PASS SQLSATURDAY

Modernizing your Data Warehouse with Azure Synapse Analytics

Andrea Benedetti, Microsoft

Own your career with interactive learning built by community and guided by data experts. Get involved. Get ahead.

Explore your PASS community

PASS MARATHON

Free online webinar events

PASS LOCAL GROUPS

Local user groups around the world

PASS SUMMIT

Connect with the global data community

PASS VIRTUAL GROUPS

Online special interest user groups

PASS.org

PASS CONNECTOR INSIGHTS

Learning on-demand and delivered to you

PASS VOLUNTEERS

Get involved

Missed PASS Summit 2019?

Get the Recordings

Download all PASS Summit sessions on Data Management, Analytics, or Architecture for only $399 USD

More options available at
PASSstuff.com

We are covering all bases to ensure our community can continue reaching new and exciting heights. Plans are underway for the in-person event you all know and love along with a new venture, a new opportunity: a PASS Summit 2020 Virtual Event.

Find out more at PASS.org/summit

PASS SQLSATURDAY

Thank you to our Global Sponsors and Supporters

Microsoft Azure

DELL Technologies

Google Cloud

IDERA

Quest

SentryOne

vmware

VIRTUAL PASS SQLSATURDAY PORDENONE | 30 MAY 2020

PASS SQLSATURDAY

Thank you to our Local Sponsors and Supporters

Microsoft

This event was sponsored by Microsoft

Learn more about SQL Server 2019 today:

-Get free training: aka.ms/sqlworkshops

-Download the SQL19 eBook: aka.ms/sql19_ebook

Andrea Benedetti

https://twitter.com/anBenedetti

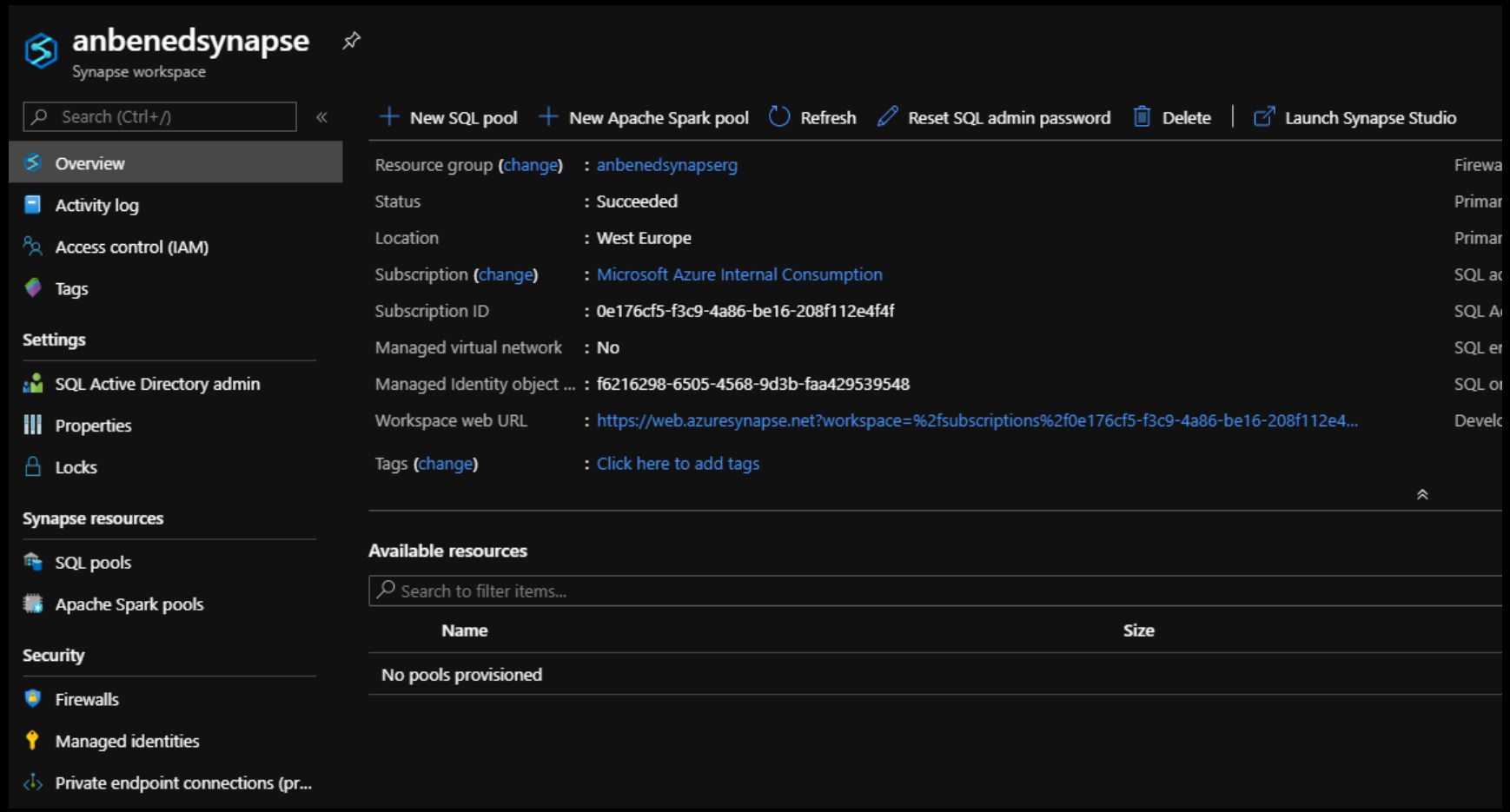
https://github.com/anbened

https://www.linkedin.com/in/abenedetti/

Andrea Benedetti


Azure Synapse Analytics

Public Preview is here ;-)



Analytics & AI is the #1 investment for business leaders, however they struggle to maximize ROI

80%

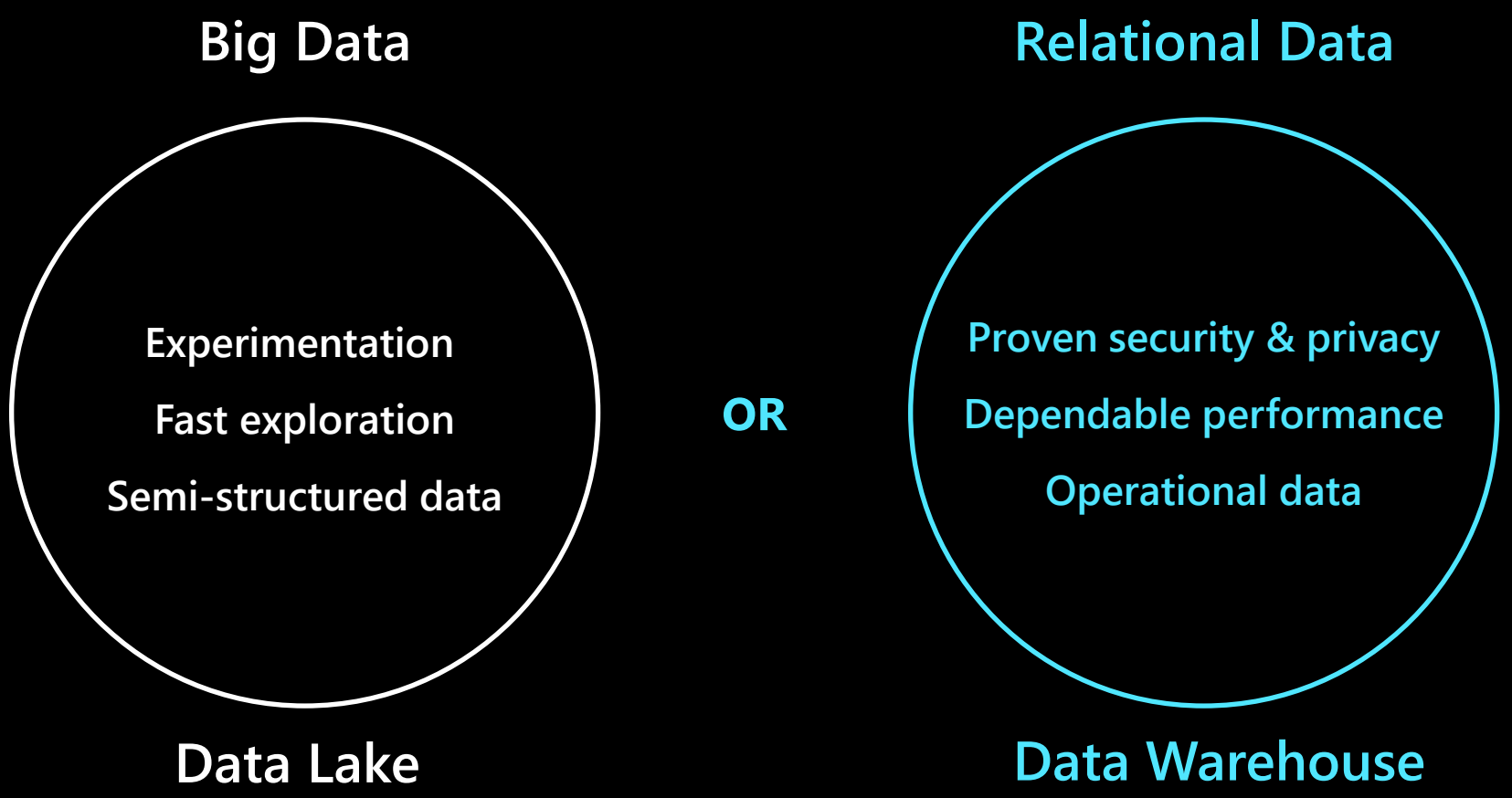
report struggling to become mature users of data*

55%

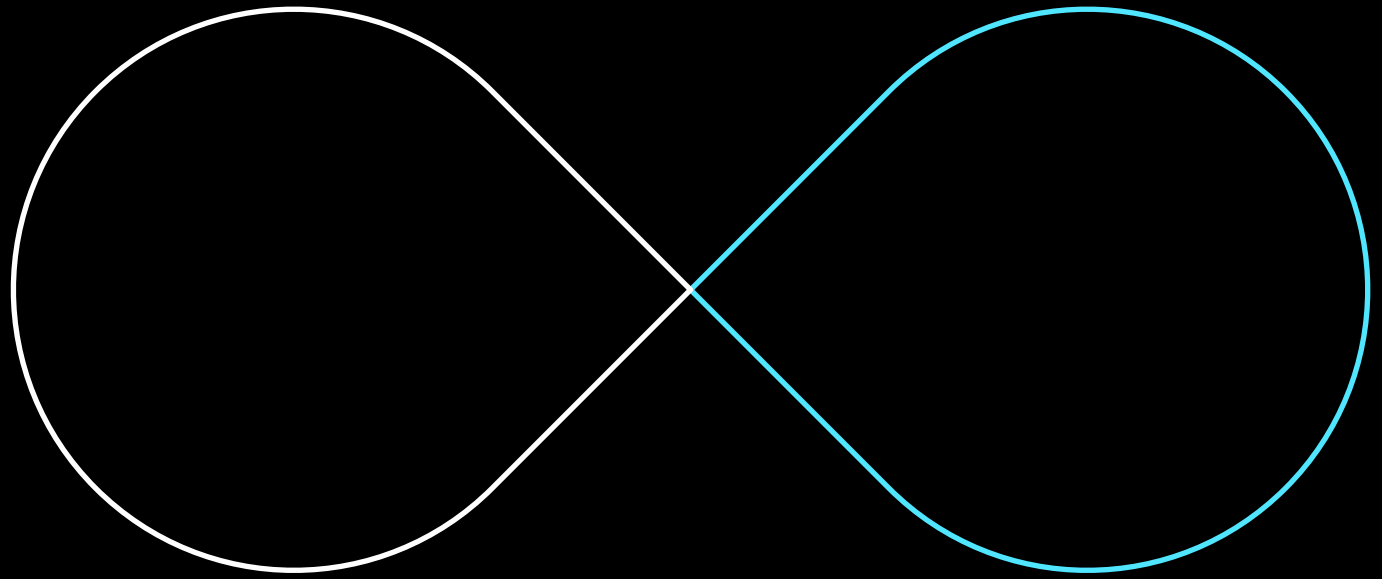
report data silos and data management difficulties as roadblocks*

* Harvard Business Review (2019), Understanding why analytics strategies fall short for some, but not for others

Businesses are forced to maintain two critical, yet independent analytics systems

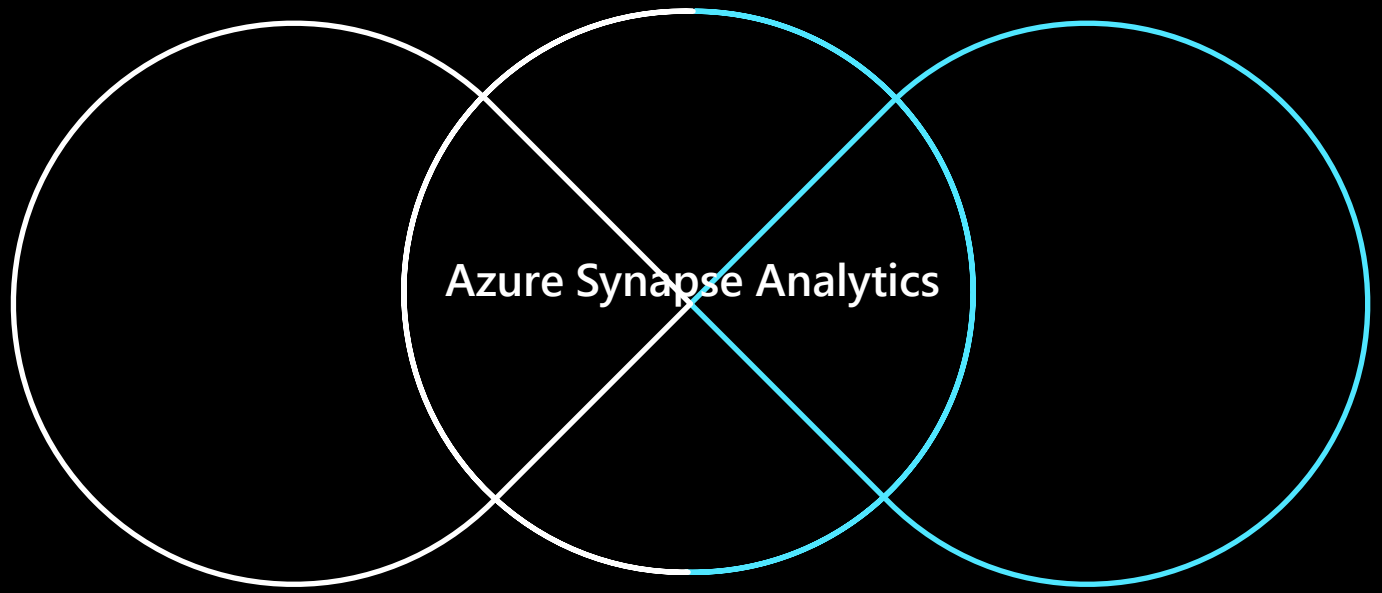


Azure brings these two worlds together, in a single service, to provide limitless analytics

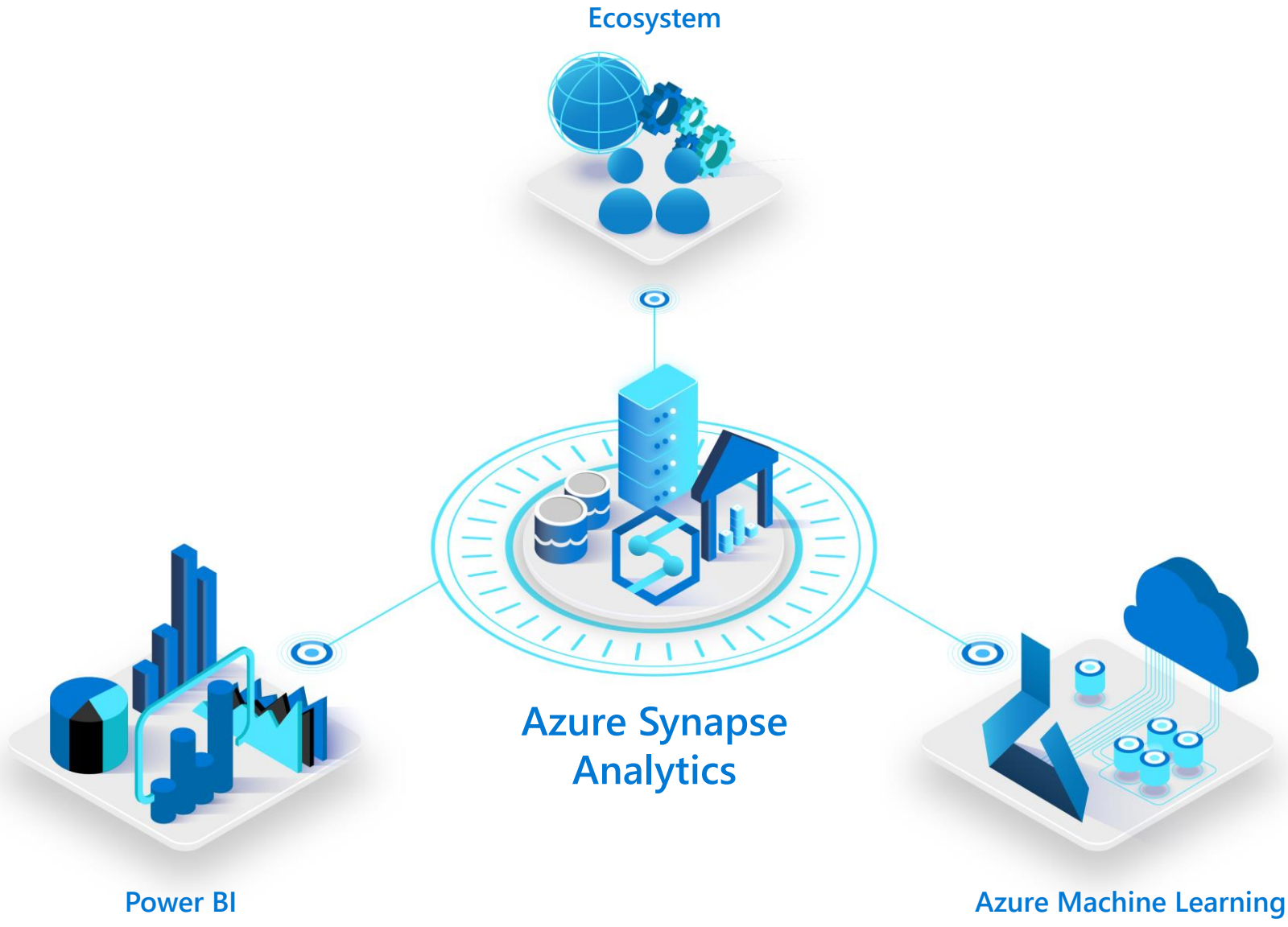


Data warehousing & big data analytics—all in one service

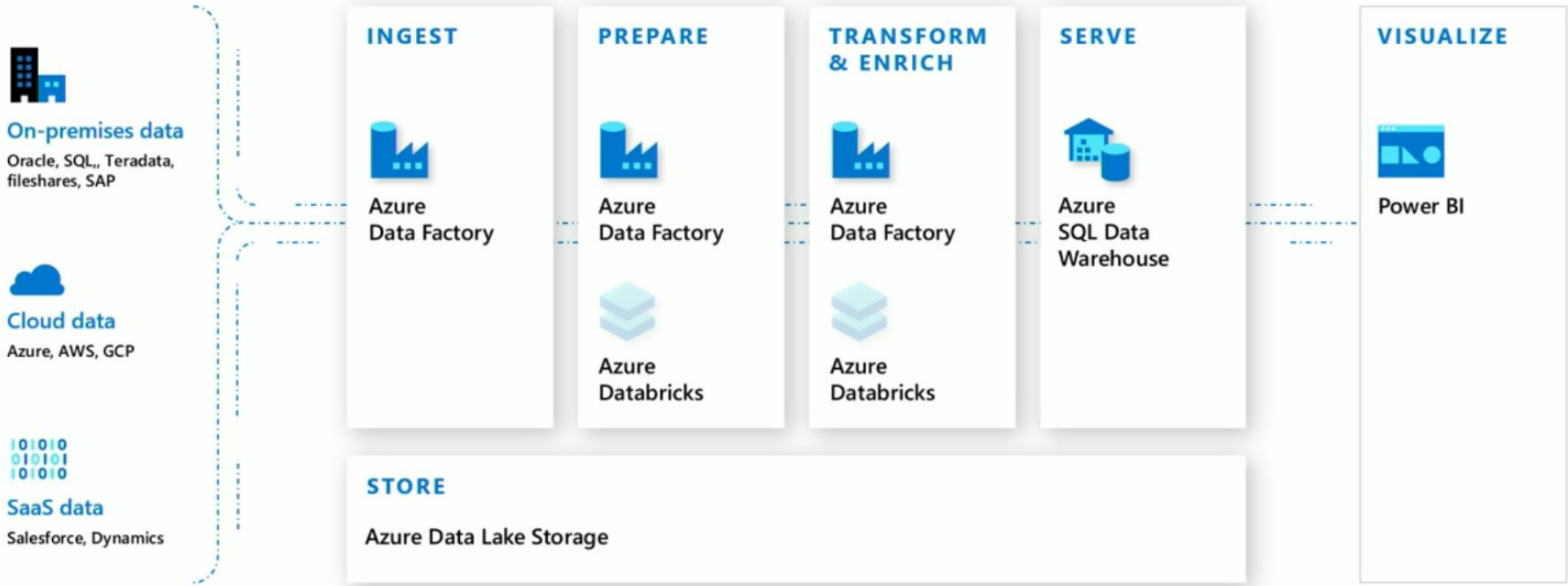
Azure brings these two worlds together, in a single service, to provide limitless analytics



Data warehousing & big data analytics—all in one service


Ecosystem
Azure Synapse Analytics
Power BI
Azure Machine Learning

Modern Data Warehouse



Azure Synapse Analytics - *Data Lakehouse*

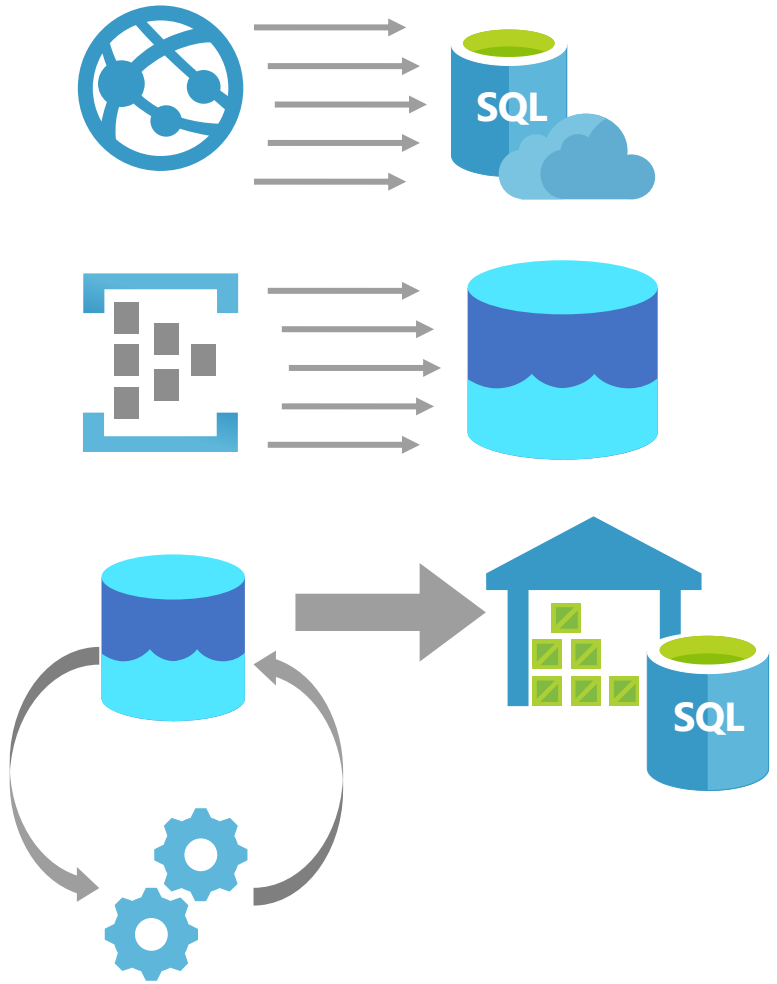
What workloads are NOT suitable?

Operational workloads (OLTP)

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

Data Preparations

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



What Workloads are Suitable?

Analytics

Store large volumes of data.

Consolidate disparate data into a single location.

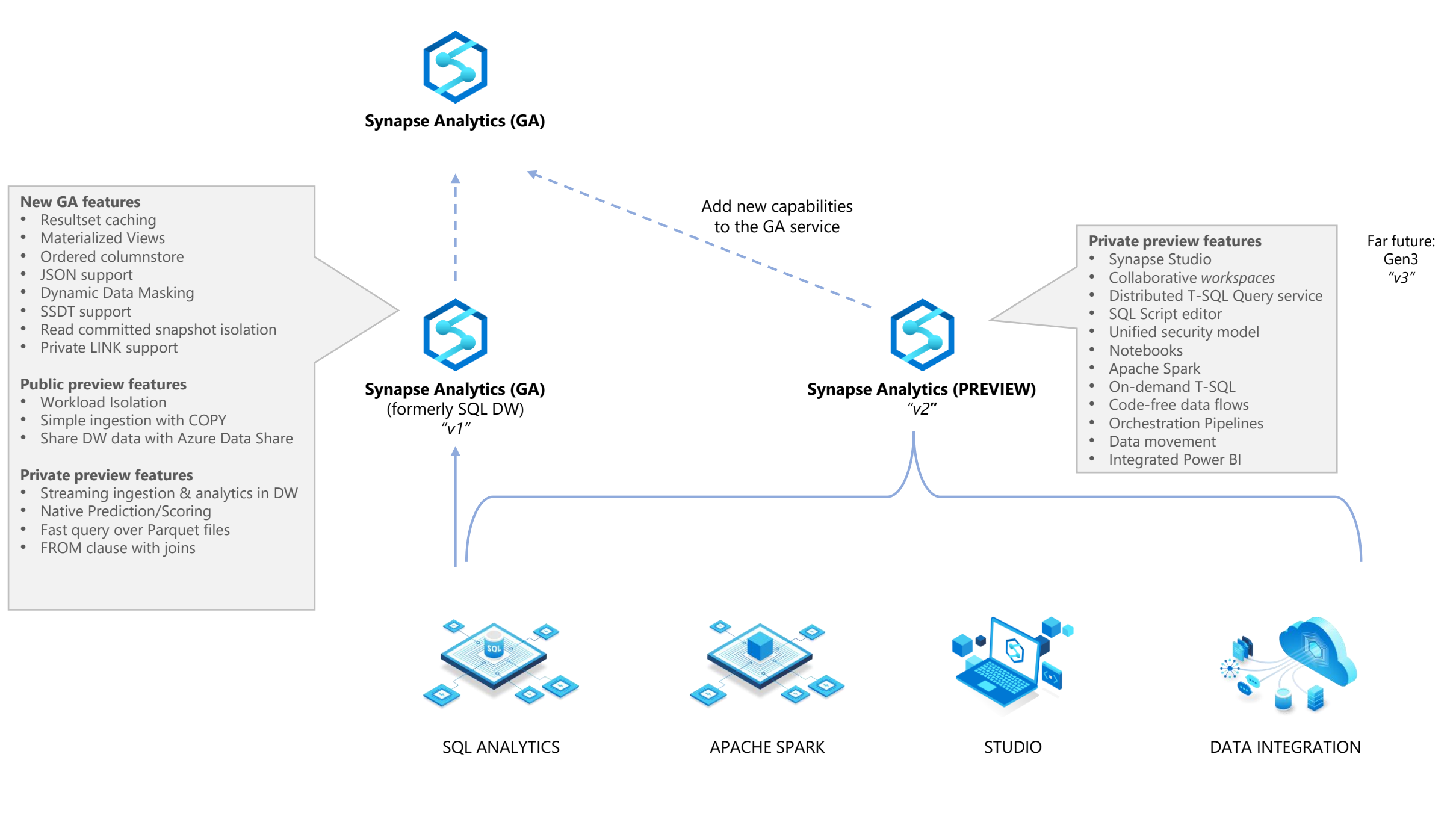
Shape, model, transform and aggregate data.

Batch/Micro-batch loads.

Perform query analysis across large datasets.

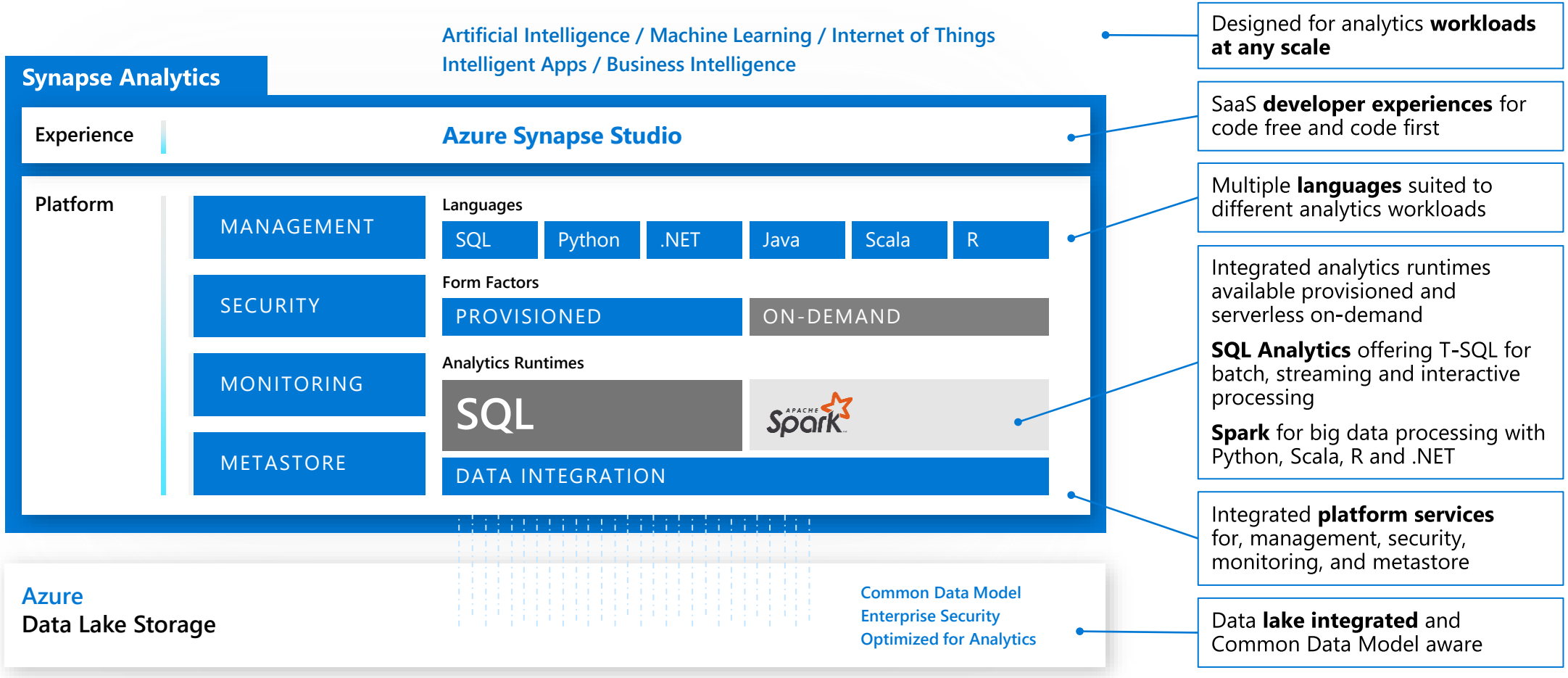
Ad-hoc reporting across large data volumes.

All using simple SQL constructs.


Synapse Analytics (GA)
Add new capabilities
to the GA service
New GA features
• Resultset caching
• Materialized Views
• Ordered columnstore
• JSON support
• Dynamic Data Masking
• SSDT support
• Read committed snapshot isolation
• Private LINK support
Public preview features
• Workload Isolation
• Simple ingestion with COPY
• Share DW data with Azure Data Share
Private preview features
• Streaming ingestion & analytics in DW
• Native Prediction/Scoring
• Fast query over Parquet files
• FROM clause with joins
Synapse Analytics (GA)
(formerly SQL DW)
"v1"
Synapse Analytics (PREVIEW)
"v2"
Private preview features
• Synapse Studio
• Collaborative workspaces
• Distributed T-SQL Query service
• SQL Script editor
• Unified security model
• Notebooks
• Apache Spark
• On-demand T-SQL
• Code-free data flows
• Orchestration Pipelines
• Data movement
• Integrated Power BI
Far future:
Gen3
"v3"
SQL
SQL ANALYTICS
APACHE SPARK
STUDIO
DATA INTEGRATION

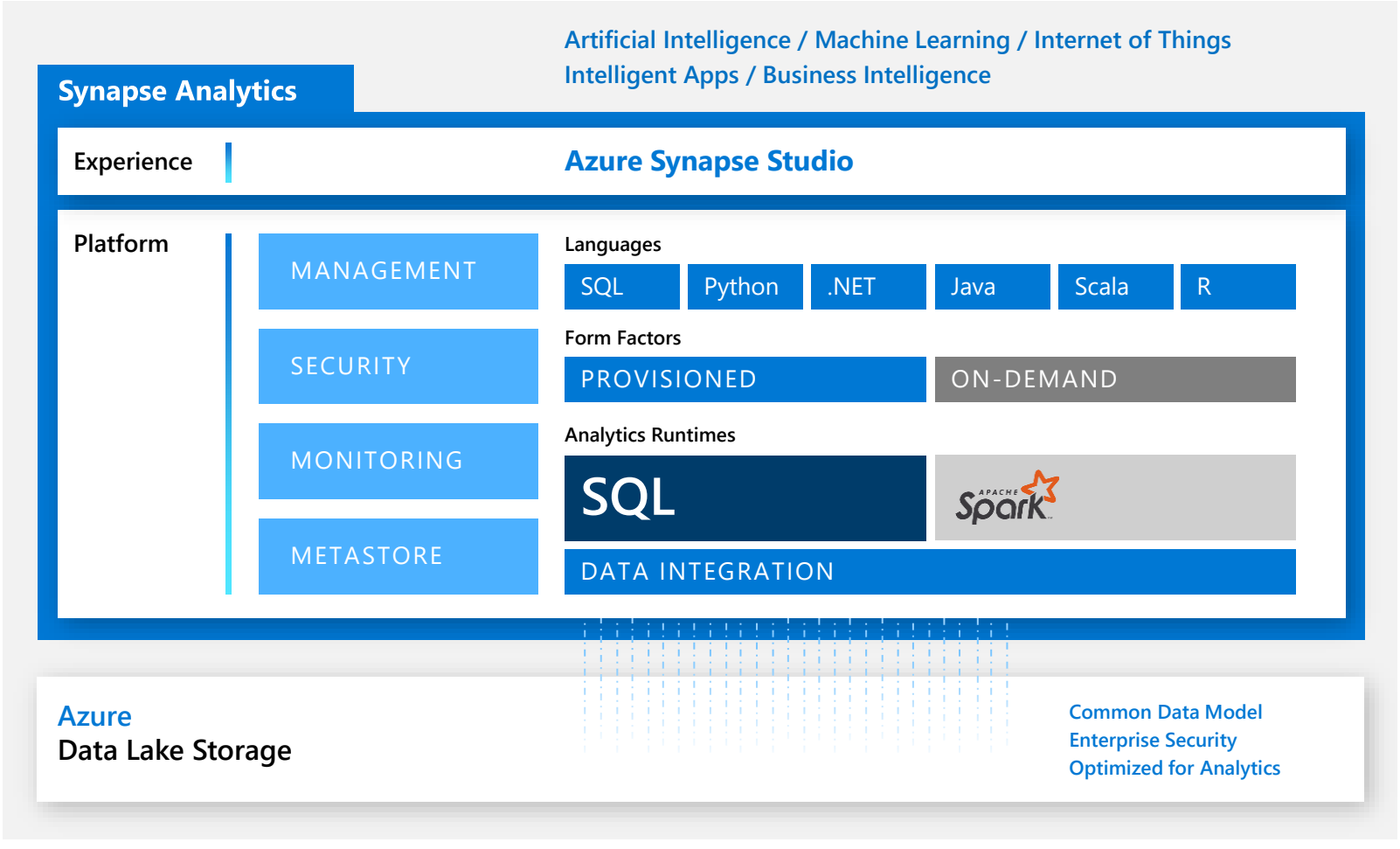
Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



Connected Services

Azure Data Catalog
Azure Data Lake Storage
Azure Data Share
Azure Databricks
Azure HDInsight
Azure Machine Learning
Power BI

3rd Party Integration

New Products/Features

- Azure Synapse Analytics – Umbrella name. For now just includes SQL DW. In preview adds a Synapse Workspace which includes SQL DW and all the new product/features below
- SQL pool or SQL analytics pool – Really just SQL Data Warehouse (SQL DW) which includes compute and storage
- Azure Synapse Studio – New product. Single pain of glass that is a web-based experience. Collaborative workspaces. Access SQL Databases, Spark tables, SQL Scripts, notebooks (supports multiple languages), Data flows (Data Integration), pipelines (Data Integration), monitoring, security. Has links to ADLS Gen2 and Power BI workspace
- Data Integration – Really just Azure Data Factory (ADF). They use the same code base. Note in Synapse Studio Data Flows are under "Develop", Pipelines are under "Orchestrate", and Datasets are under "Data" (In ADF they are all under "Author")
- Spark – Including Apache Spark is new. Similar to Spark in SQL Server 2019 BDC
- On-demand T-SQL – New feature. Was code-named Starlight Query
- T-SQL over ADLS Gen2 – New feature. Was code-named Starlight Query
- New SQL DW features (see next slides) – Some are GA now and some are in preview
- Multiple query options (see next slides) – Some are GA now and some are in preview
- Distributed Query Processor (see next slides) – some in preview or Gen3

New Synapse Features

GA features:

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

Public Preview features:

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

Private Preview features:

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

Note: private preview features require whitelisting.

Query Options

1. Provisioned SQL over relational database – Traditional SQL DW [existing]
2. Provisioned SQL over ADLS Gen2 – via external tables or openrowset [existing via external tables in PolyBase, openrowset not yet in preview]
3. *On-demand SQL over relational database - dependency on the flexible data model (data cells) over columnstore data [new, not yet in preview: the ability to query a SQL relational database (and other types of data sources) will come later]*
4. On-demand SQL over ADLS Gen2 – via external tables or openrowset [new in preview]
5. Provisioned Spark over relational database – [new in preview]
6. Provisioned Spark over ADLS Gen2 [new in preview]
7. *On-demand Spark over relational database - On-demand Spark is not supported (but provisioned Spark can auto-pause)*
8. *On-demand Spark over ADLS Gen2 – On-demand Spark is not supported (but provisioned Spark can auto-pause)*

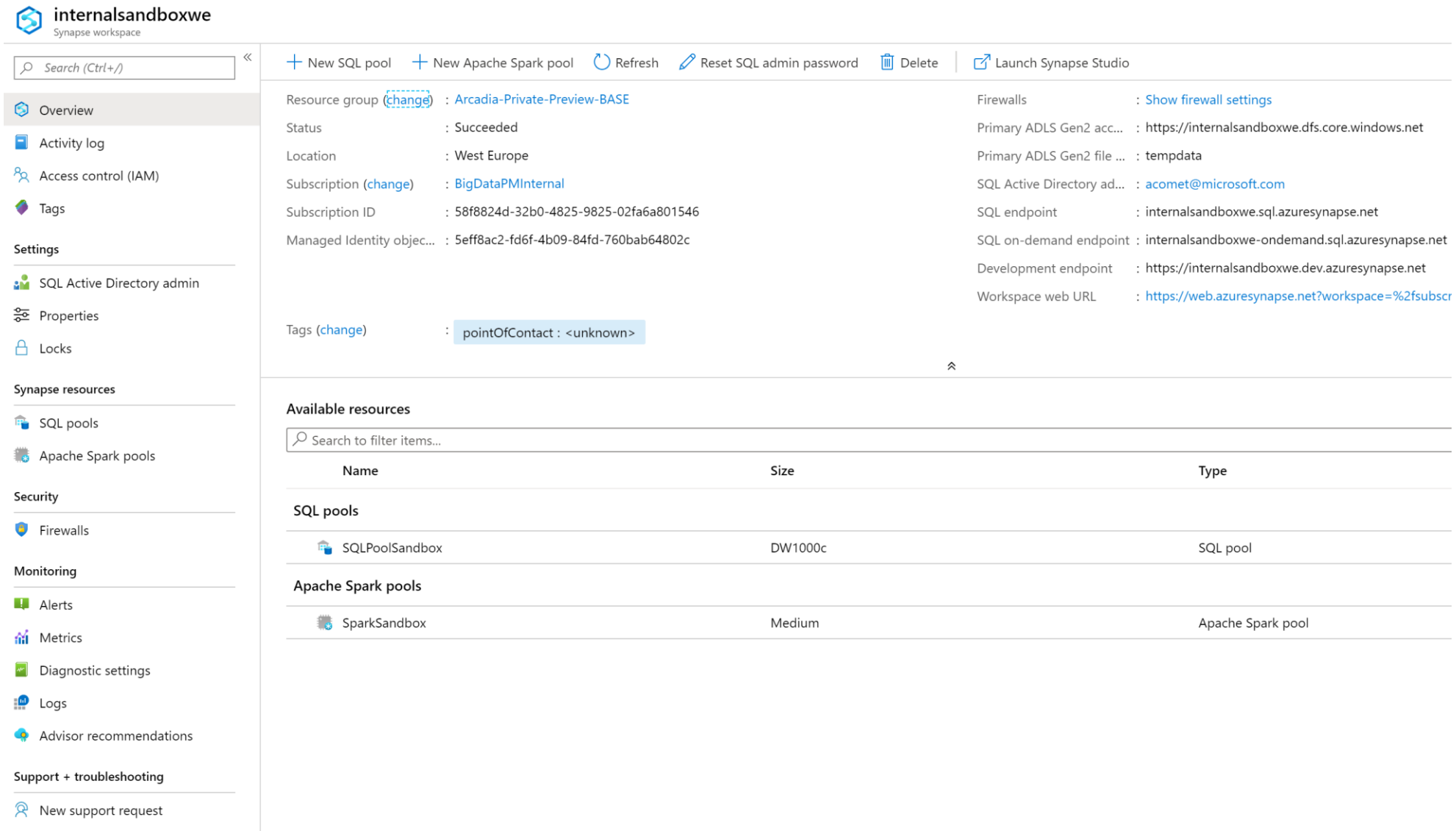
Notes:

- Separation of state (data, metadata and transactional logs) and compute
- Queries against data loaded into SQL Analytics tables are 2-3X faster compared to queries over external tables
- Copy statement: Improved performance compared to PolyBase. PolyBase is not used, but functional aspects are supported
- Warm-up for first on-demand SQL query takes about 30-40 seconds
- If you create a Spark Table, that table will be created as an external table in SQL Pool or SQL On-Demand without having to keep a Spark cluster up and running
- Currently one on-demand SQL pool but by GA will support many
- Provisioned SQL may give you better and more predictable performance due to resource reservation
- Existing PolyBase via external tables is not pushdown (#2), but #4 will be pushdown (SQL on-demand will push down queries from the front-end to back-end nodes)
- Supported file formats are parquet, csv, json
- ***Each SQL pool can currently only access tables created within its pool (there is one database per pool), while on-demand SQL can not yet query a database***

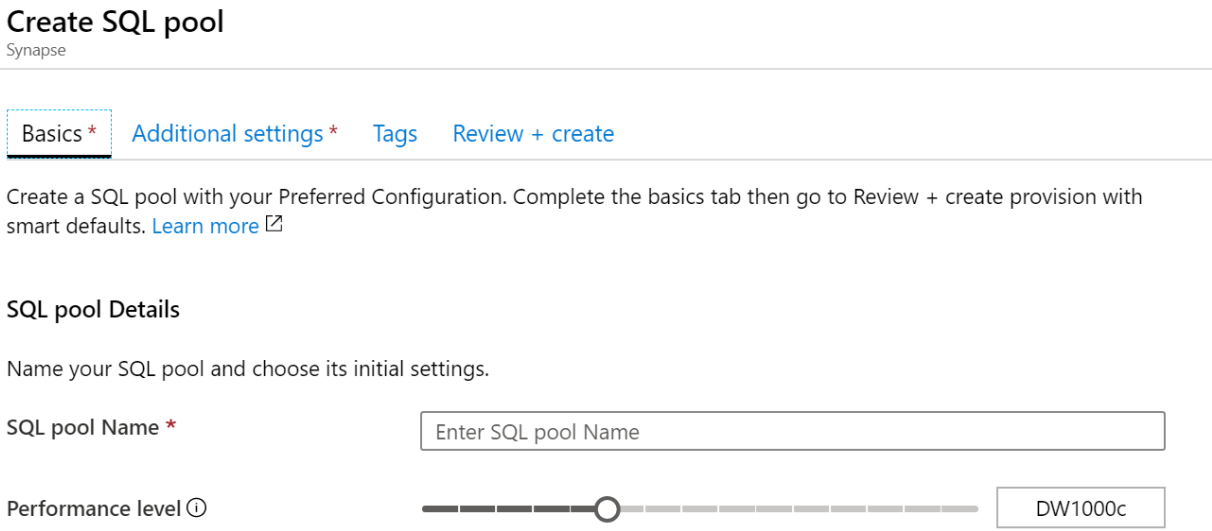
Azure Synapse Analytics features

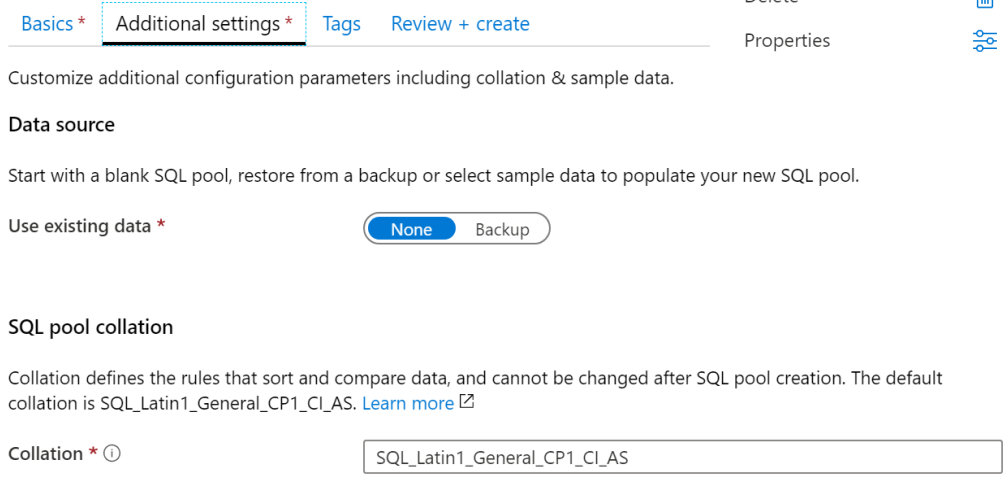
Limitless scale	GA	Preview
Provisioned compute (data warehouse)	✓	
Materialized views	✓	
Workload importance	✓	
Workload isolation		✓
On-demand query		✓
Powerful insights		
Power BI integration		✓
Azure Machine Learning integration		✓
Data lake exploration		✓
Streaming analytics (data warehouse)		✓
Apache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

Synapse workspace



SQL pools





SQL pool = SQL Data Warehouse

Apache Spark pools

New Refresh

Search to filter items...

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

Auto-pause
Scale
Delete
Properties

Create Apache Spark pool

Basics * Additional settings * Tags Summary

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

Apache Spark pool details

Name your Apache Spark pool and choose its initial settings.

Apache Spark pool name * Enter Apache Spark pool name

Node size family MemoryOptimized

Node size * Medium (8 vCPU / 64 GB)

Autoscale * Enabled Disabled

Number of nodes * 3 40

Note: There are no on-demand pools for Spark, but only billed when a Spark pool is active (<3m to start from cold. New session <30s in active pool).

Basics * Additional settings * Tags Summary

Customize additional configuration parameters including autoscale and component versions.

Auto-pause

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

Auto-pause * Enabled Disabled

Number of minutes idle * 15

Component versions

Select the Apache Spark version for your Apache Spark pool.

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

Packages

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

File upload Select a file

Microsoft

Azure Synapse Analytics

MPP Intro

Parallelism

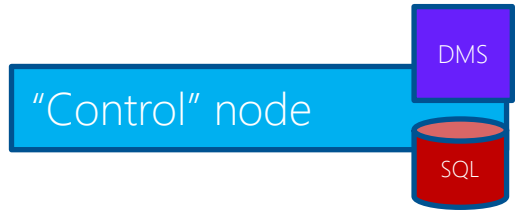
SMP - Symmetric Multiprocessing

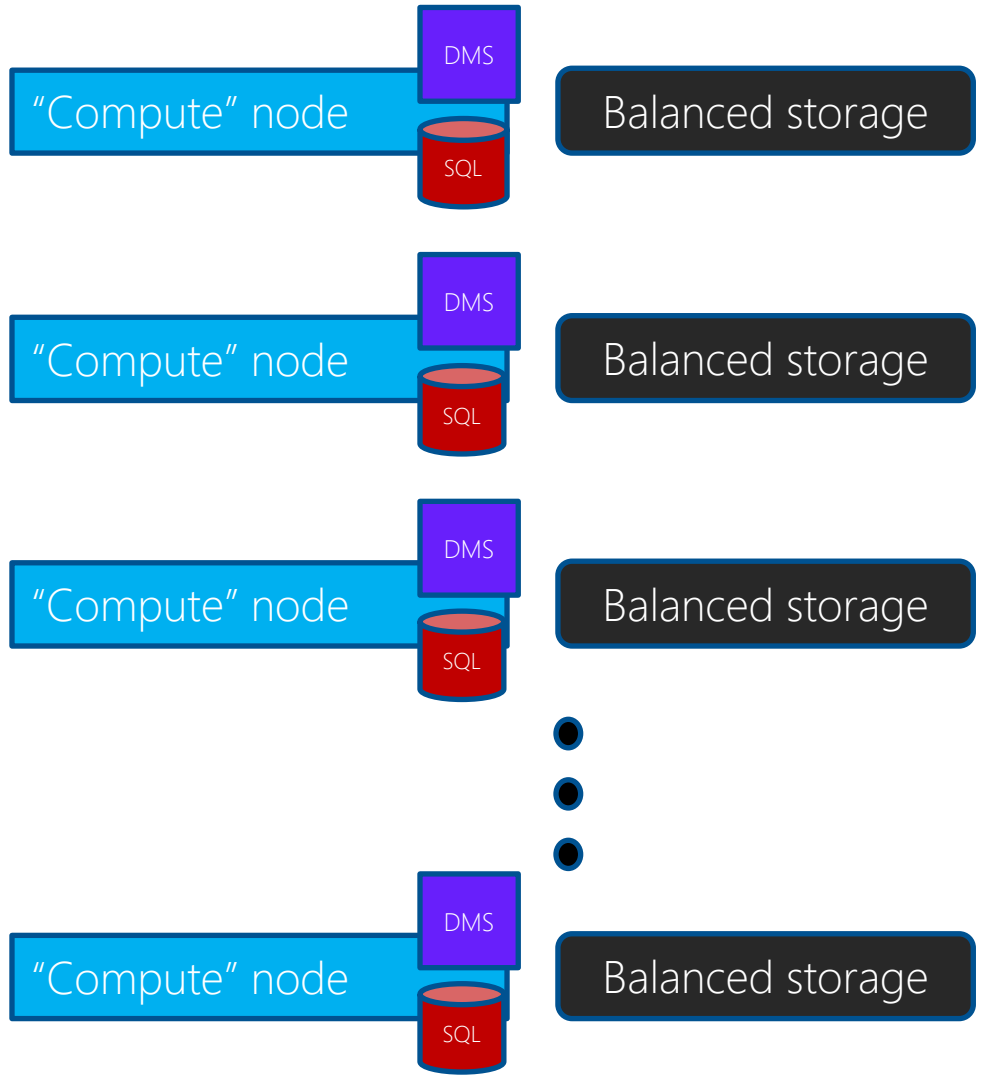
- Multiple CPUs used to complete individual processes simultaneously
- All CPUs share the same memory, disks, and network controllers (scale-up)
- All SQL Server implementations up until now have been SMP
- Mostly, the solution is housed on a shared SAN

MPP - Massively Parallel Processing

- Uses many separate CPUs running in parallel to execute a single program
- Shared Nothing: Each CPU has its own memory and disk (scale-out)
- Segments communicate using high-speed network between nodes

SQL DW Logical Architecture (overview)





Compute Node – the "worker bee" of SQL DW

- Runs Azure SQL Server DB
- Contains a "slice" of each database
- CPU is saturated by storage

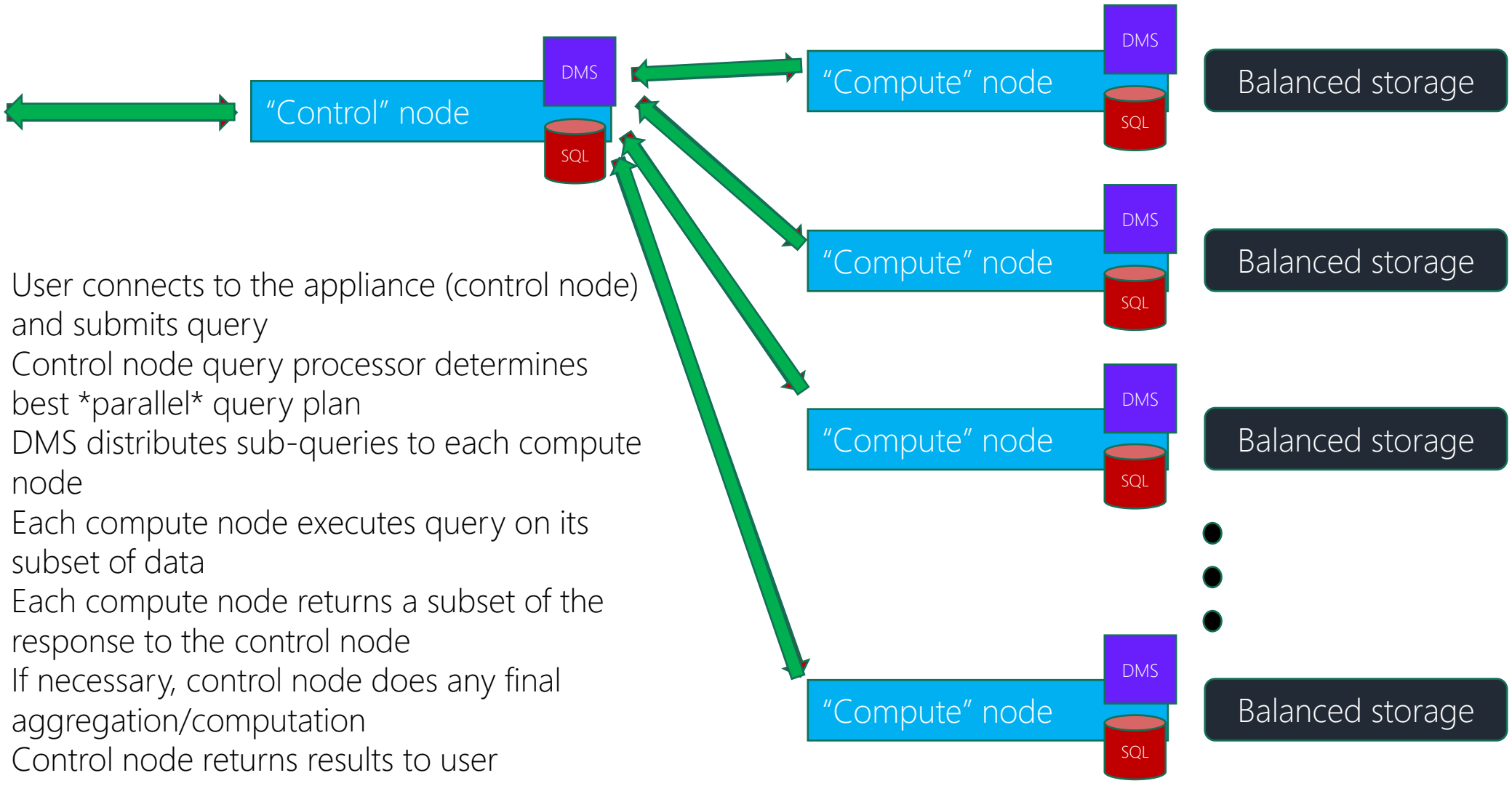
Control Node – the "brains" of the SQL DW

- Also runs Azure SQL Server DB
- Holds a "shell" copy of each database
  - Metadata, statistics, etc
- The "public face" of the appliance

Data Movement Services (DMS)

- Part of the "secret sauce" of SQL DW
- Moves data around as needed
- Enables parallel operations among the compute nodes (queries, loads, etc)

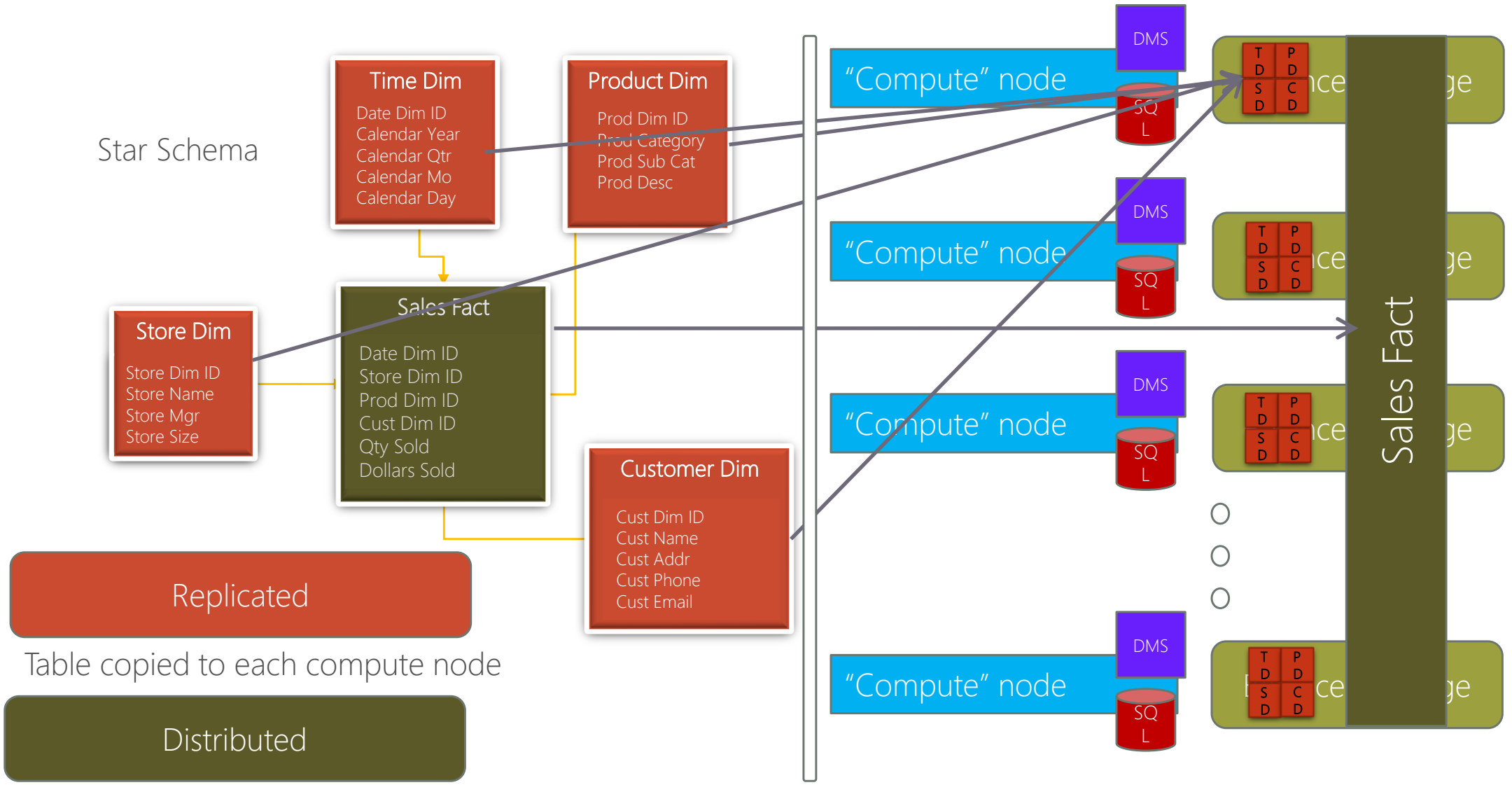
SQL DW Logical Architecture (overview)



1) User connects to the appliance (control node) and submits query
2) Control node query processor determines best *parallel* query plan
3) DMS distributes sub-queries to each compute node
4) Each compute node executes query on its subset of data
5) Each compute node returns a subset of the response to the control node
6) If necessary, control node does any final aggregation/computation
7) Control node returns results to user

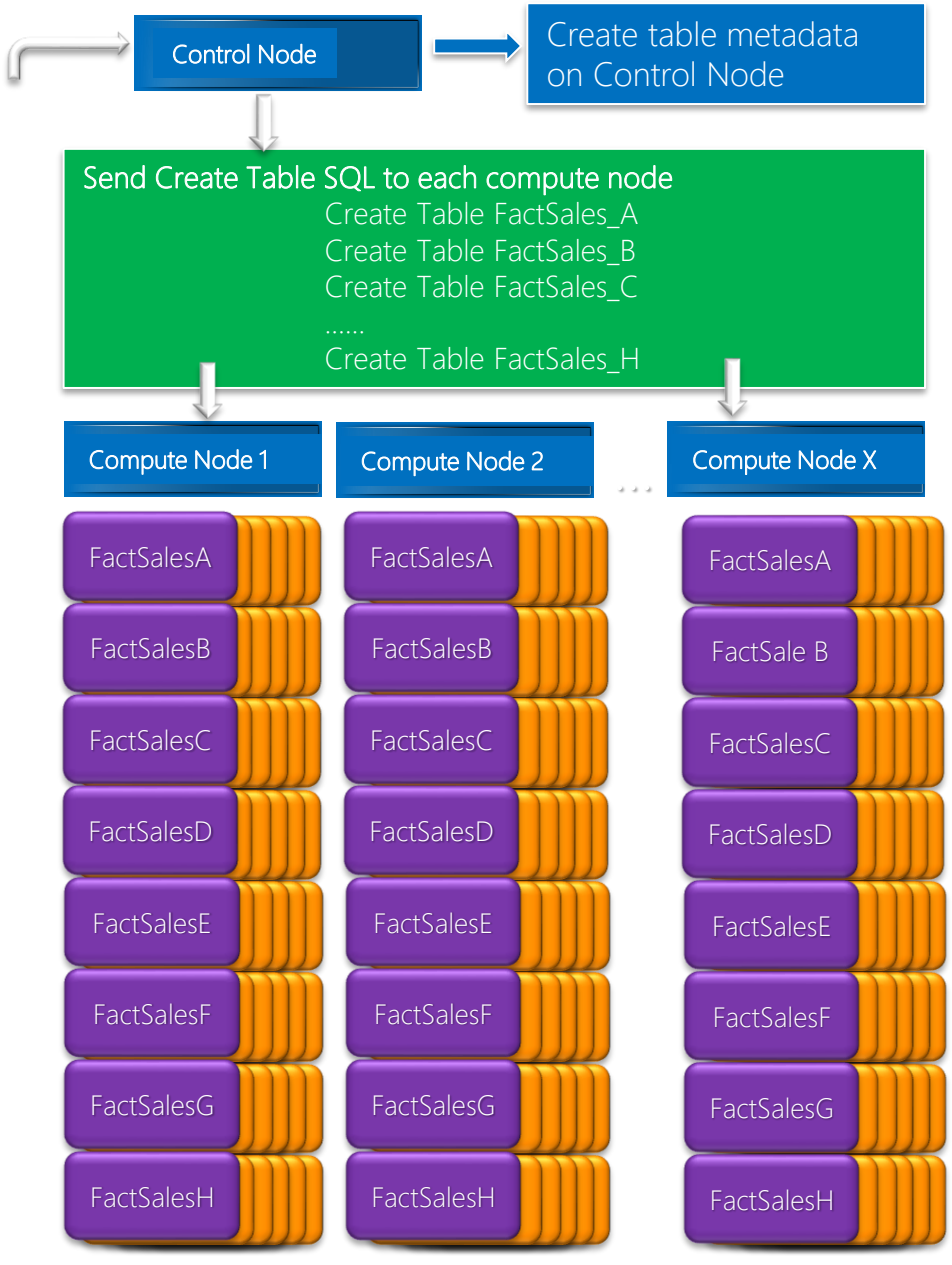
Queries running in parallel on a subset of the data, using separate pipes effectively making the pipe larger

SQL DW Data Layout Options



DATA DISTRIBUTION

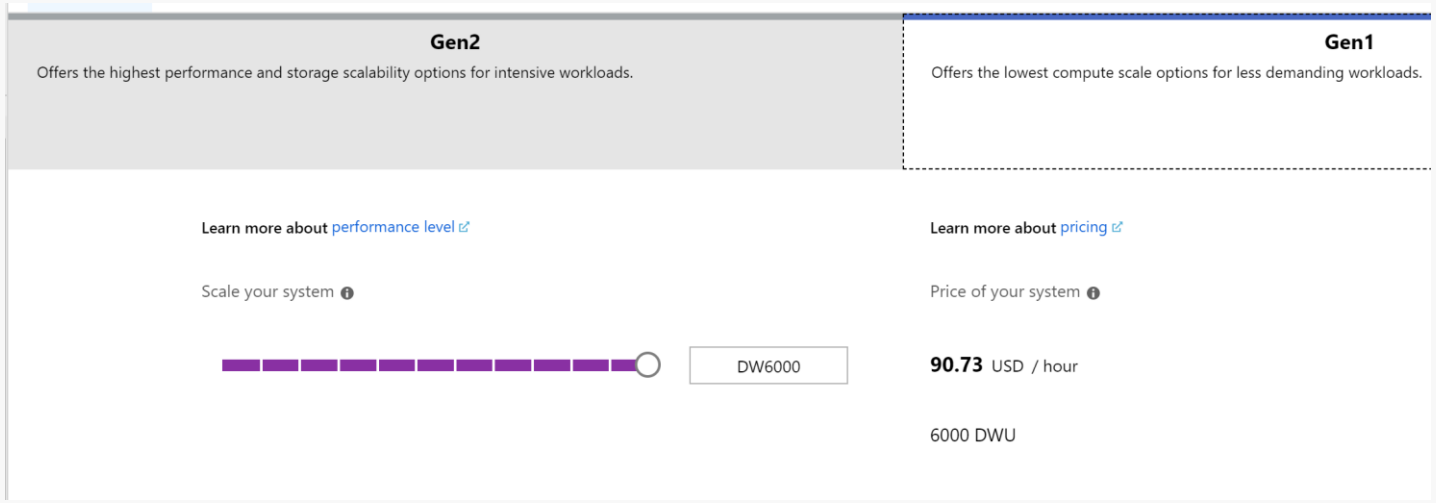
```
CREATE TABLE FactSales
        (
        ProductKey                      INT NOT NULL       ,
        OrderDateKey                    INT NOT NULL       ,
        DueDateKey                      INT NOT NULL       ,
        ShipDateKey                     INT NOT NULL       ,
        ResellerKey                     INT NOT NULL       ,
        EmployeeKey                     INT NOT NULL       ,
        PromotionKey                    INT NOT NULL       ,
        CurrencyKey                     INT NOT NULL       ,
        SalesTerritoryKey               INT NOT NULL       ,
        SalesOrderNumber                VARCHAR(20) NOT NULL,
          ) WITH
        (

         DISTRIBUTION = HASH(ProductKey),

        CLUSTERED INDEX(OrderDateKey)  ,

        PARTITION
                (OrderDateKey RANGE RIGHT FOR VALUES
                                ( 20010601,
                                20010901,
                ) ) );
```



Data Warehouse Units (DWU)

DWU
DW100
DW200
DW300
DW400
DW500
DW1000
DW1500
DW2000
DW2500
DW3000
DW5000
DW6000
DW7500
DW10000
DW15000
DW30000

```
ALTER DATABASE ContosoDW MODIFY
(service_objective = 'DW1000');
```

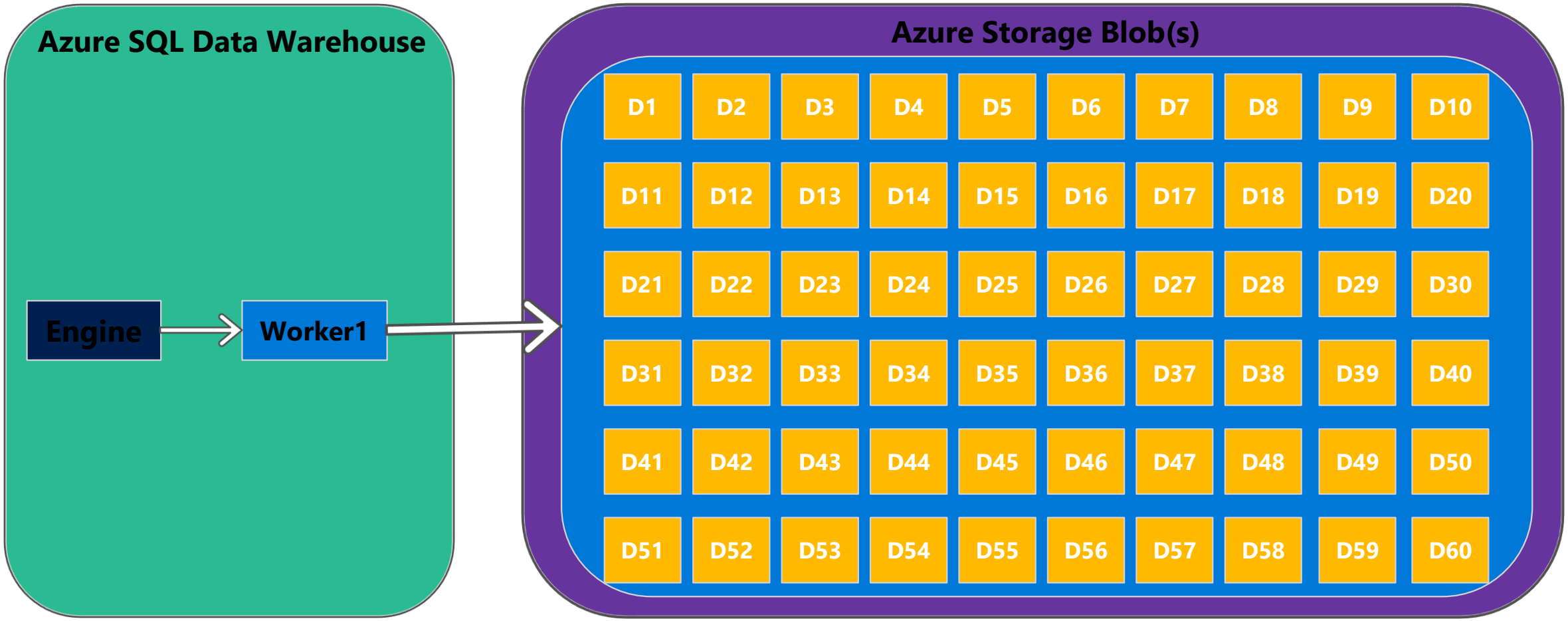


CPU

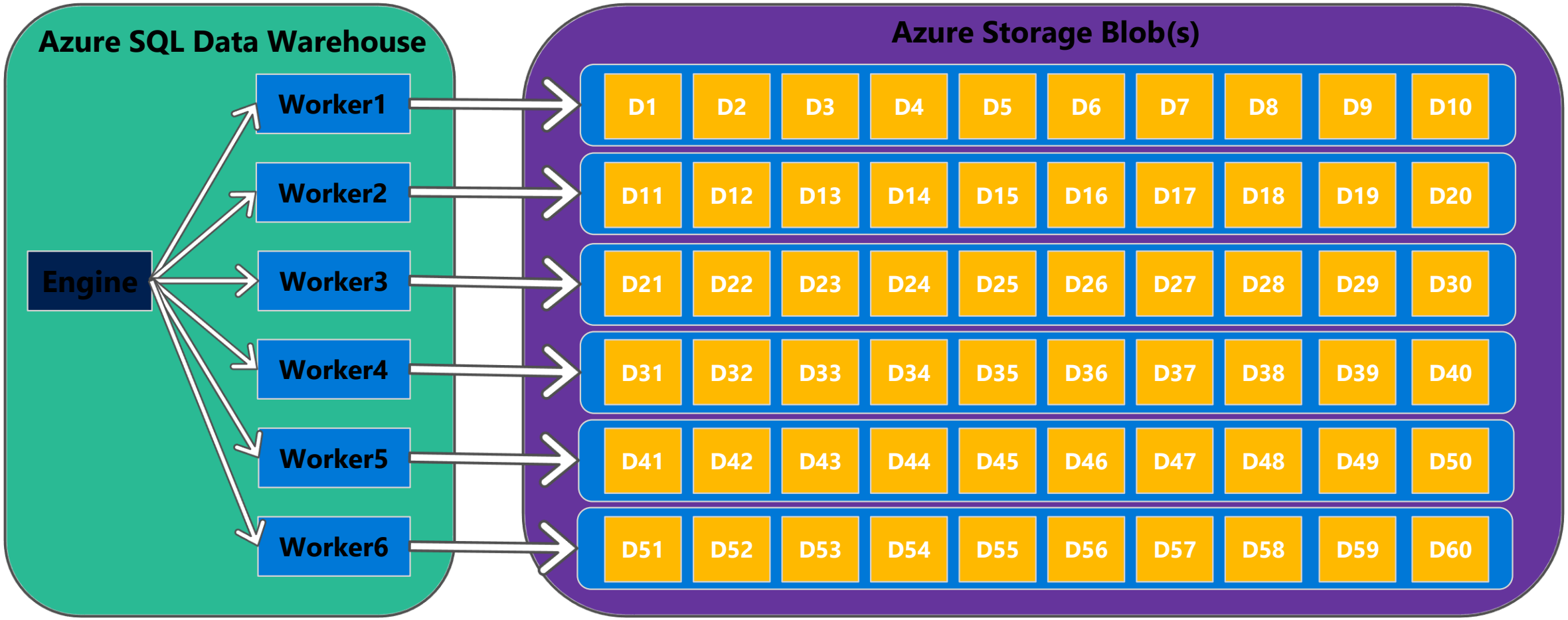
RAM

I/O

Architecture for DW100



Architecture for DW600



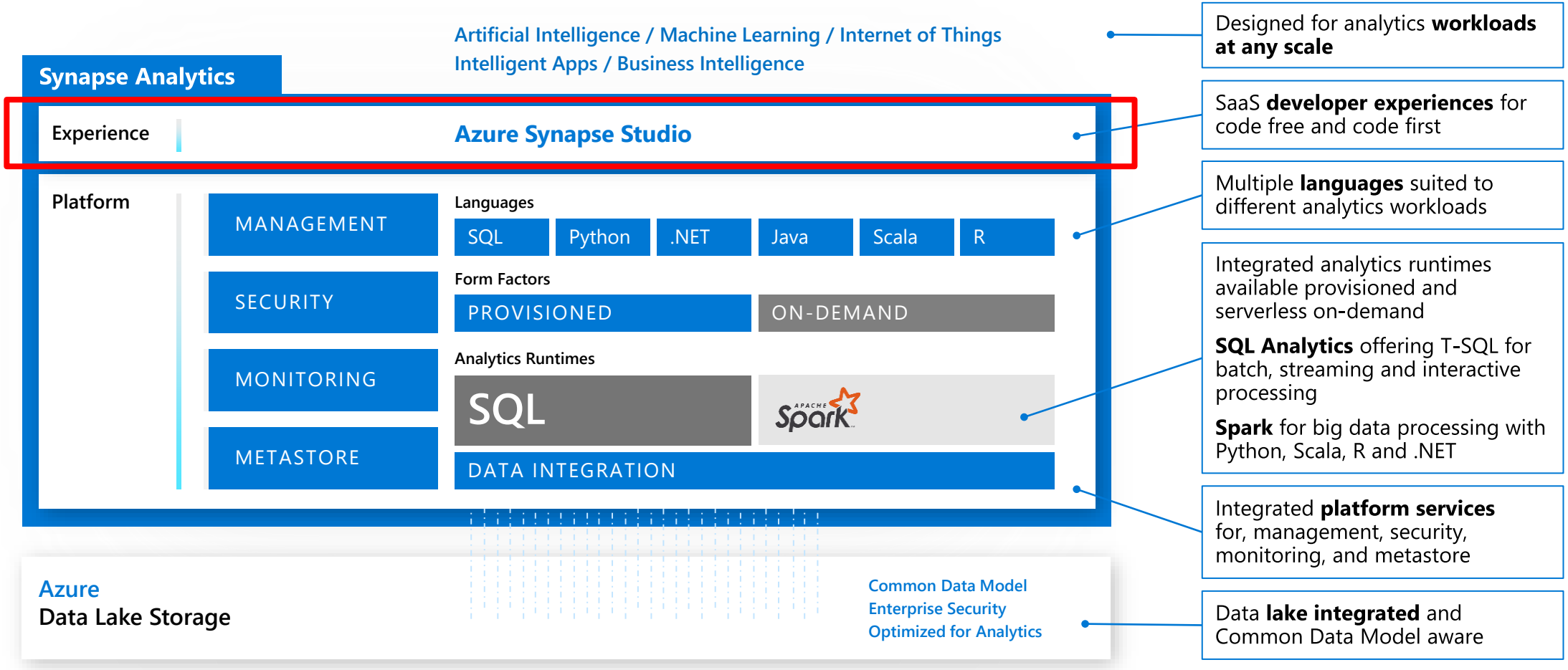
Microsoft

Azure Synapse Analytics

Studio

Azure Synapse Analytics

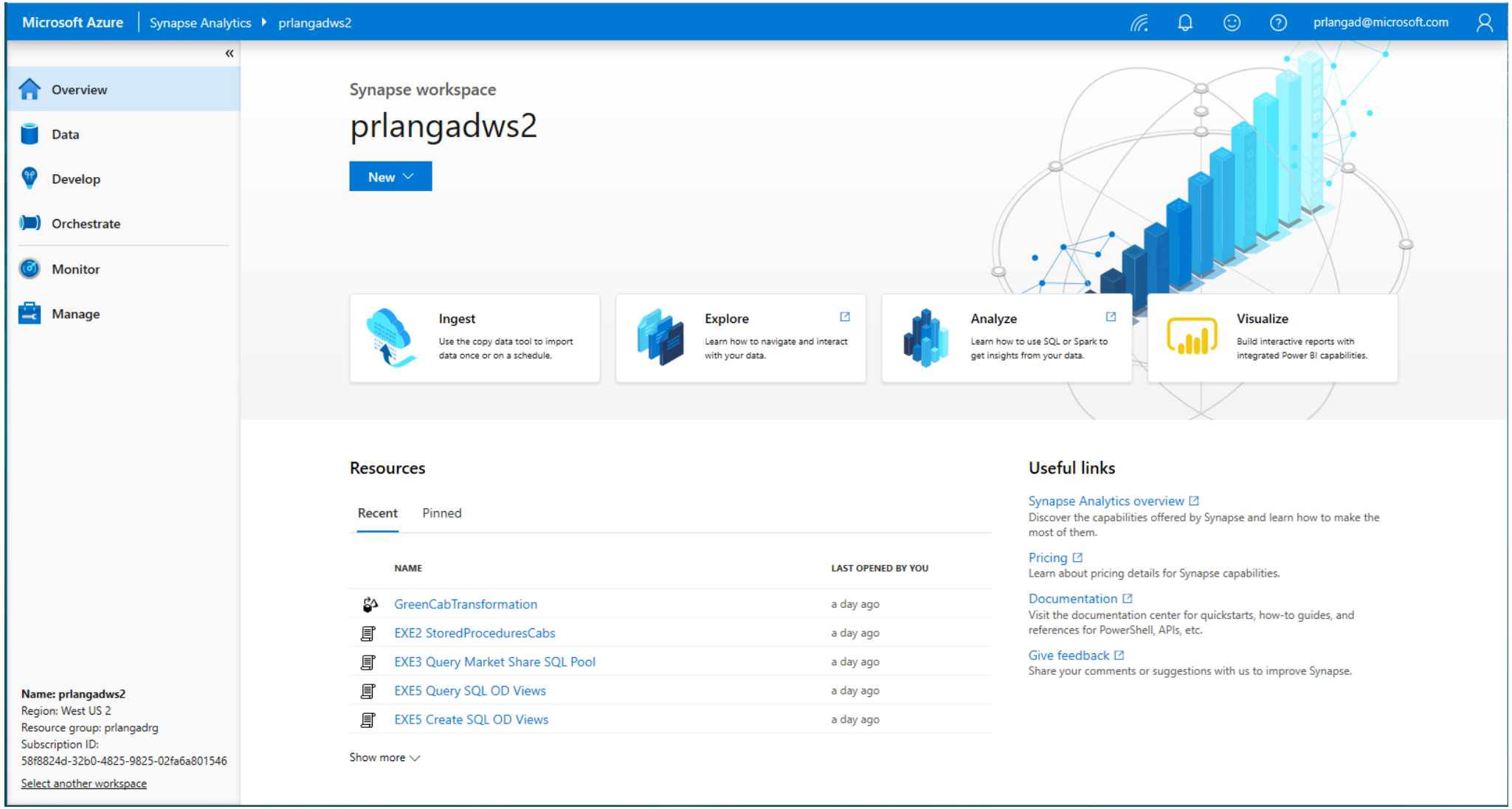
Integrated data platform for BI, AI and continuous intelligence



Studio

https://web.azuresynapse.net

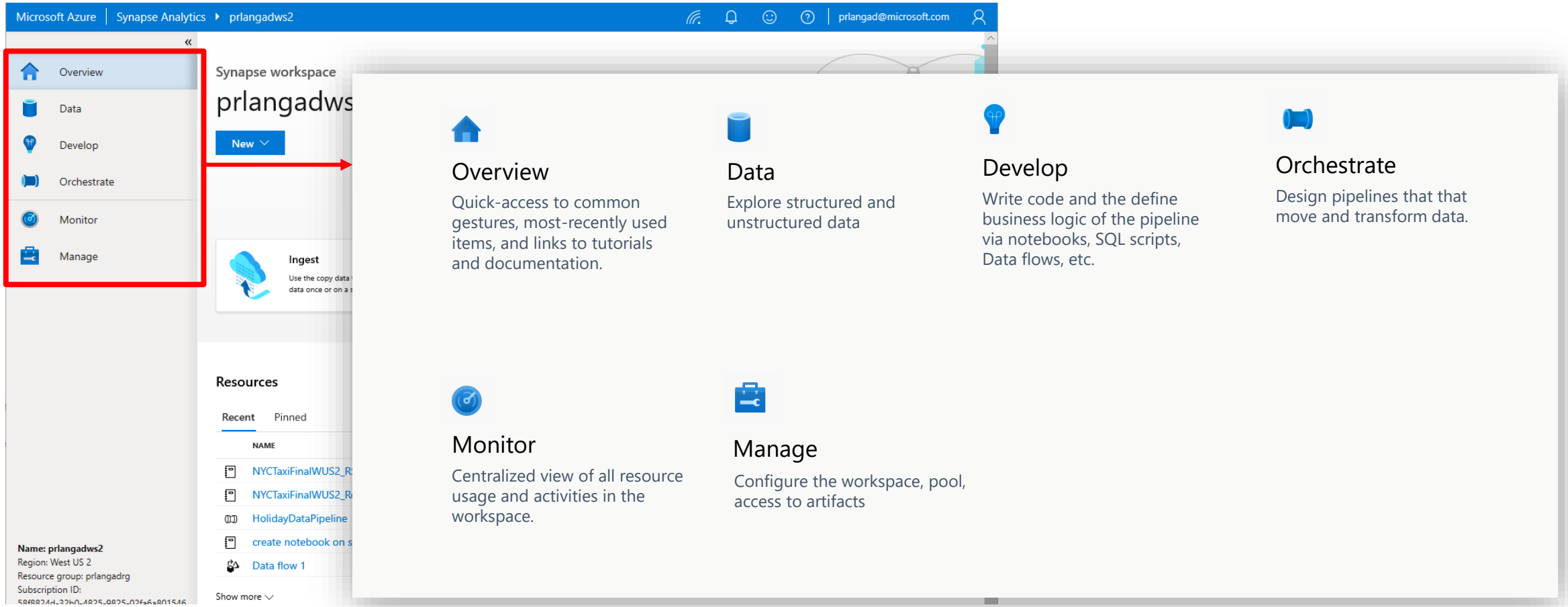
A single place for Data Engineers, Data Scientists, and IT Pros to collaborate on enterprise analytics



Synapse Studio

Synapse Studio divided into **Activity hubs**.

These organize the tasks needed for building analytics solution.



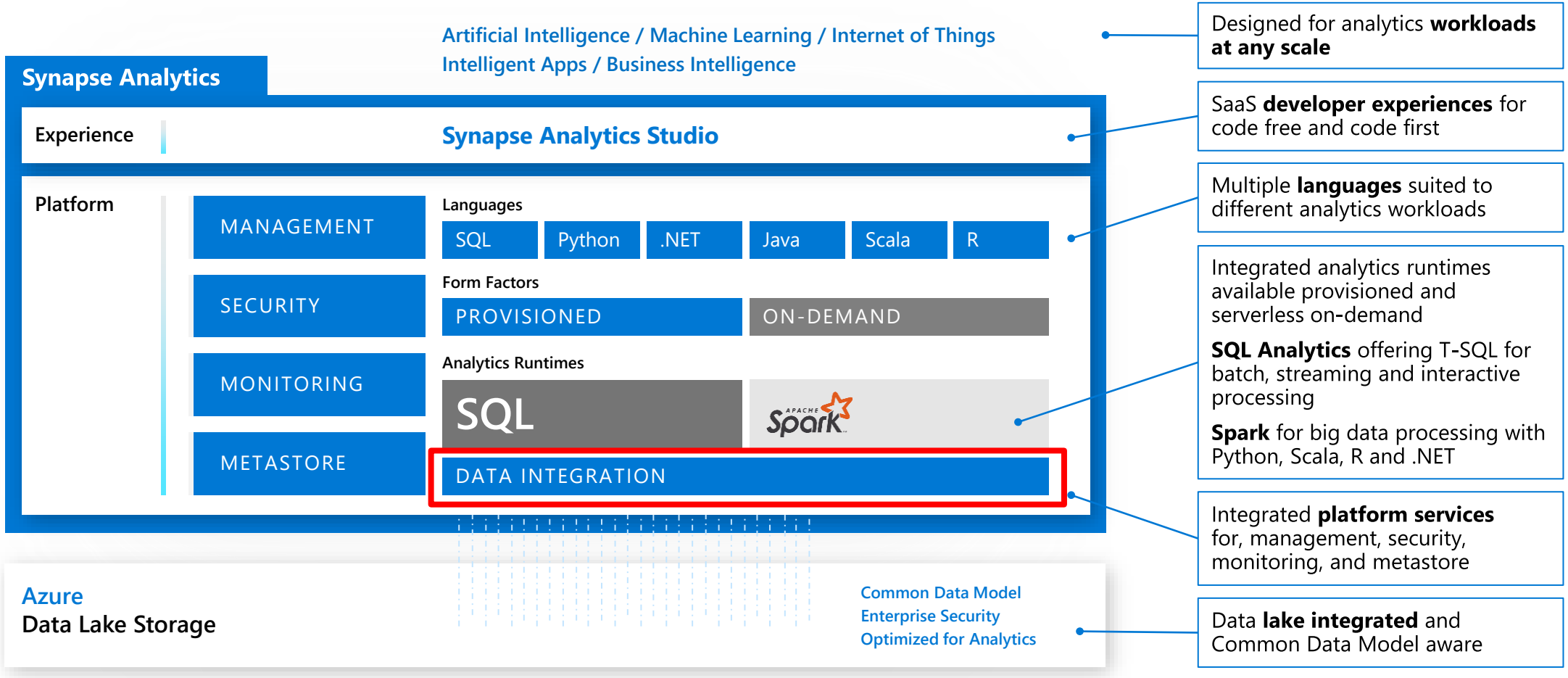
Microsoft

Data Integration

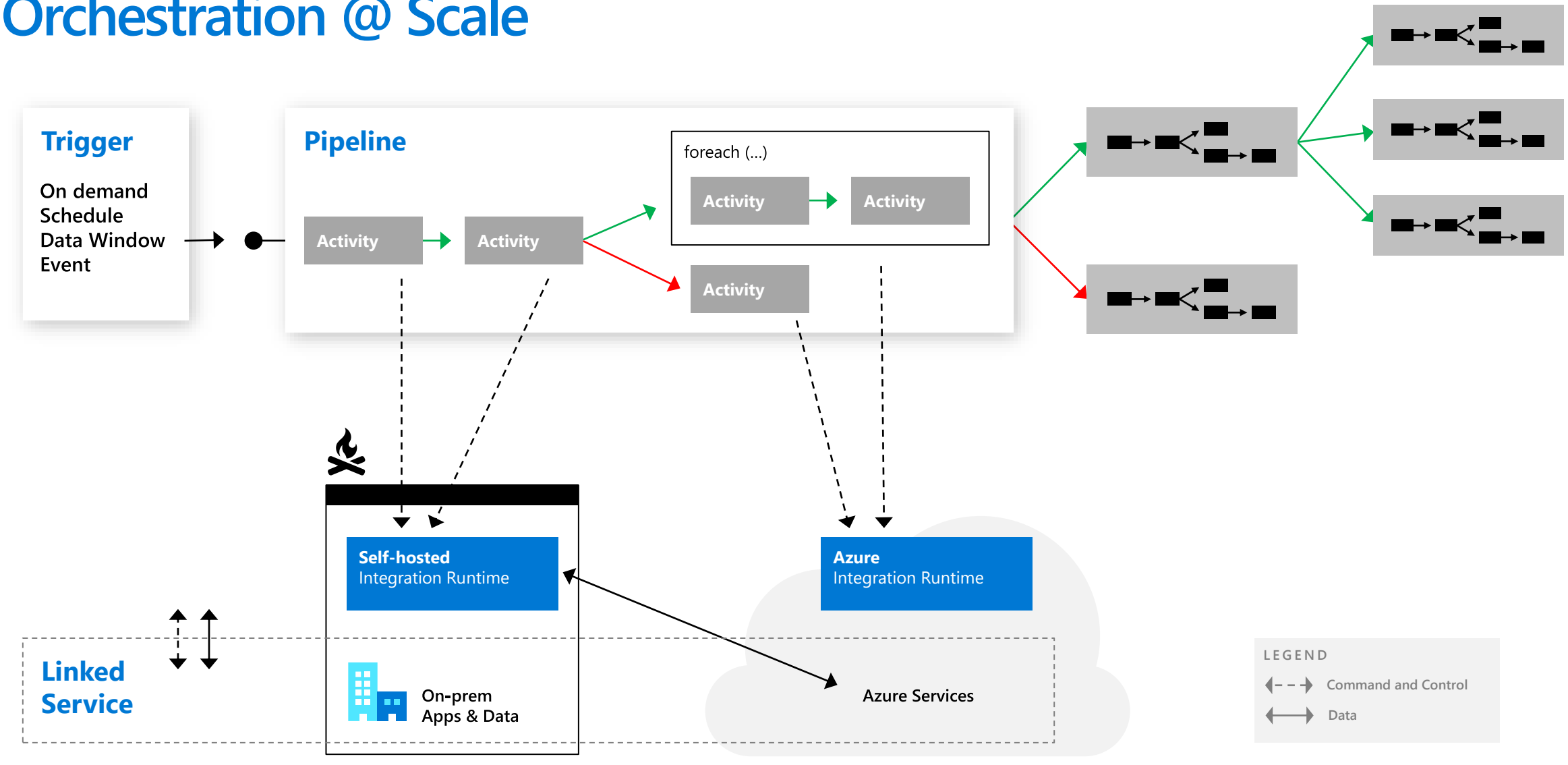
Data Integration

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Orchestration @ Scale



Data Movement

Scalable

per job elasticity

Up to 4 GB/s

Simple

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

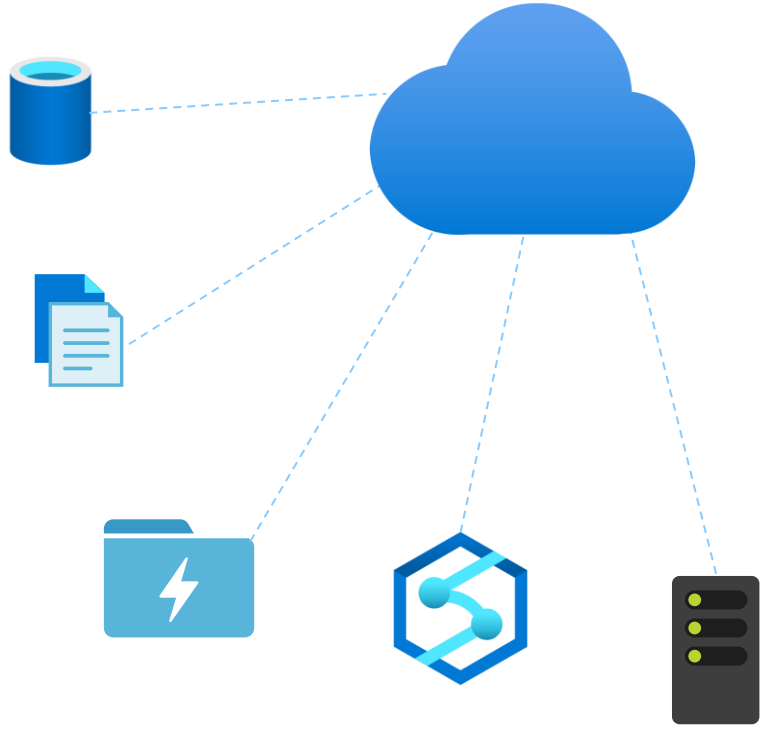
Serverless, no infrastructure to manage

Access all your data

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

Data Movement as a Service: 25 points of presence worldwide

Self-hostable Integration Runtime for hybrid movement



90+ Connectors out of the box

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

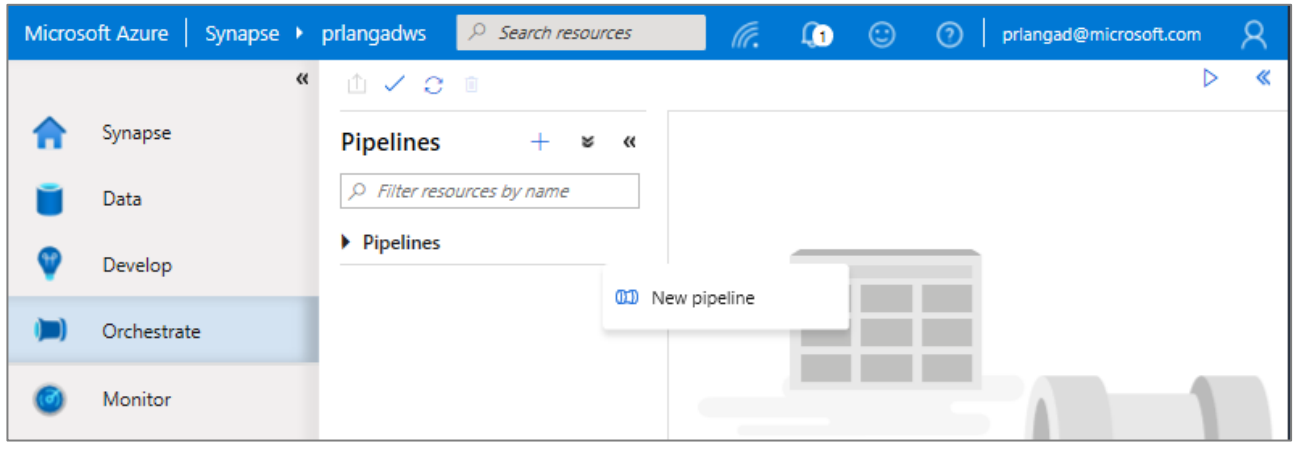
It provides ability to load data from storage account to desired linked service. Load data by manual execution of pipeline or by orchestration

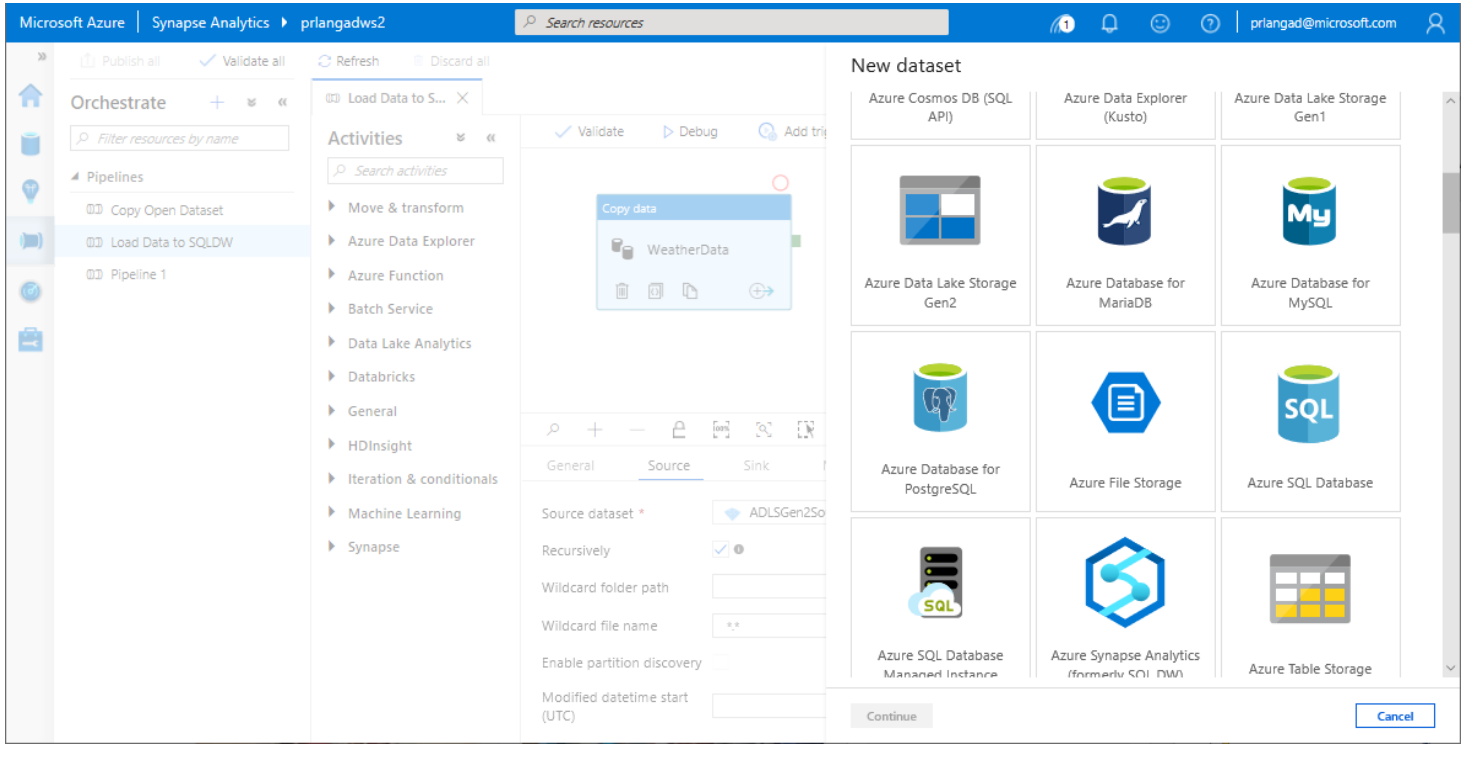
Benefits

Supports common loading patterns

Fully parallel loading into data lake or SQL tables

Graphical development experience

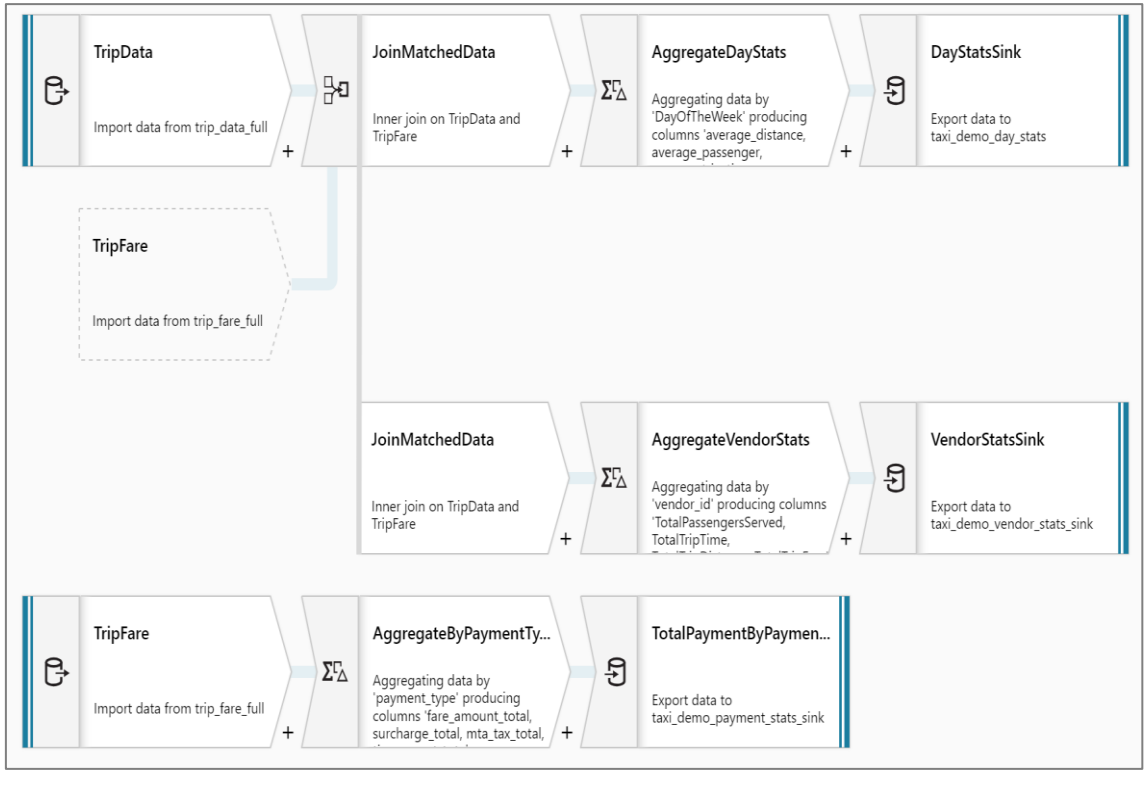




Prep & Transform Data

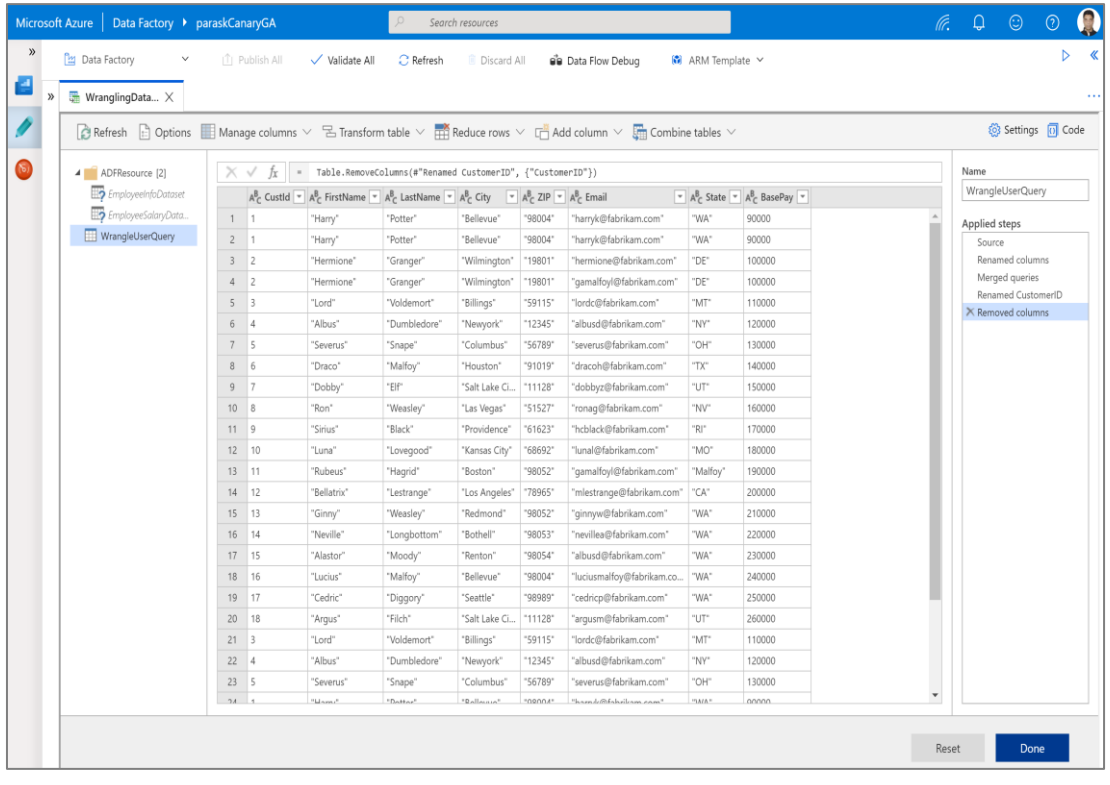
Mapping Dataflow

Code free data transformation @scale



Wrangling Dataflow

Code free data preparation @scale



Triggers

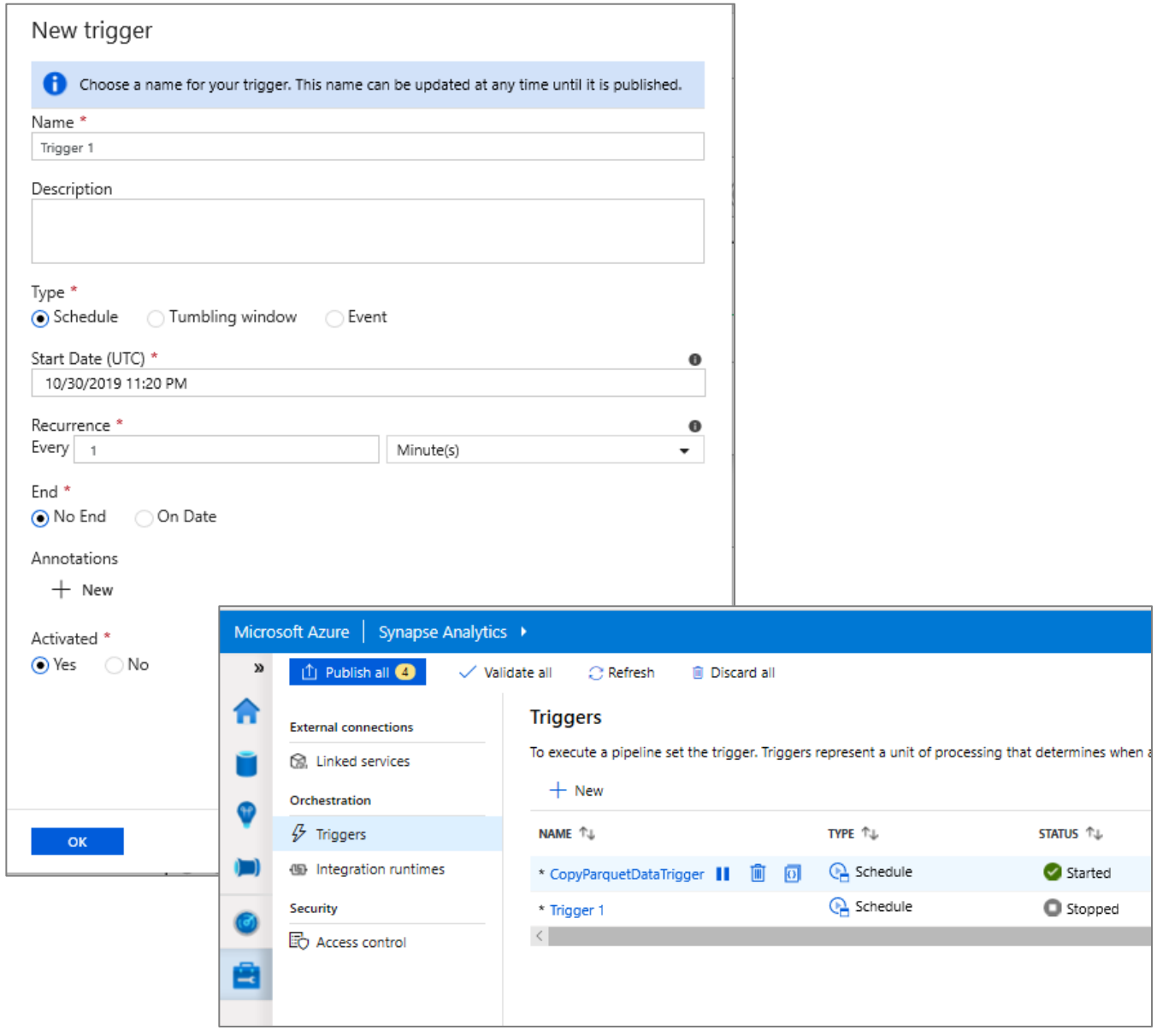
Overview

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.



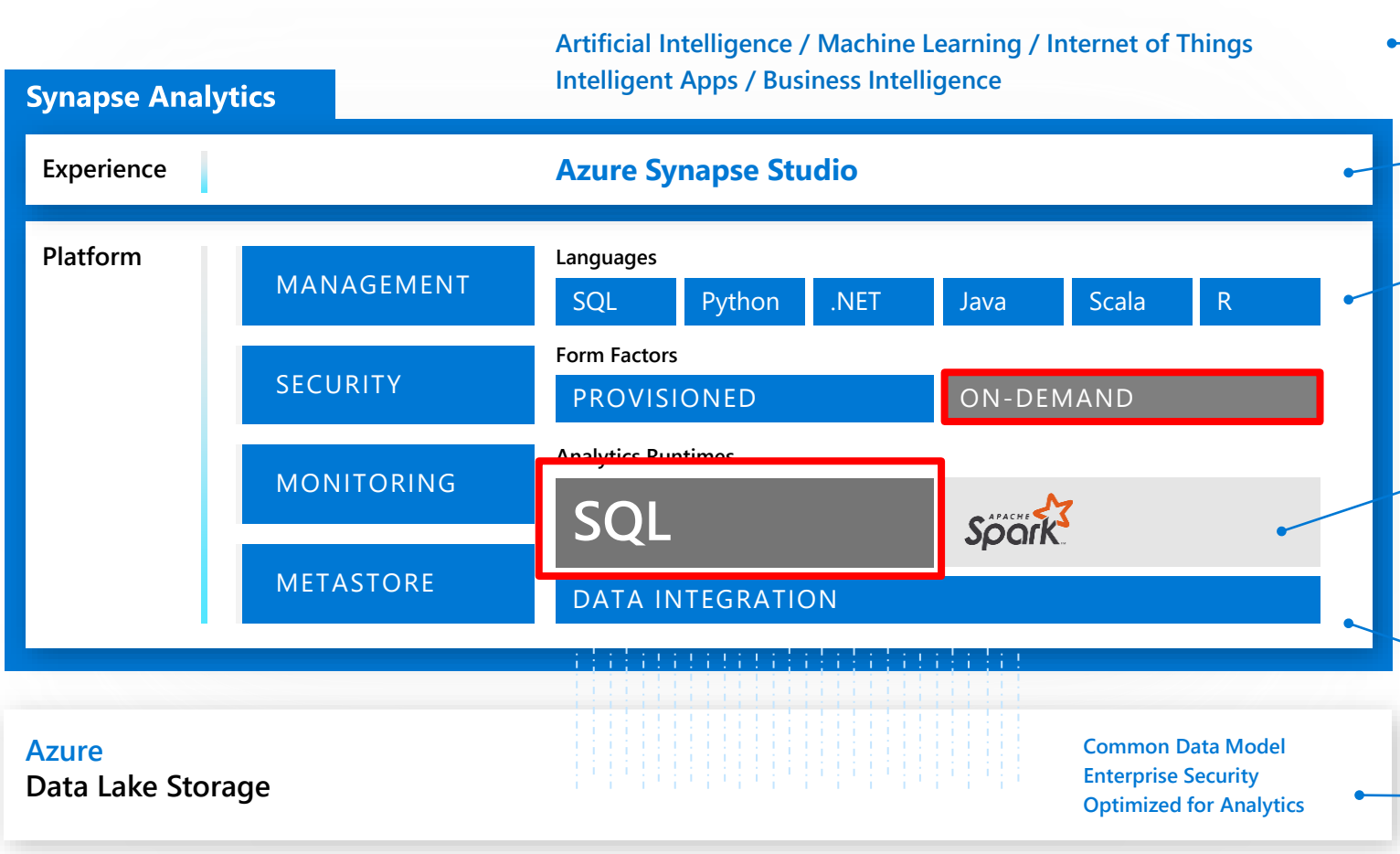
Microsoft

SQL On-Demand

SQL On Demand

Azure Synapse Analytics

Integrated data platform for BI, AI and continuous intelligence



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

Synapse SQL on-demand scenarios

Discovery and exploration	What's in this file? How many rows are there? What's the max value? **SQL On-demand reduces data lake exploration to the right-click!**
Data transformation	How to convert CSVs to Parquet quickly? How to transform the raw data? **Use the full power of T-SQL to transform the data in the data lake**

SQL On-Demand

Overview

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

Benefits

Serverless

No infrastructure

Pay only for query execution

No ETL

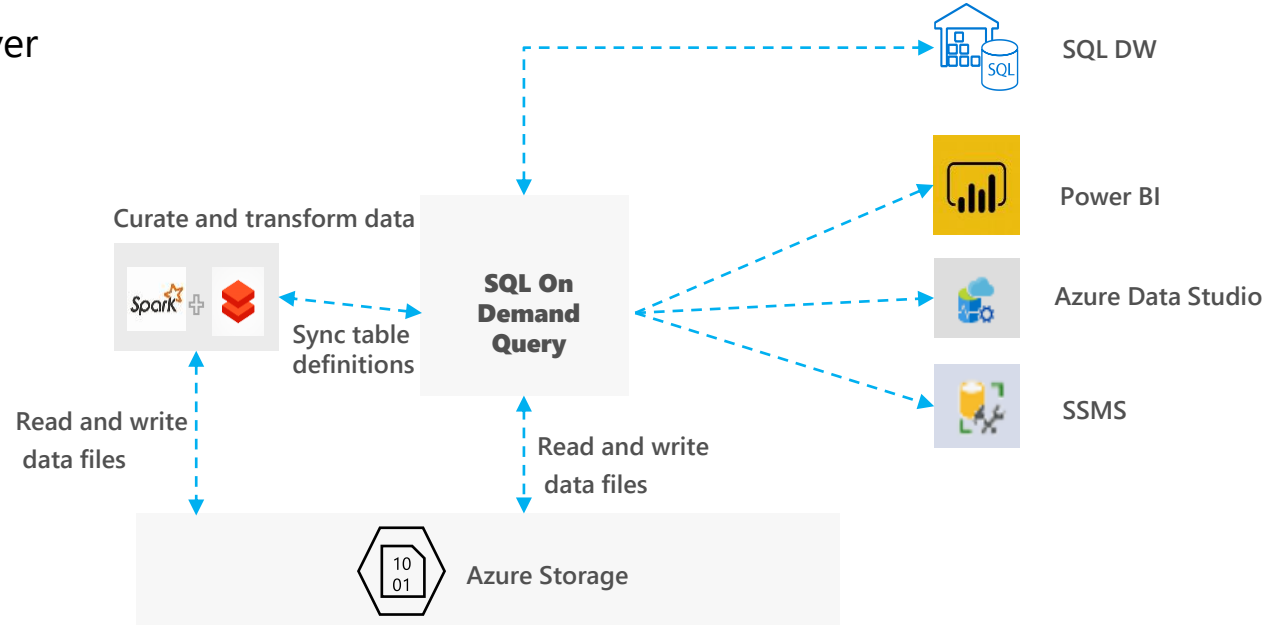
Offers security

Data integration with Databricks, HDInsight

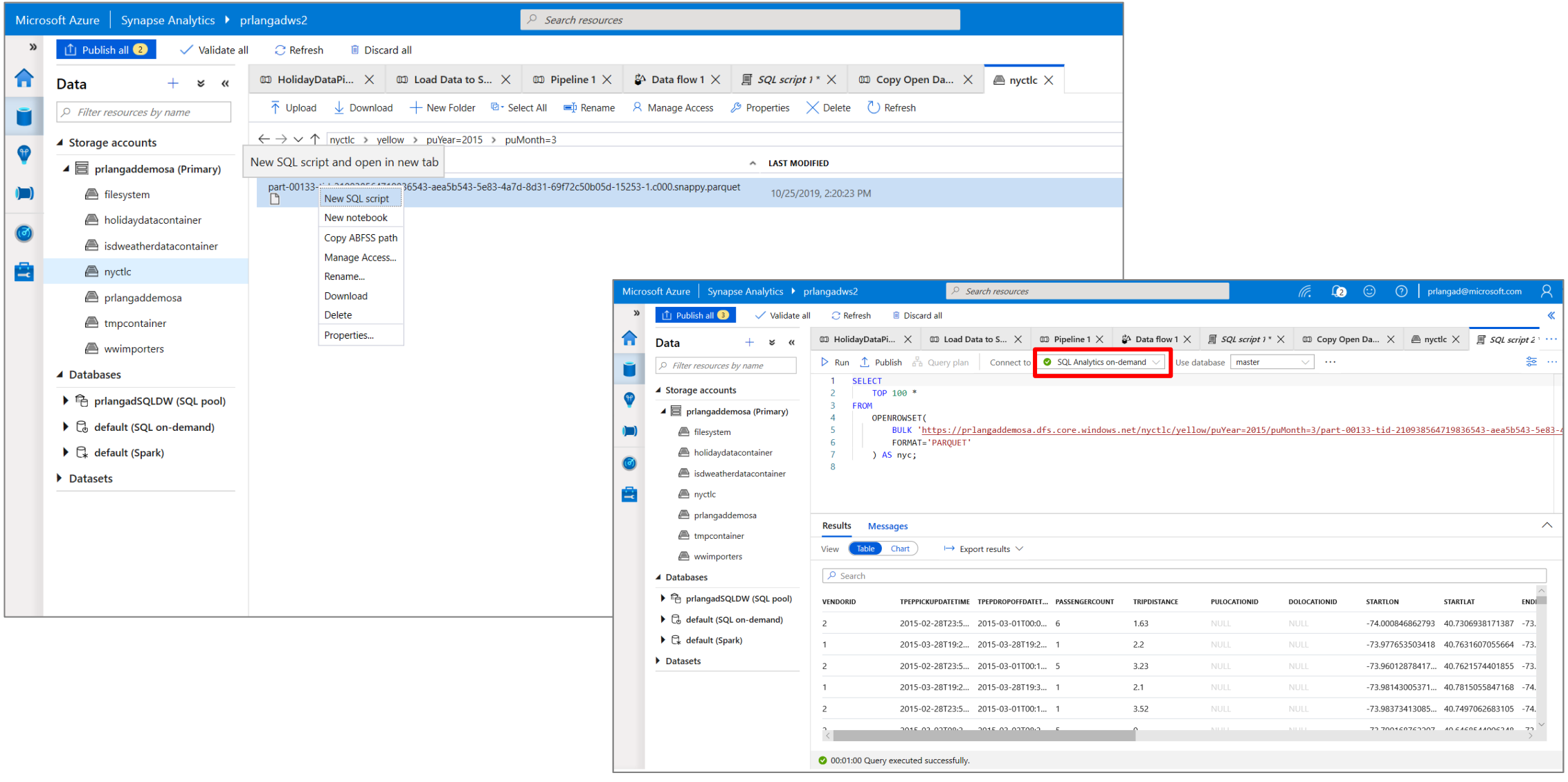
T-SQL syntax to query data

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

Support for BI ecosystem



SQL On Demand – Querying on storage



SQL On Demand – Querying CSV File

Overview

Uses OPENROWSET function to access data

Benefits

Ability to read CSV File with

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

SQL On Demand – Querying CSV File

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Read CSV file - without specifying all columns

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

	countries
1	228

SQL On Demand – Querying folders

Overview

Uses OPENROWSET function to access data from multiple files or folders

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

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

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

SQL On Demand – Querying folders

Read all files from multiple folders

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

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

Read subset of files in folder

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

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

SQL On Demand – Querying specific files

Overview

filename – Provides file name that originates row result

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

Benefits

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

Example of filename function

```
SELECT
    r.filename() AS [filename]
    ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
        BULK 'https://XXX.blob.core.windows.net/csv/taxi/yellow_tripdata_2
017-1*.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

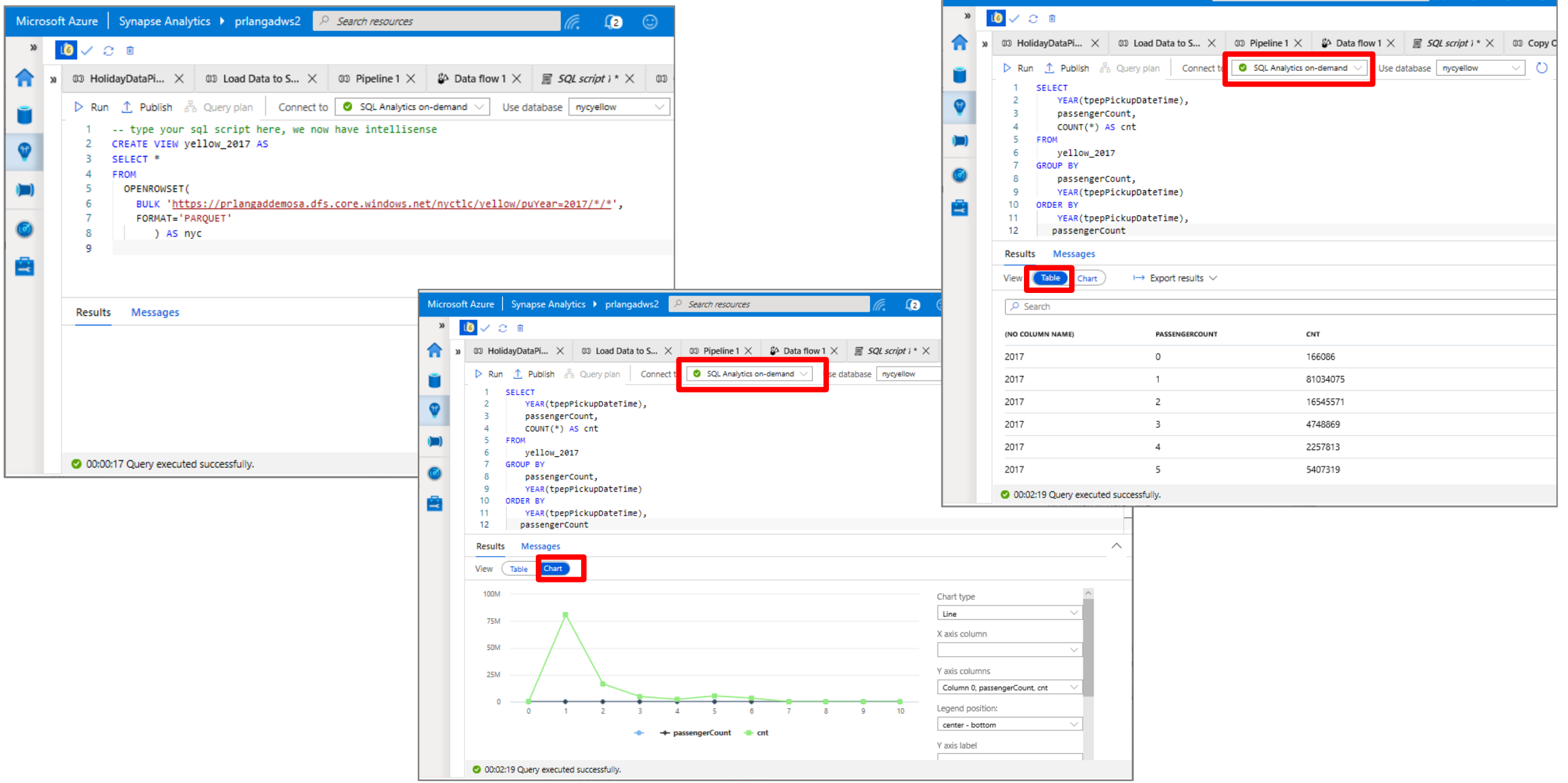
Works same as standard views

```
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/book
1.json',
        FORMAT='CSV',
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    )
    WITH (
        jsonContent varchar(8000)
    ) AS [r]
```

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

SQL On Demand – Querying JSON files

Example of JSON_VALUE function

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

	title	publisher	jsonContent
1	Probabilistic and Statistical Methods in Cryptology, An 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 );
```

Get started today

Learn more

Start building with a free trial

Attend an Analytics in a Day workshop

Azure Synapse Analytics features

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

Top documentation links

- What is SQL on-demand?: link
- What is Apache Spark in Azure Synapse Analytics?: link
- Best practices for SQL pool in Azure Synapse Analytics: link
- Best practices for SQL on-demand in Azure Synapse Analytics: link
- Azure Synapse Analytics shared metadata: link
- Use maintenance schedules to manage service updates and maintenance: link
- Cheat sheet for Azure Synapse Analytics (formerly SQL DW): link
- Best practices for SQL Analytics in Azure Synapse Analytics (formerly SQL DW): link

PASS SQLSATURDAY

Ricordatevi di compilate il feedback form ☺

https://speakerscore.com/9ssc

Thank you

#SqlSat921


PASS