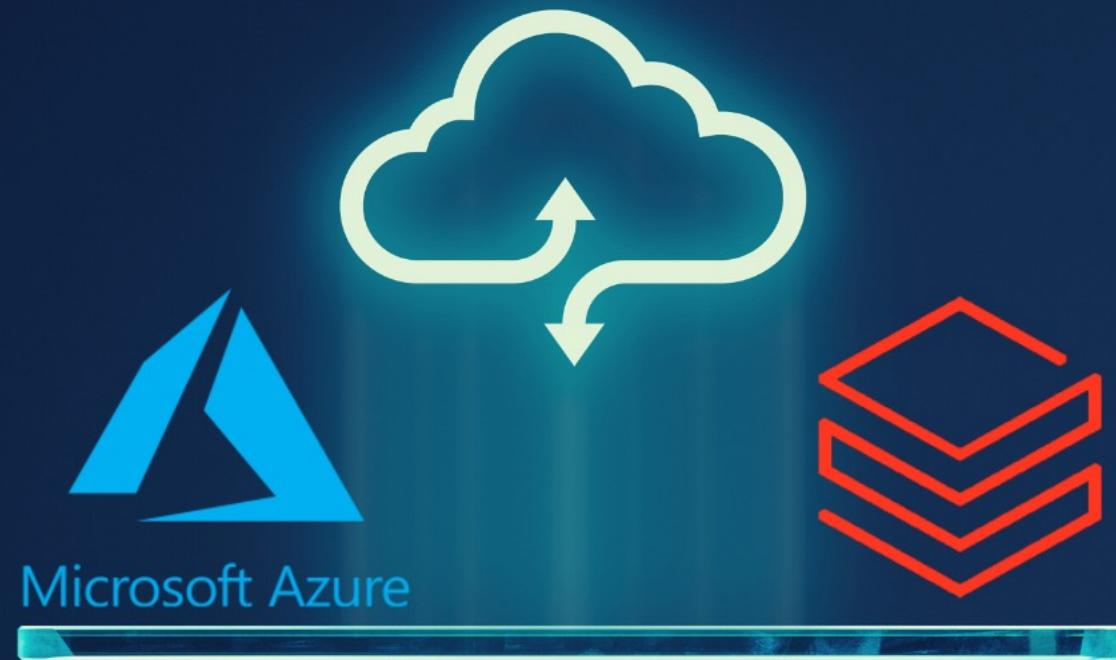


Azure Databricks For Data Engineers

Project on Formula1 Racing



About Me



Ramesh Retnasamy
Data Engineer/ Machine Learning Engineer



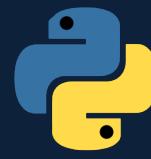
LinkedIn

<https://www.linkedin.com/in/ramesh-retnasamy/>

About this course



Azure Databricks



PySpark



Spark SQL



Delta Lake

About this course



Azure Databricks



Azure Data Lake Storage Gen2



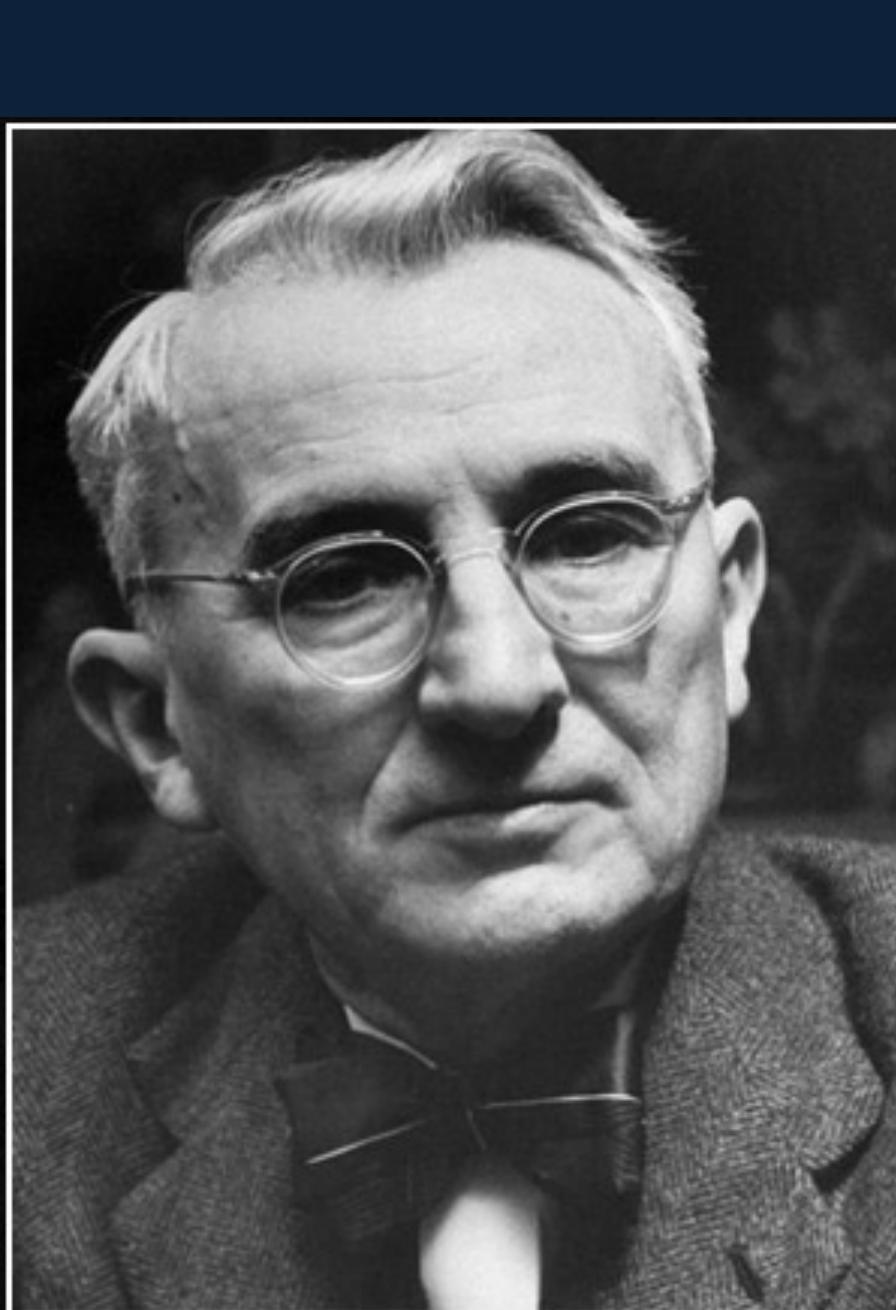
Azure Data Factory



Azure Key Vault



PowerBI

A black and white portrait of Dale Carnegie. He is shown from the chest up, wearing round-rimmed glasses and a dark, textured suit jacket over a light-colored shirt. He has a thoughtful expression, with his right hand resting against his chin. The background is dark and out of focus.

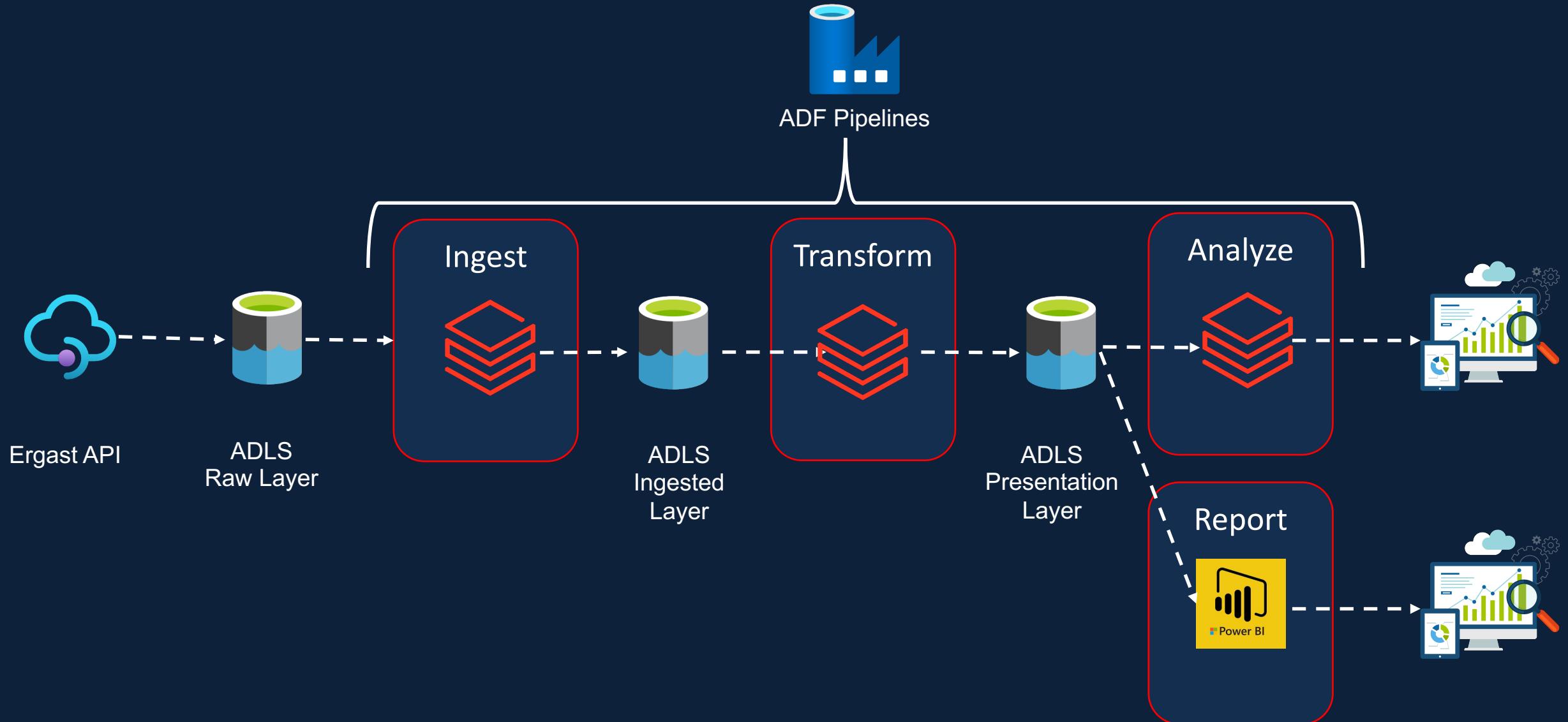
Learning is an active process. We
learn by doing.. Only knowledge that
is used sticks in your mind.

— *Dale Carnegie* —

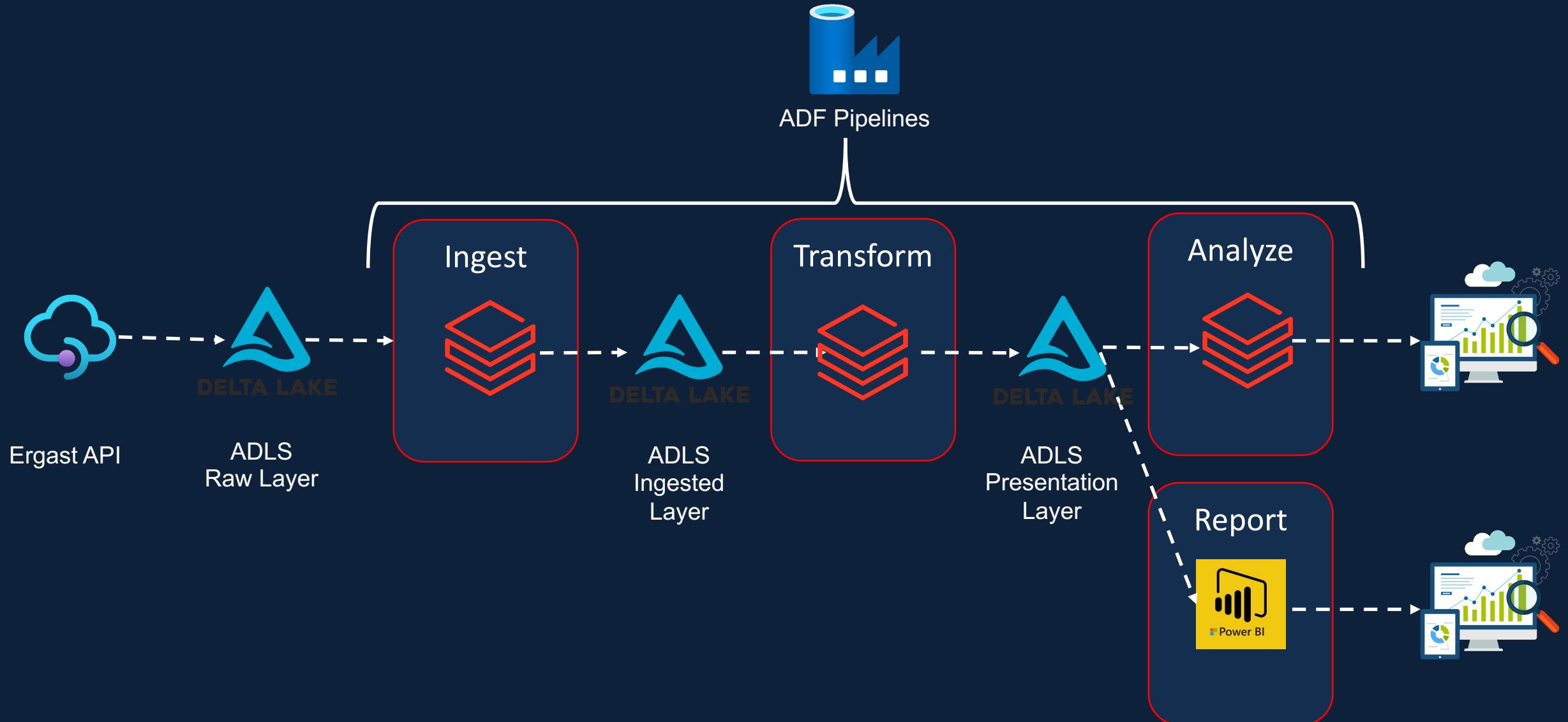


Formula1 Cloud Data Platform

Formula1 Cloud Data Platform

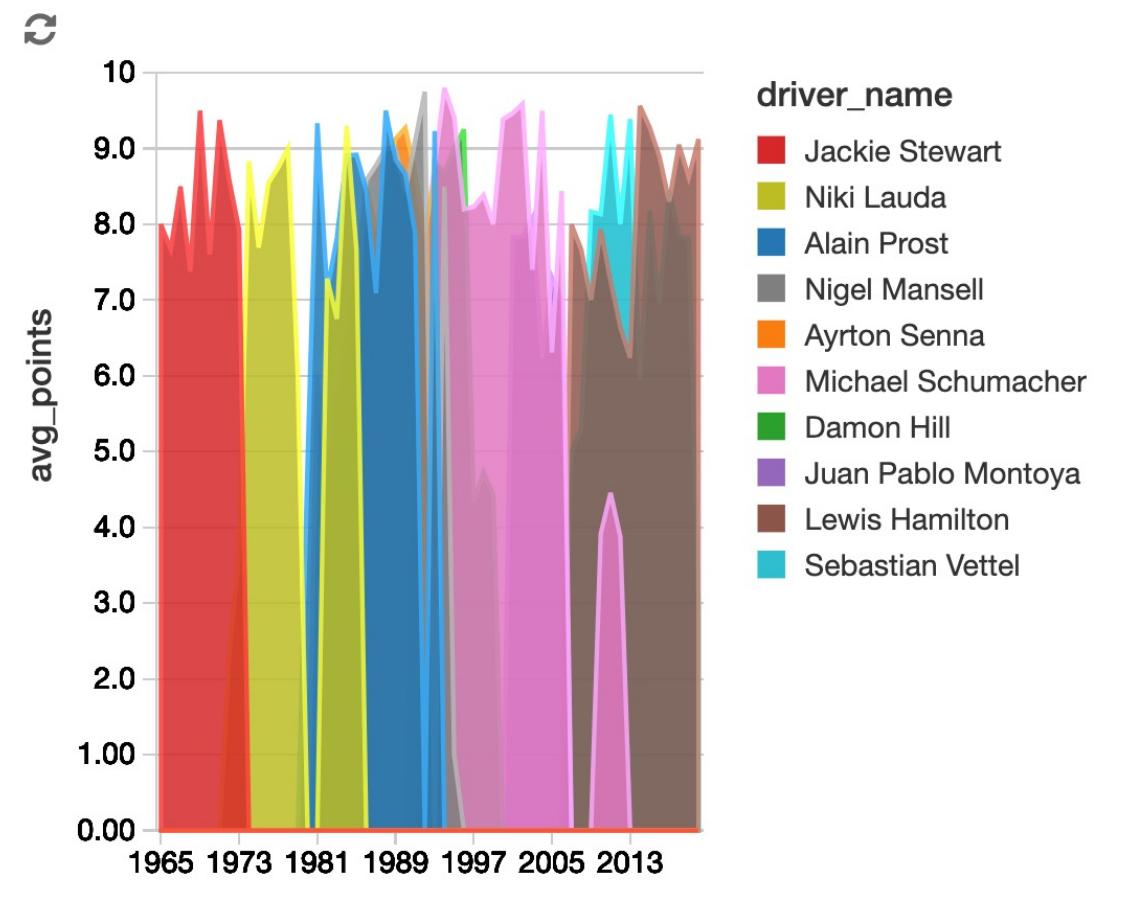
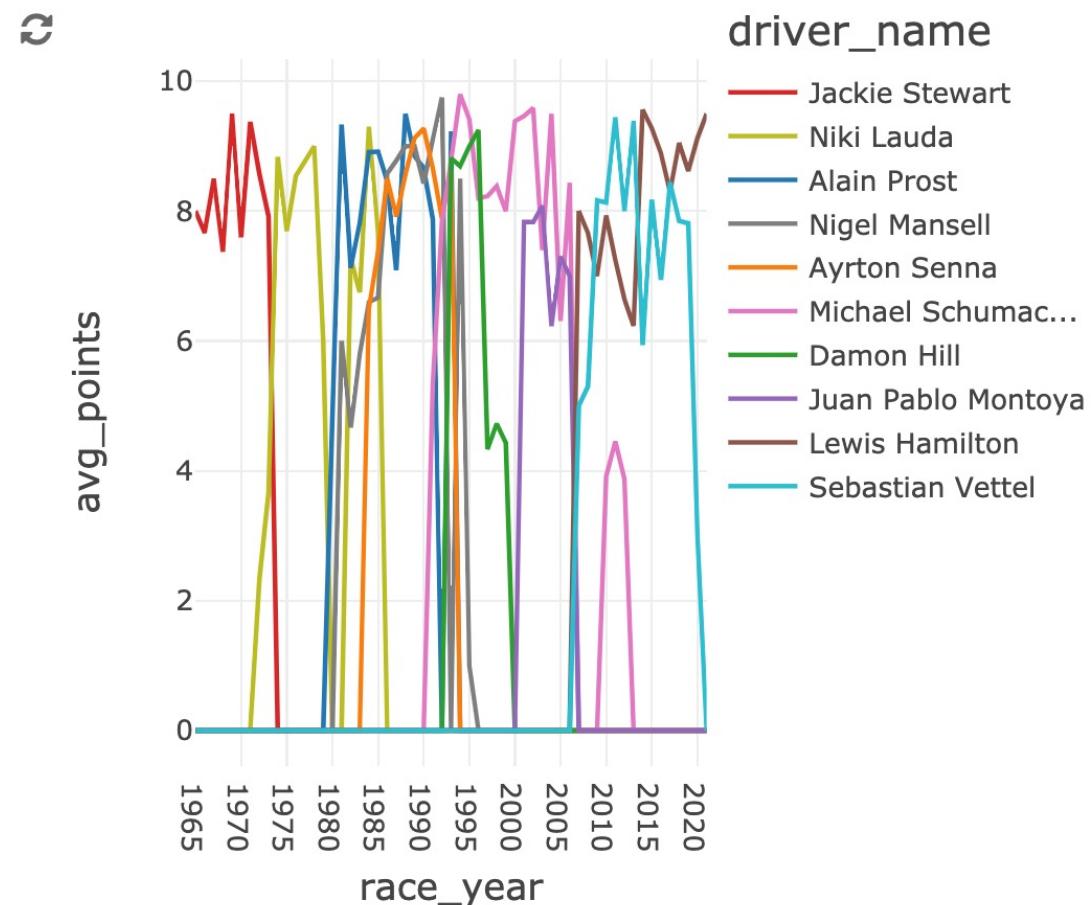


Formula1 Cloud Data Platform



Formula1 Cloud Data Platform

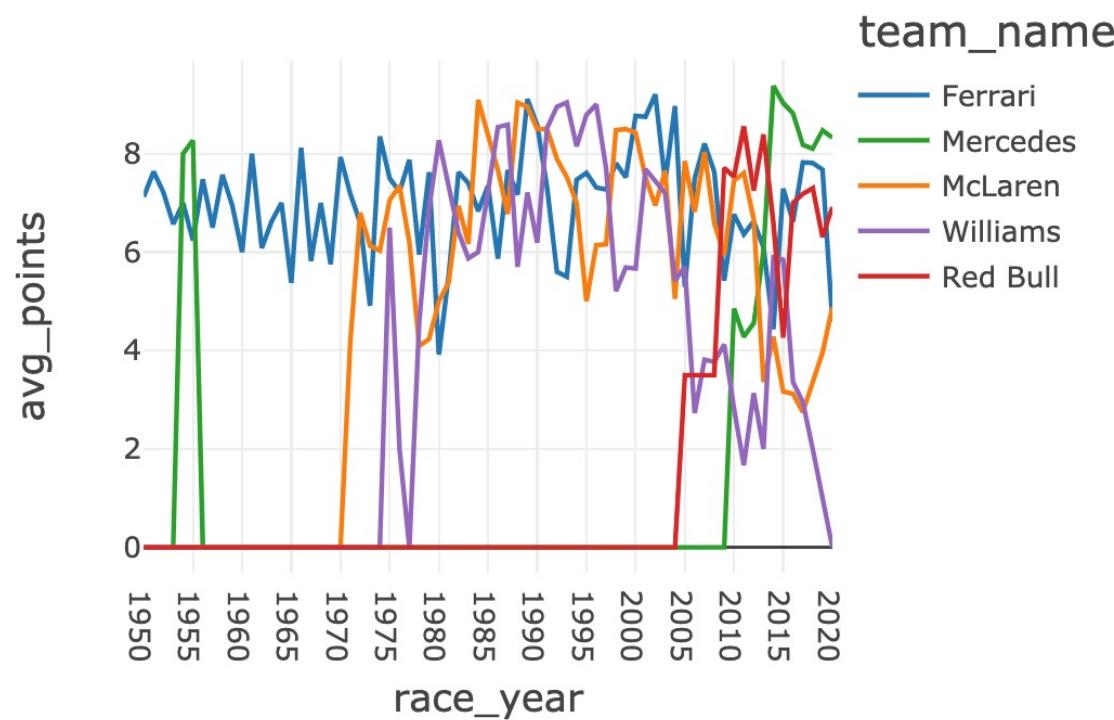
Dominant Formula 1 Drivers of All Time



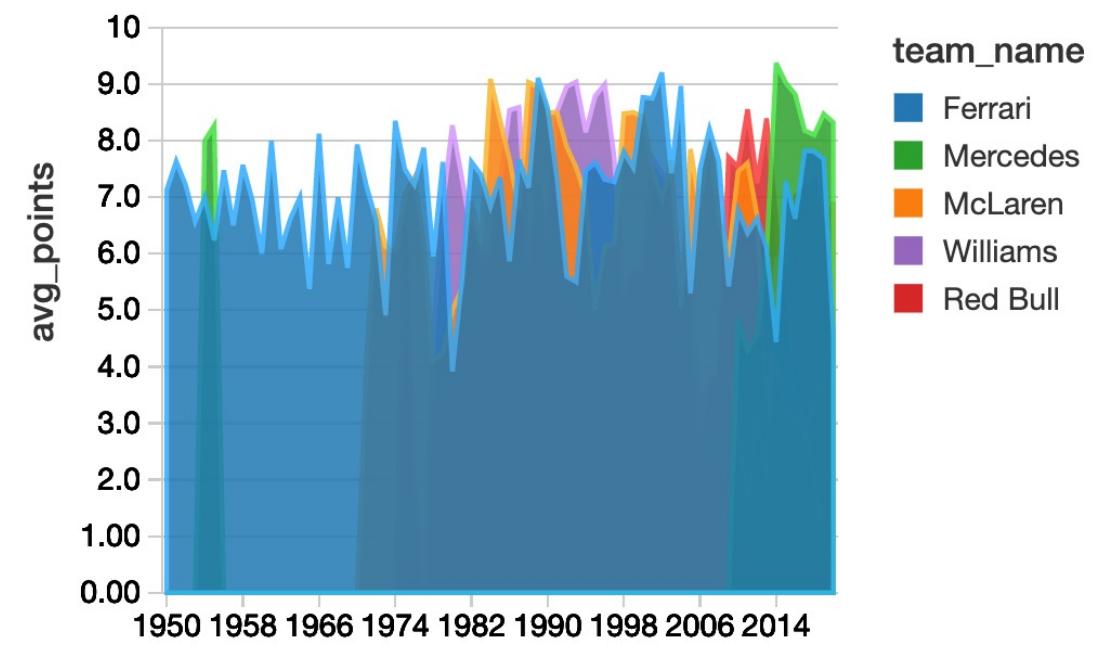
Formula1 Cloud Data Platform

Dominant Formula 1 Teams of All Time

Dominant Teams



Dominant Teams



Who is this course for

University students

IT Developers from other disciplines

AWS/ GCP/ On-prem Data Engineers

Data Architects

Who is this course not for

You are not interested in hands-on learning approach

Your only focus is Azure Data Engineering Certification

You want to Learn Spark ML or Streaming

You want to Learn Scala or Java

Pre-requisites

All code and step-by-step instructions provided

Basic Python programming knowledge required

Basic SQL knowledge required

Cloud fundamentals will be beneficial, but not mandatory

Azure Account

Our Commitments

Ask Questions, I will answer 😊

Keeping the course up to date

Udemy life time access

Udemy 30 day money back guarantee

Course Structure

Overviews

2. Azure Portal

3. Azure Databricks

7. Project Overview

8. Spark Overview

Databricks

4. Clusters

5. Notebooks

6. DBFS

12. Jobs

Spark (Python)

9. Data Ingestion 1

10. Data Ingestion 2

11. Data Ingestion 3

13. Transformation

14. Aggregations

19. Incremental Load

Spark (SQL)

15. Temp Views

16. DDL

17. DML

18. Analysis

19. Incremental Load

Delta Lake

20. Delta Lake

Orchestration

21. Azure Data Factory

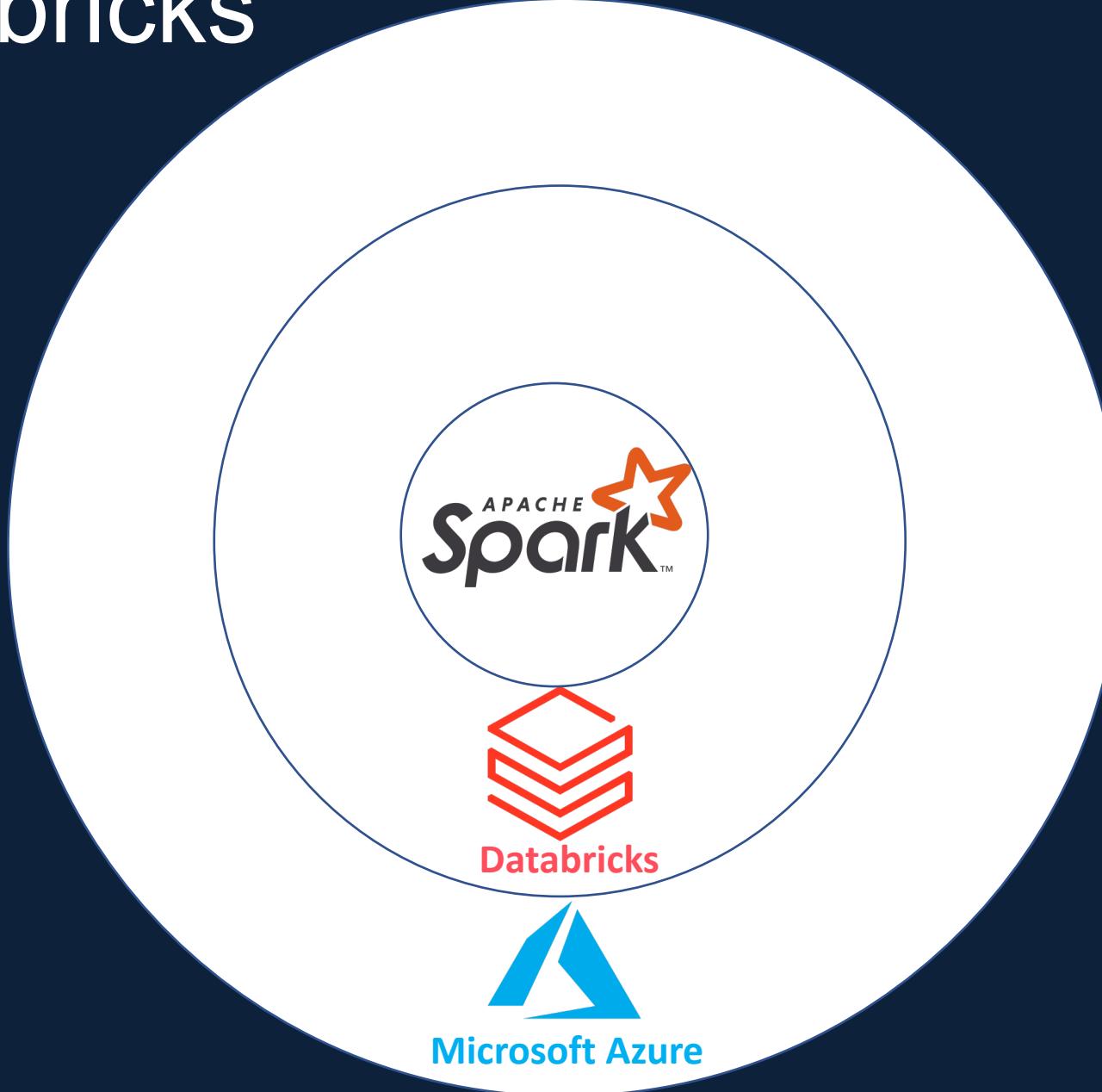
22. Connecting Other Tools



Introduction to Azure Databricks



Azure Databricks



Apache Spark

Apache Spark is a lightning-fast unified analytics engine for big data processing and machine learning



100% Open source under Apache License

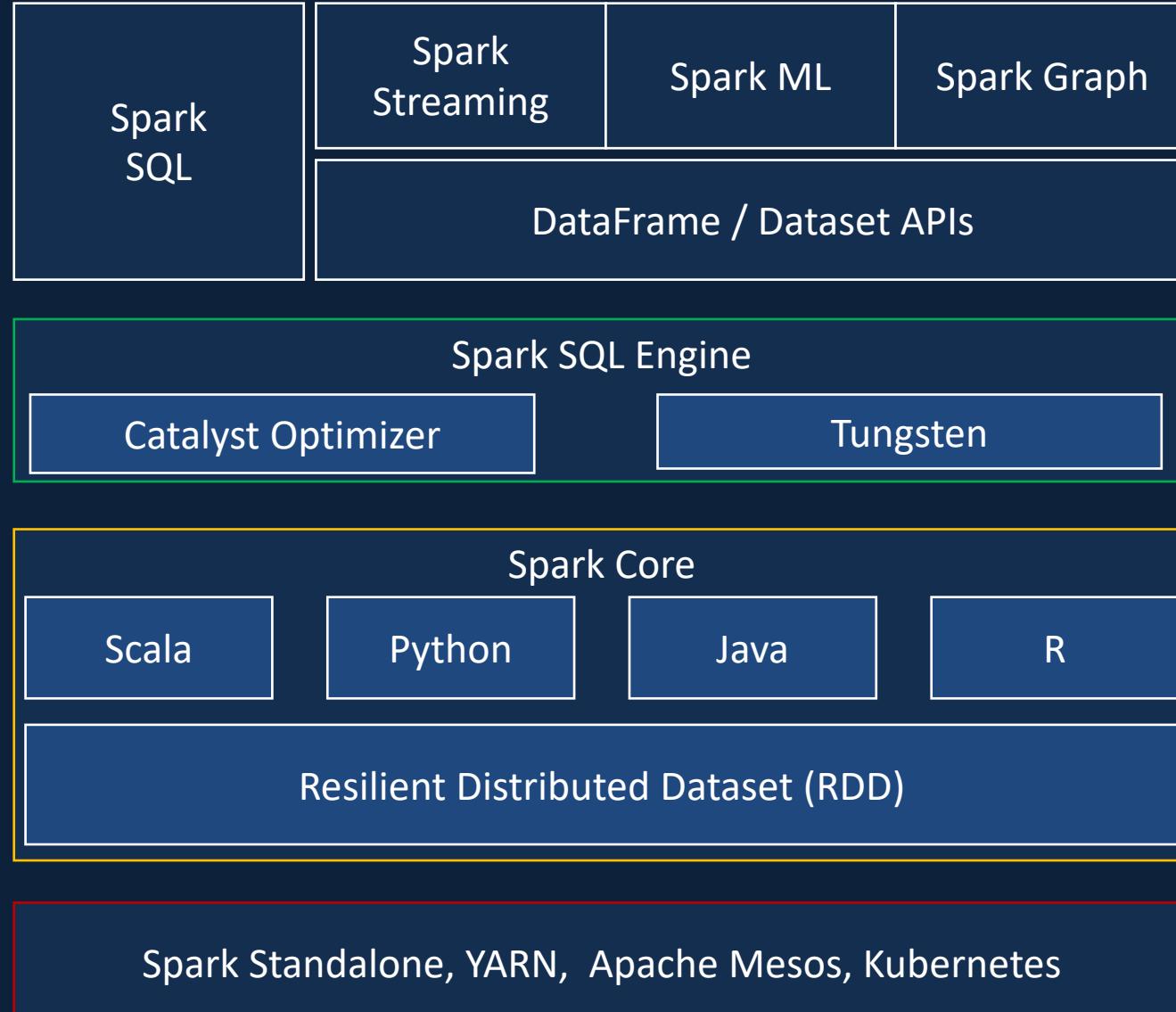
Simple and easy to use APIs

In-memory processing engine

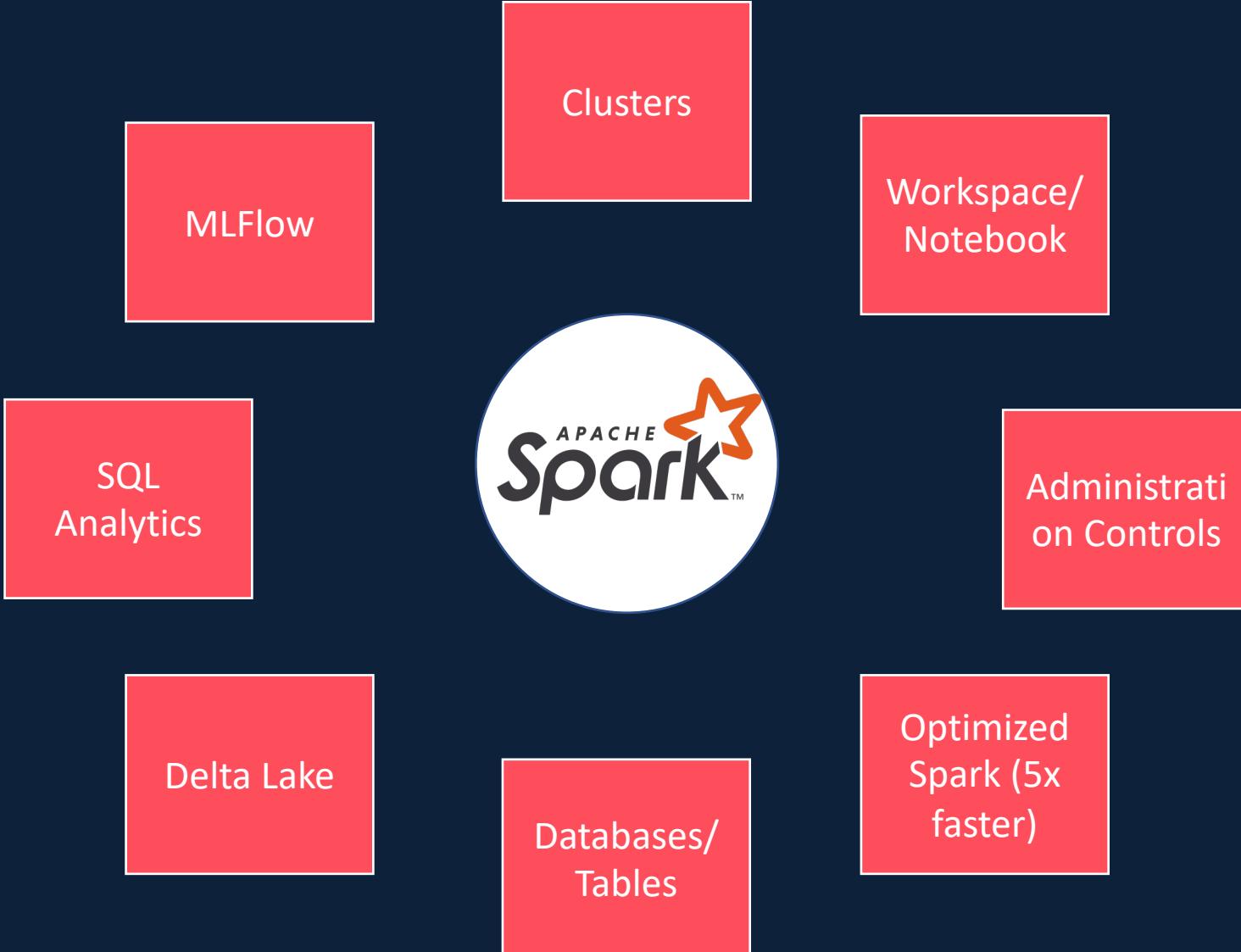
Distributed computing Platform

Unified engine which supports SQL, streaming, ML and graph processing

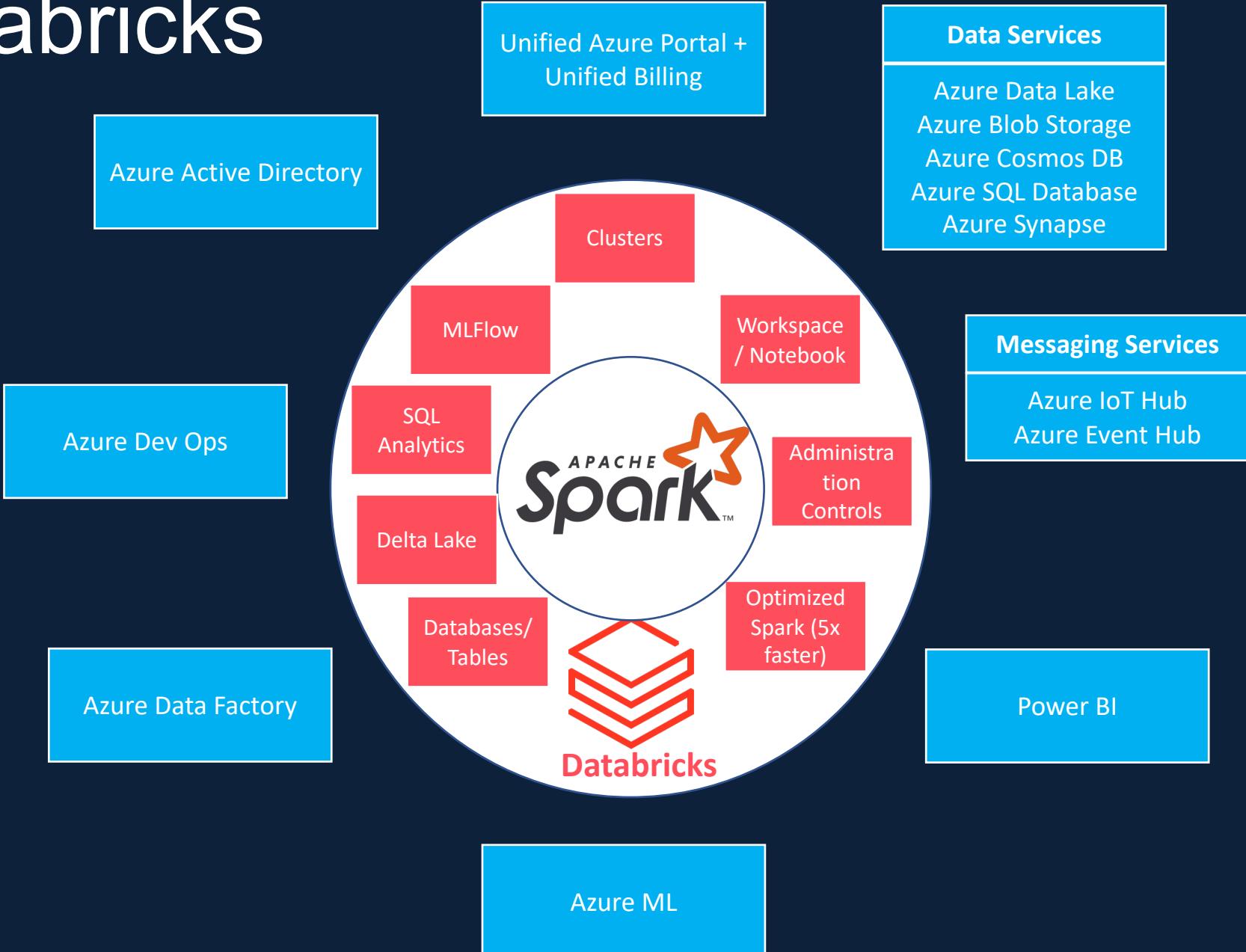
Apache Spark Architecture



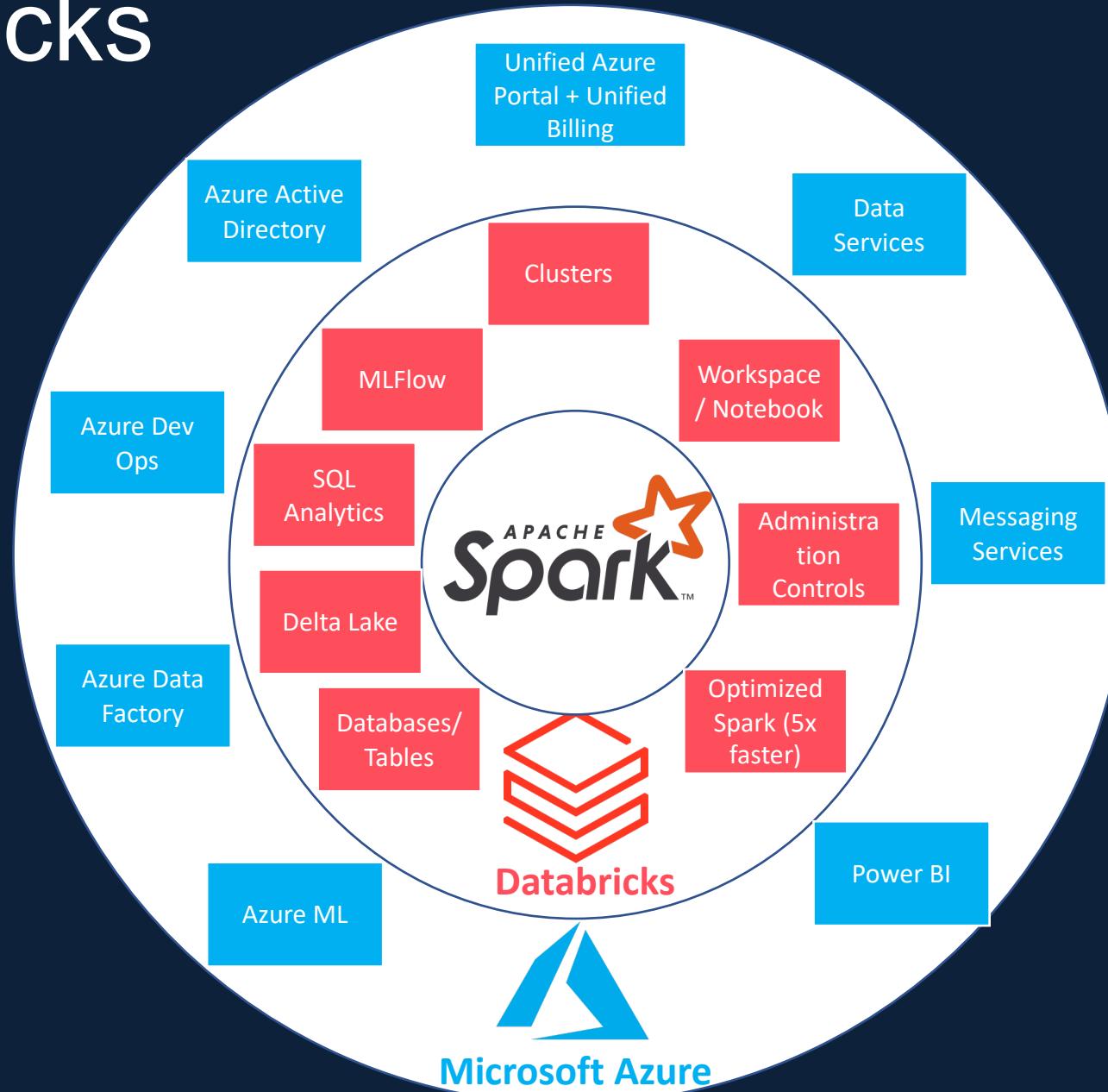
Databricks



Azure Databricks



Azure Databricks

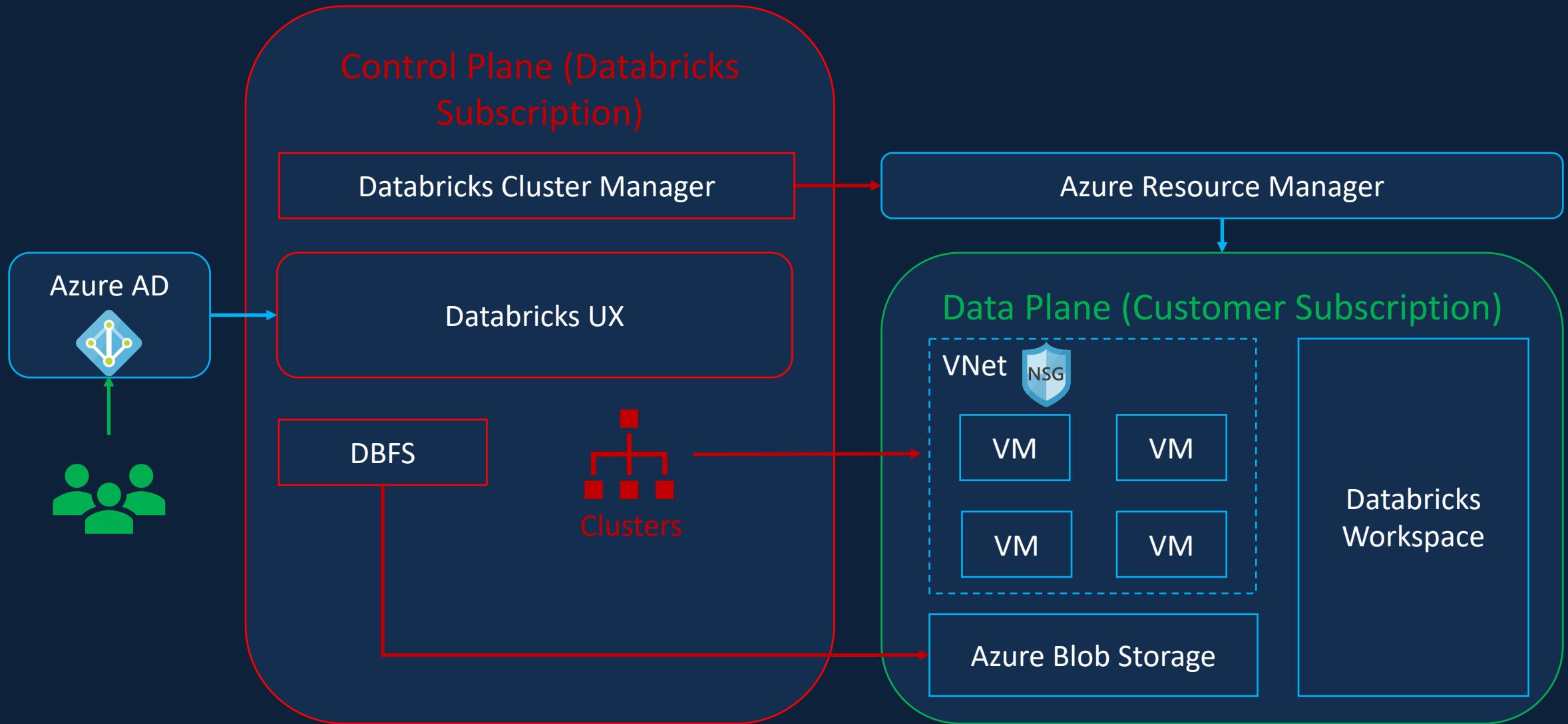


Databricks Service Overview

Creating Azure Databricks Service



Azure Databricks Architecture



Databricks Workspace Components

Notebooks

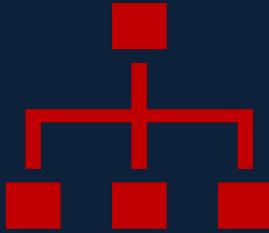
Data

Clusters

Jobs

Models

Databricks Clusters



What is Databricks Cluster

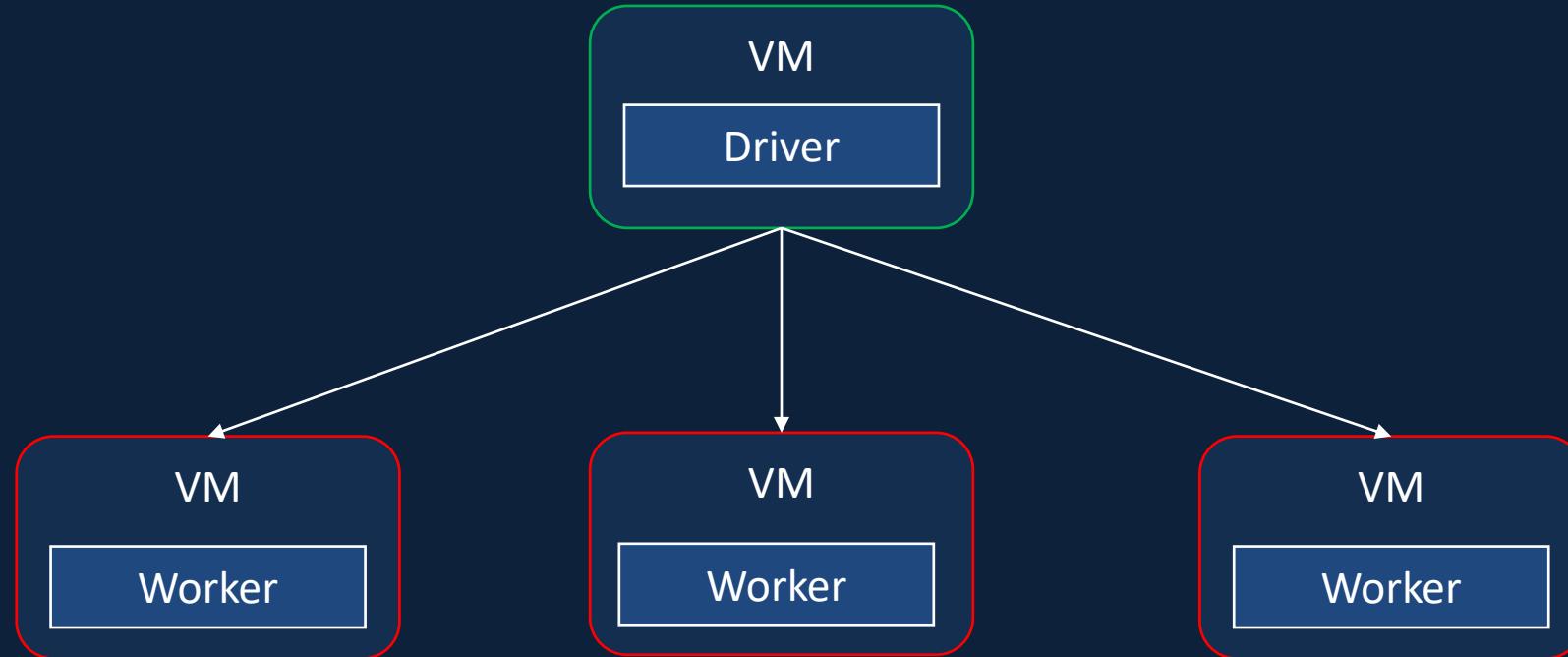
Cluster Types

Cluster Configuration

Creating a cluster

Cluster Pools

Databricks Cluster



Cluster Types

All Purpose	Job Cluster
Created manually	Created by Jobs
Persistent	Terminated at the end of the job
Suitable for interactive workloads	Suitable for automated workloads
Shared among many users	Isolated just for the job
Expensive to run	Cheaper to run

Cluster Configuration

Cluster Mode

Standard

Single User

No Process Isolation

No task preemption

Support for all DSL

For production workloads & adhoc development

High Concurrency

Multiple Users

Provides Process Isolation

Provides task preemption

Doesn't support Scala

For interactive analysis & adhoc development

Single Node

Single User

No Process Isolation

No task preemption

Support for all DSL

Light weight workload for ML & data analysis

Cluster Configuration

Cluster Mode

Databricks Runtime

Databricks Runtime

Spark

Scala, Java,
Python, R

Ubuntu
Libraries

GPU Libraries

Delta Lake

Other Databricks
Services

Databricks Runtime ML

Everything from
Databricks runtime

Popular ML Libraries (PyTorch, Keras, TensorFlow, XGBoost etc)

Databricks Runtime Genomics

Everything from
Databricks runtime

Popular open source genomic libraries (e.g. Glow, ADAM etc)
+ Popular genomic pipelines (e.g. DNASeq, RNASeq etc)

Databricks Runtime Light

Runtime option for only jobs not requiring advanced features

Cluster Configuration

Cluster Mode

Databricks Runtime

Auto Termination

Auto Termination

- Terminates the cluster after X minutes of inactivity
- Default value for Single Node and Standard clusters is 120 minutes
- High Concurrency clusters do not have a default auto termination set
- Users can specify a value between 10 and 10000 mins as the duration

Cluster Configuration

Cluster Mode

Databricks Runtime

Auto Termination

Auto Scaling

Auto Scaling

- User specifies the min and max work nodes
- Auto scales between min and max based on the workload
- Not recommended for streaming workloads

Cluster Configuration

Cluster Mode

Memory Optimized

Databricks Runtime

Compute Optimized

Auto Termination

Storage Optimized

Auto Scaling

General Purpose

Cluster VM Type/ Size

GPU Accelerated

Creating Databricks Cluster



Cluster Configuration

Cluster Mode

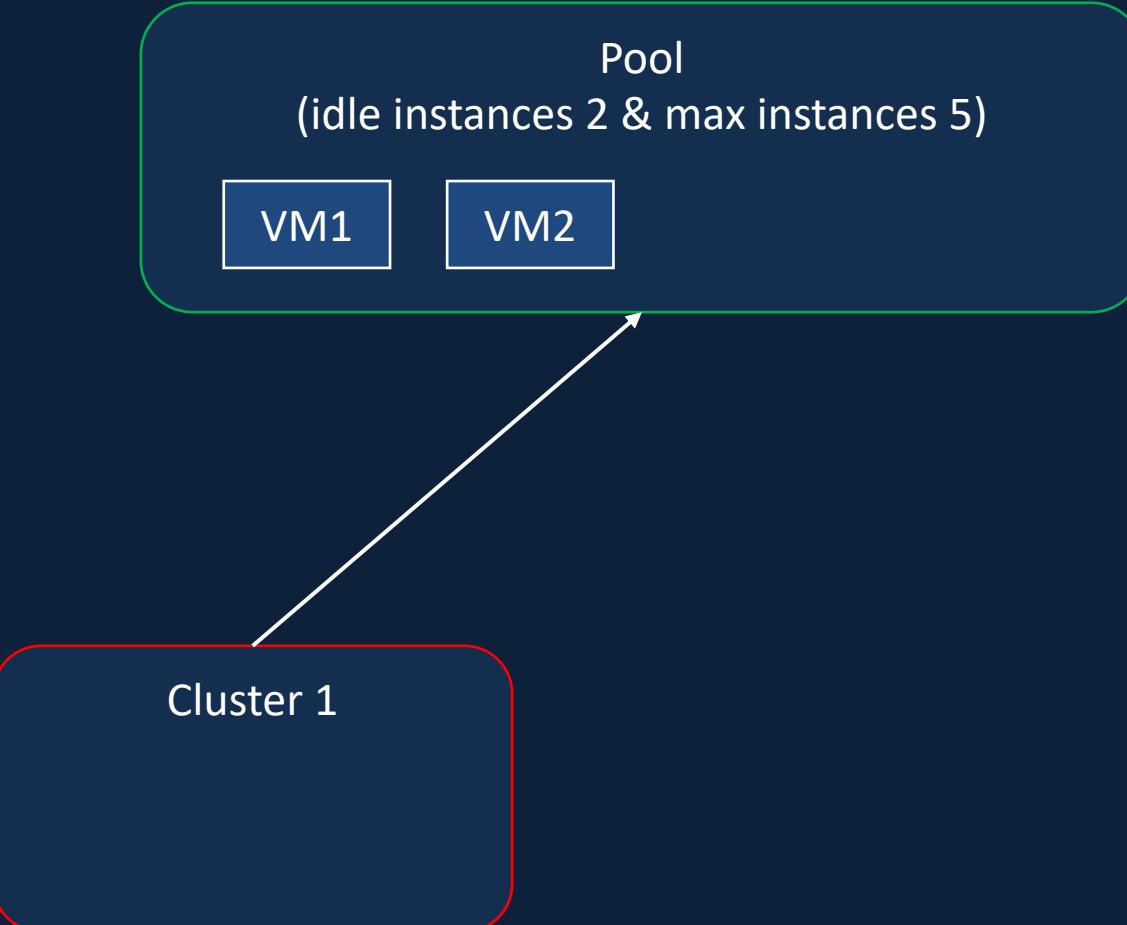
Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Pool



Cluster Configuration

Cluster Mode

Databricks Runtime

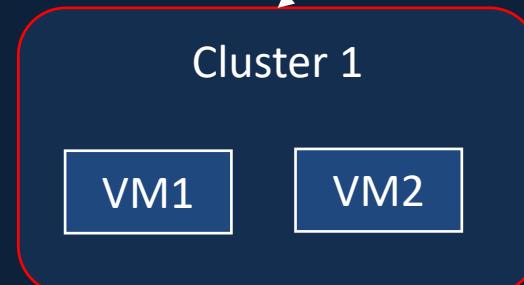
Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Pool

Pool
(idle instances 2 & max instances 5)



Cluster Configuration

Cluster Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Pool

Pool
(idle instances 2 & max instances 5)

VM3

VM4

Cluster 1

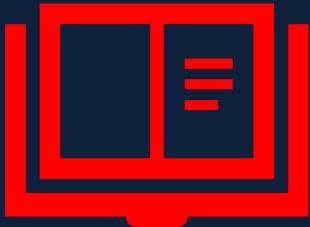
VM1

VM2

Creating Cluster Pool



Databricks Notebooks



What's a notebook

Creating a notebook

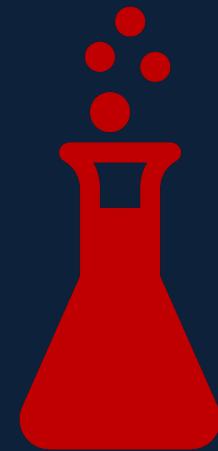
Magic Commands

Databricks Utilities

Creating Notebooks



Magic Commands



Databricks Utilities



File System Utilities

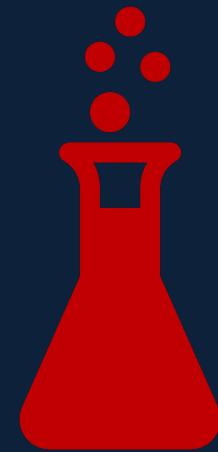
Notebook Workflow Utilities

Widget Utilities

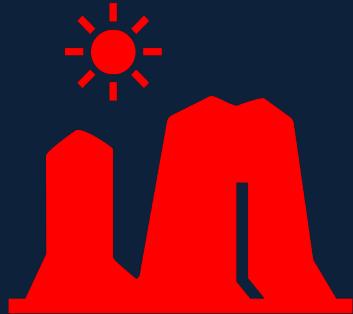
Secrets Utilities

Library Utilities

Databricks Utilities



Databricks Mounts



What is DBFS

What are Databricks mounts

Mount ADLS container to databricks

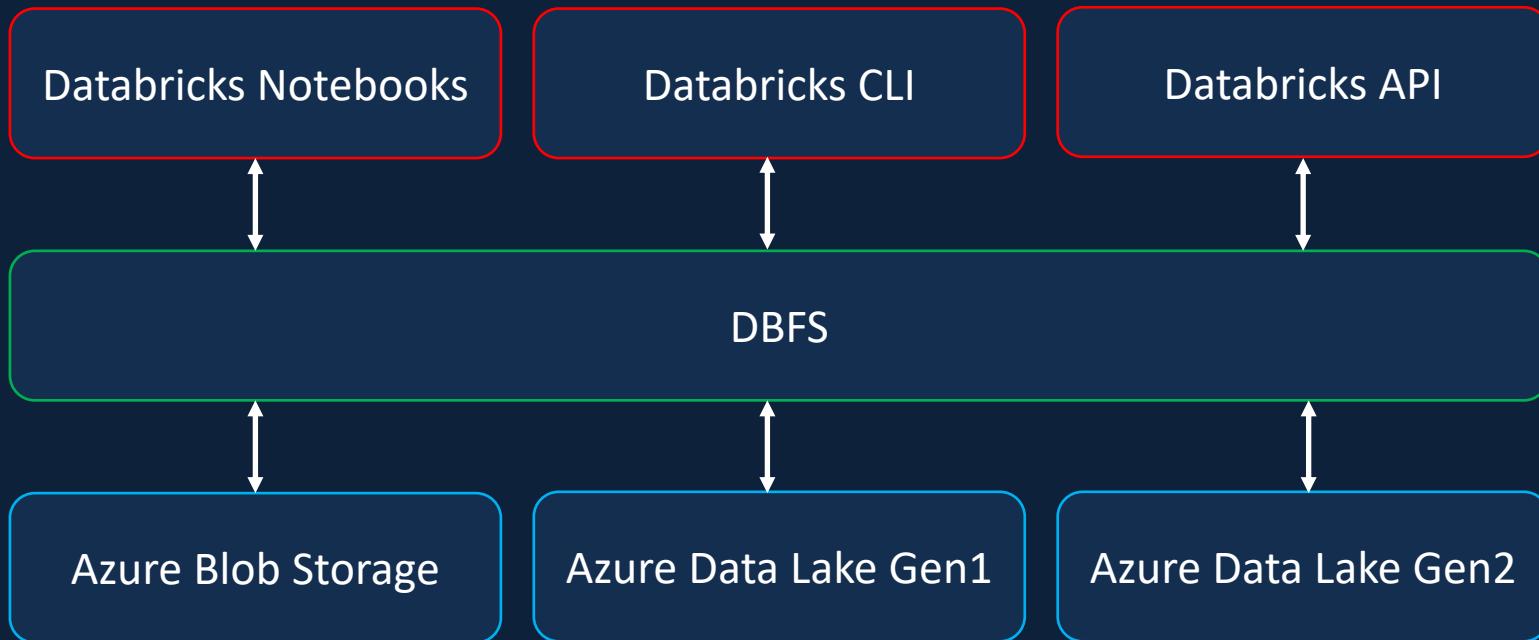
Create Service Principal

Create Azure Data Lake Storage Gen2

Creating Azure Key Vault

Creating Databricks secret scope

Databricks File System (DBFS)



Databricks File System (DBFS)

Benefits

Access data without requiring credentials

Access files using file semantics rather than storage URLs (e.g. /mnt/storage1)

Stores files to object storage (e.g. Azure Blob), so you get all the benefits from Azure

Databricks File System (DBFS)

DBFS Root

Backed up by Azure Blob Storage in Databricks created resource group

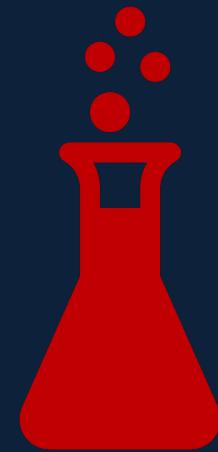
Default storage location, but not recommended for user data

Access storage via web UI (Special Folder FileStore)

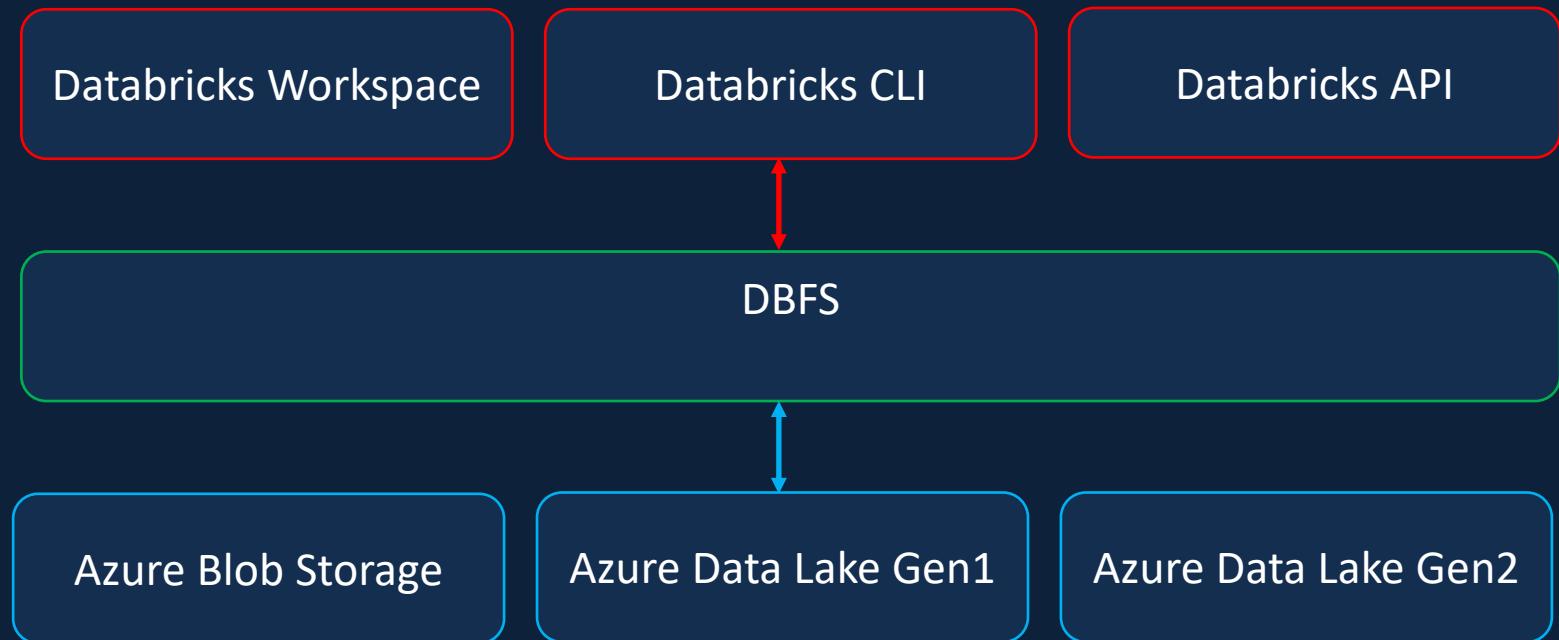
Query results are stored here (e.g. display commands)

Contains data and metadata for managed (non-external) tables

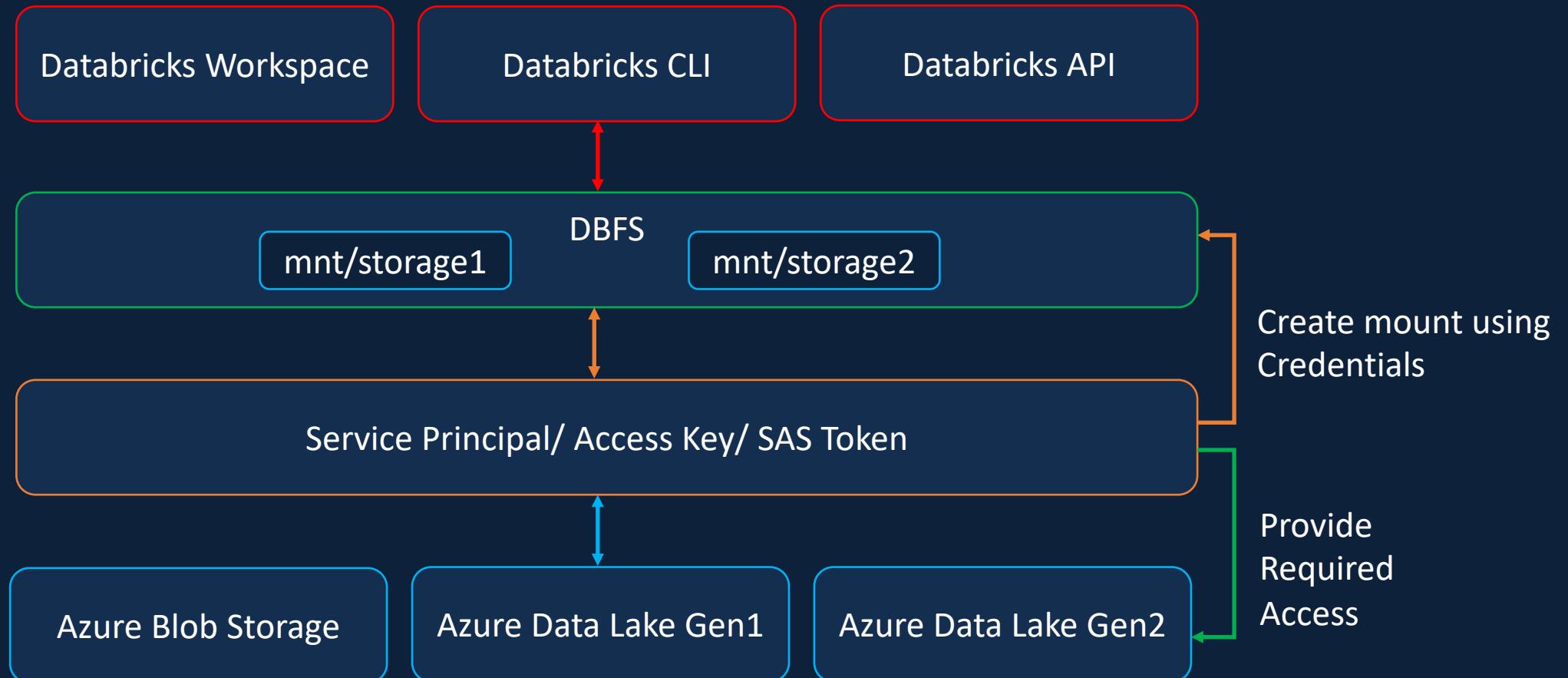
DBFS Root Demo



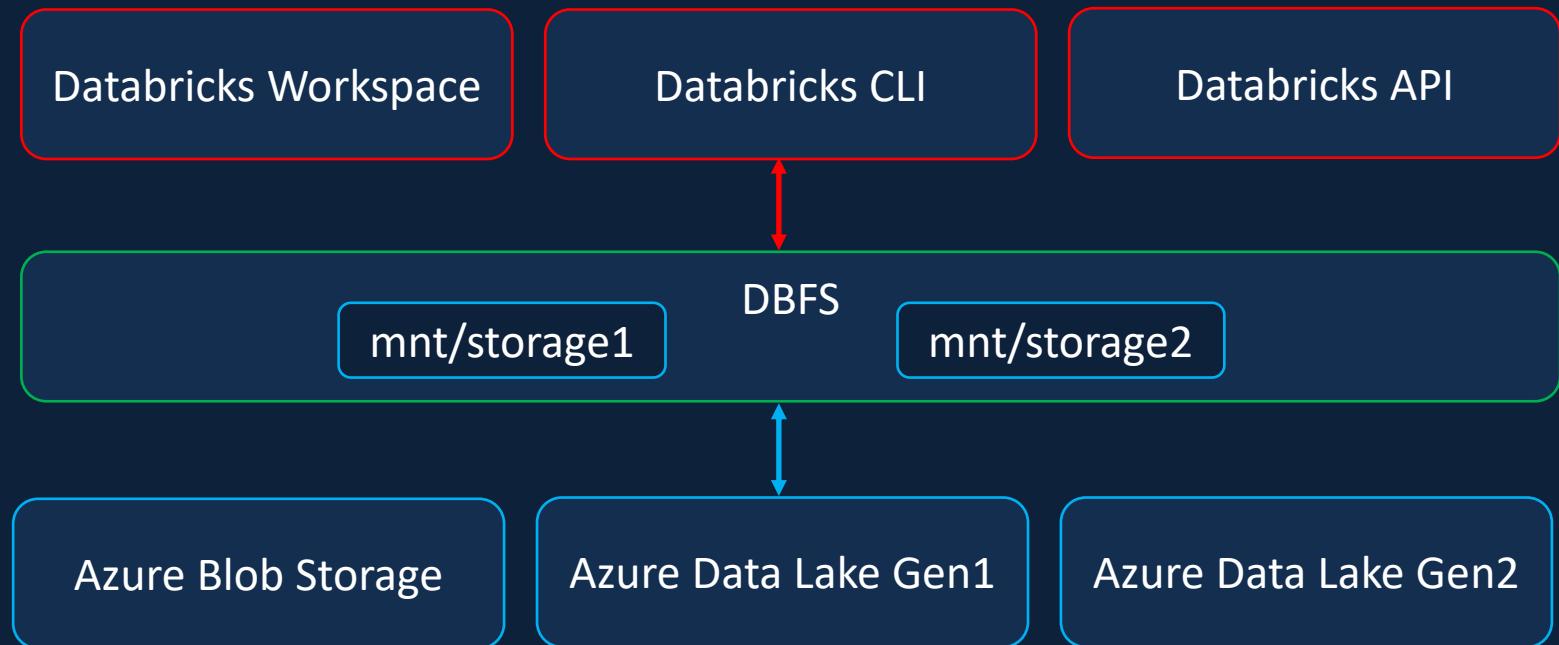
Mounting Azure Storage



Mounting Azure Storage



Mounting Azure Storage



Mounting Azure Data Lake Storage Gen2

Create Azure Storage Account (Data Lake Gen2)

Create Azure Service Principal

Provide required access to the service principal

Create the mount using the service principal

Creating Azure Storage Account (Data Lake Gen2)



Creating Azure Service Principal



Mounting Azure Data Lake Storage Gen2



Demo

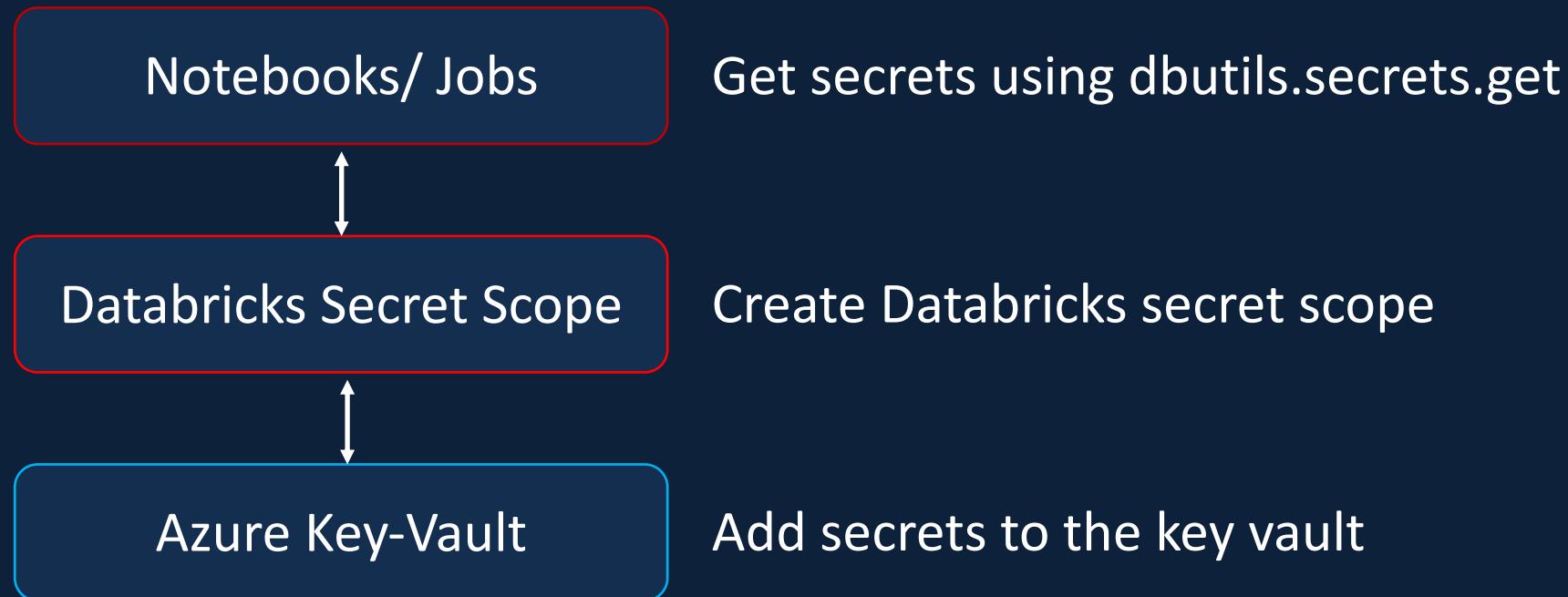
Secret Scope

Secret scopes help store the credentials securely and reference them in notebooks and jobs when required

Databricks backed Secret Scope

Azure Key-vault backed Secret Scope

Secret Scope



Demo

Project Overview



What is formula1

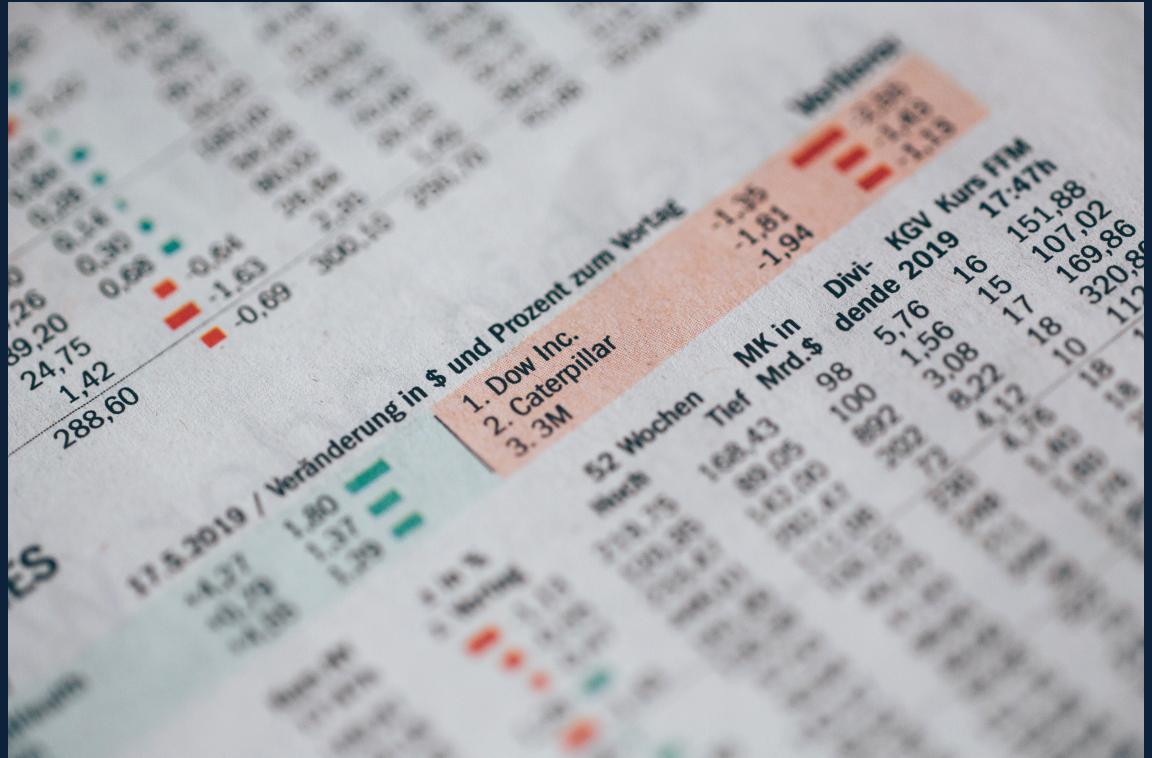
Formula1 data source & datasets

Prepare the data for the project

Project Requirements

Solution Architecture

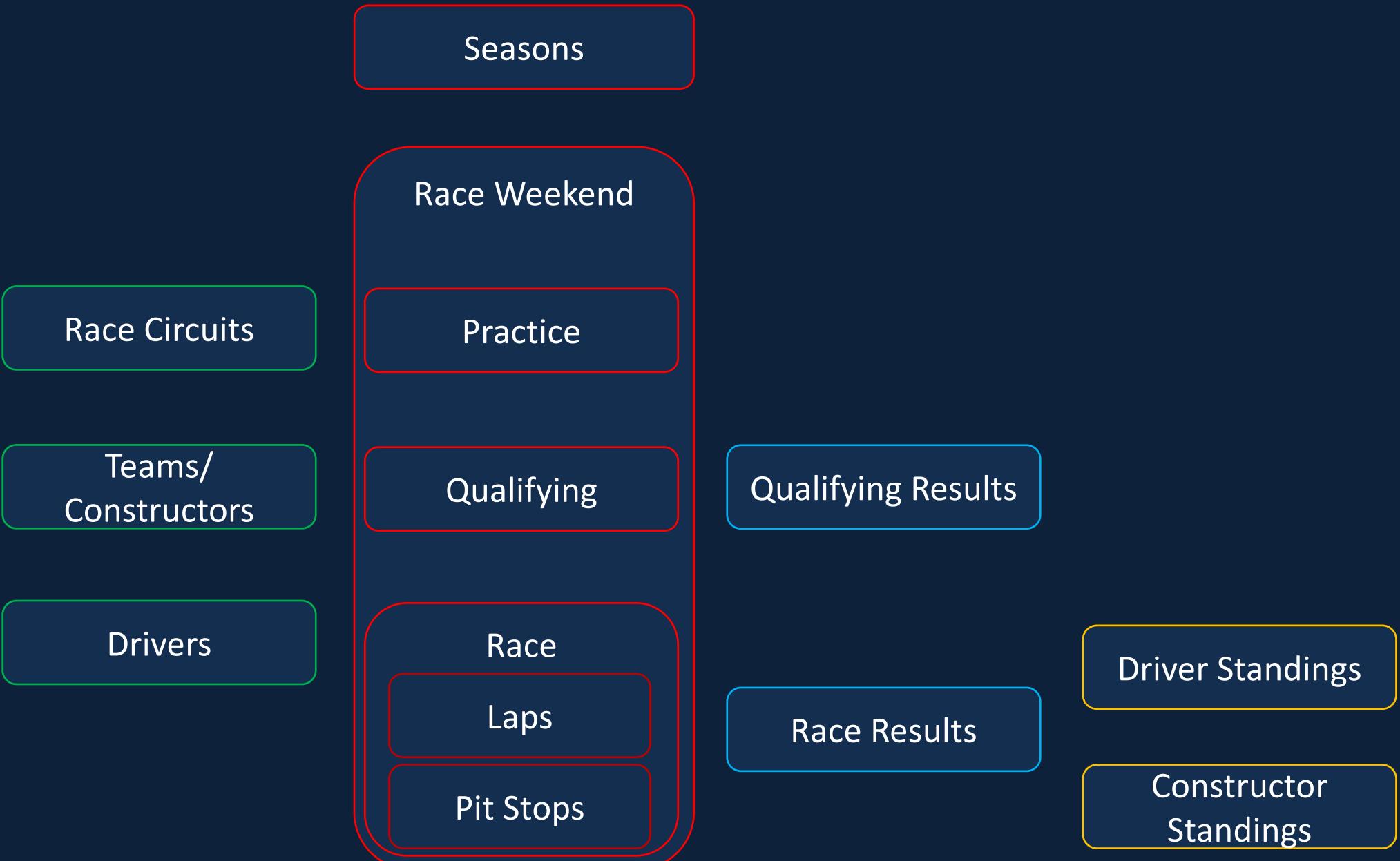
Data Overview





Formula1

Formula1 Overview



Formula1 Data Source

<http://ergast.com/mrd/>



 **Ergast Developer API** 

API Documentation

The Ergast Developer API is an experimental [web service](#) which provides a historical record of motor racing data for non-commercial purposes. Please read the [terms and conditions of use](#). The API provides data for the [Formula One](#) series, from the beginning of the world championships in 1950.

Non-programmers can query the database using the [manual interface](#) or [download the database tables in CSV format](#) for import into spreadsheets or analysis software.

If you have any comments or suggestions please post them on the [Feedback page](#). If you find any bugs or errors in the data please report them on the [Bug Reports page](#). Any enhancements to the API will be reported on the [News page](#). Example applications are shown in the [Application Gallery](#).

Overview

All API queries require a GET request using a URL of the form:

`http[s]://ergast.com/api/<series>/<season>/<round>/...`

where:

`<series>` should be set to "f1"
`<season>` is a 4 digit integer
`<round>` is a 1 or 2 digit integer

For queries concerning a whole season, or final standings, the round element may be omitted. For example:

`http://ergast.com/api/f1/2008/...`

For queries concerning the whole series both the round and the season elements may be omitted. For example:

`http://ergast.com/api/f1/...`

Index

- [API Documentation](#)
- [Season List](#)
- [Race Schedule](#)
- [Race Results](#)
- [Qualifying Results](#)
- [Standings](#)
- [Driver Information](#)
- [Constructor Information](#)
- [Circuit Information](#)
- [Finishing Status](#)
- [Lap Times](#)
- [Pit Stops](#)
- [Query Database](#)
- [Database Images](#)
- [Terms & Conditions](#)
- [Application Gallery](#)
- [Feedback](#)
- [FAQ](#)
- [Latest News](#)
- [Bug Reports](#)

Links

- [Contact Us](#)
- [Programmable Web](#)

Meta

- [Log in](#)
- [Entries RSS](#)
- [Comments RSS](#)
- [WordPress.org](#)

Search for:

Formula1 Data Source

Formula1 Data Files



Circuits	CSV
Races	CSV
Constructors	Single Line JSON
Drivers	Single Line Nested JSON
Results	Single Line JSON
PitStops	Multi Line JSON
LapTimes	Split CSV Files
Qualifying	Split Multi Line JSON Files

Import Raw Data to Data Lake



Project Requirements



Data Ingestion Requirements



Ingest All 8 files into the data lake

Ingested data must have the schema applied

Ingested data must have audit columns

Ingested data must be stored in columnar format (i.e., Parquet)

Must be able to analyze the ingested data via SQL

Ingestion logic must be able to handle incremental load

Data Transformation Requirements



Join the key information required for reporting to create a new table.

Join the key information required for Analysis to create a new table.

Transformed tables must have audit columns

Must be able to analyze the transformed data via SQL

Transformed data must be stored in columnar format (i.e., Parquet)

Transformation logic must be able to handle incremental load

Reporting Requirements



Driver Standings

Constructor Standings

Analysis Requirements



Dominant Drivers

Dominant Teams

Visualize the outputs

Create Databricks Dashboards

Scheduling Requirements



Scheduled to run every Sunday 10PM

Ability to monitor pipelines

Ability to re-run failed pipelines

Ability to set-up alerts on failures

Other Non-Functional Requirements



Ability to delete individual records

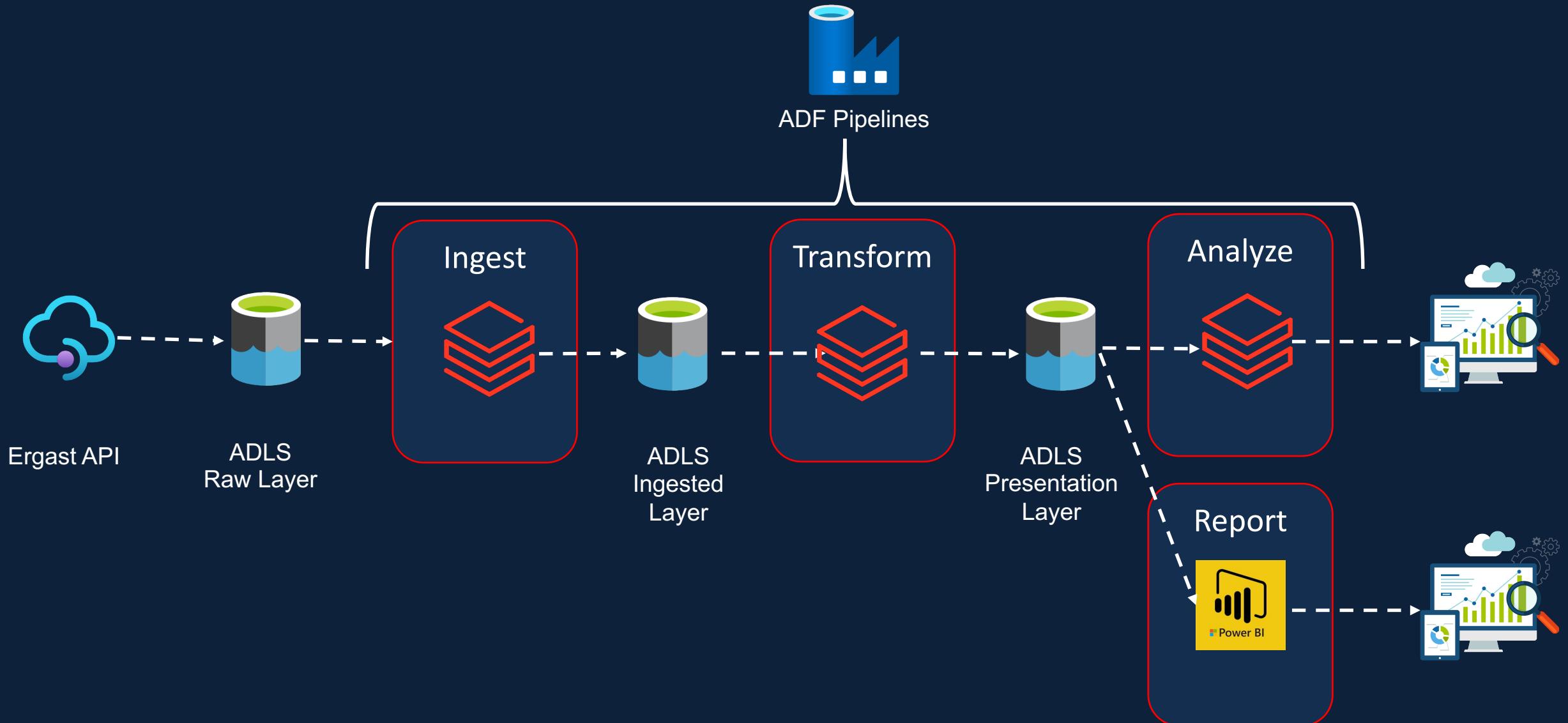
Ability to see history and time travel

Ability to roll back to a previous version

Blank

Solution Architecture Overview

Solution Architecture

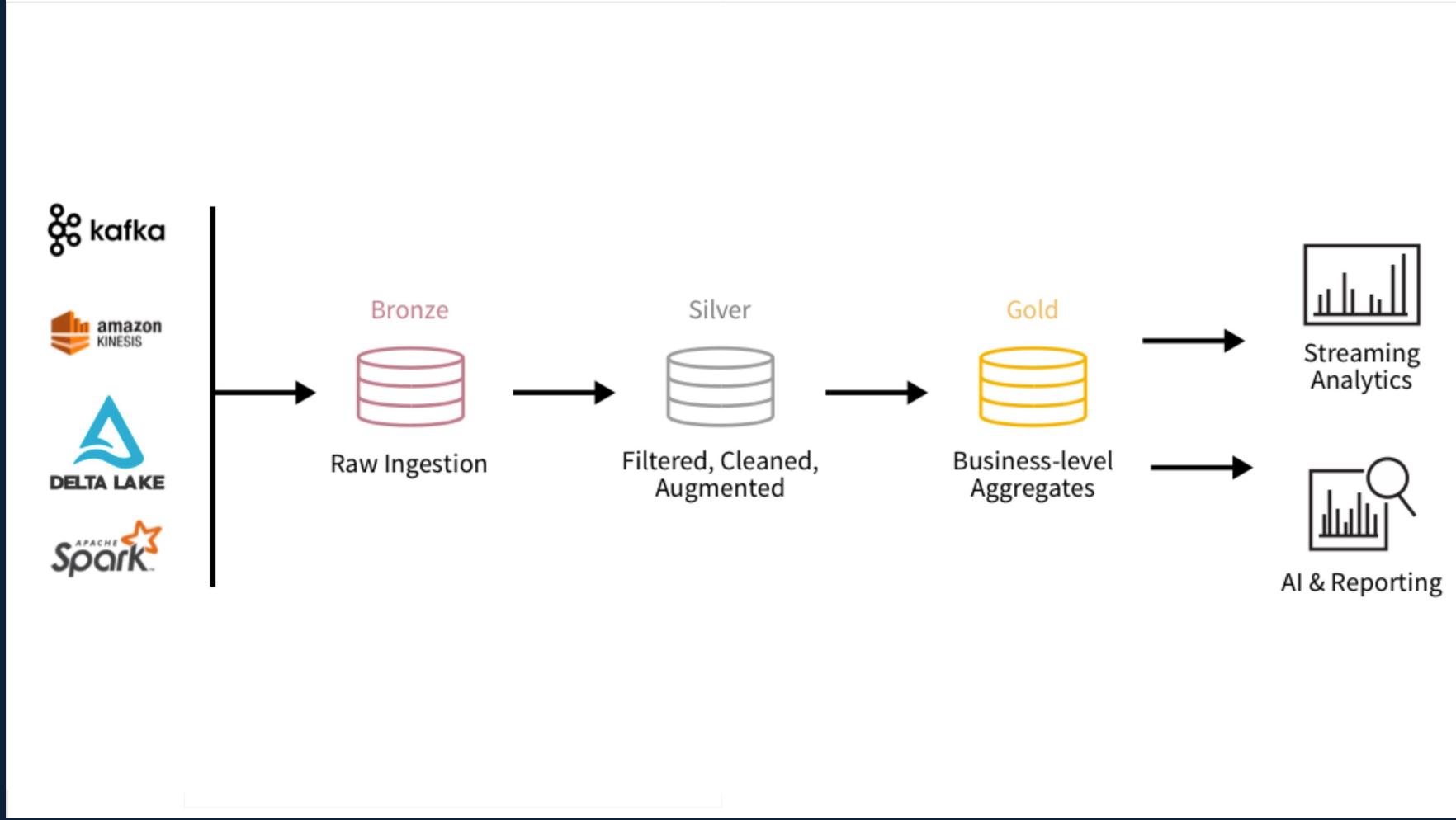


Azure Databricks Modern Analytics Architecture



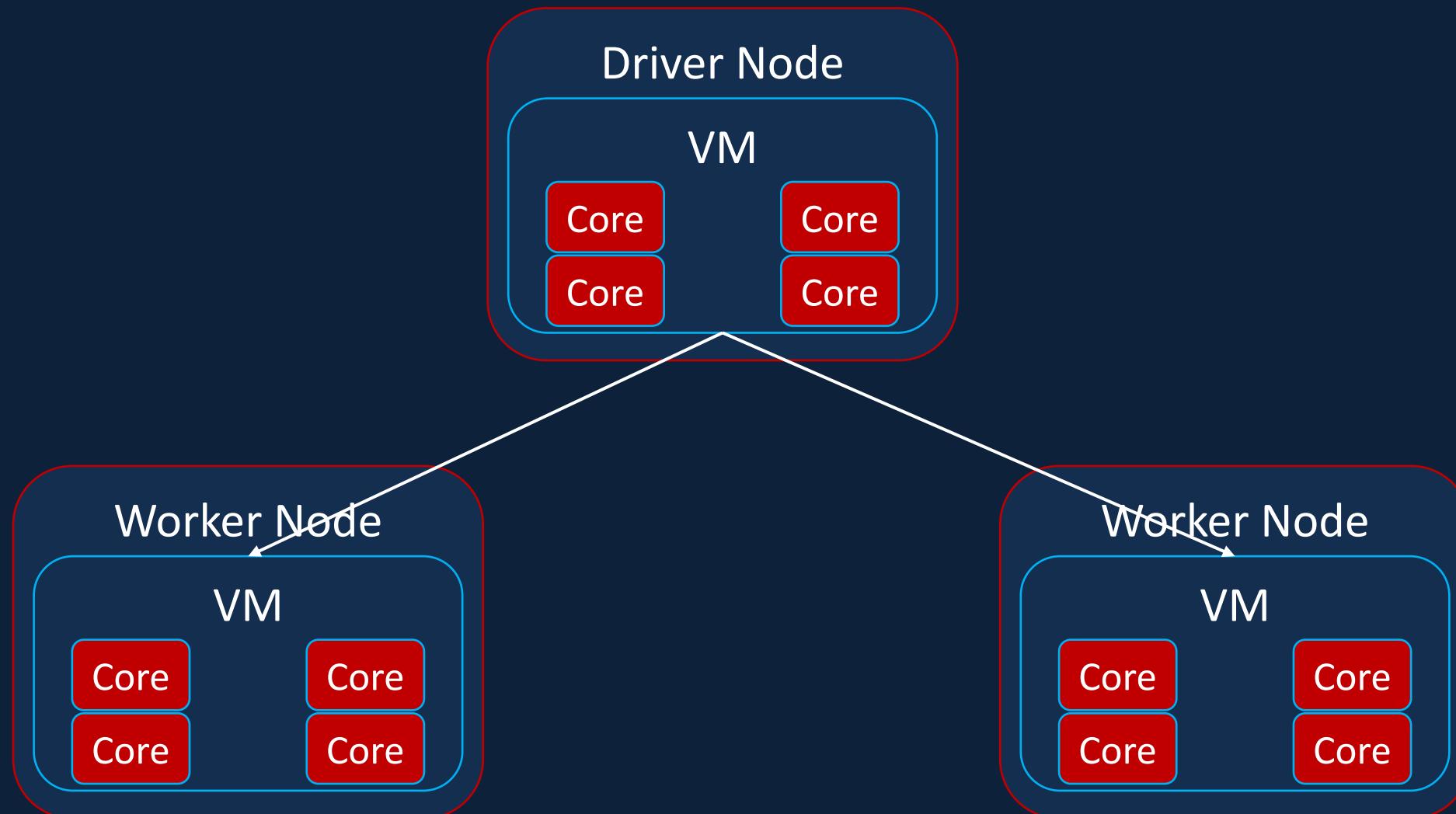
<https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture>

Databricks Architecture

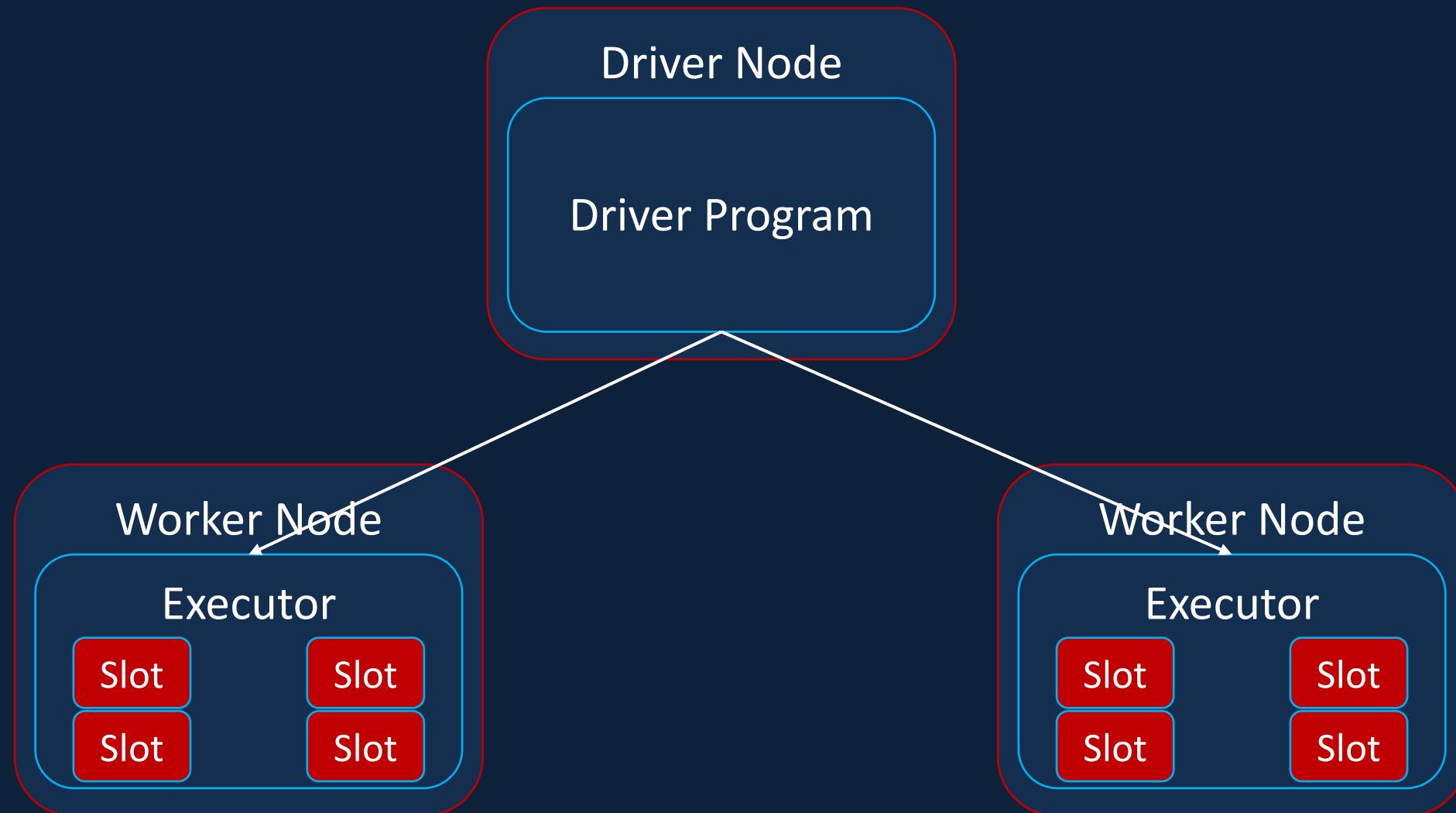


Spark Architecture

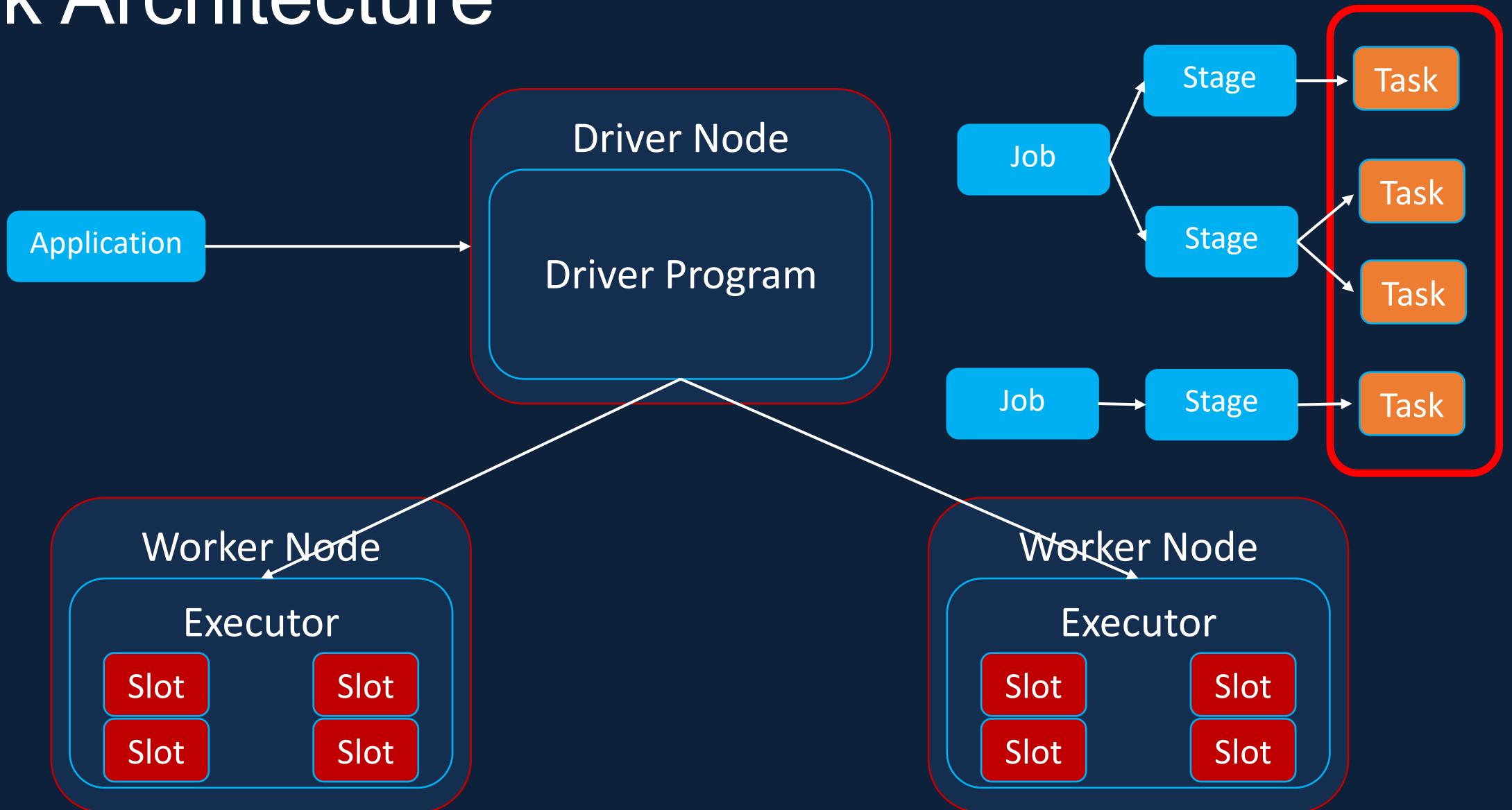
Spark Architecture



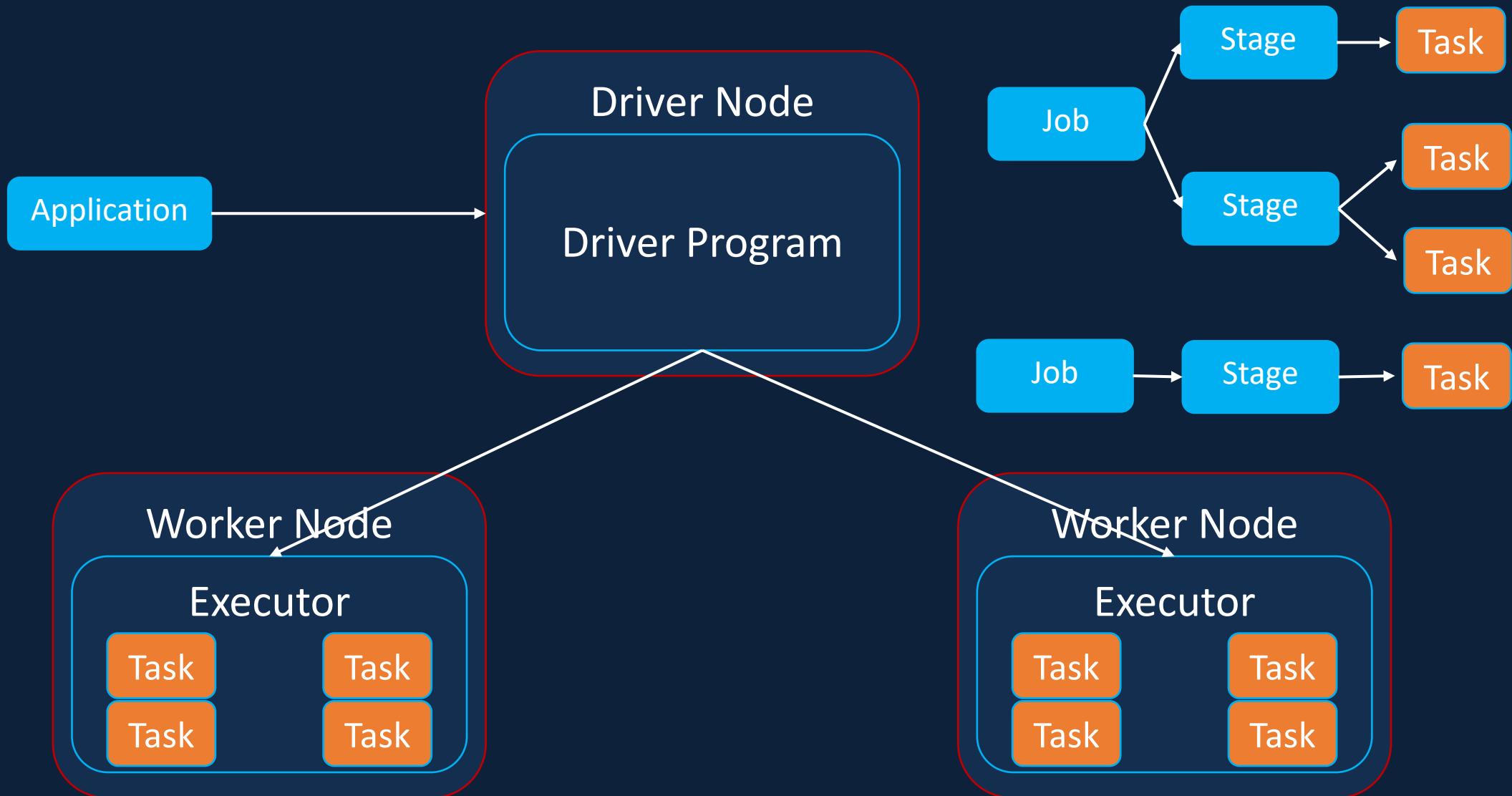
Spark Architecture



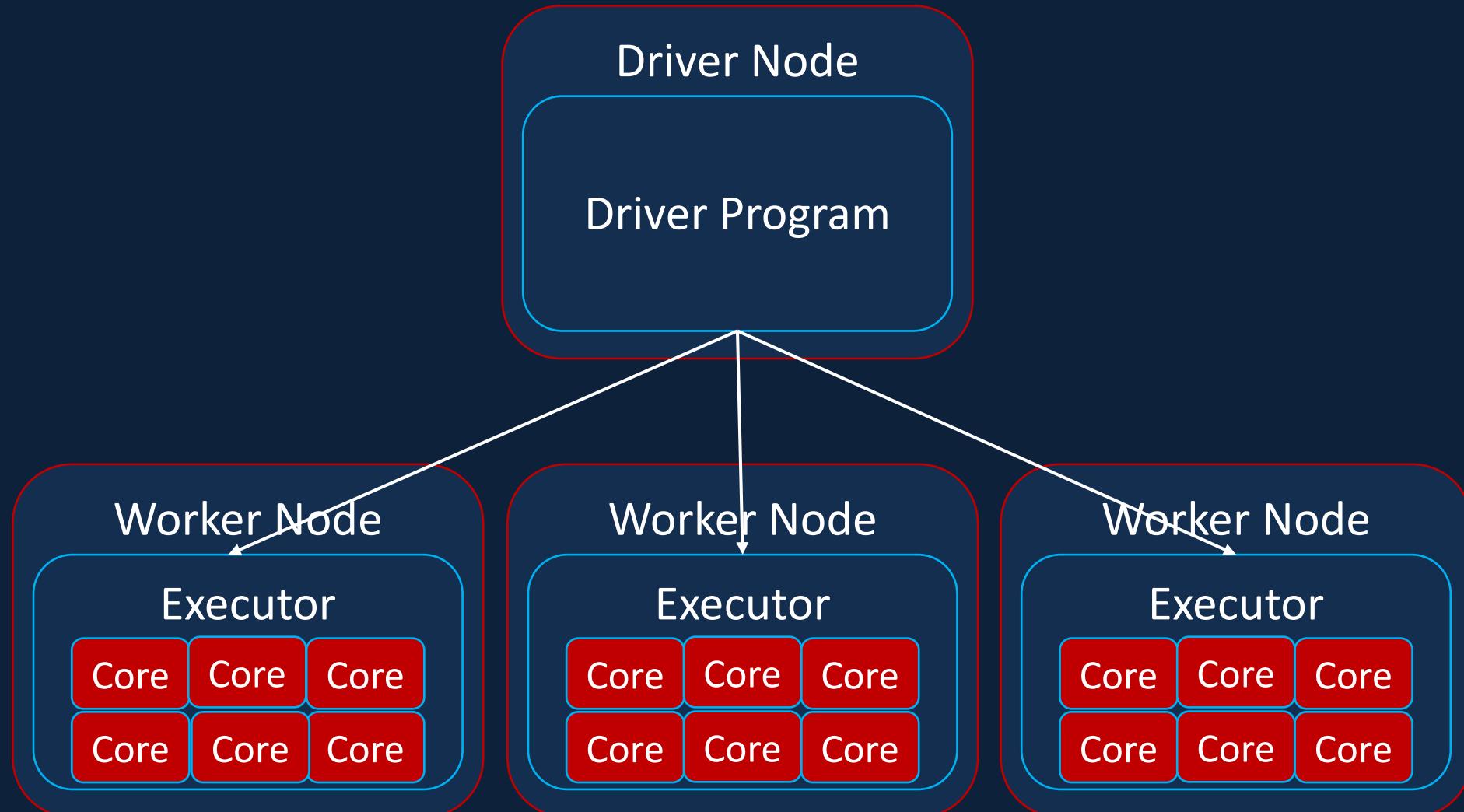
Spark Architecture



Spark Architecture



Spark Architecture – Cluster Scaling



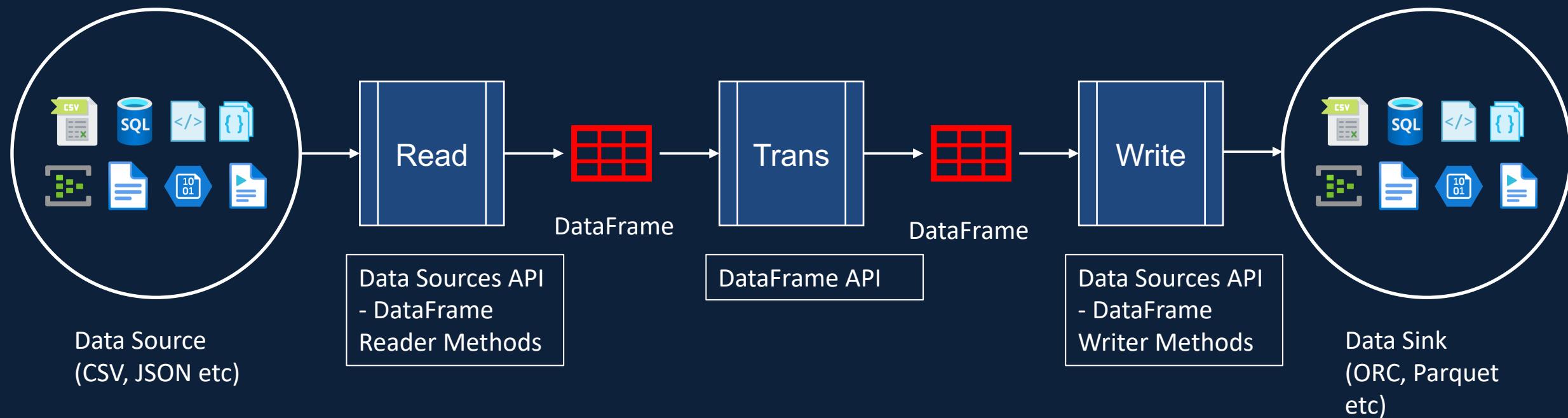
Spark DataFrame

Spark DataFrame

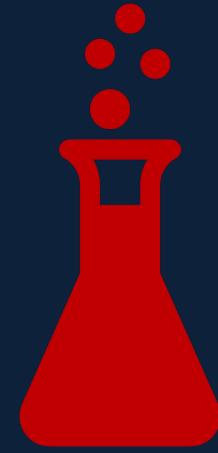
Driver	Team	Wins	Points
1 Max Verstappen	Red Bull	5	182
2 Lewis Hamilton	Mercedes	3	150
3 Sergio Perez	Red Bull	1	104
4 Lando Norris	McLaren	0	101
5 Valtteri Bottas	Mercedes	0	92
6 Charles Leclerc	Ferrari	0	62
7 Carlos Sainz Jr.	Ferrari	0	60
8 Daniel Ricciardo	McLaren	0	40
9 Pierre Gasly	AlphaTauri	0	39
10 Sebastian Vettel	Aston Martin	0	30
11 Fernando Alonso	Alpine	0	20
12 Lance Stroll	Aston Martin	0	14

Source: <https://www.bbc.co.uk/sport/formula1/drivers-world-championship/standings>

Spark DataFrame



Spark Documentation



Data Ingestion Overview

Data Ingestion Requirements



Ingest All 8 files into the data lake

Ingested data must have the schema applied

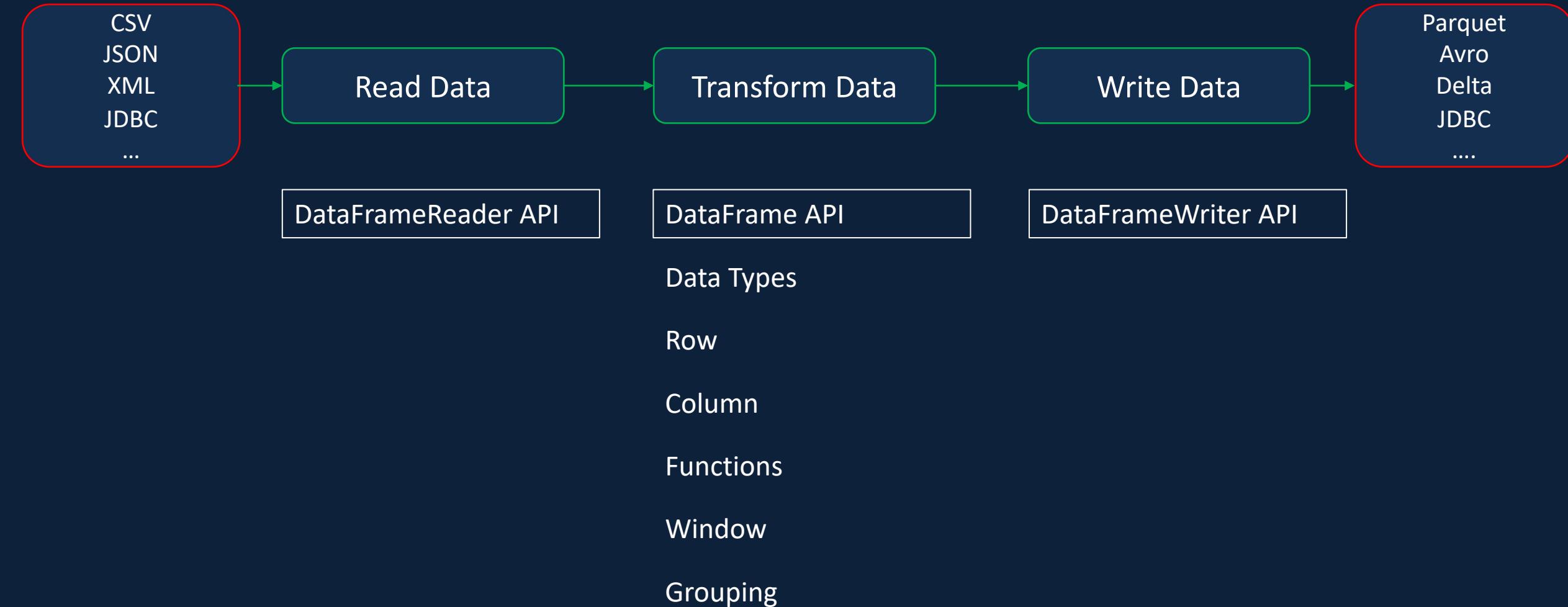
Ingested data must have audit columns

Ingested data must be stored in columnar format (i.e., Parquet)

Must be able to analyze the ingested data via SQL

Ingestion logic must be able to handle incremental load

Data Ingestion Overview



Data Ingestion Overview

File Name	File Type	Assignment/ Class work
Circuits	CSV	Class work
Races	CSV	Assignment
Constructors	Single Line JSON	Class work
Results	Single Line JSON	Assignment
Drivers	Single Line Nested JSON	Class work
PitStops	Multi Line JSON	Class work
LapTimes	Split CSV Files	Class work
Qualifying	Split Multi Line JSON Files	Assignment

Data Ingestion - Circuits



Column Name
circuitId
circuitRef
name
location
country
lat
long
alt
url (dropped)

Column Name	Data Type
circuit_id (Renamed)	Integer
circuit_ref (Renamed)	String
name	String
location	String
country	String
latitude (Renamed)	Double
longitude (Renamed)	Double
altitude (Renamed)	Integer
ingestion_date (new)	Timestamp

Data Ingestion – Races (Assignment)



Column Name
raceId
year
round
circuitId
name
date
time
url (dropped)

Column Name	Data Type
race_id (Renamed)	Integer
race_year (Renamed)	Integer
round	Integer
circuit_id (Renamed)	Integer
name	String
race_timestamp (Transformed)	Timestamp
ingestion_date (new)	Timestamp

`withColumn('race_timestamp', to_timestamp(concat(col('date'), lit(' '), col('time'))), 'yyyy-MM-dd HH:mm:ss'))`

Data Ingestion – Races (Partition By)



Column Name
raceId
year
round
circuitId
name
date
time
url (dropped)

Column Name
race_id (Renamed)
race_year (Renamed)
round
circuit_id (Renamed)
name
race_timestamp (Transformed)
ingestion_date (new)

Partition By
race_year

withColumn('race_timestamp', to_timestamp(concat(col('date'), lit(' '), col('time'))), 'yyyy-MM-dd HH:mm:ss'))

Data Ingestion - Constructors



Column Name
constructorId
constructorRef
name
nationality
url (dropped)

Column Name
constructor_id (Renamed)
constructor_ref (Renamed)
name
nationality
ingestion_date (new)

Data Ingestion - Drivers



Column Name
driverId
driverRef
number
code
name.forename
name.surname
dob
nationality
url (dropped)

Column Name
driver_id (Renamed)
driver_ref (Renamed)
number
code
name(transformed)
dob
nationality
ingestion_date (new)

Data Ingestion – Results (Assignment)

JSON

Read Data

Transform Data

Write Data

Parquet

Column Name

resultId

raceId

driverId

constructorId

number

grid

position

positionText

positionOrder

points

laps

time

milliseconds

fastestLap

rank

fastestLapTime

FastestLapSpeed

statusId (dropped)

Transform Data

Write Data

Column Name

result_id (renamed)

race_id (renamed)

driver_id (renamed)

constructor_id (renamed)

number

grid

position

position_text (renamed)

position_order (renamed)

points

laps

time

milliseconds

fastest_lap (renamed)

rank

fastest_lap_time (renamed)

fastest_lap_speed (renamed)

Ingestion_date (new)

Partition By
race_id

Data Ingestion - Pitstops



Column Name
raceId
driverId
stop
lap
time
duration
milliseconds

Column Name
race_id (renamed)
driver_id (renamed)
stop
lap
time
duration
milliseconds
Ingestion_date (new)

Data Ingestion - Laptimes



Column Name
raceId
driverId
lap
position
time
milliseconds

Column Name
race_id (renamed)
driver_id (renamed)
lap
position
time
milliseconds
Ingestion_date (new)

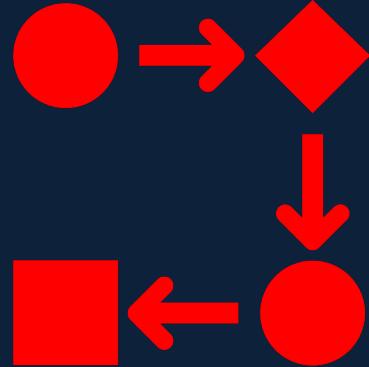
Data Ingestion – Qualifying (Assignment)



Column Name
qualifyingId
raceId
driverId
constructorId
number
position
q1
q2
q3

Column Name
qualifying_id(renamed)
race_id(renamed)
driver_id(renamed)
constructor_id(renamed)
number
position
q1
q2
q3
Ingestion_date(new)

Databricks Workflows



Include notebook

Defining notebook parameters

Notebook workflow

Databricks Jobs

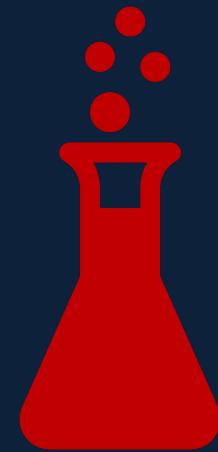
Include notebook (%run)



Passing Parameters (widgets)



Notebook Workflow



Databricks Jobs



Filter/ Join Transformations



Filter Transformation

Join Transformations

Apply Transformations to F1 Project

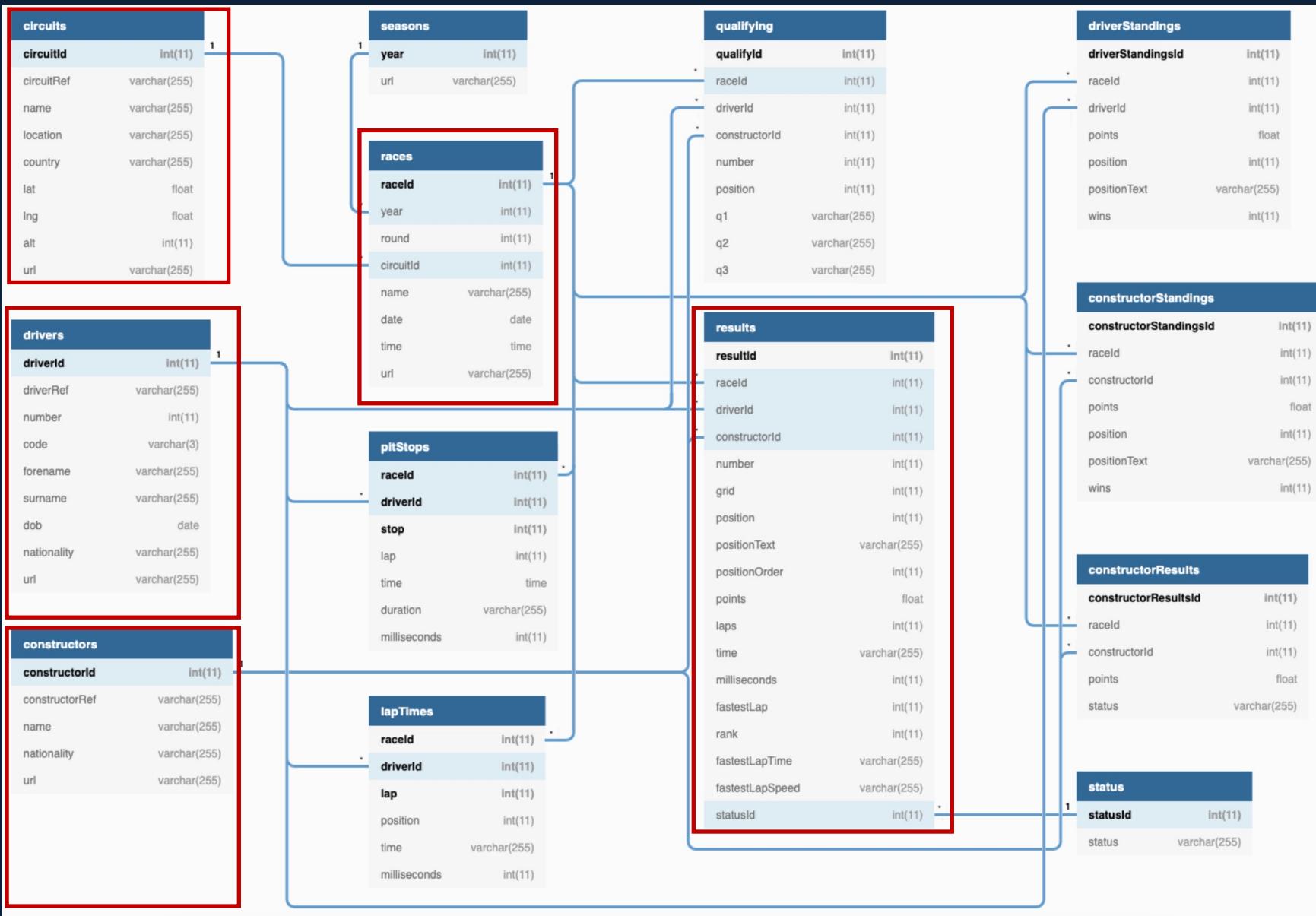
Filter Transformations

Join Transformations

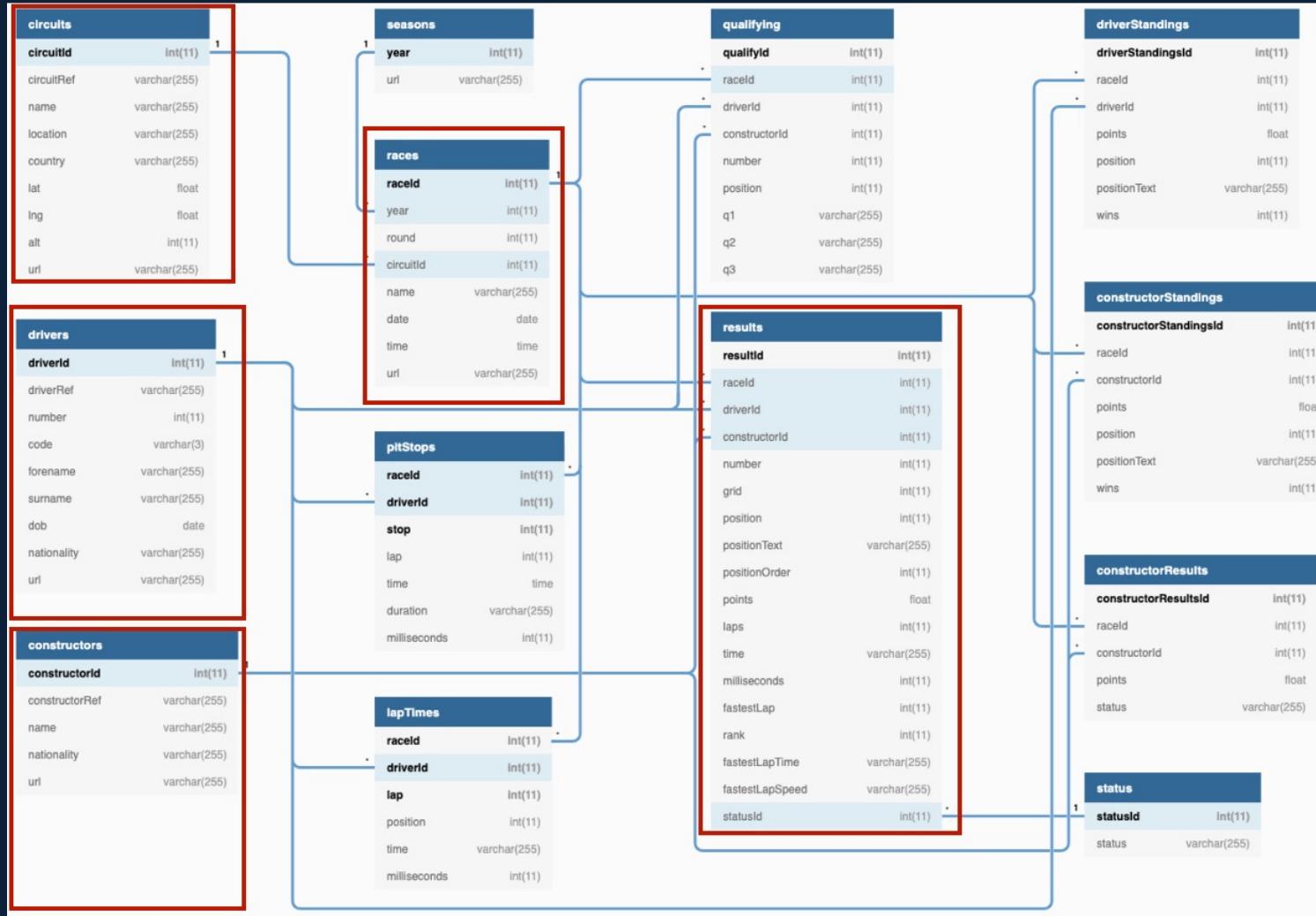
Join Transformation

Race Results

Join –Race Results



Join -Race Results

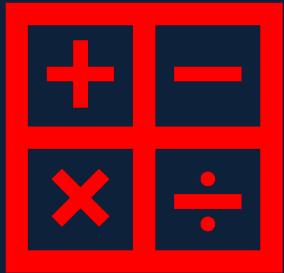


Column Name	Source
race_year	races
race_name	races
race_date	races
circuit_location	circuits
driver_name	drivers
driver_number	drivers
driver_nationality	drivers
team	constructors
grid	results
fastest_lap	results
race_time	results
points	results
created_date	current_timestamp

Set-up Environment Presentation Layer



Aggregations



Simple Aggregations

Grouped Aggregations

Window Functions

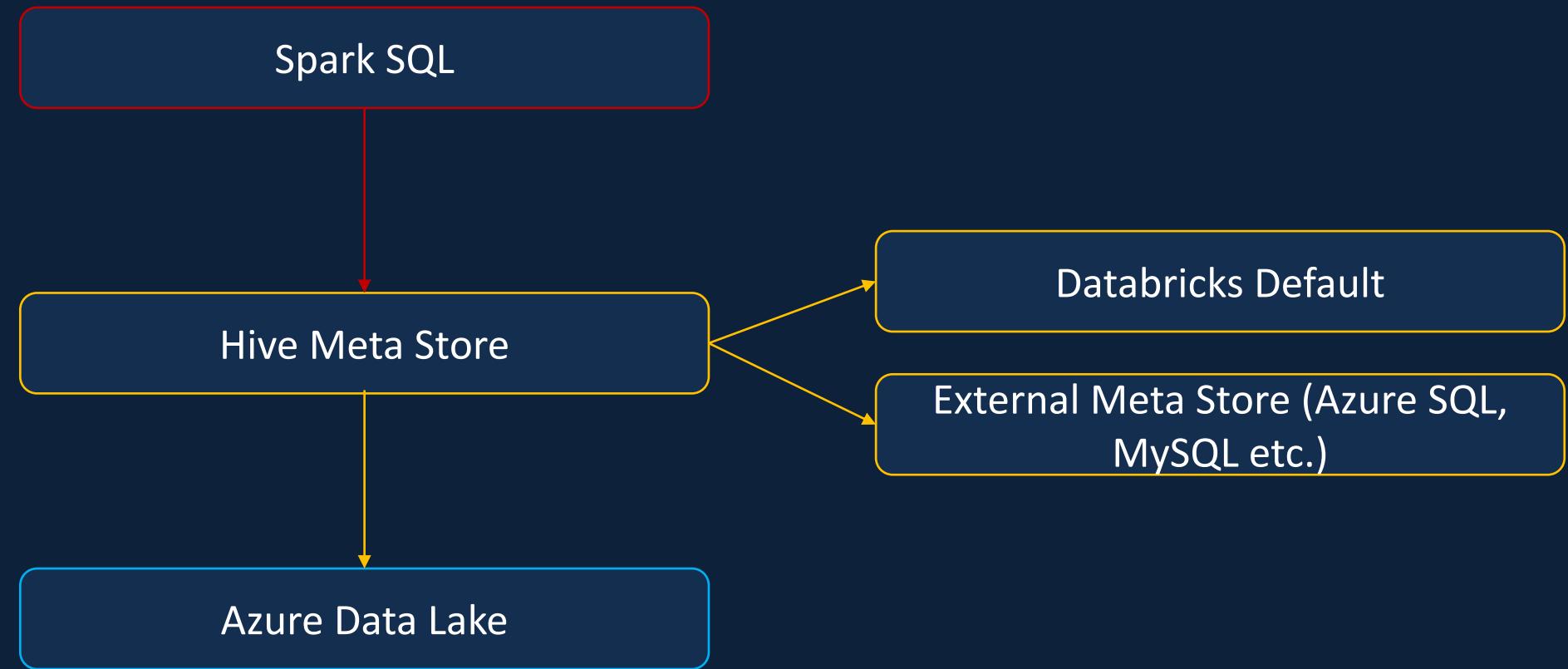
Apply Aggregations to F1 Project

Built-in Aggregate Functions

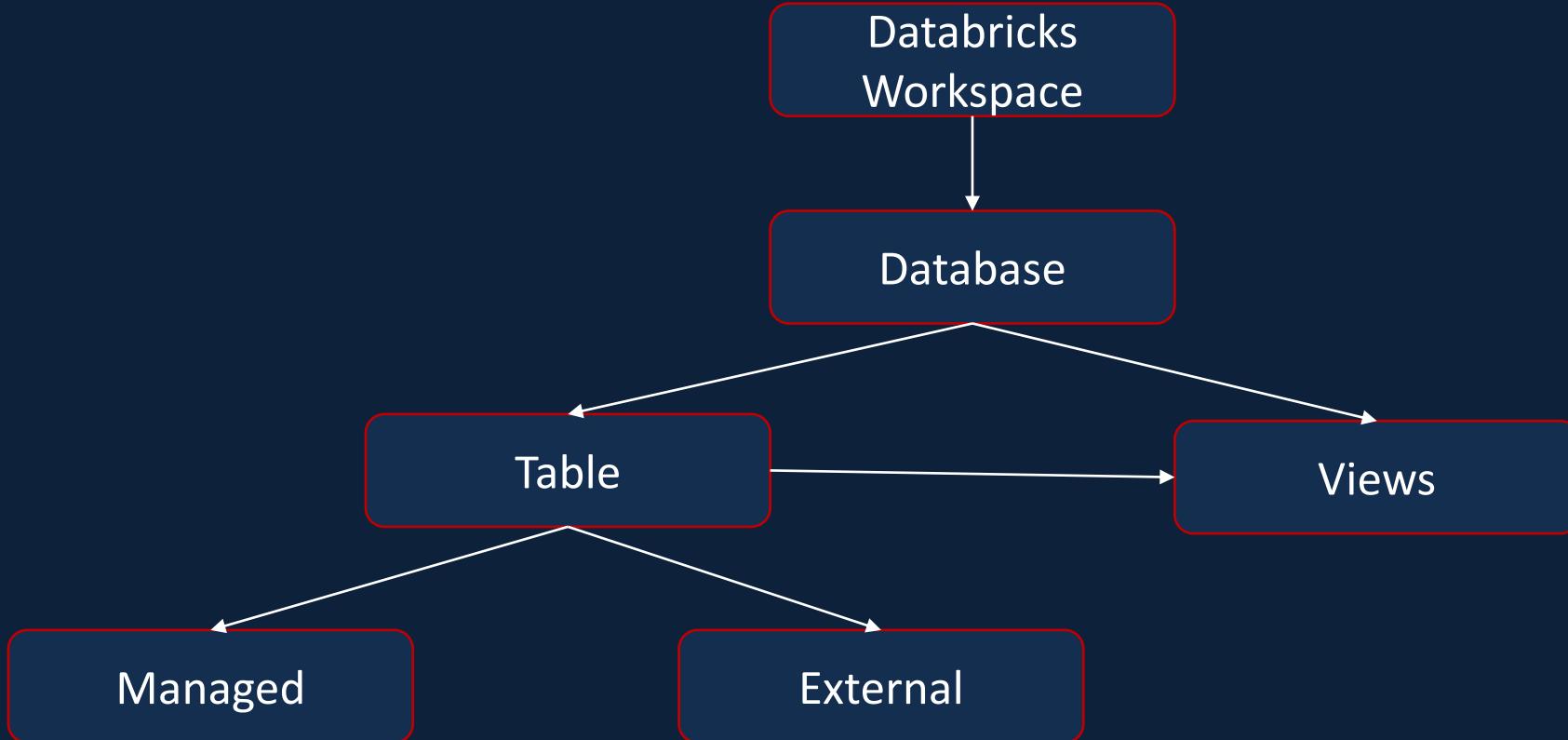
Group By

Databases/ Tables/ Views

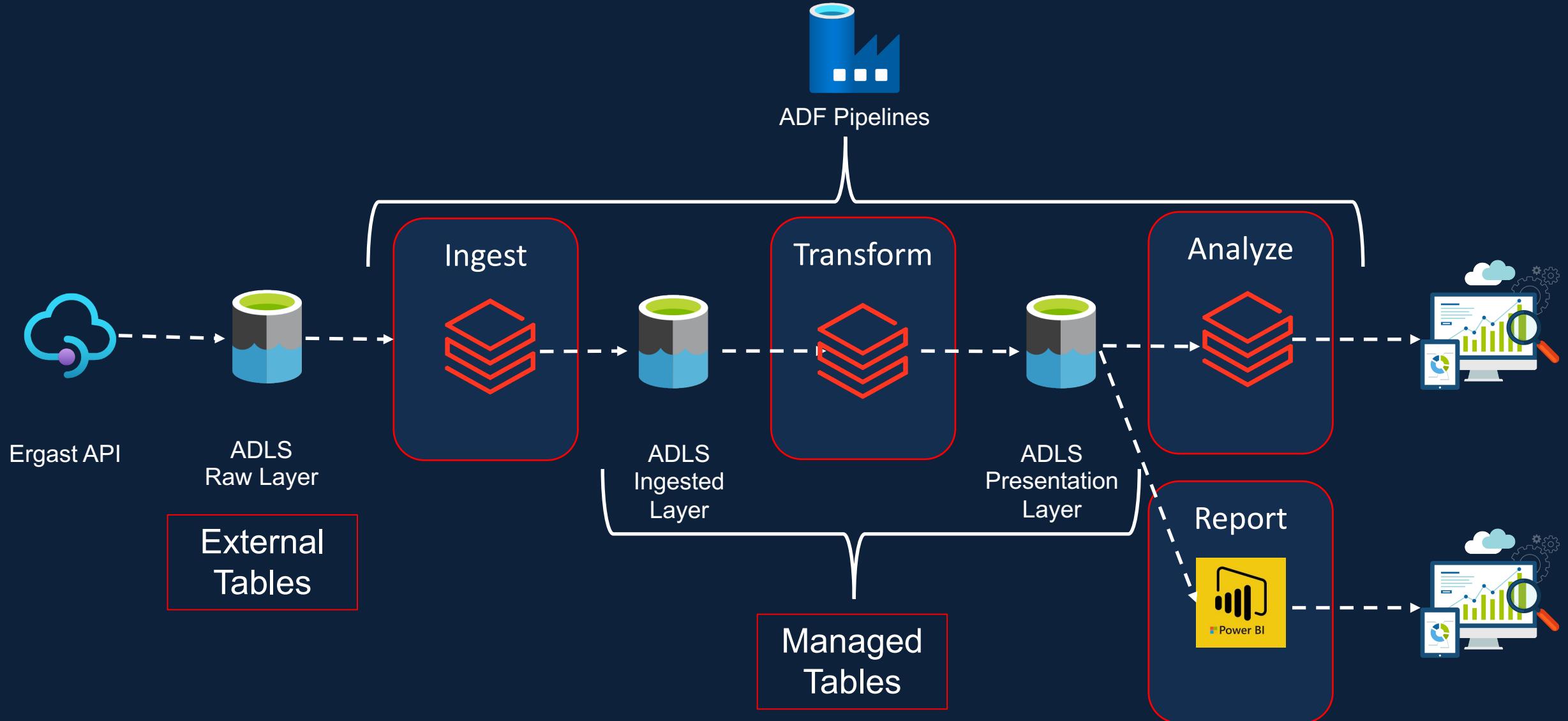
Hive Meta Store



Spark Databases/ Tables/ Views



Managed/ External Tables



Spark SQL Introduction



SQL Basics

Simple Functions

Aggregate Functions

Window Functions

Joins

Dominant Drivers/ Teams Analysis

Create a table with the data required

Granularity of the data – race_year, driver, team

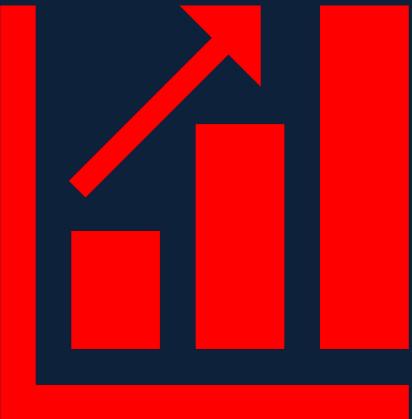
Rank the dominant drivers of all time/ last decade etc

Rank the dominant teams of all time/ last decade etc

Visualization of dominant drivers

Visualization of dominant teams

Incremental Load



Data Loading Patterns

F1 Project Load Pattern

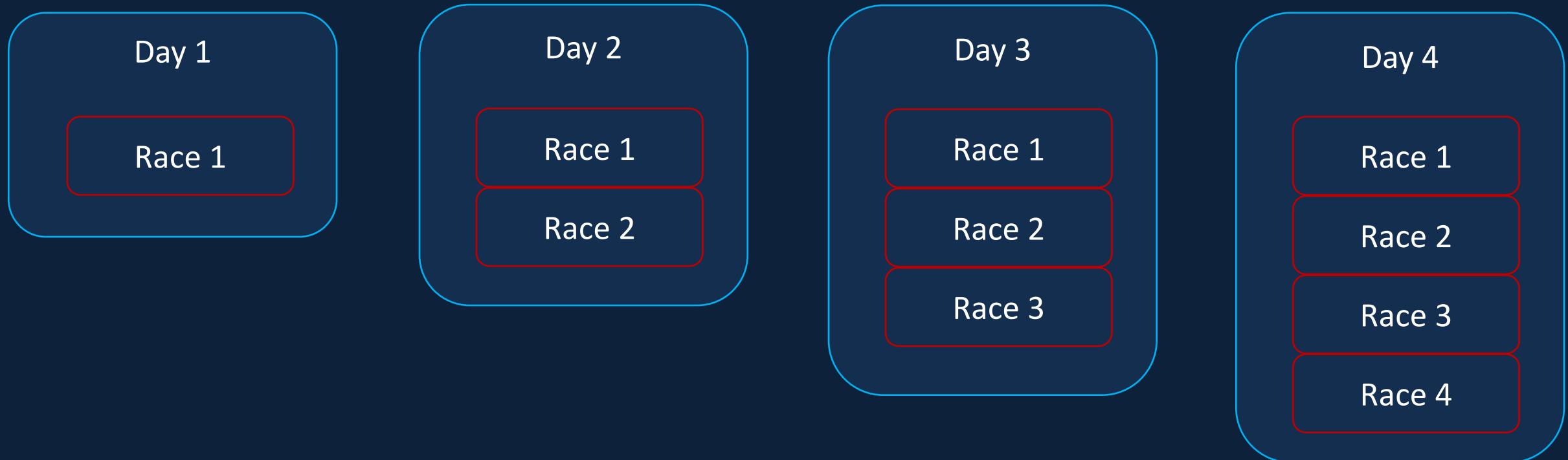
Implementation

Data Load Types

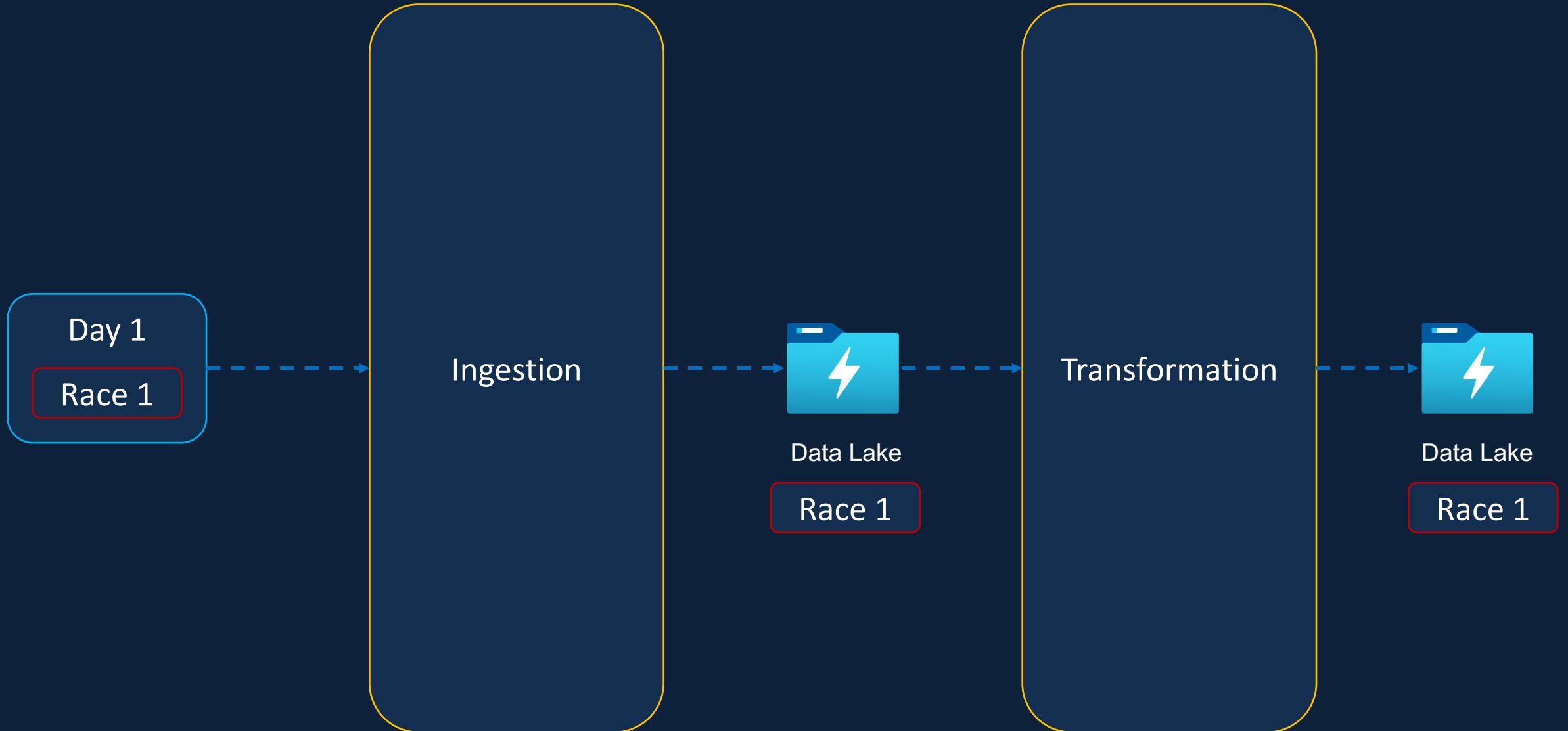
Full Load

Incremental Load

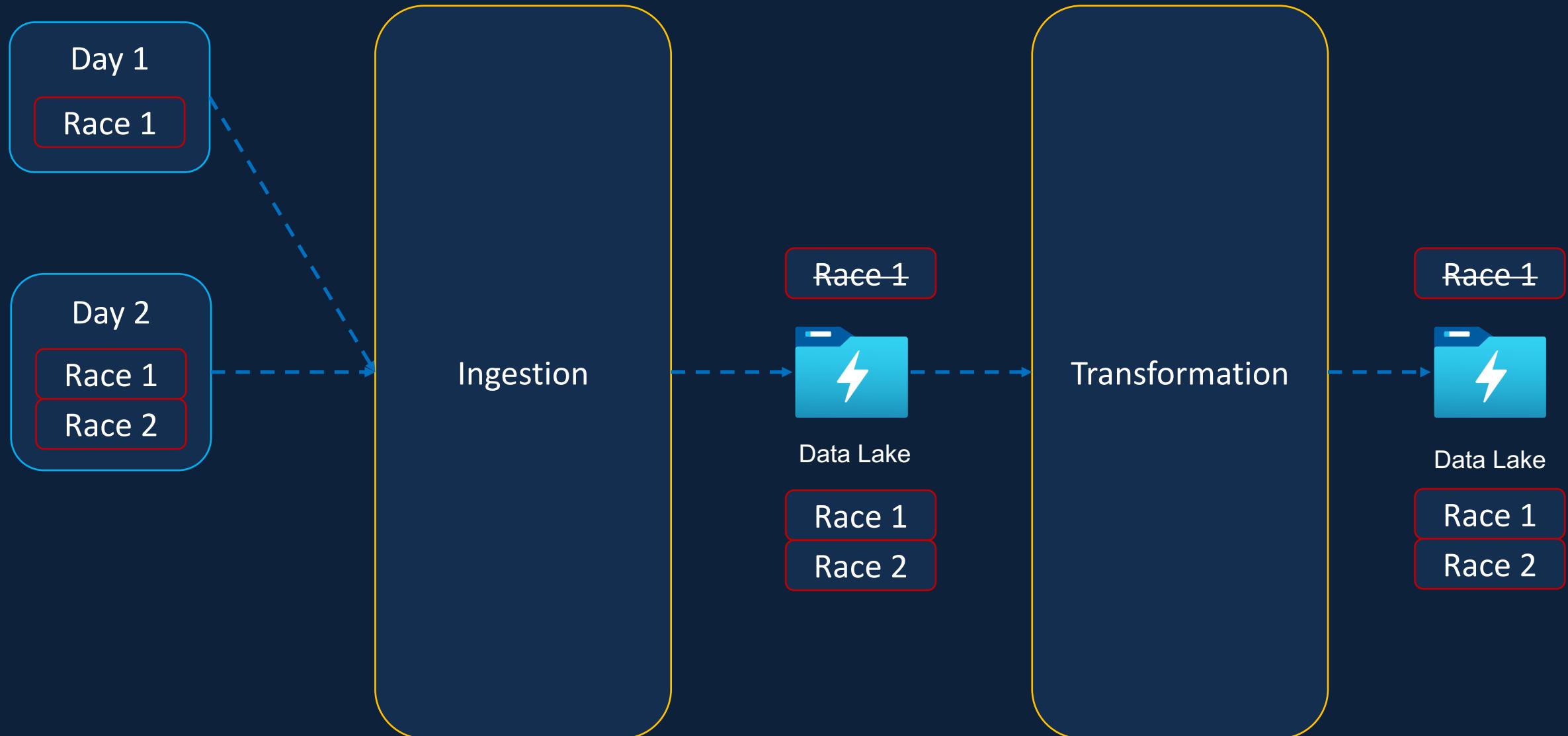
Full Dataset



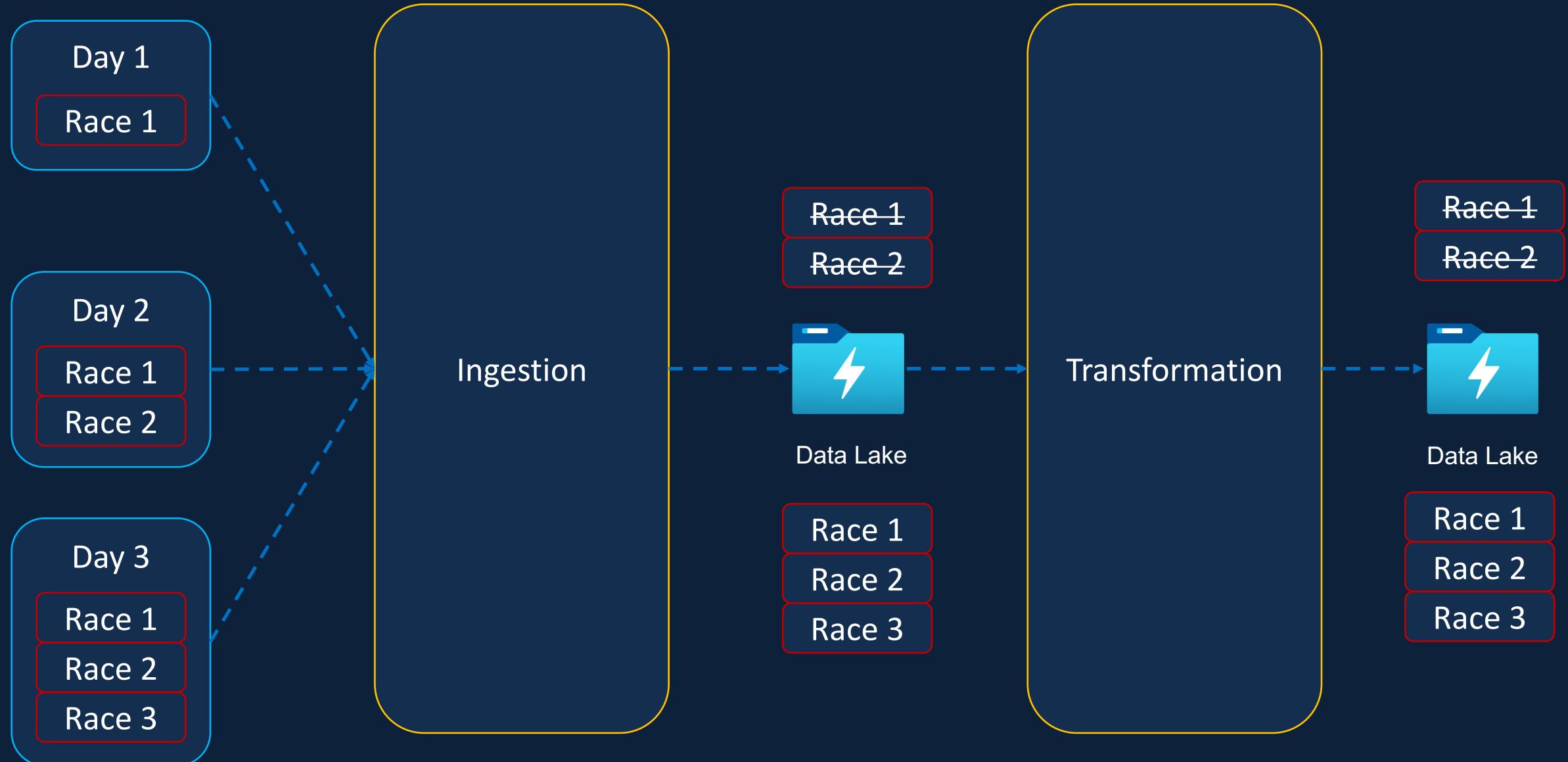
Full Refresh/ Load – Day 1



Full Refresh/ Load – Day 2



Full Refresh/ Load – Day 3



Incremental Dataset

Day 1

Race 1

Day 2

Race 2

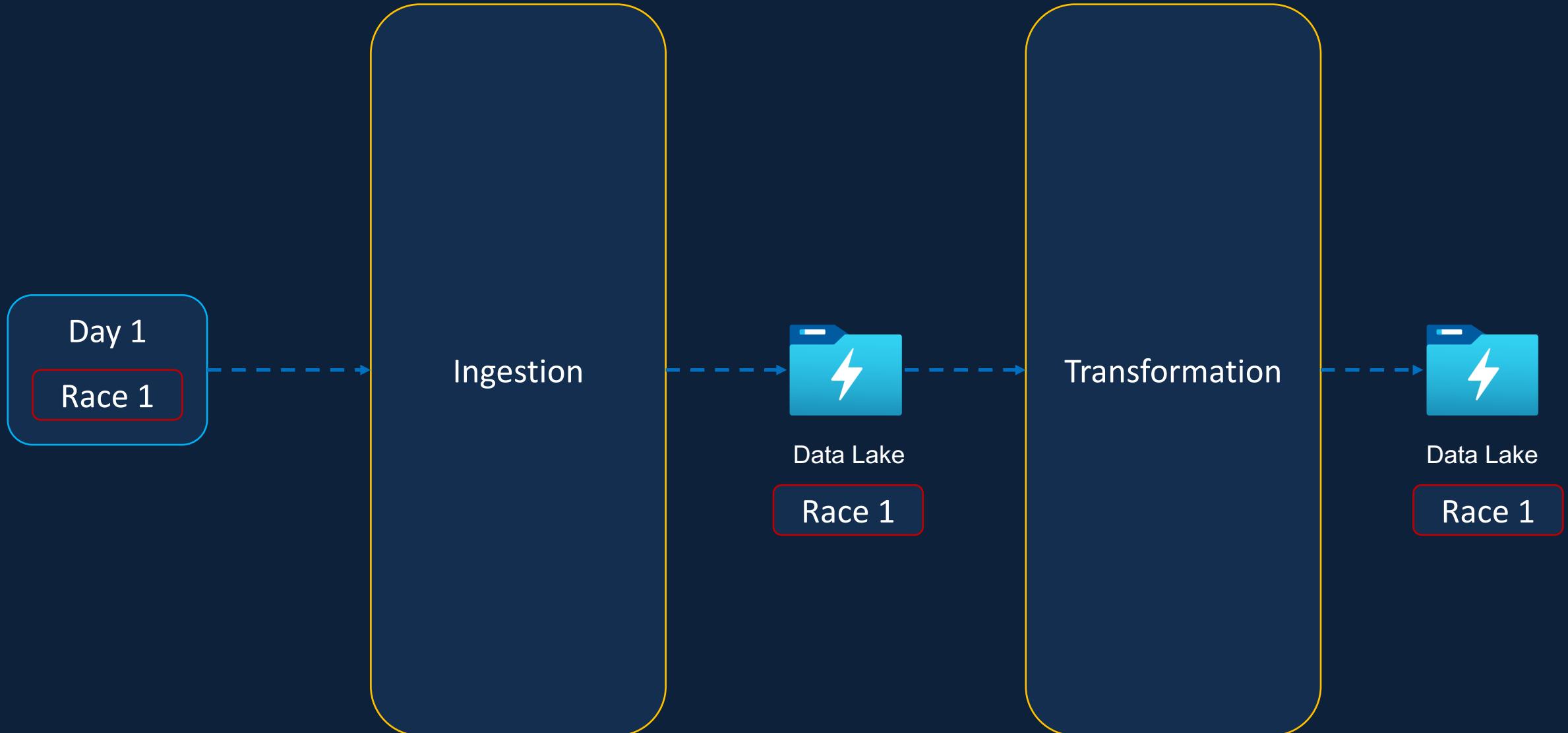
Day 3

Race 3

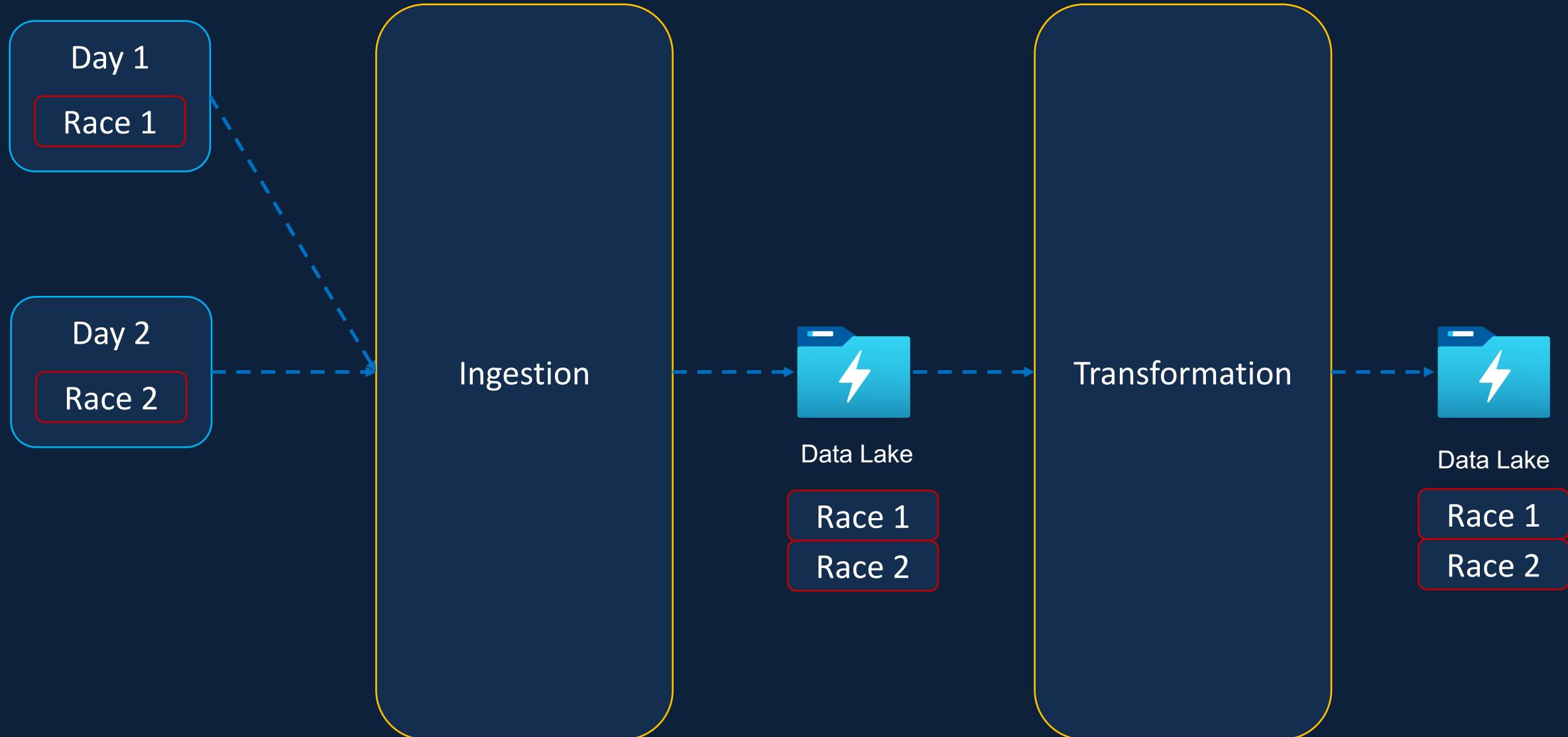
Day 4

Race 4

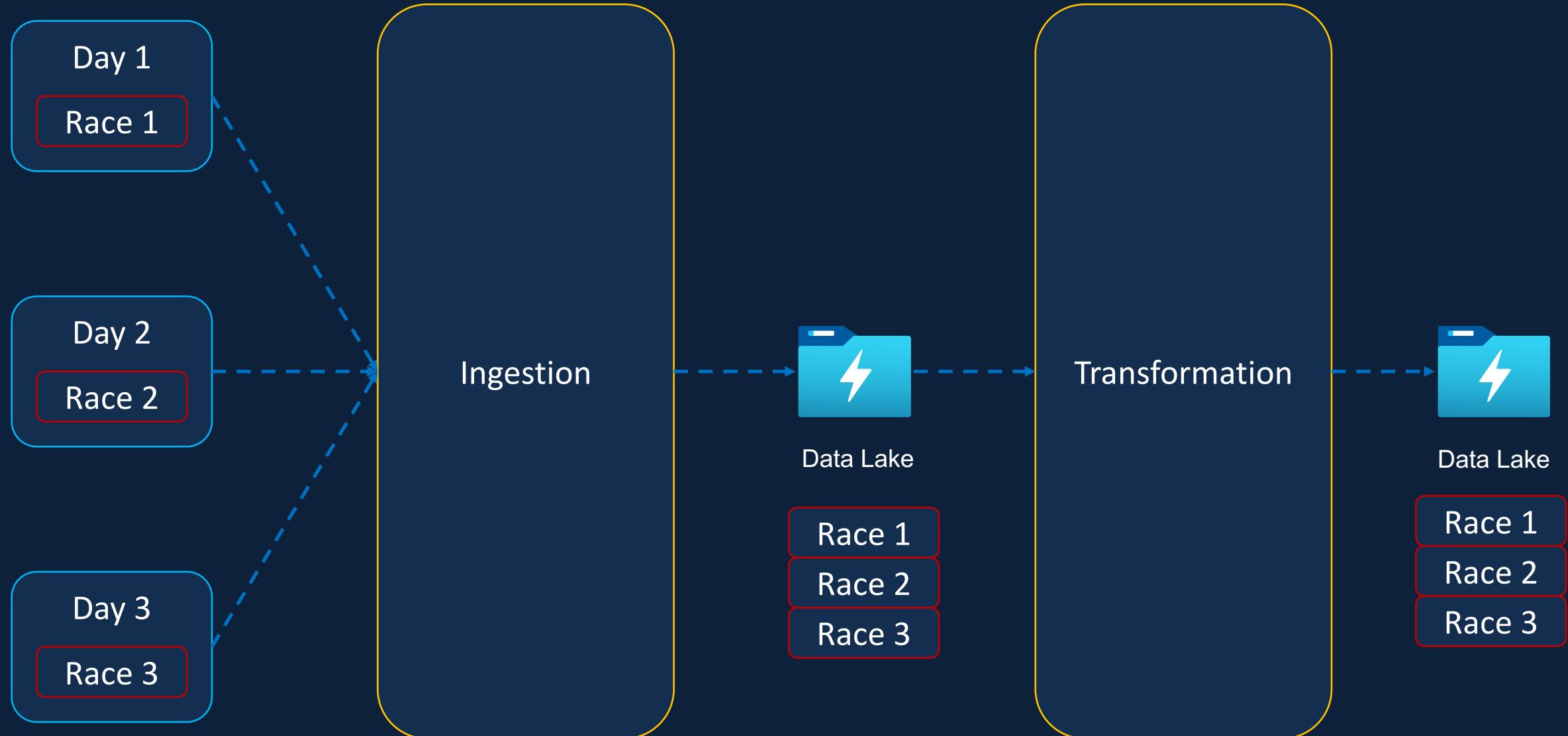
Incremental Load – Day 1



Incremental Load – Day 2



Incremental Load – Day 3



Hybrid Scenarios

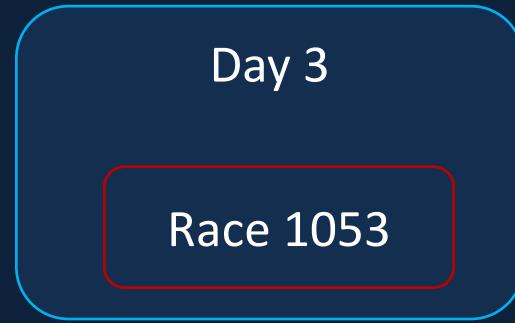
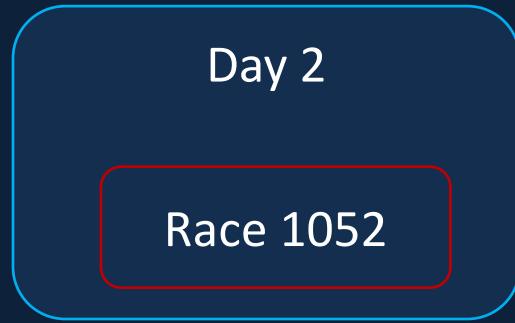
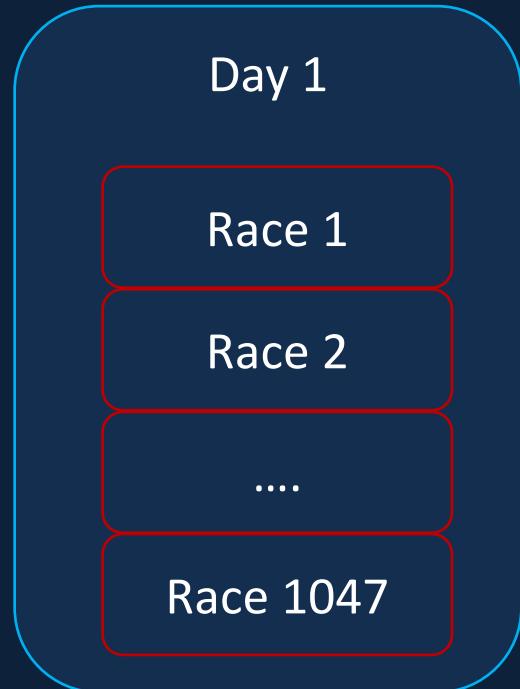
Full Dataset received, but data loaded & transformed incrementally

Incremental dataset received, but data loaded & transformed in full

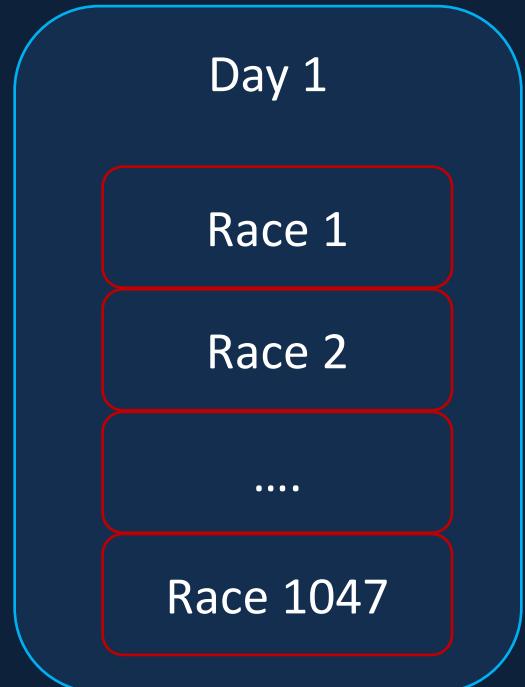
Data received contains both full and incremental files

Incremental data received. Ingested incrementally & transformed in full

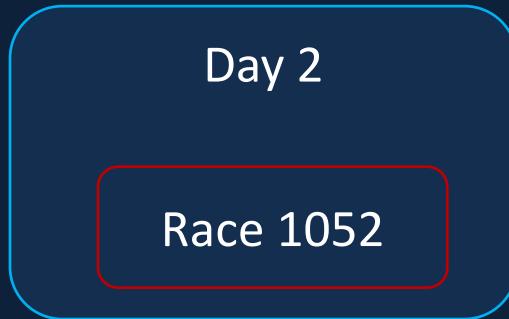
Formula1 Scenario



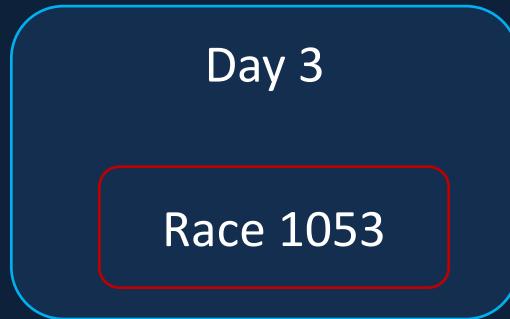
Formula1 Scenario / Solution 1



Full Refresh

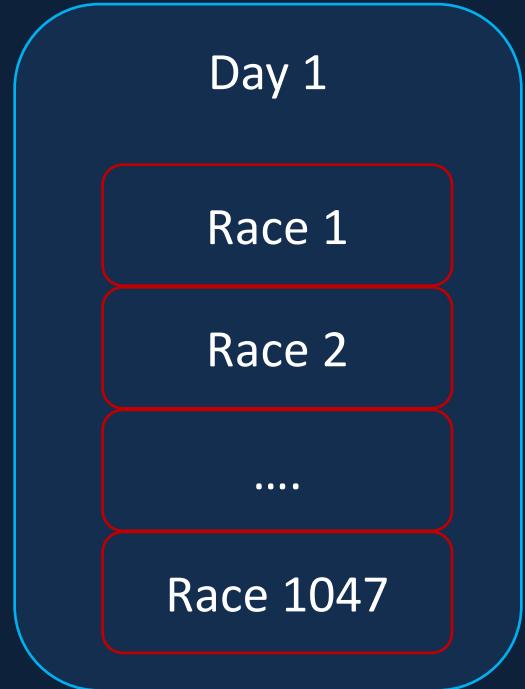


Incremental Load

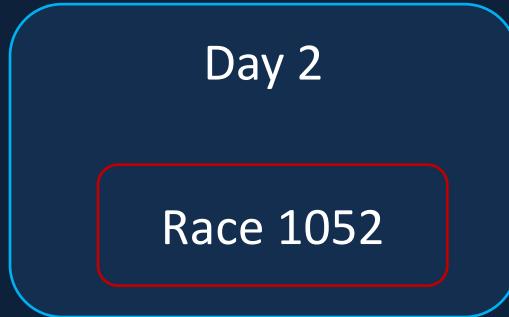


Incremental Load

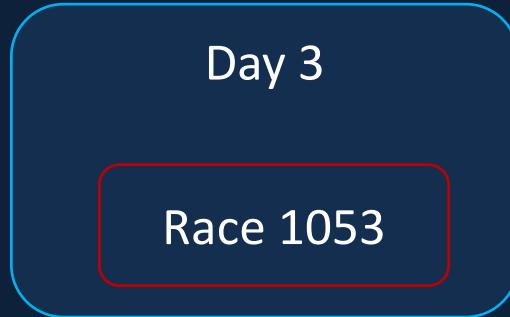
Formula1 Scenario / Solution 2



Incremental Load



Incremental Load



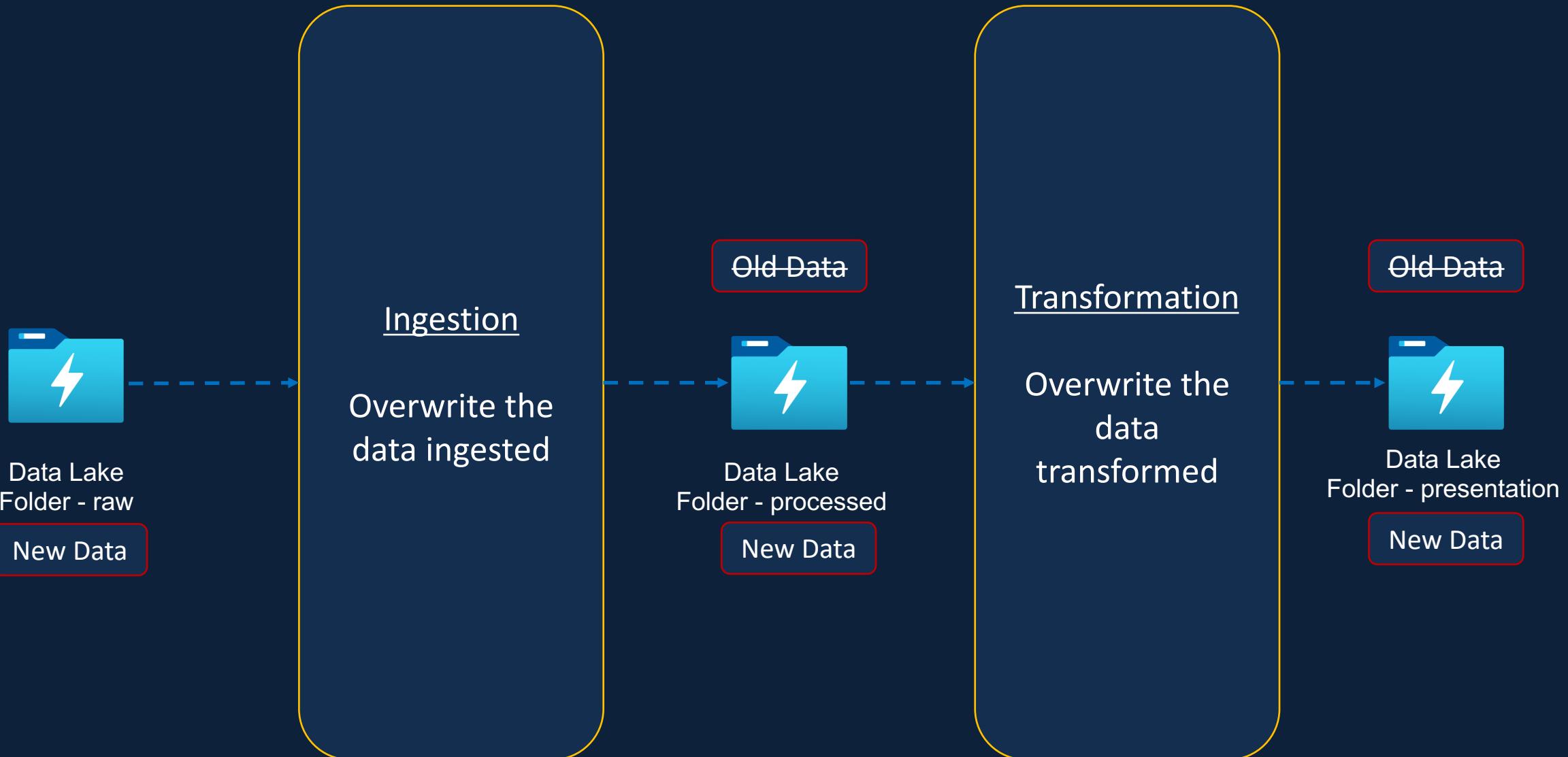
Incremental Load

Formula1 Data Files

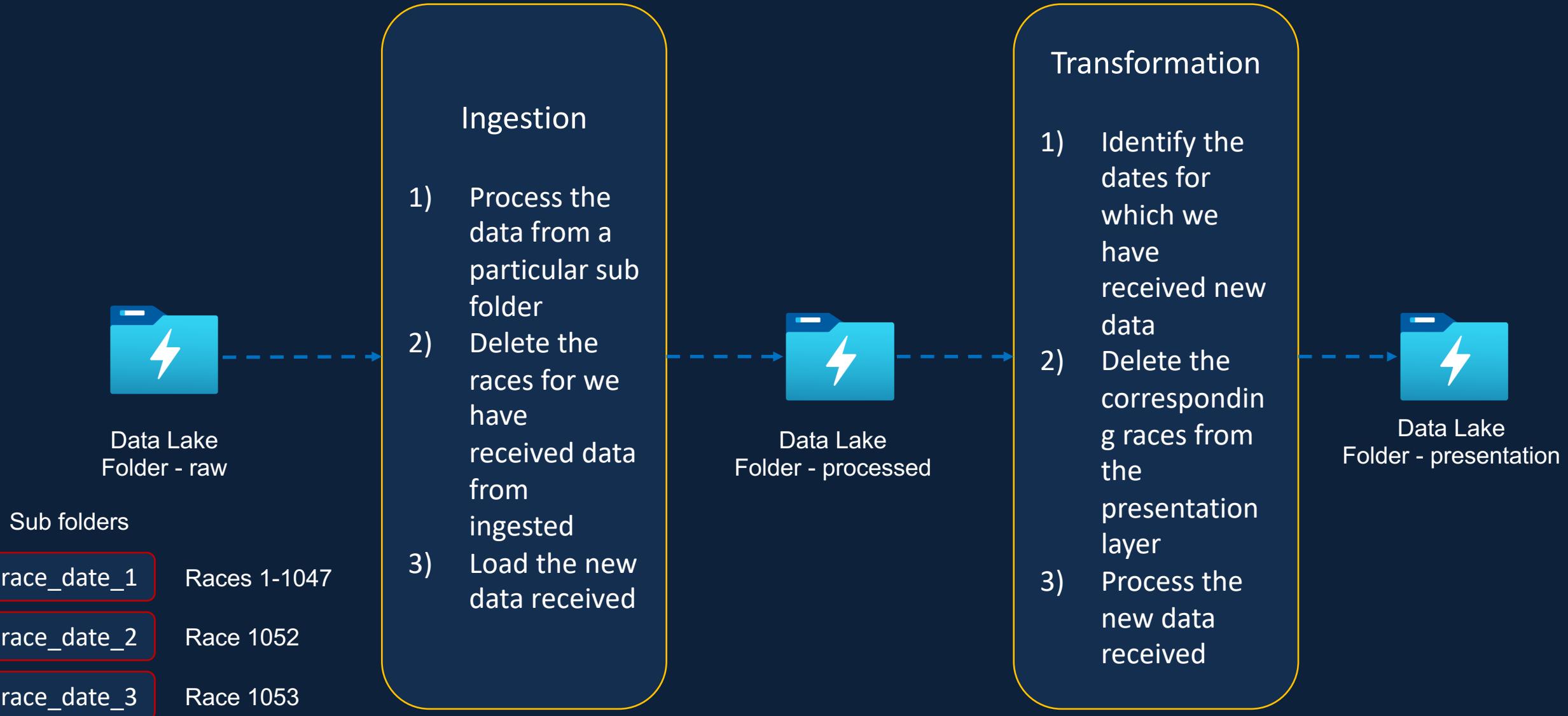


Circuits	Data from all races
Races	Data from all races
Constructors	Data from all races
Drivers	Data from all races
Results	Data only from that race
PitStops	Data only from that race
LapTimes	Data only from that race
Qualifying	Data only from that race

Current Solution



New Solution



Delta Lake



Pitfalls of Data Lakes

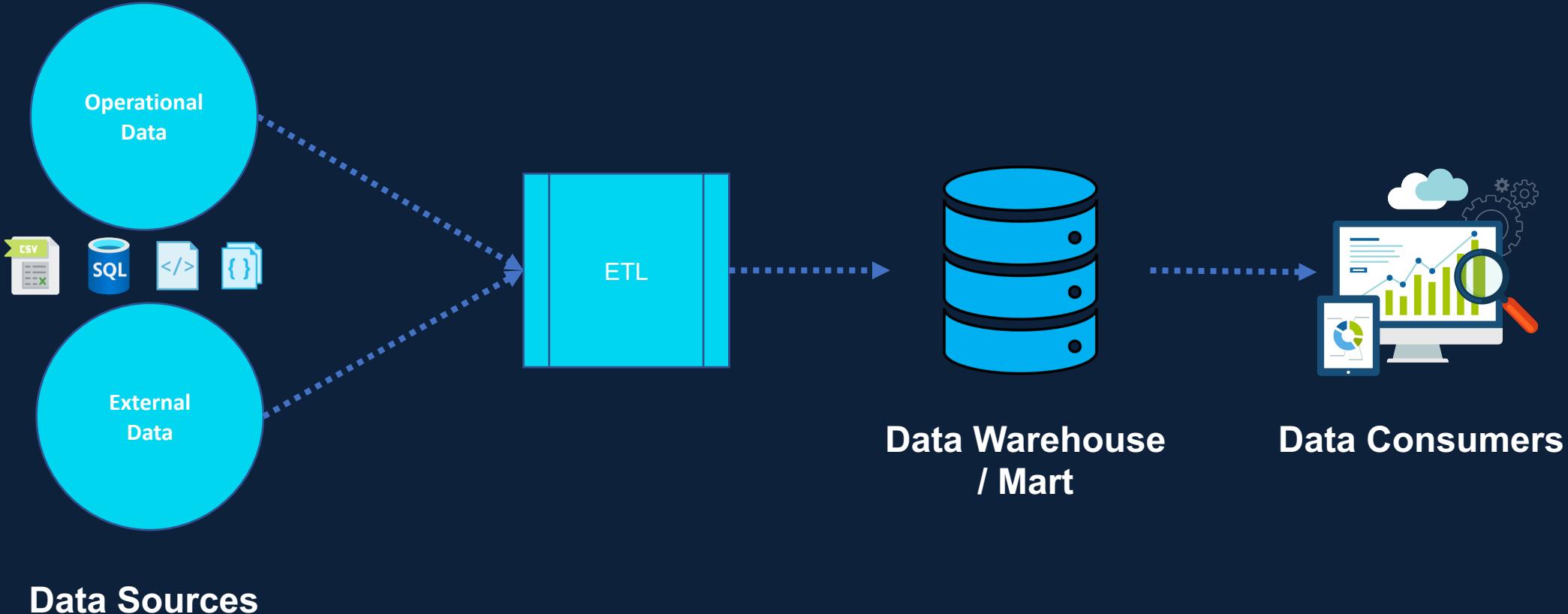
Lakehouse Architecture

Delta Lake Capabilities

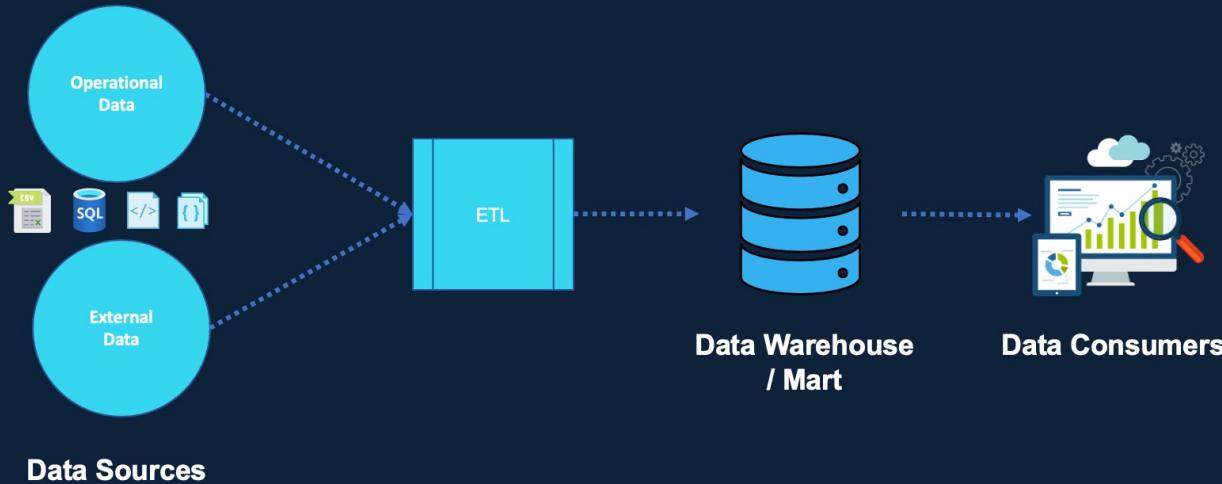
Convert F1 project to Delta Lake

Delta Lake

Data Warehouse



Data Warehouse



Lack of support for unstructured data

Longer to ingest new data

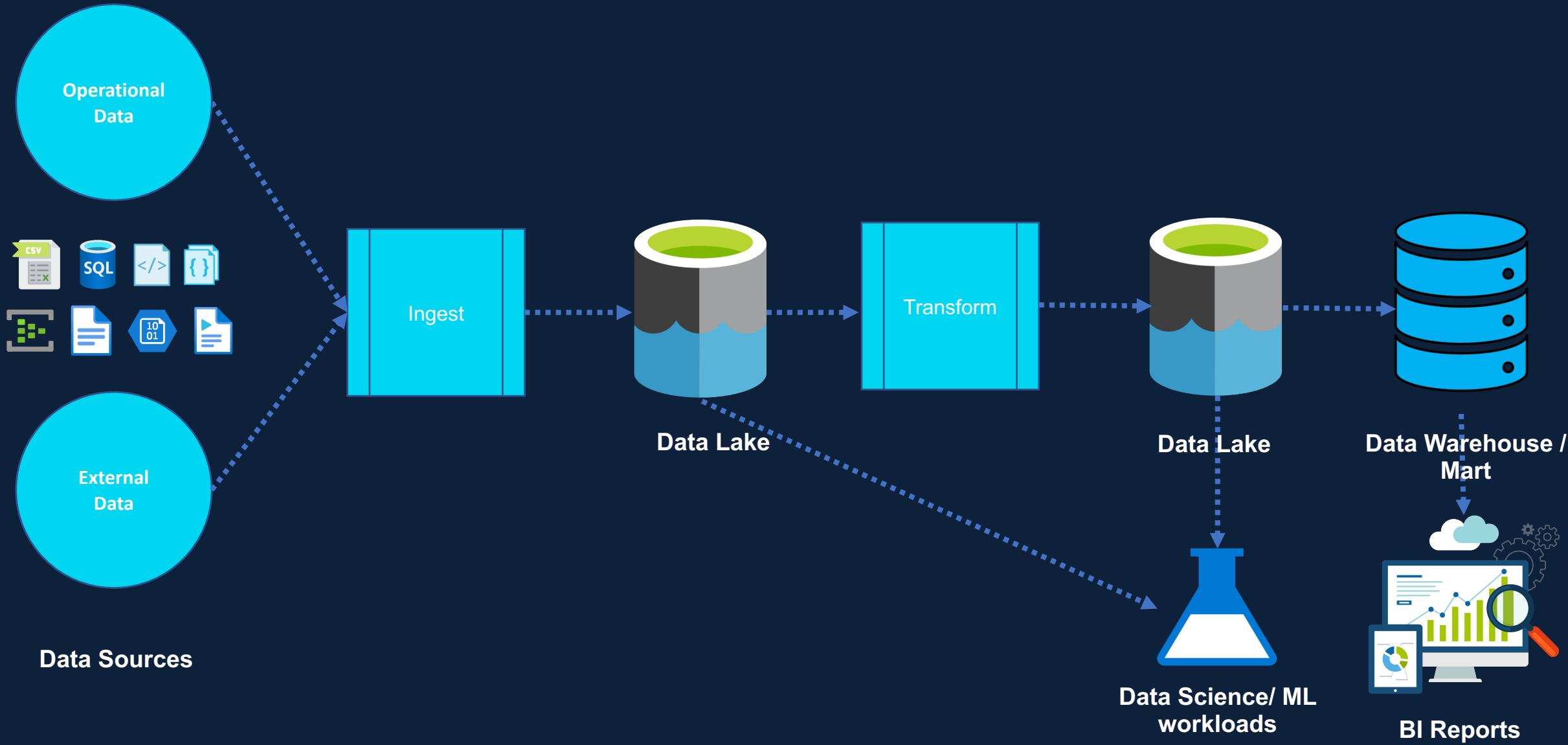
Proprietary data formats

Scalability

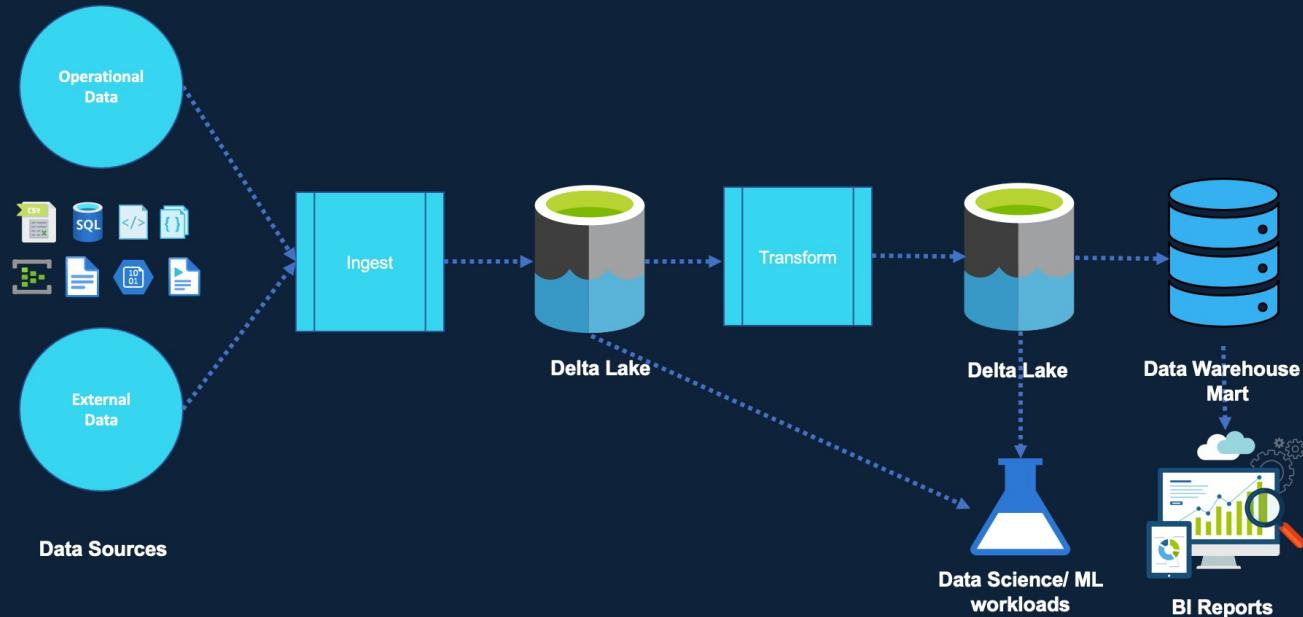
Expensive to store data

Lack of support for ML/ AI workloads

Data Lake

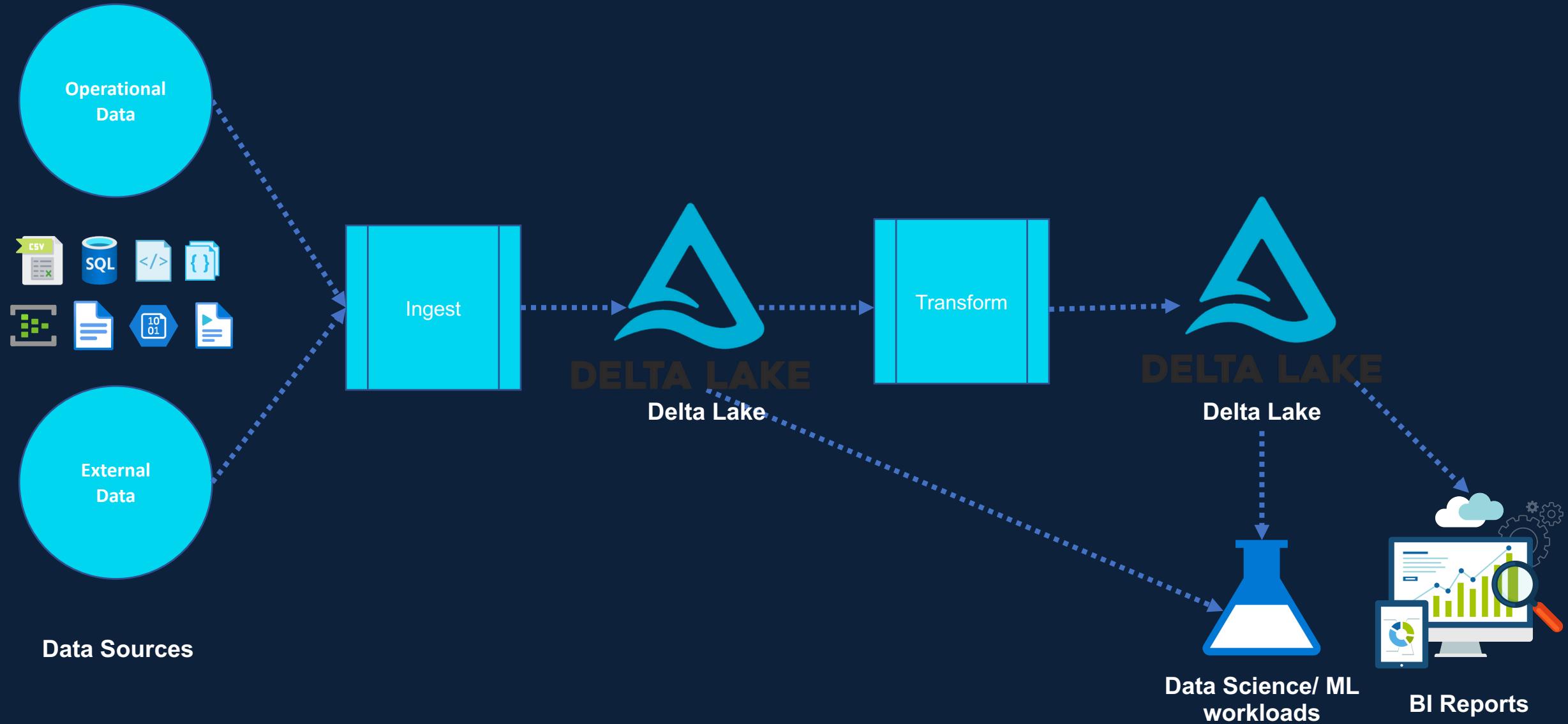


Data Lake

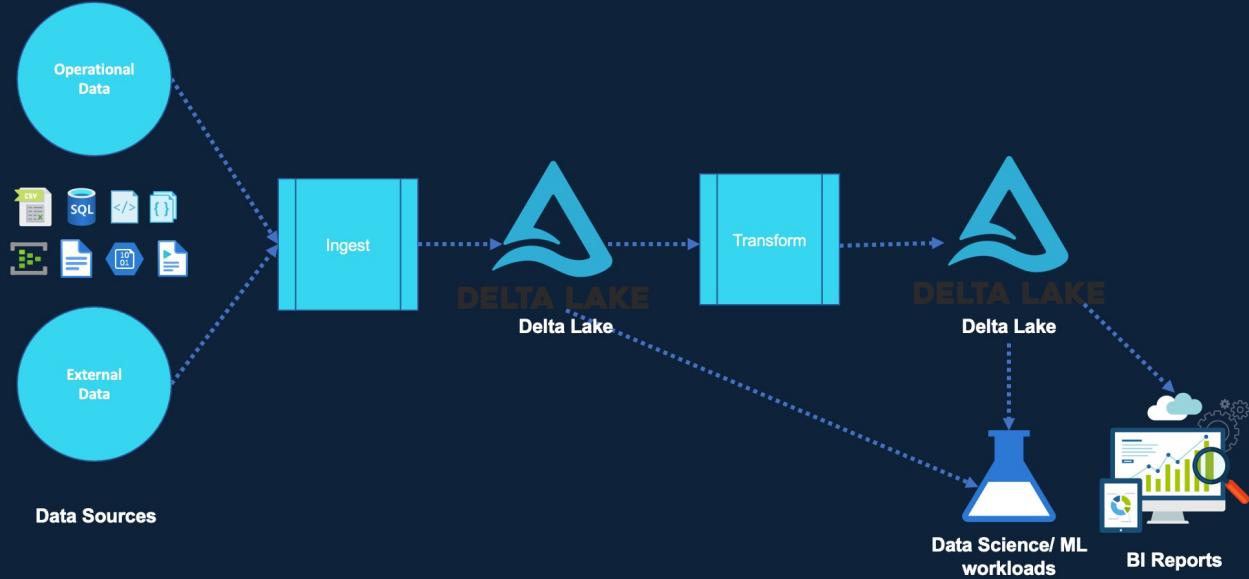


- No support for ACID transactions
- Failed jobs leave partial files
- Inconsistent reads
- Unable to handle corrections to data
- Unable to roll back any data.
- Lack of ability remove data for GDPR etc
- No history or versioning
- Poor performance
- Poor BI support
- Complex to set-up
- Lambda architecture for streaming workloads

Data Lakehouse

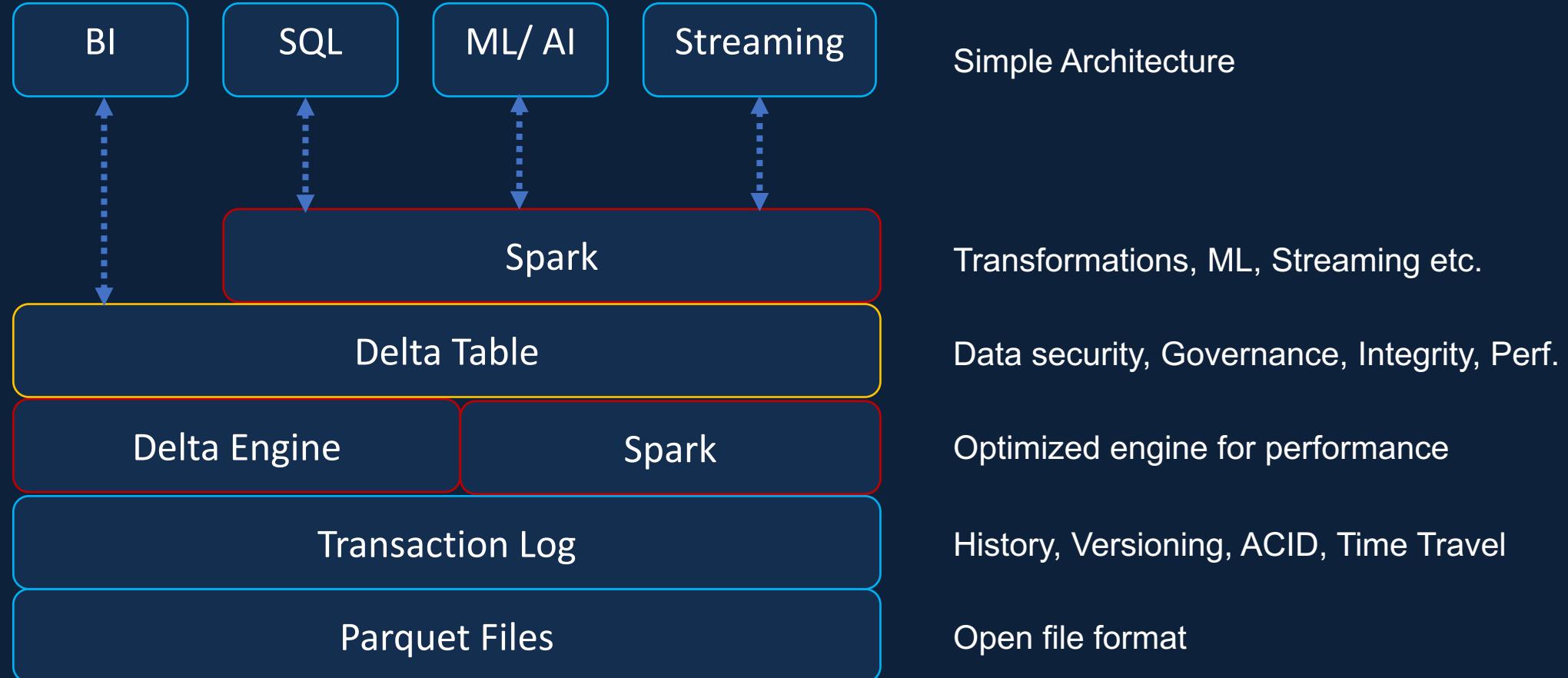


Data Lakehouse



- Handles all types of data
- Cheap cloud object storage
- Uses open source format
- Support for all types of workloads
- Ability to use BI tools directly
- ACID support
- History & Versioning
- Better performance
- Simple architecture

Delta Lake



Delta Lake Demo

Azure Data Factory



Overview

Creating Data Factory Service

Data Factory Components

Creating Pipelines

Creating Triggers

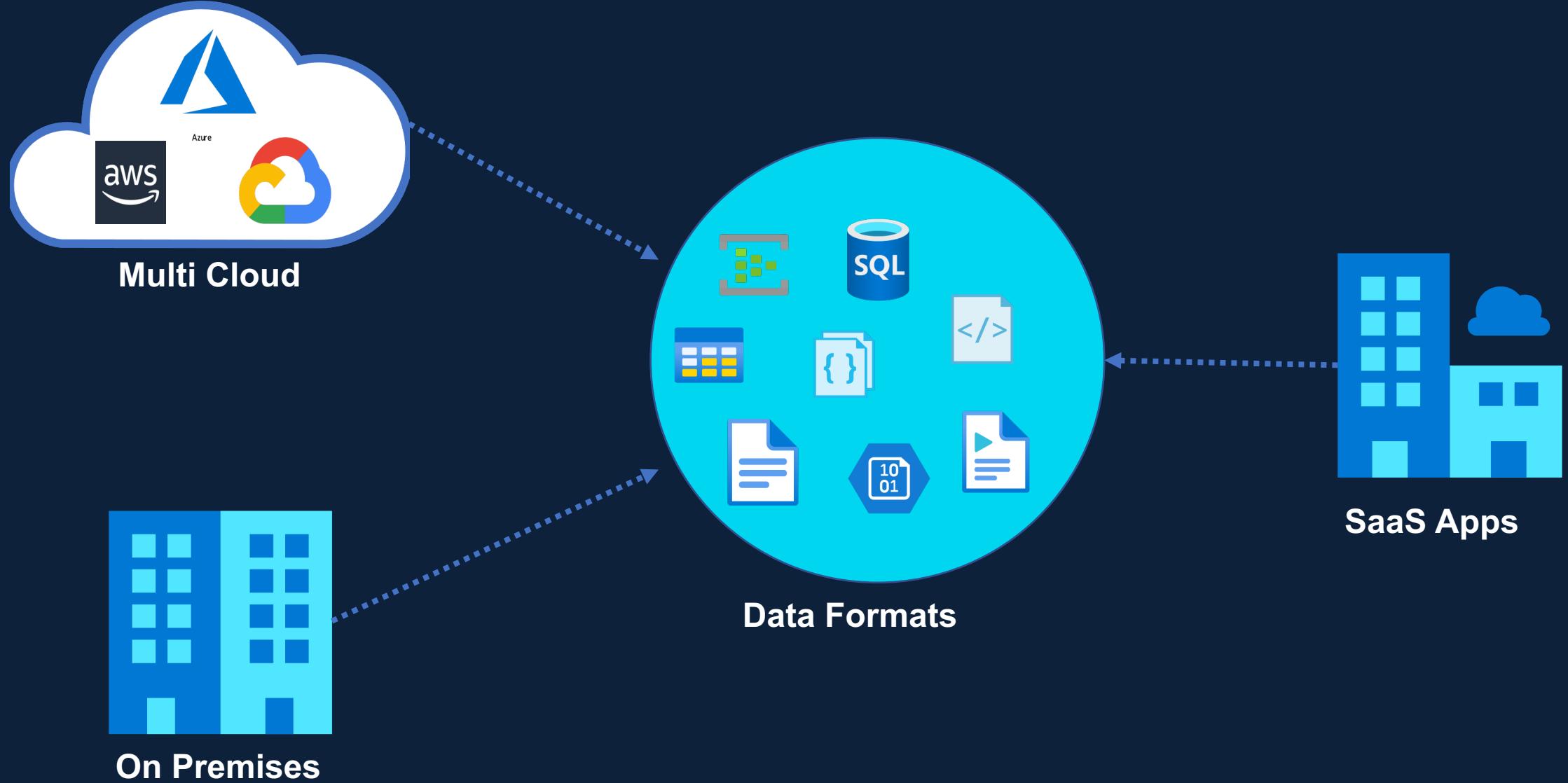
Azure Data Factory Overview

What is Azure Data Factory

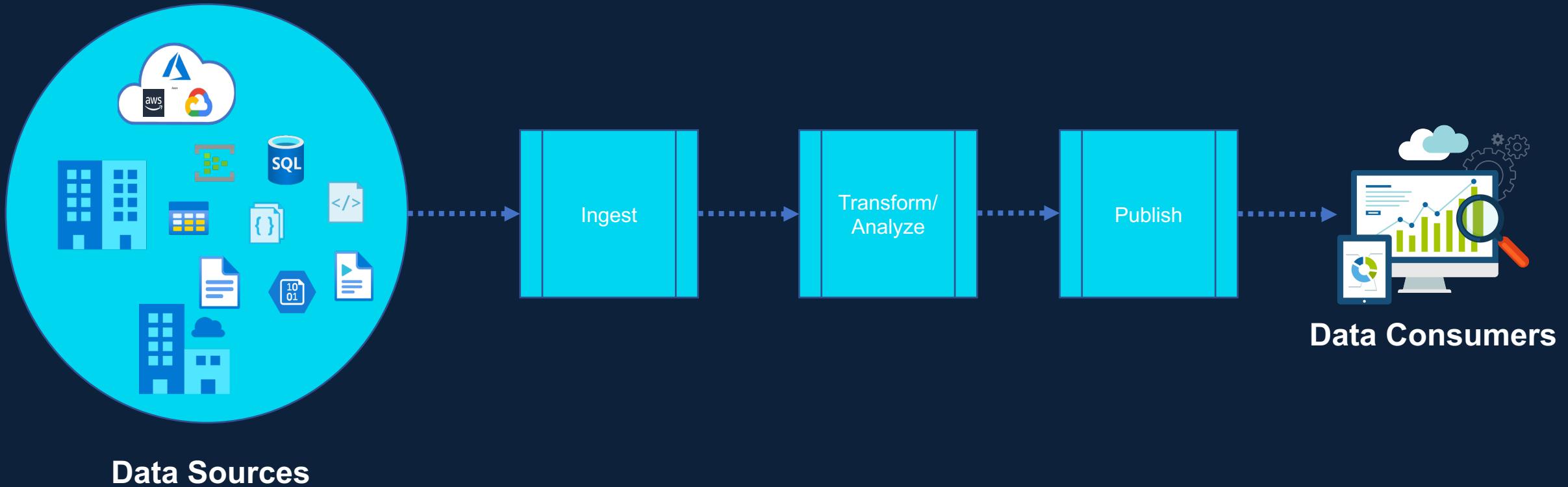


A fully managed, serverless data integration solution for ingesting, preparing and transforming all of your data at scale.

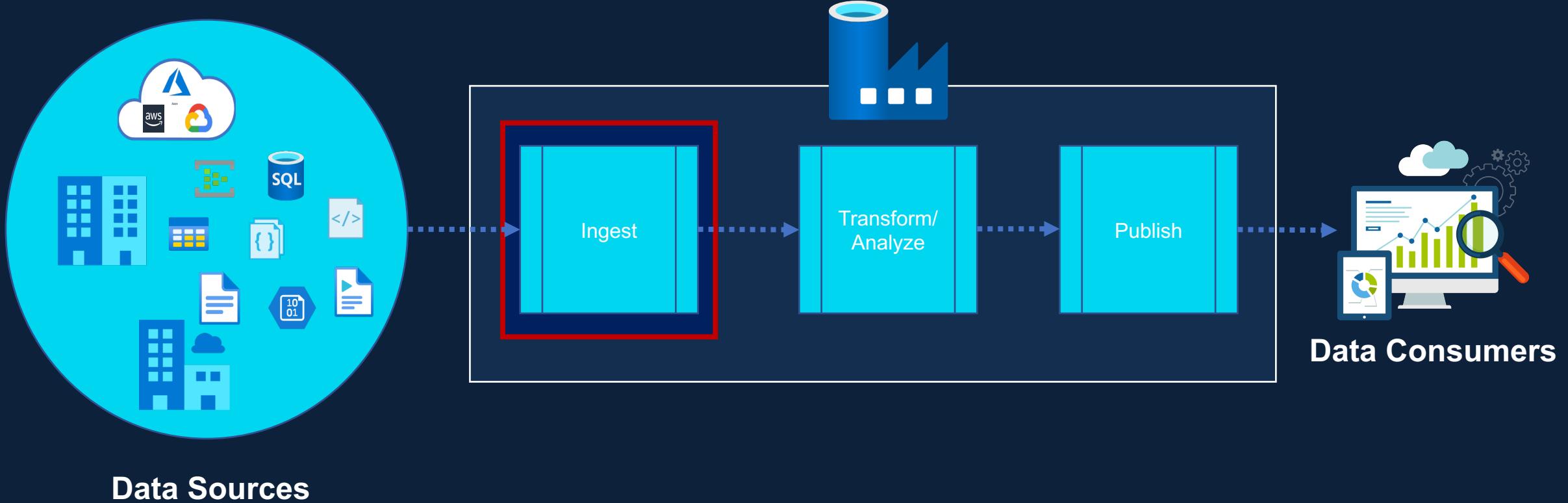
The Data Problem



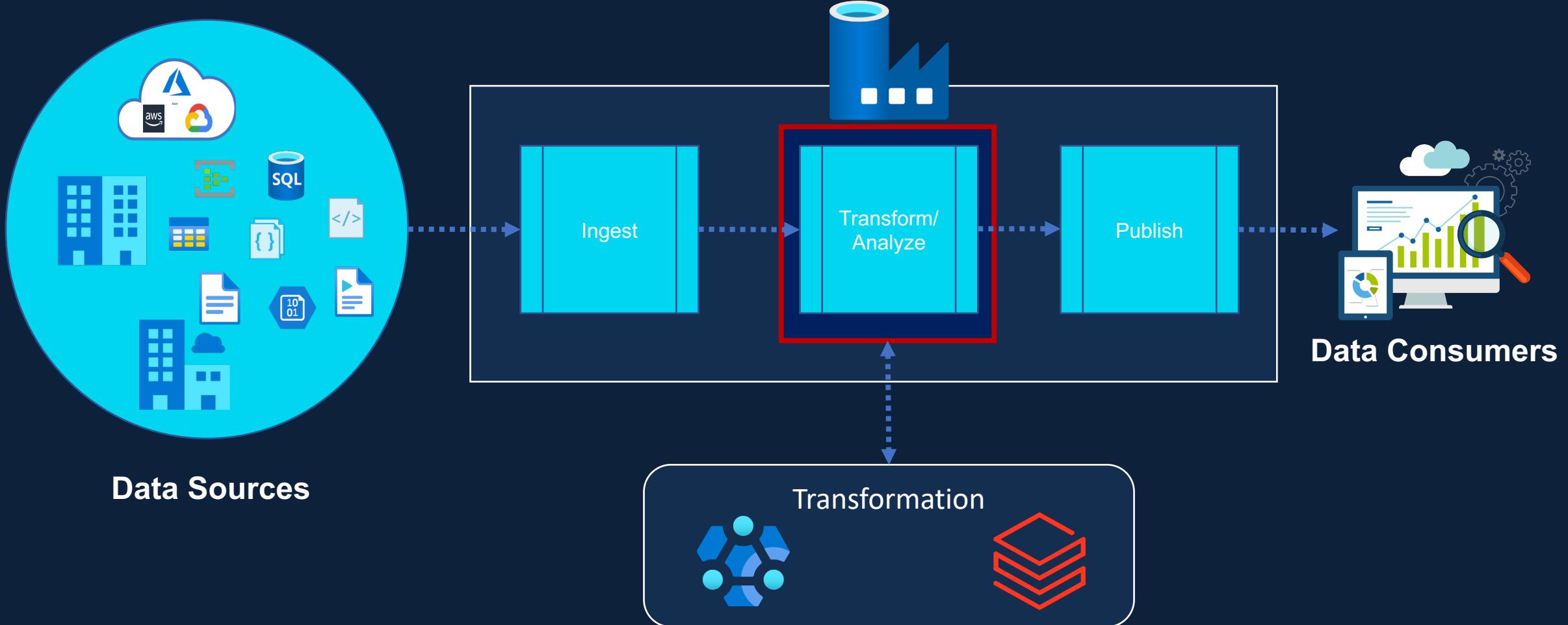
The Data Problem



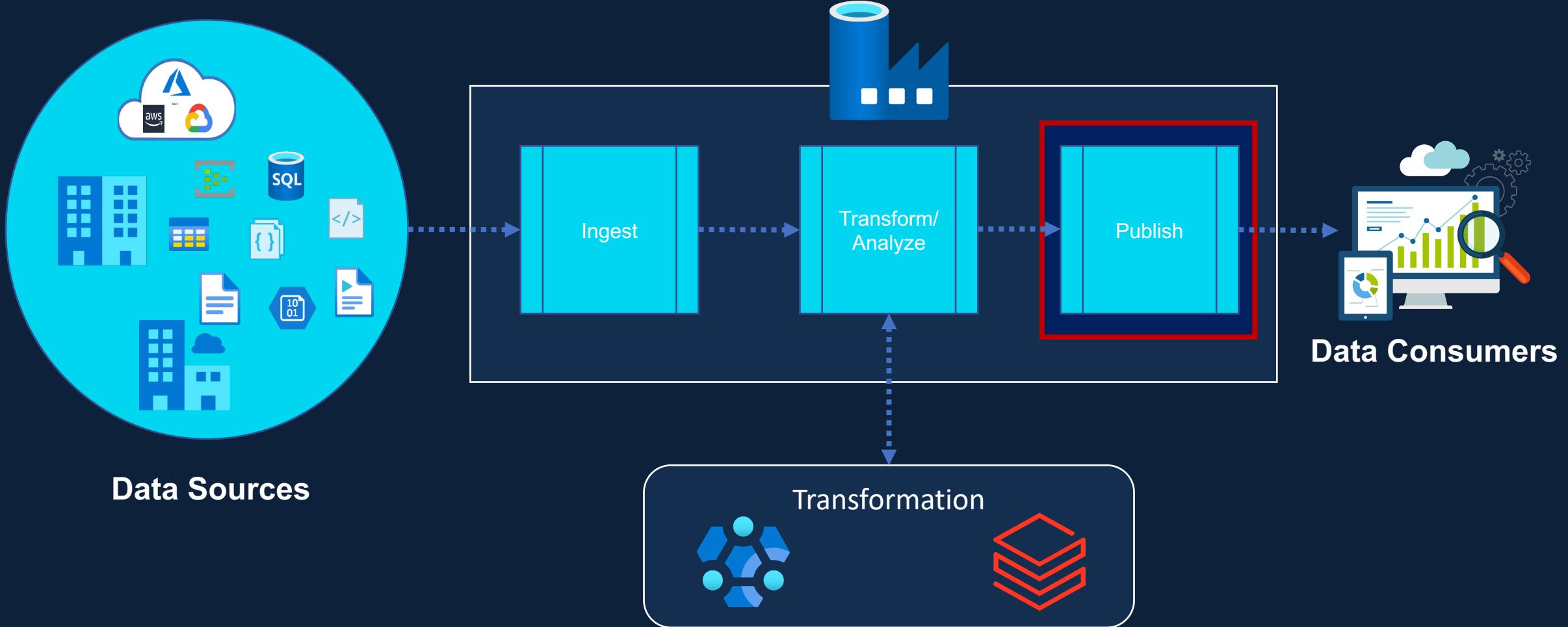
The Data Problem



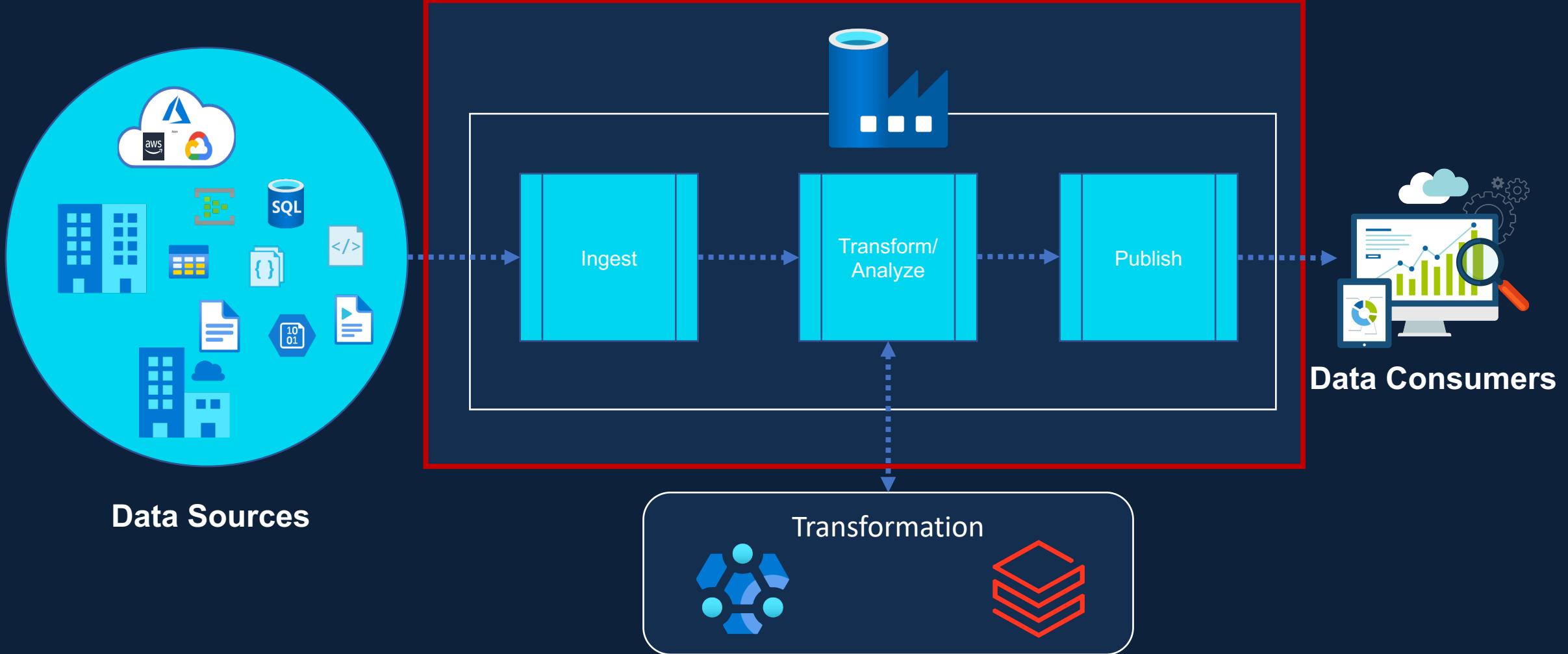
The Data Problem



The Data Problem



The Data Problem



What is Azure Data Factory



Fully Managed Service

Serverless

Data Integration Service

Data Transformation Service

Data Orchestration Service

A fully managed, serverless data integration solution for ingesting, preparing and transforming all of your data at scale.

What Azure Data Factory Is Not



Data Migration Tool

Data Streaming Service

Suitable for Complex Data Transformations

Data Storage Service

Create Data Factory Service

Data Factory Components

Trigger

Pipeline

Activity

Activity

Dataset

Linked Service

Linked Service

Storage

ADLS

SQL
Database

Compute

Azure
Databricks

Azure
HDInsight

Connecting from Power BI

Congratulations!

&

Thank you

Feedback

Ratings & Review

Thank you
&
Good Luck!