers are working on same model

A screenshot of a computer

Description automatically generated

## PBIR-Legacy versus PBIR - Report Definition Formats

Report Definitions can be saved in PBIR-Legacy format or PBIR format. PBIR-Legacy is original Power BI report definition format with entire definition stored in report.json file. PBIR is modern, more granular format which allows for editing/creating definitions using JSON schema

A screenshot of a computer

Description automatically generated

# Deploy Solutions with Lakehouses and Notebooks

Let’s start with Fabric lakehouse fundamentals. Lakehouse used to store structured data and unstructured data in a single location. Lakehouse data transparently stored in OneLake using Azure Data Lake Storage Gen2 (ADLS Gen2). Lakehouse data accessible across workloads like data engineering, data warehouse, data science & BI.

Lakehouse enables storing and refining data using medallion architecture (bronze>silver>gold)

Lakehouse has a specific structure. The **Tables** section used to store structure data in tables in Delta format. **Files** section used to store unstructured data as files.

A screenshot of a computer

Description automatically generated

Ingesting Data into Lakehouse.

* Running Spark jobs
* Running Data Pipelines
* Creating Shortcuts
* Using ADLS Gen2 APIs
* Running Fabric Gen2 Dataflows

For running Spark jobs, you can run them as notebooks or as Spark Job Definitions

For running Data Pipelines. You can build a data pipeline with activities to copy data from external sources

Creating Shortcuts allows you to create references to make files from an external source appear to be in lakehouse storage

You can use ADLS Gen2 APIs to copy files into the File section of a lakehouse. This is implemented by client application which uploads files into lakehouse storage using push approach

Running Fabric Gen2 Dataflows. Running a dataflow which imports data and saves it delta tables in lakehouse storage

### The Apache Spark Runtime

Fabric Provides Apache Spark Runtime. Spark is a unified engine for large-scale data analytics. Spark code can be executed through Spark Job Definition. Spark code can be executed using interactive Fabric Notebook.

A diagram of a spark session

Description automatically generated

Apache Spark provides APIs to read and write data to/from Lakehouse in OneLake

Steps to provisioning flow which builds out medallion architecture

* **Create bronze layer** by coping CSV files into **Files** section
* **Create silver layer** by loading CSV data into dataframes and saving them as delta tables with OLTP schema
* **Create gold layer** by transforming data from silver tables into delta tables with star schema

A screenshot of a computer

Description automatically generated

You can leverage Fabric support for Spark to create and populate lakehouse tables

* Create lakehouse (or multiple lakehouses for greater isolation)
* Create Spark Environment as workspace item for running Spark jobs (when Environments API becomes available)
* Create Notebook (or Spark Job Definition) containing ETL logic to ingest data files and to create tables
* Run Notebook using Job Scheduler – Run Job On-demand API

### Create a Lakehouse using the Create Item API

Lakehouse created without item definition – you only pass displayName and type.

public static Item CreateLakehouse(Guid WorkspaceId, string LakehouseName) {

// Item create request for lakehouse des not include item definition

var createRequest = new CreateItemRequest(LakehouseName, ItemType.Lakehouse);

// create lakehouse

return CreateItem(WorkspaceId, createRequest);

}

xx

A screenshot of a computer

Description automatically generated

Response from Create Item includes lakehouse id which is required to create notebooks

A screenshot of a computer code

Description automatically generated

### Create a Notebook using the Create Item API

Notebook created calling **CreateItem** with notebook part **CreateLakehouseTables.ipynb**

Notebook code and all other content stored in single JSON file with **ipynb** extension

Fabric also supports calling **CreateItem** with standard Python file with **py** extension

### Create Spark Environments using Create Item

ooeoe

### Configure Workspace Spark Settings

Code used in developer sample application to create lakehouse

### Create a DirectLake Semantic Models on a Lakehouse

# Deploy Solutions using Warehouses and Data Pipelines

# Deploy Solutions using Real-time Intelligence