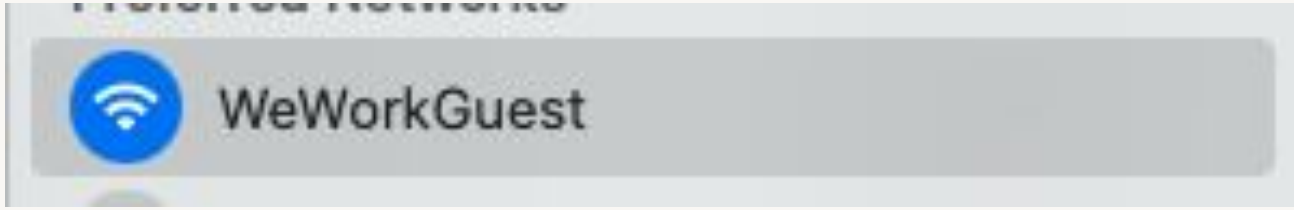
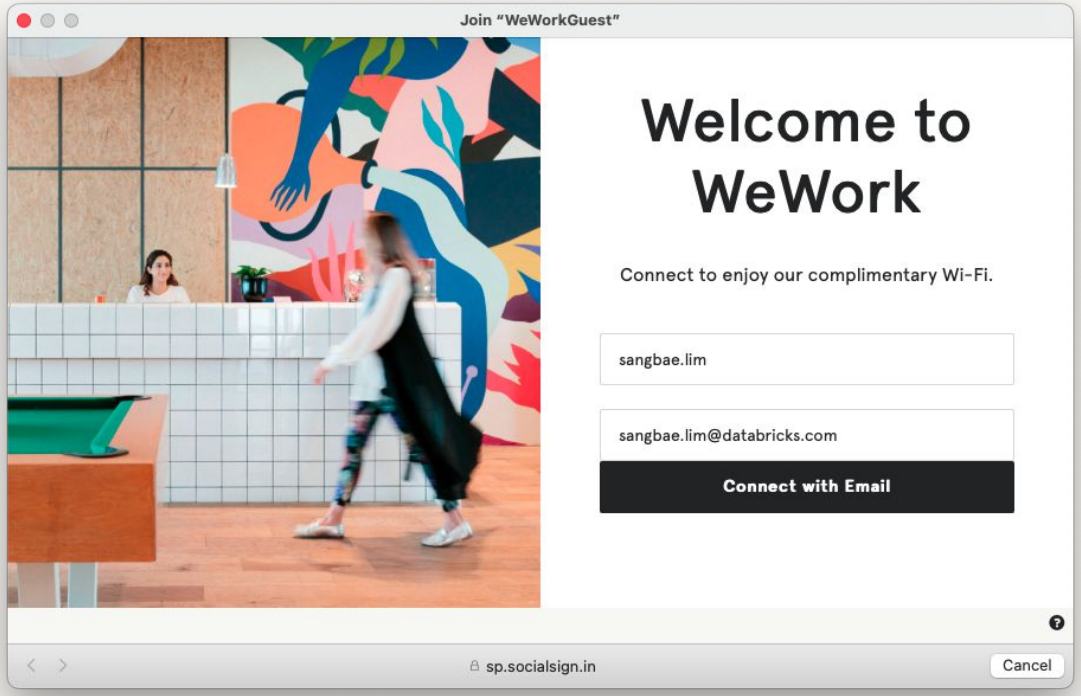
WIFI 접속 안내(WeWorkGuest)

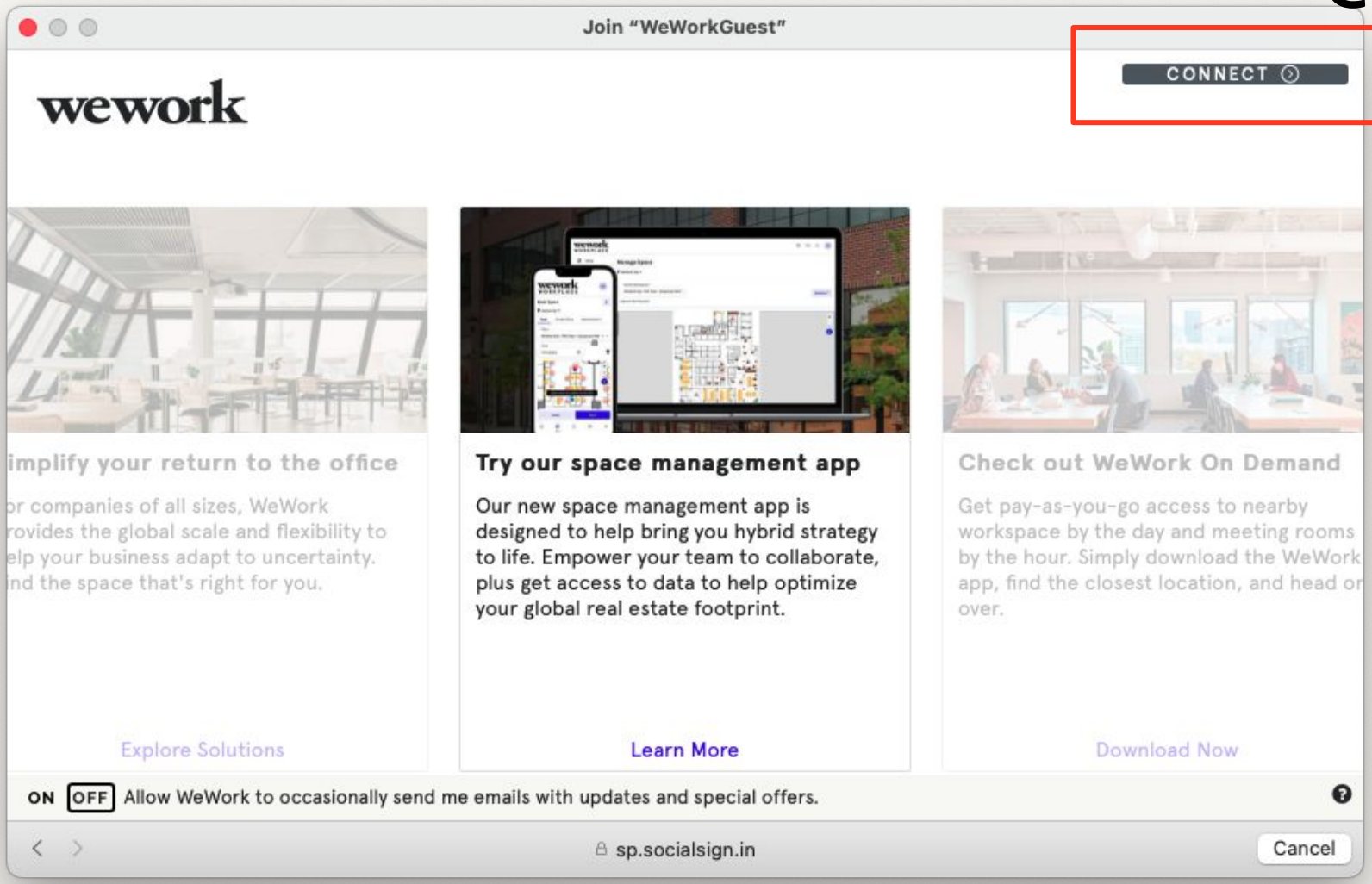




1. WeWorkGuest 선택 후
2. 이름/이메일 주소 입력
3. 다음 화면에서 오른쪽 상단 Connect 클릭

WIFI 접속 안내(WeWorkGuest)

Connect 클릭



워크샵 자료 / Hands on 등록

Hands-On Cloud Lab 등록

https://bit.ly/48eaVNM

워크샵 자료(발표 자료, 실습 노트북)

https://tinyurl.com/db-202309-hol

금일 18시 이후에 접속이 제한되오니, 다운로드 받아 주세요

* 도움이 필요하시면 주변 Databricks 직원에게 문의해 주세요

Databricks DE+BI Bootcamp

Hands-On Lab

Youngkyong Ko | Hyunwoo Shin | Databricks Korea
Last updated Sep. 2023

Course Objectives

1. 데이터브릭스에서 제공하는 서비스의 범위를 이해 합니다.
2. **Databricks Workspace** 에서 **통합 Notebook**으로 일반적인 코드 개발 작업을 할 수 있습니다.
3. **SQL**을 사용하여 다양한 소스에서 데이터를 추출 및 변환 작업을 수행하고, 결과 데이터를 **Delta Lake**에 적재 할 수 있습니다.
4. Databricks **Workflows Jobs** 으로 데이터 파이프 라인을 설정하고 실행을 할 수 있습니다.
5. **Databricks SQL** 을 사용해서 적재된 데이터를 조회하고 시각화해서 대시보드를 구성할 수 있습니다.

Agenda

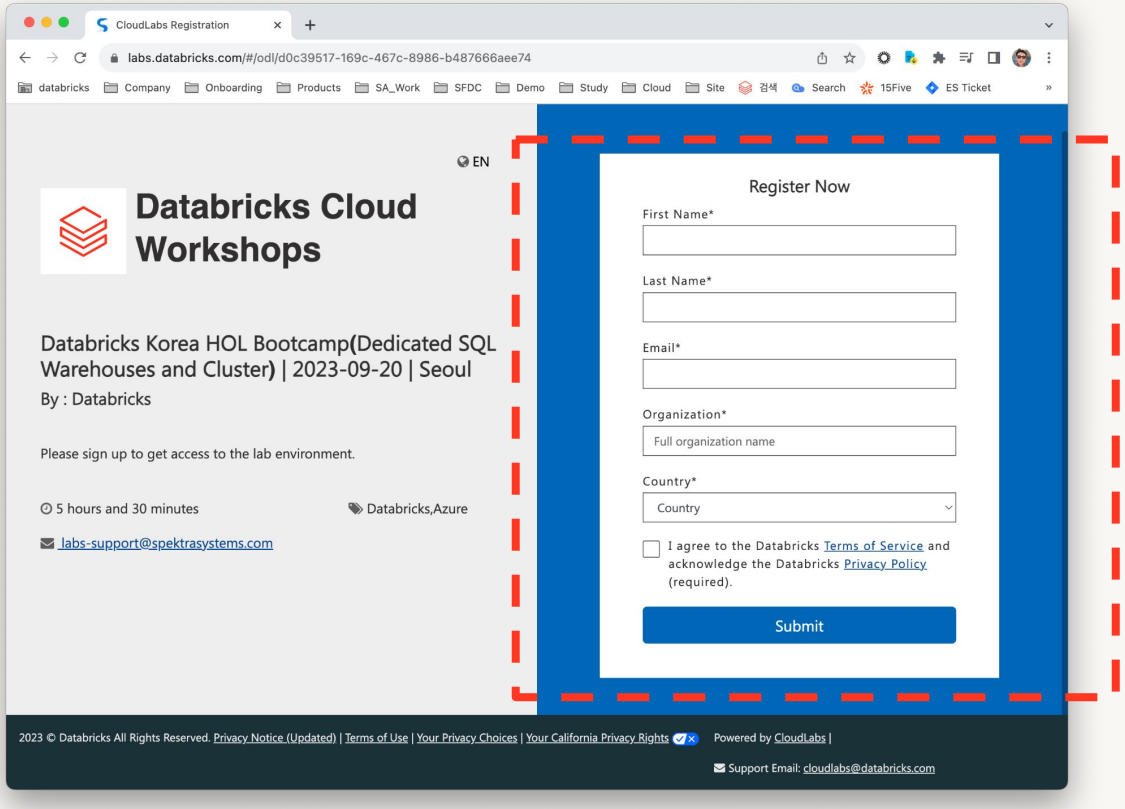
시간	주제	실습	과제
14:00	인사말 & 랩 환경 세팅		
14:15~15:15	Databricks Lakehouse 소개		
	Databricks 아키텍처		
	Workspace - 클러스터 생성	DE01	
	데이터브릭스 Notebook	DE01(1-1, 1-2)	DE01(1-3)
	델타 레이크 - Delta Table	DE02(2-1)	DE02(2-2)
Break			
15:45	오케스트레이션 - Workflows	DE03	
	Databricks SQL	DW04	
17:00	Course wrap up		

Cloudlabs 환경 설정

Cloudlabs 환경 설정

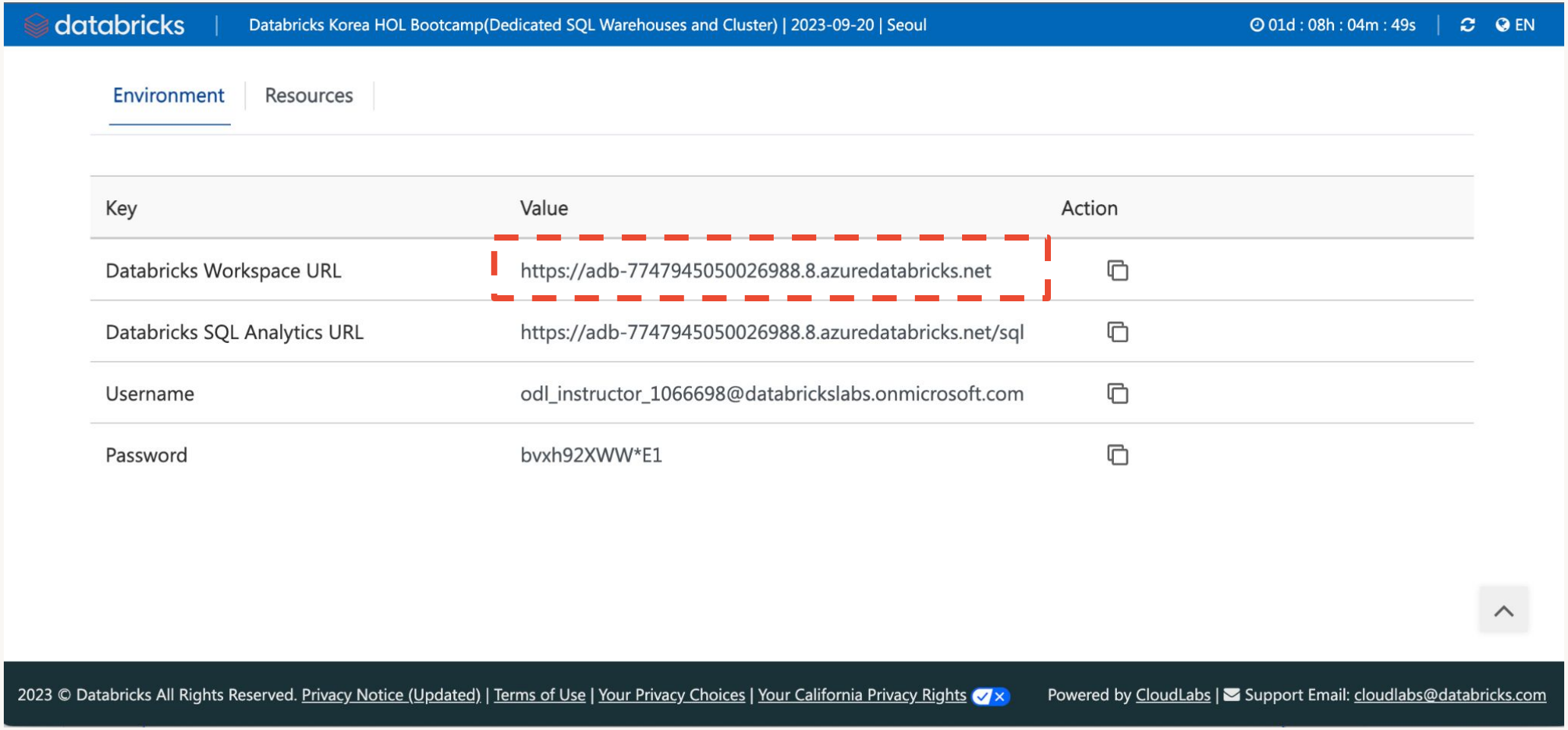
https://bit.ly/48eaVNM 로 접속하셔서 Cloudlabs에 등록해주시기 바랍니다.

* 외부에서 사내 이메일 확인이 어려우시거나 접속이 어려우신 경우 개인 이메일로 등록해주세요.
** 도움이 필요하신 경우 주변 Databricks 직원에게 문의해주세요.



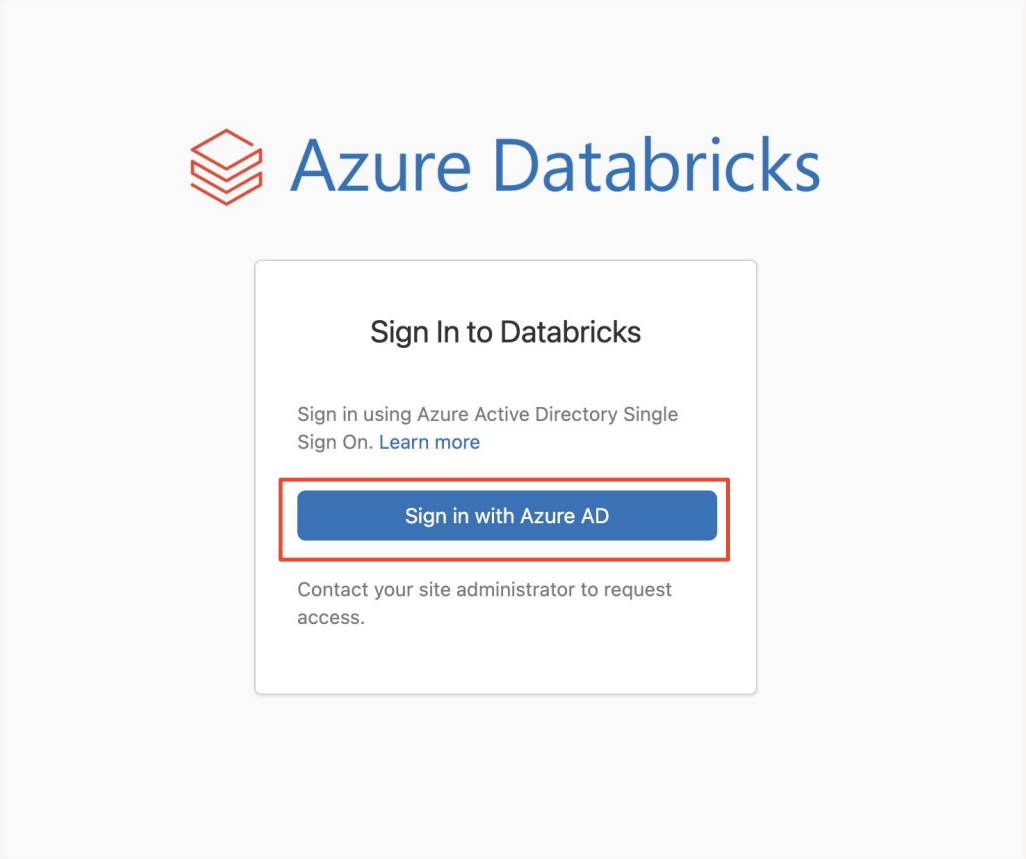
Cloudlabs 환경 세팅

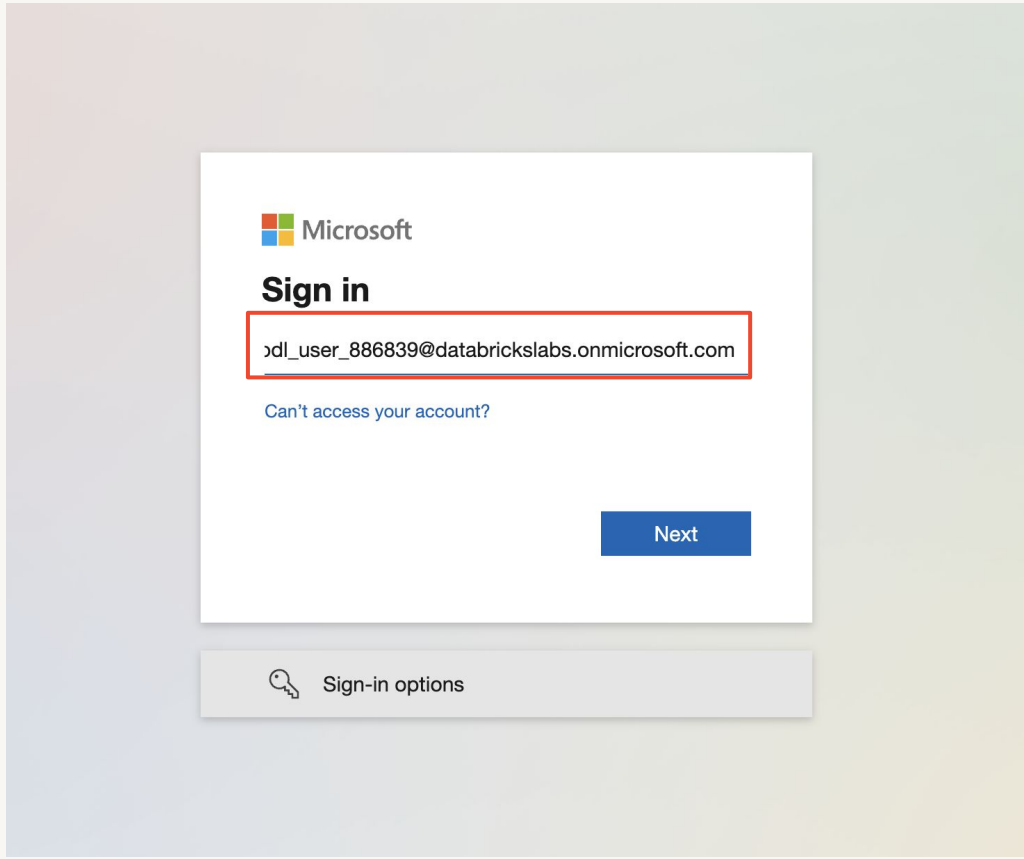
크롬의 시크릿 모드에서 접속



Sign in

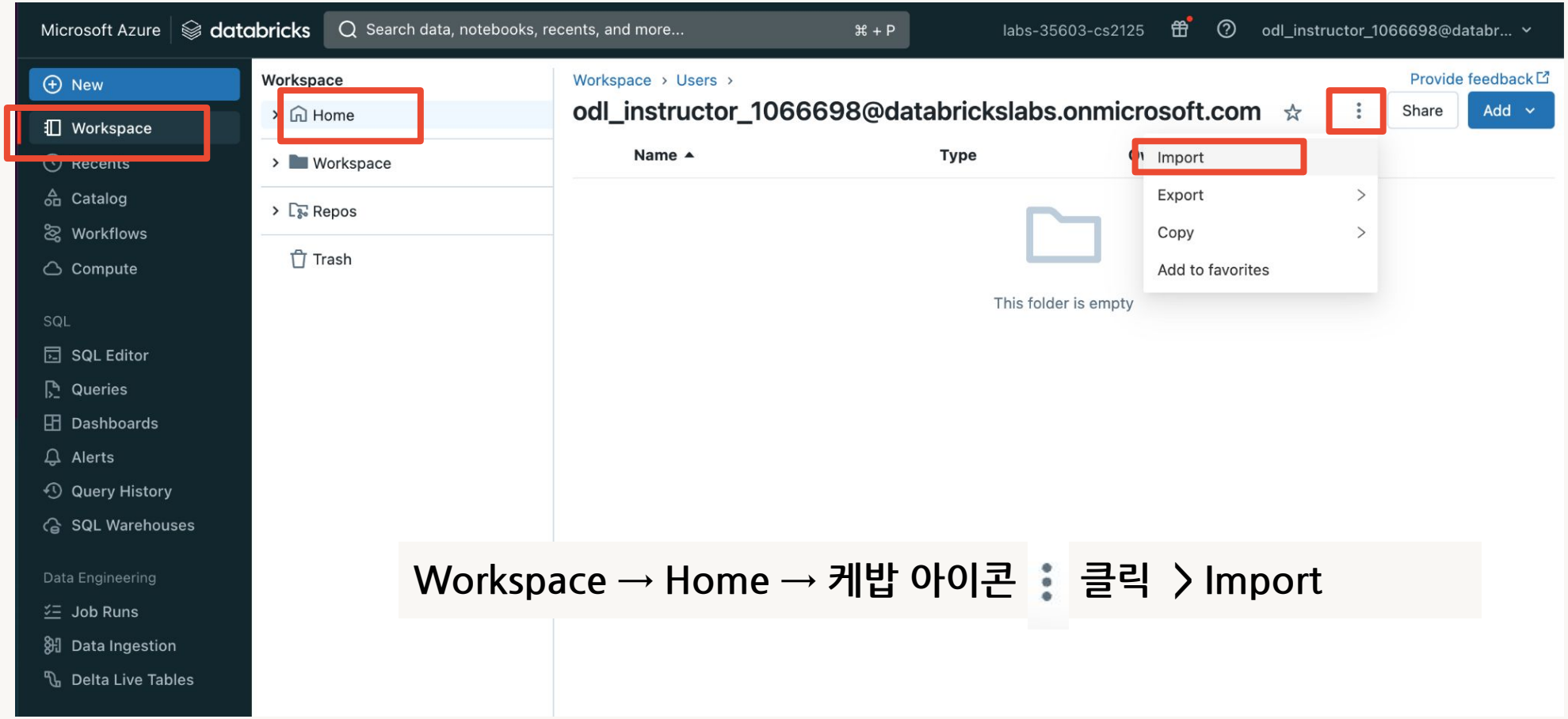
부여받은 Username / Password 로 인증



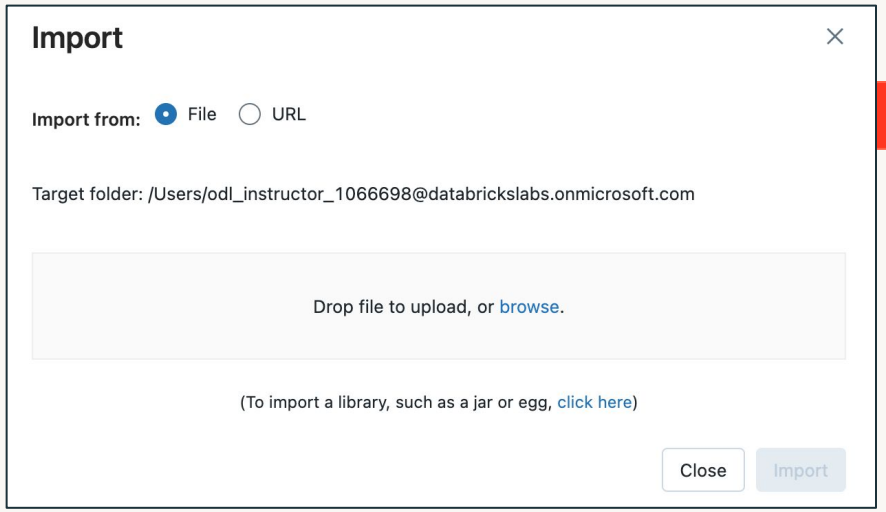


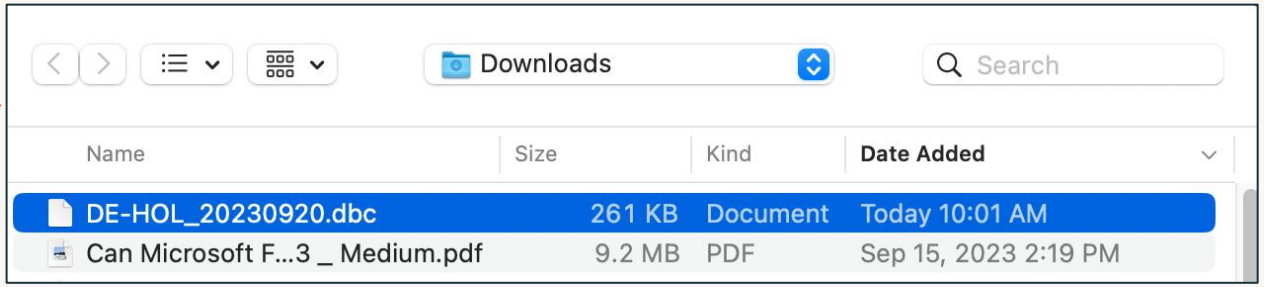
실습 노트북 다운로드 후 Import

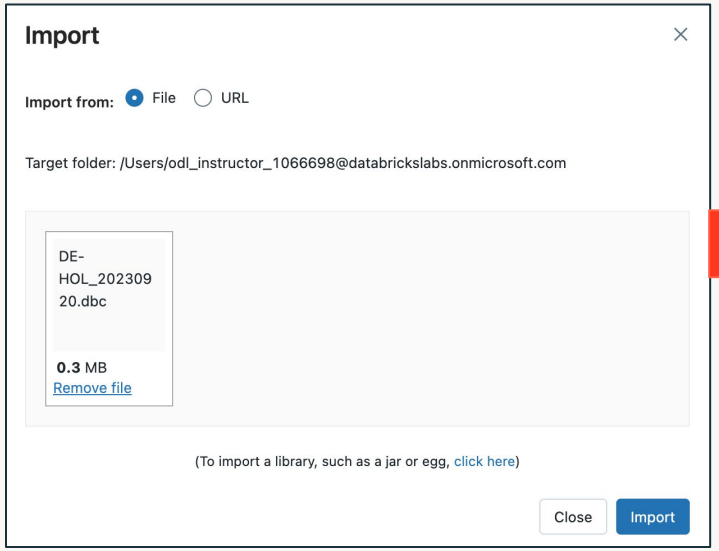
https://tinyurl.com/db-202309-hol

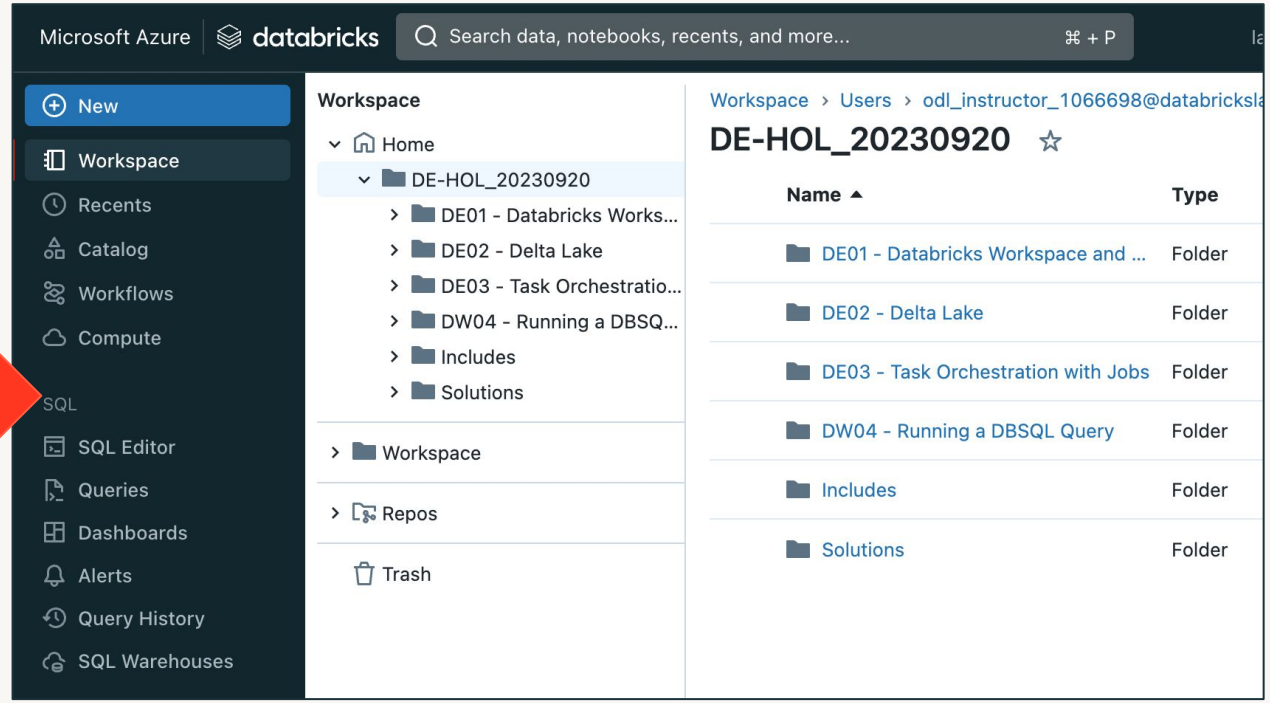


노트북 Import 후 확인









Databricks Lakehouse

databricks

The Lakehouse Company

데이터 레이크하우스의 개척자

Lakehouse

데이터, 분석, AI 워크로드를 통합하는 클라우드 데이터 플랫폼

성공적인 오픈소스 창시자

APACHE Spark™ DELTA LAKE mlflow™

글로벌 10,000개 이상의 고객사

10,000+

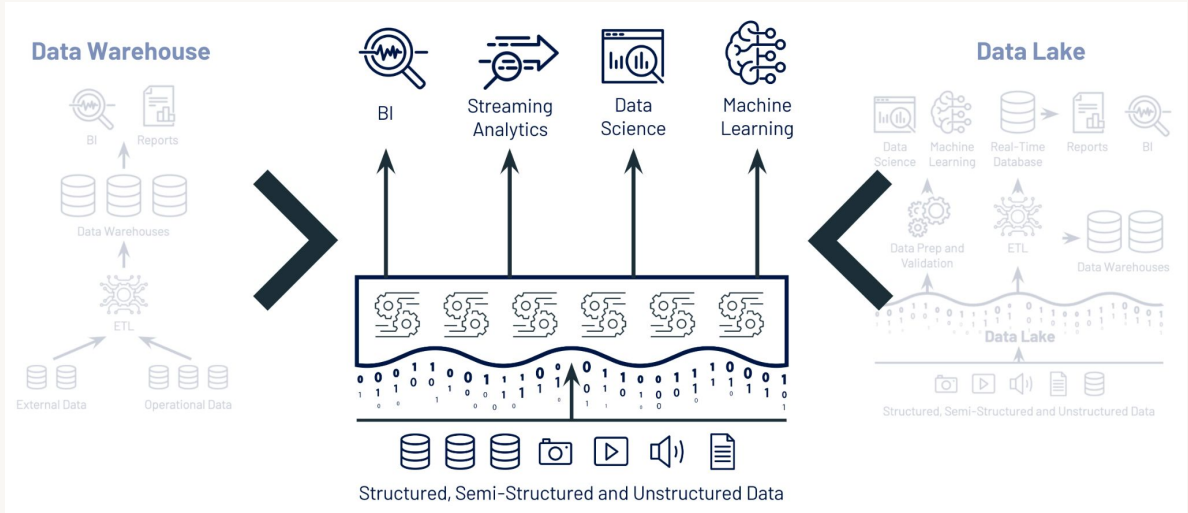
across the globe

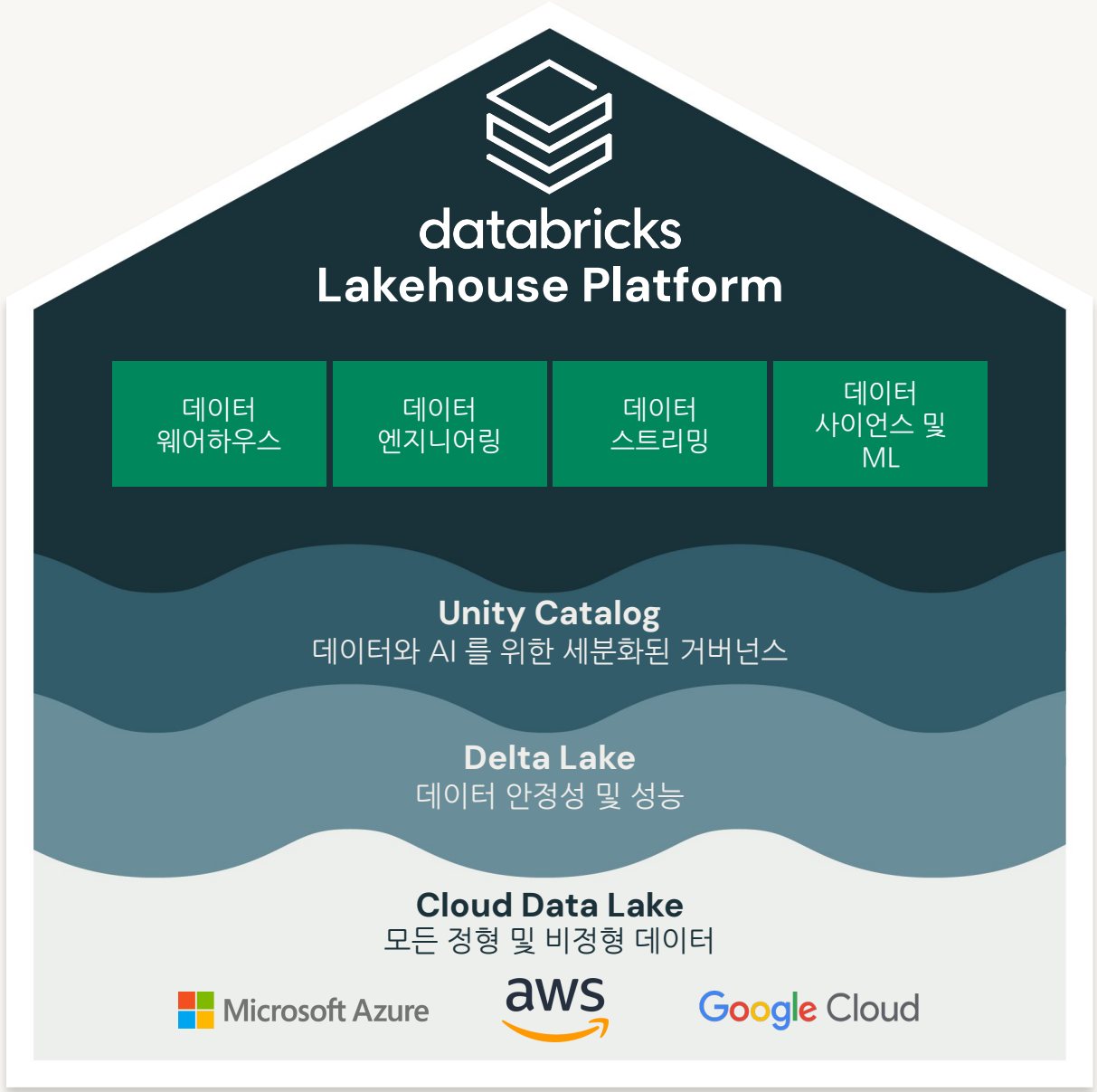
Data Lakehouse 아키텍처

Data Warehouse와 Data Lake의 사일로 문제를 단일 플랫폼으로 통합

일반적인 데이터 처리/분석/ML 환경

Databricks Lakehouse 환경





Databricks Lakehouse Platform

Simple

단일 플랫폼에서
데이터 웨어하우징 및 AI 사용 사례 통합

Multicloud

클라우드 전반에서 일관된 단일 데이터 플랫폼

Open

오픈 소스와 개방형 표준을 기반으로 구축

Data reliability and performance

Foundation of the lakehouse

Open-source table storage
with fastest
out-of-the-box performance

+

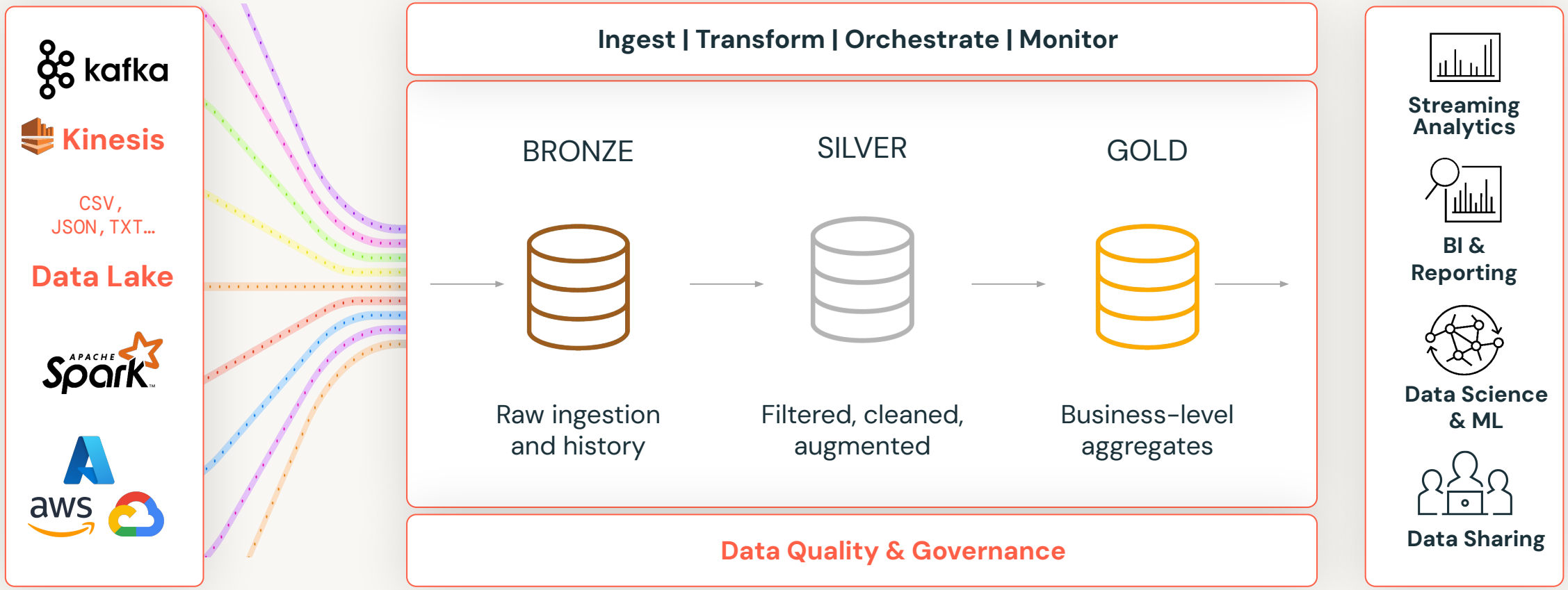
Photon

State-of-the-art
vectorized engine
for blazing fast queries

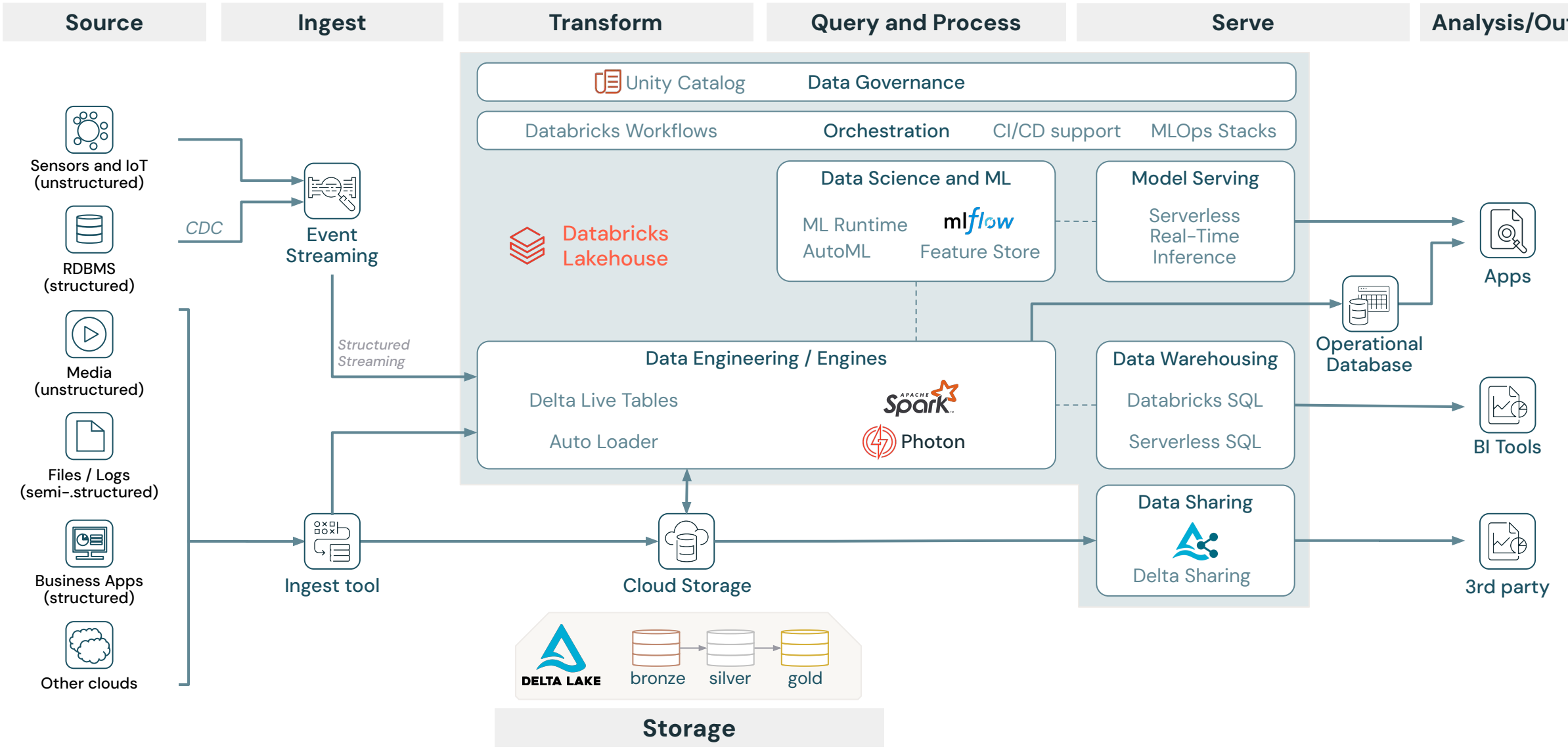
Scale, speed, and cost savings

레이크하우스에서의 데이터 흐름

다양한 소스 데이터를 간결하고 안정적인 방식으로 처리



Databricks 레이크하우스 레퍼런스 아키텍처



레이크하우스에서의 데이터 엔지니어링

높은 신뢰성과 빠른 성능으로 대규모 데이터를 안정적으로 처리

- Delta Lake 기반으로 데이터 처리의 신뢰성과 빠른 성능 제공
- Delta Live Tables (DLT)를 활용하여 ETL 파이프라인을 빠르게 개발하고 쉽게 배포, 운영
- Databricks Workflow를 이용하여 작업 흐름을 관리하고 쉽게 오케스트레이션

레이크하우스에서의 데이터 웨어하우징

Data Lake 기반의 고성능 DW

- 별도의 적재 과정없이 Delta Lake 기반으로 높은 쿼리 성능과 동시처리 제공
- 분석가를 위한 네이티브 SQL 인터페이스
- 손쉬운 BI 시각화 대시보드 및 경보 구성
- 다양한 BI 툴과 연동하여 Data Lake 의 최신 데이터를 바로 쿼리

레이크하우스에서의 AI/ML

머신러닝 전체 과정을 쉽게 개발/관리

Machine Learning

- 데이터 준비, 모델 개발, 배포까지 머신러닝 전체 과정을 mlflow로 쉽게 관리
- 추출된 feature 들을 Feature store로 공유, 재활용
- 원클릭으로 모델 빌드 AutoML 기능 제공

Data Science

- 협업 가능한 노트북
- 파이썬, 스칼라, R, SQL 언어 네이티브 지원

레이크하우스의 통합 거버넌스 체계 - Unity Catalog

다양한 데이터와 분석 asset 들을 통합 관리하는 메타스토어와 거버넌스 체계 제공

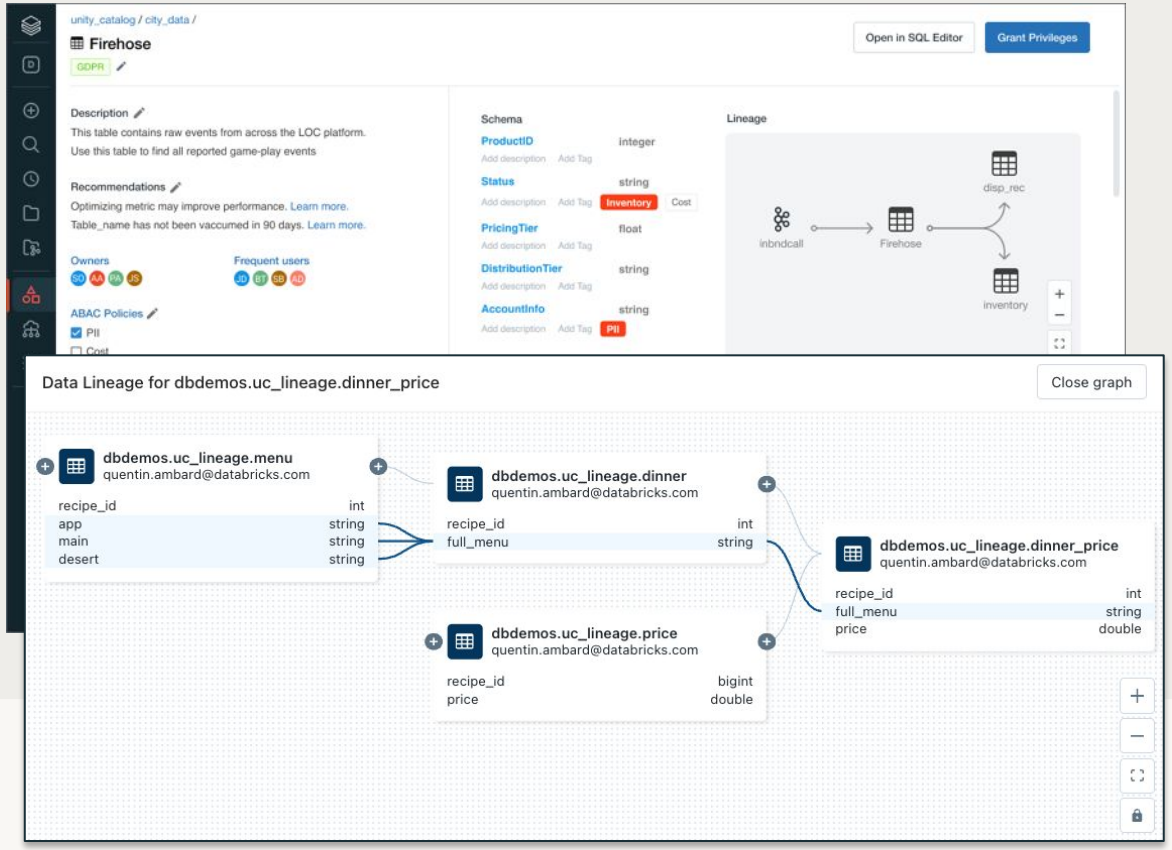
모든 데이터 asset 관리 및 통제

- 웨어하우스, 테이블, 컬럼
- 데이터 레이크, 파일
- 머신 러닝 모델
- 대시보드 및 노트북*

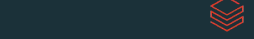
주요 기능

- 데이터 리니지 (Data lineage)
- 메타데이터 검색과 탐색
- 데이터 공유 (Delta Sharing)
- 감사 (Auditing)

* : planned features



Databricks Architecture



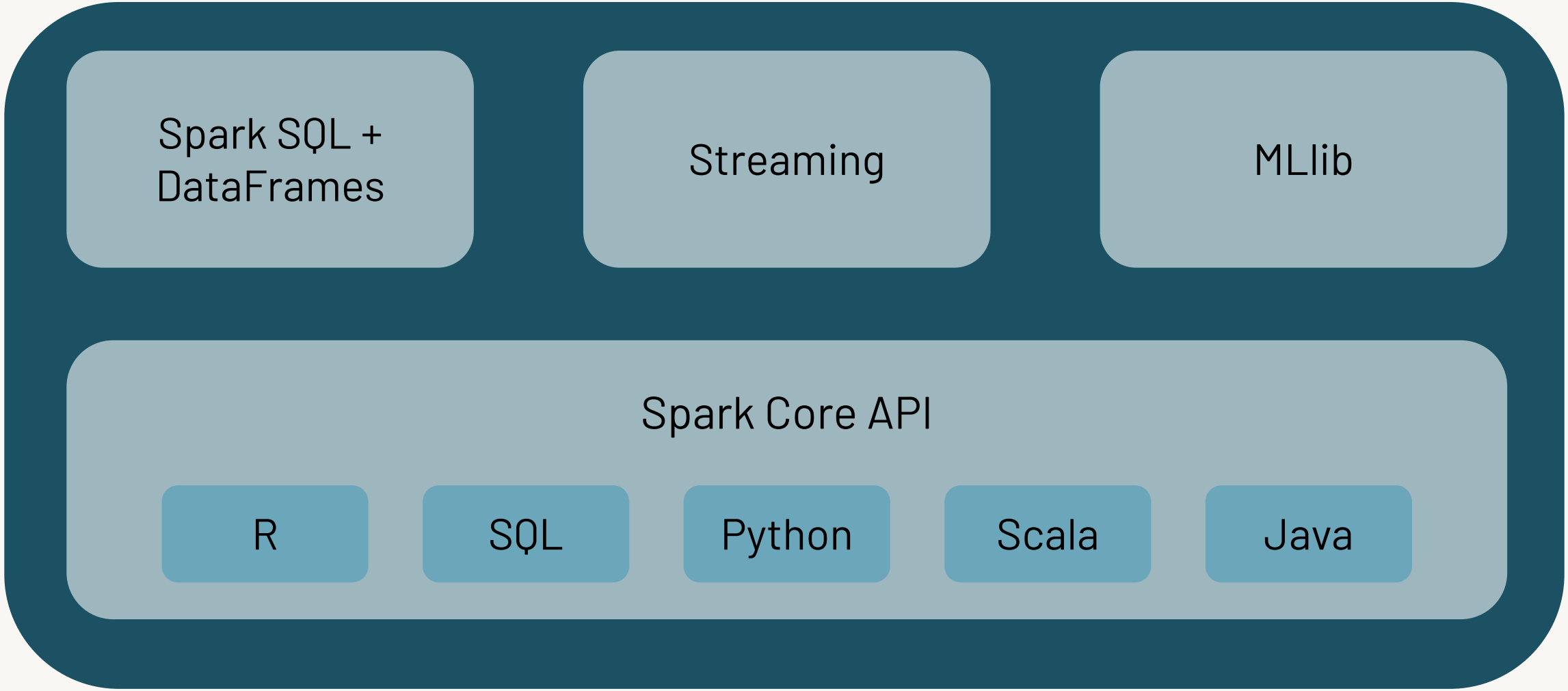
- 빅데이터 프로세싱의 사실상 시장 표준 통합 분석 엔진
- 데이터 프로세싱 영역에서 가장 큰 오픈 소스 프로젝트
- 데이터브릭스의 창립자들이 만든 기술

빠르고

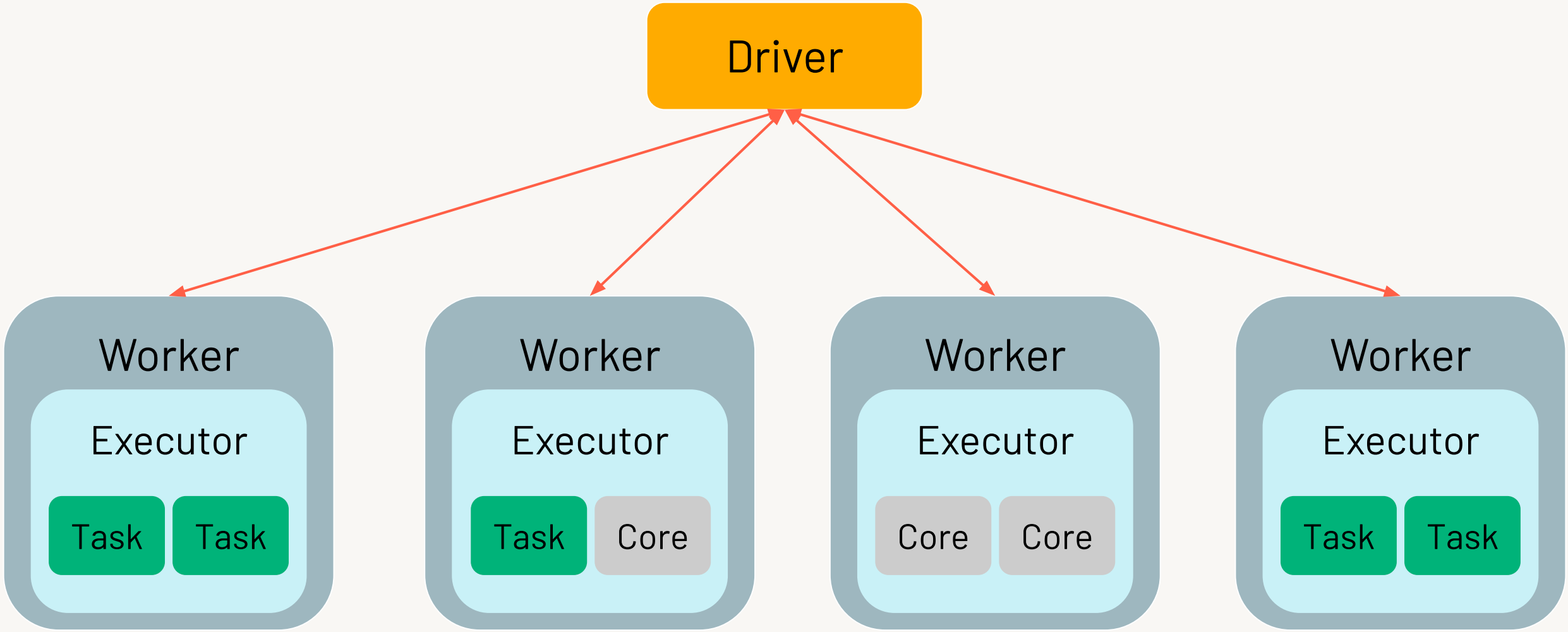
사용이 쉽고

다용도

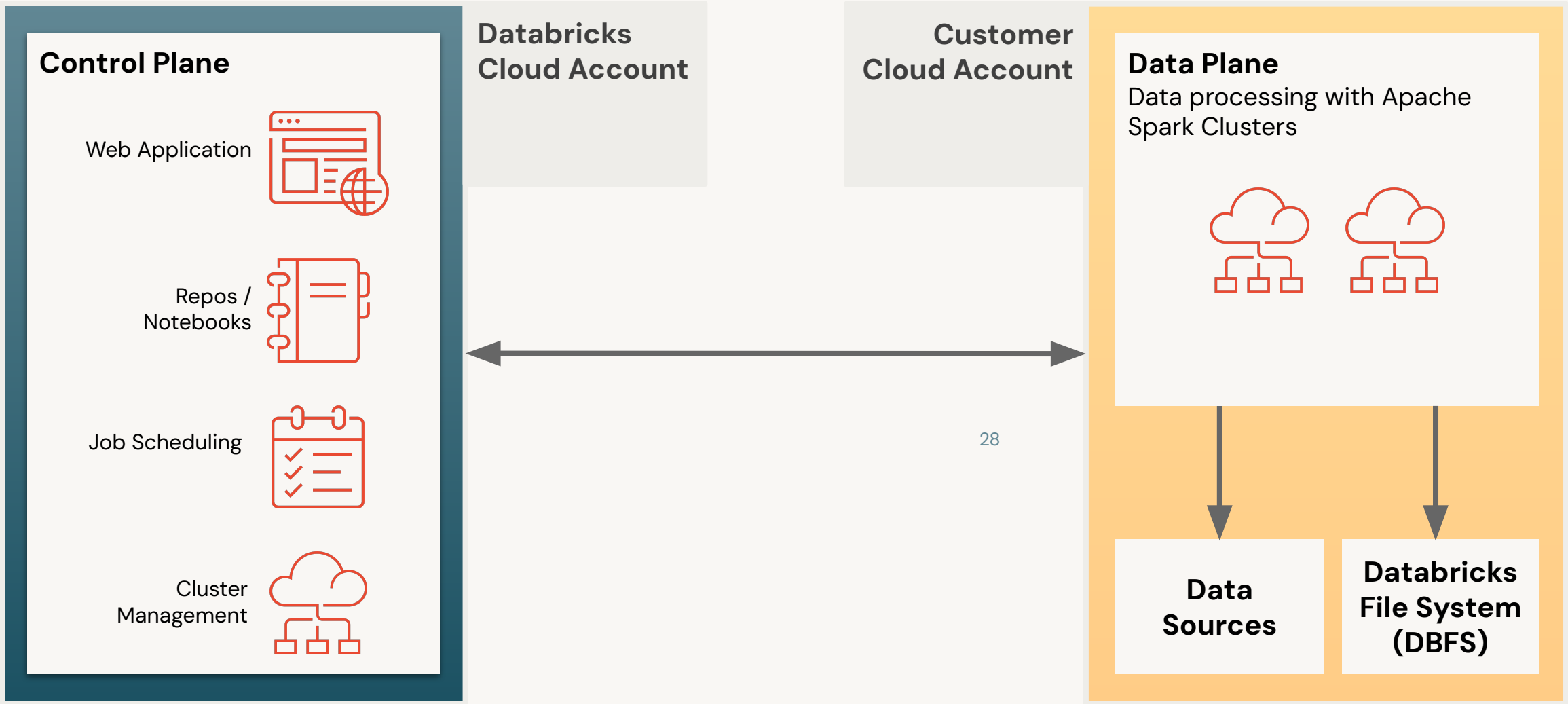
Spark API



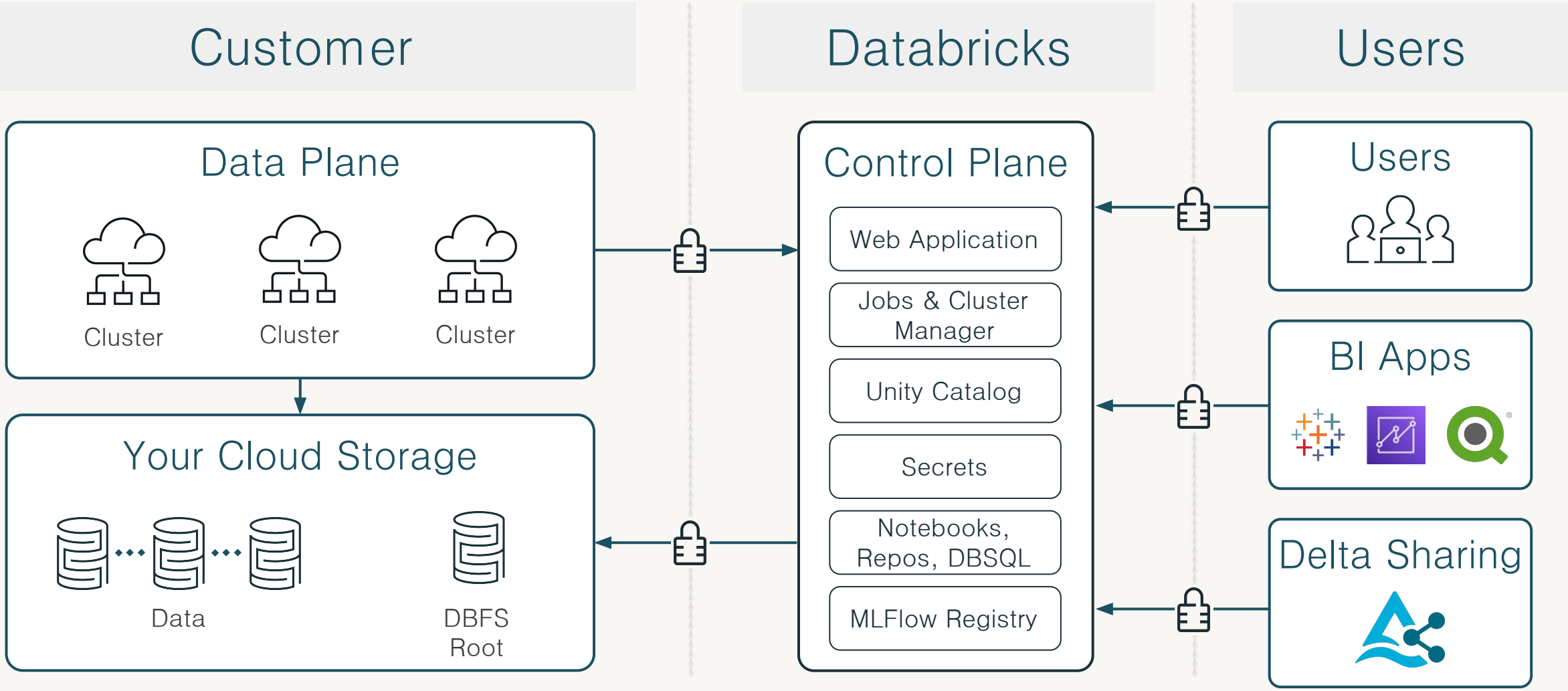
Spark Cluster



Databricks Architecture



Databricks 클라우드 플랫폼 배포 아키텍처



Clusters

Types

All-purpose Clusters

여러 사용자가 대화형 노트북을 사용하여
인터랙티브 분석과 협업을 할 수 있는
다목적 클러스터

Workspace 또는 API로 클러스터 생성
클러스터 종료 후 재시작 가능

Job Clusters

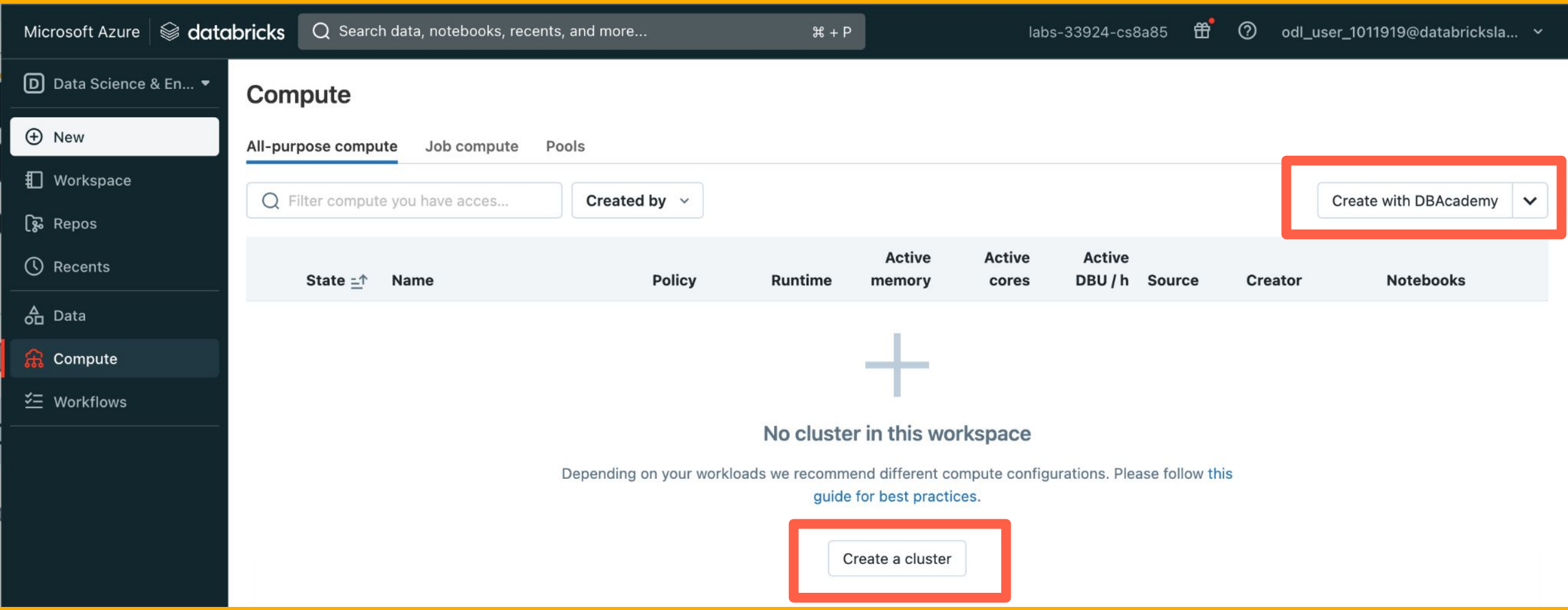
자동화된 작업 실행을 위한 클러스터

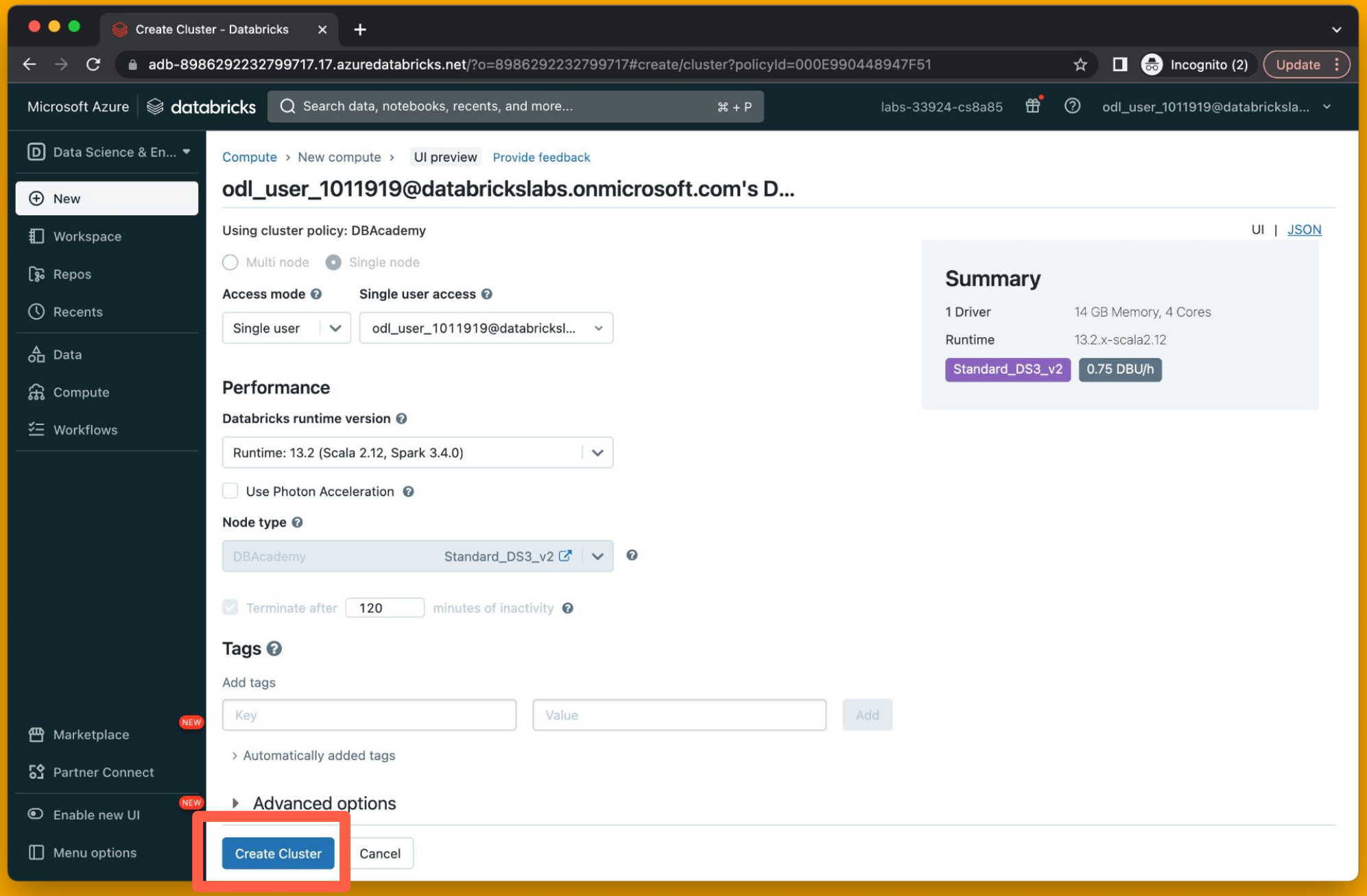
Databricks 작업 스케줄러가 작업 시작시
생성하고 작업 종료시 자동 종료

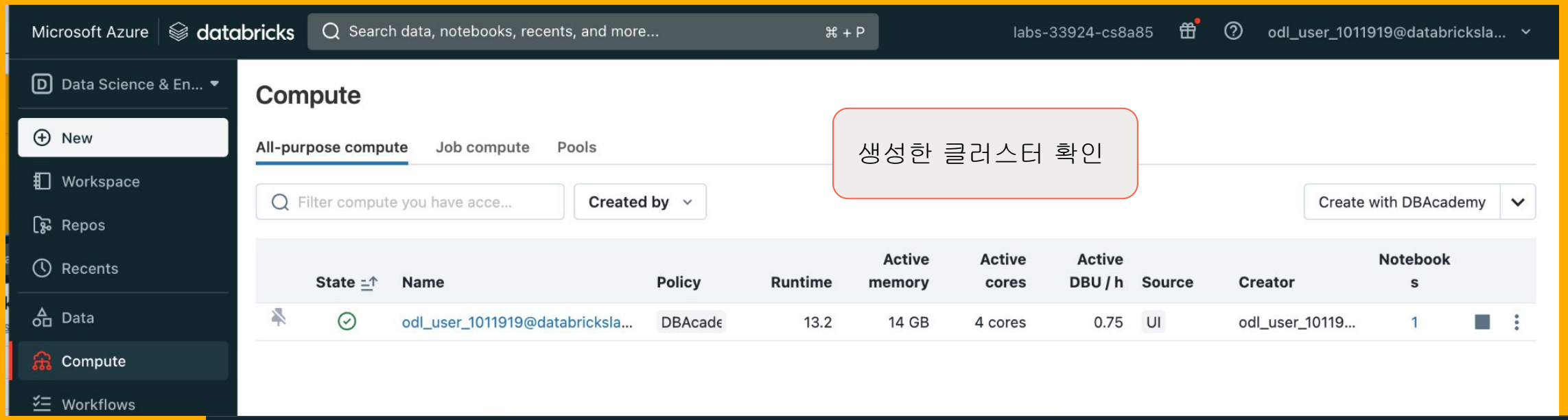
상대적으로 저렴한 비용으로 비용 절감

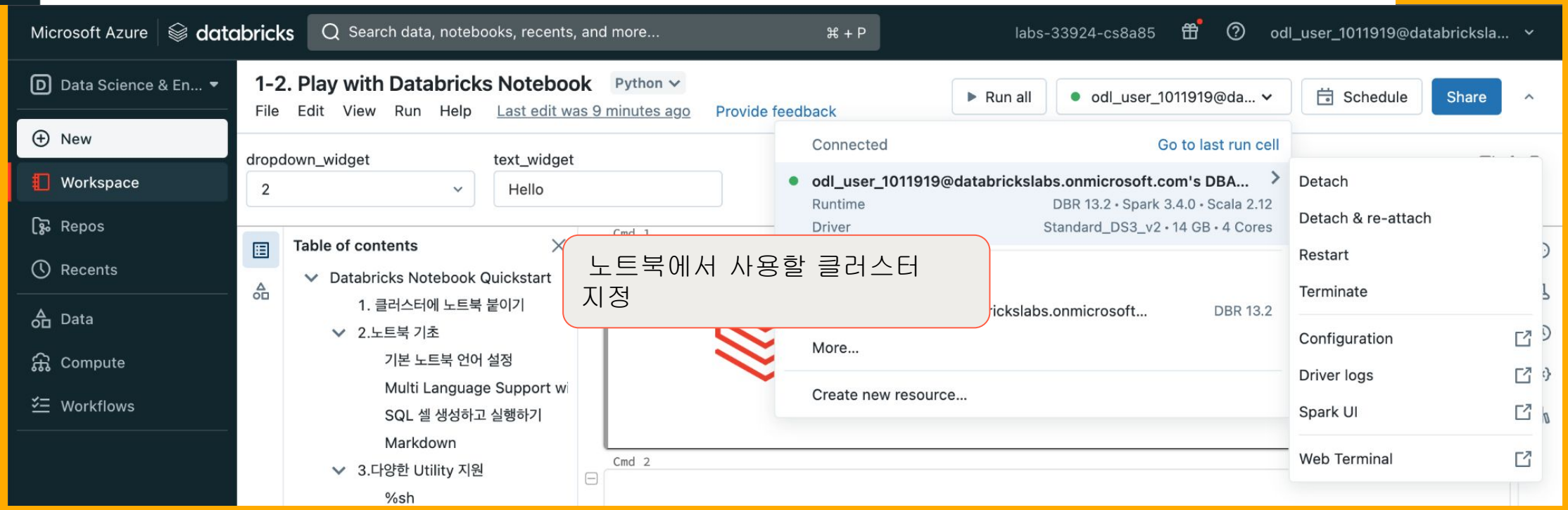
Hands on - DE01

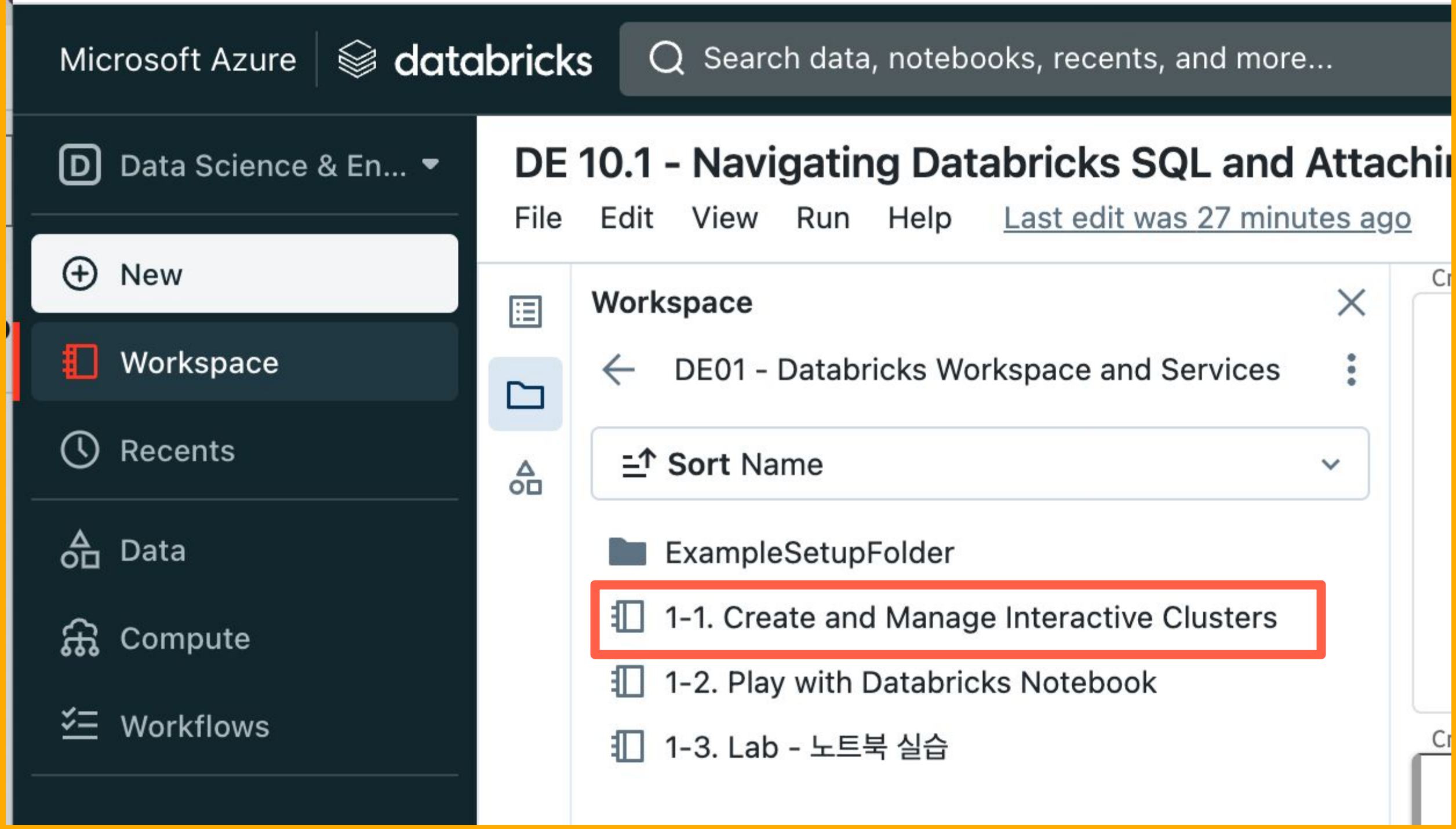
Workspace - 클러스터 생성

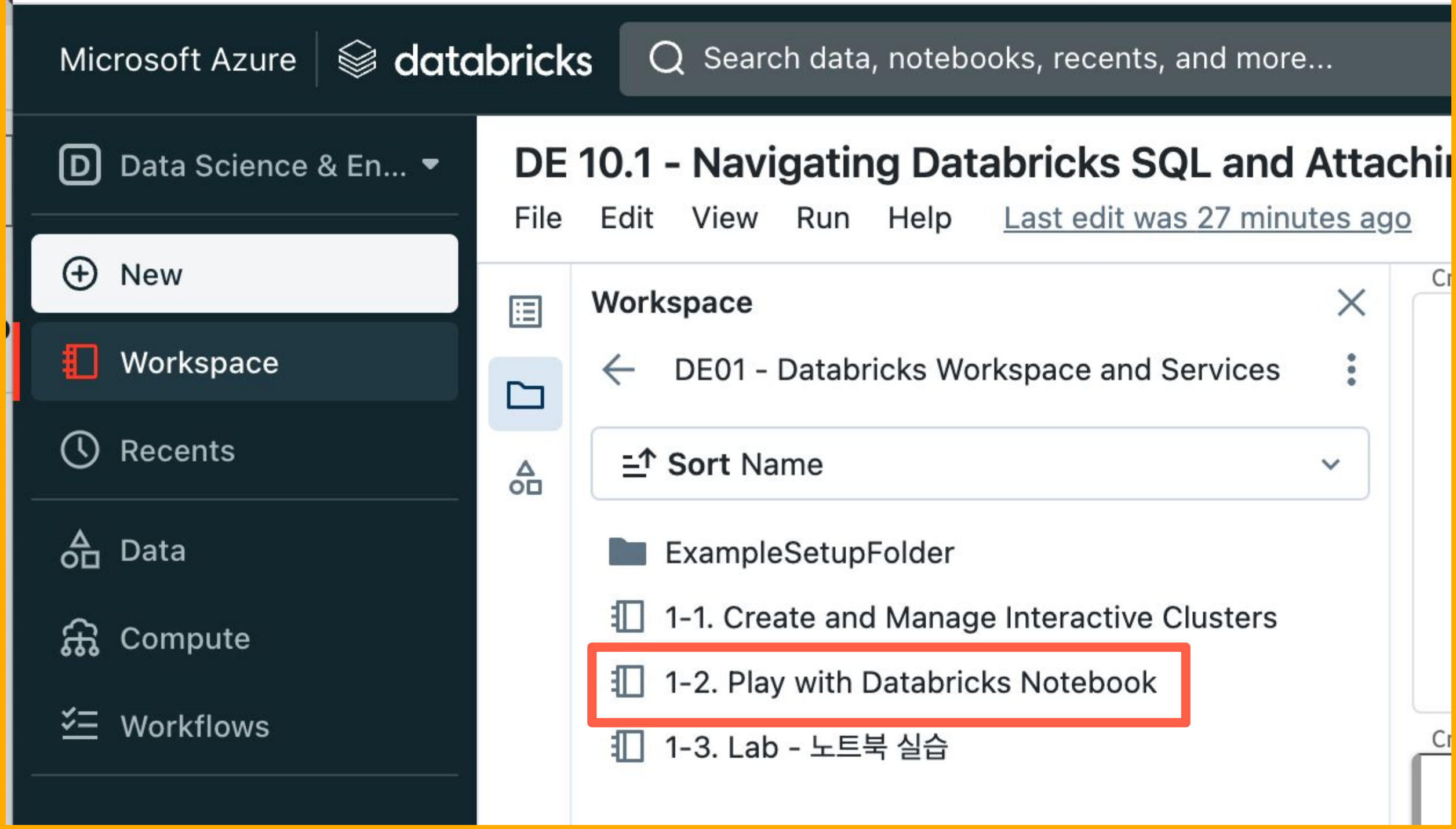









Microsoft Azure
databricks
Search data, notebooks, recents, and more...
Data Science & En...
New
Workspace
Recents
Data
Compute
Workflows
DE 10.1 - Navigating Databricks SQL and Attachi
File Edit View Run Help
Last edit was 27 minutes ago
Workspace
DE01 - Databricks Workspace and Services
Sort Name
ExampleSetupFolder
1-1. Create and Manage Interactive Clusters
1-2. Play with Databricks Notebook
1-3. Lab - 노트북 실습


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
Data Science & En...
New
Workspace
Recents
Data
Compute
Workflows
DE 10.1 - Navigating Databricks SQL and Attachi
File Edit View Run Help
Last edit was 27 minutes ago
Workspace
DE01 - Databricks Workspace and Services
Sort Name
ExampleSetupFolder
1-1. Create and Manage Interactive Clusters
1-2. Play with Databricks Notebook
1-3. Lab - 노트북 실습

What is Delta Lake?

Delta Lake Is...

- Open source (https://delta.io/)
- 표준 데이터 형식을 기반으로 구축
- 클라우드 객체 스토리지에 최적화
- 확장 가능한 메타데이터 처리를 위해 구축

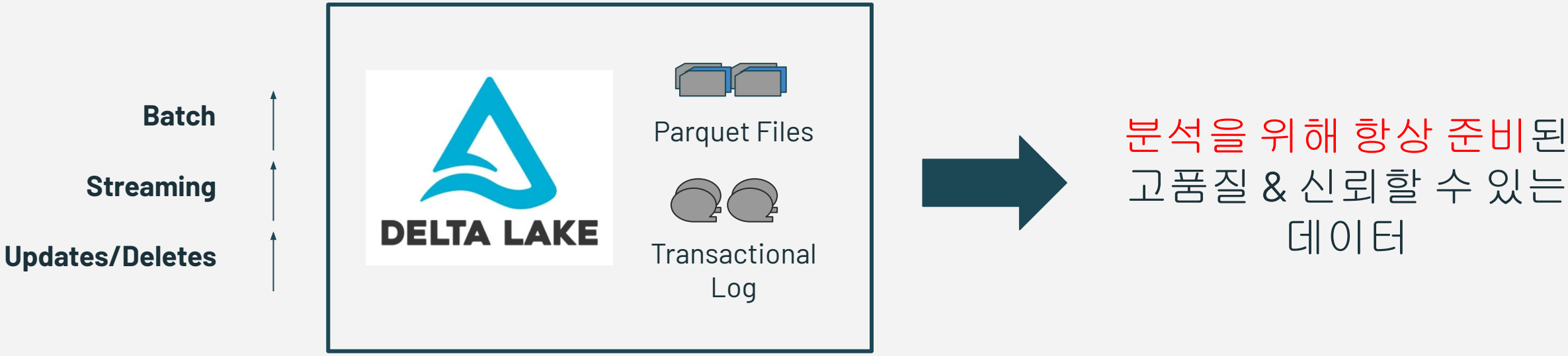
Delta Lake는 오브젝트 스토리지에 ACID를 제공

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

ACID로 해결된 문제

1. 동시에 데이터를 추가하기 어려움
2. 기존 데이터 수정의 어려움
3. 중간에 실패한 작업
4. 실시간 처리 어려움
5. 과거 데이터 버전을 유지하는데 비용 발생

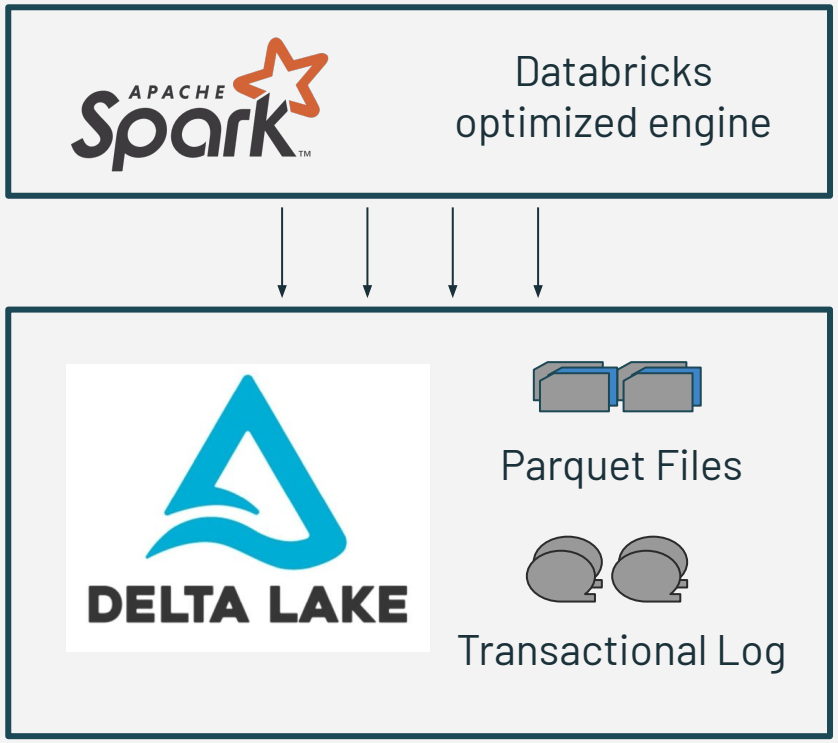
Delta Lake는 데이터 신뢰성을 보장



Key Features

- ACID Transactions
- **Schema Enforcement**
- Unified Batch & Streaming
- **Time Travel/Data Snapshots**

Delta Lake는 성능을 최적화



대규모의 고성능 쿼리

Automatically managed through databricks

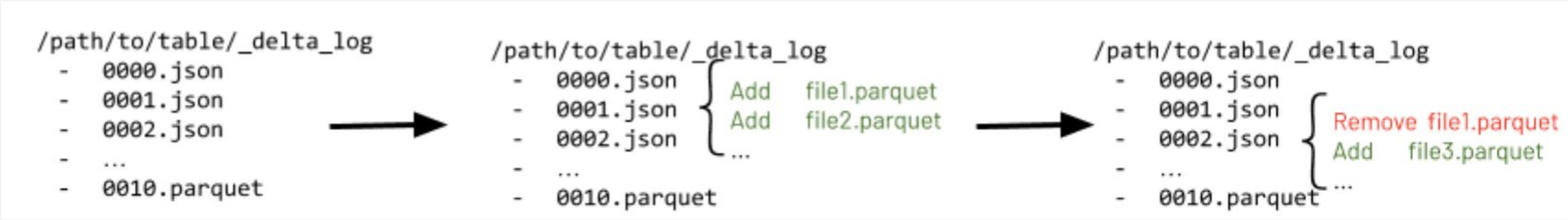
Key Features

- Indexing
- Compaction
- Data skipping
- Caching

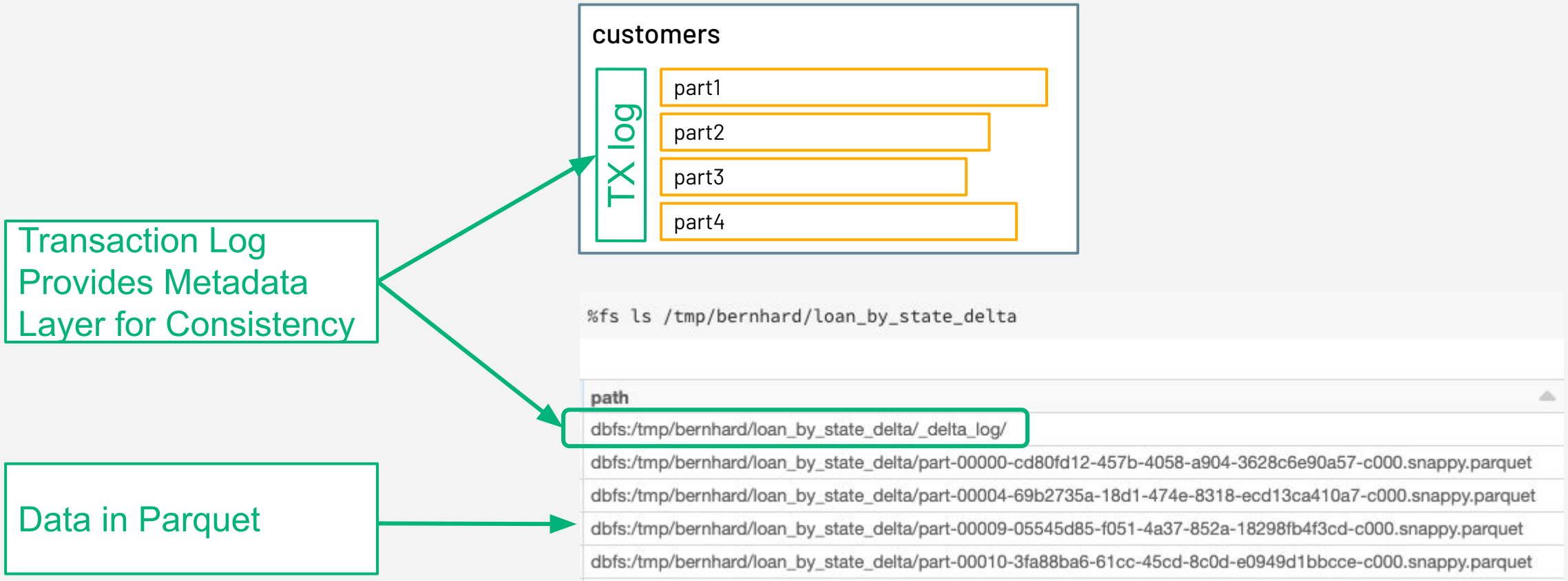
Delta Table 개요

Parquet 파일 기반으로, Transaction log를 지원

- Delta Lake 기술을 사용하여 유지하는 데이터 모음으로 3 가지 항목으로 구성됨
- Delta File(Apache Parquet 파일 사용), Transaction Log, Metastore에 등록된 테이블(optional)
- Delta Table 정의 방법 :
  - CREATE TABLE example_table USING **PARQUET**
  - CREATE TABLE example_table USING **DELTA**



Delta 테이블



https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html

Transaction Log / Metadata

Paquet Checkpoint

JSON Transaction

```
1 %fs ls /mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log
```

	path	name	size
3	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/.s3-optimization-2	.s3-optimization-2	0
4	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/00000000000000000000.checkpoint.parquet	00000000000000000000.checkpoint.parquet	28009
5	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/00000000000000000000.json	00000000000000000000.json	46221
6	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/00000000000000000001.crc	00000000000000000001.crc	94
7	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/00000000000000000001.json	00000000000000000001.json	5522
8	dbfs:/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/_last_checkpoint	_last_checkpoint	24

```
1 dfParquetTransaction = spark.read.parquet('/mnt/datalake/delta_workshop_data/lending_club_accepted_2007-2018/_delta_log/00000000000000000000.checkpoint.parquet')
2 display(dfParquetTransaction)
```

▸ (2) Spark Jobs

▸ dfParquetTransaction: pyspark.sql.dataframe.DataFrame = [txn: struct, add: struct ... 4 more fields]

	txn	add	remove	metaData	protocol	commitInfo
1	null	null	null	null	▸ {"minReaderVersion": 1, "minWriterVersion": 2}	null
	null	null	null	▸ {"id": "2739cf3a-a95b-4a56-9bd3-4dc7b7f19a84", "name": null, "description": null, "format": {"provider": "parquet", "options": {}}, "schemaString": "{\"type\":\"struct\",\"fields\":[{\"name\":\"id\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},{\"name\":\"member_id\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}}, {\"name\":\"loan_amnt\",\"type\":\"double\",\"nullable\":true,\"metadata\":{}}, {\"name\":\"funded_amnt\",\"type\":\"double\",\"nullable\":true,\"metadata\":{}}, {\"name\":\"funded_amnt_inv\",\"type\":\"double\",\"nullable\":true,\"metadata\":{}}, {\"name\":\"term\",\"type\":\"string\",\"nullable\":true,\"metadata\":{}},	null	null

Scalable Metadata

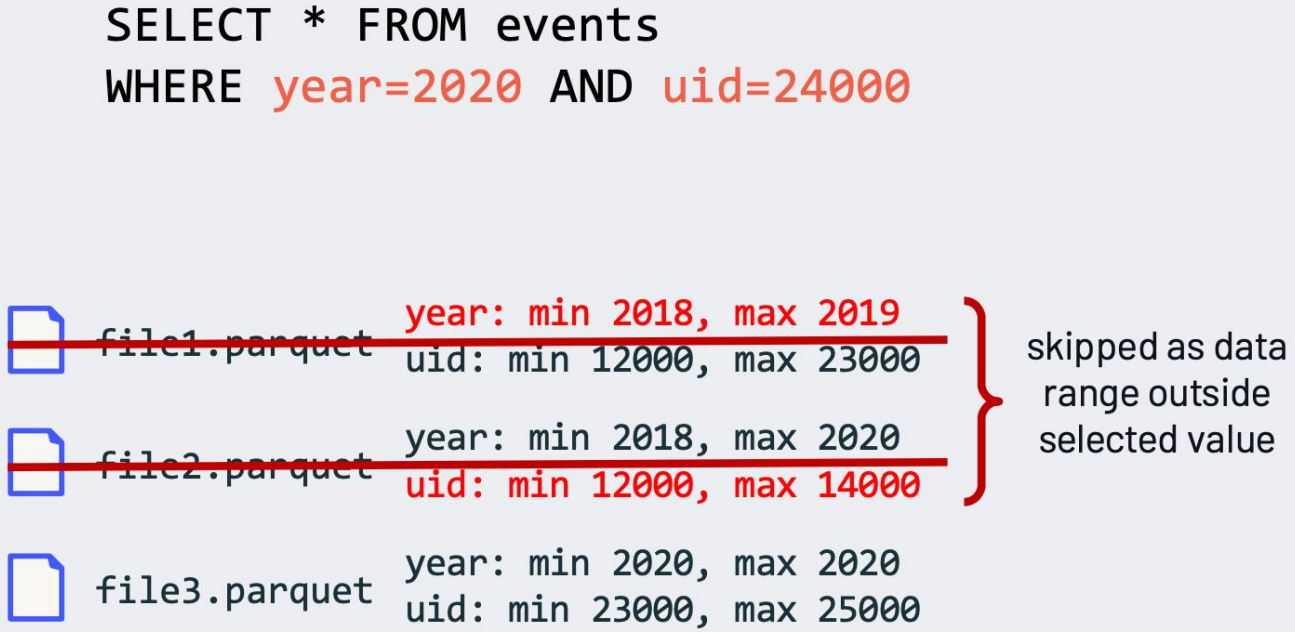
databricks

Column Stats 를 사용한 Data Skipping

File write시 Column의 Min/Max 값이 자동 수집되어 Delta Log에 저장

Min/Max 값을 이용해 쿼리 수행시 읽어야할 파일 제외한 나머지 Skip

File Footer까지 읽어야하는 Parquet Row-Group Filtering 대비 월등한 성능



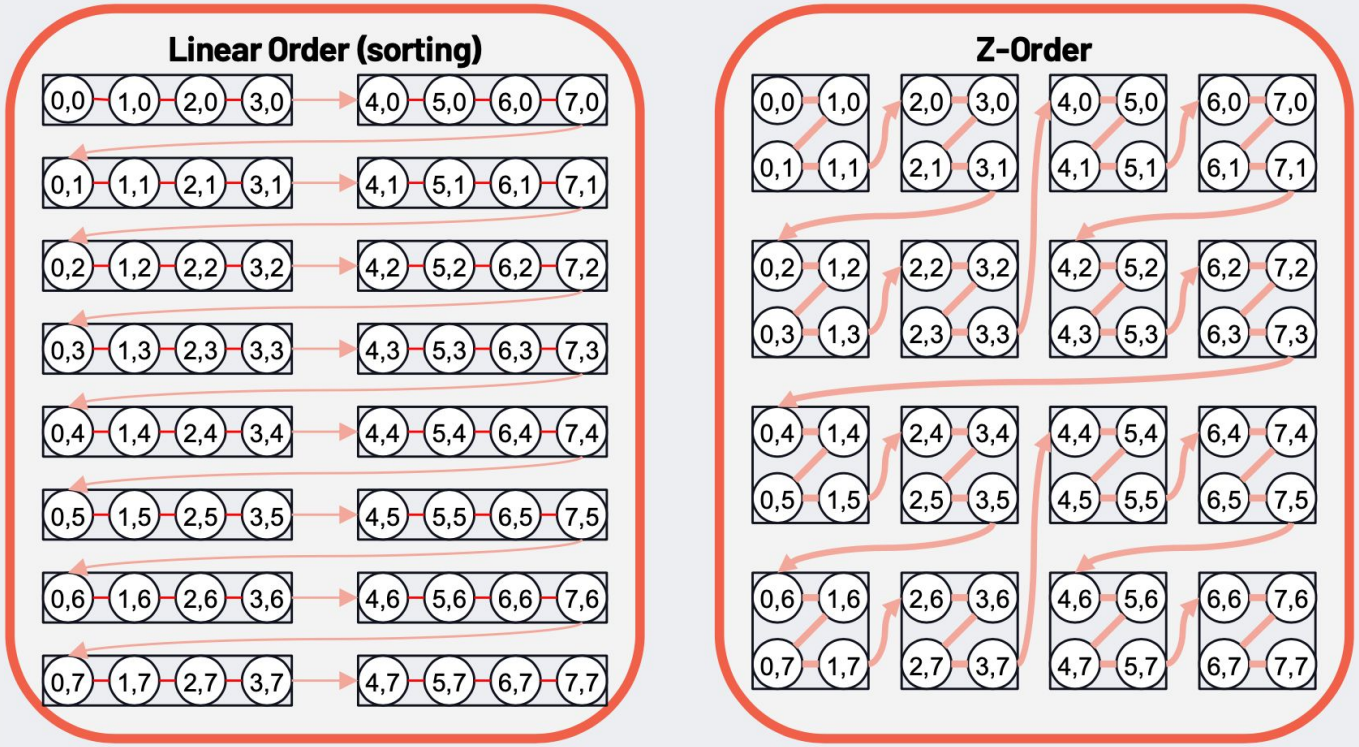
Optimize ZOrder

Maximizing Data Skipping with Data Clustering

Data Skipping 은 파일이 매우 작은 Min/Max 범위에 있을때 효과적

Zorder space filling curve 를 이용한 더 효과적인 Multi-Column Data Clustering

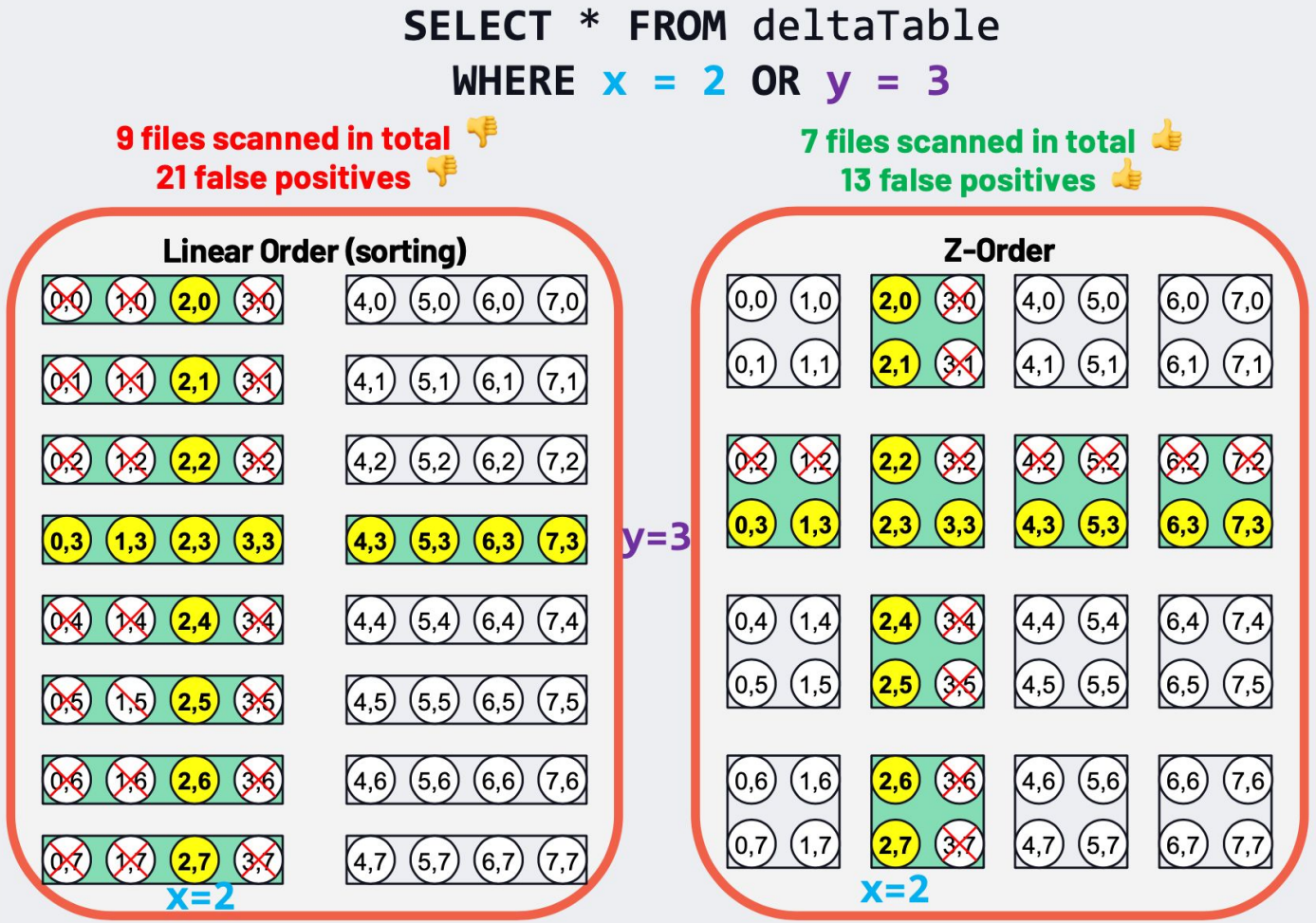
```
OPTIMIZE deltaTable ZORDER BY (x, y)
```



Optimize ZOrder

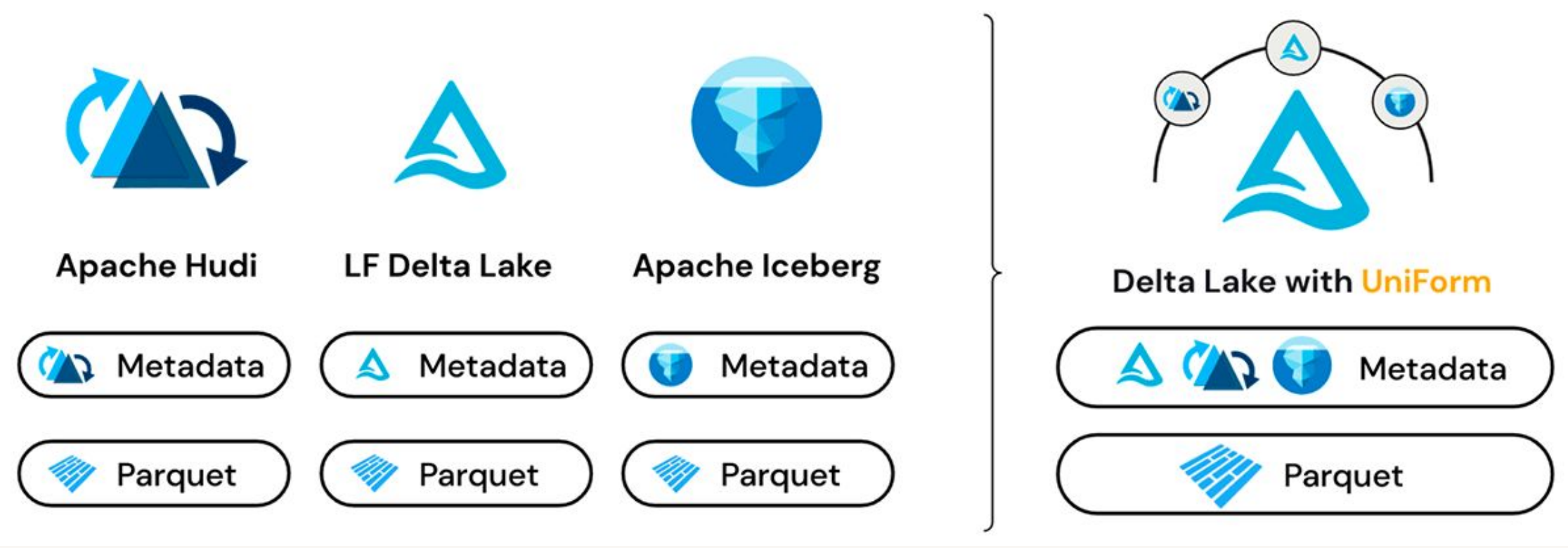
ZOrder 를 이용해서 여러 Column 필터조건이 있는 쿼리에 대한 Data Skipping 최적화

쿼리 패턴에 따라 ZOrder 컬럼 선택



UniForm in Delta 3.0 (public preview)

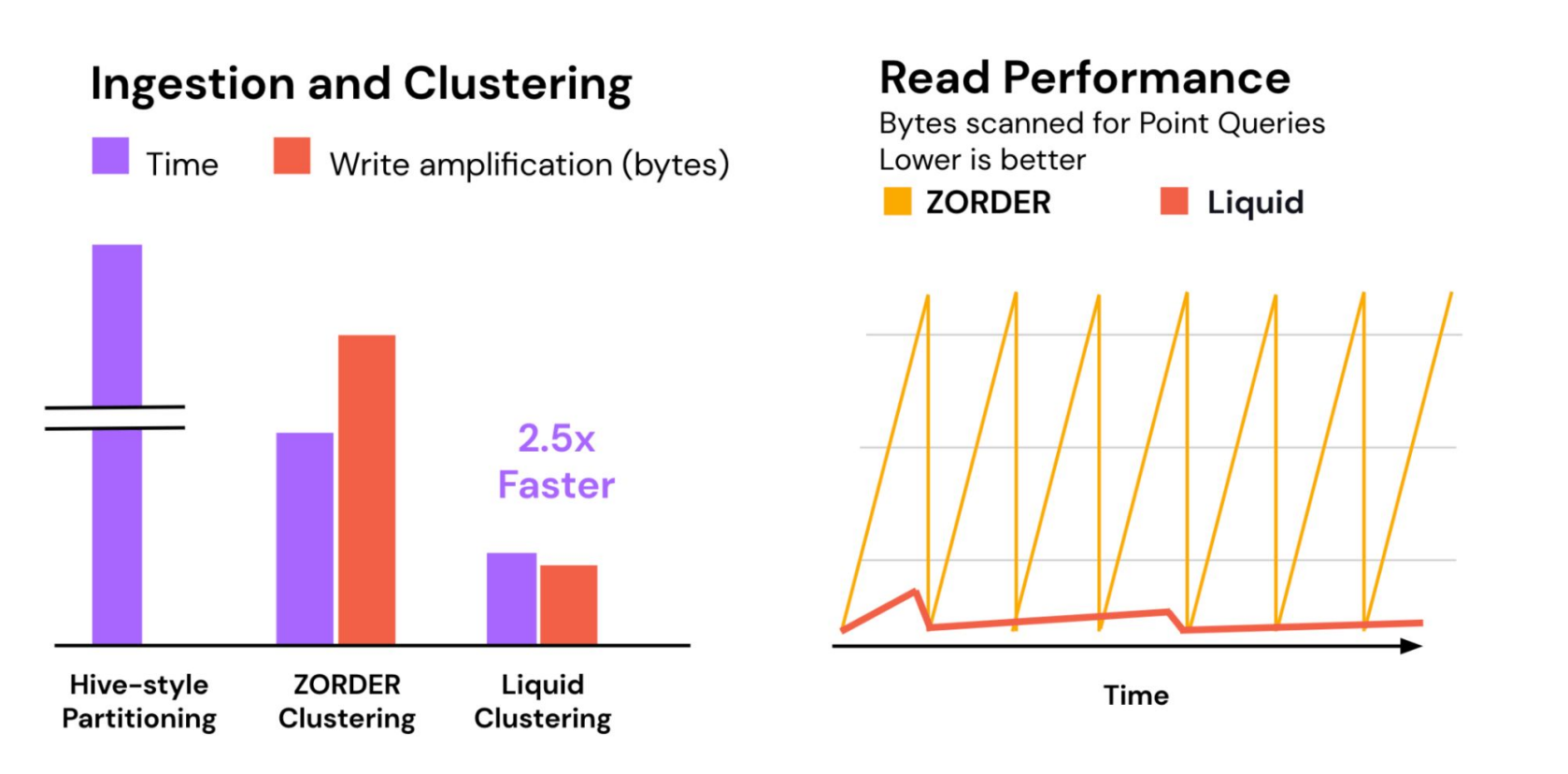
Lakehouse 상호운용성을 위한 범용 포맷



https://www.databricks.com/kr/blog/announcing-delta-lake-30-new-universal-format-and-liquid-clustering

Liquid Clustering in Delta 3.0 (Public Preview)

유연하고 효율적인 클러스터링으로 데이터 레이아웃을 자동 최적화



https://www.databricks.com/kr/blog/announcing-delta-lake-30-new-universal-format-and-liquid-clustering

Delta 테이블 시작하기

Instead of **parquet**...

```
CREATE TABLE ...
USING parquet
...

dataframe
    .write
    .format("parquet")
    .save("/data")
```

... simply say **delta**

```
CREATE TABLE ...
USING delta
…

dataframe
    .write
    .format("delta")
    .save("/data")
```

... simply say **writeStream** for streaming

```
dataframe
    .writeStream
    .format("delta")
    .outputMode("Append")
    .start("/delta/events")
```

기존 parquet 테이블로부터 Delta 테이블 사용하기

Step 1: Convert Parquet to Delta Tables

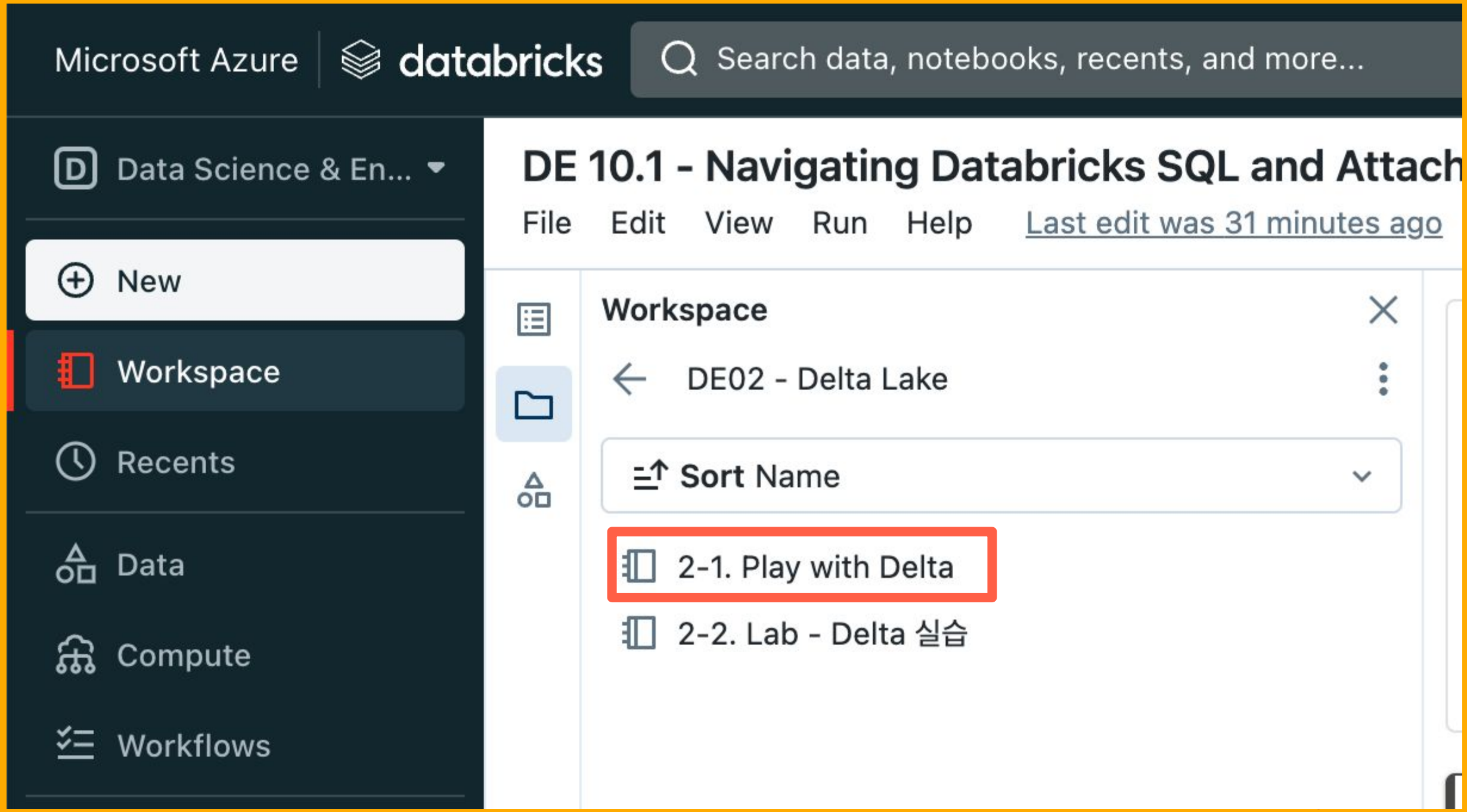
```
CONVERT TO DELTA parquet.`path/to/table` [NO STATISTICS]
[PARTITIONED BY (col_name1 col_type1, col_name2 col_type2, ...)]
```

Step 2: Optimize Layout for Fast Queries

```
OPTIMIZE events
WHERE date >= current_timestamp() - INTERVAL 1 day
ZORDER BY (eventType)
```

Delta Lake 는 Databricks 에서
생성되는 모든 테이블의
기본 포맷 입니다.

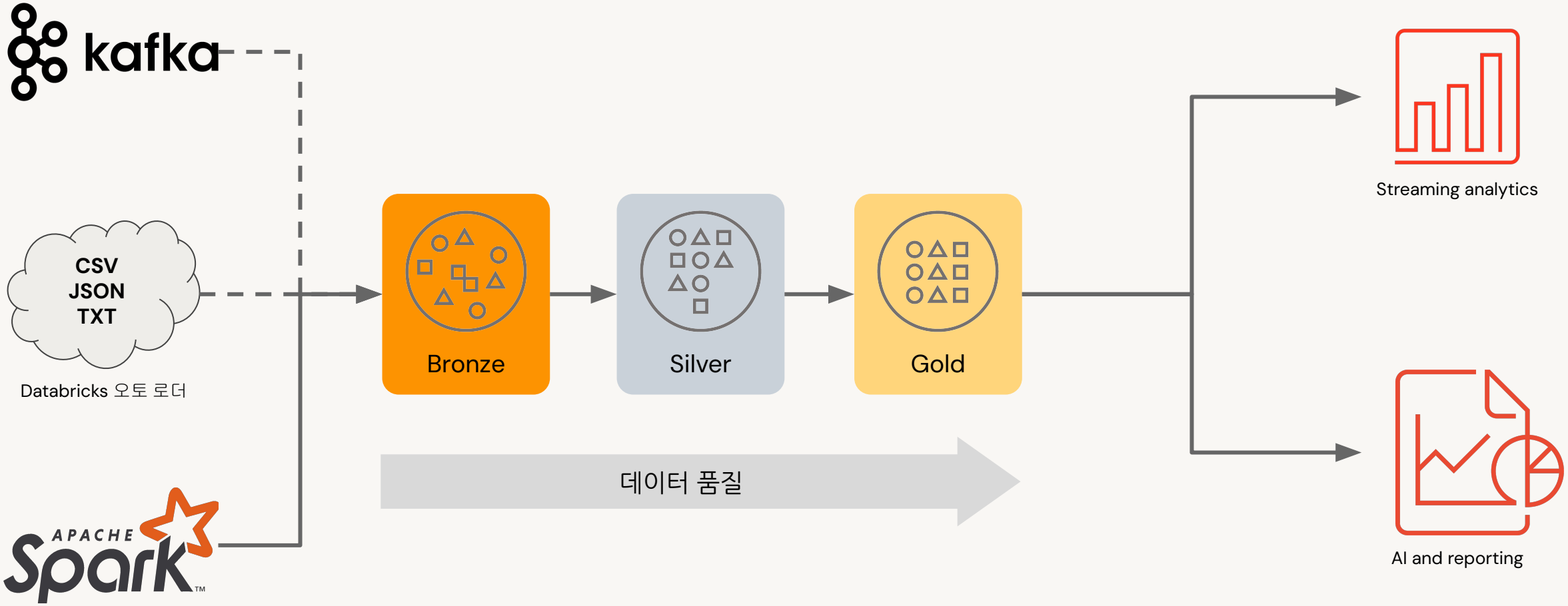
Hands on - DE02
Delta Lake - Delta Table


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
Data Science & En...
New
Workspace
Recents
Data
Compute
Workflows
DE 10.1 - Navigating Databricks SQL and Attach
File Edit View Run Help
Last edit was 31 minutes ago
Workspace
DE02 - Delta Lake
Sort Name
2-1. Play with Delta
2-2. Lab - Delta 실습

Break Time!!!

Multi-hop Architecture

Multi-Hop in the Lakehouse



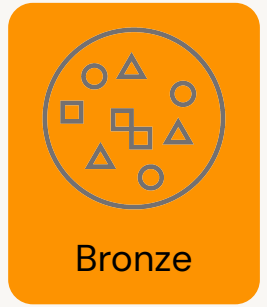
Multi-Hop in the Lakehouse

Bronze Layer

일반적으로 수집된 데이터의 원시 복사본

기존 데이터 레이크 대체

처리되지 않은 전체 데이터 기록의 효율적인 저장 및 쿼리 제공



Multi-Hop in the Lakehouse

Silver Layer

데이터 스토리지 복잡성, 대기 시간 및 중복성을 줄입니다.

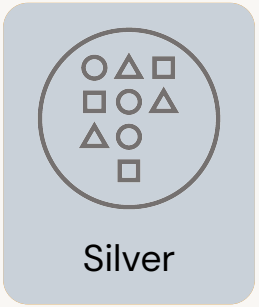
ETL 처리량 및 분석 쿼리 성능을 최적화합니다.

원래 데이터의 값을 보존합니다(집계 전).

중복 레코드 제거

프로덕션 스키마 적용

데이터 품질 검사, 손상된 데이터 검사



Multi-Hop in the Lakehouse

Gold Layer

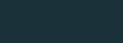
ML 애플리케이션, 리포팅, 대시보드, ad hoc 분석

일반적으로 집계가 포함된 세분화된 데이터

운영 시스템의 부담 감소

비즈니스 크리티컬 데이터에 대한 쿼리 성능 최적화

Workflows

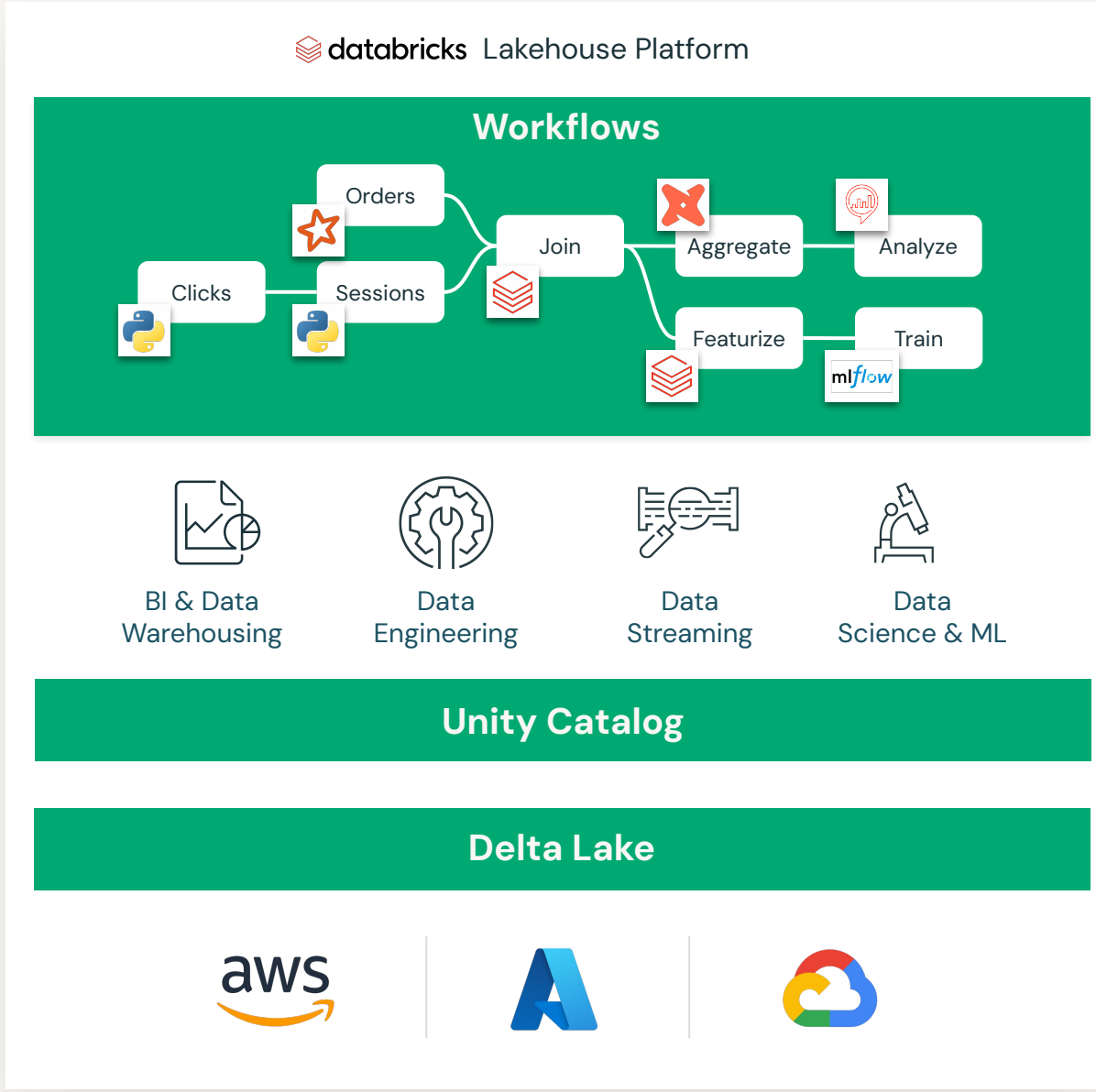


2 종류의 워크플로우

Databricks Workflows는 2개의 주요 태스크 오케스트레이션 서비스 제공

- **Workflow Jobs (Workflows)**: 모든 작업용 워크플로우
- **Delta Live Tables (DLT)**: 델타 레이크용 자동화된 스트리밍 데이터 파이프라인

Note: DLT 파이프라인은 workflow의 Task가 될 수 있음



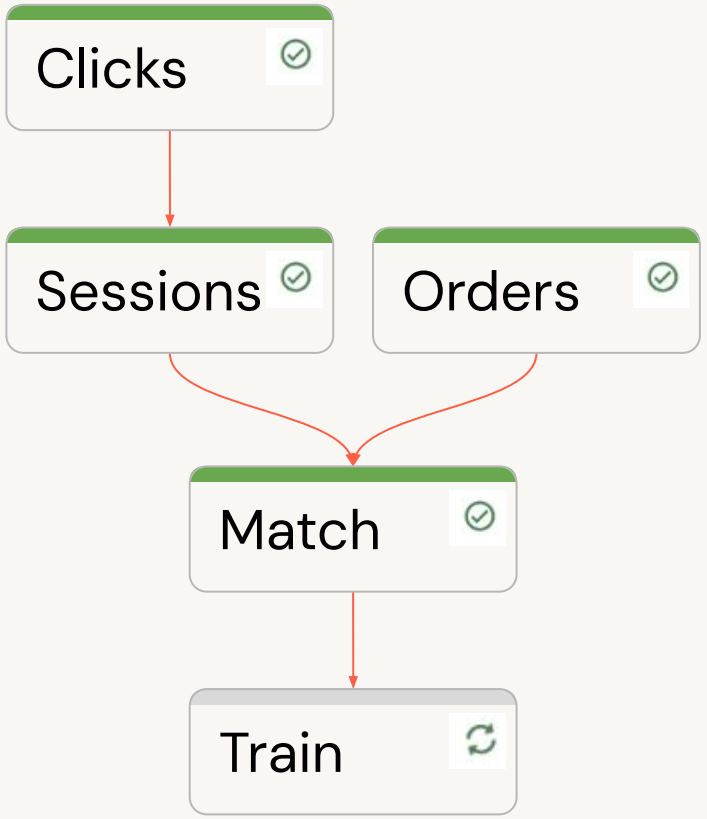
Databricks Workflows

Lakehouse 플랫폼 상에서 데이터, 분석, AI를 통합한 오케스트레이션 가능

- 단순한 관리/실행
- 직관적인 분석 (모든 단계별 상태 파악)
- 높은 신뢰성 (Managed Service)

Workflows

Orchestrate... **...any task...** **...across any platform**



Delta Live Tables

Koalas

Auto Loader

mlflow

Java

Spark

SQL

non-Spark

dbt

+more

Google Cloud

talend

Fivetran

+more

Hands on - DE03 Orchestration - Workflows

New
Workspace
Recents
Catalog
Workflows
Compute

SQL
SQL Editor
Queries
Dashboards
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion
Delta Live Tables

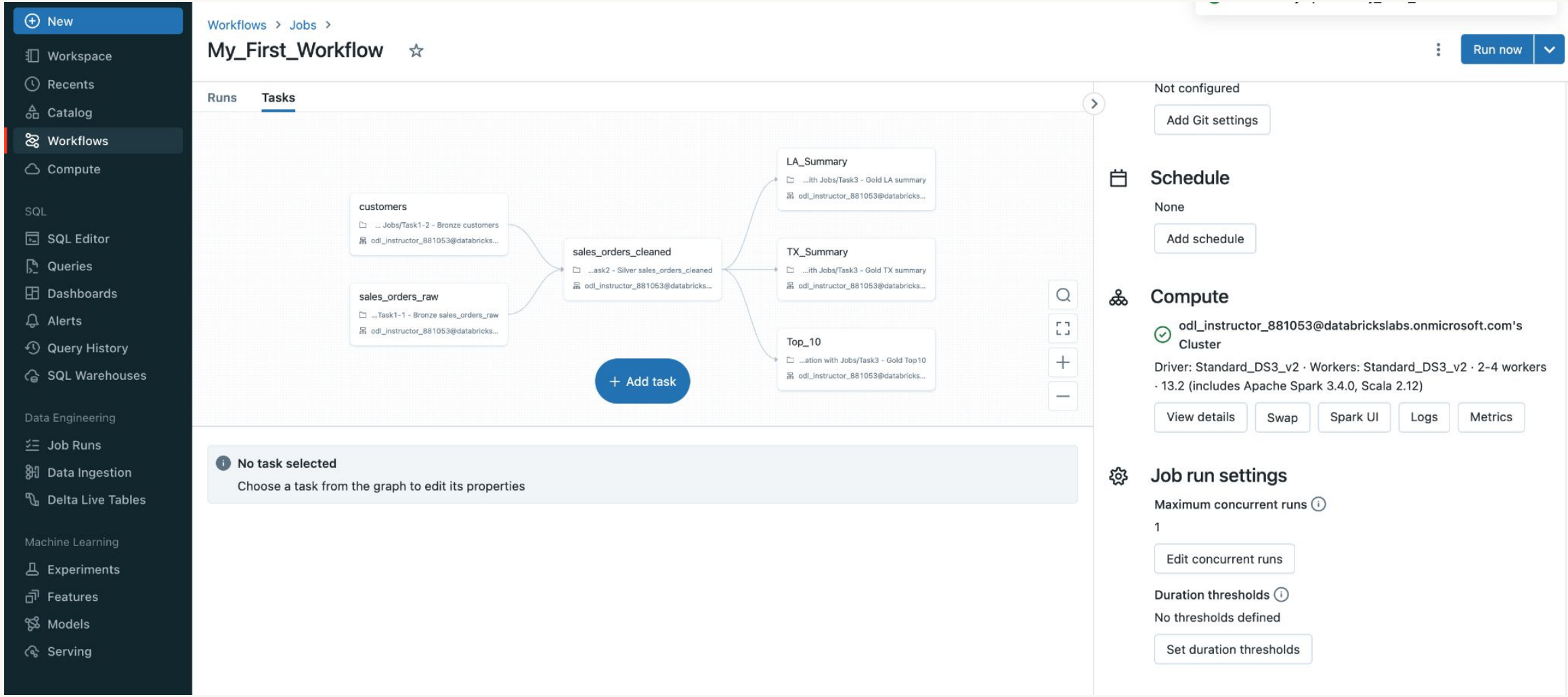
Workspace
Home
Workspace
Shared
Users
Repos
Trash

DE03 - Task Orchestration with Jobs

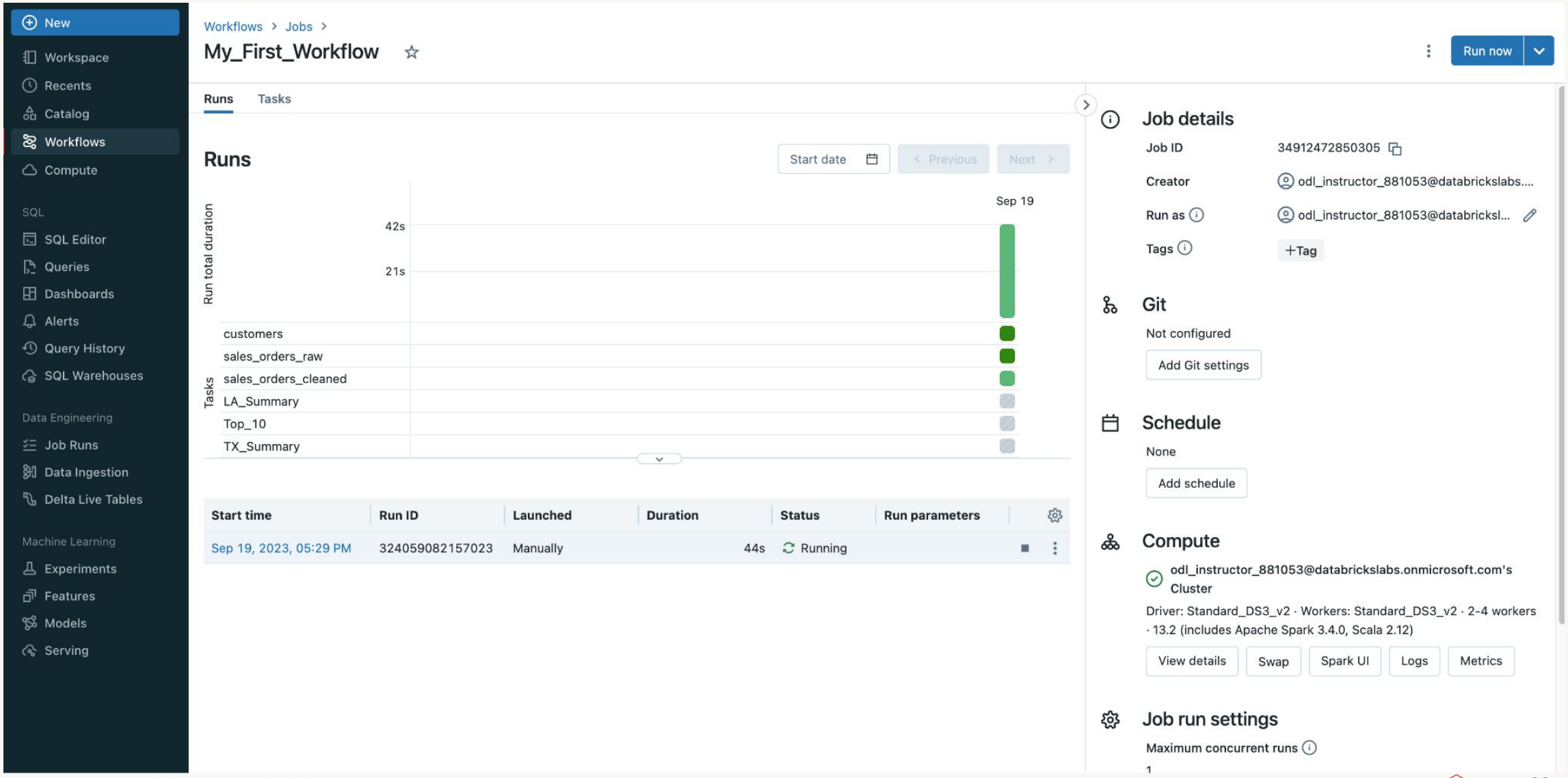
Share | Add

Name	Type	Owner	Created
3.1 - Task Orchestration wit...	Notebook	odl_instructor...	9/18/2023
Task1-1 - Bronze sales_ord...	Notebook	odl_instructor...	9/18/2023
Task1-2 - Bronze customers	Notebook	odl_instructor...	9/18/2023
Task2 - Silver sales_orders...	Notebook	odl_instructor...	9/18/2023
Task3 - Gold LA summary	Notebook	odl_instructor...	9/18/2023
Task3 - Gold Top10	Notebook	odl_instructor...	9/18/2023
Task3 - Gold TX summary	Notebook	odl_instructor...	9/18/2023

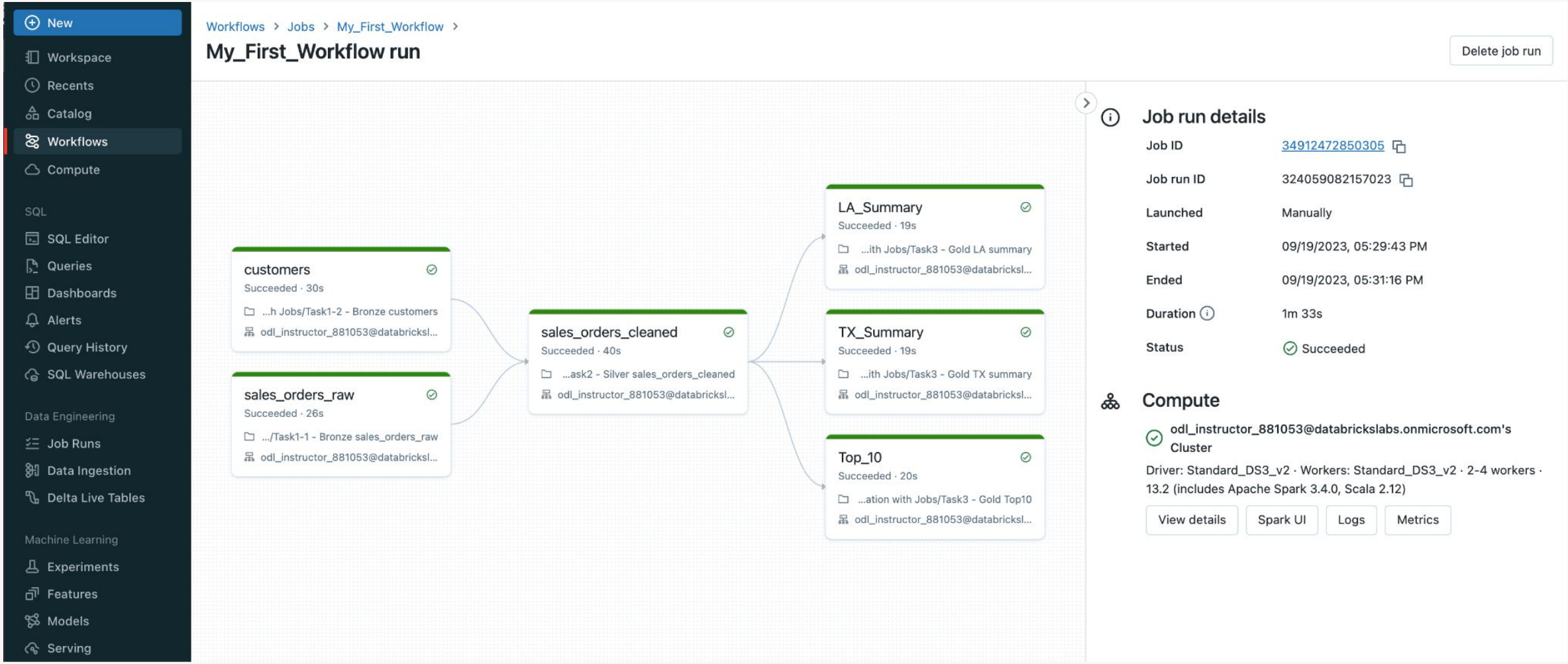
실습 (DE03 Task Orchestration with Job)



실습 (DE03 Task Orchestration with Job)



실습 (DE03 Task Orchestration with Job)



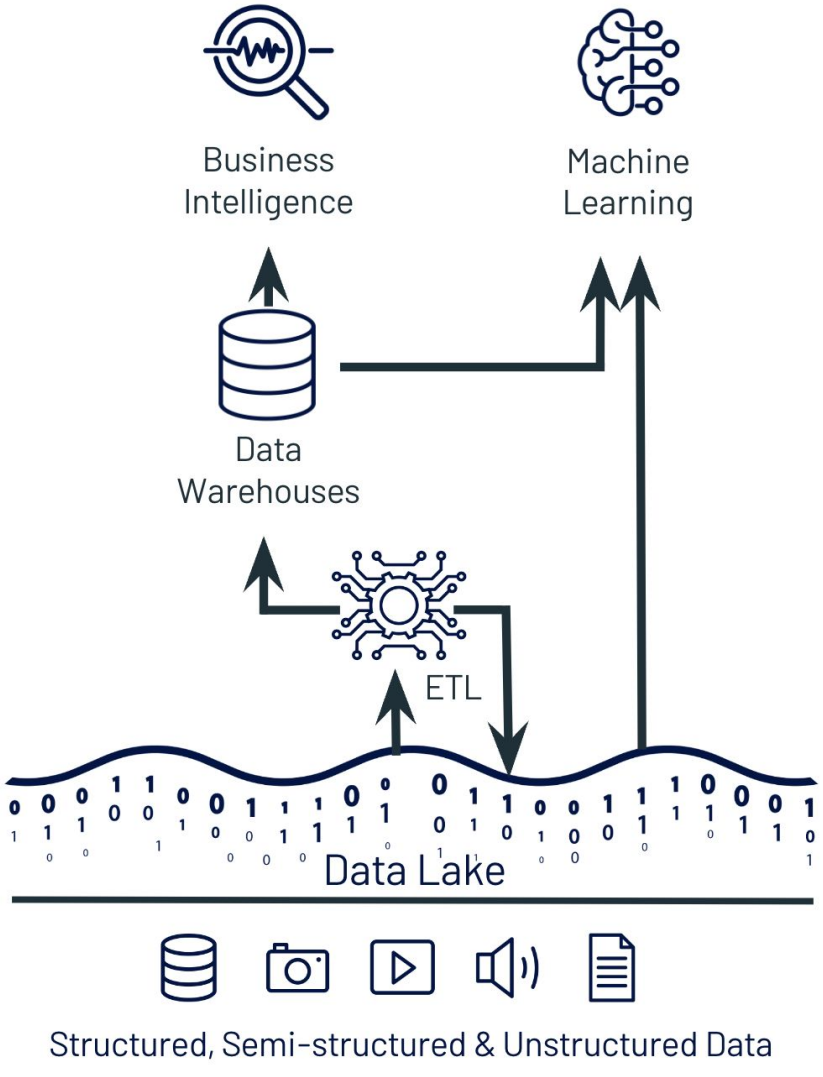
Databricks SQL

최고의 SQL 개발 경험

플랫폼 내에서 공동으로 데이터를 쿼리, 탐색 및 변환

친숙한 **ANSI SQL**을 사용하여
데이터 레이크 데이터를 쿼리하고
기본 제공되는 **SQL 쿼리 편집기**,
경고, **시각화** 및 대화형 **대시보드**를
사용하여 새로운 인사이트를
공동으로 **협업**해서 찾고 공유합니다.

Old Way: 원본은 Data Lake, BI는 Data Warehouse



중복 데이터로 인한 데이터 품질/보안/운영 이슈

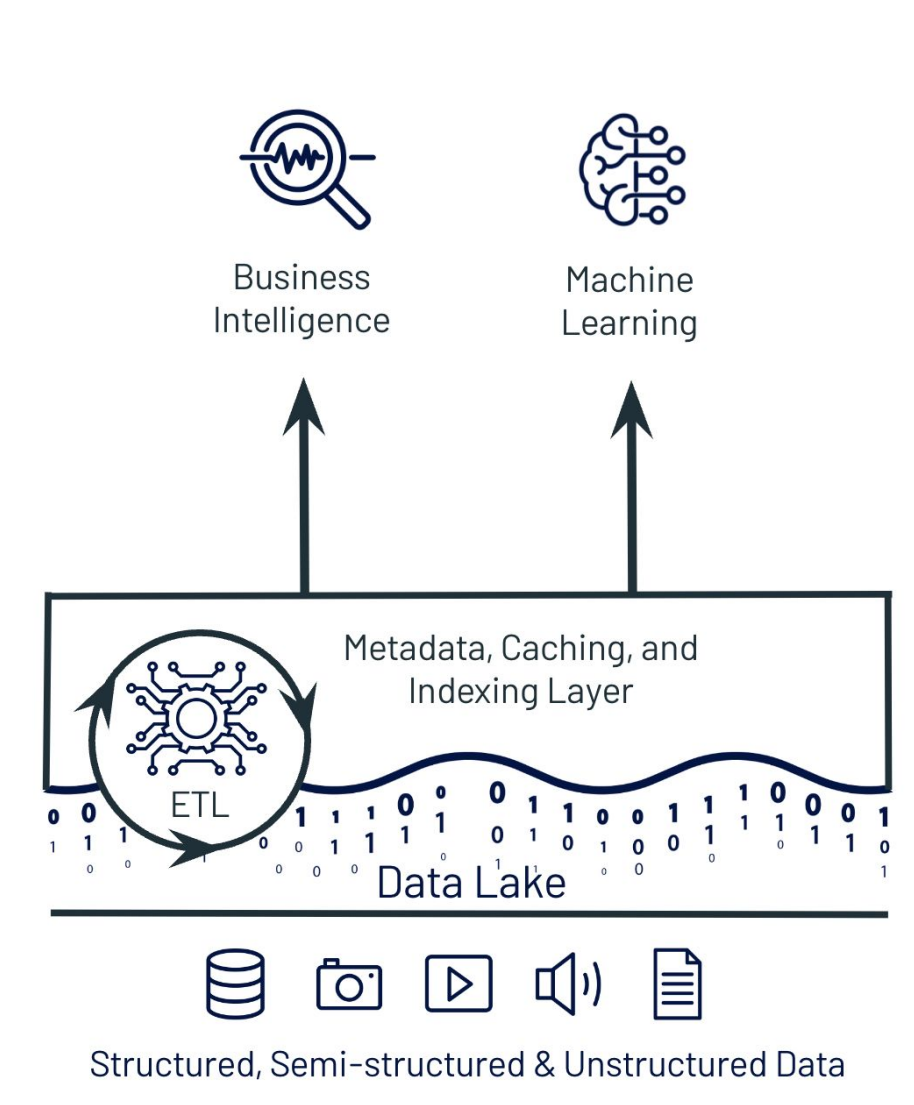
비용 중복 투자

ETL 처리 비용과 소요 시간

벤더 종속적 데이터 포맷

오픈 포맷과 **저비용 Cloud Storage**를 활용하여 상용 DW와 같거나 더 나은 **쿼리 성능**을 제공할 수 있다면?

Lakehouse: Data Lake와 Data Warehouse 통합

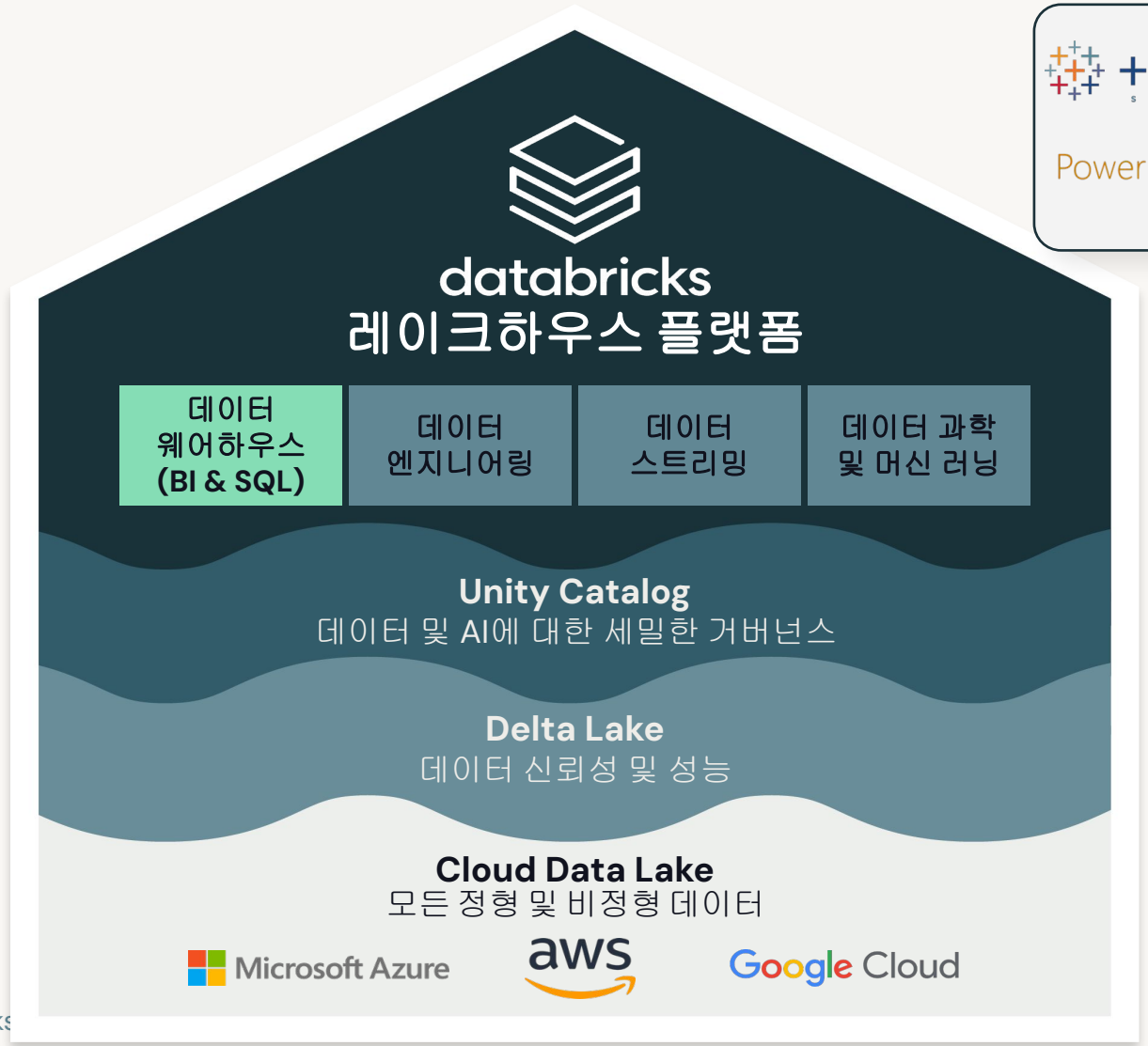


오픈 포맷 데이터 / 오픈 포맷 플랫폼

단일 소스로 데이터 신뢰성 확보

단일 플랫폼에서 BI와 AI 워크로드를 모두 수행

Databricks SQL: Lakehouse에서 BI 와 SQL 분석 수행



Databricks SQL: New World Record (100TB TPC-DS)

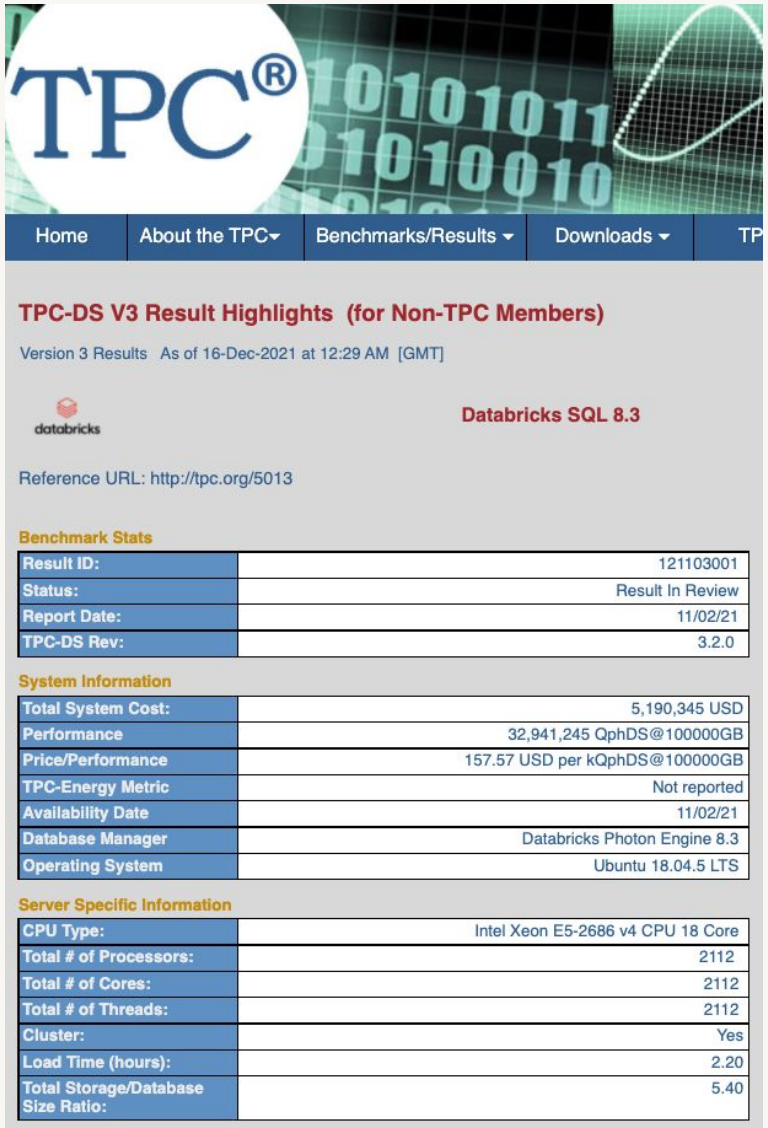
기존 최고 성능 대비 2.2배 높은 결과

Databricks Sets Official Data Warehousing Performance Record

by **Reynold Xin** and **Mostafa Mokhtar**
Posted in **COMPANY BLOG** | **November 2, 2021**

Today, we are proud to announce that **Databricks SQL** has set a **new world record in 100TB TPC-DS**, the gold standard performance benchmark for data warehousing. **Databricks SQL outperformed the previous record by 2.2x**. Unlike most other benchmark news, this result has been formally audited and reviewed by the TPC council.

These results were corroborated by research from **Barcelona Supercomputing Center**, which frequently runs TPC-DS on popular data warehouses. **Their latest research benchmarked Databricks and Snowflake, and found that Databricks was 2.7x faster and 12x better in terms of price performance**. This result validated the thesis that data warehouses such as Snowflake become prohibitively expensive as data size increases in production.



TPC®

Home | About the TPC | Benchmarks/Results | Downloads | TP

TPC-DS V3 Result Highlights (for Non-TPC Members)

Version 3 Results As of 16-Dec-2021 at 12:29 AM [GMT]

databricks

Databricks SQL 8.3

Reference URL: http://tpc.org/5013

Benchmark Stats

Result ID:	121103001
Status:	Result In Review
Report Date:	11/02/21
TPC-DS Rev:	3.2.0

System Information

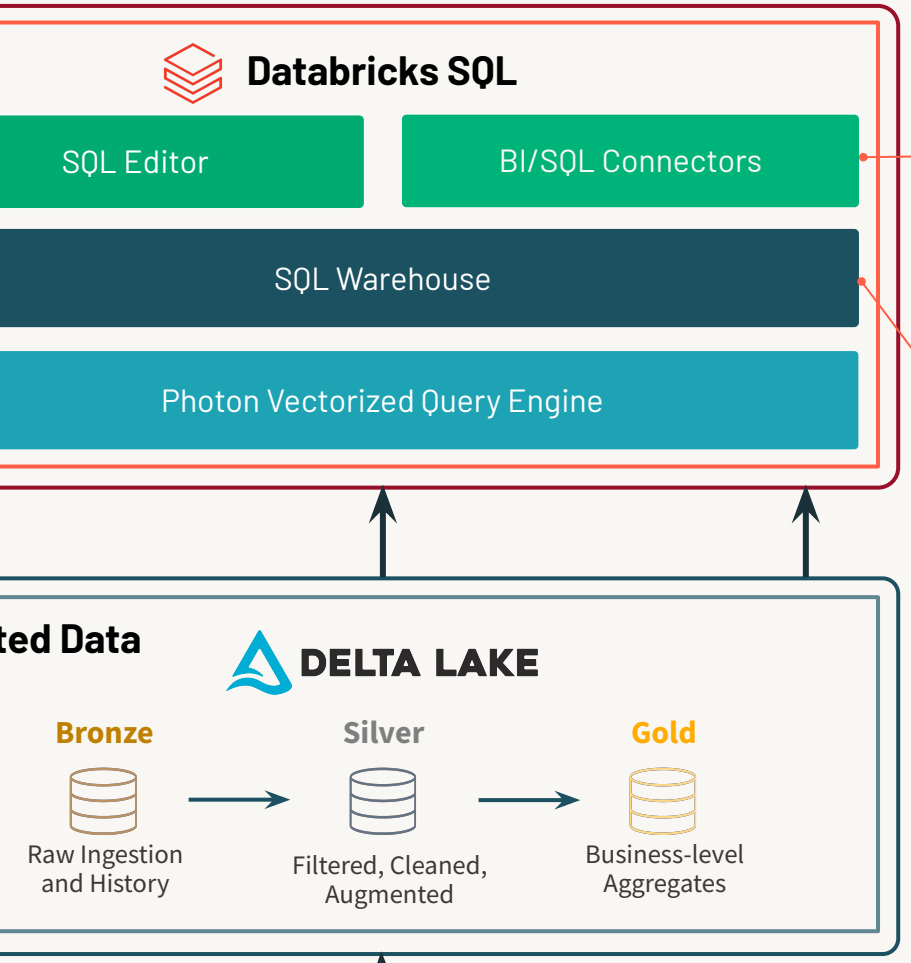
Total System Cost:	5,190,345 USD
Performance	32,941,245 QphDS@100000GB
Price/Performance	157.57 USD per kQphDS@100000GB
TPC-Energy Metric	Not reported
Availability Date	11/02/21
Database Manager	Databricks Photon Engine 8.3
Operating System	Ubuntu 18.04.5 LTS

Server Specific Information

CPU Type:	Intel Xeon E5-2686 v4 CPU 18 Core
Total # of Processors:	2112
Total # of Cores:	2112
Total # of Threads:	2112
Cluster:	Yes
Load Time (hours):	2.20
Total Storage/Database Size Ratio:	5.40

* https://dbricks.co/benchmark

Databricks SQL 구성 요소



SQL 분석 UI
데이터 탐색, SQL 쿼리, 시각화, 대시보드 공유, 자동 경고 설정 등
심플하고 친숙한 분석 환경 제공

Photon 쿼리 엔진
C++로 작성된 고성능 vectorized 실행 엔진

Delta Lake
캐싱, data skipping, 인덱싱 등을 통해 높은 처리 성능과 신뢰성 제공

최적화된 ODBC/JDBC 드라이버
대기시간과 오버헤드 개선, 데이터 전송 속도 향상, 메타데이터 검색 성능 향상

향상된 워크로드 관리, Async 병렬 IO, 자동 스케일링으로 동시처리 성능 향상
짧은 쿼리와 긴 쿼리 모두 빠르고 예측가능한 성능 제공
Spot 인스턴스/Serverless 클러스터로 비용 절감

친숙하고 편리한 SQL 분석 환경

데이터 및 스키마 탐색

편리한 SQL 쿼리 에디터

다양한 시각화 및 대시보드

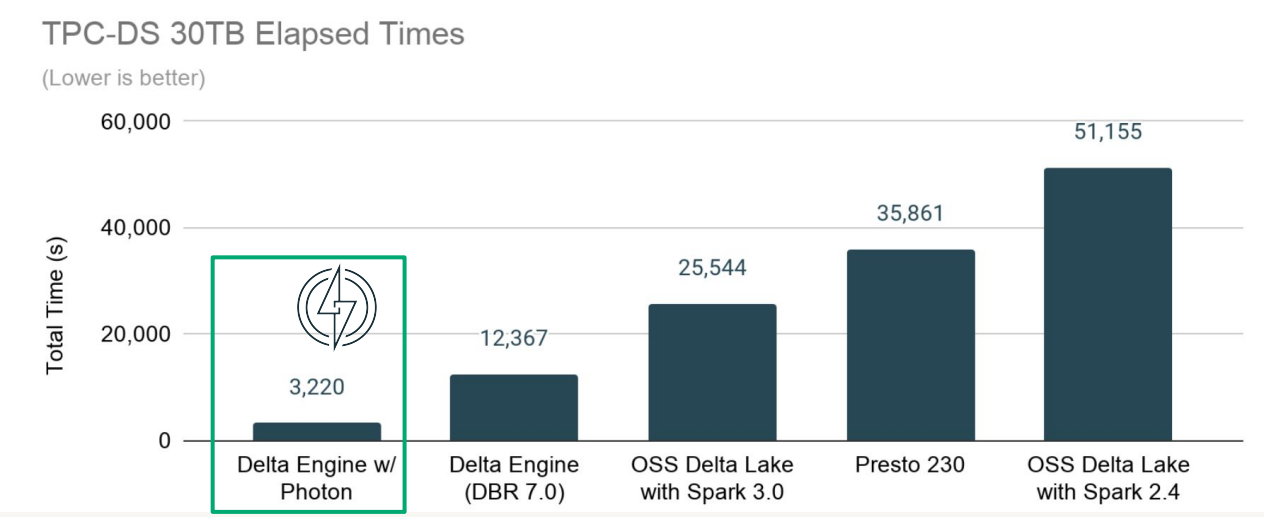
스케줄링과 알람 설정

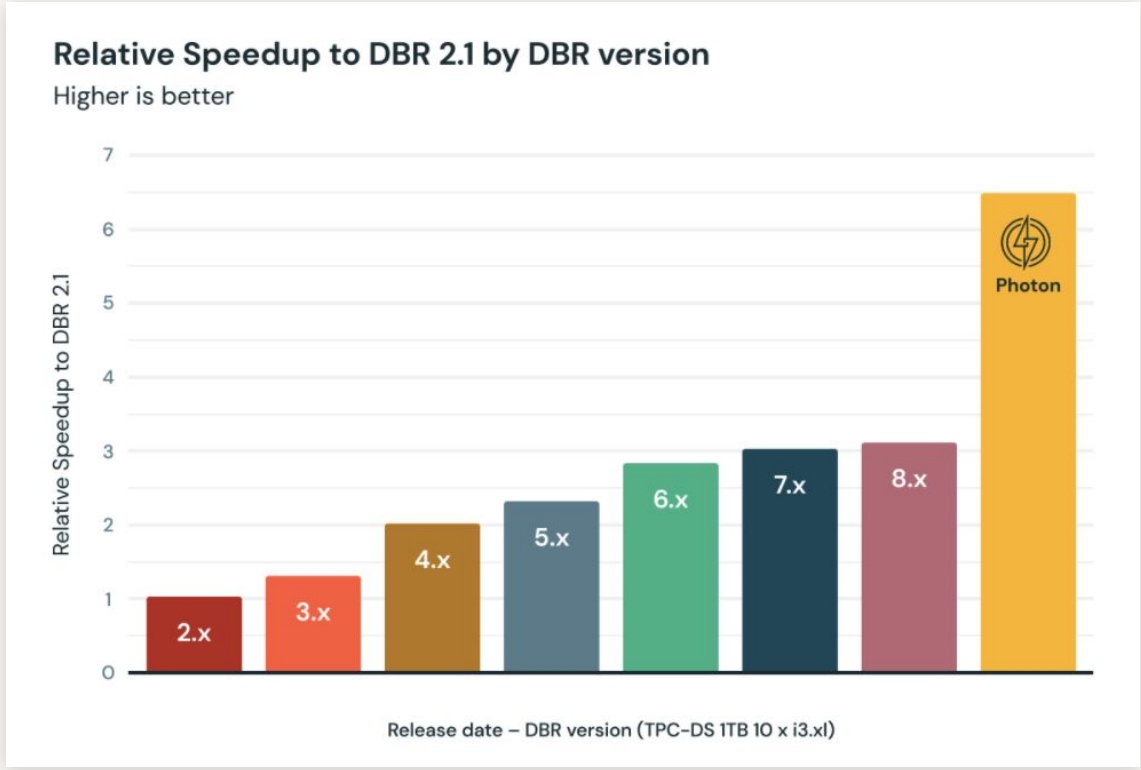
Photon – 차세대 고성능 쿼리 엔진

C++ 로 작성, 최신 CPU의 병렬처리 아키텍처를 활용하는 고성능 Vectorized 쿼리 엔진

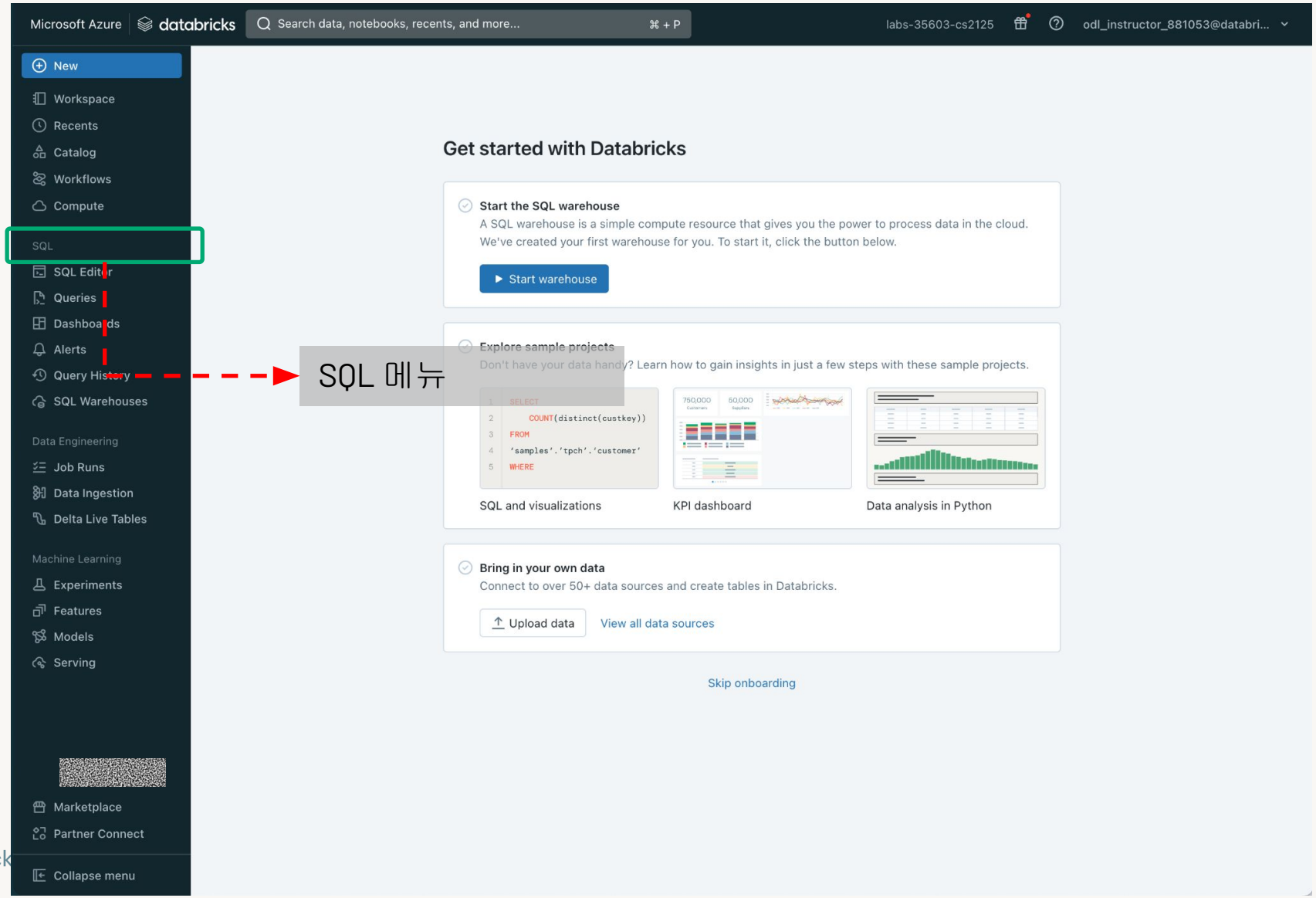
Apache Spark와 100% 호환

Only on Databricks



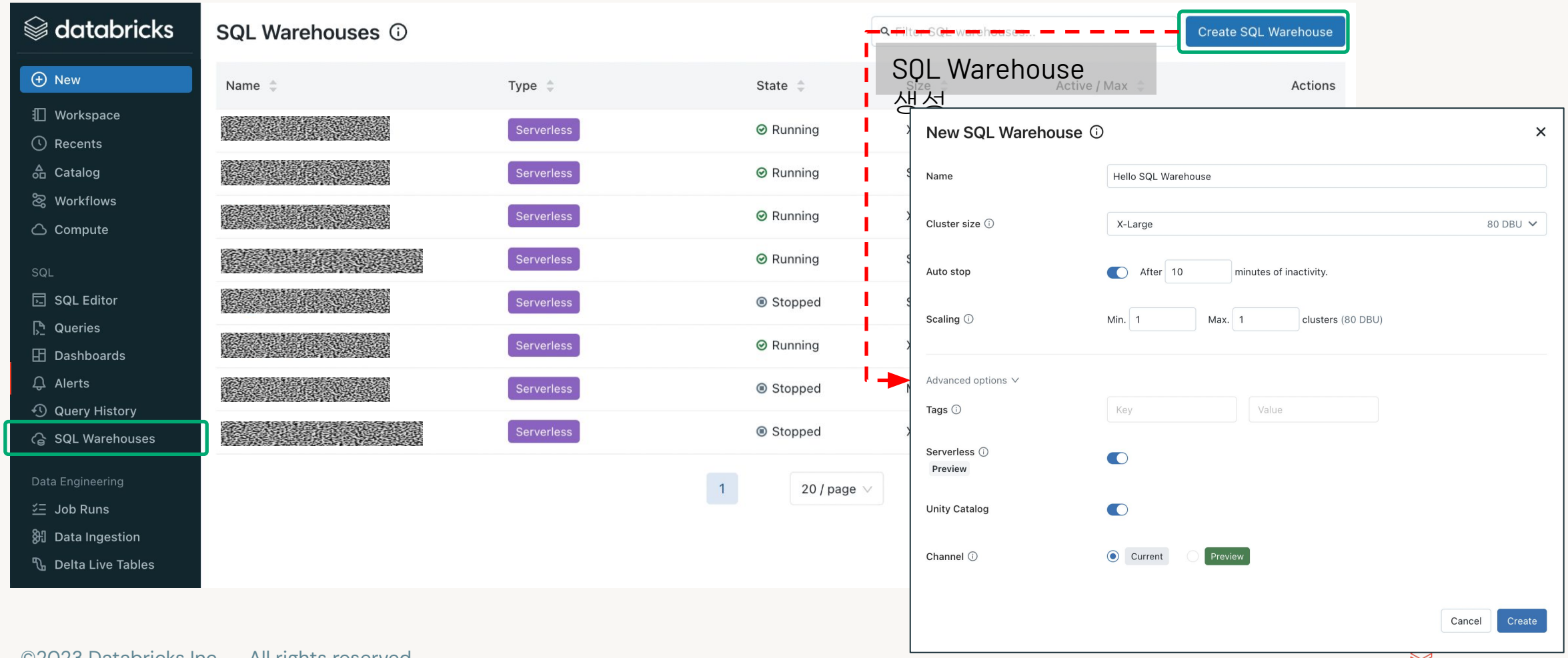


Databricks SQL 환경 설명



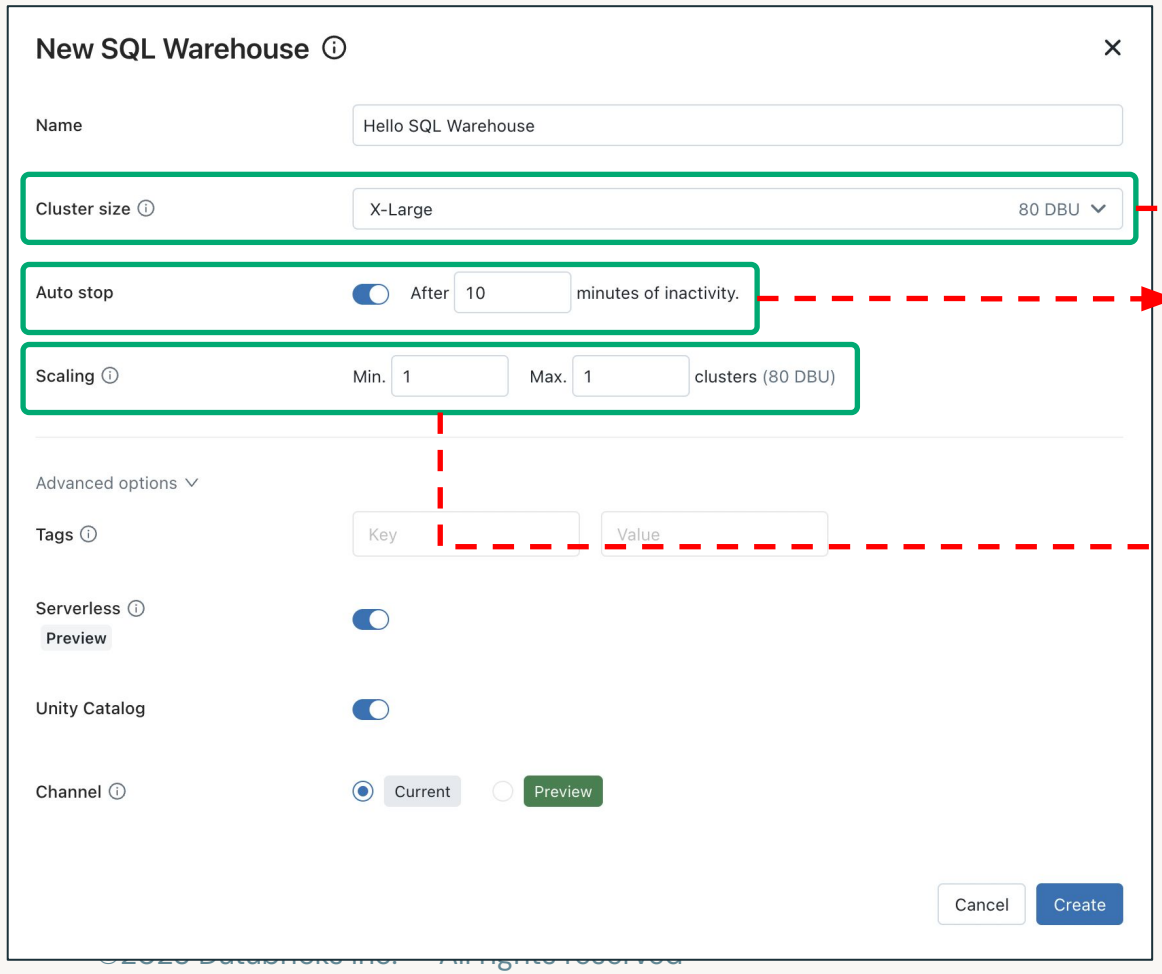
Databricks SQL 환경 설명

SQL Warehouse 생성 (또는 기존 클러스터 재구동)

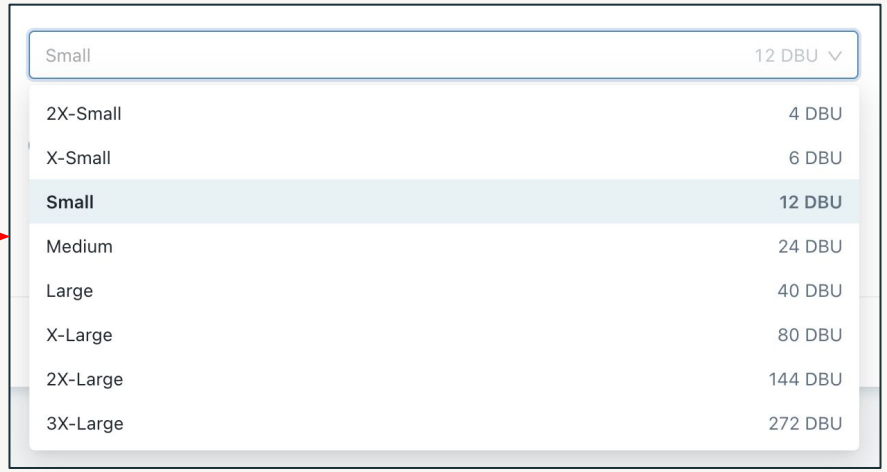


Databricks SQL 환경 설명

SQL Warehouse 생성 – 컴퓨팅 리소스 설정

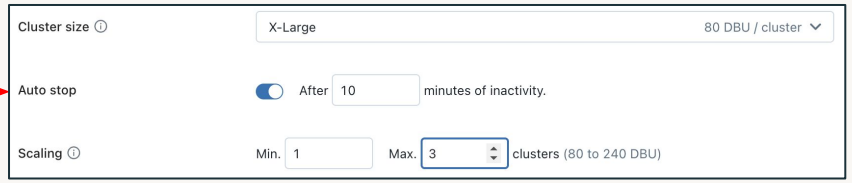


클러스터 크기 선택



지정한 시간 동안 수행되는 쿼리가 없다면 SQL Warehouse 중지

Auto scaling 설정



Databricks SQL 환경 설명

2. SQL Warehouse 생성 - 클러스터 사이즈 설정

인스턴스 크기	vCPU	Memory(GiB)
I3.2xlarge	8	61
Standard_E8ds_v4	8	64

Microsoft Azure

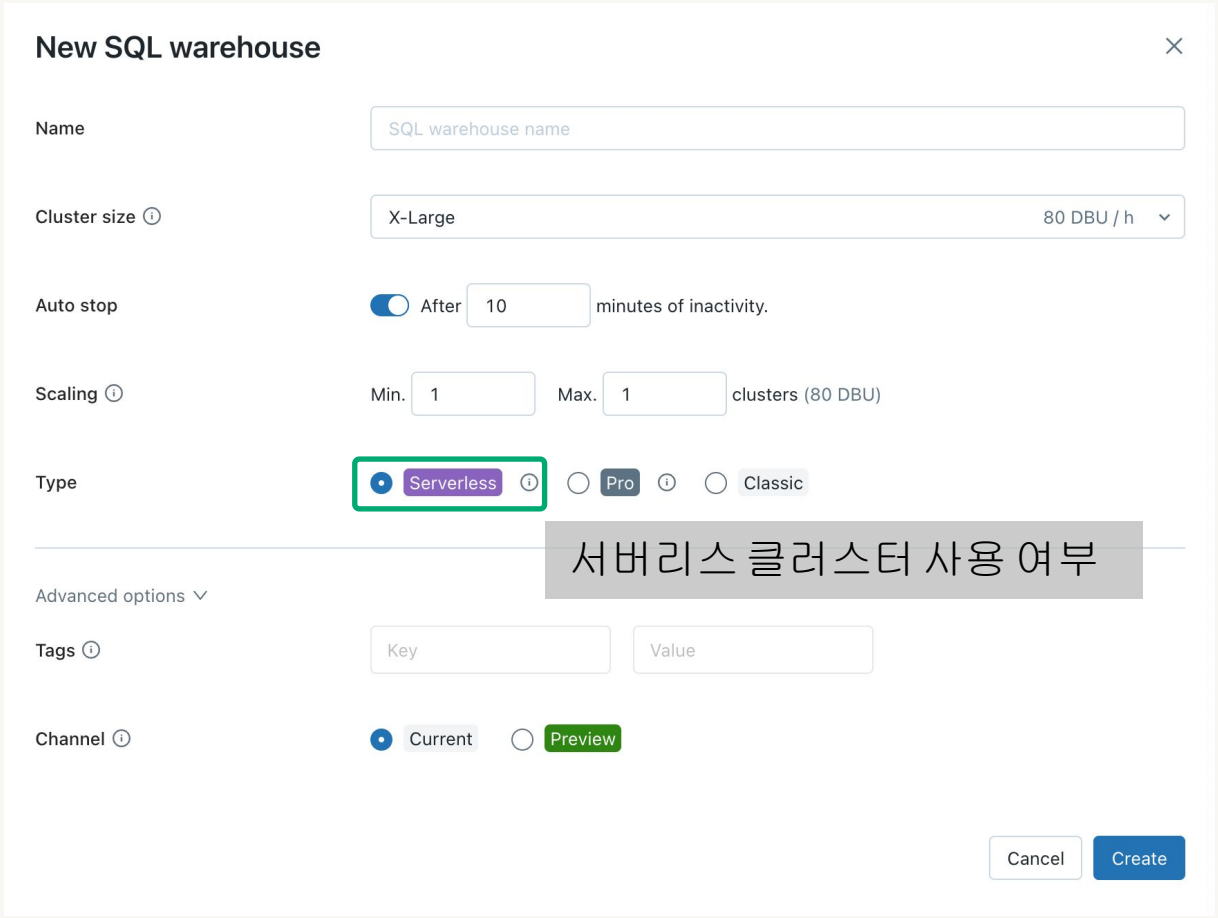
클러스터 사이즈	드라이버 크기	워커 노드 수 **(Standard_E8ds_v4)**
2X-Small	Standard_E8ds_v4	1
X-Small	Standard_E8ds_v4	2
Small	Standard_E16ds_v4	4
Medium	Standard_E32ds_v4	8
Large	Standard_E32ds_v4	16
X-Large	Standard_E64ds_v4	32
2X-Large	Standard_E64ds_v4	64
3X-Large	Standard_E64ds_v4	128
4X-Large	Standard_E64ds_v4	256

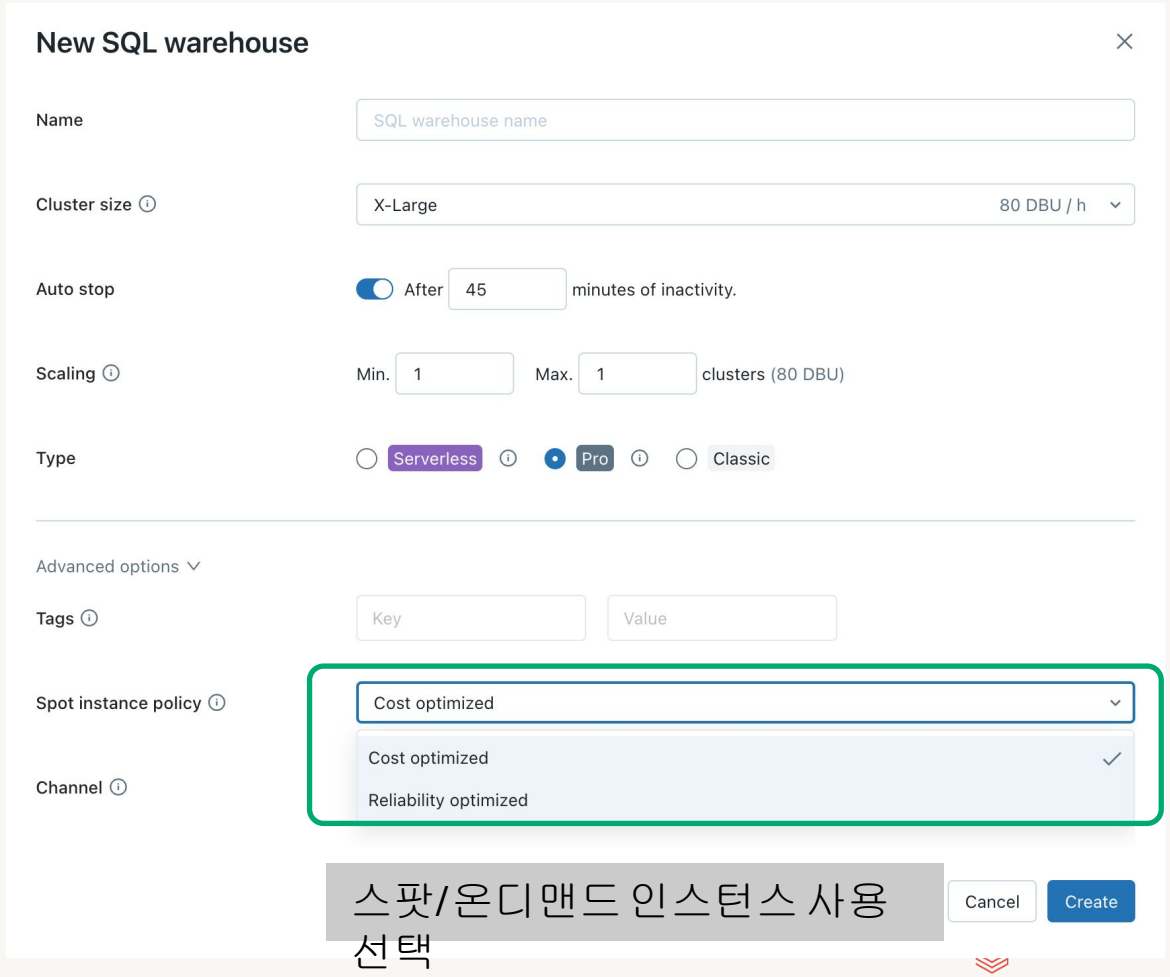
aws

클러스터 사이즈	드라이버 크기	워커 노드 수 **(i3.2xlarge)**
2X-Small	i3.2xlarge	1
X-Small	i3.2xlarge	2
Small	i3.4xlarge	4
Medium	i3.8xlarge	8
Large	i3.8xlarge	16
X-Large	i3.16xlarge	32
2X-Large	i3.16xlarge	64
3X-Large	i3.16xlarge	128
4X-Large	i3.16xlarge	256

Databricks SQL 환경 설명

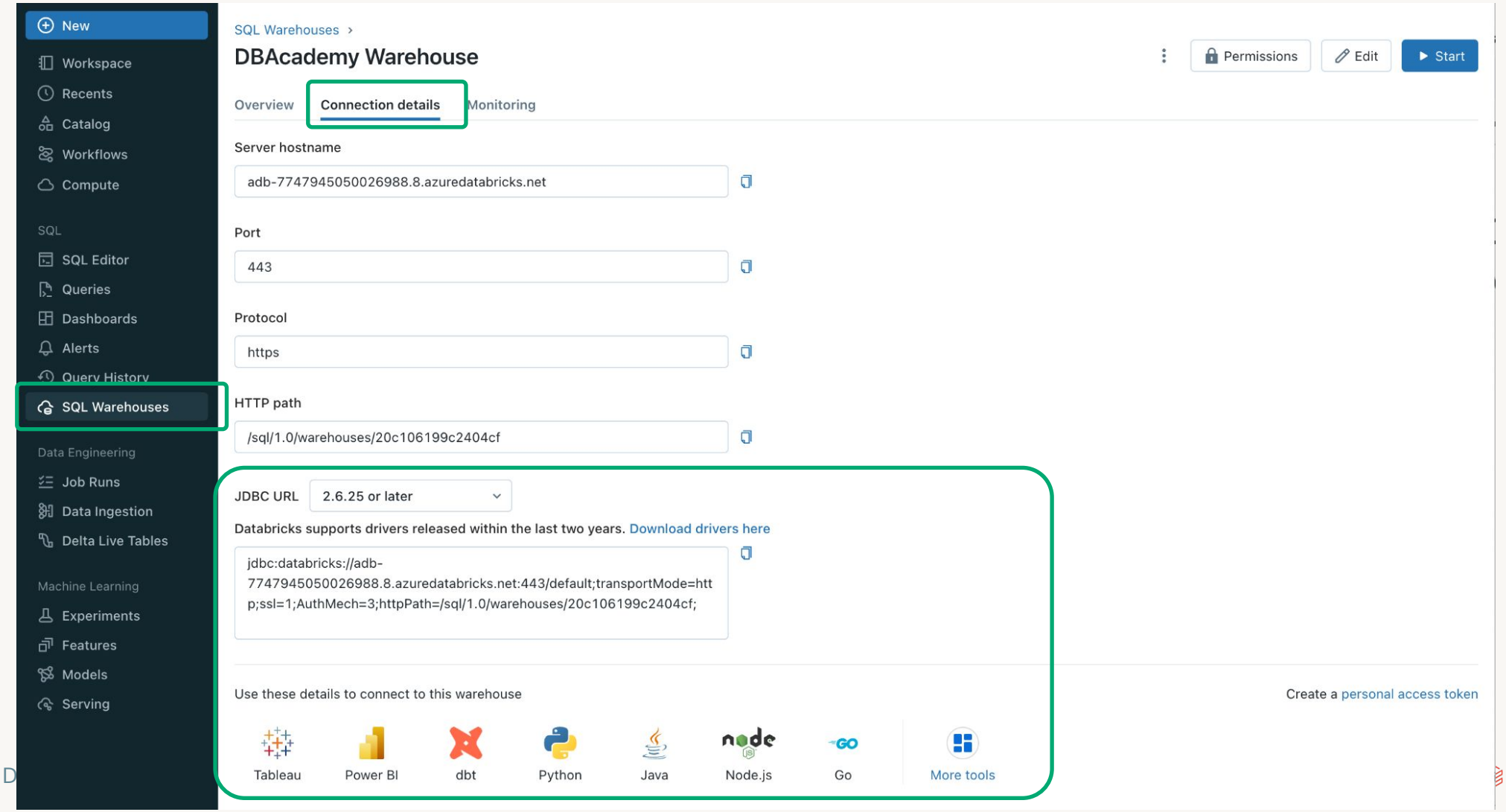
2. SQL Warehouse 생성 - Serverless compute, Spot instance policy





Databricks SQL 환경 설명

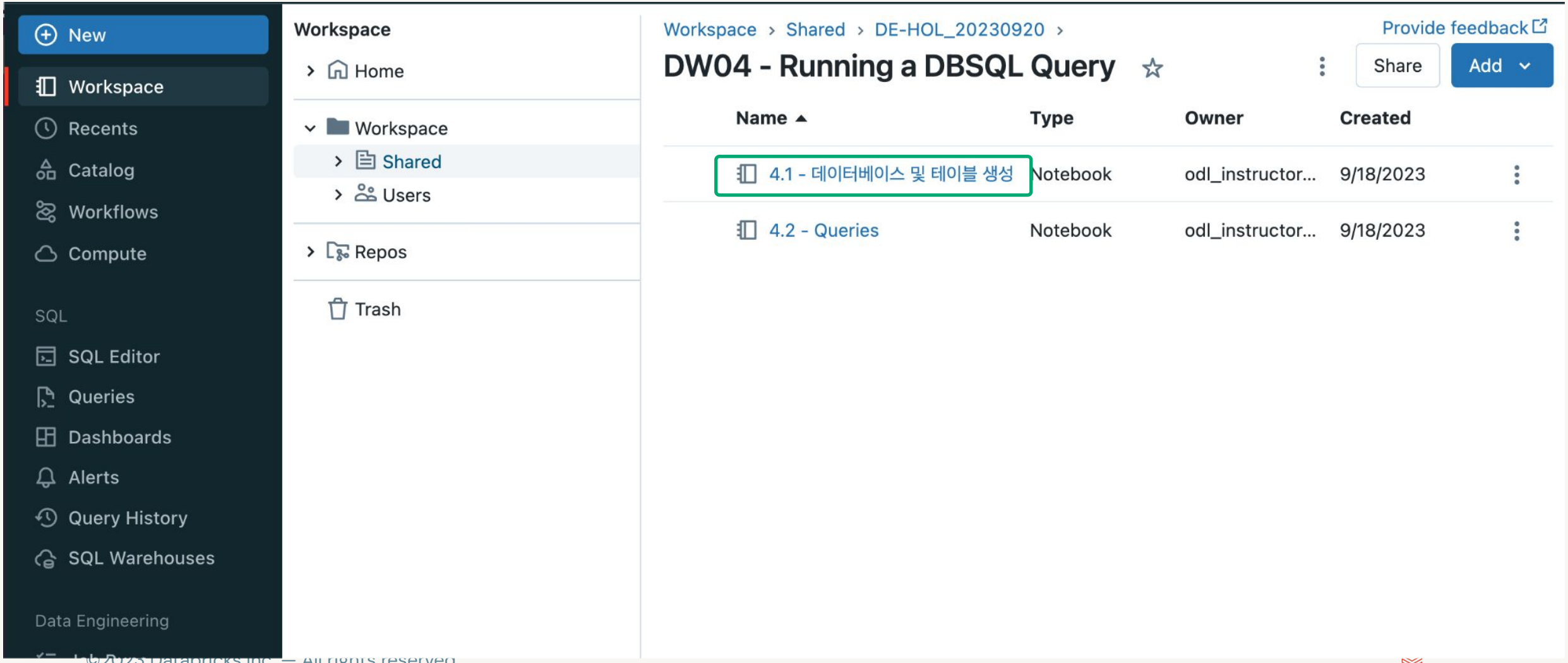
2. SQL Warehouse – 외부 BI 툴과의 연동



Hands on - DW04 Databricks SQL

Databricks SQL Demo & Lab

1. 실습을 위한 테이블 생성



Databricks SQL Demo & Lab

2. SQL Editor 에서 쿼리 수행

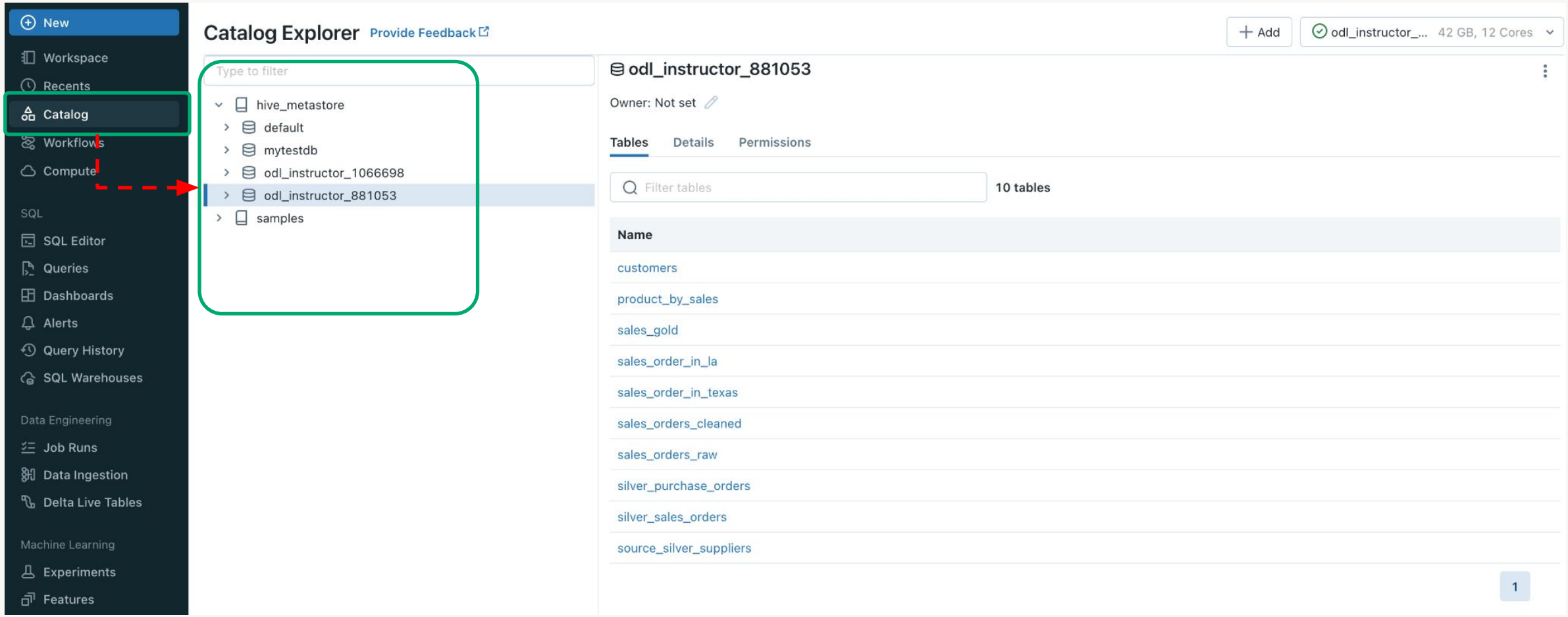
```
-- Q2. Simple aggregation: sales per product_category
SELECT SUM(total_price) total_sales,
    product_category
FROM sales_gold
GROUP BY product_category ;

-- Q3. sales per state
SELECT c.state,
    COUNT(s.customer_id) AS cust_count,
    SUM(s.total_price) AS sales_revenue
FROM sales_gold s
JOIN customers c ON c.customer_id = s.customer_id
GROUP BY c.state ;

-- Q4. total sales for Alert
SELECT sum(total_price) AS total_sales,
    count(customer_id) AS cust_cnt
FROM sales_gold ;
```

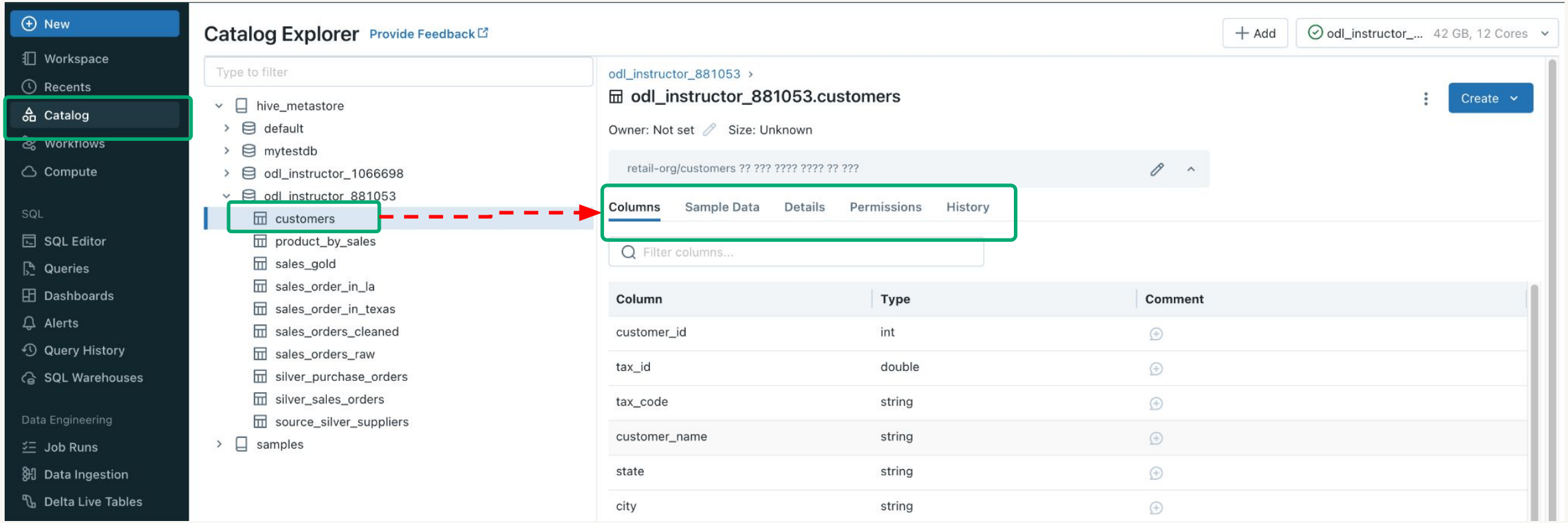
Databricks SQL Demo & Lab

3. 데이터베이스, 테이블 탐색



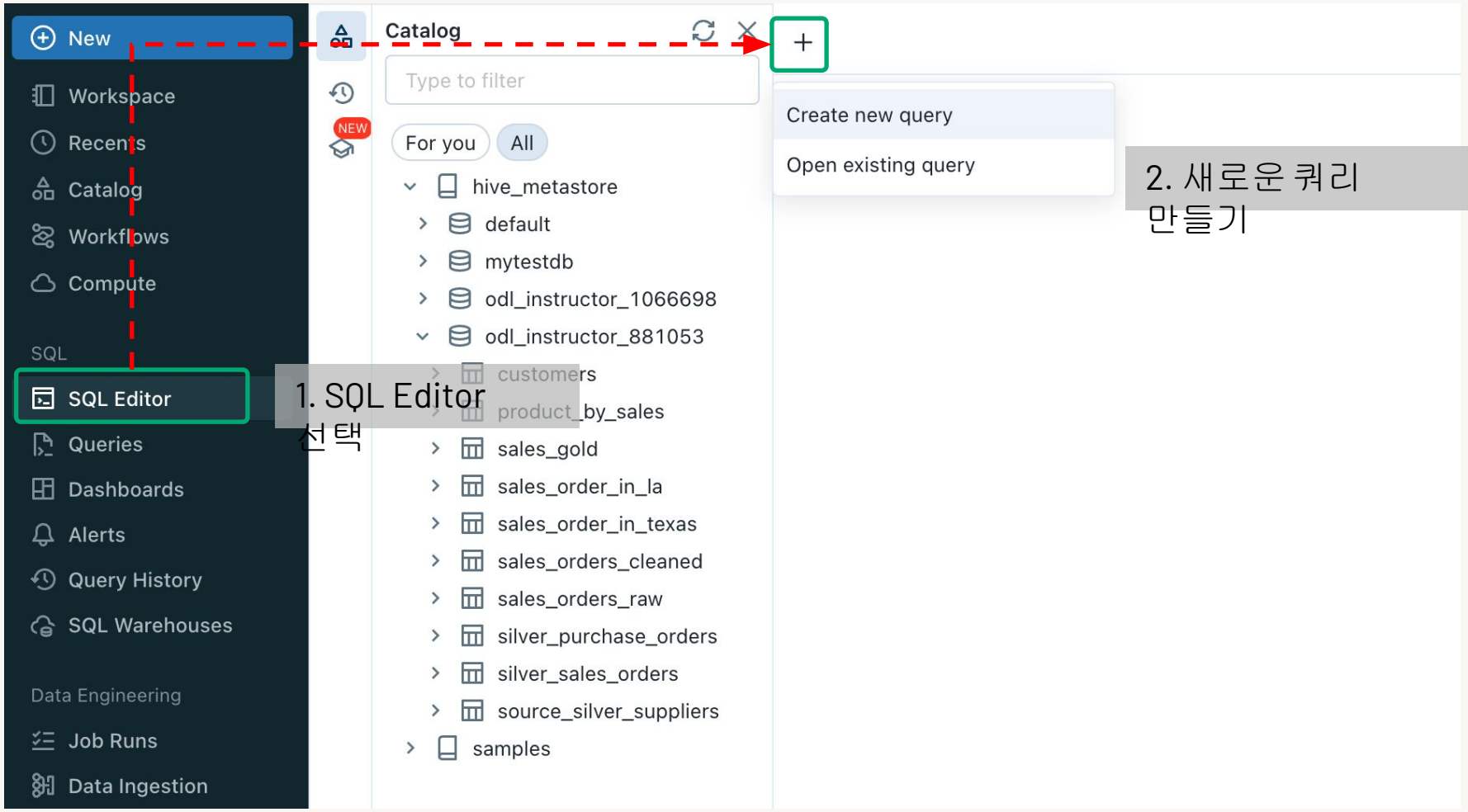
Databricks SQL Demo & Lab

3. 데이터베이스, 테이블 탐색



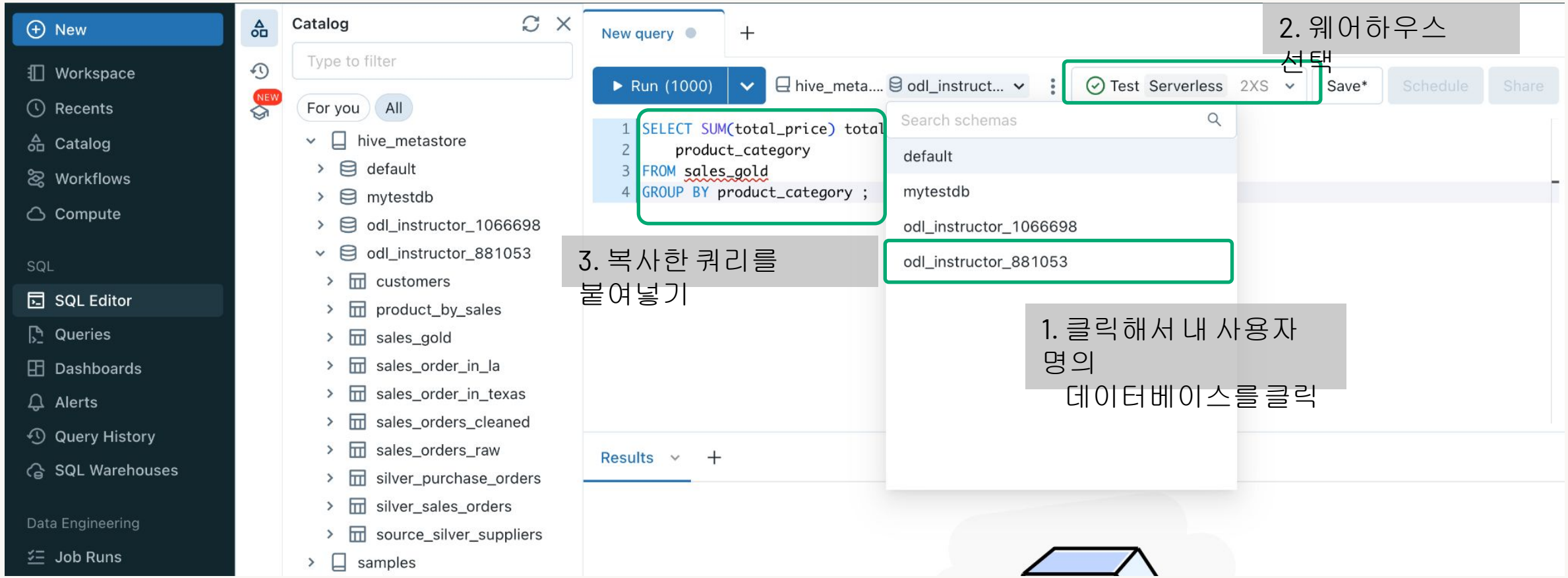
Databricks SQL Demo & Lab

4. 쿼리 작성 및 시각화



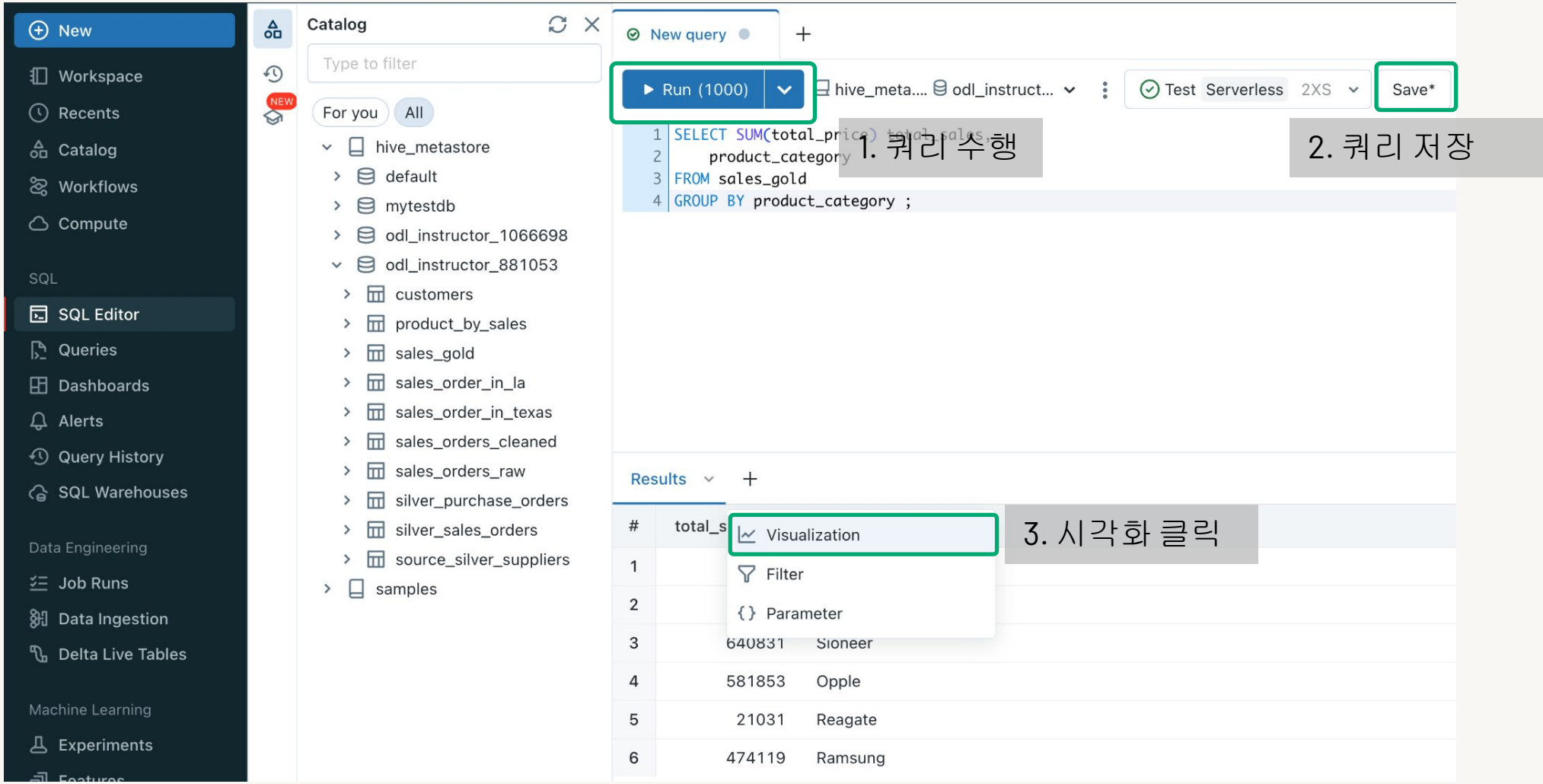
Databricks SQL Demo & Lab

4. 쿼리 작성 및 시각화



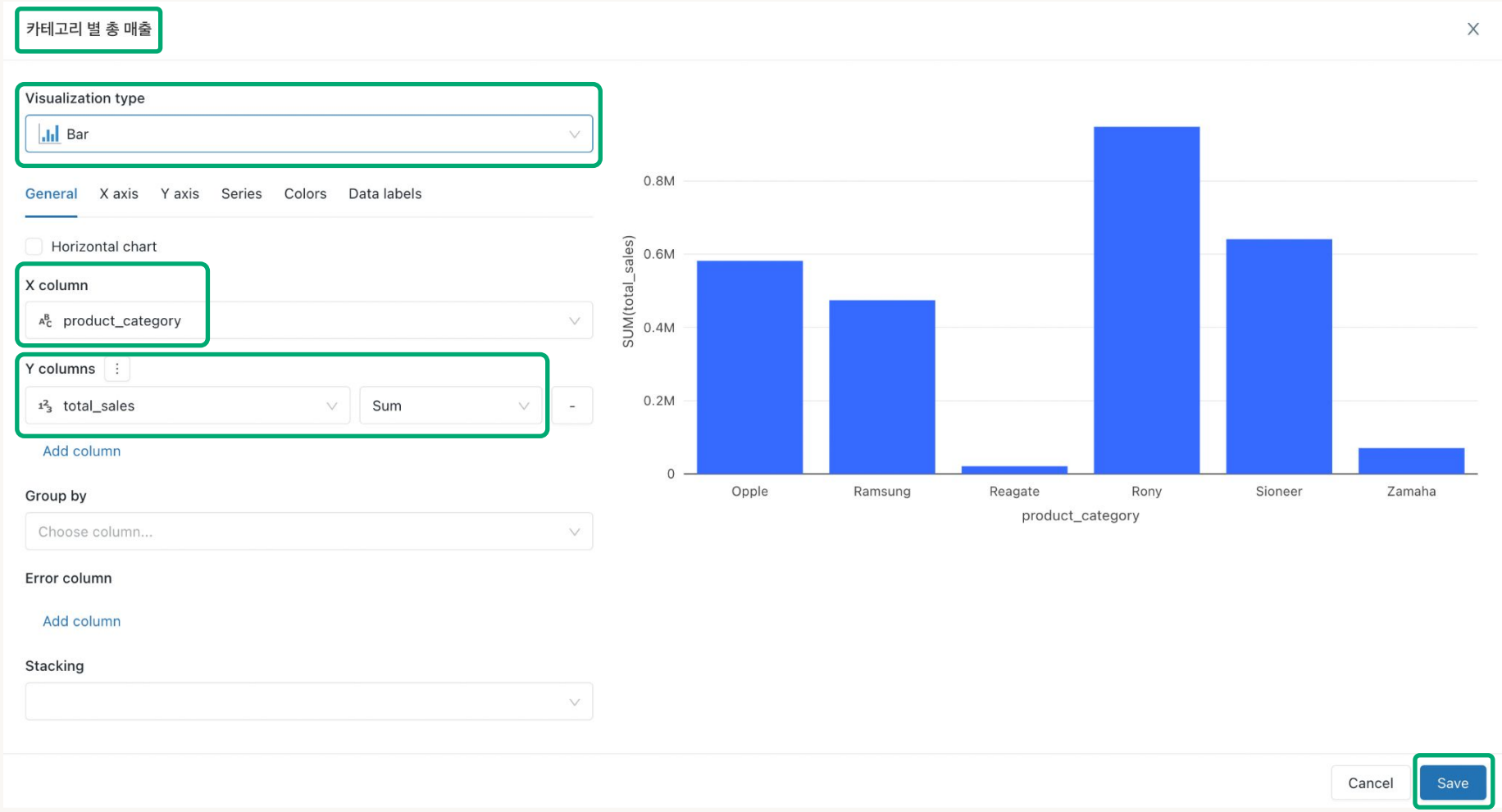
Databricks SQL Demo & Lab

4. 쿼리 작성 및 시각화



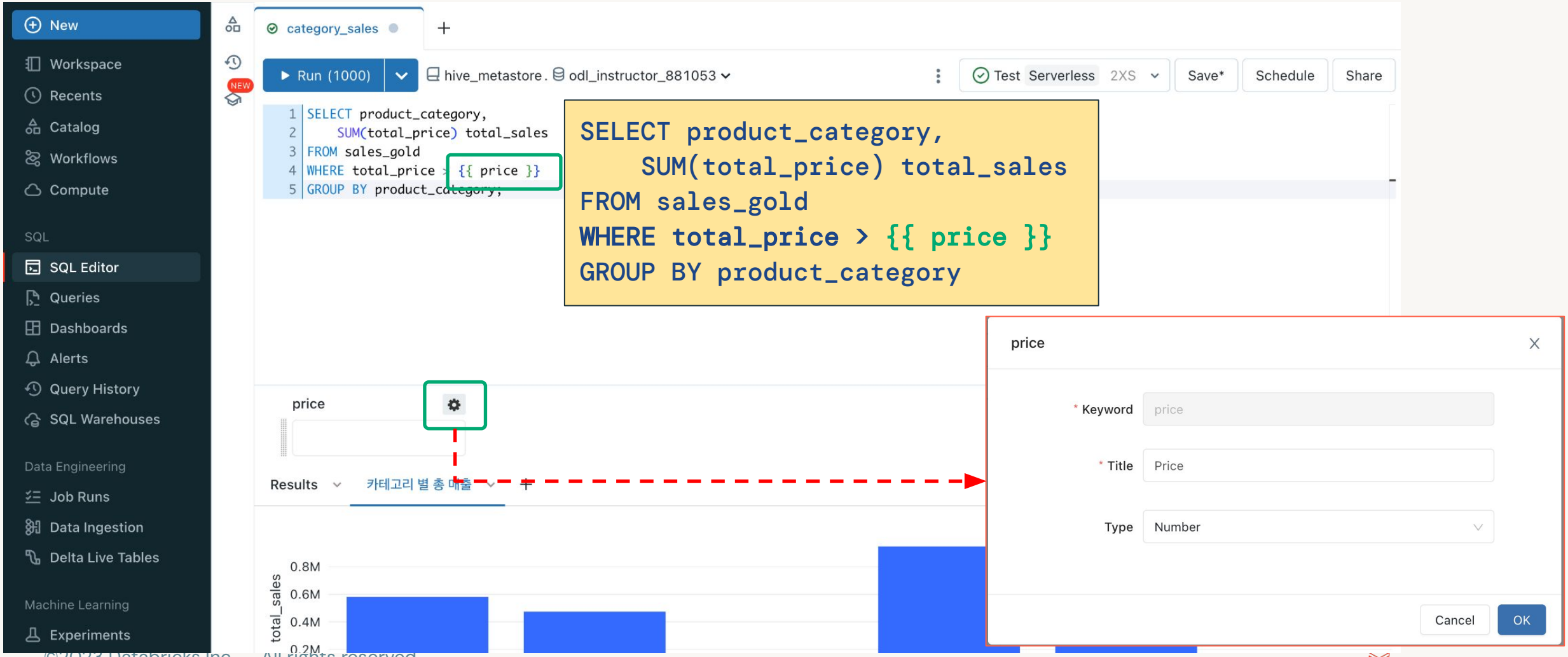
Databricks SQL Demo & Lab

5. 쿼리 작성 및 시각화



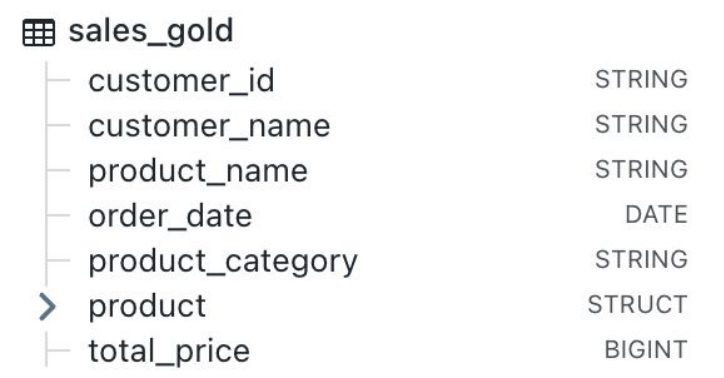
Databricks SQL Demo & Lab

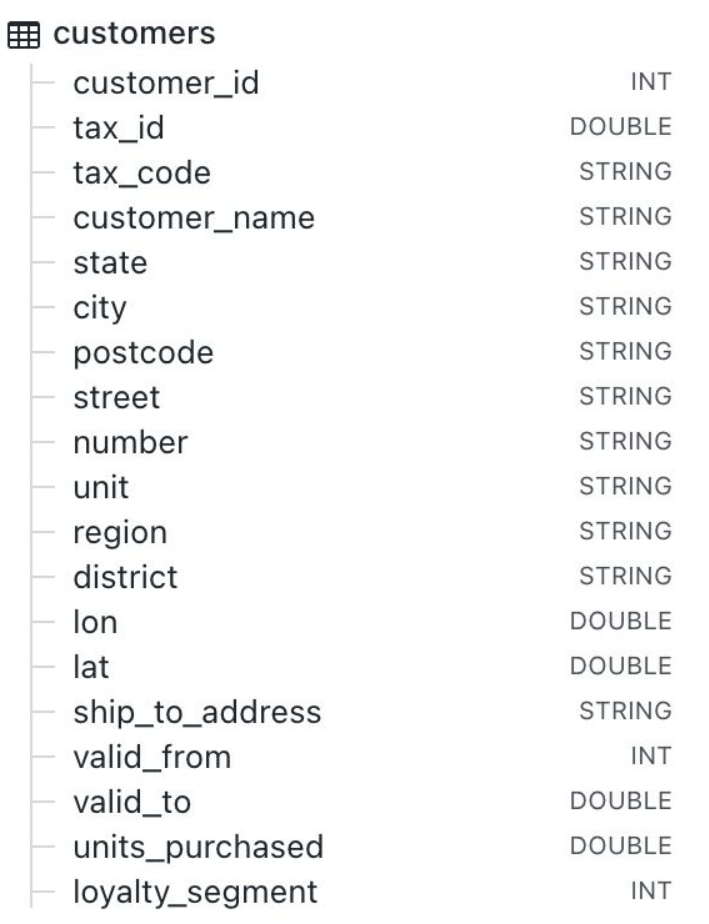
5. 쿼리 작성 및 시각화 - 매개변수



Databricks SQL Demo & Lab

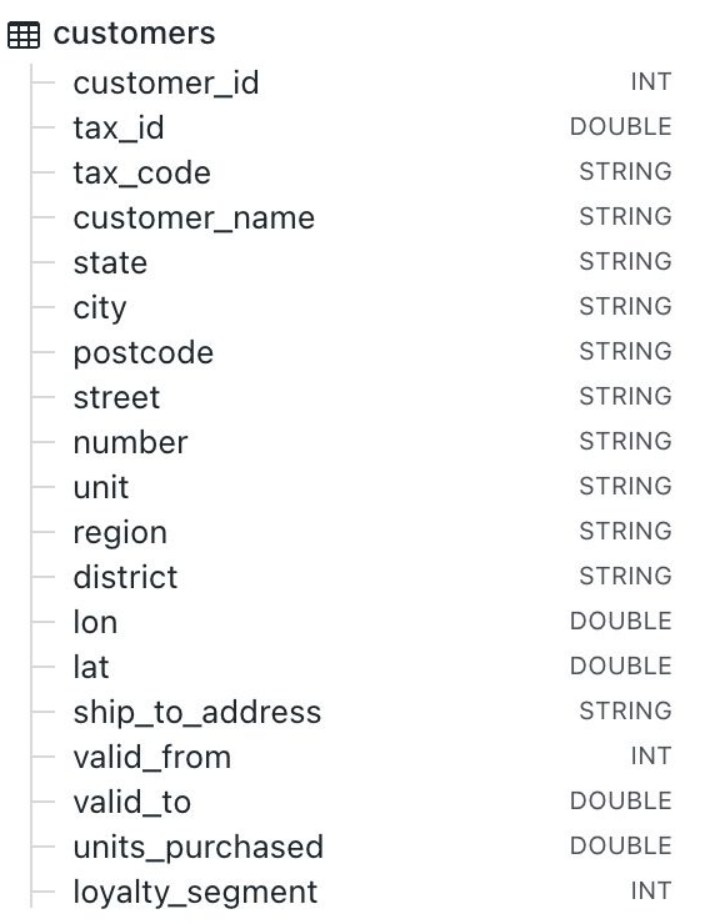
5. 쿼리 작성 및 시각화 - state 별 고객 수와 매출액 집계





Databricks SQL Demo & Lab

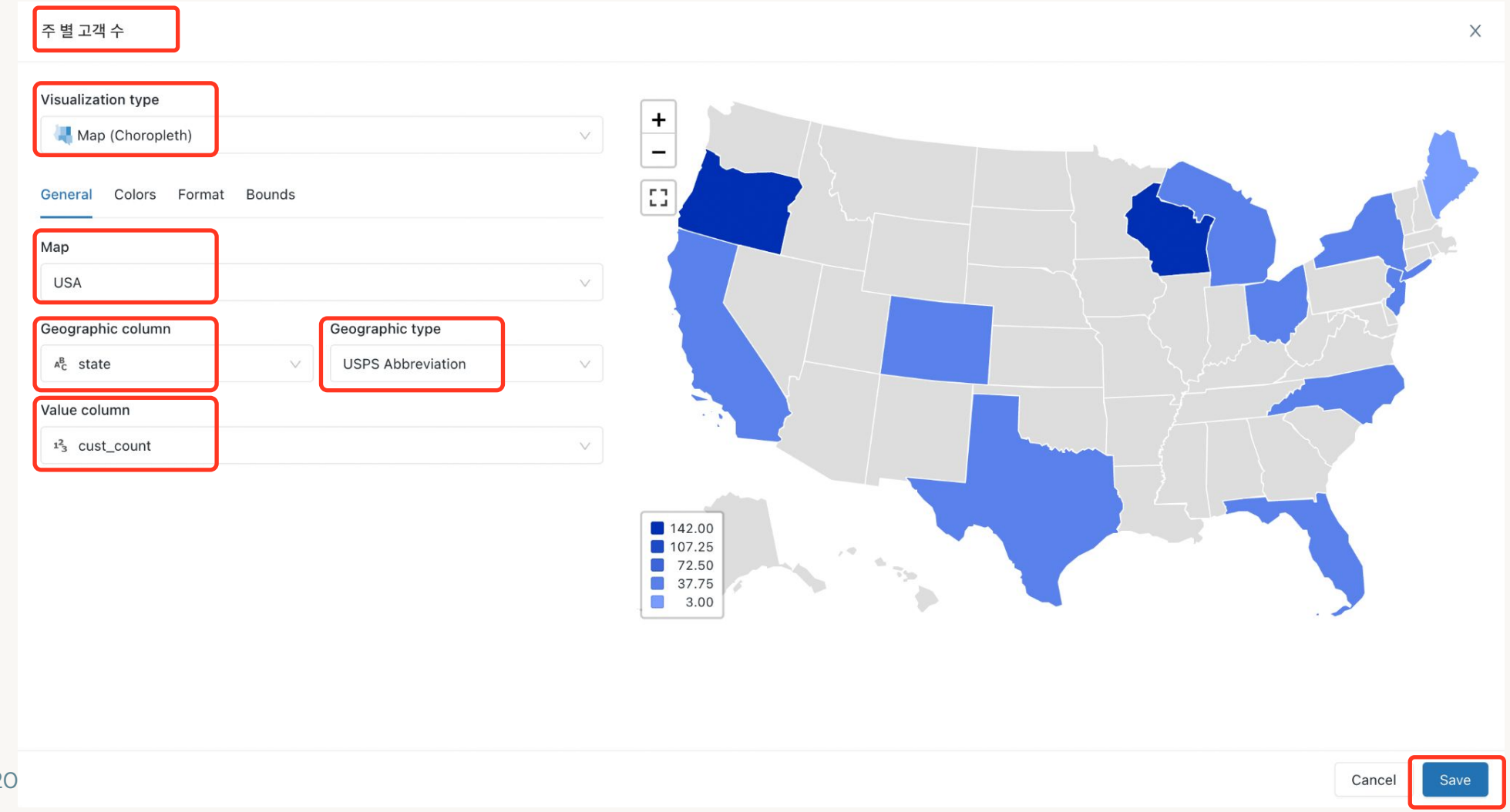
5. 쿼리 작성 및 시각화 - state 별 고객 수와 매출액 집계




```
SELECT c.state,
    COUNT(s.customer_id) AS cust_count,
    SUM(s.total_price) AS sales_revenue
FROM sales_gold s
JOIN customers c ON c.customer_id = s.customer_id
GROUP BY c.state
```

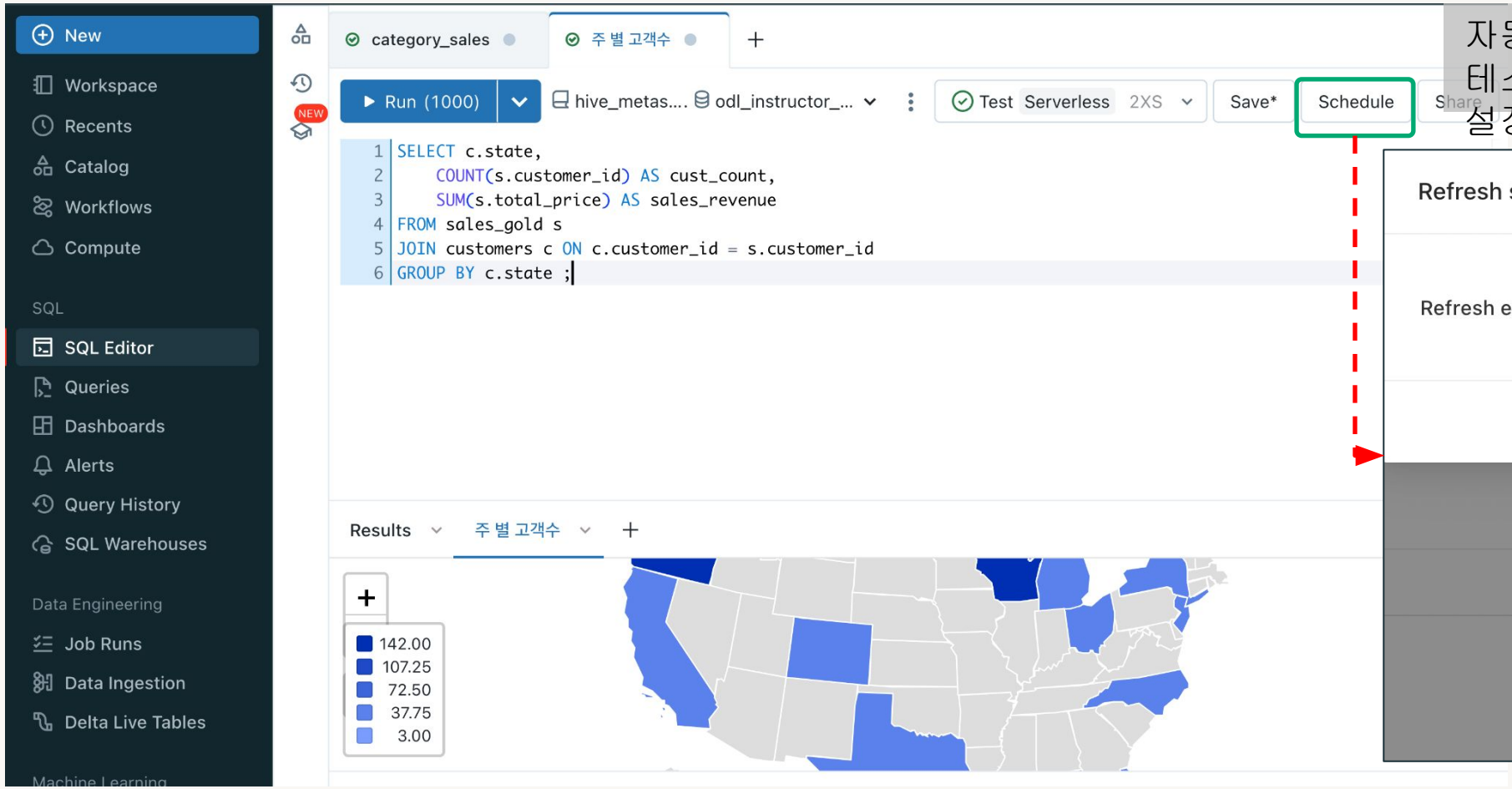
Databricks SQL Demo & Lab

5. 쿼리 작성 및 시각화 – state 별 고객 수와 매출액 집계

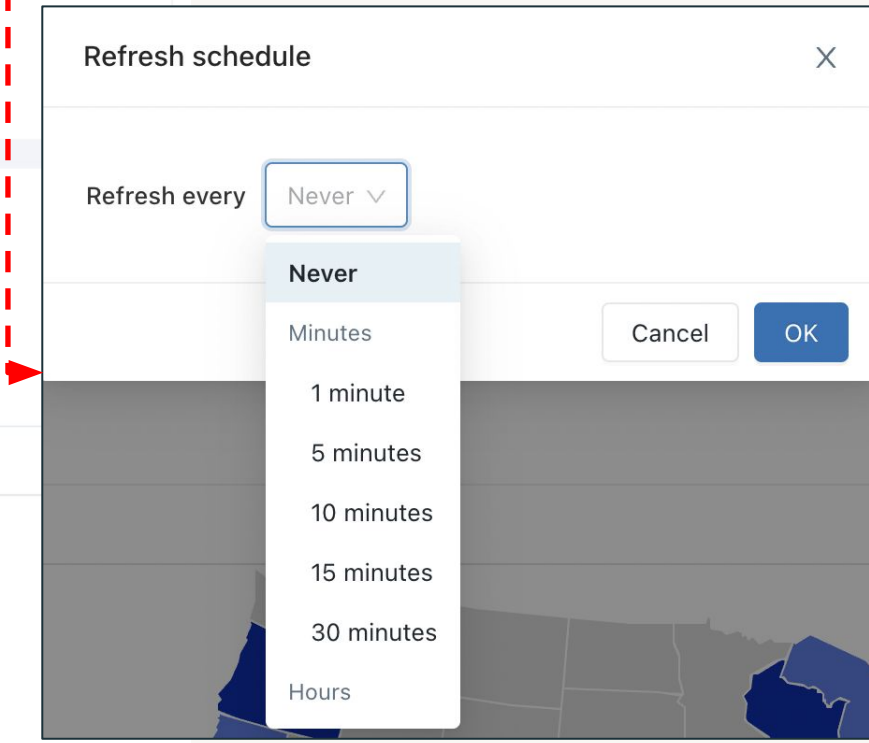


Databricks SQL Demo & Lab

5. 쿼리 작성 및 시각화 – state 별 고객 수와 매출액 집계

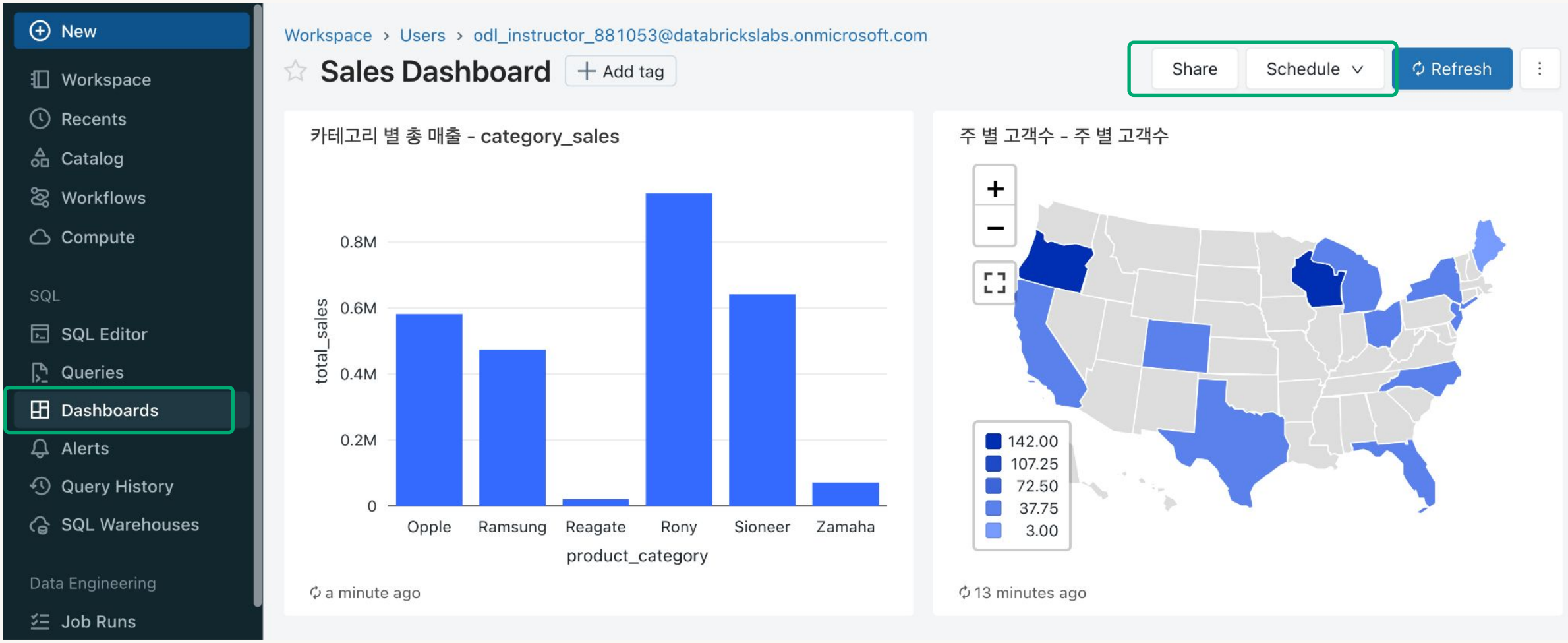


자동 Refresh 설정
테스트 후 Refresh every **Never**로 설정



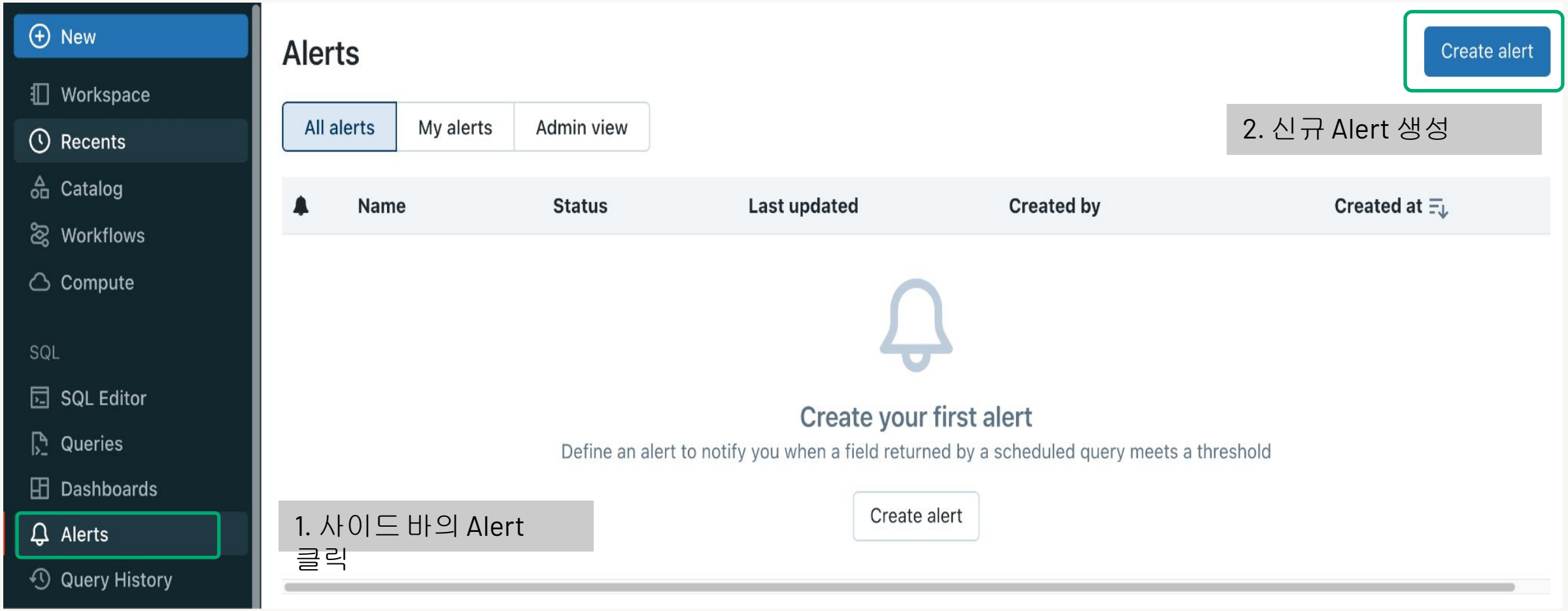
Databricks SQL Demo & Lab

6. Dashboard



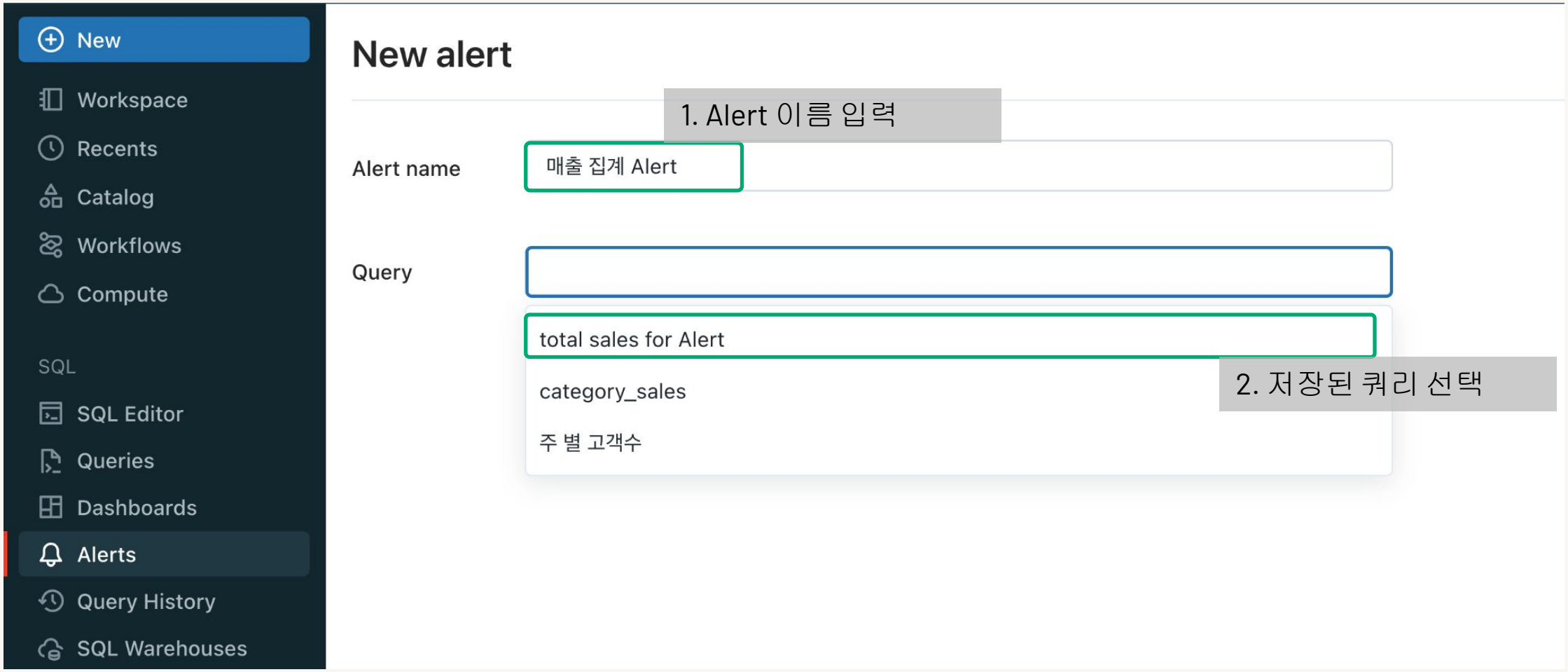
Databricks SQL Demo & Lab

7. Alert



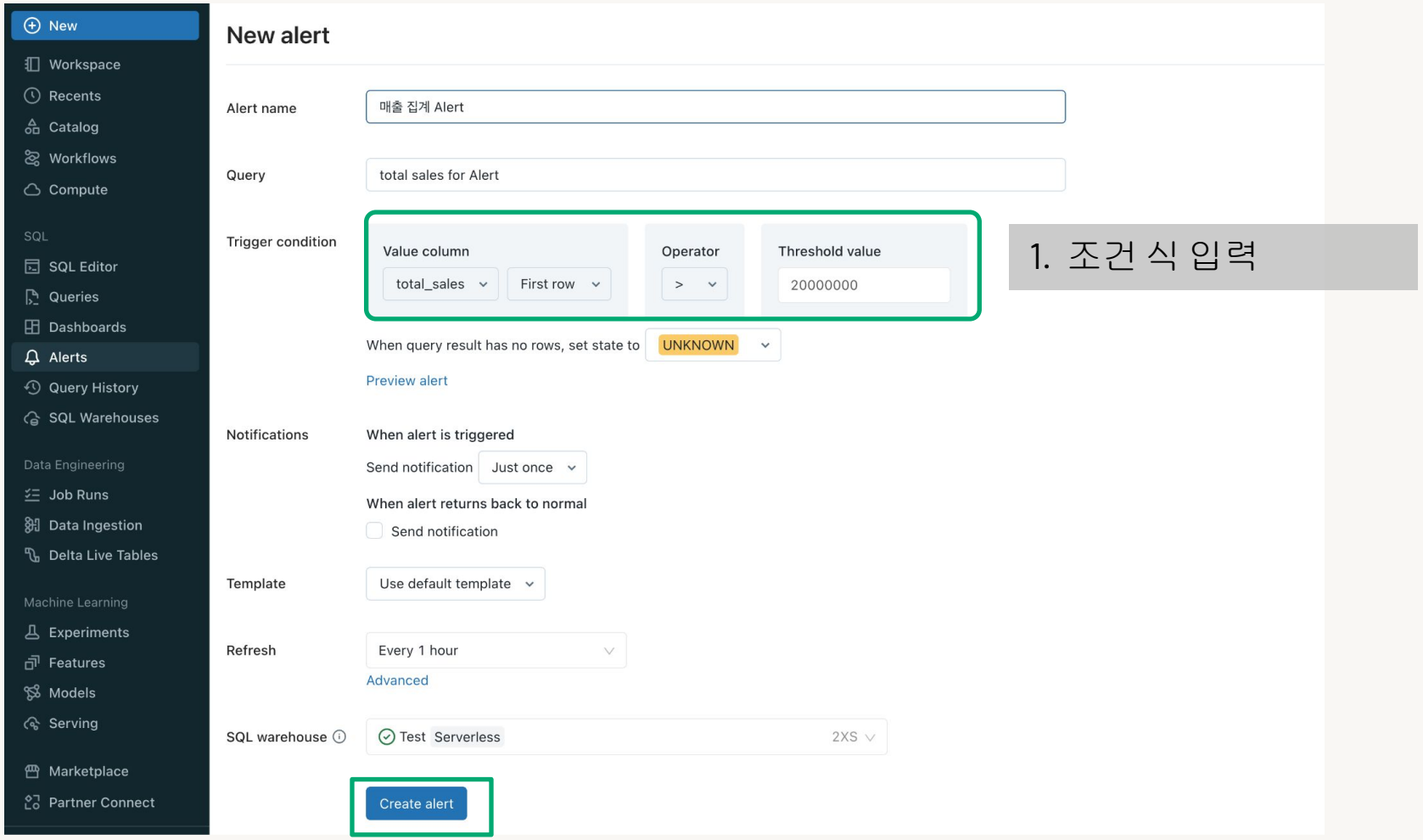
Databricks SQL Demo & Lab

7. Alert



Databricks SQL Demo & Lab

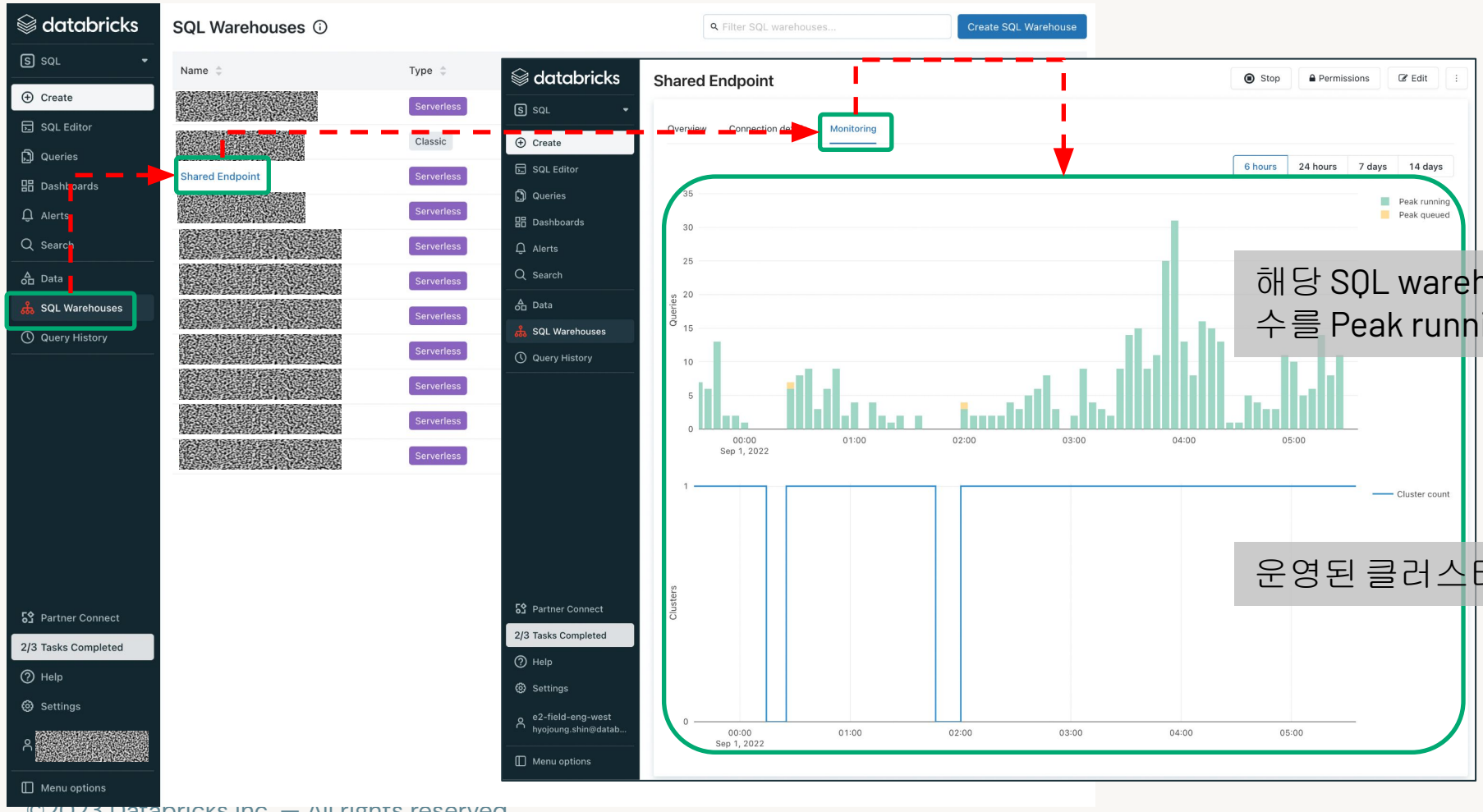
7. Alert



1. 조건 식 입력

Databricks SQL Demo & Lab

8. 모니터링, 쿼리 히스토리 및 프로파일링

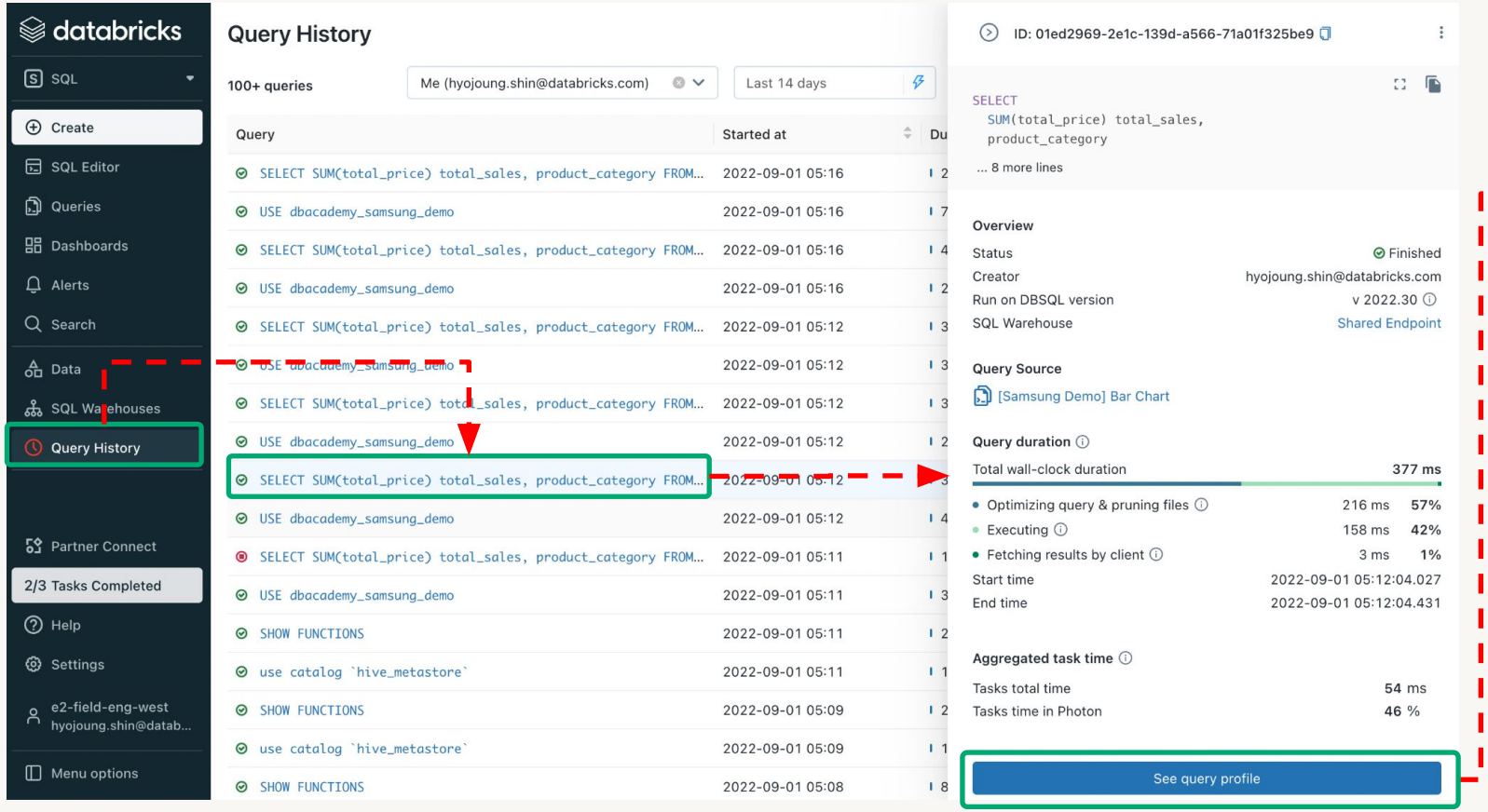


해당 SQL warehouse에서 수행된 쿼리의 수를 Peak running, Peak queued로 표시

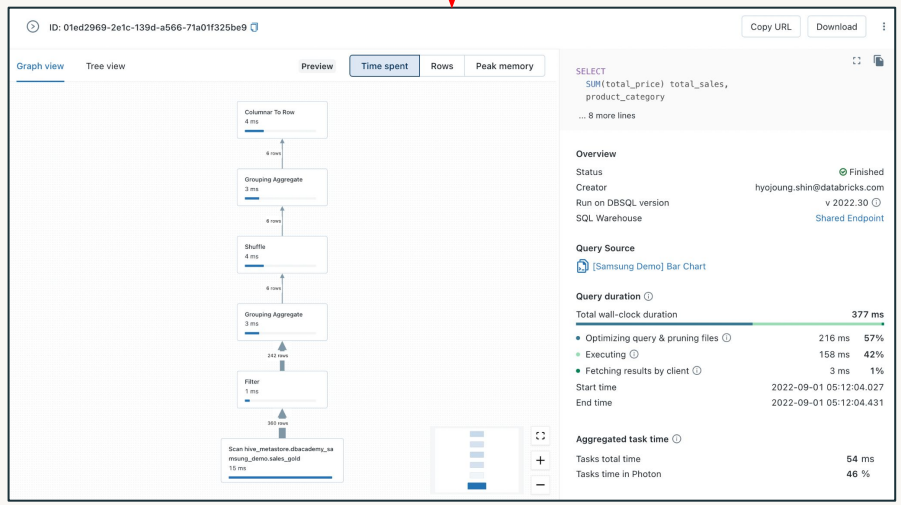
운영된 클러스터의 수

Databricks SQL Demo & Lab

8. 모니터링, 쿼리 히스토리 및 프로파일링

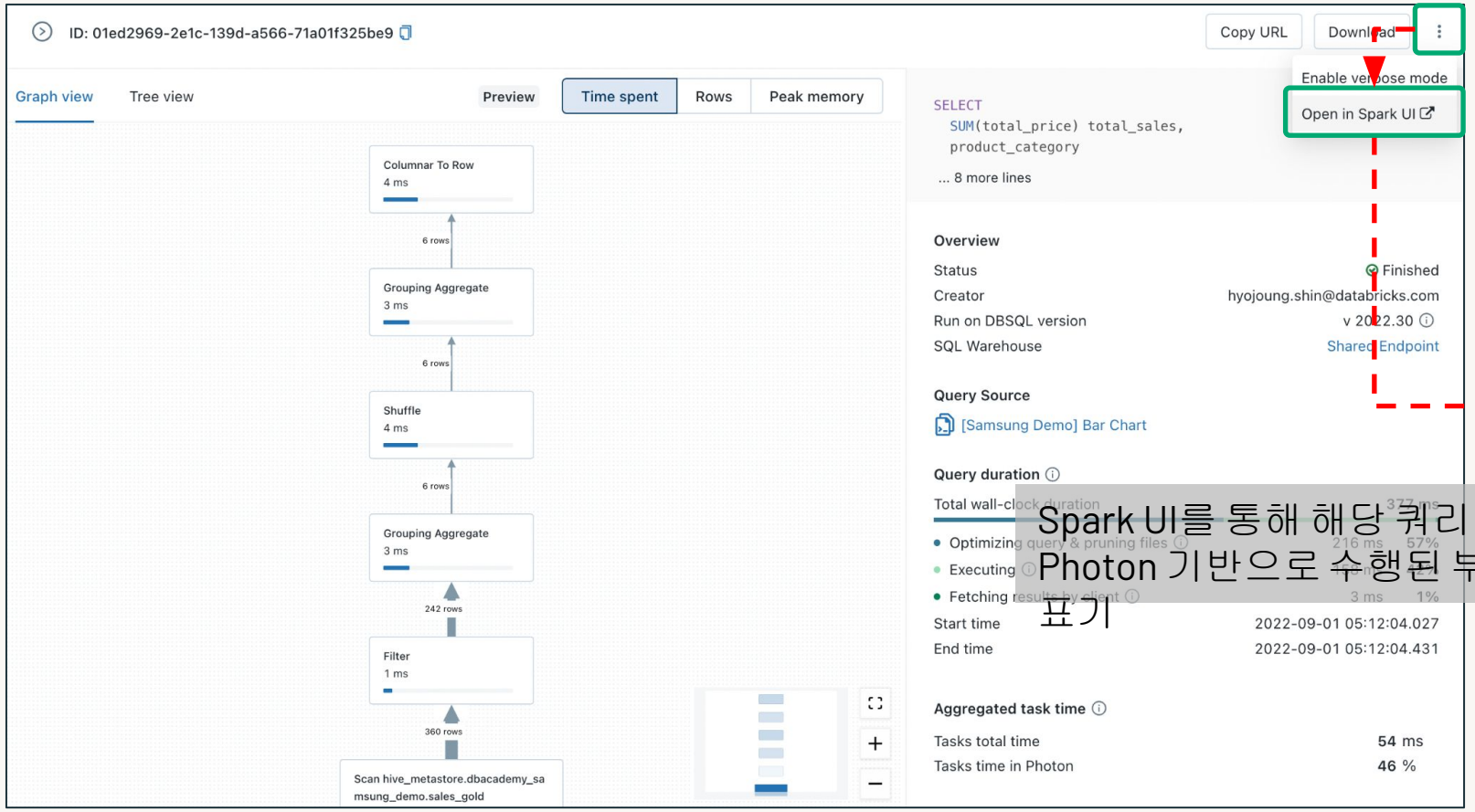


Query History로 검색 기능 제공
해당 쿼리 선택 후 view query profile 기능을 통해 상세 쿼리 실행 계획 확인 지원

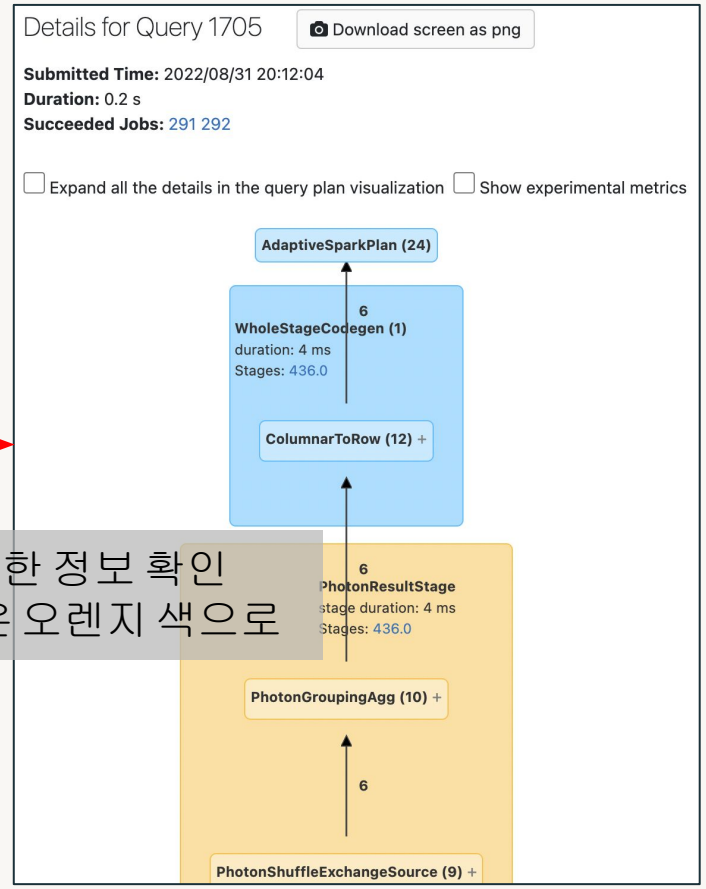


Databricks SQL Demo & Lab

8. 모니터링, 쿼리 히스토리 및 프로파일링



Spark UI를 통해 해당 쿼리에 대한 정보 확인
Photon 기반으로 수행된 부분은 오렌지 색으로 표기



Databricks SQL – Summary

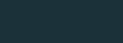
Data Lakehouse에서 BI와 SQL 워크로드 처리를 위한 고성능 + 동시성 향상 쿼리 엔진

Data Lake → DW 의 이중 적재 구조를 제거하여, TCO 절감, 분석 파이프라인 단순화

분석가들을 위한 Native SQL 인터페이스 지원

Built-in 대시보드 또는 기존 BI 툴들을 통해 빠른 인사이트 확보

Course Recap



Recap

1. 데이터브릭스에서 제공하는 서비스의 범위를 이해 합니다.
2. **Databricks Workspace**를 사용하여 **Notebook**으로 일반적인 코드 개발 작업을 할 수 있습니다.
3. **SQL**을 사용하여 다양한 소스에서 데이터를 추출 및 변환 작업을 수행하고, 결과 데이터를 **Delta Lake**에 적재 할 수 있습니다.
4. Databricks **Workflows Jobs** 으로 데이터 파이프 라인을 설정하고 실행을 할 수 있습니다.
5. **Databricks SQL** 을 사용해서 적재된 데이터를 조회하고 시각화해서 대시보드를 구성할 수 있습니다.

하반기 Databricks for Practitioners Series
다가오는 웨비나 일정

- 10월 12일 | 오후 2시–3시
Delta Lake로 레이크하우스 구축하기

- 11월 9일 | 오후 2시–3시
레이크하우스 기반의 데이터 엔지니어링 최적화 –
쉽고 경제적인 ETL 파이프라인 구축하기

- 12월 14일 | 오후 2시–3시
데이터, AI 자산의 통합 데이터 거버넌스와 공유 –
레이크하우스 유니티 카탈로그

- 1월 11일 | 오후 2시–3시
Databricks SQL 로 Warehouse 리모델링 하기

지금 등록하기

databricks

설문 참여 이벤트

설문에 참여해주신 분들께
공식 데이터브릭스 부트캠프 뱃지를 드립니다!

궁금하신 사항은 koreamarketing@databricks.com 문의를 부탁드립니다.

databricks