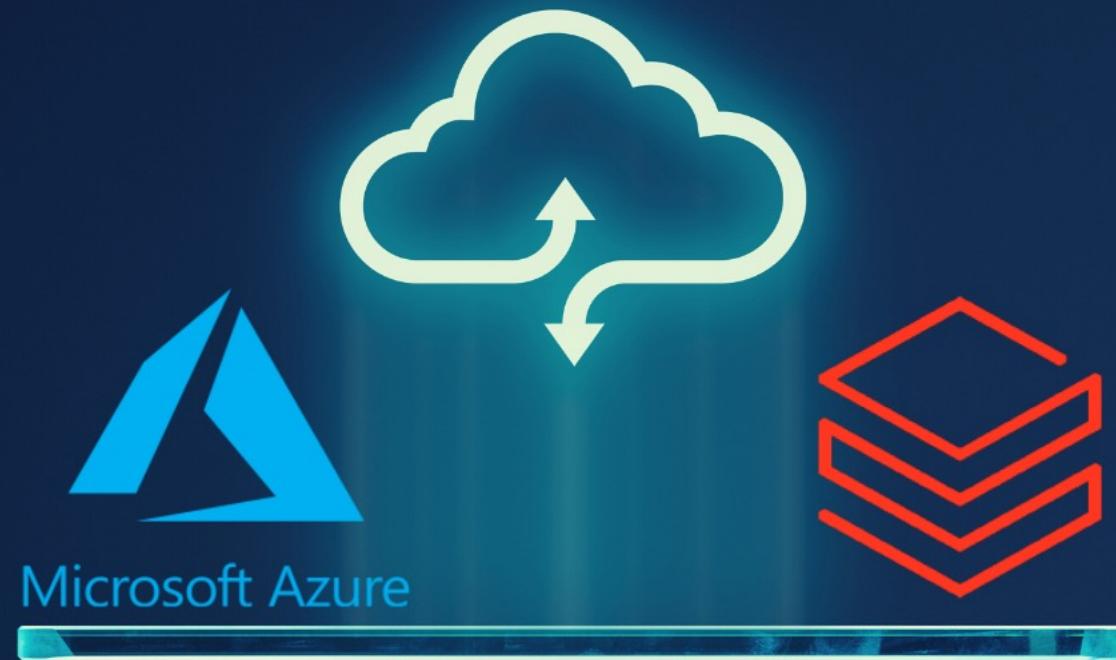


# Azure Databricks For Data Engineers

## Project on Formula1 Racing



# About Me



Ramesh Retnasamy  
Data Engineer/ Machine Learning Engineer



**LinkedIn**

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

# About this course

The primary focus of this course is Azure Databricks, which is a widely adopted SPARK based unified Data Analytics platform optimized for Microsoft Cloud.



## Azure Databricks



Python

## PySpark

Apart from learning the standard Databricks capabilities and tools, we'll also learn Spark using Python as well as SQL.



## Spark SQL



## Delta Lake

I've also created a dedicated section on Delta Lake and how we can use Delta Lake to implement the emerging Lakehouse architecture.

# About this course



## Azure Databricks



I'll take you through all the way from creating a Data Lake service for reading and writing data to the Data Lake.

### Azure Data Lake Storage Gen2

### Azure Data Factory

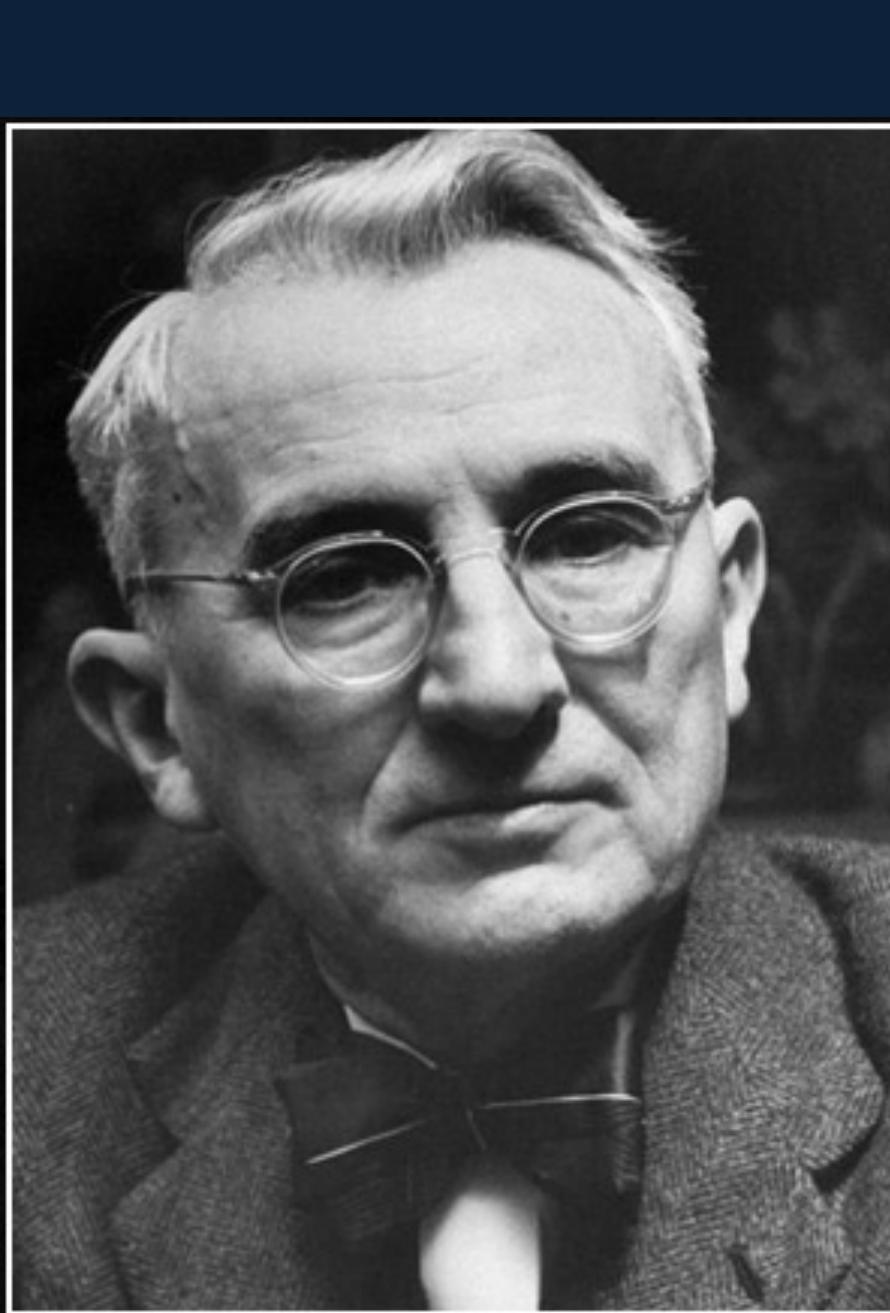
We use Azure Data Factory to create schedule and monitor data pipelines.

### Azure Key Vault

The course focuses on building a production quality data engineering solution, so secrets are managed by Azure Key Vault, and the course provides the necessary knowledge required to achieve this.

### PowerBI

I'll also take you through how we can connect to the data in Databricks from Power BI to create BI reports.



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

— *Dale Carnegie* —

I strongly believe in learning by doing.

Whenever I learn something new, I generally apply that to a project which is interesting and engaging.

This not only allows for the learning to be fun, but also reinforces the learning and make it stick.

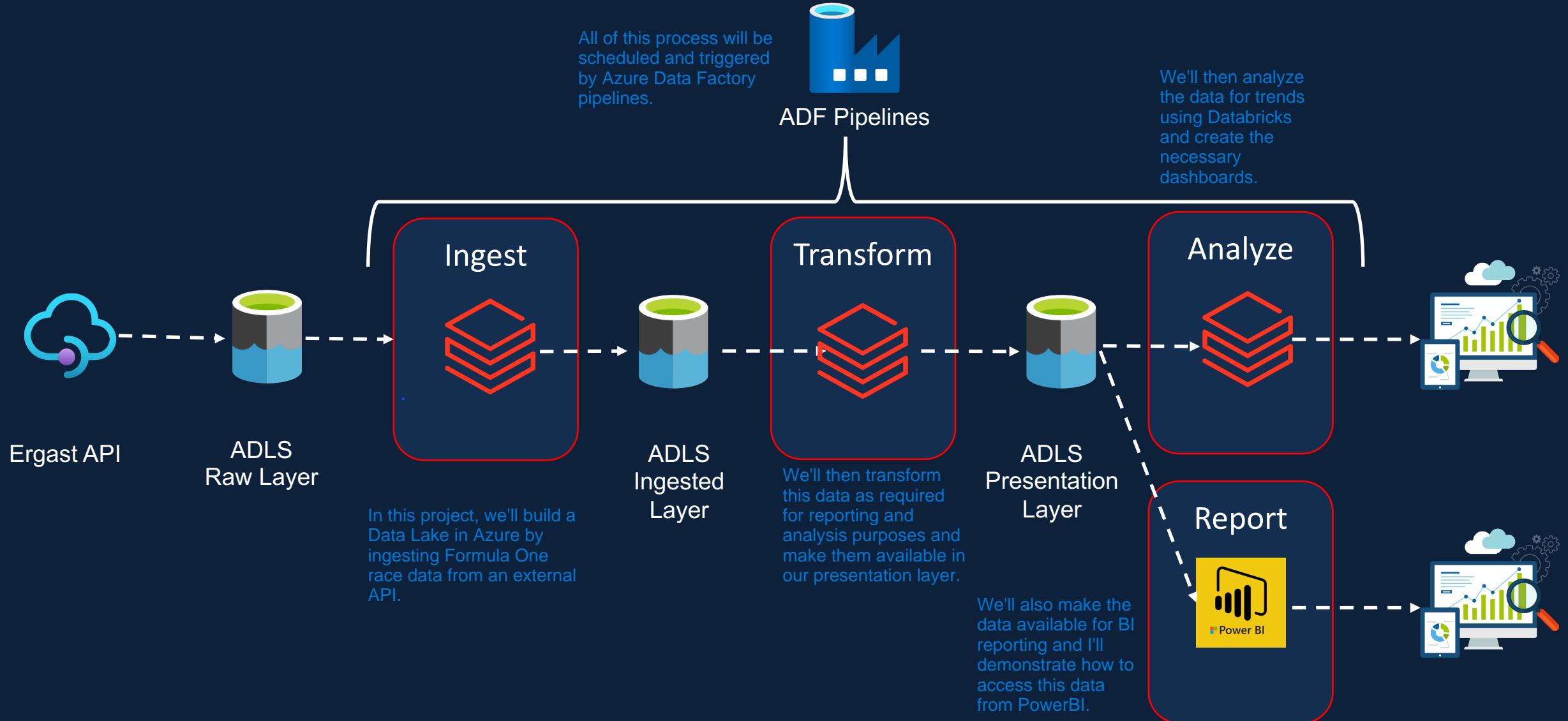
AZ QUOTES



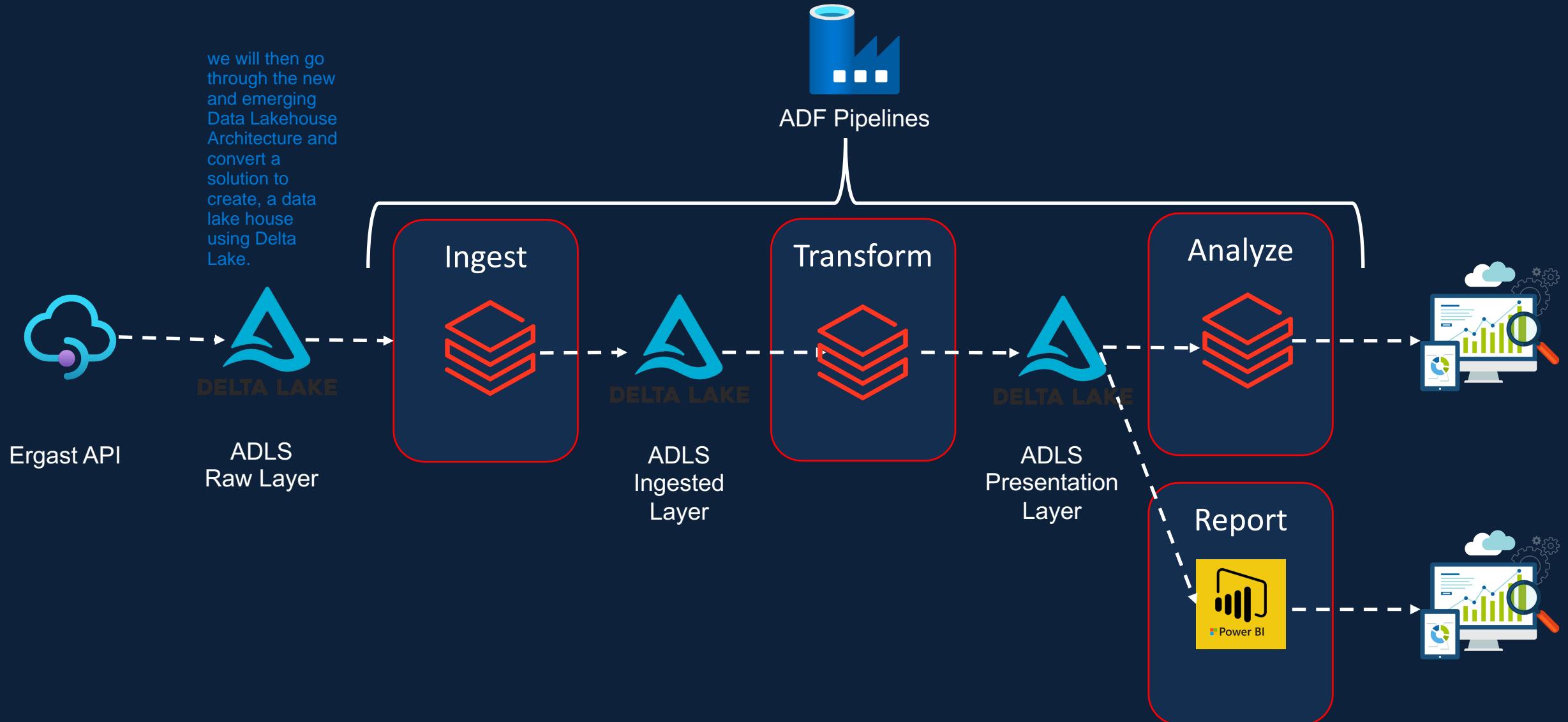
# Formula1 Cloud Data Platform

We'll be doing a project on building a Cloud Data platform for reporting and doing analysis of the data from Formula One Motorsport.

# Formula1 Cloud Data Platform



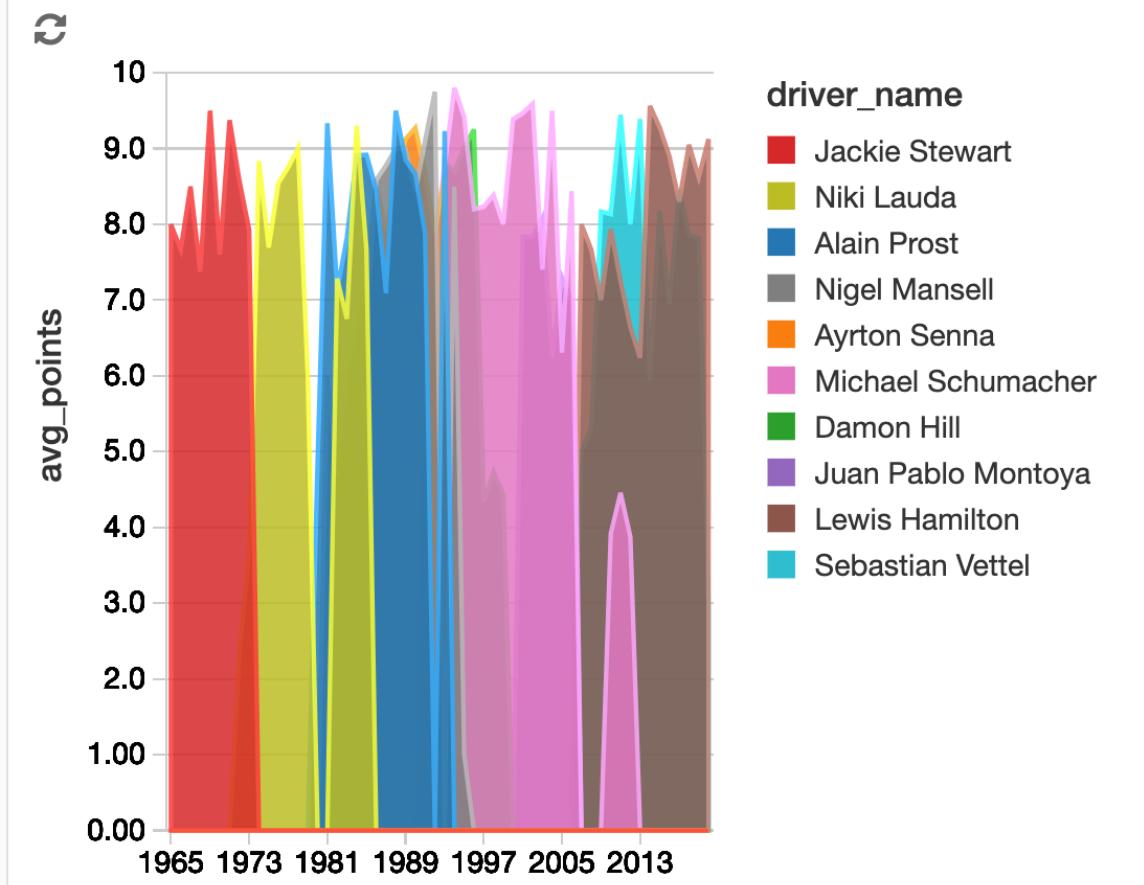
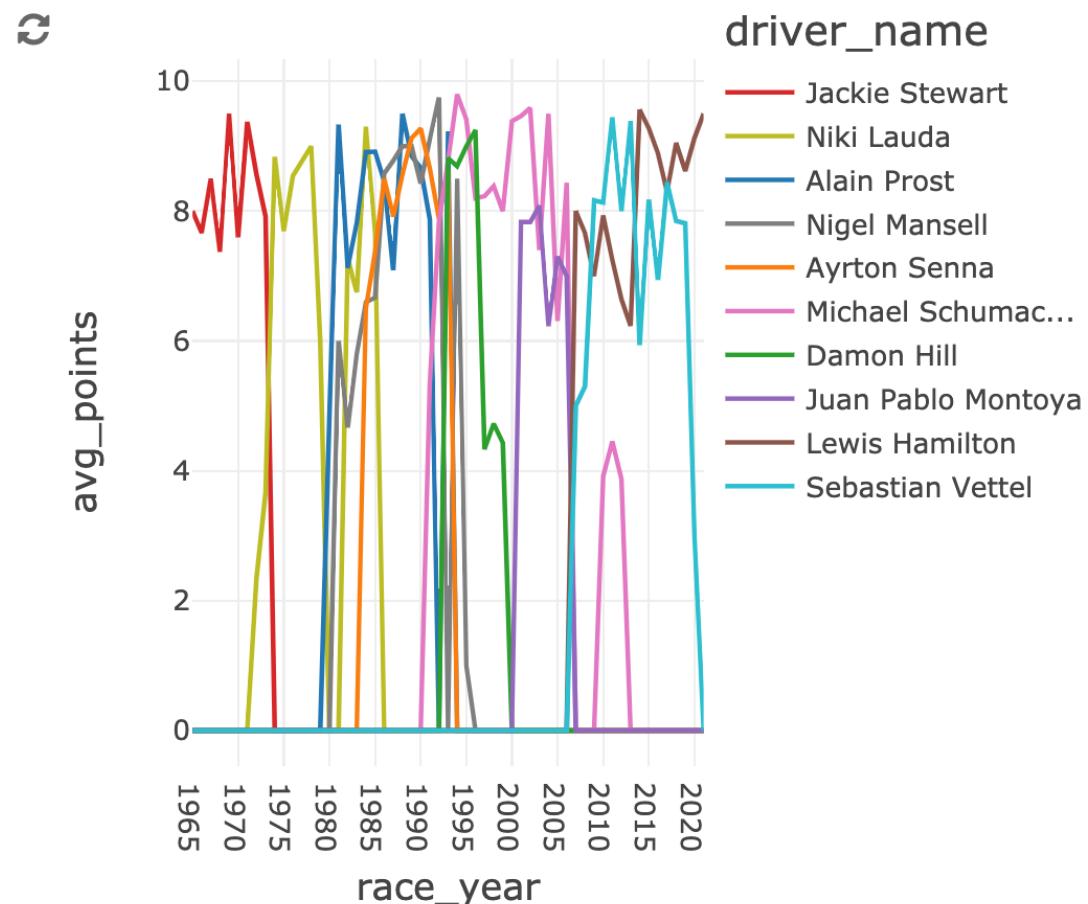
# Formula1 Cloud Data Platform



# Formula1 Cloud Data Platform

## Dominant Formula 1 Drivers of All Time

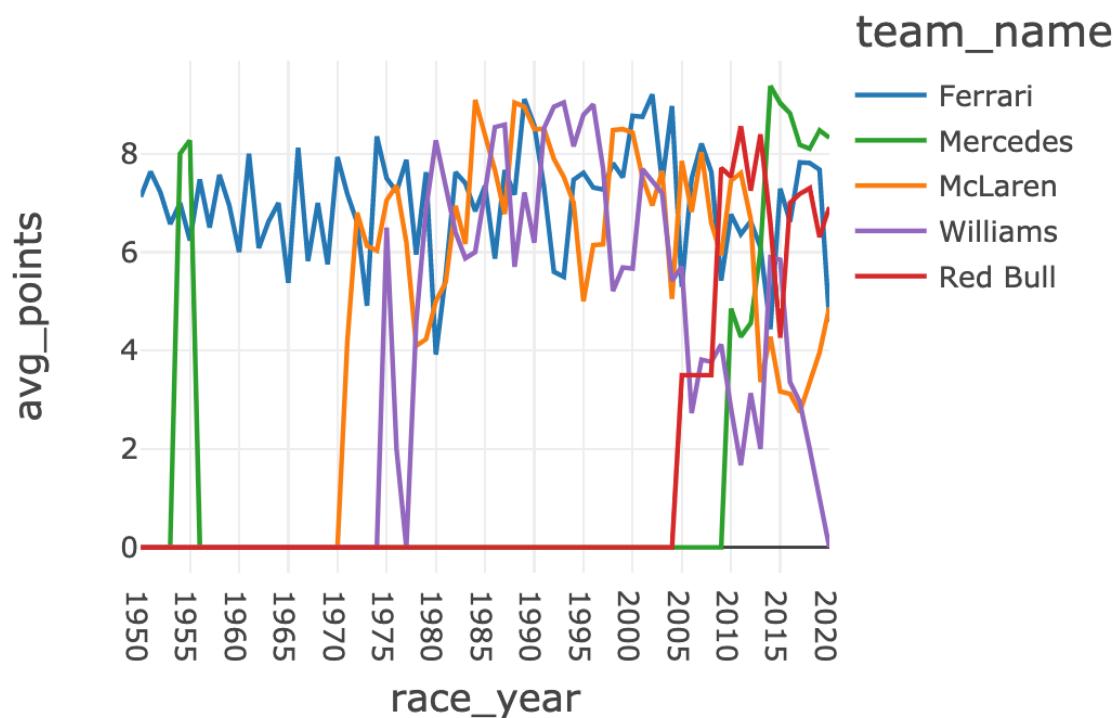
Along the way, we'll do some interesting analysis on the data to identify the most dominant drivers and the teams, in the history of Formula One.



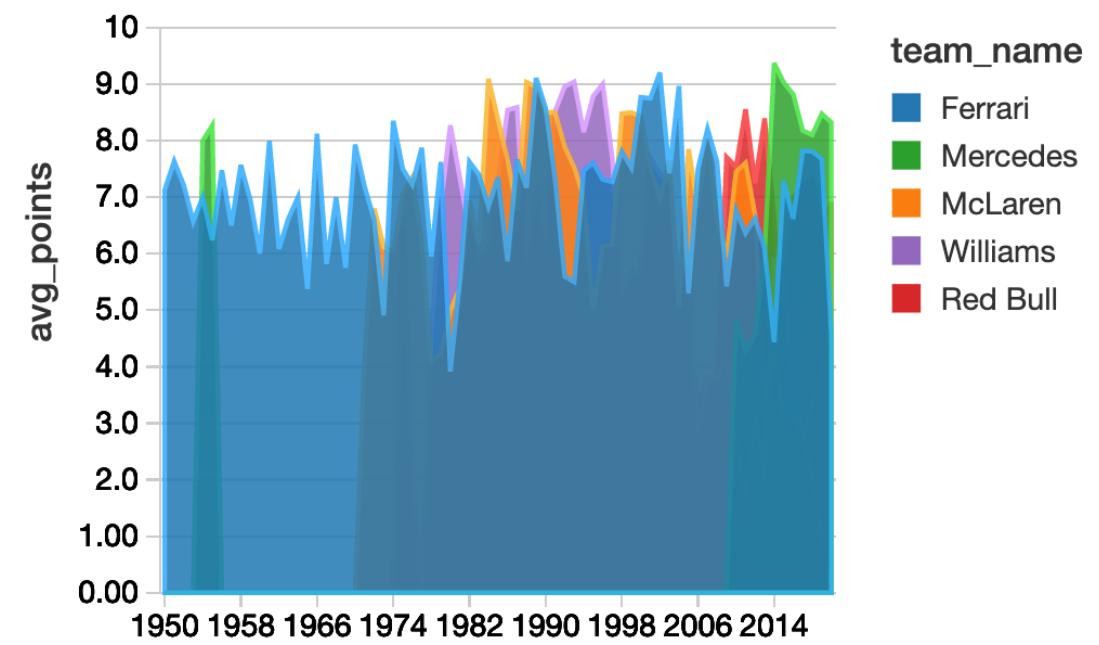
# Formula1 Cloud Data Platform

## Dominant Formula 1 Teams of All Time

Dominant Teams



Dominant Teams



# Who is this course for

**University students**

looking for a career in Data Engineering.

**IT Developers from other disciplines**

**AWS/ GCP/ On-prem Data Engineers**

Data Engineers and Data Warehouse developers currently working on, on premises technologies or other cloud platforms and want to learn Azure data technologies.

**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

There are 24 sections in this course and they can be split into six broad categories as shown here.

# Course Structure

## Oversviews

2.Azure Portal

3.Azure Databricks

9.Project Overview

10.Spark Overview

### Databricks

4. Clusters

5. Notebooks

6. Data Lake Access

7. Securing Access

8. Databricks Mounts

14. Jobs

### Spark (Python)

11. Data Ingestion 1

12. Data Ingestion 2

13. Data Ingestion 3

15. Transformation

16. Aggregations

19.Incremental Load

### Spark (SQL)

17. Temp Views

18. DDL

19. DML

20. Analysis

21.Incremental Load

### Delta Lake

22.Delta Lake

## Orchestration

23.Azure Data Factory

24.Connecting Other Tools



# Introduction to Azure Databricks



# Azure Databricks

These three offerings together makes Azure Databricks.

At the core of Azure Databricks is the open source distributed compute processing engine called Apache Spark, which is widely used in the industry for developing big data projects.



Databricks is a company created by the founders of Apache Spark, to make it easier to work with Spark by providing the necessary management layers.



Microsoft Azure

Microsoft makes the Databricks service available on its Azure Cloud platform as a first party service.

# Apache Spark

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

It's been used by the Internet giants such as Yahoo, eBay and Netflix for large scale data processing on multiple petabytes of data on clusters of thousands of nodes. Spark was built from the ground up to address the shortcomings of Hadoop. Hadoop was slow and inefficient for interactive and iterative computing jobs, and it was too complex to learn and develop.

On the other hand, Spark offers a much simpler, faster and easier APIs to develop on. Spark can be 100 X faster than Hadoop, for large scale data processing by exploiting In-memory computing and other optimizations.



It comes packaged with high level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows.

**100% Open source under Apache License**

**Simple and easy to use APIs**

**In-memory processing engine**

**Distributed computing Platform**

Similar to most other big data engines, Spark runs on a distributed computing platform.

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

Spark has a unified engine to support varying workloads. For example, it uses a single engine for streaming and batch workloads. It doesn't have separate one for each of those.

# Apache Spark Architecture

Combining all of these, Spark provides the unified platform for doing streaming, batch, machine learning and graph processing workloads using a single execution engine and a standard set of APIs.



Spark Core provides the APIs to create and manipulate these RDD collections.

Early development in Spark was done using these APIs, but it had its drawbacks.

It was difficult to use for complex operations, and it was difficult to optimize for Spark and mainly down to the developer to write the optimized code.

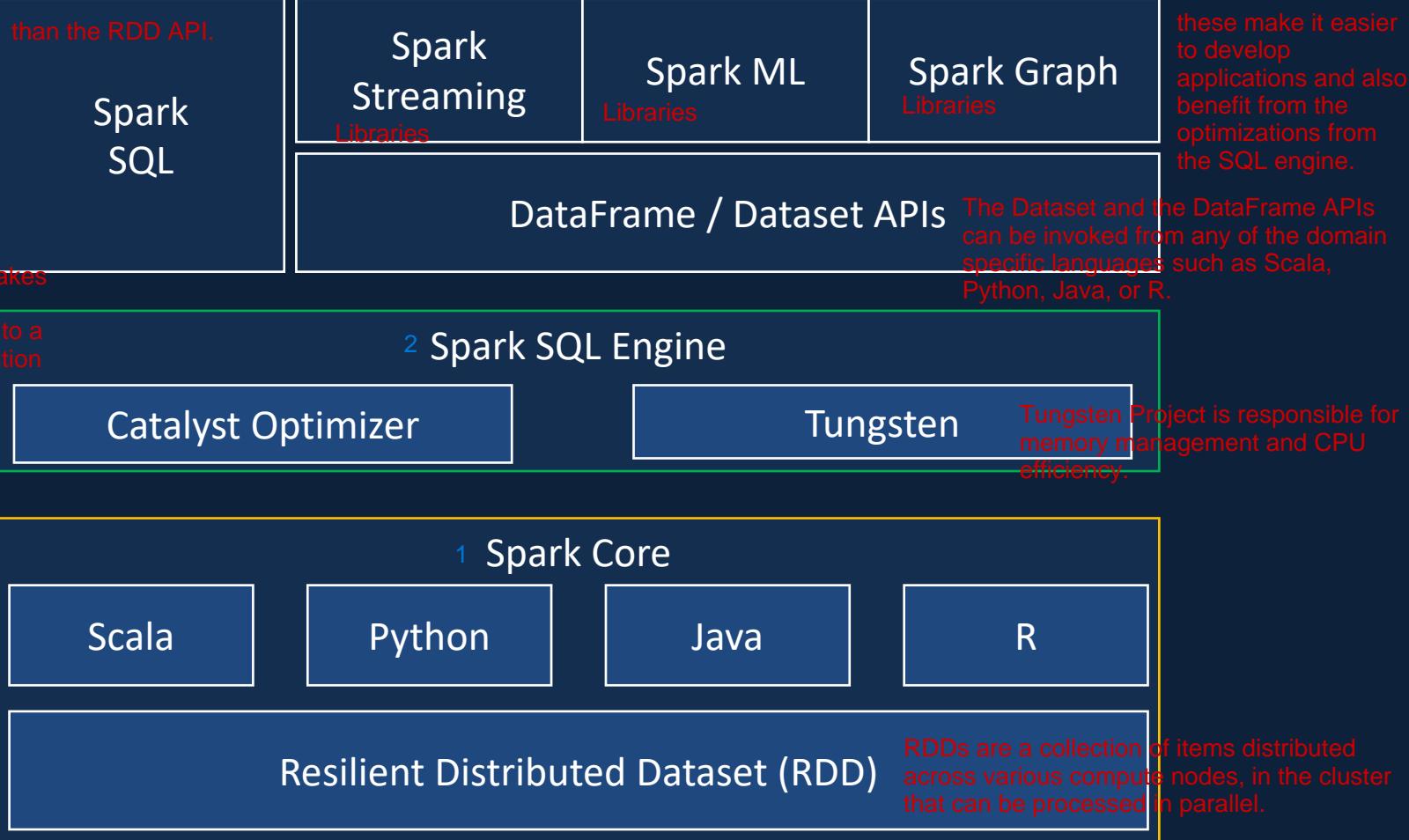
In order to optimize the workload, Spark introduced the SQL engine.

At the center of Spark Architecture is Spark Core. This contains the basic functionality of Spark.

The Spark Core takes care of scheduling tasks, memory management, fault recovery, communication with storage systems, etc. It's also home to Spark's main programming abstraction API called RDD or Resilient Distributed Datasets.

Also, Spark comes with its standalone resource manager, but you can choose other resource managers such as YARN, Apache Mesos and Spark Standalone, YARN, Apache Mesos, Kubernetes

So the recommended approach to develop applications in Spark, is to use these higher level APIs rather



Now we know Spark is a fast execution engine with an easy to use set of higher level APIs.

But, in order to work with Spark, we have to set up our own clusters, manage security, and also use third party products to write our programs.

That's where Databricks comes in.

# Databricks

Databricks is a company founded by the creators of Apache Spark to make it easier to work with Spark on the Cloud.

Databricks gives you the ability to spin up the Clusters with a few clicks.

You can choose the runtime, which is suitable for your needs.

For example, you can choose a runtime with ML libraries, support for GPU, etc. Also, you can choose from a wide range of Clusters ranging from general purpose, memory optimized, compute optimized, or GPU enabled.

Recent addition to Databricks is the SQL Analytics, which provides the data analyst a SQL based analytics environment.

This allows the analyst to explore data, create dashboards, schedule a regular refresh of the dashboard, etc..

Also, it comes with managed ML flow on Databricks, which allows us to manage the machine learning lifecycle, including experimentation, deployment, model registry, etc.

## MLFlow

## SQL Analytics

## Delta Lake

In order to provide ACID transaction capability, Databricks also comes with the Open Source project Delta Lake

## Clusters

In order for Spark to do its distributed computing, we need to spin up Clusters and install the software.



With the use of high metastore, Databricks also provides the ability to create databases and tables.

## Databases/ Tables

## Workspace/ Notebook

It provides a Jupyter Notebook style IDE with additional capabilities to create your application. Collaborate with your other colleagues and also integrate with configuration management tools such as Git.

## Administration Controls

It provides administration controls that you can use to restrict or provide access to your users, to the workspace, Clusters, etc.

## Optimized Spark (5x faster)

On top of this, Databricks provides the Spark runtime, which is highly optimized for the Databricks platform and known to be up to 5x faster than the Vanilla Apache Spark.

With the recent announcement from Google in February 2021, the Databricks Cloud platform is now available on all three major Cloud platforms, such as Microsoft Azure, AWS and Google Cloud.

But Azure's integration is deeper than others, Databricks is a first party service on Azure.

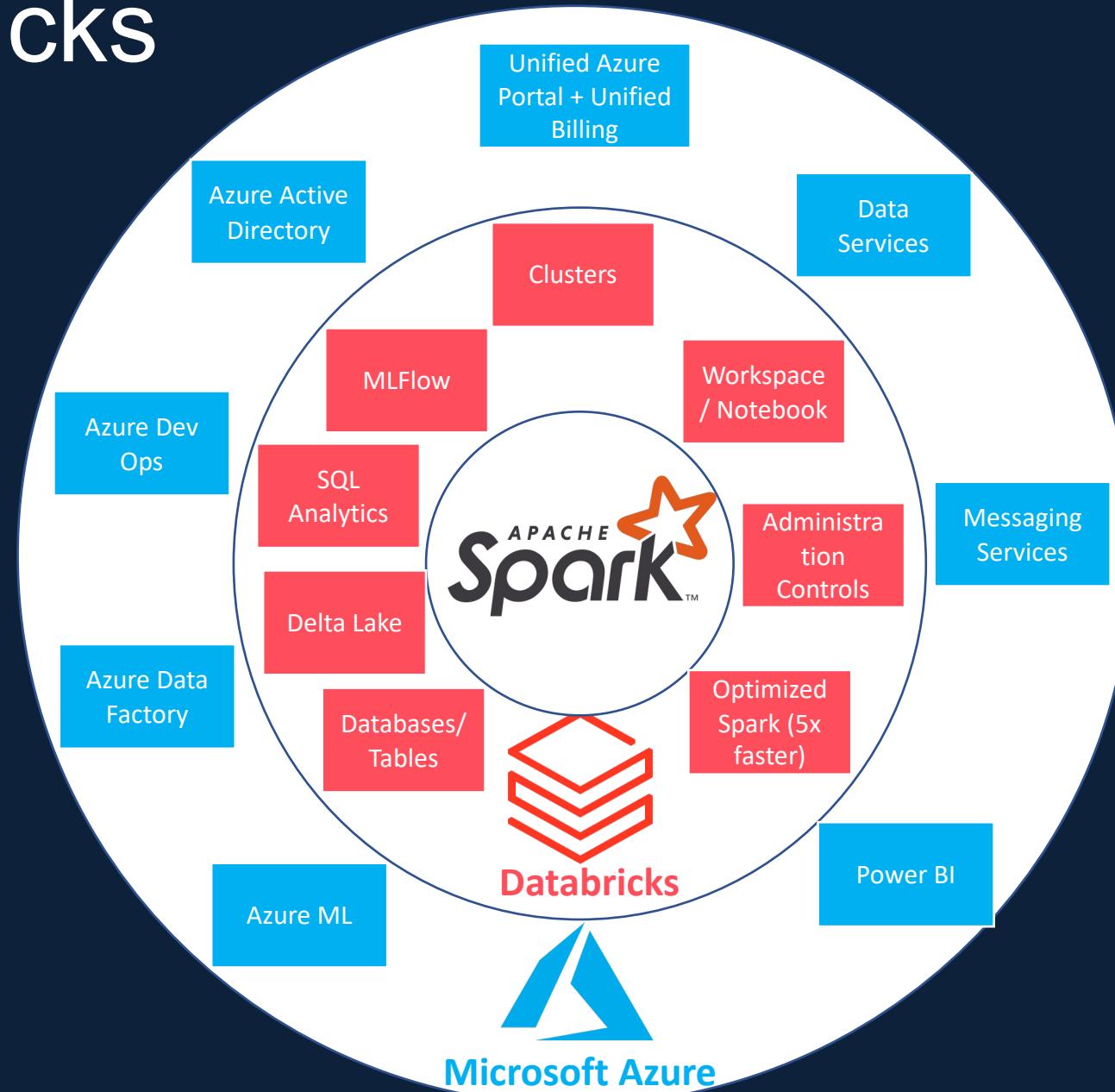
# Azure Databricks



# Azure Databricks

Databricks is a first party service on Azure. It means, on Azure you will be buying Databricks directly from Microsoft and all support requests are handled by Microsoft. As a result, it provides a unified Azure Portal for Databricks and a single unified bill for all your Azure services, including Databricks. Azure Databricks leverages, Azure security and seamlessly integrates with Azure Active Directory and single sign on.

So just to summarize, Azure Databricks is a spark based unified data analytics, platform as a service offering, that's optimized for the Microsoft Azure Cloud.



# Creating Azure Databricks Service



To create Azure Databricks

<https://portal.azure.com/#home> > login> dropdown menu (left side of nav. bar) > +create resource > search Azure Databricks > create

In Region choose --> UK South

Pricing Tier ---> In between Premium and standard tier

V - 8

There is no upfront cost for creating the workspace. You're only charged for the compute resources used within Databricks and also the storage within Azure.

The Databricks SQL only comes in premium tier, all others are exactly the same. In terms of performance, there is not much difference between the two tiers. Databricks recently added Unity Catalog that provides Cross Workspace Governance, and that's only available in premium tier.

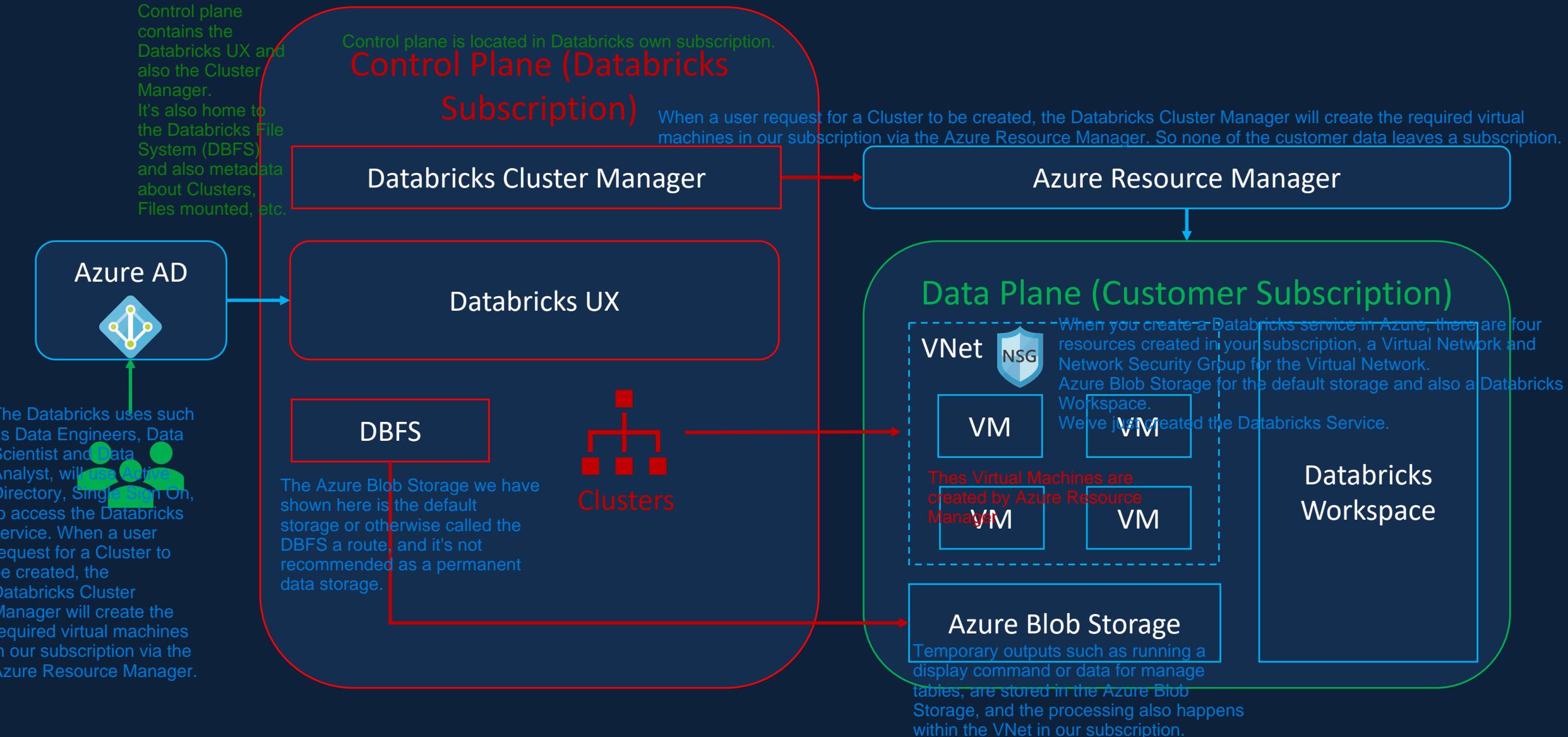
Also, things like Delta Sharing, Audit Logs and Cluster Policies are only available in premium tier.

Premium Tier also provides additional security options such as Role-based access control, Active Directory credential passthrough, etc.

As a developer, you may not need to understand how it works under the hood, but it is better to understand the architecture, so that you can reason out about where your data is stored and also where your compute is located.

# Azure Databricks Architecture

Databricks Architecture is basically split into two parts, one called the Control Plane and another one called the Data Plane.



# Databricks Workspace Components

Notebooks

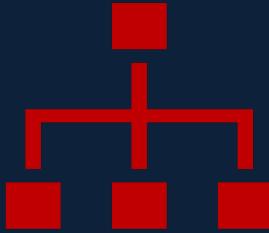
Data

Clusters

Jobs

Models

# Databricks Clusters



What is Databricks Cluster

Cluster Types

Cluster Configuration

Creating a cluster

Pricing

Cost Control

Cluster Pools

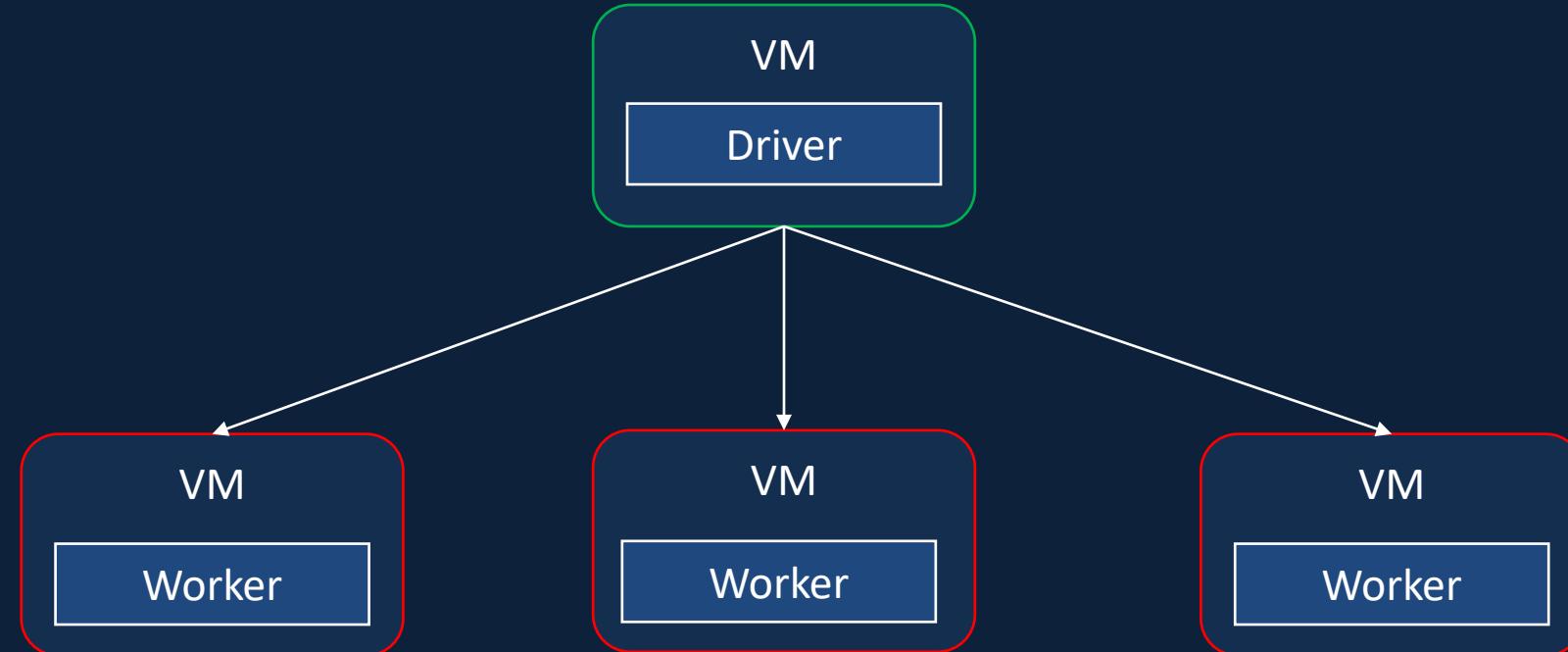
Cluster Policy

# Databricks Cluster

A Cluster is basically a collection of Virtual Machines. In a Cluster, there is usually a Driver node, which orchestrates the tasks performed by one or more worker nodes.

Clusters allow us to treat this group of computers, as a single compute engine via the Driver node.

Databricks Clusters enable us to run different types of workloads, such as ETL for Data Engineering, Data Science and Machine Learning workloads.



# Cluster Types

All Purpose	Job Cluster
Created manually <small>via the Graphical User Interface, the CLI or the API.</small>	Created by Jobs
Persistent <small>they can be terminated and restarted at any point in time,</small>	Terminated at the end of the job <small>They cannot be restarted. So they're no longer usable once the job has been completed.</small>
Suitable for interactive workloads	Suitable for automated workloads <small>such as running an ETL pipeline or Machine Learning workflow at a regular interval.</small>
Shared among many users <small>and they are good for collaborative analysis,</small>	Isolated just for the job
Expensive to run	Cheaper to run

# Cluster Configuration

When we come to create the Cluster, We will be presented with a number of configuration options as shown here.

## Cluster Configuration

**Policy** 

Unrestricted | 

Multi node  Single node

**Access mode**  **Single user access** 

Single user |  Ramesh Retnasamy (az.adm1... | 

### Performance

**Databricks runtime version** 

Runtime: 11.3 LTS (Scala 2.12, Spark 3.3.0) | 

Use Photon Acceleration 

Worker type	Min workers	Max workers	
Standard_DS3_v2	14 GB Memory, 4 Cores   	2	8

  Spot instances 

**Driver type**

Same as worker 14 GB Memory, 4 Cores | 

Enable autoscaling 

Terminate after  minutes of inactivity 

# Cluster Configuration

## Single/ Multi Node

Multi Node Cluster will have one Driver Node and one or more Worker Nodes. When you run a Spark Job against a Multi Node Cluster, the Driver Node will distribute the tasks to run on the Worker Nodes in parallel, and returns the result. They give us the ability to horizontally scale the Cluster depending on your workload.

We can basically keep adding Worker Nodes as we need. These are the default type of Clusters used for Spark Jobs and suitable for large workloads.

### Multi Node

### Single Node

Single Node Cluster will have only one node, which is the Driver Node and there are no Worker Nodes.

Even though, there are no Worker Nodes, Single Node Clusters also supports Spark workloads.

When you run a Spark Job, the Driver Node acts as both the driver and the worker

As there are no Worker Nodes, the Single Node Clusters are not horizontally scalable, so they're not suitable for large ETL workloads. They're mainly targeted for lightweight Machine Learning and Data Analysis workloads which don't require any distributed compute capacity.

# Cluster Configuration

## Single/ Multi Node

### Access Mode

Shared access mode allows the Cluster to be shared amongst more than one user, but it provides process isolation. Each process gets its environment, so one process can't see the data or the credential used by the other one.

It's only available on premium workspaces.  
Also, it only supports Python and SQL workloads.

No Isolation Shared also allows the Cluster to be shared amongst more than one user.  
It's available on both standard and premium workspaces.  
Also, it supports all four languages Python, Scala, SQL and R.

The main difference between this and the Shared access mode is that, No Isolation Shared access mode doesn't provide any process isolation. So failure in one user's process may affect the others.  
Also, they don't offer any task preemption, so one running process may use all the resources and the others may fail.

And most importantly, as everything is shared, it's considered less secure

As the name suggests, Single User access mode only allows a single user to access the Cluster.

### Single User

Only One User Access  
Supports Python, SQL, Scala, R

Supports all 4 languages

### Shared

Multiple User Access  
Only available in Premium. Supports Python, SQL

### No Isolation Shared

Multiple User Access  
Supports Python, SQL

### Custom

Legacy Configuration

Custom access mode is not an option, when you create the Cluster using the latest user interface. You're only likely to see this if you have already created a Cluster using the legacy configurations.

# Cluster Configuration

Single/ Multi Node

Access Mode

## Databricks Runtime

Databricks runtimes are the set of core libraries that run on Databricks Clusters.  
At the time of recording, Databricks offers four types of runtimes.

Databricks Runtime includes an optimized version of Apache Spark Library, Java, Scala, Python and R Libraries, Ubuntu and its accompanying system Libraries, GPU Libraries for GPU enabled Clusters, Delta Lake Libraries and also other Libraries for Databricks services that integrate with other components of the platform such as Notebooks, Jobs and Cluster Manager.

Spark

### Databricks Runtime

Scala, Java,  
Python, R

Ubuntu  
Libraries

GPU  
Libraries

Delta Lake

Other Databricks Services

Everything from  
Databricks runtime

### Databricks Runtime ML

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

Everything from  
Databricks runtime

### Photon Runtime

#### Photon Engine

The Photon Engine, which is the Databricks native vectorized query engine, that runs SQL workloads faster and reduces your cost per workload.

Databricks Runtime Light is the runtime option for only jobs not requiring advanced features such as auto scaling, reliability and improved performance.

Also, it's only suitable for Automated Workloads. You can't use it for Interactive Workloads or Notebook Jobs.

**Databricks Runtime Light**  
Runtime option for only jobs not requiring advanced features

# Cluster Configuration

Single/ Multi Node

Access Mode

Databricks Runtime

## Auto Termination

Auto Termination is a nice feature that will avoid unnecessary costs on idle Clusters.

It's especially useful on Ad-hoc clusters for preventing them, running during evenings and weekends when they're not in use.

You can specify when to terminate your Databricks Cluster, if the cluster has not been in use. It will be terminated after the number of minutes specified.

When you create a Multi Node Cluster, you can specify the minimum and the maximum number of Worker Nodes.

## Auto Termination

- Terminates the cluster after X minutes of inactivity
- Default value for Single Node and Standard clusters is 120 minutes
- Users can specify a value between 10 and 10000 mins as the duration

# Cluster Configuration

Single/ Multi Node

Auto Scaling will automatically add or remove nodes from the Cluster depending on your workload. This can result in optimum utilization of the Cluster.

This is especially useful if you're unsure about the workload upfront or your workload changes throughout the process.

Access Mode

They're not recommended for streaming workloads, even if specified Databricks defaults to the maximum number of Worker Nodes.

A streaming workload refers to a scenario where a large amount of data is processed continuously in real-time, instead of processing it in batches or storing it for later use. This concept is widely used in applications like video streaming, social media, and big data processing where timely analysis and action are crucial.

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

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

## Cluster VM Type/ Size

There are a wide array of Azure VM types available for us to use.  
Databricks groups them into small number of easy to understand groups.

### Memory Optimized

Memory Optimized instance types are recommended for memory intensive applications. For example, a Machine Learning workload that caches a lot of data in memory.

### Compute Optimized

Compute Optimized instance types can be useful for structured streaming applications, where you need to make sure that the processing rate is above the input rate at peak times of the day.

These can also be used for Distributed Analytics and Data Science Applications.

### Storage Optimized

Storage Optimized instance types are recommended for use cases requiring high disk throughput and I/O.

### General Purpose

General Purpose instance types are recommended for Enterprise Grade applications and analytics with In-memory caching.

### GPU Accelerated

GPU Accelerated instance types are recommended for Deep Learning Models, that are data and compute intensive.

# Cluster Configuration

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

The final configuration option is Cluster Policy. As you have seen, there are a lot of options to choose from when you are configuring a Cluster. This could easily overwhelm a Data Engineer or a Machine Learning Engineer, and creating Clusters become the sole responsibility of the administrator. Because it's too difficult to configure for a Standard Data Engineer or a Machine Learning Engineer. Also, without careful consideration, users could accidentally create Clusters which are oversized and too expensive to run. Cluster Policies help us avoid these common issues.

Administrators can create Cluster policies with restrictions, and assign them to users or groups.

You can change the minimum and the maximum values depending on your workload. In this example, when the cluster starts up, it will be allocated with two Worker nodes and additional workers are added depending on the workload. But it'll never exceed eight. This ensures that you're only paying for additional capacity when there is a need, and also your expenses are capped at eight nodes. You can disable auto scaling by clicking on the tick box here.

Min workers      Max workers

2      8

Spot instances ?

You can also request to use unused Azure capacity via Spot instances if it's available, by selecting the tick box here. This will save you cost of running your application, but please be mindful that you could be evicted, from Spot instances when they become unavailable. I would recommend using them, for development purposes or, non critical workloads only.

The screenshot shows the 'Cluster Configuration' page. At the top, there's a dropdown for 'Policy' set to 'Unrestricted'. Below it, there are two radio buttons: 'Multi node' (selected) and 'Single node'. Under 'Access mode', there are two dropdowns: 'Single user' (selected) and 'Ramesh Retnasamy (az.adm1...'. The 'Performance' section includes 'Databricks runtime version' (Runtime: 11.3 LTS (Scala 2.12, Spark 3.3.0)), a checkbox for 'Use Photon Acceleration' (unchecked), and sections for 'Worker type' (Standard\_DS3\_v2, 14 GB Memory, 4 Cores), 'Driver type' (Same as worker, 14 GB Memory, 4 Cores), and checkboxes for 'Enable autoscaling' (checked) and 'Terminate after 120 minutes of inactivity' (checked). On the right side, there are 'Min workers' (2) and 'Max workers' (8) input fields, and a checkbox for 'Spot instances' (unchecked).

# Cluster Configuration

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

**Cluster Configuration**

**Policy** ?

Personal Compute

**Single user access** ?

Ramesh Retnasamy (az.adm1@outlook.com)

**Performance**

**Databricks runtime version** ?

Runtime: 11.3 LTS ML (Scala 2.12, Spark 3.3.0)

**Node type** ?

Standard\_DS3\_v2      14 GB Memory, 4 Cores

Terminate after  minutes of inactivity ?

eg.: A Personal Compute cluster policy has took up the option of Multi Node, and the user can only create a Single Node Cluster.  
Also, it defaulted the runtime version to ML Runtime, limited the node types and also Auto Termination set to 20 minutes.

# Cluster Configuration

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

In summary, Cluster policies simplify the user interface, thus enabling standard users to create Clusters and take away the need for administrators to be involved in every decision.  
And most importantly, it achieves cost control by limiting the maximum size of the Clusters  
But please note that this is only available on premium tier.

Simplifies the user interface

Enables standard users to create clusters

Achieves cost control

Only available on premium tier

# Creating Databricks Cluster



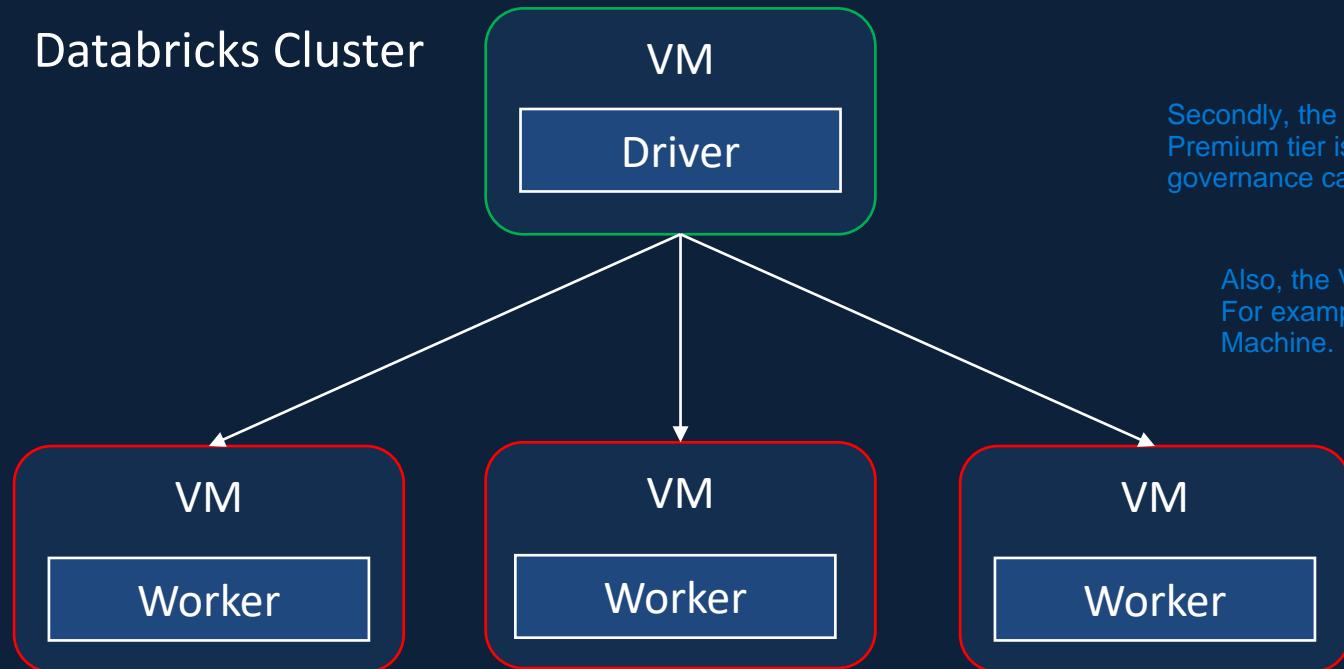
Please note that the Azure Pricing calculator also gives you the total cost of running the Cluster, but knowing how it's calculated gives you the ability to tweak the options and to get the best possible price for your needs.

# Azure Databricks Cluster Pricing



# Azure Databricks Pricing Factors

Databricks Cluster



Price of a Databricks Cluster depends on a number of factors.

Firstly, it depends on the type of Workload, for example, whether we're using All Purpose compute, Jobs compute, Databricks SQL, Photon Engine, etc..

**Workload (All Purpose/ Jobs/ SQL/ Photon)**

Secondly, the tier of the Databricks workspace influences the pricing. Premium tier is more expensive than standard, but offers a number of security and governance capabilities.

**Tier (Premium/ Standard)**

Also, the Virtual Machine type influences the price. For example, a GPU enabled Virtual Machine is more expensive than a General Purpose Virtual Machine.

**VM Type (General Purpose/ GPU/ Optimized)**

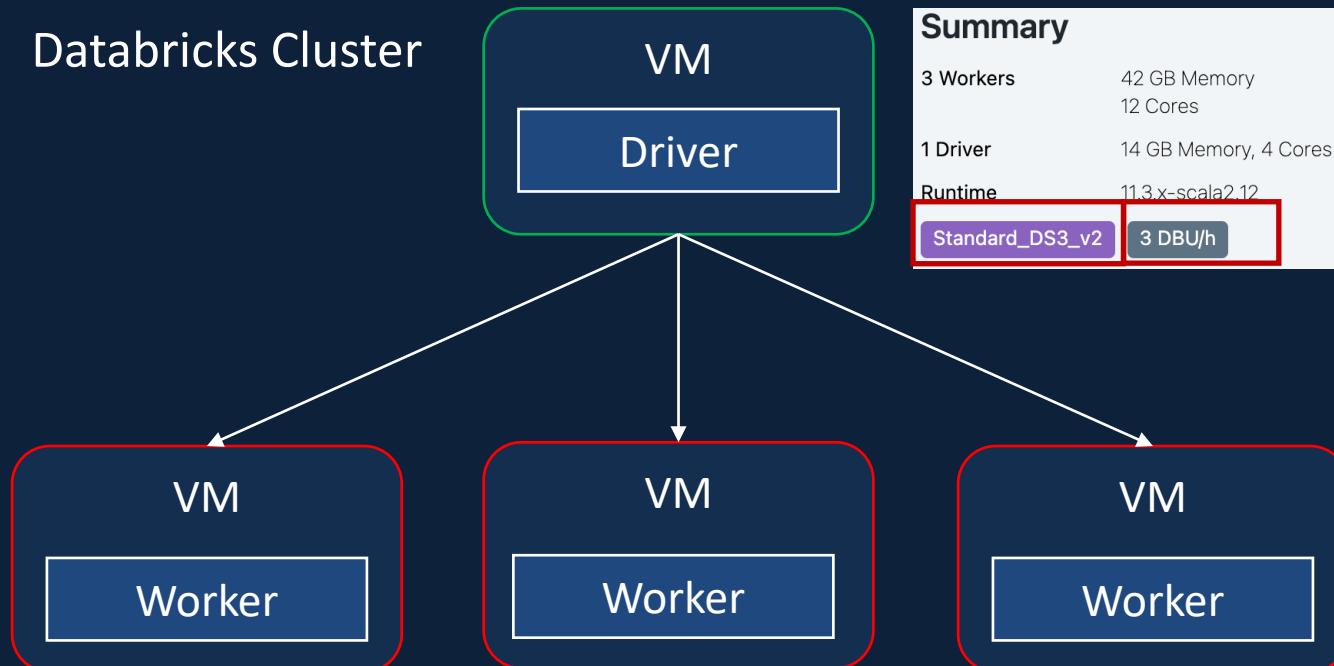
**Purchase Plan (Pay As You Go/ Pre-Purchase)**

Finally, Azure offers a number of pre-purchase plans under which you get discounts for buying compute capacity in advance.

# Azure Databricks Pricing Calculation

But how do we know the actual price that we need to pay for running our Cluster?

There are two things we need to look out for.



Firstly

A Databricks Unit (DBU) is a normalized unit of processing power on the Databricks Lakehouse Platform used for measurement and pricing purposes

For this example Cluster, the number of Databricks units is three as shown in the summary.  
We need to pay for these three Databricks units.

Secondly, the Virtual Machines themselves.

We need to pay for the Virtual Machine separately too.

In this example, we are using four instances of Virtual Machine type standard DS3\_v2 Linux machines.

So we need to calculate that as well.

The total of these two components will be the cost of running the Cluster.

In addition to this, there will be some small charges for the virtual network, public IP addresses, etc. But they're generally negligible compared to the cost of running the Cluster.

# Azure Databricks Pricing Calculation

A simple calculator may look like this.

Firstly, we multiply the number of Databricks units by price for the workload and pricing tier to get the DBU cost or the Databricks Unit Cost.

$$\boxed{\text{No of DBU}} \times \boxed{\text{Price based on workload/ tier}} = \boxed{\text{DBU Cost}}$$

We can then identify the price of the Virtual Machine for the Driver node to get the cost of the Driver node.

$$\boxed{1 (\text{Driver})} \times \boxed{\text{Price of VM}} = \boxed{\text{Driver Node Cost}}$$

We can then add all these three outputs to get the total cost of the Cluster.

$$+ \boxed{\text{DBU Cost}} + \boxed{\text{Driver Node Cost}} = \boxed{\text{Total Cost of the Cluster}}$$

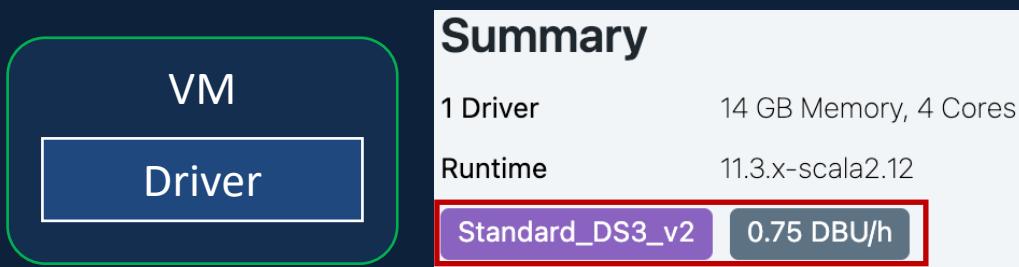
Then we can identify the price of the Worker node and multiply that by the number of Worker nodes to get the total cost of the Worker nodes.

$$\boxed{\text{No of workers}} \times \boxed{\text{Price of VM}} = \boxed{\text{Worker Node Cost}}$$

You might be wondering why I'm doing the Driver and the Worker nodes are separate lines.  
In our example, we use the same node type for both the Driver and the Worker nodes.  
But it's not uncommon to have different VM types for the Driver and the Worker nodes depending on your workload.

# Azure Databricks Pricing Calculation

## Single Node Cluster



Workload - All Purpose

Pricing Tier - Premium

VM Type - General Purpose Standard DS3\_V2

Purchase Plan - Pay As You Go

# Azure Databricks Pricing Calculation

Workload	DBU prices—standard tier	DBU prices—premium tier
All-Purpose Compute**	\$0.40/DBU-hour	\$0.55/DBU-hour
Jobs Compute**	\$0.15/DBU-hour	\$0.30/DBU-hour
Jobs Light Compute	\$0.07/DBU-hour	\$0.22/DBU-hour
SQL Compute	-	\$0.22/DBU-hour
SQL Pro Compute	-	\$0.37/DBU-hour
Serverless SQL	-	\$0.475/DBU-hour

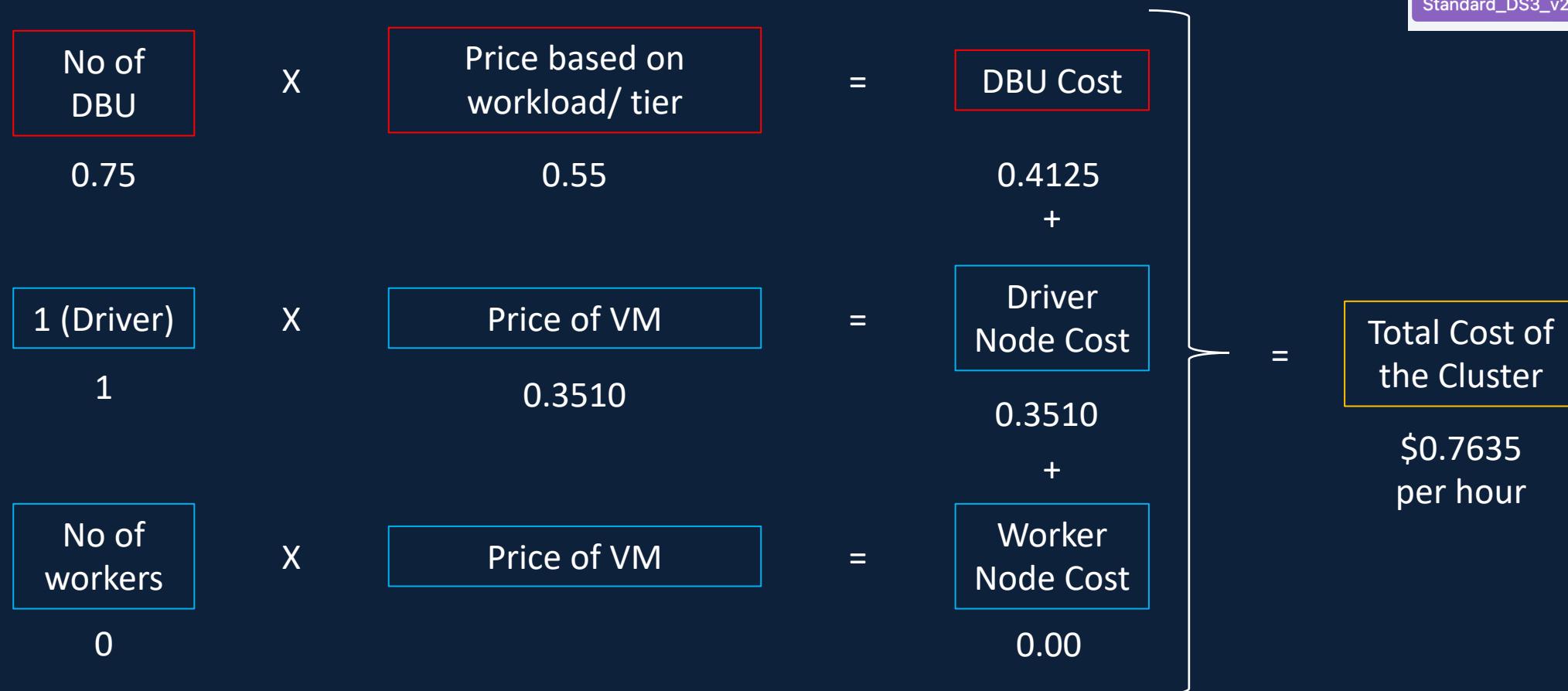
<https://azure.microsoft.com/en-gb/pricing/details/databricks/>

# Azure Databricks Pricing Calculation

Instance	vCPU(\$)	RAM	Temporary storage	Pay as you go	1 year savings plan	3 year savings plan	Spot	Add to estimate
DS1 v2	1	3.5 GiB	7 GiB	\$0.0878/hour	\$0.0760/hour ~13% savings	\$0.0600/hour ~31% savings	\$0.0105/hour ~87% savings	<button>+</button>
DS2 v2	2	7 GiB	14 GiB	\$0.1760/hour	\$0.1524/hour ~13% savings	\$0.1203/hour ~31% savings	\$0.0211/hour ~87% savings	<button>+</button>
DS3 v2	4	14 GiB	28 GiB	\$0.3510/hour	\$0.3040/hour ~13% savings	\$0.2399/hour ~31% savings	\$0.0422/hour ~87% savings	<button>+</button>
DS4 v2	8	28 GiB	56 GiB	\$0.7020/hour	\$0.6080/hour ~13% savings	\$0.4798/hour ~31% savings	\$0.0843/hour ~87% savings	<button>+</button>
DS5 v2	16	56 GiB	112 GiB	\$1.4050/hour	\$1.2169/hour ~13% savings	\$0.9603/hour ~31% savings	\$0.1687/hour ~87% savings	<button>+</button>

<https://azure.microsoft.com/en-gb/pricing/details/virtual-machines/linux/#pricing>

# Azure Databricks Pricing Calculation



# Estimated cost for doing the course

Azure Data Lake Storage

Azure Data Factory

Azure Databricks Job Cluster

Azure Databricks Cluster Pool

Azure Databricks All Purpose Cluster

\$0.76 per hour for a small single node cluster on premium tier

Depends on the number of hours in use

Past student experience – 20 to 30 hours to complete the course

Past *Pay As You Go* student experience – \$15 to \$25

Within the credit offered by Free / Student Subscription

# Cost Control

Service	Action to be taken
Azure Data Lake Storage	None – Cost Negligible
Azure Data Factory	None - Billed only for execution of the pipeline
Azure Databricks Job Cluster	None - Destroyed once the job completes
Azure Databricks Cluster Pool	Delete the cluster pool at end of the lesson
Azure Databricks All Purpose Cluster	Set Auto Termination to 20 minutes

Set budget alerts on your subscription

# Cluster Pools

Single/ Multi Node

Access Mode

Databricks Runtime

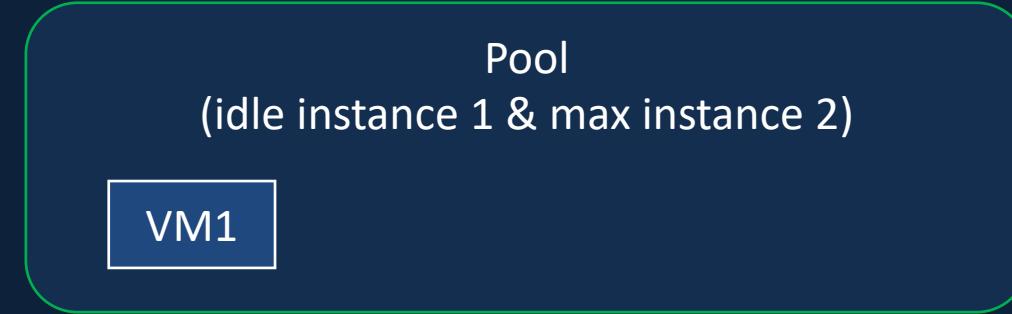
Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

Cluster Pool



# Cluster Pools

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

Cluster Pool

Pool  
(idle instances 1 & max instances 2)

VM2

Cluster 1

VM1

# Cluster Pools

Single/ Multi Node

Access Mode

Databricks Runtime

Auto Termination

Auto Scaling

Cluster VM Type/ Size

Cluster Policy

Cluster Pool

Pool  
(idle instances 1 & max instances 2)

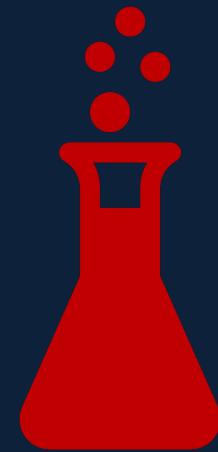
Cluster 1

VM1

Cluster 2

VM2

# Creating Cluster Pool

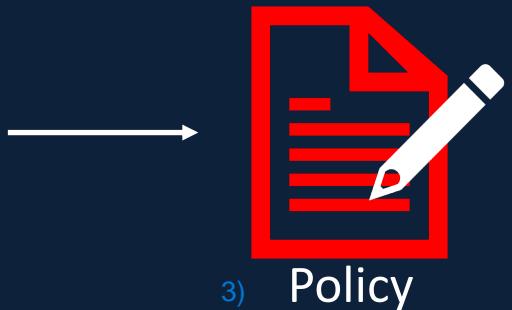


# Cluster Policy



# Cluster Policy

4) An administrator can create a policy and assign that to a set of users.

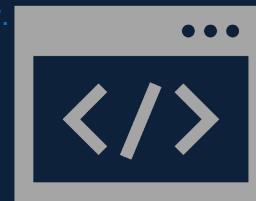


Admin

5) When the user chooses the policy, they'll now have a Simpler User Interface because some of the options are now hidden or pre-populated with fixed values,

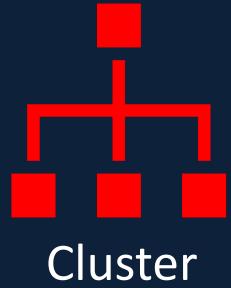


1) As we've seen previously, Cluster configuration is quite complicated. Creating the wrong type of Cluster could increase the cost of running the project and also can affect data security.



Config  
GUI

2) So if we expect standard users to create the Cluster and keep the cost down, we should simplify the interface and make it easier for them. That's where Cluster Policies come in.



Cluster

Policies are basically a set of rules which allows us to hide or take out the configuration options, that are not required on the user interface.

## Hide Attributes

They allow us to fix the values of some of the configuration parameters and restrict access for the users so that they're not changing them.

## Fix Values

Also, in a set of values, a policy allows us to select default values on the user interface.

## Set Default Values

And finally, cluster policies take away the need for an administrator to be involved in every cluster creation decision and empowers standard users to create them.

## Simple User Interface

## Achieve Cost Control

## Standardize Cluster Configs

## Empowers standard users

6) they'll now have a Simpler User Interface because some of the options are now hidden or pre-populated with fixed values. Only allowing the users to select certain type of nodes or setting Auto Termination by Default, allows us to keep the cost down.

It helps us having standard type of Clusters created by all users in a specific group.

# Cluster Policy



It's a great feature, but please note that it is in public preview, at the time of recording in December 2022. So it's lacking features and being developed, so please expect to see some fixes and improvements over time.

**Public Preview (December 2022)**

A public preview is a pre-release version of a product, service, or feature that is made available to a limited audience, typically the general public, for testing and feedback purposes.

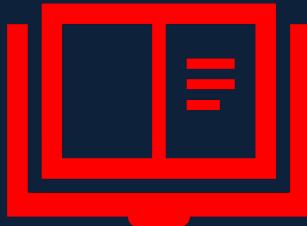
**Only available on Premium Tier**

Also, most importantly, cluster policies are only available for workspaces created in premium tier.

# Create Cluster Policy



# Databricks Notebooks



What's a notebook

Creating a notebook

I'll show you how to create a notebook in Databricks and attach to a Cluster, Execute commands, etc.

Magic Commands

We'll then look at how magic commands help us mix and match, a writing code in different languages within the same notebook.

Databricks Utilities

I will then take you through the various Databricks Utilities available and see them in action.

Import Project Solution Notebooks

Finally, I'll show you how to import the course project that I developed while teaching this course.

# Creating Notebooks

Databricks offers a Jupyter style notebook, with some additional capabilities to carry out development on its environment.

Basically, a notebook is a collection of cells, that run commands on a Databricks Cluster.  
The primary purpose of the Magic Commands, is to override the default language in a Notebook.



# Magic Commands



The primary purpose of the Magic Commands, is to override the default language in a Notebook.

For example, you may want to switch from Python to Scala within the same notebook.

There are also some auxiliary magic commands which allow us to create documentation in our notebooks and access the file system, etc..

# Databricks Utilities

Databricks have been coming up with a number of utilities recently.

Some are in preview and others are available for general availability.

In the next section of the course,

we'll be using the File System Utilities to mount containers from Azure Data Lake Storage into Databricks.



Databricks Utilities make it easier to combine different types of tasks in a Single notebook.

For example, they allow us to combine file operations with ETL tasks.

These utilities can only be run from Python, Scala or R cells in a Notebook.

They cannot be run from a SQL cell.

Amongst those, the following are the most commonly used utilities.

## File System Utilities

We've already seen the File System Utilities in the last lesson. It allows us to access databricks file system from a notebook and you can use various file system level operations.

Also, we'll be using Secret Utilities to get the secrets from Azure Key Vault.

## Secrets Utilities

Secrets Utilities allow us to get secret values from secrets, which are stored in secret scopes backed by Databricks or Azure Key Vault.

## Widget Utilities

Widget Utilities allows us to parameterized notebooks so that a calling notebook or another application, for example, a Data Factory Pipeline can pass a parameter value to the notebook at runtime. This is really useful to make a notebook reusable.

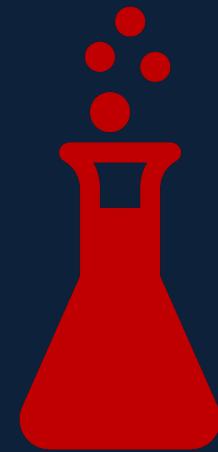
As part of the Formula One project development, we will use Widget Utilities to pass parameters into notebooks and Workflow Utilities to chain notebooks together.

## Notebook Workflow Utilities

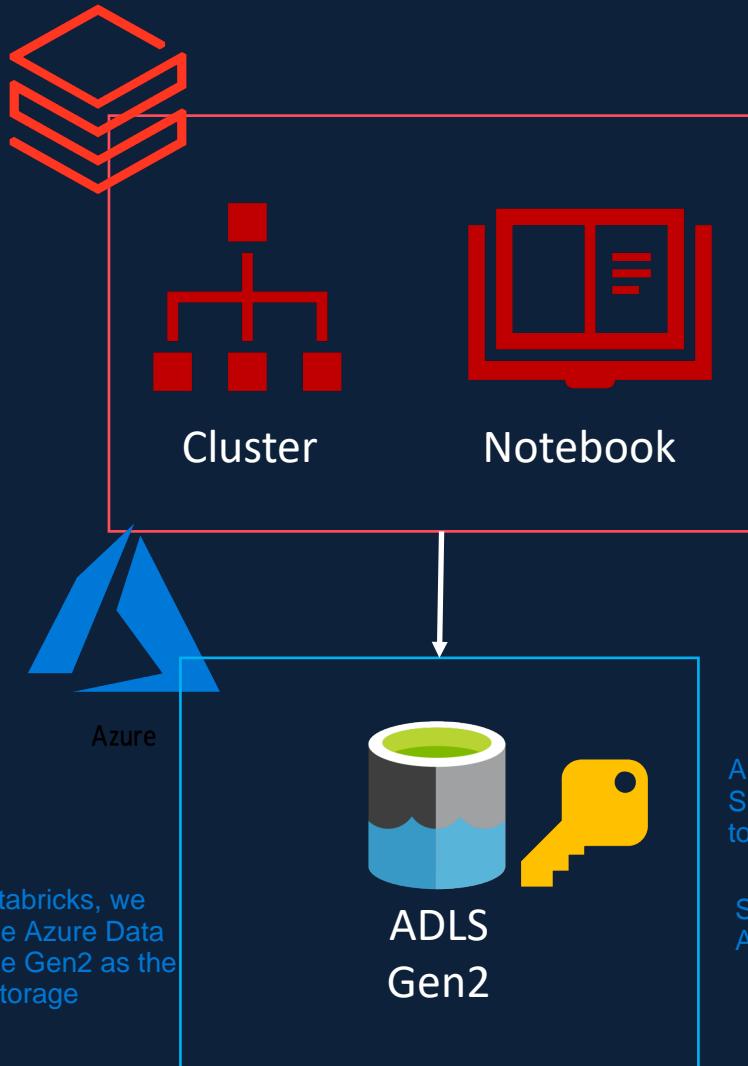
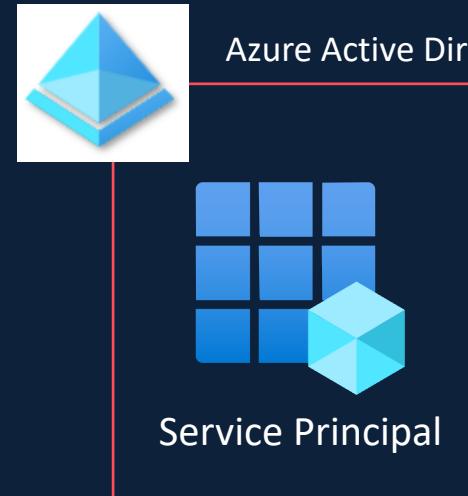
Notebook Workflow Utilities allow us to invoke one notebook from another and chain them together.

# Databricks Utilities

Now that we have a good understanding about Databricks Notebooks, which allows us to write code and also Clusters that give us the compute capacity required to execute notebooks, it's time to discuss about how to access the data storage from Databricks. So that we can process the data and gain insights from that.



# Access Azure Data Lake - Section Overview



Each Azure storage account comes with an Access Key, that we can use to access the storage account.

## Storage Access Keys

Also, we can generate a special kind of key called Shared Access Signature or otherwise referred to as SAS token, and we can use that to access the storage account.

## Shared Access Signature (SAS Token)

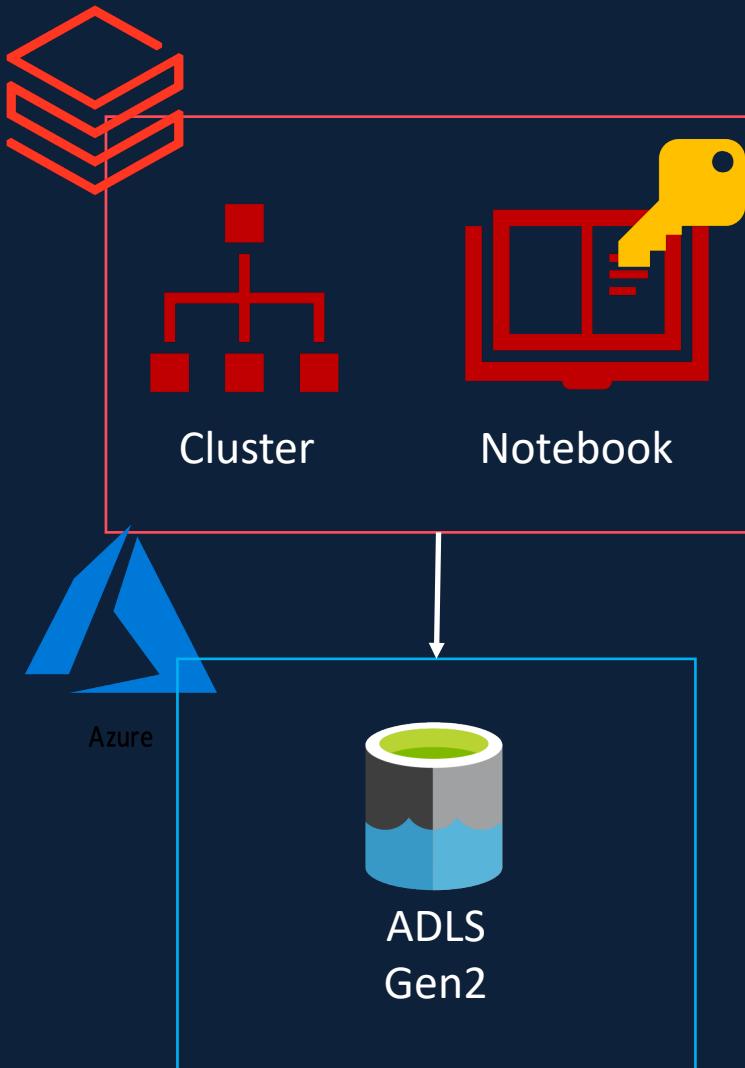
SAS tokens lets us manage access at a more granular level than the Access Key.

## Service Principal

We can also create a Service Principle and give the required access for the Data Lake to the Service Principle and use those credentials to access the storage account.

# Access Azure Data Lake - Section Overview

All of these three options can take two forms.



The first one is to use these credentials in the notebook and authenticate to the Data Lake.  
The authentication in this scenario will be valid just for the duration of the session, i.e. until the notebook has been detached to the cluster.  
This is called Session Scoped Authentication.

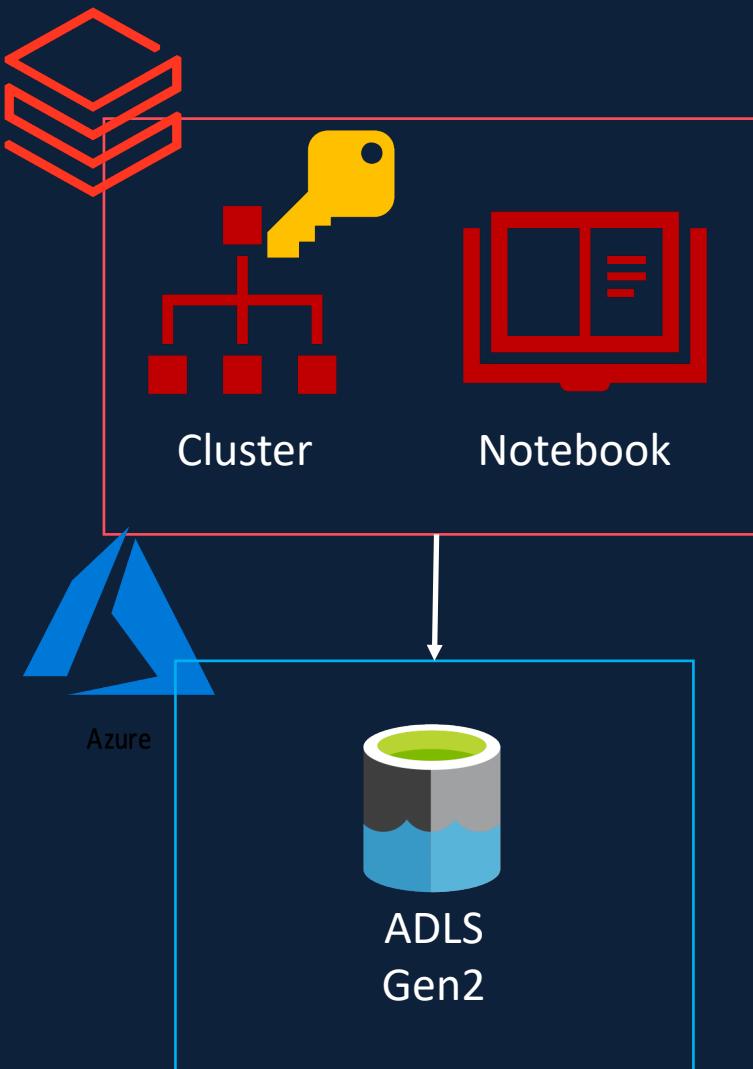
**Session Scoped Authentication**

**Storage Access Keys**

**Shared Access Signature (SAS Token)**

**Service Principal**

# Access Azure Data Lake - Section Overview



The other option is to use these credentials in the Cluster and authenticate from the Cluster.

The authentication will happen when the Cluster starts and it will be valid until the Cluster has been terminated.

All the notebooks connected to this Cluster will have access to the data.  
This is called Cluster Scoped Authentication.

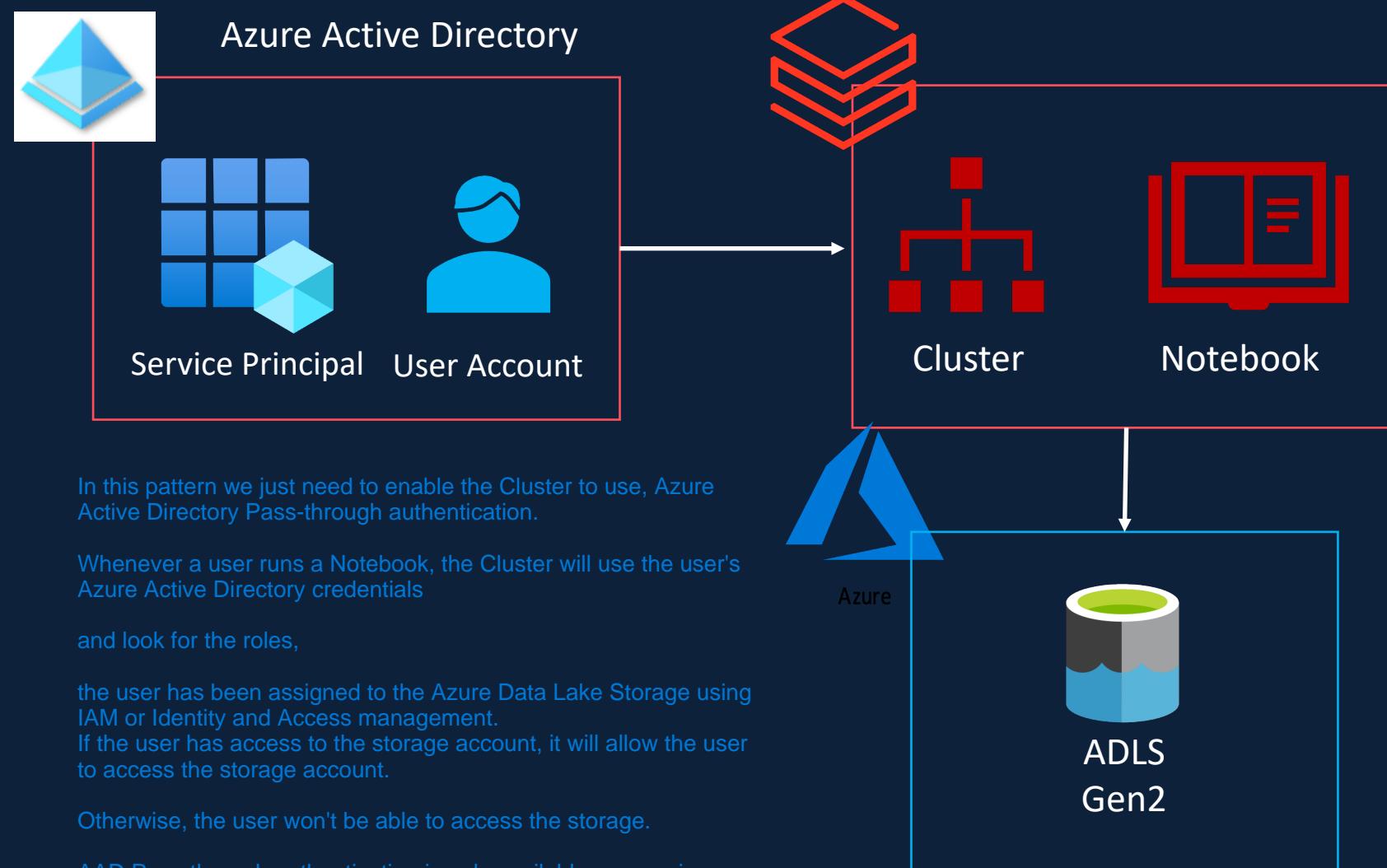
## Cluster Scoped Authentication

Storage Access Keys

Shared Access Signature (SAS Token)

Service Principal

# Access Azure Data Lake - Section Overview

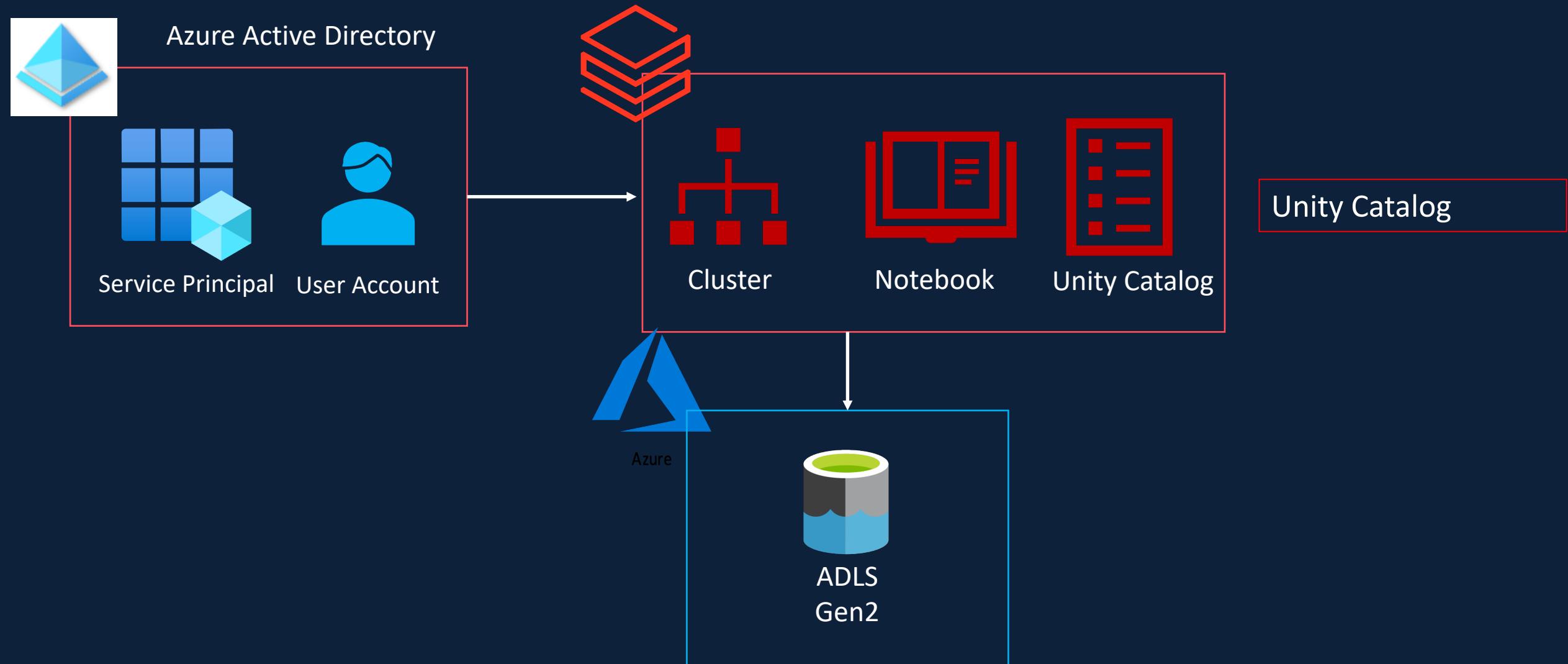


## AAD Passthrough Authentication

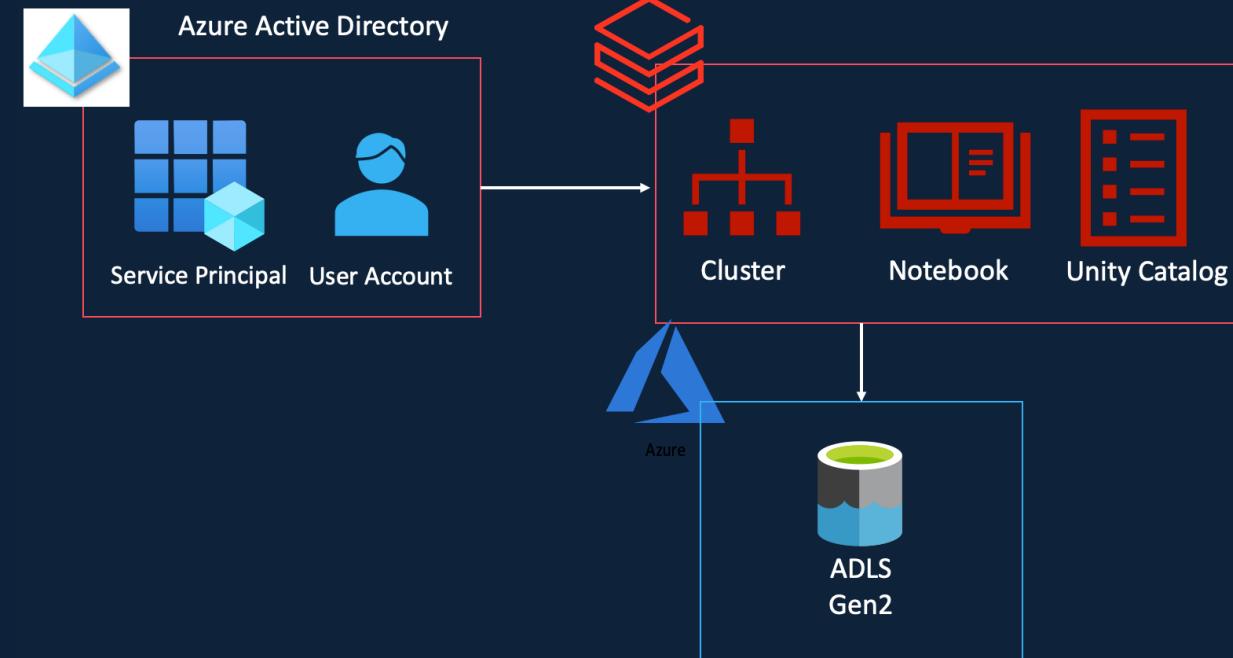
Apart from these, there are two more forms of authentication patterns available in Databricks.

First one is called the AAD Pass-through authentication, or otherwise referred to as the Azure Active Directory Pass-through authentication.

# Access Azure Data Lake - Section Overview



# Access Azure Data Lake - Section Overview



Create Azure Data Lake Gen2 Storage

Access Data Lake using Access Keys

Access Data Lake using SAS Token

Access Data Lake using Service Principal

Using Cluster Scoped Authentication

Access Data Lake using AAD Credential Pass-through

Recommended approach for the course

# Create Azure Data Lake Gen2 Storage (ADLS Gen2)



# Access Azure Data Lake Gen2 using Access Keys



# Authenticate Using Access Keys



ADLS Gen2

Each storage account comes with 2 keys

Gives full Access to the storage account

Keys can be rotated (regenerated)

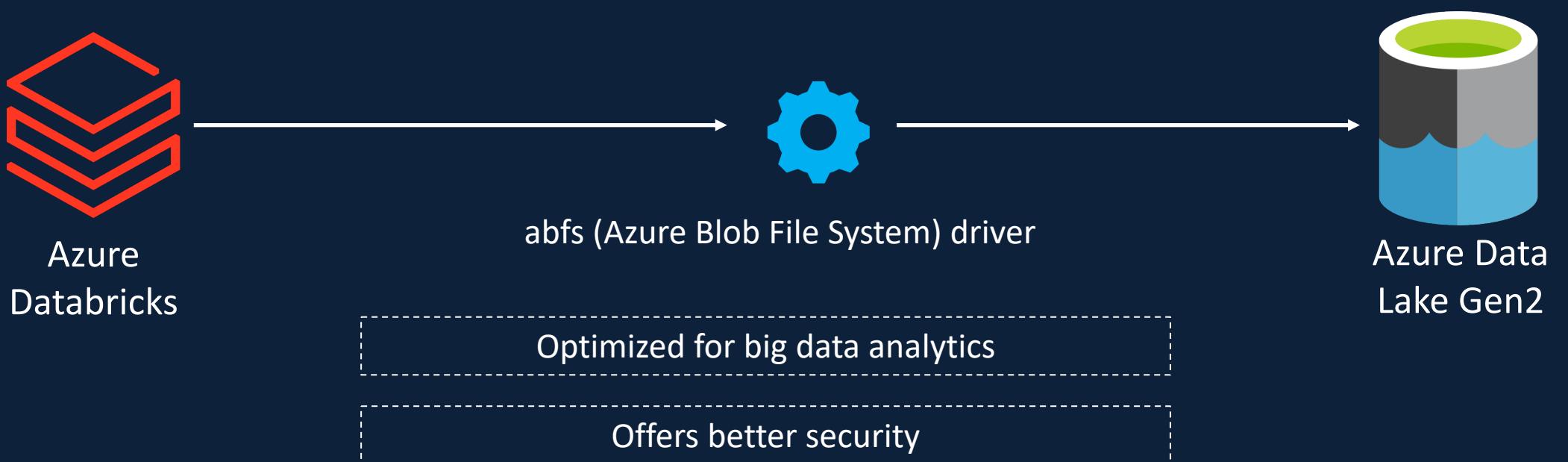
# Access Keys – Spark Configuration



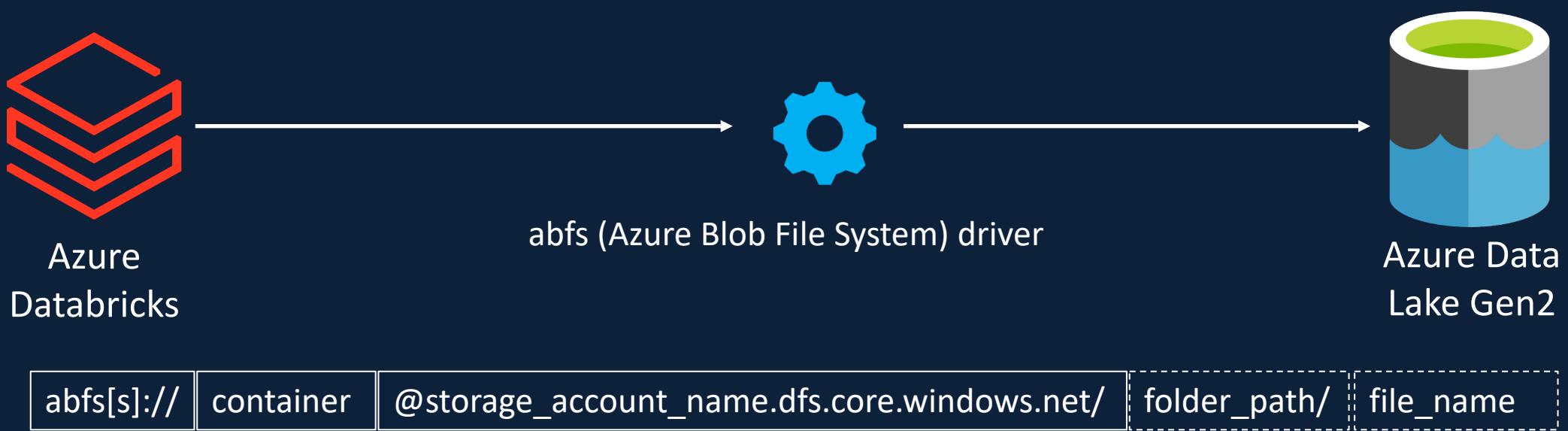
```
spark.conf.set("fs.azure.account.key.<storage-account>.dfs.core.windows.net", "<access key>")
```

```
spark.conf.set(  
    "fs.azure.account.key.formula1dl.dfs.core.windows.net",  
    "30RoyW+laxV39N0JZ7XWRSS0imUGp2lKdE65nRbHrJ9UHc1fqLyJN+j+Qunhev+YL8+CwPLenWn+ASTg8bfJg=")
```

# Access Keys – abfs driver



# Access Keys – abfs driver

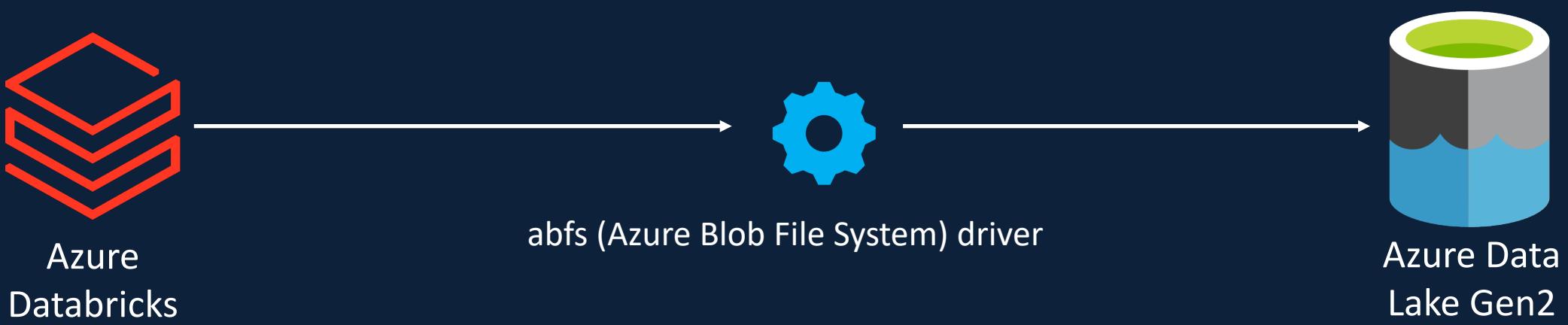


abfss://demo@formula1dl.dfs.core.windows.net/

abfss://demo@formula1dl.dfs.core.windows.net/test/

abfss://demo@formula1dl.dfs.core.windows.net/test/circuits.csv

# Authenticate Using Access Keys



```
spark.conf.set("fs.azure.account.key.<storage-account>.dfs.core.windows.net", "<access key>")
```

```
dbutils.fs.ls("abfss://demo@formula1dl.dfs.core.windows.net/")
```

# Access Azure Data Lake using Shared Access Signature (SAS Token)



# Shared Access Signature



ADLS Gen2

Provides fine grained access to the storage

Restrict access to specific resource types/ services

Allow specific permissions

Restrict access to specific time period

Limit access to specific IP addresses

Recommended access pattern for external clients

# Shared Access Signature



```
spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net",  
"SAS")
```

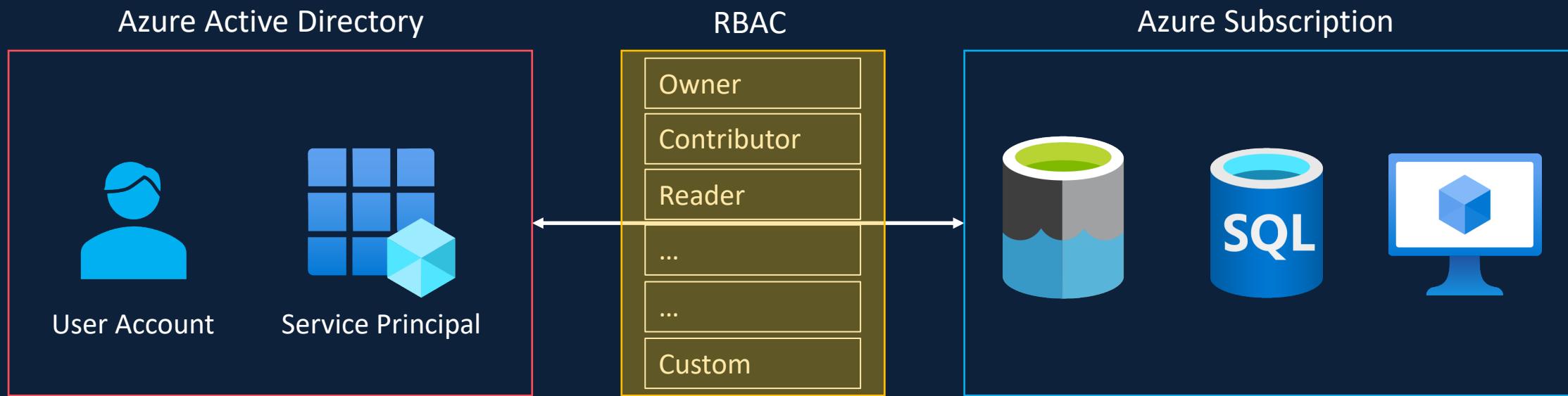
```
spark.conf.set("fs.azure.sas.token.provider.type.<storage-account>.dfs.core.windows.net",  
"org.apache.hadoop.fs.azurebfs.sas.FixedSASTokenProvider")
```

```
spark.conf.set("fs.azure.sas.fixed.token.<storage-account>.dfs.core.windows.net", "<token>")
```

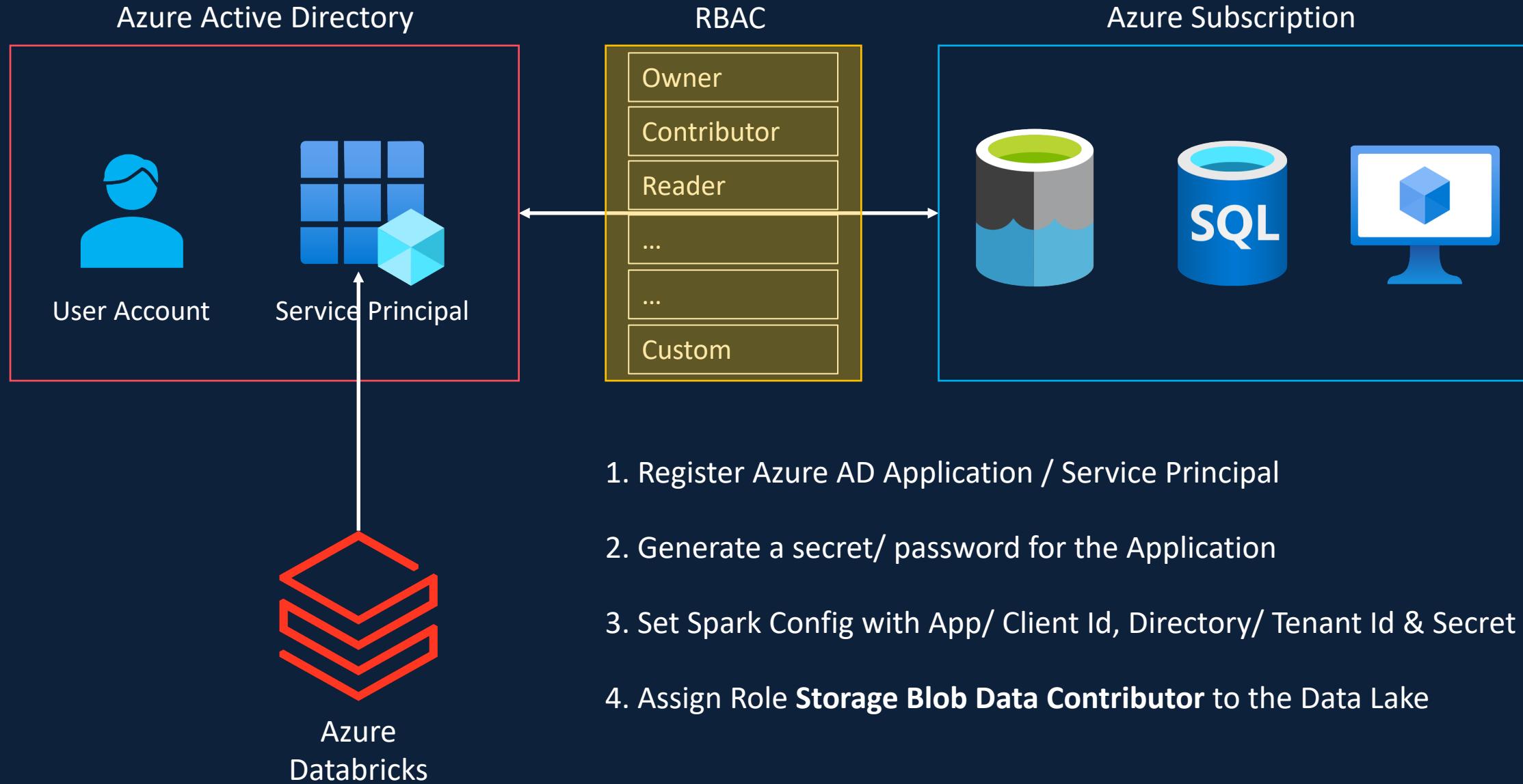
# Access Azure Data Lake using Service Principal



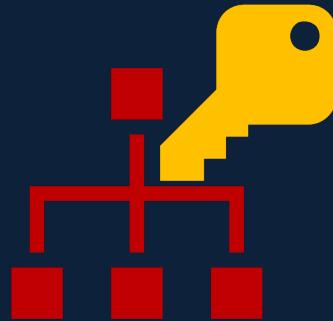
# Service Principal



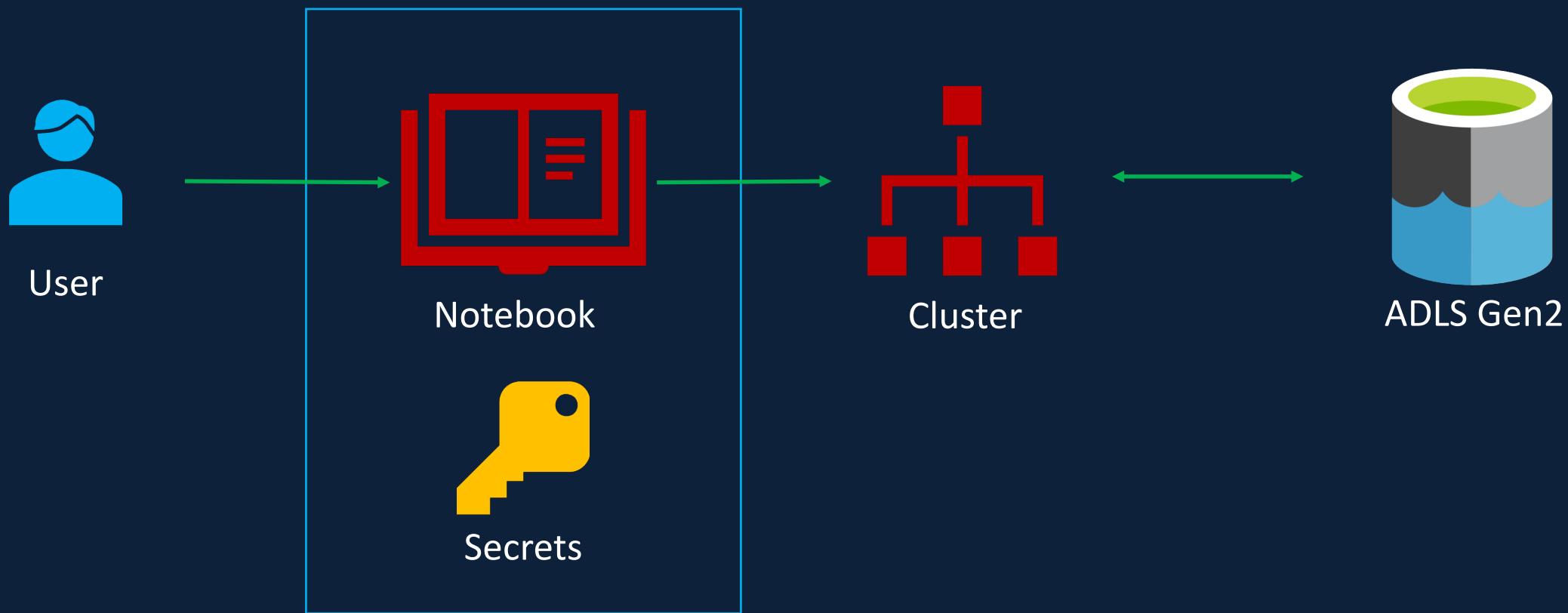
# Service Principal



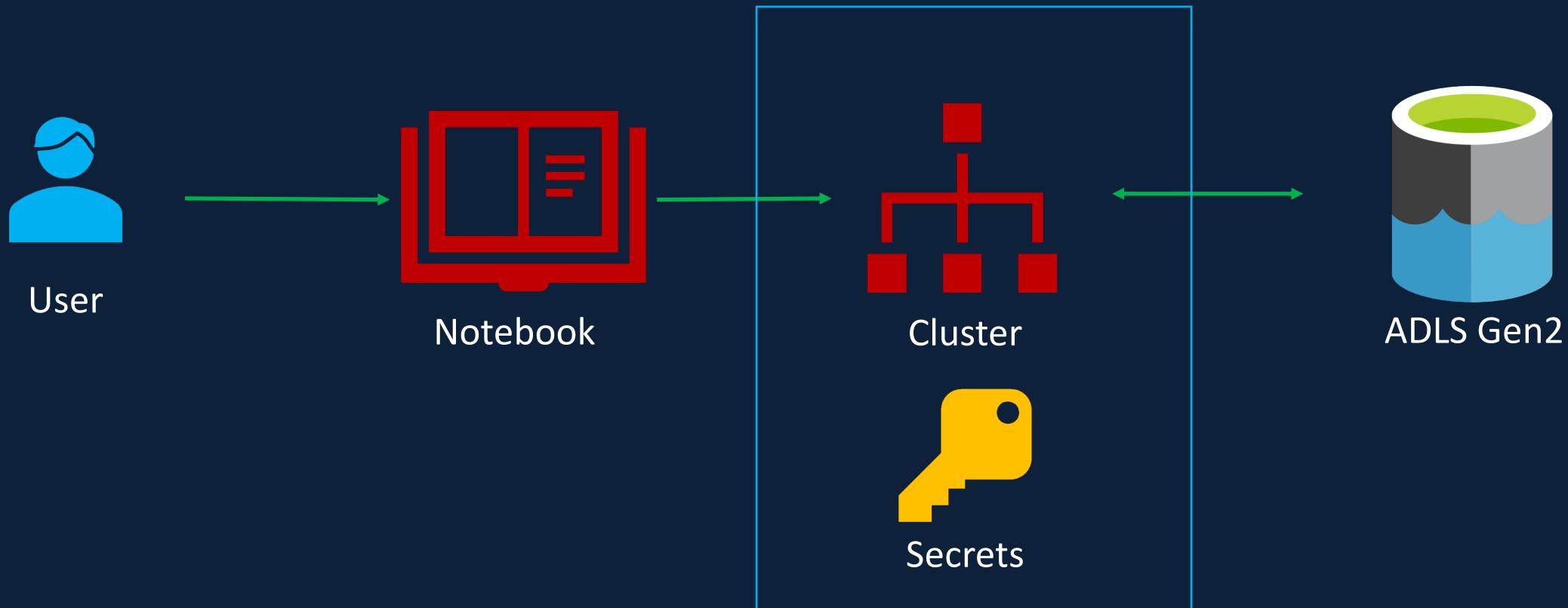
# Cluster Scoped Authentication



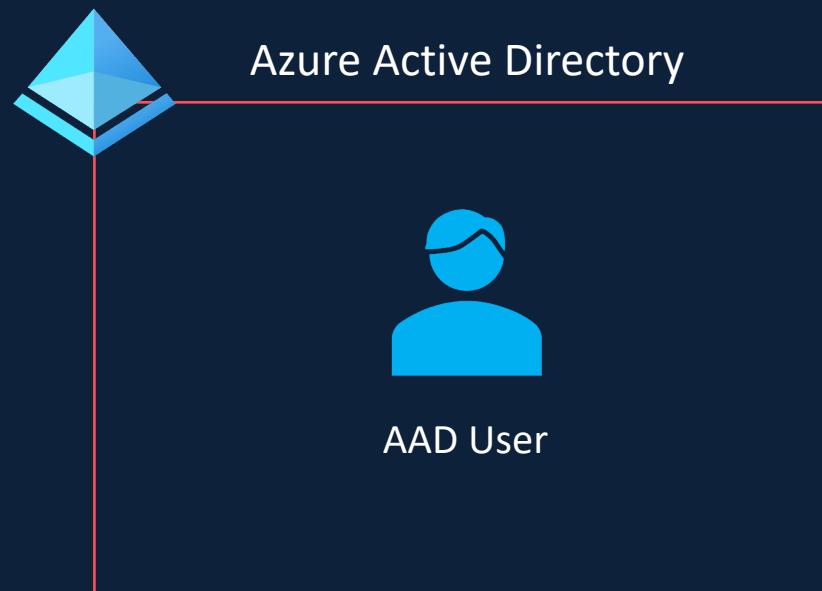
# Session Scoped Authentication



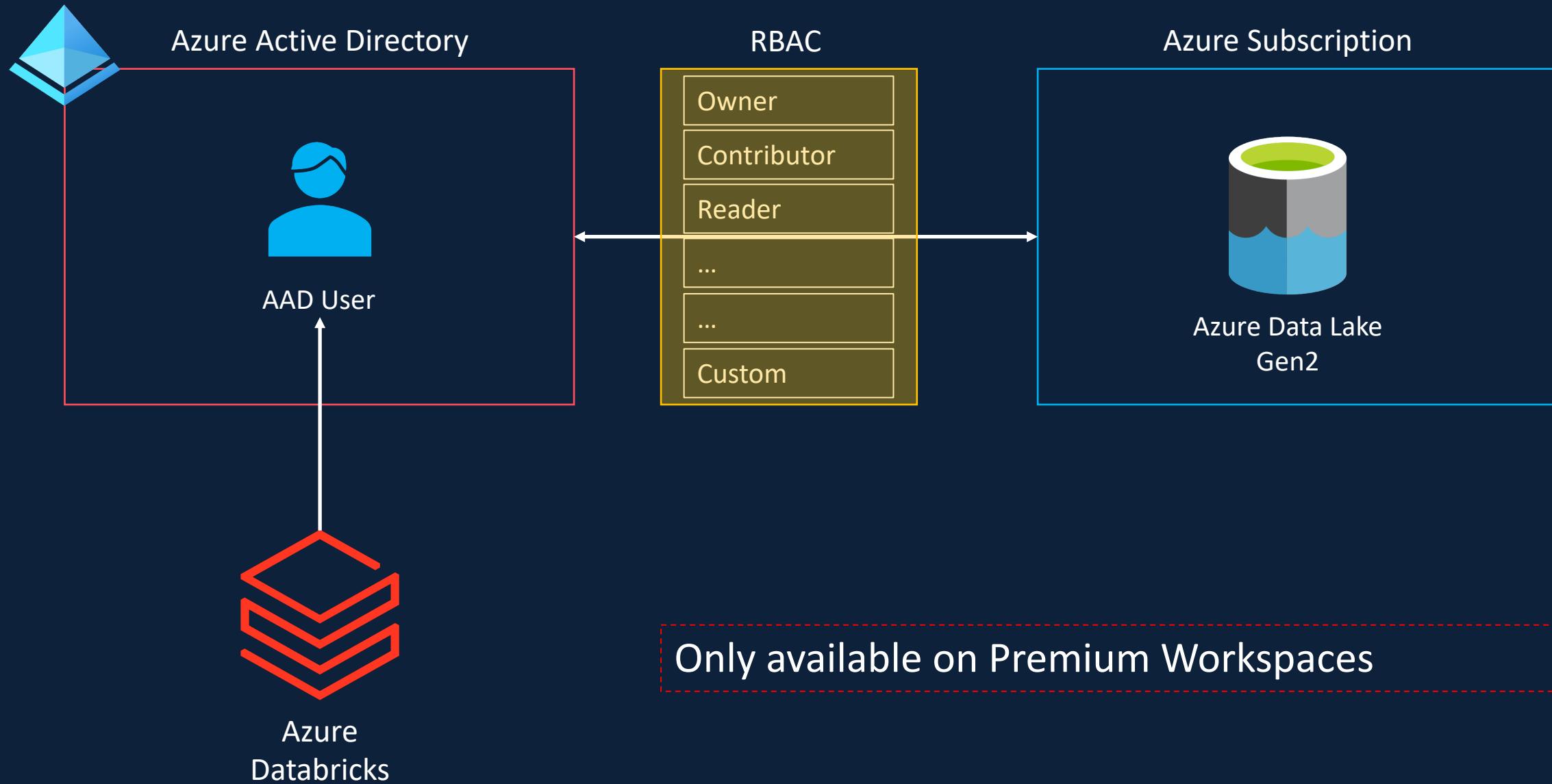
# Cluster Scoped Authentication



# AAD Credential Passthrough



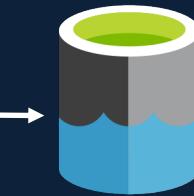
# AAD Credential Passthrough



# Recommended Access Pattern For The Course

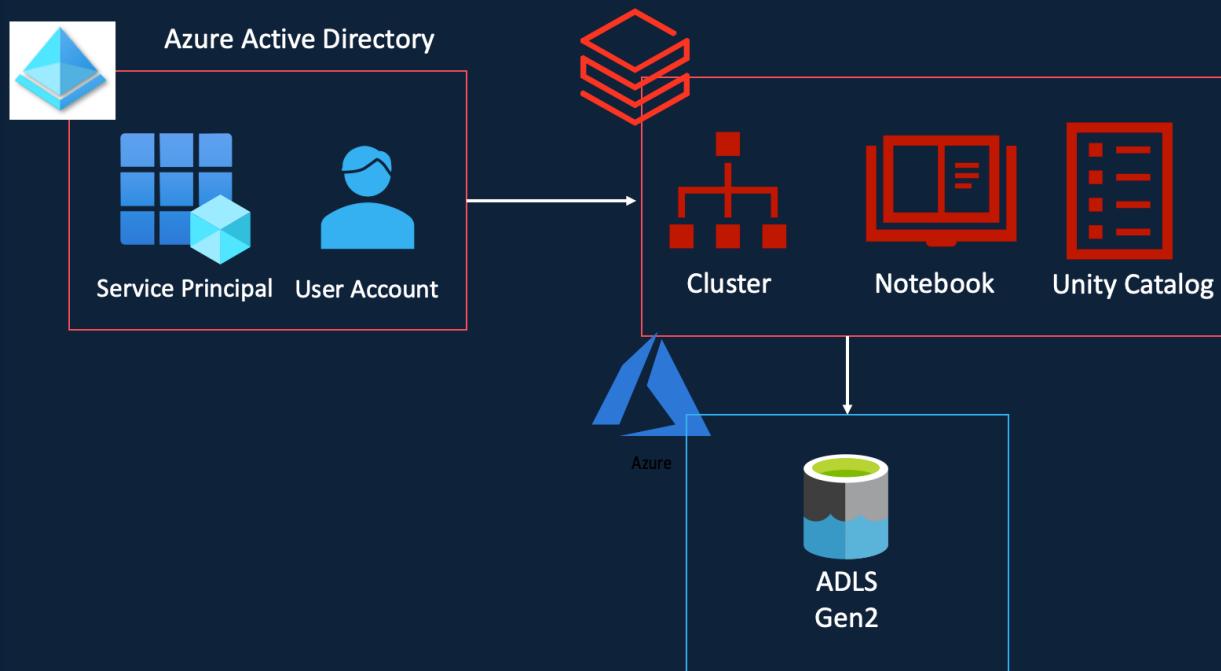


Azure  
Databricks



Azure Data  
Lake Gen2

# Recommended Access Pattern



Access Data Lake using Access Keys

Access Data Lake using SAS Token

Access Data Lake using Service Principal

Access Data Lake using Cluster Scoped Authentication

Access Data Lake using AAD Credential Pass-through

Securing Credentials & Secrets

Databricks Mounts

# Recommended Access Pattern

Student Subscription

Company Subscription  
without Access to AAD

Free Subscription

Pay-as-you-go  
Subscription

Any other Subscription  
with Access to AAD



Using Cluster Scoped Authentication (via  
Access Keys)



Using Service Principal

# Securing Secrets



Databricks Secret Scope

Azure Key Vault

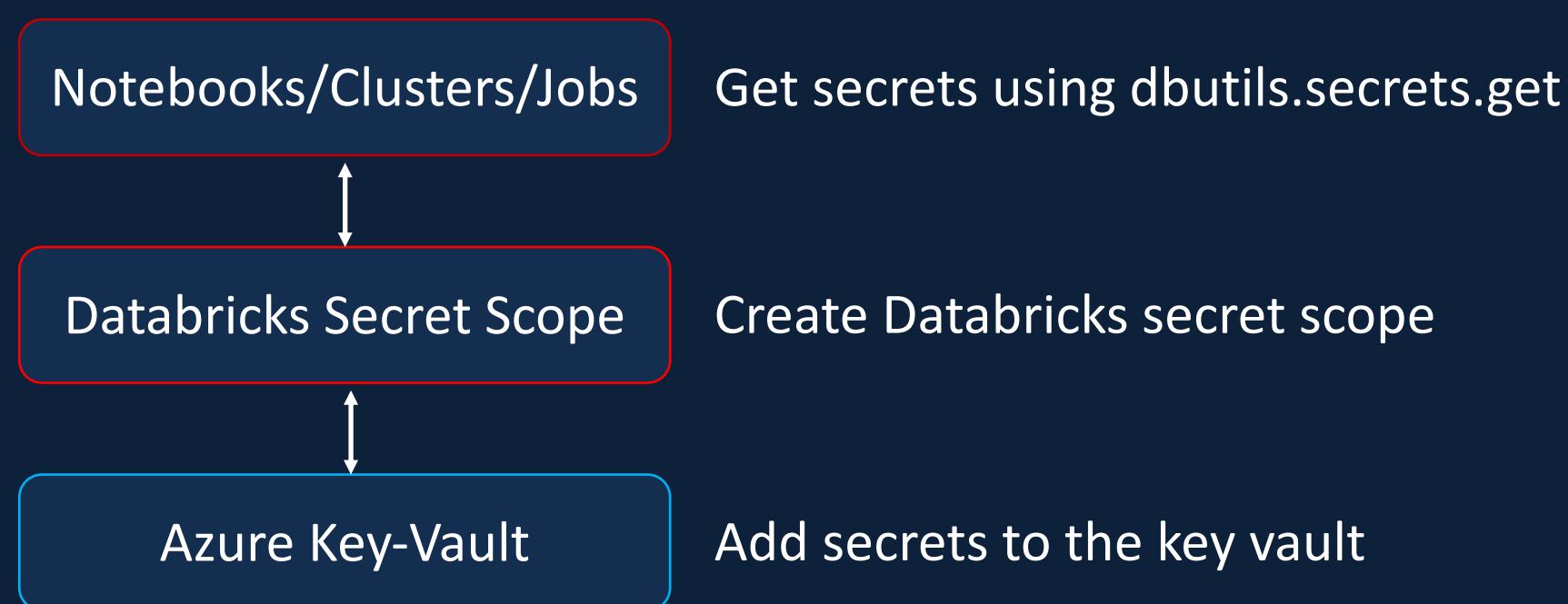
# Securing Secrets – Section Overview

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

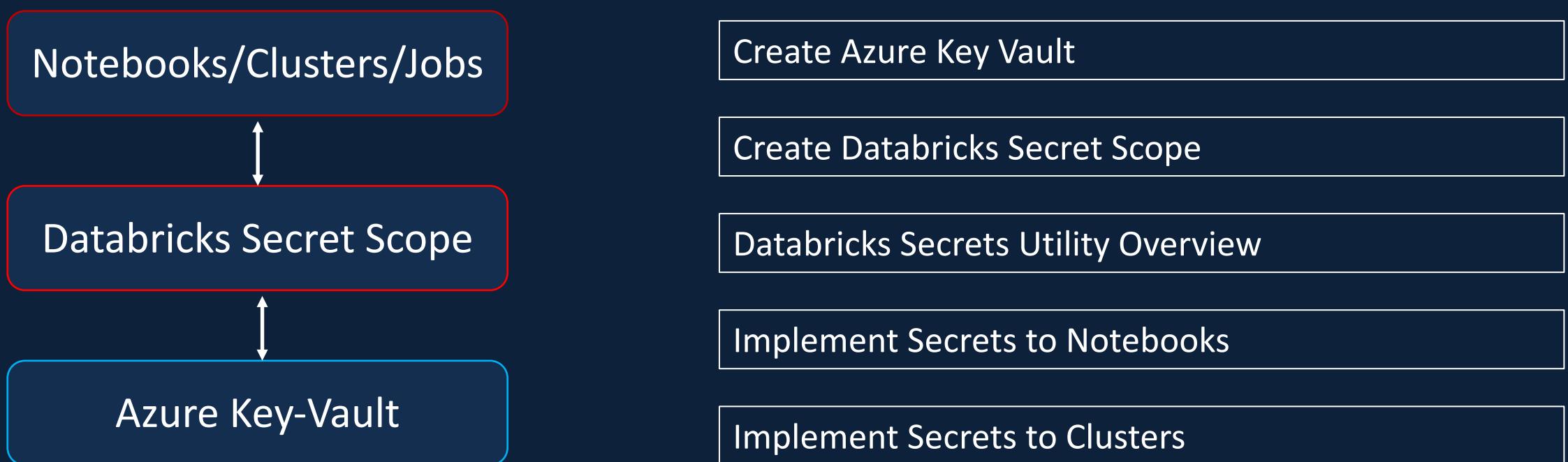
Databricks backed Secret Scope

Azure Key-vault backed Secret Scope

# Securing Secrets – Section Overview



# Securing Secrets – Section Overview



# Creating Azure Key Vault



# Creating Secret Scope

# Databricks Secrets Utility

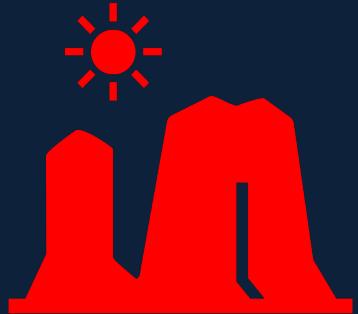
## (dbutils.secrets)

# Implement Secrets Utility in Databricks Notebooks

# Implement Secrets Utility in Databricks Notebooks (Assignment)

# Implement Secrets Utility in Databricks Clusters

# Databricks Mounts

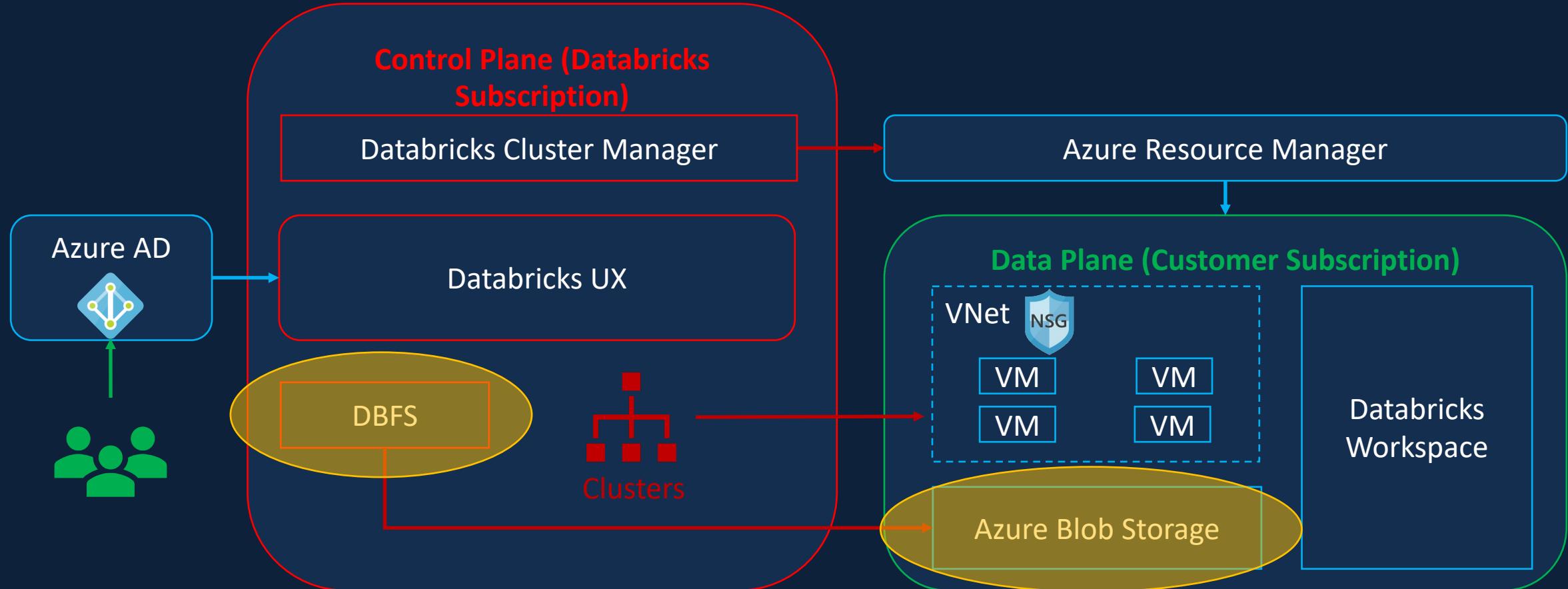


What is Databricks File System (DBFS)

What are Databricks mounts

Mount ADLS container to Databricks

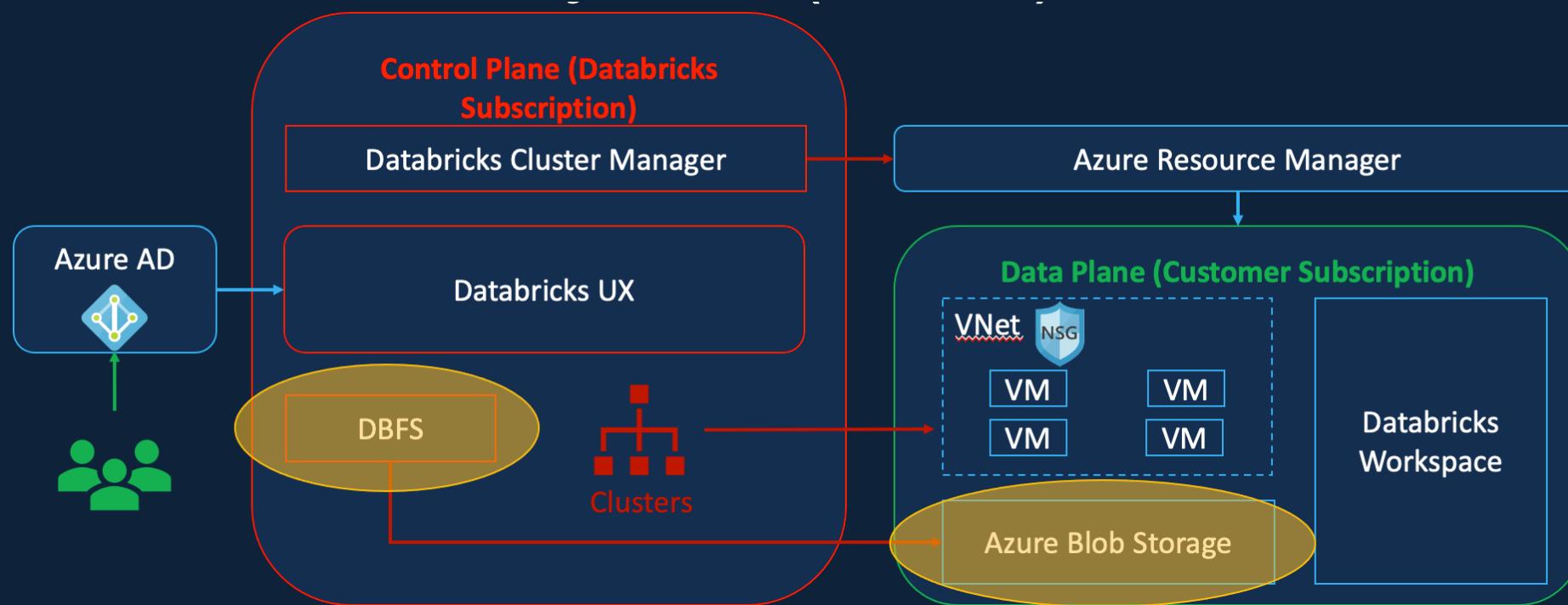
# Databricks File System (DBFS)



Databricks File System (DBFS) is a distributed file system mounted on the Databricks workspace

DBFS Root is the default storage for a Databricks workspace created during workspace deployment

# DBFS Root



Backed by the default Azure Blob Storage

Can be accessed via the web UI

Query results are stored here

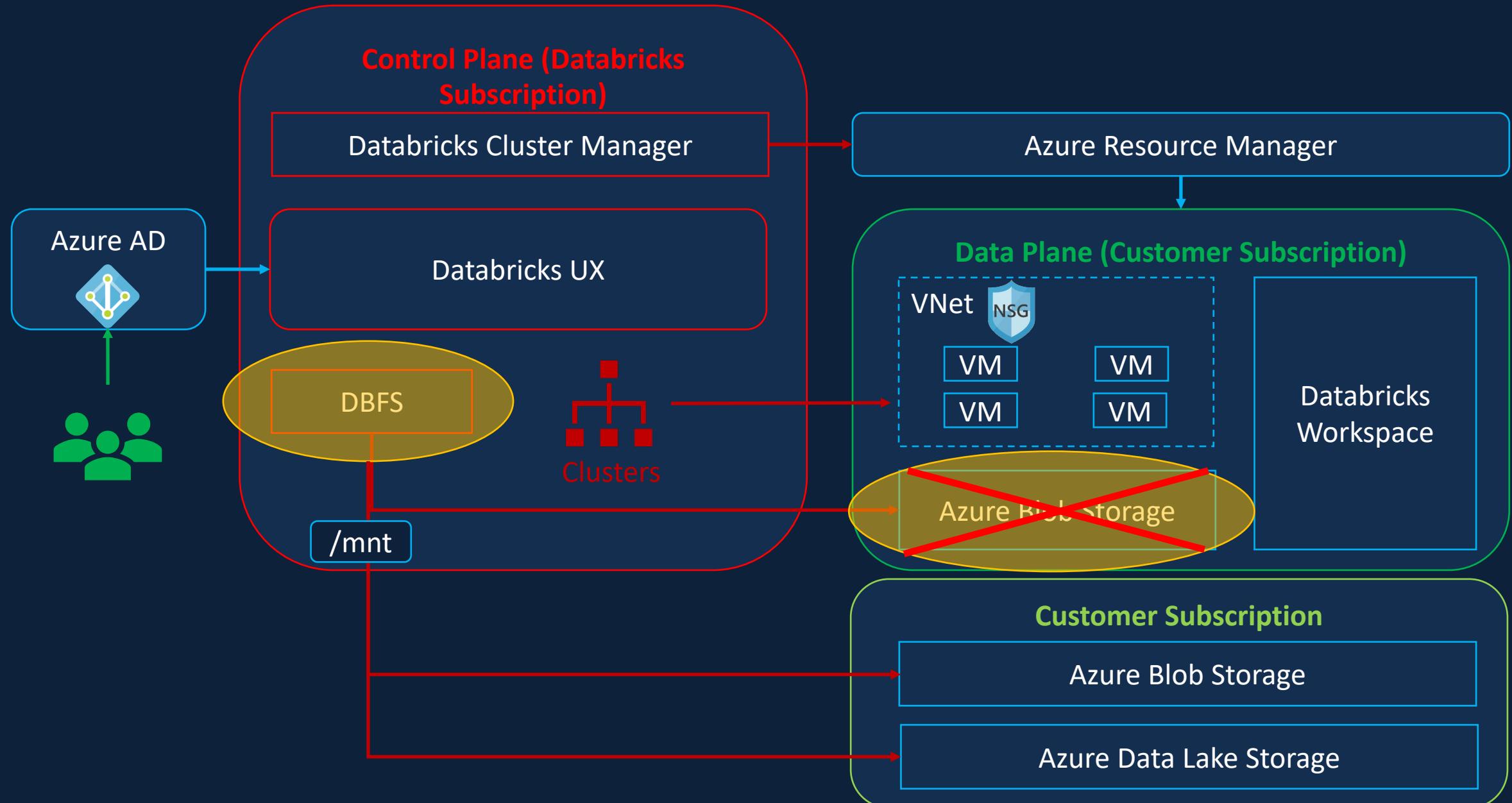
Default storage location for managed tables

Default storage location, but not recommended for user data

# DBFS Root Demo



# Databricks Mounts



# Databricks Mounts

## Benefits

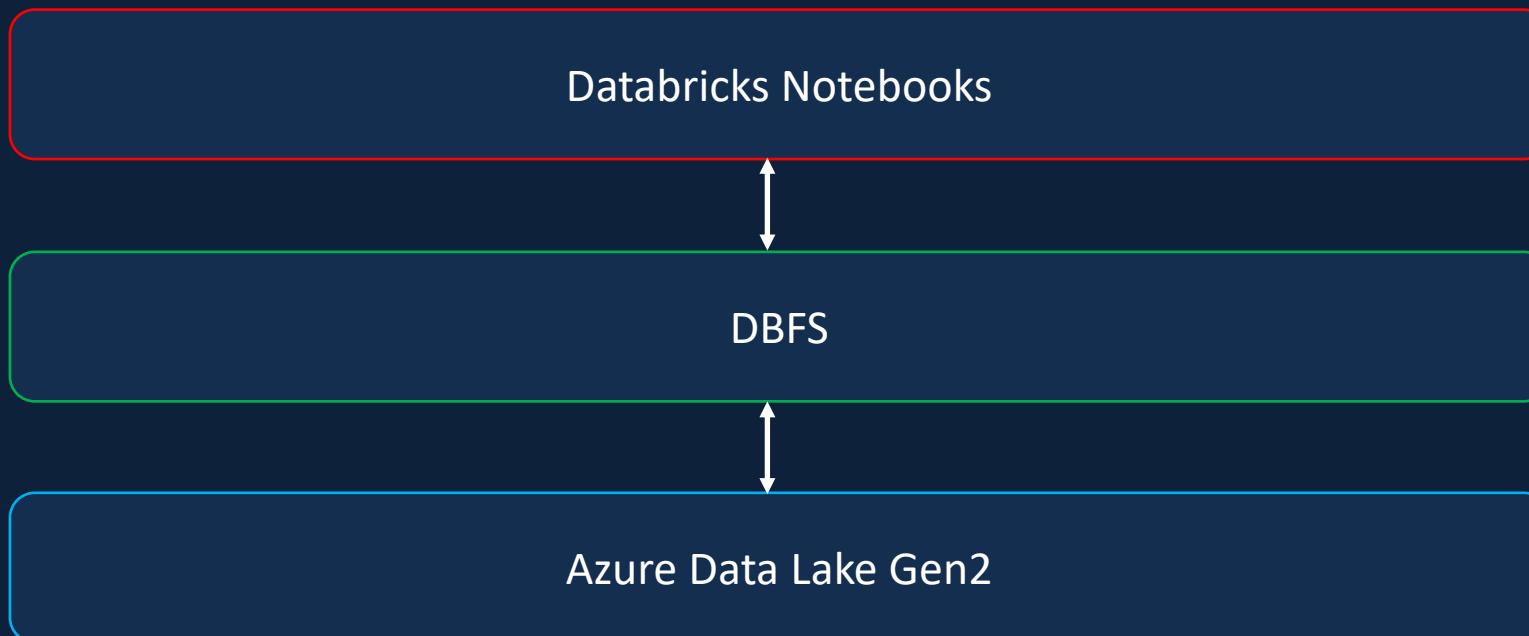
Access data without requiring credentials

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

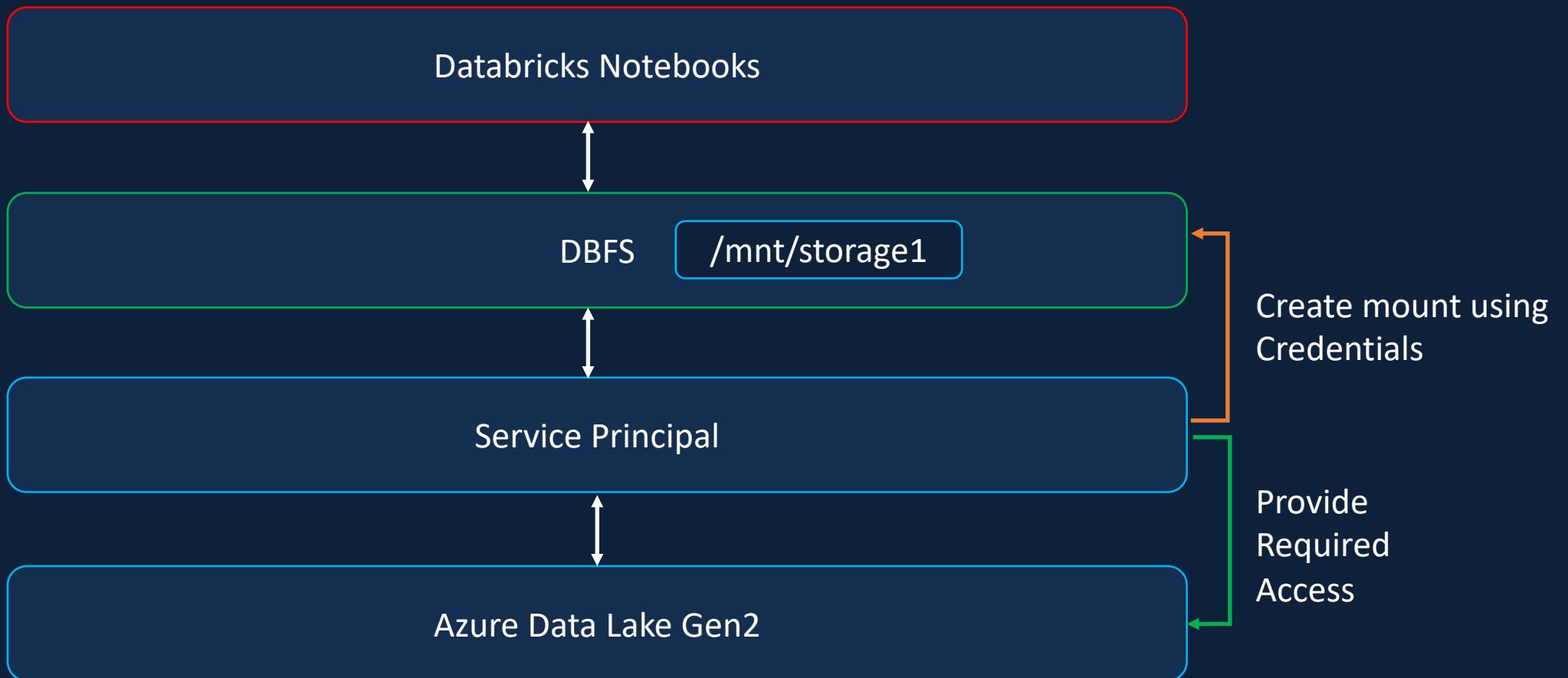
Stores files to object storage (e.g. Azure Blob), so you get all the benefits from Azure

Recommended solution for access Azure Storage until the introduction  
of Unity Catalog (Generally Available from end of 2022)

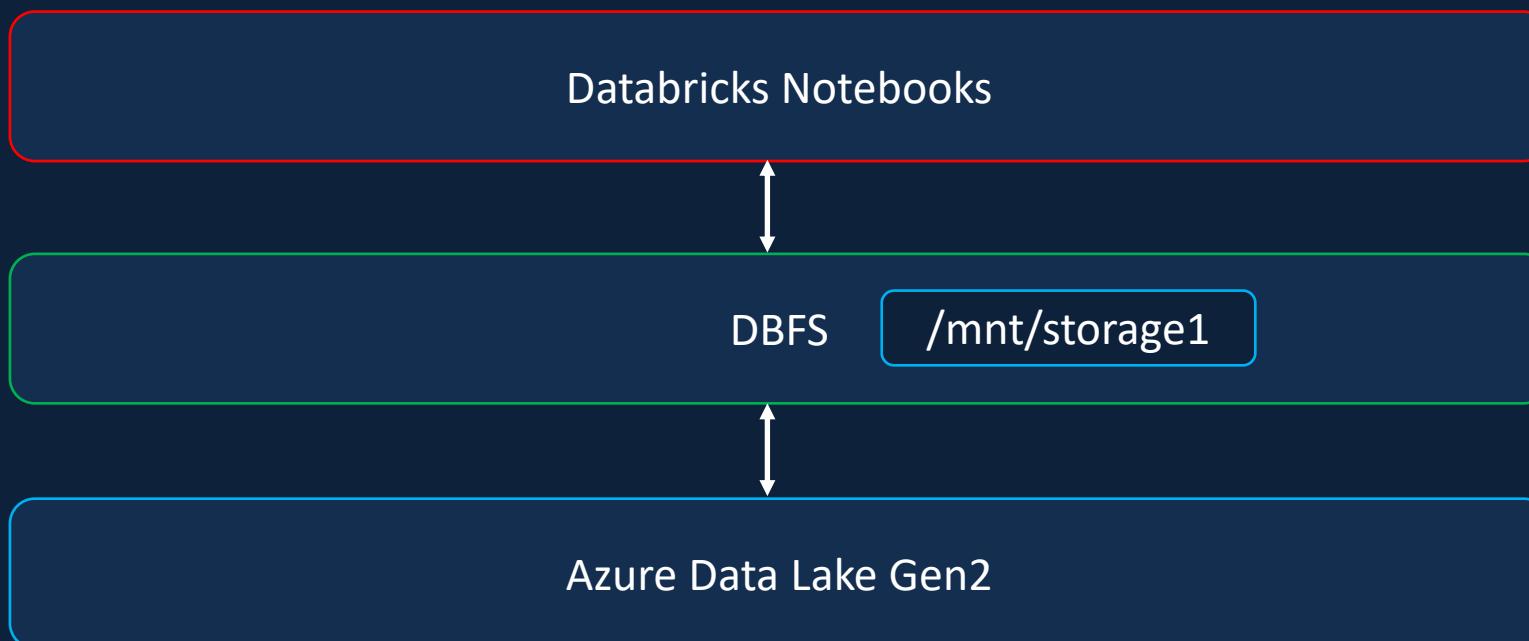
# Databricks Mounts



# Databricks Mounts



# Databricks Mounts



# Mounting Azure Data Lake Storage Gen2



# Project Overview



I'll then walk you through the Project Requirements.

This will be functional requirements such as transformations required, dashboards and reports to be created as well as non-functional requirements such as scheduling, monitoring and alerting.

What is formula1

Formula1 data source & datasets

Prepare the data for the project

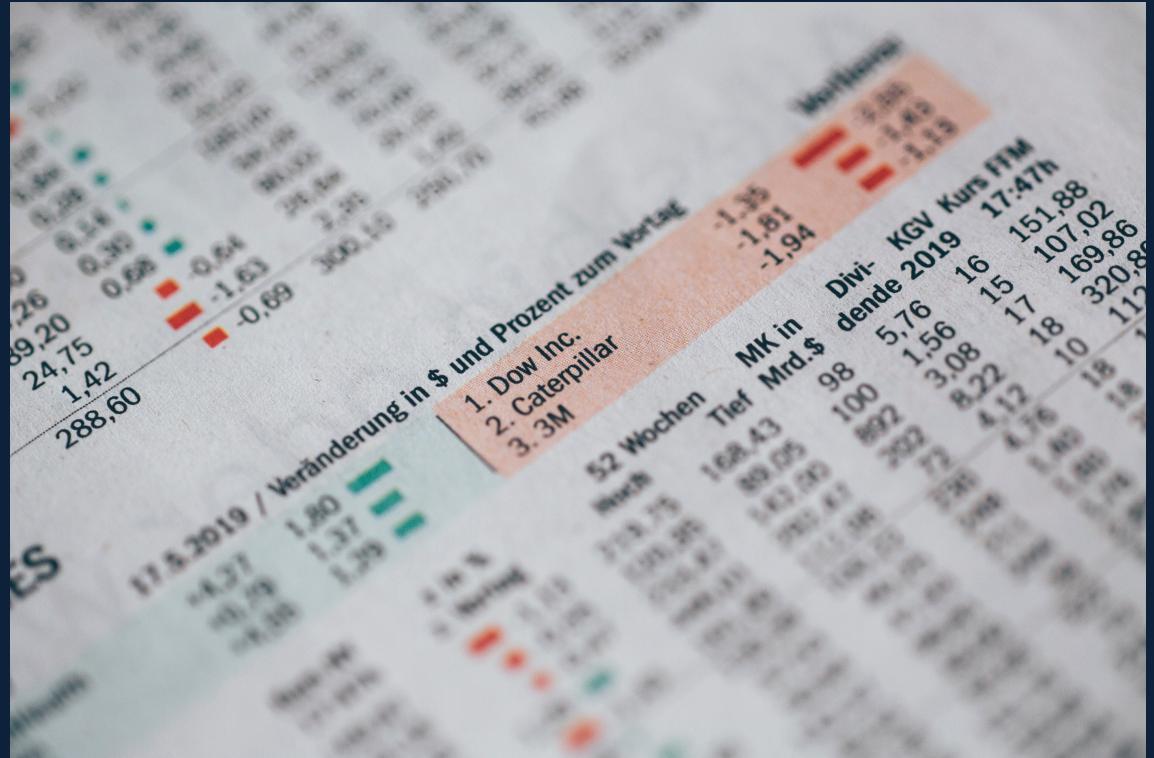
Project Requirements

Solution Architecture

I'll then walk you through the Solution Architecture, Stock Design that we'll be using to implement our project.

I'll also provide references to the Industry Standard Architectures available in Microsoft as well as Databricks blogs and websites.

By the end of this section, I hope you will have a good understanding about the project we'll be developing as part of this course, as well as the Solution Architecture we are going to be using for the solution.





# Formula1

# Formula1 Overview

Similar to an IPL cricket English Premier League football or NFL  
Formula One season happens once a year, roughly over 20 races.

Usually there is only one race in a circuit each year, but due to the COVID outbreak in 2020, there were more than one race conducted in some of the circuits.

## Race Circuits

There are roughly ten teams that participate in each season and they are also called constructors in Formula One.

## Teams/ Constructors

Each team has two drivers and each driver is assigned a specific car.

## Drivers

There are also reserve drivers in each team, but only to participate in races and they are the only two we are going to talk about here.

## Seasons

Raceway can spans from Friday to Sunday.

## Race Weekend

Each race happens over a weekend.

There are two practice sessions on a Friday and a final practice session on Saturday morning.

They don't count towards any points or achievements, so we won't focus on these.

## Practice

There is a qualifying session on Saturday afternoon and it happens over three different stages.

## Qualifying

The qualifying results decide the grid position of the driver as to where he'll start the race. The higher the driver qualifies the forward, he gets to start the race, which is a massive advantage.

Unlike qualifying sessions, which are decided over a single lap, races span multiple laps, roughly between 50 and 70 laps, depending on the length of the circuit.

## Qualifying Results

Also during the race, drivers make pit stops to change the tires or to replace the damaged car.

## Race

Unlike qualifying sessions, which are decided over a single lap, races span multiple laps, roughly between 50 and 70 laps, depending on the length of the circuit.

## Laps

Also during the race, drivers make pit stops to change the tires or to replace the damaged car.

## Pit Stops

## Race Results

Based on the race results. Drivers and constructors standings are decided.

## Constructor Standings

Whichever driver is on the top of the standings at the end of the season is the drivers champion. And similarly, the team that's leading the constructors standings becomes the Constructors champion.

## Driver Standings

# Formula1 Data Source



There is a third party developer API called Ergaste that makes the data available for all races from 1950 onwards.  
So let's visit the website and explore it a bit further.

<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

As you saw, there are 12 files that we can download from the Ergaste API, but out of the 12 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

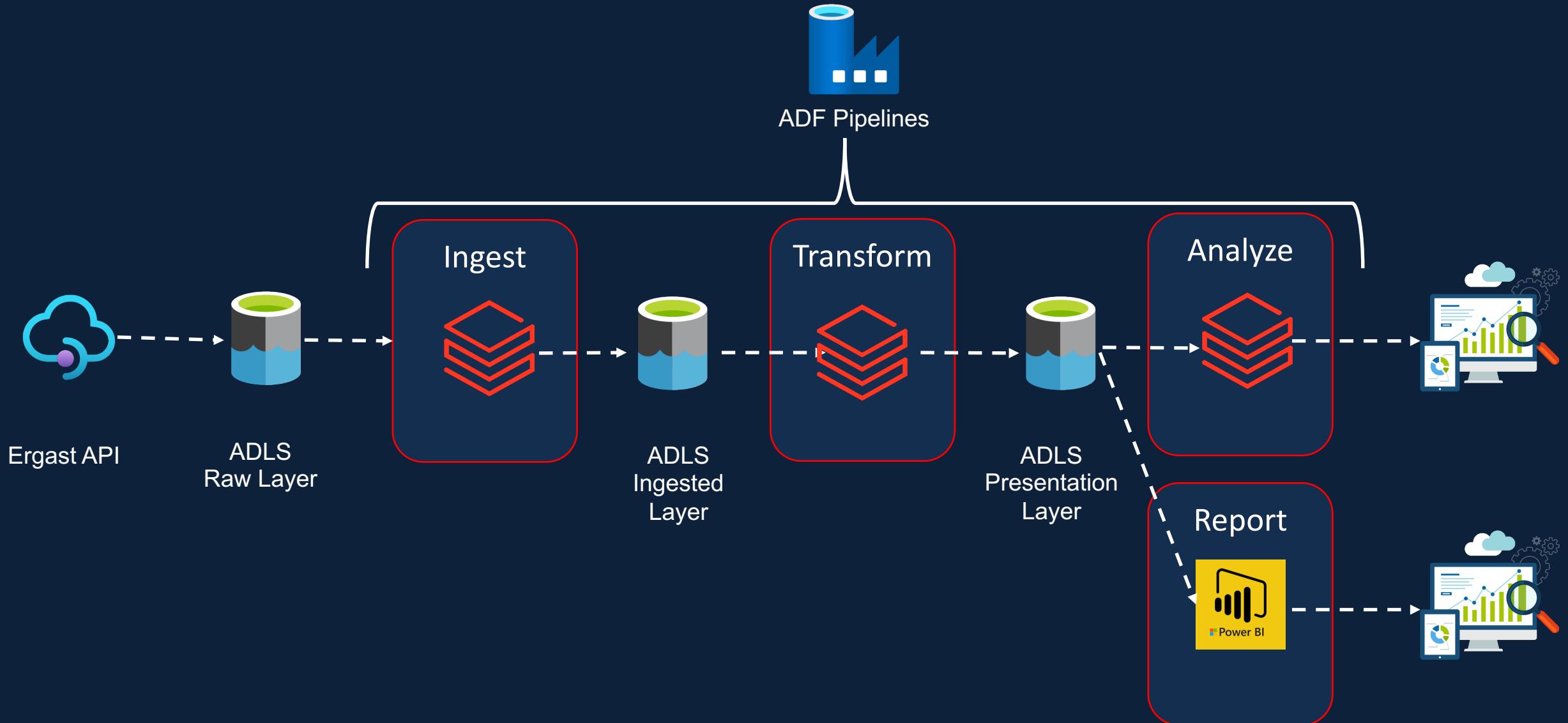
Ability to see history and time travel

Ability to roll back to a previous version

Blank

# Solution Architecture Overview

# Solution Architecture

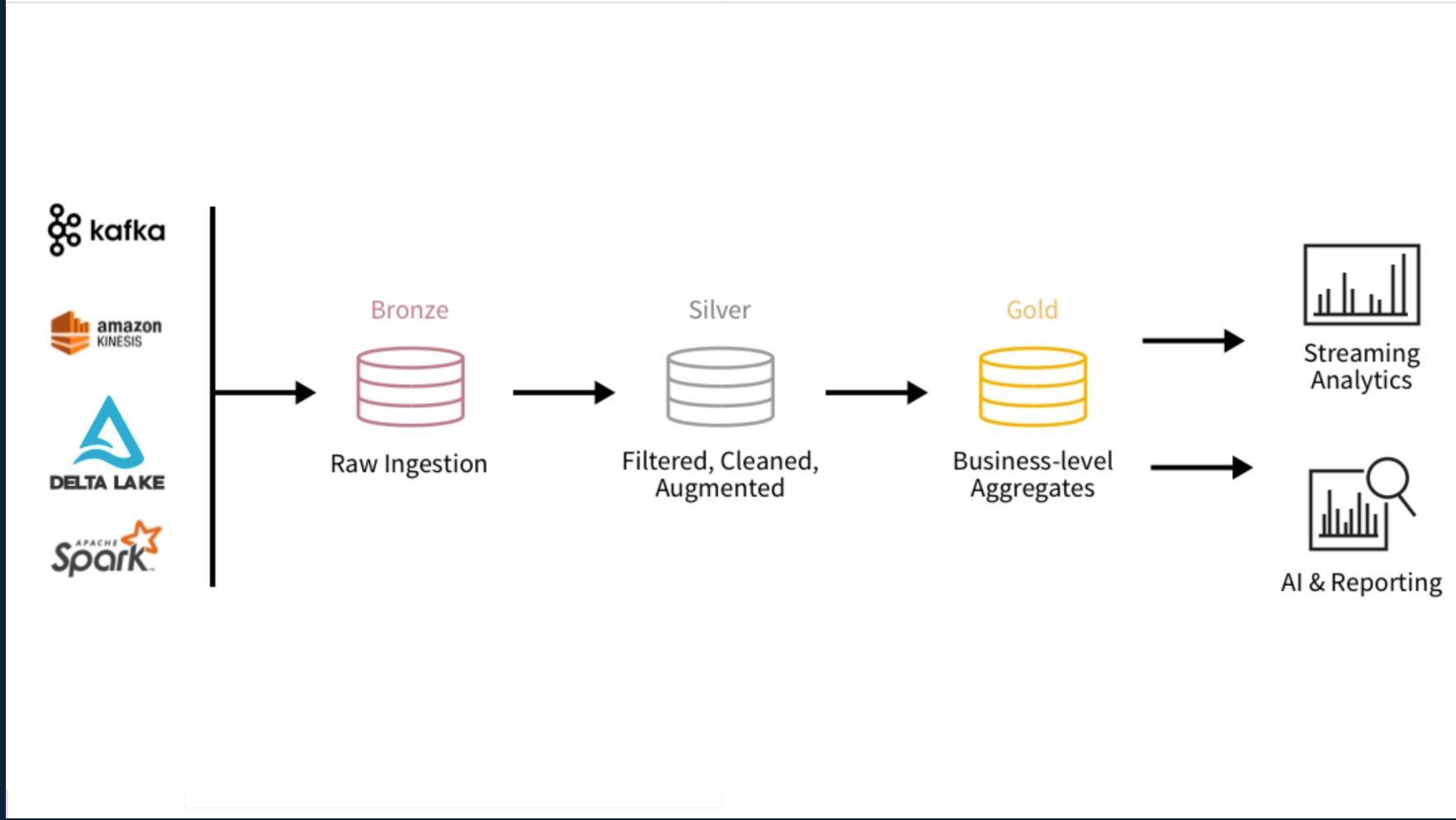


# Azure Databricks Modern Analytics Architecture



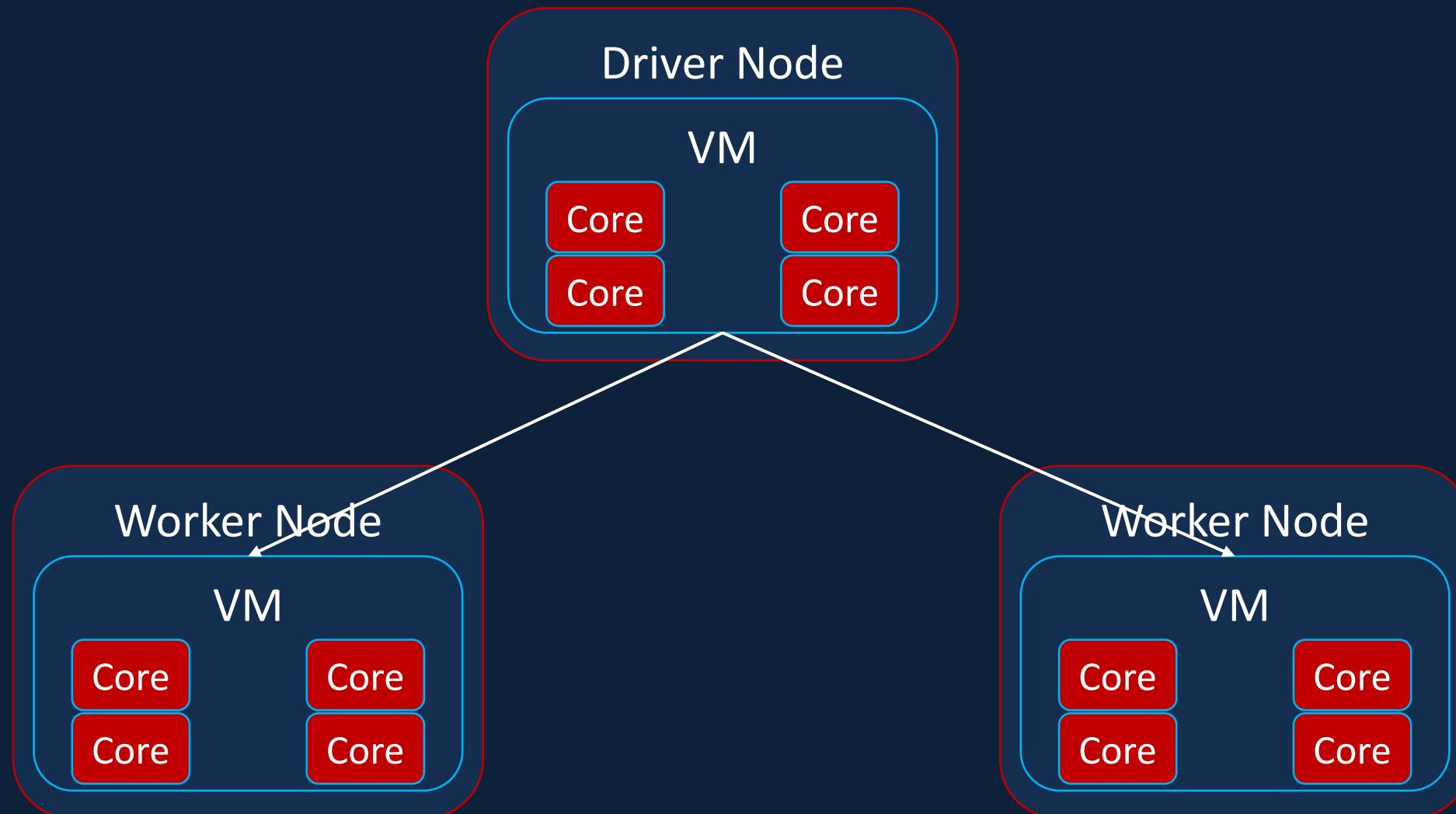
<https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/azure-databricks-modern-analytics-architecture>

# Databricks Architecture

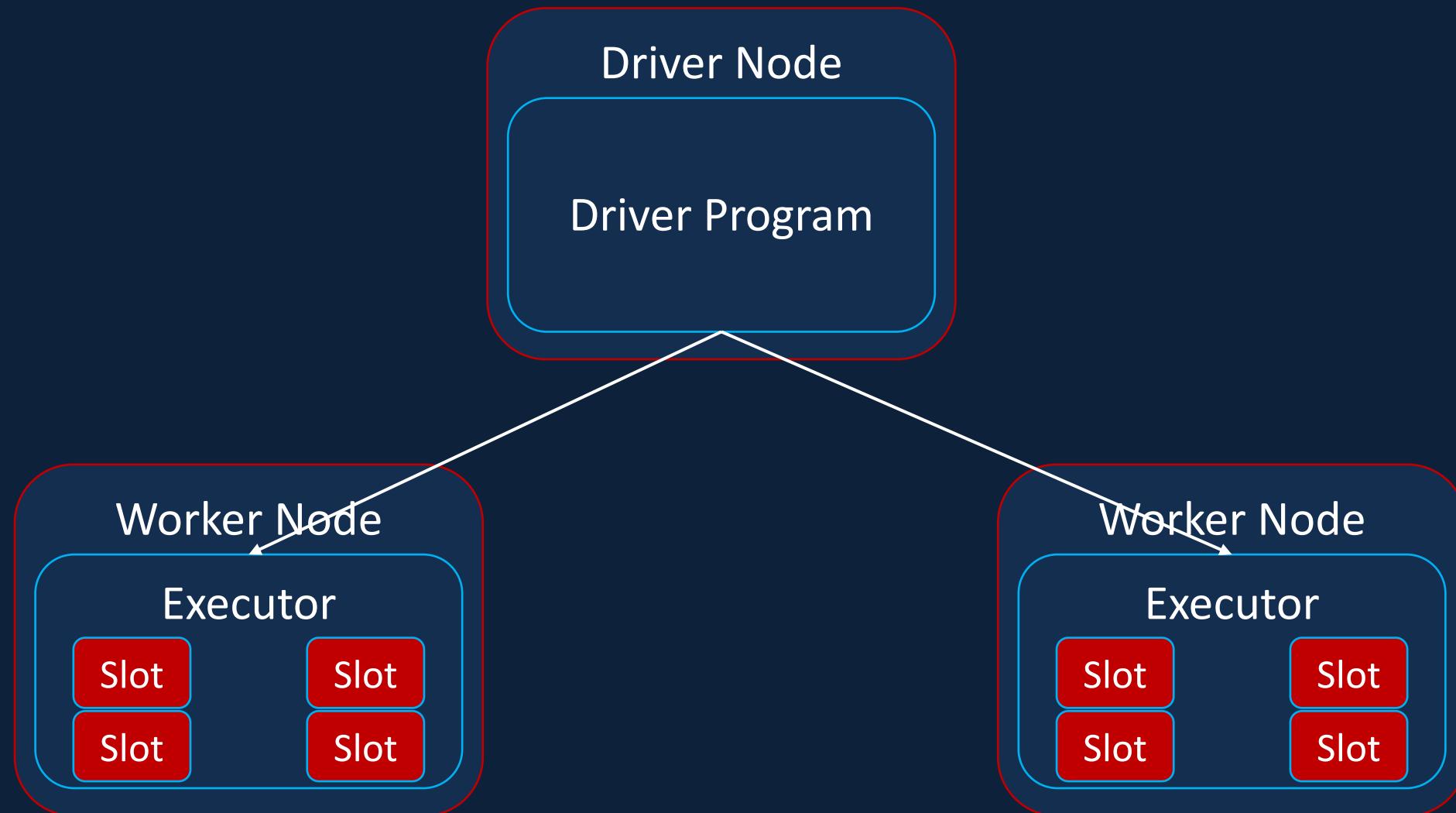


# Spark Architecture

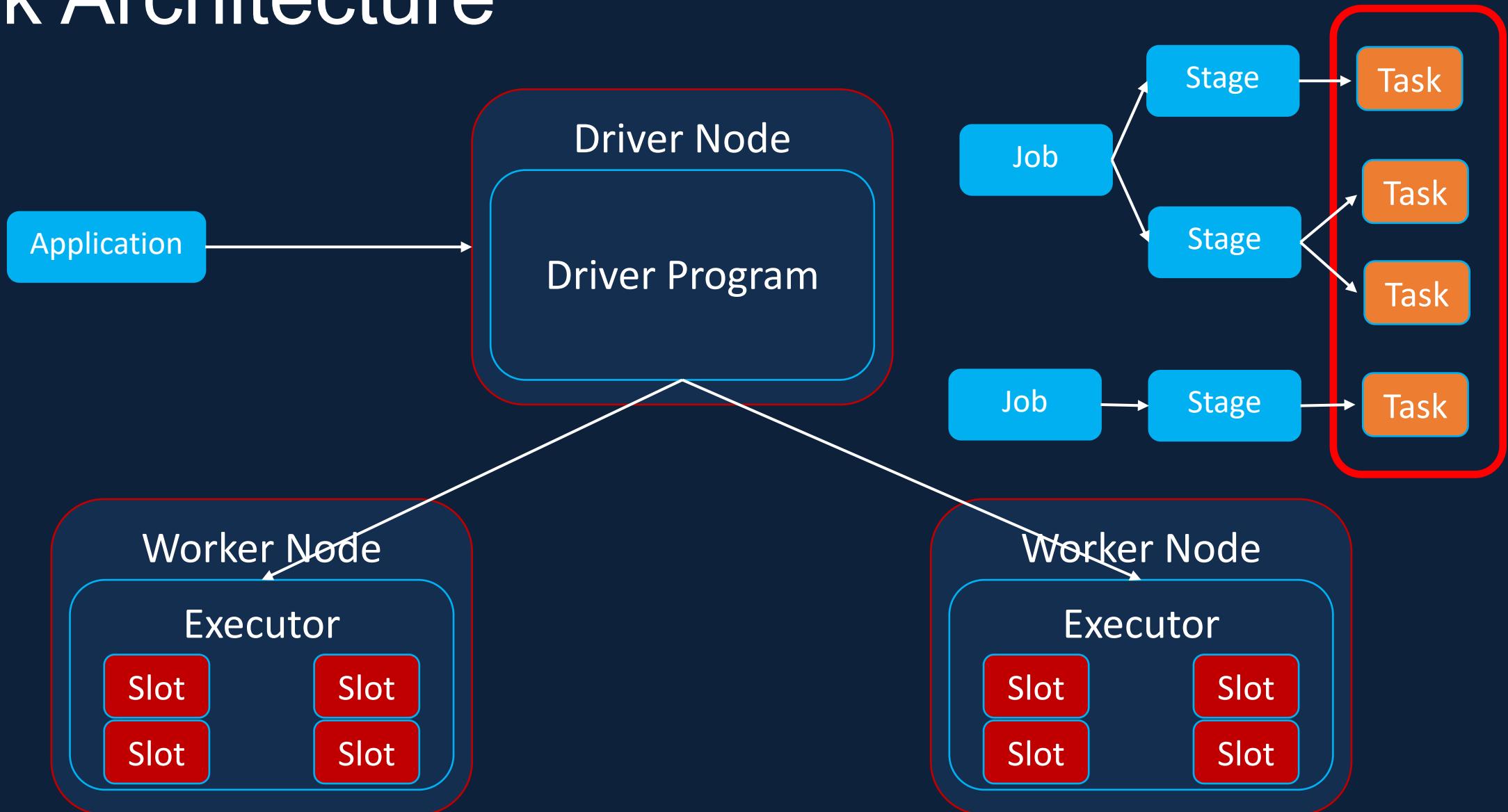
# Spark Architecture



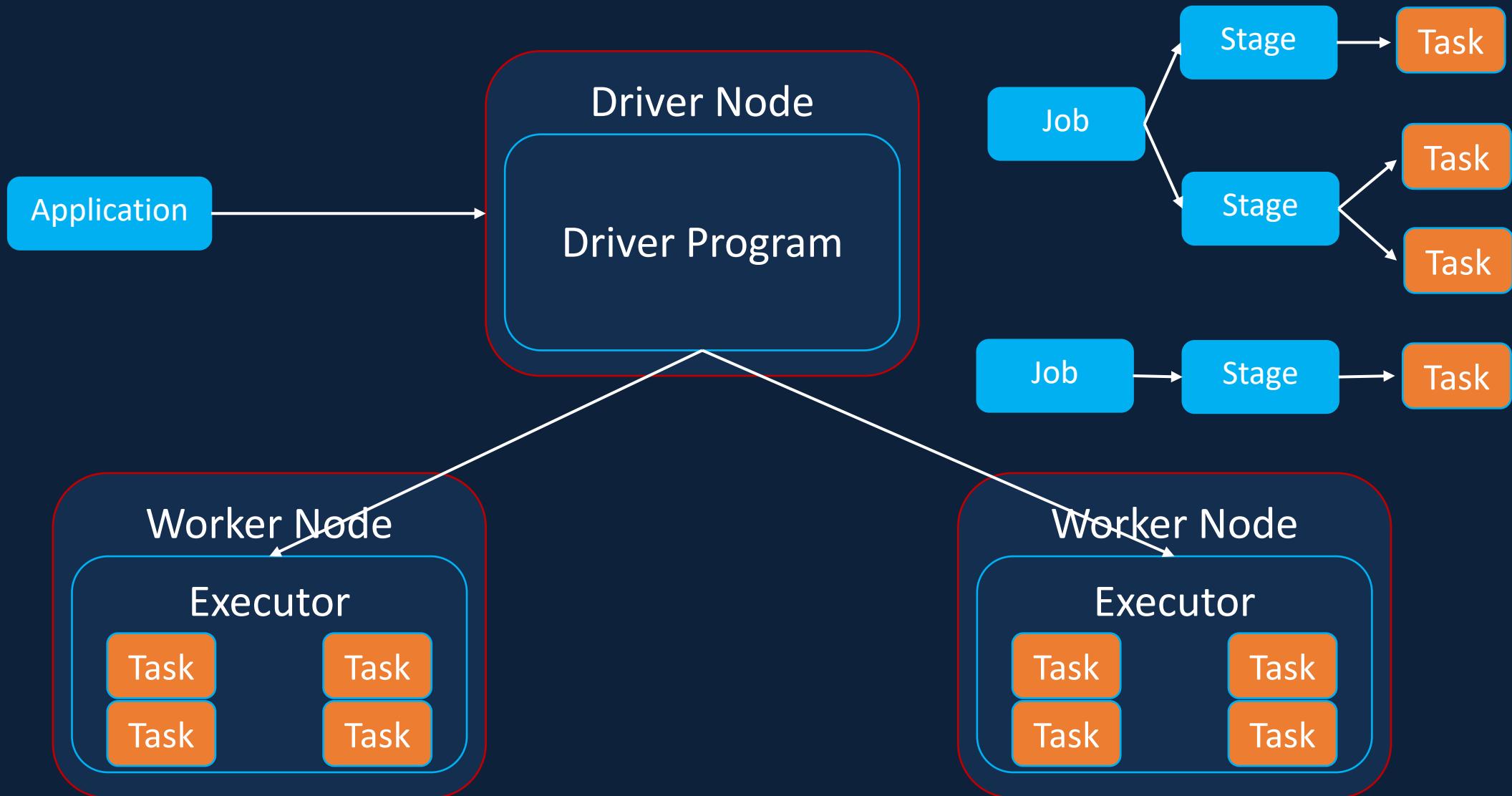
# Spark Architecture



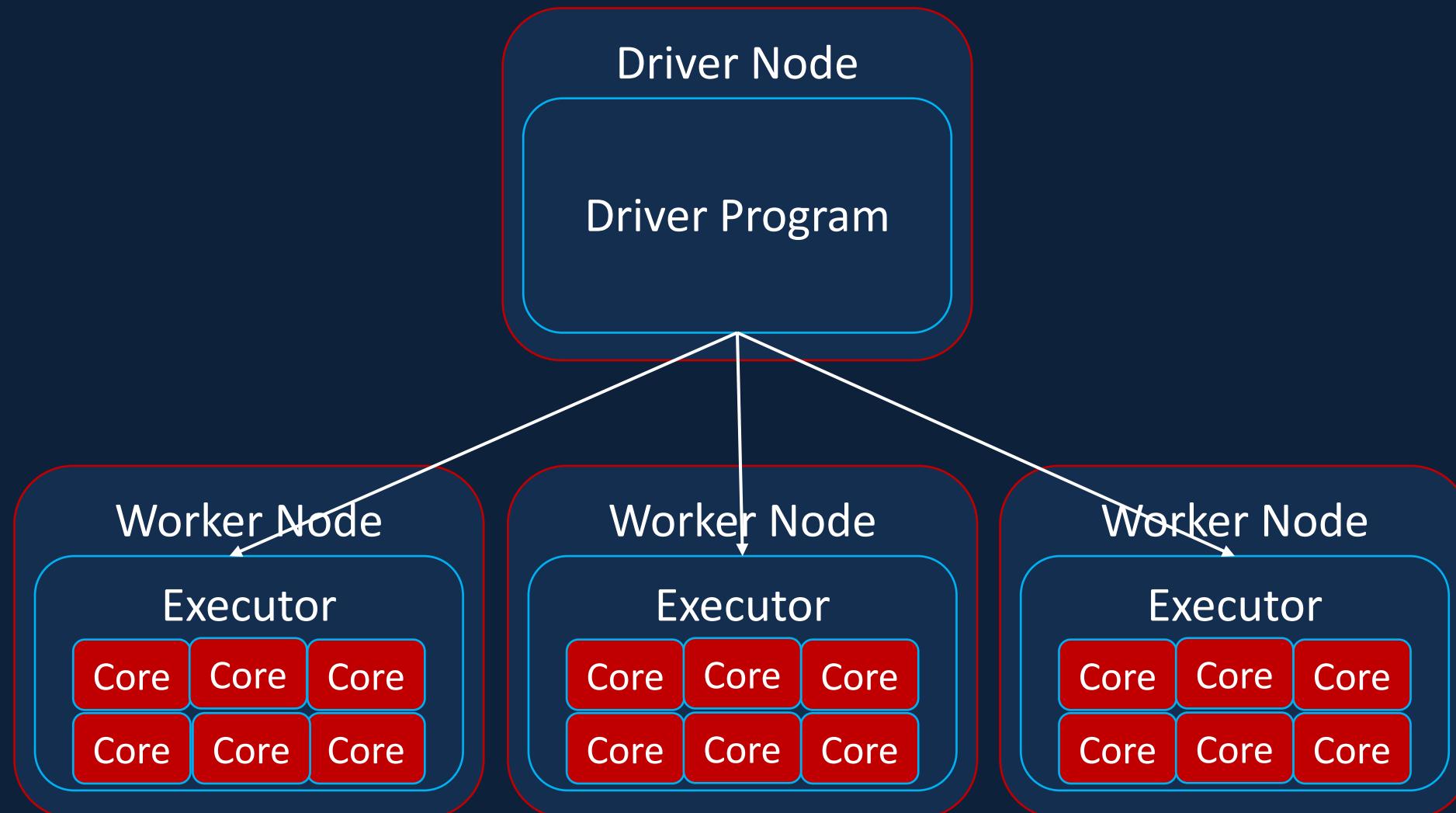
# Spark Architecture



# Spark Architecture



# Spark Architecture – Cluster Scaling



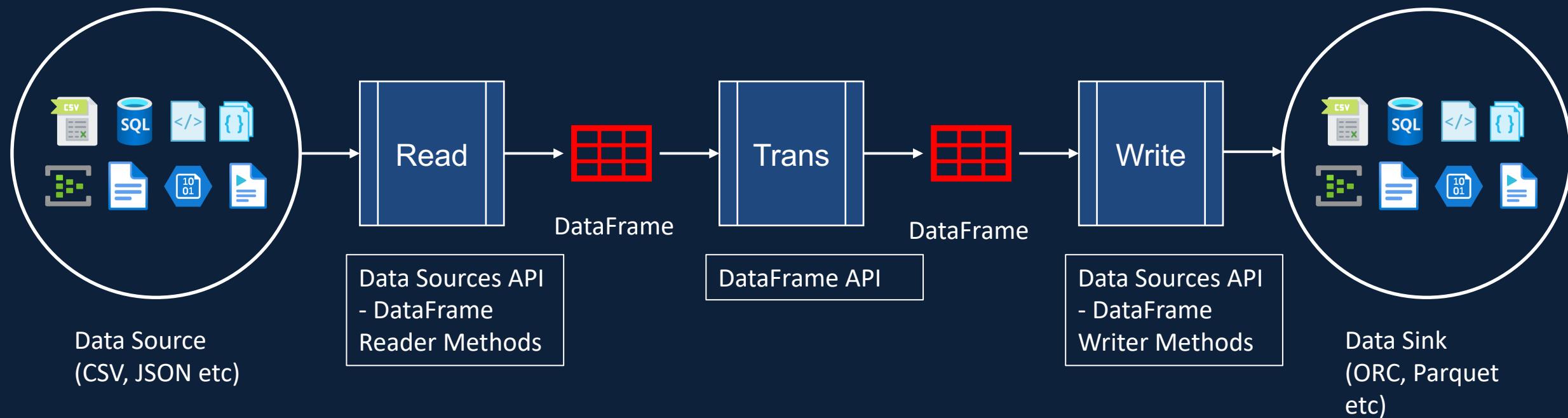
# Spark DataFrame

# 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

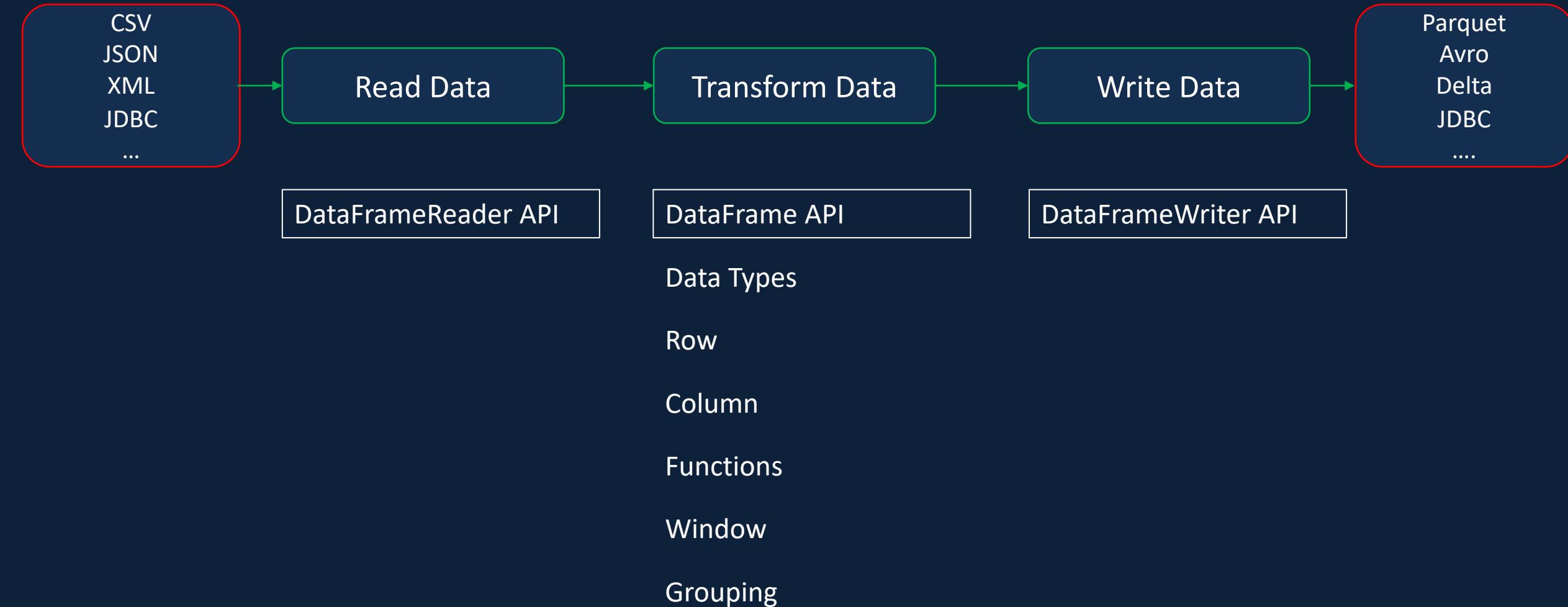
Ingested data must have audit columns

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)

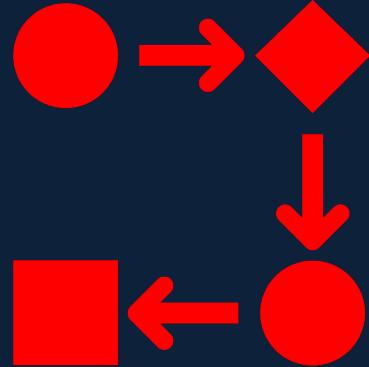
# Data Ingestion – Qualifying (Assignment)



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

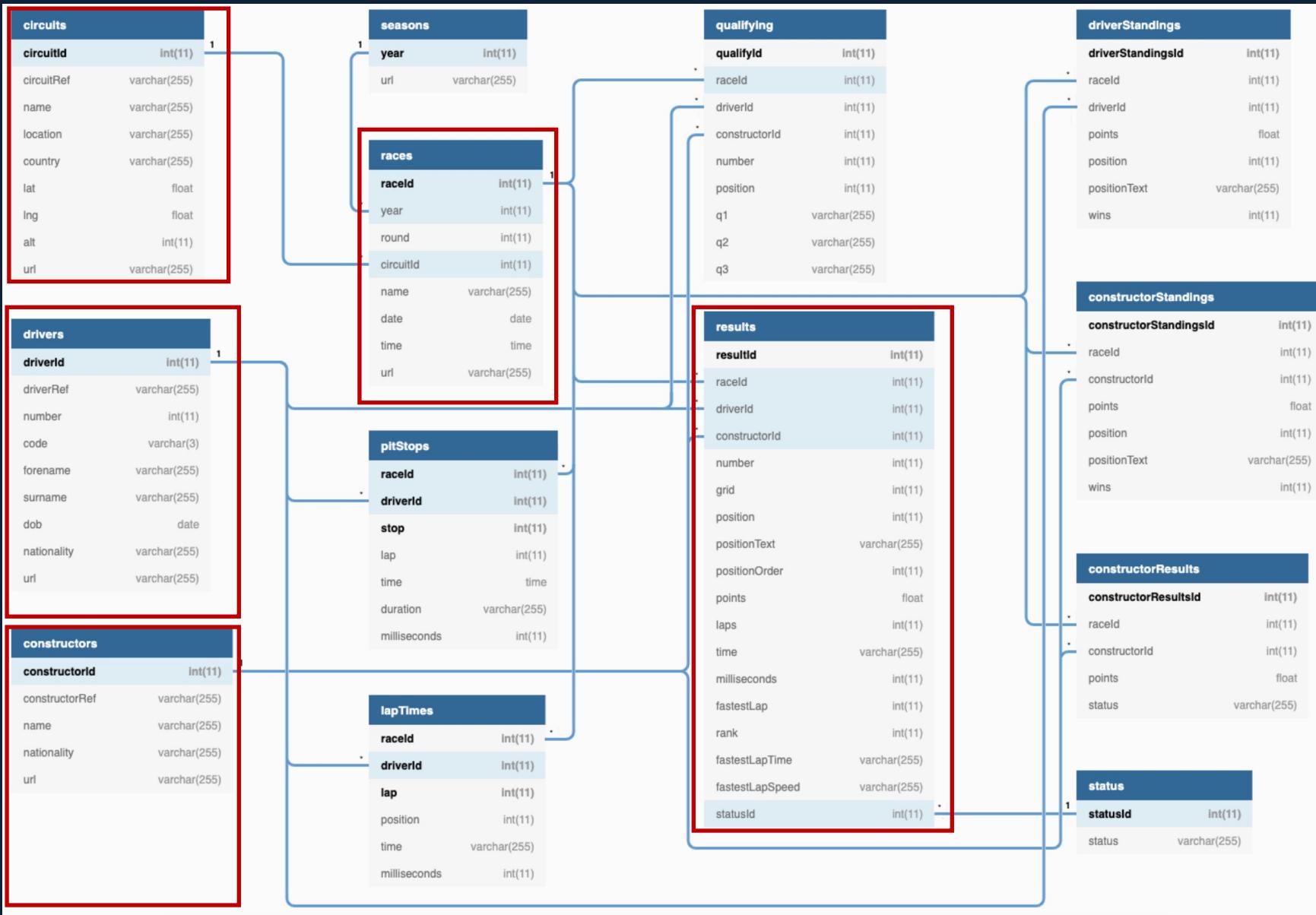
# Filter Transformations

# Join Transformations

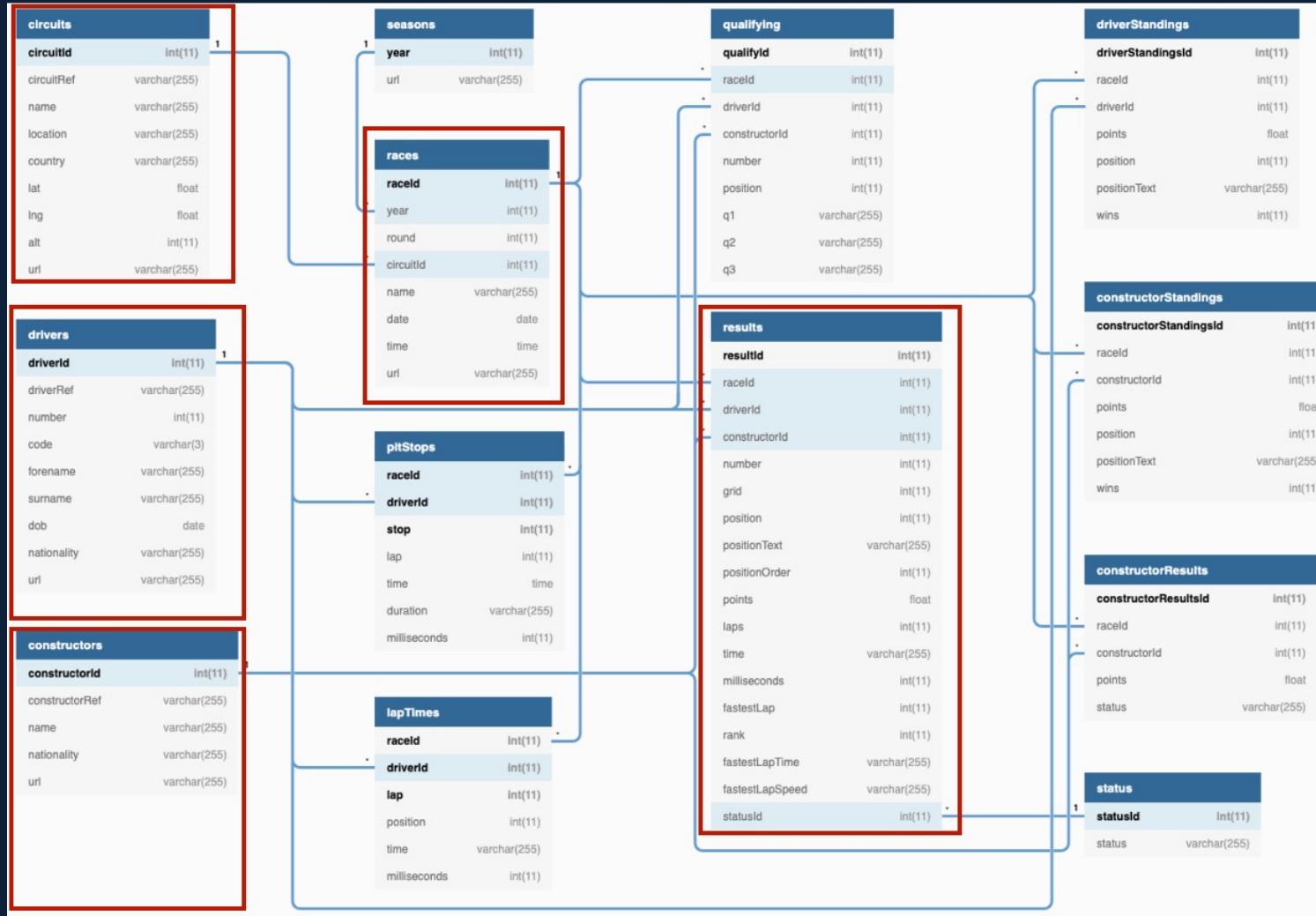
# Join Transformation

## Race Results

# Join –Race Results



# Join -Race Results

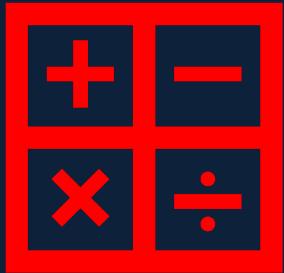


Column Name	Source
race_year	races
race_name	races
race_date	races
circuit_location	circuits
driver_name	drivers
driver_number	drivers
driver_nationality	drivers
team	constructors
grid	results
fastest_lap	results
race_time	results
points	results
created_date	current_timestamp

# Set-up Environment Presentation Layer



# Aggregations



Simple Aggregations

Grouped Aggregations

Window Functions

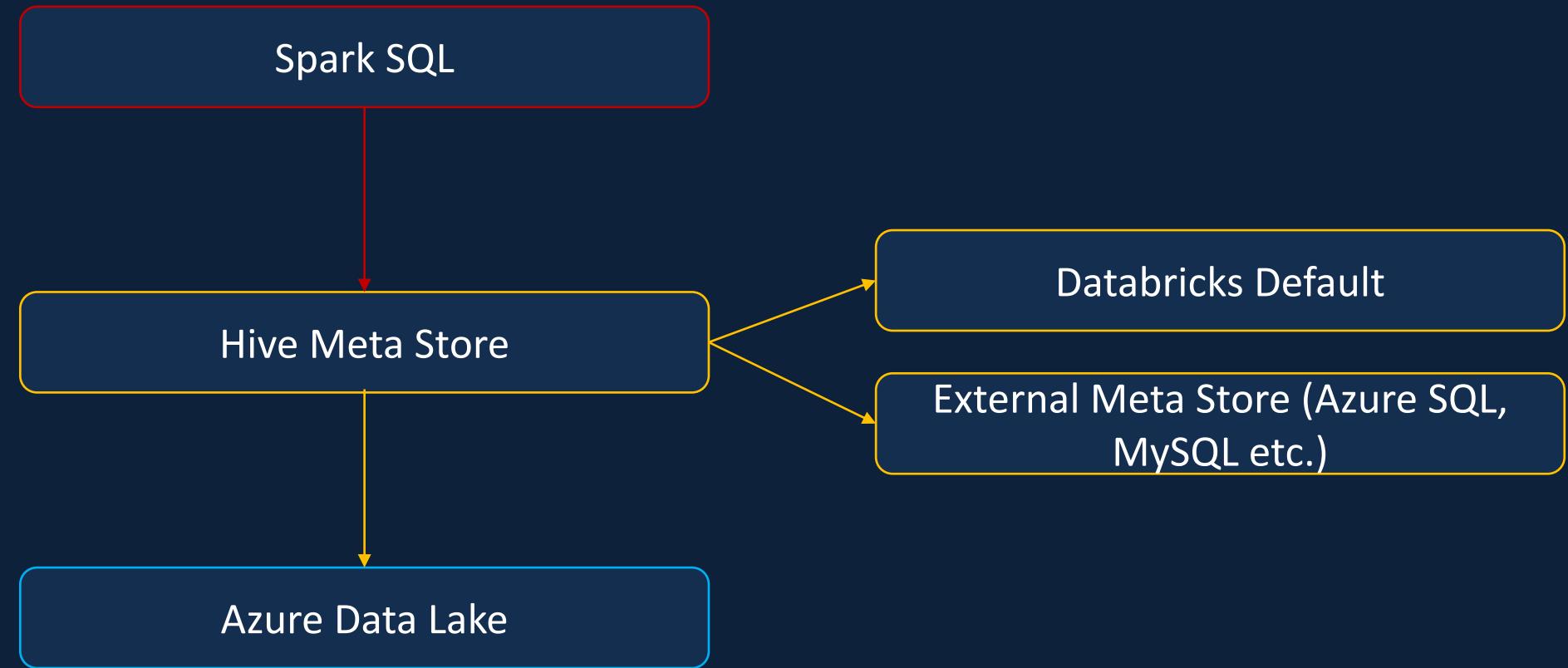
Apply Aggregations to F1 Project

# Built-in Aggregate Functions

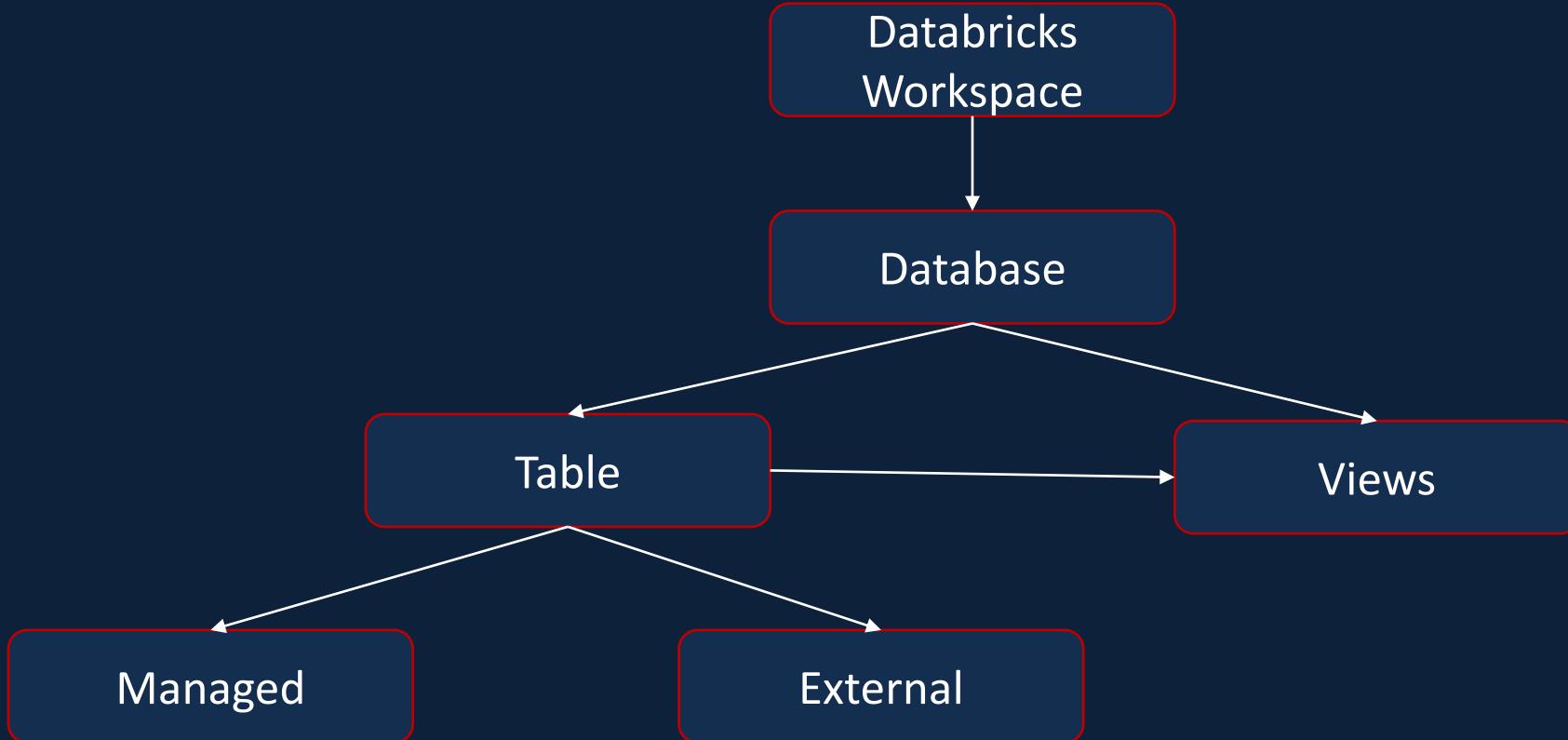
# Group By

# Databases/ Tables/ Views

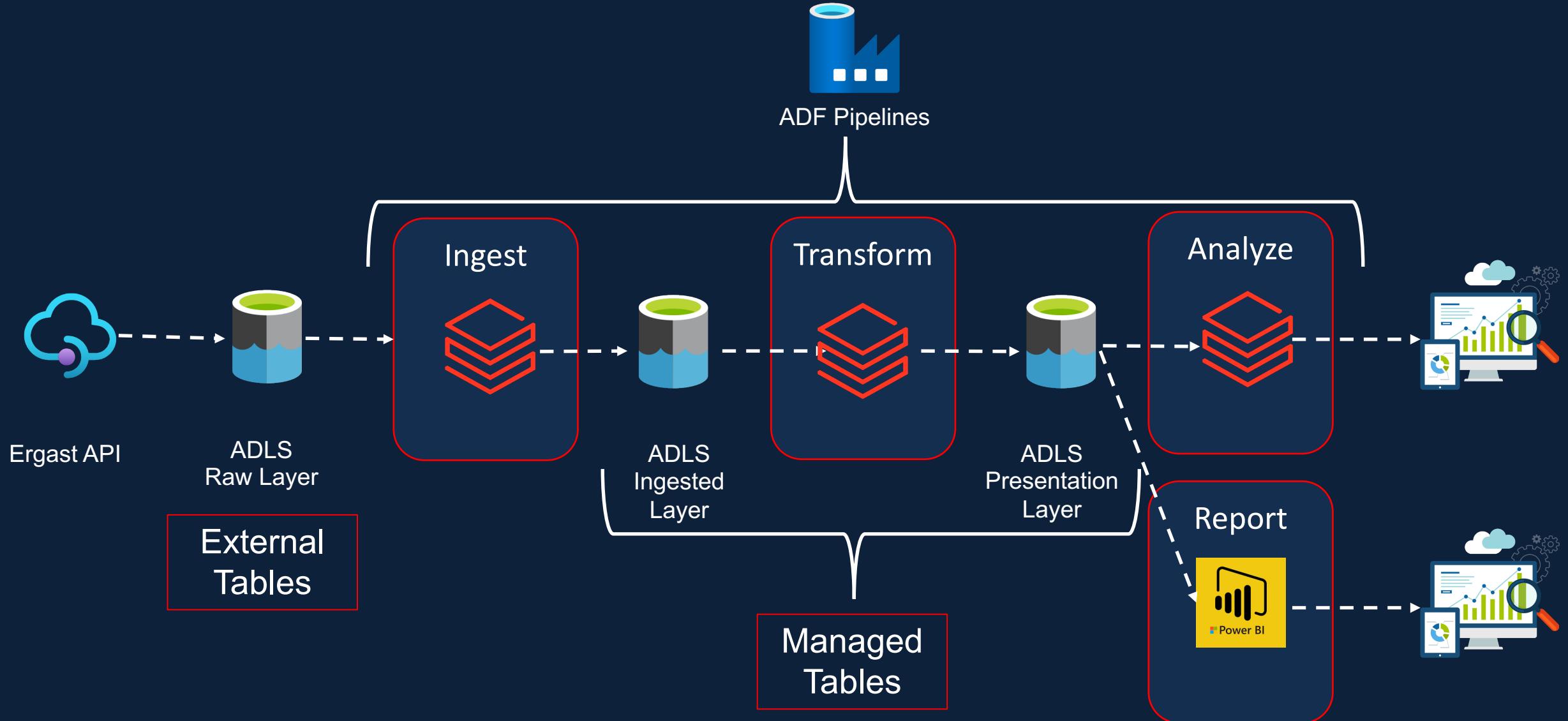
# Hive Meta Store



# 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

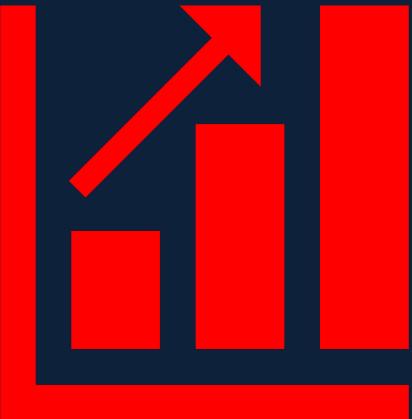
Rank the dominant drivers of all time/ last decade etc

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

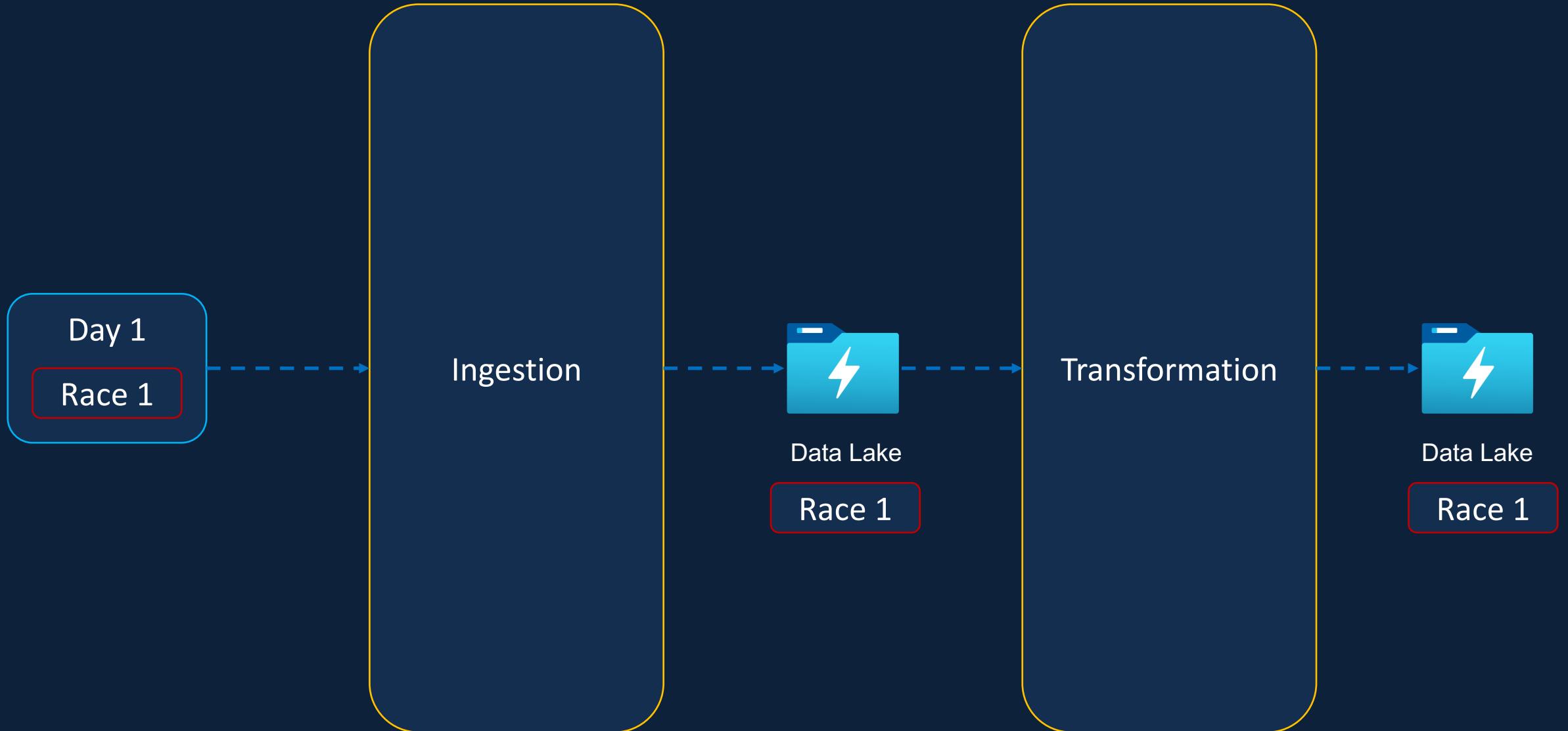
Full Load

Incremental Load

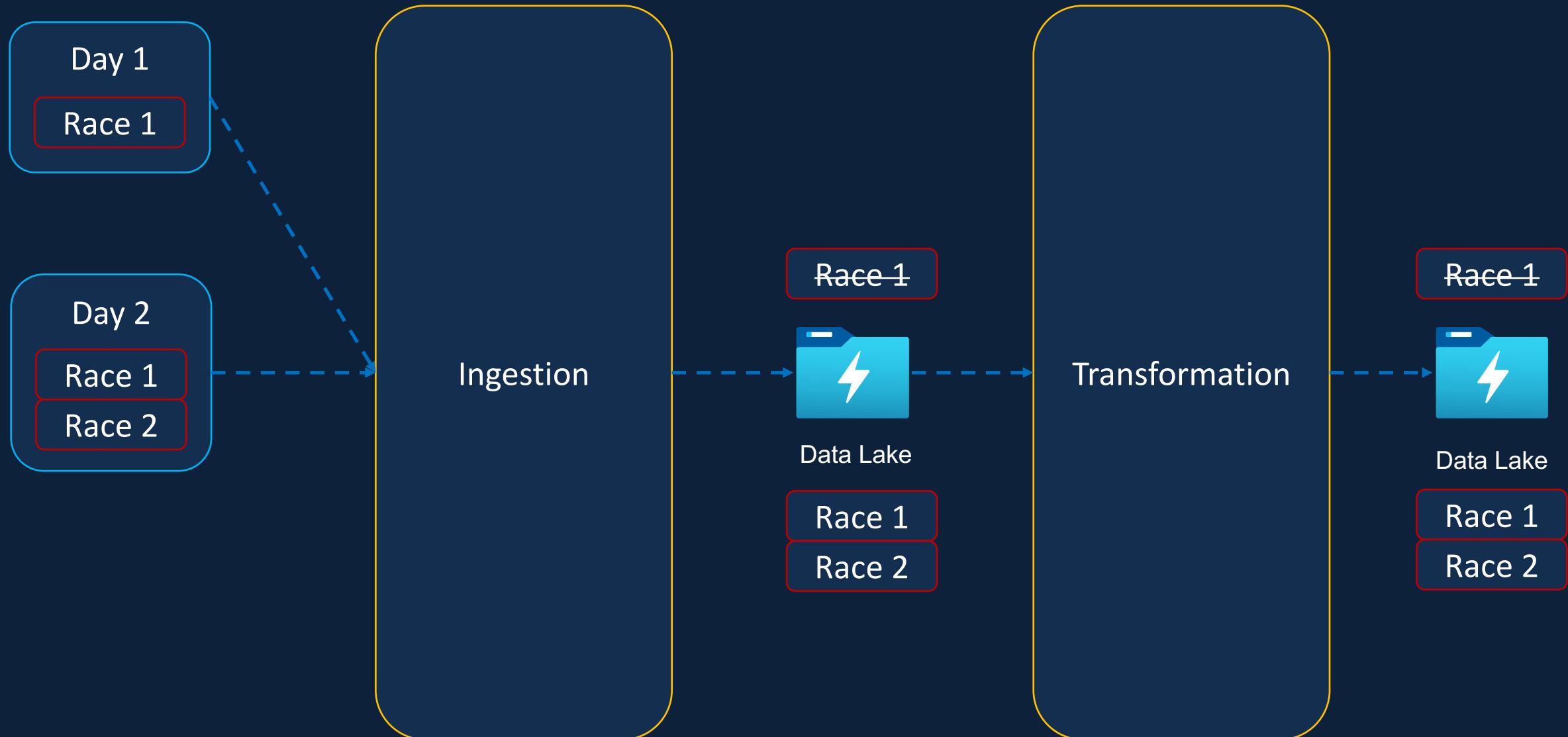
# Full Dataset



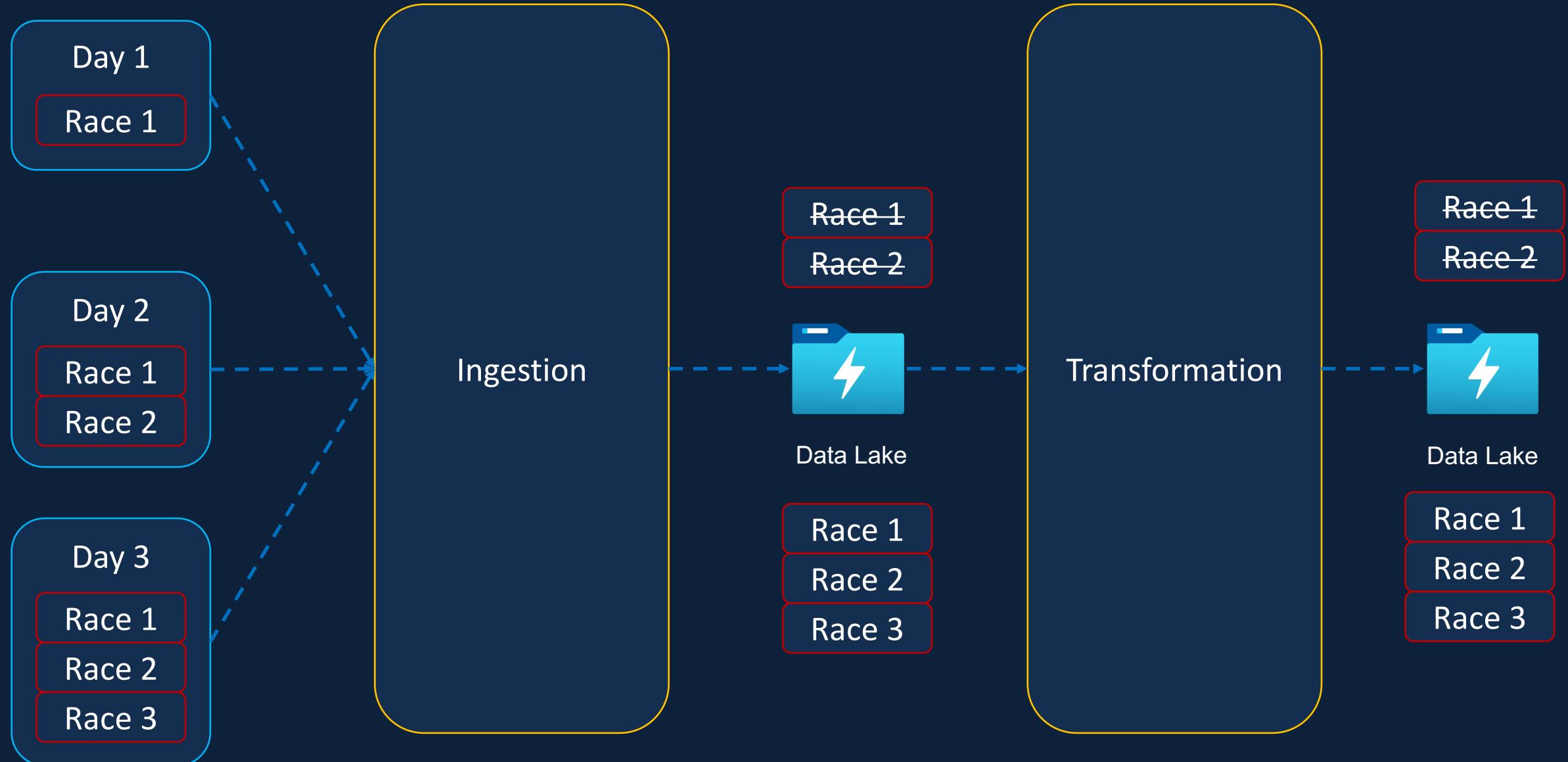
# Full Refresh/ Load – Day 1



# Full Refresh/ Load – Day 2



# Full Refresh/ Load – Day 3



# Incremental Dataset

Day 1

Race 1

Day 2

Race 2

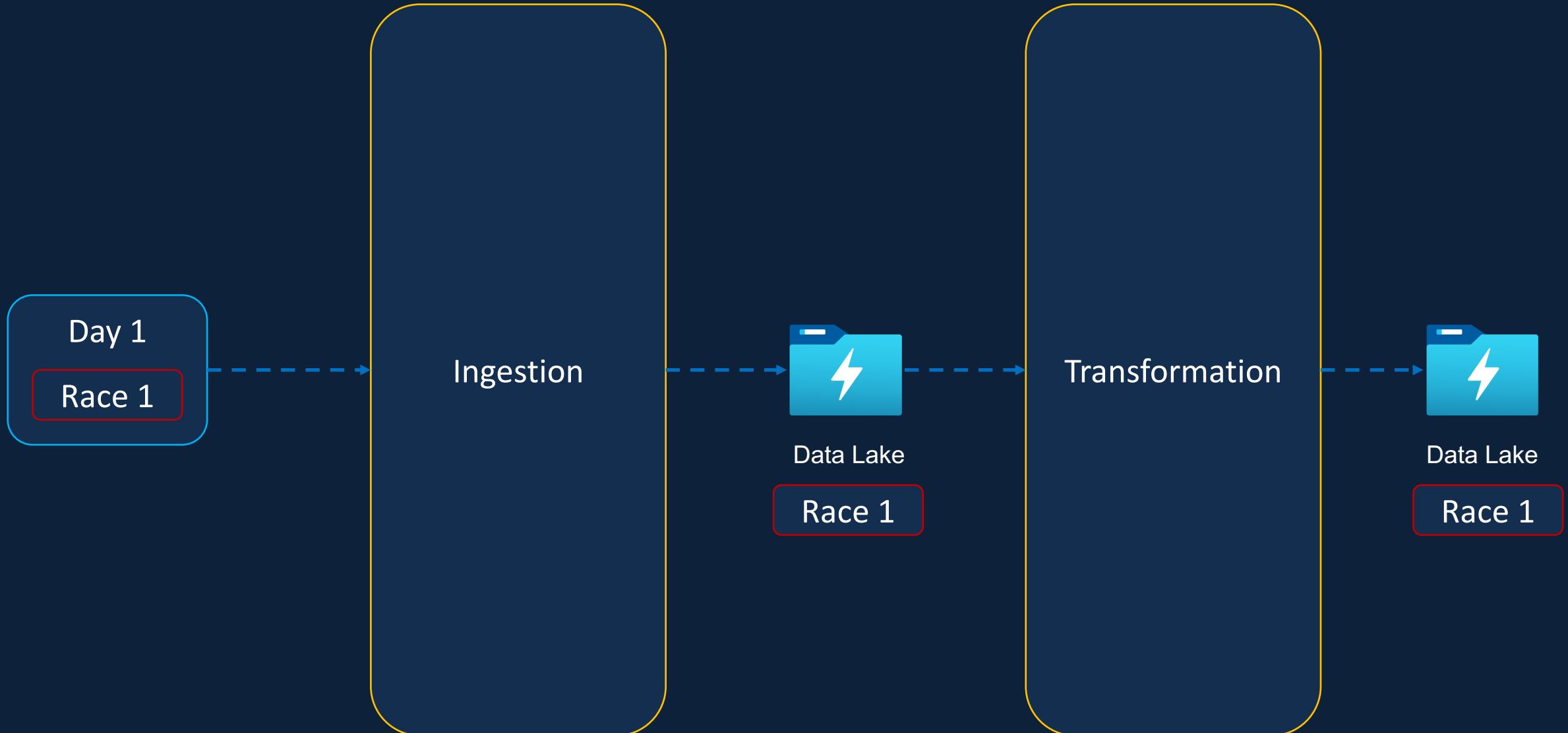
Day 3

Race 3

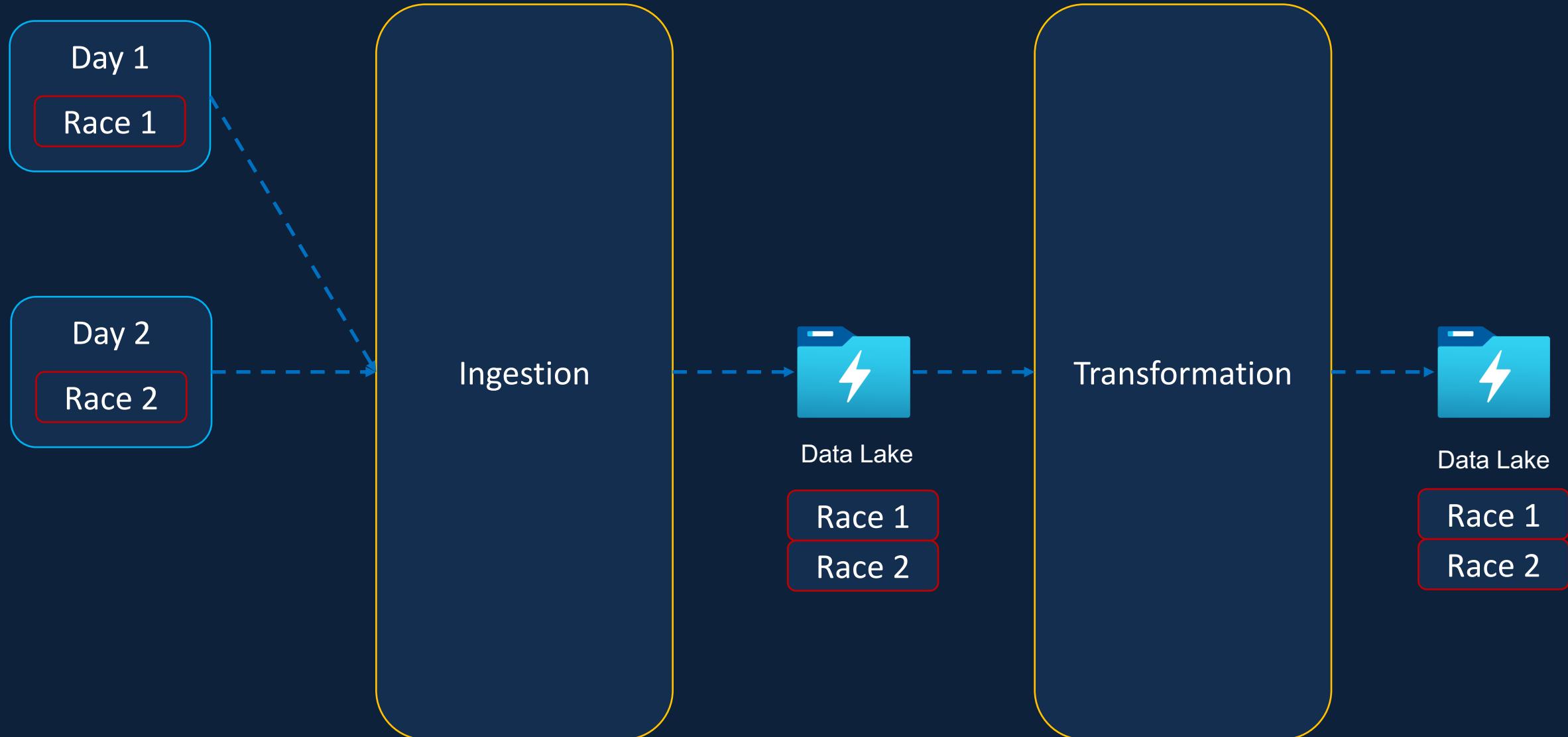
Day 4

Race 4

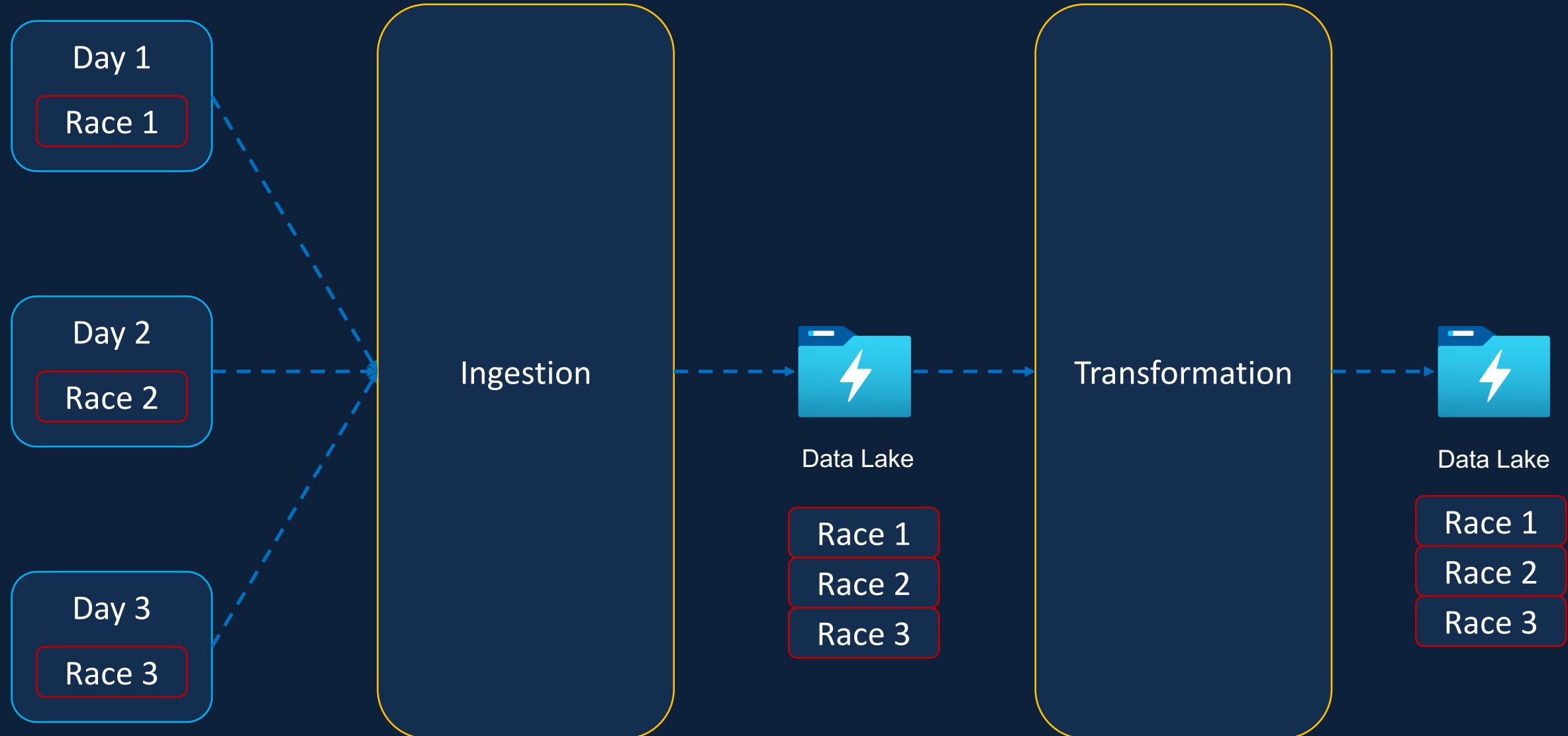
# Incremental Load – Day 1



# Incremental Load – Day 2



# Incremental Load – Day 3



# Hybrid Scenarios

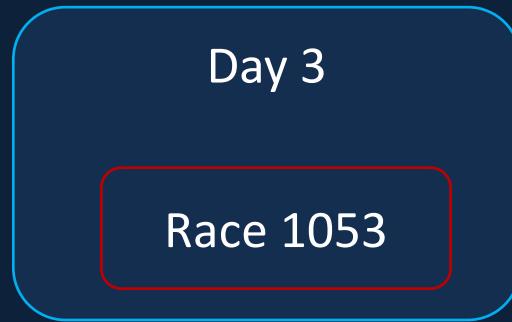
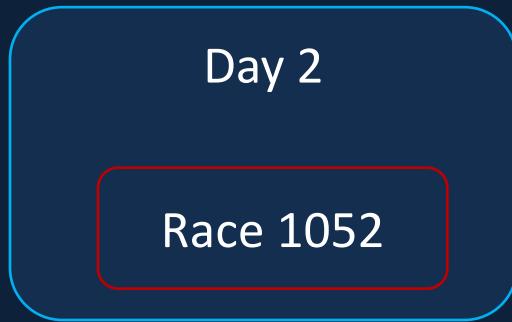
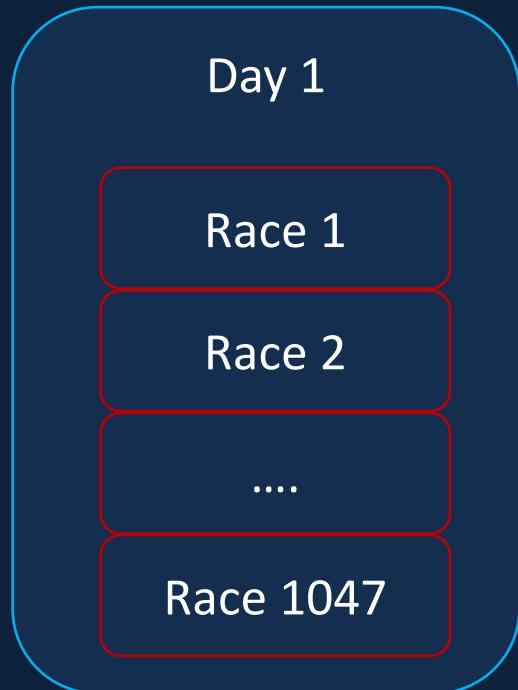
Full Dataset received, but data loaded & transformed incrementally

Incremental dataset received, but data loaded & transformed in full

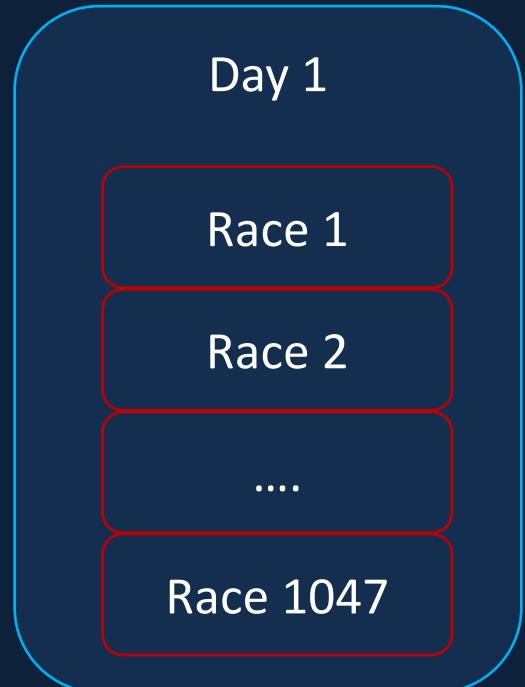
Data received contains both full and incremental files

Incremental data received. Ingested incrementally & transformed in full

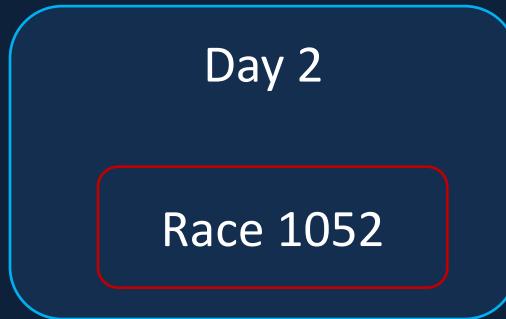
# Formula1 Scenario



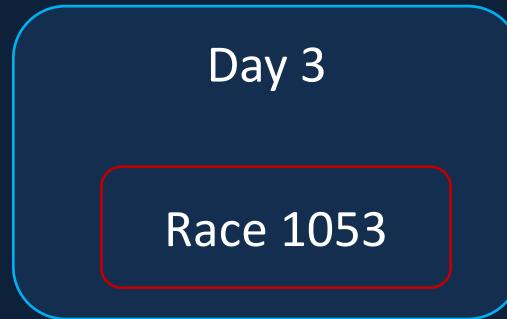
# Formula1 Scenario / Solution 1



Full Refresh

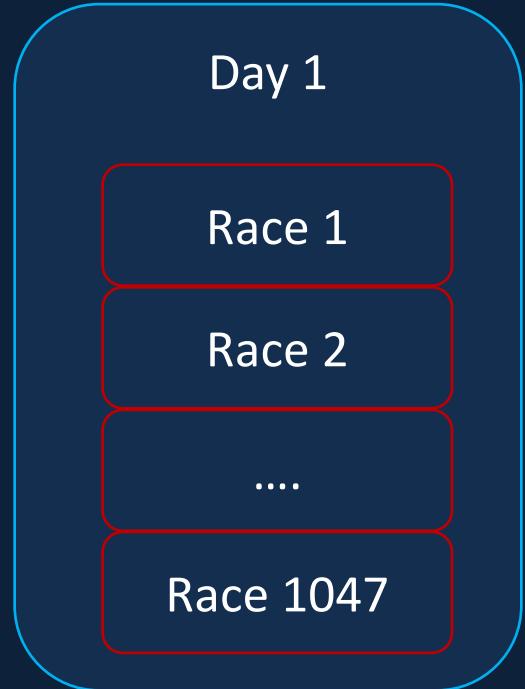


Incremental Load

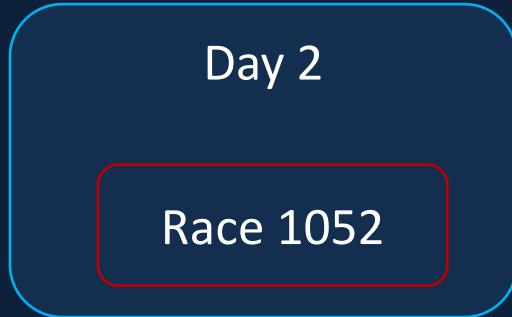


Incremental Load

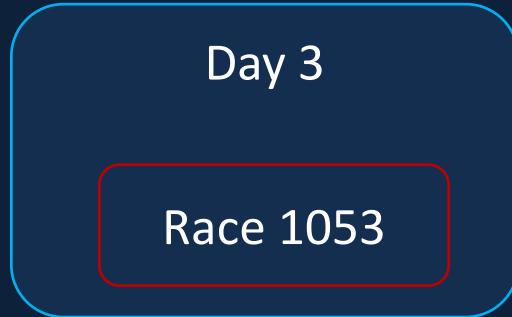
# Formula1 Scenario / Solution 2



Incremental Load



Incremental Load



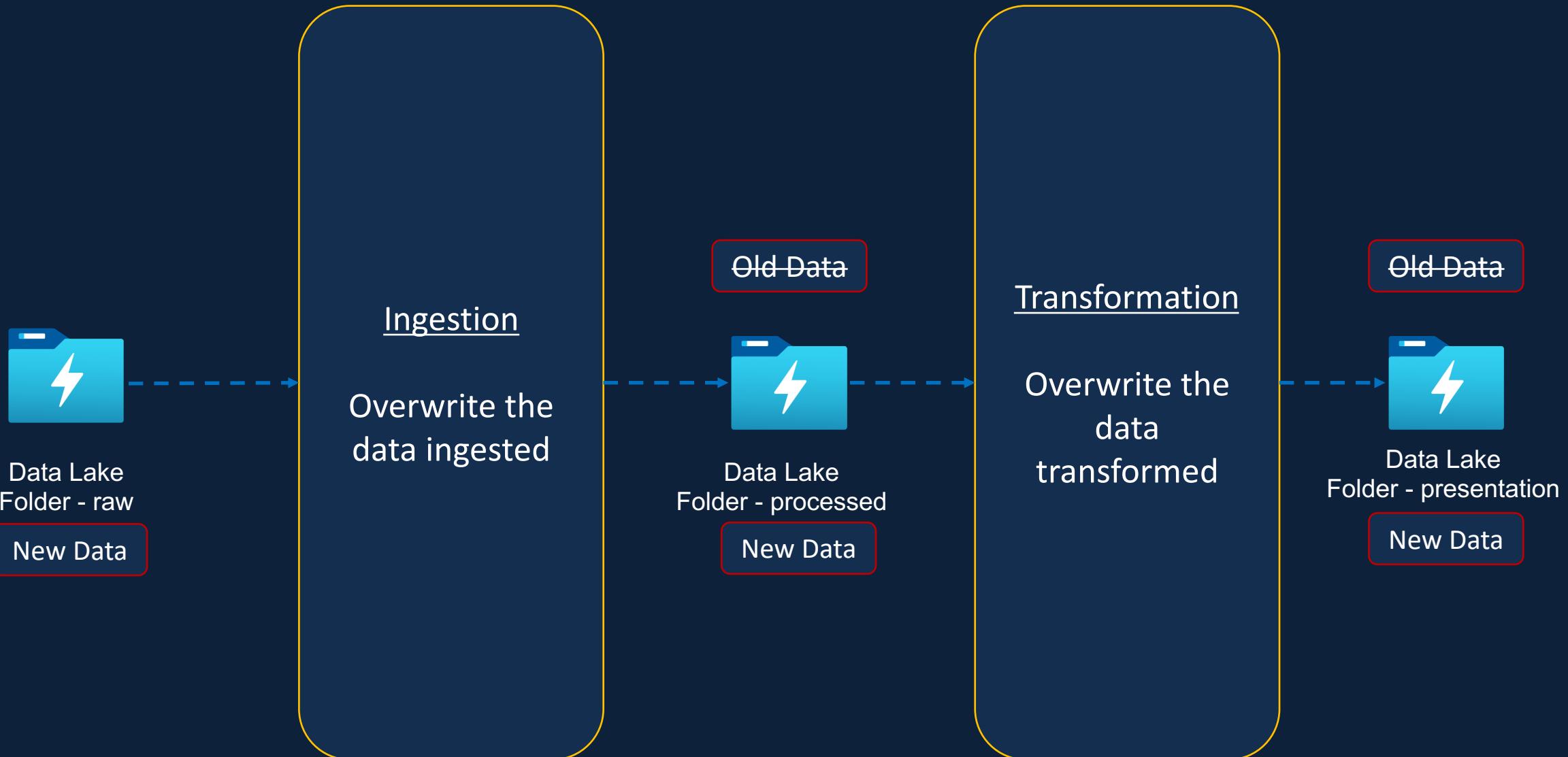
Incremental Load

# Formula1 Data Files

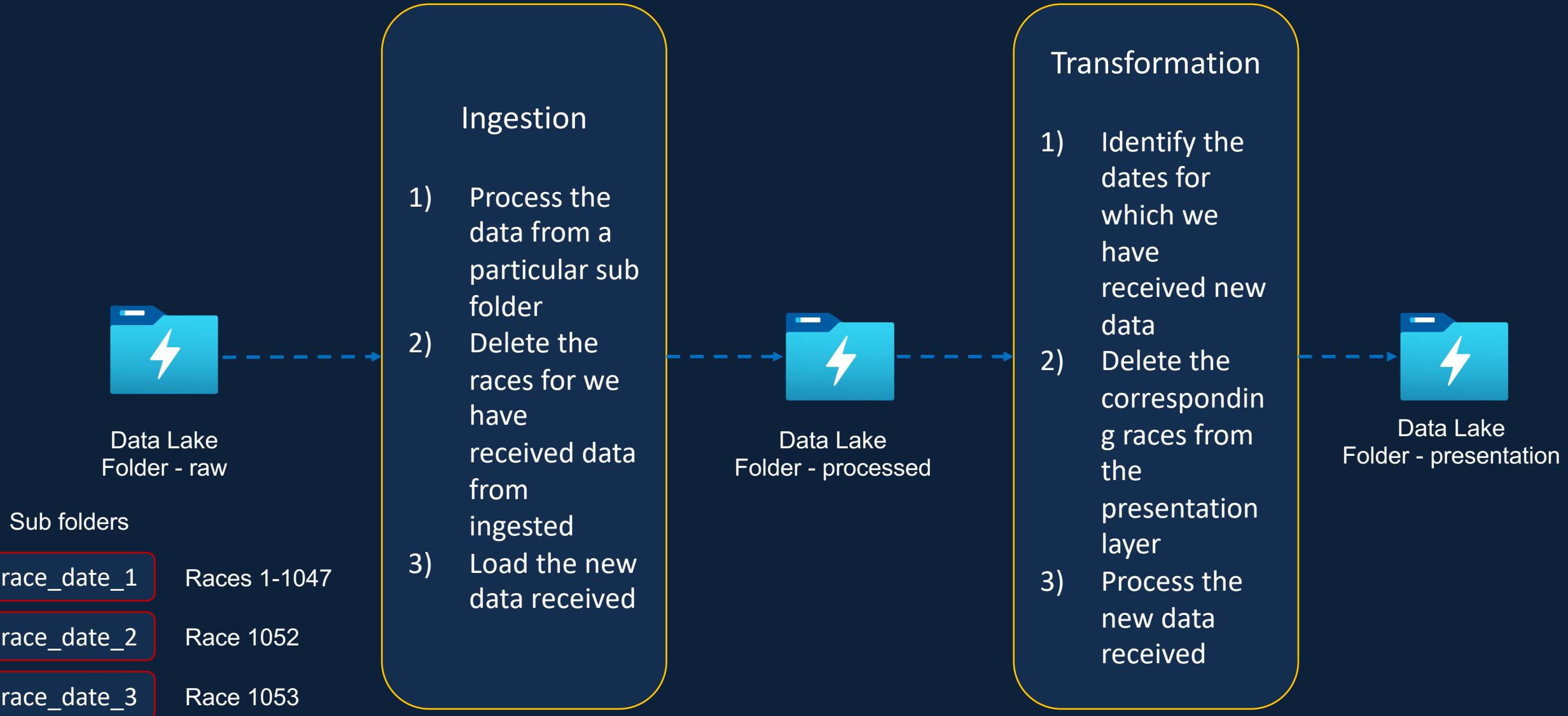


Circuits	Data from all races
Races	Data from all races
Constructors	Data from all races
Drivers	Data from all races
Results	Data only from that race
PitStops	Data only from that race
LapTimes	Data only from that race
Qualifying	Data only from that race

# Current Solution



# New Solution



# Delta Lake



Pitfalls of Data Lakes

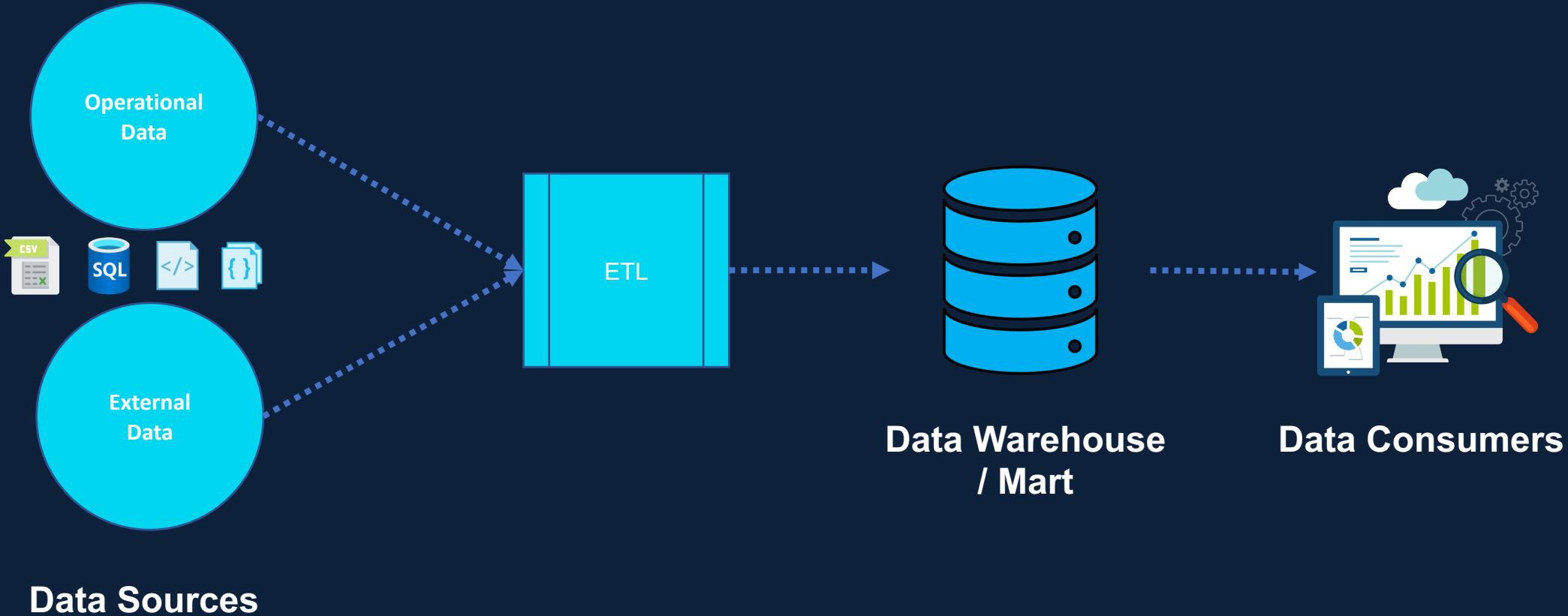
Lakehouse Architecture

Delta Lake Capabilities

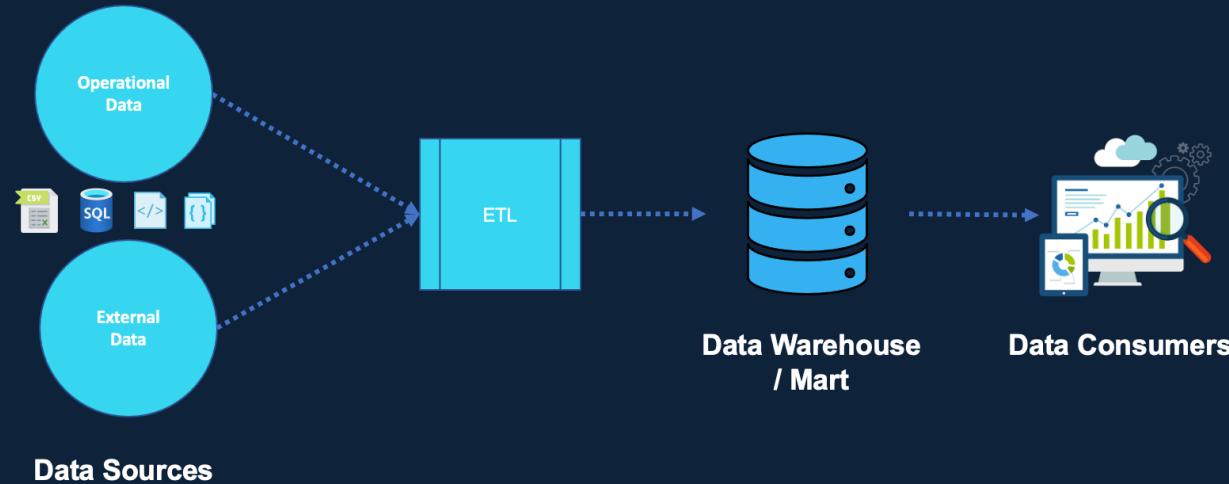
Convert F1 project to Delta Lake

# Delta Lake

# Data Warehouse



# Data Warehouse



Lack of support for unstructured data

Longer to ingest new data

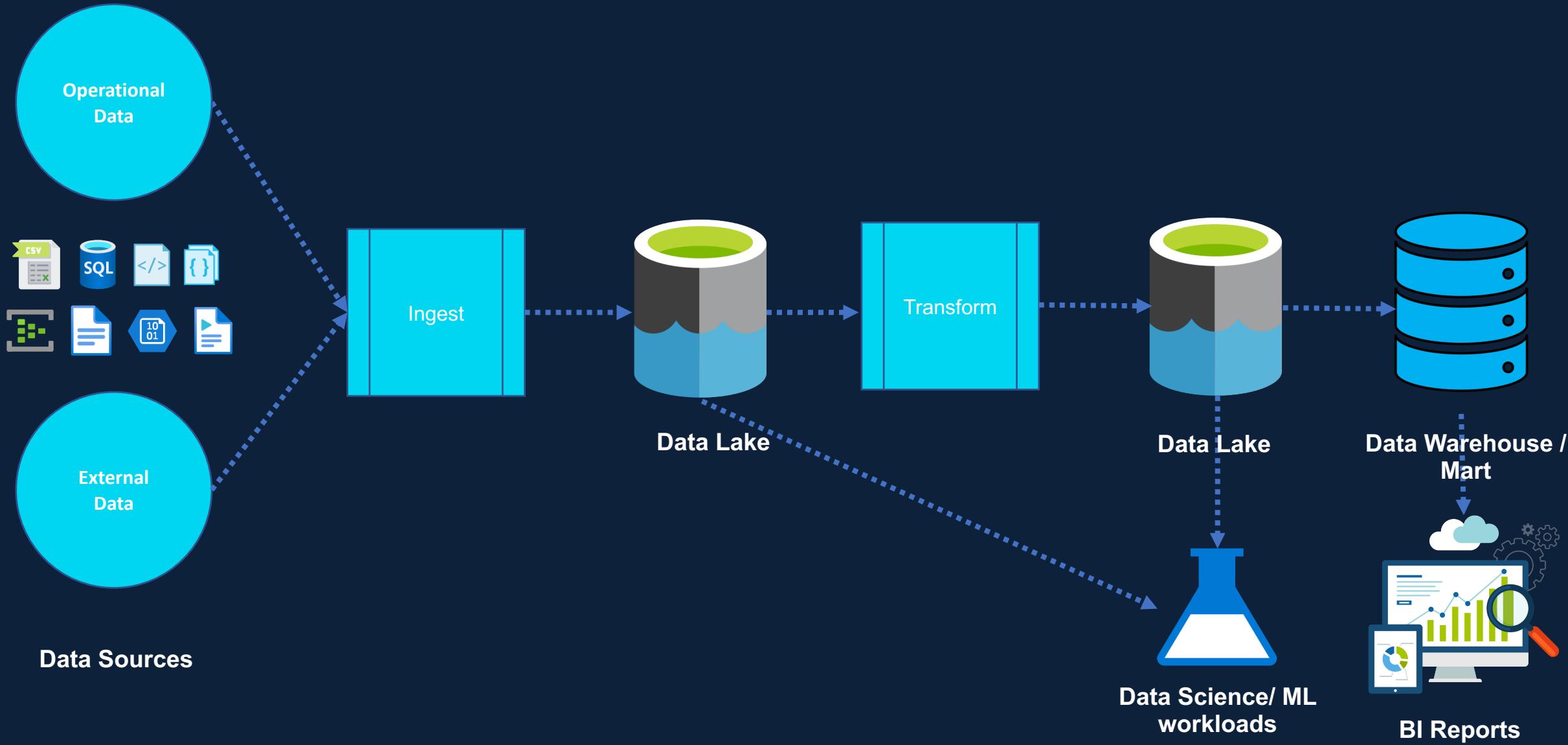
Proprietary data formats

Scalability

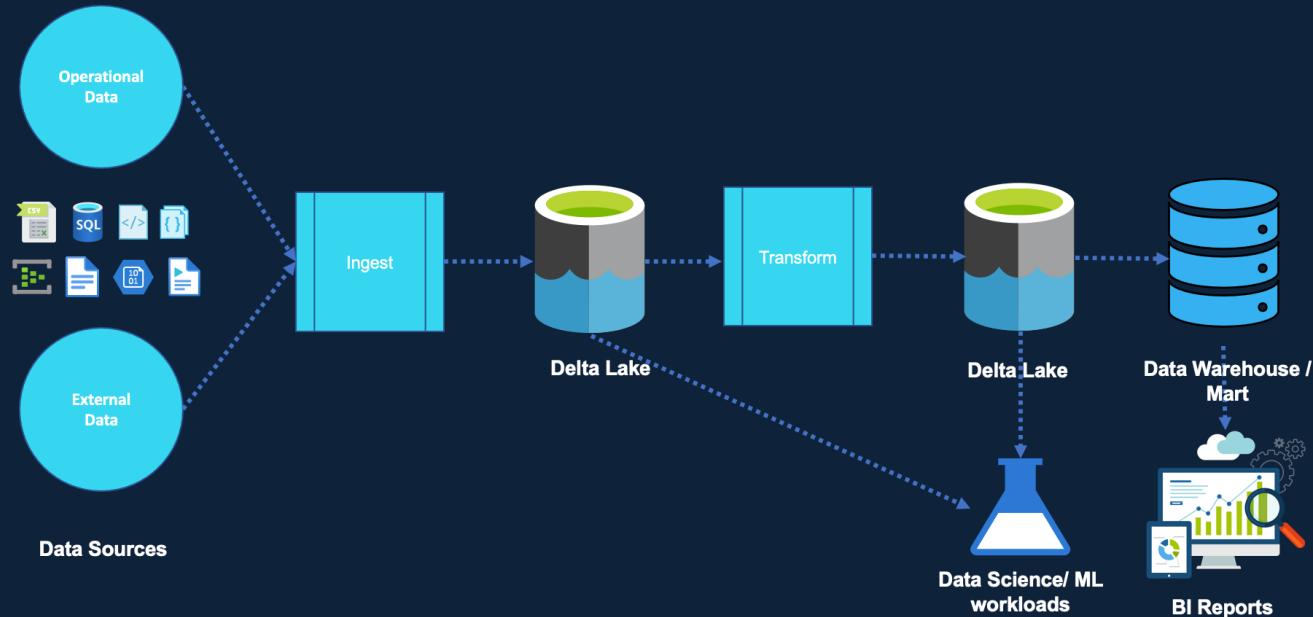
Expensive to store data

Lack of support for ML/ AI workloads

# Data Lake

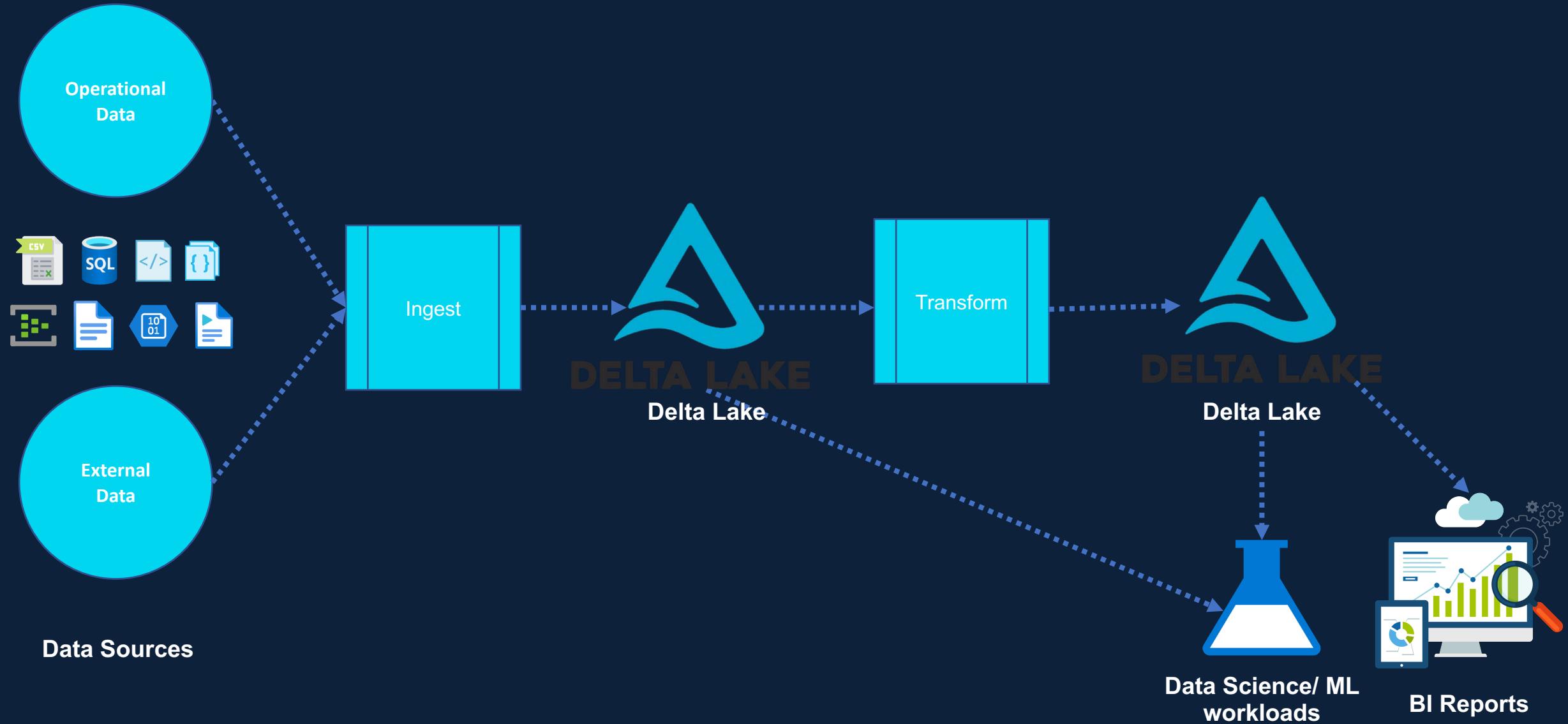


# Data Lake

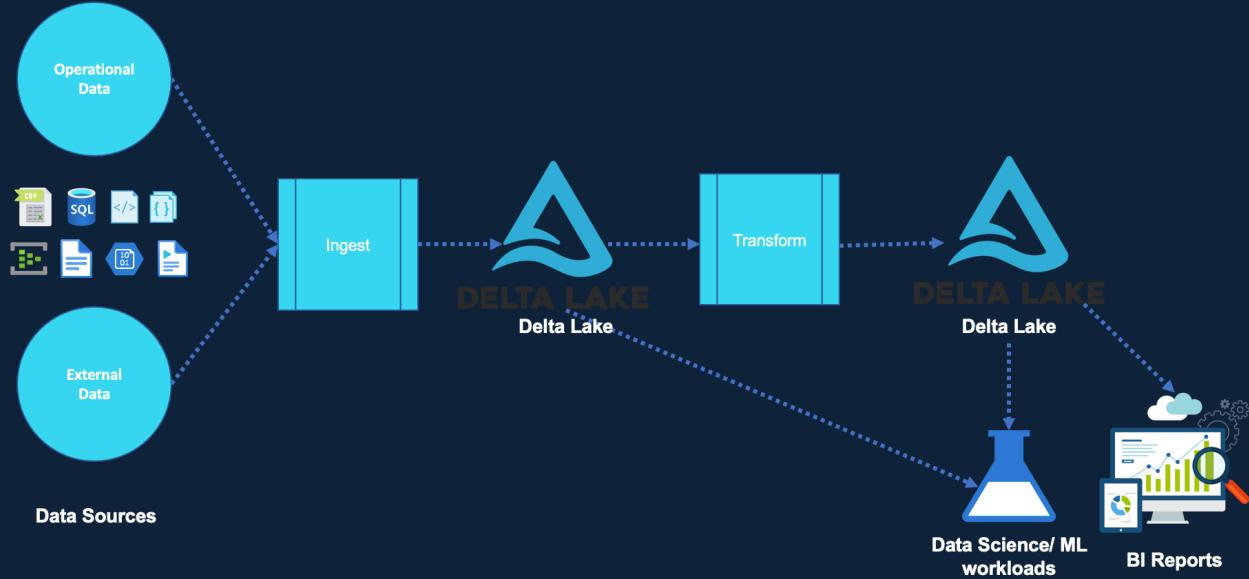


- No support for ACID transactions
- Failed jobs leave partial files
- Inconsistent reads
- Unable to handle corrections to data
- Unable to roll back any data.
- Lack of ability remove data for GDPR etc
- No history or versioning
- Poor performance
- Poor BI support
- Complex to set-up
- Lambda architecture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

# Data Lakehouse

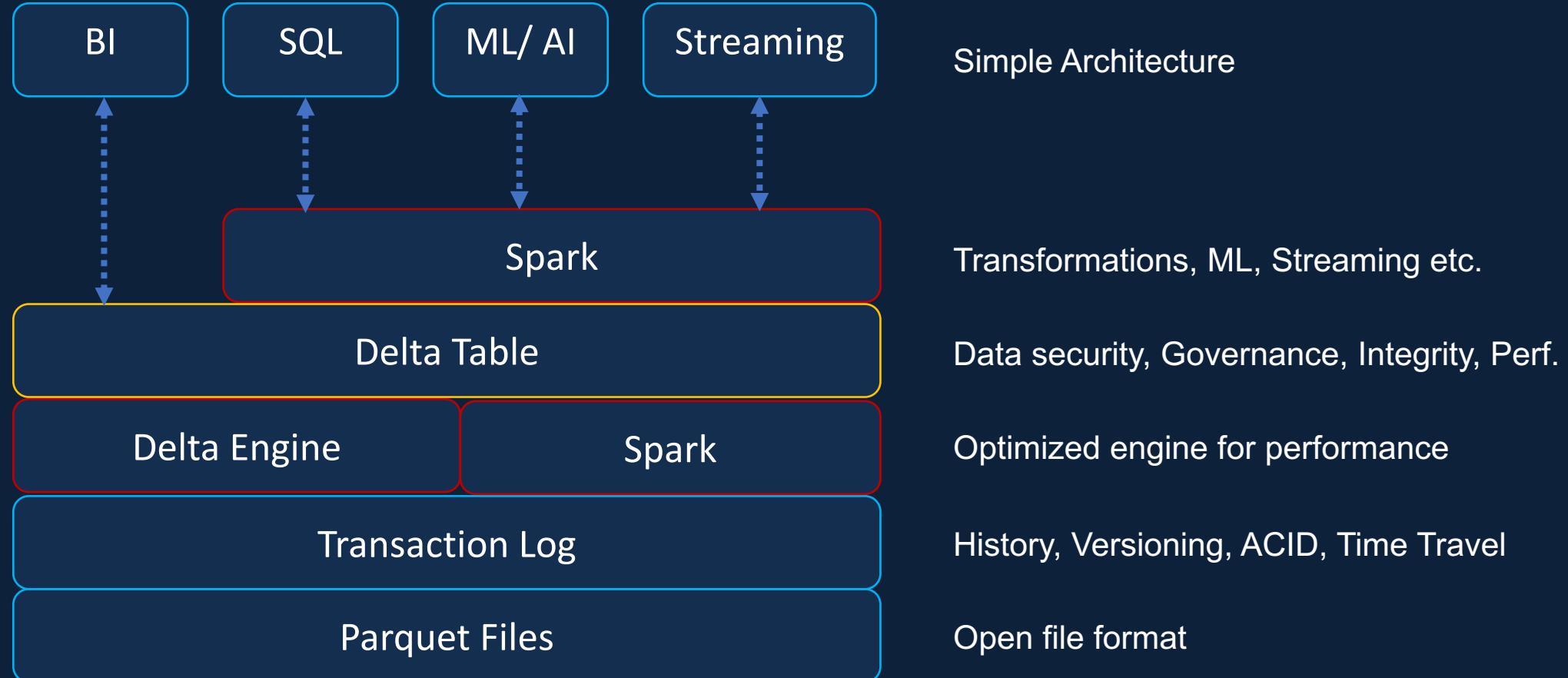


# Data Lakehouse



- 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

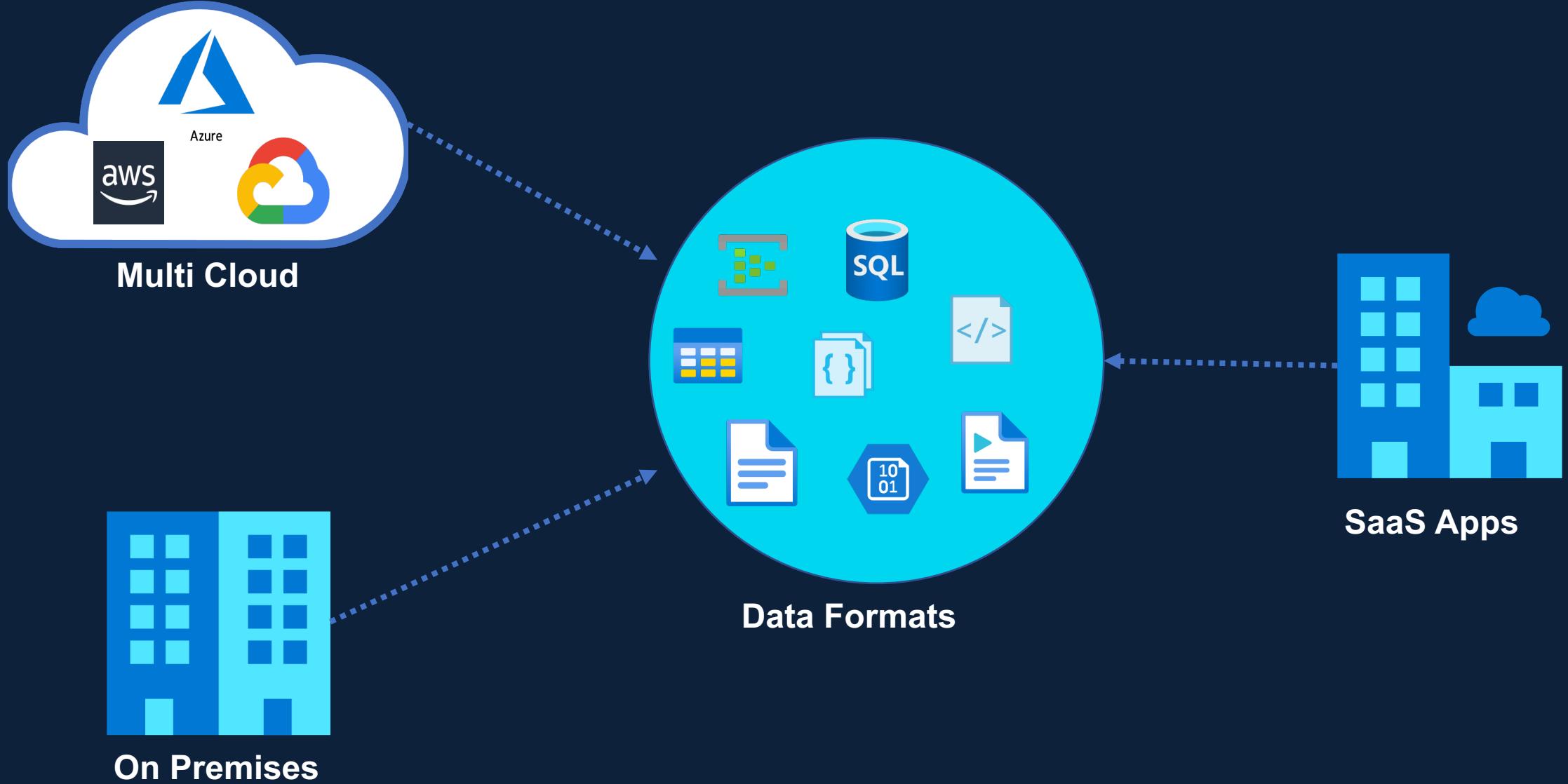
# Azure Data Factory Overview

# What is Azure Data Factory

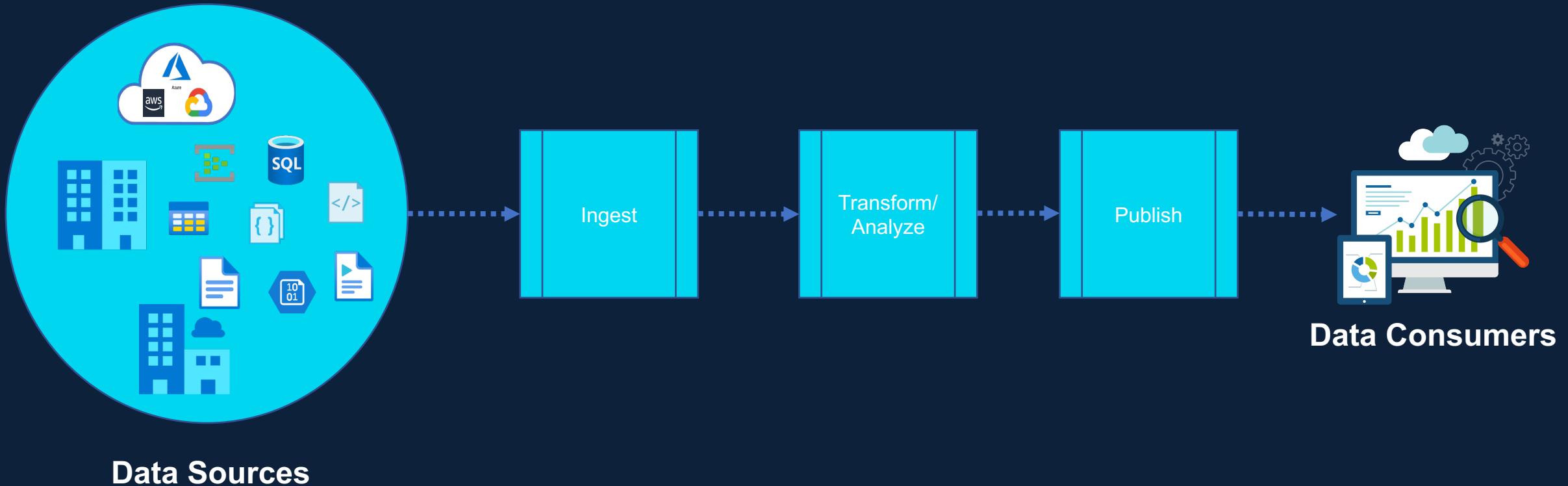


A fully managed, serverless data integration solution for ingesting, preparing and transforming all of your data at scale.

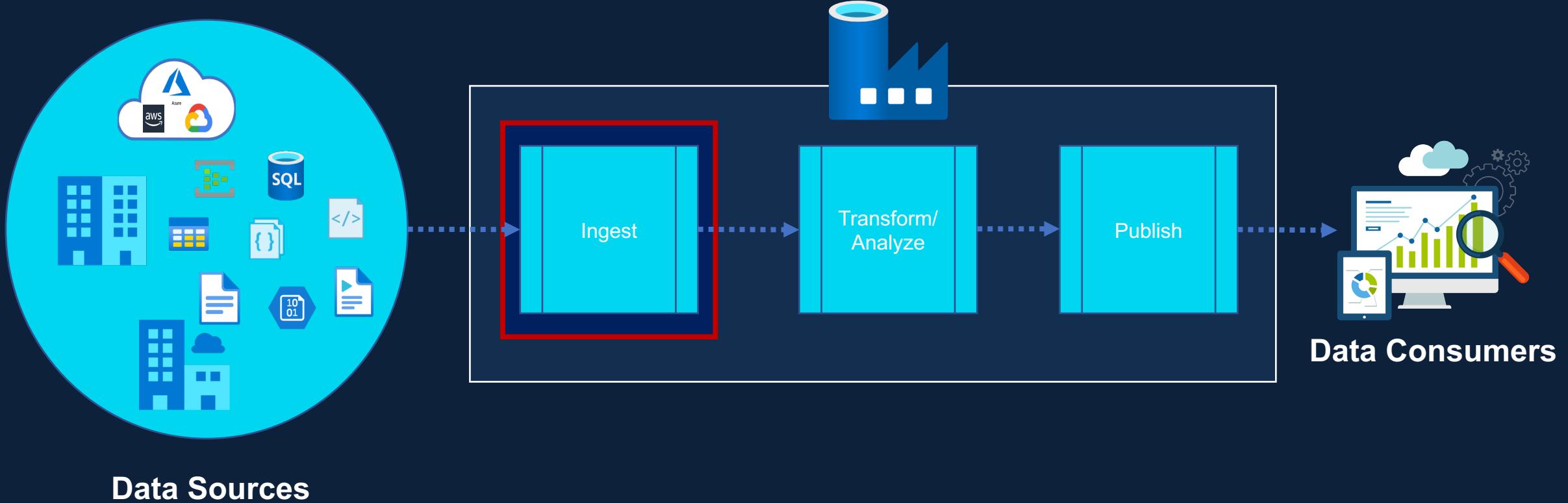
# The Data Problem



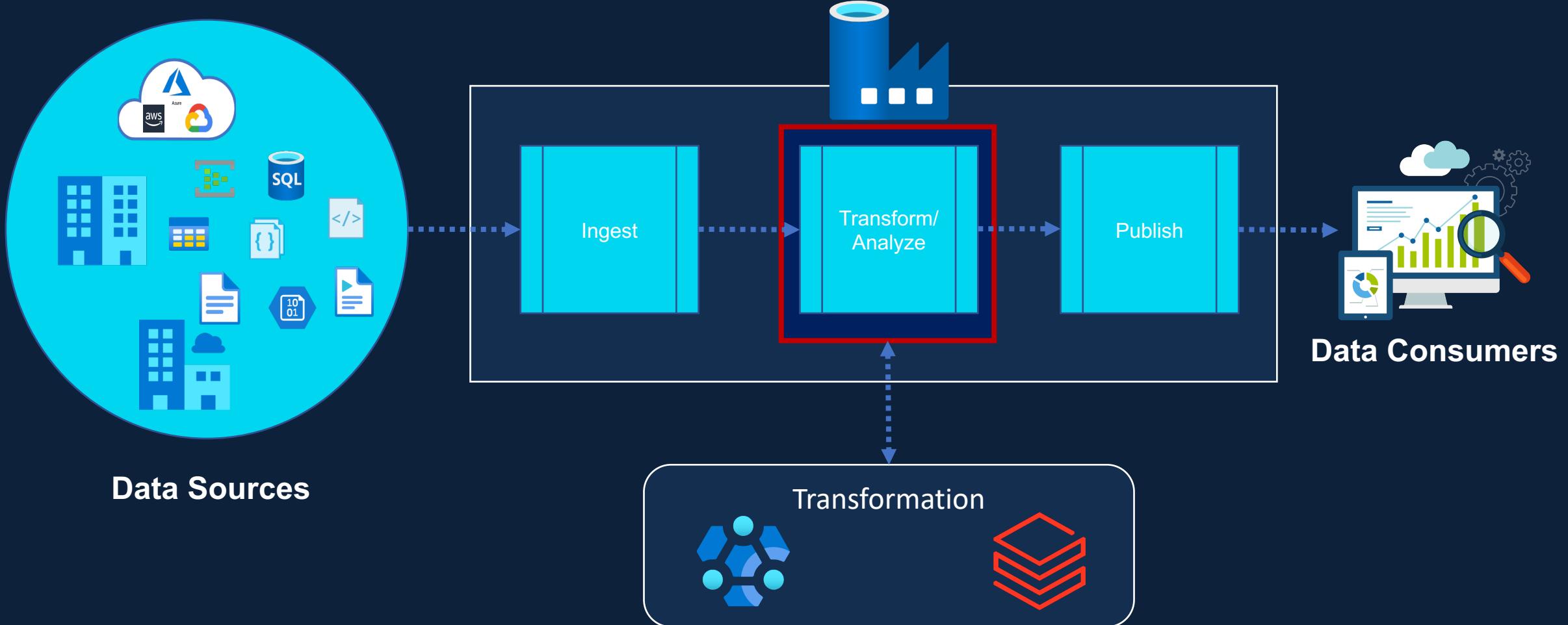
# The Data Problem



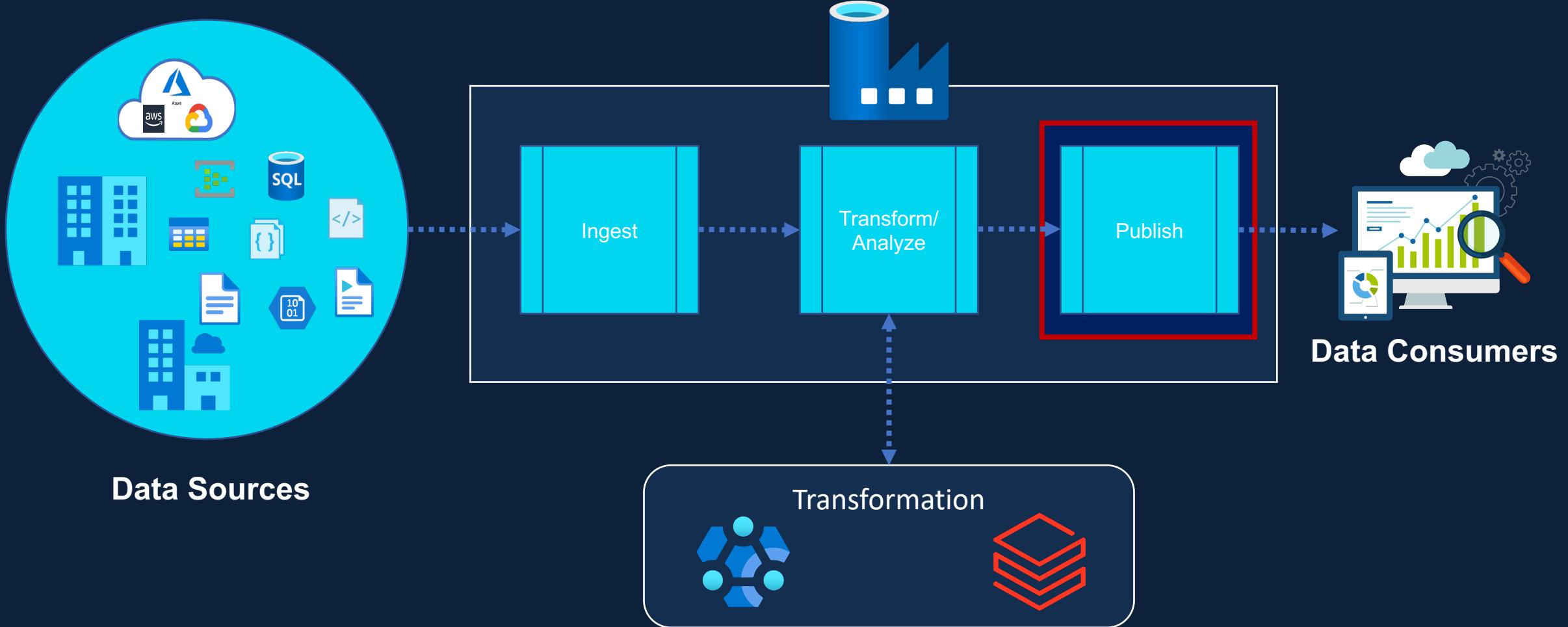
# The Data Problem



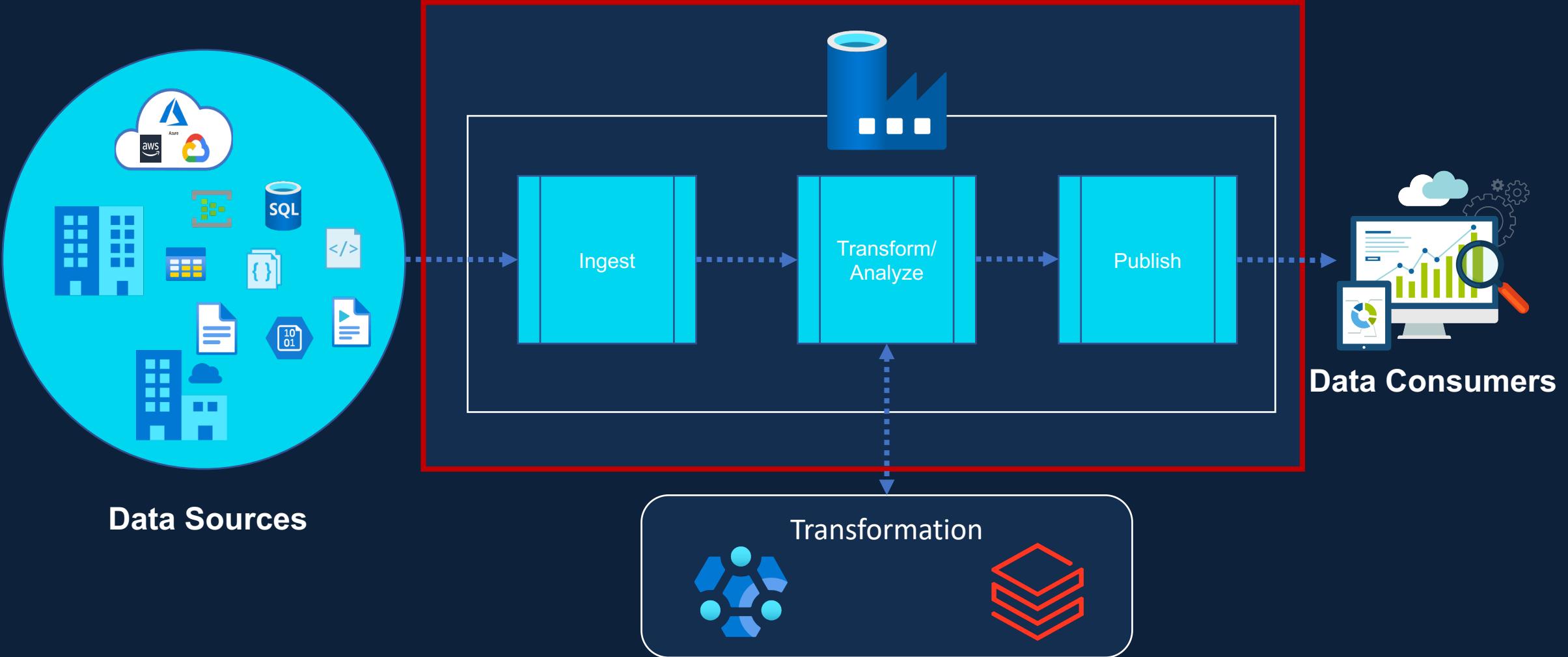
# The Data Problem



# 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

# Unity Catalog - Introduction



Unity Catalog Overview

Enable Unity Catalog Metastore

Cluster Configurations

Unity Catalog Object Model

Access External Data Lake

# Unity Catalog



# Unity Catalog



Unity Catalog is a Databricks offered unified solution for implementing data governance in the Data Lakehouse

# Unity Catalog



Unity Catalog

Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Data Governance

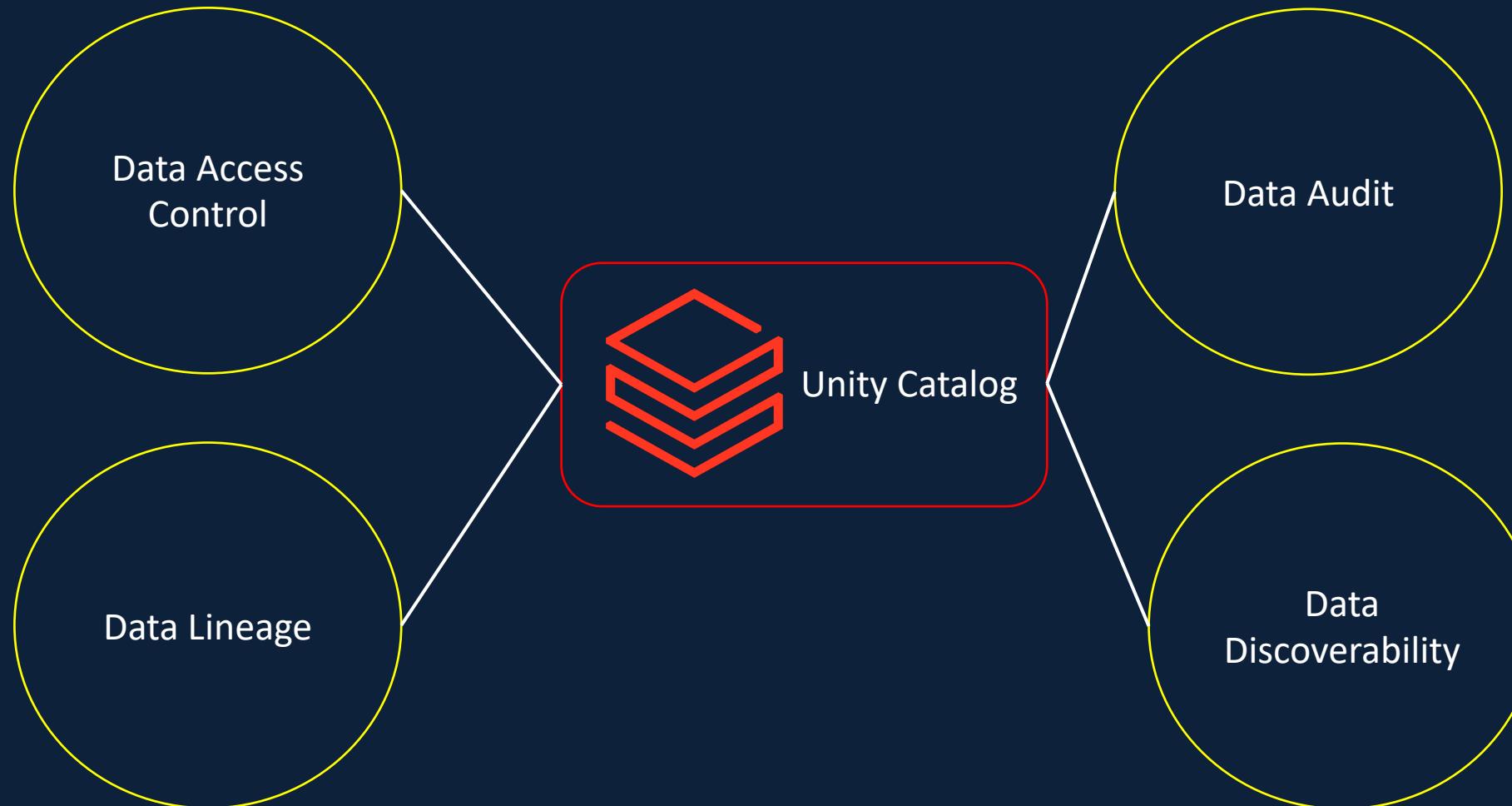
Is the process of managing the availability, usability, integrity and security of the data present in an enterprise

Controls access to the data for the users

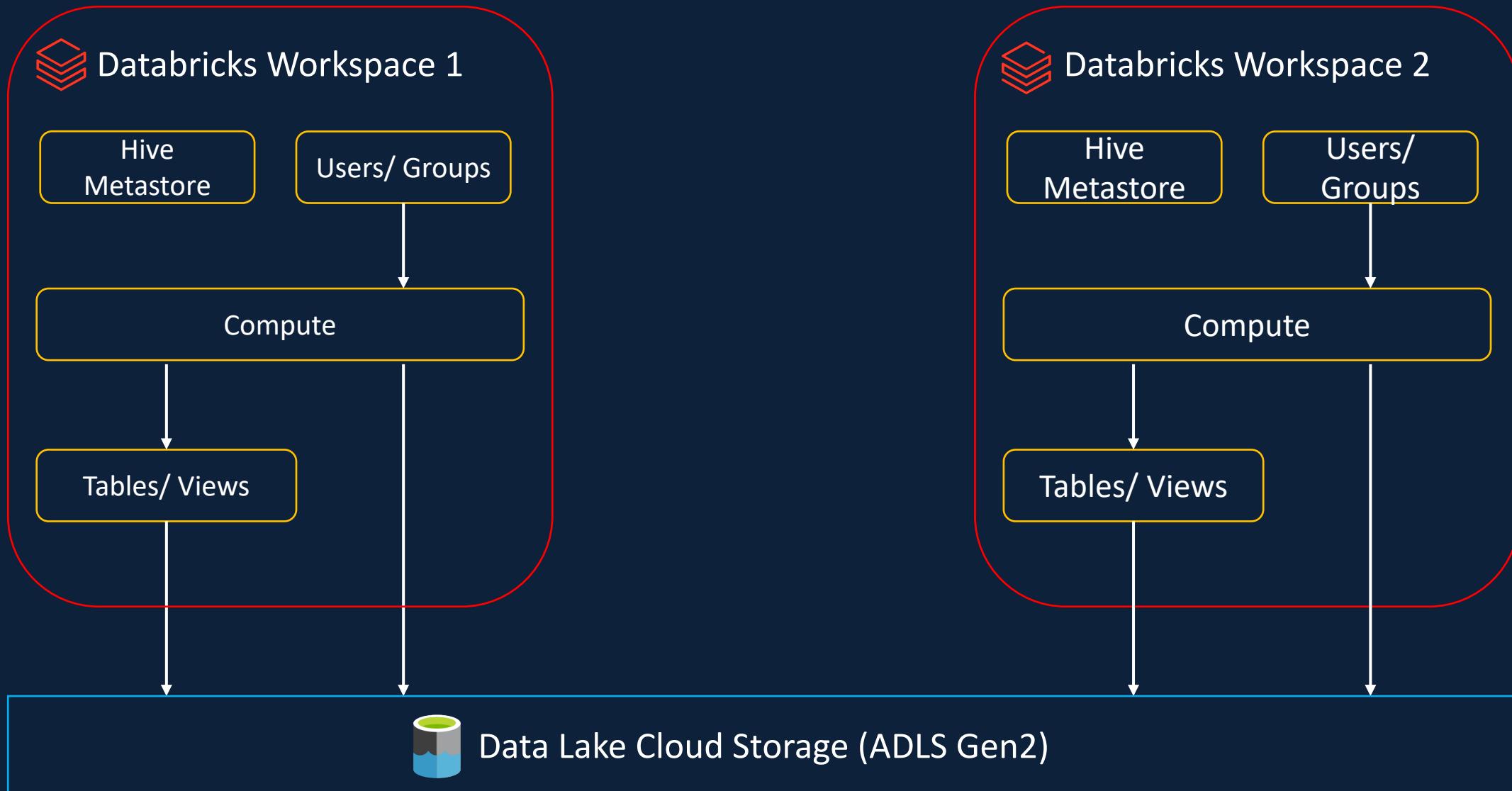
Ensures that the data is trust worthy and not misused

Helps implement privacy regulations such as GDPR, CCPA etc

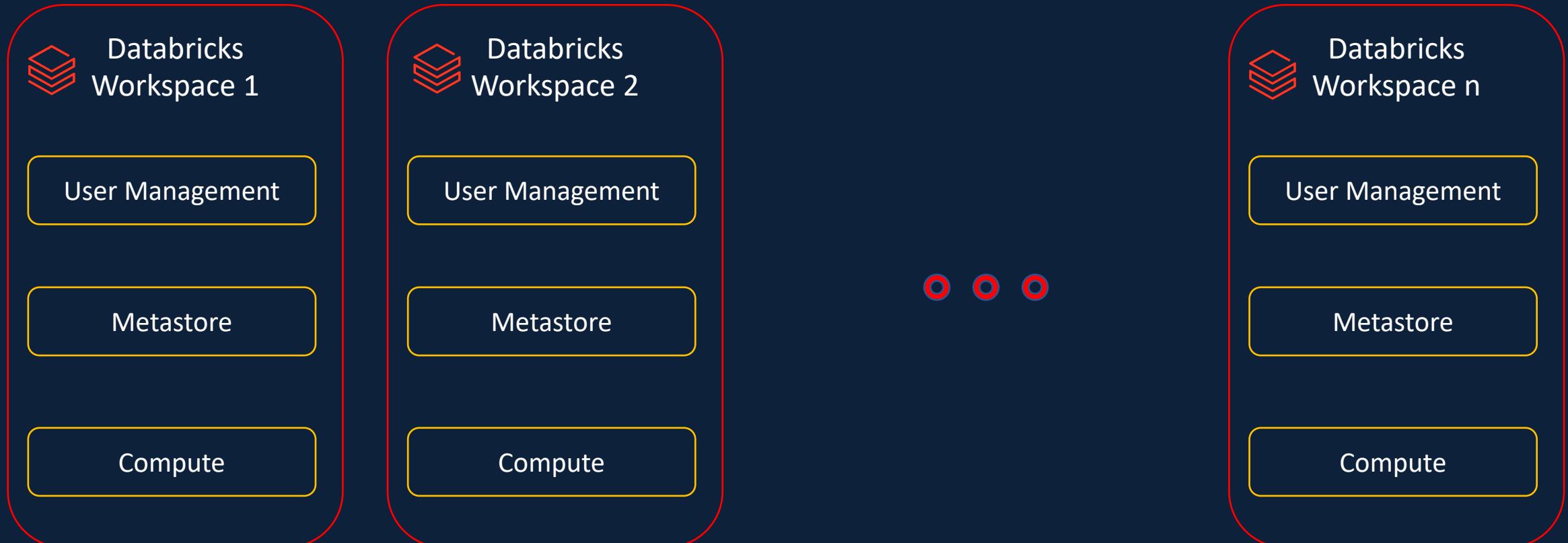
# Data Governance



# Without Unity Catalog



# Without Unity Catalog



# With Unity Catalog

## Without Unity Catalog

Databricks  
Workspace 1

User Management

Metastore

Compute

Databricks  
Workspace 2

User Management

Metastore

Compute

Databricks Account

## With Unity Catalog

Databricks Unity Catalog

User Management

Metastore

Databricks  
Workspace 1

Compute

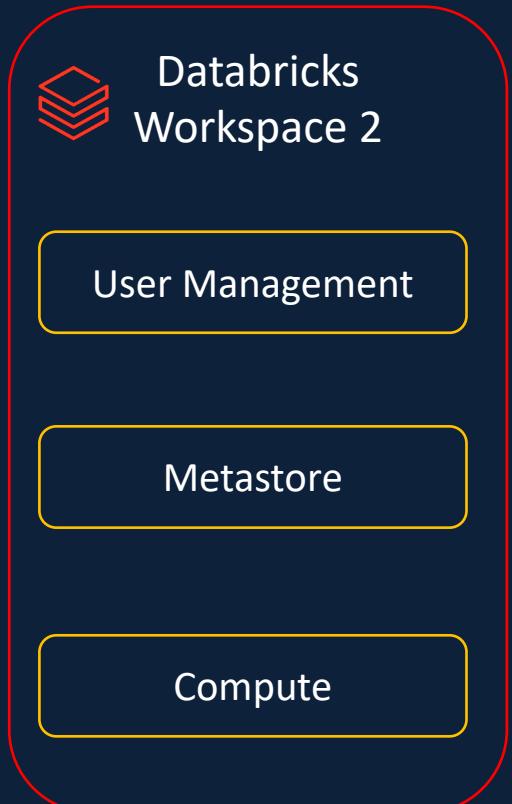
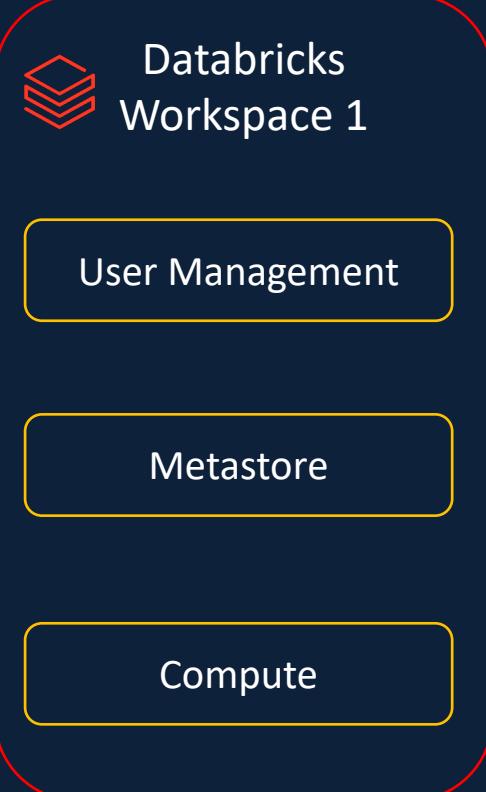
Databricks  
Workspace 2

Compute

Databricks Account

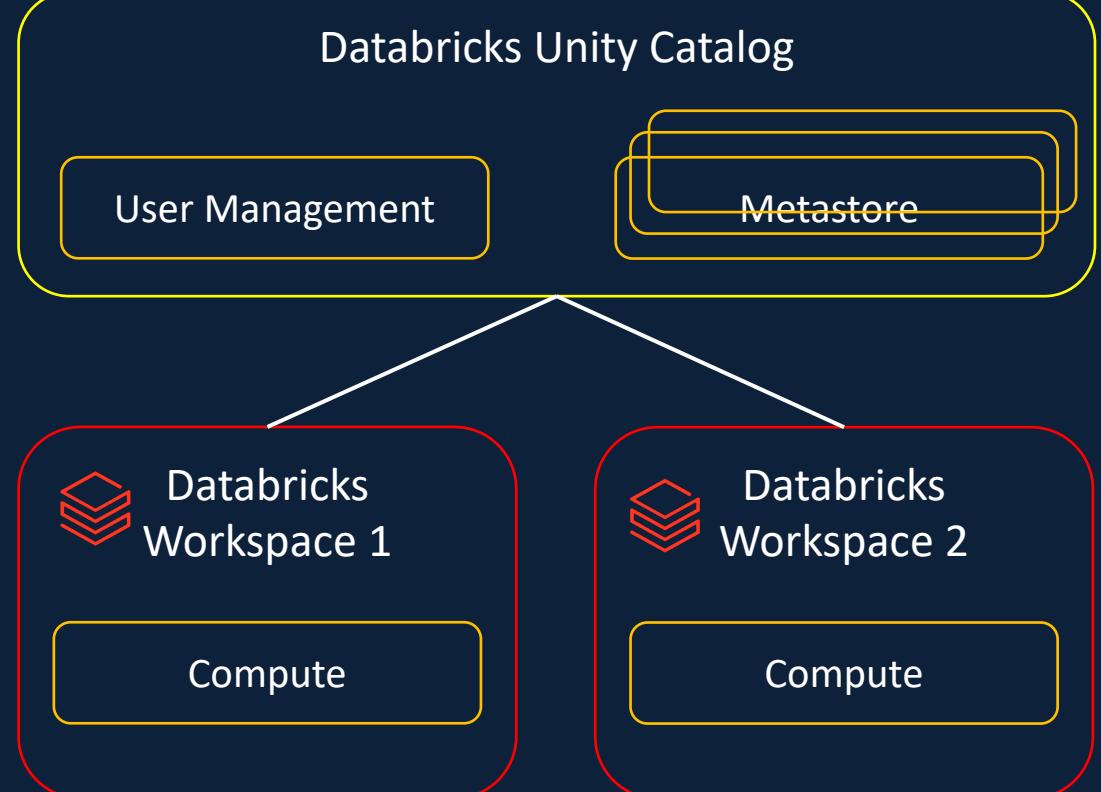
# With Unity Catalog

## Without Unity Catalog



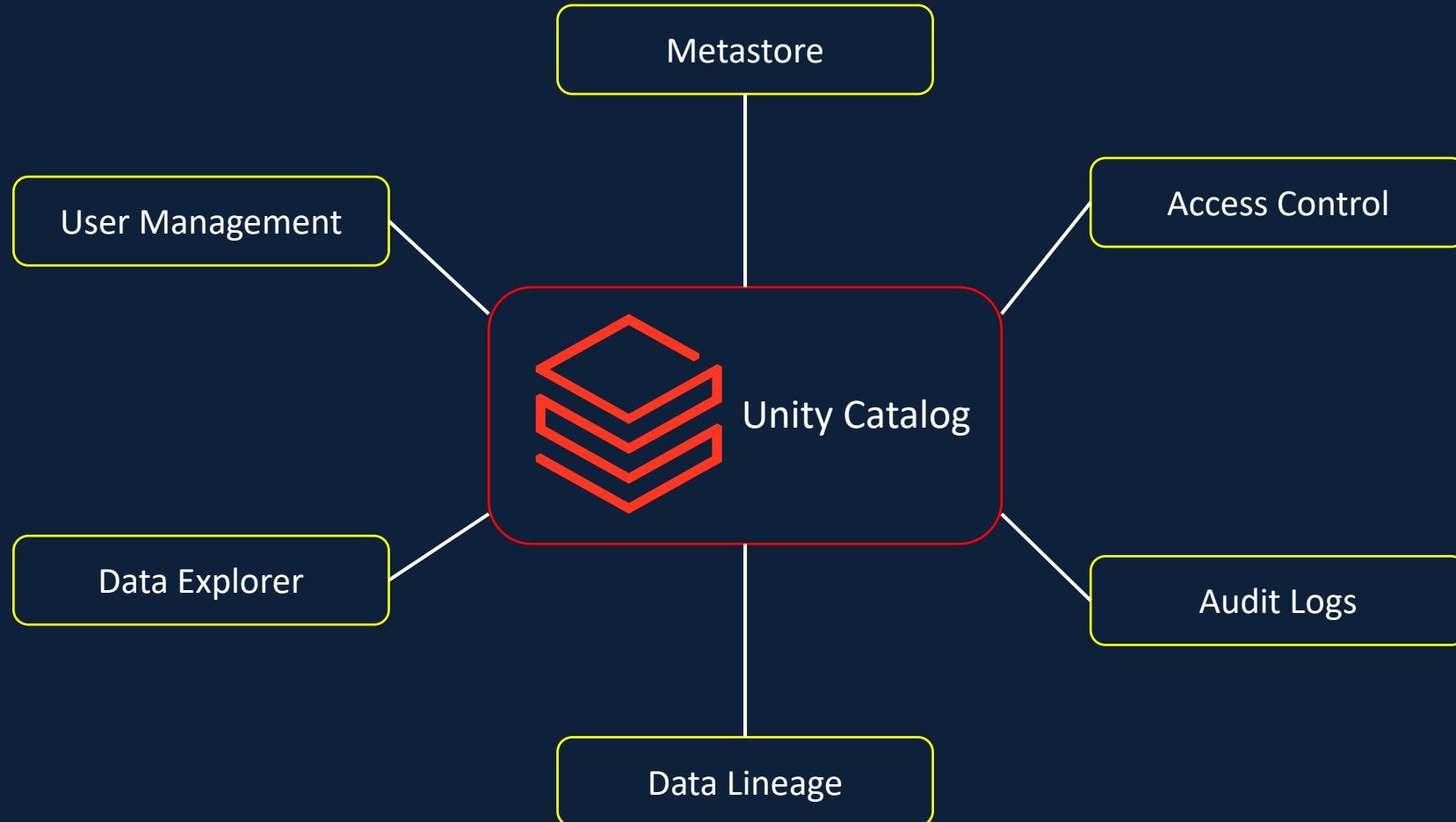
Databricks Account

## With Unity Catalog

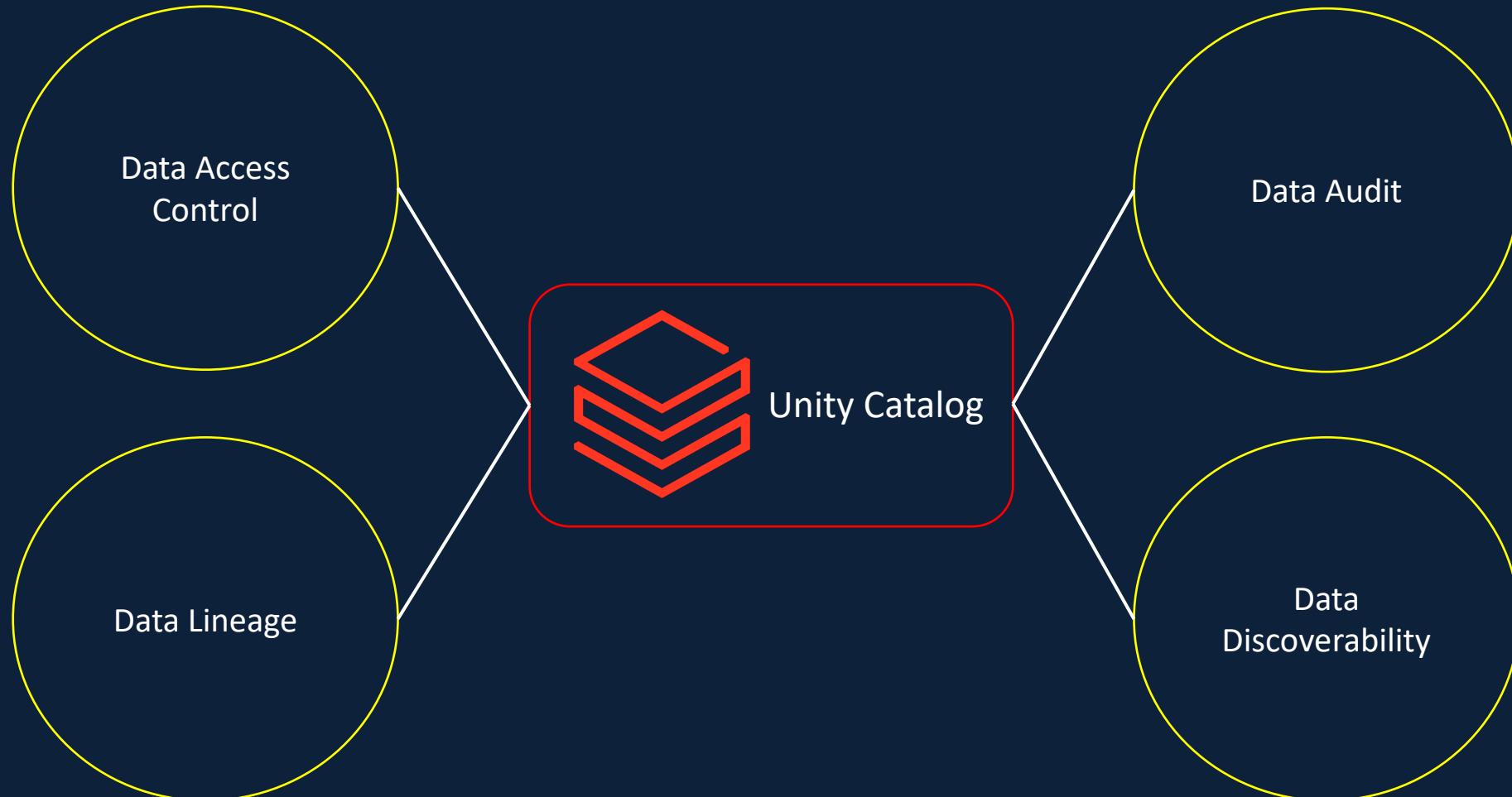


Databricks Account

# Unity Catalog

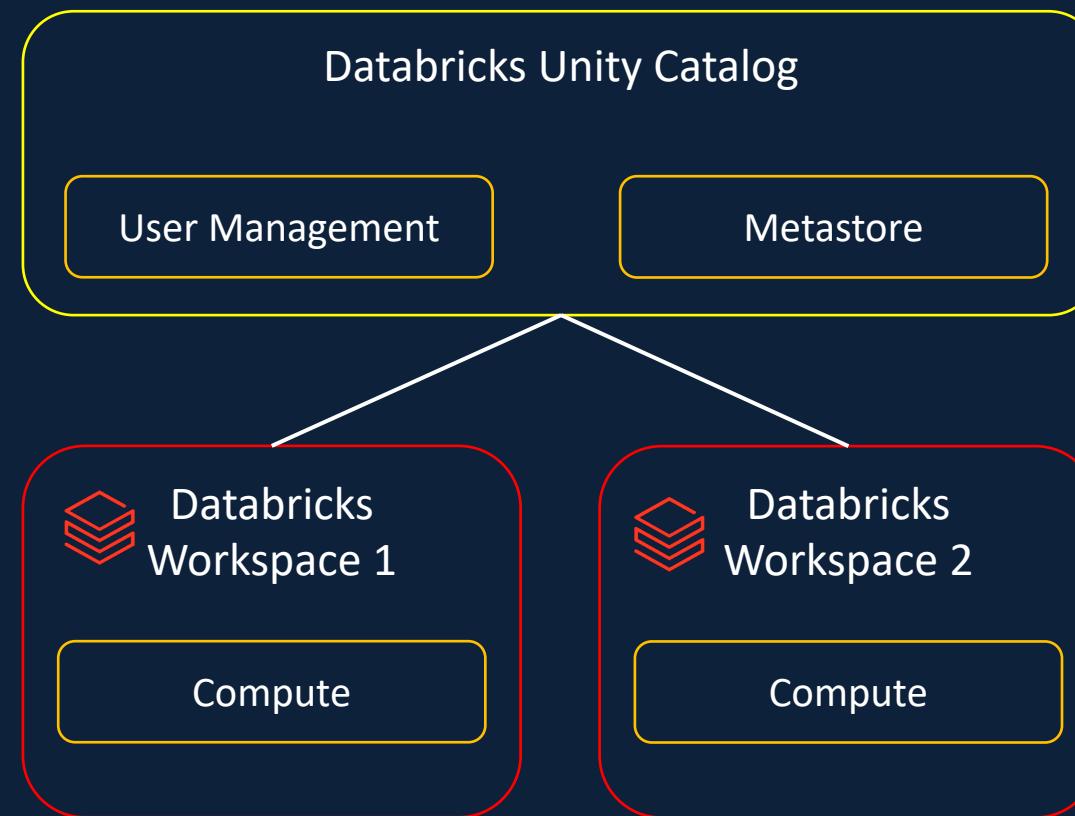


# Unity Catalog

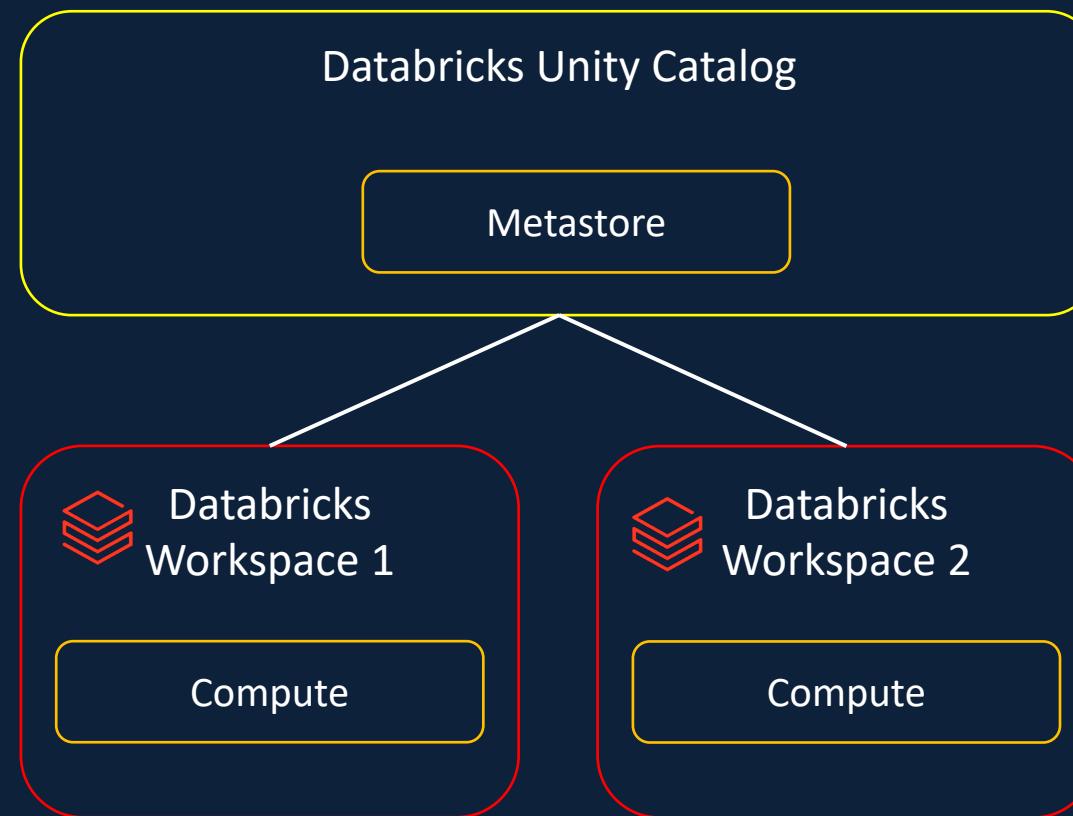


Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

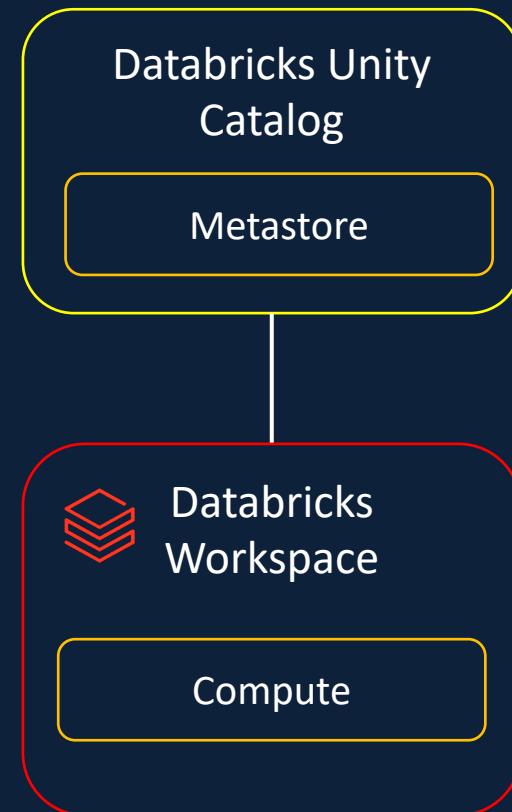
# Unity Catalog Set-up



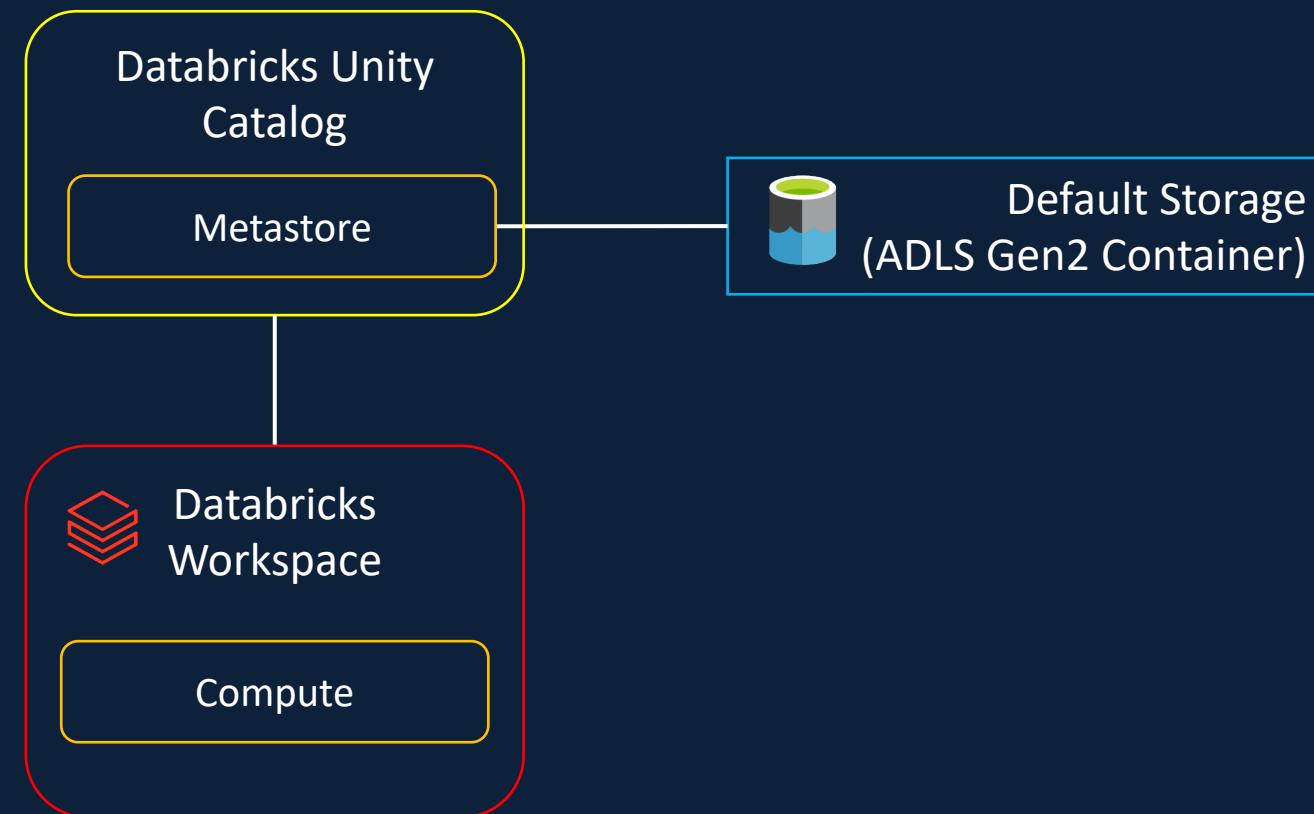
# Unity Catalog Set-up



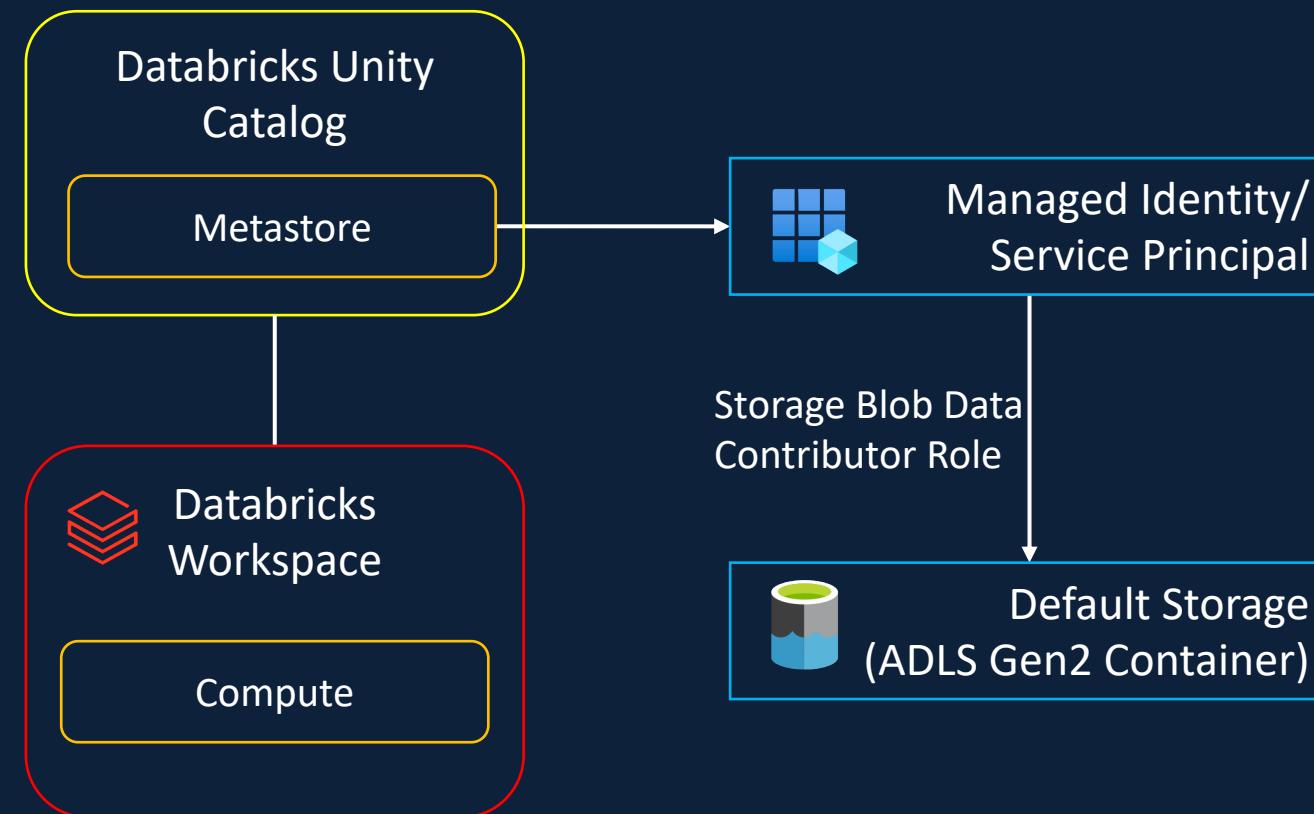
# Unity Catalog Set-up



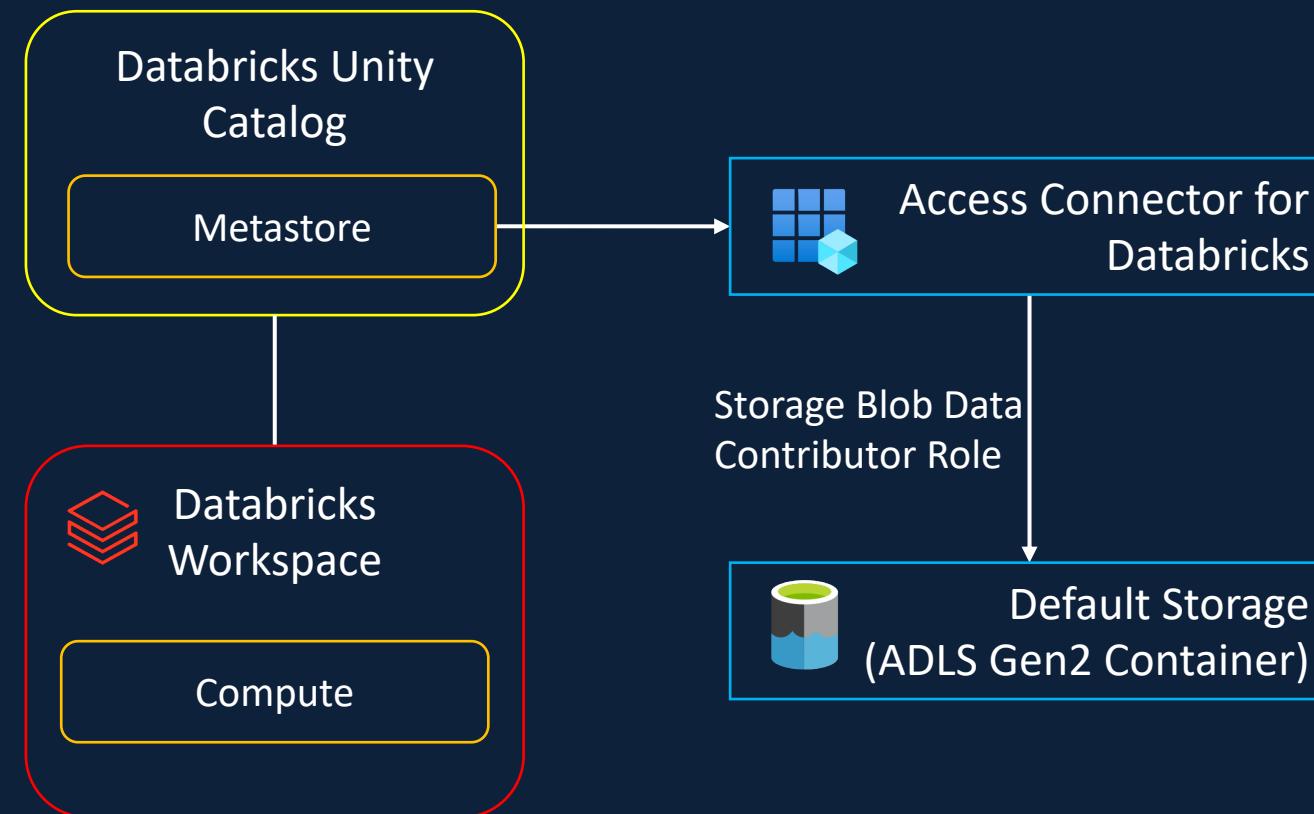
# Unity Catalog Set-up



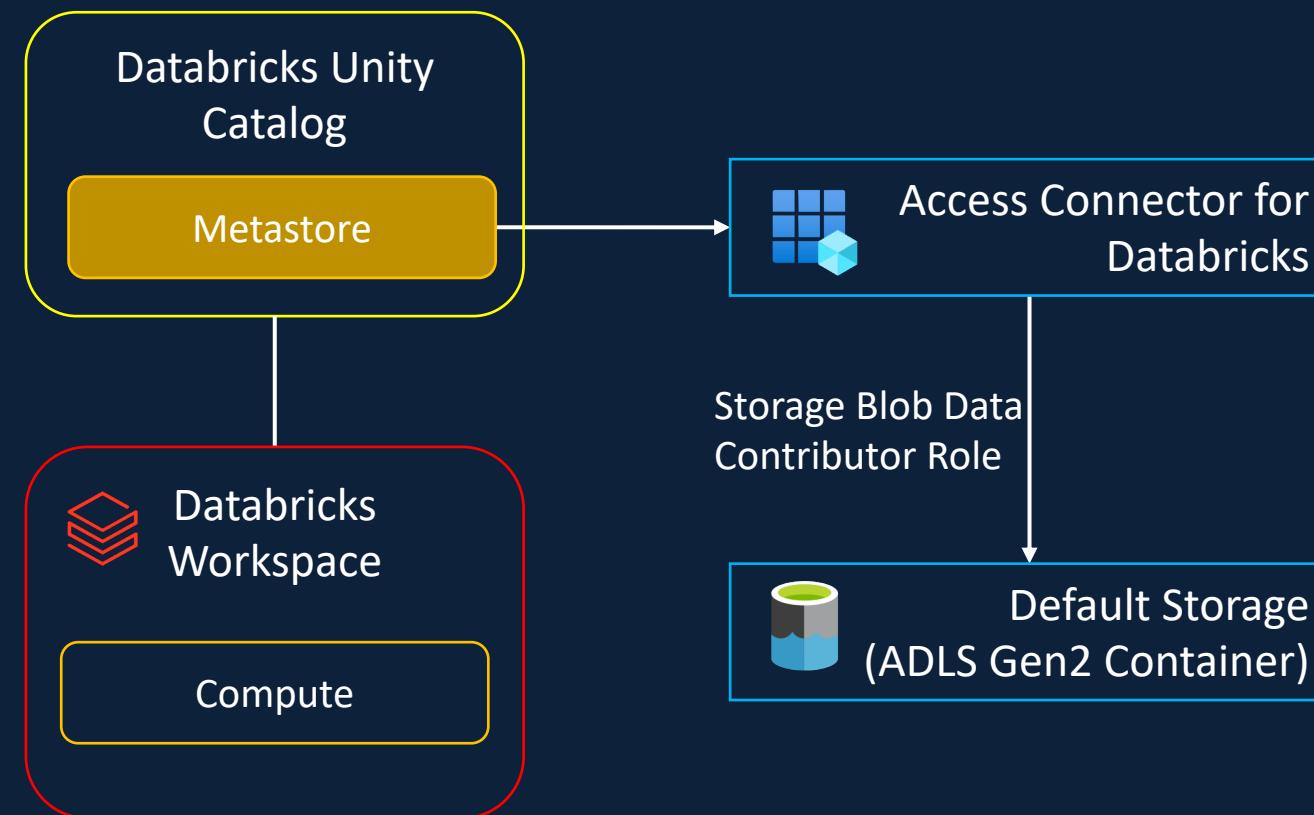
# Unity Catalog Set-up



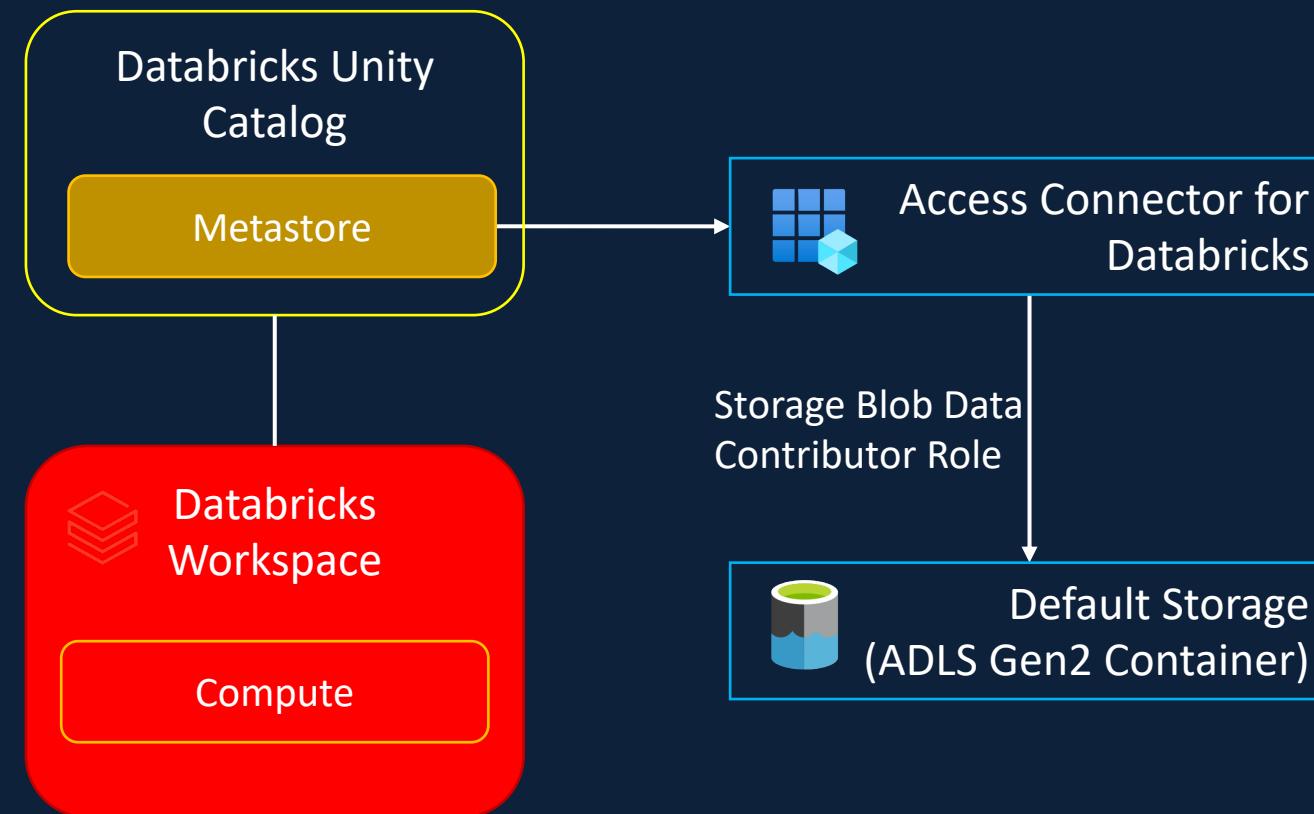
# Unity Catalog Set-up



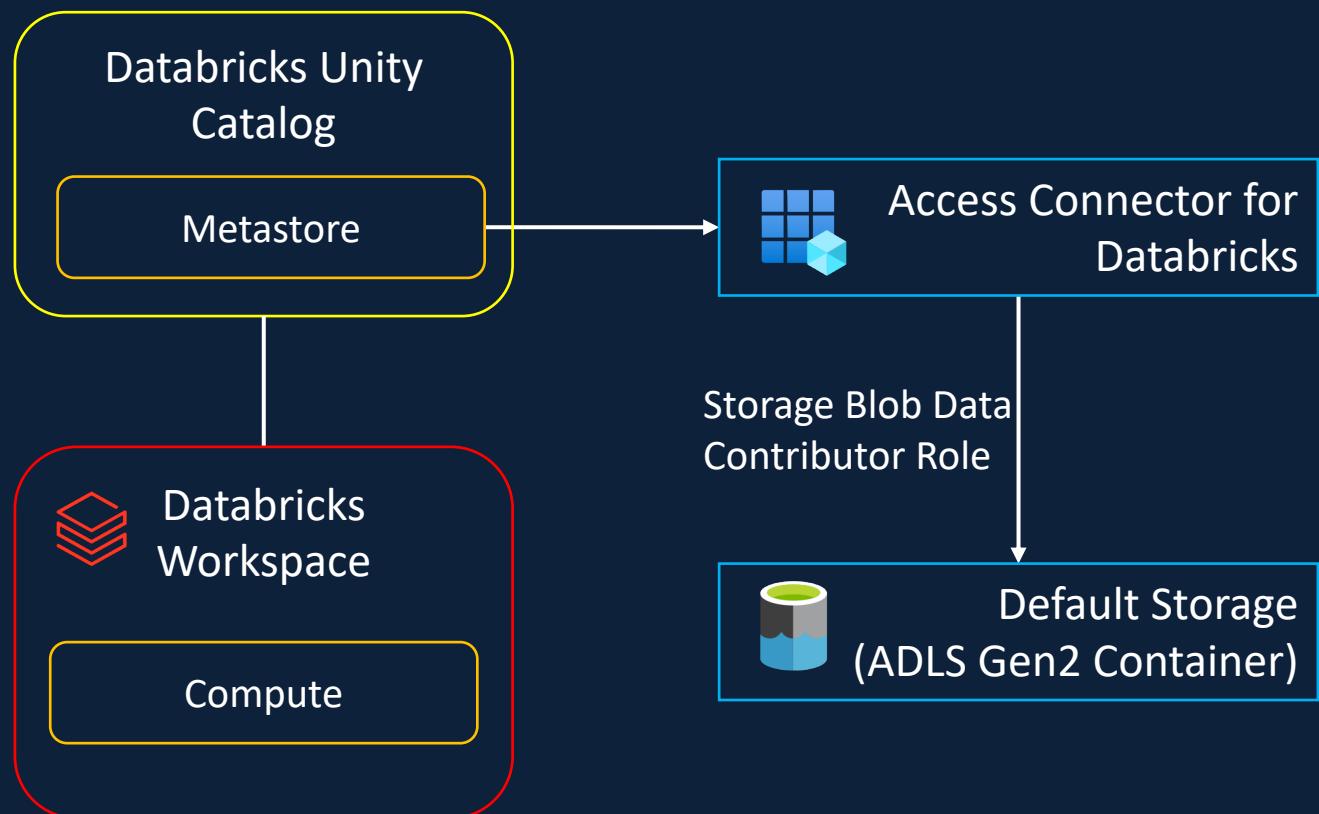
# Unity Catalog Set-up



# Unity Catalog Set-up



# Unity Catalog Set-up



Create Azure Databricks Workspace

Create Azure Data Lake Gen2

Create Access Connector

Add role Storage Blob Data Contributor

Create Unit Catalog Metastore

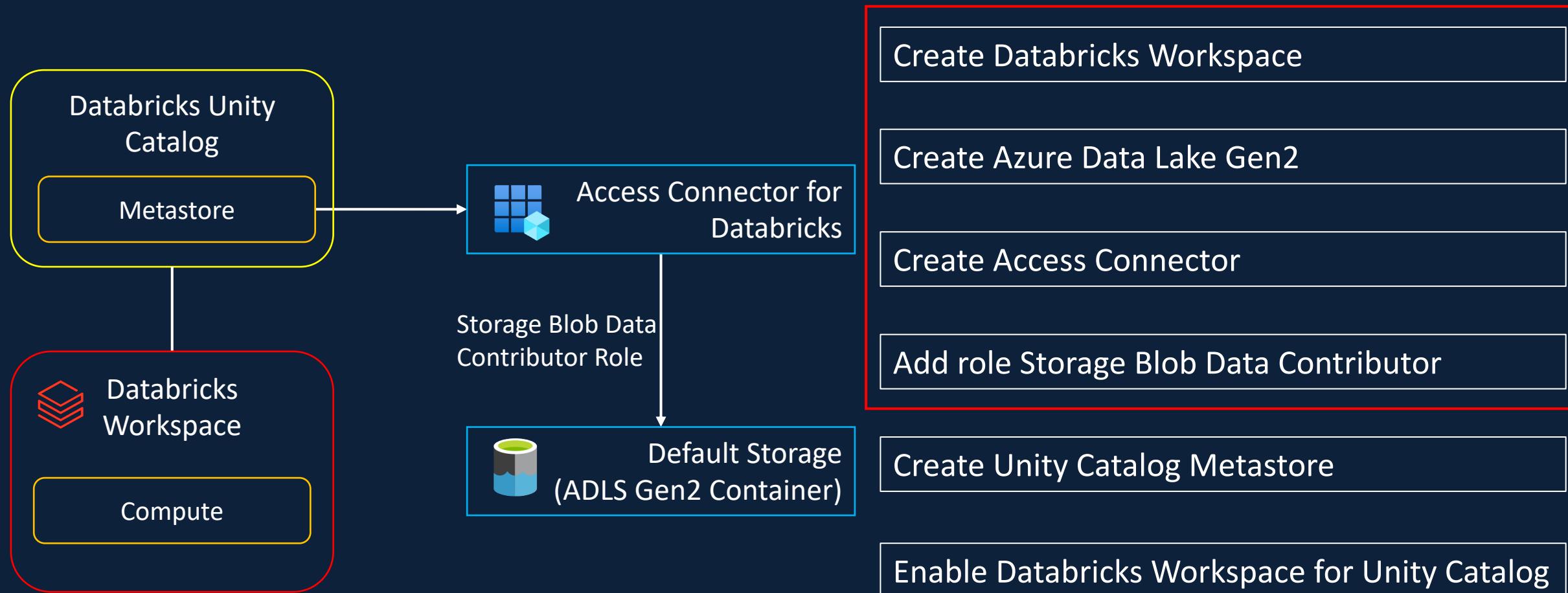
Enable Databricks Workspace for Unity Catalog

# Unity Catalog Set-up

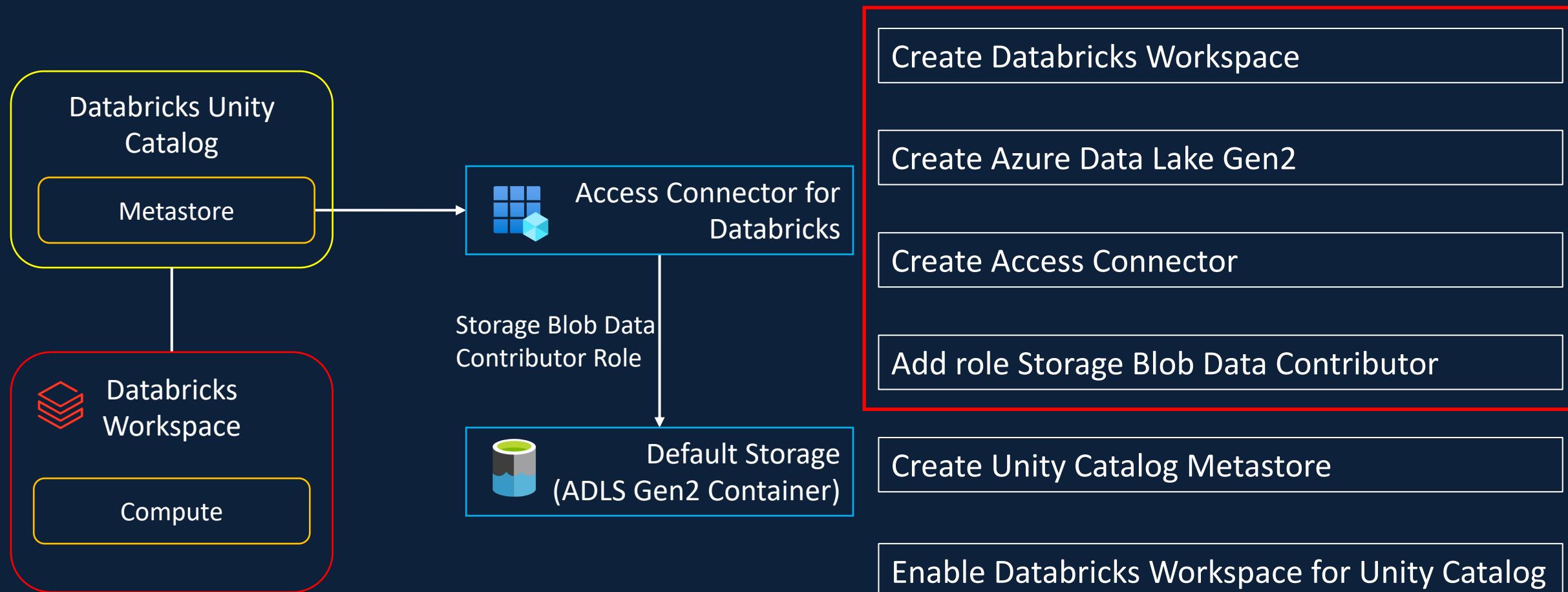
## Pre-requisites



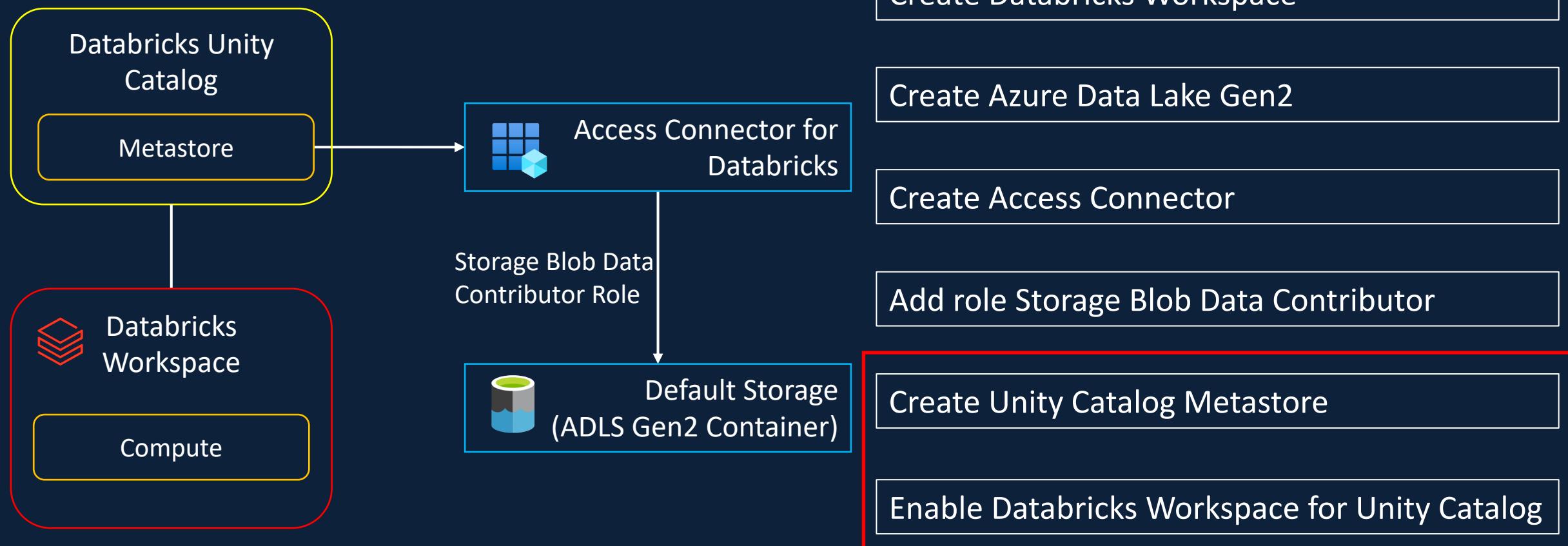
# Unity Catalog Set-up



# Unity Catalog Set-up



# Unity Catalog Set-up



# Cluster Configuration



# Unity Catalog Object Model



# Unity Catalog Object Model

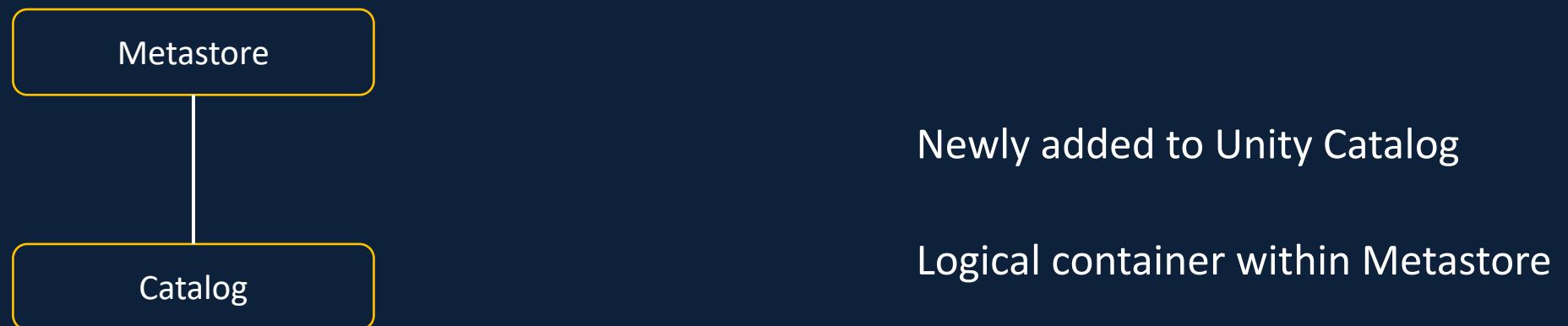
Metastore

Top level container

Only one per region

Paired with Default ADLS Storage

# Unity Catalog Object Model



# Unity Catalog Object Model

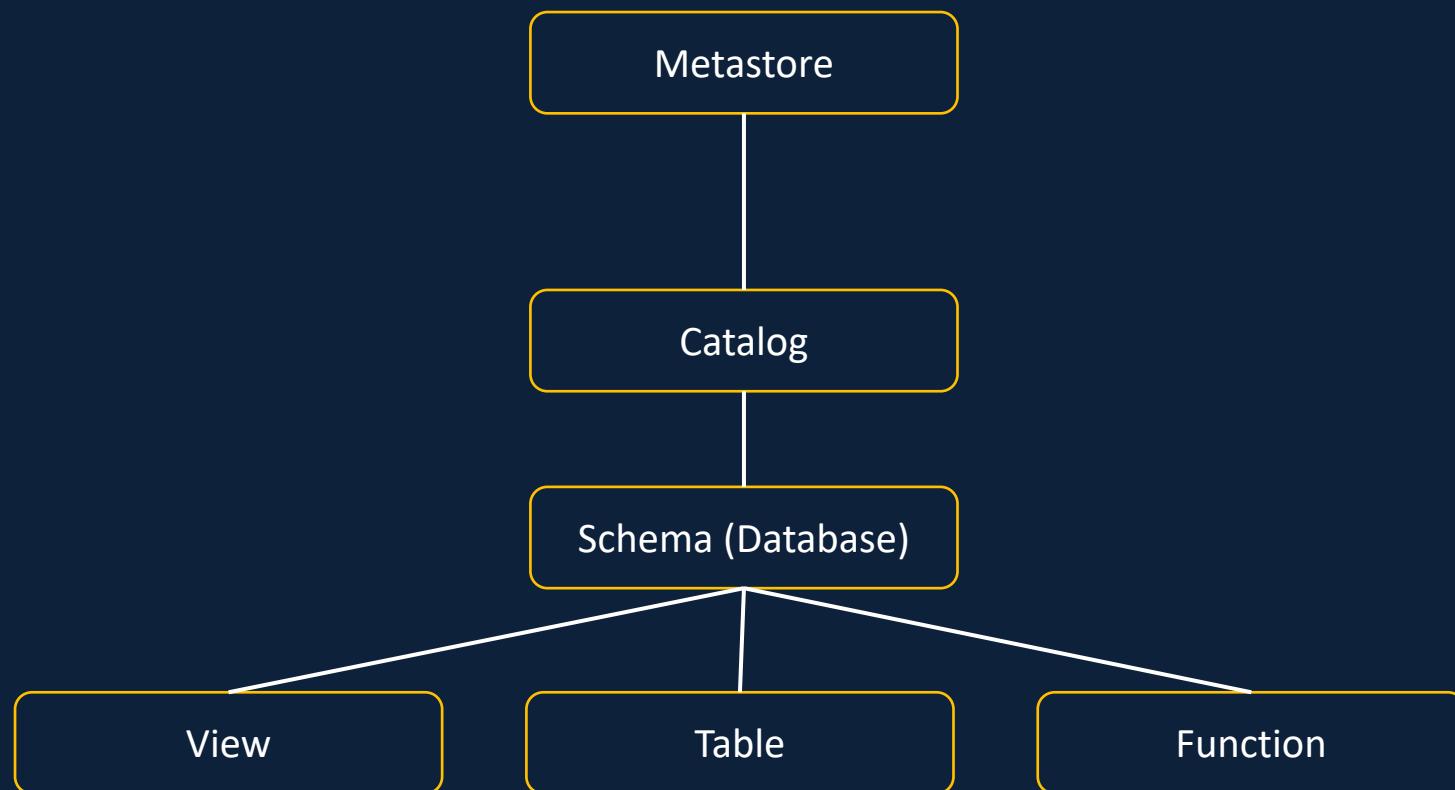


Next level container within Catalogs

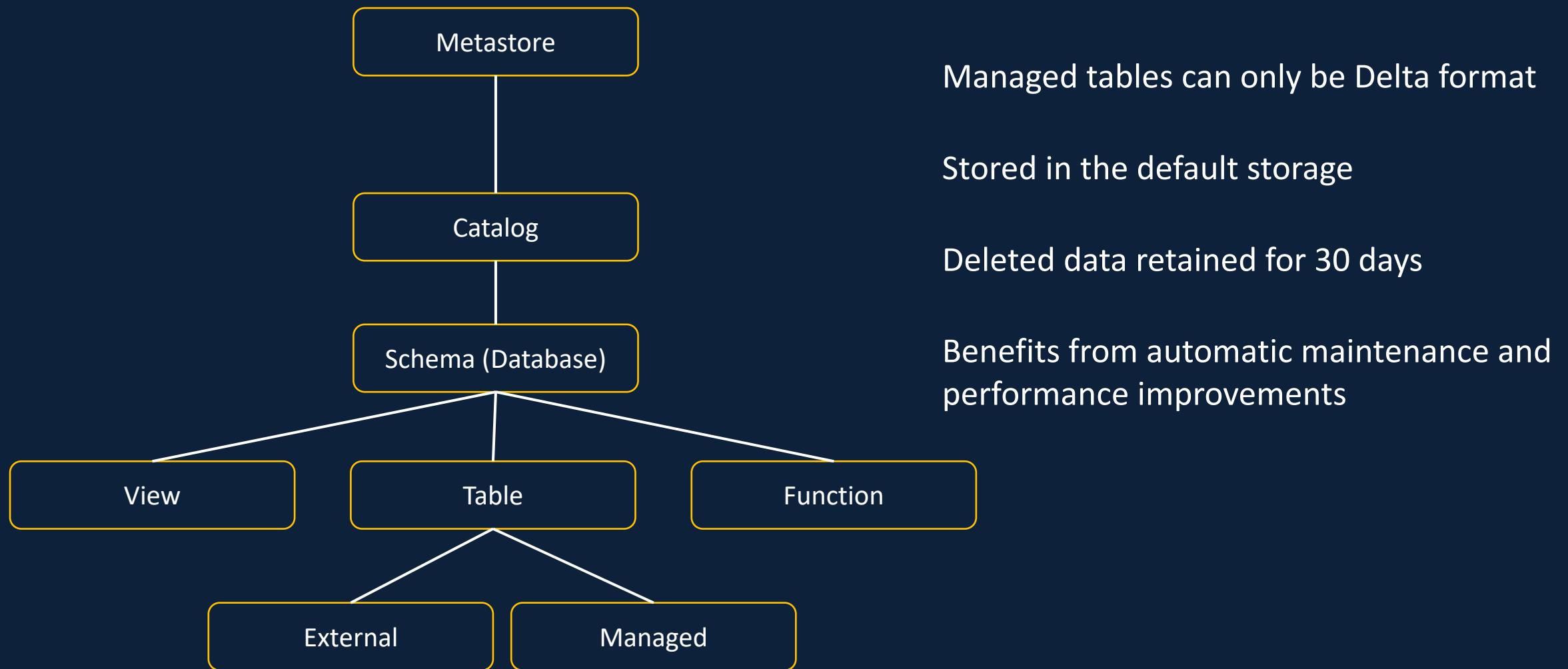
Schemas and Databases are the same

Use Schema instead of Database

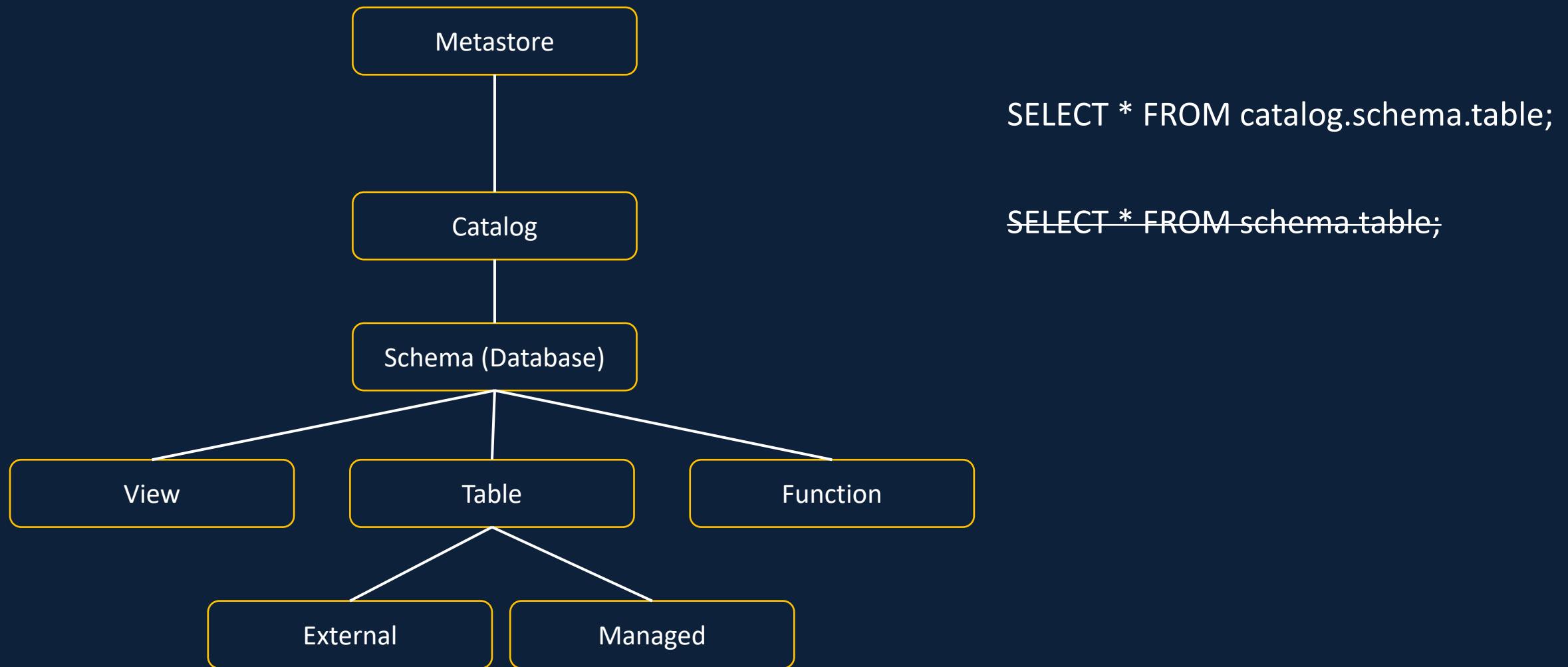
# Unity Catalog Object Model



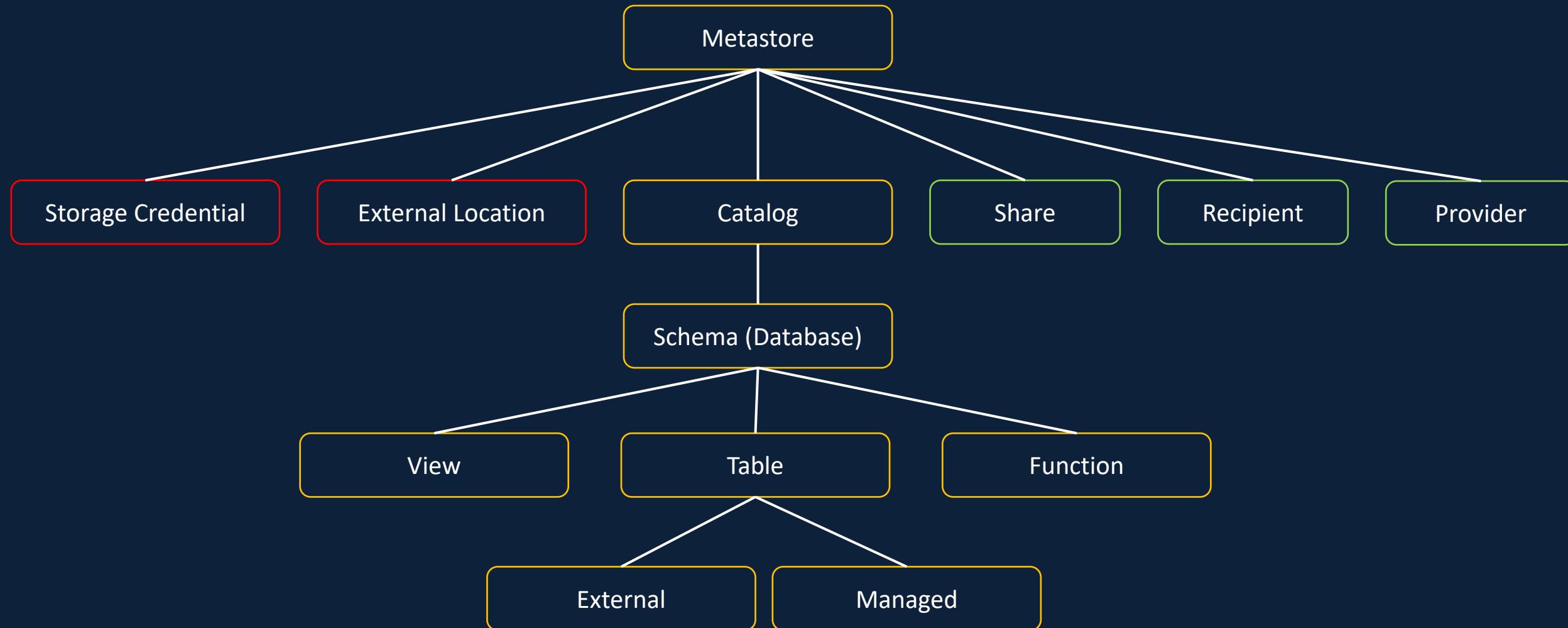
# Unity Catalog Object Model



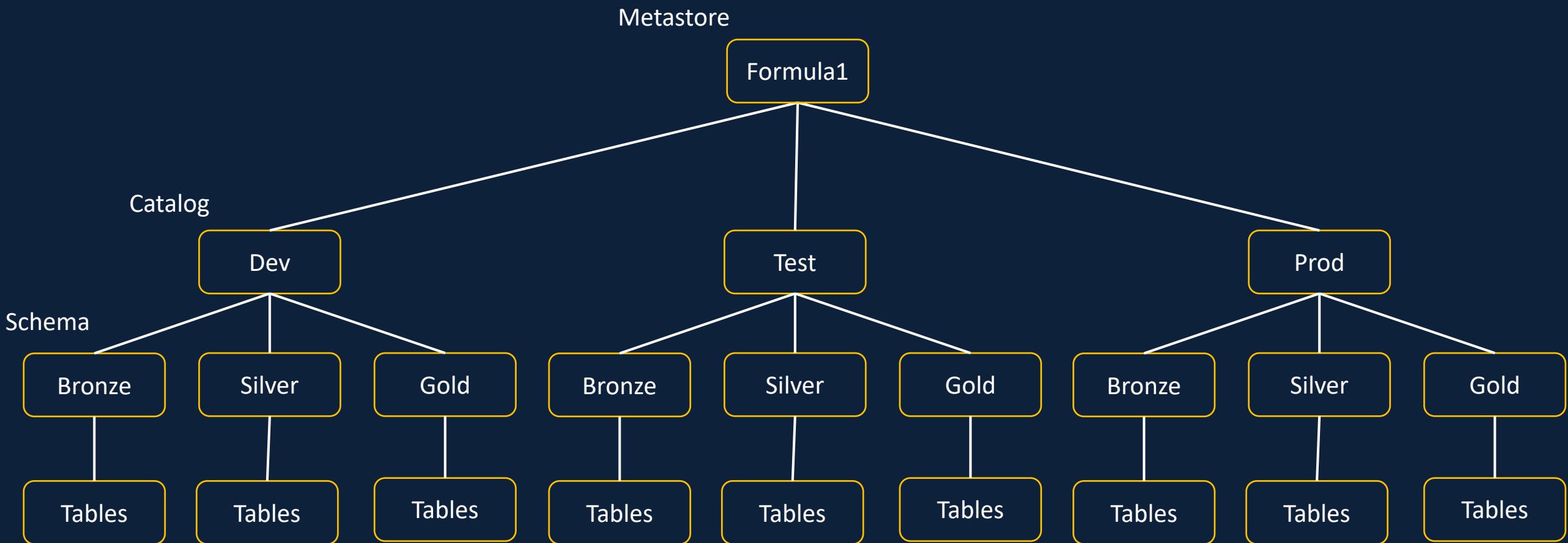
# Unity Catalog Object Model



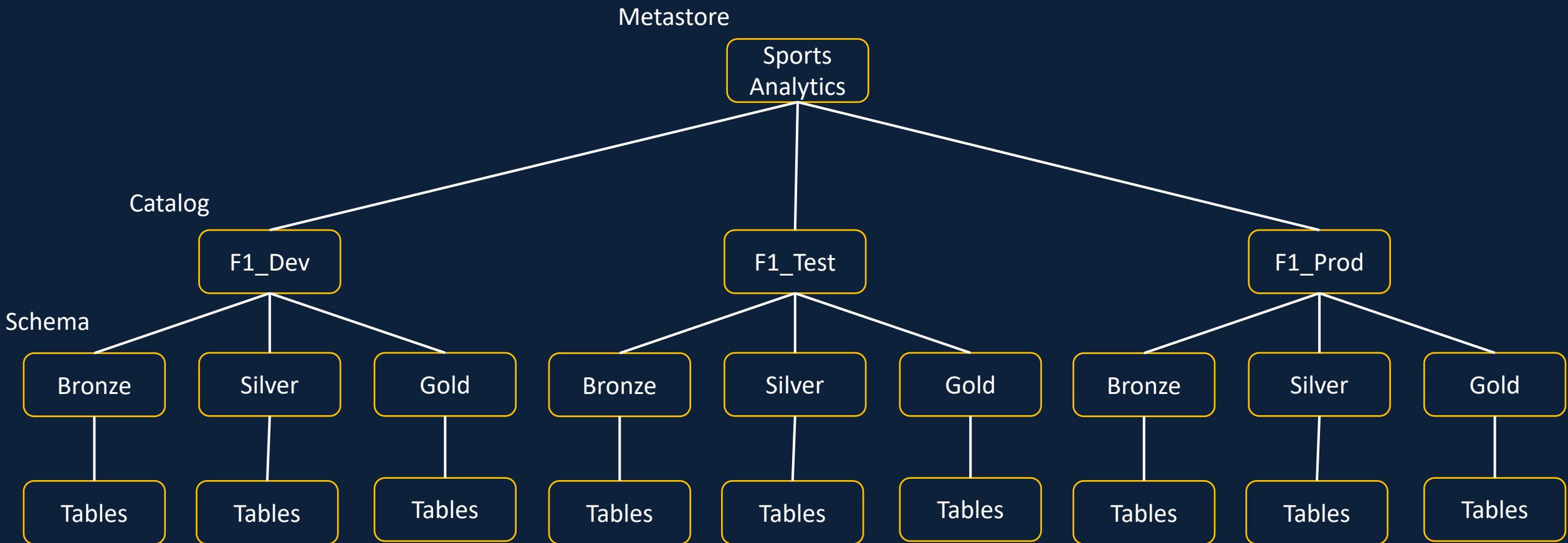
# Unity Catalog Object Model



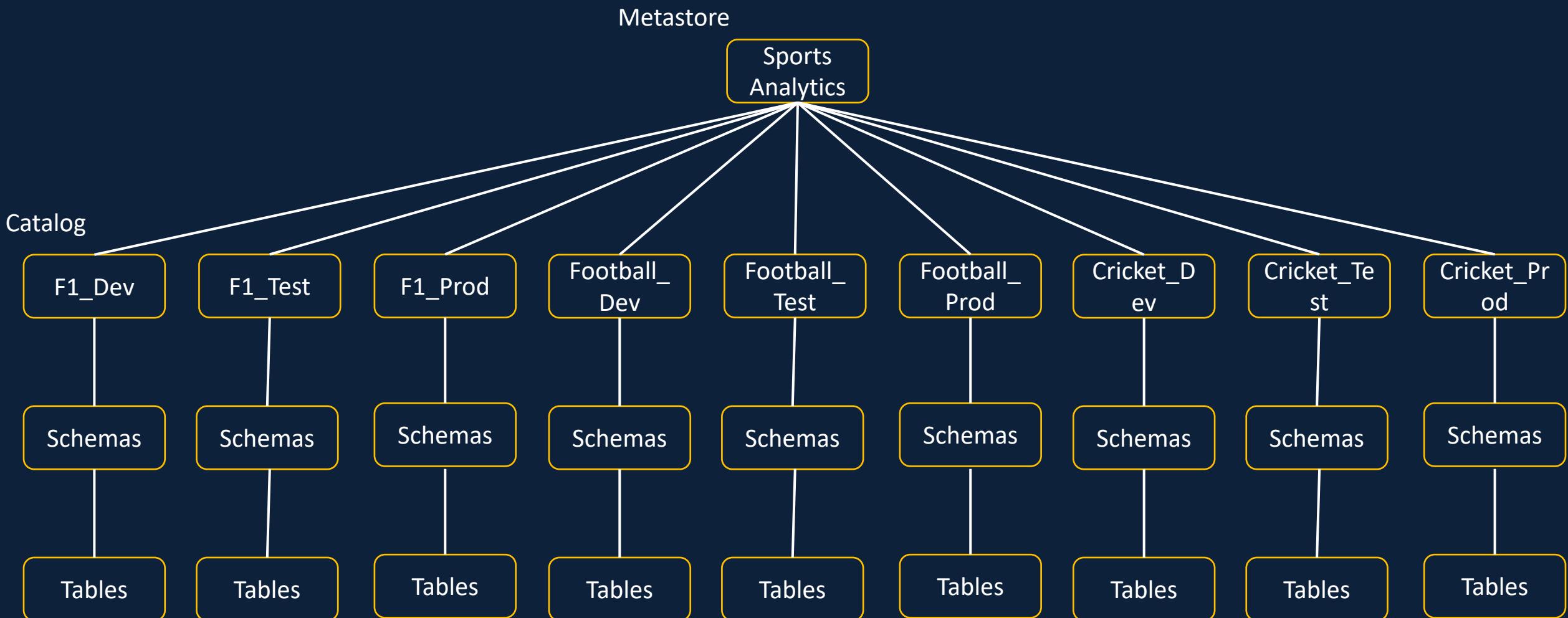
# Unity Catalog Object Model



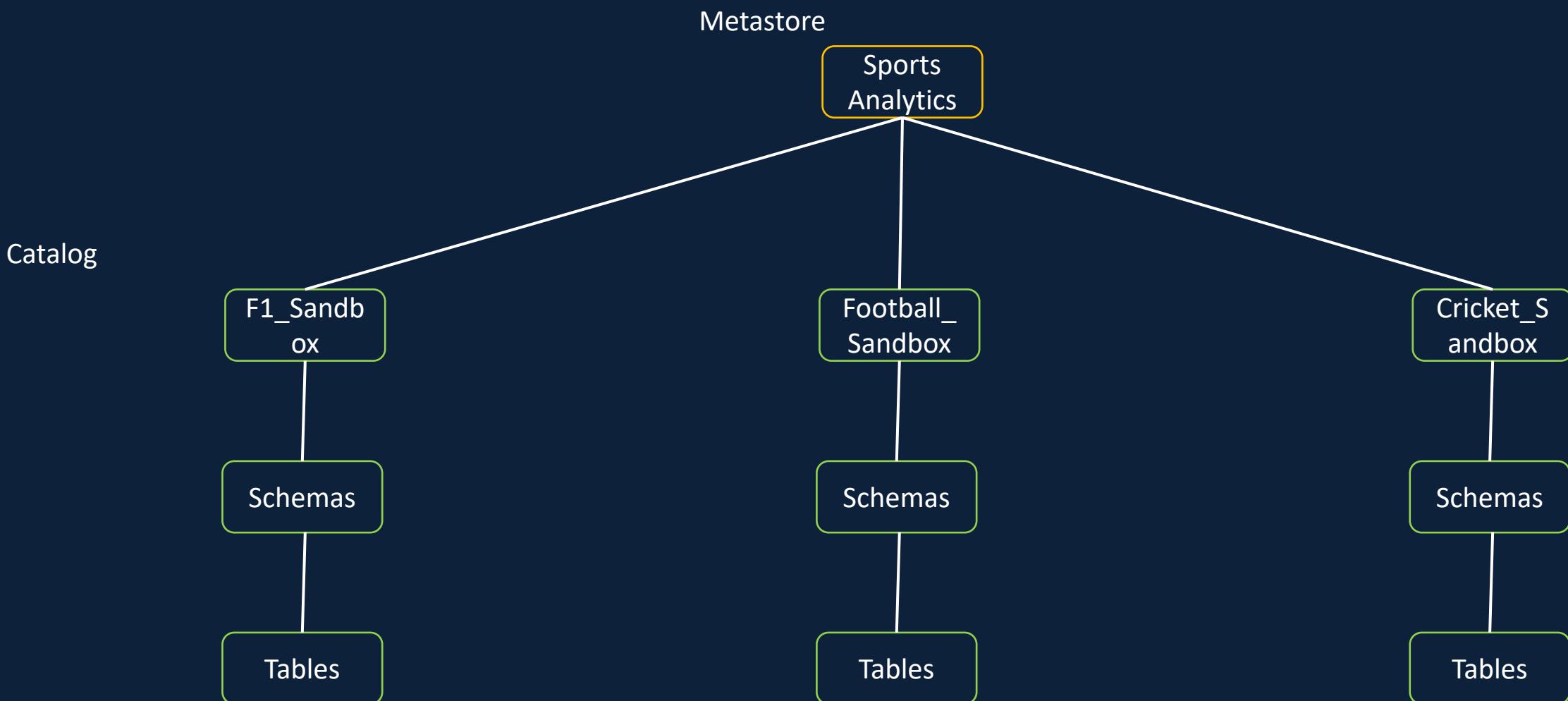
# Unity Catalog Object Model



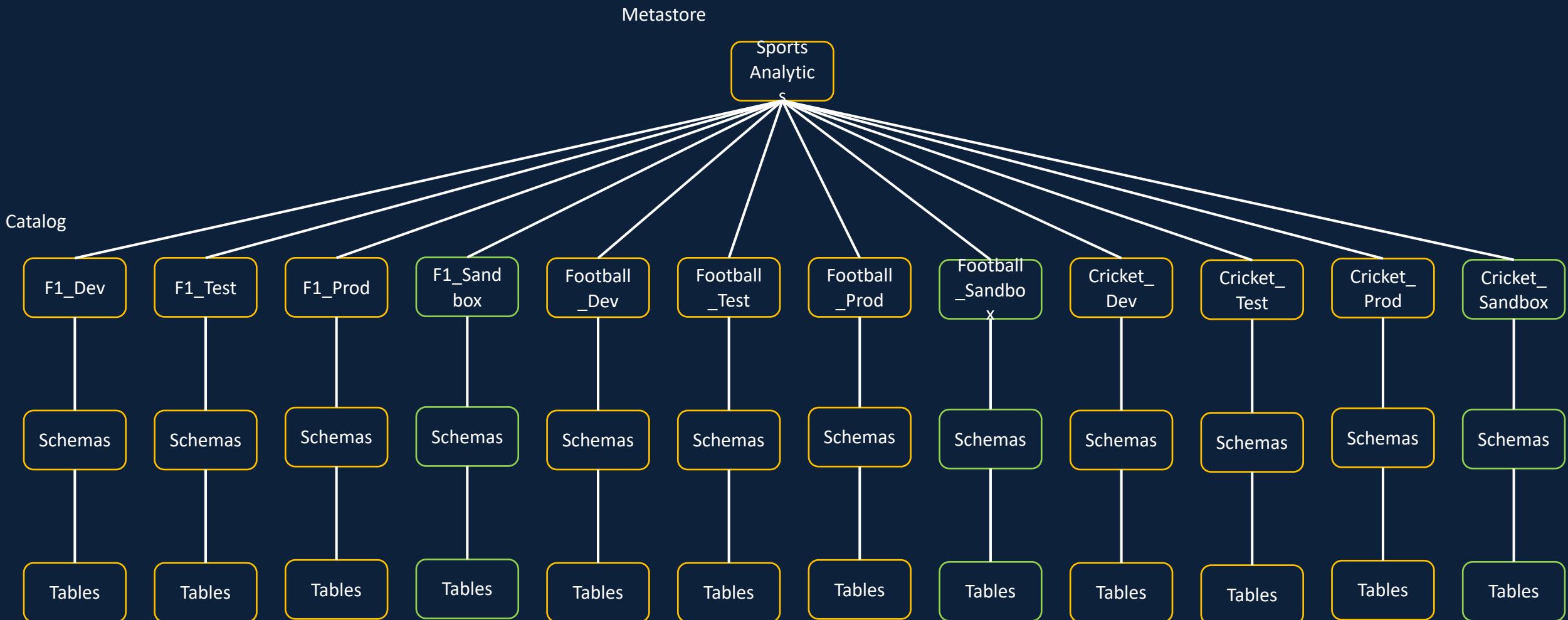
# Unity Catalog Object Model



# Unity Catalog Object Model



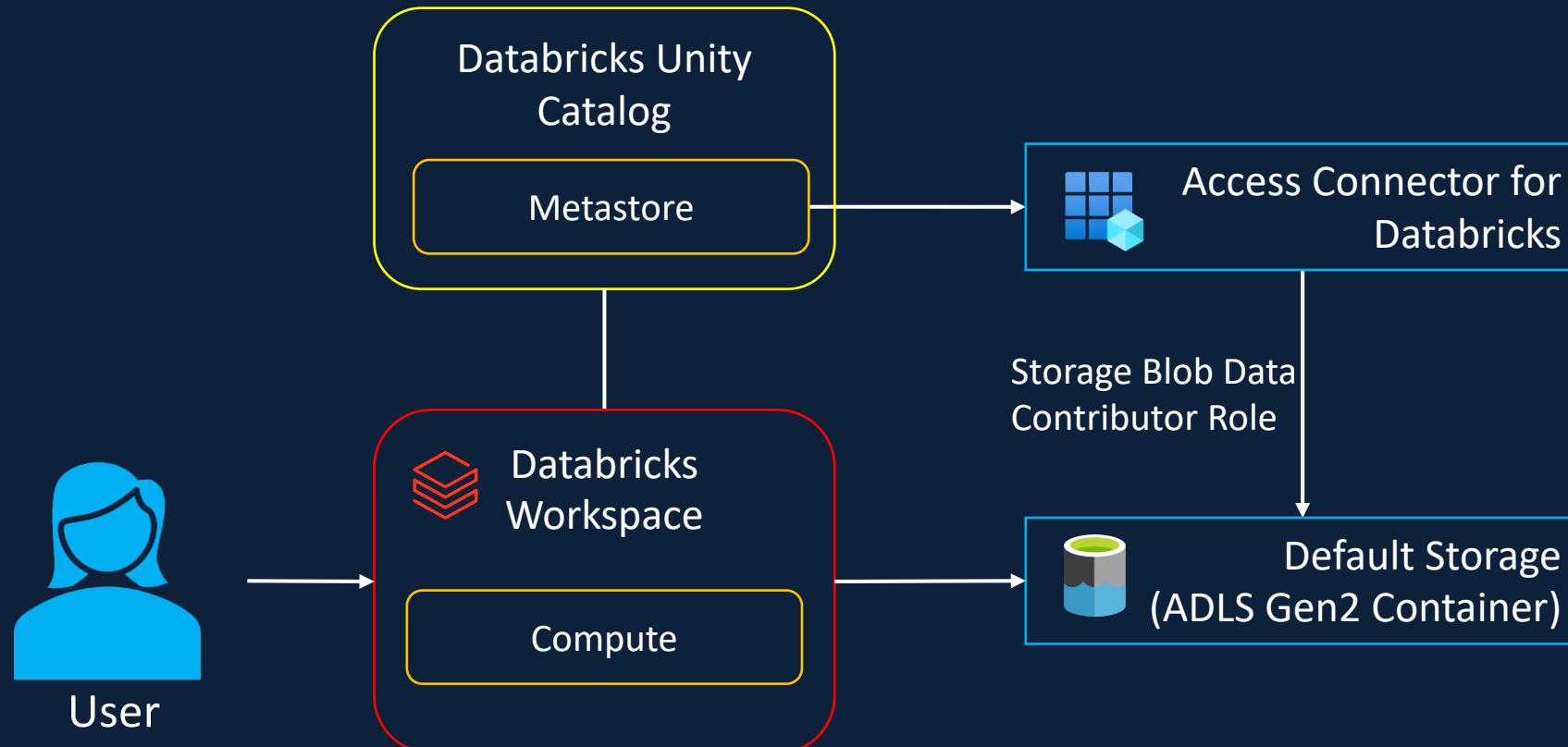
# Unity Catalog Object Model



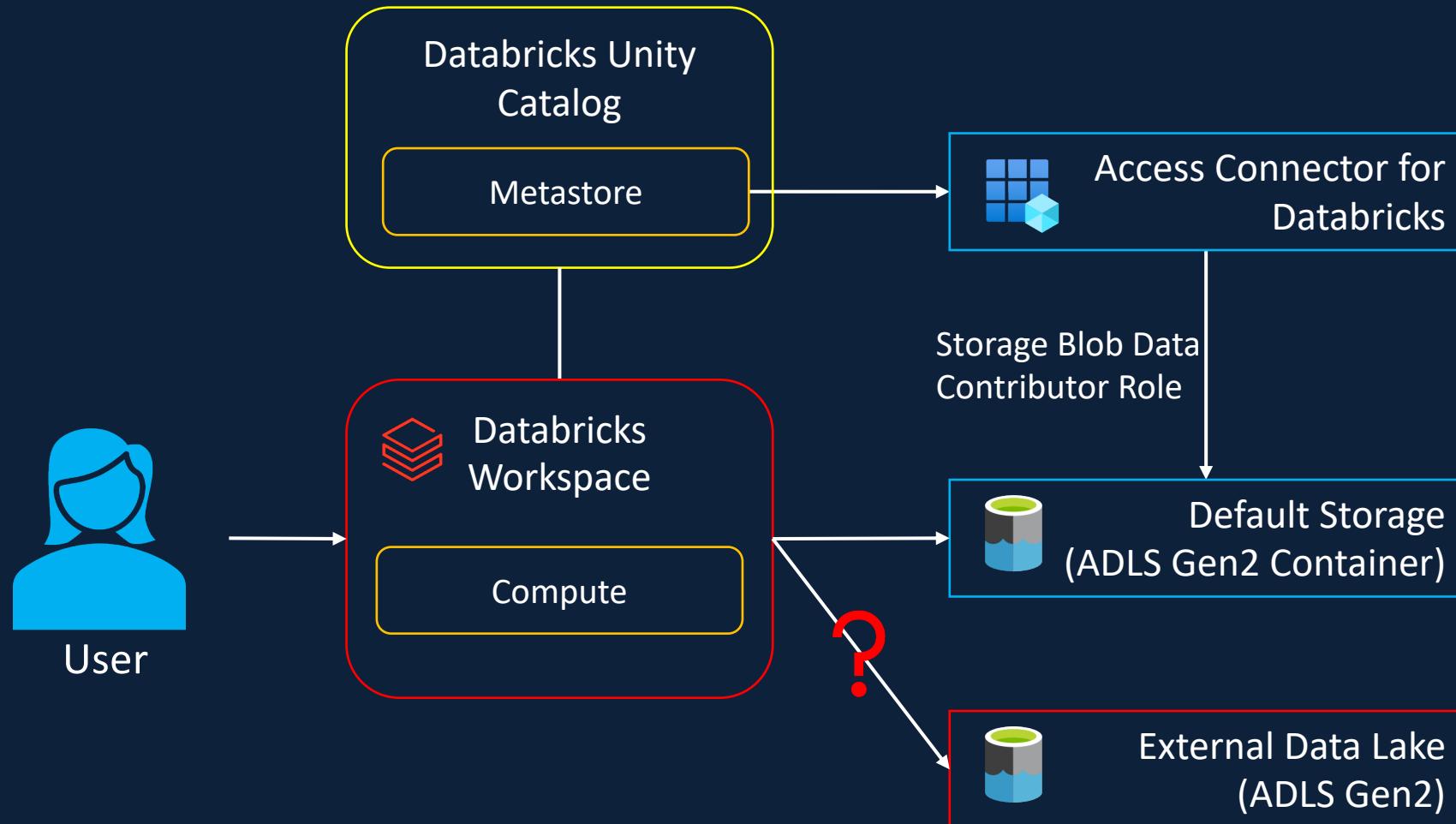
# Accessing External Locations



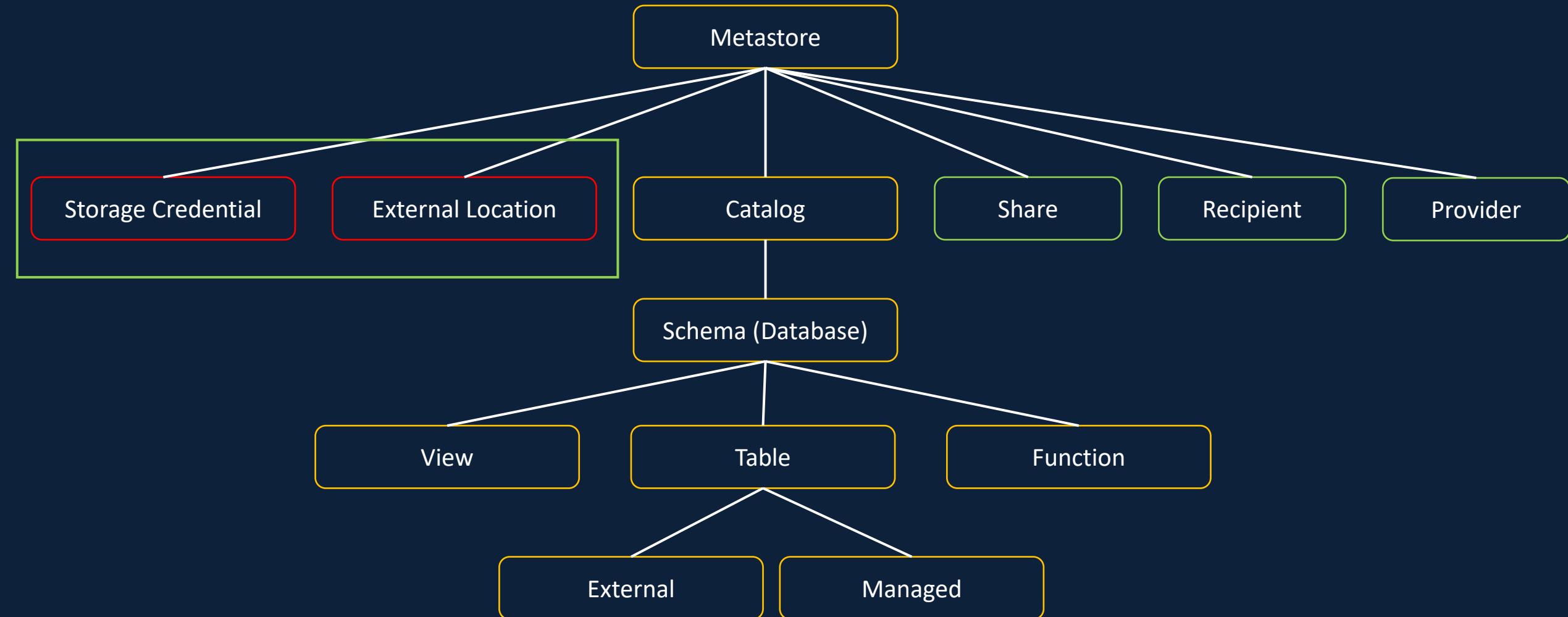
# Unity Catalog Configuration



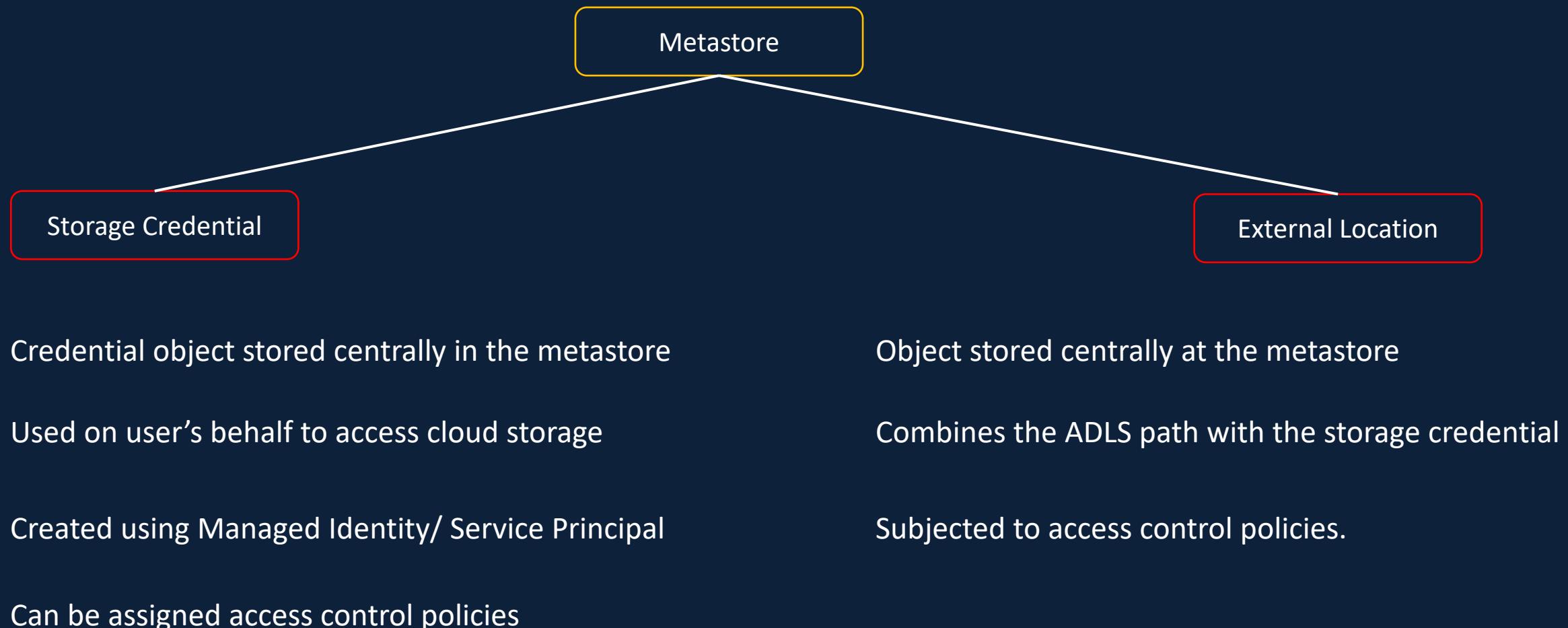
# Accessing External Locations



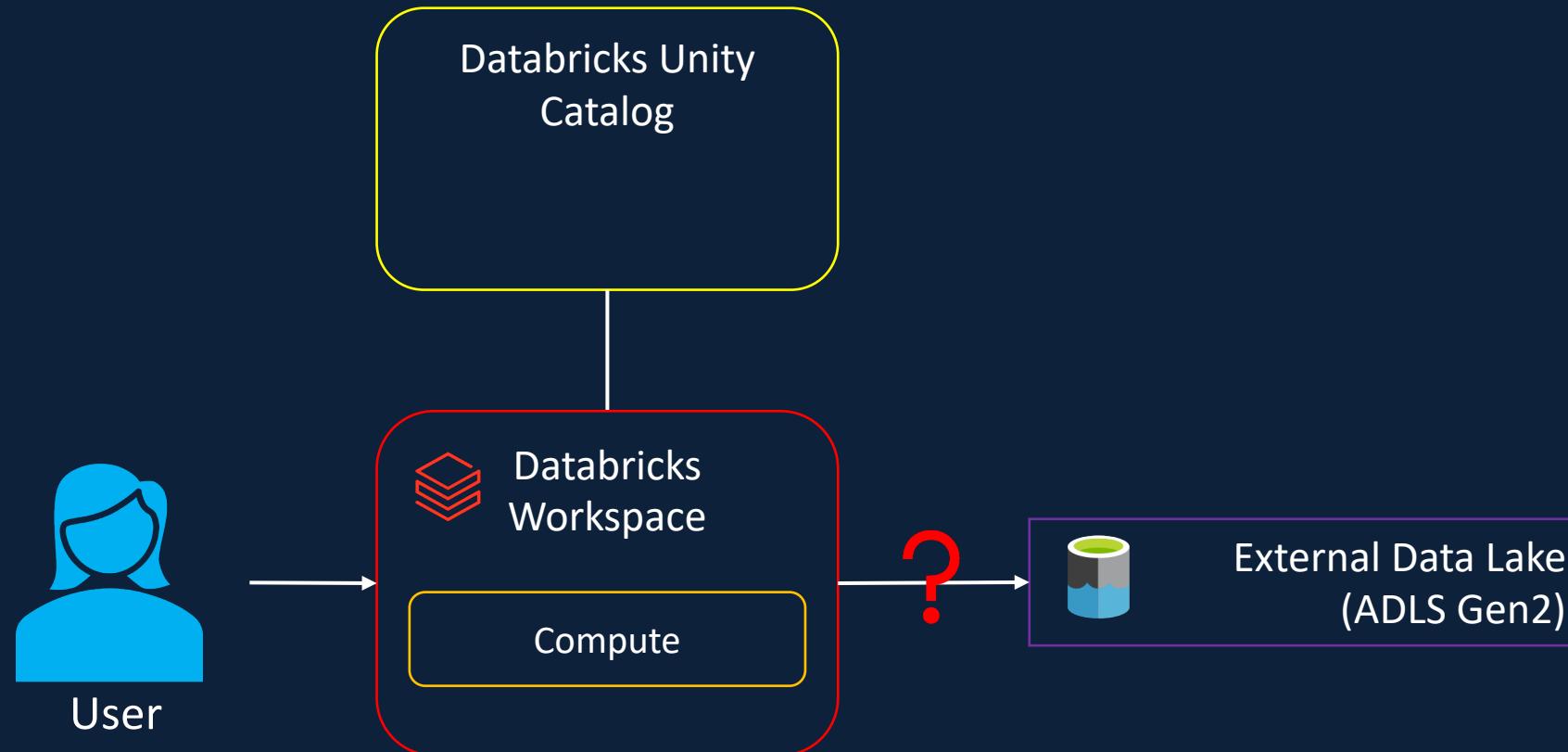
# Unity Catalog Object Model



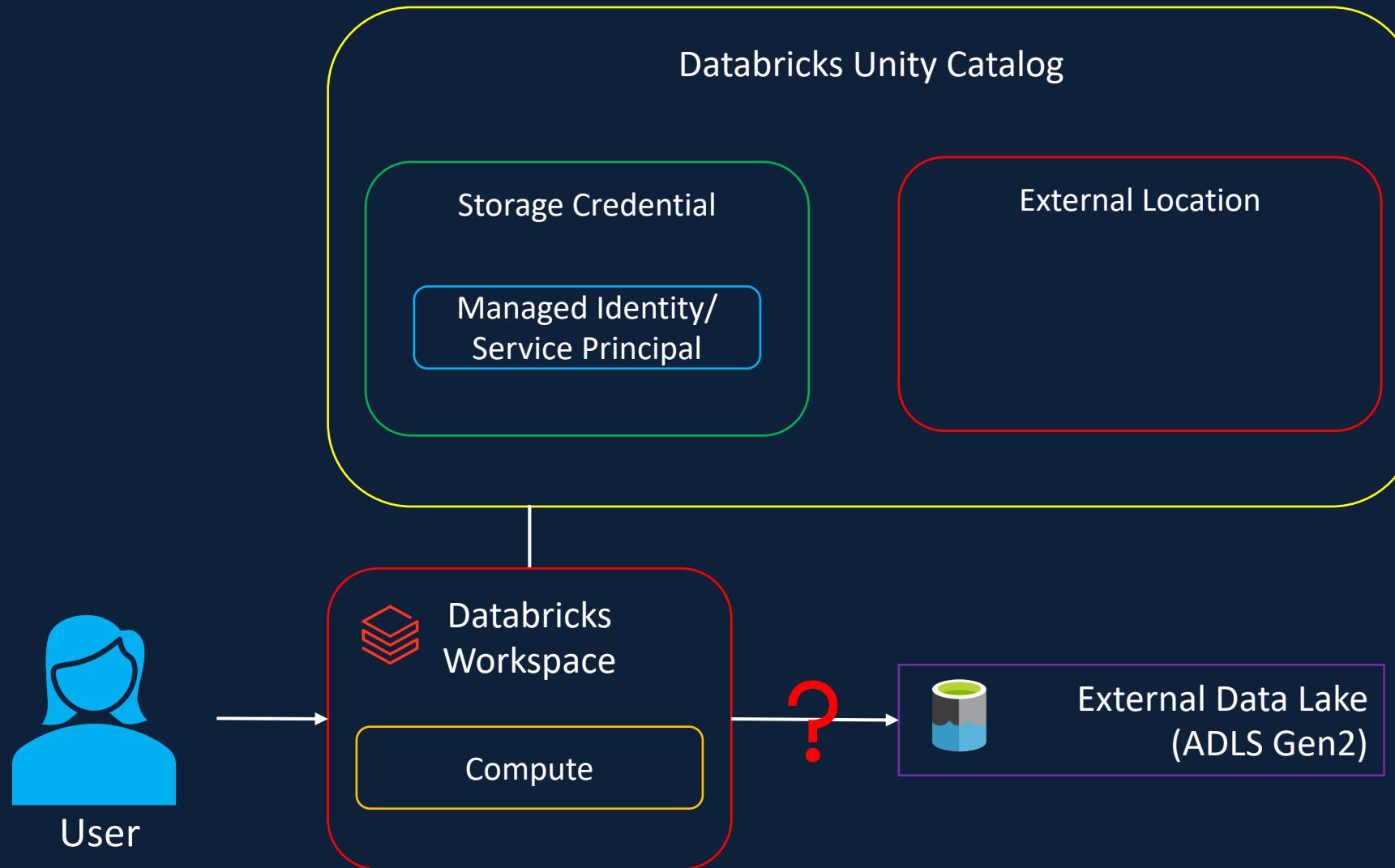
# Storage Credential/ External Location



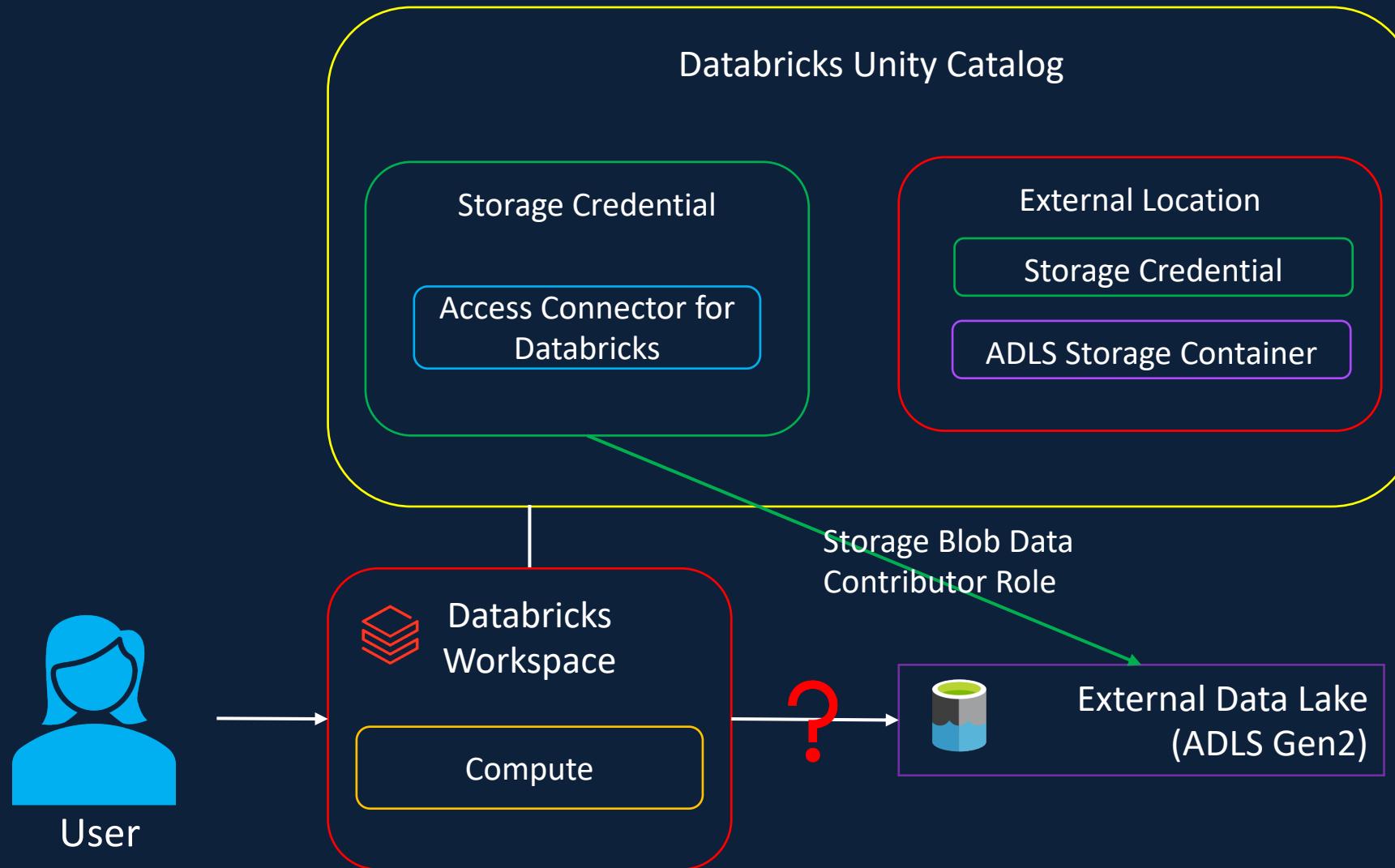
# Accessing External Locations



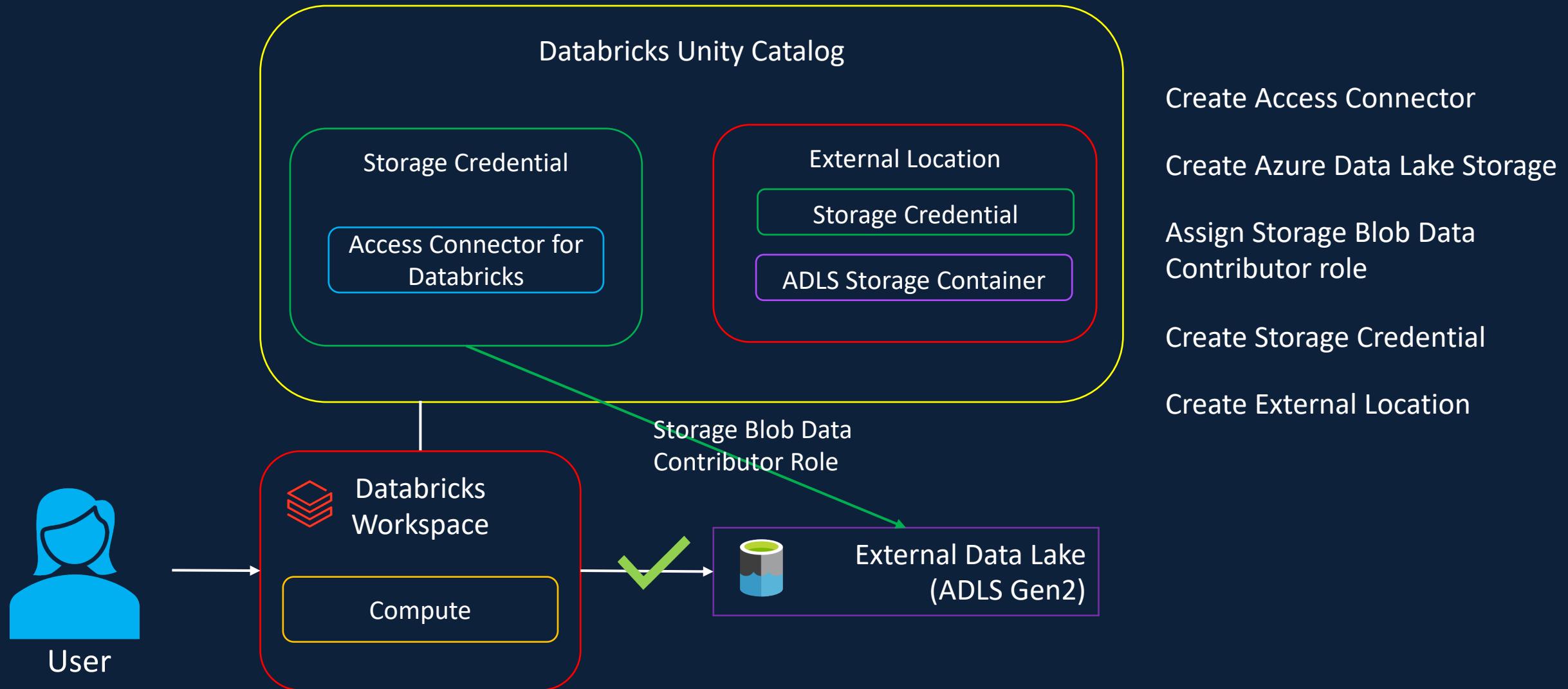
# Accessing External Locations



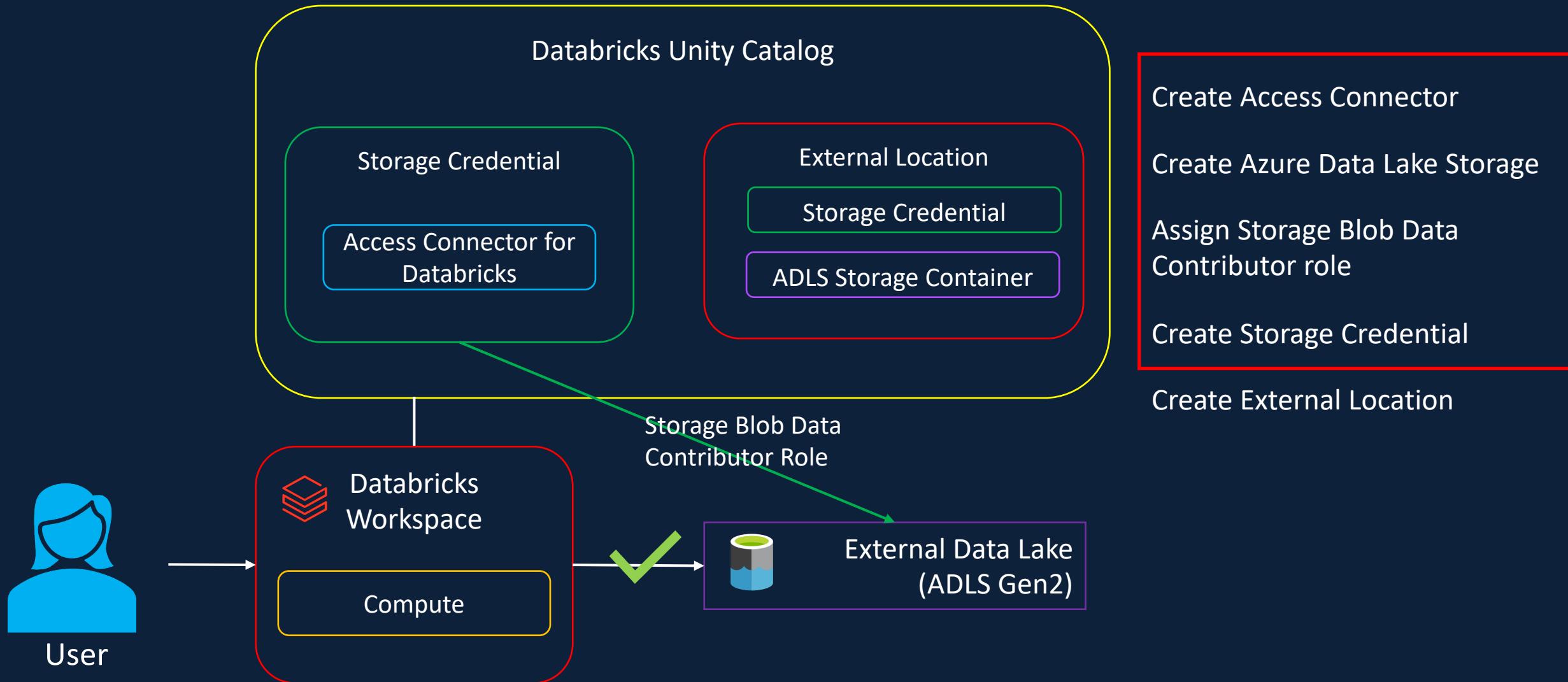
# Accessing External Locations



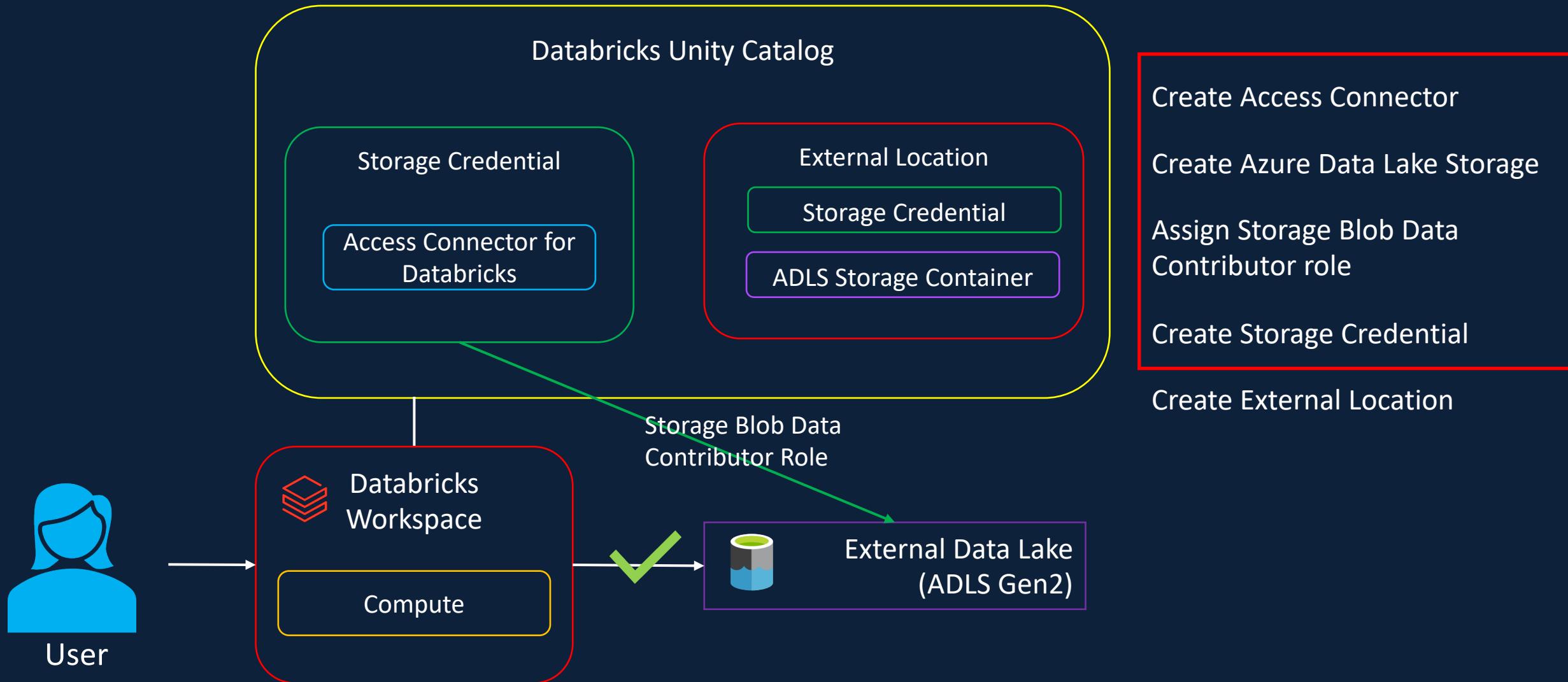
# Accessing External Locations



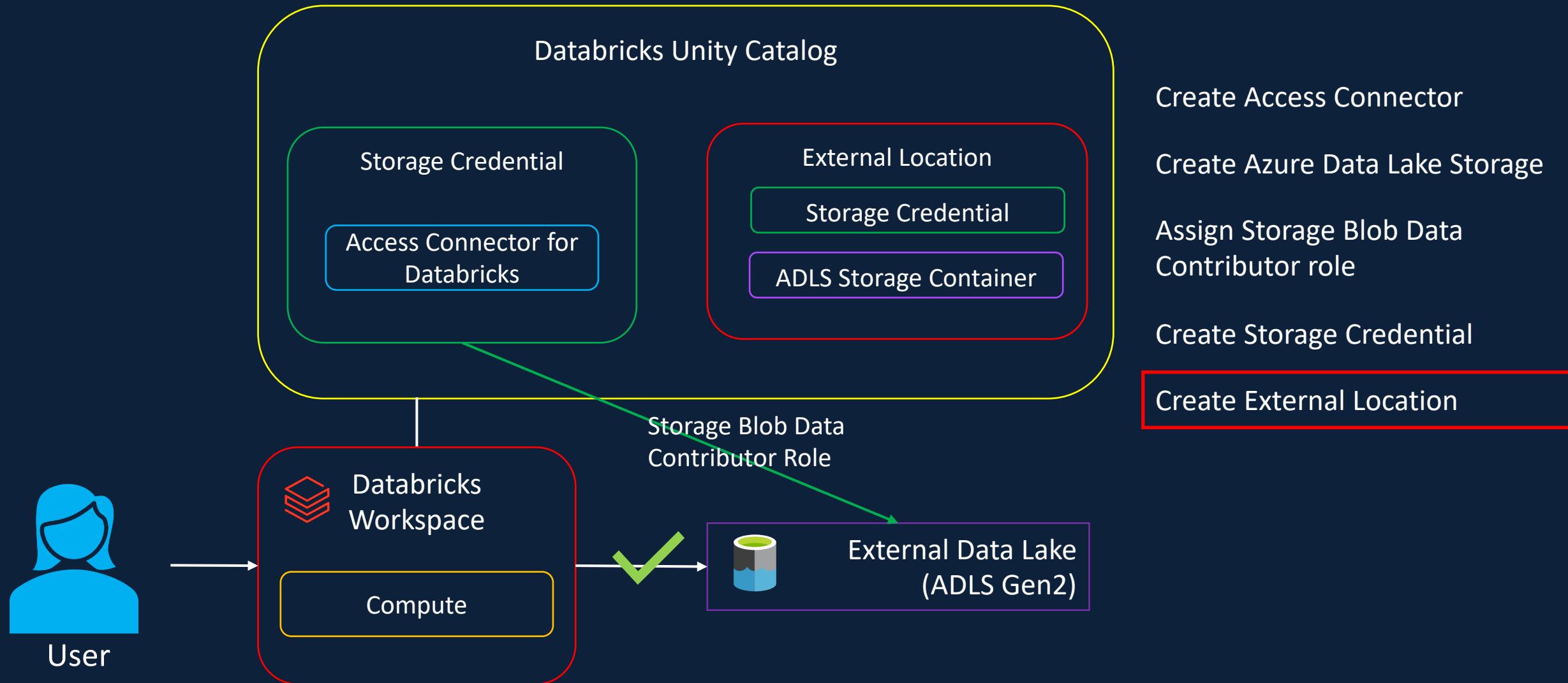
# Creating Storage Credential



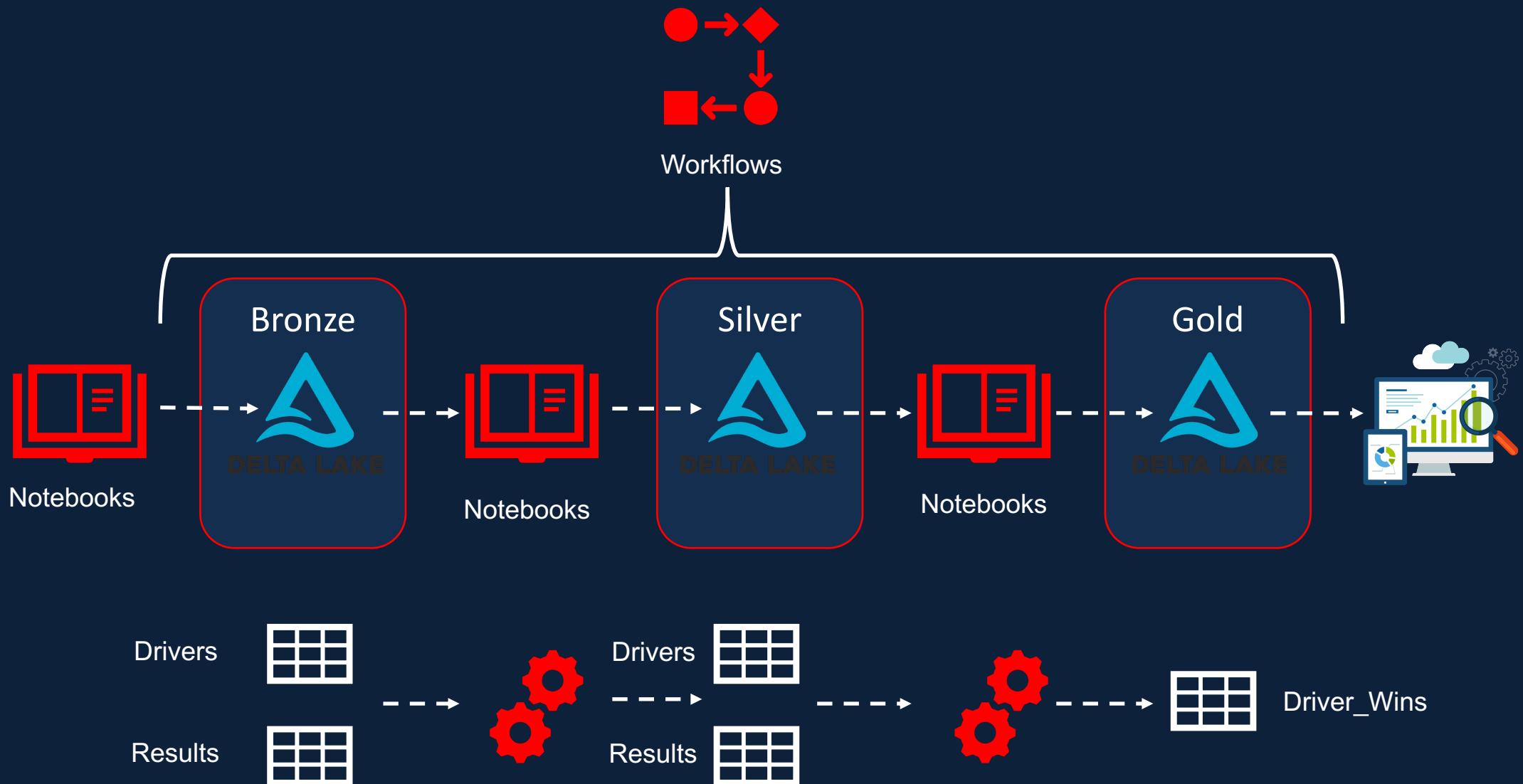
# Creating Storage Credential



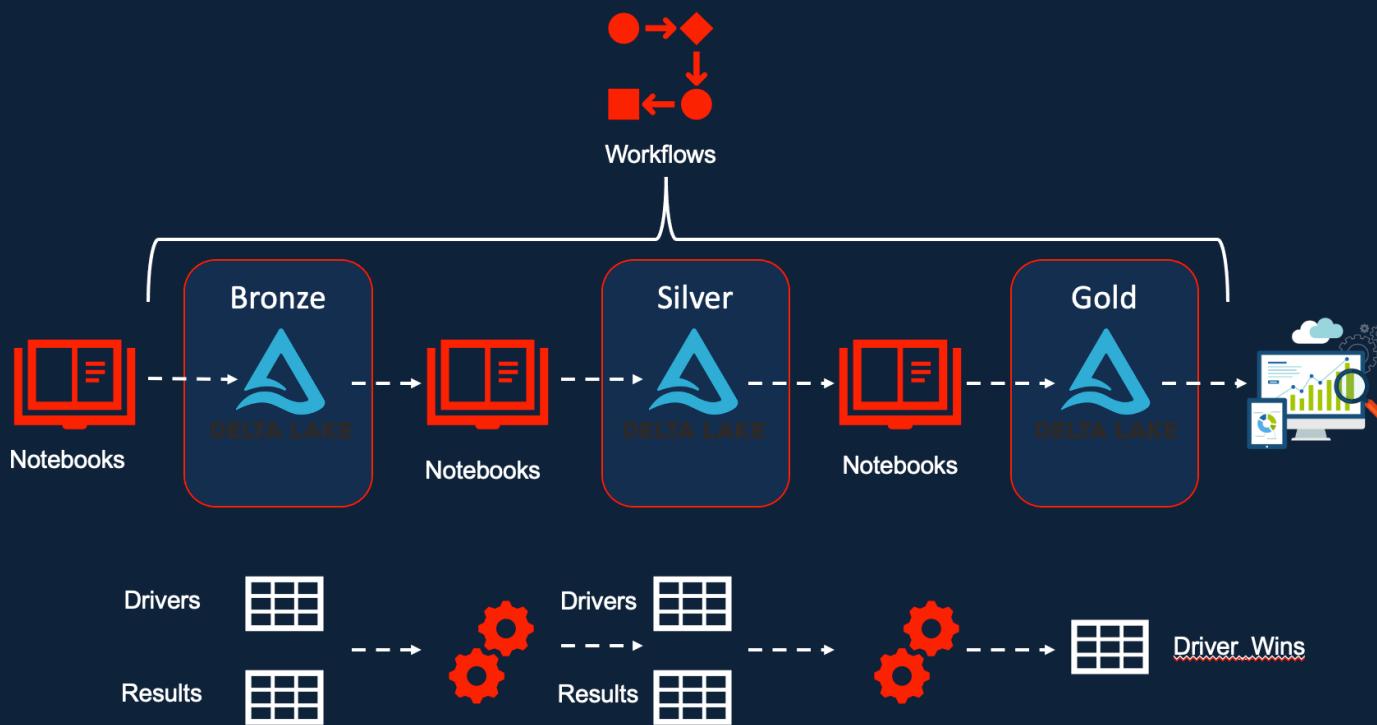
# Creating External Location



# Mini Project - Overview



# Mini Project - Overview



Create Storage Containers

Create External Locations

Create Catalog & Schemas

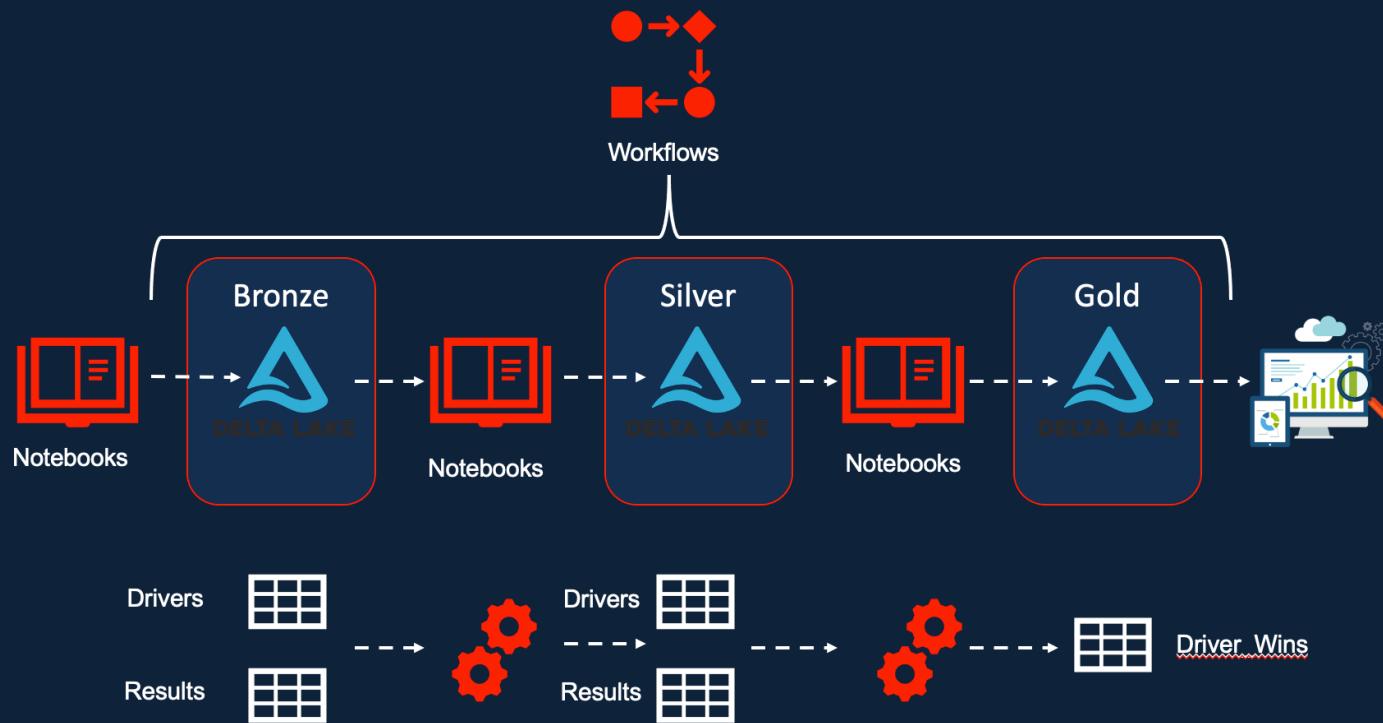
Create Bronze Tables (External)

Create Silver Tables (Managed)

Create Gold Table (Managed)

Create Databricks Workflow

# Mini Project – Create External Locations



Create Storage Containers

Create External Locations

Create Catalog & Schemas

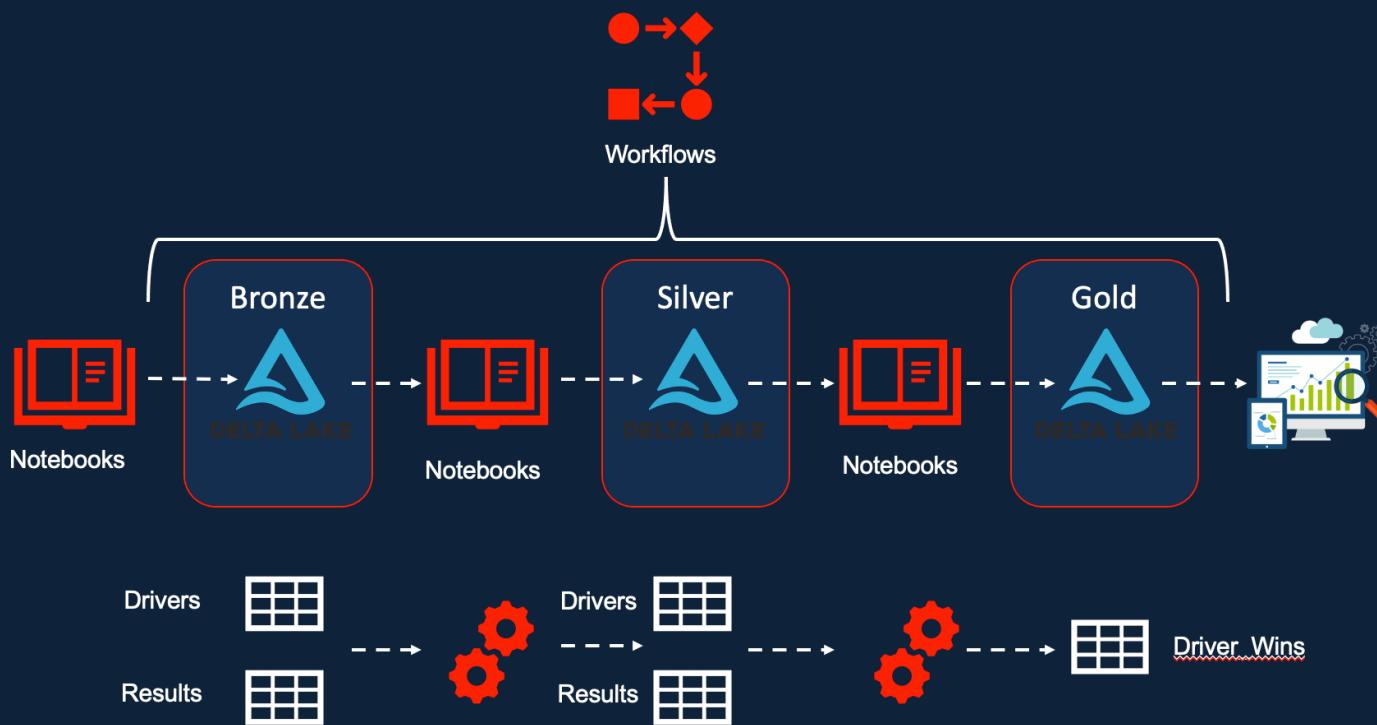
Create Bronze Tables (External)

Create Silver Tables (Managed)

Create Gold Table (Managed)

Create Databricks Workflow

# Mini Project – Create Catalogs & Schemas



Create Storage Containers

Create External Locations

Create Catalog & Schemas

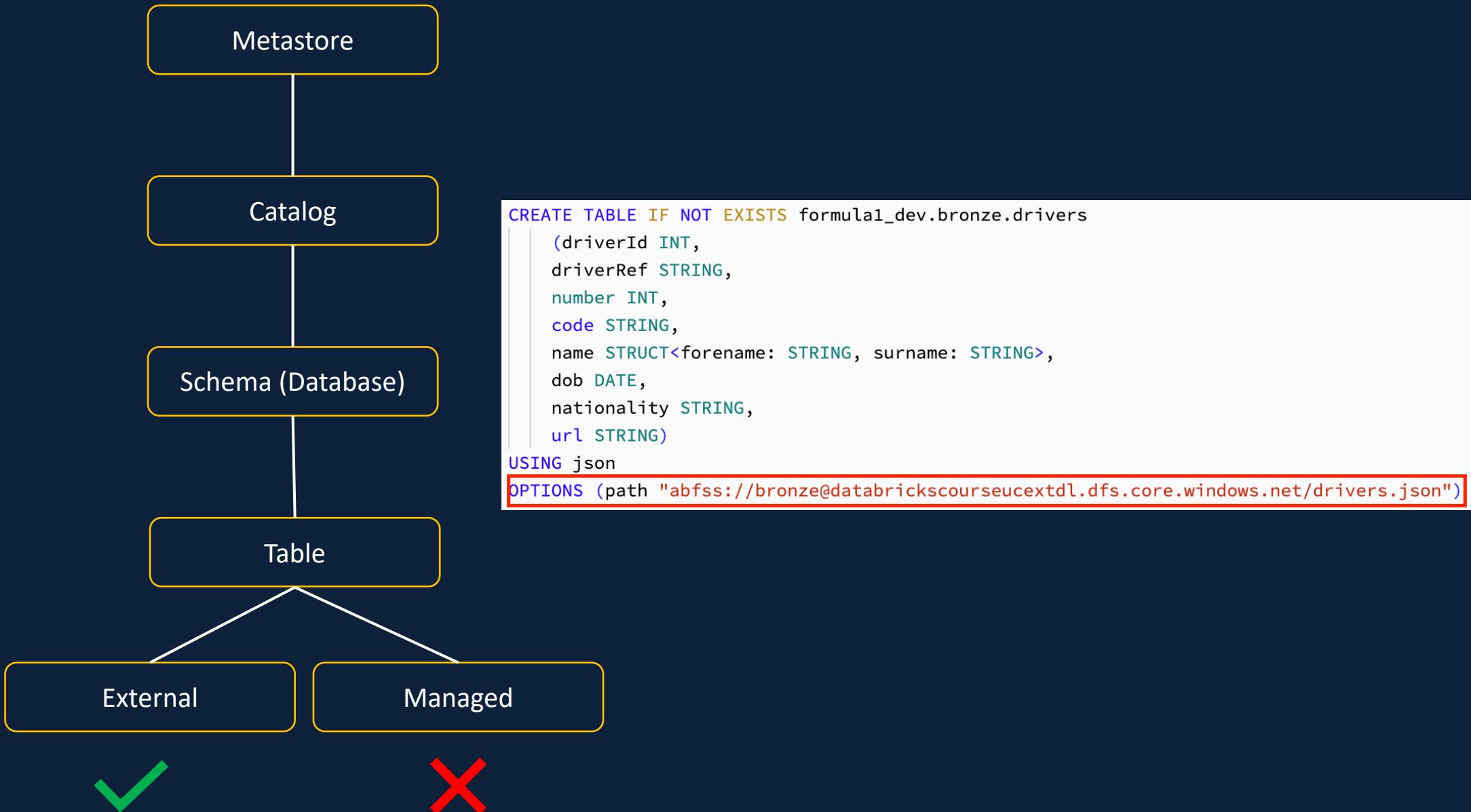
Create Bronze Tables (External)

Create Silver Tables (Managed)

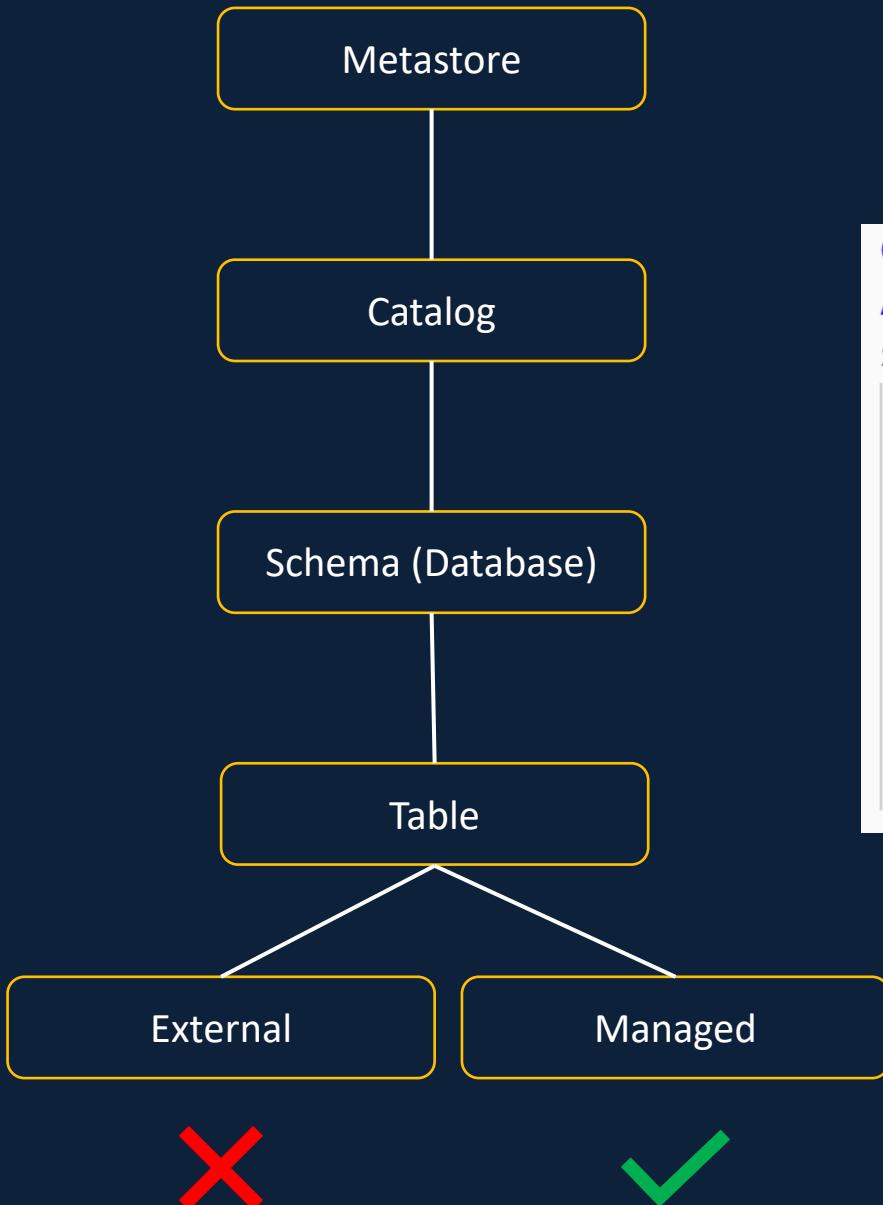
Create Gold Table (Managed)

Create Databricks Workflow

# Mini Project – Create Catalogs & Schemas

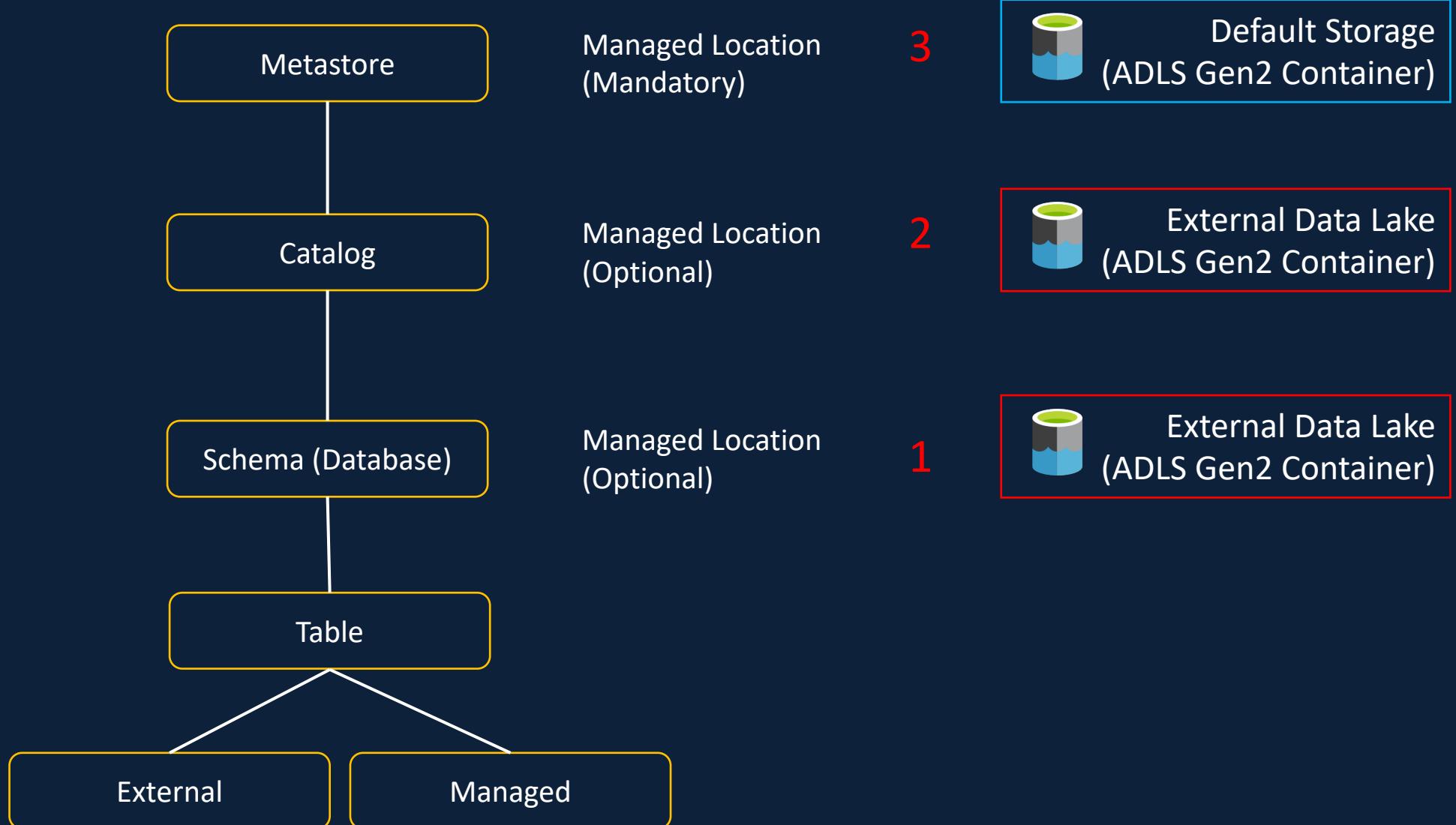


# Mini Project – Create Catalogs & Schemas

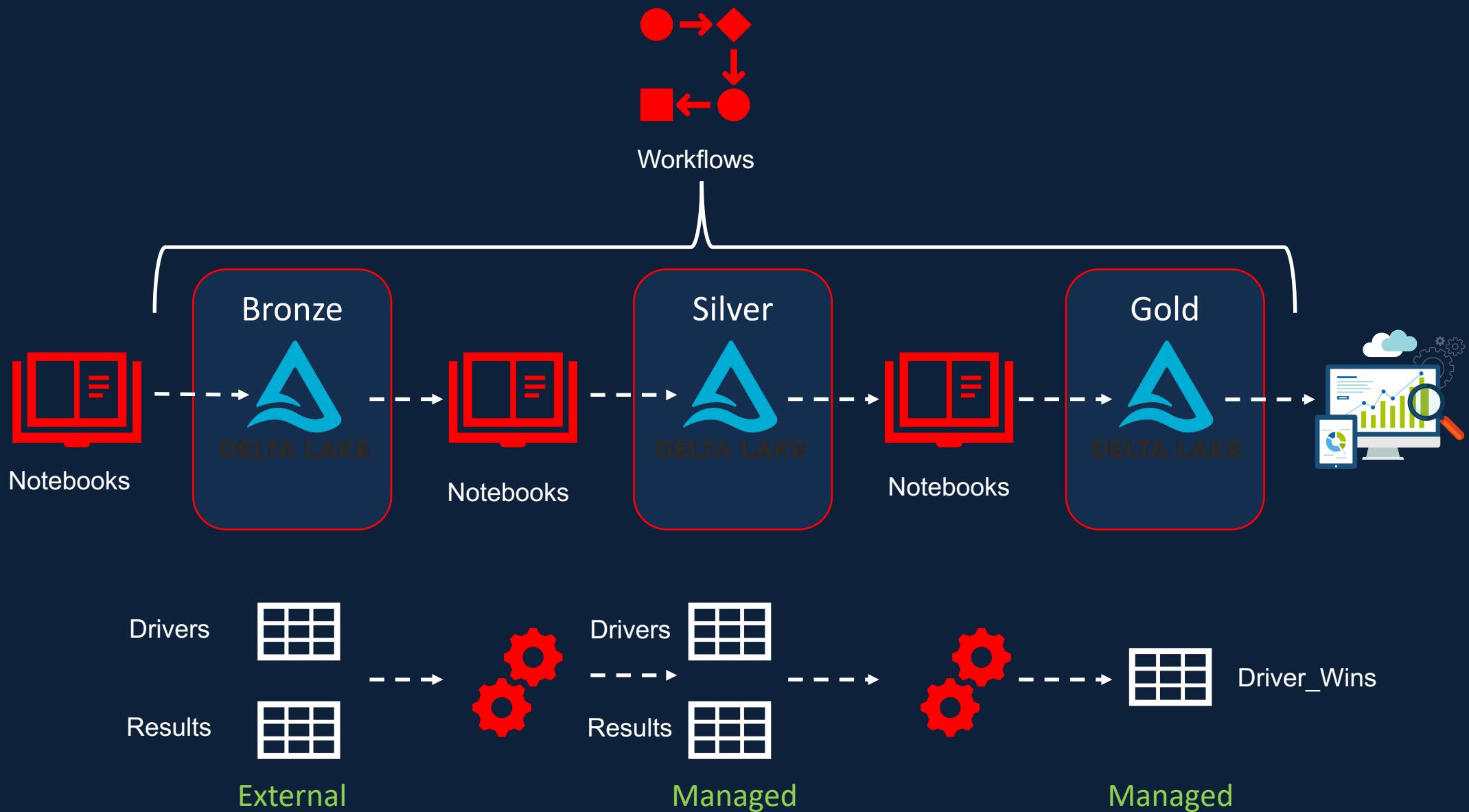


```
CREATE TABLE formula1_dev.silver.drivers  
AS  
SELECT driverId AS driver_id,  
driverRef AS driver_ref,  
number,  
code,  
concat(name.forename, ' ', name.surname) AS name,  
dob,  
nationality,  
current_timestamp() AS ingestion_date  
FROM formula1_dev.bronze.drivers;
```

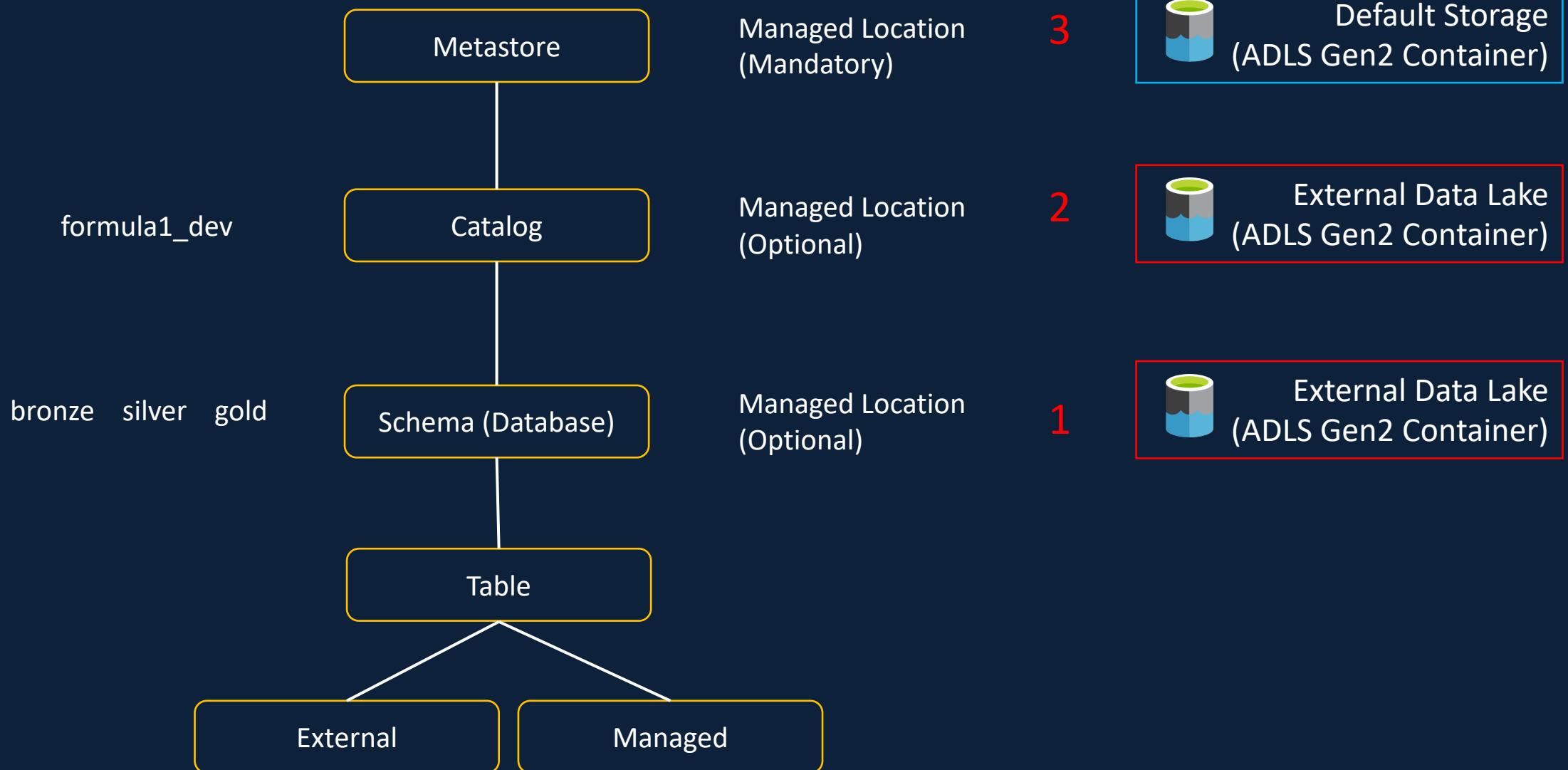
# Mini Project – Create Catalogs & Schemas



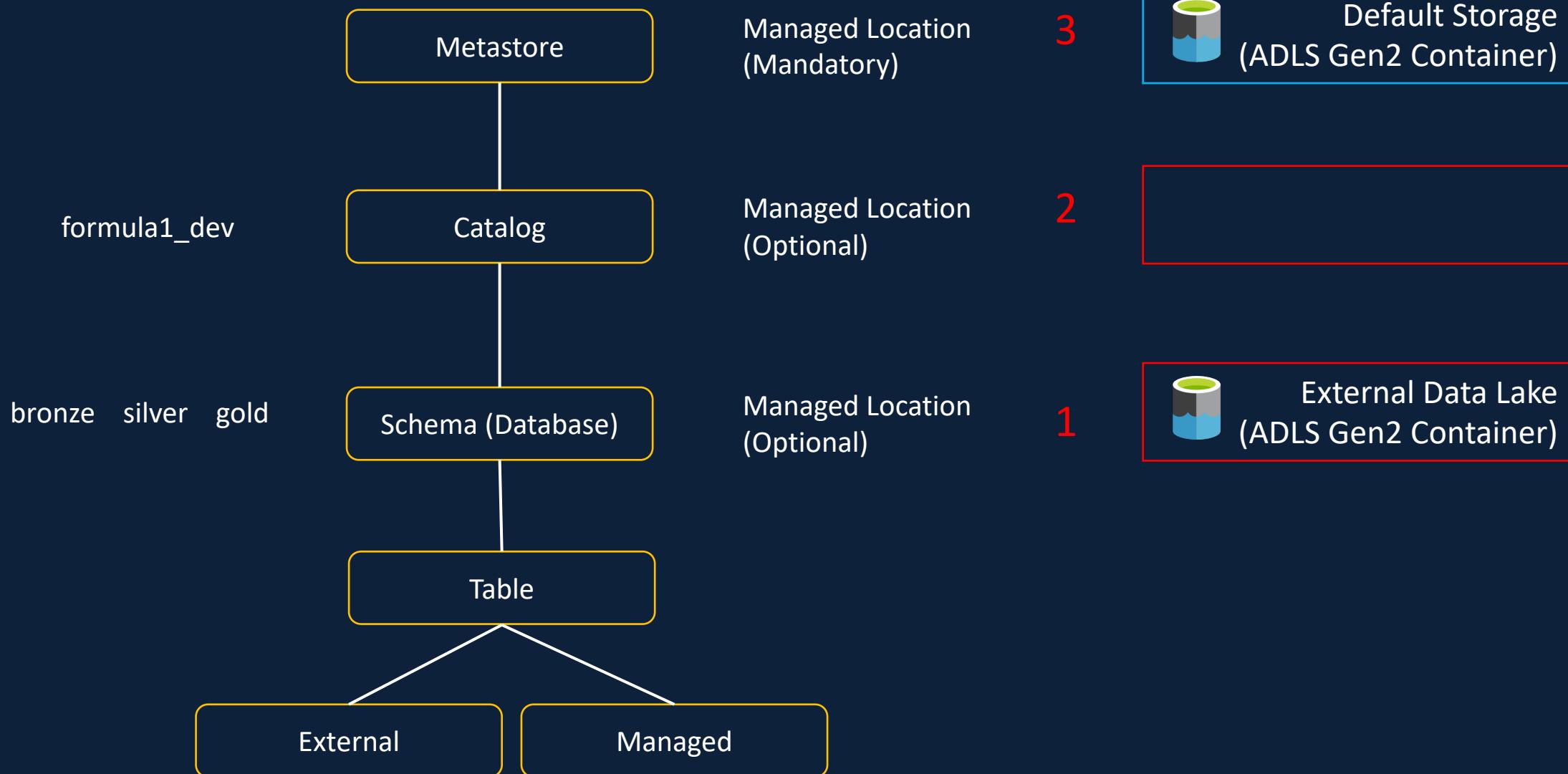
# Mini Project – Create Catalogs & Schemas



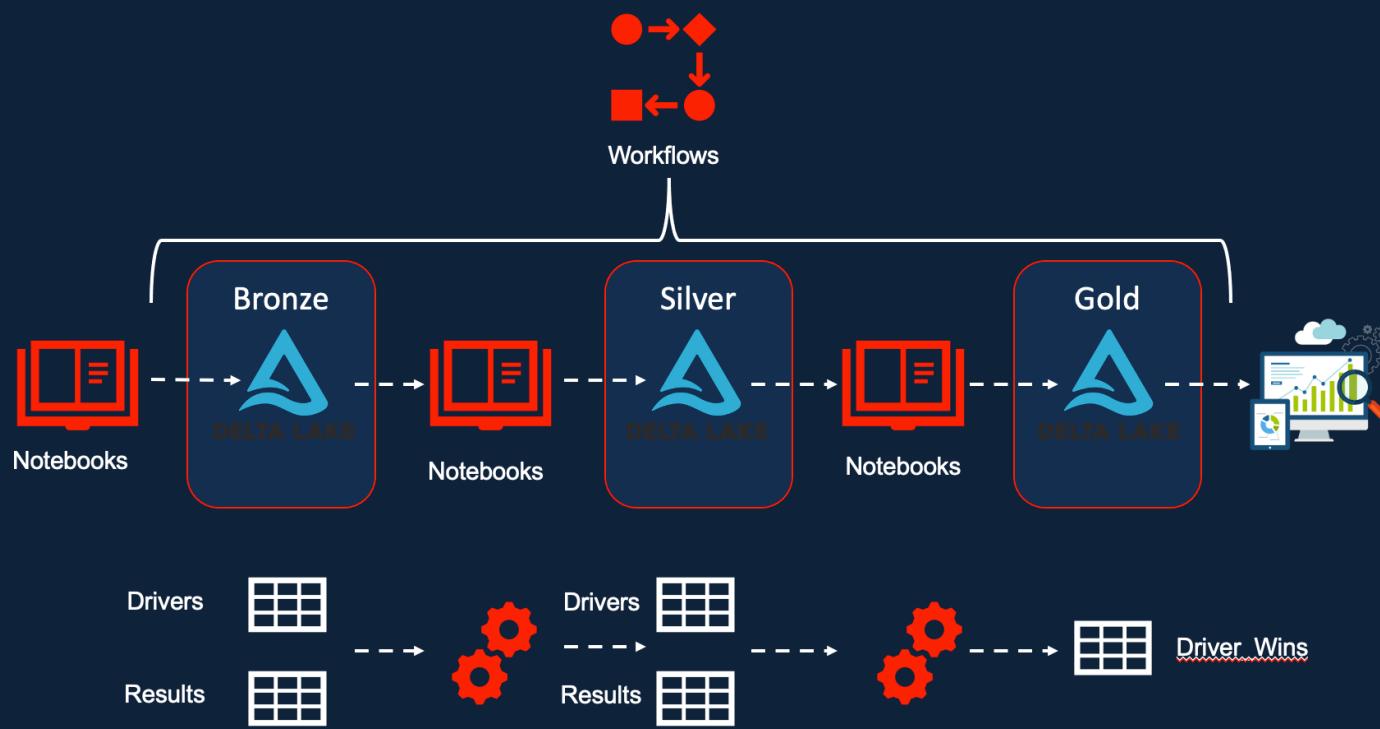
# Mini Project – Create Catalogs & Schemas



# Mini Project – Create Catalogs & Schemas



# Mini Project – Create External Tables



Create Storage Containers

Create External Locations

Create Catalog & Schemas

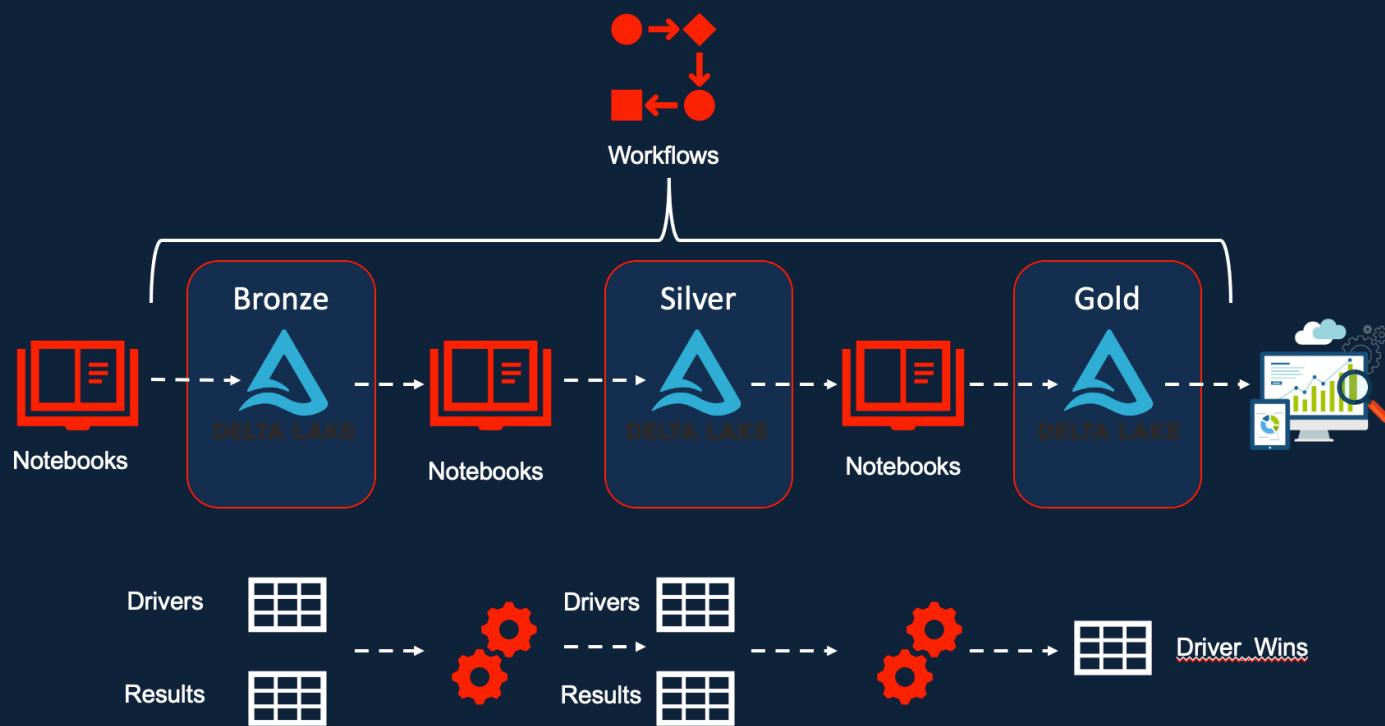
Create Bronze Tables (External)

Create Silver Tables (Managed)

Create Gold Table (Managed)

Create Databricks Workflow

# Mini Project – Create Managed Tables



Create Storage Containers

Create External Locations

Create Catalog & Schemas

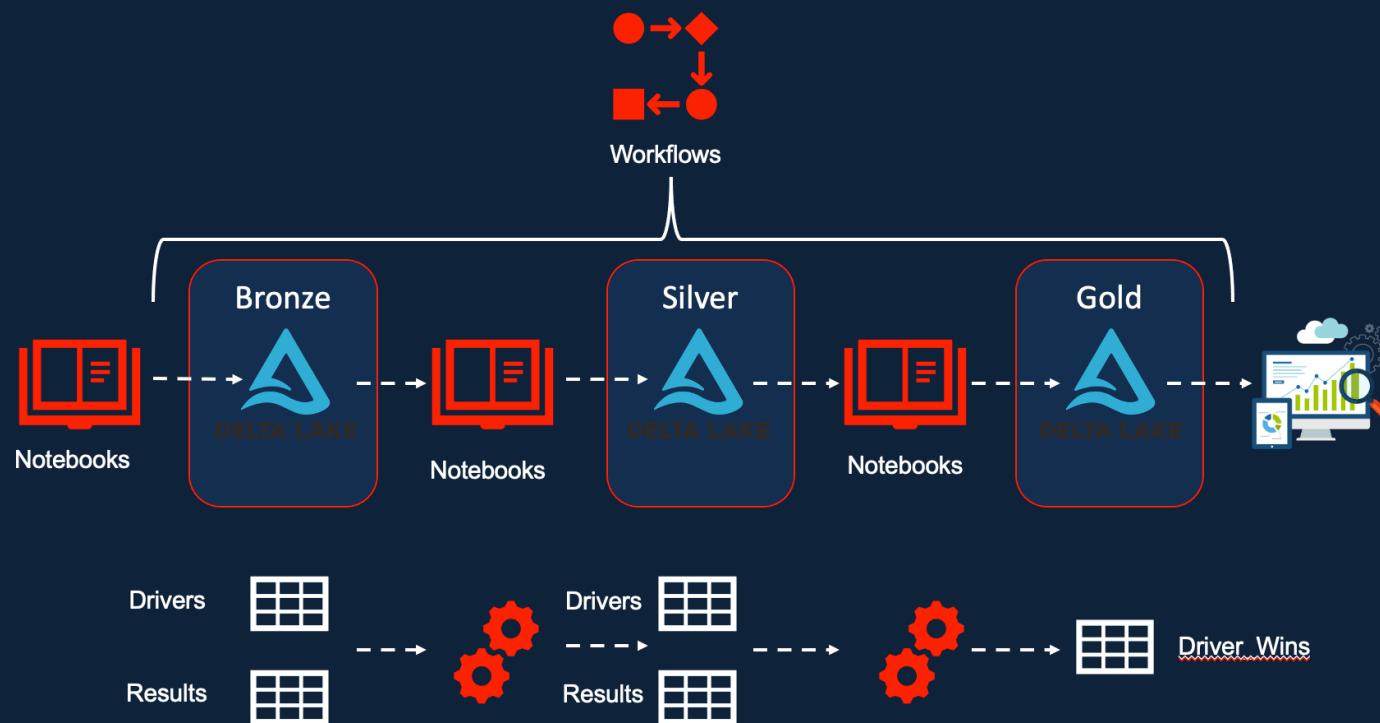
Create Bronze Tables (External)

Create Silver Tables (Managed)

Create Gold Table (Managed)

Create Databricks Workflow

# Mini Project – Create Workflow



Create Storage Containers

Create External Locations

Create Catalog & Schemas

Create Bronze Tables (External)

Create Silver Tables (Managed)

Create Gold Table (Managed)

Create Databricks Workflow

# Unity Catalog – Key capabilities



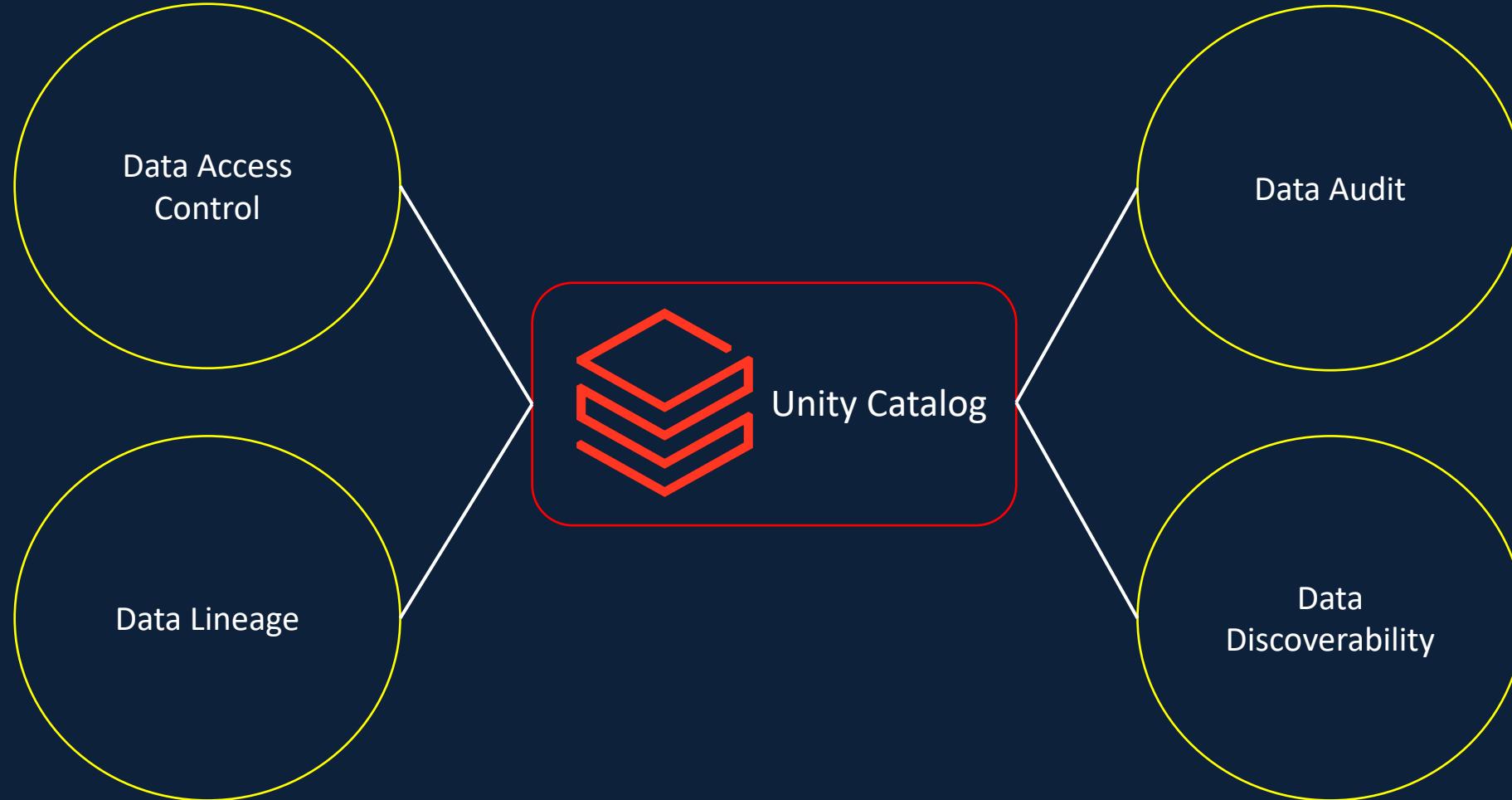
Data Discovery

Data Audit

Data Lineage

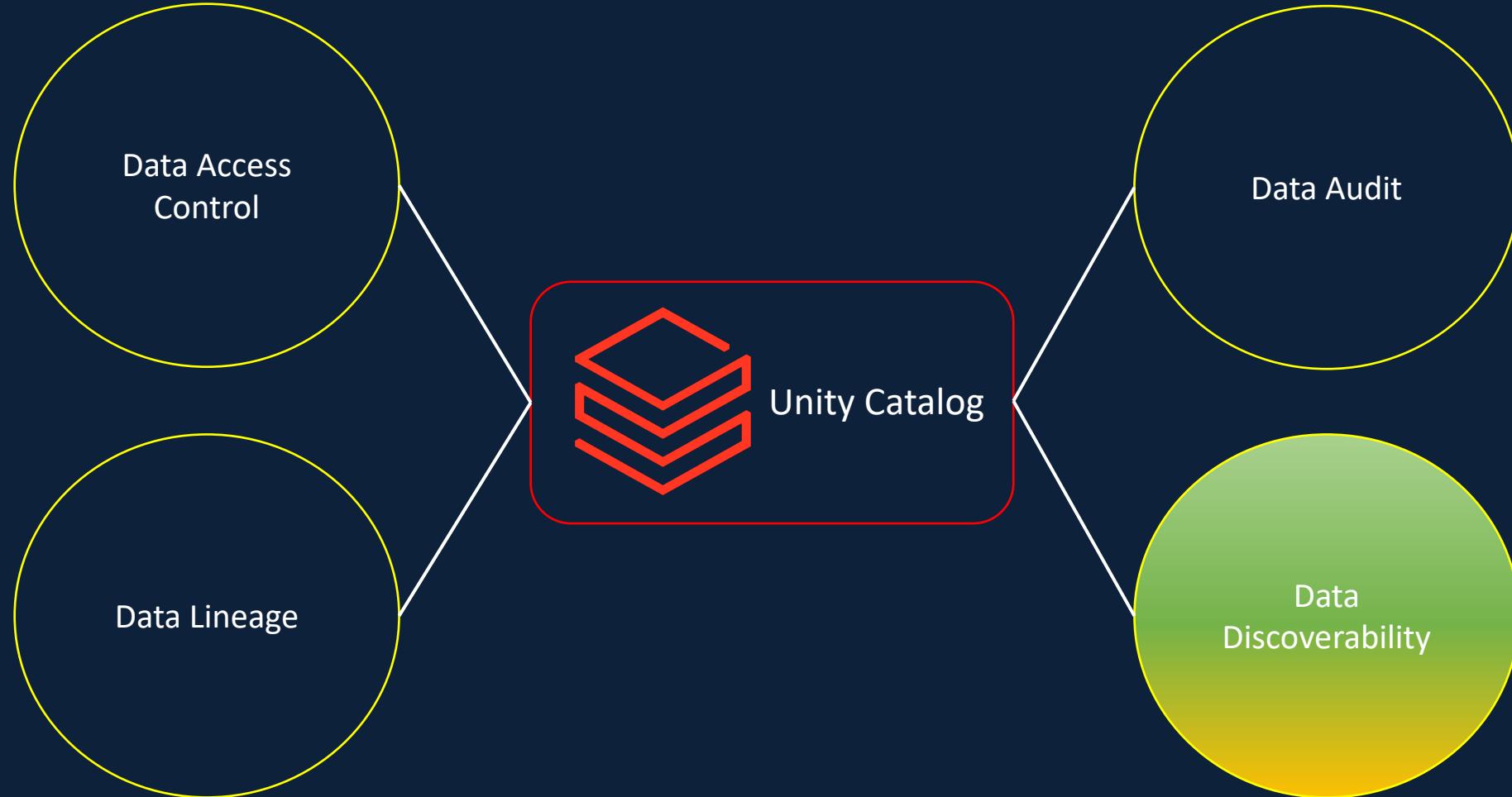
Data Access Control

# Unity Catalog – Data Discoverability



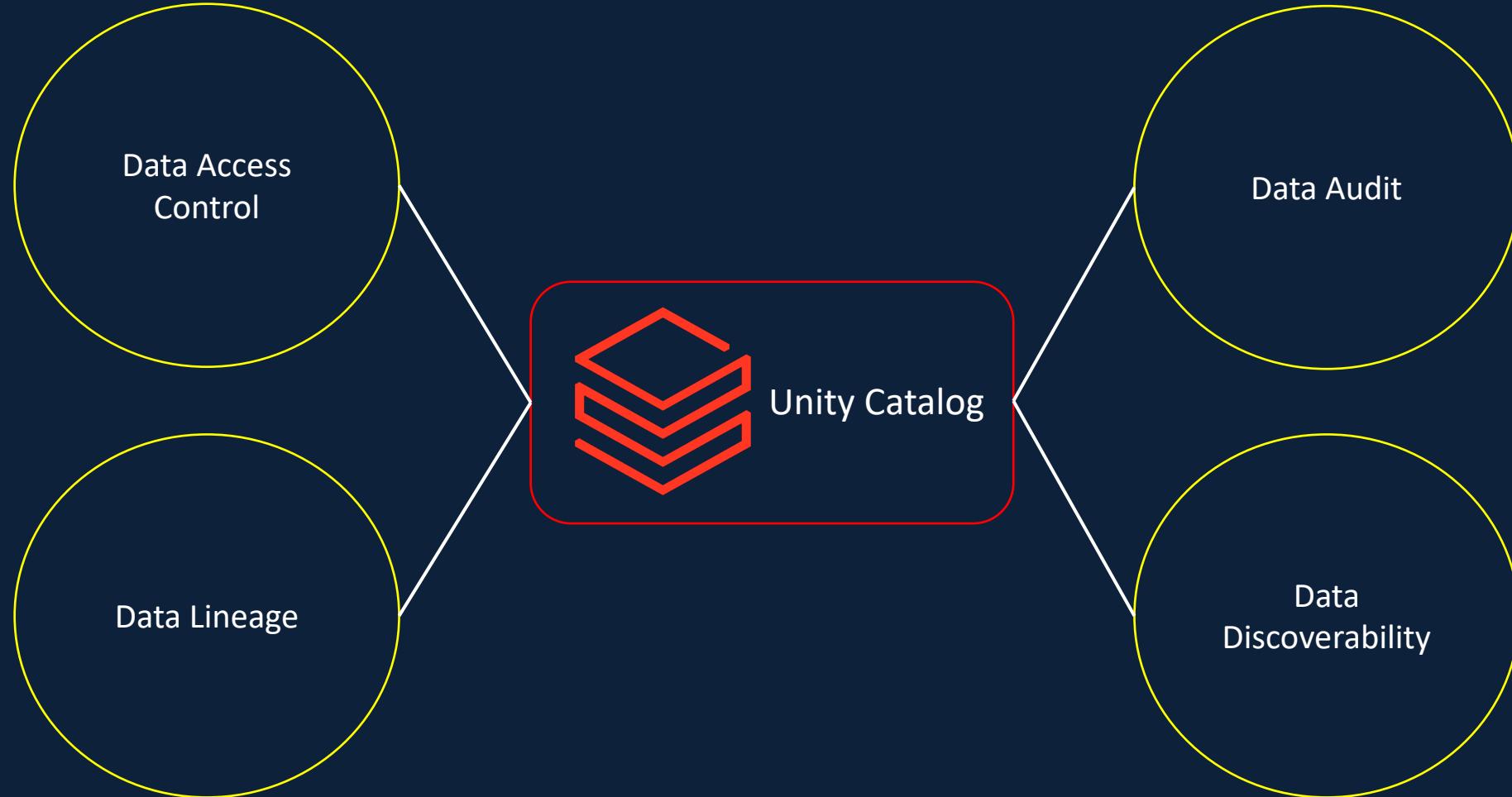
Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Discoverability



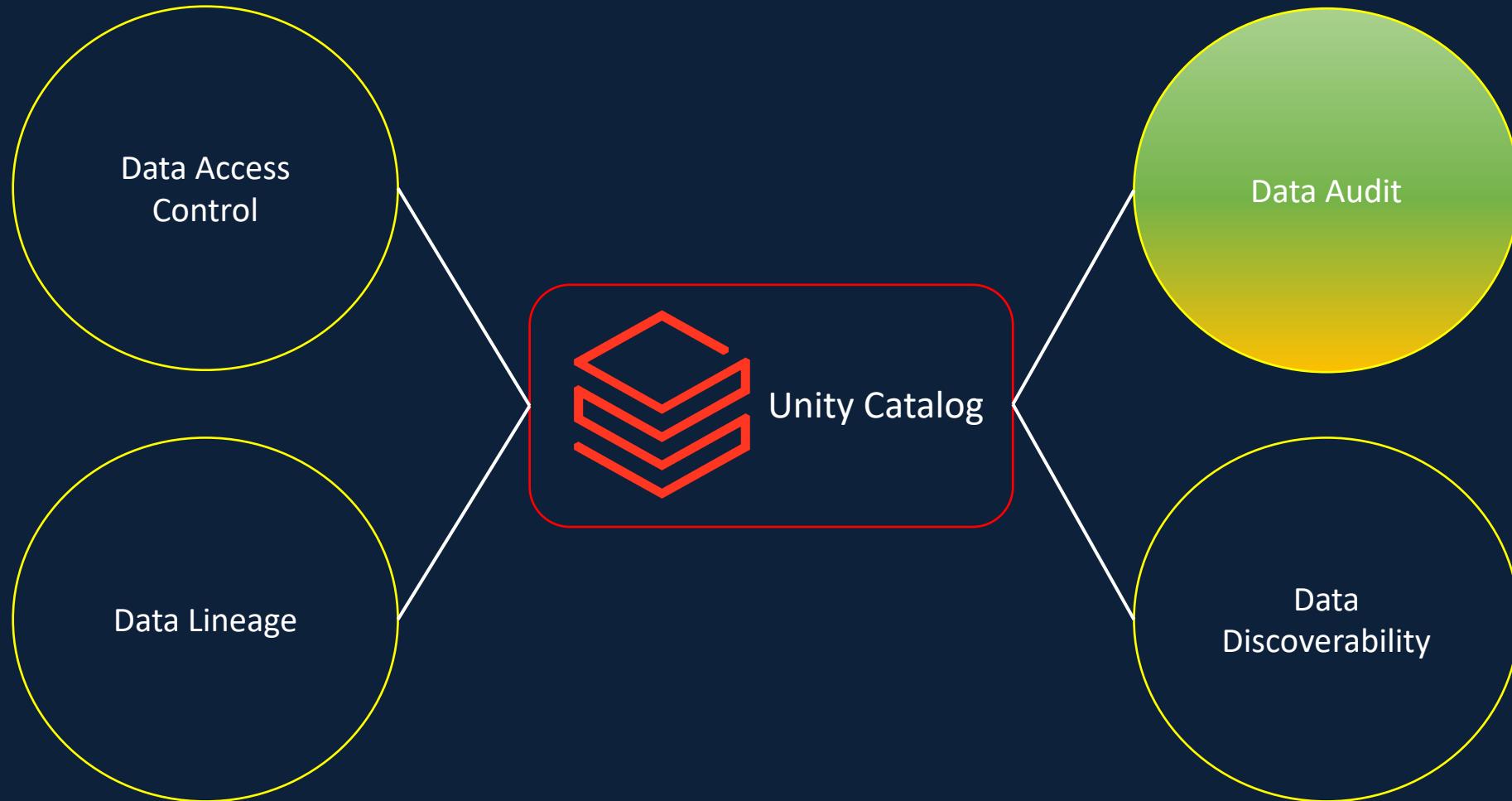
Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Audit



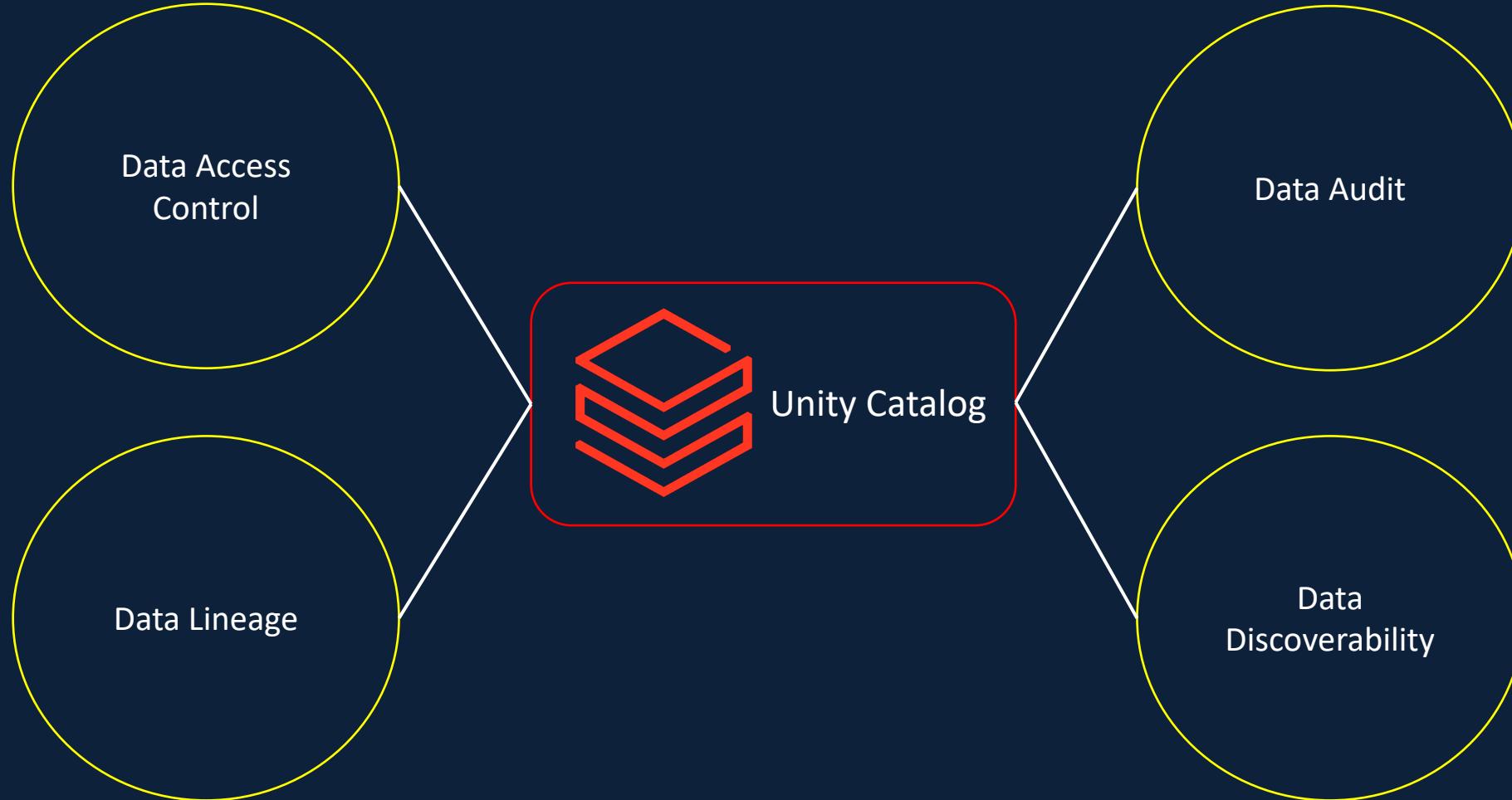
Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Discoverability



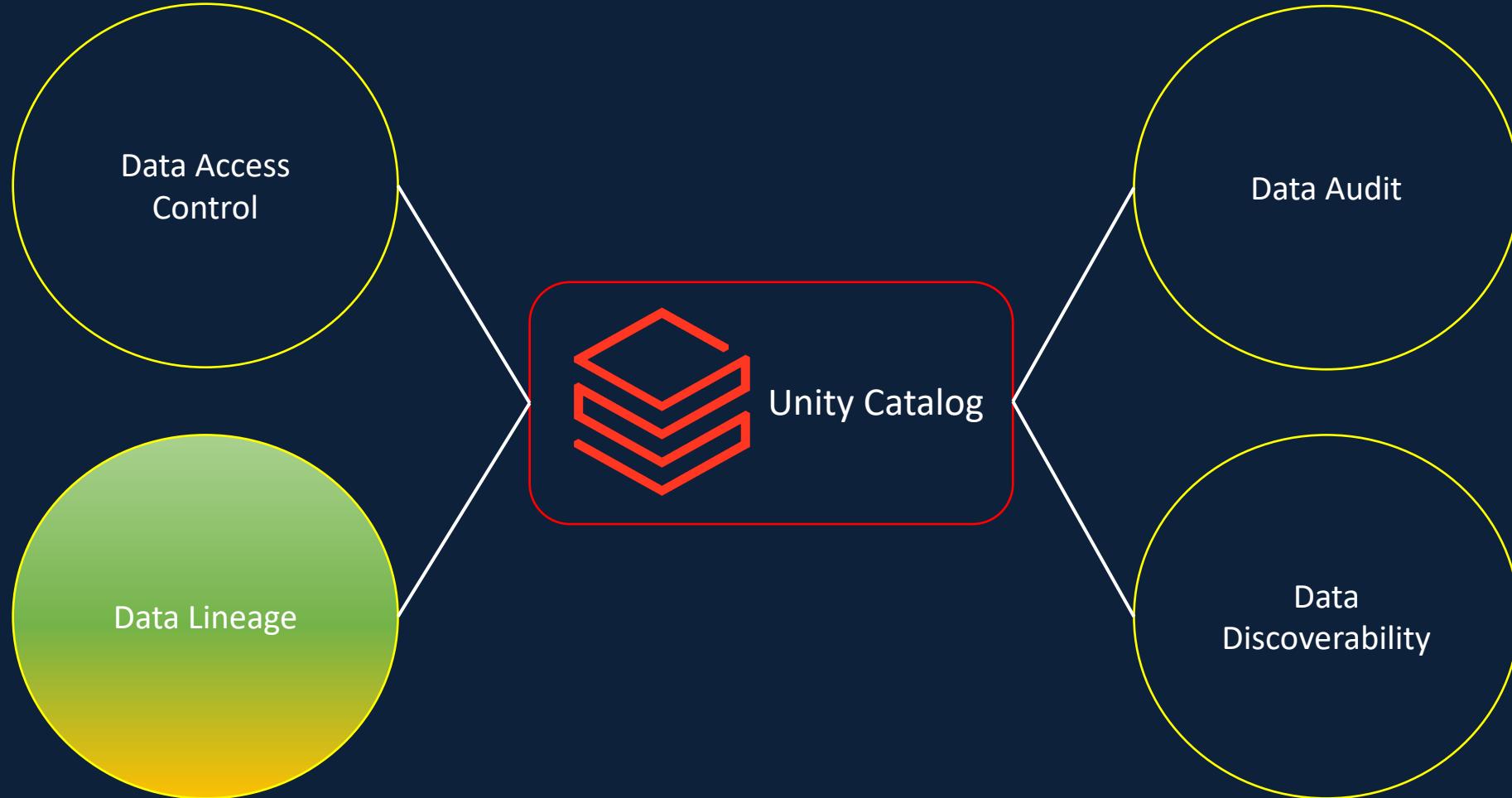
Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Lineage



Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Lineage



Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Data Lineage

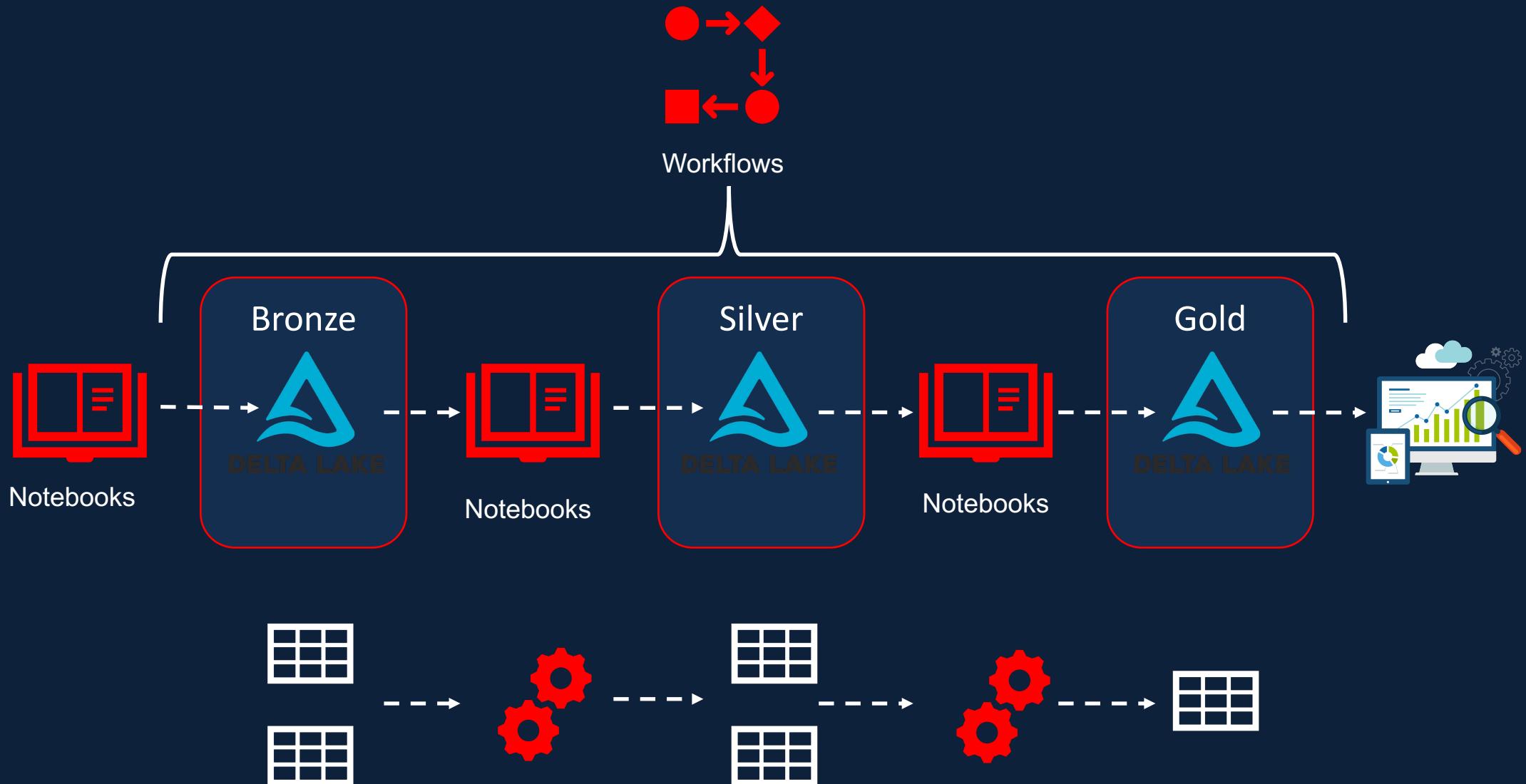
Data Lineage is the process of following/ tracking the journey of data within a pipeline.

What is the origin

How has it changed/ transformed

What is the destination

# Data Lineage



# Data Lineage - Benefits

Root cause analysis

Improved impact analysis

Trust & Reliability

Better compliance

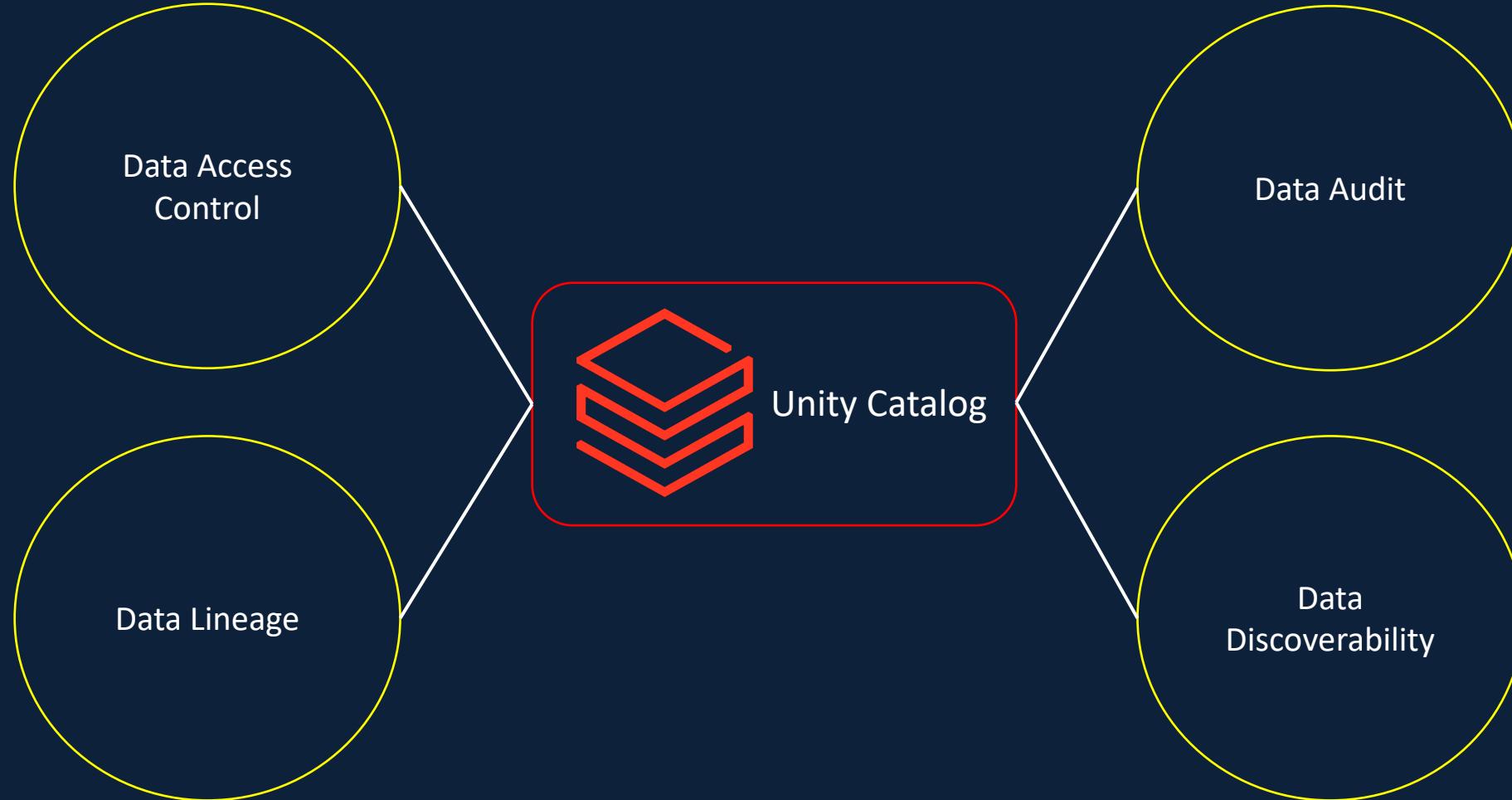
# Data Lineage - Limitations

Only available for tables registered in Unity Catalog Metastore

Available only for the last 30 days

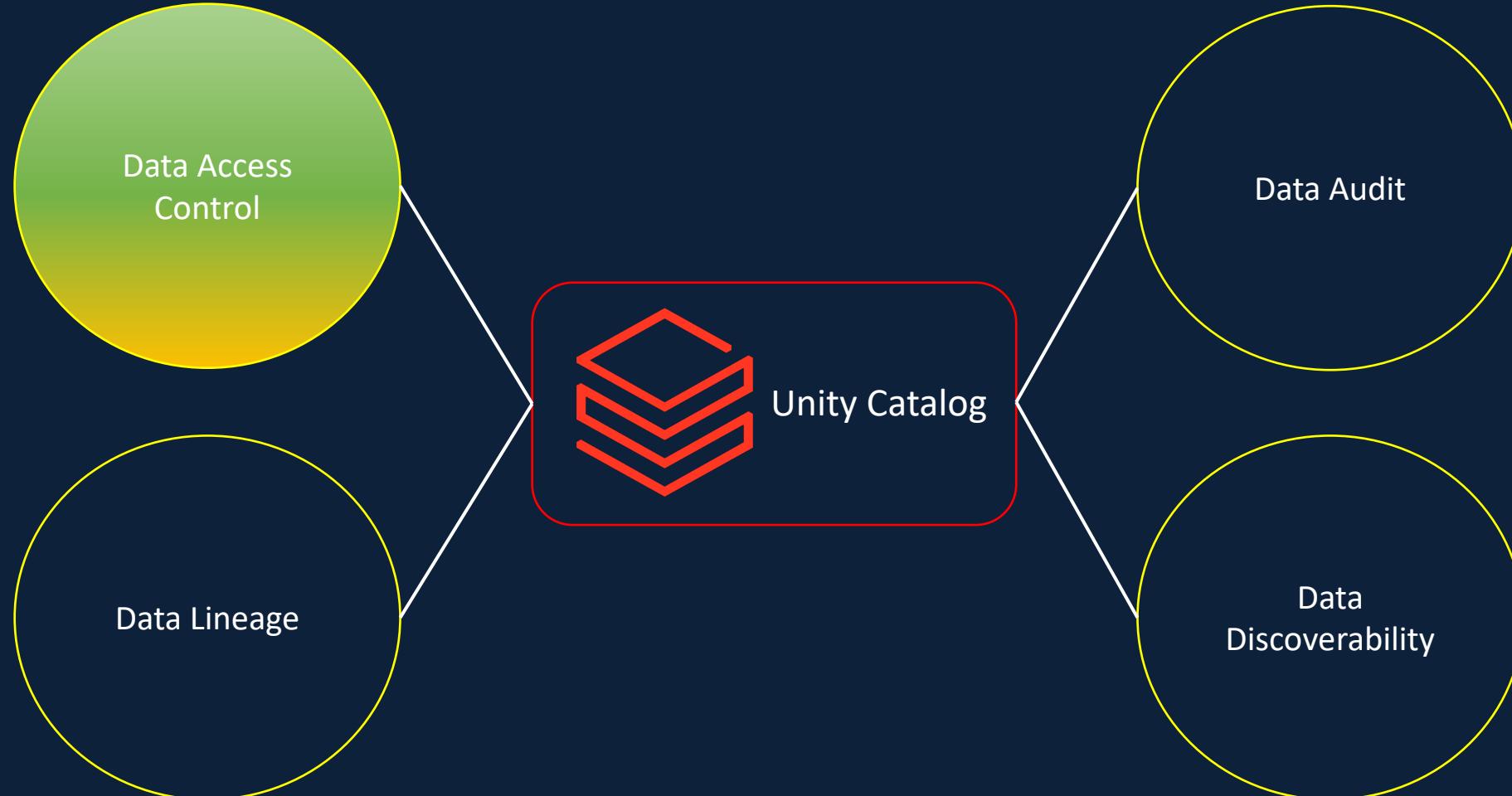
Limited column level lineage

# Unity Catalog – Data Access Control



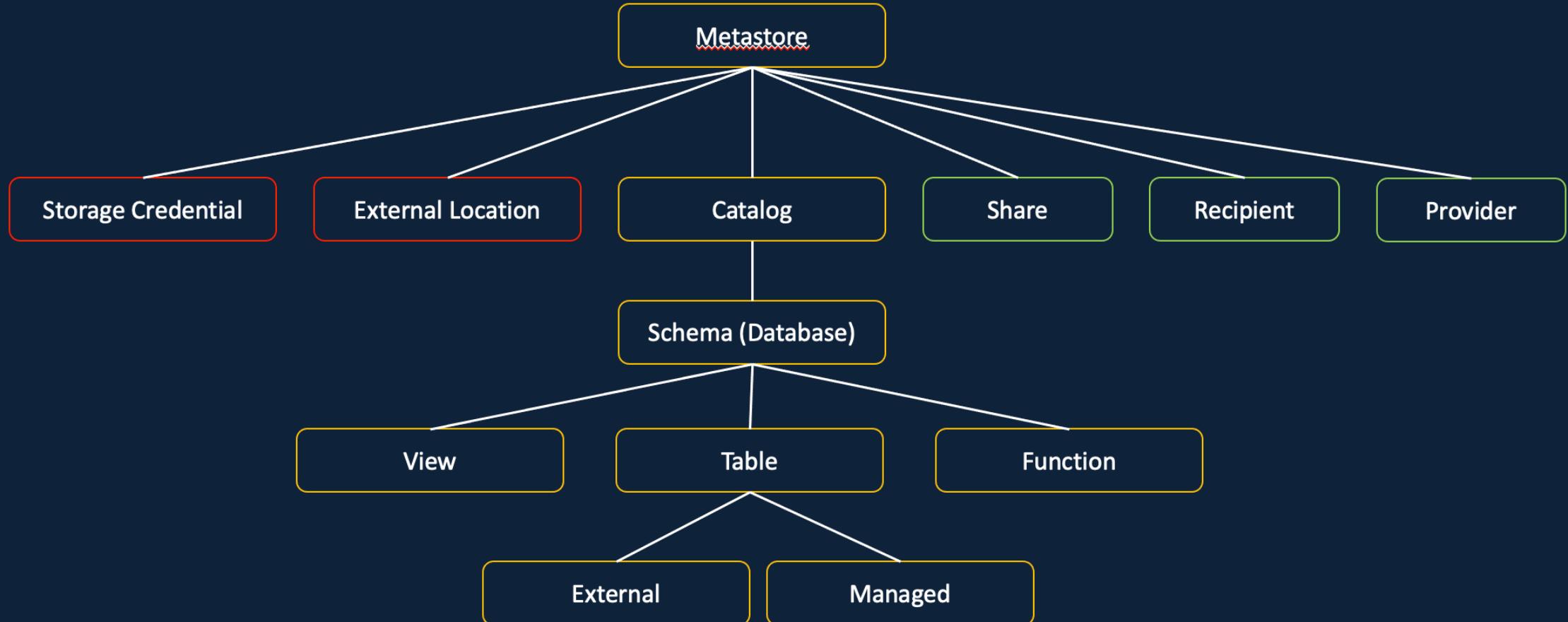
Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

# Unity Catalog – Data Access Control

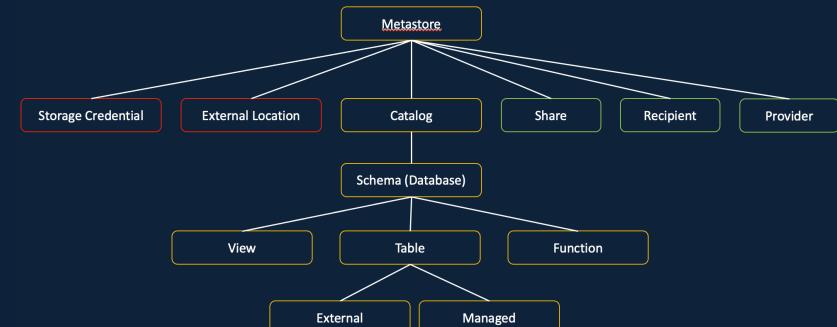
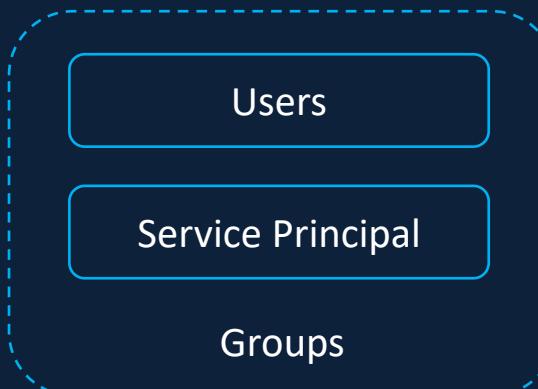


Unity Catalog is a Databricks offered **unified solution** for implementing **data governance** in the Data Lakehouse

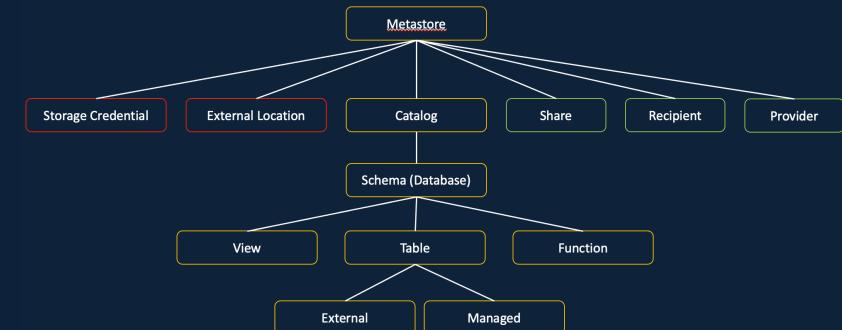
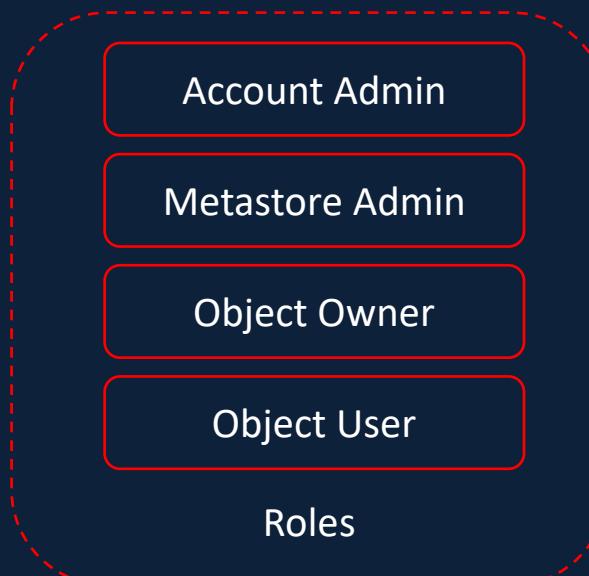
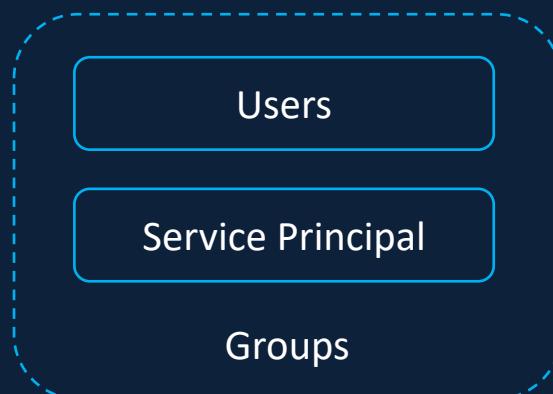
# Unity Catalog - Securable Objects



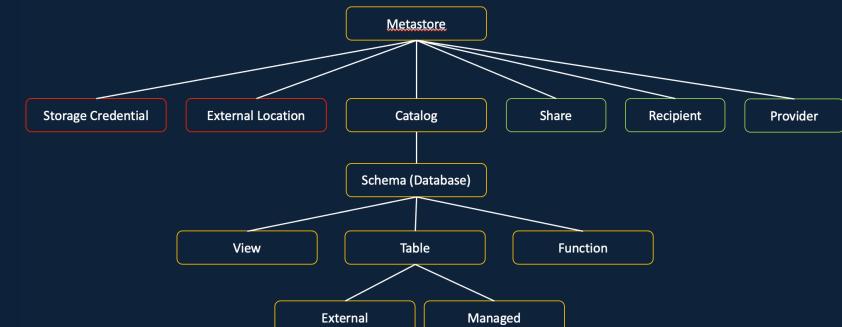
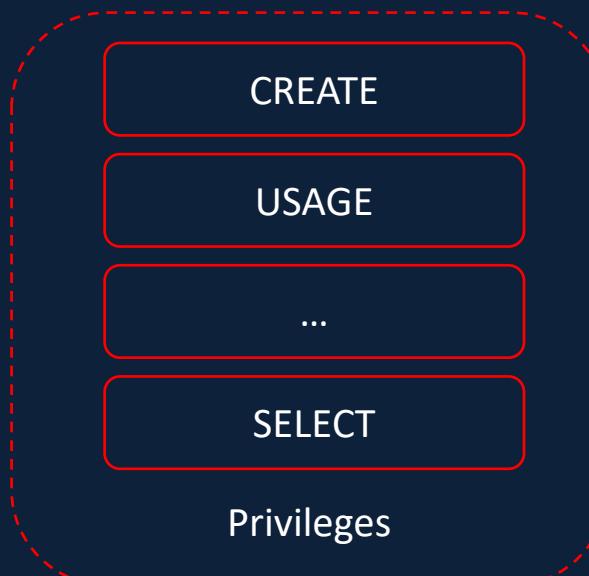
# Unity Catalog - Security Model



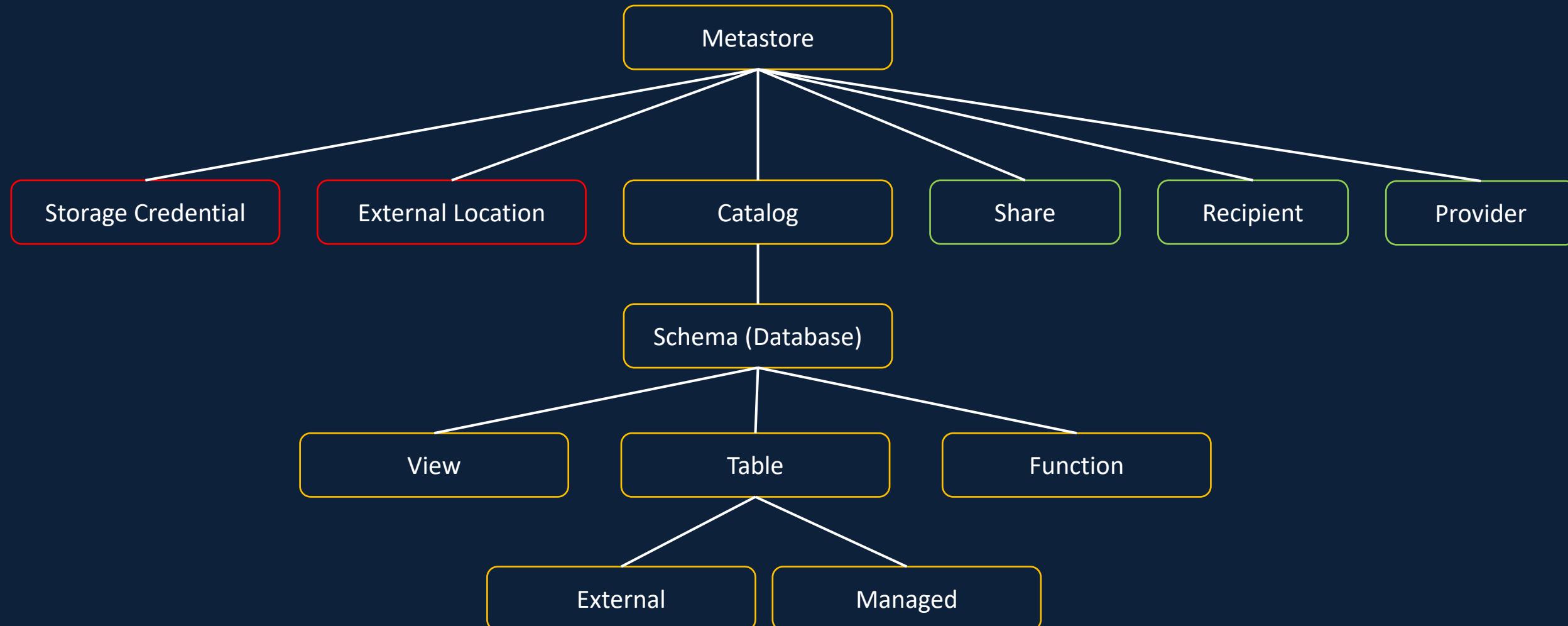
# Security Model – Role Based Access



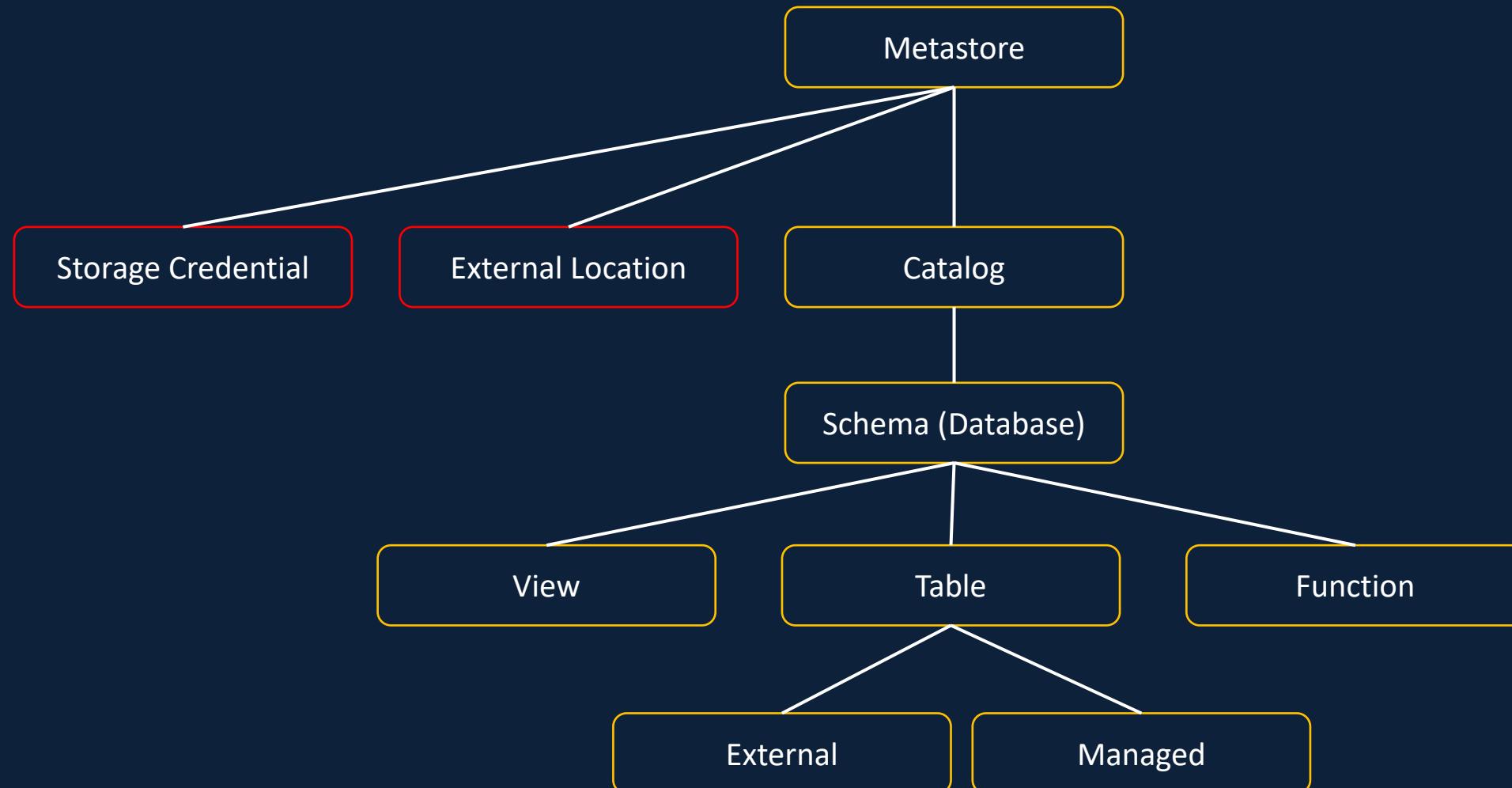
# Security Model – Access Control List



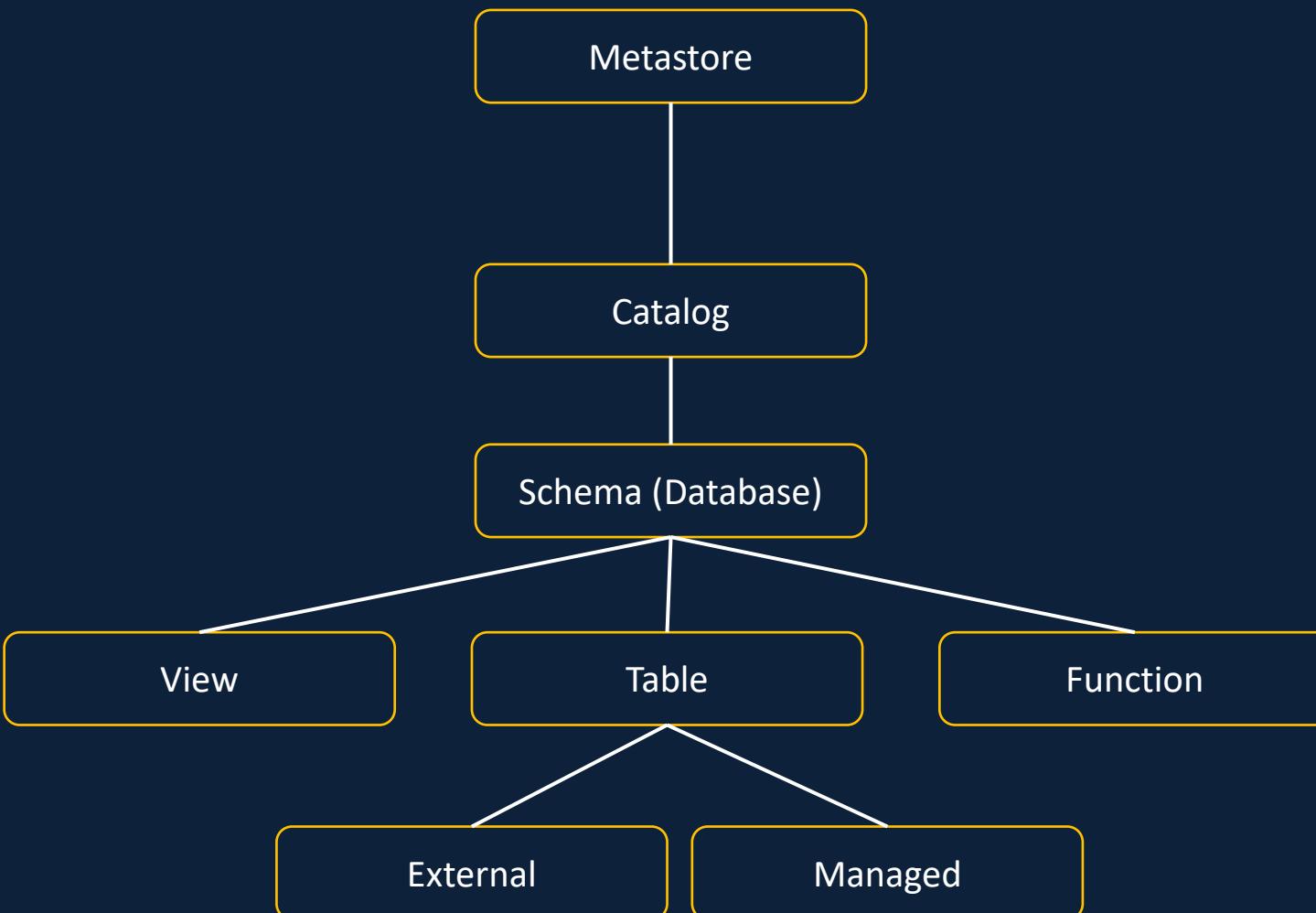
# Security Model – Access Control List



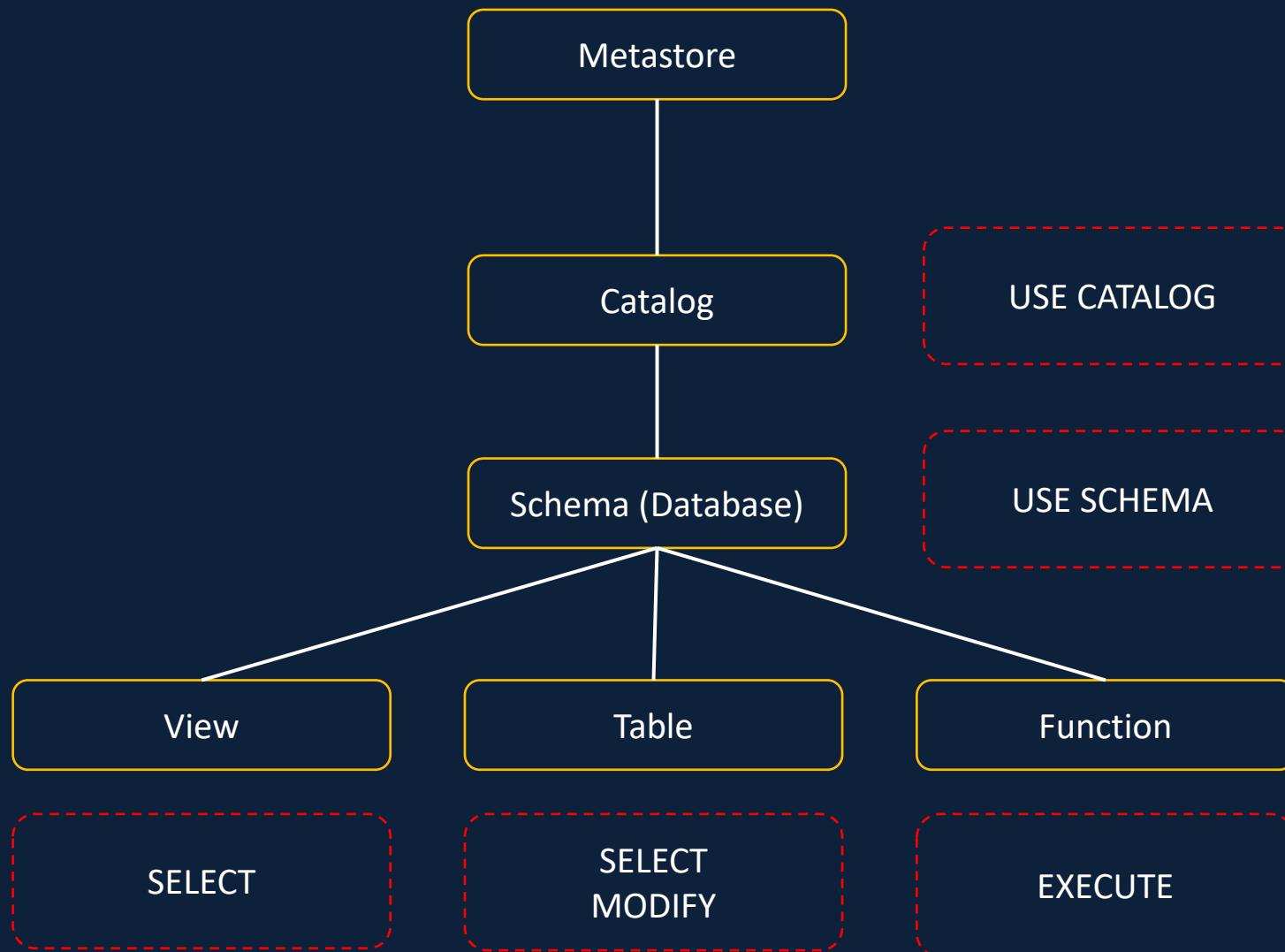
# Security Model – Access Control List



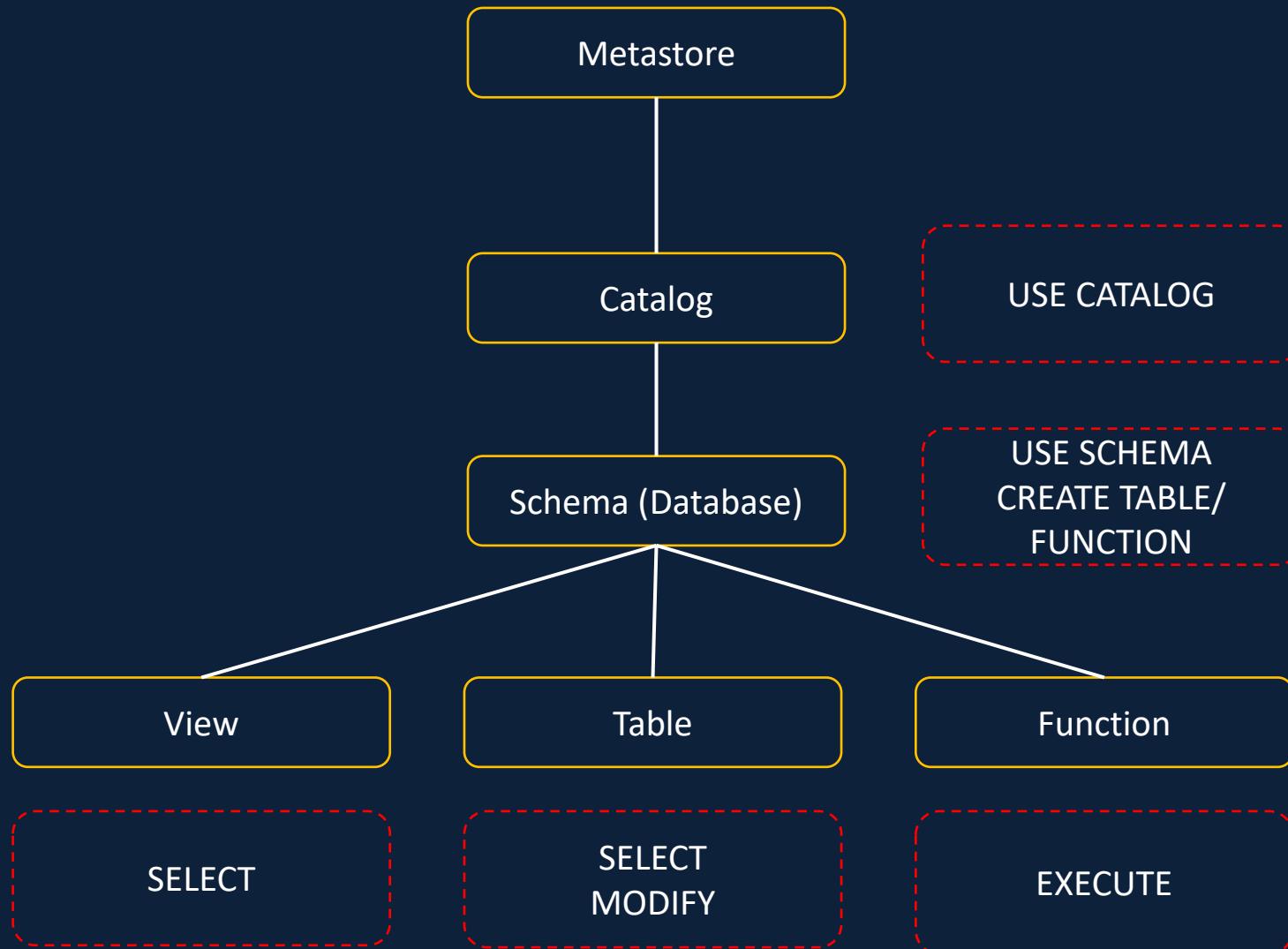
# Security Model – Access Control List



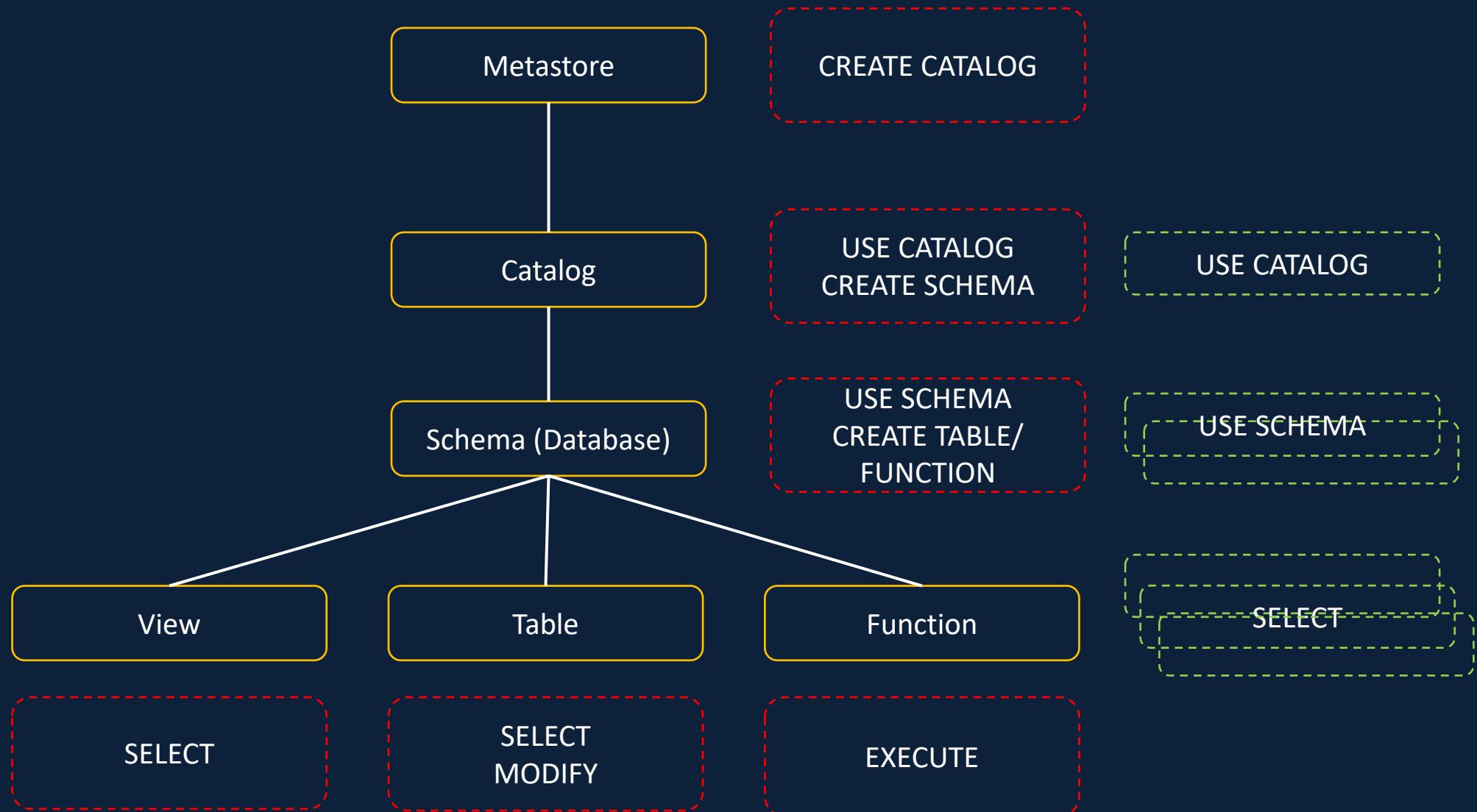
# Security Model – Access Control List



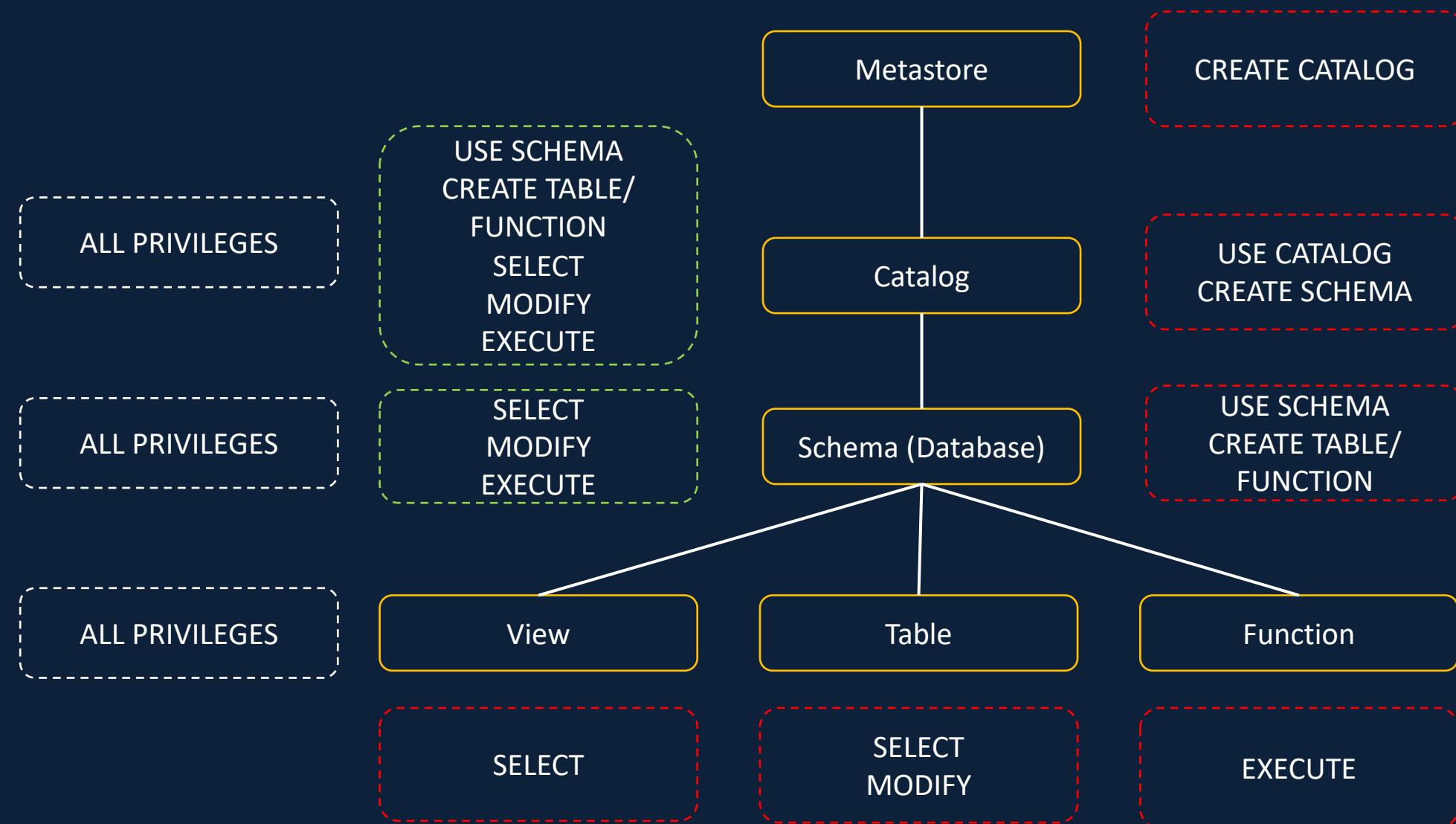
# Security Model – Access Control List



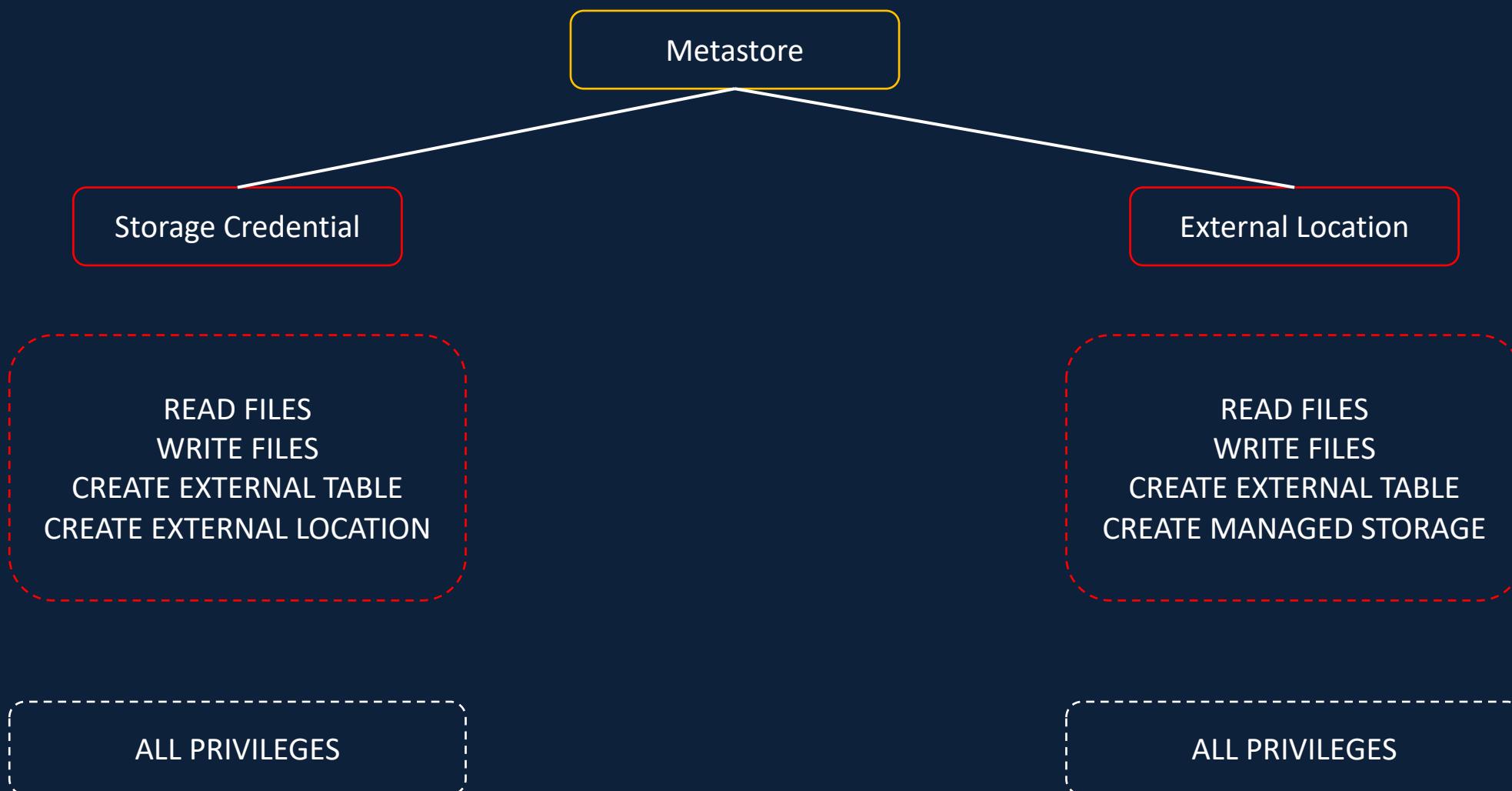
# Security Model – Access Control List



# Access Control List – Inheritance Model



# Security Model – Access Control List



Congratulations!

&

Thank you

# Feedback

# Ratings & Review

Thank you  
&  
Good Luck!

# Document History

Version	Date	Description
1	July 2021	Initial Version
2	Dec 2022	Sections 3 to 5 updated for UI Changes
3	Mar 2023	Revamped Databricks Mounts and Access to ADLS sections as a whole
4	May 2023	Addition of Unity Catalog