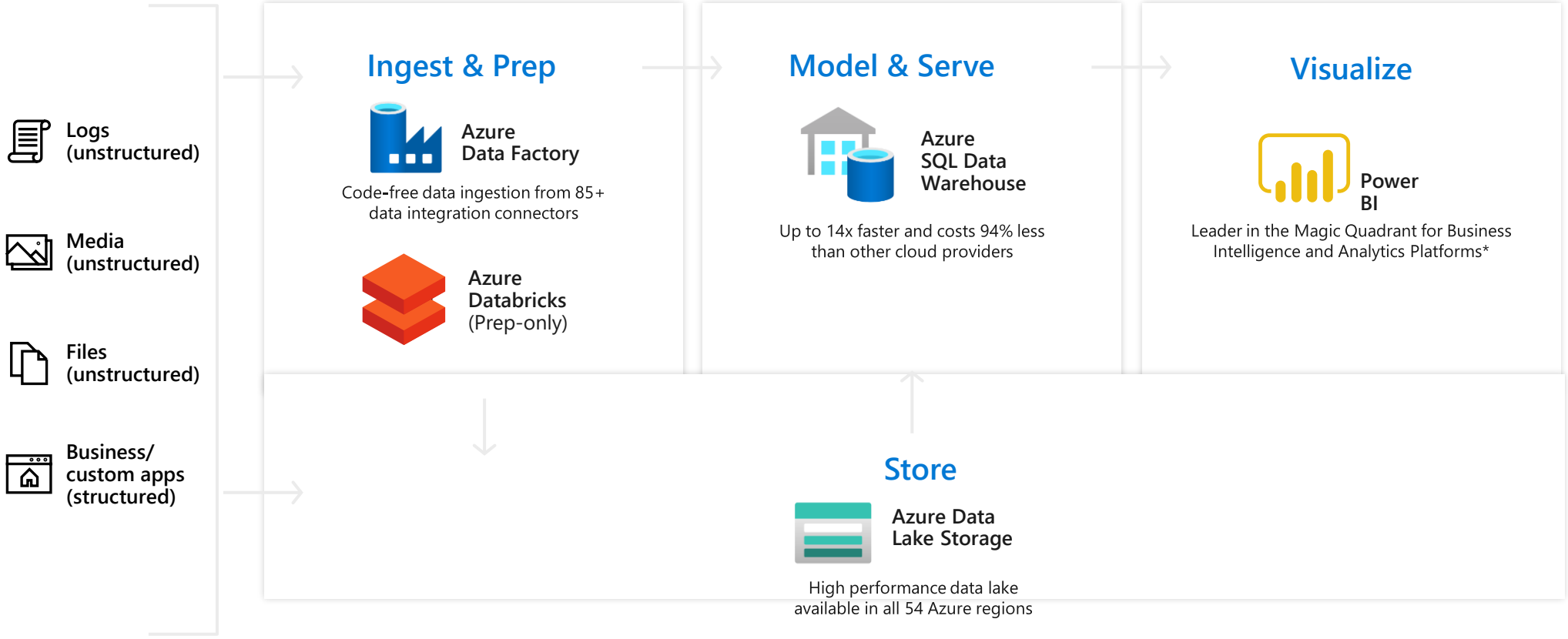
Azure Synapse Analytics

Sidney Cirqueira

Agenda

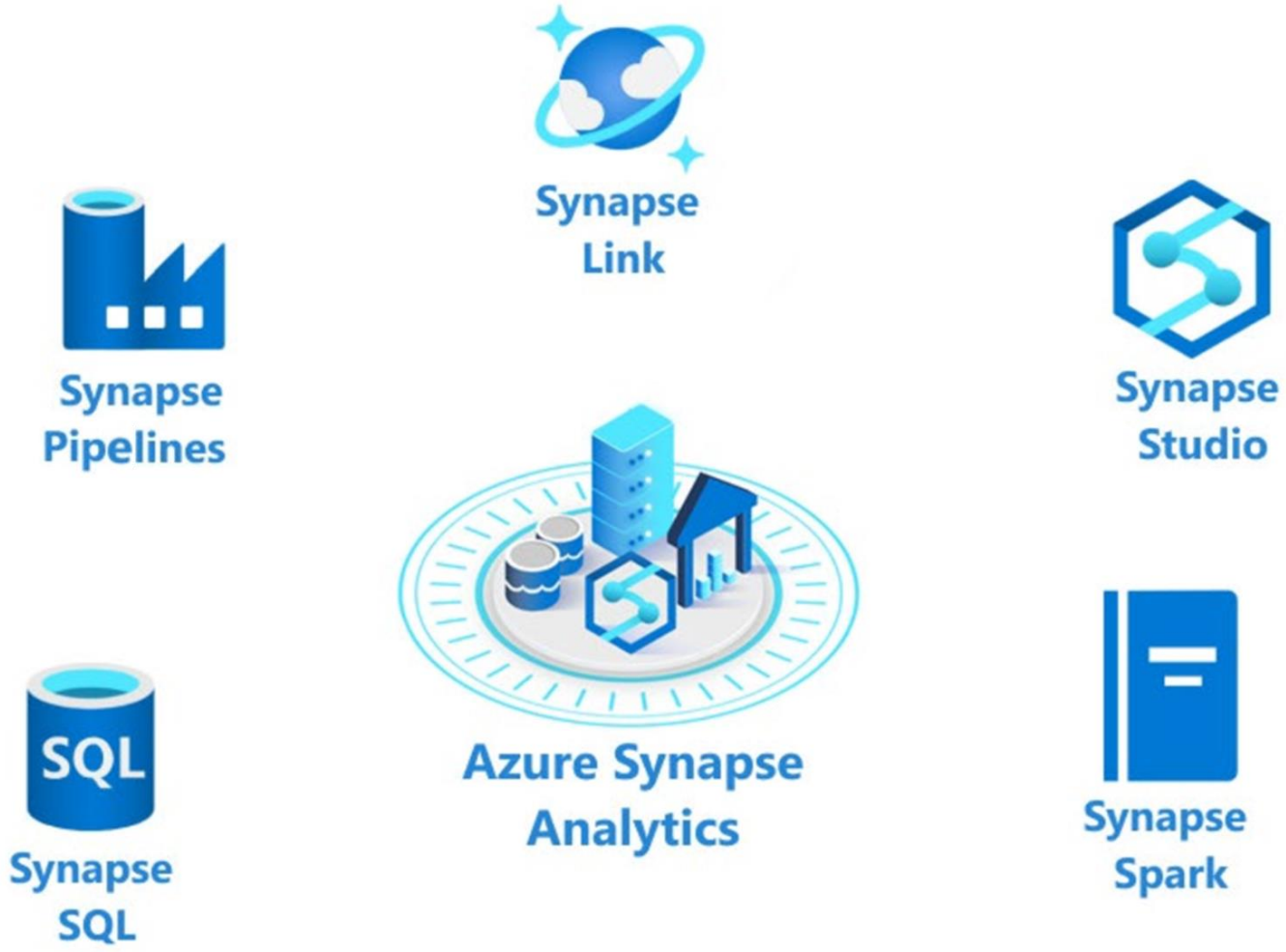
- Overview
- Studio
- SQL Serverless
- SQL Pool Dedicated
- Monitoring / Management
- Security
- Spark Analytics
- Pipelines
- Synapse Link for Cosmos DB
- DevOps
- DP 203

Default Architecture Modern Data Warehouse

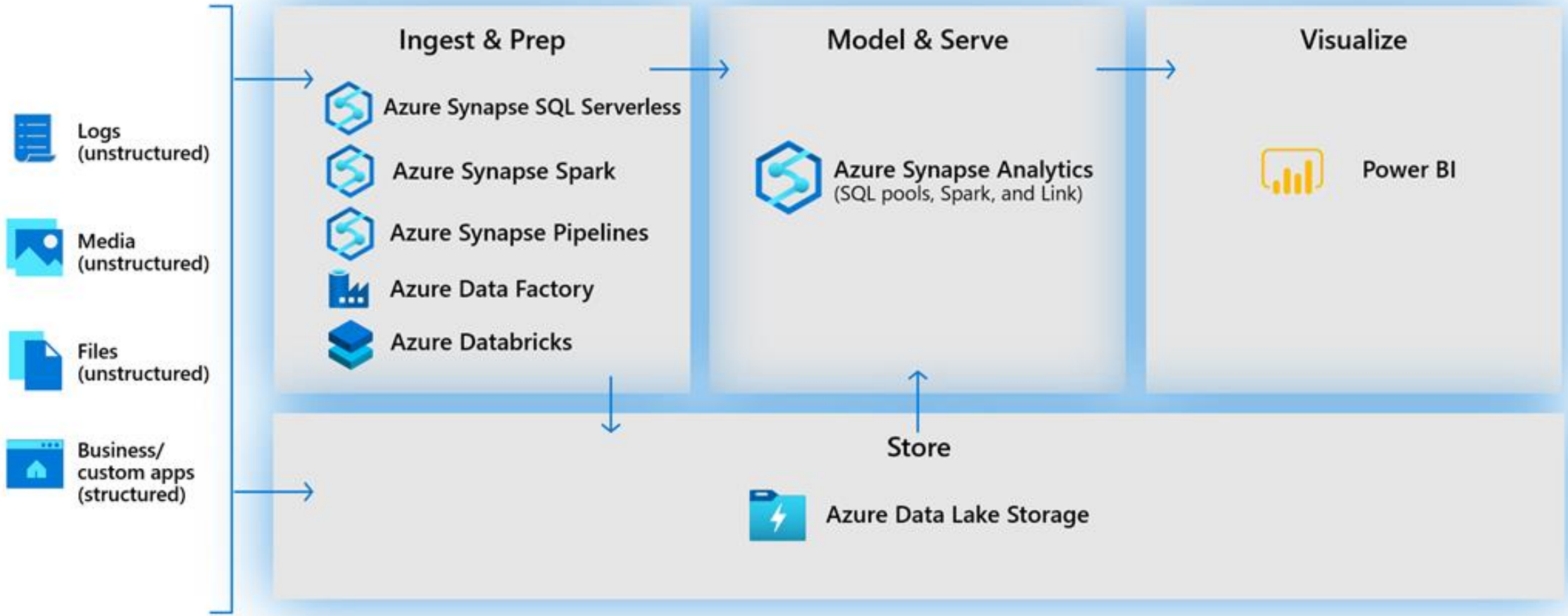


https://gigaom.com/report/data-warehouse-cloud-benchmark/

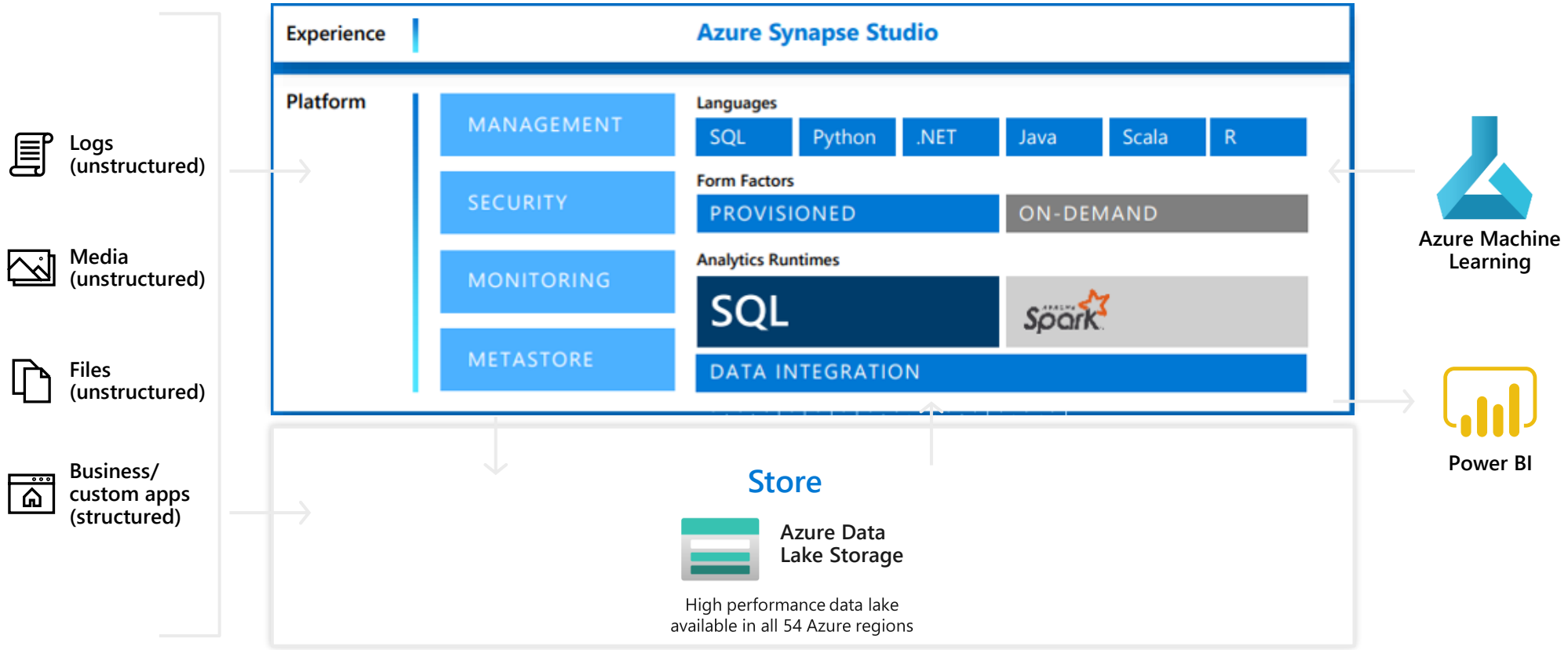
Introduction to Azure Synapse Analytics



Design a Modern Data Warehouse using Azure Synapse Analytics

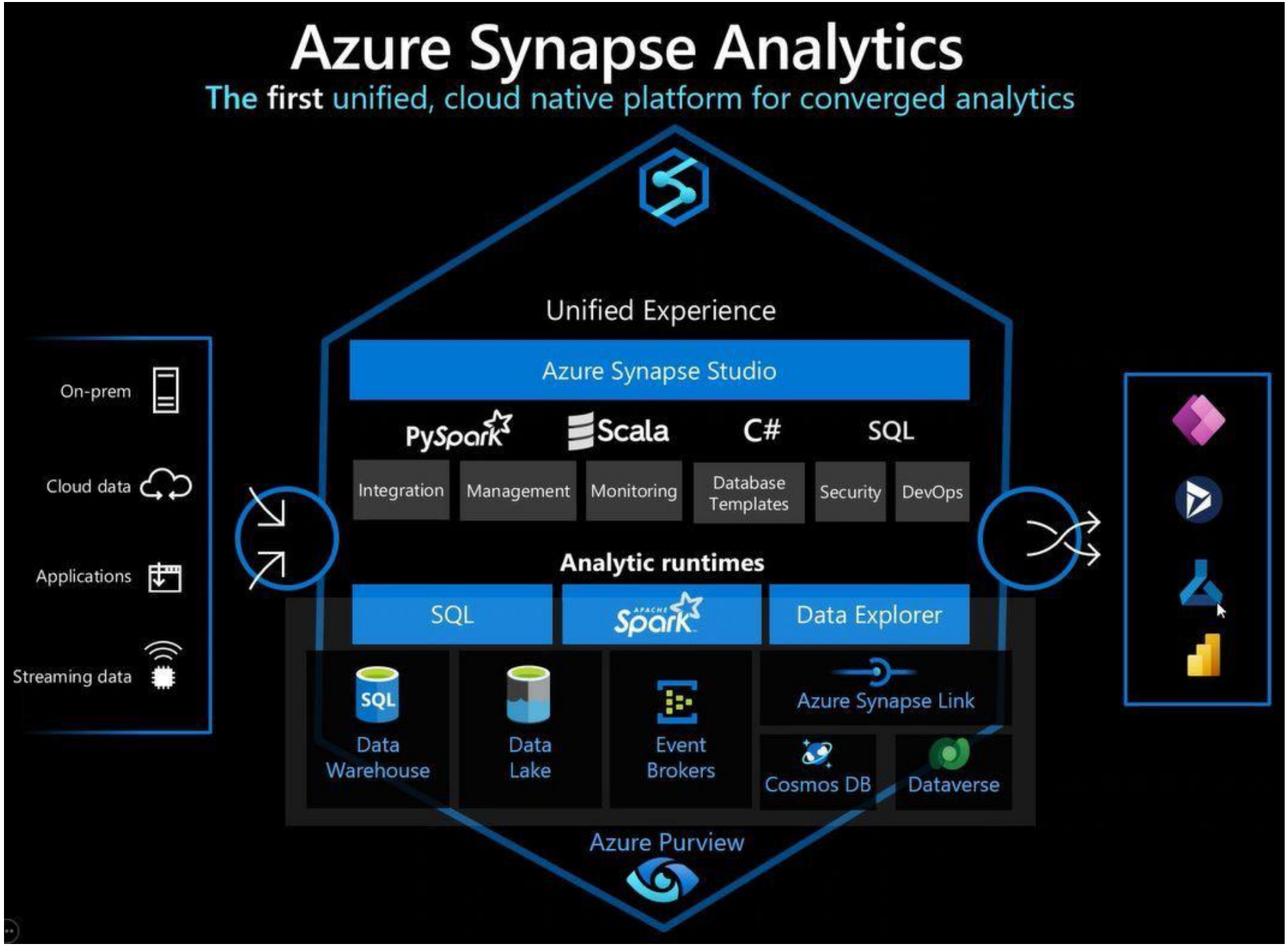


Azure Synapse Analytics Evolution (Workspace)



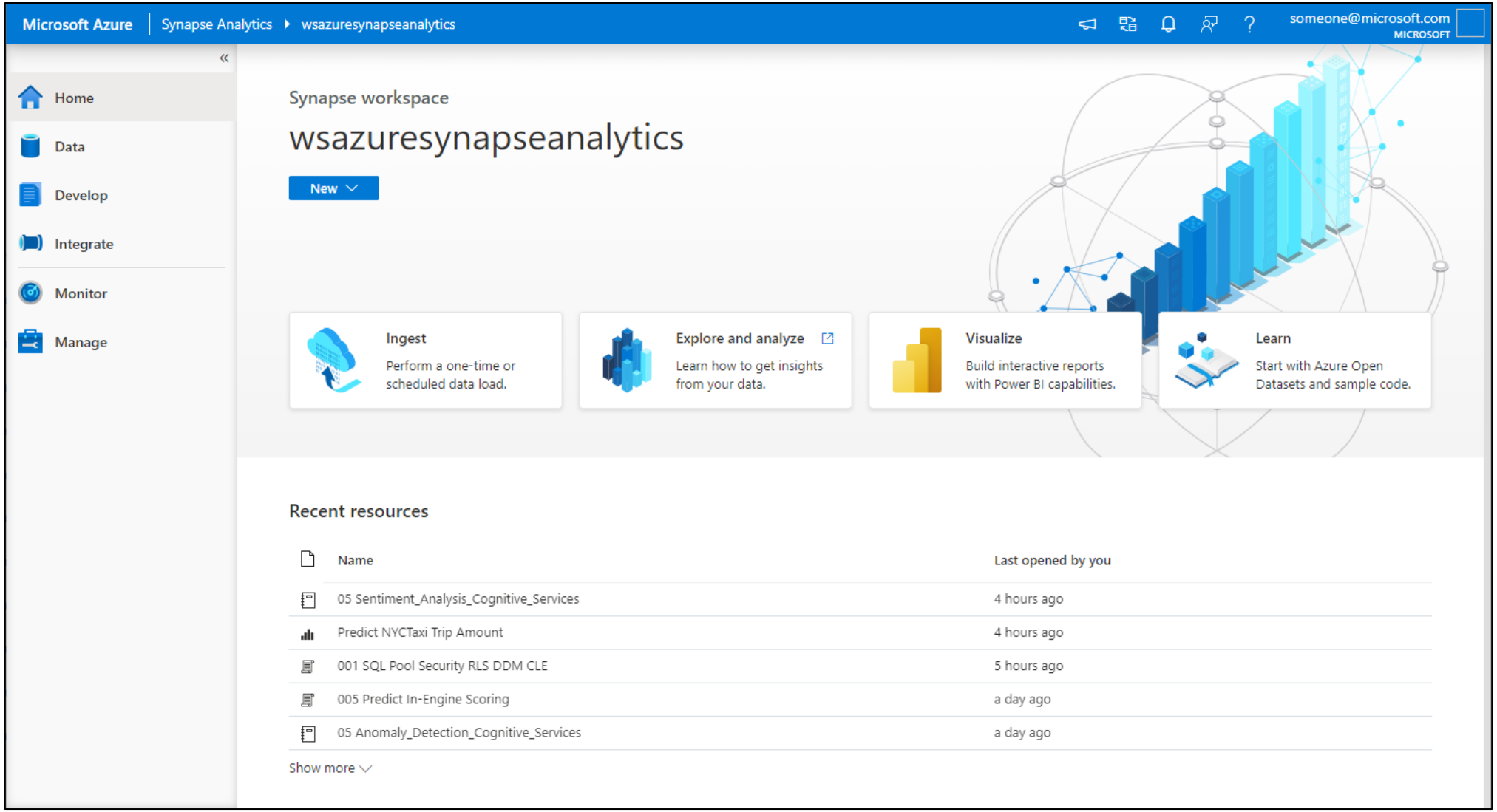
Azure Synapse Analytics

The first unified, cloud native platform for converged analytics



Studio

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



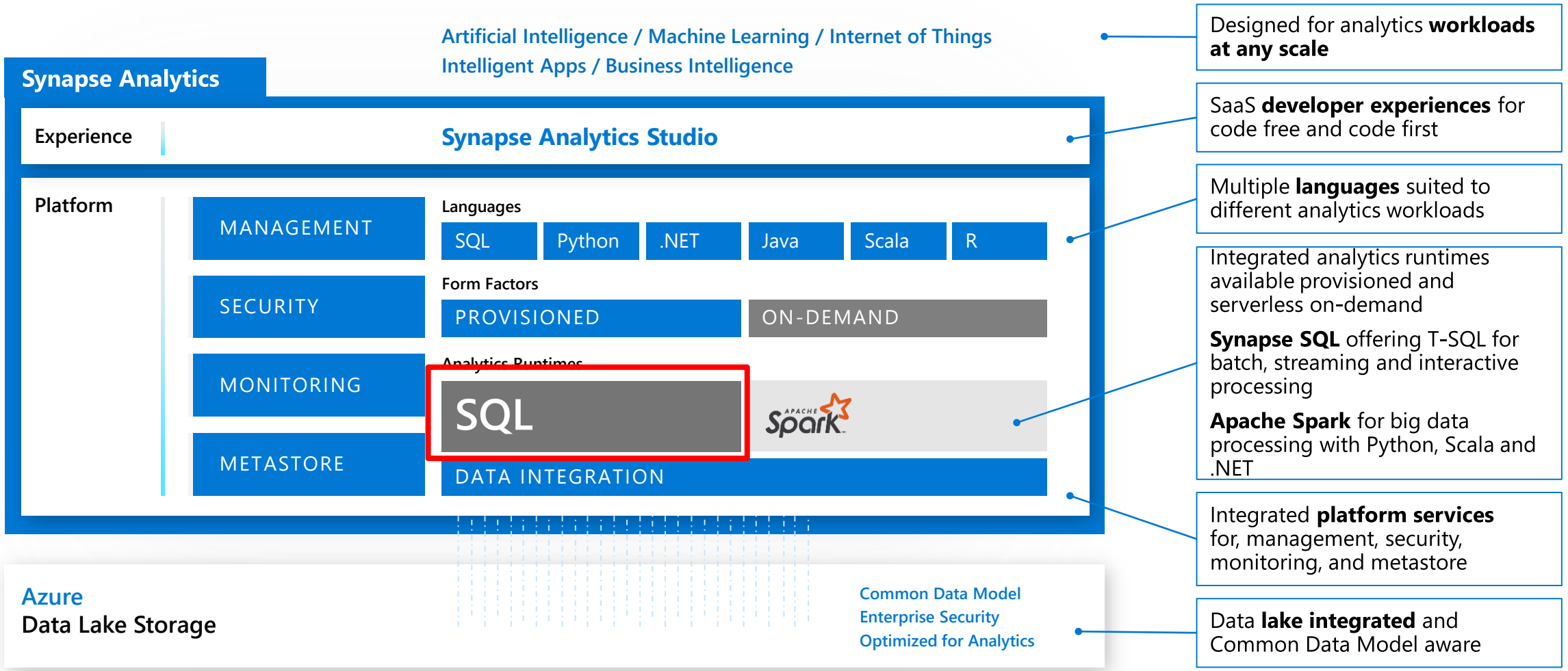
Microsoft

Azure Synapse Analytics

Synapse SQL

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Key features

Rich surface area

- T-SQL language for data analytics
- Supporting large number of languages and tools
- Enterprise-grade security

SQL Provisioned

- Modern Data Warehouse
- Indexing and caching
- Import and query external data
- Workload management

SQL Serverless

- Querying external data
- Model raw files as virtual tables and views
- Easy data transformation

Window 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, UNBOUNDING 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

Window 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

Window 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

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

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, () );
```

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

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

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 }

Synapse SQL - clients and tools

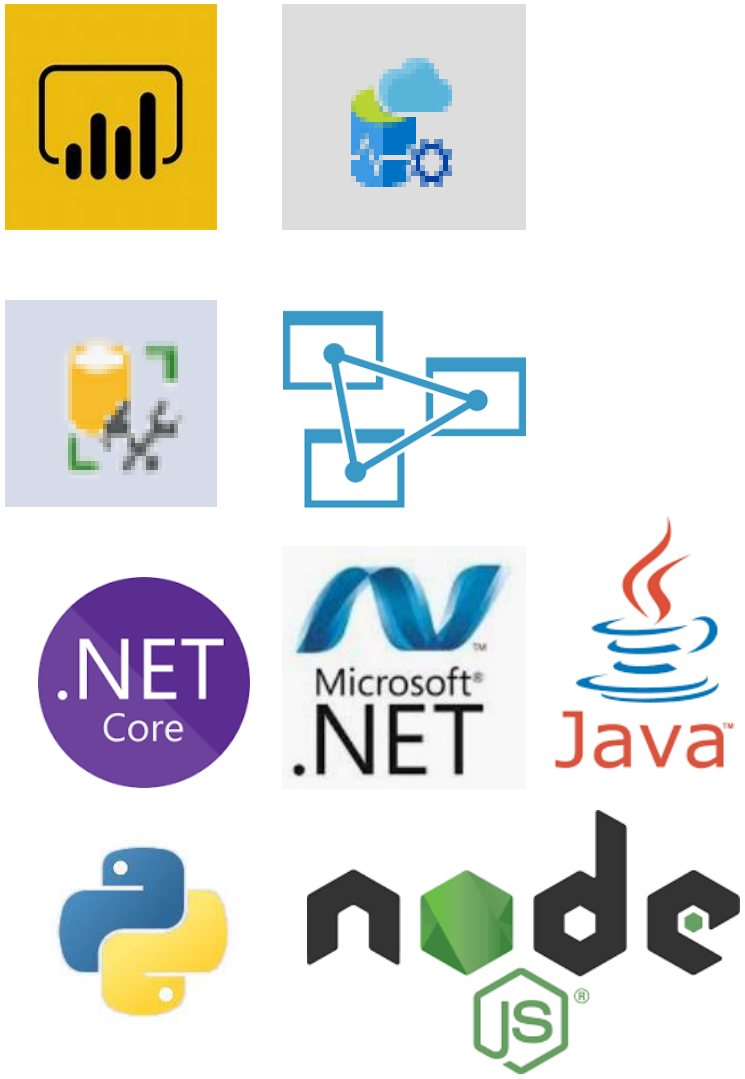
Tools

Web IDE - Synapse Studio

Client tools - Azure Data Studio, SSMS,

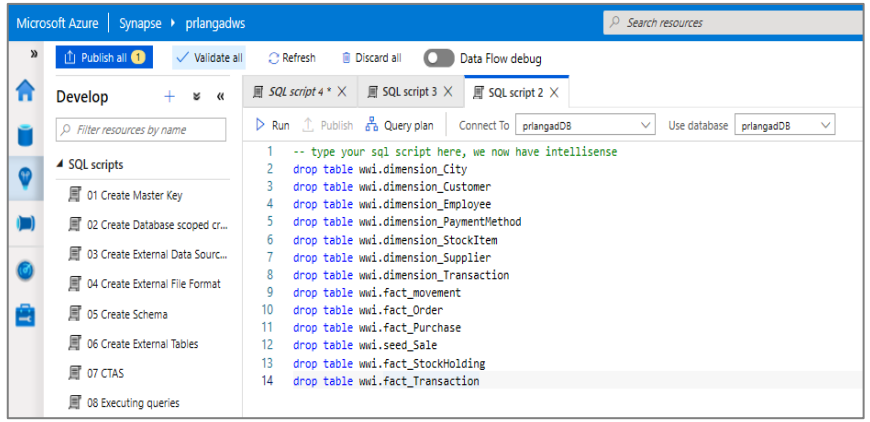
Any tool/library that uses standard SQL can access SQL serverless

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

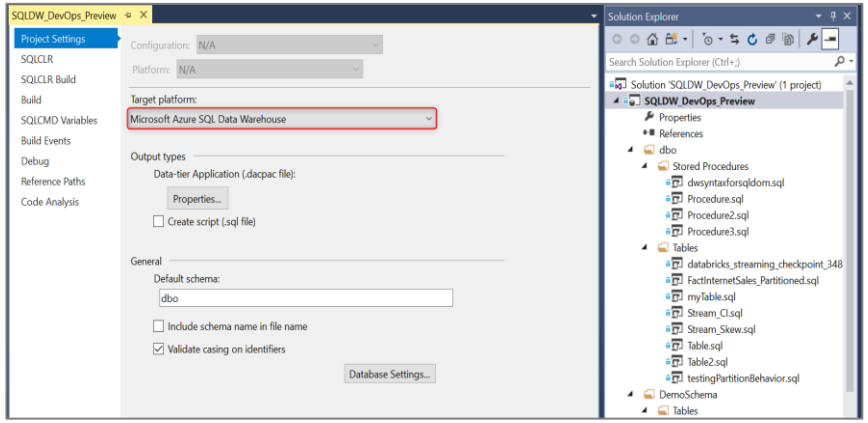


Developer Tools

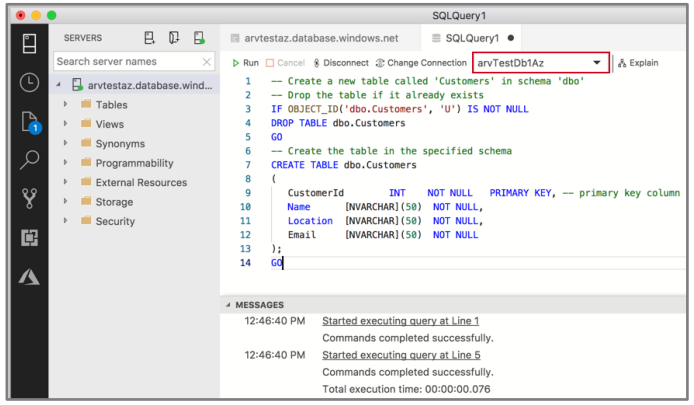
Azure Synapse Analytics



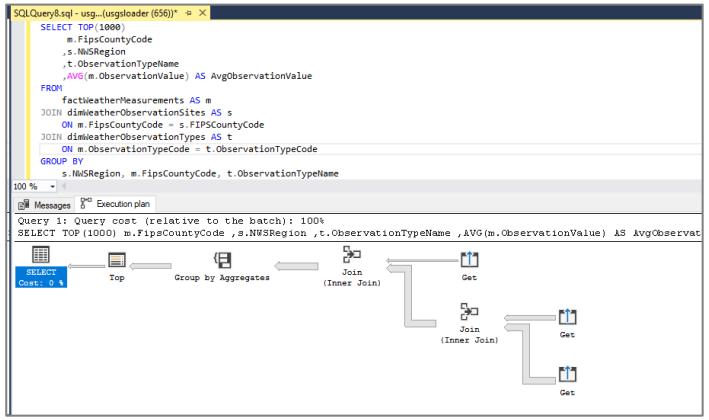
Visual Studio - SSDT database projects



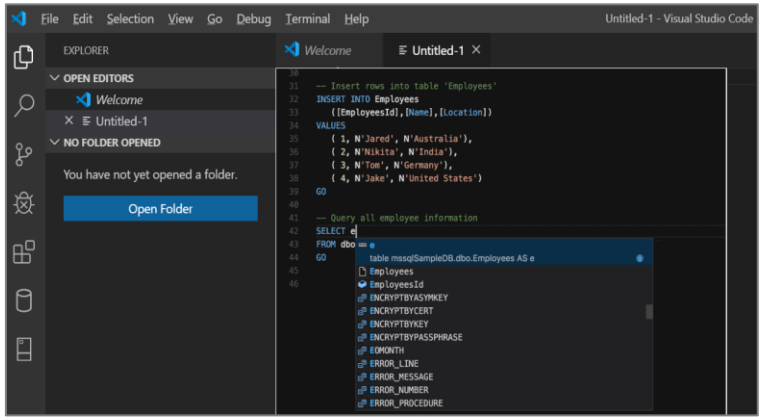
Azure Data Studio (queries, extensions etc.)



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



Visual Studio Code



Azure Synapse Analytics

Synapse SQL (serverless model)

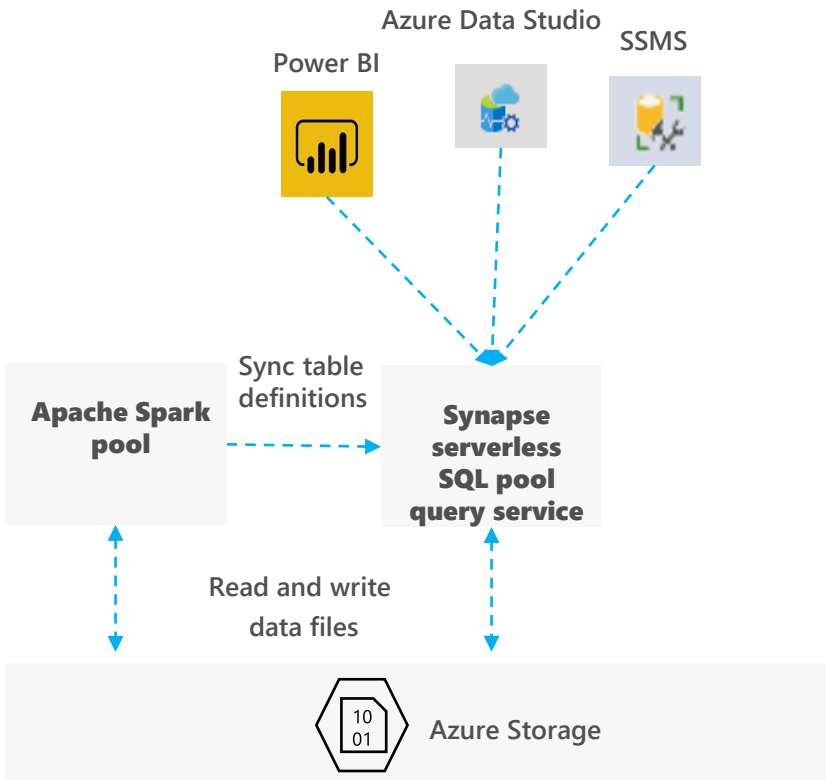
serverless SQL pool

Overview

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

Benefits

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



Recommended usage scenarios

Quick data exploration

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

Logical Data Warehouse

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

Easy data transformation

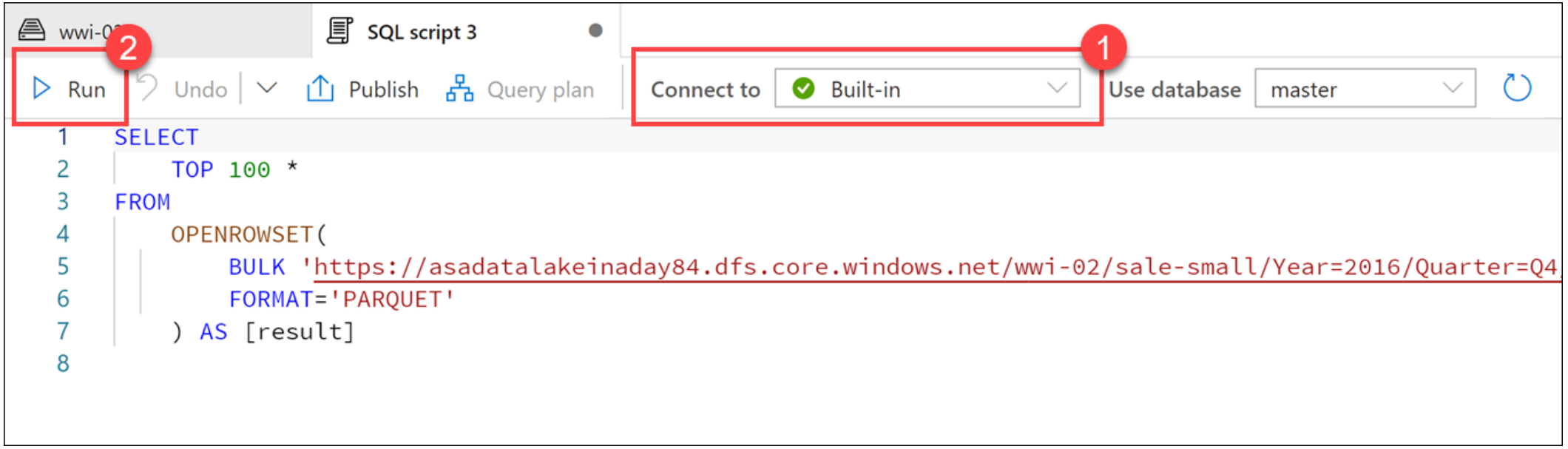
- Transform CSV to parquet format
- Move data between containers and accounts
- Save the results of queries on external storage

Query data in the lake using Azure Synapse serverless SQL pools

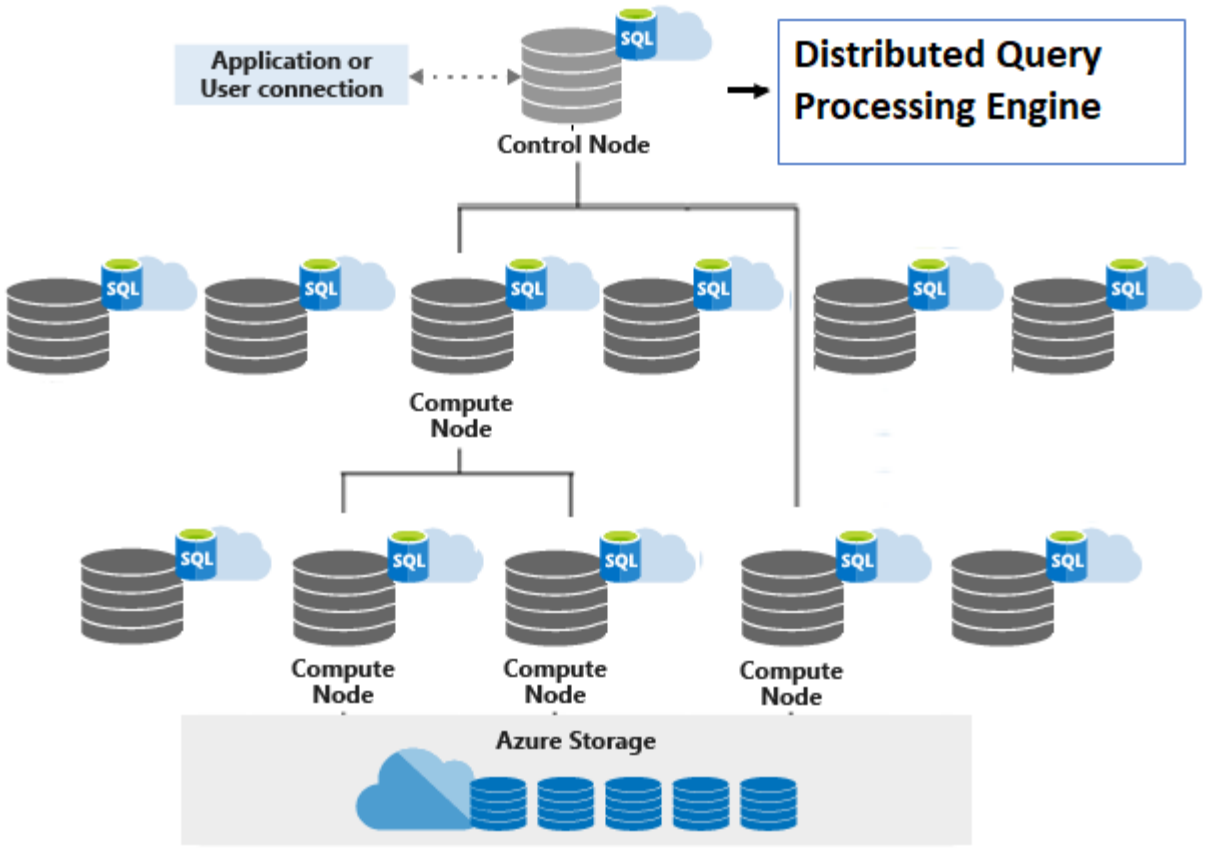
Parquet

Json

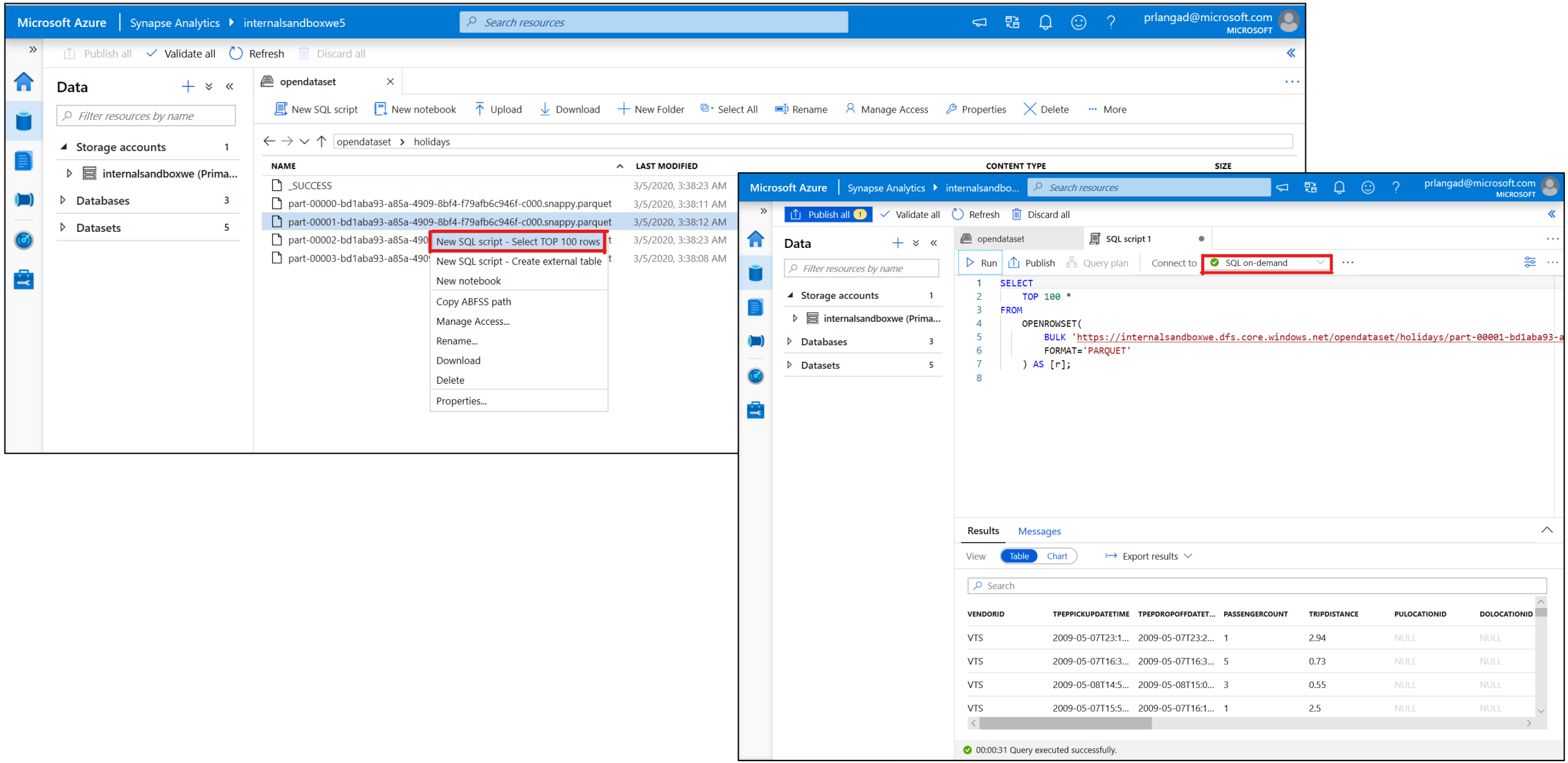
DelimitedText



serverless SQL pool architecture



Easily explore files on storage



Easily query files in various formats

Overview

Use OPENROWSET function to access data stored in various file formats

Benefits

Enables you to read CSV, parquet, and JSON files

Provides unified T-SQL interface for all file types

Use standard SQL language to transform and analyze returned data

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

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

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Automatic schema inference

Overview

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

Benefits

No need to up-front analyze file structure to query the file

OPENROWSET identifies columns and their types based on underlying file metadata.

Perfect solution for data exploration where schema is unknown.

The functionality is available for both parquet & CSV files.

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

```
SELECT
  TOP 100 *
FROM
  OPENROWSET(
    BULK 'https://azuresynapsesa.dfs.core.windows.net/default/RetailData/StoreDemoGraphics.csv',
    FORMAT = 'CSV',
    PARSER_VERSION='2.0',
HEADER_ROW = TRUE) AS [result]
```

StoreId	RatioAge60	CollegeRatio	Income	HighIncome15...	LargeHH	MinoritiesRatio	More1FullTime...	DistanceNeare...	SalesN
2	0.232864734	0.248934934	10.55320518	0.463887065	0.103953406	0.114279949	0.303585347	2.110122129	1.1428
5	0.117368032	0.32122573	10.92237097	0.535883355	0.103091585	0.053875277	0.410568032	3.801997814	0.6818

Defined the query result schema inline

Overview

Specify columns and types at query time.

Benefits

Define result schema at query time in WITH clause.

No need for external format files.

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

Improve performance by column elimination in parquet files.

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Customize the content parsing to fit your case

Overview

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

Benefits

Ability to read CSV files with custom format

- With or without header row
- Handle any new-line terminator (Windows or Unix style)
- Use custom field terminator and quote character
- Read UTF-8 and UTF-18 encoded files
- Use only a subset of columns by specifying column position after column types

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

Second, fourth, seventh and ninth columns are returned

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Easily query multiple files, with wildcards

Overview

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

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

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

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

Query partitioned data, using the folder structure

Overview

Uses OPENROWSET function to access data partitioned in sub-folders

Benefits

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

Eliminate sub-folders/partitions before the query starts execution

Query Spark/Hive partitioned data sets

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

GROUP BY    r.filepath() ,r.filepath(1) ,r.filepath(2)
ORDER BY    filepath
```

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

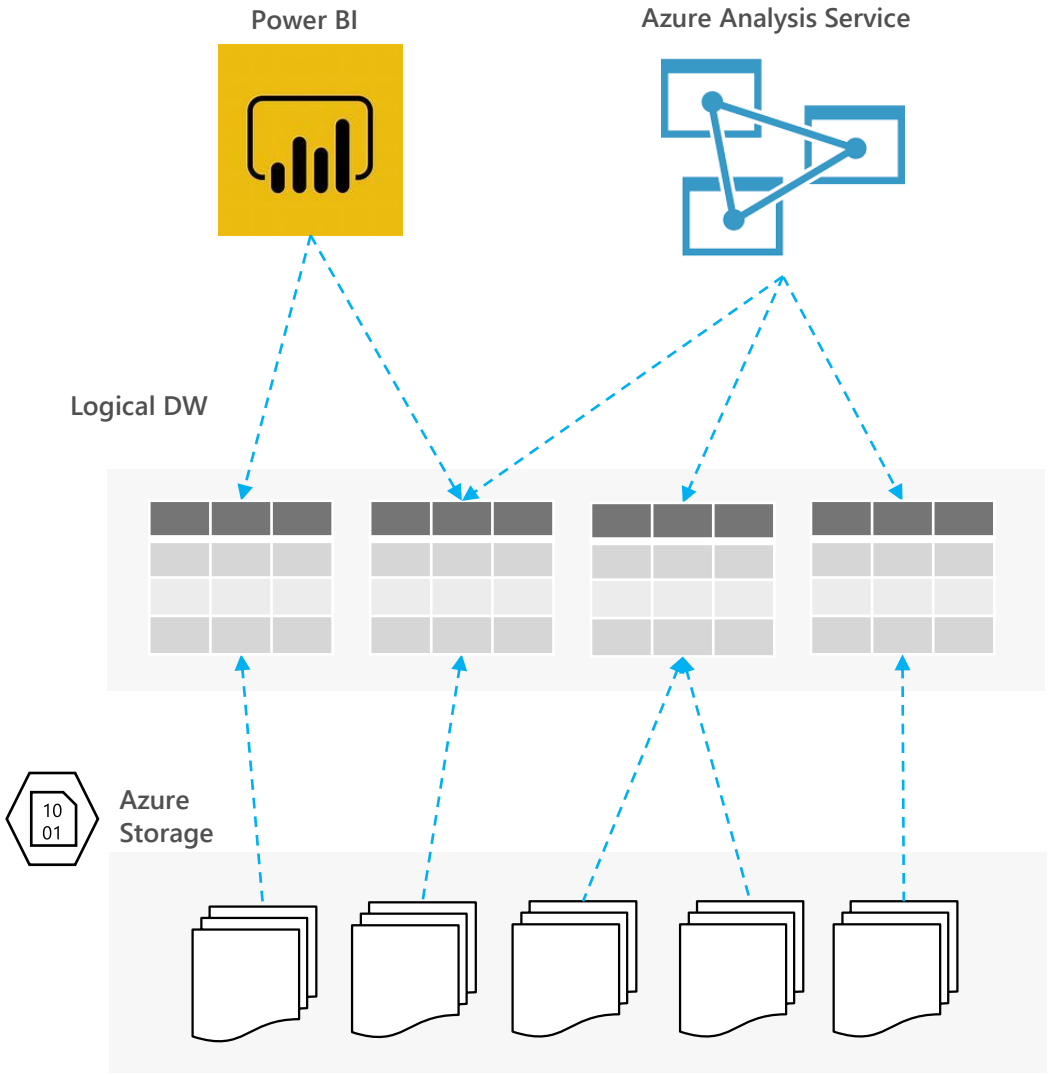
Synapse serverless SQL pool as a logical data warehouse

Overview

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

Benefits

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



Logical Data Warehouse views

Overview

serverless SQL pool logical data warehouse views are created on external files placed in customer Azure storage

Benefits

Create SQL views on externally stored data

Access files using the view from various tools and language

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

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP VIEW IF EXISTS  populationView
GO

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

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

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

Logical Data Warehouse - tables

Overview

Create external tables that reference external files in your serverless SQL pool logical data warehouse

Benefits

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

Join and transform multiple tables in the same query.

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

Manage column statistics in external tables.

Manage access rights per table.

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

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

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

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

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

Easy data transformation

Overview

Easily perform data transformations of Azure Storage files using SQL queries

Optimize data pipeline - achieve more using serverless SQL pool

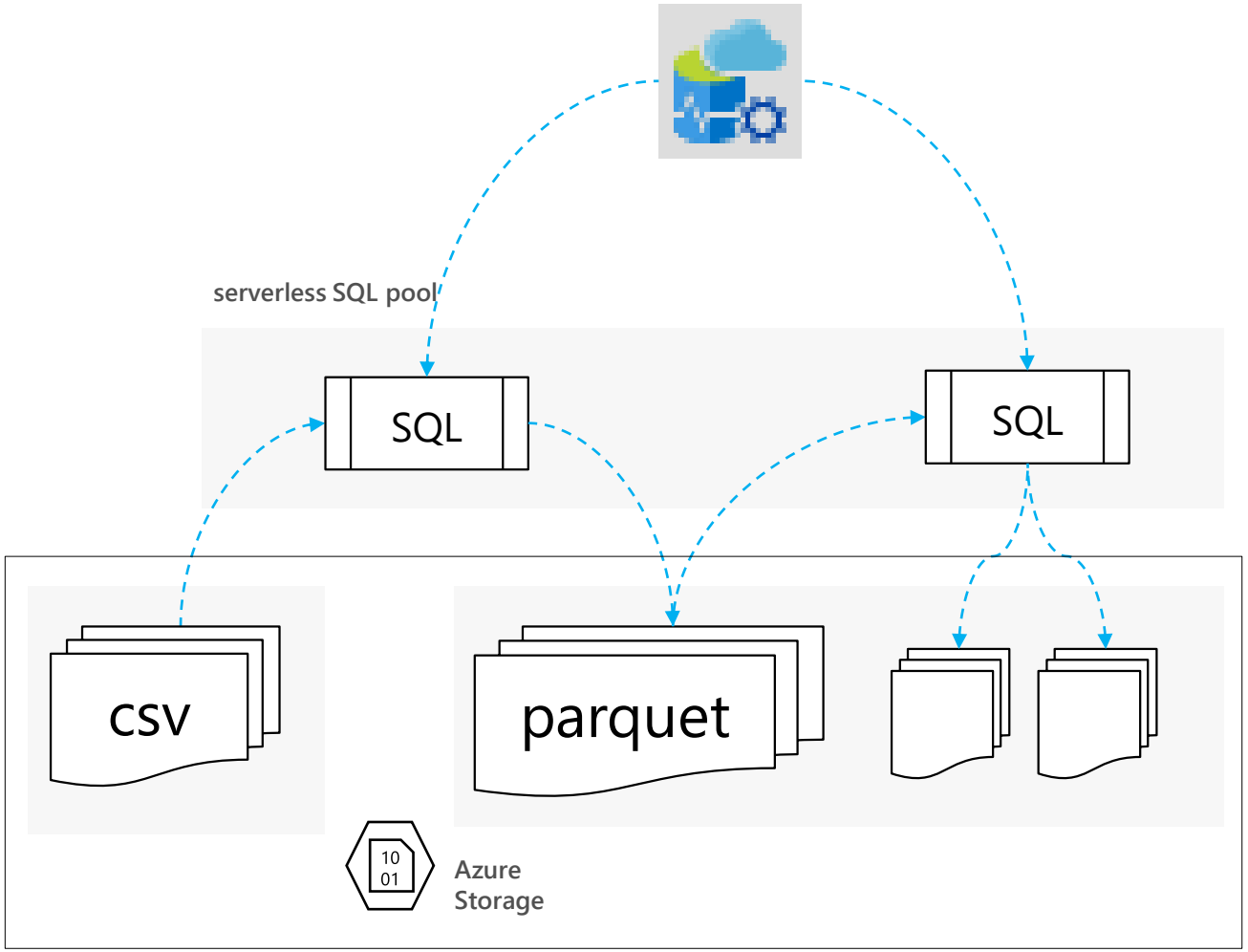
Benefits

Single statement transformations:

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

SQL ETL pipelines

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



Easy data transformation with CETAS

Overview

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

Benefits

Select any data set and store it in parquet format.

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

Use saved data using external table.

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

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

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

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

Automatic syncing of Spark tables

Overview

Tables created in Spark pool are automatically created as external tables that reference external files in your serverless SQL pool logical data warehouse

Benefits

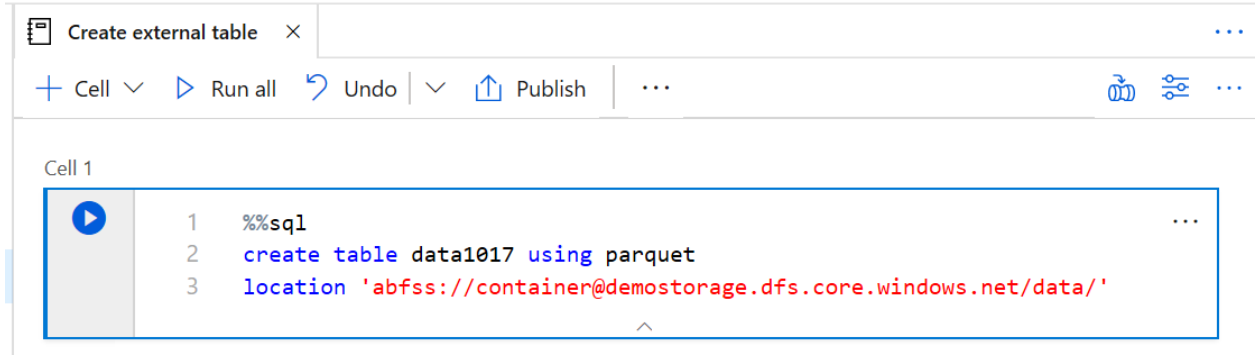
Tables designed using Spark languages are immediately available in serverless SQL pool.

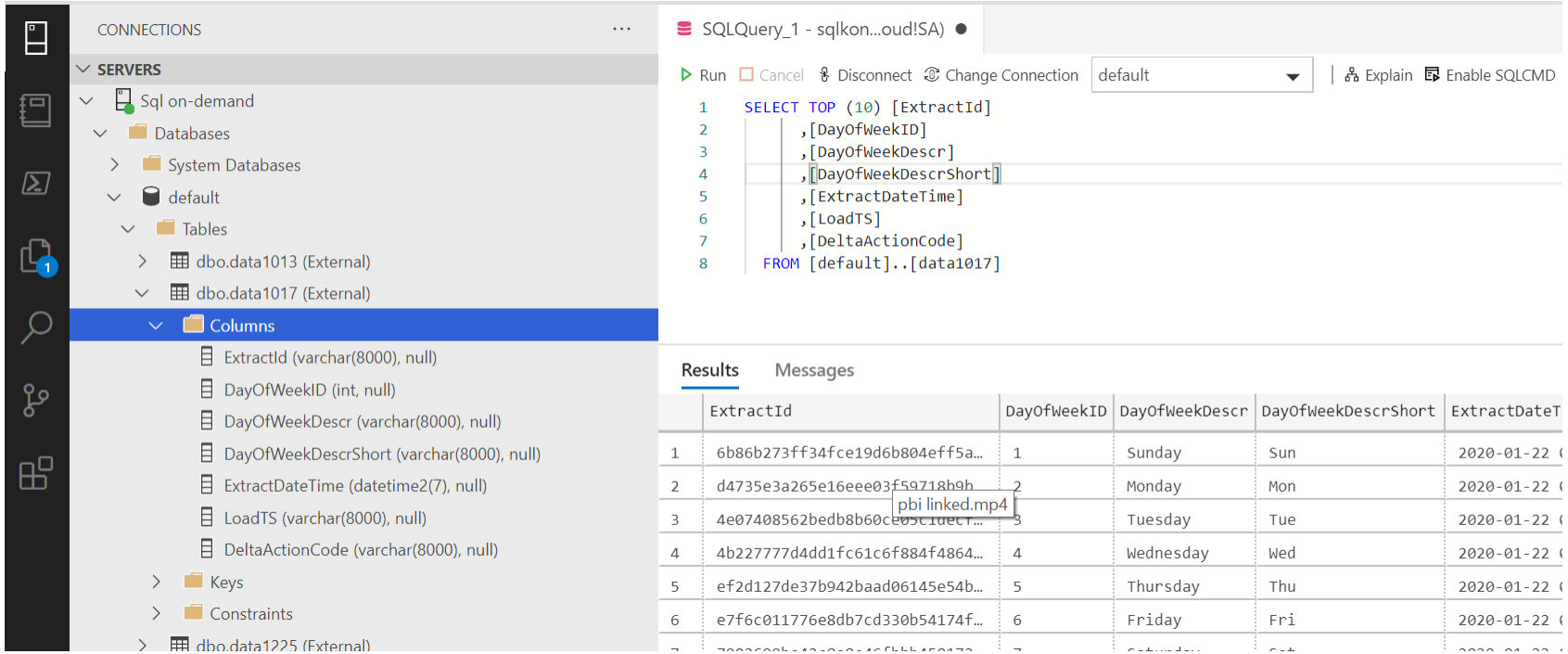
Schema definition matches original

Spark table updates are applied in serverless SQL pool

No need to manually create SQL tables that match Spark tables

Spark and serverless SQL pool tables reference the same external files.





Create metadata objects in Azure Synapse serverless SQL pools

Database

```
CREATE DATABASE [YourDatabaseName]
```

Database scoped credential

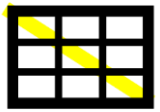
```
CREATE DATABASE SCOPED CREDENTIAL [sqlondemand]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&'
```

External data source

```
CREATE EXTERNAL DATA SOURCE SqlOnDemandDemo WITH (
    LOCATION = 'https://sqlondemandstorage.blob.core.windows.net',
    CREDENTIAL = sqlondemand );
```

External file format

```
CREATE EXTERNAL FILE FORMAT QuotedCsvWithHeaderFormat
WITH
( FORMAT_TYPE = DELIMITEDTEXT,
FORMAT_OPTIONS ( FIELD_TERMINATOR = ',', STRING_DELIMITER = '"',
FIRST_ROW = 2 ) );
```



External Table

```
CREATE EXTERNAL TABLE populationExternalTable
( [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
[year] smallint, [population] bigint )
WITH
( LOCATION = 'csv/population/population.csv',
DATA_SOURCE = sqlondemanddemo,
FILE_FORMAT = QuotedCSVWithHeaderFormat );
```

Securing access to data in a data lake when using Azure Synapse Analytics

Best Practices - Serverless SQL Pools

Co-locate storage and serverless SQL pools

Consider Azure Storage throttling

Prepare files for querying (CSV, JSON -> Parquet)

Push wildcards to lower levels in the path

Use appropriate data types and check inferred data types

Use filename and filepath functions to target specific partitions

Use PARSER_VERSION 2.0 to query CSV files

Use CETAS to enhance query performance and joins

Choose SAS credentials over Azure AD pass-through (for now)

References

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/on-demand-workspace-overview

https://techcommunity.microsoft.com/t5/azure-synapse-analytics/serverless-architecture-and-concepts-what-is-it/ba-p/2078800

https://www.microsoft.com/en-us/research/publication/polaris-the-distributed-sql-engine-in-azure-synapse/

https://www.vldb.org/pvldb/vol13/p3204-saborit.pdf

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/create-use-external-tables

https://www.dataplatformschool.com/blog/synapse-databricks-benchmark/

Microsoft

Azure Synapse Analytics

Metastore

Metastore

Overview

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

Benefits

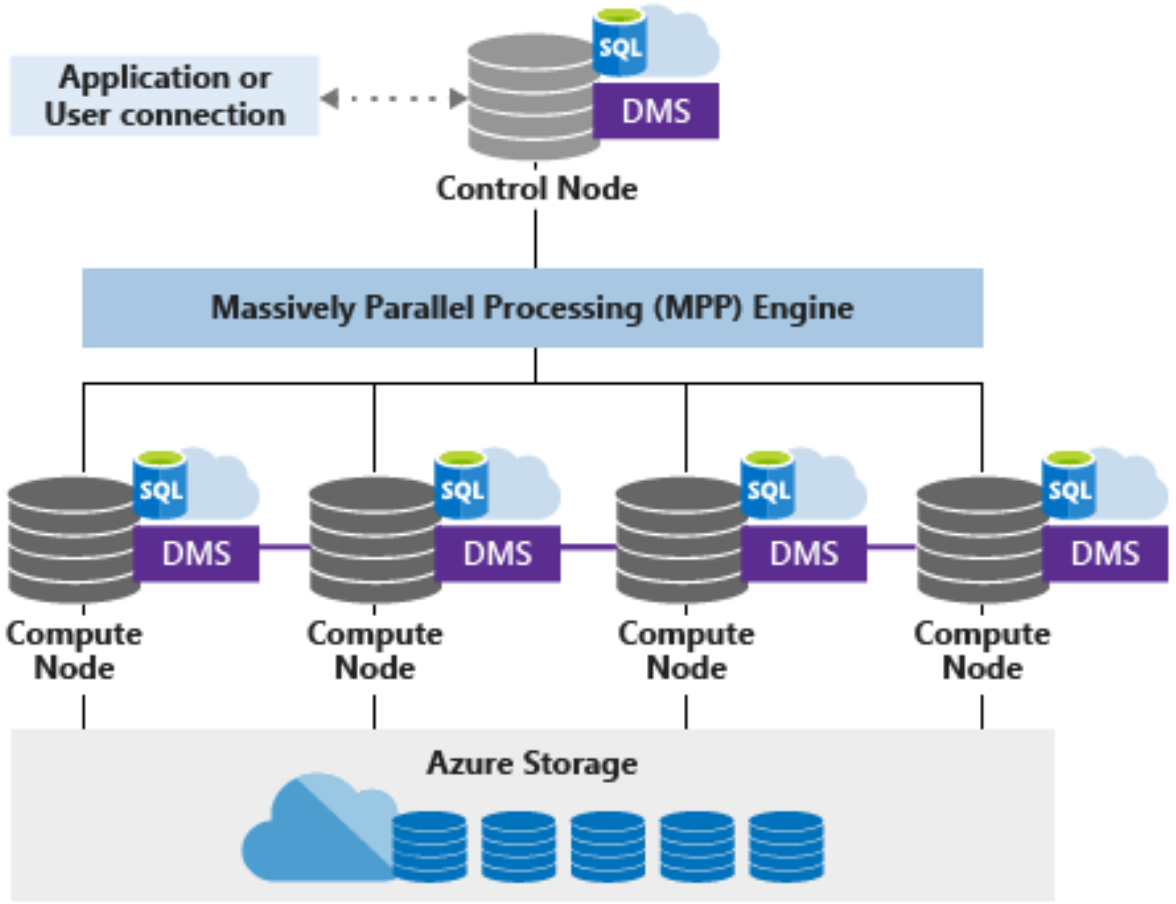
- The shared metadata model supports the modern data warehouse pattern.
- The Spark created databases and all their tables become visible in any of the Azure Synapse workspace Spark pool instances and can be used from any of the Spark jobs provided necessary permissions are provided.
- Databases are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

Microsoft

Azure Synapse Analytics

Synapse SQL (Dedicated model)

Dedicated SQL pool architecture



SQL Dedicated Pool Architecture

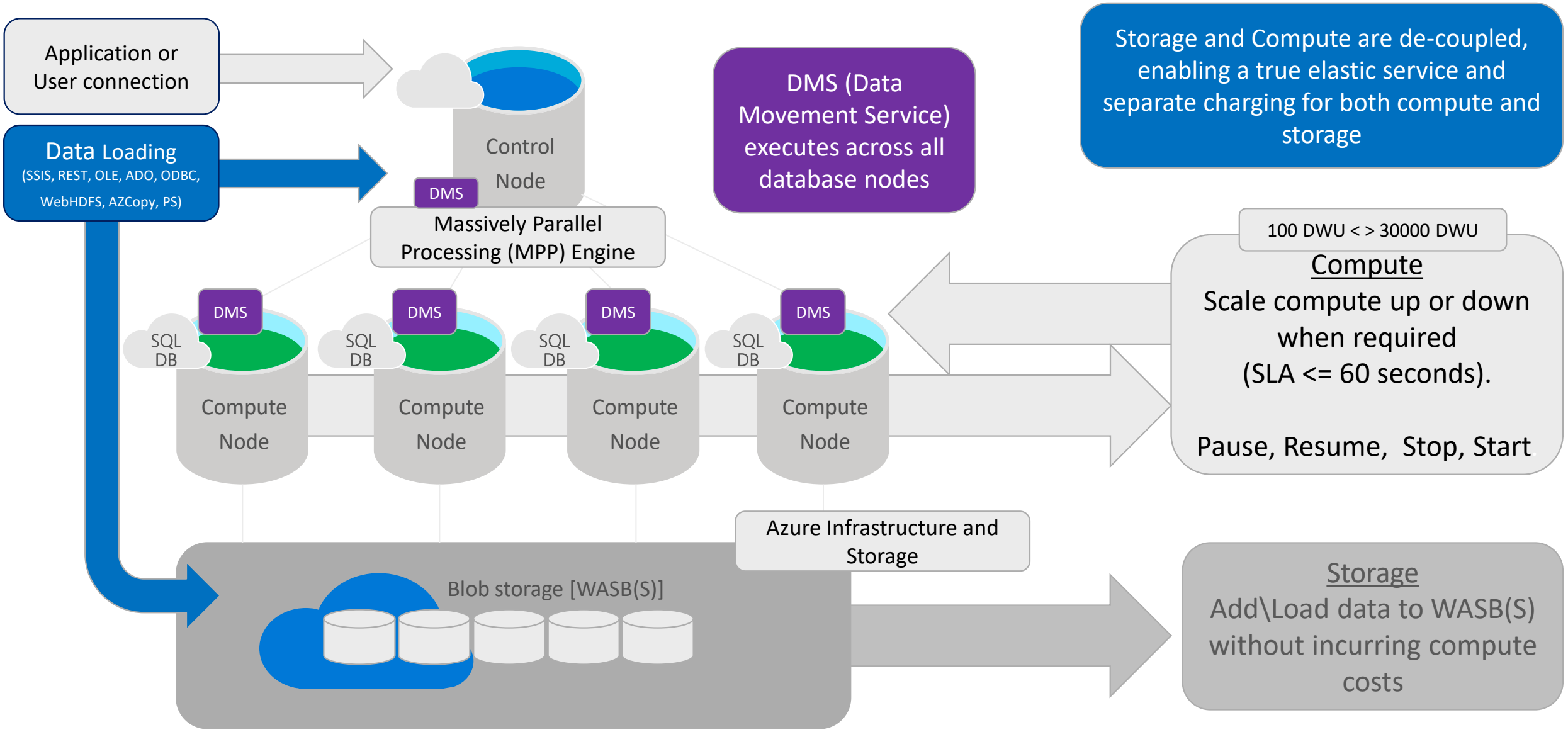


Table design

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

Table design performance considerations

Table Distribution	Recommended Use
Hash distribution	Tables that are larger than 2 GBs with infrequent insert/update/delete operations, works well for large fact tables in a star schema
Round robin distribution	Default distribution, when little is known about the data or how it will be used. Use this distribution for staging tables.
Replicated tables	Smaller lookup tables, less than 1.5 GB in size.

Common table distribution methods

Table Category	Recommended Distribution Option
Fact	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. Operations that benefit: COUNT(DISTINCT(<hashed_key>)) OVER PARTITION BY <hashed_key> most JOIN <table_name> ON <hashed_key> GROUP BY <hashed_key>
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.

Index design

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);
```

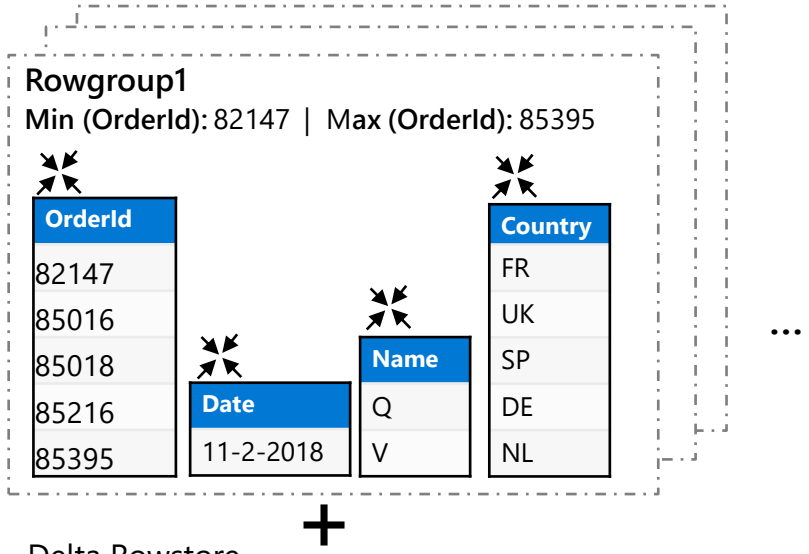
SQL Analytics Columnstore Tables

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

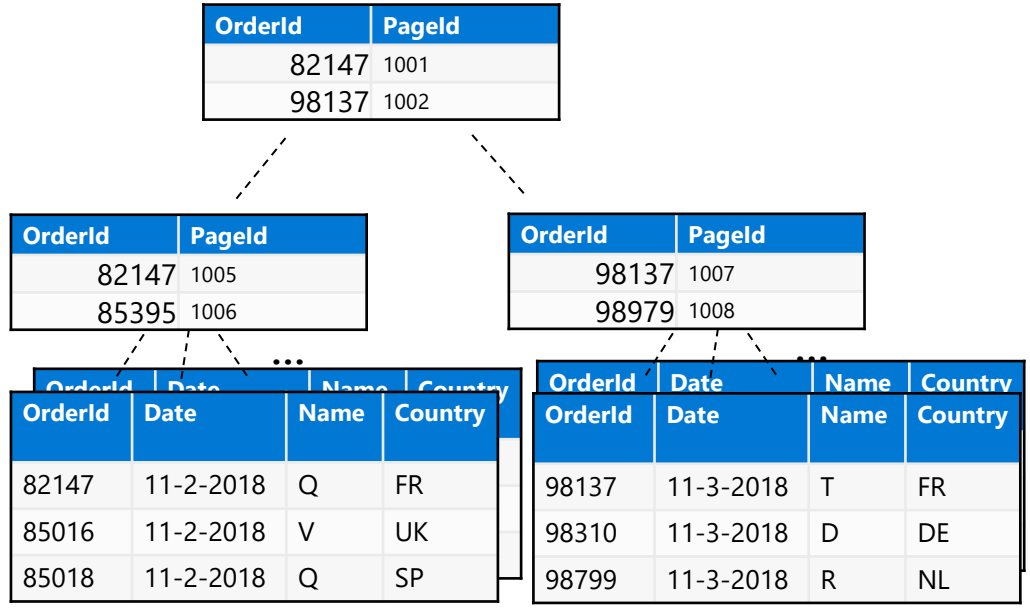
(OrderId)



- 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

The problem with CCI

The index builder doesn't sort data before compressing it into segments, which means that segments with overlapping value ranges could occur.

When this happens, queries might read more segments from disk and take longer to finish.

The solution

Ordered CCIs – data is sorted in memory by the order key(s) before the index builder compresses the segments and thus segment overlapping is reduced.

```
-- 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')
```

Ordered CCI

- Queries against tables with ordered columnstore segments can take advantage of improved segment elimination to drastically reduce the time needed to service a query.
- Columnstore Segments are automatically updated as data is inserted, updated, or deleted in data warehouse tables.
- New data resulting from the same batch of DML or data loading operations is sorted within that batch (no global sorting across all data in the table).
- REBUILD can be used to sort all data in the table (which is an offline operation, done one partition at a time).

Ordered CCI - Performance

The number of overlapping segments depends on:

-The size of data to sort

-Available memory (use XLARGERC on a higher DWU to allow more memory for data sorting before compressing occurs)

-MAXDOP (create ordered CCI with MAXDOP =1 as each thread used for ordered CCI works on a subset of data and sorts it locally)

Pre-sort the data by the sort key(s) before loading

Choosing the right index

Clustered Columnstore indexes (CCI) are best for fact tables.

CCI offer the highest level of data compression and best query performance for tables with over 100 million rows.

Heap tables are best for small lookup tables and recommended for tables with less than 100 million rows.

Clustered Indexes may outperform CCI when very few rows need to be retrieved quickly.

- Add non-clustered indexes to improve performance for less selective queries.
- Each additional index added to a table increases storage space required and processing time during data loads.

Speed load performance by staging data in heap tables and temporary tables prior to running transformations.

Data Loading

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

Create External Table As Select (Polybase)

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 );
```

Polybase vs Copy

Polybase

- GA, stable
- Needs CONTROL permission
- Enables querying via external tables
- Row width
- Delimiters in text
- Fixed line delimiter
- Code complexity

Copy

- GA, stable
- Relaxed permission
- Slightly slower, but improving
- No row width limit
- Supports delimiters in text
- Supports custom column and row delimiters

Best Practices

Ingest - Structuring ADLS Gen2

- Separate storage accounts for each environment: dev, test, & production.

- Use a common folder structure to organize data by degree of refinement.

ADLS Gen 2 Filesystem

Ingest from on-premises data sources

Fastest is done by batch:

- Extract from data source to multiple CSV/Parquet files
- Use AzCopy to upload to ADLS'

Alternative is query-insert:

- Set up SSIS self-hosted integration runtime on-premises
- Use Synapse Pipeline to extract/copy
- Use Synapse Pipeline to execute load procedure

Large Migrations:

- Use Azure Data Box where available

Ingest from Cloud Data Sources

Options:

- Extract using Synapse Pipelines
- Write to ADLS as Parquet files
- AzCopy is a fast move for files from S3 to ADLS

Ingest File Data Sources

Look out for these file format challenges…

Invalid file format

- Multiple row types
- Ragged columns

Row size > 1Mb

Datetime format/s (e.g., use of nanosecond date time)

NULL value literal/s

Free form text

Parquet partitions

XML data

Use of non-standard line delimiters (e.g., CR)

…and try these Solutions

- Use Spark to pre-process and fix data errors
- Flatten and parse XML in Spark
- Use COPY to ingest complex CSV instead of Polybase

Ingest and Store – Formats

For batch flat files, Azure Synapse Analytics supports CSV, Parquet, ORC, and JSON formats.

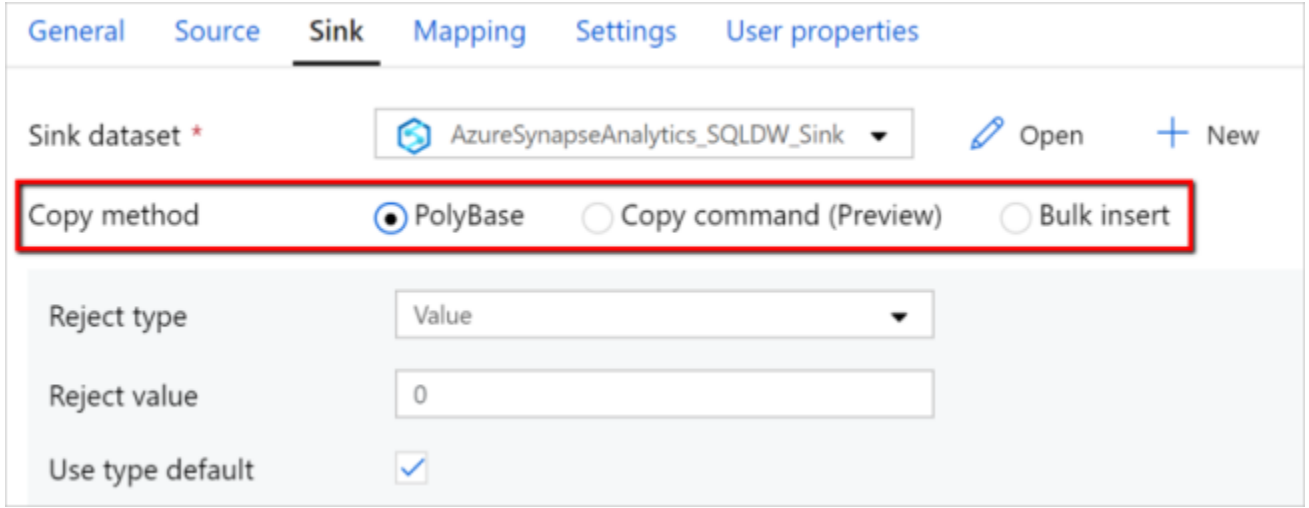
Ingest streaming data messages/events via Event Hub or IoT Hub.

Parquet format recommended for storing ingested data at various levels of refinement.

Ingest – Synapse Pipelines

- Un-check USE TYPE DEFAULT, it is not a best practice.

- Land data in ADLS Gen2, then ingest using Polybase / COPY.
  - This means you can re-ingest the same data set without having to repeat extracts, and better demonstrate ingestion performance.



Ingest and Store – Loading staging tables

Indexing

Use Heap tables

Speed load performance by staging data in heap tables and temporary tables prior to running transformations.

Only load to a CCI table if the test requires a load to a single table, then complex end-user queries against that table.

Ingest and Store – Loading staging tables

Distribution

Use Round Robin Distribution for:

Potentially useful tables created from raw input.

Temporary staging tables used in data preparation.

Other distribution considerations:

Never load to a REPLICATED table

Load to a ROUND_ROBIN table if the test is ONLY raw ingestion performance, or if the table is very small

Load to a HASH table if the task is a pipeline with subsequent transformations using the loaded table

Ingest – Scaling to shorten duration

Ingestion duration is correlated with the number of DWU's allocated to the SQL Pool.

For every *doubling* of the DWU's you *halve* the ingestion time.

2d = t/2

d: DWU

T: ingestion time

Only applies from DWU500c – DWU30000c

Export to files with CETAS

CETAS = parallel operation that creates external table metadata and exports the SELECT query results to a set of files in your storage account.

Store frequently used parts of queries, like joined reference tables, to a new set of files. You can then join to this single external table instead of repeating common joins in multiple queries.

As CETAS generates Parquet files, statistics will be automatically created when the first query targets this external table, resulting in improved performance.

Performance Patterns

Result-set caching

Overview

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

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

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

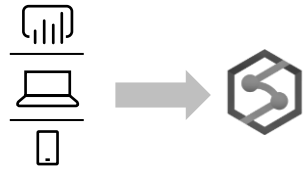
Result cache is evicted regularly based on a time-aware least recently used algorithm (TLRU).

Benefits

- Enhances performance when same result is requested repetitively
- Reduced load on server for repeated queries
- Offers monitoring of query execution with a result cache hit or miss

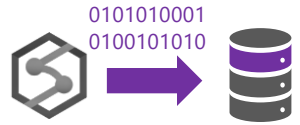
```
-- Turn on/off result-set caching for a database
-- Must be run on the MASTER database
ALTER DATABASE {database_name}
SET RESULT_SET_CACHING { ON | OFF }

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

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

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

Result-set caching flow

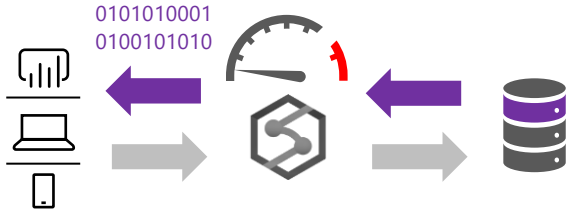


1. Client sends query to SQL pool

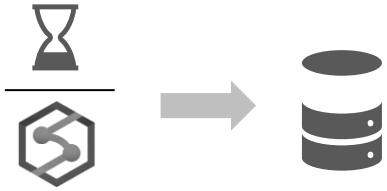
2. Query is processed using compute nodes which pull data from remote storage, process query and output back to client app



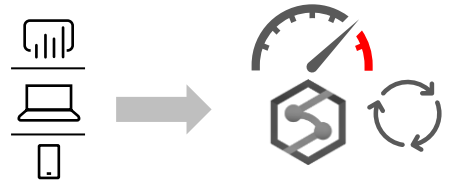
Query results are cached in remote storage so subsequent requests can be served immediately



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

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;
```

Indexed Materialized Views

- Indexed views cache the schema and data for a view in DW remote storage. They are useful for improving the performance of 'SELECT' statement queries that include aggregations

- Indexed 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 Synapse 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

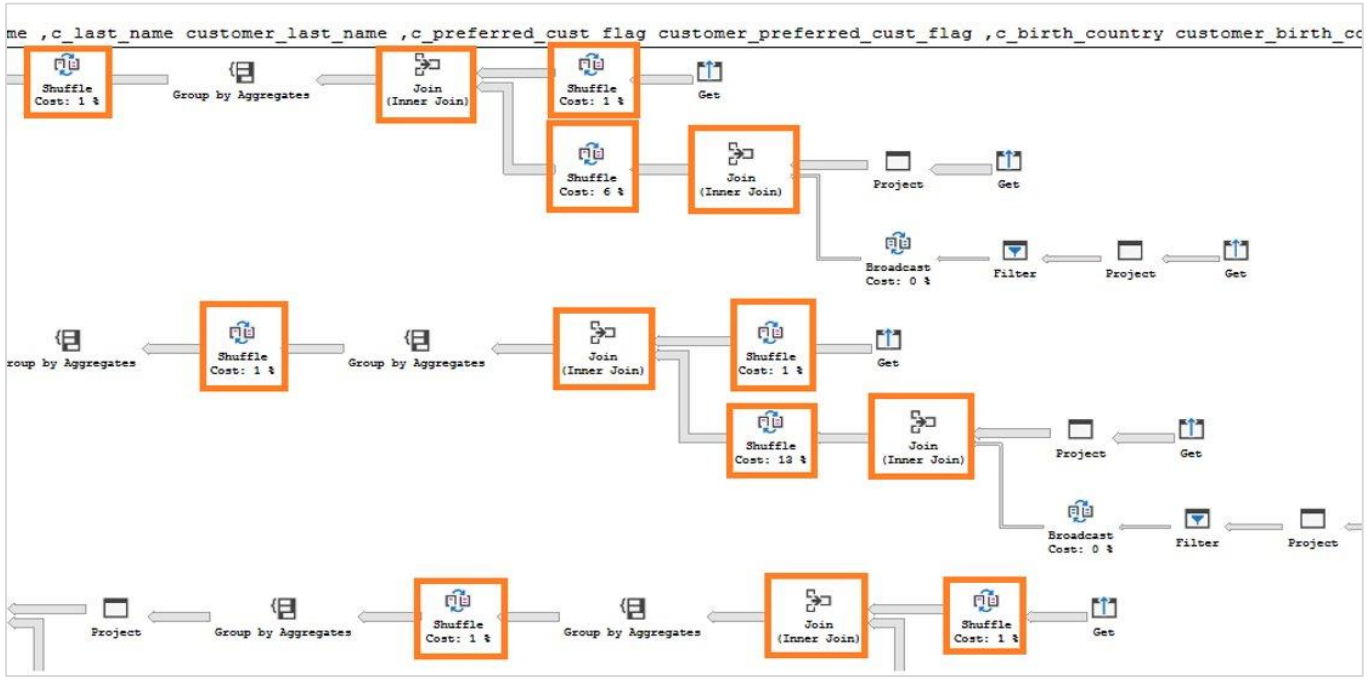
Materialized views - example

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

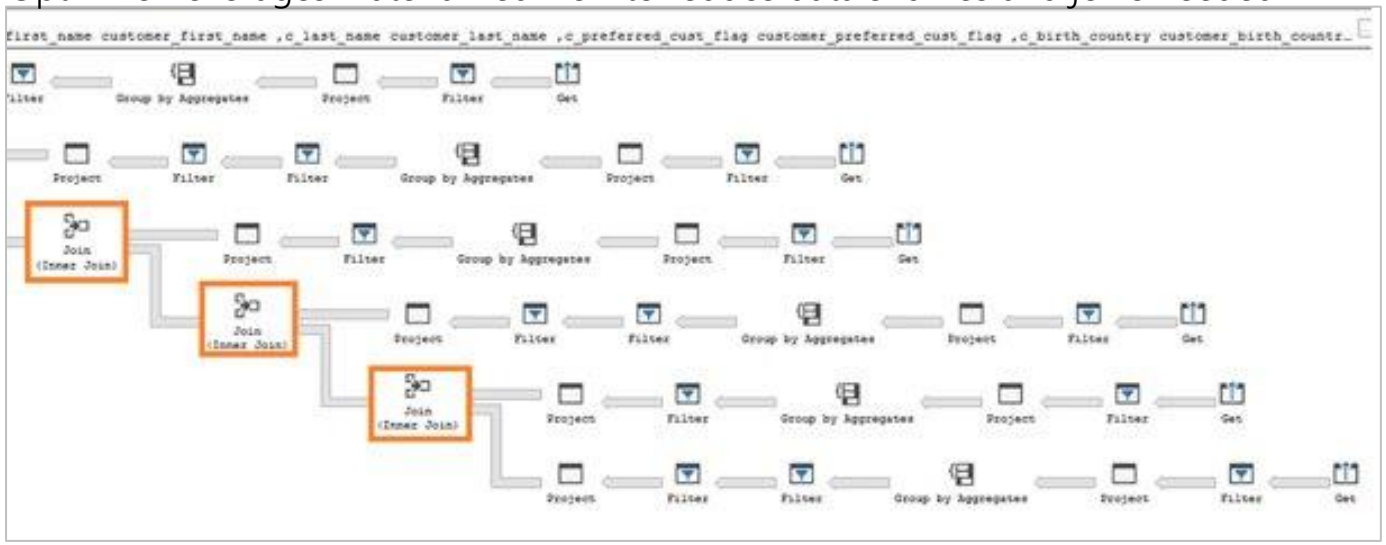
SQL pool query optimizer automatically leverages the indexed view to speed up the same query. Notice that the query does not need to reference the view directly

Original query – no changes have been made to query

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

Execution time: 6 seconds

Optimizer leverages materialized view to reduce data shuffles and joins needed



Materialized views- Recommendations

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

EXPLAIN WITH_RECOMMENDATIONS - provides query plan with recommendations to optimize the SQL statement performance.

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

Performance Anti-Patterns

Too many partitions

- Partitions can be useful when maintaining current rows in very large fact tables. Partition switching is a good alternative to full CTAS
- Partitioning CCIs is only useful when the row count is greater than 60million * #partitions
- In general, avoid partitions, particularly in POCs

Views on Views

- Views on Views will not support performance optimization using Materialized Views (more later)

- Views cannot be distributed

References

https://docs.microsoft.com/en-us/learn/modules/use-data-loading-best-practices-azure-synapse-analytics/

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-partition

https://docs.microsoft.com/pt-br/azure/synapse-analytics/sql-data-warehouse/memory-concurrency-limits?toc=/azure/synapse-analytics/toc.json&bc=/azure/synapse-analytics/breadcrumb/toc.json

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/data-loading-best-practices

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/resource-classes-for-workload-management

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-best-practices

https://docs.microsoft.com/en-us/sql/relational-databases/indexes/columnstore-indexes-overview?view=azure-sqldw-latest&preserve-view=true

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/data-load-columnstore-compression

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-overview

Monitor & Manage

Workload Management

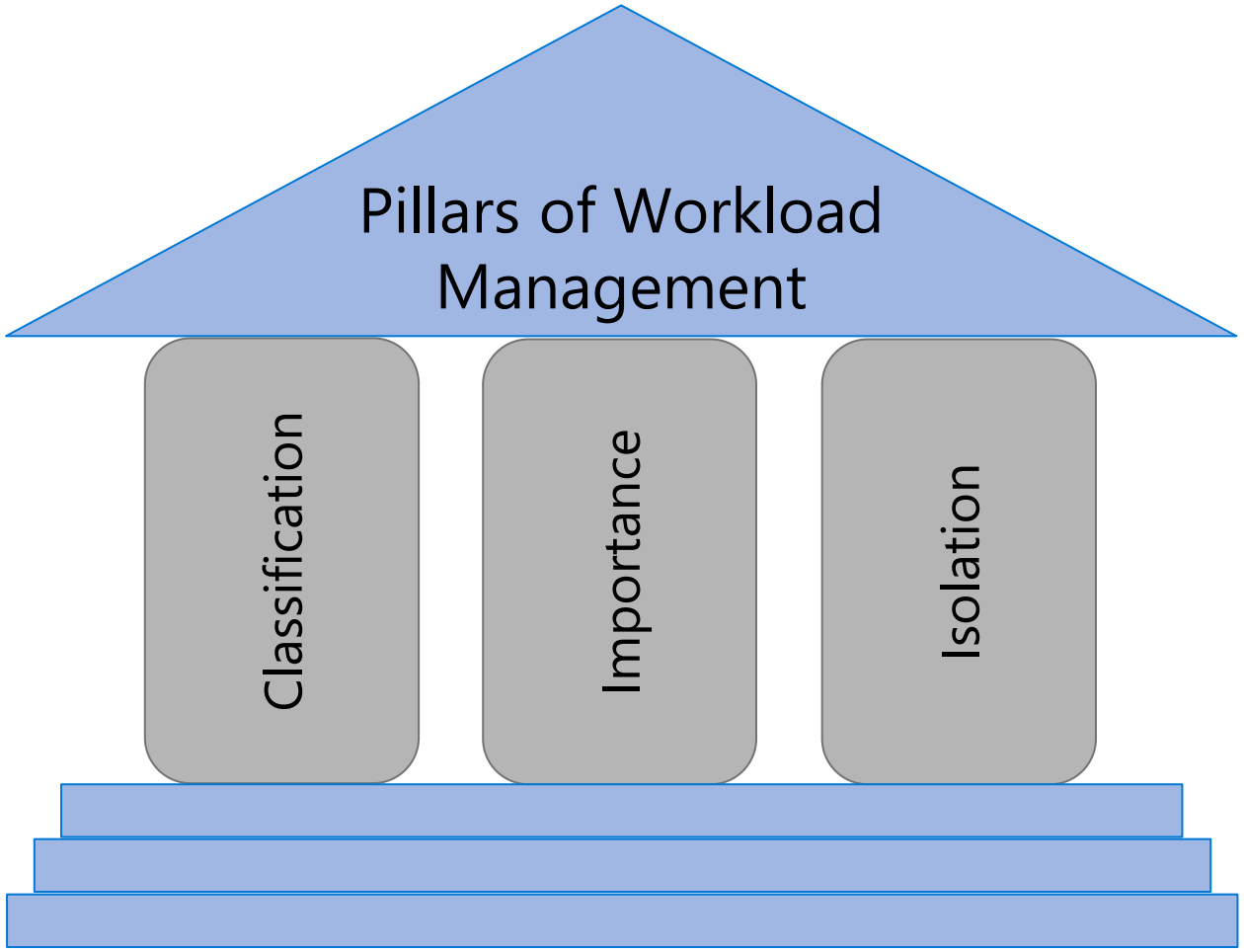
Overview

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

Synapse is moving away from Resource Class and Concurrency Slots to Workload Management.

The three pillars of workload management are

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



Workload classification

Overview

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

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

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

Benefits

Map queries to both Resource Management and Workload Isolation concepts.

Monitoring DMVs

sys.workload_management_workload_classifiers

sys.workload_management_workload_classifier_details

Query DMVs to view details about all active workload classifiers.

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

***WORKLOAD_GROUP**: maps to an existing resource class*
***IMPORTANCE**: specifies relative importance of request*
***MEMBERNAME**: database user, role, AAD login or AAD group*

Workload importance

Overview

Queries past the concurrency limit enter a FiFo queue

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

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

Example Video

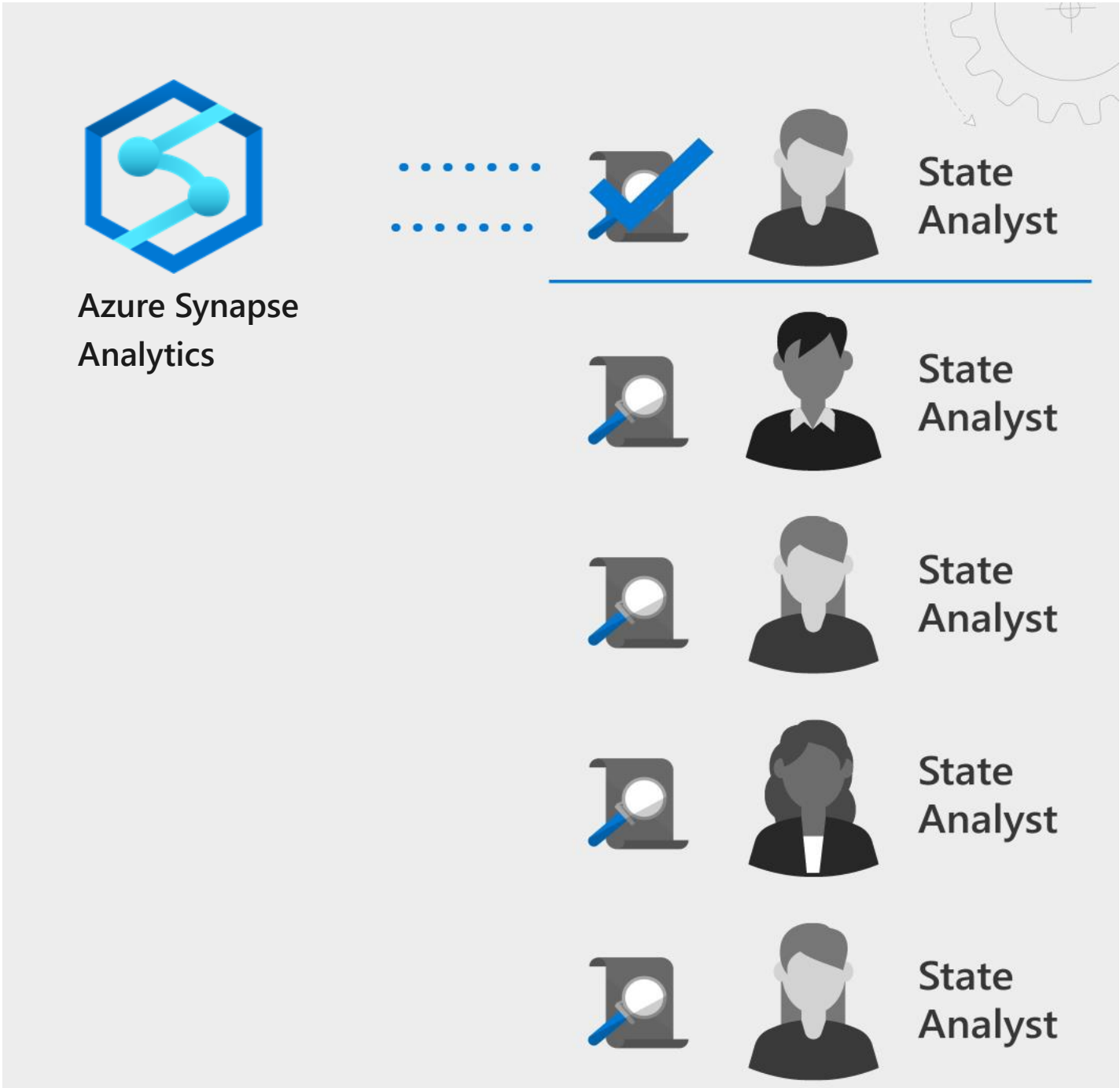
State analysts have normal importance.

National analyst is assigned high importance.

State analyst queries execute in order of arrival

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

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



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 take 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

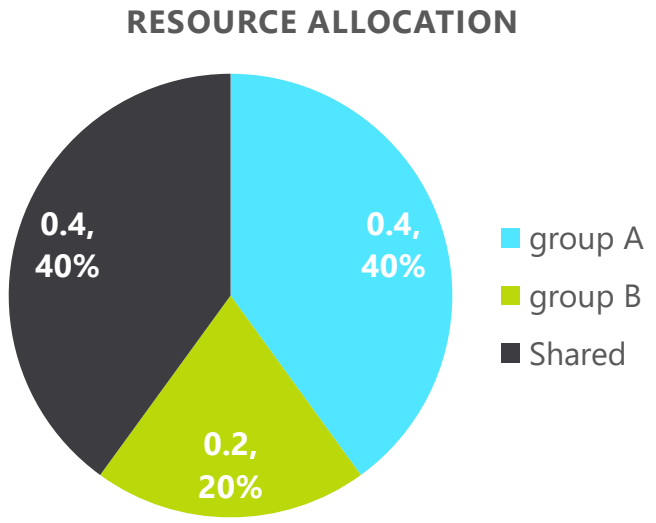
Set Query timeout value. Get DBAs out of the business of killing runaway queries

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;
```

Automatic statistics management

Overview

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

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

Statistics are considered out of date when:

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

```
-- Turn on/off auto-create statistics settings
ALTER DATABASE {database_name}
SET AUTO_CREATE_STATISTICS { ON | OFF }

-- Turn on/off auto-update statistics settings
ALTER DATABASE {database_name}
SET AUTO_UPDATE_STATISTICS { ON | OFF }

-- Configure synchronous/asynchronous update
ALTER DATABASE {database_name}
SET AUTO_UPDATE_STATISTICS_ASYNC { ON | OFF }

-- Check statistics settings for a database
SELECT      is_auto_create_stats_on,
            is_auto_update_stats_on,
            is_auto_update_stats_async_on
FROM        sys.databases
```

References

https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/resource-classes-for-workload-management

Microsoft

Azure Synapse Analytics

Security

Security in Azure Synapse Analytics

Securing with firewalls

Overview

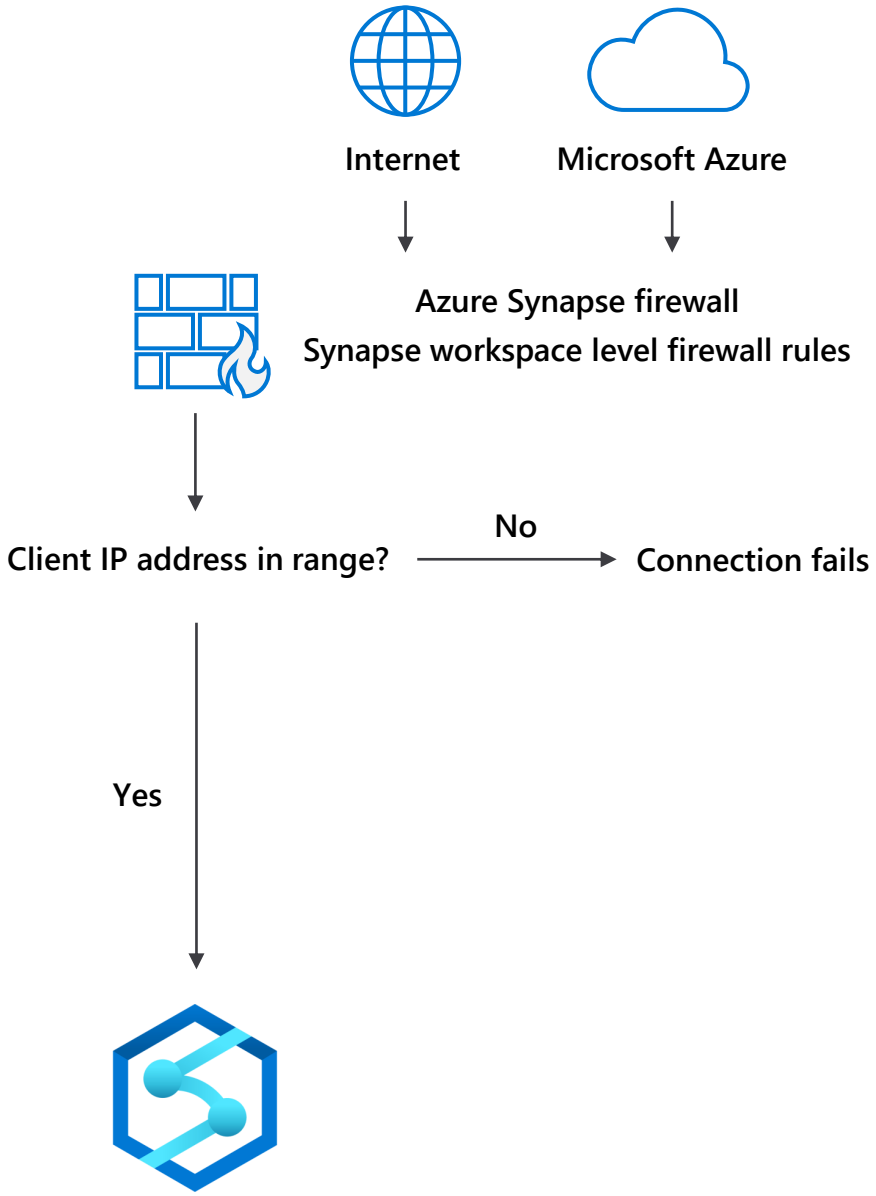
Access to your Azure Synapse Analytics is blocked by the firewall.

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

Rules

Allow specific or range of allowed IP addresses.

Allow Azure applications to connect.



IP Firewall Rules

Overview

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

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

Key Points

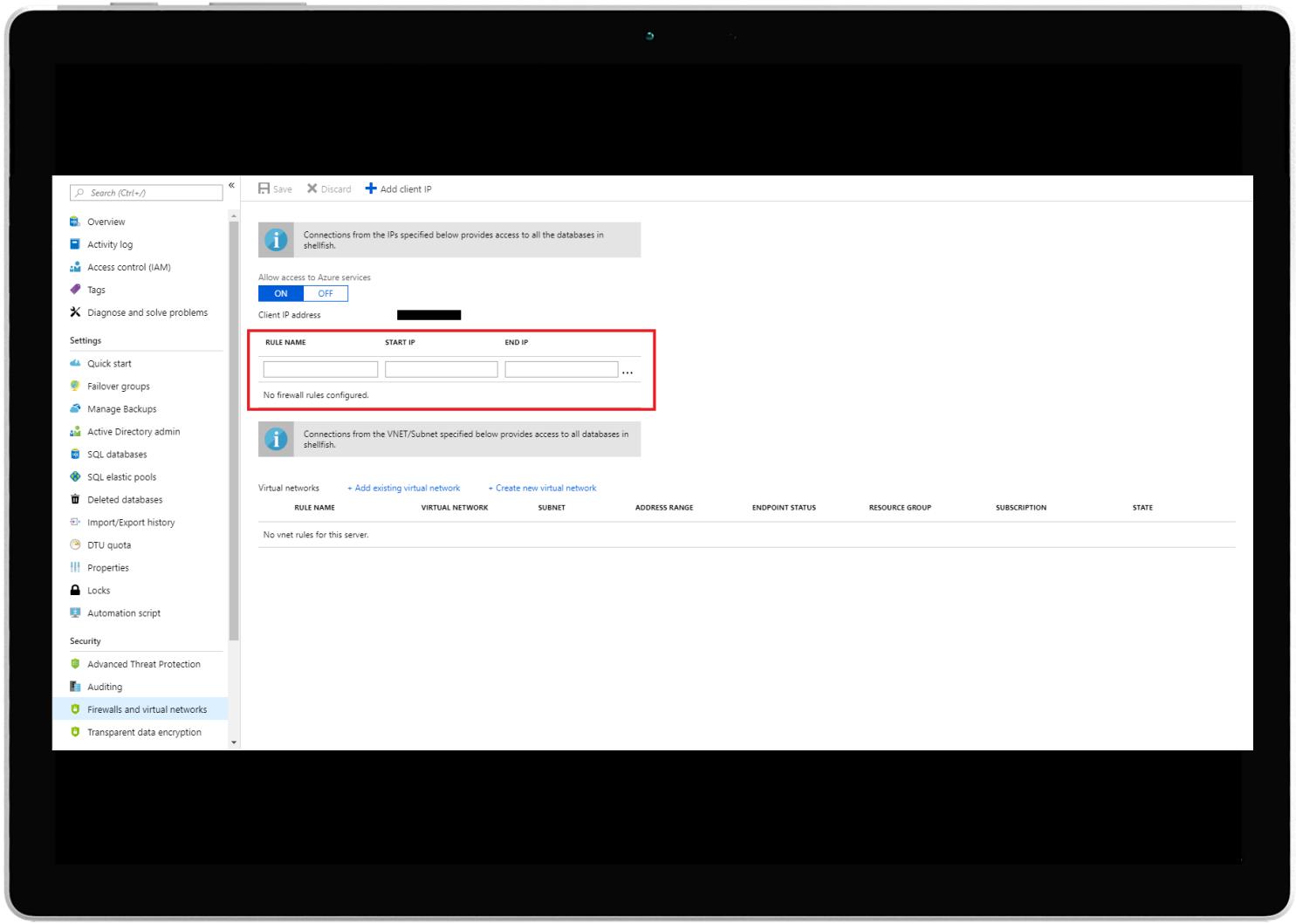
- Customers can also use SQL Server Management Studio (SSMS) to connect to the SQL resources (dedicated SQL pool and serverless SQL pool) in their workspace.
- Customers must ensure that the firewall on the network and local computer allow outgoing communication on TCP ports 80, 443 and 1443 for Synapse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

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/{subscriptionI
d}/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/{subscriptionI
d}/resourceGroups/{resourceGroupName}/providers/Microsoft
.Sql/servers/{serverName}/firewallRules/{firewallRuleName
}?api-version=2014-04-01

GET
https://management.azure.com/subscriptions/{subscriptionI
d}/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'
```

Managed VNet

Overview

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

Benefits

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

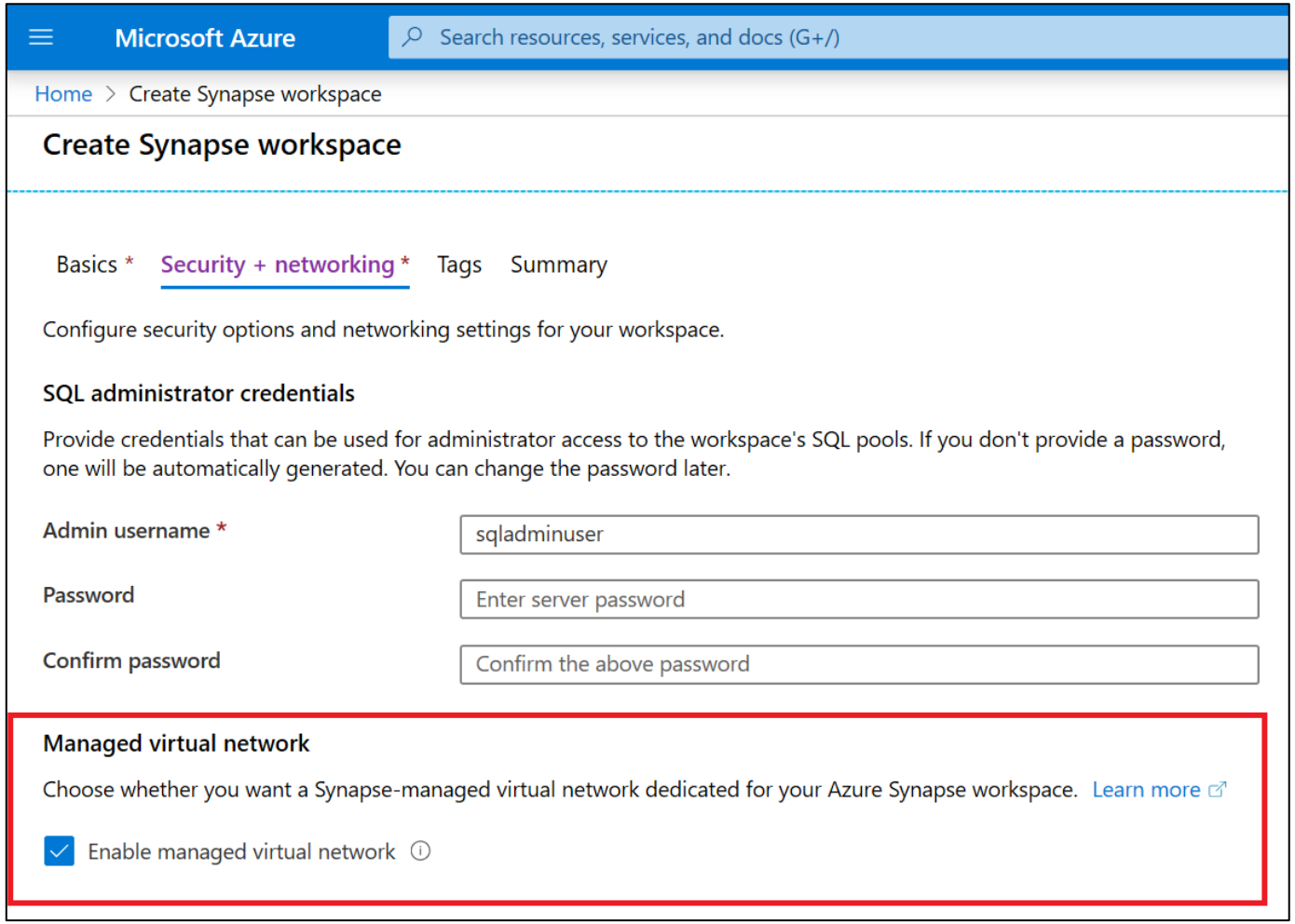
Managed VNet

Overview

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

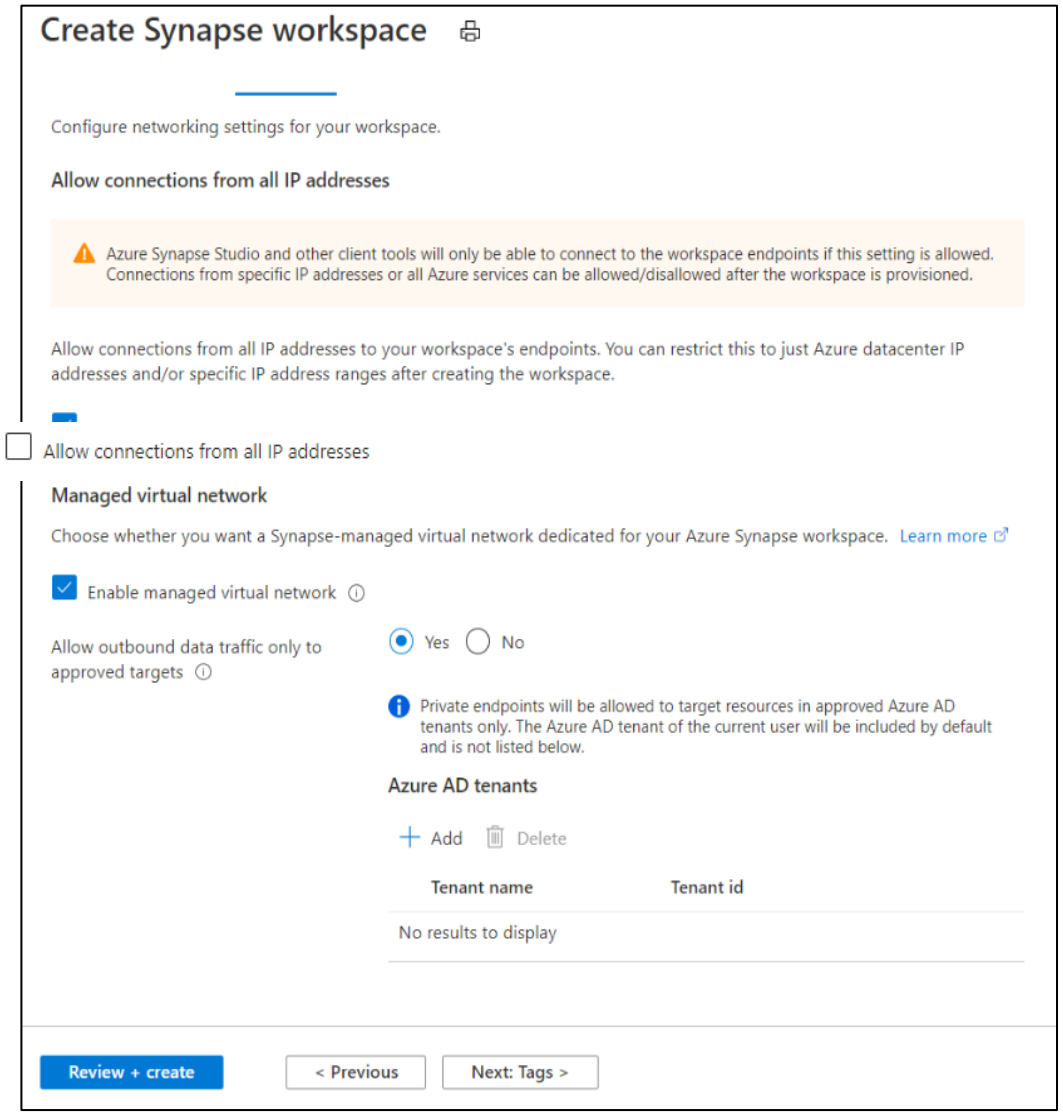
User cannot reconfigure a workspace that does not have a Managed workspace VNet associated with it and associate a VNet to it.

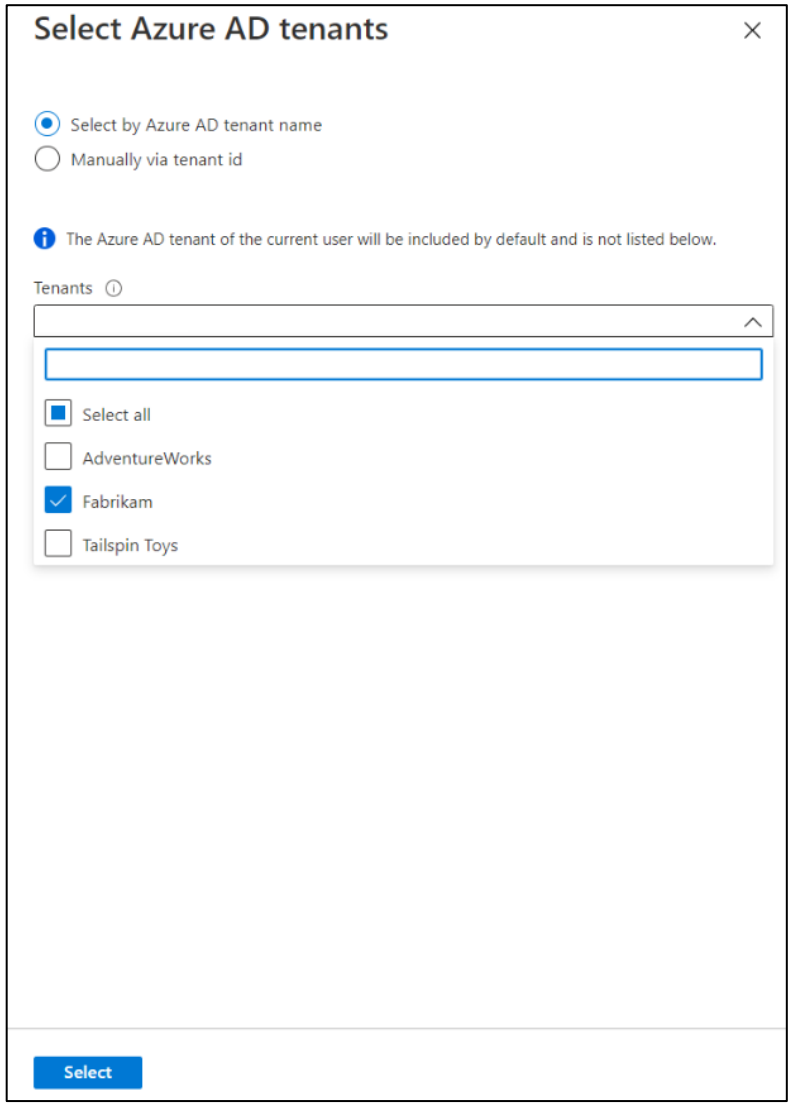
Private links are supported only in Synapse workspaces that have a managed VNet associated to it.



Data Exfiltration

Allow outbound data traffic over Synapse managed private endpoints to only approved tenants





Private Endpoints

Overview

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

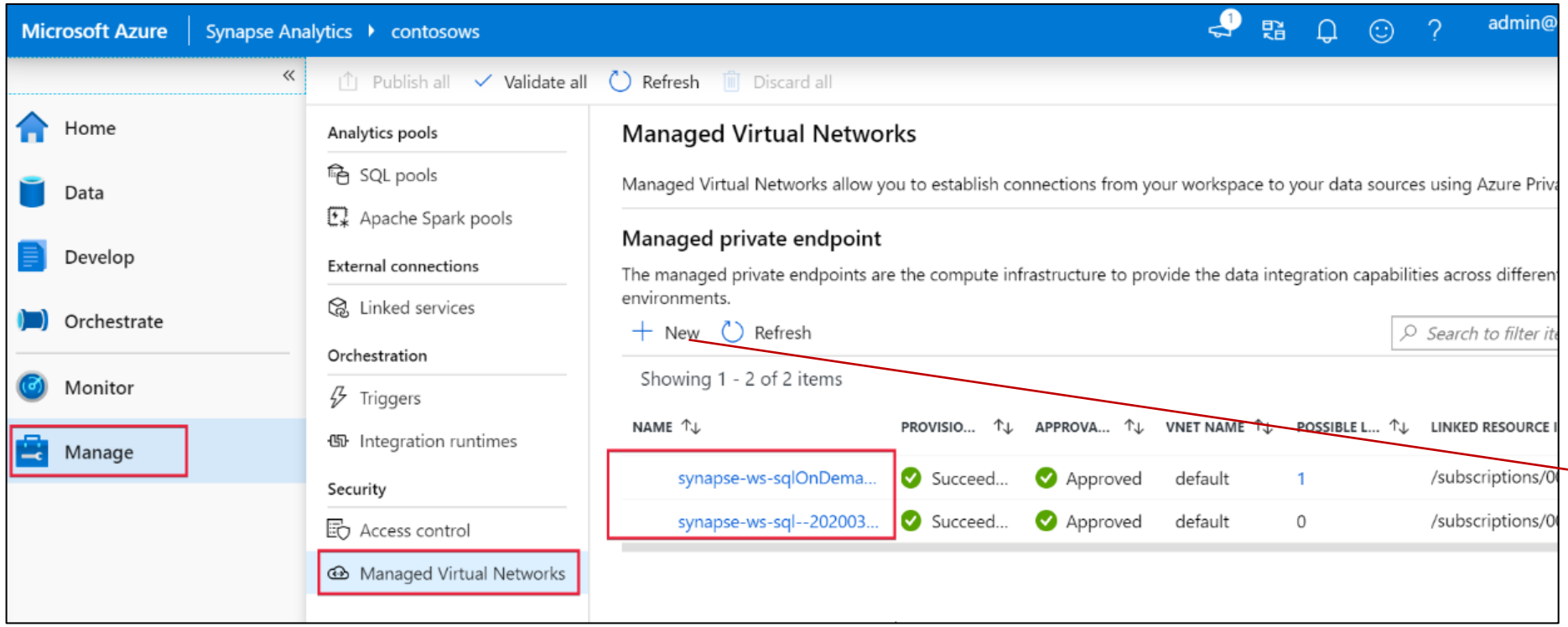
Benefits

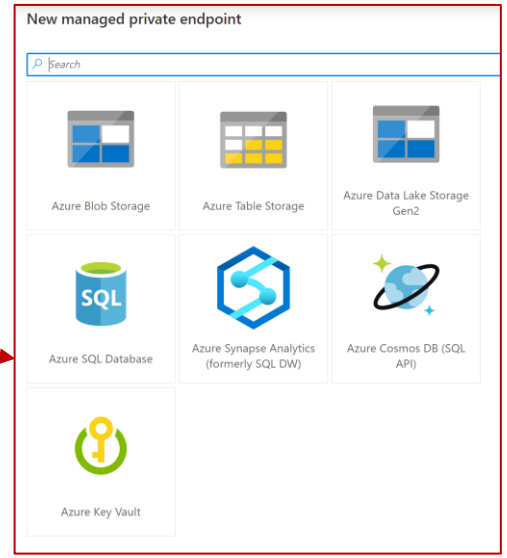
- Private link enables customers to access Azure services and Azure hosted customer/partner services from their Azure VNet securely.
- With use of private link, traffic between customer's VNet and workspace traverses entirely over the Microsoft backbone network.
- Private link protects against data exfiltration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

Private Endpoints for Synapse SQL (dedicated & serverless)

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

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





Managed VNets in the Synapse Workspace

The two Managed private endpoints to SQL pool and serverless SQL automatically created in Synapse when the Managed workspace VNet is deployed.

Synapse Workspace

Managed VNet

Synapse Studio

Managed private endpoint

Managed private endpoint to serverless SQL

Managed private endpoint to SQL Pool

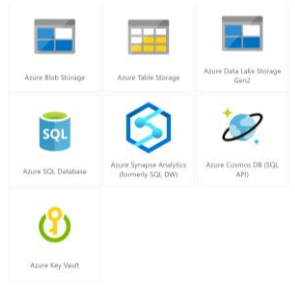
Spark Clusters

Workspace Serverless SQL

Workspace SQL Pool

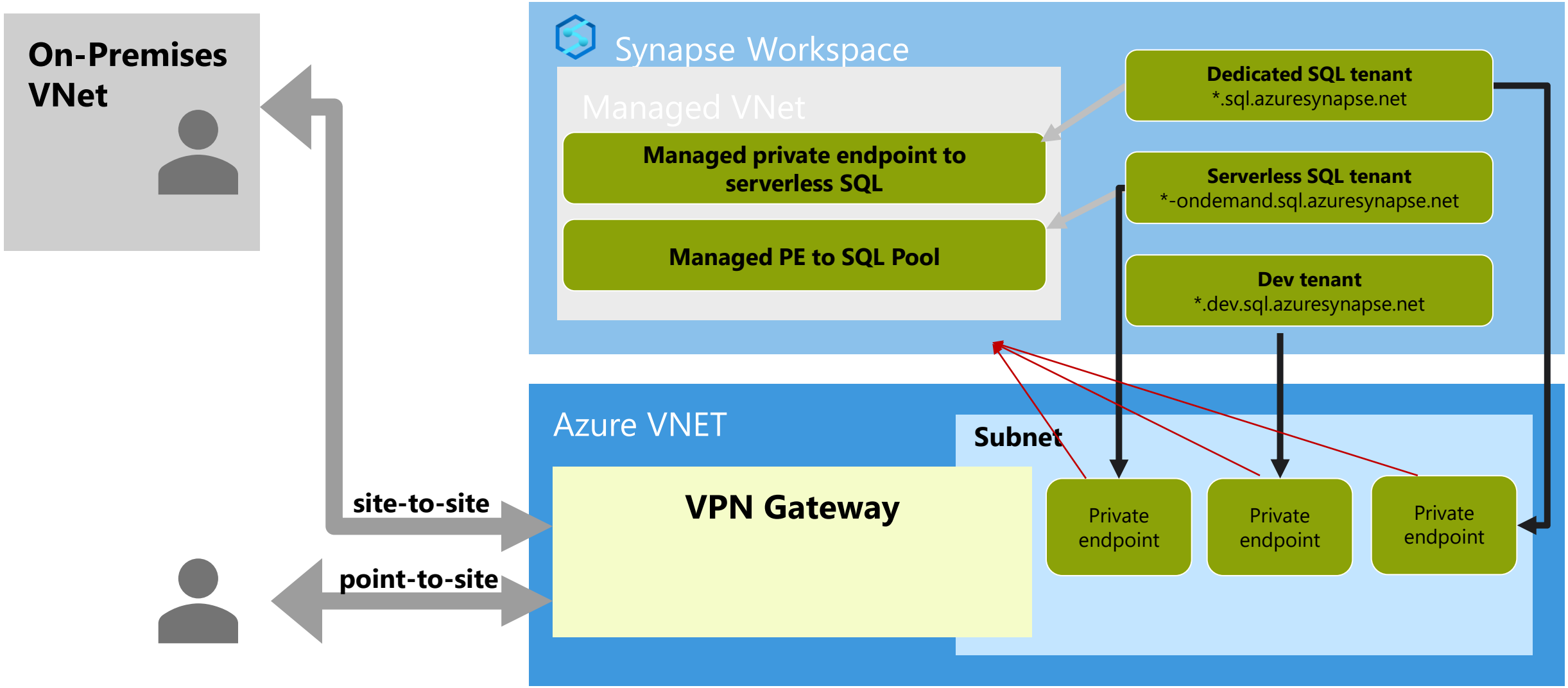
Internet

Other data sources that support Private Link (e.g. Cosmos DB, Azure Storage, ADLS etc.)

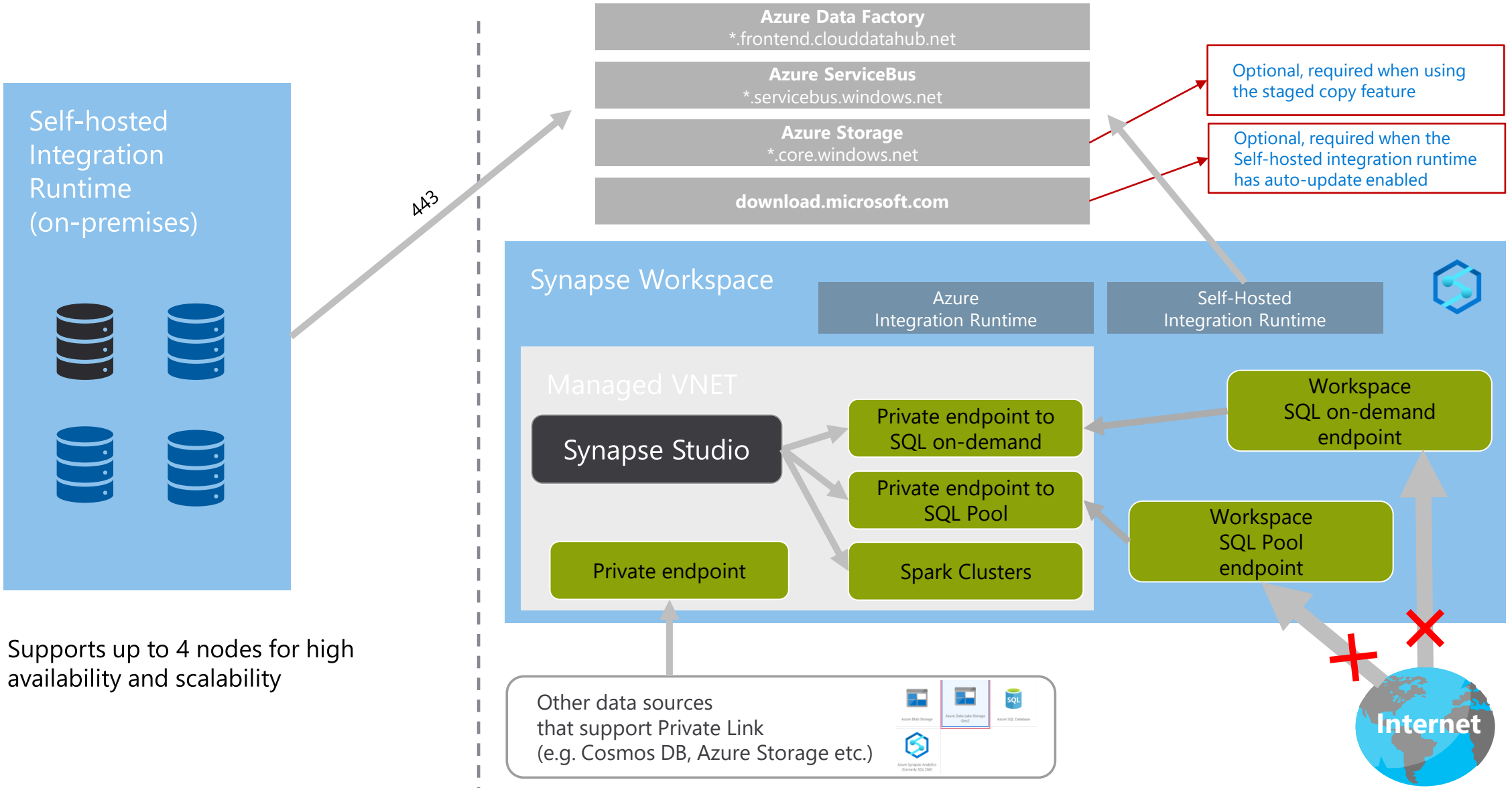


The private endpoint establishes a private link between your Azure Synapse Analytics workspace and your data sources where traffic between your VNet and workspace traverses entirely over the Microsoft backbone network.

Connecting to Synapse from On-Premises



Self-hosted Integration Runtime in Synapse Workspace

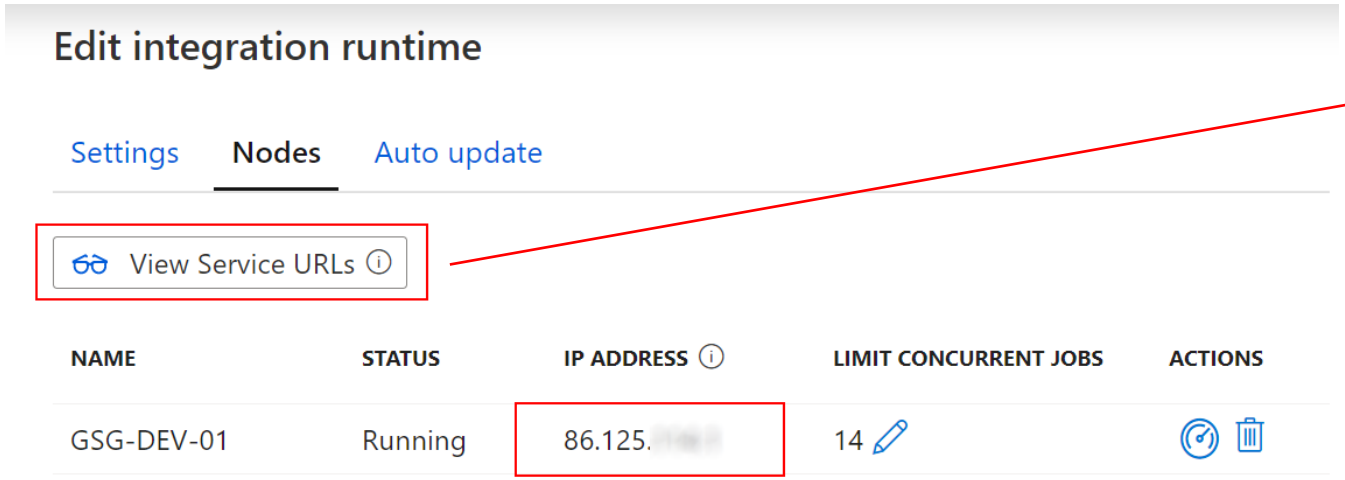


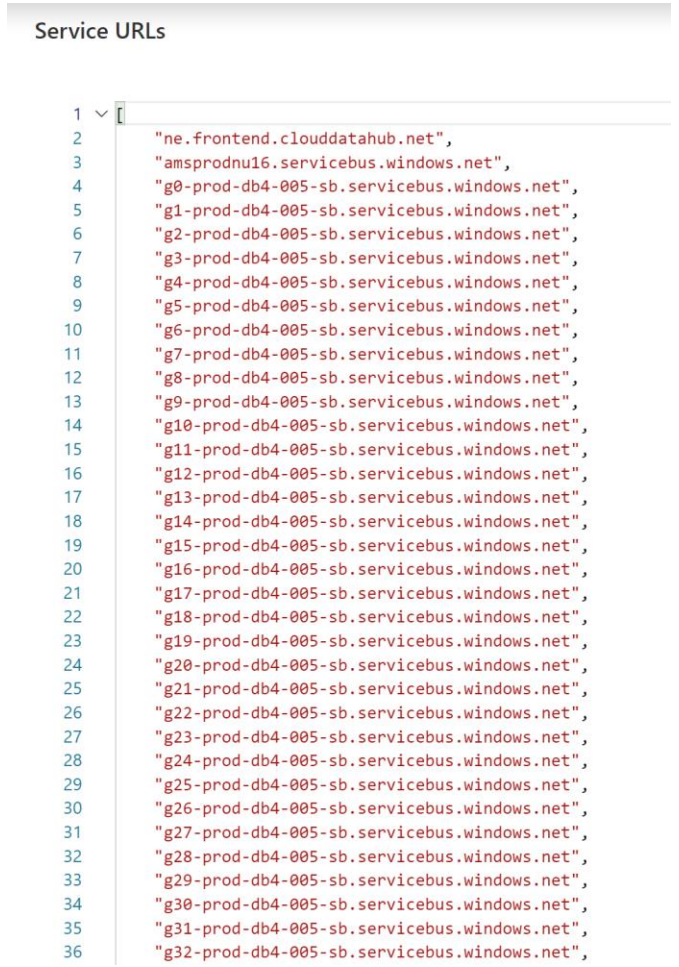
Supports up to 4 nodes for high availability and scalability

Self-hosted Integration Runtime in Synapse Workspace

Allowing only specific public Azure endpoints

Allowing only IR-specific IP addresses





Azure Active Directory authentication

Overview

Manage user identities in one location.

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

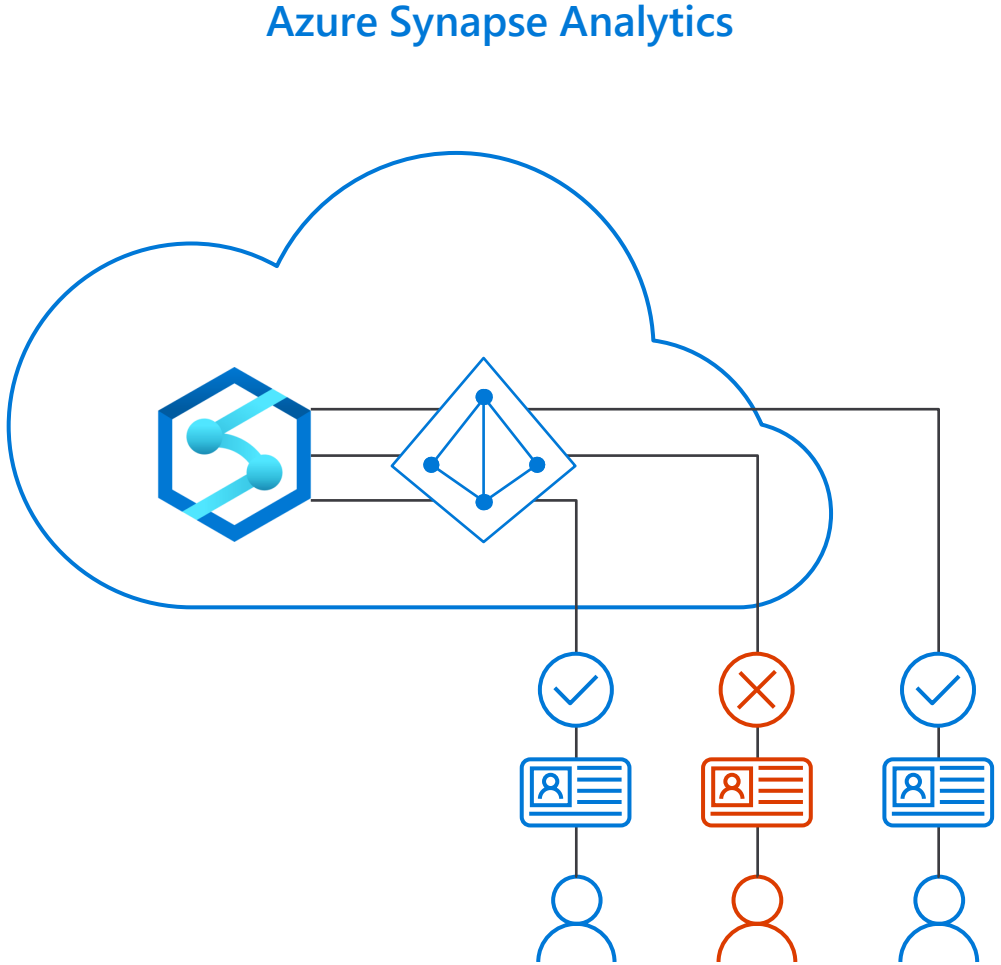
Benefits

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

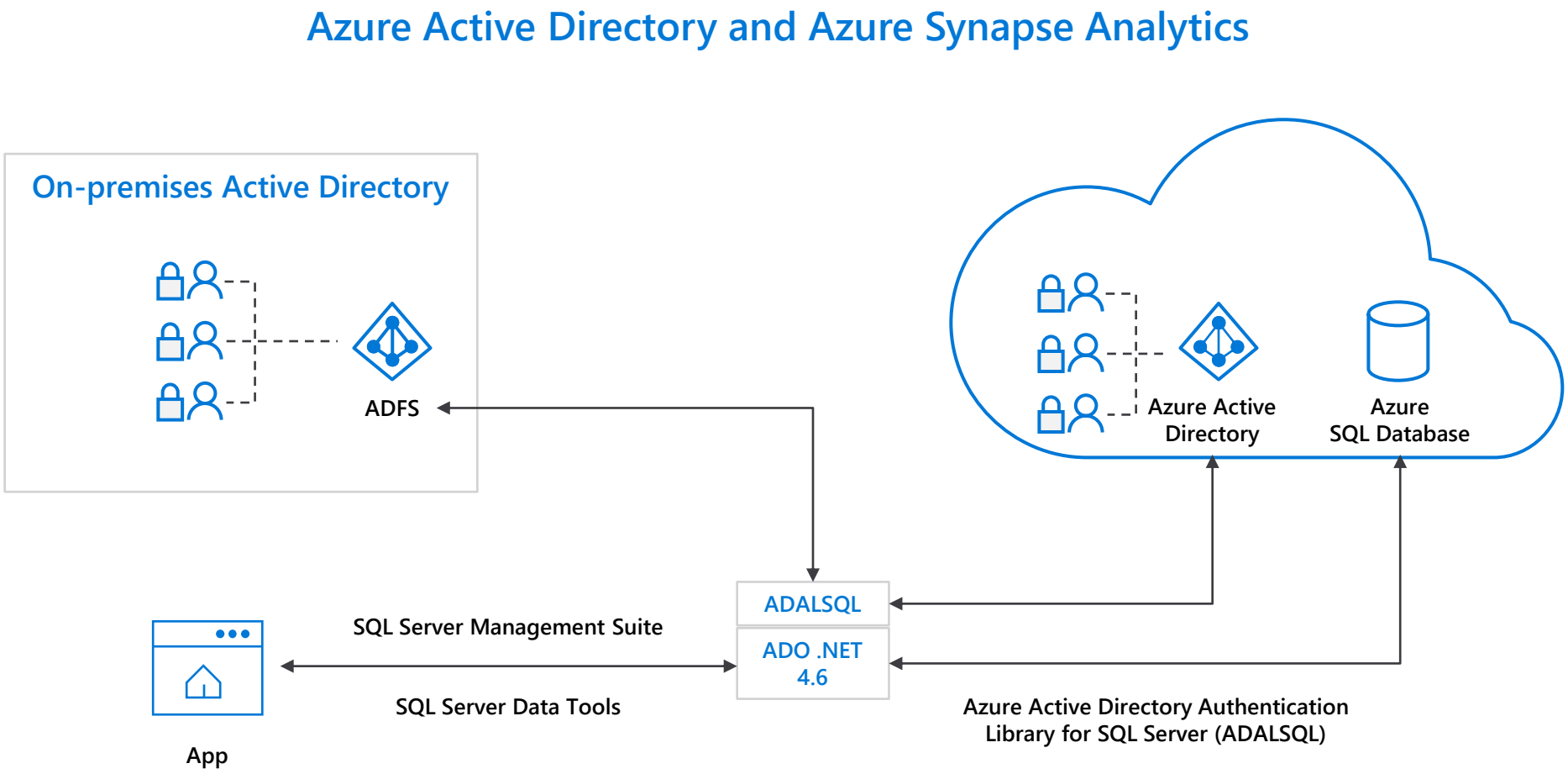
Allows password rotation in a single place

Alternative to SQL Server authentication

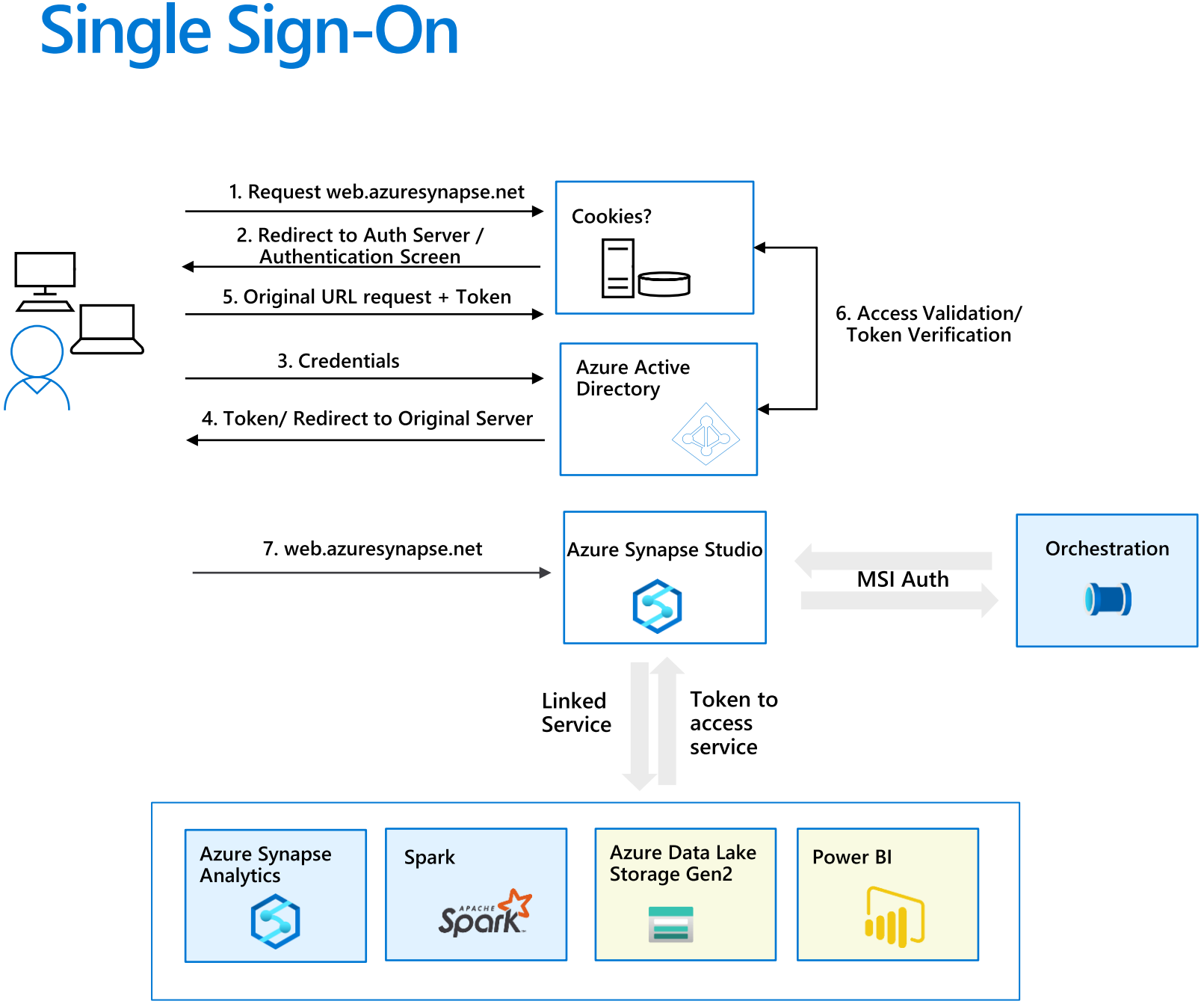
Eliminates the need to store passwords



Azure Active Directory trust architecture



Single Sign-On



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

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

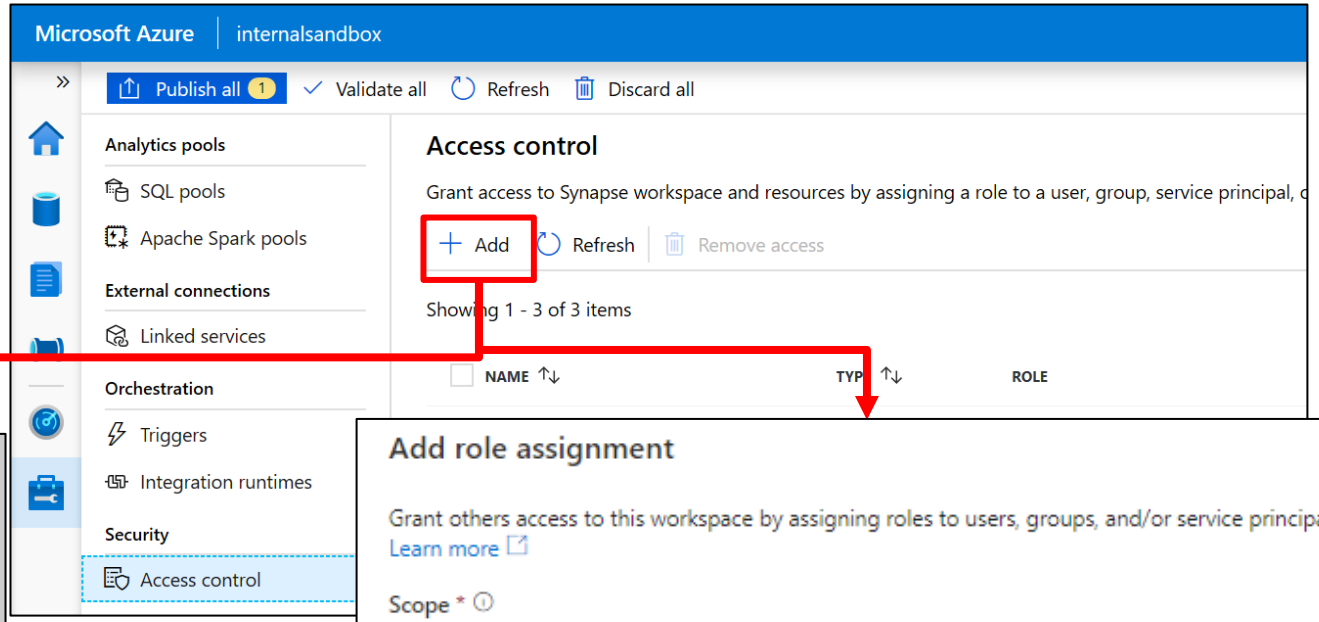
1. ADLS Gen2
2. Azure Synapse Analytics
3. Power BI
4. Spark – Spark Livy API
5. management.azure.com – resource provisioning
6. Develop artifacts – dev.workspace.net
7. Graph endpoints

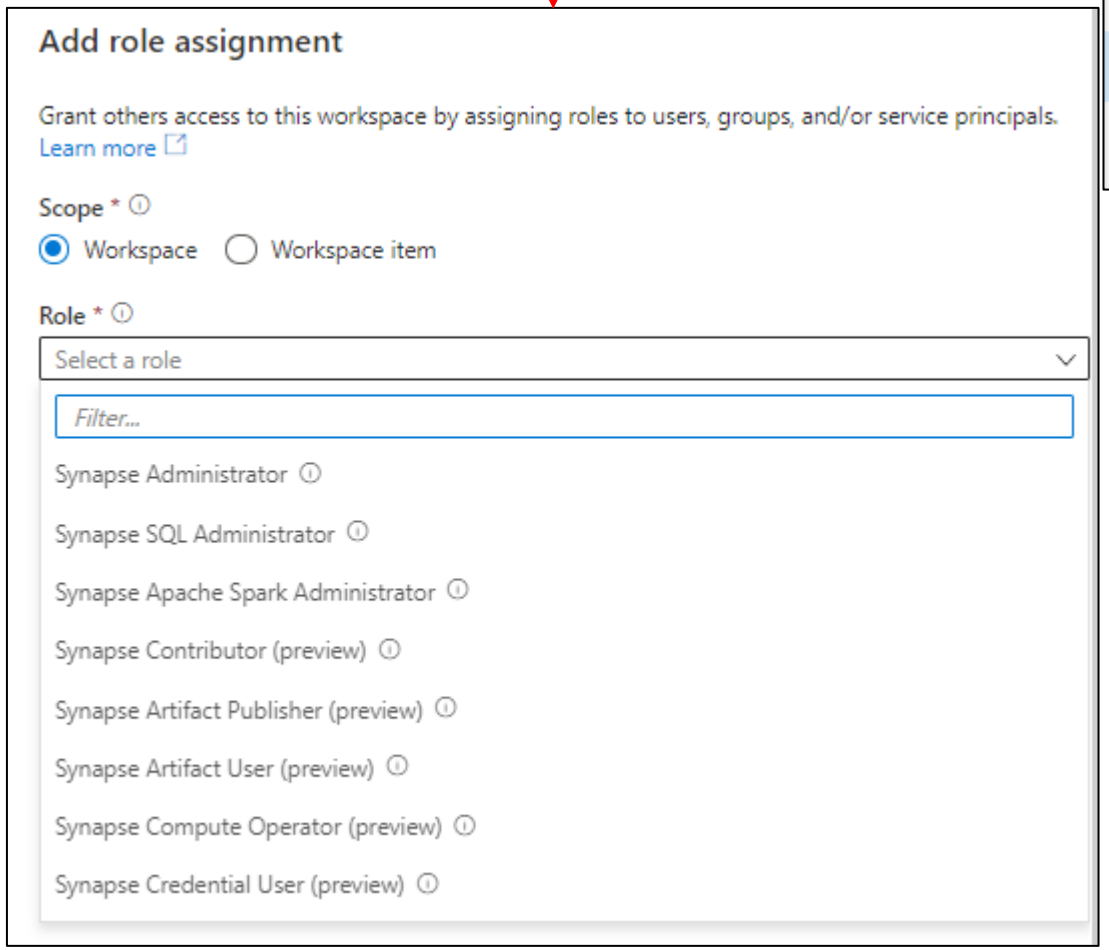
MSI authentication - Orchestration uses workspace MSI auth for automation

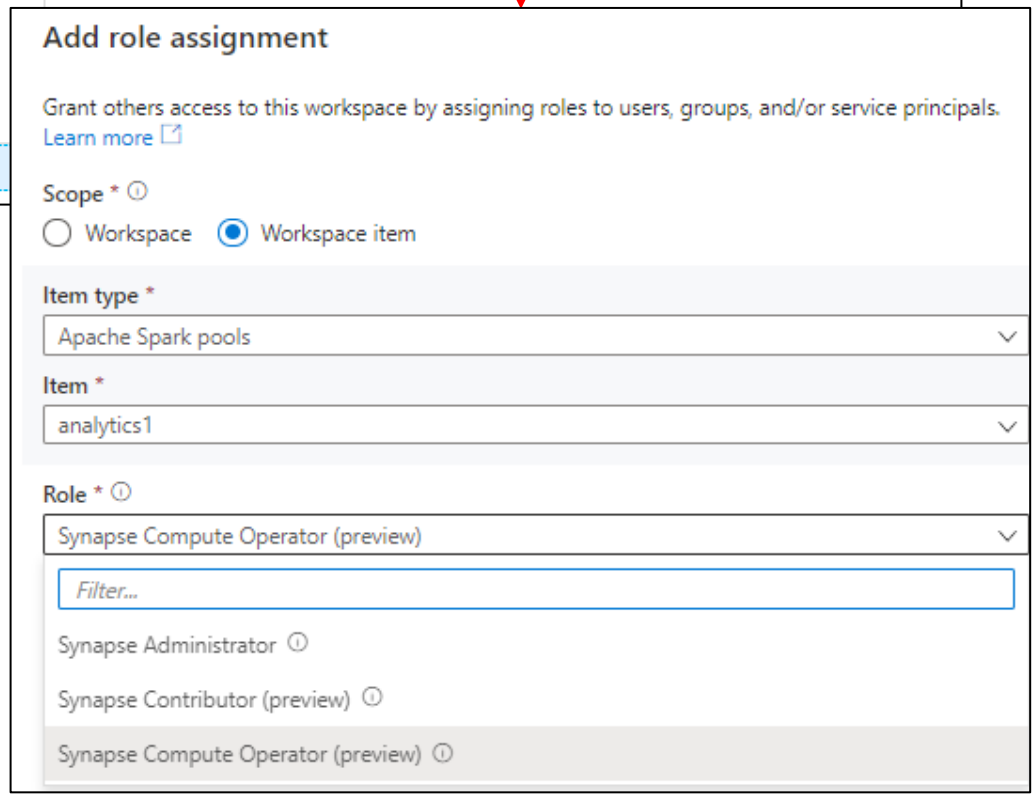
Access Control

Overview

It provides access control management to workspace resources and artifacts







Synapse RBAC Roles and permitted actions

Action	Synapse Administrator	Synapse Contributor	Synapse Artifact Author	Synapse Artifact Reader	Synapse Compute Manager	Synapse Credential User	Synapse Managed Private Endpoint Administrator	Synapse Reader
workspaces/read	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
workspaces/roleAssignments/write, delete	Yes							
workspaces/managedPrivateEndpoint/write, delete	Yes						Yes	
workspaces/bigDataPools/useCompute/action	Yes	Yes			Yes			
workspaces/bigDataPools/viewLogs/action	Yes	Yes			Yes			
workspaces/integrationRuntimes/useCompute/action	Yes	Yes			Yes			
workspaces/integrationRuntimes/viewLogs/action	Yes	Yes			Yes			
workspaces/artifacts/read	Yes	Yes	Yes	Yes				
workspaces/notebooks/write, delete	Yes	Yes	Yes					
workspaces/sparkJobDefinitions/write, delete	Yes	Yes	Yes					
workspaces/sqlScripts/write, delete	Yes	Yes	Yes					
workspaces/dataFlows/write, delete	Yes	Yes	Yes					
workspaces/pipelines/write, delete	Yes	Yes	Yes					
workspaces/triggers/write, delete	Yes	Yes	Yes					
workspaces/datasets/write, delete	Yes	Yes	Yes					
workspaces/libraries/write, delete	Yes	Yes	Yes					
workspaces/linkedServices/write, delete	Yes	Yes	Yes					
workspaces/credentials/write, delete	Yes	Yes	Yes					
workspaces/notebooks/viewOutputs/action	Yes	Yes						
workspaces/pipelines/viewOutputs/action	Yes	Yes						
workspaces/linkedServices/useSecret/action	Yes					Yes		
workspaces/credentials/useSecret/action	Yes					Yes		

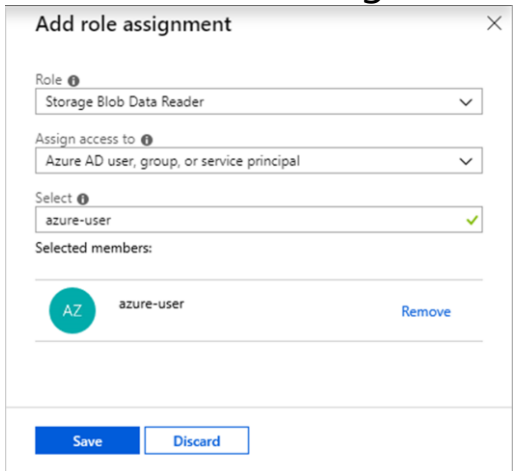
Access control – serverless SQL pool

Overview

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

Benefits

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




```
--Create Login to a a single serverless SQL pool database
CREATE LOGIN [alias@domain.com] FROM EXTERNAL PROVIDER;

-- create user under that login
use yourdb -- Use your DB name
go
CREATE USER alias FROM LOGIN [alias@domain.com];

To grant full access to a user to all serverless SQL pool databases
CREATE LOGIN [alias@domain.com] FROM EXTERNAL PROVIDER;
ALTER SERVER ROLE  sysadmin  ADD MEMBER [alias@domain.com];

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

-- enable access to specified storage using SAS token
CREATE CREDENTIAL [https://XXX.blob.core.windows.net/csv]
WITH IDENTITY  = 'SHARED ACCESS SIGNATURE',
        SECRET  = 'sv=2014-02-
14&sr=b&si=TestPolicy&sig=o%2B5%2F0C%2BLm7tWWft'

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

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


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

Enterprise-grade security

VNet

Azure Synapse Analytics

Data Protection

Access Control

Authentication

Network Security

Threat Protection

Defense-in-Depth

Industry-leading compliance

ISO 27001

SOC 1 Type 2

SOC 2 Type 2

PCI DSS Level 1

Cloud Controls Matrix

ISO 27018

Content Delivery and Security Association

Shared Assessments

FedRAMP JAB P-ATO

HIPAA / HITECH

FIPS 140-2

21 CFR Part 11

FERPA

DISA Level 2

CJIS

IRS 1075

ITAR-ready

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

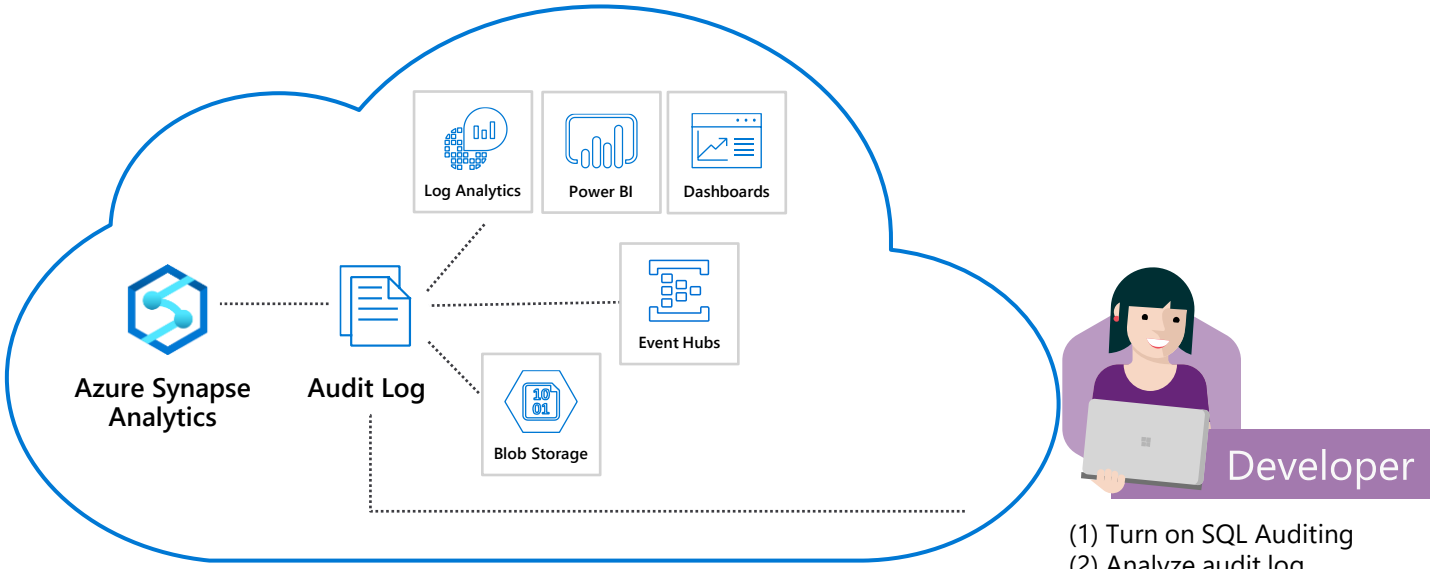
New Zealand GCIO

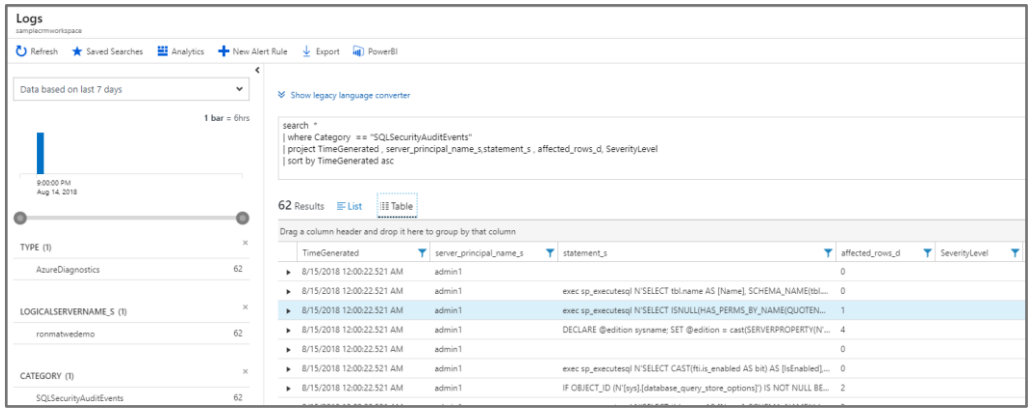
Japan Financial Services

ENISA IAF

SQL auditing in Azure Log Analytics and Event Hubs

Gain insight into database audit log

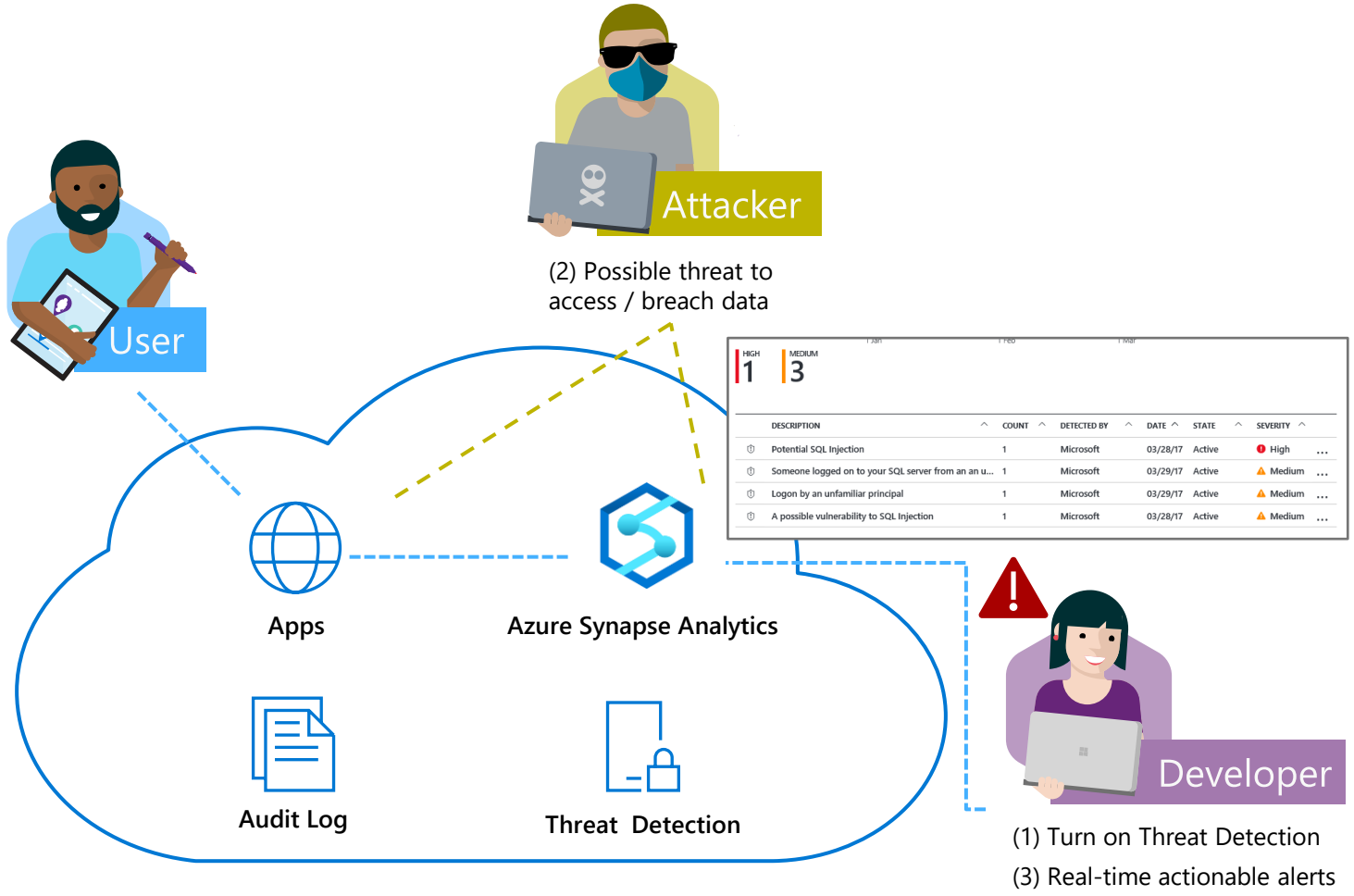




- ✓ **Configurable via audit policy**
- ✓ **SQL audit logs can reside in**
  - Azure Storage account
  - Azure Log Analytics
  - Azure Event Hubs
- ✓ **Rich set of tools for**
  - Investigating security alerts
  - Tracking access to sensitive data

SQL threat detection

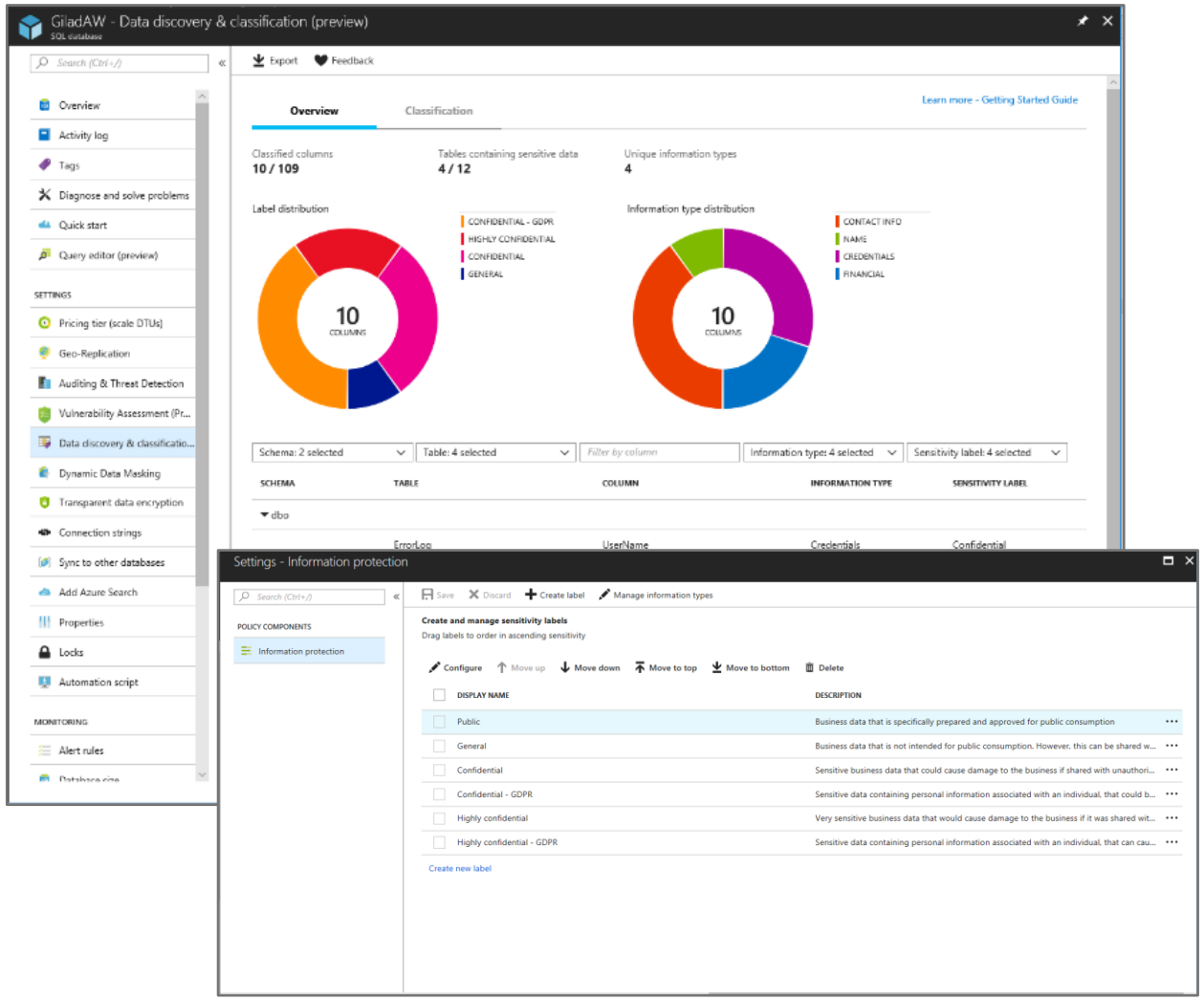
Detect and investigate anomalous database activity



- ✓ 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



- ✓ 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

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

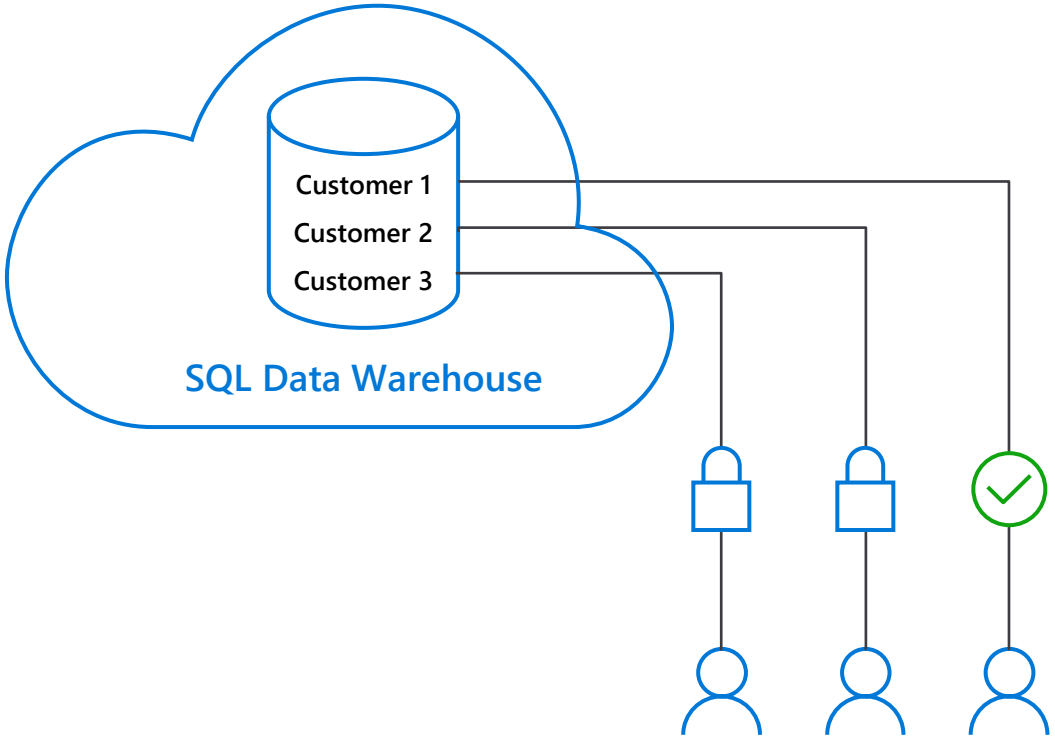
Fine grained access control of specific rows in a database table.

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
```

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.

Dynamic Data Masking

Overview

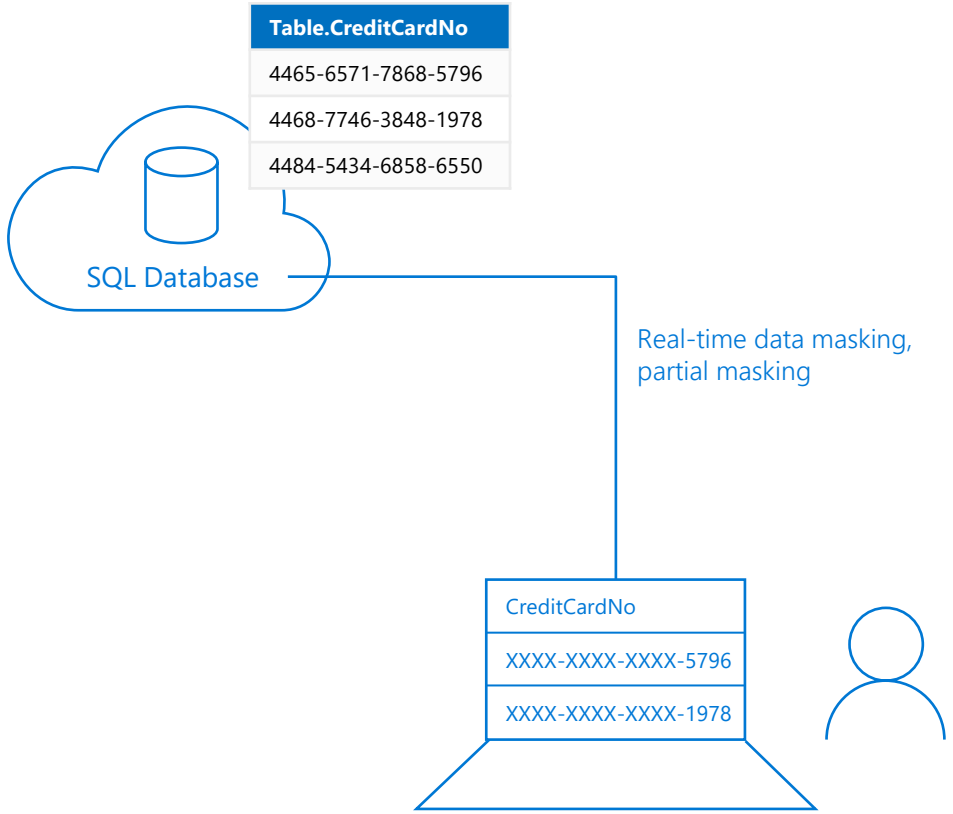
Prevent abuse of sensitive data by hiding it from users

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.)



Column Level Encryption

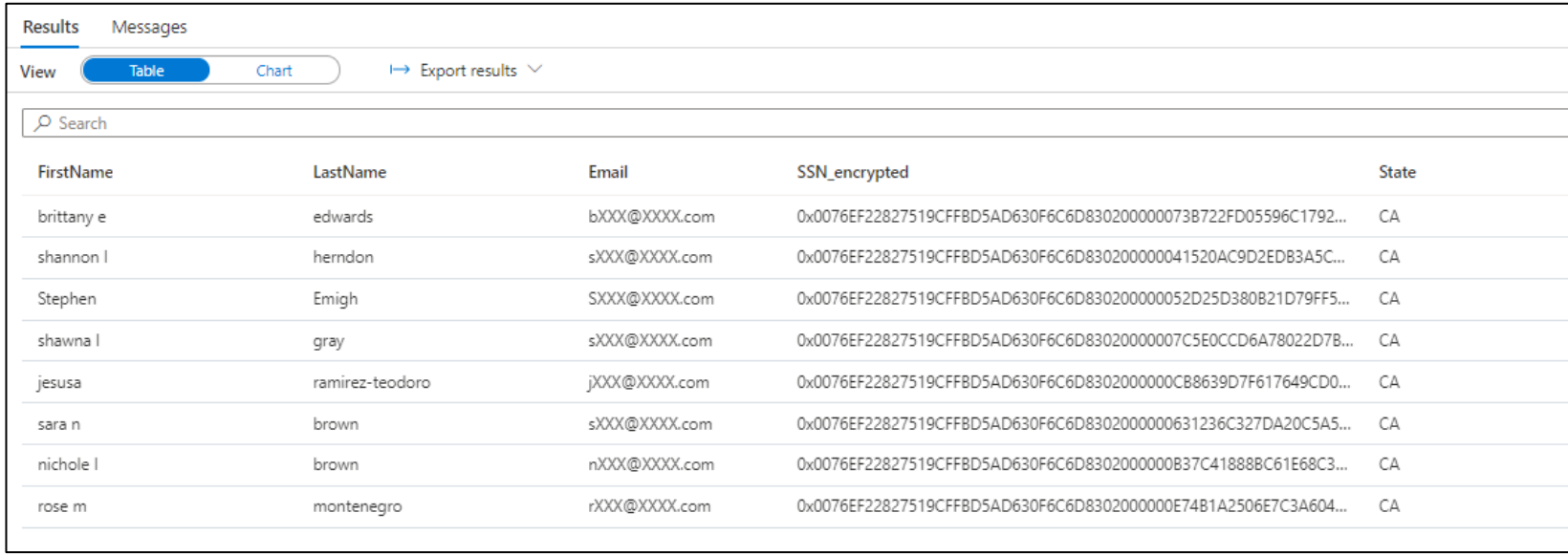
Overview

It helps to implement fine-grained protection of sensitive data within a table in dedicated SQL pool.

The data in CLE enforced columns is encrypted on disk. User need to use DECRYPTBYKEY function to decrypt it.

5 step process to set up CLE

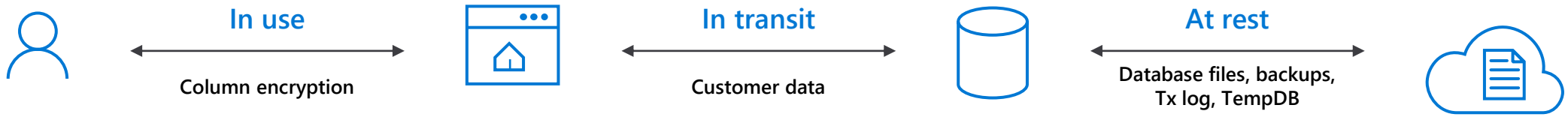
1. Create master key
2. Create certificate
3. Configure symmetric key for encryption
4. Encrypt the column data
5. Close symmetric key



FirstName	LastName	Email	SSN_encrypted	State
brittany e	edwards	bXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D830200000073B722FD05596C1792...	CA
shannon l	herndon	sXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D830200000041520AC9D2EDB3A5C...	CA
Stephen	Emigh	SXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D830200000052D25D380B21D79FF5...	CA
shawna l	gray	sXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D83020000007C5E0CCD6A78022D7B...	CA
jesusa	ramirez-teodoro	jXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D8302000000CB8639D7F617649CD0...	CA
sara n	brown	sXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D8302000000631236C327DA20C5A5...	CA
nichole l	brown	nXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D8302000000B37C41888BC61E68C3...	CA
rose m	montenegro	rXXX@XXXX.com	0x0076EF22827519CFFBD5AD630F6C6D8302000000E74B1A2506E7C3A604...	CA

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.

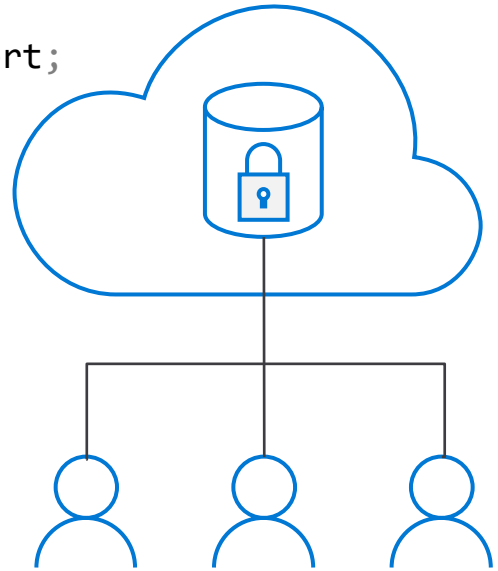
Service OR User managed keys.

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

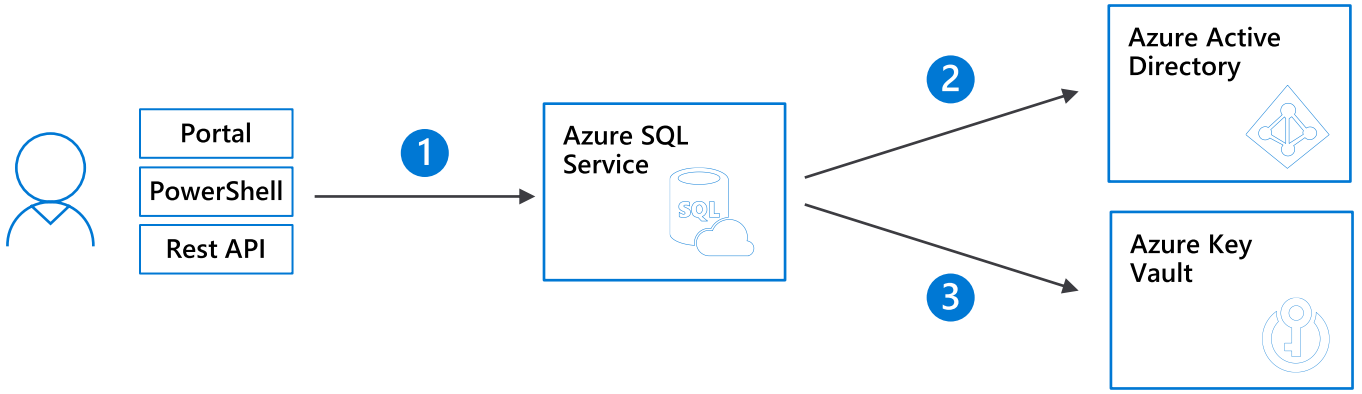
Benefits with User Managed Keys

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.



1. The Key Vault admin grants vault access to the SQL Database server using its unique Azure Active Directory (AD) identity
2. The server uses its Azure AD identity to authenticate with Azure AD for access to your Key Vault
3. The server sends get, wrap key, and unwrap key request to the asymmetric key in key Vault for database encryption key protection.

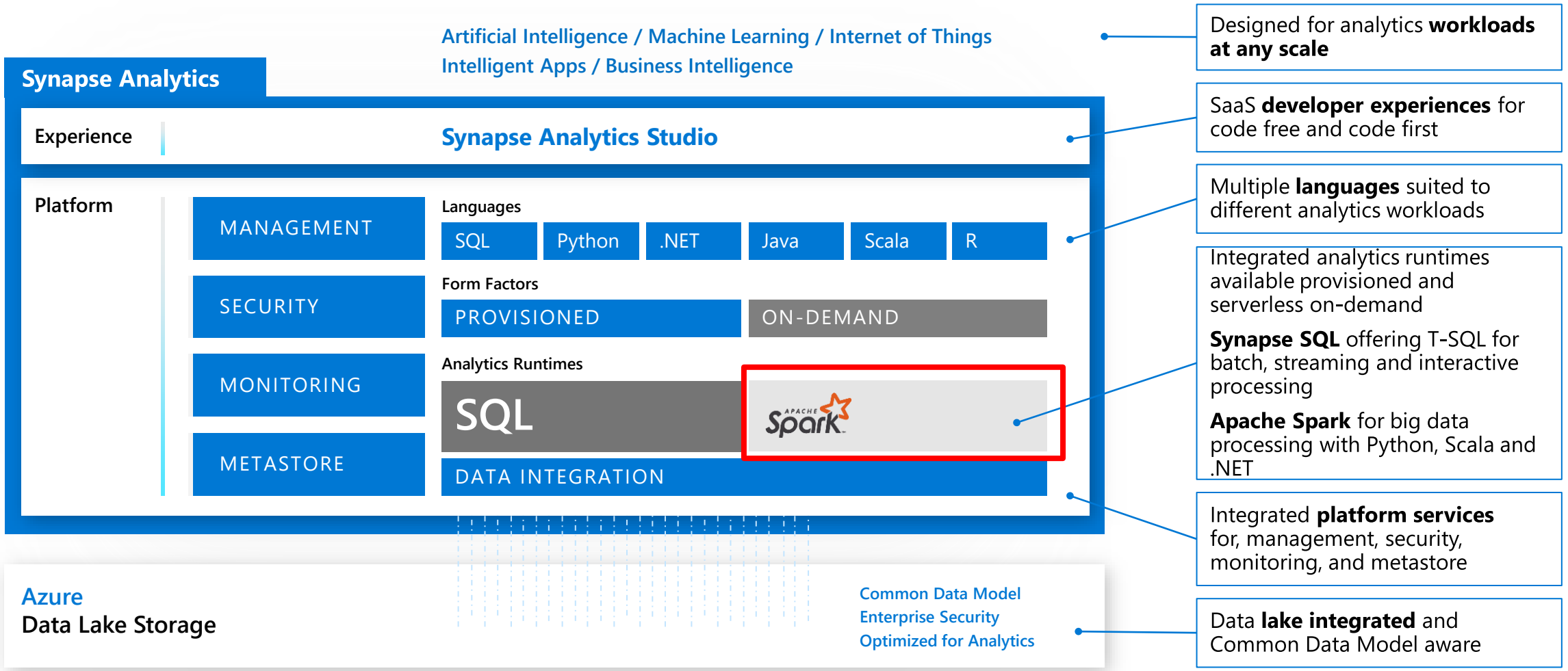
Microsoft

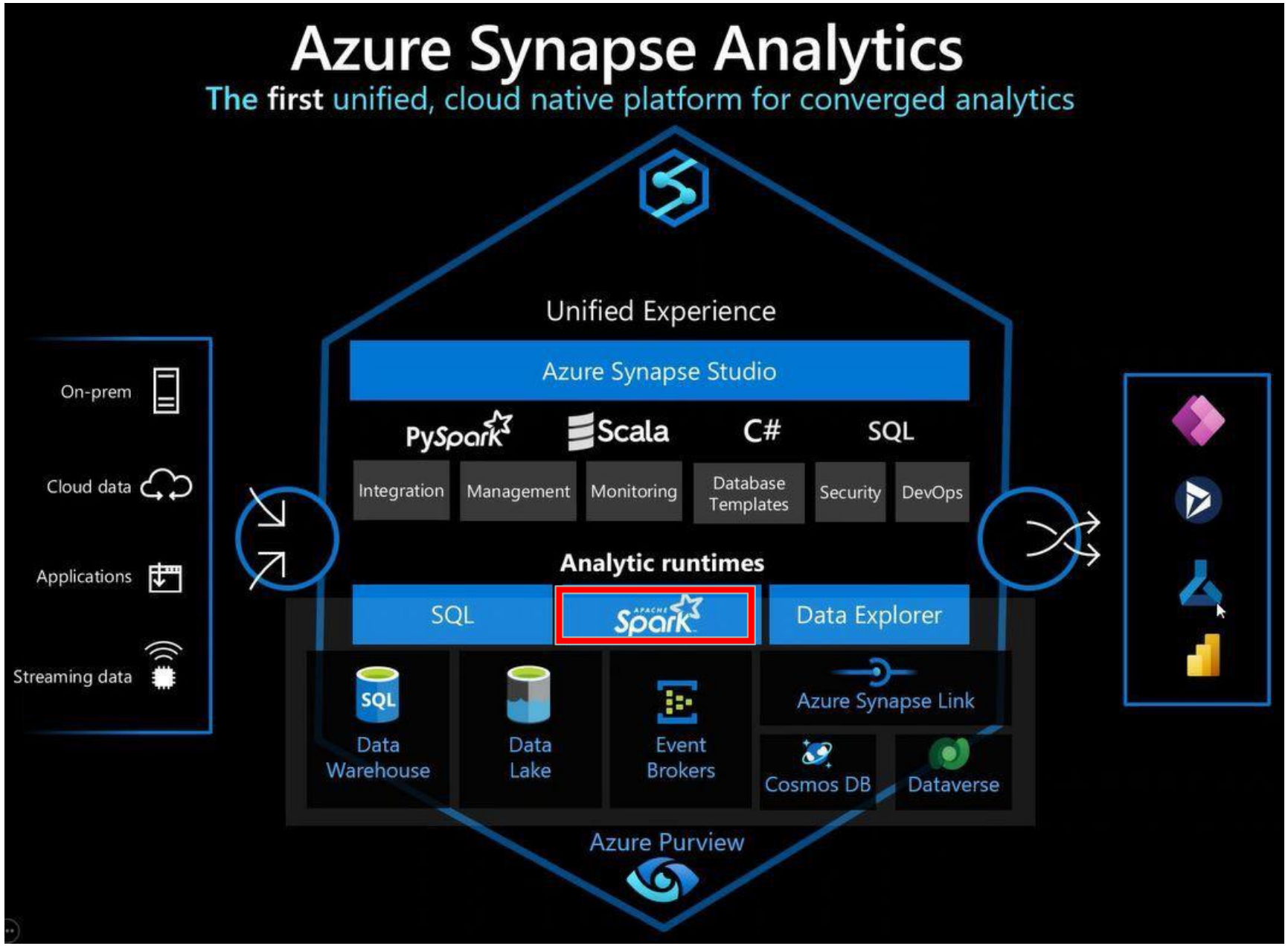
Azure Synapse Analytics

Apache Spark

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight

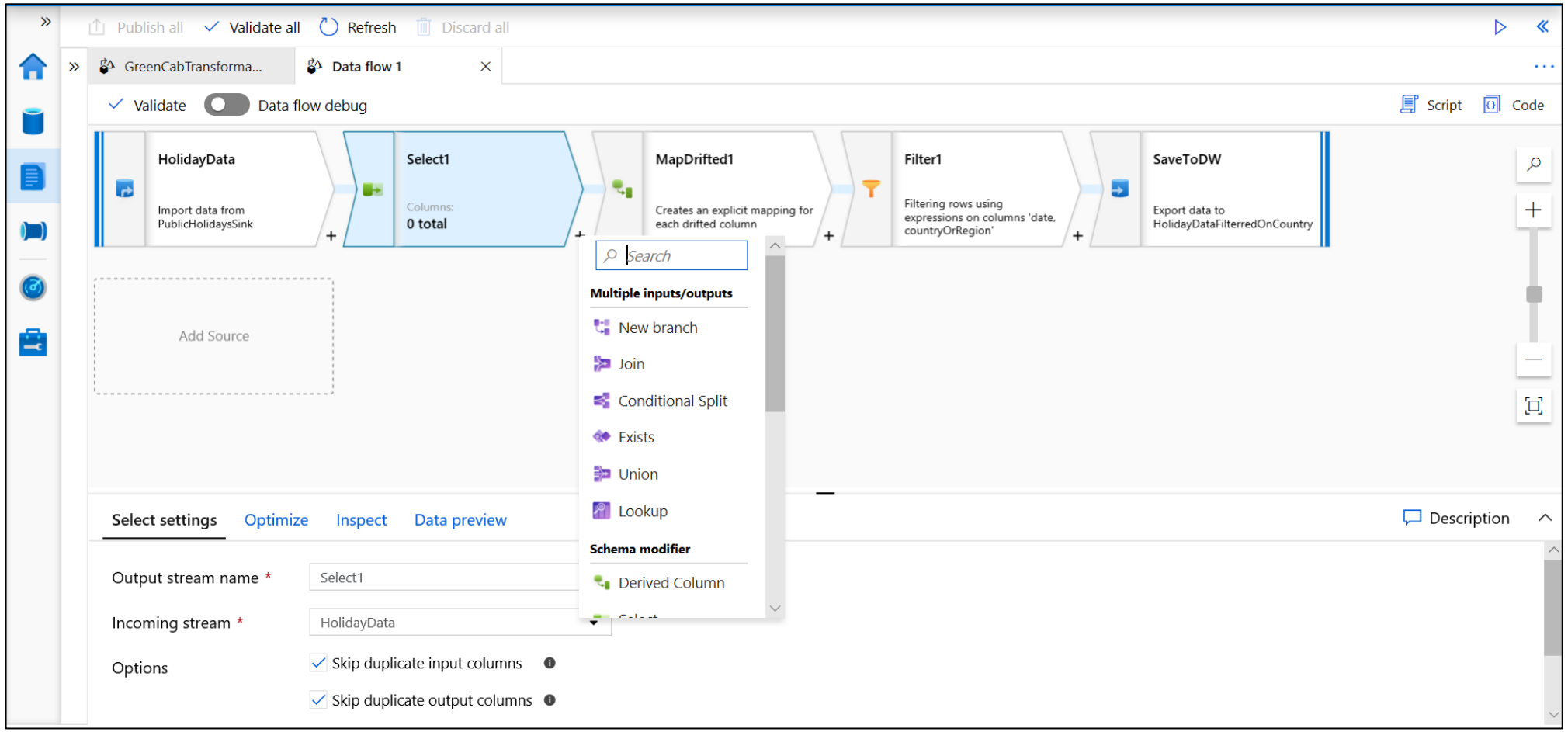



Azure Synapse Analytics
The first unified, cloud native platform for converged analytics
Unified Experience
Azure Synapse Studio
PySpark
Scala
C#
SQL
Integration
Management
Monitoring
Database Templates
Security
DevOps
Analytic runtimes
SQL
Apache Spark
Data Explorer
On-prem
Cloud data
Applications
Streaming data
Data Warehouse
Data Lake
Event Brokers
Azure Synapse Link
Cosmos DB
Dataverse
Azure Purview

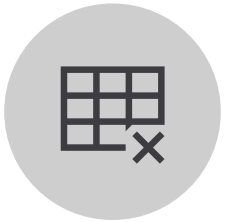
Develop Hub - Data Flows

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

Provides a code-free experience.



Dataflow Capabilities



Handle upserts, updates, deletes on sql sinks

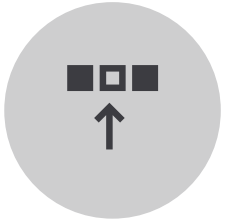
Add new partition methods

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)

Azure Synapse Apache Spark - Summary

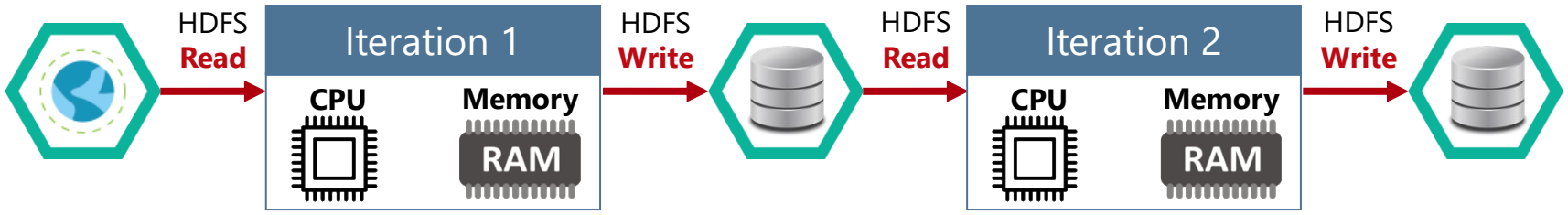
- **Apache Spark 3.1 derivation**
  - Linux Foundation Delta Lake 1.0 support
  - .Net Core 3.1 support
  - Python 3.8 + Anacondas support
- **Tightly coupled to other Azure Synapse services**
  - Integrated security and sign on
  - Integrated Metadata
  - Integrated and simplified provisioning
  - Integrated UX including nteract based notebooks
  - Fast load of Synapse SQL (provisioned) 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

Azure Synapse Apache Spark - Summary

- **Apache Spark 3.1 derivation**
  - Linux Foundation Delta Lake 1.0 support
  - .Net Core 3.1 support
  - Python 3.8 + Anacondas support
- **Tightly coupled to other Azure Synapse services**
  - Integrated security and sign on
  - Integrated Metadata
  - Integrated and simplified provisioning
  - Integrated UX including nteract based notebooks
  - Fast load of Synapse SQL (provisioned) 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

Motivation for Apache Spark

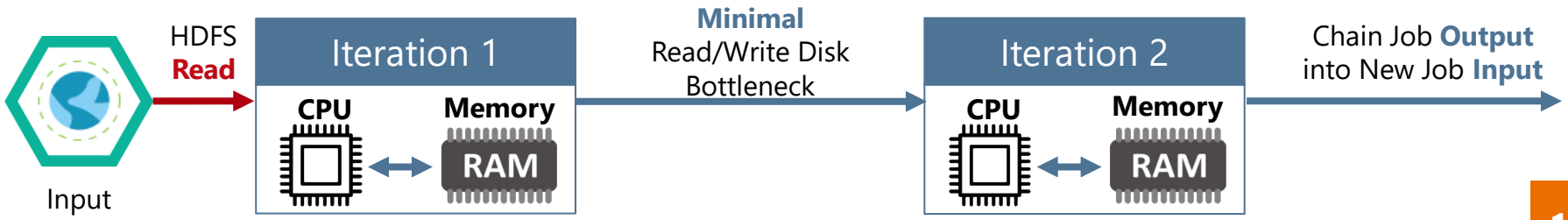
Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O



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**

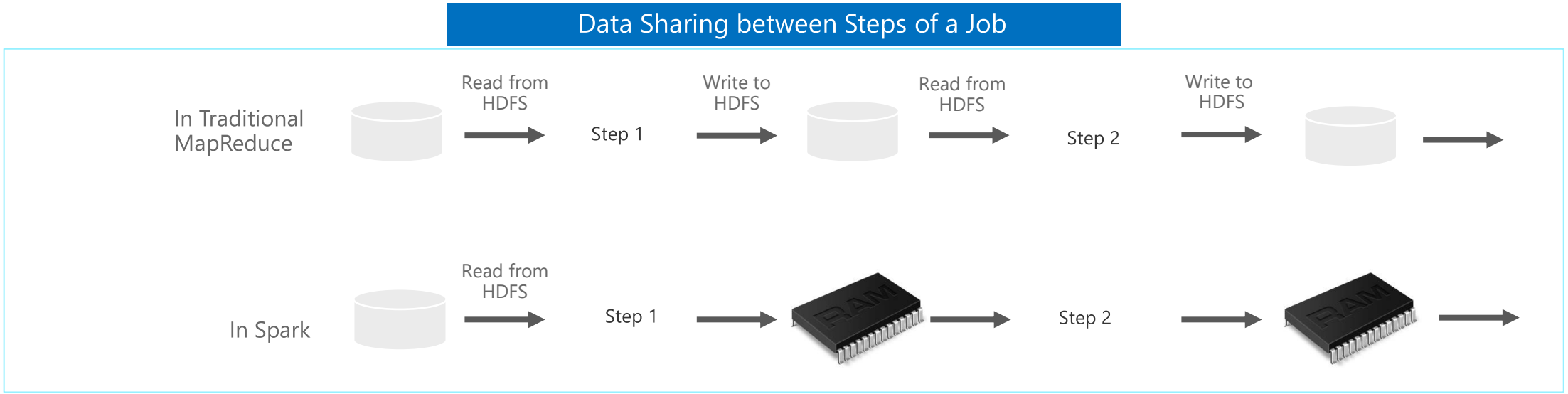
Solution: Keep data **in-memory** with a new distributed execution engine



10–100x faster than network & disk

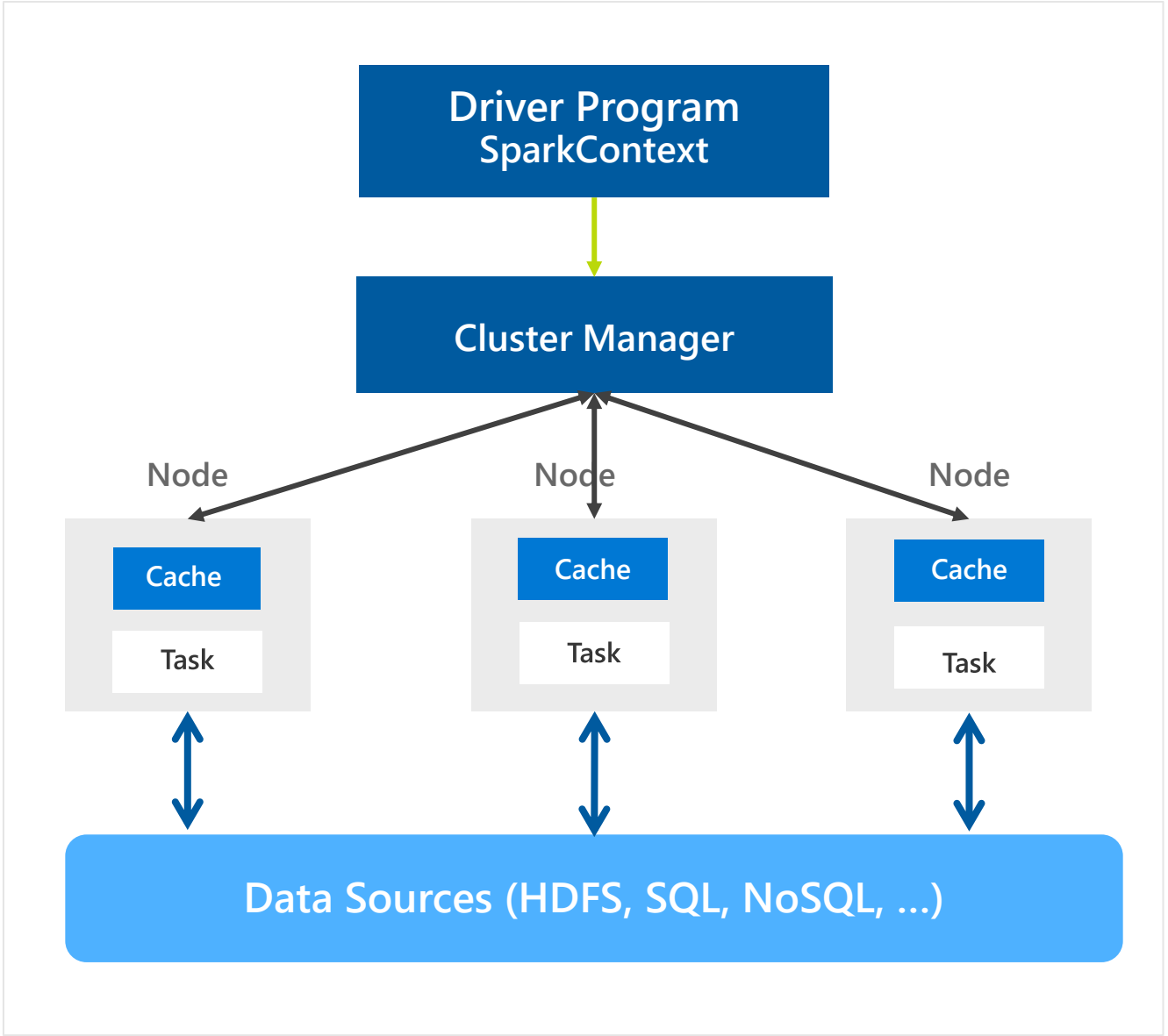
What makes Spark fast

- **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

Mllib/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

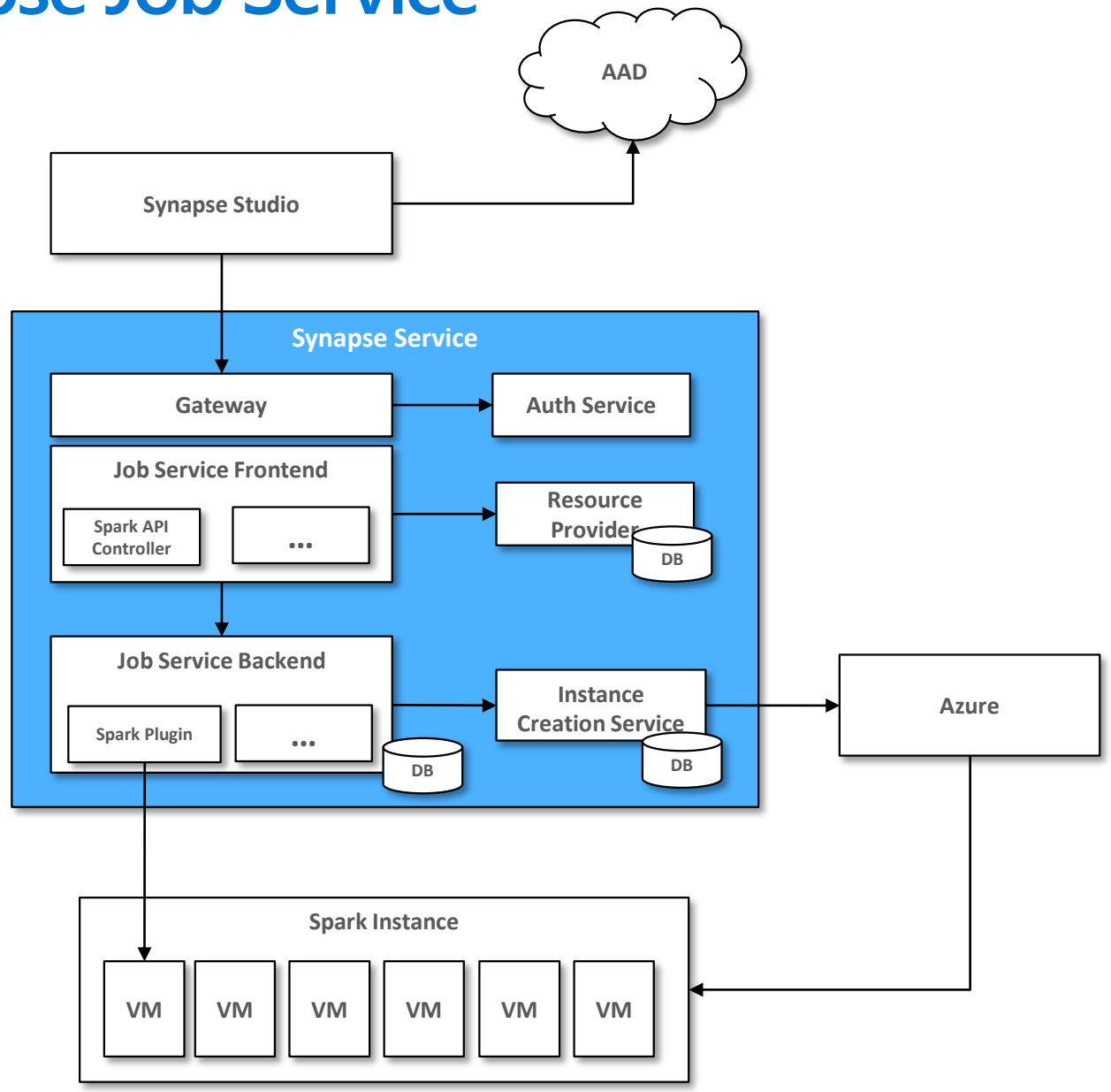
- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.

Microsoft

Azure Synapse Apache Spark

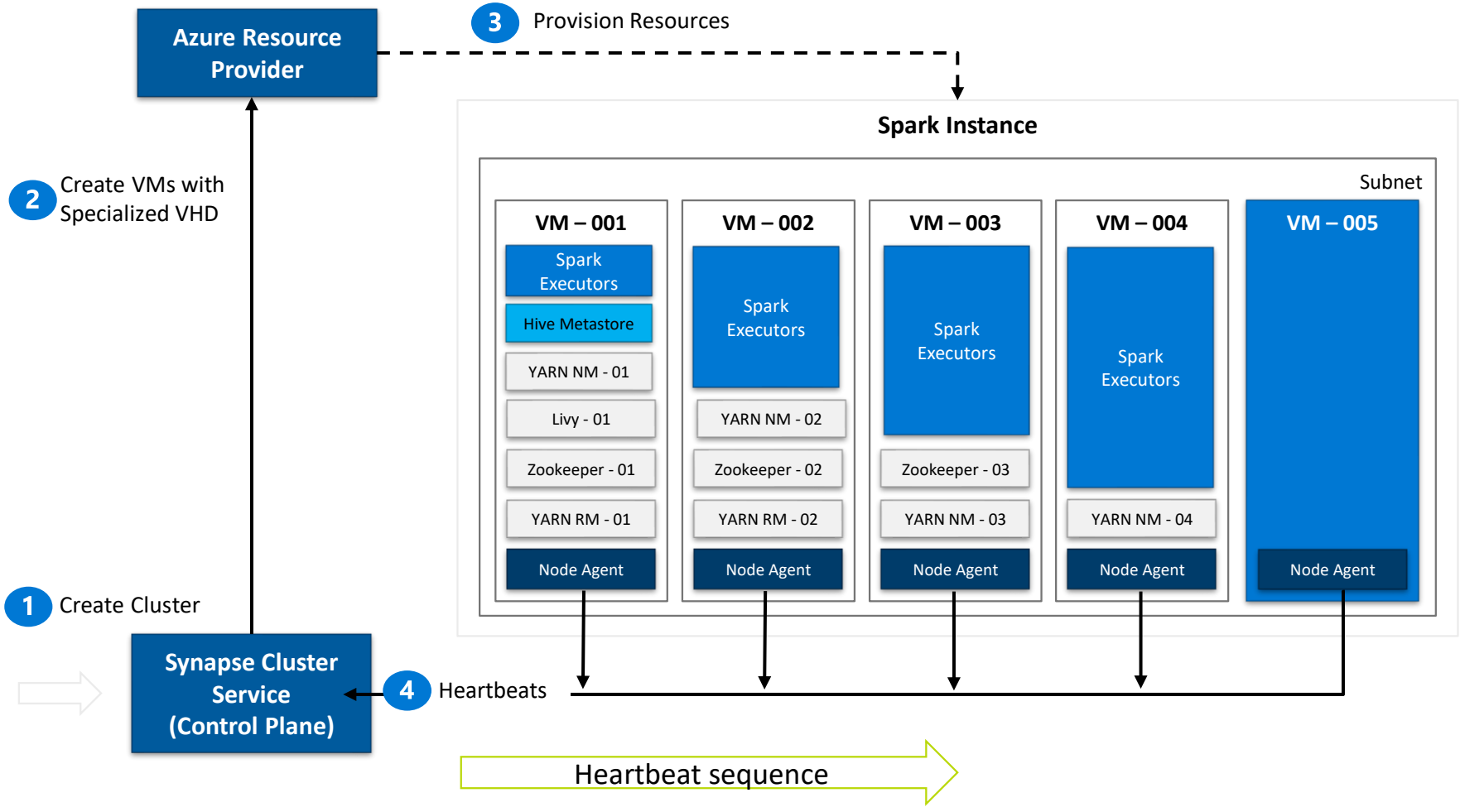
Architecture Overview

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.

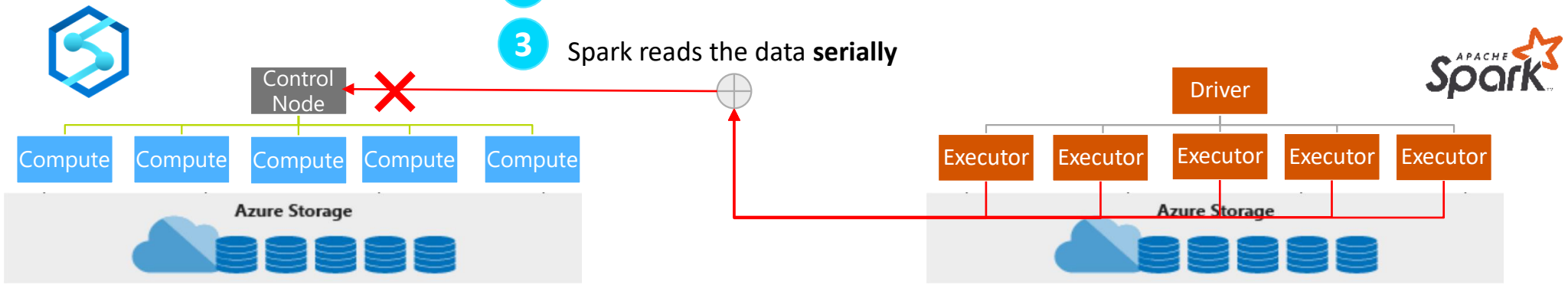
Synapse Spark Instances



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 end-point to the Job Service.

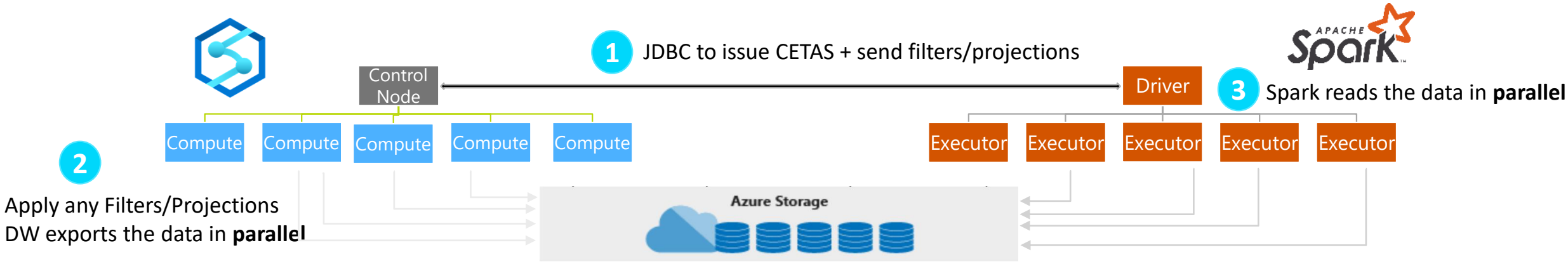
Existing Approach: JDBC

1. JDBC to open connection
2. Apply any Filters/Projections
3. Spark reads the data **serially**



New Approach: JDBC and Polybase

1. JDBC to issue CETAS + send filters/projections
2. Apply any Filters/Projections
   DW exports the data in **parallel**
3. Spark reads the data in **parallel**



User Provisioned Workspace-Default Data Lake

Code-Behind Experience

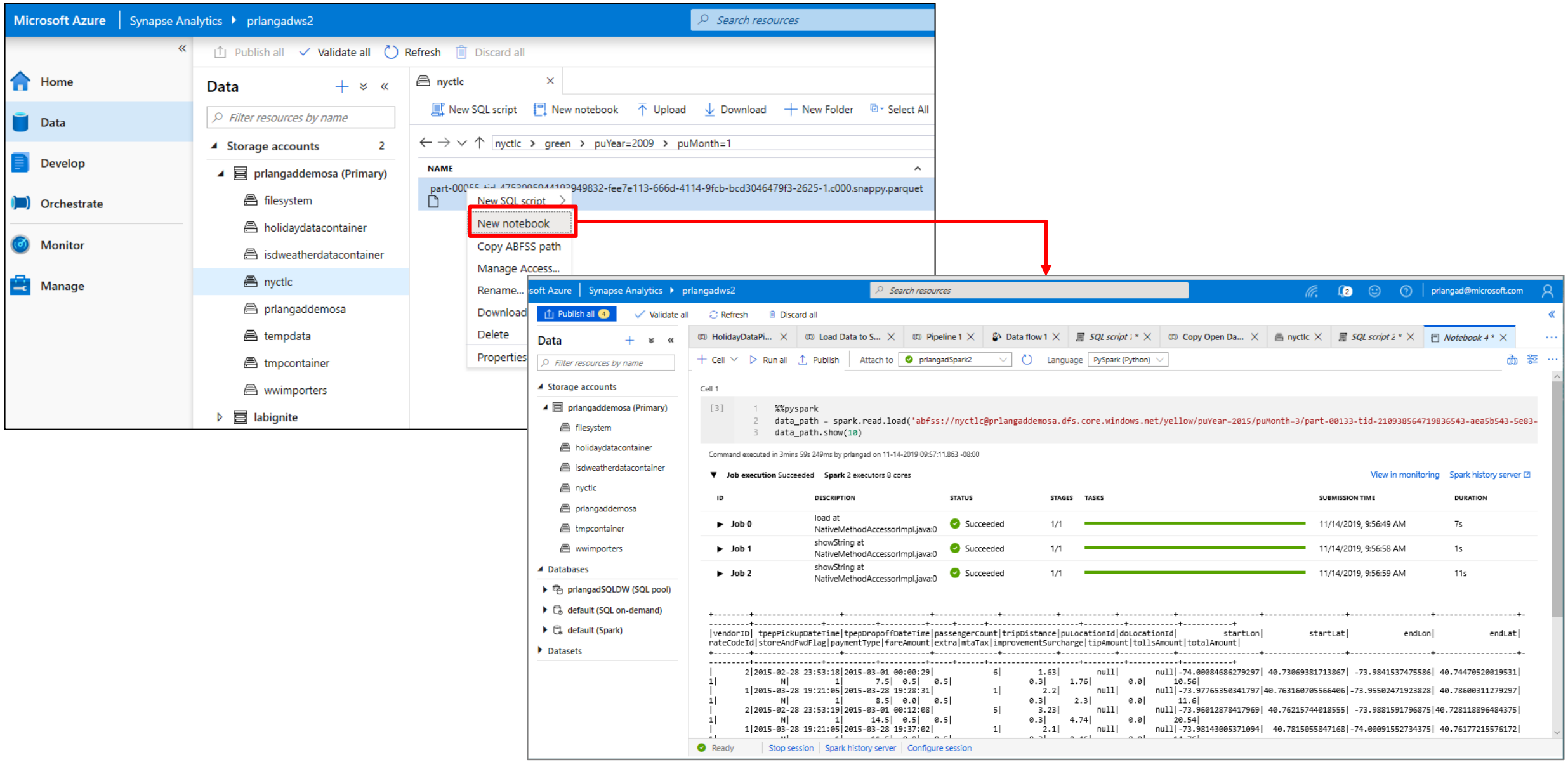
Existing Approach

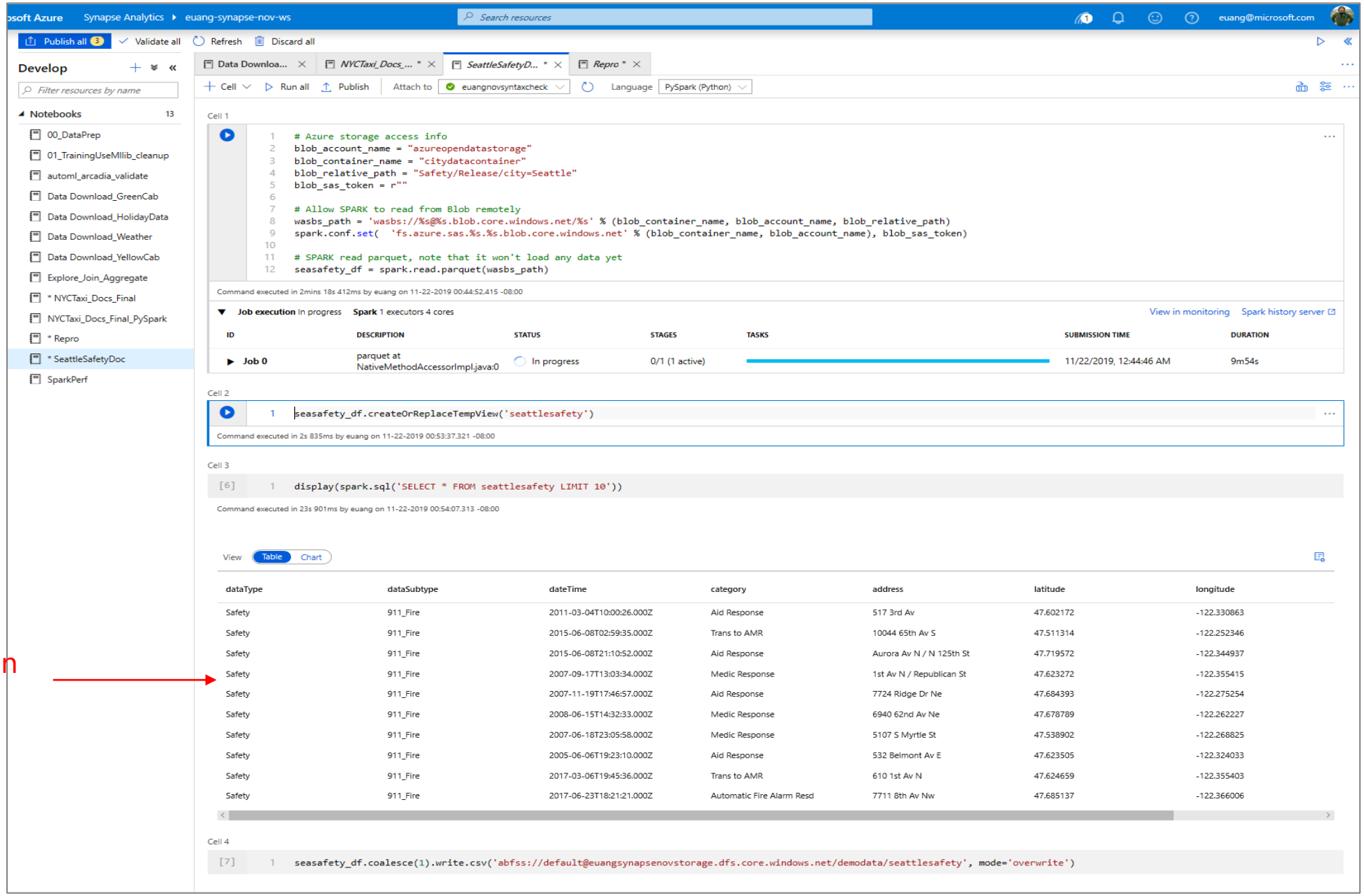
```
val jdbcUsername = "<SQL DB ADMIN USER>"
val jdbcPwd = "<SQL DB ADMIN PWD>"
val jdbcHostname = "servername.database.windows.net”
val jdbcPort = 1433
val jdbcDatabase ="<AZURE SQL DB NAME>“


val jdbc_url =
s"jdbc:sqlserver://${jdbcHostname}:${jdbcPort};database=${jdbcDatabase};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.databas
e.windows.net;loginTimeout=60;“

val connectionProperties = new Properties()

connectionProperties.put("user", s"${jdbcUsername}")
connectionProperties.put("password", s"${jdbcPwd}")

val sqlTableDf = spark.read.jdbc(jdbc_url, “dbo.Tbl1", connectionProperties)
```

New Approach

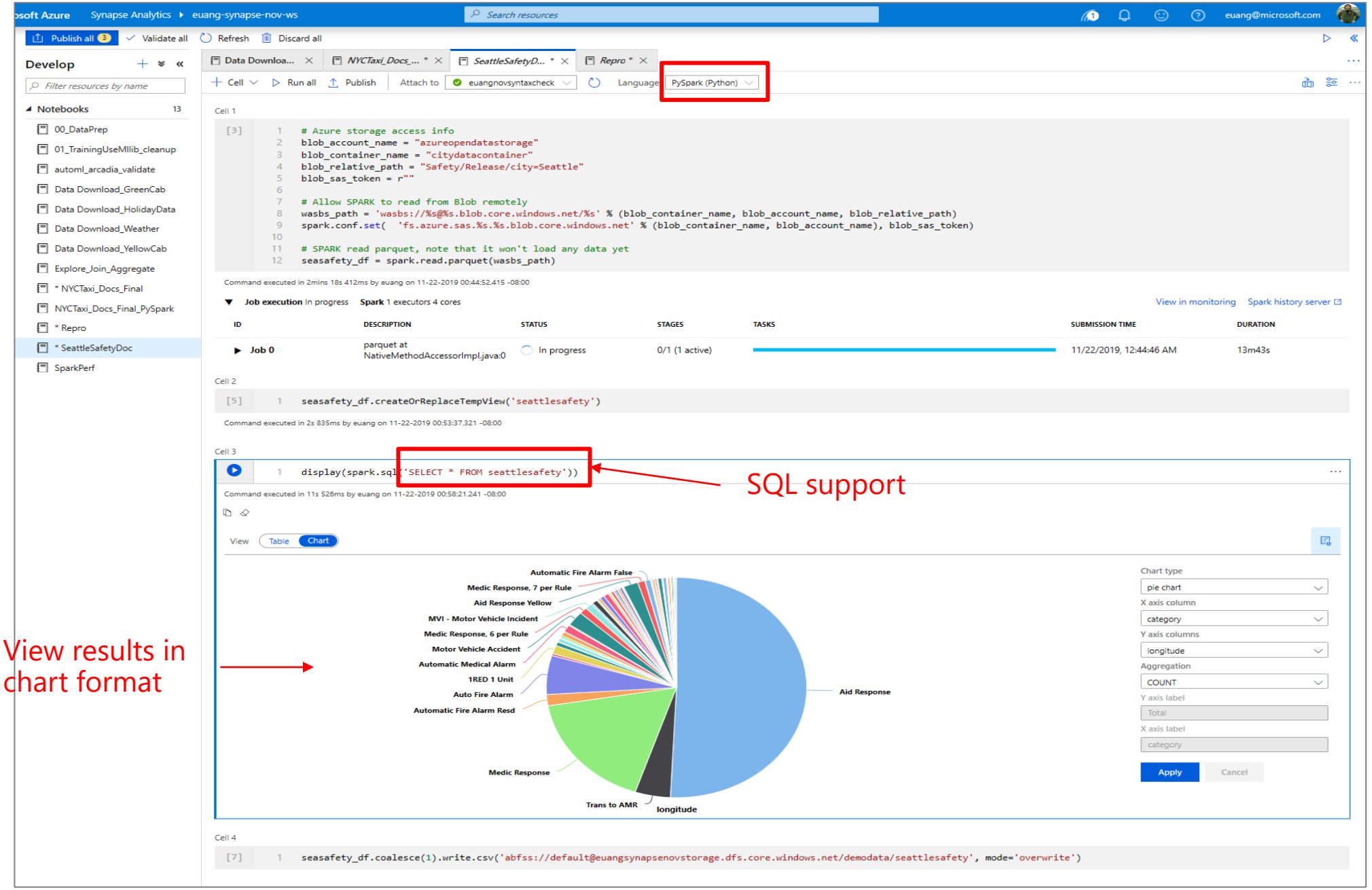
```
// Construct a Spark DataFrame from SQL Provisioned
var df = spark.read.sqlanalytics("sql1.dbo.Tbl1")

// Write the Spark DataFrame into SQL Provisioned
df.write.sqlanalytics(“sql1.dbo.Tbl2”)
```

Create Notebook on files in storage





View results in table format


Synapse Analytics
euang-synapse-nov-ws
Search resources
euang@microsoft.com
Publish all
Validate all
Refresh
Discard all
Develop
Filter resources by name
Notebooks
00_DataPrep
01_TrainingUseMllib_cleanup
automl_arcadia_validate
Data Download_GreenCab
Data Download_HolidayData
Data Download_Weather
Data Download_YellowCab
Explore_Join_Aggregate
* NYCTaxi_Docs_Final
NYCTaxi_Docs_Final_PySpark
* Repro
* SeattleSafetyDoc
SparkPerf
Cell
Run all
Publish
Attach to
euangnovsyntaxcheck
Language
PySpark (Python)
Cell 1
# Azure storage access info
blob_account_name = "azureopendatastorage"
blob_container_name = "citydatacontainer"
blob_relative_path = "Safety/Release/city=Seattle"
blob_sas_token = r""
# Allow SPARK to read from Blob remotely
wasbs_path = 'wasbs://%s@%s.blob.core.windows.net/%s' % (blob_container_name, blob_account_name, blob_relative_path)
spark.conf.set('fs.azure.sas.%s.%s.blob.core.windows.net' % (blob_container_name, blob_account_name), blob_sas_token)
# SPARK read parquet, note that it won't load any data yet
seasafety_df = spark.read.parquet(wasbs_path)
Command executed in 2mins 18s 412ms by euang on 11-22-2019 00:44:52.415 -08:00
Job execution In progress
Spark 1 executors 4 cores
View in monitoring
Spark history server
ID
DESCRIPTION
STATUS
STAGES
TASKS
SUBMISSION TIME
DURATION
Job 0
parquet at NativeMethodAccessorImpl.java:0
In progress
0/1 (1 active)
11/22/2019, 12:44:46 AM
13m43s
Cell 2
seasafety_df.createOrReplaceTempView('seattlesafety')
Command executed in 2s 835ms by euang on 11-22-2019 00:53:37.321 -08:00
Cell 3
display(spark.sql('SELECT * FROM seattlesafety'))
Command executed in 11s 526ms by euang on 11-22-2019 00:58:21.241 -08:00
View
Table
Chart
Automatic Fire Alarm False
Medic Response, 7 per Rule
Aid Response Yellow
MVI - Motor Vehicle Incident
Medic Response, 6 per Rule
Motor Vehicle Accident
Automatic Medical Alarm
1RED 1 Unit
Auto Fire Alarm
Automatic Fire Alarm Resd
Medic Response
Trans to AMR
longitude
Aid Response
Chart type
pie chart
X axis column
category
Y axis columns
longitude
Aggregation
COUNT
Y axis label
Total
X axis label
category
Apply
Cancel
Cell 4
seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')
SQL support
View results in chart format

Publish all 1 | Validate all | Refresh | Discard all

Develop

Filter resources by name

Notebooks 13

- 00_DataPrep
- 01_TrainingUseMllib_cleanup
- automl_arcadia_validate
- Data Download_GreenCab
- Data Download_HolidayData
- Data Download_Weather
- Data Download_YellowCab
- Explore_Join_Aggregate
- * NYCTaxi_Docs_Final
- NYCTaxi_Docs_Final_PySpark
- Repro
- SeattleSafetyDoc
- SparkPerf

Data Downloa... | NYCTaxi_Docs_... *

Cell | Run all | Publish | Attach to: Select Spark pool | Language: PySpark (Python)

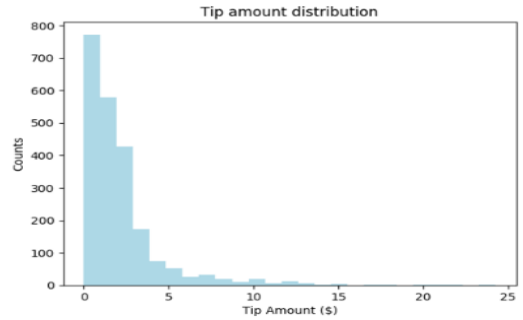
```
# Creating a temp table allows easier manipulation during the session, they are not persisted between sessions,
# for that write the data to storage like above.
sampled_taxi_df.createOrReplaceTempView("nytaxi")
```

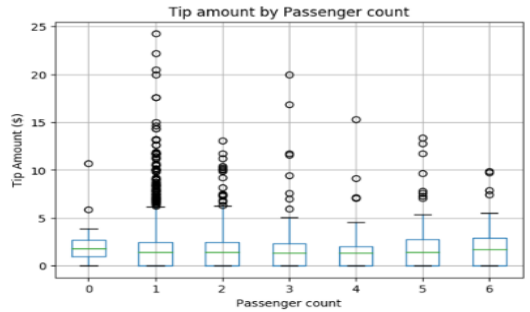
Exploratory Data Analysis

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

Cell 9

```
#The charting package needs a Pandas dataframe or numpy array do the conversion
sampled_taxi_pd_df = sampled_taxi_df.toPandas()

# Look at tips by amount count histogram
ax1 = sampled_taxi_pd_df['tipAmount'].plot(kind='hist', bins=25, facecolor='lightblue')
ax1.set_title('Tip amount distribution')
ax1.set_xlabel('Tip Amount ($)')
ax1.set_ylabel('Counts')
plt.suptitle('')
plt.show()

# How many passengers tip'd by various amounts
ax2 = sampled_taxi_pd_df.boxplot(column=['tipAmount'], by=['passengerCount'])
ax2.set_title('Tip amount by Passenger count')
ax2.set_xlabel('Passenger count')
ax2.set_ylabel('Tip Amount ($)')
plt.suptitle('')
plt.show()

# Look at the relationship between fare and tip amounts
ax = sampled_taxi_pd_df.plot(kind='scatter', x= 'fareAmount', y = 'tipAmount', c='blue', alpha = 0.10, s=2.5*(sampled_taxi_pd_df['passengerCount']))
ax.set_title('Tip amount by Fare amount')
ax.set_xlabel('Fare Amount ($)')
ax.set_ylabel('Tip Amount ($)')
plt.axis([-2, 80, -2, 20])
plt.suptitle('')
plt.show()
```





Tip amount by Fare amount

Exploratory data analysis with graphs – histogram, boxplot etc

Not Started | Configure session

Library Management - Python

Overview

Customers can add new python libraries at Spark pool level

Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

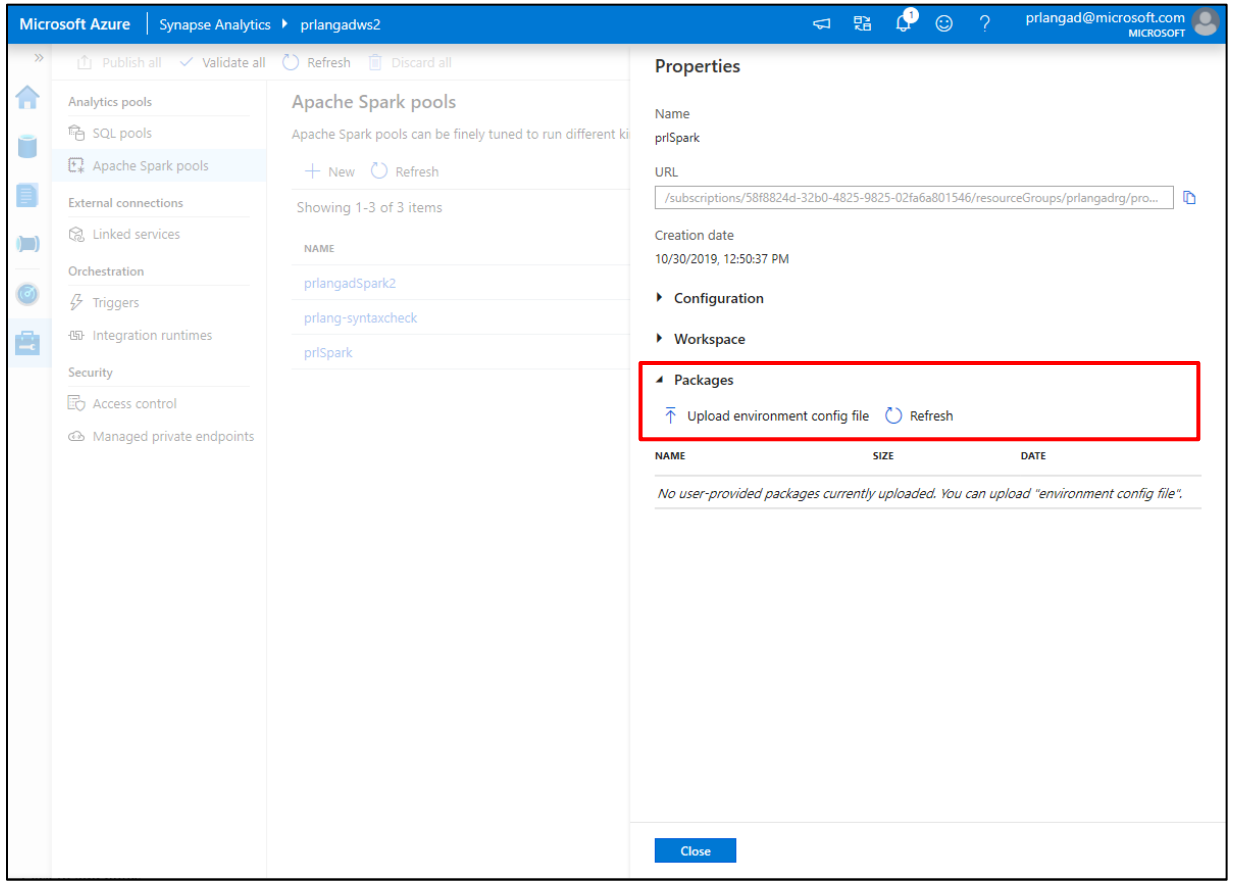
Update versions of existing libraries on your cluster

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

Constraints

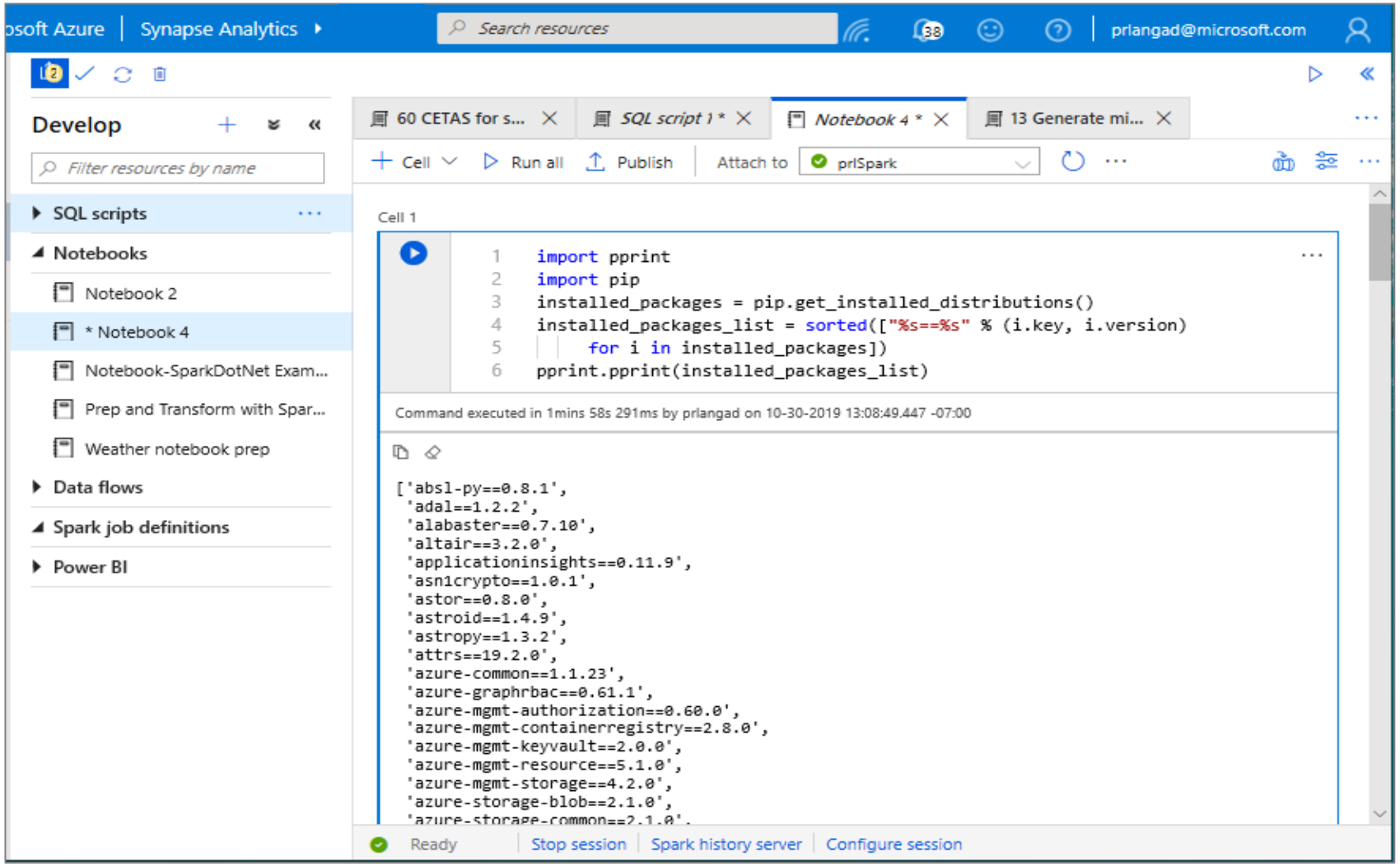
The library version must exist on PyPI repository

Version downgrade of an existing library not allowed



Library Management - Python

Get list of installed libraries with version information



Spark ML Algorithms

Classification and Regression	• Linear Models (SVMs, logistic regression, linear regression) • Naïve Bayes • Decision Trees • Ensembles of trees (Random Forest, Gradient-Boosted Trees) • Isotonic regression
Clustering	• k-means and streaming k-means • Gaussian mixture • Power iteration clustering (PIC) • Latent Dirichlet allocation (LDA)
Collaborative Filtering	• Alternating least squares (ALS)
Dimensionality Reduction	• SVD • PCA
Frequent Pattern Mining	• FP-growth • Association rules
Basic Statistics	• 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

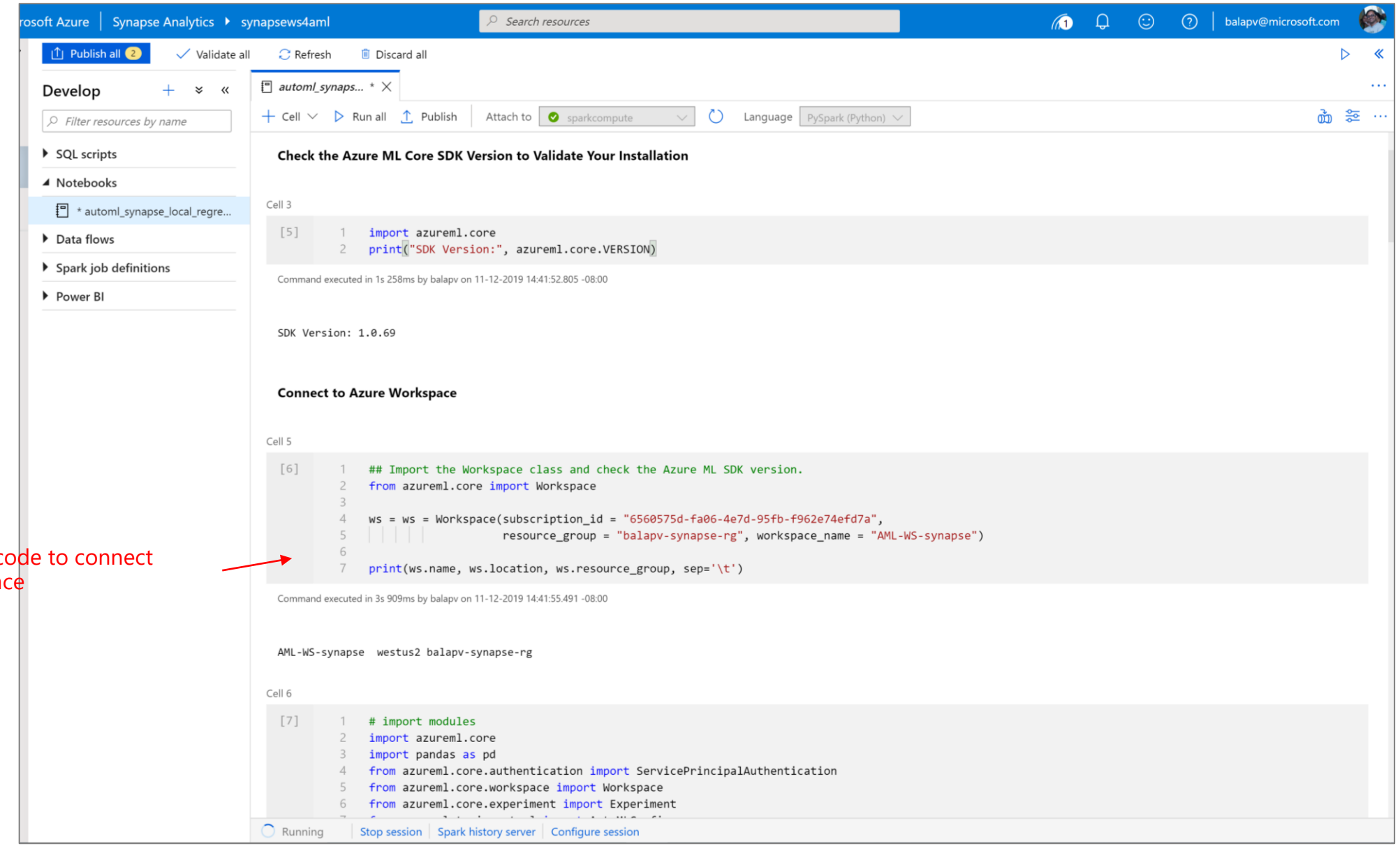
Microservice Orchestration

Multilingual Binding Generation

www.aka.ms/spark

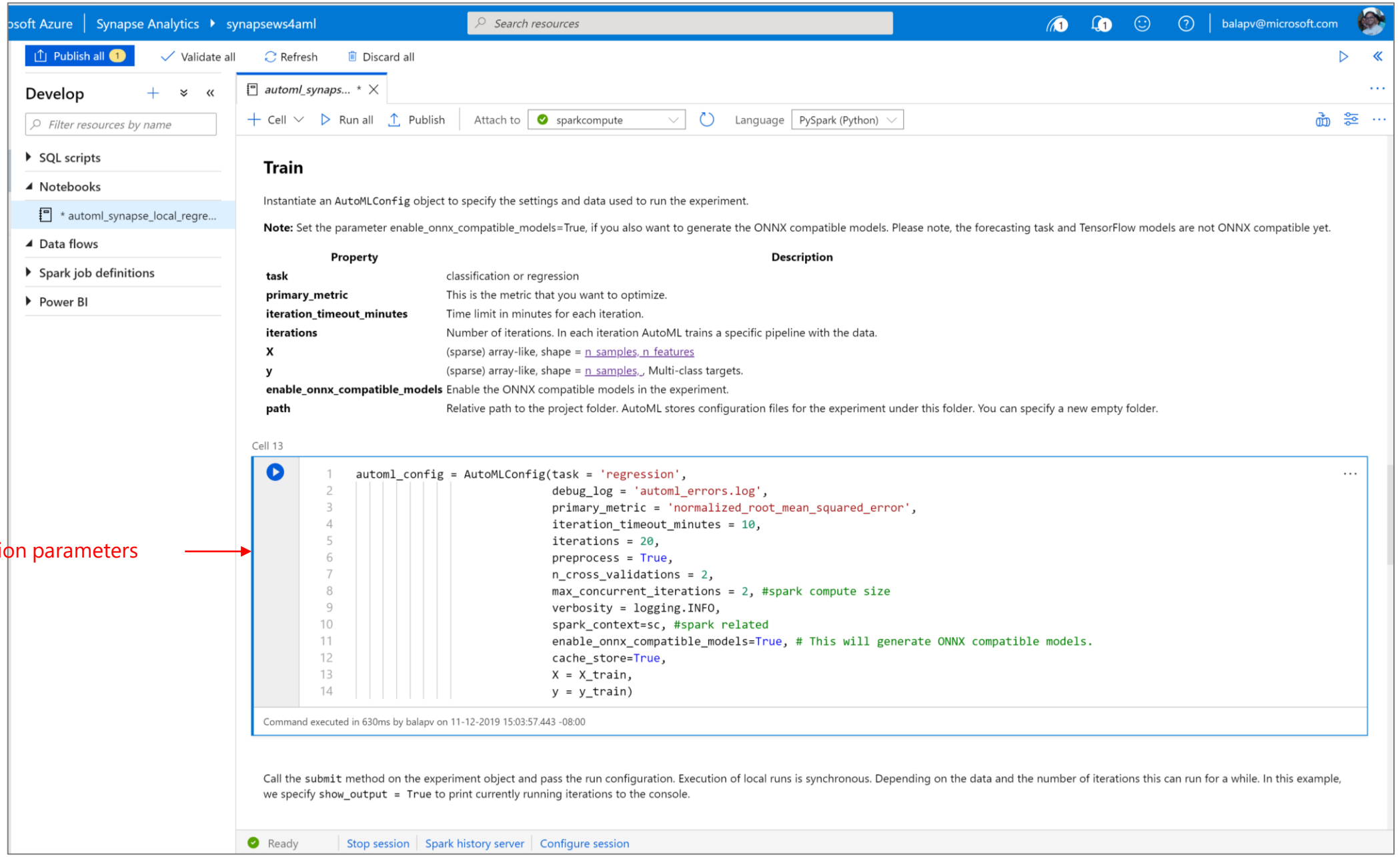
Azure/mmlspark

Synapse Notebook: Connect to AML workspace



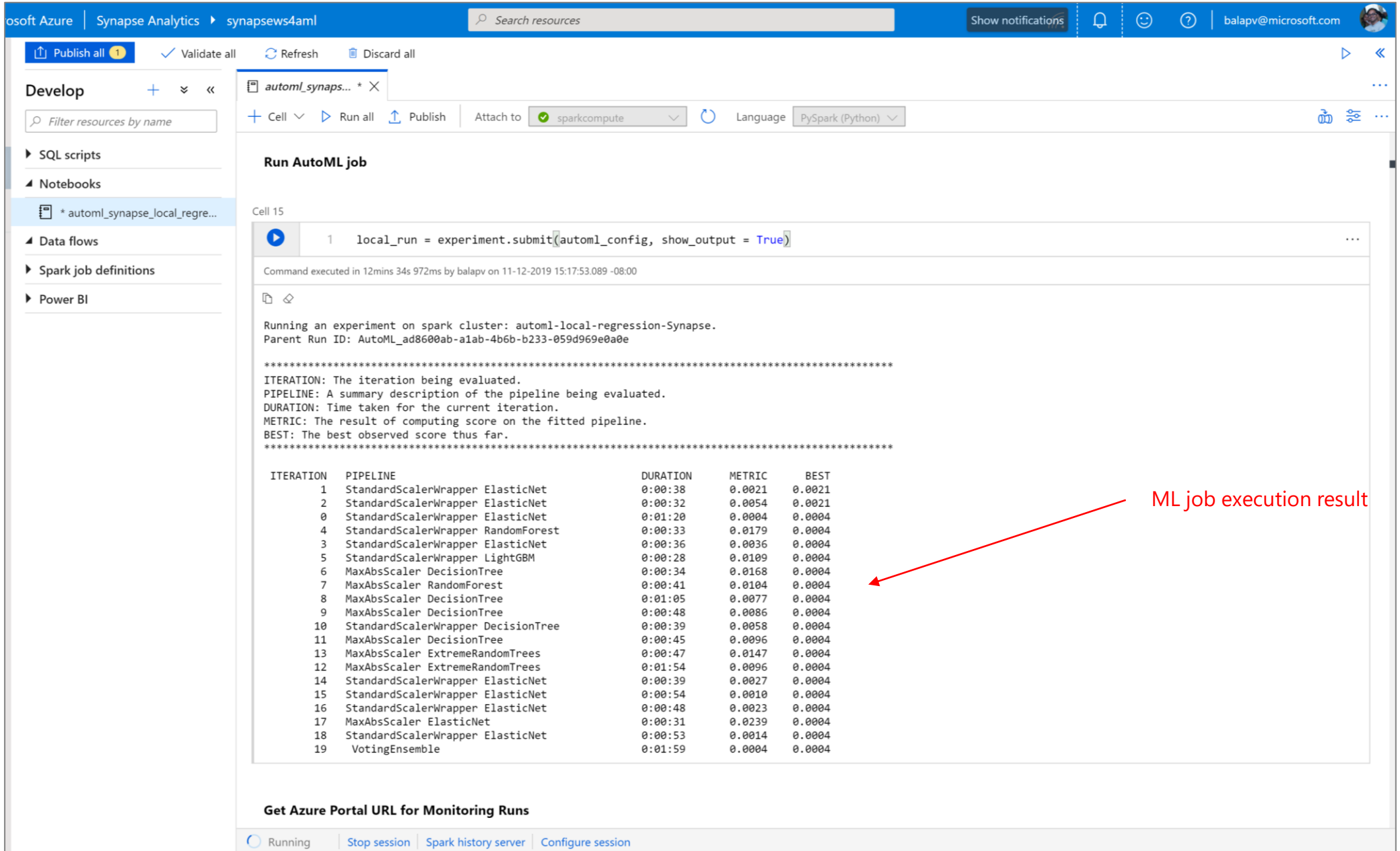
Simple code to connect workspace

Synapse Notebook: Configure AML job to run on Synapse



Configuration parameters

Synapse Notebook: Run AML job



References

https://azure.microsoft.com/en-us/updates/apache-spark-31-for-azure-synapse-analytics-now-generally-available/

https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/apache-spark-version-support

https://techcommunity.microsoft.com/t5/azure-synapse-analytics/managed-and-external-table-on-serverless/ba-p/2188190

https://techcommunity.microsoft.com/t5/azure-synapse-analytics/use-spark-scala-to-write-data-from-adls-to-synapse-dedicated/ba-p/2210382

Ingest data: https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/synapse-spark-sql-pool-import-export

https://techcommunity.microsoft.com/t5/azure-synapse-analytics/kickstart-your-apache-spark-learning-in-azure-synapse-with/ba-p/1988136

https://techcommunity.microsoft.com/t5/azure-synapse-analytics/apache-spark-in-azure-synapse-performance-update/ba-p/2243534

https://databricks.com/session_na20/hyperspace-an-indexing-subsystem-for-apache-spark

https://github.com/microsoft/hyperspace

https://github.com/microsoft/hyperspace/blob/master/notebooks/python/Hitchhikers%20Guide%20to%20Hyperspace.ipynb

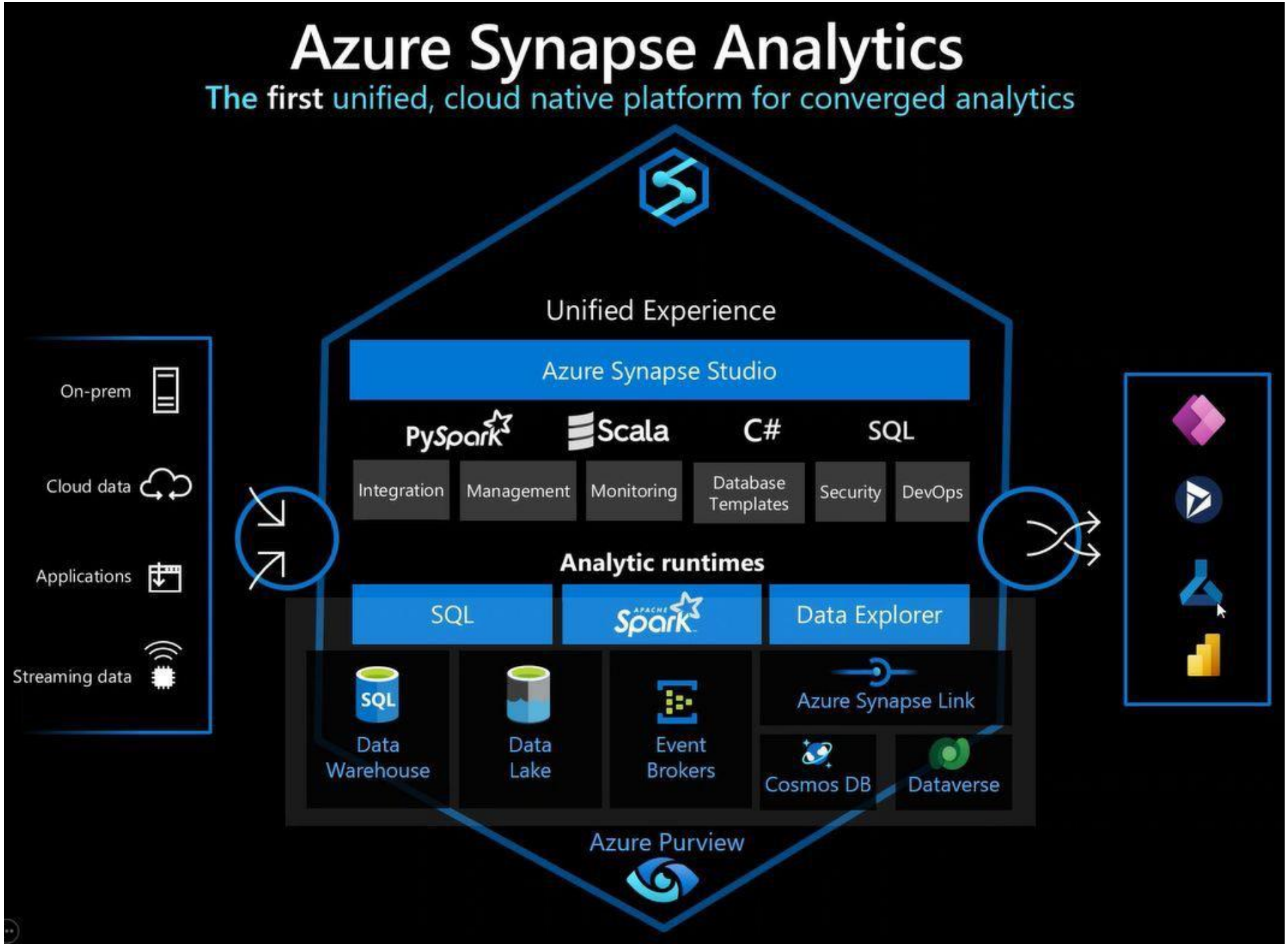
https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/query-delta-lake-format

Apache Spark in Azure Synapse - Performance Update - Microsoft Tech Community

Spark Performance Optimization (microsoft.com)

Azure Synapse Analytics

The first unified, cloud native platform for converged analytics

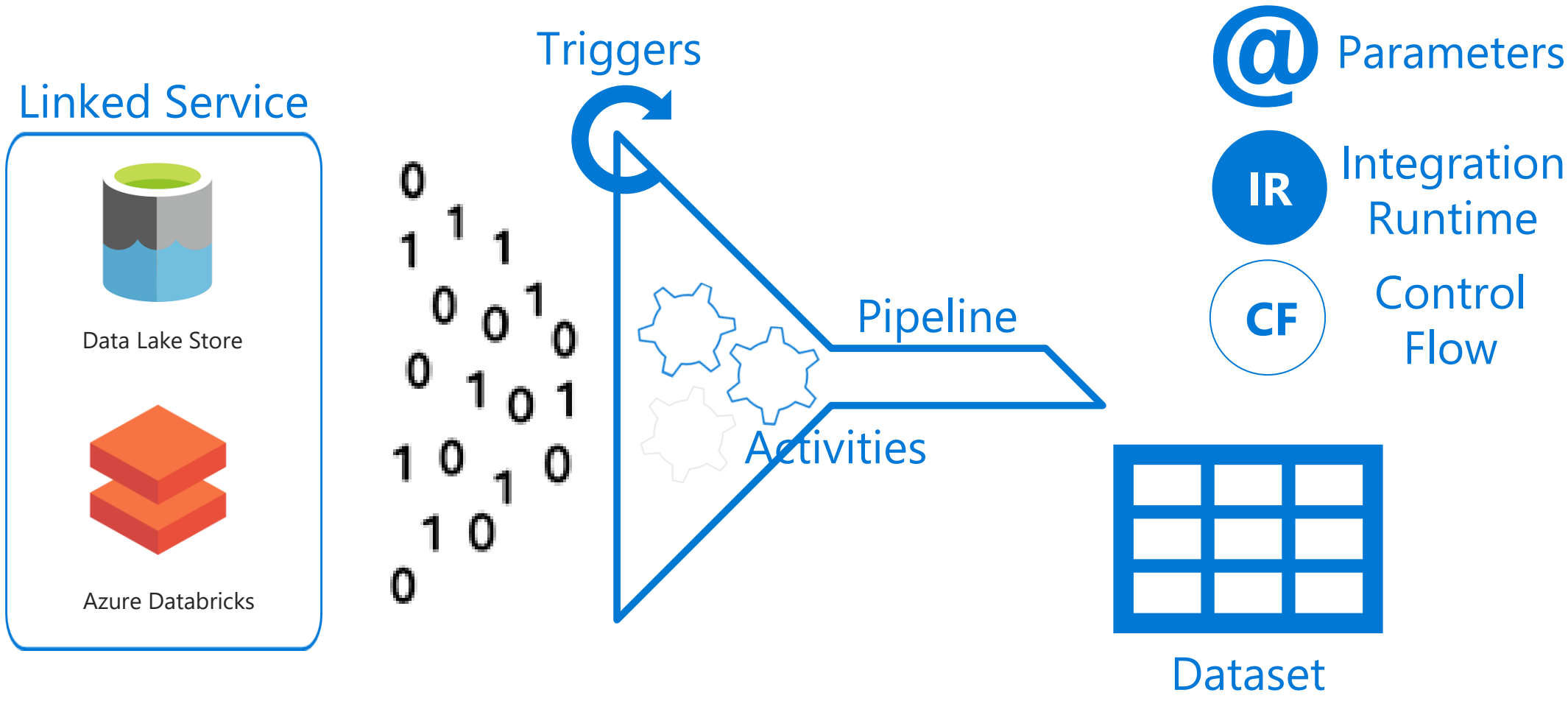


Microsoft

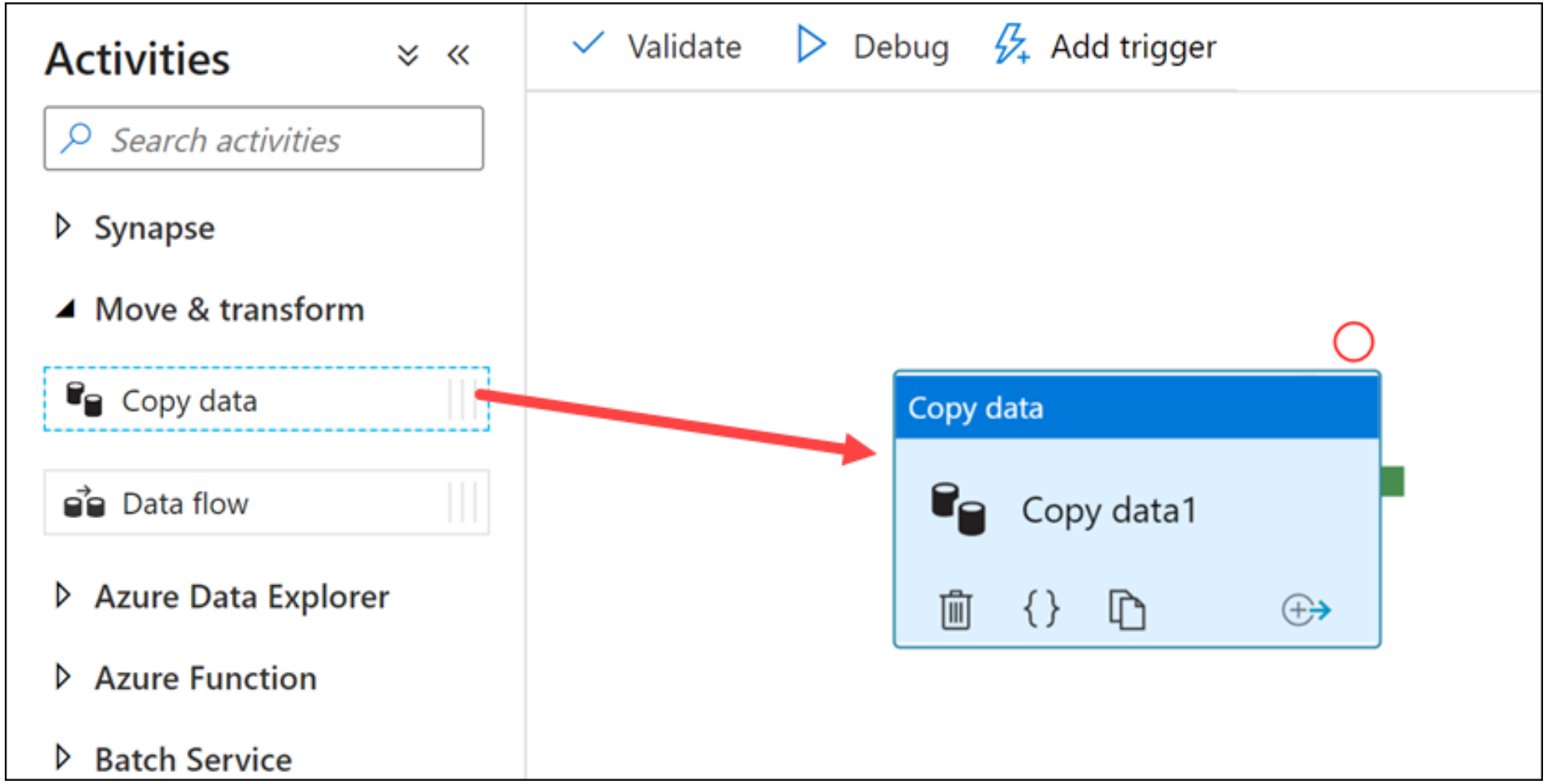
Azure Synapse Analytics

Pipelines

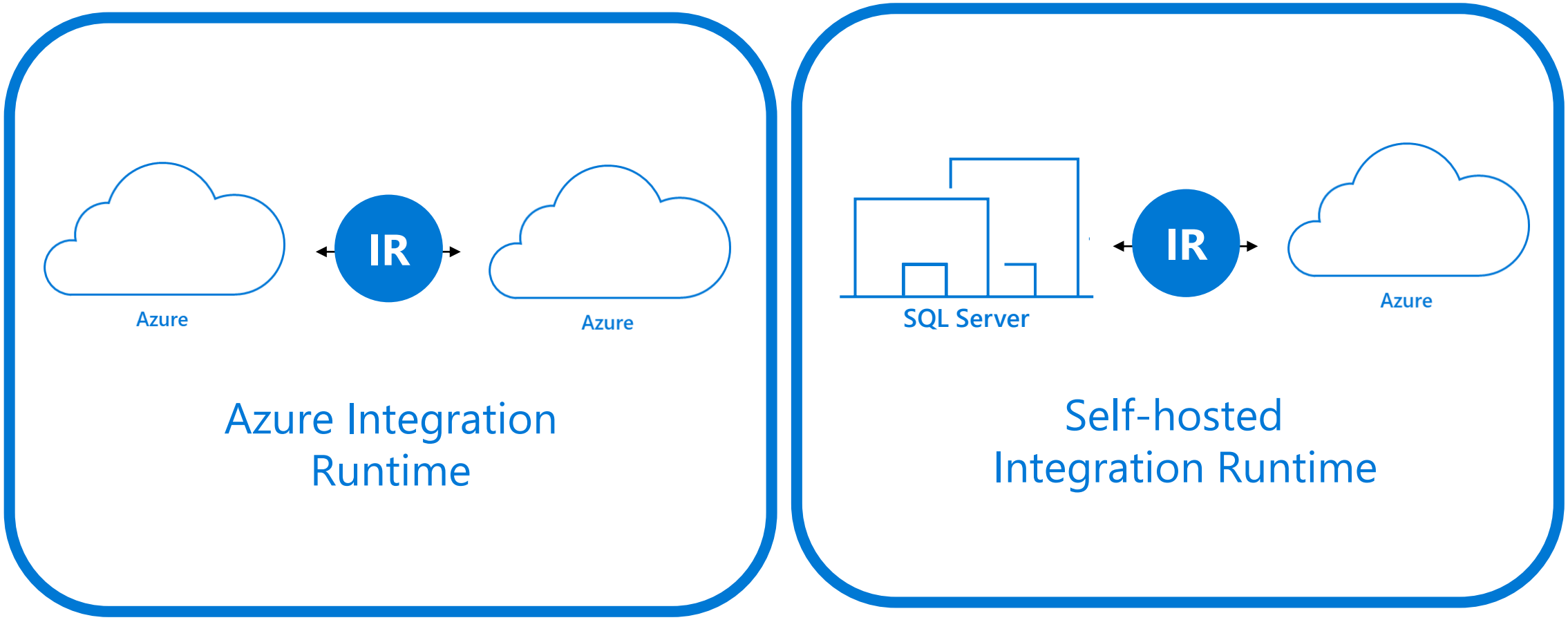
Azure Data Factory



Petabyte-scale ingestion



Understanding integration



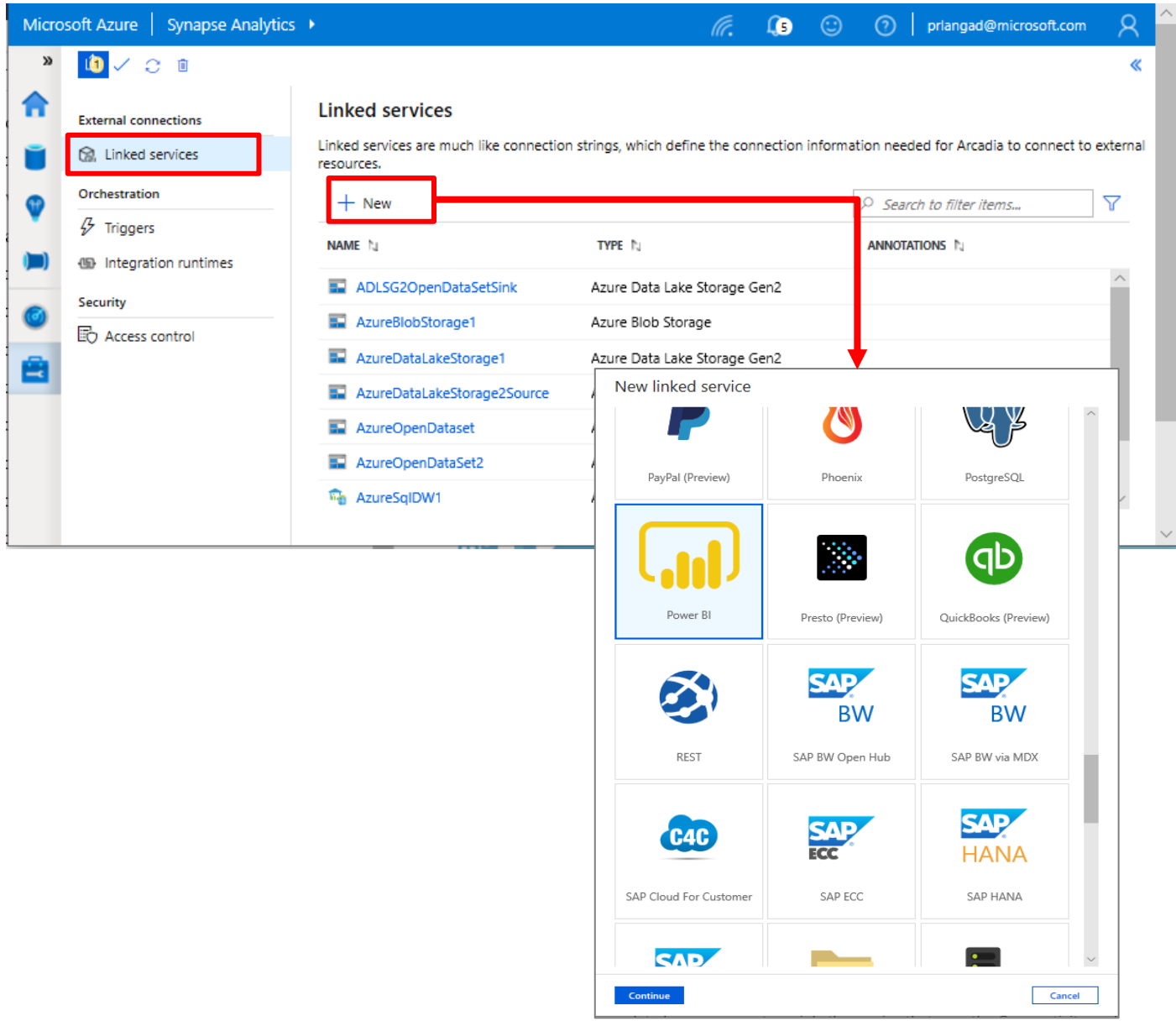
Linked services

Overview

Linked services define 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



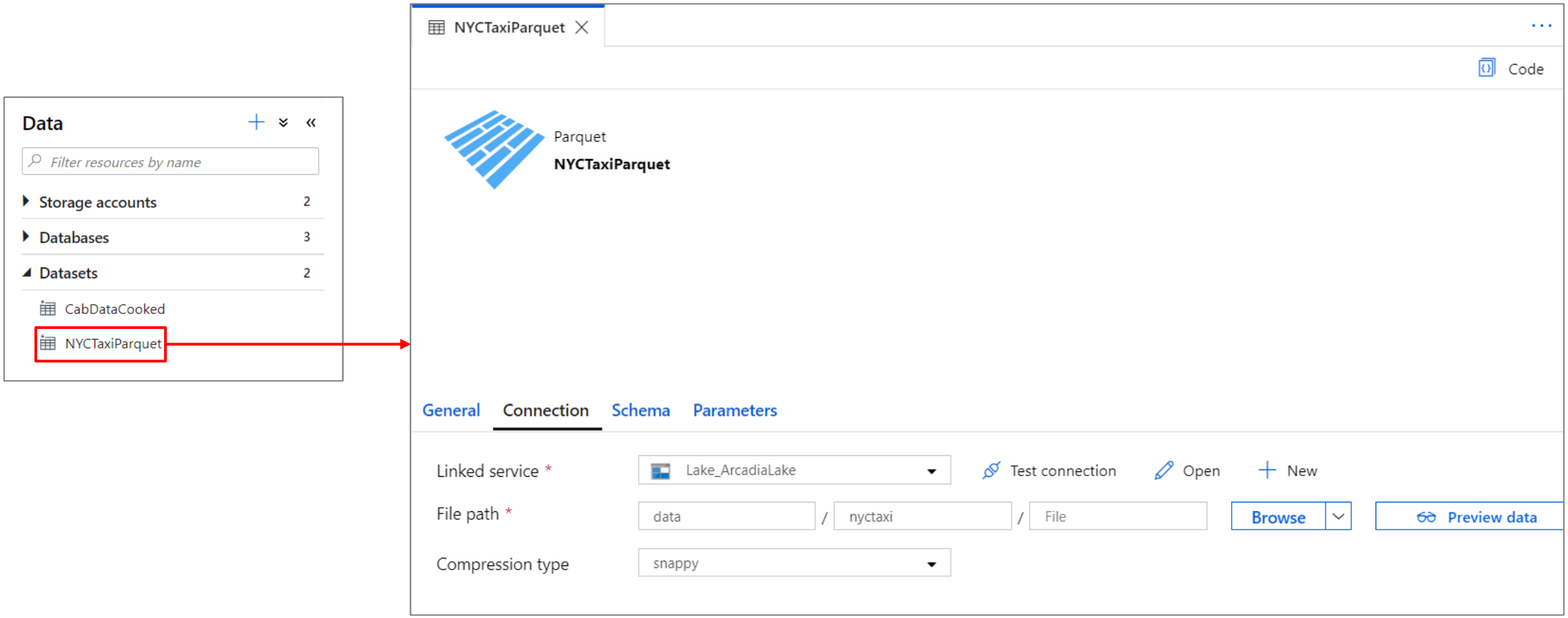
90+ Connectors out of the box

Azure (15)	Database & DW (26)		File Storage (6)	File Formats(6)	NoSQL (3)	Services and App (28)		Generic (4)
Blob storage	Amazon Redshift	Oracle	Amazon S3	AVRO	Cassandra	Amazon MWS	Oracle Service Cloud	Generic HTTP
Cosmos DB - SQL API	DB2	Phoenix	File system	Binary	Couchbase	CDS for Apps	PayPal	Generic OData
Cosmos DB - MongoDB API	Drill	PostgreSQL	FTP	Delimited Text	MongoDB	Concur	QuickBooks	Generic ODBC
Data Explorer	Google BigQuery	Presto	Google Cloud Storage	JSON		Dynamics 365	Salesforce	Generic REST
Data Lake Storage Gen1	Greenplum	SAP BW Open Hub	HDFS	ORC		Dynamics AX	SF Service Cloud	
Data Lake Storage Gen2	HBase	SAP BW via MDX	SFTP	Parquet		Dynamics CRM	SF Marketing Cloud	
Database for MariaDB	Hive	SAP HANA				Google AdWords	SAP C4C	
Database for MySQL	Apache Impala	SAP table				HubSpot	SAP ECC	
Database for PostgreSQL	Informix	Spark				Jira	ServiceNow	
File Storage	MariaDB	SQL Server				Magento	Shopify	
SQL Database	Microsoft Access	Sybase				Marketo	Square	
SQL Database MI	MySQL	Teradata				Office 365	Web table	
SQL Data Warehouse	Netezza	Vertica				Oracle Eloqua	Xero	
Search index	Snowflake					Oracle Responsys	Zoho	
Table storage								

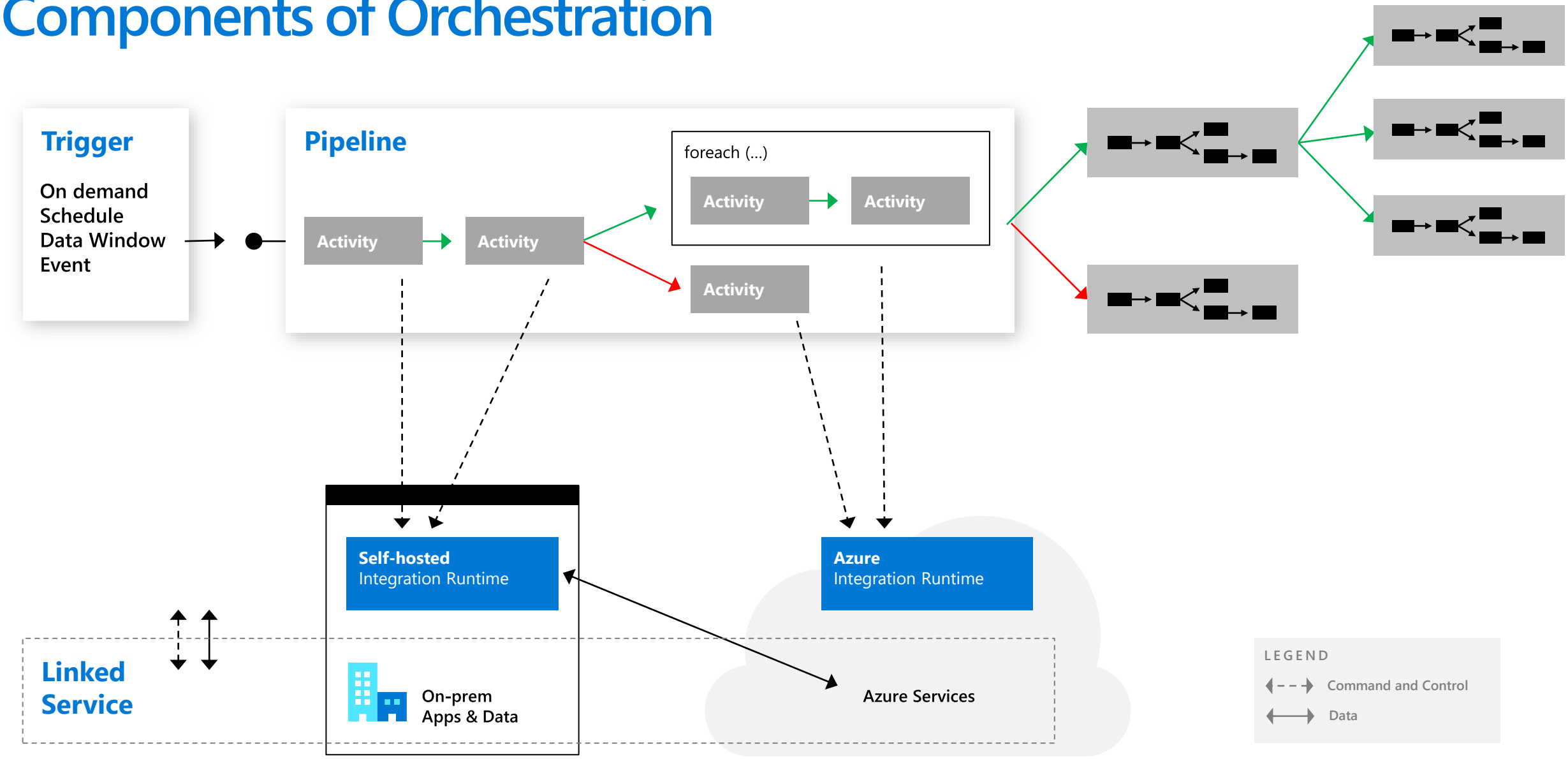
Datasets

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.



Components of Orchestration

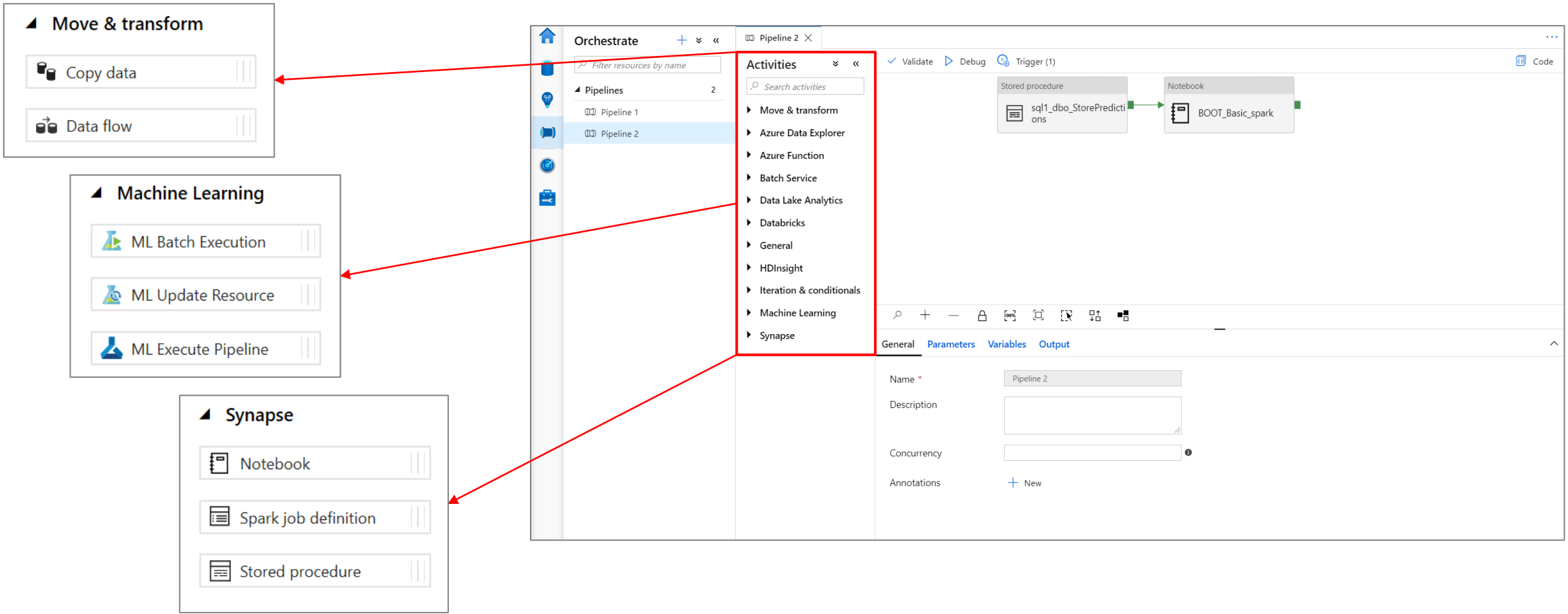


Synapse Pipelines shares codebase with Azure Data Factory

Pipelines

Create pipelines to ingest, transform and load data with 90+ inbuilt connectors.

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



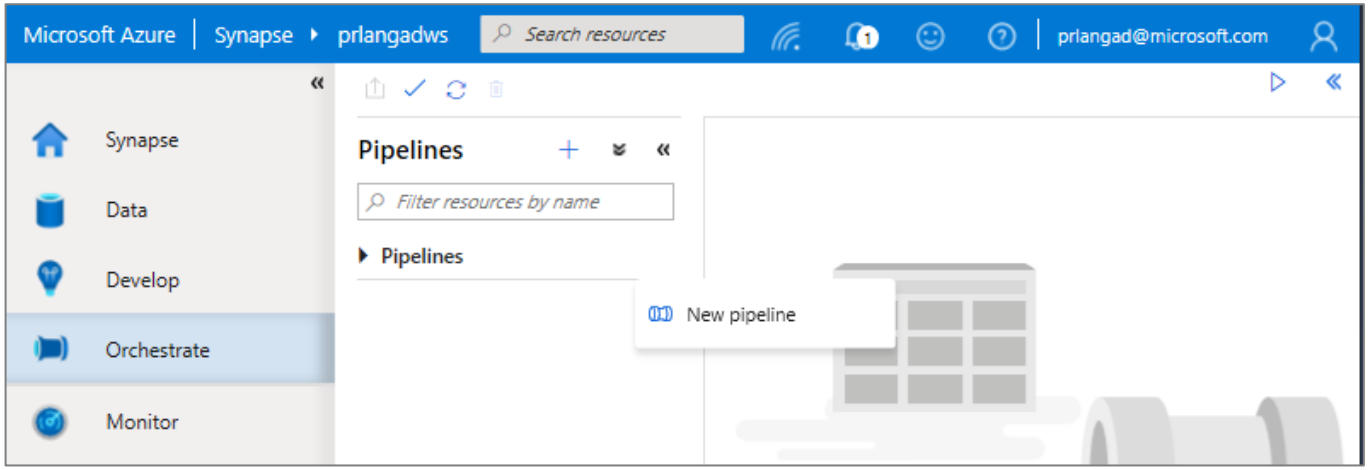
Pipelines

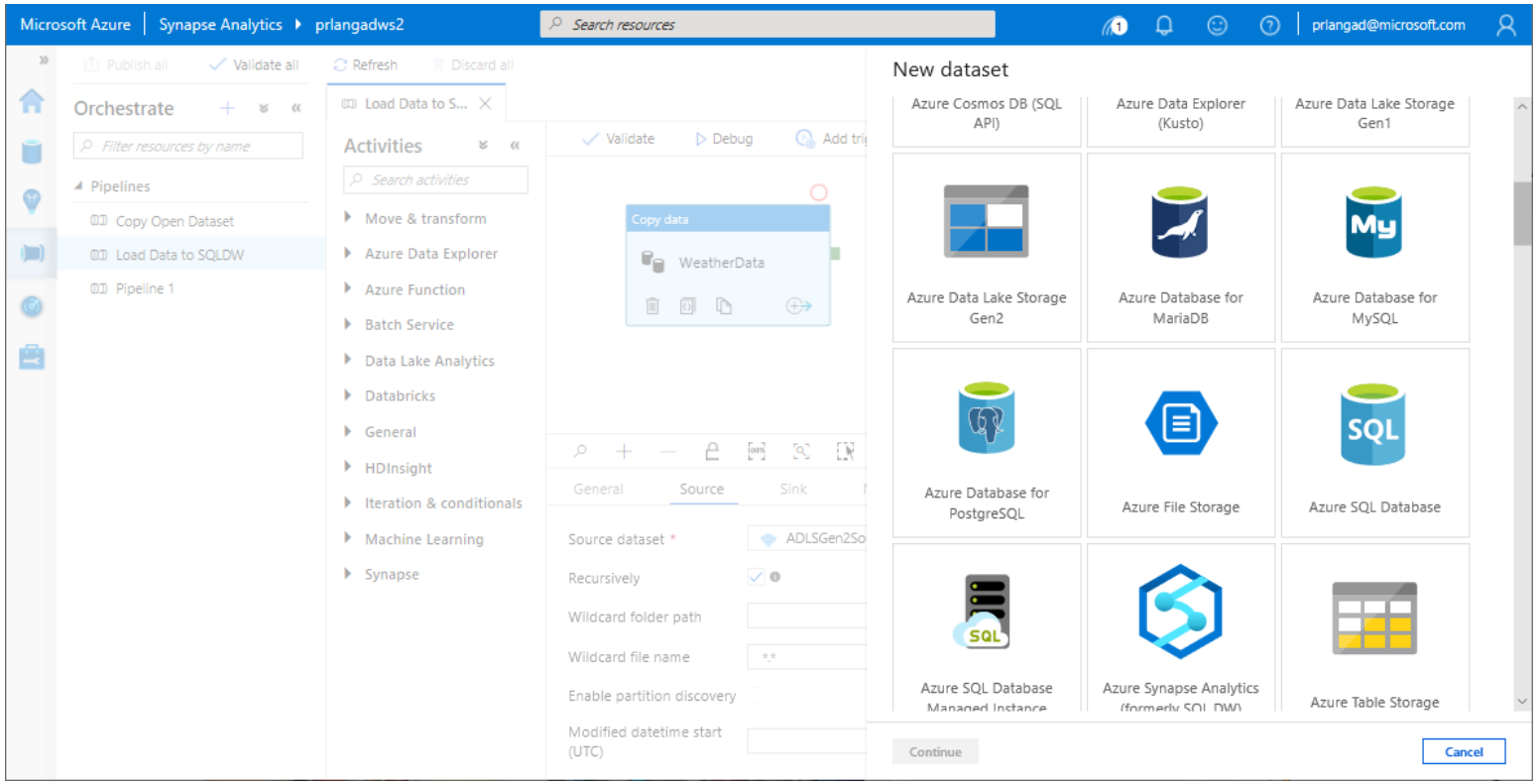
Overview

- Provide 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.

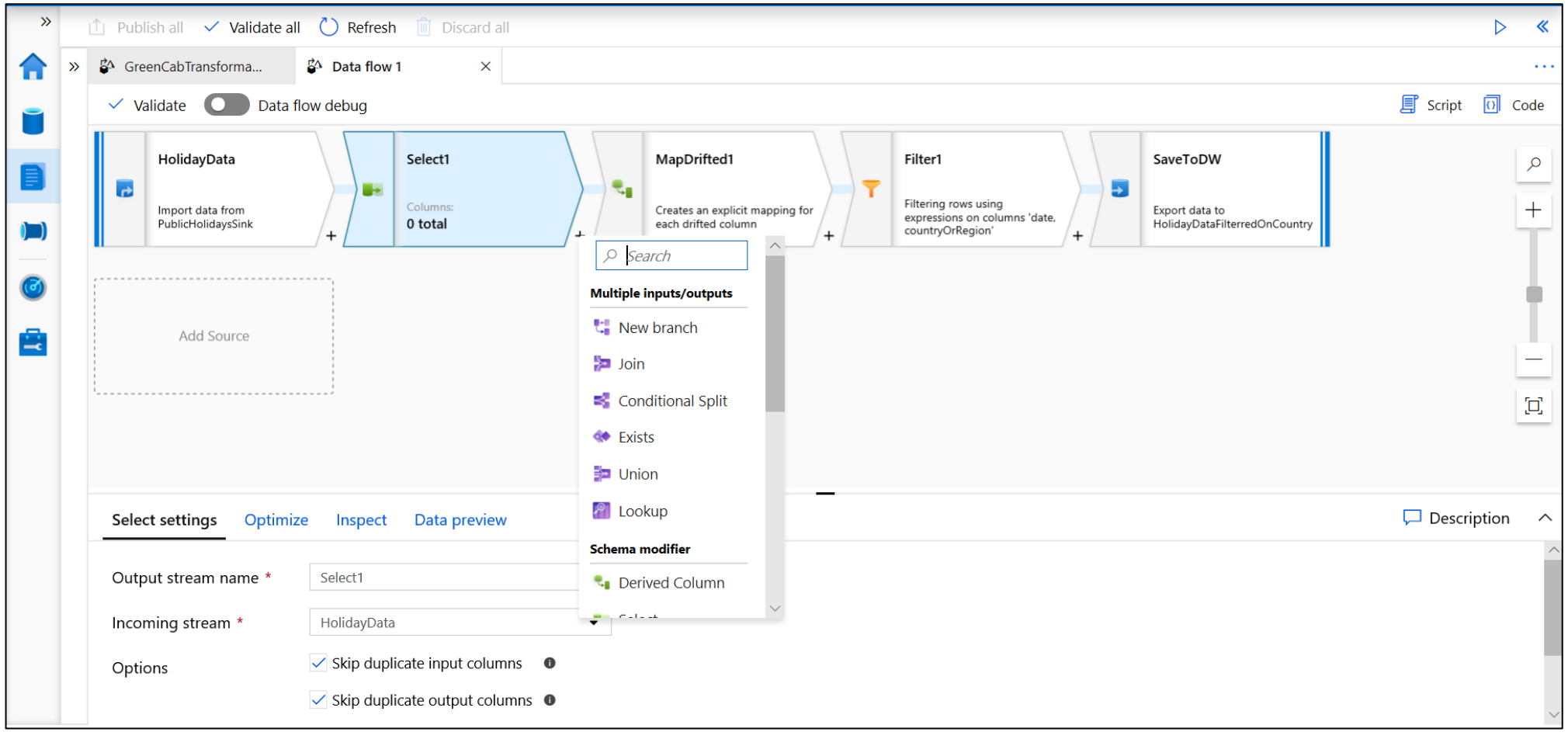




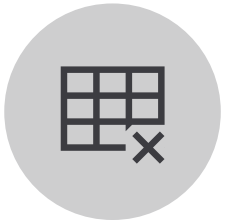
Develop Hub - Data Flows

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

Provides a code-free experience.



Dataflow Capabilities



Handle upserts, updates, deletes on sql sinks

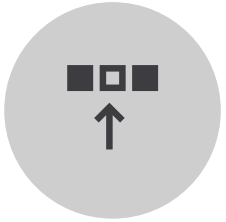
Add new partition methods

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)

Triggers

Overview

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

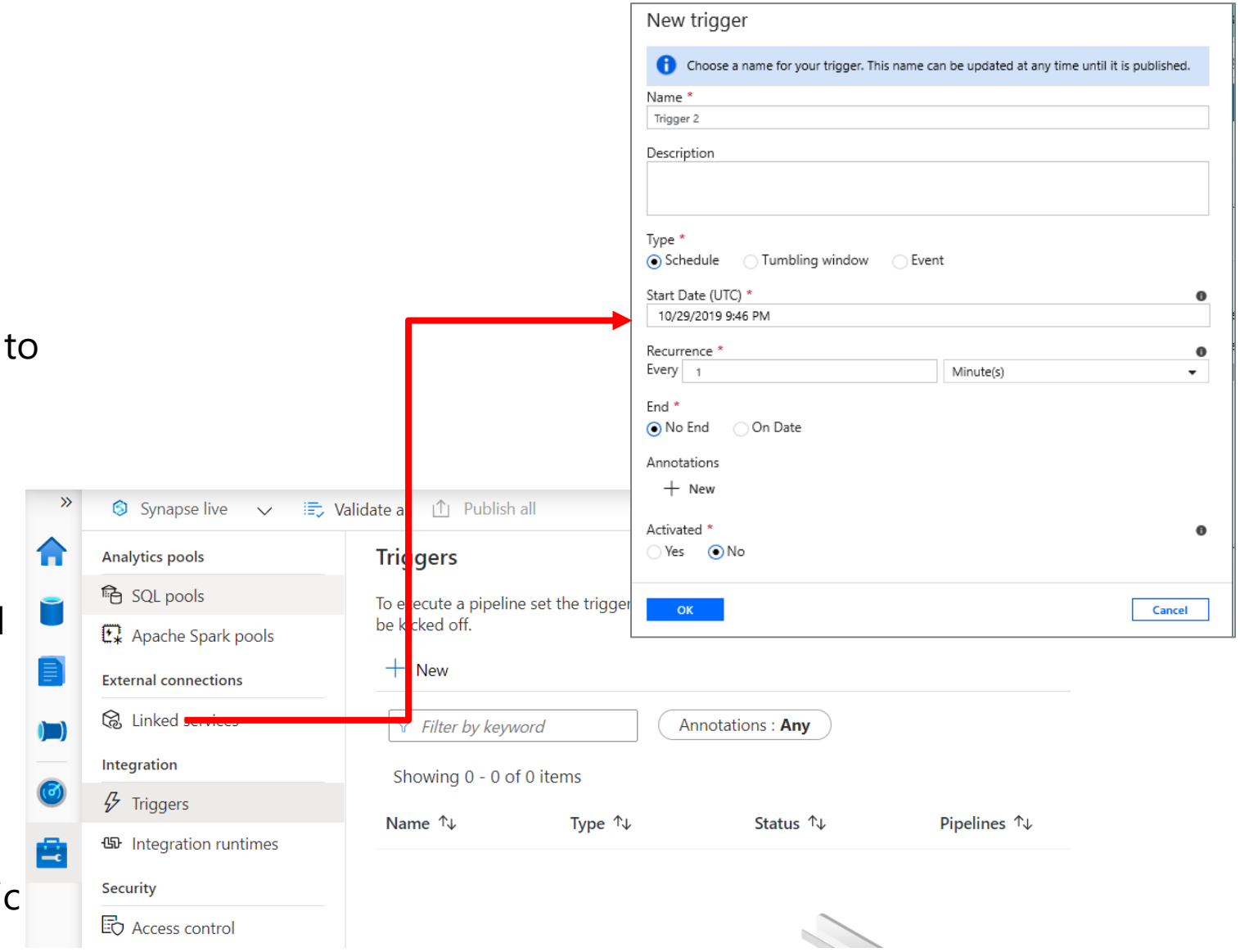
Data Integration offers 3 trigger types as –

1. Schedule – gets fired at a schedule with information of start date, recurrence, end date

2. Event – gets fired on specified Storage 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.



Manage – Integration runtimes

Overview

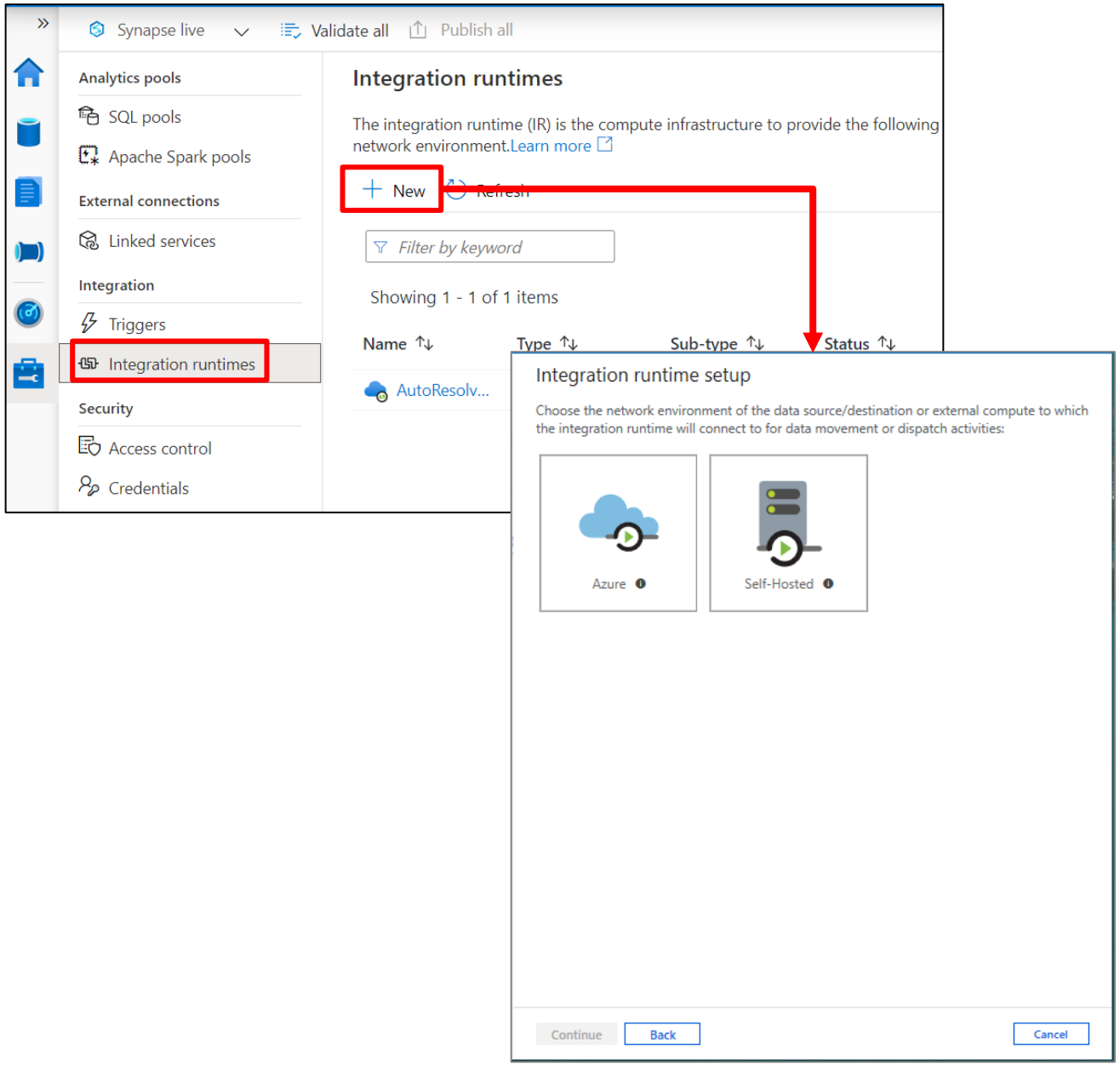
Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

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



Data Movement with Integration Runtimes

Scalable

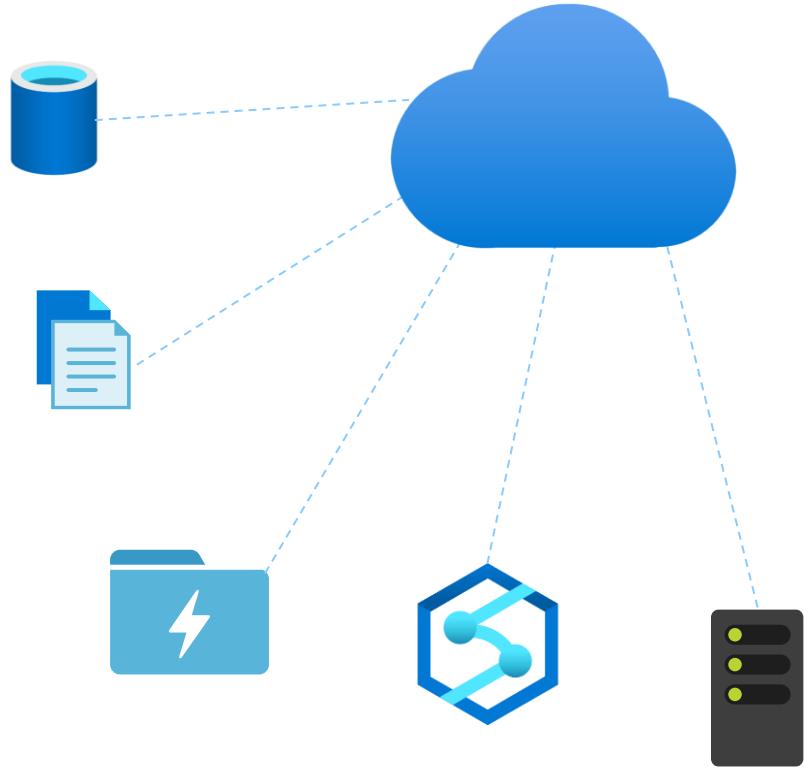
- per job elasticity
- Up to 4 GB/s

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



References

https://docs.microsoft.com/en-us/learn/modules/orchestrate-data-movement-transformation-azure-data-factory/2-understand-control-flow

https://www.techtalkcorner.com/azure-data-factory-connectors/

https://www.packtpub.com/product/azure-data-factory-cookbook/9781800565296

https://www.cathrinewilhelmsen.net/introduction-azure-data-factory/

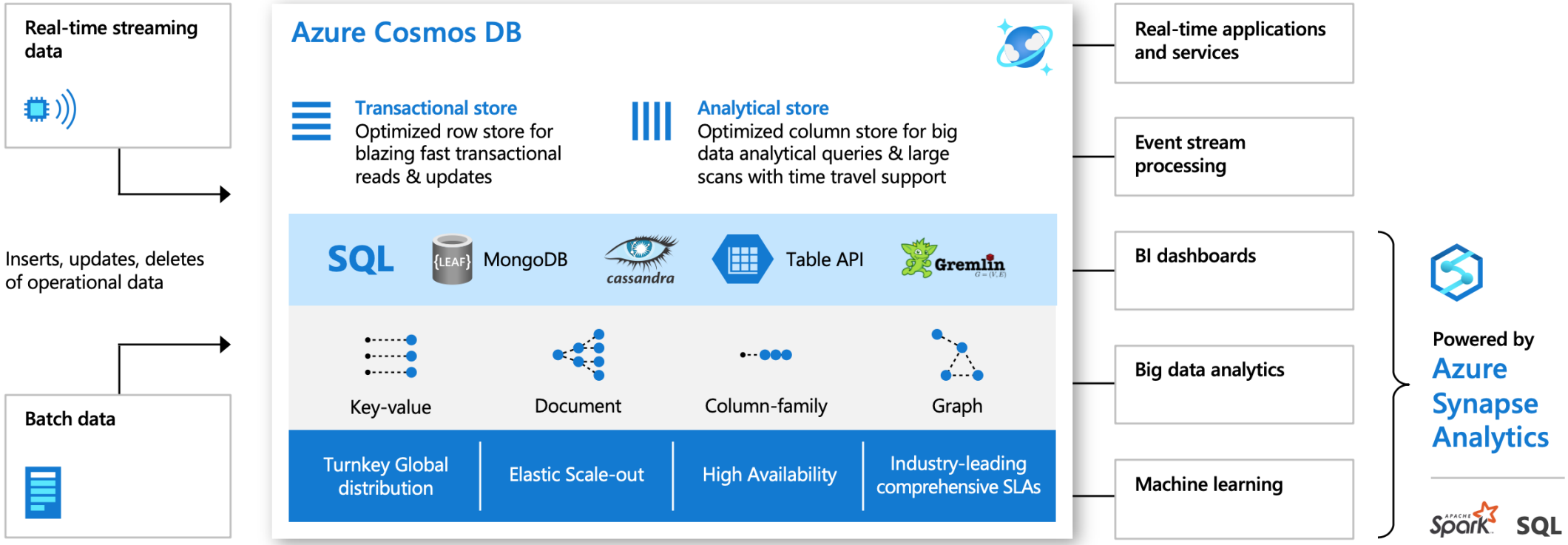
https://docs.microsoft.com/en-us/azure/data-factory/introduction

Microsoft

Azure Synapse Analytics

Synapse Link for Cosmos DB

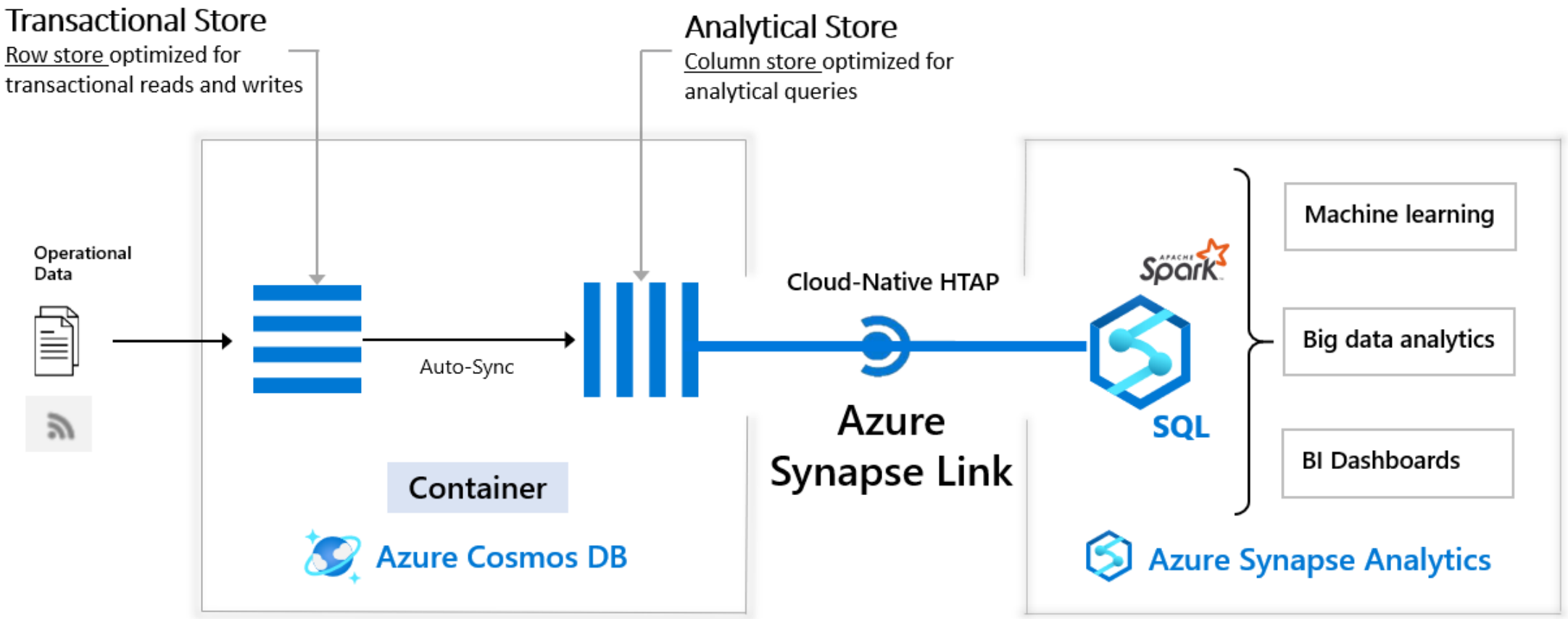
(Cosmos DB + Synapse) ❤ Analytics



Synapse Link for Cosmos DB integration

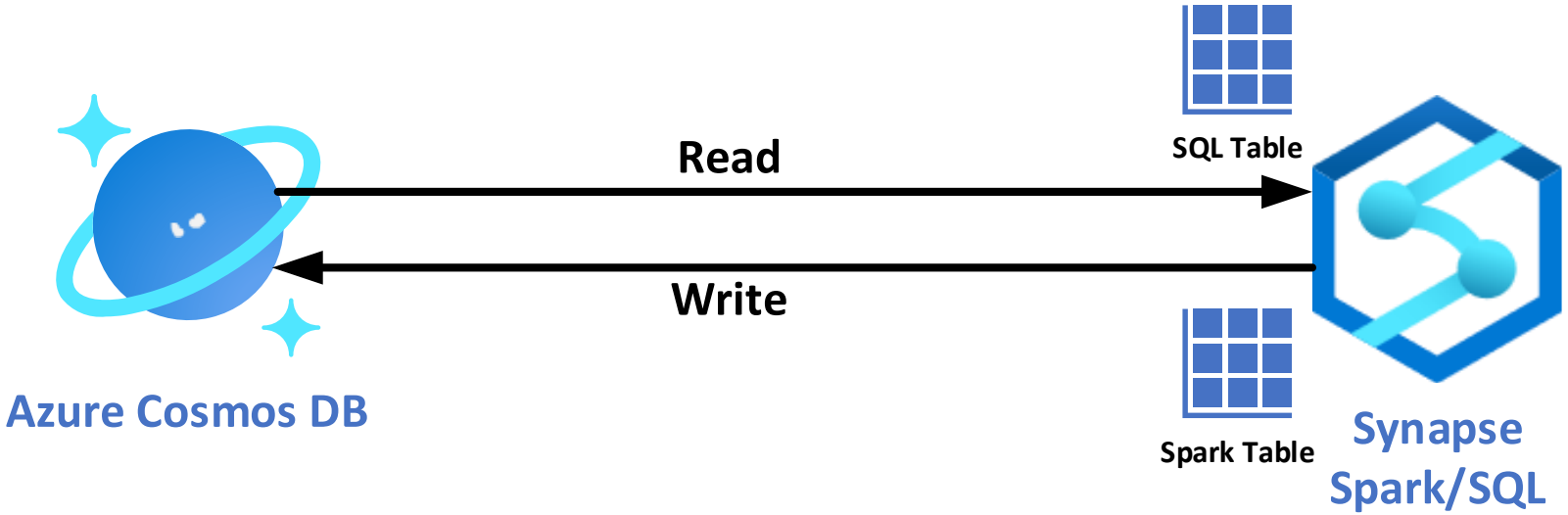
Cloud native hybrid transactional and analytical processing (HTAP) capability

In-place analytics over operational data in Azure Cosmos DB



Modern Data Warehouse Architecture

With Synapse Link for Cosmos DB



HTAP reduces architectural complexity by collapsing the stack to:

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

Benefits of running HTAP in Cosmos DB and Synapse

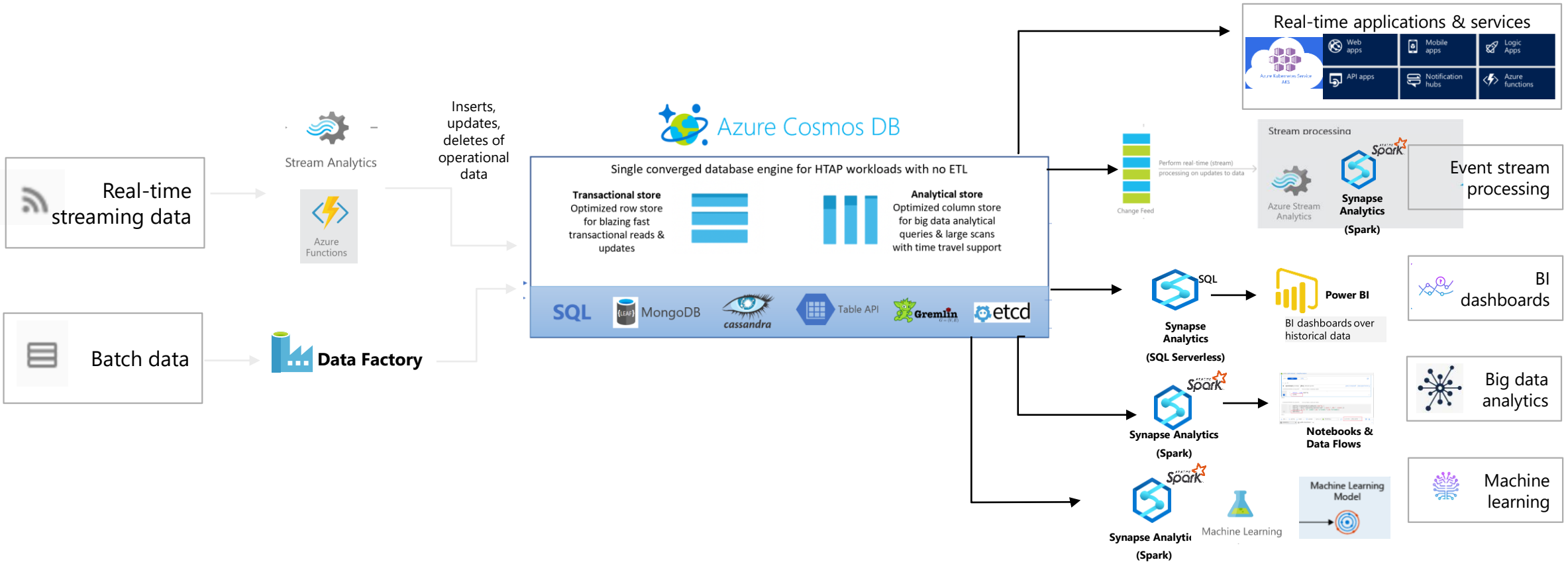
Operational Needs

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

Analytics Needs

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

High level reference architecture for HTAP workloads



References

https://docs.microsoft.com/en-us/azure/cosmos-db/synapse-link?toc=/azure/synapse-analytics/toc.json&bc=/azure/synapse-analytics/breadcrumb/toc.json

Microsoft

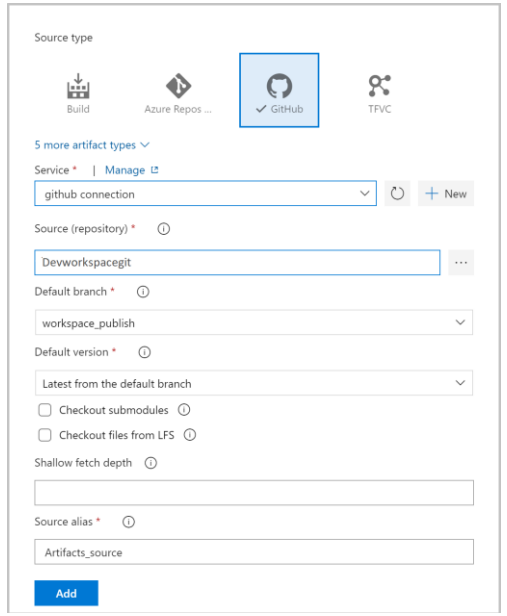
Azure Synapse Analytics

Azure DevOps & Github Integration

Continuous integration and delivery (CI/CD)

Overview

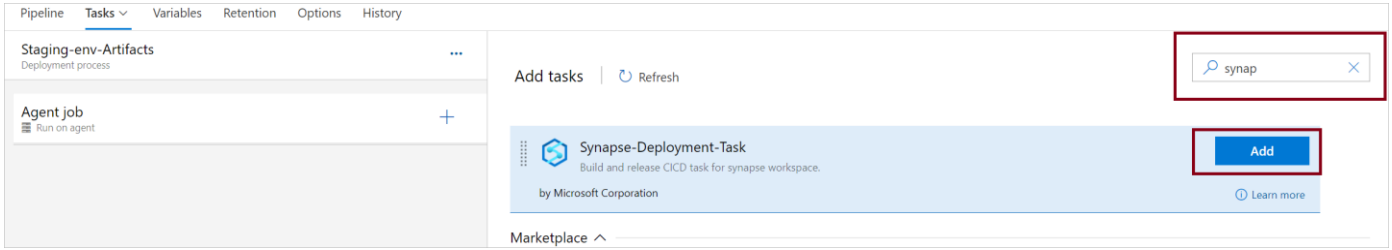
Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.

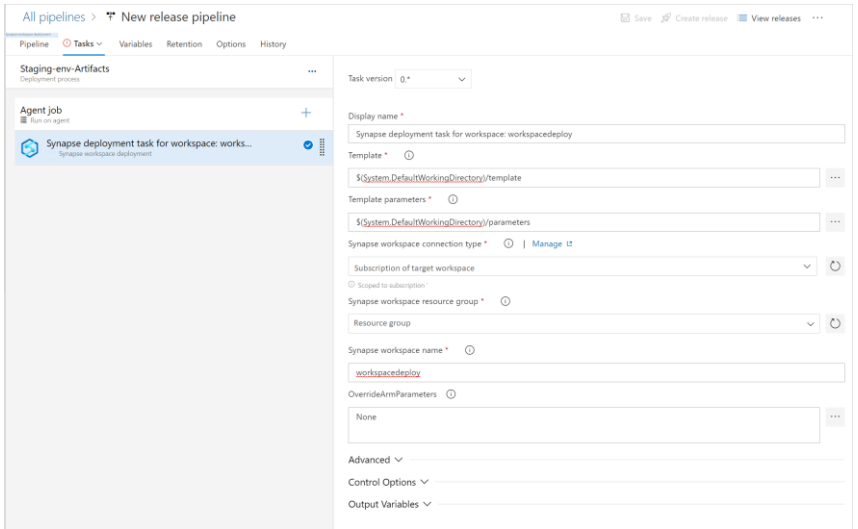


Benefits

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

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





References

https://techcommunity.microsoft.com/t5/data-architecture-blog/ci-cd-in-azure-synapse-analytics-part-1/ba-p/1964172

Continuous integration and delivery for Synapse workspace - Azure Synapse Analytics | Microsoft Docs

Microsoft

End