
Microsoft Azure Solutions Business
Welcome to
Azure Synapse Technical Boot Camp
Day 2

A look into Day 2

Optimize & Query	10:00-10:05	Welcome	Main Call
	10:05-10:45	Data Warehouse Optimization	
	10:45-12:00	Lab: Data Warehouse Optimization Part 1	Table Group Call
	12:00-12:30	Activity: Data Warehouse Optimization	
	12:30-12:45	Break	
	12:45-1:15	Data Warehouse Optimization Part 2	Main Call
	1:15-2:00	Lab: Data Warehouse Optimization Part 2	Table Group Call
Security	2:00-3:00	Break	Main Call
	3:00-3:45	Security	
	3:45-4:15	Activity: Security	Table Group Call
	4:15-4:30	Break	
	4:30-5:15	Lab: Security	
	5:15-5:20	Closing	Main Call

Today we will be learning and collaborating across three spaces:

- Main call (this meeting)
- Table Group channel within the event Team
- CloudLabs Learner Portal & Synapse environment

Presentation/ Whole Group

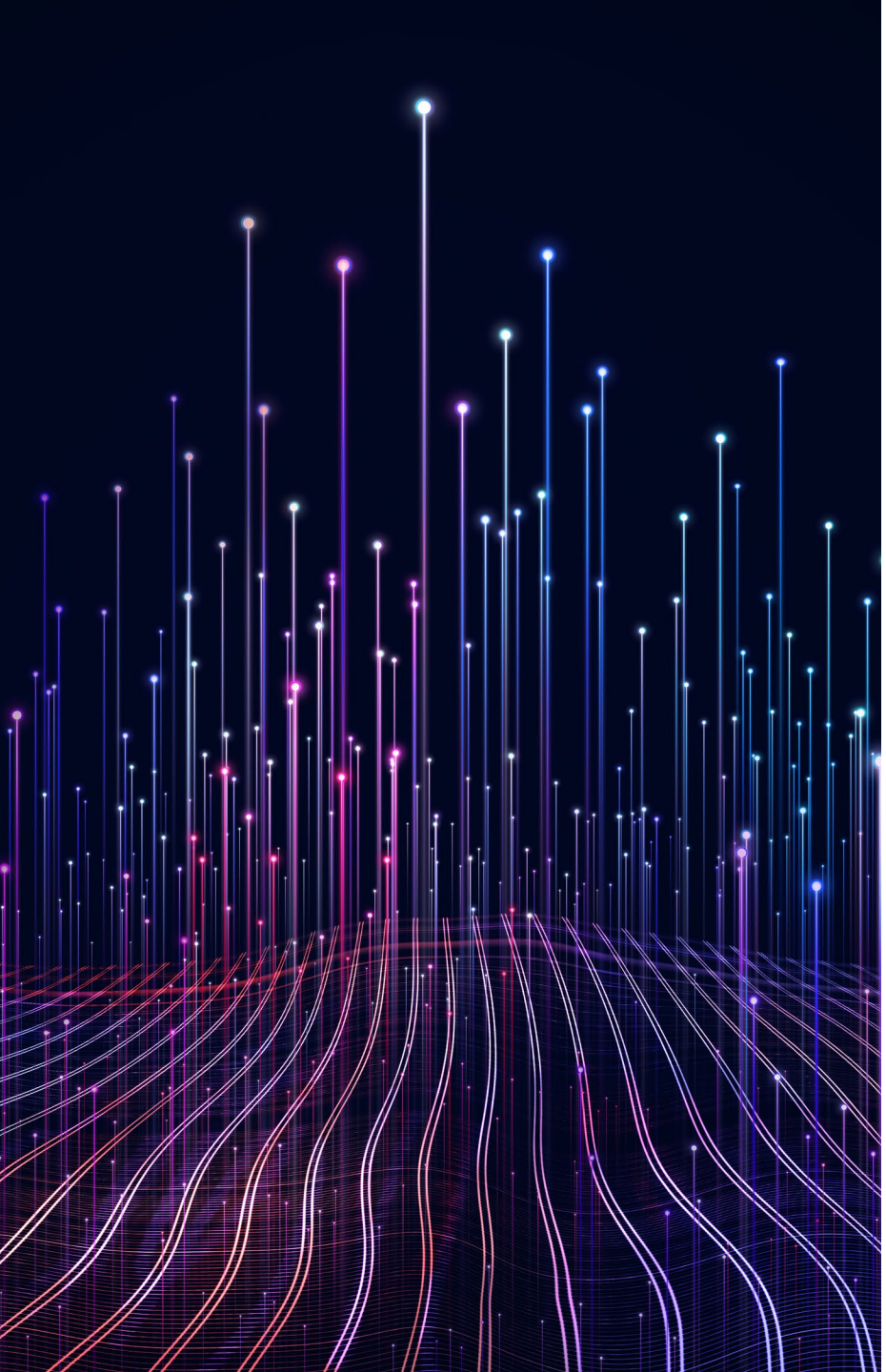
Lab

Activity/ Discussion/ Group Work

Announcements

Microsoft Azure Solutions Business

DW Optimization



Agenda

1 **Table design**

Distributions and partitions.

2 **SQL Capabilities**

Windowing, approximate count, group by and JSON support.

3 **Performance Patterns**

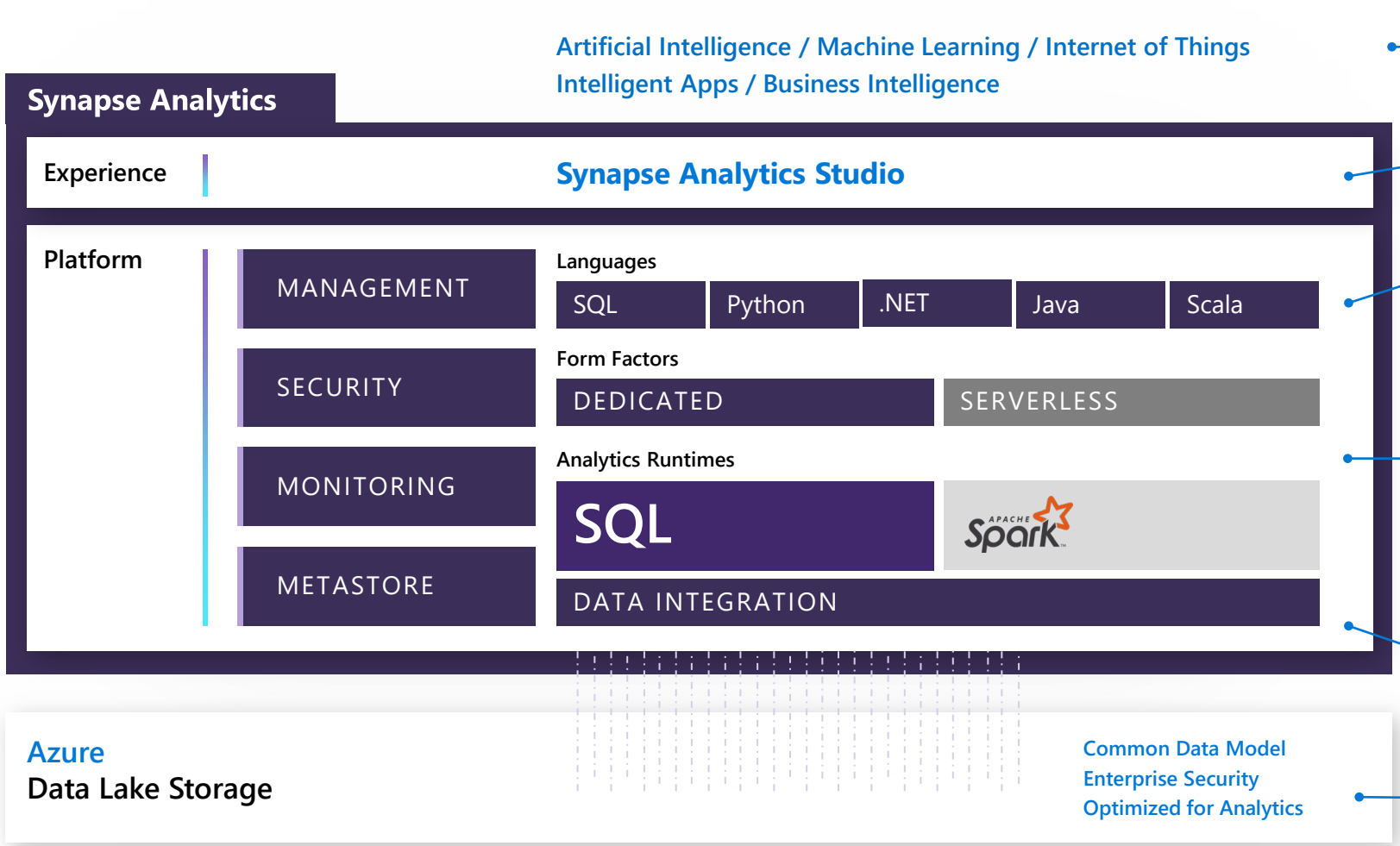
Best practice patterns for performance.

4 **Perf Anti-Patterns**

Counterexamples that negatively impact performance.

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 dedicated and serverless

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

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

Integrated **platform services** for, management, security, monitoring, and meta-store

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

Question...

What is the primary factor for Azure Synapse winning a POC?

What about losing a POC?

Table design

Tables – Distributions

Round-robin distributed

Distributes table rows evenly across all distributions at random.

Hash distributed

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

Replicated

Full copy of table accessible on each Compute node.

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

Tables – Partitions

Overview

Table partitions divide data into smaller groups

In most cases, partitions are created on a date column

Supported on all table types

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

Benefits

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

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

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

Tables – Distributions & Partitions

Logical table structure

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

Physical data distribution

(Hash distribution (OrderId), Date partitions)

Distribution1
(OrderId 80,000 – 100,000)

11-2-2018 partition

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

11-3-2018 partition

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

… x 60 distributions (shards)

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

Common table distribution methods

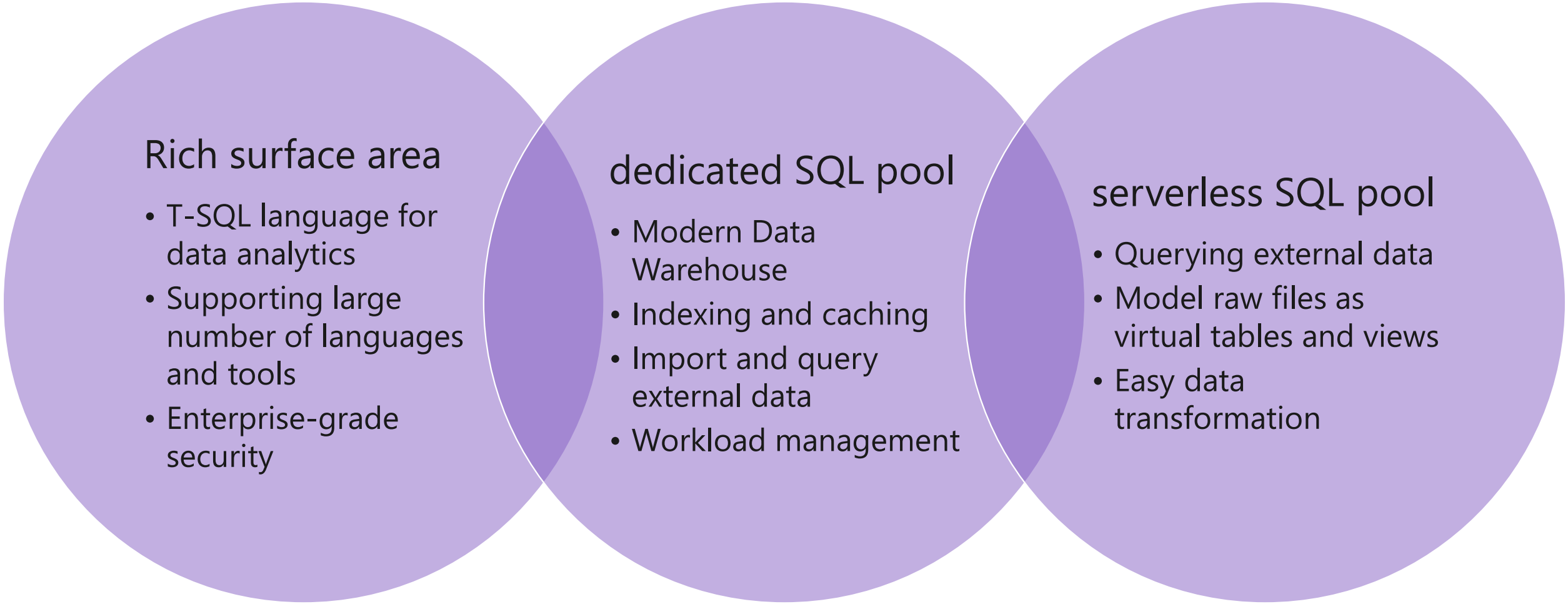
Table Category	Recommended Distribution Option
Fact	Use **hash-distribution with clustered columnstore index**. Performance improves because hashing enables the platform to localize certain operations within the node itself during query execution. Operations that benefit: COUNT(DISTINCT(<hashed_key>)) 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.

Question...

Why is the best practice to load into a staging table, and then CTAS into the production table?

SQL Capabilities

Key features



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
- Score machine learning models in ONNX format

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

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

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

Approximate execution

HyperLogLog accuracy

Reduce query latency in exchange for a small reduction in accuracy.

Returns 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

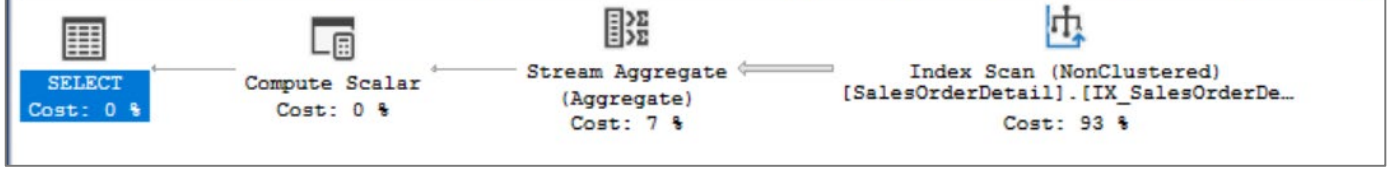
	Approx_Distinct_OrderKey
1	540

```
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

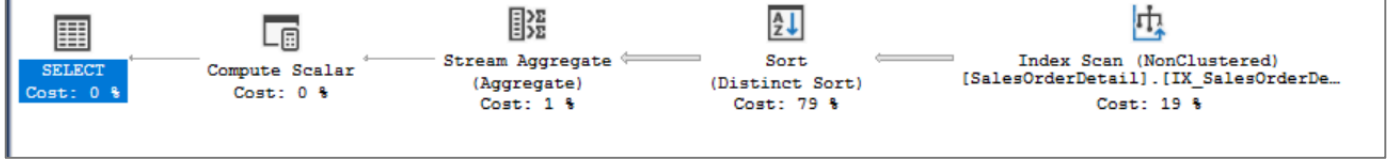
	Distinct_OrderKey
1	542

```
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]
```



Transactions

Overview

Default level is READ UNCOMMITTED

Can switch to READ COMMITTED SNAPSHOT ISOLATION

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

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

Size limits

Limits are applied per distribution

The maximum transaction size depends on the spread of data across distributions

Gen2

DWU	Cap per distribution (GB)	Number of Distributions	MAX transaction size (GB)	# Rows per distribution	Max Rows per transaction
DW100c	1	60	60	4,000,000	240,000,000
DW200c	1.5	60	90	6,000,000	360,000,000
DW300c	2.25	60	135	9,000,000	540,000,000
DW400c	3	60	180	12,000,000	720,000,000
DW500c	3.75	60	225	15,000,000	900,000,000
DW1000c	7.5	60	450	30,000,000	1,800,000,000
DW1500c	11.25	60	675	45,000,000	2,700,000,000

Assumes even distribution of data and average row length of 250 bytes

State management

XACT_STATE() behaves differently than in SQL Server

Uses -2 to mark failed transaction (SQL Server uses -1)

Once in -2 state, no further statements are allowed until roll back (SQL Server still permits reads)

Limitations

No distributed transactions, no nesting, no save points, no DDL

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

No pre-compiled code

Think of SPs as containers for SQL logic

Limitations

Maximum 8 nesting levels (compared to 32 in SQL Server)

No support for @@NESTLEVEL

Cannot consume the result set of a stored procedure with INSERT

No CLR, no encryption, no replication, to table-valued parameters, no execution contexts, no return statement

```
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;
```

Synapse SQL - clients and tools

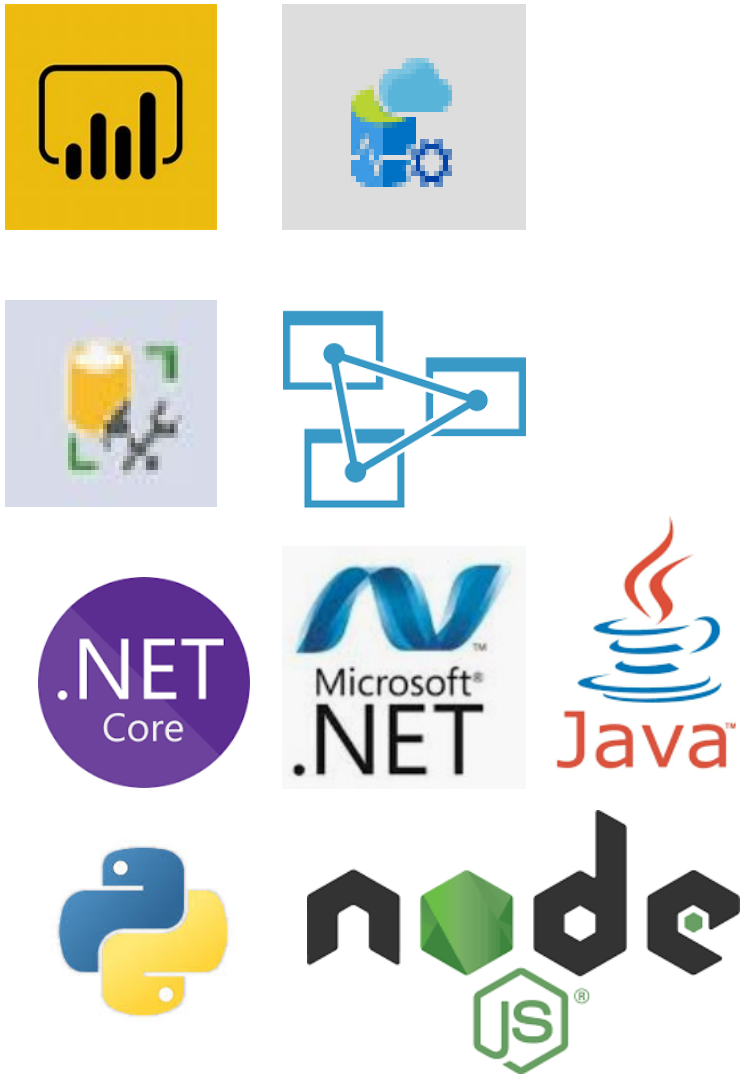
Tools

Web IDE - Synapse Studio

Client tools - Azure Data Studio, SSMS,

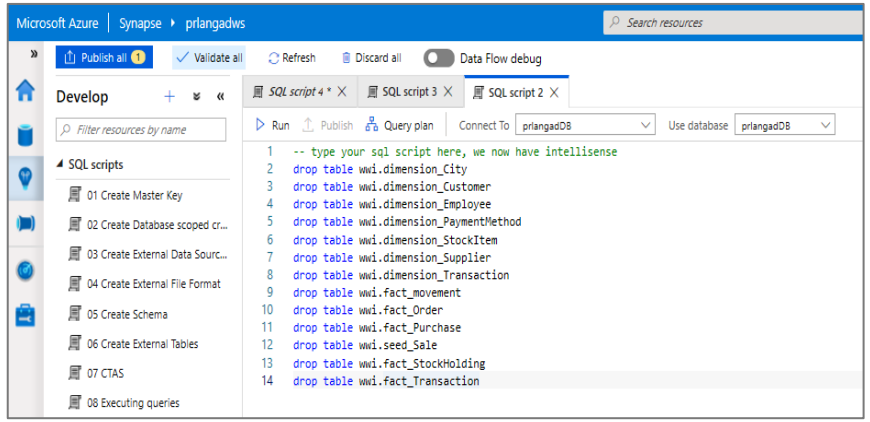
Any tool/library that uses standard SQL can access serverless SQL pool

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

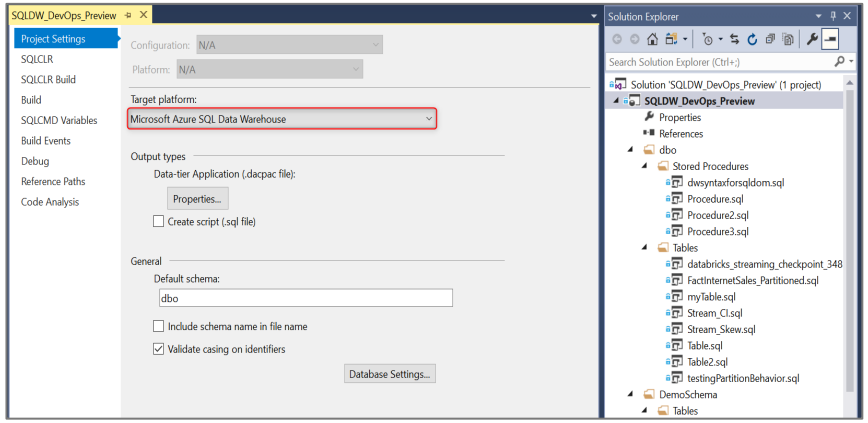


Developer Tools

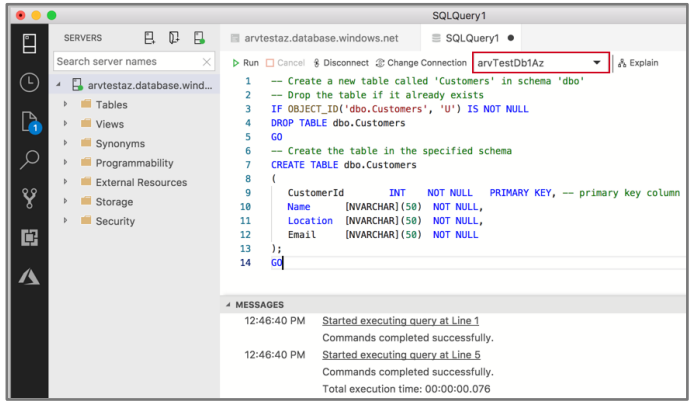
Azure Synapse Analytics



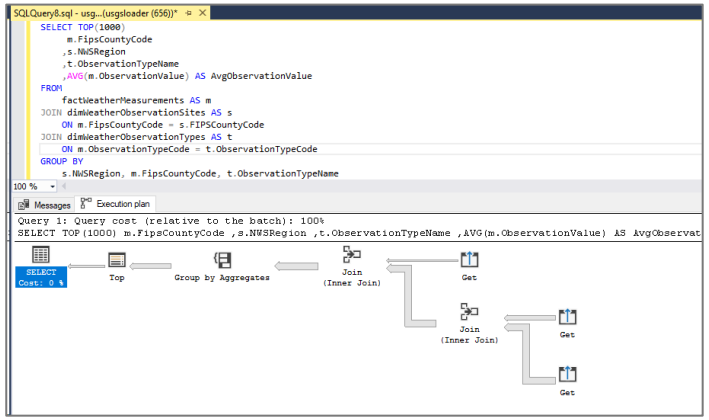
Visual Studio - SSDT database projects



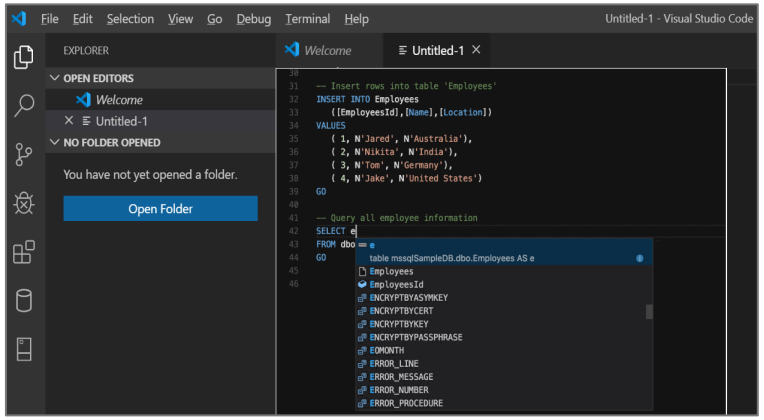
Azure Data Studio (queries, extensions etc.)



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



Visual Studio Code



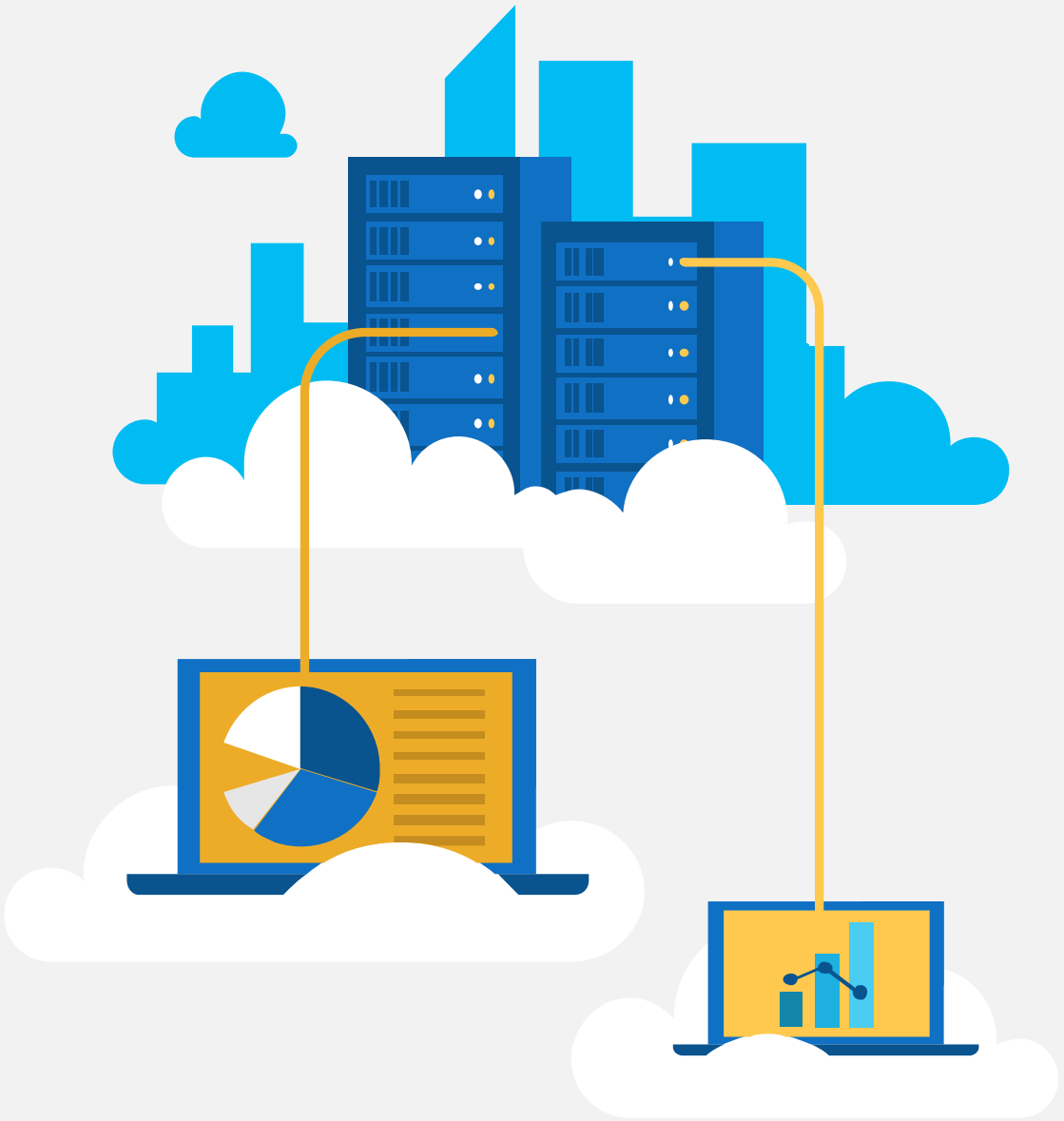
Pop Quiz 1

What reduction in accuracy does APPROXIMATE_COUNT_DISTINCT (HyperLogLog) show on average?

A) 2%

B) 0.1%

C) 0.5%



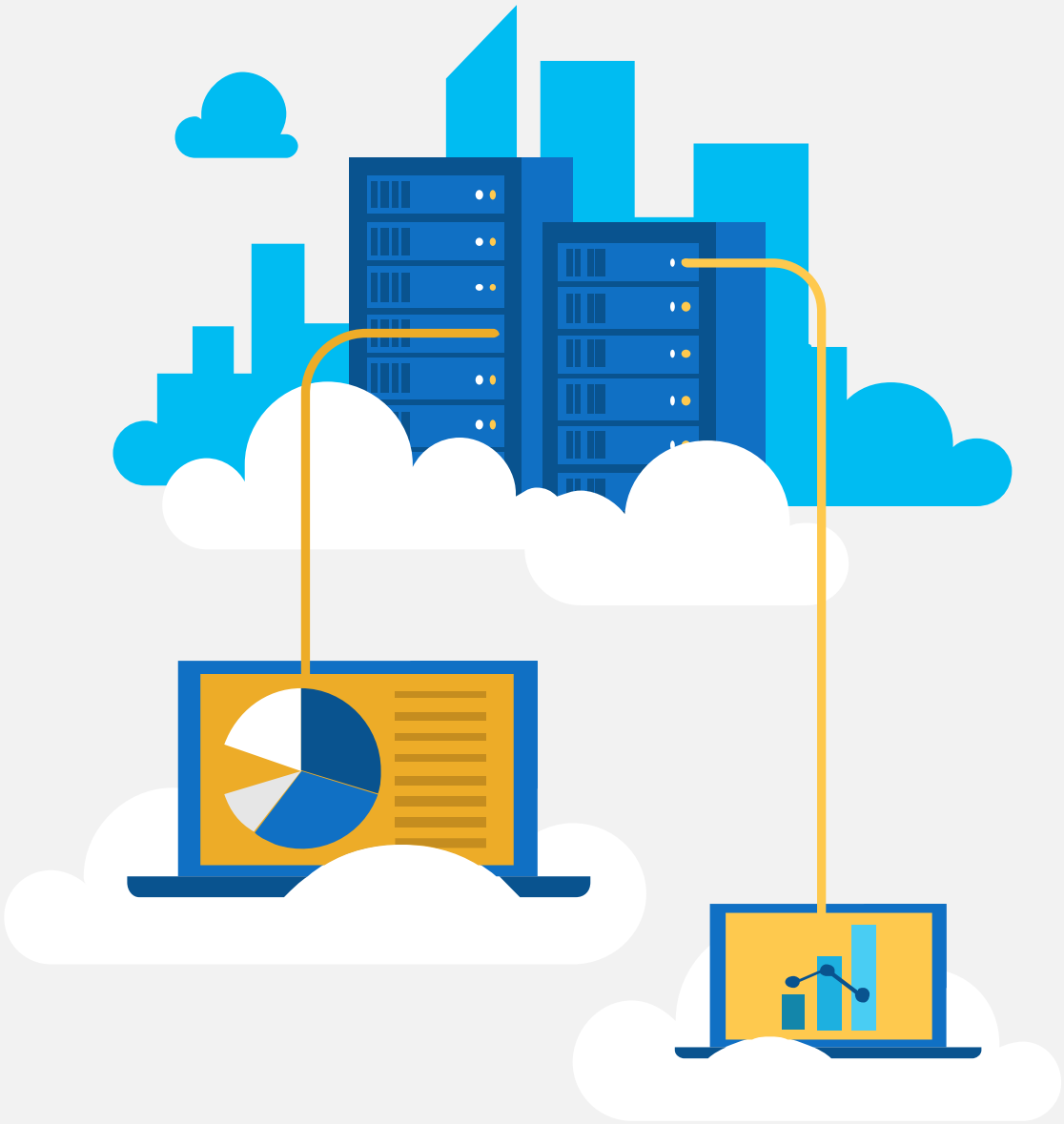
Pop Quiz 1

What reduction in accuracy does APPROXIMATE_COUNT_DISTINCT (HyperLogLog) show on average?

A)
2%

B)
0.1%

C)
0.5%



SQL Recap

- Query JSON using Azure Synapse SQL in conjunction with T-SQL OPENJSON, JSON_VALUE, and JSON_Query statements.
- Modify JSON data during UPDATE using the JSON_MODIFY statement.
- Use APPROXIMATE_COUNT_DISTINCT for better count query performance- results within average 2% accuracy of the true count.

Performance Patterns

Distributed table design recommendations

Hash Distribution:

- Large fact tables exceeding several GBs with frequent inserts should use a hash distribution.

Round Robin Distribution:

- Potentially useful tables created from raw input.
- Temporary staging tables used in data preparation.

Replicated Tables:

- Lookup tables that range in size from 100’s MBs to 1.5 GBs should be replicated. Works best when table size is less than 2 GB compressed.

Performance Anti-Patterns

Too many partitions

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

Views on Views

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

- Views cannot be distributed

Microsoft Azure Solutions Business

Thank you

⌛ Break Time!

We are at break for the next 15 minutes.

Please feel free to step away and we will see you back in your **Table Group call** when break ends.

7:00-7:05	Welcome
7:05-7:45	Data Warehouse Optimization
7:45-8:00	Break
8:00-9:00	Lab: Data Warehouse Optimization Part 1
9:00-9:30	Activity: Data Warehouse Optimization
9:30-10:00	Data Warehouse Optimization Part 2
10:00-10:15	Break
10:15-11:00	Lab: Data Warehouse Optimization Part 2
11:00-12:00	Break
12:00-12:30	Security
12:30-13:25	Activity: Security
13:25-14:15	Lab: Security
14:15-14:20	Closing

If at anytime you require assistance, please send a message to the "Need help – Ask Here" channel in the Microsoft Teams site for this event

Build Hands-on Lab:
Data Warehouse Optimization Part 1

OUTCOMES

As a result of participating in this lab, you will be able to **better optimize Data Warehouse query performance within deployed Azure Synapse Analytics solutions**.

ACTIVITY

60 minutes | Independent | Table Group Channel & CloudLabs Environment

Independently review and complete Exercises 1-2 in the Lab 3 Guide.

Login to your **CloudLabs environment** and work through a set of tasks you would typically follow in work associated with optimizing your customer's data after you have completed your data ingestion.

The Lab Guide is available in the 'Files' tab of the General channel. Engage your Table Group call for support and troubleshooting.

Welcome	Data Warehouse Optimization Part 1	Break	**Lab: Data Warehouse Optimization**	Activity: Data Warehouse Optimization	Data Warehouse Optimization Part 2	Break	Lab: Data Warehouse Optimization Part 2	Break	Security	Activity: Security	Lab: Security	Closing

Breakout Activity:
Data Warehouse Optimization

OUTCOMES

As a result of participating in this activity, you will better be able to **implement optimization strategizes for the data warehouse using SQL based approached in Azure Synapse Analytics.**

ACTIVITY

30 minutes

Learning Adviser

Table Group Channel

As a team review the challenges posed and decide how to address them.

Choose a "driver" who will open their **CloudLabs** environment and then complete the challenges as a team using the driver's Azure Synapse Analytics Studio.

The challenges cover table and index configuration and query performance optimization. Team results are collected in the event leaderboard.

Welcome	Data Warehouse Optimization Part 1	Break	Lab: Data Warehouse Optimization	**Activity: Data Warehouse Optimization**	Data Warehouse Optimization Part 2	Break	Lab: Data Warehouse Optimization Part 2	Break	Security	Activity: Security	Lab: Security	Closing

Welcome Back!

7:00-7:05	Welcome	
7:05-7:45	Data Warehouse Optimization	Main Call
7:45-8:00	Break	
8:00-9:00	Lab: Data Warehouse Optimization Part 1	
9:00-9:30	Activity: Data Warehouse Optimization	Table Group Call
9:30-10:00	Data Warehouse Optimization Part 2	
10:00-10:15	Break	Main Call
10:15-11:00	Lab: Data Warehouse Optimization Part 2	Table Group Call
11:00-12:00	Break	
12:00-12:30	Security	Main Call
12:30-13:25	Activity: Security	
13:25-14:15	Lab: Security	Table Group Call
14:15-14:20	Closing	Main Call

- Presentation/ Whole Group
- Lab
- Activity/ Discussion/ Group Work
- Announcements

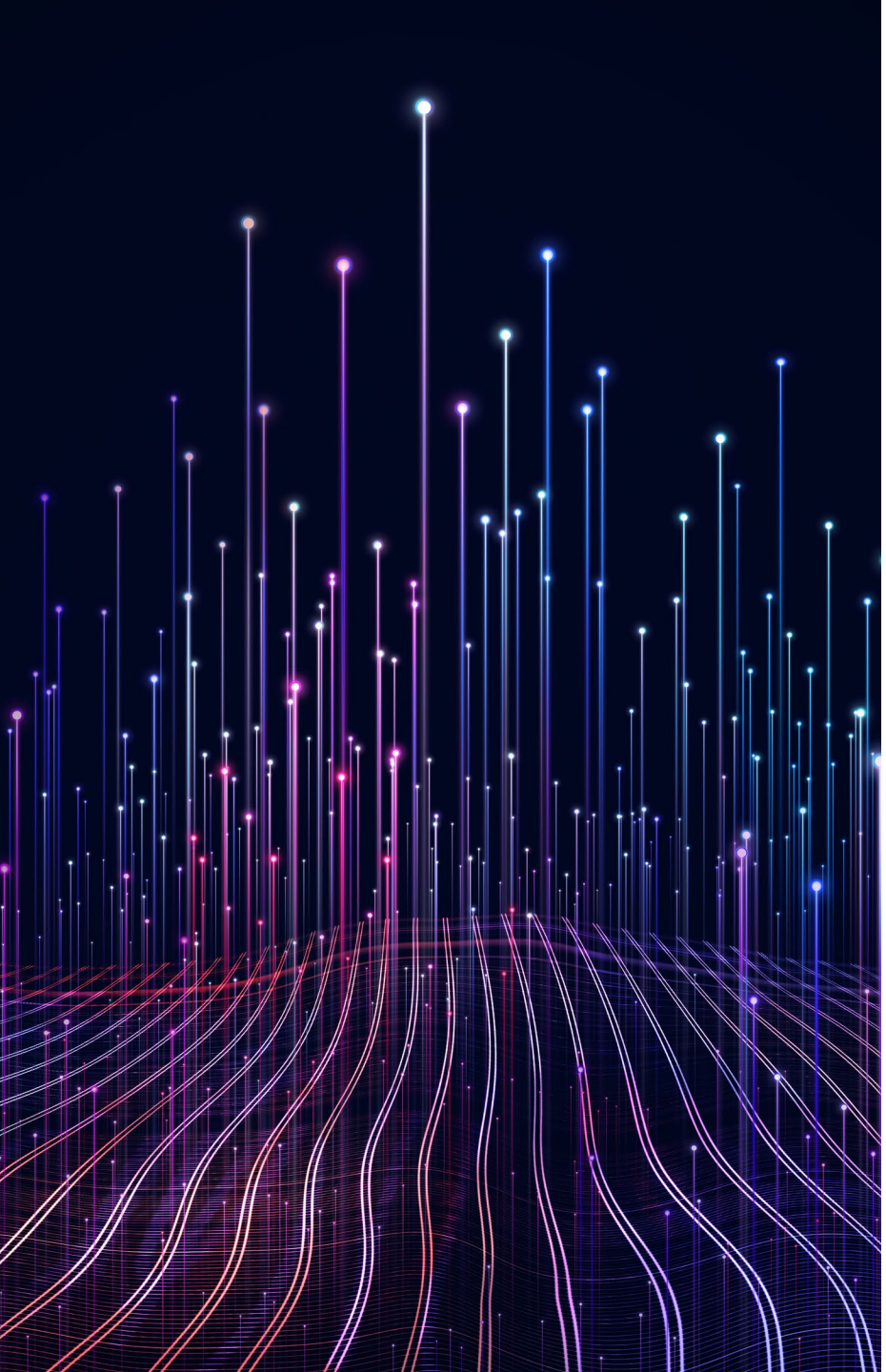
Microsoft Azure Solutions Business

DW Optimization Part 2

Question...

How many indexing methods are there for a Synapse table?

Can you name them all?



Agenda

1 **Performance Patterns**

Result set caching and materialized views.

2 **Index design**

Clustered columnstore index and ordered variant, clustered index, heap and non-clustered index.

3 **Perf Anti-Patterns**

Approaches the impede or limit performance.

Performance Patterns

Result-set caching

Overview

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

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

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

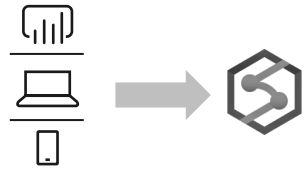
Result cache is evicted regularly based on a time-aware least recently used algorithm (TLRU).

Benefits

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

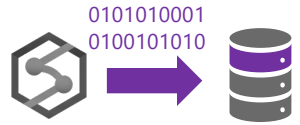
```
-- Turn on/off result-set caching for a database
-- Must be run on the MASTER database
ALTER DATABASE {database_name}
SET RESULT_SET_CACHING { ON | OFF }

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

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

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

Result-set caching flow

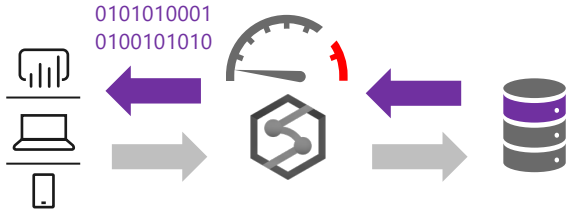


1. Client sends query to SQL pool

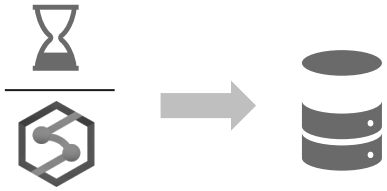
2. Query is processed using compute nodes which pull data from remote storage, process query and output back to client app



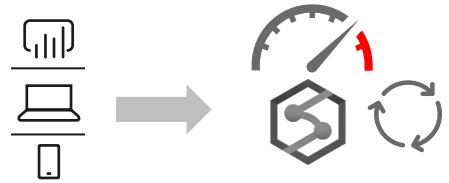
Query results are cached in remote storage so subsequent requests can be served immediately



3. Subsequent executions for the same query bypass compute nodes and can be fetched instantly from persistent cache in remote storage



4. Remote storage cache is evicted regularly based on time, cache usage, and any modifications to underlying table data.



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

Materialized views

Overview

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

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

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

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

Benefits

- Automatic and synchronous data refresh with data changes in base tables. No user action is required.
- High availability and resiliency as regular tables

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

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

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

Indexed Materialized Views

- Indexed views cache the schema and data for a view in DW remote storage. They are useful for improving the performance of 'SELECT' statement queries that include aggregations
- Indexed views are automatically updated when data in underlying tables are changed. This is a synchronous operation that occurs as soon as the data is changed.
- The auto caching functionality allows Synapse Query Optimizer to consider using indexed view even if the view is not referenced in the query
- Supported aggregations: MAX, MIN, AVG, COUNT, COUNT_BIG, SUM, VAR, STDEV

Materialized views - example

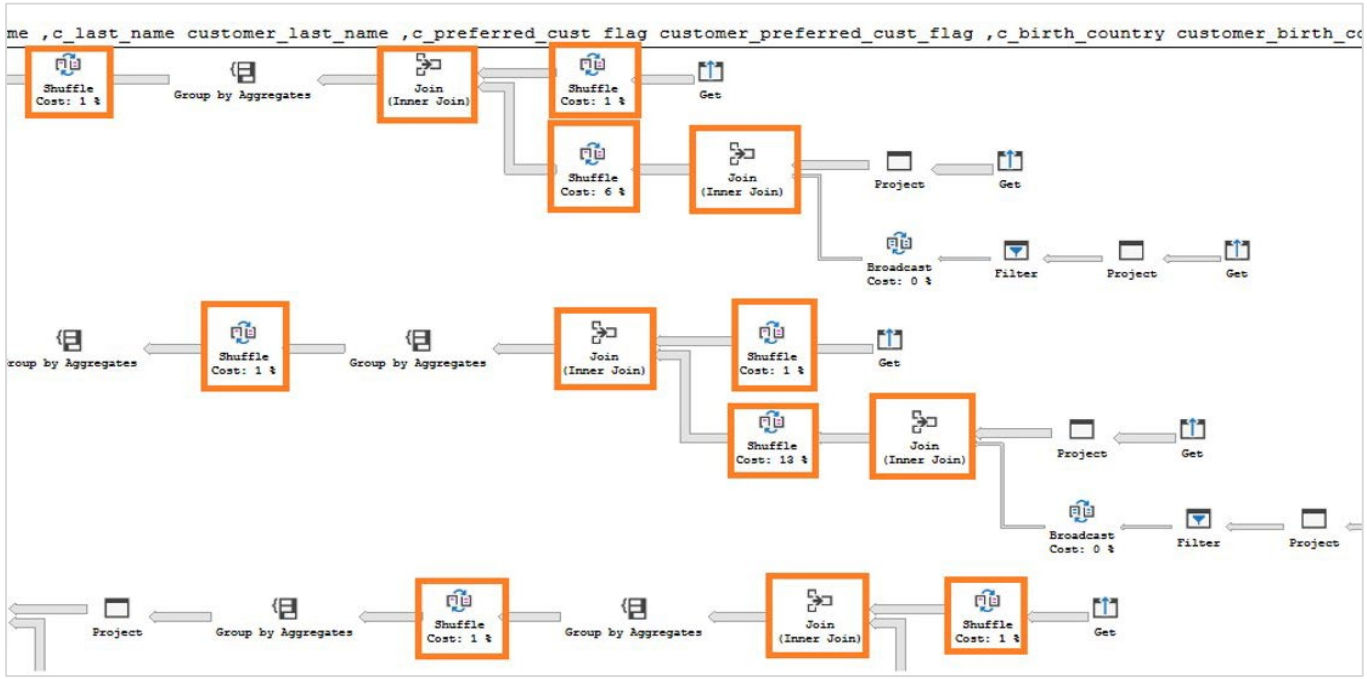
In this example, a query to get the year total sales per customer is shown to have a lot of data shuffles and joins that contribute to slow performance:

No relevant indexed views created on the data warehouse

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

Execution time: 103 seconds

Lots of data shuffles and joins needed to complete query



Materialized views - example

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

Original query – get year total sales per customer

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

Create indexed view with hash distribution on customer_id column

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

Indexed (materialized) views - example

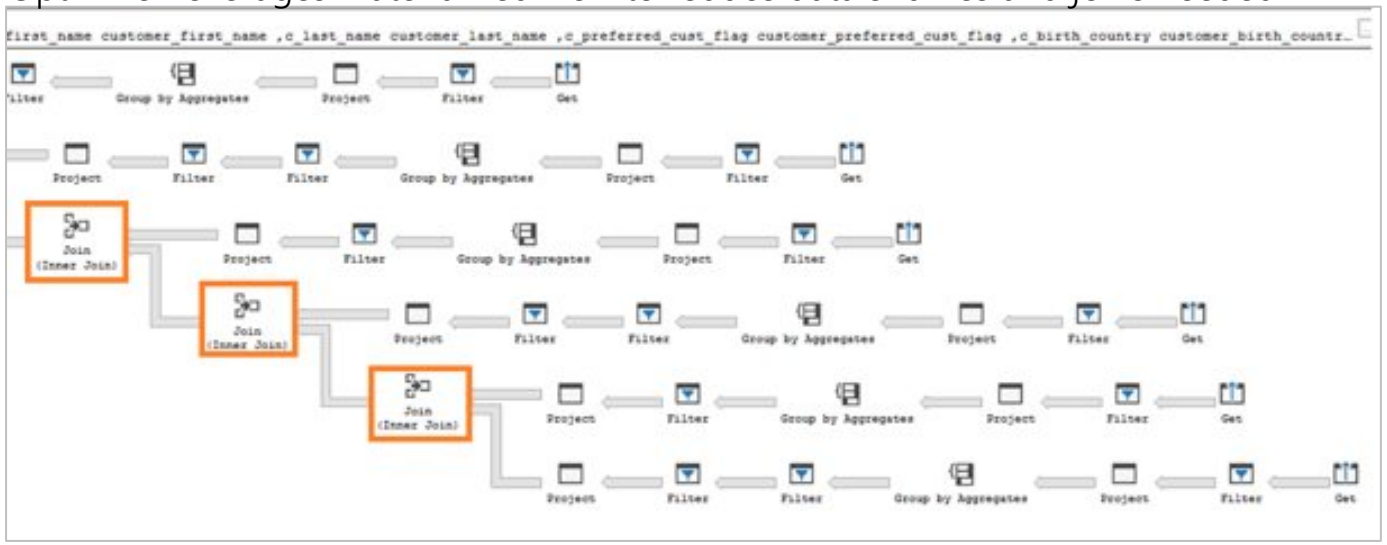
SQL pool query optimizer automatically leverages the indexed view to speed up the same query. Notice that the query does not need to reference the view directly

Original query – no changes have been made to query

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

Execution time: 6 seconds

Optimizer leverages materialized view to reduce data shuffles and joins needed



Materialized views- Recommendations

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

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

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

Index design

Tables – Indexes

Clustered Columnstore index (Default Primary)

Highest level of data compression

Best overall query performance

Clustered index (Primary)

Performant for looking up a single to few rows

Heap (Primary)

Faster loading and landing temporary data

Best for small lookup tables

Nonclustered indexes (Secondary)

Enable ordering of multiple columns in a table

Allows multiple nonclustered on a single table

Can be created on any of the above primary indexes

More performant lookup queries

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

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

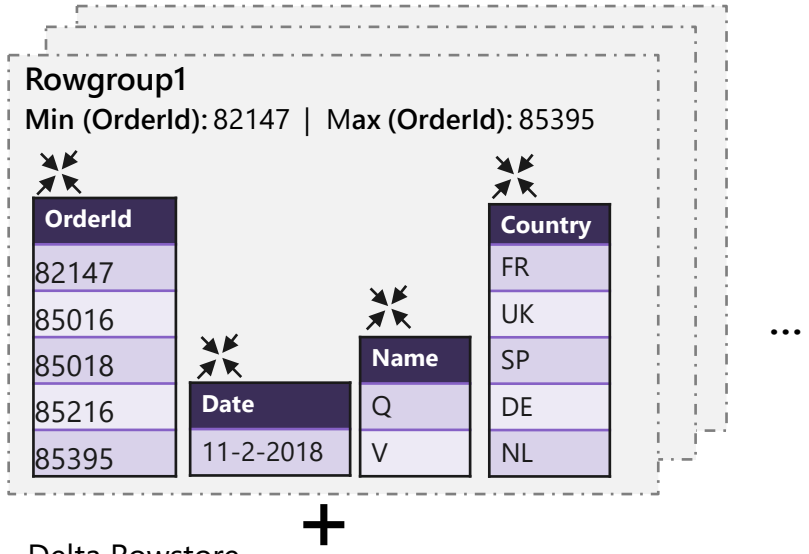
SQL Analytics Columnstore Tables

Logical table structure

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

Clustered columnstore index

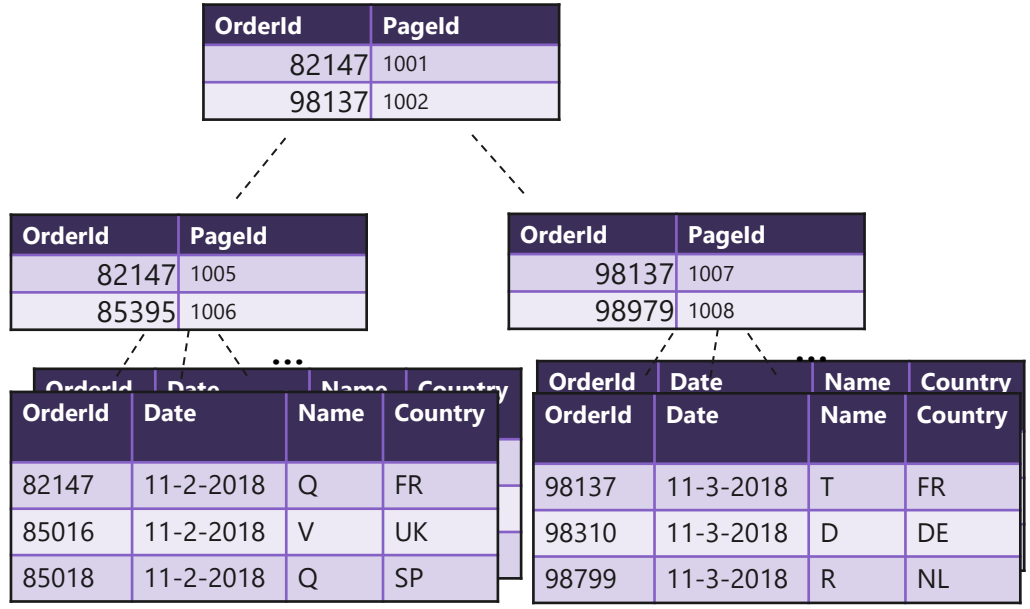
(OrderId)



- Data stored in compressed columnstore segments after being sliced into groups of rows (rowgroups/micro-partitions) for maximum compression
- Rows are stored in the delta rowstore until the number of rows is large enough to be compressed into a columnstore

Clustered/Non-clustered rowstore index

(OrderId)



- Data is stored in a B-tree index structure for performant lookup queries for particular rows.
- Clustered rowstore index: The leaf nodes in the structure store the data values in a row (as pictured above)
- Non-clustered (secondary) rowstore index: The leaf nodes store pointers to the data values, not the values themselves

Ordered Clustered Columnstore Indexes

The problem with CCI

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

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

The solution

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

```
-- Create Table with Ordered Columnstore Index
CREATE TABLE sortedOrderTable
(
   OrderId  INT NOT NULL,
   Date     DATE NOT NULL,
   Name     VARCHAR(2),
   Country  VARCHAR(2)
)
WITH
(
 CLUSTERED COLUMNSTORE INDEX ORDER (OrderId)
)
```

```
-- Create Clustered Columnstore Index on existing table
CREATE CLUSTERED COLUMNSTORE INDEX cciOrderId
ON dbo.OrderTable ORDER (OrderId)

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

Ordered CCI

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

- Columnstore Segments are automatically updated as data is inserted, updated, or deleted in data warehouse tables.

- New data resulting from the same batch of DML or data loading operations is sorted within that batch (no global sorting across all data in the table).

- REBUILD can be used to sort all data in the table (which is an offline operation, done one partition at a time).

Ordered CCI - Performance

The number of overlapping segments depends on:

- The size of data to sort
- Available memory (use XLARGERC on a higher DWU to allow more memory for data sorting before compressing occurs)
- MAXDOP (create ordered CCI with MAXDOP =1 as each thread used for ordered CCI works on a subset of data and sorts it locally)

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

Choosing the right index

- Clustered Columnstore indexes (CCI) are best for fact tables.
- CCI offer the highest level of data compression and best query performance for tables with over 100 million rows.
- Heap tables are best for small lookup tables and recommended for tables with less than 100 million rows.
- Clustered Indexes may outperform CCI when very few rows need to be retrieved quickly.
  - Add non-clustered indexes to improve performance for less selective queries.
  - Each additional index added to a table increases storage space required and processing time during data loads.
- Speed load performance by staging data in heap tables and temporary tables prior to running transformations.

Performance Anti-Patterns

Too many indexes

- Start without indexes. The overhead of maintaining them can be greater than their value.

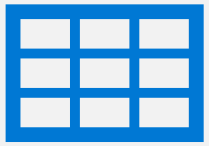
- A primary-key non-clustered index may improve performance of joins when fact tables are joined to very large (billion+) dimensions

Pop Quiz 1

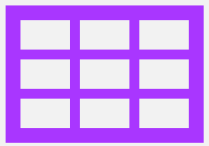
Match the tables with their recommended index!

dbo.dimProduct
1.5k rows
Product information

dbo.Sales
150M rows
Single sales lookups



dbo.LineItem
30B rows
Primary fact table



stg.stagingLineItem
10k rows
Staging table for loads

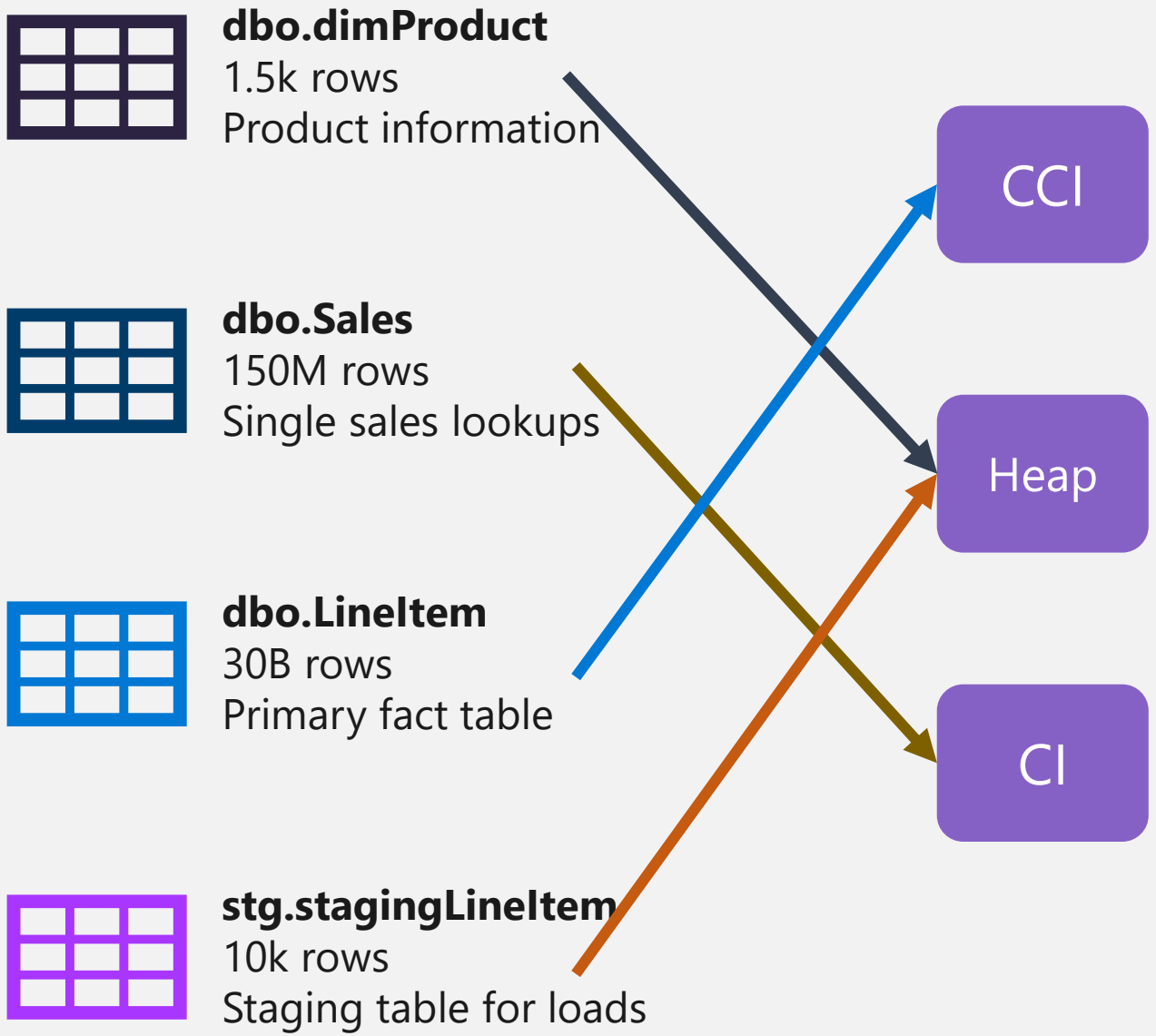
CCI

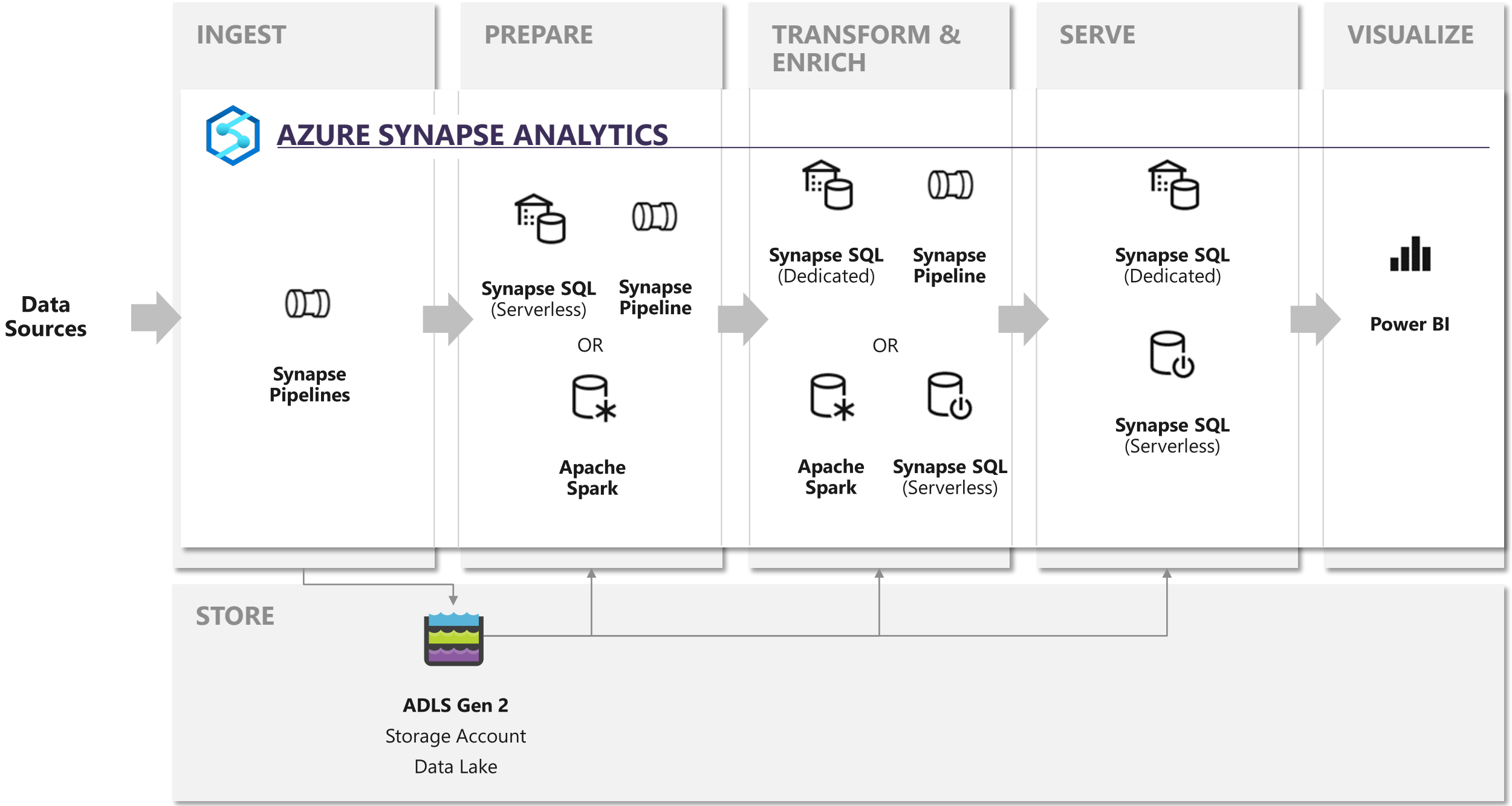
Heap

CI

Pop Quiz 1

Match the tables with their recommended index!




INGEST
PREPARE
TRANSFORM & ENRICH
SERVE
VISUALIZE
AZURE SYNAPSE ANALYTICS
Data Sources
Synapse Pipelines
Synapse SQL (Serverless)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Dedicated)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Serverless)
Synapse SQL (Dedicated)
Synapse SQL (Serverless)
Power BI
STORE
ADLS Gen 2
Storage Account
Data Lake

Microsoft Azure Solutions Business

Thank you

⌛ Break Time!

We are at break for the next 15 minutes.

Please feel free to step away and we will see you back in your **Table Group call** when break ends.

7:00-7:05	Welcome
7:05-7:45	Data Warehouse Optimization
7:45-8:00	Break
8:00-9:00	Lab: Data Warehouse Optimization Part 1
9:00-9:30	Activity: Data Warehouse Optimization
9:30-10:00	Data Warehouse Optimization Part 2
10:00-10:15	Break
10:15-11:00	Lab: Data Warehouse Optimization Part 2
11:00-12:00	Break
12:00-12:30	Security
12:30-13:25	Activity: Security
13:25-14:15	Lab: Security
14:15-14:20	Closing

If at anytime you require assistance, please send a message to the "Need help – Ask Here" channel in the Microsoft Teams site for this event

Build Hands-on Lab:
Data Warehouse Optimization Part 2

OUTCOMES

As a result of participating in this lab, you will be better able to **optimize the data warehouse table configuration and understand the impact on query performance**.

ACTIVITY

45 minutes | Independent | Table Group Channel & CloudLabs Environment

Independently review and complete Exercises 1-5 in the Lab 4 Guide.

Login to your **CloudLabs environment** and work through a set of tasks you would typically follow in work associated with optimizing your customer's data after you have completed your data ingestion.

The Lab Guide is available in the 'Files' tab of the General channel. Engage your Table Group call for support and troubleshooting.

⌛ Break Time!

We are at break for the next 60 minutes.

Please feel free to step away and we will see you back in your **Main call** when break ends.

7:00-7:05	Welcome
7:05-7:45	Data Warehouse Optimization
7:45-8:00	Break
8:00-9:00	Lab: Data Warehouse Optimization Part 1
9:00-9:30	Activity: Data Warehouse Optimization
9:30-10:00	Data Warehouse Optimization Part 2
10:00-10:15	Break
10:15-11:00	Lab: Data Warehouse Optimization Part 2
11:00-12:00	Break
12:00-12:30	Security
12:30-13:25	Activity: Security
13:25-14:15	Lab: Security
14:15-14:20	Closing

If at anytime you require assistance, please send a message to the "Need help – Ask Here" channel in the Microsoft Teams site for this event

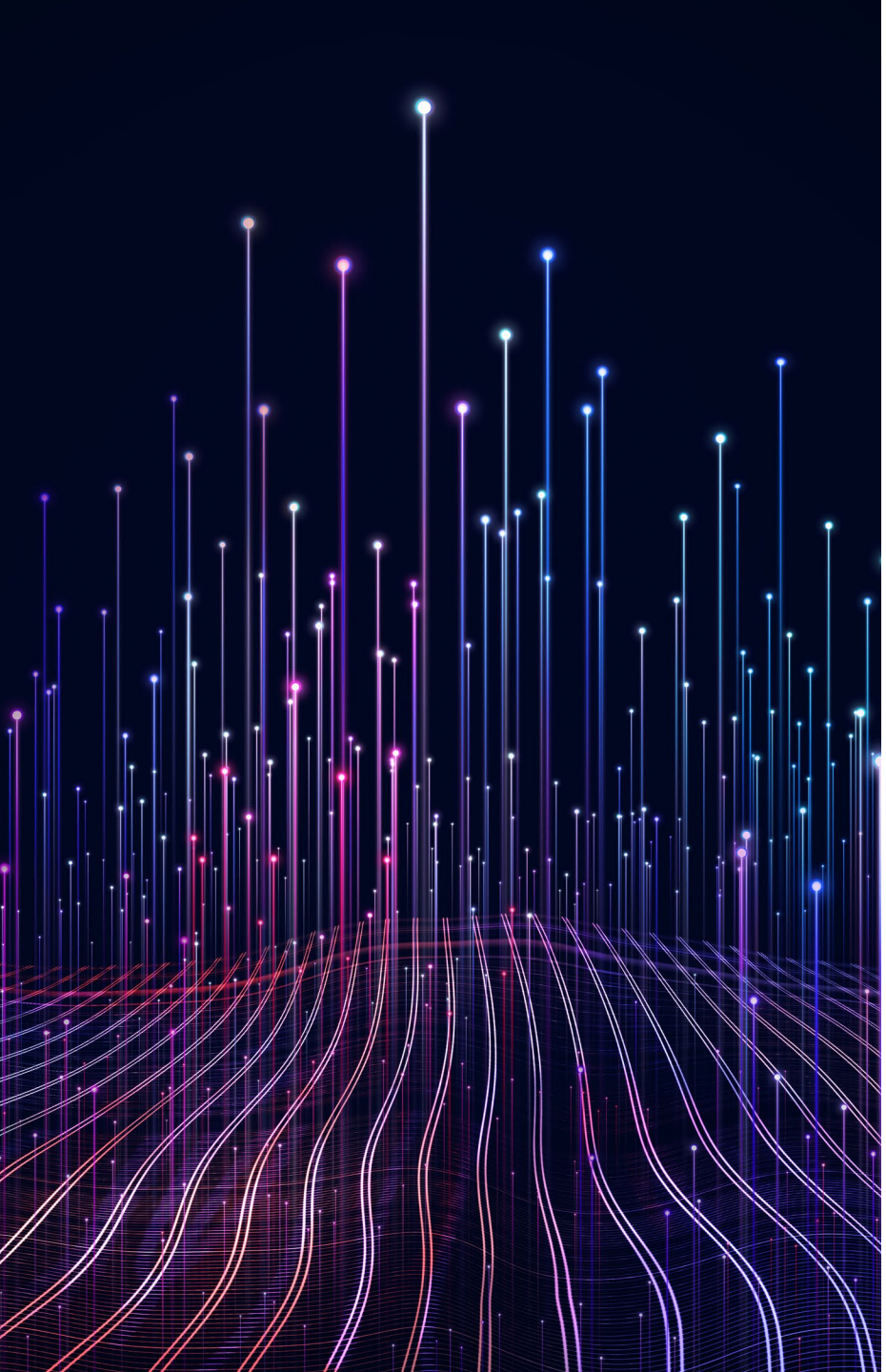
Welcome Back!

7:00-7:05	Welcome	
7:05-7:45	Data Warehouse Optimization	Main Call
7:45-8:00	Break	
8:00-9:00	Lab: Data Warehouse Optimization Part 1	Table Group Call
9:00-9:30	Activity: Data Warehouse Optimization	
9:30-10:00	Data Warehouse Optimization Part 2	Main Call
10:00-10:15	Break	
10:15-11:00	Lab: Data Warehouse Optimization Part 2	Table Group Call
11:00-12:00	Break	Main Call
12:00-12:30	Security	
12:30-13:25	Activity: Security	Table Group Call
13:25-14:15	Lab: Security	
14:15-14:20	Closing	Main Call

- Presentation/ Whole Group
- Lab
- Activity/ Discussion/ Group Work
- Announcements

Microsoft Azure Solutions Business

Security in Azure Synapse



Agenda

1 **Security baseline**

Synapse Analytics security baseline

2 **Network and data security**

Synapse Managed Virtual Network and Managed private endpoints

Securing data

Security baseline for Synapse Analytics

The Security Baseline for Synapse Analytics

Contains recommendations that will help you improve the security posture of your deployment	
Network security	Protect with VNETs Monitor configuration and traffic of VNETs, subnets and network interfaces Use Advanced Threat Protection (ATP) and DDoS Protection, Record network packets, Use tags for network security groups
Logging and monitoring	Update time synchronization for your compute deployments, Central security log management, Enable auditing on the Azure SQL server-level, Configure security log storage retention, Use advanced Threat Protection (ATP) for Azure SQL Database in conjunction with Azure Security Center to monitor and alert on anomalous activity
Identity and access control	Maintain an inventory of administrative accounts Change default passwords where applicable Use dedicated administrative accounts, Use AAD single sign-on Use multi-factor authentication for all AAD-based access Use a Privileged Access Workstation (PAW) with Multi-Factor Authentication (MFA) configured to log into and configure Azure resources Use Azure Active Directory security reports for generation of logs and alerts when suspicious or unsafe activity occurs in the environment. Manage Azure resources from only approved locations Create an Azure Active Directory (AD) administrator for the Azure SQL Database server in your Synapse SQL pool. Regularly review and reconcile user access In support scenarios where Microsoft needs to access data related to the Azure SQL Database in your Synapse SQL pool, Azure Customer Lockbox provides an interface for you to review and approve or reject data access requests Use Azure Active Directory (Azure AD) Identity Protection and risk detection features to configure automated responses to detected suspicious actions related to user identities. Configure Azure Active Directory (AD) authentication with Azure SQL and enable diagnostic settings for Azure Active Directory user accounts, sending the audit logs and sign-in logs to a Log Analytics workspace. Configure desired alerts within Log Analytics

Security Baseline for Synapse Analytics

Contains recommendations that will help you improve the security posture of your deployment	
Data protection	Maintain an inventory of sensitive Information; Isolate systems storing or processing sensitive information; For any Azure SQL Database in your Synapse SQL pool storing or processing sensitive information, mark the database and related resources as sensitive using tags. Configure Private Link in conjunction with network security group (NSG) service tags on your Azure SQL Database instances to prevent the exfiltration of sensitive information. Encrypt all sensitive information in transit. Use the Azure Synapse SQL Data Discovery & Classification feature. Data Discovery & Classification provides advanced capabilities built into Azure SQL Database for discovering, classifying, labeling & protecting the sensitive data in your databases. Use RBAC to manage access to Azure SQL databases in your Synapse SQL pool Encrypt sensitive information at rest: use TDE (Transparent data encryption); Log and alert on changes to critical Azure resources (Azure Monitor : Activity Log and alerts on Synapse in the Azure portal)
Vulnerability management	Run automated vulnerability scanning tools; Compare back-to-back vulnerability scans (Vulnerability Assessment is a scanning service built into Azure Synapse SQL.); Use a risk-rating process to prioritize the remediation of discovered vulnerabilities (Data Discovery & Classification built into Azure Synapse SQL)
Inventory and asset management	Use automated asset discovery solution (Azure Resource Graph); Maintain asset metadata (Apply tags to Azure resources). Delete unauthorized Azure resources. Define a list of approved Azure resources related to your Synapse SQL pool. Monitor for unapproved Azure resources (use Azure Policy and Azure Resource Graph) Use only approved Azure services Any resource related to your Synapse SQL pool that is required for business operations, but may incur higher risk for the organization, should be isolated within its own virtual machine and/or virtual network and sufficiently secured with either an Azure Firewall or Network Security Group.
Secure configuration	Use Azure Policy aliases in the "Microsoft.Sql" namespace to create custom policies to audit or enforce the configuration of resources related to your Synapse SQL pool. You may also make use of built-in policy definitions for Azure Database/Server, such as: Deploy Threat Detection on SQL servers / SQL Server should use a virtual network service endpoint. If using custom Azure Policy definitions, use Azure DevOps or Azure Repos to securely store and manage your code. Implement automated configuration monitoring for Azure resources (Azure Security Center) Manage Azure secrets securely:TDE; Use Managed Identities to provide Azure services with an automatically managed identity in Azure Active Directory (AD); Implement Credential Scanner to identify credentials within your code

Security Baseline for Synapse Analytics

Contains recommendations that will help you improve the security posture of your deployment	
Malware defense	Pre-scan files to be uploaded to non-compute Azure resources (Microsoft anti-malware is enabled on the underlying host that supports Azure services. Pre-scan any content being uploaded to non-compute Azure resources, such as App Service, Data Lake Storage, Blob Storage, Azure SQL Server, etc.)
Data recovery	Ensure regular automated back-ups: Snapshots of your Synapse SQL pool are automatically taken throughout the day creating restore points. Periodically test your restore points to ensure your snapshots are valid. Ensure protection of backups and customer-managed keys: By default, data in a storage account is encrypted with Microsoft-managed keys. You can rely on Microsoft-managed keys for the encryption of your data, or you can manage encryption with your own keys. If you are managing your own keys with Key Vault, ensure soft-delete is enabled

Security in Azure Synapse Analytics

Securing with firewalls

Overview

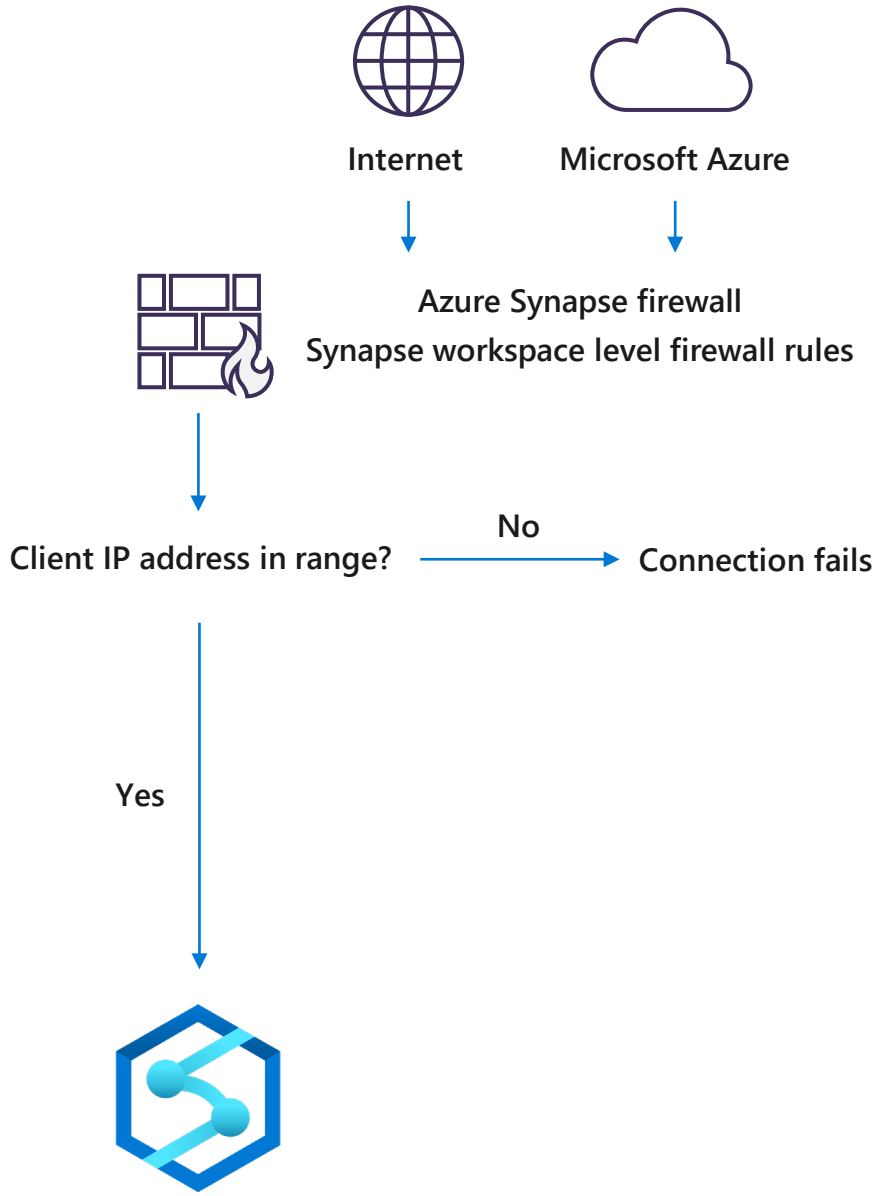
Access to your Azure Synapse Analytics is blocked by the firewall.

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

Rules

Allow specific or range of allowed IP addresses.

Allow Azure applications to connect.



IP Firewall Rules

Overview

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

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

Key Points

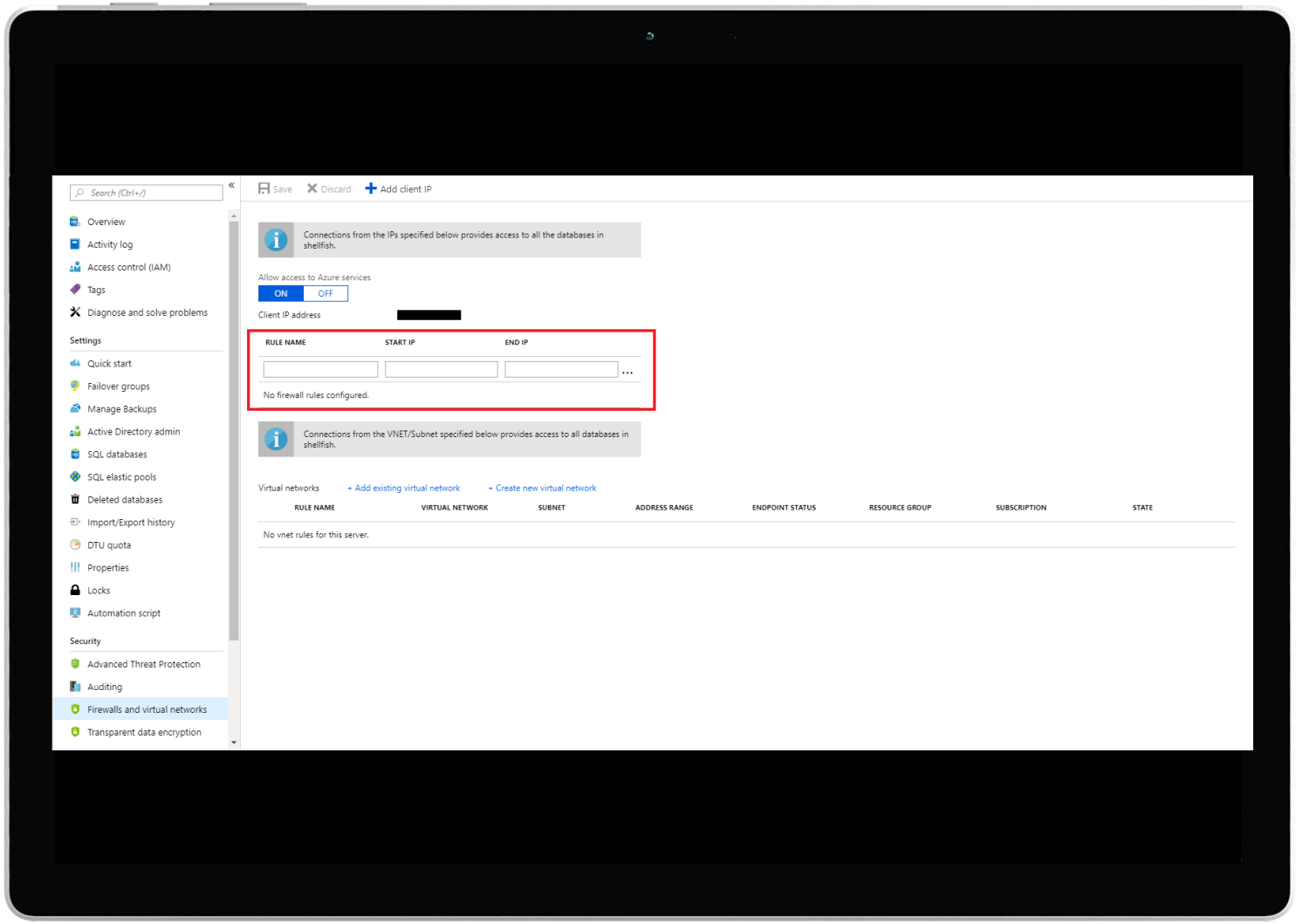
- Customers can also use SQL Server Management Studio (SSMS) to connect to the SQL resources (dedicated SQL pool and serverless SQL pool) in their workspace.
- Customers must ensure that the firewall on the network and local computer allow outgoing communication on TCP ports 80, 443 and 1443 for Synapse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

Firewall configuration on the portal

By default, Azure blocks all external connections to port 1433

Configure with the following steps:

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



Firewall configuration using REST API

Managing firewall rules through REST API must be authenticated.

For information, see Authenticating Service Management Requests.

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

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

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

To list firewall rules, execute the GET.

```
PUT
https://management.azure.com/subscriptions/{subscriptionI
d}/resourceGroups/{resourceGroupName}/providers/Microsoft
.Sql/servers/{serverName}/firewallRules/{firewallRuleName
}?api-version=2014-04-01REQUEST BODY
{
  "properties": {
    "startIpAddress": "0.0.0.3",
    "endIpAddress": "0.0.0.3"
  }
}

DELETE
https://management.azure.com/subscriptions/{subscriptionI
d}/resourceGroups/{resourceGroupName}/providers/Microsoft
.Sql/servers/{serverName}/firewallRules/{firewallRuleName
}?api-version=2014-04-01

GET
https://management.azure.com/subscriptions/{subscriptionI
d}/resourceGroups/{resourceGroupName}/providers/Microsoft
.Sql/servers/{serverName}/firewallRules/{firewallRuleName
}?api-version=2014-04-01
```

Firewall configuration using PowerShell/T-SQL

Windows PowerShell Azure cmdlets

New-AzureRmSqlServerFirewallRule

Get-AzureRmSqlServerFirewallRule

Set-AzureRmSqlServerFirewallRule

Transact SQL

sp_set_firewall_rule

sp_delete_firewall_rule

```
# PS Allow external IP access to SQL DW
PS C:\> New-AzureRmSqlServerFirewallRule
          -ResourceGroupName "myResourceGroup" `
          -ServerName $servername `
          -FirewallRuleName "AllowSome"
          -StartIpAddress "0.0.0.0"
          -EndIpAddress "0.0.0.0“

-- T-SQL Allow external IP access to SQL DW
EXECUTE sp_set_firewall_rule
          @name = N'ContosoFirewallRule’,
          @start_ip_address = '192.168.1.1’,
          @end_ip_address = '192.168.1.10'
```

Managed VNet

Overview

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

Benefits

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

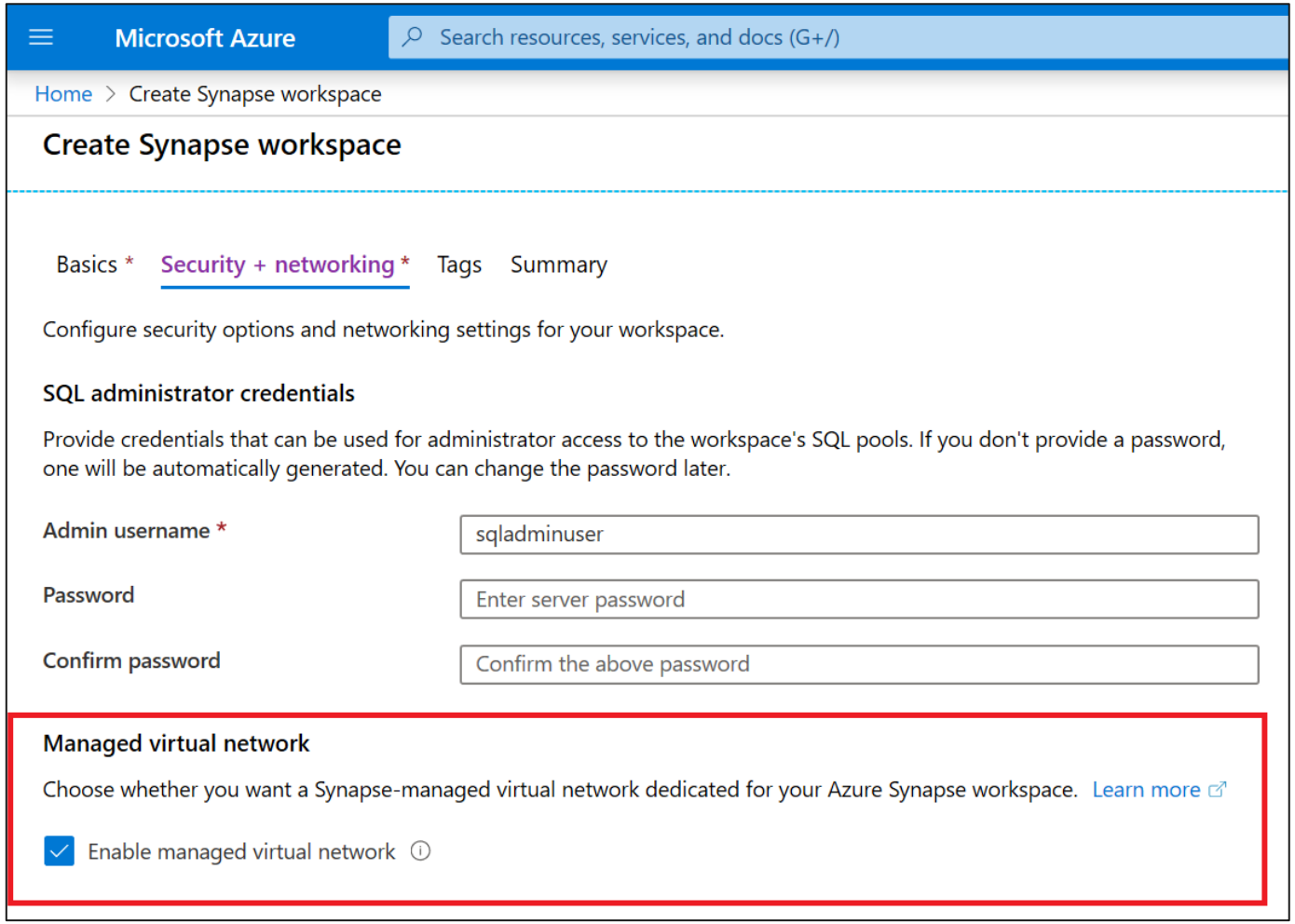
Managed VNet

Overview

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

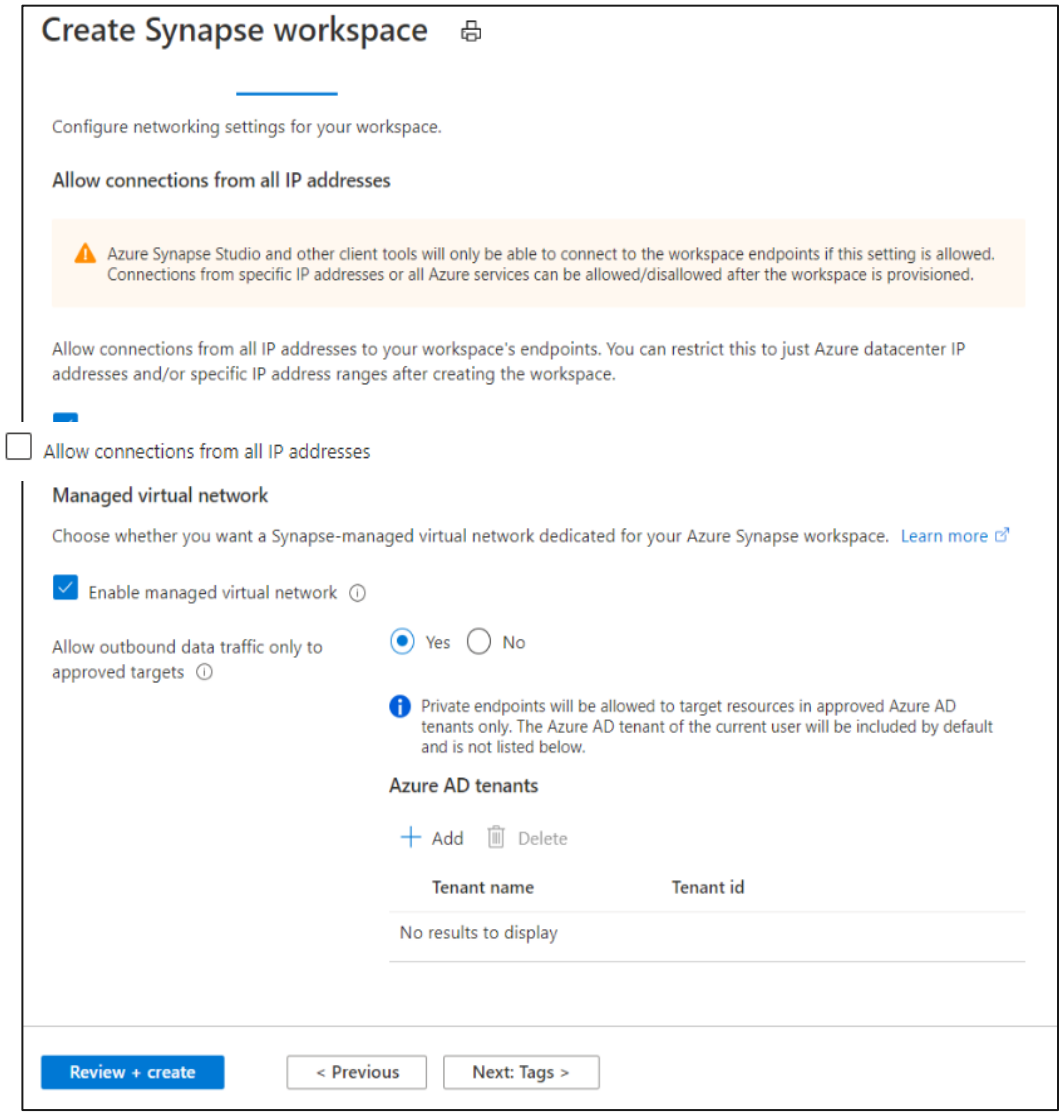
User cannot reconfigure a workspace that does not have a Managed workspace VNet associated with it and associate a VNet to it.

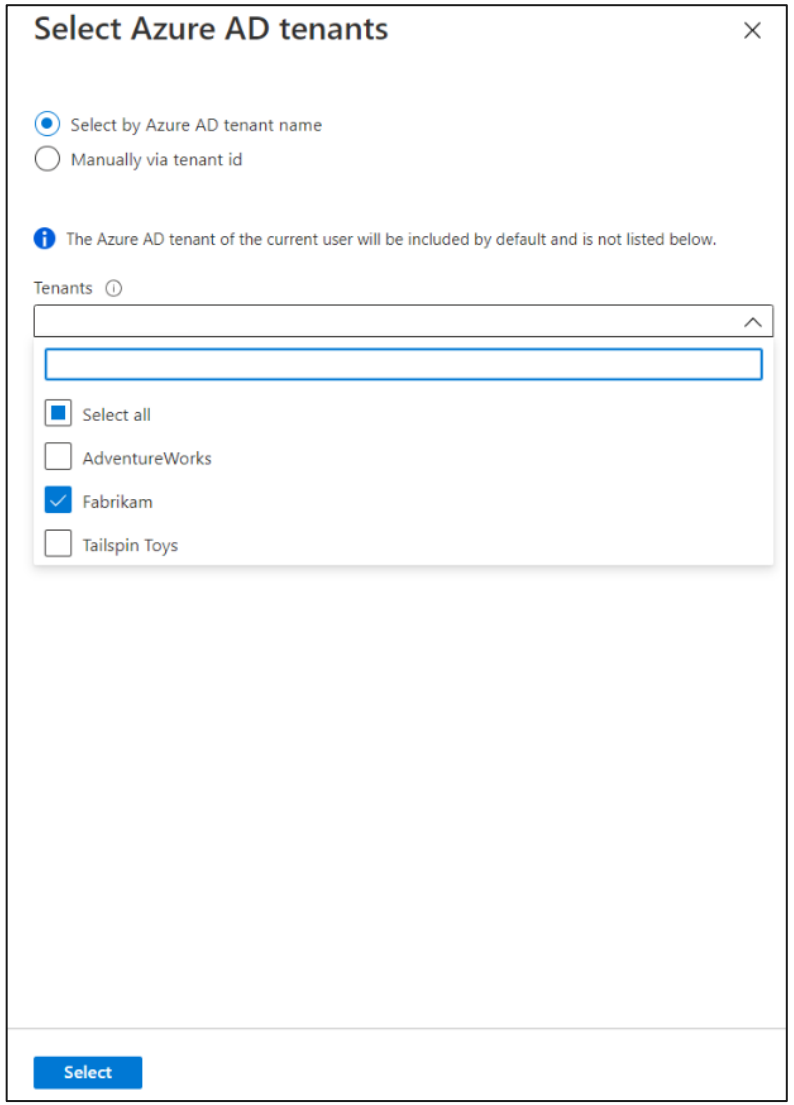
Private links are supported only in Synapse workspaces that have a managed VNet associated to it.



Data Exfiltration

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





Private Endpoints

Overview

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

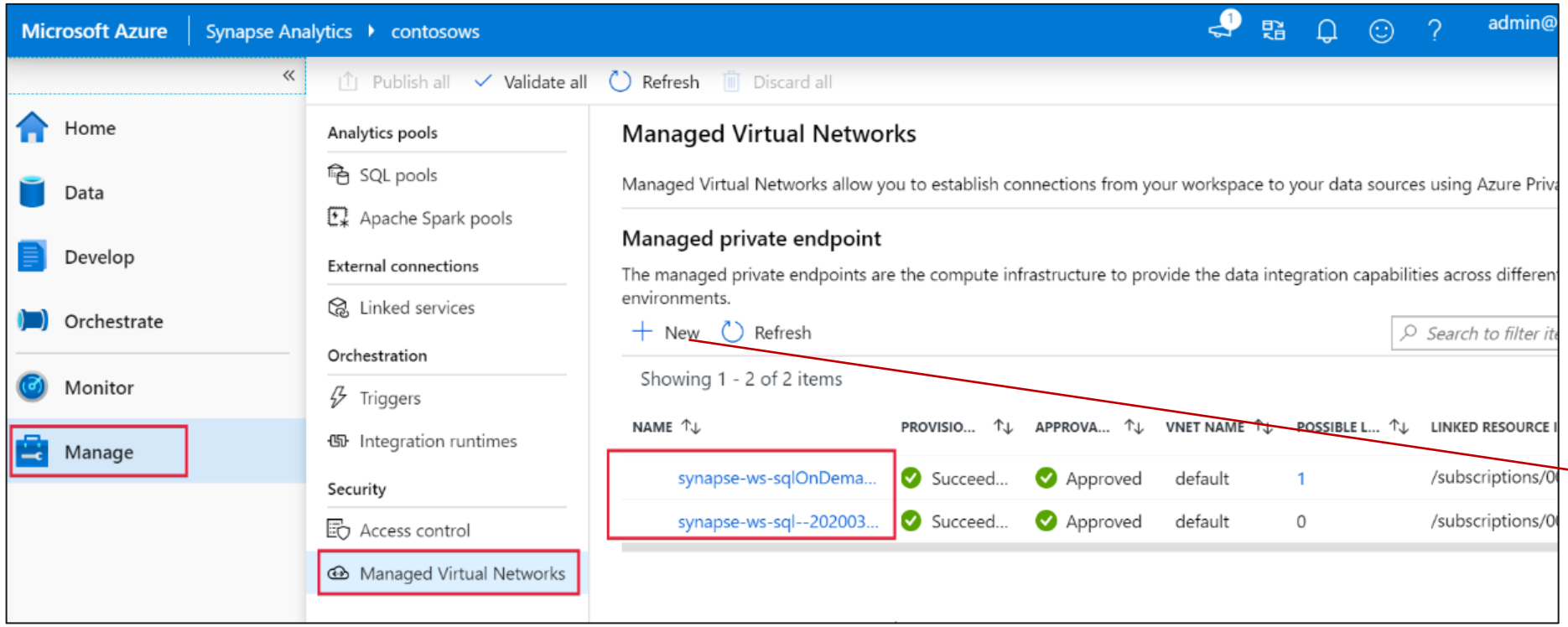
Benefits

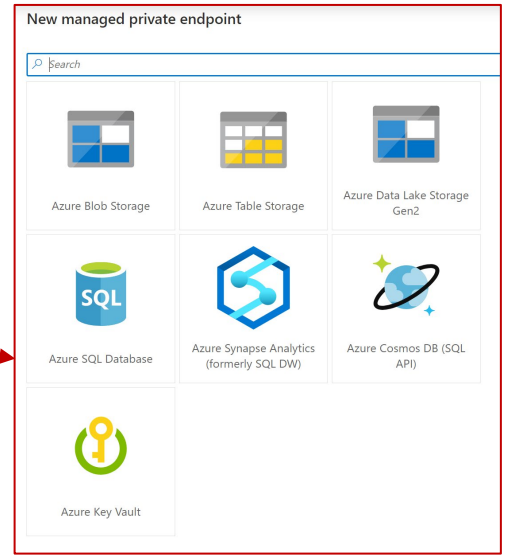
- Private link enables customers to access Azure services and Azure hosted customer/partner services from their Azure VNet securely.
- With use of private link, traffic between customer's VNet and workspace traverses entirely over the Microsoft backbone network.
- Private link protects against data exfiltration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

Private Endpoints for Synapse SQL (dedicated & serverless)

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

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





Managed VNets in the Synapse Workspace

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

Synapse Workspace

Managed VNet

Synapse Studio

Managed private endpoint

Managed private endpoint to serverless SQL

Managed private endpoint to SQL Pool

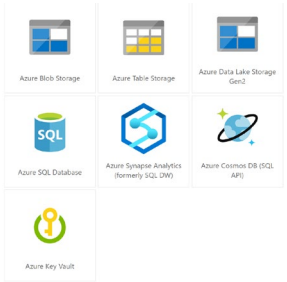
Spark Clusters

Workspace Serverless SQL

Workspace SQL Pool

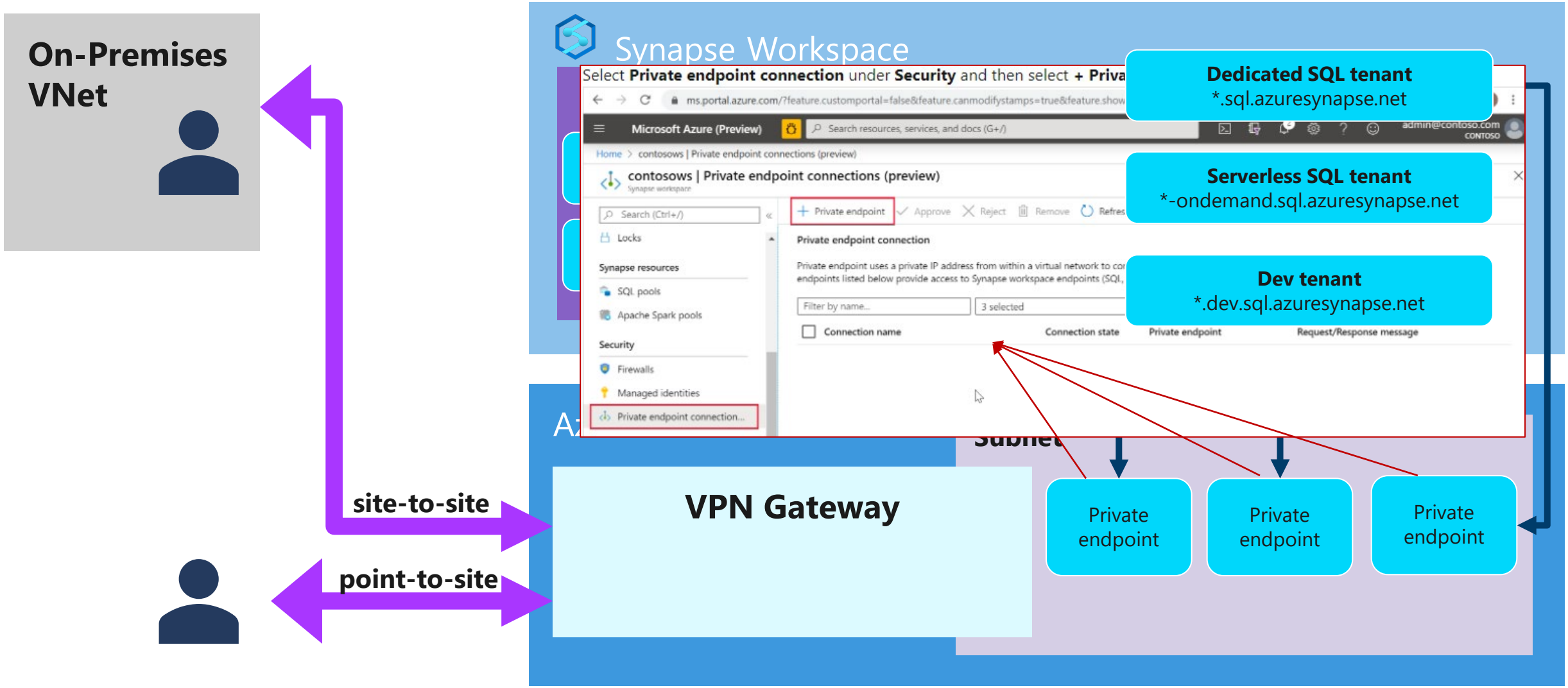
Internet

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

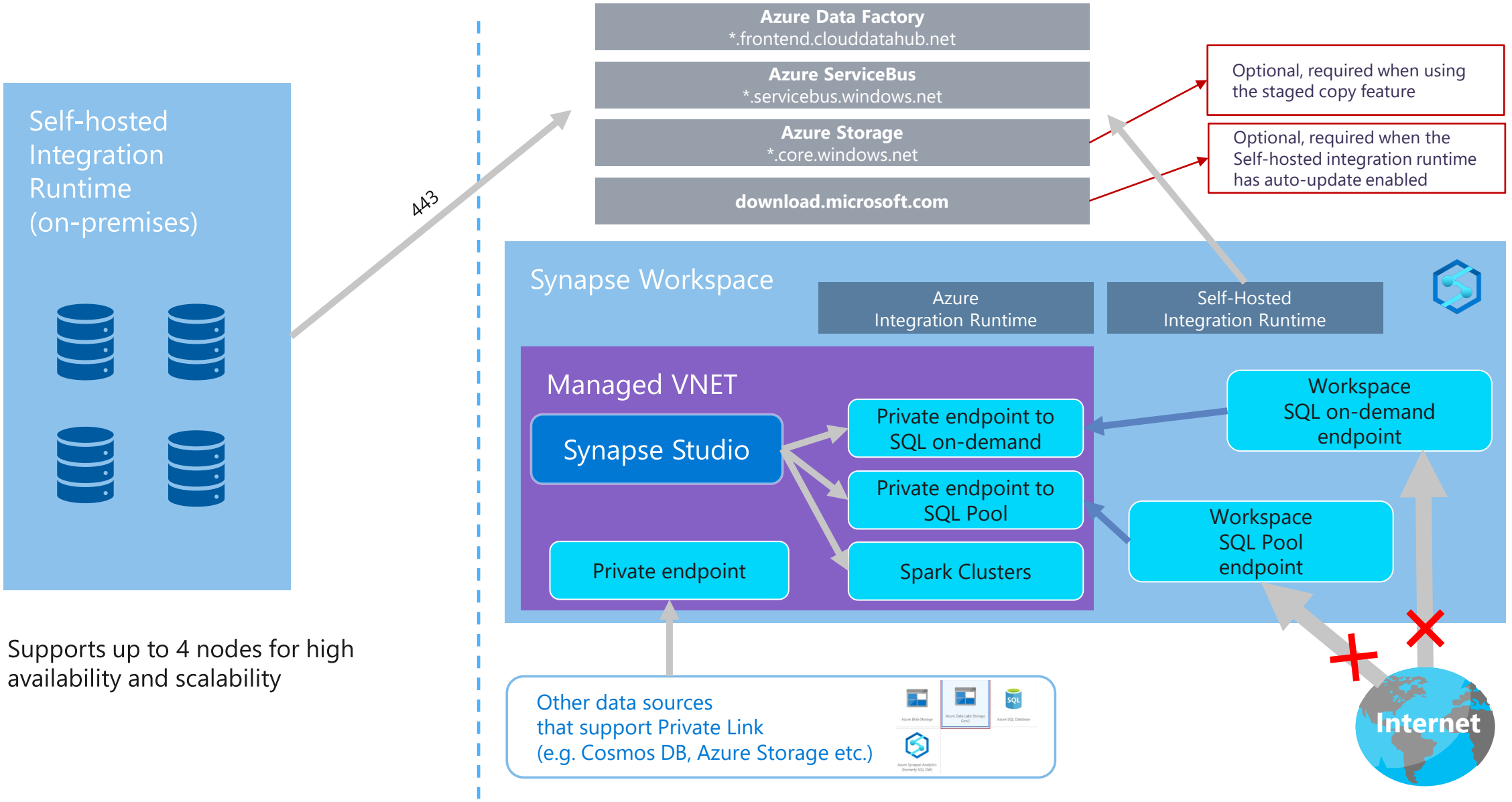


The private endpoint establishes a private link between your Azure Synapse Analytics workspace and your data sources where traffic between your VNet and workspace traverses entirely over the Microsoft backbone network.

Connecting to Synapse from On-Premises



Self-hosted Integration Runtime in Synapse Workspace

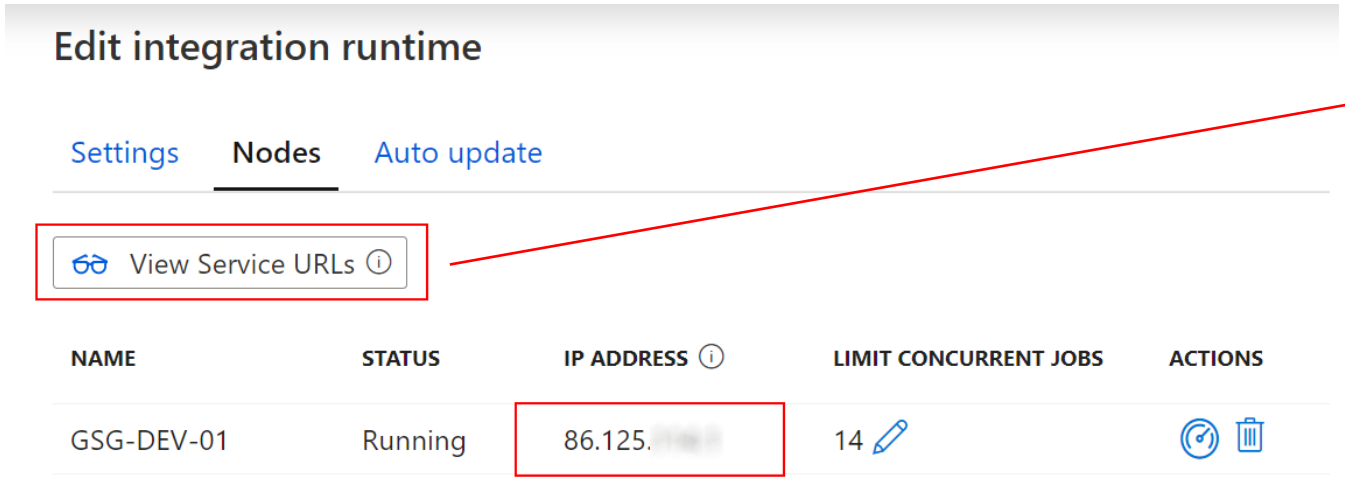


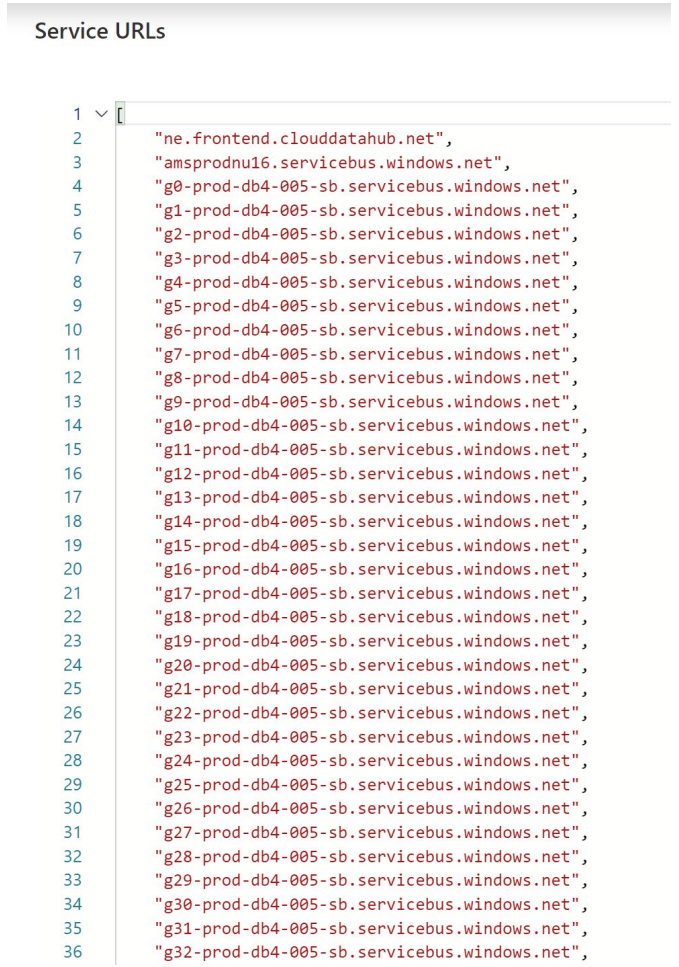
Supports up to 4 nodes for high availability and scalability

Self-hosted Integration Runtime in Synapse Workspace

Allowing only specific public Azure endpoints

Allowing only IR-specific IP addresses





Azure Active Directory authentication

Overview

Manage user identities in one location.

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

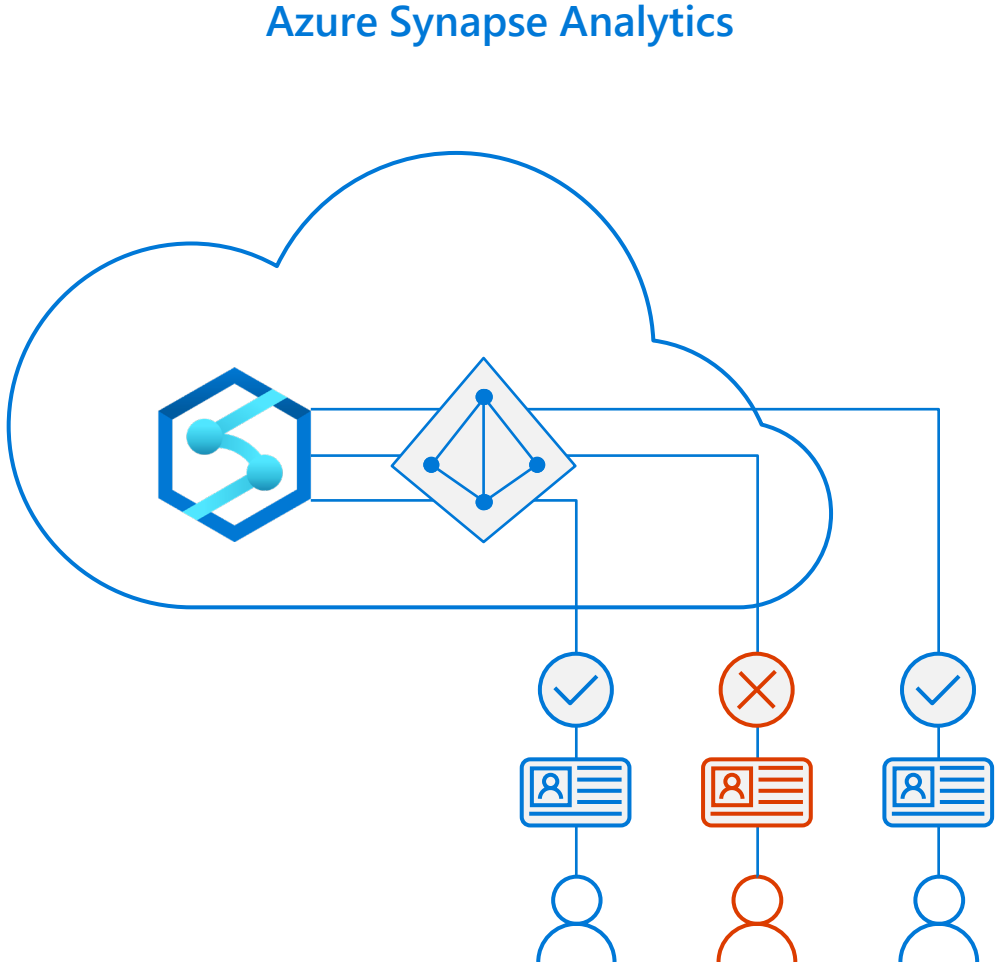
Benefits

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

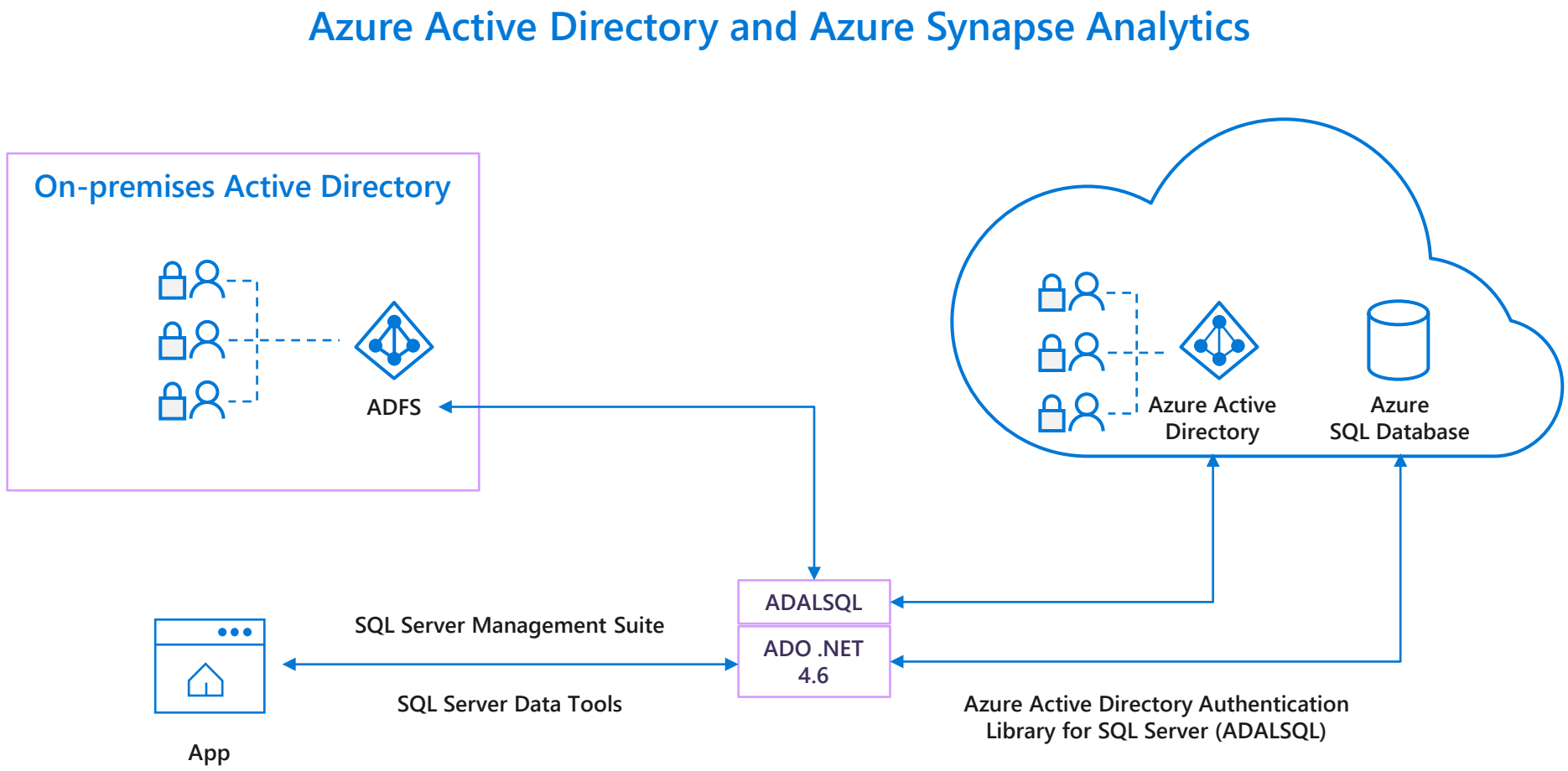
Allows password rotation in a single place

Alternative to SQL Server authentication

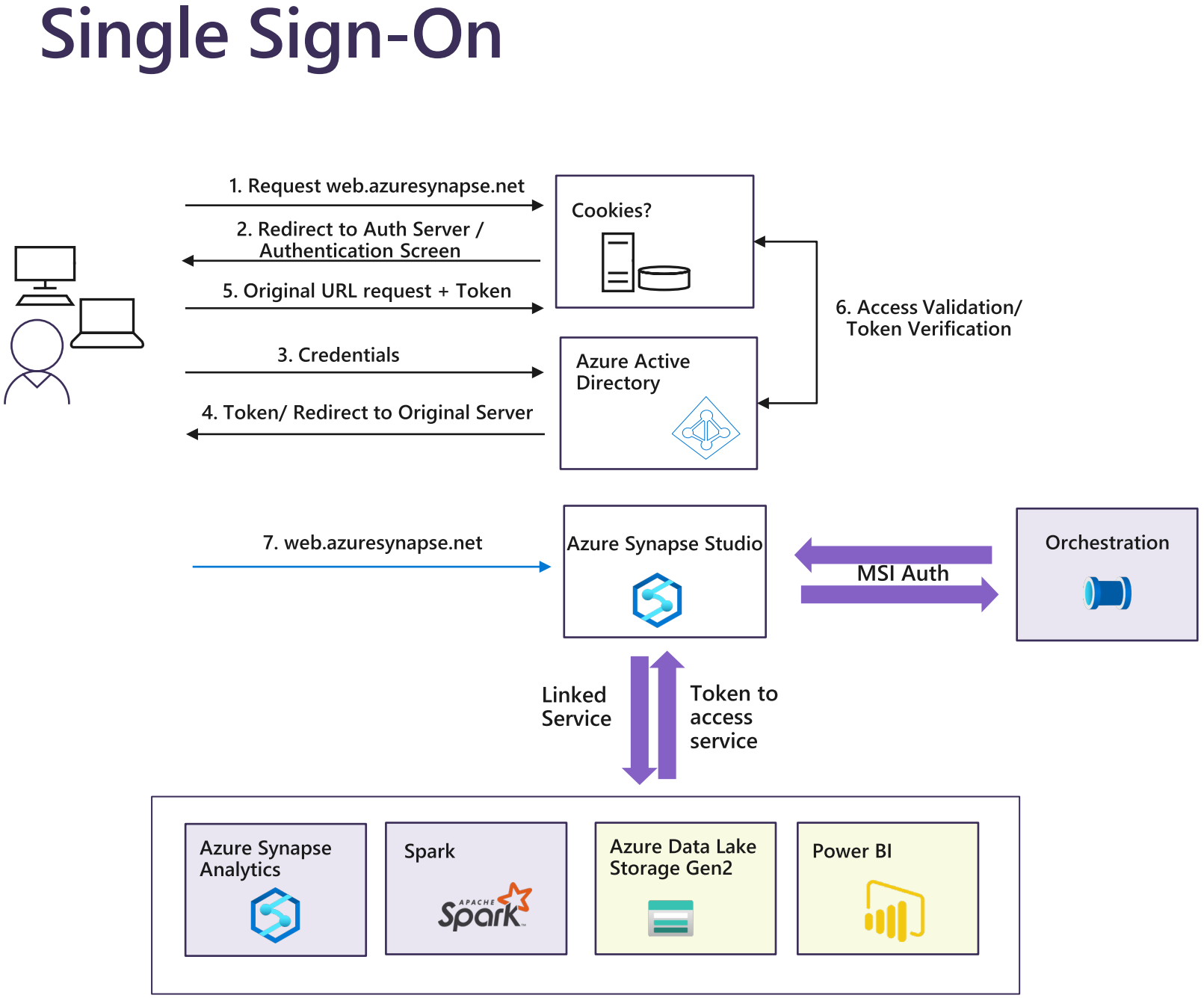
Eliminates the need to store passwords



Azure Active Directory trust architecture



Single Sign-On



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

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

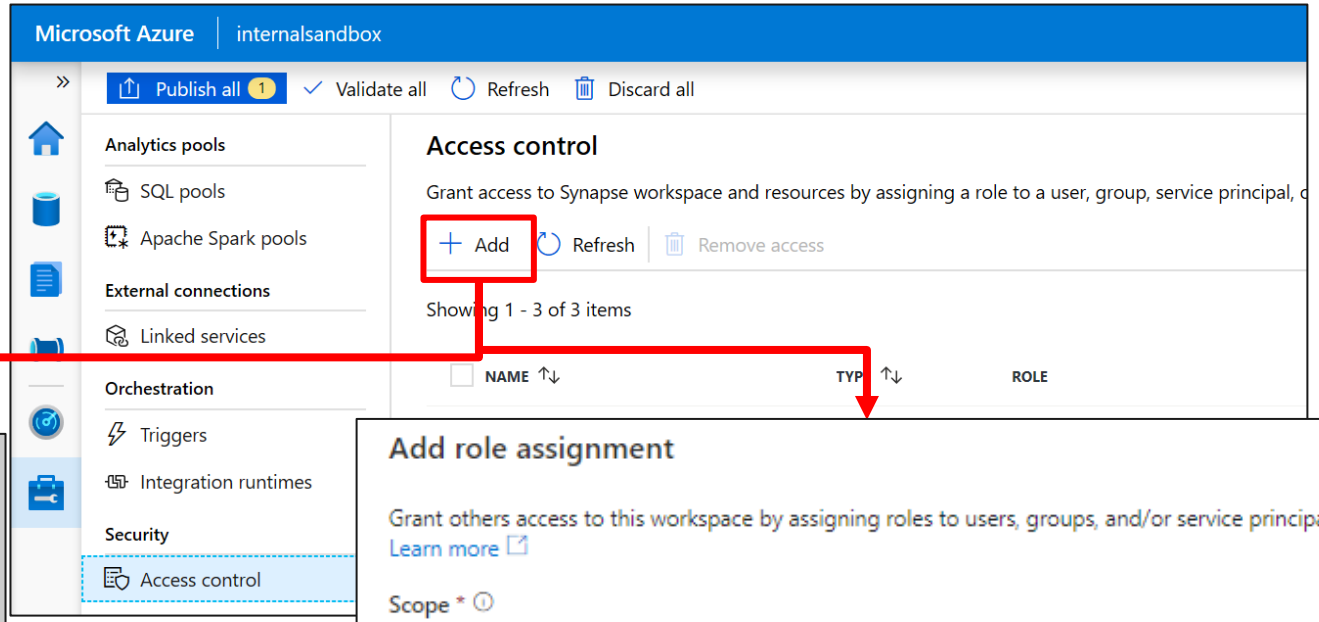
1. ADLS Gen2
2. Azure Synapse Analytics
3. Power BI
4. Spark – Spark Livy API
5. management.azure.com – resource provisioning
6. Develop artifacts – dev.workspace.net
7. Graph endpoints

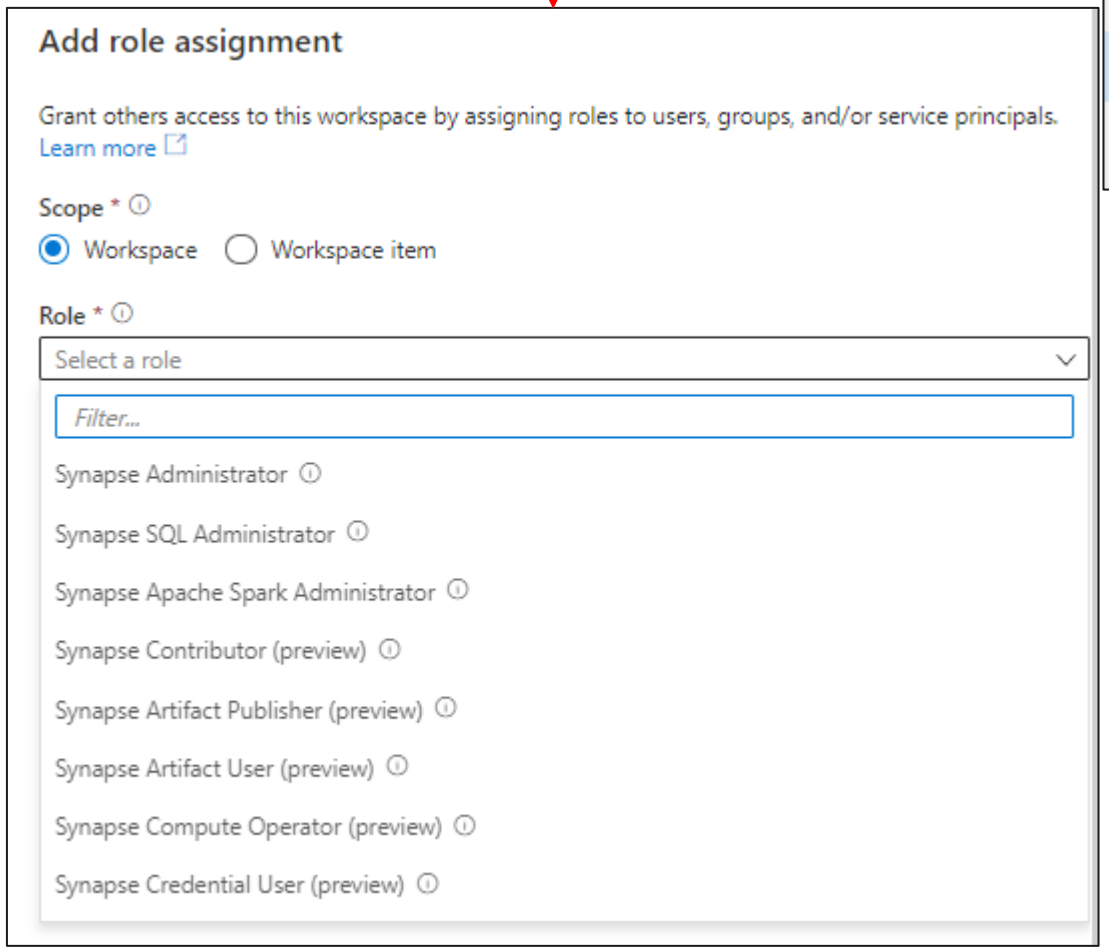
MSI authentication - Orchestration uses workspace MSI auth for automation

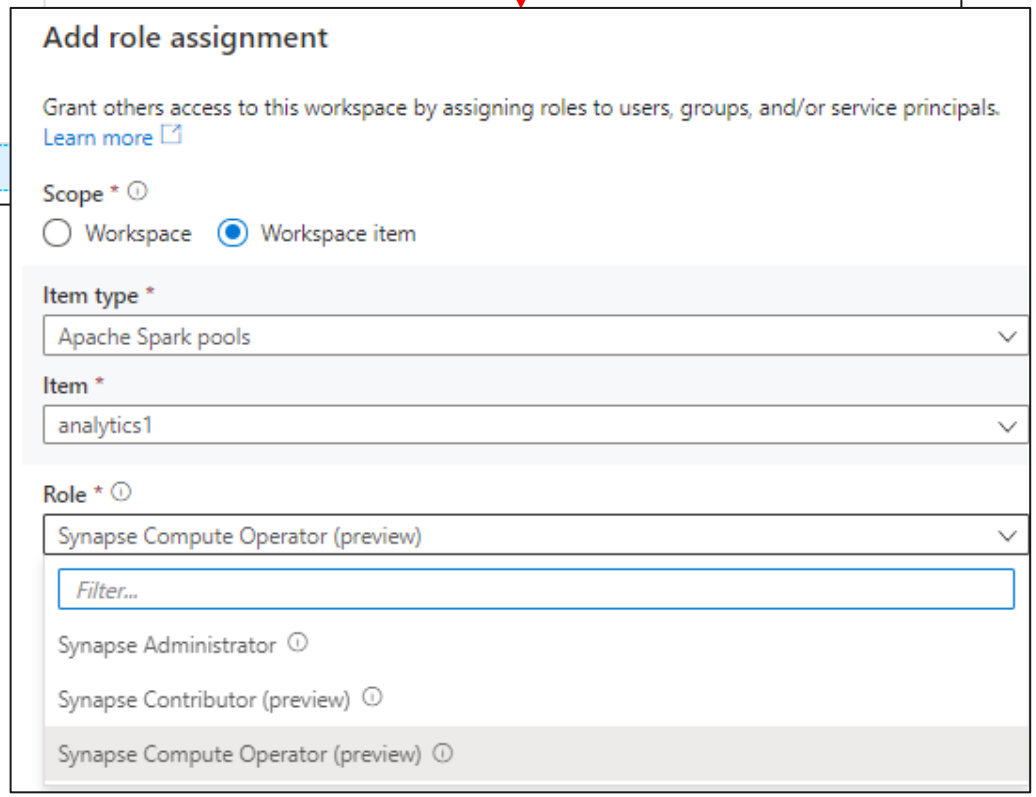
Access Control

Overview

It provides access control management to workspace resources and artifacts







Synapse RBAC Roles and permitted actions

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

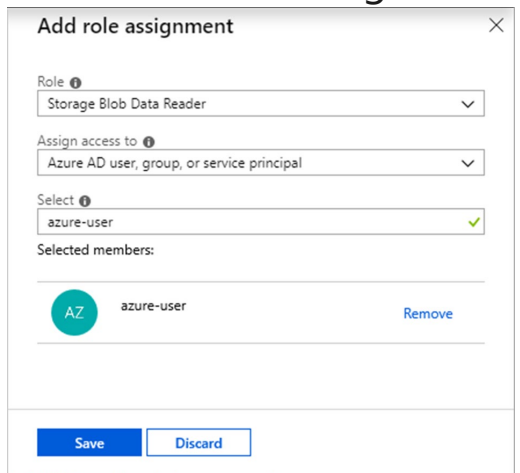
Access control – serverless SQL pool

Overview

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

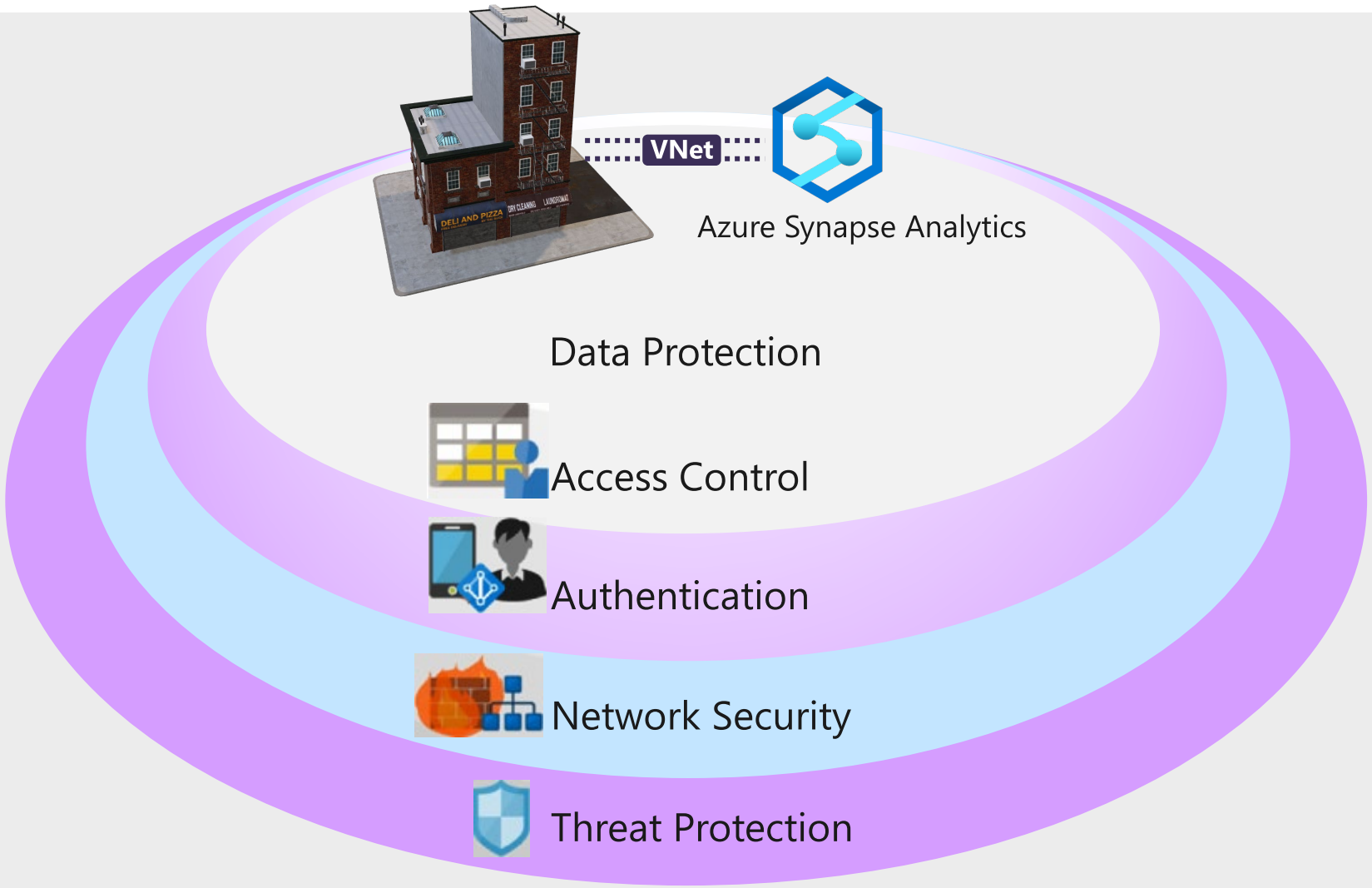
Benefits

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




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

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

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

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

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

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

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


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

Enterprise-grade security



Defense-in-Depth

Industry-leading compliance

ISO 27001

SOC 1 Type 2

SOC 2 Type 2

PCI DSS Level 1

Cloud Controls Matrix

ISO 27018

Content Delivery and Security Association

Shared Assessments

FedRAMP JAB P-ATO

HIPAA / HITECH

FIPS 140-2

21 CFR Part 11

FERPA

DISA Level 2

CJIS

IRS 1075

ITAR-ready

Section 508 VPAT

European Union Model Clauses

EU Safe Harbor

United Kingdom G-Cloud

China Multi Layer Protection Scheme

China GB 18030

China CCCPPF

Singapore MTCS Level 3

Australian Signals Directorate

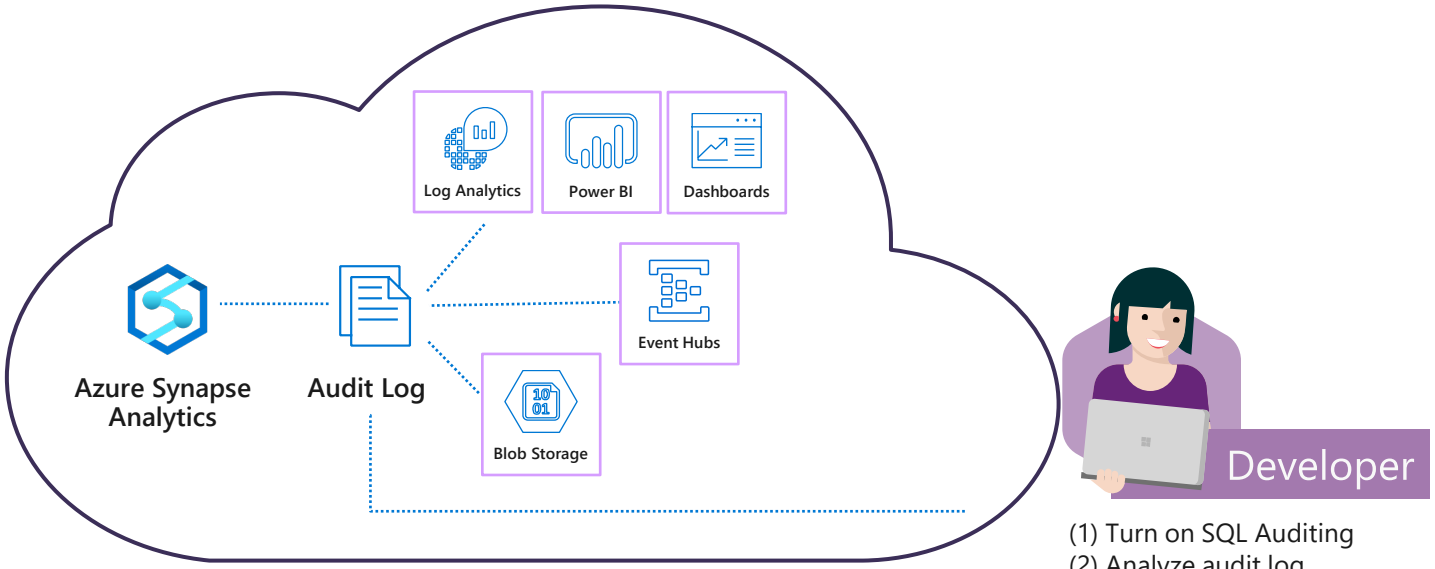
New Zealand GCIO

Japan Financial Services

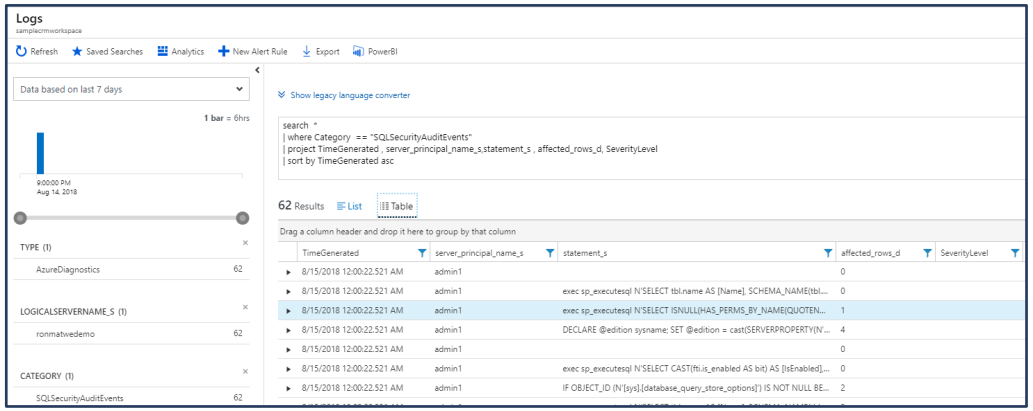
ENISA IAF

SQL auditing in Azure Log Analytics and Event Hubs

Gain insight into database audit log



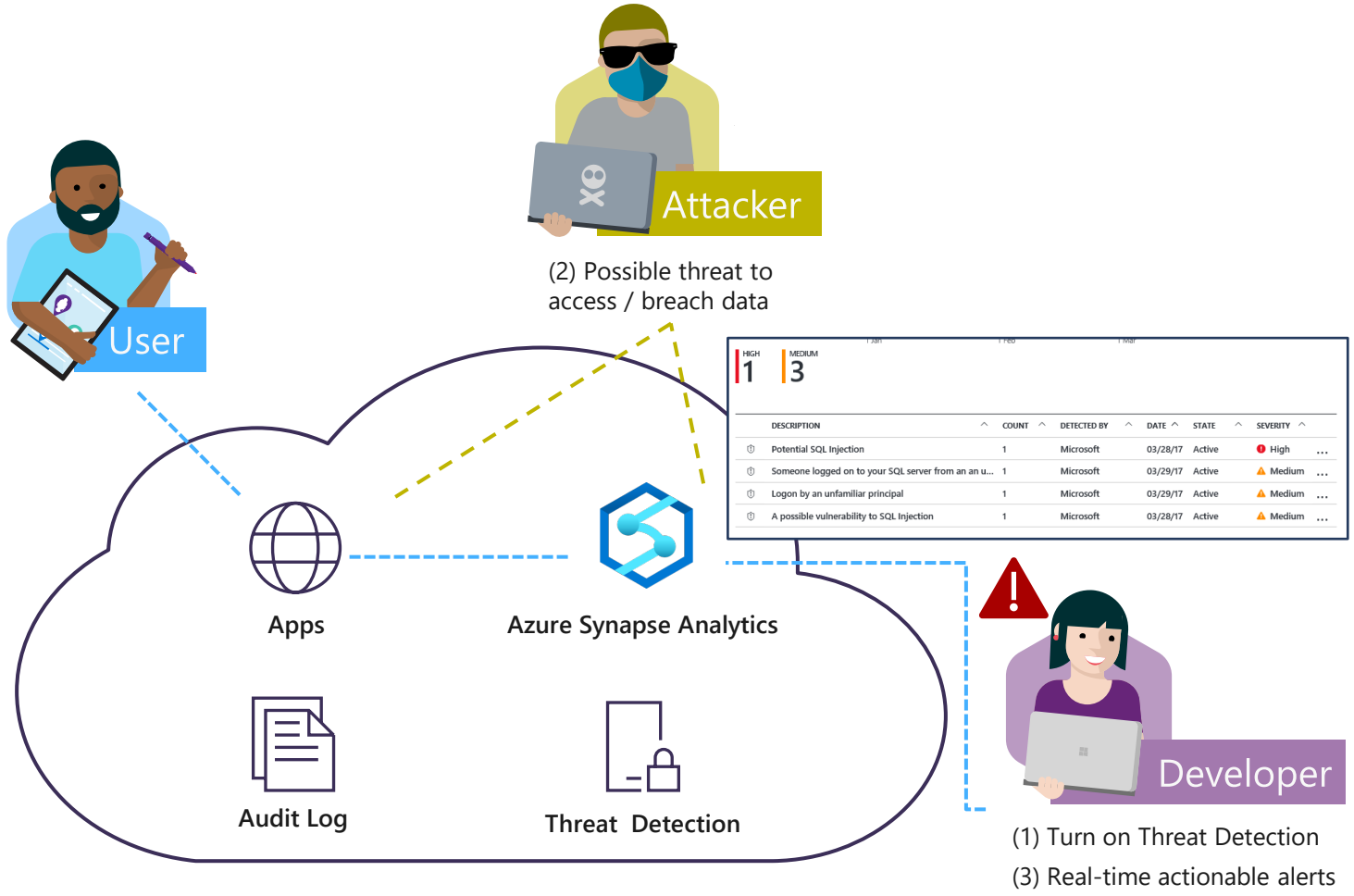
(1) Turn on SQL Auditing
(2) Analyze audit log



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

SQL threat detection

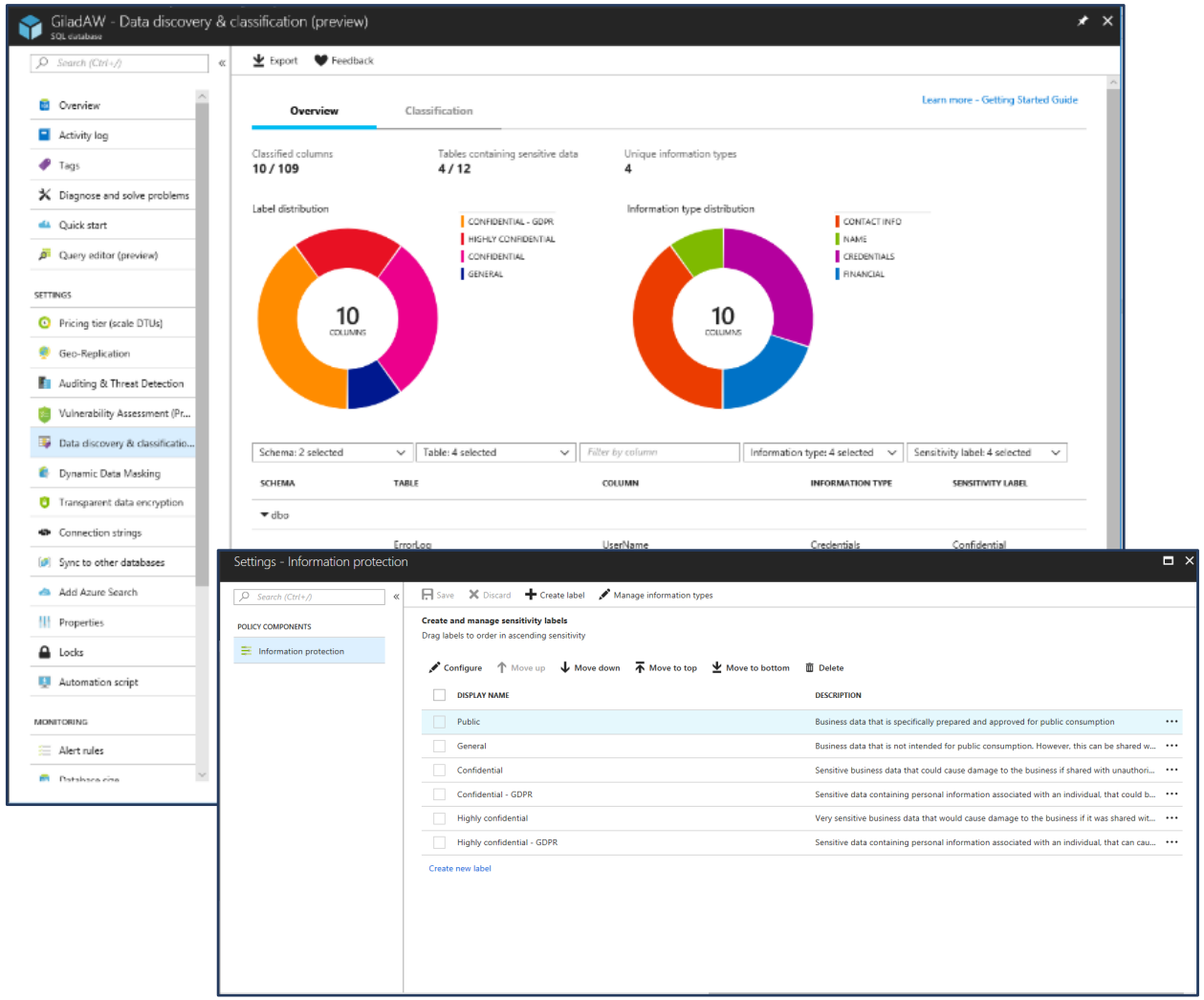
Detect and investigate anomalous database activity



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

SQL Data Discovery & Classification

Discover, classify, protect and track access to sensitive data



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

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

Overview

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

Prevent unauthorized queries against certain tables.

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

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

Row-level security (RLS)

Overview

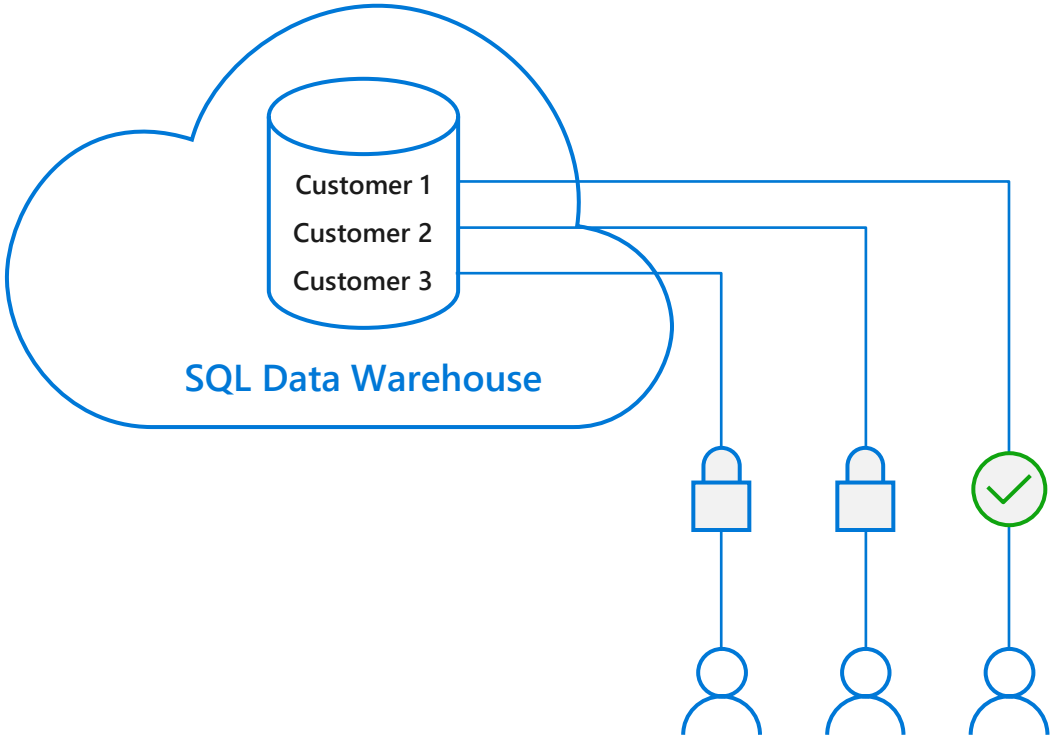
Fine grained access control of specific rows in a database table.

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

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

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

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



Row-level security

Creating policies

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

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

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


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

Column-level security

Overview

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

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

Administer via GRANT T-SQL statement.

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

Dynamic Data Masking

Overview

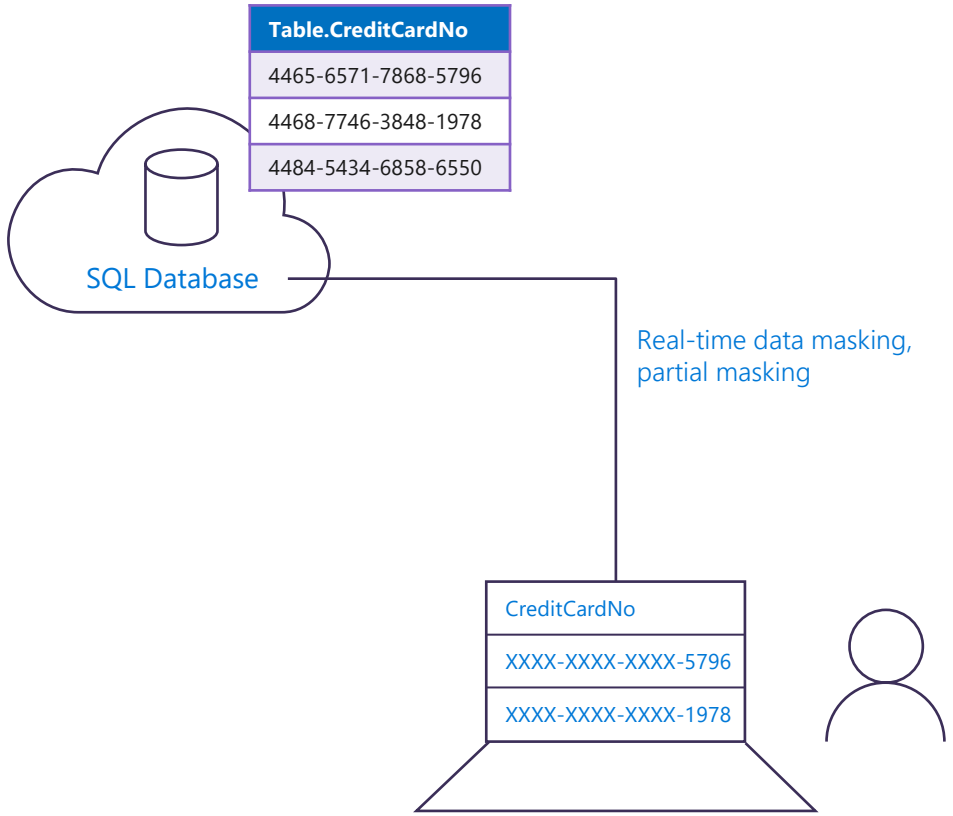
Prevent abuse of sensitive data by hiding it from users

Easy configuration in new Azure Portal

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

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

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



Column Level Encryption

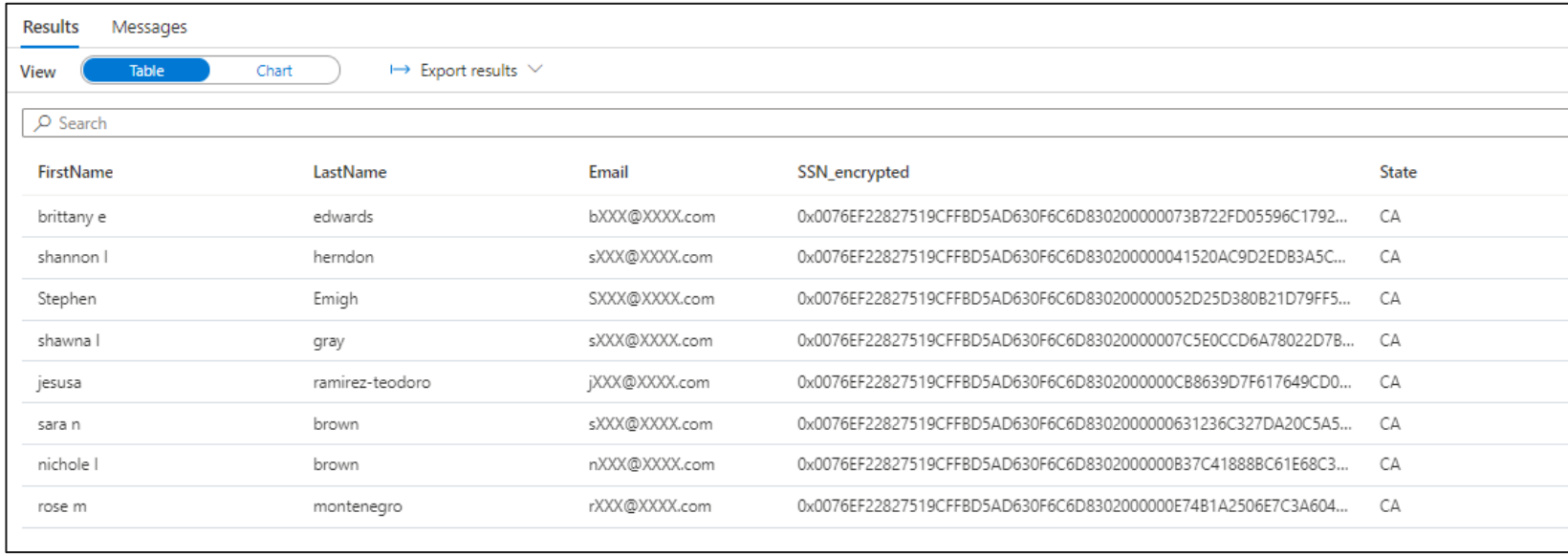
Overview

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

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

5 step process to set up CLE

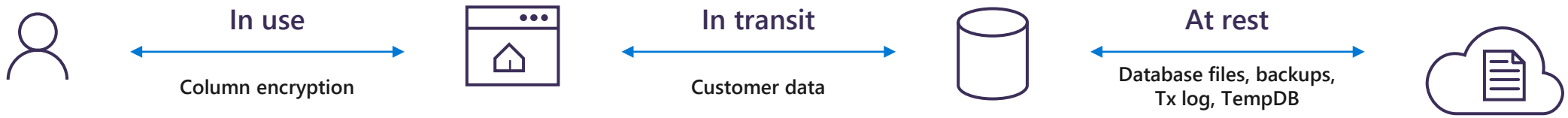
1. Create master key
2. Create certificate
3. Configure symmetric key for encryption
4. Encrypt the column data
5. Close symmetric key



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

Types of data encryption

Data Encryption	Encryption Technology	Customer Value
In transit	Transport Layer Security (TLS) from the client to the server TLS 1.2	Protects data between client and server against snooping and man-in-the-middle attacks
At rest	Transparent Data Encryption (TDE) for Azure Synapse Analytics	Protects data on the disk User or Service Managed key management is handled by Azure, which makes it easier to obtain compliance



Transparent data encryption (TDE)

Overview

All customer data encrypted at rest

TDE performs real-time I/O encryption and decryption of the data and log files.

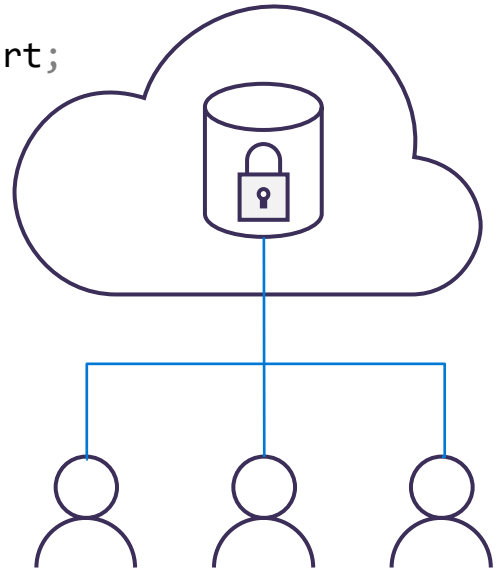
Service OR User managed keys.

Application changes kept to a minimum.

Transparent encryption/decryption of data in a TDE-enabled client driver.

Compliant with many laws, regulations, and guidelines established across various industries.

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<UseStrongPasswordHere>';
go
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My DEK Certificate';
go
USE MyDatabase;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
GO
ALTER DATABASE MyDatabase
SET ENCRYPTION ON;
GO
```



Transparent data encryption (TDE)

Key Vault

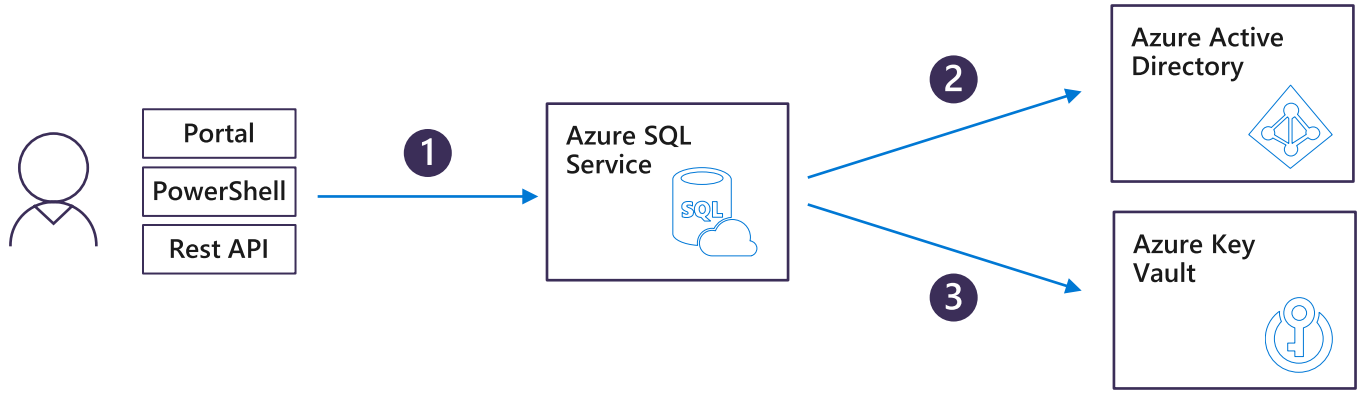
Benefits with User Managed Keys

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

Highly available and scalable cloud-based key store.

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

Configurable via Azure Portal, PowerShell, and REST API.



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

Microsoft Azure Solutions Business

Thank you

Breakout Activity: Security

OUTCOMES

As a result of participating in this activity, you will better be able to **apply security concepts including row level security, object level security, column level security, sensitive data discovery and classification and access controls to a customer scenario.**

ACTIVITY

55 minutes | Learning Adviser | Table Group Channel

As a team review the WWI requirements provided in student guide.

Open the **whiteboard** and as a team answer the provided challenge questions.

The questions are pre-loaded into the whiteboard.

Build Hands-on Lab: Security

OUTCOMES

As a result of participating in this lab, you will be better able to **articulate several security-related steps that cover an end-to-end security story for Azure Synapse Analytics.**

ACTIVITY

50 minutes | Independent | Table Group Channel & CloudLabs Environment

Independently review and complete Exercises 1-3 in the Lab 5 Guide.

Login to your **CloudLabs environment** and work through a set of tasks you would typically follow in work associated with ingesting and integrating your customer's data.

The Lab Guide is available in the 'Files' tab of the General channel. Engage your Table Group call for support and troubleshooting.

Closing

Thank you for your participation in today's Azure Synapse Technical Boot Camp!

TODAY

We learned:

- ✓ Implement optimization strategizes for the data warehouse using SQL based approached in Azure Synapse Analytics.
- ✓ Apply security concepts including row level security, object level security, column level security, sensitive data discovery and classification and access controls to a customer scenario

TOMORROW

We will learn to:

- Implement a process to train a machine learning model
- Address scenarios to monitor and manage Azure solutions

Microsoft Azure
Solutions Business

Mentimeter Poll

Scan QR Code

or

Go to www.menti.com and use code

6897 5086

Microsoft Azure Solutions Business