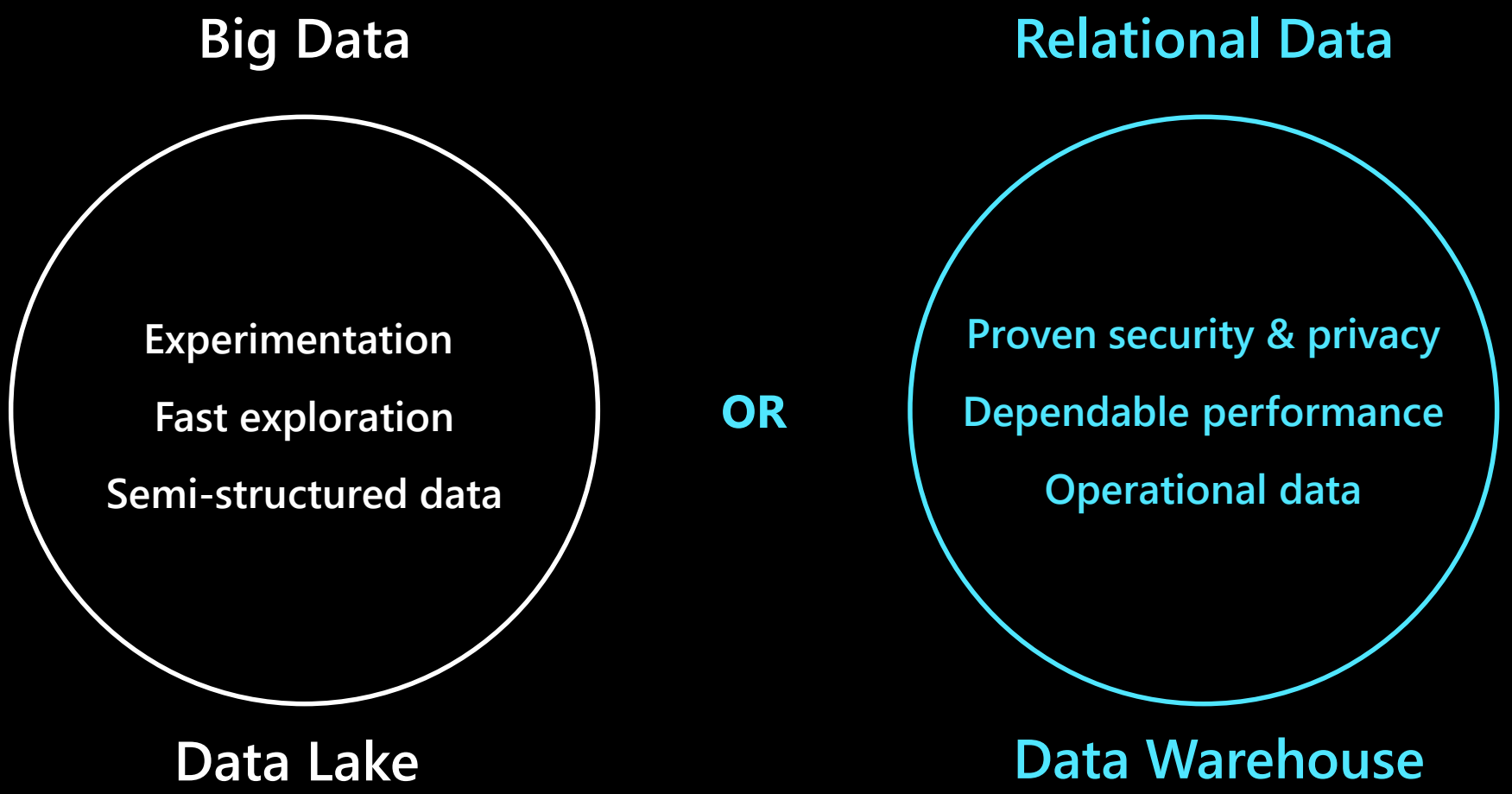
Azure Synapse Analytics

Andrea Benedetti

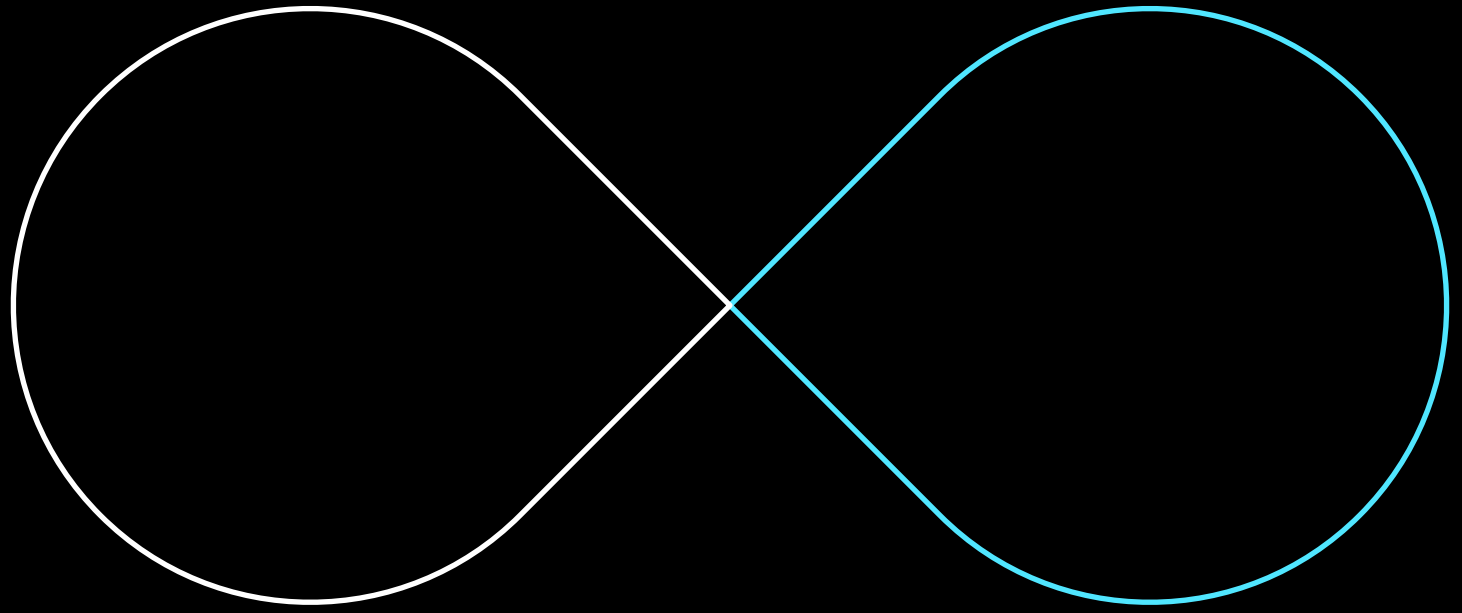
Sr. Cloud Architect / Data & AI Engineer | Microsoft Italia

/in/abenedetti @anBenedetti https://github.com/anbened

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



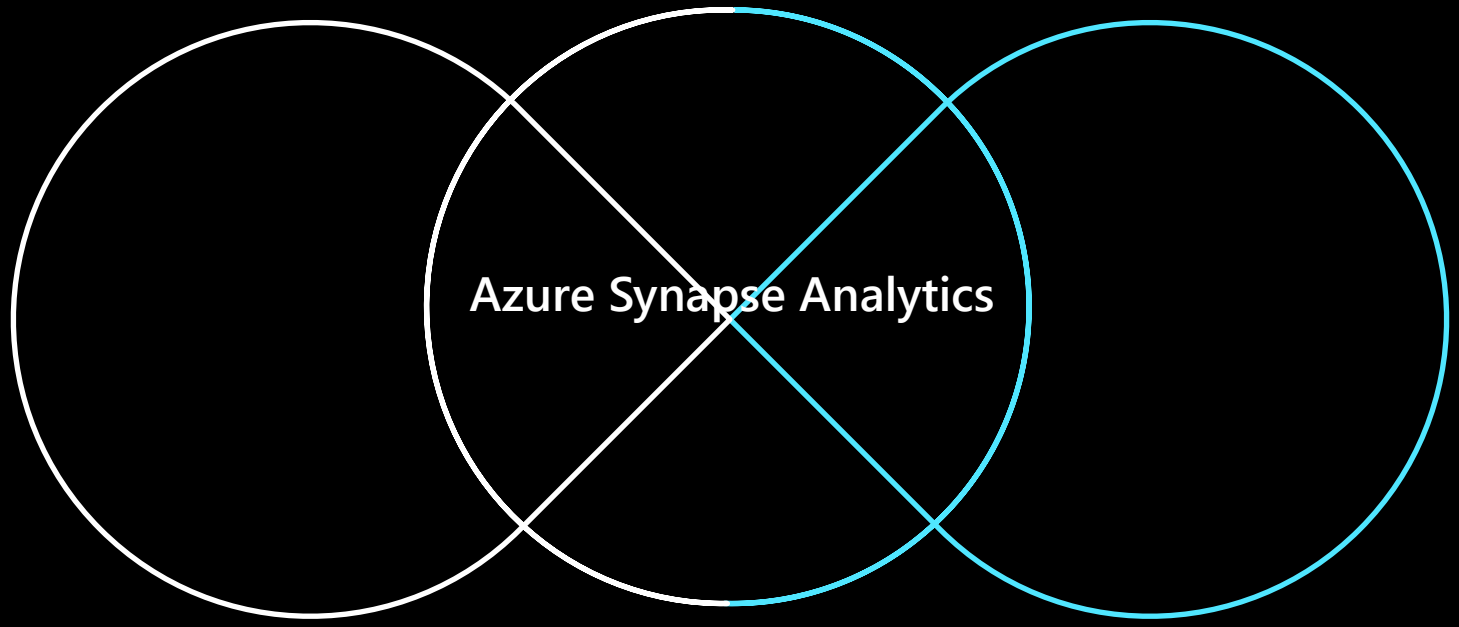
Azure brings these two worlds together, in a single service, to provide limitless analytics



Welcome to limitless

Data warehousing & big data analytics—all in one service

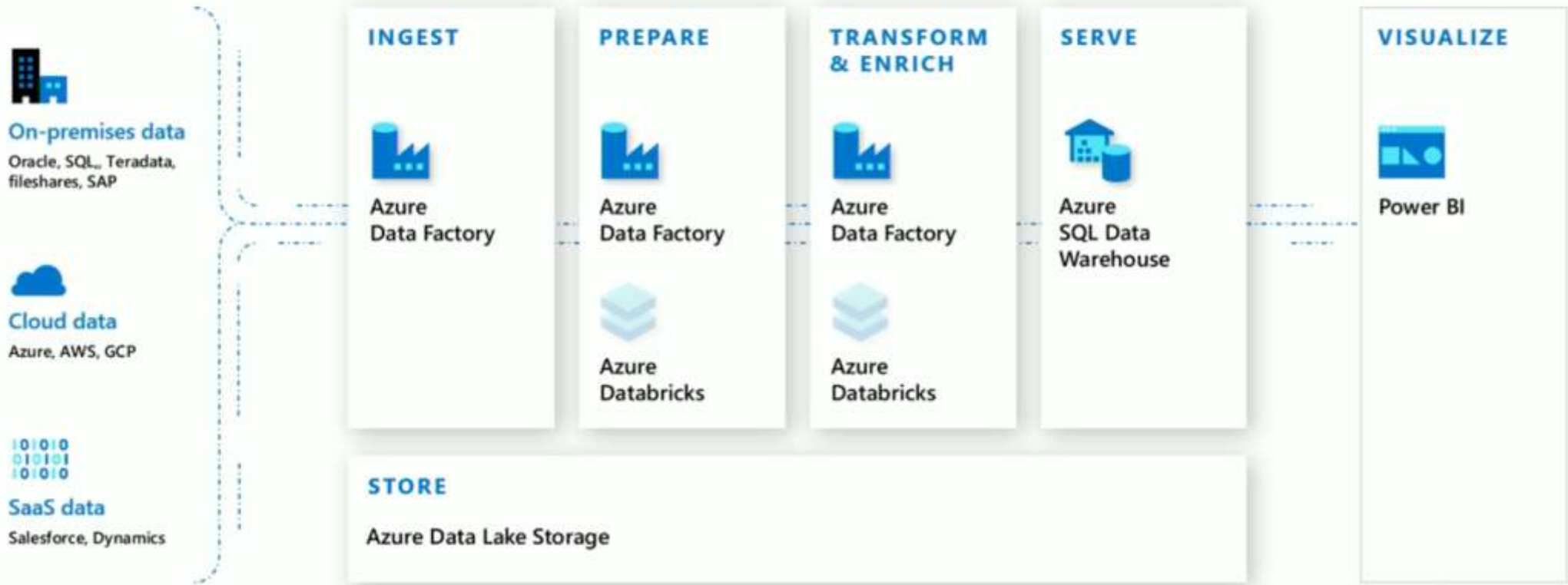
Azure brings these two worlds together, in a single service, to provide limitless analytics



Welcome to limitless

Data warehousing & big data analytics—all in one service

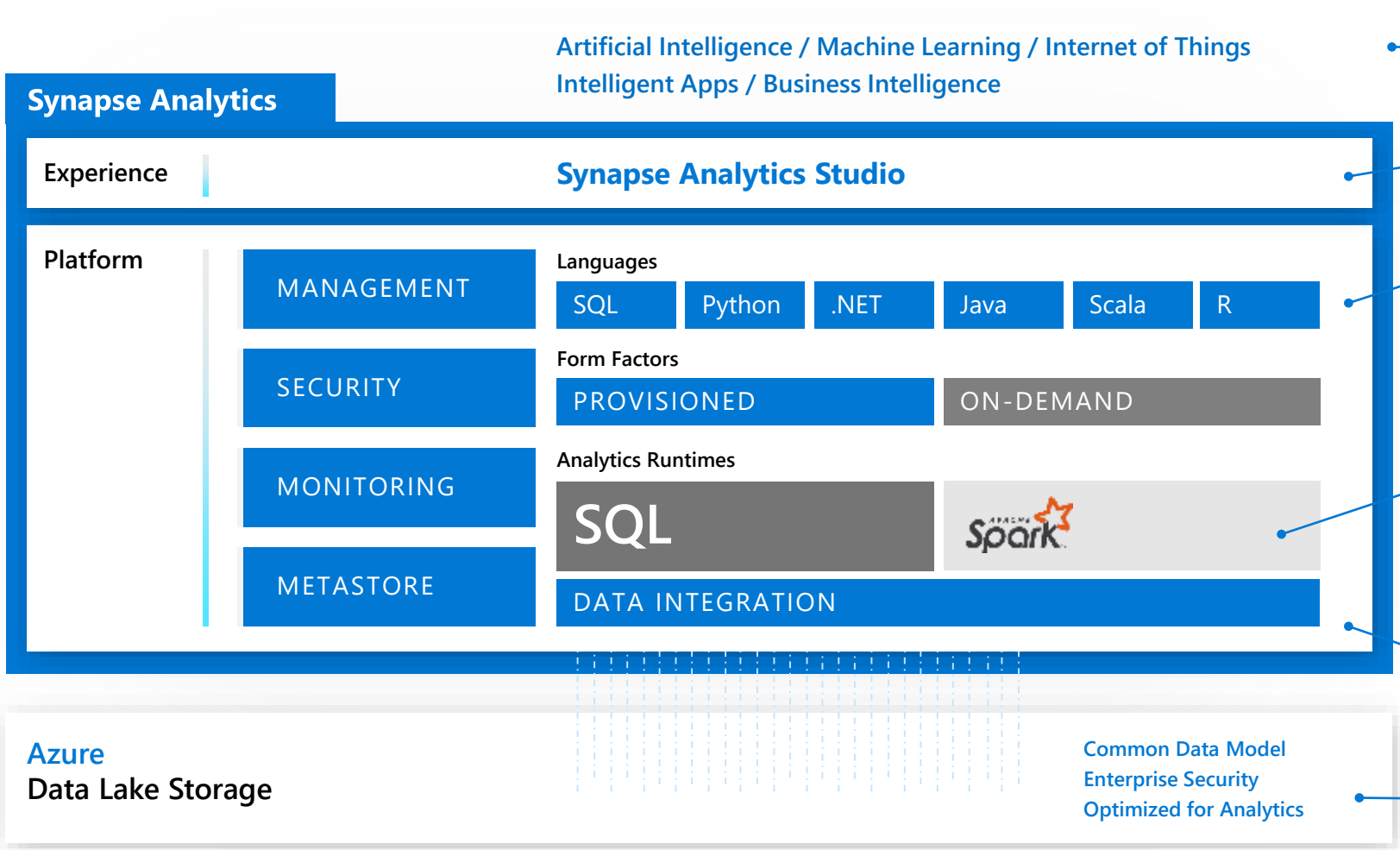
Modern Data Warehouse



Azure Synapse Analytics

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available provisioned and serverless on-demand

SQL Analytics offering T-SQL for batch, streaming and interactive processing

Spark for big data processing with Python, Scala, R and .NET

Integrated **platform services** for, management, security, monitoring, and metastore

Data **lake integrated** and Common Data Model aware

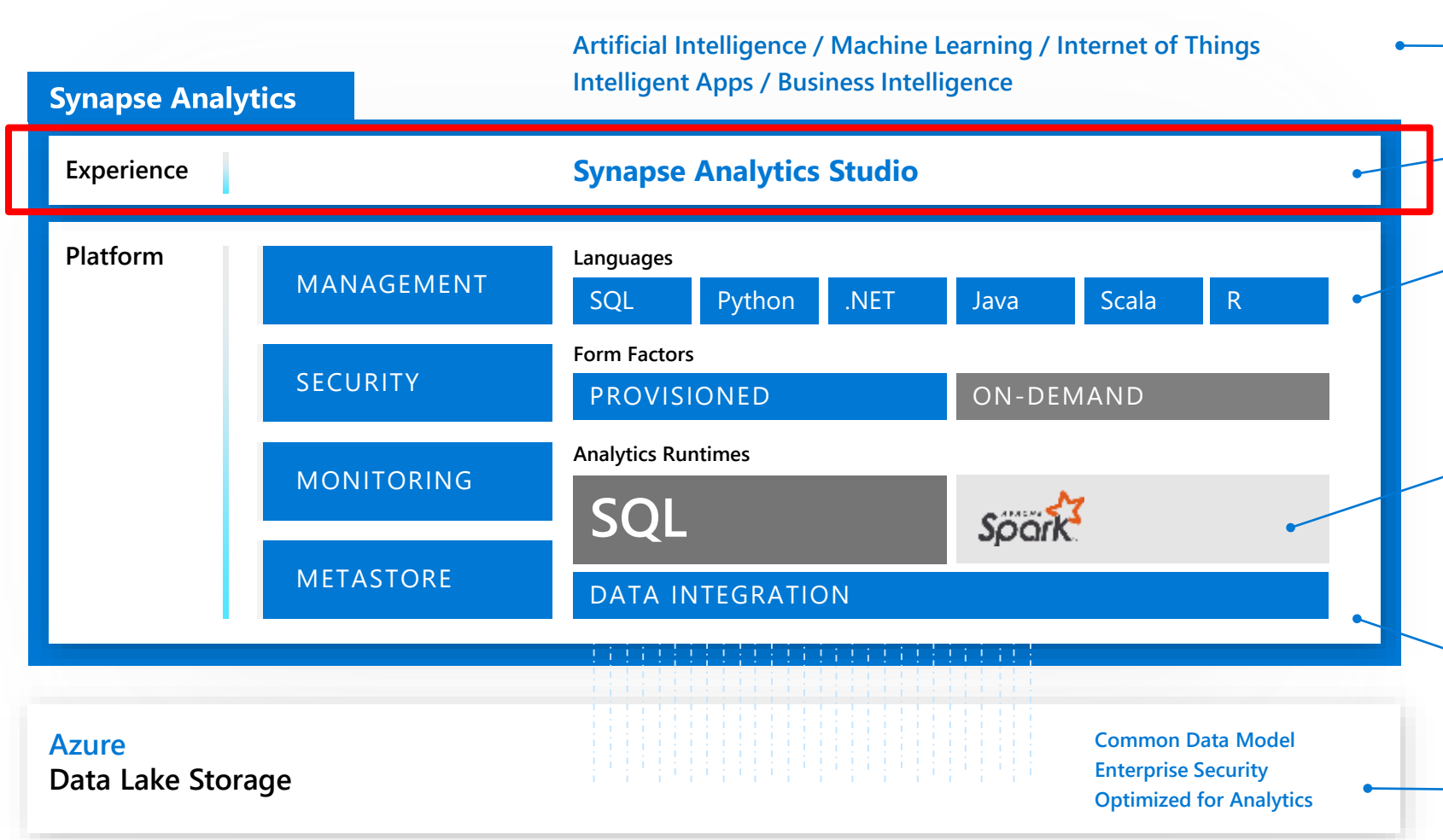
Microsoft

Azure Synapse Analytics

Studio

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available provisioned and serverless on-demand

SQL Analytics offering T-SQL for batch, streaming and interactive processing

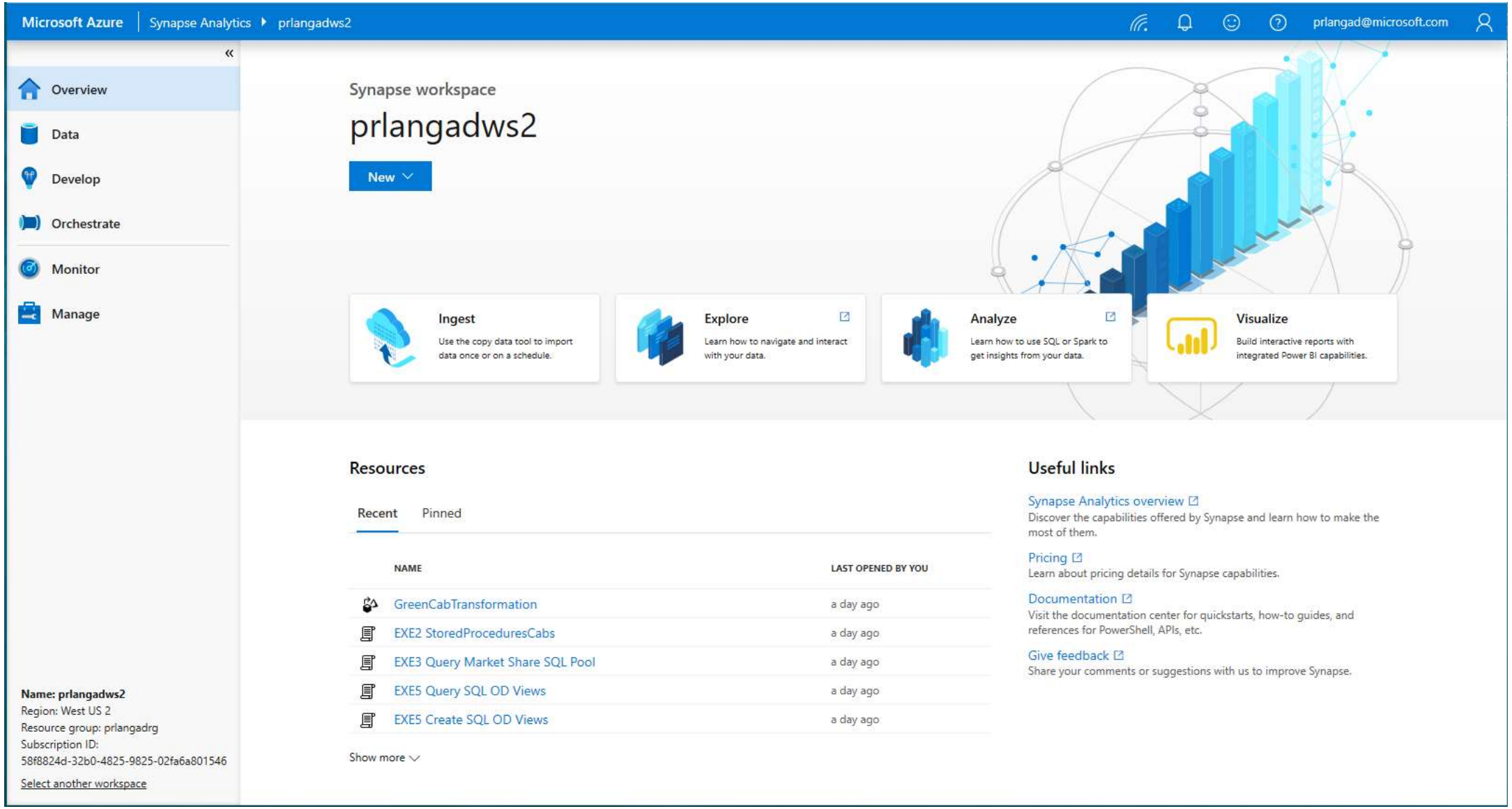
Spark for big data processing with Python, Scala, R and .NET

Integrated **platform services** for, management, security, monitoring, and metastore

Data **lake integrated** and Common Data Model aware

Studio

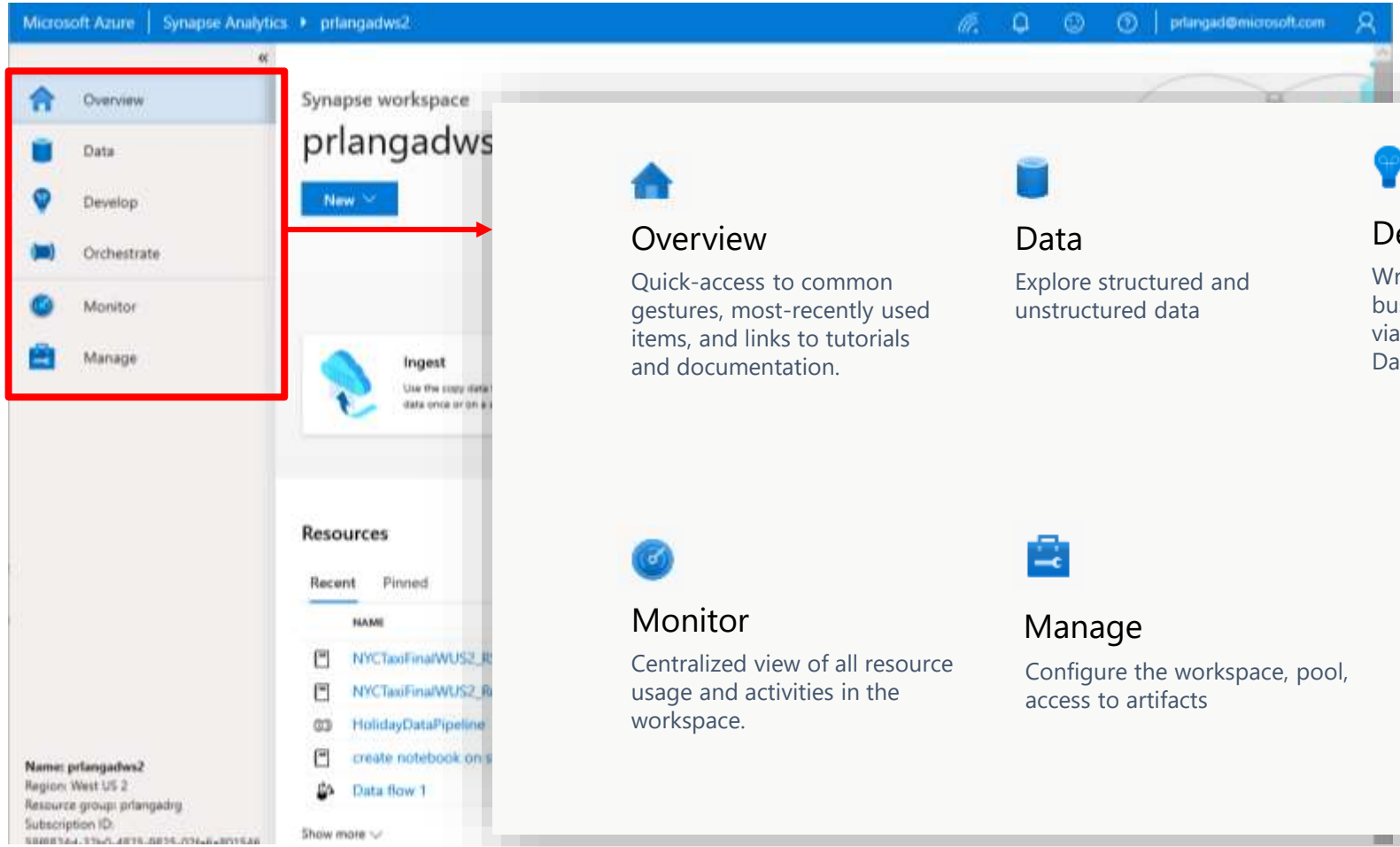
A single place for Data Engineers, Data Scientists, and IT Pros to collaborate on enterprise analytics



Synapse Studio

Synapse Studio divided into **Activity hubs**.

These organize the tasks needed for building analytics solution.



Overview

Quick-access to common gestures, most-recently used items, and links to tutorials and documentation.

Data

Explore structured and unstructured data

Develop

Write code and the define business logic of the pipeline via notebooks, SQL scripts, Data flows, etc.

Orchestrate

Design pipelines that that move and transform data.

Monitor

Centralized view of all resource usage and activities in the workspace.

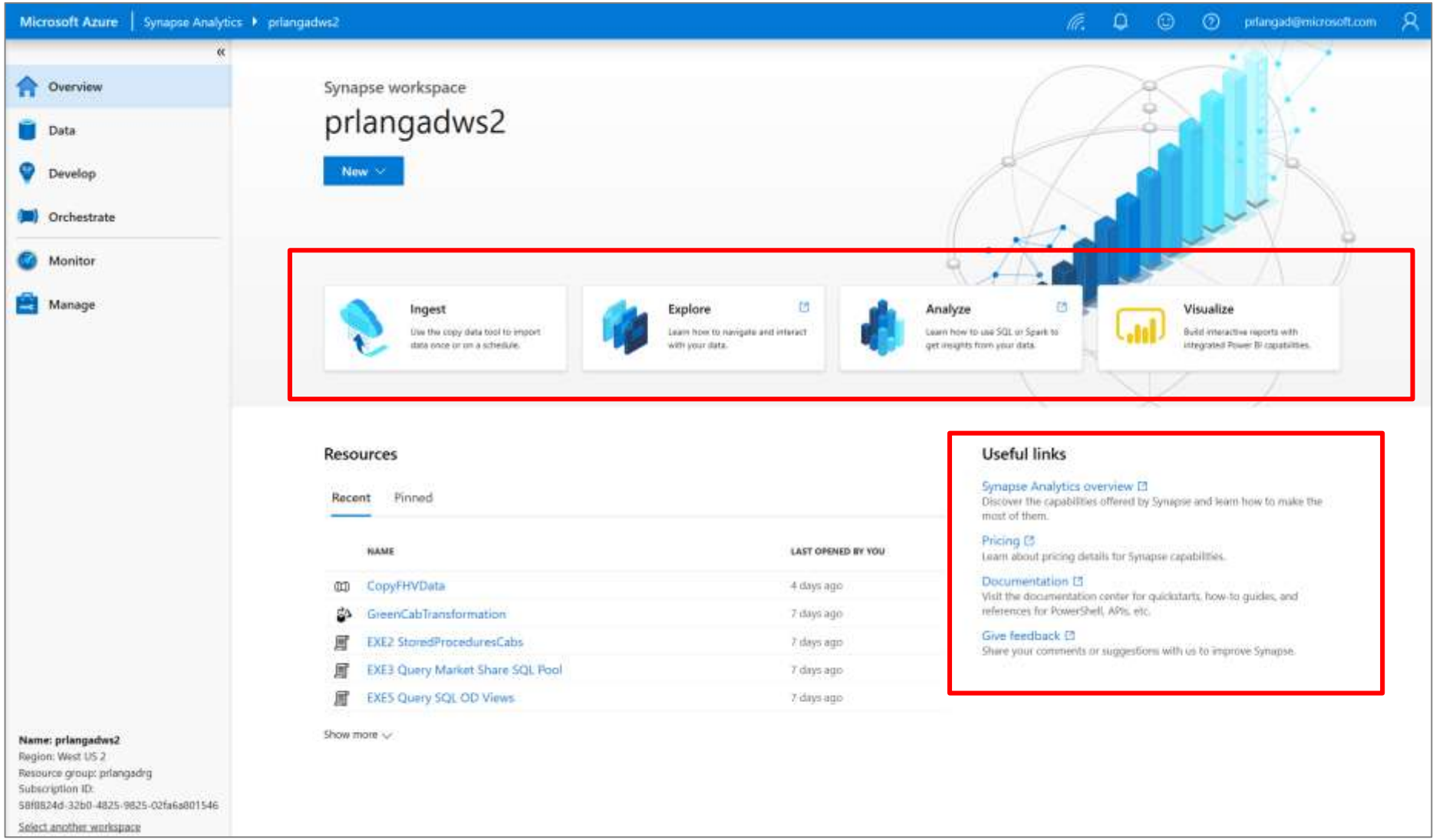
Manage

Configure the workspace, pool, access to artifacts

Synapse Studio
Overview hub

Overview Hub

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

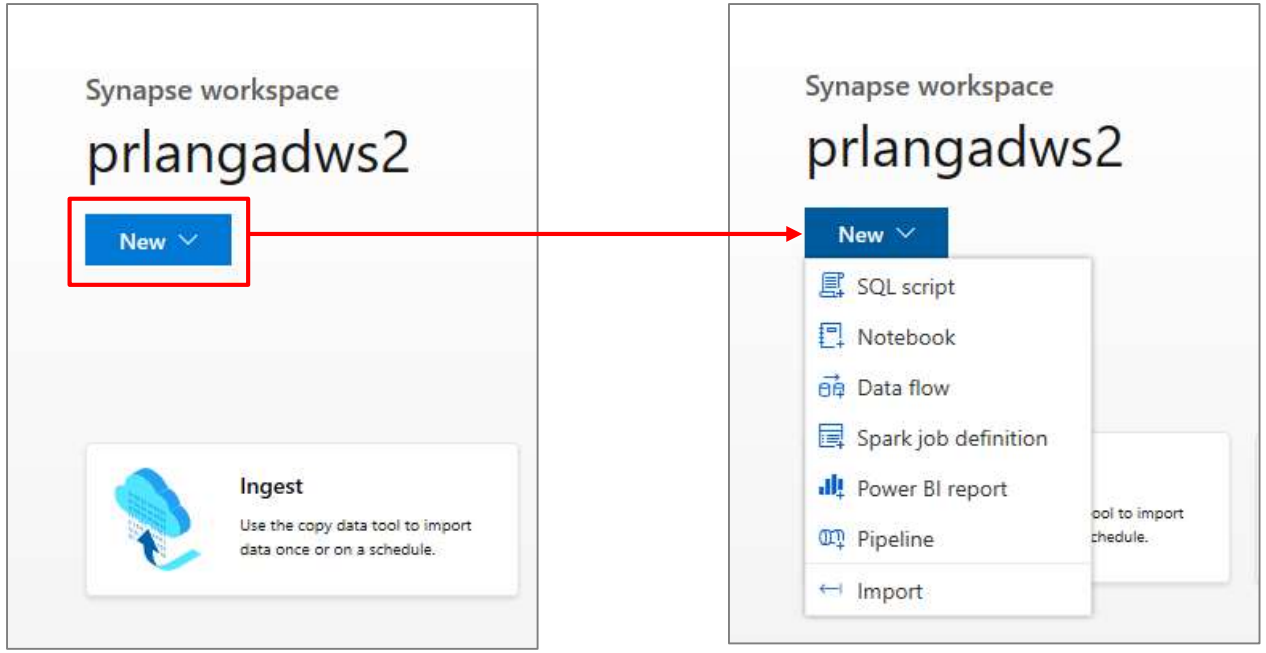


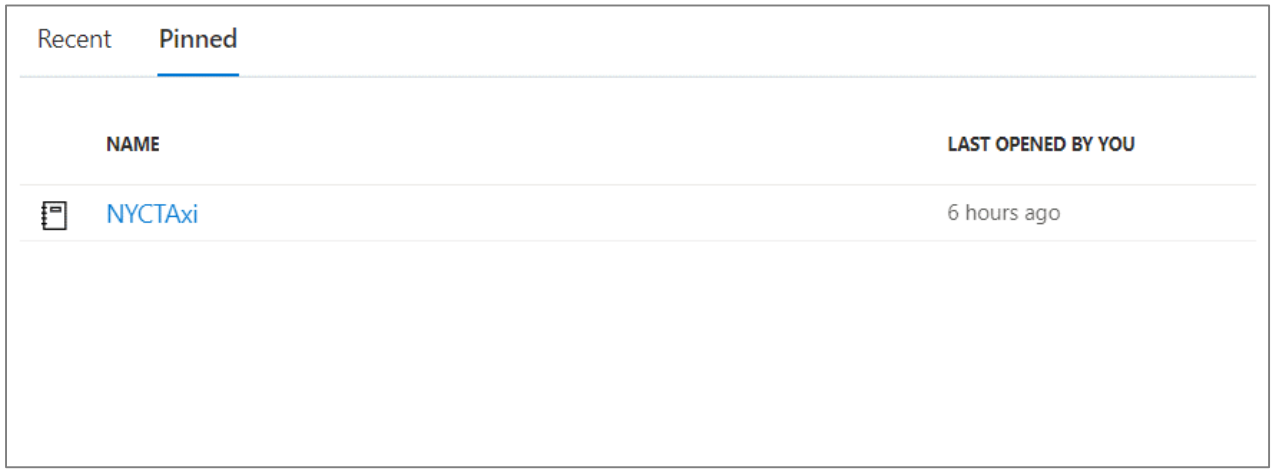
Overview Hub

Overview

New dropdown – offers quickly start work item

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

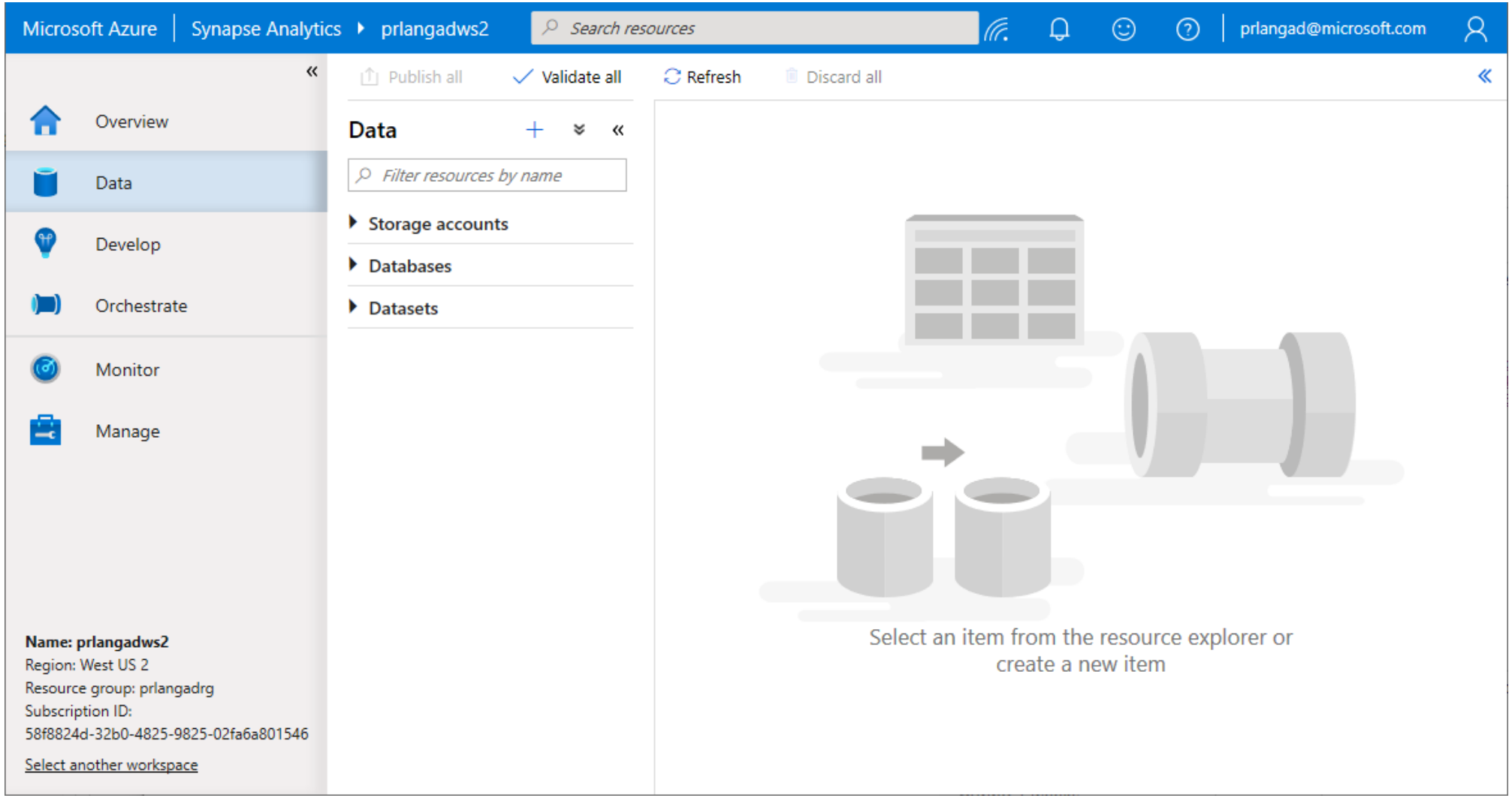




Synapse Studio
Data hub

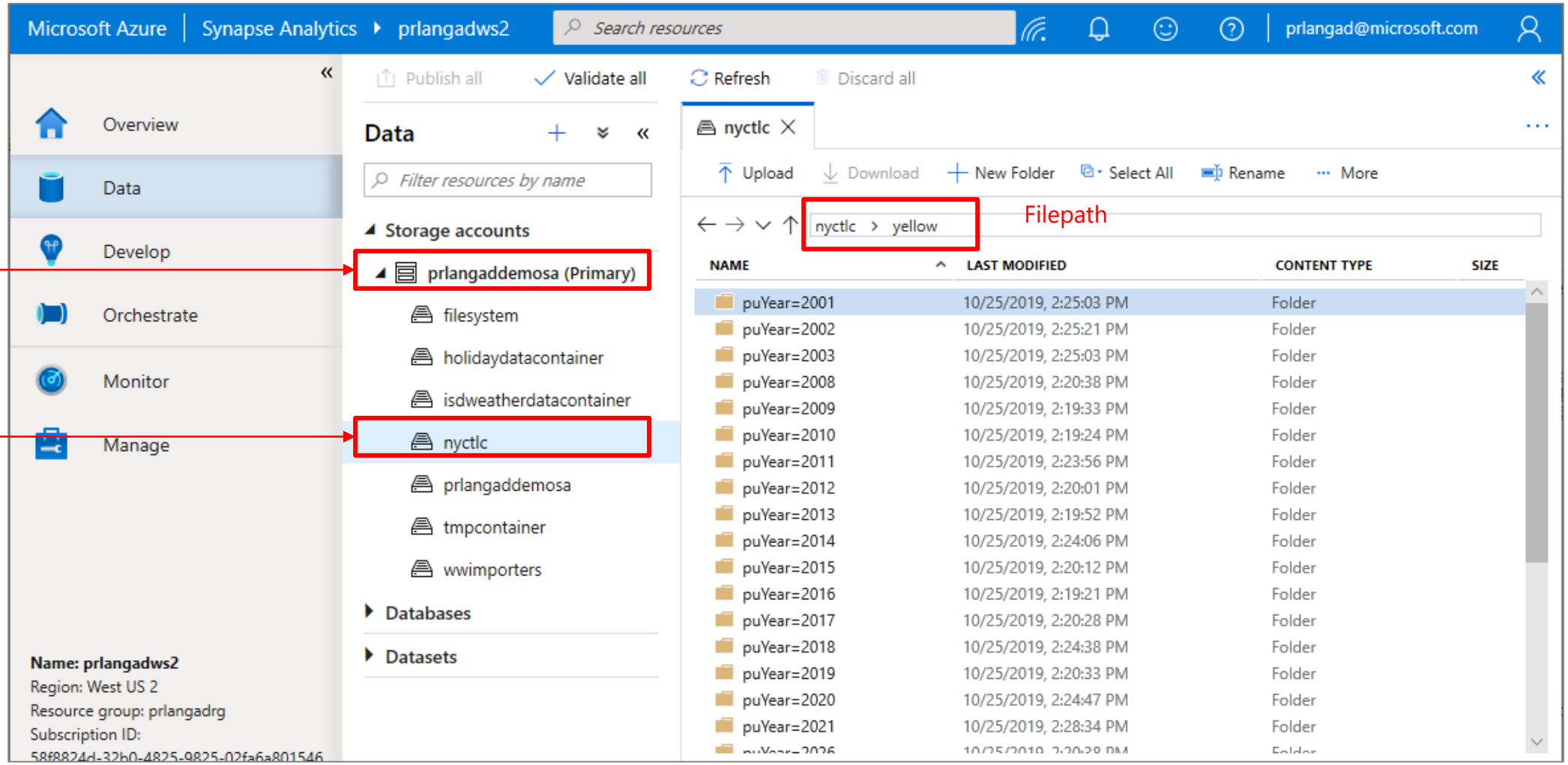
Data Hub

Explore data inside the workspace and in linked storage accounts



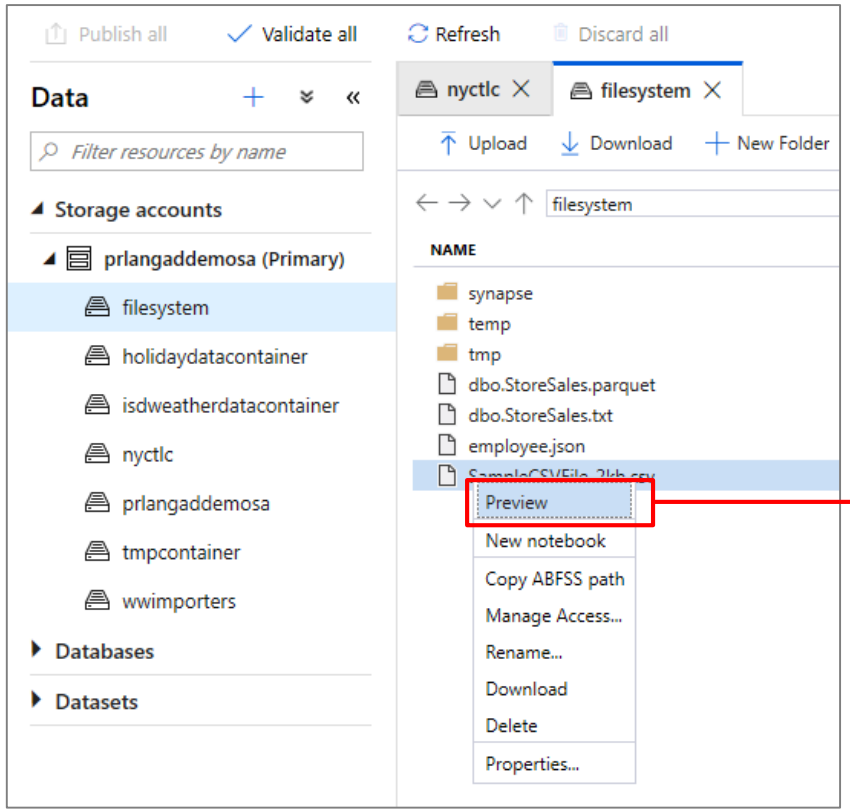
Data Hub – Storage accounts

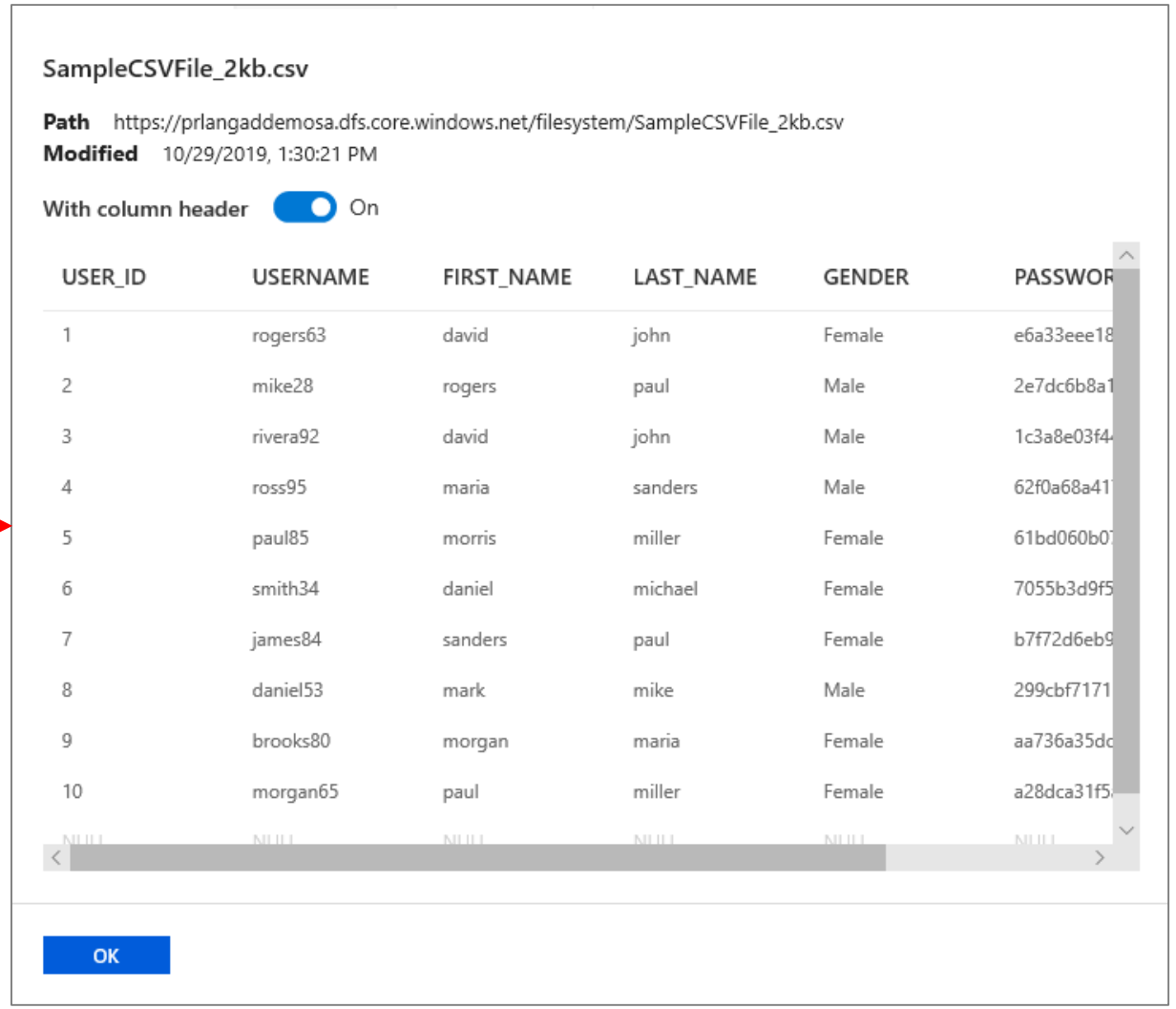
Browse Azure Data Lake Storage Gen2 accounts and filesystems – navigate through folders to see data



Data Hub – Storage accounts

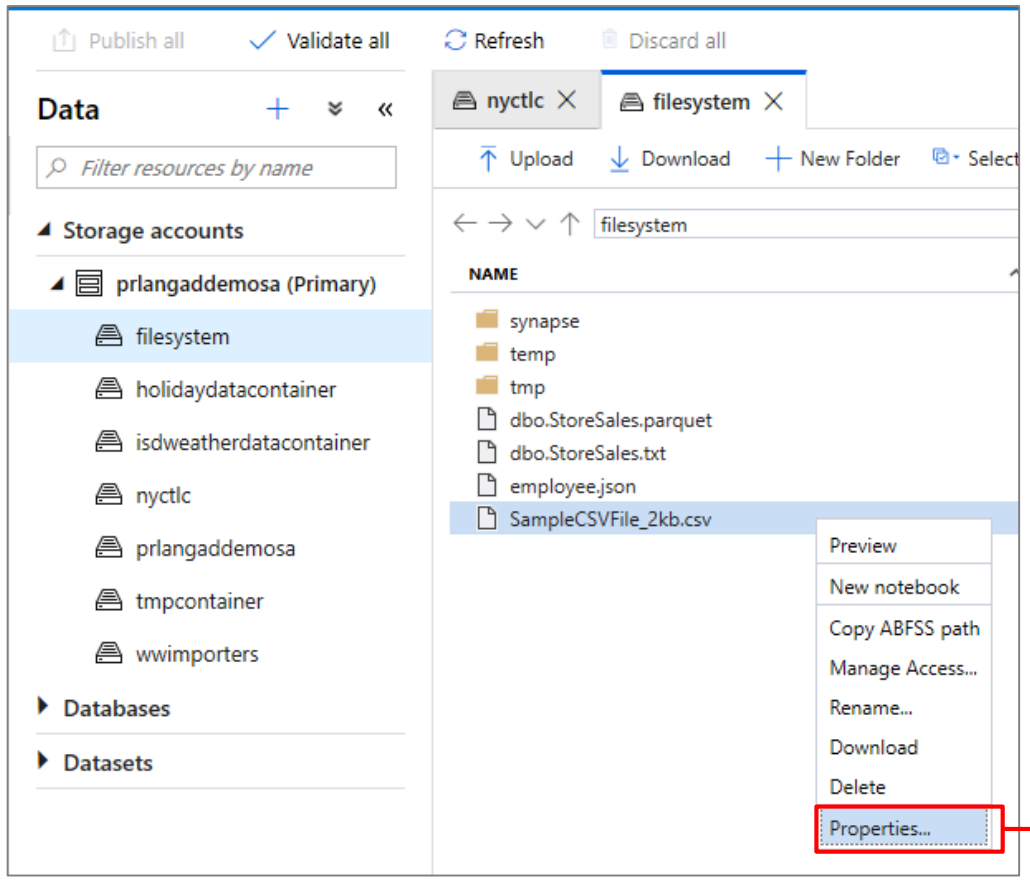
Preview a sample of your data

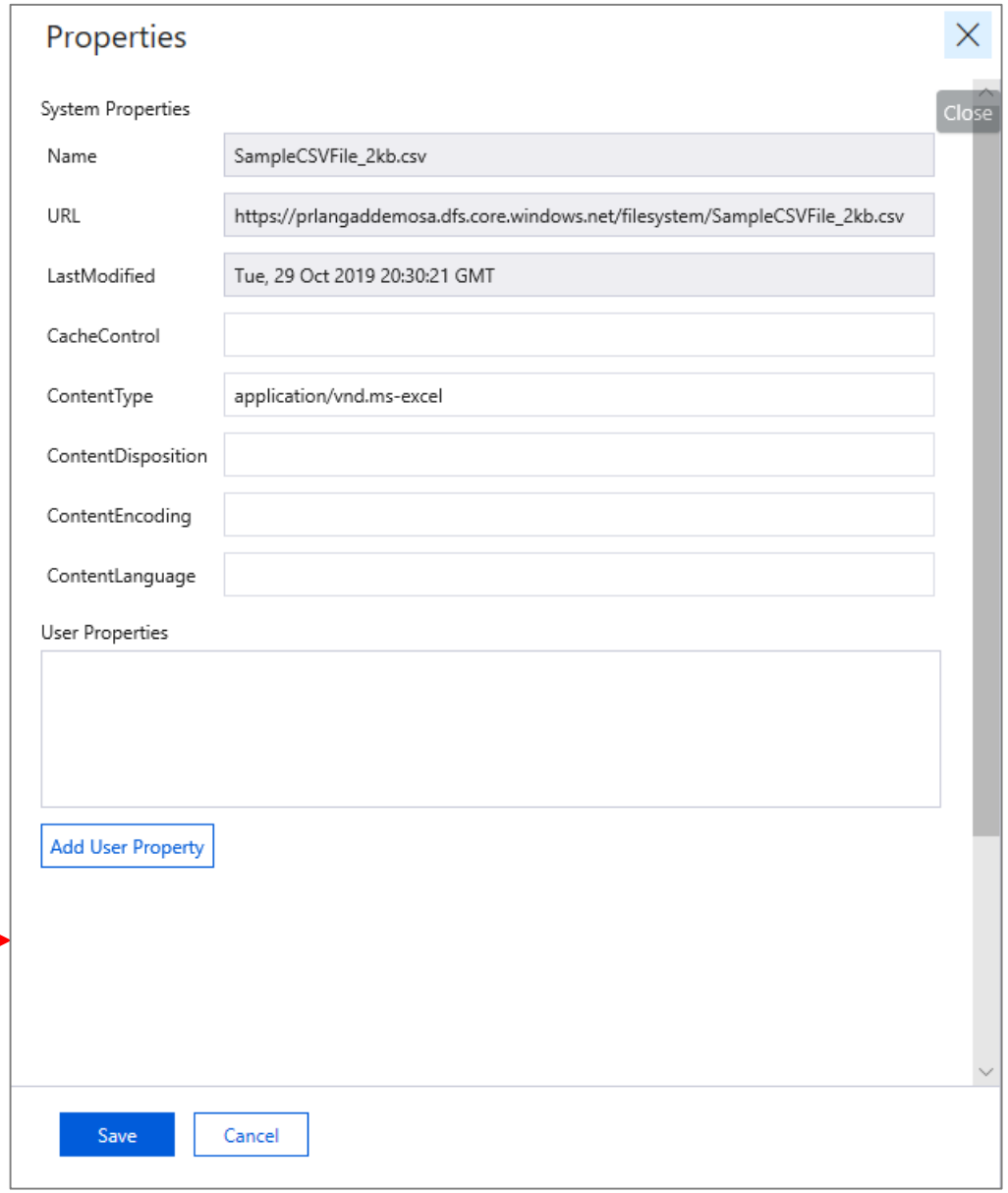




Data Hub – Storage accounts

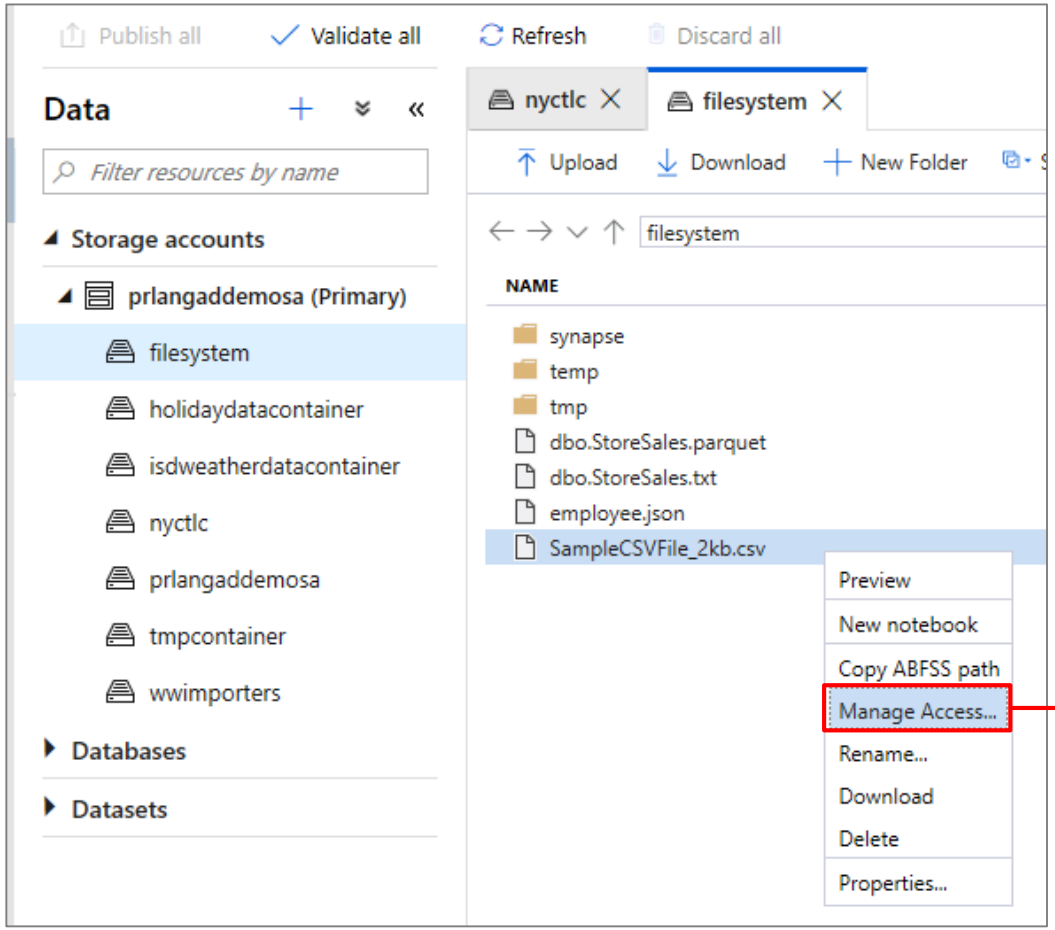
See basic file properties

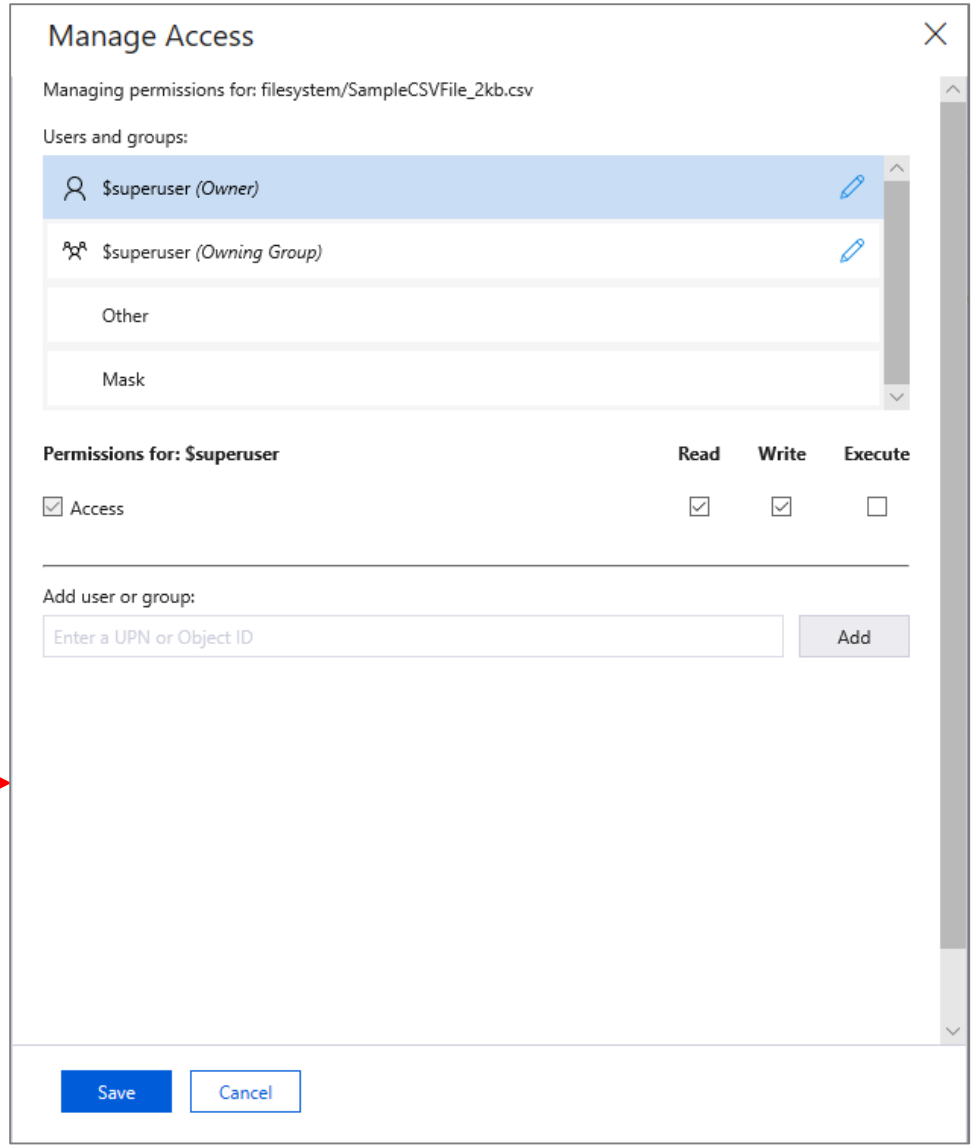




Data Hub – Storage accounts

Manage Access - Configure standard POSIX ACLs on files and folders

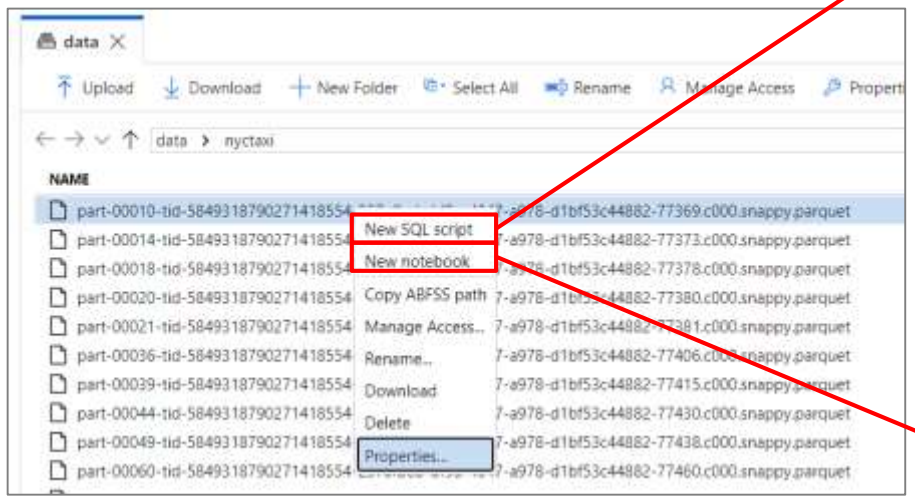


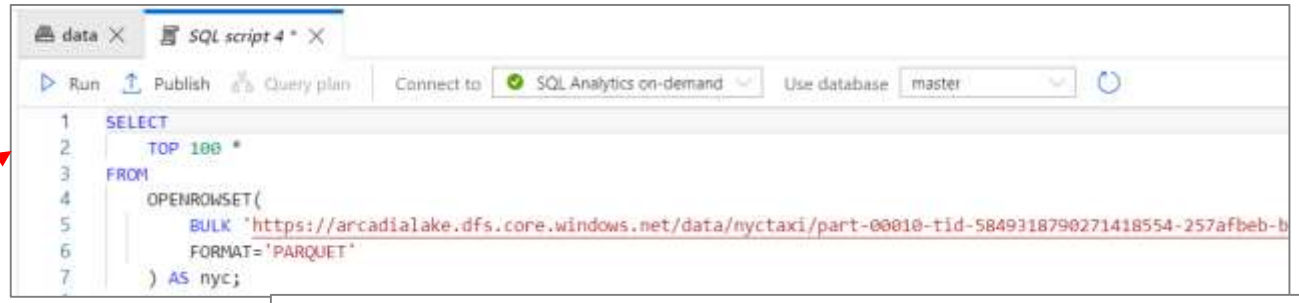


Data Hub – Storage accounts

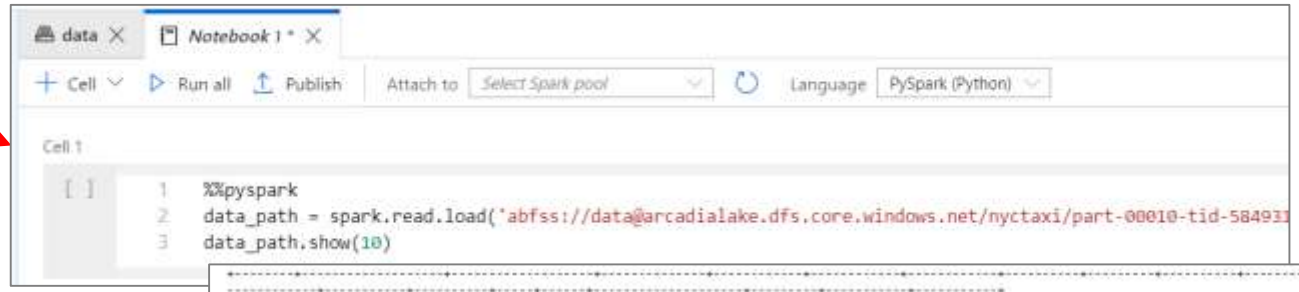
Two simple gestures to start analyzing with SQL scripts or with notebooks.

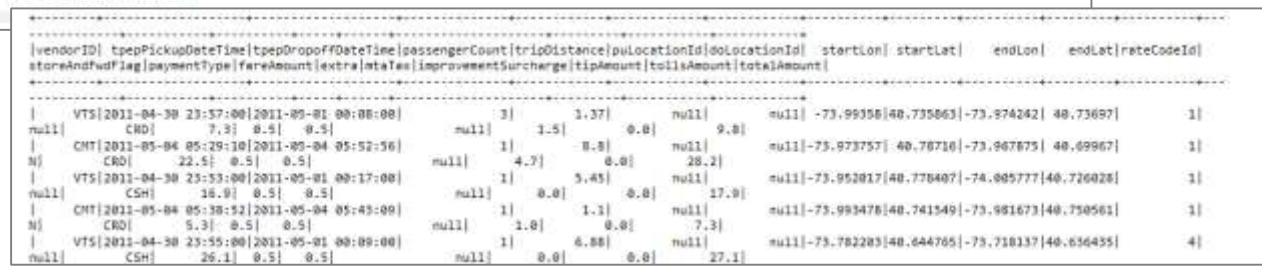
T-SQL or PySpark auto-generated.





STARTLON	STARTLAT	ENDLON	ENDLAT	RATECODEID	STOREANDFWDFLAG	PAYMENTTYPE	FAREAMOU
-73.95661	40.783125	-73.962751	40.772417	1	N	CSH	5.3
-73.998248	40.724852	-73.991723	40.715288	1	NULL	CRD	6.1
-73.961869	40.811094	-73.9683	40.78687	1	N	CSH	8.1
-74.000057	40.717537	-73.99465	40.698172	1	NULL	CSH	8.1
-73.967002	40.757073	-73.965254	40.766673	1	N	CSH	4.5

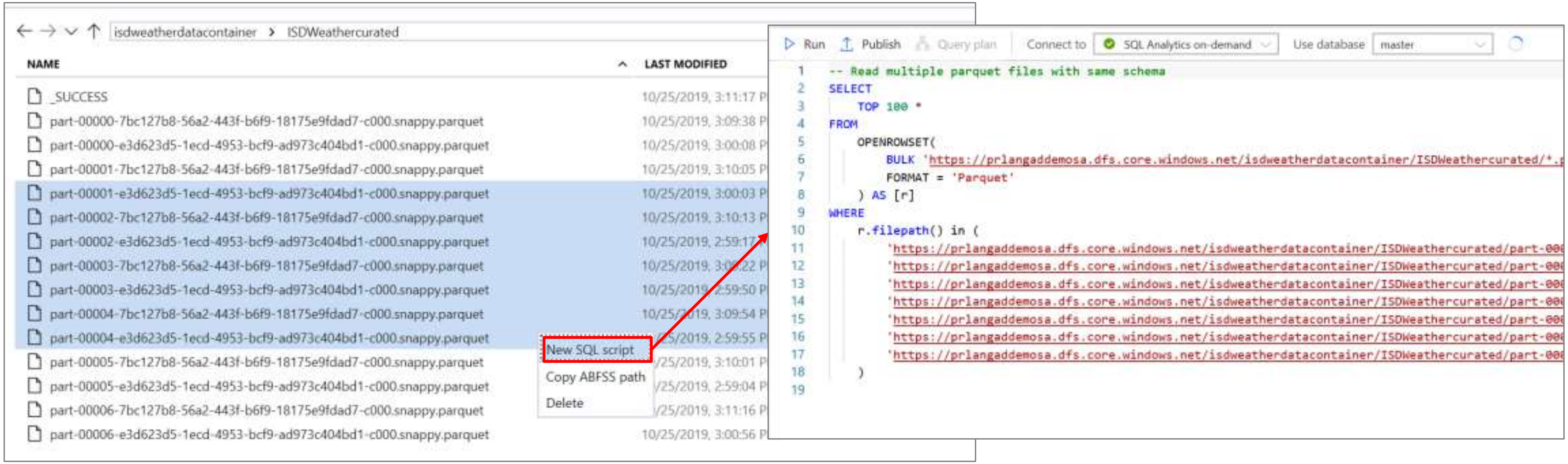




Data Hub – Storage accounts

SQL Script from Multiple files

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



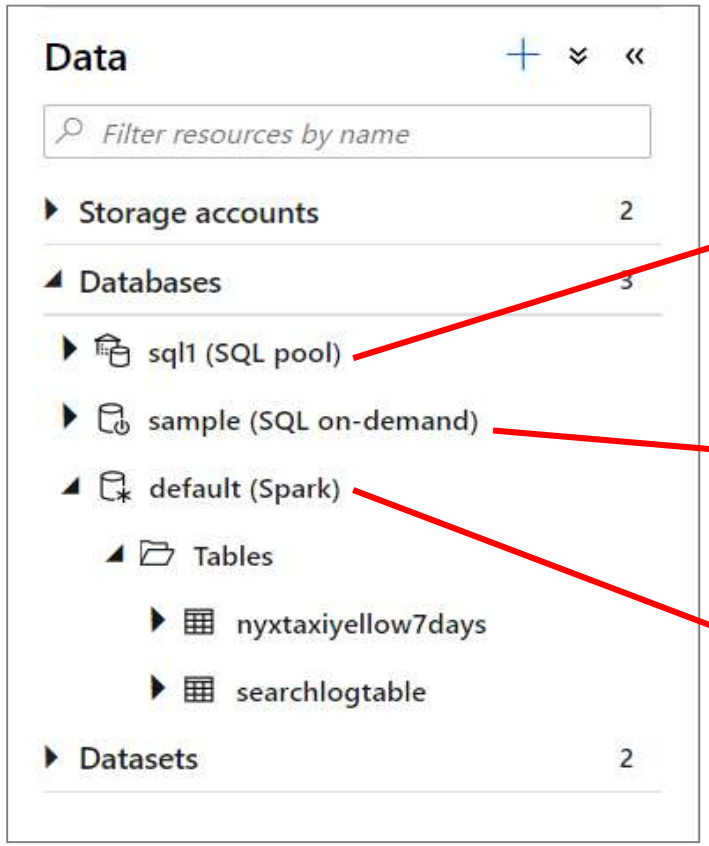
Data Hub – Databases

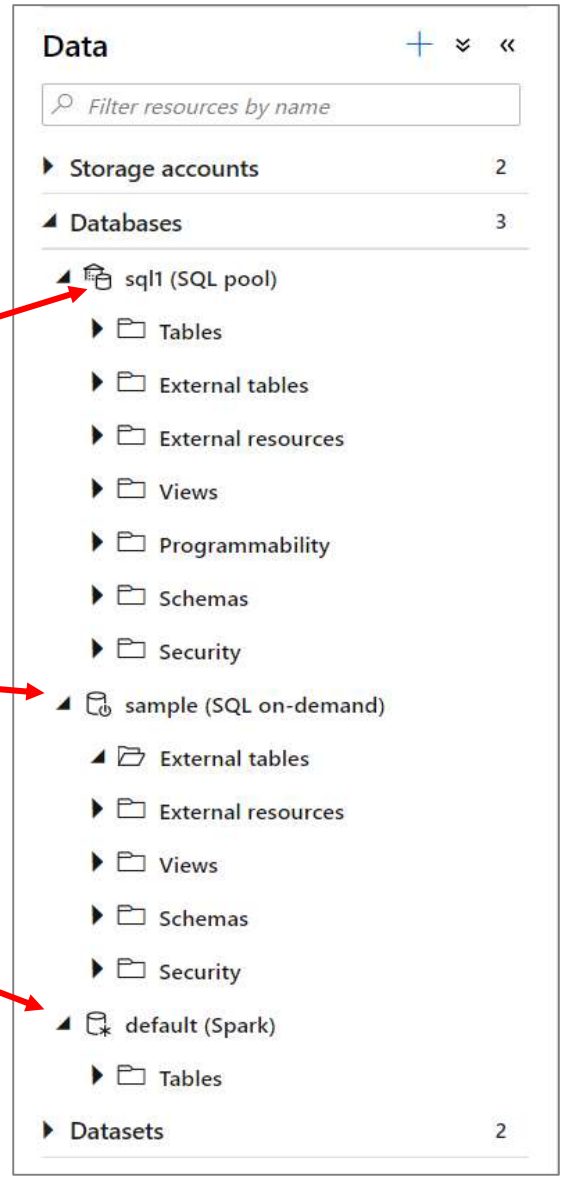
Explore the different kinds of databases that exist in a workspace.

SQL pool

SQL on-demand

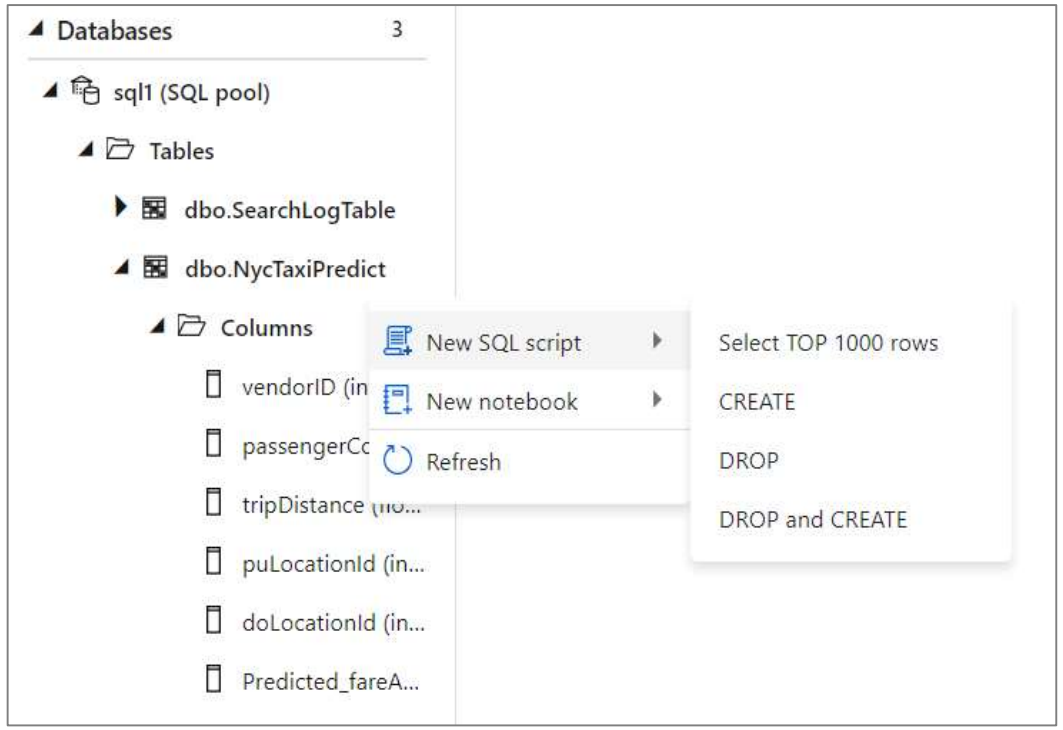
Spark



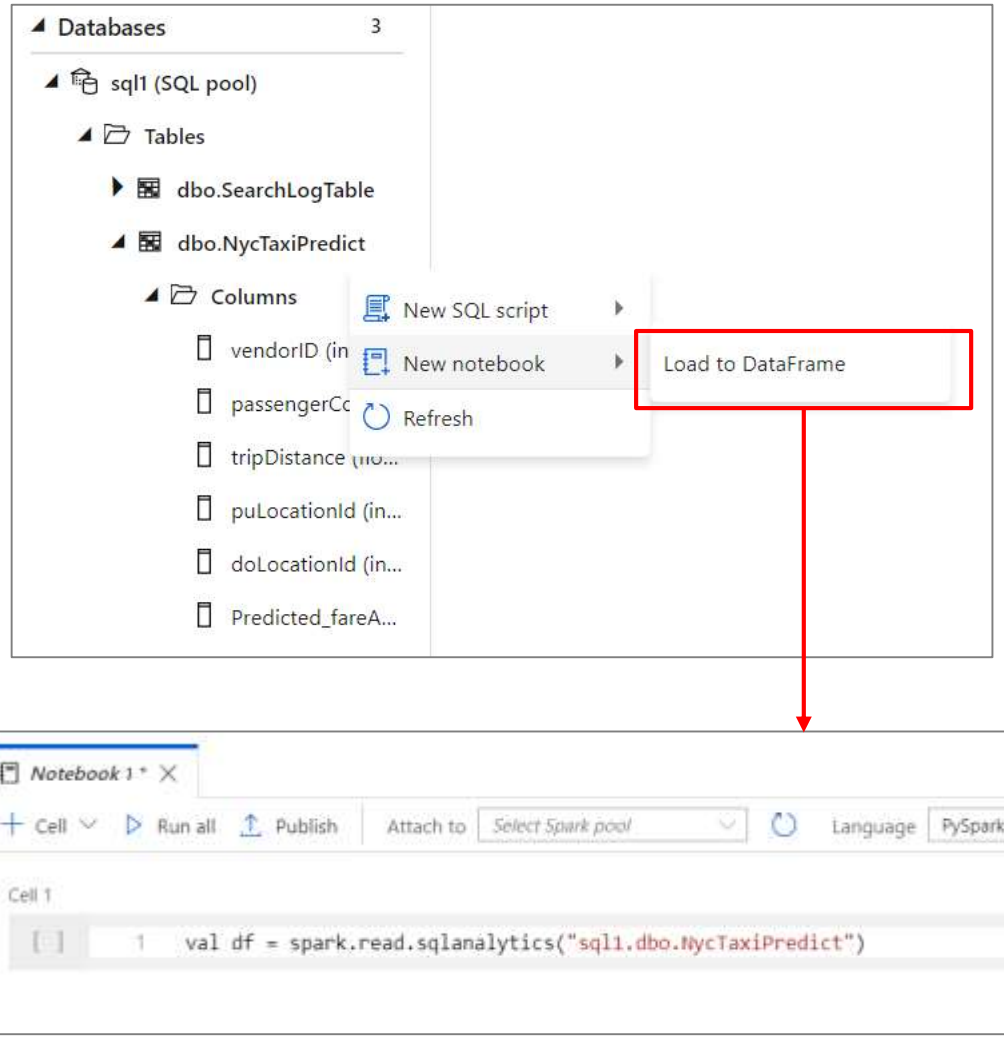


Data Hub – Databases

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

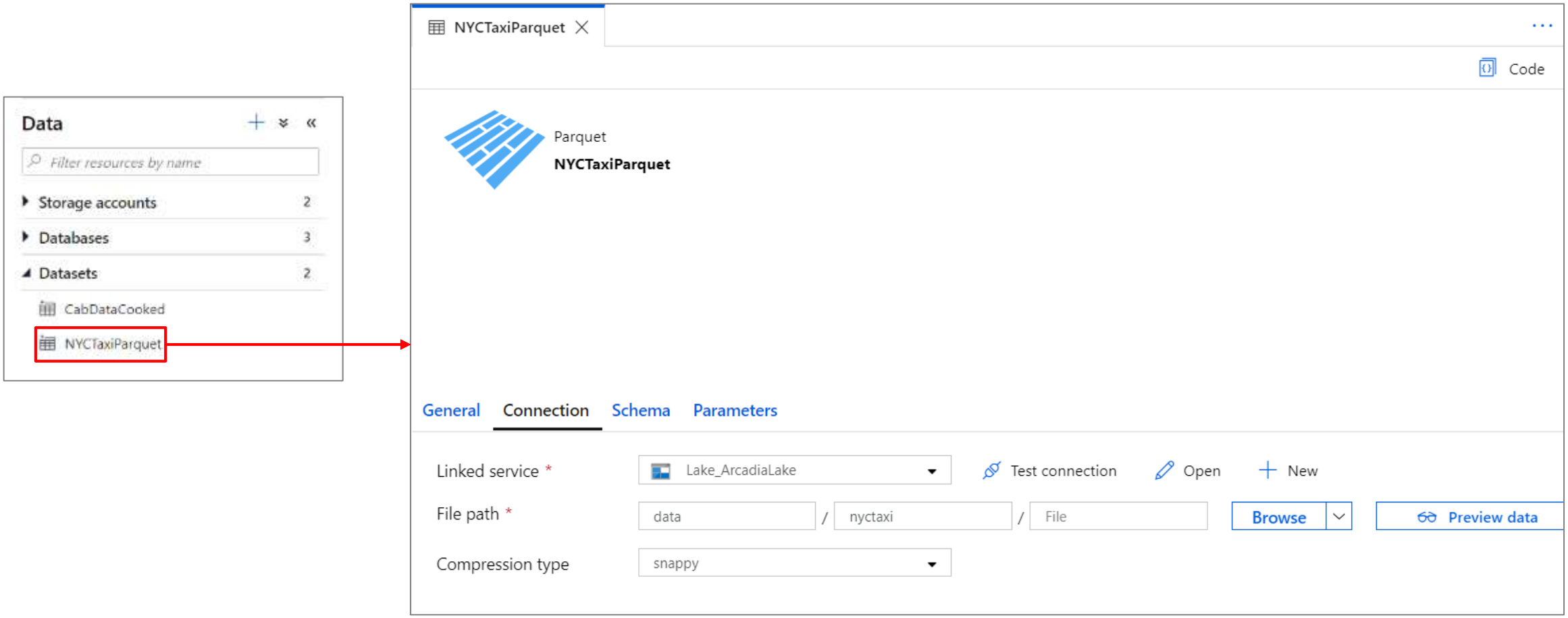


Starting from a table, auto-generate a single line of PySpark code that makes it easy to load a SQL table into a Spark dataframe



Data Hub – Datasets

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



Synapse Studio
Develop hub

Develop Hub

Overview

It provides development experience to query, analyze, model data

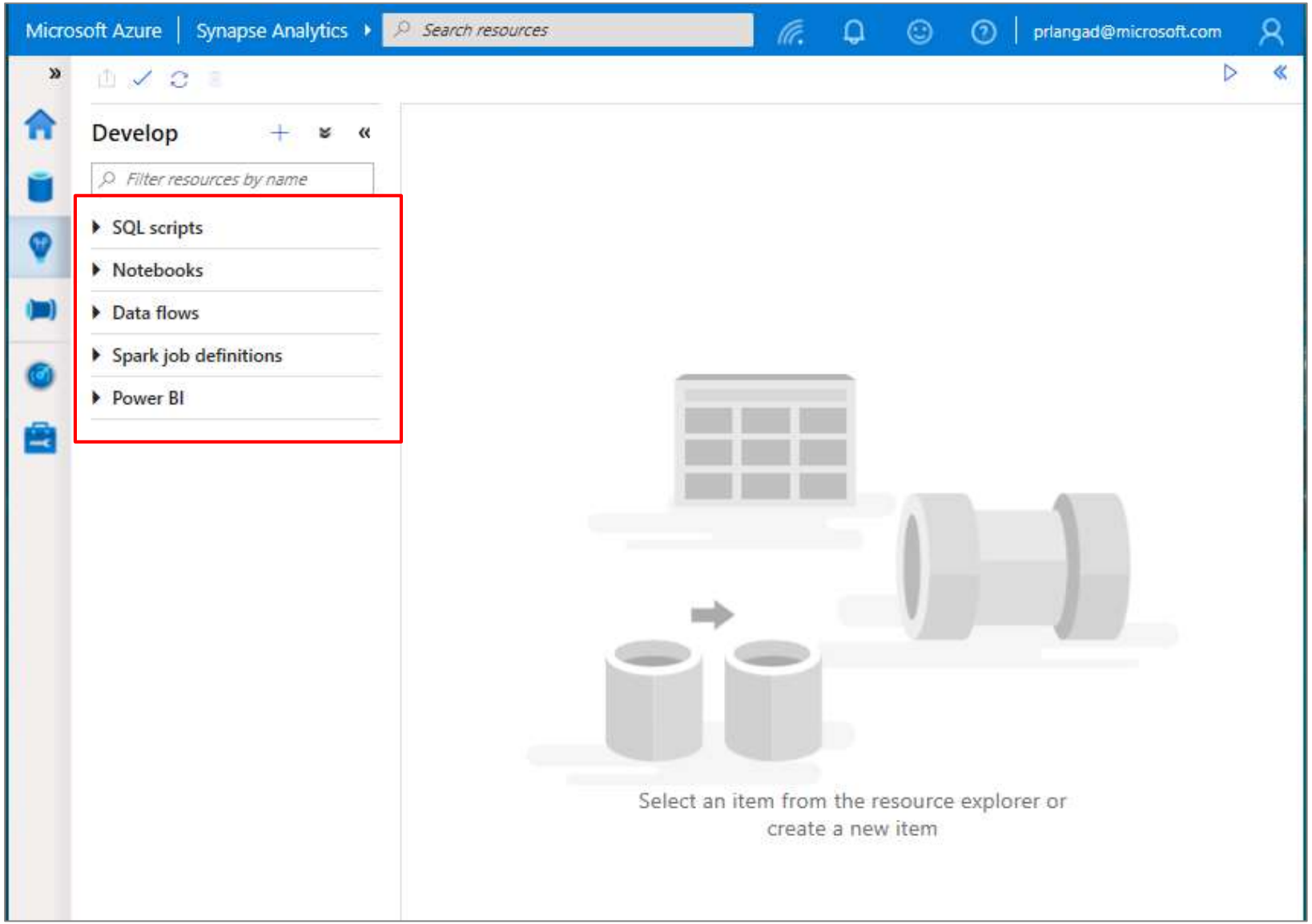
Benefits

Multiple languages to analyze data under one umbrella

Switch over notebooks and scripts without loosing content

Code intellisense offers reliable code development

Create insightful visualizations



Develop Hub - SQL scripts

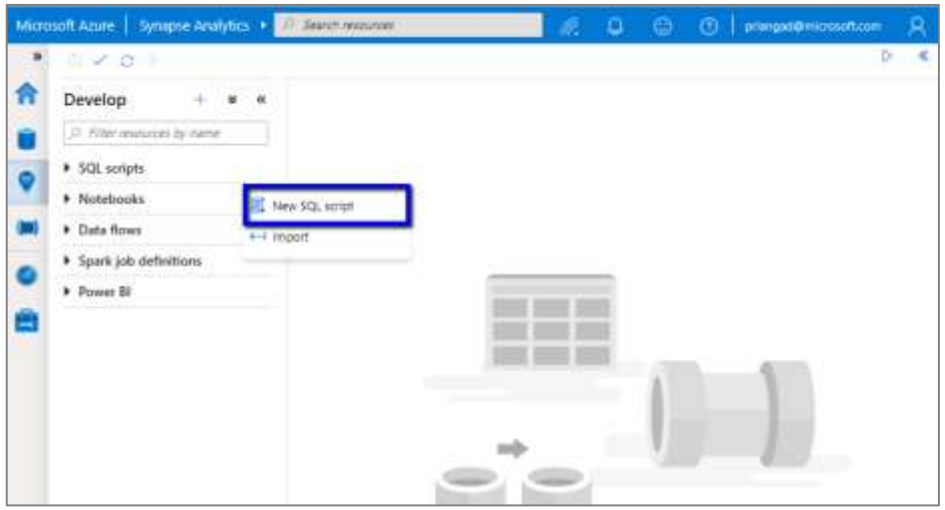
SQL Script

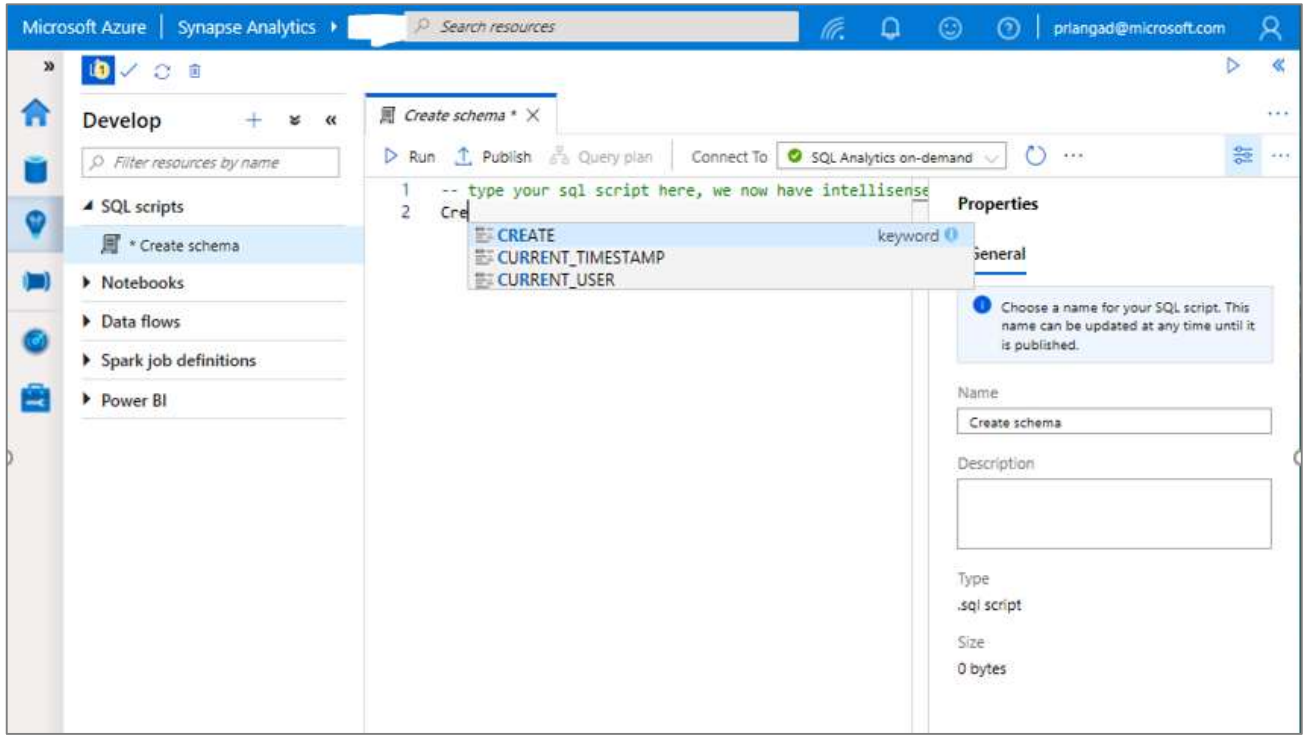
Authoring SQL Scripts

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

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

Language support and intellisense

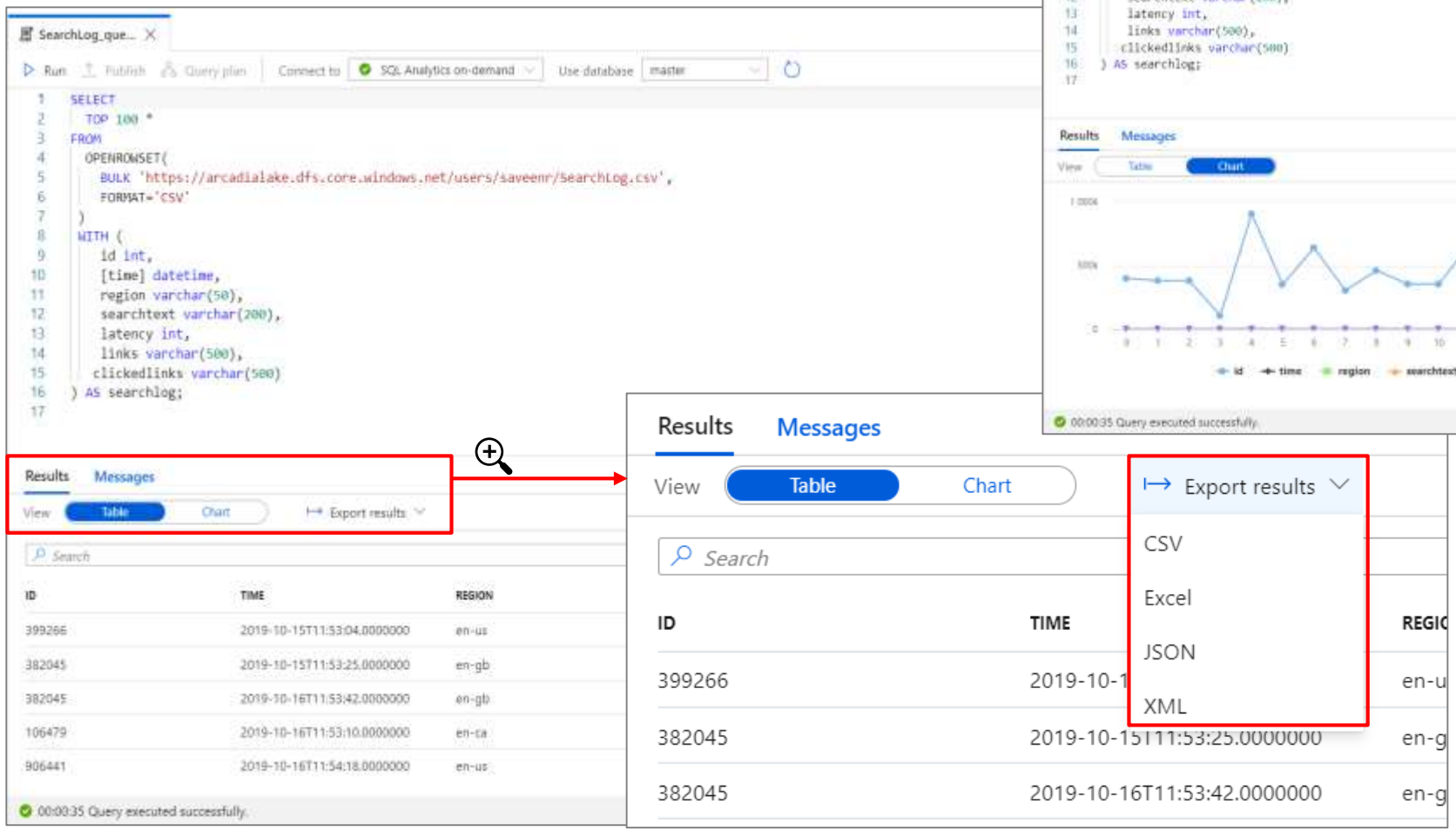




Develop Hub - SQL scripts

SQL Script

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



Develop Hub - Notebooks

Notebooks

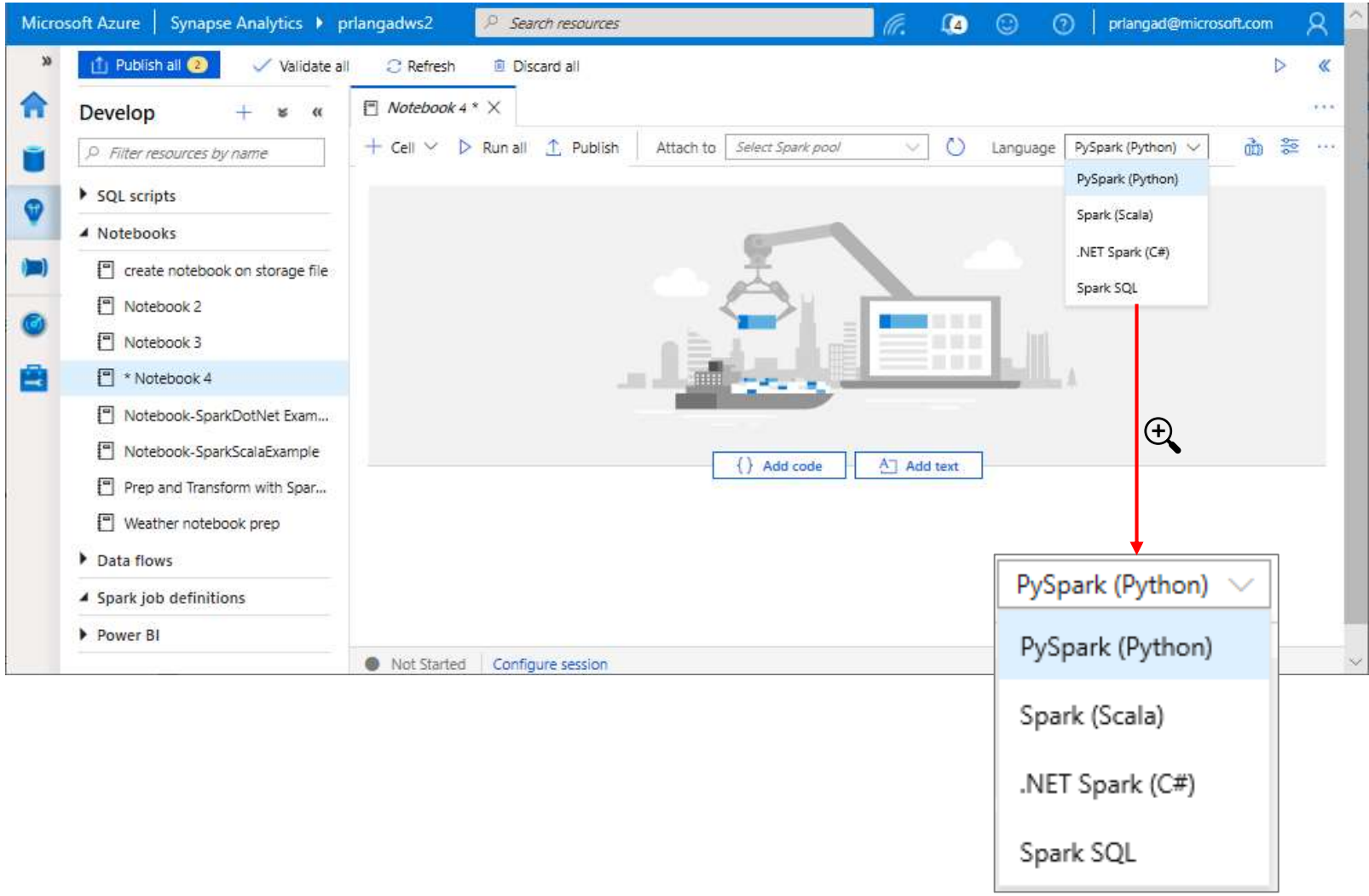
Allows to write multiple languages in one notebook

%%<Name of language>

Offers use of temporary tables across languages

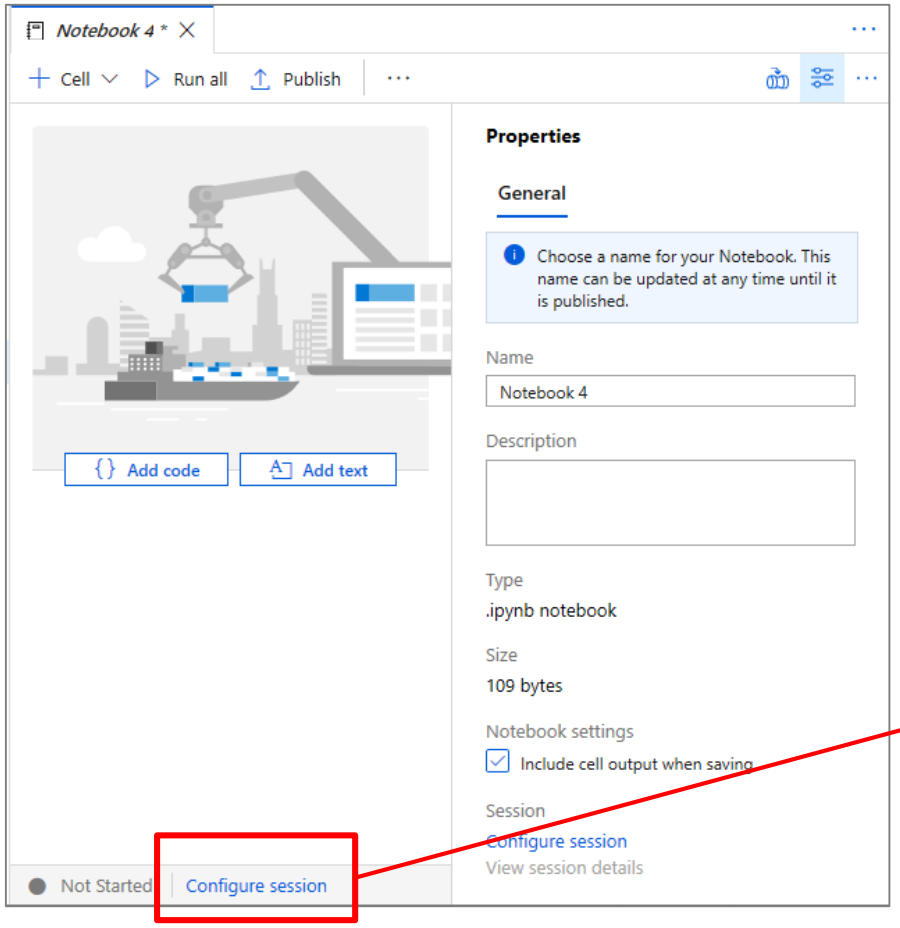
Language support for Syntax highlight, syntax error, syntax code completion, smart indent, code folding

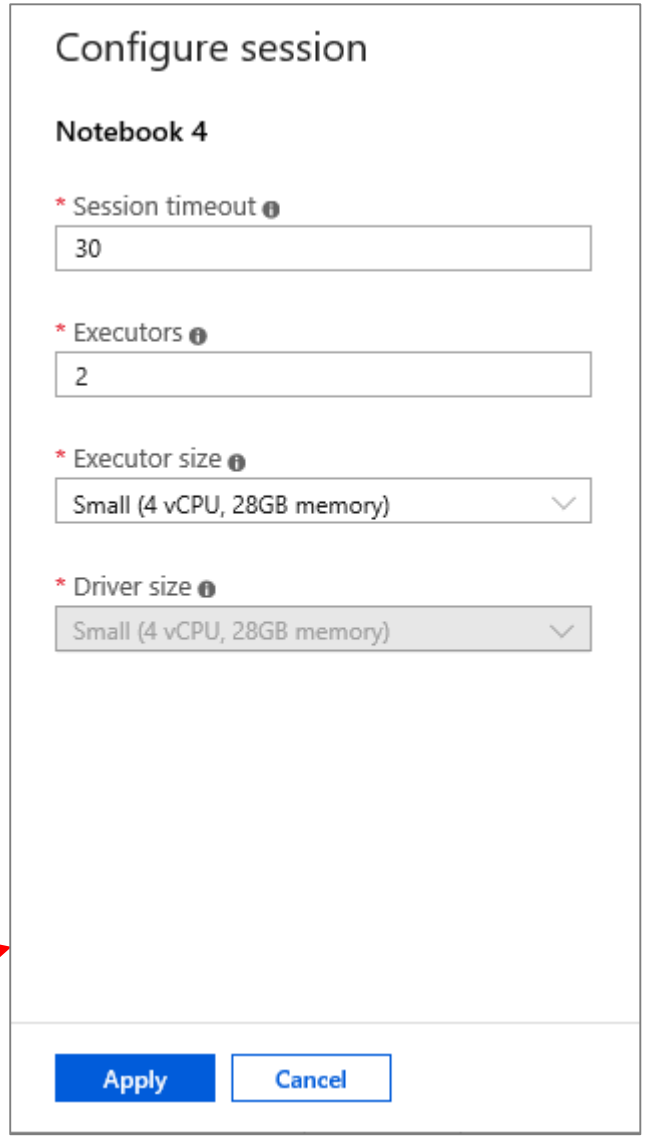
Export results



Develop Hub - Notebooks

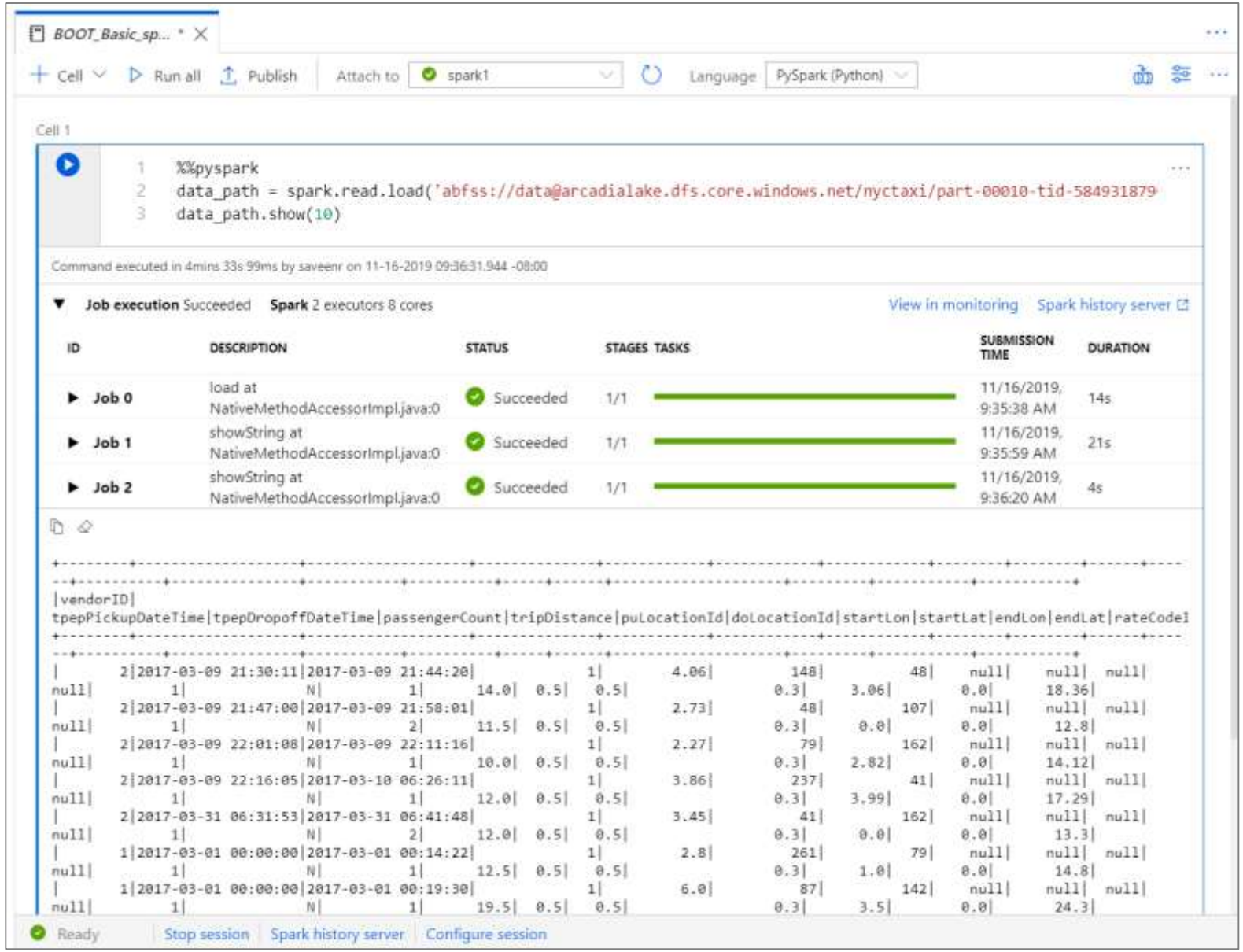
Configure session allows developers to control how many resources are devoted to running their notebook.





Develop Hub - Notebooks

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



Dataflow Capabilities

Handle upserts, updates, deletes on sql sinks

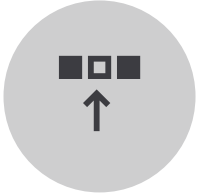
Add new partition methods

Add schema drift support

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

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

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



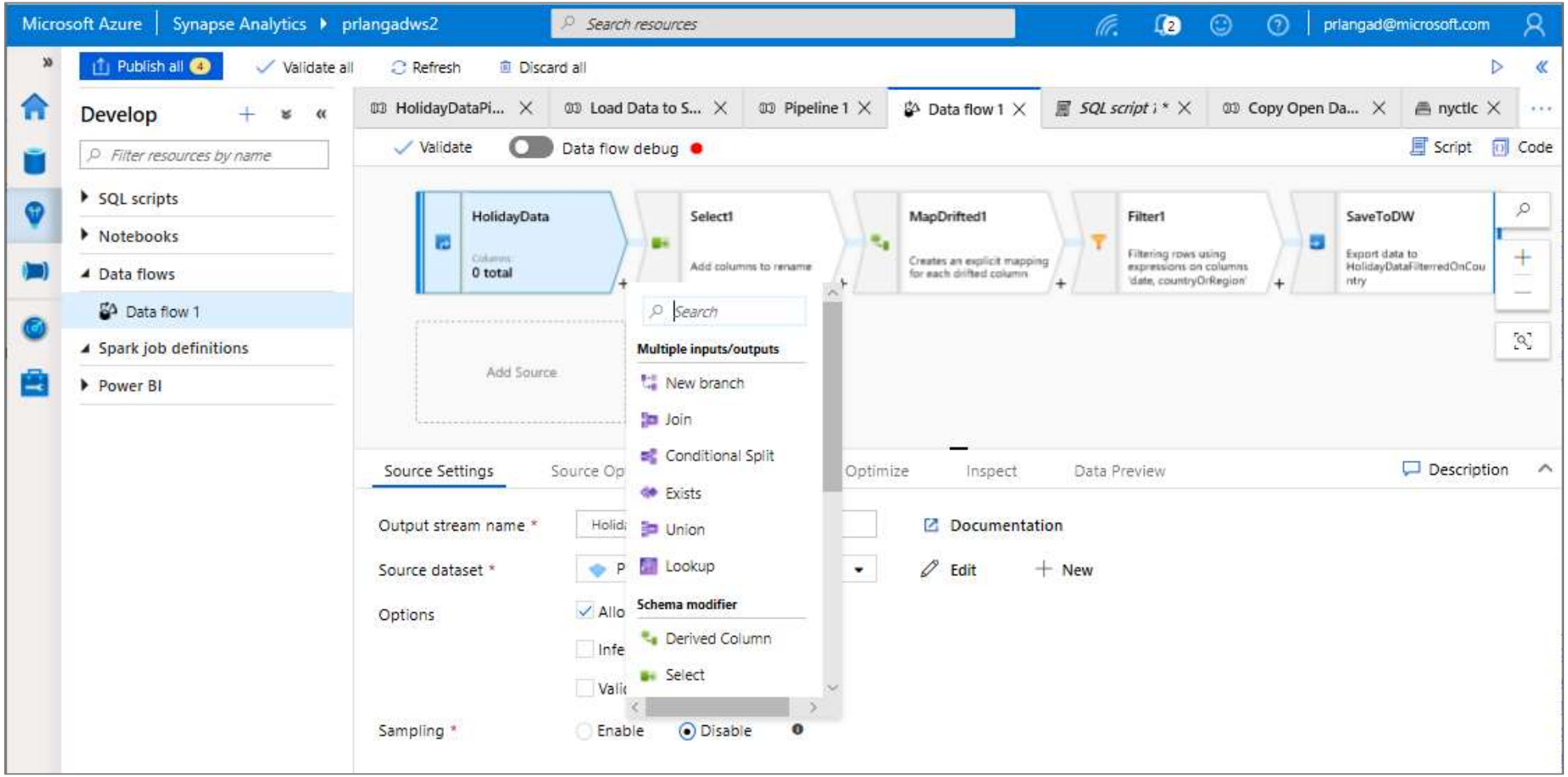
Data lineage – Capturing sink column lineage & impact analysis(invaluable if this is for enterprise deployment)

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

Develop Hub - Data Flows

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

Provides a code-free experience.



Develop Hub – Power BI

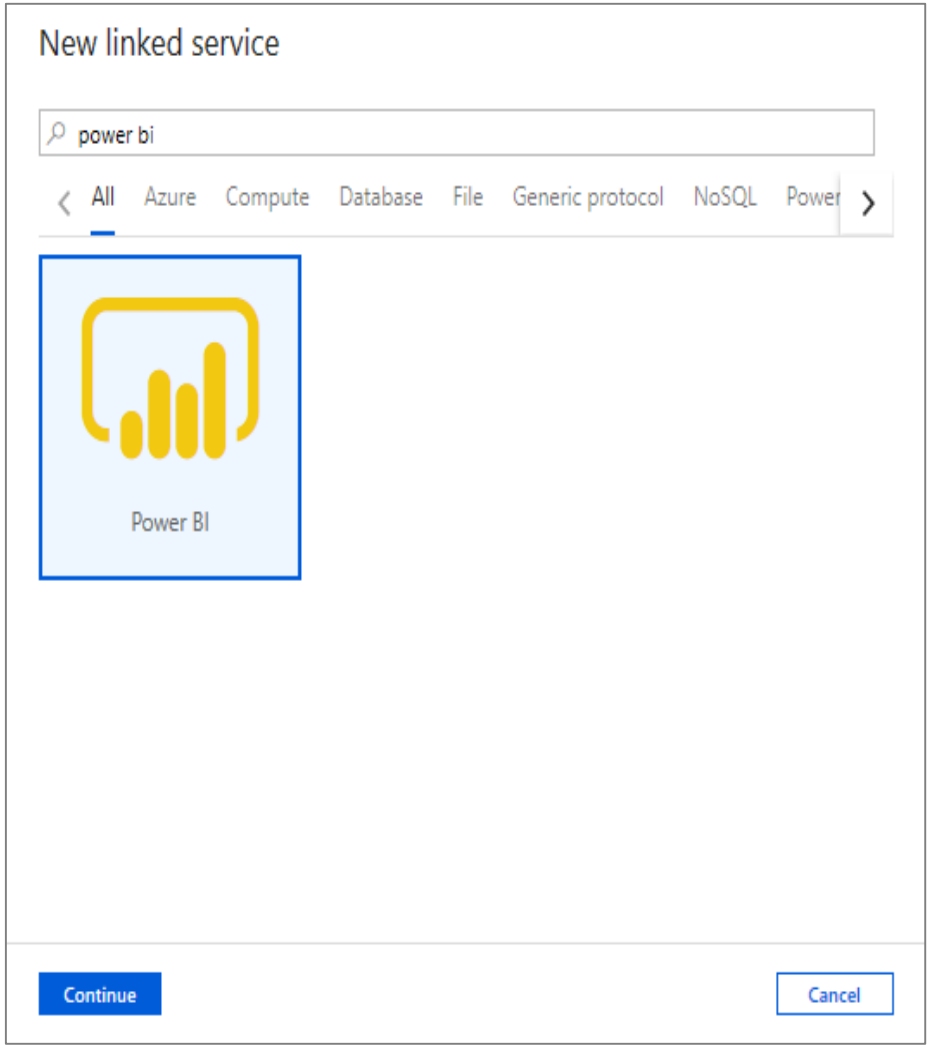
Overview

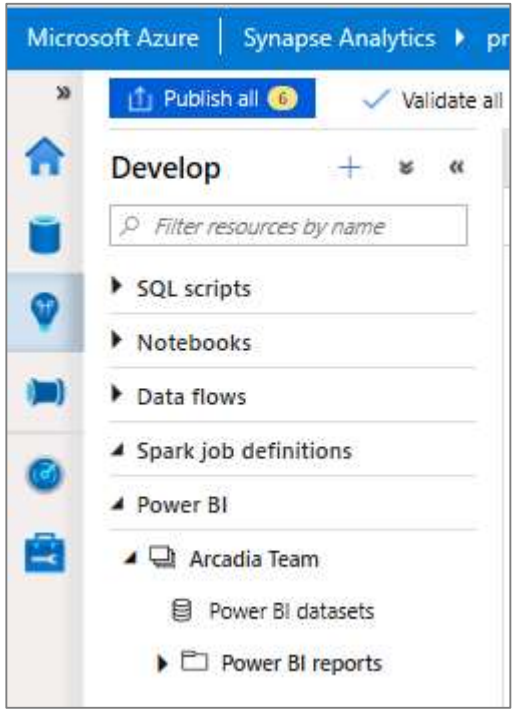
Create Power BI reports in the workspace

Provides access to published reports in the workspace

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

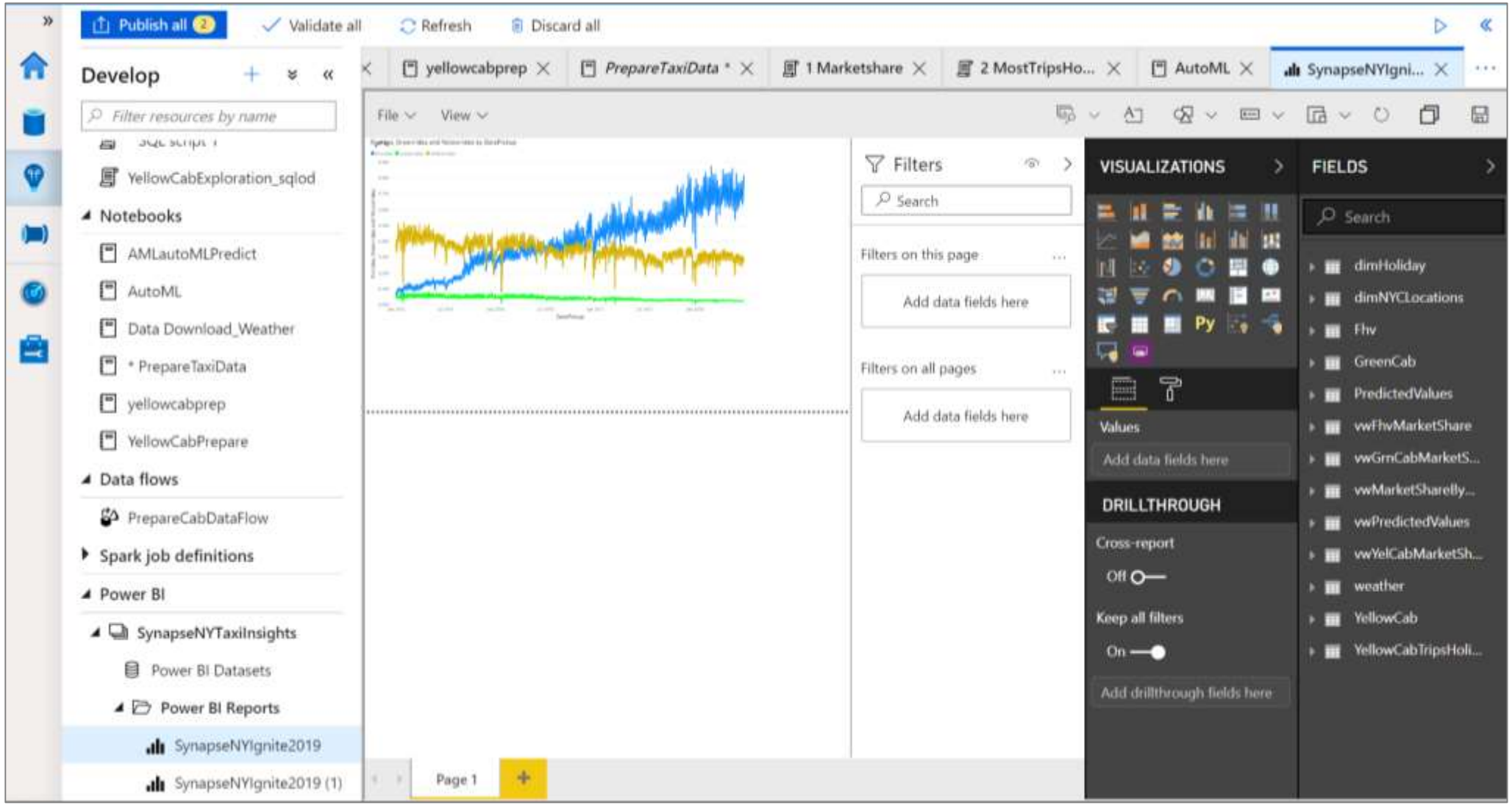
Visually explore and analyze data





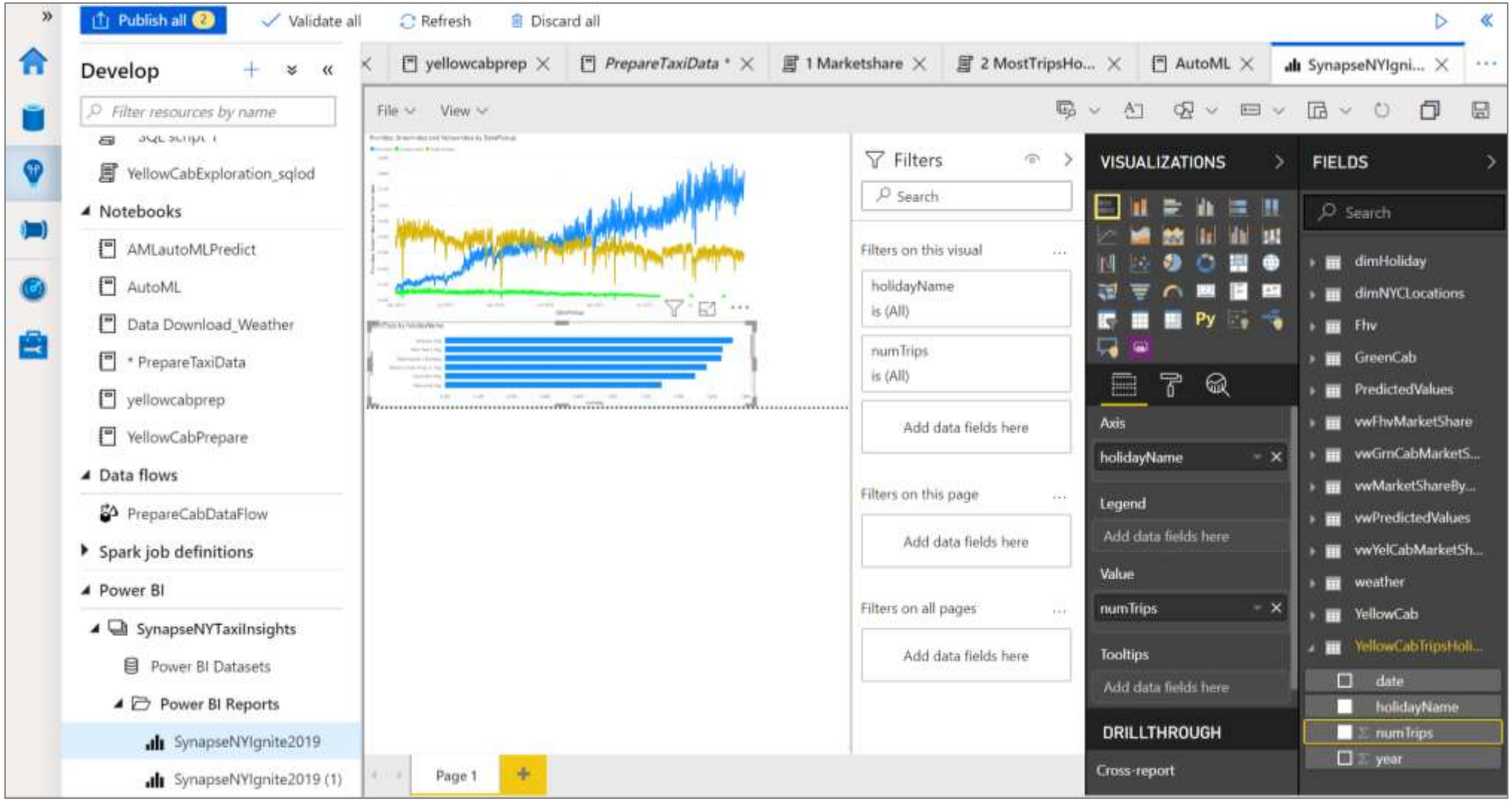
Develop Hub – Power BI

View published reports in Power BI workspace



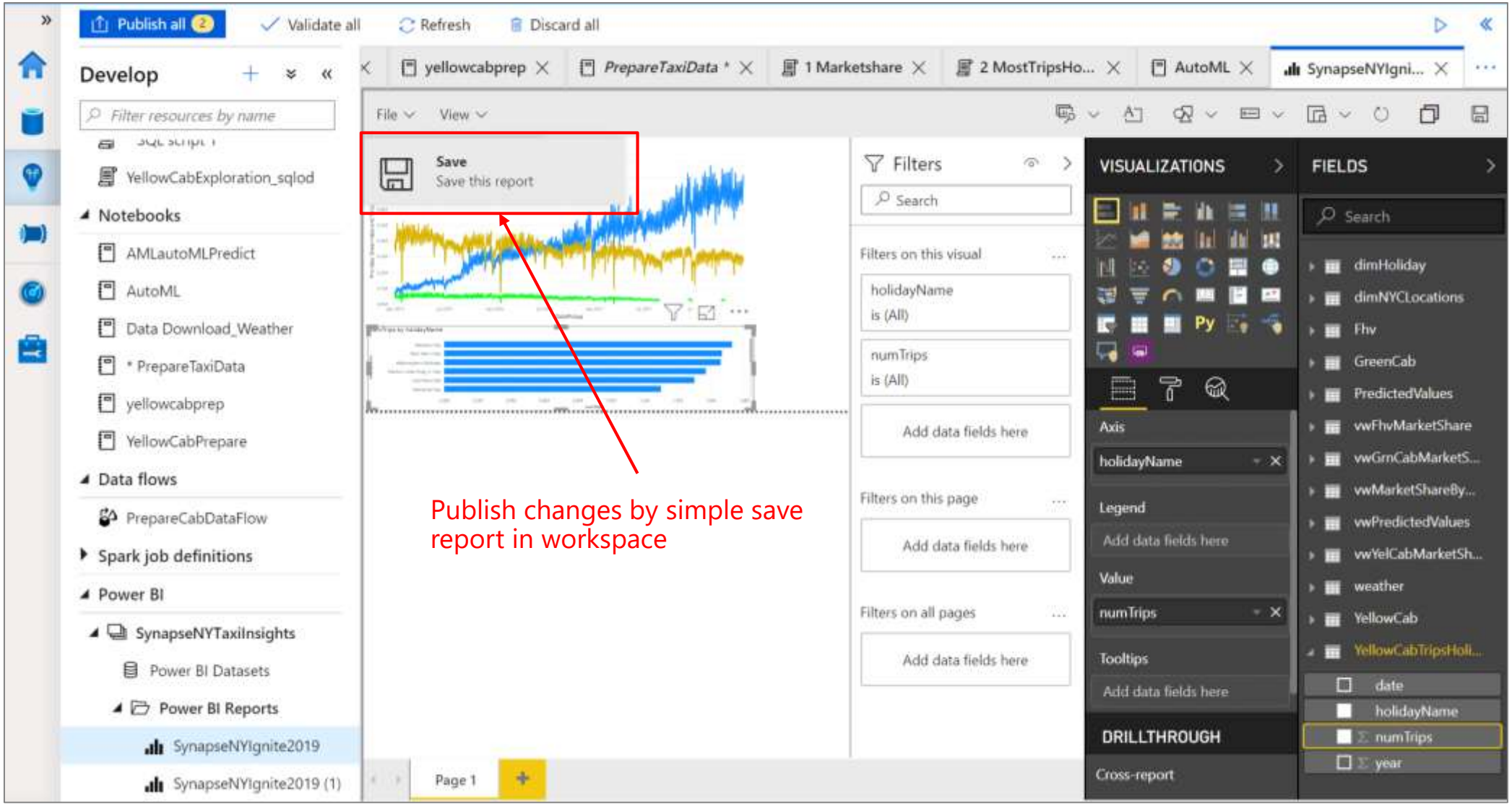
Develop Hub – Power BI

Edit reports in Synapse workspace



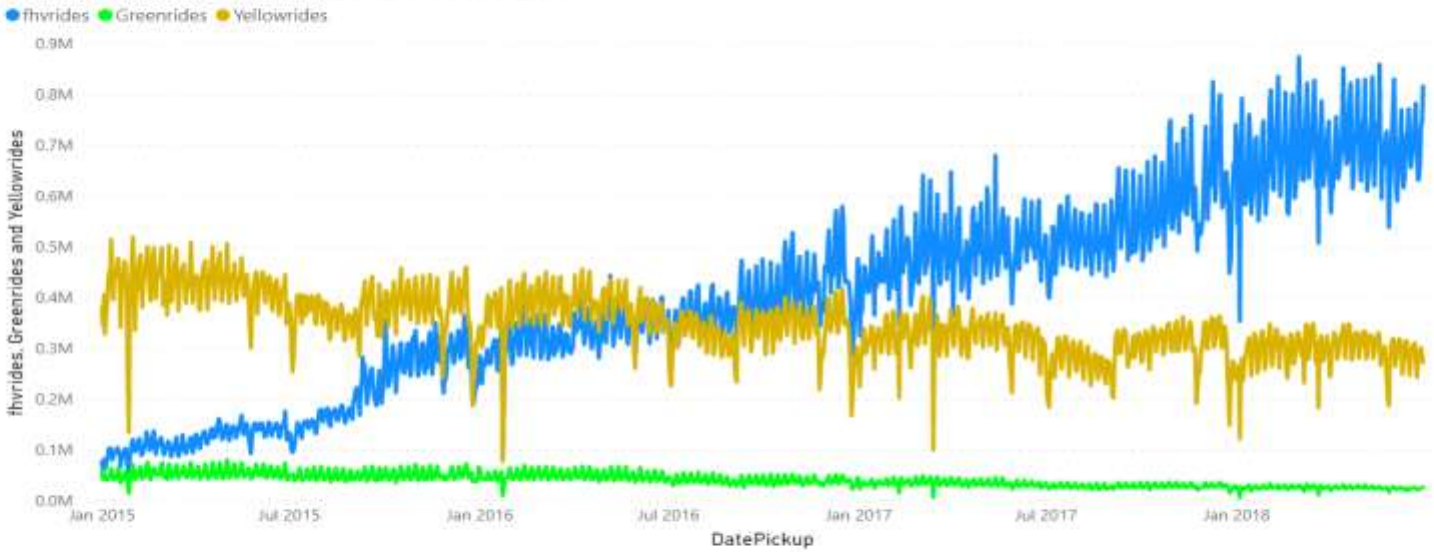
Develop Hub – Power BI

Publish edited reports in Synapse workspace to Power BI workspace

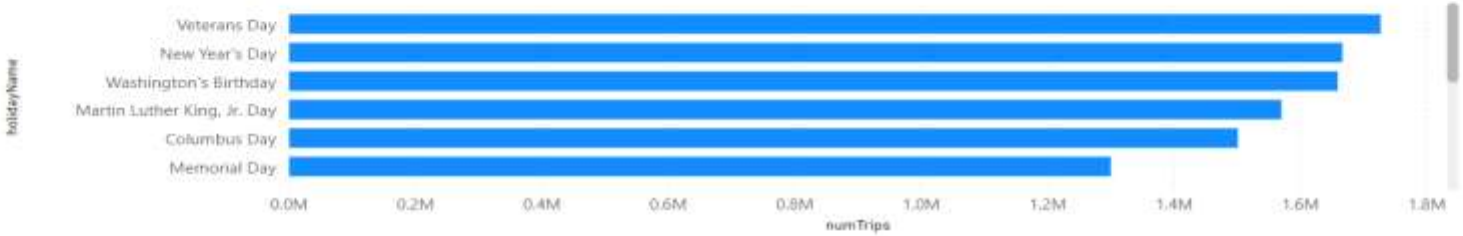


Home
Favorites
Recent
Apps
Shared with me
Workspaces
SynapseNYTaxiInsi...
Get data

fhvrides, Greenrides and Yellowrides by DatePickup



numTrips by holidayName

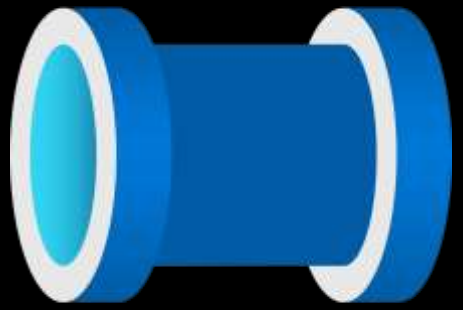


Filters

Search

There aren't any filters to display.

Real-time publish on save

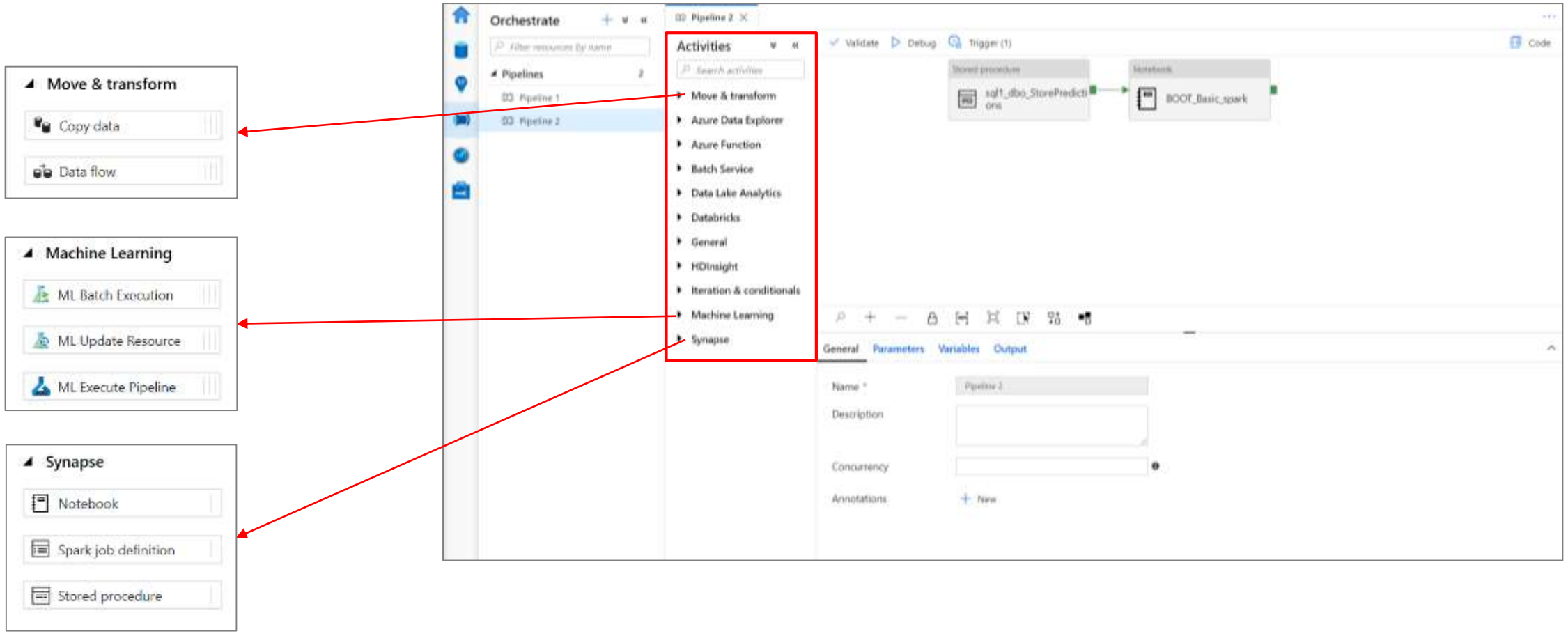


Synapse Studio
Orchestrate hub

Orchestrate Hub

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

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

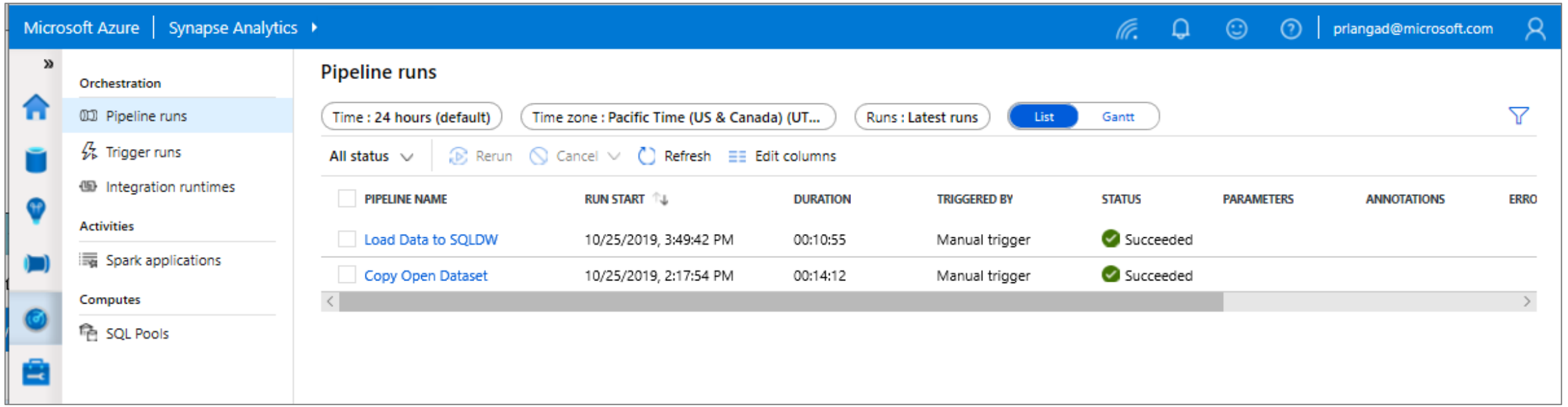


Synapse Studio
Monitor hub

Monitor Hub

Overview

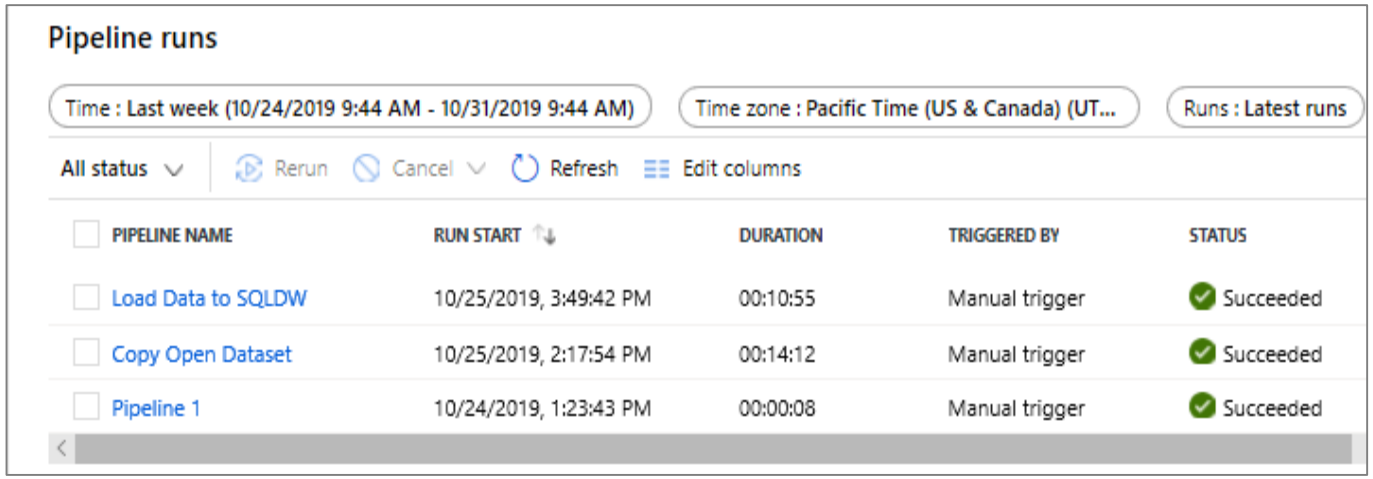
This feature provides ability to monitor orchestration, activities and compute resources.



Monitoring Hub - Orchestration

Overview

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



Benefits

Track all/specific pipelines

Monitor pipeline run and activity run details

Find the root cause of pipeline failure or activity failure

Monitoring Hub - Spark applications

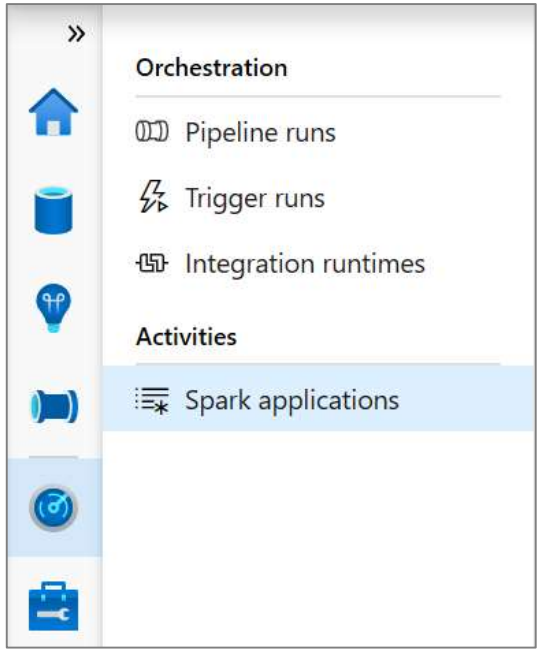
Overview

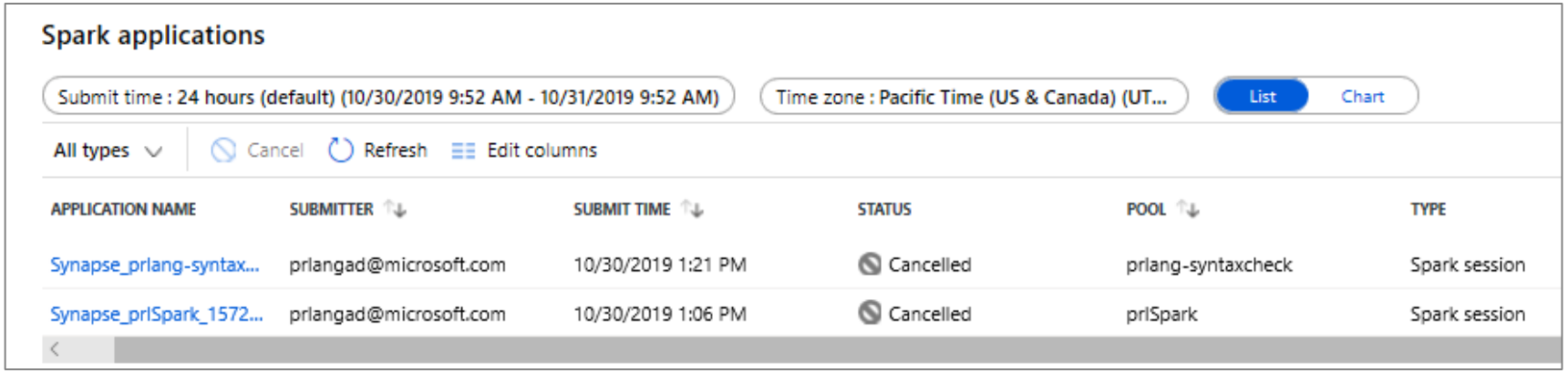
Monitor Spark pools, Spark applications for the progress and status of activities

Benefits

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

Track the usage of resources



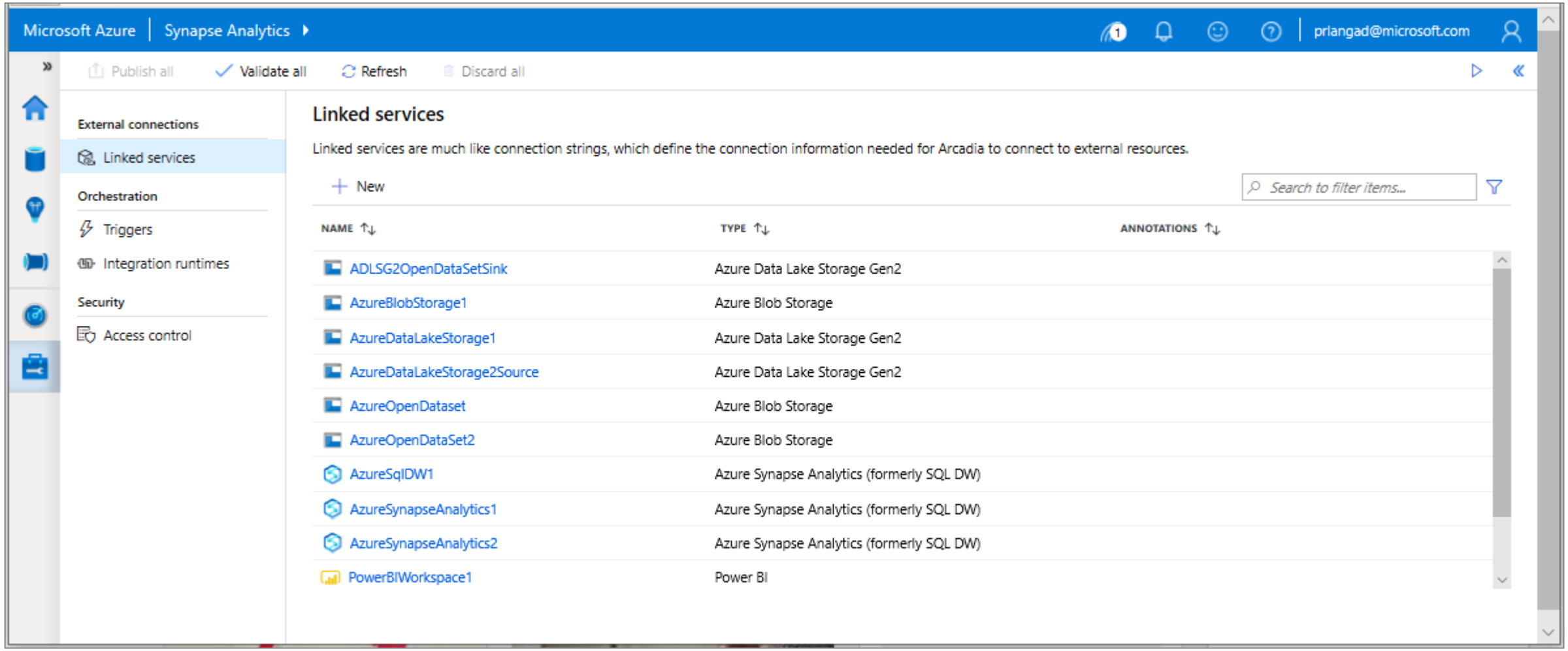


Synapse Studio
Manage hub

Manage Hub

Overview

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



Manage – Linked services

Overview

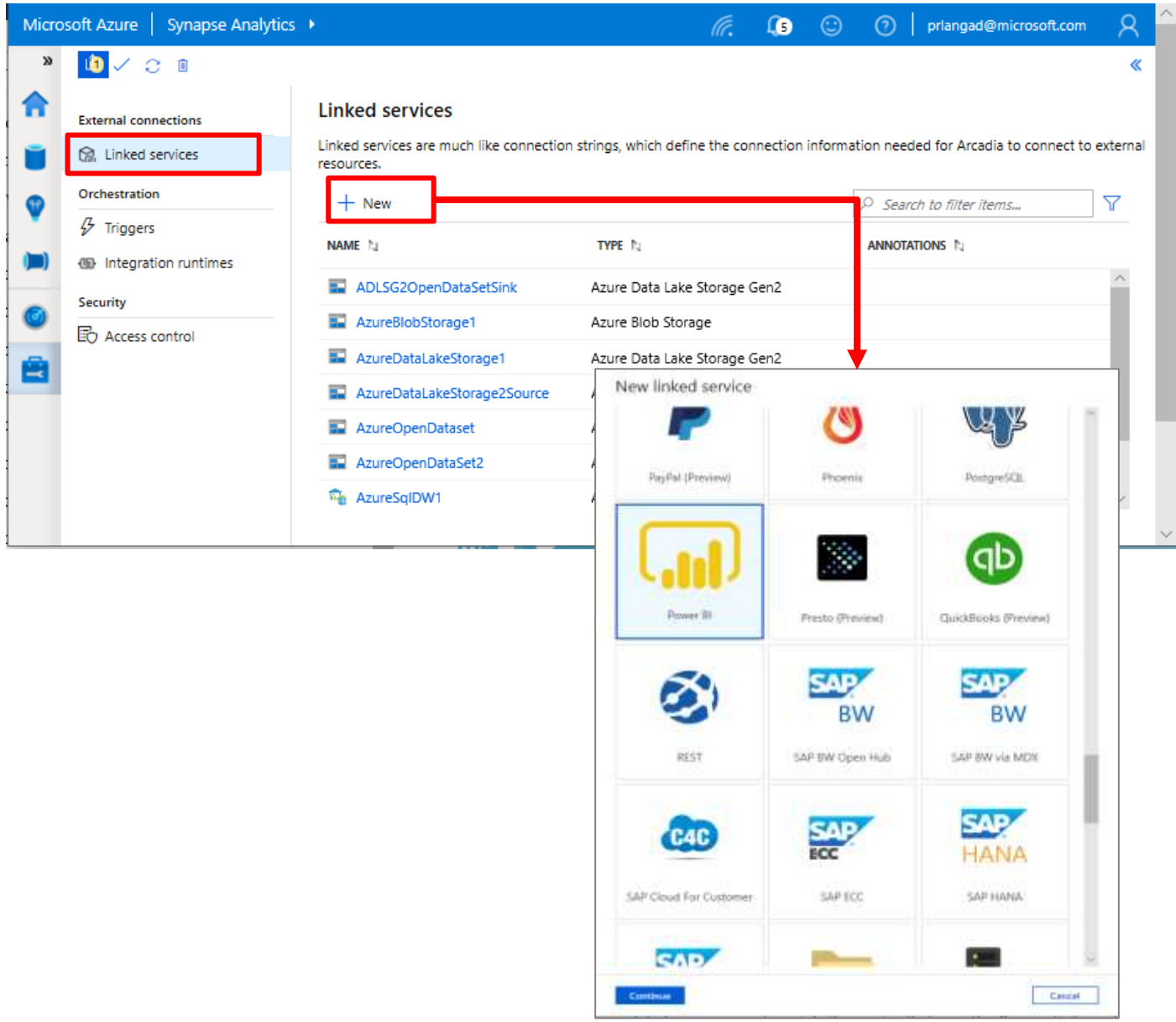
It defines the connection information needed to connect to external resources.

Benefits

Offers pre-build 90+ connectors

Easy cross platform data migration

Represents data store or compute resources



Manage – Access Control

Overview

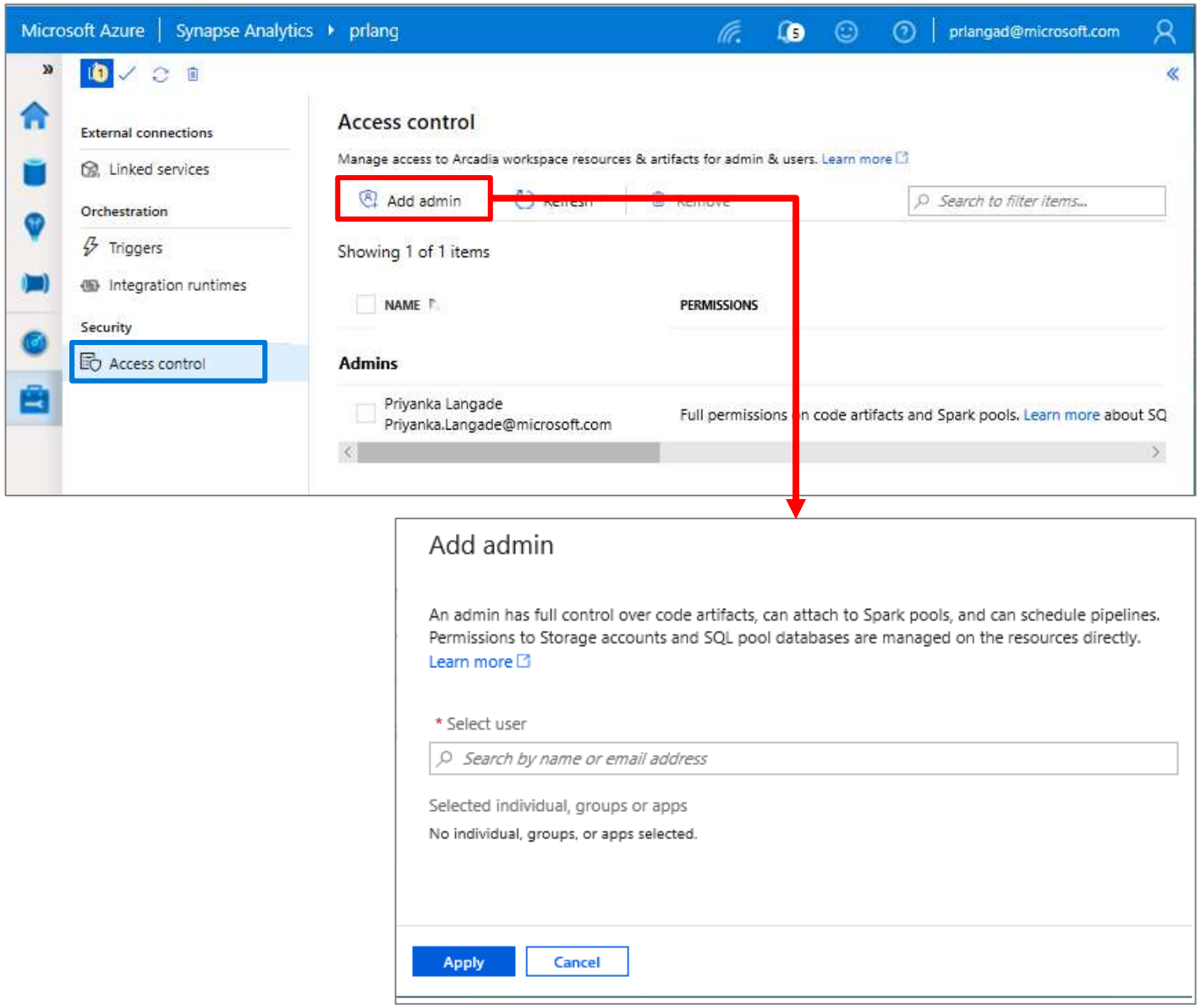
It provides access control management to workspace resources and artifacts for admin and users

Benefits

Share workspace with the team

Increases productivity

Manage permissions on code artifacts and Spark pools



Manage – Triggers

Overview

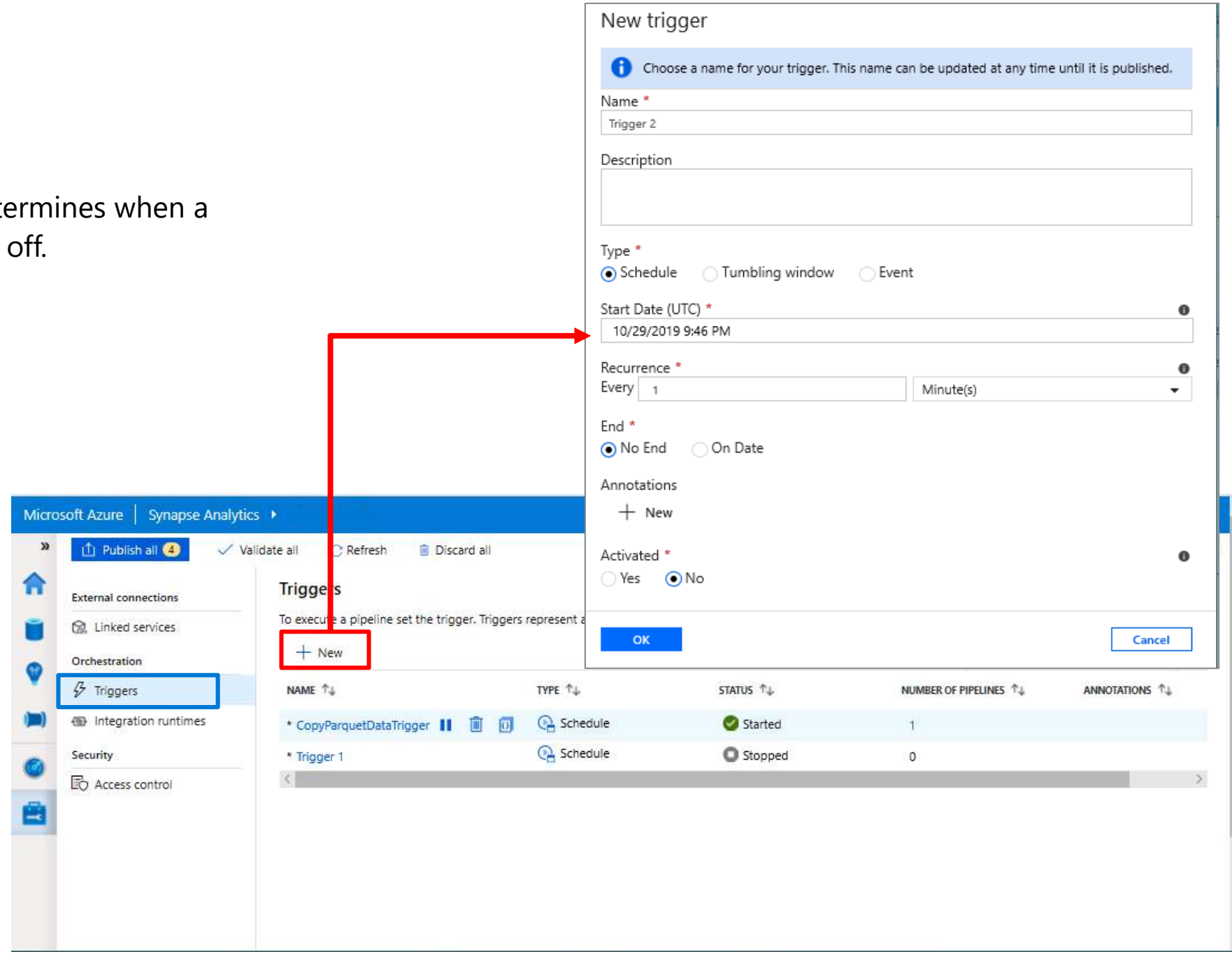
It defines a unit of processing that determines when a pipeline execution needs to be kicked off.

Benefits

Create and manage

- Schedule trigger
- Tumbling window trigger
- Event trigger

Control pipeline execution



Manage – Integration runtimes

Overview

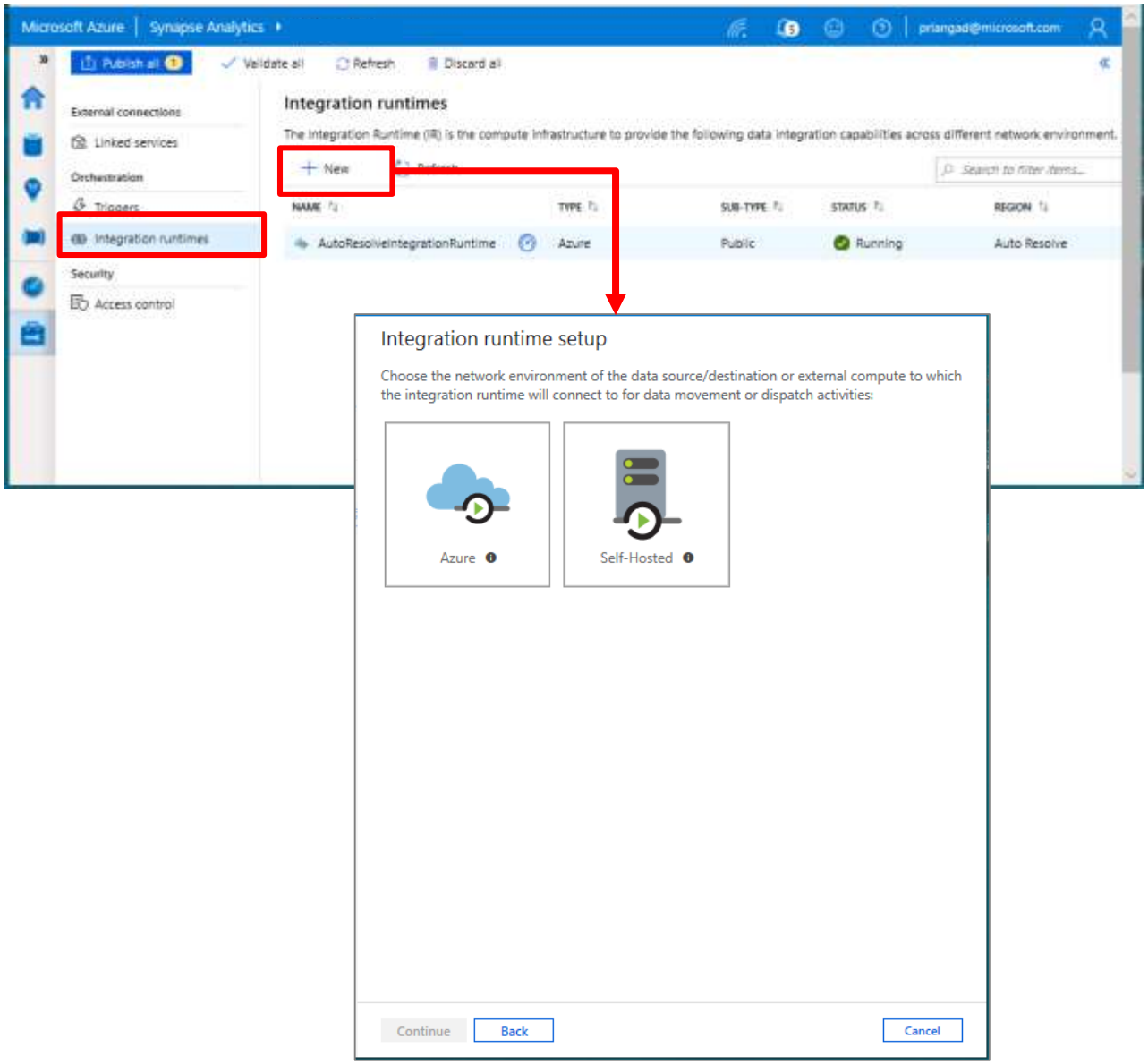
Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

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



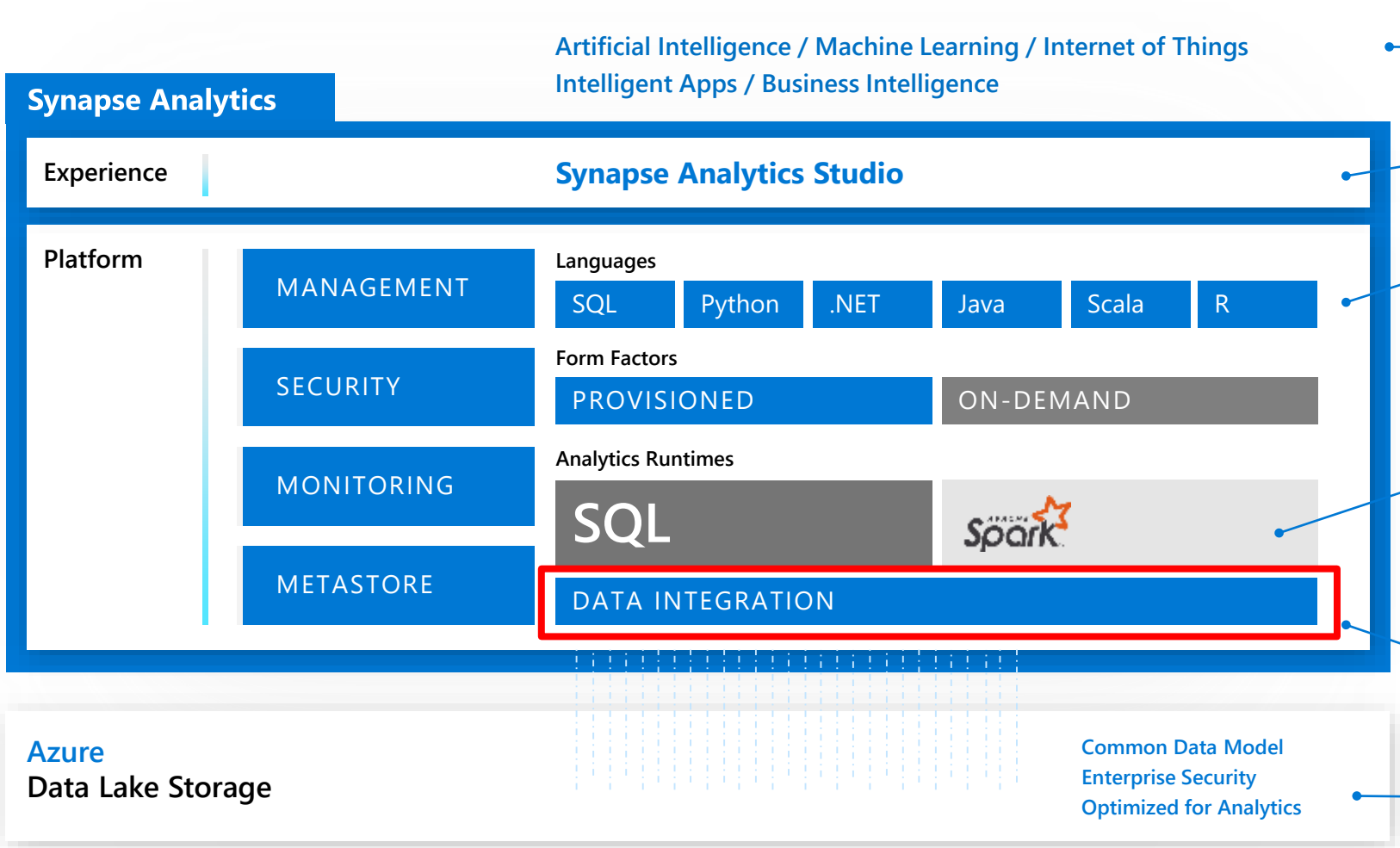
Microsoft

Azure Synapse Analytics

Data Integration

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available provisioned and serverless on-demand

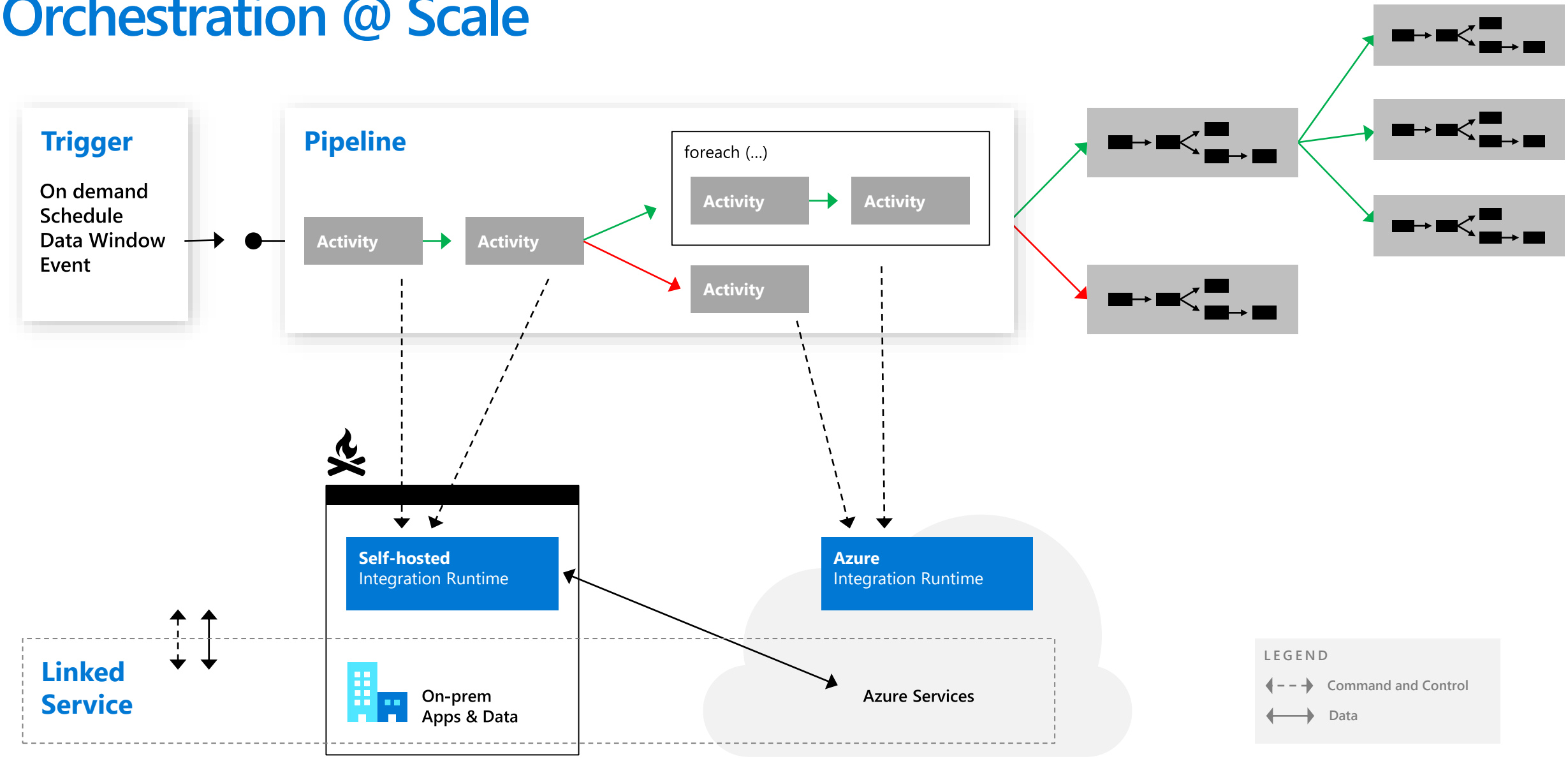
SQL Analytics offering T-SQL for batch, streaming and interactive processing

Spark for big data processing with Python, Scala, R and .NET

Integrated **platform services** for, management, security, monitoring, and metastore

Data **lake integrated** and Common Data Model aware

Orchestration @ Scale



Data Movement

Scalable

per job elasticity

Up to 4 GB/s

Simple

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

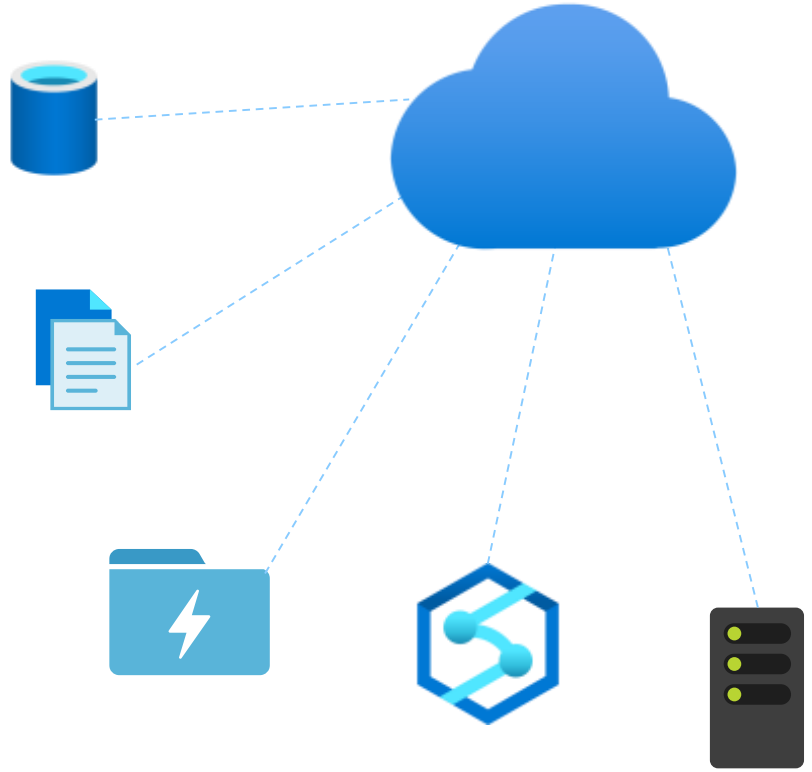
Serverless, no infrastructure to manage

Access all your data

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

Data Movement as a Service: 25 points of presence worldwide

Self-hostable Integration Runtime for hybrid movement



90+ Connectors out of the box

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						Oracle Responsys	Zoho	
Table storage								

Pipelines

Overview

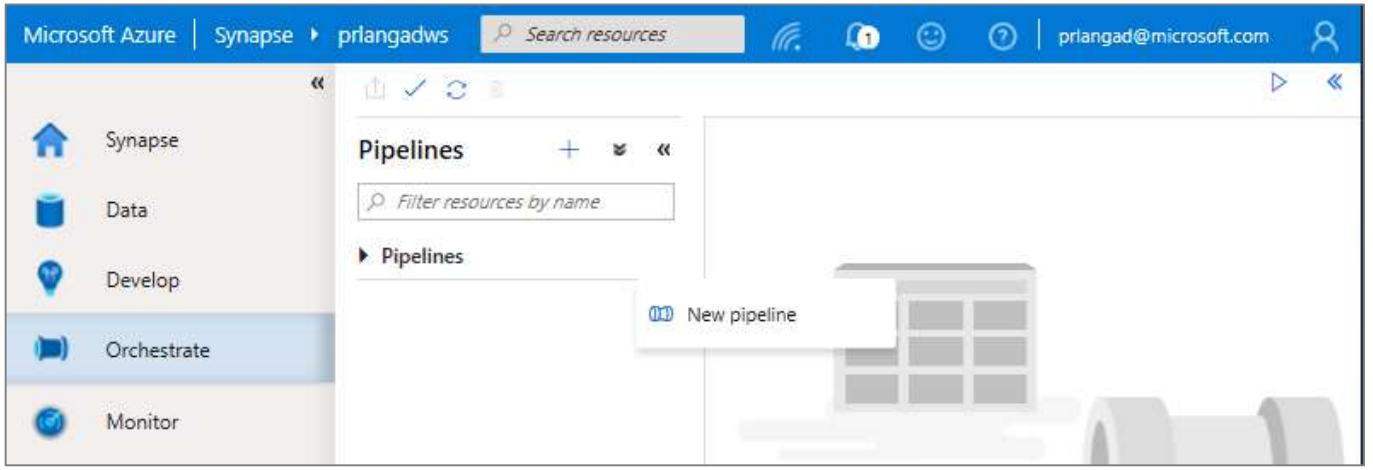
It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

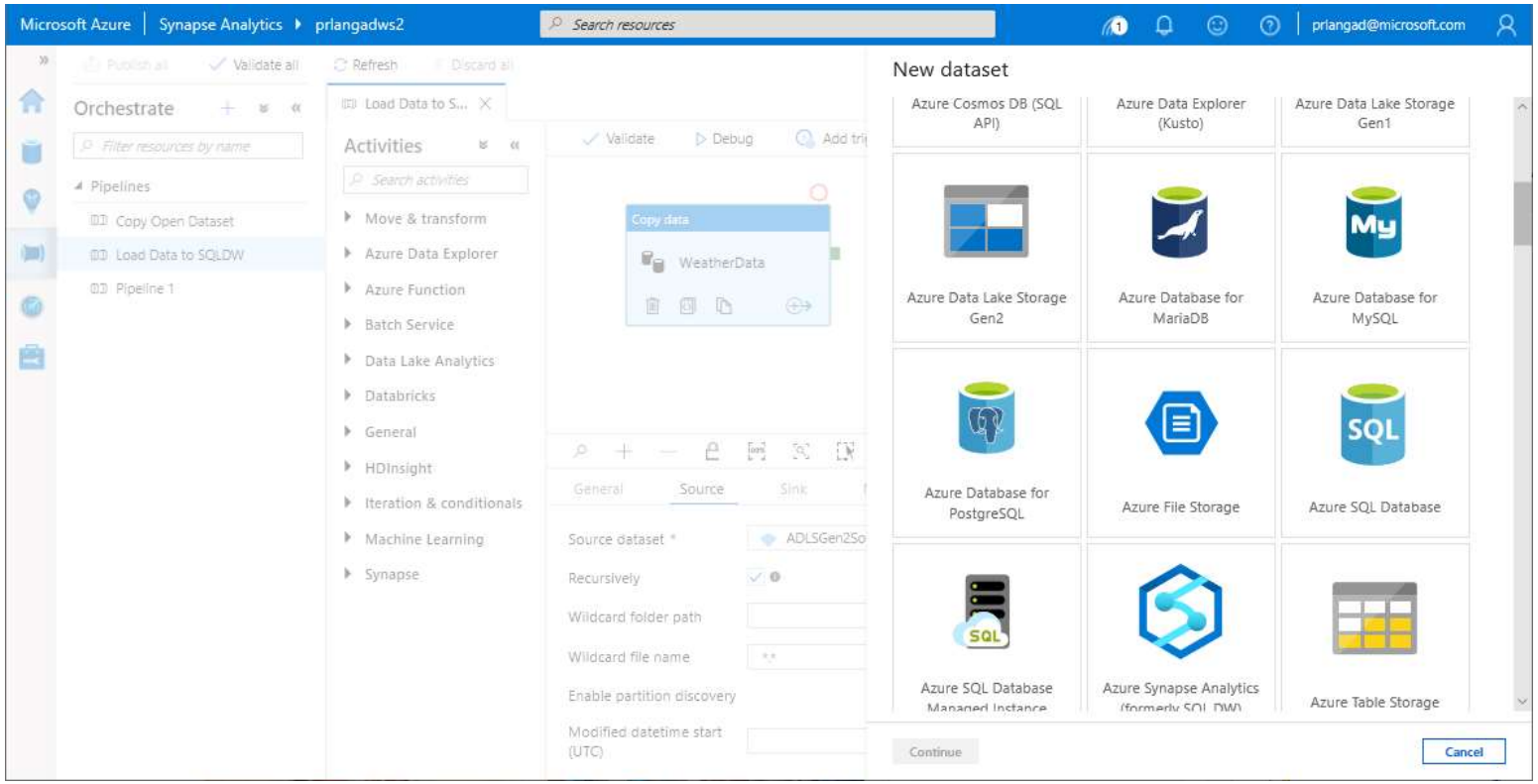
Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

Graphical development experience

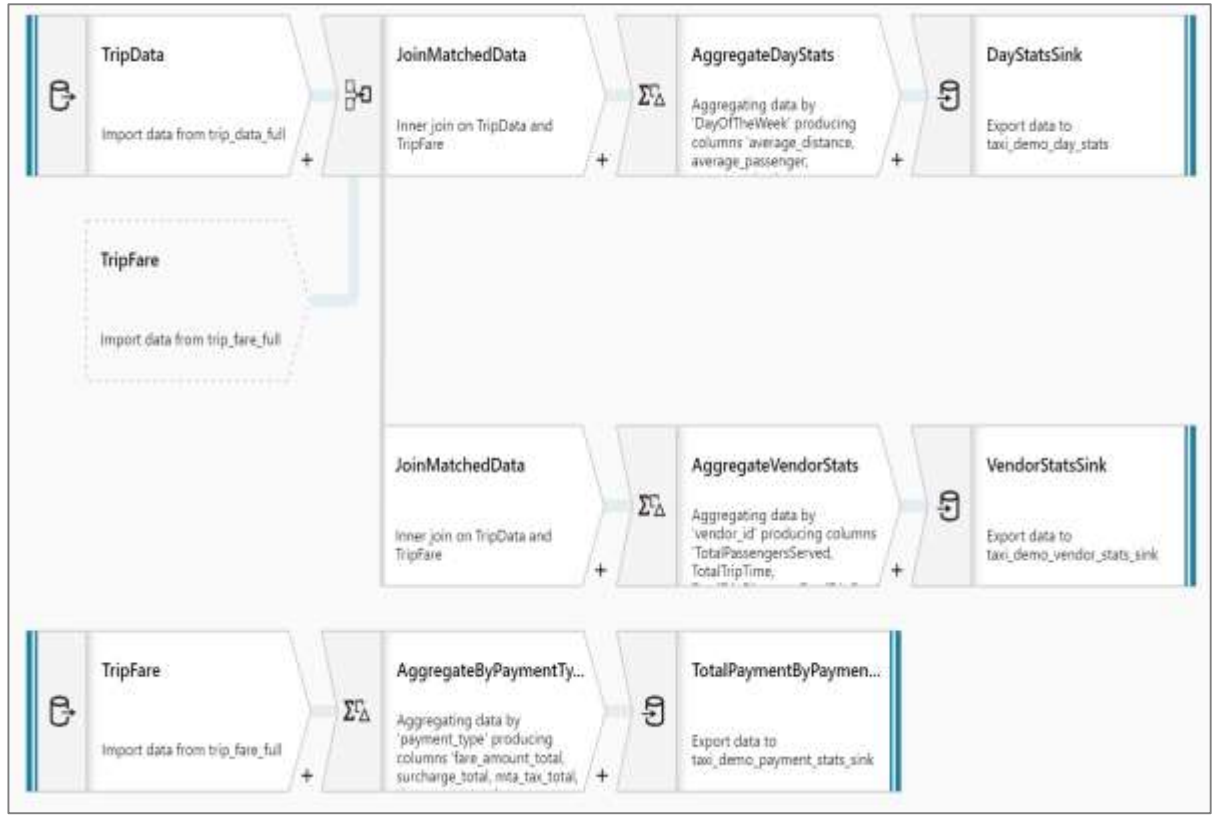




Prep & Transform Data

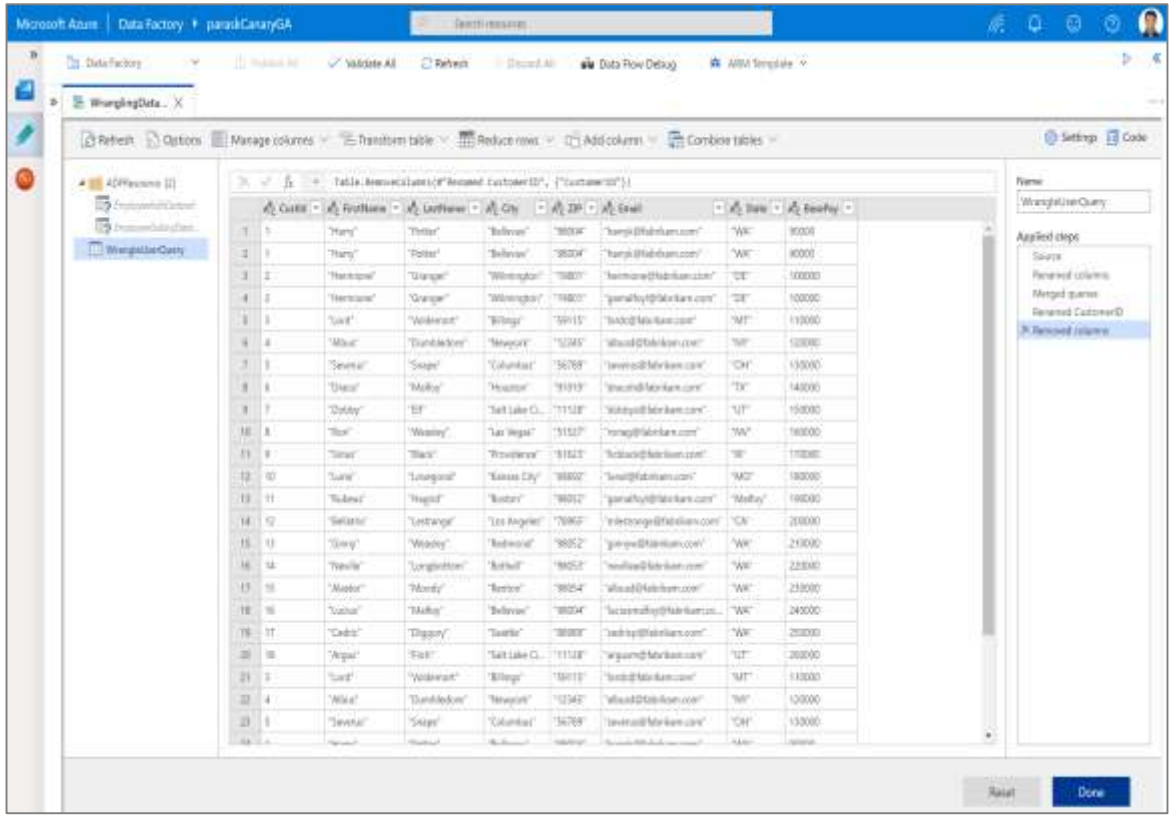
Mapping Dataflow

Code free data transformation @scale



Wrangling Dataflow

Code free data preparation @scale



Triggers

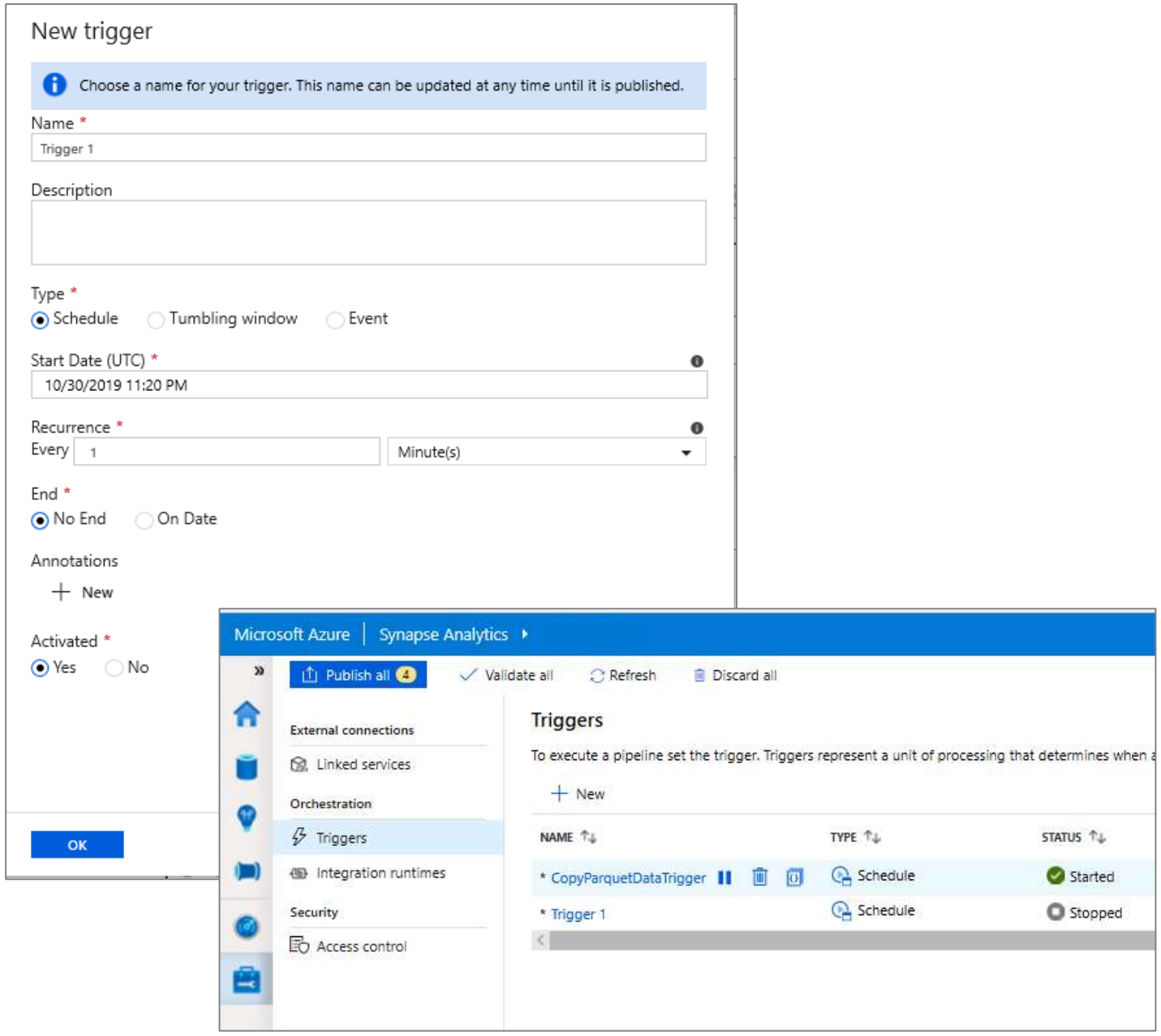
Overview

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.



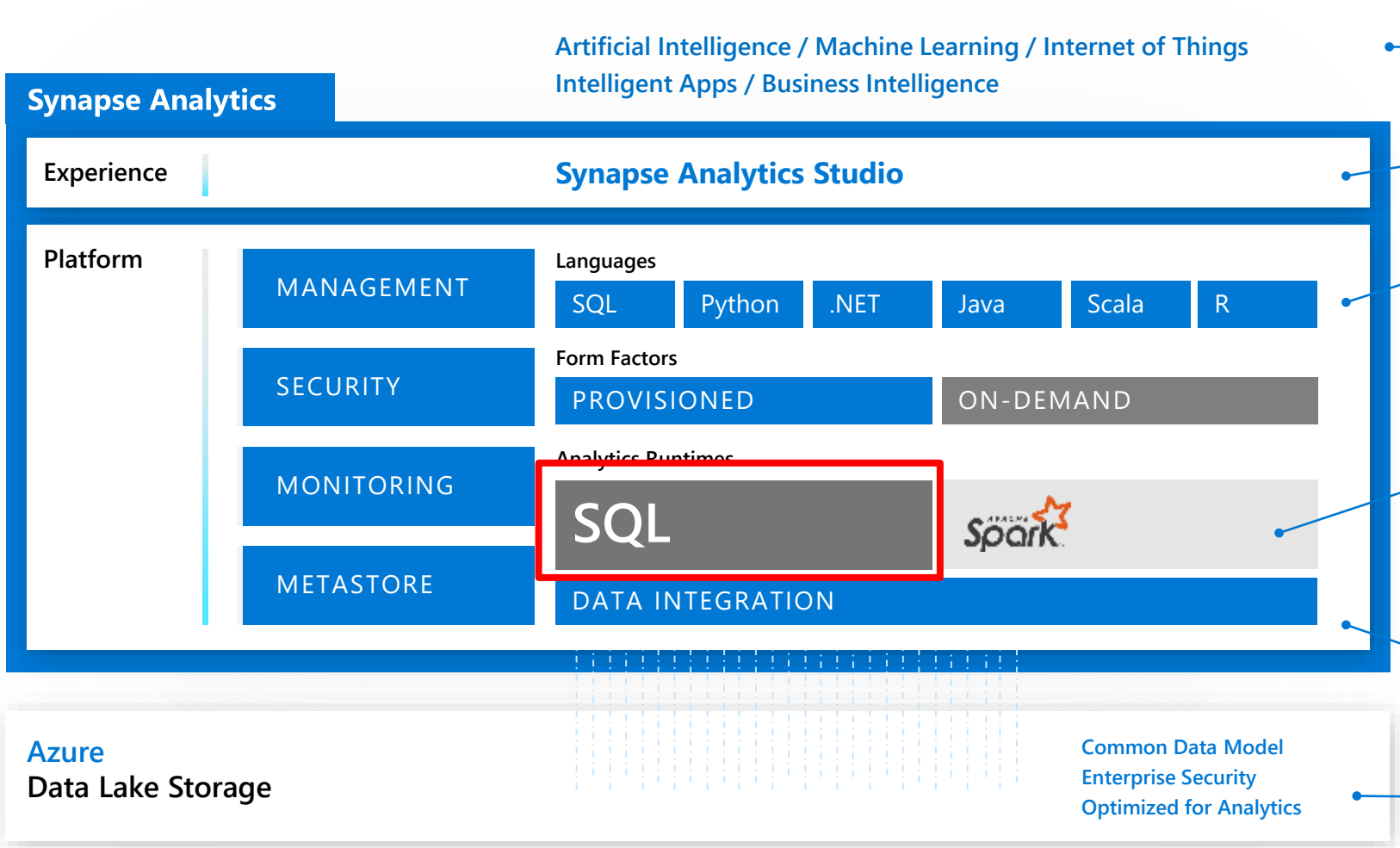
Microsoft

Azure Synapse Analytics

SQL Analytics

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available provisioned and serverless on-demand

SQL Analytics offering T-SQL for batch, streaming and interactive processing

Spark for big data processing with Python, Scala, R and .NET

Integrated **platform services** for, management, security, monitoring, and metastore

Data **lake integrated** and Common Data Model aware

Comprehensive SQL functionality

Advanced storage system

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

T-SQL Querying

- Windowing aggregates
- Approximate execution (Hyperloglog)
- JSON data support

Complete SQL object model

- Tables
- Views
- Stored procedures
- Functions

Windowing functions

OVER clause

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

Computes a value for each row in the window

Aggregate functions

COUNT, MAX, AVG, SUM, APPROX_COUNT_DISTINCT, MIN, STDEV, STDEVP, STRING_AGG, VAR, VARP, GROUPING, GROUPING_ID, COUNT_BIG, CHECKSUM_AGG

Ranking functions

RANK, NTILE, DENSE_RANK, ROW_NUMBER

Analytical functions

LAG, LEAD, FIRST_VALUE, LAST_VALUE, CUME_DIST, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK

ROWS | RANGE

PRECEDING, 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

Windowing Functions (continued)

Analytical functions

LAG, LEAD, FIRST_VALUE, LAST_VALUE, CUME_DIST, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK

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

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

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

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

Windowing Functions (continued)

ROWS | RANGE

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

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

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

Approximate execution

HyperLogLog accuracy

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

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

APPROX_COUNT_DISTINCT

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

Use Case: Approximating web usage trend behavior

```
-- Syntax
APPROX_COUNT_DISTINCT ( expression )

-- The approximate number of different order keys by order status from the orders table.
SELECT O_OrderStatus, APPROX_COUNT_DISTINCT(O_OrderKey) AS Approx_Distinct_OrderKey
FROM dbo.Orders
GROUP BY O_OrderStatus
ORDER BY O_OrderStatus;
```

Approximate execution

APPROX_COUNT_DISTINCT

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

100 %

Results | Messages

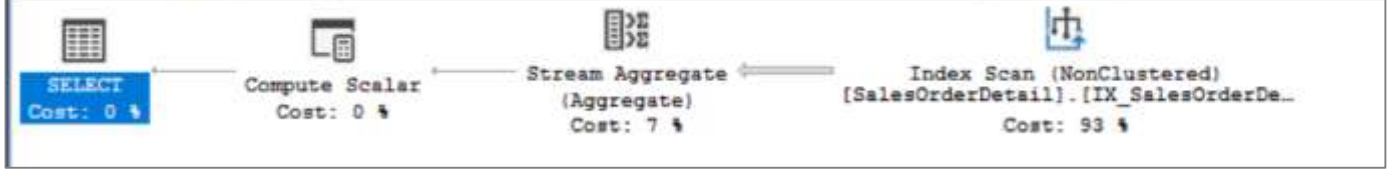
	Approx_Distinct_OrderKey
1	540

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

100 %

Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT APPROX_COUNT_DISTINCT([SalesOrderDetailID]) AS Approx_Distinct_OrderKey FROM [S



COUNT DISTINCT

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

100 %

Results | Messages

	Distinct_OrderKey
1	542

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

100 %

Messages | Execution plan

Query 1: Query cost (relative to the batch): 100%
SELECT COUNT (DISTINCT [SalesOrderDetailID]) AS Distinct_OrderKey FROM [SalesLT].[SalesOrderDetail]

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

Snapshot isolation

Overview

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

This prevents dirty reads.

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

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

Isolation level

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

READ_COMMITTED_SNAPSHOT

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

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

JSON data support – insert JSON data

Overview

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

JSON data is stored using standard NVARCHAR table columns.

Benefits

Transform arrays of JSON objects into table format

Performance optimization using clustered columnstore indexes and memory optimized tables

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

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

JSON data support – read JSON data

Overview

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

- ISJSON – verify if text is valid JSON
- JSON_VALUE – extract a scalar value from a JSON string
- JSON_QUERY – extract a JSON object or array from a JSON string

Benefits

Ability to get standard columns as well as JSON column

Perform aggregation and filter on JSON values

```
-- Return all rows with valid JSON data
SELECT CustomerId, OrderDetails
FROM   CustomerOrders
WHERE  ISJSON(OrderDetails) > 0;
```

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

```
-- Extract values from JSON string
SELECT CustomerId,
      Country,
      JSON_VALUE(OrderDetails,'$.StoreId') AS StoreId,
      JSON_QUERY(OrderDetails,'$.Item') AS ItemDetails
FROM   CustomerOrders;
```

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

JSON data support – modify and operate on JSON data

Overview

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

Modify JSON data with the following:

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

Benefits

Flexibility to update JSON string using T-SQL

Convert hierarchical data into flat tabular structure

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

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

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

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

Stored Procedures

Overview

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

Promotes flexibility and modularity.

Supports parameters and nesting.

Benefits

Reduced server/client network traffic, improved performance

Stronger security

Easy maintenance

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

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

Tables – Indexes

Clustered Columnstore index (Default Primary)

Highest level of data compression

Best overall query performance

Clustered index (Primary)

Performant for looking up a single to few rows

Heap (Primary)

Faster loading and landing temporary data

Best for small lookup tables

Nonclustered indexes (Secondary)

Enable ordering of multiple columns in a table

Allows multiple nonclustered on a single table

Can be created on any of the above primary indexes

More performant lookup queries

```
-- Create table with index
CREATE TABLE orderTable
(
   OrderId  INT NOT NULL,
   Date     DATE NOT NULL,
   Name     VARCHAR(2),
   Country  VARCHAR(2)
)
WITH
(
CLUSTERED COLUMNSTORE INDEX |
   HEAP |
   CLUSTERED INDEX (OrderId)
);


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

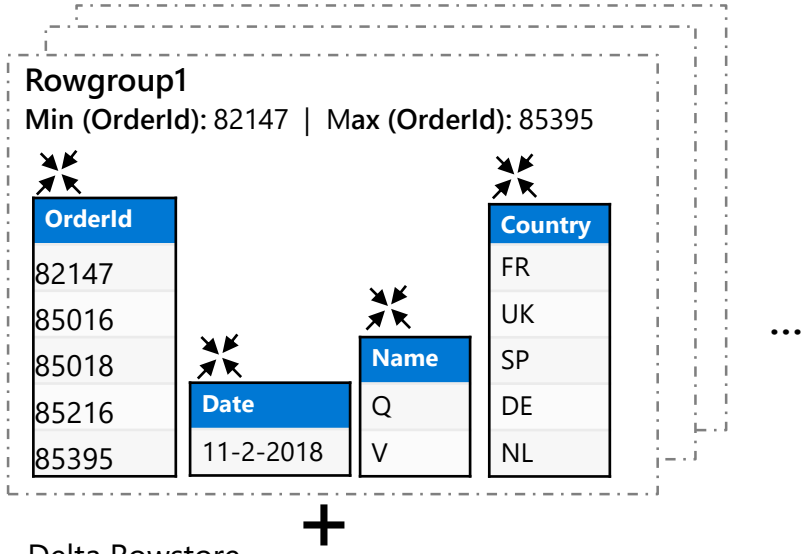
SQL Analytics Columnstore Tables

Logical table structure

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

Clustered columnstore index

(OrderId)



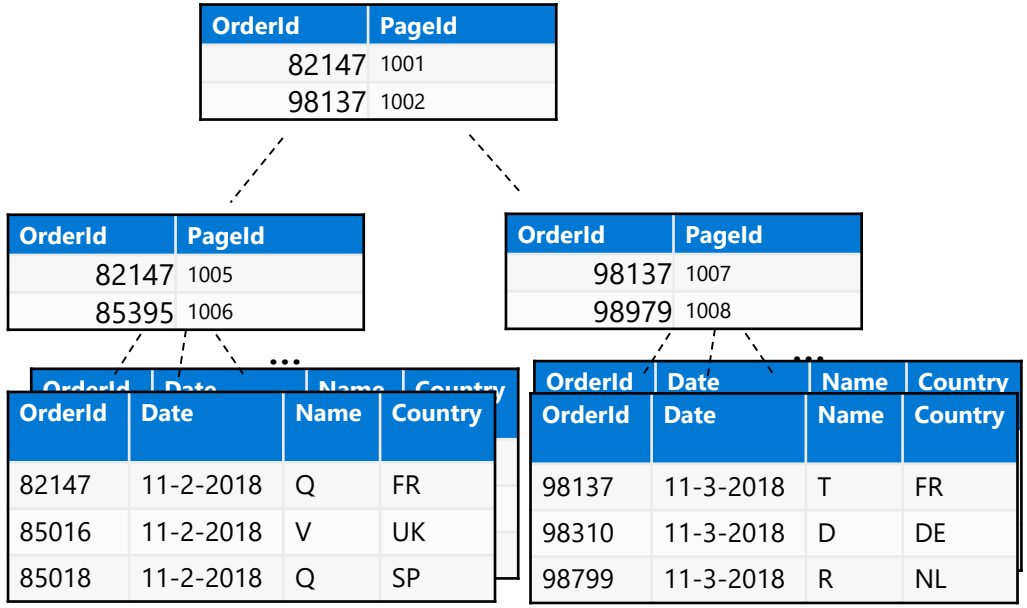
Delta Rowstore

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

Ordered Clustered Columnstore Indexes

Overview

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

```
-- Create Table with Ordered Columnstore Index
CREATE TABLE sortedOrderTable
(
   OrderId  INT NOT NULL,
   Date     DATE NOT NULL,
   Name     VARCHAR(2),
   Country  VARCHAR(2)
)
WITH
(
 CLUSTERED COLUMNSTORE INDEX ORDER (OrderId)
)
-- Create Clustered Columnstore Index on existing table
CREATE CLUSTERED COLUMNSTORE INDEX cciOrderId
ON dbo.OrderTable ORDER (OrderId)
```

```
-- Insert data into table with ordered columnstore index
INSERT INTO sortedOrderTable
VALUES (1, '01-01-2019','Dave', 'UK')
```

Tables – Distributions

Round-robin distributed

Distributes table rows evenly across all distributions at random.

Hash distributed

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

Replicated

Full copy of table accessible on each Compute node.

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

Tables – Partitions

Overview

Table partitions divide data into smaller groups

In most cases, partitions are created on a date column

Supported on all table types

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

Benefits

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

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

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

Tables – Distributions & Partitions

Logical table structure

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

Physical data distribution

(Hash distribution (OrderId), Date partitions)

Distribution1

(OrderId 80,000 – 100,000)

11-2-2018 partition

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

11-3-2018 partition

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

• • •

x 60 distributions (shards)

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

Common table distribution methods

Table Category	Recommended Distribution Option
Fact	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.

Materialized views

Overview

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

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

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

Supported aggregations: MAX, MIN, AVG, COUNT, COUNT_BIG, SUM, VAR, STDEV

Benefits

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

High availability and resiliency as regular tables

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

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

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

Materialized views - example

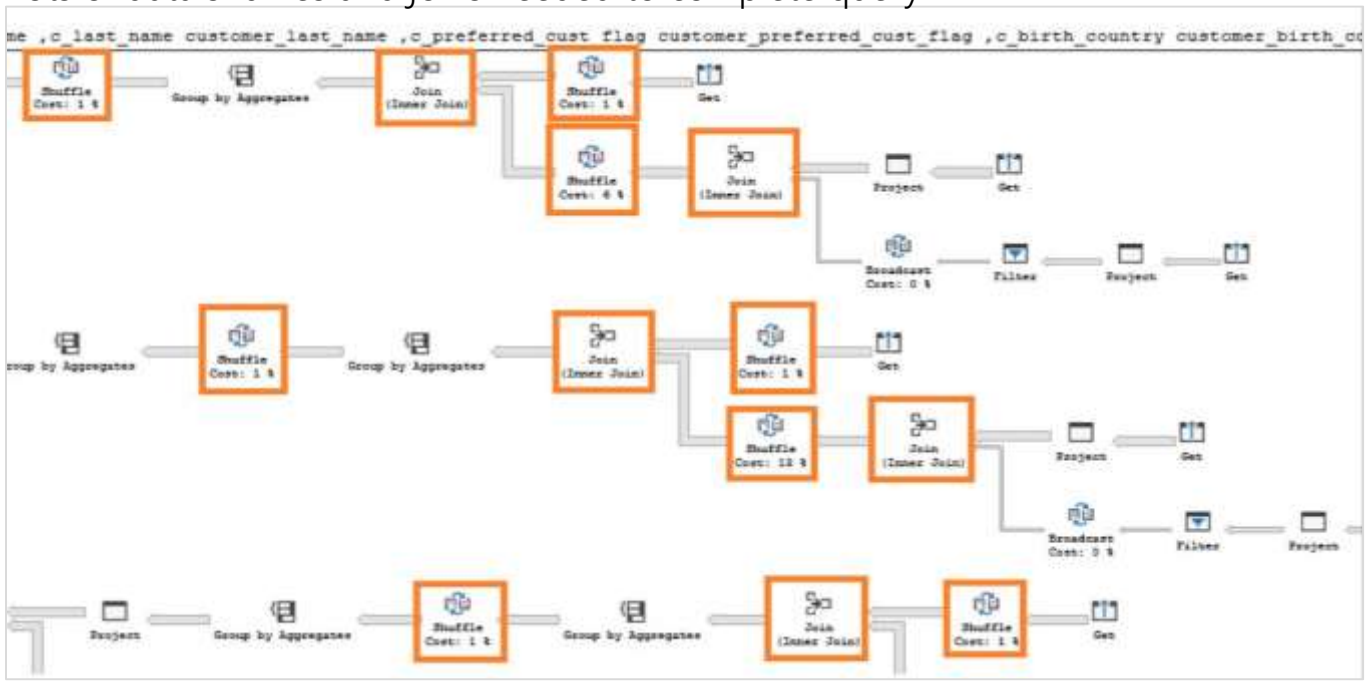
In this example, a query to get the year total sales per customer is shown to have a lot of data shuffles and joins that contribute to slow performance:

No relevant indexed views created on the data warehouse

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

Execution time: 103 seconds

Lots of data shuffles and joins needed to complete query



Materialized views - example

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

Original query – get year total sales per customer

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

Create indexed view with hash distribution on customer_id column

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

Indexed (materialized) views - example

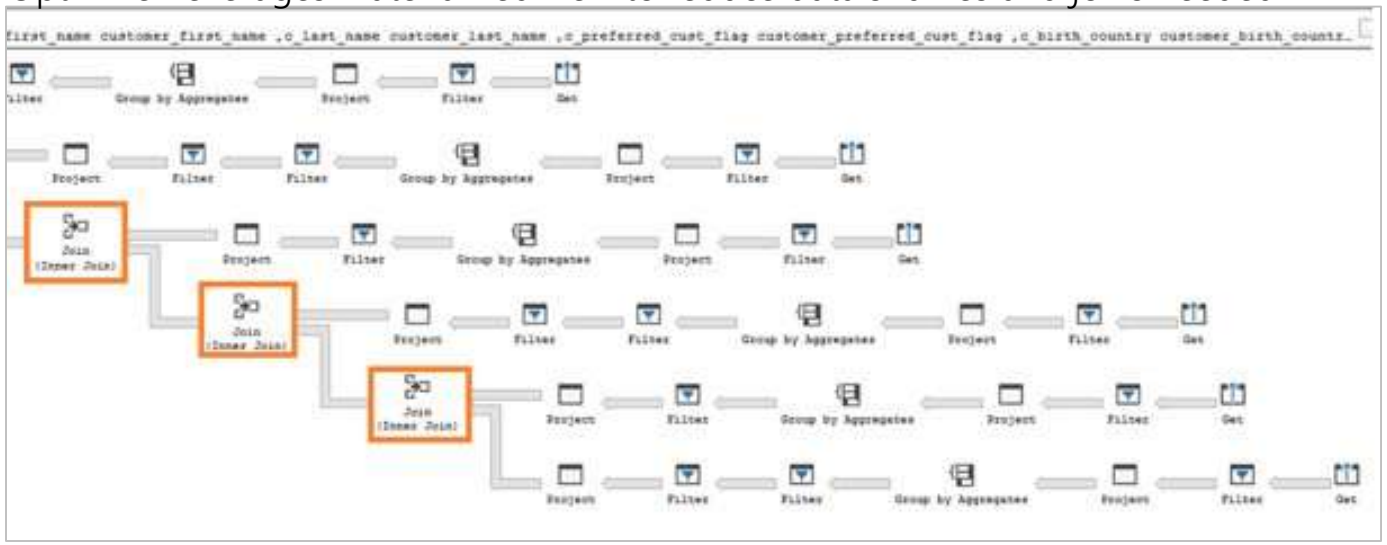
SQL pool query optimizer automatically leverages the indexed view to speed up the same query. Notice that the query does not need to reference the view directly

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
```

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 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

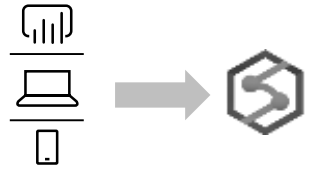
Enhances performance when same result is requested repetitively

Reduced load on server for repeated queries

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

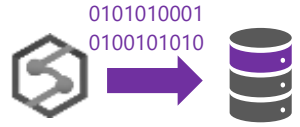
```
-- Turn on/off result-set caching for a database
-- Must be run on the MASTER database
ALTER DATABASE {database_name}
SET RESULT_SET_CACHING { ON | OFF }

-- 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

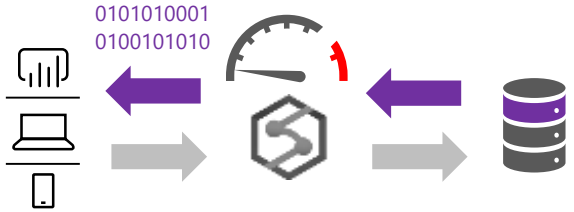


1. Client sends query to SQL pool

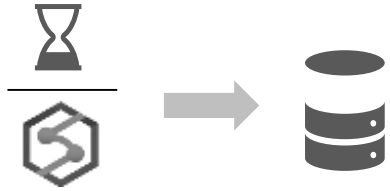
2. Query is processed using compute nodes which pull data from remote storage, process query and output back to client app



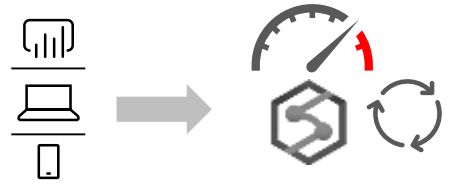
Query results are cached in remote storage so subsequent requests can be served immediately



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

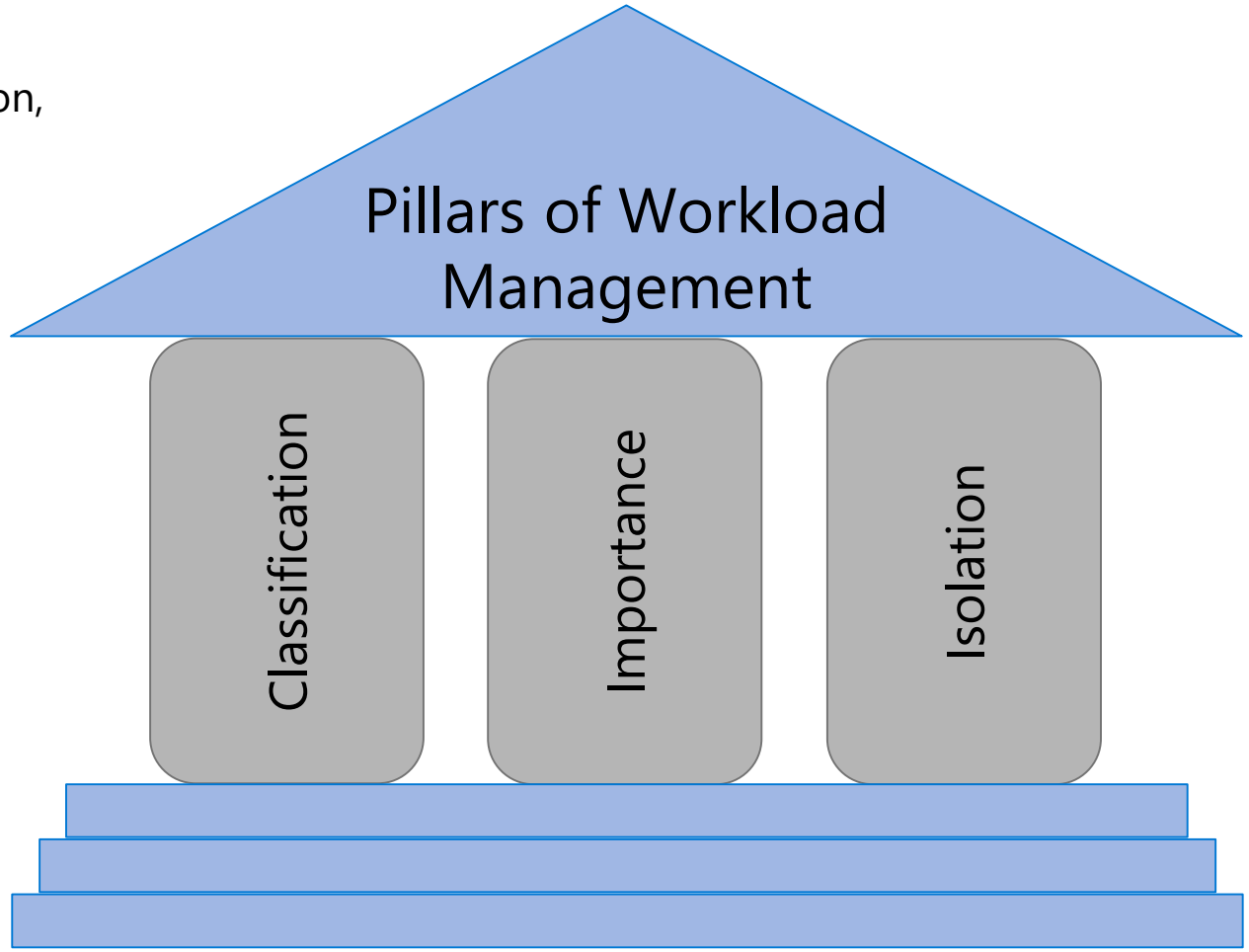
Workload Management

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

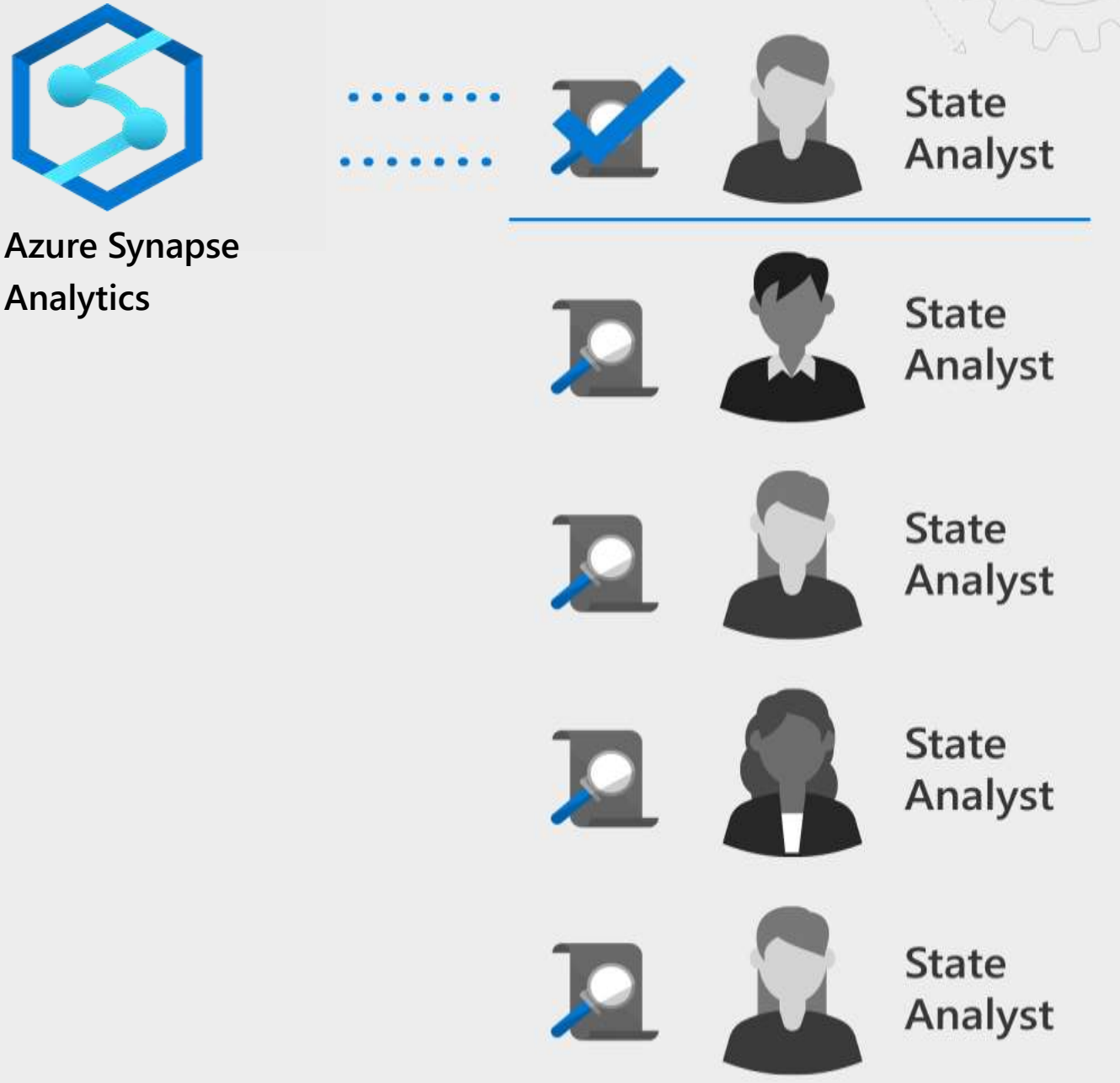
State analysts have normal importance.

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 SQL Analytics offline.

Benefits

Reserve resources for a group of requests

Limit the amount of resources a group of requests can consume

Shared resources accessed based on importance level

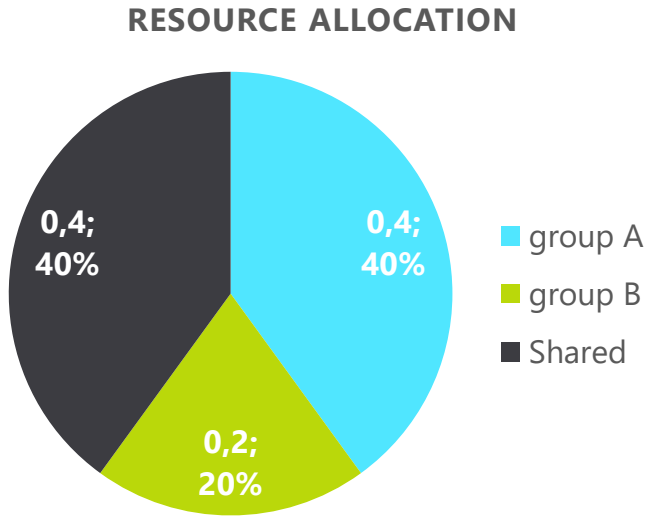
Set Query timeout value. Get DBAs out of the business of killing runaway queries

Monitoring DMVs

sys.workload_management_workload_groups

Query to view configured workload group.

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



Dynamic Management Views (DMVs)

Overview

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

Benefits:

Simple SQL syntax

Returns result in table format

Easier to read and copy result

SQL Monitor with DMVs

Overview

Offers monitoring of

-all open, closed sessions

-count sessions by user

-count completed queries by user

-all active, complete queries

-longest running queries

-memory consumption

Count sessions by user

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

List all open sessions

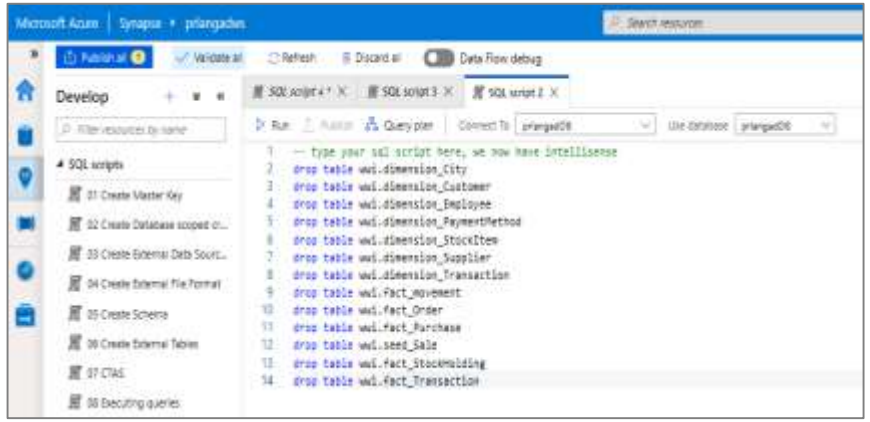
```
-- List all open sessions
SELECT * FROM sys.dm_pdw_exec_sessions where status <> 'Closed' and session_id <>
session_id();
```

List all active queries

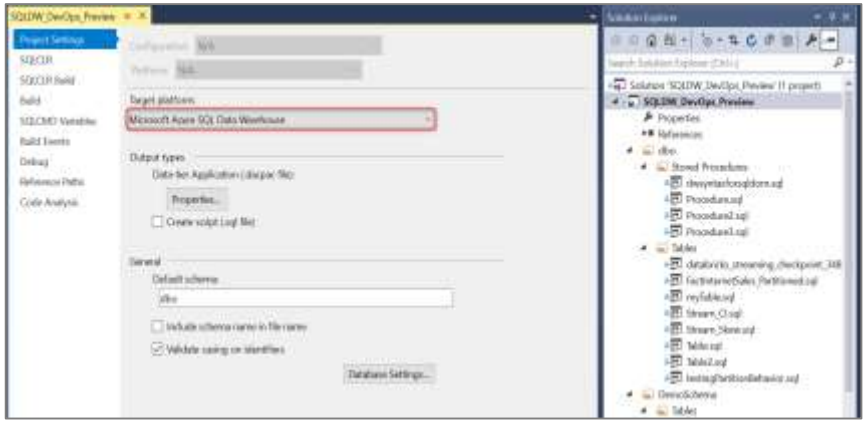
```
-- List all active queries
SELECT * FROM sys.dm_pdw_exec_requests WHERE status not in
('Completed','Failed','Cancelled') AND session_id <> session_id() ORDER BY submit_time
DESC;
```

Developer Tools

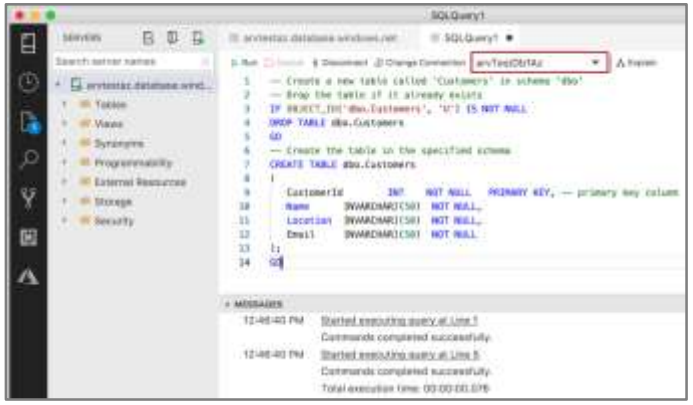
Azure Synapse Analytics



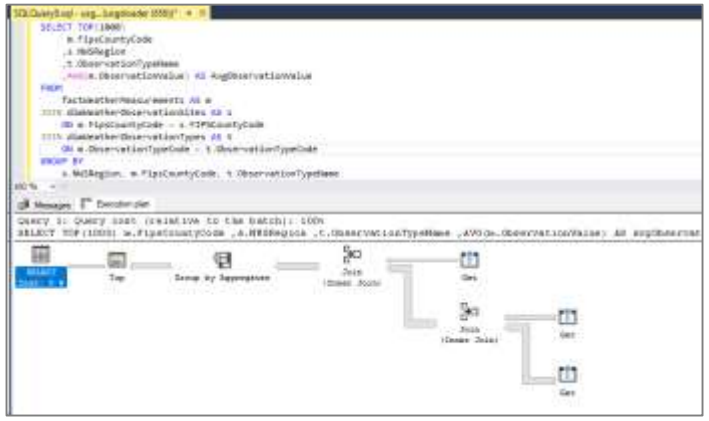
Visual Studio - SSDT database projects



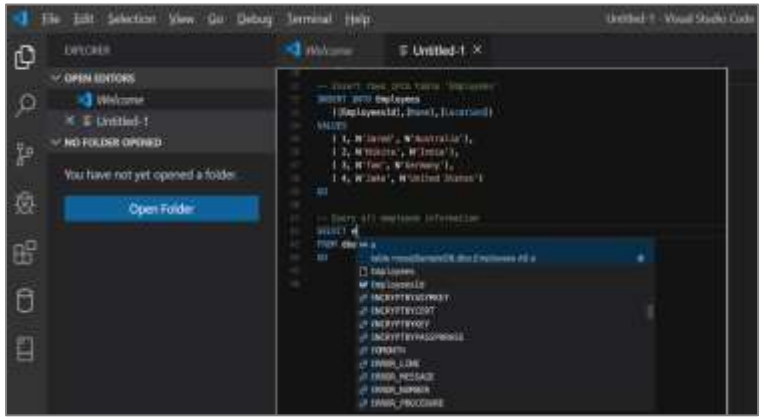
Azure Data Studio (queries, extensions etc.)



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

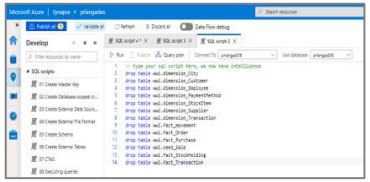


Visual Studio Code



Developer Tools

Azure Synapse Analytics

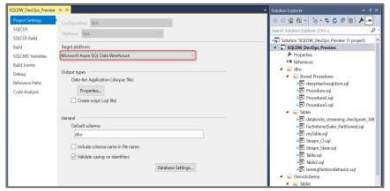


Azure Cloud Service

Offers end-to-end lifecycle for analytics

Connects to multiple services

Visual Studio - SSDT database projects



Runs on Windows

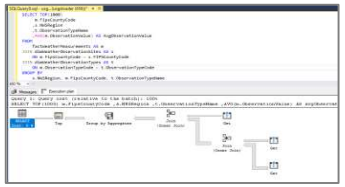
Create, maintain database code, compile, code refactoring

Azure Data Studio

Runs on Windows, Linux, macOS

Light weight editor, (queries and extensions)

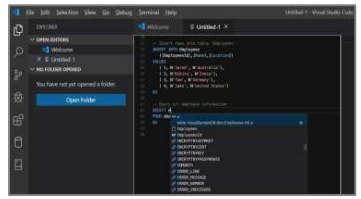
SQL Server Management Studio



Runs on Windows

Offers GUI support to query, design and manage

Visual Studio Code



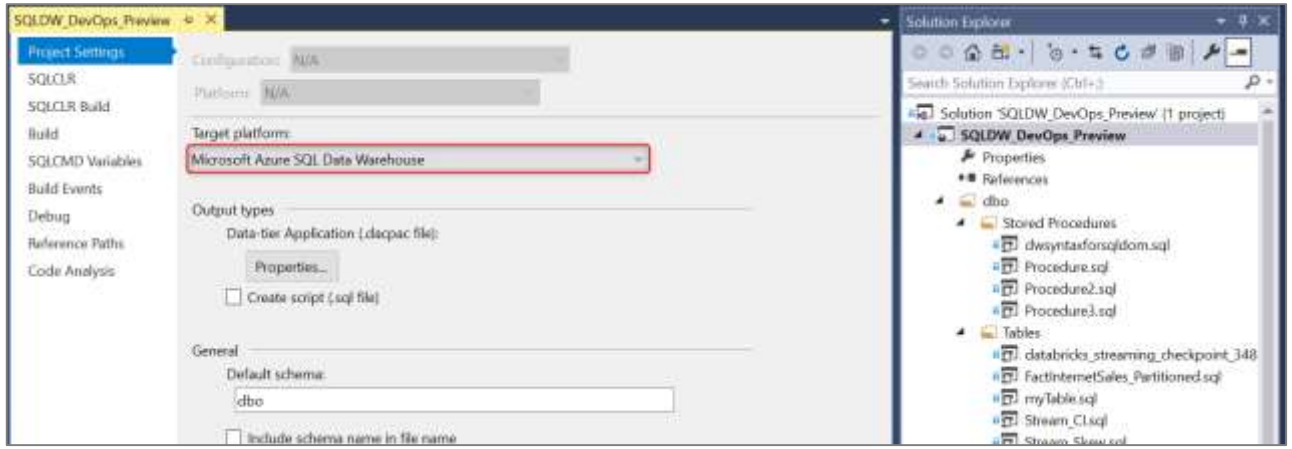
Runs on Windows, Linux, macOS

Offers development experience with light-weight code editor

Continuous integration and delivery (CI/CD)

Overview

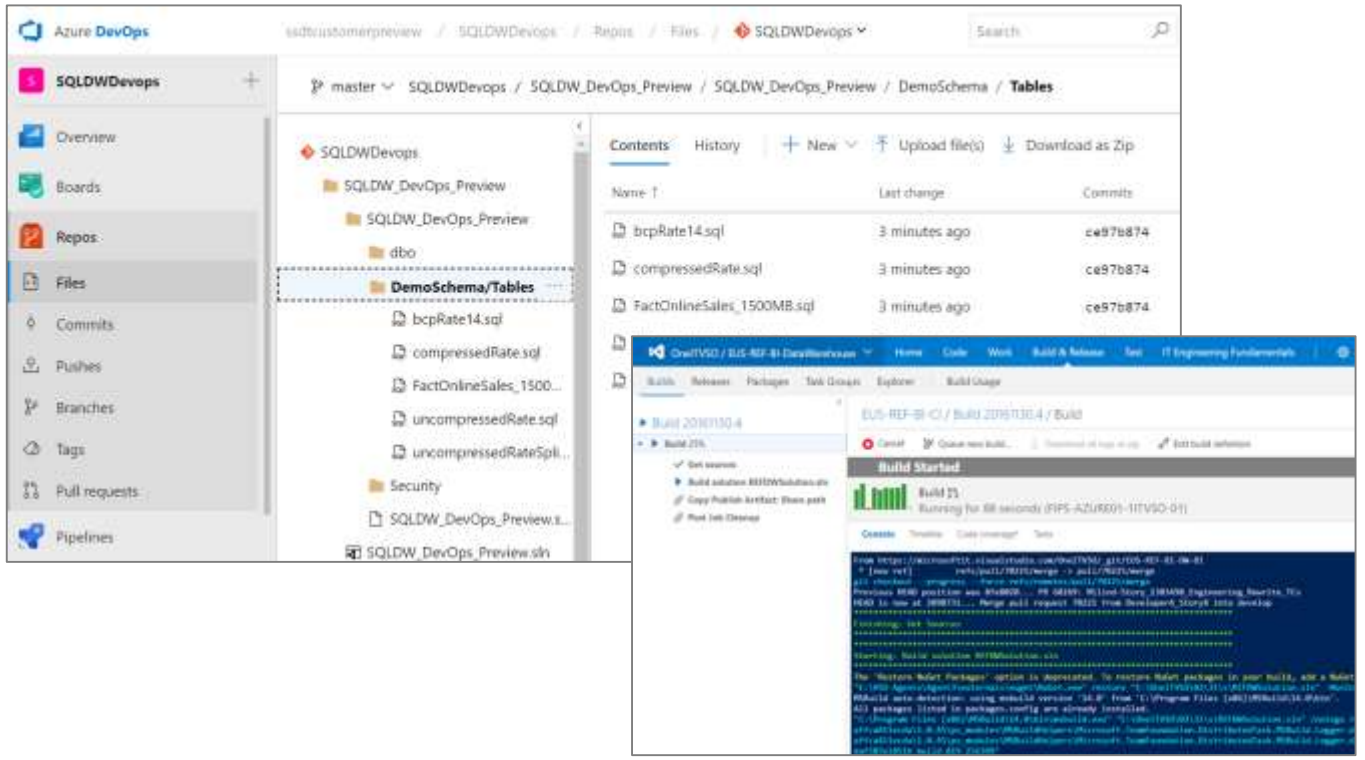
Database project support in SQL Server Data Tools (SSDT) allows teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy and test schema changes.



Benefits

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

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



Azure Advisor recommendations

Suboptimal Table Distribution

Reduce data movement by replicating tables

Data Skew

Choose new hash-distribution key

Slowest distribution limits performance

Cache Misses

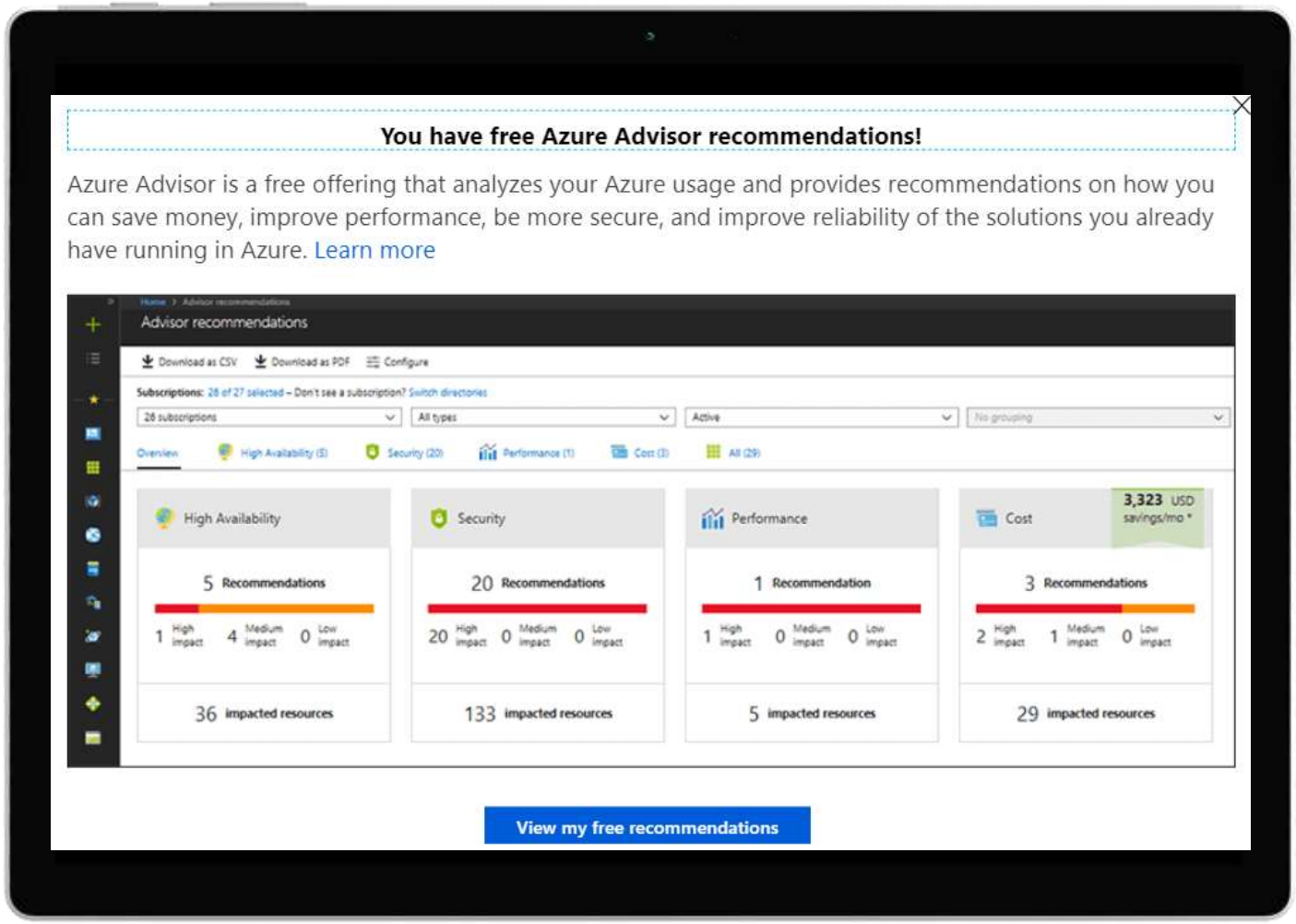
Provision additional capacity

Tempdb Contention

Scale or update user resource class

Suboptimal Plan Selection

Create or update table statistics



Maintenance windows

Overview

Choose a time window for your upgrades.

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

Windows can be from 3 to 8 hours.

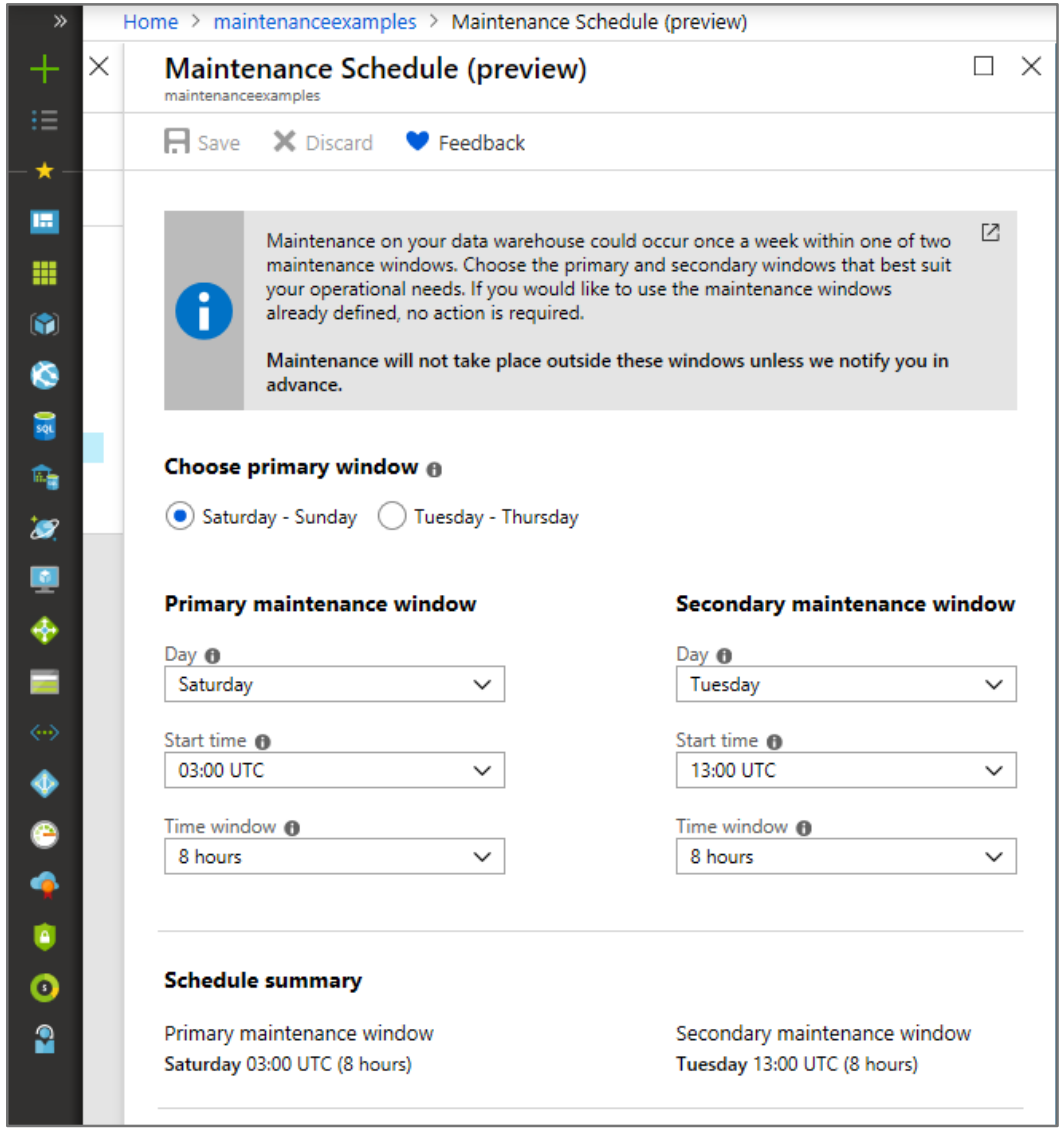
24-hour advance notification for maintenance events.

Benefits

Ensure upgrades happen on your schedule.

Predictable planning for long-running jobs.

Stay informed of start and end of maintenance.



Automatic statistics management

Overview

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

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

Statistics are considered out of date when:

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

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

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

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

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

SQL On-Demand

SQL On Demand

SQL On-Demand

Overview

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

Benefits

Serverless

No infrastructure

Pay only for query execution

No ETL

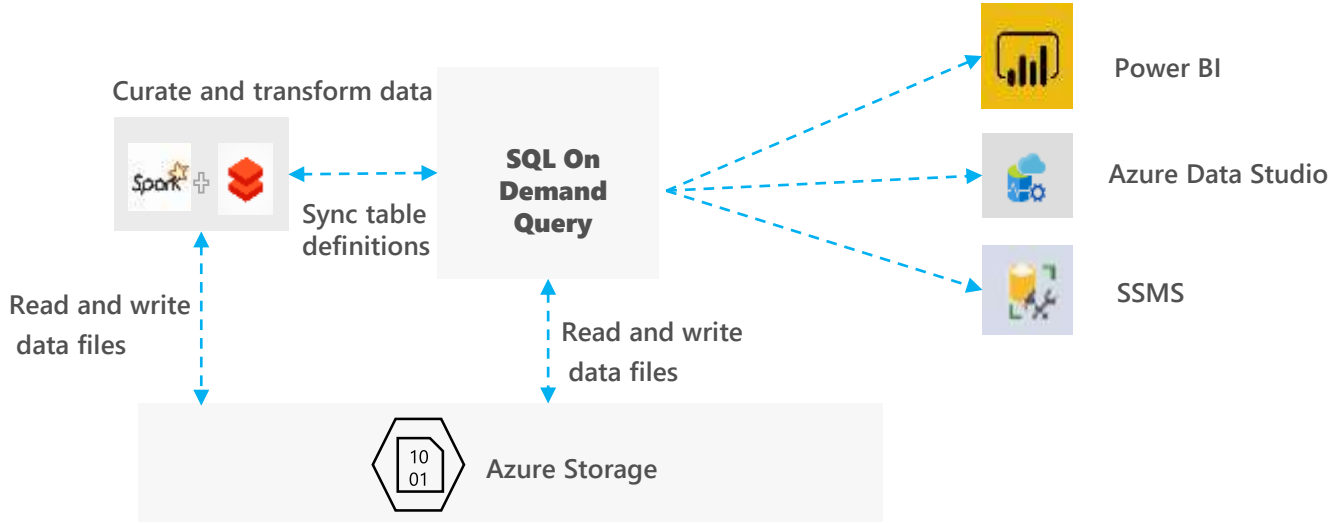
Offers security

Data integration with Databricks, HDInsight

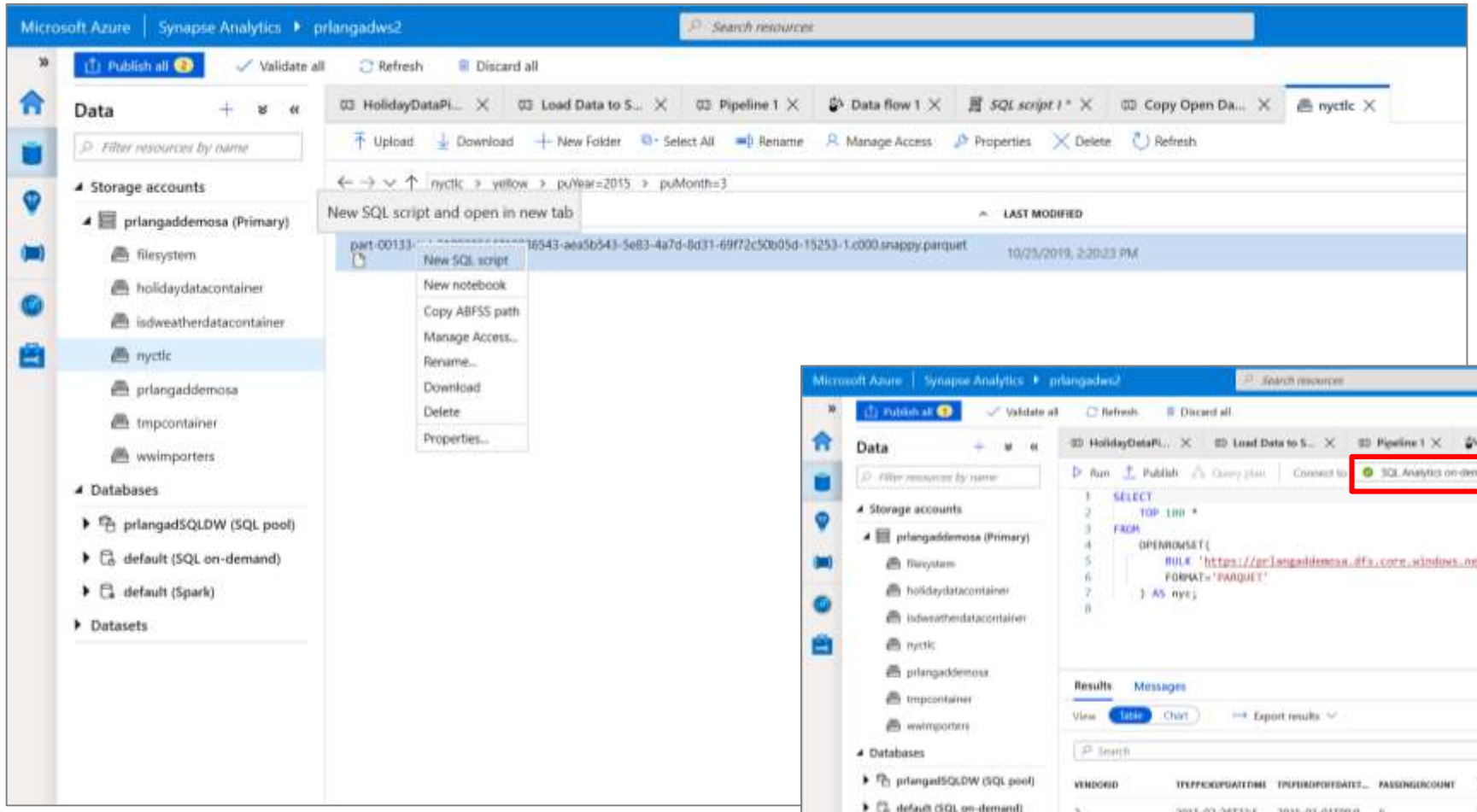
T-SQL syntax to query data

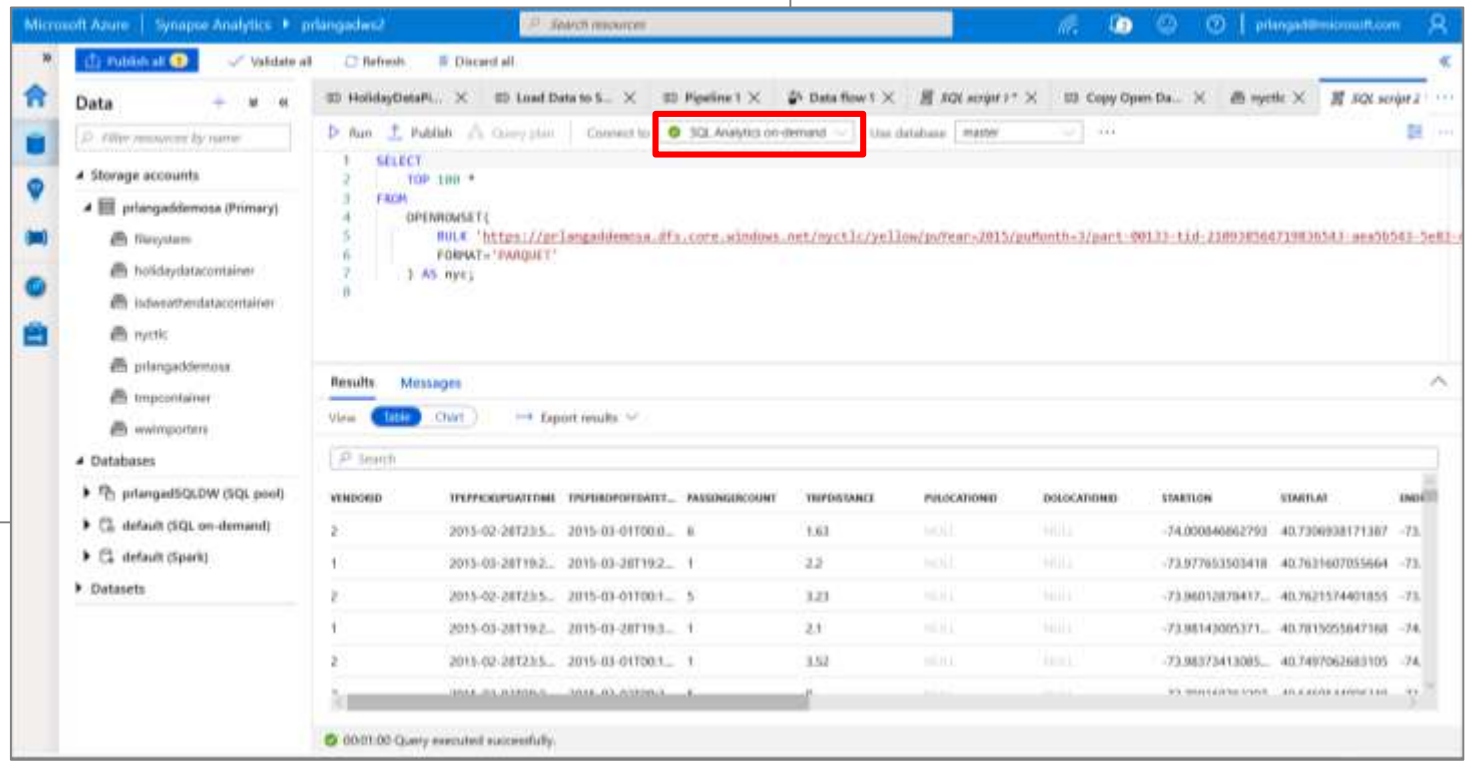
Supports data in various formats (Parquet, CSV, JSON)

Support for BI ecosystem



SQL On Demand – Querying on storage





SQL On Demand – Querying CSV File

Overview

Uses OPENROWSET function to access data

Benefits

Ability to read CSV File with

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

SQL On Demand – Querying CSV File

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Read CSV file - without specifying all columns

```
SELECT
   COUNT(DISTINCT country_name) AS countries
FROM OPENROWSET(
      BULK 'https://XXX.blob.core.windows.net/csv/population/population.csv
',
      FORMAT = 'CSV',
      FIELDTERMINATOR =',',
      ROWTERMINATOR = '\n'
   )
WITH (
      [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2 2
) AS [r]
```

	countries
1	228

SQL On Demand – Querying folders

Overview

Uses OPENROWSET function to access data from multiple files or folders

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

```
SELECT YEAR(pickup_datetime) as [year], SUM(passenger_count) AS passengers_total,
COUNT(*) AS [rides_total]
FROM OPENROWSET(
BULK 'https://XXX.blob.core.windows.net/csv/taxi/*.*',
FORMAT = 'CSV'
, FIRSTROW = 2 )
WITH (
        vendor_id VARCHAR(100) COLLATE Latin1_General_BIN2,
        pickup_datetime DATETIME2,
        dropoff_datetime DATETIME2,
        passenger_count INT,
        trip_distance FLOAT,
        rate_code INT,
        store_and_fwd_flag VARCHAR(100) COLLATE Latin1_General_BIN2,
        pickup_location_id INT,
        dropoff_location_id INT,
        payment_type INT,
        fare_amount FLOAT,
        extra FLOAT, mta_tax FLOAT,
        tip_amount FLOAT,
        tolls_amount FLOAT,
        improvement_surcharge FLOAT,
        total_amount FLOAT
        ) AS nyc
GROUP BY YEAR(pickup_datetime)
ORDER BY YEAR(pickup_datetime)
```

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

SQL On Demand – Querying folders

Read all files from multiple folders

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

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

Read subset of files in folder

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

	payment_type	fare_total
1	1	1026072325.579…
2	2	441093322.8000…
3	3	10435183.04
4	4	3304550.99
5	5	14

SQL On Demand – Querying specific files

Overview

filename – Provides file name that originates row result

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

Benefits

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

Example of filename function

```
SELECT
    r.filename() AS [filename]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2017-1*.csv',
        FORMAT = 'CSV',
        FIRSTROW = 2
    )
    WITH (
        vendor_id INT,
        pickup_datetime DATETIME2,
        dropoff_datetime DATETIME2,
        passenger_count SMALLINT,
        trip_distance FLOAT,
        <...columns>
    ) AS [r]

GROUP BY r.filename()

ORDER BY [filename]
```

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

SQL On Demand – Querying specific files

Example of filepath function

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

WHERE r.filepath(1) IN ('2017')
    AND r.filepath(2) IN ('10', '11', '12')

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

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

SQL On Demand – Querying Parquet files

Overview

Uses OPENROWSET function to access data

Benefits

Ability to specify column names of interest

Offers auto reading of column names and data types

Provides target specific partitions using filepath function

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

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

SQL On Demand – Creating views

Overview

Create views using SQL On Demand queries

Benefits

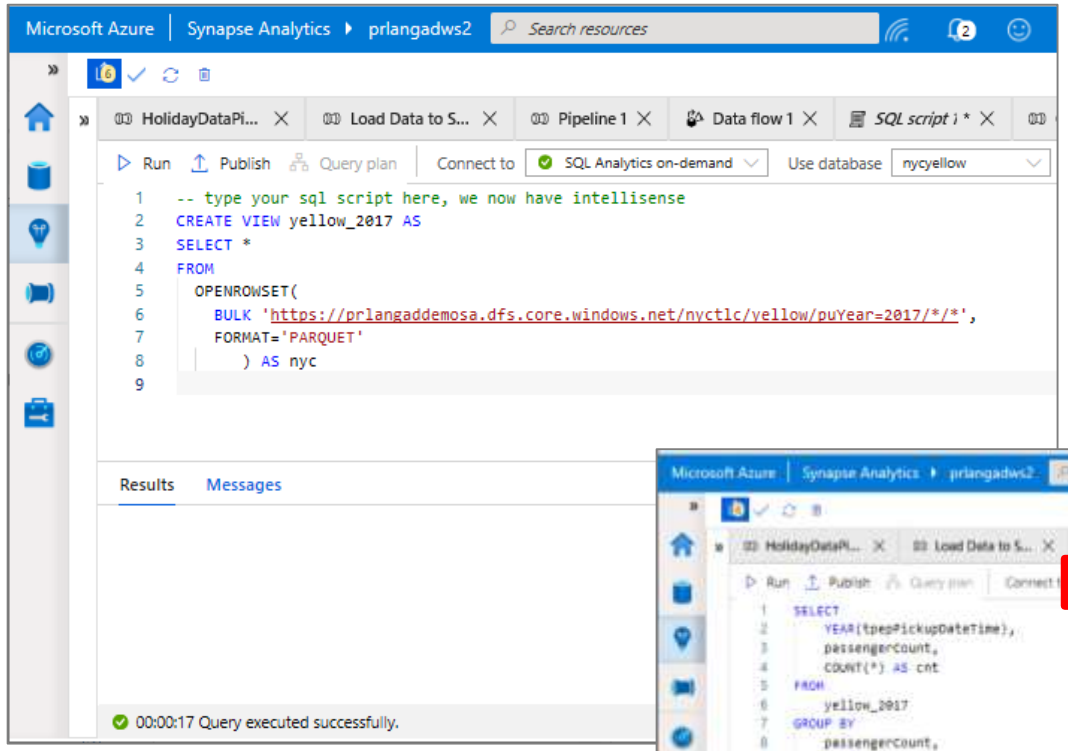
Works same as standard views

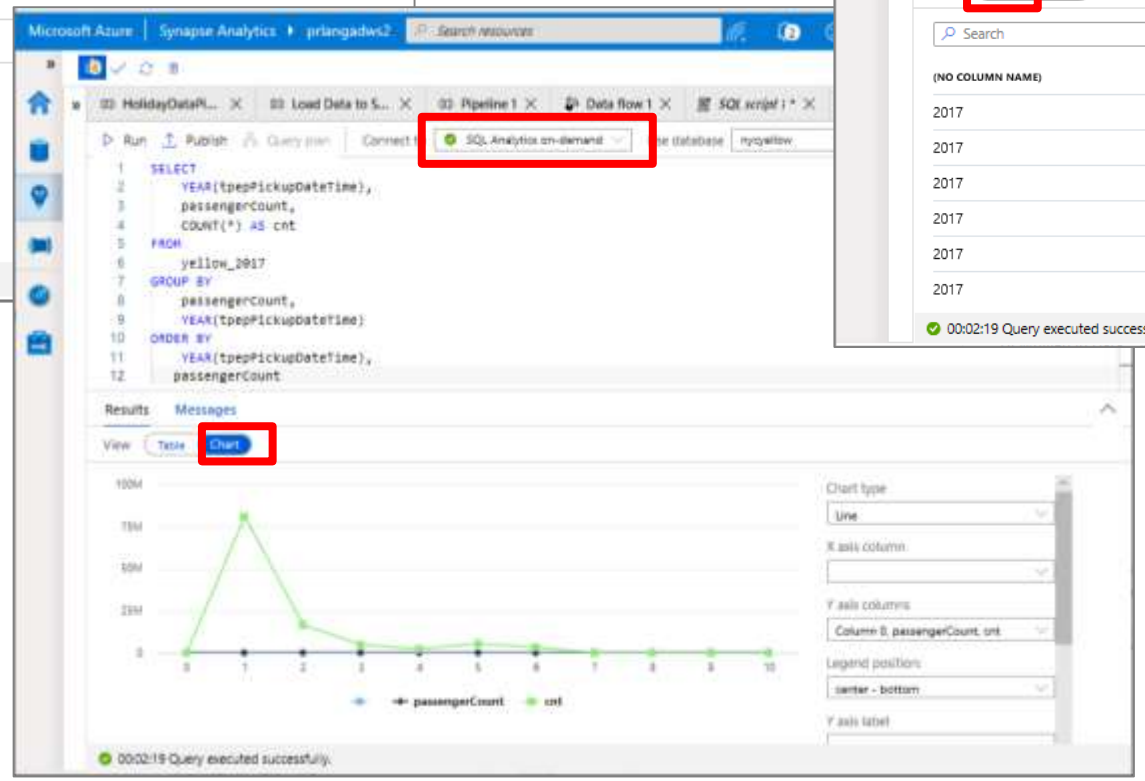
```
USE [mydbname]
GO

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

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

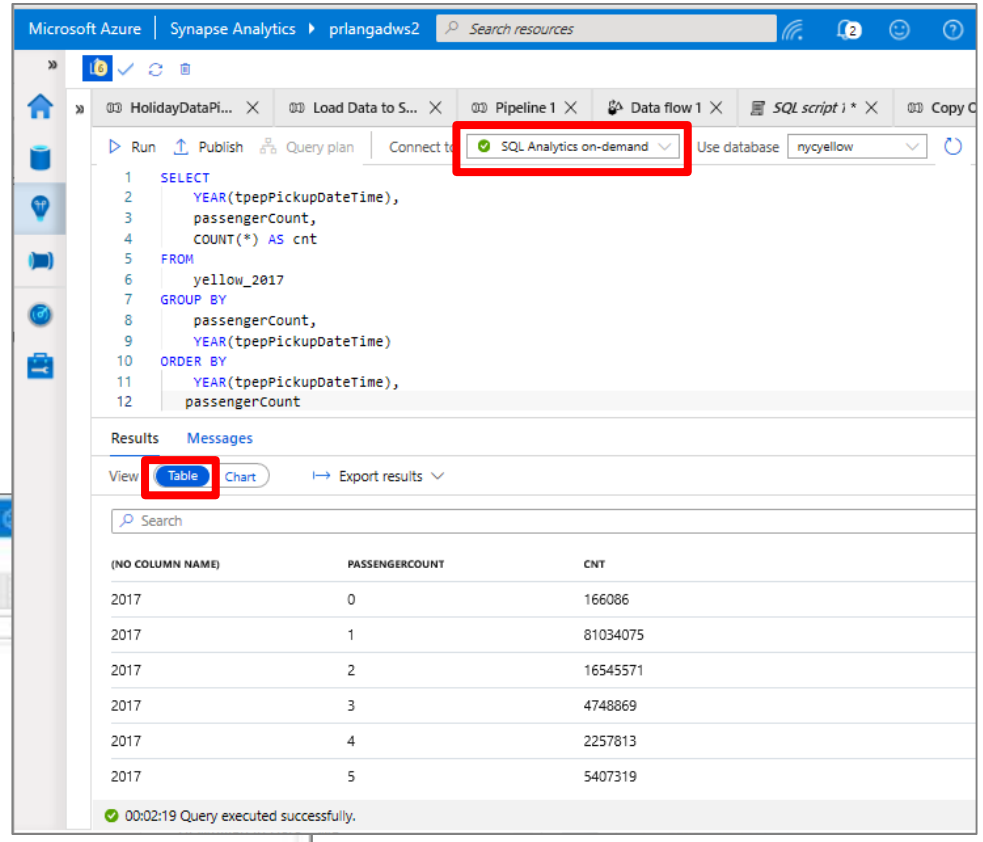
```
SELECT
   country_name, population
FROM populationView
WHERE
   [year] = 2019
ORDER BY
   [population] DESC
```

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

SQL On Demand – Creating views







SQL On Demand – Querying JSON files

Overview

Read JSON files and provides data in tabular format

Benefits

Supports OPENJSON, JSON_VALUE and JSON_QUERY functions

```
SELECT *
FROM
   OPENROWSET(
      BULK 'https://XXX.blob.core.windows.net/json/books/book1.json',
      FORMAT='CSV',
      FIELDTERMINATOR ='0x0b',
      FIELDQUOTE = '0x0b',
      ROWTERMINATOR = '0x0b'
   )
   WITH (
      jsonContent varchar(8000)
   ) AS [r]
```

	jsonContent
1	{"_id": "kim95", "type": "Book", "title": "Modern Databas…

SQL On Demand – Querying JSON files

Example of JSON_VALUE function

```
SELECT
   JSON_VALUE(jsonContent, '$.title') AS title,
   JSON_VALUE(jsonContent, '$.publisher') as publisher,
   jsonContent
FROM
   OPENROWSET(
      BULK 'https://XXX.blob.core.windows.net/json/books/*.json',
      FORMAT='CSV',
      FIELDTERMINATOR ='0x0b',
      FIELDQUOTE = '0x0b',
      ROWTERMINATOR = '0x0b'
   )
   WITH (
      jsonContent varchar(8000)
   ) AS [r]
WHERE
   JSON_V
n Introduc
```

	title	publisher	jsonContent
1	Probabilistic and Statistical Methods in Cryptology, An Int…	Springer	{"_id": "neuen…

Example of JSON_QUERY function

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

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

Create External Table As Select

Overview

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

Steps:

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

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

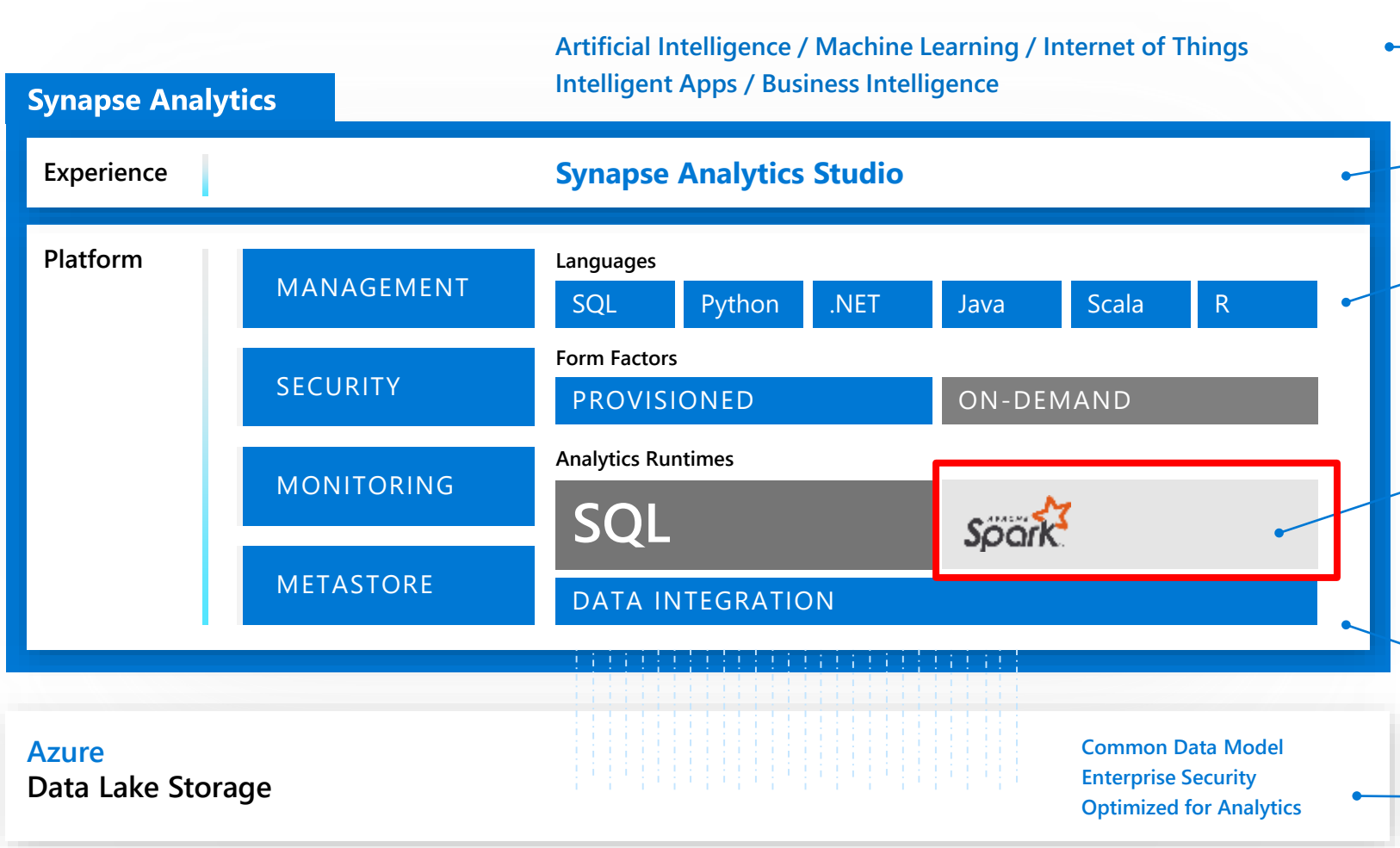
Microsoft

Azure Synapse Analytics

Spark

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available provisioned and serverless on-demand

SQL Analytics offering T-SQL for batch, streaming and interactive processing

Spark for big data processing with Python, Scala, R and .NET

Integrated **platform services** for, management, security, monitoring, and metastore

Data **lake integrated** and Common Data Model aware

Azure Synapse Apache Spark - Summary

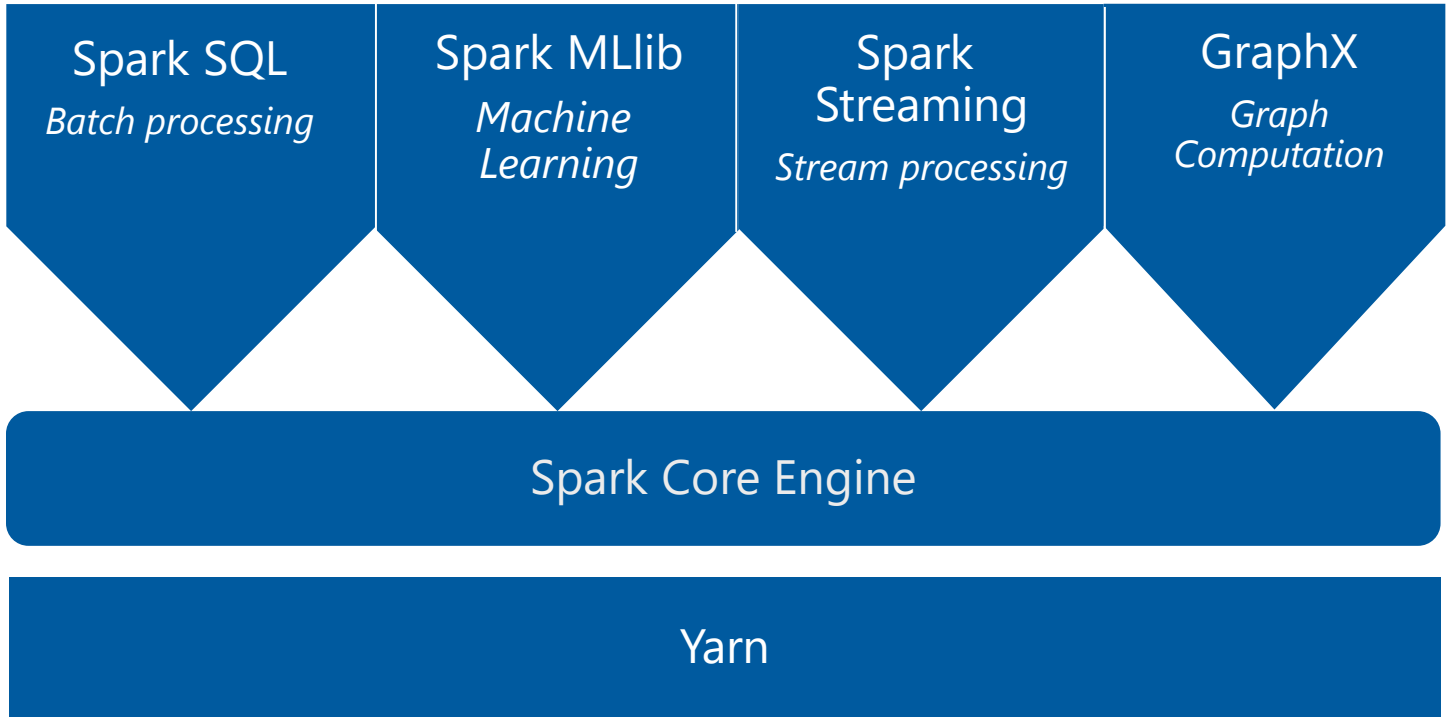
- Apache Spark 2.4 derivation
  - Linux Foundation Delta Lake 0.4 support
  - .Net Core 3.0 support
  - Python 3.6 + Anacondas support
- Tightly coupled to other Azure Synapse services
  - Integrated security and sign on
  - Integrated Metadata
  - Integrated and simplified provisioning
  - Integrated UX including nteract based notebooks
  - Fast load of SQL Analytics pools
- Core scenarios
  - Data Prep/Data Engineering/ETL
  - Machine Learning via Spark ML and Azure ML integration
  - Extensible through library management
- Efficient resource utilization
  - Fast Start
  - Auto scale (up and down)
  - Auto pause
  - Min cluster size of 3 nodes
- Multi Language Support
  - .Net (C#), PySpark, Scala, Spark SQL, Java

Apache Spark

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

Spark Unifies:

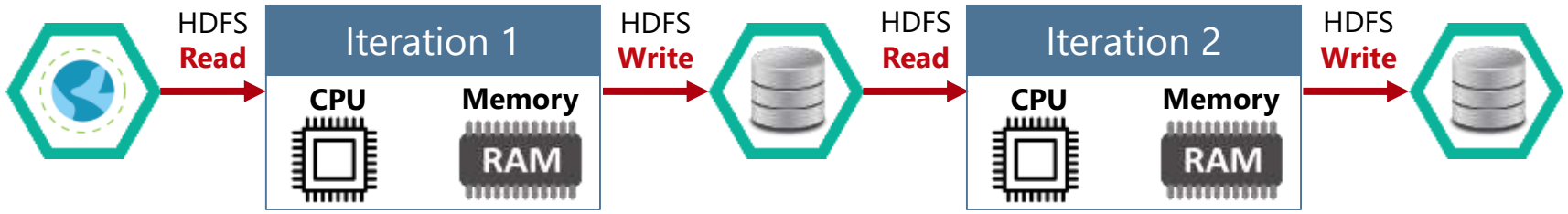
- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing



http://spark.apache.org

Motivation for Apache Spark

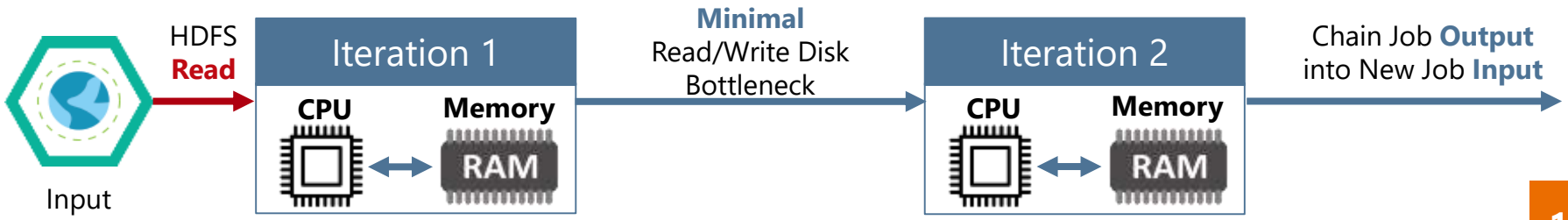
Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of (slow) disk I/O



Motivation for Apache Spark

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

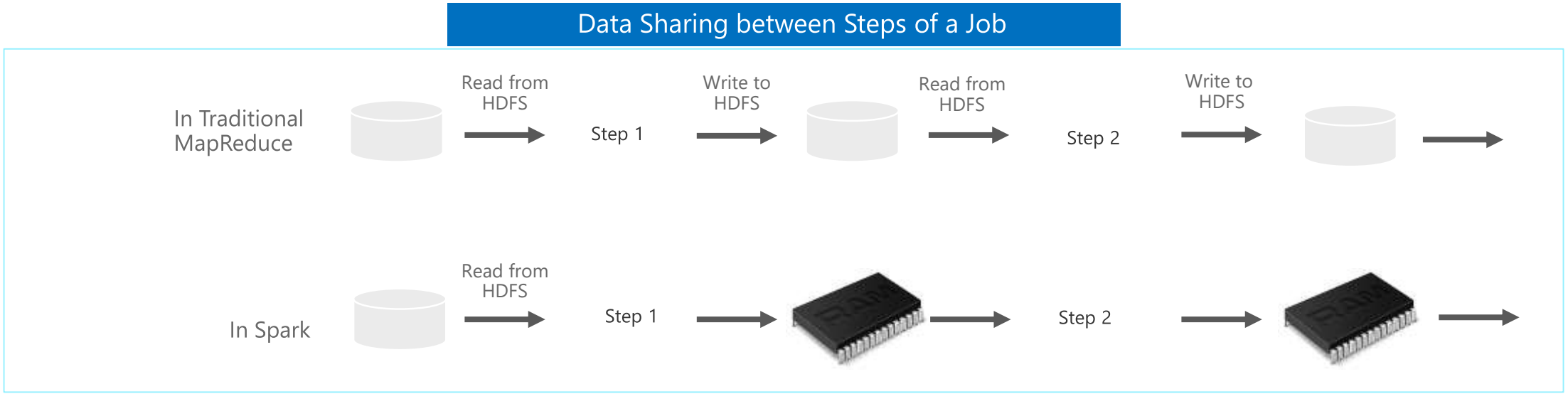
Solution: Keep data **in-memory** with a new distributed execution engine



10–100x faster than network & disk

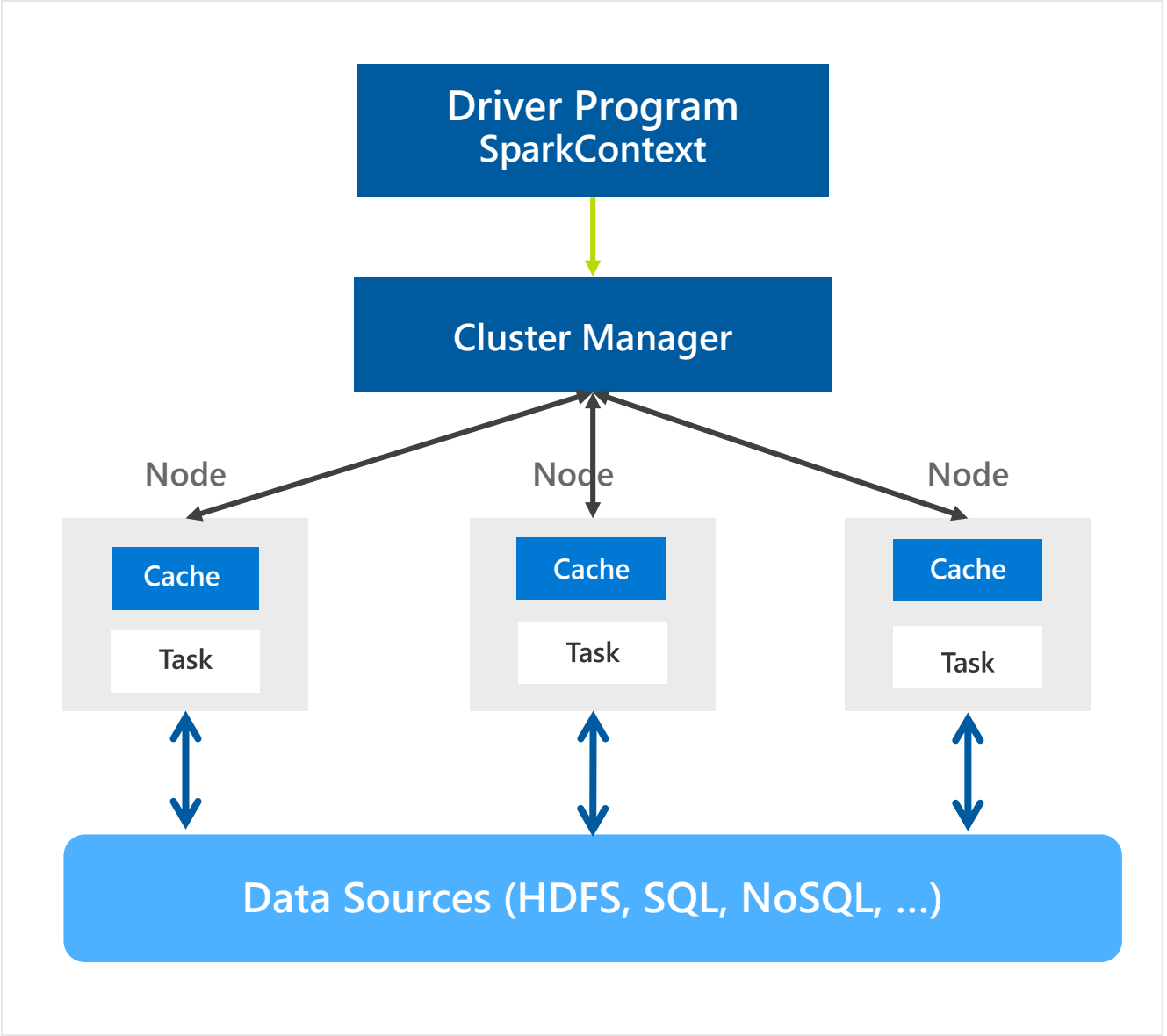
What makes Spark fast

- **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

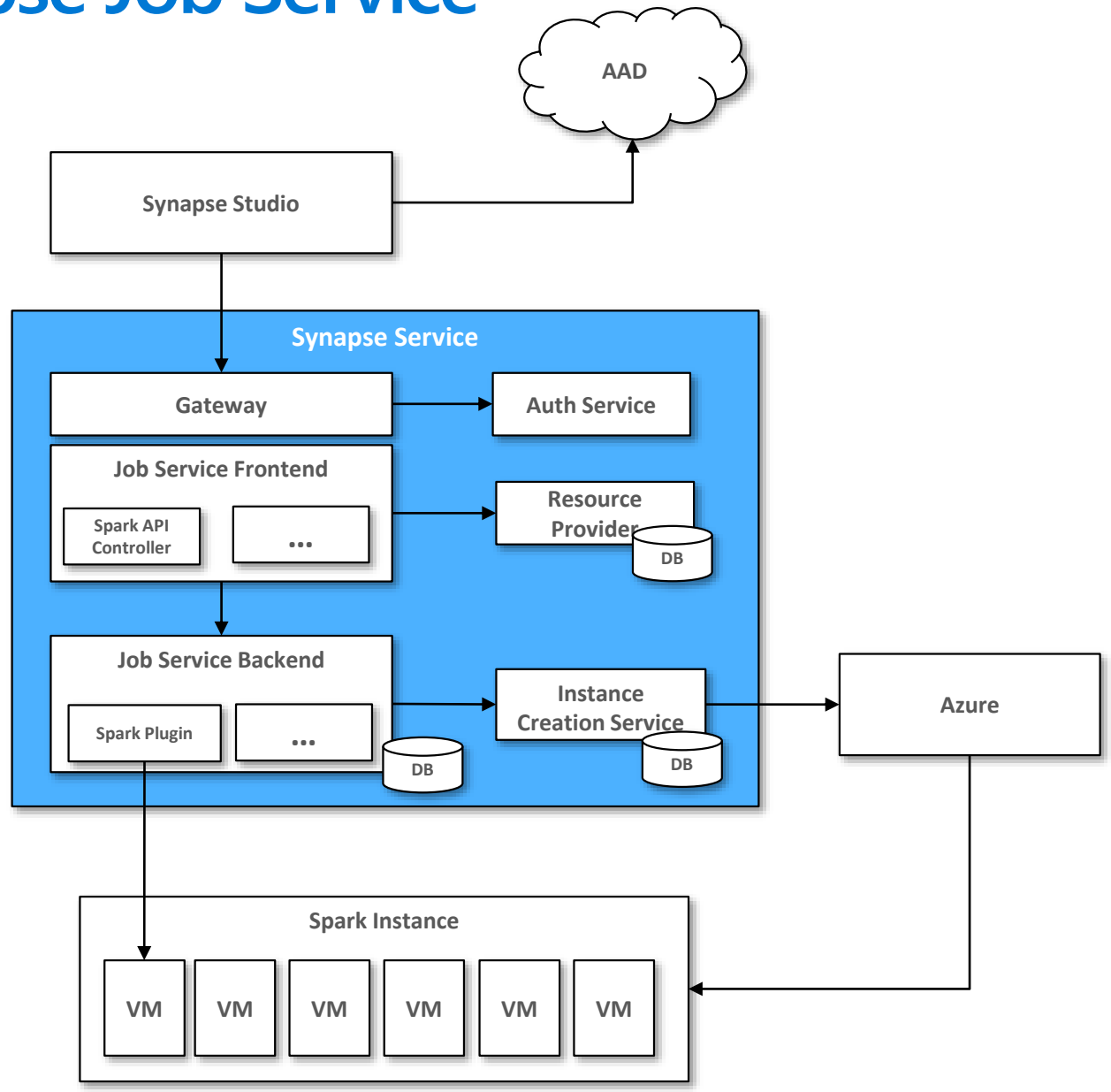
- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.

Microsoft

Azure Synapse Apache Spark

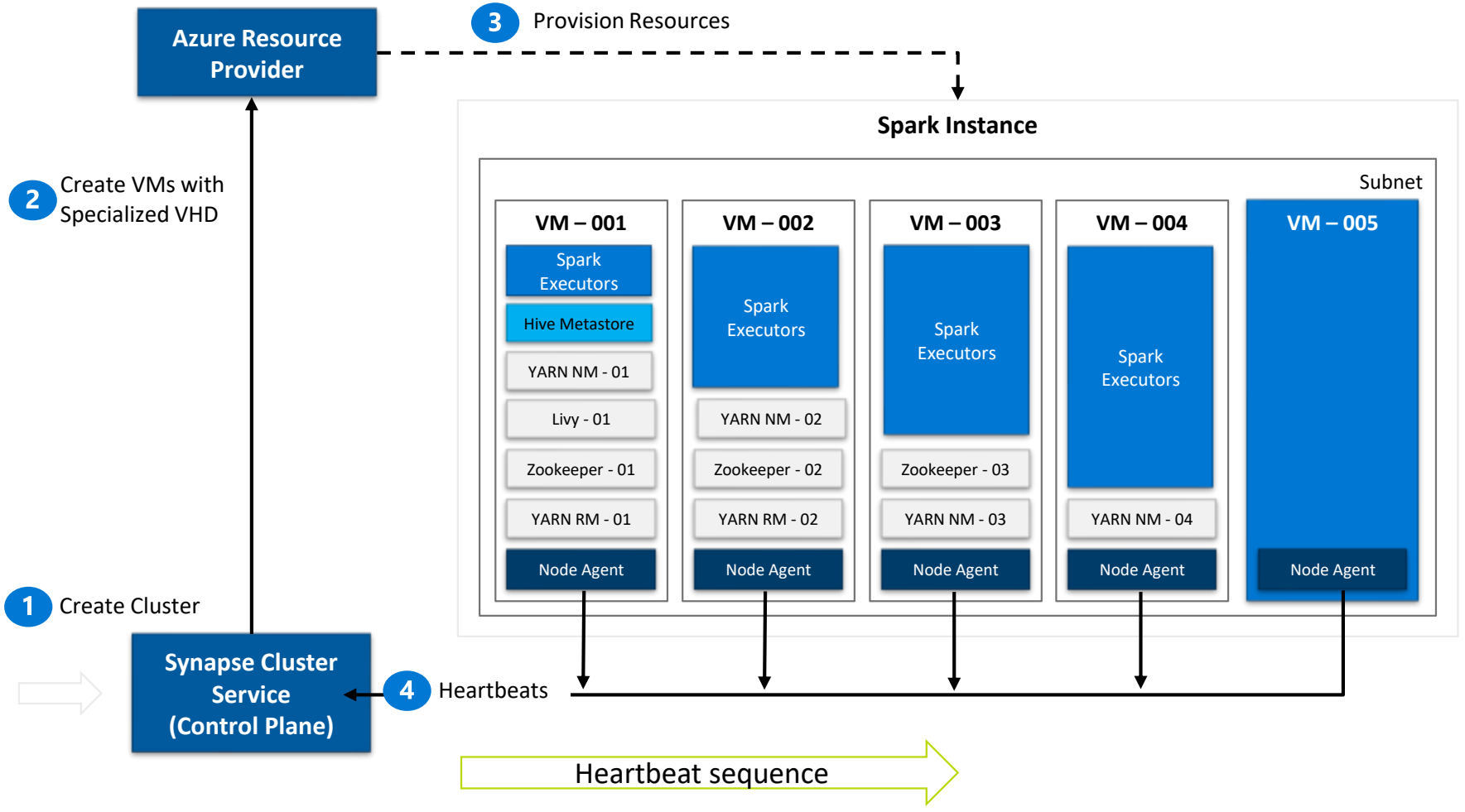
Architecture Overview

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.

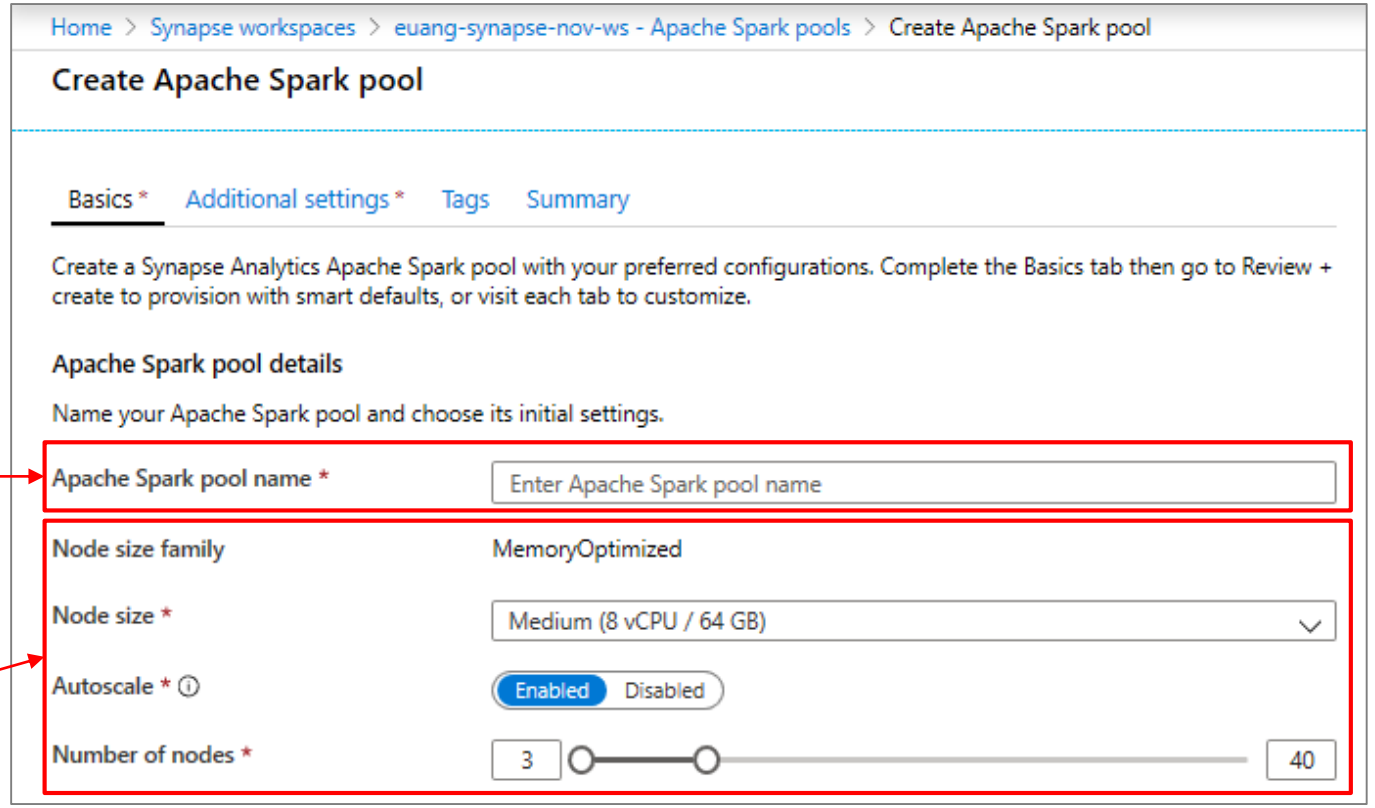
Creating a Spark pool (1 of 2)

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

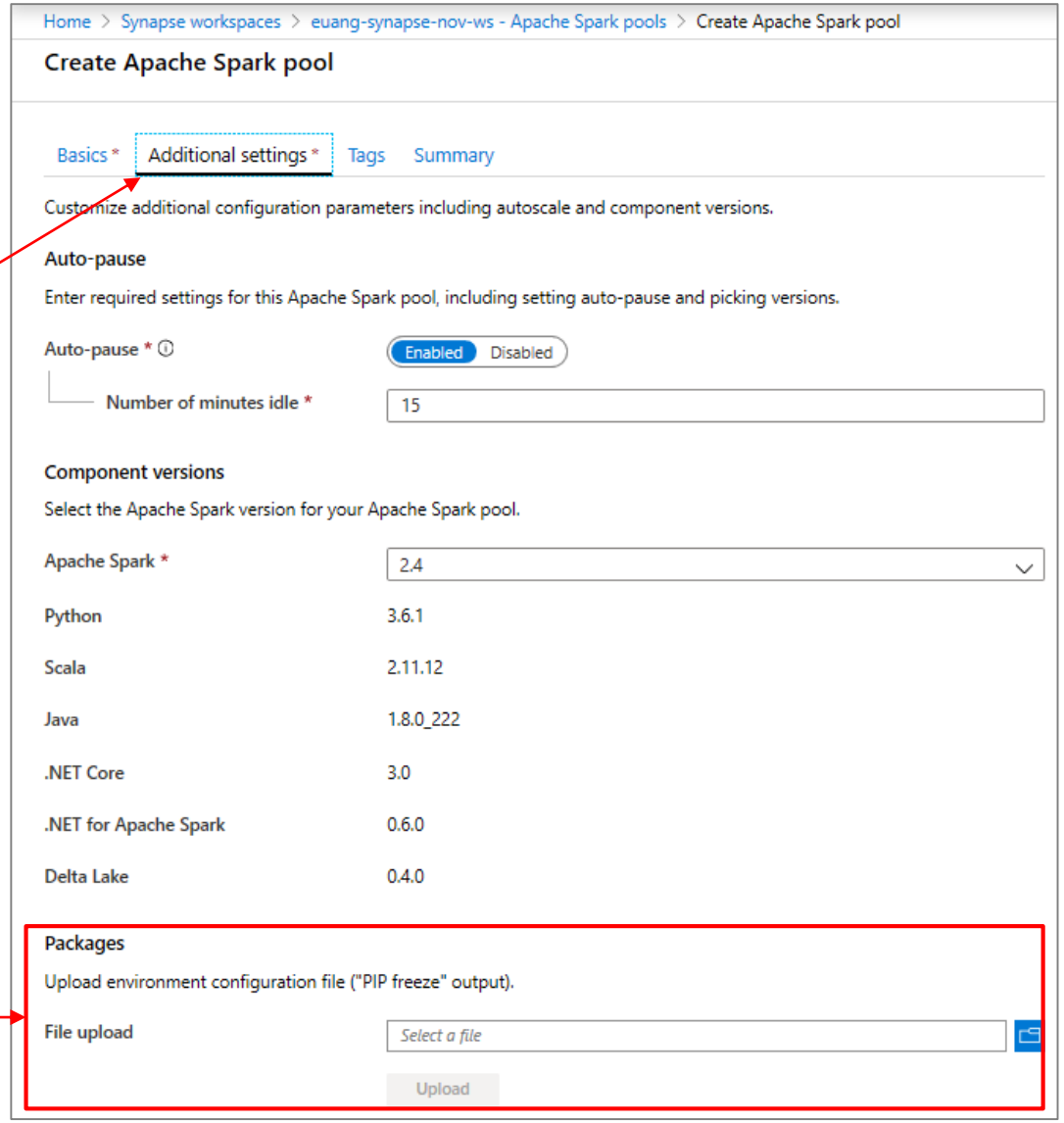


Creating a Spark pool (2 of 2) - optional

Additional Settings offer optional settings to customize Spark pool

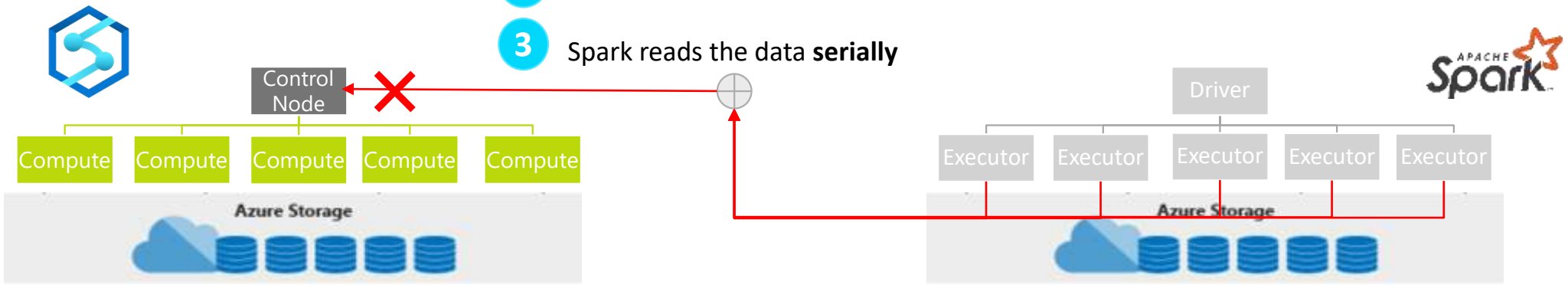
Customize component versions, auto-pause

Import libraries by providing text file containing library name and version



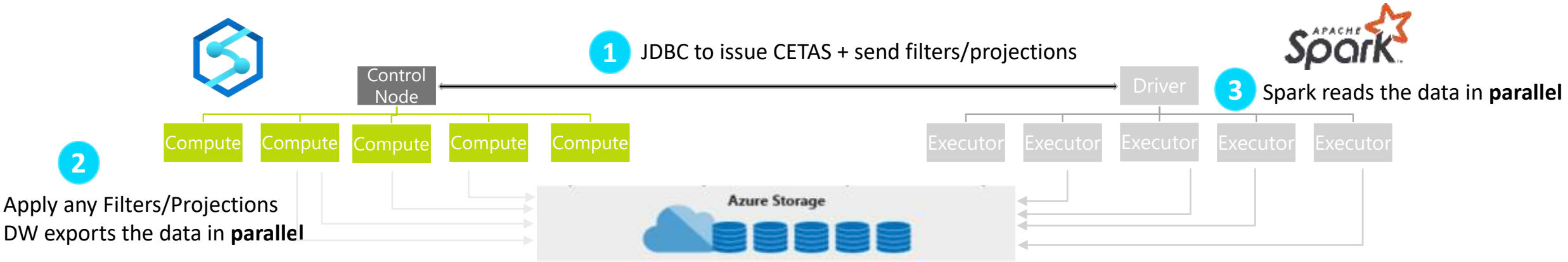
Existing Approach: JDBC

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

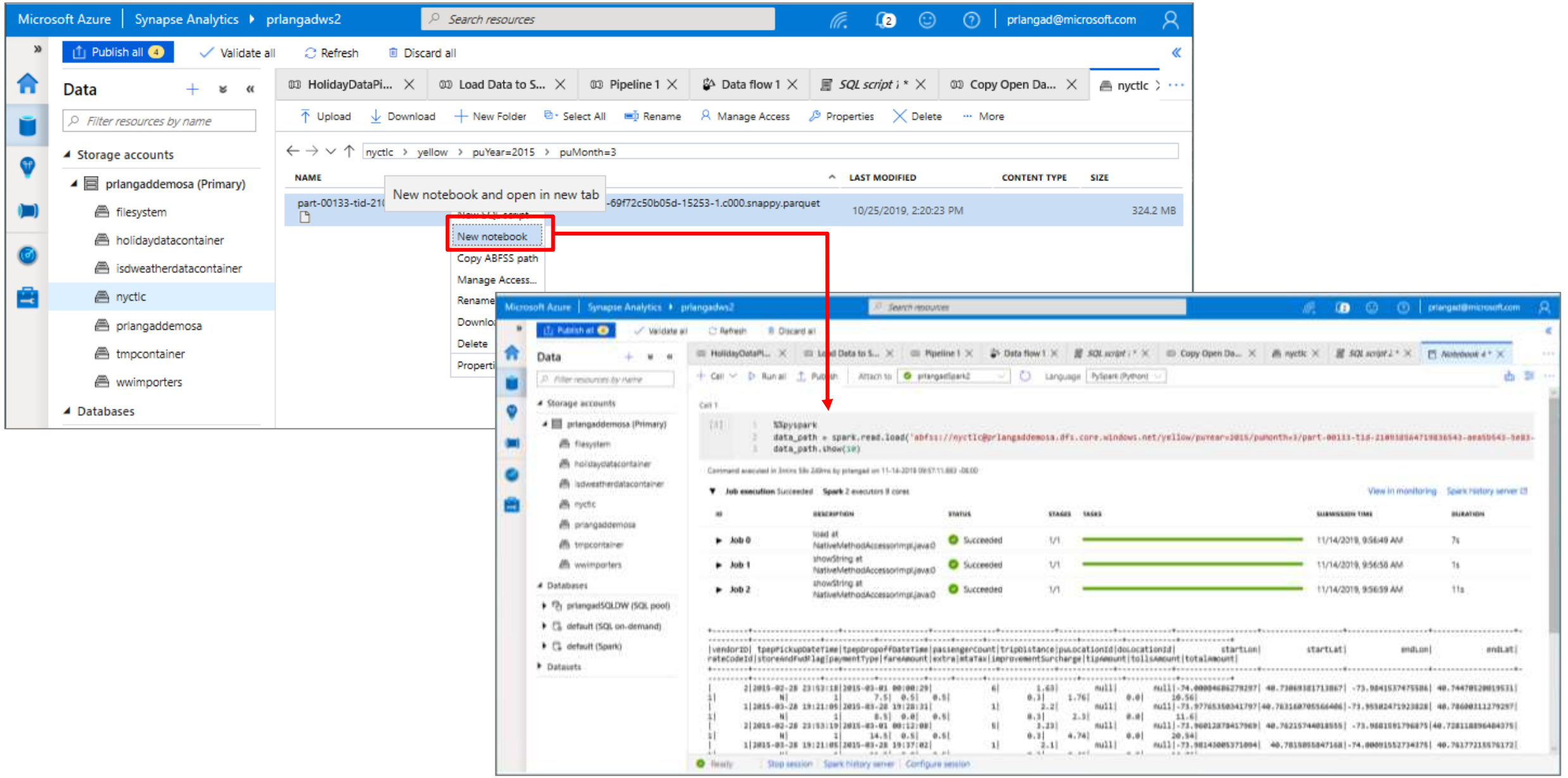
Existing Approach

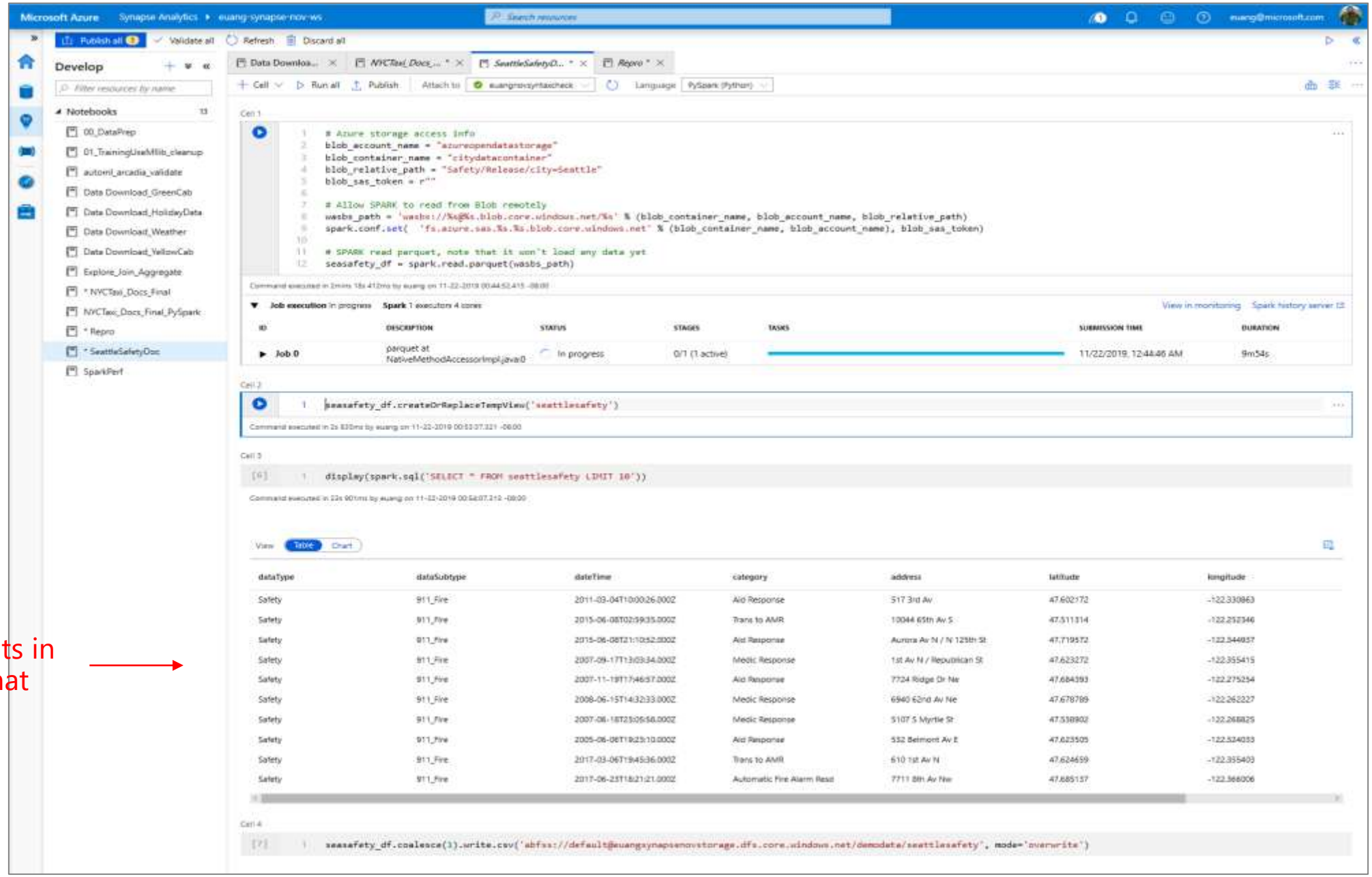
```
val jdbcUsername = "<SQL DB ADMIN USER>"
val jdbcPwd = "<SQL DB ADMIN PWD>"
val jdbcHostname = "servername.database.windows.net”
val jdbcPort = 1433
val jdbcDatabase ="<AZURE SQL DB NAME>“


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

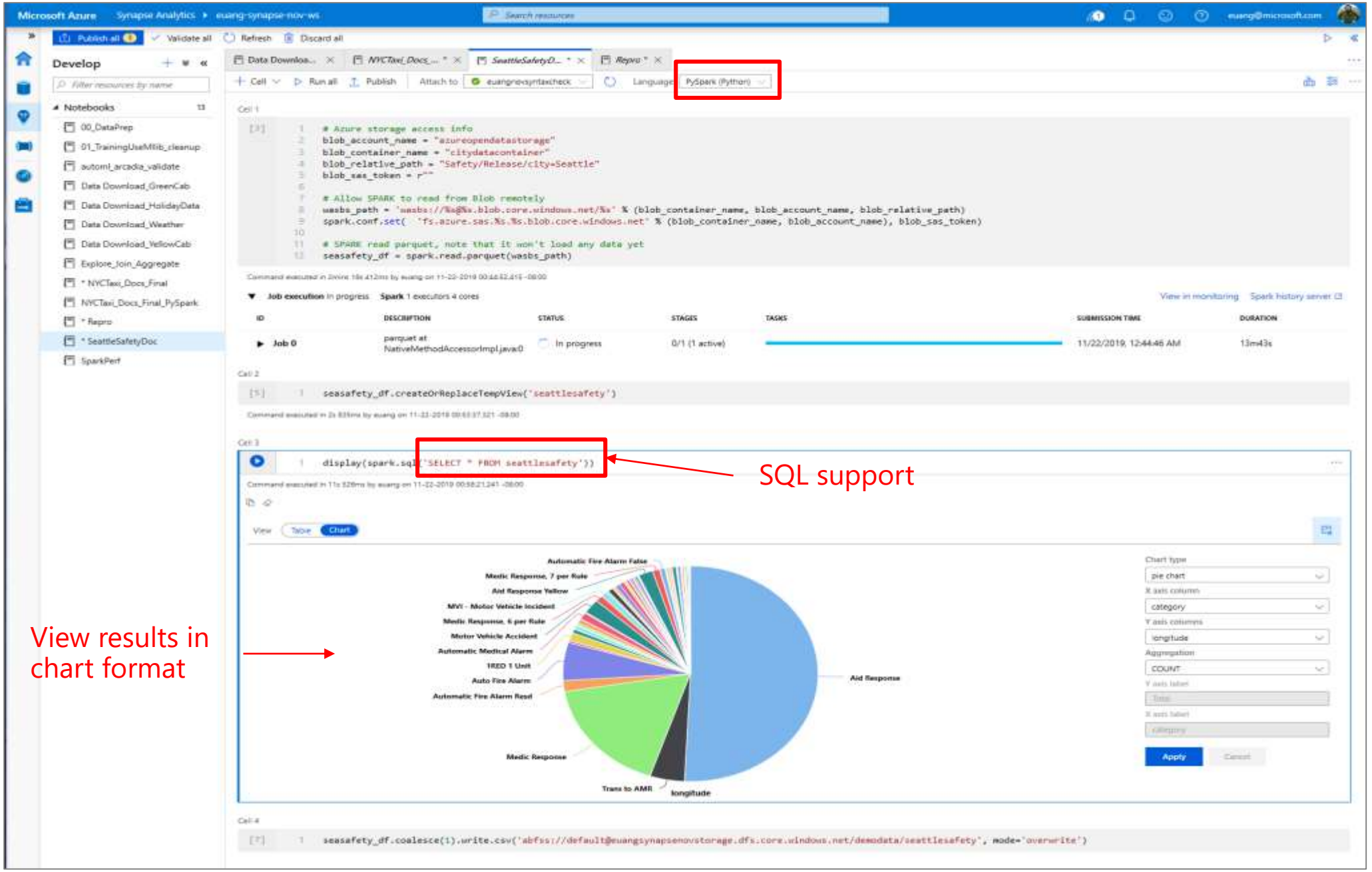
```
// Construct a Spark DataFrame from SQL Pool
var df = spark.read.sqlanalytics("sql1.dbo.Tbl1")

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

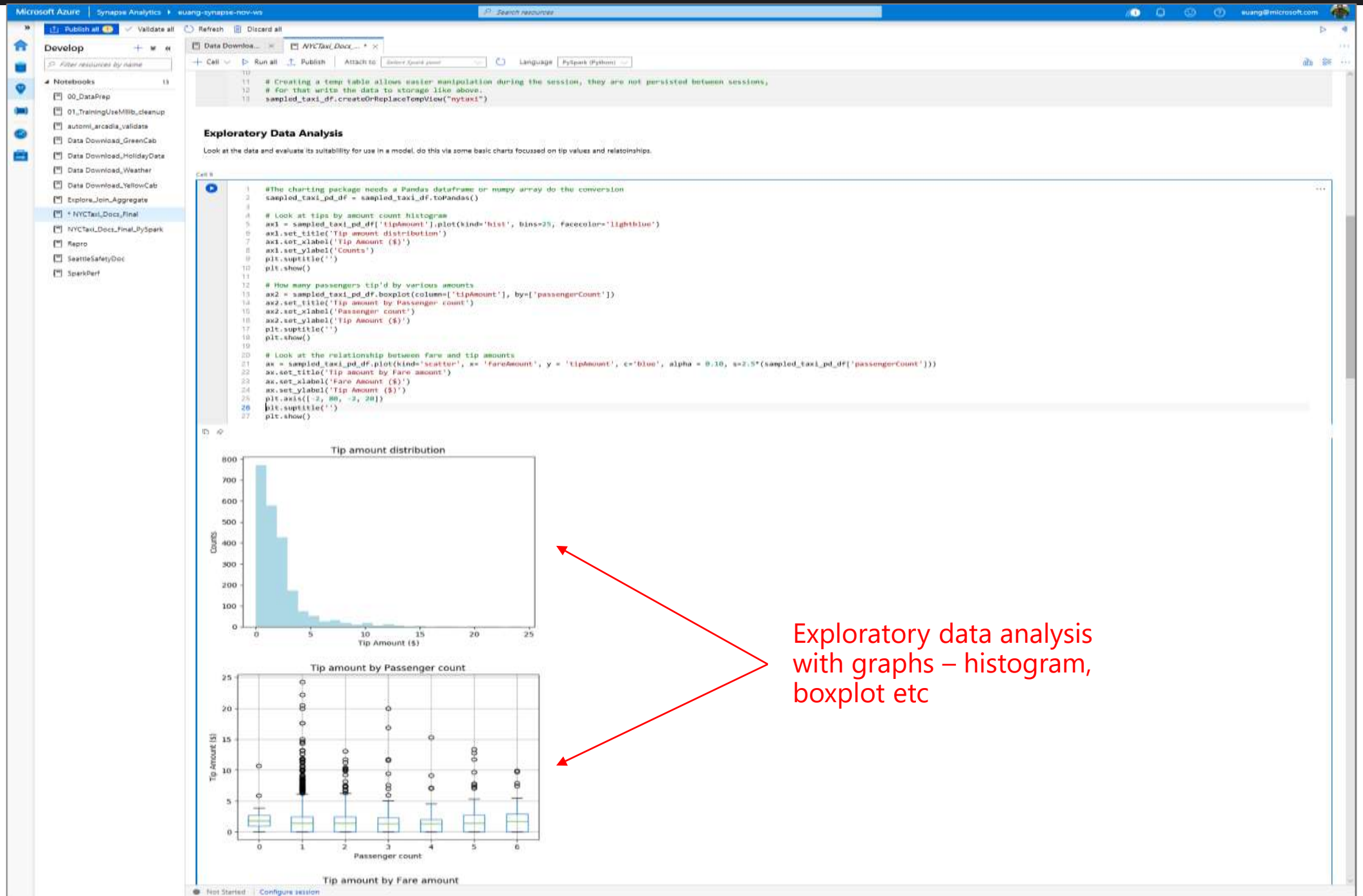
Create Notebook on files in storage





View results in table format


Microsoft Azure
Synapse Analytics
Develop
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
Publish all
Validate all
Refresh
Discard all
Cell
Run all
Publish
Attach to
euangreasyntaxcheck
Language
PySpark (Python)
# 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)
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
seasafety_df.createOrReplaceTempView('seattlesafety')
display(spark.sql('SELECT * FROM seattlesafety'))
SQL support
View
Table
Chart
View results in chart format
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
TRED 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
Apply
Cancel
seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')



Exploratory data analysis with graphs – histogram, boxplot etc

Library Management - Python

Overview

Customers can add new python libraries at Spark pool level

Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

Update versions of existing libraries on your cluster

Libraries will get installed for your Spark pool during cluster creation

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

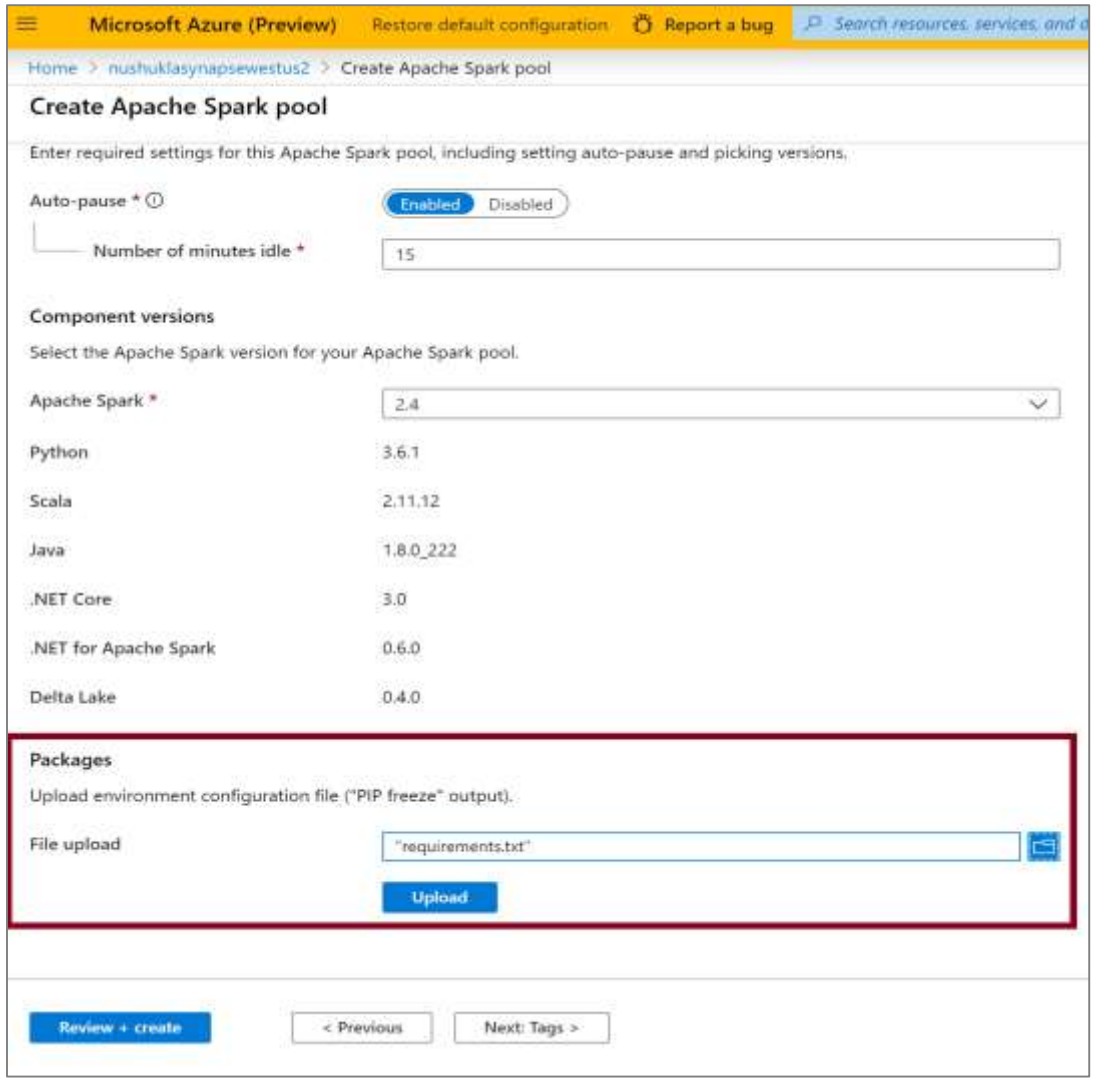
Constraints

The library version must exist on PyPI repository

Version downgrade of an existing library not allowed

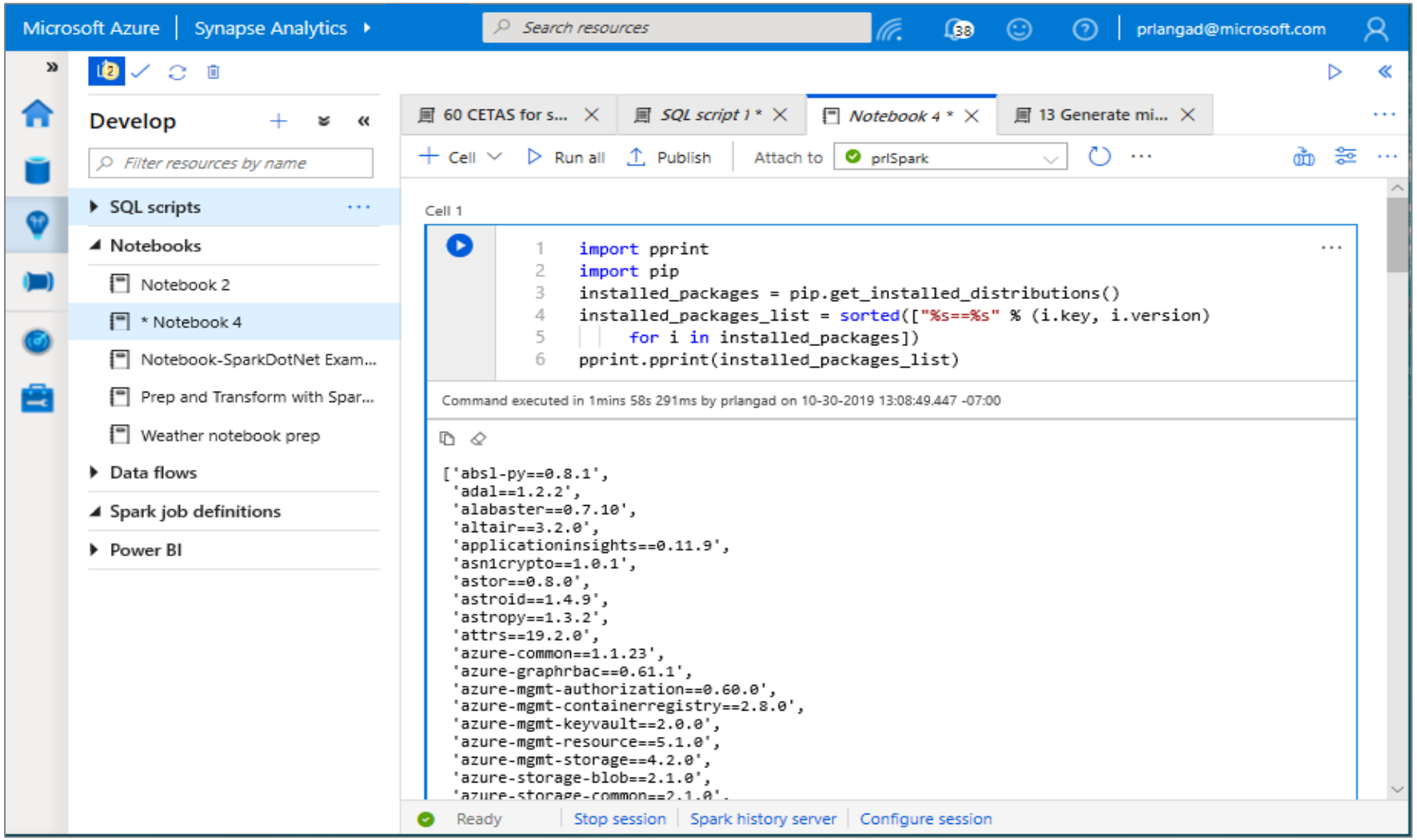
In the Portal

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



Library Management - Python

Get list of installed libraries with version information



Spark ML Algorithms

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

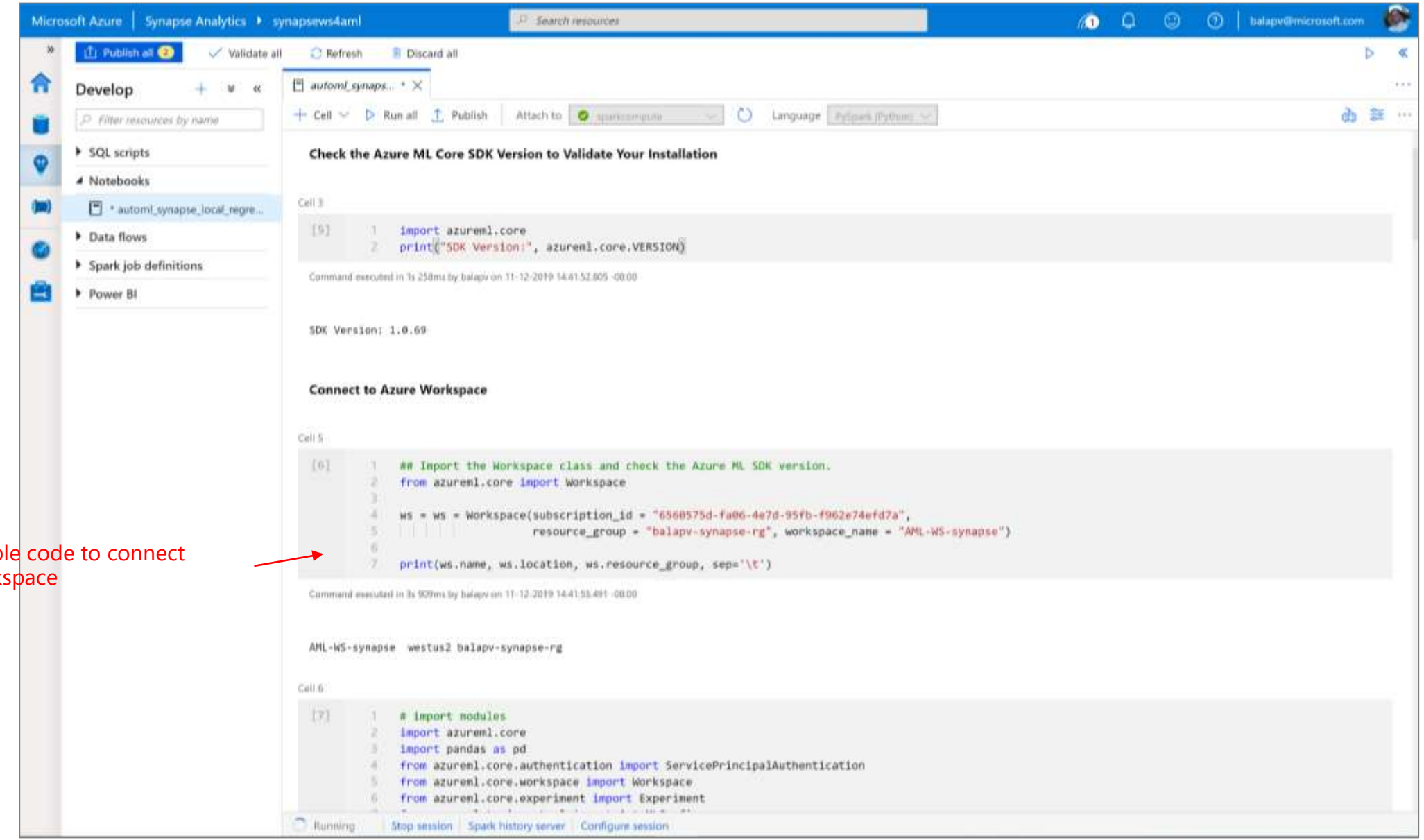
Microservice Orchestration

Multilingual Binding Generation

www.aka.ms/spark

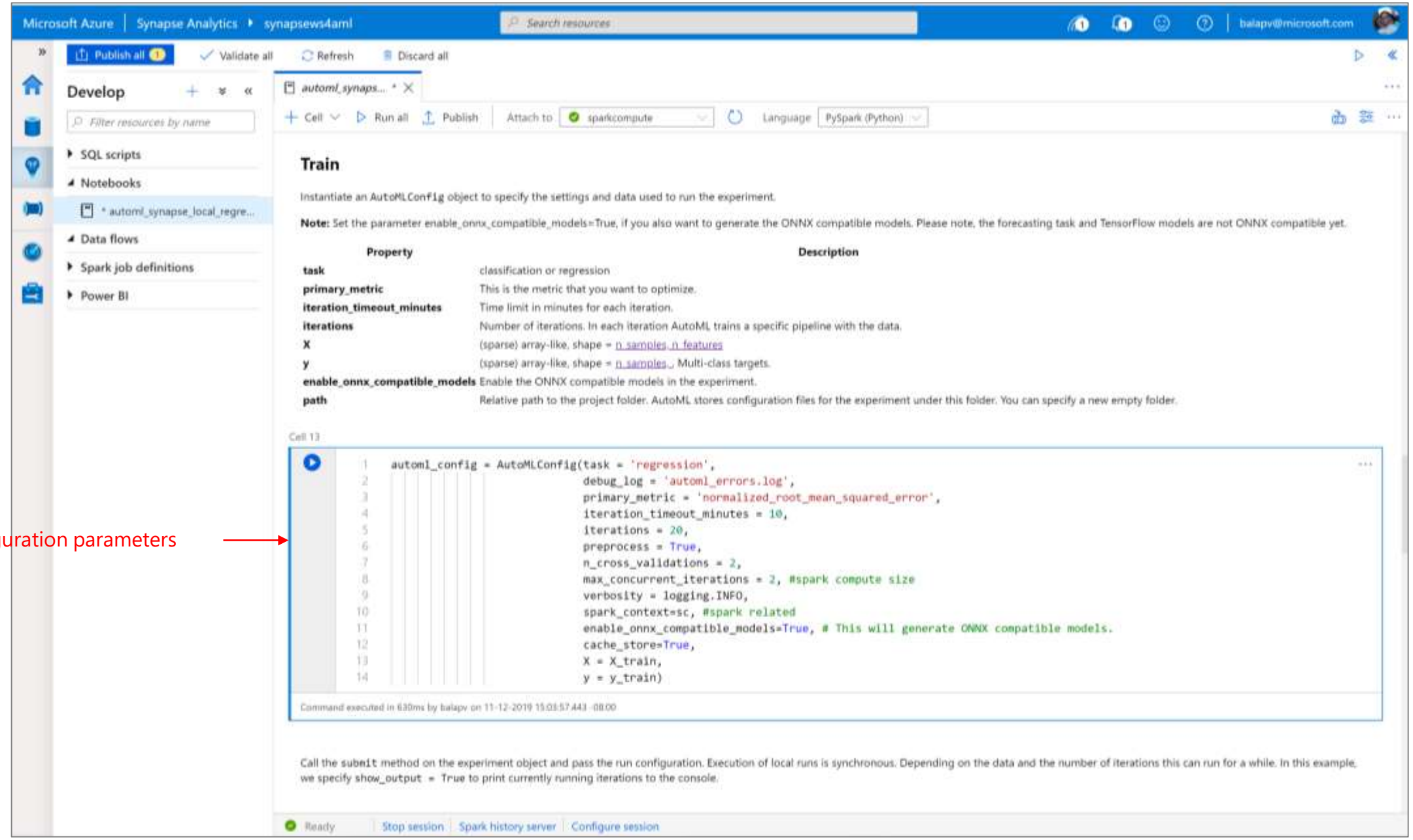
Azure/mmlspark

Synapse Notebook: Connect to AML workspace



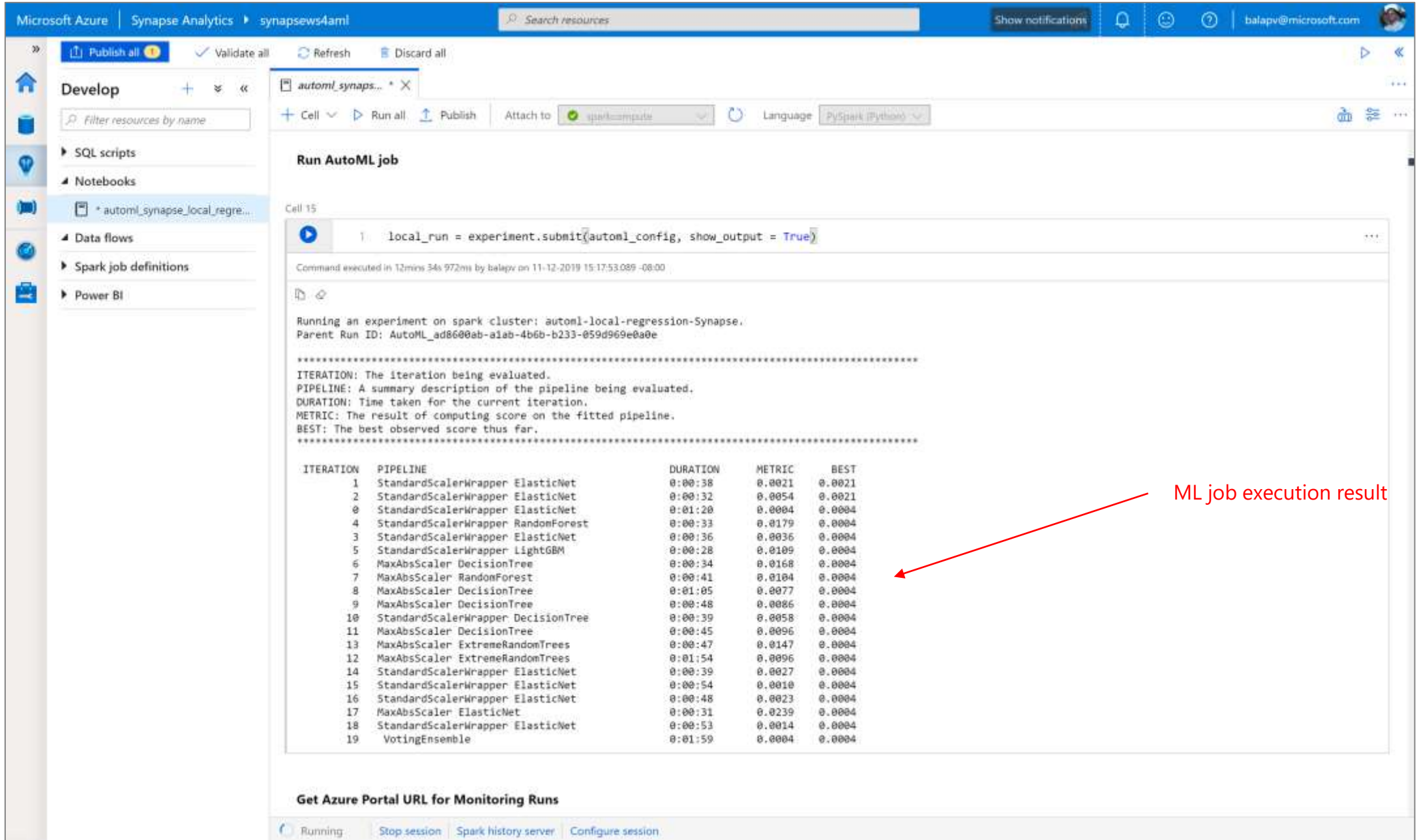
Simple code to connect workspace

Synapse Notebook: Configure AML job to run on Synapse



Configuration parameters

Synapse Notebook: Run AML job



Industry-leading security and compliance

Enterprise-grade security

Industry-leading compliance

ISO 27001

SOC 1 Type 2

SOC 2 Type 2

PCI DSS Level 1

Cloud Controls Matrix

ISO 27018

Content Delivery and Security Association

Shared Assessments

FedRAMP JAB P-ATO

HIPAA / HITECH

FIPS 140-2

21 CFR Part 11

FERPA

DISA Level 2

CJIS

IRS 1075

ITAR-ready

Section 508 VPAT

European Union Model Clauses

EU Safe Harbor

United Kingdom G-Cloud

China Multi Layer Protection Scheme

China GB 18030

China CCCPPF

Singapore MTCS Level 3

Australian Signals Directorate

New Zealand GCIO

Japan Financial Services

ENISA IAF

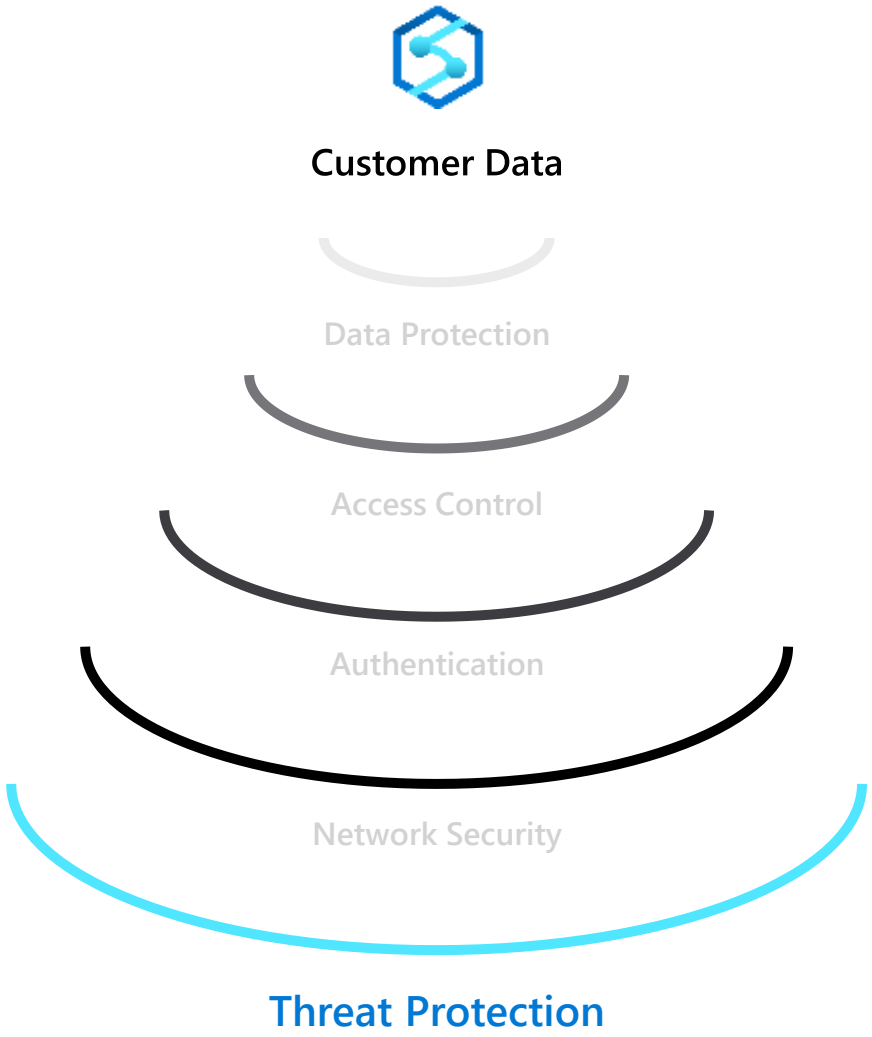
Threat Protection - Business requirements

How do we enumerate and track potential SQL vulnerabilities?

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

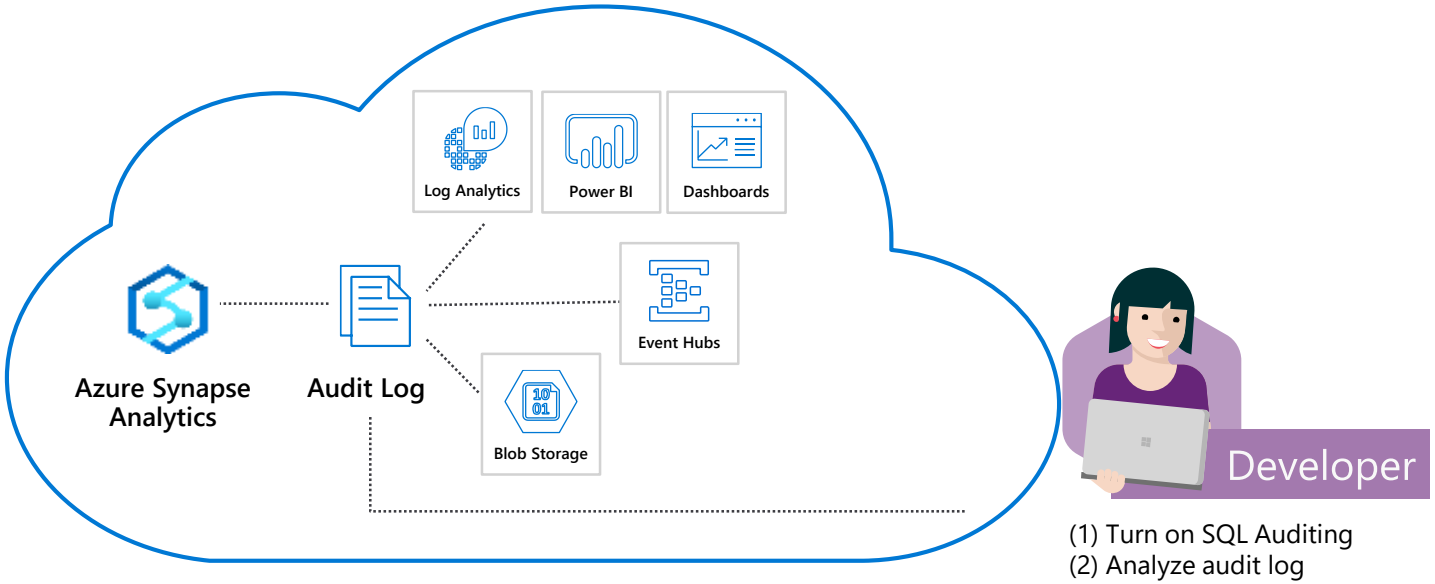
How do we discover and alert on suspicious database activity?

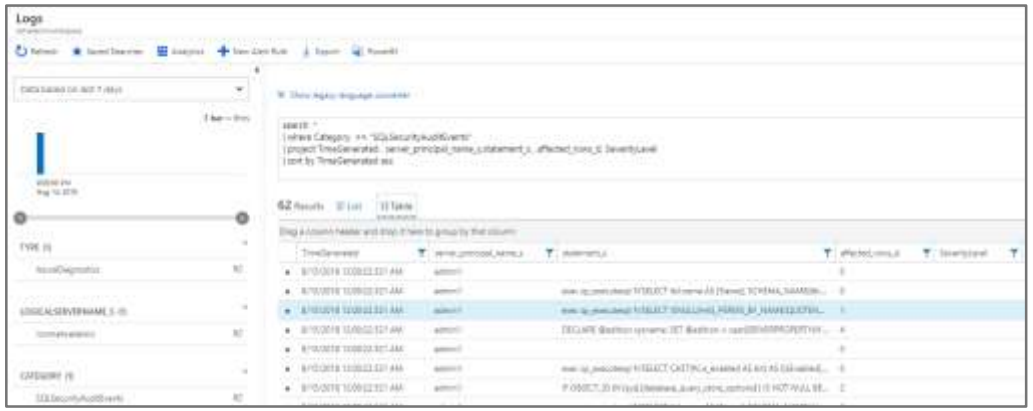
To detect and resolve any data exfiltration or SQL injection attacks.



SQL auditing in Azure Log Analytics and Event Hubs

Gain insight into database audit log

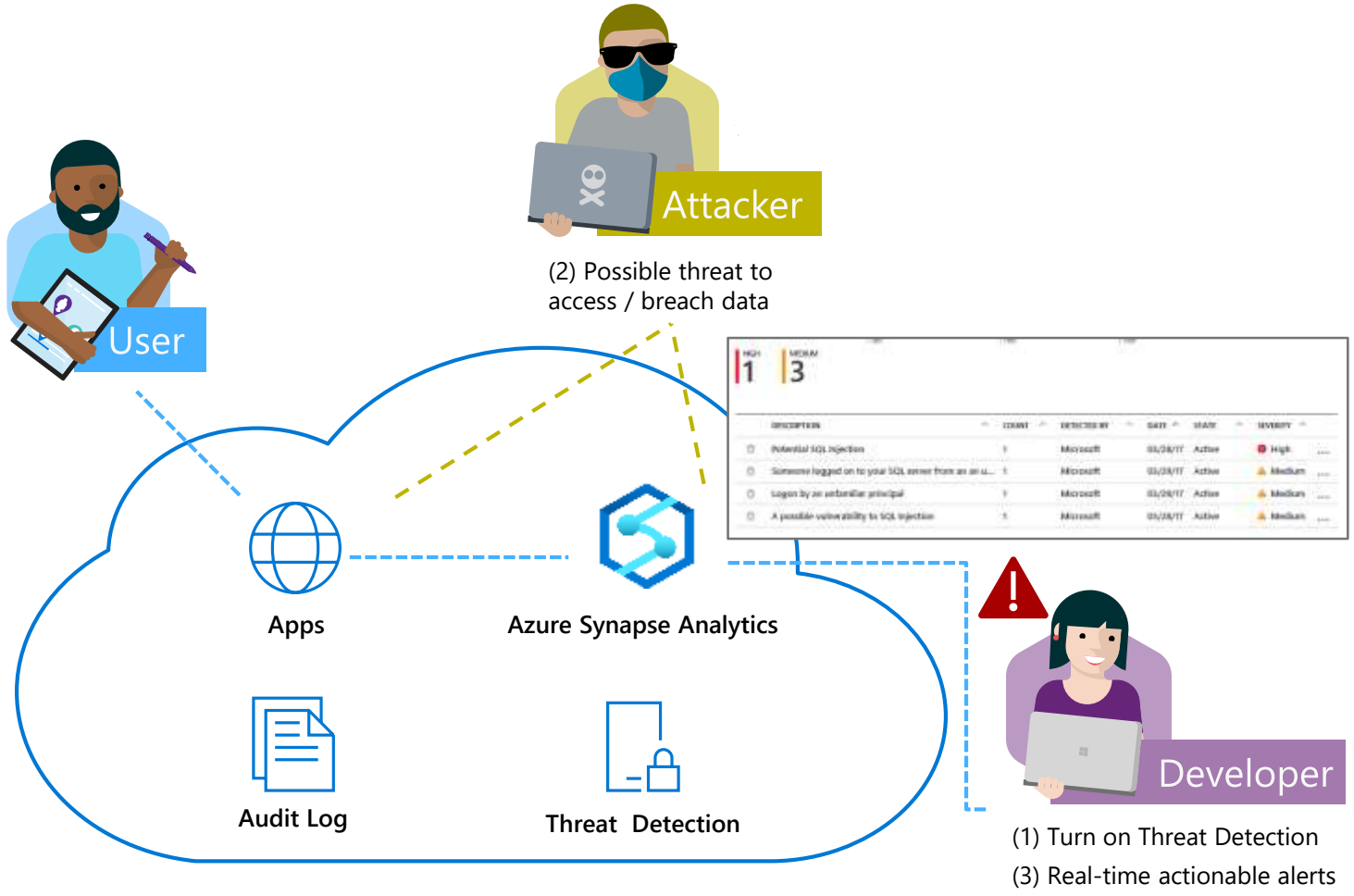




- ✓ **Configurable via audit policy**
- ✓ **SQL audit logs can reside in**
  - Azure Storage account
  - Azure Log Analytics
  - Azure Event Hubs
- ✓ **Rich set of tools for**
  - Investigating security alerts
  - Tracking access to sensitive data

SQL threat detection

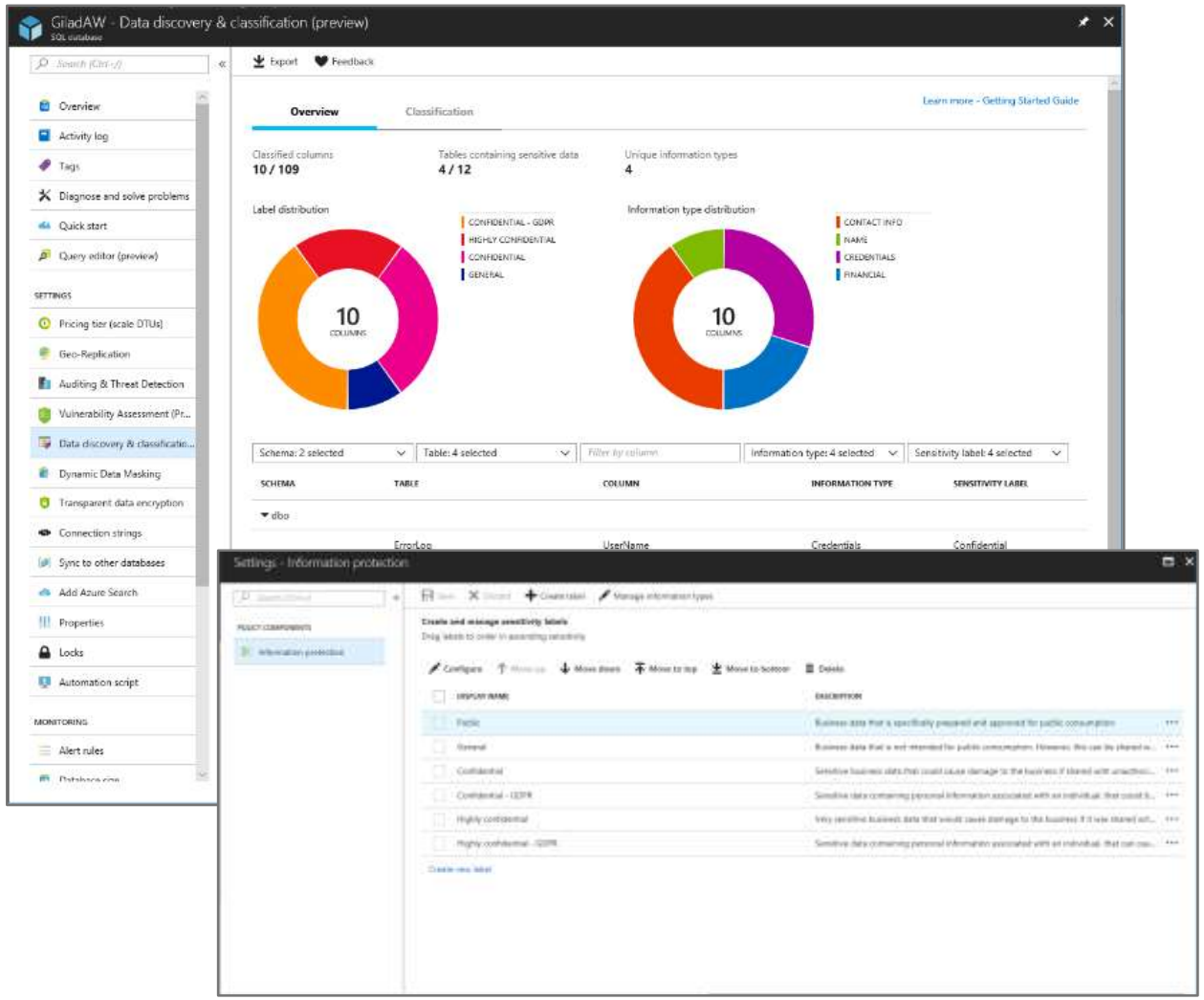
Detect and investigate anomalous database activity



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

SQL Data Discovery & Classification

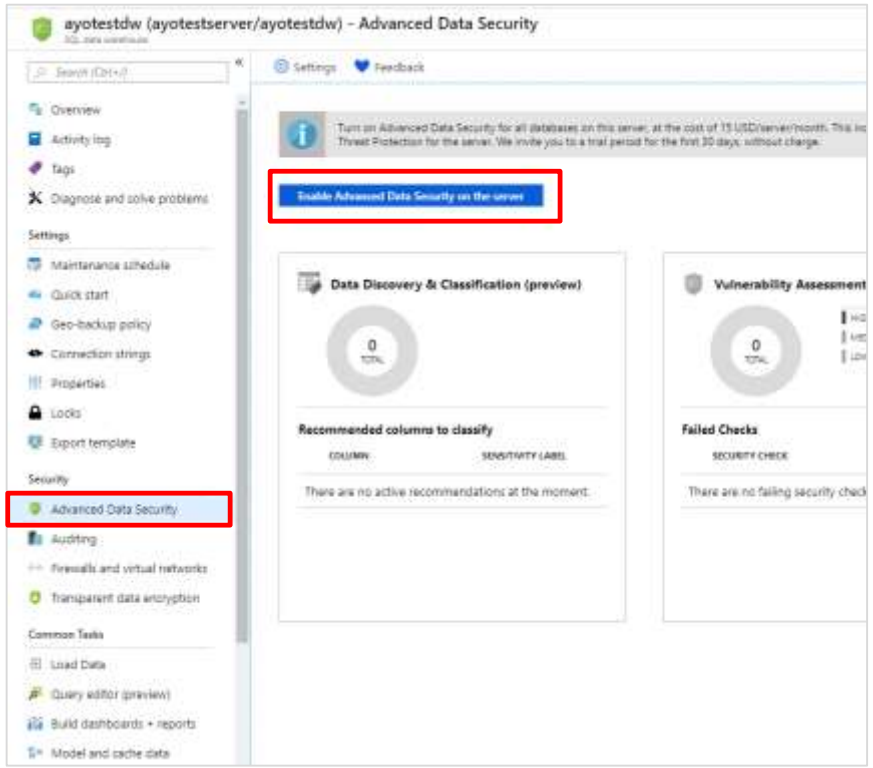
Discover, classify, protect and track access to sensitive data



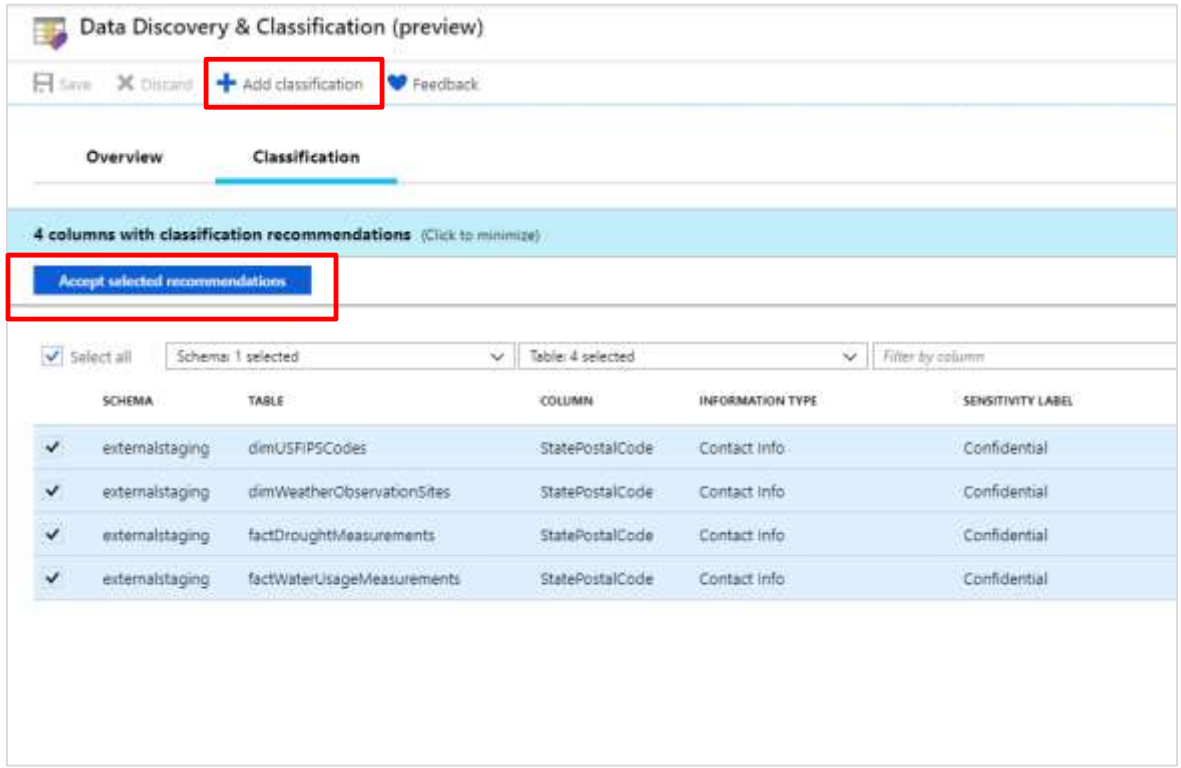
- ✓ Automatic discovery of columns with sensitive data
- ✓ Add persistent sensitive data labels
- ✓ Audit and detect access to the sensitive data
- ✓ Manage labels for your entire Azure tenant using Azure Security Center

SQL Data Discovery & Classification - setup

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

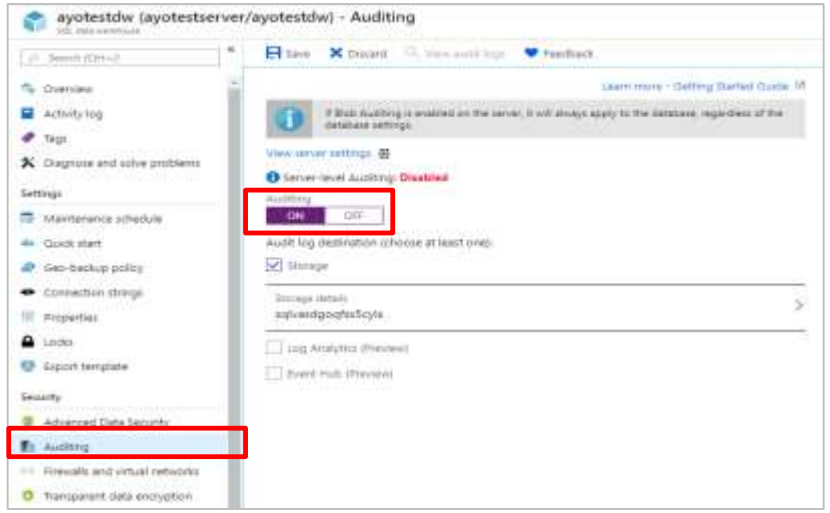


Step 2: Use recommendations and/or manual classification to classify all the sensitive columns in your tables

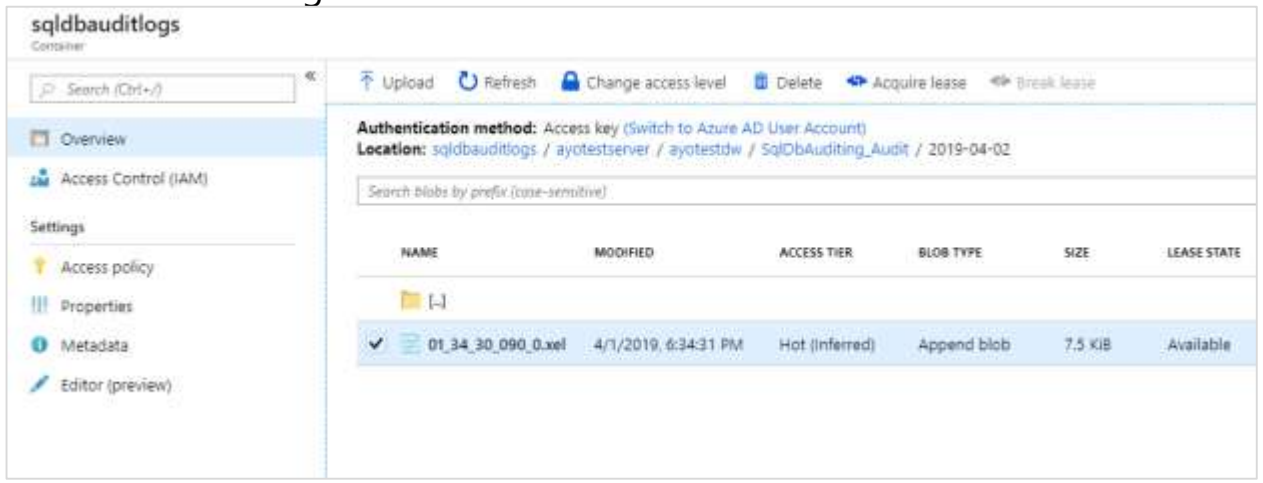


SQL Data Discovery & Classification – audit sensitive data access

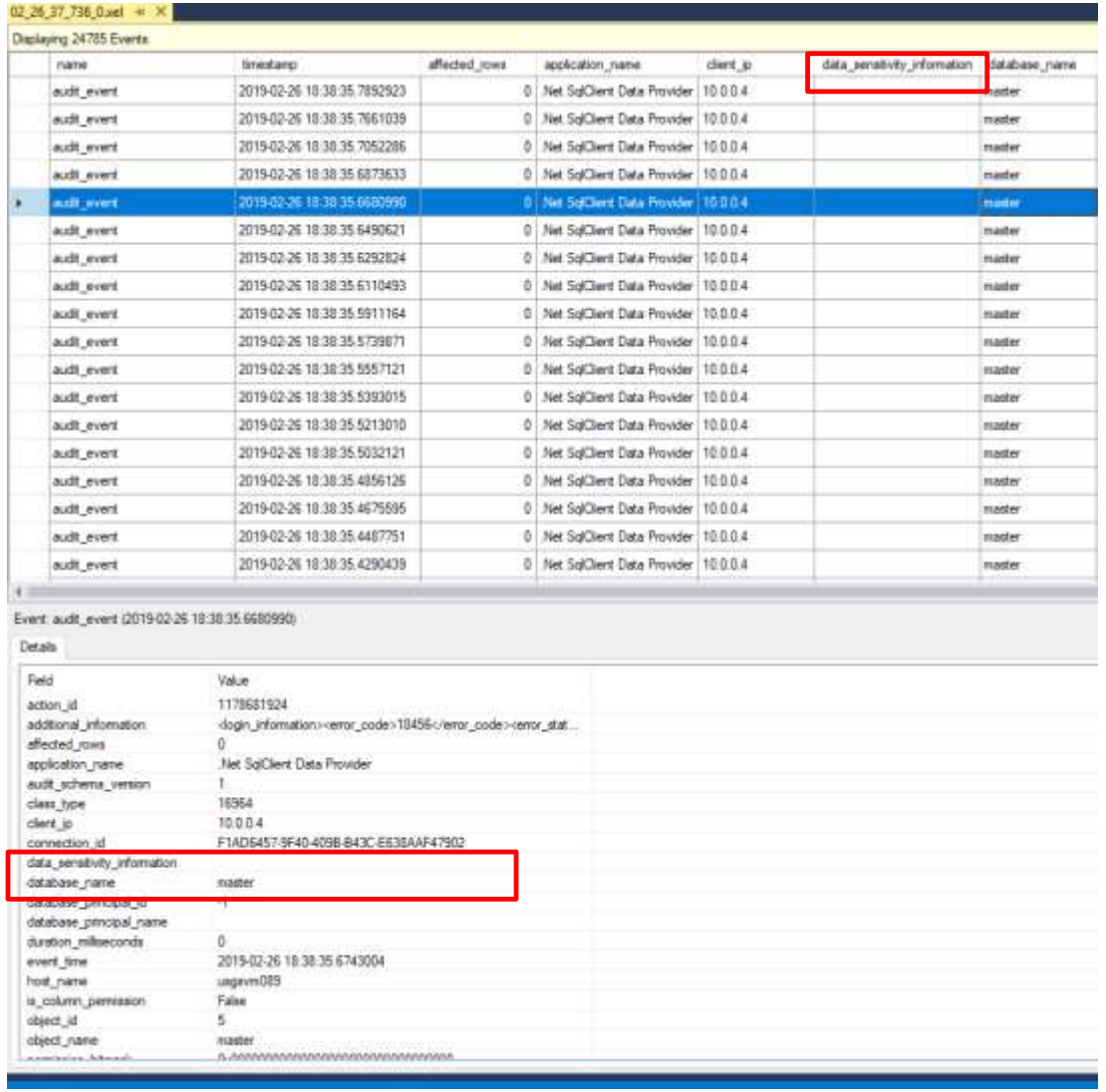
Step 1: Configure auditing for your target Data warehouse. This can be configured for just a single data warehouse or all databases on a server.



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



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



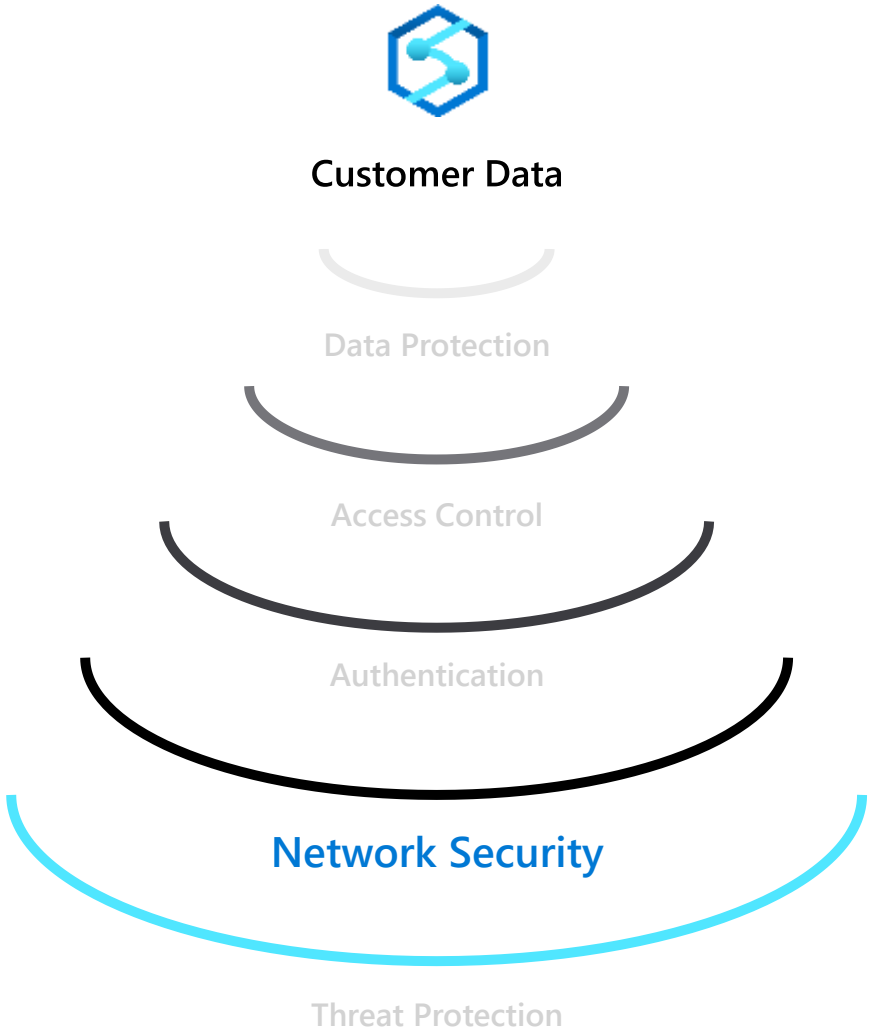
Network Security - Business requirements

How do we implement network isolation?

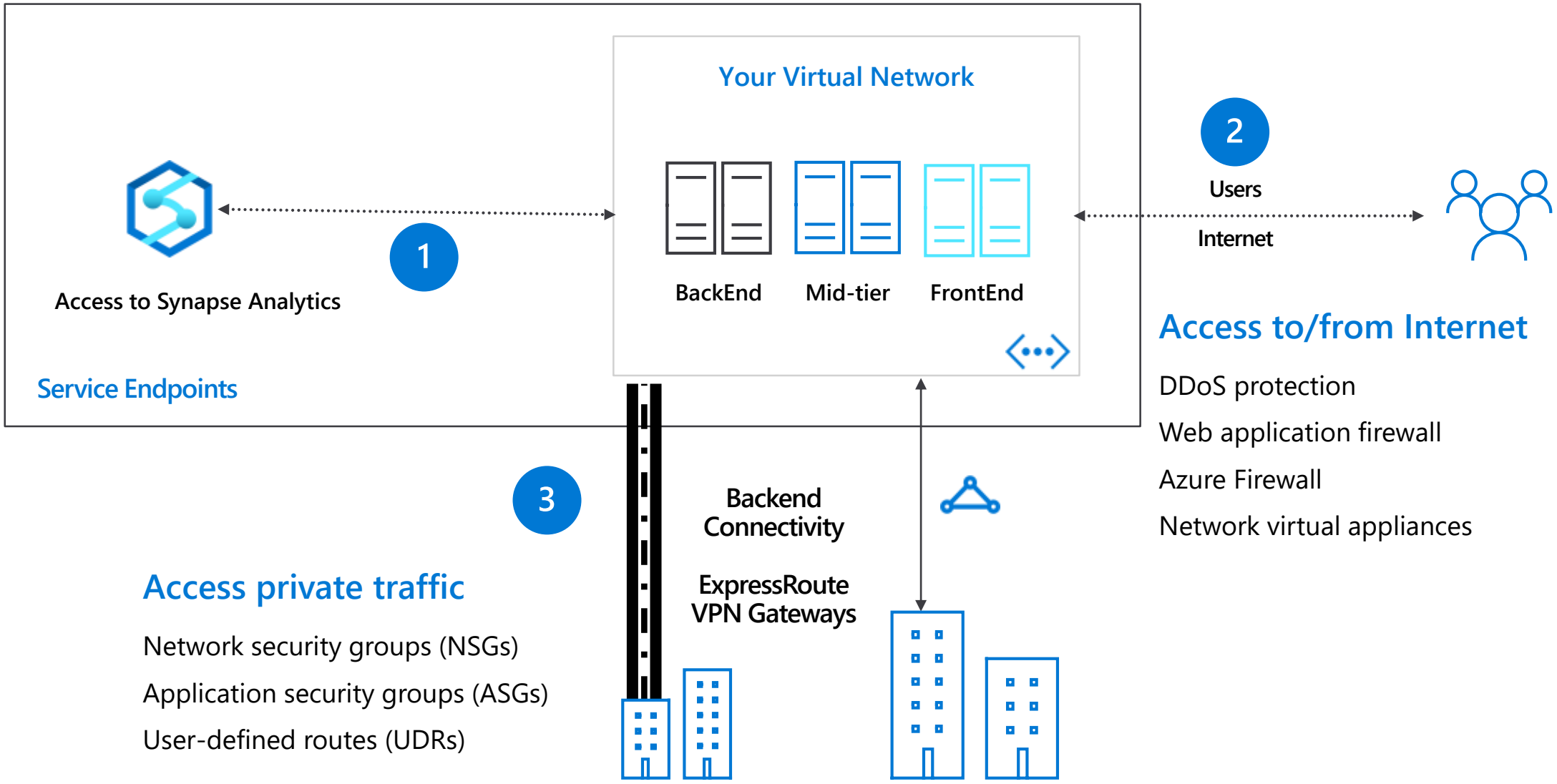
Data at different levels of security needs to be accessed from different locations.

How do we achieve separation?

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



Azure networking: application-access patterns



Access to/from Internet

DDoS protection

Web application firewall

Azure Firewall

Network virtual appliances

Access private traffic

Network security groups (NSGs)

Application security groups (ASGs)

User-defined routes (UDRs)

Securing with firewalls

Overview

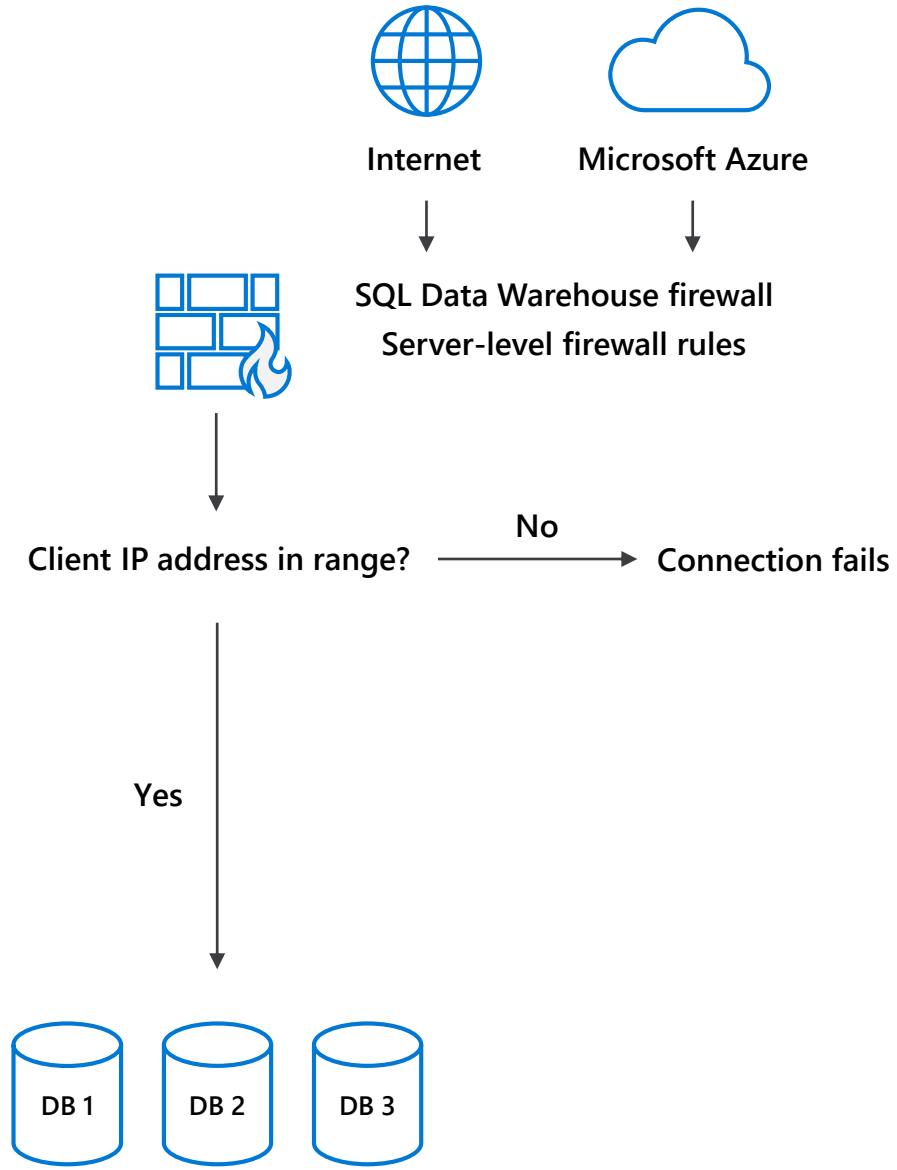
By default, all access to your Azure Synapse Analytics is blocked by the firewall.

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

Rules

Allow specific or range of whitelisted IP addresses.

Allow Azure applications to connect.

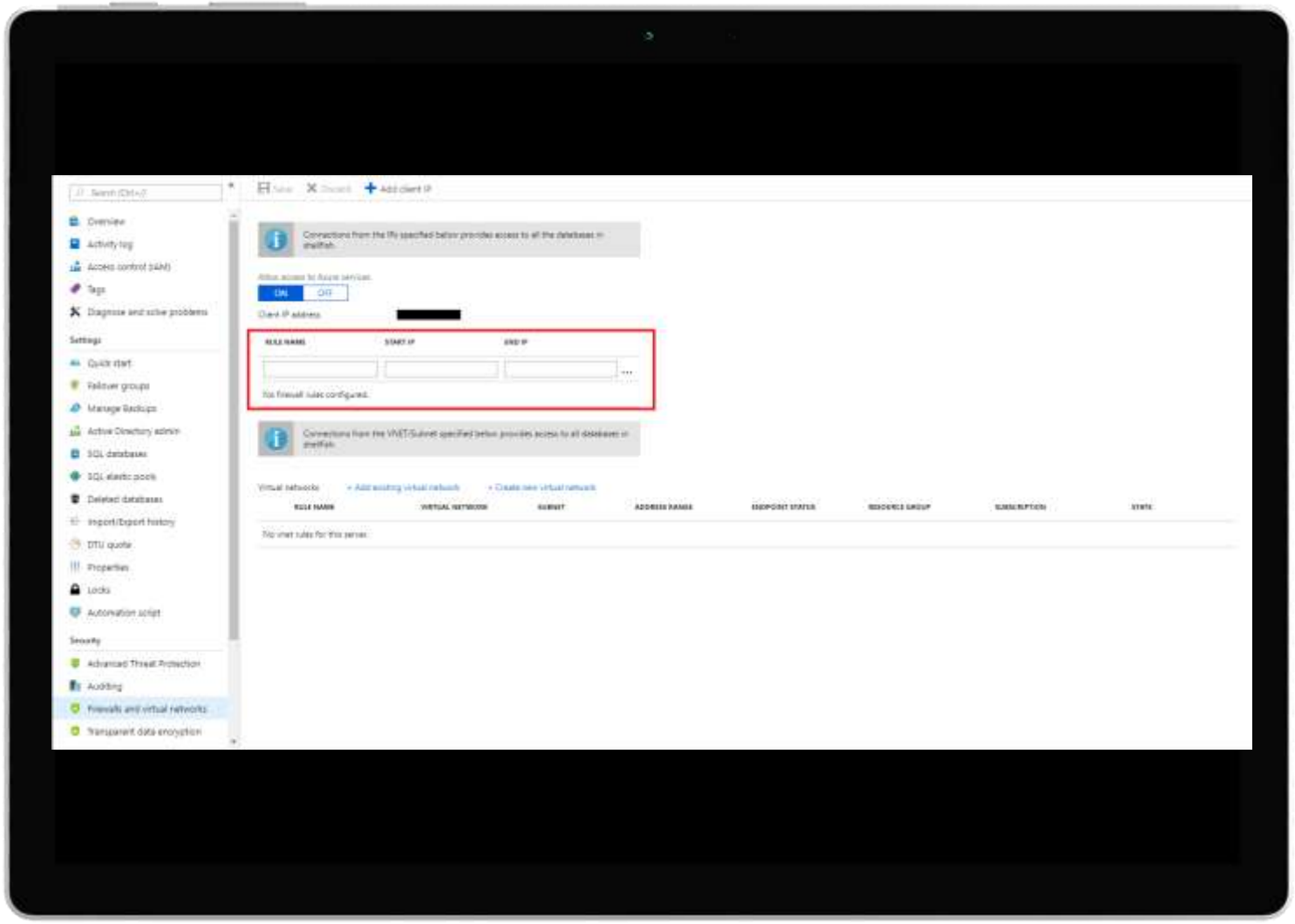


Firewall configuration on the portal

By default, Azure blocks all external connections to port 1433

Configure with the following steps:

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



Firewall configuration using REST API

Managing firewall rules through REST API must be authenticated.

For information, see Authenticating Service Management Requests.

Server-level rules can be created, updated, or deleted using REST API.

To create or update a server-level firewall rule, execute the PUT method.

To remove an existing server-level firewall rule, execute the DELETE method.

To list firewall rules, execute the GET.

```
PUT
https://management.azure.com/subscriptions/{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'
```

VNET configuration on Azure portal

Configure with the following steps:

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

REST API and PowerShell alternatives available

Note:

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

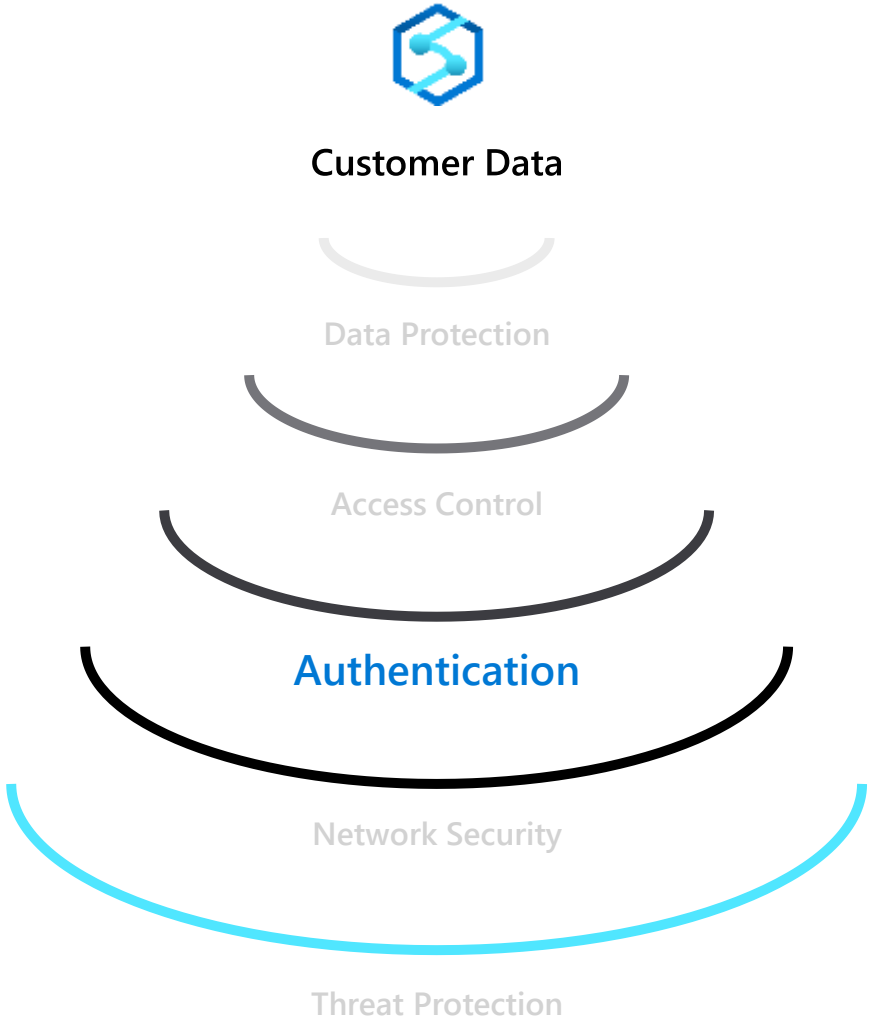
There must first be a virtual network service endpoint for the rule to reference.

Authentication - Business requirements

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

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

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



Azure Active Directory authentication

Overview

Manage user identities in one location.

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

Benefits

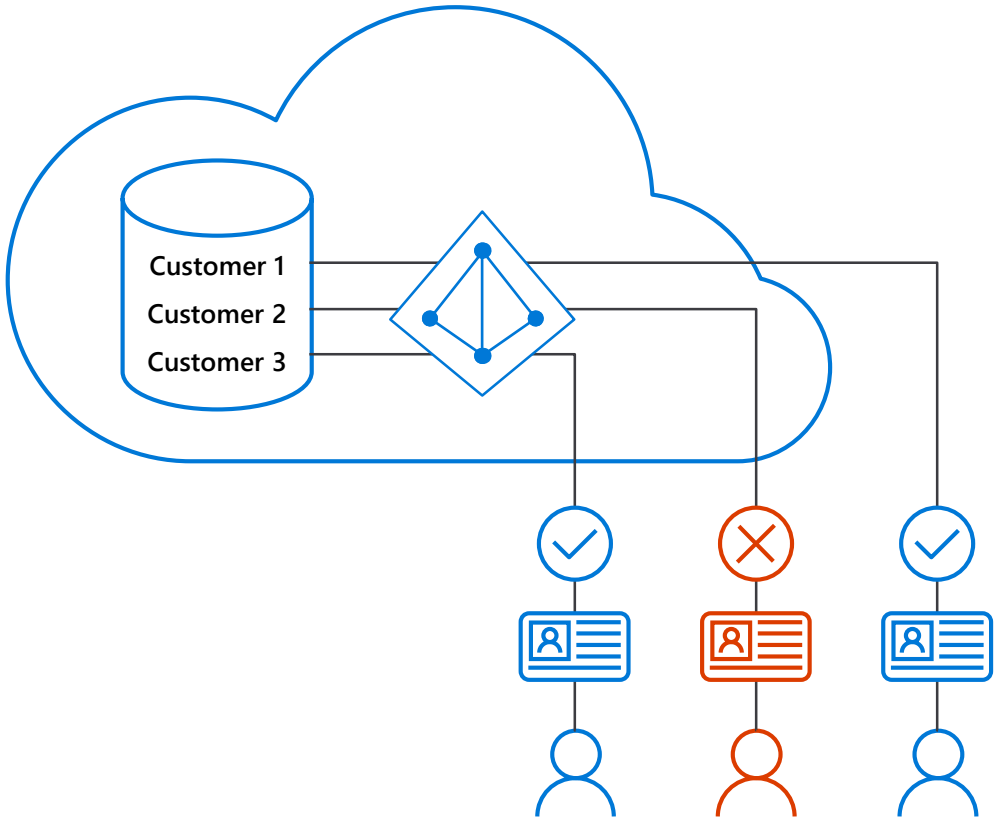
Alternative to SQL Server authentication

Limits proliferation of user identities across databases

Allows password rotation in a single place

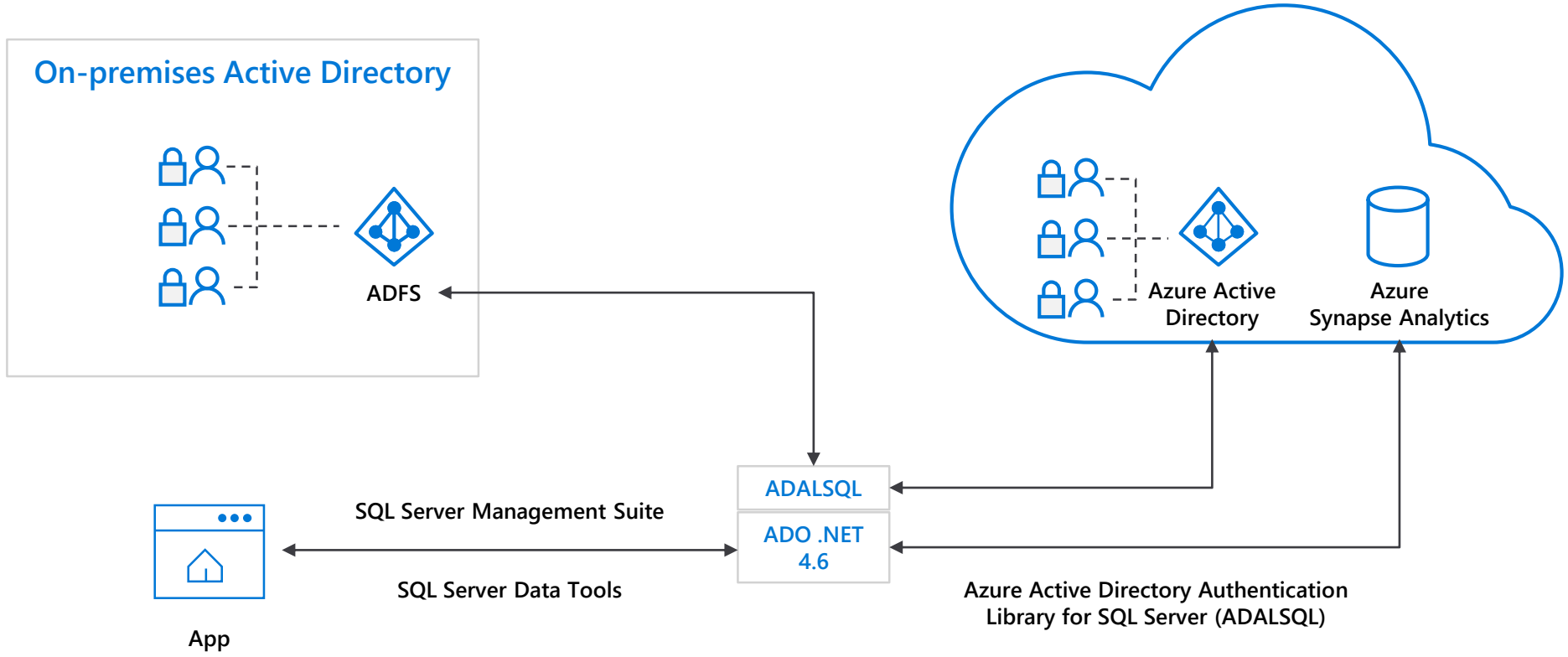
Enables management of database permissions by using external Azure Active Directory groups

Eliminates the need to store passwords



Azure Active Directory trust architecture

Azure Active Directory and Azure Synapse Analytics



SQL authentication

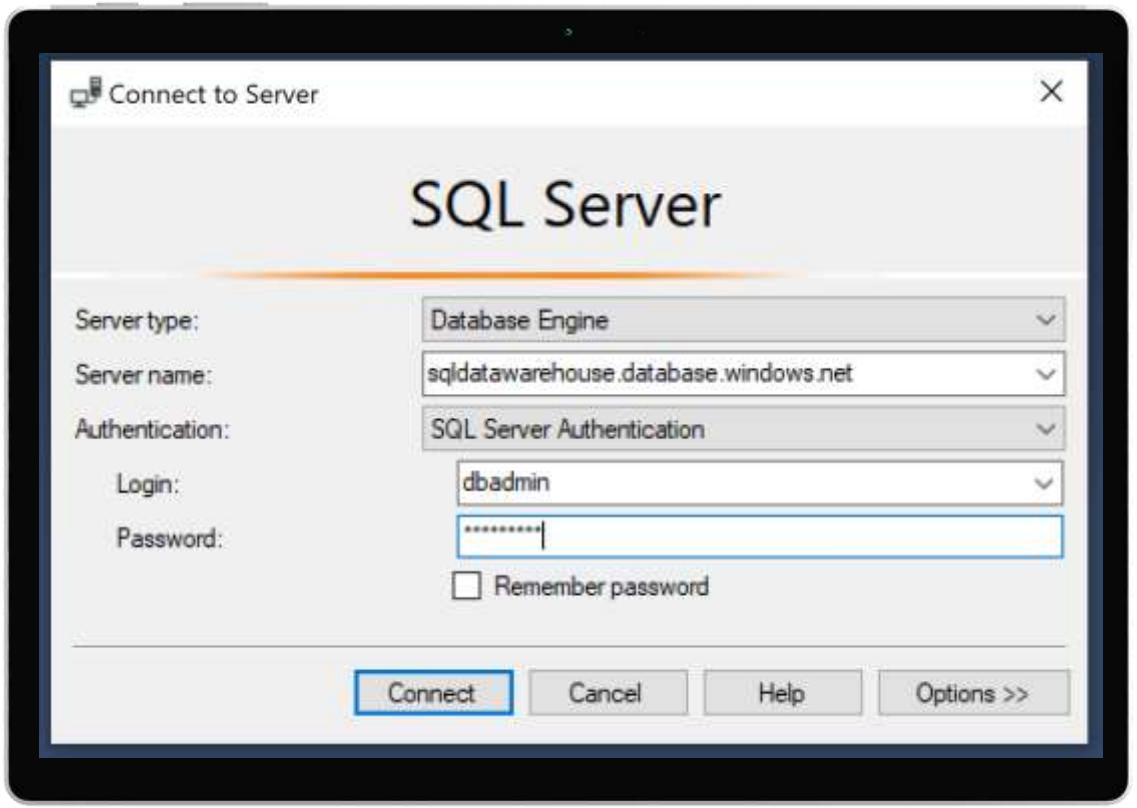
Overview

This authentication method uses a username and password.

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

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

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



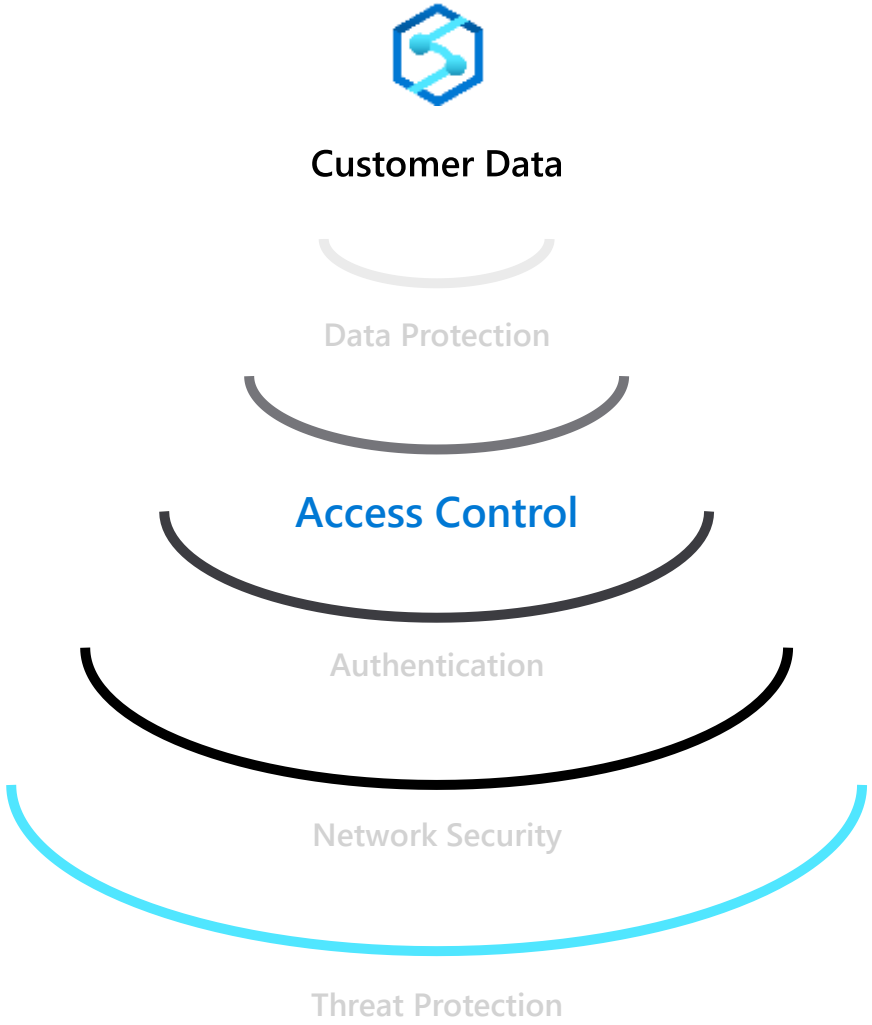

```
-- Connect to master database and create a login
CREATE LOGIN ApplicationLogin WITH PASSWORD = 'Str0ng_password';
CREATE USER ApplicationUser FOR LOGIN ApplicationLogin;

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

Access Control - Business requirements

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

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

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



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

Overview

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

Prevent unauthorized queries against certain tables.

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

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

Row-level security (RLS)

Overview

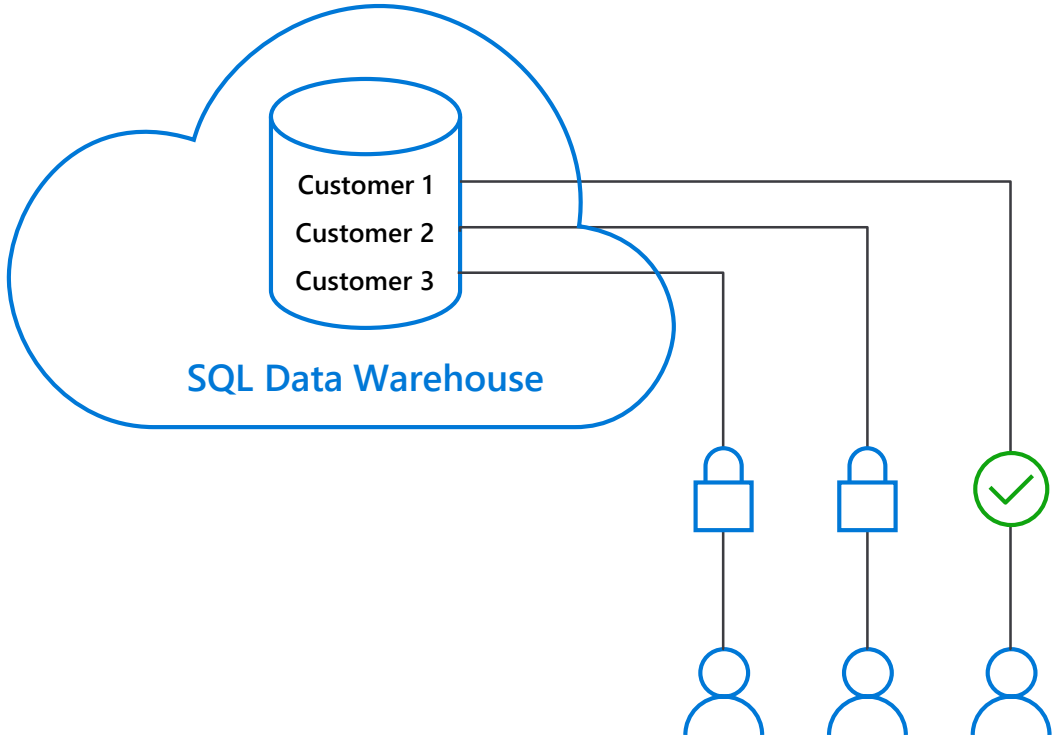
Fine grained access control of specific rows in a database table.

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

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

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

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



Row-level security

Creating policies

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

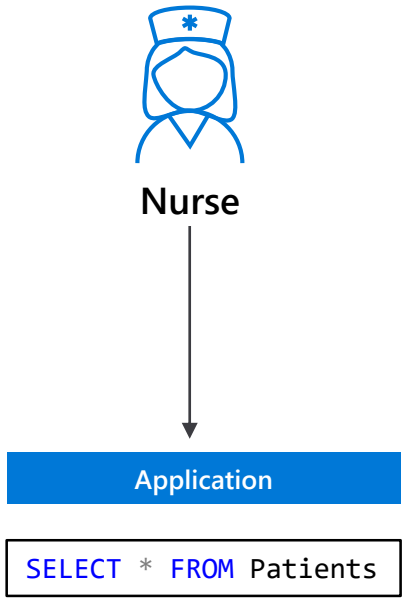
The following examples demonstrate the use of the CREATE SECURITY POLICY syntax

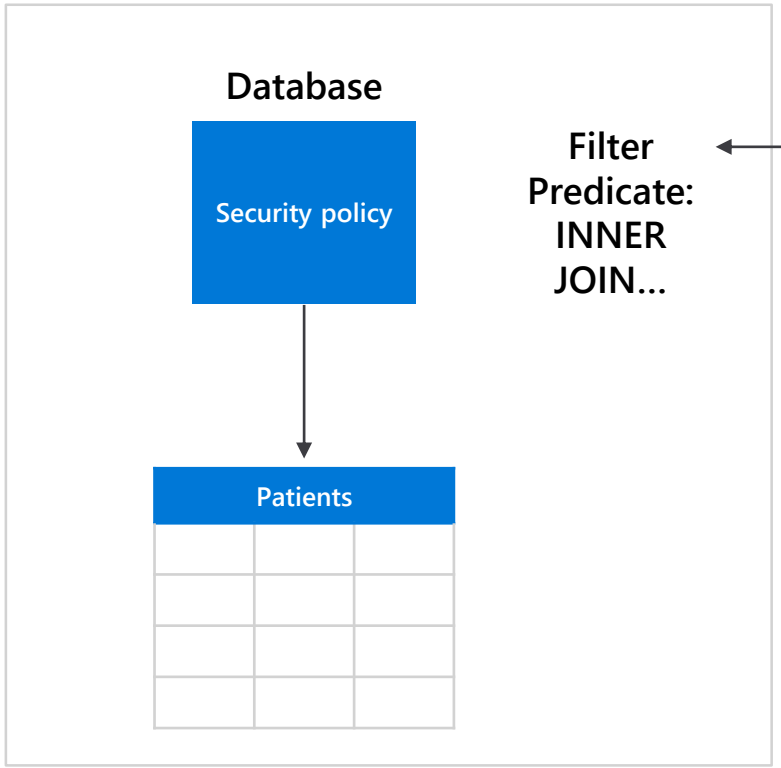
```
-- The following syntax creates a security policy with a filter predicate for the Customer table
CREATE SECURITY POLICY [FederatedSecurityPolicy]
ADD FILTER PREDICATE [rls].[fn_securitypredicate]([CustomerId])
ON [dbo].[Customer];


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

Row-level security

Three steps:

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





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:

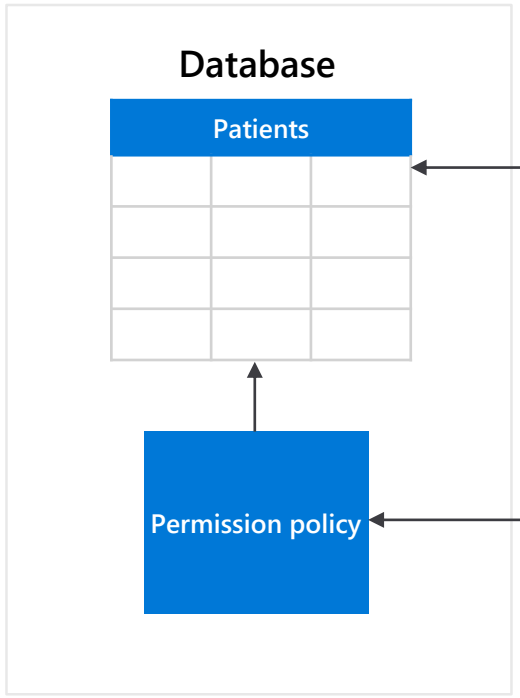
1. Policy manager creates permission policy in T-SQL, binding the policy to the Patients table on a specific group.
2. App user (for example, a nurse) selects from Patients table.
3. Permission policy prevents access on sensitive data.

Nurse

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



Policy manager

```
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

Data Protection - Business requirements

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

What key management options do I have?

Dynamic Data Masking

Overview

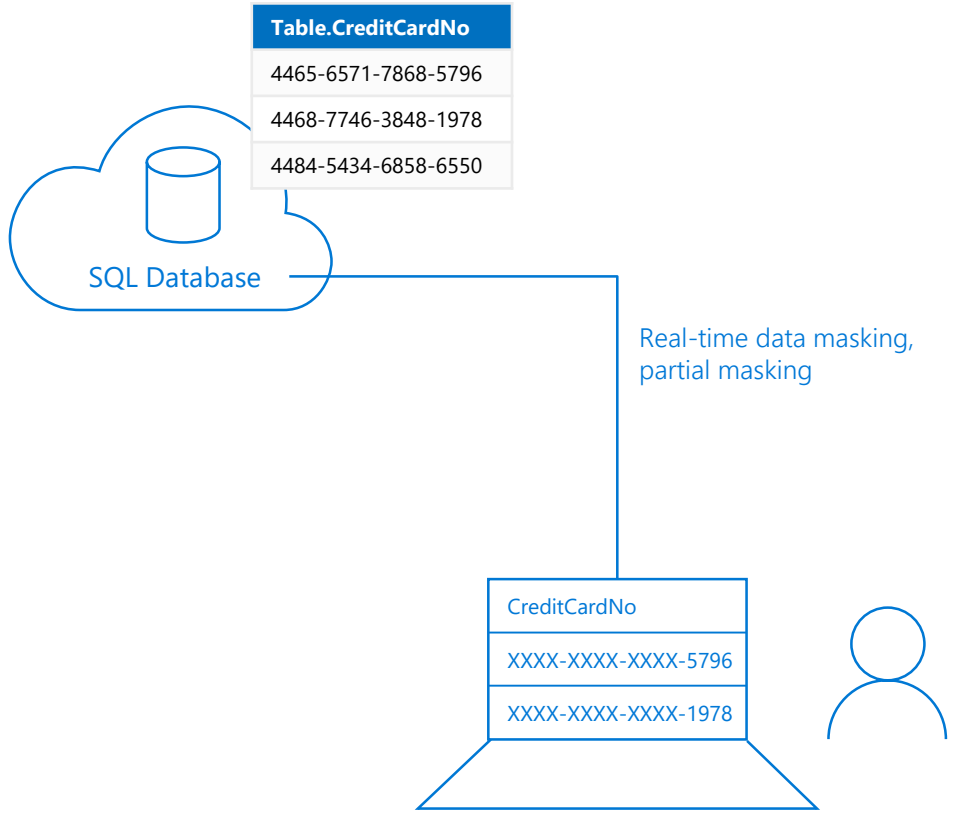
Prevent abuse of sensitive data by hiding it from users

Easy configuration in new Azure Portal

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

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

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



Dynamic Data Masking

Three steps

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

2. The app-user selects from the Employee table

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

Security officer

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)

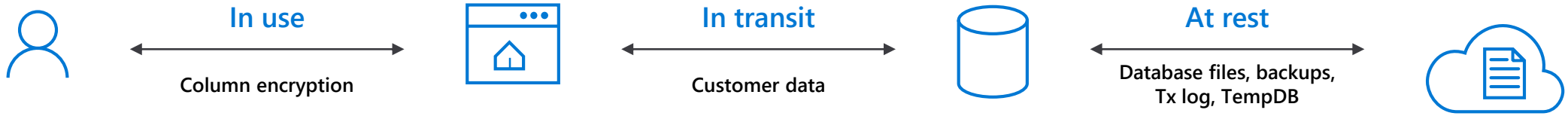
	First Name	Social Security Num...	Email	Salary
1	LILA	758-10-9637	lila.barnett@comcast.net	1012794
2	JAMIE	113-29-4314	jamie.brown@ntlworld.com	1025713
3	SHELLEY	550-72-2028	shelley.lynn@charter.net	1040131
4	MARCELLA	903-94-5665	marcella.estrada@comcast.net	1040753
5	GILBERT	376-79-4787	gilbert.juarez@verizon.net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

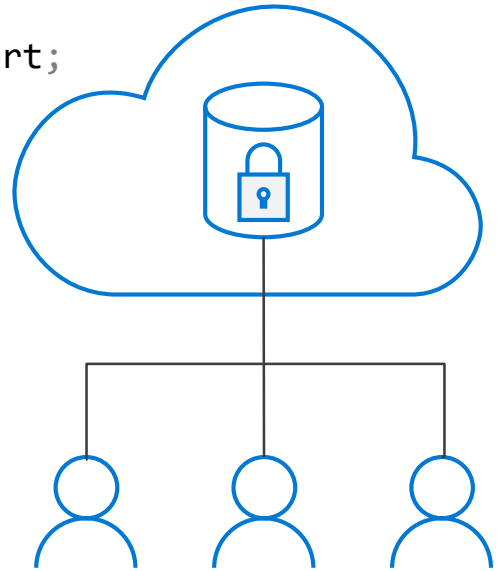
Service OR User managed keys.

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

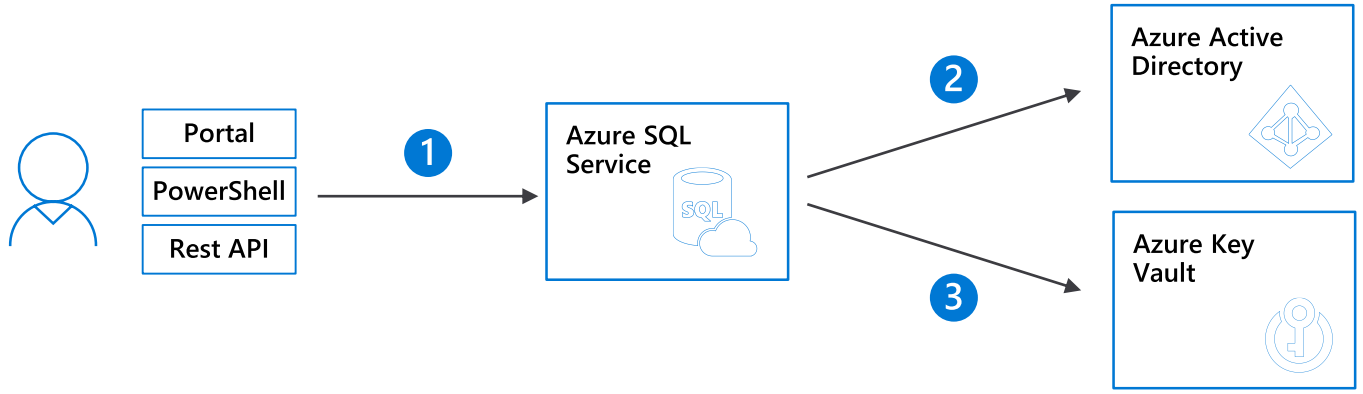
Benefits with User Managed Keys

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

Highly available and scalable cloud-based key store.

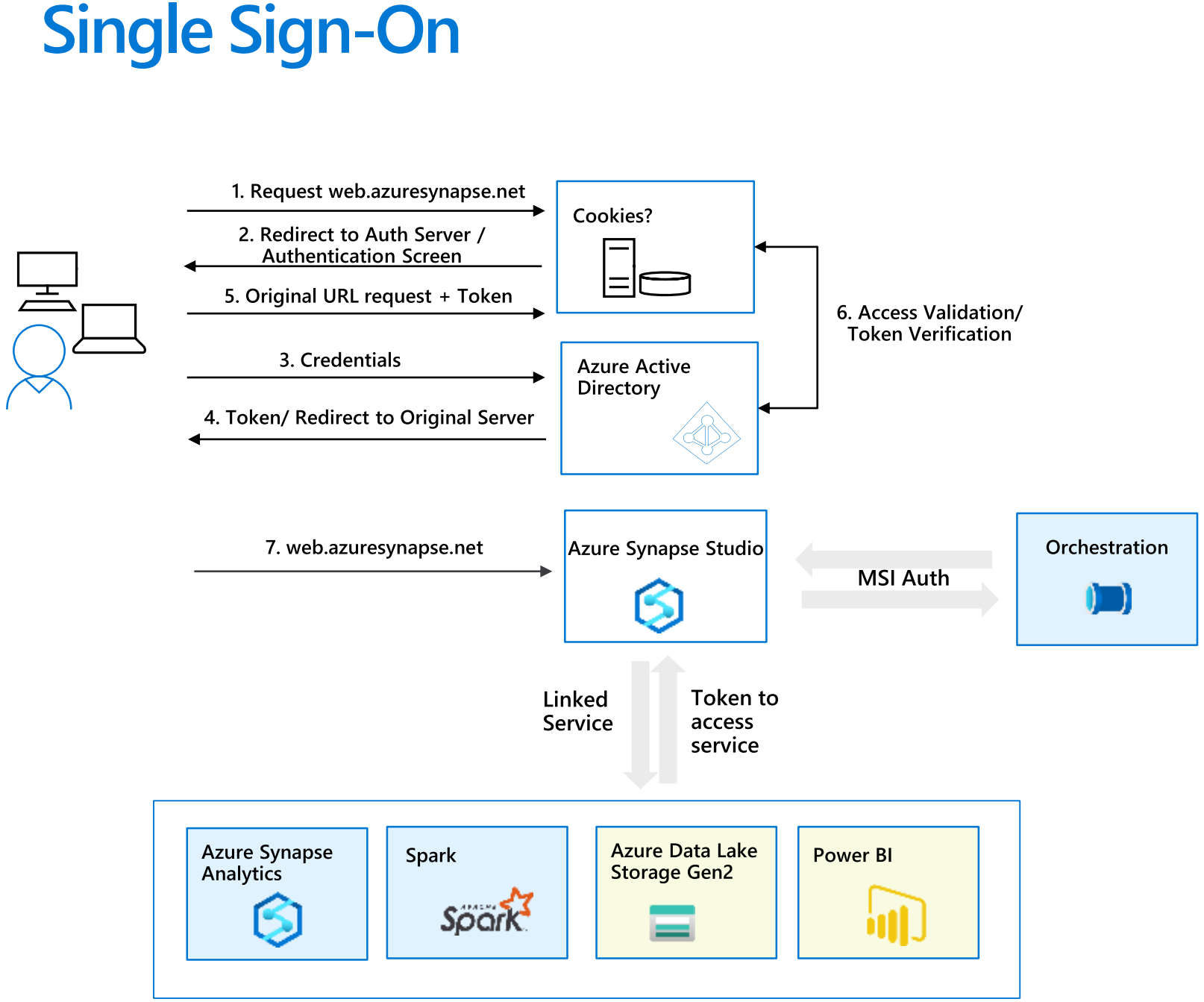
Central key management that allows separation of key management and data.

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.

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 MSI auth for automation

End