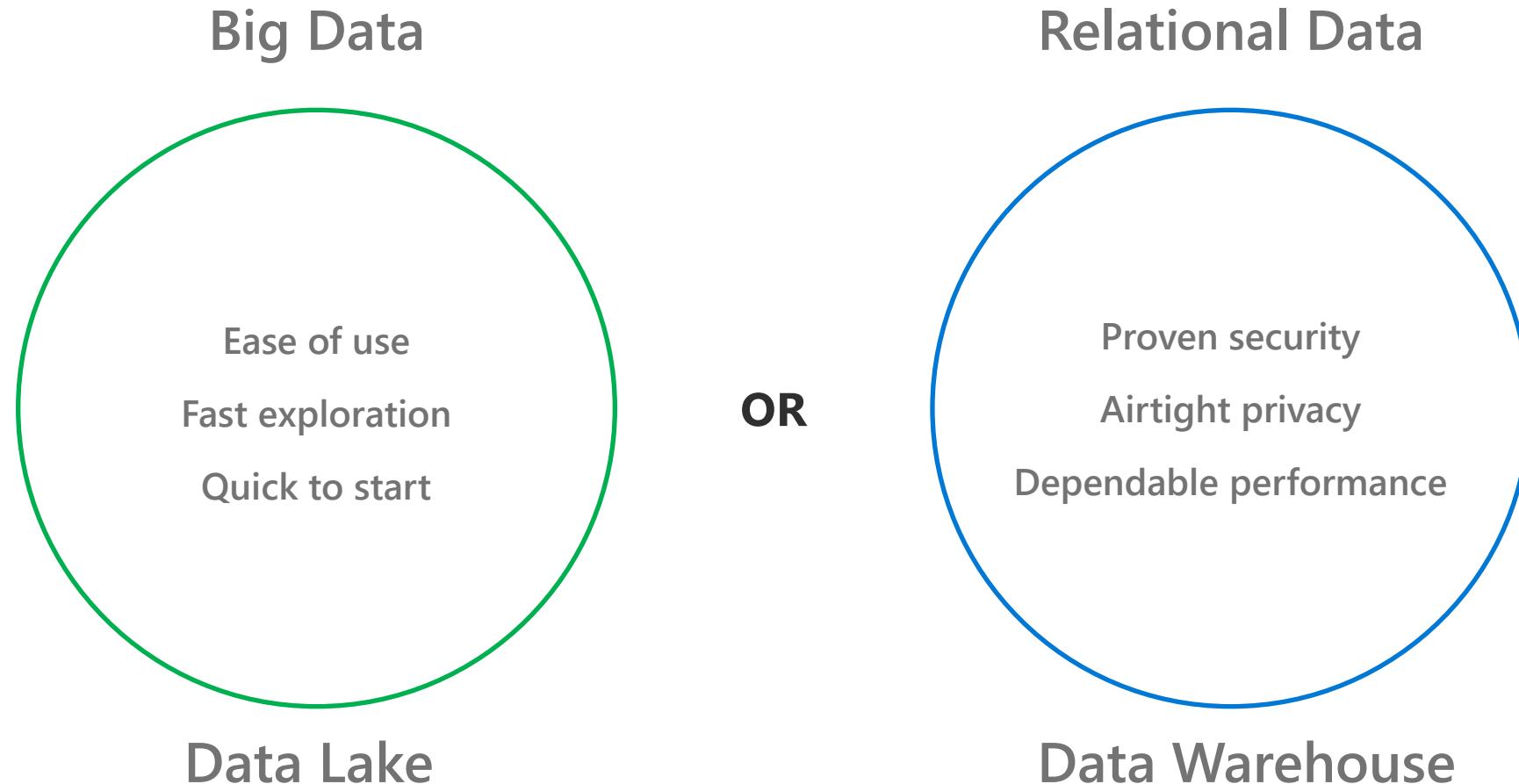


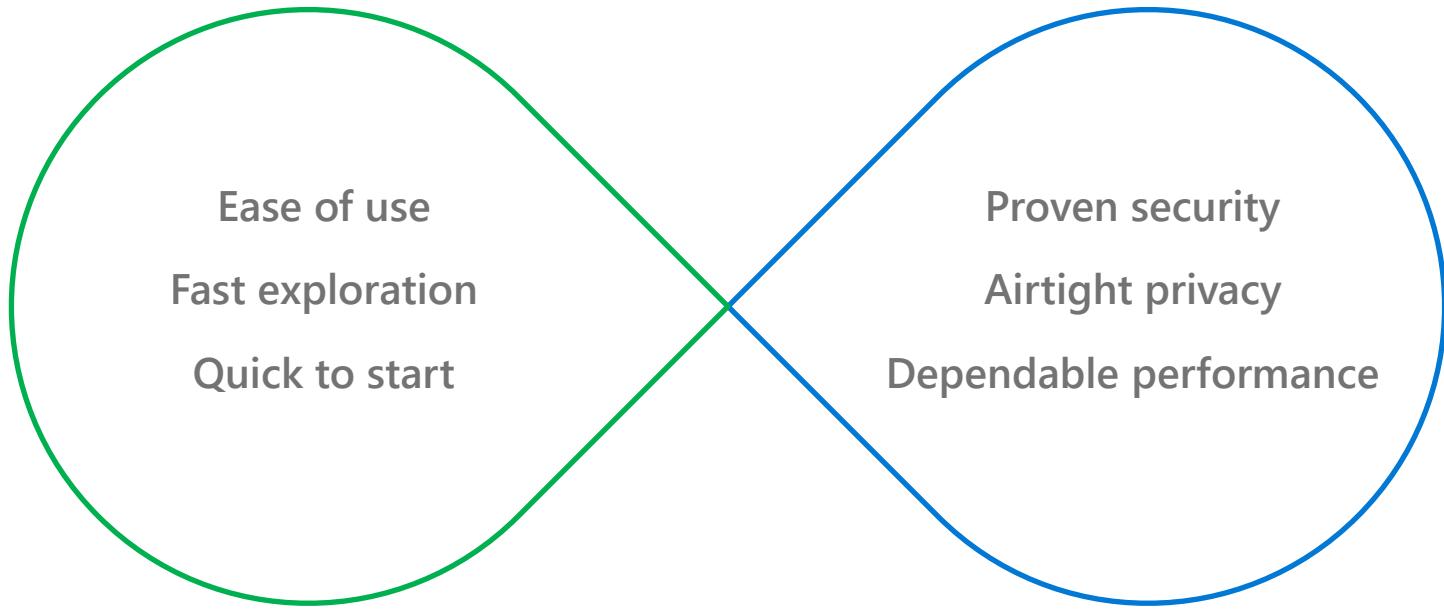


# Azure Synapse Analytics

This is a result of businesses being forced to maintain two critical, yet independent analytics systems



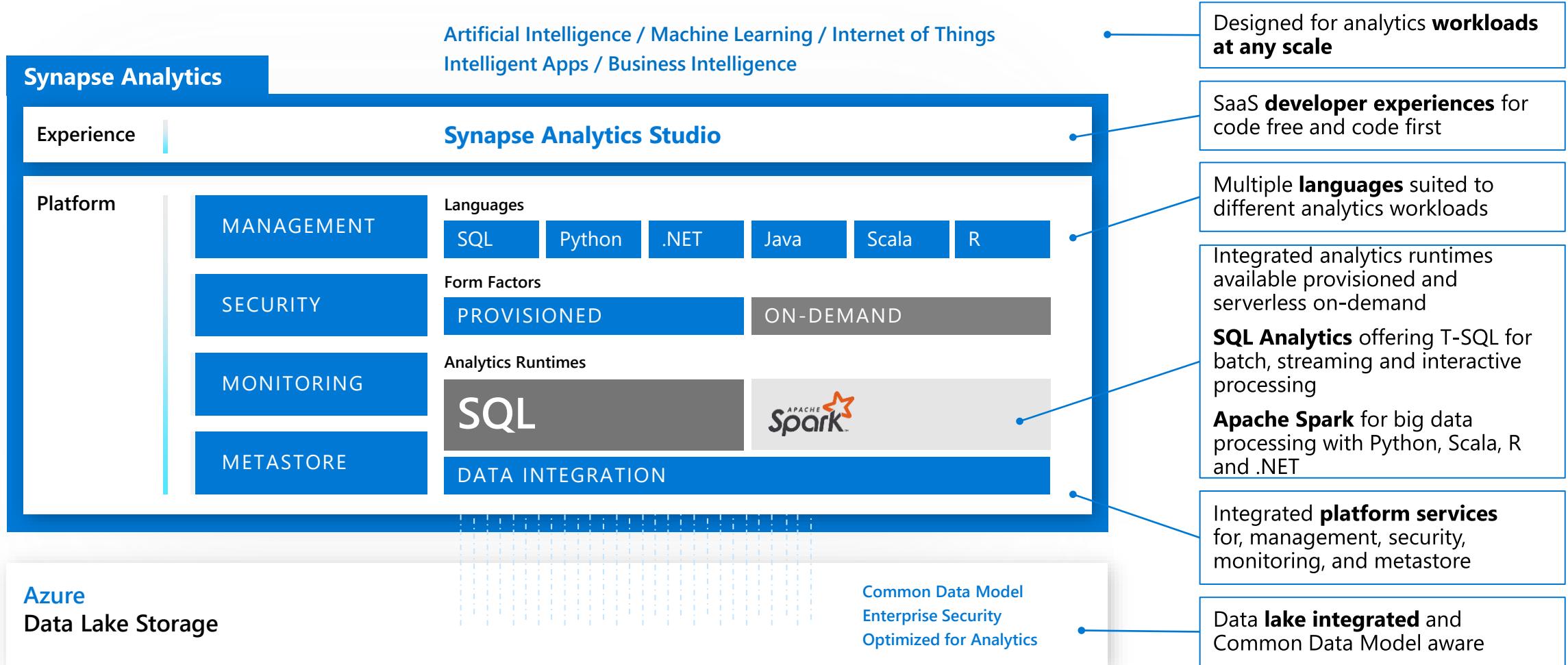
# Azure brings these two worlds together, in a single service



## Welcome to Azure Synapse Analytics

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# AZURE SYNAPSE ANALYTICS

Built-in tools:

- Notebooks
- SQL Editor

Single pane of glass:

- Security
- Monitoring
- CI/CD
- Metastore
- Governance

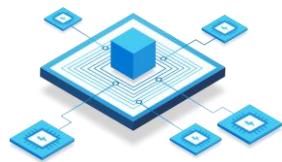


Workloads

- Data Warehousing
- Data Virtualization
- Logical Warehouses
- Data Integration
- Spark/ML
- Stream Analytics
- Business Intelligence



Synapse SQL



Synapse Spark



Synapse Pipelines



Synapse Studio

# Sections



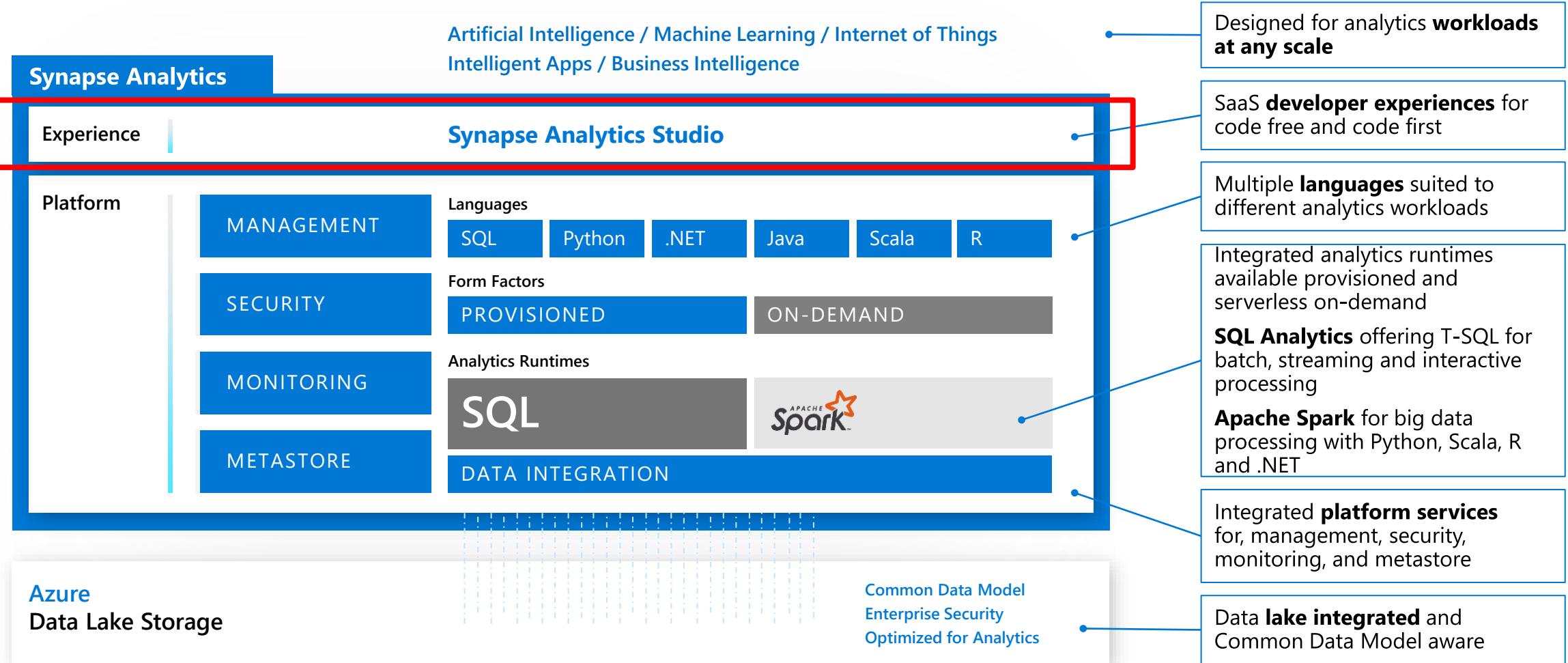
- [Studio](#)
- [Orchestration](#)
- [SQL Provisioned](#)
- [SQL Serverless](#)
- [Spark Analytics](#)
- [Security](#)
- [Monitoring](#)
- [Management](#)
- [Metastore](#)
- [Synapse Link for Cosmos DB](#)



# Azure Synapse Analytics Studio

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Studio

A single place for Data Engineers, Data Scientists, and IT Pros to collaborate on enterprise analytics

The screenshot shows the Microsoft Azure Synapse Analytics Studio interface. At the top, the navigation bar displays "Microsoft Azure | Synapse Analytics > internalsandboxwe5". The top right corner shows a user profile with the email "prlangad@microsoft.com" and the Microsoft logo. Below the navigation bar, the main header reads "Synapse workspace" and "internalsandboxwe5". A "New" button is located in the top left of the workspace area. The workspace features a large, stylized graphic of a globe with bar charts and network connections. Below this graphic are four main functional buttons: "Ingest", "Explore", "Analyze", and "Visualize", each with a brief description and a small icon.

**Resources**

Recent    Pinned

NAME	LAST OPENED BY YOU
CopyPipeline_0313	13 minutes ago

**Useful links**

- [Synapse Analytics overview](#) Discover the capabilities offered by Synapse and learn how to make the most of them.
- [Pricing](#) Learn about pricing details for Synapse capabilities.
- [Documentation](#) Visit the documentation center for quickstarts, how-to guides, and references for PowerShell, APIs, etc.
- [Give feedback](#) Share your comments or suggestions with us to improve Synapse.

# Synapse Studio

Synapse Studio divided into **Activity hubs**.

These organize the tasks needed for building analytics solution.

The screenshot shows the Microsoft Azure Synapse Studio interface. On the left, there is a vertical navigation bar with the following items:

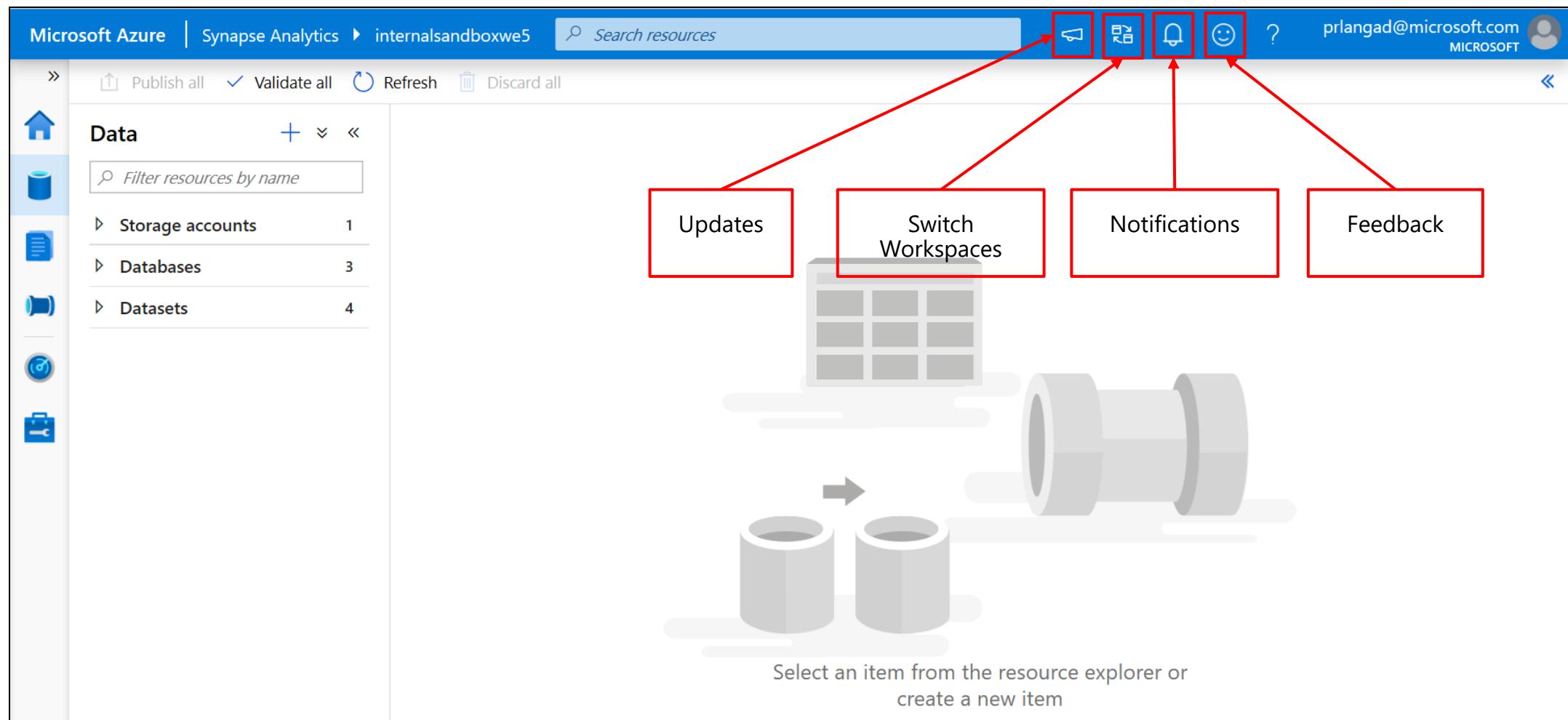
- Home (selected)
- Data
- Develop
- Orchestrate
- Monitor
- Manage

A red box highlights the "Home" item, and a red arrow points from it to the "New" button in the center of the screen. The center of the screen displays the "Synapse workspace" with the name "intern" and a "New" button. Below this, there are six activity hubs arranged in a grid:

- Overview**: Quick-access to common gestures, most-recently used items, and links to tutorials and documentation.
- Data**: Explore structured and unstructured data.
- Develop**: Write code and define business logic of the pipeline via notebooks, SQL scripts, Data flows, etc.
- Orchestrate**: Design pipelines that move and transform data.
- Monitor**: Centralized view of all resource usage and activities in the workspace.
- Manage**: Configure the workspace, pool, access to artifacts.

# Overview Hub

Ease of access to get updates, to switch workspaces, to get notifications and to provide feedback





# Synapse Studio Overview hub

# Overview Hub

It is a starting point for the activities with key links to tasks, artifacts and documentation

The screenshot shows the Microsoft Azure Synapse Analytics Overview Hub. The top navigation bar includes 'Microsoft Azure' and 'Synapse Analytics' with a workspace path 'internalsandboxwe5'. The right side of the header shows a user profile for 'prlangad@microsoft.com' and 'MICROSOFT'.

The main area is titled 'Synapse workspace' and 'internalsandboxwe5'. It features a 'New' button and four primary activity cards:

- Ingest**: Use the copy data tool to import data once or on a schedule.
- Explore**: Learn how to navigate and interact with your data.
- Analyze**: Learn how to use SQL or Spark to get insights from your data.
- Visualize**: Build interactive reports with integrated Power BI capabilities.

A large red box highlights the 'Ingest', 'Explore', 'Analyze', and 'Visualize' cards.

Below these cards is a section titled 'Resources' with tabs 'Recent' (selected) and 'Pinned'. A table lists a single pinned item:

NAME	LAST OPENED BY YOU
CopyPipeline_0313	13 minutes ago

To the right is a 'Useful links' section with a red box around it, containing the following links:

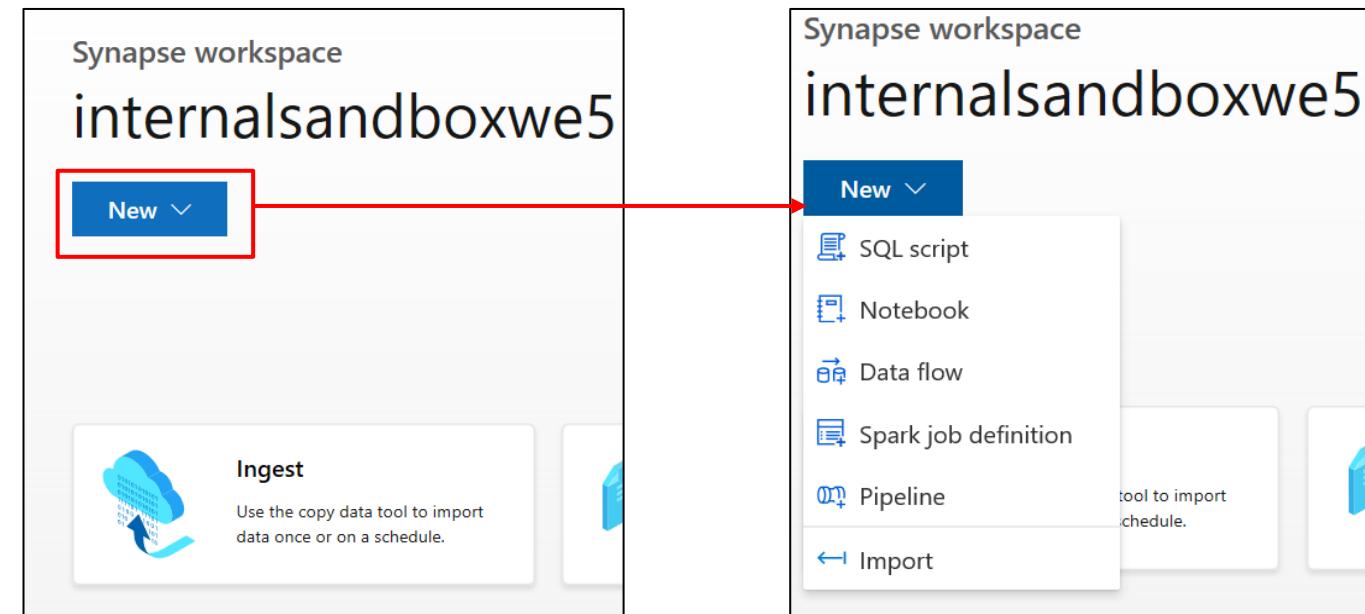
- Synapse Analytics overview**: Discover the capabilities offered by Synapse and learn how to make the most of them.
- Pricing**: Learn about pricing details for Synapse capabilities.
- Documentation**: Visit the documentation center for quickstarts, how-to guides, and references for PowerShell, APIs, etc.
- Give feedback**: Share your comments or suggestions with us to improve Synapse.

# Overview Hub

## Overview

**New** dropdown – offers quickly start work item

**Recent & Pinned** – Lists recently opened code artifacts. Pin selected ones for quick access



Recent	Pinned
NAME	LAST OPENED BY YOU
BOOT_AMLautoMLPredict	6 hours ago
SQLConnector	6 hours ago
TaxiCreateSparkTable	6 hours ago
Notebook 1	6 hours ago
NYCTAx1	6 hours ago

Show more ▾

Recent	Pinned
NAME	LAST OPENED BY YOU
NYCTAx1	6 hours ago



# Synapse Studio **Data hub**

# Data Hub

Explore data inside the workspace and in linked storage accounts

Microsoft Azure | Synapse Analytics > prlangadws2

The screenshot shows the Microsoft Azure Synapse Analytics Data Hub interface. The left sidebar has 'Data' selected. The main area has a 'Data' section with tabs 'Workspace' (highlighted with a red box) and 'Linked'. Below is a search bar and a list of resources under 'Databases': prlangadSQLDW (SQL pool), nycyellow (SQL on-demand), and default (Spark).

Microsoft Azure | Synapse Analytics > pr

The screenshot shows the Microsoft Azure Synapse Analytics Data Hub interface. The left sidebar has 'Data' selected. The main area has a 'Data' section with tabs 'Workspace' and 'Linked' (highlighted with a red box). Below is a search bar and a list of resources under 'Storage accounts': prlangadws2 (Primary - pr...), ADLSG2OpenDataSetSink ..., AzureDataLakeStorage1 (p...), AzureDataLakeStorage2So..., and labignite (labignite). At the bottom, there is a 'Datasets' section with a count of 17.

# Data Hub – Storage accounts

Browse Azure Data Lake Storage Gen2 accounts and filesystems – navigate through folders to see data

Linked ADLS Gen2 Account

Container (filesystem)

The screenshot shows the Microsoft Azure Synapse Analytics Studio interface. On the left, there's a sidebar titled 'Data' with sections for 'Workspace' and 'Linked'. Under 'Linked', a red box highlights the 'Storage accounts' section, which contains a list of accounts. One account, 'prlangadw2 (Primary - prlangaddemos)', is selected and also highlighted with a red box. Below it are other containers: 'filesystem (Primary)', 'holidaydatacontainer', and 'isdweatherdatacontainer'. Another red box highlights the 'nyctlc' container. The main workspace shows a breadcrumb trail 'nyctlc > yellow' with a red box around it. To the right is a table titled 'File path' with columns for NAME, LAST MODIFIED, CONTENT TYPE, and SIZE. The table lists 29 cached items, all of which are folders named 'puYear=nnnn' where nnnn ranges from 2001 to 2084. The last item listed is 'puYear=2084'.

NAME	LAST MODIFIED	CONTENT TYPE	SIZE
puYear=2001	10/25/2019, 2:25:03 PM	Folder	
puYear=2002	10/25/2019, 2:25:21 PM	Folder	
puYear=2003	10/25/2019, 2:25:03 PM	Folder	
puYear=2008	10/25/2019, 2:20:38 PM	Folder	
puYear=2009	10/25/2019, 2:19:33 PM	Folder	
puYear=2010	10/25/2019, 2:19:24 PM	Folder	
puYear=2011	10/25/2019, 2:23:56 PM	Folder	
puYear=2012	10/25/2019, 2:20:01 PM	Folder	
puYear=2013	10/25/2019, 2:19:52 PM	Folder	
puYear=2014	10/25/2019, 2:24:06 PM	Folder	
puYear=2015	10/25/2019, 2:20:12 PM	Folder	
puYear=2016	10/25/2019, 2:19:21 PM	Folder	
puYear=2017	10/25/2019, 2:20:28 PM	Folder	
puYear=2018	10/25/2019, 2:24:38 PM	Folder	
puYear=2019	10/25/2019, 2:20:33 PM	Folder	
puYear=2020	10/25/2019, 2:24:47 PM	Folder	
puYear=2021	10/25/2019, 2:28:34 PM	Folder	
puYear=2026	10/25/2019, 2:20:38 PM	Folder	
puYear=2029	10/25/2019, 2:28:13 PM	Folder	
puYear=2031	10/25/2019, 2:25:21 PM	Folder	
puYear=2032	10/25/2019, 2:28:22 PM	Folder	
puYear=2033	10/25/2019, 2:28:34 PM	Folder	
puYear=2037	10/25/2019, 2:25:03 PM	Folder	
puYear=2038	10/25/2019, 2:28:22 PM	Folder	
puYear=2041	10/25/2019, 2:24:47 PM	Folder	
puYear=2042	10/25/2019, 2:20:38 PM	Folder	
puYear=2053	10/25/2019, 2:20:39 PM	Folder	
puYear=2084	10/25/2019, 2:20:38 PM	Folder	

# Data Hub – Storage accounts

Preview a sample of your data

The screenshot illustrates the process of previewing data from a storage account in Azure Synapse Studio.

**Left Panel:** Shows the "Data" hub with a list of storage accounts, databases, and datasets.

**Middle Panel:** Shows the contents of the "tempdata" folder. A red arrow points from the "Preview" option in the context menu of the "SampleCSVFile\_2kb.csv" file to the right panel.

**Right Panel:** Displays the preview of the "SampleCSVFile\_2kb.csv" file. The file path is [https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile\\_2kb.csv](https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv). The file was modified on 10/29/2019, 1:30:21 PM. The "With column header" toggle is turned on. The data preview shows 10 rows of user information:

USER_ID	USERNAME	FIRST_NAME	LAST_NAME	GENDER	PASSWORD
1	rogers63	david	john	Female	e6a33eee18
2	mike28	rogers	paul	Male	2e7dc6b8a1
3	rivera92	david	john	Male	1c3a8e03f4
4	ross95	maria	sanders	Male	62f0a68a41
5	paul85	morris	miller	Female	61bd060b07
6	smith34	daniel	michael	Female	7055b3d9f5
7	james84	sanders	paul	Female	b7f72d6eb9
8	daniel53	mark	mike	Male	299cbf7171
9	brooks80	morgan	maria	Female	aa736a35dc
10	morgan65	paul	miller	Female	a28dca31f5

An "OK" button is visible at the bottom of the preview dialog.

# Data Hub – Storage accounts

See basic file properties

The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a navigation pane with sections for Data, Storage accounts (1 item), Databases (3 items), and Datasets (4 items). Under Storage accounts, 'internalsandboxwe (Primary)' is expanded, showing containers like holidaydatacontainer, isdweatherdatacontainer, nyctlc, opendataset, tempdata (which is selected and highlighted in blue), and yasofian. The 'tempdata' container is selected in the main workspace, showing a list of files including 'SampleCSVFile\_2kb.csv'. A context menu is open over this file, listing options: Preview, New notebook, Copy ABFSS path, Manage Access..., Rename..., Download, Delete, and Properties... (the last option is highlighted with a red box).

A 'Properties' dialog box is open, showing system properties for the selected file. The 'Name' field is 'New folder/SampleCSVFile\_2kb.csv'. The 'URL' field is 'https://internalsandboxwe.dfs.core.windows.net/tempdata/New folder/SampleCSVF...'. The 'LastModified' field is '2020-03-13T23:51:02.000Z'. The 'ContentType' field is 'application/vnd.ms-excel'. The 'ContentDisposition' field is empty. The 'ContentEncoding' field is empty. The 'ContentLanguage' field is empty. There is a section for 'User Properties' which is currently empty. At the bottom, there are 'Save' and 'Cancel' buttons.

System Properties	
Name	New folder/SampleCSVFile_2kb.csv
URL	https://internalsandboxwe.dfs.core.windows.net/tempdata/New folder/SampleCSVF...
LastModified	2020-03-13T23:51:02.000Z
CacheControl	
ContentType	application/vnd.ms-excel
ContentDisposition	
ContentEncoding	
ContentLanguage	
User Properties	

# Data Hub – Storage accounts

Manage Access - Configure standard POSIX ACLs on files and folders

The screenshot shows the Azure Synapse Analytics Studio Data Hub. On the left, there's a sidebar with a search bar and sections for Storage accounts, Databases, and Datasets. Under Storage accounts, 'internalsandboxwe (Primary)' is expanded, showing containers like 'holidaydatacontainer', 'isdweatherdatacontainer', 'nyctlc', 'opendataset', and 'tempdata'. The 'tempdata' container is selected and highlighted in blue. On the right, a detailed view of the 'tempdata' container shows a file named 'SampleCSVFile\_2kb.csv'. A context menu is open over this file, listing options: Preview, New notebook, Copy ABFSS path, Manage Access..., Rename..., Download, Delete, and Properties... . The 'Manage Access...' option is highlighted with a red box and has a red arrow pointing to the corresponding section in the 'Manage Access' dialog.

The screenshot shows the 'Manage Access' dialog for the file 'SampleCSVFile\_2kb.csv'. The title bar says 'Manage Access'. It lists 'Users, groups, and service principals:' with '\$superuser (Owner)' and '\$superuser (Owning Group)' selected. There are buttons for 'Edit' and 'Delete' next to each entry. Below this is a section for 'Permissions for: \$superuser' with checkboxes for 'Read', 'Write', and 'Execute'. The 'Access' checkbox is checked. At the bottom, there's a section to 'Add user, group, or service principal:' with a text input field and an 'Add' button. At the very bottom are 'Save' and 'Cancel' buttons.

# Data Hub – Storage accounts

Two simple gestures to start analyzing with SQL scripts or with notebooks

## T-SQL or PySpark auto-generated

NAME	
_SUCCESS	
part-00000-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet	New SQL script and open in new tab
part-00000-e3d623d5	
part-00001-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet	New SQL script > Select TOP 100 rows
part-00001-e3d623d	
part-00002-7bc127b	New notebook
part-00002-e3d623d	Create external table
part-00003-7bc127b	
part-00003-e3d623d	
part-00004-7bc127b	
part-00004-e3d623d	
part-00005-7bc127b	
part-00005-e3d623d	
part-00006-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet	
part-00006-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet	

The screenshot shows a Jupyter Notebook interface with two cells. The top cell contains a SQL query to read data from a Parquet file stored in Azure Blob Storage. The results are displayed as a table with columns: USAF, WBAN, DATETIME, LATITUDE, LONGITUDE, and ELEVATION. The bottom cell contains PySpark code to read the same data from a different location and show the first 100 rows.

```
1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/ISDWeatherData.parquet',
6         FORMAT='PARQUET'
7     ) AS [r];
8
```

USAF	WBAN	DATETIME	LATITUDE	LONGITUDE	ELEVATION
999999	04222	2019-05-01T00:2...	40.651	-122.607	432
999999	04222	2019-05-01T01:3...	40.651	-122.607	432
999999	04222	2019-05-01T02:4...	40.651	-122.607	432
999999	04222	2019-05-01T03:5...	40.651	-122.607	432
999999	04222	2019-05-01T05:0...	40.651	-122.607	432
999999	04222	2019-05-01T06:1...	40.651	-122.607	432

```
[3] 1 %%pyspark
2 data_path = spark.read.load('abfss://opendataset@internalsandboxwe.dfs.core.windows.net/isdweatherdata.parquet')
3 data_path.show(100)
```

usaf	wban	datetime	latitude	longitude	elevation	windAngle	windSpeed	temperature	seaLvlPressure	cloudCoverage	presentWeatherIndicator	pastWeatherIndicator	precipTime	precipDepth	snowDepth	stationName	countryOrRegion	year	day	month
999999	04222	2019-05-01 00:00:00	40.651	-122.607	432.0	null	2.1	19.8	1010.2	Cloudy	Present	Past	2019-05-01 00:00:00	0.0	0.0	REDDING 12 WNW	US	2019	1	5
999999	04222	2019-05-01 00:05:00	40.651	-122.607	432.0	null	2.1	19.7	1010.2	Cloudy	Present	Past	2019-05-01 00:05:00	0.0	0.0	REDDING 12 WNW	US	2019	1	5
999999	04222	2019-05-01 00:10:00	40.651	-122.607	432.0	null	2.1	19.5	1010.2	Cloudy	Present	Past	2019-05-01 00:10:00	0.0	0.0	REDDING 12 WNW	US	2019	1	5
999999	04222	2019-05-01 00:15:00	40.651	-122.607	432.0	null	2.1	19.2	1010.2	Cloudy	Present	Past	2019-05-01 00:15:00	0.0	0.0	REDDING 12 WNW	US	2019	1	5

# Data Hub – Storage accounts

Generate scripts to create external tables

synapsecontainer > opendataset > ISDWeathercurated

**NAME**

- \_SUCCESS
- part-00000-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00000-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00001-7bc127b8-5e9fdad7-c000.snappy.parquet
- part-00001-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00002-7bc127b8-5e9fdad7-c000.snappy.parquet
- part-00002-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00003-7bc127b8-5e9fdad7-c000.snappy.parquet
- part-00003-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00004-7bc127b8-5e9fdad7-c000.snappy.parquet
- part-00004-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00005-7bc127b8-5e9fdad7-c000.snappy.parquet
- part-00005-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00006-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00006-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet

Create external table

External tables provide a convenient way to persist the schema of data residing in your data lake which can be reused for future adhoc analytics. [Learn more](#)

Source file format  
Parquet

Select SQL pool \*  DefSQLPool

Select a database \*  DefSQLPool

External table name \*  dbo.TestData

Checking this box will automatically create an external table when opening script

**Open script** **Cancel**

```

1 IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name =
2 CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
3 WITH ( FORMAT_TYPE = PARQUET)
4 GO
5 IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name =
6 CREATE EXTERNAL DATA SOURCE [opendataset_internalsandboxwe_dfs_core_
7 WITH
8 ( LOCATION   = 'abfss://opendataset@internalsandboxwe.dfs.core.wi
9          TYPE      = HADOOP
10 )
11 GO
12 CREATE EXTERNAL TABLE dbo.TestData
13 (
14     [usaf] varchar(max),
15     [wban] varchar(max),
16     [datetime] datetime2,
17     [latitude] float,
18     [longitude] float,
19     [elevation] float,
20     [windAngle] int,
21     [windSpeed] float,
22     [temperature] float,
23     [seaLvlPressure] float,
24     [cloudCoverage] varchar(max),
25     [presentWeatherIndicator] int,
26     [pastWeatherIndicator] int,
27     [precipTime] float,

```

# Data Hub – Storage accounts

Create external tables with simple right click on files in storage account

synapsecontainer > opendataset > ISDWeathercurated

NAME

- \_SUCCESS
- part-00000-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00000-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00001-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet**
- part-00001-e3d623d
- part-00002-7bc127b
- part-00002-e3d623d
- part-00003-7bc127b
- part-00003-e3d623d
- part-00004-7bc127b
- part-00004-e3d623d
- part-00005-7bc127b
- part-00005-e3d623d
- part-00006-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00006-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet

New SQL script and open in new tab  
New notebook  
Select TOP 100 rows  
Create external table  
Copy ABFSS path  
Manage Access...  
Rename...  
Download  
Delete  
Properties...  
part-00006-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet

**Create external table**

External tables provide a convenient way to persist the schema of data residing in your data lake which can be reused for future adhoc analytics. [Learn more](#)

Source file format  
Parquet

Select SQL pool \*  DefSQLPool

Select a database \*  DefSQLPool

External table name \*  dbo.TestData

Checking this box will automatically create an external table when opening script

**Open script** **Cancel**

Data

Filter resources by name

- DefSQLPool (SQL pool)
  - Tables
  - External tables
    - dbo.TestData**
    - ext T100\_packet.tcn

Run Publish Query plan ...

```
1 SELECT TOP 100 * FROM dbo.TestData
2 Go
```

# Data Hub – Storage accounts

SQL Script from Multiple files

Multi-select of files generates a SQL script that analyzes all those files together

The screenshot shows the Azure Synapse Analytics Studio interface. On the left, there is a file browser window titled "NAME" containing a list of parquet files. A red arrow points from the "New SQL script" option in the context menu of one of the files to the query editor on the right. The query editor contains a T-SQL script for reading multiple parquet files from a blob storage container.

NAME

- \_SUCCESS
- part-00000-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00000-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00001-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00001-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00002-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00002-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00003-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00003-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00004-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00004-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00005-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00005-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet
- part-00006-7bc127b8-56a2-443f-b6f9-18175e9fdad7-c000.snappy.parquet
- part-00006-e3d623d5-1ecd-4953-bcf9-ad973c404bd1-c000.snappy.parquet

New SQL script > Select TOP 100 rows snappy.parquet

New notebook

Copy ABFSS path

Delete

Run Publish Query plan Connect to SQL on-demand Use database master

```
1 -- Read multiple parquet files with same schema
2 SELECT
3     TOP 100 *
4 FROM
5     OPENROWSET(
6         BULK 'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
7         FORMAT = 'Parquet'
8     ) AS [r]
9 WHERE
10    r.filepath() in (
11        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
12        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
13        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
14        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
15        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
16        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
17        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
18        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
19        'https://internalsandboxwe.dfs.core.windows.net/opendataset/isdweatherdatacontainer/',
20    )
```

# Data Hub – Databases

Explore the different kinds of databases that exist in a workspace.

SQL Provisioned

SQL Serverless

Spark

The screenshot shows the left side of the Azure Synapse Studio interface. It features a navigation pane with the following structure:

- Data**: The main title.
- Storage accounts**: 2 items.
- Databases**: 3 items.
  - sql1 (SQL pool)**
  - sample (SQL on-demand)**
  - default (Spark)**
    - Tables**:
      - nyxtaxiyellow7days**
      - searchlogtable**
- Datasets**: 2 items.

The screenshot shows the right side of the Azure Synapse Studio interface. It displays the detailed structure of the selected database, with red arrows pointing from specific items in the left pane to their corresponding details in the right pane:

- Storage accounts**: 2 items.
- Databases**: 3 items.
  - sql1 (SQL pool)**
    - Tables**
    - External tables**
    - External resources**
    - Views**
    - Programmability**
    - Schemas**
    - Security**
  - sample (SQL on-demand)**
    - External tables**
    - External resources**
    - Views**
    - Schemas**
    - Security**
  - default (Spark)**
    - Tables**
- Datasets**: 2 items.

# Data Hub – Databases

Familiar gesture to generate T-SQL scripts from SQL metadata objects such as tables.

Databases 3

- sql1 (SQL pool)
- Tables
  - dbo.SearchLogTable
  - dbo.NycTaxiPredict
    - Columns
      - New SQL script
      - New notebook
      - Select TOP 1000 rows
      - CREATE
      - Drop
      - Drop and Create

Starting from a table, auto-generate a single line of PySpark code that makes it easy to load a SQL table into a Spark dataframe

Databases 3

- sql1 (SQL pool)
  - Tables
    - dbo.SearchLogTable
    - dbo.NycTaxiPredict
      - Columns
        - New SQL script
        - New notebook
        - Refresh
        - Load to DataFrame

Notebook 1 \* X

+ Cell ▾ Run all Publish Attach to Select Spark pool ▾ Language PySpark (Python) ▾

Cell 1

```
[ ] 1 val df = spark.read.sqlanalytics("sql1.dbo.NycTaxiPredict")
```

# Data Hub – Datasets

Orchestration datasets describe data that is persisted. Once a dataset is defined, it can be used in pipelines and sources of data or as sinks of data.

The screenshot shows the Azure Synapse Analytics Studio interface for managing datasets. On the left, a sidebar titled 'Data' lists resources: Storage accounts (2), Databases (3), and Datasets (2). The 'NYCTaxiParquet' dataset is highlighted with a red box and has a red arrow pointing from the sidebar to its configuration page on the right. The main area displays the 'NYCTaxiParquet' dataset details, including its Parquet file icon and name. The configuration page includes tabs for General, Connection, Schema, and Parameters. Under the Connection tab, the linked service is set to 'Lake\_ArcadiaLake'. The File path is specified as 'data / nyctaxi / File'. The Compression type is set to 'snappy'. There are also buttons for Test connection, Open, New, Browse, and Preview data.



# Synapse Studio

## Develop hub

# Develop Hub

## Overview

It provides development experience to query, analyze, model data

## Benefits

Multiple languages to analyze data under one umbrella

Switch over notebooks and scripts without loosing content

Code intellisense offers reliable code development

Create insightful visualizations

Microsoft Azure | Synapse Analytics > p Search resources

Home Data Develop + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

Filter resources by name

SQL scripts	27
Notebooks	11
Data flows	2
Spark job definitions	1
Power BI	1

prlangad@microsoft.com MICROSOFT

# Develop Hub - SQL scripts

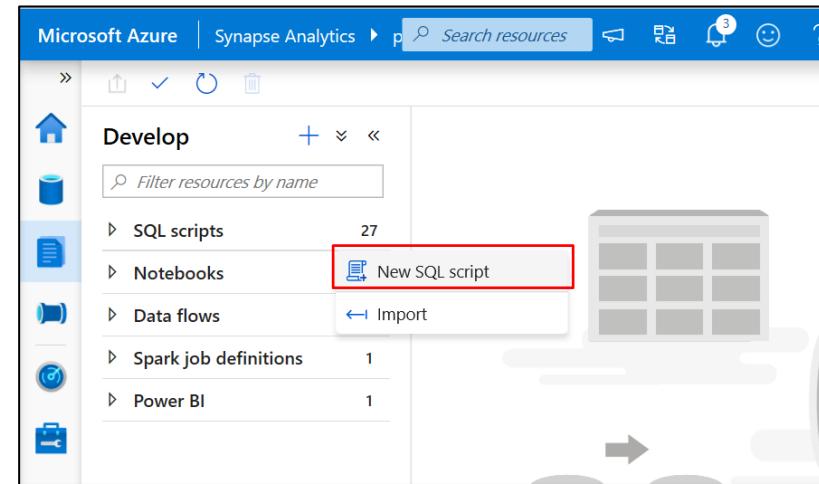
## SQL Script

### Authoring SQL Scripts

Execute SQL script on SQL provisioned or SQL serverless

Publish individual SQL script or multiple SQL scripts through Publish all feature

Language support and intellisense

A screenshot of the Microsoft Azure Synapse Analytics Develop Hub showing an open SQL script editor. The script content is: 1 -- type your sql script here, we now have intellisense 2 CREATE. In the bottom right corner, there is an 'intellisense' suggestion box with three items: 'CREATE', 'CURRENT\_TIMESTAMP', and 'CURRENT\_USER'. The user's email 'prlangad@microsoft.com' and Microsoft logo are visible in the top right corner of the interface.

# Develop Hub - SQL scripts

## SQL Script

View results in Table or Chart form and export results in several popular formats

The screenshot shows the Azure Synapse Analytics Develop Hub interface. On the left, a code editor displays a T-SQL script for reading CSV data from a blob storage location and creating a temporary table named 'searchlog'. The script uses OPENROWSET and BULK options to read the data.

Below the code editor, the results pane is visible. It includes tabs for 'Results' (highlighted with a red box) and 'Messages'. Under 'Results', there are 'Table' and 'Chart' views. The 'Table' view shows a sample of the log data with columns ID, TIME, and REGION. The 'Chart' view shows a line chart of the data over time, with the Y-axis ranging from 0 to 1,000k and the X-axis from 0 to 22. The chart legend includes id, time, region, searchtext, latency, links, and clickedlinks.

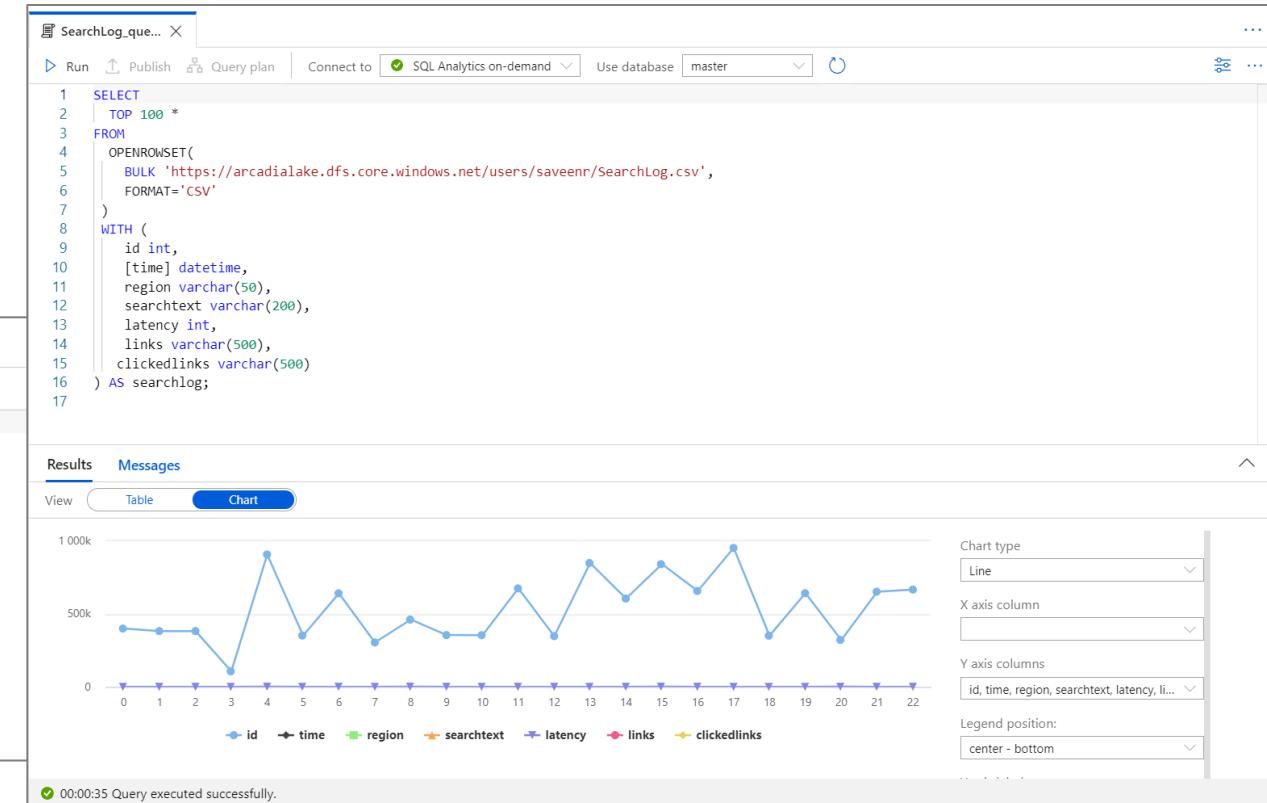
On the right, a separate results pane shows the same data in a 'Table' view. A red box highlights the 'Export results' button, which is part of a dropdown menu listing CSV, Excel, JSON, and XML options.

```

1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://arcadialake.dfs.core.windows.net/users/saveenr/SearchLog.csv',
6         FORMAT='CSV'
7     )
8     WITH (
9         id int,
10        [time] datetime,
11        region varchar(50),
12        searchtext varchar(200),
13        latency int,
14        links varchar(500),
15        clickedlinks varchar(500)
16    ) AS searchlog;
17
  
```

ID	TIME	REGION
399266	2019-10-15T11:53:04.0000000	en-us
382045	2019-10-15T11:53:25.0000000	en-gb
382045	2019-10-16T11:53:42.0000000	en-gb
106479	2019-10-16T11:53:10.0000000	en-ca
906441	2019-10-16T11:54:18.0000000	en-us

00:00:35 Query executed successfully.



# Develop Hub - Notebooks

## Notebooks

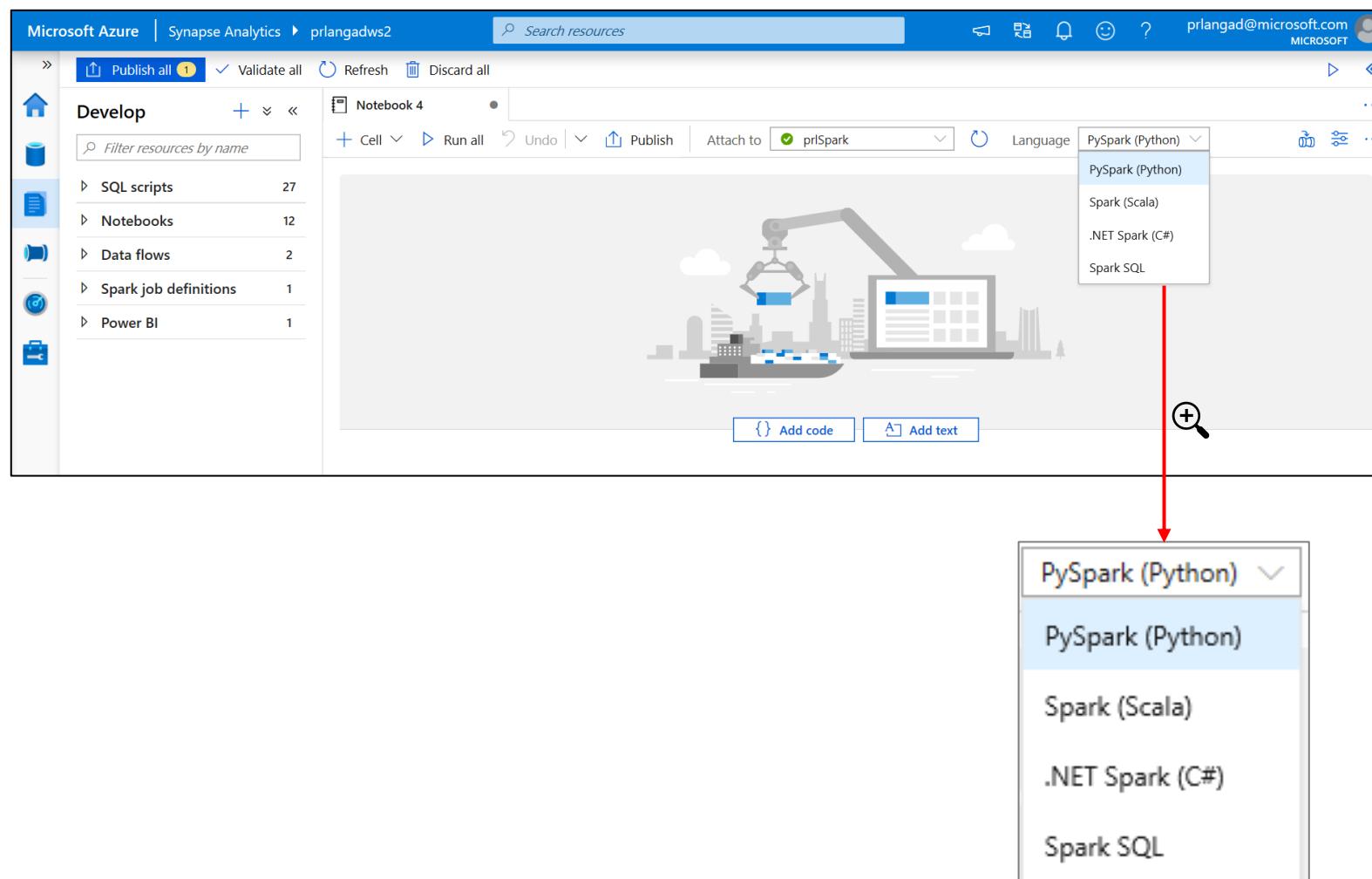
Allows to write multiple languages in one notebook

`%%<Name of language>`

Offers use of temporary tables across languages

Language support for Syntax highlight, syntax error, syntax code completion, smart indent, code folding

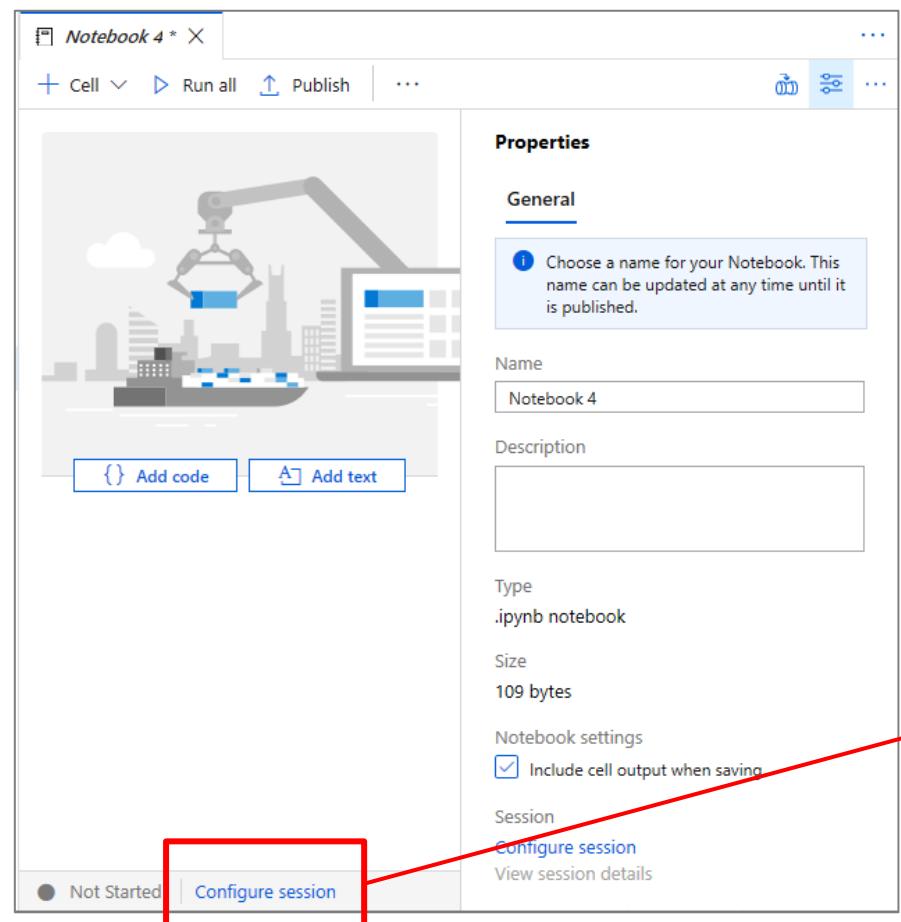
Export results



# Develop Hub - Notebooks

Configure session allows developers to control how many resources are devoted to running their notebook.

Provides quick links to monitor session and Spark history server



## Configure session

Session name  
Synapse\_prlSpark\_1584564592131  
Application ID  
application\_1584564653137\_0001  
Livy session ID  
8  
[View in monitoring](#)  
[Spark history server](#)

Status  
Ready

Session timeout \* ⓘ  
30

Executors \* ⓘ  
2

Executor size \* ⓘ  
Small (4 vCPU, 28GB memory)

Driver size \* ⓘ  
Small (4 vCPU, 28GB memory)

Apply     Cancel

# Develop Hub - Notebooks

As notebook cells run, the underlying Spark application status is shown. Providing immediate feedback and progress tracking.

The screenshot shows the Microsoft Azure Synapse Analytics Develop Hub - Notebooks interface. At the top, there's a navigation bar with 'Microsoft Azure' and 'Synapse Analytics'. A search bar says 'Search resources' and there are icons for notifications and help. On the right, it shows the email 'prlangad@microsoft.com' and the 'MICROSOFT' logo.

The main area has a toolbar with 'Publish all', 'Validate all', 'Refresh', and 'Discard all'. Below that, a breadcrumb trail shows 'opendataset' and 'Notebook 1'. The toolbar also includes 'Cell', 'Run all', 'Undo', 'Publish', 'Attach to' (set to 'SparkPoolDef'), 'Language' (set to 'PySpark (Python)'), and other options.

The notebook content starts with 'Cell 1' containing the following Python code:

```
%pyspark
data_path = spark.read.load('abfss://opendata@internalsandboxwe.dfs.core.windows.net/holidays/part-00000-bd1ab'
                            ...
data_path.show(100)
```

Below the code, a message says 'Command executed in 2mins 44s 998ms by prlangad on 03-19-2020 11:31:56.458 -07:00'.

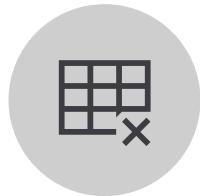
A section titled 'Job execution Succeeded' shows 'Spark 2 executors 8 cores' with three jobs listed:

ID	DESCRIPTION	STATUS	STAGES	TASKS	SUBMISSION TIME	DURATION
▶ Job 0	load at NativeMethodAccessImpl.java:0	<span style="color: green;">✓ Succeeded</span>	1/1	<div style="width: 100%; background-color: #2e7131;"></div>	3/19/2020, 11:31:35 AM	6s
▶ Job 1	showString at NativeMethodAccessImpl.java:0	<span style="color: green;">✓ Succeeded</span>	1/1	<div style="width: 100%; background-color: #2e7131;"></div>	3/19/2020, 11:31:43 AM	1s
▶ Job 2	showString at NativeMethodAccessImpl.java:0	<span style="color: green;">✓ Succeeded</span>	1/1	<div style="width: 100%; background-color: #2e7131;"></div>	3/19/2020, 11:31:45 AM	9s

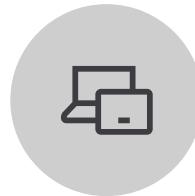
Below the table, there's a preview of the data being processed, showing columns like vendorID, tpepPickupDateTime, tpepDropoffDateTime, passengerCount, tripDistance, puLocationId, doLocationId, startLon, startLat, endLon, endLat, rateCodeId, storeAndFwdFlag, paymentType, fareAmount, extra, mtaTax, improvementSurcharge, tipAmount, tollsAmount, and totalAmount. The data is presented in a tabular format with some rows partially visible.

At the bottom, there's a footer with a checkmark icon, the text 'Ready (Stop session)', and 'Configure session'.

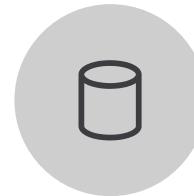
# Dataflow Capabilities



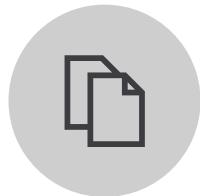
Handle upserts, updates, deletes on sql sinks



Add new partition methods



Add schema drift support



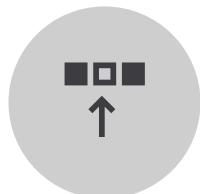
Add file handling (move files after read, write files to file names described in rows etc)



New inventory of functions (for e.g. Hash functions for row comparison)



Commonly used ETL patterns(Sequence generator/Lookup transformation/SCD...)



Data lineage – Capturing sink column lineage & impact analysis(invaluable if this is for enterprise deployment)

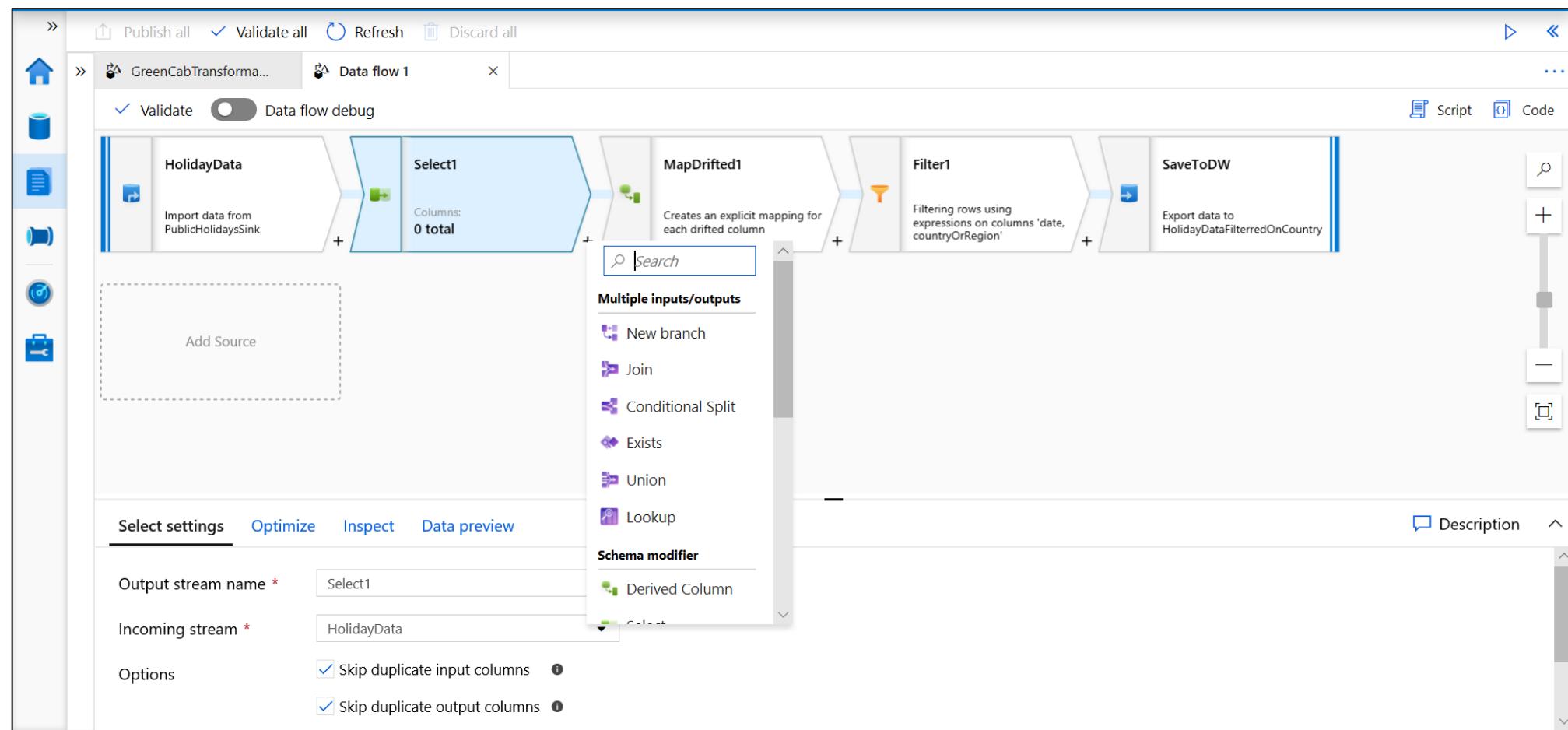


Implement commonly used ETL patterns as templates(SCD Type1, Type2, Data Vault)

# Develop Hub - Data Flows

Data flows are a visual way of specifying how to transform data.

Provides a code-free experience.



# Develop Hub – Power BI

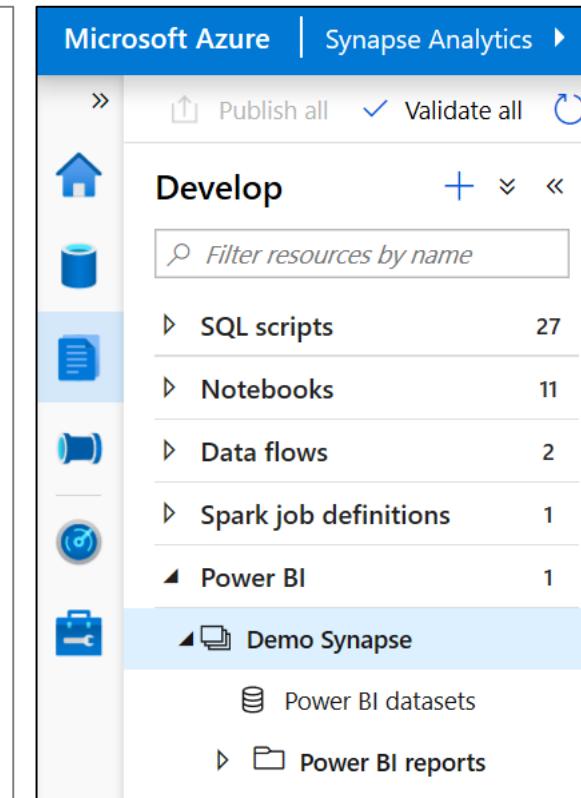
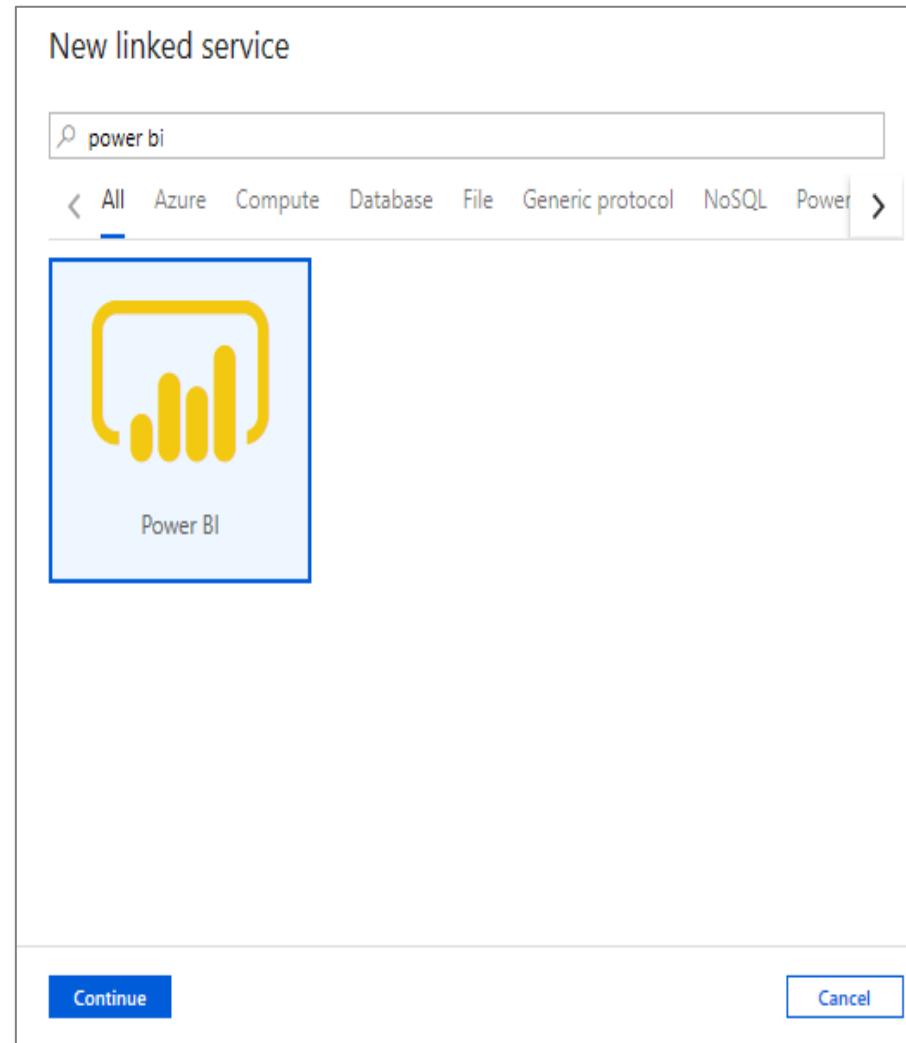
## Overview

Create Power BI reports in the workspace

Provides access to published reports in the workspace

Update reports real time from Synapse workspace to get it reflected on Power BI service

Visually explore and analyze data



# Develop Hub – Power BI

View published reports in Power BI workspace

The screenshot shows the Microsoft Azure Power BI workspace interface. The left sidebar displays the 'Develop' section with a list of resources: SQL scripts (9), Notebooks (6), Data flows (1), Power BI (1), and a folder named 'gaming-telemetry' containing Power BI datasets and reports. The 'Report' item under 'Power BI reports' is selected.

The main content area displays a report titled 'GAME STUDIO'. The report features a header with a game console and controller image, followed by a 'What If...' section asking 'We increase free game addons by: 1'. It includes two large callout boxes: 'Users (Forecast) 7,361,707' (7,346,291 last month) and 'Extra Users 252.8K' (+3.4% Users Increase). Below these are two visualizations: a 'Total Users vs "What If" Analysis' table and a 'What If' Analysis Forecast' line chart.

The 'Total Users vs "What If" Analysis' table provides detailed user data by region and age group:

Region	Users	Forecast	Extra Users
APAC	1,268.5K	1,273.7K	45.4K
18-22	96.8K	97.7K	4.0K
22-26	436.0K	435.5K	13.4K
26-30	462.9K	464.0K	15.6K
30-34	75.0K	76.3K	3.4K
34-40	24.0K	24.2K	1.1K
41-60	27.1K	27.5K	1.3K
>60	146.7K	148.5K	6.7K
EMEA	844.9K	846.5K	30.4K
18-22	66.8K	67.5K	2.7K
22-26	291.8K	290.7K	9.1K
26-30	306.9K	307.1K	10.4K
30-34	50.4K	50.9K	2.3K
34-40	16.3K	16.4K	0.7K
41-60	18.5K	18.7K	0.9K
Total	7,346.3K	7,361.7K	252.8K

The 'What If' Analysis Forecast' chart shows the projected growth of users over time from August 2019 to November 2019, with a current total of 24.5M users.

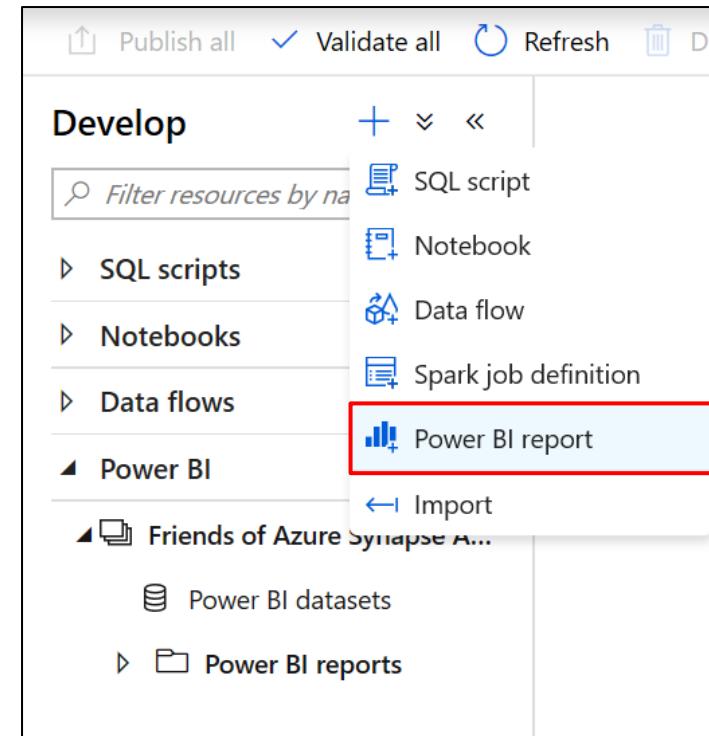
The right sidebar contains sections for 'VISUALIZATIONS' and 'FIELDS', along with filter and drill-through options.

# Develop Hub – Power BI

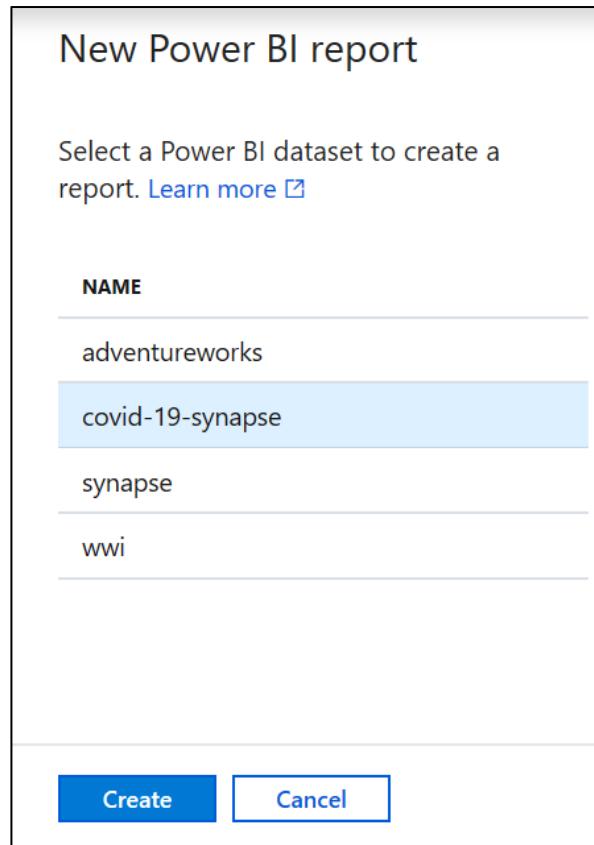
Create new reports from existing published Power BI datasets

Create new Power BI datasets

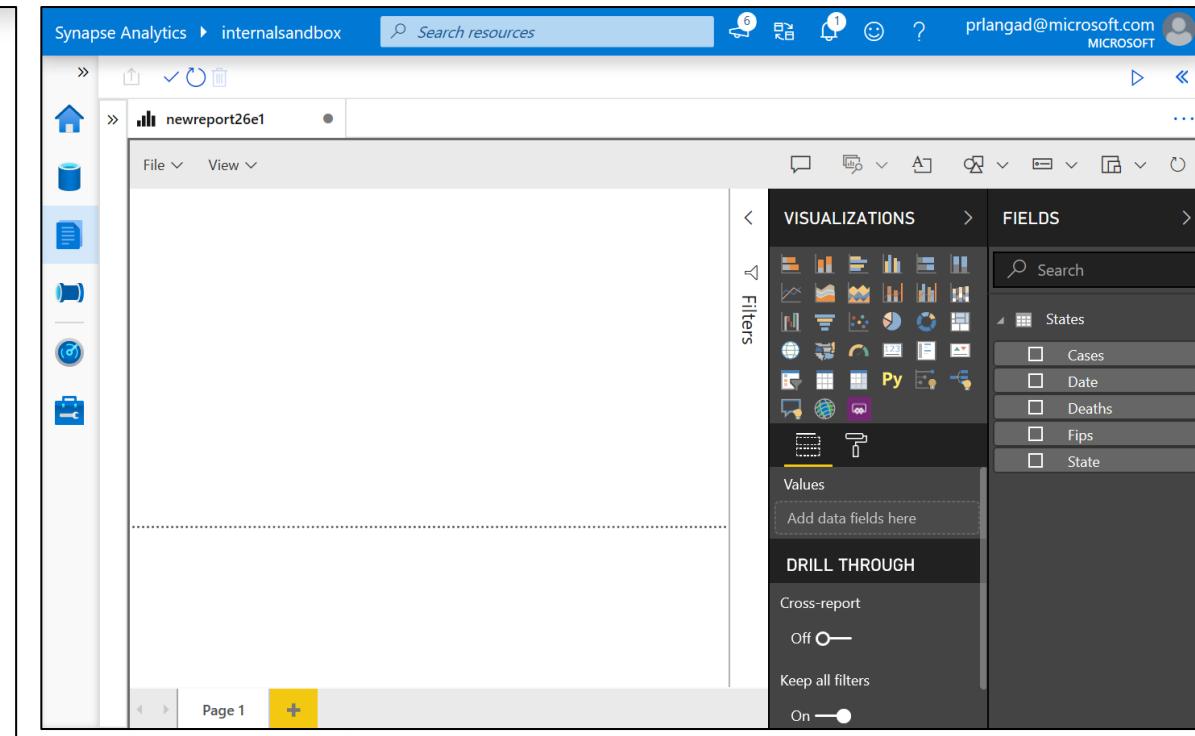
1



2



3



# Develop Hub – Power BI

Edit reports in Synapse workspace

The screenshot shows the Microsoft Azure Synapse Analytics Develop Hub interface. On the left, the navigation pane is open under the 'Develop' section, showing options like SQL scripts, Notebooks, Data flows, and Power BI. Under Power BI, 'gaming-telemetry' is selected, and 'Report' is highlighted. The main area displays a Power BI report titled 'GAME STUDIO'. The report features a header with a game console image and a 'Total Users' summary of 24.5M. It includes a 'What If...' card where users can increase free game addons by 1, showing a forecast of 7,361,707 users and an extra 252.8K users. Below this is a table comparing 'Users' and 'Forecast' for regions APAC and EMEA across age groups. To the right is a line chart titled 'What If' Analysis Forecast' showing user growth from August 2019 to November 2019. The bottom of the report has tabs for 'Historical', 'Forecast', and 'Predictions'. The right side of the screen shows the 'Visualizations' and 'Fields' panes, which contain various data models and fields related to the report's data source.

**GAME STUDIO**

**What If...**  
We increase free game addons by:  
1

**Total Users vs "What If" Analysis**

Region	Users	Forecast	Extra Users
APAC	1,268.5K	1,273.7K	45.4K
18-22	96.8K	97.7K	4.0K
22-26	436.0K	435.5K	13.4K
26-30	462.9K	464.0K	15.6K
30-34	75.0K	76.3K	3.4K
34-40	24.0K	24.2K	1.1K
41-60	27.1K	27.5K	1.3K
>60	146.7K	148.5K	6.7K
EMEA	844.9K	846.5K	30.4K
18-22	66.8K	67.5K	2.7K
22-26	291.8K	290.7K	9.1K
26-30	306.9K	307.1K	10.4K
30-34	50.4K	50.9K	2.3K
34-40	16.3K	16.4K	0.7K
41-60	18.5K	18.7K	0.9K
<b>Total</b>	<b>7,346.3K</b>	<b>7,361.7K</b>	<b>252.8K</b>

**"What If" Analysis Forecast**

Users Forecast

Date

Historical Forecast Predictions

Visualizations Fields

# Develop Hub – Power BI

Publish edited reports in Synapse workspace to Power BI workspace

The screenshot shows the Microsoft Azure Synapse Analytics Develop Hub interface. On the left, a sidebar lists resources under 'Develop': SQL scripts (9), Notebooks (6), Data flows (1), Power BI (1), and Power BI datasets, Power BI reports, and Reports (selected). A red arrow points from the text 'Publish changes by simple save report in workspace' to the 'Save this report' button in the top navigation bar, which is highlighted with a red box.

The main area displays a Power BI report titled 'GAME STUDIO'. The report features a large image of an Xbox console and controller. Key statistics include 'Total Users 24.5M' and a 'What If...' analysis showing a forecast of 7,613,619 users. The 'VISUALIZATIONS' and 'FIELDS' panes on the right show various charts and data fields used in the report.

**Total Users vs "What If" Analysis**

Region	Users	Forecast	Extra Users
APAC	1,268.5K	1,319.0K	90.7K
18-22	96.8K	101.8K	8.1K
22-26	436.0K	448.9K	26.7K
26-30	462.9K	479.5K	31.1K
30-34	75.0K	79.7K	6.7K
34-40	24.0K	25.3K	2.2K
41-60	27.1K	28.8K	2.5K
>60	146.7K	155.1K	13.3K
EMEA	844.9K	876.7K	60.7K
18-22	66.8K	70.2K	5.5K
22-26	291.8K	299.6K	18.0K
26-30	306.9K	317.5K	20.9K
30-34	50.4K	53.2K	4.5K
34-40	16.3K	17.2K	1.5K
41-60	18.5K	19.6K	1.8K
Total	7,346.3K	7,613.6K	504.8K

**"What If" Analysis Forecast**

Users (Forecast) **7,613,619**  
7,346,291 Last month

Extra Users **504.8K**  
+6.9% Users Increase

Forecast Extra Users

Publish changes by simple save report in workspace



File ▾ View ▾ Edit report ▾ Explore ▾ Refresh ▾ Pin a live Page

 Reset to default

Comments

Bookmarks ▾

Usage metrics

View related

Favorite

Subscribe

Share



Home

Favorites

Recent

Apps

Shared with me

Workspaces

gaming-telemetry

Real-time publish on save



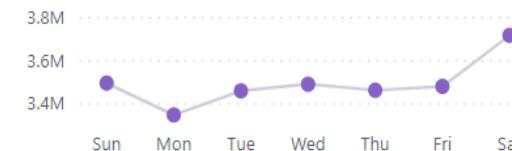
Historical

Forecast

Predictions

Select a Platform:

Console

Total Users  
**24.5M**

## GAME STUDIO

### What If...

We increase free game addons by:

2

Users (Forecast)

**7,613,619**7,346,291  
Last month

Extra Users

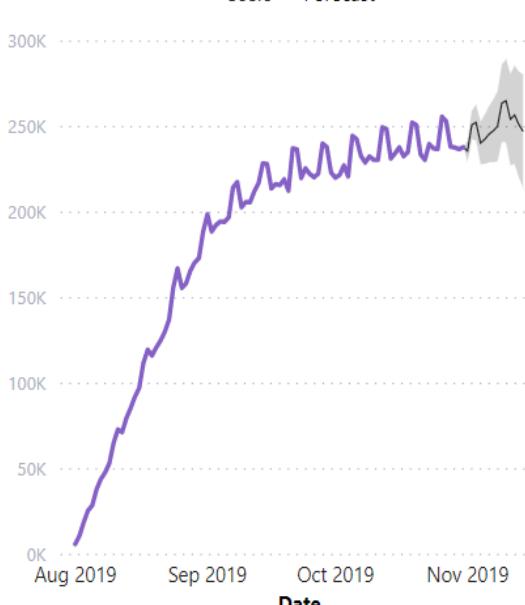
**504.8K**+6.9%  
Users Increase

### Total Users vs "What If" Analysis

Region	Users	Forecast	Extra Users
APAC	<b>1,268.5K</b>	<b>1,319.0K</b>	<b>90.7K</b>
	18-22 96.8K	101.8K	8.1K
	22-26 436.0K	448.9K	26.7K
	26-30 462.9K	479.5K	31.1K
	30-34 75.0K	79.7K	6.7K
	34-40 24.0K	25.3K	2.2K
	41-60 27.1K	28.8K	2.5K
EMEA	<b>844.9K</b>	<b>876.7K</b>	<b>60.7K</b>
	18-22 66.8K	70.2K	5.5K
	22-26 291.8K	299.6K	18.0K
	26-30 306.9K	317.5K	20.9K
	30-34 50.4K	53.2K	4.5K
	34-40 16.3K	17.2K	1.5K
	41-60 18.5K	19.6K	1.8K
Total	<b>7,346.3K</b>	<b>7,613.6K</b>	<b>504.8K</b>

### "What If" Analysis Forecast

● Users ● Forecast

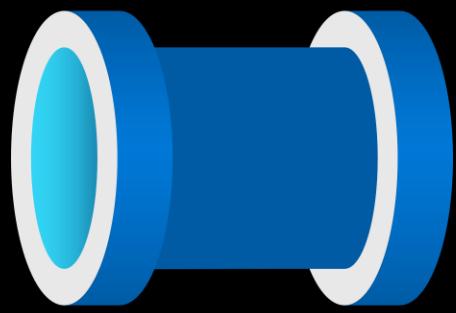


Tabular

Map

Forecast

Extra Users



# Synapse Studio **Orchestrate hub**

# Orchestrate Hub

It provides ability to create pipelines to ingest, transform and load data with 90+ inbuilt connectors.

Offers a wide range of activities that a pipeline can perform.

The screenshot shows the Azure Synapse Orchestrate Hub interface. On the left, there are three collapsed sections: 'Synapse' (Notebook, Spark job definition, Stored procedure), 'Move & transform' (Copy data, Data flow), and 'Machine Learning' (ML Batch Execution, ML Update Resource, ML Execute Pipeline). Red arrows point from each of these sections to the corresponding activity categories in the central 'Activities' pane. The 'Activities' pane is expanded, showing a list of 14 connector-based activities: Synapse, Move & transform, Azure Data Explorer, Azure Function, Batch Service, Databricks, Data Lake Analytics, General, HDInsight, Iteration & conditionals, and Machine Learning. To the right of the activities list, a pipeline editor window is open, showing two 'Copy data' activities: one for 'GreenTaxi' and one for 'YellowTaxi'. The pipeline editor has tabs for General, Parameters, Variables, and Output, with the General tab selected. The pipeline name is 'Copy Open Dataset'.



# Synapse Studio Monitor hub

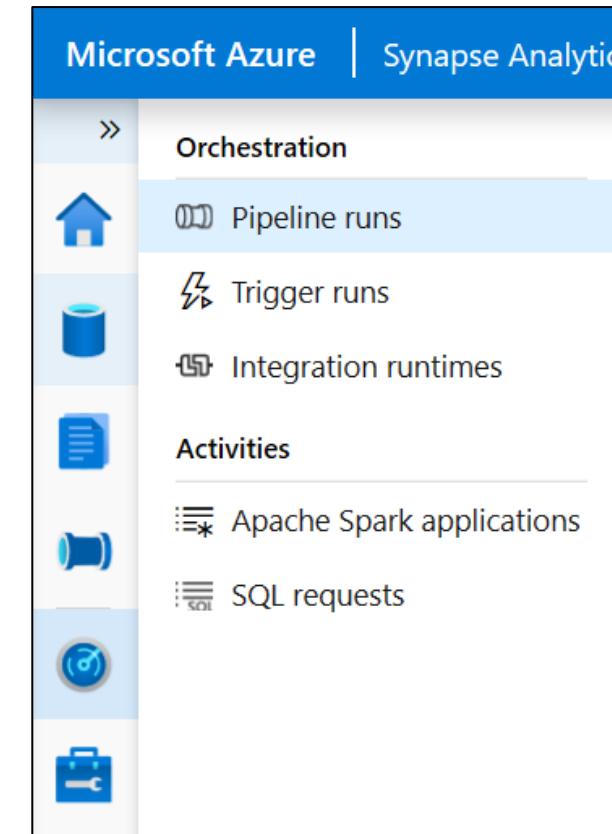
# Monitor Hub

## Overview

This feature provides single pane of glass to monitor orchestration, activities for Apache Spark Application and SQL requests.

## Benefits

Offers additional filters to monitor specific activities or orchestration



# Monitoring Hub - Orchestration

## Overview

Monitor orchestration in the Synapse workspace for the progress and status of pipeline

Pipeline runs				
Time : Last week (10/24/2019 9:44 AM - 10/31/2019 9:44 AM)		Time zone : Pacific Time (US & Canada) (UT...)		Runs : Latest runs
All status	Rerun	Cancel	Refresh	Edit columns
<input type="checkbox"/> PIPELINE NAME	RUN START ↑	DURATION	TRIGGERED BY	STATUS
<input type="checkbox"/> Load Data to SQLDW	10/25/2019, 3:49:42 PM	00:10:55	Manual trigger	<span style="color: green;">✓ Succeeded</span>
<input type="checkbox"/> Copy Open Dataset	10/25/2019, 2:17:54 PM	00:14:12	Manual trigger	<span style="color: green;">✓ Succeeded</span>
<input type="checkbox"/> Pipeline 1	10/24/2019, 1:23:43 PM	00:00:08	Manual trigger	<span style="color: green;">✓ Succeeded</span>

## Benefits

Track all/specific pipelines

Monitor pipeline run and activity run details

Find the root cause of pipeline failure or activity failure

Trigger runs			
Time : Last 30 days (3/17/20 11:48 PM - 4/16/20 11:48 PM)		Time zone : Pacific Time (US & Canada) (UT...)	
All status	Refresh	Edit columns	
Showing 1 - 1 items			
TRIGGER NAME	TRIGGER TYPE	TRIGGER TIME ↑	STATUS
TriggerCopy_csvdata100	ScheduleTrigger	4/10/20, 12:14:00 AM	<span style="color: green;">✓ Succeeded</span>

# Monitoring Hub – SQL requests

## Overview

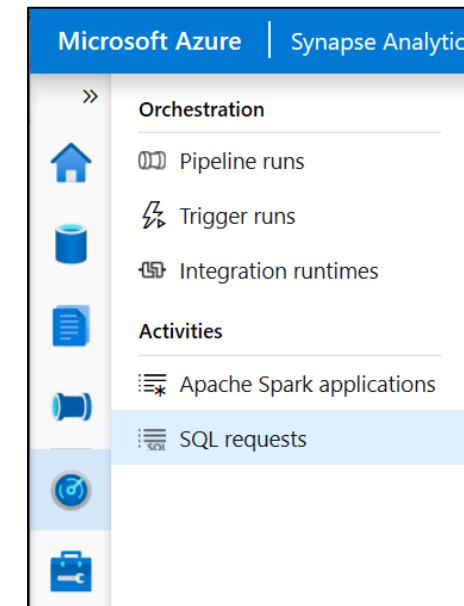
Monitor SQL requests for the progress and status of activities

## Benefits

Apply filter for pool to get SQL requests per compute pool

Additional available filters include

1. Start time
2. End time
3. Request ID
4. Session ID
5. Submitter
6. Workload group



The screenshot shows the main content area of the Microsoft Azure Synapse Analytics Monitoring Hub. The title bar indicates the user is viewing the 'internalsandboxwe5' workspace. The sidebar on the left is identical to the one in the previous screenshot. The main area displays a table titled 'SQL requests' with the following columns: STATUS, POOL, SUBMITTER, SESSION ID, SUBMIT TIME, and DURATION. A dropdown menu is open over the 'POOL' column, showing options like 'All status', 'End time', 'Request ID', 'Submitter', 'Session ID', and 'Workload group'. The table lists 1701 to 1744 of 1744 rows, all of which are 'Completed' status. The 'Workload group' column shows values like 'SQLPoolDef' and 'SQLProvider'. The 'SUBMIT TIME' column shows dates from March 20, 2020, at 02:22:17 AM to 02:22:58 AM. The 'DURATION' column shows times ranging from 10m 21s to 11m 5s.

STATUS	POOL	SUBMITTER	SESSION ID	SUBMIT TIME	DURATION
Completed	SQLPoolDef	prlangad@microsoft.com	SID366	03-20-20 02:22:58 AM	10m 40s
Completed	SQLPoolDef	prlangad@microsoft.com	SID366	03-20-20 02:22:54 AM	10m 44s
Completed	SQLPoolDef	prlangad@microsoft.com	SID365	03-20-20 02:22:34 AM	11m 4s
Completed	SQLPoolDef	prlangad@microsoft.com	SID365	03-20-20 02:22:33 AM	11m 5s
Completed	SQLPoolDef	prlangad@microsoft.com	SID364	03-20-20 02:22:17 AM	11m 21s

# Monitoring Hub - Spark applications

## Overview

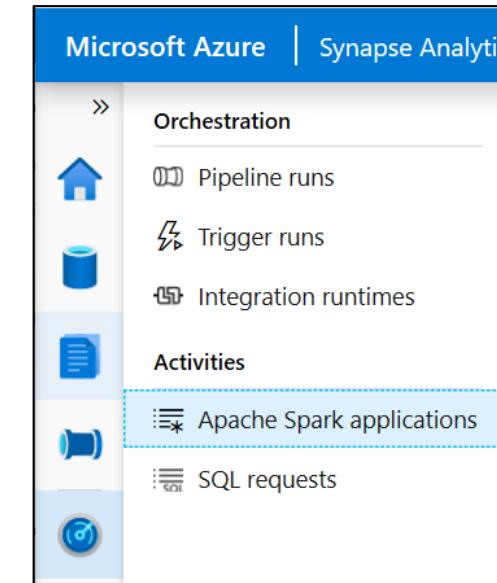
Monitor Spark pools, Spark applications for the status of activities

## Benefits

Apply filter for pool to get Apache Spark Applications per pool

Additional available filters include

1. Application Name
2. Livy ID
3. Status
4. End time



Microsoft Azure | Synapse Analytics > internalsandboxwe5

Orchestration

- Pipeline runs
- Trigger runs
- Integration runtimes

Activities

- Apache Spark applications** (highlighted)
- SQL requests

Apache Spark applications

Submit time : Last 30 days (2/18/20 12:23 PM ...)

Time zone : Pacific Time (US & Canada) (UT...)

All types Refresh Edit columns

Showing 1 - 17 of 17 items

APPLICATION NAME	SUBMITTER	SUBMIT TIME	STATUS	POOL
Synapse_SparkPoolDef_1584642555231	prlangad@microsoft.com	3/19/20, 11:29:15 AM	Cancelled	SparkPoolDef
Synapse_SparkPoolDef_1584603630854	bapule@microsoft.com	3/19/20, 12:40:31 AM	Cancelled	SparkPoolDef
Synapse_SparkPoolDef_1584565550012	prlangad@microsoft.com	3/18/20, 2:05:50 PM	Cancelled	SparkPoolDef
Synapse_SparkPoolDef_1584449792356	agchell@microsoft.com	3/17/20, 5:56:38 AM	Cancelled	SparkPoolDef

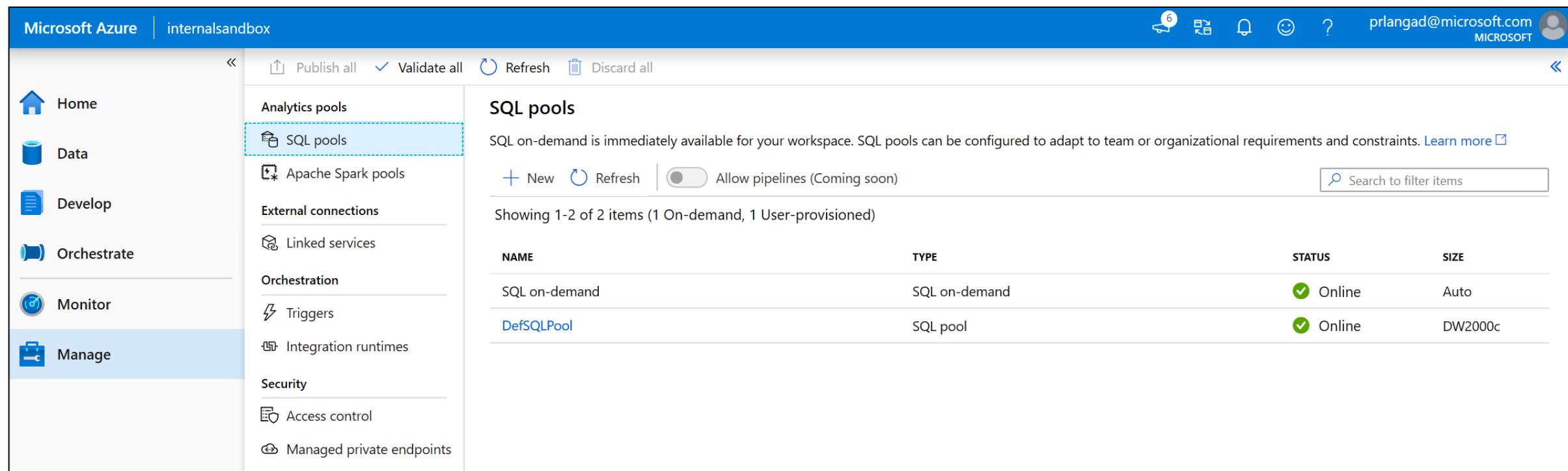


# Synapse Studio Manage hub

# Manage Hub

## Overview

This feature provides ability to manage Analytics pools, Linked Services, Orchestration and Security.



The screenshot shows the Microsoft Azure Manage Hub interface. The left sidebar has a 'Manage' section selected, which is highlighted with a blue background. The main content area is titled 'SQL pools'. It displays two items: 'SQL on-demand' (Type: SQL on-demand, Status: Online, Size: Auto) and 'DefSQLPool' (Type: SQL pool, Status: Online, Size: DW2000c). A search bar at the top right allows filtering of items. The top navigation bar includes options like 'Publish all', 'Validate all', 'Refresh', and 'Discard all'.

NAME	TYPE	STATUS	SIZE
SQL on-demand	SQL on-demand	Online	Auto
DefSQLPool	SQL pool	Online	DW2000c

# Manage – SQL pools

## Overview

Provides ability to Pause and Resume, change Scale, Assign Tags from Studio.

Microsoft Azure | internalsandbox

» Publish all ✓ Validate all ⏪ Refresh Discard all

**Analytics pools**

- SQL pools
- Apache Spark pools

**External connections**

- Linked services

**Orchestration**

- Triggers
- Integration runtimes

**Security**

- Access control
- Managed private endpoints

**SQL pools**

SQL on-demand is immediately available for your workspace. SQL pools can be provisioned on demand.

+ New ⏪ Refresh Allow pipelines (Coming soon)

Showing 1-2 of 2 items (1 On-demand, 1 User-provisioned)

NAME	TYPE
SQL on-demand	SQL on-demand
DefSQLPool	SQL pool

⋮ Open ⏪ Pause Scale

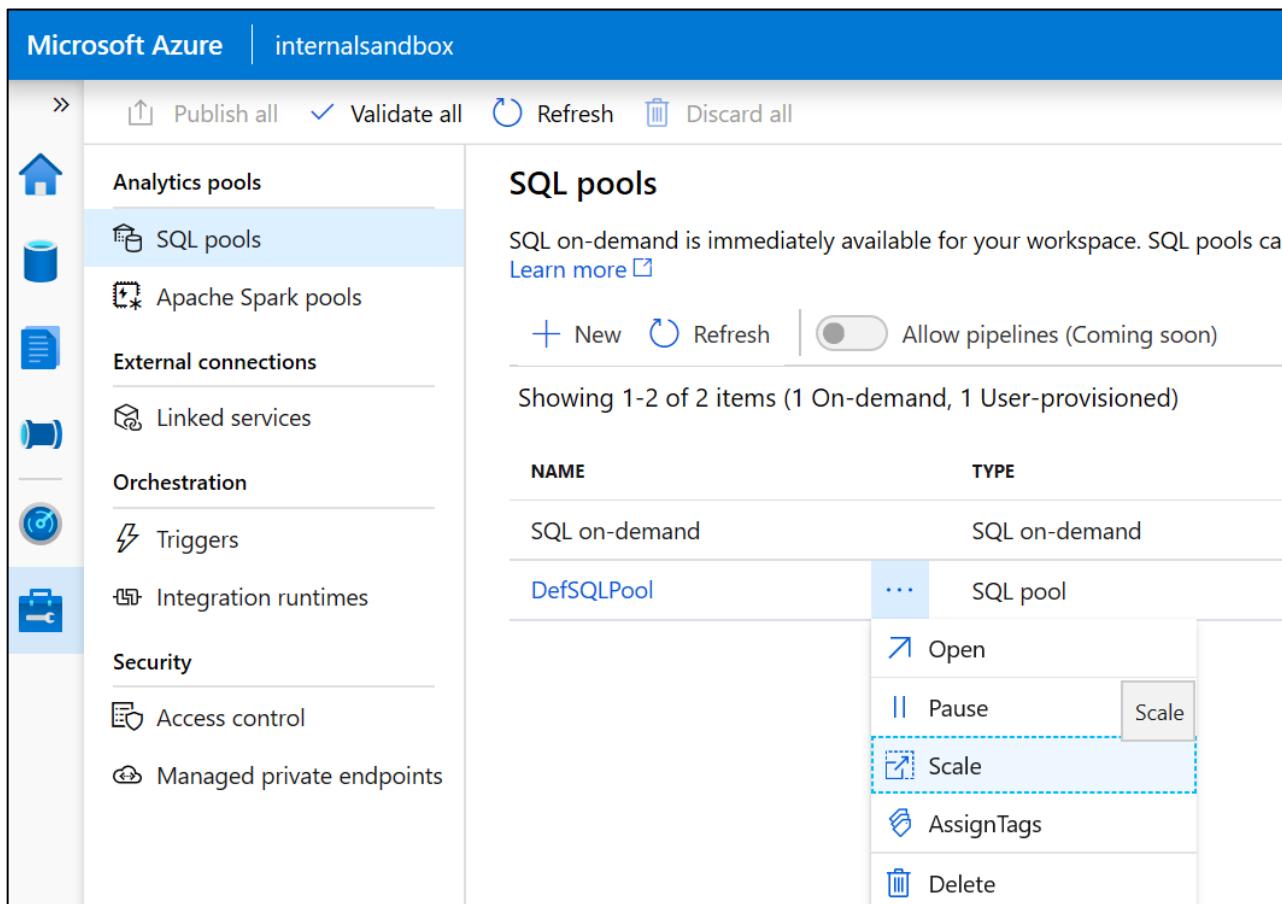
Scale

DefSQLPool

Configure the settings that best align to the workload needs on the SQL pool. [Learn more about performance levels](#)

Performance level

DW2000c



Scale

DefSQLPool

Configure the settings that best align to the workload needs on the SQL pool. [Learn more about performance levels](#)

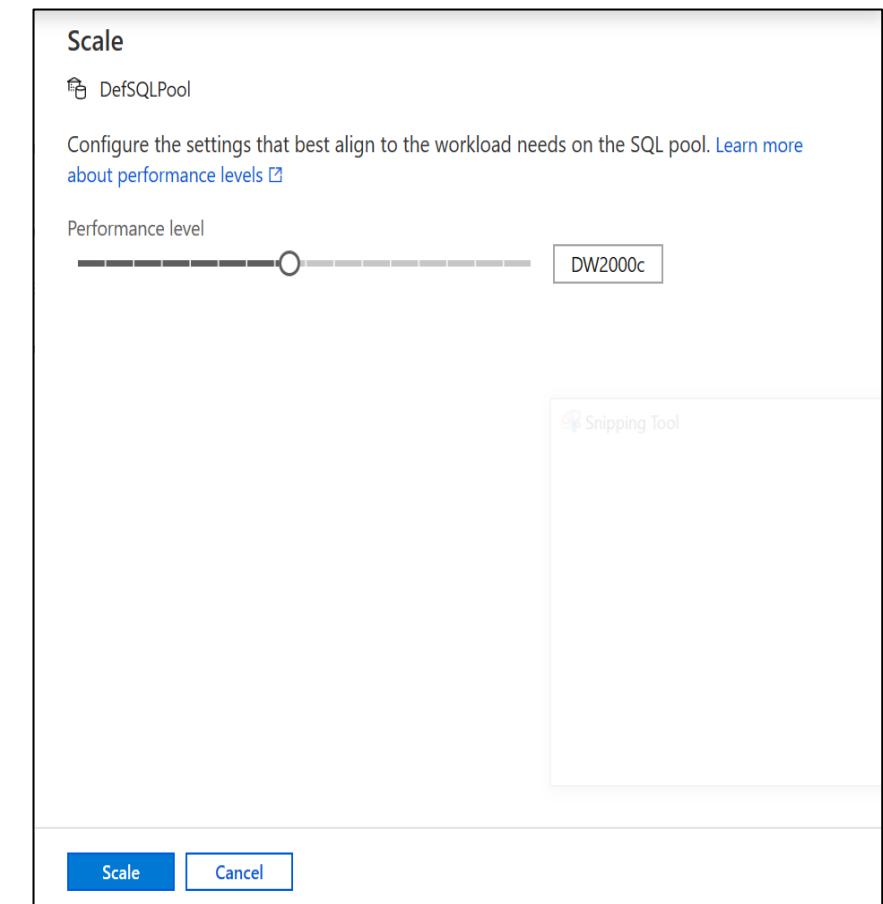
Performance level

DW2000c

Snipping Tool

⋮

Scale Cancel



# Manage – Apache Spark pools

## Overview

Provides ability to Pause, Scale, Assign Tags and upload packages from Studio.

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, there's a navigation sidebar with icons for Home, SQL pools, Apache Spark pools (which is selected), External connections, Linked services, Orchestration, Triggers, Integration runtimes, Security, Access control, and Managed private endpoints. The main area has a blue header bar with 'Microsoft Azure' and 'Synapse Analytics' and a breadcrumb trail 'internalsandbox'. Below the header are buttons for 'Publish all', 'Validate all', 'Refresh', and 'Discard all'. The main content area is titled 'Apache Spark pools' and displays a table with one item: 'DefSparkPool'. A context menu is open over this item, with options: 'Open' (highlighted with a red box), 'Auto-pause', 'Scale', 'AssignTags', and 'Delete'. To the right, a 'Properties' panel shows details for 'DefSparkPool': Name (DefSparkPool), URL (/subscriptions/58f8824d-32b0-4825-9825-02fa6a801546/resourceGroups/SynapseSandboxRG/prov...), Creation date (03/31/2020, 4:06:19 PM), Configuration, Workspace, and Packages (with an 'Upload environment config file' button). A red box highlights the 'Packages' section. At the bottom of the properties panel is a 'Close' button.

NAME	SIZE
DefSparkPool	Open this Apache Spark

**Properties**

Name  
DefSparkPool

URL  
/subscriptions/58f8824d-32b0-4825-9825-02fa6a801546/resourceGroups/SynapseSandboxRG/prov...

Creation date  
03/31/2020, 4:06:19 PM

Configuration

Workspace

Packages

Upload environment config file

Refresh

NAME	SIZE	DATE
requirements.txt	26B	4/8/2020, 5:24:19 PM

Close

# Manage – Linked services

## Overview

It defines the connection information needed to connect to external resources.

## Benefits

Offers pre-build 90+ connectors

Easy cross platform data migration

Represents data store or compute resources

The screenshot shows the Microsoft Azure Synapse Analytics 'Manage – Linked services' interface. On the left, there's a sidebar with icons for Analytics pools, SQL pools, Apache Spark pools, External connections (with 'Linked services' highlighted), Orchestration, Triggers, Integration runtimes, Security, Access control, and Managed private endpoints. The main area has tabs for 'Analytics pools', 'SQL pools', 'Apache Spark pools', and 'Linked services'. The 'Linked services' tab is selected, displaying a list of 35 items with a red box around the '+ New' button. A red arrow points from this button to a 'New linked service' dialog box on the right. This dialog lists various connectors: PayPal (Preview), Phoenix, PostgreSQL, Power BI (selected and highlighted with a blue border), Presto (Preview), QuickBooks (Preview), REST, SAP BW Open Hub, SAP BW via MDX, SAP Cloud For Customer, SAP ECC, SAP HANA, and several SAP-related options like SAP BW, SAP ECC, and SAP HANA. At the bottom of the dialog are 'Continue' and 'Cancel' buttons.

# Manage – Triggers

## Overview

It defines a unit of processing that determines when a pipeline execution needs to be kicked off.

## Benefits

### Create and manage

- Schedule trigger
- Tumbling window trigger
- Event trigger

### Control pipeline execution

The screenshot shows the 'Triggers' management interface in the Azure Synapse Analytics portal. On the left, there's a sidebar with various navigation items: Analytics pools, SQL pools, Apache Spark pools, External connections, Linked services, Orchestration, Triggers (which is highlighted with a red box), Integration runtimes, Security, Access control, and Managed private endpoints. Above the sidebar are buttons for Publish all, Validate all, Refresh, and Discard all. The main area is titled 'Triggers' and contains the message 'To execute a pipeline set the trigger type'. Below this is a 'New' button, also highlighted with a red box. To the right of the 'New' button is a detailed 'New trigger' dialog box. The dialog box has the following fields:

- Name \***: Trigger 2
- Description**: (empty)
- Type \***: Schedule (radio button selected)
- Start Date (UTC) \***: 10/29/2019 9:46 PM
- Recurrence \***: Every 1 Minute(s)
- End \***: No End (radio button selected)
- Annotations**: + New
- Activated \***: No (radio button selected)

At the bottom of the dialog are 'OK' and 'Cancel' buttons. In the bottom right corner of the main interface, there's a table showing a trigger named 'HolidayUpdateTrigger' with a status of 'Schedule' and 'Started'.

# Manage – Integration runtimes

## Overview

Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

## Benefits

Offers Azure Integration Runtime or Self-Hosted Integration Runtime

Azure Integration Runtime – provides fully managed, serverless compute in Azure

Self-Hosted Integration Runtime – use compute resources in on-premises machine or a VM inside private network

The screenshot shows the Microsoft Azure portal interface for managing integration runtimes. On the left, there's a sidebar with various navigation items: Home, Data, Develop, Orchestrate, Monitor, and Manage (which is currently selected and highlighted in blue). The main content area has a title 'Integration runtimes' and a sub-section 'The integration runtime (IR) is the compute infrastructure to'. Below this, there's a 'New' button (highlighted with a red box), a 'Refresh' button, and a message 'Showing 1 - 1 of 1 items'. At the bottom of the main pane, there are 'NAME ↑' and 'TYPE ↑' sort headers. To the right of the main pane, a modal window titled 'Integration runtime setup' is open. It asks 'Choose the network environment of the data source/destination or external compute to which the integration runtime will connect to for data movement or dispatch activities'. It shows two options: 'Azure' (represented by a cloud icon) and 'Self-Hosted' (represented by a server icon). At the bottom of the modal, there are 'Continue', 'Back', and 'Cancel' buttons.

# Manage – Access Control

## Overview

It provides access control management to workspace resources and artifacts for admins

## Benefits

Share workspace with the team

Increases productivity

Manage permissions on SQL pools and Spark pools

The screenshot shows the Microsoft Azure Access Control interface for a workspace named 'internalsandbox'. The left sidebar lists various workspace resources: Analytics pools, SQL pools, Apache Spark pools, External connections, Linked services, Orchestration, Triggers, Integration runtimes, Security, and Access control. The 'Access control' item is highlighted with a red box. The main area displays a table of current role assignments:

NAME	TYPE	ROLE
Priyanka Langade Priyanka.Langade@microsoft.com	Individual	Workspace admin
Syna		

A red arrow points from the 'Add' button in the top navigation bar to the 'Add role assignment' dialog box. This dialog box contains fields for selecting a role (a dropdown menu labeled 'Select a role') and checkboxes for 'Select all', 'Workspace admin', 'SQL admin', and 'Apache Spark admin'. At the bottom are 'Apply' and 'Cancel' buttons.

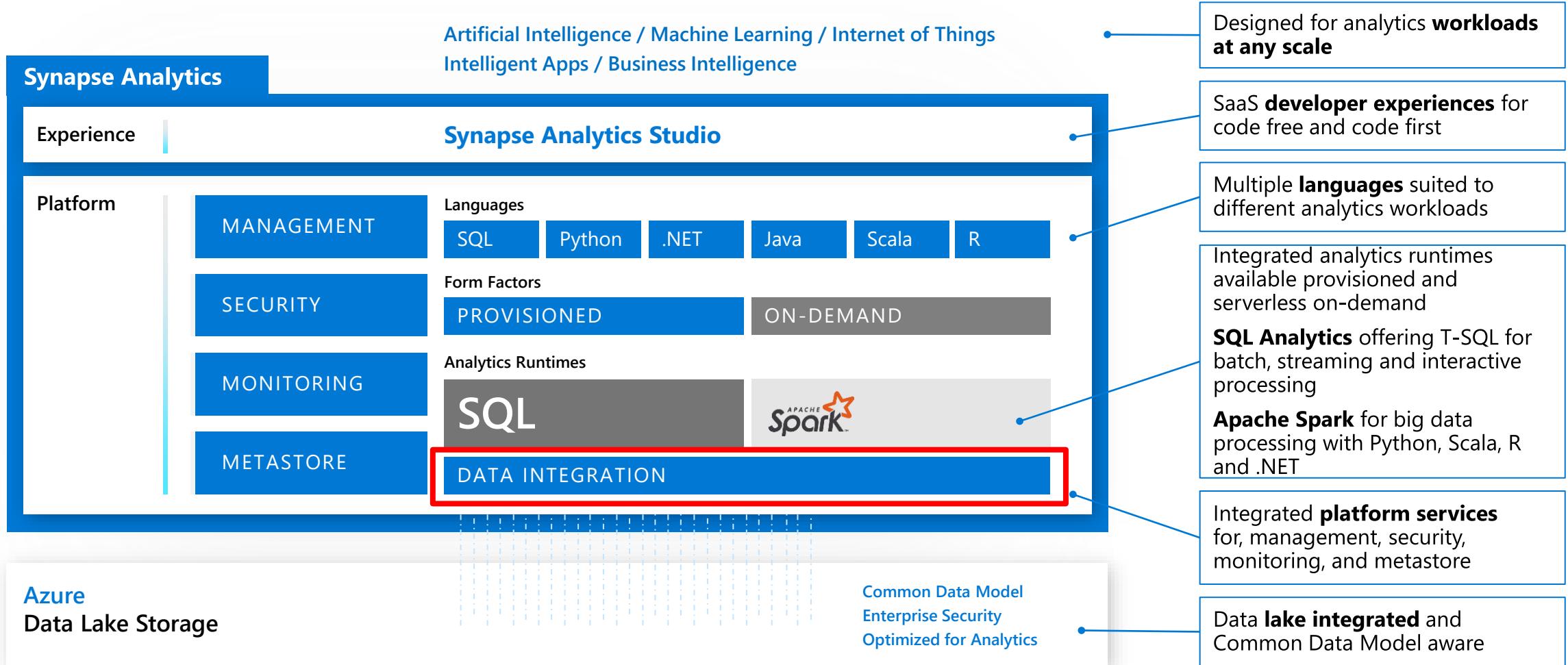


# Azure Synapse Analytics

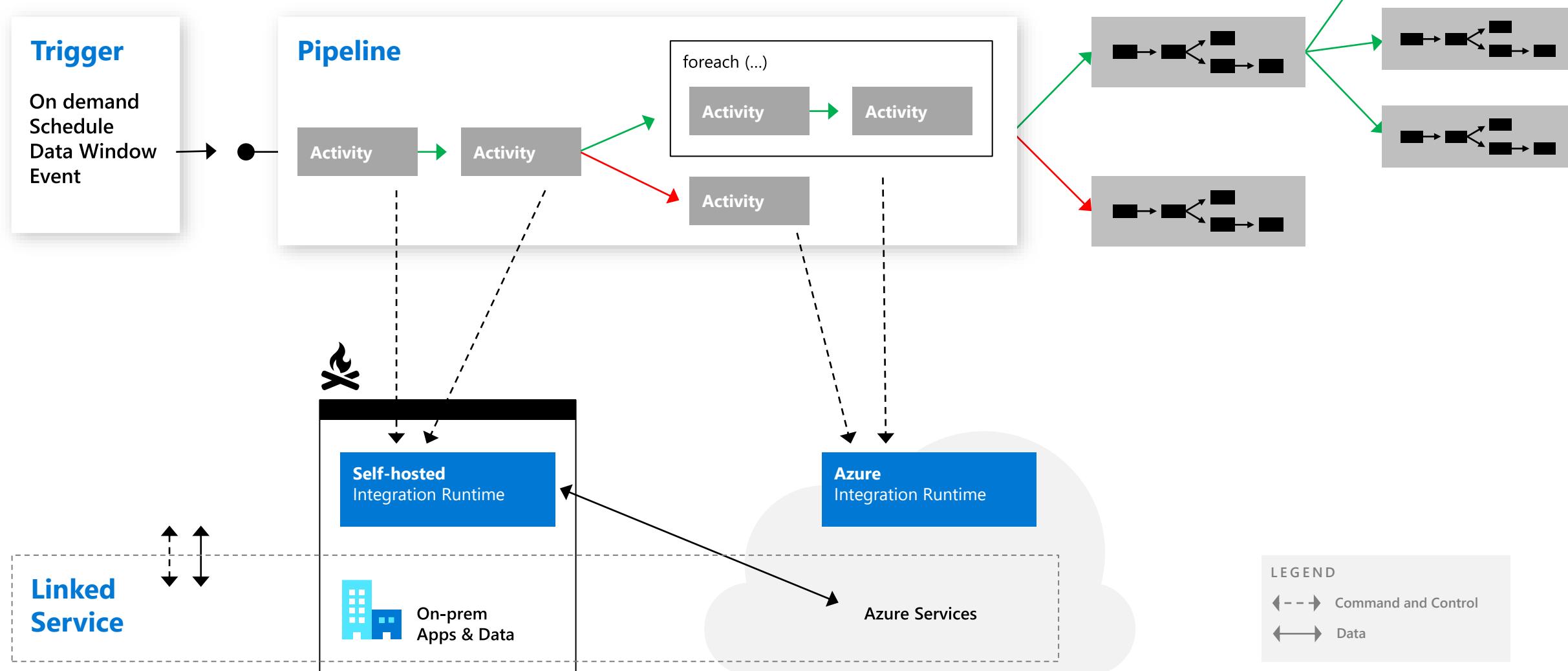
## Orchestration

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Orchestration @ Scale



# Data Movement

## Scalable

per job elasticity

Up to 4 GB/s

## Simple

Visually author or via code (Python, .NET, etc.)

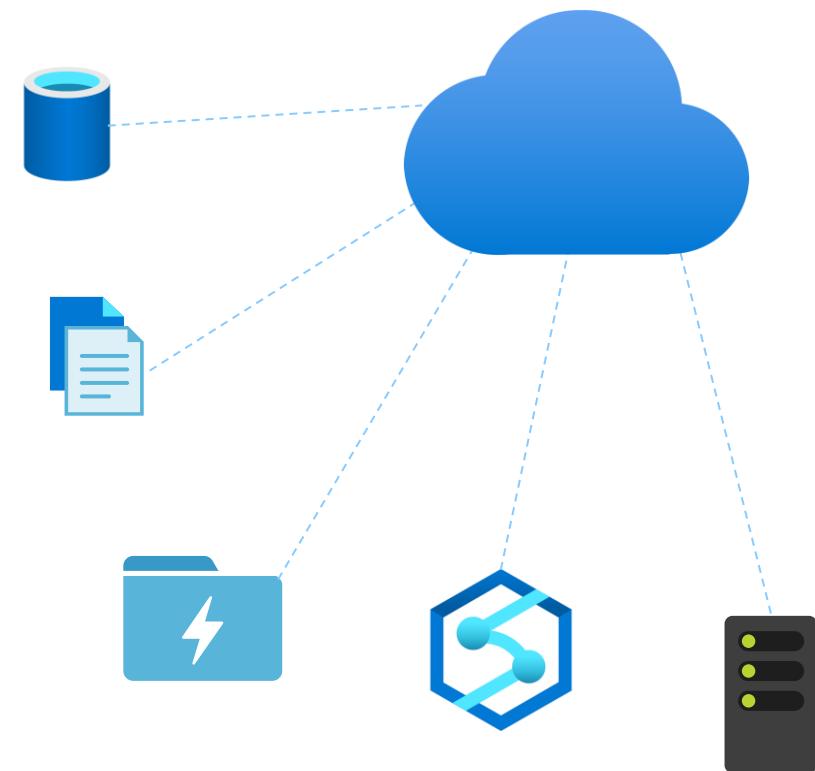
Serverless, no infrastructure to manage

## Access all your data

90+ connectors provided and growing (cloud, on premises, SaaS)

Data Movement as a Service: 25 points of presence worldwide

Self-hostable Integration Runtime for hybrid movement

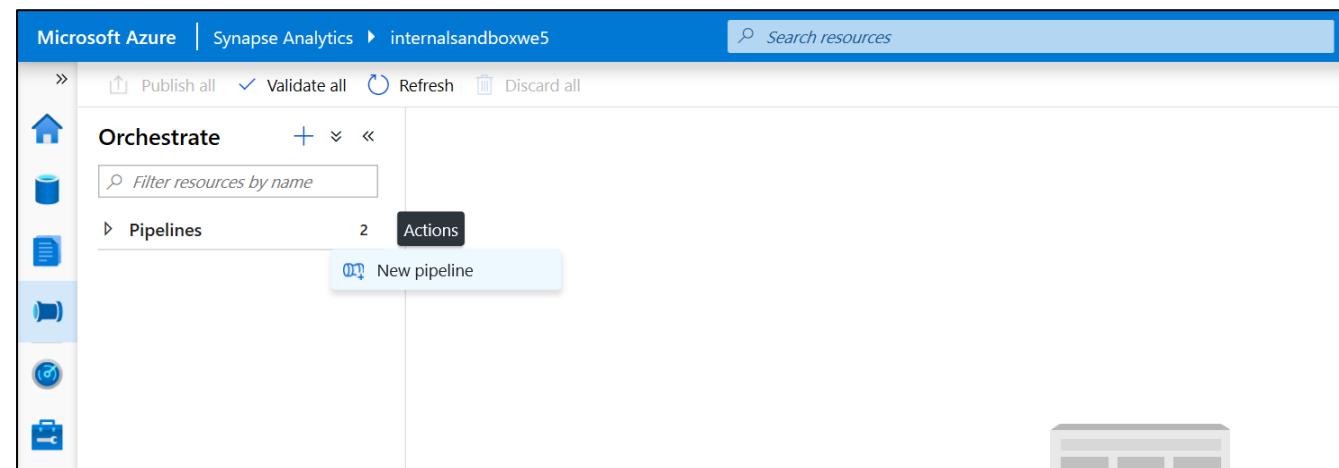


# 90+ Connectors out of the box

# Pipelines

## Overview

It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

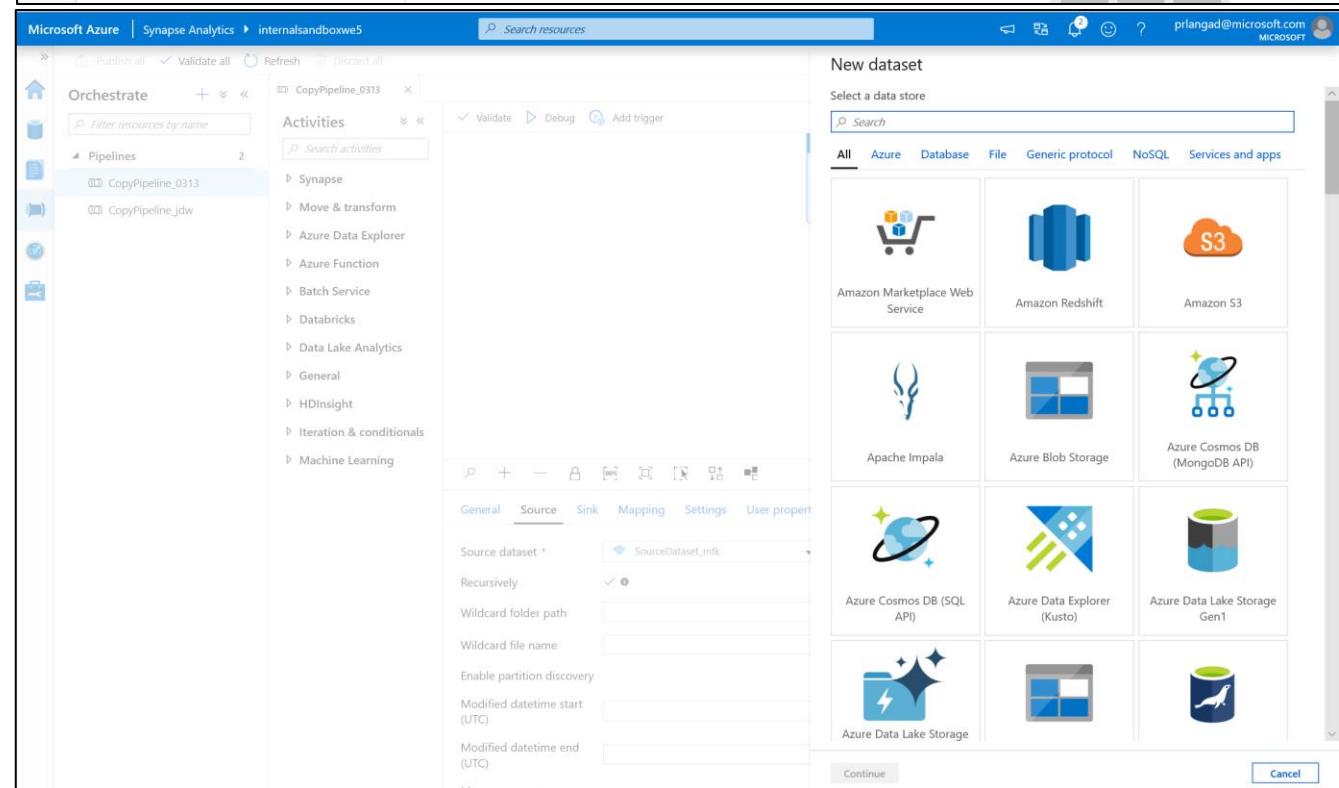


## Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

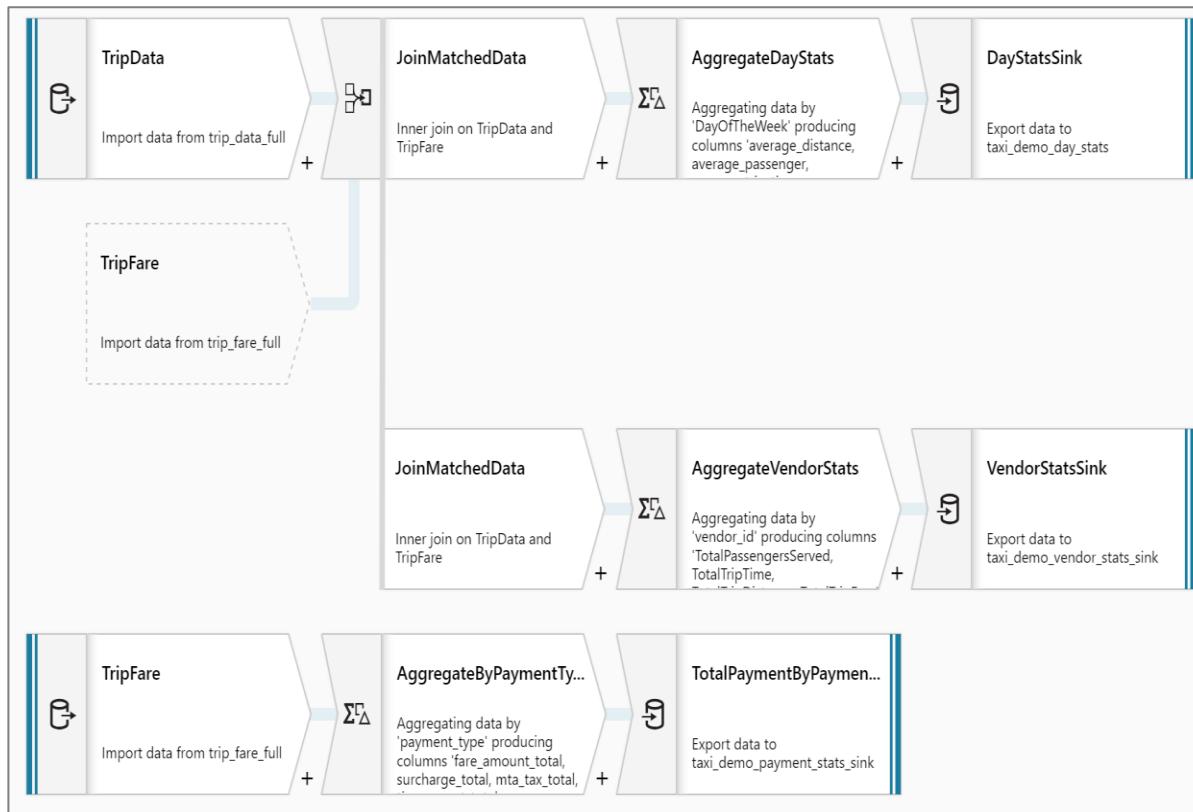
Graphical development experience



# Prep & Transform Data

## Mapping Dataflow

Code free data transformation @scale



## Wrangling Dataflow

Code free data preparation @scale

	CustId	FirstName	LastName	City	ZIP	Email	State	BasePay
1	1	"Harry"	"Potter"	"Bellevue"	"98004"	"harryk@fabrikam.com"	"WA"	90000
2	1	"Harry"	"Potter"	"Bellevue"	"98004"	"harryk@fabrikam.com"	"WA"	90000
3	2	"Hermione"	"Granger"	"Wilmington"	"19801"	"hermione@fabrikam.com"	"DE"	100000
4	2	"Hermione"	"Granger"	"Wilmington"	"19801"	"gamalloy@fabrikam.com"	"DE"	100000
5	3	"Lord"	"Voldemort"	"Billings"	"59115"	"lordc@fabrikam.com"	"MT"	110000
6	4	"Albus"	"Dumbledore"	"Newyork"	"12345"	"albusd@fabrikam.com"	"NY"	120000
7	5	"Severus"	"Snape"	"Columbus"	"56789"	"severus@fabrikam.com"	"OH"	130000
8	6	"Draco"	"Malfoy"	"Houston"	"91019"	"dracoh@fabrikam.com"	"TX"	140000
9	7	"Dobby"	"Elf"	"Salt Lake Ci..."	"11128"	"dobbyz@fabrikam.com"	"UT"	150000
10	8	"Ron"	"Weasley"	"Las Vegas"	"51527"	"ronag@fabrikam.com"	"NV"	160000
11	9	"Sirius"	"Black"	"Providence"	"61623"	"hblack@fabrikam.com"	"RI"	170000
12	10	"Luna"	"Lovegood"	"Kansas City"	"68692"	"lunal@fabrikam.com"	"MO"	180000
13	11	"Rubeus"	"Hagrid"	"Boston"	"98052"	"gamalloy@fabrikam.com"	"Malfoy"	190000
14	12	"Bellatrix"	"Lestrange"	"Los Angeles"	"78965"	"mlestrange@fabrikam.com"	"CA"	200000
15	13	"Ginny"	"Weasley"	"Redmond"	"98052"	"ginnyw@fabrikam.com"	"WA"	210000
16	14	"Neville"	"Longbottom"	"Bothell"	"98053"	"nevilea@fabrikam.com"	"WA"	220000
17	15	"Alastor"	"Moody"	"Renton"	"98054"	"albusd@fabrikam.com"	"WA"	230000
18	16	"Lucius"	"Malfoy"	"Bellevue"	"98004"	"luciusmalfoy@fabrikam.co..."	"WA"	240000
19	17	"Cedric"	"Diggory"	"Seattle"	"98989"	"cedricp@fabrikam.com"	"WA"	250000
20	18	"Argus"	"Filch"	"Salt Lake Ci..."	"11128"	"argusm@fabrikam.com"	"UT"	260000
21	3	"Lord"	"Voldemort"	"Billings"	"59115"	"lordc@fabrikam.com"	"MT"	110000
22	4	"Albus"	"Dumbledore"	"Newyork"	"12345"	"albusd@fabrikam.com"	"NY"	120000
23	5	"Severus"	"Snape"	"Columbus"	"56789"	"severus@fabrikam.com"	"OH"	130000
24	1	"Mason"	"Bettie"	"Bellmore"	"098004"	"masonb@fabrikam.com"	"MA"	00000

# Triggers

## Overview

Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.

Data Integration offers 3 trigger types as –

1. Schedule – gets fired at a schedule with information of start date, recurrence, end date
2. Event – gets fired on specified event
3. Tumbling window – gets fired at a periodic time interval from a specified start date, while retaining state

It also provides ability to monitor pipeline runs and control trigger execution.

The screenshot shows the Azure Synapse Analytics Data Integration interface. On the left, there's a navigation sidebar with options like 'Analytics pools', 'SQL pools', 'Apache Spark pools', 'External connections', 'Linked services', 'Orchestration', 'Triggers' (which is selected and highlighted in blue), 'Integration runtimes', 'Security', 'Access control', and 'Managed Virtual Networks'. The main area is titled 'Triggers' with the sub-instruction: 'To execute a pipeline set the trigger. Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.' Below this, there's a search bar with the placeholder 'Search to filter items...' and a table showing one item: 'HolidayUpdateTrigger' (Type: Schedule, Status: Started). At the top, there's a 'New trigger' dialog box with fields for Name (set to 'Trigger 1'), Description, Type (selected 'Schedule'), Start Date (set to '10/30/2019 11:20 PM'), Recurrence (set to 'Every 1 Minute(s)'), End (set to 'No End'), Annotations (with a '+ New' button), and Activated (set to 'Yes').

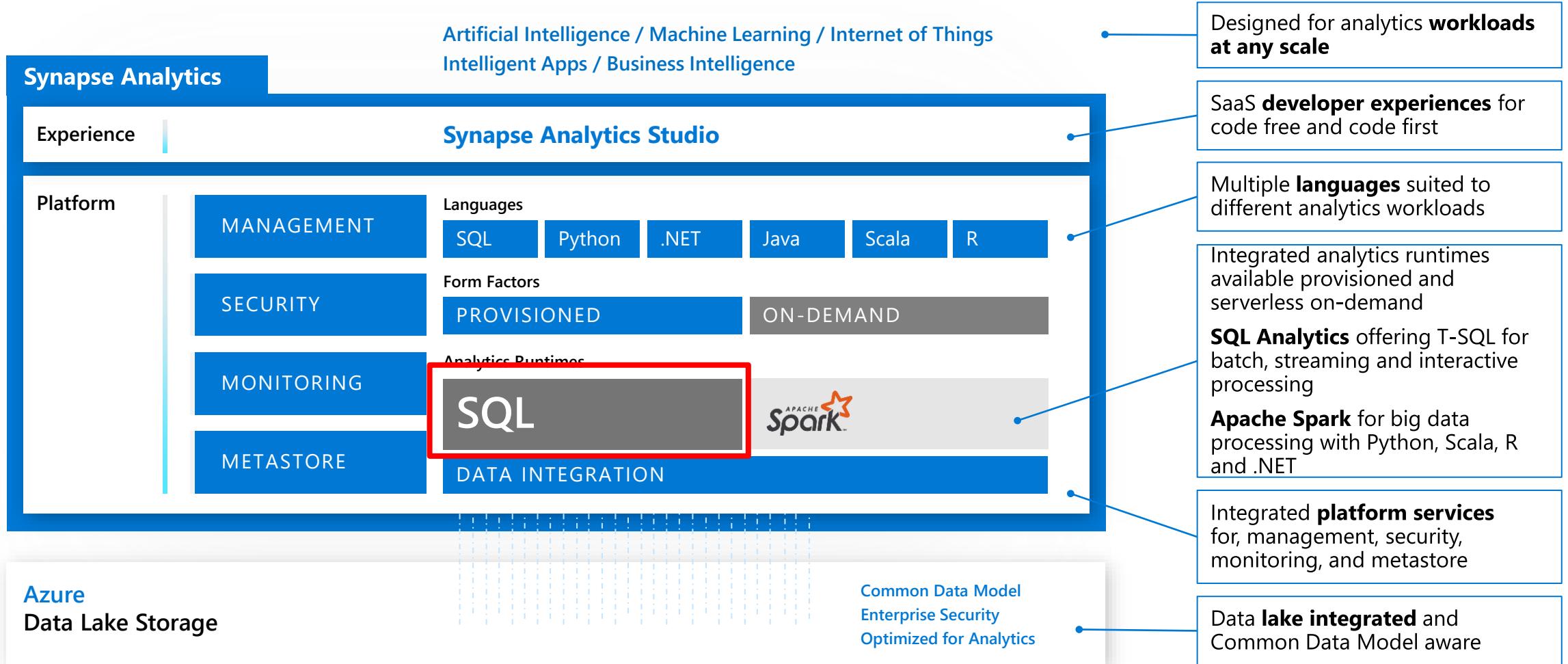


# Azure Synapse Analytics

## SQL Provisioned

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



# Comprehensive SQL functionality



## Advanced storage system

- Columnstore Indexes
- Table partitions
- Distributed tables
- Isolation modes
- Materialized Views
- Nonclustered Indexes
- Result-set caching

## T-SQL Querying

- Windowing aggregates
- Approximate execution (Hyperloglog)
- JSON data support
- Score machine learning models in ONNX format

## Complete SQL object model

- Tables
- Views
- Stored procedures
- Functions

# Windowing functions

## OVER clause

Defines a window or specified set of rows within a query result set

Computes a value for each row in the window

## Aggregate functions

COUNT, MAX, AVG, SUM, APPROX\_COUNT\_DISTINCT, MIN, STDEV, STDEVP, STRING\_AGG, VAR, VARP, GROUPING, GROUPING\_ID, COUNT\_BIG, CHECKSUM\_AGG

## Ranking functions

RANK, NTILE, DENSE\_RANK, ROW\_NUMBER

## Analytical functions

LAG, LEAD, FIRST\_VALUE, LAST\_VALUE, CUME\_DIST, PERCENTILE\_CONT, PERCENTILE\_DISC, PERCENT\_RANK

## ROWS | RANGE

PRECEDING, UNBOUNDED PRECEDING, CURRENT ROW, BETWEEN, FOLLOWING, UNBOUNDED FOLLOWING

`SELECT`

```
ROW_NUMBER() OVER(PARTITION BY PostalCode ORDER BY SalesYTD DESC) AS "Row Number",
LastName,
SalesYTD,
PostalCode
FROM Sales
WHERE SalesYTD <> 0
ORDER BY PostalCode;
```

Row Number	LastName	SalesYTD	PostalCode
1	Mitchell	4251368.5497	98027
2	Blythe	3763178.1787	98027
3	Carson	3189418.3662	98027
4	Reiter	2315185.611	98027
5	Vargas	1453719.4653	98027
6	Ansman-Wolfe	1352577.1325	98027
1	Pak	4116870.2277	98055
2	Varkey Chudukaktil	3121616.3202	98055
3	Saraiva	2604540.7172	98055
4	Ito	2458535.6169	98055
5	Valdez	1827066.7118	98055
6	Mensa-Annan	1576562.1966	98055
7	Campbell	1573012.9383	98055
8	Tsoflias	1421810.9242	98055

# Windowing Functions (continued)

## Analytical functions

LAG, LEAD, FIRST\_VALUE, LAST\_VALUE, CUME\_DIST, PERCENTILE\_CONT, PERCENTILE\_DISC, PERCENT\_RANK

-- PERCENTILE\_CONT, PERCENTILE\_DISC

```
SELECT DISTINCT Name AS DepartmentName
,PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY ph.Rate)
    OVER (PARTITION BY Name) AS MedianCont
,PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY ph.Rate)
    OVER (PARTITION BY Name) AS MedianDisc
FROM HumanResources.Department AS d
INNER JOIN HumanResources.EmployeeDepartmentHistory AS dh
    ON dh.DepartmentID = d.DepartmentID
INNER JOIN HumanResources.EmployeePayHistory AS ph
    ON ph.BusinessEntityID = dh.BusinessEntityID
WHERE dh.EndDate IS NULL;
```

DepartmentName	MedianCont	MedianDisc
Document Control	16.8269	16.8269
Engineering	34.375	32.6923
Executive	54.32695	48.5577
Human Resources	17.427850	16.5865

--LAG Function

```
SELECT BusinessEntityID,
YEAR(QuotaDate) AS SalesYear,
SalesQuota AS CurrentQuota,
LAG(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS PreviousQuota
FROM Sales.SalesPersonQuotaHistory
WHERE BusinessEntityID = 275 and YEAR(QuotaDate) IN ('2005','2006');
```

BusinessEntityID	SalesYear	CurrentQuota	PreviousQuota
275	2005	367000.00	0.00
275	2005	556000.00	367000.00
275	2006	502000.00	556000.00
275	2006	550000.00	502000.00
275	2006	1429000.00	550000.00
275	2006	1324000.00	1429000.00

# Windowing Functions (continued)

## ROWS | RANGE

PRECEDING, UNBOUNDING PRECEDING, CURRENT ROW, BETWEEN, FOLLOWING, UNBOUNDED FOLLOWING

```
-- First_Value  
  
SELECT JobTitle, LastName, VacationHours AS VacHours,  
FIRST_VALUE(LastName) OVER (PARTITION BY JobTitle  
ORDER BY VacationHours ASC ROWS UNBOUNDED PRECEDING ) AS FewestVacHours  
FROM HumanResources.Employee AS e  
INNER JOIN Person.Person AS p  
ON e.BusinessEntityID = p.BusinessEntityID  
ORDER BY JobTitle;
```



JobTitle	FewestVacHours	LastName	VacHours
<hr/>			
Accountant	Moreland	58	Moreland
Accountant	Seamans	59	Moreland
Accounts Manager	Liu	57	Liu
Accounts Payable Specialist	Tomic	63	Tomic
Accounts Payable Specialist	Sheperdigian	64	Tomic
Accounts Receivable Specialist	Poe	60	Poe
Accounts Receivable Specialist	Spoon	61	Poe
Accounts Receivable Specialist	Walton	62	Poe

# Approximate execution

## HyperLogLog accuracy

Will return a result with a 2% accuracy of true cardinality on average.

e.g. COUNT (DISTINCT) returns 1,000,000, HyperLogLog will return a value in the range of 999,736 to 1,016,234.

## APPROX\_COUNT\_DISTINCT

Returns the approximate number of unique non-null values in a group.

## Use Case: Approximating web usage trend behavior

-- Syntax

```
APPROX_COUNT_DISTINCT( expression )
```

-- The approximate number of different order keys by order status from the orders table.

```
SELECT O_OrderStatus, APPROX_COUNT_DISTINCT(O_OrderKey) AS Approx_Distinct_OrderKey  
FROM dbo.Orders  
GROUP BY O_OrderStatus  
ORDER BY O_OrderStatus;
```

# Approximate execution

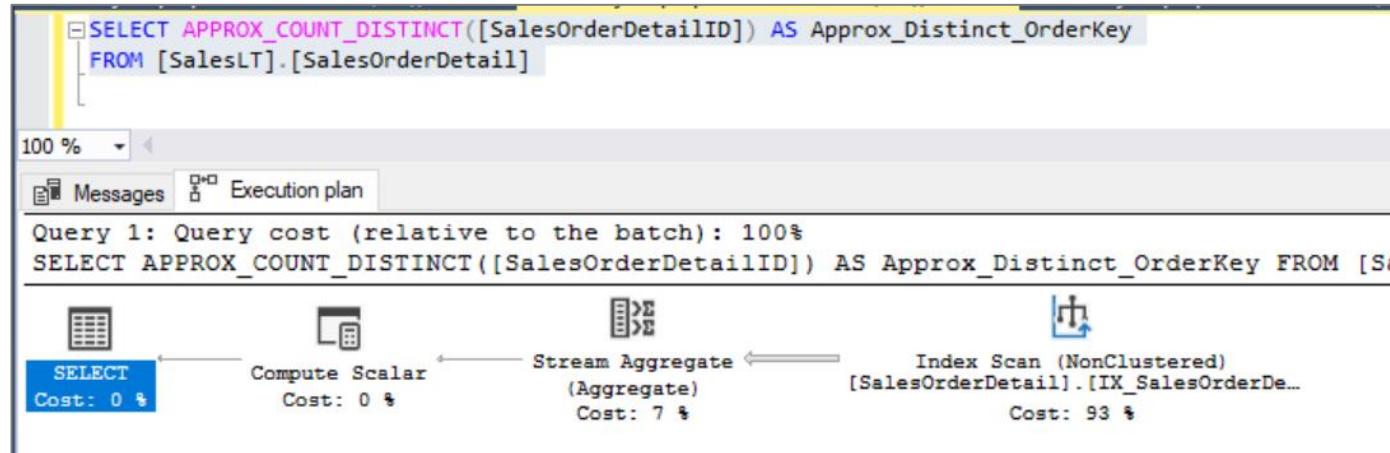
## APPROX\_COUNT\_DISTINCT

```
SELECT APPROX_COUNT_DISTINCT([SalesOrderDetailID]) AS Approx_Distinct_OrderKey
FROM [SalesLT].[SalesOrderDetail]
```

100 %

Results Messages

Approx_Distinct_OrderKey
540



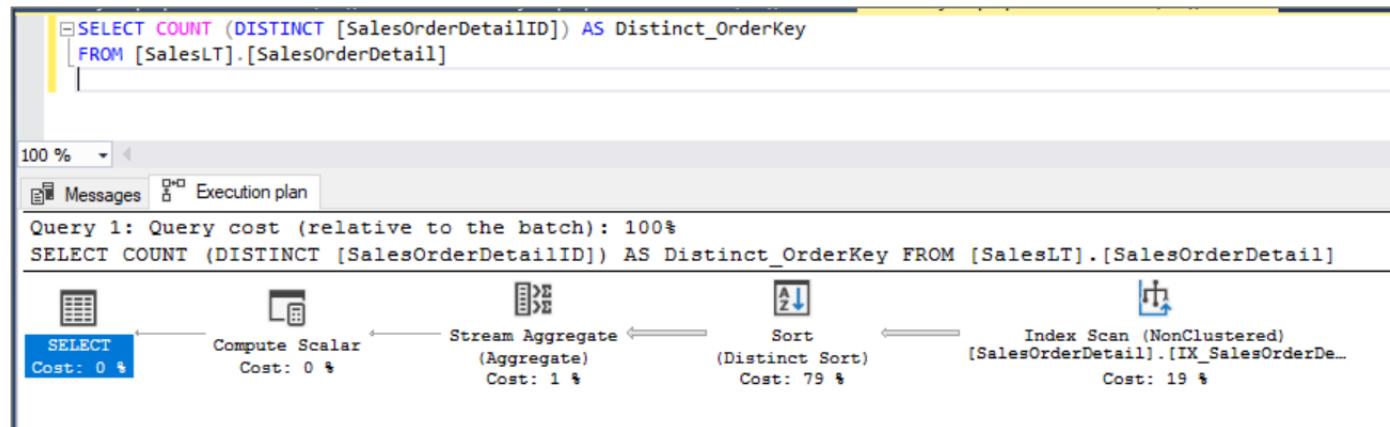
## COUNT DISTINCT

```
SELECT COUNT (DISTINCT [SalesOrderDetailID]) AS Distinct_OrderKey
FROM [SalesLT].[SalesOrderDetail]
```

100 %

Results Messages

Distinct_OrderKey
542



# Group by options

## Group by with rollup

Creates a group for each combination of column expressions.

Rolls up the results into subtotals and grand totals

Calculate the aggregates of hierarchical data

-- GROUP BY ROLLUP Example --

```
SELECT Country,
Region,
SUM(Sales) AS TotalSales
FROM Sales
GROUP BY ROLLUP (Country, Region);
```

-- Results --

## Grouping sets

Combine multiple GROUP BY clauses into one GROUP BY CLAUSE.

Equivalent of UNION ALL of specified groups.

-- GROUP BY SETS Example --

```
SELECT Country,
SUM(Sales) AS TotalSales
FROM Sales
GROUP BY GROUPING SETS ( Country, () );
```



Country	Region	TotalSales
Canada	Alberta	100
Canada	British Columbia	500
Canada	NULL	600
United States	Montana	100
United States	NULL	100
NULL	NULL	700

# Snapshot isolation

## Overview

Specifies that statements cannot read data that has been modified but not committed by other transactions.

This prevents dirty reads.

```
ALTER DATABASE MyDatabase  
SET ALLOW_SNAPSHOT_ISOLATION ON
```

```
ALTER DATABASE MyDatabase SET  
READ_COMMITTED_SNAPSHOT ON
```

## Isolation level

- READ COMMITTED
- REPEATABLE READ
- SNAPSHOT
- READ UNCOMMITTED
- SERIALIZABLE

## **READ\_COMMITTED\_SNAPSHOT**

**OFF** (Default) – Uses shared locks to prevent other transactions from modifying rows while running a read operation

**ON** – Uses row versioning to present each statement with a transactionally consistent snapshot of the data as it existed at the start of the statement. Locks are not used to protect the data from updates.

# JSON data support – insert JSON data

## Overview

The JSON format enables representation of complex or hierarchical data structures in tables.

JSON data is stored using standard NVARCHAR table columns.

## Benefits

Transform arrays of JSON objects into table format

Performance optimization using clustered columnstore indexes and memory optimized tables

```
-- Create Table with column for JSON string
CREATE TABLE CustomerOrders
(
    CustomerId BIGINT NOT NULL,
    Country NVARCHAR(150) NOT NULL,
    OrderDetails NVARCHAR(3000) NOT NULL -- NVARCHAR column for JSON
) WITH (DISTRIBUTION = ROUND_ROBIN)

-- Populate table with semi-structured data
INSERT INTO CustomerOrders
VALUES
( 101, -- CustomerId
'Bahrain', -- Country
N'[{ "StoreId": "AW73565",
    "Order": { "Number": "SO43659",
        "Date": "2011-05-31T00:00:00"
    },
    "Item": { "Price": 2024.40, "Quantity": 1 }
}]' -- OrderDetails
)
```

# JSON data support – read JSON data

## Overview

Read JSON data stored in a string column with the following:

- **ISJSON** – verify if text is valid JSON
- **JSON\_VALUE** – extract a scalar value from a JSON string
- **JSON\_QUERY** – extract a JSON object or array from a JSON string

## Benefits

Ability to get standard columns as well as JSON column

Perform aggregation and filter on JSON values

-- Return all rows with valid JSON data

```
SELECT CustomerId, OrderDetails
FROM CustomerOrders
WHERE ISJSON(OrderDetails) > 0;
```

CustomerId	OrderDetails
101	N'[{ StoreId": "AW73565", "Order": { "Number": "SO43659", "Date": "2011-05-31T00:00:00" }, "Item": { "Price": 2024.40, "Quantity": 1 }}]'

-- Extract values from JSON string

```
SELECT CustomerId,
Country,
JSON_VALUE(OrderDetails,'$.StoreId') AS StoreId,
JSON_QUERY(OrderDetails,'$.Item') AS ItemDetails
FROM CustomerOrders;
```

CustomerId	Country	StoreId	ItemDetails
101	Bahrain	AW73565	{ "Price": 2024.40, "Quantity": 1 }

# JSON data support – modify and operate on JSON data

## Overview

Use standard table columns and values from JSON text in the same analytical query.

Modify JSON data with the following:

- **JSON\_MODIFY** – modifies a value in a JSON string
- **OPENJSON** – convert JSON collection to a set of rows and columns

## Benefits

Flexibility to update JSON string using T-SQL

Convert hierarchical data into flat tabular structure

```
-- Modify Item Quantity value
UPDATE CustomerOrders SET OrderDetails =
JSON_MODIFY(OrderDetails, '$.OrderDetails.Item.Quantity', 2)
```

### OrderDetails

```
N'[{"StoreId": "AW73565", "Order": { "Number": "SO43659",
"Date": "2011-05-31T00:00:00" }, "Item": { "Price": 2024.40, "Quantity": 2}}]'
```

```
-- Convert JSON collection to rows and columns
SELECT CustomerId,
StoreId,
OrderDetails.OrderDate,
OrderDetails.OrderPrice
FROM CustomerOrders
CROSS APPLY OPENJSON (CustomerOrders.OrderDetails)
WITH ( StoreId    VARCHAR(50) '$.StoreId',
OrderNumber  VARCHAR(100) '$.Order.Date',
OrderDate    DATETIME   '$.Order.Date',
OrderPrice   DECIMAL    '$.Item.Price',
OrderQuantity INT       '$.Item.Quantity'
) AS OrderDetails
```

CustomerId	StoreId	OrderDate	OrderPrice
101	AW73565	2011-05-31T00:00:00	2024.40

# Stored Procedures

## Overview

It is a group of one or more SQL statements or a reference to a Microsoft .NET Framework common language runtime (CLR) method.

Promotes flexibility and modularity.

Supports parameters and nesting.

## Benefits

Reduced server/client network traffic, improved performance

Stronger security

Easy maintenance

```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SET NOCOUNT ON;
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment;
GO

-- Execute a stored procedures
EXECUTE HumanResources.uspGetAllEmployees;
GO
-- Or
EXEC HumanResources.uspGetAllEmployees;
GO
-- Or, if this procedure is the first statement within a batch:
HumanResources.uspGetAllEmployees;
```

# Tables – Indexes

## Clustered Columnstore index (Default Primary)

Highest level of data compression

Best overall query performance

## Clustered index (Primary)

Performant for looking up a single to few rows

## Heap (Primary)

Faster loading and landing temporary data

Best for small lookup tables

## Nonclustered indexes (Secondary)

Enable ordering of multiple columns in a table

Allows multiple nonclustered on a single table

Can be created on any of the above primary indexes

More performant lookup queries

-- Create table with index

```
CREATE TABLE orderTable
```

(

    OrderId INT NOT NULL,

    Date DATE NOT NULL,

    Name VARCHAR(2),

    Country VARCHAR(2)

)

WITH

(

    CLUSTERED COLUMNSTORE INDEX |

        HEAP |

        CLUSTERED INDEX (OrderId)

);

-- Add non-clustered index to table

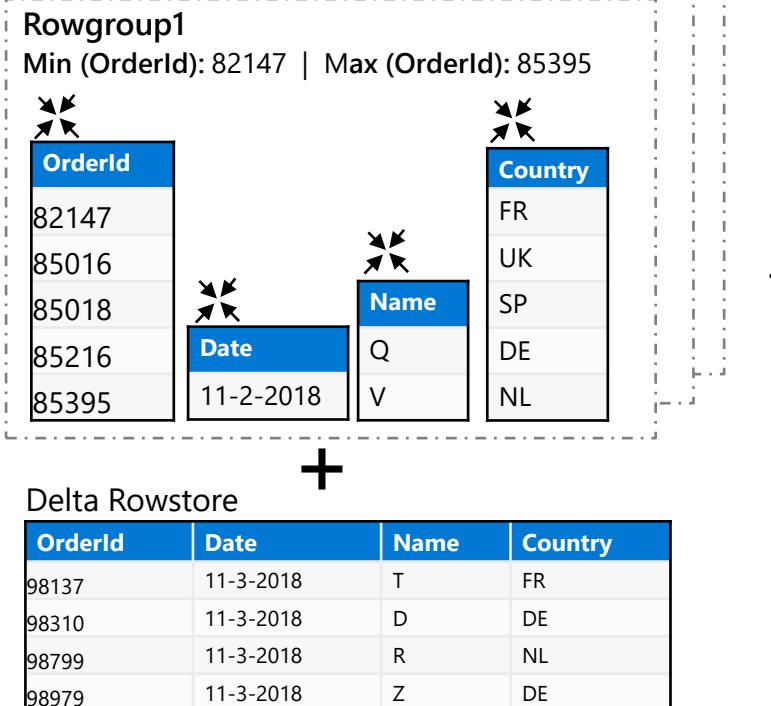
```
CREATE INDEX NameIndex ON orderTable (Name);
```

# SQL Analytics Columnstore Tables

## Logical table structure

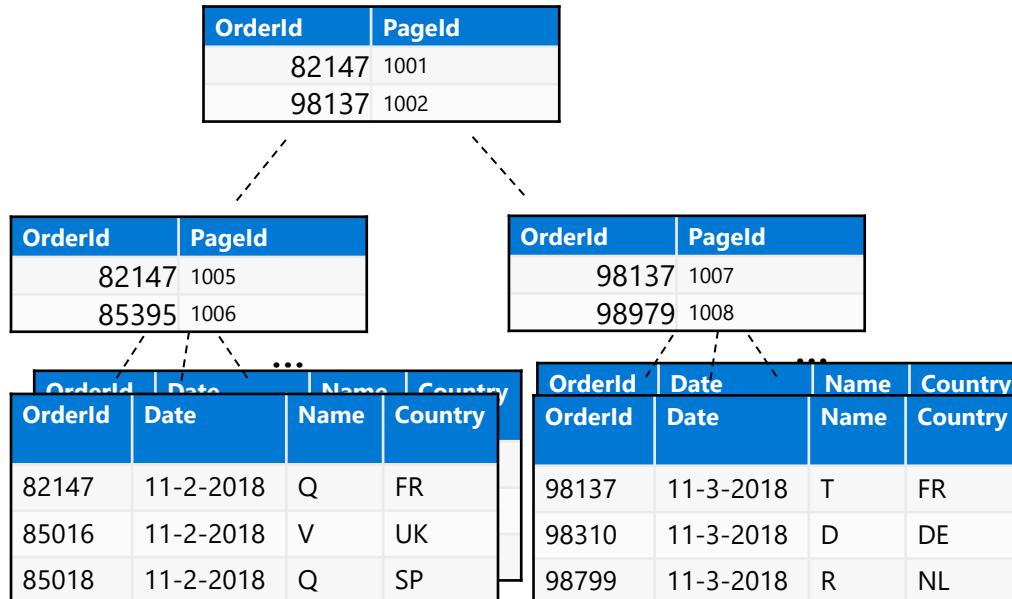
OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

## Clustered columnstore index (OrderId)



- Data stored in compressed columnstore segments after being sliced into groups of rows (rowgroups/micro-partitions) for maximum compression
- Rows are stored in the delta rowstore until the number of rows is large enough to be compressed into a columnstore

## Clustered/Non-clustered rowstore index (OrderId)



- Data is stored in a B-tree index structure for performant lookup queries for particular rows.
- Clustered rowstore index: The leaf nodes in the structure store the data values in a row (as pictured above)
- Non-clustered (secondary) rowstore index: The leaf nodes store pointers to the data values, not the values themselves

# Ordered Clustered Columnstore Indexes

## Overview

Queries against tables with ordered columnstore segments can take advantage of improved segment elimination to drastically reduce the time needed to service a query.

### -- Create Table with Ordered Columnstore Index

```
CREATE TABLE sortedOrderTable
```

```
(  
    OrderId INT NOT NULL,  
    Date DATE NOT NULL,  
    Name VARCHAR(2),  
    Country VARCHAR(2)
```

```
)  
WITH  
(  
    CLUSTERED COLUMNSTORE INDEX ORDER (OrderId)
```

### -- Create Clustered Columnstore Index on existing table

```
CREATE CLUSTERED COLUMNSTORE INDEX cciOrderId  
ON dbo.OrderTable ORDER (OrderId)
```

```
-- Insert data into table with ordered columnstore index
```

```
INSERT INTO sortedOrderTable
```

```
VALUES (1, '01-01-2019','Dave', 'UK')
```

# Tables – Distributions

## Round-robin distributed

Distributes table rows evenly across all distributions at random.

## Hash distributed

Distributes table rows across the Compute nodes by using a deterministic hash function to assign each row to one distribution.

## Replicated

Full copy of table accessible on each Compute node.

```
CREATE TABLE dbo.OrderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH([OrderId]) |
        ROUND ROBIN |
        REPLICATED
);
```

# Tables – Partitions

## Overview

Table partitions divide data into smaller groups

In most cases, partitions are created on a date column

Supported on all table types

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

## Benefits

Improves efficiency and performance of loading and querying by limiting the scope to subset of data.

Offers significant query performance enhancements where filtering on the partition key can eliminate unnecessary scans and eliminate IO.

```
CREATE TABLE partitionedOrderTable
(
    OrderId INT NOT NULL,
    Date DATE NOT NULL,
    Name VARCHAR(2),
    Country VARCHAR(2)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH([OrderId]),
    PARTITION (
        [Date] RANGE RIGHT FOR VALUES (
            '2000-01-01', '2001-01-01', '2002-01-01',
            '2003-01-01', '2004-01-01', '2005-01-01'
        )
    )
);
```

# Tables – Distributions & Partitions

## Logical table structure

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

## Physical data distribution

( Hash distribution (OrderId), Date partitions )

### Distribution1

(OrderId 80,000 – 100,000)

#### 11-2-2018 partition

OrderId	Date	Name	Country
85016	11-2-2018	V	UK
85018	11-2-2018	Q	SP
85216	11-2-2018	Q	DE
85395	11-2-2018	V	NL
82147	11-2-2018	Q	FR
86881	11-2-2018	D	UK
...	...	...	...

#### 11-3-2018 partition

OrderId	Date	Name	Country
93080	11-3-2018	R	UK
94156	11-3-2018	S	FR
96250	11-3-2018	Q	NL
98799	11-3-2018	R	NL
98015	11-3-2018	T	UK
98310	11-3-2018	D	DE
98979	11-3-2018	Z	DE
98137	11-3-2018	T	FR
...	...	...	...

...

x 60 distributions (shards)

- Each shard is partitioned with the same date partitions
- A minimum of 1 million rows per distribution and partition is needed for optimal compression and performance of clustered Columnstore tables

# Common table distribution methods

Table Category	Recommended Distribution Option
Fact	<p>Use hash-distribution with clustered columnstore index. Performance improves because hashing enables the platform to localize certain operations within the node itself during query execution.</p> <p>Operations that benefit:</p> <p>COUNT(DISTINCT( &lt;hashed_key&gt; ))</p> <p>OVER PARTITION BY &lt;hashed_key&gt;</p> <p>most JOIN &lt;table_name&gt; ON &lt;hashed_key&gt;</p> <p>GROUP BY &lt;hashed_key&gt;</p>
Dimension	Use replicated for smaller tables. If tables are too large to store on each Compute node, use hash-distributed.
Staging	Use round-robin for the staging table. The load with CTAS is faster. Once the data is in the staging table, use INSERT...SELECT to move the data to production tables.

# Materialized views

## Overview

A materialized view pre-computes, stores, and maintains its data like a table.

Materialized views are automatically updated when data in underlying tables are changed. This is a synchronous operation that occurs as soon as the data is changed.

The auto caching functionality allows Azure Synapse Analytics Query Optimizer to consider using indexed view even if the view is not referenced in the query.

Supported aggregations: MAX, MIN, AVG, COUNT, COUNT\_BIG, SUM, VAR, STDEV

## Benefits

Automatic and synchronous data refresh with data changes in base tables. No user action is required.

High availability and resiliency as regular tables

```
-- Create indexed view
CREATE MATERIALIZED VIEW Sales.vw_Orders
WITH
(
    DISTRIBUTION = ROUND_ROBIN |
    HASH(ProductID)
)
AS
    SELECT SUM(UnitPrice*OrderQty) AS Revenue,
        OrderDate,
        ProductID,
        COUNT_BIG(*) AS OrderCount
    FROM Sales.SalesOrderDetail
    GROUP BY OrderDate, ProductID;
GO

-- Disable index view and put it in suspended mode
ALTER INDEX ALL ON Sales.vw_Orders DISABLE;

-- Re-enable index view by rebuilding it
ALTER INDEX ALL ON Sales.vw_Orders REBUILD;
```

# Materialized views - example

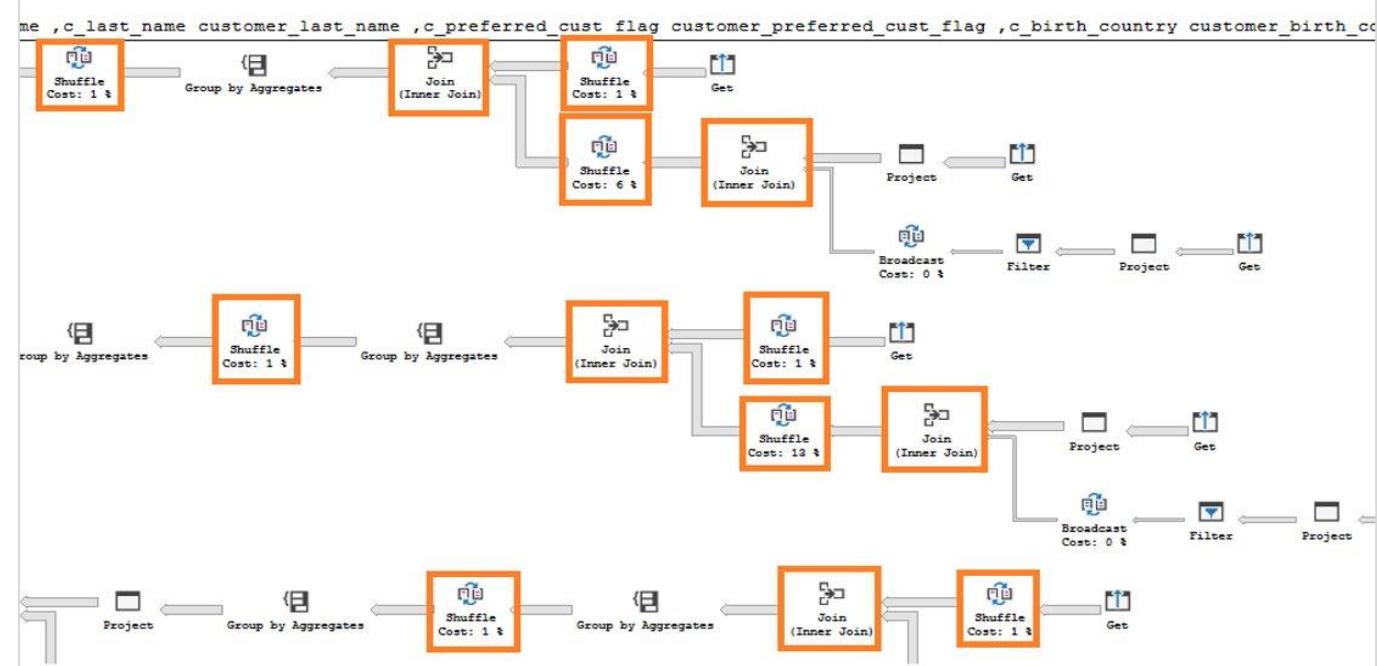
In this example, a query to get the year total sales per customer is shown to have a lot of data shuffles and joins that contribute to slow performance:

No relevant indexed views created on the data warehouse

```
-- Get year total sales per customer
(WITH year_total AS
    SELECT customer_id,
        first_name,
        last_name,
        birth_country,
        login,
        email_address,
        d_year,
        SUM(ISNULL(list_price - wholesale_cost -
            discount_amt + sales_price, 0)/2)year_total
    FROM customer cust
    JOIN catalog_sales sales ON cust.sk = sales.sk
    JOIN date_dim ON sales.sold_date = date_dim.date
    GROUP BY customer_id, first_name,
        last_name,birth_country,
        login,email_address ,d_year
)
SELECT TOP 100 ...
FROM year_total ...
WHERE ...
ORDER BY ...
```

**Execution time:** 103 seconds

Lots of data shuffles and joins needed to complete query



# Materialized views - example

Now, we add an indexed view to the data warehouse to increase the performance of the previous query. This view can be leveraged by the query even though it is not directly referenced.

Original query – get year total sales per customer

```
-- Get year total sales per customer
(WITH year_total AS
    SELECT customer_id,
           first_name,
           last_name,
           birth_country,
           login,
           email_address,
           d_year,
           SUM(ISNULL(list_price - wholesale_cost -
           discount_amt + sales_price, 0)/2)year_total
      FROM customer cust
     JOIN catalog_sales sales ON cust.sk = sales.sk
     JOIN date_dim ON sales.sold_date = date_dim.date
    GROUP BY customer_id, first_name,
             last_name,birth_country,
             login,email_address ,d_year
)
SELECT TOP 100 ...
   FROM year_total ...
  WHERE ...
 ORDER BY ...
```

Create indexed view with hash distribution on customer\_id column

```
-- Create indexed view for query
CREATE INDEXED VIEW nbViewCS WITH (DISTRIBUTION=HASH(customer_id)) AS
SELECT customer_id,
       first_name,
       last_name,
       birth_country,
       login,
       email_address,
       d_year,
       SUM(ISNULL(list_price - wholesale_cost - discount_amt +
       sales_price, 0)/2) AS year_total
  FROM customer cust
 JOIN catalog_sales sales ON cust.sk = sales.sk
 JOIN date_dim ON sales.sold_date = date_dim.date
 GROUP BY customer_id, first_name,
          last_name,birth_country,
          login, email_address, d_year
```

# Indexed (materialized) views - example

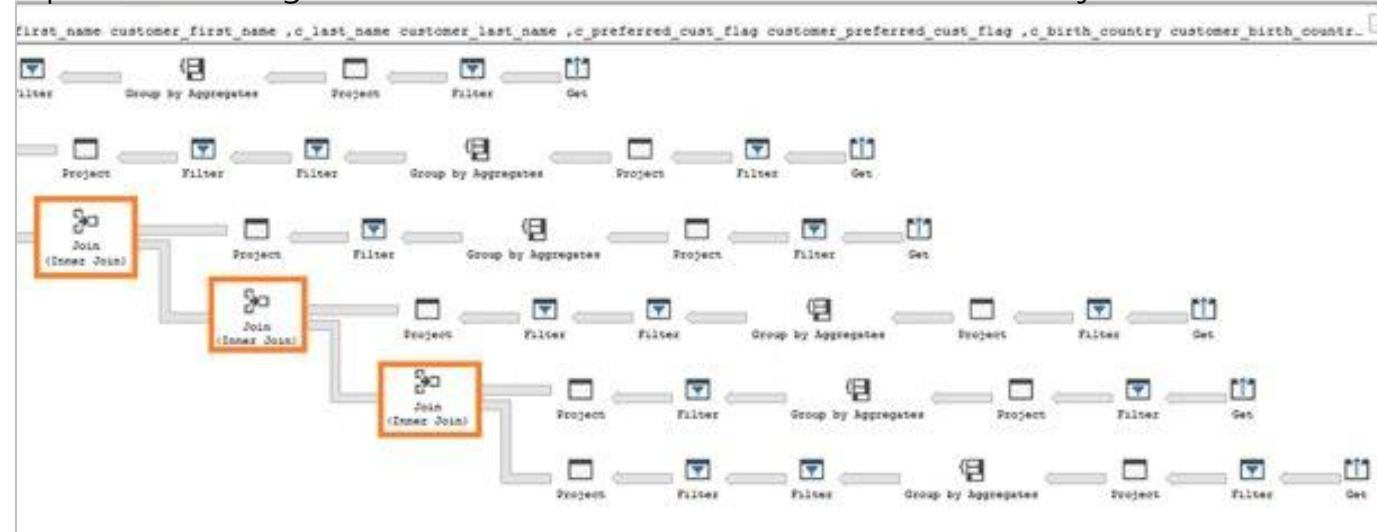
Synapse SQL (provisioned) query optimizer automatically leverages the indexed view to speed up the same query.  
 Notice that the query does not need to reference the view directly

Original query – no changes have been made to query

```
-- Get year total sales per customer
(WITH year_total AS
  SELECT customer_id,
         first_name,
         last_name,
         birth_country,
         login,
         email_address,
         d_year,
         SUM(ISNULL(list_price - wholesale_cost -
                    discount_amt + sales_price, 0)/2)year_total
    FROM customer cust
   JOIN catalog_sales sales ON cust.sk = sales.sk
   JOIN date_dim ON sales.sold_date = date_dim.date
 GROUP BY customer_id, first_name,
          last_name,birth_country,
          login,email_address ,d_year
)
SELECT TOP 100 ...
FROM year_total ...
WHERE ...
ORDER BY ...
```

**Execution time:** 6 seconds

Optimizer leverages materialized view to reduce data shuffles and joins needed



# Materialized views- Recommendations

**EXPLAIN** - provides query plan for SQL statement without running the statement; view estimated cost of the query operations.

**EXPLAIN WITH\_RECOMMENDATIONS** - provides query plan with recommendations to optimize the SQL statement performance.

```
EXPLAIN WITH_RECOMMENDATIONS
select count(*)
from (
    select distinct c_last_name, c_first_name, d_date
    from store_sales, date_dim, customer
    where store_sales.ss_sold_date_sk = date_dim.d_date_sk
    and store_sales.ss_customer_sk = customer.c_customer_sk
    and d_month_seq between 1194 and 1194+11)
except
    (select distinct c_last_name, c_first_name, d_date
    from catalog_sales, date_dim, customer
    where catalog_sales.cs_sold_date_sk = date_dim.d_date_sk
    and catalog_sales.cs_bill_customer_sk = customer.c_customer_sk and
    d_month_seq between 1194 and 1194+11)
) top_customers
```

# Result-set caching

## Overview

Cache the results of a query in provisioned SQL storage. This enables interactive response times for repetitive queries against tables with infrequent data changes.

The result-set cache persists even if SQL provisioned is paused and resumed later.

Query cache is invalidated and refreshed when underlying table data or query code changes.

Result cache is evicted regularly based on a time-aware least recently used algorithm (TLRU).

## Benefits

Enhances performance when same result is requested repetitively

Reduced load on server for repeated queries

Offers monitoring of query execution with a result cache hit or miss

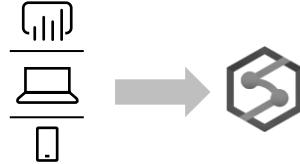
```
-- Turn on/off result-set caching for a database  
-- Must be run on the MASTER database  
ALTER DATABASE {database_name}  
SET RESULT_SET_CACHING { ON | OFF }
```

```
-- Turn on/off result-set caching for a client session  
-- Run on target Azure Synapse Analytics  
SET RESULT_SET_CACHING {ON | OFF}
```

```
-- Check result-set caching setting for a database  
-- Run on target Azure Synapse Analytics  
SELECT is_result_set_caching_on  
FROM sys.databases  
WHERE name = {database_name}
```

```
-- Return all query requests with cache hits  
-- Run on target data warehouse  
SELECT *  
FROM sys.dm_pdw_request_steps  
WHERE command like '%DWResultCacheDb%'  
    AND step_index = 0
```

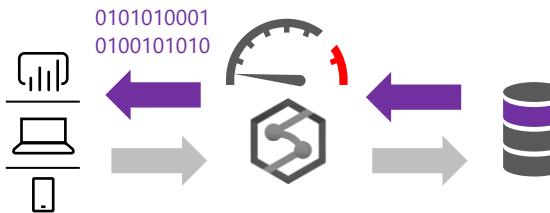
# Result-set caching flow



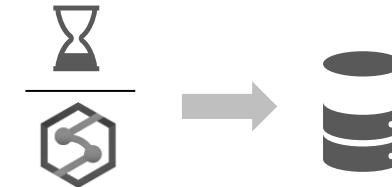
- 1 Client sends query to SQL provisioned



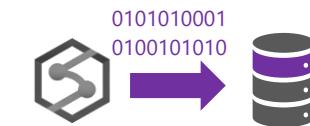
- 2 Query is processed using compute nodes which pull data from remote storage, process query and output back to client app



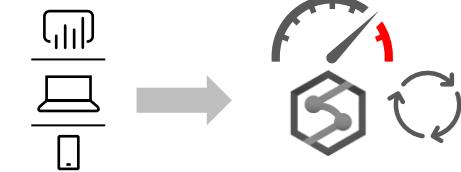
- 3 Subsequent executions for the same query bypass compute nodes and can be fetched instantly from persistent cache in remote storage



- 4 Remote storage cache is evicted regularly based on time, cache usage, and any modifications to underlying table data.



+  
Query results are cached in remote storage so subsequent requests can be served immediately



- 5 Cache will need to be regenerated if query results have been evicted from cache

# COPY command

## Overview

Copies data from source to destination

## Benefits

Retrieves data from all files from the folder and all its subfolders.

Supports multiple locations from the same storage account, separated by comma

Supports Azure Data Lake Storage (ADLS) Gen 2 and Azure Blob Storage.

Supports CSV, PARQUET, ORC file formats

```
COPY INTO test_1
FROM 'https://XXX.blob.core.windows.net/customerdatasets/test_1.txt'
WITH (
    FILE_TYPE = 'CSV',
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>'),
    FIELDQUOTE = """",
    FIELDTERMINATOR=';',
    ROWTERMINATOR='0XA',
    ENCODING = 'UTF8',
    DATEFORMAT = 'ymd',
    MAXERRORS = 10,
    ERRORFILE = '/errorsfolder/'--path starting from the storage container,
    IDENTITY_INSERT
)
```

```
COPY INTO test_parquet
FROM 'https://XXX.blob.core.windows.net/customerdatasets/test.parquet'
WITH (
    FILE_FORMAT = myFileFormat
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>')
)
```

# Create External Table As Select

## Overview

Creates an external table and then exports results of the Select statement. These operations will import data into the database for the duration of the query

### Steps:

1. Create Master Key
2. Create Credentials
3. Create External Data Source
4. Create External Data Format
5. Create External Table

```
-- Create a database master key if one does not already exist
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!Info'
;

-- Create a database scoped credential with Azure storage account key as the secret.
CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY = '<my_account>'
, SECRET   = '<azure_storage_account_key>'
;
-- Create an external data source with CREDENTIAL option.
CREATE EXTERNAL DATA SOURCE MyAzureStorage
WITH
(
    LOCATION  = 'wasbs://daily@logs.blob.core.windows.net/'
, CREDENTIAL = AzureStorageCredential
, TYPE      = HADOOP
)
-- Create an external file format
CREATE EXTERNAL FILE FORMAT MyAzureCSVFormat
WITH (FORMAT_TYPE = DELIMITEDTEXT,
      FORMAT_OPTIONS(
          FIELD_TERMINATOR = ',',
          FIRST_ROW = 2))
--Create an external table
CREATE EXTERNAL TABLE dbo.FactInternetSalesNew
WITH(
    LOCATION = '/files/Customer',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureCSVFormat
)
AS SELECT T1.* FROM dbo.FactInternetSales T1 JOIN dbo.DimCustomer T2
ON ( T1.CustomerKey = T2.CustomerKey )
OPTION ( HASH JOIN );
```

# Predict

## Overview

It provides ability to import existing machine learning models and score them within provisioned SQL. It takes ONNX (Open Neural Network Exchange) and data as inputs and generates prediction based on model.

## Benefits

1. It empowers data engineers to successfully deploy machine learning models with the familiar T-SQL interface
2. It offers seamless collaboration with data scientists
3. It generates new columns, but the number of columns and their data types depends on the type of model that was used for prediction.

### Syntax:

```
PREDICT
(
    MODEL = @model | model_literal,
    DATA = object AS <table_alias>
)
WITH ( <result_set_definition> )
<result_set_definition> ::= 
{
    { column_name
        data_type
    }
    [,...n]
}
MODEL = @model | model_literal
```

### Example:

```
DECLARE @model varbinary(max) = (SELECT Model FROM Models WHERE Id = <>);
SELECT d.*, p.Score
FROM PREDICT(MODEL = @model,
    DATA = dbo.mytable AS d) WITH (Score float) AS p;
```

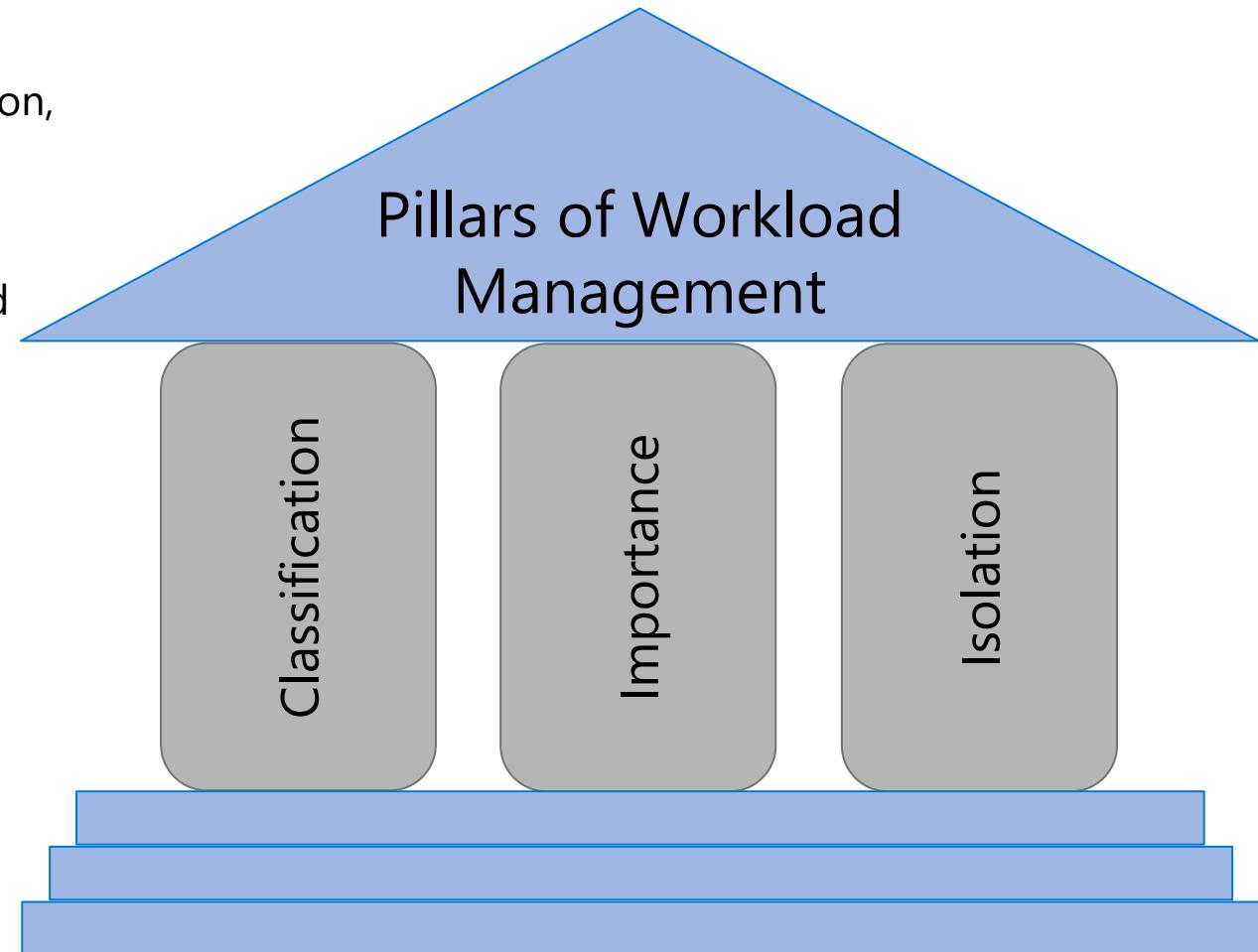
# Workload Management

## Overview

It manages resources, ensures highly efficient resource utilization, and maximizes return on investment (ROI).

The three pillars of workload management are

1. Workload Classification – To assign a request to a workload group and setting importance levels.
2. Workload Importance – To influence the order in which a request gets access to resources.
3. Workload Isolation – To reserve resources for a workload group.



# Workload classification

## Overview

Map queries to allocations of resources via pre-determined rules.

Use with workload importance to effectively share resources across different workload types.

If a query request is not matched to a classifier, it is assigned to the default workload group.

## Benefits

Map queries to both Resource Management and Workload Isolation concepts.

## Monitoring DMVs

[sys.workload\\_management\\_workload\\_classifiers](#)

[sys.workload\\_management\\_workload\\_classifier\\_details](#)

Query DMVs to view details about all active workload classifiers.

```
CREATE WORKLOAD CLASSIFIER classifier_name
WITH
(
    WORKLOAD_GROUP = 'name'
    , MEMBERNAME = 'security_account'
    [ [,] IMPORTANCE = {LOW|BELOW_NORMAL|NORMAL|ABOVE_NORMAL|HIGH} ]
    [ [,] WLM_LABEL = 'label' ]
    [ [,] WLM_CONTEXT = 'name' ]
    [ [,] START_TIME = 'start_time' ]
    [ [,] END_TIME = 'end_time' ]
)[ ; ]
```

*WORKLOAD\_GROUP: maps to an existing resource class*

*IMPORTANCE: specifies relative importance of request*

*MEMBERNAME: database user, role, AAD login or AAD group*

# Workload importance

## Overview

Queries past the concurrency limit enter a FiFo queue

By default, queries are released from the queue on a first-in, first-out basis as resources become available

Workload importance allows higher priority queries to receive resources immediately regardless of queue

## Example Video

State analysts have normal importance.

National analyst is assigned high importance.

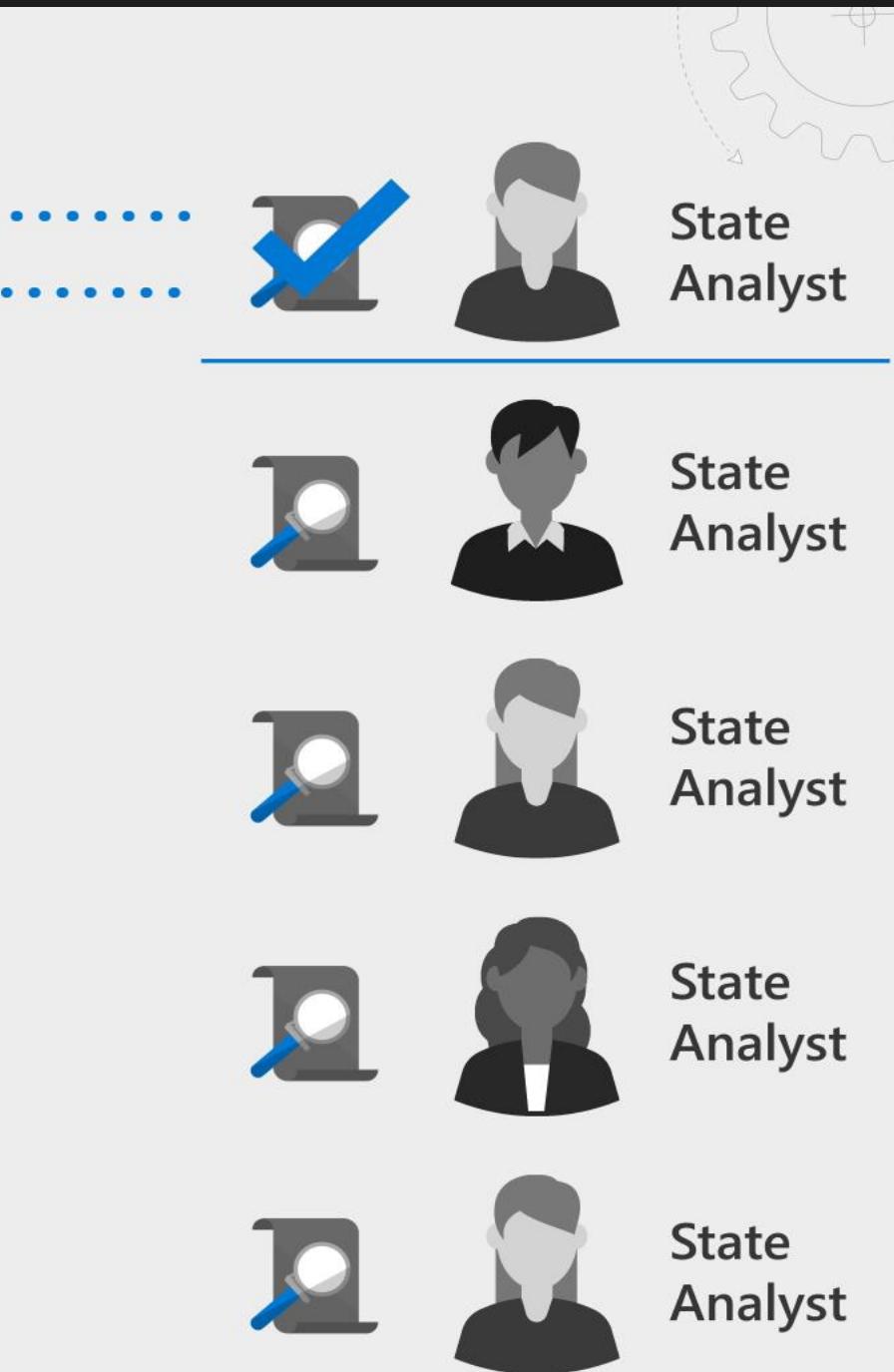
State analyst queries execute in order of arrival

When the national analyst's query arrives, it jumps to the top of the queue

```
CREATE WORKLOAD CLASSIFIER National_Analyst
WITH
(
    WORKLOAD_GROUP = 'analyst'
    ,IMPORTANCE = HIGH
    ,MEMBERNAME = 'National_Analyst_Login')
```



Azure Synapse  
Analytics



# Workload Isolation

## Overview

Allocate fixed resources to workload group.

Assign maximum and minimum usage for varying resources under load. These adjustments can be done live without having to SQL Analytics offline.

## Benefits

Reserve resources for a group of requests

Limit the amount of resources a group of requests can consume

Shared resources accessed based on importance level

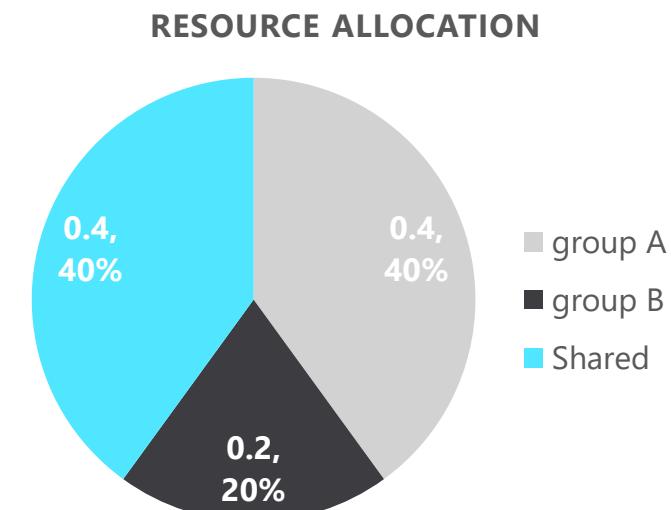
Set Query timeout value. Get DBAs out of the business of killing runaway queries

## Monitoring DMVs

[sys.workload\\_management\\_workload\\_groups](#)

Query to view configured workload group.

```
CREATE WORKLOAD GROUP group_name
WITH
(
    MIN_PERCENTAGE_RESOURCE = value
    , CAP_PERCENTAGE_RESOURCE = value
    , REQUEST_MIN_RESOURCE_GRANT_PERCENT = value
    [[,] REQUEST_MAX_RESOURCE_GRANT_PERCENT = value ]
    [[,] IMPORTANCE = {LOW | BELOW_NORMAL | NORMAL | ABOVE_NORMAL | HIGH} ]
    [[,] QUERY_EXECUTION_TIMEOUT_SEC = value ]
)[;]
```



# Dynamic Management Views (DMVs)

## Overview

Dynamic Management Views (DMV) are queries that return information about model objects, server operations, and server health.

## Benefits:

Simple SQL syntax

Returns result in table format

Easier to read and copy result

# SQL Monitor with DMVs

## Overview

Offers monitoring of

- all open, closed sessions
- count sessions by user
- count completed queries by user
- all active, complete queries
- longest running queries
- memory consumption

Count sessions by user

--count sessions by user

```
SELECT login_name, COUNT(*) as session_count FROM sys.dm_pdw_exec_sessions  
where status = 'Closed' and session_id <> session_id() GROUP BY login_name;
```

List all open sessions

-- List all open sessions

```
SELECT * FROM sys.dm_pdw_exec_sessions where status <> 'Closed' and session_id <>  
session_id();
```

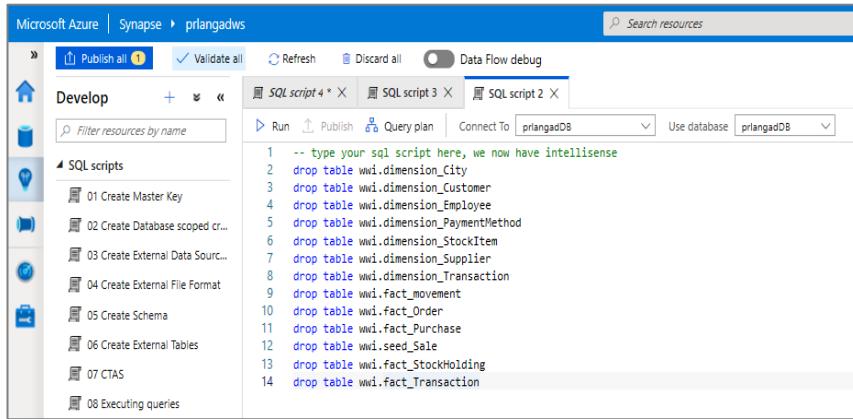
List all active queries

-- List all active queries

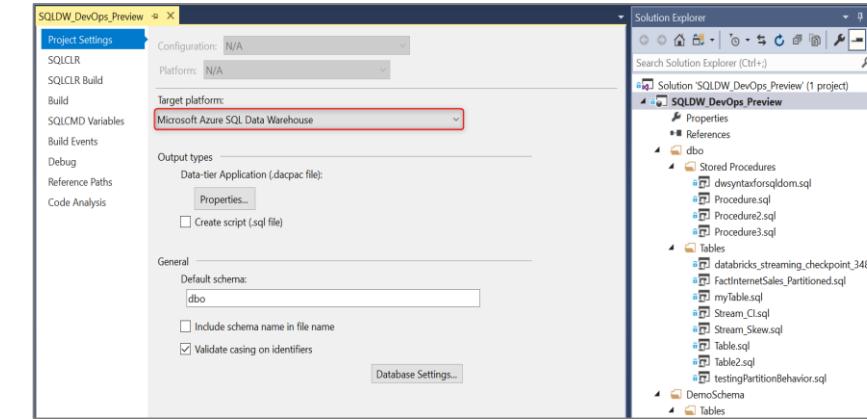
```
SELECT * FROM sys.dm_pdw_exec_requests WHERE status not in  
('Completed', 'Failed', 'Cancelled') AND session_id <> session_id() ORDER BY submit_time  
DESC;
```

# Developer Tools

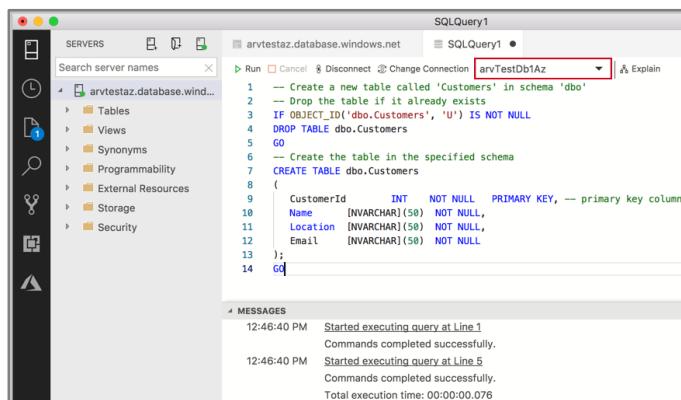
## Azure Synapse Analytics



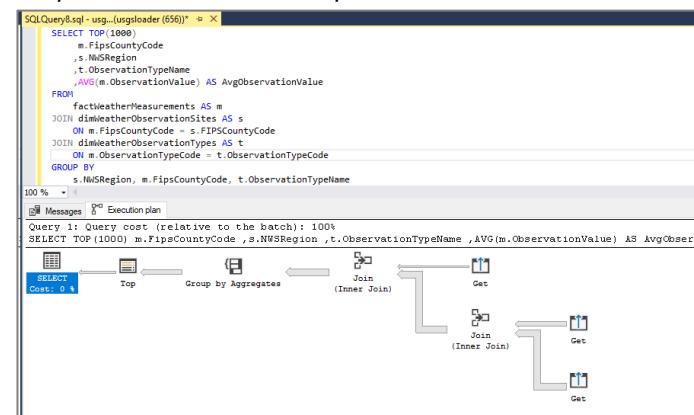
## Visual Studio - SSDT database projects



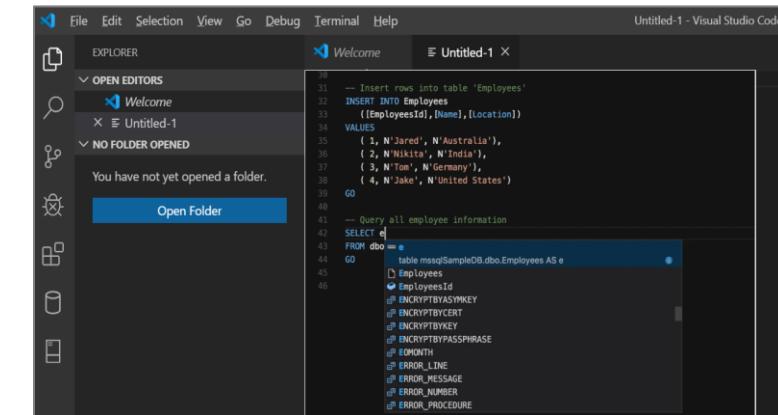
## Azure Data Studio (queries, extensions etc.)



## SQL Server Management Studio (queries, execution plans etc.)



## Visual Studio Code



# Continuous integration and delivery (CI/CD)

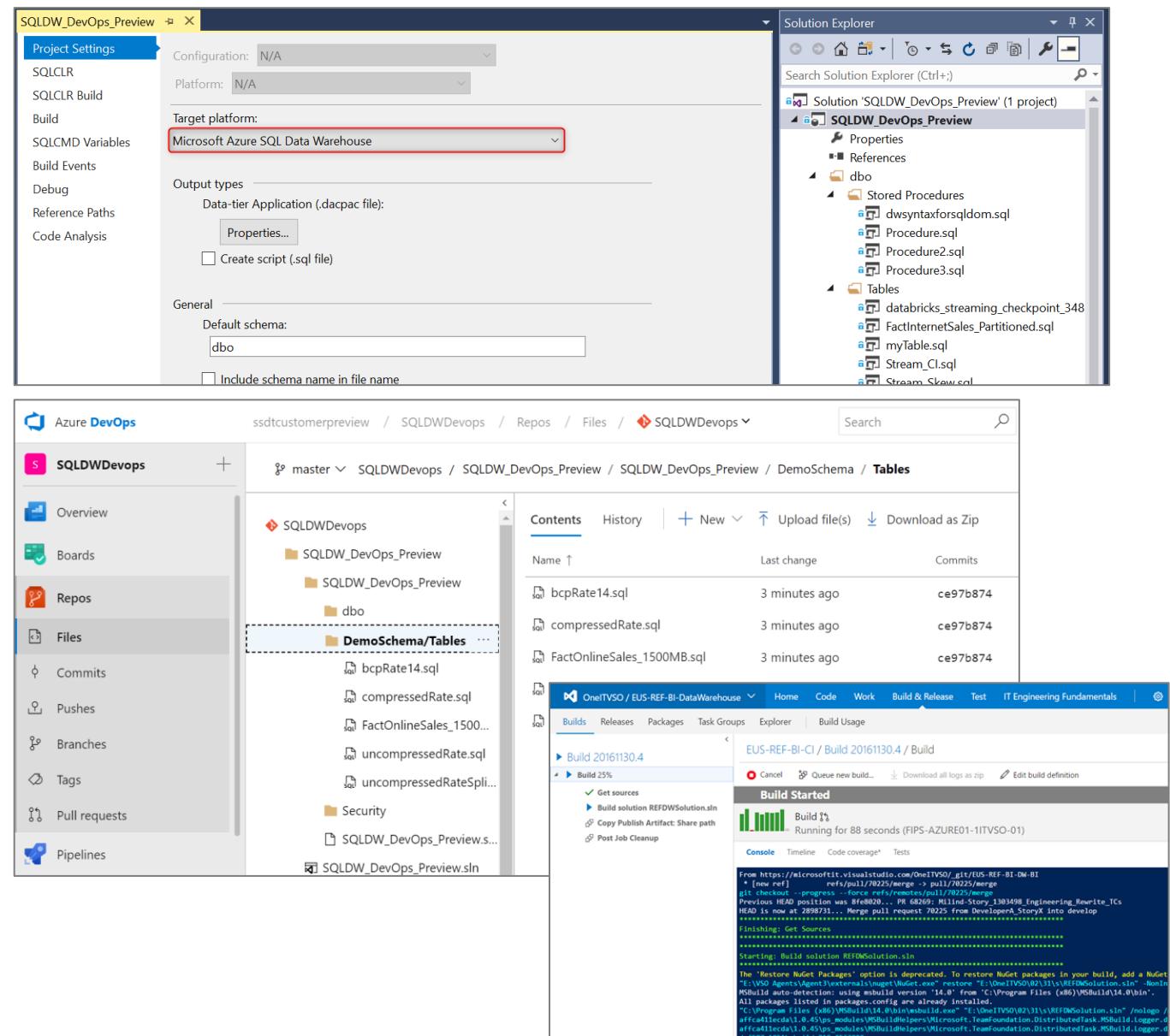
## Overview

Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.

## Benefits

Database project support includes first-class integration with Azure DevOps. This adds support for:

- Azure Pipelines** to run CI/CD workflows for any platform (Linux, macOS, and Windows)
- Azure Repos** to store project files in source control
- Azure Test Plans** to run automated check-in tests to verify schema updates and modifications
- Growing ecosystem of third-party integrations that can be used to complement existing workflows (Timetracker, Microsoft Teams, Slack, Jenkins, etc.)



# Azure Advisor recommendations

## Suboptimal Table Distribution

Reduce data movement by replicating tables

## Data Skew

Choose new hash-distribution key

Slowest distribution limits performance

## Cache Misses

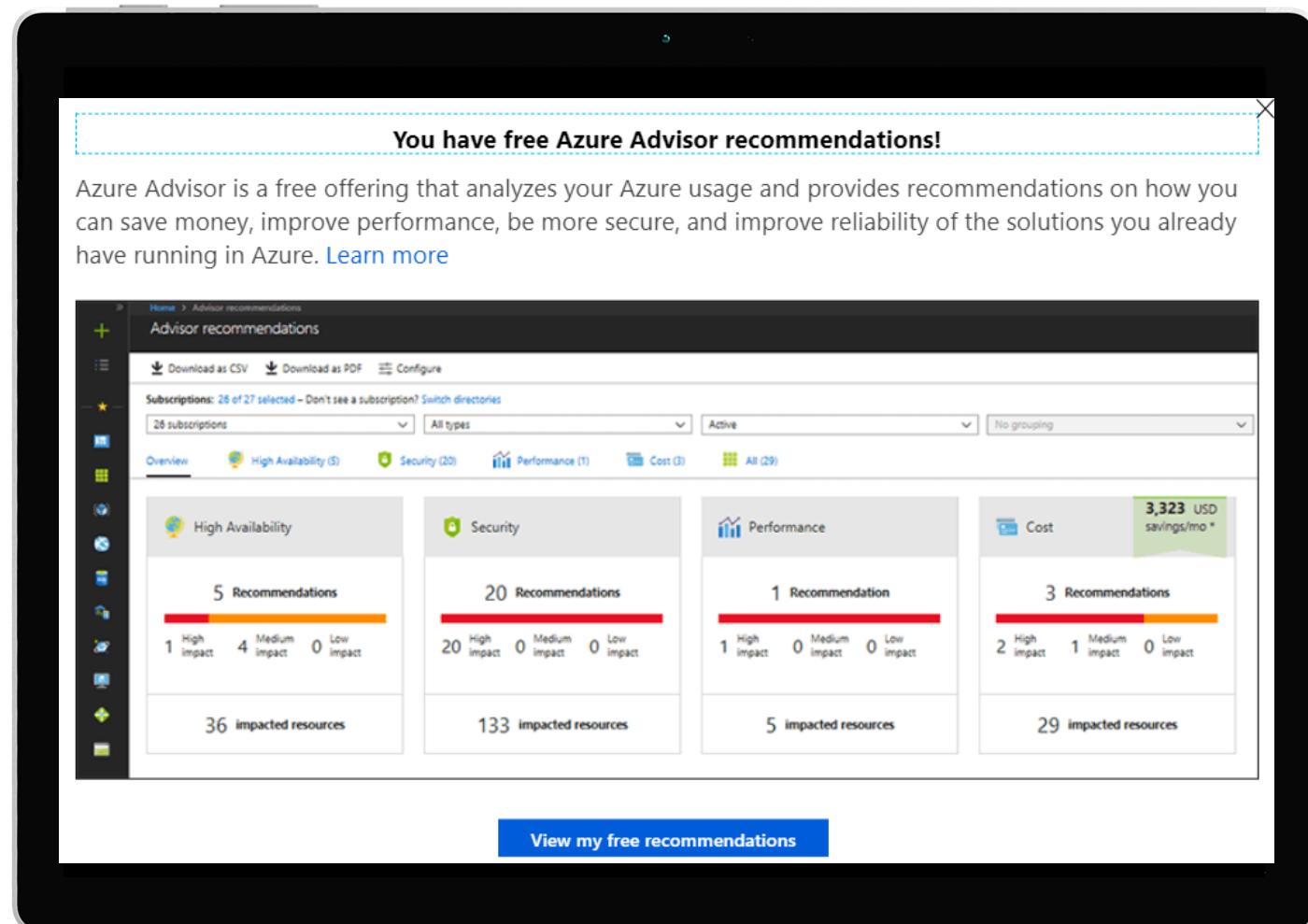
Provision additional capacity

## Tempdb Contention

Scale or update user resource class

## Suboptimal Plan Selection

Create or update table statistics



# Maintenance windows

## Overview

Choose a time window for your upgrades.

Select a primary and secondary window within a seven-day period.

Windows can be from 3 to 8 hours.

24-hour advance notification for maintenance events.

## Benefits

Ensure upgrades happen on your schedule.

Predictable planning for long-running jobs.

Stay informed of start and end of maintenance.

**Maintenance Schedule (preview)**

Maintenance on your data warehouse could occur once a week within one of two maintenance windows. Choose the primary and secondary windows that best suit your operational needs. If you would like to use the maintenance windows already defined, no action is required.

Maintenance will not take place outside these windows unless we notify you in advance.

**Choose primary window**

Saturday - Sunday  Tuesday - Thursday

Primary maintenance window	Secondary maintenance window
Day <small>i</small> Saturday	Day <small>i</small> Tuesday
Start time <small>i</small> 03:00 UTC	Start time <small>i</small> 13:00 UTC
Time window <small>i</small> 8 hours	Time window <small>i</small> 8 hours

**Schedule summary**

Primary maintenance window  
Saturday 03:00 UTC (8 hours)

Secondary maintenance window  
Tuesday 13:00 UTC (8 hours)

# Automatic statistics management

## Overview

Statistics are automatically created and maintained for SQL provisioned. Incoming queries are analyzed, and individual column statistics are generated on the columns that improve cardinality estimates to enhance query performance.

Statistics are automatically updated as data modifications occur in underlying tables. By default, these updates are synchronous but can be configured to be asynchronous.

Statistics are considered out of date when:

- There was a data change on an empty table
- The number of rows in the table at time of statistics creation was 500 or less, and more than 500 rows have been updated
- The number of rows in the table at time of statistics creation was more than 500, and more than  $500 + 20\%$  of rows have been updated

-- Turn on/off auto-create statistics settings

```
ALTER DATABASE {database_name}
```

```
SET AUTO_CREATE_STATISTICS { ON | OFF }
```

-- Turn on/off auto-update statistics settings

```
ALTER DATABASE {database_name}
```

```
SET AUTO_UPDATE_STATISTICS { ON | OFF }
```

-- Configure synchronous/asynchronous update

```
ALTER DATABASE {database_name}
```

```
SET AUTO_UPDATE_STATISTICS_ASYNC { ON | OFF }
```

-- Check statistics settings for a database

```
SELECT      is_auto_create_stats_on,  
            is_auto_update_stats_on,  
            is_auto_update_stats_async_on  
FROM        sys.databases
```



# Azure Synapse Analytics

## SQL Serverless

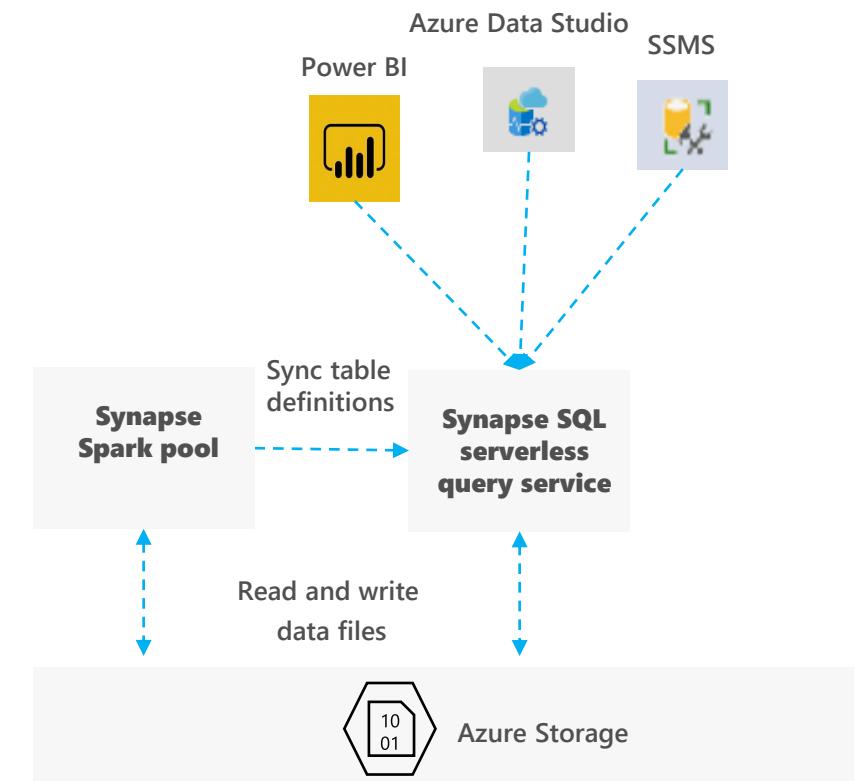
# SQL Serverless

## Overview

An interactive query service that enables you to use standard T-SQL queries over files in Azure storage.

## Benefits

- Use SQL to work with files on Azure storage
  - Directly query files on Azure storage using T-SQL
  - Logical Data Warehouse on top of Azure storage
  - Easy data transformation of Azure storage files
- Supports any tool or library that uses T-SQL to query data
- Automatically synchronize tables from Spark
- Serverless
  - No infrastructure, no upfront cost, no resource reservation
  - Pay only for query execution (per data processed)



# Recommended usage scenarios

## Quick data exploration

- Easily explore schema and data in files on Azure storage
- Supports various file formats (Parquet, CSV, JSON)
- Direct connector to Azure storage for large BI ecosystem

## Logical Data Warehouse

- Model raw files as virtual tables and views
- Use any tool that works with SQL to analyze files
- Use enterprise-grade security model

## Easy data transformation

- Transform CSV to parquet format
- Move data between containers and accounts
- Save the results of queries on external storage

# Easily explore files on storage

The screenshot illustrates the integration of Azure Storage and Azure Synapse Analytics. On the left, the Azure portal navigation bar shows 'Microsoft Azure | Synapse Analytics > internalsandboxwe5'. The main area is divided into two panes:

- Left Pane (File Explorer):** Shows the 'opendataset' folder under 'internalsandboxwe5'. It lists several parquet files and a new SQL script named 'New SQL script - Select TOP 100 rows'. A context menu is open over this script, with the option 'New SQL script - Select TOP 100 rows' highlighted with a red box.
- Right Pane (SQL Editor):** Displays the SQL script content:
 

```

1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://internalsandboxwe.dfs.core.windows.net/opendataset/holidays/part-00001-bd1aba93-a85a-4909-8bf4-f79afb6c946f-c000.snappy.parquet'
6         FORMAT='PARQUET'
7     ) AS [r];
8 
```

 The 'Connect to' dropdown is set to 'SQL on-demand' (highlighted with a red box). The 'Results' tab is selected, showing a table of data with columns: VENDORID, TPEPICKUPDATETIME, TPEPDROPOFFDATETIME, PASSENGERCOUNT, TRIPDISTANCE, PULOCATIONID, and DOLOCATIONID. The results show four rows for vendor VTS, with trip distances ranging from 0.55 to 2.94 miles.

# Easily query files in various formats

## Overview

Use OPENROWSET function to access data stored in various file formats

## Benefits

Enables you to read CSV, parquet, and JSON files

Provides unified T-SQL interface for all file types

Use standard SQL language to transform and analyze returned data

- Use JSON functions to get the data from underlying files.
- Use JSON functions to get data from PARQUET nested types

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
    FORMAT = 'CSV')
WITH (
    country_code VARCHAR(4),
    country_name VARCHAR(50),
    year INT,
    population INT
) AS nyc
```

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.parquet',
    FORMAT = 'PARQUET') AS nyc
```

```
SELECT TOP 10 *
    JSON_VALUE(jsonContent, '$.countryCode') AS country_code,
    JSON_VALUE(jsonContent, '$.countryName') AS country_name,
    JSON_VALUE(jsonContent, '$.year') AS year
    JSON_VALUE(jsonContent, '$.population') AS population
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/json/taxi/*.json',
    FORMAT='CSV',
    FIELDTERMINATOR = '0x0b',
    FIELDQUOTE = '0x0b',
    ROWTERMINATOR = '0x0b'
)
WITH ( jsonContent varchar(MAX) ) AS json_line
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

# Automatic schema inference

## Overview

OPENROWSET will automatically determine columns and types of data stored in external file.

## Benefits

No need to up-front analyze file structure to query the file  
OPENROWSET identifies columns and their types based on underlying file metadata.

Perfect solution for data exploration where schema is unknown.

Currently available only for parquet files.

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.parquet',
    FORMAT = 'PARQUET') AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

# Defined the query result schema inline

## Overview

Specify columns and types at query time.

## Benefits

Define result schema at query time in WITH clause.

No need for external format files.

Explicitly define exact return types, their sizes, and collations.

Improve performance by column elimination in parquet files.

```
SELECT TOP 10 *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
    FORMAT - 'CSV')
WITH (
    country_code VARCHAR(4),
    country_name VARCHAR(50),
    year INT,
    population INT
) AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

# Customize the content parsing to fit your case

## Overview

Uses OPENROWSET function to access data from various types of CSV files.

## Benefits

Ability to read CSV files with custom format

- With or without header row
- Handle any new-line terminator (Windows or Unix style)
- Use custom field terminator and quote character
- Read UTF-8 and UTF-16 encoded files
- Use only a subset of columns by specifying column position after column types

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) 2,
    [country_name] VARCHAR (100) 4,
    [year] smallint 7,
    [population] bigint 9
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

Second, fourth, seventh and ninth columns are returned

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

# Easily query multiple files, with wildcards

## Overview

Uses OPENROWSET function to access data from multiple files or folders using wildcards in path

## Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

```
SELECT YEAR(pickup_datetime) AS [year],  
       SUM(passenger_count) AS passengers_total,  
       COUNT(*) AS [rides_total]  
FROM OPENROWSET(  
    BULK 'https://XYZ.blob.core.windows.net/csv/taxi/year=*/month=1/*.parquet',  
    FORMAT = 'PARQUET') AS nyc  
GROUP BY YEAR(pickup_datetime)  
ORDER BY YEAR(pickup_datetime)
```

	year	passengers_total	rides_total
1	2001	14	10
2	2002	29	16
3	2003	22	16
4	2008	378	188
5	2009	594	353
6	2016	102093687	61758523
7	2017	184464988	113496932
8	2018	86272771	53925040
9	2019	37	29
...	2020	6	6

# Query partitioned data, using the folder structure

## Overview

Uses OPENROWSET function to access data partitioned in sub-folders

## Benefits

Use filepath() function to access actual values from file paths.

Eliminate sub-folders/partitions before the query starts execution

Query Spark/Hive partitioned data sets

```
SELECT  
    r.filepath(1) AS [year]  
    ,r.filepath(2) AS [month]  
    ,COUNT_BIG(*) AS [rows]  
FROM OPENROWSET(  
    BULK 'https://XYZ.blob.core.windows.net/year=*/month=/*/*.parquet',  
    FORMAT = 'PARQUET') AS [r]  
WHERE r.filepath(1) IN ('2017')  
    AND r.filepath(2) IN ('10', '11', '12')  
GROUP BY r.filepath(),r.filepath(1),r.filepath(2)  
ORDER BY filepath
```

year	month	rows
2017	10	9768815
2017	11	9284803
2017	12	9508276

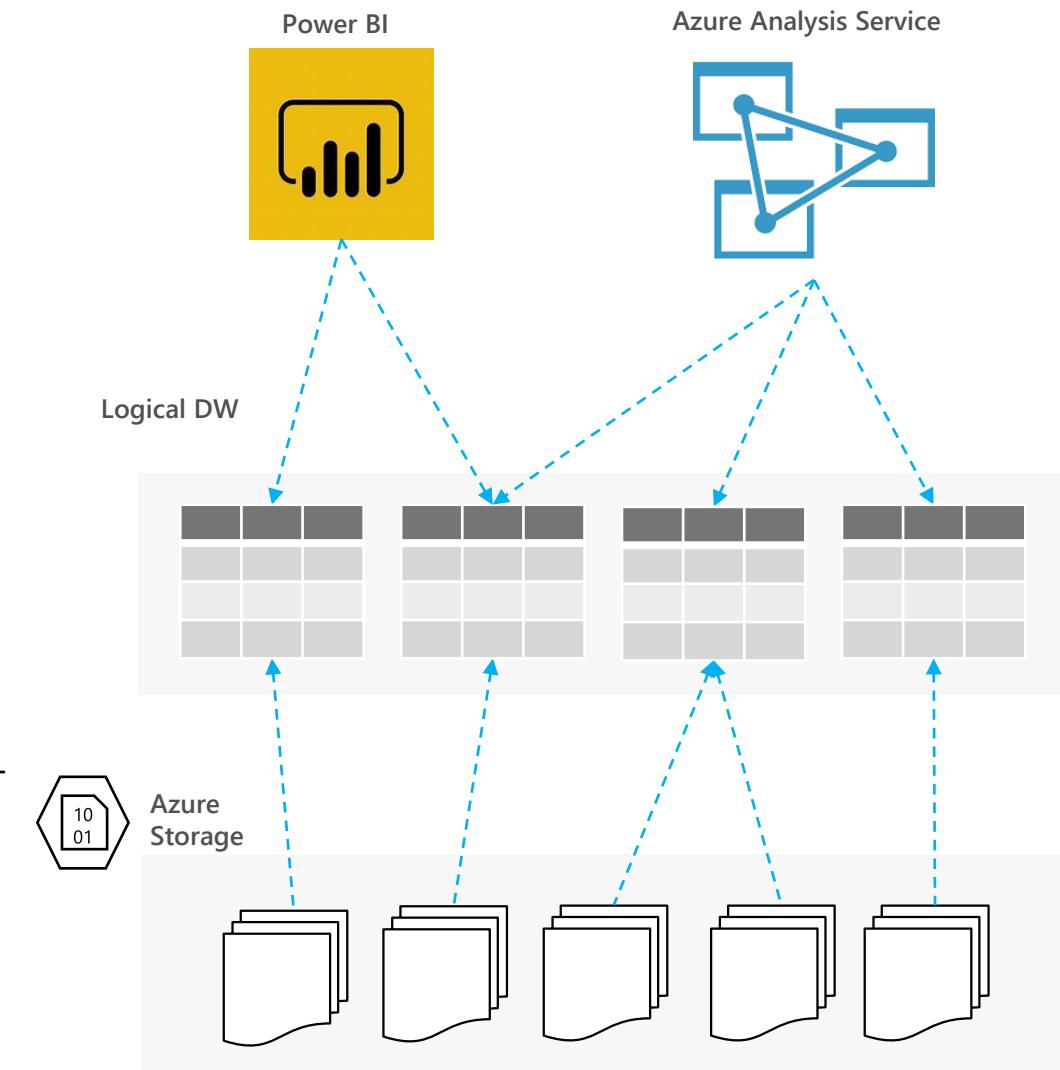
# Synapse SQL serverless as a logical data warehouse

## Overview

Logical relational layer on top of physical files in Azure Storage.

## Benefits

- Abstract physical storage and file formats using well understandable relational concepts such as tables and views.
- Direct connector to Azure storage for large ecosystem of BI tools
- BI tools that use SQL can work with files on storage
  - Analytic tools use external tables that represent proxy to actual files.
  - No need for custom connectors in BI tools.
- Provides complex data processing (joining and aggregation) on top of raw files.
- Apply enterprise-ready security model and access control using battle-tested SQL Server permission model on top of Azure storage files



# Logical Data Warehouse views

## Overview

SQL Serverless Logical Data Warehouse views are created on external files placed in customer Azure storage

## Benefits

Create SQL views on externally stored data

Access files using the view from various tools and language

Leverage rich T-SQL language to process and analyze data in external files exposed via views

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP VIEW IF EXISTS populationView
GO

CREATE VIEW populationView AS
SELECT *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/population/*.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5),
    [country_name] VARCHAR (100),
    [year] smallint,
    [population] bigint
) AS [r]
```

```
SELECT
    country_name, population
FROM populationView
WHERE
    [year] = 2019
ORDER BY
    [population] DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

# Creating views

The screenshot illustrates the process of creating a view in Azure Synapse Analytics using the SQL editor and visualizing the results.

**Top Left:** The main Azure Synapse Analytics interface shows the "internalsandbox..." workspace. The "Data" section is selected, displaying storage accounts, databases, and datasets. A "Develop" tab is open, showing the code for creating a view named "yellow\_2017". The code uses an OPENROWSET command to bulk load data from a Parquet file located at [https://internalsandboxwe.dfs.core.windows.net/opendataset/nyctlc/yellow/puYear=2017/\\*/\\*](https://internalsandboxwe.dfs.core.windows.net/opendataset/nyctlc/yellow/puYear=2017/*/*).

```

CREATE VIEW yellow_2017 AS
Select *
FROM
OPENROWSET(
    BULK 'https://internalsandboxwe.dfs.core.windows.net/opendataset/nyctlc/yellow/puYear=2017/\*/\*',
    FORMAT='PARQUET'
) AS [];

```

**Bottom Left:** A secondary "Results" tab shows the execution of the same query. The results are displayed in a chart view, where the X-axis represents the number of passengers (0 to 10) and the Y-axis represents the count (0 to 100M). The data shows a sharp peak at 1 passenger (~80M) and a smaller peak at 0 passengers (~10M).

**Top Right:** Another "Develop" tab shows the same view creation code. The results tab is selected, displaying the query results as a table. The table has three columns: (NO COLUMN NAME), PASSENGERCOUNT, and CNT. The data shows the count of trips for each passenger count from 0 to 5.

(NO COLUMN NAME)	PASSENGERCOUNT	CNT
2017	0	166086
2017	1	81034075
2017	2	16545571
2017	3	4748869
2017	4	2257813
2017	5	5407319

**Bottom Right:** The same "Develop" tab is shown again, but the results tab is not visible. Instead, the chart view is selected, showing the same data as the bottom-left chart.

# Logical Data Warehouse - tables

## Overview

Create external tables that reference external files in your SQL Serverless Logical Data Warehouse

## Benefits

Create external tables that reference set of files on Azure storage.

Join and transform multiple tables in the same query.

Enables you to analyze external files with the same experience that you have in classic databases.

Manage column statistics in external tables.

Manage access rights per table.

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP TABLE IF EXISTS dbo.Population
GO

CREATE EXTERNAL TABLE dbo.Population (
    country_code VARCHAR (5) COLLATE Latin1_General_BIN2,
    country_name VARCHAR (100) COLLATE Latin1_General_BIN2,
    year smallint,
    population bigint
)
WITH(
    LOCATION = '/csv/population/population-* .csv',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureCSVFormat
)
```

```
CREATE STATISTICS stat_country_name
ON dbo.Population(country_name);
```

```
SELECT
    country_name, population
FROM population
WHERE year = 2019
ORDER BY population DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

# Easy data transformation

## Overview

Easily perform data transformations of Azure Storage files using SQL queries

Optimize data pipeline - achieve more using SQL serverless

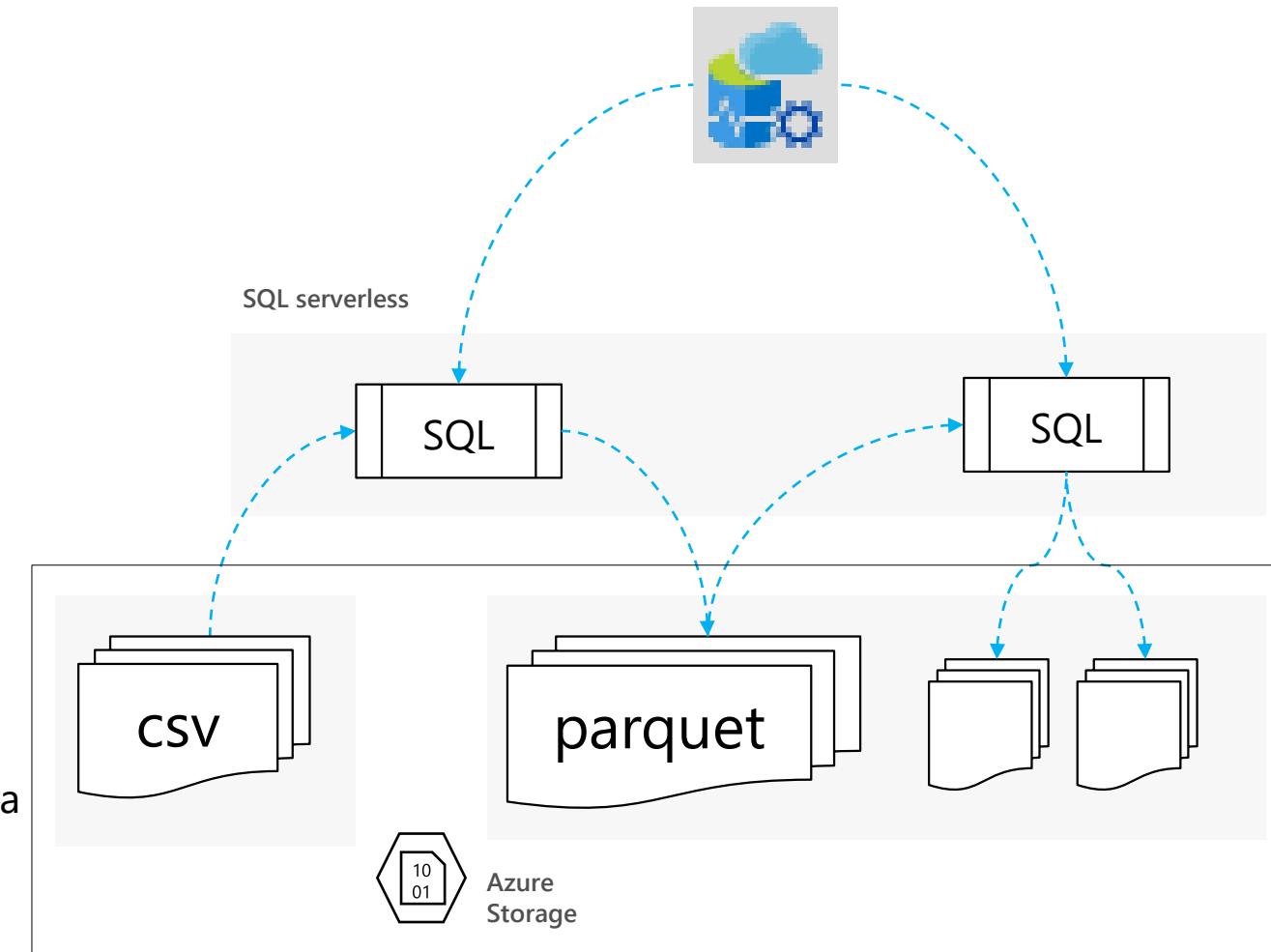
## Benefits

Single statement transformations:

- convert CSV or JSON files to Parquet
- copy files from one storage account to another
- re-partition data to new location(s)
- store results of your query on Azure Storage

SQL ETL pipelines

- Use SQL commands to transform data
- Chain SQL statement for build ETL process
- Materialize reports created on the current snapshot of data



# Easy data transformation with CETAS

## Overview

Create external tables as select (CETAS) enables you to easily transform data and store the results of query on Azure storage

## Benefits

Select any data set and store it in parquet format.

Pre-calculate and store results of query and store them permanently on Azure storage.

Use saved data using external table.

Improve performance of your reports by permanently storing the result based on current snapshot of data as parquet files.

```
-- copy CSV dataset into parquet data set
CREATE EXTERNAL TABLE parquet.Population
WITH(
    LOCATION = '/parquet/population',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureParquetFormat )
AS
SELECT *
FROM csv.Population

-- pre-create report using new parquet data-set
CREATE EXTERNAL TABLE parquet.PopulationByMonth2017
WITH(
    LOCATION = '/parquet/population/bymonth/2017',
    DATA_SOURCE = MyAzureStorage,
    FILE_FORMAT = MyAzureParquetFormat )
AS
SELECT month = p.month, population = COUNT ( p.population )
FROM parquet.Population p
WHERE p.year = 2017
GROUP BY p.month

-- Reporting tools can now directly read data from pre-created report
SELECT *
FROM parquet.PopulationByMonth2017
```

# Automatic syncing of Spark tables

## Overview

Tables created in Spark pool are automatically created as external tables that reference external files in your SQL serverless Logical Data Warehouse

## Benefits

Tables designed using Spark languages are immediately available in SQL serverless.

Schema definition matches original

Spark table updates are applied in SQL serverless

No need to manually create SQL tables that match Spark tables

Spark and SQL serverless tables references the same external files.

The screenshot shows the Azure Synapse Analytics studio interface. On the left, there's a sidebar with icons for Connections, Servers, Databases, Tables, Columns, Keys, Constraints, and External Tables. The 'Tables' section is currently selected, showing two external tables: 'dbo.data1017' and 'dbo.data1017' (External). The 'Columns' section is also visible, listing columns like ExtractId, DayOfWeekID, DayOfWeekDescr, DayOfWeekDescrShort, ExtractDateTime, LoadTS, and DeltaActionCode.

In the main area, there are two panes. The top pane is titled 'Create external table' and contains the following SQL code:

```
1 %%sql
2 create table data1017 using parquet
3 location 'abfss://container@demostorage.dfs.core.windows.net/data/'
```

The bottom pane shows a SQL query in the 'SQLQuery\_1' editor:

```
1 SELECT TOP (10) [ExtractId]
2 , [DayOfWeekID]
3 , [DayOfWeekDescr]
4 , [DayOfWeekDescrShort]
5 , [ExtractDateTime]
6 , [LoadTS]
7 , [DeltaActionCode]
8 FROM [default]..[data1017]
```

The results pane displays the data from the 'data1017' table, showing six rows of data with columns ExtractId, DayOfWeekID, DayOfWeekDescr, DayOfWeekDescrShort, ExtractDateTime, and LoadTS.

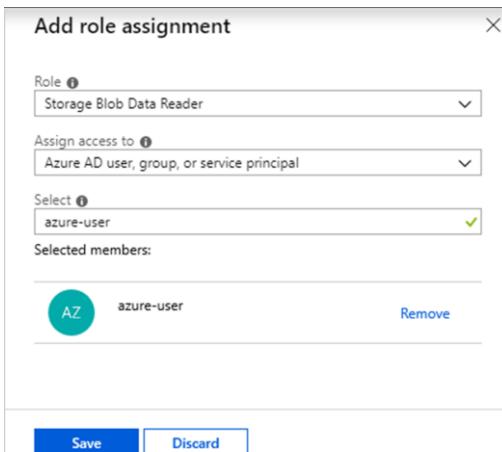
# Access control

## Overview

Enterprise-grade security model enables you to control who can access data.

## Benefits

- Use Azure Active Directory users or native SQL logins.
- SAS tokens, AAD or workspace identity access
- Specify access methods in credential
- Grant access to storage by referencing storage credential
- Enable some logins to access external tables
- Add AAD role assignments directly on Azure storage.



```
-- create built-in logins username/password
CREATE LOGIN login1 WITH PASSWORD = '<some strong password>'

-- create logins from your Azure Active Directory tenant
CREATE LOGIN login2 FROM EXTERNAL_PROVIDER

-- enable impersonation using workspace Managed Identity
CREATE CREDENTIAL [ManagedIdentity]
WITH IDENTITY = 'Managed Identity'

-- enable access to specified storage using SAS token
CREATE CREDENTIAL [https://XXX.blob.core.windows.net/csv]
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'sv=2014-02-14&sr=b&si=TestPolicy&sig=o%2B5%2FOC%2BLm7tWWft'

-- grant login1 to use SAS token defined in credential for storage account
GRANT REFERENCES CREDENTIAL::[https://XXX.blob.core.windows.net/csv]
TO LOGIN = 'login1'

-- grant login2 to use Managed Identity
GRANT REFERENCES CREDENTIAL::[ManagedIdentity]
TO LOGIN = 'login2'

-- grant login2 to select external data via table
GRANT SELECT ON OBJECT::[dbo.population] TO LOGIN = 'login2'
```

# Synapse SQL serverless - clients and tools

## Tools

Web IDE - Synapse Studio

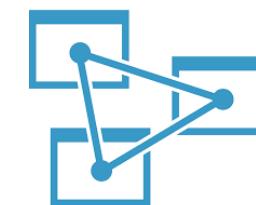


Client tools - Azure Data Studio, SSMS,



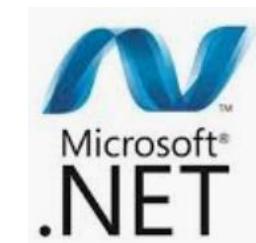
Any tool/library that uses standard SQL can access  
SQL serverless

- PowerBI
- Azure Analytic Services
- Client languages and drivers that works with Azure SQL can be used to access SQL serverless

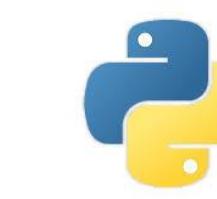


## Limitations

Max query duration 30min



~10TB max data that can be processed per query



Use desktop tools (ADS, SSMS) instead of Synapse Studio if  
you are returning multi-GB of data per query. Web interface  
is not designed for huge data exports.

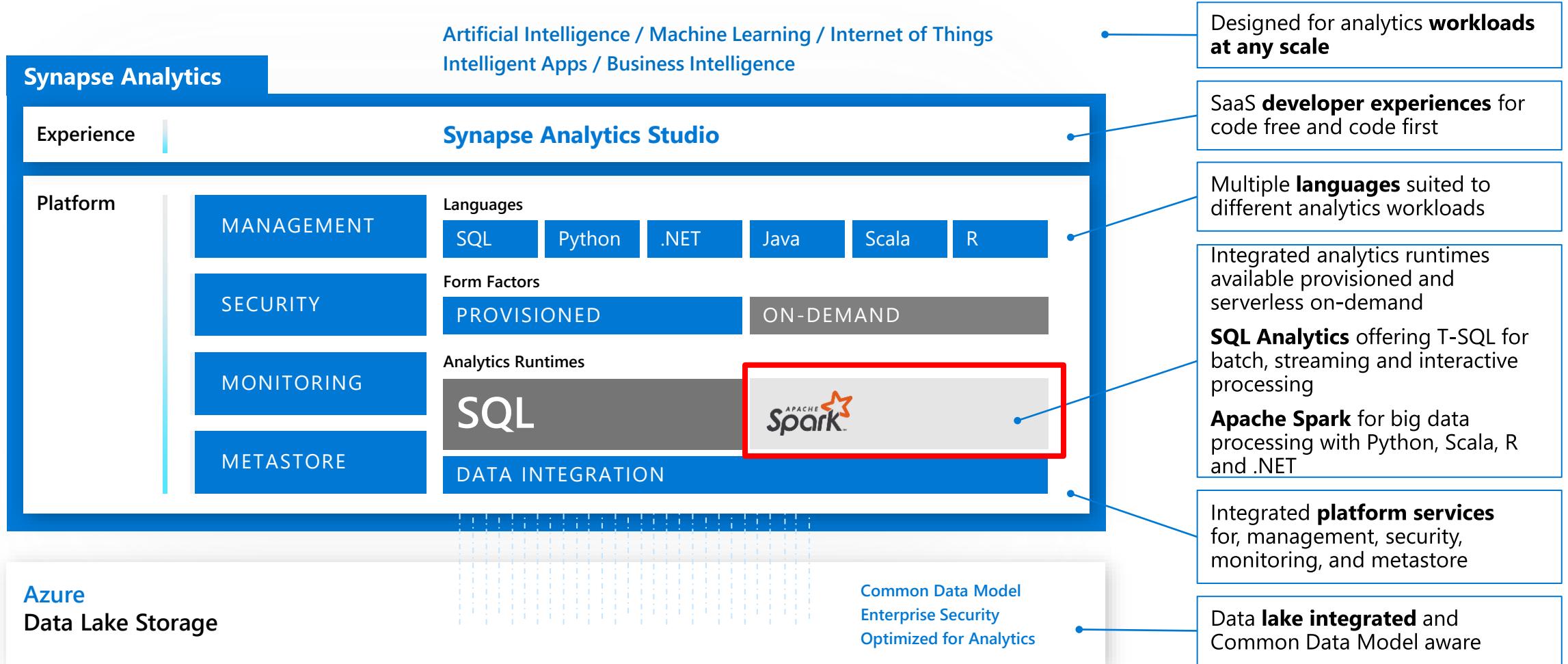


# Azure Synapse Analytics

## Apache Spark

# Azure Synapse Analytics

Limitless analytics service with unmatched time to insight





# Azure Synapse Apache Spark - Summary

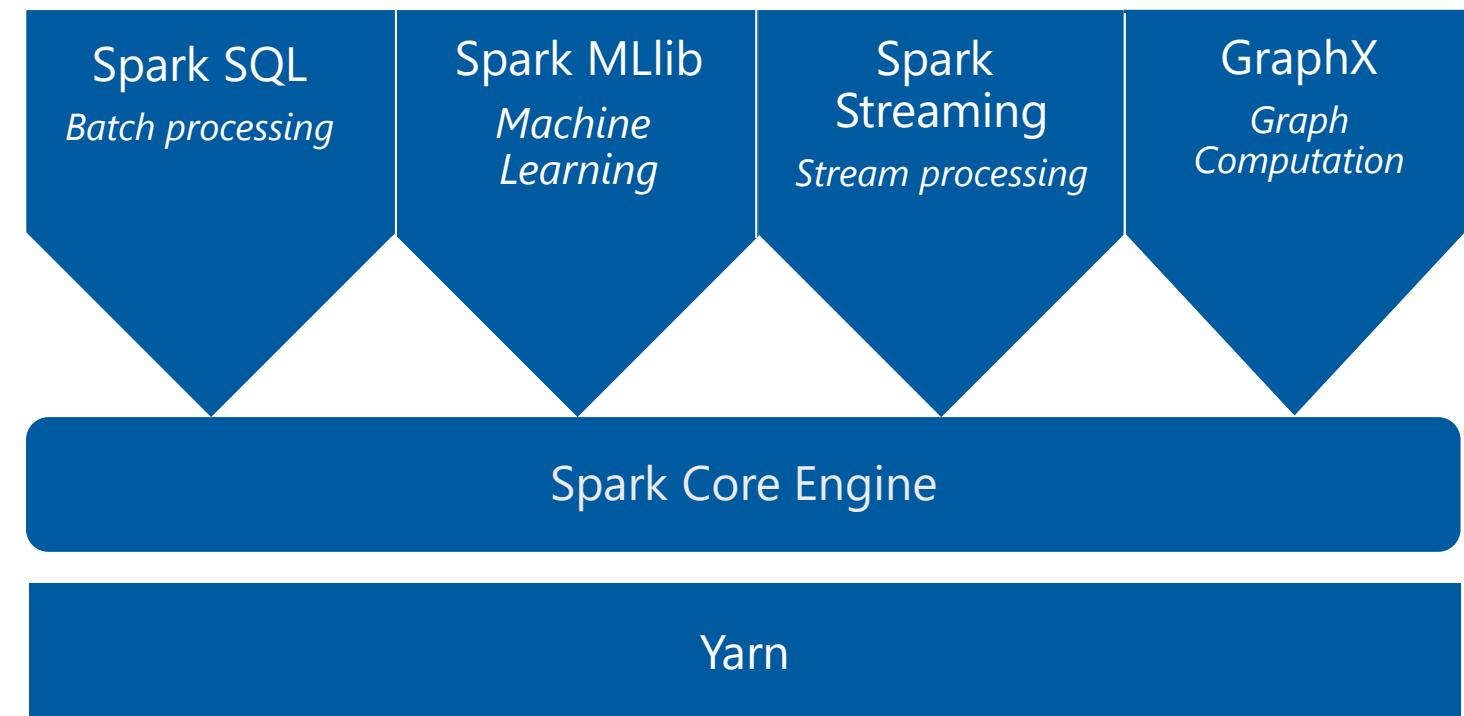
- **Apache Spark 2.4 derivation**
  - Linux Foundation Delta Lake 0.5 support
  - .Net Core 3.0 support
  - Python 3.6 + Anacondas support
- **Tightly coupled to other Azure Synapse services**
  - Integrated security and sign on
  - Integrated Metadata
  - Integrated and simplified provisioning
  - Integrated UX including nteract based notebooks
  - Fast load of SQL Analytics pools
- **Core scenarios**
  - Data Prep/Data Engineering/ETL
  - Machine Learning via Spark ML and Azure ML integration
  - Extensible through library management
- **Efficient resource utilization**
  - Fast Start
  - Auto scale (up and down)
  - Auto pause
  - Min cluster size of 3 nodes
- **Multi Language Support**
  - .Net (C#), PySpark, Scala, Spark SQL, Java

# Apache Spark

An unified, open source, parallel, data processing framework for Big Data Analytics

## Spark Unifies:

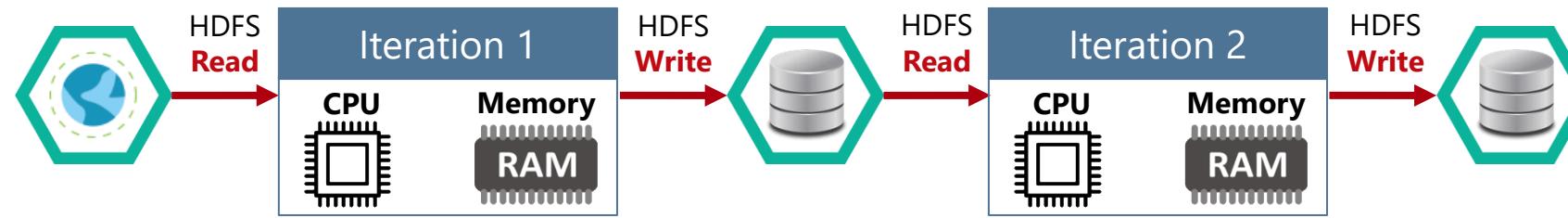
- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing



<http://spark.apache.org>

# Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O

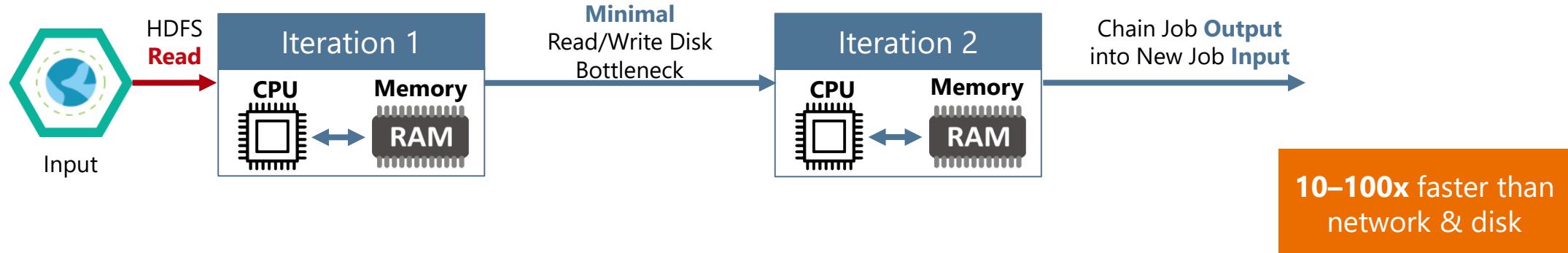


# Motivation for Apache Spark

Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of **(slow) disk I/O**

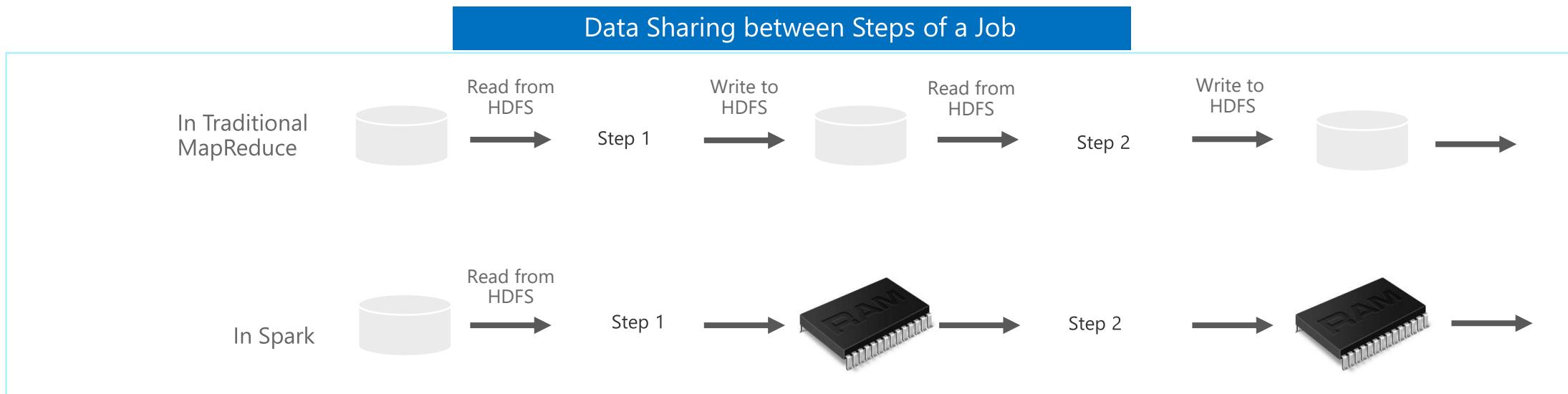


Solution: Keep data **in-memory** with a new distributed execution engine



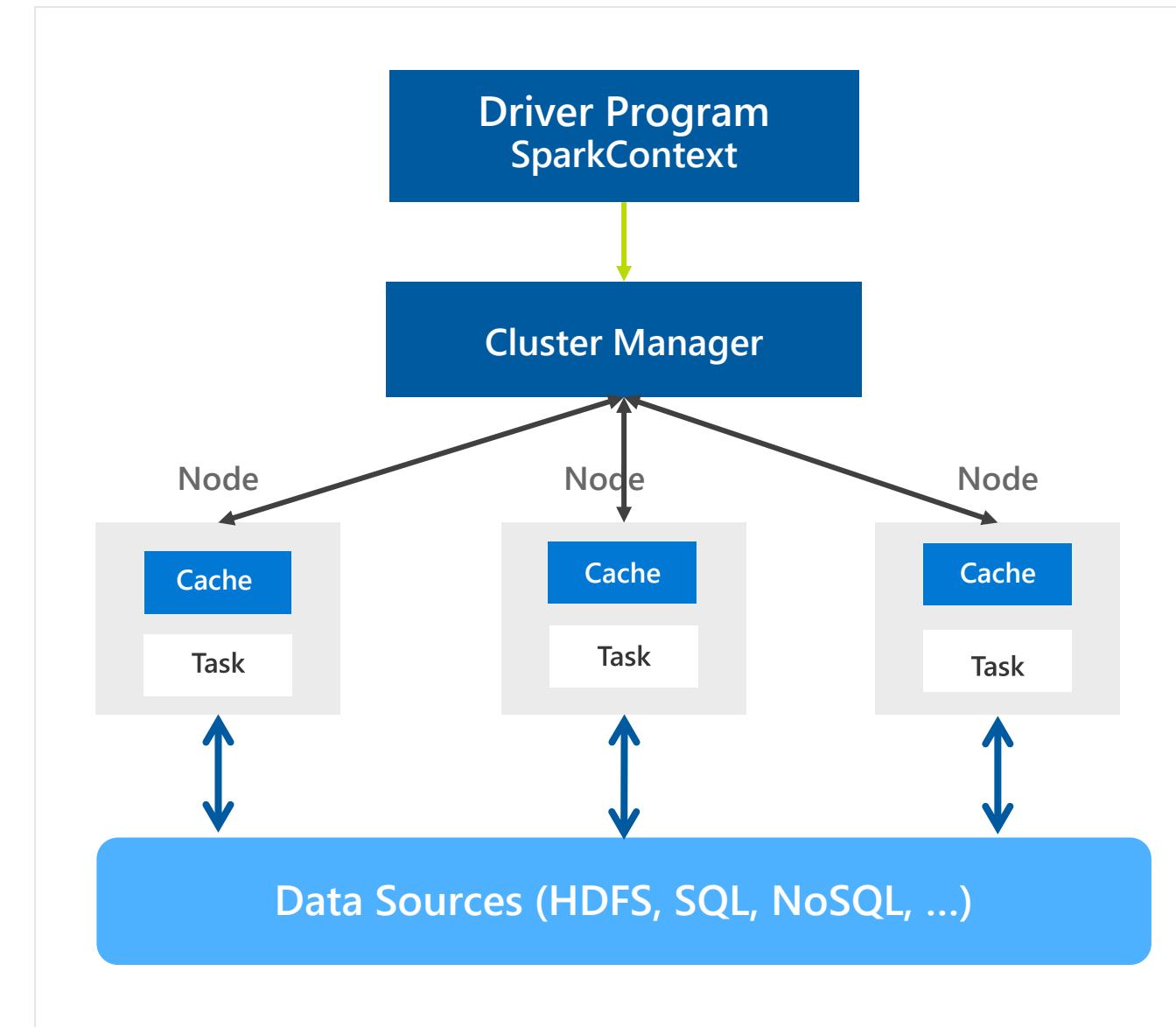
# What makes Spark fast

- **In-memory cluster computing:** Spark provides primitives for *in-memory* cluster computing. A Spark job can *load and cache* data into memory and query it repeatedly (iteratively) much quicker than disk-based systems.
- **Scala Integration:** Spark integrates into the Scala programming language, letting you manipulate distributed datasets like local collections. No need to structure everything as map and reduce operations
- **Faster Data-sharing:** Data-sharing between operations is faster as data is in-memory:
  - In (traditional) Hadoop data is shared through HDFS which is expensive. HDFS maintains three replicas.
  - Spark stores data in-memory *without any replication*.



# General Spark Cluster Architecture

- 'Driver' runs the user's 'main' function and executes the various parallel operations on the worker nodes.
- The results of the operations are collected by the driver
- The worker nodes read and write data from/to Data Sources including HDFS.
- Worker node also cache transformed data in memory as RDDs (Resilient Data Sets).
- Worker nodes and the Driver Node execute as VMs in public clouds (AWS, Google and Azure).



# Spark Component Features

## Spark SQL

- Unified data access: Query structured data sets with SQL or DataFrame APIs
- Fast, familiar query language across all your enterprise data
- Use BI tools to connect and query via JDBC or ODBC drivers

## Mlib/SparkML

- Predictive and prescriptive analytics
- Machine learning algorithms for:
  - Clustering
  - Classification
  - Regression
  - etc.
- Smart application design from pre-built, out-of-the-box statistical and algorithmic models

## Spark Streaming

- Micro-batch event processing for near-real time analytics
- e.g. Internet of Things (IoT) devices, Twitter feeds, Kafka (event hub), etc.
- Spark's engine drives some action or outputs data in batches to various data stores

## GraphX

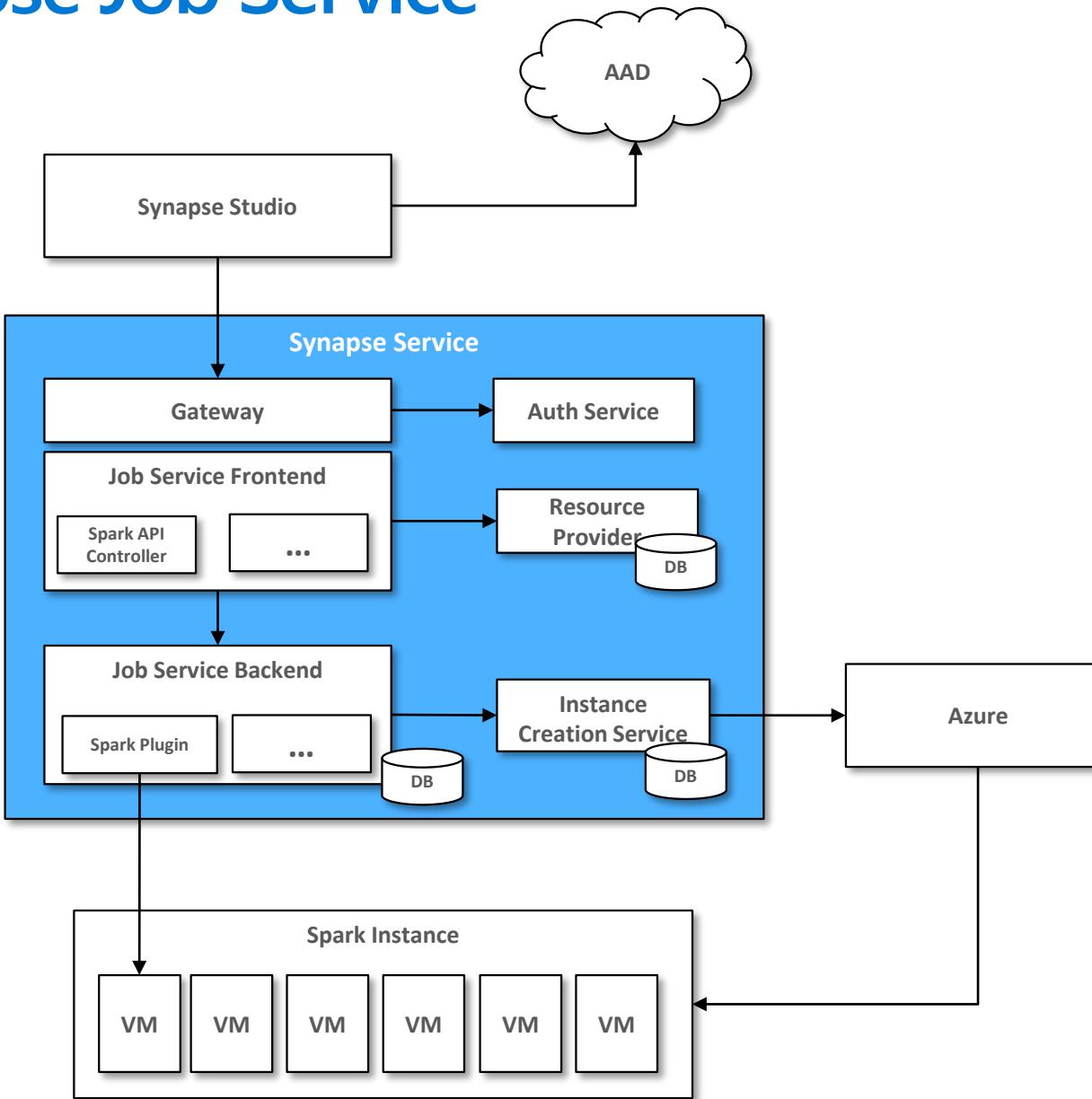
- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.



# Azure Synapse Apache Spark

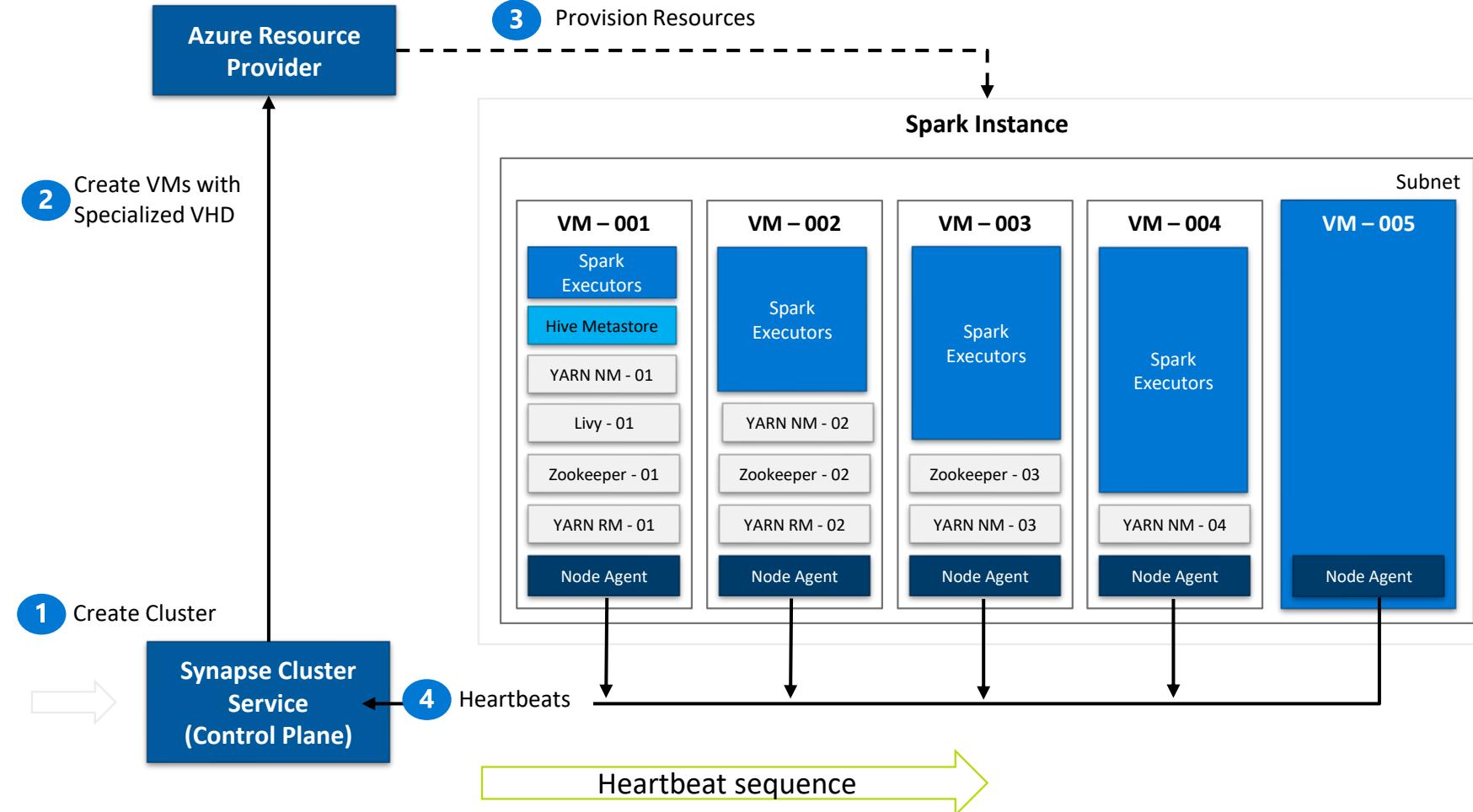
## Architecture Overview

# Synapse Job Service



- User creates Synapse Workspace and Spark pool and launches Synapse Studio.
- User attaches Notebook to Spark pool and enters one or more Spark statements (code blocks).
- The Notebook client gets user token from AAD and sends a Spark session create request to Synapse Gateway.
- Synapse Gateway authenticates the request and validates authorizations on the Workspace and Spark pool and forwards it to the Spark (Livy) controller hosted in Synapse Job Service frontend.
- The Job Service frontend forwards the request to Job Service backend that creates two jobs – one for creating the cluster and the other for creating the Spark session.
- The Job service backend contacts Synapse Resource Provider to obtain Workspace and Spark pool details and delegates the cluster creation request to Synapse Instance Service.
- Once the instance is created, the Job Service backend forwards the Spark session creation request to the Livy endpoint in the cluster.
- Once the Spark session is created the Notebook client sends Spark statements to the Job Service frontend.
- Job Service frontend obtains the actual Livy endpoint for the cluster created for the particular user from the backend and sends the statement directly to Livy for execution.

# Synapse Spark Instances



1. Synapse Job Service sends request to Cluster Service for creating BBC clusters per the description in the associated Spark pool.
2. Cluster Service sends request to Azure using Azure SDK to create VMs (required plus additional) with specialized VHD.
3. The specialized VHD contains bits for all the services that are required by the Cluster type (for e.g. Spark) with prefetch instrumentation.
4. Once VM boots up, the Node Agent sends heartbeat to Cluster Service for getting node configuration.
5. The nodes are initialized and assigned roles based on their first heartbeat.
6. Extra nodes get deleted on first heartbeat.
7. After Cluster Service considers the cluster ready, it returns the Livy endpoint to the Job Service.

# Creating a Spark pool (1 of 2)

Provision Spark Pool through Azure Portal with default settings or per requirements

Basic Settings – Minimum details required from user

Home > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > Create Apache Spark pool

## Create Apache Spark pool

Basics \* Additional settings \* Tags Summary

Create a Synapse Analytics Apache Spark pool with your preferred configurations. Complete the Basics tab then go to Review + create to provision with smart defaults, or visit each tab to customize.

Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name \*

Enter Apache Spark pool name

Node size family

MemoryOptimized

Node size \*

Medium (8 vCPU / 64 GB)

Autoscale \* ⓘ

Enabled  Disabled

Number of nodes \*

3  40

Only required field from user

Default Settings

# Creating a Spark pool (2 of 2) - optional

Additional Settings offer optional settings to customize Spark pool

Customize component versions, auto-pause

Import libraries by providing text file containing library name and version

Home > prlangadws2 > Create Apache Spark pool

Create Apache Spark pool

Basics \* Additional settings \* Tags Summary

Customize additional configuration parameters including autoscale and component versions.

**Auto-pause**

Enter required settings for this Apache Spark pool, including setting auto-pause and picking versions.

Auto-pause \* ⓘ Enabled Disabled

Number of minutes idle \* 15

**Component versions**

Select the Apache Spark version for your Apache Spark pool.

Component	Version
Apache Spark *	2.4
Python	3.6.1
Scala	2.11.12
Java	1.8.0_222
.NET Core	3.1
.NET for Apache Spark	0.10.0
Delta Lake	0.5.0

**Packages**

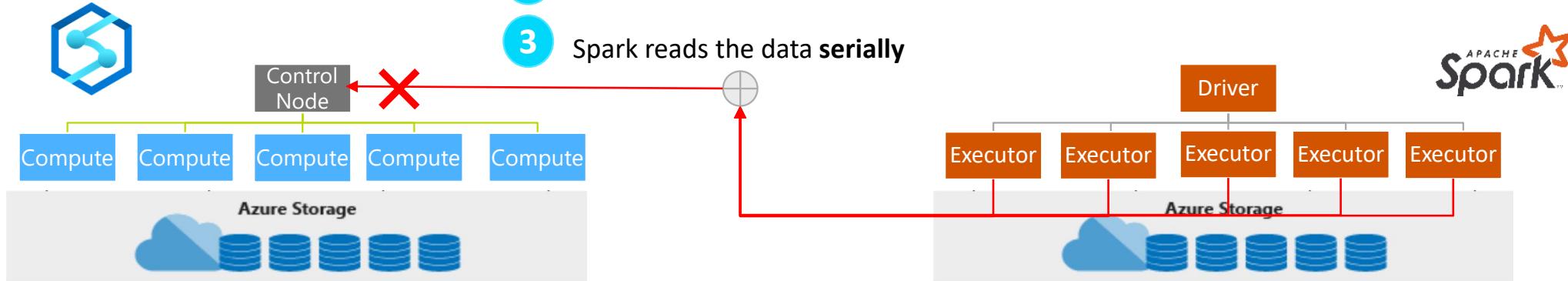
Upload environment configuration file ("PIP freeze" output).

File upload Select a file

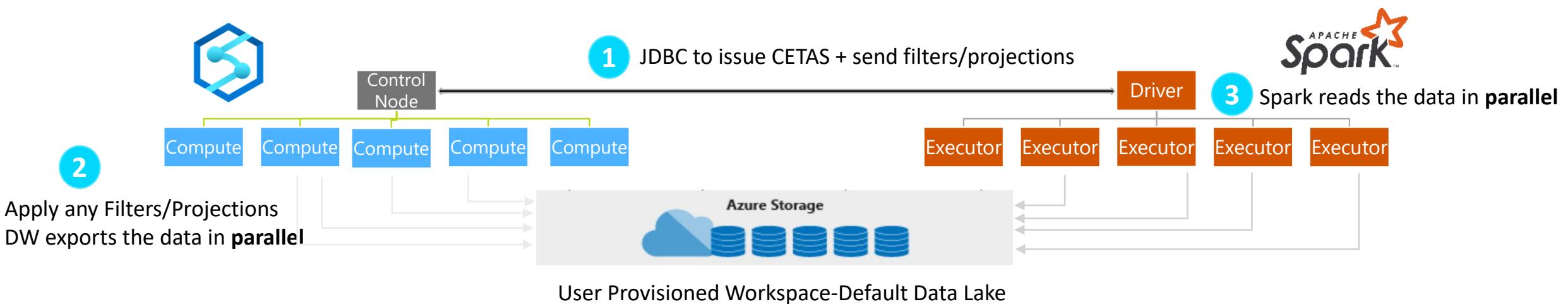
Upload

Review + create < Previous Next: Tags >

## Existing Approach: JDBC



## New Approach: JDBC and Polybase



# Code-Behind Experience

## Existing Approach

```
val jdbcUsername = "<SQL DB ADMIN USER>"  
val jdbcPwd = "<SQL DB ADMIN PWD>"  
val jdbcHostname = "servername.database.windows.net"  
val jdbcPort = 1433  
val jdbcDatabase = "<AZURE SQL DB NAME>"  
  
val jdbc_url =  
  s"jdbc:sqlserver://${jdbcHostname}:${jdbcPort};database=${jdbcDatabase};"  
  encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.databas  
e.windows.net;loginTimeout=60;"  
  
val connectionProperties = new Properties()  
  
connectionProperties.put("user", s"${jdbcUsername}")  
connectionProperties.put("password", s"${jdbcPwd}")  
  
val sqlTableDf = spark.read.jdbc(jdbc_url, "dbo.Tbl1", connectionProperties)
```

## New Approach

```
// Construct a Spark DataFrame from SQL Provisioned  
var df = spark.read.sqlAnalytics("sql1.dbo.Tbl1")  
  
// Write the Spark DataFrame into SQL Provisioned  
df.write.sqlAnalytics("sql1.dbo.Tbl2")
```

# Create Notebook on files in storage

The screenshot illustrates the process of creating a Notebook on files stored in Azure Storage. The interface is divided into two main sections:

- Left Panel (Data Explorer):** Shows the Azure Data Explorer sidebar with options like Home, Data, Develop, Orchestrate, Monitor, and Manage. The Data section is selected. Below it, Storage accounts are listed, including prlangaddemo (Primary) which contains filesystem, holidaydatacontainer, isdweatherdatacontainer, nyctic, prlangaddemo, tempdata, tmpcontainer, and wwidporters. A folder named labignite is also visible.
- Right Panel (Job History and Notebook View):** This panel shows the results of a previous job execution. It includes a Job execution history table and a preview of a dataset named "nyctic".

A red arrow points from the "New notebook" option in the context menu of a file in the Storage account to the right panel, indicating the transition from file selection to notebook creation.

**Job Execution History:**

ID	DESCRIPTION	STATUS	STAGES	TASKS	SUBMISSION TIME	DURATION
Job 0	load at NativeMethodAccessorImpl.java:0	Succeeded	1/1		11/14/2019, 9:56:49 AM	7s
Job 1	showString at NativeMethodAccessorImpl.java:0	Succeeded	1/1		11/14/2019, 9:56:58 AM	1s
Job 2	showString at NativeMethodAccessorImpl.java:0	Succeeded	1/1		11/14/2019, 9:56:59 AM	11s

**Dataset Preview (nyctic):**

Vendor ID	Dropoff Date Time	Pickup Date Time	Passenger Count	Trip Distance	Pickup Location ID	Dropoff Location ID	Start Lon	Start Lat	End Lon	End Lat
2	2015-02-28 23:53:18	2015-03-01 00:00:29	6	1.63	null	null	-74.00084686279297	40.73069381713867	-73.9841537475586	40.74470520019531
1	N	1	7.5	0.5	0.5	1	0.3	1.76	0.0	10.56
1	2015-03-01 19:21:05	2015-03-28 19:28:31	1	2.2	null	null	-73.97765350341797	40.763160705566406	-73.95502471923828	40.78600311279297
1	N	1	8.5	0.0	0.5	1	0.3	2.3	0.0	11.6
2	2015-02-28 23:53:19	2015-03-01 00:12:08	5	3.23	null	null	-73.96012878417969	40.76215744018555	-73.9881591796875	40.72818896484375
1	N	1	14.5	0.5	0.5	1	0.3	4.74	0.0	28.54
1	2015-03-28 19:21:05	2015-03-28 19:37:02	1	2.1	null	null	-73.98143005371094	40.7815055847168	-74.000891552734375	40.76177215576172



View results in chart format

SQL support

The screenshot shows the Azure Synapse Analytics workspace interface. On the left, there's a sidebar with 'Develop' selected, showing a list of notebooks. A red box highlights the 'Language' dropdown set to 'PySpark (Python)'. In the main area, Cell 1 contains PySpark code for reading data from Azure Blob Storage. Cell 2 shows a command to create a temporary view named 'seattlesafety'. Cell 3 contains a SQL query 'SELECT \* FROM seattlesafety' with a red box around it, followed by a pie chart visualization titled 'Aid Response' with a legend listing categories like 'Automatic Fire Alarm False', 'Medic Response, 7 per Rule', 'Aid Response Yellow', etc. A red arrow points from the 'View results in chart format' text to the pie chart. Cell 4 contains code for writing the DataFrame to a CSV file.

```

[3]
1 # Azure storage access info
2 blob_account_name = "azuredopendatastorage"
3 blob_container_name = "citydatacontainer"
4 blob_relative_path = "Safety/Release/city=Seattle"
5 blob_sas_token = r""
6
7 # Allow SPARK to read from Blob remotely
8 wasbs_path = 'wasbs://%' % (blob_container_name, blob_account_name, blob_relative_path)
9 spark.conf.set( 'fs.azure.sas.%s.blob.core.windows.net' % (blob_container_name, blob_account_name), blob_sas_token)
10
11 # SPARK read parquet, note that it won't load any data yet
12 seasafety_df = spark.read.parquet(wasbs_path)

Command executed in 2mins 18s 412ms by euang on 11-22-2019 00:44:52.415 -08:00

Job execution In progress Spark 1 executors 4 cores
ID DESCRIPTION STATUS STAGES TASKS SUBMISSION TIME DURATION
Job 0 parquet at NativeMethodAccessImpl.java:0 In progress 0/1 (1 active) 11/22/2019, 12:44:46 AM 13m43s

[5] 1 seasafety_df.createOrReplaceTempView('seattlesafety')

Command executed in 2s 835ms by euang on 11-22-2019 00:53:37.321 -08:00

Cell 3
1 display(spark.sql('SELECT * FROM seattlesafety'))
Command executed in 11s 526ms by euang on 11-22-2019 00:58:21.241 -08:00

View Table Chart
Aid Response
Longitude
Trans to AMR
Medic Response
Automatic Fire Alarm Resd
Auto Fire Alarm
IRED 1 Unit
Automatic Medical Alarm
Motor Vehicle Accident
Medic Response, 6 per Rule
MVI - Motor Vehicle Incident
Aid Response Yellow
Medic Response, 7 per Rule
Automatic Fire Alarm False
Chart type pie chart
X axis column
Y axis columns longitude
Aggregation COUNT
Y axis label Total
X axis label category
Apply Cancel
[7] 1 seasafety_df.coalesce(1).write.csv('abfss://default@euangsypnsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')

```

Develop + Discard all

Data Download... \* NYCTaxi\_Docs... \* Language PySpark (Python)

Cell 9

```

1 # Creating a temp table allows easier manipulation during the session, they are not persisted between sessions,
2 # for that write the data to storage like above.
3 sampled_taxi_df.createOrReplaceTempView("nytaxi")

```

**Exploratory Data Analysis**

Look at the data and evaluate its suitability for use in a model, do this via some basic charts focussed on tip values and relationships.

```

1 #The charting package needs a Pandas dataframe or numpy array do the conversion
2 sampled_taxi_pd_df = sampled_taxi_df.toPandas()
3
4 # Look at tips by amount count histogram
5 ax1 = sampled_taxi_pd_df['tipAmount'].plot(kind='hist', bins=25, facecolor='lightblue')
6 ax1.set_title('Tip amount distribution')
7 ax1.set_xlabel('Tip Amount ($)')
8 ax1.set_ylabel('Counts')
9 plt.subtitle('')
10 plt.show()
11
12 # How many passengers tip'd by various amounts
13 ax2 = sampled_taxi_pd_df.boxplot(column=['tipAmount'], by=['passengerCount'])
14 ax2.set_title('Tip amount by Passenger count')
15 ax2.set_xlabel('Passenger count')
16 ax2.set_ylabel('Tip Amount ($)')
17 plt.subtitle('')
18 plt.show()
19
20 # Look at the relationship between fare and tip amounts
21 ax = sampled_taxi_pd_df.plot(kind='scatter', x='fareAmount', y = 'tipAmount', c='blue', alpha = 0.10, s=2.5*(sampled_taxi_pd_df['passengerCount']))
22 ax.set_xlabel('Fare Amount ($)')
23 ax.set_ylabel('Tip Amount ($)')
24 plt.axis([-2, 80, -2, 20])
25 plt.subtitle('')
26 plt.show()
27

```

Exploratory data analysis with graphs – histogram, boxplot etc

# Library Management - Python

## Overview

Customers can add new python libraries at Spark pool level

## Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

Update versions of existing libraries on your cluster

Ability to specify different requirements file for different pools within the same workspace

## Constraints

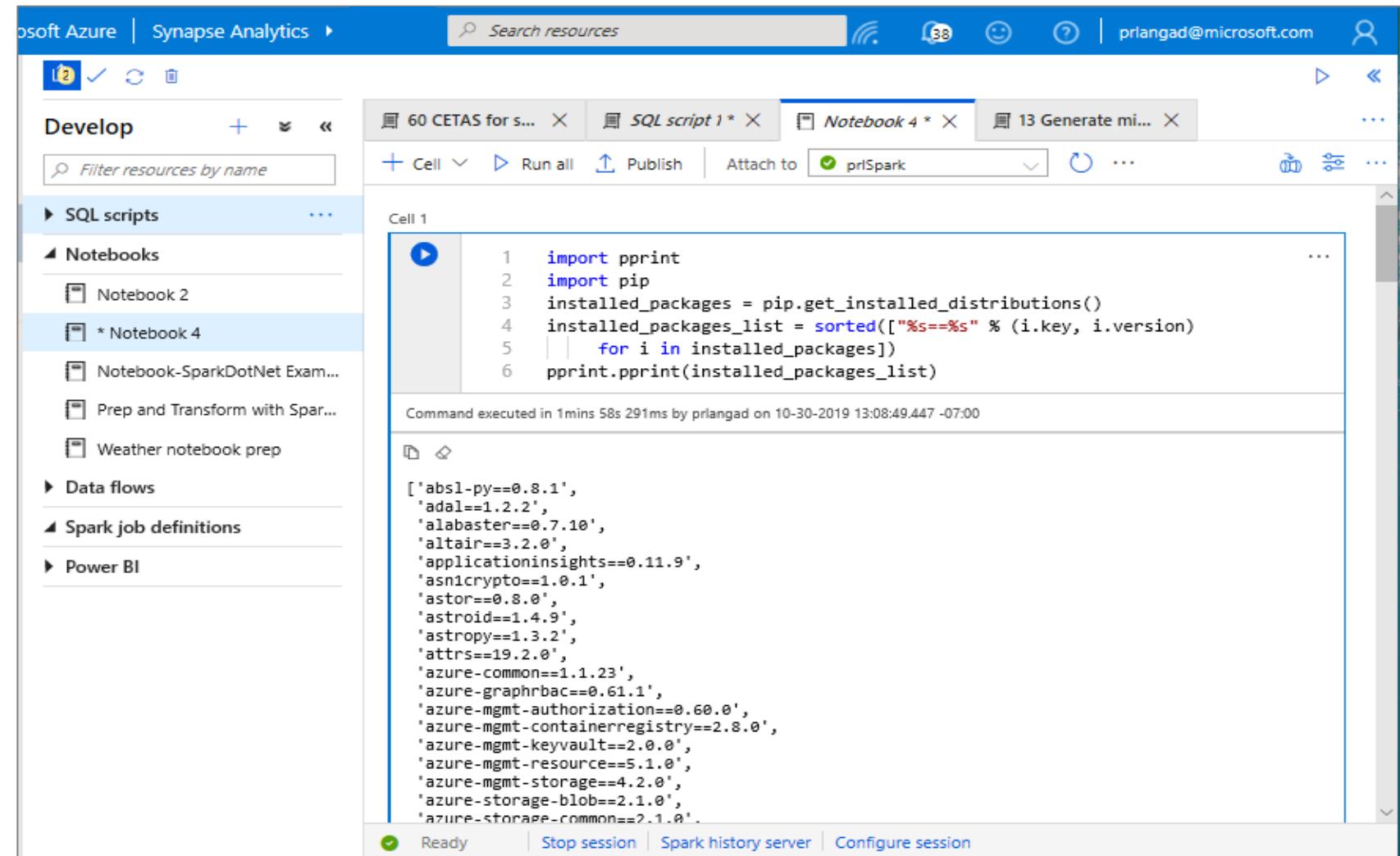
The library version must exist on PyPI repository

Version downgrade of an existing library not allowed

The screenshot shows the Azure Synapse Analytics Library Management interface. On the left, there's a sidebar with options like Analytics pools, SQL pools, Apache Spark pools (which is selected), External connections, Linked services, Orchestration, Triggers, Integration runtimes, Security, Access control, and Managed private endpoints. The main area is titled 'Apache Spark pools' and shows three items: 'priLangadSpark2', 'priLang-syntaxcheck', and 'priSpark'. To the right is a 'Properties' panel for 'priSpark'. It includes fields for Name ('priSpark'), URL ('/subscriptions/56f8824d-32b0-4825-9825-02fa6a801546/resourceGroups/priLangadrg/pro...'), Creation date ('10/30/2019, 12:50:37 PM'), and sections for Configuration and Workspace. Below these is a 'Packages' section with a red box around it, containing a 'Upload environment config file' button and a 'Refresh' button. A message says 'No user-provided packages currently uploaded. You can upload "environment config file".' At the bottom right of the main area is a 'Close' button.

# Library Management - Python

Get list of installed libraries with version information



The screenshot shows the Azure Synapse Analytics interface with the 'Develop' workspace selected. In the center, a notebook titled 'Notebook 4' is open. A code cell in this notebook contains the following Python script:

```
1 import pprint
2 import pip
3 installed_packages = pip.get_installed_distributions()
4 installed_packages_list = sorted(['%s==%s' % (i.key, i.version)
5         for i in installed_packages])
6 pprint.pprint(installed_packages_list)
```

When run, the cell outputs a long list of installed Python packages and their versions, such as:

```
['absl-py==0.8.1',
 'adal==1.2.2',
 'alabaster==0.7.10',
 'altair==3.2.0',
 'applicationinsights==0.11.9',
 'asn1crypto==1.0.1',
 'astor==0.8.0',
 'astroid==1.4.9',
 'astropy==1.3.2',
 'attrs==19.2.0',
 'azure-common==1.1.23',
 'azure-graphrbac==0.61.1',
 'azure-mgmt-authorization==0.60.0',
 'azure-mgmt-containerregistry==2.8.0',
 'azure-mgmt-keyvault==2.0.0',
 'azure-mgmt-resource==5.1.0',
 'azure-mgmt-storage==4.2.0',
 'azure-storage-blob==2.1.0',
 'azure-storage-common==2.1.0']
```

The notebook also shows other tabs like 'SQL script 1' and 'Notebook 4'. The status bar at the bottom indicates the session is 'Ready'.

# Spark ML Algorithms

## Spark ML Algorithms

Classification and Regression	<ul style="list-style-type: none"><li>• Linear Models (SVMs, logistic regression, linear regression)</li><li>• Naïve Bayes</li><li>• Decision Trees</li><li>• Ensembles of trees (Random Forest, Gradient-Boosted Trees)</li><li>• Isotonic regression</li></ul>
Clustering	<ul style="list-style-type: none"><li>• k-means and streaming k-means</li><li>• Gaussian mixture</li><li>• Power iteration clustering (PIC)</li><li>• Latent Dirichlet allocation (LDA)</li></ul>
Collaborative Filtering	<ul style="list-style-type: none"><li>• Alternating least squares (ALS)</li></ul>
Dimensionality Reduction	<ul style="list-style-type: none"><li>• SVD</li><li>• PCA</li></ul>
Frequent Pattern Mining	<ul style="list-style-type: none"><li>• FP-growth</li><li>• Association rules</li></ul>
Basic Statistics	<ul style="list-style-type: none"><li>• Summary statistics</li><li>• Correlations</li><li>• Stratified sampling</li><li>• Hypothesis testing</li><li>• Random data generation</li></ul>

# Microsoft Machine Learning for Apache Spark

v1.0-rc

Microsoft's Open Source  
Contributions to Apache Spark



Distributed  
Machine Learning



Fast Model  
Deployment



Microservice  
Orchestration



Multilingual Binding  
Generation

[www.aka.ms/spark](http://www.aka.ms/spark)

 [Azure/mmlspark](https://github.com/Azure/mmlspark)

## Synapse Notebook: Connect to AML workspace

The screenshot shows the Azure Synapse Notebook interface. The left sidebar lists resources under 'Develop': SQL scripts, Notebooks (selected), Data flows, Spark job definitions, and Power BI. The main area displays a notebook cell titled 'Check the Azure ML Core SDK Version to Validate Your Installation'. The code in Cell 3 is:

```
[5] 1 import azureml.core  
2 print("SDK Version:", azureml.core.VERSION)
```

The output shows the command was executed in 1s 258ms by balapv on 11-12-2019 14:41:52.805 -08:00, and the output is 'SDK Version: 1.0.69'.

Below this, a section titled 'Connect to Azure Workspace' is shown. Cell 5 contains the following code:

```
[6] 1 ## Import the Workspace class and check the Azure ML SDK version.  
2 from azureml.core import Workspace  
3  
4 ws = Workspace(subscription_id = "6560575d-fa06-4e7d-95fb-f962e74efd7a",  
5 | | | | | resource_group = "balapv-synapse-rg", workspace_name = "AML-WS-synapse")  
6  
7 print(ws.name, ws.location, ws.resource_group, sep='\t')
```

The output shows the command was executed in 3s 909ms by balapv on 11-12-2019 14:41:55.491 -08:00, and the output is 'AML-WS-synapse westus2 balapv-synapse-rg'.

Cell 6 contains the following code:

```
[7] 1 # import modules  
2 import azureml.core  
3 import pandas as pd  
4 from azureml.core.authentication import ServicePrincipalAuthentication  
5 from azureml.core.workspace import Workspace  
6 from azureml.core.experiment import Experiment
```

At the bottom of the notebook, there are buttons for 'Running', 'Stop session', 'Spark history server', and 'Configure session'.

A red arrow points from the text 'Simple code to connect workspace' to the code in Cell 5.

# Synapse Notebook: Configure AML job to run on Synapse

The screenshot shows the Azure Synapse Analytics notebook interface. On the left, the sidebar lists resources under 'Develop': SQL scripts, Notebooks (with 'automl\_synapse\_local\_regr...' selected), Data flows, Spark job definitions, and Power BI. The main area is titled 'Train' and contains documentation for the AutoMLConfig object. It includes a note about enabling ONNX compatible models and a table of properties and their descriptions. Below this is a code cell labeled 'Cell 13' containing Python code for configuring an AutoML experiment. A red arrow points from the text 'Configuration parameters' to the code cell.

**Configuration parameters** →

```

1 automl_config = AutoMLConfig(task = 'regression',
2                               debug_log = 'automl_errors.log',
3                               primary_metric = 'normalized_root_mean_squared_error',
4                               iteration_timeout_minutes = 10,
5                               iterations = 20,
6                               preprocess = True,
7                               n_cross_validations = 2,
8                               max_concurrent_iterations = 2, #spark compute size
9                               verbosity = logging.INFO,
10                              spark_context=sc, #spark related
11                              enable_onnx_compatible_models=True, # This will generate ONNX compatible models.
12                              cache_store=True,
13                              X = X_train,
14                              y = y_train)

```

Call the `submit` method on the experiment object and pass the run configuration. Execution of local runs is synchronous. Depending on the data and the number of iterations this can run for a while. In this example, we specify `show_output = True` to print currently running iterations to the console.

## Synapse Notebook: Run AML job

The screenshot shows the Azure Synapse Notebook interface. The top navigation bar includes 'Microsoft Azure | Synapse Analytics > synapsews4aml', a search bar, 'Show notifications' with a bell icon, and a user profile for 'balapv@microsoft.com'. The left sidebar under 'Develop' lists resources: SQL scripts, Notebooks (including 'automl\_synapse\_local\_regression'), Data flows, Spark job definitions, and Power BI.

The main workspace is titled 'automl\_synapse\_local\_regression' and shows a 'Run AutoML job' section. Cell 15 contains the command:

```
1 local_run = experiment.submit(automl_config, show_output = True)
```

Output from Cell 15:

```
Command executed in 12mins 34s 972ms by balapv on 11-12-2019 15:17:53.089 -08:00
```

Running an experiment on spark cluster: automl-local-regression-Synapse.  
Parent Run ID: AutoML\_ad8600ab-a1ab-4b6b-b233-059d969e0a0e

```
*****  
ITERATION: The iteration being evaluated.  
PIPELINE: A summary description of the pipeline being evaluated.  
DURATION: Time taken for the current iteration.  
METRIC: The result of computing score on the fitted pipeline.  
BEST: The best observed score thus far.  
*****
```

ITERATION	PIPELINE	DURATION	METRIC	BEST
1	StandardScalerWrapper ElasticNet	0:00:38	0.0021	0.0021
2	StandardScalerWrapper ElasticNet	0:00:32	0.0054	0.0021
0	StandardScalerWrapper ElasticNet	0:01:20	0.0004	0.0004
4	StandardScalerWrapper RandomForest	0:00:33	0.0179	0.0004
3	StandardScalerWrapper ElasticNet	0:00:36	0.0036	0.0004
5	StandardScalerWrapper LightGBM	0:00:28	0.0109	0.0004
6	MaxAbsScaler DecisionTree	0:00:34	0.0168	0.0004
7	MaxAbsScaler RandomForest	0:00:41	0.0104	0.0004
8	MaxAbsScaler DecisionTree	0:01:05	0.0077	0.0004
9	MaxAbsScaler DecisionTree	0:00:48	0.0086	0.0004
10	StandardScalerWrapper DecisionTree	0:00:39	0.0058	0.0004
11	MaxAbsScaler DecisionTree	0:00:45	0.0096	0.0004
13	MaxAbsScaler ExtremeRandomTrees	0:00:47	0.0147	0.0004
12	MaxAbsScaler ExtremeRandomTrees	0:01:54	0.0096	0.0004
14	StandardScalerWrapper ElasticNet	0:00:39	0.0027	0.0004
15	StandardScalerWrapper ElasticNet	0:00:54	0.0010	0.0004
16	StandardScalerWrapper ElasticNet	0:00:48	0.0023	0.0004
17	MaxAbsScaler ElasticNet	0:00:31	0.0239	0.0004
18	StandardScalerWrapper ElasticNet	0:00:53	0.0014	0.0004
19	VotingEnsemble	0:01:59	0.0004	0.0004

A red arrow points from the text 'ML job execution result' to the output table.

At the bottom, there's a link 'Get Azure Portal URL for Monitoring Runs' and session controls: 'Running' (radio button), 'Stop session', 'Spark history server', and 'Configure session'.



# Azure Synapse Analytics Security

# Securing with firewalls

## Overview

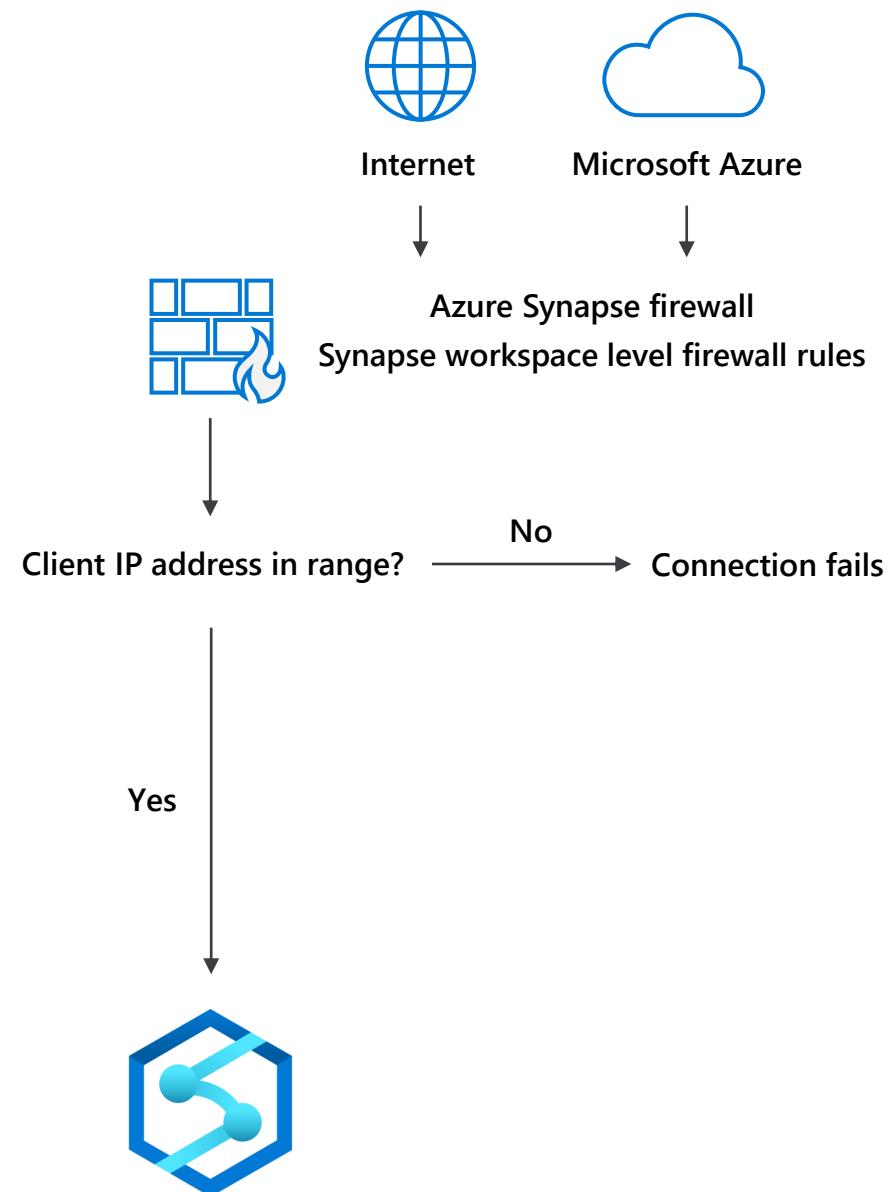
By default, all access to your Azure Synapse Analytics is blocked by the firewall.

Firewall also manages virtual network rules that are based on virtual network service endpoints.

## Rules

Allow specific or range of whitelisted IP addresses.

Allow Azure applications to connect.



# IP Firewall Rules

## Overview

IP firewall rules grant or deny access to user's Synapse workspace based on the originating IP address of each request.

IP firewall rules configured at the workspace level apply to all public endpoints of the workspace (SQL provisioned, SQL serverless, and Development).

## Key Points

- Customers can also use SQL Server Management Studio (SSMS) to connect to the SQL resources (SQL provisioned and SQL serverless) in their workspace.
- Customers must ensure that the firewall on the network and local computer allow outgoing communication on TCP ports 80, 443 and 1443 for Synapse Studio.
- Customers must also allow outgoing communication on UDP port 53 for Synapse Studio.
- To connect using tools such as SSMS and Power BI, user must allow outgoing communication on TCP port 1433.

# Managed VNet

## Overview

Creating a workspace with a Managed workspace VNet associated with it ensures that user's workspace is network isolated from other workspaces. The VNet associated with your workspace is managed by Azure Synapse. This VNet is called a Managed workspace VNet.

## Benefits

- With a Managed workspace customers can offload the burden of managing the VNet to Azure Synapse.
- Customers don't have to configure inbound NSG rules on their own VNets to allow Azure Synapse management traffic to enter their VNet.
- Customers don't need to create a subnet for your Spark clusters based on peak load.
- Managed workspace VNet along with Managed private endpoints protects against data exfiltration.

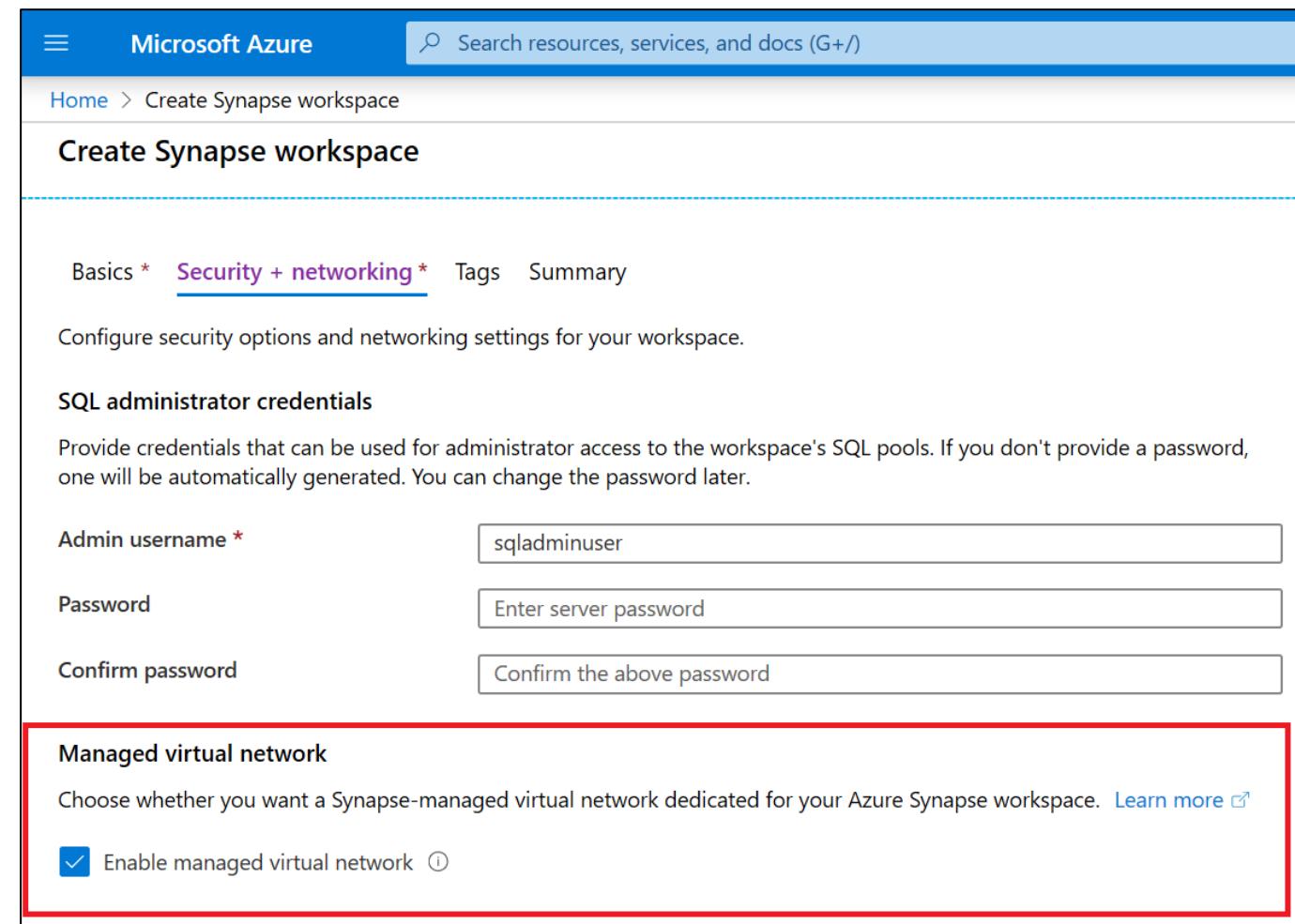
# Managed VNet

## Overview

During Azure Synapse workspace creation, user can choose to associate it to a managed VNet. User cannot change this workspace configuration after the workspace is created.

User cannot reconfigure a workspace that does not have a Managed workspace VNet associated with it and associate a VNet to it.

Private links are supported only in Synapse workspaces that have a managed VNet associated to it.



The screenshot shows the 'Create Synapse workspace' wizard in the Microsoft Azure portal. The 'Security + networking' tab is active. The 'SQL administrator credentials' section is visible, showing fields for Admin username (sqladminuser), Password, and Confirm password. Below this, the 'Managed virtual network' section is highlighted with a red border. It contains a description: 'Choose whether you want a Synapse-managed virtual network dedicated for your Azure Synapse workspace.' A link to 'Learn more' is provided. An unchecked checkbox labeled 'Enable managed virtual network' is present.

Microsoft Azure

Search resources, services, and docs (G+/)

Home > Create Synapse workspace

Create Synapse workspace

Basics \* Security + networking \* Tags Summary

Configure security options and networking settings for your workspace.

SQL administrator credentials

Provide credentials that can be used for administrator access to the workspace's SQL pools. If you don't provide a password, one will be automatically generated. You can change the password later.

Admin username \*

sqladminuser

Password

Enter server password

Confirm password

Confirm the above password

Managed virtual network

Choose whether you want a Synapse-managed virtual network dedicated for your Azure Synapse workspace. [Learn more](#)

Enable managed virtual network

# Private Endpoints

## Overview

Managed private endpoints are private endpoints created in the Managed workspace VNet establishing a private link to Azure resources. Managed private endpoints are only supported in Azure Synapse workspaces with a Managed workspace VNet.

## Benefits

- Private link enables customers to access Azure services and Azure hosted customer/partner services from their Azure VNet securely.
- With use of private link, traffic between customer's VNet and workspace traverses entirely over the Microsoft backbone network.
- Private link protects against data exfiltration risks.
- Private endpoint uses a private IP address from customer's VNet to effectively bring the service into their VNet.
- Private endpoints are mapped to a specific resource in Azure and not the entire service.

# Private Endpoints for SQL Provisioned and SQL Serverless

## Overview

SQL provisioned and SQL serverless use multi-tenant infrastructure that is not deployed into the Managed workspace VNet.

Azure Synapse creates two managed private endpoints to SQL provisioned and SQL serverless in that workspace. Customers do not get charged for these two Managed private endpoints.

The screenshot shows the Microsoft Azure Synapse Analytics portal interface. The left sidebar has a 'Manage' button highlighted with a red box. Below it is a 'Managed Virtual Networks' button also highlighted with a red box. The main content area is titled 'Managed Virtual Networks' and describes how they allow establishing connections from the workspace to data sources using Azure Private Link. It features a 'Managed private endpoint' section with a 'New' button and a table listing two items:

NAME ↑	PROVISIONING STATE ↑↓	APPROVAL STATE ↑↓	VNET NAME ↑↓	POSSIBLE LOCATIONS ↑↓	LINKED RESOURCE ID ↑↓
synapse-ws-sqlOnDemand...	✓ Succeeded	✓ Approved	default	1	/subscriptions/0...
synapse-ws-sql--202003...	✓ Succeeded	✓ Approved	default	0	/subscriptions/0...

# Azure Active Directory authentication

## Overview

Manage user identities in one location.

Enable access to Azure Synapse Analytics and other Microsoft services with Azure Active Directory user identities and groups.

## Benefits

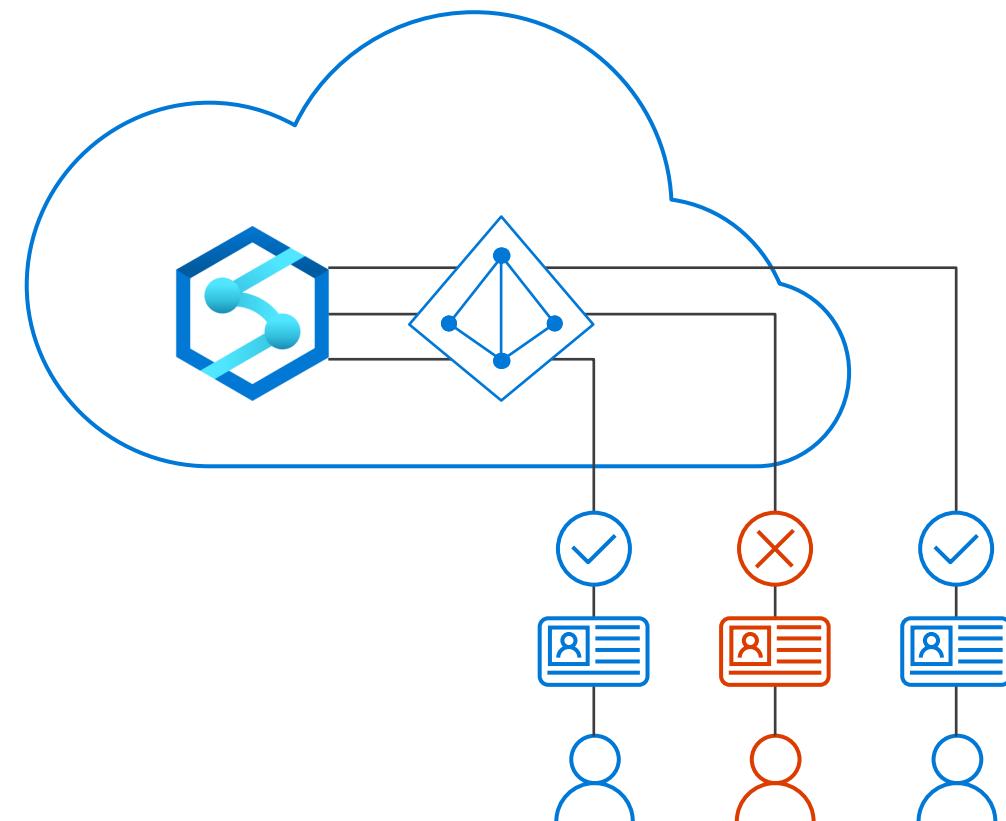
Enables management of database permissions by using external Azure Active Directory groups

Allows password rotation in a single place

Alternative to SQL Server authentication

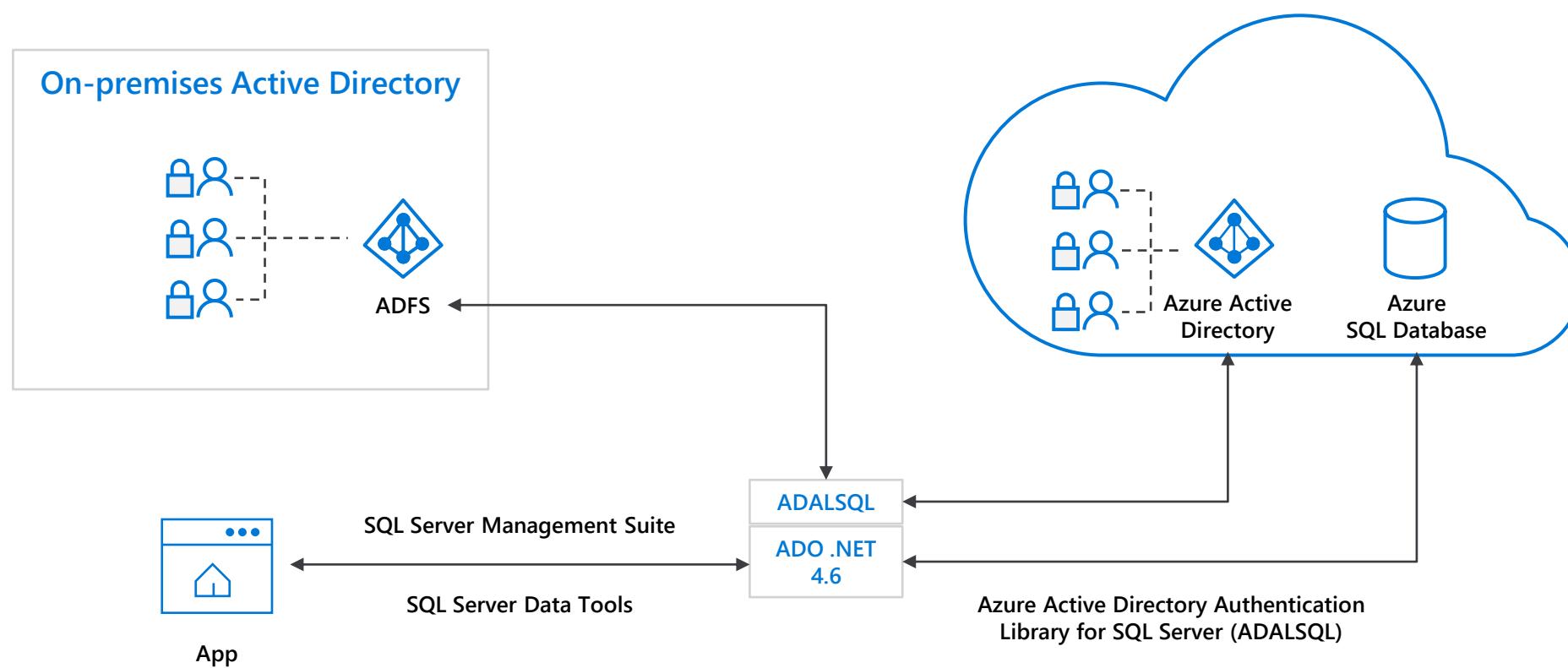
Eliminates the need to store passwords

## Azure Synapse Analytics



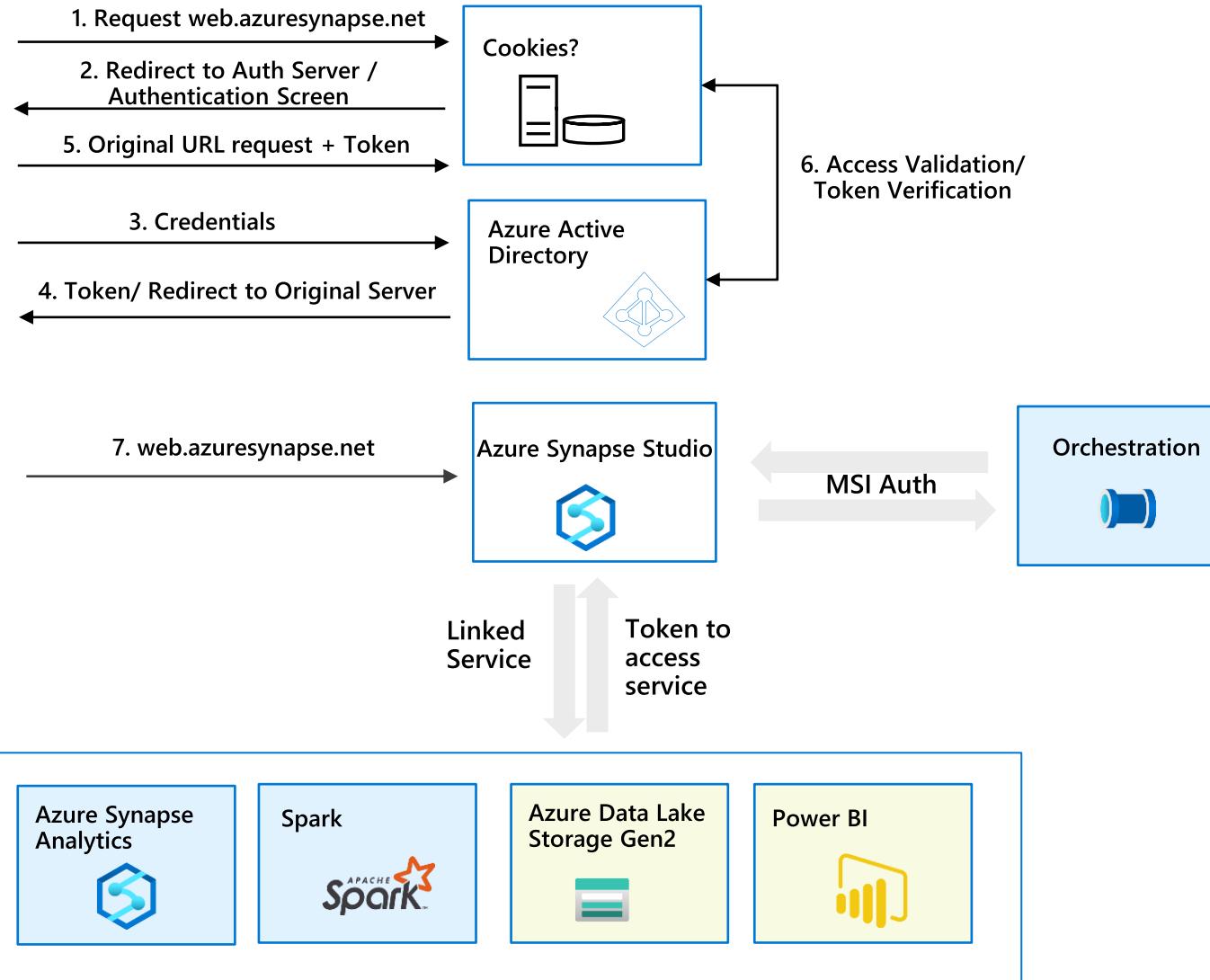
# Azure Active Directory trust architecture

## Azure Active Directory and Azure Synapse Analytics



# Single Sign-On

Synapse Foundation Components  
 Synapse Linked Services



**Implicit authentication** - User provides login credentials once to access Azure Synapse Workspace

**AAD authentication** - Azure Synapse Studio will request token to access each linked services as user. A separate token is acquired for each of the below services:

1. ADLS Gen2
2. Azure Synapse Analytics
3. Power BI
4. Spark – Spark Livy API
5. `management.azure.com` – resource provisioning
6. Develop artifacts – `dev.workspace.net`
7. Graph endpoints

**MSI authentication** - Orchestration uses workspace MSI auth for automation

# Access Control

## Overview

It provides access control management to workspace resources and artifacts for admins

## Benefits

Share workspace with the team

Increases productivity

Manage permissions on SQL pools and Spark pools

The screenshot shows the Microsoft Azure Access Control interface for a workspace named "internalsandbox". The left sidebar lists various workspace resources: Analytics pools, SQL pools, Apache Spark pools, External connections, Linked services, Orchestration, Triggers, Integration runtimes, Security, and Access control. The "Access control" item is highlighted with a red box. The main area displays a table of current role assignments:

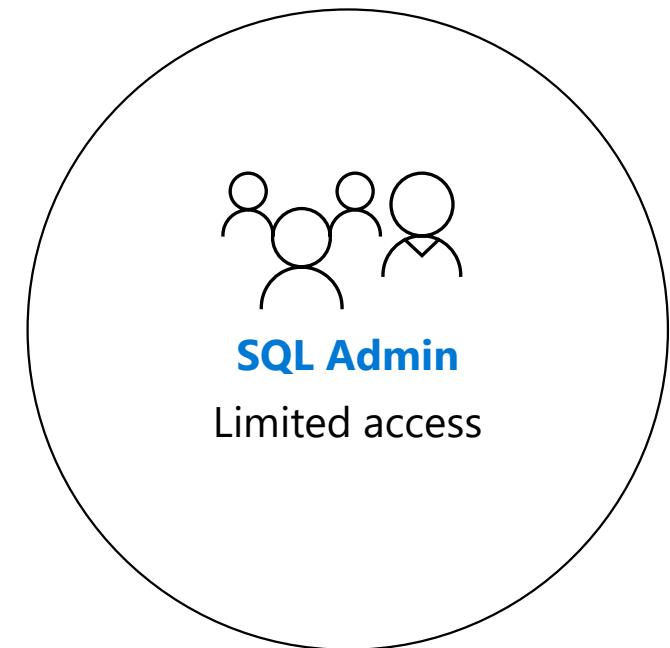
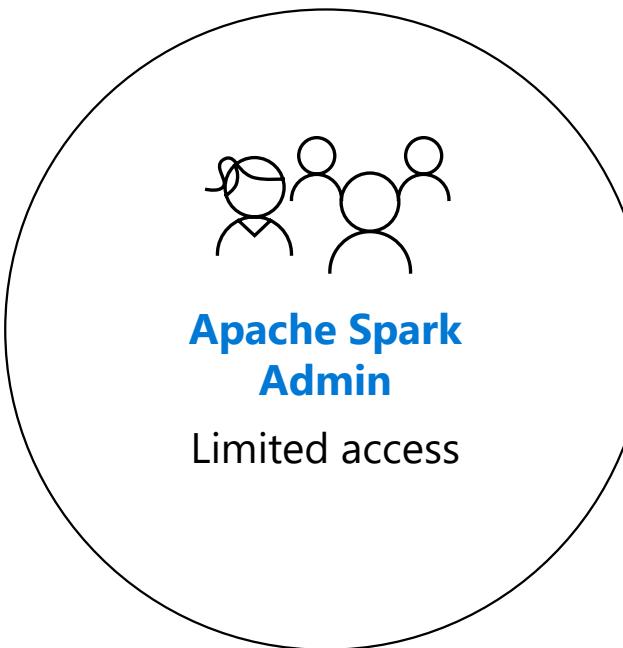
NAME	TYPE	ROLE
Priyanka Langade Priyanka.Langade@microsoft.com	Individual	Workspace admin
Syna		

A red arrow points from the "Add" button in the top navigation bar to the "Add role assignment" dialog box. This dialog box contains fields for selecting a role (a dropdown menu labeled "Select a role") and checkboxes for "Select all", "Workspace admin", "SQL admin", and "Apache Spark admin". At the bottom are "Apply" and "Cancel" buttons.

# Synapse Roles

## Overview

Azure Synapse workspace is managed by three different roles which are Workspace Admin, Apache Spark Admin, SQL Admin



# Synapse Roles - Permissions

## Common Permissions for all Admins

- Open Synapse Studio
- View Home hub
- View Data Hub
- Data Hub / See Databases
- Data Hub / See objects in databases
- Use the Develop hub
- Use the Orchestrate hub
- Use the Manage Hub
- Manage Hub / Linked services
- Manage Hub / Integration runtimes
- Manage Hub / Private Endpoints



Workspace Admin



Apache Spark  
Admin



SQL Admin

# Synapse Roles - Permissions



## Workspace Admin - Specific Permissions

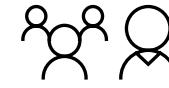
- Develop Hub / Author Dataflows
- Orchestrate hub / Use Pipelines
- Manage Hub / Triggers
- Manage Hub / Access Control (assign users to Synapse workspace roles)
- Additional permissions
  - Apache Spark Admin Permissions
  - SQL Admin Permissions

# Synapse Roles - Permissions



## Apache Spark Admin – Specific Permissions

- Data Hub / Access data in Spark databases
- Develop Hub / Author Spark Job Definitions
- Develop Hub / Author Notebooks
- Manage Hub / Spark pools



## SQL Admin – Specific Permissions

- Data Hub / Access data in SQL provisioned databases
- Data Hub / Access data in SQL serverless databases
- Data Hub / Access data in Spark databases
- Develop Hub / Author SQL Scripts
- Manage Hub / SQL pools

**Industry-leading security  
and compliance**

# Enterprise-grade security



Defense-in-Depth

# Industry-leading compliance



ISO 27001



SOC 1 Type 2



SOC 2 Type 2



PCI DSS Level 1



Cloud Controls Matrix



ISO 27018



Content Delivery and Security Association



Shared Assessments



FedRAMP JAB P-ATO



HIPAA / HITECH



FIPS 140-2



21 CFR Part 11



FERPA



DISA Level 2



CJIS



IRS 1075 / ITAR-ready



European Union Model Clauses



EU Safe Harbor



United Kingdom G-Cloud



China Multi Layer Protection Scheme



China GB 18030



China CCCPPF



Singapore MTCS Level 3



Australian Signals Directorate



New Zealand GCIO



Japan Financial Services



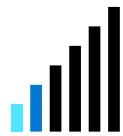
ENISA IAF

# Threat Protection - Business requirements



**How do we enumerate and track potential SQL vulnerabilities?**

To mitigate any security misconfigurations before they become a serious issue.



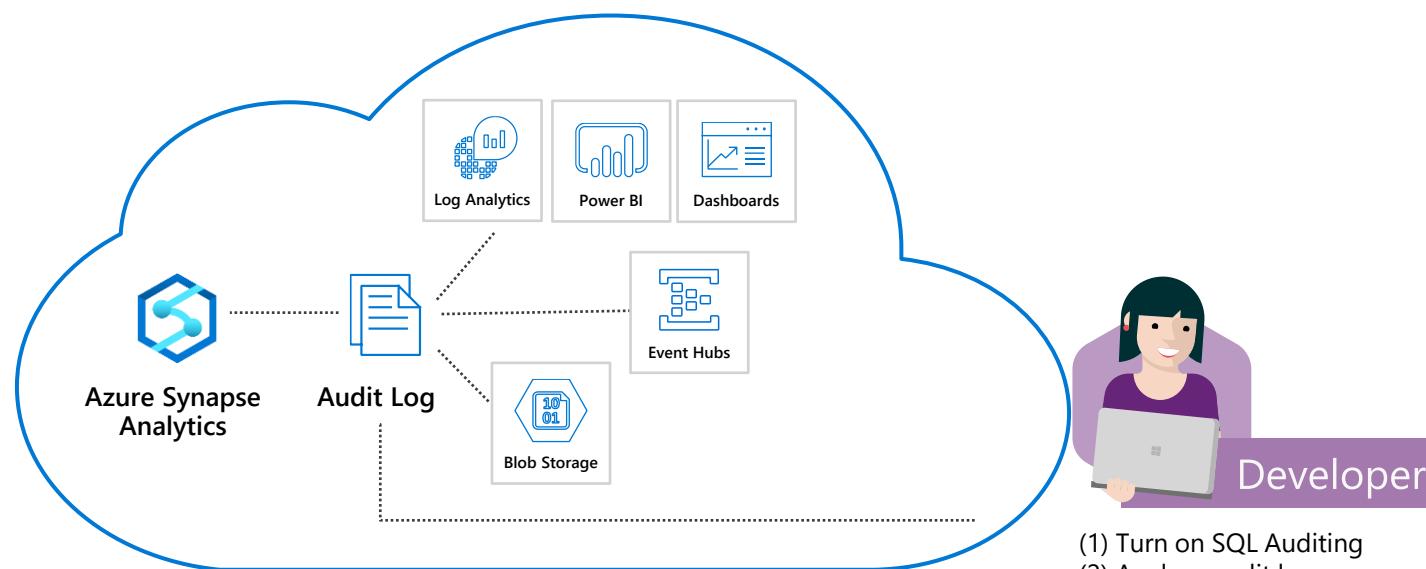
**How do we discover and alert on suspicious database activity?**

To detect and resolve any data exfiltration or SQL injection attacks.



# SQL auditing in Azure Log Analytics and Event Hubs

Gain insight into database audit log

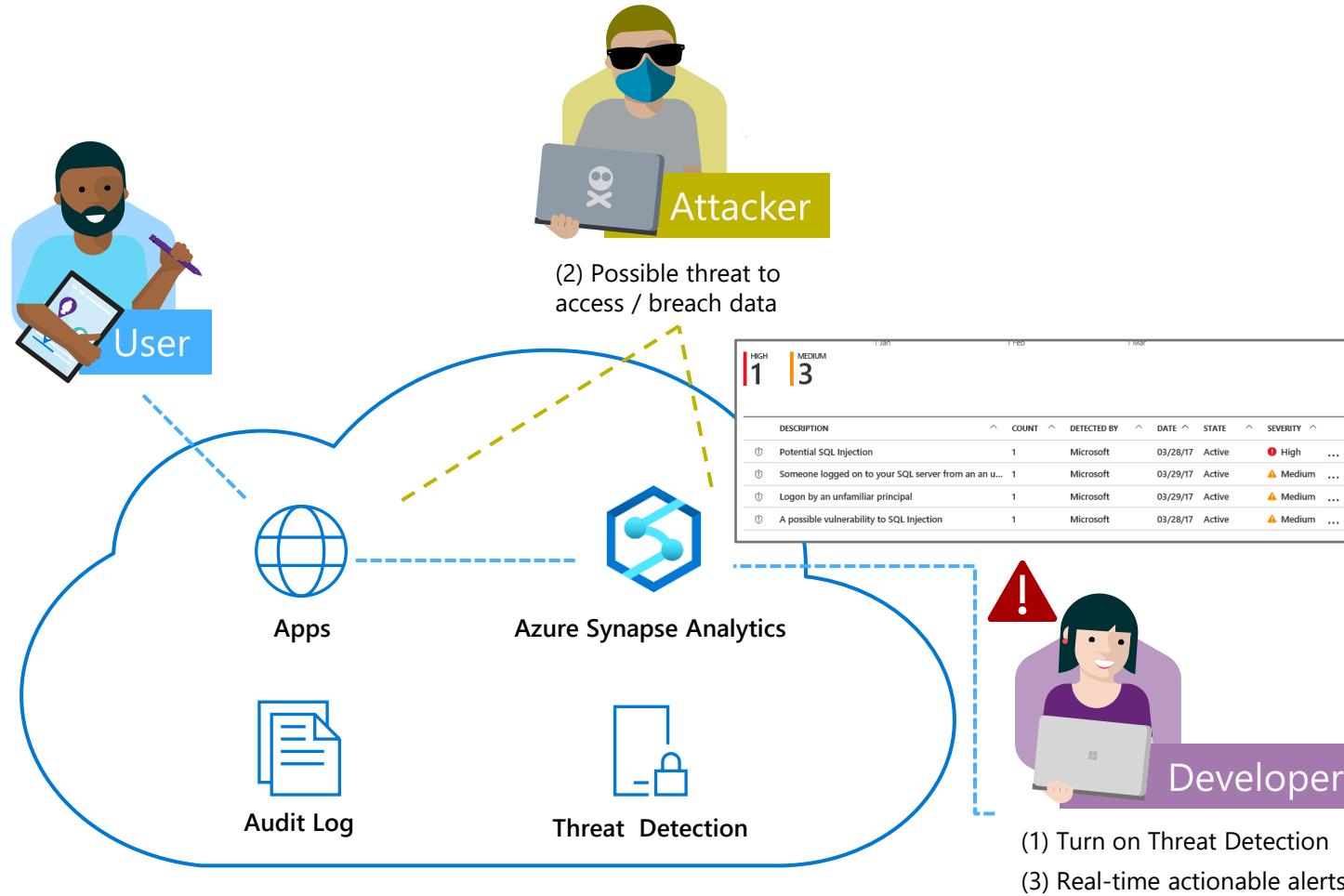


TimeGenerated	server_principal_name_s	statement_s	affected_rows_d	SeverityLevel
8/15/2018 12:00:22.521 AM	admin1	exec sp_executesql N'SELECT t1.name AS [Name], SCHEMA_NAME(t1...	0	
8/15/2018 12:00:22.521 AM	admin1	exec sp_executesql N'SELECT ISNULL(HAS_PERMS_BY_NAME(QUOTEN...	0	
8/15/2018 12:00:22.521 AM	admin1	DECLARE @edition sysname; SET @edition = cast(SERVERPROPERTY(N...	1	
8/15/2018 12:00:22.521 AM	admin1	exec sp_executesql N'SELECT CAST(t1.is_enabled AS bit) AS [isEnabled]...	0	
8/15/2018 12:00:22.521 AM	admin1	IF OBJECT_ID ('[sys].[database_query_store_options]') IS NOT NULL BE...	2	

- ✓ Configurable via audit policy
- ✓ SQL audit logs can reside in
  - Azure Storage account
  - Azure Log Analytics
  - Azure Event Hubs
- ✓ Rich set of tools for
  - Investigating security alerts
  - Tracking access to sensitive data

# SQL threat detection

Detect and investigate anomalous database activity



- ✓ Detects potential SQL injection attacks
- ✓ Detects unusual access & data exfiltration activities
- ✓ Actionable alerts to investigate & remediate
- ✓ View alerts for your entire Azure tenant using Azure Security Center

# SQL Data Discovery & Classification

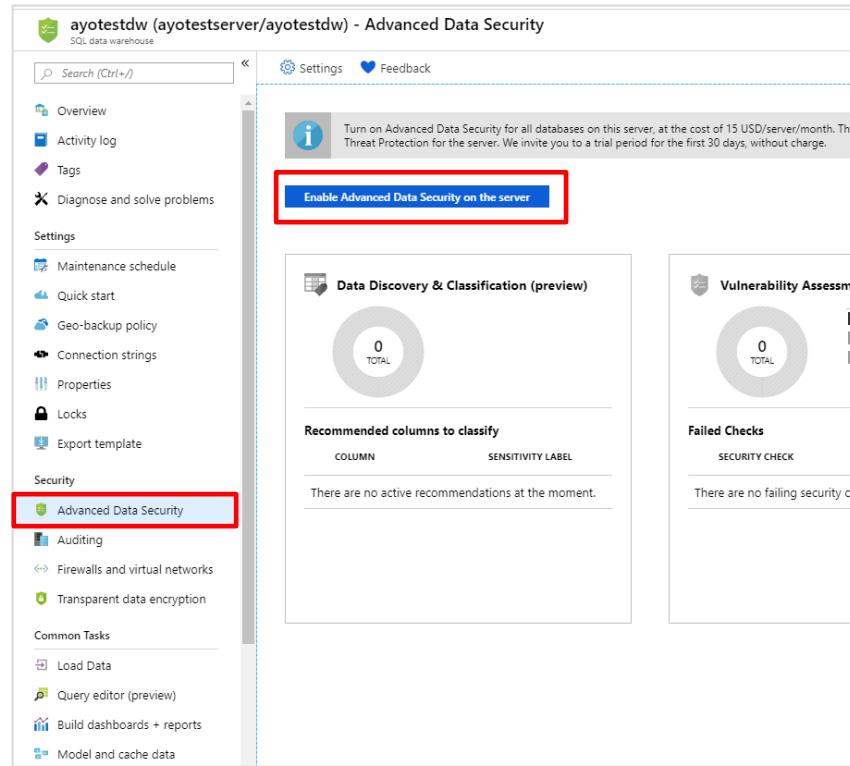
## Discover, classify, protect and track access to sensitive data

The screenshot shows the Azure portal interface for 'Data discovery & classification (preview)'. The main dashboard displays various metrics: Classified columns (10 / 109), Tables containing sensitive data (4 / 12), Unique information types (4), and Information type distribution (CONTACT INFO: NAME, CREDENTIALS, FINANCIAL). Below this, a donut chart shows the distribution of columns across different sensitivity labels. A secondary window titled 'Settings - Information protection' shows a list of sensitivity labels: Public, General, Confidential, Confidential - GDPR, Highly confidential, and Highly confidential - GDPR, each with a detailed description.

- ✓ Automatic **discovery** of columns with sensitive data
- ✓ Add **persistent sensitive data labels**
- ✓ Audit and detect access to the sensitive data
- ✓ Manage labels for your entire Azure tenant using Azure Security Center

# SQL Data Discovery & Classification - setup

## Step 1: Enable Advanced Data Security on the logical SQL Server



ayotestdw (ayotestserver/ayotestdw) - Advanced Data Security

Turn on Advanced Data Security for all databases on this server, at the cost of 15 USD/server/month. This includes Threat Protection for the server. We invite you to a trial period for the first 30 days, without charge.

**Enable Advanced Data Security on the server**

**Recommended columns to classify**

COLUMN	SENSITIVITY LABEL
There are no active recommendations at the moment.	

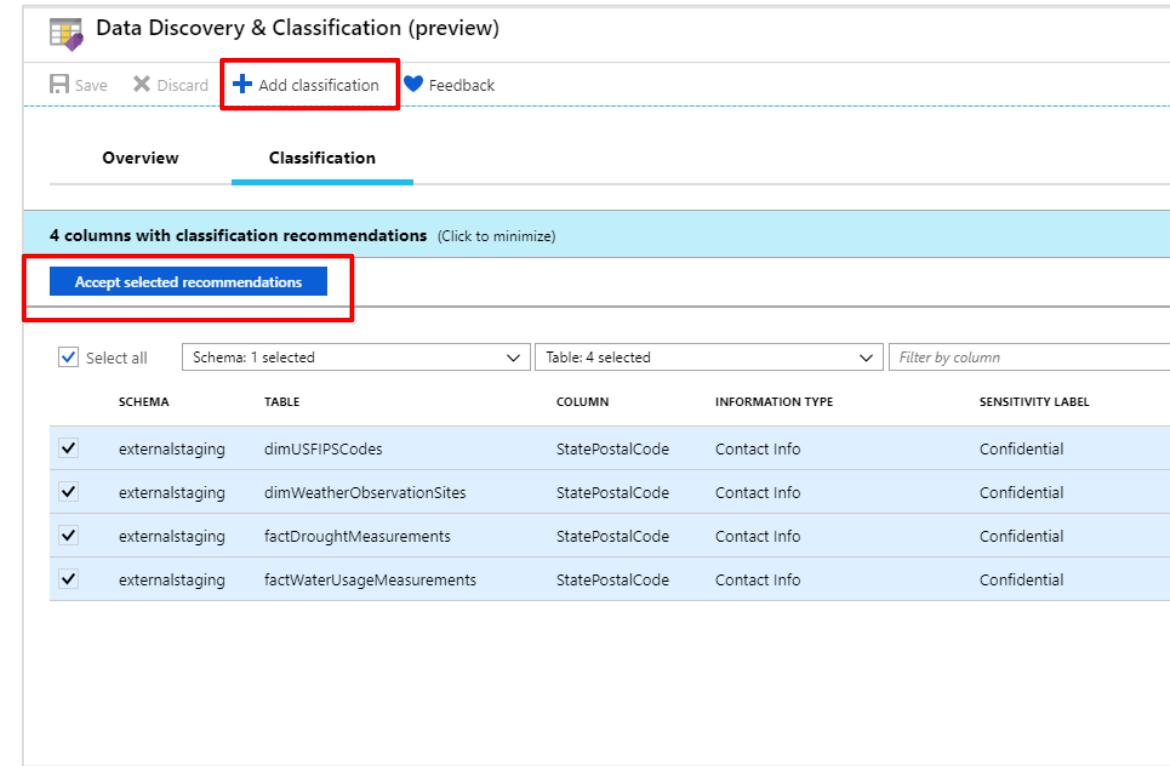
**Vulnerability Assessment**

TOTAL	HIG	MED	LOW
0			

**Failed Checks**

SECURITY CHECK	
There are no failing security checks.	

## Step 2: Use recommendations and/or manual classification to classify all the sensitive columns in your tables



Data Discovery & Classification (preview)

**Add classification**

**Classification**

4 columns with classification recommendations (Click to minimize)

**Accept selected recommendations**

Select all Schema: 1 selected Table: 4 selected Filter by column

SCHEMA	TABLE	COLUMN	INFORMATION TYPE	SENSITIVITY LABEL
externalstaging	dimUSFIPSCodes	StatePostalCode	Contact Info	Confidential
externalstaging	dimWeatherObservationSites	StatePostalCode	Contact Info	Confidential
externalstaging	factDroughtMeasurements	StatePostalCode	Contact Info	Confidential
externalstaging	factWaterUsageMeasurements	StatePostalCode	Contact Info	Confidential

# SQL Data Discovery & Classification – audit sensitive data access

**Step 1:** Configure auditing for your target Data warehouse. This can be configured for just a single data warehouse or all databases on a server.

The screenshot shows the 'ayotestdw (ayotestserver/ayotestdw) - Auditing' page in the Azure portal. The 'Auditing' section is highlighted with a red box. The 'Auditing' toggle switch is set to 'ON'. Below it, there's a note: 'If Blob Auditing is enabled on the server, it will always apply to the database, regardless of the database settings.' There are sections for 'View server settings' and 'Audit log destination (choose at least one): Storage'.

**Step 2:** Navigate to audit logs in storage account and download 'xel' log files to local machine.

The screenshot shows the 'sqldbauditlogs' container in the Azure Storage account. The blob '01\_34\_30\_090\_0.xel' is selected and highlighted with a red box. The blob details show the location as 'sqldbauditlogs / ayotestserver / ayotestdw / SqldbAuditing\_Audit / 2019-04-02'.

**Step 3:** Open logs using extended events viewer in SSMS. Configure viewer to include 'data\_sensitivity\_information' column

The screenshot shows the SSMS extended events viewer displaying audit logs from '02\_26\_37\_736\_0.xel'. The 'data\_sensitivity\_information' column is highlighted with a red box. A detailed view of the first event is shown below, with the 'data\_sensitivity\_information' field also highlighted with a red box.

name	timestamp	affected_rows	application_name	client_ip	data_sensitivity_information	database_name
audit_event	2019-02-26 18:38:35.7892923	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.7661039	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.7052286	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.6873633	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.6680990	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.6490621	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.6292824	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.6110493	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5911164	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5739871	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5557121	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5393015	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5213010	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.5032121	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.4956126	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.4675595	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.4487751	0	Net SqlClient Data Provider	10.0.0.4		master
audit_event	2019-02-26 18:38:35.4290439	0	Net SqlClient Data Provider	10.0.0.4		master

Event: audit\_event (2019-02-26 18:38:35.6680990)

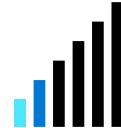
Field	Value
action_id	1176681924
additional_information	<login_information><error_code>18456</error_code><error_stat...
affected_rows	0
application_name	Net SqlClient Data Provider
audit_schema_version	1
class_type	16964
client_ip	10.0.0.4
connection_id	F1AD6457-9F40-409B-B43C-E638AAFA47902
data_sensitivity_information	
database_name	master
database_principal_id	-1
database_principal_name	
duration_milliseconds	0
event_time	2019-02-26 18:38:35.6743004
host_name	usgsvm089
is_column_permission	False
object_id	5
object_name	master

# Network Security - Business requirements



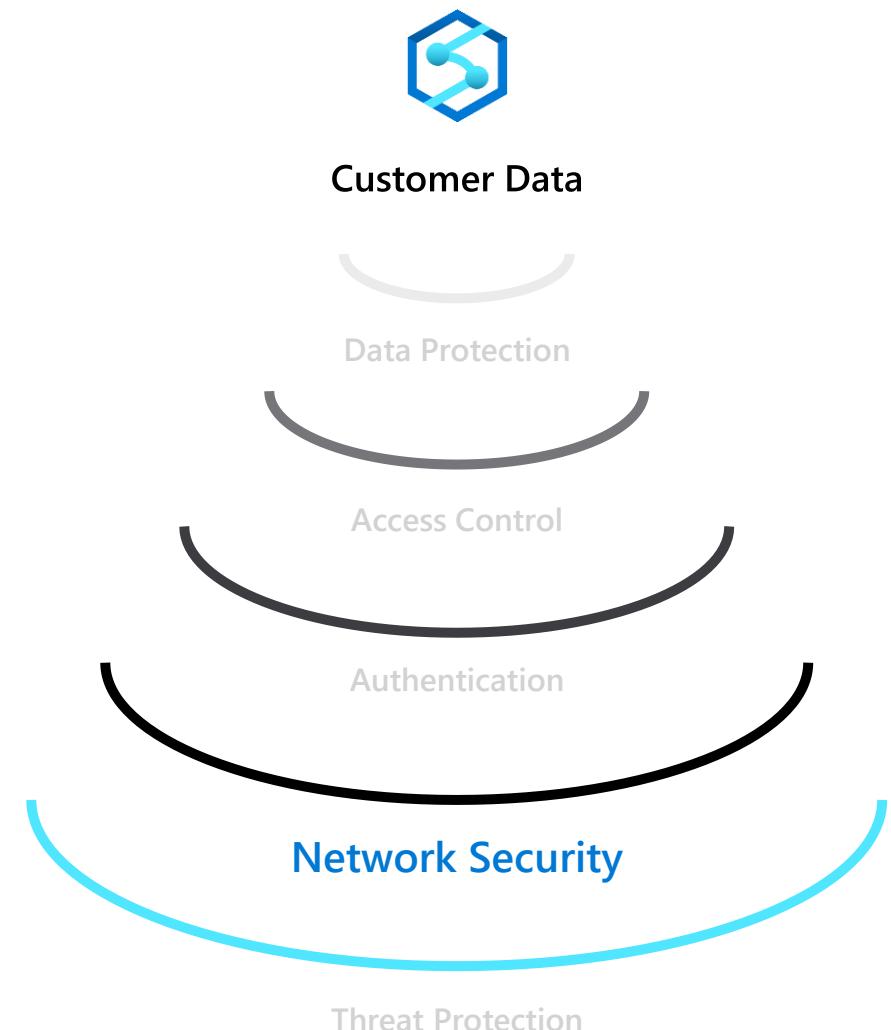
## How do we implement network isolation?

Data at different levels of security needs to be accessed from different locations.

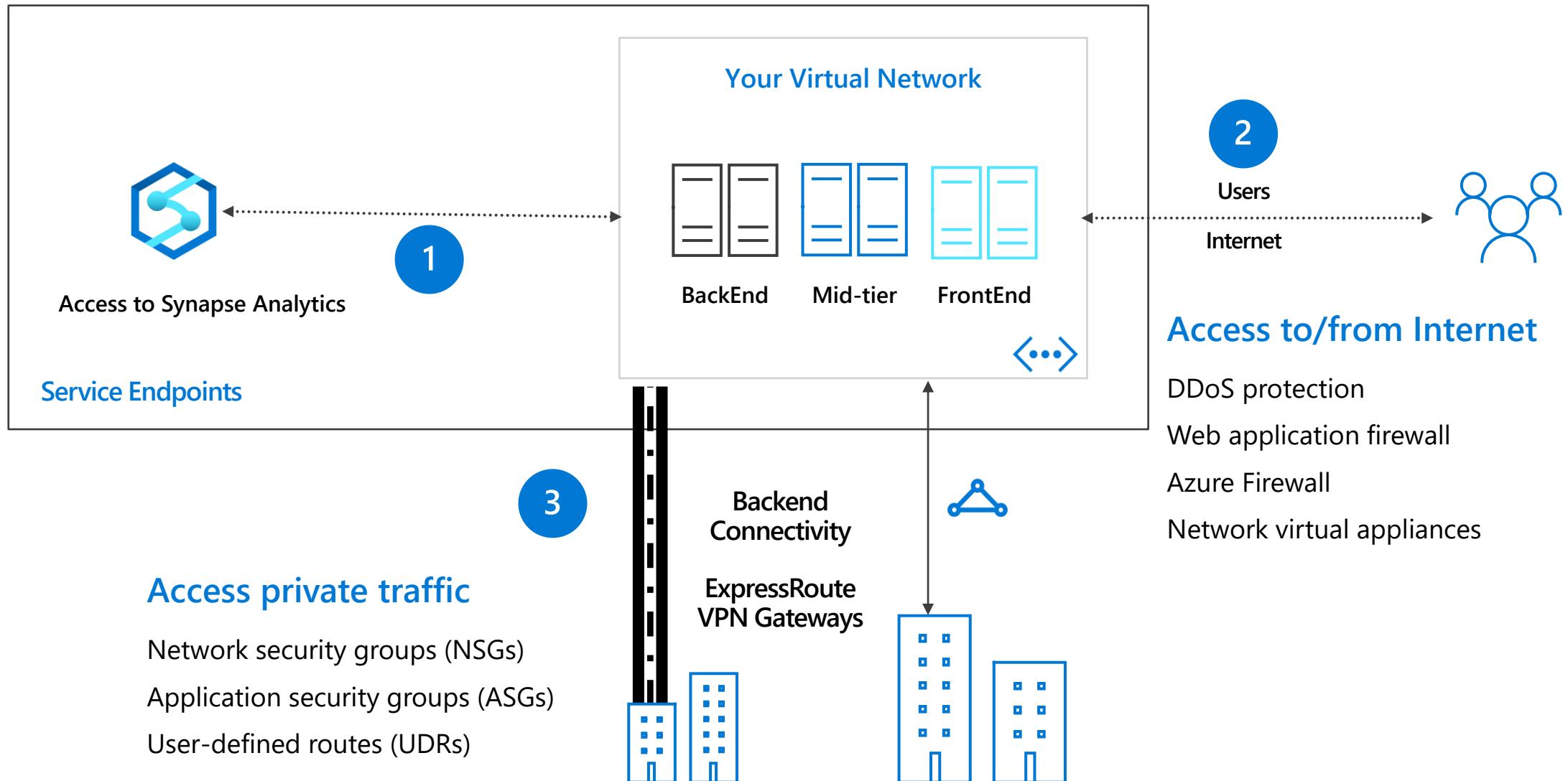


## How do we achieve separation?

Disallowing access to entities outside the company's network security boundary.



# Azure networking: application-access patterns

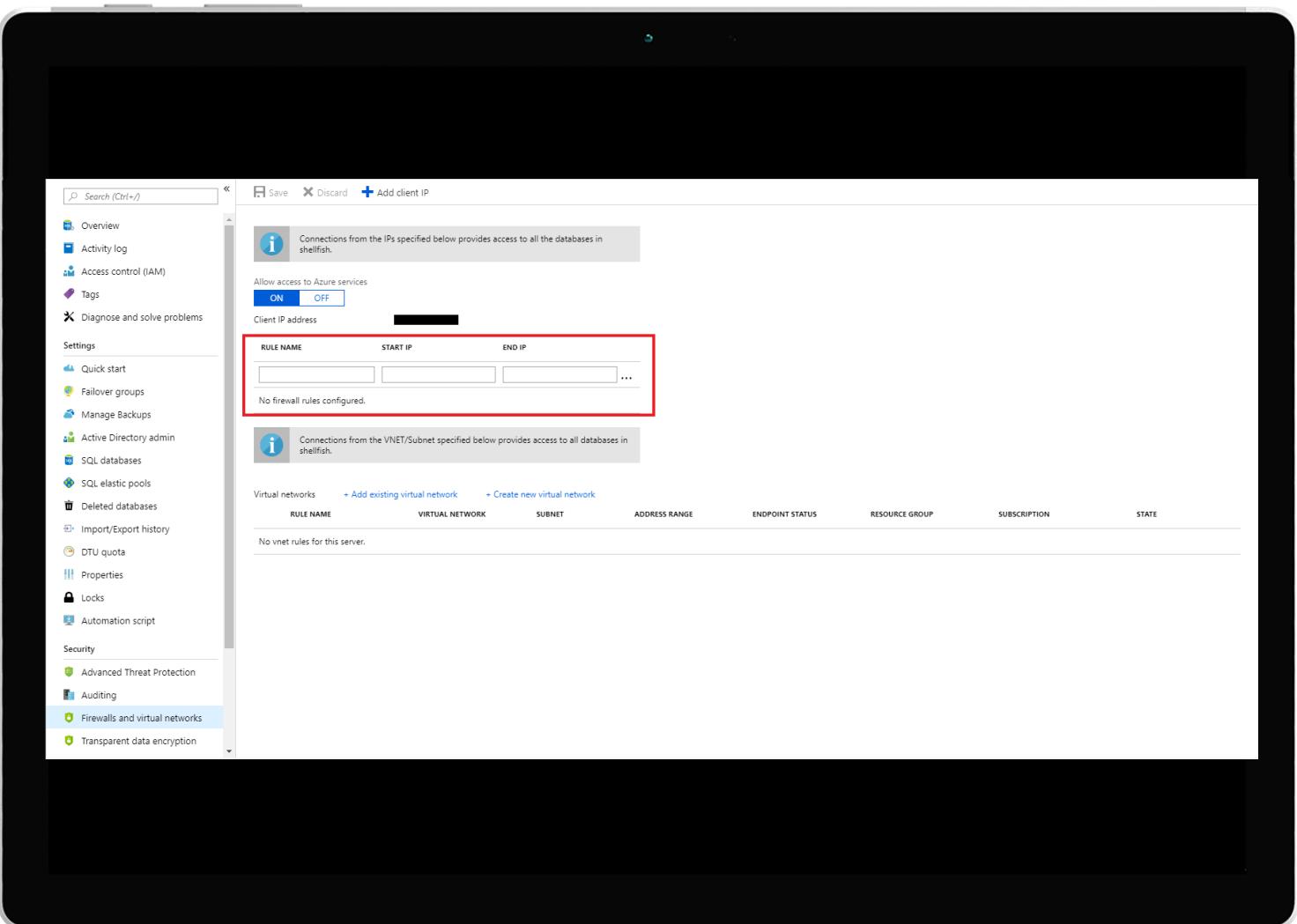


# Firewall configuration on the portal

By default, Azure blocks all external connections to port 1433

Configure with the following steps:

Azure Synapse Analytics Resource:  
Server name > Firewalls and virtual networks



# Firewall configuration using REST API

Managing firewall rules through REST API must be authenticated.

For information, see [Authenticating Service Management Requests](#).

Server-level rules can be created, updated, or deleted using [REST API](#).

To create or update a server-level firewall rule, execute the [PUT](#) method.

To remove an existing server-level firewall rule, execute the [DELETE](#) method.

To list firewall rules, execute the [GET](#).

PUT

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01REQUEST BODY
{
  "properties": {
    "startIpAddress": "0.0.0.3",
    "endIpAddress": "0.0.0.3"
  }
}
```

DELETE

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01
```

GET

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Sql/servers/{serverName}/firewallRules/{firewallRuleName}?api-version=2014-04-01
```

# Firewall configuration using PowerShell/T-SQL

## Windows PowerShell Azure cmdlets

```
New-AzureRmSqlServerFirewallRule
```

```
Get-AzureRmSqlServerFirewallRule
```

```
Set-AzureRmSqlServerFirewallRule
```

## Transact SQL

```
sp_set_firewall_rule
```

```
sp_delete_firewall_rule
```

```
# PS Allow external IP access to SQL DW  
PS C:\> New-AzureRmSqlServerFirewallRule  
    -ResourceGroupName "myResourceGroup" `  
    -ServerName $servername `  
    -FirewallRuleName "AllowSome" `  
    -StartIpAddress "0.0.0.0" `  
    -EndIpAddress "0.0.0.0"  
  
-- T-SQL Allow external IP access to SQL DW  
EXECUTE sp_set_firewall_rule  
    @name = N'ContosoFirewallRule',  
    @start_ip_address = '192.168.1.1',  
    @end_ip_address = '192.168.1.10'
```

# VNET configuration on Azure portal

## Configure with the following steps:

Azure Synapse Analytics Resource:

Server name > Firewalls and virtual networks

REST API and PowerShell alternatives available

### Note:

By default, VMs on your subnets cannot communicate with your SQL Data Warehouse.

There must first be a virtual network service endpoint for the rule to reference.

The screenshot shows the 'Firewall / Virtual Networks' settings for a SQL server named 'gm-sql-db-server-svr1'. The 'Allow access to Azure services' toggle is set to 'OFF'. A client IP address '73.118.201.137' is listed. Below, two rules are defined: 'gm-ip-rule-ir1' (IP range 172.27.26.0 - 172.27.26.255) and 'gm-ip-rule-ir2' (IP range 73.118.201.0 - 73.118.201.255). A note states that connections from the specified VNET/Subnet provide access to all databases. At the bottom, there are buttons for 'Virtual networks' (with '+ Add existing' highlighted by a red box), '+ Create new', and columns for 'RULE NAME', 'RESOURCE GROUP/VNET NAME', and 'SUBNET'.

RULE NAME	START IP	END IP	Actions
gm-ip-rule-ir1	172.27.26.0	172.27.26.255	...
gm-ip-rule-ir2	73.118.201.0	73.118.201.255	...

# Authentication - Business requirements

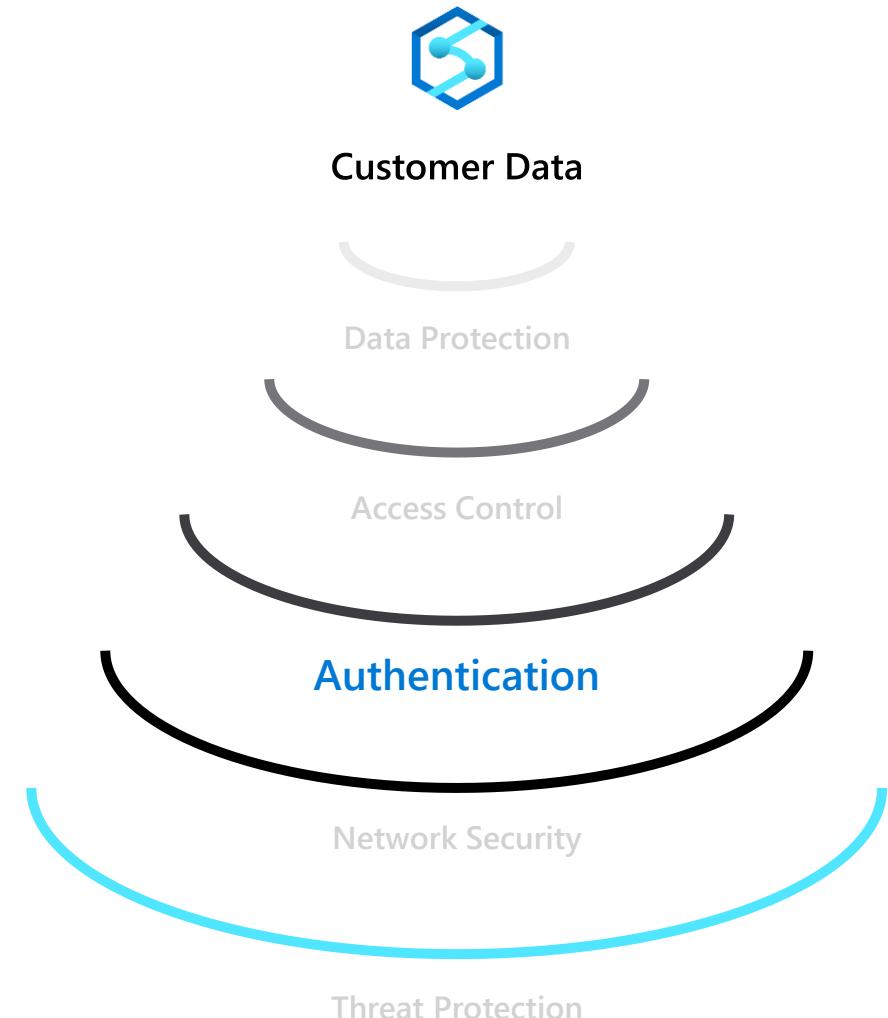


**How do I configure Azure Active Directory with Azure Synapse Analytics?**

I want additional control in the form of multi-factor authentication



**How do I allow non-Microsoft accounts to be able to authenticate?**



# Azure Active Directory authentication

## Overview

Manage user identities in one location.

Enable access to Azure Synapse Analytics and other Microsoft services with Azure Active Directory user identities and groups.

## Azure Synapse Analytics

## Benefits

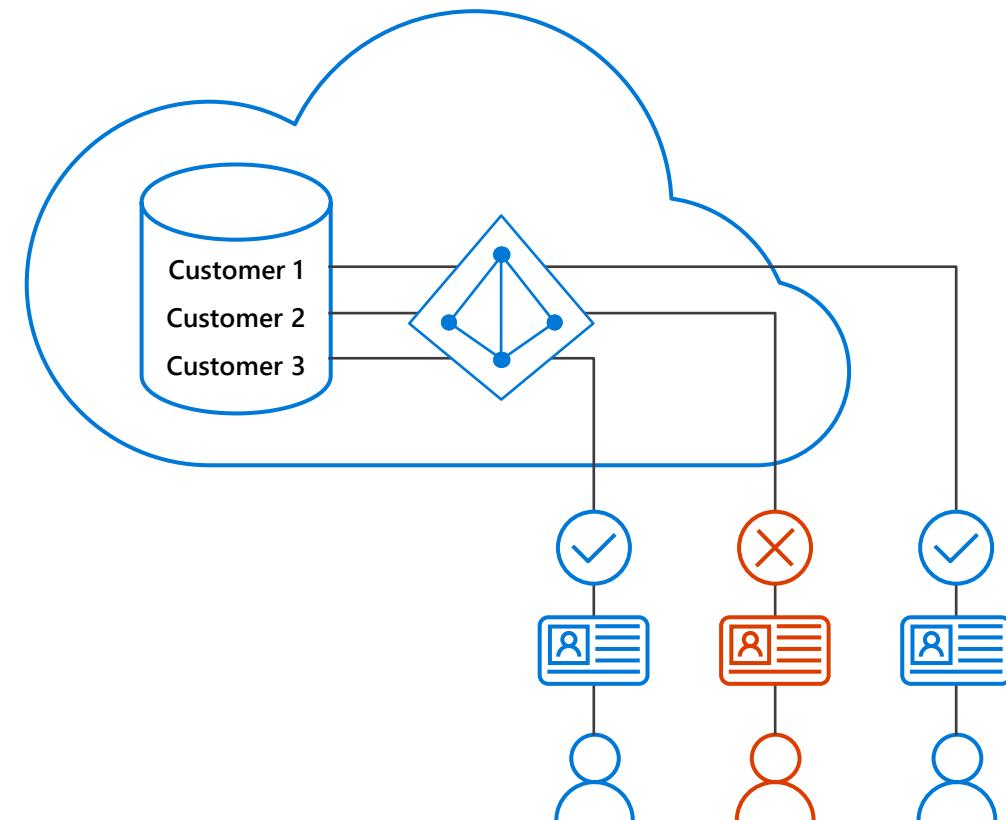
Alternative to SQL Server authentication

Limits proliferation of user identities across databases

Allows password rotation in a single place

Enables management of database permissions by using external Azure Active Directory groups

Eliminates the need to store passwords



# SQL authentication

## Overview

This authentication method uses a username and password.

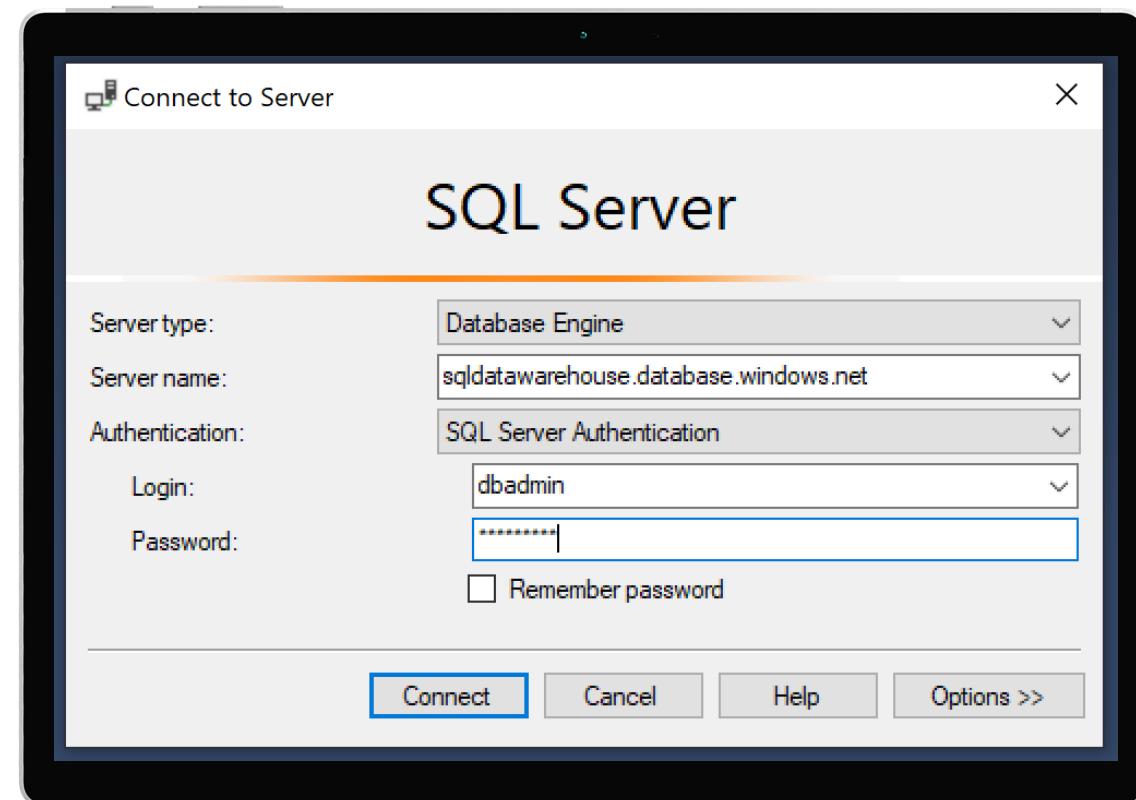
When you created the logical server for your data warehouse, you specified a "server admin" login with a username and password.

Using these credentials, you can authenticate to any database on that server as the database owner.

Furthermore, you can create user logins and roles with familiar SQL Syntax.

```
-- Connect to master database and create a login  
CREATE LOGIN ApplicationLogin WITH PASSWORD = 'Str0ng_password';  
CREATE USER ApplicationUser FOR LOGIN ApplicationLogin;
```

```
-- Connect to SQL DW database and create a database user  
CREATE USER DatabaseUser FOR LOGIN ApplicationLogin;
```



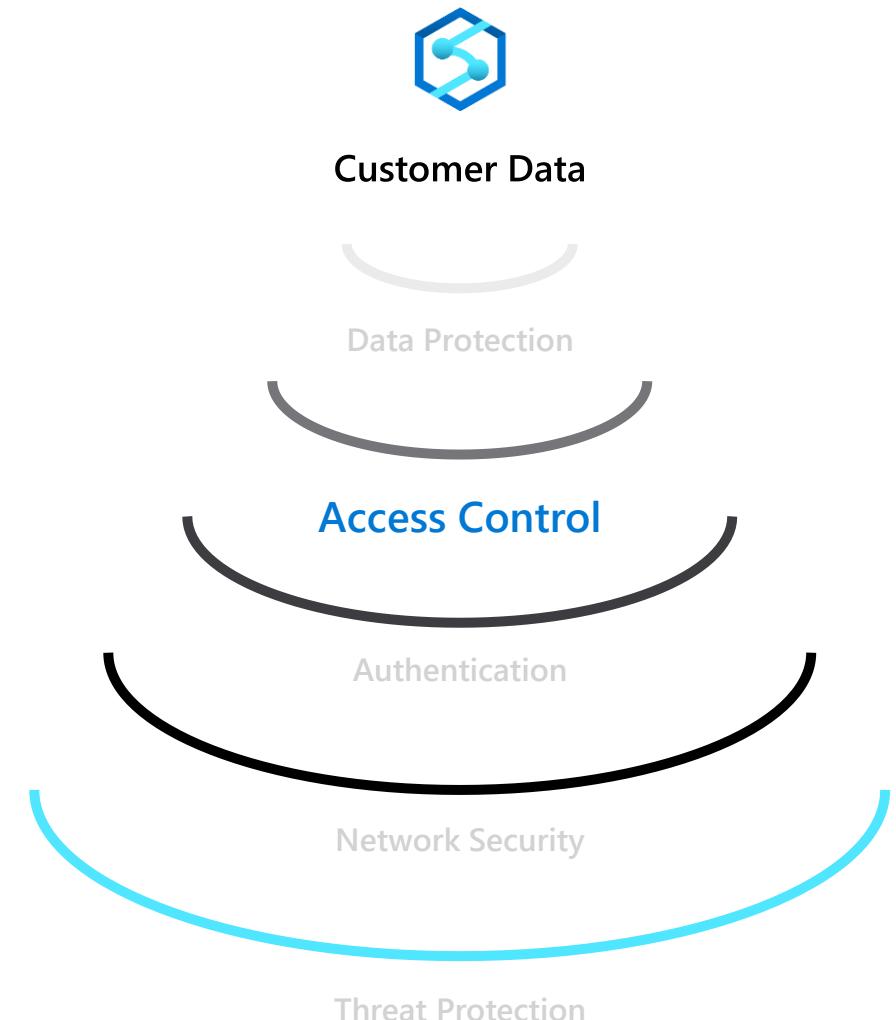
# Access Control - Business requirements



How do I restrict access to sensitive data to specific database users?

How do I ensure users only have access to relevant data?

For example, in a hospital only medical staff should be allowed to see patient data that is relevant to them—and not every patient's data.



# Object-level security (tables, views, and more)

## Overview

GRANT controls permissions on designated tables, views, stored procedures, and functions.

Prevent unauthorized queries against certain tables.

Simplifies design and implementation of security at the database level as opposed to application level.

```
-- Grant SELECT permission to user RosaQdM on table Person.Address in the AdventureWorks2012 database
GRANT SELECT ON OBJECT::Person.Address TO RosaQdM;
GO

-- Grant REFERENCES permission on column BusinessEntityID in view HumanResources.vEmployee to user Wanida
GRANT REFERENCES(BusinessEntityID) ON OBJECT::HumanResources.vEmployee TO Wanida WITH GRANT OPTION;
GO

-- Grant EXECUTE permission on stored procedure HumanResources.uspUpdateEmployeeHireInfo to an application role called Recruiting11
USE AdventureWorks2012;
GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo TO RECRUITING 11;
GO
```

# Row-level security (RLS)

## Overview

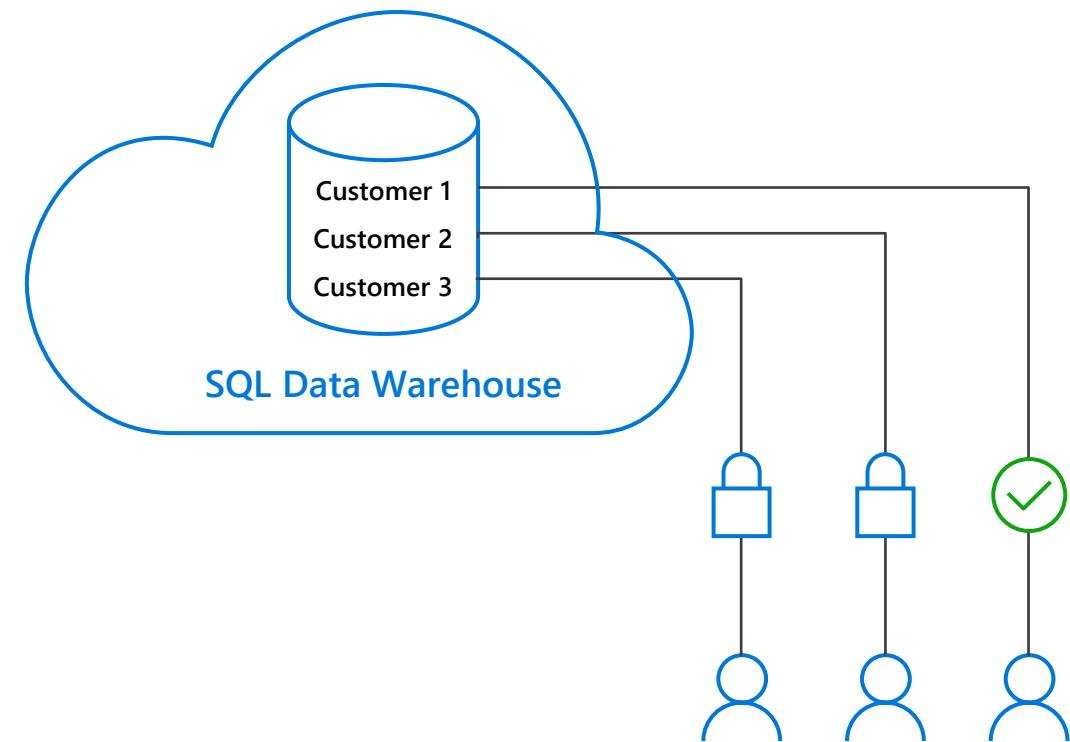
Fine grained access control of specific rows in a database table.

Help prevent unauthorized access when multiple users share the same tables.

Eliminates need to implement connection filtering in multi-tenant applications.

Administer via SQL Server Management Studio or SQL Server Data Tools.

Easily locate enforcement logic inside the database and schema bound to the table.



# Row-level security

## Creating policies

Filter predicates silently filter the rows available to read operations (SELECT, UPDATE, and DELETE).

The following examples demonstrate the use of the CREATE SECURITY POLICY syntax

```
-- The following syntax creates a security policy with a filter predicate for the Customer table
CREATE SECURITY POLICY [FederatedSecurityPolicy]
ADD FILTER PREDICATE [rls].[fn_securitypredicate]([CustomerId])
ON [dbo].[Customer];

-- Create a new schema and predicate function, which will use the application user ID stored in CONTEXT_INFO to filter rows.
CREATE FUNCTION rls.fn_securitypredicate (@AppUserId int)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN (
SELECT 1 AS fn_securitypredicate_result
WHERE
DATABASE_PRINCIPAL_ID() = DATABASE_PRINCIPAL_ID('dbo') -- application context
AND CONTEXT_INFO() = CONVERT(VARBINARY(128), @AppUserId));
GO
```

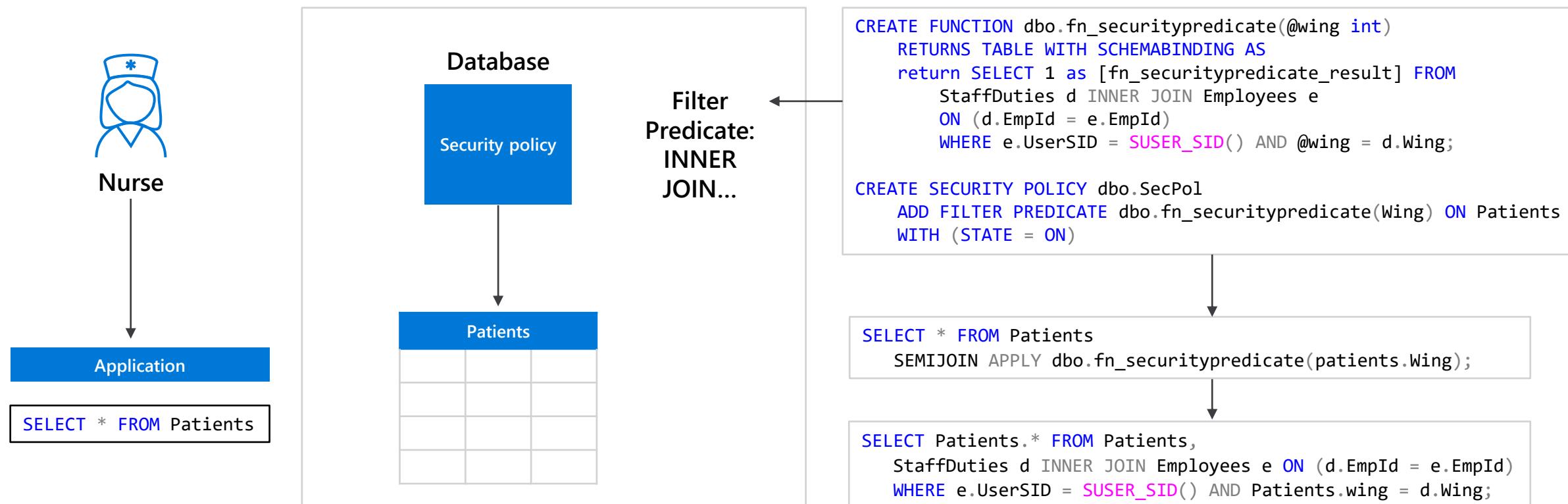
# Row-level security

## Three steps:

1. Policy manager creates filter predicate and security policy in T-SQL, binding the predicate to the patients table.
2. App user (e.g., nurse) selects from Patients table.
3. Security policy transparently rewrites query to apply filter predicate.



Policy manager



# Column-level security

## Overview

Control access of specific columns in a database table based on customer's group membership or execution context.

Simplifies the design and implementation of security by putting restriction logic in database tier as opposed to application tier.

Administer via GRANT T-SQL statement.

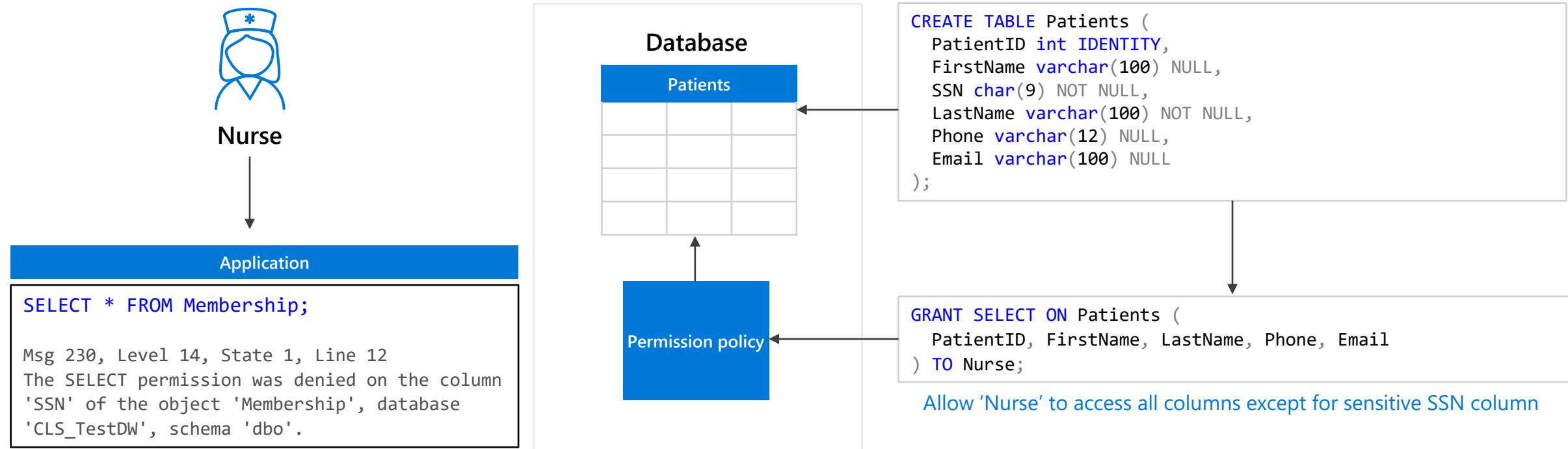
Both Azure Active Directory (AAD) and SQL authentication are supported.



# Column-level security

## Three steps:

1. Policy manager creates permission policy in T-SQL, binding the policy to the Patients table on a specific group.
2. App user (for example, a nurse) selects from Patients table.
3. Permission policy prevents access on sensitive data.



Queries executed as 'Nurse' will fail if they include the SSN column

# Data Protection - Business requirements



How do I protect sensitive data against unauthorized (high-privileged) users?

What key management options do I have?



# Dynamic Data Masking

## Overview

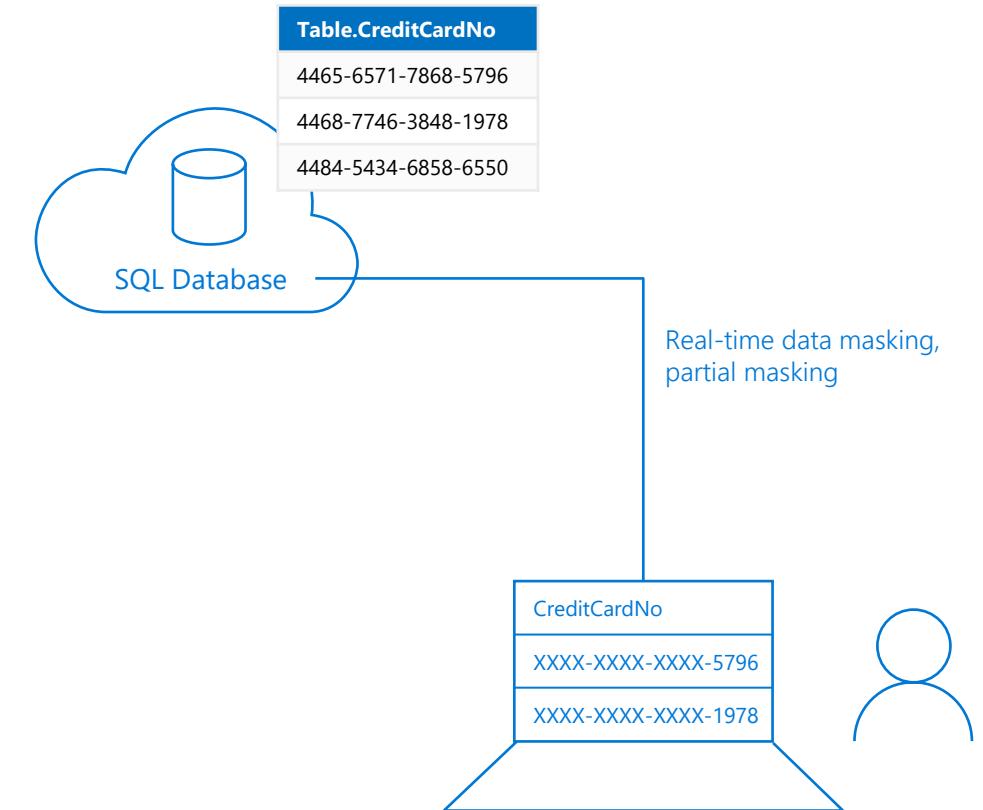
Prevent abuse of sensitive data by hiding it from users

Easy configuration in new Azure Portal

Policy-driven at table and column level, for a defined set of users

Data masking applied in real-time to query results based on policy

Multiple masking functions available, such as full or partial, for various sensitive data categories (credit card numbers, SSN, etc.)



# Dynamic Data Masking

## Three steps

1. Security officer defines dynamic data masking policy in T-SQL over sensitive data in the Employee table. The security officer uses the built-in masking functions (default, email, random)
2. The app-user selects from the Employee table
3. The dynamic data masking policy obfuscates the sensitive data in the query results for non-privileged users



Security officer

```

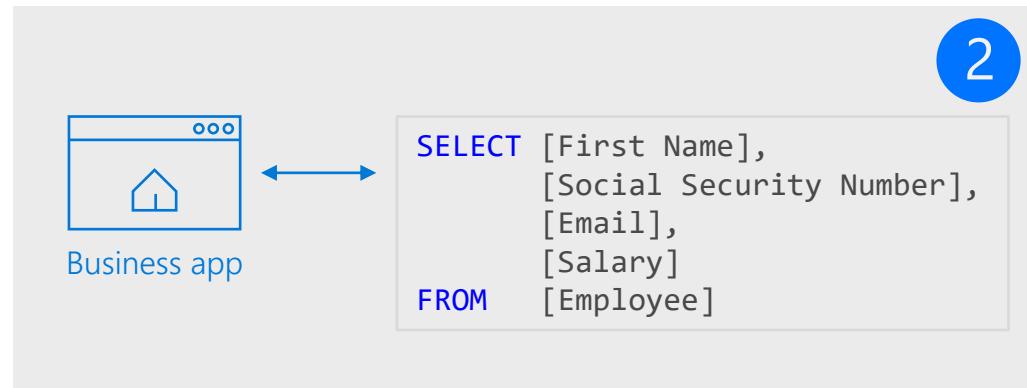
ALTER TABLE [Employee]
ALTER COLUMN [SocialSecurityNumber]
ADD MASKED WITH (FUNCTION = 'DEFAULT()')

ALTER TABLE [Employee]
ALTER COLUMN [Email]
ADD MASKED WITH (FUNCTION = 'EMAIL()')

ALTER TABLE [Employee]
ALTER COLUMN [Salary]
ADD MASKED WITH (FUNCTION = 'RANDOM(1,20000)')

GRANT UNMASK to admin1
    
```

1



2

Diagram illustrating Step 3:

	First Name	Social Security Num...	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net	1041308

Non-masked data (admin login)

	First Name	Social Security Number	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net	1041308

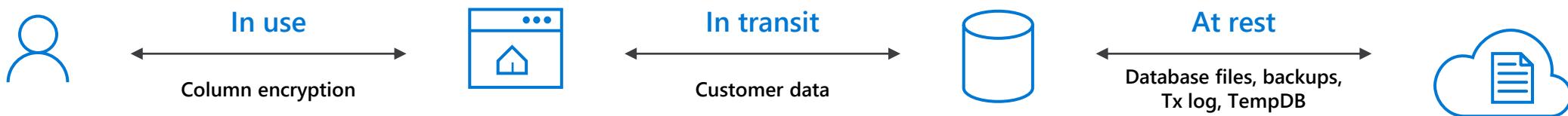
Masked data (admin1 login)

	First Name	Social Security Number	Email	Salary
1	LILA	XXX-XX-XX37	IXX@XXXX.net	8940
2	JAMIE	XXX-XX-XX14	jXX@XXXX.com	19582
3	SHELLEY	XXX-XX-XX28	sXX@XXXX.net	3713
4	MARCELLA	XXX-XX-XX65	mXX@XXXX.net	11572
5	GILBERT	XXX-XX-XX87	gXX@XXXX.net	4487

3

# Types of data encryption

Data Encryption	Encryption Technology	Customer Value
In transit	Transport Layer Security (TLS) from the client to the server TLS 1.2	Protects data between client and server against snooping and man-in-the-middle attacks
At rest	Transparent Data Encryption (TDE) for Azure Synapse Analytics	Protects data on the disk User or Service Managed key management is handled by Azure, which makes it easier to obtain compliance



# Transparent data encryption (TDE)

## Overview

All customer data encrypted at rest

TDE performs real-time I/O encryption and decryption of the data and log files.

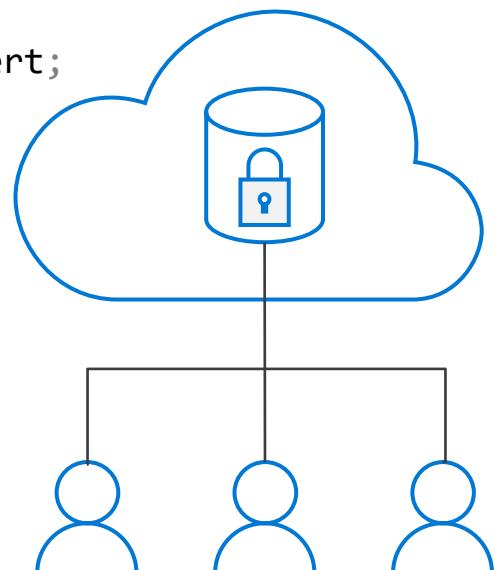
Service OR User managed keys.

Application changes kept to a minimum.

Transparent encryption/decryption of data in a TDE-enabled client driver.

Compliant with many laws, regulations, and guidelines established across various industries.

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<UseStrongPasswordHere>';
go
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My DEK Certificate';
go
USE MyDatabase;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
GO
ALTER DATABASE MyDatabase
SET ENCRYPTION ON;
GO
```



# Transparent data encryption (TDE)

## Key Vault

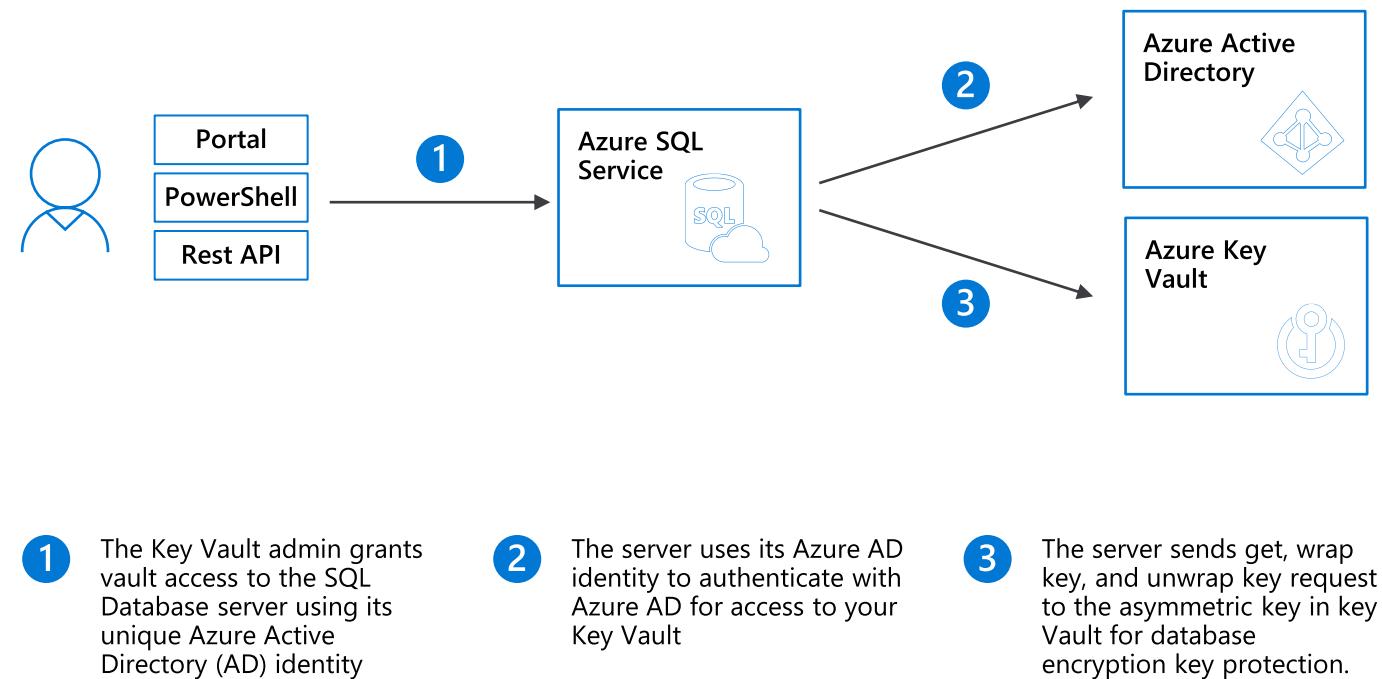
### Benefits with User Managed Keys

Assume more control over who has access to your data and when.

Highly available and scalable cloud-based key store.

Central key management that allows separation of key management and data.

Configurable via Azure Portal, PowerShell, and REST API.





# Azure Synapse Analytics

## Metastore

# Metastore

## Overview

It offers the different computational engines of a workspace to share databases and Parquet-backed tables between its Apache Spark pools, SQL serverless, and SQL provisioned.

## Benefits

- The shared metadata model supports the modern data warehouse pattern.
- The Spark created databases and all their tables become visible in any of the Azure Synapse workspace Spark pool instances and can be used from any of the Spark jobs provided necessary permissions are provided.
- Databases are created automatically in the SQL serverless metadata.
- The external and managed tables created by Spark job are made accessible as external tables in the SQL serverless metadata in the dbo schema of the corresponding database.
- Spark created databases and their Parquet-backed tables will be mapped into the SQL pools for which metadata synchronization enabled.



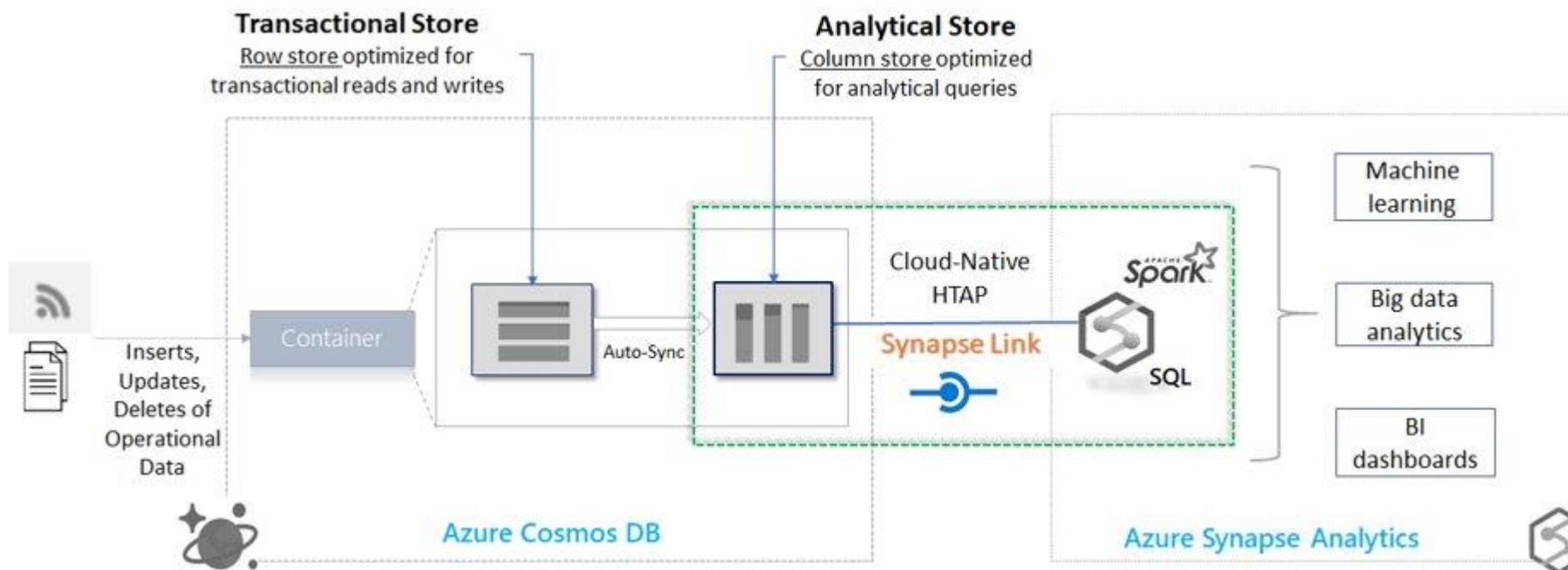
# Azure Synapse Analytics

## Synapse Link for Cosmos DB

# Synapse Link for Cosmos DB integration

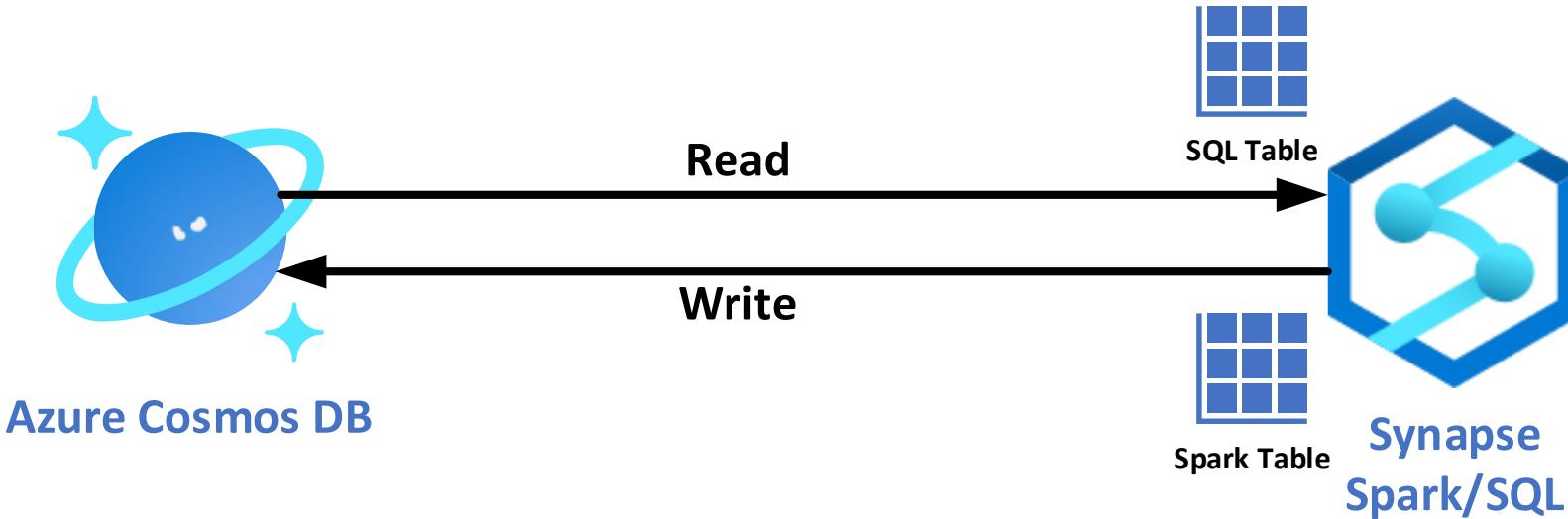
Cloud native hybrid transactional and analytical processing (HTAP) capability

In-place analytics over operational data in Azure Cosmos DB



# Modern Data Warehouse Architecture

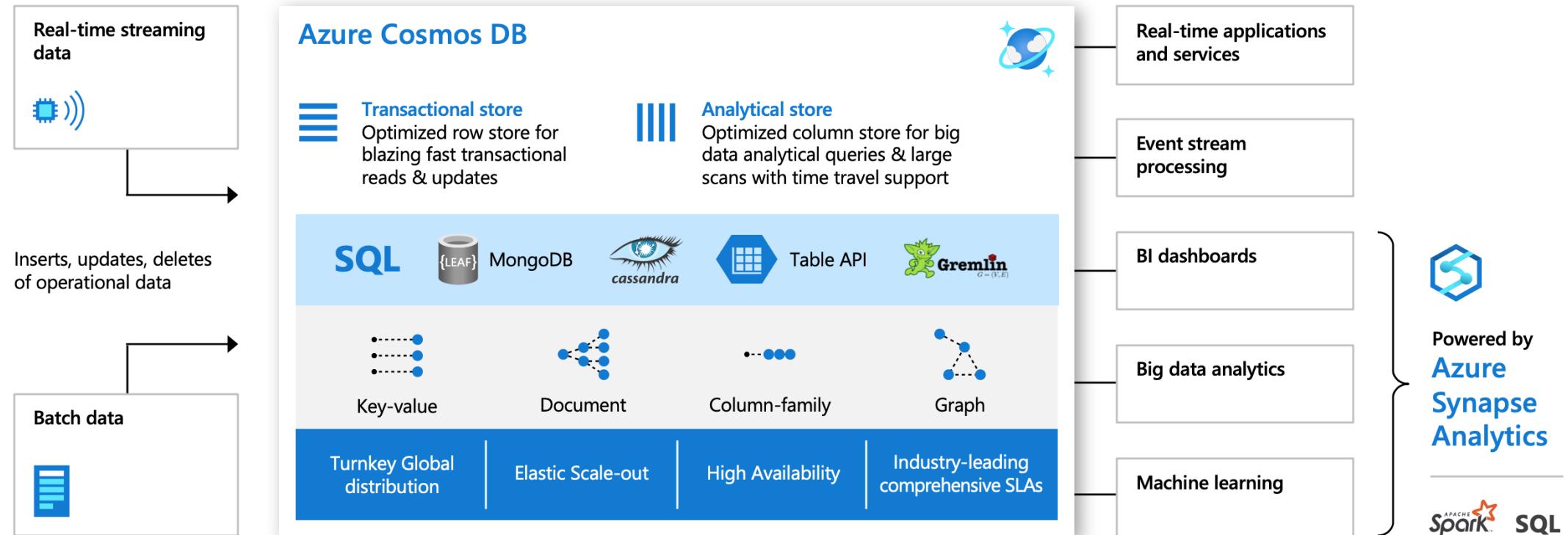
## *With Synapse Link for Cosmos DB*



HTAP reduces architectural complexity by collapsing the stack to:

- Accelerate new insight to perform near-real time analytics at the source with no impact on the transactional workload performance
- Deliver analytics and operational insights through Spark or SQL

# (Cosmos DB + Synapse) ❤ Analytics



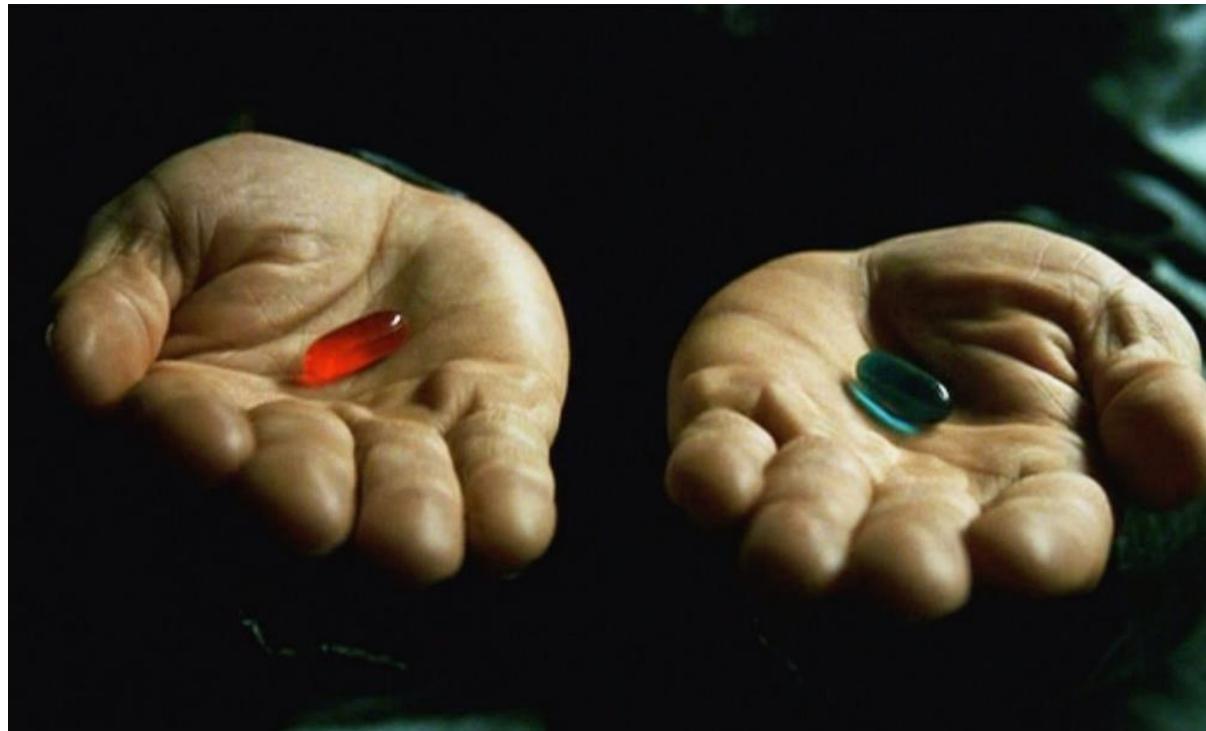
# Benefits of running HTAP in Cosmos DB and Synapse

## Operational Needs

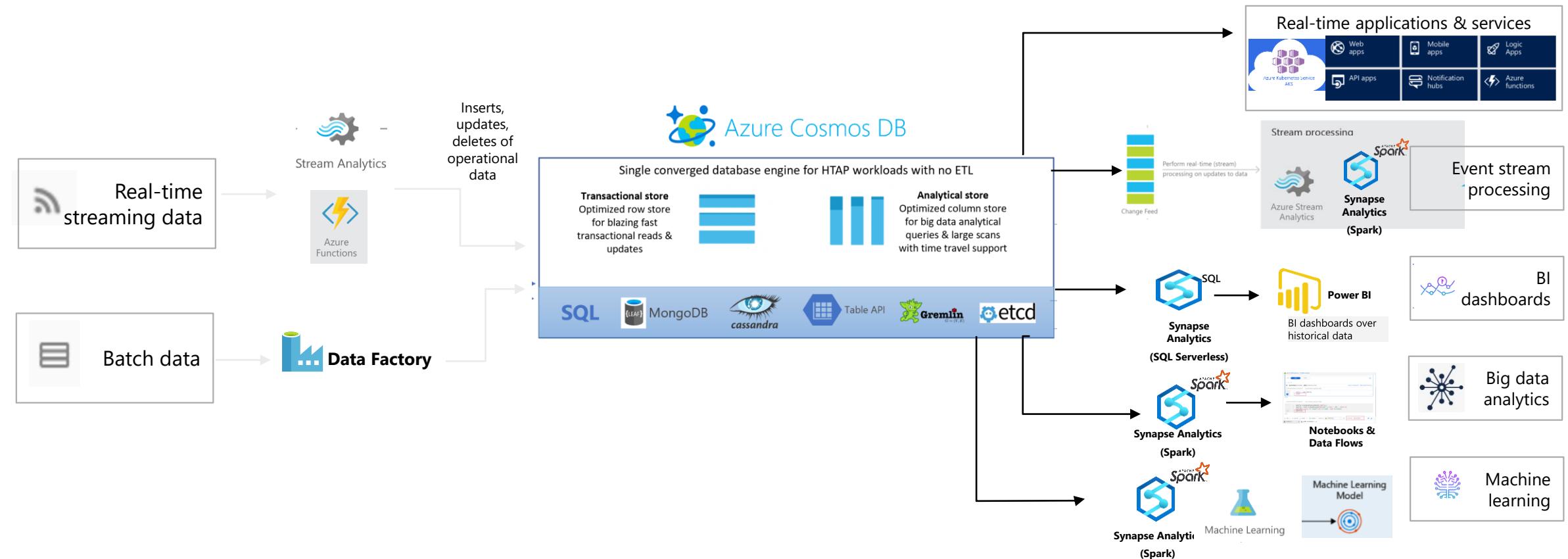
- High throughput ingestion of both batch & streaming feeds with real-time indexing
- Global distribution with active-active setup
- Apps to perform real-time CRUD & queries using developer friendly NoSQL APIs
- No downtime elastic scaling of database
- SLAs for latency/availability/consistency/throughput

## Analytics Needs

- Run analytics without impacting operational performance
- Low cost on storage and compute to run near-real time analytics workloads
- Analyze in-place data, eliminating ETL complexity, with Synapse Spark and SQL
- Rich BI ecosystem of SQL available for cheaper & faster queries over analytical storage
- Native integration of Azure Enterprise AI ecosystem for training and scoring ML models over analytical storage



# High level reference architecture for HTAP workloads





End