Microsoft

Modern Data Warehouse Using Azure Synapse Analytics

Pawel Potasinski
Senior Program Manager

@pawelpotasinski
/in/pawelpotasinski

Today's data realities

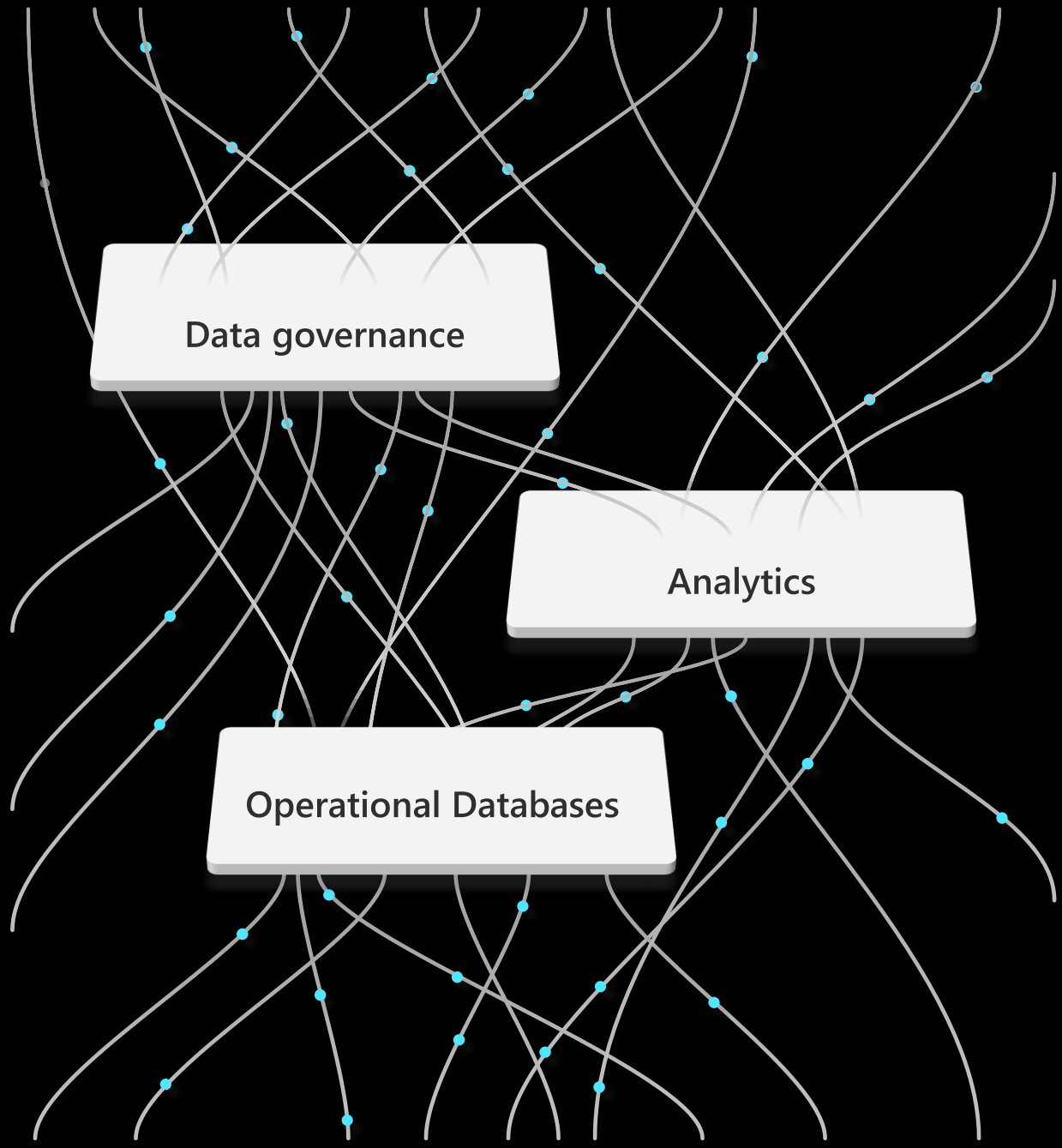
What data do I have?

Is it trustworthy?

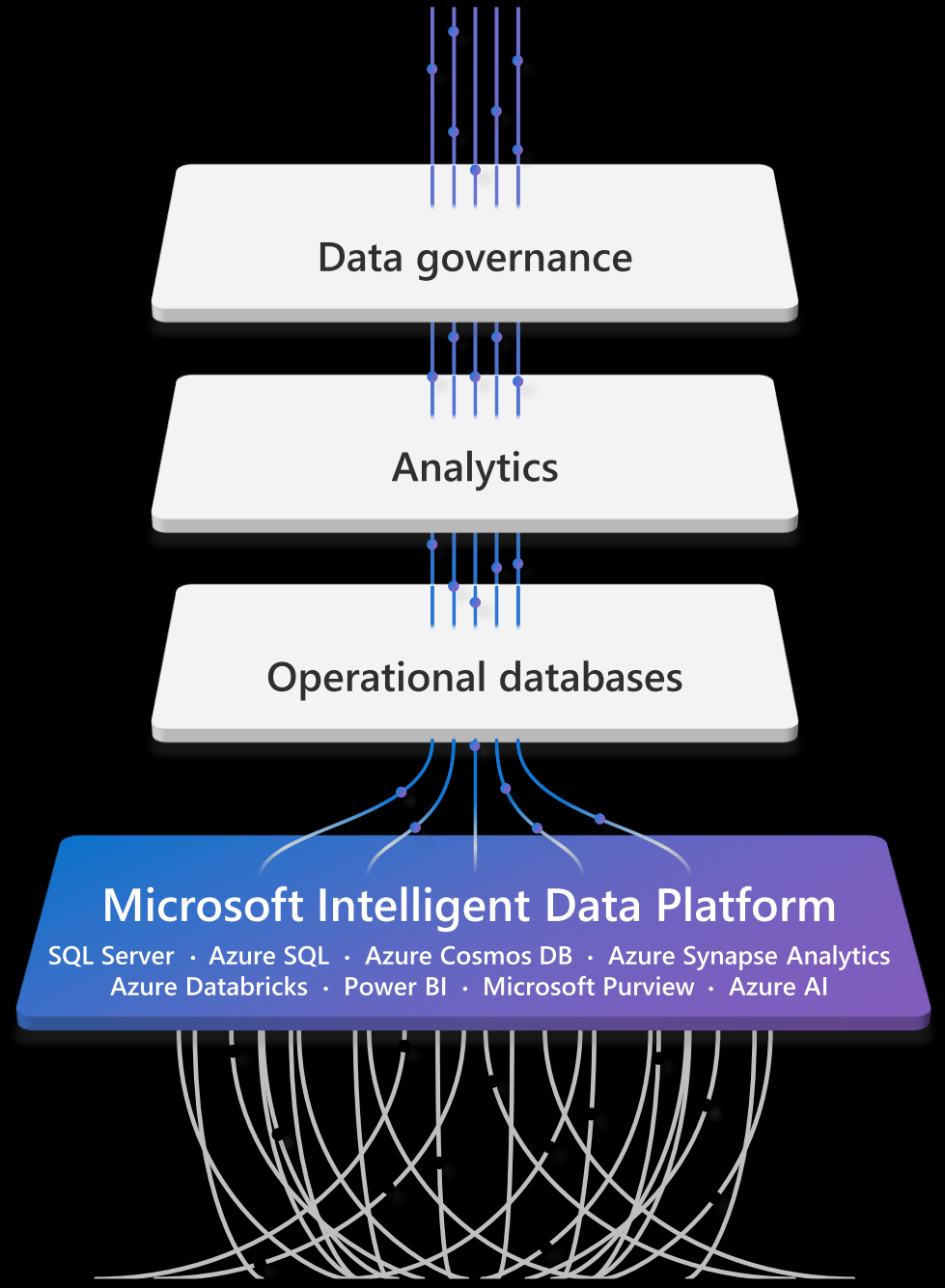
Can people access the data needed to make the right decisions?

How can I enable faster business insights?

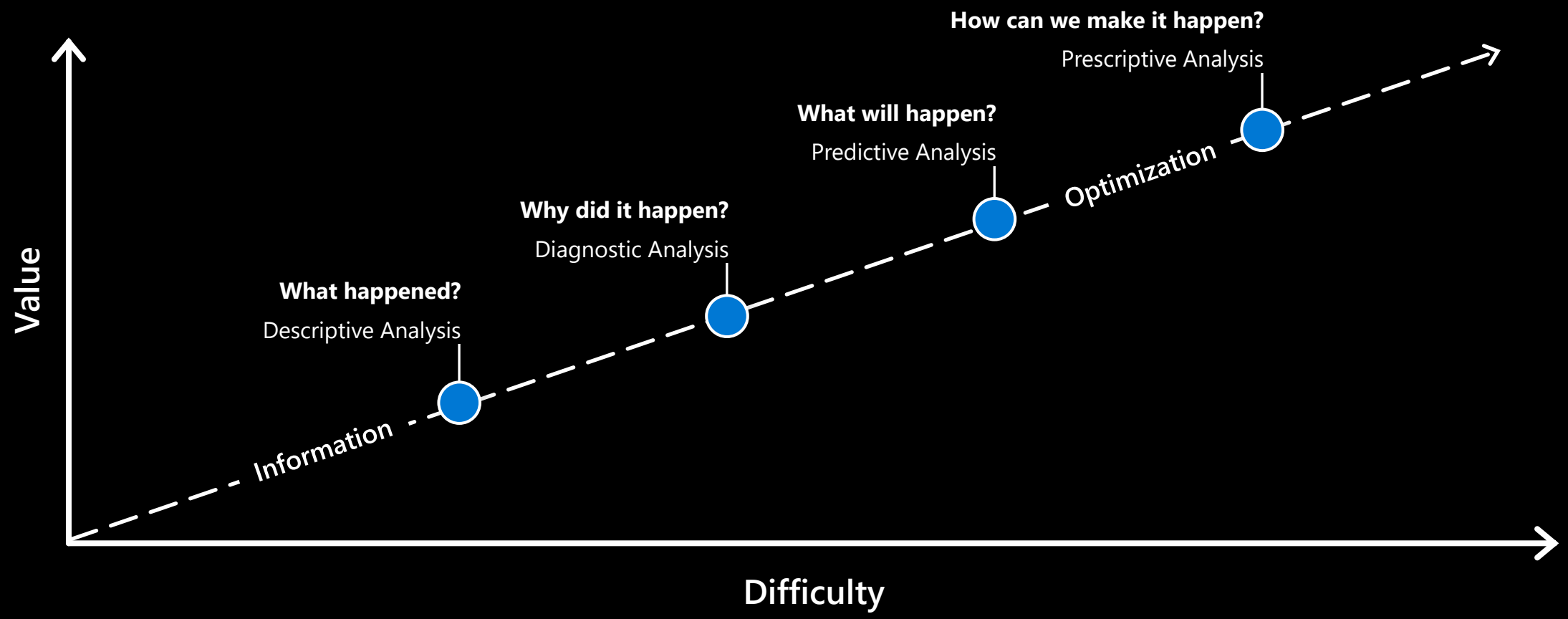
What's my compliance exposure?



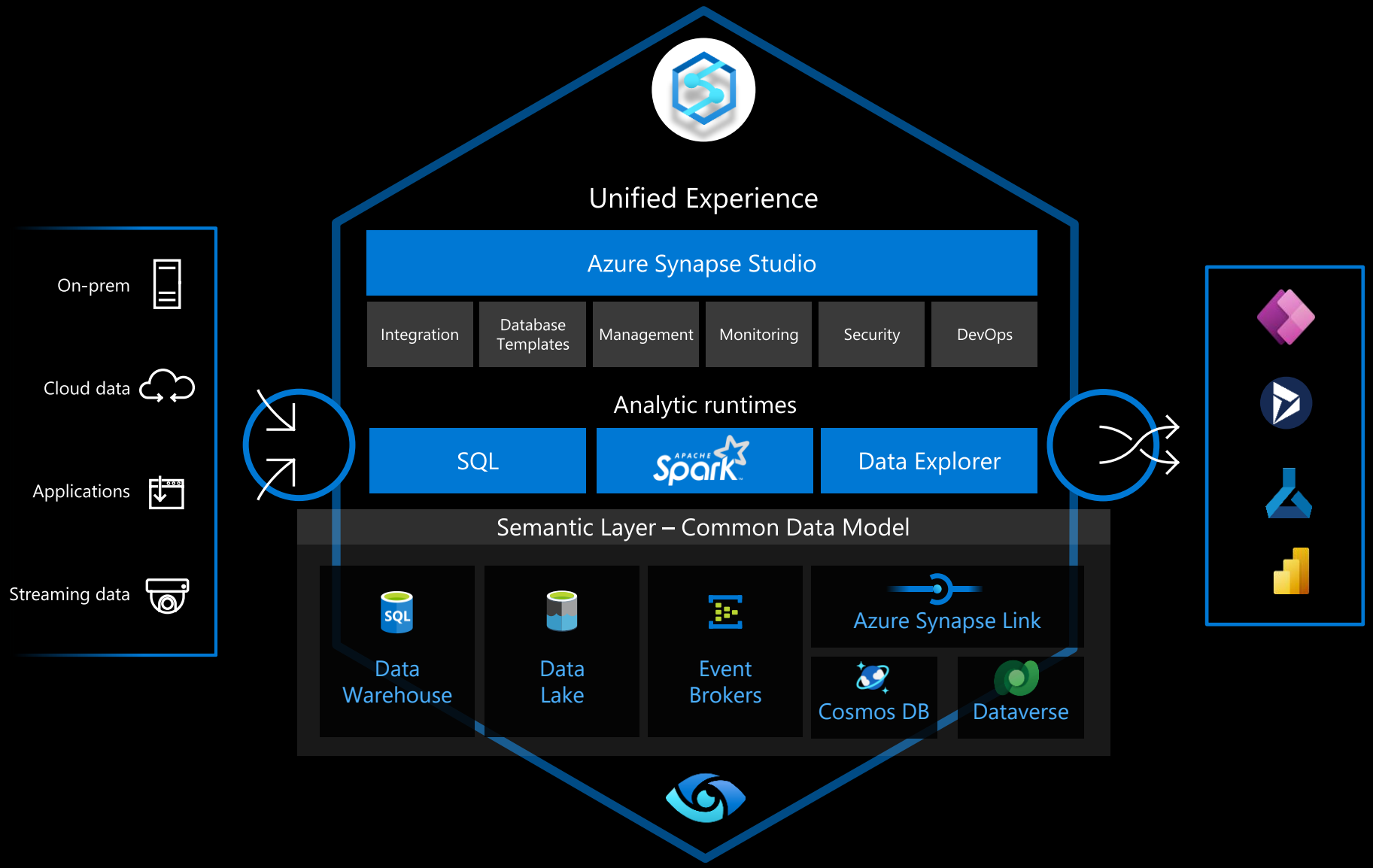
Introducing Microsoft Intelligent Data Platform



The Evolution of Analytics



Azure Synapse Analytics

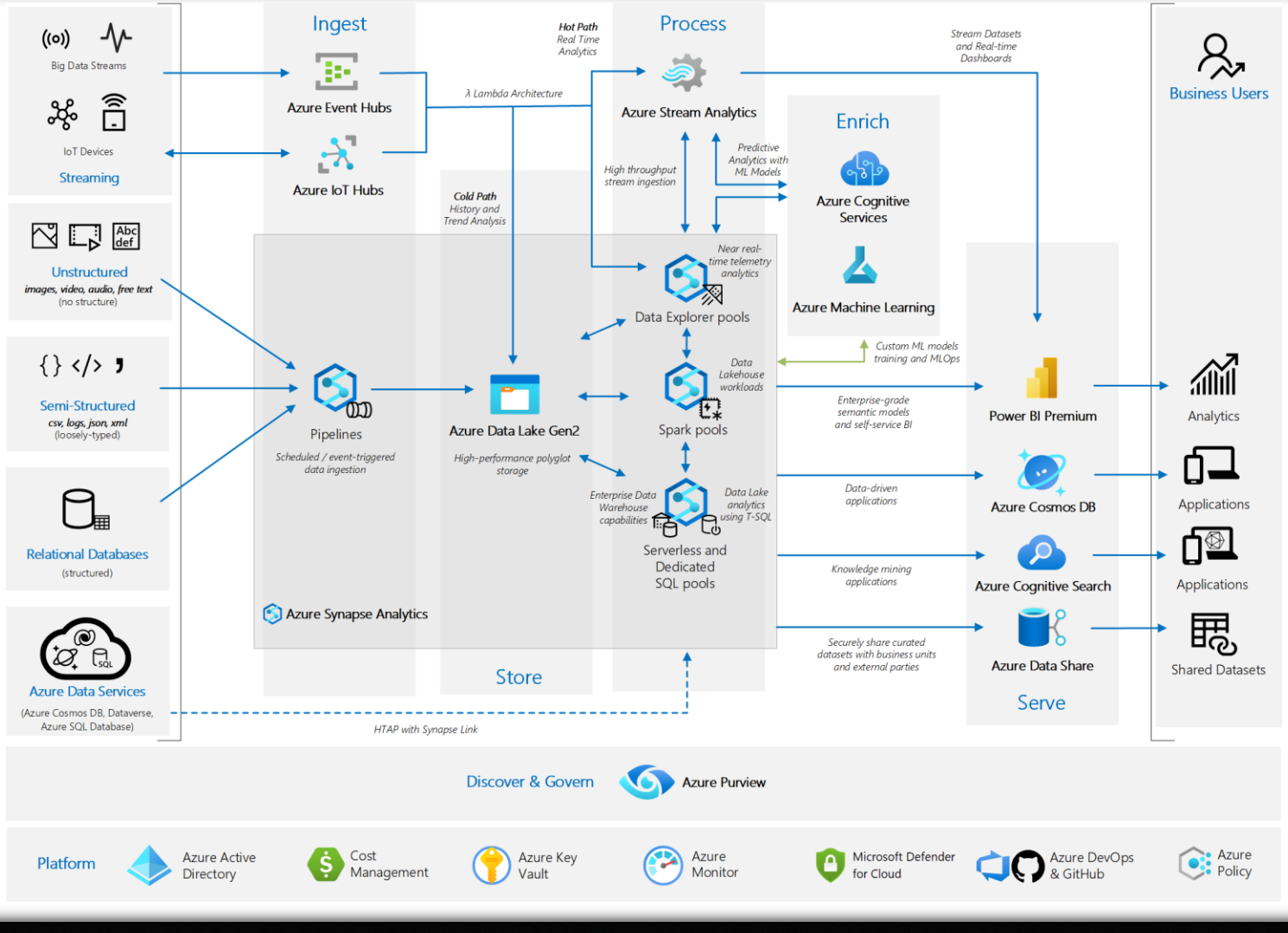


Synapse + Power BI

- Data Integration
- Data Engineering
- Data Warehousing
- Data Science
- Observational Analytics
- Business Intelligence
- Governance

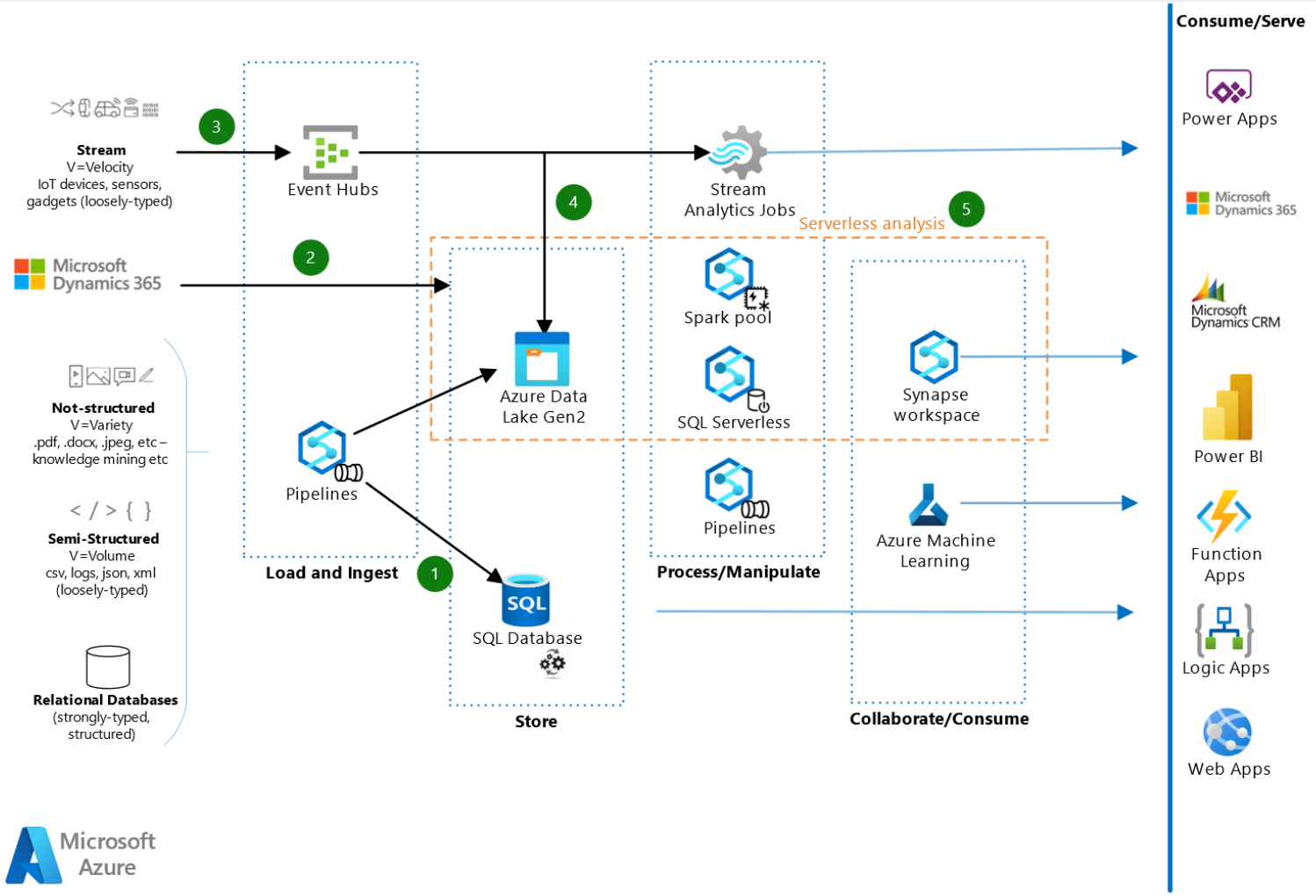
Analytics on Azure

End-to-end for Enterprise



Analytics on Azure

Small and Medium Business

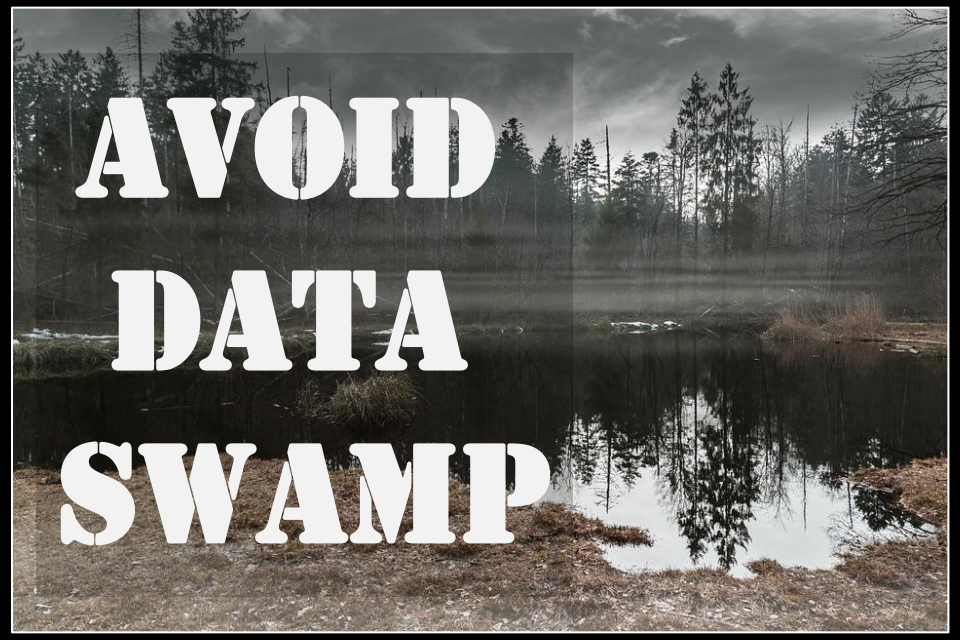


Why Data Lake?

- Storage is the **landing zone** for data in the cloud
- Same pattern for **all major clouds** (AWS, Azure, GCP)
- It's **cheap** and can be cheaper using access tiers
- It's **redundant** (HDFS nature + geo-replication features)
- It's great for **fast analytics** (load & analyze)
- It can provide **performance** (compressed data formats)
- Often offers the most efficient way to **load data into data warehouse** (Polybase in Azure Synapse)
- Data can be **easily consumed** by modern BI tools

Principles for Data Lake

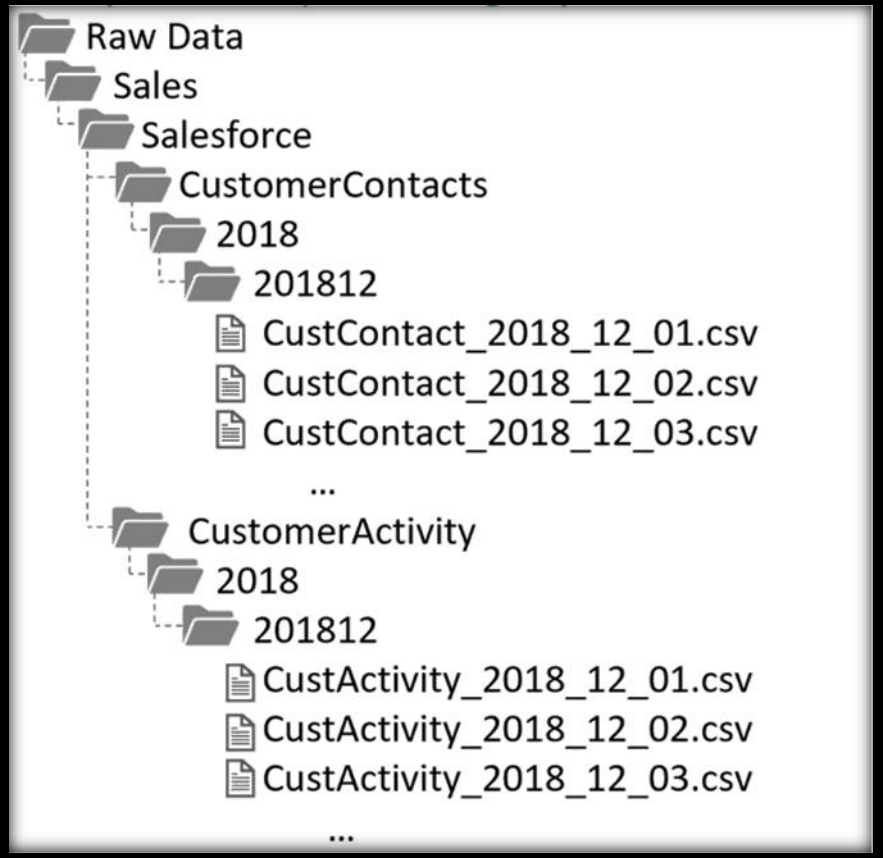
- Data Lake is NOT just a storage
- Data Lake requires careful planning
  - Zones and rules for moving data among them
  - Data ownership
  - Hierarchies
  - Naming conventions
  - File formats
  - Partitioning
  - Data access patterns
  - Metadata management
- Consistency and automation required



Data Lake Hierarchy

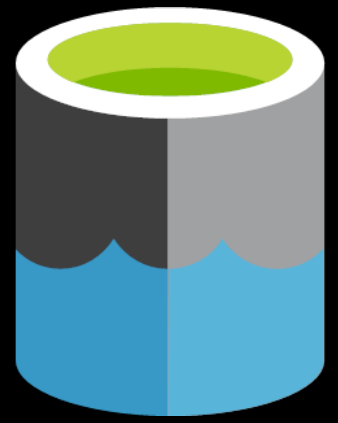
Example:

- Data Lake Zone (RAW/STAGE/…)
  - Business Area (Sales/Logistics/…)
    - Data Source (ERP/CRM/…)
      - Year (2019 / year=2019)
        - Month (01 / month=01)
          …

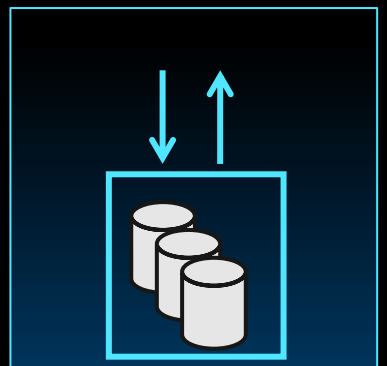


Azure Data Lake Storage Gen2

- Hierarchical namespace
- Fine-grained security via RBAC and ACLs
- Optimized for analytics and big data
- Parallelized reads and writes
- Scaled-out over multiple nodes
- Hot/cold/archive tiers
- Azure Storage APIs
- Integrated with Azure Synapse and Power BI

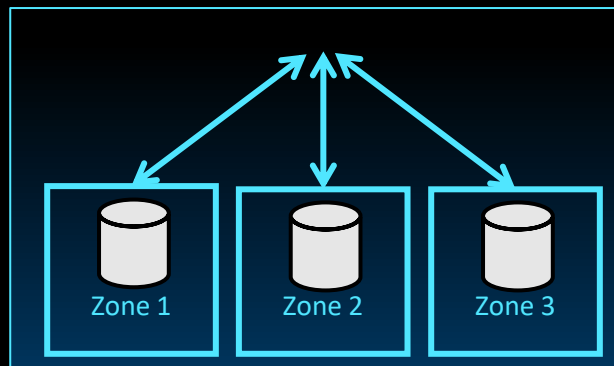


Storage Redundancy Options



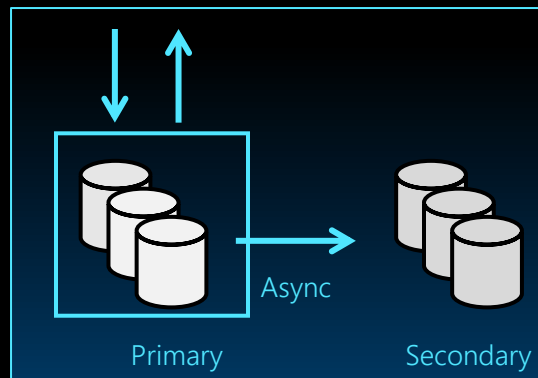
LRS
(Locally Redundant Storage)

3 replicas, 1 region



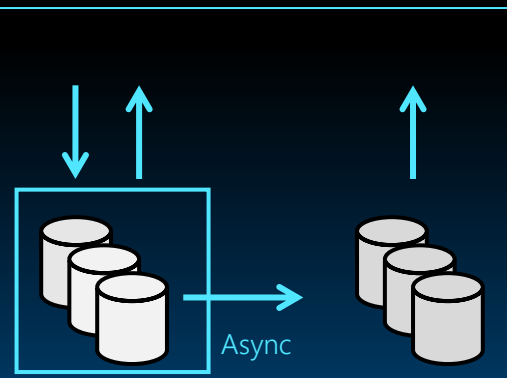
ZRS
(Zone Redundant Storage)

3 replicas **across 3 Zones**



GRS
(Geo Redundant Storage)

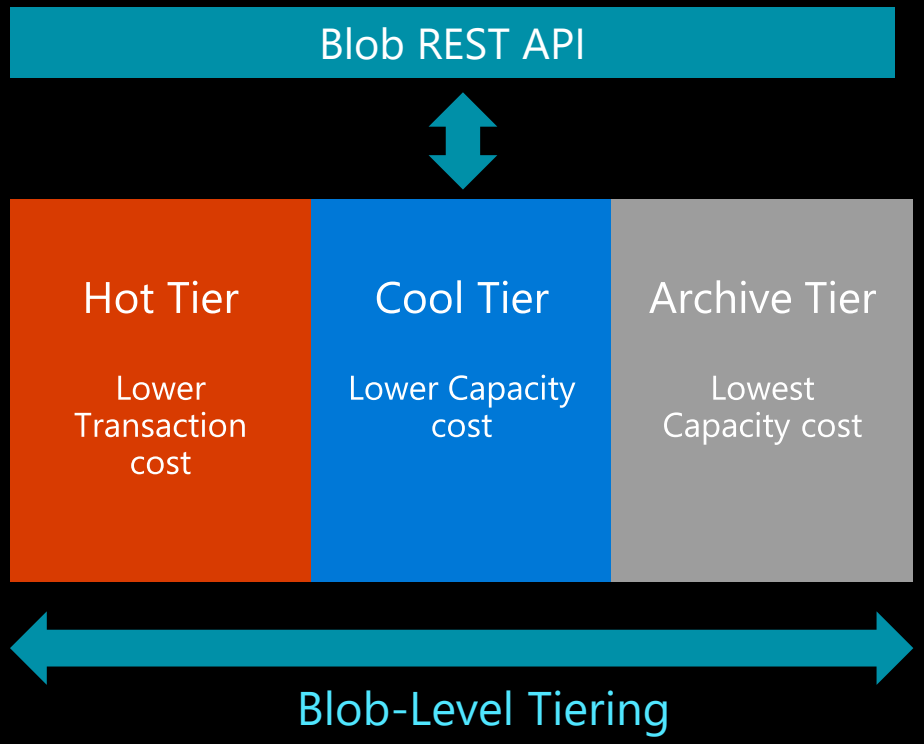
6 replicas, 2 regions (3/region)



RA-GRS
(Read Access – Geo Redundant Storage)

GRS + Read access to secondary

Blob Level Tiering and Lifecycle



Data Formats

Parquet

- Open Source (https://parquet.apache.org/)
- Column-oriented
- Data compression
- Metadata

Delta

- Open Source (https://delta.io/)
- ACID transactions
- Scalable metadata handling
- Schema enforcement and evolution
- Upserts and deletes
- Time Travel
- Audit History

CSV

- Flat files column and row separators
- Readable by humans
- Optional column headers
- Can be compressed (ZIP)

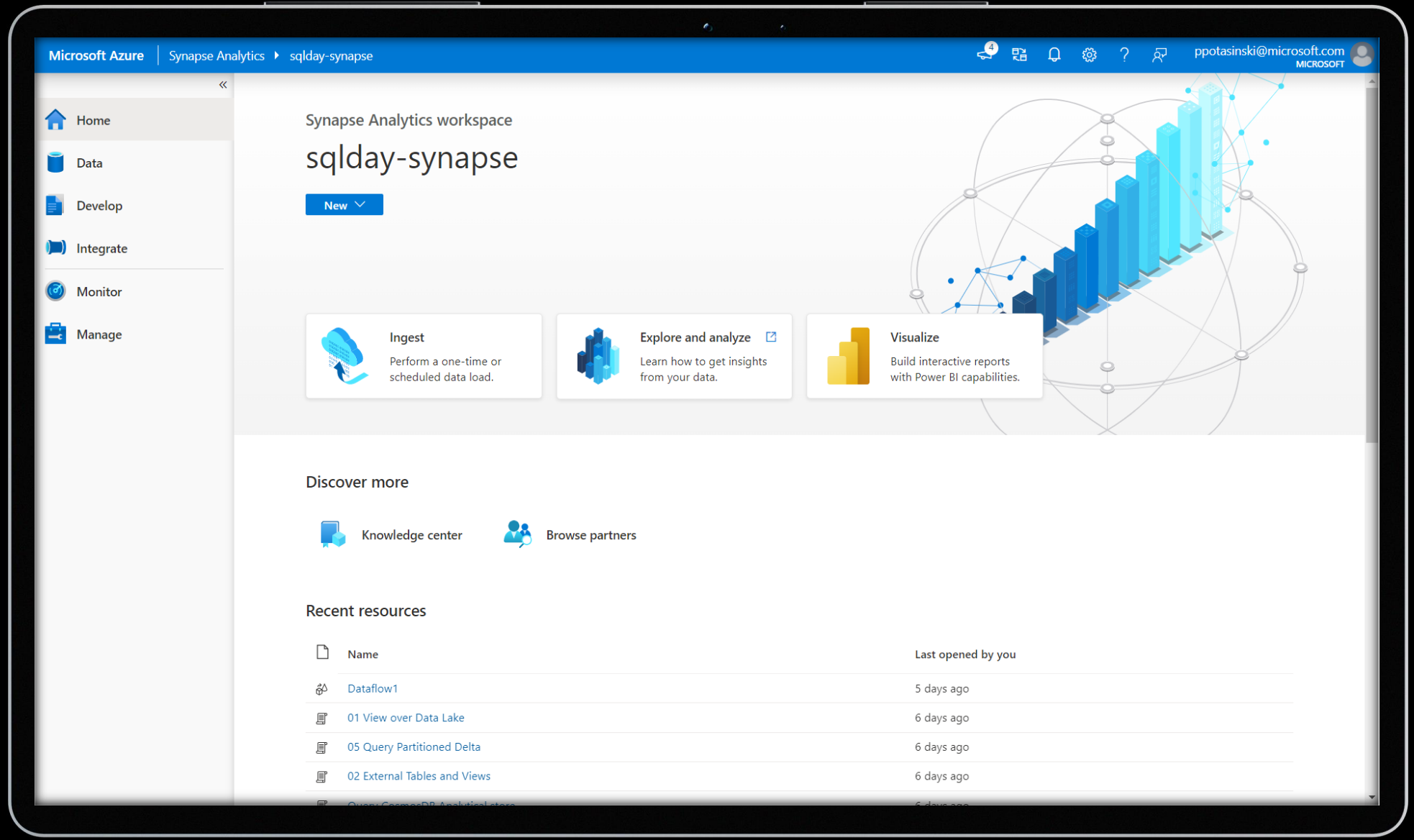
JSON

- JavaScript Object Notation
- Popular for storing and transporting data
- Readable by humans
- Contains arrays, objects, and key-value pairs

Microsoft

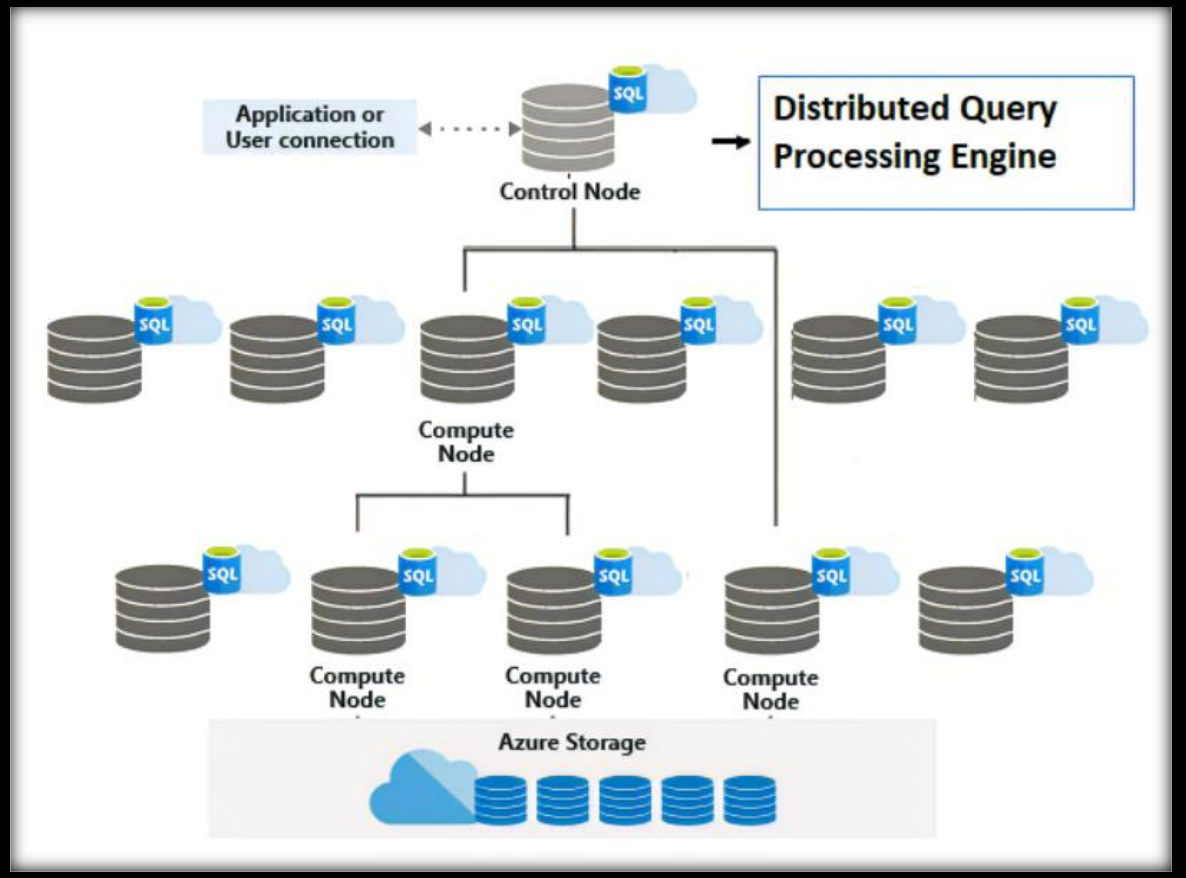
Azure Synapse Analytics

Synapse Studio



Serverless SQL Pool

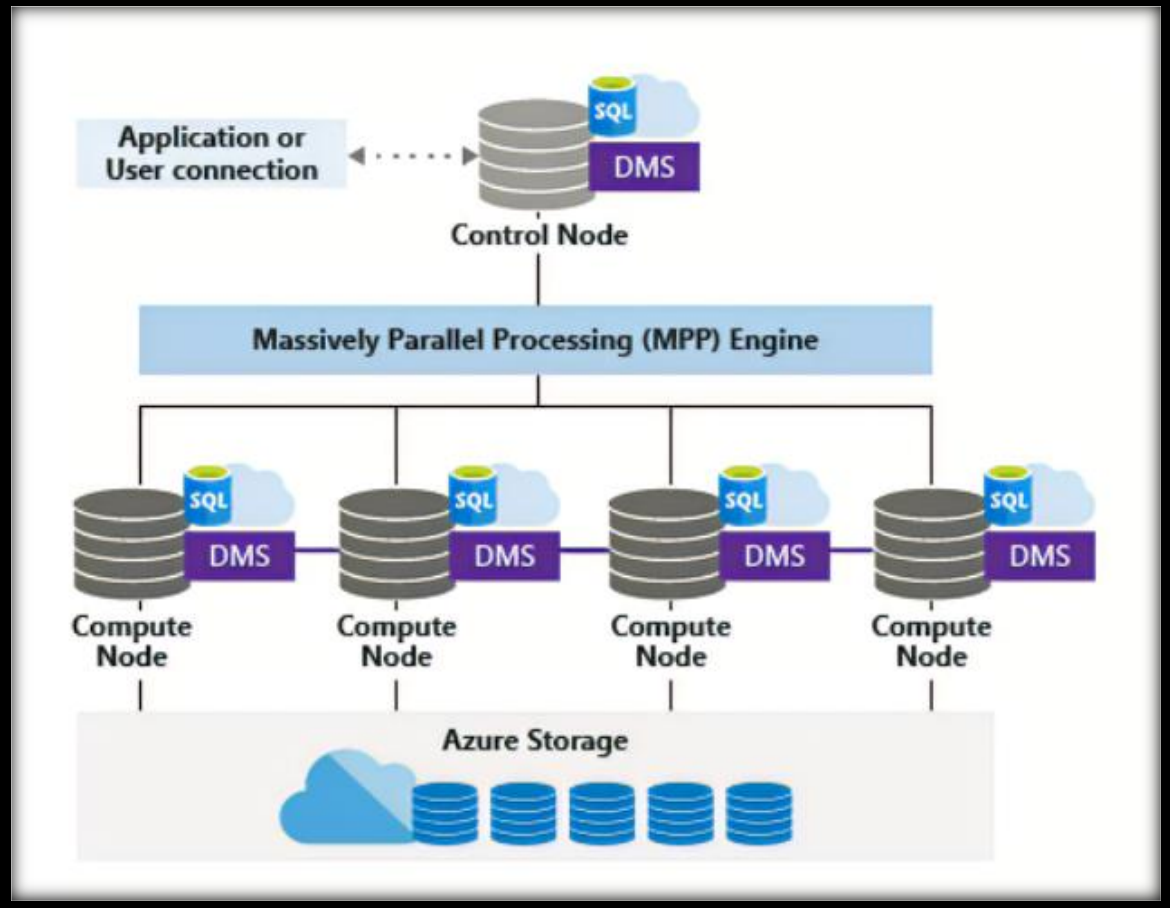
- Use T-SQL language
- Supports any tool or library that uses T-SQL to query data
- Auto Scale & Manage
- Pay-per-use model (data processed)
- Easy to use
- Automatic schema inferene
- Shared metadata with Spark pool
- Querying multiple storages (Data Lake, CosmosDB, Dataverse)



[POLARIS: The Distributed SQL Engine in Azure Synapse]

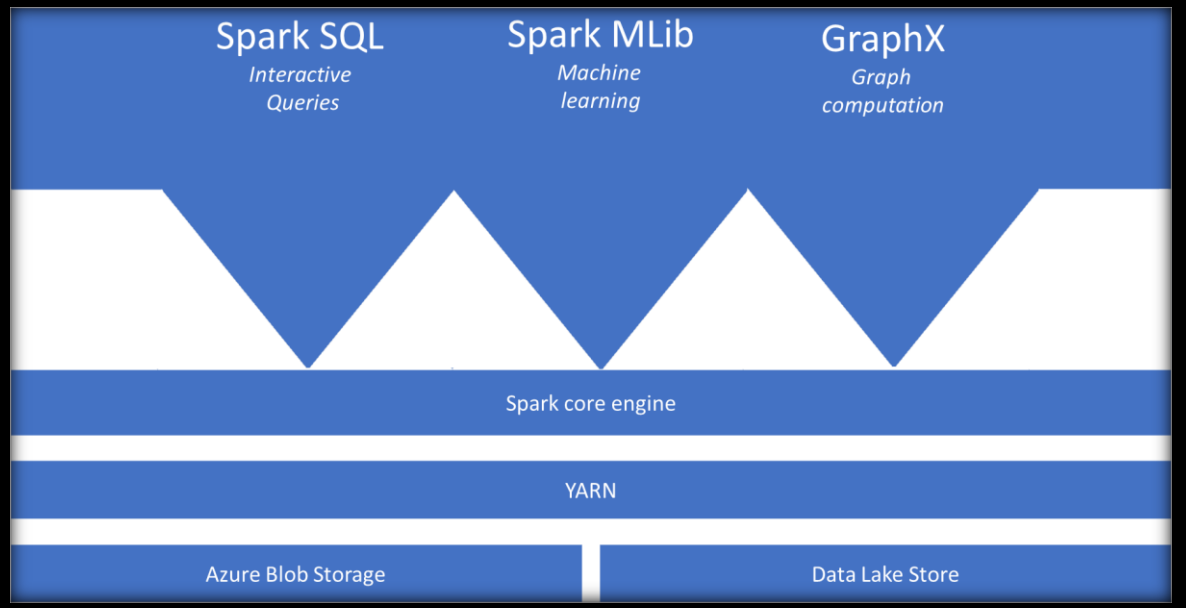
Dedicated SQL Pool

- Massively Parallel Processing (MPP)
- Shared nothing architecture
- One Control Node
- Multiple Compute Nodes
  - Always 60 distributions of data
  - Can scale-out
- Data Movement Service (DMS)
- SQL Data Warehouse Gen2
  - Optimized for performance
- Can pause and resume
  - Cost optimization
  - Pause/resume are offline operations



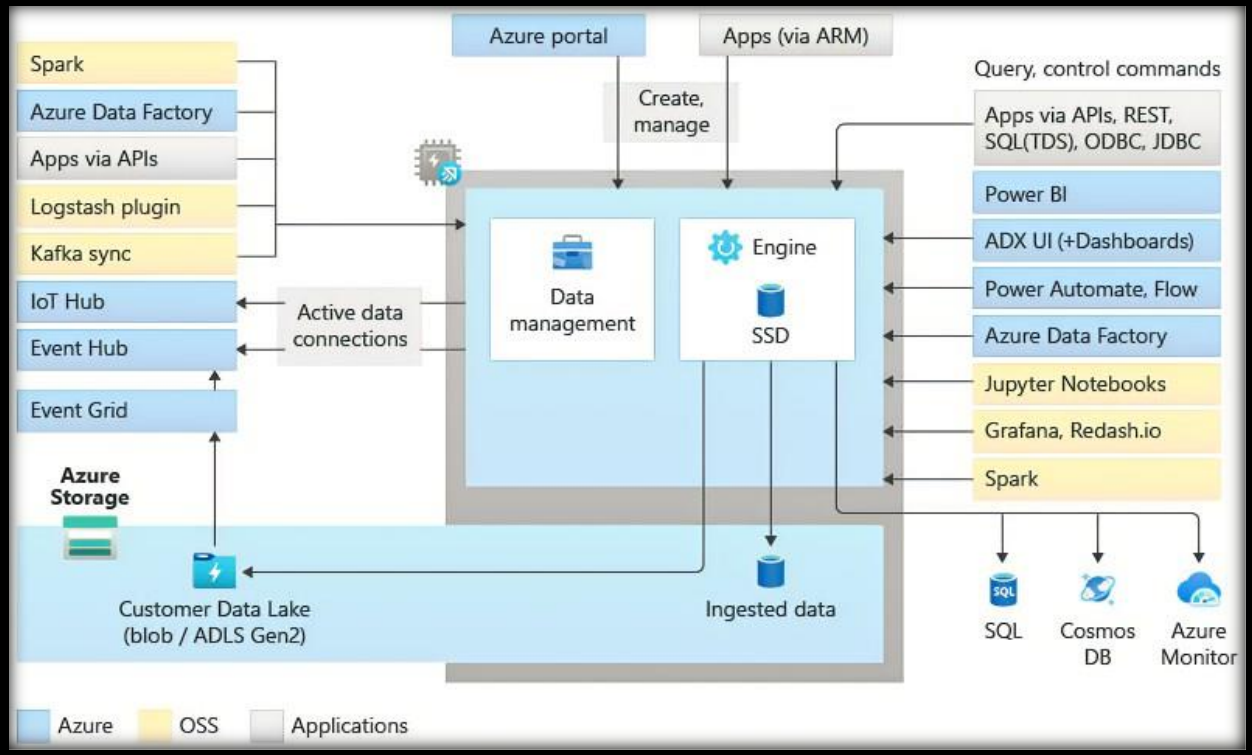
Spark Pool

- Serverless Apache Spark
- Distributed in-memory cluster computing
- Supports multiple languages (Python, Scala, SQL, .NET)
- Notebooks for interactive work
- Can handle different use cases (data explortion, data enigneering, machine learning, streaming)
- Newest version supported – 3.2



Data Explorer Pool (PREVIEW)

- Interactive query experience over log and telemetry data
- Runs on clusters with auto-scaling
- Automatic indexing, compressing, caching and serving distributed queries
- Kusto Query Language (KQL)
- Easy ingestion (Event Hubs, Kafka, Data Lake)
- No index maintenance

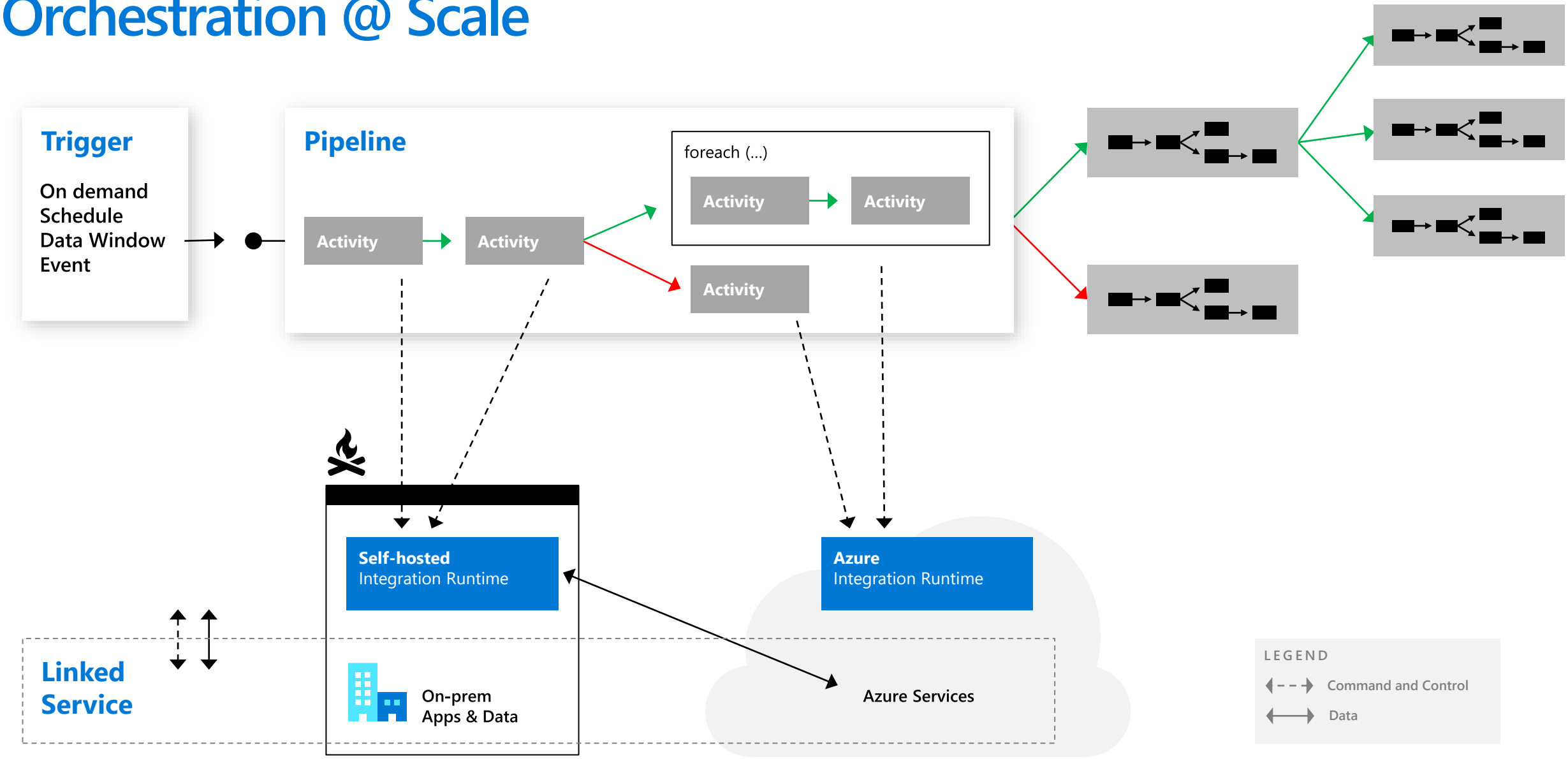


Microsoft

Azure Synapse Analytics

Data Integration

Orchestration @ Scale



Data Movement

Scalable

per job elasticity

Up to 4 GB/s

Simple

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

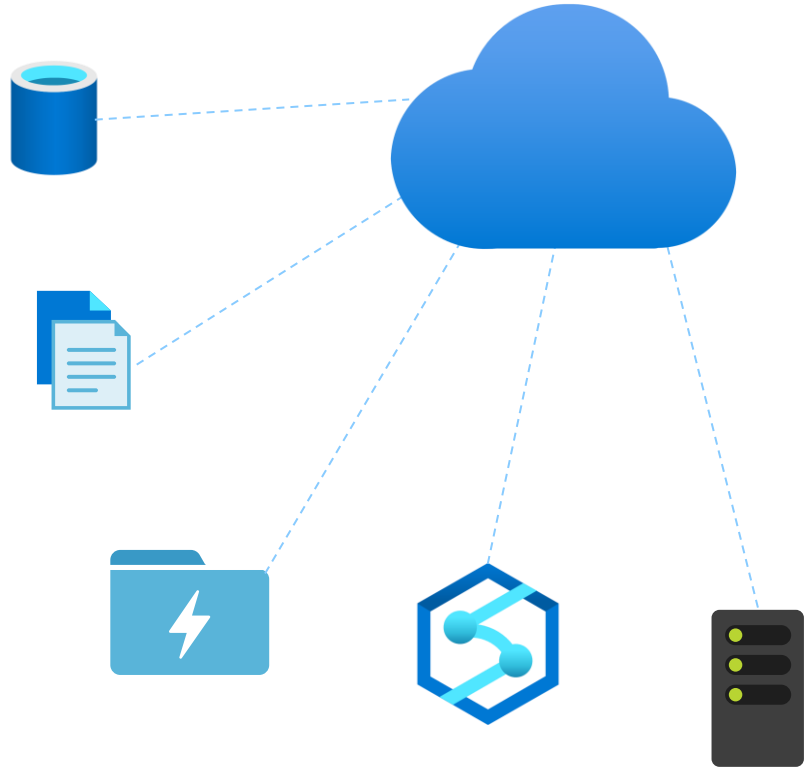
Serverless, no infrastructure to manage

Access all your data

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

Data Movement as a Service: 25 points of presence worldwide

Self-hostable Integration Runtime for hybrid movement



Pipelines

Overview

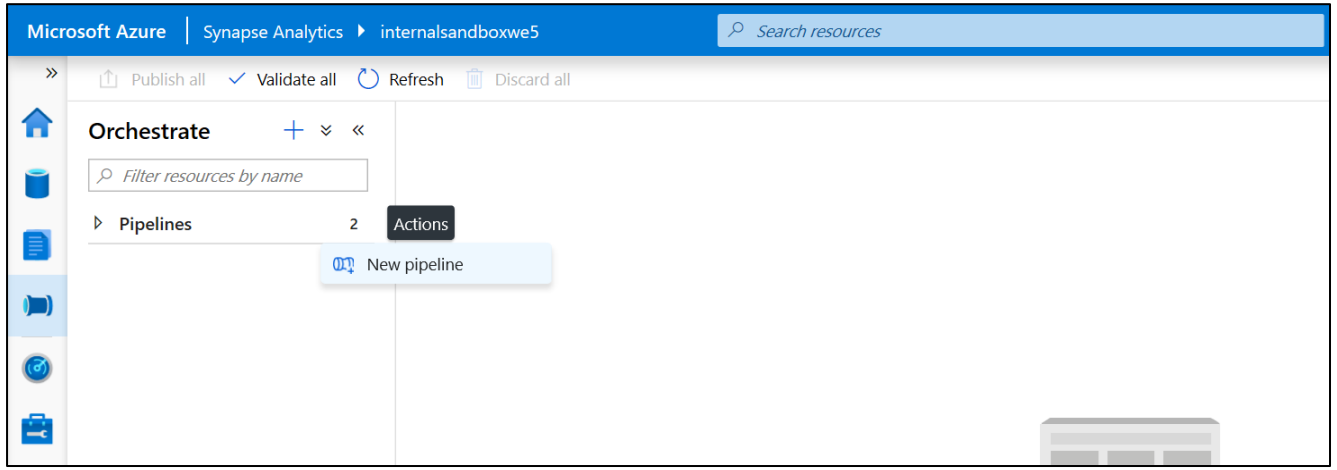
It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

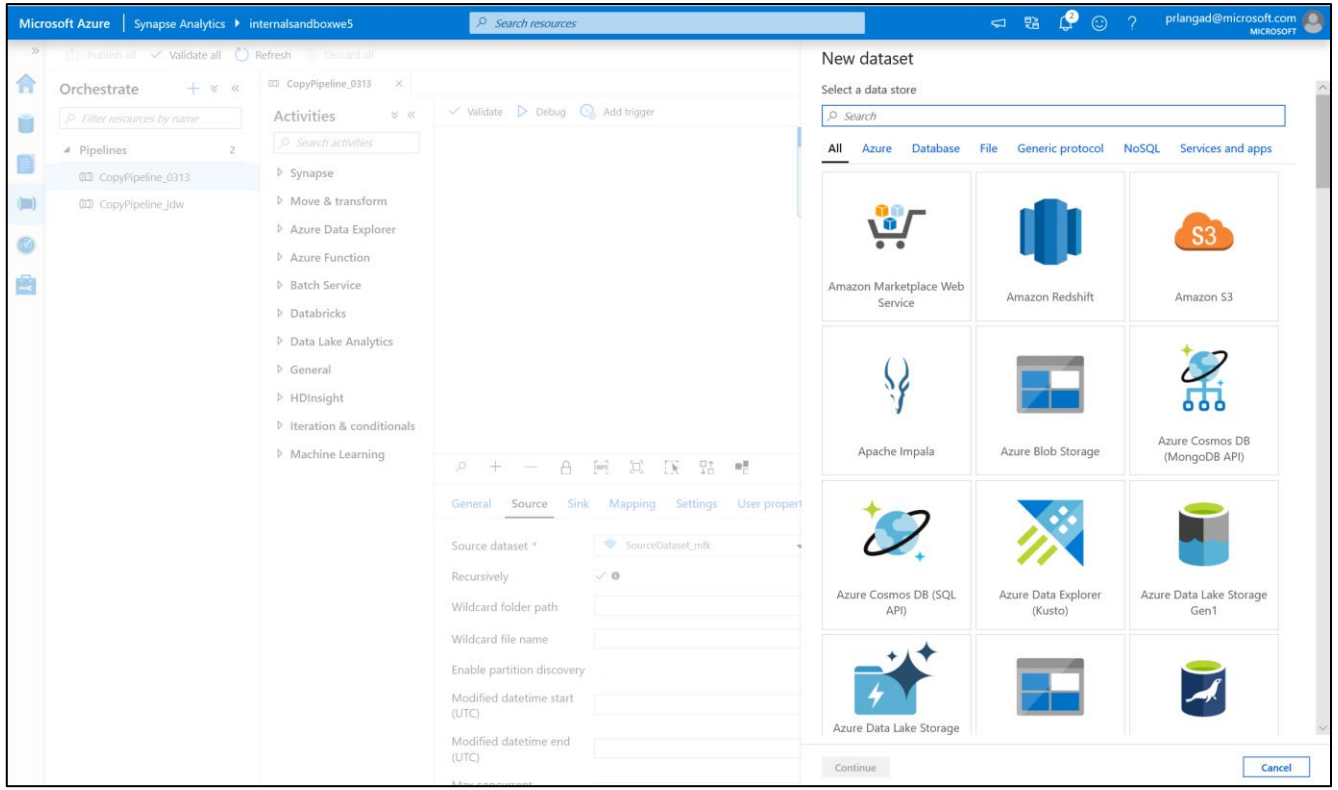
Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

Graphical development experience





Data Flows

Low code data transformation at scale

Visually designed

Executed in pipelines

Powered by Apache Spark clusters

Interactive debugging with data preview

Built-in error handling

Partitioning scheme of the Spark cluster

Triggers

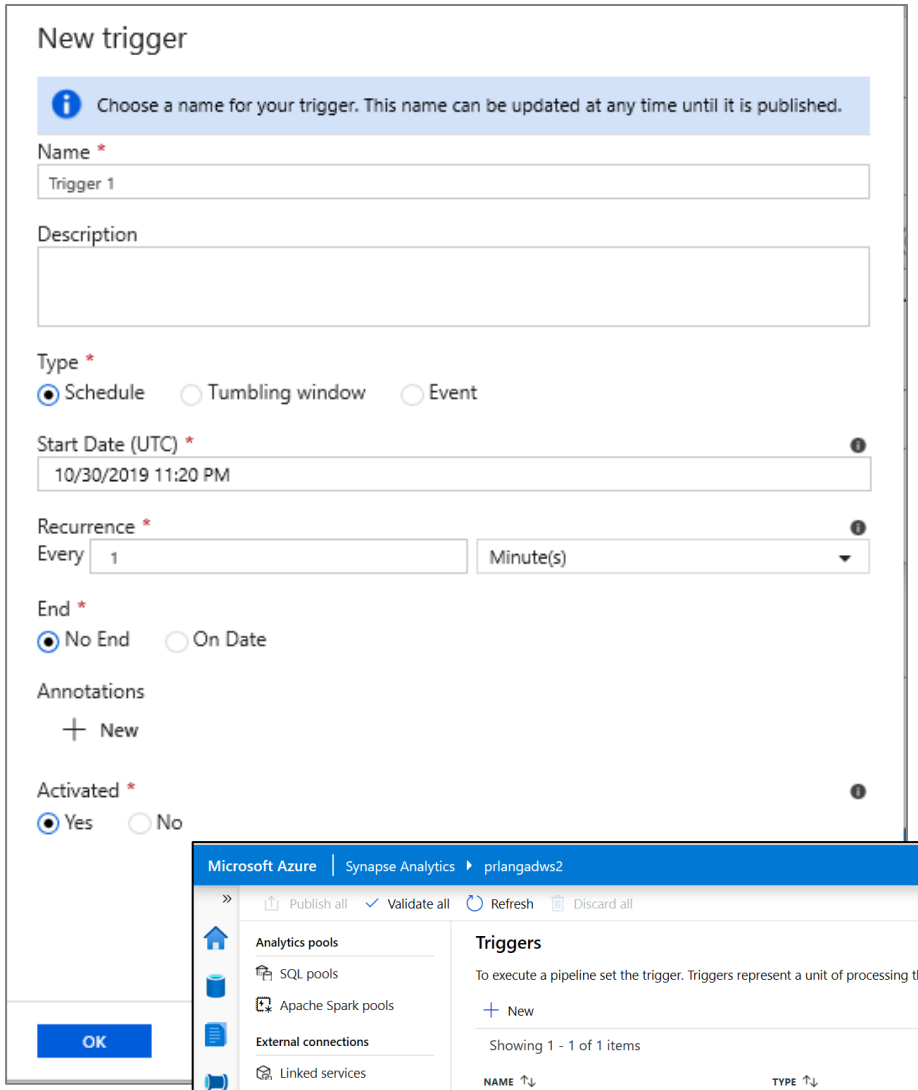
Overview

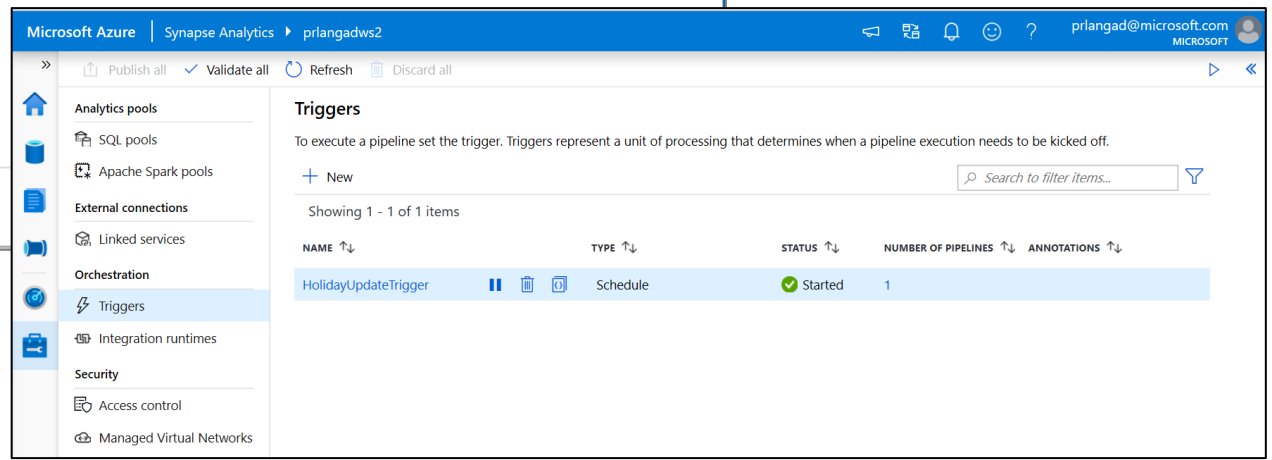
Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.

Data Integration offers 3 trigger types as –

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

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

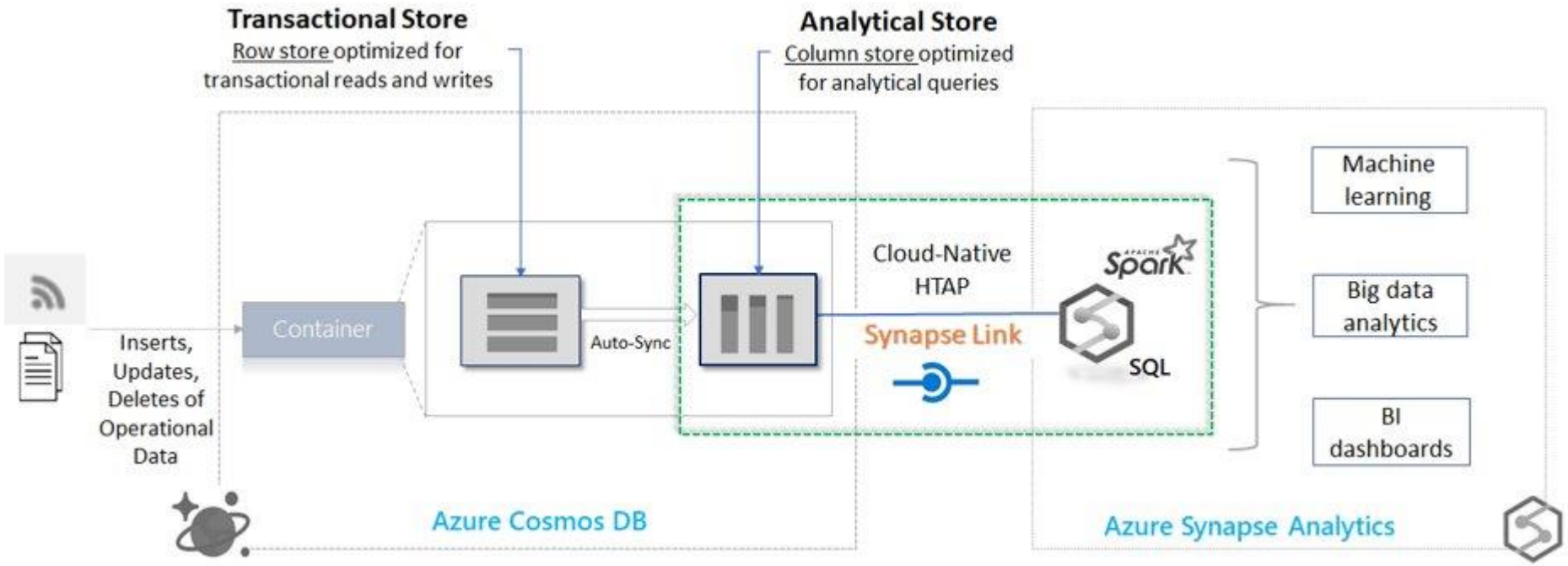




Synapse Link for Cosmos DB integration

Cloud native hybrid transactional and analytical processing (HTAP) capability

In-place analytics over operational data in Azure Cosmos DB

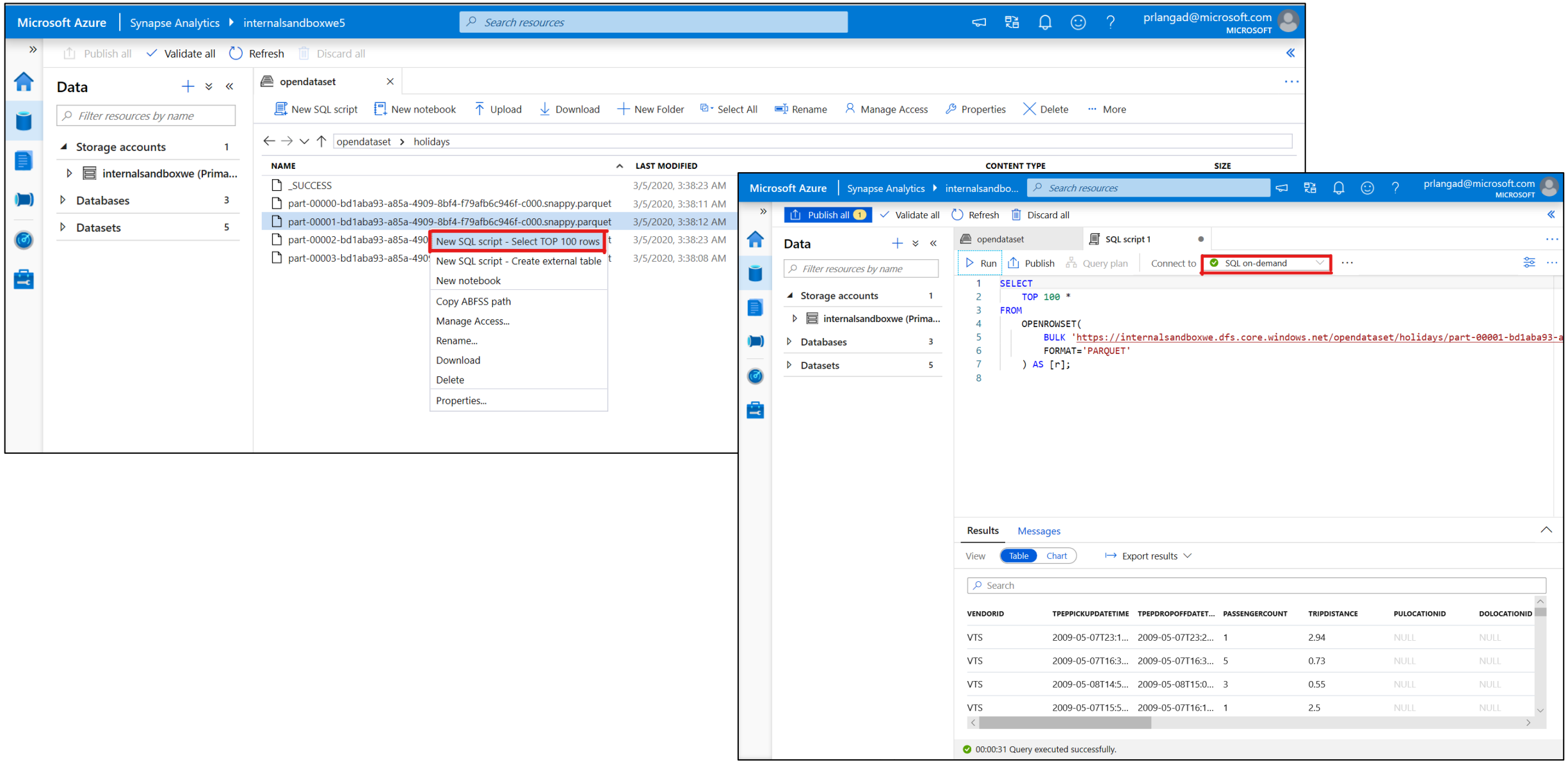


Microsoft

Azure Synapse Analytics

Synapse serverless SQL pool

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/csv/taxi/*.parquet',
        FORMAT = 'PARQUET') AS nyc
```

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Automatic schema inference

Overview

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

Benefits

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

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

Perfect solution for data exploration where schema is unknown.

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

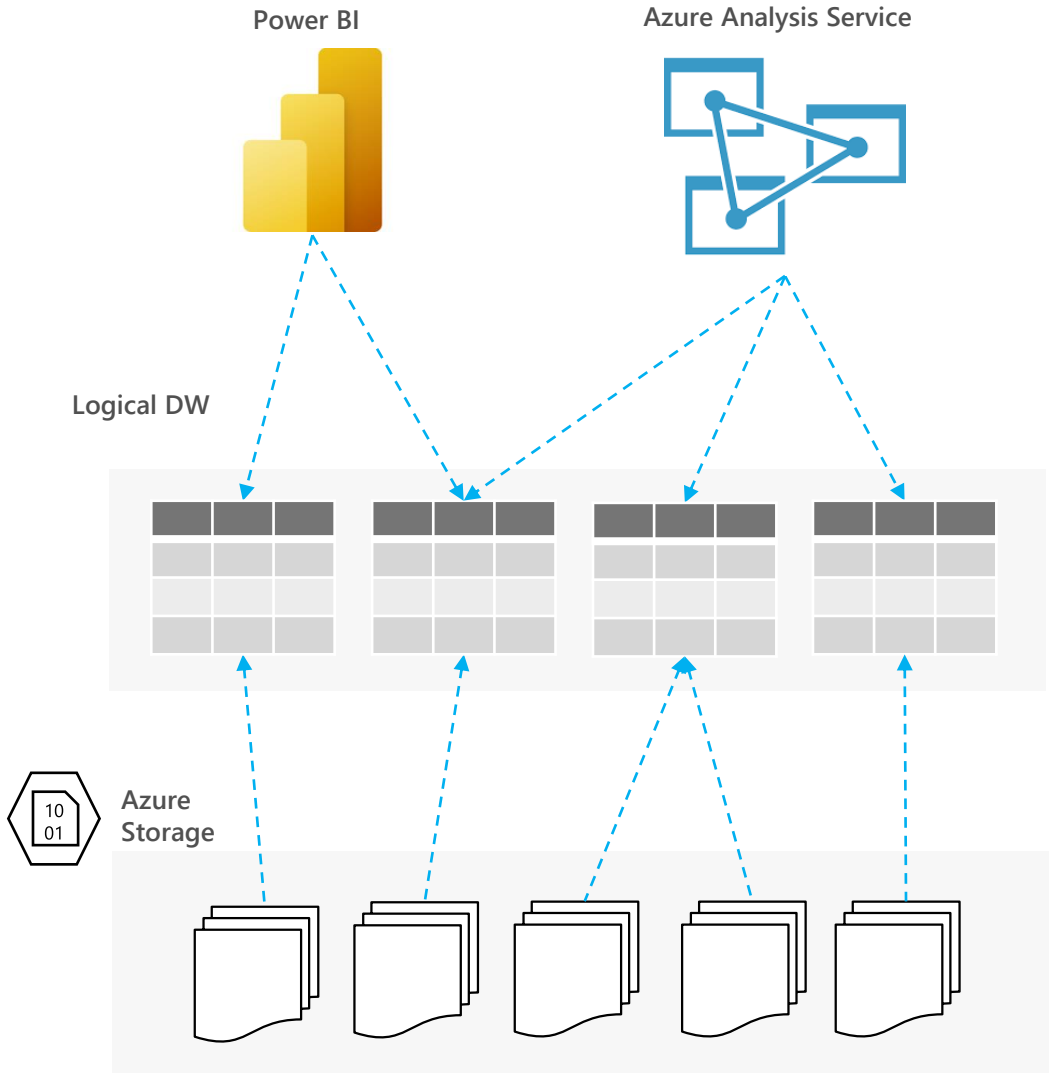
Synapse serverless SQL pool as a logical data warehouse

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

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

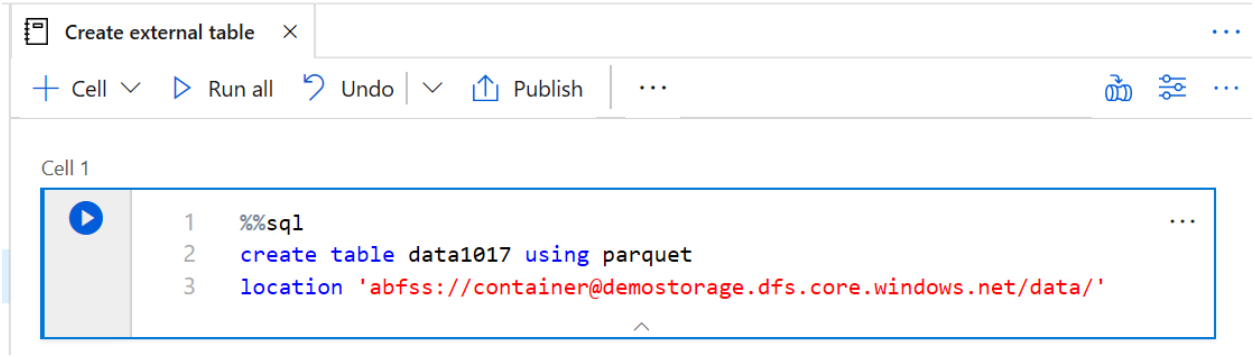
Tables designed using Spark languages are immediately available in serverless SQL pool.

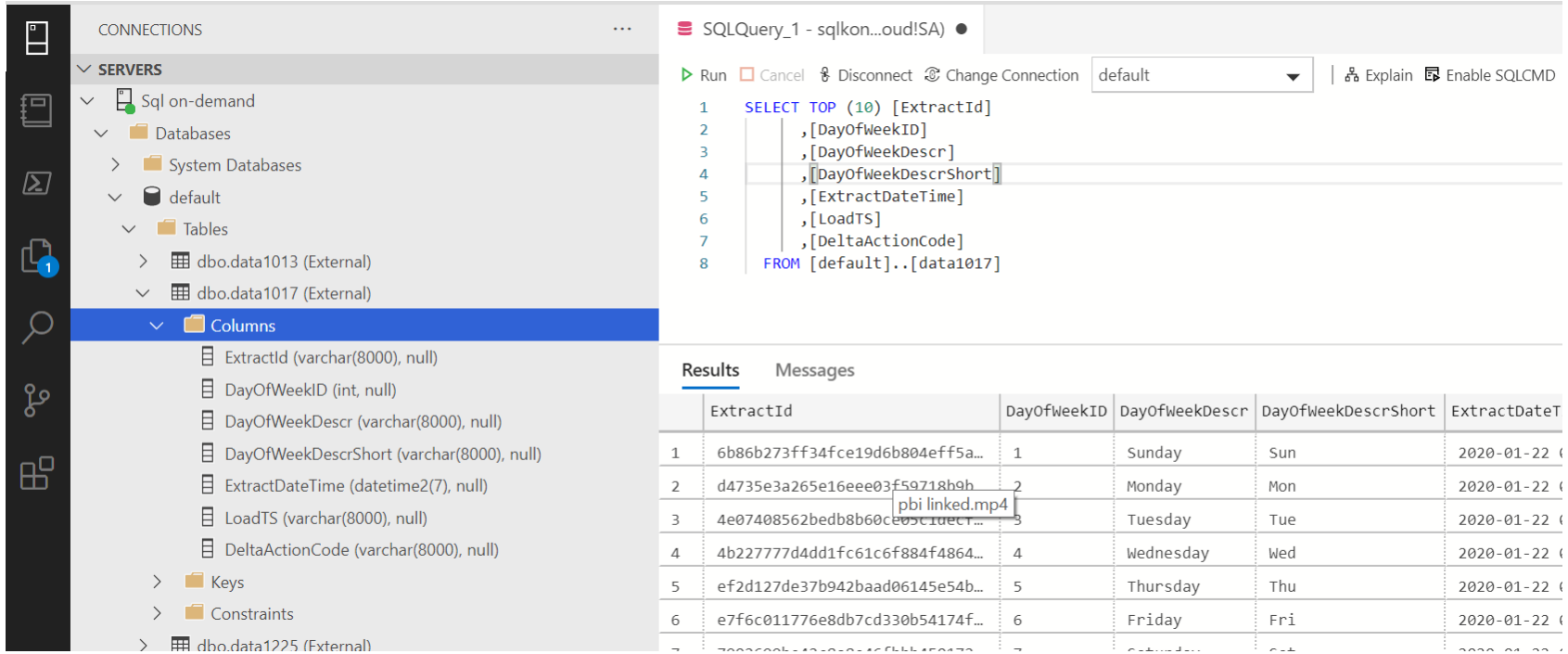
Schema definition matches original

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 references the same external files.





Microsoft

Azure Synapse Analytics

Synapse dedicated SQL pool

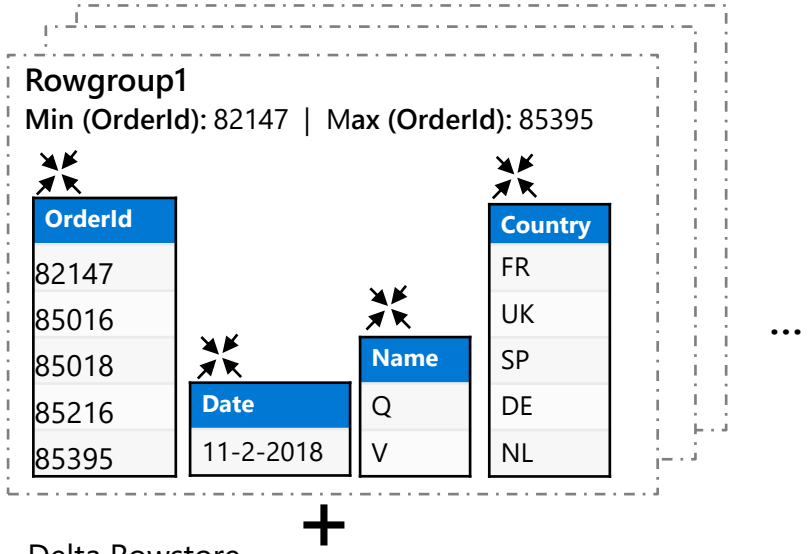
Dedicate SQL Pool Columnstore Tables

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)



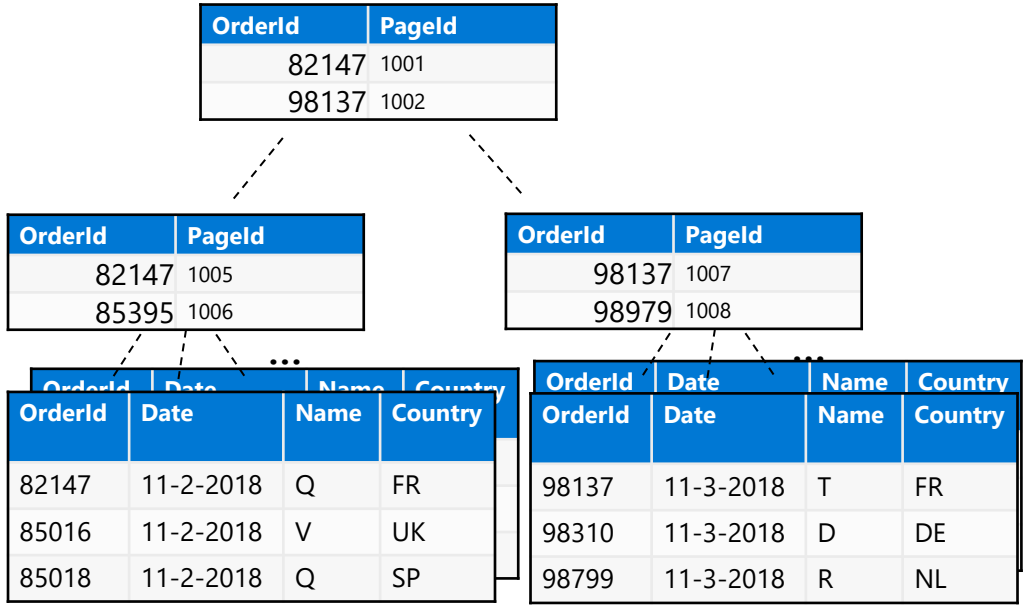
Delta Rowstore

OrderId	Date	Name	Country
98137	11-3-2018	T	FR
98310	11-3-2018	D	DE
98799	11-3-2018	R	NL
98979	11-3-2018	Z	DE

- 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

Table Distribution – Round Robin

Default distribution

Use if:

- There is no obvious joining key
- There is no good candidate column for hash distributing the table
- The table does not share a common join key with other tables
- The join is less significant than other joins in the query
- The table is a temporary staging table

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]        int           NOT NULL
,   [DateKey]               datetime      NOT NULL
,   [StoreKey]              int           NOT NULL
,   [ProductKey]            int           NOT NULL
,   [PromotionKey]          int           NOT NULL
,   [CurrencyKey]           int           NOT NULL
,   [CustomerKey]           int           NOT NULL
,   [SalesOrderNumber]      nvarchar(20)  NOT NULL
,   [SalesOrderLineNumber]  int               NULL
,   [SalesQuantity]         int           NOT NULL
,   [SalesAmount]           money         NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = ROUND_ROBIN
);
```

Table Distribution – Hash

Works well for large fact tables

Can provide high performance

Choosing distribution column is crucial

- Only one column can be chosen

Consider if:

- The table size on disk is 2GB+
- The table has frequent insert, update, and delete operations

```
CREATE TABLE [build].[FactOnlineSales]
(
    [OnlineSalesKey]        int           NOT NULL
,   [DateKey]               datetime      NOT NULL
,   [StoreKey]              int           NOT NULL
,   [ProductKey]            int           NOT NULL
,   [PromotionKey]          int           NOT NULL
,   [CurrencyKey]           int           NOT NULL
,   [CustomerKey]           int           NOT NULL
,   [SalesOrderNumber]      nvarchar(20)  NOT NULL
,   [SalesOrderLineNumber]  int               NULL
,   [SalesQuantity]         int           NOT NULL
,   [SalesAmount]           money         NOT NULL
)
WITH
(   CLUSTERED COLUMNSTORE INDEX
,   DISTRIBUTION = HASH([ProductKey])
);
```

Table Distribution – Replicate

Minimized data movement

Consider if:

- Table size on disk is <2GB
- Table is used in joins that would require extensive data movement
- Scaling oof DWU occurs rarely

```
CREATE TABLE [dbo].[DimSalesTerritory_REPLICATE]
WITH
  (
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = REPLICATE
  )
AS SELECT * FROM [dbo].[DimSalesTerritory]
OPTION  (LABEL  = 'CTAS : DimSalesTerritory_REPLICATE')
;

-- Switch table names
RENAME OBJECT [dbo].[DimSalesTerritory]
TO [DimSalesTerritory_old];
RENAME OBJECT [dbo].[DimSalesTerritory_REPLICATE]
TO [DimSalesTerritory];

DROP TABLE [dbo].[DimSalesTerritory_old];
```

Create External Table As Select

Overview

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

Steps:

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

```
-- Create a database master key if one does not already exist
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!nfo'
;
-- Create a database scoped credential with Azure storage account key as the secret.
CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY   = '<my_account>'
,   SECRET     = '<azure_storage_account_key>'
;
-- Create an external data source with CREDENTIAL option.
CREATE EXTERNAL DATA SOURCE MyAzureStorage
WITH
(   LOCATION   = 'wasbs://daily@logs.blob.core.windows.net/'
,   CREDENTIAL = AzureStorageCredential
,   TYPE       = HADOOP
)
-- Create an external file format
CREATE EXTERNAL FILE FORMAT MyAzureCSVFormat
WITH (FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS(
       FIELD_TERMINATOR = ',',
       FIRST_ROW = 2)
--Create an external table
CREATE EXTERNAL TABLE dbo.FactInternetSalesNew
WITH(
      LOCATION = '/files/Customer',
      DATA_SOURCE = MyAzureStorage,
      FILE_FORMAT = MyAzureCSVFormat
  )
AS SELECT T1.* FROM dbo.FactInternetSales T1 JOIN dbo.DimCustomer T2
ON ( T1.CustomerKey = T2.CustomerKey )
OPTION ( HASH JOIN );
```

COPY command

Overview

Copies data from source to destination

Benefits

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

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

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

Supports CSV, PARQUET, ORC file formats

```
COPY INTO test_1
FROM 'https://XXX.blob.core.windows.net/customerdatasets/test_1.txt'
WITH (
   FILE_TYPE = 'CSV',
   CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>'),
   FIELDQUOTE = '"',
   FIELDTERMINATOR=';',
   ROWTERMINATOR='0X0A',
   ENCODING = 'UTF8',
   DATEFORMAT = 'ymd',
   MAXERRORS = 10,
   ERRORFILE = '/errorsfolder/'--path starting from the storage container,
   IDENTITY_INSERT
   )
```

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

Result-set caching

Overview

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

The result-set cache persists even if dedicated 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

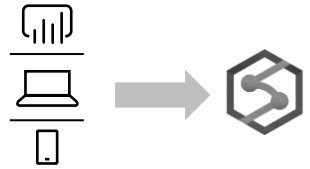
Enhances performance when same result is requested repetitively

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
```

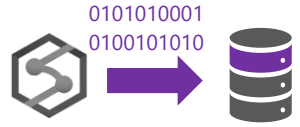
Result-set caching flow



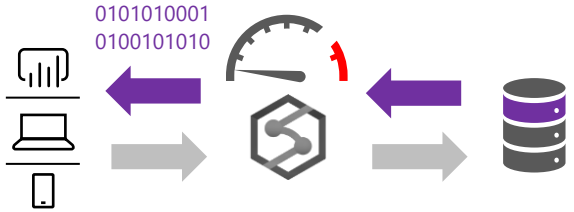
1. Client sends query to dedicated SQL pool

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

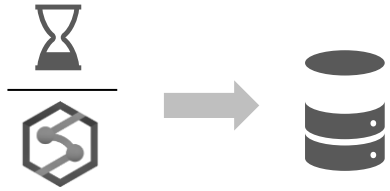
+



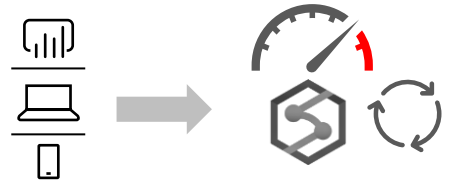
Query results are cached in remote storage so subsequent requests can be served immediately



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;
```

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
```

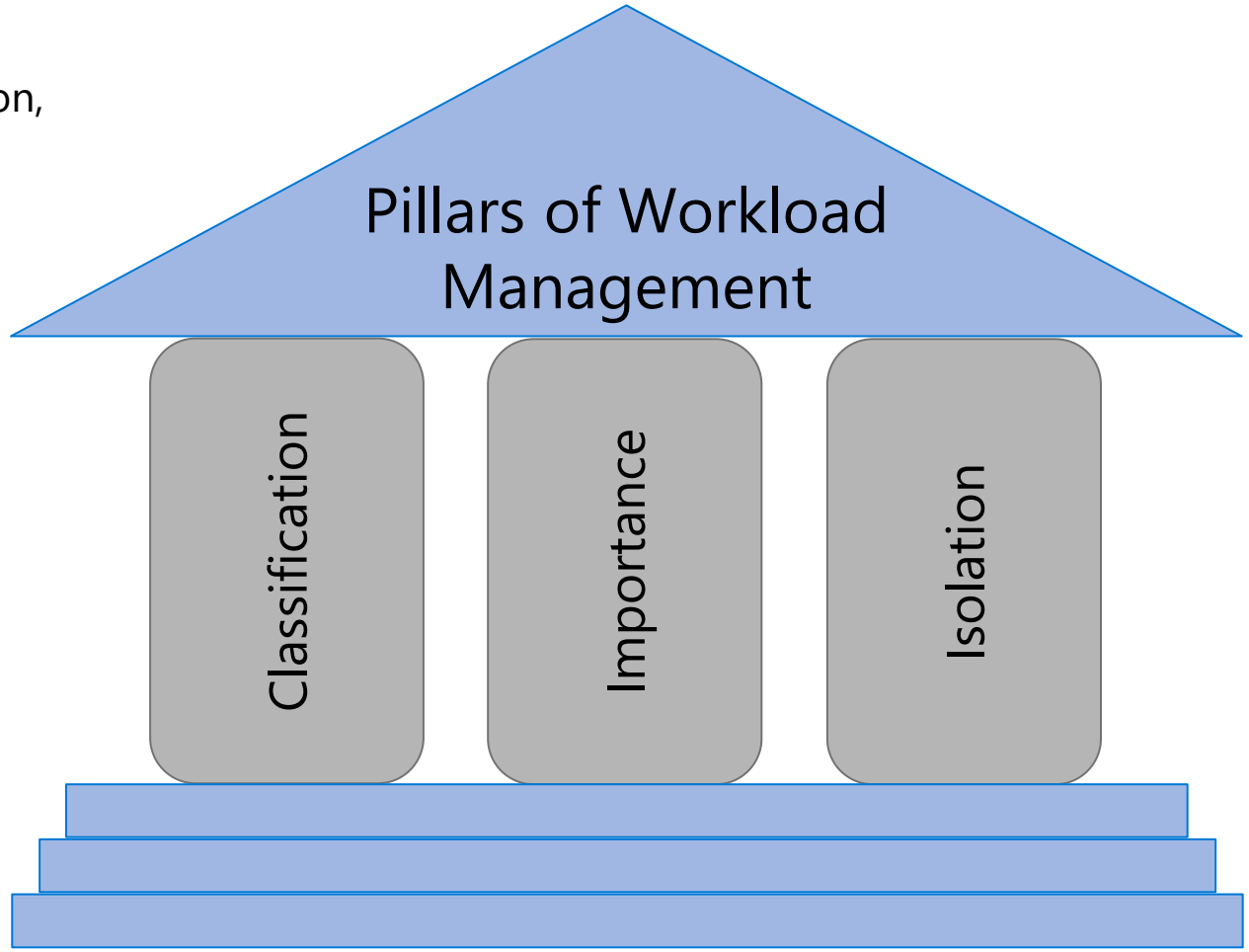
Workload Management

Overview

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

The three pillars of workload management are

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



Workload classification

Overview

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

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

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

Benefits

Map queries to both Resource Management and Workload Isolation concepts.

Monitoring DMVs

sys.workload_management_workload_classifiers
sys.workload_management_workload_classifier_details

Query DMVs to view details about all active workload classifiers.

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

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

Workload importance

Overview

Queries past the concurrency limit enter a FiFo queue

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

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

Example Video

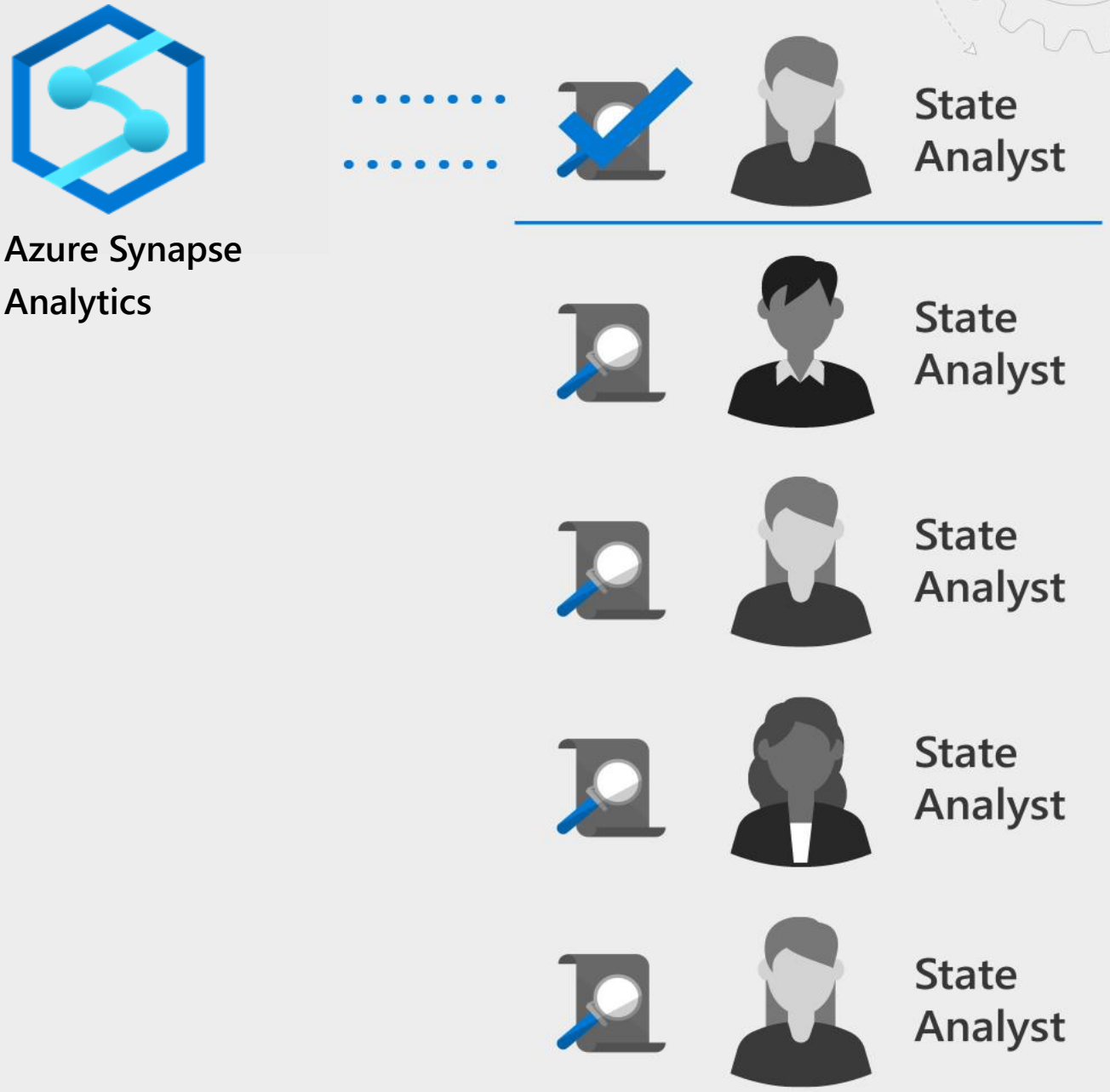
State analysts have normal importance.

National analyst is assigned high importance.

State analyst queries execute in order of arrival

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

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



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 Synapse SQL (provisioned) 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

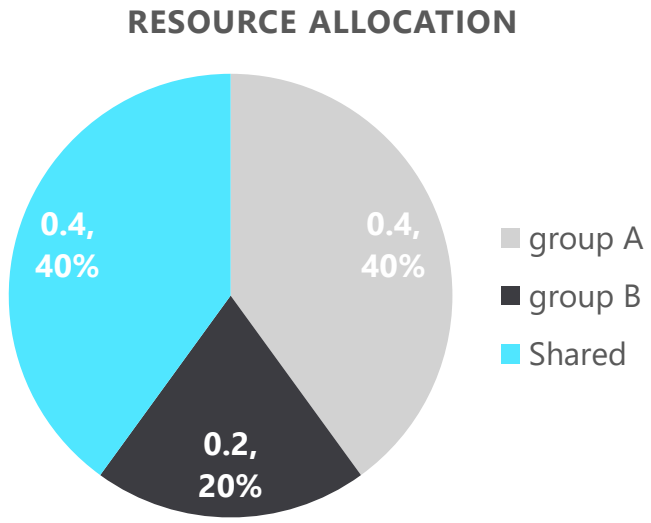
Set Query timeout value. Get DBAs out of the business of killing runaway queries

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 ]
)[ ; ]
```



Continuous integration and delivery (CI/CD)

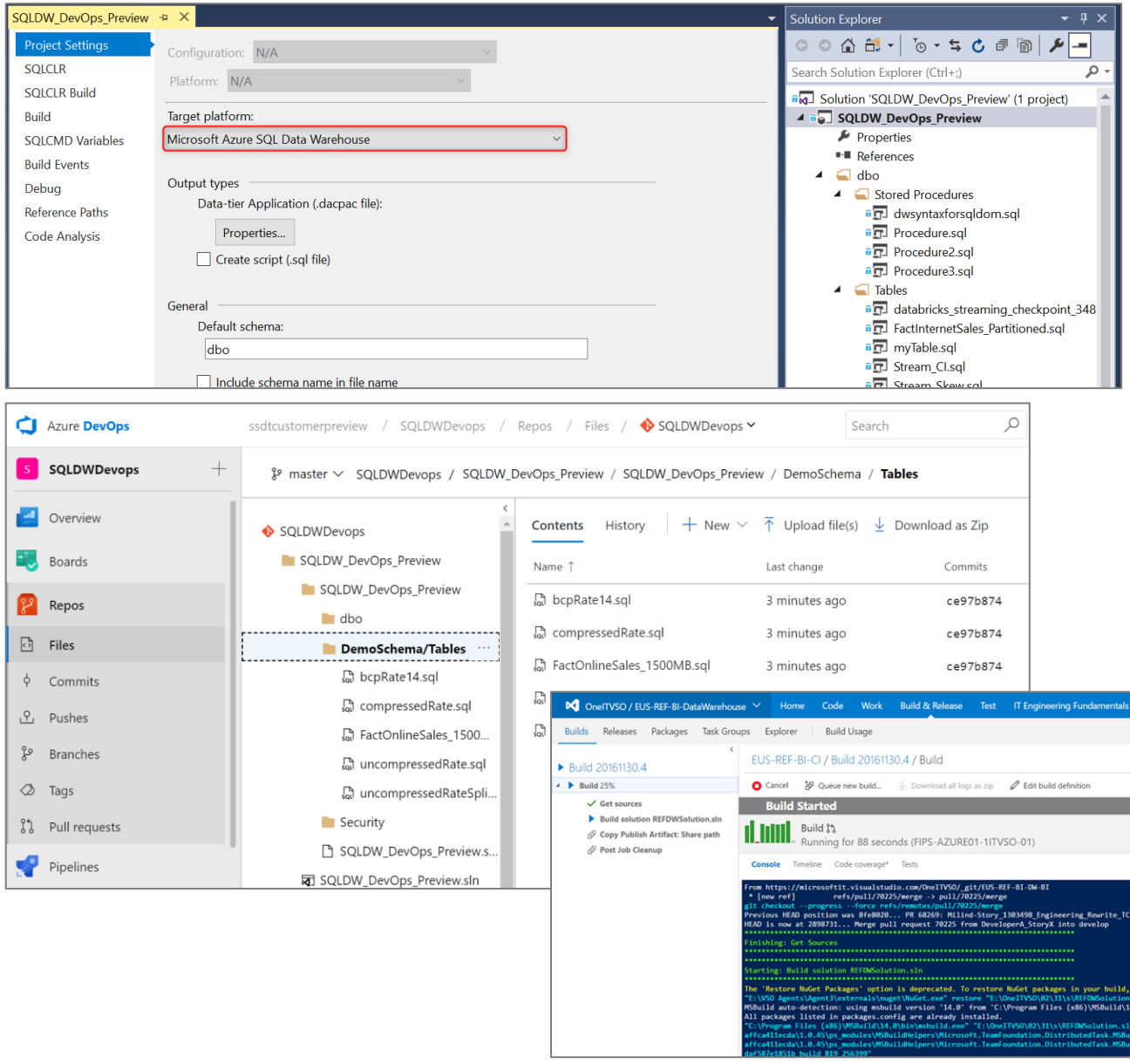
Overview

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.)



Automatic statistics management

Overview

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

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

Statistics are considered out of date when:

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

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

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

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

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

Maintenance windows

Overview

Choose a time window for your upgrades.

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

Windows can be from 3 to 8 hours.

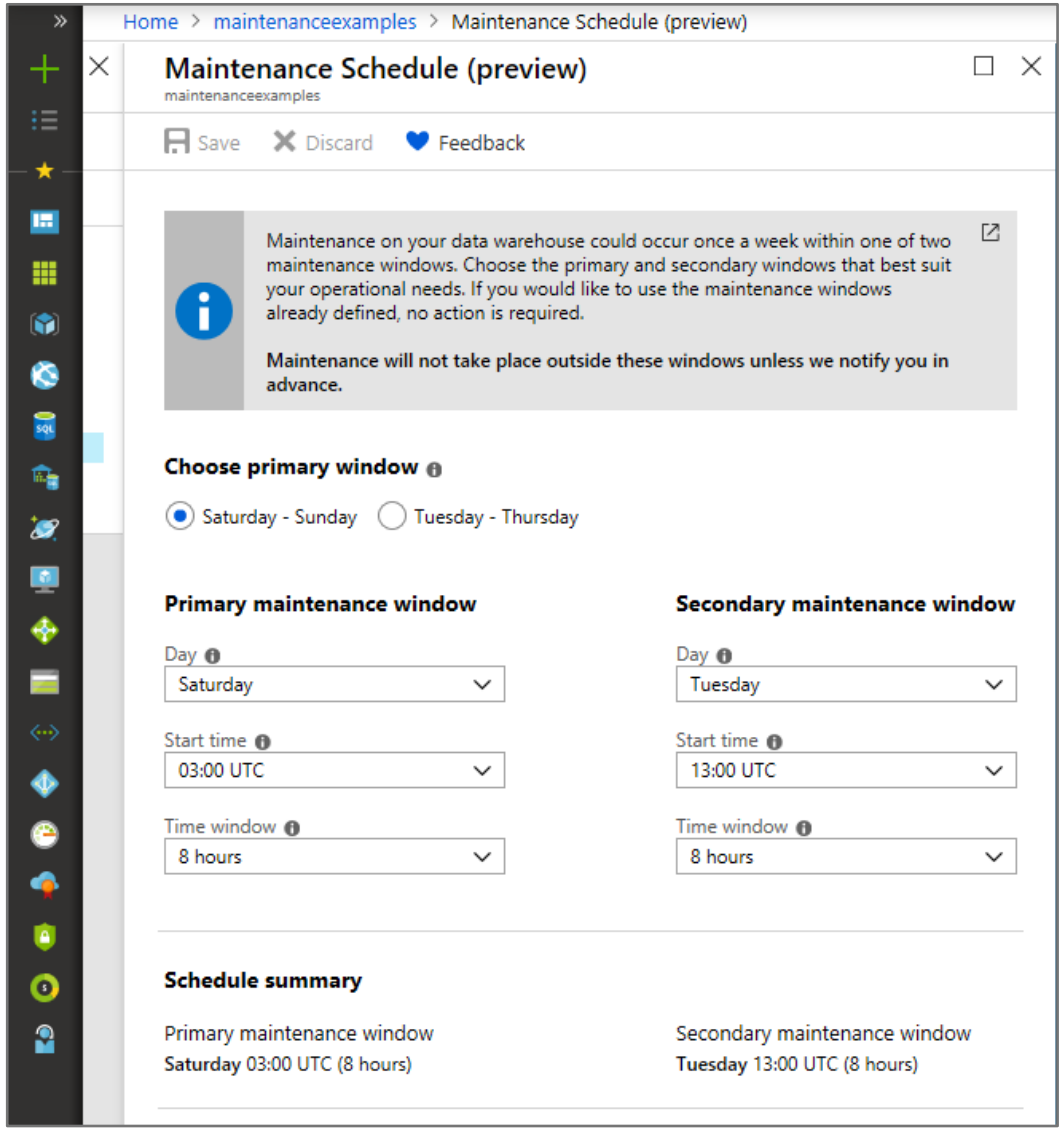
24-hour advance notification for maintenance events.

Benefits

Ensure upgrades happen on your schedule.

Predictable planning for long-running jobs.

Stay informed of start and end of maintenance.



Azure Advisor recommendations

Suboptimal Table Distribution

Reduce data movement by replicating tables

Data Skew

Choose new hash-distribution key

Slowest distribution limits performance

Cache Misses

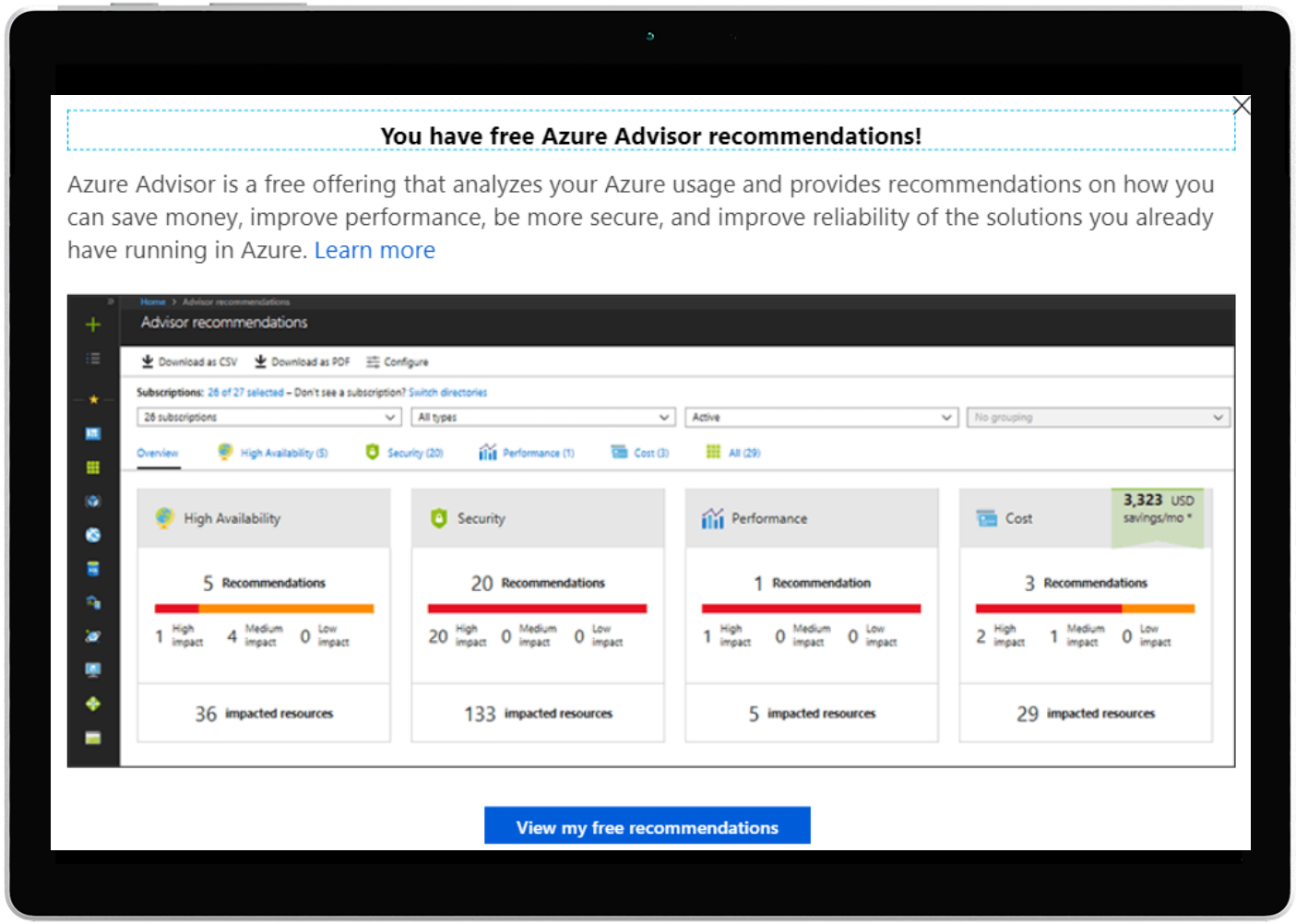
Provision additional capacity

Tempdb Contention

Scale or update user resource class

Suboptimal Plan Selection

Create or update table statistics



Microsoft

Azure Synapse Analytics

Apache Spark

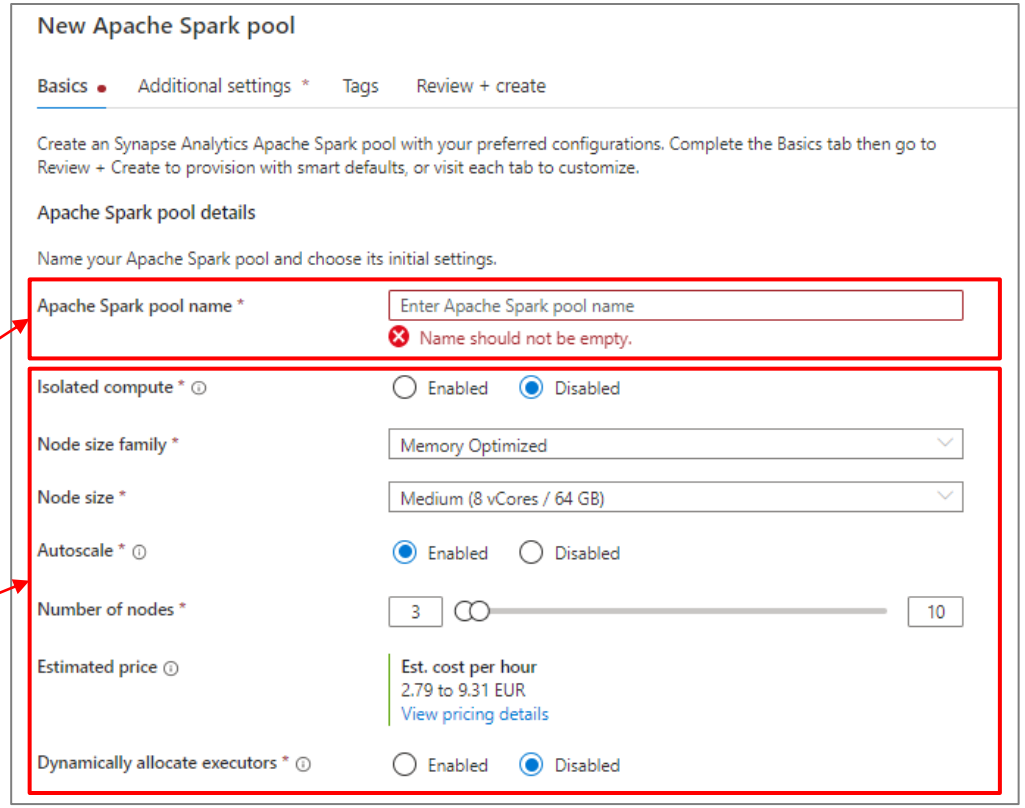
Creating a Spark pool (1 of 2)

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

Basic Settings – Minimum details required from user

Only required field from user

Default Settings

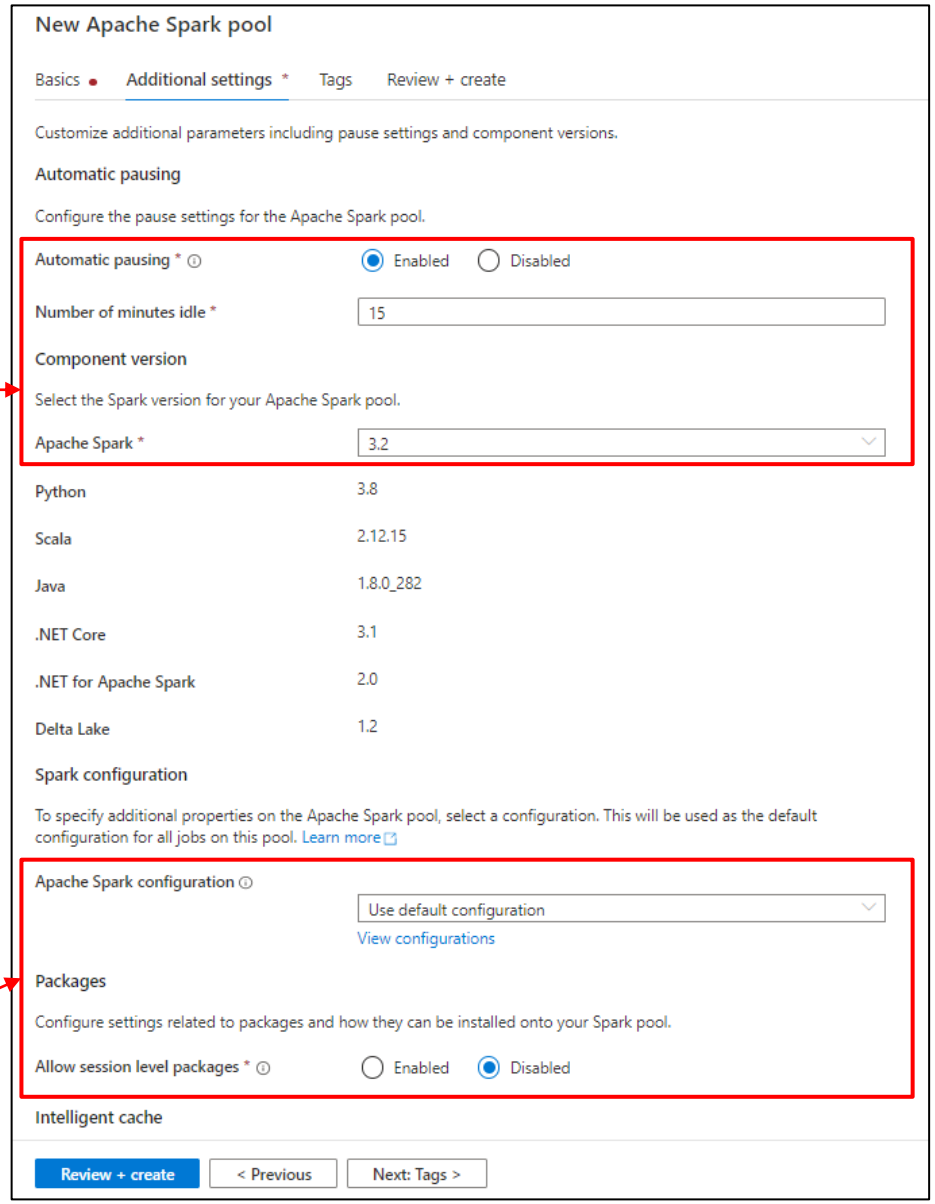


Creating a Spark pool (2 of 2) - optional

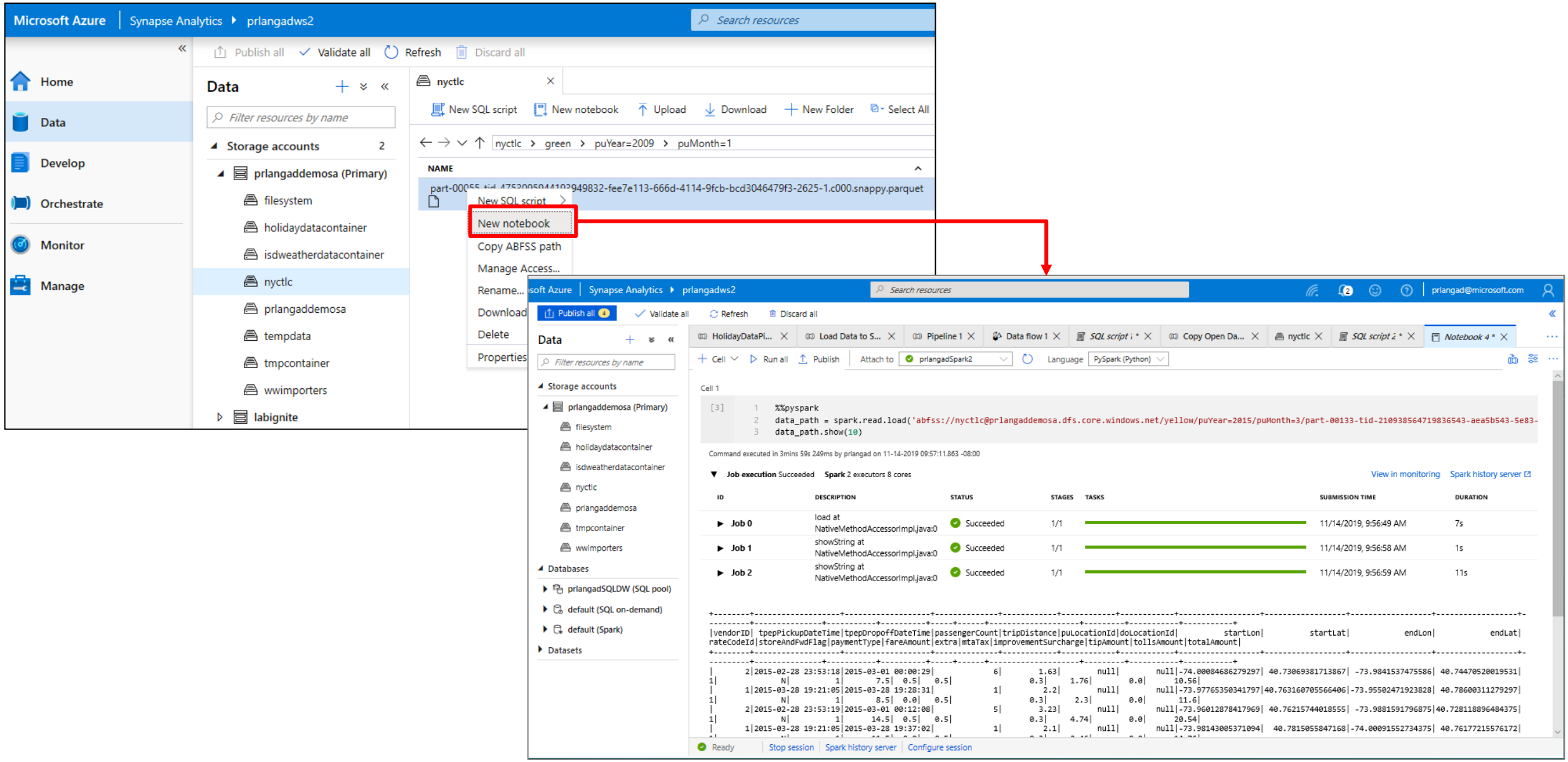
Additional Settings offer optional settings to customize Spark pool

Customize component versions, auto-pause

Import libraries using Apache Spark configuration or session settings



Create Notebook on files in storage



Synapse ML

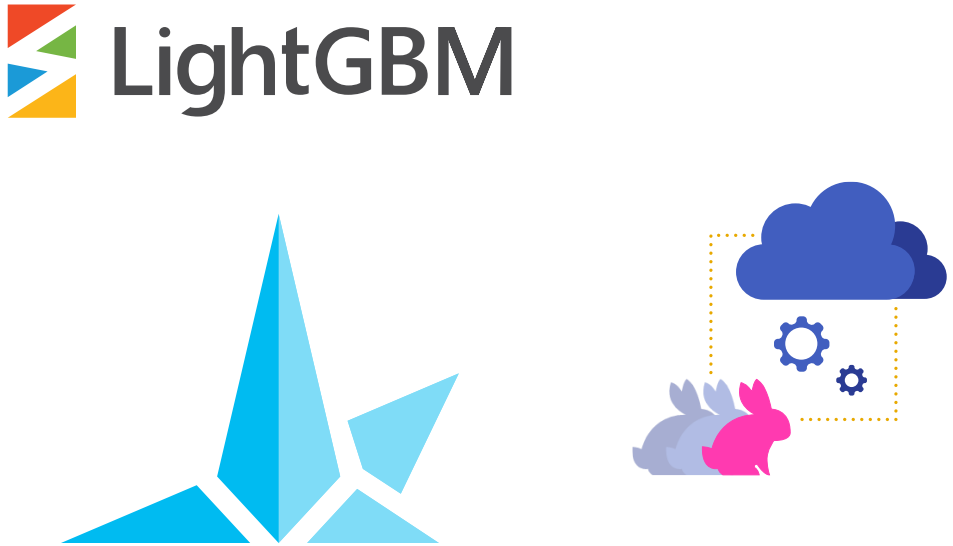
Open-Source Spark Library

Distributed ML algorithms for Apache Spark, e.g.:

- Linear models & reinforcement learning: Vowpal Wabbit
- Gradient boosted trees: LightGBM
- Novelty detection: Isolation Forests

Cognitive Services Integration for Spark

Language Binding Generators: Python, R, Java (, .NET)



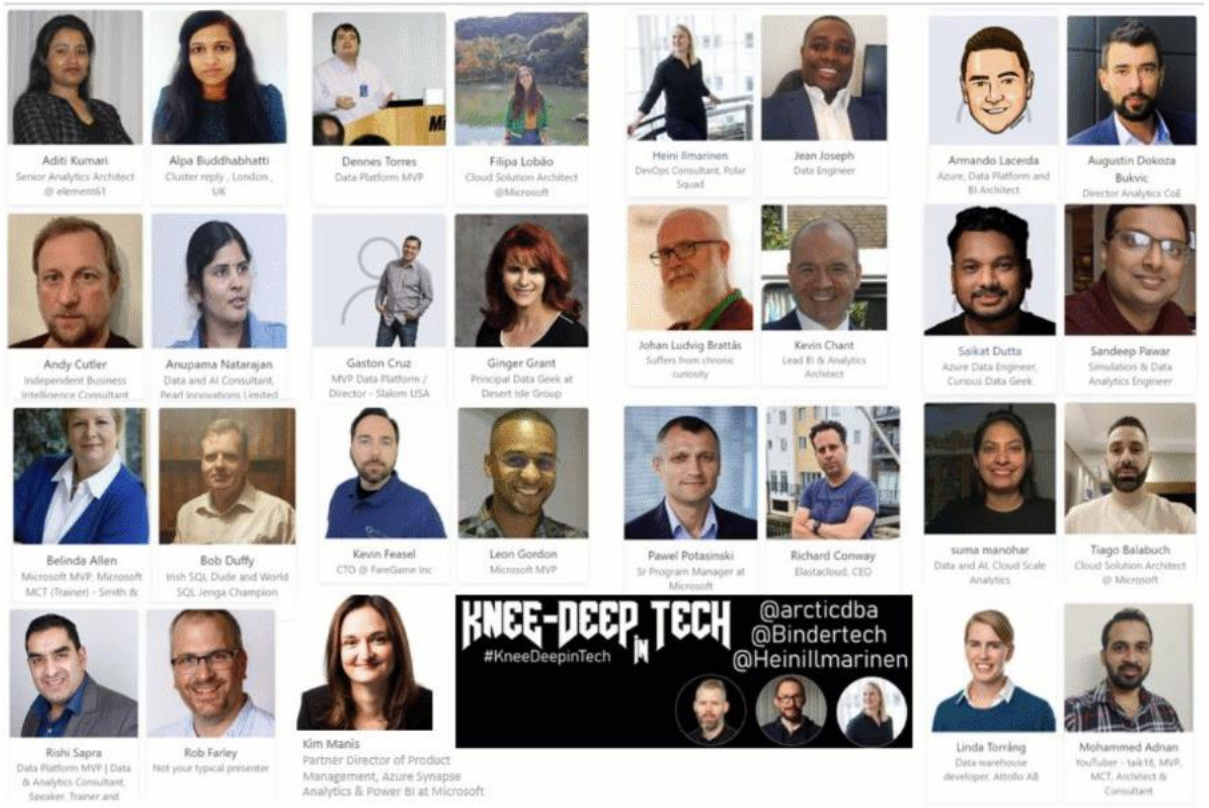
https://aka.ms/spark

Synapse live | Validate all | Publish all 1

SynapseML CogSvcs...

Run all | Undo | Publish | Outline | Attach to SparkPoolM1 | Language PySpark (Python) | Variables

Ready



+ Code + Markdown

```
1  from synapse.ml.cognitive import *
2
3  df = spark.createDataFrame([
4          ("https://dlssynapsedeveastus1.blob.core.windows.net/public/DataTobogganSpeakers.jpg", )
5      ], ["image", ])
6
7  ri = (ReadImage()
8      .setLinkedService("LS_CV_SYNAPSE_DEV_EASTUS_1")
9      .setImageUrlCol("image")
10     .setOutputCol("ocr"))
```

Microsoft

Azure Synapse Analytics

Security

Securing with firewalls

Overview

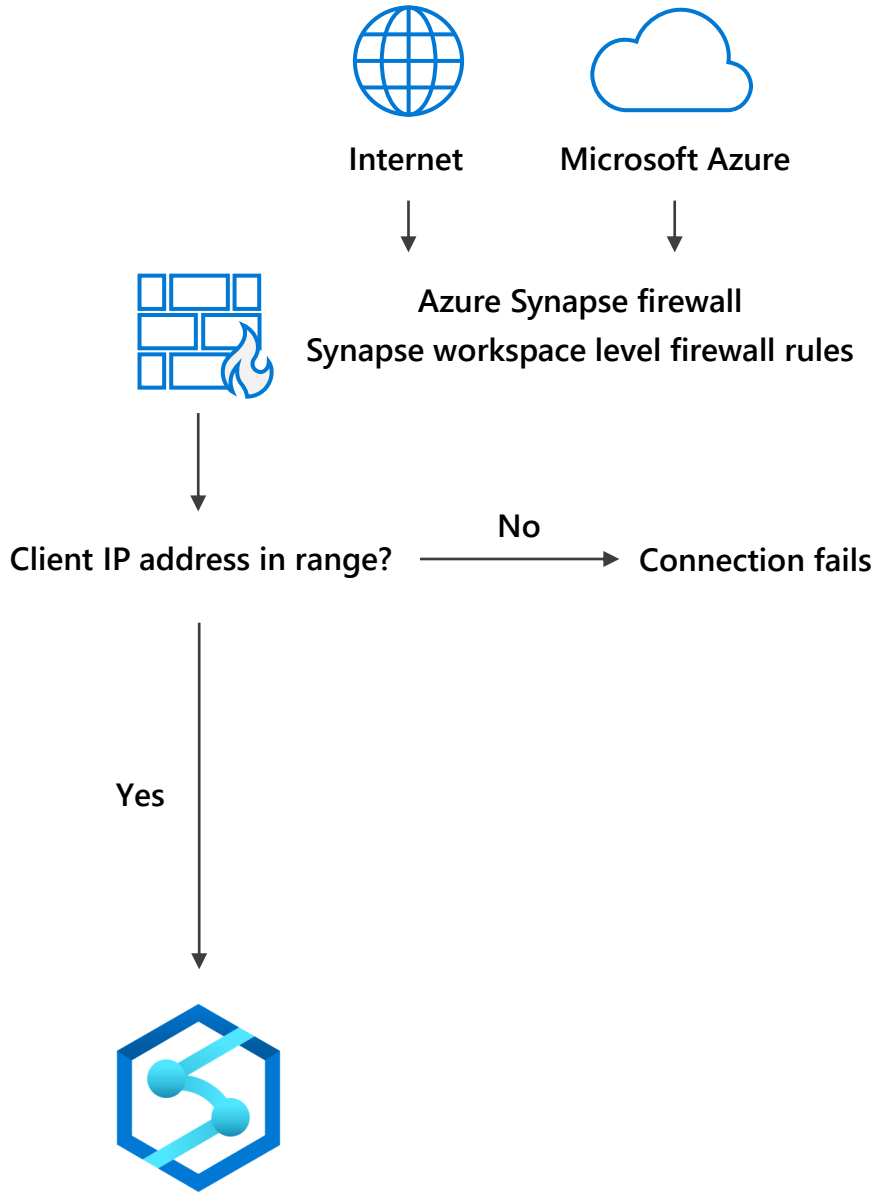
Access to your Azure Synapse Analytics is blocked by the firewall.

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

Rules

Allow specific or range of whitelisted IP addresses.

Allow Azure applications to connect.



IP Firewall Rules

Overview

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

IP firewall rules configured at the workspace level apply to all public endpoints of the workspace (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.

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.

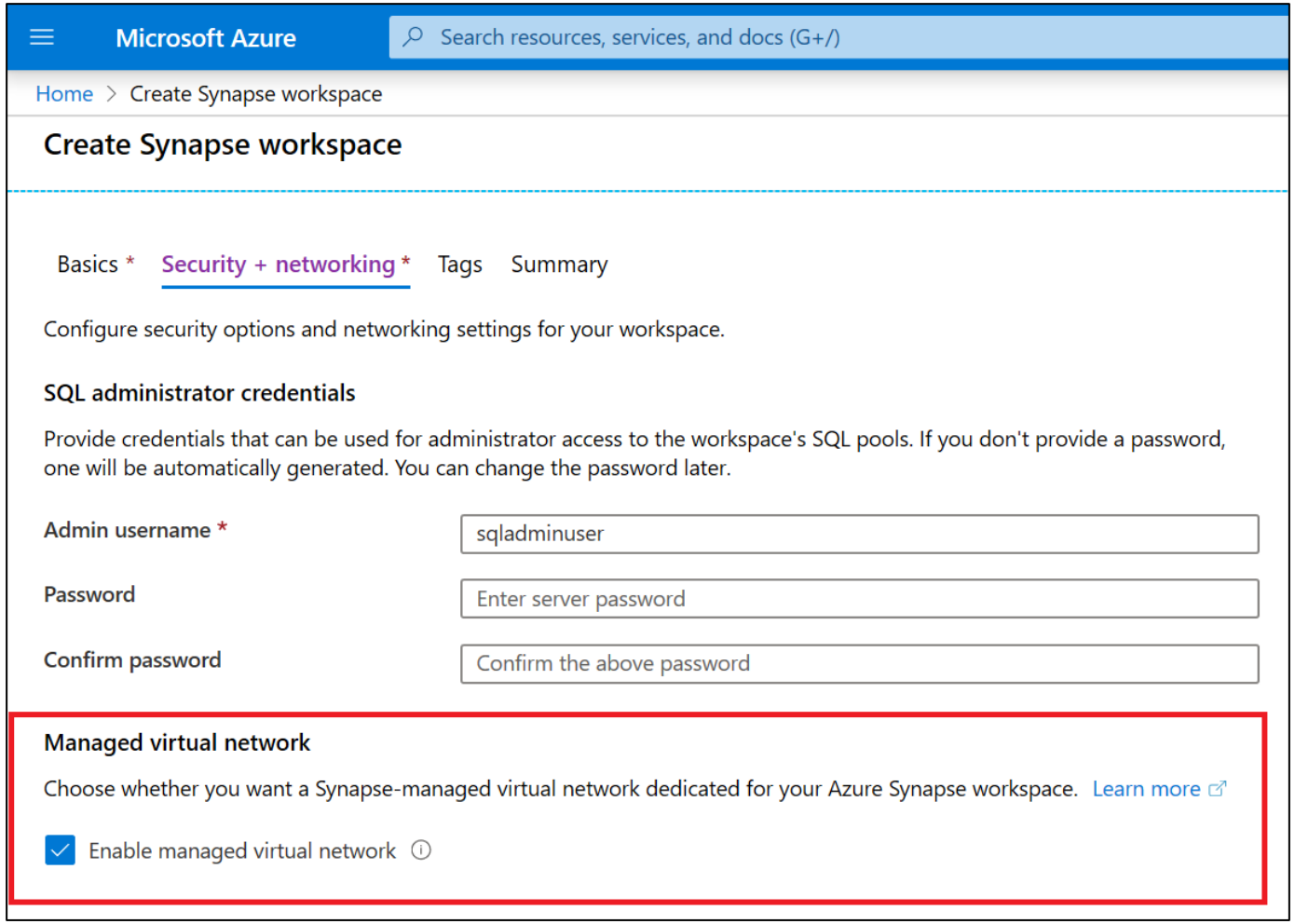
Managed VNet

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.



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

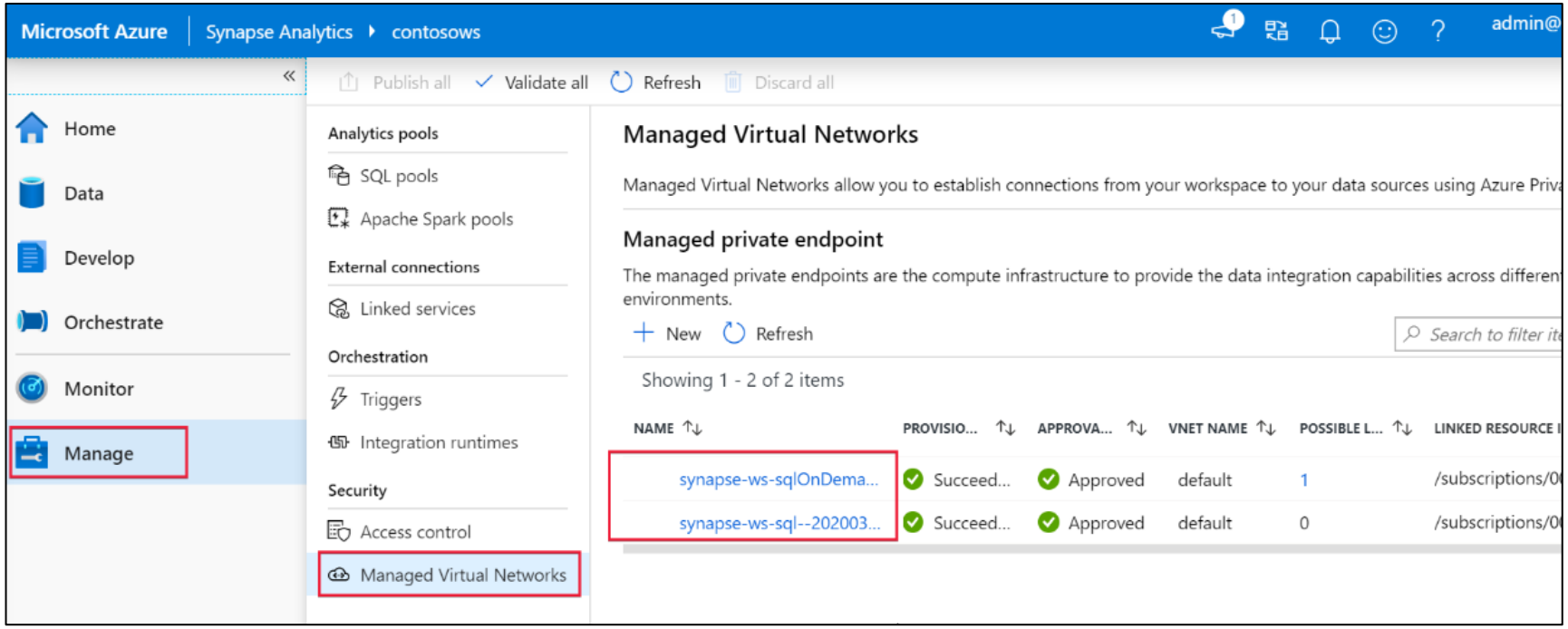
- Private link enables customers to access Azure services and Azure hosted customer/partner services from their Azure VNet securely.
- With use of private link, traffic between customer's VNet and workspace traverses entirely over the Microsoft backbone network.
- Private link protects against data exfiltration 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 (provisioned & serverless)

Overview

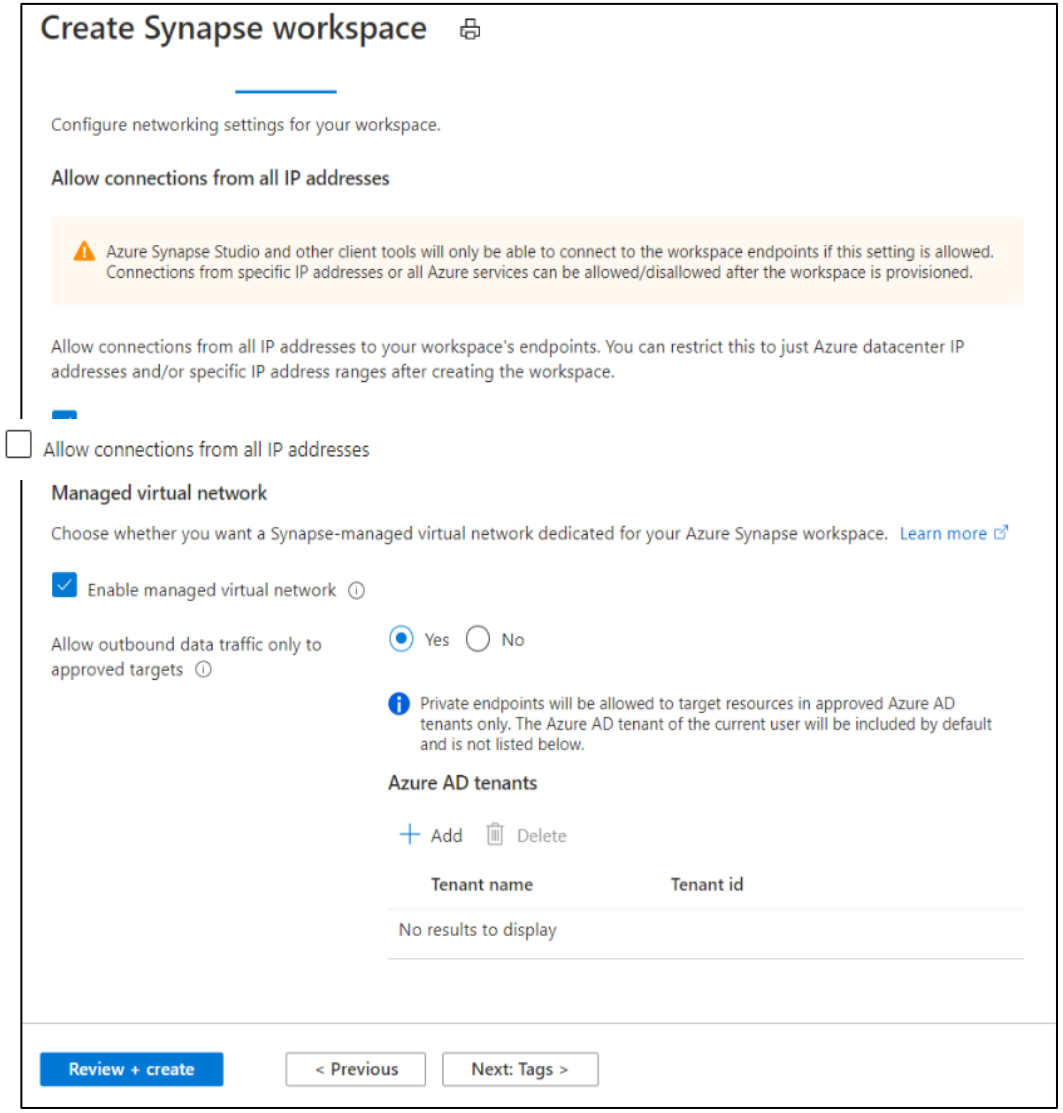
Dedicated SQL pool and serverless SQL pool use multi-tenant infrastructure that is not deployed into the Managed workspace VNet.

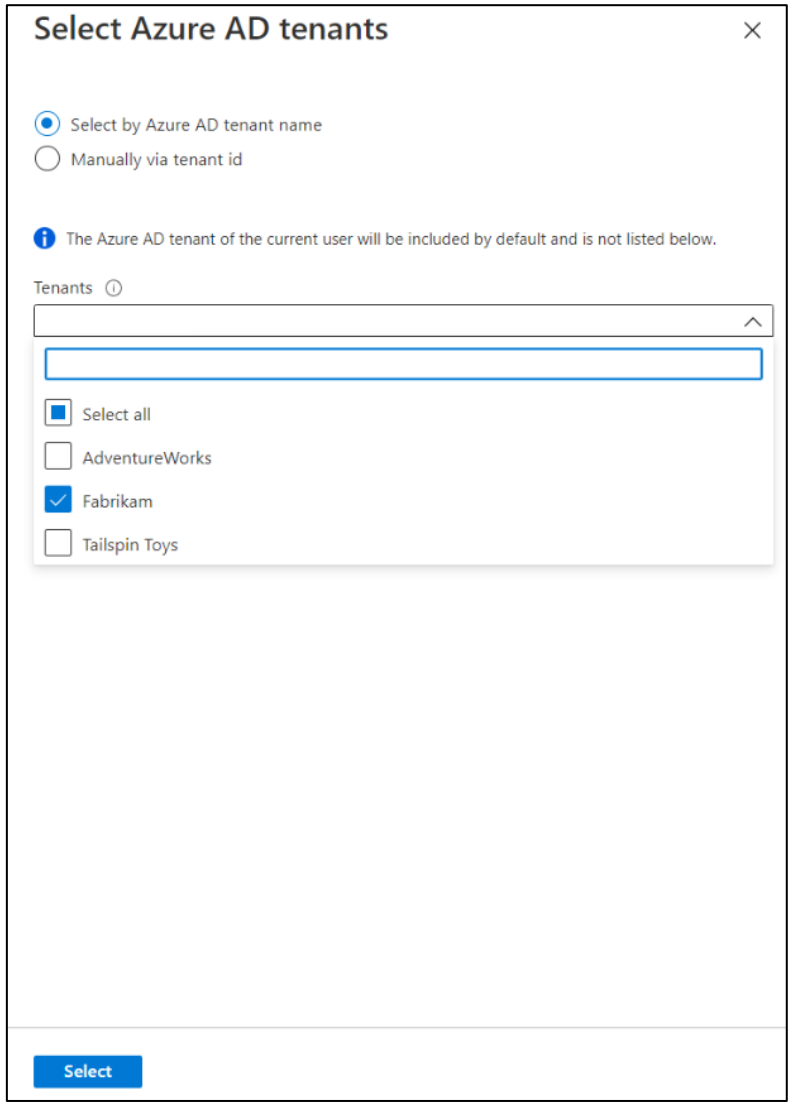
Azure Synapse creates two managed private endpoints to dedicated SQL pool and serverless SQL pool in that workspace. Customers do not get charged for these two Managed private endpoints.



Data Exfiltration

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





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.

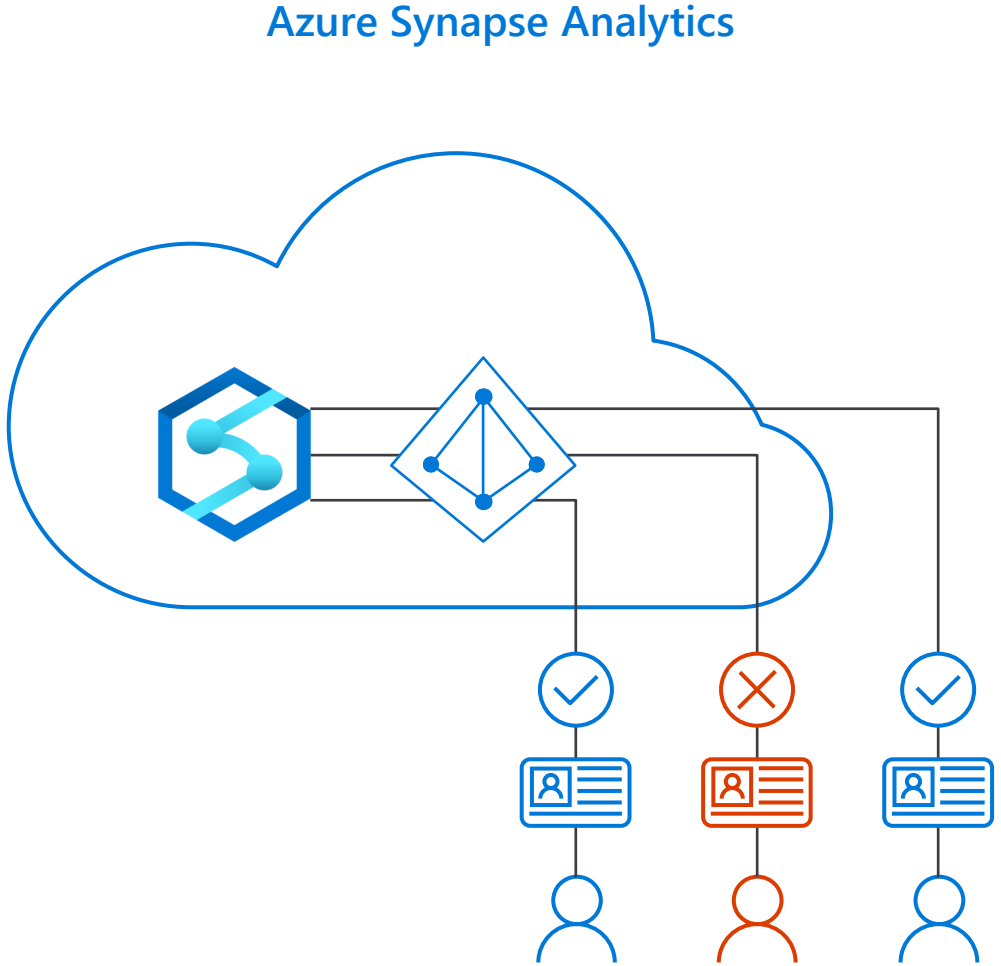
Benefits

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

Allows password rotation in a single place

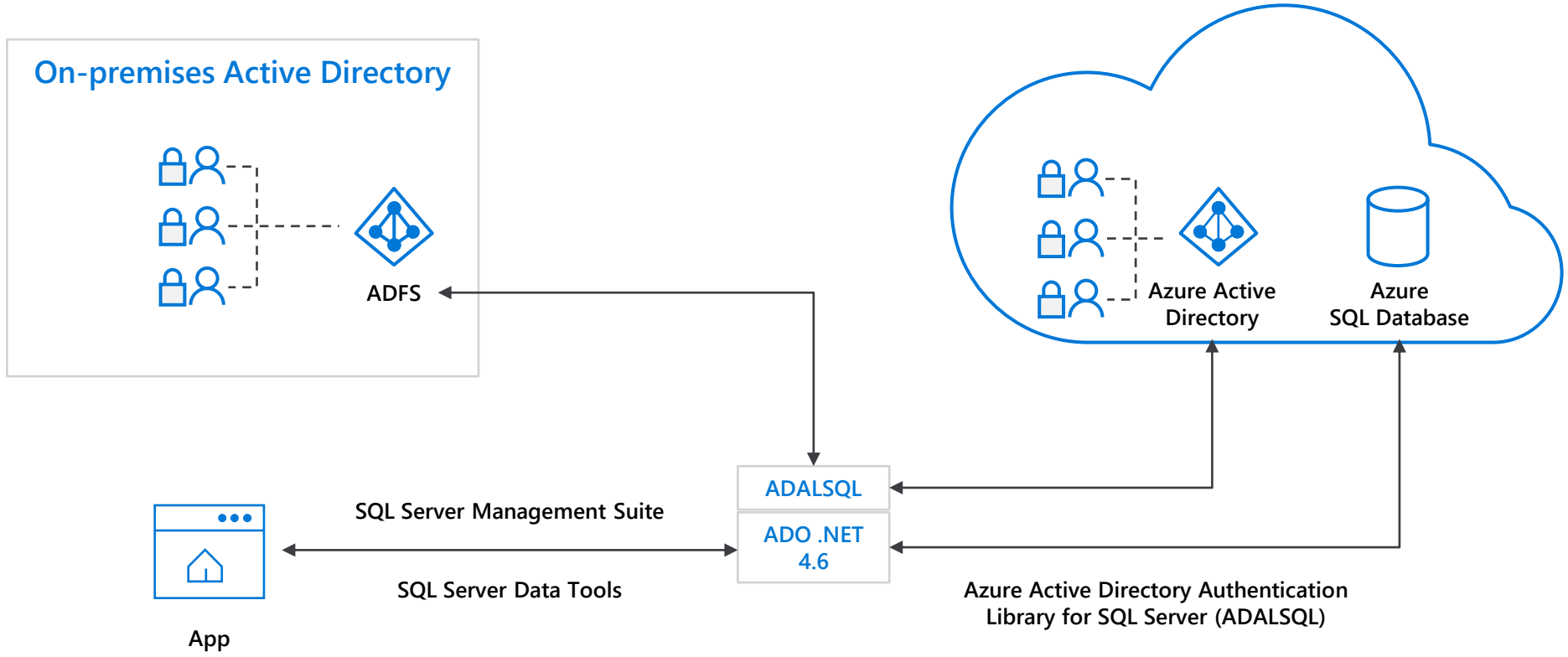
Alternative to SQL Server authentication

Eliminates the need to store passwords

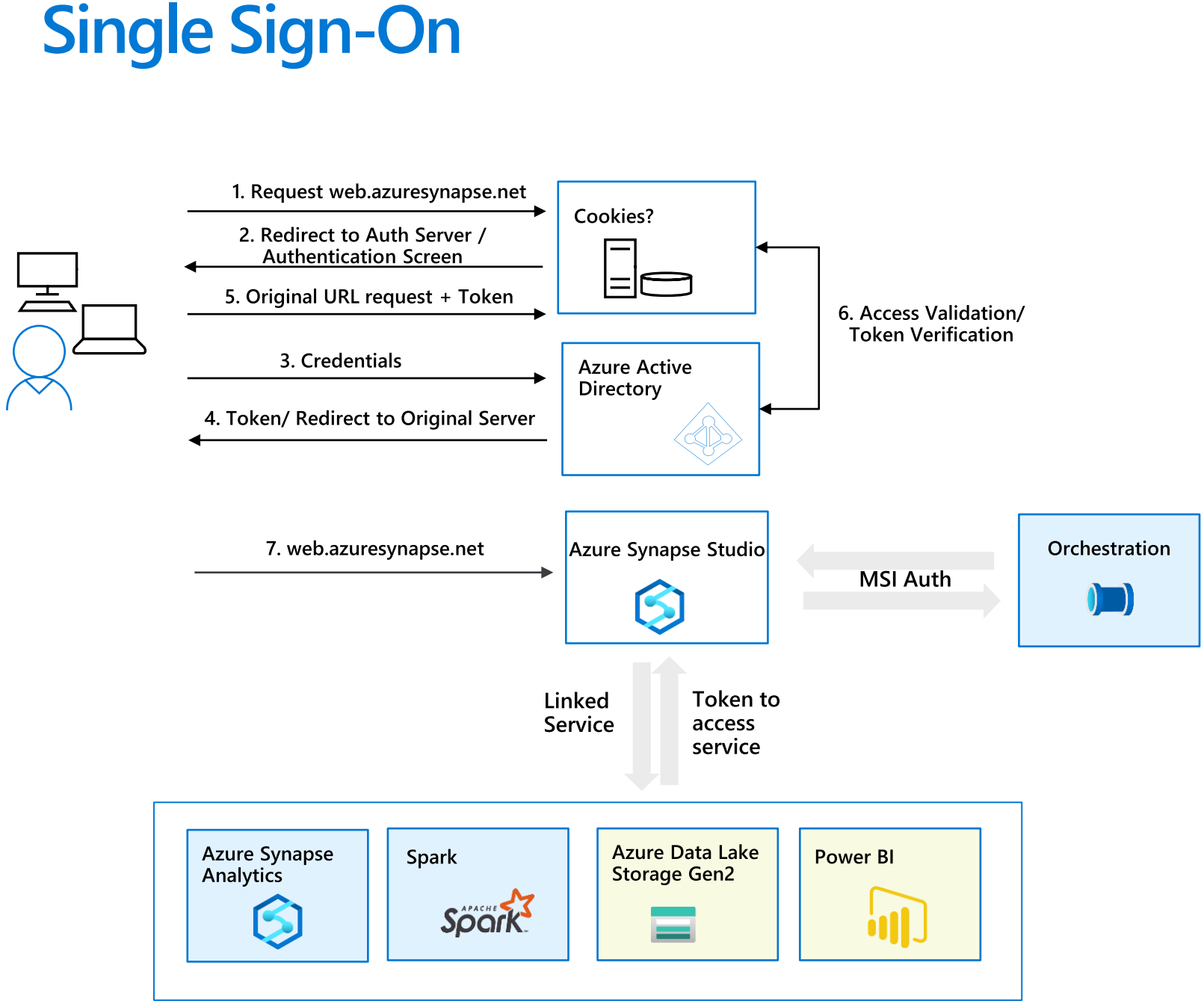


Azure Active Directory trust architecture

Azure Active Directory and Azure Synapse Analytics



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:

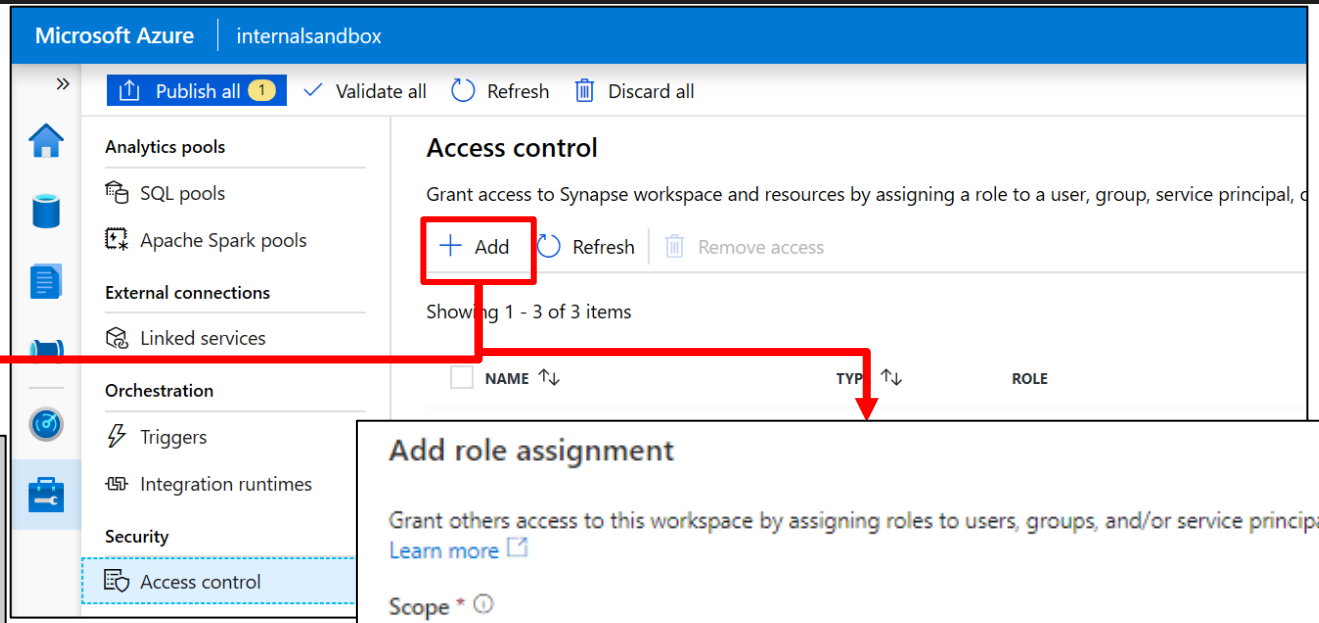
1. ADLS Gen2
2. Azure Synapse Analytics
3. Power BI
4. Spark – Spark Livy API
5. management.azure.com – resource provisioning
6. Develop artifacts – dev.workspace.net
7. Graph endpoints

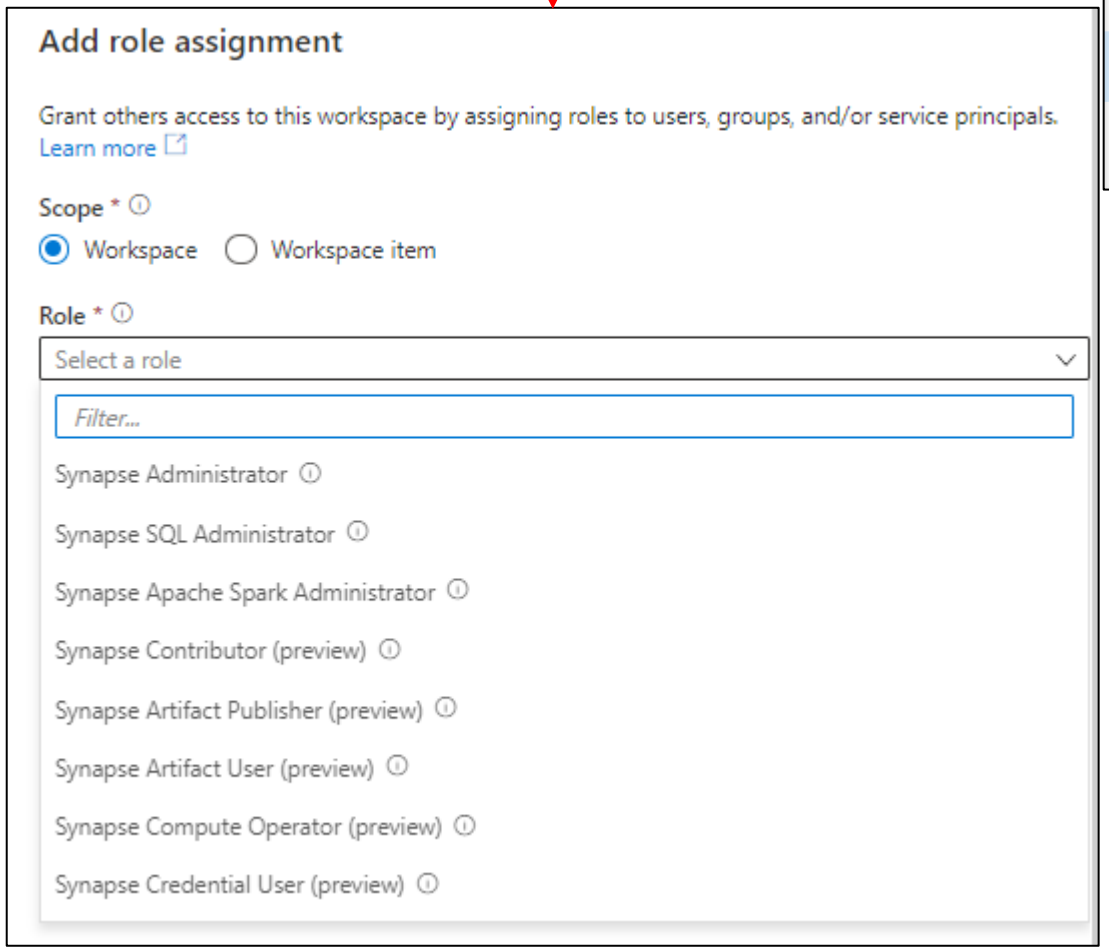
MSI authentication - Orchestration uses workspace MSI auth for automation

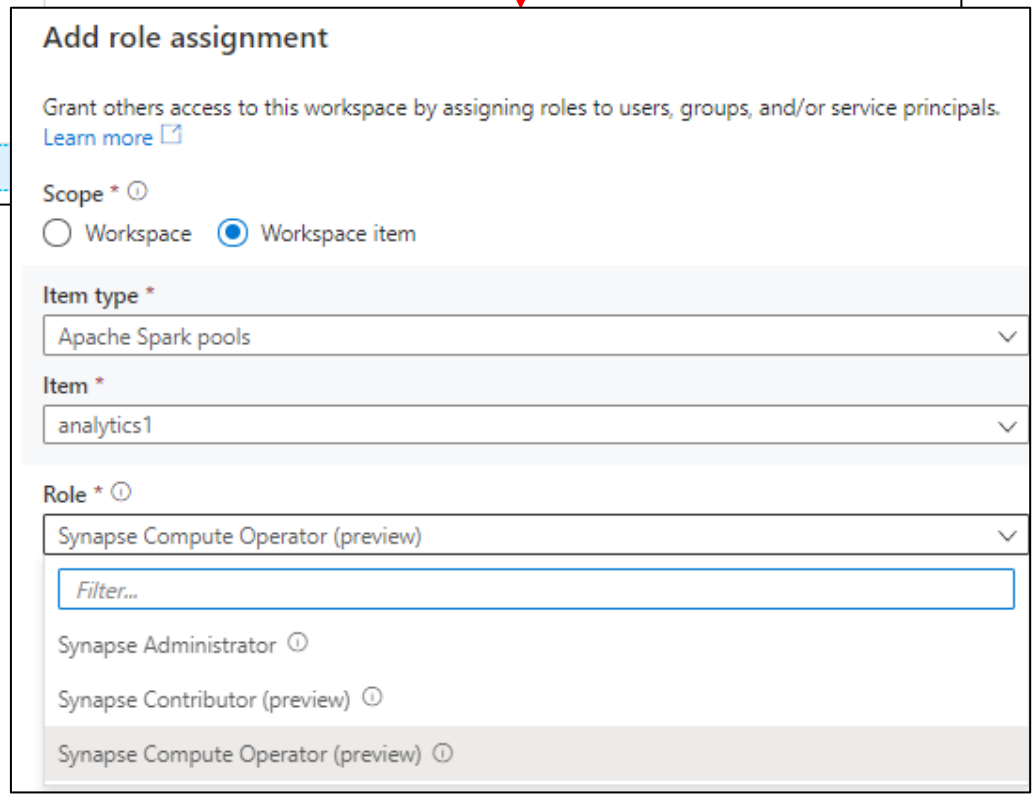
Access Control

Overview

It provides access control management to workspace resources and artifacts







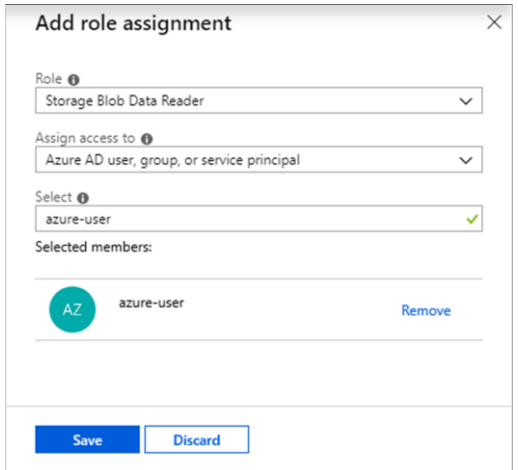
Access control – serverless SQL pool

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'
```

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

Overview

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

Prevent unauthorized queries against certain tables.

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

```
-- Grant SELECT permission to user RosaQdM on table Person.Address in the AdventureWorks2012 database
GRANT SELECT ON OBJECT::Person.Address TO RosaQdM;
GO
-- Grant REFERENCES permission on column BusinessEntityID in view HumanResources.vEmployee to user Wanida
GRANT REFERENCES(BusinessEntityID) ON OBJECT::HumanResources.vEmployee to Wanida with GRANT OPTION;
GO
-- Grant EXECUTE permission on stored procedure HumanResources.uspUpdateEmployeeHireInfo to an application role called Recruiting11
USE AdventureWorks2012;
GRANT EXECUTE ON OBJECT::HumanResources.uspUpdateEmployeeHireInfo TO RECRUITING 11;
GO
```

Row-level security (RLS)

Overview

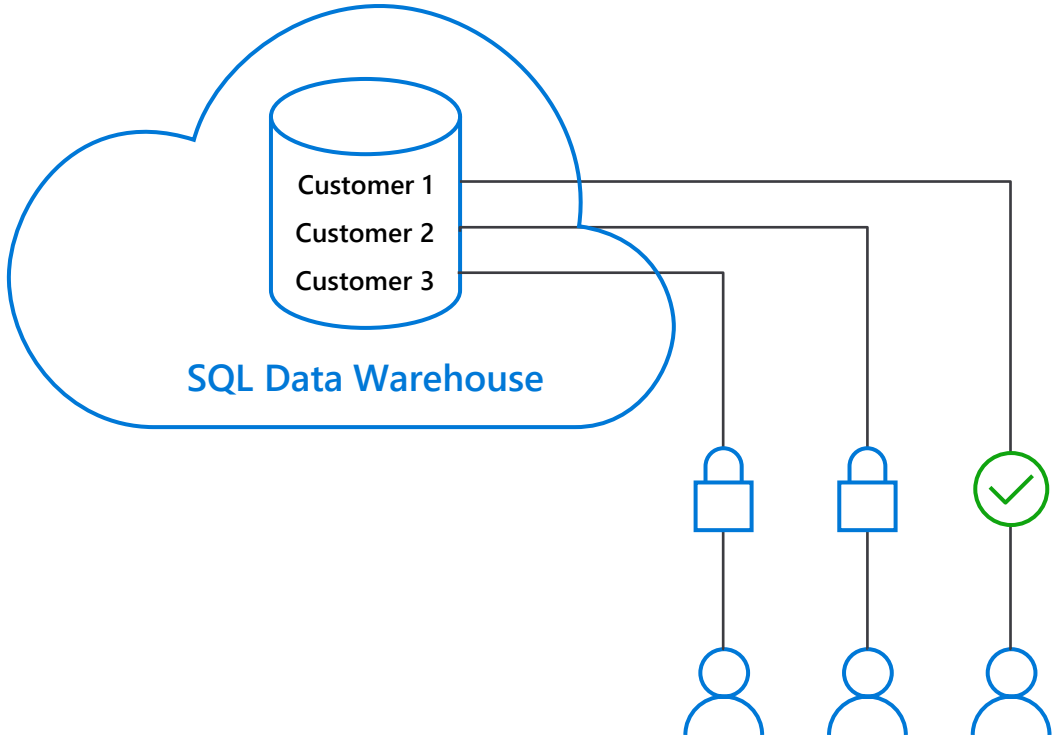
Fine grained access control of specific rows in a database table.

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

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

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

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



Row-level security

Creating policies

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

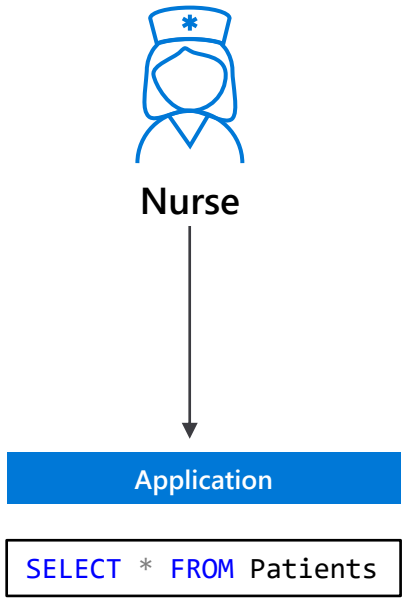
The following examples demonstrate the use of the CREATE SECURITY POLICY syntax

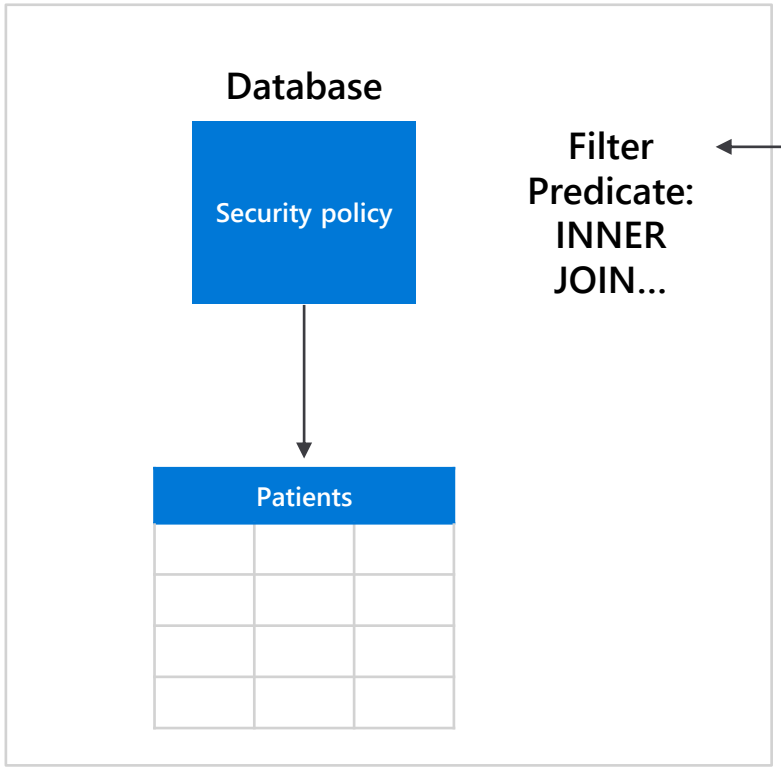
```
-- The following syntax creates a security policy with a filter predicate for the Customer table
CREATE SECURITY POLICY [FederatedSecurityPolicy]
ADD FILTER PREDICATE [rls].[fn_securitypredicate]([CustomerId])
ON [dbo].[Customer];


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

Row-level security

Three steps:

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





Filter Predicate: INNER JOIN...

Policy manager

```
CREATE FUNCTION dbo.fn_securitypredicate(@wing int)
    RETURNS TABLE WITH SCHEMABINDING AS
    return SELECT 1 as [fn_securitypredicate_result] FROM
        StaffDuties d INNER JOIN Employees e
        ON (d.EmpId = e.EmpId)
        WHERE e.UserSID = SUSER_SID() AND @wing = d.Wing;

CREATE SECURITY POLICY dbo.SecPol
    ADD FILTER PREDICATE dbo.fn_securitypredicate(Wing) ON Patients
    WITH (STATE = ON)
```

```
SELECT * FROM Patients
   SEMIJOIN APPLY dbo.fn_securitypredicate(patients.Wing);
```

```
SELECT Patients.* FROM Patients,
   StaffDuties d INNER JOIN Employees e ON (d.EmpId = e.EmpId)
   WHERE e.UserSID = SUSER_SID() AND Patients.wing = d.Wing;
```

Column-level security

Overview

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

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

Administer via GRANT T-SQL statement.

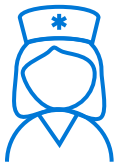
Both Azure Active Directory (AAD) and SQL authentication are supported.

Column-level security

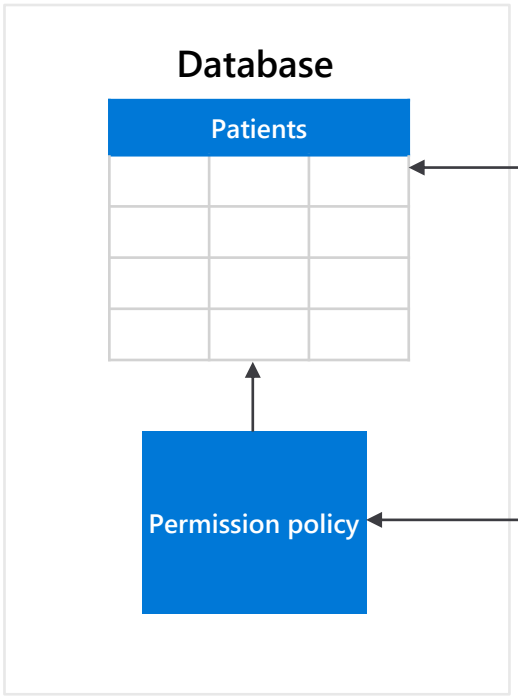
Three steps:

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

Policy manager



Nurse




```
CREATE TABLE Patients (
  PatientID int IDENTITY,
  FirstName varchar(100) NULL,
  SSN char(9) NOT NULL,
  LastName varchar(100) NOT NULL,
  Phone varchar(12) NULL,
  Email varchar(100) NULL
);
```

```
GRANT SELECT ON Patients (
  PatientID, FirstName, LastName, Phone, Email
) TO Nurse;
```

Allow 'Nurse' to access all columns except for sensitive SSN column

Application

```
SELECT * FROM Membership;

Msg 230, Level 14, State 1, Line 12
The SELECT permission was denied on the column
'SSN' of the object 'Membership', database
'CLS_TestDW', schema 'dbo'.
```

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

Dynamic Data Masking

Overview

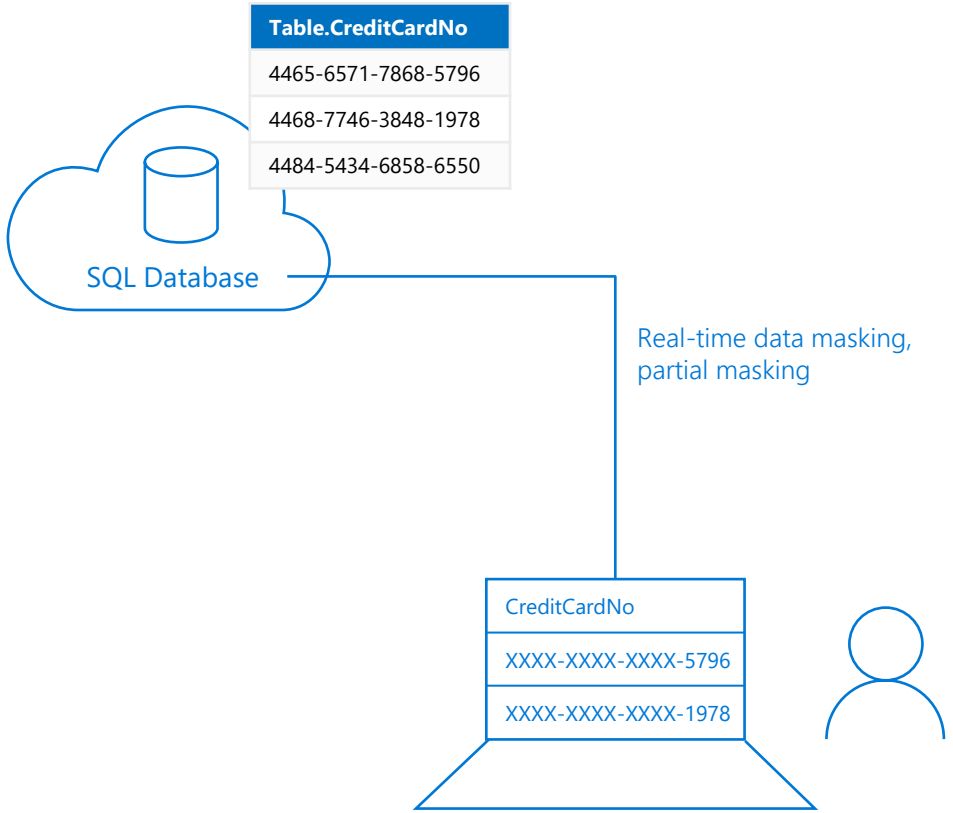
Prevent abuse of sensitive data by hiding it from users

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

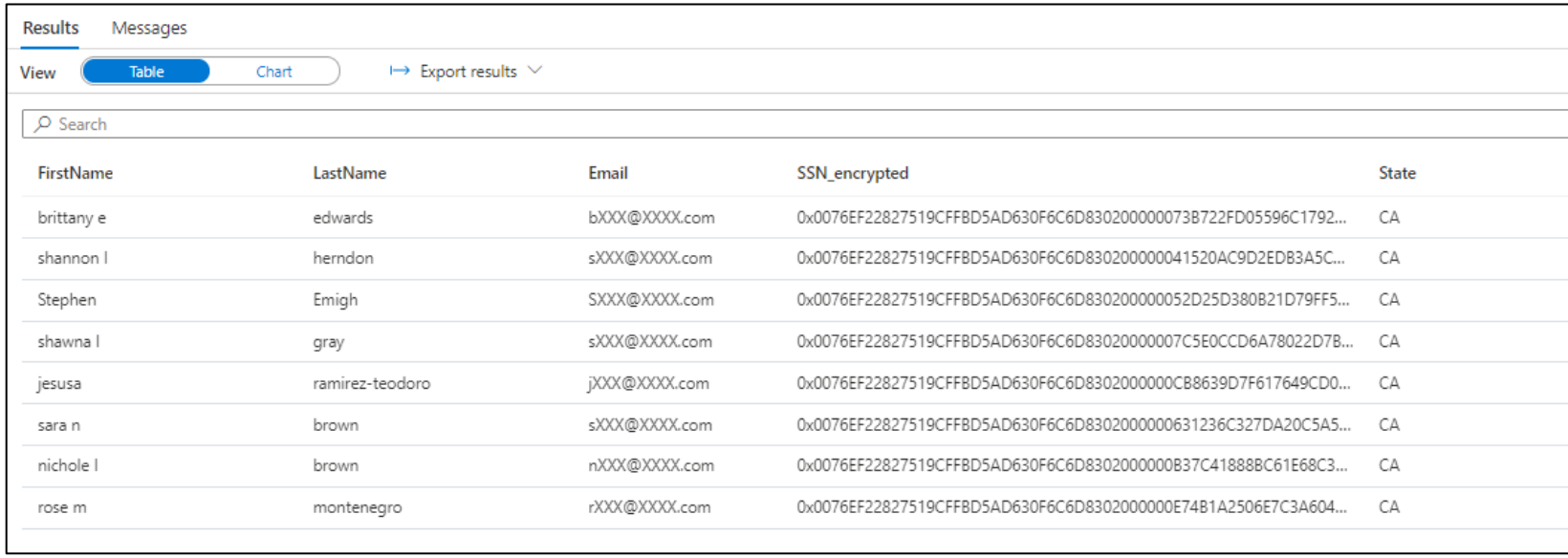
Overview

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

Dynamic Data Masking

Three steps

1. Security officer defines dynamic data masking policy in T-SQL over sensitive data in the Employee table. The security officer uses the built-in masking functions (default, email, random)

2. The app-user selects from the Employee table

3. The dynamic data masking policy obfuscates the sensitive data in the query results for non-privileged users

Security officer

1

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

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

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

GRANT UNMASK to admin1
```

2

Business app

```
SELECT [First Name],
       [Social Security Number],
       [Email],
       [Salary]
FROM   [Employee]
```

3

Non-masked data (admin login)

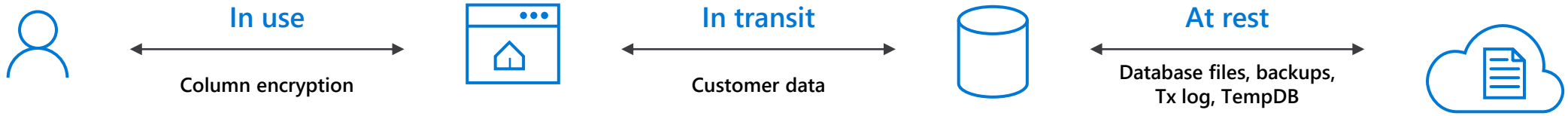
	First Name	Social Security Num...	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net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

Masked data (admin1 login)

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

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.

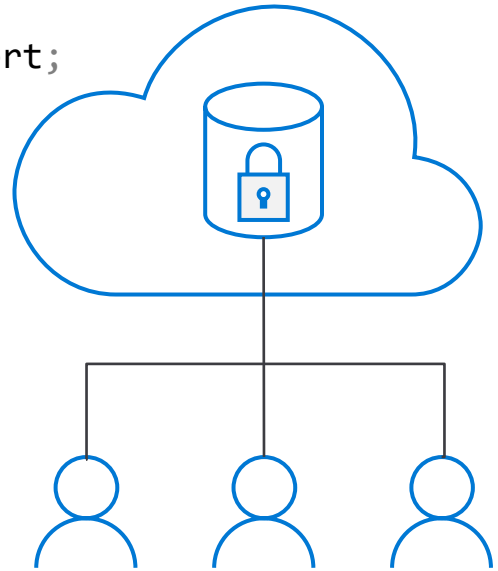
Service OR User managed keys.

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

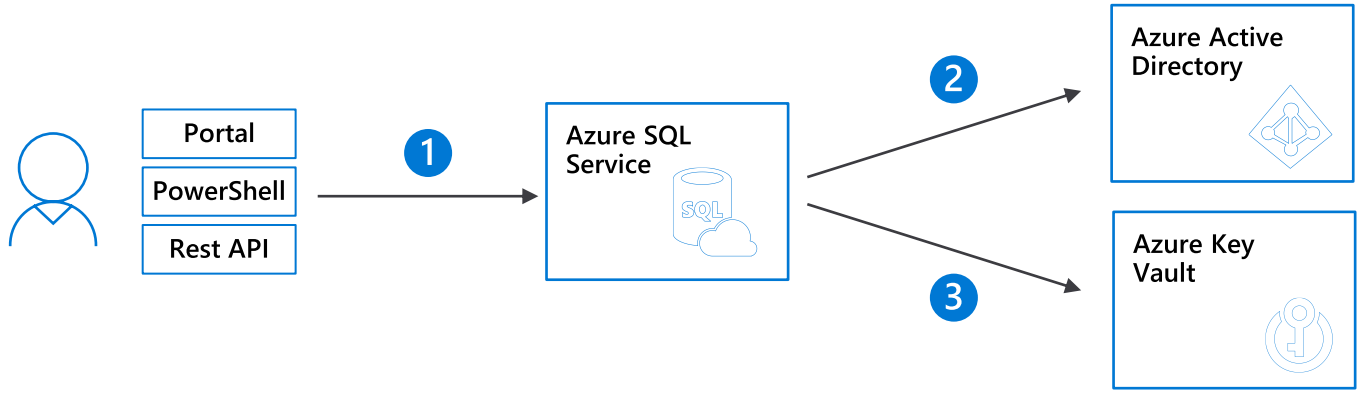
Benefits with User Managed Keys

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 Community

Azure Synapse Community Resources

Community Call to Action

- ☑ Follow us on Twitter: @Azure_Synapse
- ☑ Read and comment on our blog: https://aka.ms/SynapseBlog
- ☑ Check monthly updates blog: https://aka.ms/SynapseMonthlyUpdate
- ☑ Subscribe to YouTube channel: https://aka.ms/SynapseYouTube
- ☑ Share ideas and vote for them: https://aka.ms/SynapseIdeas
- ☑ Join Azure Synapse Influencers program: https://aka.ms/SynapseInfluencers

Product

- Product page: https://aka.ms/Synapse
- Documentation: https://aka.ms/SynapseDocs
- Reference architectures: https://aka.ms/SynapseArchitectures
- Security whitepaper: http://aka.ms/SynapseSecurity

Technical Discussions and Q&A

- Microsoft Q&A: https://aka.ms/SynapseQuestions
- StackOverflow: https://aka.ms/SynapseStackOverflow

Quickstart

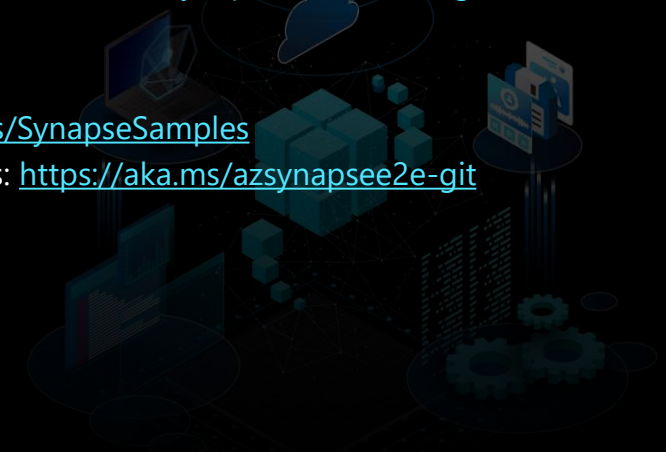
- Get started in 60 minutes: https://aka.ms/SynapseGetStarted
- Azure Synapse Analytics Toolkit: https://aka.ms/SynapseToolkit

Free Learning

- MS Learn – learning paths: https://aka.ms/SynapseLearningPaths
- Synapse Practitioner: https://aka.ms/SynapsePractitioner
- Microsoft Virtual Training Days: https://aka.ms/SynapseMVTD
- 30-Day Cloud Skills Challenge: https://aka.ms/SynapseSkillsChallenge

Samples & Accelerators

- Samples on GitHub: https://aka.ms/SynapseSamples
- Accelerator – End-to-End Analytics: https://aka.ms/azsynapsee2e-git



Microsoft

Modern Data Warehouse Using Azure Synapse Analytics

Pawel Potasinski
Senior Program Manager

@pawelpotasinski
/in/pawelpotasinski

Get this slide deck from:
https://github.com/pawelpo/presentations