



Azure Synapse Analytics

James Serra

Data & AI Architect

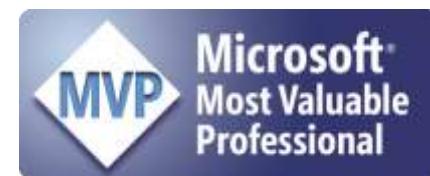
Microsoft, NYC MTC

JamesSerra3@gmail.com

Blog: JamesSerra.com

About Me

- Microsoft, Big Data Evangelist
- In IT for 30 years, worked on many BI and DW projects
- Worked as desktop/web/database developer, DBA, BI and DW architect and developer, MDM architect, PDW/APS developer
- Been perm employee, contractor, consultant, business owner
- Presenter at PASS Business Analytics Conference, PASS Summit, Enterprise Data World conference
- Certifications: MCSE: Data Platform, Business Intelligence; MS: Architecting Microsoft Azure Solutions, Design and Implement Big Data Analytics Solutions, Design and Implement Cloud Data Platform Solutions
- Blog at JamesSerra.com
- Former SQL Server MVP
- Author of book "Reporting with Microsoft SQL Server 2012"



Agenda

- Introduction
- Studio
- Data Integration
- SQL Analytics
- Data Storage and Performance Optimizations
- SQL On-Demand
- Spark
- Security
- Connected Services

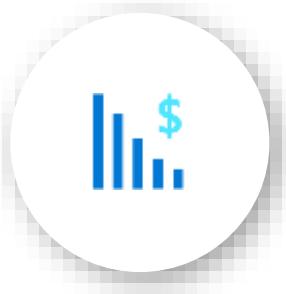


Azure Synapse Analytics is a limitless analytics service, that brings together enterprise data warehousing and Big Data analytics. It gives you the freedom to query data on your terms, using either serverless on-demand or provisioned resources, at scale. Azure Synapse brings these two worlds together with a unified experience to ingest, prepare, manage, and serve data for immediate business intelligence and machine learning needs.

Azure Synapse – SQL Analytics

focus areas

Best in class price per performance



Up to 94% less expensive than competitors

Industry-leading security



Defense-in-depth security and 99.9% financially backed availability SLA

Workload aware query execution



Manage heterogenous workloads through workload priorities and isolation

Data flexibility



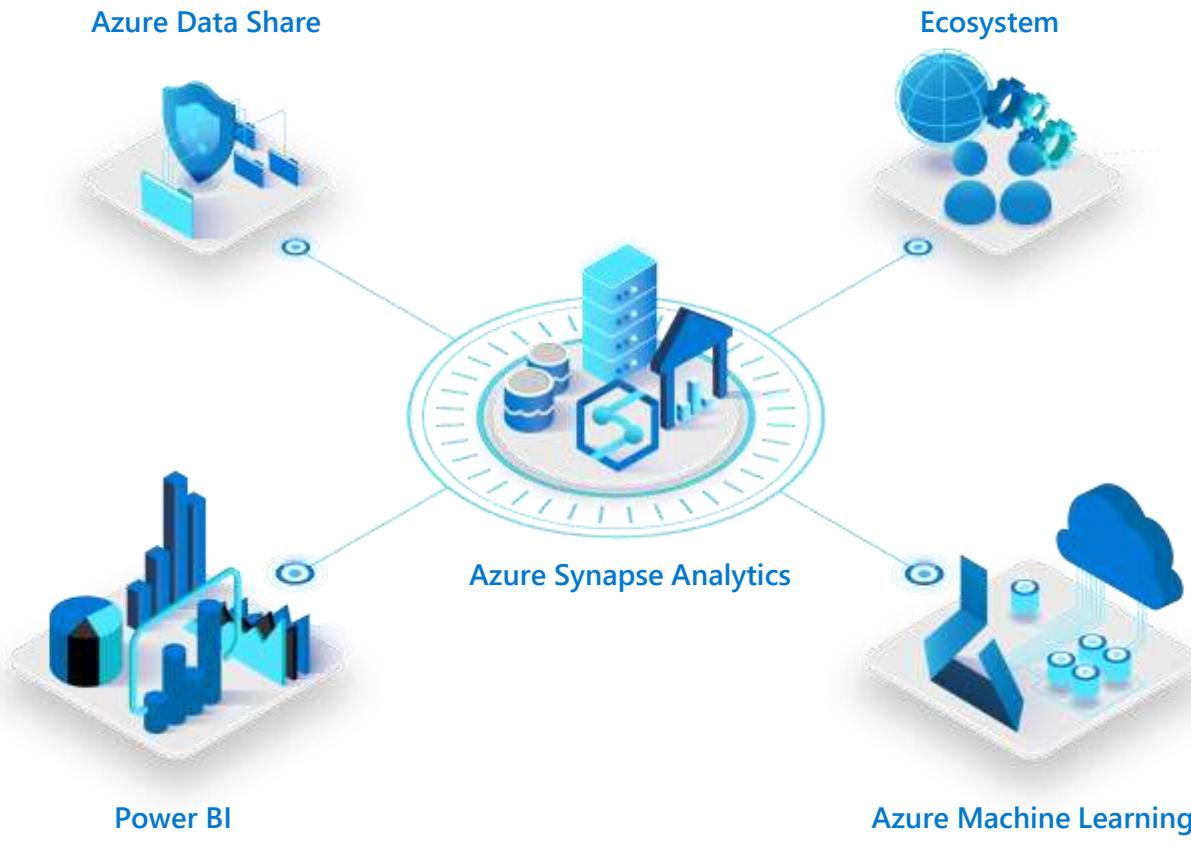
Ingest variety of data sources to derive the maximum benefit.
Query all data.

Developer productivity



Use preferred tooling for SQL data warehouse development

Leveraging ISV partners with Azure Synapse Analytics



+ many more

Full backward compatibility with Azure SQL Data Warehouse for data integration and orchestration

Additional analytics capabilities in Azure Synapse unlocks new ISV scenarios

Azure Synapse + ISV can bring data continuity with Azure Machine Learning and Power BI

Reduce migration effort by reusing existing partner platforms

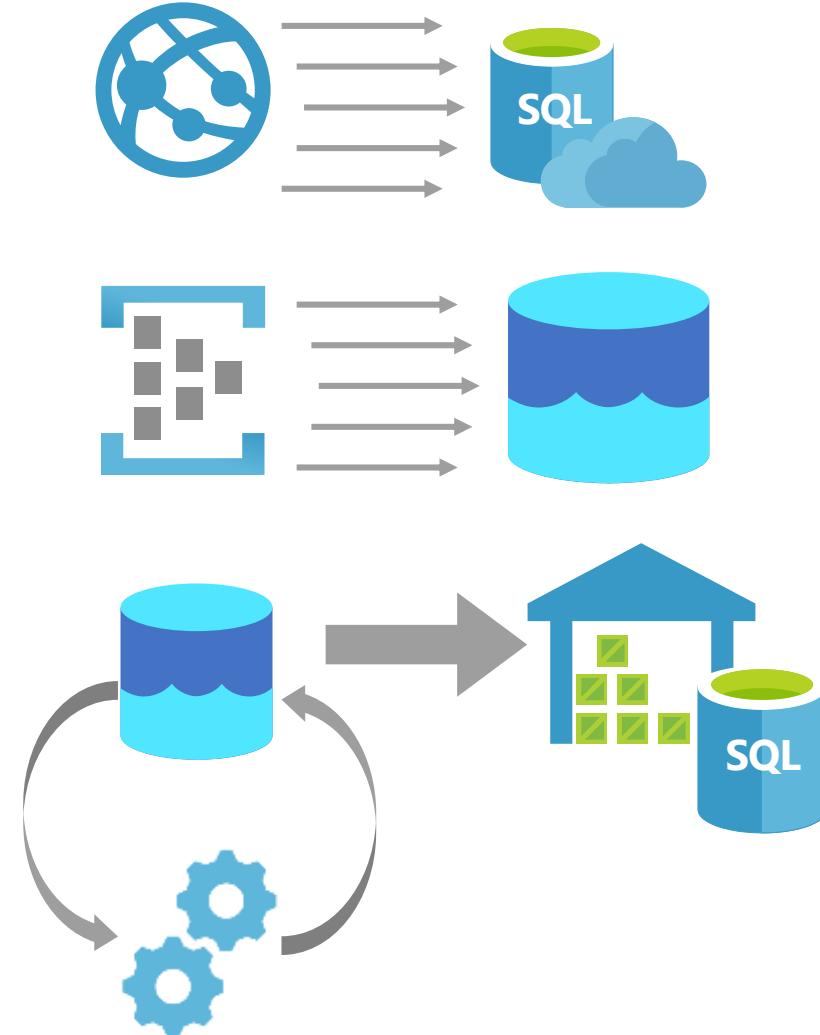
What workloads are NOT suitable?

Operational workloads (OLTP)

- High frequency reads and writes.
- Large numbers of singleton selects.
- High volumes of single row inserts.

Data Preparations

- Row by row processing needs.
- Incompatible formats (XML).



What Workloads are Suitable?

Analytics

Store large volumes of data.

Consolidate disparate data into a single location.

Shape, model, transform and aggregate data.

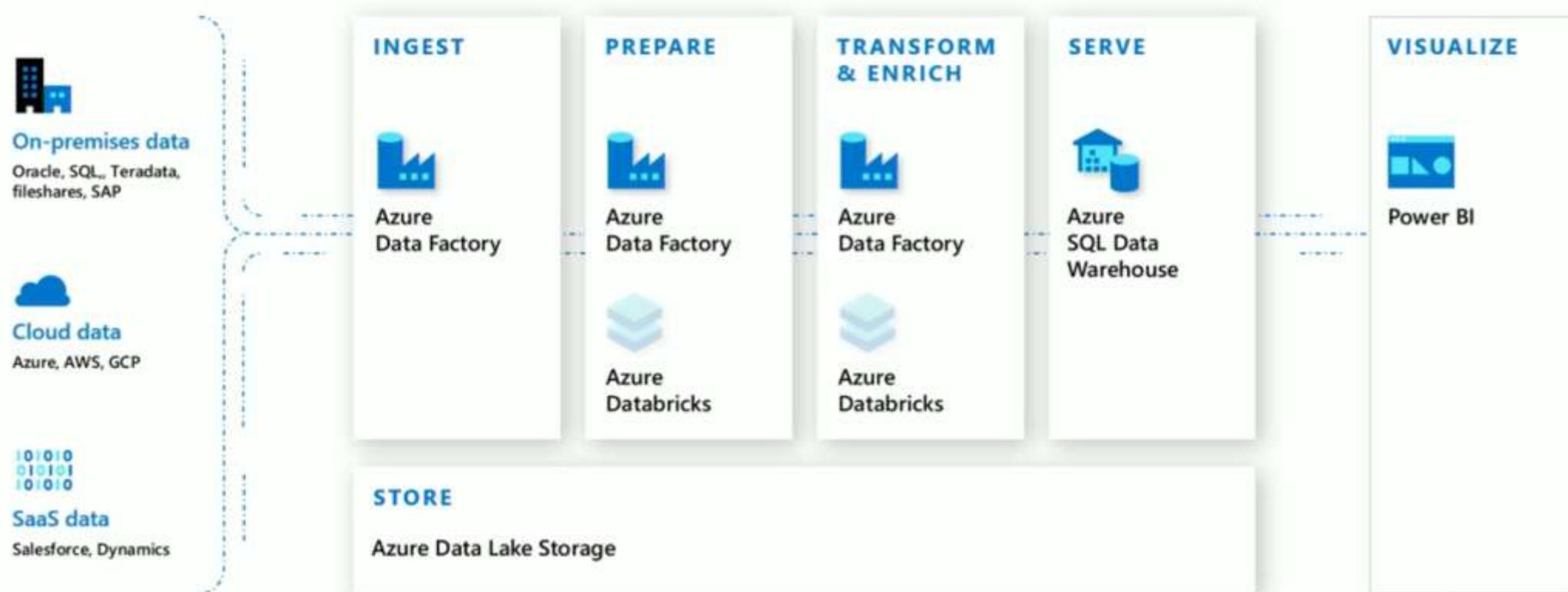
Batch/Micro-batch loads.

Perform query analysis across large datasets.

Ad-hoc reporting across large data volumes.

All using simple SQL constructs.

Modern Data Warehouse

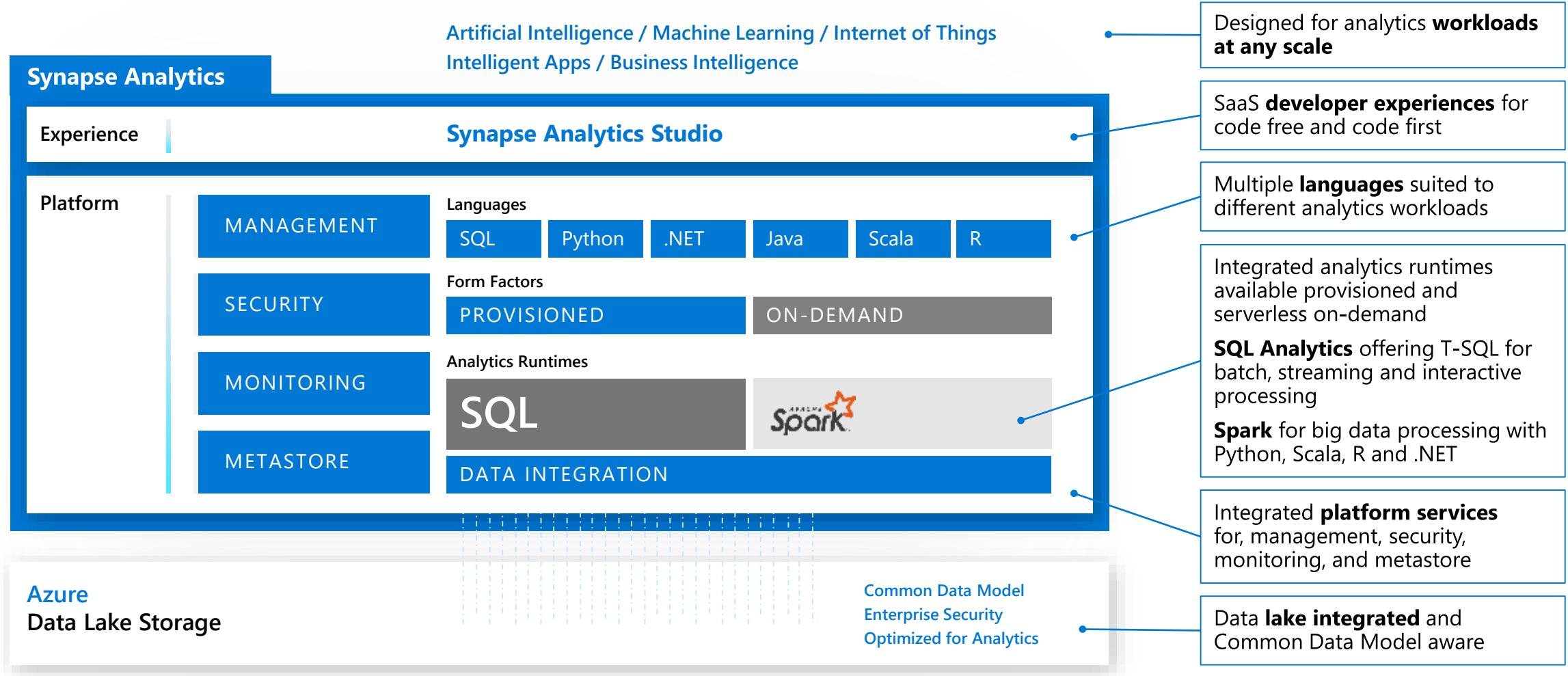


Azure Synapse Analytics



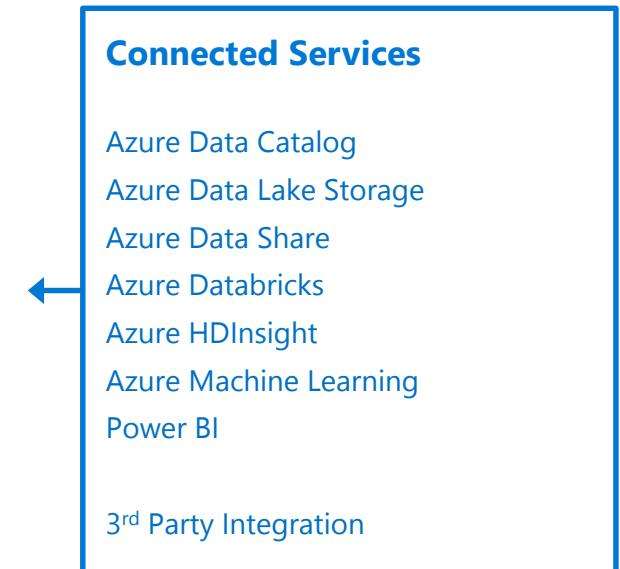
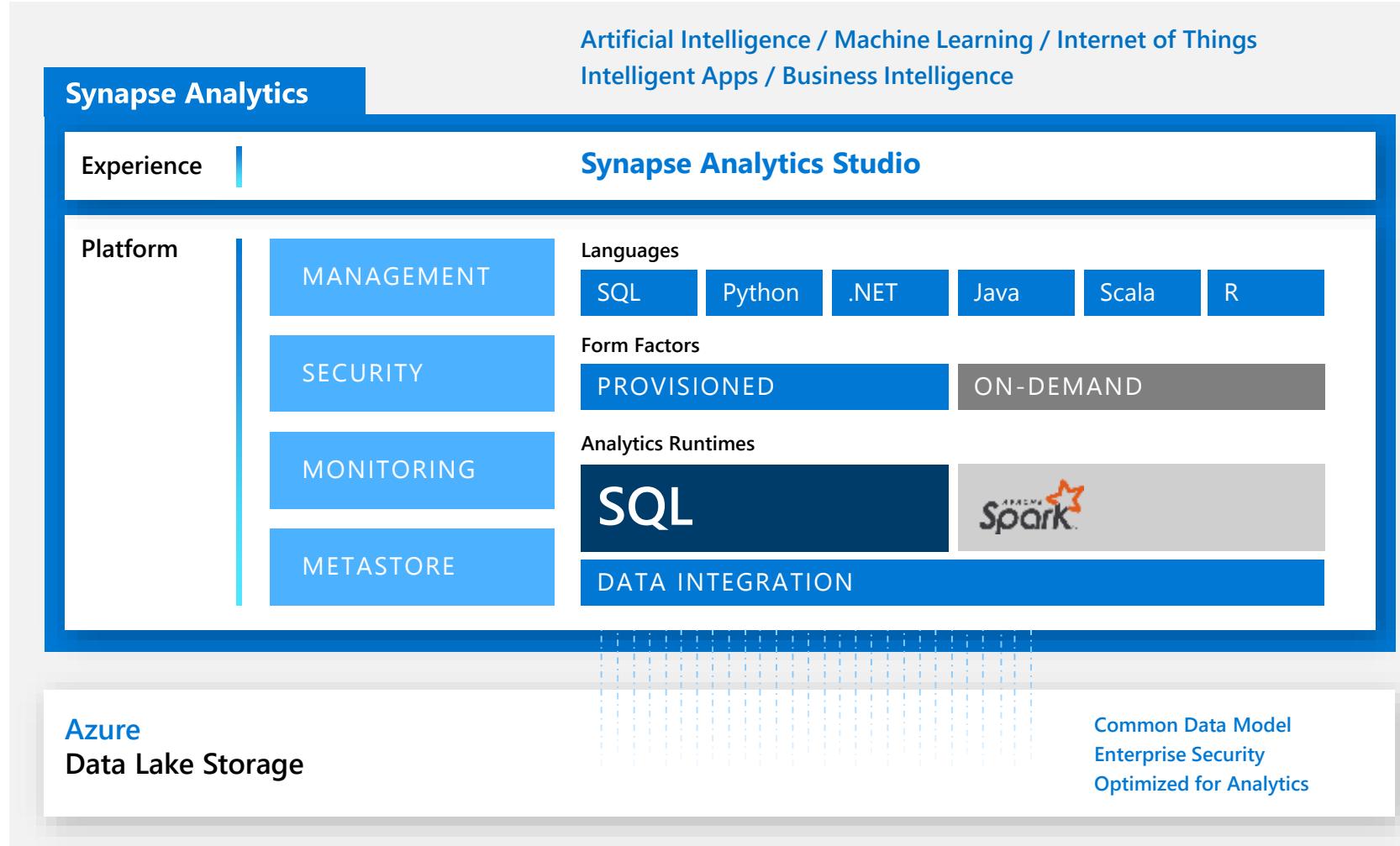
Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Provisioning Synapse workspace

Providing Synapse is easy

Subscription

Resource Group

Workspace Name

Region

Data Lake Storage Account

Home > Synapse workspaces > Create Synapse workspace

Create Synapse workspace

Basics * Security + networking Tags Summary

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Subscription * ⓘ

Resource group * ⓘ Select existing...

Workspace details

Name your workspace, select a location, and choose a primary Data Lake Storage Gen2 file system to serve as the default location for logs and job output.

Workspace name * Enter workspace name

Region *

Select Data Lake Storage Gen2 * ⓘ From subscription Manually via URL

Account Name *

File system name *

Info The managed identity of the workspace will be assigned the [Storage Blob Data Contributor](#) role on the selected Data Lake Storage Gen2 file system, granting it full data access.

Synapse workspace

 internal sandboxwe
Synapse workspace

Search (Ctrl + F)

+ New SQL pool + New Apache Spark pool Refresh Reset SQL admin password Delete Launch Synapse Studio

Resource group (change) : Arcadia-Private-Preview-BASE
Status : Succeeded
Location : West Europe
Subscription (change) : BigDataPMinternal
Subscription ID : 58f8824d-32b0-4825-9825-02fa6a801546
Managed Identity object... : 5eff8ac2-fd6f-4b09-84fd-760bab64802c

Firewalls : Show firewall settings
Primary ADLS Gen2 account : https://internalsandboxwe.dfs.core.windows.net
Primary ADLS Gen2 file system : tempdata
SQL Active Directory ad... : acomet@microsoft.com
SQL endpoint : internalsandboxwe.sql.azuresynapse.net
SQL on-demand endpoint : internalsandboxwe-ondemand.sql.azuresynapse.net
Development endpoint : https://internalsandboxwe.dev.azuresynapse.net
Workspace web URL : https://web.azuresynapse.net?workspace=%2bsubscr

Tags (change) : pointOfContact : <unknown>

Available resources

Search to filter items...

| Name | Size | Type |
|--------------------|---------|-------------------|
| SQL pools | | |
| SQLPoolSandbox | DW1000c | SQL pool |
| Apache Spark pools | | |
| SparkSandbox | Medium | Apache Spark pool |

SQL pools

Apache Spark pools

Firewalls

Alerts

Metrics

Diagnostic settings

Logs

Advisor recommendations

New support request

SQL pools

+ New Refresh

Search to filter items...

| Name | Type | Status | Size |
|-----------------|-------------------------|----------|---------|
| SQL on-demand | SQL Analytics on-demand | N/A | N/A |
| SQLPoolSandbox | SQL Analytics pool | ✓ Online | DW1000c |
| SQLSandboxLarge | SQL Analytics pool | ✓ Online | DW2000c |
| SQLSandboxSmall | SQL Analytics pool | ✓ Online | DW100c |

Create SQL pool

Synapse

Basics * Additional settings * Tags Review + create

Create a SQL pool with your Preferred Configuration. Complete the basics tab then go to Review + create provision with smart defaults. [Learn more](#)

SQL pool Details

Name your SQL pool and choose its initial settings.

SQL pool Name *

Enter SQL pool Name

Performance level ⓘ



DW1000c

Basics *

Additional settings *

Tags

Review + create

Customize additional configuration parameters including collation & sample data.

Data source

Start with a blank SQL pool, restore from a backup or select sample data to populate your new SQL pool.

Use existing data *

None Backup

SQL pool collation

Collation defines the rules that sort and compare data, and cannot be changed after SQL pool creation. The default collation is SQL_Latin1_General_CI_AS. [Learn more](#)

Collation * ⓘ

SQL_Latin1_General_CI_AS

SQL Analytics pool = SQL Data Warehouse

Apache Spark pools

[+ New](#) [Refresh](#)

| Name | ↑↓ | Size |
|--------------|----|--|
| SparkSandbox | | Medium (8 vCPU / 64 GB) - 3 to 20 nodes |
| SparkSmall | | Small (4 vCPU / 32 GB) - 3 to 20 nodes |
| SparkLarge | | Large (16 vCPU / 128 GB) - 3 to 80 nodes |

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 *

Node size family

MemoryOptimized

Node size *

Autoscale * ⓘ

[Enabled](#) [Disabled](#)

Number of nodes *

Note: There are no on-demand pools for Spark

[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](#)

15

Component versions

Select the Apache Spark version for your Apache Spark pool.

Apache Spark *

Python

3.6.1

Scala

2.11.12

Java

1.8.0_222

.NET Core

3.0

.NET for Apache Spark

0.6.0

Delta Lake

0.4.0

Packages

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

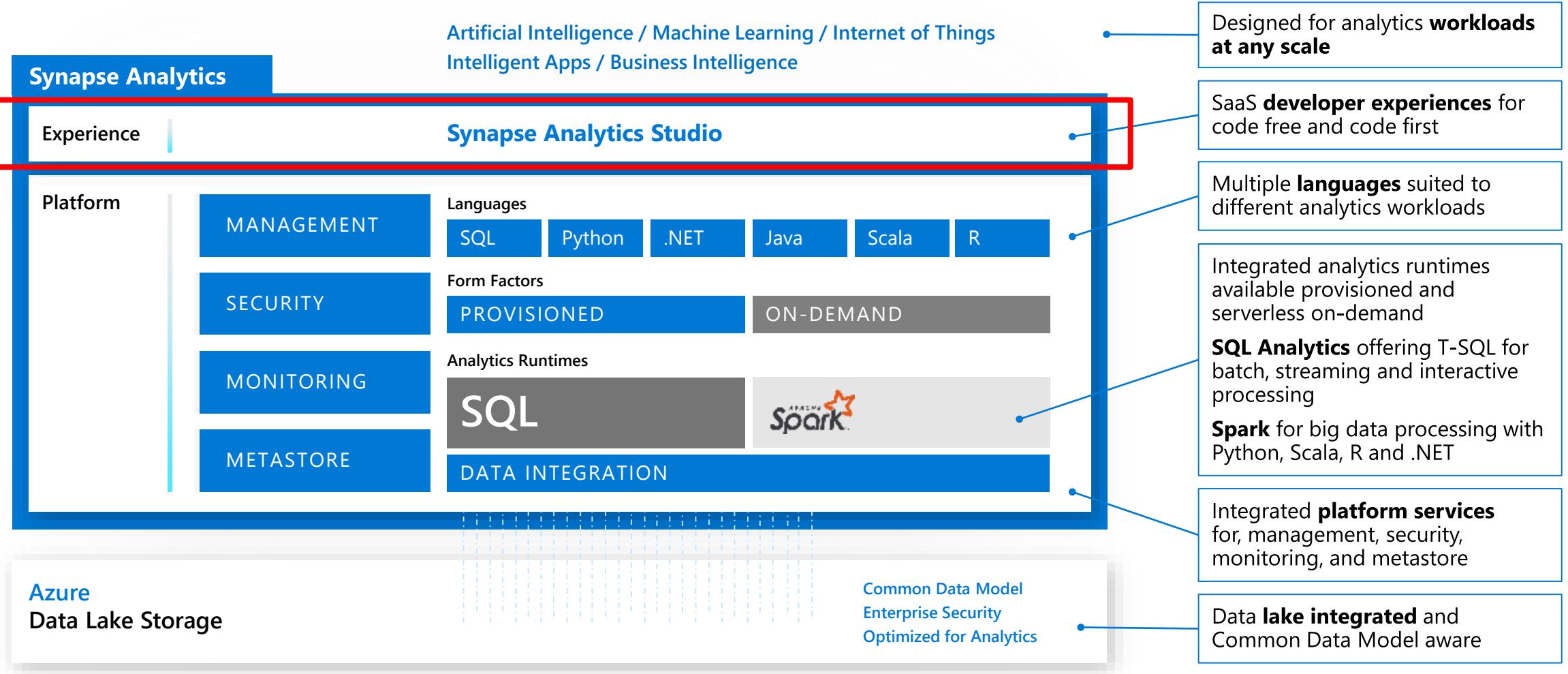
File upload



Azure Synapse Analytics Studio

Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Studio

<https://web.azuresynapse.net>

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

Microsoft Azure | Synapse Analytics ▶ prlangadws2

Synapse workspace
prlangadws2

New ▾

Overview Data Develop Orchestrate Monitor Manage

Ingest Explore Analyze Visualize

Resources

Recent Pinned

| NAME | LAST OPENED BY YOU |
|----------------------------------|--------------------|
| GreenCabTransformation | a day ago |
| EXE2 StoredProceduresCabs | a day ago |
| EXE3 Query Market Share SQL Pool | a day ago |
| EXE5 Query SQL OD Views | a day ago |
| EXE5 Create SQL OD Views | a day ago |

Show more ▾

Name: prlangadws2
Region: West US 2
Resource group: prlangadrg
Subscription ID: 58f8824d-32b0-4825-9825-02fa6a801546
[Select another workspace](#)

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's a sidebar with a red box highlighting the 'Overview' section. A red arrow points from the 'Overview' button in the sidebar to the 'Overview' card in the main content area. The main content area is divided into six activity hubs:

- 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.

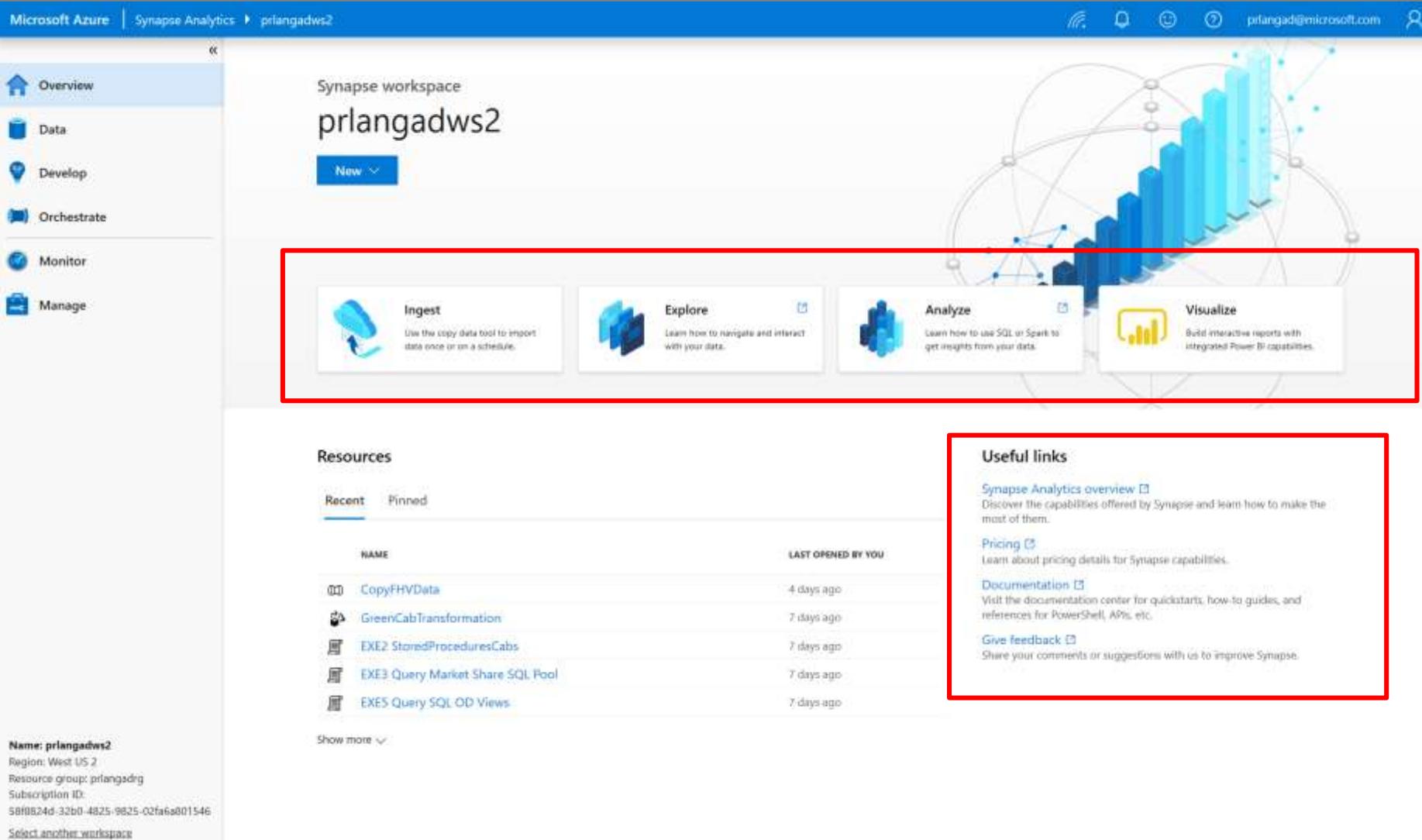
At the bottom left, there's a summary of the workspace details: Name: prlangadws2, Region: West US 2, Resource group: prlangadrg, Subscription ID: 50000000-0000-0000-0000-000000000000, and a 'Show more' link.



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 for the workspace "prlangadws2". The left sidebar includes links for Overview, Data, Develop, Orchestrate, Monitor, and Manage. The main area features a "Synapse workspace" header with the name "prlangadws2" and a "New" button. Below this is a large circular graphic illustrating data flow and analysis. Four cards are displayed: "Ingest" (using the copy data tool), "Explore" (navigating and interacting with data), "Analyze" (using SQL or Spark), and "Visualize" (building reports with Power BI). A red box highlights these four cards. Further down, there are sections for "Resources" (Recent and Pinned items like CopyPHVData, GreenCabTransformation, etc.) and "Useful links" (Synapse Analytics overview, Pricing, Documentation, and Give feedback). At the bottom, workspace details are shown: Name: prlangadws2, Region: West US 2, Resource group: prlangadrg, Subscription ID: 58f0824d-32b0-4825-9825-02fa6a801546, and a link to Select another workspace.

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.

Resources

Recent Pinned

| NAME | LAST OPENED BY YOU |
|----------------------------------|--------------------|
| CopyPHVData | 4 days ago |
| GreenCabTransformation | 7 days ago |
| EXE2_StoredProcedureCabs | 7 days ago |
| EXE3_Query Market Share SQL Pool | 7 days ago |
| EXE5_Query SQL OD Views | 7 days ago |

Show more ↴

Name: prlangadws2
Region: West US 2
Resource group: prlangadrg
Subscription ID: 58f0824d-32b0-4825-9825-02fa6a801546
[Select another workspace](#)

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.

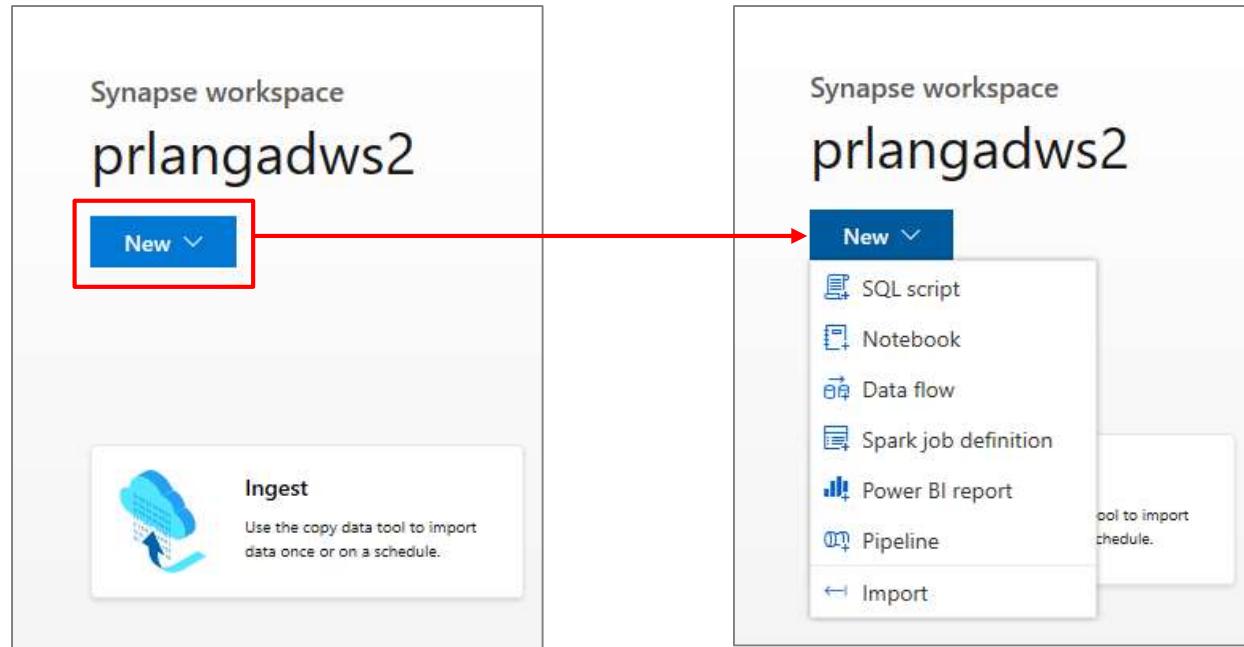
[Give feedback](#) ⓘ
Share your comments or suggestions with us to improve Synapse.

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 | | | prlangad@microsoft.com |

Overview | Data (selected) | Develop | Orchestrate | Monitor | Manage

Publish all Validate all Refresh Discard all

Data

Storage accounts

Databases

Datasets

Select an item from the resource explorer or create a new item

Name: prlangadws2
Region: West US 2
Resource group: prlangadrg
Subscription ID:
58f8824d-32b0-4825-9825-02fa6a801546
[Select another workspace](#)

Data Hub – Storage accounts

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

ADLS Gen2 Account

Container (filesystem)

The screenshot shows the Microsoft Azure Data Hub interface for managing storage accounts. On the left, a sidebar lists navigation options: Overview, Data (selected), Develop, Orchestrate, Monitor, and Manage. Below this, account details are shown: Name: prlangadws2, Region: West US 2, Resource group: prlangadrg, Subscription ID: 58f8824d-32b0-4825-9825-02fa6a801546. The main area is titled 'Data' and shows a 'Storage accounts' section. A red box highlights the 'prlangaddemosa (Primary)' storage account. Another red box highlights the 'nyctlc' container within it. The 'Filepath' bar shows the path 'nyctlc > yellow'. The right side displays a file browser with a list of folders named 'puYear=2001' through 'puYear=2026', all of which are 'Folders' and were modified on 10/25/2019.

| 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, 2:20:20 PM | Folder | |

Data Hub – Storage accounts

Preview a sample of your data

The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'nyctlc' is selected, and under it, 'filesystem' is also selected. In the main area, a file named 'SampleCSVFile_2kb.csv' is listed under 'nyctlc'. A red arrow points from the 'Preview' option in the context menu for this file to the preview window on the right.

SampleCSVFile_2kb.csv

Path https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv
Modified 10/29/2019, 1:30:21 PM

With column header On

| 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 |

OK

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 sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'prlangaddemosa (Primary)' is expanded, showing its contents: filesystem, holidaydatacontainer, isdweatherdatacontainer, nyctlc, prlangaddemosa, tmpcontainer, and wwidporters. The 'filesystem' item is selected. In the main area, there are tabs for 'nyctlc' and 'filesystem'. The 'filesystem' tab is active, showing a list of files and folders: synapse, temp, tmp, dbo.StoreSales.parquet, dbo.StoreSales.txt, employee.json, and SampleCSVFile_2kb.csv. The 'SampleCSVFile_2kb.csv' file is selected. A context menu is open for this file, with options: Preview, New notebook, Copy ABFSS path, Manage Access..., Rename..., Download, Delete, and Properties... (the last option is highlighted with a red box). At the top of the main area, there are buttons for Publish all, Validate all, Refresh, Discard all, Upload, Download, New Folder, and Select.

The screenshot shows the 'Properties' dialog box for the 'SampleCSVFile_2kb.csv' file. The dialog is divided into 'System Properties' and 'User Properties' sections. In the System Properties section, the Name is 'SampleCSVFile_2kb.csv', the URL is 'https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv', and the LastModified date is 'Tue, 29 Oct 2019 20:30:21 GMT'. The Content Type is listed as 'application/vnd.ms-excel'. The User Properties section is currently empty. At the bottom, there are 'Save' and 'Cancel' buttons. A red arrow points from the 'Properties...' option in the context menu on the left to the 'Properties...' button in the dialog box on the right.

| Properties | |
|-----------------------------------|--|
| System Properties | |
| Name | SampleCSVFile_2kb.csv |
| URL | https://prlangaddemosa.dfs.core.windows.net/filesystem/SampleCSVFile_2kb.csv |
| LastModified | Tue, 29 Oct 2019 20:30:21 GMT |
| CacheControl | |
| ContentType | application/vnd.ms-excel |
| ContentDisposition | |
| ContentEncoding | |
| ContentLanguage | |
| User Properties | |
| Add User Property | |

Data Hub – Storage accounts

Manage Access - Configure standard POSIX ACLs on files and folders

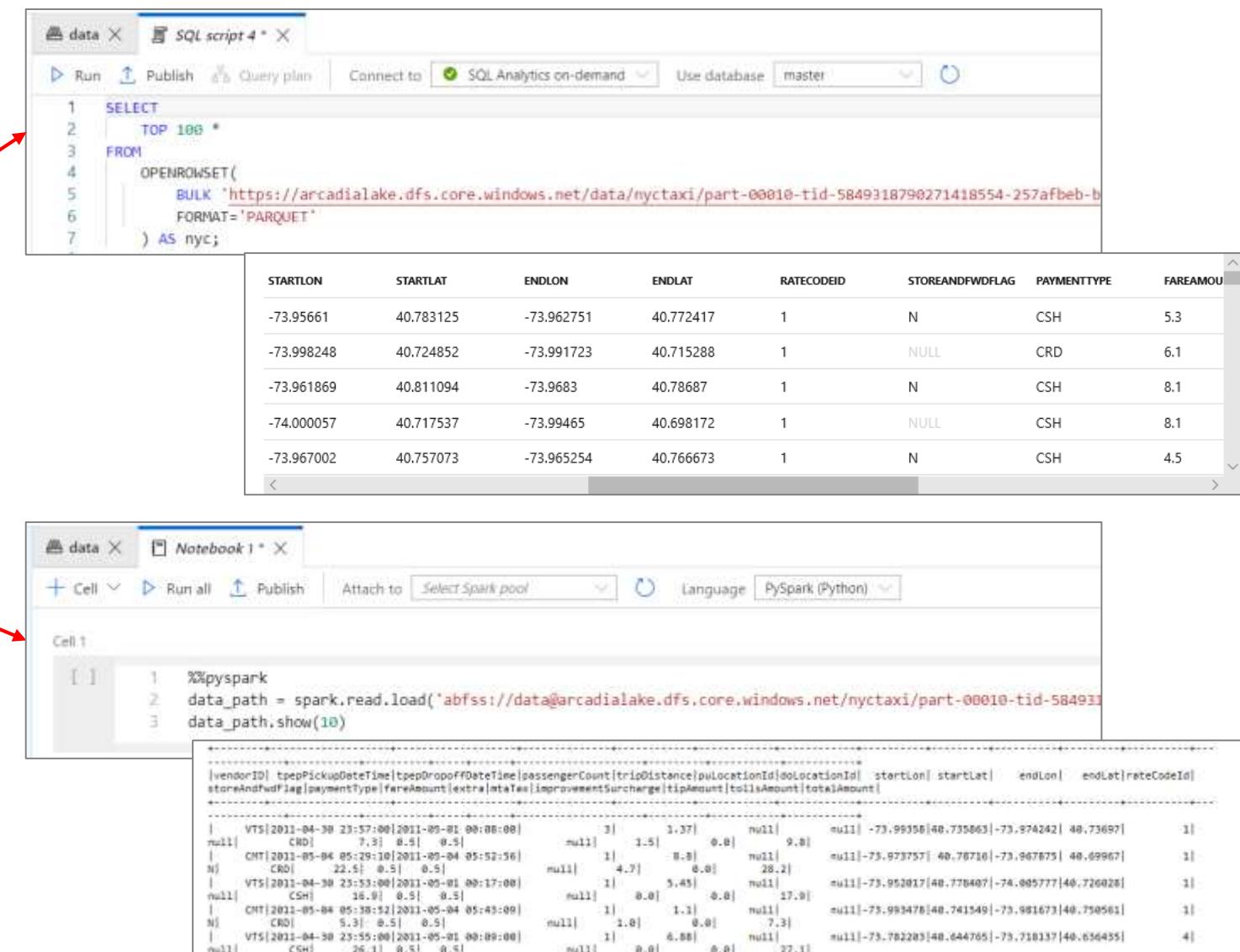
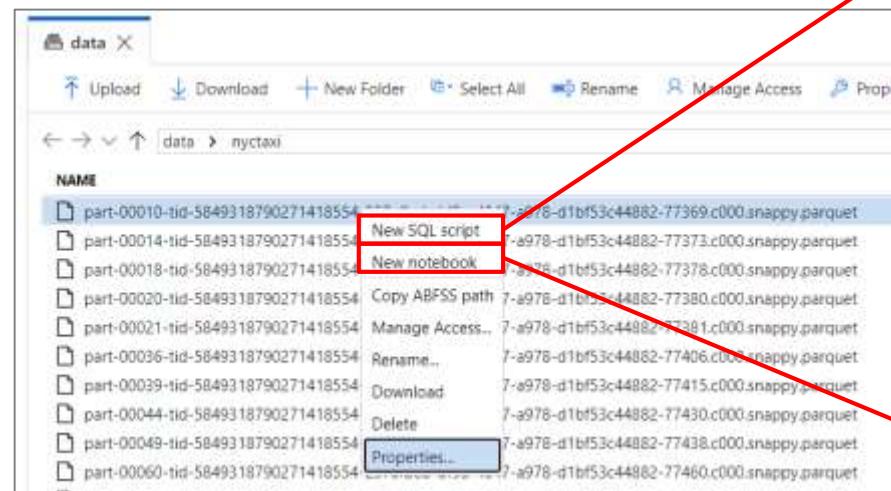
The screenshot shows the Azure Synapse Studio Data Hub interface. On the left, there's a navigation sidebar with sections for Data, Storage accounts, Databases, and Datasets. Under Storage accounts, 'prlangaddemo (Primary)' is selected, showing its contents: filesystem, holidaydatacontainer, isdweatherdatacontainer, nyctlc, prlangaddemo, tmpcontainer, and wwimporters. The 'filesystem' item is also selected. In the main workspace, there are two tabs: 'nyctlc' and 'filesystem'. The 'filesystem' tab is active, displaying a list of files and folders: synapse, temp, tmp, dbo.StoreSales.parquet, dbo.StoreSales.txt, employee.json, and SampleCSVFile_2kb.csv. A context menu is open over the 'SampleCSVFile_2kb.csv' 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 a red arrow pointing to the corresponding window on the right.

The screenshot shows the 'Manage Access' dialog box for the file 'SampleCSVFile_2kb.csv'. The title bar says 'Manage Access' and the sub-header says 'Managing permissions for: filesystem/SampleCSVFile_2kb.csv'. It lists 'Users and groups': '\$superuser (Owner)' and '\$superuser (Owning Group)'. Below that are 'Other' and 'Mask' sections. Under 'Permissions for: \$superuser', there are checkboxes for 'Read', 'Write', and 'Execute', all of which are checked. There's also a section to 'Add user or group' with a text input field 'Enter a UPN or Object ID' and a 'Add' button. At the 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



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 'isdweatherdatacontainer > ISDWeathercurated'. It lists several Parquet files, including '_SUCCESS' and multiple part files. A red arrow points from the 'New SQL script' option in a context menu over one of the part files to the query editor on the right. The query editor contains a T-SQL script for reading multiple Parquet files:

```
1 -- Read multiple parquet files with same schema
2 SELECT
3     TOP 100 *
4 FROM
5     OPENROWSET(
6         BULK 'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/+',
7         FORMAT = 'Parquet'
8     ) AS [r]
9 WHERE
10    r.filepath() in (
11        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00000'
12        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00001'
13        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00002'
14        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00003'
15        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00004'
16        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00005'
17        'https://prlangaddemosa.dfs.core.windows.net/isdweatherdatacontainer/ISDWeathercurated/part-00006'
18    )
```

Data Hub – Databases

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

The diagram illustrates the structure of databases in a workspace, comparing two views: a smaller left pane and a larger right pane.

Left Pane (Data Hub View):

- Storage accounts:** 2 items
- Databases:** 3 items
 - sql1 (SQL pool)** (marked with a red arrow pointing to the right pane)
 - sample (SQL on-demand)** (marked with a red arrow pointing to the right pane)
 - default (Spark)** (marked with a red arrow pointing to the right pane)
 - Tables:** 2 items
 - nytaxiyellow7days**
 - searchlogtable**- Datasets:** 2 items

Right Pane (Detailed Database View):

- Filter resources by name**
- 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.

Azure Synapse Studio interface showing the 'Databases' section. A context menu is open over the 'dbo.NycTaxiPredict' table's columns. The menu options include:

- New SQL script
- 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

Azure Synapse Studio interface showing the 'Databases' section. A context menu is open over the 'dbo.NycTaxiPredict' table's columns. The 'Load to DataFrame' option is highlighted with a red box. A red arrow points down to the generated PySpark code in the notebook below.

The generated PySpark code in the notebook is:

```
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 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

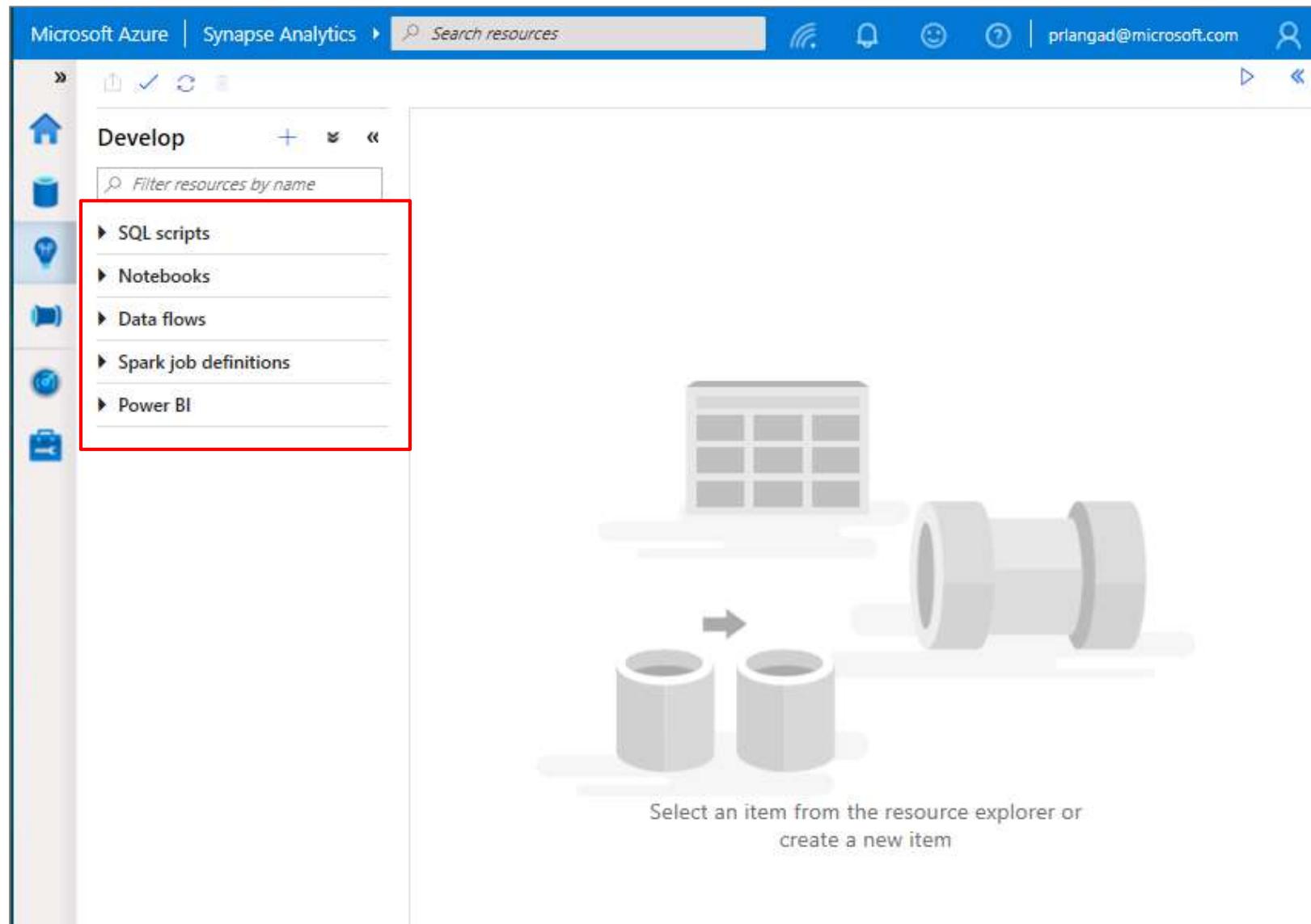
Benefits

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



Develop Hub - SQL scripts

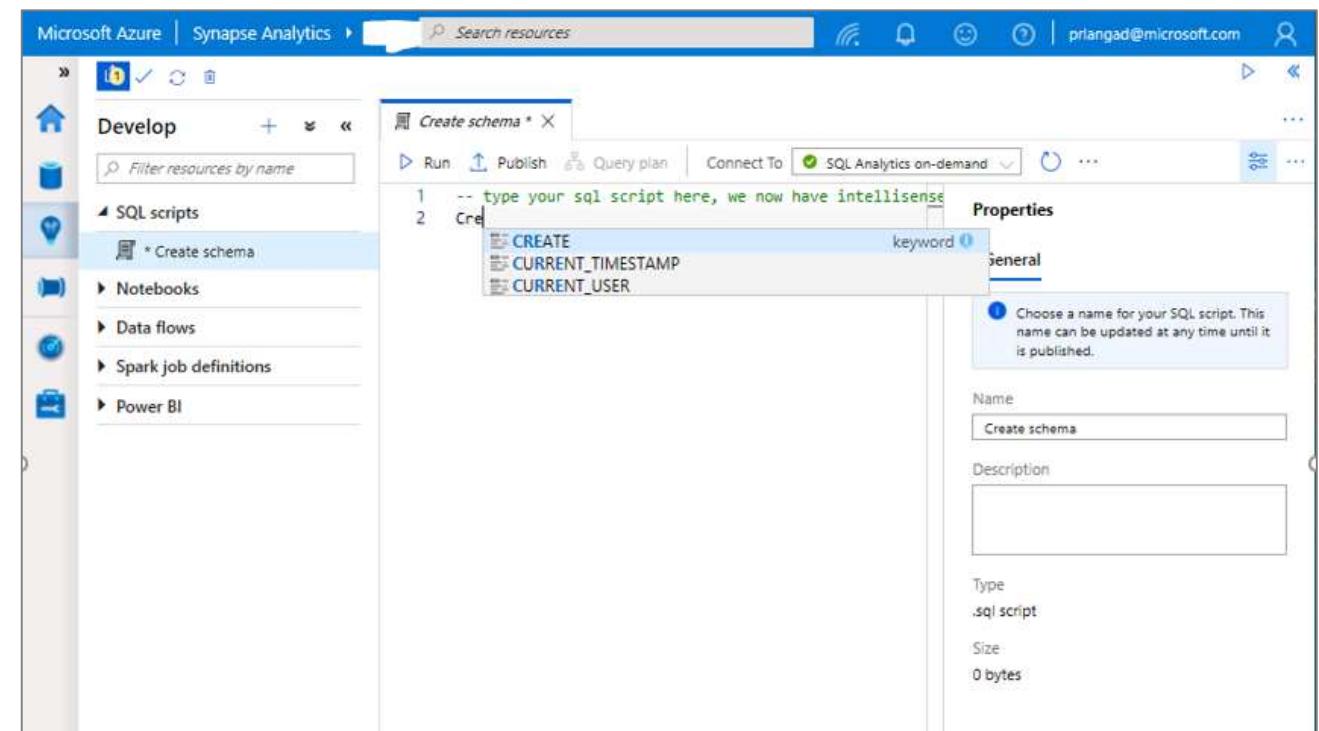
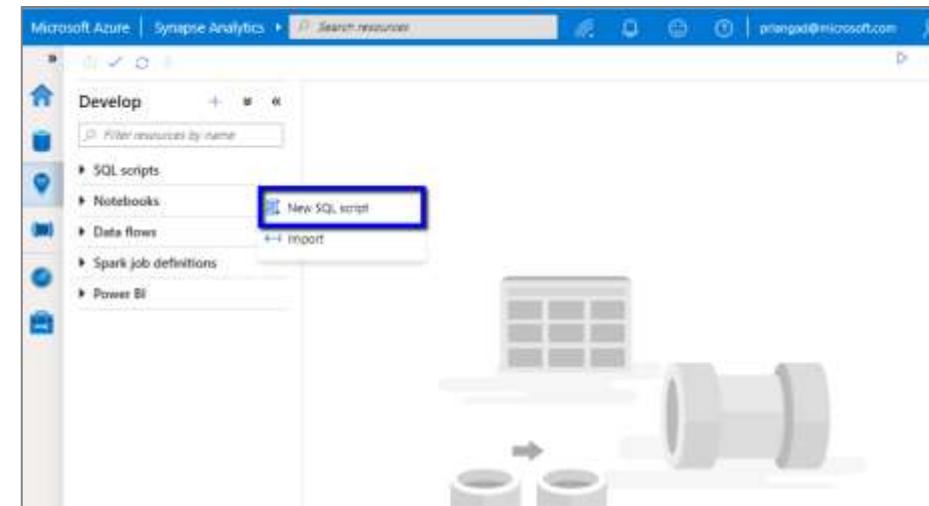
SQL Script

Authoring SQL Scripts

Execute SQL script on provisioned SQL Pool or SQL On-demand

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

Language support and intellisense



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 Studio interface. At the top, there's a toolbar with 'Run', 'Publish', 'Query plan', 'Connect to', 'SQL Analytics on-demand', 'Use database master', and a refresh button. Below the toolbar, a code editor window displays a T-SQL script:

```

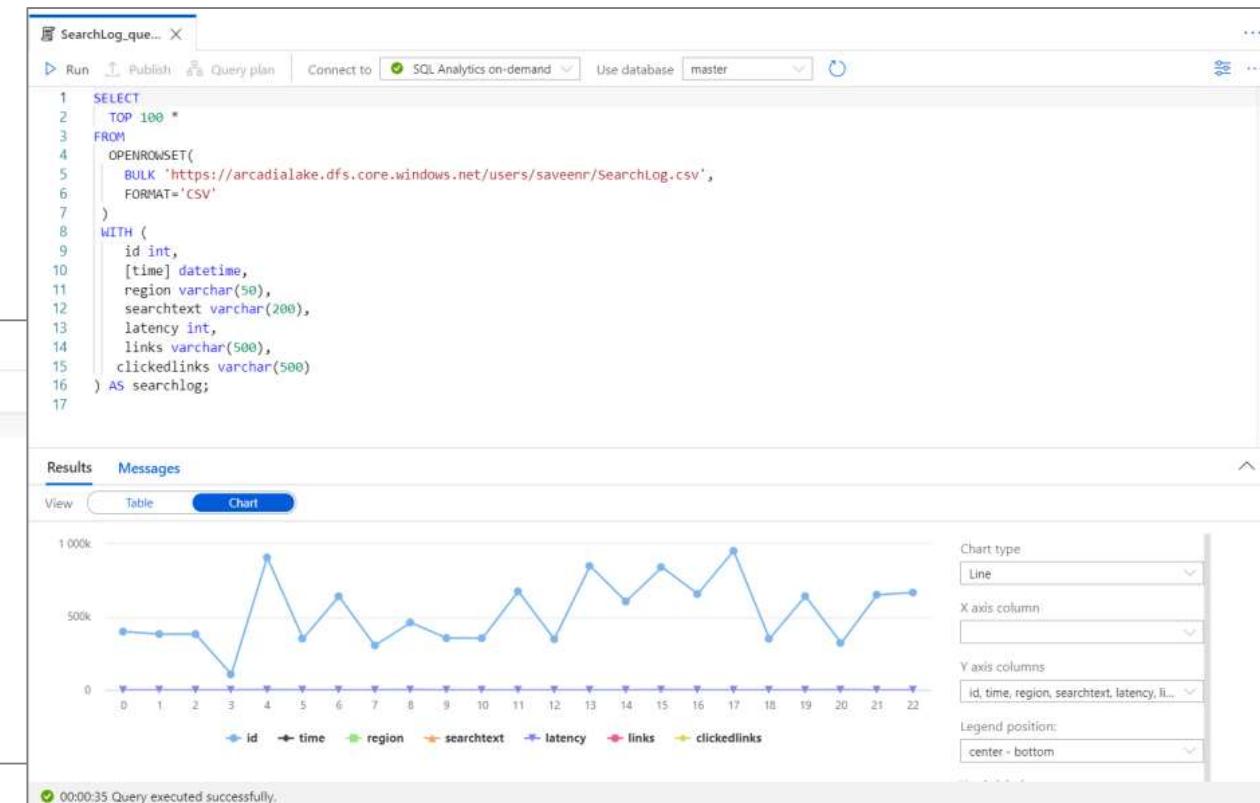
1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://arcadialake.dfs.core.windows.net/users/saveenr/SearchLog.csv',
6         FORMAT='CSV'
7     )
8     WITH (
9         id int,
10        [time] datetime,
11        region varchar(50),
12        searchtext varchar(200),
13       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

```

Below the code editor is a results pane with tabs for 'Results' (selected) and 'Messages'. It has buttons for 'Table' (selected), 'Chart', and 'Export results'. A red box highlights the 'Table' button and the 'Export results' dropdown. The 'Table' view shows a list of search log entries:

| 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 |

A red box highlights the 'Export results' dropdown, which lists 'CSV', 'Excel', 'JSON', and 'XML'.



Develop Hub - Notebooks

Notebooks

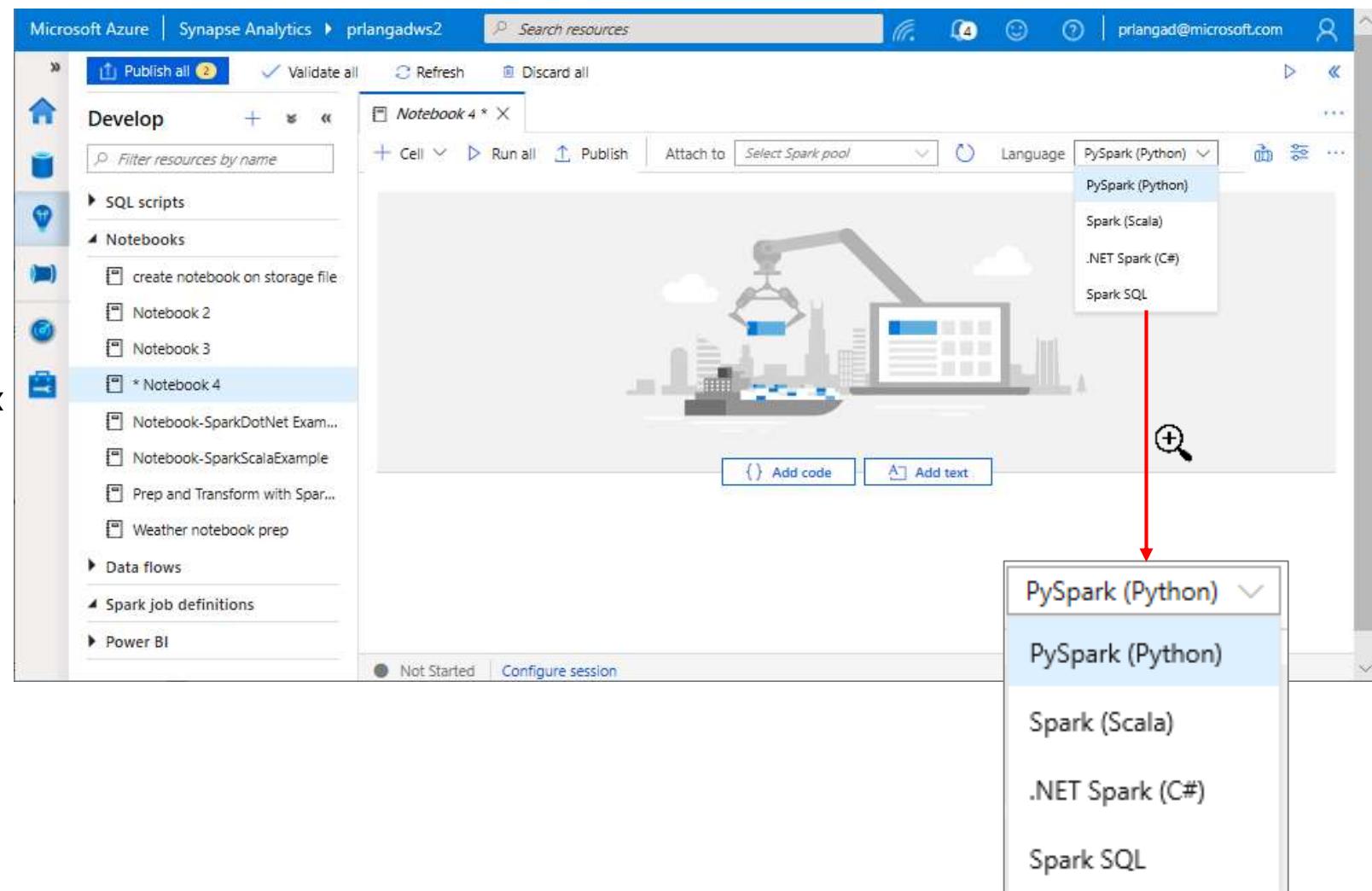
Allows to write multiple languages in one notebook

`%%<Name of language>`

Offers use of temporary tables across languages

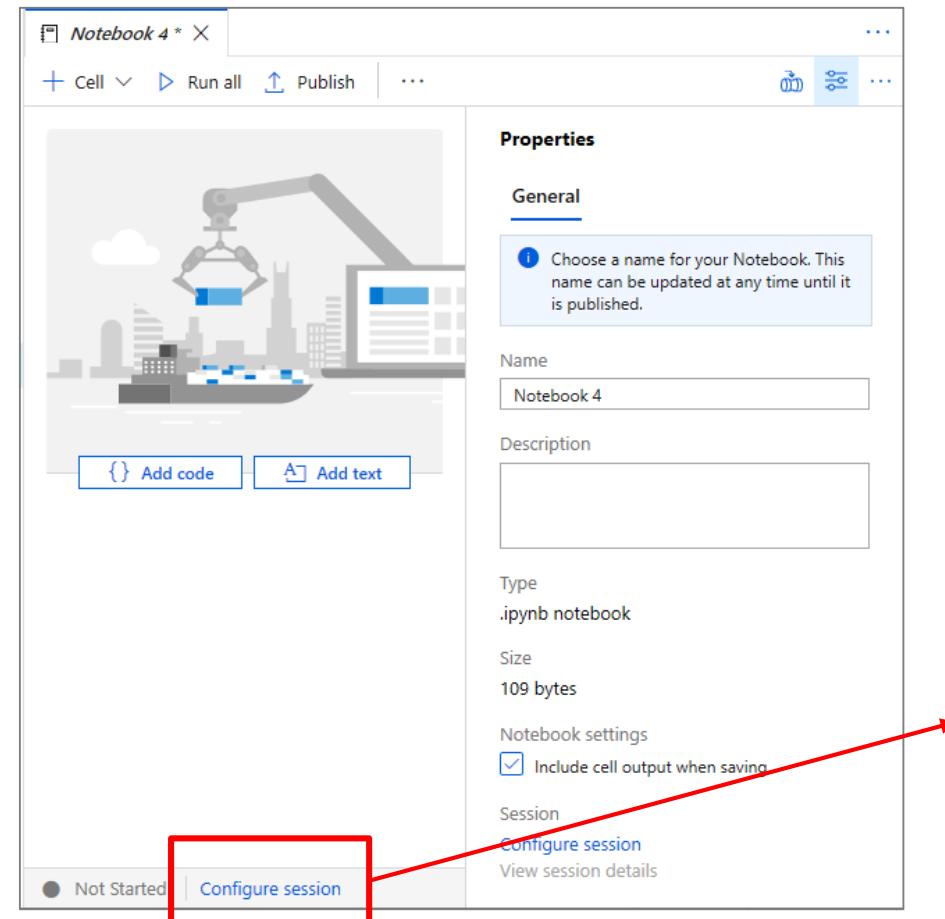
Language support for Syntax highlight, syntax error, syntax code completion, smart indent, code folding

Export results



Develop Hub - Notebooks

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



Configure session

Notebook 4

* Session timeout

* Executors

* Executor size

* Driver size

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 Azure Synapse Studio interface for developing notebooks. At the top, there's a toolbar with options like Cell, Run all, Publish, Attach to, Language (set to PySpark (Python)), and a session dropdown (spark1). Below the toolbar, a cell titled "Cell 1" contains the following PySpark code:

```

1 %%pyspark
2 data_path = spark.read.load('abfss://data@arcadialake.dfs.core.windows.net/nyctaxi/part-00010-tid-584931879')
3 data_path.show(10)

```

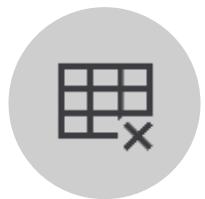
Below the code, a message indicates the command was executed in 4mins 33s 99ms by saveenr on 11-16-2019 09:36:31.944 -08:00. A section titled "Job execution Succeeded" shows three tasks (Job 0, Job 1, Job 2) completed on Spark 2 executors with 8 cores. Each task has a green checkmark and a duration of 14s, 21s, and 4s respectively. To the right, there are links to "View in monitoring" and "Spark history server".

At the bottom, a preview of the DataFrame is shown with columns: vendorID, tpepPickupDateTime, tpepDropoffDateTime, passengerCount, tripDistance, puLocationId, doLocationId, startLon, startLat, endLon, endLat, rateCodeId, and lpepDropoffDateTime. The data is presented as a table of rows, with the first few rows visible:

| | vendorID | tpepPickupDateTime | tpepDropoffDateTime | passengerCount | tripDistance | puLocationId | doLocationId | startLon | startLat | endLon | endLat | rateCodeId | lpepDropoffDateTime |
|---|----------|---------------------|---------------------|----------------|--------------|--------------|--------------|----------|----------|--------|--------|------------|---------------------|
| 1 | 2 | 2017-03-09 21:30:11 | 2017-03-09 21:44:20 | 1 | 14.0 | 0.5 | 0.5 | 4.06 | 148 | 48 | null | null | 2017-03-09 21:44:20 |
| 1 | 2 | 2017-03-09 21:47:00 | 2017-03-09 21:58:01 | 1 | 11.5 | 0.5 | 0.5 | 2.73 | 48 | 107 | null | null | 2017-03-09 21:58:01 |
| 1 | 2 | 2017-03-09 22:01:08 | 2017-03-09 22:11:16 | 1 | 10.0 | 0.5 | 0.5 | 2.27 | 79 | 162 | null | null | 2017-03-09 22:11:16 |
| 1 | 2 | 2017-03-09 22:16:05 | 2017-03-10 06:26:11 | 1 | 12.0 | 0.5 | 0.5 | 3.86 | 237 | 41 | null | null | 2017-03-10 06:26:11 |
| 1 | 2 | 2017-03-31 06:31:53 | 2017-03-31 06:41:48 | 1 | 19.5 | 0.5 | 0.5 | 3.45 | 41 | 162 | null | null | 2017-03-31 06:41:48 |
| 1 | 1 | 2017-03-01 00:00:00 | 2017-03-01 00:14:22 | 1 | 12.5 | 0.5 | 0.5 | 2.8 | 261 | 79 | null | null | 2017-03-01 00:14:22 |
| 1 | 1 | 2017-03-01 00:00:00 | 2017-03-01 00:19:30 | 1 | 19.5 | 0.5 | 0.5 | 6.0 | 87 | 142 | null | null | 2017-03-01 00:19:30 |
| 1 | 1 | 2017-03-01 00:00:00 | 2017-03-01 00:19:30 | 1 | 19.5 | 0.5 | 0.5 | 0.3 | 3.5 | 0.0 | 0.0 | 0.0 | 24.3 |

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

Dataflow Capabilities



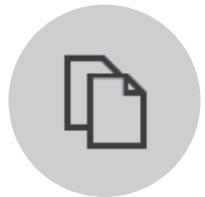
Handle upserts, updates, deletes on sql sinks



Add new partition methods



Add schema drift support



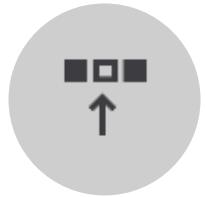
Add file handling (move files after read, write files to file names described in rows etc)



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)

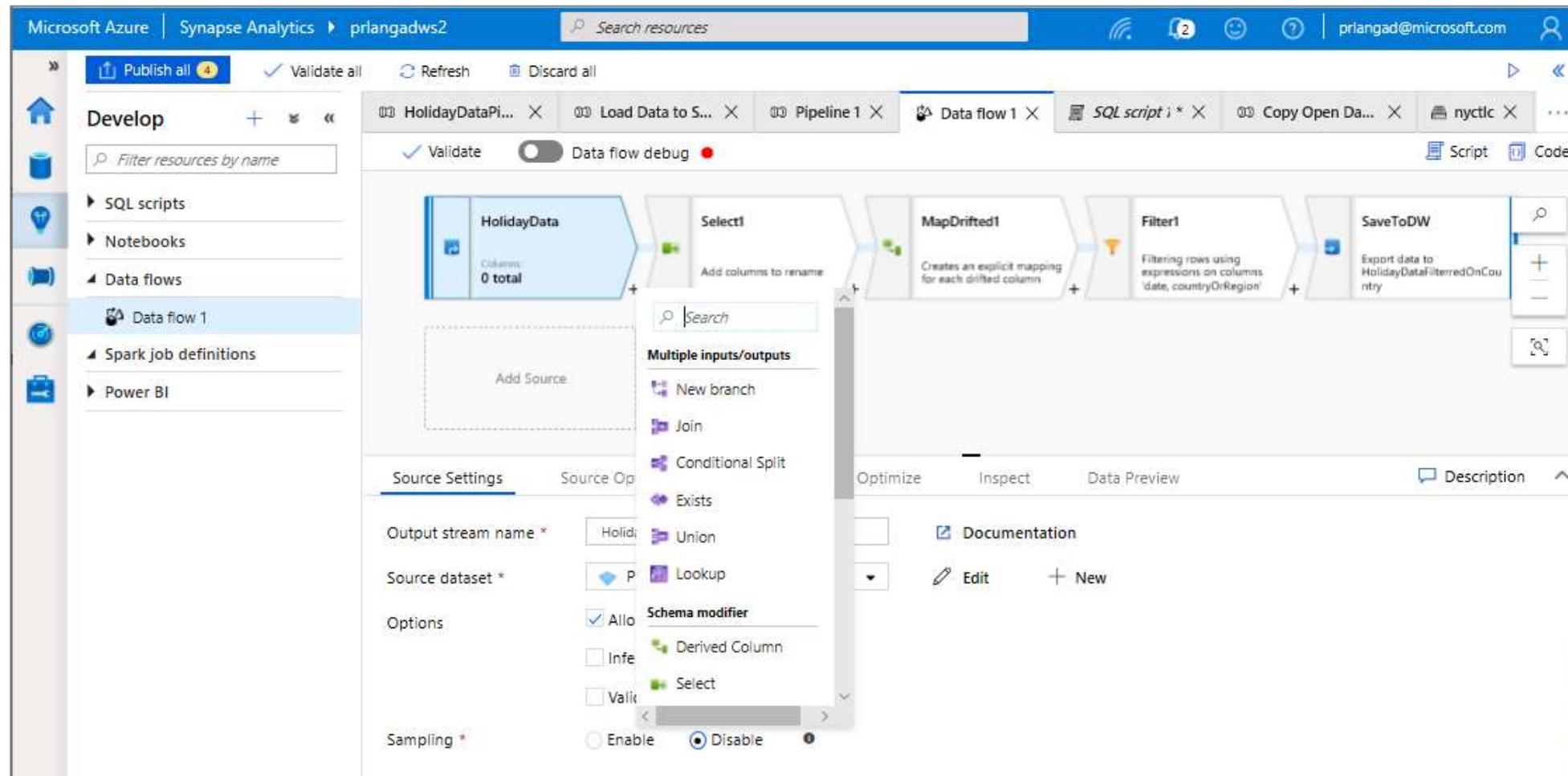


Implement commonly used ETL patterns as templates(SCD Type1, Type2, Data Vault)

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

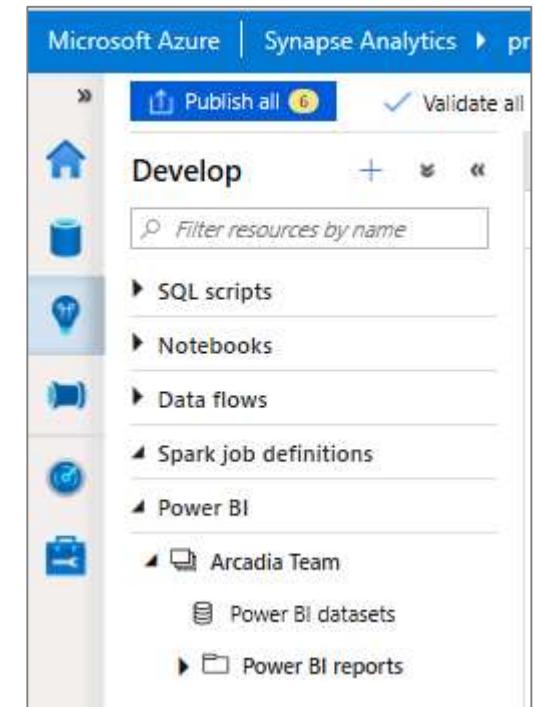
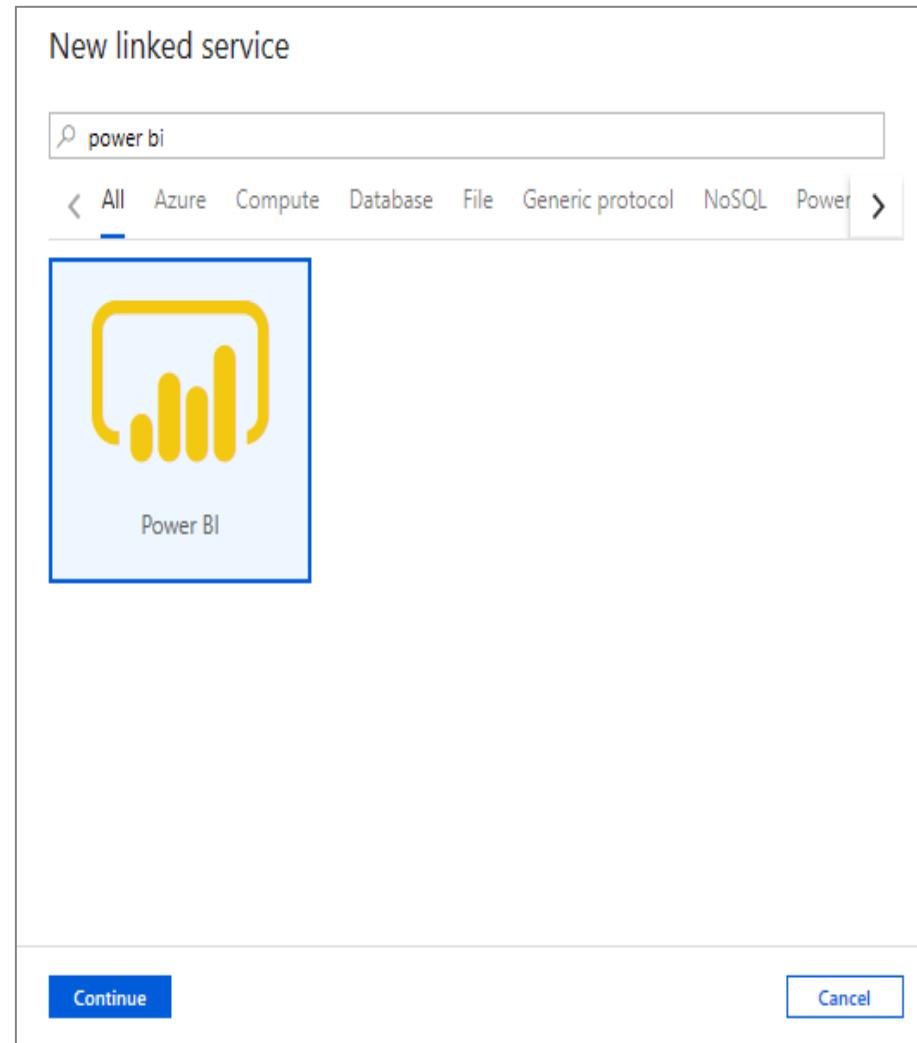
Overview

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 Azure Synapse Analytics Develop Hub interface for Power BI. On the left, the sidebar lists resources under 'Develop' including Notebooks, Data flows, Spark job definitions, and Power BI. Under Power BI, there are sections for 'SynapseNYTaxiInsights' (Power BI Datasets) and 'Power BI Reports'. A report titled 'SynapseNYIgnite2019' is currently selected, showing a line chart with three data series: GreenCab (green), YellowCab (yellow), and PredictedValues (blue). The chart tracks 'MarketShare' over time from Jan-2011 to Jun-2012. The top navigation bar includes 'Publish all' (with 2 pending), 'Validate all', 'Refresh', and 'Discard all'. The right side of the screen displays the Power BI visualization and field settings. The 'VISUALIZATIONS' pane shows various chart icons. The 'FIELDS' pane lists data fields such as dimHoliday, dimNYCLocations, Fhv, GreenCab, PredictedValues, wwFhvMarketShare, wwGrnCabMarketS..., wwMarketShareBy..., wwPredictedValues, wwYelCabMarketSh..., weather, YellowCab, and YellowCabTripsHoli... . The 'Filters' pane allows adding data fields for filters on this page and all pages. The 'DRILLTHROUGH' pane is set to 'Off' and 'Keep all filters' is turned off.

Develop Hub – Power BI

Edit reports in Synapse workspace

The screenshot shows the Azure Synapse Analytics Power BI Develop Hub interface. On the left, a sidebar lists resources: Notebooks (YellowCabExploration_sqlod, yellowcabprep, YellowCabPrepare, AMLautoMLPredict, AutoML, Data Download_Weather, * PrepareTaxiData), Data flows (PrepareCabDataFlow), Spark job definitions, Power BI (SynapseNYTaxiInsights, Power BI Datasets, Power BI Reports, SynapseNYIgnite2019, SynapseNYIgnite2019 (1)). The main area displays a report titled "SynapseNYIgnite2019". The report contains a line chart showing "Mondays Greenrides and Yellowrides by Date" from Jan-2014 to Jan-2015, overlaid with a bar chart of "numTrips". The "VISUALIZATIONS" pane on the right shows various chart and table icons. The "FIELDS" pane lists data fields like dimHoliday, dimNYCLocations, Fhv, GreenCab, PredictedValues, vwFhvMarketShare, vwGmCabMarketS..., vwMarketShareBy..., vwPredictedValues, vwYelCabMarketSh..., weather, YellowCab, and YellowCabTripsHoli... (with "numTrips" currently selected). The "Filters" pane shows filters for "holidayName" (is (All)) and "numTrips" (is (All)). The "Axis" section shows "holidayName" as the axis field. The "Legend" and "Value" sections also reference "numTrips". The "ToolTips" section has "Add data fields here". The "DRILLTHROUGH" section shows "date", "holidayName", and "Σ numTrips" (which is highlighted in yellow). The "Cross-report" section shows "year". The top navigation bar includes "Publish all", "Validate all", "Refresh", and "Discard all". The bottom navigation bar shows the current page as "Page 1" and a "+" button.

Develop Hub – Power BI

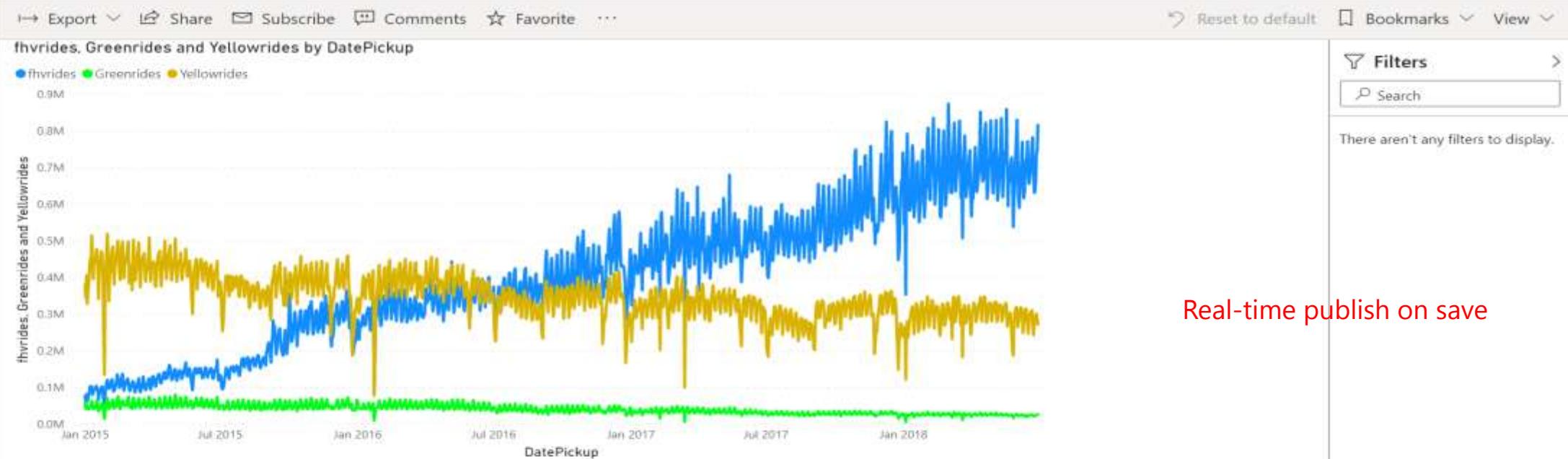
Publish edited reports in Synapse workspace to Power BI workspace

The screenshot shows the Azure Synapse Analytics Develop Hub interface. On the left, there's a sidebar with icons for Home, Datasets, Notebooks, Data Flows, Spark job definitions, Power BI, and SynapseNYTaxiInsights. Under Power BI, 'SynapseNYIgnite2019' is selected. The main area displays a Power BI report with a line chart and a bar chart. In the top left of the report view, there is a 'Save' button with the text 'Save this report'. A red box surrounds this button, and a red arrow points from it down to the text 'Publish changes by simple save report in workspace' located below the report preview. The right side of the screen shows the 'Visualizations' and 'Fields' panes, which contain various chart and field selection options.

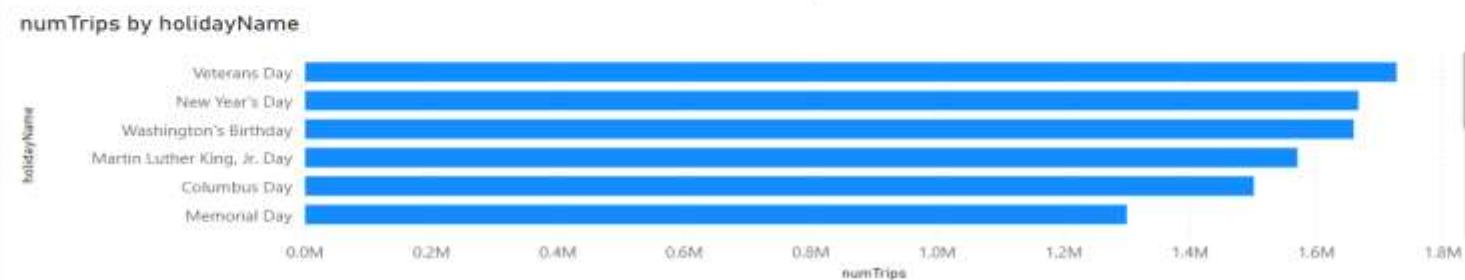
Save this report

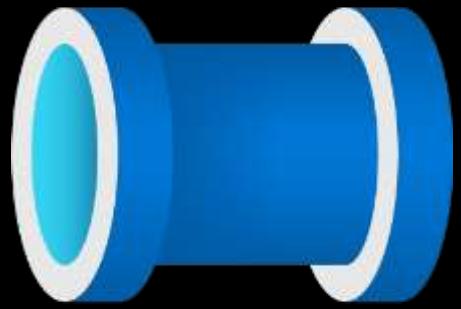
Publish changes by simple save report in workspace

- Home
- Favorites
- Recent
- Apps
- Shared with me
- Workspaces
- SynapseNYTaxiInsi...



Real-time publish on save



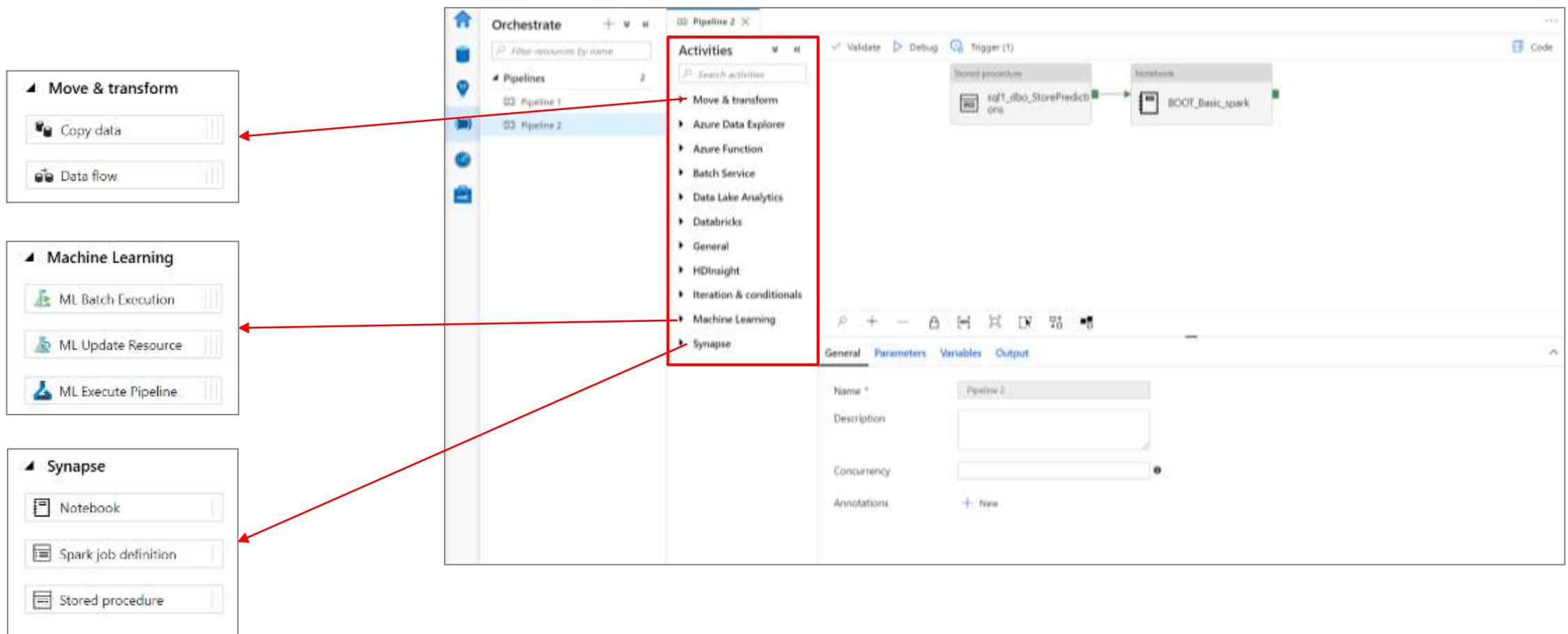


Synapse Studio Orchestrate hub

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.



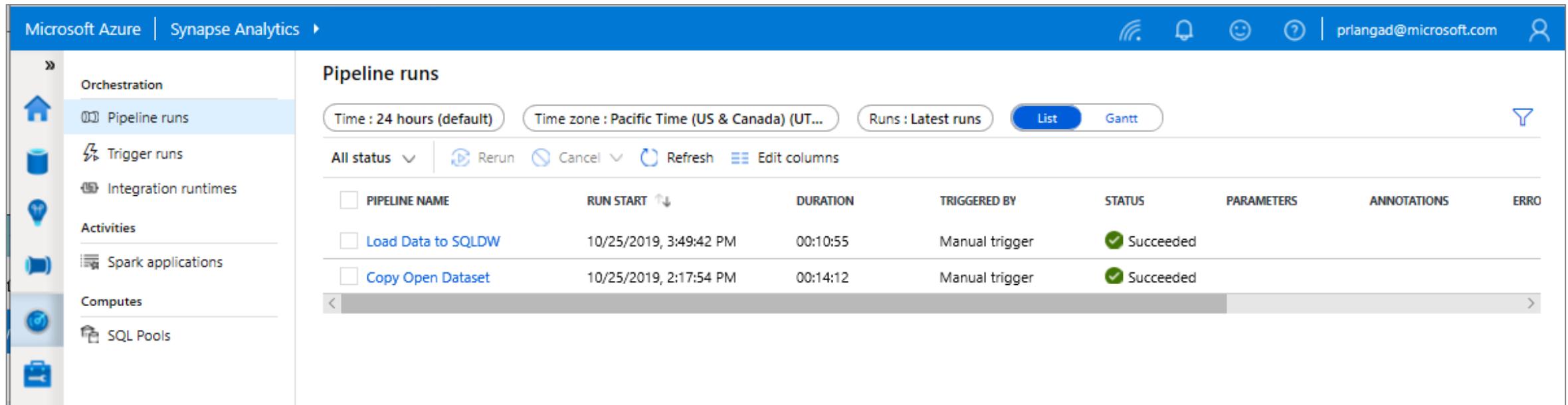


Synapse Studio **Monitor hub**

Monitor Hub

Overview

This feature provides ability to monitor orchestration, activities and compute resources.



The screenshot shows the Microsoft Azure Synapse Analytics Monitor Hub interface. The left sidebar has categories: Orchestration (selected), Pipeline runs, Trigger runs, Integration runtimes, Activities, Spark applications, Computes, and SQL Pools. The main area is titled "Pipeline runs" with filters: Time : 24 hours (default), Time zone : Pacific Time (US & Canada) (UT...), Runs : Latest runs, List (selected), Gantt, All status, Rerun, Cancel, Refresh, and Edit columns. It lists two pipeline runs:

| Pipeline Name | Run Start | Duration | Triggered By | Status |
|--------------------|------------------------|----------|----------------|-----------|
| Load Data to SQLDW | 10/25/2019, 3:49:42 PM | 00:10:55 | Manual trigger | Succeeded |
| Copy Open Dataset | 10/25/2019, 2:17:54 PM | 00:14:12 | Manual trigger | Succeeded |

Monitoring Hub - Orchestration

Overview

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

Benefits

Track all/specific pipelines

Monitor pipeline run and activity run details

Find the root cause of pipeline failure or activity failure

| 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 |
| Load Data to SQLDW | 10/25/2019, 3:49:42 PM | | 00:10:55 | Manual trigger | ✓ Succeeded |
| Copy Open Dataset | 10/25/2019, 2:17:54 PM | | 00:14:12 | Manual trigger | ✓ Succeeded |
| Pipeline 1 | 10/24/2019, 1:23:43 PM | | 00:00:08 | Manual trigger | ✓ Succeeded |

Monitoring Hub - Spark applications

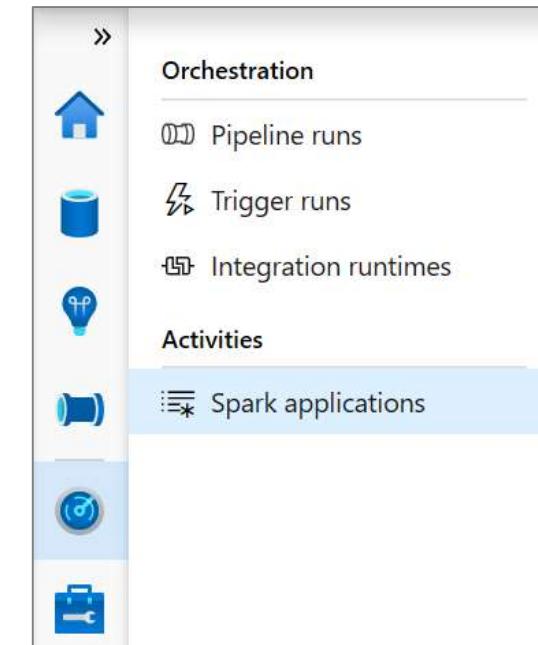
Overview

Monitor Spark pools, Spark applications for the progress and status of activities

Benefits

Monitor Spark pools for the status as paused, active, resume, scaling and upgrading

Track the usage of resources



| Spark applications | | | | | |
|--|------------------------|--|--------------|--------------------|---------------|
| Submit time : 24 hours (default) (10/30/2019 9:52 AM - 10/31/2019 9:52 AM) | | Time zone : Pacific Time (US & Canada) (UT...) | | List | Chart |
| All types | Cancel | Refresh | Edit columns | | |
| APPLICATION NAME | SUBMITTER | SUBMIT TIME | STATUS | POOL | TYPE |
| Synapse_prlang-syntax... | prlangad@microsoft.com | 10/30/2019 1:21 PM | Cancelled | prlang-syntaxcheck | Spark session |
| Synapse_prlSpark_1572... | prlangad@microsoft.com | 10/30/2019 1:06 PM | Cancelled | prlSpark | Spark session |



Synapse Studio Manage hub

Manage Hub

Overview

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

The screenshot shows the Microsoft Azure Synapse Analytics Studio Manage Hub interface. The left sidebar contains navigation links: External connections, **Linked services** (which is selected and highlighted in blue), Orchestration, Triggers, Integration runtimes, Security, and Access control. The top bar includes buttons for Publish all, Validate all, Refresh, Discard all, and user information (prlangad@microsoft.com). The main content area is titled "Linked services" and contains a description: "Linked services are much like connection strings, which define the connection information needed for Arcadia to connect to external resources." Below this is a table with columns: NAME, TYPE, and ANNOTATIONS. The table lists ten linked services:

| NAME | TYPE | ANNOTATIONS |
|-----------------------------|---|-------------|
| ADLSG2OpenDataSetSink | Azure Data Lake Storage Gen2 | |
| AzureBlobStorage1 | Azure Blob Storage | |
| AzureDataLakeStorage1 | Azure Data Lake Storage Gen2 | |
| AzureDataLakeStorage2Source | Azure Data Lake Storage Gen2 | |
| AzureOpenDataset | Azure Blob Storage | |
| AzureOpenDataSet2 | Azure Blob Storage | |
| AzureSqlDW1 | Azure Synapse Analytics (formerly SQL DW) | |
| AzureSynapseAnalytics1 | Azure Synapse Analytics (formerly SQL DW) | |
| AzureSynapseAnalytics2 | Azure Synapse Analytics (formerly SQL DW) | |
| PowerBIWorkspace1 | Power BI | |

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 'Linked services' blade in the Microsoft Azure Synapse Analytics portal. On the left, a navigation menu includes 'External connections' (selected), 'Linked services' (highlighted with a red box), 'Orchestration', 'Triggers', 'Integration runtimes', 'Security', and 'Access control'. A 'New' button is also highlighted with a red box. The main area displays a list of existing linked services with columns for NAME, TYPE, and ANNOTATIONS. Below this is a grid of connector icons labeled 'New linked service'. A red arrow points from the 'New' button to the first icon in the grid, which is 'Power BI'.

| NAME | TYPE | ANNOTATIONS |
|-----------------------------|------------------------------|-------------|
| ADLSG2OpenDataSetSink | Azure Data Lake Storage Gen2 | |
| AzureBlobStorage1 | Azure Blob Storage | |
| AzureDataLakeStorage1 | Azure Data Lake Storage Gen2 | |
| AzureDataLakeStorage2Source | | |
| AzureOpenDataset | | |
| AzureOpenDataSet2 | | |
| AzureSqlDW1 | | |

| New linked service | | |
|------------------------|------------------|----------------------|
| PayPal (Preview) | Phoenix | PostgreSQL |
| Power BI | Presto (Preview) | QuickBooks (Preview) |
| REST | SAP BW Open Hub | SAP BW via MDX |
| SAP Cloud for Customer | SAP ECC | SAP HANA |
| SAP | SAP | SAP |

Manage – Access Control

Overview

It provides access control management to workspace resources and artifacts for admin and users

Benefits

Share workspace with the team

Increases productivity

Manage permissions on code artifacts and Spark pools

Microsoft Azure | Synapse Analytics > prlangad

Access control

Manage access to Arcadia workspace resources & artifacts for admin & users. [Learn more](#)

Add admin Refresh Remove

Showing 1 of 1 items

| NAME | PERMISSIONS |
|--|---|
| Priyanka Langade Priyanka.Langade@microsoft.com | Full permissions on code artifacts and Spark pools. Learn more about SQ |

Add admin

An admin has full control over code artifacts, can attach to Spark pools, and can schedule pipelines. Permissions to Storage accounts and SQL pool databases are managed on the resources directly. [Learn more](#)

* Select user

Search by name or email address

Selected individual, groups or apps
No individual, groups, or apps selected.

Apply **Cancel**

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 Azure Synapse Analytics portal with the 'Triggers' section selected. A red box highlights the '+ New' button in the center of the trigger list. A larger red box encloses the 'New trigger' configuration dialog box on the right side of the screen. This dialog box contains fields for Name (Trigger 2), Description, Type (Schedule selected), Start Date (UTC) (10/29/2019 9:46 PM), Recurrence (Every 1 Minute(s)), End (No End selected), Annotations, and Activated (No selected). The main list below shows two triggers: 'CopyParquetDataTrigger' (Started, Schedule) and 'Trigger 1' (Stopped, Schedule).

| NAME | TYPE | STATUS | NUMBER OF PIPELINES | ANNOTATIONS |
|--------------------------|----------|---------|---------------------|-------------|
| * CopyParquetDataTrigger | Schedule | Started | 1 | |
| * Trigger 1 | Schedule | Stopped | 0 | |

Manage – Integration runtimes

Overview

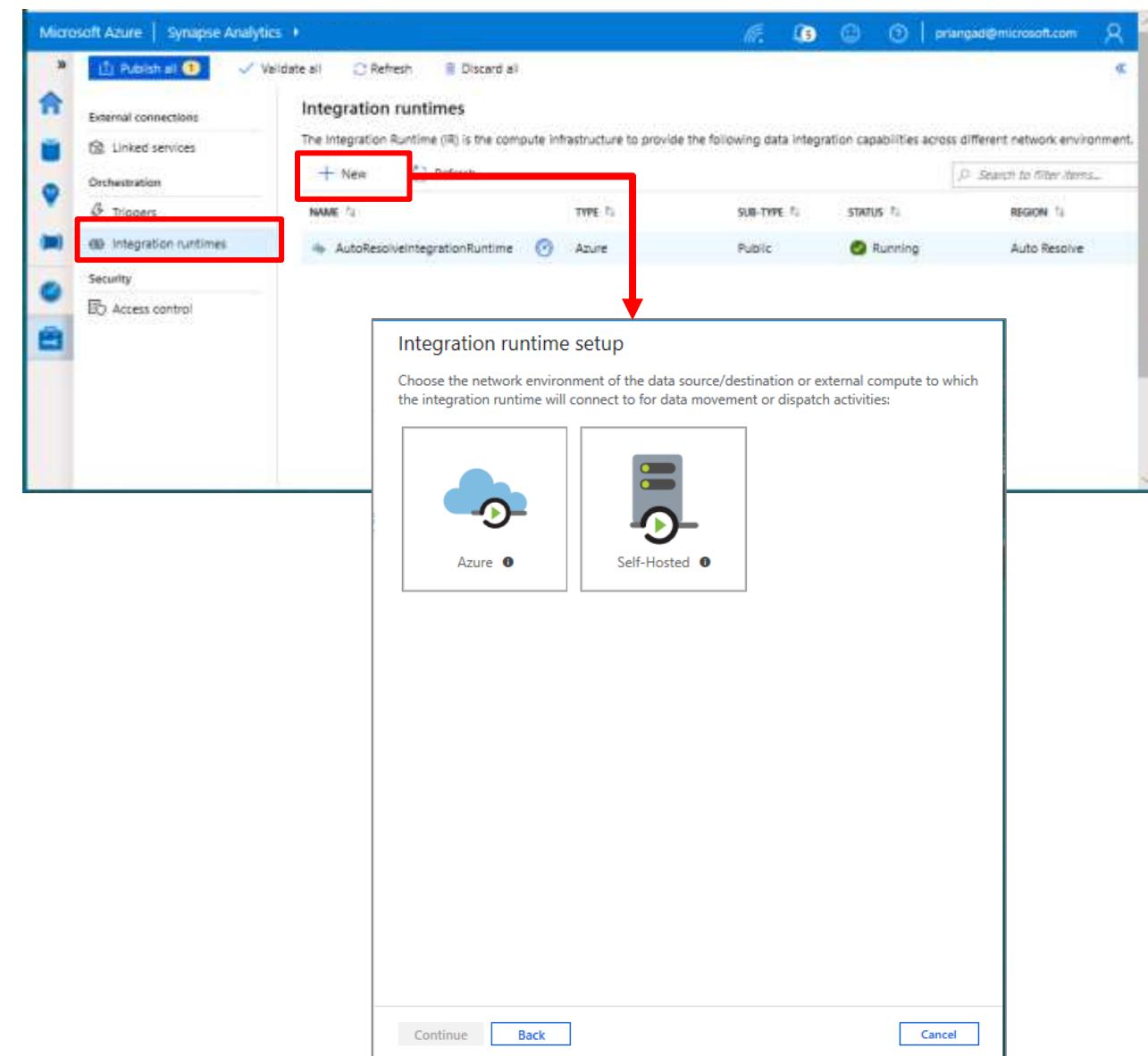
Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

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



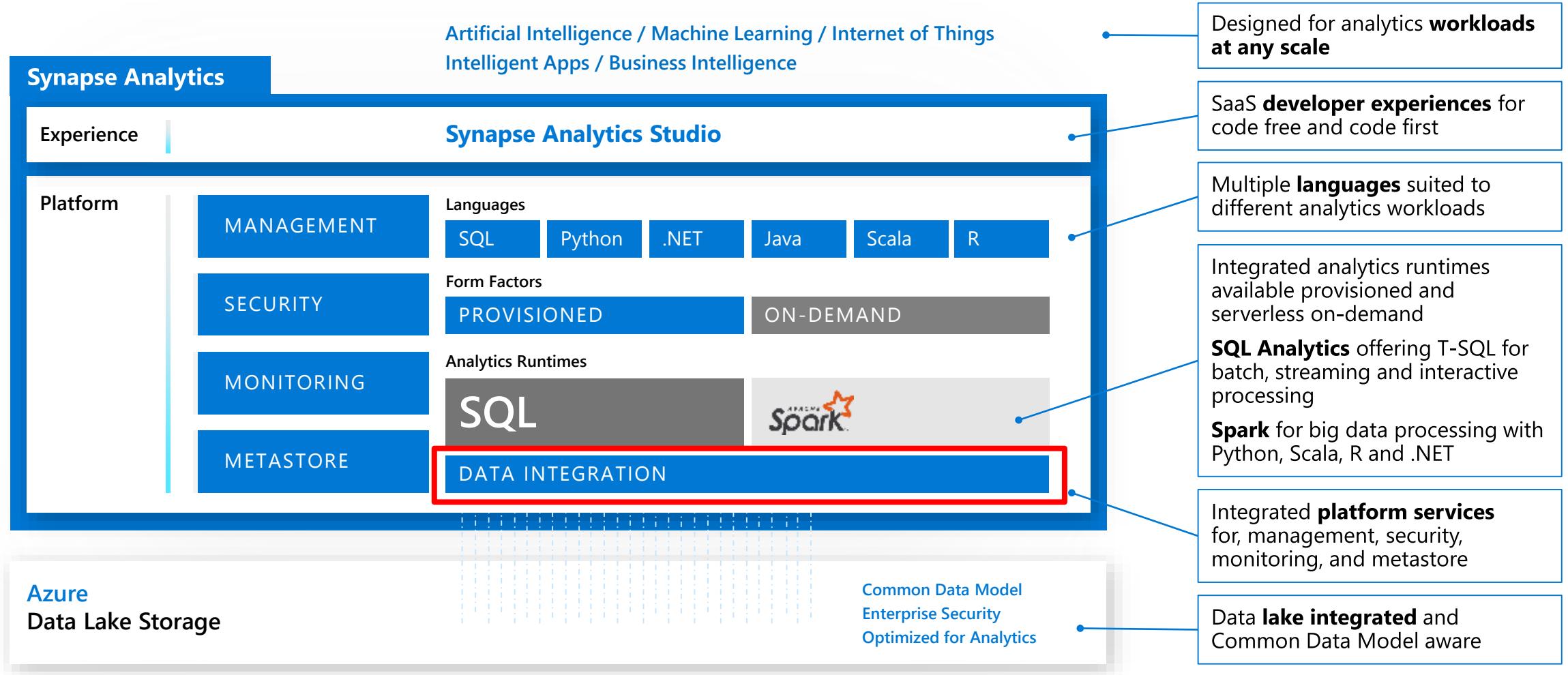


Azure Synapse Analytics

Data Integration

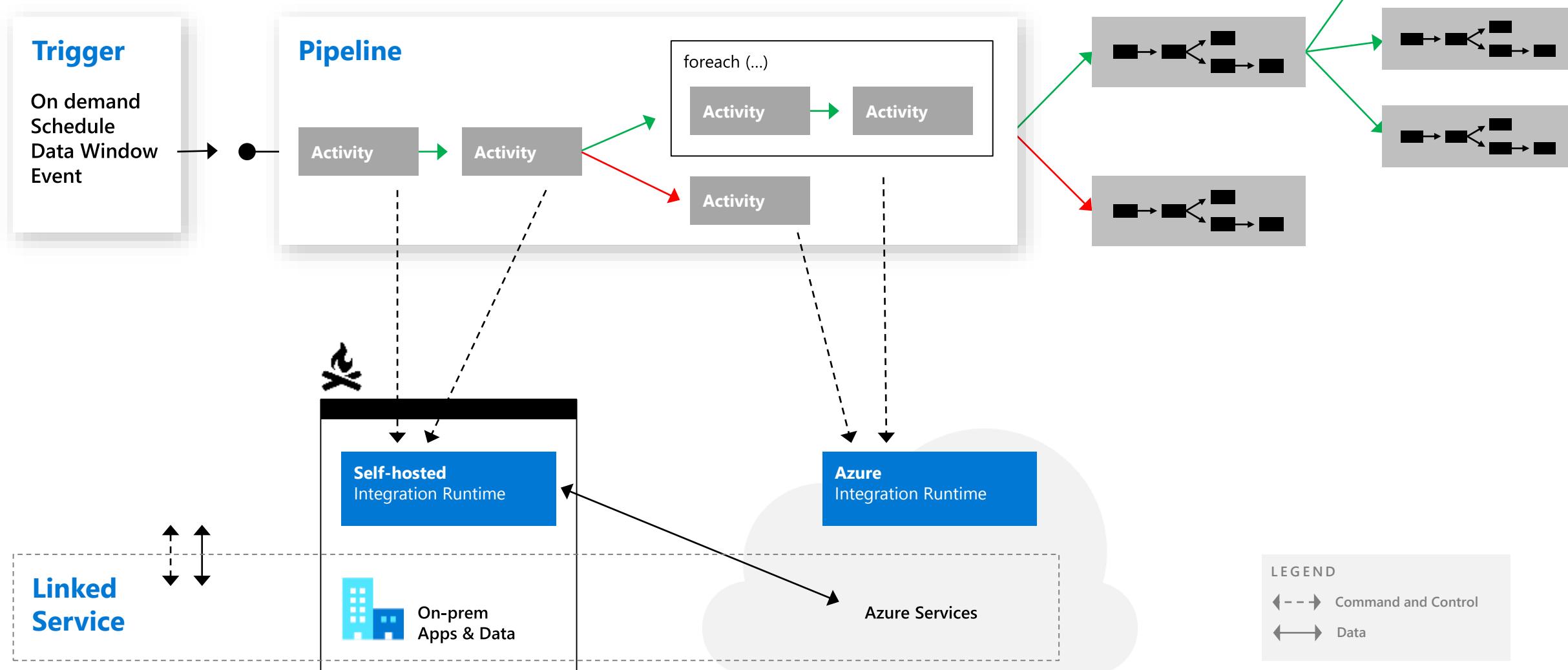
Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Data Integration = Separate version Azure Data Factory (ADF). Will have 1-click migration

Orchestration @ Scale



Data Movement

Scalable

per job elasticity

Up to 4 GB/s

Simple

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

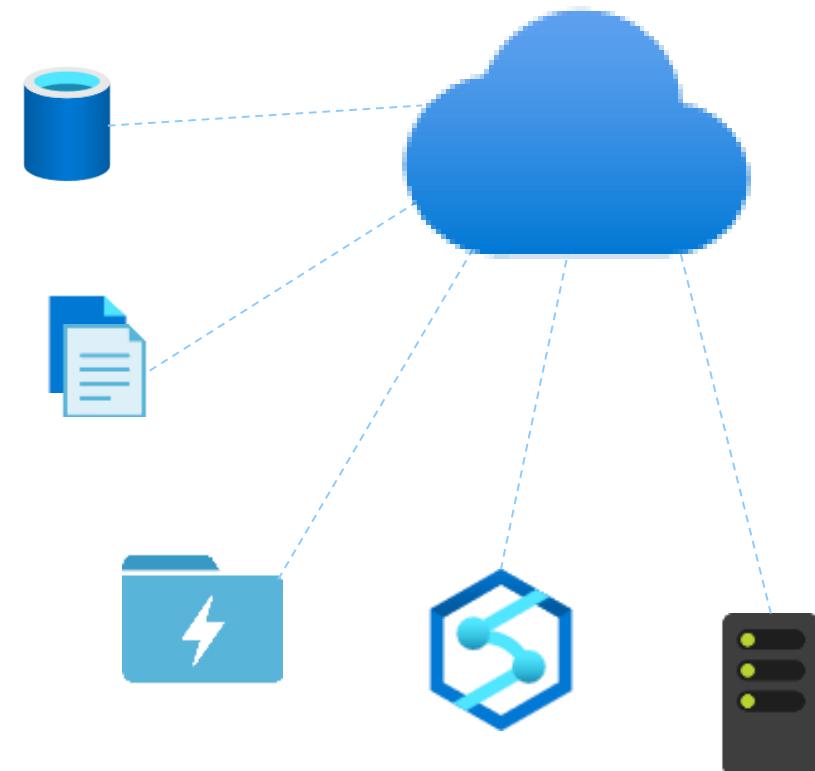
Serverless, no infrastructure to manage

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

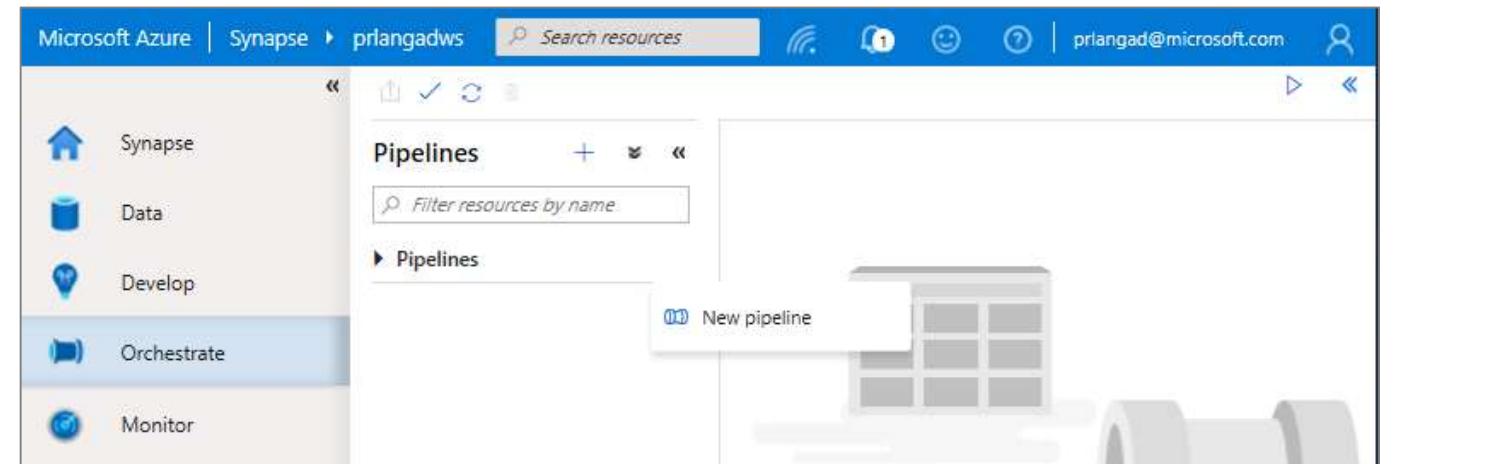
It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

Graphical development experience

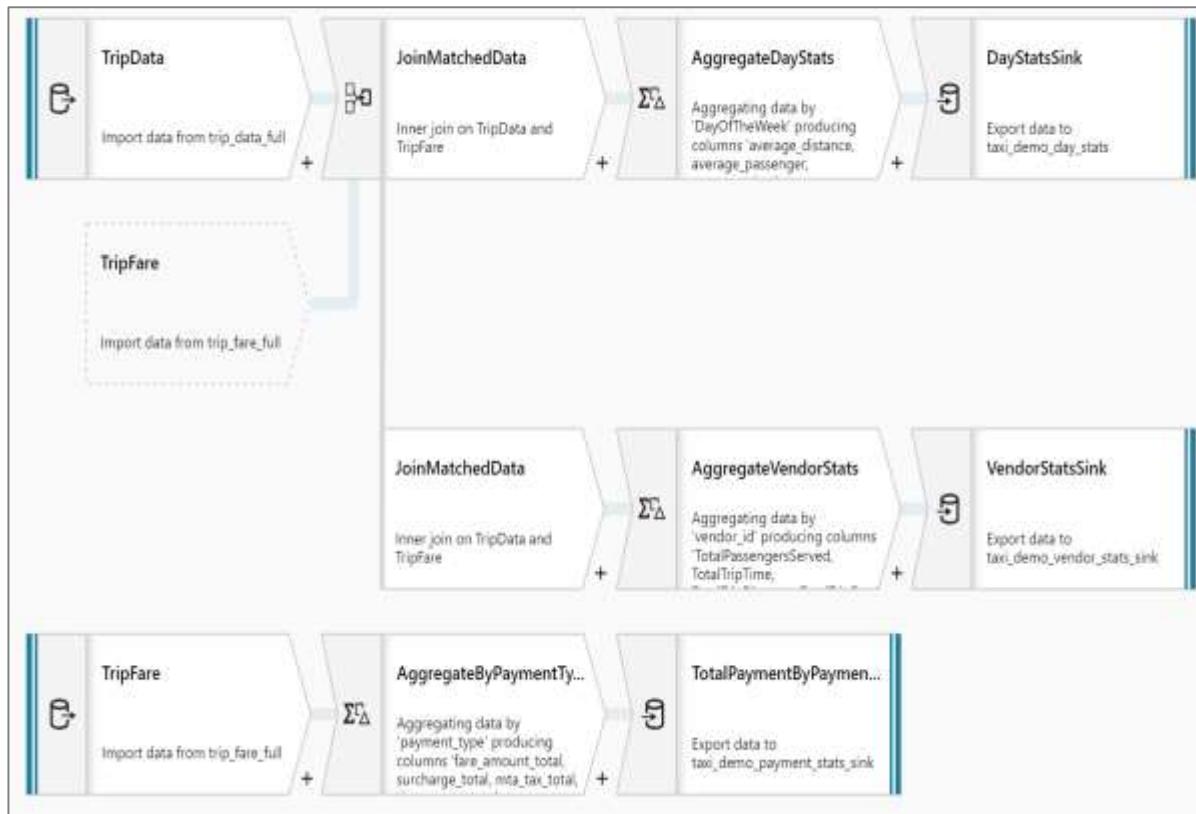


| Azure Cosmos DB (SQL API) | Azure Data Explorer (Kusto) | Azure Data Lake Storage Gen1 |
|-------------------------------------|---|------------------------------|
| | | |
| Azure Data Lake Storage Gen2 | Azure Database for MariaDB | Azure Database for MySQL |
| | | |
| Azure Database for PostgreSQL | Azure File Storage | Azure SQL Database |
| | | |
| Azure SQL Database Managed Instance | Azure Synapse Analytics (formerly SQL DW) | Azure Table Storage |
| | | |

Prep & Transform Data

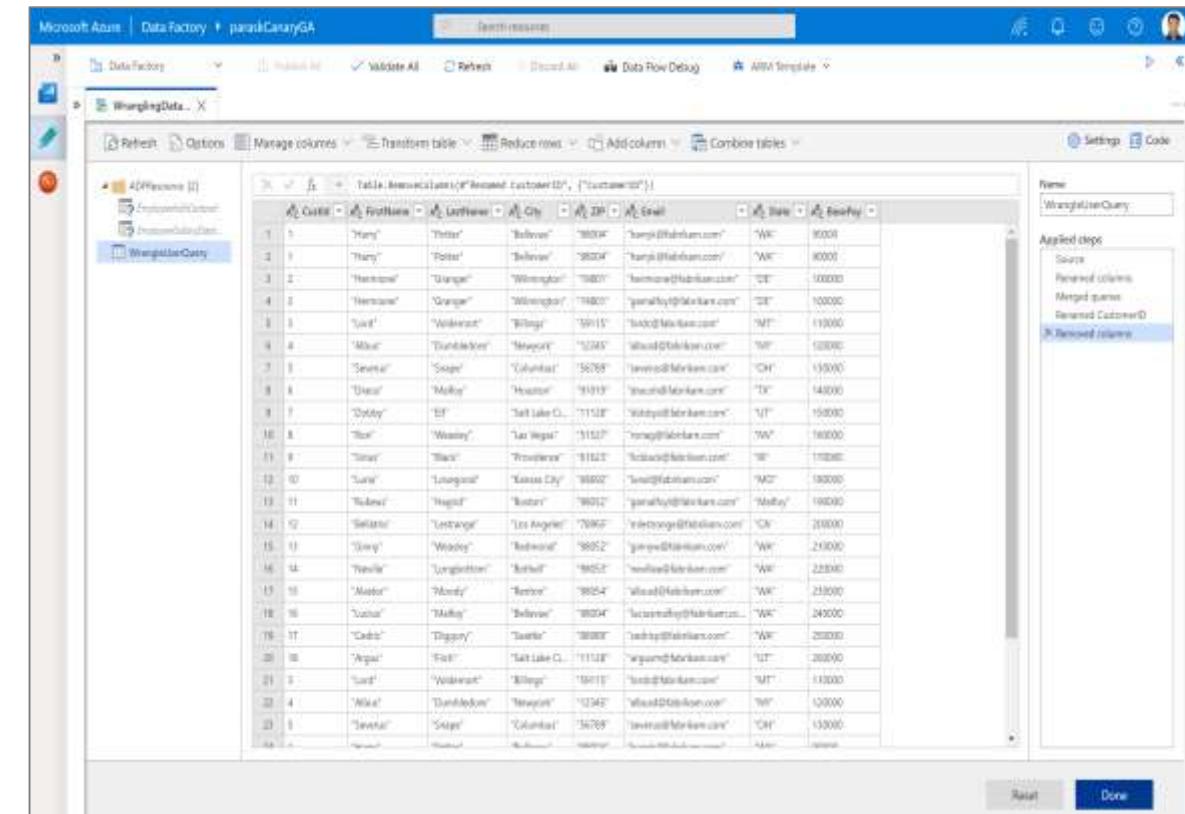
Mapping Dataflow

Code free data transformation @scale



Wrangling Dataflow

Code free data preparation @scale



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 icons for External connections, Linked services, Orchestration, Triggers (which is selected and highlighted in blue), Integration runtimes, Security, and Access control. Above the sidebar, there are buttons for Publish all (with a count of 4), Validate all, Refresh, and Discard all. To the right of the sidebar, there's a large panel titled 'Triggers'. It contains a brief description: 'To execute a pipeline set the trigger. Triggers represent a unit of processing that determines when a pipeline runs.' Below this is a '+ New' button. A table lists existing triggers:

| NAME | TYPE | STATUS |
|--------------------------|----------|---------|
| * CopyParquetDataTrigger | Schedule | Started |
| * Trigger 1 | Schedule | Stopped |

 At the bottom of the 'Triggers' panel is a 'OK' button. Overlaid on the bottom right of the main area is a 'New trigger' dialog box. It has fields for 'Name *' (set to 'Trigger 1'), 'Description' (empty), 'Type *' (set to 'Schedule'), 'Start Date (UTC) *' (set to '10/30/2019 11:20 PM'), 'Recurrence *' (set to 'Every 1 Minute(s)'), 'End *' (set to 'No End'), and 'Annotations' (with a '+ New' link). There's also an 'Activated *' section with 'Yes' selected. At the bottom of the dialog is an 'OK' button.

Manage – Linked Services

Overview

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

Benefits

Offers pre-build 85+ connectors

Easy cross platform data migration

Represents data store or compute resources

NOTE: Linked Services are all for Data Integration except for Power BI (eventually ADC, Databricks)

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, a sidebar menu includes 'External connections', 'Linked services' (which is selected and highlighted in blue), 'Orchestration', 'Triggers', 'Integration runtimes', 'Security', and 'Access control'. The main area is titled 'Linked services' and contains a sub-header: 'Linked services are much like connection strings, which define the connection information needed for Arcadia to connect to external resources.' Below this is a table with columns 'NAME', 'TYPE', and 'ANNOTATIONS'. A blue arrow points from the 'New' button in the sidebar to the 'New linked service' section of the modal dialog. The modal dialog is titled 'New linked service (Power BI)'. It has a note: 'Choose a name for your linked service. This name cannot be updated later.' A 'Name' field is filled with 'PowerBIWorkspace1'. There are 'Description' and 'Workspace name' fields, both currently empty. At the bottom of the dialog are 'Continue' and 'Cancel' buttons.

Manage – Integration runtimes

Overview

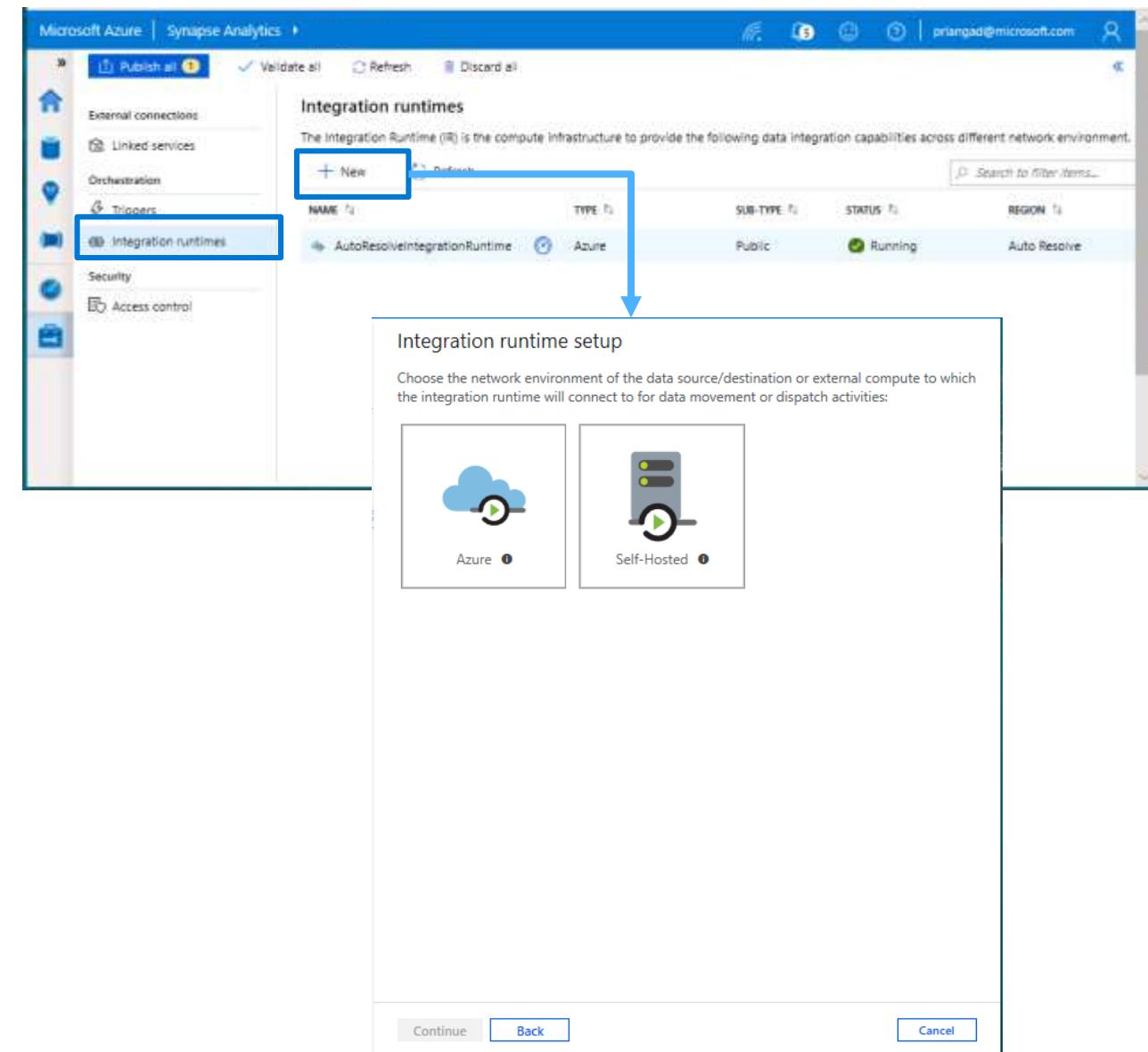
It is the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked Services.

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



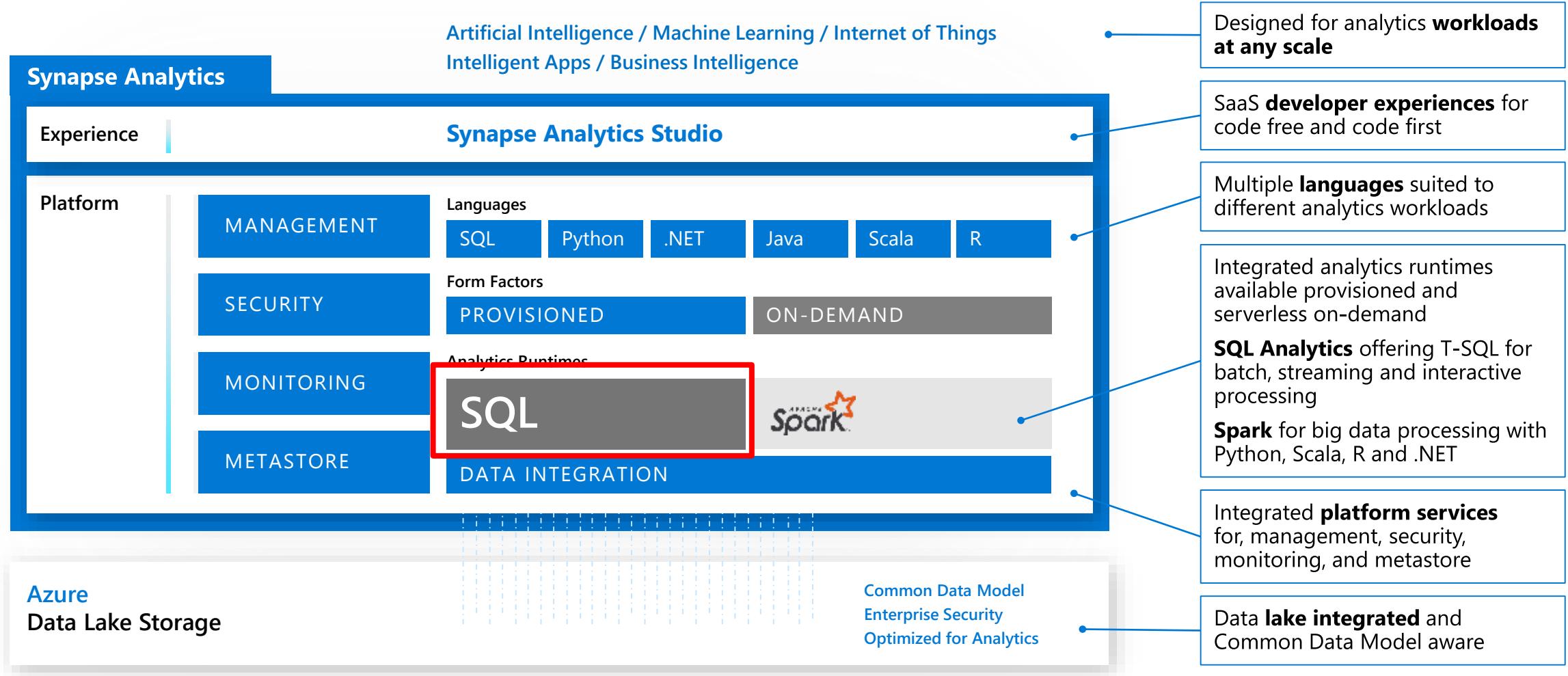


Azure Synapse Analytics

SQL Analytics

Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



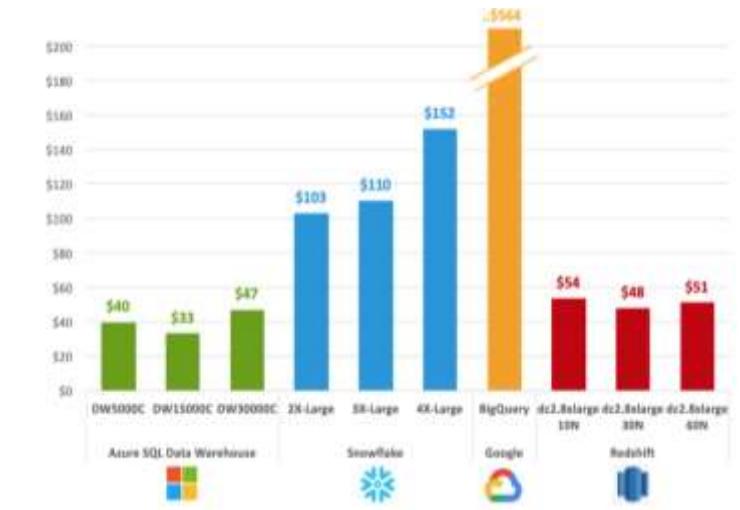
Platform: Performance

Overview

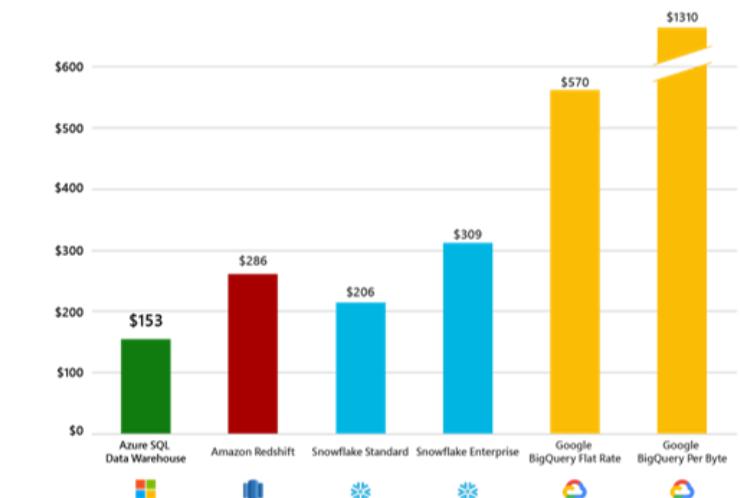
SQL Data Warehouse's industry leading price-performance comes from leveraging the Azure ecosystem and core SQL Server engine improvements to produce massive gains in performance.

These benefits require no customer configuration and are provided out-of-the-box for every data warehouse

- **Gen2 adaptive caching** – using non-volatile memory solid-state drives (NVMe) to increase the I/O bandwidth available to queries.
- **Azure FPGA-accelerated networking enhancements** – to move data at rates of up to 1GB/sec per node to improve queries
- **Instant data movement** – leverages multi-core parallelism in underlying SQL Servers to move data efficiently between compute nodes.
- **Query Optimization** – ongoing investments in distributed query optimization



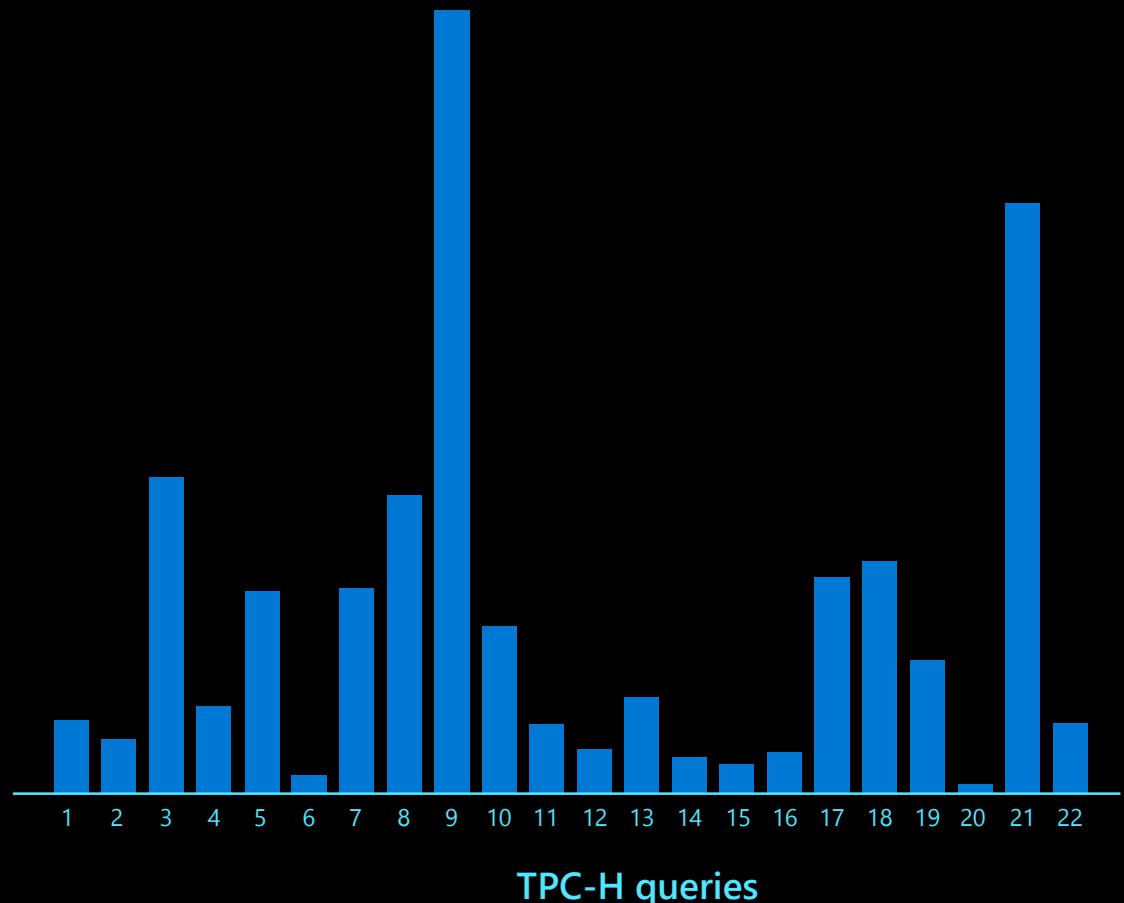
TPC-H 30TB Cloud DW Benchmark



TPC-DS 30TB Cloud DW Benchmark

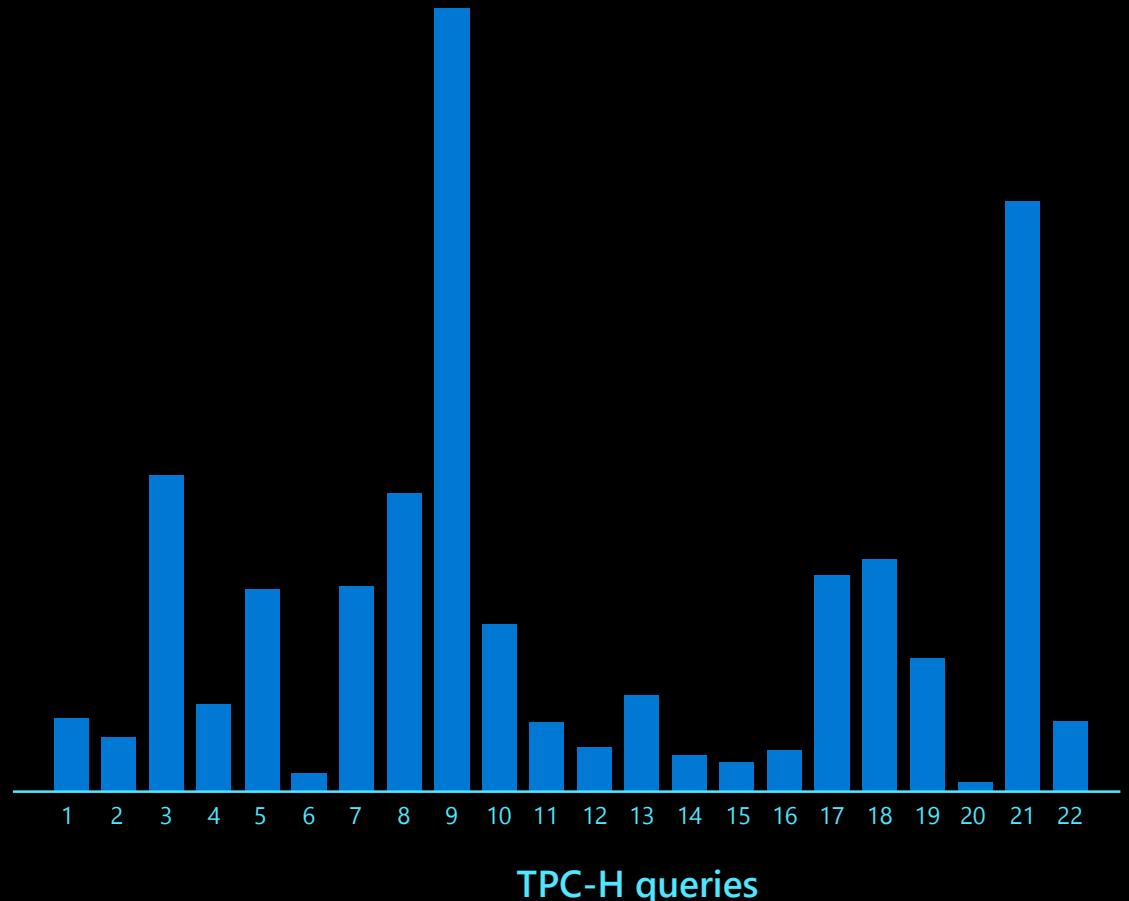
The first and only
analytics system to have
run all TPC-H queries
at petabyte-scale

TPC-H 1 Petabyte query times



Azure Synapse is the first
and only analytics
system to have run all
TPC-H queries at 1
petabyte-scale

TPC-H 1 Petabyte Query Execution



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

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 | LastName | VacHours | FewestVacHours |
|--------------------------------|--------------|----------|----------------|
| 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

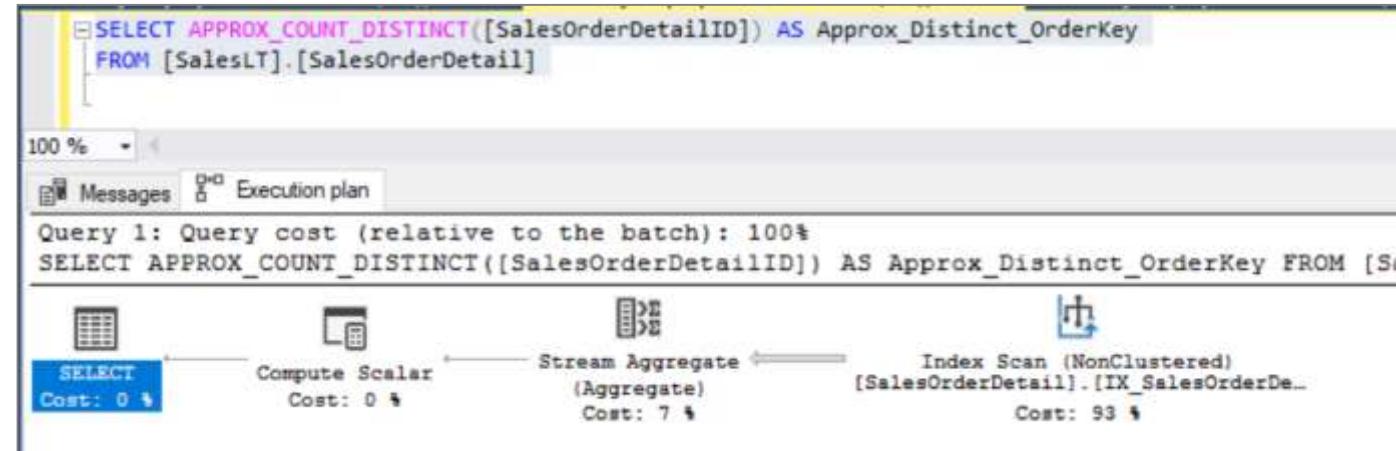
APPROX_COUNT_DISTINCT

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

100 %

Results Messages

| Approx_Distinct_OrderKey |
|--------------------------|
| 540 |



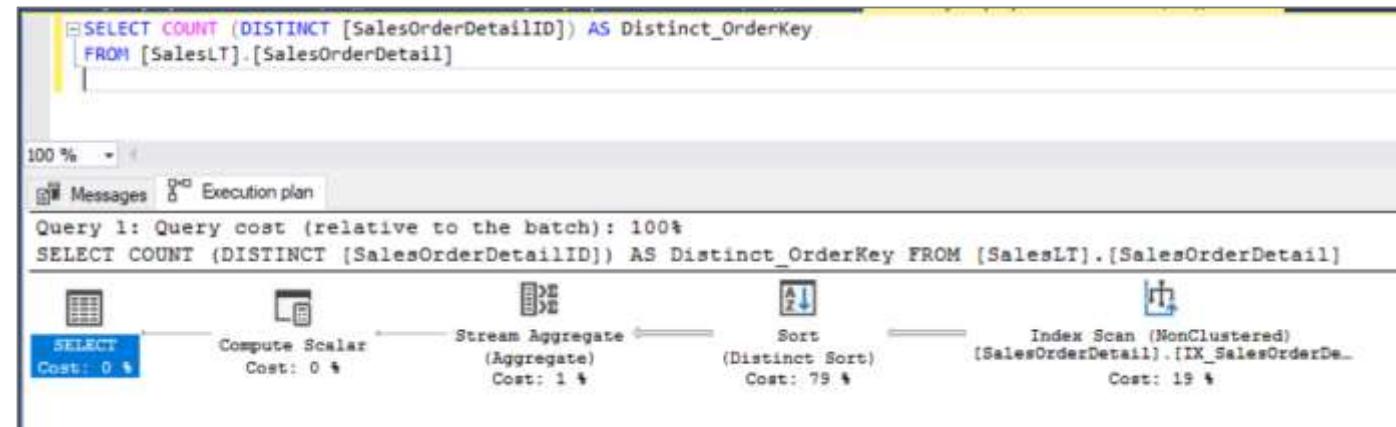
COUNT DISTINCT

```
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.

Isolation level

- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE
- READ UNCOMMITTED

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.

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

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

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 runtime language (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;
```



Azure Synapse Analytics

Data Storage and Performance Optimizations

Database Tables

Optimized Storage

Reduce Migration Risk

Less Data Scanned

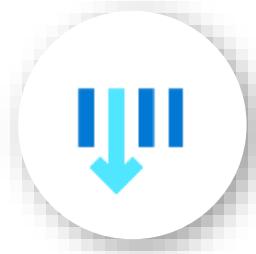
Smaller Cache Required

Smaller Clusters

Faster Queries



Columnar Storage



Columnar Ordering



Table Partitioning



Hash Distribution



Nonclustered Indexes

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)
);

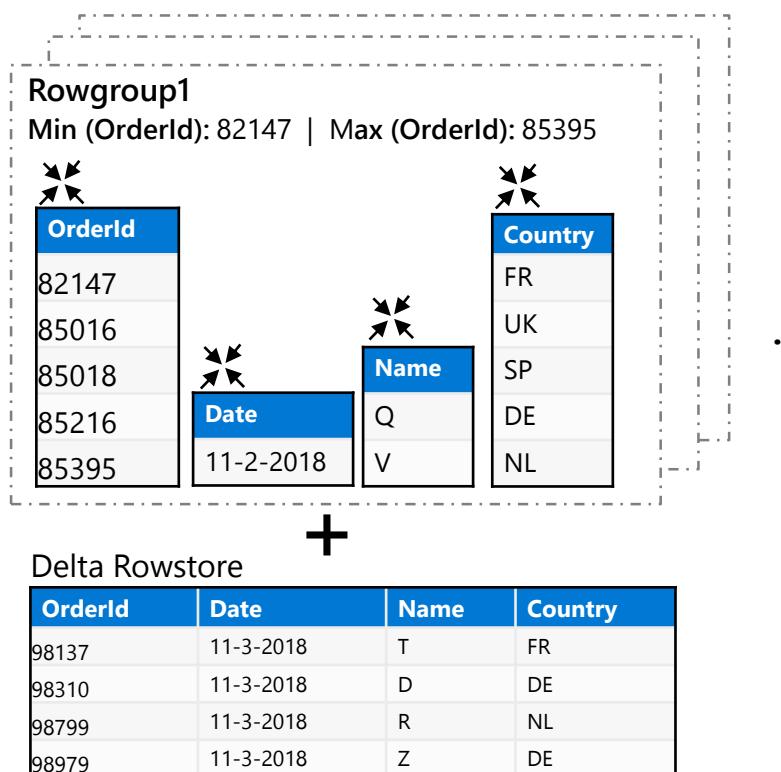
-- Add non-clustered index to table
CREATE INDEX NameIndex ON orderTable (Name);
```

SQL Analytics Columnstore Tables

Logical table structure

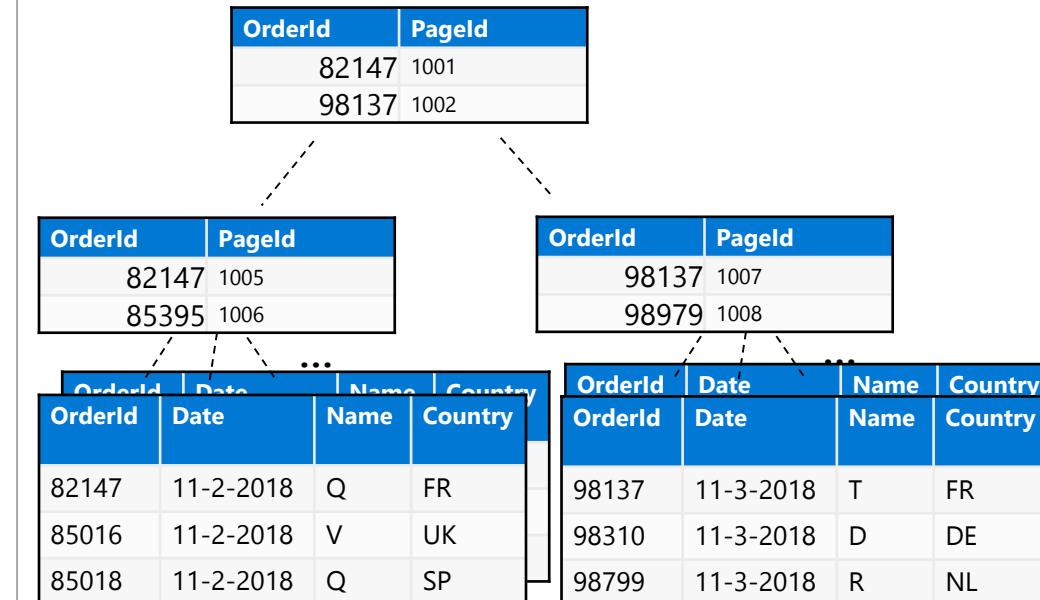
| OrderId | Date | Name | Country |
|---------|-----------|------|---------|
| 85016 | 11-2-2018 | V | UK |
| 85018 | 11-2-2018 | Q | SP |
| 85216 | 11-2-2018 | Q | DE |
| 85395 | 11-2-2018 | V | NL |
| 82147 | 11-2-2018 | Q | FR |
| 86881 | 11-2-2018 | D | UK |
| 93080 | 11-3-2018 | R | UK |
| 94156 | 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 |
| ... | ... | ... | ... |

...

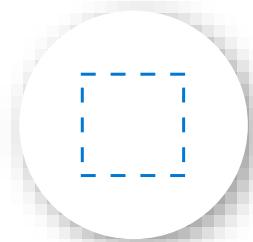
x 60 distributions (shards)

- 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

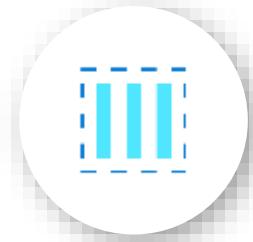
Common table distribution methods

| 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(<hashed_key>))</p> <p>OVER PARTITION BY <hashed_key></p> <p>most JOIN <table_name> ON <hashed_key></p> <p>GROUP BY <hashed_key></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. |

Database Views



Views

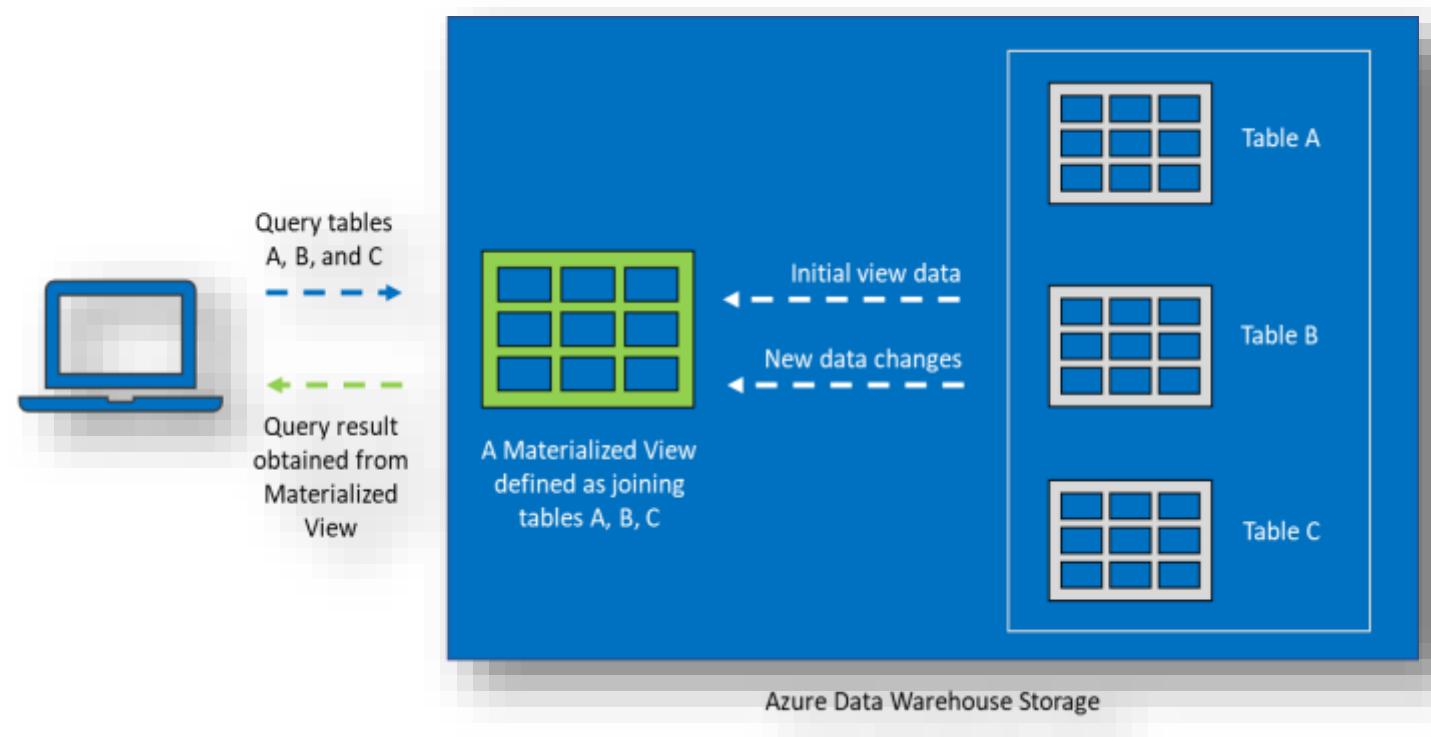


Materialized Views

Best in class price performance

Interactive dashboarding with Materialized Views

- Automatic data refresh and maintenance
- Automatic query rewrites to improve performance
- Built-in advisor



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

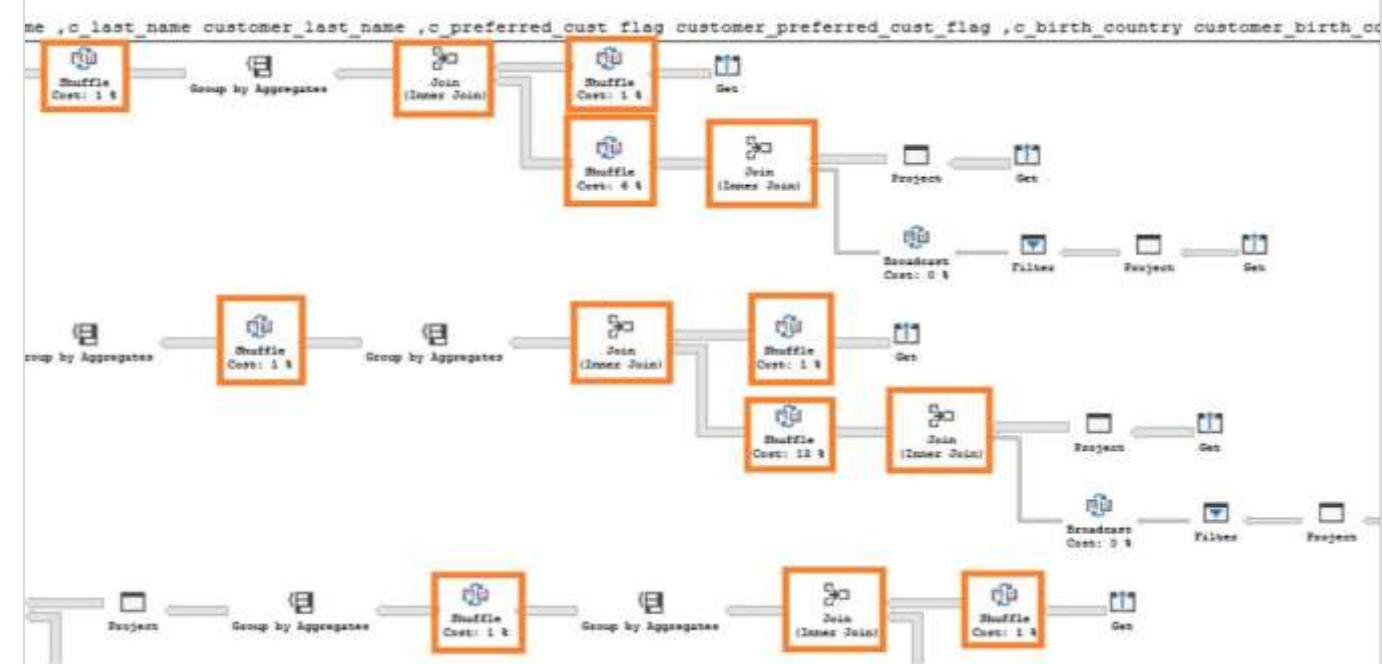
In this example, a query to get the year total sales per customer is shown to have a lot of data shuffles and joins that contribute to slow performance:

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

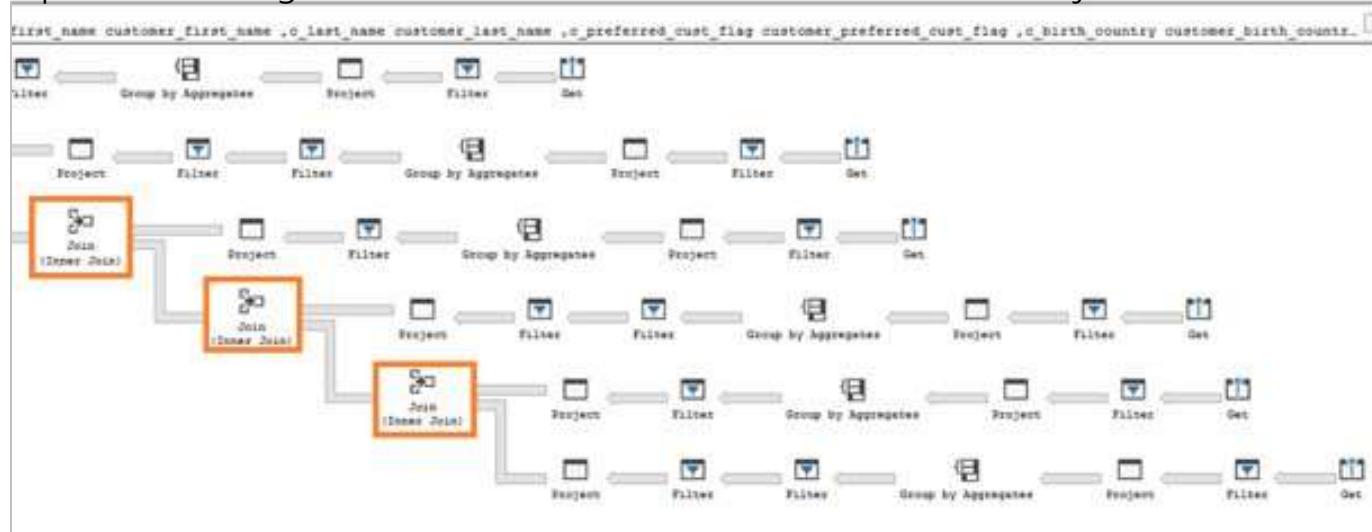
The SQL Data Warehouse query optimizer automatically leverages the indexed view to speed up the same query.
 Notice that the query does not need to reference the view directly

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 Data Warehouse SQL statement without running the statement; view estimated cost of the query operations.

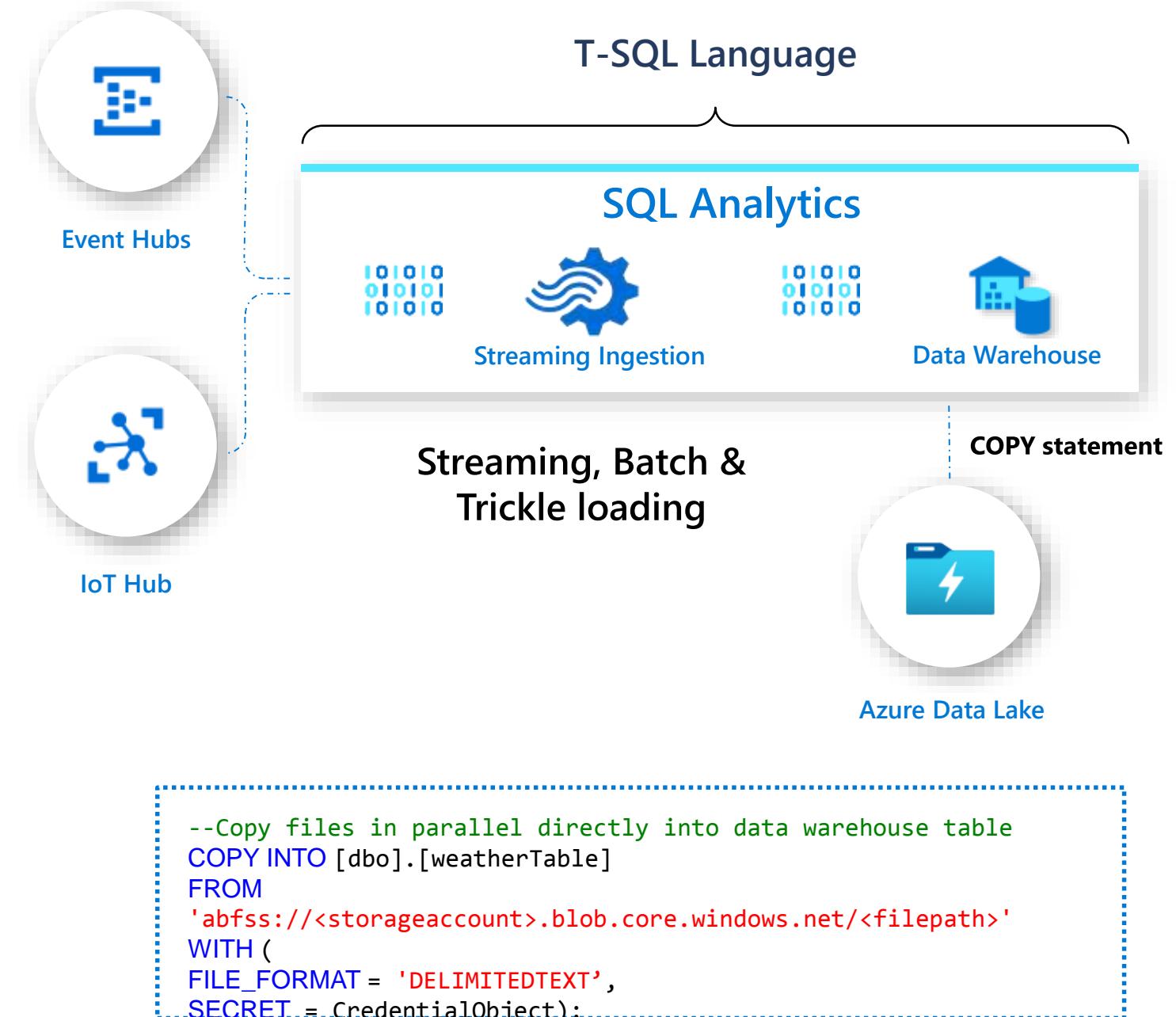
[EXPLAIN WITH_RECOMMENDATIONS](#) - provides query plan with recommendations to optimize the SQL statement performance.

```
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
```

Heterogenous Data Preparation & Ingestion

COPY statement

- Simplified permissions (no CONTROL required)
- No need for external tables
- Standard CSV support (i.e. custom row terminators, escape delimiters, SQL dates)
- User-driven file selection (wild card support)



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.

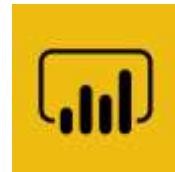
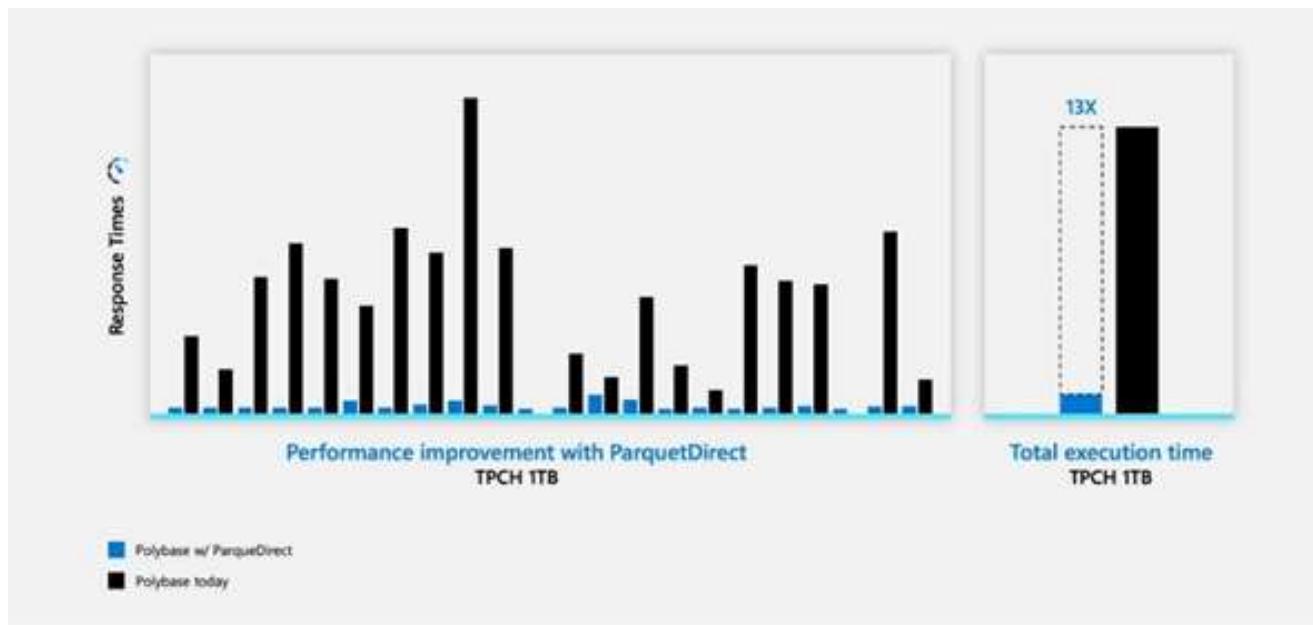
Supports CSV, PARQUET, ORC file formats

```
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='0X0A',
    ENCODING = 'UTF8',
    DATEFORMAT = 'ymd',
    MAXERRORS = 10,
    ERRORFILE = '/errorsfolder/--path starting from
the storage container,
    IDENTITY_INSERT
)
```

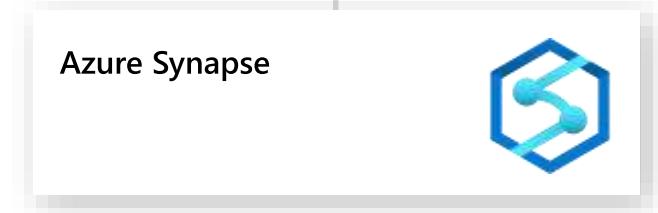
```
COPY INTO test_parquet
FROM
'https://XXX.blob.core.windows.net/customerdatasets/test.parquet'
WITH (
    FILE_FORMAT = myFileFormat
    CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>')
)
```

Data Flexibility – Parquet Direct

Overview



Dashboards, Reports, Ad-hoc analytics



Azure Synapse

Parquet

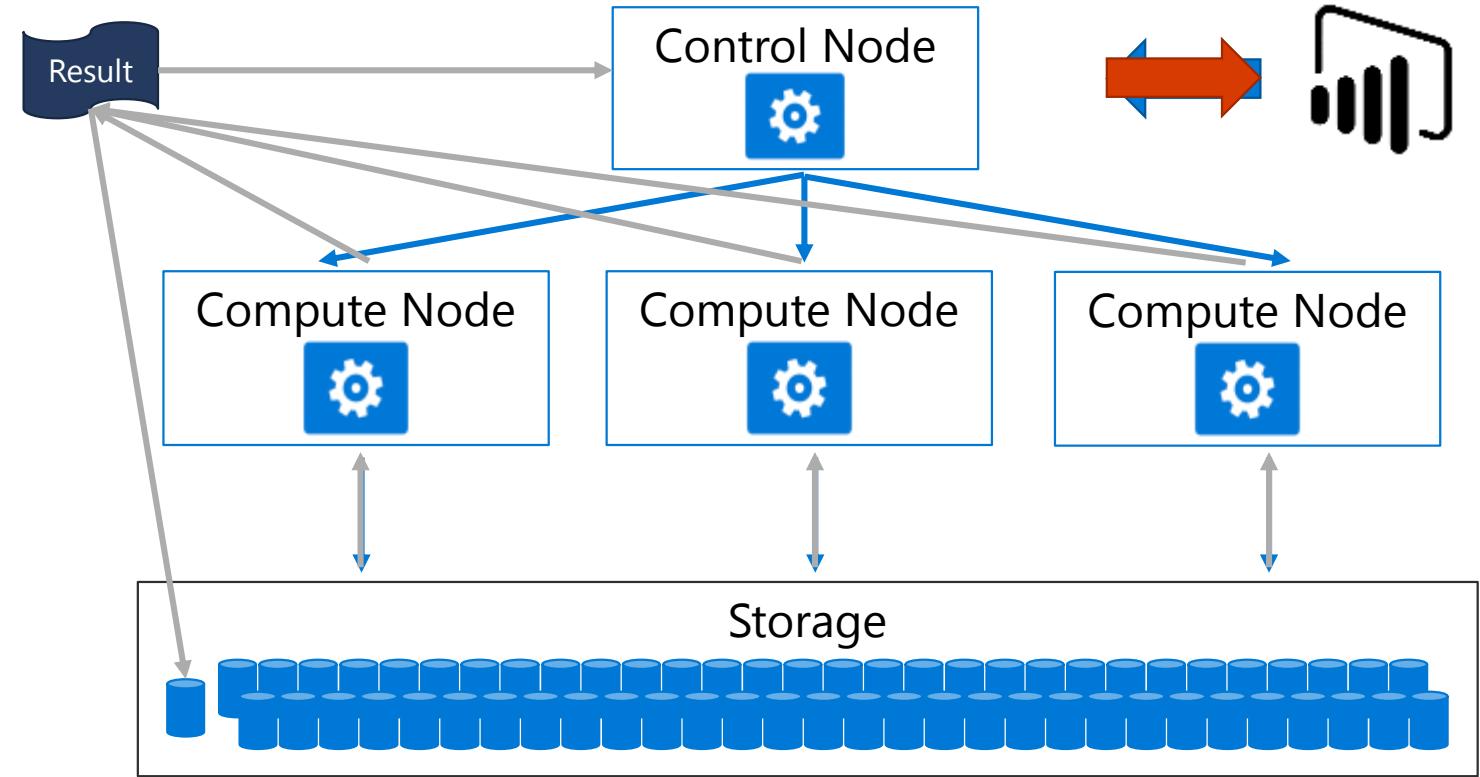


Data Lake Storage

Best in class price performance

Interactive dashboarding with Resultset Caching

- Millisecond responses with resultset caching
- Cache survives pause/resume/scale operations
- Fully managed cache (1TB in size)



Enable caching: **Alter Database <DBNAME> Set Result_Set_Caching ON**
Purge cache: **DBCC DropResultSetCache**

Result-set caching

Overview

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

The result-set cache persists even if a data warehouse 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

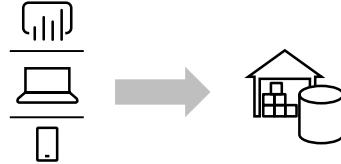
Enhances performance when same result is requested repetitively

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 data warehouse  
SET RESULT_SET_CACHING {ON | OFF}  
  
-- Check result-set caching setting for a database  
-- Run on target data warehouse  
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



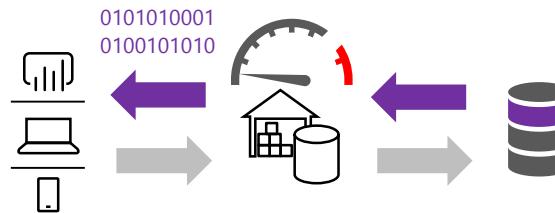
- 1 Client sends query to DW



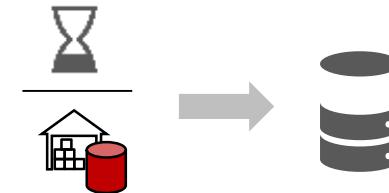
- 2 Query is processed using DW compute nodes which pull data from remote storage, process query and output back to client app



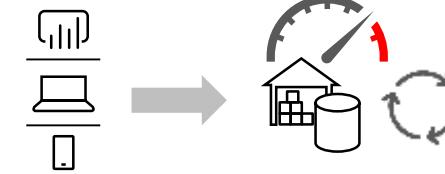
+
Query results are cached in remote storage so subsequent requests can be served immediately



- 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.



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

Resource classes

Overview

Pre-determined resource limits defined for a user or role.

Benefits

Govern the system memory assigned to each query.

Effectively used to control the number of concurrent queries that can run on a data warehouse.

Exemptions to concurrency limit:

CREATE|ALTER|DROP (TABLE|USER|PROCEDURE|VIEW|LOGIN)

CREATE|UPDATE|DROP (STATISTICS|INDEX)

SELECT from system views and DMVs

EXPLAIN

Result-Set Cache

TRUNCATE TABLE

ALTER AUTHORIZATION

CREATE|UPDATE|DROP STATISTICS

```
/* View resource classes in the data warehouse */
SELECT name
FROM sys.database_principals
WHERE name LIKE '%rc%' AND type_desc = 'DATABASE_ROLE';

/* Change user's resource class to 'largerc' */
EXEC sp_addrolemember 'largerc', 'loaduser';

/* Decrease the loading user's resource class */
EXEC sp_droprolemember 'largerc', 'loaduser';
```

Resource class types

Static Resource Classes

Allocate the same amount of memory independent of the current service-level objective (SLO).

Well-suited for fixed data sizes and loading jobs.

Dynamic Resource Classes

Allocate a variable amount of memory depending on the current SLO.

Well-suited for growing or variable datasets.

All users default to the *smallrc* dynamic resource class.

Static resource classes:

staticrc10 | staticrc20 | staticrc30 |
staticrc40 | staticrc50 | staticrc60 |
staticrc70 | staticrc80

Dynamic resource classes:

smallrc | mediumrc | largerc | xlargerc

| Resource Class | Percentage Memory | Max. Concurrent Queries |
|----------------|-------------------|-------------------------|
| smallrc | 3% | 32 |
| mediumrc | 10% | 10 |
| largerc | 22% | 4 |
| xlargerc | 70% | 1 |

Concurrency slots

Overview

Queries running on a DW compete for access to system resources (CPU, IO, and memory).

To guarantee access to resources, running queries are assigned a chunk of system memory (a **concurrency slot**) for processing the query. The amount given is determined by the resource class of the user executing the query. Higher DW SLOs provide more memory and concurrency slots

@DW1000c: 40 concurrency slots

Memory (concurrency slots)



| |
|------------------------------------|
| Smallrc query (1 slot each) |
| Staticrc20 query (2 slots each) |
| Mediumrc query (4 slots each) |
| Xlargerc query (28 slots each) |

Concurrent query limits

Overview

The limit on how many queries can run at the same time is governed by two properties:

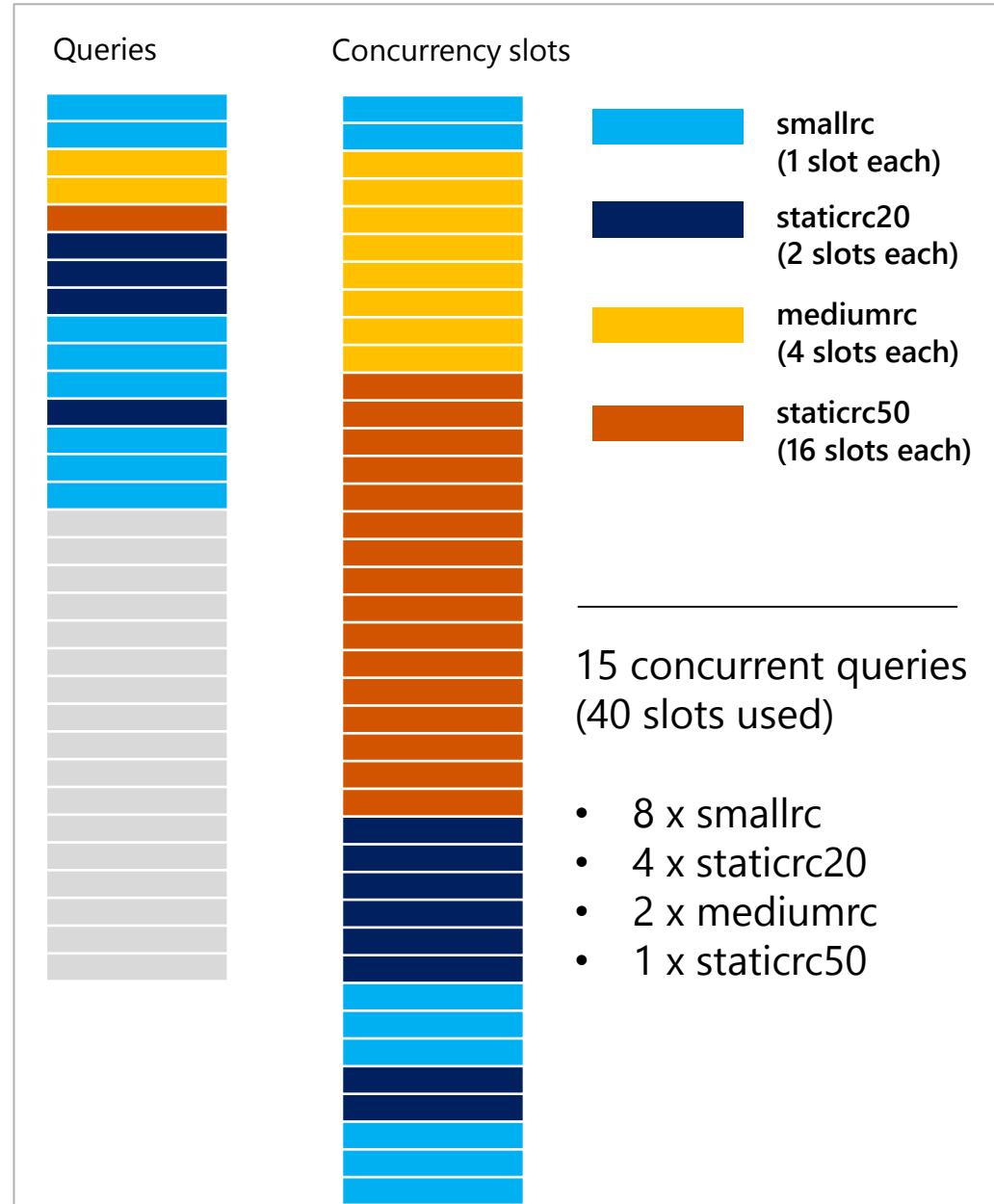
- The max. concurrent query count for the DW SLO
- The total available memory (concurrency slots) for the DW SLO

Increase the concurrent query limit by:

- Scaling up to a higher DW SLO (up to 128 concurrent queries)
- Using lower resource classes that use less memory per query

Concurrency limits based on resource classes

@DW1000c: 32 max concurrent queries, 40 slots



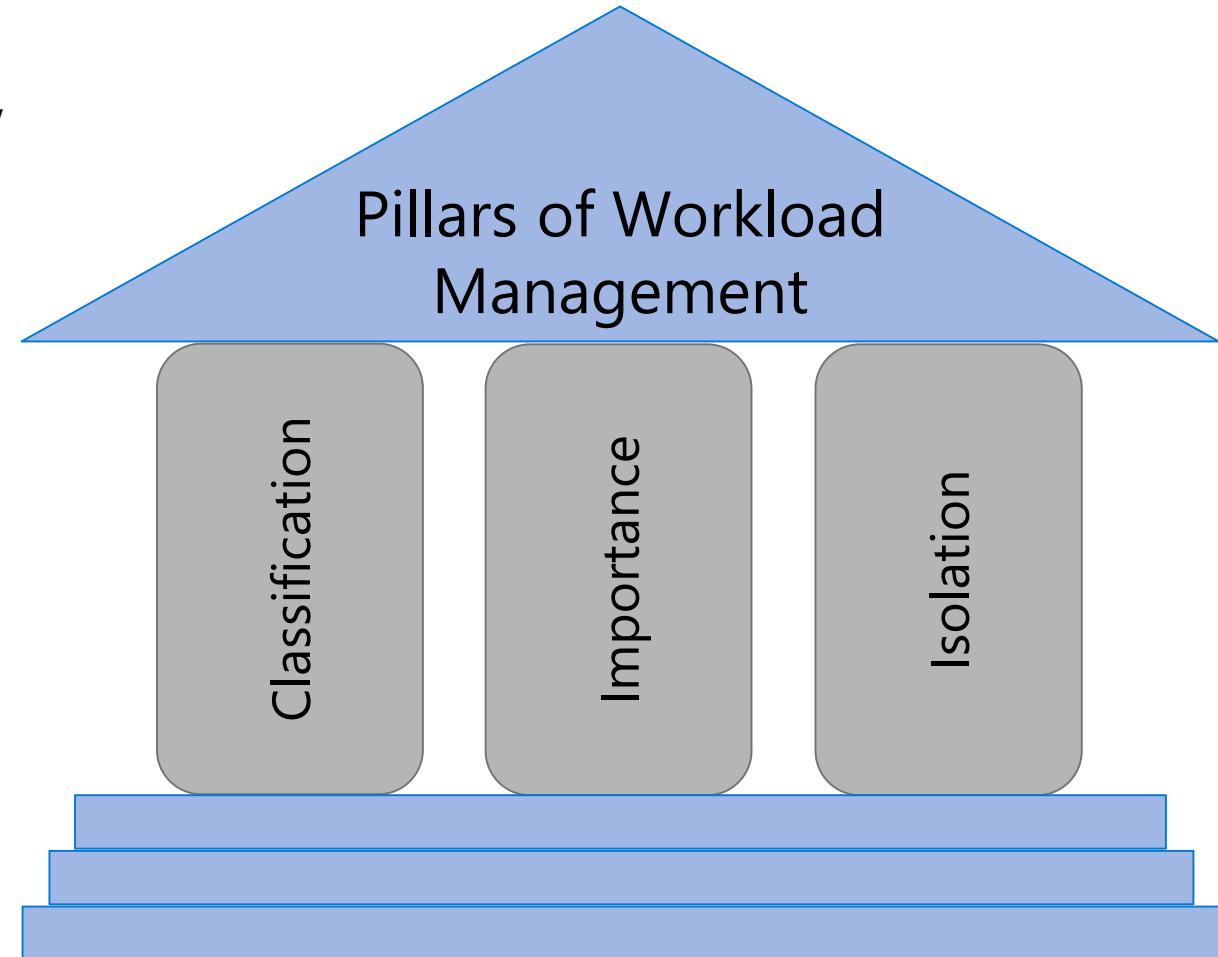
Workload Management

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 (smallrc resource class).

Benefits

Map queries to both Resource Management and Workload Isolation concepts.

Manage groups of users with only a few classifiers.

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 = '<Resource Class>']
    [IMPORTANCE = { LOW
                    |
                    BELOW_NORMAL
                    |
                    NORMAL
                    |
                    ABOVE_NORMAL
                    |
                    HIGH
                }
    ]
    [MEMBERNAME = 'security_account']
)
```

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

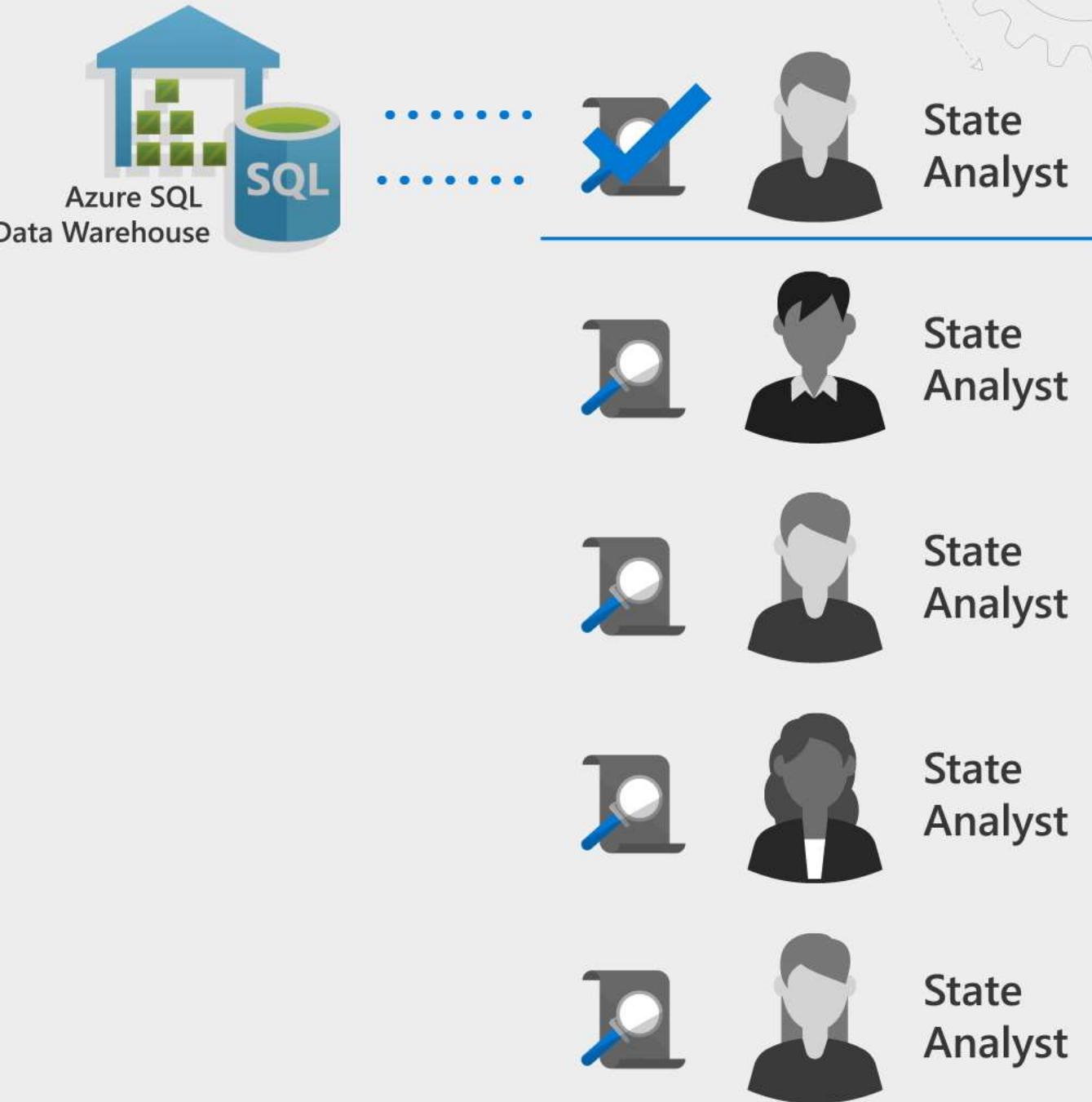
State analysts have normal importance.

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 = 'smallrc']
    [IMPORTANCE = HIGH]
    [MEMBERNAME = 'National_Analyst_Login']
```



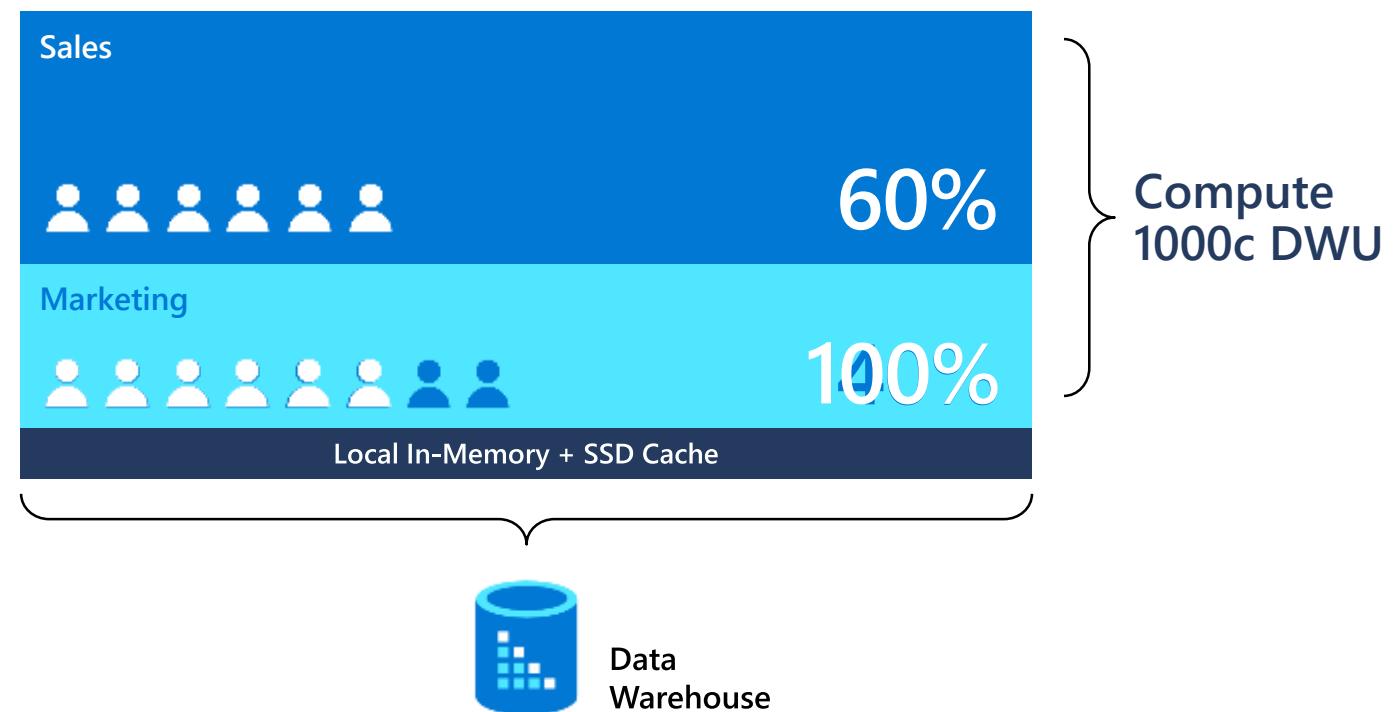
Workload aware query execution

Workload Isolation

- Multiple workloads share deployed resources
- Reservation or shared resource configuration
- Online changes to workload policies

```
CREATE WORKLOAD GROUP Sales  
WITH  
(  
    [ MIN_PERCENTAGE_RESOURCE = 60 ]  
    [ CAP_PERCENTAGE_RESOURCE = 100 ]  
    [ MAX_CONCURRENCY = 6 ] )
```

Intra Cluster Workload Isolation (Scale In)



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

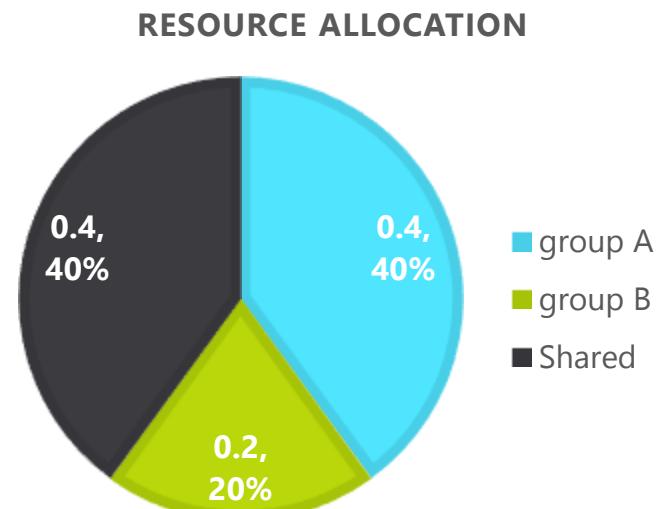
Set Query timeout value. Get DBAs out of the business of killing runaway queries

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

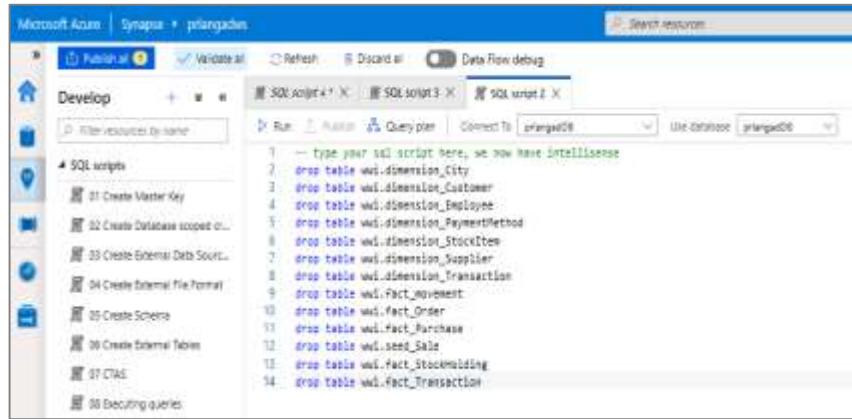
```
-- List all open sessions
SELECT * FROM sys.dm_pdw_exec_sessions where status <> 'Closed'
and session_id <> session_id();
```

List all active queries

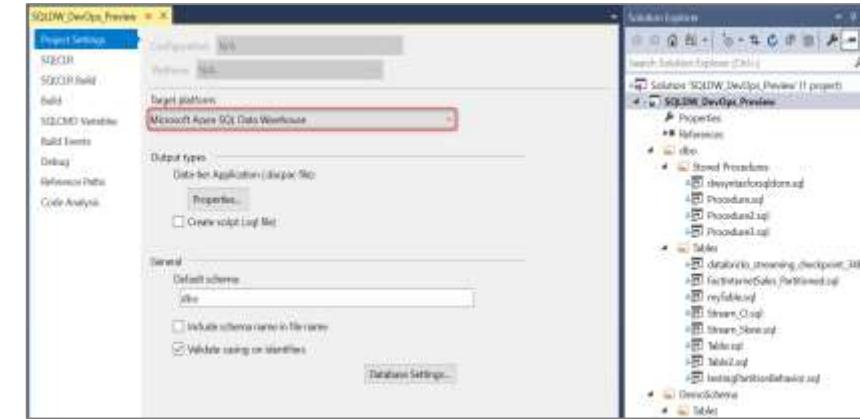
```
-- List all active queries
SELECT * FROM sys.dm_pdw_exec_requests WHERE status not in
('Completed','Failed','Cancelled') AND session_id <> session_id()
ORDER BY submit_time DESC;
```

Developer Tools

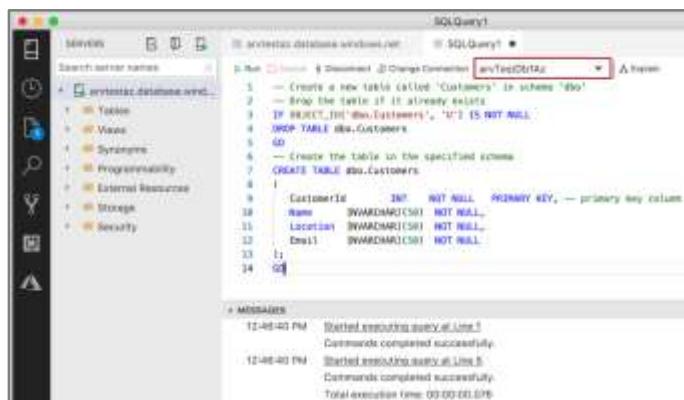
Azure Synapse Analytics



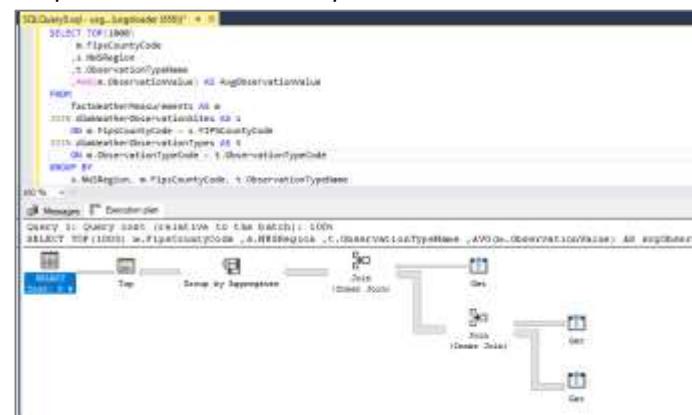
Visual Studio - SSDT database projects



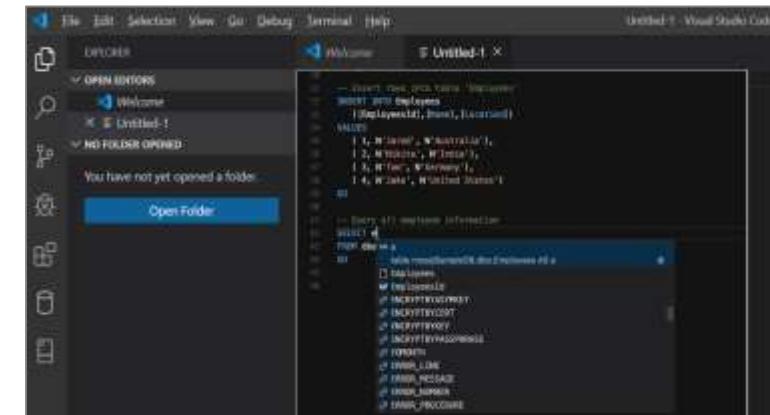
Azure Data Studio (queries, extensions etc.)



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

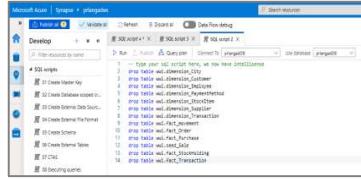


Visual Studio Code



Developer Tools

Azure Synapse Analytics

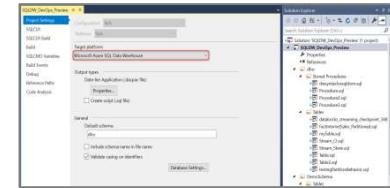


Azure Cloud Service

Offers end-to-end lifecycle for analytics

Connects to multiple services

Visual Studio - SSDT database projects



Runs on Windows

Create, maintain database code, compile, code refactoring

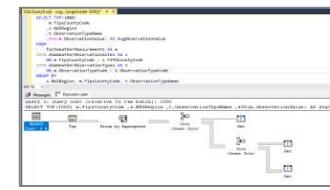
Azure Data Studio



Runs on Windows, Linux, macOS

Light weight editor, (queries and extensions)

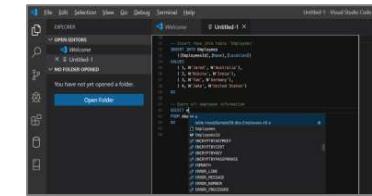
SQL Server Management Studio



Runs on Windows

Offers GUI support to query, design and manage

Visual Studio Code



Runs on Windows, Linux, macOS

Offers development experience with light-weight code editor

Continuous integration and delivery (CI/CD)

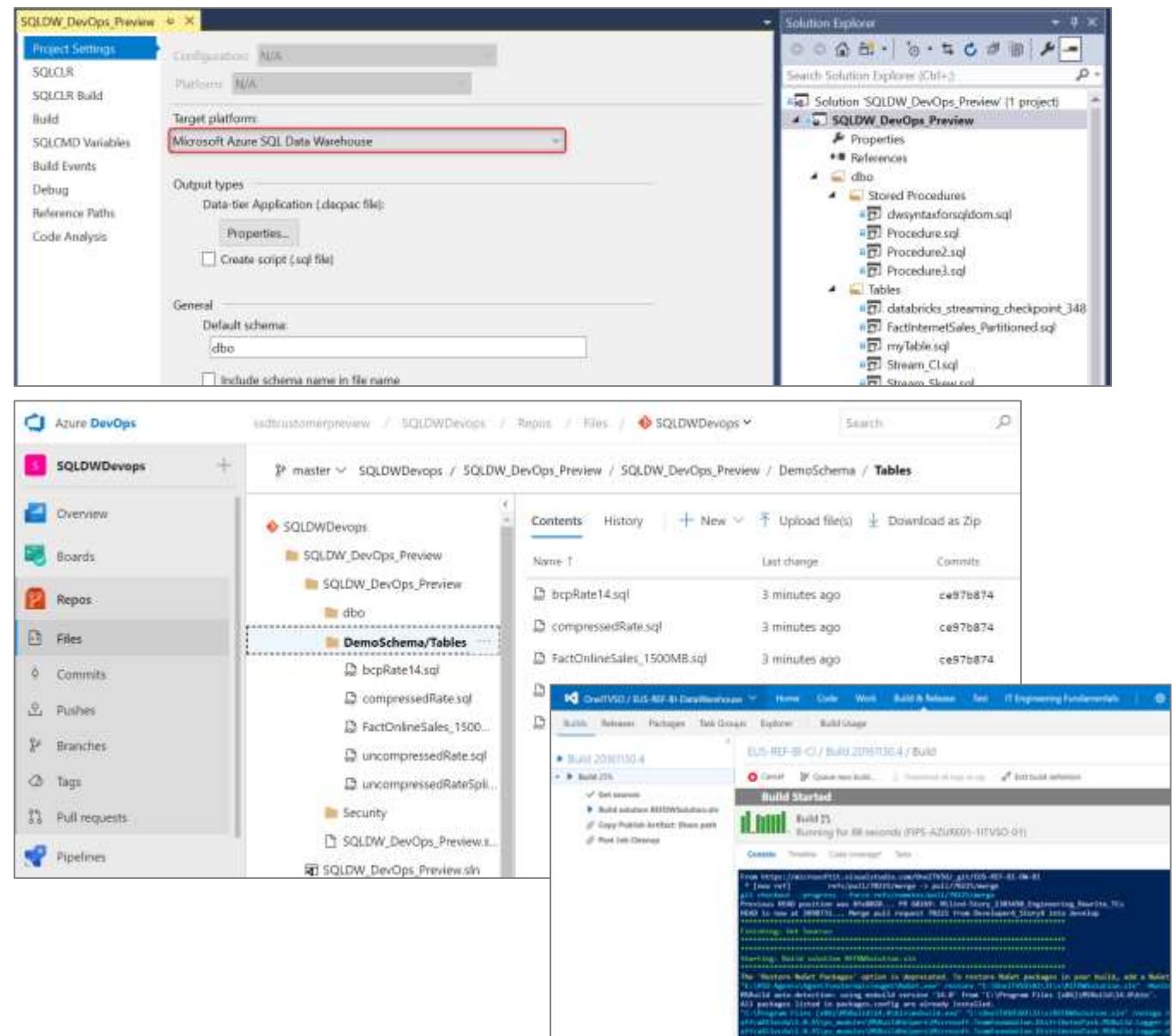
Overview

Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled data warehouse, 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

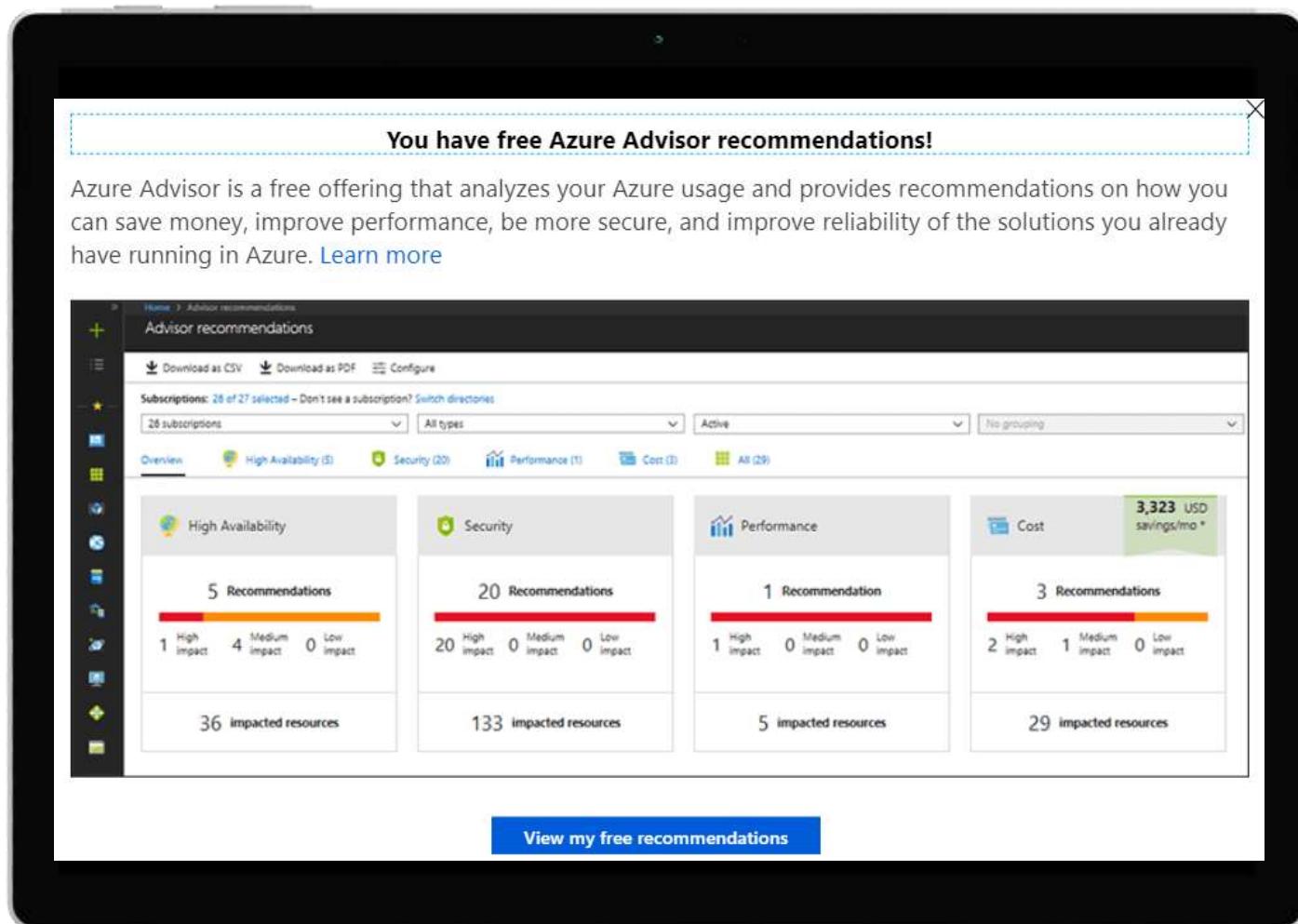
Provision additional capacity

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 pool. 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 }
```

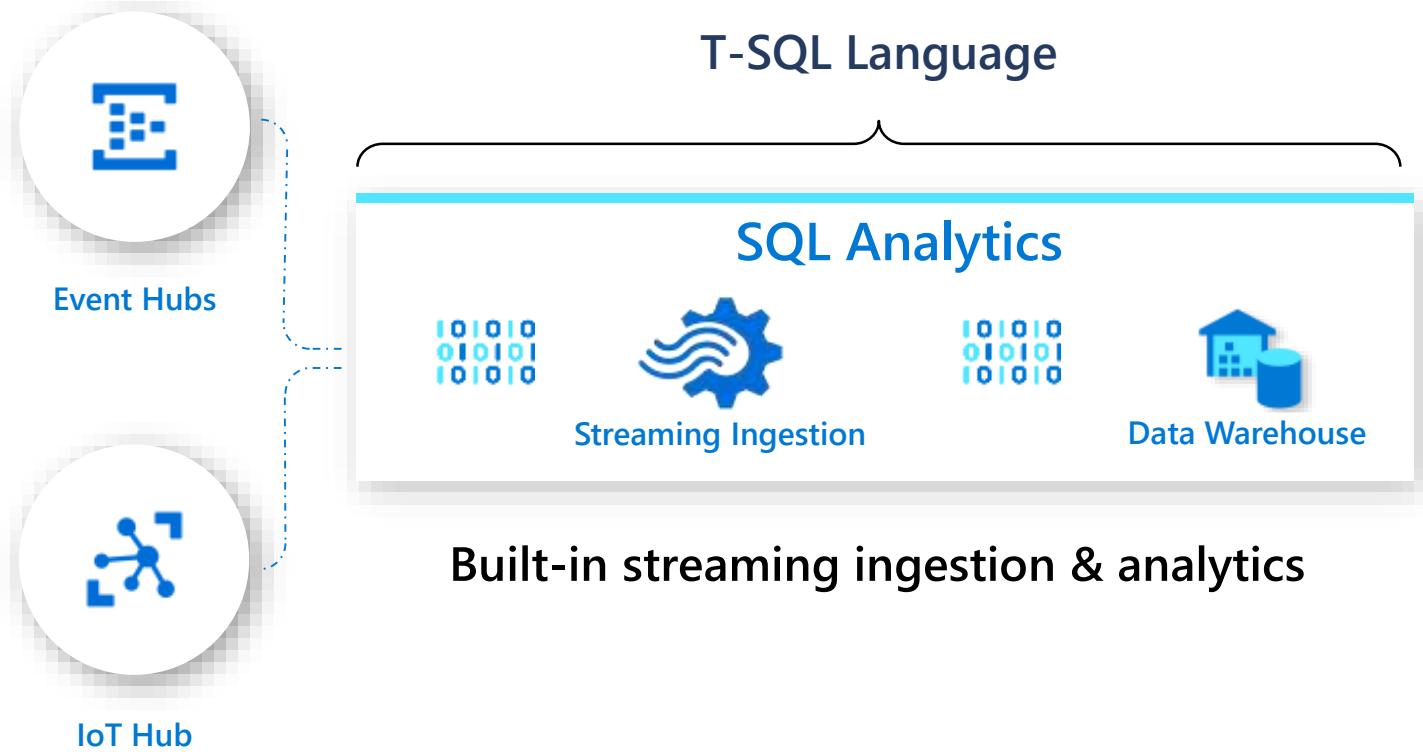
-- Check statistics settings for a database

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

Heterogenous Data Preparation & Ingestion

Native SQL Streaming

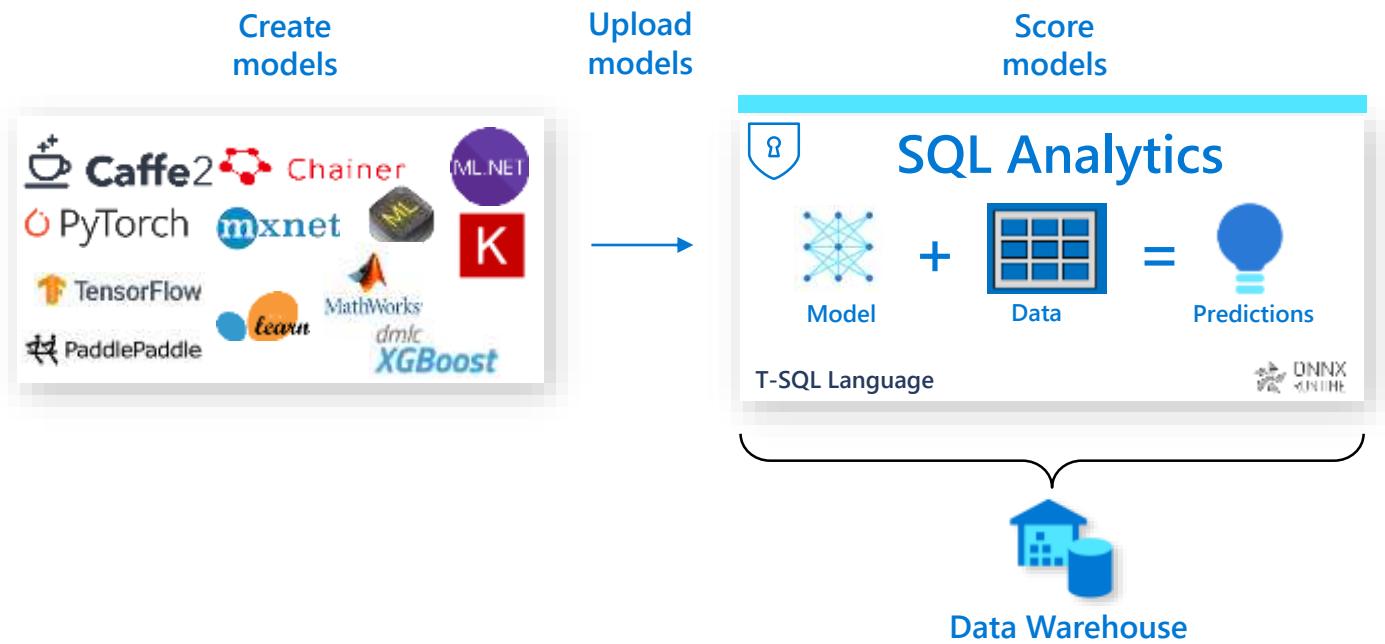
- High throughput ingestion (up to 200MB/sec)
- Delivery latencies in seconds
- Ingestion throughput scales with compute scale
- Analytics capabilities (SQL-based queries for joins, aggregations, filters)
- *Removes the need to use Spark for streaming*



Machine Learning enabled DW

Native PREDICT-ion

- T-SQL based experience (interactive./batch scoring)
- Interoperability with other models built elsewhere
- Execute scoring where the data lives

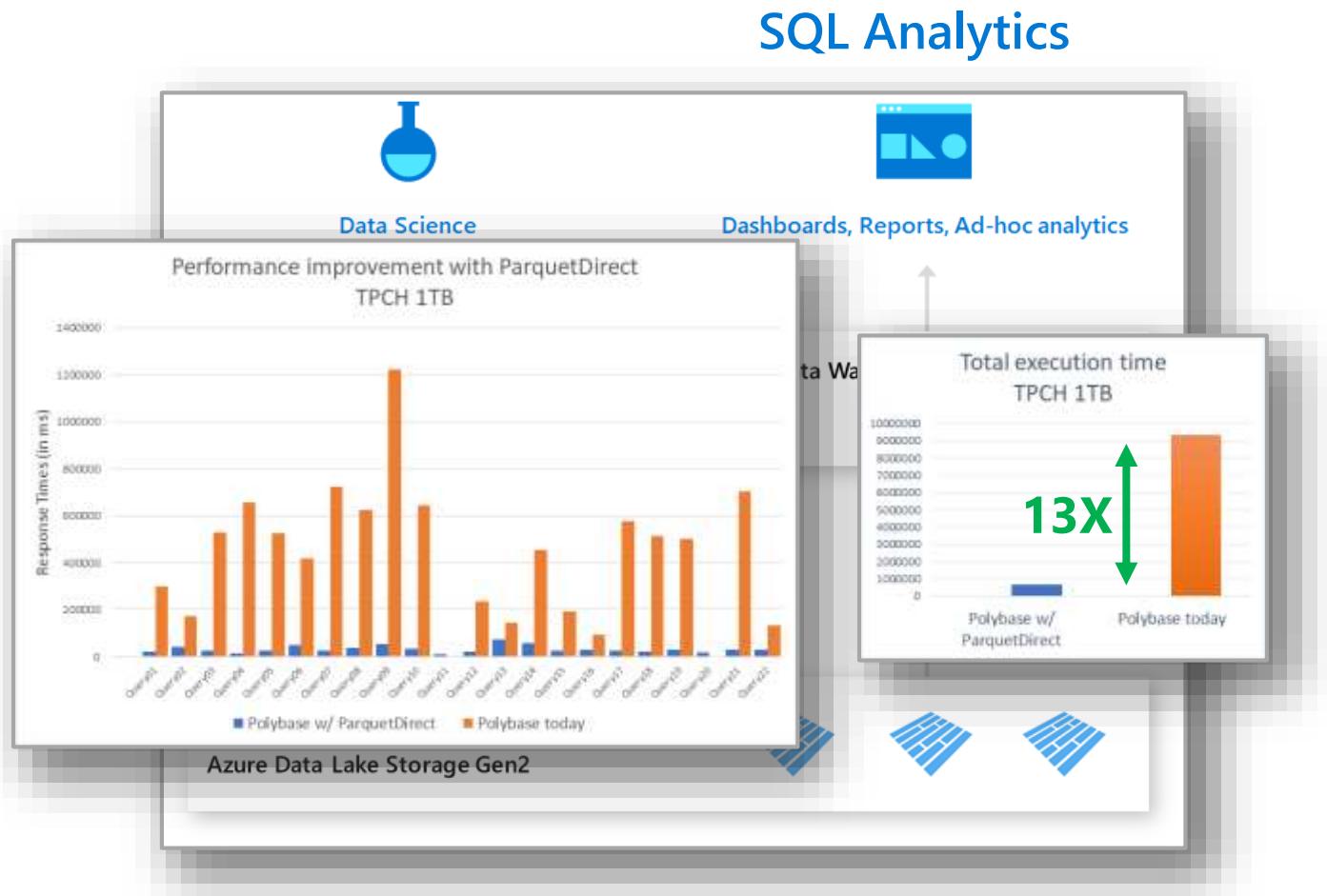


```
--T-SQL syntax for scoring data in SQL DW
SELECT d.*, p.Score
FROM PREDICT(MODEL = @onnx_model, DATA =
dbo.mytable AS d)
WITH (Score float) AS p;
```

Data Lake Integration

ParquetDirect for interactive data lake exploration

- >10X performance improvement
- Full columnar optimizations (optimizer, batch)
- Built-in transparent caching (SSD, in-memory, resultset)



Azure Data Share

Enterprise data sharing

- Share from DW to DW/DB/other systems
- Choose data format to receive data in (CSV, Parquet)
- One to many data sharing
- Share a single or multiple datasets

| Feature | Azure Data Share |
|---|------------------|
| Multiple Data Store Support Sharing from Azure Data Lake, Azure Storage, Azure SQL Data Warehouse, Azure SQL DB | Yes |
| Heterogenous Data Sharing Flexible sharing from/to heterogenous data stores | Yes |
| Single pane of glass Centrally managed data sharing experience | Yes |
| Governed data sharing Customer can specify terms of use | Yes |
| Snapshot based sharing Perform analytics on data for unrestricted computation & no compromise on performance | Yes |



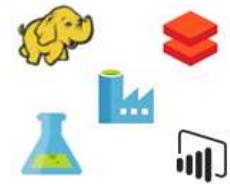
Any Azure Data Sources

Share data from any Azure regions and data stores



Single Pane of Glass

Manage and monitor data sharing with multiple organizations



Rich Analytics Tools

Use Azure analytics tools to prepare data and derive insights



Governance

Control data access governed by enterprise policies



Monetization

Charge for data or cost of data curation and access

SQL Analytics

new features available

GA features:

- **Performance:** Resultset caching
- **Performance:** Materialized Views
- **Performance:** Ordered columnstore
- **Heterogeneous data:** JSON support
- **Trustworthy computation:** Dynamic Data Masking
- **Continuous integration & deployment:** SSDT support
- **Language:** Read committed snapshot isolation

Public preview features:

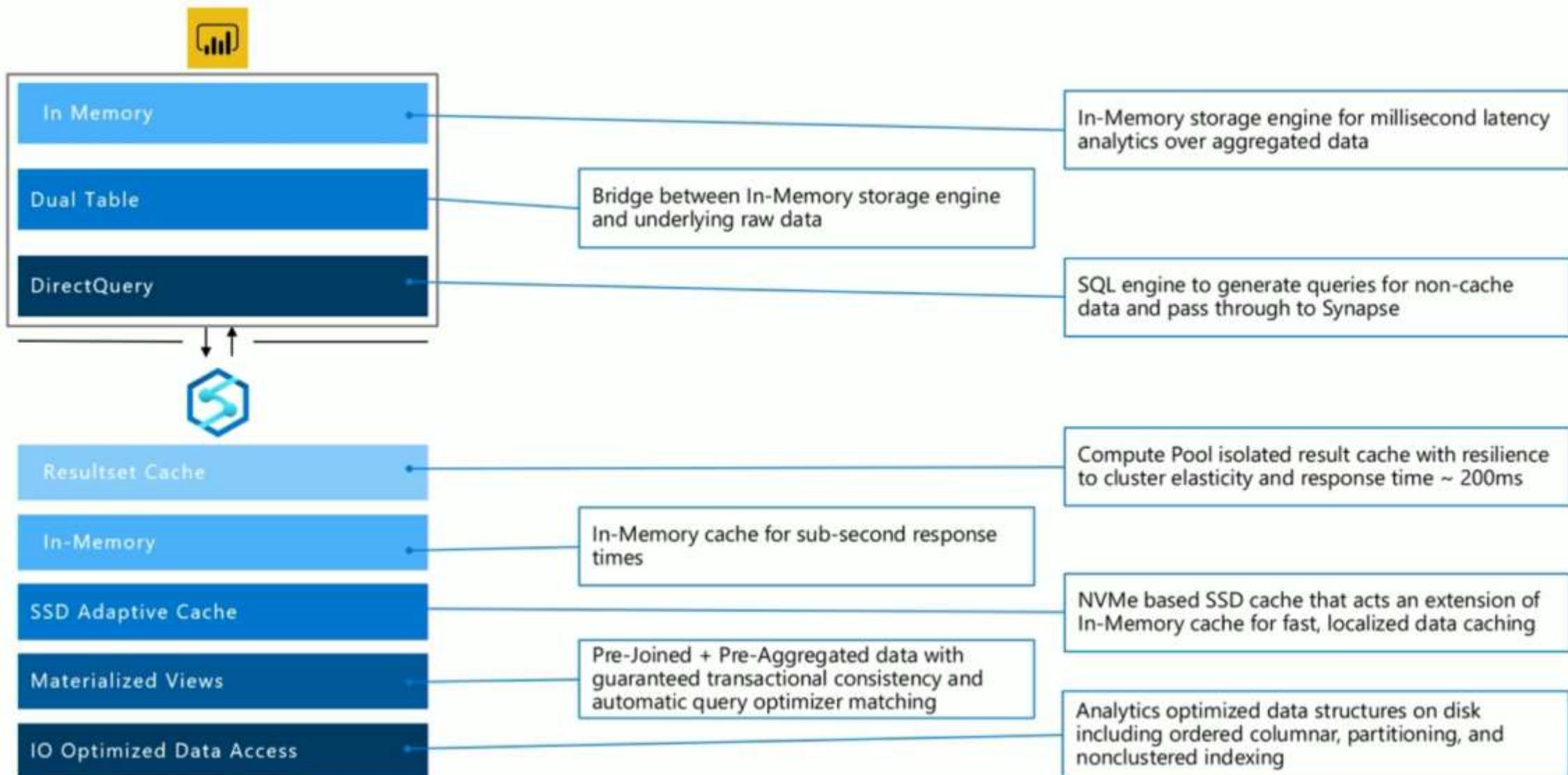
- **Workload management:** Workload Isolation
- **Data ingestion:** Simple ingestion with COPY
- **Data Sharing:** Share DW data with Azure Data Share
- **Trustworthy computation:** Private LINK support

Private preview features:

- **Data ingestion:** Streaming ingestion & analytics in DW
- **Built-in ML:** Native Prediction/Scoring
- **Data lake enabled:** Fast query over Parquet files
- **Language:** Updateable distribution column
- **Language:** FROM clause with joins
- **Language:** Multi-column distribution support
- **Security:** Column-level Encryption

Note: private preview features require whitelisting

Power BI Aggregations and Synapse query performance





Azure Synapse Analytics

SQL On-Demand

Query Options

1. Provisioned SQL over relational database – Traditional SQL DW [existing]
2. Provisioned SQL over ADLS Gen2 – via external tables or openrowset [existing via PolyBase]
3. On-demand SQL over relational database - dependency on the flexible data model (data cells) over columnstore data (preview) [new]
4. On-demand SQL over ADLS Gen2 – via external tables or openrowset [new]
5. Provisioned Spark over relational database – Not possible
6. Provisioned Spark over ADLS Gen2 [new]
7. On-demand Spark over relational database - On-demand Spark is not supported
8. On-demand Spark over ADLS Gen2 – On-demand Spark is not supported

Notes:

- Separation of state (data, metadata and transactional logs) and compute
- Queries against data loaded into SQL Analytics tables are faster 2-3X compared to queries over external tables
- Improved performance compared to PolyBase. PolyBase is not used, but functional aspects are supported
- SQL on-demand will push down queries from the front-end to back-end nodes
- Warm-up for first on-demand query takes about 20-25 seconds
- If you create a Spark Table, that table will be created as an external table in SQL Pool or On-Demand without having to keep a Spark cluster up and running

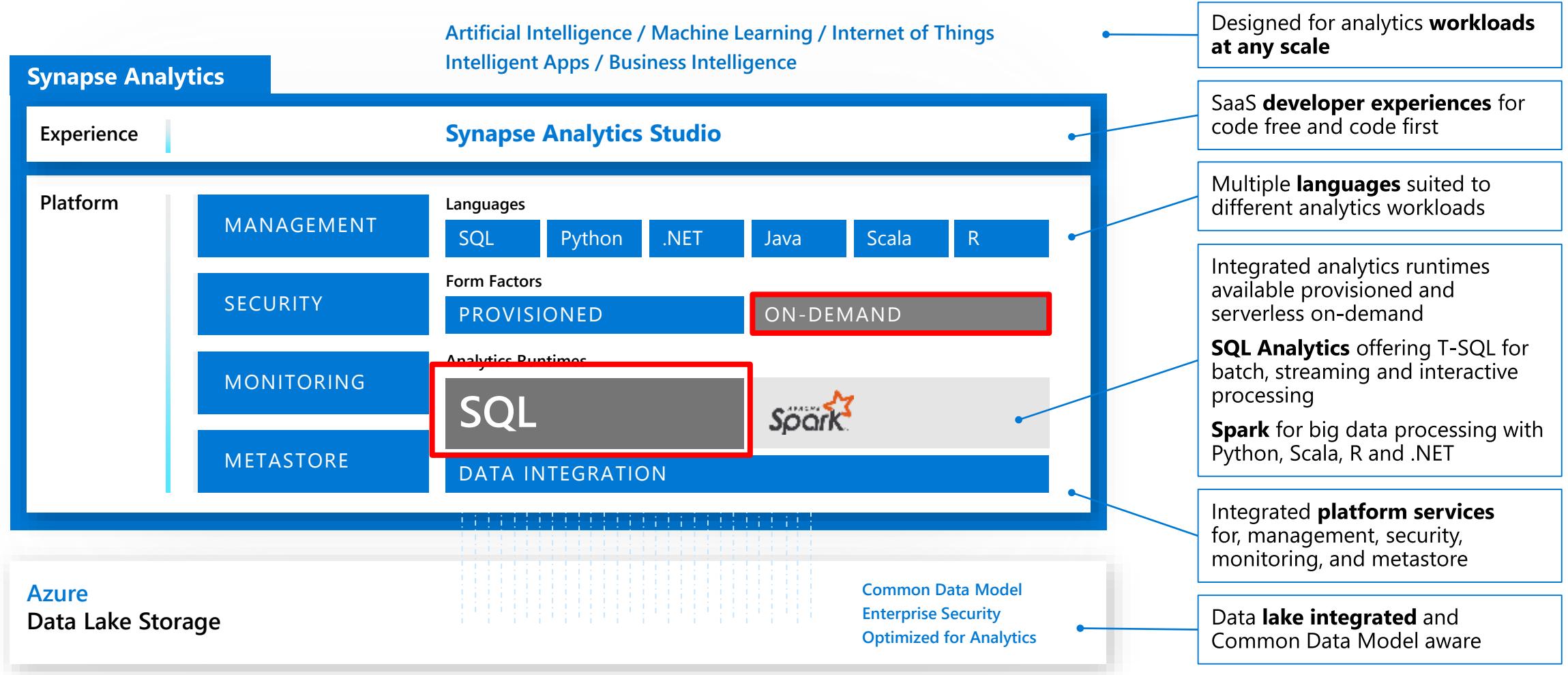
Distributed Query Processor (DQP)

- **Auto-scale compute nodes** - Instruct the underlying fabric the need for more compute power to adjust to peaks during the workload. If compute power is granted, the Polaris DQP will re-distribute tasks leveraging the new compute container. Note that in-flight tasks in the previous topology continue running, while new queries get the new compute power with the new re-balancing
- **Compute node fault tolerance** - Recover from faulty nodes while a query is running. If a node fails the DQP re-schedules the tasks in the faulted node through the remainder of the healthy topology
- **Compute node hot spot: rebalance queries or scale out nodes** - Can detect hot spots in the existing topology. That is, overloaded compute nodes due to data skew. In the advent of a compute node running hot because of skewed tasks, the DQP can decide to re-schedule some of the tasks assigned to that compute node amongst others where the load is less
- **Multi-cluster** - Multiple compute pools accessing the same data
- **Cross-database queries** – A query can specify multiple databases

These features work for both on-demand and provisioned over ADLS Gen2 and relational databases

Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Synapse SQL on-demand scenarios

Discovery and exploration

What's in this file? How many rows are there? What's the max value?

SQL On-demand reduces data lake exploration to the right-click!

Data transformation

How to convert CSVs to Parquet quickly? How to transform the raw data?

Use the full power of T-SQL to transform the data in the data lake

SQL On-Demand

Overview

An interactive query service that provides T-SQL queries over high scale data in Azure Storage.

Benefits

Serverless

No infrastructure

Pay only for query execution

No ETL

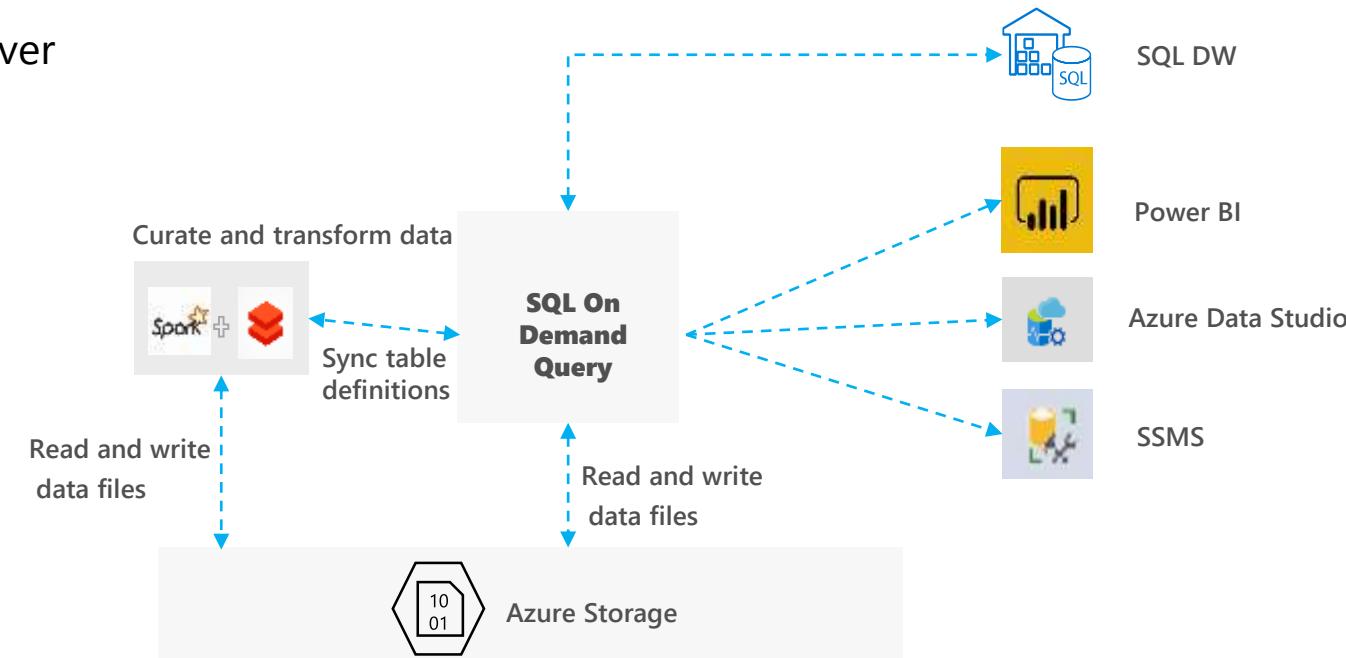
Offers security

Data integration with Databricks, HDInsight

T-SQL syntax to query data

Supports data in various formats (Parquet, CSV, JSON)

Support for BI ecosystem



SQL On Demand – Querying on storage

The screenshot shows two instances of the Microsoft Azure Synapse Analytics portal.

Left Instance:

- Shows a storage account named `prlangaddemo (Primary)` with a container named `nytic` selected.
- A context menu is open over a file named `part-00133`, with the option `New SQL script` highlighted.
- The URL for the selected file is displayed as `https://prlangaddemo.azuredatalakestorage.net/nytic/c/yellow/puYear=2015/puMonth=3/part-00133-tid:218928564719830543-aea5b543-5e83-4a7d-8d31-69f72c500050-15253-1.c000.snappy.parquet`.

Right Instance:

- Shows a database named `matter` selected.
- The `Connect to` dropdown is set to `SQL Analytics on-demand`.
- The SQL query window contains the following code:


```
1 SELECT
2   TOP 100 *
3   FROM
4   OPENROWSET(
5     FILEBROWSER,
6     'https://prlangaddemo.azuredatalakestorage.net/nytic/c/yellow/puYear=2015/puMonth=3/part-00133-tid:218928564719830543-aea5b543-5e83-4a7d-8d31-69f72c500050-15253-1.c000.snappy.parquet',
7     FORMAT='PARQUET'
8   ) AS nytes
```
- The results pane displays the following table data (partial results shown):

| VENDORID | TPEPICKUPDATETIME | TPEPDISPATCHDATE_ | PASSENGERCOUNT | TRIPDISTANCE | PILOCATIONID | DOLocationID | STARTSTATION | ENDSTATION | TRIPID |
|----------|--------------------|--------------------|----------------|--------------|--------------|--------------|--------------------|------------------|--------|
| 2 | 2015-02-28T23:5... | 2015-03-01T00:0... | 6 | 1.63 | 40011 | 40011 | -74.000540662793 | 40.7306938171387 | -73 |
| 1 | 2015-03-28T19:2... | 2015-03-28T19:2... | 1 | 2.2 | 40011 | 40011 | -73.977653503418 | 40.7621607035664 | -73 |
| 2 | 2015-02-28T23:5... | 2015-03-01T00:1... | 5 | 3.23 | 40011 | 40011 | -73.96012879417... | 40.7621574462185 | -73 |
| 1 | 2015-03-28T19:2... | 2015-03-28T19:3... | 1 | 2.1 | 40011 | 40011 | -73.9814305371... | 40.7815255847368 | -74 |
| 2 | 2015-02-28T23:5... | 2015-03-01T00:1... | 1 | 3.52 | 40011 | 40011 | -73.98373413085... | 40.7497062683105 | -74 |
| 5 | 2015-02-28T23:5... | 2015-03-01T00:1... | 6 | 1.63 | 40011 | 40011 | -73.9814305371... | 40.7815255847368 | -74 |
- The status message at the bottom right indicates: `00:01:00-Query executed successfully.`

SQL On Demand – Querying CSV File

Overview

Uses OPENROWSET function to access data

Benefits

Ability to read CSV File with

- no header row, Windows style new line
- no header row, Unix-style new line
- header row, Unix-style new line
- header row, Unix-style new line, quoted
- header row, Unix-style new line, escape
- header row, Unix-style new line, tab-delimited
- without specifying all columns

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

| | country_code | country_name | year | population |
|---|--------------|--------------|------|------------|
| 1 | LU | Luxembourg | 2017 | 594130 |

SQL On Demand – Querying CSV File

Read CSV file - header row, Unix-style new line

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population-
unix-hdr/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '0x0a',
    FIRSTROW = 2
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

| | country_code | country_name | year | population |
|---|--------------|--------------|------|------------|
| 1 | LU | Luxembourg | 2017 | 594130 |

Read CSV file - without specifying all columns

```
SELECT
    COUNT(DISTINCT country_name) AS countries
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/popul-
ation/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_name] VARCHAR (100) COLLATE Latin1_Gener-
al_BIN2 2
) AS [r]
```

| | countries |
|---|-----------|
| 1 | 228 |

SQL On Demand – Querying folders

Overview

Uses OPENROWSET function to access data from multiple files or folders

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://XXX.blob.core.windows.net/csv/taxi/*.*',
    FORMAT = 'CSV'
    , FIRSTROW = 2 )
WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    rate_code INT,
    store_and_fwd_flag VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_location_id INT,
    dropoff_location_id INT,
    payment_type INT,
    fare_amount FLOAT,
    extra FLOAT, mta_tax FLOAT,
    tip_amount FLOAT,
    tolls_amount FLOAT,
    improvement_surcharge FLOAT,
    total_amount FLOAT
) 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 |

SQL On Demand – Querying folders

Read all files from multiple folders

```
SELECT YEAR(pickup_datetime) AS [year],
       SUM(passenger_count) AS passengers_total,
       COUNT(*) AS [rides_total]
  FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/t*i/',
    FORMAT = 'CSV',
    FIRSTROW = 2
  )
  WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    <... columns>
  ) 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 |

Read subset of files in folder

```
SELECT
  payment_type,
  SUM(fare_amount) AS fare_total
  FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-* .csv',
    FORMAT = 'CSV',
    FIRSTROW = 2
  )
  WITH (
    vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count INT,
    trip_distance FLOAT,
    <...columns>
  ) AS nyc
  GROUP BY payment_type
  ORDER BY payment_type
```

| | payment_type | fare_total |
|---|--------------|-------------------|
| 1 | 1 | 1026072325.579... |
| 2 | 2 | 441093322.8000... |
| 3 | 3 | 10435183.04 |
| 4 | 4 | 3304550.99 |
| 5 | 5 | 14 |

SQL On Demand – Querying specific files

Overview

filename – Provides file name that originates row result

filepath – Provides full path when no parameter is passed or part of path when parameter is passed that originates result

Benefits

Provides source name/path of file/folder for row result set

Example of filename function

```

SELECT
    r.filename() AS [filename]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_201
7-1*.csv',
    FORMAT = 'CSV',
    FIRSTROW = 2
)
WITH (
    vendor_id INT,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count SMALLINT,
    trip_distance FLOAT,
    <...columns>
) AS [r]
GROUP BY r.filename()
ORDER BY [filename]

```

| | filename | rows |
|---|-----------------------------|---------|
| 1 | yellow_tripdata_2017-10.csv | 9768815 |
| 2 | yellow_tripdata_2017-11.csv | 9284803 |
| 3 | yellow_tripdata_2017-12.csv | 9508276 |

SQL On Demand – Querying specific files

Example of filepath function

```

SELECT
    r.filepath() AS filepath
    ,r.filepath(1) AS [year]
    ,r.filepath(2) AS [month]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_*.csv',
    FORMAT = 'CSV',
    FIRSTROW = 2
)
WITH (
    vendor_id INT,
    pickup_datetime DATETIME2,
    dropoff_datetime DATETIME2,
    passenger_count SMALLINT,
    trip_distance FLOAT,
    <... columns>
) 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

```

| filepath | year | month | rows |
|--|------|-------|---------|
| https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-10.csv | 2017 | 10 | 9768815 |
| https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-11.csv | 2017 | 11 | 9284803 |
| https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-12.csv | 2017 | 12 | 9508276 |

SQL On Demand – Querying Parquet files

Overview

Uses OPENROWSET function to access data

Benefits

Ability to specify column names of interest

Offers auto reading of column names and data types

Provides target specific partitions using filepath function

```

SELECT
    YEAR(pickup_datetime),
    passenger_count,
    COUNT(*) AS cnt
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/parquet/taxi/\*/\*/\*',
        FORMAT='PARQUET'
    ) WITH (
        pickup_datetime DATETIME2,
        passenger_count INT
    ) AS nyc
GROUP BY
    passenger_count,
    YEAR(pickup_datetime)
ORDER BY
    YEAR(pickup_datetime),
    passenger_count
  
```

| | (No column name) | passenger_count | cnt |
|---|------------------|-----------------|----------|
| 1 | 2016 | 0 | 2557 |
| 2 | 2016 | 1 | 43735845 |
| 3 | 2016 | 2 | 9056714 |
| 4 | 2016 | 3 | 2610541 |
| 5 | 2016 | 4 | 1309639 |
| 6 | 2016 | 5 | 3086097 |
| 7 | 2016 | 6 | 1956607 |

SQL On Demand – Creating views

Overview

Create views using SQL On Demand queries

Benefits

Works same as standard views

```
USE [mydbname]
GO

IF EXISTS(select * FROM sys.views where name = 'populationView')
DROP VIEW populationView
GO

CREATE VIEW populationView AS
SELECT *
FROM OPENROWSET(
    BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [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 |

SQL On Demand – Creating views

The screenshot shows three windows from the Microsoft Azure Synapse Analytics studio:

- Top Left Window:** A query editor titled "SQL script i *". It contains the following T-SQL code:


```

1 -- type your sql script here, we now have intellisense
2 CREATE VIEW yellow_2017 AS
3 SELECT *
4 FROM
5 OPENROWSET(
6   BULK 'https://prlangaddemosa.dfs.core.windows.net/nyctlc/yellow/puYear=2017/*/*',
7   FORMAT='PARQUET'
8 ) AS nyc
9 
```
- Bottom Left Window:** A query editor titled "SQL script i *". It contains the same T-SQL code as the top window. Below the code, the status message "00:00:17 Query executed successfully." is displayed.
- Bottom Right Window:** A chart viewer titled "Chart". It displays a line chart with "passengerCount" on the X-axis (ranging from 0 to 10) and "TBL" on the Y-axis (ranging from 0 to 10M). The chart shows a single data series represented by green dots connected by a line. The legend indicates "passengerCount" (blue dot) and "tbl" (green dot).

The screenshot shows a Microsoft Azure Synapse Analytics studio interface with the following details:

- Top Bar:** Shows "Microsoft Azure | Synapse Analytics > prlangadws2". The "Connect to" dropdown is highlighted with a red box and set to "SQL Analytics on-demand".
- Query Editor:** Displays the same T-SQL code as the other windows:


```

1 SELECT
2   YEAR(tpcepPickupDateTime),
3   passengerCount,
4   COUNT(*) AS cnt
5 FROM
6   yellow_2017
7 GROUP BY
8   passengerCount,
9   YEAR(tpcepPickupDateTime)
10 ORDER BY
11   YEAR(tpcepPickupDateTime),
12   passengerCount

```
- Results View:** Shows the results of the query in a "Table" format. The data is as follows:

| (NO COLUMN NAME) | PASSENGERCOUNT | CNT |
|------------------|----------------|----------|
| 2017 | 0 | 166086 |
| 2017 | 1 | 81034075 |
| 2017 | 2 | 16545571 |
| 2017 | 3 | 4748869 |
| 2017 | 4 | 2257813 |
| 2017 | 5 | 5407319 |
- Message:** Below the results, the message "00:02:19 Query executed successfully." is shown.

SQL On Demand – Querying JSON files

Overview

Read JSON files and provides data in tabular format

Benefits

Supports OPENJSON, JSON_VALUE and JSON_QUERY functions

```
SELECT *
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/json/books/book
1.json',
        FORMAT='CSV',
        FIELDTERMINATOR = '0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
```

| jsonContent |
|--|
| 1 {"_id": "kim95", "type": "Book", "title": "Modern Databas... |

SQL On Demand – Querying JSON files

Example of JSON_VALUE function

```

SELECT
    JSON_VALUE(jsonContent, '$.title') AS title,
    JSON_VALUE(jsonContent, '$.publisher') AS publisher,
    jsonContent
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/json/books/*.json',
        FORMAT='CSV',
        FIELDTERMINATOR = '0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
WHERE
    JSON_VALUE(jsonContent, '$.title') = 'Probabilistic and Statistical Methods in Cryptology, An Introduction by Selected Topics'

```

| | title | publisher | jsonContent |
|---|---|-----------|---------------------|
| 1 | Probabilistic and Statistical Methods in Cryptology, An Introduction by Selected Topics | Springer | {"_id": "neuen...", |

Example of JSON_QUERY function

```

SELECT
    JSON_QUERY(jsonContent, '$.authors') AS authors,
    jsonContent
FROM
    OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/json/books/*.json',
        FORMAT='CSV',
        FIELDTERMINATOR = '0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
WHERE
    JSON_VALUE(jsonContent, '$.title') = 'Probabilistic and Statistical Methods in Cryptology, An Introduction by Selected Topics'

```

| | authors | jsonContent |
|---|---------------------------|---|
| 1 | ["Daniel Neuenschwander"] | {"_id": "neuenschwander04", "type": "Book", "title": "Probabi..." |

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!nfo'
;

-- 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 );
```

SQL scripts > View and export results

The screenshot shows the Azure Synapse Analytics Studio interface. On the left, a code editor displays a T-SQL script for reading a CSV file from Azure Data Lake Storage into a temporary table named 'searchlog'. The script uses the OPENROWSET function with BULK and FORMAT='CSV' parameters. The temporary table has columns: id (int), [time] (datetime), region (varchar(50)), searchtext (varchar(200)), latency (int), links (varchar(500)), and clickedlinks (varchar(500)). A comment indicates it's AS 'searchlog';.

The main area shows the results of the query. There are two tabs: 'Results' and 'Messages'. The 'Results' tab is selected, displaying the data in a table format. The table has columns: ID, TIME, REGION, SEARCHTEXT, LATENCY, LINKS, and CLICKEDLINKS. The data includes:

| ID | TIME | REGION | SEARCHTEXT | LATENCY | LINKS | CLICKEDLINKS |
|--------|-----------------------------|--------|---------------------|---------|-----------------------|---------------------|
| 399266 | 2019-10-15T11:53:04.0000000 | en-us | how to make nachos | 73 | www.nachos.com;... | NULL |
| 382045 | 2019-10-15T11:53:25.0000000 | en-gb | best ski resorts | 614 | skiresorts.com;ski... | ski-europe.com;w... |
| 382045 | 2019-10-16T11:53:42.0000000 | en-gb | broken leg | 74 | mayoclinic.com/h... | mayoclinic.com/h... |
| 106479 | 2019-10-16T11:53:10.0000000 | en-ca | south park episodes | 24 | southparkstudios... | southparkstudios... |
| 906441 | 2019-10-16T11:54:18.0000000 | en-us | cosmos | 1213 | cosmos.com;wiki... | NULL |

A message at the bottom states: "00:00:35 Query executed successfully."

On the right, there is a 'Results' tab and a 'Messages' tab. The 'Results' tab is selected. Below it is a 'View' dropdown with 'Table' (selected) and 'Chart' options. To the right of the table is an 'Export results' button with a dropdown menu showing options: CSV, Excel, JSON, XML, and REGIO. The 'REGIO' option is highlighted.

SQL scripts > View results (chart)

SearchLog_que... X

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

```
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
```

Results Messages

View Table Chart

| Index | Value (approx.) |
|-------|-----------------|
| 0 | 400k |
| 1 | 400k |
| 2 | 400k |
| 3 | 100k |
| 4 | 900k |
| 5 | 350k |
| 6 | 600k |
| 7 | 300k |
| 8 | 450k |
| 9 | 350k |
| 10 | 350k |
| 11 | 600k |
| 12 | 350k |
| 13 | 800k |
| 14 | 550k |
| 15 | 800k |
| 16 | 600k |
| 17 | 1000k |
| 18 | 350k |
| 19 | 600k |
| 20 | 350k |
| 21 | 650k |
| 22 | 650k |

Chart type: Line
X axis column:
Y axis columns: id, time, region, searchtext, latency, li...
Legend position: center - bottom

00:00:35 Query executed successfully.

Convert from CSV to Parquet on-demand

```
/*
CREATE EXTERNAL DATA SOURCE [CsvDataSource] WITH (
    LOCATION = 'https://showdemoweu.dfs.core.windows.net/data'
)

CREATE EXTERNAL FILE FORMAT [ParquetFF] WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
);
*/

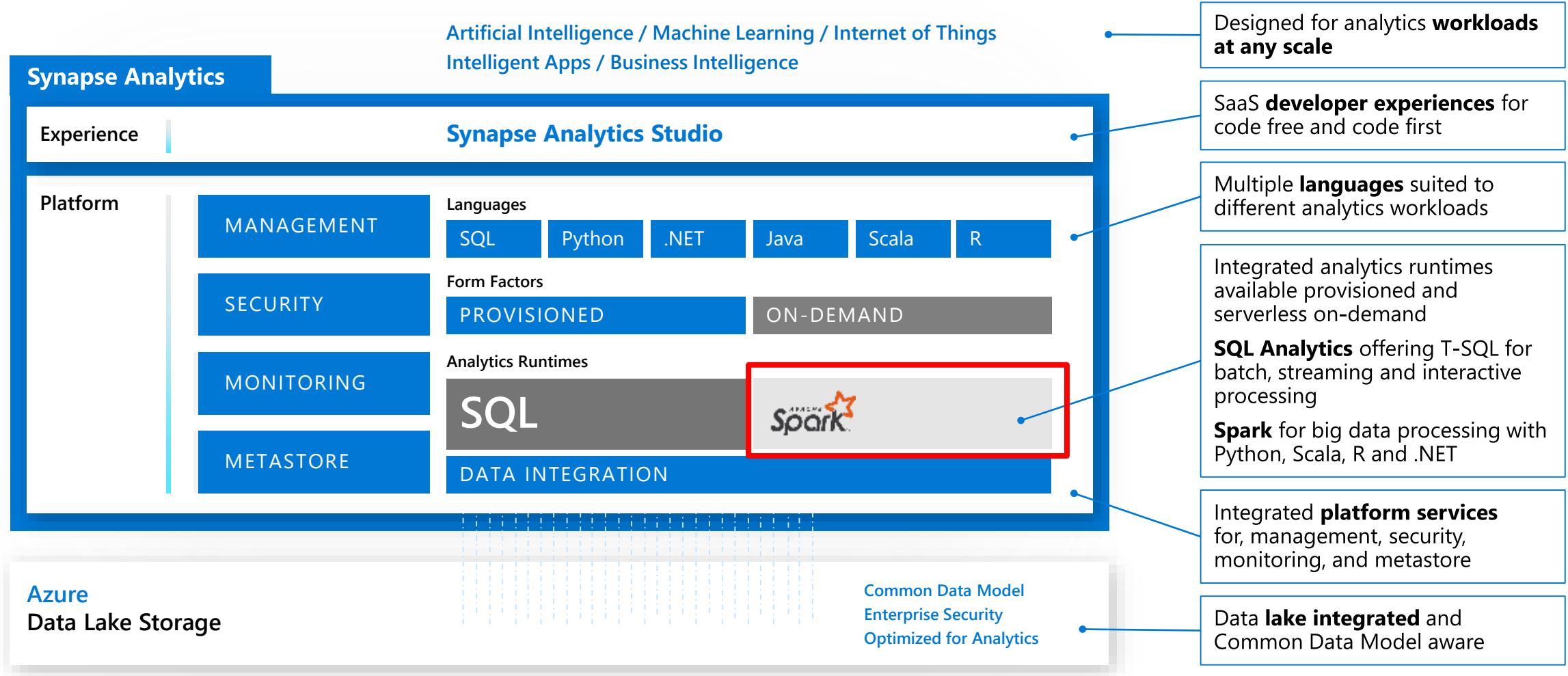
CREATE EXTERNAL TABLE [dbo].[Populationv8] WITH (
    LOCATION = 'populationConvertedv3/',
    DATA_SOURCE = [CsvDataSource],
    FILE_FORMAT = [ParquetFF]
) AS
SELECT
    *
FROM
    OPENROWSET(
        BULK 'https://showdemoweu.dfs.core.windows.net/data/population\_csv/population.csv',
        FORMAT='CSV'
    ) WITH (
        CountryCode varchar(4),
        CountryName varchar(64),
        Year int,
        PopulationCount int
    ) AS r;
```



Azure Synapse Analytics Spark

Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Azure Synapse Apache Spark - Summary



- **Apache Spark 2.4 derivation**
 - Linux Foundation Delta Lake 0.4 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 interact 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

Languages

Overview

Supports multiple languages to develop notebook

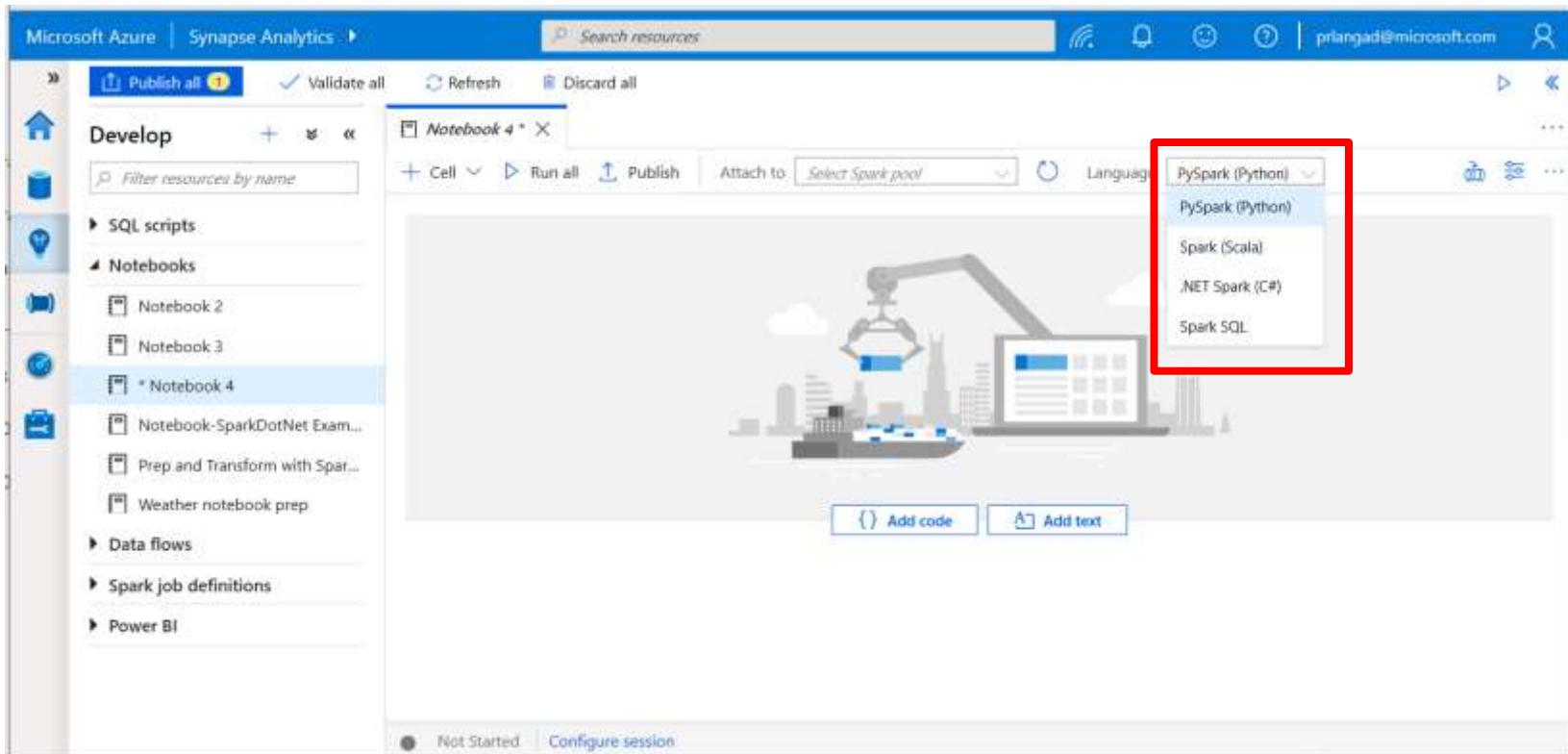
- PySpark (Python)
- Spark (Scala)
- .NET Spark (C#)
- Spark SQL
- Java
- R (early 2020)

Benefits

Allows to write multiple languages in one notebook

%%<Name of language>

Offers use of temporary tables across languages



Notebooks > Configure Session

The screenshot shows the Azure Synapse Analytics Studio interface. On the left, there is a preview area with a robotic arm icon. Below it are buttons for 'Add code' and 'Add text'. At the top, there are buttons for 'Cell', 'Run all', 'Publish', 'Attach to', 'Select Spark pool', 'Language' (set to PySpark (Python)), and a three-dot menu. On the right, the 'Properties' panel is open under the 'General' tab. It contains the following information:

- Name:** Notebook 1
- Description:** (empty)
- Type:** ipynb notebook
- Size:** 109 bytes
- Notebook settings:** Include cell output when saving
- Session:** [Configure session](#) (highlighted in blue), [View session details](#)

At the bottom, there are tabs for 'Not Started' and 'Configure session'.

Configure session

BOOT_Basic_spark

* Session timeout

* Executors

* Executor size

* Driver size

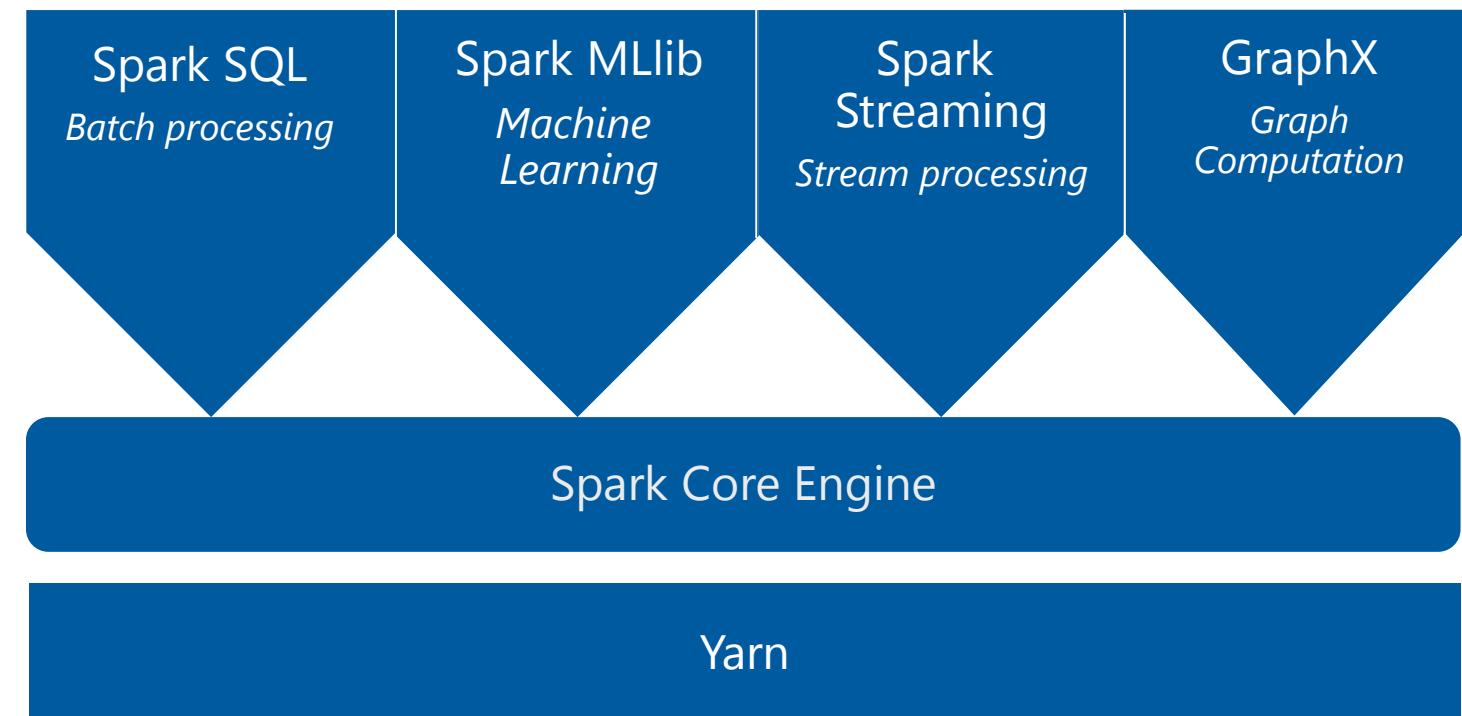
[Apply](#) [Cancel](#)

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

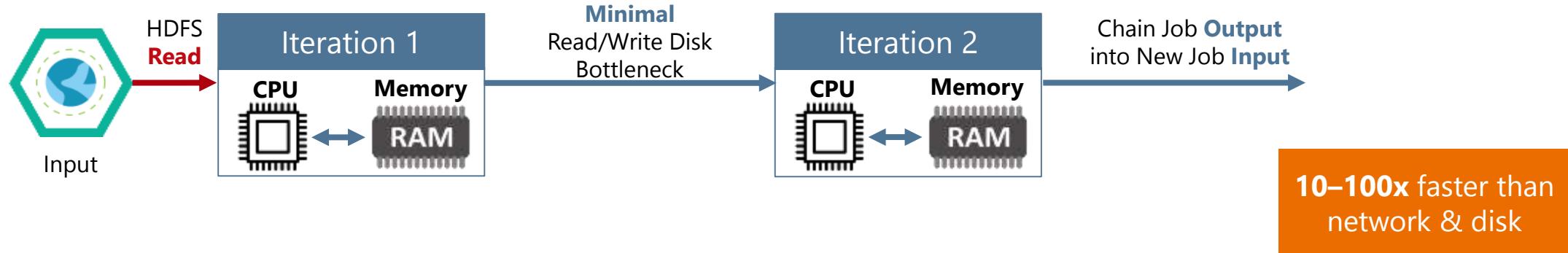


Motivation for Apache Spark

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

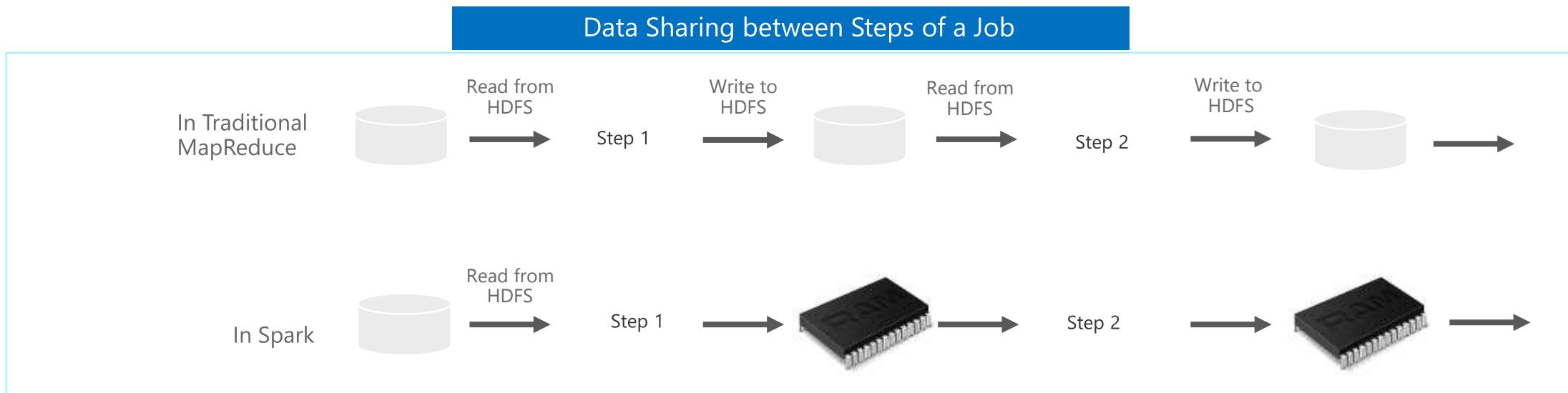


Solution: Keep data **in-memory** with a new distributed execution engine



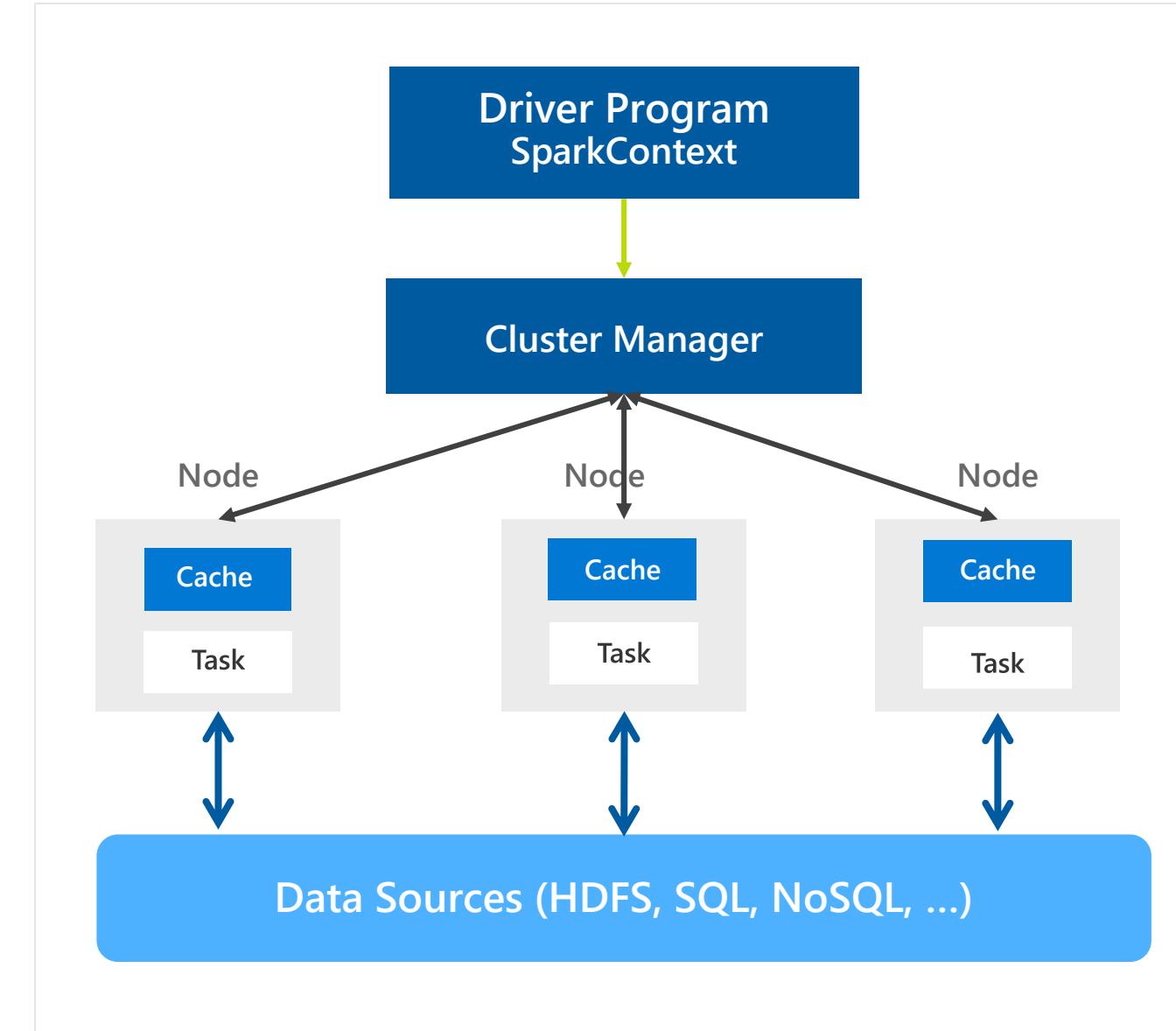
What makes Spark fast

- **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

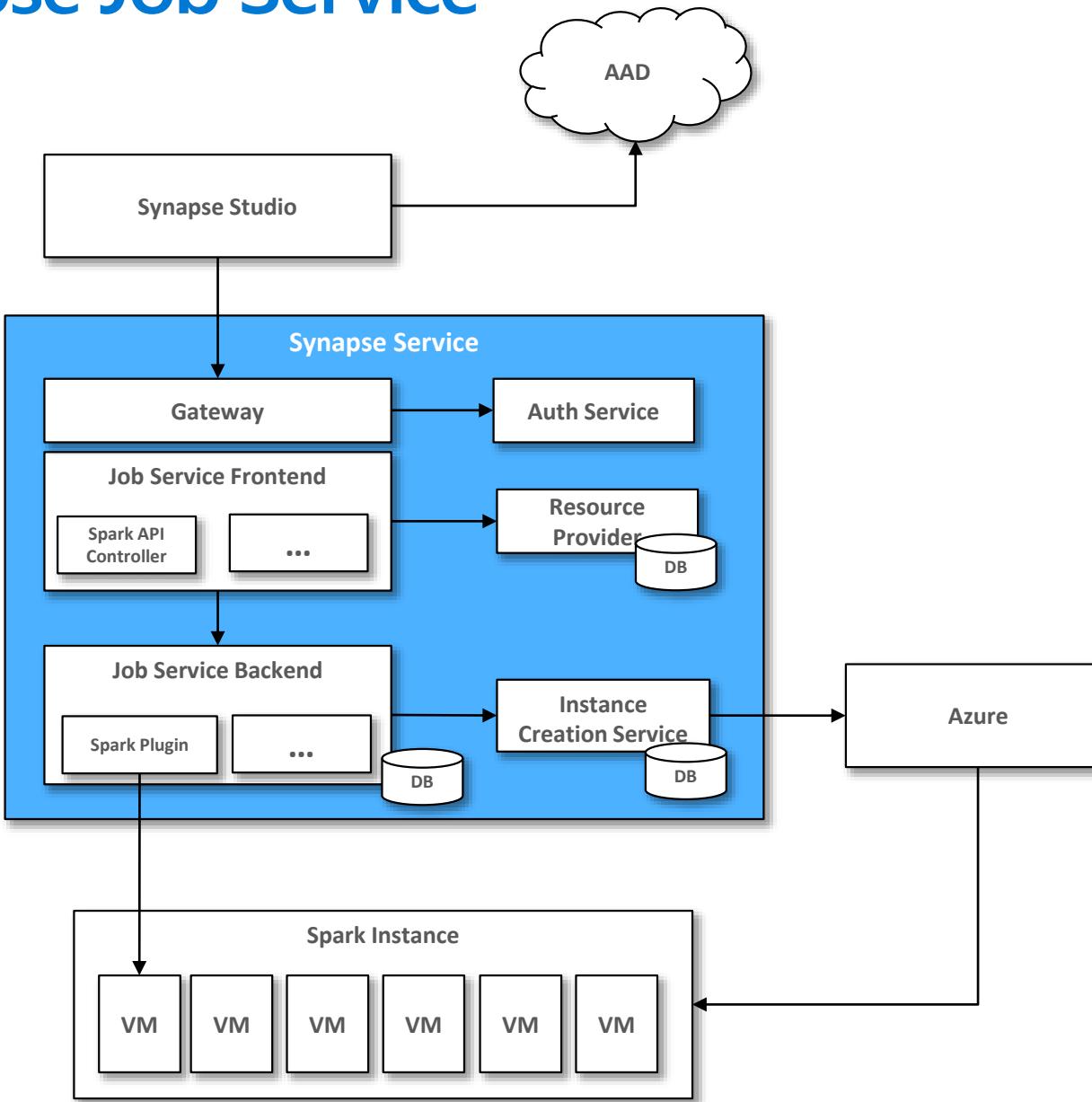
- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.



Azure Synapse Apache Spark

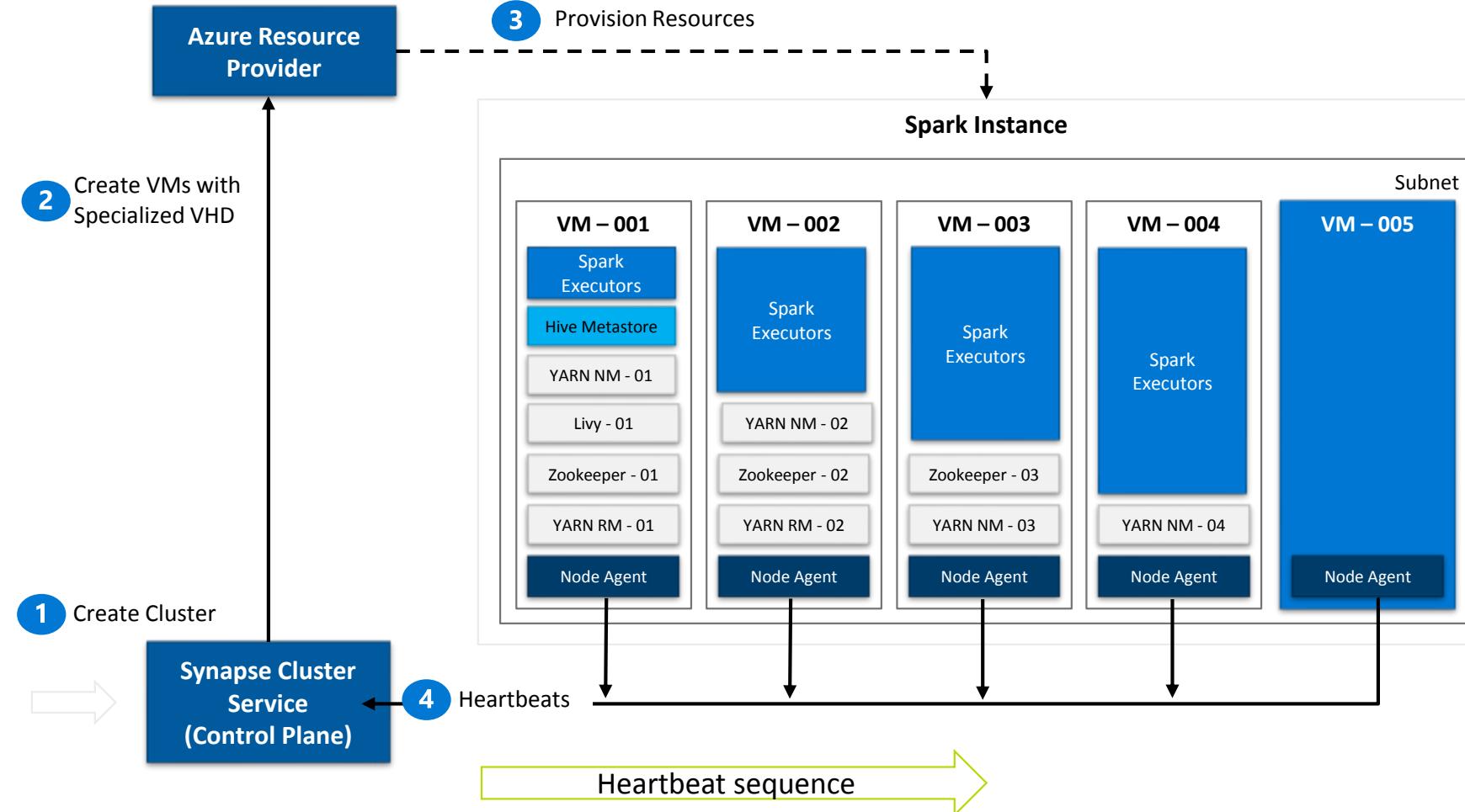
Architecture Overview

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 > Synapse workspaces > euang-synapse-nov-ws - Apache Spark pools > 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.

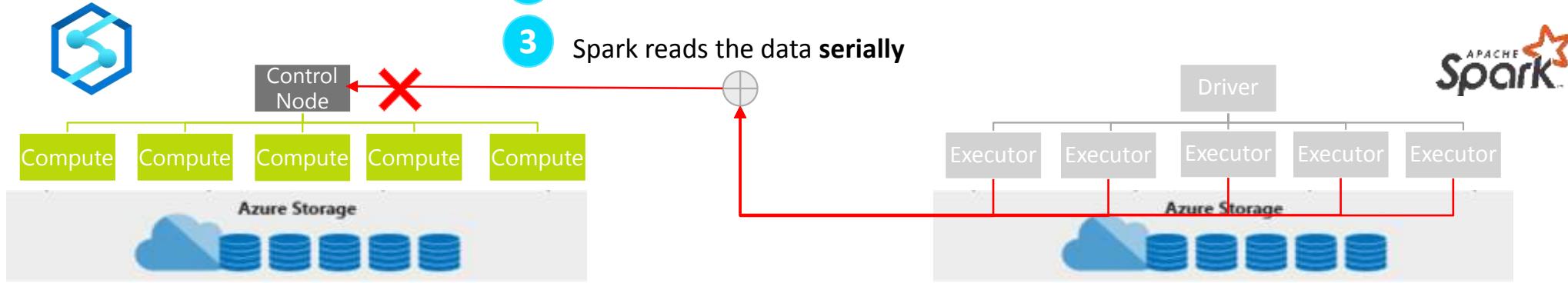
| | |
|-----------------------|-----------|
| Apache Spark * | 2.4 |
| Python | 3.6.1 |
| Scala | 2.11.12 |
| Java | 1.8.0_222 |
| .NET Core | 3.0 |
| .NET for Apache Spark | 0.6.0 |
| Delta Lake | 0.4.0 |

Packages

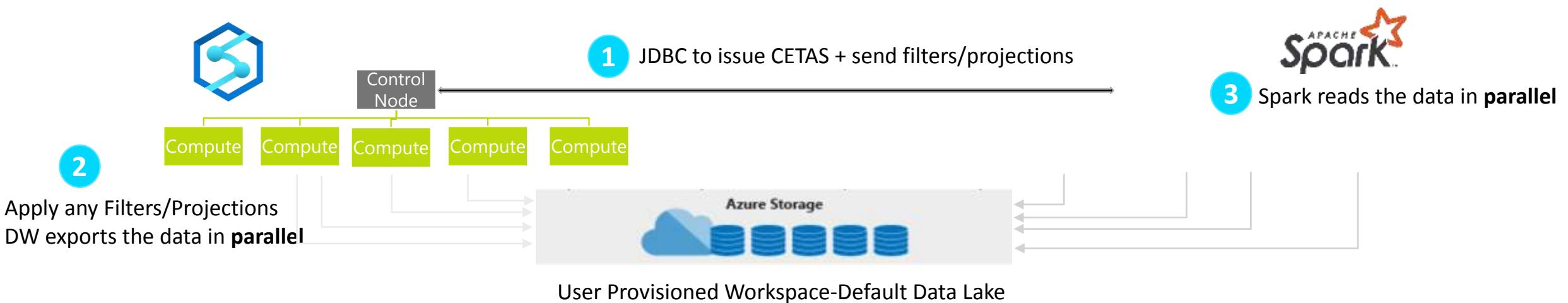
Upload environment configuration file ("PIP freeze" output).

File upload Upload

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

Create Notebook on files in storage

The screenshot illustrates the process of creating a new notebook from data stored in Azure Storage.

Left Panel (Storage View):

- Shows a list of Storage accounts: prlangaddemosa (Primary), nyctic, and others.
- The nyctic account is selected.
- A parquet file named "part-00133-tid-216-69f72c50b05d-15253-1.c000.snappy.parquet" is highlighted.
- A context menu is open over the file, with the "New notebook" option highlighted by a red box.

Bottom Panel (Spark History Server):

- The "nyctic" workspace is selected.
- A new notebook titled "New notebook" is open in the center.
- The code cell contains the following PySpark command:

```
$ pyspark  
data_path = spark.read.load('abfss://nyctic@prlangaddemosa.dfs.core.windows.net/yellow/puYear=2015/puMonth=3/part-00133-tid-216-69f72c50b05d-15253-1.c000.snappy.parquet')
```

- The cell status is "Succeeded".
- The job execution details show three executors (Job 0, Job 1, Job 2) completed successfully.
- The data preview shows a table with columns: vendorID, tpepPickupDateTime, tpepDropoffDateTime, passengerCount, tripDistance, puLocationId, doLocationId, ratecodeID, storeAndFlag, paymentType, fareAmount, extra, surtax, tipAmount, tollsAmount, totalAmount.
- The data preview displays several rows of taxi trip data.

Microsoft Azure Synapse Analytics > euang-synapse-now-ws

Develop Publish all Validate all Refresh Discard all

Data Download * NYCTaxi_Docs... * SeattleSafetyD... * Repo *

Cell 1

```

1 # Azure storage access info
2 blob_account_name = "azuresynapsedatostorage"
3 blob_container_name = "citydatacontainer"
4 blob_relative_path = "Safety/Release/city=Seattle"
5 blob_sas_token = r""

6 # Allow SPARK to read from Blob remotely
7 wasbs_path = 'wasbs://'+blob_account_name+'.blob.core.windows.net/'+blob_container_name+'/'+blob_relative_path
8 spark.conf.set('fs.azure.sas.%s.blob.core.windows.net' % (blob_container_name, blob_account_name), blob_sas_token)
9
10 # SPARK read parquet, note that it won't load any data yet
11 seattlesafety_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 (active) | | 11/22/2019 12:44:46 AM | 9m54s |

View in monitoring Spark history server

Cell 2

```

1 seattlesafety_df.createOrReplaceTempView('seattlesafety')

```

Command executed in 2s 830ms by euang on 11-22-2019 00:44:57.321 -08:00

Cell 3

```

[6] 1 display(spark.sql("SELECT * FROM seattlesafety LIMIT 10"))

```

Command executed in 12s 907ms by euang on 11-22-2019 00:45:07.212 -08:00

View Table Chart

| datatype | dataSubtype | dateTime | category | address | latitude | longitude |
|----------|-------------|--------------------------|---------------------------|--------------------------|-----------|-------------|
| Safety | 911_Fire | 2011-03-04T10:00:26.000Z | Aid Response | 517 3rd Av | 47.602172 | -122.330863 |
| Safety | 911_Fire | 2015-06-08T02:09:35.000Z | Trans to AMR | 10044 65th Av S | 47.511314 | -122.292346 |
| Safety | 911_Fire | 2015-06-08T21:10:52.000Z | Aid Response | Aurora Av N / N 125th St | 47.719572 | -122.344057 |
| Safety | 911_Fire | 2007-09-17T13:03:34.000Z | Medic Response | 1st Av N / Republican St | 47.623272 | -122.355415 |
| Safety | 911_Fire | 2007-11-19T17:46:57.000Z | Aid Response | 7724 Ridge Dr Ne | 47.684393 | -122.275254 |
| Safety | 911_Fire | 2008-06-15T14:32:33.000Z | Medic Response | 6940 62nd Av Ne | 47.678769 | -122.262227 |
| Safety | 911_Fire | 2007-06-18T23:05:58.000Z | Medic Response | 5107.5 Myrtle St | 47.538902 | -122.268829 |
| Safety | 911_Fire | 2005-06-08T19:25:10.000Z | Aid Response | 532 Belmont Av E | 47.623505 | -122.324033 |
| Safety | 911_Fire | 2017-03-06T19:45:36.000Z | Trans to AMR | 610 1st Av N | 47.624659 | -122.355403 |
| Safety | 911_Fire | 2017-06-25T16:21:21.000Z | Automatic Fire Alarm Read | 7711 8th Av Ne | 47.685157 | -122.368006 |

Cell 4

```

[7] 1 seattlesafety_df.coalesce(1).write.csv('abfss://default@euangsynapsestorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')

```

View results in
table format



Microsoft Azure Synapse Analytics > euang-synapse-nov-ws

Develop Language: PySpark (Python) PySpark (Python)

Cell 1

```
[3] 1 # Azure storage access info
2 blob_account_name = "azuresynapsesas"
3 blob_container_name = "citydatacontainer"
4 blob_relative_path = "Safety/Release/city=Seattle"
5 blob_sas_token = r""

6 # Allow SPARK to read from Blob remotely
7 wasbs_path = "wasbs://<blob_container_name>.<blob_account_name>.blob.core.windows.net/<blob_relative_path>"
spark.conf.set( 'fs.azure.sas.<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 2min 10s 412ms by euang on 11-22-2019 00:14:32.415 -08:00
```

Job execution in progress: Spark 1 executors 4 cores

| ID | DESCRIPTION | STATUS | STAGES | TAGS | SUBMISSION TIME | DURATION |
|-------|---|-------------|----------------|------|-------------------------|----------|
| Job 0 | parquet at NativeMethodAccessorsImpl.java:0 | In progress | 0/1 (1 active) | | 11/22/2019, 12:44:46 AM | 13m43s |

Cell 2

```
[4] 1 seasafety_df.createOrReplaceTempView("seattlesafety")
```

Command executed in 2s 839ms by euang on 11-22-2019 00:15:37.321 -08:00

Cell 3

```
[5] 1 display(spark.sql('SELECT * FROM seattlesafety'))
```

Command executed in 11s 320ms by euang on 11-22-2019 00:38:21.241 -08:00

View Table Chart

SQL support

View results in chart format

A pie chart showing the distribution of various incident types. The largest category is Aid Response (blue), followed by Medic Response (green), and then several smaller categories like Automatic Fire Alarm False, Medic Response 7 per Rule, and Auto Response Yellow.

| Category | Count |
|-------------------------------|-------|
| Aid Response | ~450 |
| Medic Response | ~150 |
| Automatic Fire Alarm False | ~10 |
| Medic Response 7 per Rule | ~10 |
| Auto Response Yellow | ~10 |
| AVVI - Motor Vehicle Incident | ~10 |
| Medic Response 6 per Rule | ~10 |
| Motor Vehicle Accident | ~10 |
| Automatic Medical Alarm | ~10 |
| TRED 1 Unit | ~10 |
| Auto Fire Alarm | ~10 |
| Automatic Fire Alarm Reset | ~10 |

Chart type: pie chart
X axis column: longitude
Y axis column: longitude
Aggregation: COUNT
Y axis label: Time
X axis label: category

Cell 4

```
[6] 1 seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapse-nov-ws.dfs.core.windows.net/desodata/seattlesafety', mode='overwrite')
```

Microsoft Azure | Synapse Analytics | euang-synapse-nov-2023 | Search resources

Develop + ×

Data Download... NYCTaxi_Docs_...

Cell 1 Run all Publish Attach to Enter Spark session Language PySpark (Python)

```
11 # Creating a temp table allows easier manipulation during the session, they are not persisted between sessions;
12 # for that write the data to storage like above;
13 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 focused on tip values and relationships.

Cell 8

```
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 = 0.1, 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
```

Tip amount distribution

Tip amount by Passenger count

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

Libraries will get installed for your Spark pool during cluster creation

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

In the Portal

Specify the new requirements while creating Spark Pool in Additional Settings blade

Microsoft Azure (Preview) [Restore default configuration](#) [Report a bug](#) [Search resources, services, and data](#)

Home > nushuklasynapsewestus2 > Create Apache Spark pool

Create Apache Spark pool

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.

| Apache Spark * | 2.4 |
|-----------------------|-----------|
| Python | 3.6.1 |
| Scala | 2.11.12 |
| Java | 1.8.0_222 |
| .NET Core | 3.0 |
| .NET for Apache Spark | 0.6.0 |
| Delta Lake | 0.4.0 |

Packages

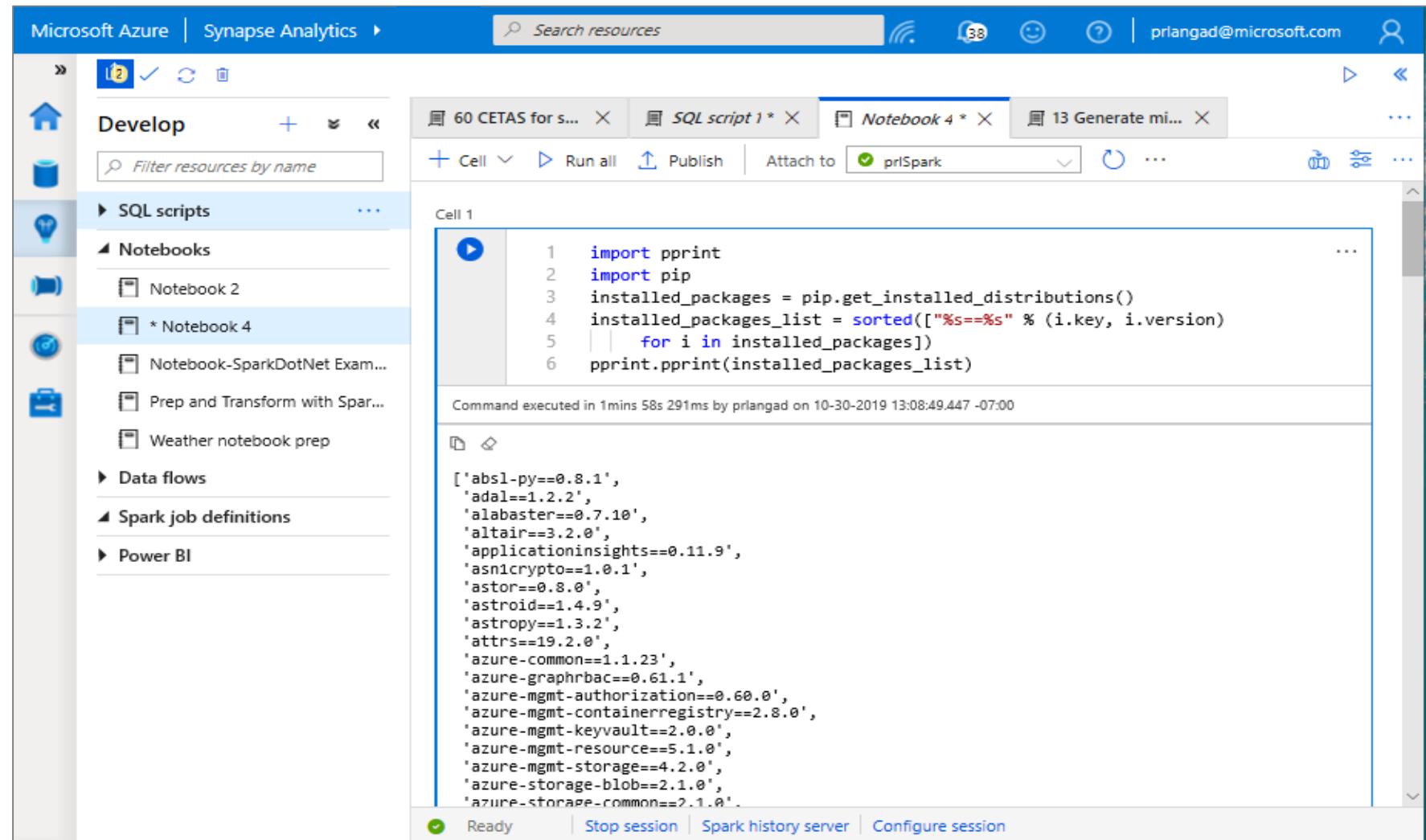
Upload environment configuration file ("PIP freeze" output).

File upload "requirements.txt"

[Review + create](#) [< Previous](#) [Next: Tags >](#)

Library Management - Python

Get list of installed libraries with version information



The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, the 'Develop' sidebar is open, showing a list of resources: SQL scripts, Notebooks, Data flows, Spark job definitions, and Power BI. A 'Notebook 4' item is selected. In the main workspace, there are four tabs at the top: '60 CETAS for s...', 'SQL script 1 * X', 'Notebook 4 * X' (which is active), and '13 Generate mi... X'. Below the tabs, a toolbar includes 'Cell', 'Run all', 'Publish', 'Attach to' (set to 'priSpark'), and other options. A code cell labeled 'Cell 1' 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)
```

The output of the cell shows a long list of installed Python packages and their versions:

```
['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']
```

At the bottom of the cell, it says 'Command executed in 1mins 58s 291ms by prlangad on 10-30-2019 13:08:49.447 -07:00'. The status bar at the bottom of the interface shows 'Ready'.

Spark ML Algorithms

Spark ML Algorithms

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

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

 Azure/mmlspark

Synapse Notebook: Connect to AML workspace

The screenshot shows a Microsoft Azure Synapse Analytics notebook interface. The left sidebar shows 'Develop' resources: SQL scripts, Notebooks, Data flows, Spark job definitions, and Power BI. The main area has a title bar with 'Search resources' and user info 'balapv@microsoft.com'. Below is a toolbar with 'Cell', 'Run all', 'Publish', 'Attach to', 'Language' set to 'PySpark Python', and other controls.

Check the Azure ML Core SDK Version to Validate Your Installation

Cell 3

```
[9] 1 import azureml.core  
2 print("SDK Version:", azureml.core.VERSION)
```

Command executed in 1s 258ms by balapv on 11-12-2019 14:41:52.805 -00:00

SDK Version: 1.0.69

Connect to Azure Workspace

Cell 5

```
[6] 1 ## Import the Workspace class and check the Azure ML SDK version.  
2 from azureml.core import Workspace  
3  
4 ws = 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')
```

Command executed in 3s 909ms by balapv on 11-12-2019 14:41:55.491 -00:00

AML-WS-synapse westus2 balapv-synapse-rg

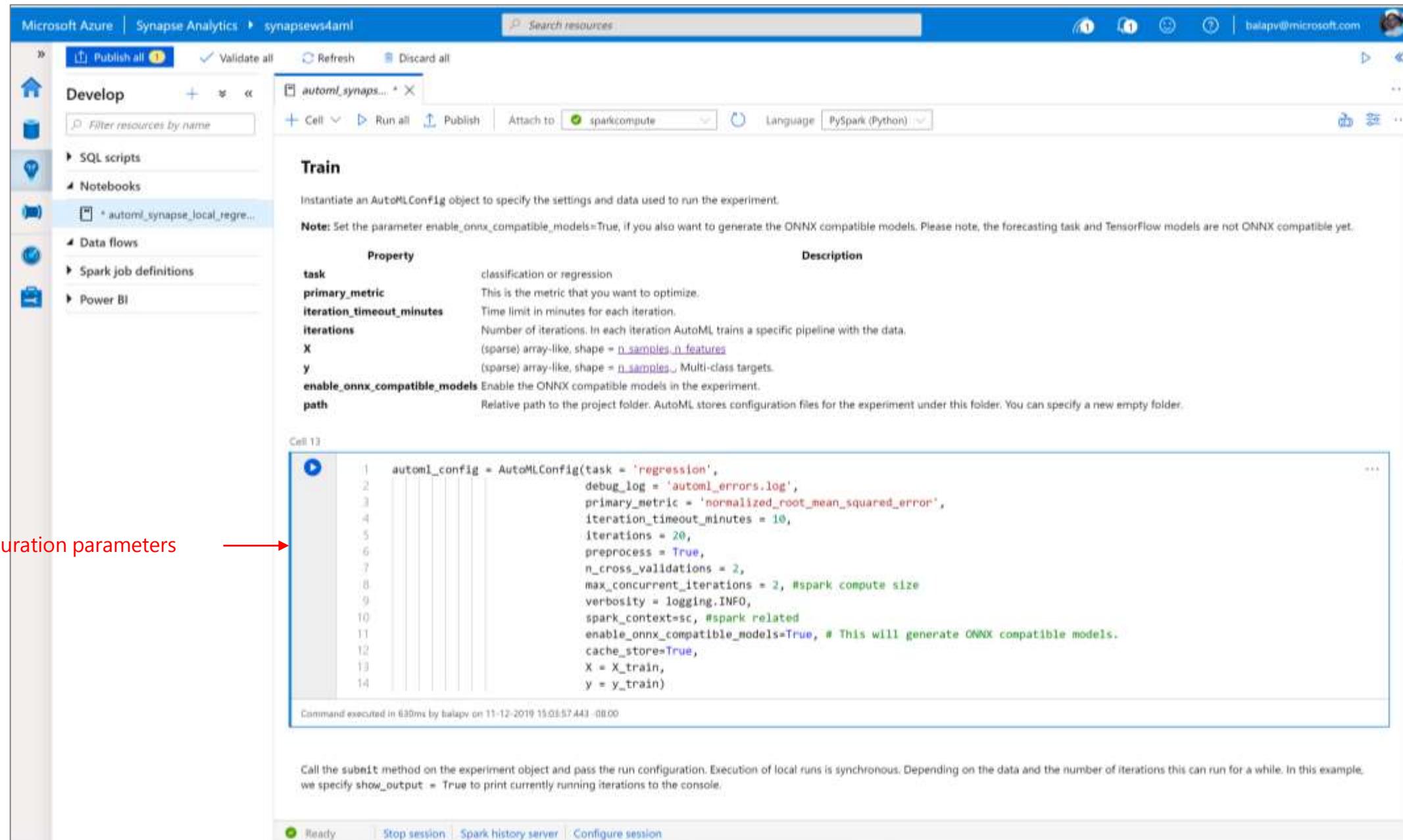
Cell 6

```
[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
```

Running Stop session Spark history server Configure session

Simple code to connect workspace →

Synapse Notebook: Configure AML job to run on Synapse



The screenshot shows the Microsoft Azure Synapse Analytics notebook interface. The left sidebar shows a navigation tree with 'Develop' selected, followed by 'SQL scripts', 'Notebooks' (with 'automl_synapse_local_regr...' highlighted), 'Data flows', 'Spark job definitions', and 'Power BI'. The main area has tabs for 'Train' and 'Test'. The 'Train' tab contains documentation for the AutoMLConfig object, including properties like task (classification or regression), primary_metric (the metric to optimize), iteration_timeout_minutes (time limit per iteration), iterations (number of iterations), X (input features), y (target values), enable_onnx_compatible_models (for ONNX compatibility), and path (relative project folder). Below this is a code cell labeled 'Cell 13' containing Python code to initialize an AutoMLConfig object for regression with specific parameters. 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.

Ready | Stop session | Spark history server | Configure session

Synapse Notebook: Run AML job

Microsoft Azure | Synapse Analytics > synapsews4aml | Search resources | Show notifications | balapv@microsoft.com

Publish all (1) | Validate all | Refresh | Discard all

Develop | Filter resources by name

SQL scripts | Notebooks | automl_synapse_local_regression.ipynb * X

Cell | Run all | Publish | Attach to sparkcompute | Language: PySpark (Python)

Run AutoML job

Cell 15

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

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 | 0.0184 | 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 |

Get Azure Portal URL for Monitoring Runs

Running | Stop session | Spark history server | Configure session

ML job execution result

**Industry-leading security
and compliance**

Enterprise-grade security



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



Section 508 VPAT



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

Comprehensive Security

| Category | Feature | |
|-------------------|--------------------------------------|---|
| Data Protection | Data in Transit | ✓ |
| | Data Encryption at Rest | ✓ |
| | Data Discovery and Classification | ✓ |
| Access Control | Object Level Security (Tables/Views) | ✓ |
| | Row Level Security | ✓ |
| | Column Level Security | ✓ |
| Authentication | Dynamic Data Masking | ✓ |
| | SQL Login | ✓ |
| | Azure Active Directory | ✓ |
| Network Security | Multi-Factor Authentication | ✓ |
| | Virtual Networks | ✓ |
| | Firewall | ✓ |
| Threat Protection | Azure ExpressRoute | ✓ |
| | Thread Detection | ✓ |
| | Auditing | ✓ |
| | Vulnerability Assessment | ✓ |

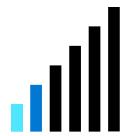


Threat Protection - Business requirements



How do we enumerate and track potential SQL vulnerabilities?

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



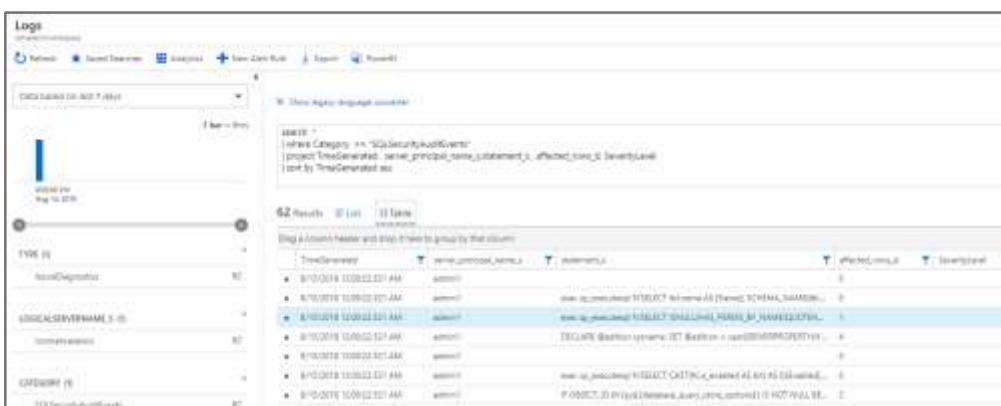
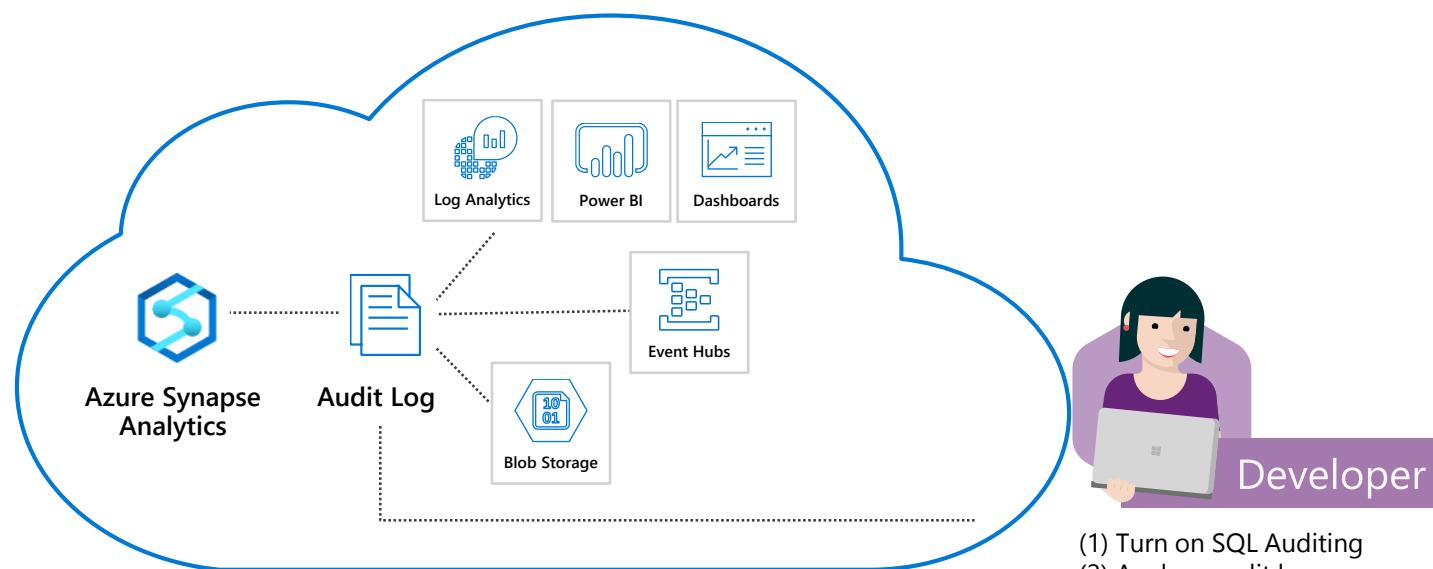
How do we discover and alert on suspicious database activity?

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



SQL auditing in Azure Log Analytics and Event Hubs

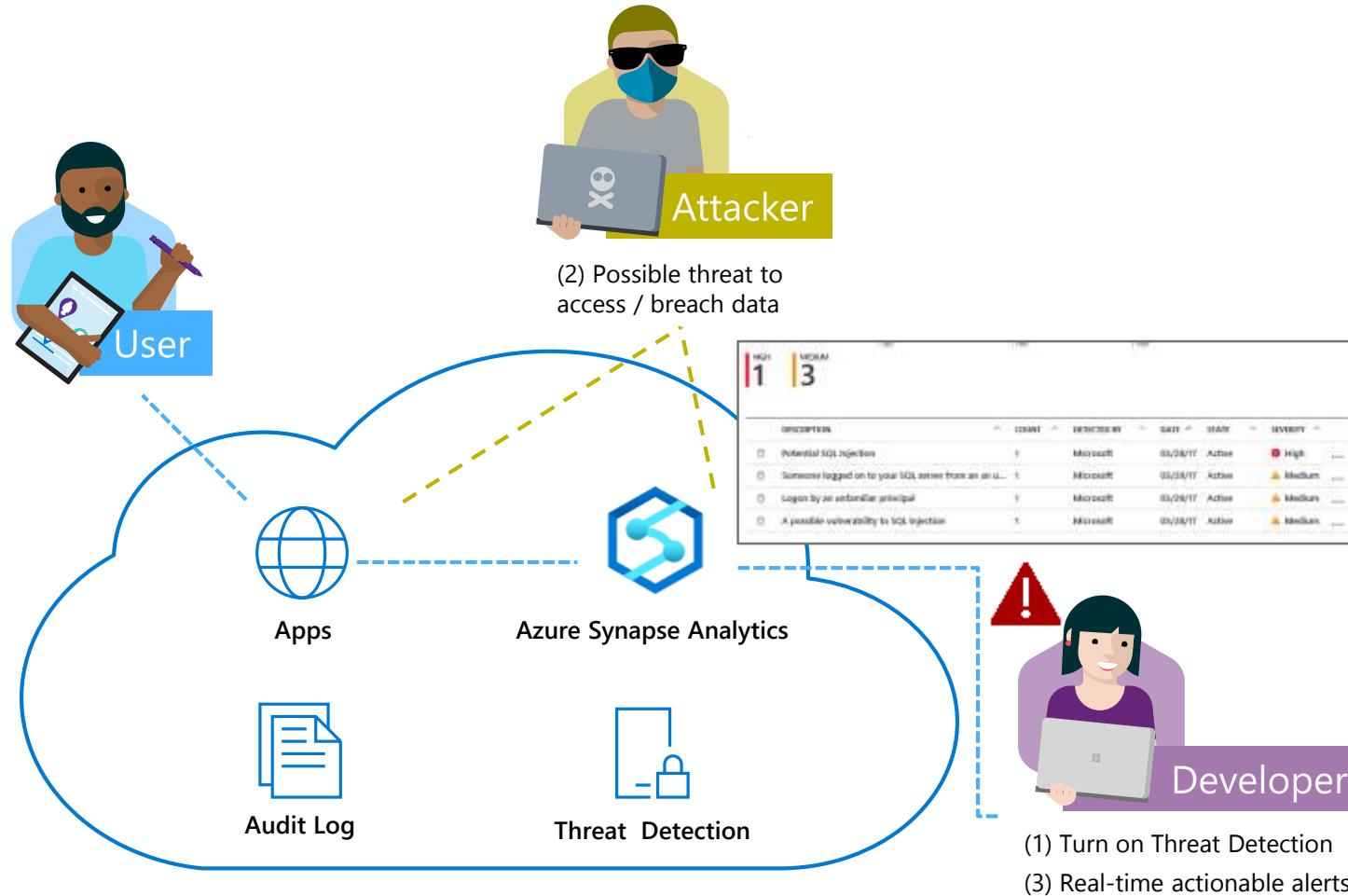
Gain insight into database audit log



- ✓ 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

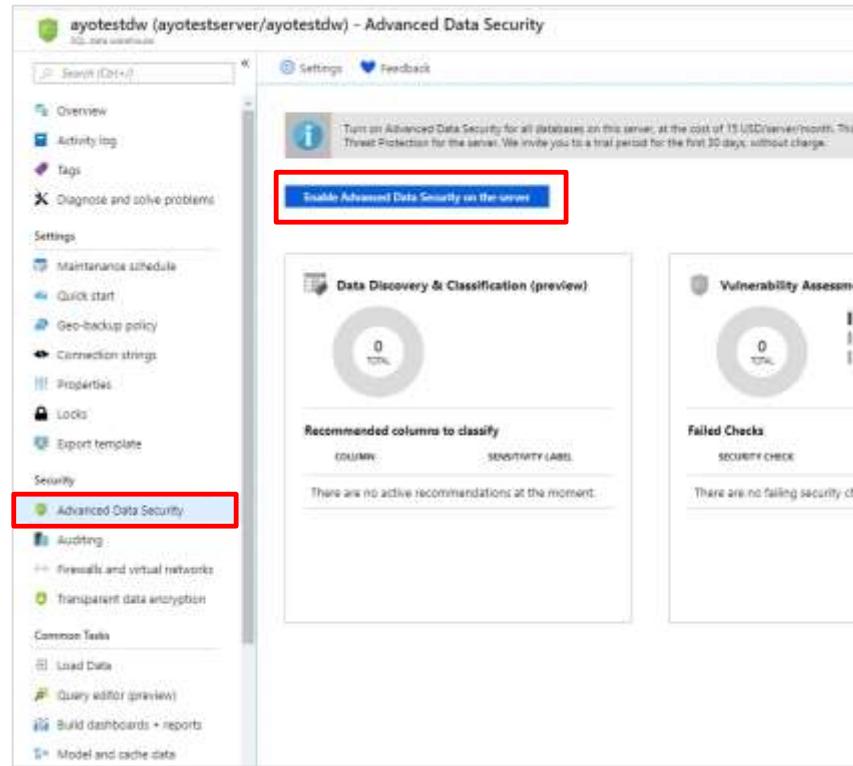
Discover, classify, protect and track access to sensitive data

The screenshot shows the Azure portal interface for SQL Data Discovery & Classification. The main dashboard displays two donut charts: one for 'Label distribution' (10 columns) and another for 'Information type distribution' (10 columns). Below these charts, there's a table with columns: Schema, Table, Column, Information Type, and Sensitivity Label. A separate window titled 'Settings - Information protection' is open, showing a list of sensitivity labels with their descriptions. The labels include 'General', 'Confidential', 'Confidential - GDPR', 'Highly confidential', and 'Highly confidential - GDPR'. The 'Confidential' label is selected.

- ✓ 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

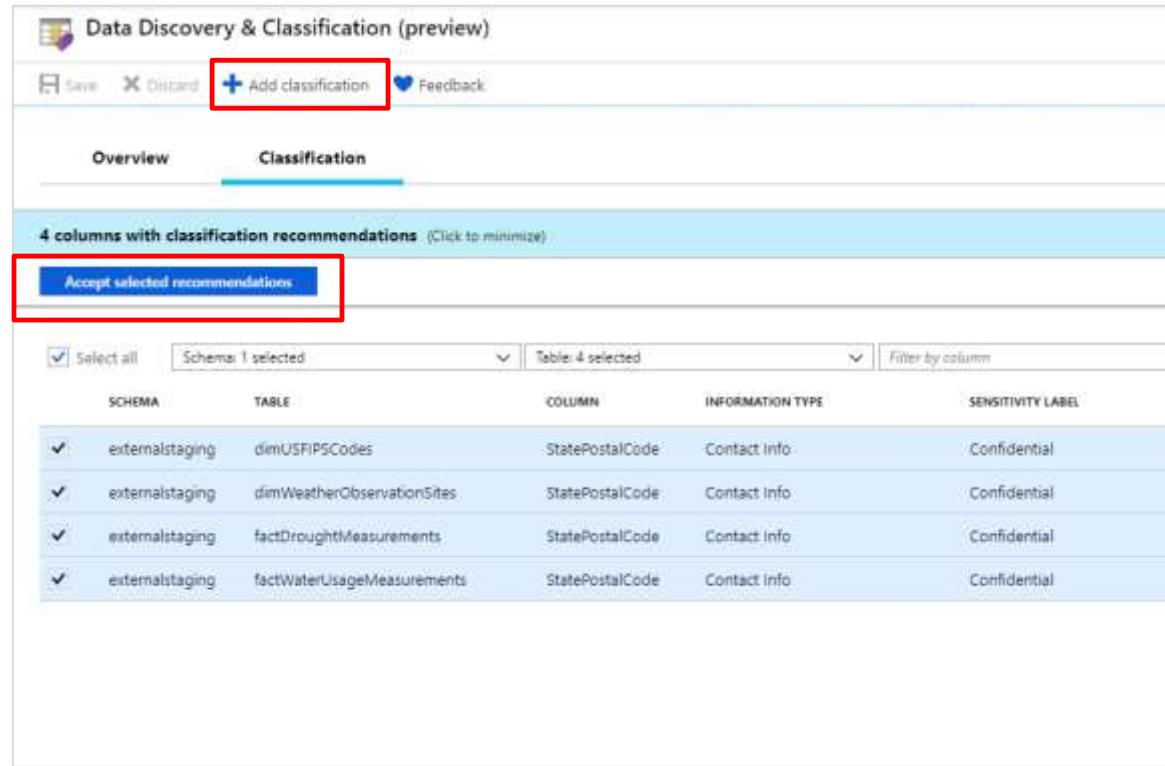
SQL Data Discovery & Classification - setup

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



The screenshot shows the 'ayotestdw (ayotestserver/ayotestdw) - Advanced Data Security' blade. On the left, there's a sidebar with various navigation items like Overview, Activity log, and Diagnose and solve problems. Under the 'Security' section, 'Advanced Data Security' is selected and highlighted with a red box. In the main area, there's a button labeled 'Enable Advanced Data Security on the server' which is also highlighted with a red box. Below it, there are sections for 'Data Discovery & Classification (preview)' and 'Vulnerability Assessment', both showing 0 results.

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



The screenshot shows the 'Data Discovery & Classification (preview)' blade. At the top, there's a 'Classification' tab which is selected and highlighted with a red box. Below it, there's a button labeled '+ Add classification'. The main area displays a table titled '4 columns with classification recommendations' with the following data:

| 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 |

At the bottom of the table, there's a button labeled 'Accept selected recommendations' which is also highlighted with a red box.

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 Azure portal interface for managing a SQL Data Warehouse named 'ayotestdw'. In the left sidebar, under 'Security', the 'Auditing' option is selected and highlighted with a red box. On the main page, the 'Audit' switch is turned 'ON'. Below it, there are sections for 'Audit log destination' (set to 'Storage') and 'Storage details' (showing 'azuriteblobstorage'). Other options like 'Log Analytics (Preview)' and 'Event Hub (Preview)' are also listed.

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

The screenshot shows the Azure portal's storage blob service. Under the 'sqldbauditlogs' container, a blob named '01_34_30_090.xel' is listed. The blob was uploaded on April 1, 2019, at 6:34:31 PM. The blob type is 'Append blob' and its size is 7.5 KB. The 'LEASE STATE' column shows 'Available'.

Step 3: Open logs using extended events viewer in SSMS. Configure viewer to include 'data_sensitivity_information' column

The screenshot shows the Microsoft Extended Events Viewer in SQL Server Management Studio (SSMS). It is displaying the contents of the '01_34_30_090.xel' file. The table has columns: name, timestamp, affected_rows, application_name, client_ip, data_sensitivity_information, and database_name. A red box highlights the 'data_sensitivity_information' column in the table header. Another red box highlights the 'data_sensitivity_information' row in the 'Details' pane for the first audit_event entry, which shows the value 'master'.

| 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 18:38:35.7052285 | 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 18:38:35.4856126 | 0 | .Net SqlClient Data Provider | 10.0.0.4 | | master |
| audit_event | 2019-02-26 18:38:35.4675695 | 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 18:38:35.4290438 | 0 | .Net SqlClient Data Provider | 10.0.0.4 | | master |

Event: audit_event (2019-02-26 18:38:35.6680990)

Details

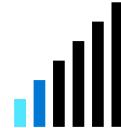
| | |
|------------------------------|--|
| Field | Value |
| action_id | 1178561924 |
| additional_information | <login_information><error_code>18456</error_code><error_state> |
| affected_rows | 0 |
| application_name | .Net SqlClient Data Provider |
| auth_scheme_version | 1 |
| client_ip | 169.54 |
| client_ip | 10.0.0.4 |
| connection_id | F1AD6457-9F40-409B-B43C-E633AAF47902 |
| data_sensitivity_information | master |
| database_name | master |
| database_principal_id | 1 |
| database_principal_name | |
| duration_milliseconds | 0 |
| event_time | 2019-02-26 18:38:35.6743004 |
| host_name | sqlvm093 |
| is_column_permission | False |
| object_id | 5 |
| object_name | master |
| process_id | 1 |
| process_name | sqlagent |

Network Security - Business requirements



How do we implement network isolation?

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

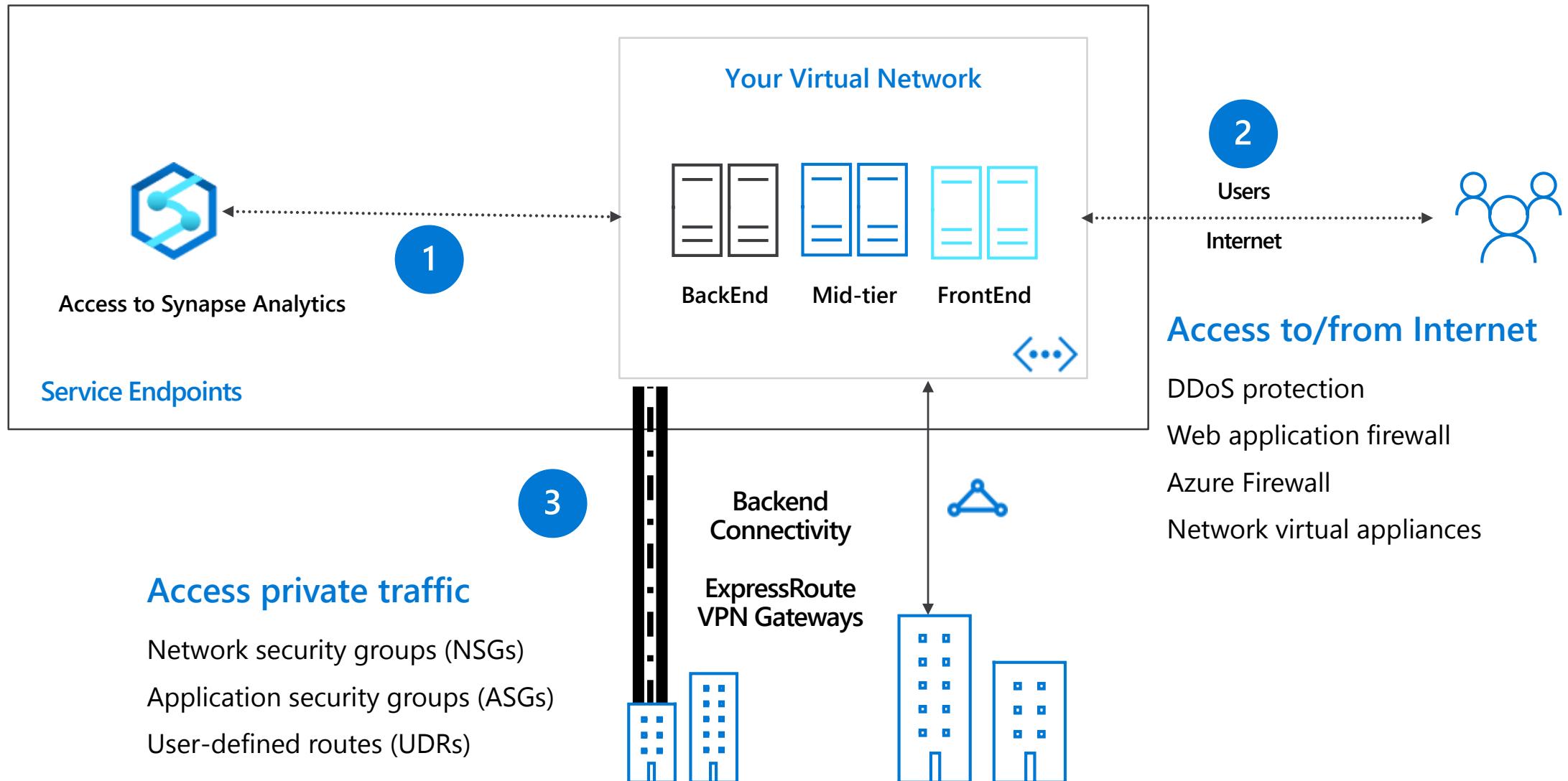


How do we achieve separation?

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



Azure networking: application-access patterns

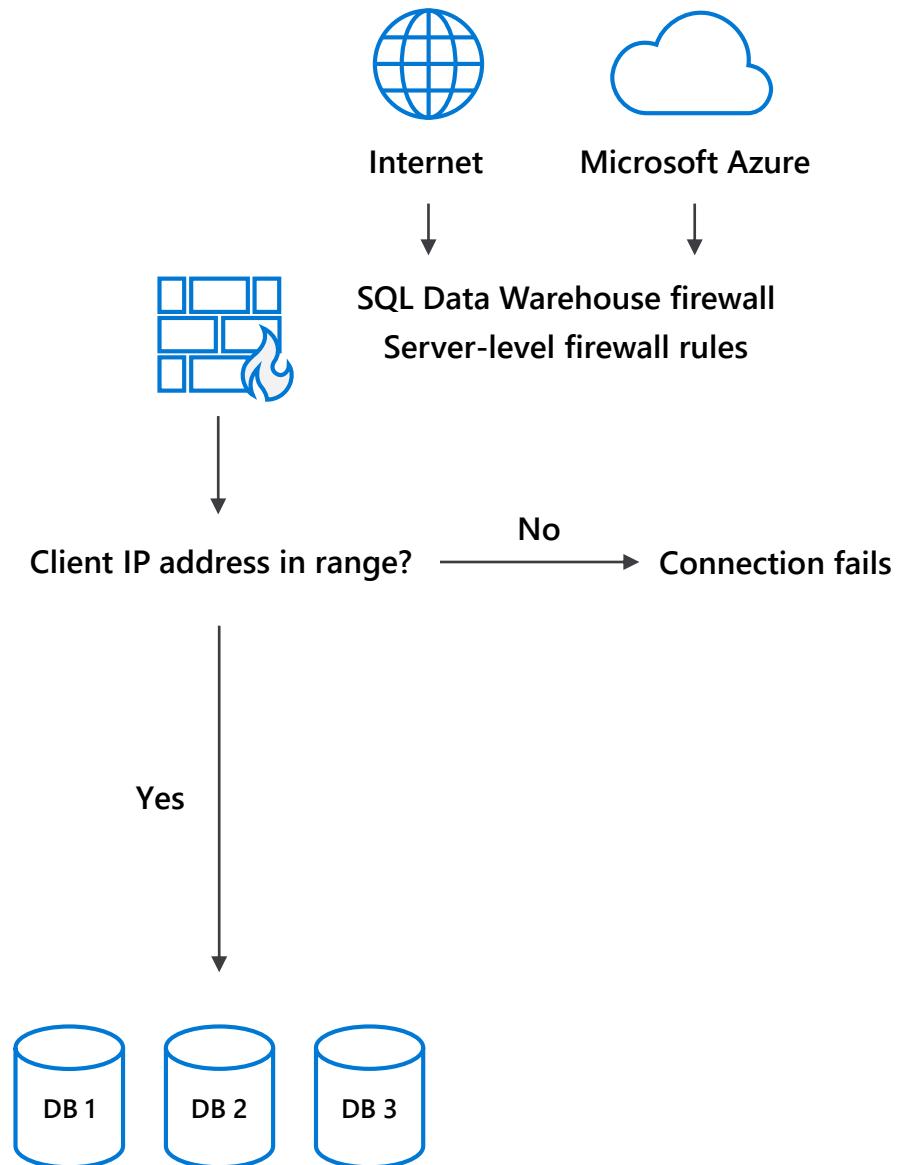


Securing with firewalls

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.

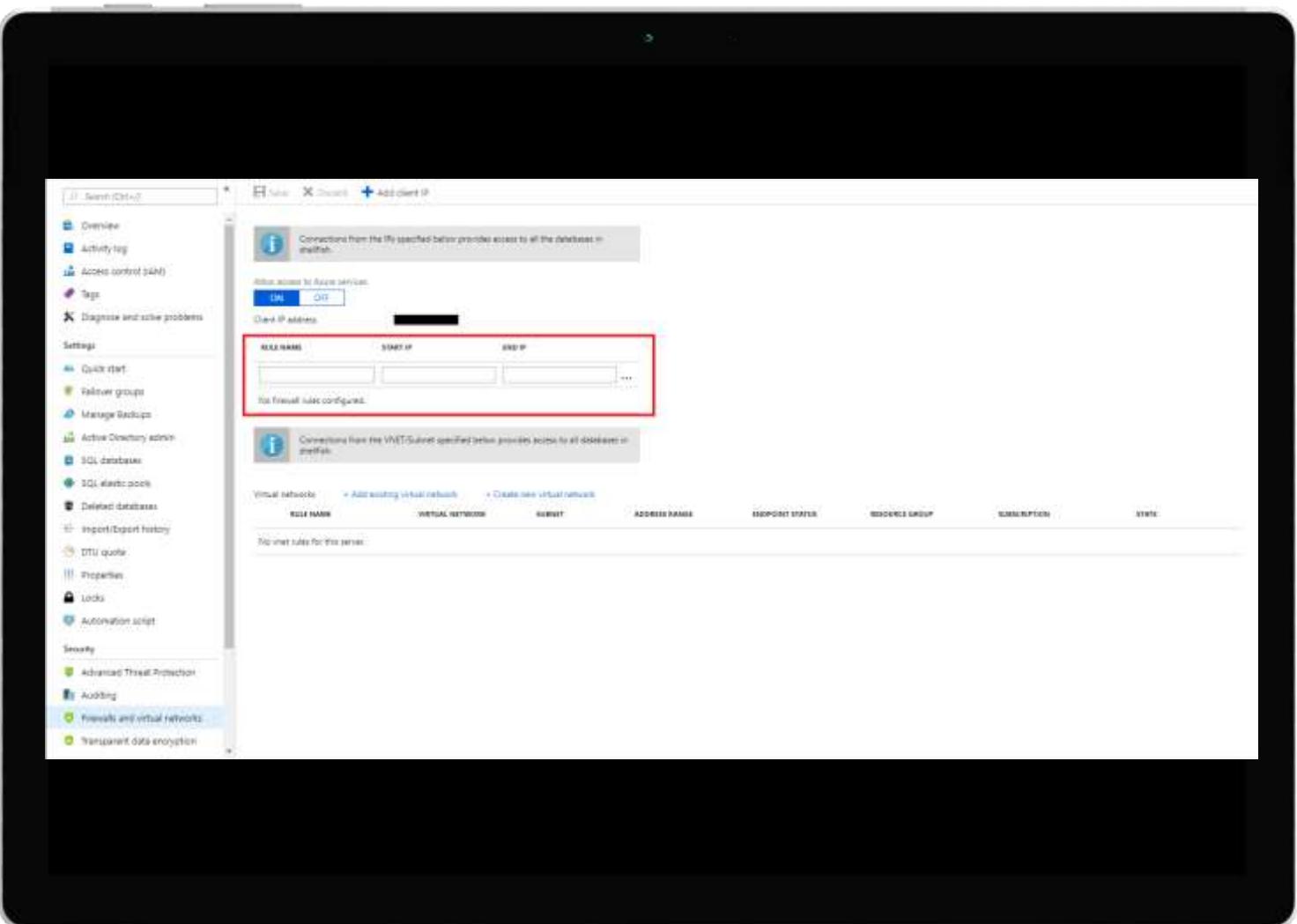
Allow Azure applications to connect.

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'. At the top, there are 'Save' and 'Discard' buttons, and a '+ Add client IP' button which is highlighted with a red box. Below this is an information icon with the text: 'Connections from the IPs specified below provides access to all the databases in gm-sql-db-server-svr1.' Underneath, there is a toggle switch for 'Allow access to Azure services' which is set to 'OFF'. A 'Client IP address' field contains the value '73.118.201.137'. The main table lists two IP rules:

| 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 | ... |

Below the table is another information icon with the text: 'Connections from the VNET/Subnet specified below provides access to all databases in gm-sql-db-server-svr1.' At the bottom, there are buttons for 'Virtual networks' (disabled), '+ Add existing' (highlighted with a red box), and '+ Create new'. There is also a table with columns: RULE NAME, RESOURCE GROUP/VNET NAME, and SUBNET.

Authentication - Business requirements

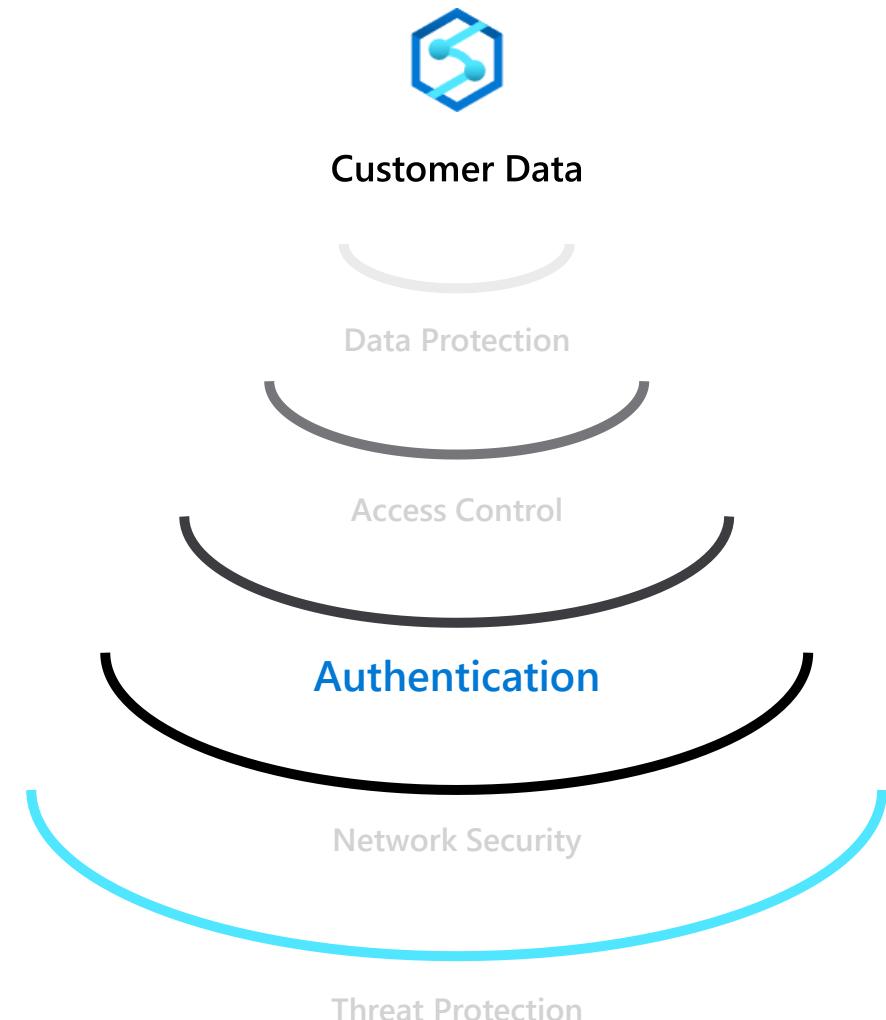


How do I configure Azure Active Directory with Azure Synapse Analytics?

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

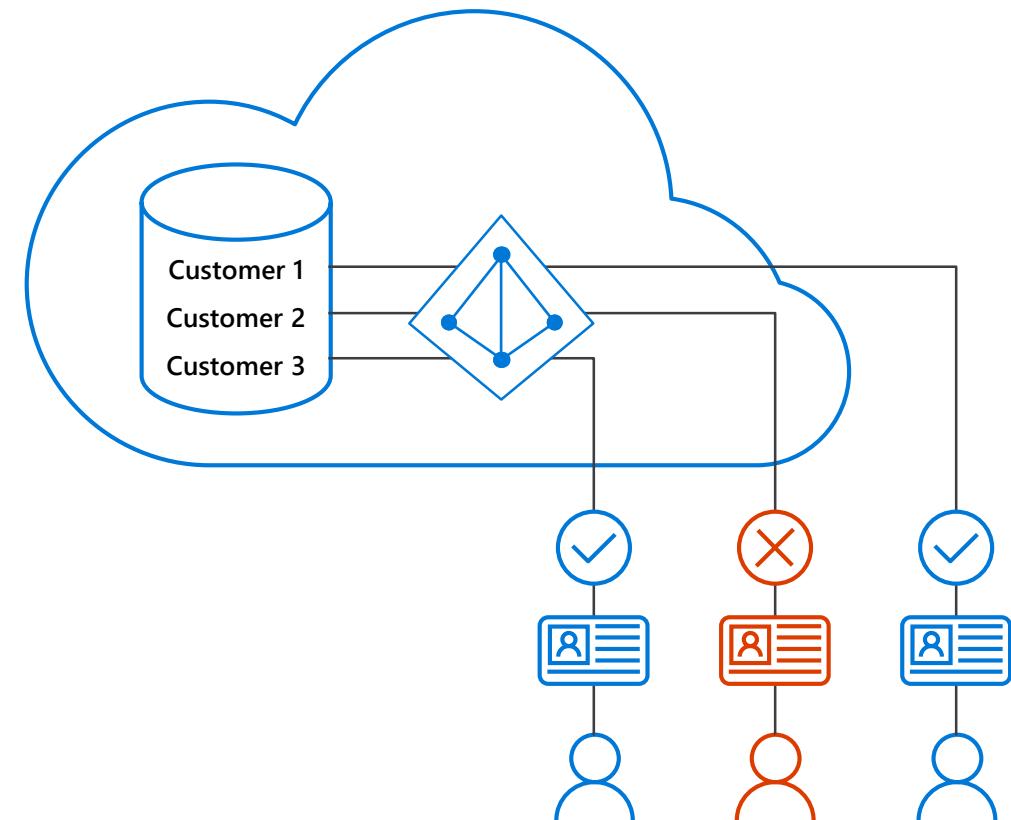
Alternative to SQL Server authentication

Limits proliferation of user identities across databases

Allows password rotation in a single place

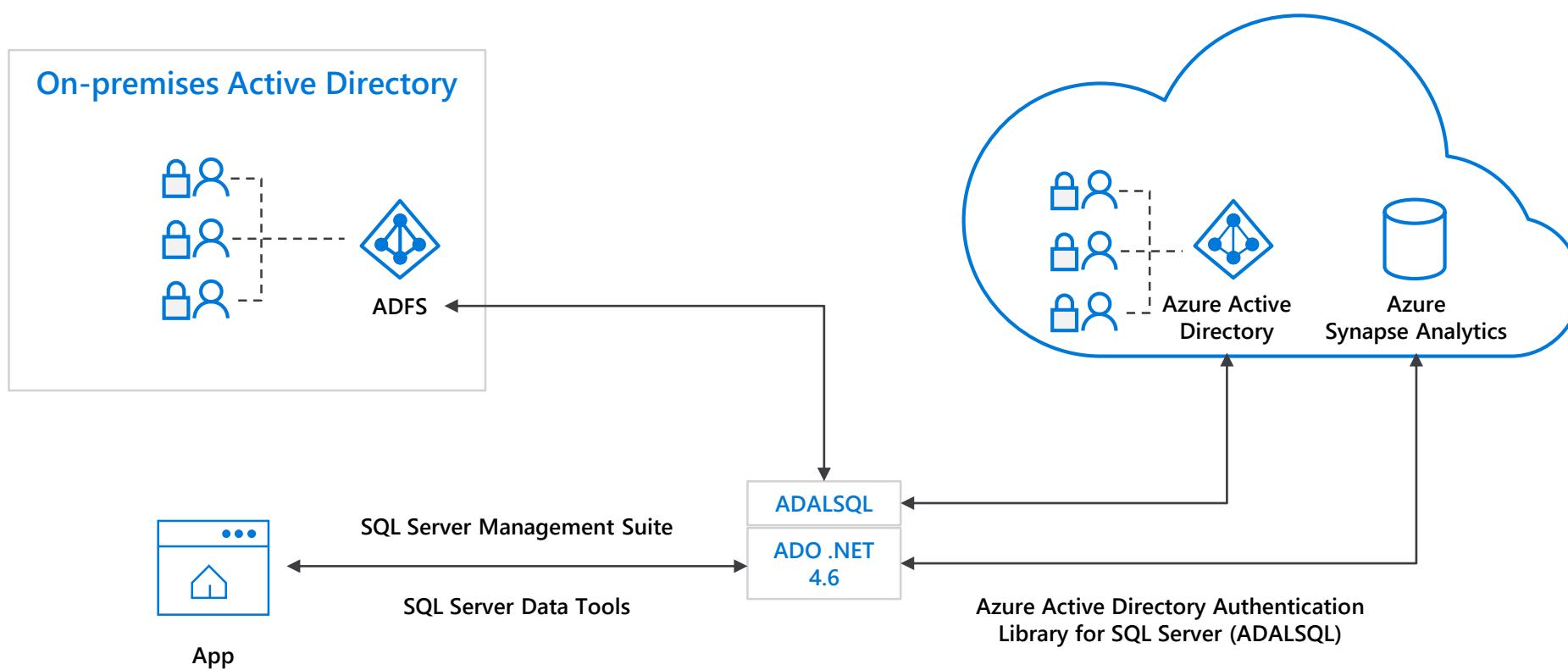
Enables management of database permissions by using external Azure Active Directory groups

Eliminates the need to store passwords



Azure Active Directory trust architecture

Azure Active Directory and Azure Synapse Analytics



SQL authentication

Overview

This authentication method uses a username and password.

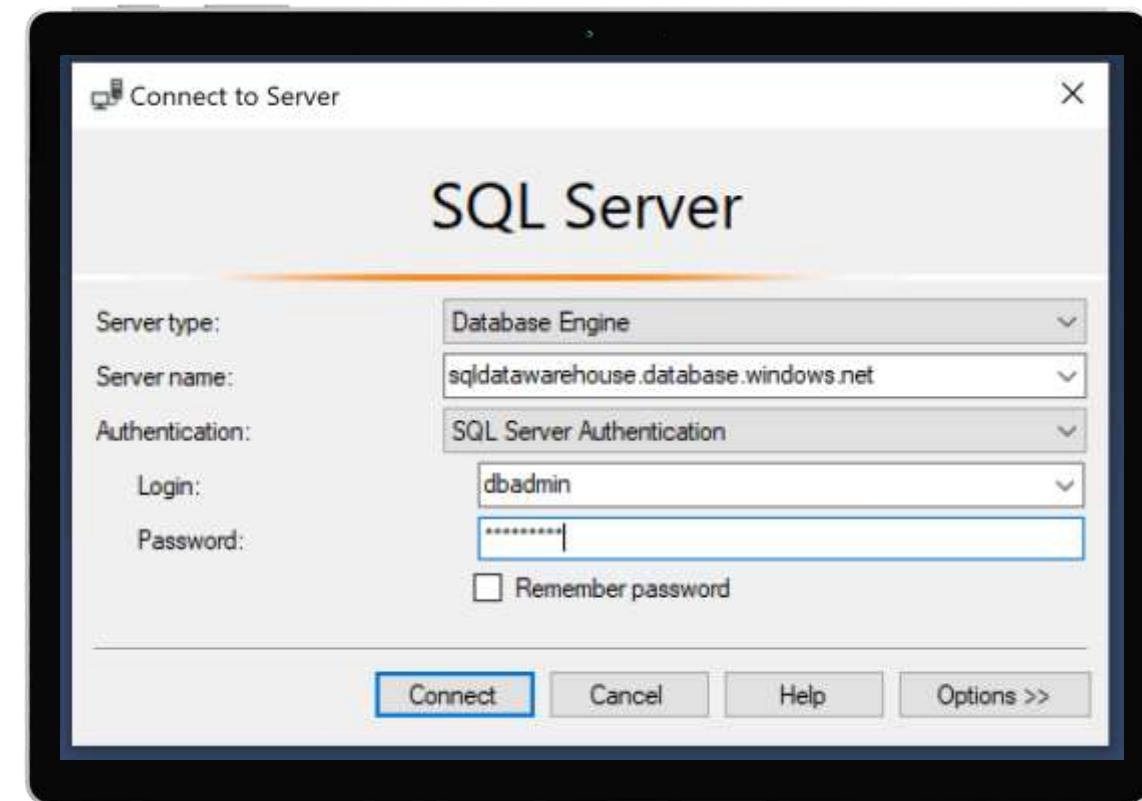
When you created the logical server for your data warehouse, you specified a "server admin" login with a username and password.

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;
```



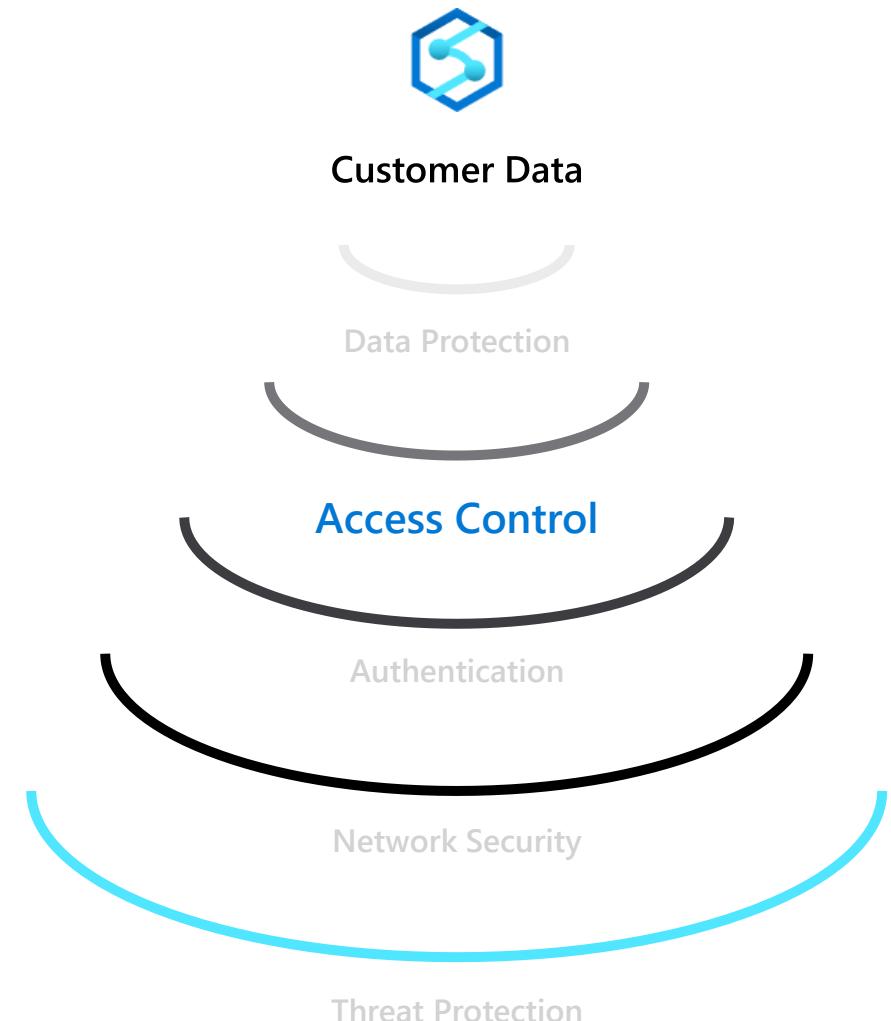
Access Control - Business requirements



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

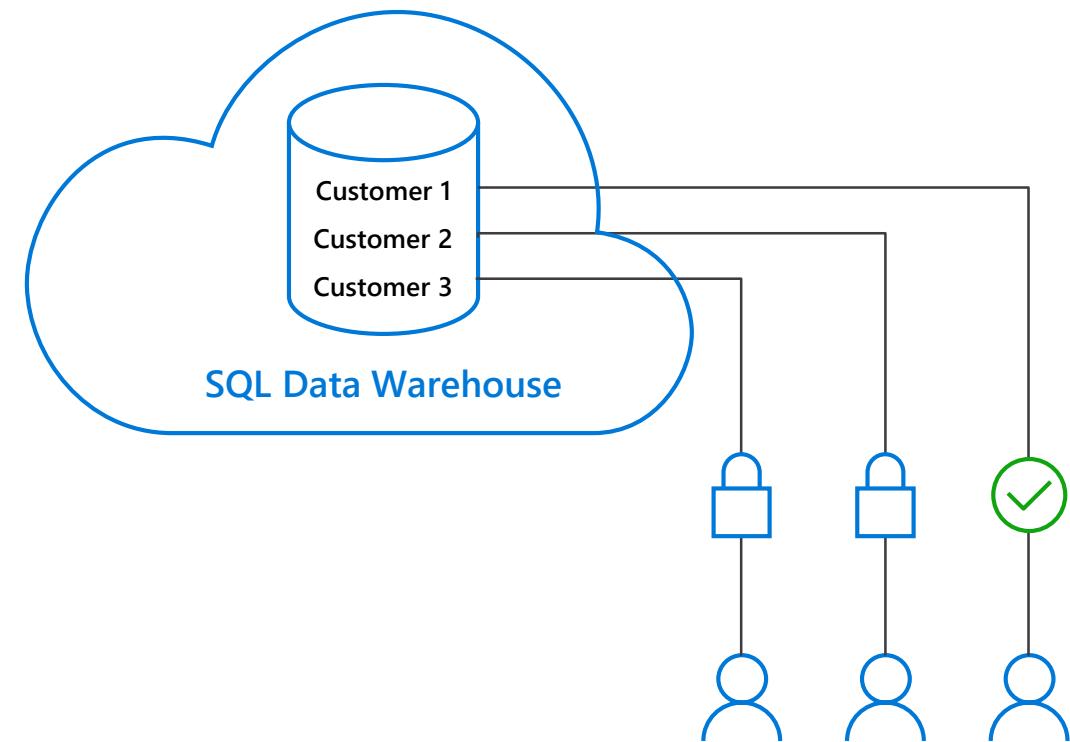
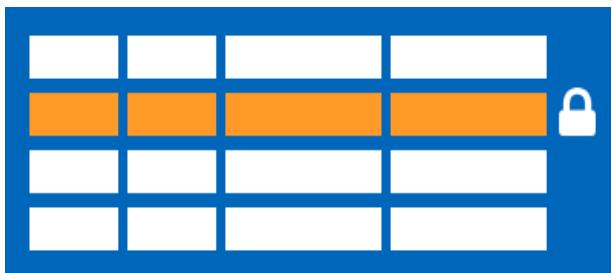
Fine grained access control of specific rows in a database table.

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
```

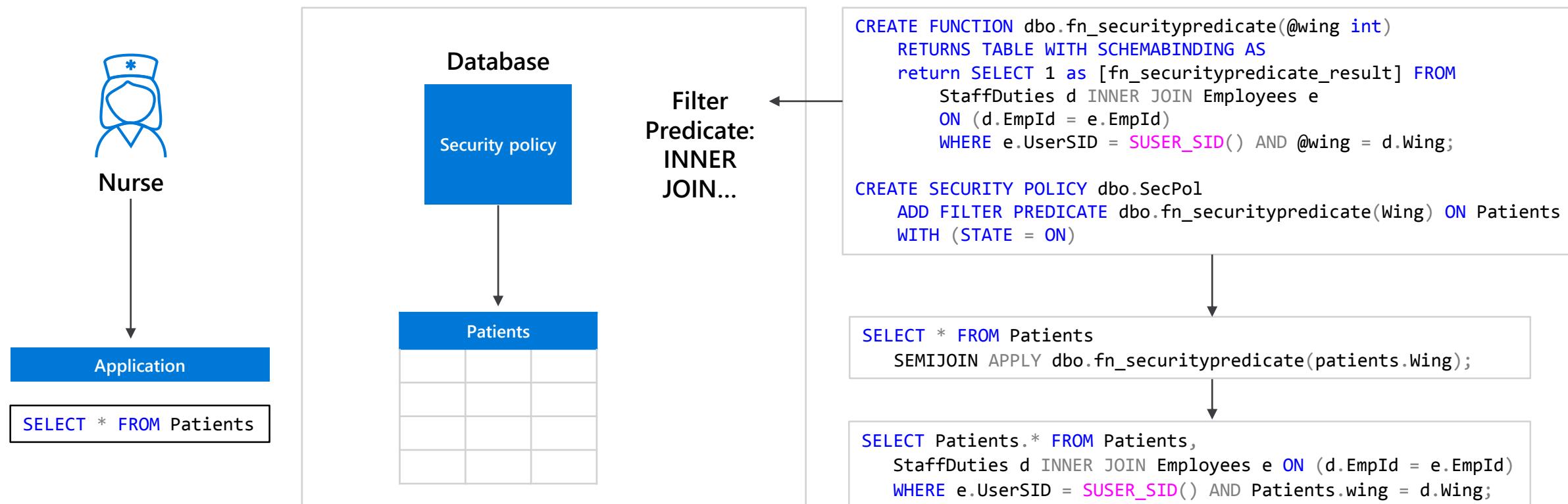
Row-level security

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

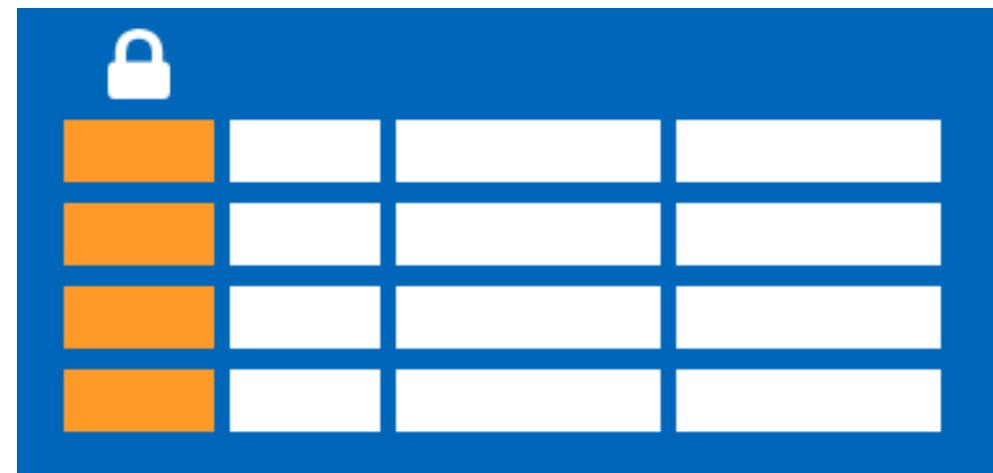
Overview

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.

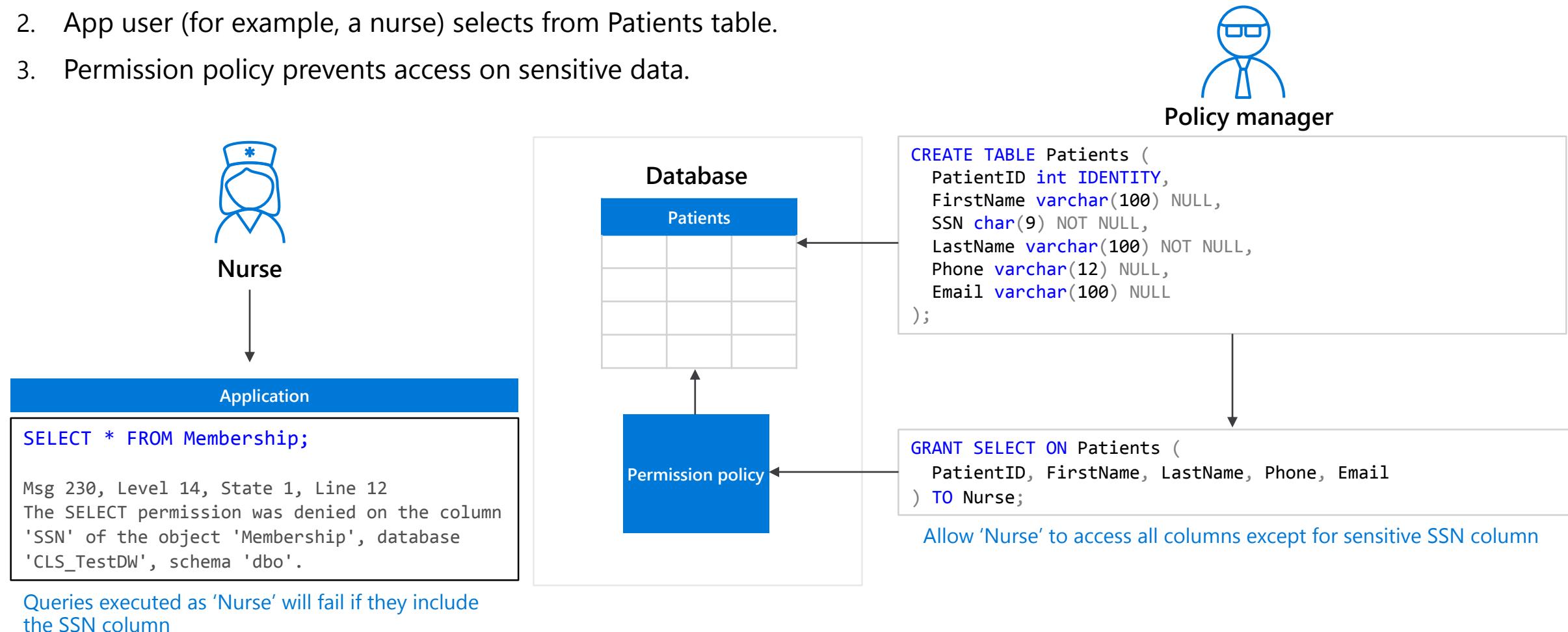
Both Azure Active Directory (AAD) and SQL authentication are supported.



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.

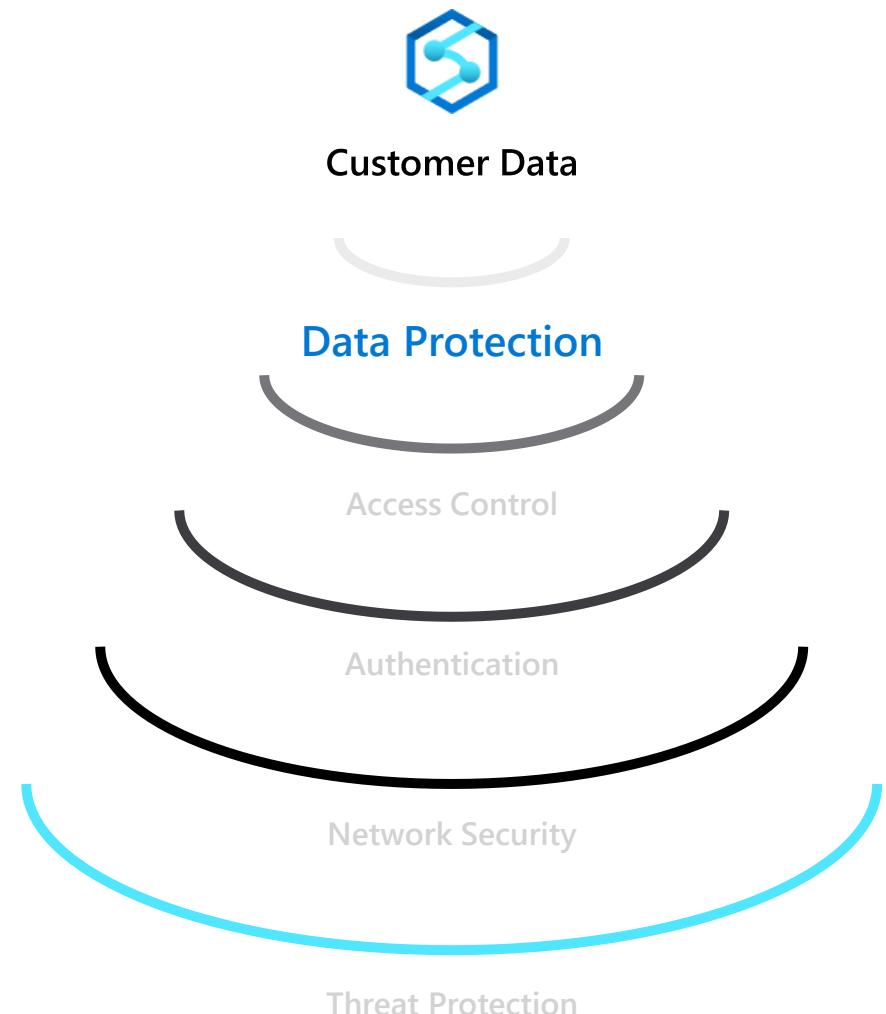


Data Protection - Business requirements



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

What key management options do I have?



Dynamic Data Masking

Overview

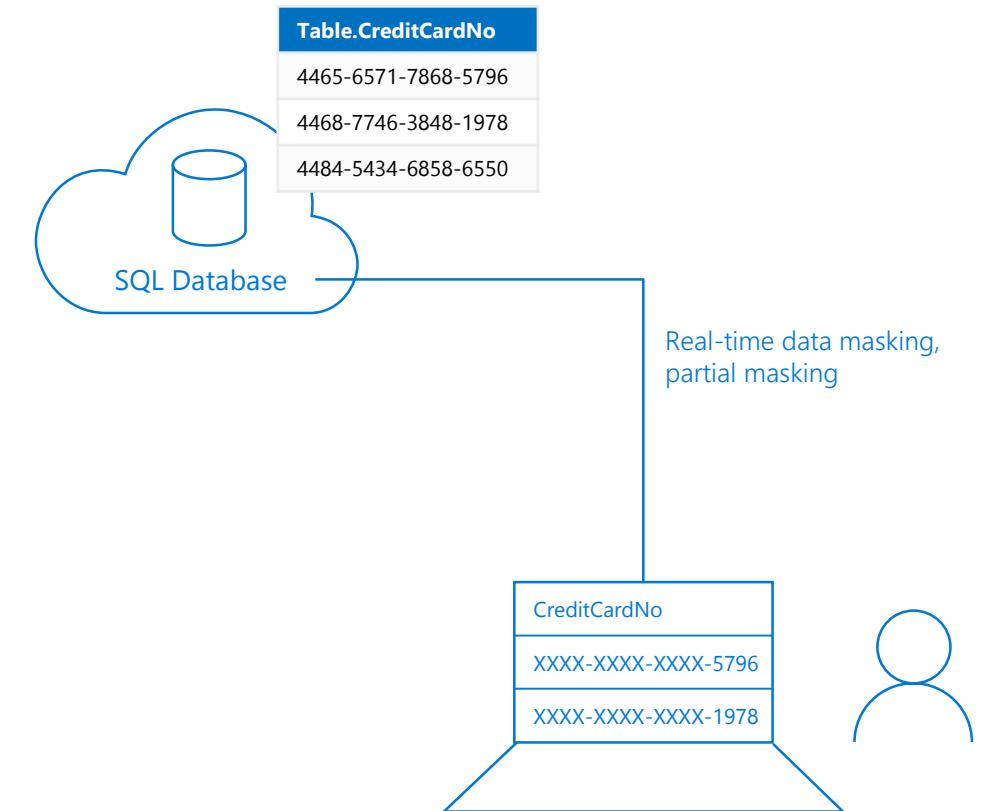
Prevent abuse of sensitive data by hiding it from users

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

```

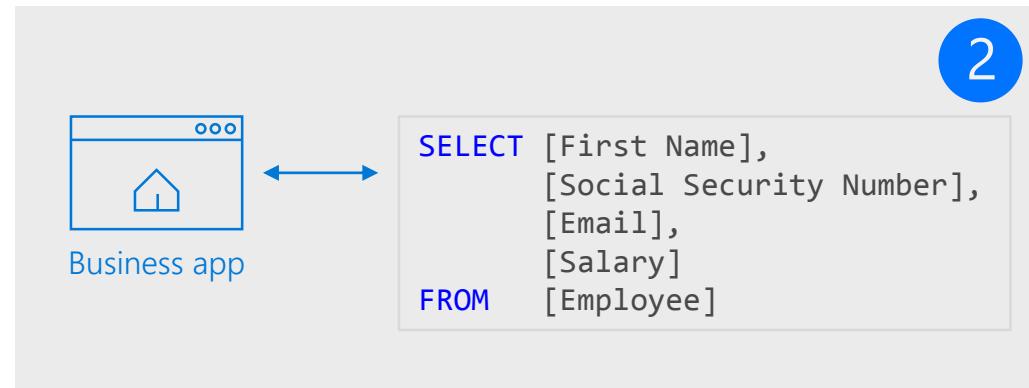
ALTER TABLE [Employee]
ALTER COLUMN [SocialSecurityNumber]
ADD MASKED WITH (FUNCTION = 'DEFAULT()')

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 |

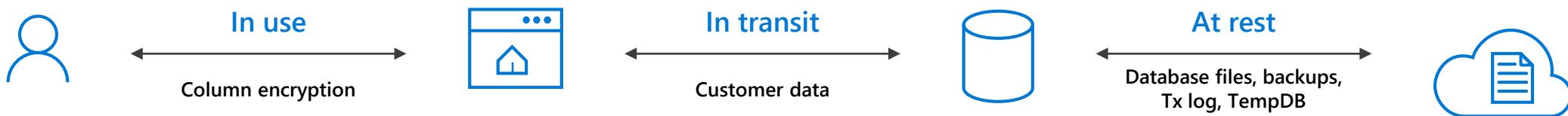
Masked data (admin1 login)

| | 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.

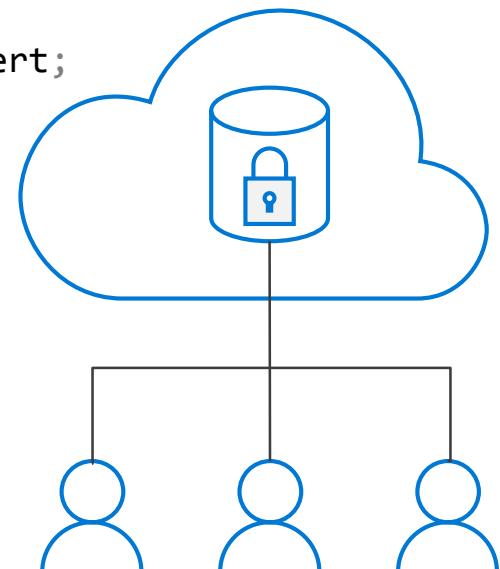
Service OR User managed keys.

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

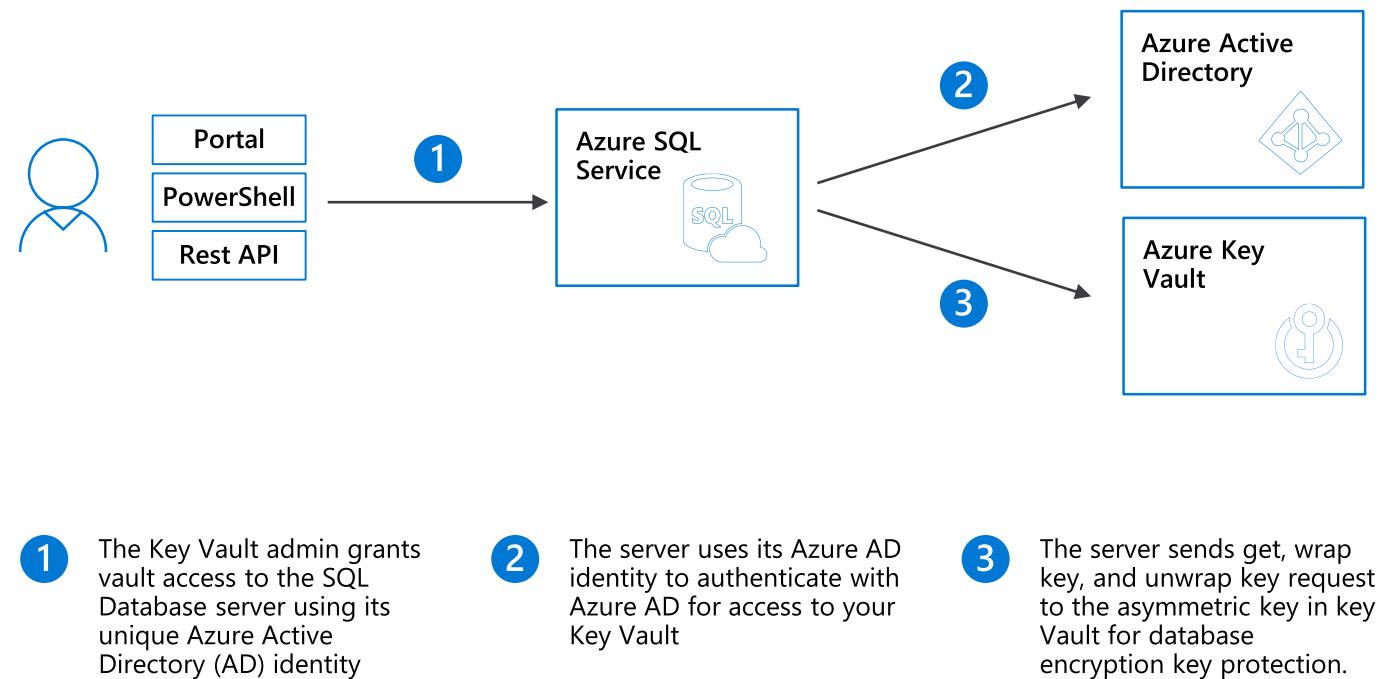
Benefits with User Managed Keys

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

Highly available and scalable cloud-based key store.

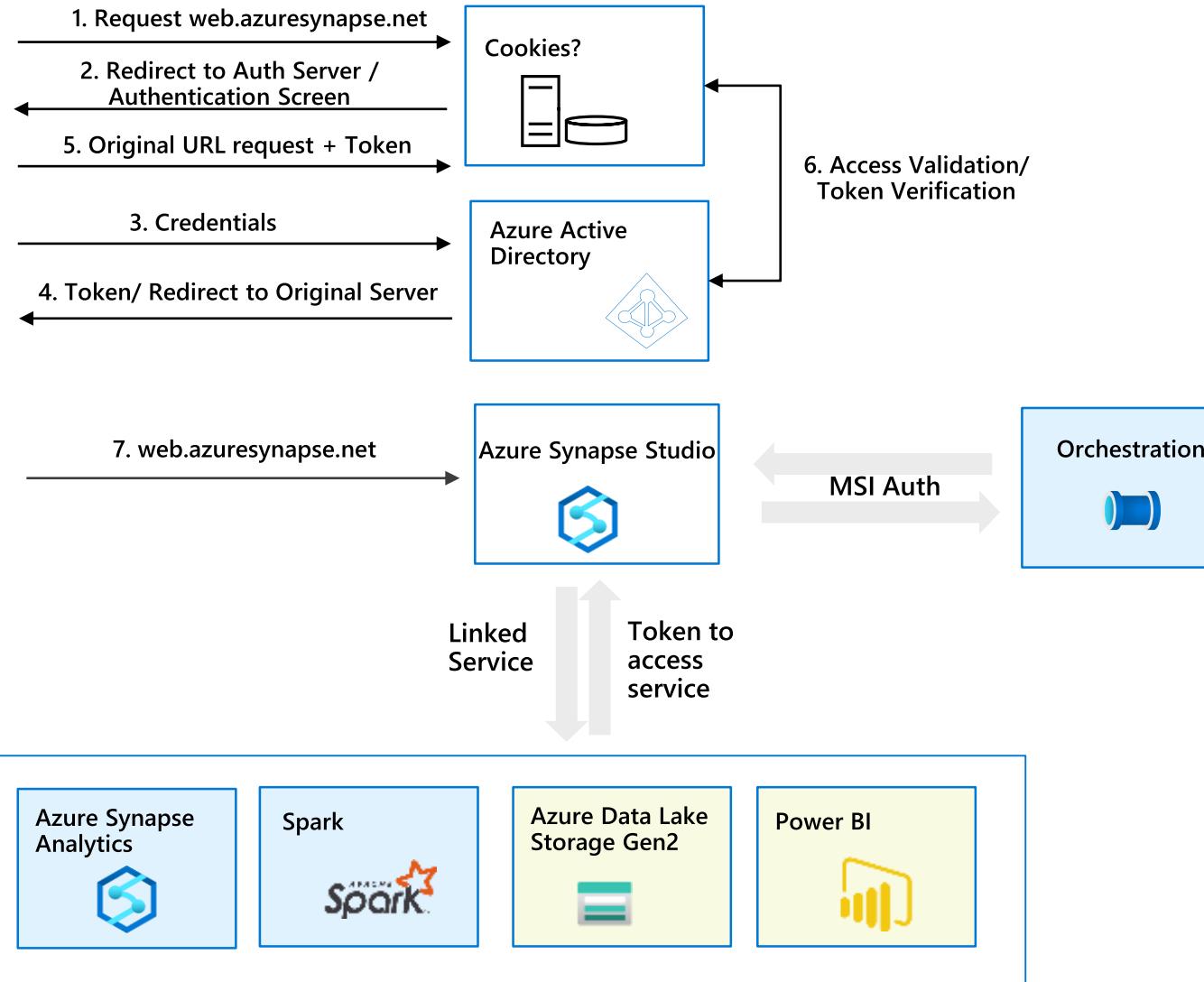
Central key management that allows separation of key management and data.

Configurable via Azure Portal, PowerShell, and REST API.



Single Sign-On

Synapse Foundation Components
 Synapse Linked Services

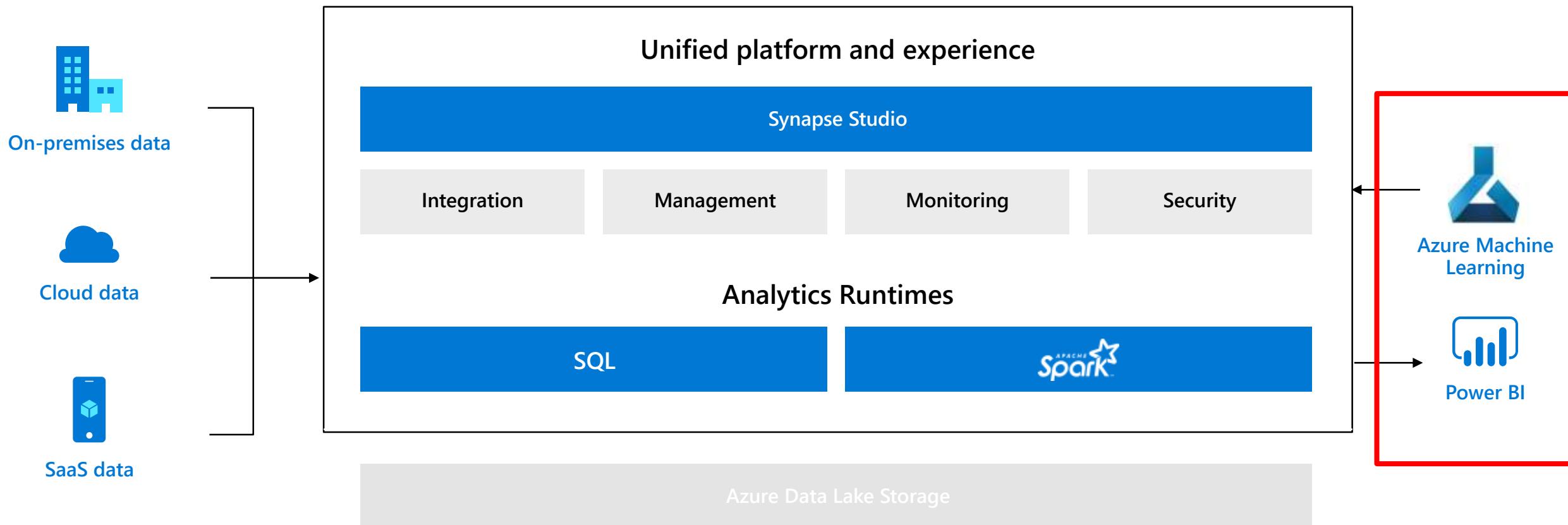




Azure Synapse Analytics Connected Services

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Azure Machine Learning

Overview

Data Scientists can use Azure ML notebooks to do (distributed) data preparation on Synapse Spark compute.

Benefits

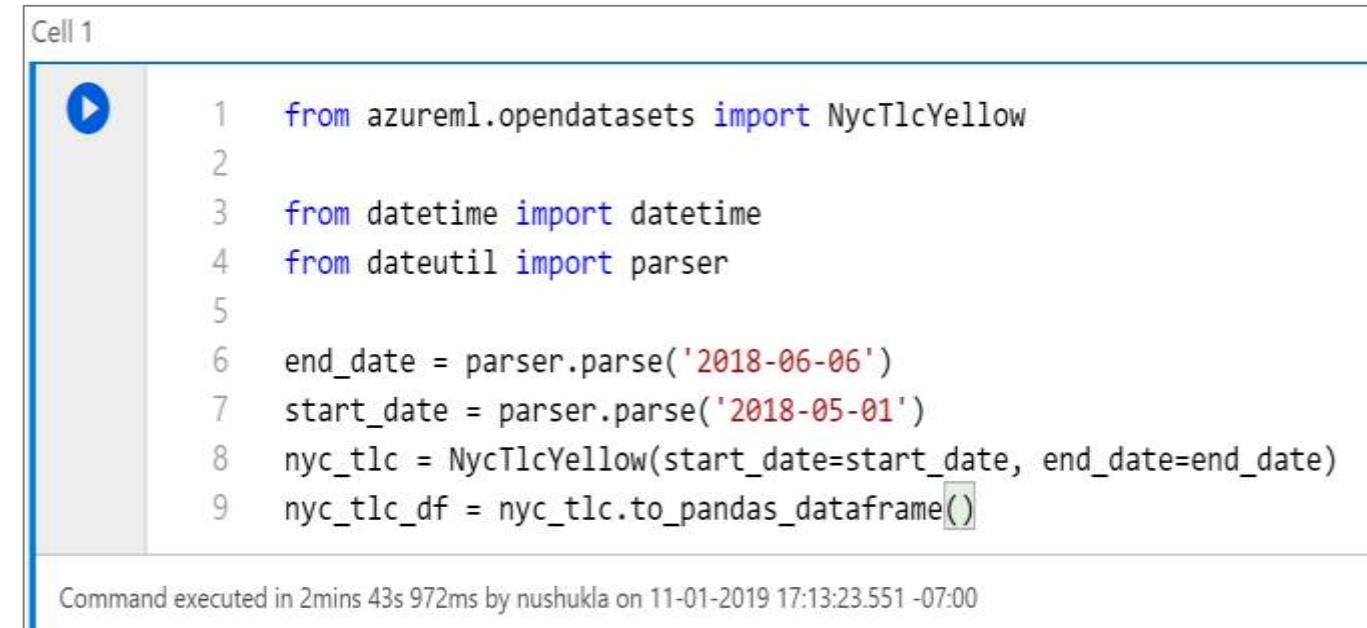
Connect to your existing Azure ML workspace and project

Use the AutoML Classifier for classification or regression problem

Train the model

Access open datasets

Cell 1



The screenshot shows a notebook cell titled "Cell 1". It contains the following Python code:

```
1 from azureml.opendatasets import NycTlcYellow
2
3 from datetime import datetime
4 from dateutil import parser
5
6 end_date = parser.parse('2018-06-06')
7 start_date = parser.parse('2018-05-01')
8 nyc_tlc = NycTlcYellow(start_date=start_date, end_date=end_date)
9 nyc_tlc_df = nyc_tlc.to_pandas_dataframe()
```

Below the code, a status message indicates: "Command executed in 2mins 43s 972ms by nushukla on 11-01-2019 17:13:23.551 -07:00".

Azure Machine Learning (continued)

Configure AutoML and Train the Models

Cell 9

```
1 l_config = AutoMLConfig(task = 'regression', debug_log = 'automl_errors.log',
2                             primary_metric = 'normalized_root_mean_squared_error', iteration_timeout_minutes = 10,
3                             iterations = 2, preprocess = True, n_cross_validations = 2, max_concurrent_iterations = 2,
4                             verbosity = logging.INFO, spark_context=sc, enable_onnx_compatible_models=True, cache_store=True)
```

Cell 10

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

Best Model

Cell 12

```
[ ] 1 best_run, fitted_model = local_run.get_output(return_onnx_model=True)
2 print(fitted_model)
```

Portal URL for Monitoring Runs

Cell 14

```
[ ] 1 more Insights of experiment
2 displayHTML("<a href={} target='_blank'>Your experiment in Azure Portal: {}</a>".format(local_run.get_portal_url(), local_r
```

Power BI

Overview

Power BI is a business analytics service that delivers insights to enable fast, informed decisions

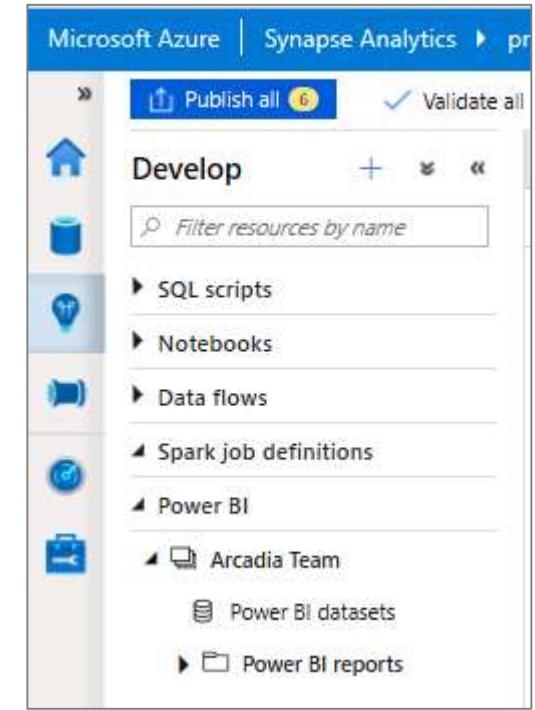
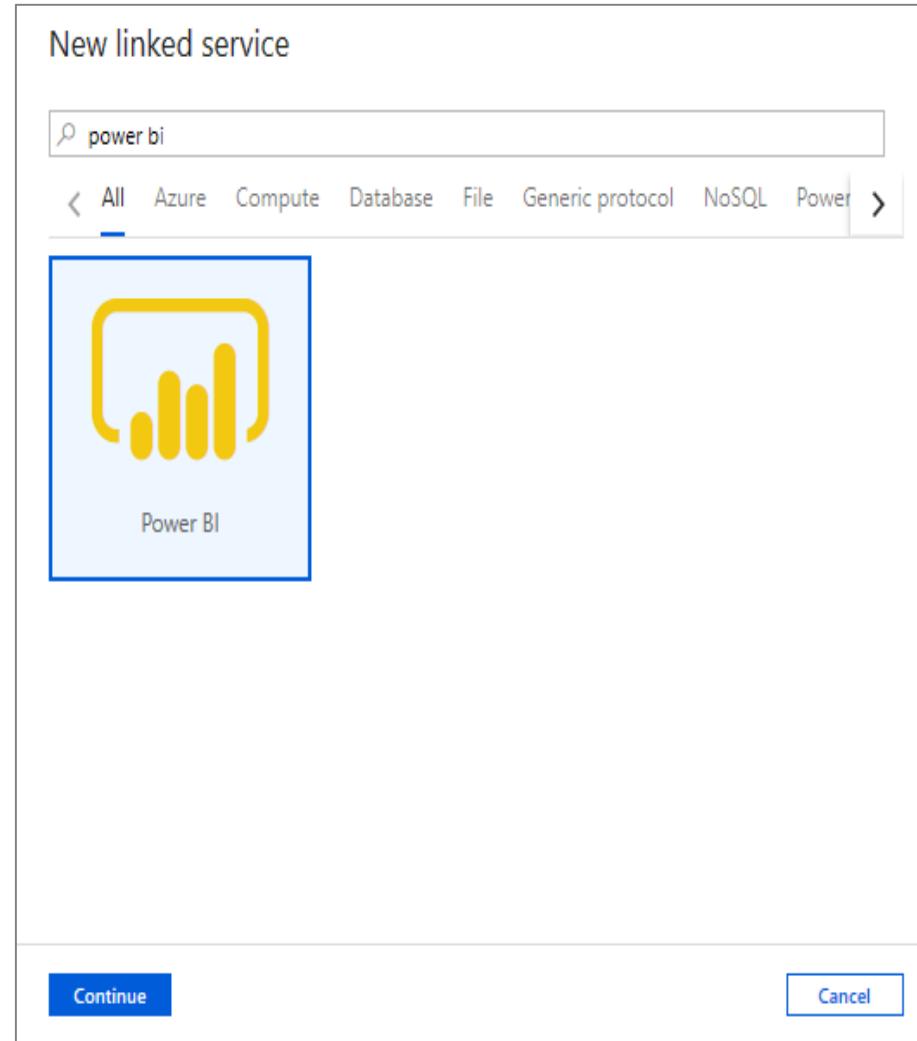
Benefits

Create Power BI reports in the workspace

Have access to published reports in workspace

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

Visually explore and analyze data



The screenshot shows the Microsoft Power BI service interface. The top navigation bar includes 'Publish all' (2), 'Validate all', 'Refresh', and 'Discard all'. Below the navigation is a ribbon menu with tabs: 'yellowcabprep', 'PrepareTaxiData *', '1 Marketshare', '2 MostTripsHo...', 'AutoML', and 'SynapseNYIgnite...'. The left sidebar, titled 'Develop', contains sections for 'Notebooks' (YellowCabExploration_sqlo, AMLautoMLPredict, AutoML, Data Download_Weather, * PrepareTaxiData, yellowcabprep, YellowCabPrepare) and 'Data flows' (PrepareCabDataFlow). The main workspace displays a chart titled 'Synapse GreenCab and YellowCab by Day/Hour' showing 'Running Distance per Hour' over time from Jan-2019 to Jun-2019, with data for GreenCab (blue) and YellowCab (yellow). To the right of the chart are three panels: 'Filters' (Search, Filters on this page, Add data fields here, Filters on all pages, Add data fields here), 'VISUALIZATIONS' (a grid of visualization icons), and 'FIELDS' (a list of data fields: dimHoliday, dimNYCLocations, Fhv, GreenCab, PredictedValues, vwFhvMarketShare, vwGrnCabMarketShare, vwMarketShareBy..., vwPredictedValues, vwYelCabMarketShare, weather, YellowCab, YellowCabTripsHoli...). At the bottom, there are navigation buttons for 'Page 1' and a '+' button.

Power BI Report Overview:

Report Title: yellowcabprep

Visuals:

- Line chart showing "Trips, Dispatches and Returns by Date/Price". The chart displays three data series over time: Trips (blue), Dispatches (yellow), and Returns (green). The Y-axis represents the count of trips, dispatches, or returns.
- Bar chart showing "Trips by Holiday Name". The chart displays the number of trips for various holidays. The X-axis lists the holidays, and the Y-axis shows the count of trips.

Filters:

- Filters on this visual:**
 - holidayName: is (All)
 - numTrips: is (All)
- Filters on this page:** Add data fields here
- Filters on all pages:** Add data fields here

Visualizations:

- Line chart (selected): Trips, Dispatches and Returns by Date/Price
- Bar chart: Trips by Holiday Name

Fields:

- Axis:** holidayName
- Legend:** Add data fields here
- Value:** numTrips
- Tooltips:** Add data fields here
- DRILLTHROUGH:** numTrips
- Cross-report:** Add data fields here

Available Fields:

- dimHoliday
- dimNYCLocations
- Fhv
- GreenCab
- PredictedValues
- vwFhvMarketShare
- vwGrnCabMarketS...
- vwMarketShareBy...
- vwPredictedValues
- vwYelCabMarketSh...
- weather
- YellowCab
- YellowCabTripsHoli...
- date
- holidayName
- numTrips
- year

Report Navigation:

- Page 1
- Page 2

The screenshot shows the Microsoft Power BI service interface. The left sidebar contains navigation links for 'Develop' (selected), 'Notebooks', 'Data flows', 'Spark job definitions', 'Power BI', and 'SynapseNYTaxiInsights'. Under 'SynapseNYTaxiInsights', there are 'Power BI Datasets' and 'Power BI Reports' (selected). The main area displays a report titled 'yellowcabprep' with two visualizations: a line chart showing 'Predicted Daily Market Share' over time and a bar chart showing 'Trips by holidayName'. The 'Filters' pane on the right shows filters for 'holidayName' (is (All)) and 'numTrips' (is (All)). The 'Visualizations' pane lists various chart types, and the 'Fields' pane lists data fields like 'dimHoliday', 'dimNYCLocations', 'Fhv', 'GreenCab', etc., with 'YellowCabTripsHoli...' selected.

Azure Synapse Analytics features

| Limitless scale | GA | Preview |
|--------------------------------------|----|---------|
| Provisioned compute (data warehouse) | ✓ | |
| Materialized views | ✓ | |
| Workload importance | ✓ | |
| Workload isolation | | ✓ |
| On-demand query | | ✓ |
| Powerful insights | | |
| Power BI integration | ✓ | |
| Azure Machine Learning integration | ✓ | |
| Data lake exploration | ✓ | |
| Streaming analytics (data warehouse) | ✓ | |
| Apache Spark integration | ✓ | |
| Unified experience | | |
| Hybrid data ingestion | ✓ | |
| Azure Synapse studio | | ✓ |
| Unmatched security | | |
| Column- and row-level security | ✓ | |
| Dynamic data masking | ✓ | |
| Private endpoints | | ✓ |

Migration Path

SQL DW – All of the data warehousing features that were generally available in Azure SQL Data Warehouse (intelligent workload management, dynamic data masking, materialized views, etc.) continue to be generally available today. Businesses can continue running their existing data warehouse workloads in production today with Azure Synapse and will automatically benefit from the new capabilities which are in preview (unified experience with Azure Synapse studio, query-as-a-service, built-in data integration, integrated Apache Spark, etc.) once they become generally available in 2020 and can use them in production if they choose to do so. Customers will not have to migrate any workloads

Azure Data Factory - Continue using Azure Data Factory. When the new functional of data integration within Azure Synapse becomes generally available, we will provide the capability to import your Azure Data Factory pipelines into Azure Synapse. Your existing Azure Data Factory accounts and pipelines will work with Azure Synapse if you choose not to import them into the Azure Synapse workspace. Note that Azure-SSIS Integration Runtime (IR) will not be supported in Synapse

Power BI – Customers link to a Power BI workspace within Azure Synapse Studio so no migration needed

ADLS Gen2 – Customers link to ADLS Gen2 within Azure Synapse Studio so no migration needed

Azure Databricks – TBD

Azure HDInsight - The Spark runtime within the Azure Synapse service is different from HDInsight

Q & A



James Serra, Big Data Evangelist

Email me at: JamesSerra3@gmail.com

Follow me at: @JamesSerra

Link to me at: www.linkedin.com/in/JamesSerra

Visit my blog at: JamesSerra.com (where this slide deck is posted under the "Presentations" tab)