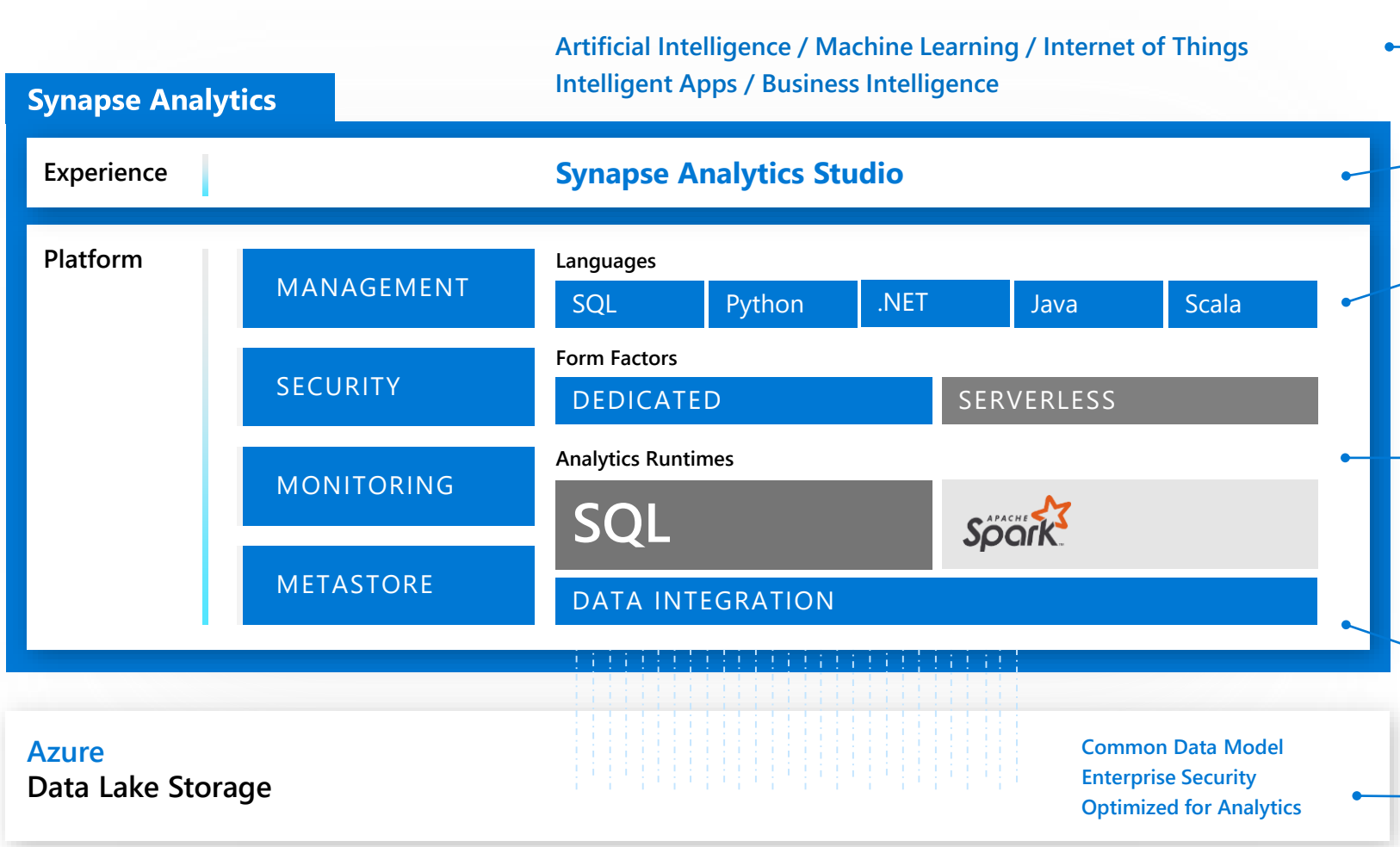
Microsoft Cloud Innovator MEGAZONE MEGAZONE CLOUD

Azure Synapse Analytics

Azure Synapse Analytics

통찰력을 갖춘 분석 서비스



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

Sections

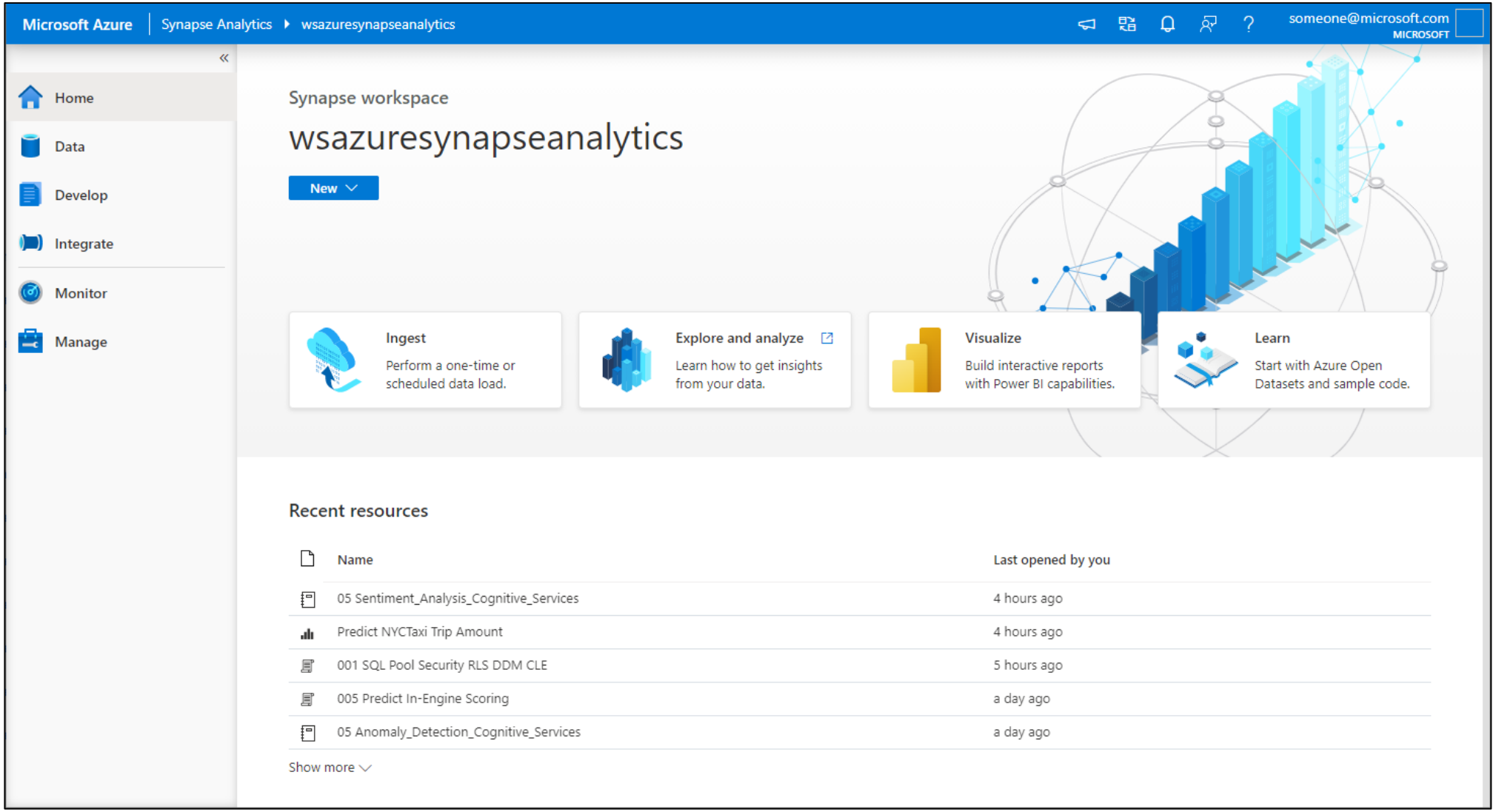
- Studio
- Integrate
- Synapse SQL
  - Dedicated SQL Pool
  - Serverless SQL Pool
- Spark Analytics

Azure Synapse Analytics

Studio

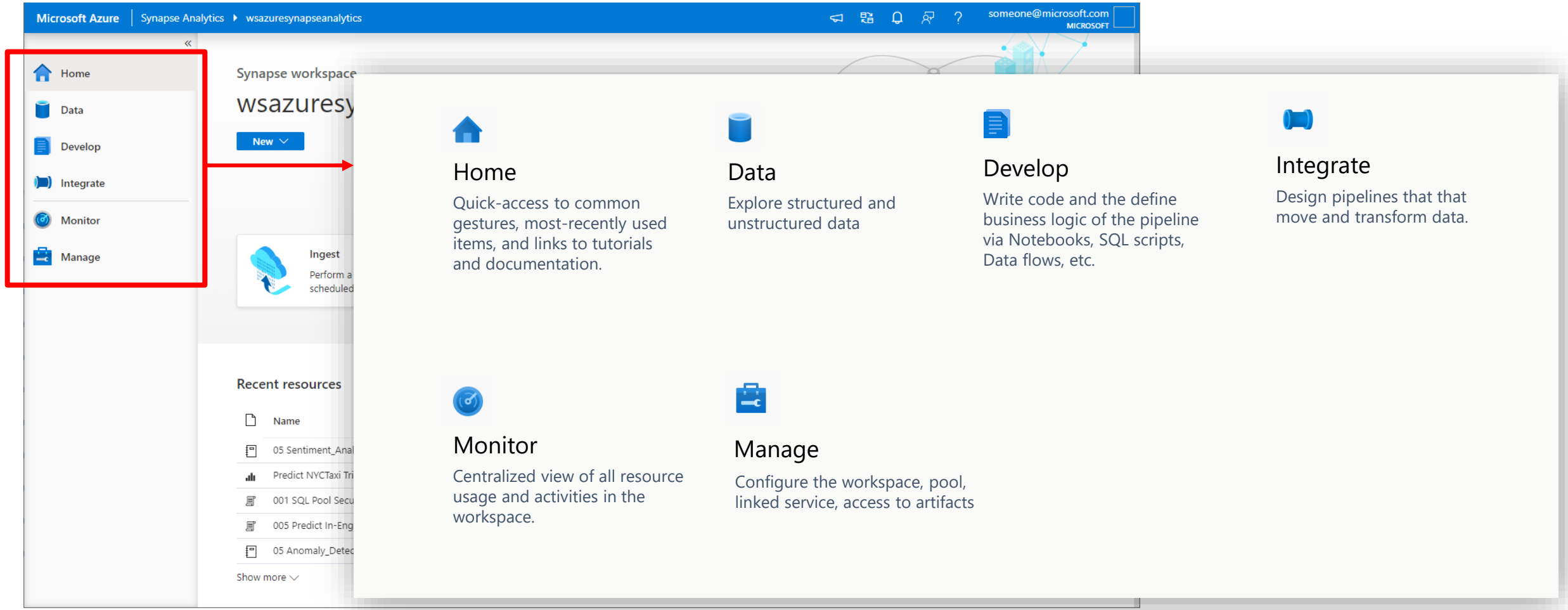
Studio

Studio는 작업자의 분석과 협업을 위한 작업 공간입니다.



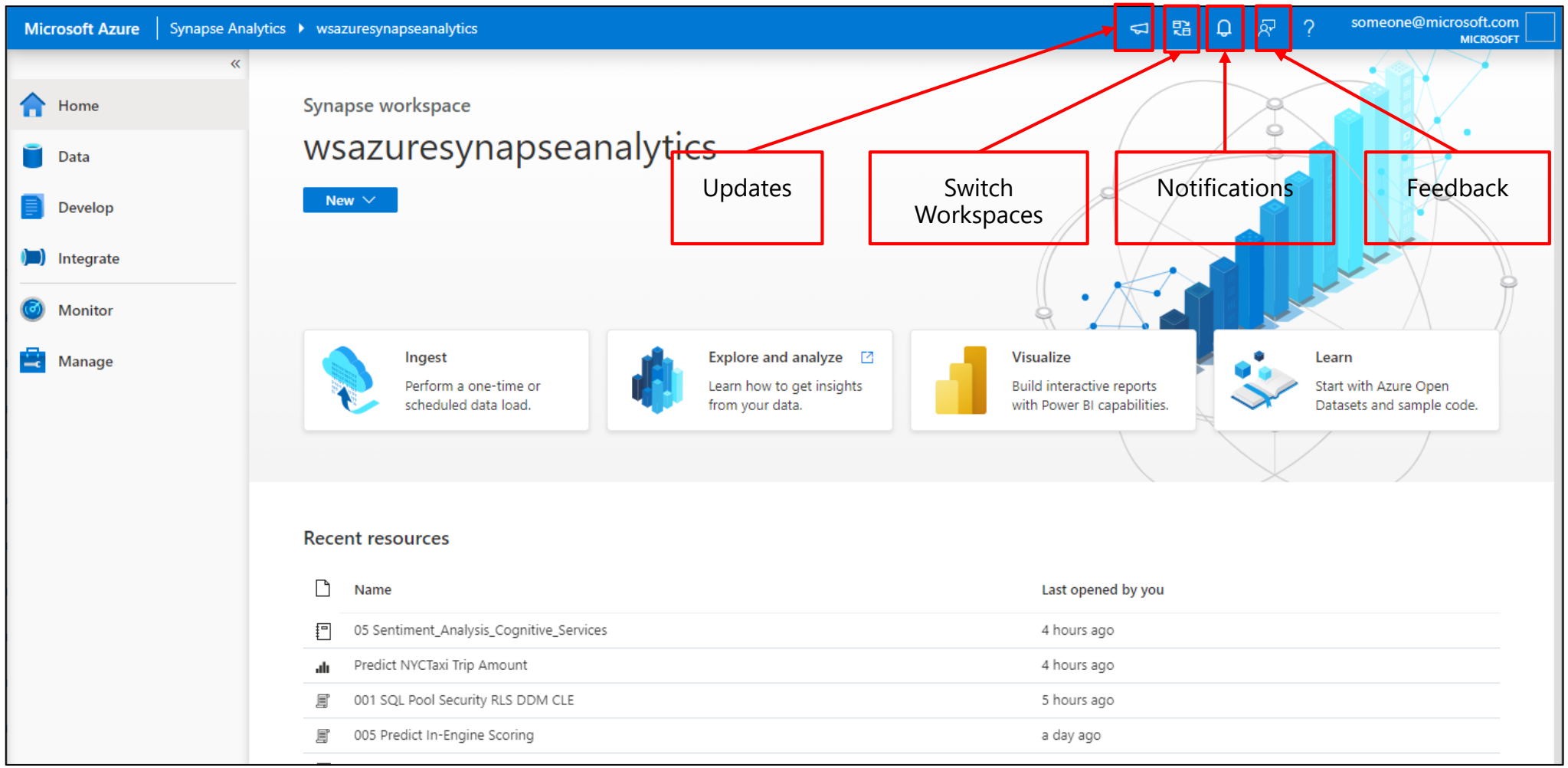
Synapse Studio

Synapse Studio는 작업을 위해 각 Activity hubs 로 나뉘고, 각 Activity hub는 분석을 위한 하위 Activity 들로 구성되어 있습니다.



Home Hub

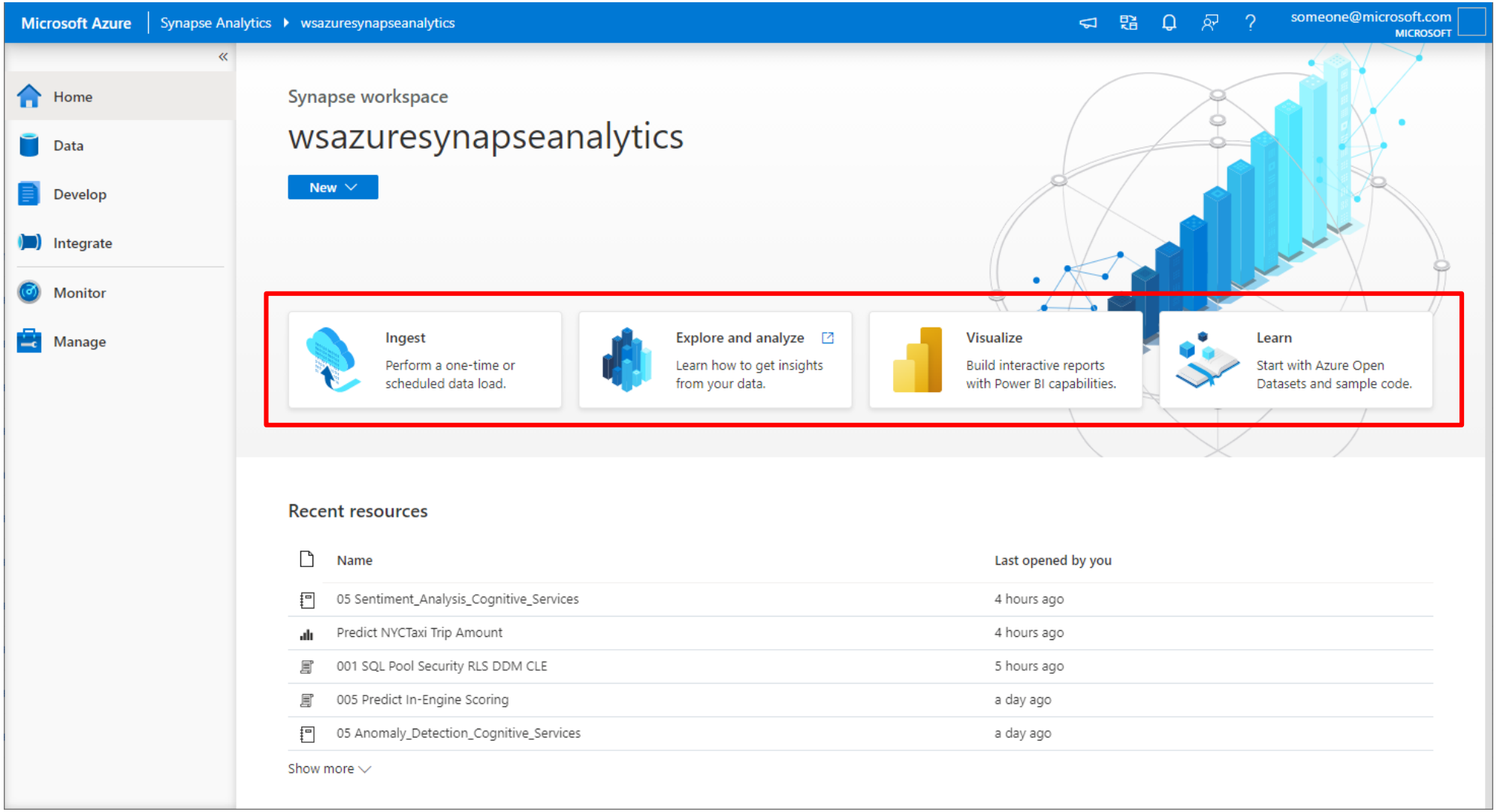
Synapse를 구성하는 Resource 업데이트 릴리즈, 작업공간 전환, 알림, 피드백 제공에 쉽게 접근 가능합니다.



Synapse Studio
Home hub

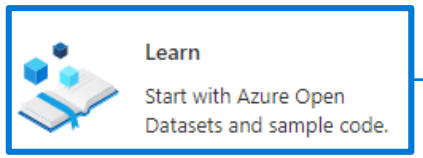
Home Hub

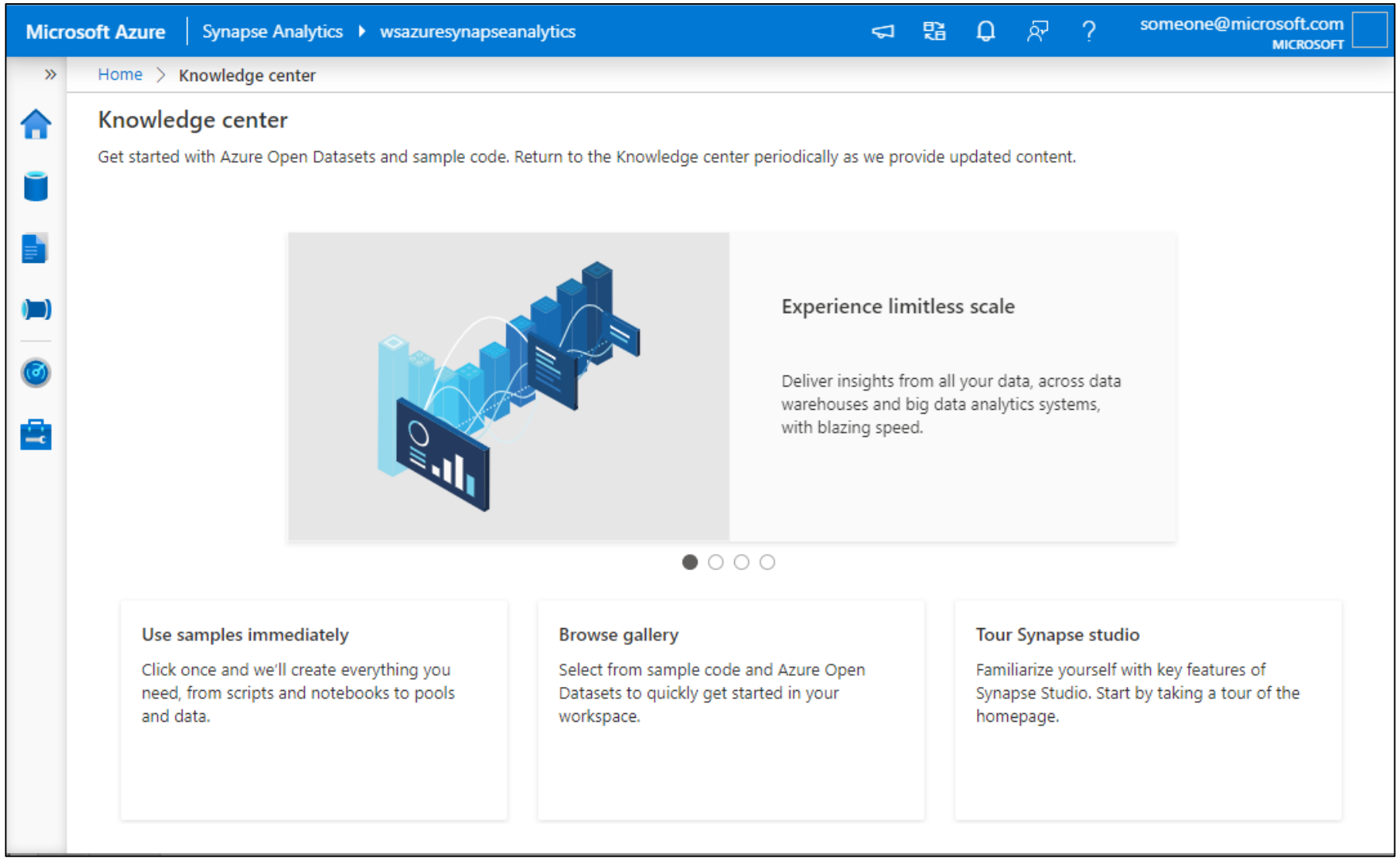
사용자들의 학습을 위해 데이터 수집 작업, 학습 문서로의 링크, 샘플 데이터를 제공합니다.



Home Hub - Learn

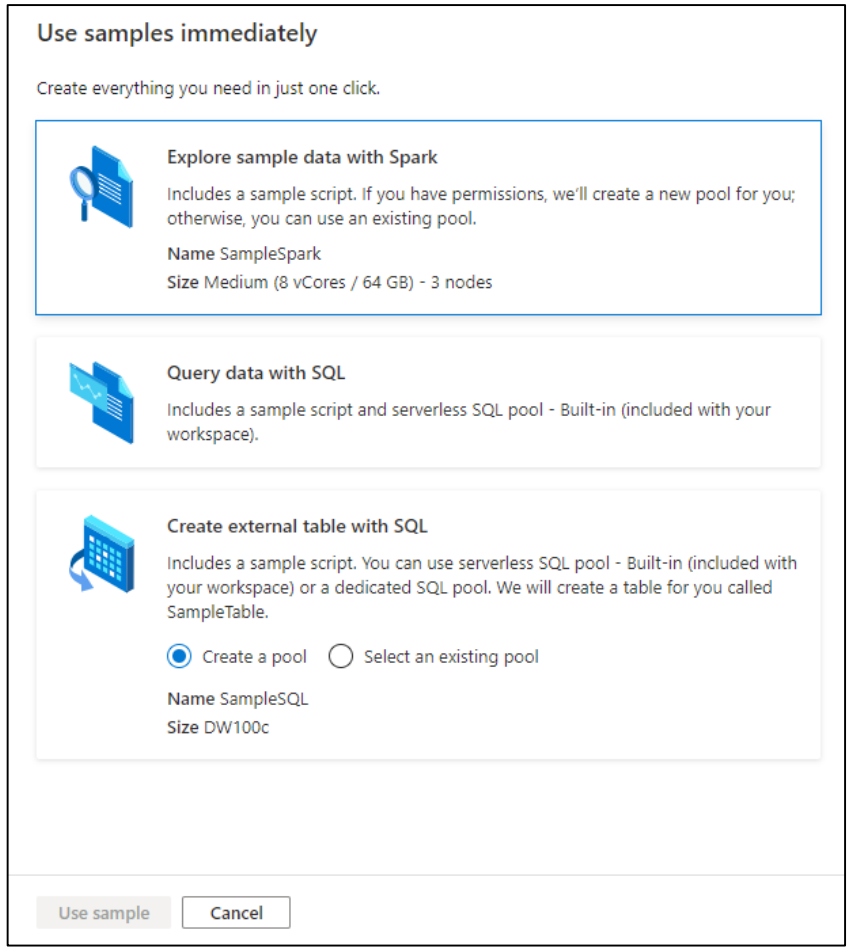
Knowledge Center는 사용자에게 학습 할 수 있는 샘플 데이터를 제공합니다.

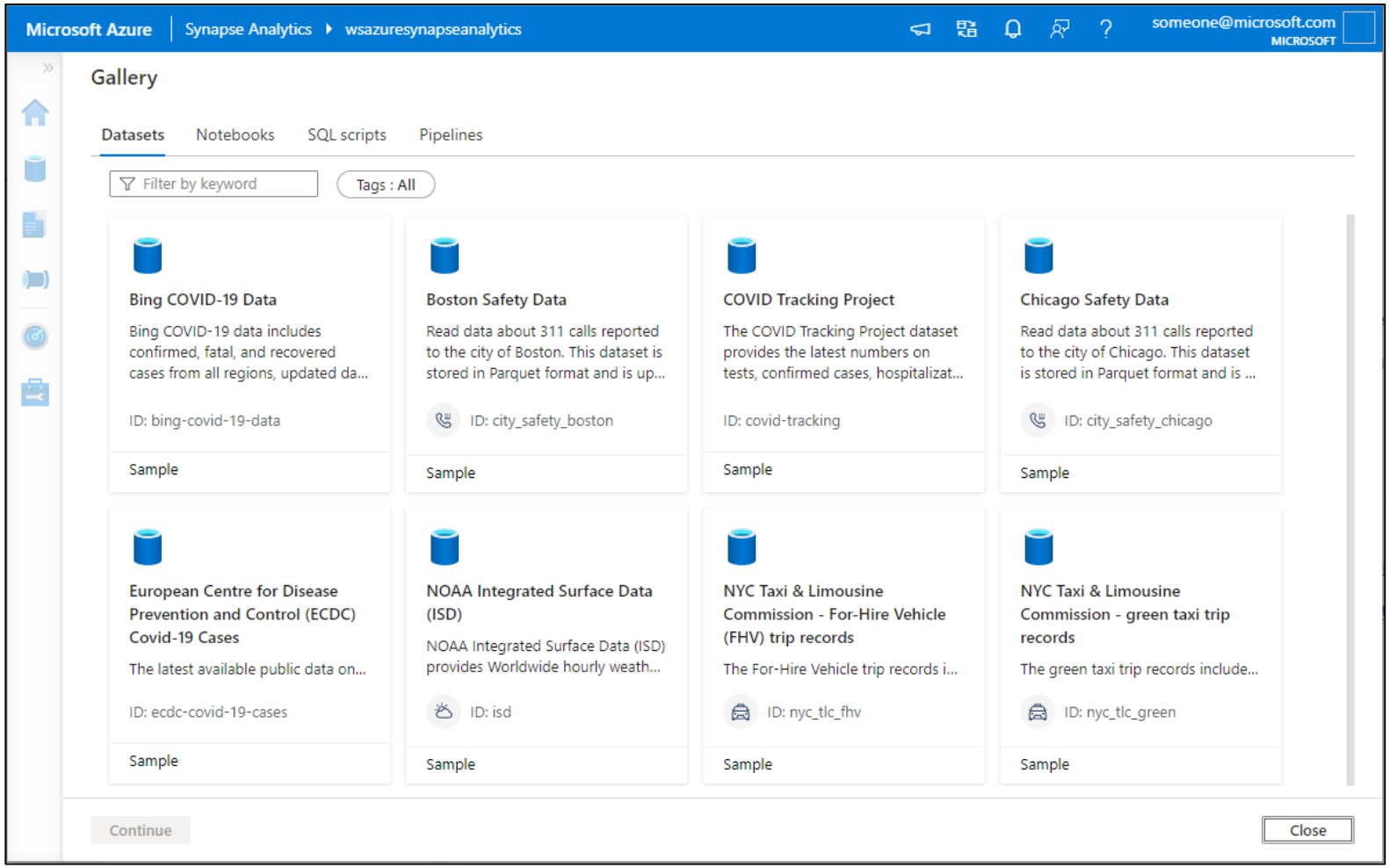




Knowledge center

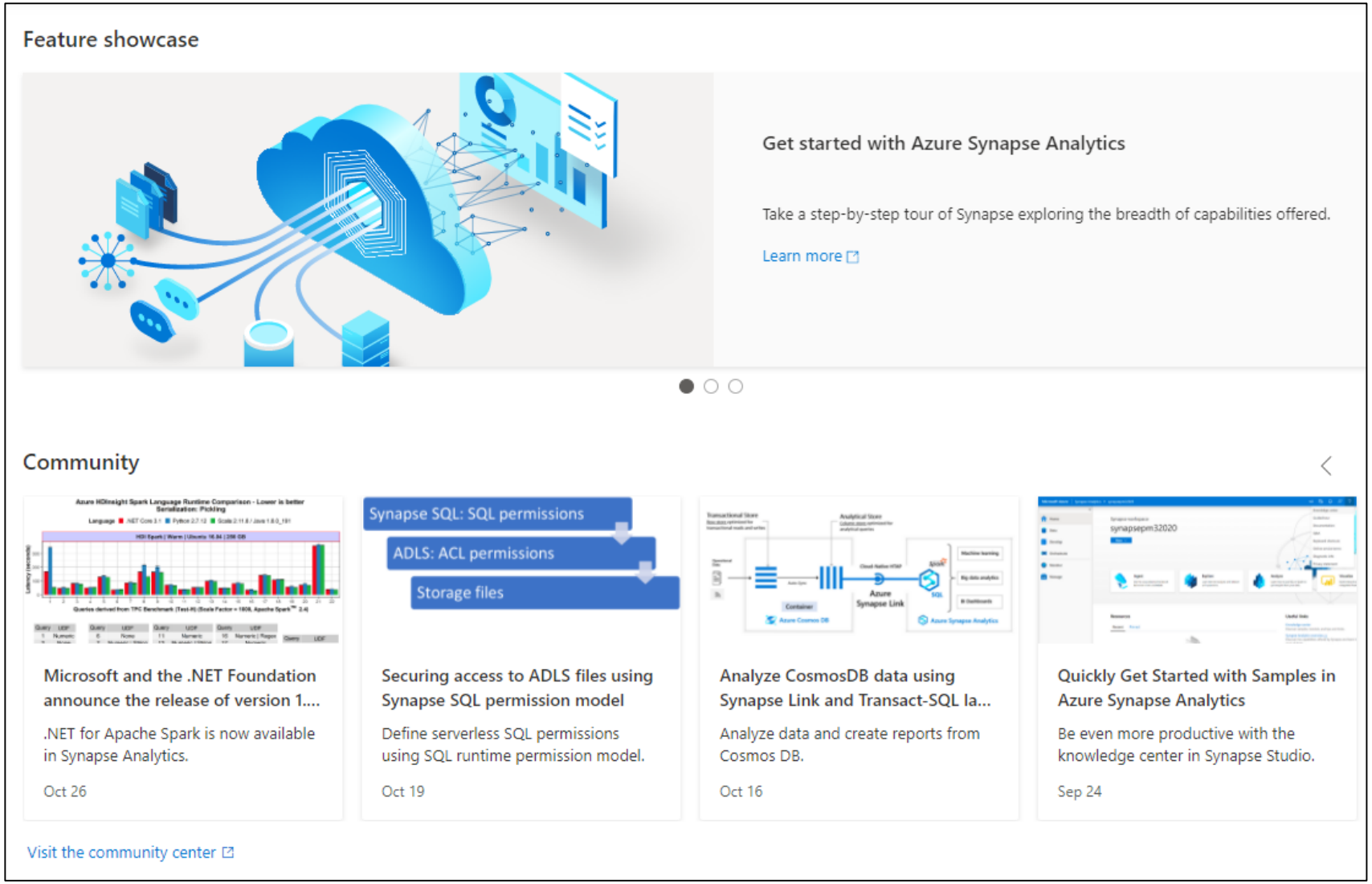
Knowledge Center는 간편한 시작과 학습을 위한 open datasets, sample Notebooks, SQL scripts and pipeline templates 을 제공합니다.





Home Hub

사용자가 Synapse Analytics의 기능을 학습할 수 있도록 기술 문서, 커뮤니티 센터로 이동하는 링크를 제공합니다.

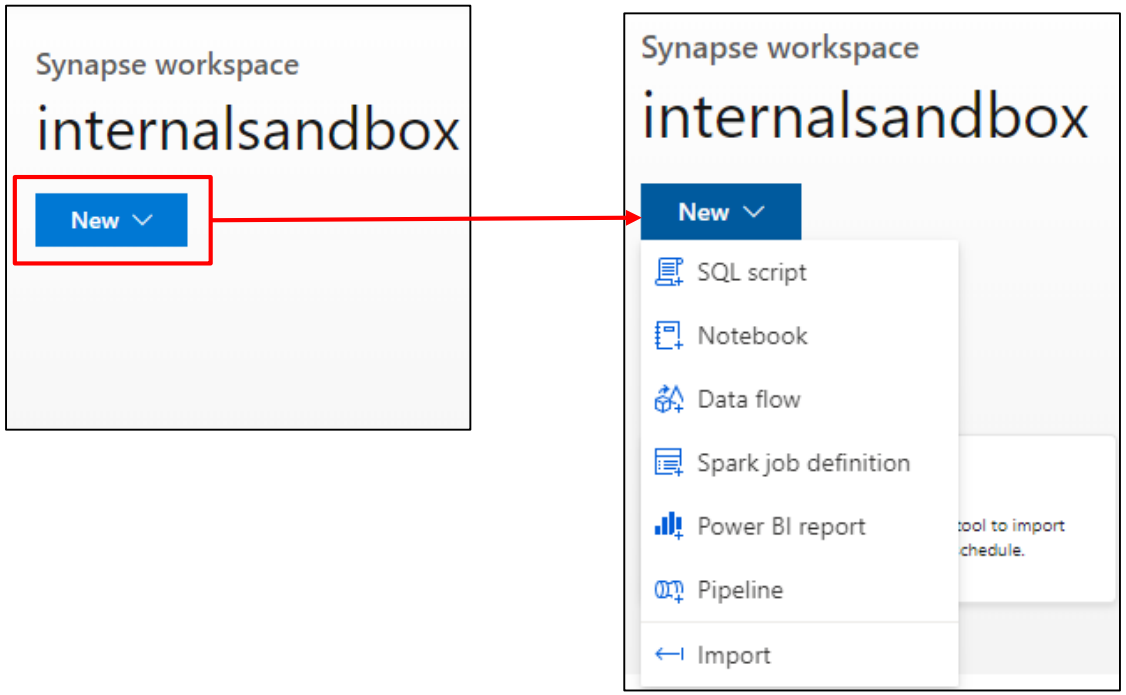


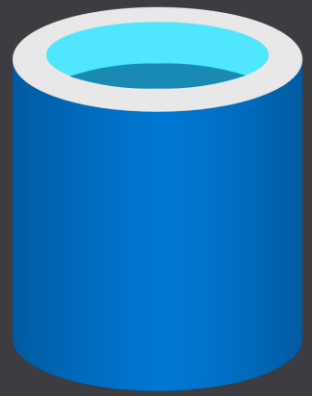
Home Hub

Overview

New dropdown – 빠른 작업을 시작할 수 있도록 단축 키를 제공 합니다.

Recent & Pinned – 최근에 사용했던 SQL Script, Spark Code 등 가공물 내역을 나열합니다.

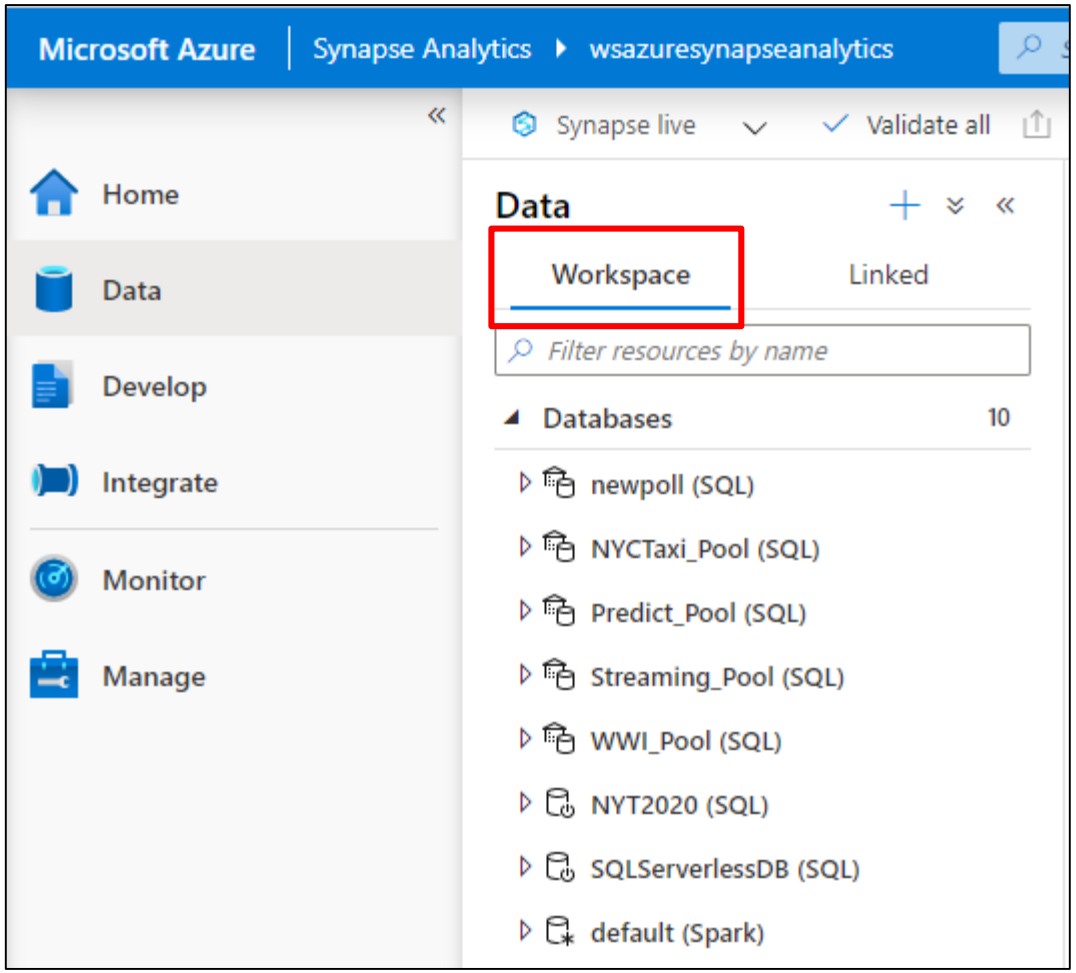


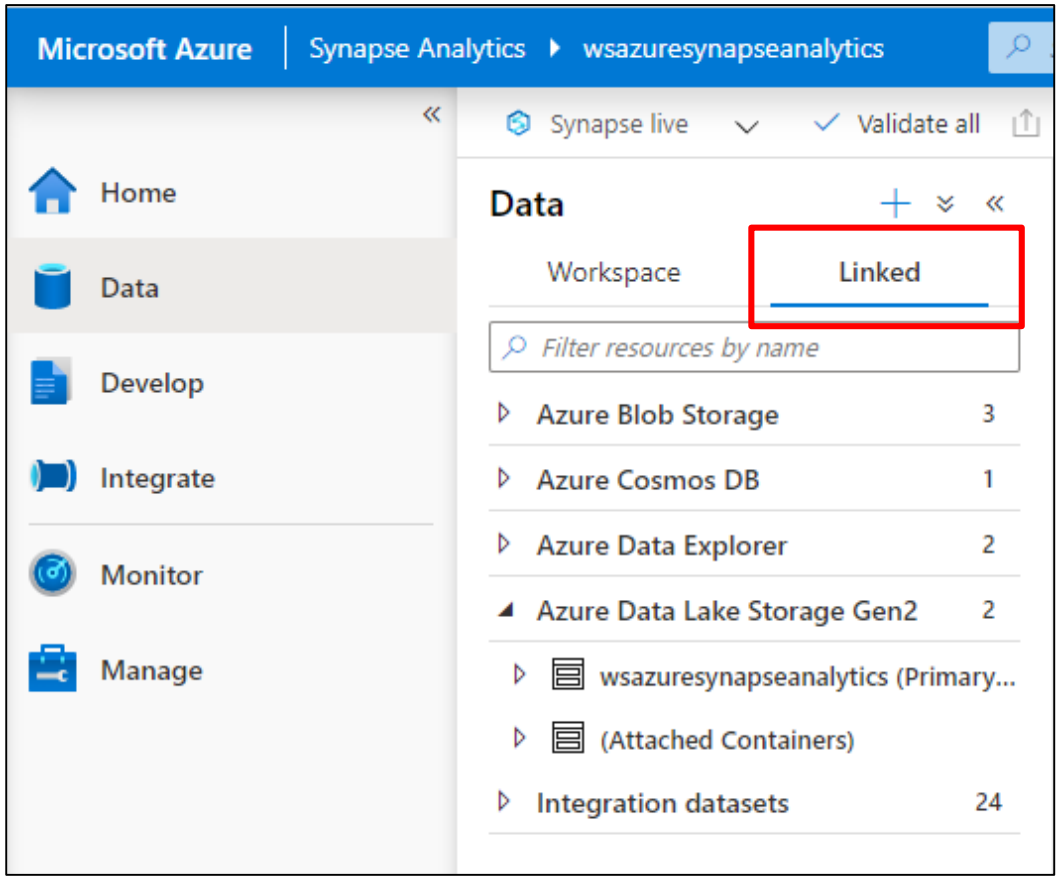


Synapse Studio
Data hub

Data Hub

Workspace의 내부 데이터와 Workspace에 Link 된 Storage Account의 데이터를 탐색할 수 있습니다.





Data Hub – Linked Storage

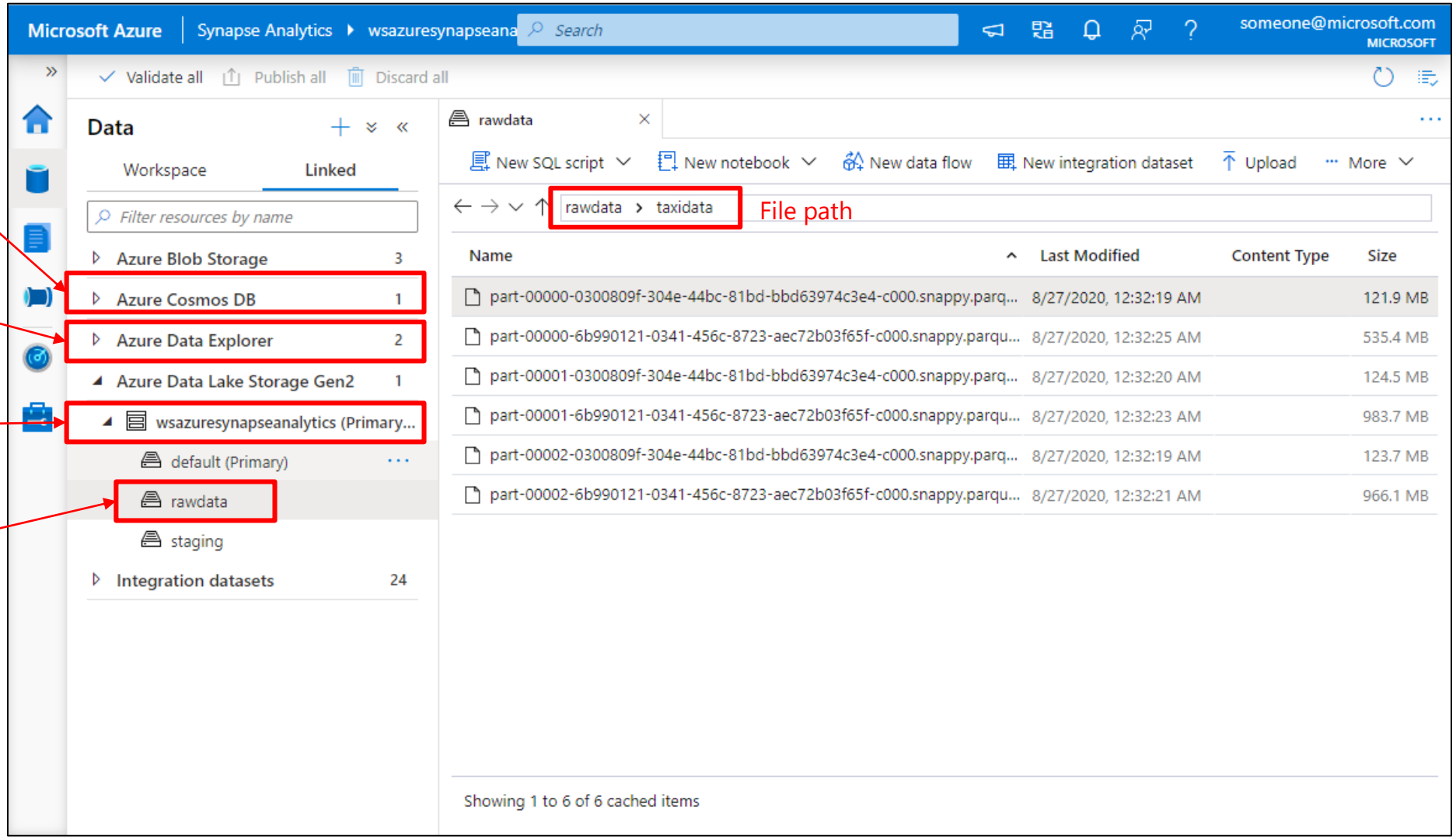
Azure Data Lake Storage Gen2 Accounts의 Filesystems, Azure Data Explorer 의 Clusters, Azure Cosmos DB의 Containers 데이터를 찾아볼 수 있습니다.

Linked Cosmos DB Analytical Store

Linked Azure Data Explorer

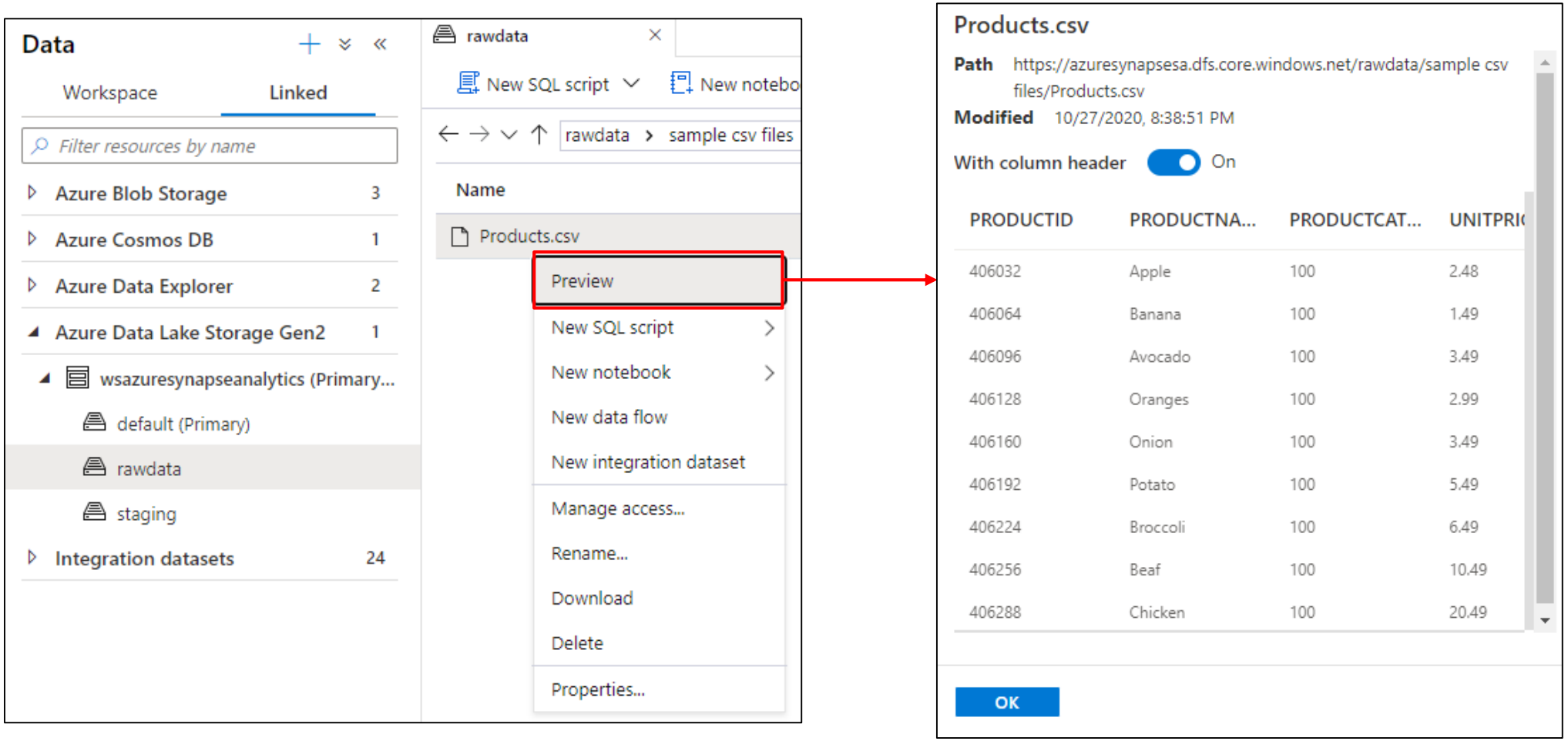
Linked ADLS Gen2 Account

Container (filesystem)



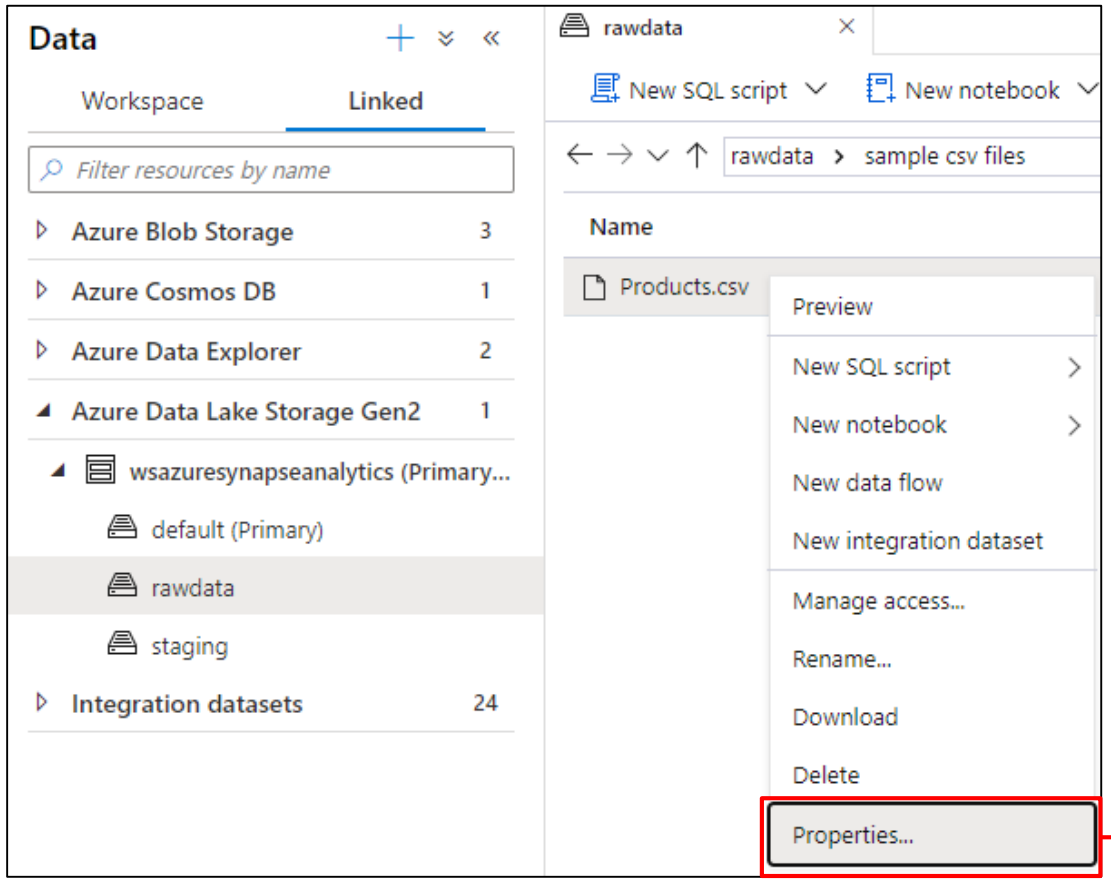
Data Hub – Storage accounts

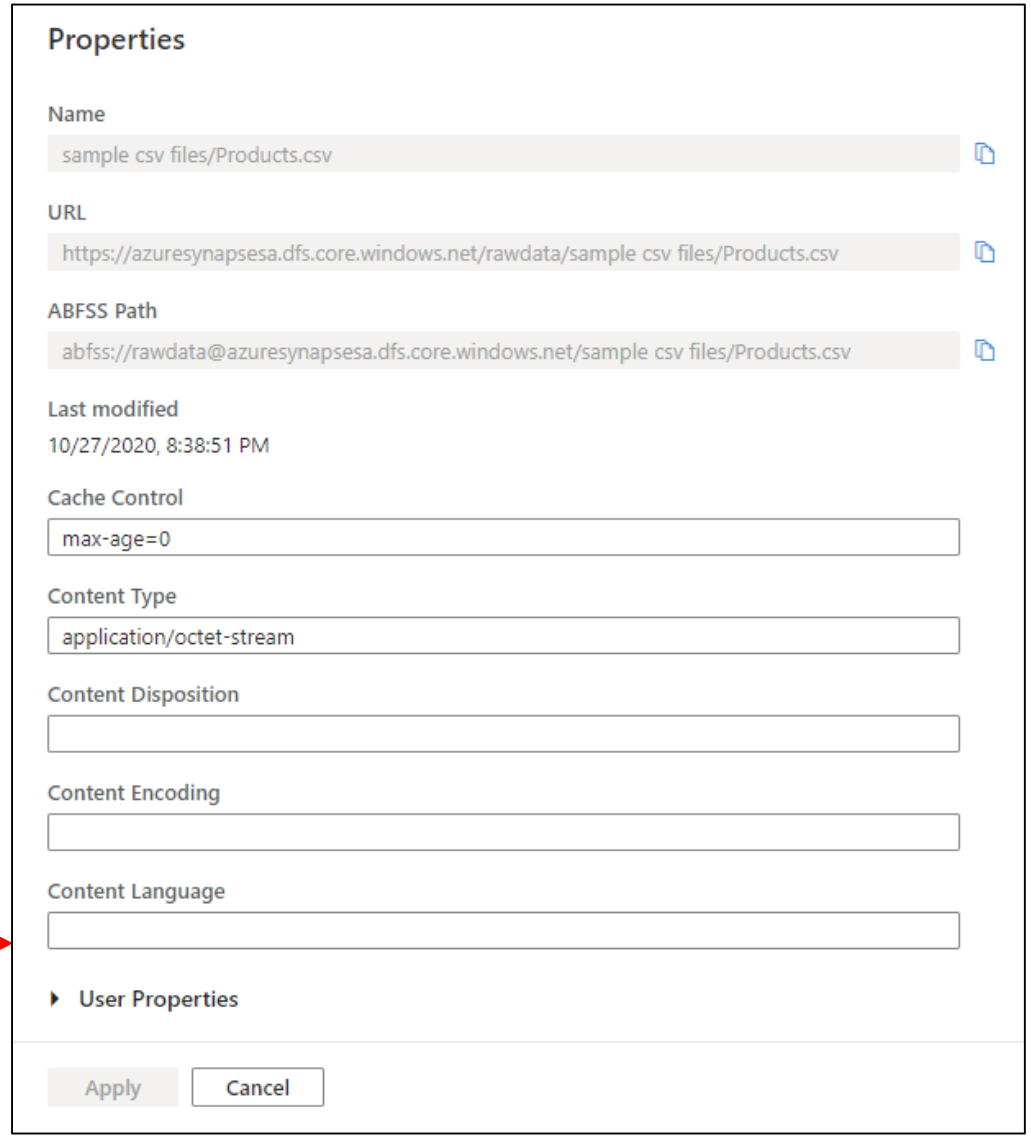
데이터 미리보기 기능을 지원합니다.



Data Hub – Storage accounts

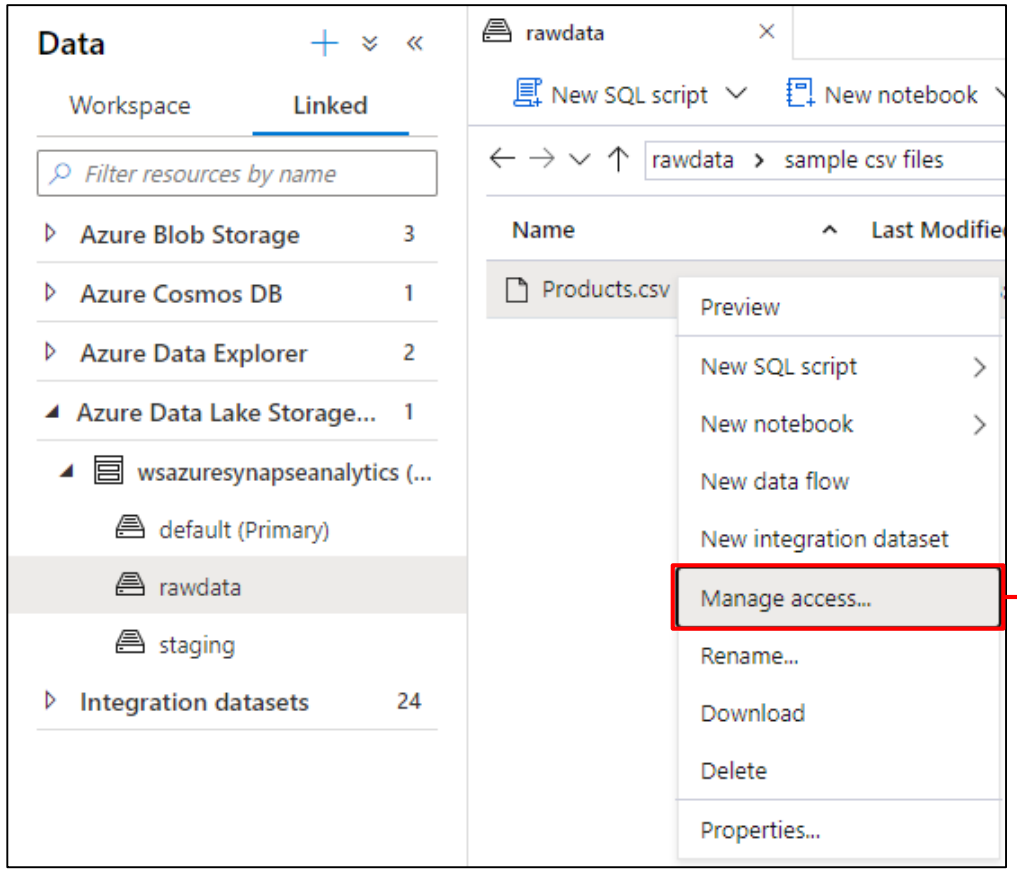
파일의 속성 확인이 가능합니다.

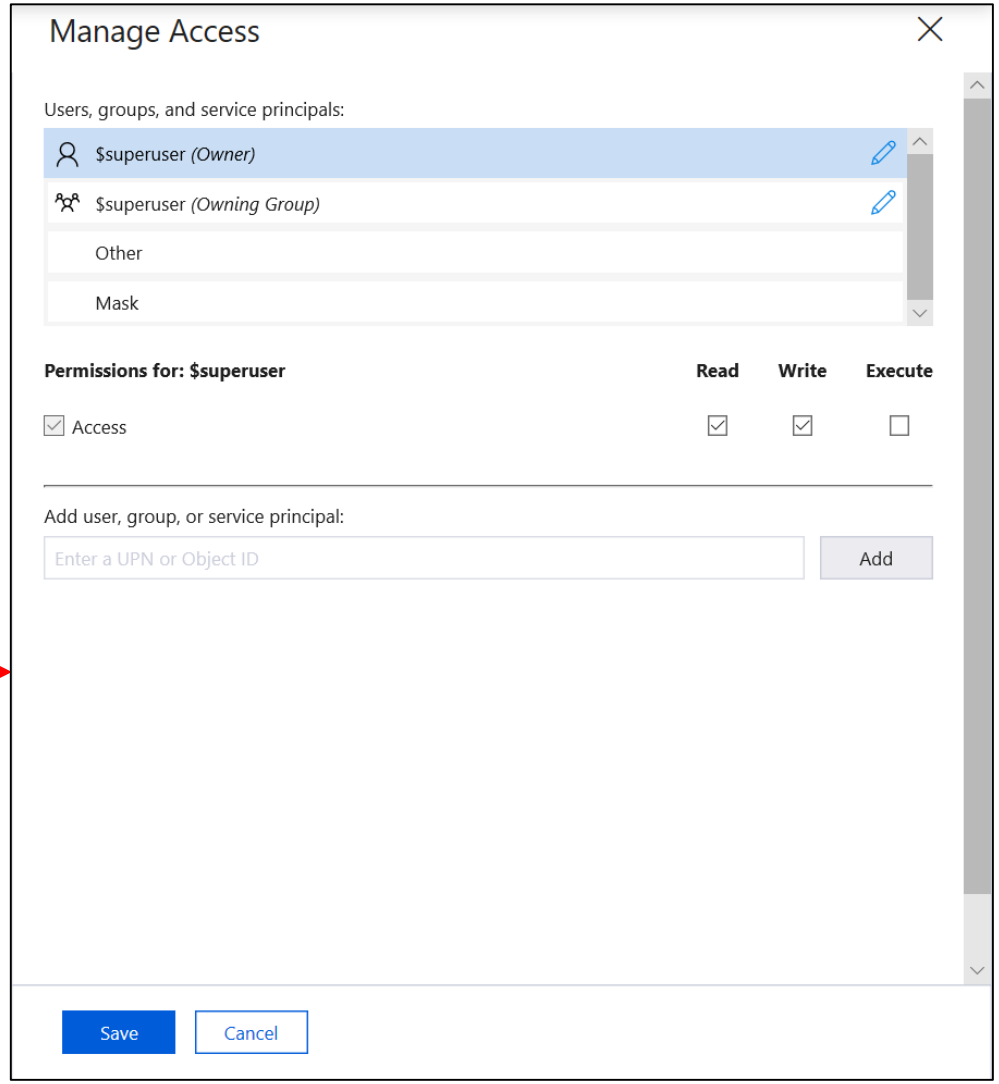




Data Hub – Storage accounts

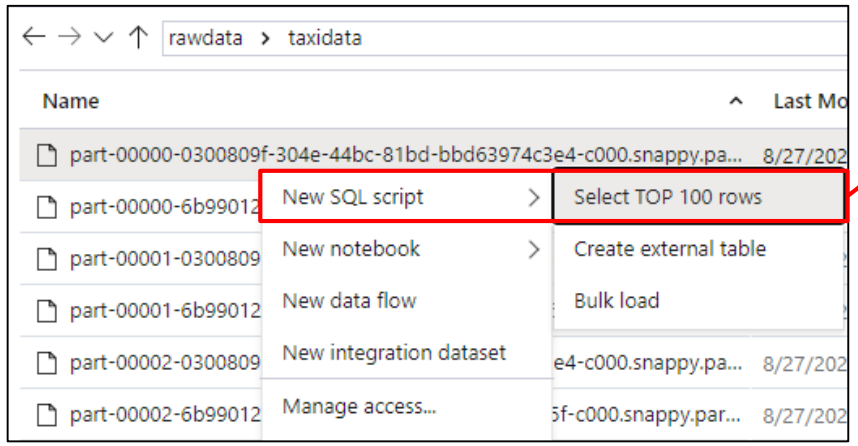
접근 관리 기능 - 폴더와 파일에 관한 standard POSIX ACLs 설정이 가능합니다.

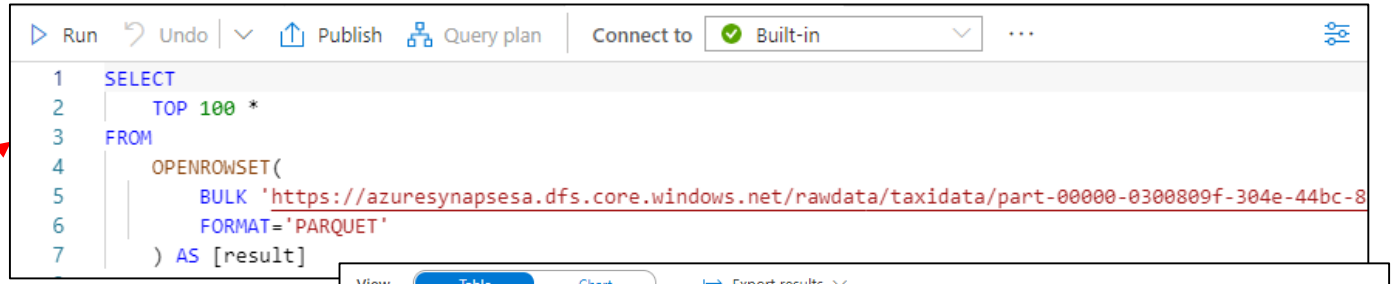


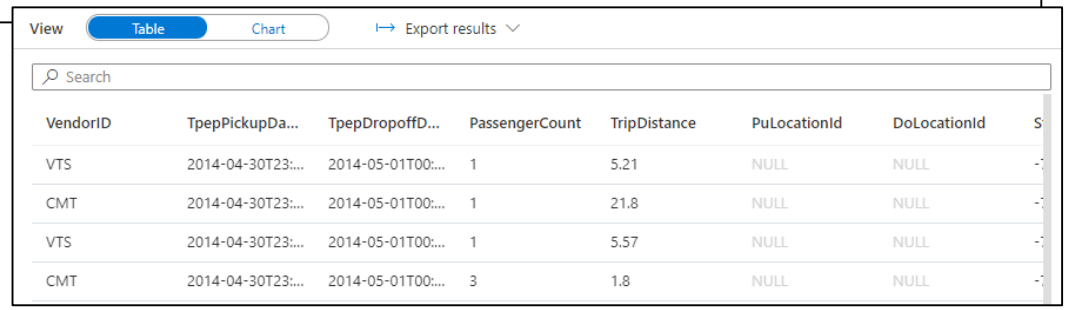


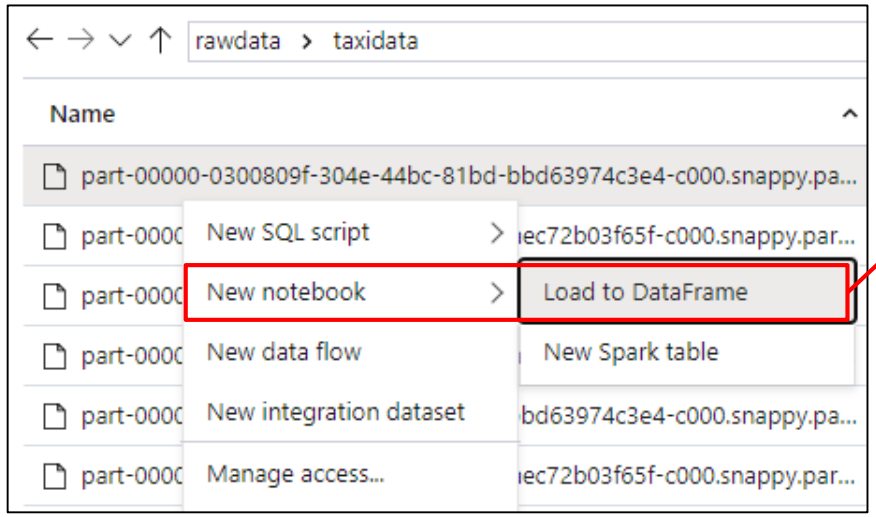
Data Hub – Storage accounts

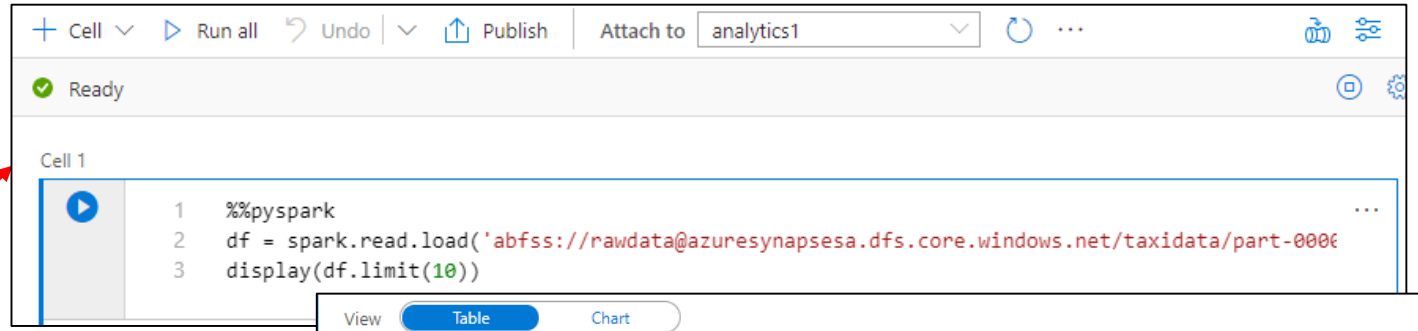
Storage Account 내부 파일의 데이터를 조회하기 위해 Studio 에서는 SQL scripts 과 Notebook code 를 자동 생성하는 기능이 있습니다.

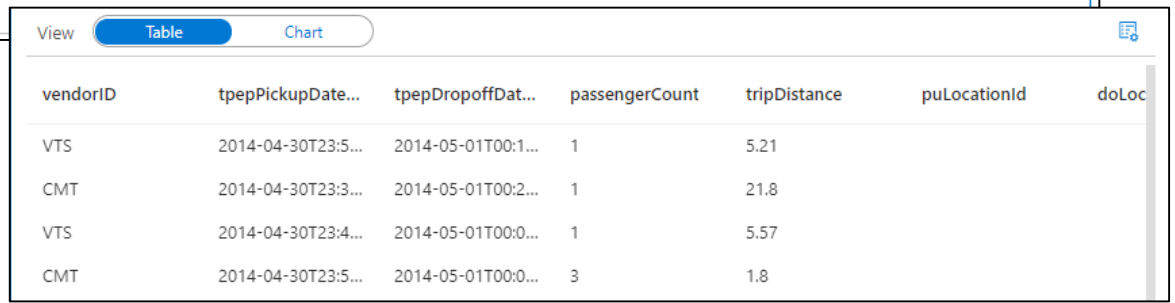






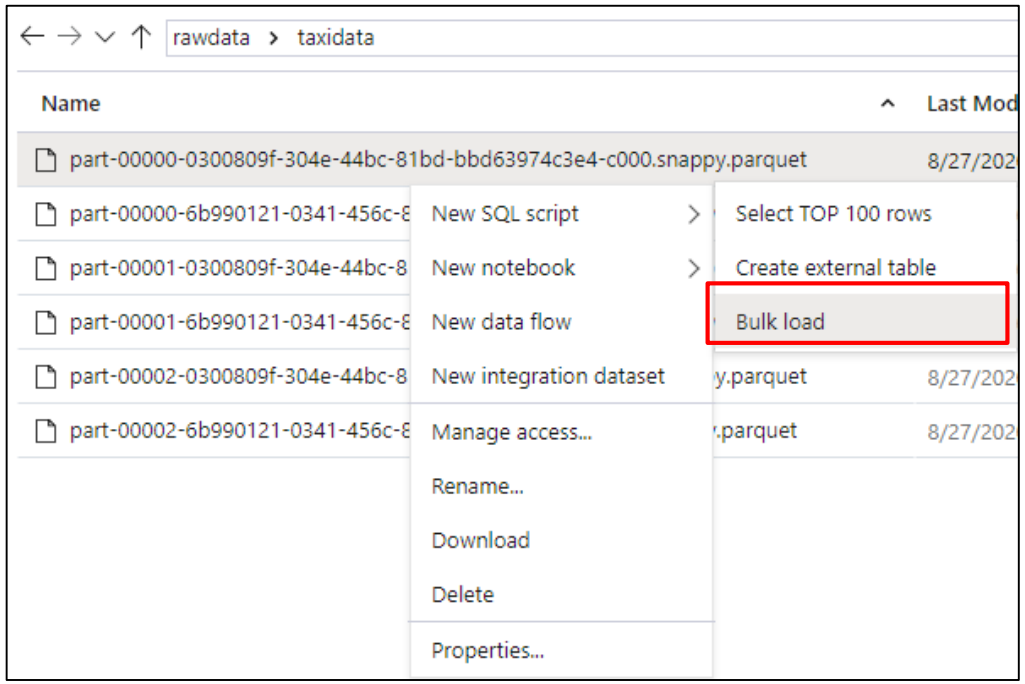


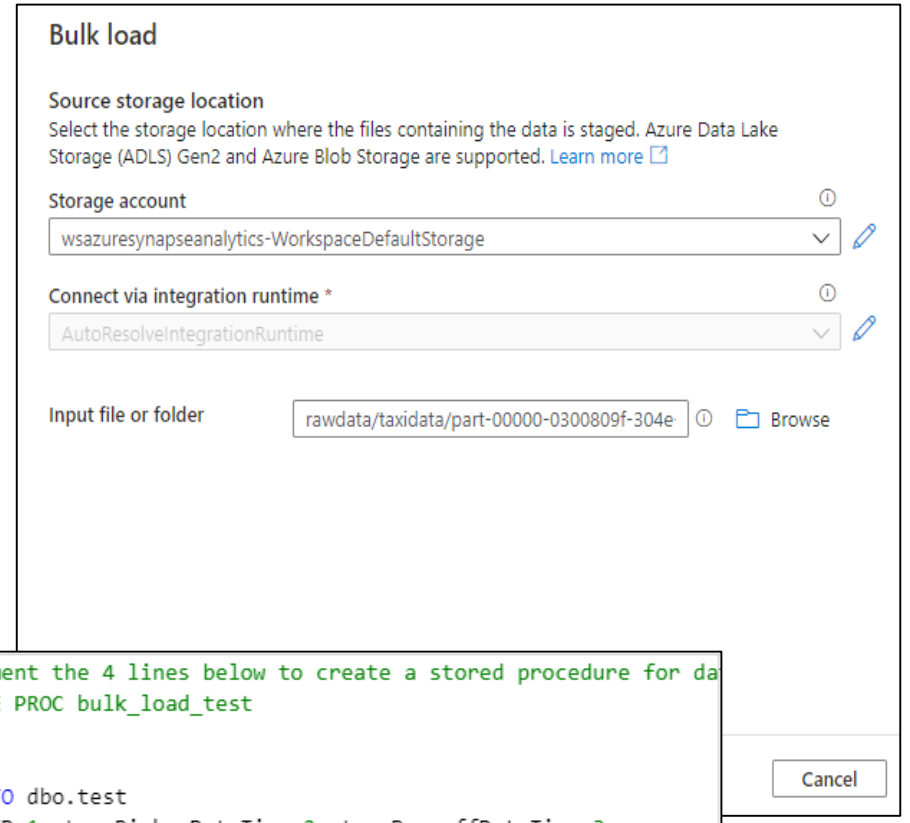




Data Hub – Bulk load wizard

- Bulk Load Wizard 기능을 사용하여 Storage Container 또는 Storage File 를 활용한 Data의 Bulk Load가 가능합니다.
- Data load를 위해 자동 T-SQL script 생성을 지원합니다.
- 파이프라인과 통합을 가능하게 하는 Stored Procedure 생성을 제공합니다.

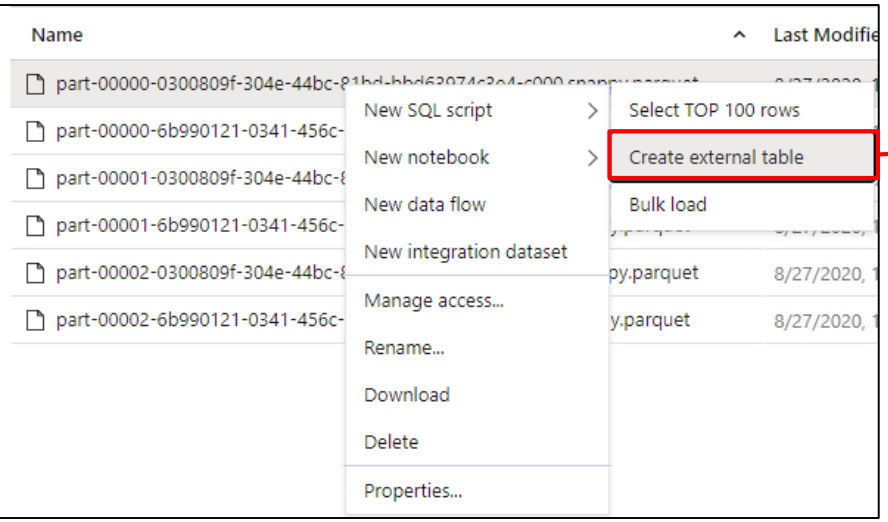


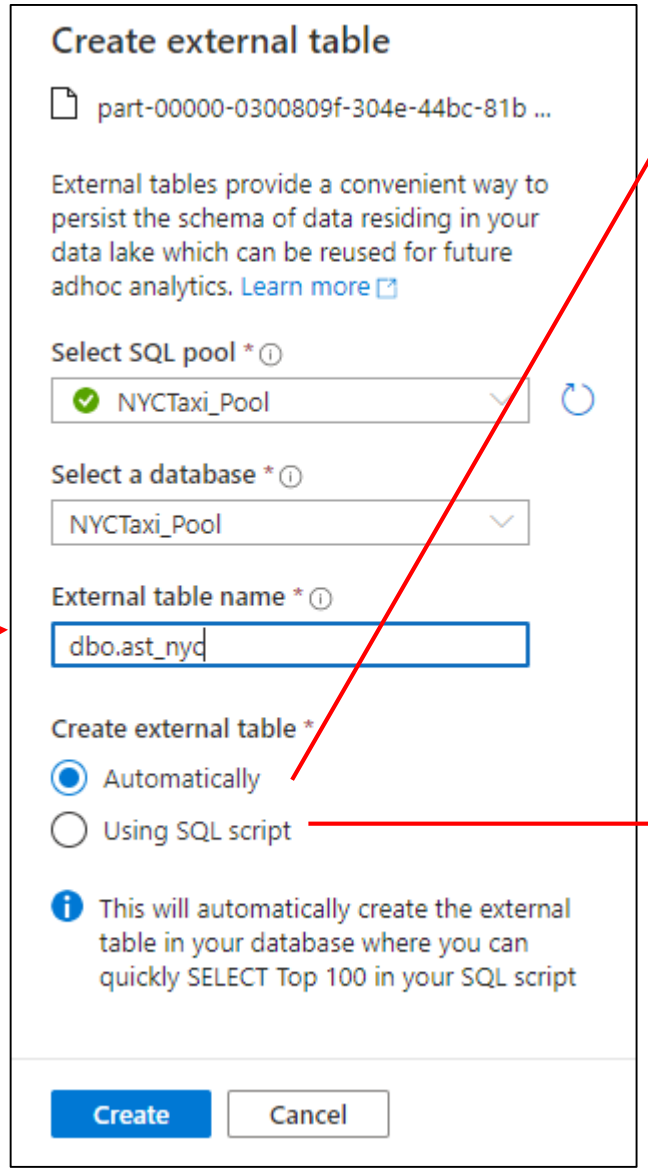


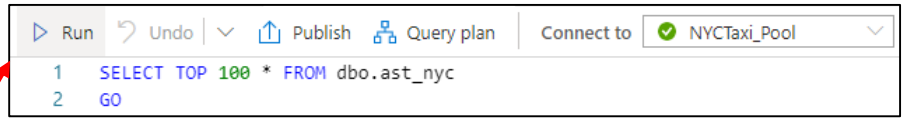

```
1  --Uncomment the 4 lines below to create a stored procedure for da
2  --CREATE PROC bulk_load_test
3  --AS
4  --BEGIN
5  COPY INTO dbo.test
6  (vendorID 1, tpepPickupDateTime 2, tpepDropoffDateTime 3, passeng
7  FROM 'https://azuresynapsesa.dfs.core.windows.net/rawdata/taxidat
8  WITH
9  (
10     FILE_TYPE = 'PARQUET'
11     ,MAXERRORS = 0
12     ,IDENTITY_INSERT = 'OFF'
13 )
14 --END
15 GO
16
17 SELECT TOP 100 * FROM test
18 GO
```

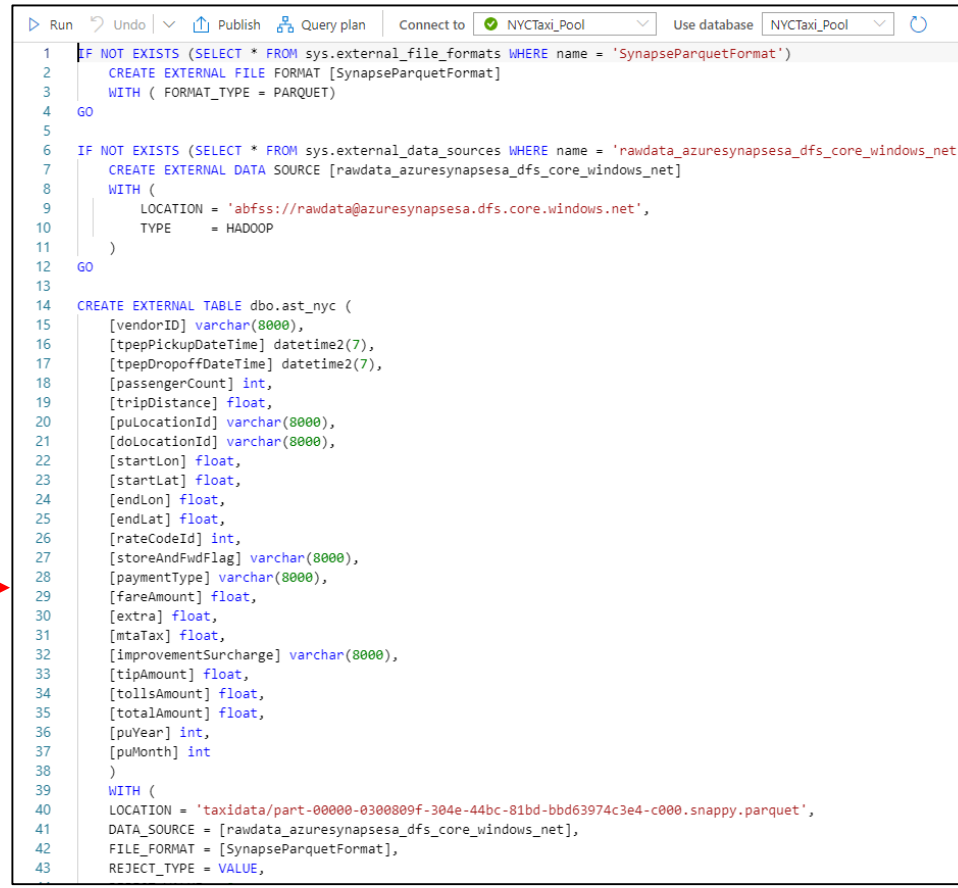
Data Hub – Storage accounts

- External File Format, External File 원본 위치 지정 및 External Table을 자동으로 만들어 생산성을 향상 시킬 수 있습니다.
- External File Format, External File 원본 위치 및 External Table을 생성하는 스크립트를 제공하여 수동으로 실행 가능합니다.



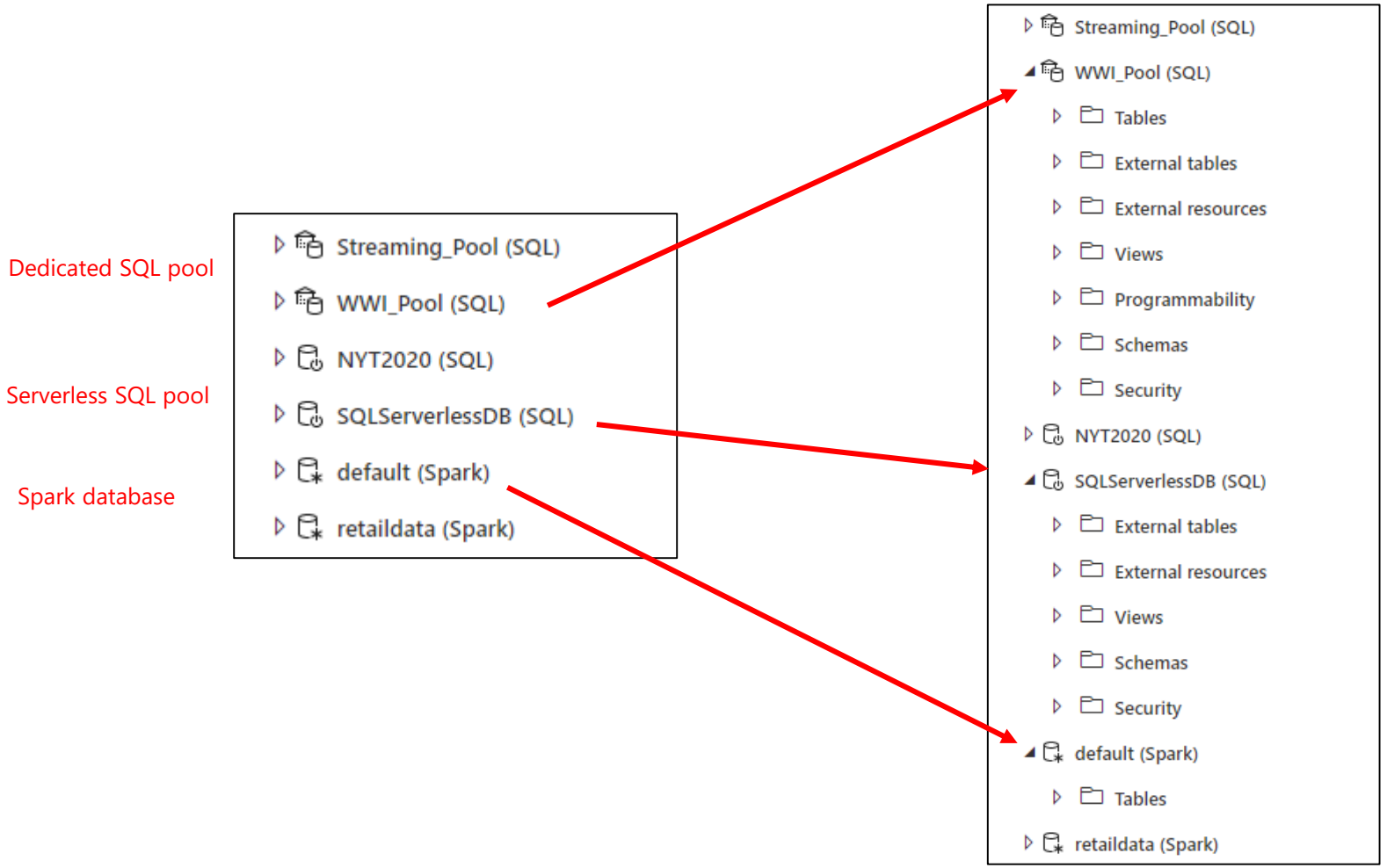






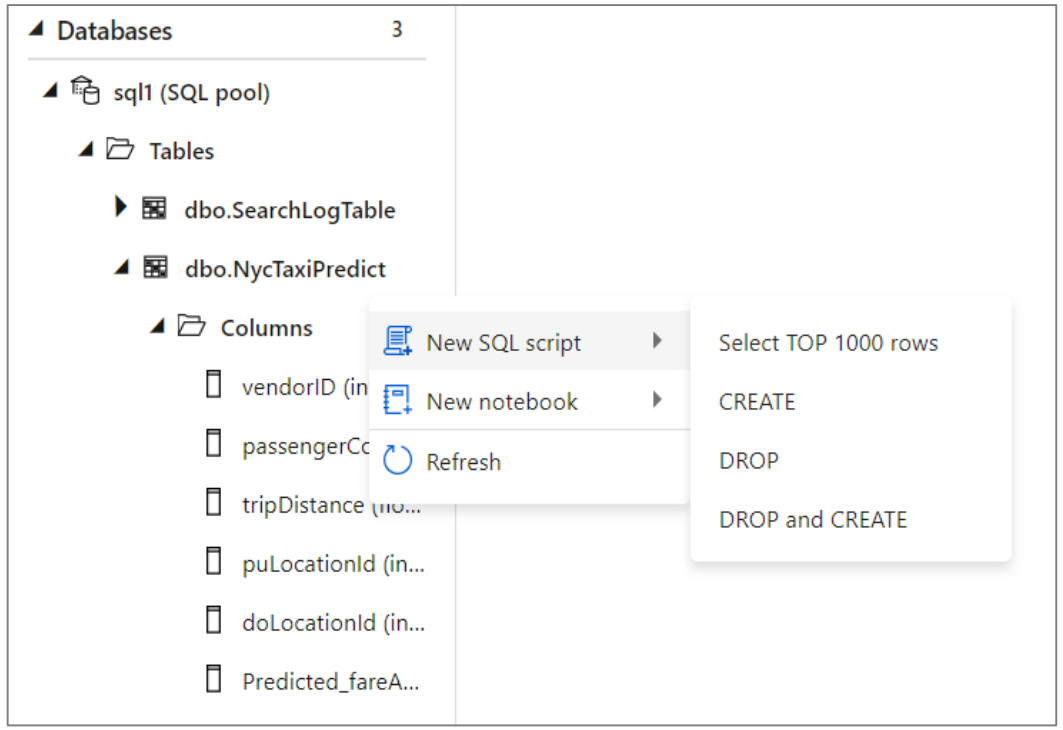
Data Hub – Databases

Workspace 내부에 존재하는 데이터베이스를 탐색합니다.

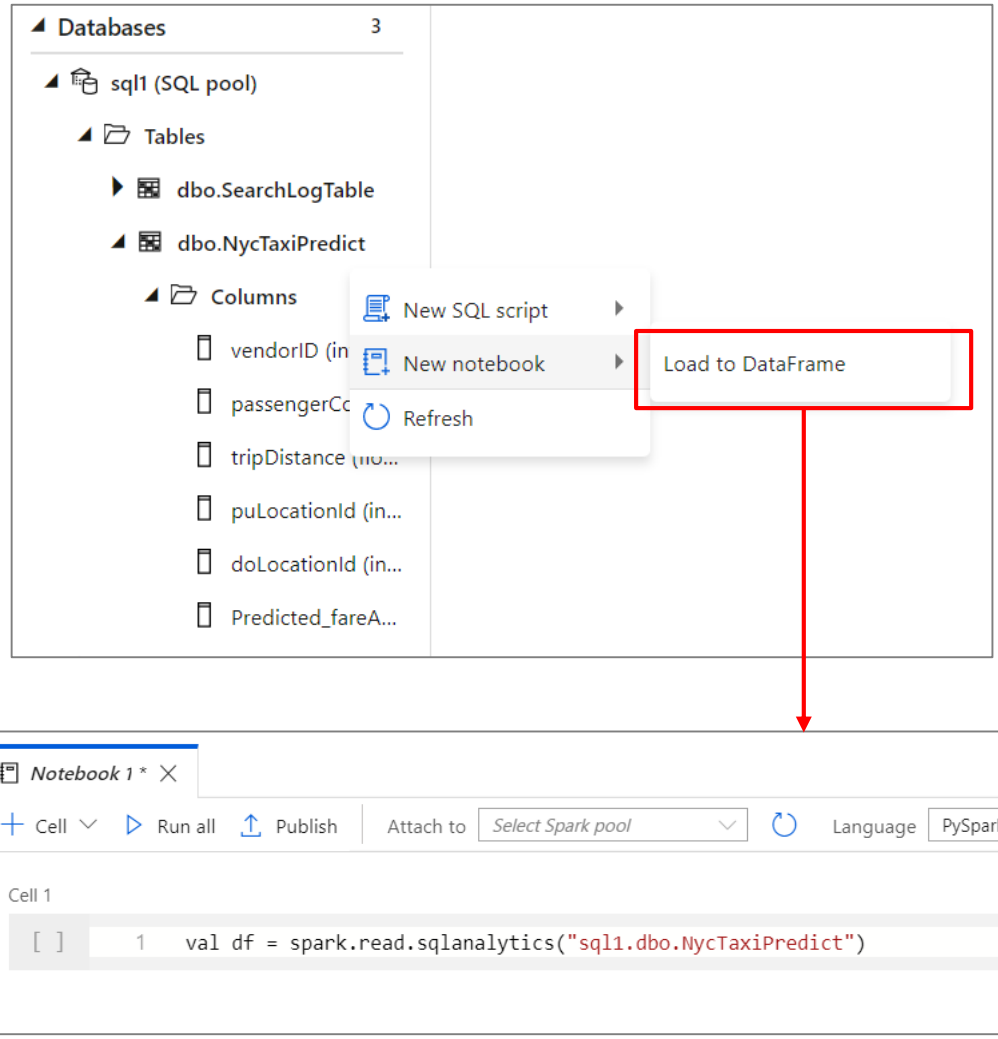


Data Hub – Databases

- 테이블과 같은 Object 에서 간단히 데이터를 조회할 수 있는 T-SQL Scripts 를 생성할 수 있습니다.

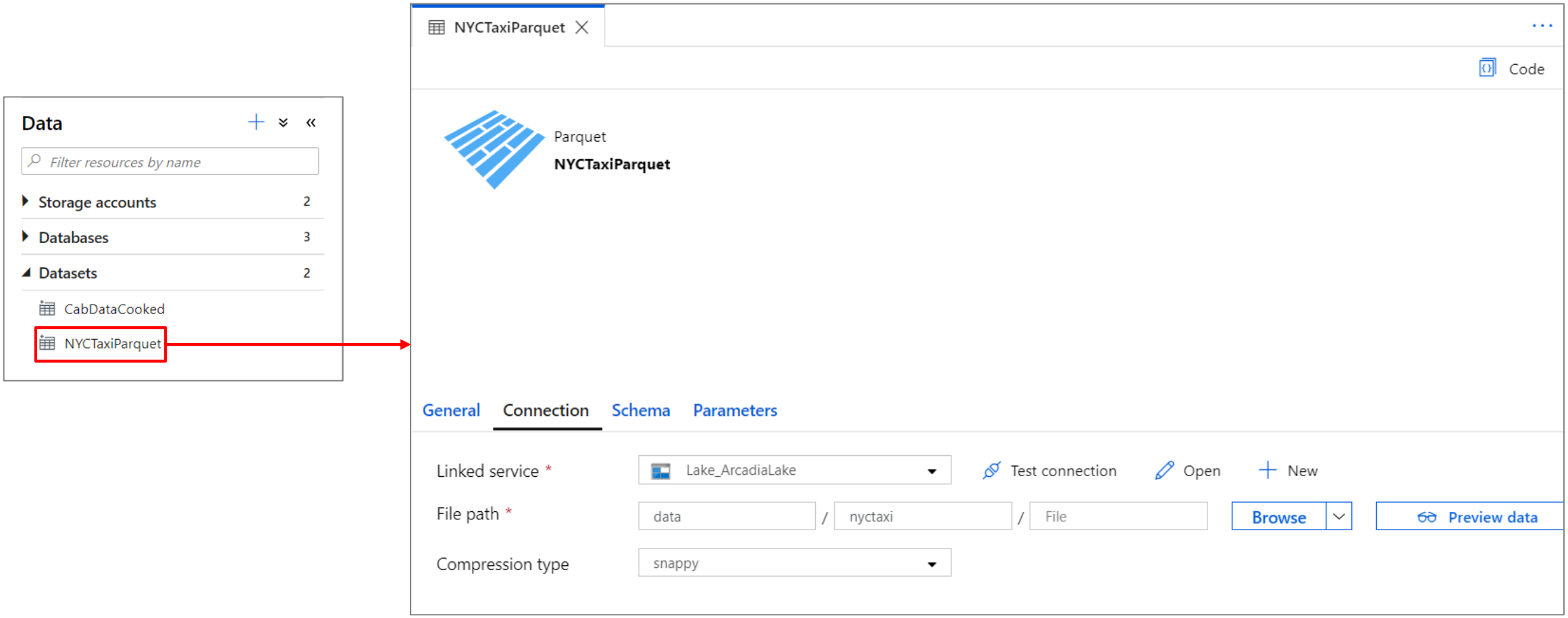


- SQL Pool 의 Table 에서 Spark DataFrame 으로 데이터를 조회할 수 있는 PySpark Code 를 자동 생성할 수 있습니다.



Data Hub – Datasets

데이터 세트가 정의되면 파이프라인에서 데이터 소스 또는 데이터 싱크로 사용할 수 있습니다.



Synapse Studio
Develop hub

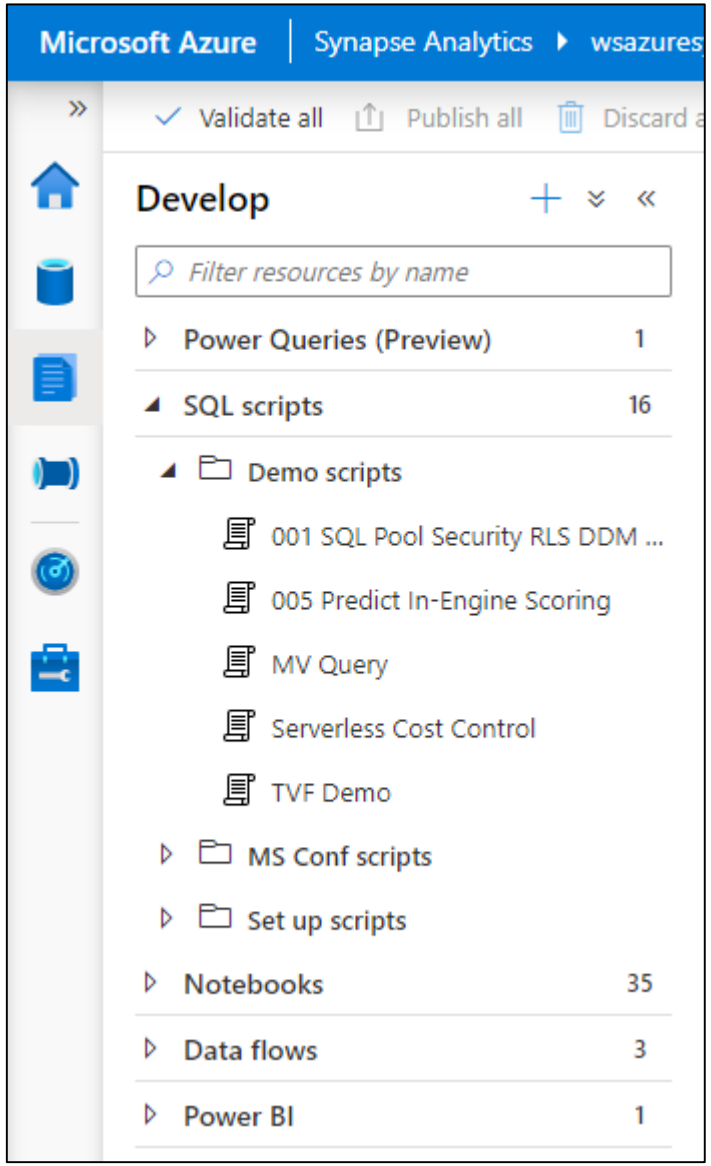
Develop Hub

Overview

Data Hub는 쿼리, 분석, 모델 데이터에 관한 개발 환경을 제공합니다.

Benefits

- 개발 Hub에서 다양한 언어를 사용하여 데이터 분석이 가능합니다.
- SQL Script와 Notebook Code를 손쉽게 전환하며 분석할 수 있습니다.
- SQL Script와 Notebook Code 작성에 최적화되어 있습니다.
- 데이터를 시각화하여 간단하게 볼 수 있습니다.

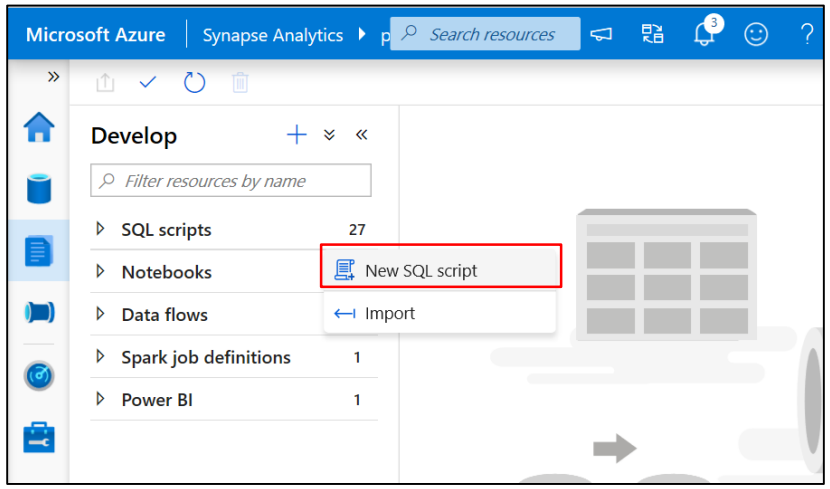


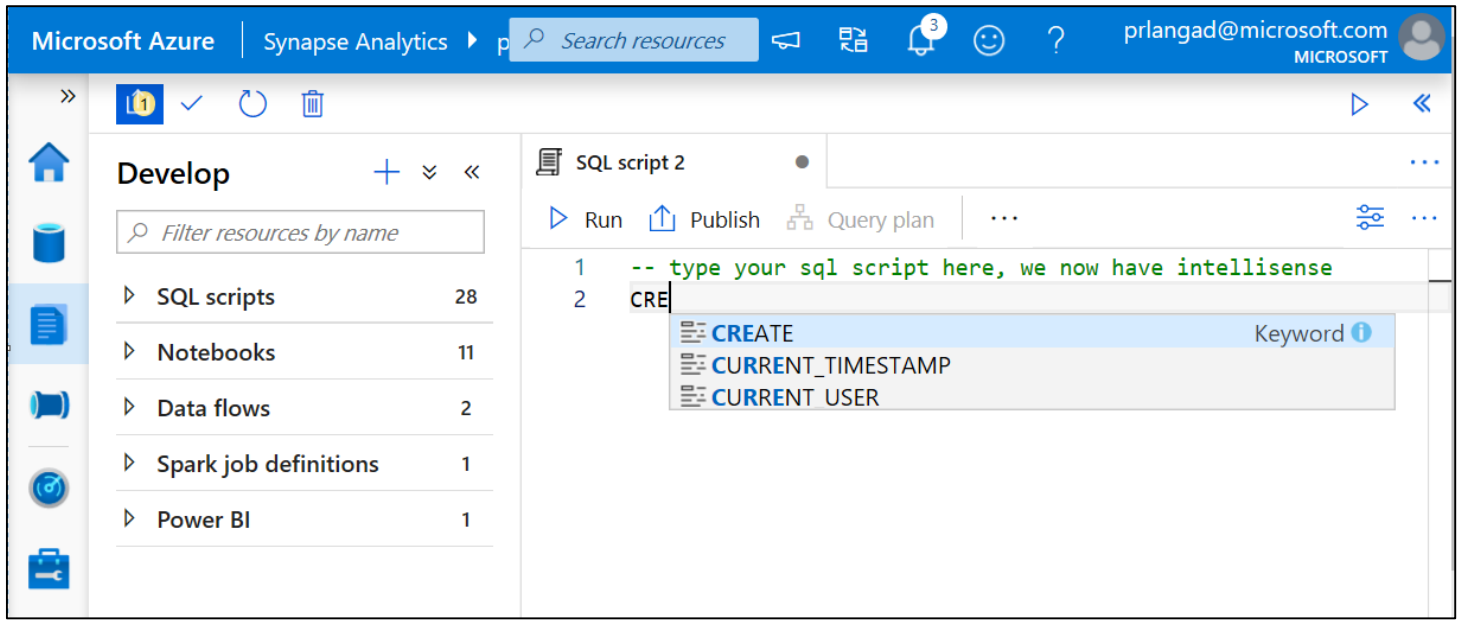
Develop Hub - SQL scripts

SQL Script

전용 SQL 풀 서버와 Serverless SQL 풀에서 실행 가능합니다.

단일 SQL 스크립트나 여러 개의 SQL 스크립트를 동시에 사용 가능합니다.

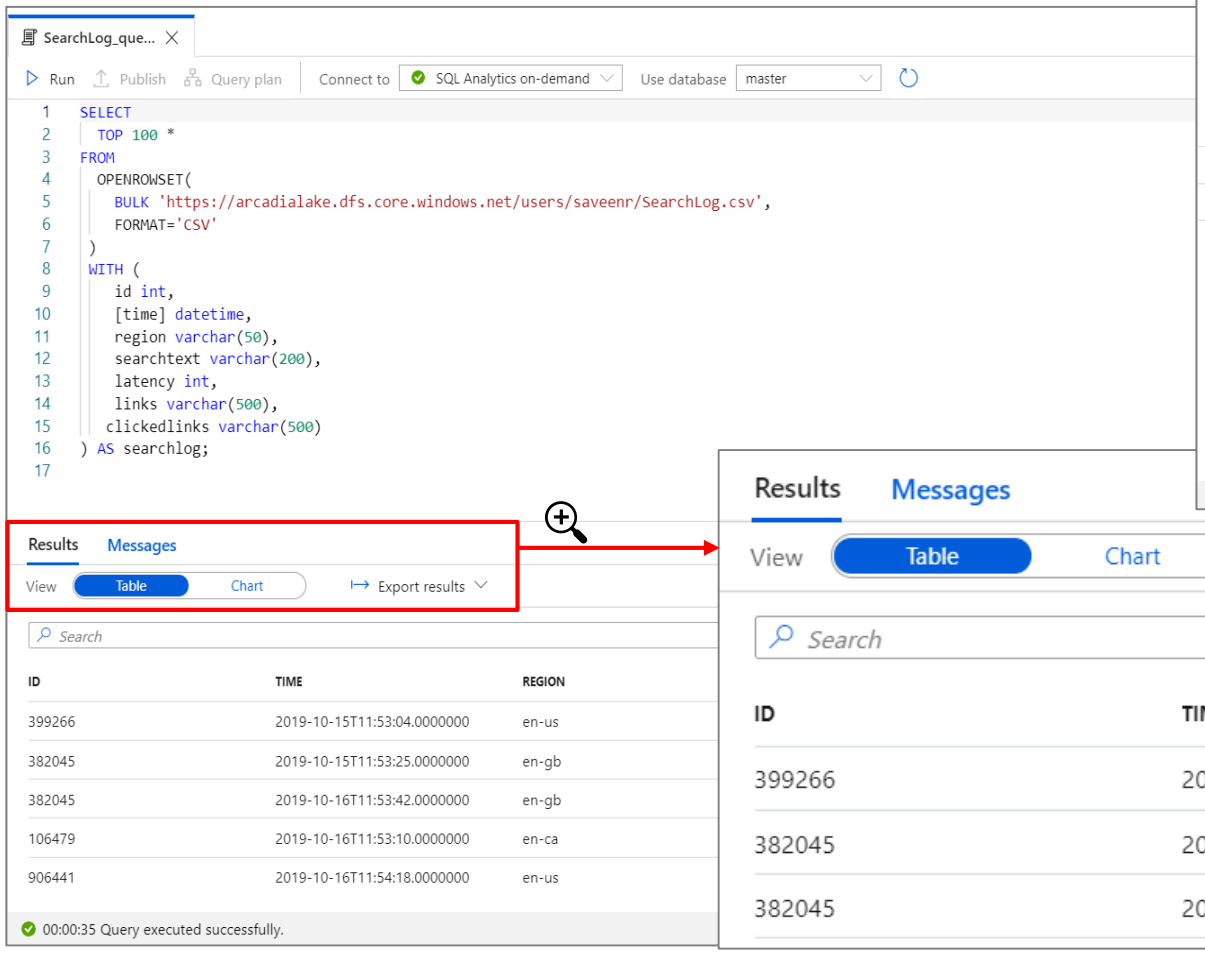


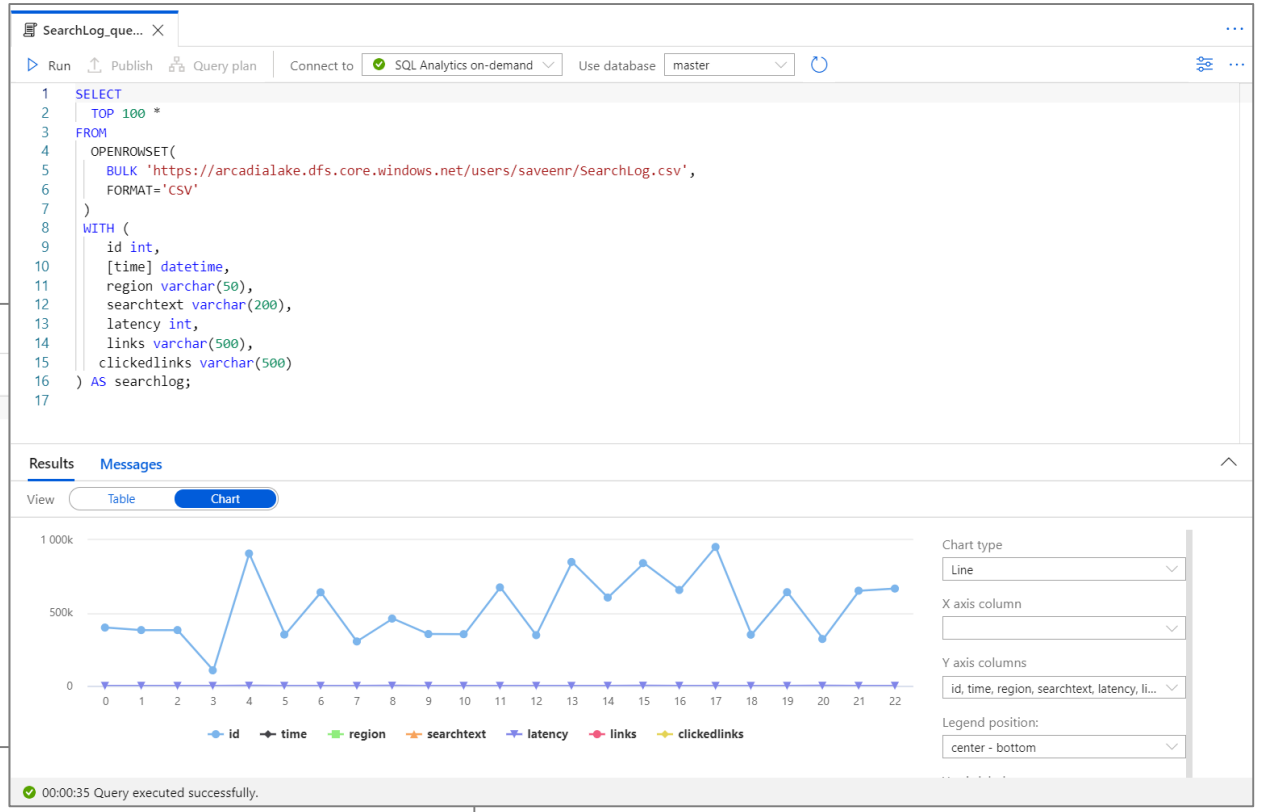


Develop Hub - SQL scripts

SQL Script

테이블 또는 차트 형식 안의 결과값을 볼 수 있으며 CSV나 JSON 파일로 추출할 수 있습니다.





Develop Hub - Notebooks

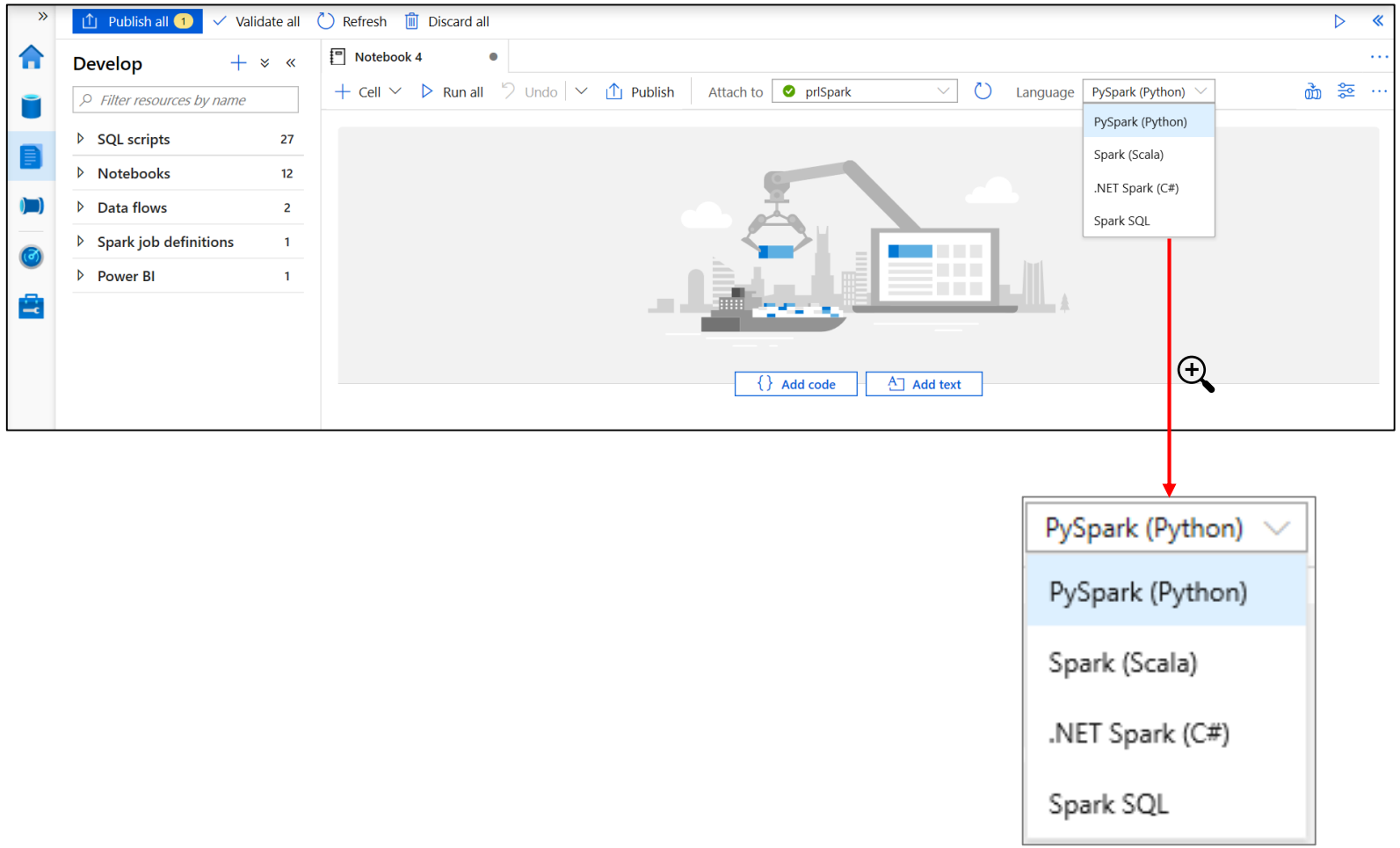
Notebooks

Notebook Code로 Python, Scala, C#, Spark SQL과 같은 언어 사용이 가능합니다.

해당 언어들을 통해 임시 테이블 생성을 제공합니다.

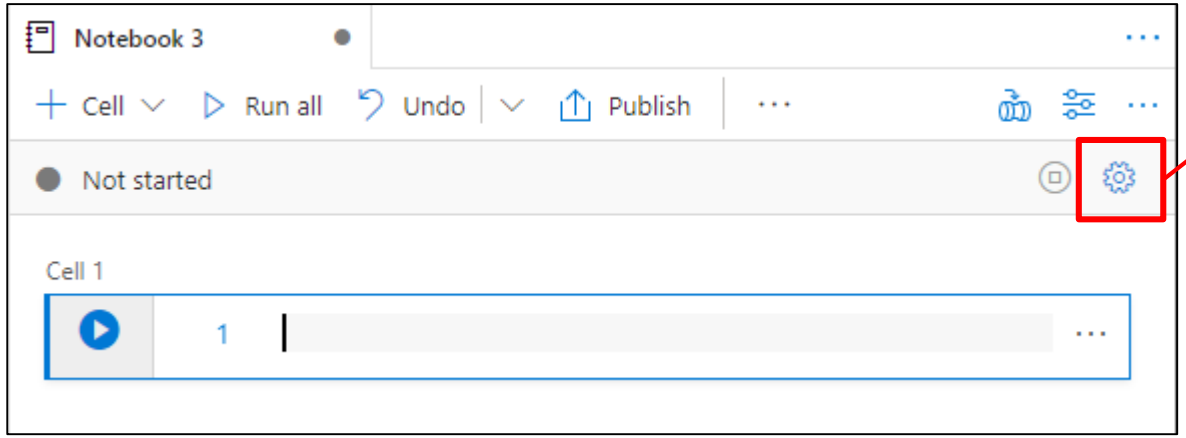
Syntax 문법 에러, 들여 쓰기 등의 코드 작성시 편의성을 제공합니다.

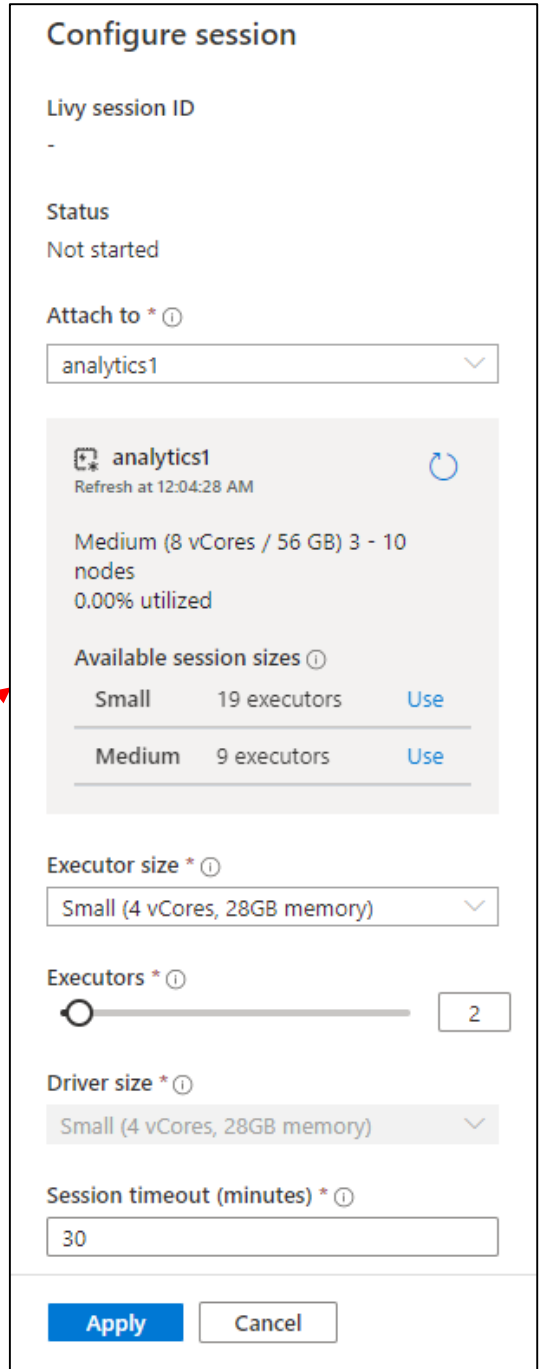
결과값을 추출할 수 있습니다.



Develop Hub - Notebooks

Configure Session은 개발자가 Notebook을 사용하는데 인프라 영역의 리소스 사용량을 늘리거나 줄여 관리하는 기능입니다.

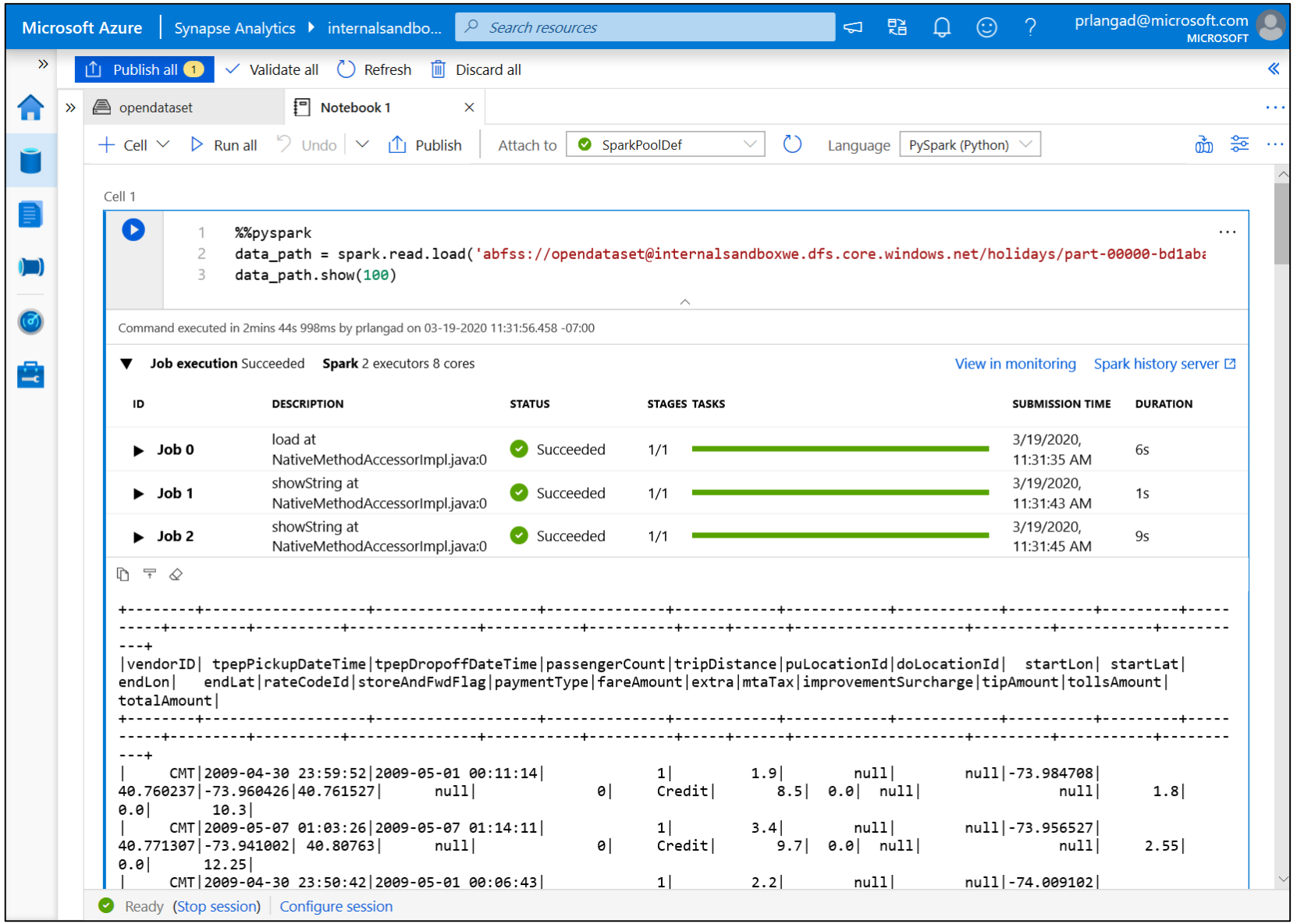




Develop Hub - Notebooks

Notebook에 있는 Cell을 실행하면, Spark 애플리케이션 상태가 아래로 보여지게 됩니다.

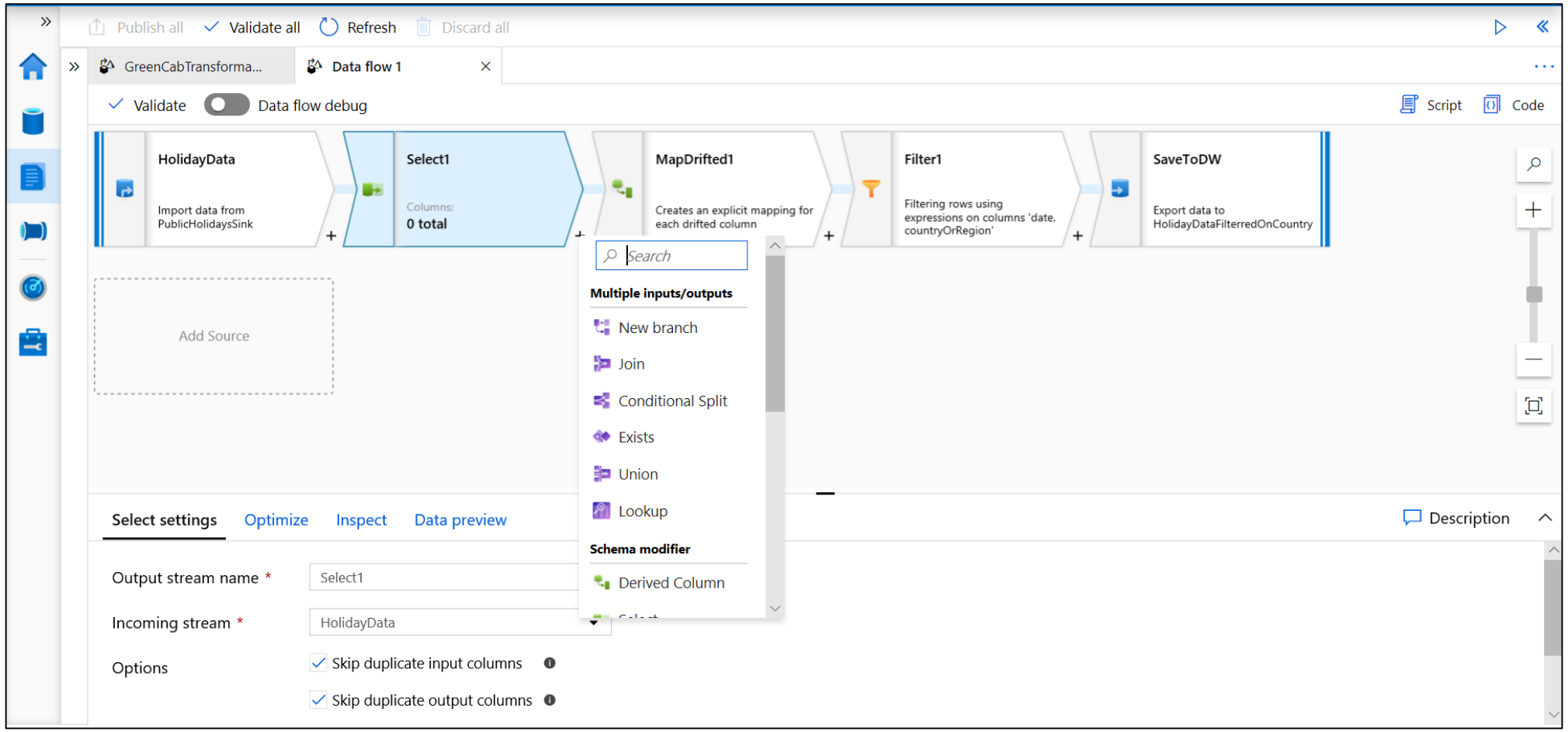
어플리케이션 상태를 확인하여 즉각적인 피드백과 진행 상황을 확인할 수 있습니다.



Develop Hub - Data Flows

데이터 흐름 매핑은 데이터 변환을 시각적으로 지정하는 방법을 제공하는 파이프라인 활동입니다.

코드의 작성 없이 데이터 흐름을 정의할 수 있습니다.



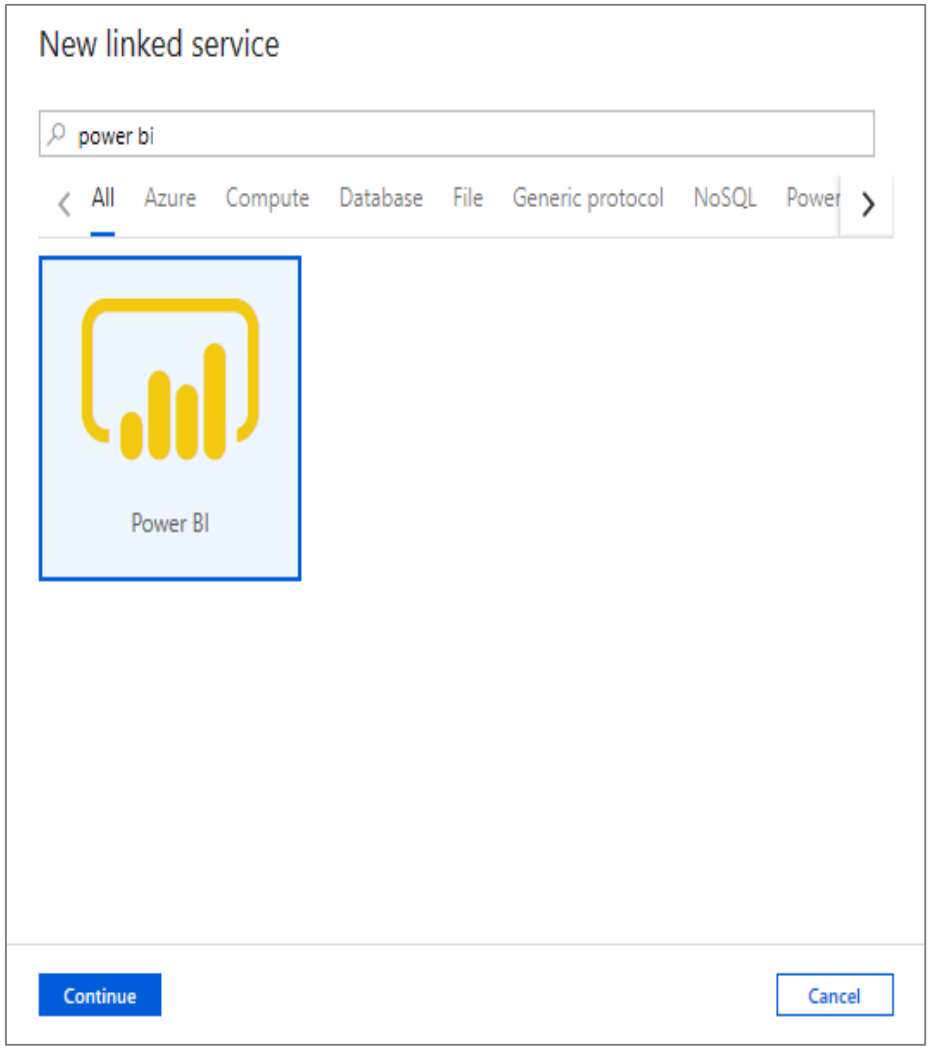
Develop Hub – Power BI

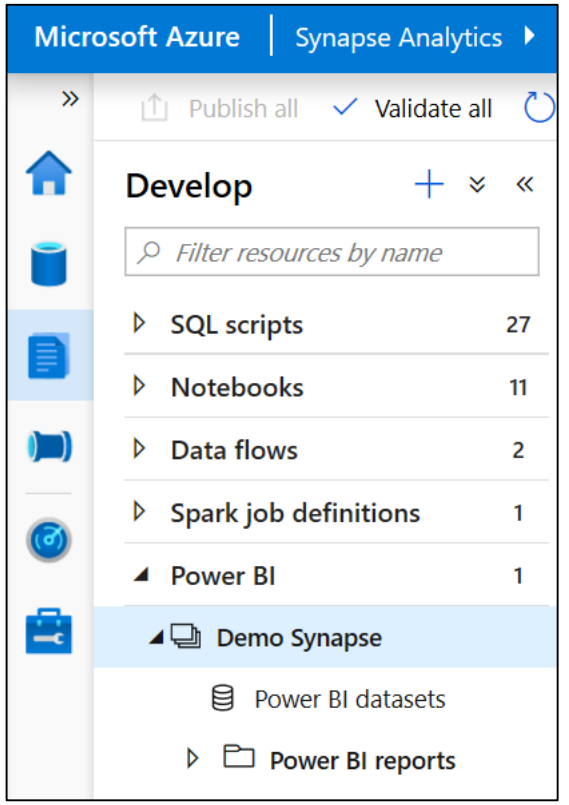
Overview

Power BI 리포트는 워크스페이스 안에서 생성할 수 있습니다.

워크스페이스 안에서 완성된 리포트에 접근이 가능하고 실시간으로 Synapse 워크스페이스로부터 업데이트가 가능합니다.

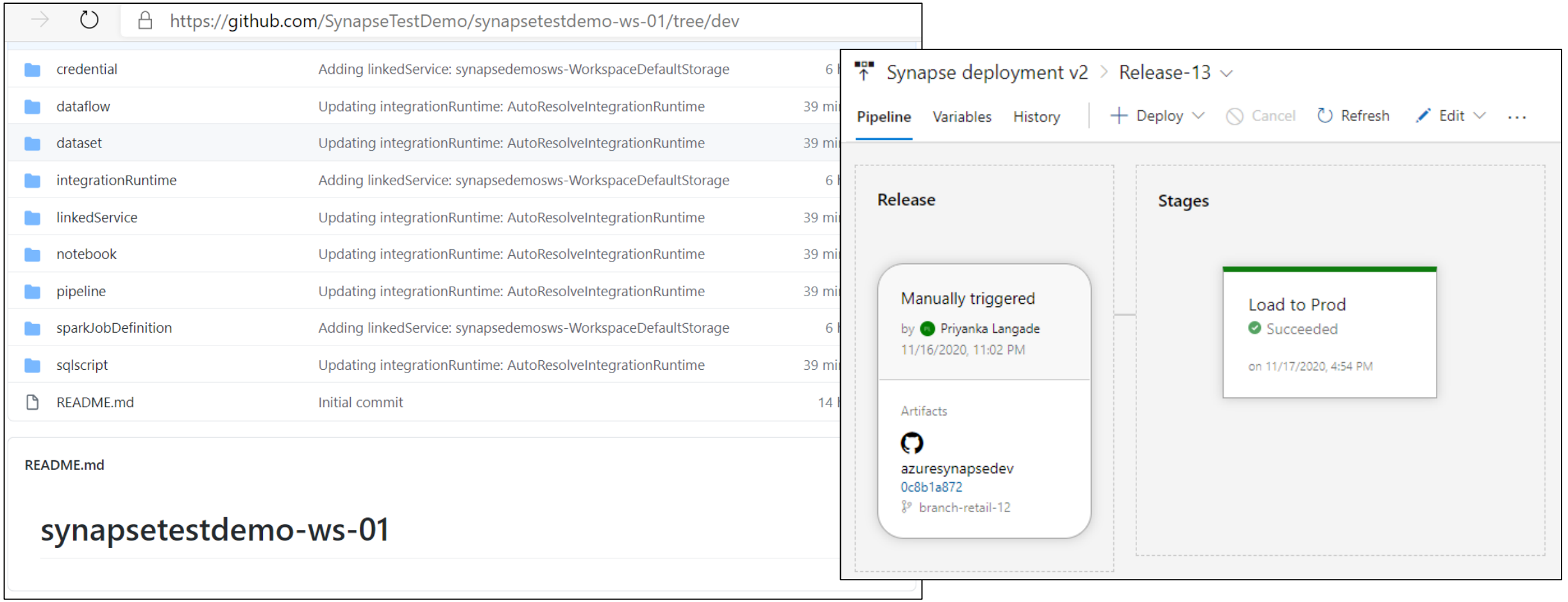
분석된 데이터를 시각화하여 볼 수 있습니다.

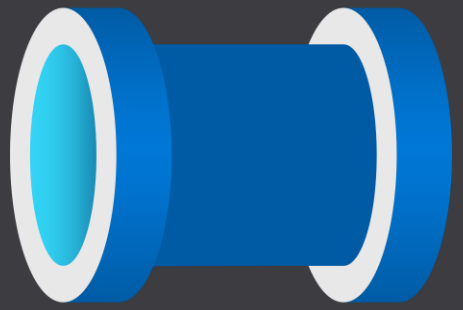




Develop – CI/CD

Synapse에 Repository를 생성하여 Artifacts들의 CI/CD 구현이 가능합니다.

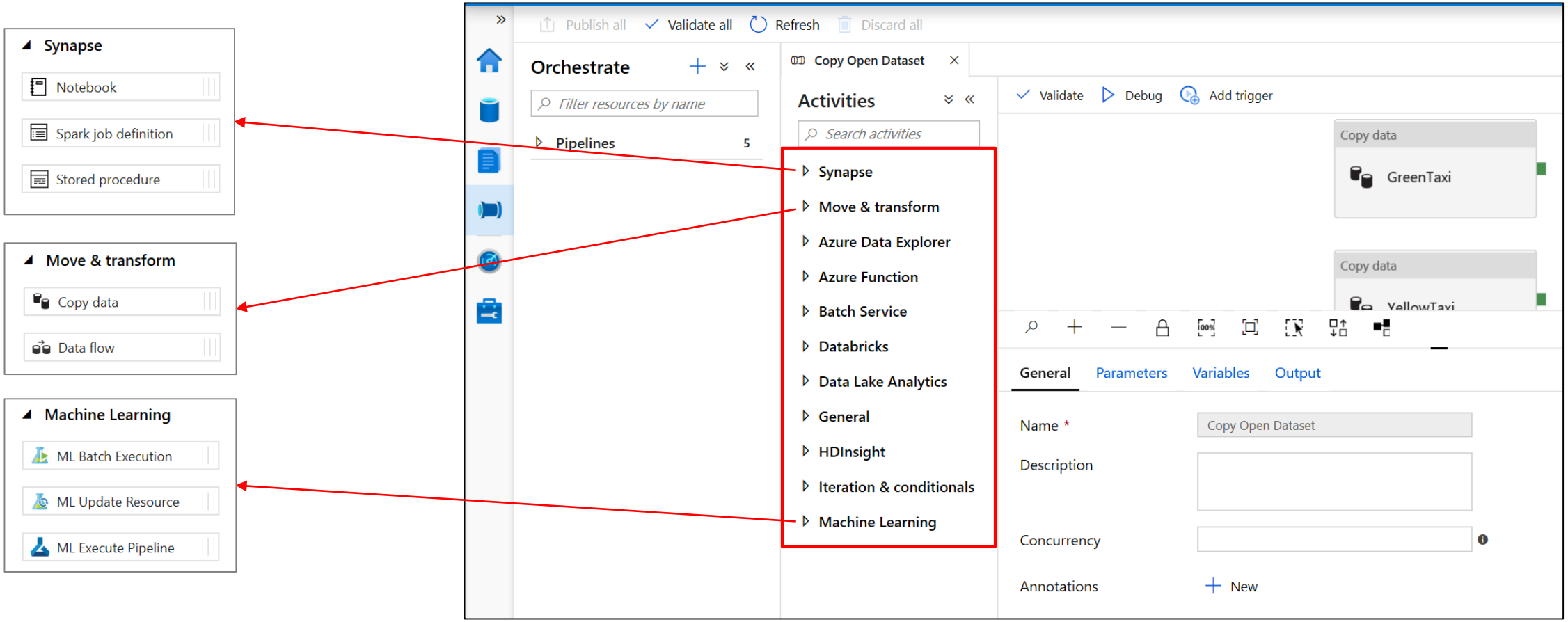




Synapse Studio
Integrate hub

Integrate Hub

Integrate Hub에서는 시냅스 내부/외부로 연결할 수 있는 90개 이상의 Connector들을 이용하여 데이터의 수집, 변형, 적재를 위한 파이프라인을 생성합니다. 파이프라인에 의해 수행되는 Activity는 지원 범위가 넓습니다.



Synapse Studio
Monitor hub

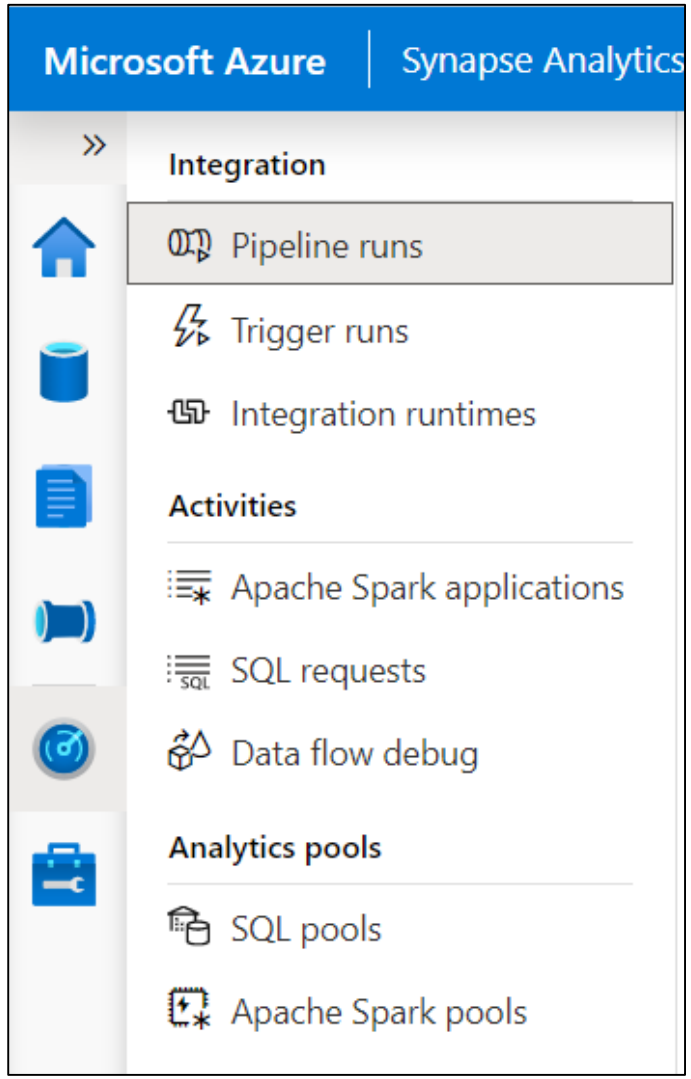
Monitor Hub

Overview

모니터 허브에서는 Apache Spark Application과 SQL Activity, 오케스트레이션의 각각에 대한 상황을 제공합니다.

Benefits

오케스트레이션이나 구체적인 Activity의 상황을 필터링을 통해 모니터링을 할 수 있습니다.



Monitor Hub - Integration

Overview

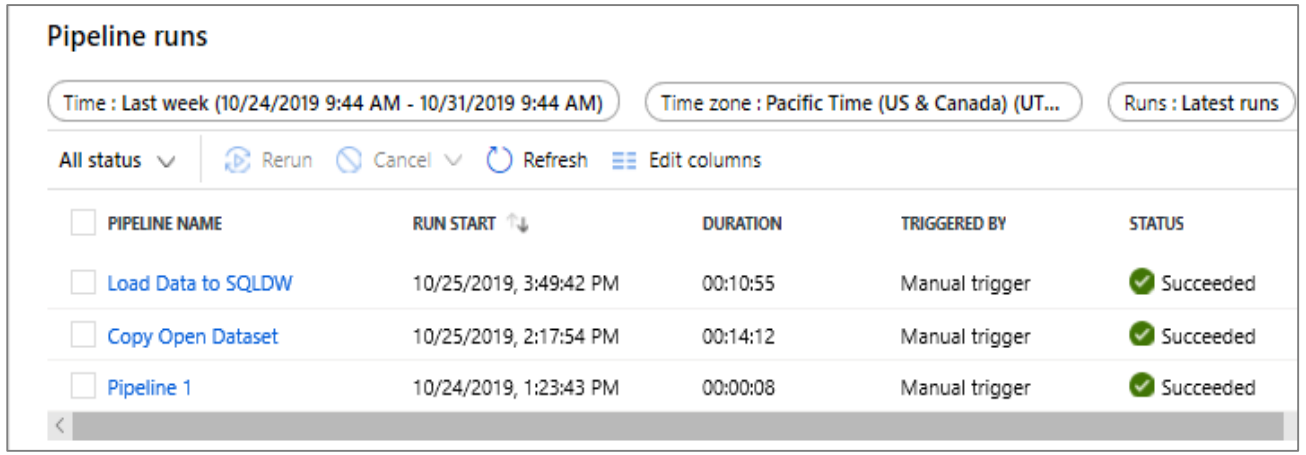
Pipeline의 진행 상황을 위한 Synapse 워크스페이스 안에서 모니터 오케스트레이션을 합니다.

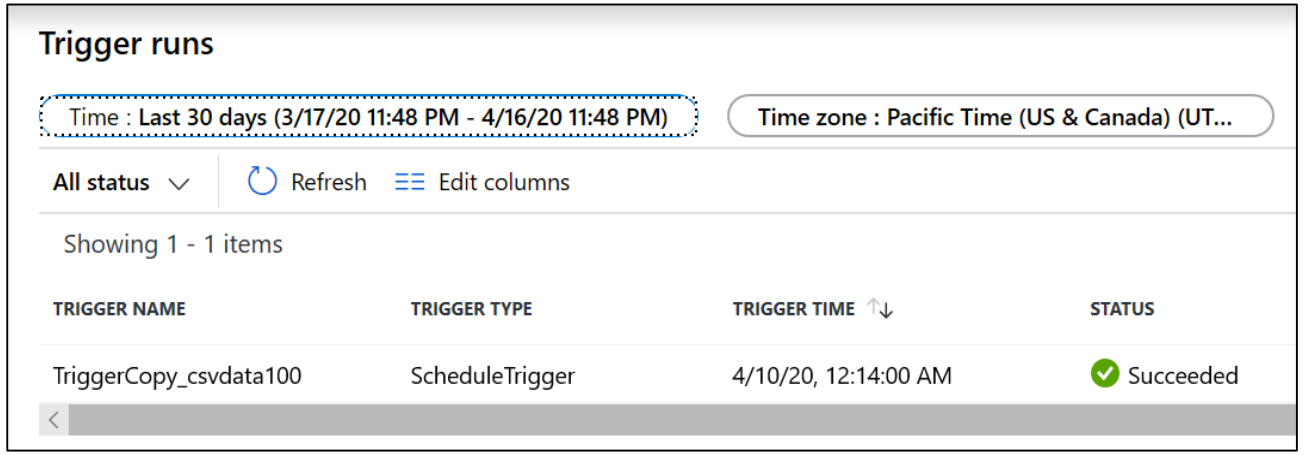
Benefits

모든 Pipeline 또는 특정 Pipeline을 추적합니다.

Activity의 구체적인 실행과 Pipeline을 모니터링합니다.

Activity의 실패 또는 Pipeline의 실패 원인을 찾아냅니다.





Monitor Hub - Spark applications

Overview

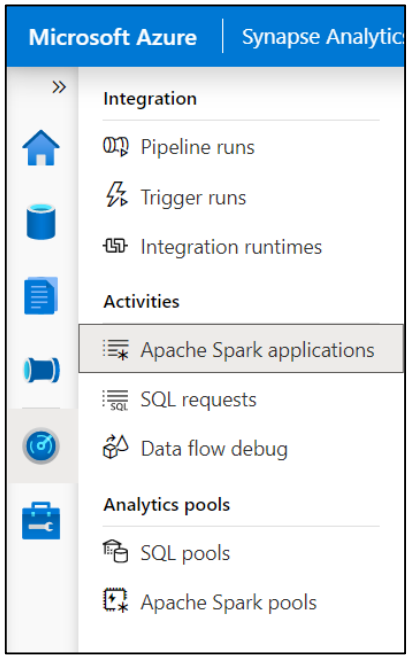
Activity의 상태를 위한 Spark Pool, Spark Application 상태를 모니터링합니다.

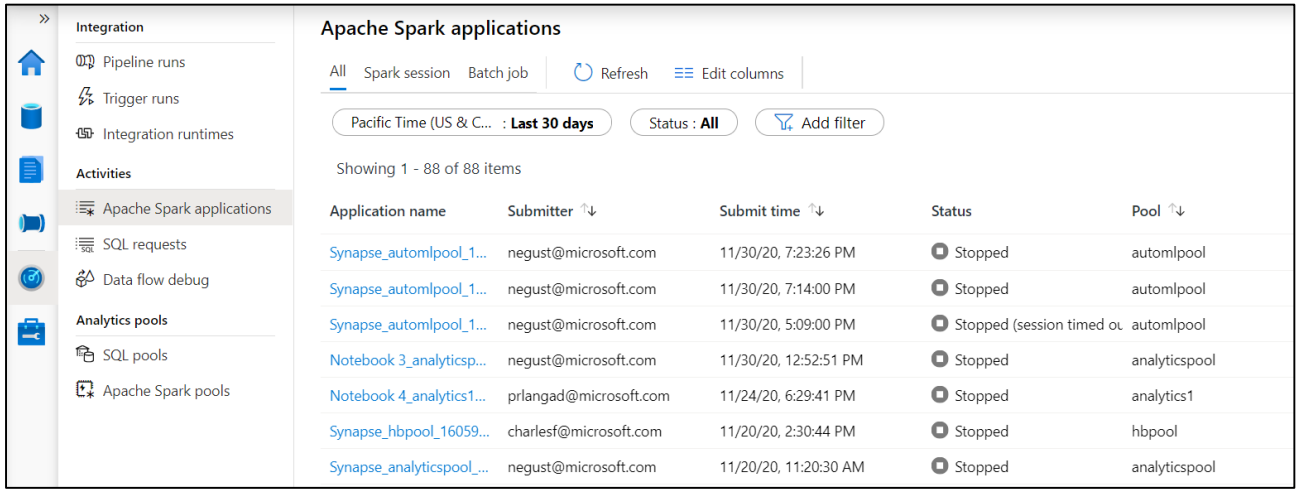
Benefits

Apache Spark Application의 Pool에 대한 필터링 적용

이용 가능한 필터링 제공

1. 애플리케이션 이름
2. Live ID
3. 상태
4. 완성 시간





Monitor Hub – SQL requests

Overview

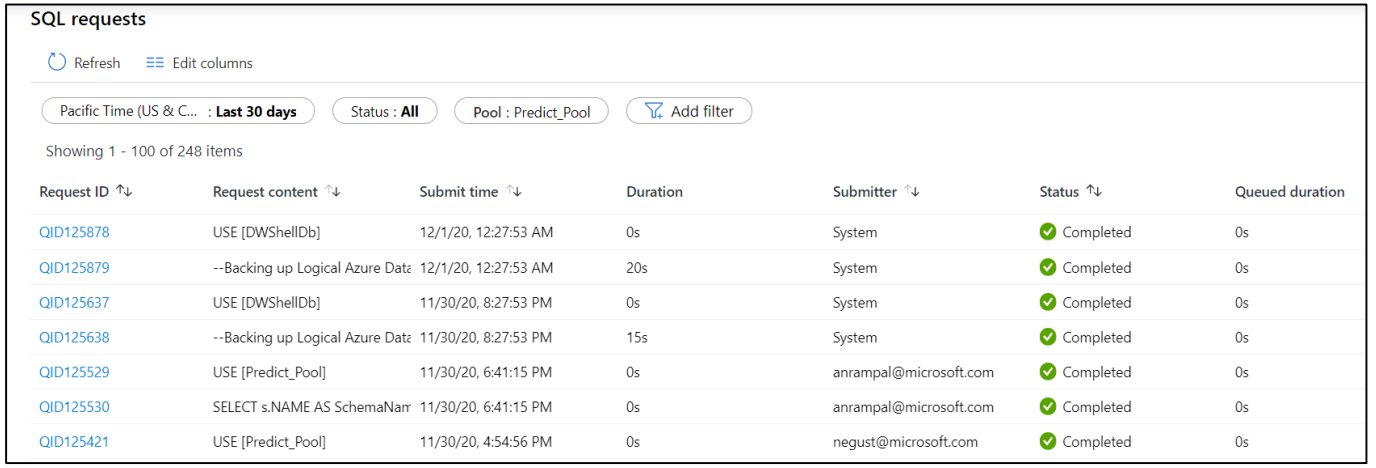
Activity의 진행 상태를 위한 SQL 요청을 모니터링합니다.

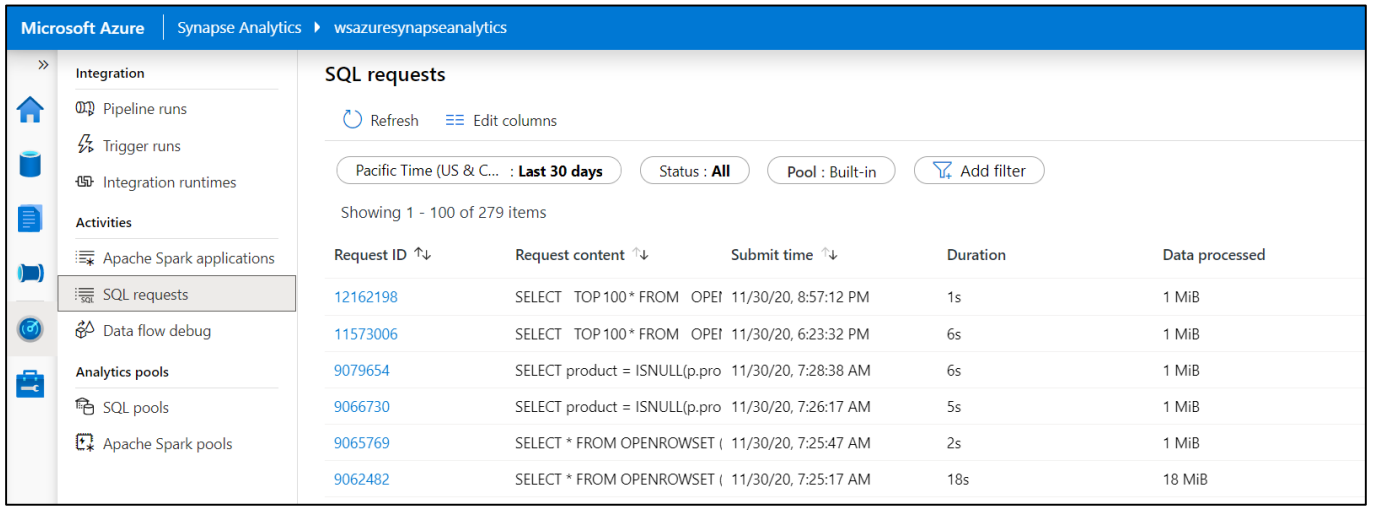
Benefits

Pool이 SQL 요청을 얻기 위해 필터링 적용합니다.

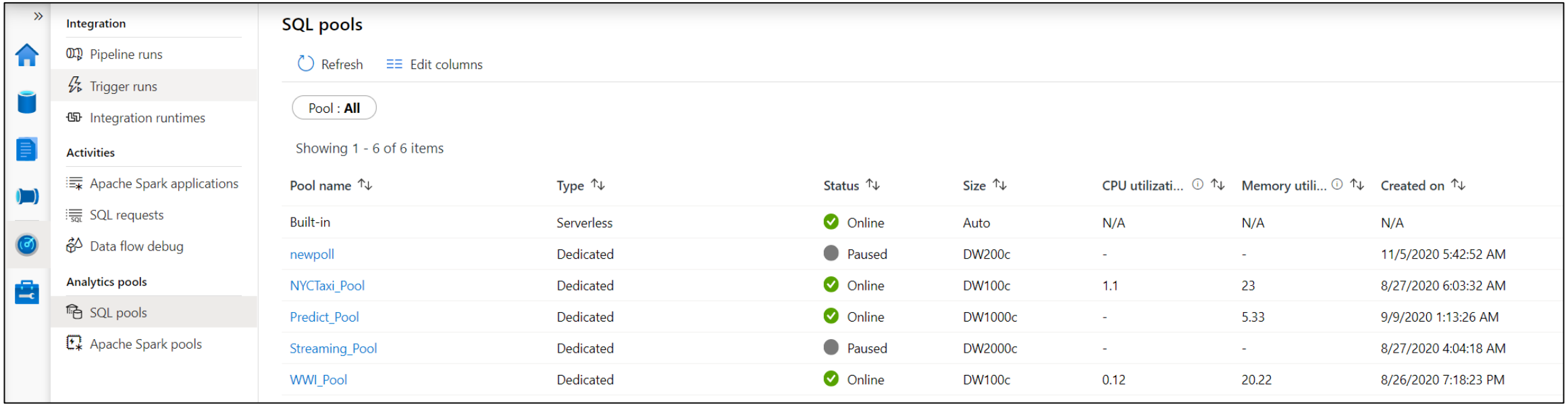
쿼리 구문을 허용합니다.
이용가능한 추가 필터링 제공

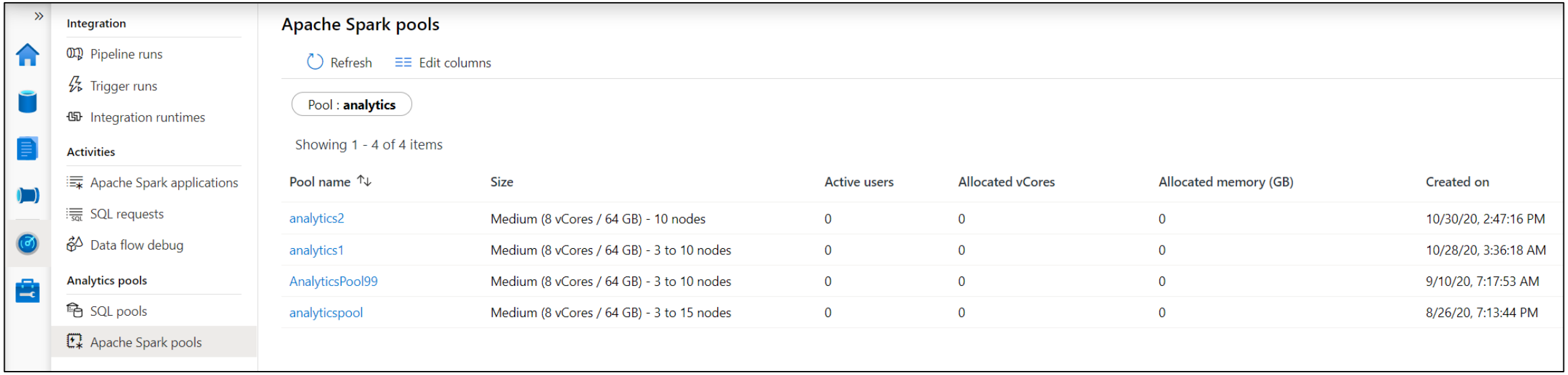
1. 시작시간
2. 종료시간
3. 요청 ID
4. 세션 ID
5. 제공자
6. 워크로드 그룹





Monitor Hub – Analytics pools



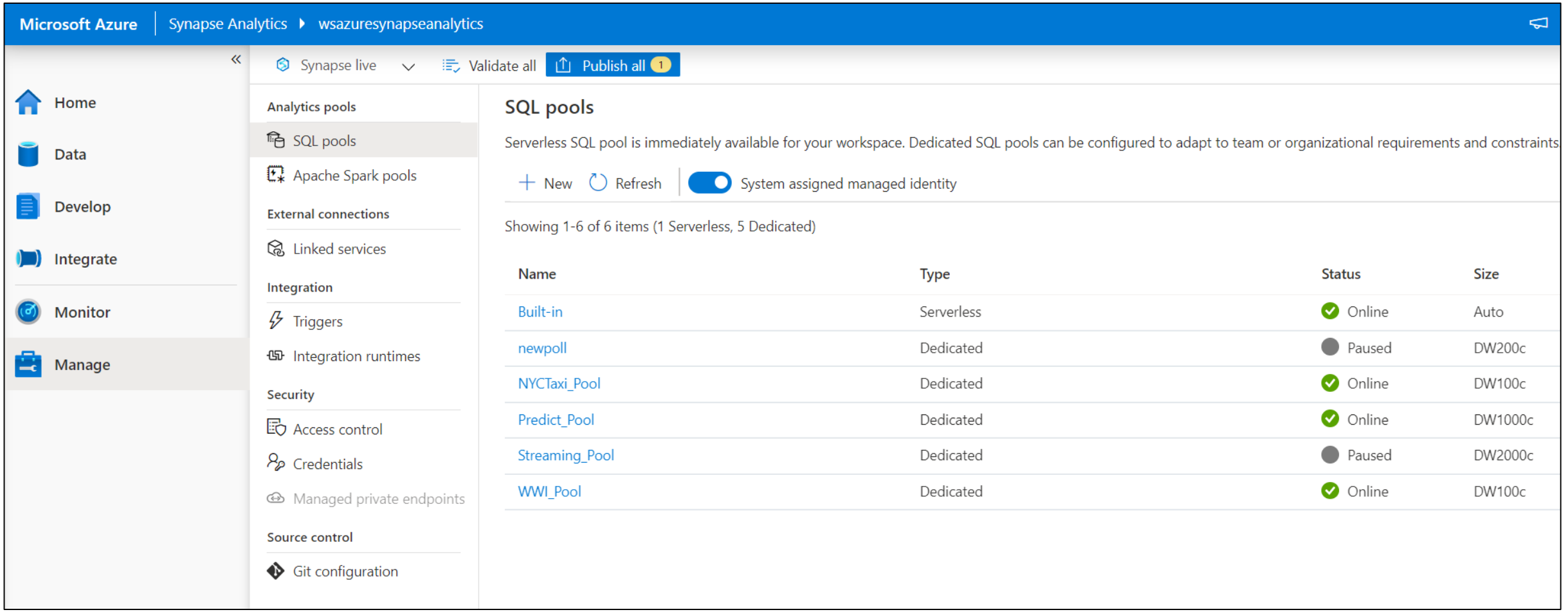


Synapse Studio
Manage hub

Manage Hub

Overview

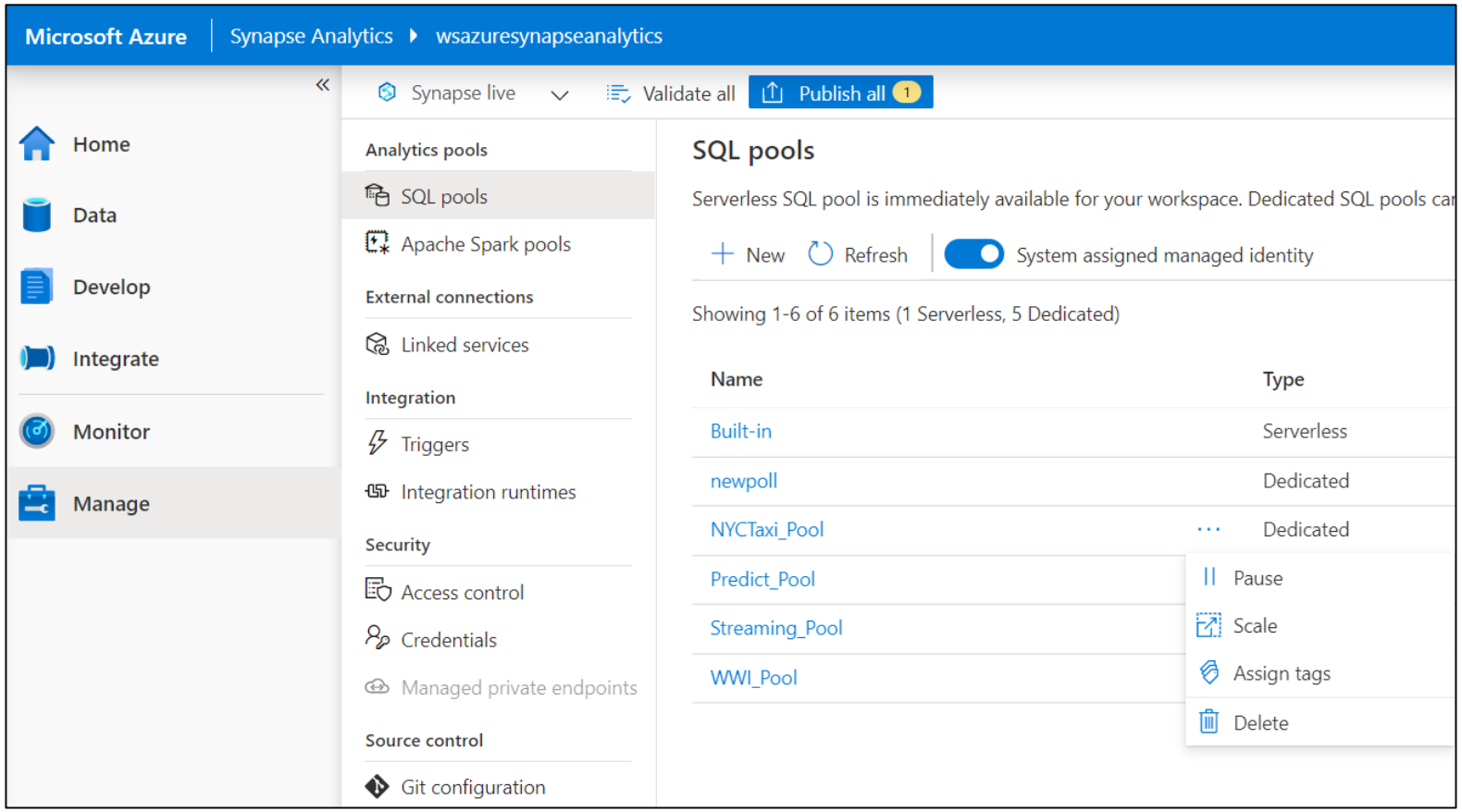
Manage Hub은 분석 풀, 연결 서비스, 통합, 보안 그리고 Source Control을 제공합니다.

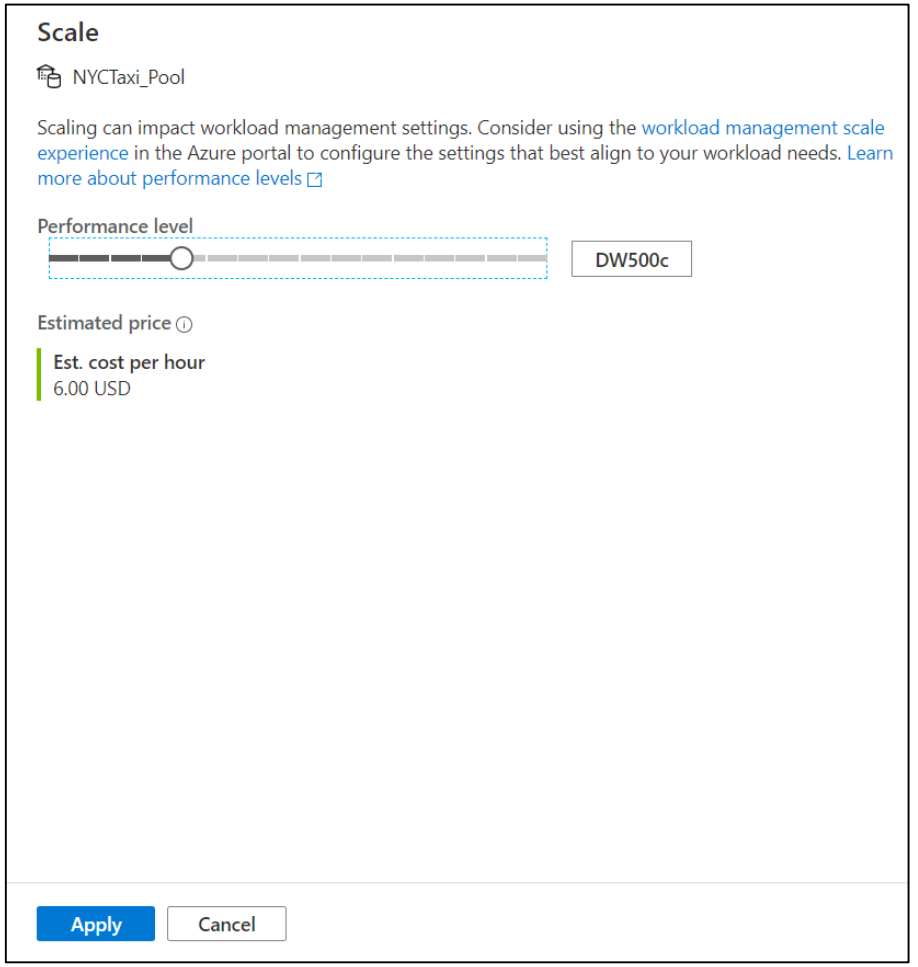


Manage – dedicated SQL pools

Overview

전용 SQL 풀은 정지, 재시작, 스케일 변경이 가능하며, 스튜디오에서 Tag를 지정할 수 있습니다.

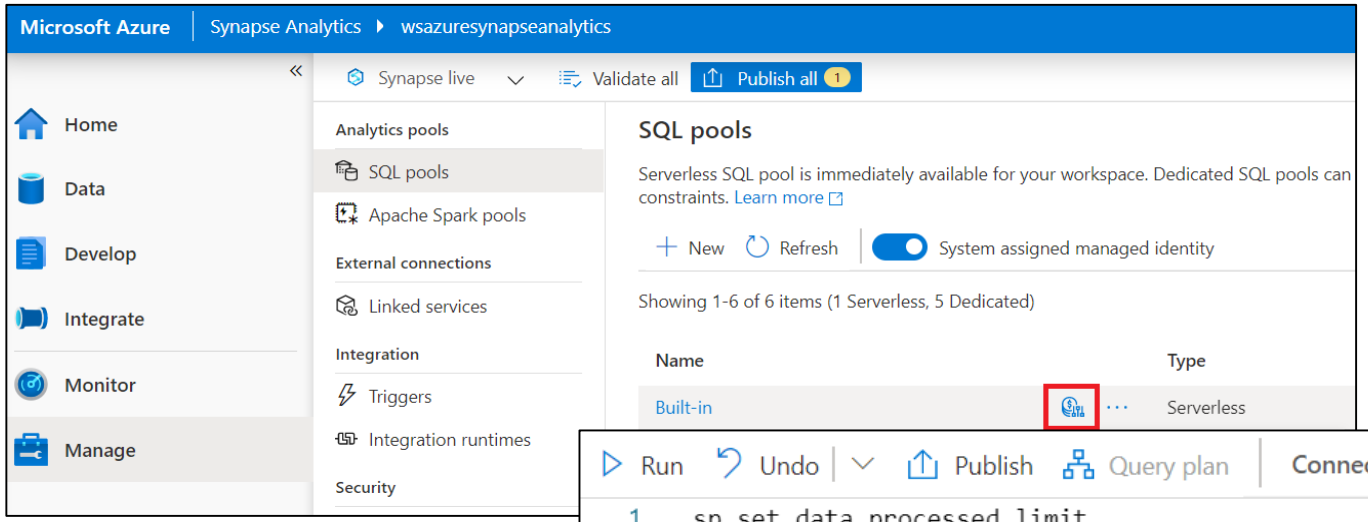


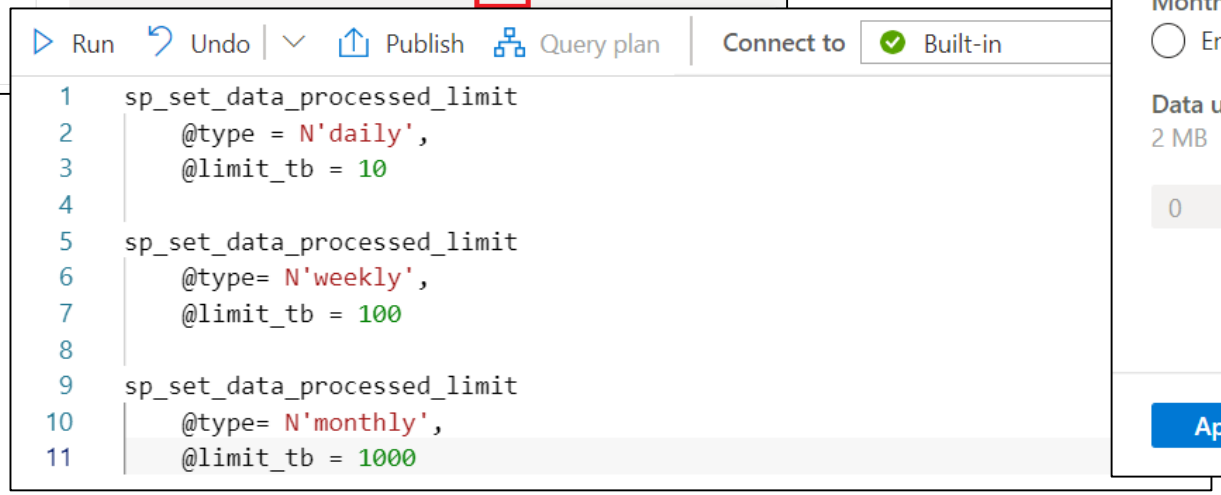


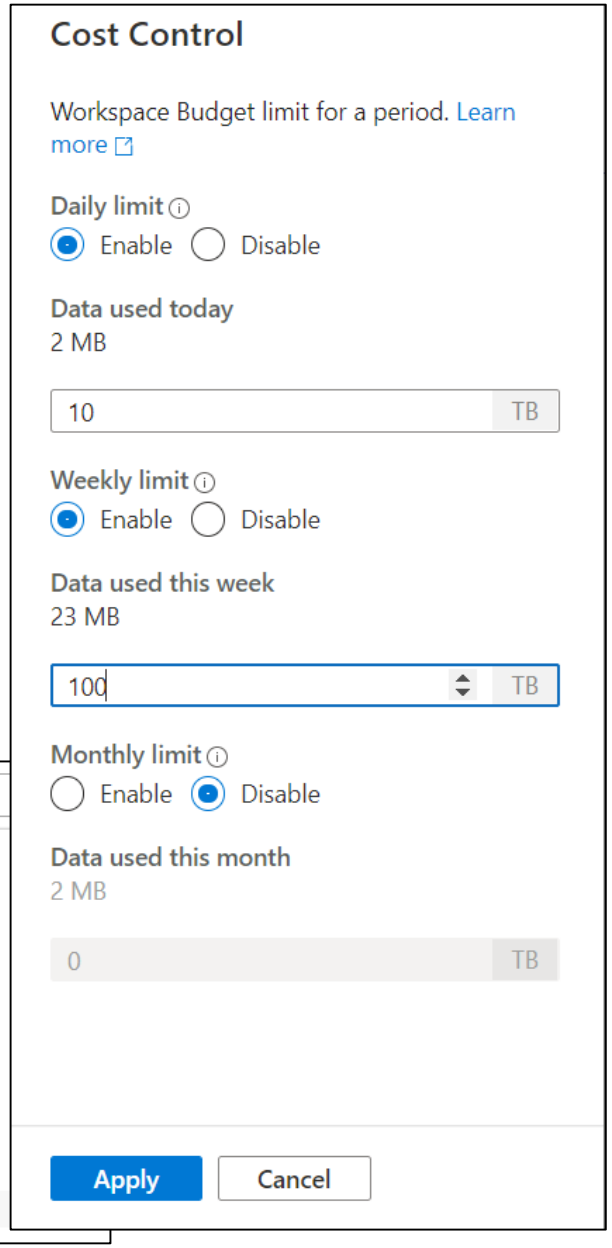
Manage – serverless SQL pools

Overview

Serverless SQL 풀은 T-SQL 또는 시냅스 스튜디오 안에서 진행되고 있는 데이터의 양에 대한 리소스를 측정하고 관리합니다.



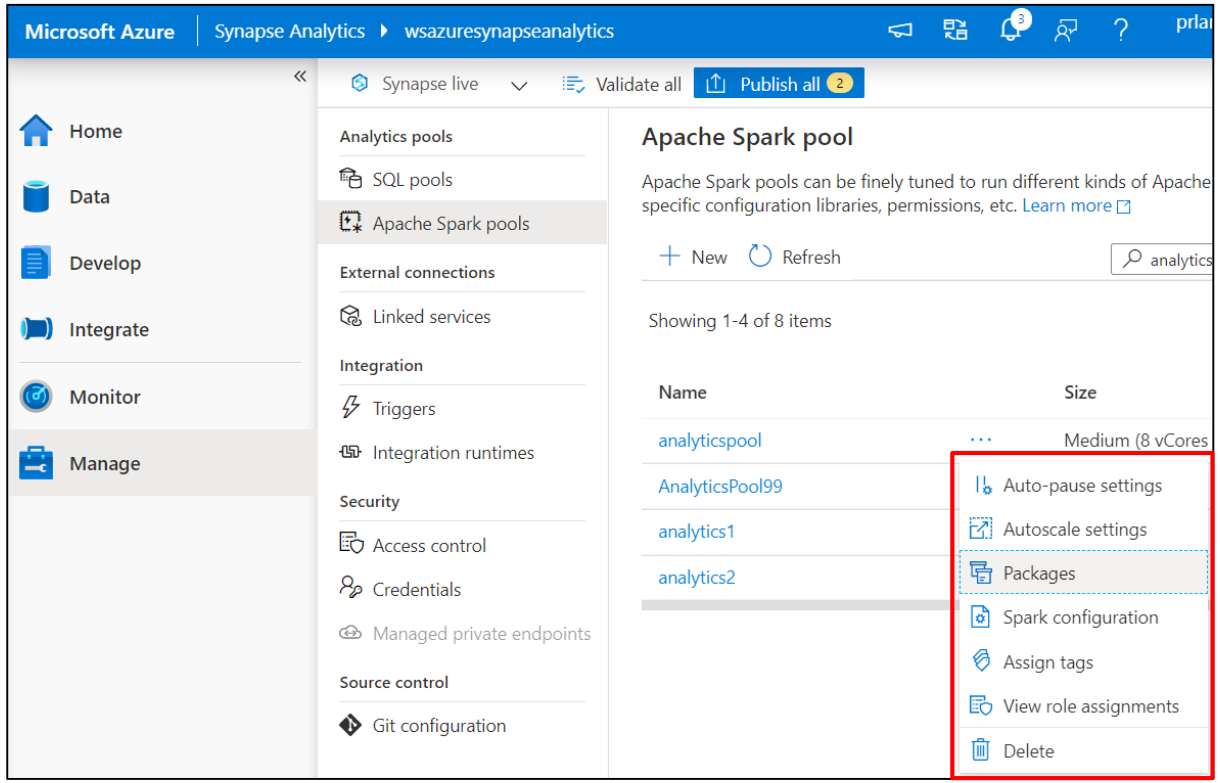


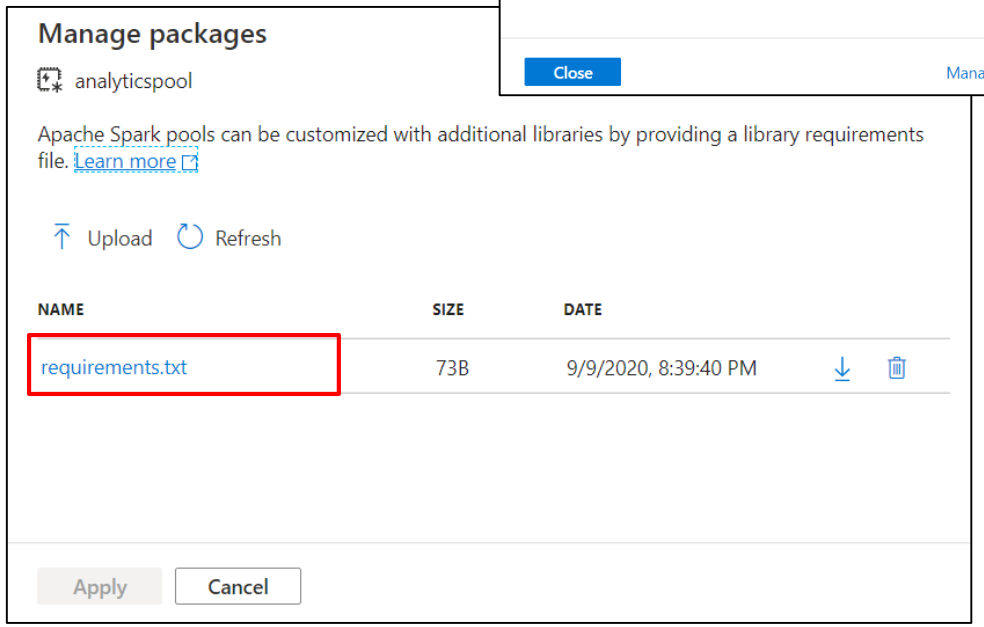


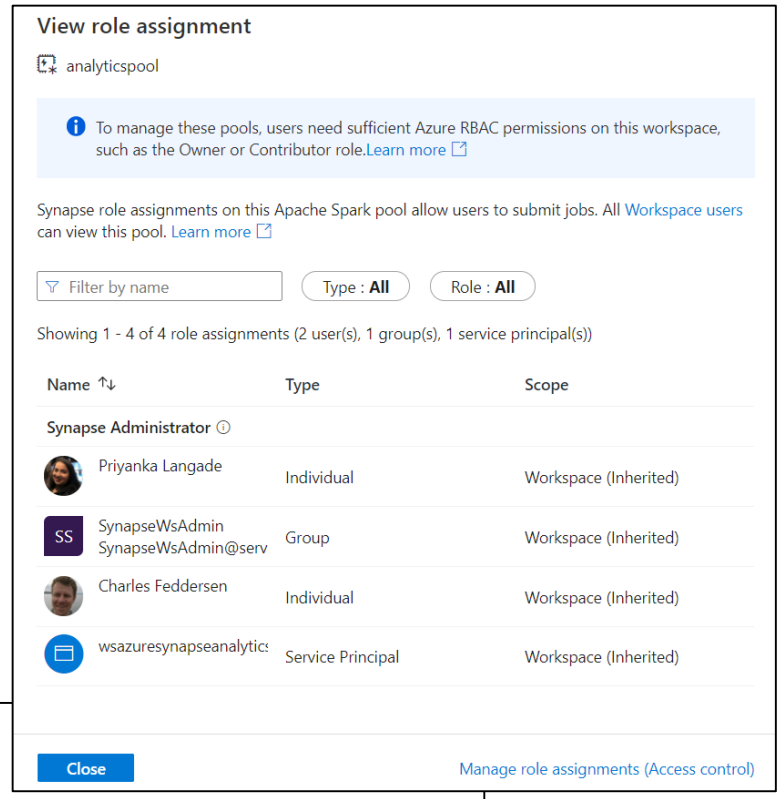
Manage – Apache Spark pools

Overview

Apache Spark 풀은 스튜디오에서 패키지, 구성 관리를 업로드하고, 태그를 지정해 주며 스케일 관리를 합니다. 또한 스파크 풀의 역할 지정을 허용합니다.







Manage – Linked services

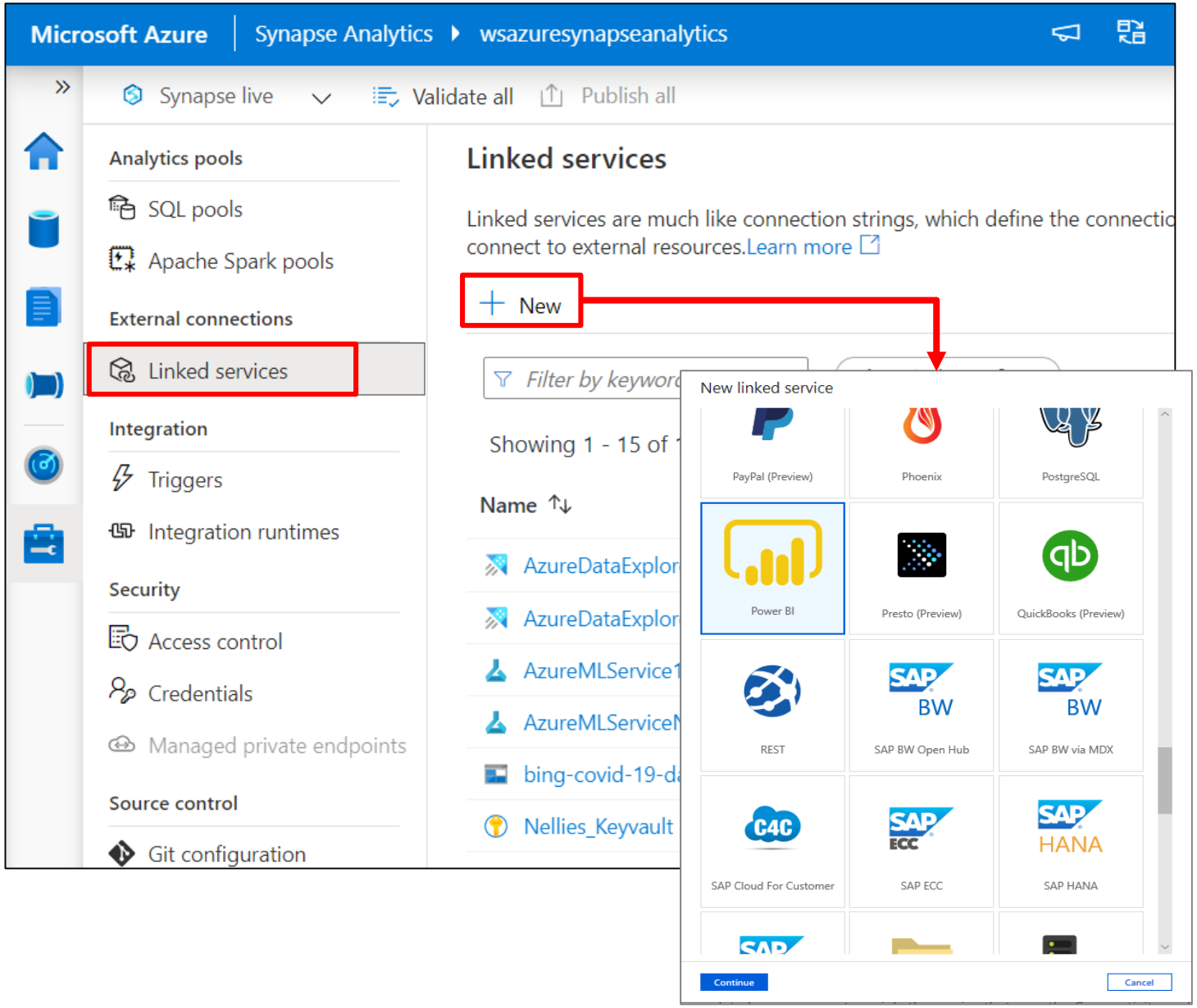
Overview

Linked Service는 외부 리소스에 연결하려는 연결 정보를 정의합니다.

Benefits

90개 이상의 Connector를 제공합니다.

컴퓨팅 리소스 또는 데이터 스토어를 나타냅니다.



Manage – Triggers

Overview

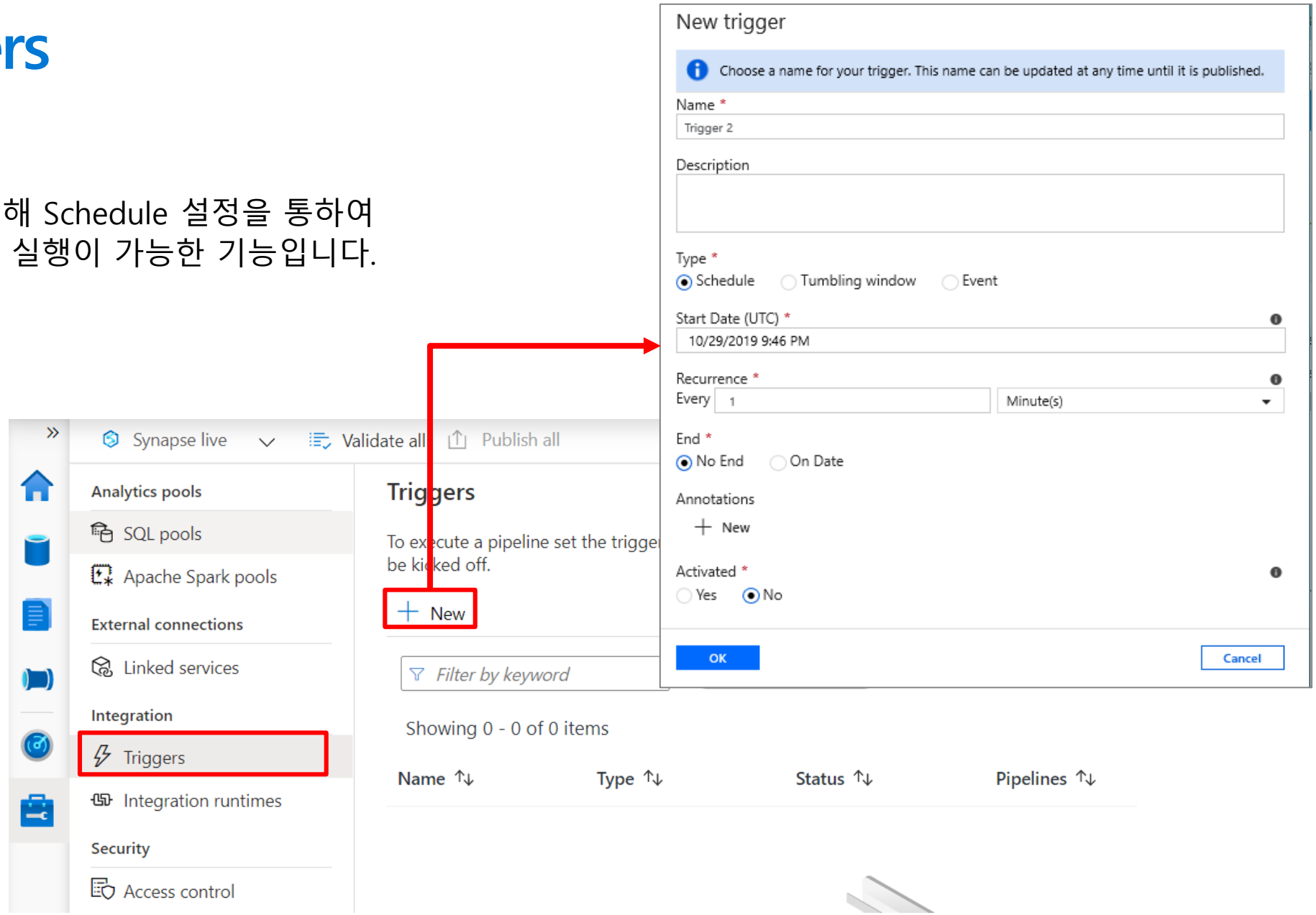
Trigger는 Pipeline을 수행하기 위해 Schedule 설정을 통하여 자동 실행 or 사용자에 의한 수동 실행이 가능한 기능입니다.

Benefits

Trigger를 생성하고 관리할 수 있습니다.

- 트리거의 3가지 종류

Schedule 트리거
Dumbling Window 트리거
Event 트리거



Manage – Integration runtimes

Overview

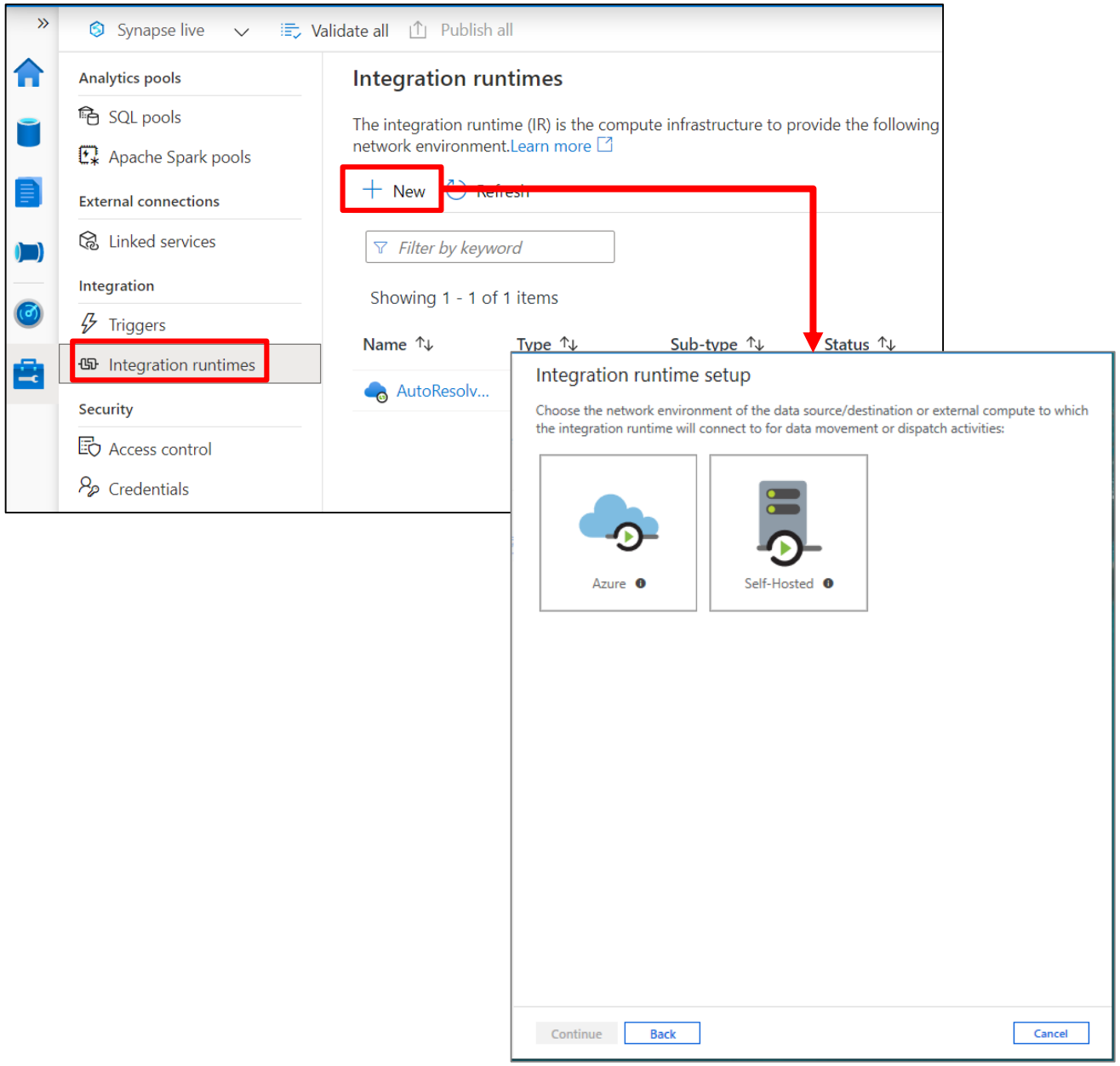
Benefits

Integration runtime은 서로 다른 네트워크 환경에서 통합 환경을 제공하기 위한 Pipeline에 의해 사용된 컴퓨팅 Infra Structure 입니다.

Integration runtime은 Activity와 Linked Service 사이의 다리 역할을 제공합니다.

Integration runtime은 Azure Integration runtime 또는 Self-Hosted된 Integration runtime을 제공합니다.
Azure Integration runtime은 완전히 관리되는 Azure 내의 서버리스 컴퓨팅을 제공합니다.

Self-Hosted 통합 런타임은 온프레미스 서버 또는 사설 네트워크 안의 가상 머신 안의 컴퓨팅 리소스를 사용합니다.



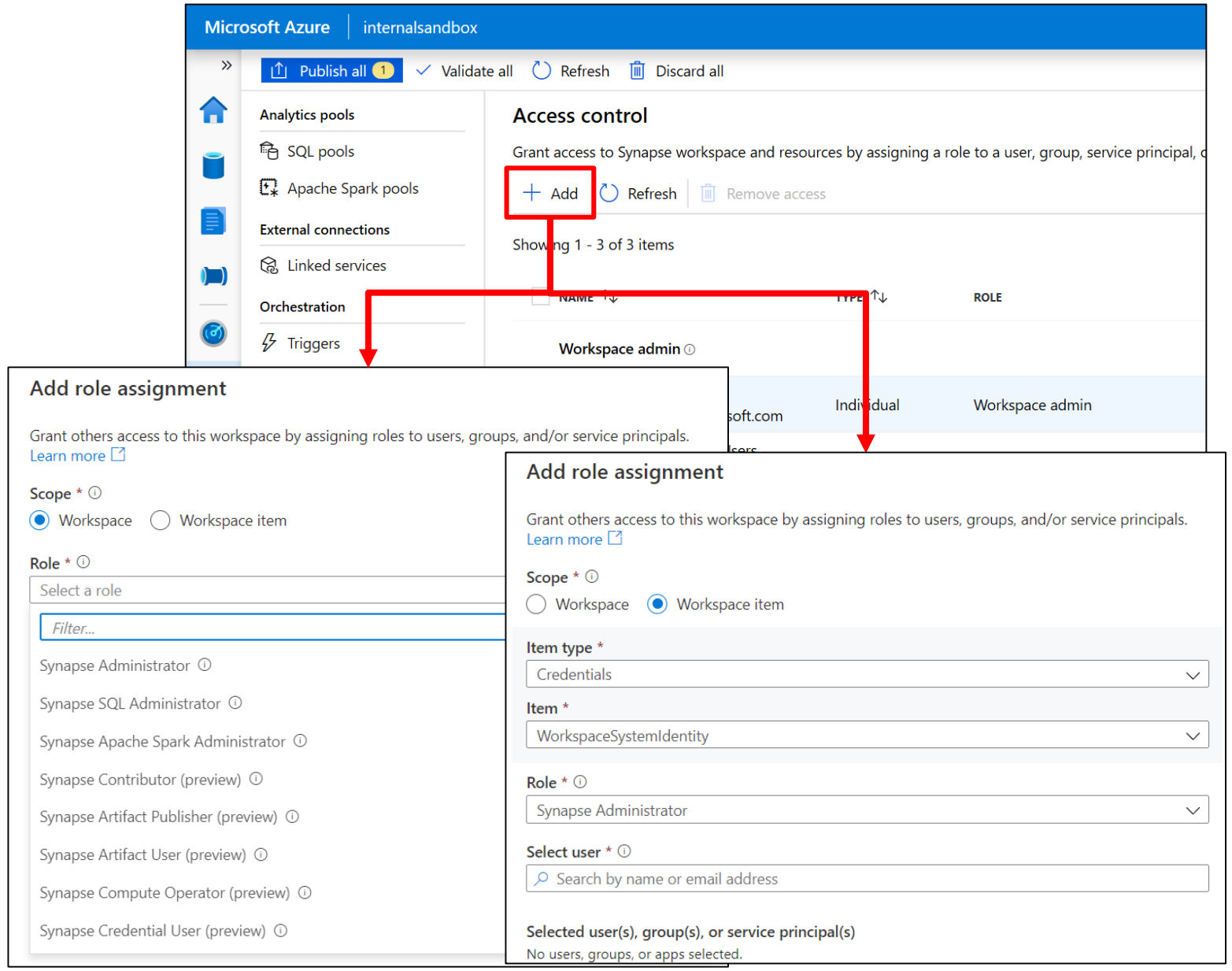
Manage – Access Control

Overview

Access Control은 워크스페이스 리소스부터 관리자의 Artifact까지 접근 제어 관리를 제공합니다.

Benefits

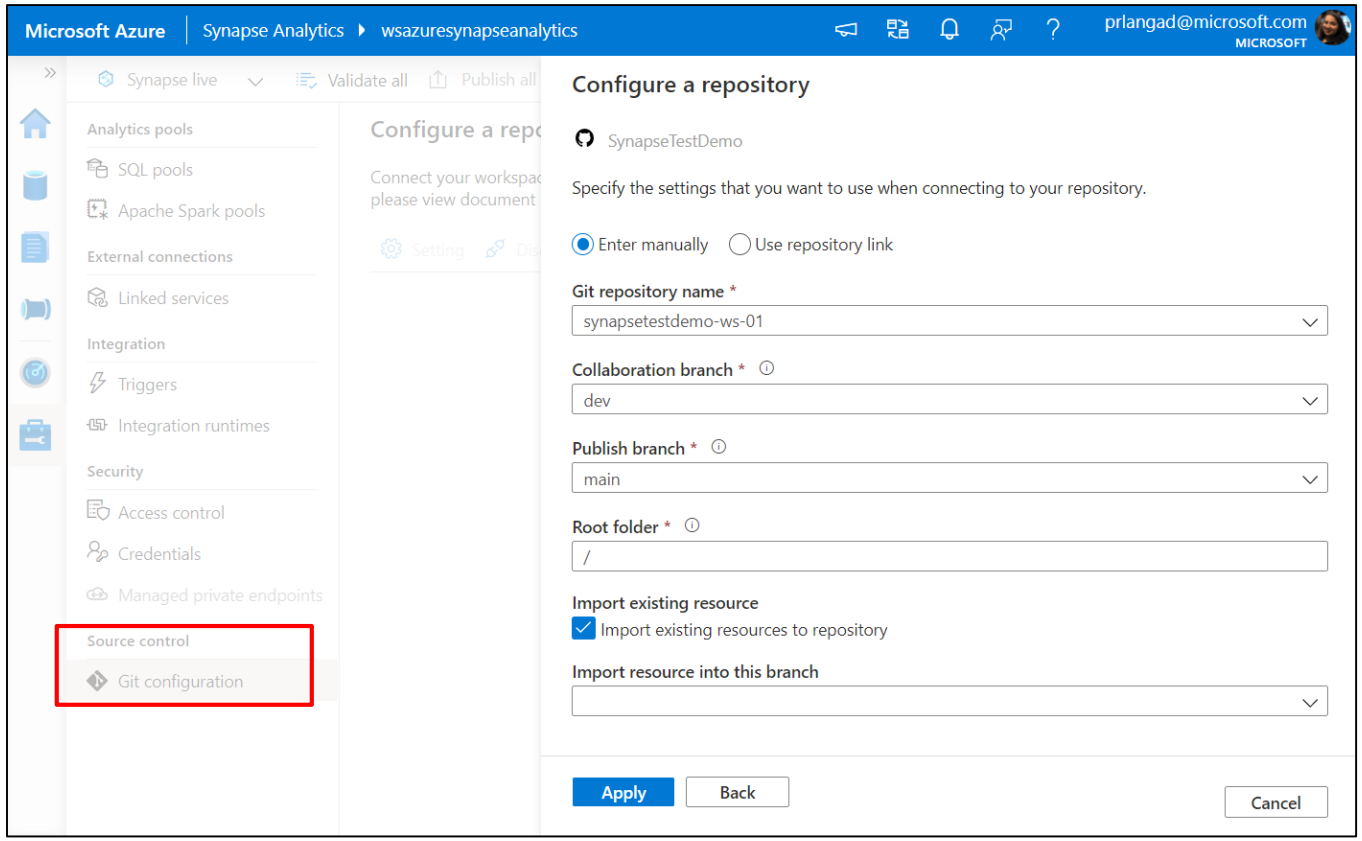
- 팀 내에서 워크스페이스를 공유
- 생산성의 증가
- 단계적인 허가 가능 수준 지정
- 스파크 풀, 통합 런타임, 연결 서비스, 비밀번호의 권한 관리

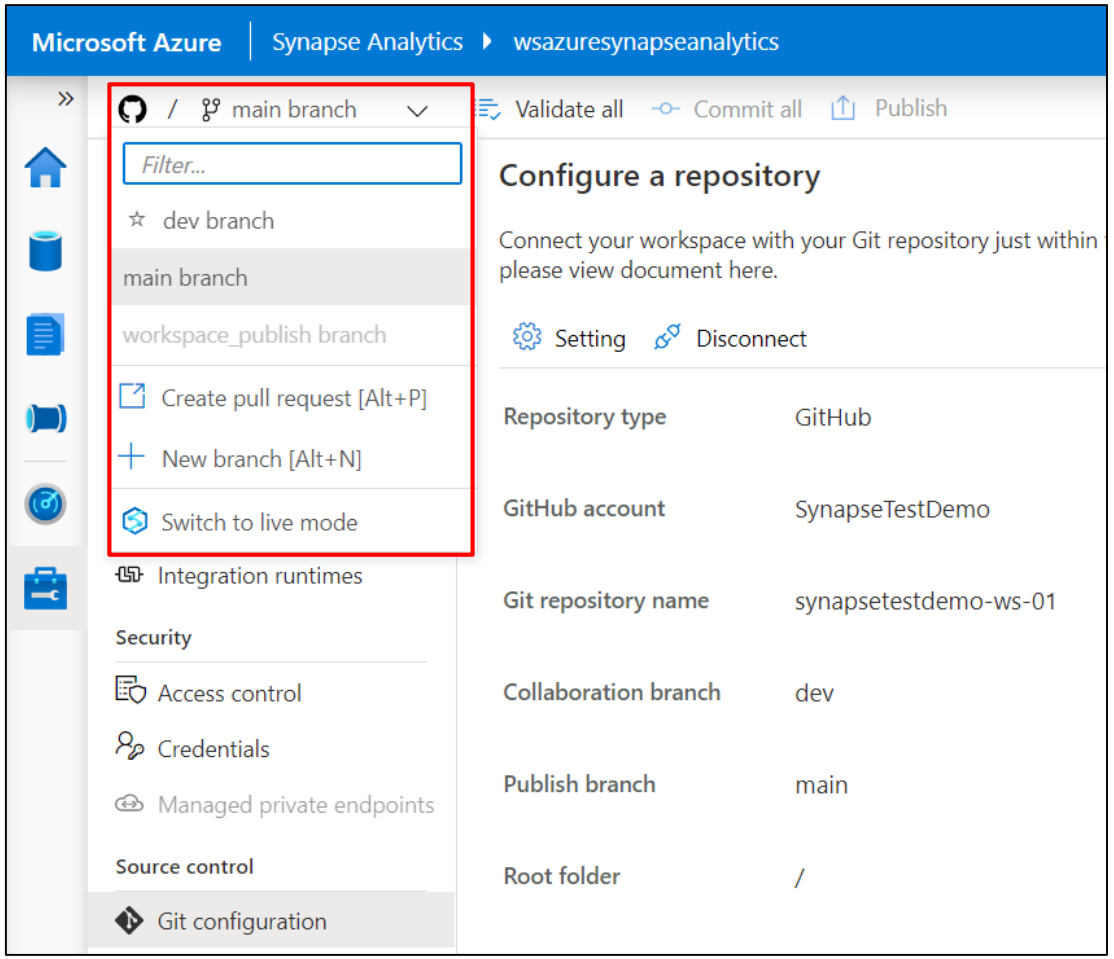


Manage – Source Control

Overview

소스 제어는 Git configuration에서 Azure devops hub, Github을 연동할 수 있습니다.

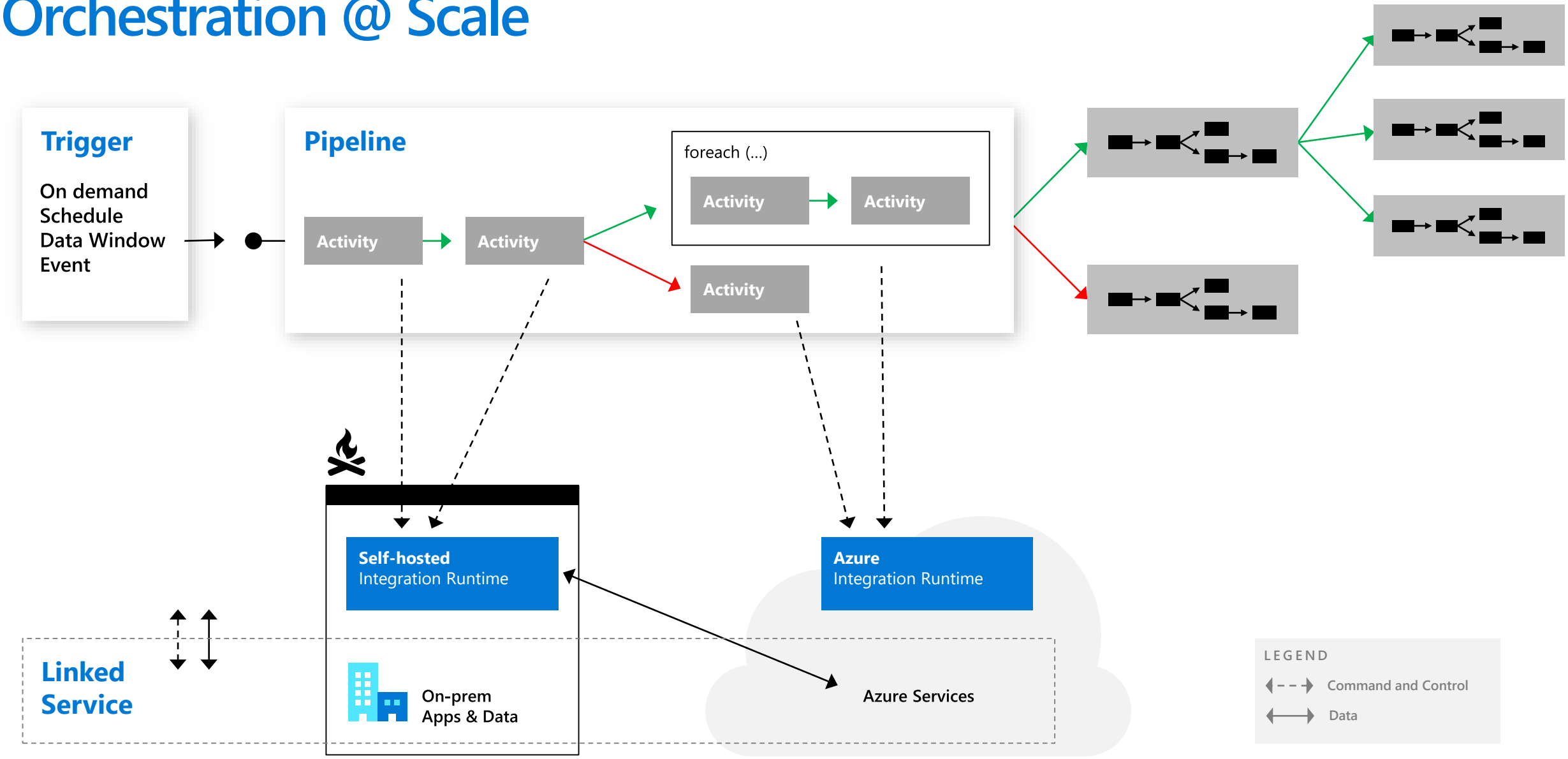




Azure Synapse Analytics

Integrate

Orchestration @ Scale



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								

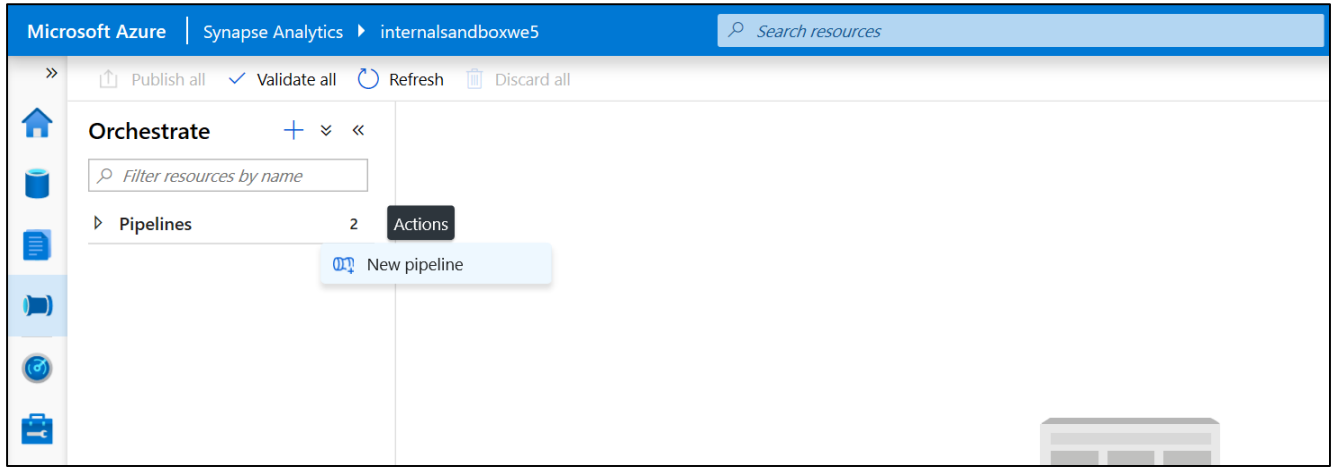
Data Movement - Scalable

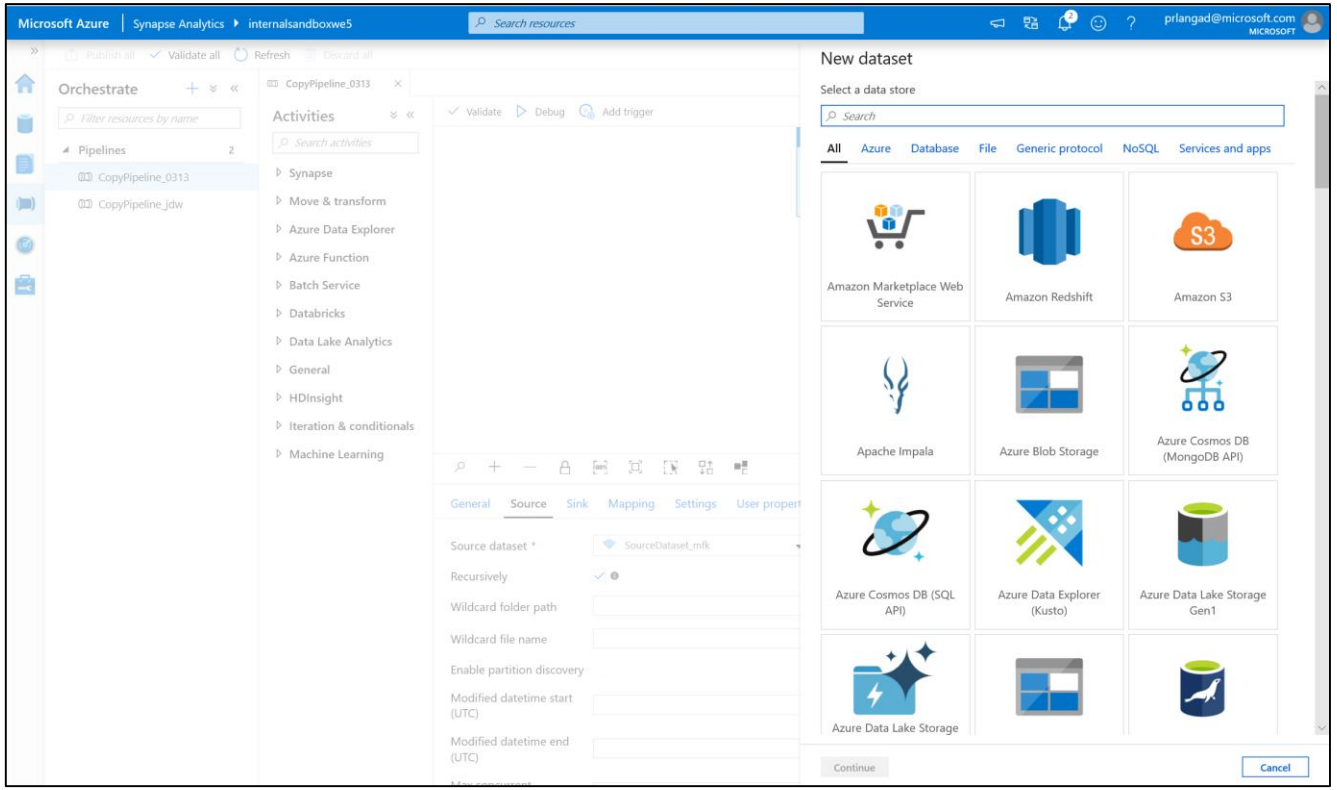
Per job elasticity - Up to 4 GB/s

Pipelines

Overview

Pipeline은 Storage Account 와 Linked Service 로 부터 데이터를 적재할 수 있는 기능을 지원합니다. Pipeline 은 자동 혹은 수동으로 수행하여 Data를 적재 할 수 있습니다.





Benefits

Supports common loading patterns

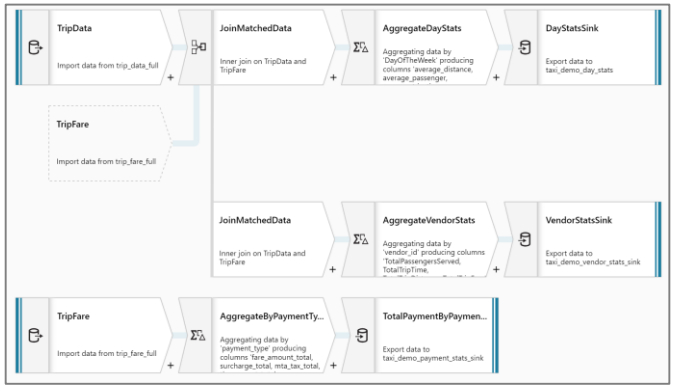
Fully parallel loading into data lake or SQL tables

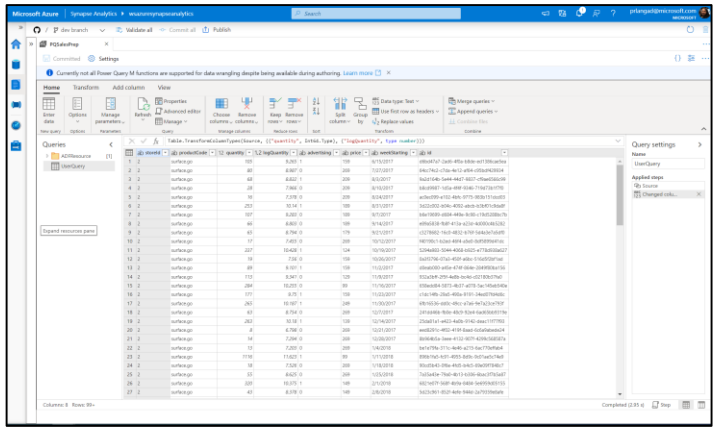
Graphical development experience

Prep & Transform Data

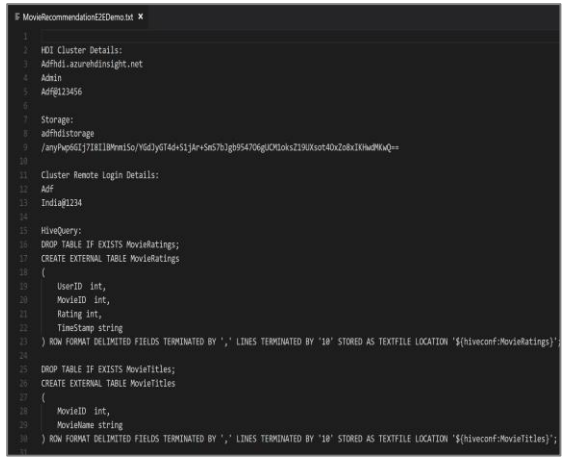
Overview

데이터 정제(cleaning), 변형, 집계, 전환 등의 기능을 지원합니다.





...
not



Benefits

Cloud scale via Spark execution

Guided experience to easily build resilient data flows

Flexibility to transform data per user's comfort

Monitor and manage dataflows from a single pane of glass

Triggers

Overview

트리거는 파이프라인을 실행하는 기능입니다.

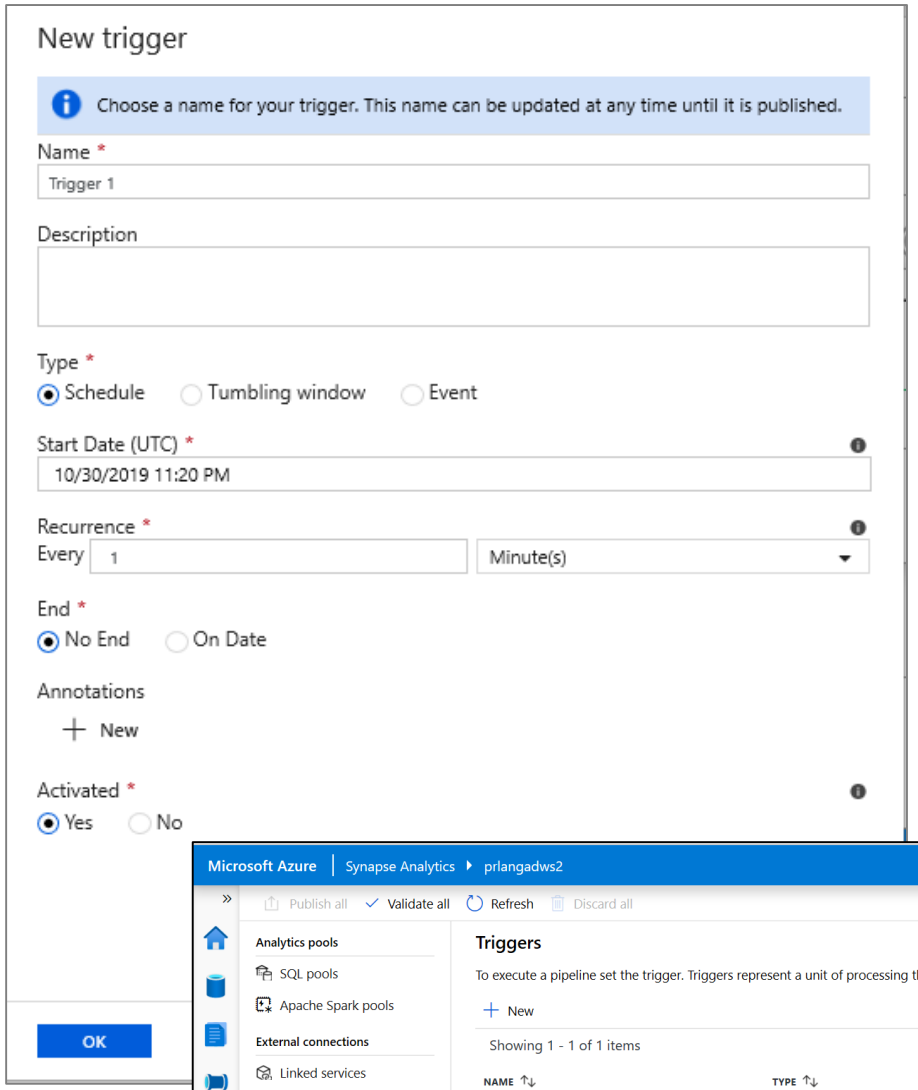
3가지 종류의 트리거를 제공합니다.

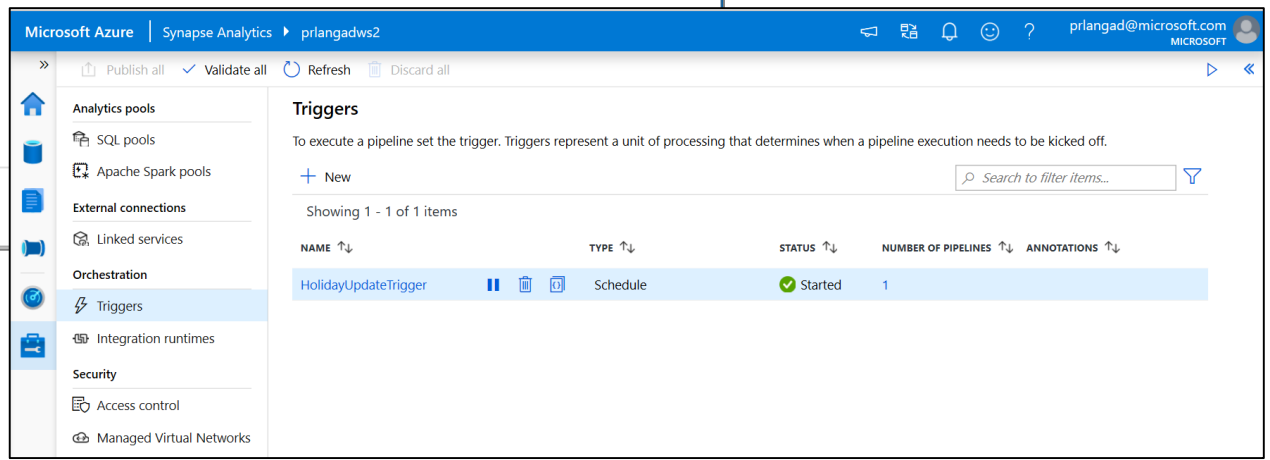
1. Schedule – 시작일, 반복, 종료일의 예약에 따라 수행됩니다.

2. Event – 지정한 특정 이벤트가 발생했을 때 수행됩니다.

3. Tumbling Window – 지정된 시작일로부터 일정한 시간 간격으로 파이프라인을 수행합니다.

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





Azure Synapse Analytics

Synapse SQL

Key features

Rich Surface Area

- **T-SQL**을 사용하여 데이터 분석
- 엔터프라이즈급 보안

Dedicated SQL Pool

- 엔터프라이즈급 **DW**
- 인덱싱 **&** 캐싱 지원
- 외부 데이터 조회 및 **LOAD**
- 워크로드 관리

Serverless SQL Pool

- 외부 데이터 조회
- 원시 데이터를 테이블 및 뷰로 모델화
- 쉬운 데이터 변환

Synapse SQL - clients and tools

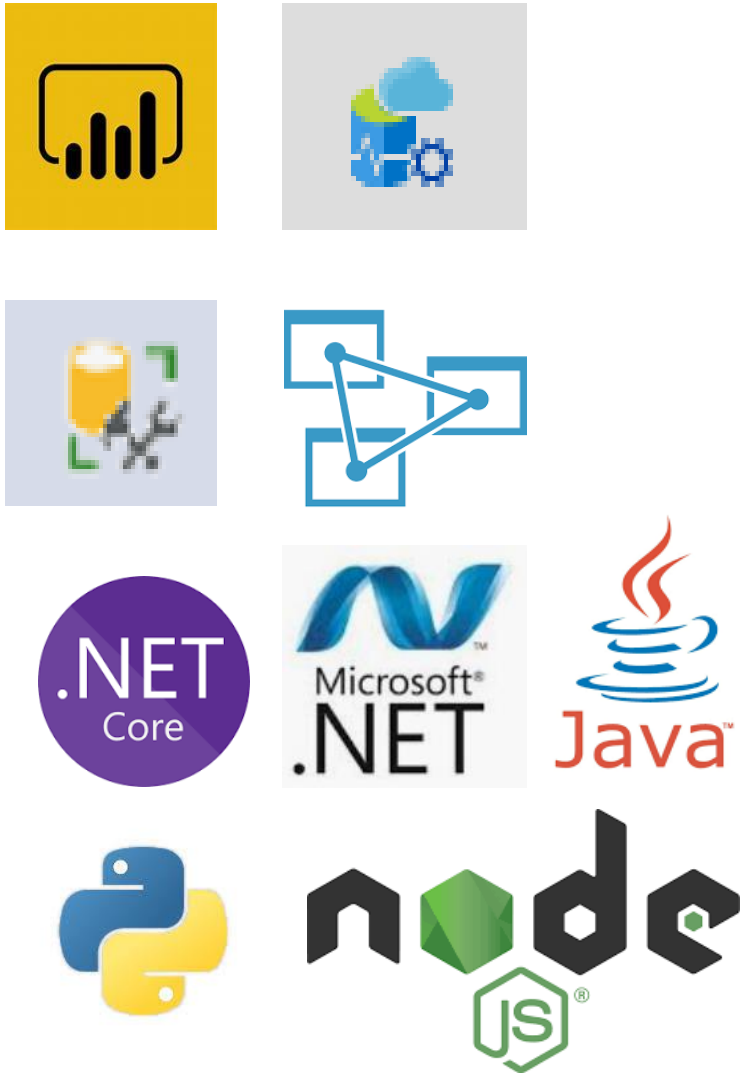
Tools

Web IDE - Synapse Studio

Client tools - Azure Data Studio, SSMS,

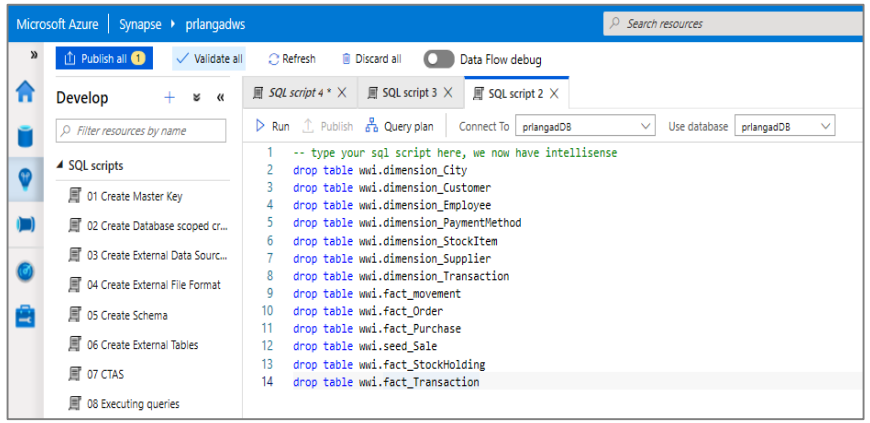
Any tool/library that uses standard SQL can access serverless SQL pool

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

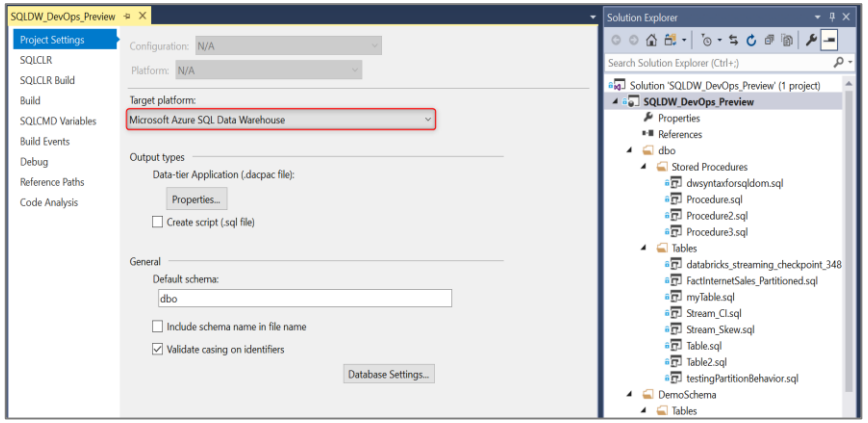


Developer Tools

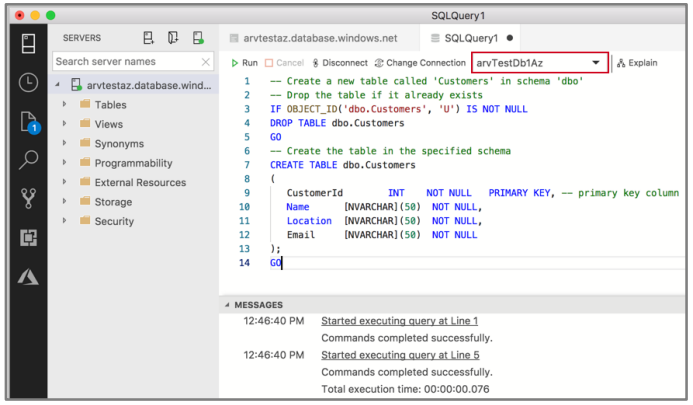
Azure Synapse Analytics



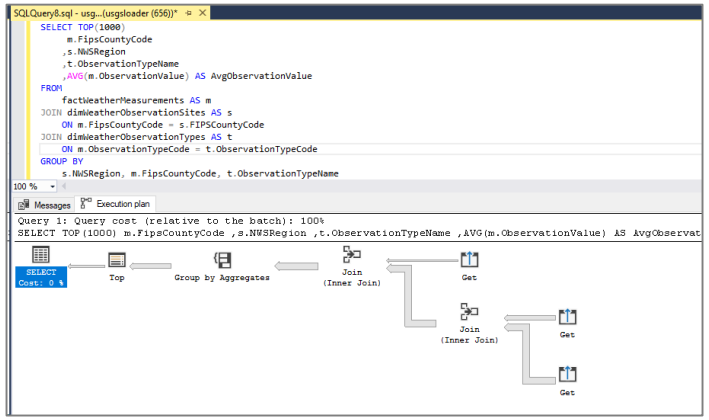
Visual Studio - SSDT database projects



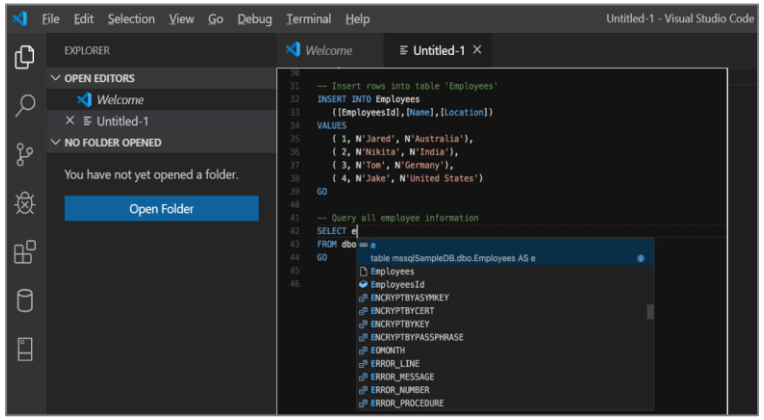
Azure Data Studio (queries, extensions etc.)



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



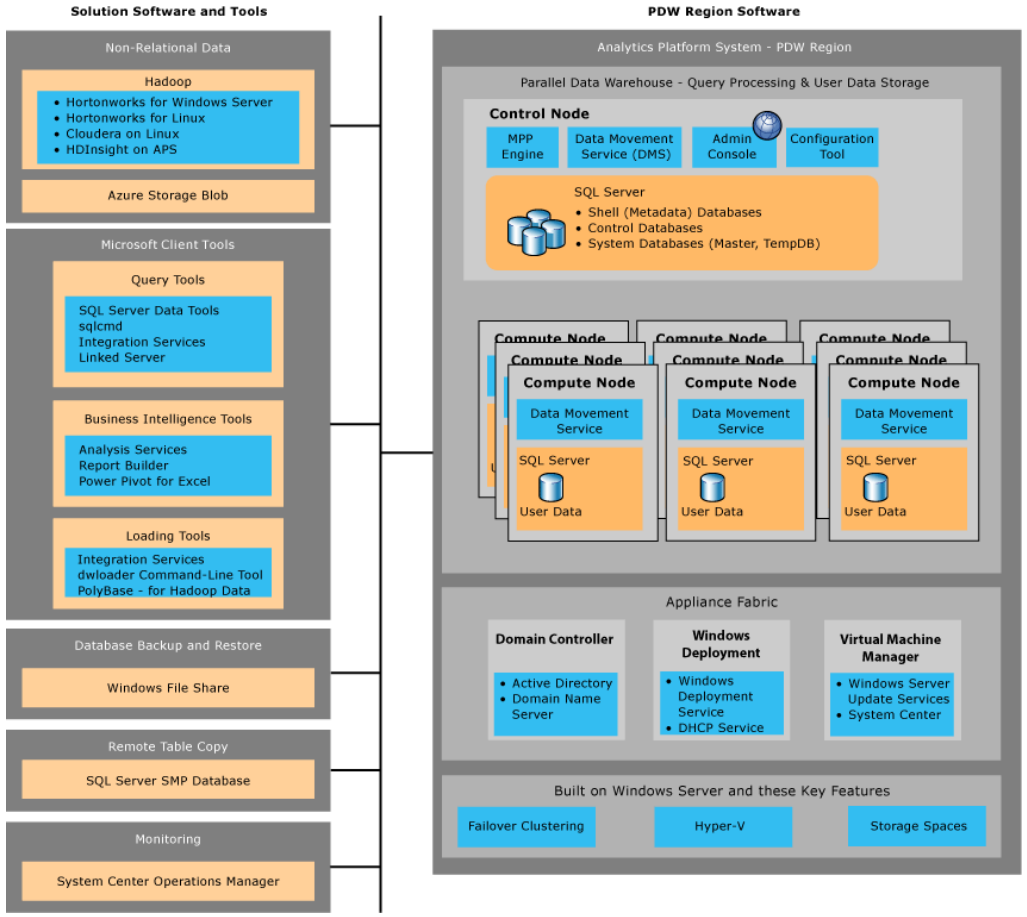
Visual Studio Code



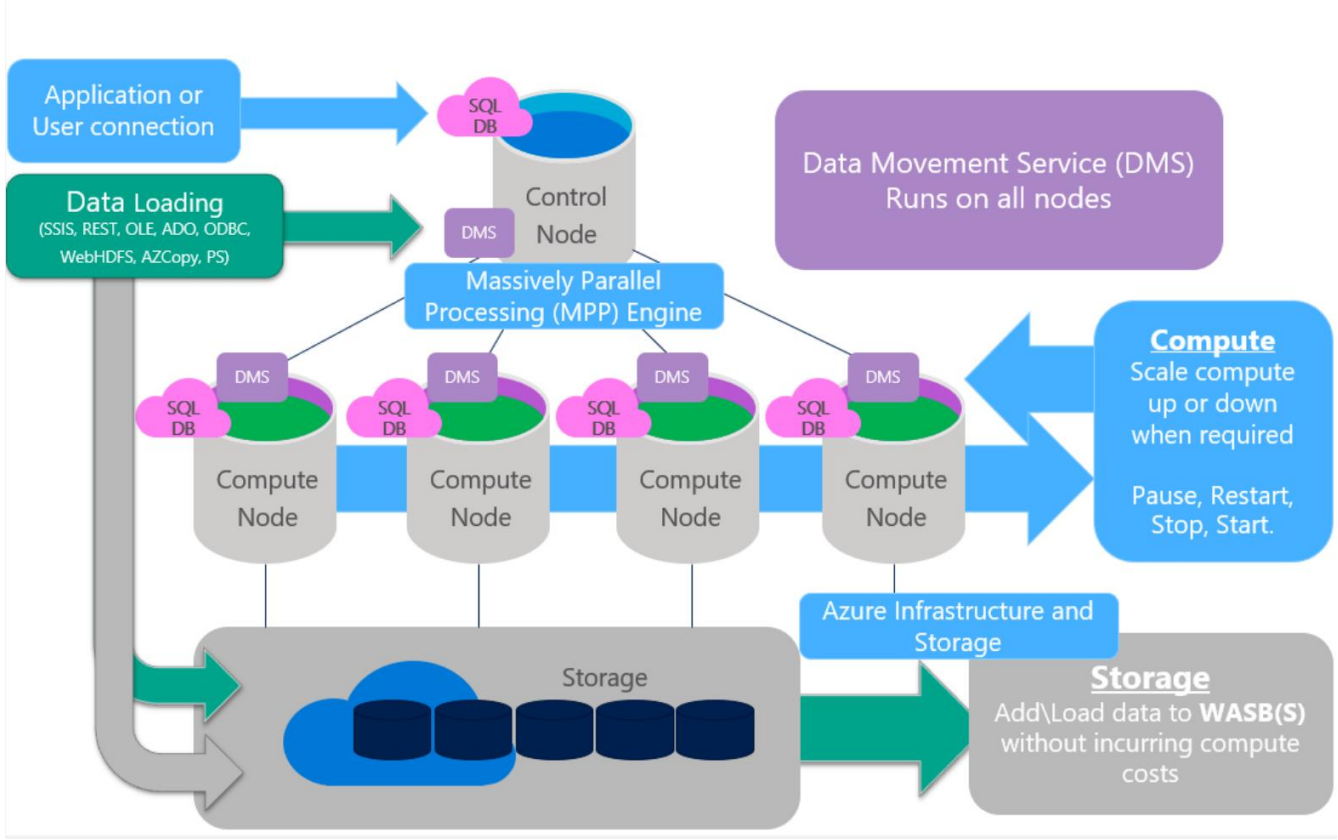
Azure Synapse Analytics

Synapse dedicated SQL pool

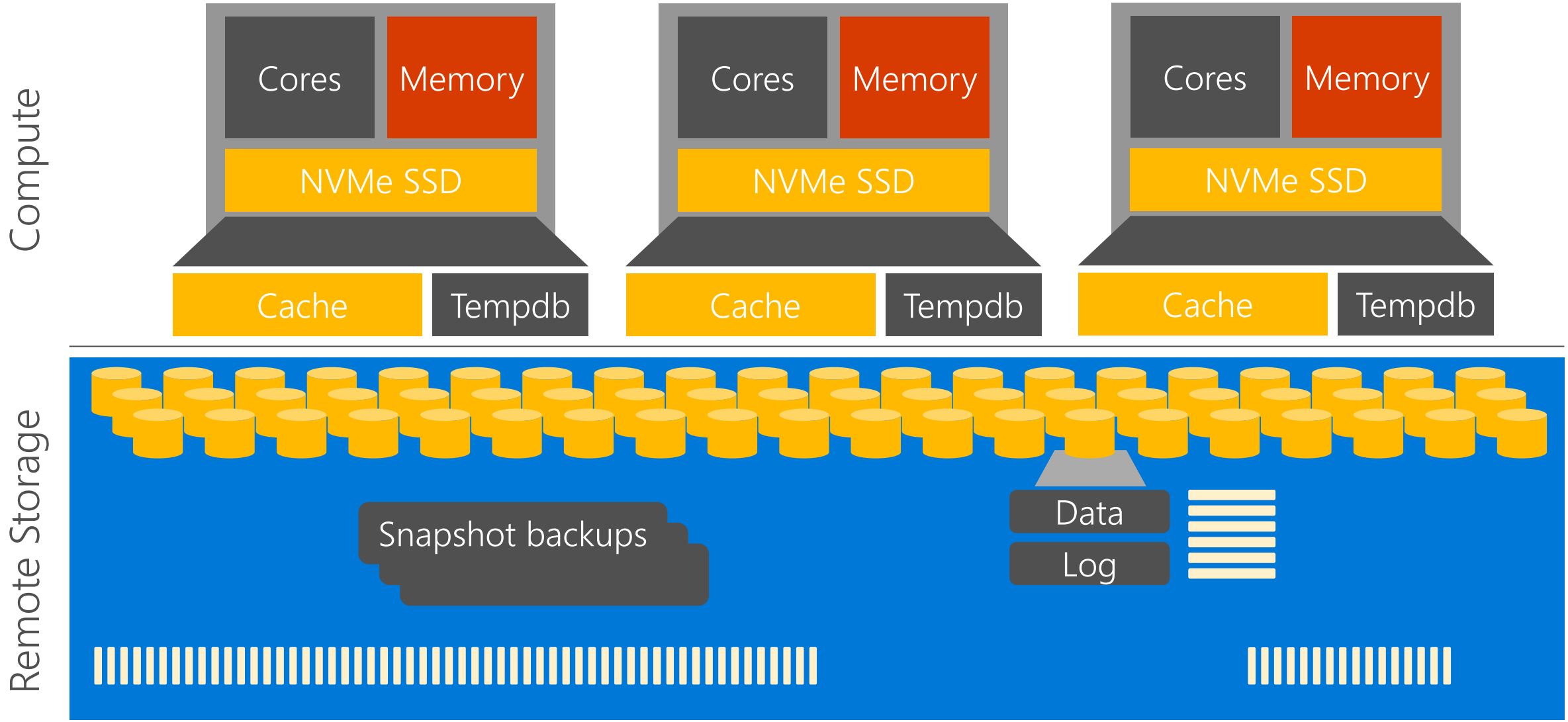
PDW Architecture (Parallel Data Warehouse)



Synapse SQL Pool Architecture

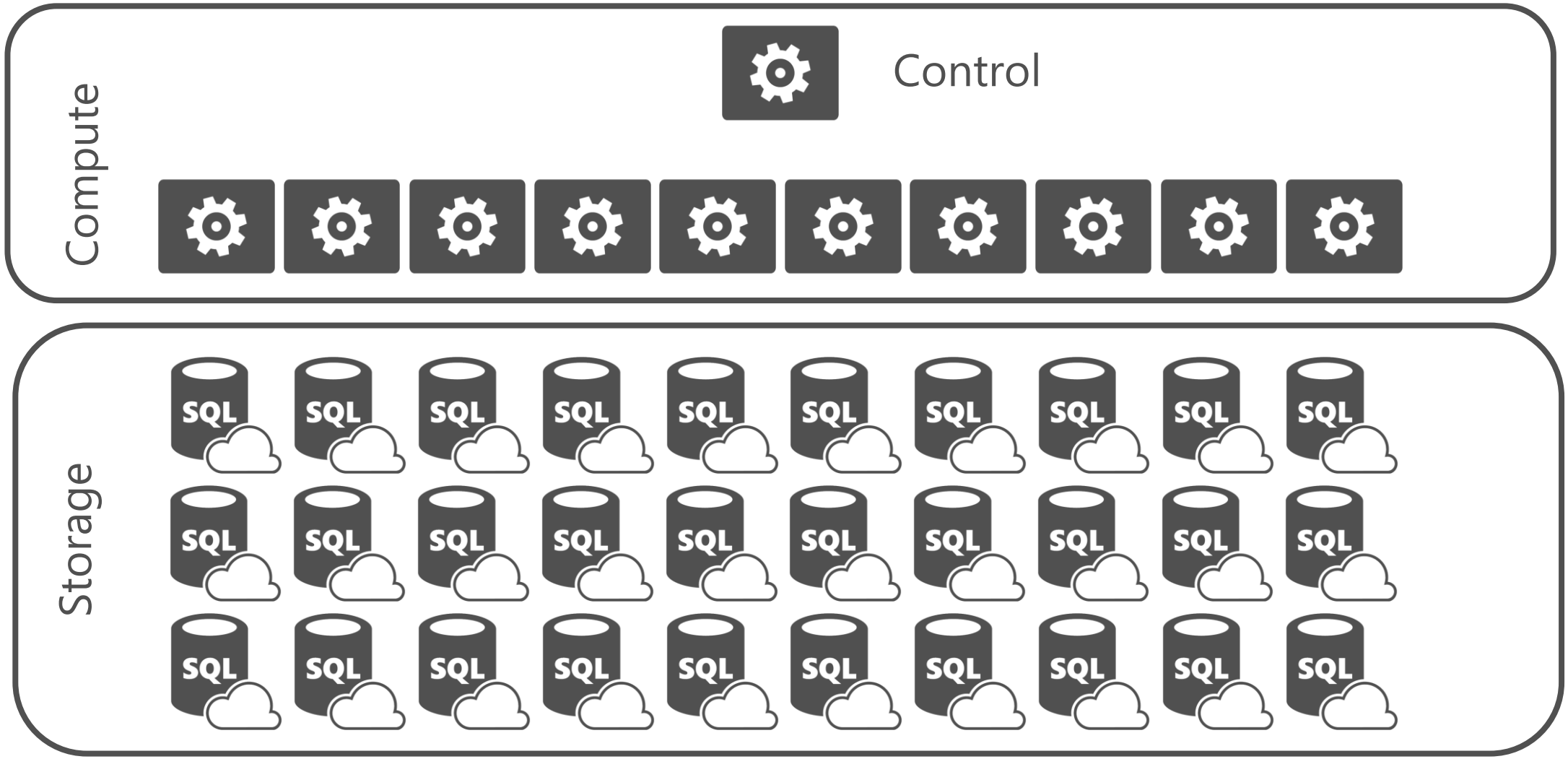


TIERED STORAGE MODEL AUTOMATES DATA TEMPERATURE

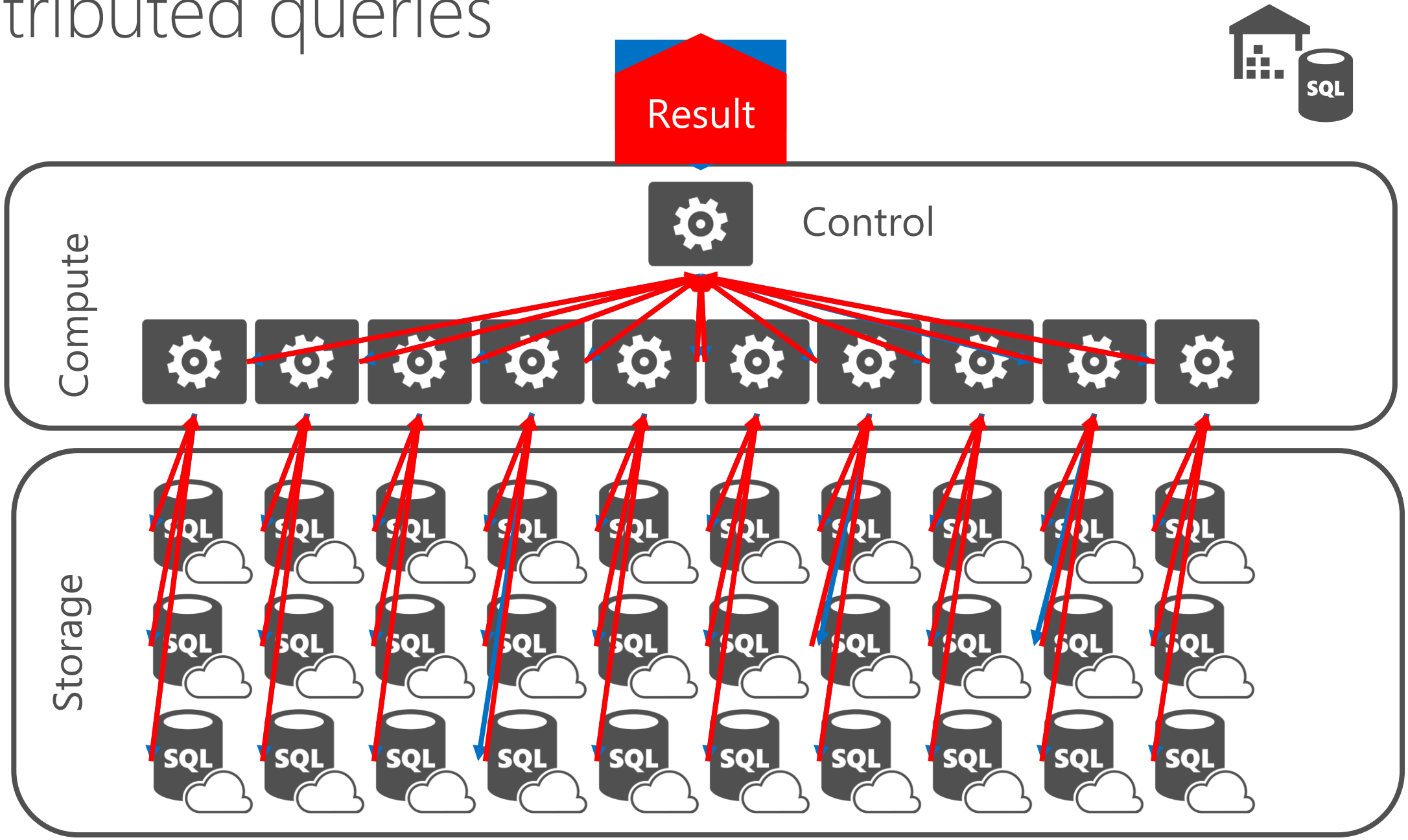


Synapse Analytics Architectural overview

Logical overview



Distributed queries



Scale operations

Data Warehouse Units

Normalized amount of compute

Converts to billing units i.e. what you pay

DWUc
100
200
300
400
500
600
1000
1200
1500
2000
3000
6000
...
30000

Management Interfaces

	Portal	PowerShell	T-SQL
Create	✓	✓	⊘
Pause and resume	✓	✓	⊘
Scale	✓	✓	✓

Scaling via T-SQL and PowerShell

T-SQL

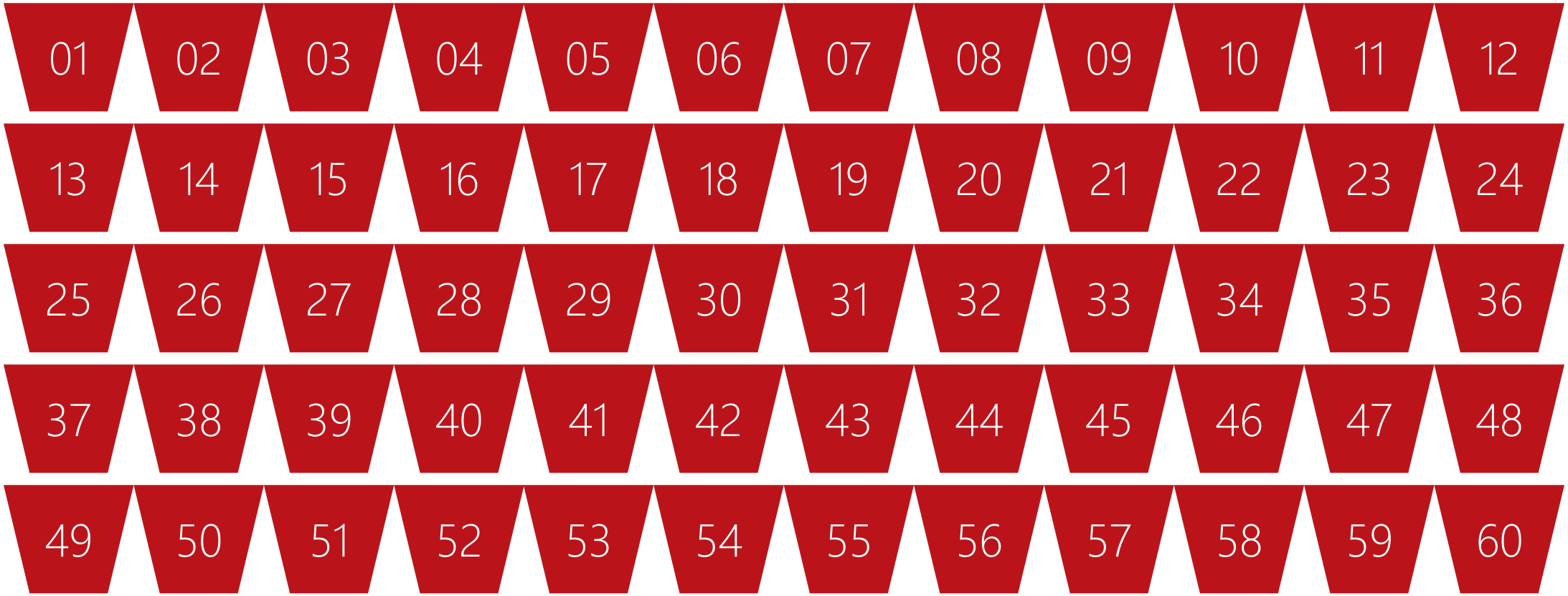
```
ALTER DATABASE ContosoRetailDW
MODIFY (service_objective = 'DW100');
```

PowerShell

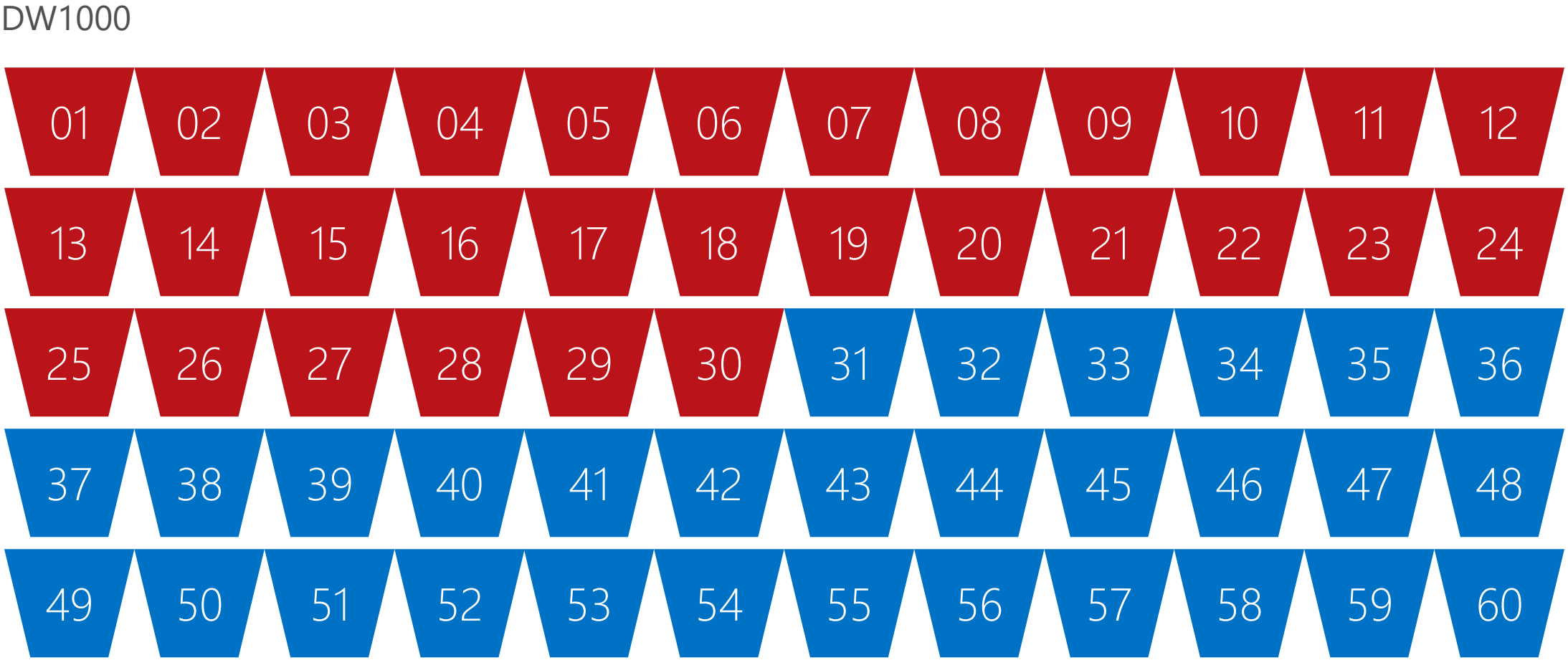
```
Set-AzureRmSqlDatabase `
    -ResourceGroupName "RG_name" `
    -ServerName "SRV_name" `
    -DatabaseName "DB_name" `
    -RequestedServiceObjectiveName "DW1000"
```

Mapping Compute in SQLDW

DW100

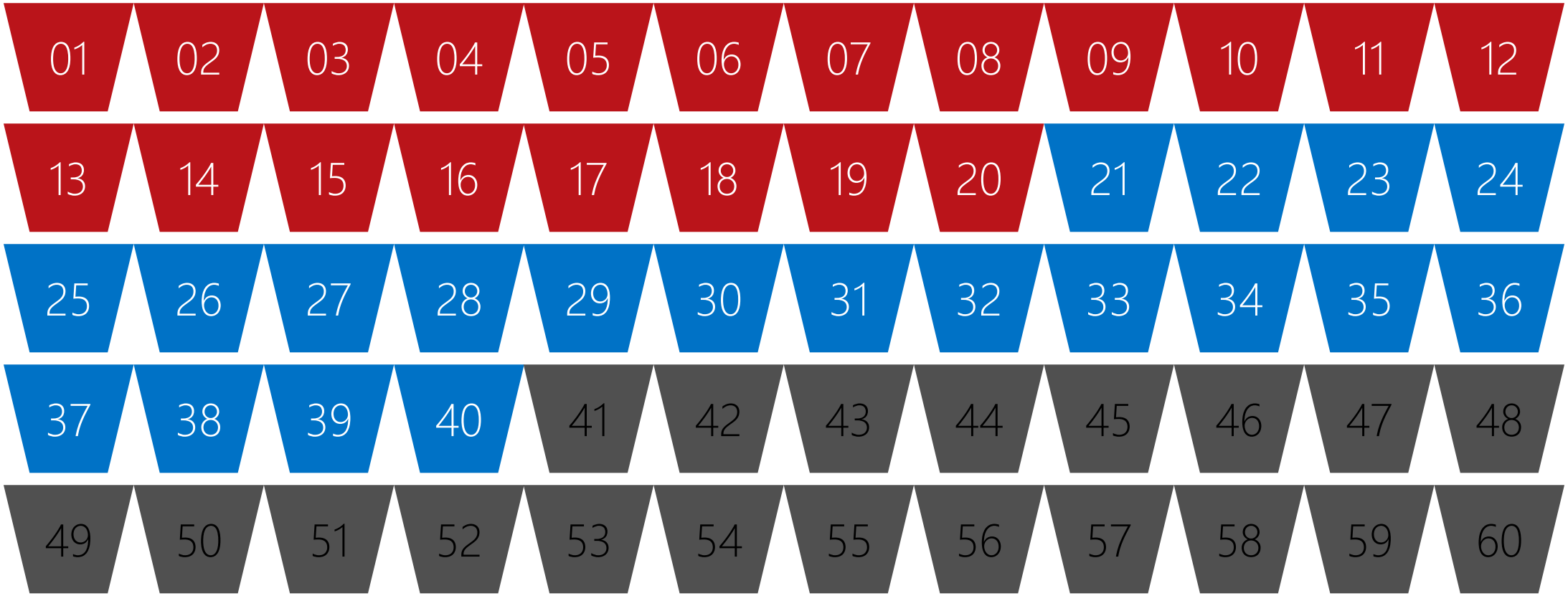


Mapping Compute in SQLDW



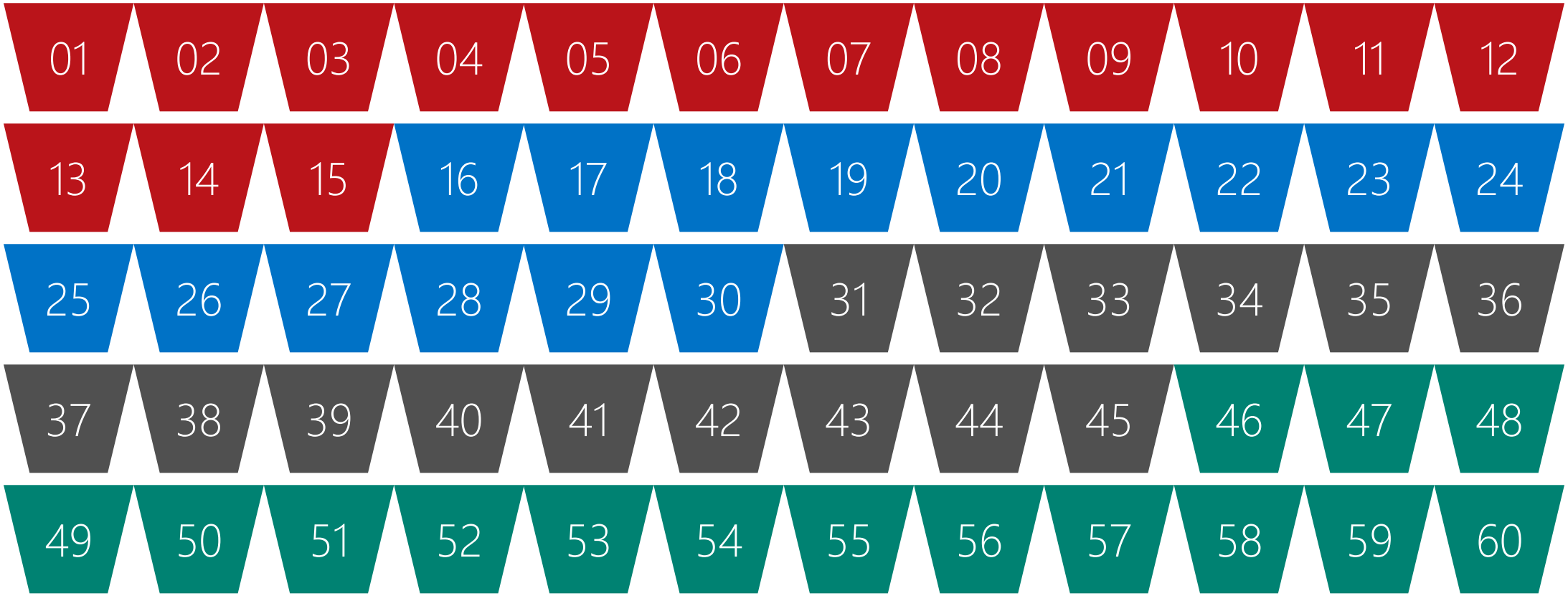
Mapping Compute in SQLDW

DW1500



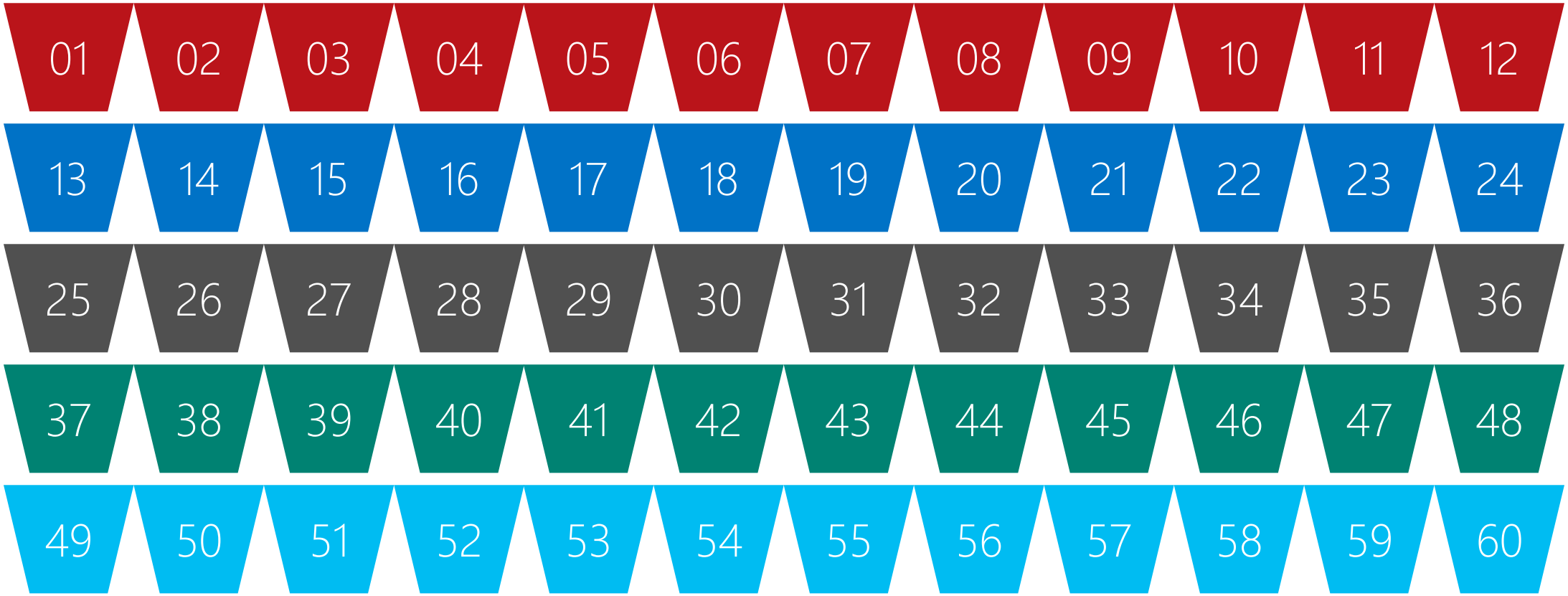
Mapping Compute in SQLDW

DW2000



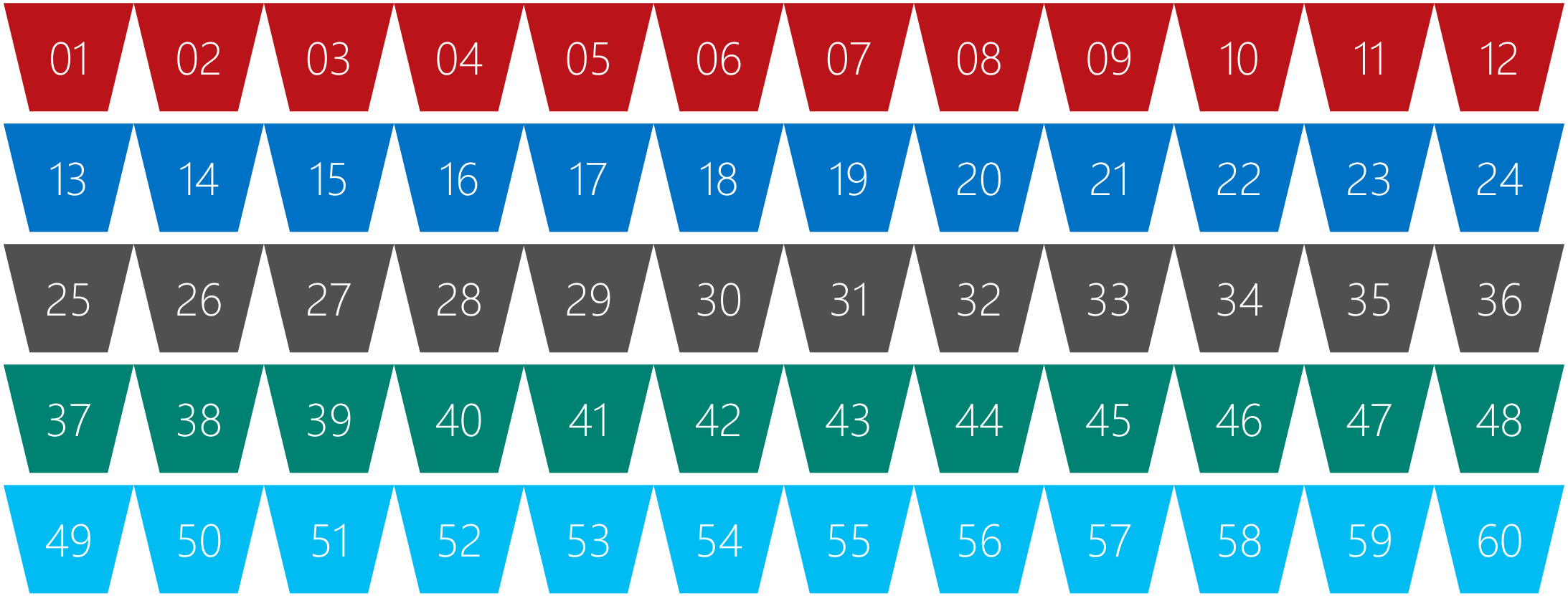
Mapping Compute in SQLDW

DW2500



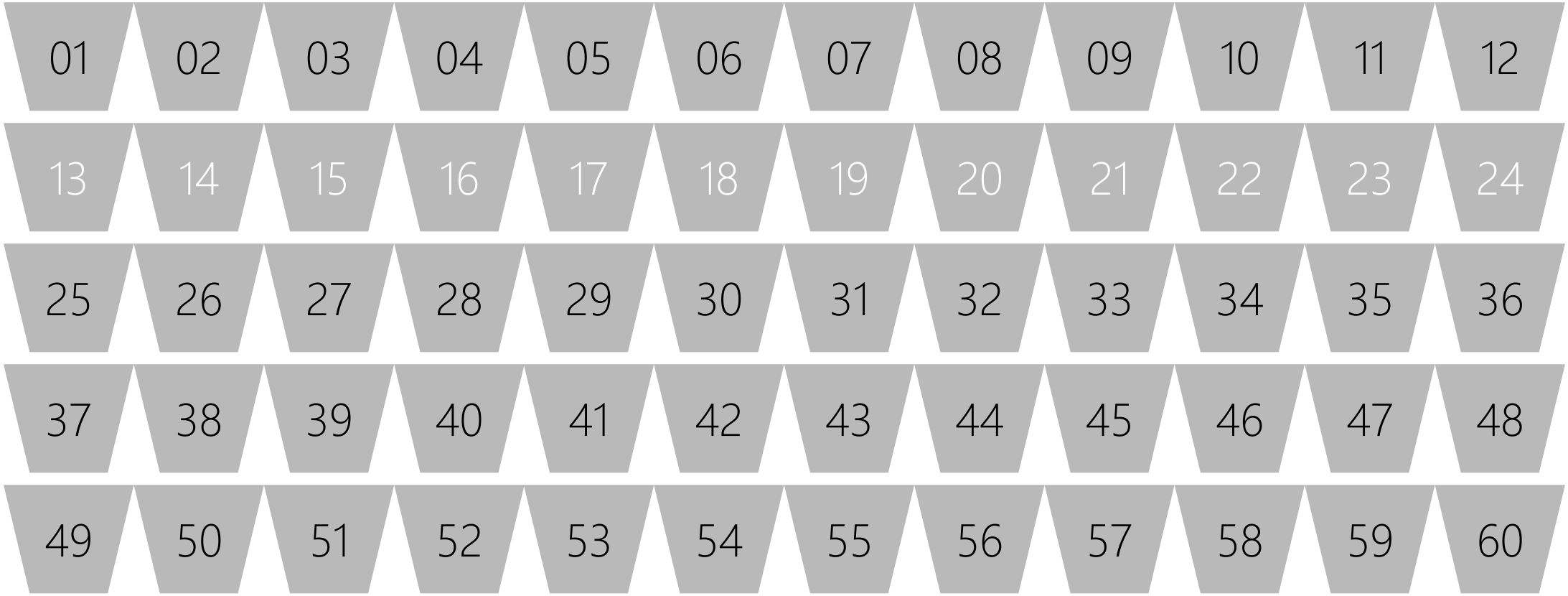
Pausing compute in SQLDW

DW2500



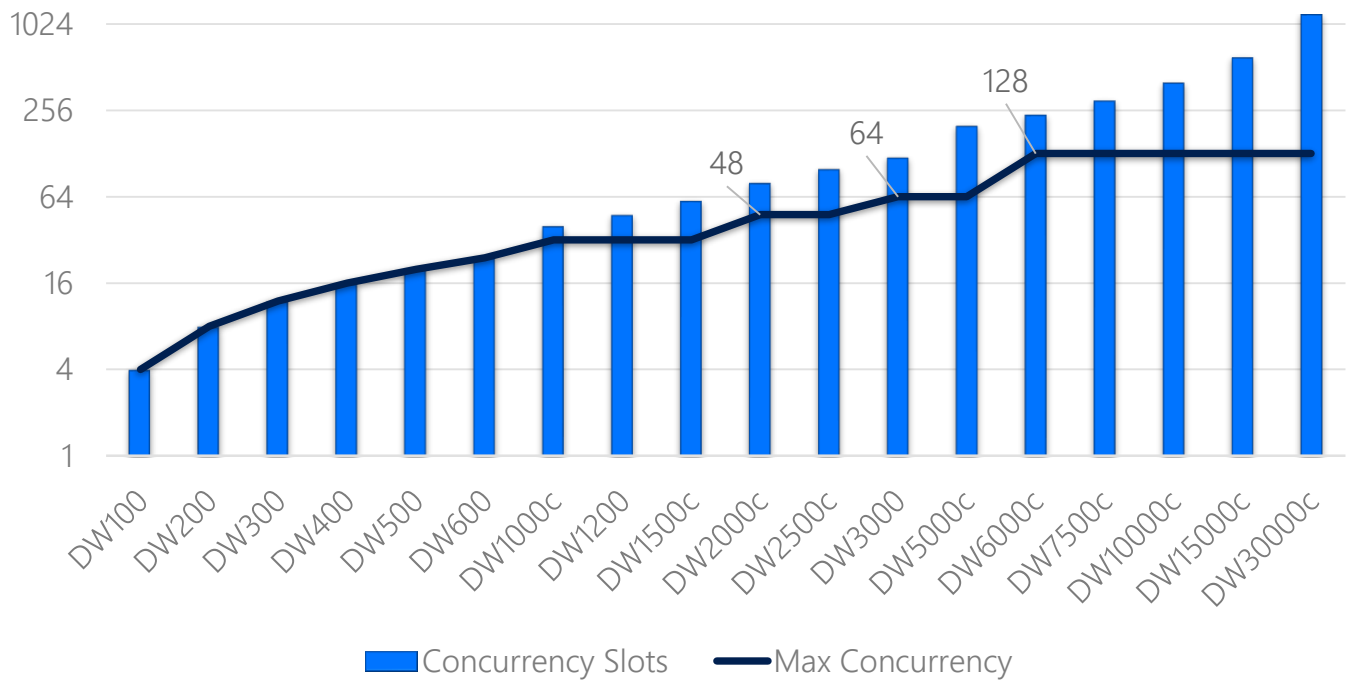
Resuming compute in SQLDW

DW2500



Concurrency

Concurrent Query and Slots



서비스 수준	최대 동시 쿼리 수	사용 가능한 동시성 슬롯 수	쿼리당 필요한 최소의 리소스% (작업 그룹 사용시)
DW100c	4	4	25%
DW200c	8	8	12.5%
DW300c	12	12	8%
DW400c	16	16	6.25%
DW500c	20	20	5%
DW1000c	32	40	3%
DW1500c	32	60	3%
DW2000c	48	80	2%
DW2500c	48	100	2%
DW3000c	64	120	1.5%
DW5000c	64	200	1.5%
DW6000c	128	240	0.75%
DW7500c	128	300	0.75%
DW10000c	128	400	0.75%
DW15000c	128	600	0.75%
DW30000c	128	1200	0.75%

Resource Class

정적 리소스 클래스

- 데이터의 집합 또는 크기가 고정적일 때 사용
- 현재 성능 수준에 관계없이 동일한 양의 메모리를 할당
- 서비스 성능 수준을 높이면 더 많은 쿼리를 수행 가능

- 정적 리소스 클래스 종류
  -. Staticrc10, Staticrc20, Staticrc30, Staticrc40, Staticrc50, Staticrc60, Staticrc70, Staticrc80

동적 리소스 클래스

- 데이터 양이 증가하거나 변화하는 경우에 적합
- 서비스 수준을 높이면 쿼리에 더 많은 메모리 할당

- 동적 리소스 클래스 종류
  -. smallrc, mediumrc, largerc, xlargerc

- 리소스 클래스 메모리 할당

서비스 수준	smallrc	mediumrc	largerc	xlargerc
DW100c	25%	25%	25%	70%
DW200c	12.5%	12.5%	22%	70%
DW300c	8%	10%	22%	70%
DW400c	6.25%	10%	22%	70%
DW500c	5%	10%	22%	70%
DW1000c – DW30000c	3%	10%	22%	70%

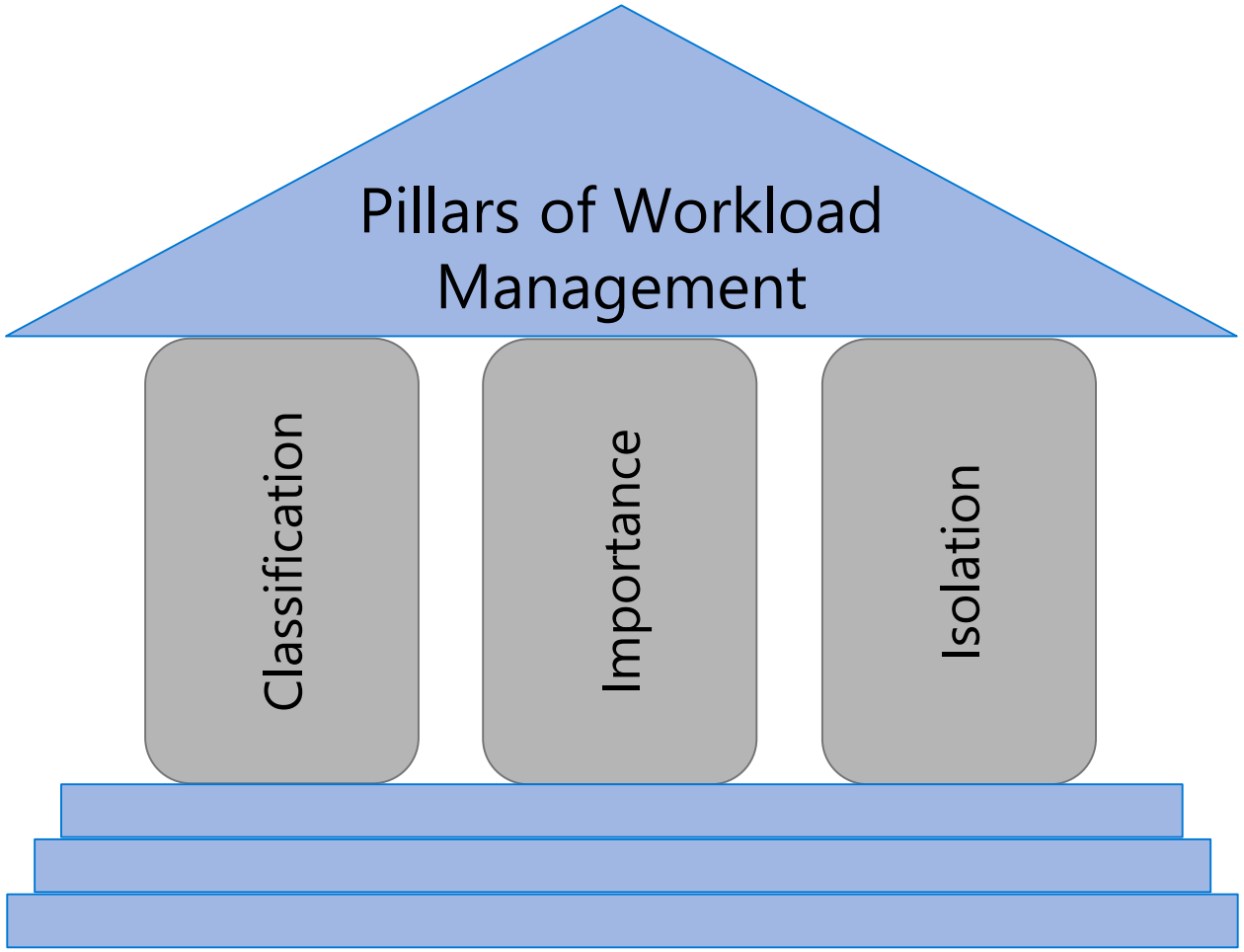
Workload Management

Overview

리소스의 효율적인 관리를 위해 사용

워크로드 관리 3대 요소

- 워크로드 분류 – 작업 요청을 그룹에 할당하고 중요도 설정
- 워크로드 중요도 – 그룹에 설정된 중요도에 따라 리소스의 액세스에 영향 받음
- 워크로드 독립성 – 워크로드 그룹에 리소스 예약



Workload Isolation

Overview

- 워크로드 그룹에 고정 리소스를 할당
- 로드 중인 다양한 리소스에 대한 최대 및 최소 사용량을 할당하며 Synapse SQL을 Offline하지 않고 리소스 조정 가능

Benefits

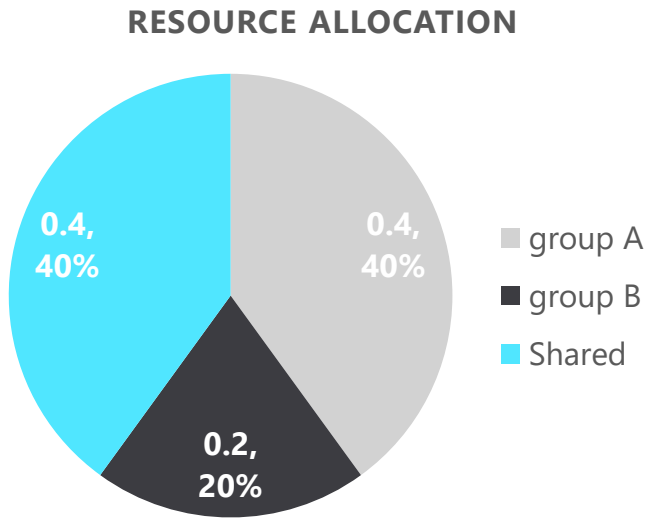
- 요청 그룹에 대한 리소스 예약
- 요청 그룹이 사용할 수 있는 리소스 양 제한
- 중요도에 따라 액세스되는 공유 리소스
- 쿼리 타임아웃 설정에 의한 DBA 업무 감소

Monitoring DMVs

sys.workload_management_workload_groups

Query to view configured workload group.

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



Workload Classification & Importance

워크로드 분류

- 미리 생성된 워크로드의 분류에 따라 리소스를 매핑
  워크로드 중요도와 함께 사용하여 리소스를 효과적으로 공유 가능

워크로드 중요도

- 워크로드 중요도에 따라 대기열에 관계없이 우선 순위가
  높은 쿼리가 리소스를 즉시 액세스 할 수 있음
- 워크로드 중요도를 생략했을 경우 중요도는 NORMAL 로 설정됨

Monitoring DMVs

sys.workload_management_workload_classifiers
sys.workload_management_workload_classifier_details

Query DMVs to view details about all active workload classifiers.

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

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

User & Authentication

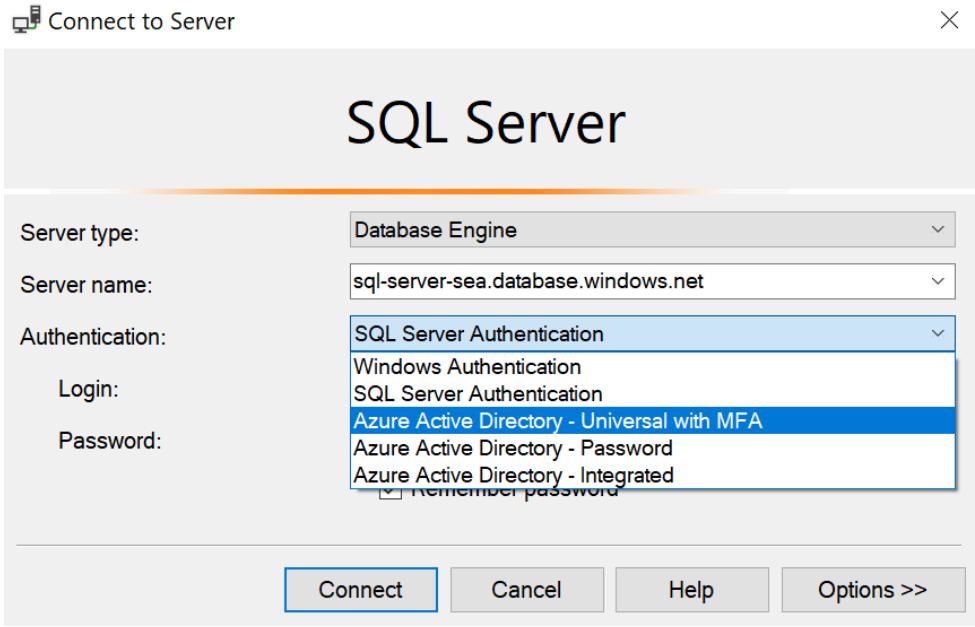
-- Master Database

```
CREATE LOGIN MedRCLogin WITH PASSWORD = 'rladydtjs$1971';

CREATE USER LoadingUser FOR LOGIN MedRCLogin;
```

-- Sqlpool Database

```
CREATE USER LoadingUser FOR LOGIN MedRCLogin;

GRANT CONTROL ON DATABASE::[yongdw] to LoadingUser;

EXEC sp_addrolemember 'mediumrc', 'LoadingUser';
```

```
Create Login Mary@domainname.net From EXTERNAL PROVIDER;
Create User Mary From Login Mary@domainname.net;
```



Tables & Distribution

Table Distribution Options

	Hash Distributed	Round Robin (Default)	Replicated
분산 방법	• 해시 알고리즘 리턴 값에 의해 노드로 데이터 분산 • 해시 알고리즘에 의한 결과값이 같으면 같은 노드에 데이터 분산 • 해쉬 키 값은 Single Column Only	• 모든 노드에 데이터가 분산 • 비용 대비 단순한 저장 방식	• 모든 노드에 데이터 복제
장점	• 아래 작업 수행 시 속도가 빠름 **COUNT (DISTINCT <hashed_key>)** **OVER PARTITION BY <hashed_key>** **JOIN <table_name> ON <hashed_key>** **GROUP BY <hashed_key>**	• 초기 적재 속도가 빠름	• 쿼리 계획을 단순화하고 데이터 이동 감소 • Hash 테이블과 조인 시 최적
사용 예	• Fact Table	• 초기 데이터 이관 시 Data Load • 스테이징 테이블에 사용	• Dimension Table (Fact Table 을 설명해주는 Table) • Small Table
유의 사항	• 특정 노드로 데이터 쏠림 발생 • NULLS, Default 값 사용시 부적합 • Hash key 값은 update 되면 안됨	• 쿼리 수행 시 노드간 데이터 이동 발생	• 한 노드에서 두개의 복제 테이블이 조인 시 많은 공간이 소비됨

Creating tables : Round Robin, Hash

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

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

Creating tables : Replicated

```
CREATE TABLE dbo.DimCustomer
(    CustomerKey           int             NOT NULL
,    GeographyKey          int             NULL
,    CustomerAlternateKey  nvarchar(15)    NOT NULL
,    Title                 nvarchar(8)     NULL
,    FirstName             nvarchar(50)    NULL
,    LastName              nvarchar(50)    NULL
,    BirthDate             date            NULL
,    Gender                nvarchar(1)     NULL
,    EmailAddress          nvarchar(50)    NULL
,    YearlyIncome          money           NULL
,    DateFirstPurchase     date            NULL
)
WITH
(    CLUSTERED COLUMNSTORE INDEX
,    DISTRIBUTION = REPLICATED
)
```

Hash Distributed

```
CREATE TABLE ProductSales
WITH (DISTRIBUTION=HASH(AccountID))
AS...
```

ProductSales – Raw Data

AccountI	SalesAmt	…
47	$1,234.36	…
36	$2,345.47	…
14	$3,456.58	…
25	$4,567.69	…
48	$5,678.70	…
37	$6,789.81	…
…	…	…

Hash(47)

Round Robin Distributed

```
CREATE TABLE ProductSales
WITH (DISTRIBUTION = ROUND_ROBIN)
AS ...
```

ProductSales – Raw Data

AccountID	SalesAmt	...
47	$1,234.36	...
36	$2,345.47	...
14	$3,456.58	...
25	$4,567.69	...
47	$5,678.70	...
37	$6,789.81	...
42	$1,632.25	...
88	$4,453.21	...
52	$7,892.81	...
91	$9,549.64	...
88	$2,498.14	...
42	$3,145.99	...
23	$3,145.99	...

데이터의 이동은 왜 발생하는가?

Data has to be co-located to operated on…

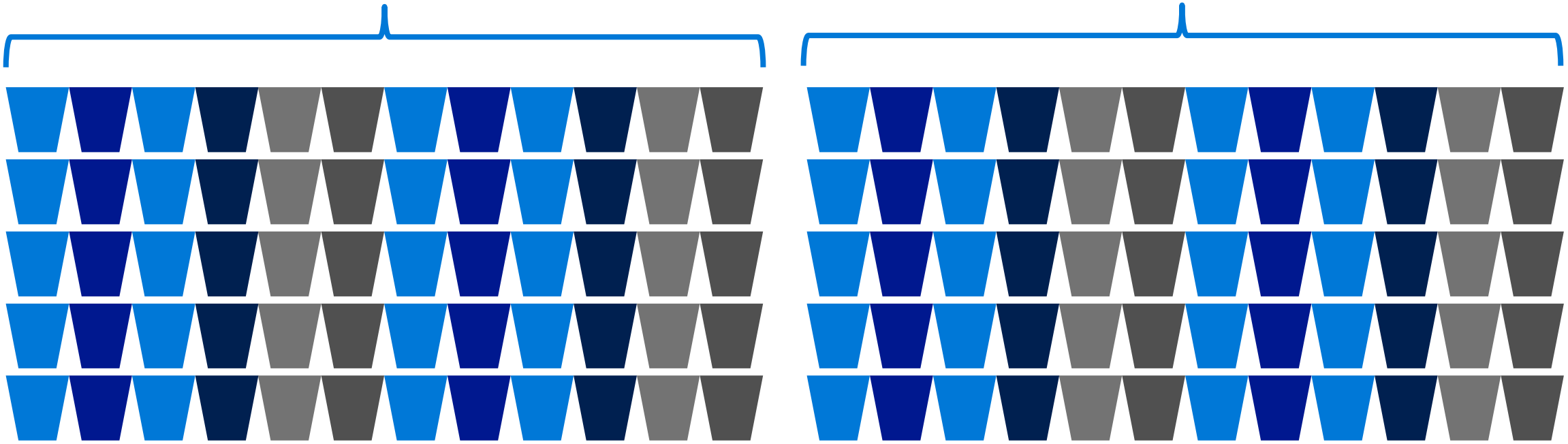
일반 원인:

- 호환 불가능한 조인
- 호환 불가능한 집계

HASH table Join – 데이터 이동 미발생

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

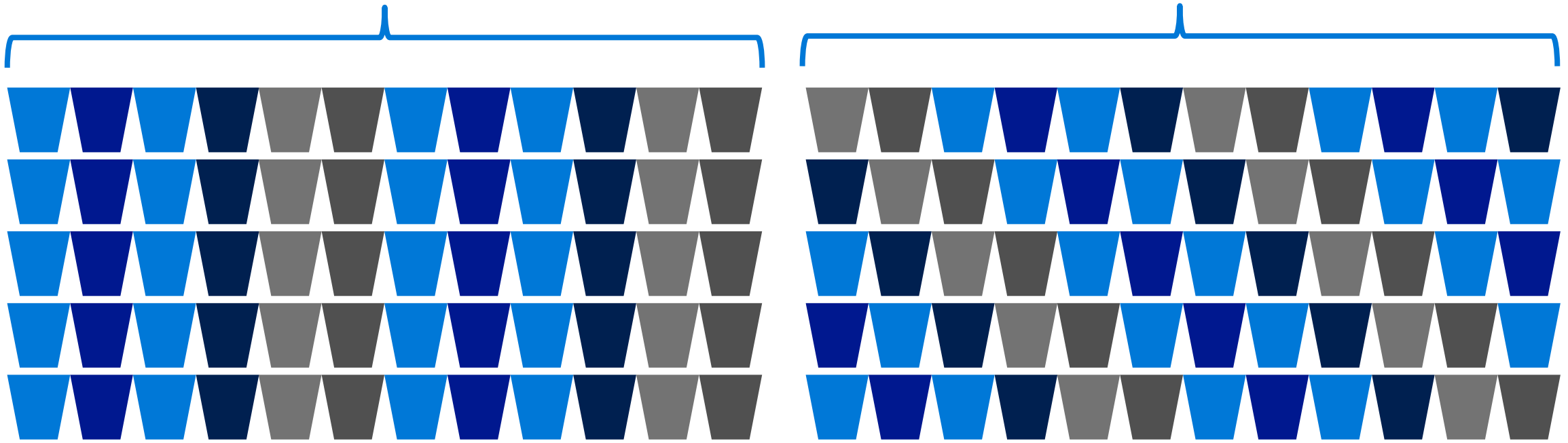
Web_Sales HASH([ProductKey])
[ProductKey] **INT** NULL



HASH table Join – 데이터 이동 발생

Store_Sales HASH([ProductKey])
[ProductKey] **INT** NULL

Web_Sales HASH([ProductKey])
[ProductKey] **BIGINT** NULL



데이터 이동 Case Study

Aggregation - Compatible

Resolved completely on each compute node
No Data Movement

1. Hash Distribution Key is contained in the group by keys
2. Count Distinct on distribution key

```
-- FactOnlineSales distributed on ProductKey

SELECT      COUNT_BIG(*)
FROM        [cso].[FactOnlineSales]
GROUP BY    [ProductKey]

SELECT  COUNT_BIG(DISTINCT ([ProductKey]))
FROM    [cso].[FactOnlineSales]
```

Aggregation - Incompatible

Partially aggregated on each node
Shuffle move co-locates rows with same group by key

1. Table is round robin distributed
2. Hash Distribution key is not contained in group by keys
3. Count Distinct on non-distribution key or on round robin table

```
-- FactOnlineSales distributed on ProductKey

SELECT      COUNT_BIG(*)
FROM        [cso].[FactOnlineSales]
GROUP BY    [StoreKey]

SELECT  COUNT_BIG(DISTINCT [DateKey])
FROM    [cso].[FactOnlineSales]
```

Data Movement Types for a Query

DMS Operation	Description
ShuffleMoveOperation	Distribution → Hash algorithm → New distribution Changing the distribution column in preparation for join.
PartitionMoveOperation	Distribution → Control Node Aggregations - count(*) is count on nodes, sum of count
BroadcastMoveOperation	Distribution → Copy to all distributions Changes distributed table to replicated table for join.
TrimMoveOperation	Replicated table → Hash algorithm → Distribution When a replicated table needs to become distributed. Needed for outer joins.
MoveOperation	Control Node → Copy to all distributions Data moved from Control Node back to Compute Nodes resulting in a replicated table for further processing.

Tables – Partitions

Overview

- 테이블 데이터를 작은 그룹으로 분리하는 작업
- 대부분 날짜 컬럼을 파티션 키를 사용
- 모든 테이블에 파티션 기능 지원

RANGE RIGHT – Used for time partitions

RANGE LEFT – Used for number partitions

Benefits

- 데이터 로딩, 조회에 효율성 및 속도 증가
- 데이터 조회 시, 불필요한 스캔의 감소와 이로 인한 I/O 의 감소로 상당한 쿼리 속도 증가

```
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 – DW Distributions & Partitions Illustrated

논리적 Table 구조

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
...	...	...	...

물리적 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)

- 각 Shard 는 같은 Date로 파티션 되어있음
- 최적의 압축과 Clustered Columnstore table의 퍼포먼스 향상을 위해서는 각 파티션에 최소 100만 건의 row가 필요

Tables – Partitions

	logical_table_name	row_group_id	state	state_desc	total_rows	trim_reason_desc	physical_name
1	FactSalesQuotaCCI	3	3	COMPRESSED	514925	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_1
2	FactSalesQuotaCCI	3	3	COMPRESSED	512875	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_2
3	FactSalesQuotaCCI	3	3	COMPRESSED	511350	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_3
4	FactSalesQuotaCCI	3	3	COMPRESSED	509161	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_4
5	FactSalesQuotaCCI	3	3	COMPRESSED	512640	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_5
6	FactSalesQuotaCCI	3	3	COMPRESSED	512640	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_6
7	FactSalesQuotaCCI	3	3	COMPRESSED	512638	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_7
8	FactSalesQuotaCCI	3	3	COMPRESSED	511216	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_8
9	FactSalesQuotaCCI	3	3	COMPRESSED	511216	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_9
10	FactSalesQuotaCCI	3	3	COMPRESSED	512360	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_10
11	FactSalesQuotaCCI	3	3	COMPRESSED	511216	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_11
12	FactSalesQuotaCCI	3	3	COMPRESSED	510102	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_12
13	FactSalesQuotaCCI	3	3	COMPRESSED	509792	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_13
14	FactSalesQuotaCCI	3	3	COMPRESSED	512640	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_14
15	FactSalesQuotaCCI	3	3	COMPRESSED	512388	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_15
16	FactSalesQuotaCCI	3	3	COMPRESSED	513046	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_16
17	FactSalesQuotaCCI	3	3	COMPRESSED	513105	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_17
18	FactSalesQuotaCCI	3	3	COMPRESSED	512640	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_18
19	FactSalesQuotaCCI	3	3	COMPRESSED	512607	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_19
20	FactSalesQuotaCCI	3	3	COMPRESSED	511583	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_20
21	FactSalesQuotaCCI	3	3	COMPRESSED	512156	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_21
22	FactSalesQuotaCCI	3	3	COMPRESSED	512479	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_22
23	FactSalesQuotaCCI	3	3	COMPRESSED	511480	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_23
24	FactSalesQuotaCCI	3	3	COMPRESSED	512171	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_24
25	FactSalesQuotaCCI	3	3	COMPRESSED	512753	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_25
26	FactSalesQuotaCCI	3	3	COMPRESSED	511216	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_26
27	FactSalesQuotaCCI	3	3	COMPRESSED	509843	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_27
28	FactSalesQuotaCCI	3	3	COMPRESSED	511065	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_28
29	FactSalesQuotaCCI	3	3	COMPRESSED	510639	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_29
30	FactSalesQuotaCCI	3	3	COMPRESSED	510366	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_30
31	FactSalesQuotaCCI	3	3	COMPRESSED	511247	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_31
32	FactSalesQuotaCCI	3	3	COMPRESSED	511917	BULKLOAD	Table_0916c48a65e14db285dfed9c57386023_32

Tables – Indexes

Clustered Columnstore index (Default Primary)

- 최적의 데이터 압축 지원
- 최고의 쿼리 퍼포먼스 지원

Clustered index (Primary)

- 단일 행, 극소수 행을 조회할 때 최적

  (Multi-Column Index 가능, But 1개 이상의 Index 생성 불가)

Heap (Primary)

- 임시 데이터를 Loading 시 최적
- 사이즈가 작은 Look up Table 조회 시 사용

Nonclustered indexes (Secondary)

- Table에 Multiple Columns Index 를 허용
- Single Table에 여러 개의 Multiple Non-Clustered 허용
- Primary Indexes 대안으로 Non-Clustered Indexes 생성 가능
- Lookup 테이블 조회에 성능 기대

```
-- 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);
```

Creating tables

```
CREATE TABLE [dbo].[DimStore]
(
    [StoreKey]          int             NOT NULL
,   [GeographyKey]      int             NOT NULL
,   [StoreName]         nvarchar(100)   NOT NULL
,   [StoreType]         nvarchar(15)        NULL
,   [StoreDescription]  nvarchar(300)   NOT NULL
,   [Status]            nvarchar(20)    NOT NULL
,   [OpenDate]          datetime        NOT NULL
,   [CloseDate]         datetime            NULL
,   [ETLLoadID]         int                 NULL
,   [LoadDate]          datetime            NULL
,   [UpdateDate]        datetime            NULL
)
WITH
(   CLUSTERED INDEX([StoreKey])
,   DISTRIBUTION = ROUND_ROBIN
)
;
```

Row

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

Column

Distribution

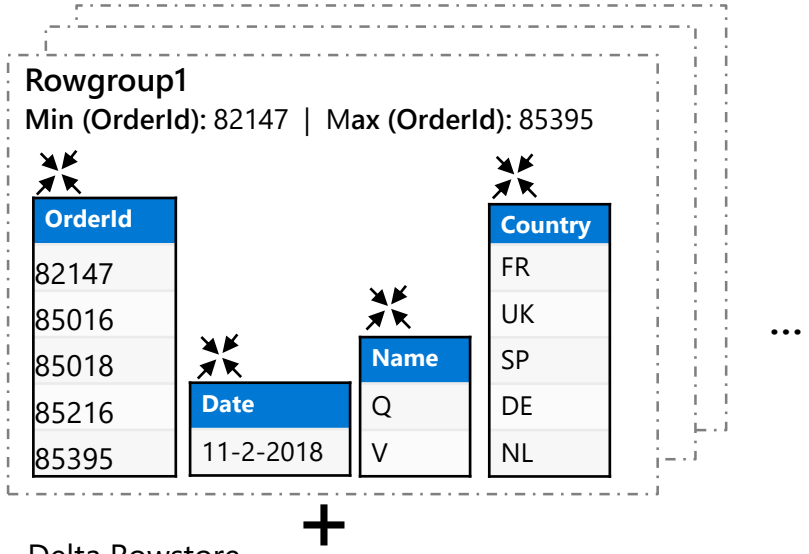
Synapse SQL (provisioned) Columnstore Tables

Logical table structure

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

Clustered columnstore index

(OrderId)



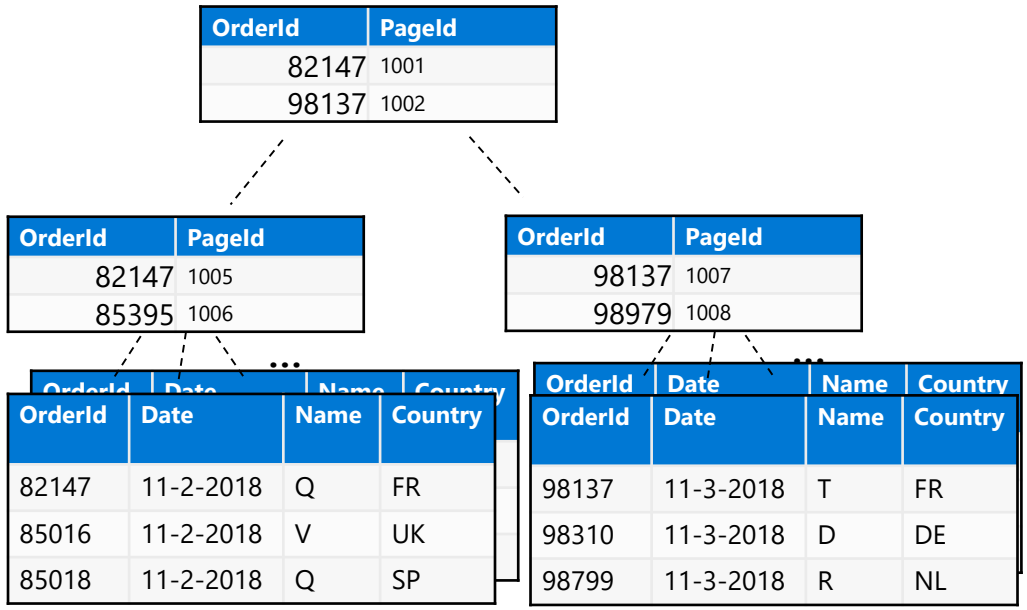
Delta Rowstore

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

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

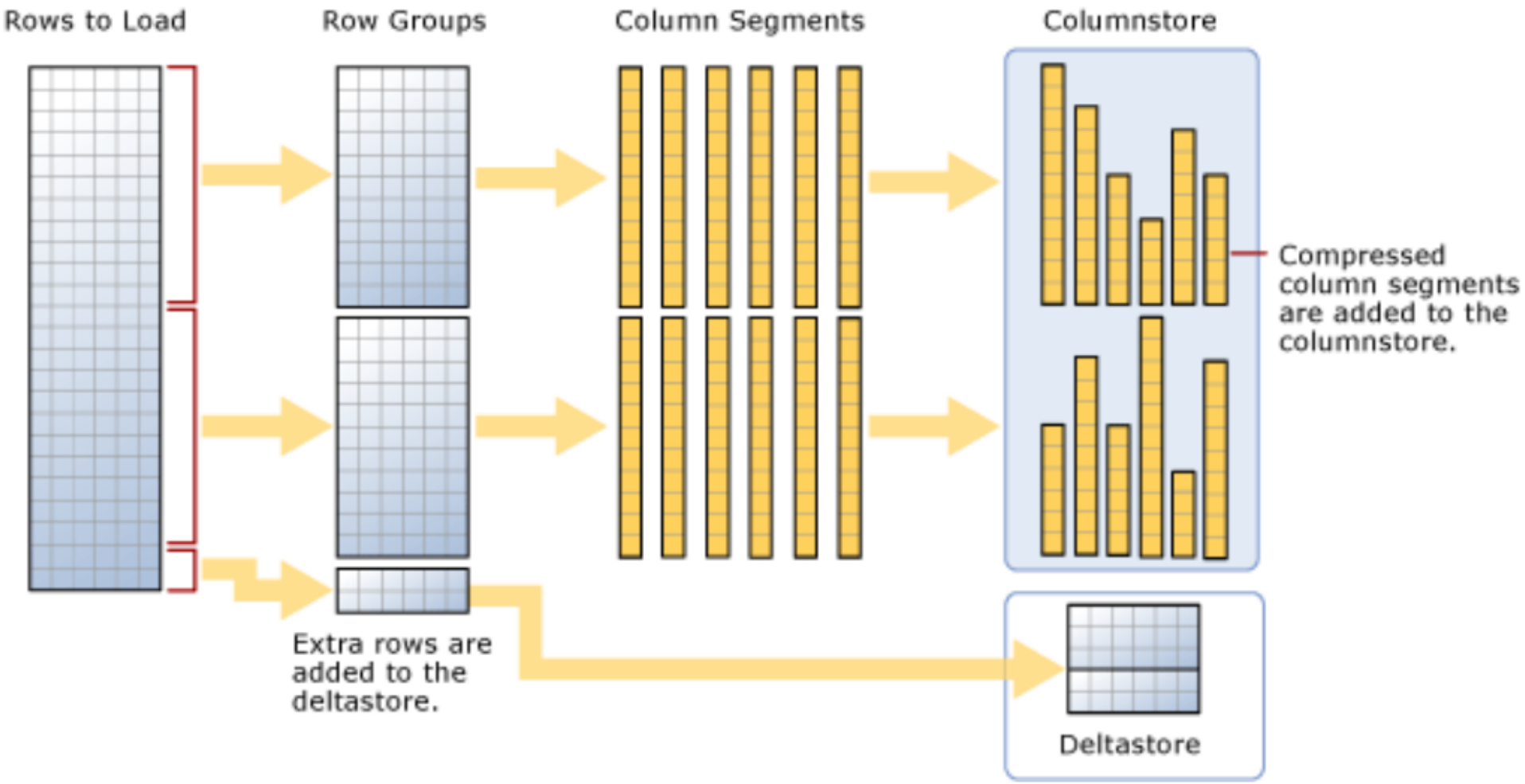
Clustered/Non-clustered rowstore index

(OrderId)



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

Columnstore Storage Model



Ordered Clustered Columnstore Indexes

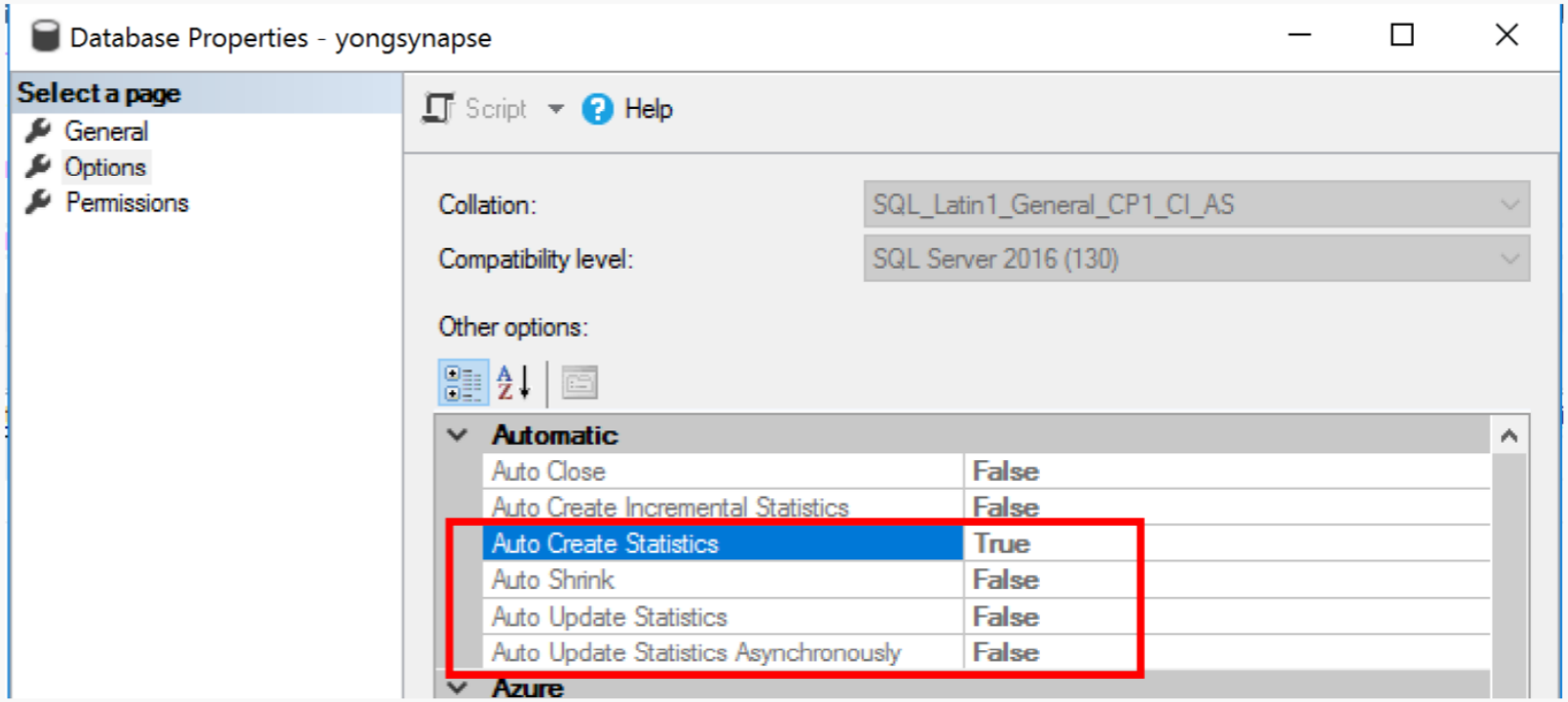
Overview

Ordered Clustered Columnstore Index를 사용하면 테이블 조회 시 필요한 세그먼트만 읽음으로써 쿼리 시간을 크게 단축할 수 있습니다.

```
-- 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')
```

Statistics



CREATE STATISTICS [statistics_name] ON [schema_name].[table_name]([column_name]) WITH FULLSCAN;

CREATE STATISTICS col1_stats ON dbo.table1 (col1) WITH SAMPLE = 50 PERCENT;

UPDATE STATISTICS [dbo].[table1] ([stats_col1]);

‘Optimized for Elasticity’ Sizing & Storage tiers

Sizing factors

Database capacity
Tempdb
Concurrency & Memory
Load
Transaction size
Memory management

DWU

Sizing Node Count / CPU & Memory

Node Count

Performance level	Compute nodes
DW100c	1
DW200c	1
DW300c	1
DW400c	1
DW500c	1
DW1000c	2
DW1500c	3
DW2000c	4
DW2500c	5
DW3000c	6
DW5000c	10
DW6000c	12
DW7500c	15
DW10000c	20
DW15000c	30
DW30000c	60

CPU (Hyperthreaded vCore)

Convert to DTU = DWU * 9

vCore = DTU / 100

= DWU * 9 / 100

= (6000 * 9) / 100

= 54000 / 100

= 540 vCore

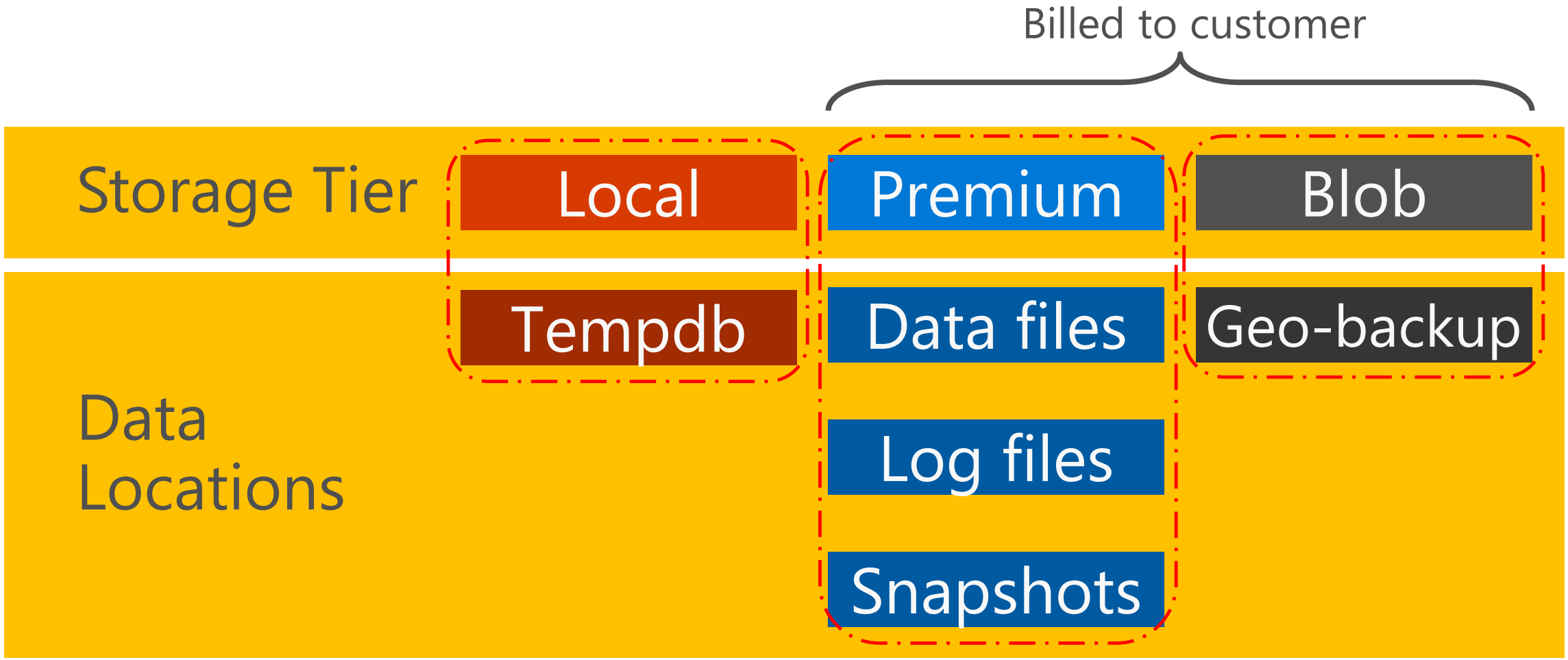
Memory

DWU * 60

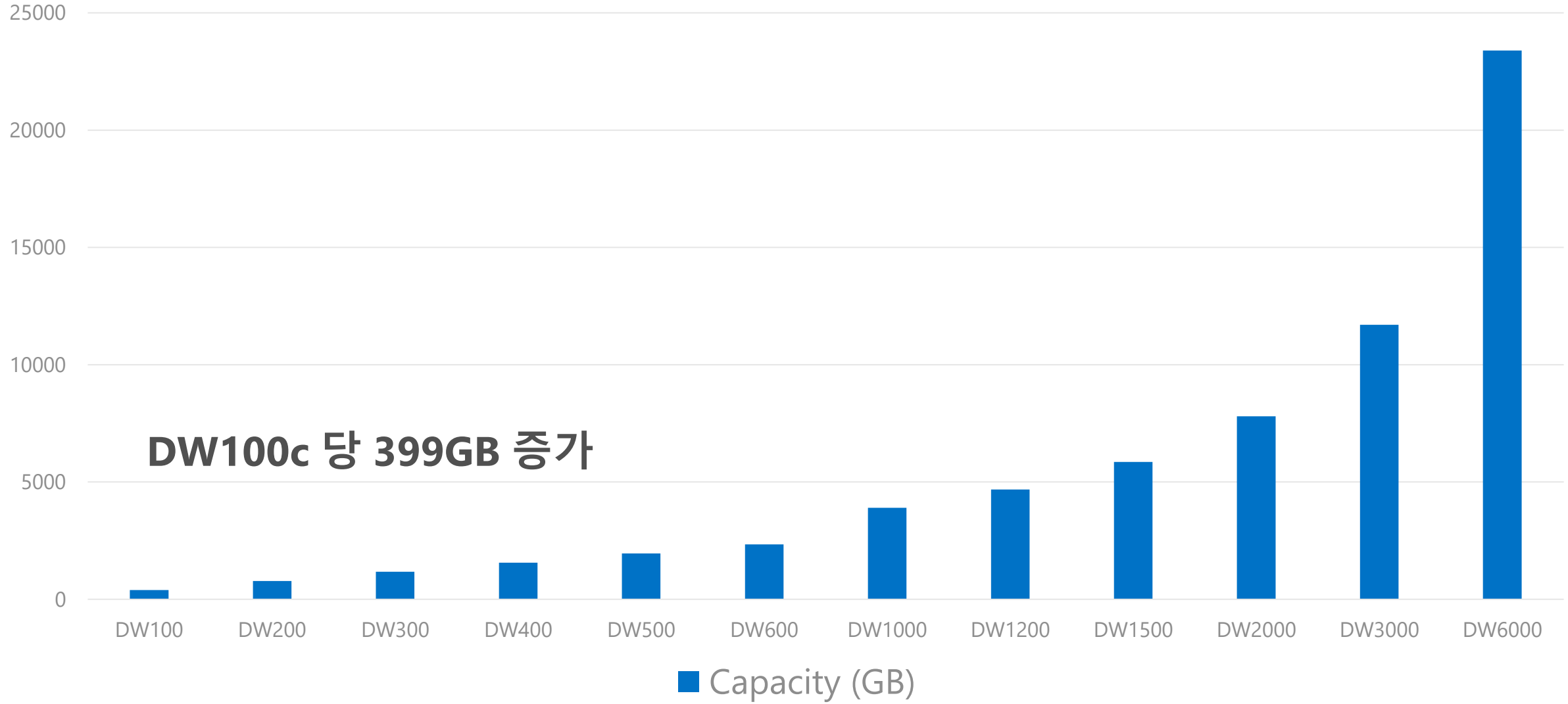
(DW6000c * 60 = 3.6 Tb)

Performance level	Memory per data warehouse (GB)
DW100c	60
DW200c	120
DW300c	180
DW400c	240
DW500c	300
DW1000c	600
DW1500c	900
DW2000c	1,200
DW2500c	1,500
DW3000c	1,800
DW5000c	3,000
DW6000c	3,600
DW7500c	4,500
DW10000c	6,000
DW15000c	9,000
DW30000c	18,000

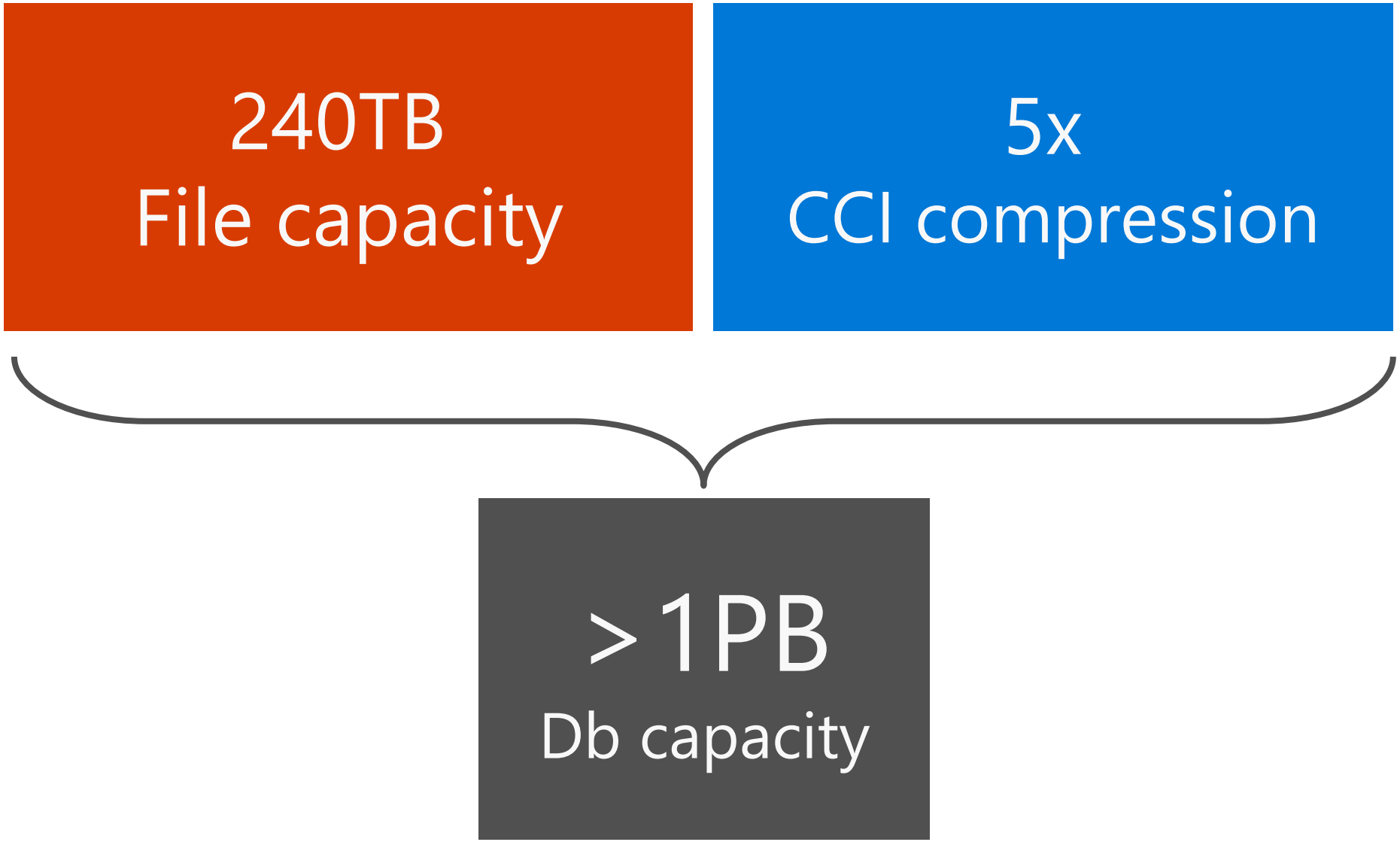
Storage tiers / Data locations



Local Storage: Tempdb sizing



Premium Storage : DB FILE Capacity Limits



Storage : Snapshots & backups

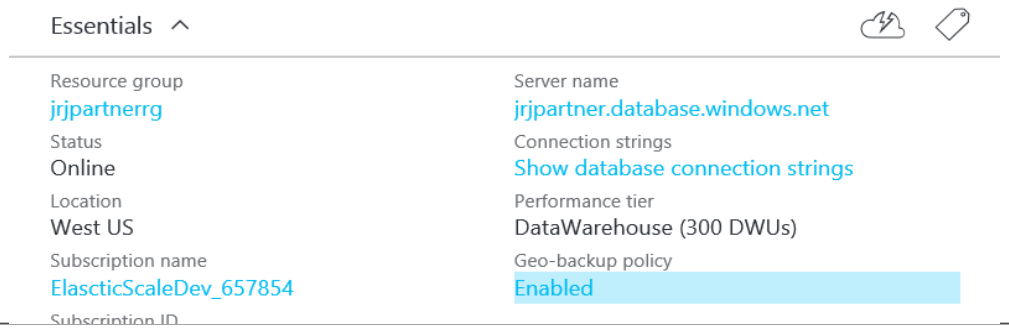
Premium Storage: Snapshots

Frequency	Retention
8 hours	7 days

RPO : 8 Hours

Blob Storage: Geo-redundant backups

Frequency and Retention	
1 Geo-backup	24hr RPO



Loading

Sizing for the data load

Delimited text 가이드

- 파일의 SIZE 가 클 경우, 데이터를 나누어 여러 개의 파일로 분할하여 사용합니다.
- 하나의 파일에 하나의 Reader를 매칭합니다.
- Delimited text는 가장 빠른 속도를 제공합니다.

압축된 텍스트 파일은 병렬 액세스에 제한이 발생할 수 있습니다.

파일의 데이터를 분할하거나 다른 파일 포맷을 사용합니다.

DWU	Readers	Writers
DW100	8	60
DW200	16	60
DW300	24	60
DW400	32	60
DW500	40	60
DW600	48	60
DW1000	60	60

Loading Option

PolyBase	BCP	SSIS	ADF
• T-SQL 을 사용한 적재방식 • 적재 속도가 빠르고 가장 선호되는 적재 옵션 • CTAS 를 사용한 초기 데이터 적재 • 증분 데이터는 INSERT/INTO 를 사용함. OR CTAS를 사용하여 별도 테이블을 만들고 나중에 최종 테이블과 파티션 SWITCH 사용	• **B**ulk **C**opy **P**rogram UTILITY • 명령어를 사용한 데이터 적재 방식 • 사이즈가 작은 파일에 사용 (**< 10 GB**) • DWU 수준을 올린다고 적재 성능이 높아지지 않음 • Thread 를 병렬로 지정하여 퍼포먼스를 향상시킬 수 있음	• MS 에서 제공하는 데이터 이관, 통합 기능이 있는 Tool 을 이용한 데이터 적재 방식 • 퍼포먼스 향상을 위해 병렬 Threads 를 증가시켜야 함	• 파이프라인을 이용한 데이터 적재 방식 (Polybase 방식) • T-SQL 스크립트를 작성하지 않고 단순히 화면에서 설정을 세팅하므로 작업 방식이 매우 단순

Polybase Sample

```
CREATE MASTER KEY;

CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY = 'user',
    SECRET =
'dIzoQupXULTk/Dy09gXWbCfP2hD6TJ8H9Lf0ypOT+ubQ3uy//PB9tNxbt04WN47dT0strIHBw83zAkB1bm+Xiw==‘
;

CREATE EXTERNAL DATA SOURCE AzureStorage
WITH (
    TYPE = HADOOP,
    LOCATION = 'wasbs://datacontainer@yongstorage.blob.core.windows.net',
    CREDENTIAL = AzureStorageCredential
);
CREATE EXTERNAL FILE FORMAT TextFile
WITH (
    FORMAT_TYPE = DelimitedText,
    FORMAT_OPTIONS (FIELD_TERMINATOR = ',')
);
```

Polybase Sample

```
CREATE EXTERNAL TABLE dbo.DimDate2External (
    DateId INT NOT NULL,
    CalendarQuarter TINYINT NOT NULL,
    FiscalQuarter TINYINT NOT NULL
)
WITH (
    LOCATION='/datedimension/',
    DATA_SOURCE=AzureStorage,
    FILE_FORMAT=TextFile
);

CREATE TABLE dbo.DimDate2
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT * FROM [dbo].[DimDate2External];
```

Polybase Sample

```
CREATE EXTERNAL TABLE dbo.DimDate2External (
    DateId INT NOT NULL,
    CalendarQuarter TINYINT NOT NULL,
    FiscalQuarter TINYINT NOT NULL
)
WITH (
    LOCATION='/datedimension/',
    DATA_SOURCE=AzureStorage,
    FILE_FORMAT=TextFile
);

CREATE TABLE dbo.DimDate2
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = ROUND_ROBIN
)
AS
SELECT * FROM [dbo].[DimDate2External];
```

Azure Synapse Analytics

Synapse serverless SQL pool

Recommended usage scenarios

Quick data exploration

- Azure storage 내부 파일의 스키마와 데이터 조회가 가능합니다.
- Parquet, CSV, JSON 파일 포맷을 지원합니다.
- BI Tool을 위해 Azure storage에 액세스 할 수 있는 Direct connector를 지원합니다.

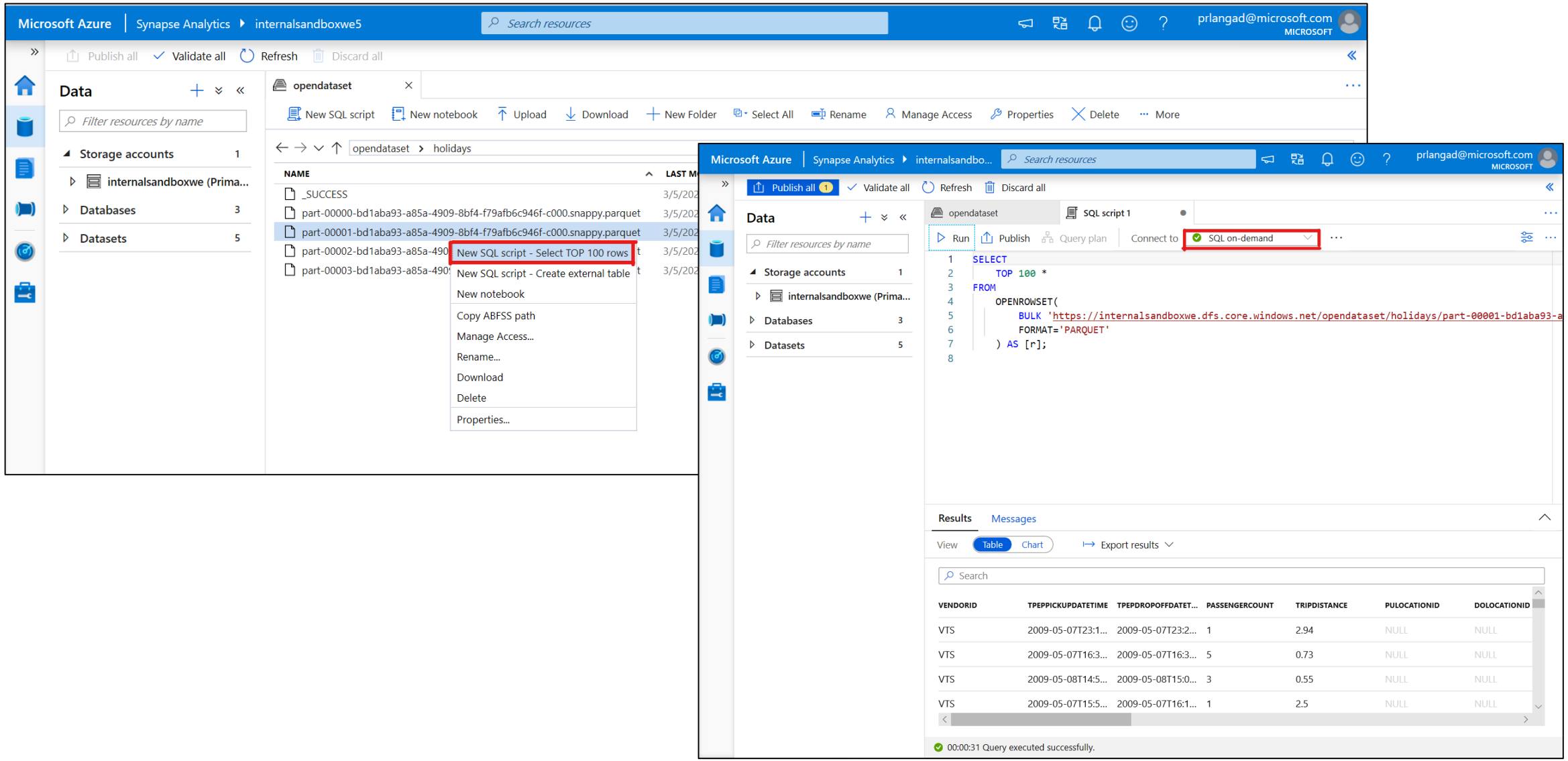
Logical Data Warehouse

- Azure storage 내부 raw files을 virtual tables 와 views 로 생성이 가능합니다.
- SQL을 사용하는 Tool을 이용하여 file 분석이 가능합니다.
- enterprise-grade security model

Easy data transformation

- CSV을 parquet 포맷으로 변경을 지원합니다.
- 컨테이너와 Storage 계정들을 오가며 데이터 이동이 가능합니다.
- 외부 스토리지에 대한 쿼리 결과를 저장 가능합니다.

Easily explore files on storage



Easily query files in various formats

Overview

다양한 파일 포맷에 저장된 데이터에 접속하기 위해 OPENROWSET 함수를 사용합니다

Benefits

- CSV, parquet, and JSON 파일 조회가 가능합니다.
- 모든 파일에 대해 통합된 T-SQL 사용이 가능합니다.
- 데이터의 변환과 조회 결과 데이터 분석을 위해 standard SQL을 사용합니다.

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

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

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

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Automatic schema inference

Overview

OPENROWSET은 외부 파일의 컬럼과 컬럼의 데이터 타입을 자동으로 정의합니다.

Benefits

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

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

Perfect solution for data exploration where schema is unknown.

The functionality is available for both parquet & CSV files.

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

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

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

Defined the query result schema inline

Overview

사용자가 쿼리의 WITH 절에 컬럼명과 컬럼의 데이터 타입을 명시할 수 있습니다.

Benefits

Define result schema at query time in WITH clause.

No need for external format files.

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

Improve performance by column elimination in parquet files.

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

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Customize the content parsing to fit your case

Overview

OPENROWSET 함수를 이용하여 사용자가 파일의 구분자를 명시할 수 있고, 원하는 컬럼만 선택적으로 명시하여 결과 값을 불러올 수 있습니다.

Benefits

Ability to read CSV files with custom format

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

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

Second, fourth, seventh and ninth columns are returned

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Easily query multiple files, with wildcards

Overview

OPENROWSET 함수에서 WildCards를 사용하여 여러 폴더와 여러 파일을 동시에 질의하여 결과를 얻을 수 있습니다.

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

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

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

Query partitioned data, using the folder structure

Overview

OPENROWSET 함수를 사용하여 여러 개의 파일로부터 원하는 데이터만을 조회할 수 있습니다.

Benefits

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

Eliminate sub-folders/partitions before the query starts execution

Query Spark/Hive partitioned data sets

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

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

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

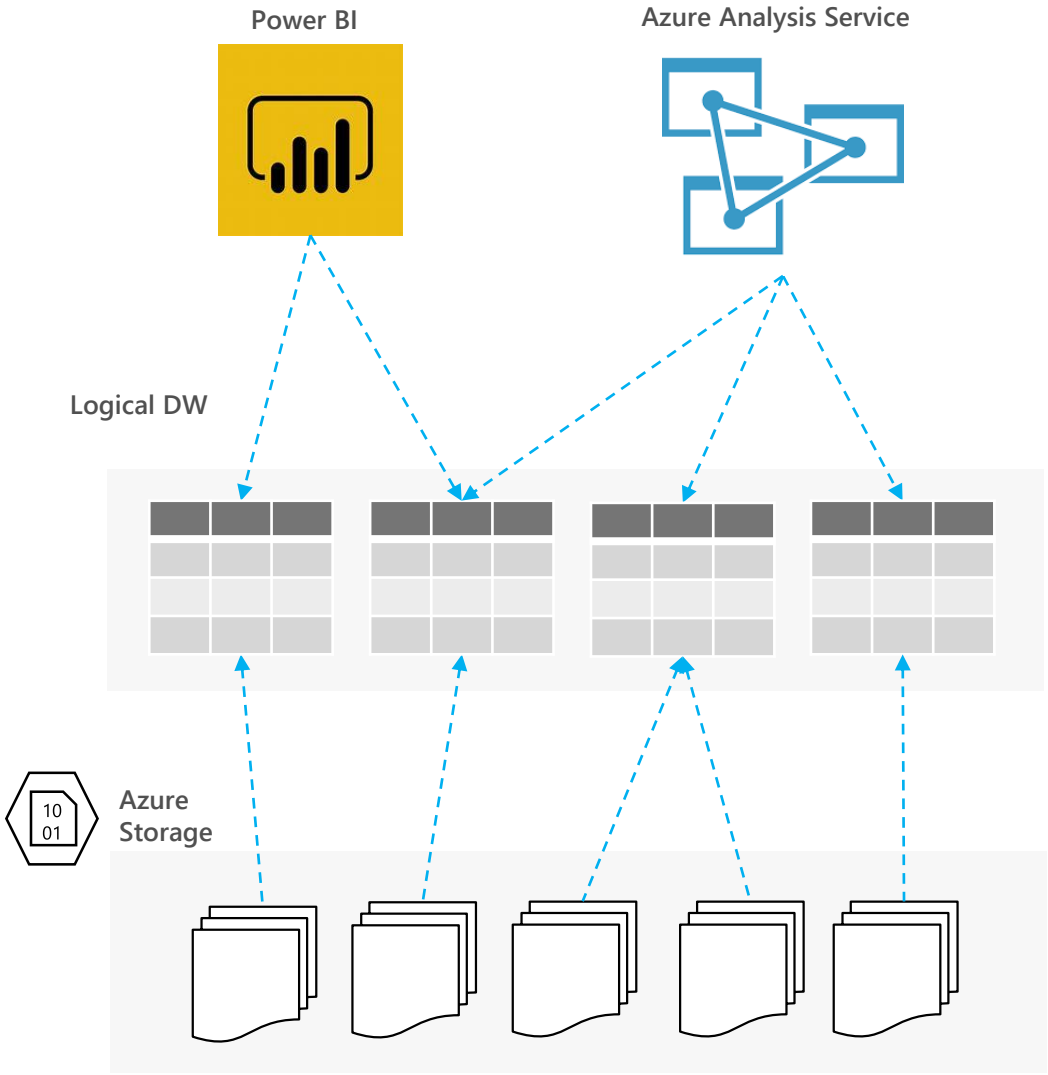
Synapse serverless SQL pool as a logical data warehouse

Overview

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

Benefits

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



Logical Data Warehouse views

Overview

Serverless SQL pool은 논리적 Data Warehouse로서 Azure Storage에 있는 파일들을 사용하여 View 를 생성할 수 있습니다.

Benefits

Create SQL views on externally stored data

Access files using the view from various tools and language

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

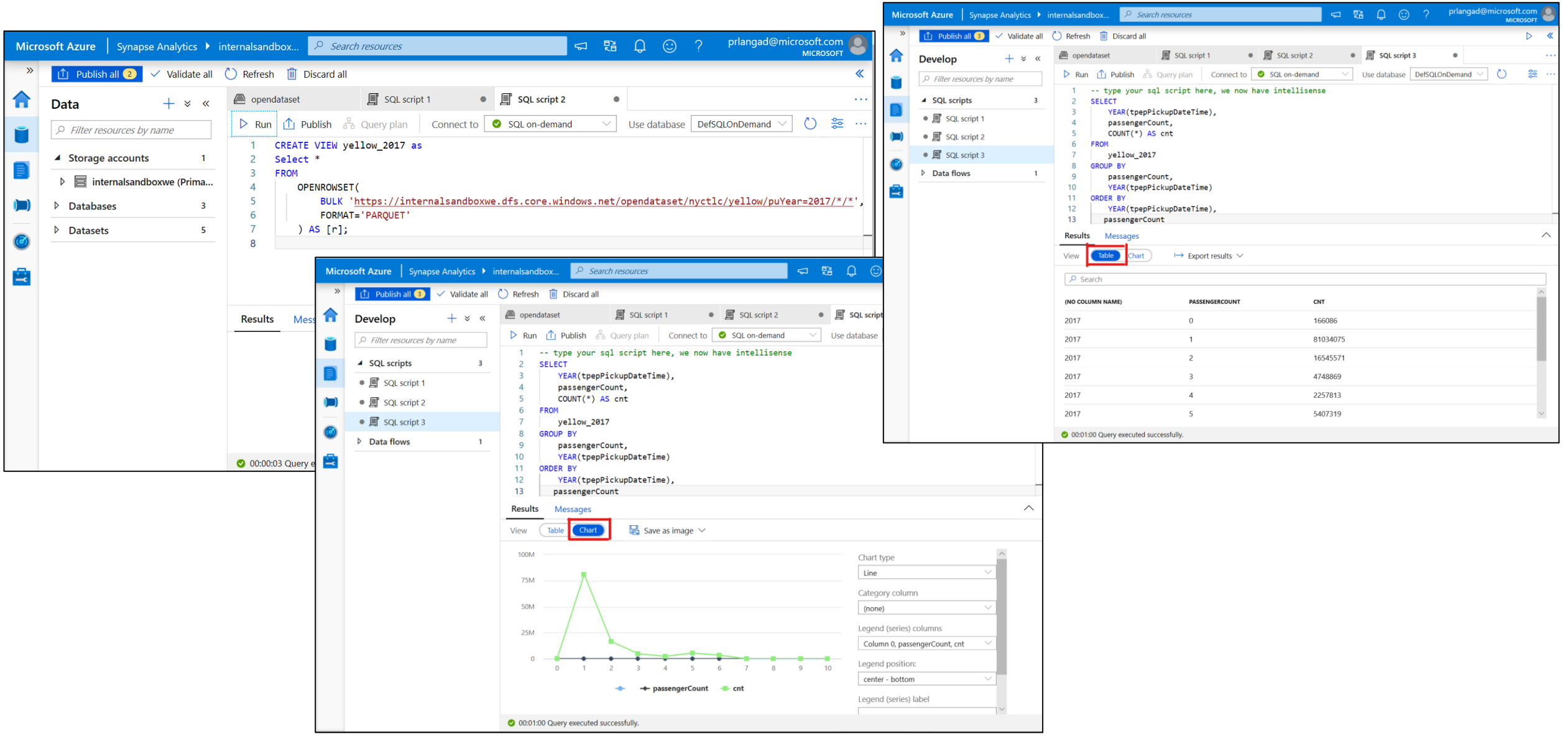
Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP VIEW IF EXISTS  populationView
GO

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

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

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

Creating views



Logical Data Warehouse - tables

Overview

Serverless SQL Pool은 논리적 Data Warehouse로서 Azure Storage에 있는 파일들을 사용하여 External Files을 생성할 수 있습니다.

Benefits

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

Join and transform multiple tables in the same query.

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

Manage column statistics in external tables.

Manage access rights per table.

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

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

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

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

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

Easy data transformation

Overview

SQL queries을 사용하여 Azure Storage Files의 데이터 포맷을 쉽게 변환할 수 있습니다.

Optimize data pipeline - achieve more using serverless SQL pool

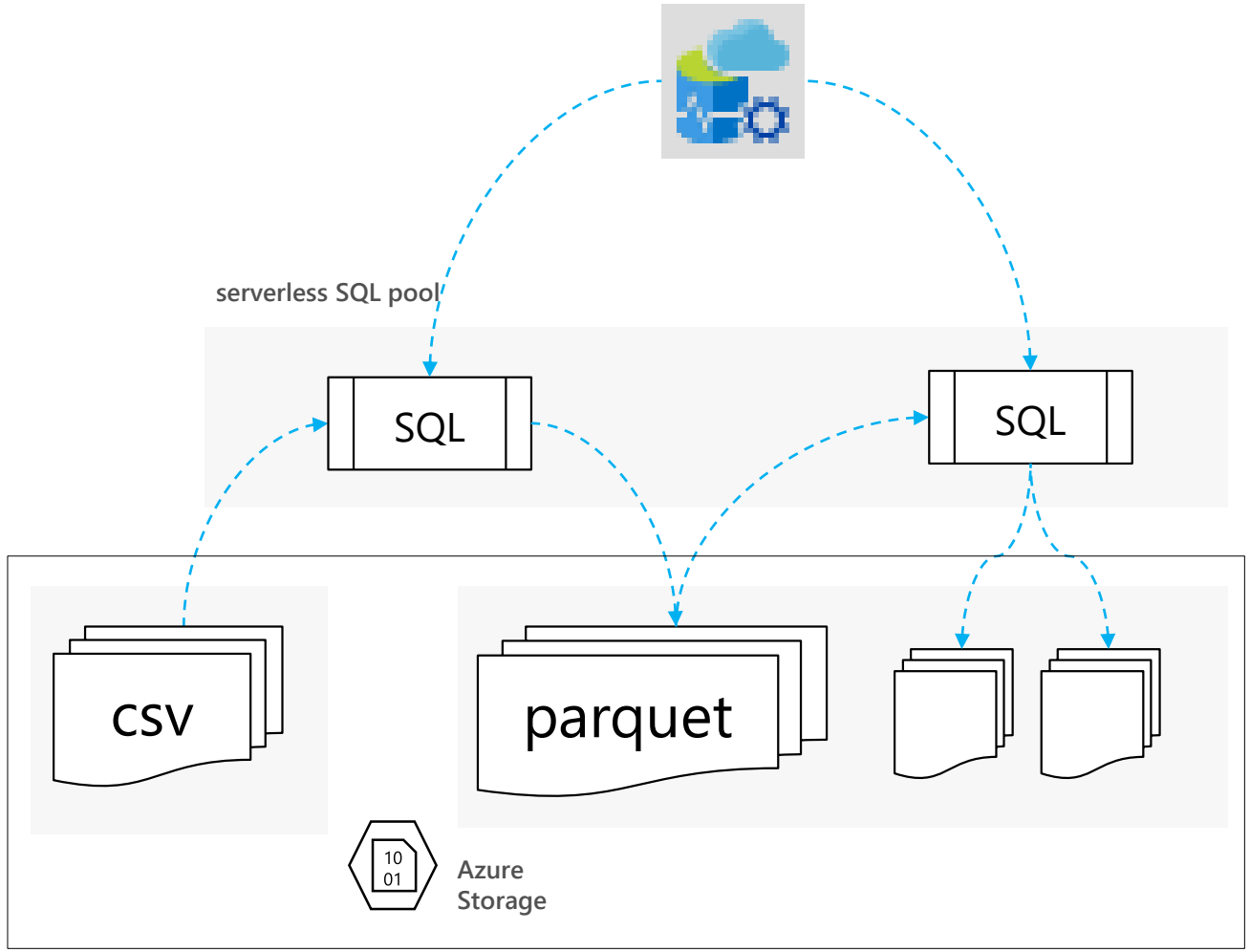
Benefits

Single statement transformations:

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

SQL ETL pipelines

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



Easy data transformation with CETAS

Overview

Create External Tables As Select (CETAS)를 이용하여 쉬운 데이터 포맷 변환이 가능하며 Azure storage에 쿼리 결과를 저장할 수 있습니다.

Benefits

Select any data set and store it in parquet format.

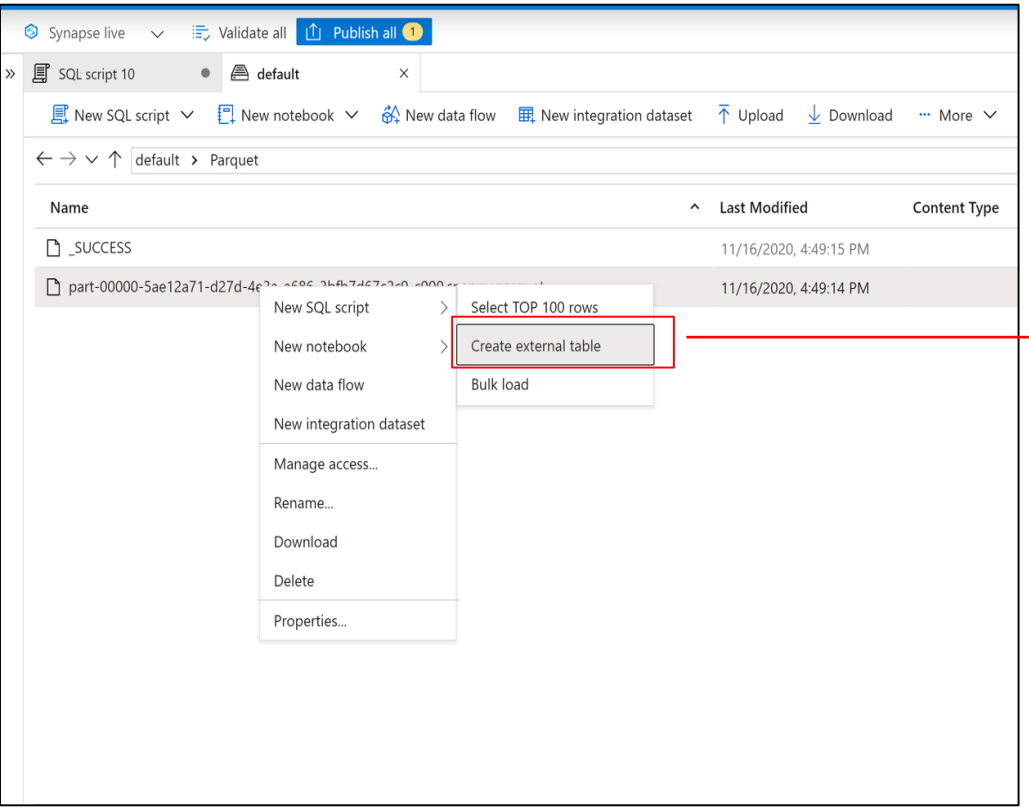
Pre-calculate and store results of query and store them permanently on Azure storage.

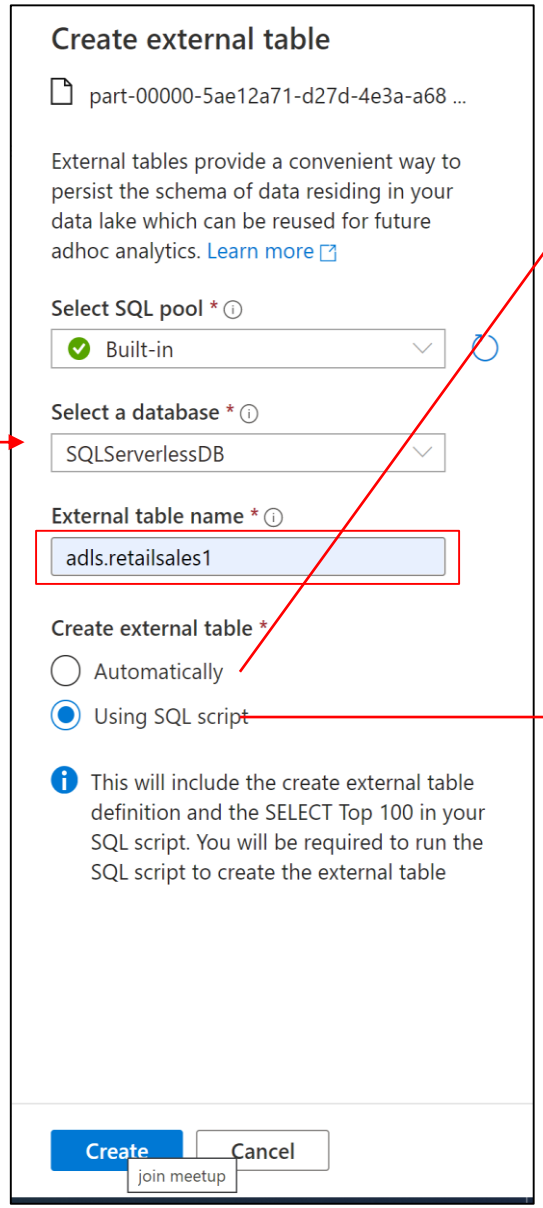
Use saved data using external table.

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

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

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

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

UI based data transformation





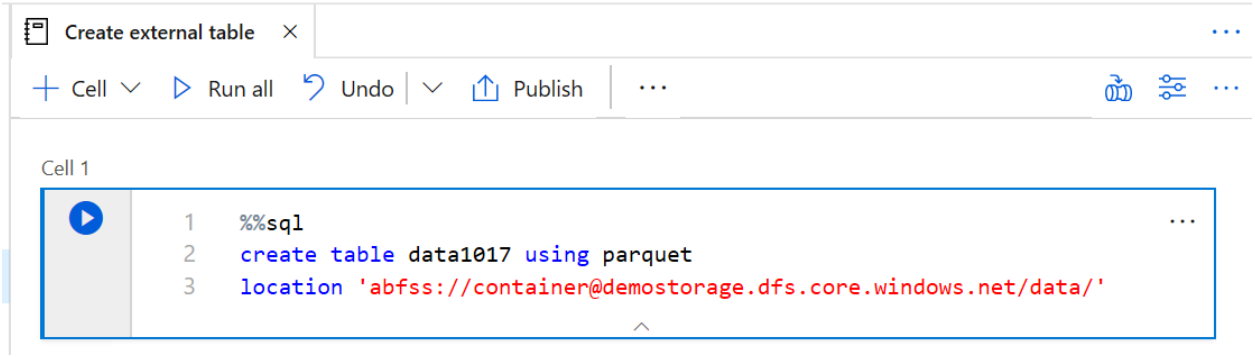

```
SELECT TOP 100 * FROM adls.retailsale
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'default_azuresynapsesa_dfs_core_windows_net')
    CREATE EXTERNAL DATA SOURCE [default_azuresynapsesa_dfs_core_windows_net]
    WITH (
        LOCATION   = 'https://azuresynapsesa.dfs.core.windows.net/default',
    )
Go

CREATE EXTERNAL TABLE adls.retailsale (
    [storeId] varchar(8000),
    [productCode] varchar(8000),
    [quantity] varchar(8000),
    [logQuantity] varchar(8000),
    [advertising] varchar(8000),
    [price] varchar(8000),
    [weekStarting] varchar(8000),
    [id] varchar(8000)
    )
    WITH (
    LOCATION = 'Parquet/part-00000-5ae12a71-d27d-4e3a-a686-3bfb7d67c2c9-c000.snappy.parquet',
    DATA_SOURCE = [default_azuresynapsesa_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
GO

SELECT TOP 100 * FROM adls.retailsale
```

Automatic syncing of Spark tables

Overview

Spark Pool에서 테이블을 생성하면 Serverless SQL Pool에서 해당 테이블을 참조할 수 있는 External Table이 자동 생성된다.



Benefits

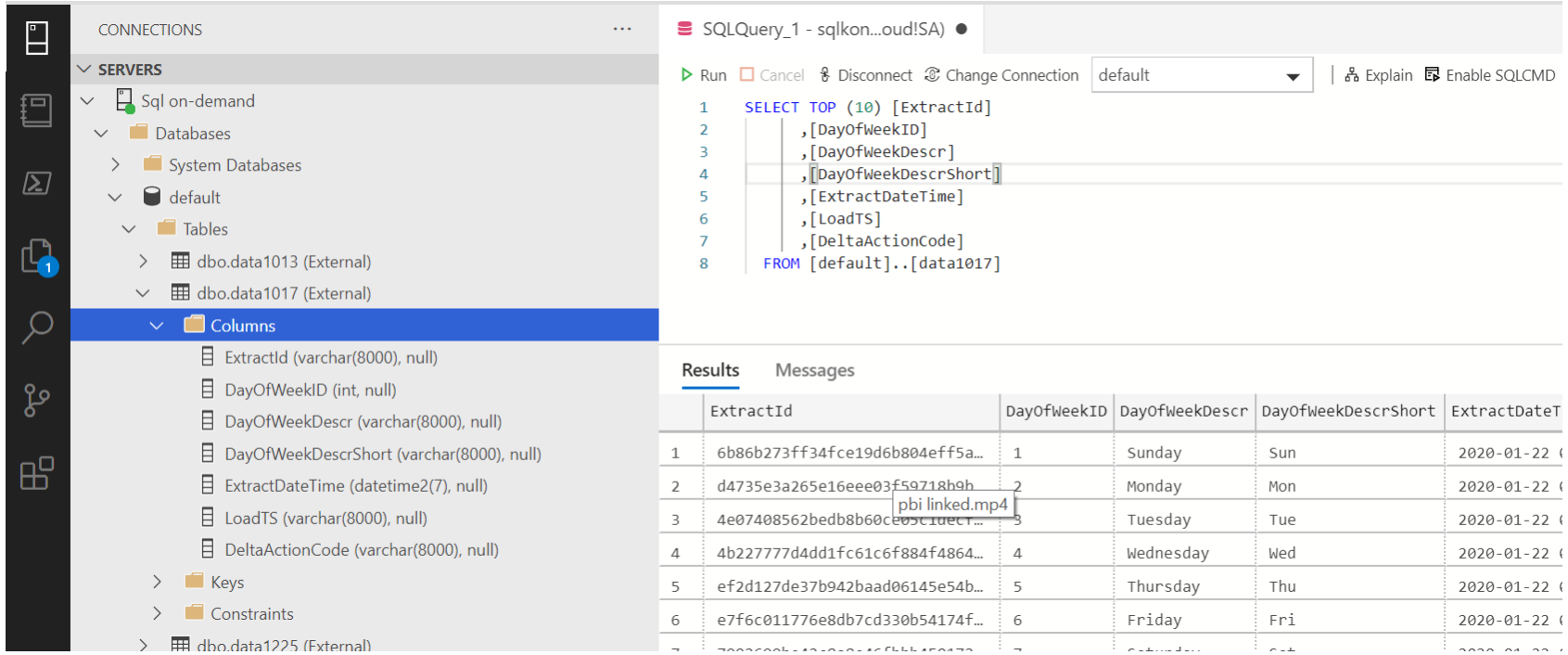
Tables designed using Spark languages are immediately available in serverless SQL pool.

Schema definition matches original

Spark table updates are applied in serverless SQL pool

No need to manually create SQL tables that match Spark tables

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



Azure Synapse Analytics

Apache Spark

Azure Synapse Apache Spark - Summary

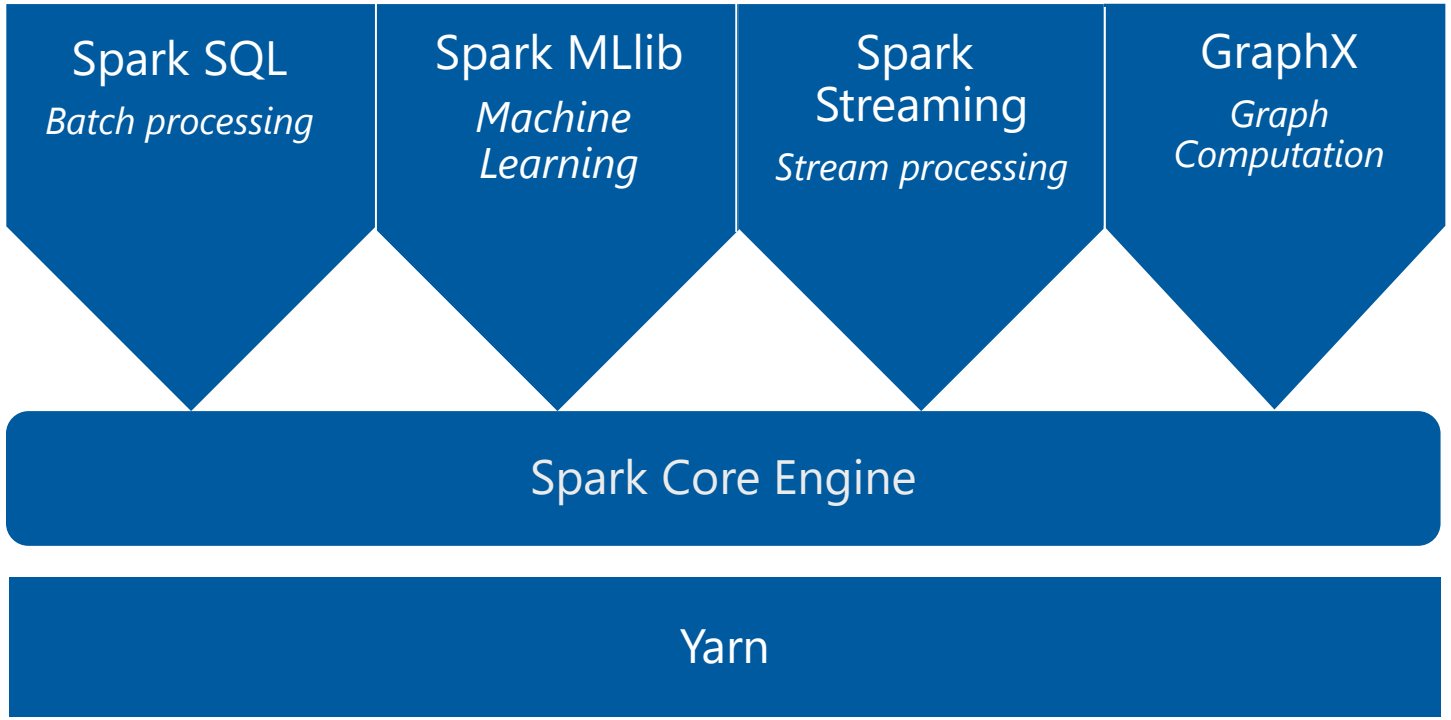
- **Apache Spark 2.4 derivation**
  - Linux Foundation Delta Lake 0.6 지원
  - .Net Core 3.0 지원
  - Python 3.6 + Anacondas 지원
- 다른 **Azure Synapse services** 와의 연결성
  - 통합된 보안과 Sign on
  - 통합된 Metadata
  - 통합 및 간소화된 provisioning
  - nteract 기반 노트북이 포함된 통합된 UX
  - Synapse SQL (provisioned) pools에 대한 Fast load 제공
- **Core scenarios**
  - Data Prep/Data Engineering/ETL
  - Spark ML과 Azure ML 통합을 통한 Machine Learning
  - Library management를 통한 확장성
- 효율적인 자원 활용
  - 빠른 시작
  - Auto scale (up and down)
  - Auto pause
  - 최소 3개 노드 필요
- 다양한 언어 지원
  - .Net (C#), PySpark, Scala, Spark SQL, Java

Apache Spark

빅데이터 분석을 위한 **unified open source**인, 병렬 데이터 처리 프레임워크

Spark Unifies:

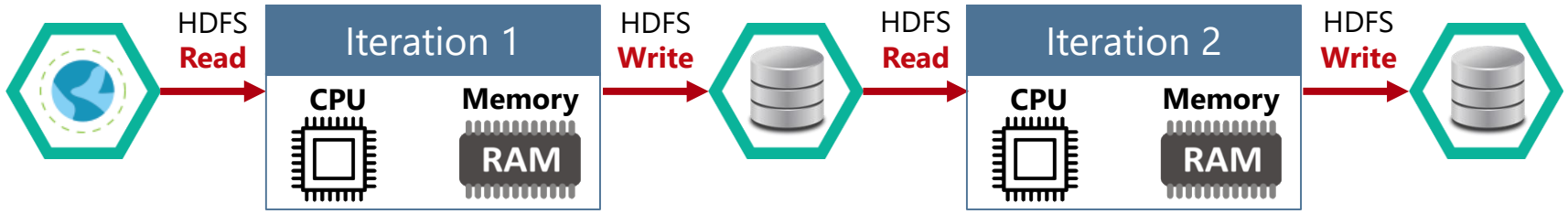
- 배치 처리
- SQL 상호작용
- 실시간 처리
- Machine Learning
- Deep Learning
- Graph 처리



http://spark.apache.org

Motivation for Apache Spark

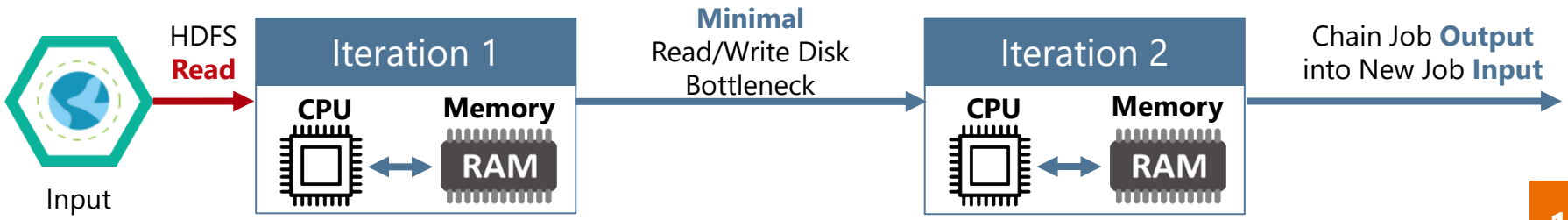
기존 방법: 복잡하거나, 온라인 event-hub 처리 등과 같은 MapReduce job에는 많은 disk I/O가 발생됨.



Motivation for Apache Spark

기존 문제점: 복잡하거나, 온라인 event-hub 처리 등과 같은 MapReduce job에는 많은 disk I/O가 발생됨.

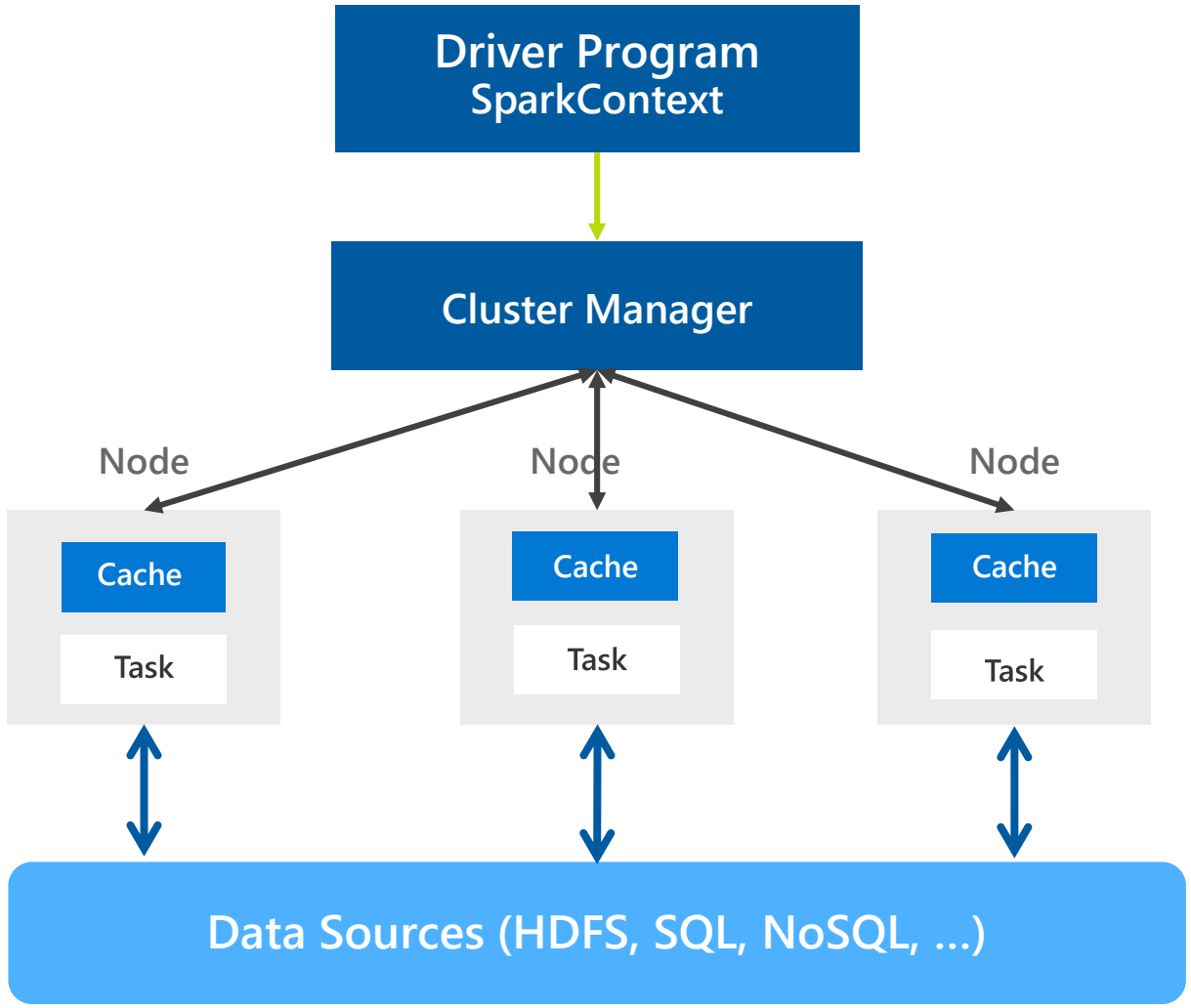
해결 방법: 새로운 분산 처리 엔진을 이용하여, data를 in-memory 형식으로 사용.



10–100x faster than network & disk

General Spark Cluster Architecture

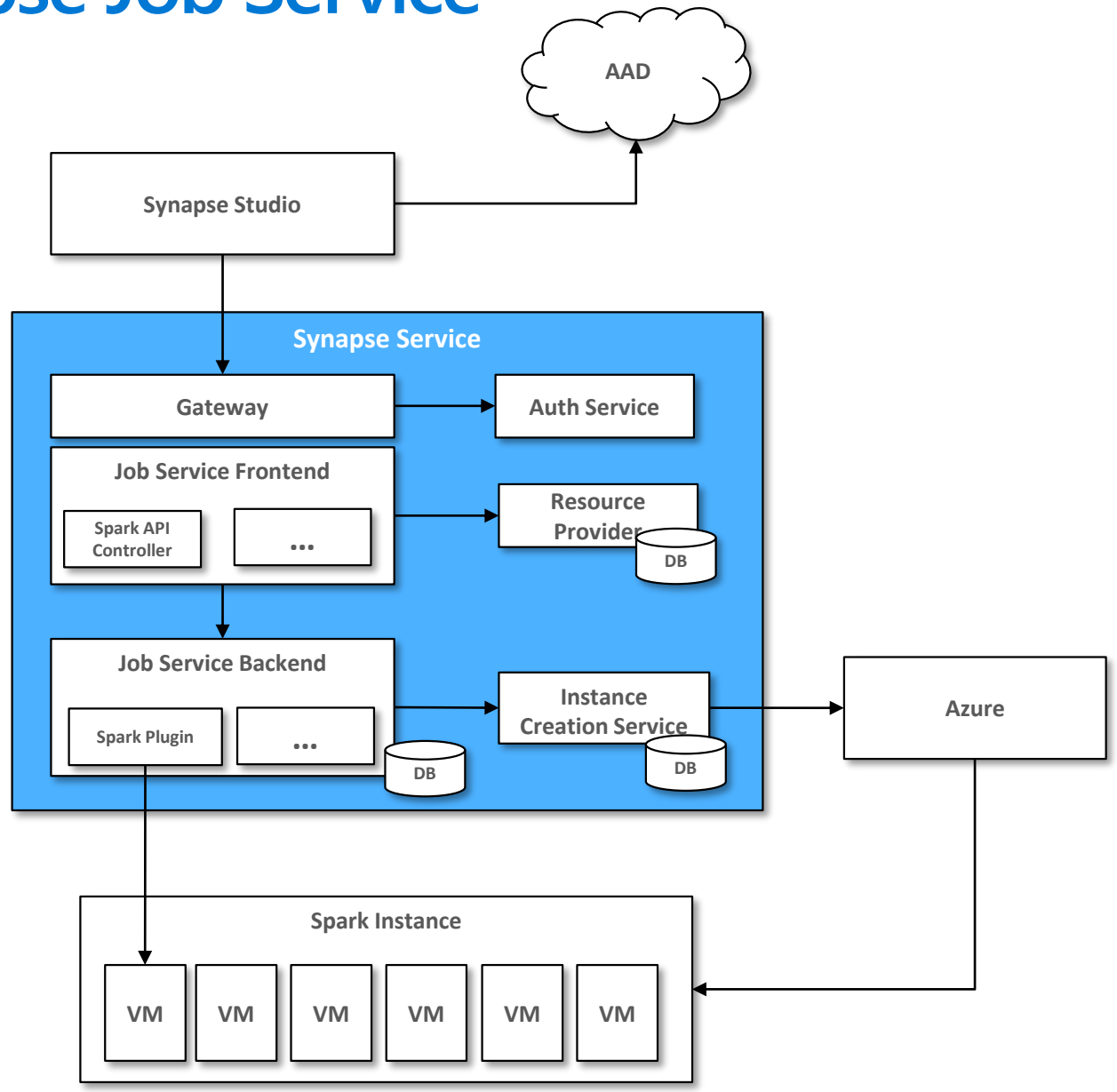
- 'Driver'는 사용자의 main() 함수를 구동하고, 다양한 병렬 작업들을 worker node에서 구동하게 합니다.
- 작업들의 결과는 'Driver'에 의해 취합되어 집니다.
- Worker node들은 HDFS, SQL과 같은 Data Source들로부터 데이터를 읽고 쓰게 됩니다.
- Worker node는 변환된 데이터를 메모리에 RDD로 cache합니다.
- Worker node들과 Driver Node는 public cloud의 VM들과 같이 실행됩니다.



Azure Synapse Apache Spark

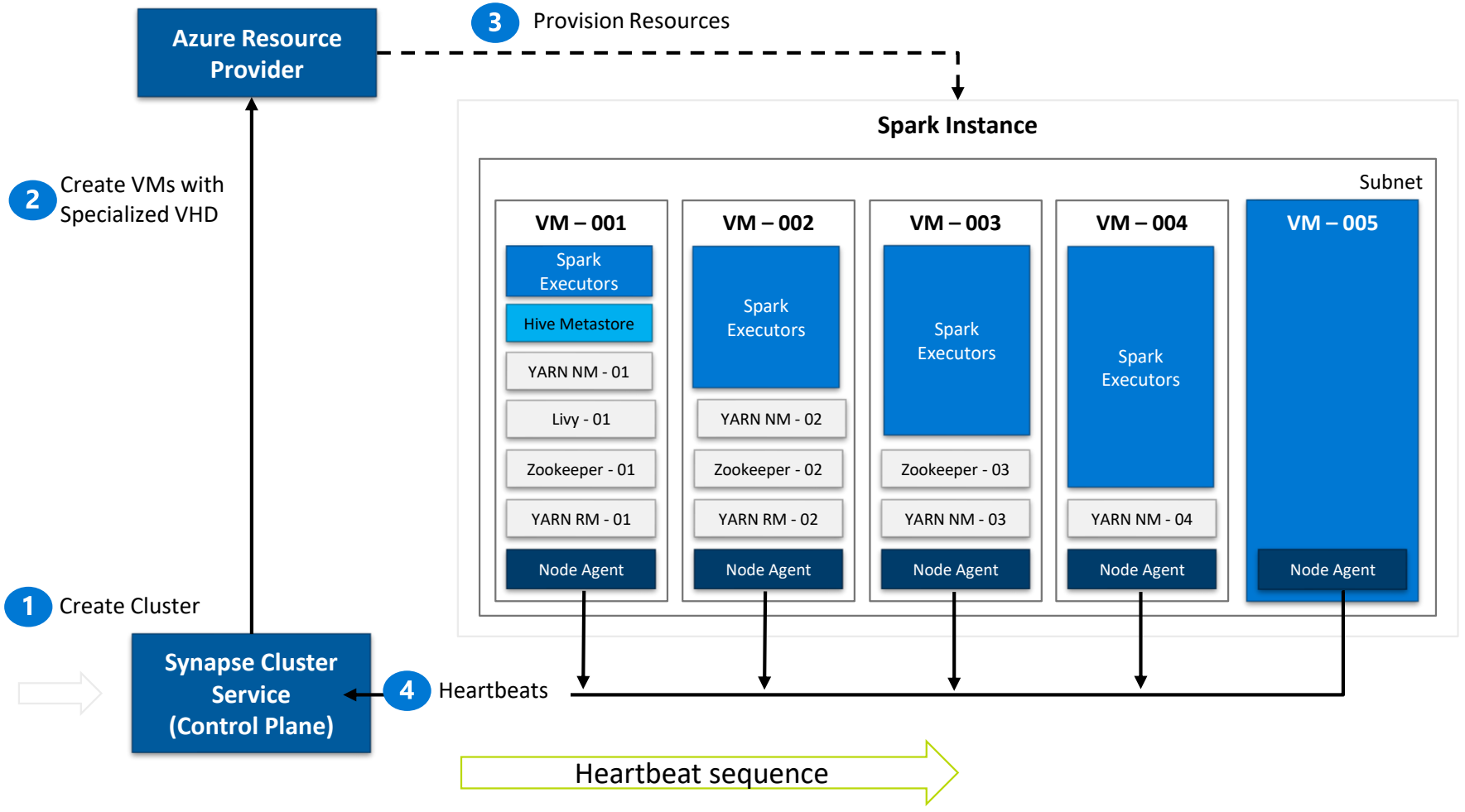
Architecture Overview

Synapse Job Service



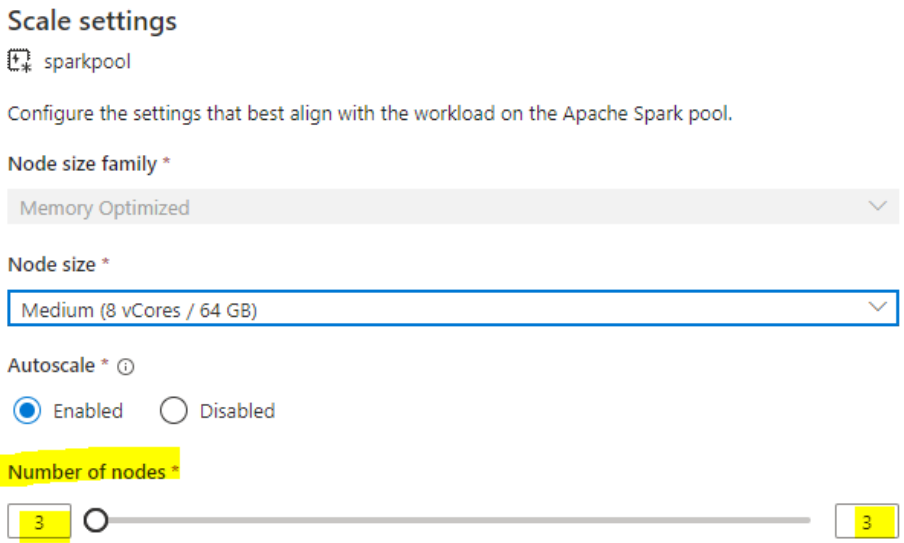
1. Synapse Workspace와 spark pool을 생성
2. Notebook을 spark pool에 연결하고 spark 코드를 입력
3. Notebook client는 AAD로 부터 사용자 토큰을 받게 되며, Synapse Gateway에 Spark session 생성 요청을 보냄
4. 이후, Synapse Gateway는 workspace와 spark pool에 대한 인증과 권한을 확인한 후, Job Service frontend에서 호스팅하는 Spark (Livy) controller 로 해당 요청을 포워딩 함
5. Job Service frontend는 Job Service backend로, 2개의 job을 생성하는 요청
하나는 cluster 생성 요청이고, 다른 하나는 spark session 생성 요청
6. Job Service backend는 Workspace와 Spark pool의 자세한 정보를 얻기 위해, Synapse Resource Provider와 연락하게 되고, Synapse Instance Service에게 cluster 생성을 위임
7. Instance가 만들어지게 되면, Job Service backend는 spark session 생성 요청을, cluster 안에 있는 Livy endpoint로 요청하게 됨
8. Spark session이 만들어지게 되면, Notebook client는 Spark statement들을 Job Service frontend로 보내게 됨
9. Job Service frontend는 backend로 부터 만들어진, 해당 cluster를 생성한 특정 사용자를 위해 만들어진 실제 Livy endpoint 를 가지고 있으며, statement 실행을 위해 직접 statement를 보냄

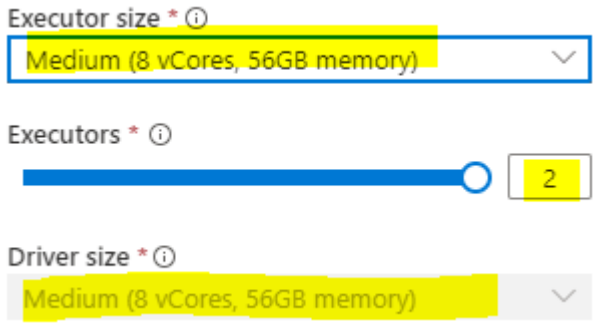
Synapse Spark Instances

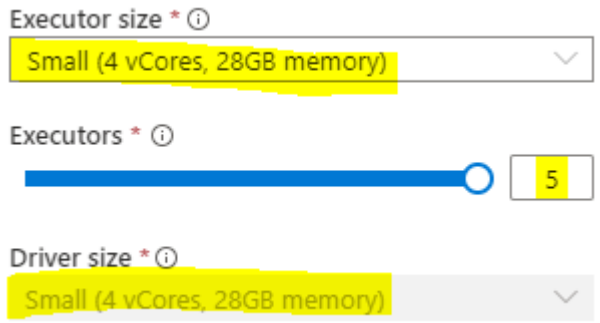


1. Synapse Job Service는 Spark pool에 설명에 따라, Cluster Service에게 cluster 생성 요청을 보냅니다.
2. Cluster Service는 Azure SDK를 사용하여 specialized VHD를 이용하여 VM을 만들도록 Azure에 요청합니다.
3. Specialized VHD는 cluster 타입(eg. Spark)에 의해 필요한 모든 서비스들이 포함되어져 있습니다.
4. VM이 부팅 되면, Node Agent는 heartbeat을 Cluster Service로 보내고 node configuration을 받습니다.
5. 첫번째 heartbeat을 기반으로 하여, node들은 초기화되고, role들이 assign 됩니다.
6. extra 노드들은 첫번째 heartbeat에서 삭제됩니다.
7. Cluster Service가 cluster가 준비되었다고 판단될 때, livy endpoint를 Job Service에게 반환합니다.

Synapse Spark Instances

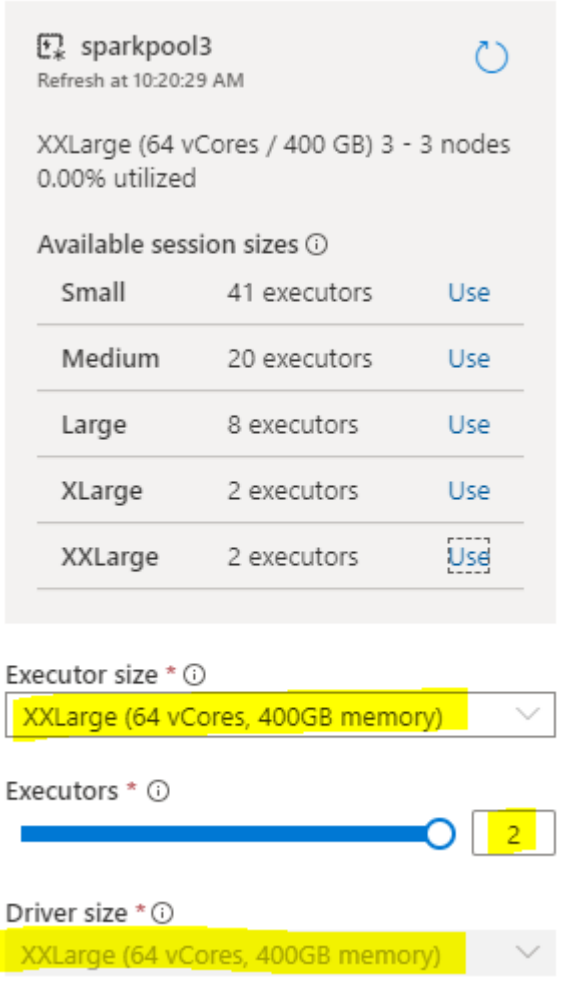


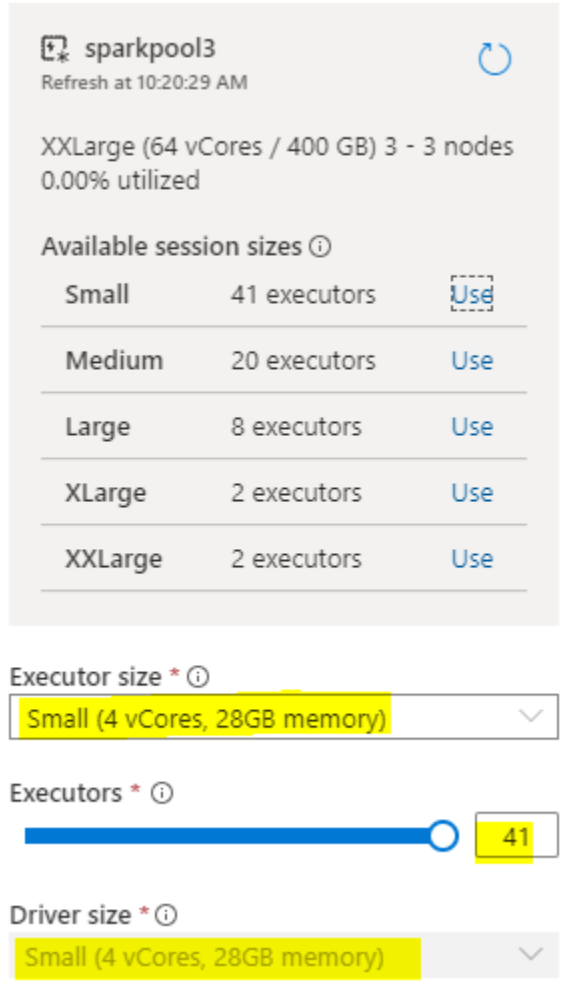




- 기존의 Spark와 같이, Executor의 크기를 지정할 수 있습니다.
- 노드의 최대 크기까지 Executor를 정할 수 있습니다.

Synapse Spark Instances





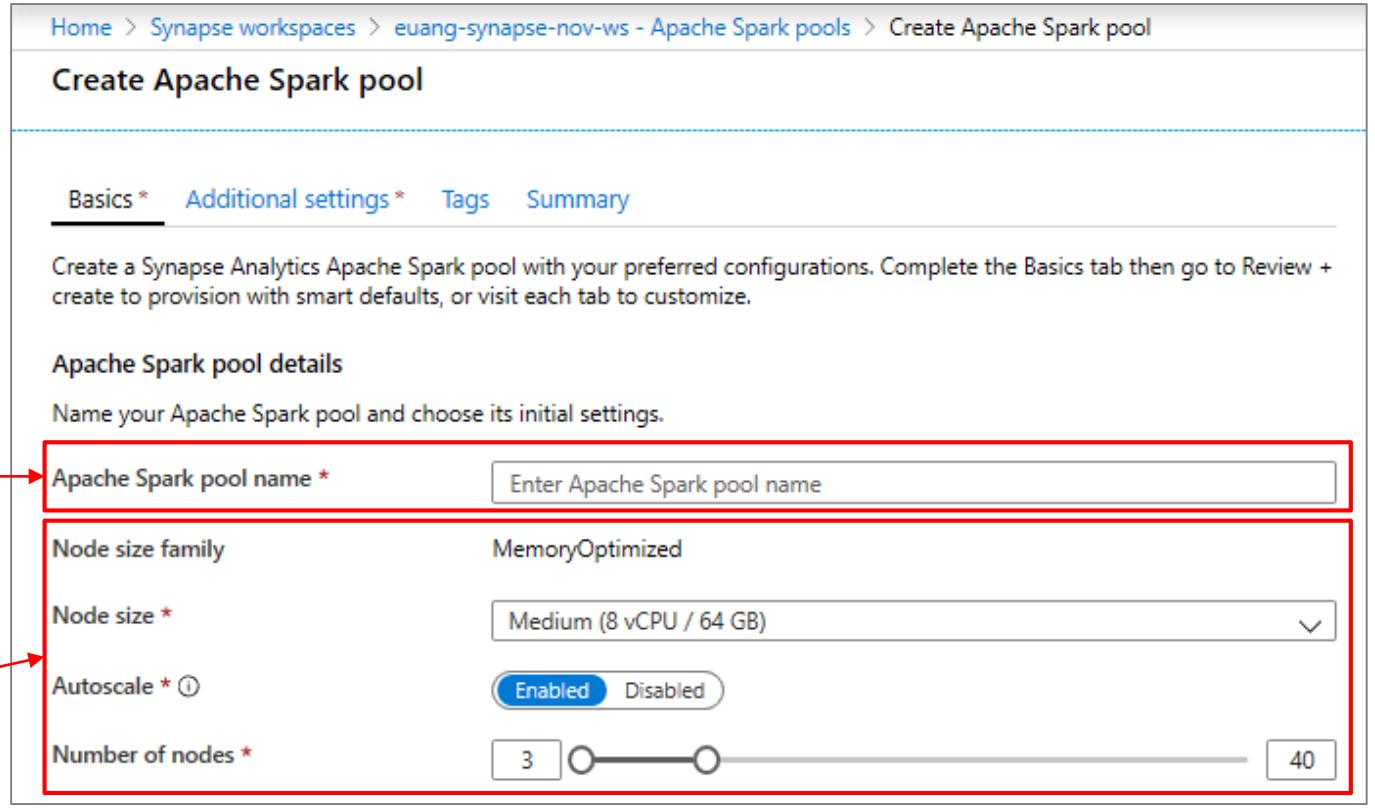
Creating a Spark pool (1 of 2)

Azure Portal을 이용한 Spark Pool 생성 할 때, 기본설정 되어진 부분을 통해 쉽게 생성이 가능합니다.

Basics 탭에서 입력해야 하는 부분은 pool 이름만 설정하면 다른 부분은 기본 설정값이 들어 있어 생성이 쉽습니다.

사용자 입력이 필수인 부분

기본 설정 값이 입력되어져 있음

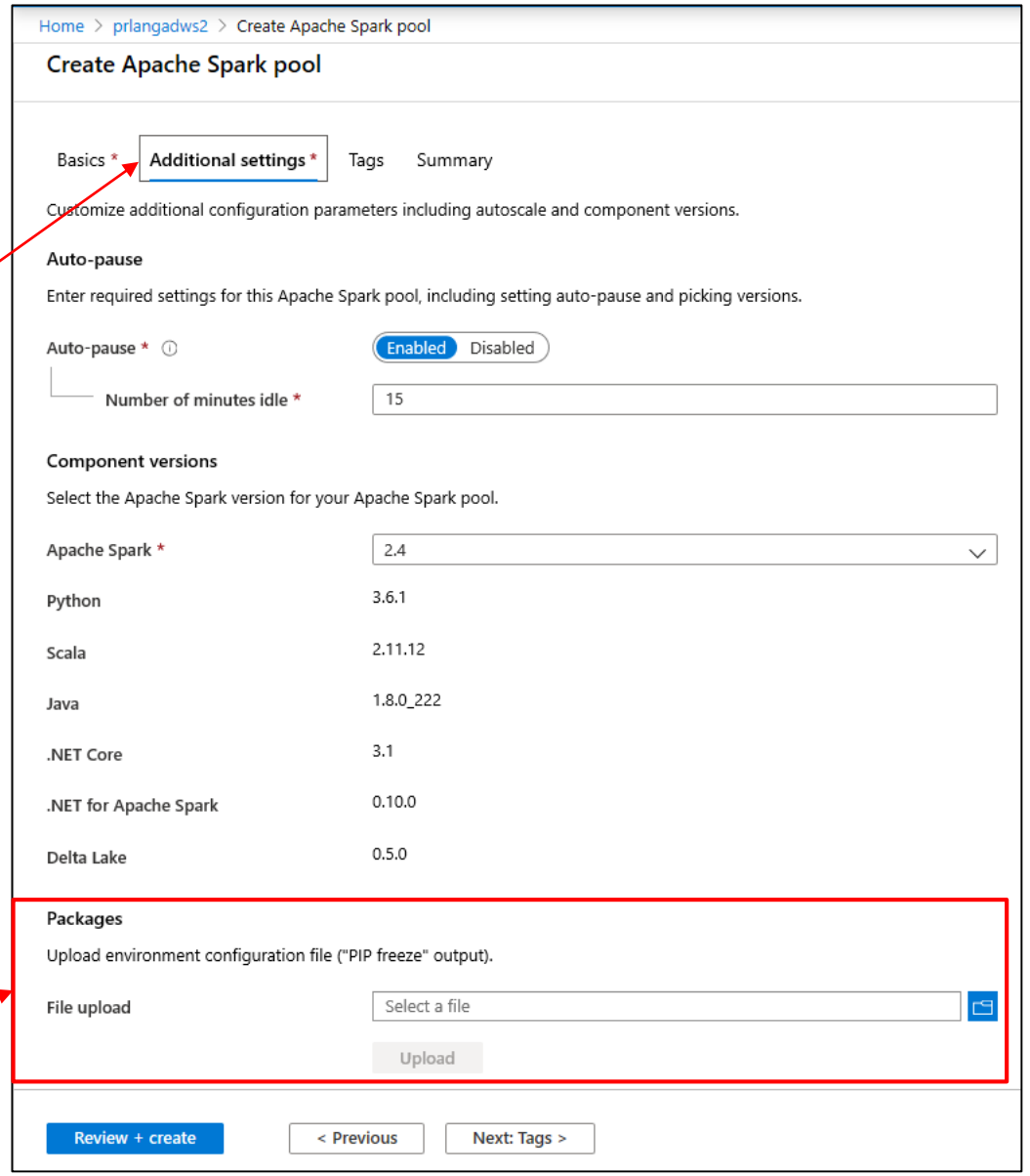


Creating a Spark pool (2 of 2) - optional

Additional settings 탭을 통해, 세부적인 설정 또한 가능합니다.

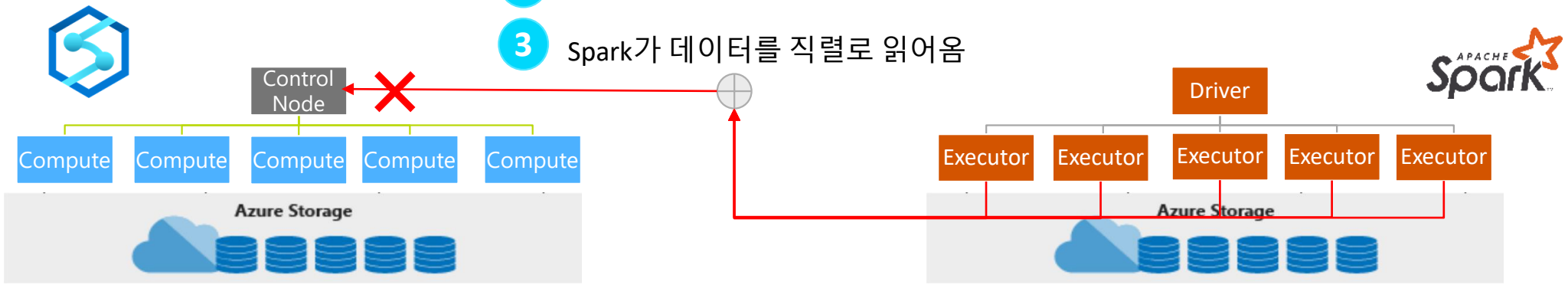
Spark의 버전, auto-pause의 시간 설정 가능

사용하는 package 이름과 버전 업로드하여, 라이브러리 import 가능



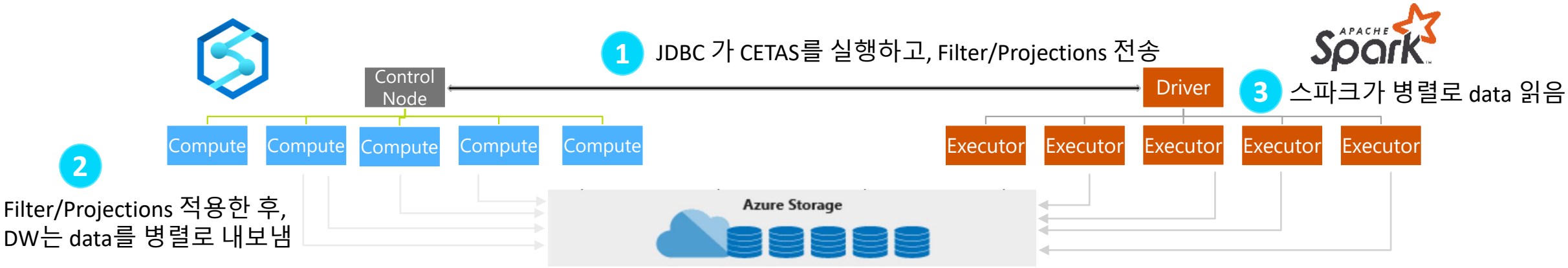
Existing Approach: JDBC

1. JDBC 를 이용하여 connection 열기
2. Filter/Projections 적용
3. Spark가 데이터를 직렬로 읽어옴



New Approach: JDBC and Polybase

1. JDBC 가 CETAS를 실행하고, Filter/Projections 전송
2. Filter/Projections 적용한 후, DW는 data를 병렬로 내보냄
3. 스파크가 병렬로 data 읽음



사용자 프로비저닝한 작업영역 - 기본 Data Lake

Code-Behind Experience

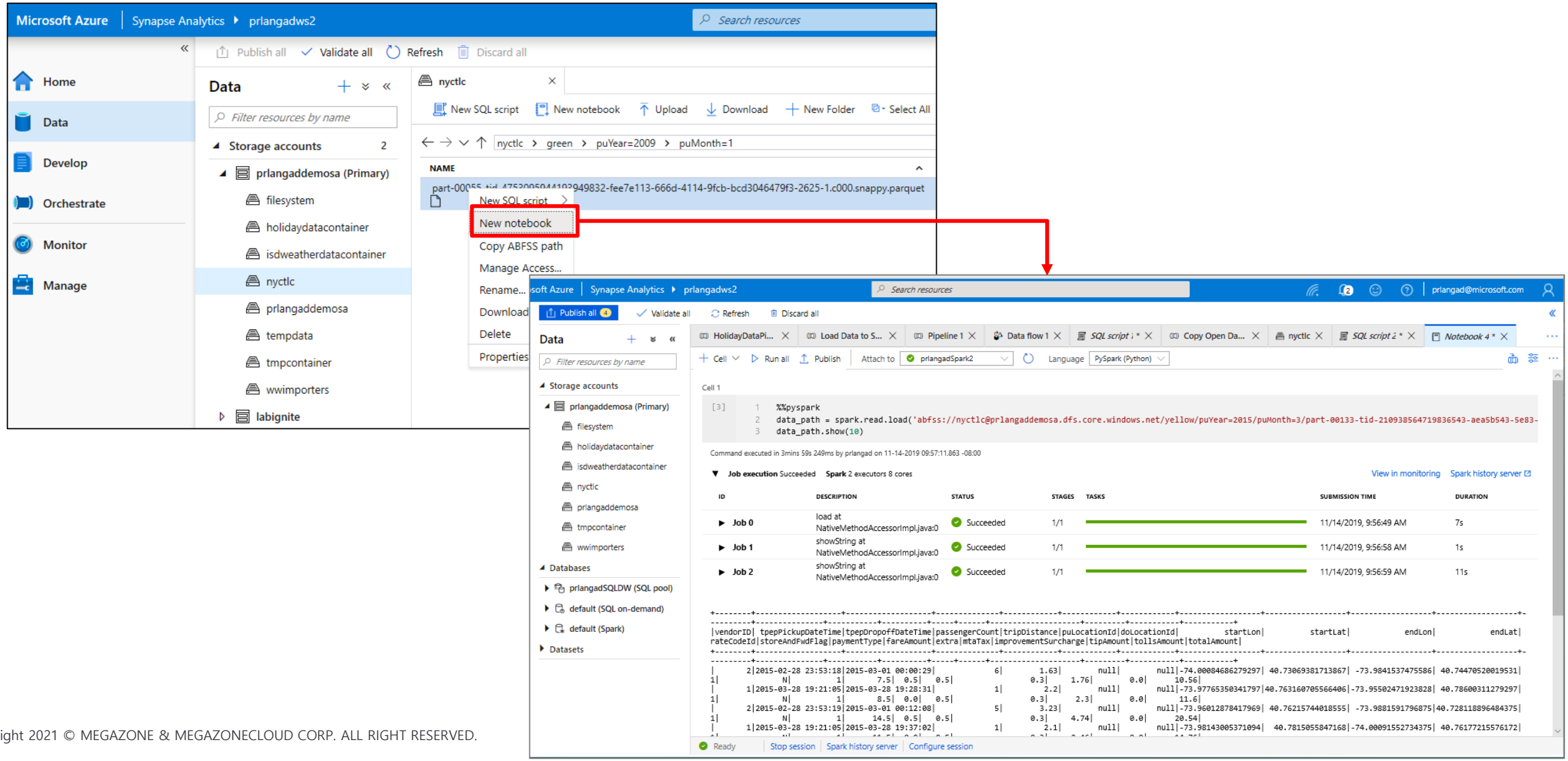
Existing Approach

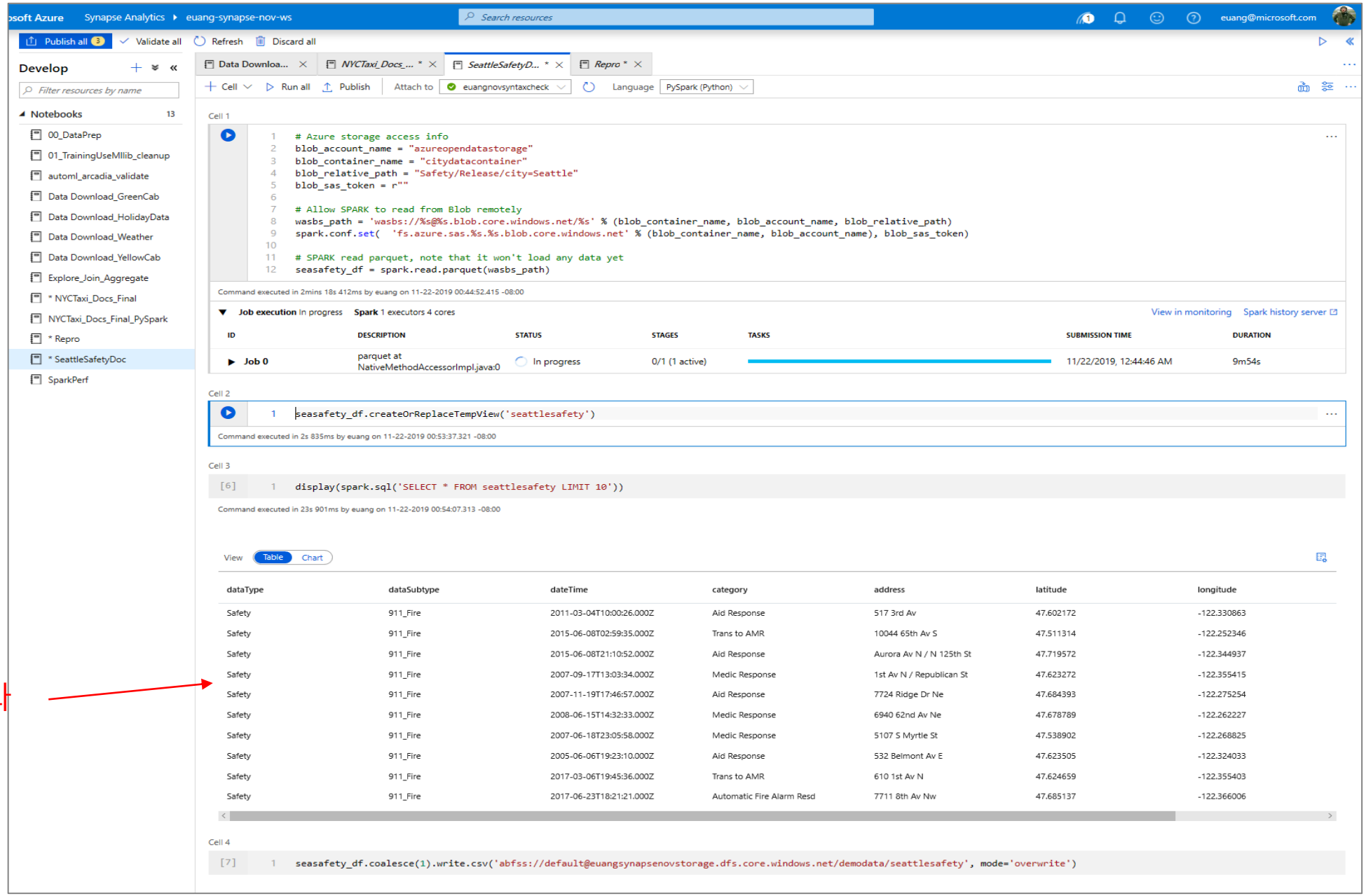
```
val jdbcUsername = "<SQL DB ADMIN USER>"
val jdbcPwd = "<SQL DB ADMIN PWD>"
val jdbcHostname = "servername.database.windows.net”
val jdbcPort = 1433
val jdbcDatabase ="<AZURE SQL DB NAME>“


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

val connectionProperties = new Properties()

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

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

New Approach

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

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

Create Notebook on files in storage





테이블 포멧으로 결과 출력

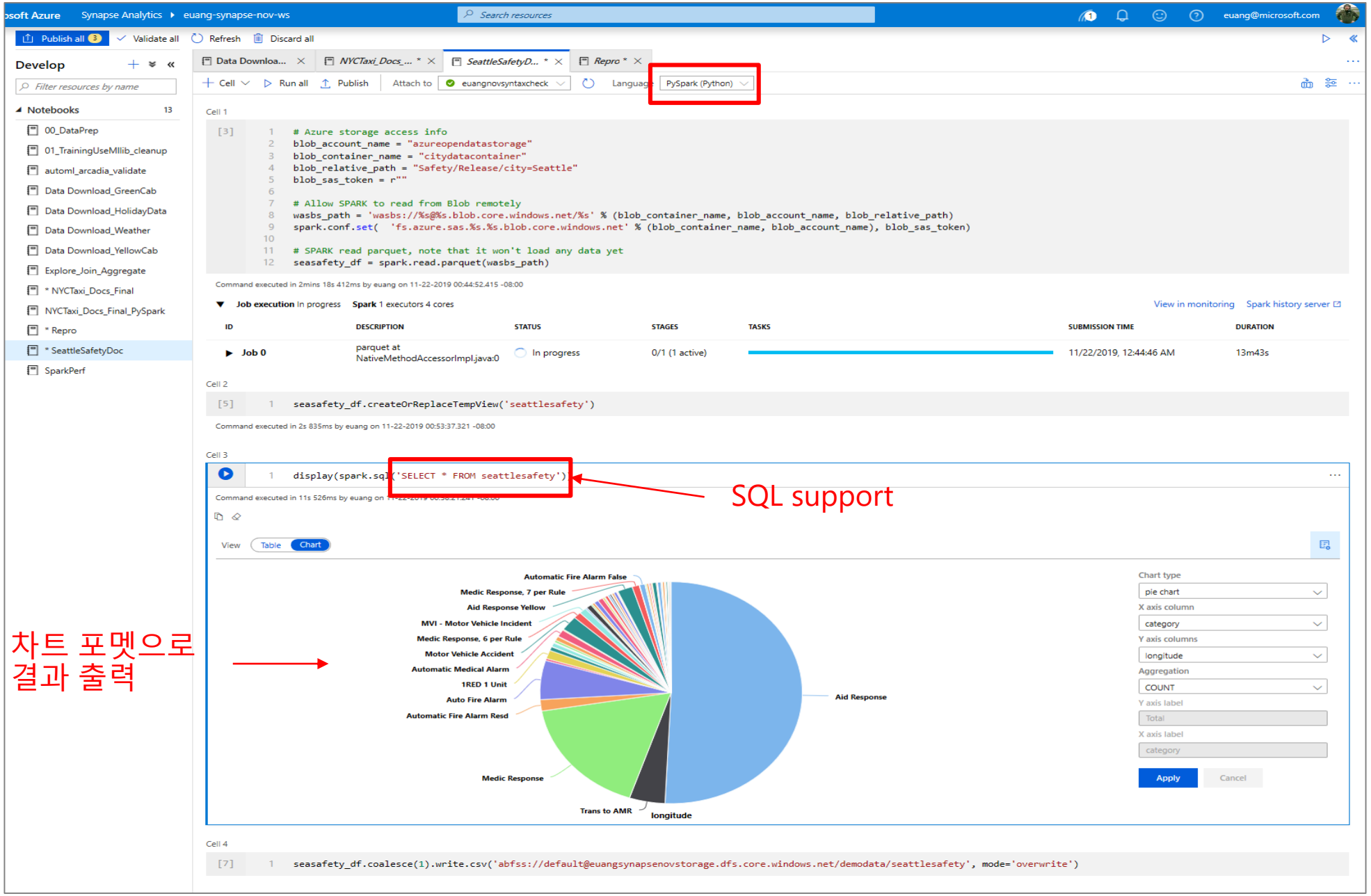
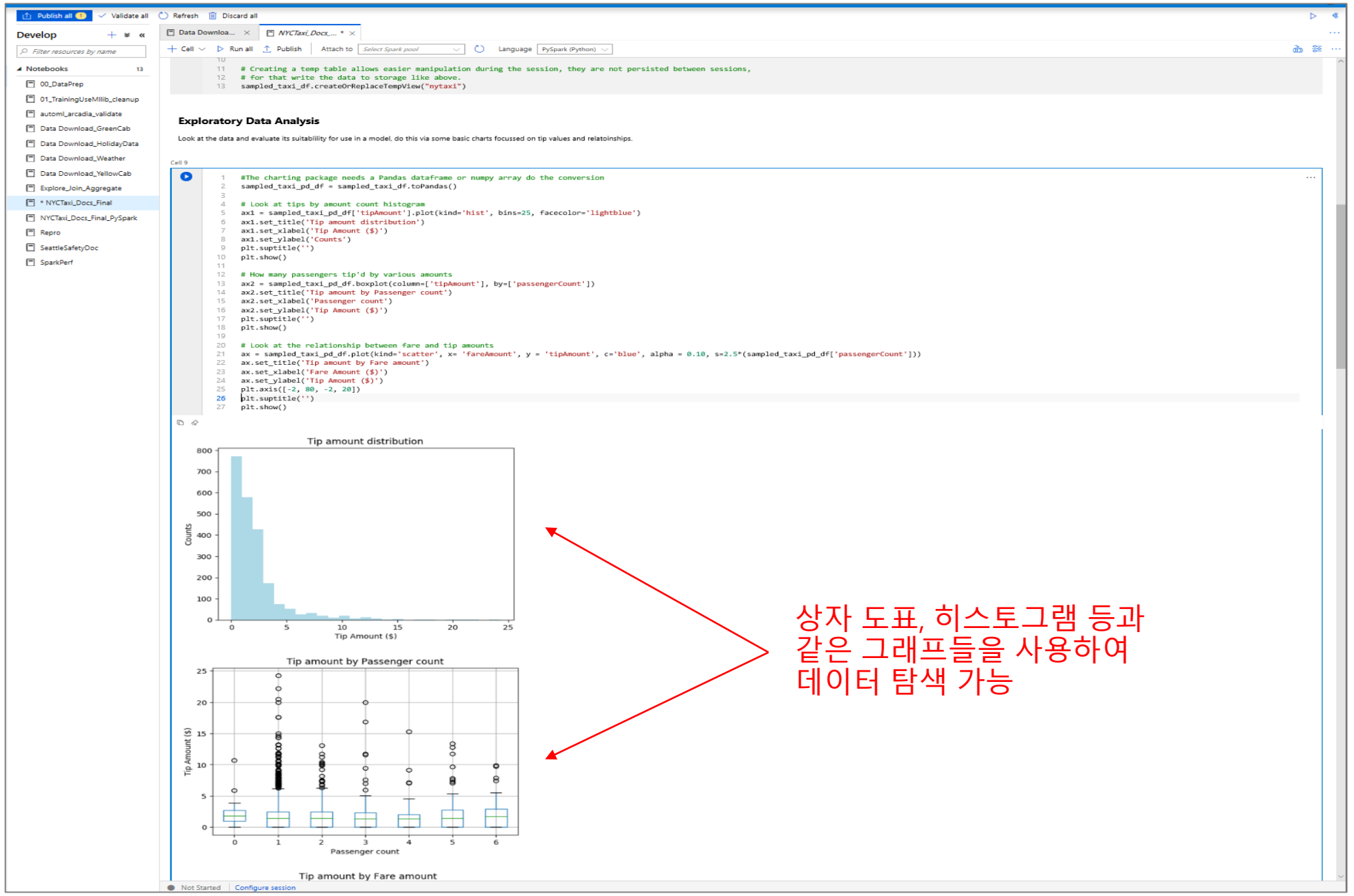


차트 포멧으로
결과 출력



상자 도표, 히스토그램 등과
같은 그래프들을 사용하여
데이터 탐색 가능

Library Management - Python

Overview

Spark pool 수준에서 python library 추가 가능

Benefits

Input requirements.txt in simple pip freeze format

Add new libraries to your cluster

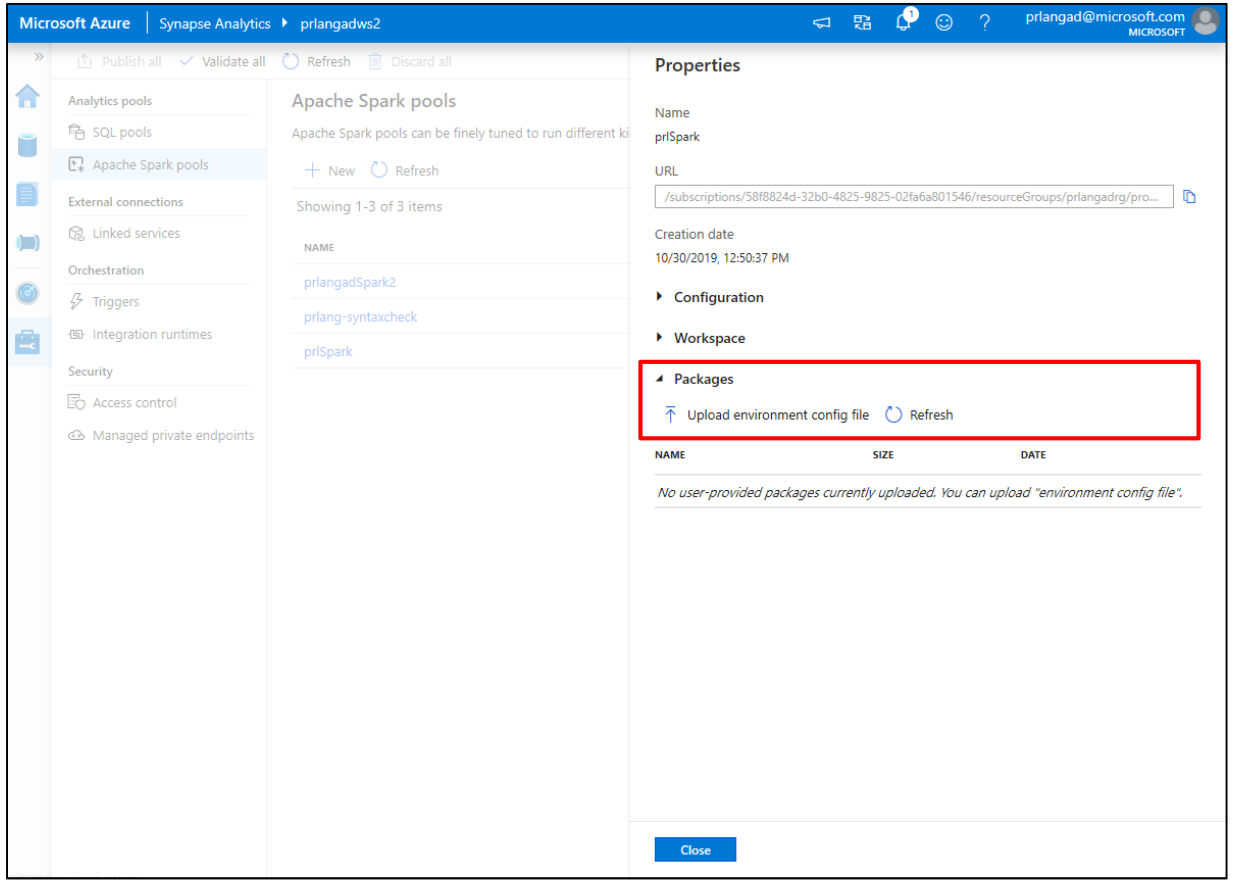
Update versions of existing libraries on your cluster

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

Constraints

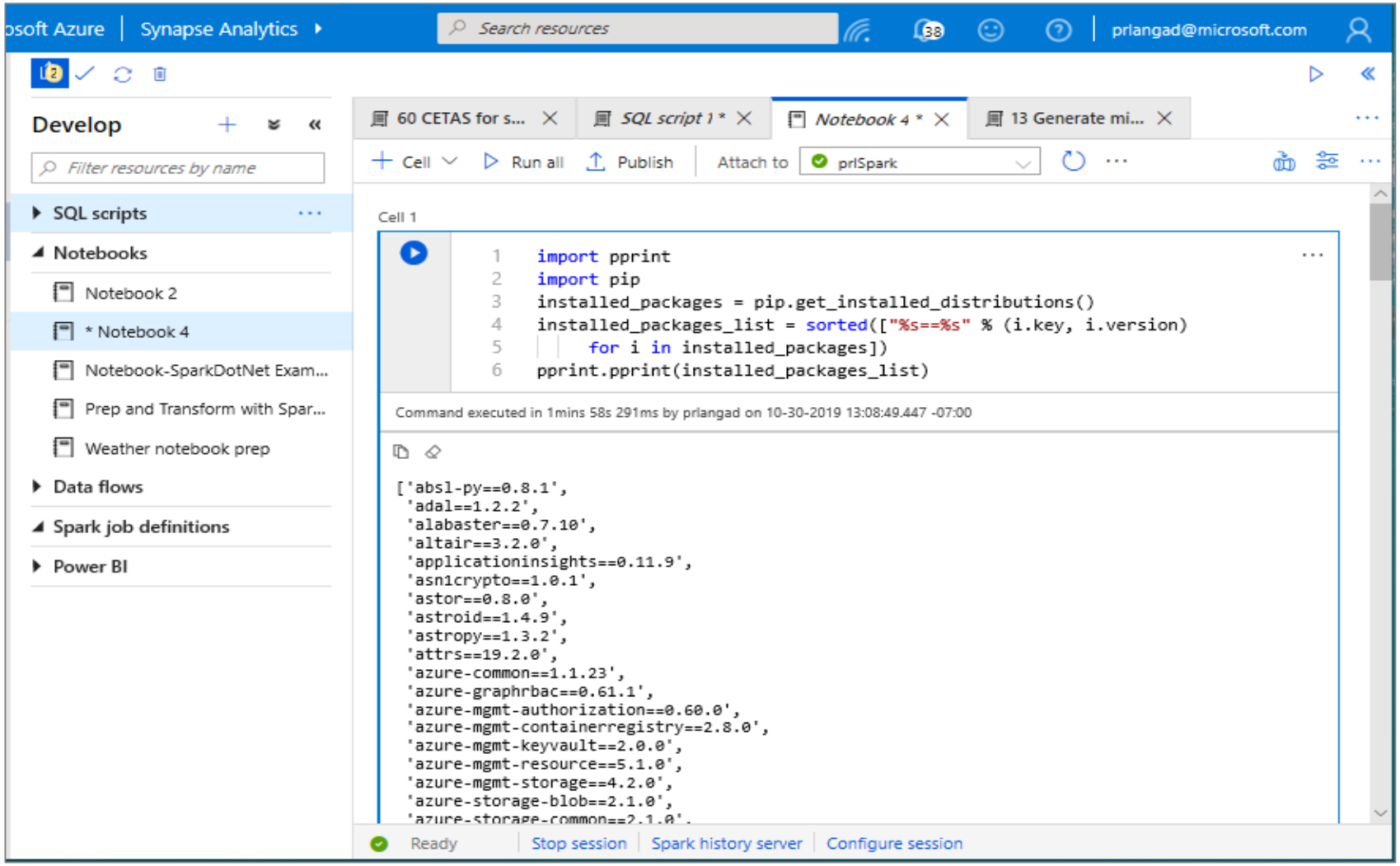
The library version must exist on PyPI repository

Version downgrade of an existing library not allowed



Library Management - Python

설치된 library 리스트 및 버전 정보 확인 가능



Spark ML Algorithms

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

Microsoft Spark Utilities

Overview

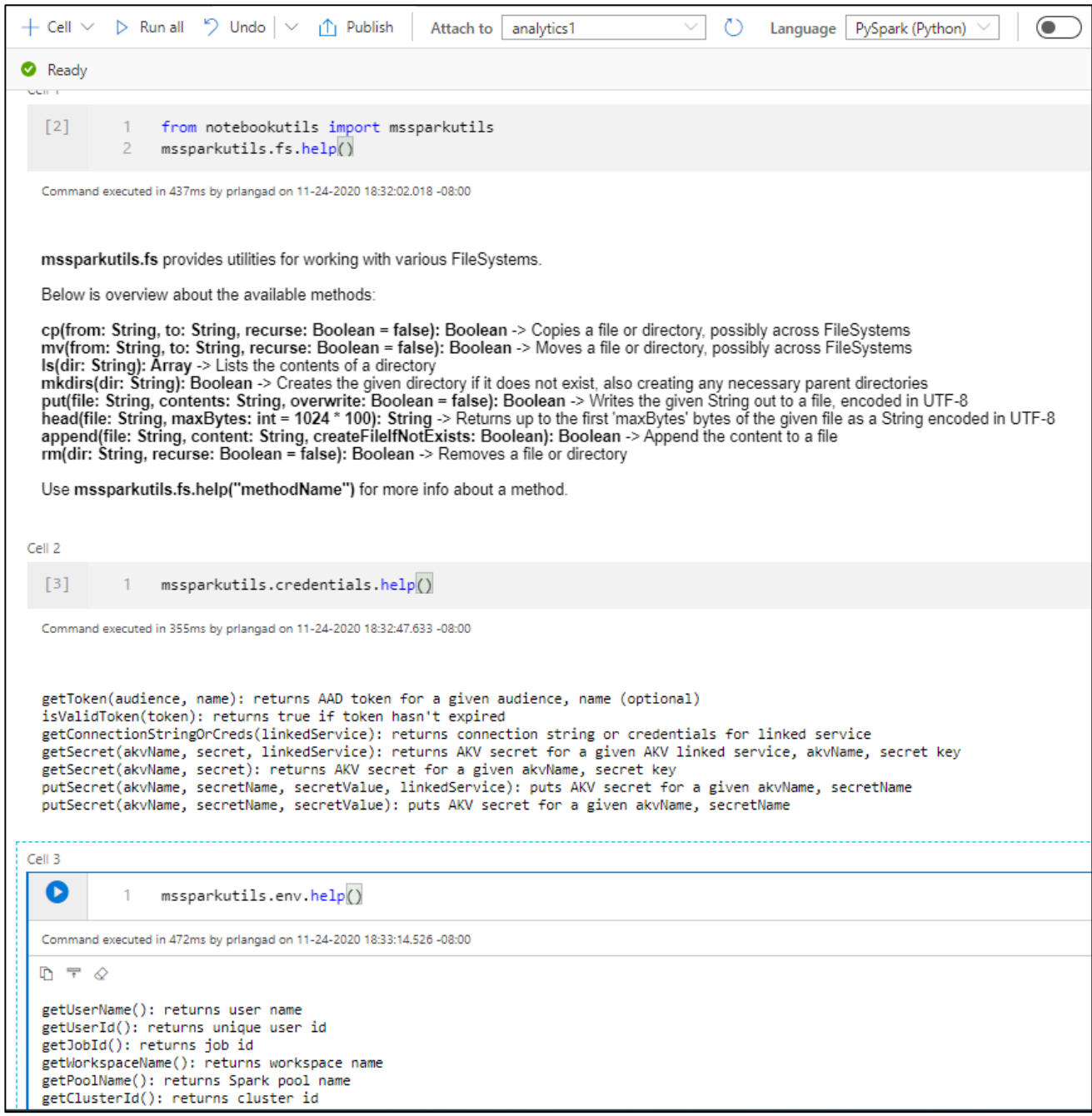
ADLS Gen2와 Azure Blob Storage를 포함한 다양한 file system들과 작업 가능한 유틸리티 제공

Benefits

It supports multiple methods for file systems such as List, Copy, Move, Write, Append, Delete file or directory, View file properties, Create new directory, Preview file content.

It supports environment utilities to get username, user id, job id, workspace name, pool name, cluster id.

It supports to get the access tokens of linked services and manage secrets in Azure Key Vault.



Hyperspace

Overview

Hyperspace를 이용하여 Apache Spark 사용자가 인덱스를 생성할 수 있음

Benefits

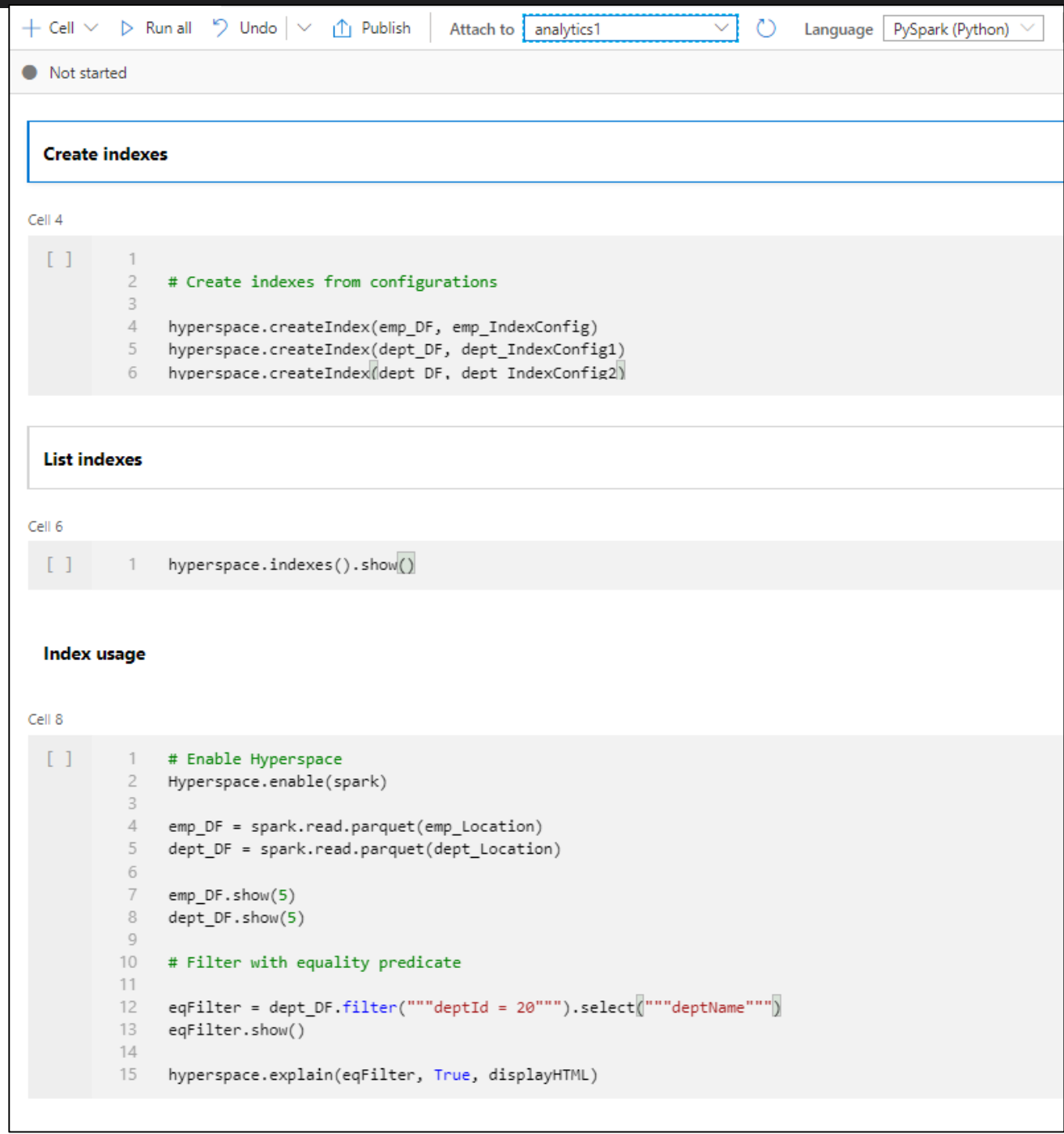
It helps accelerate your workloads or queries containing filters on predicates with high selectivity or a join that requires heavy shuffles.

Maintain the indexes through a multi-user concurrency model.

Leverage these indexes automatically, within your Spark workloads, without any changes to your application code for query/workload acceleration.

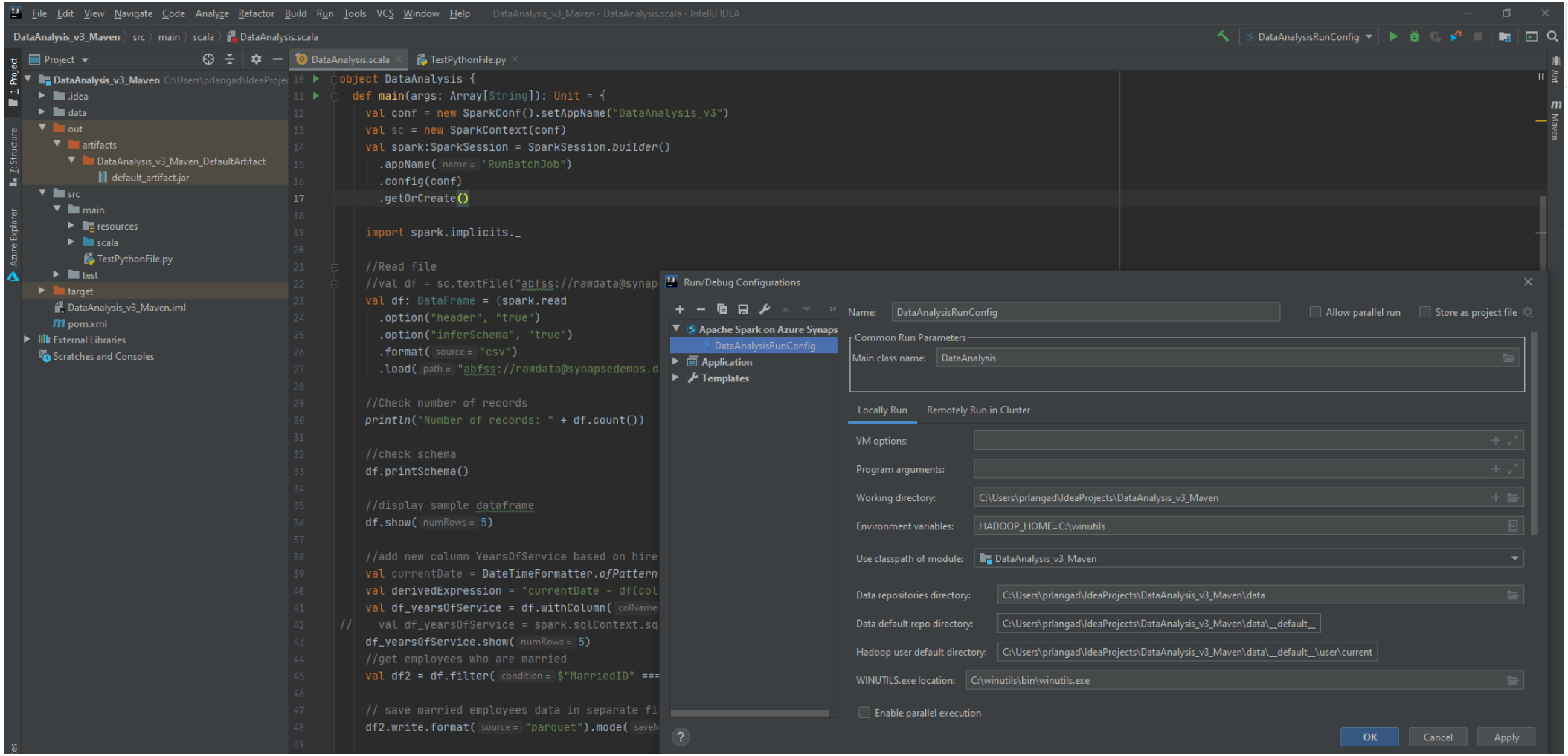
It supports index operations as create index, list index, restore index, delete index, vacuum index

Languages supported: Scala, Python, .NET

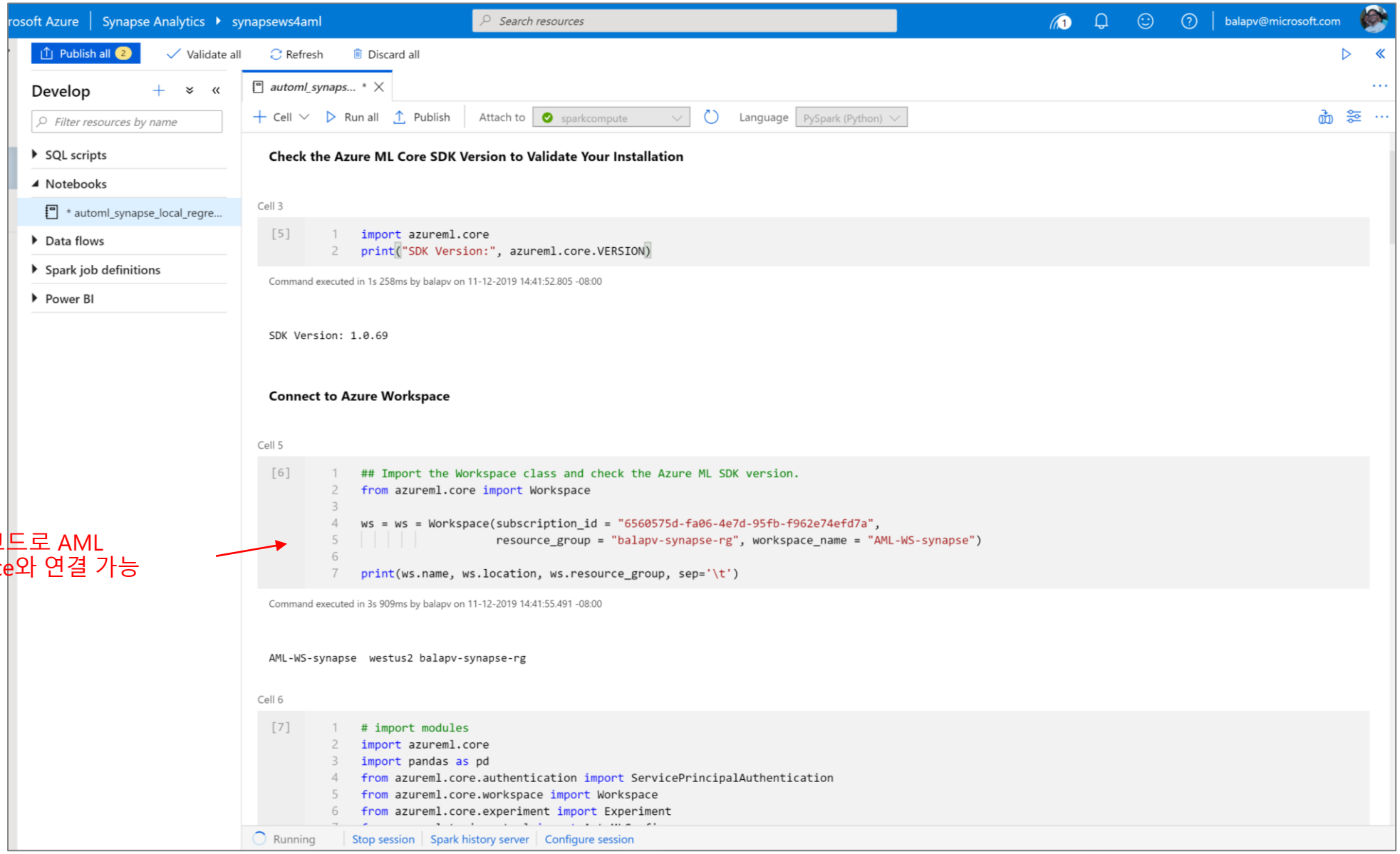


IntelliJ IDE

Apache Spark application과 spark pool에 submit 할 수 있는 IntelliJ 플러그인 지원

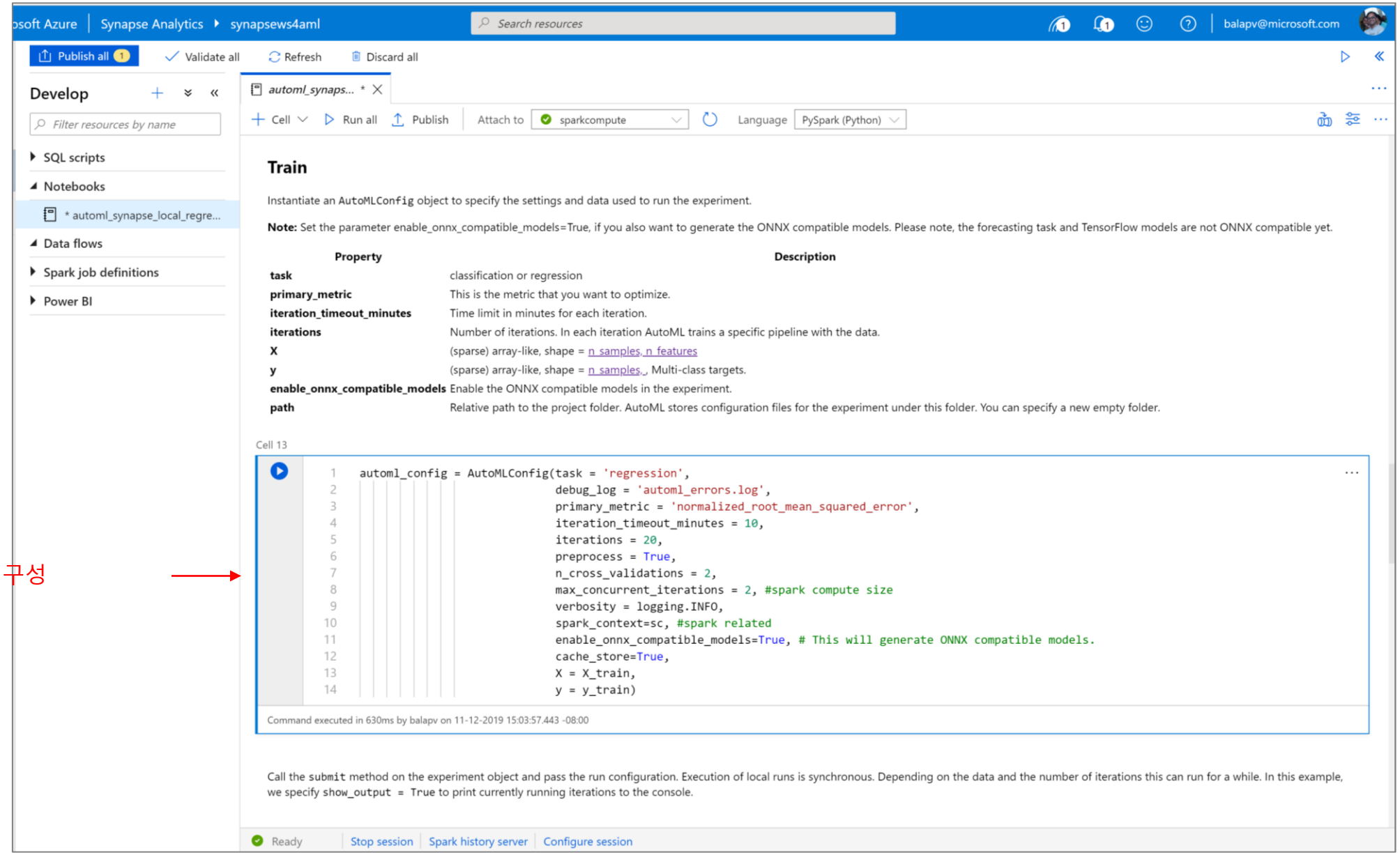


Synapse Notebook: Connect to AML workspace



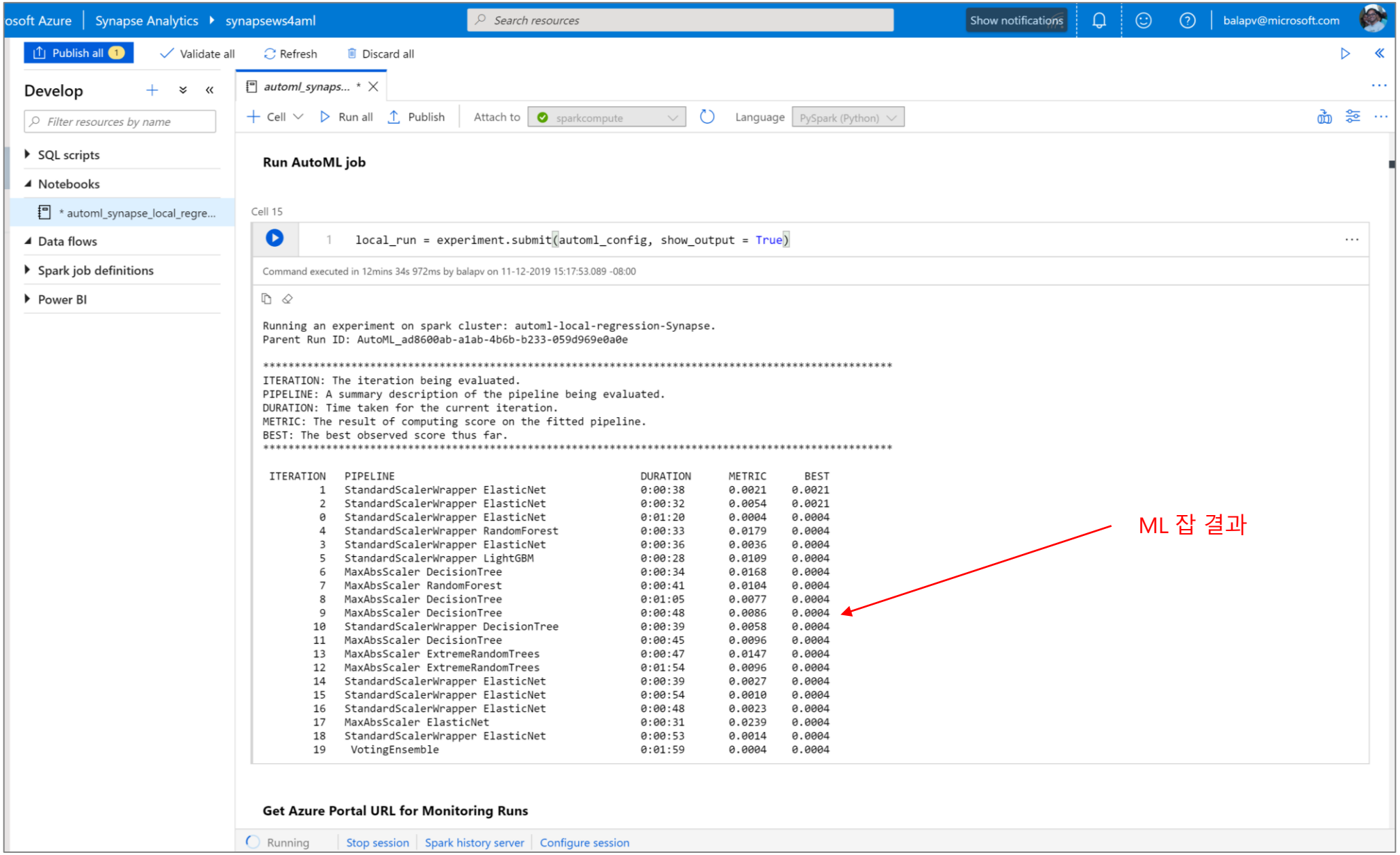
간단한 코드로 AML workspace와 연결 가능

Synapse Notebook: Configure AML job to run on Synapse



Parameter 구성 →

Synapse Notebook: Run AML job



End