DeepLearning.AI

Data Storage and Queries

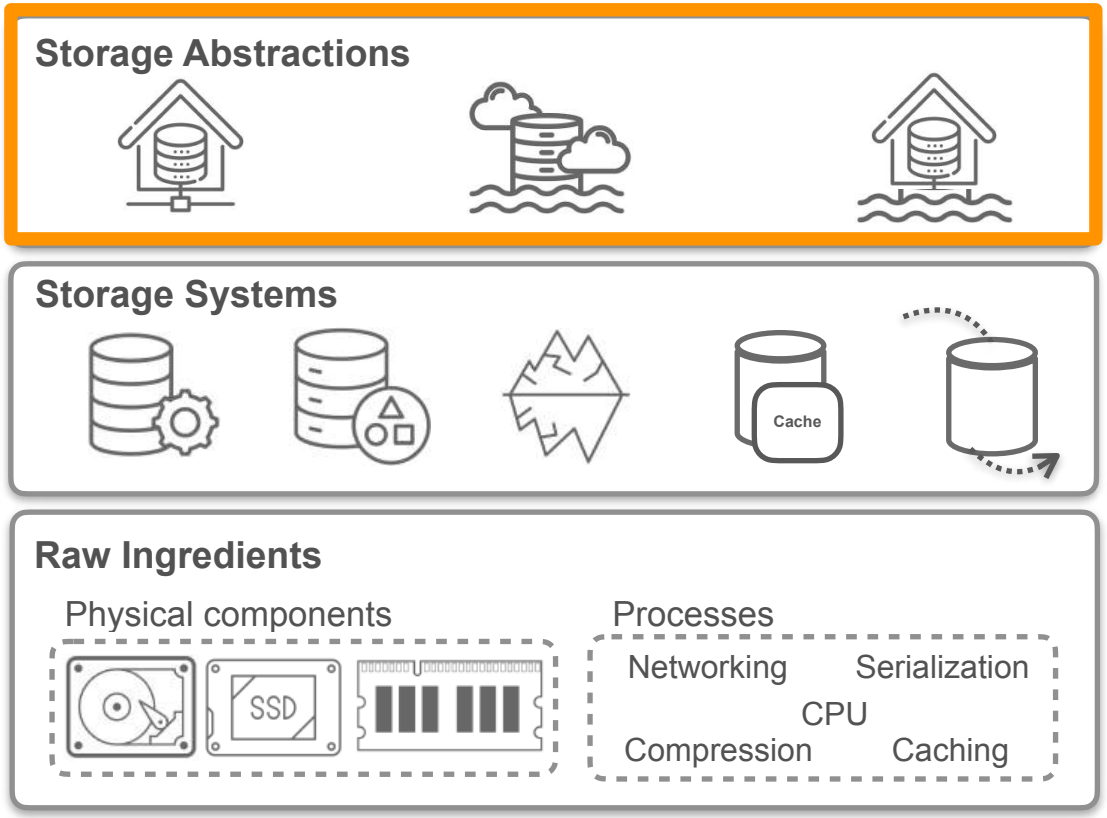
Storage Abstractions

DeepLearning.AI

Storage Abstractions

Week 2 Overview

Storage Hierarchy



Storage Hierarchy

Data Warehouse

Cloud data warehouse

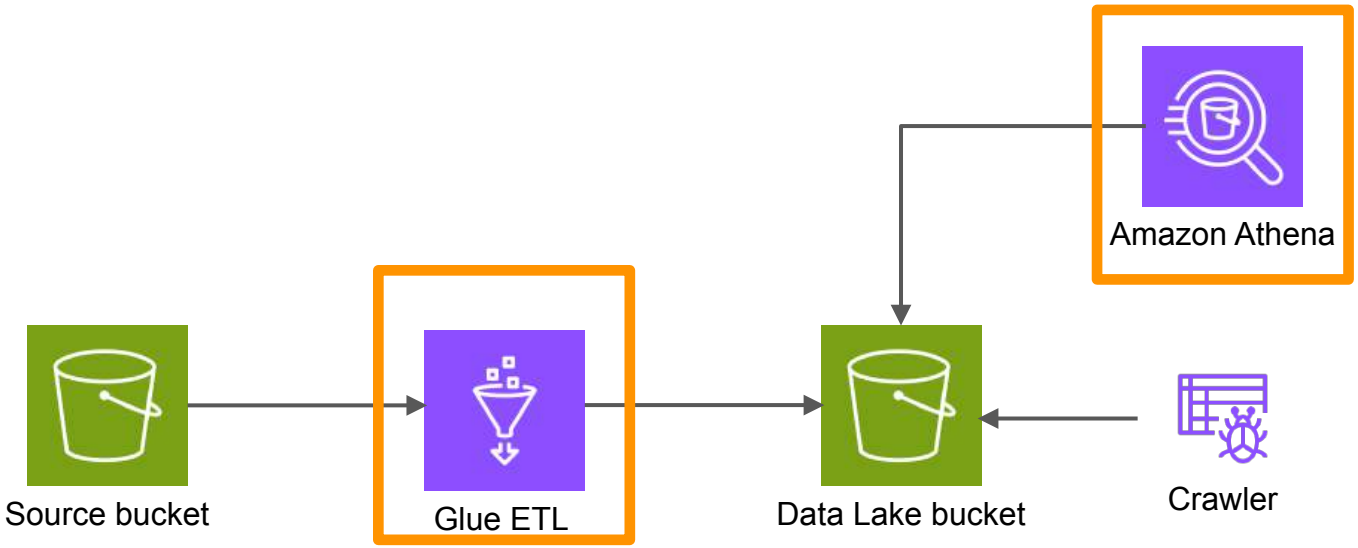
Data Lake

Supports growing storage needs

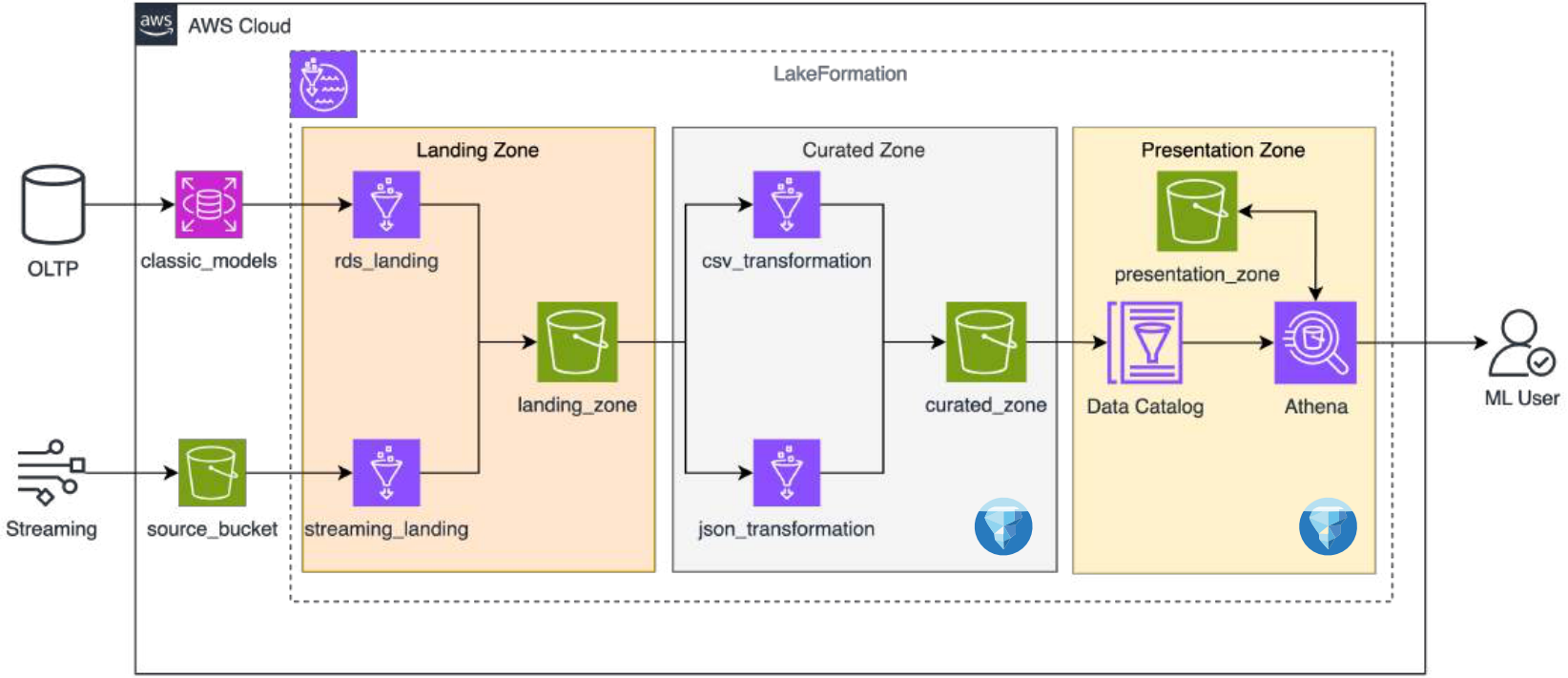
Data Lakehouse

Combines the advantages of data warehouses and data lakes

Week 2 Lab - Simple Data Lake



Week 2 Lab - Building a Data Lakehouse



ICEBERG

DeepLearning.AI

Storage Abstractions

Data Warehouse - Key Architectural ideas

Data Warehouse

Bill Inmon

"Father" of the
Data Warehouse

Data Warehouse:

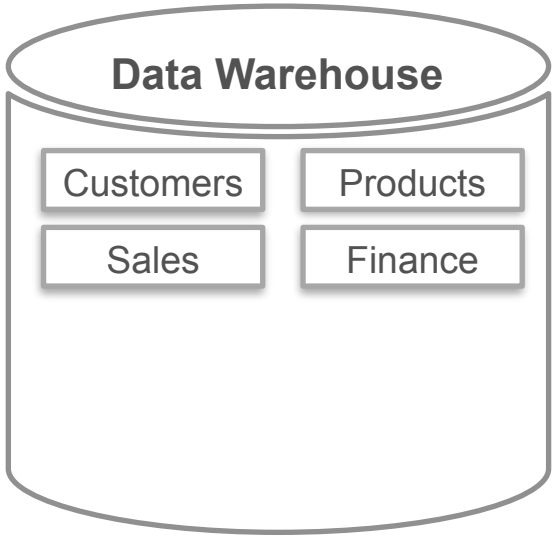
A subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions.

Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)



Data Warehouse

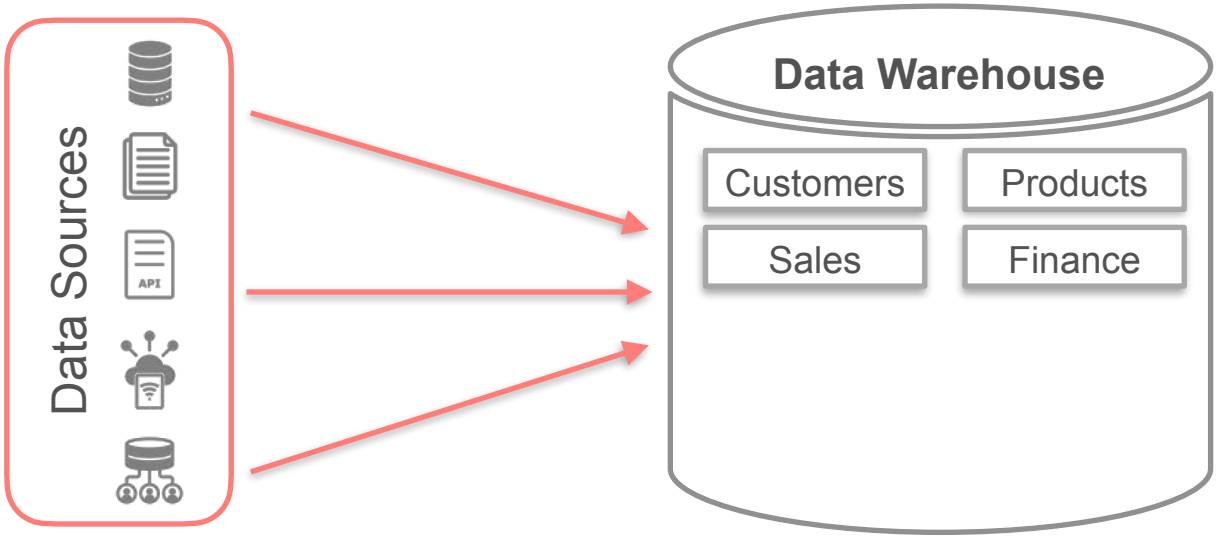
Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format



Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

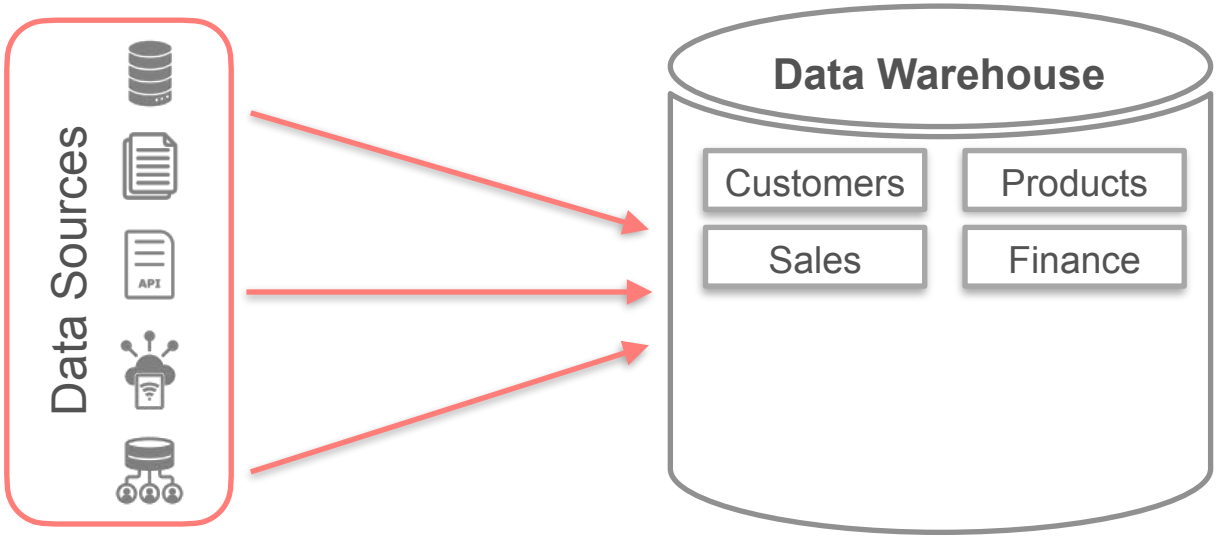
(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile

Data is read-only and cannot be deleted or updated



Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

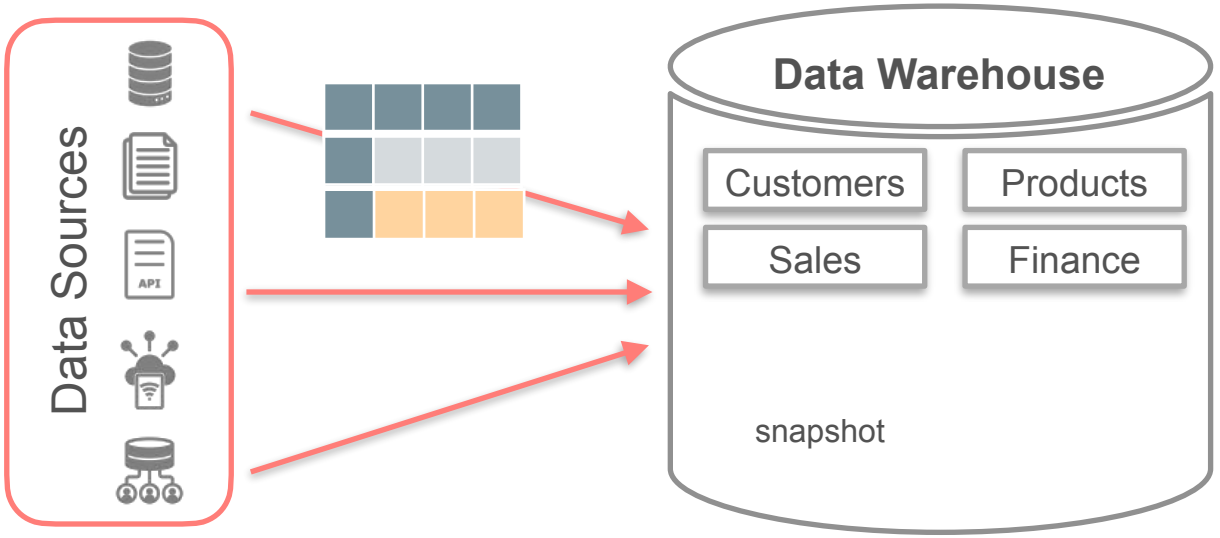
(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile

Data is read-only and cannot be deleted or updated



Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

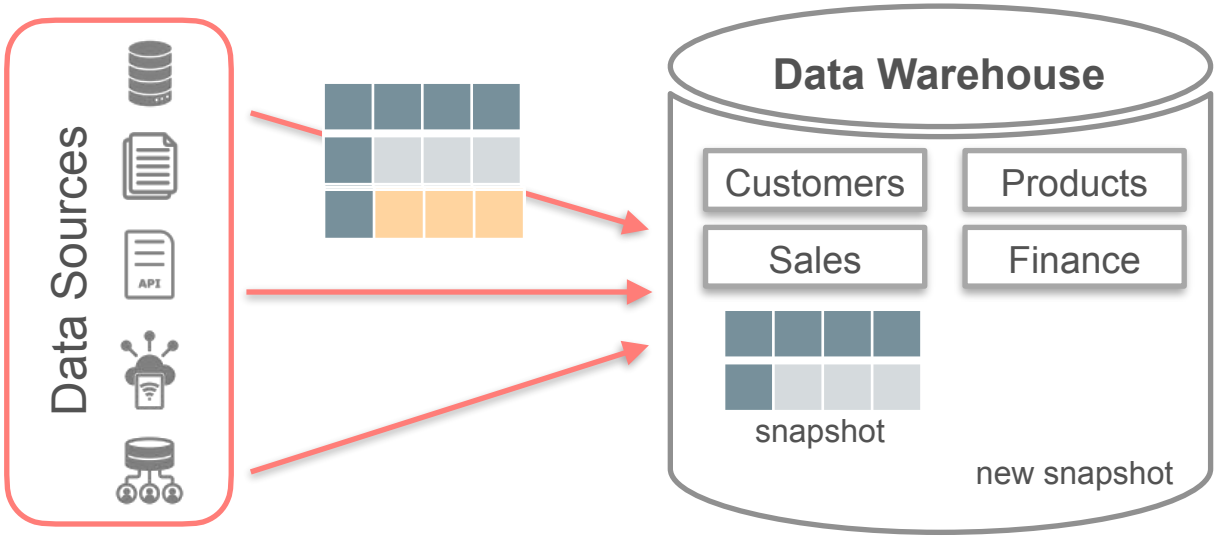
(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile

Data is read-only and cannot be deleted or updated



Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

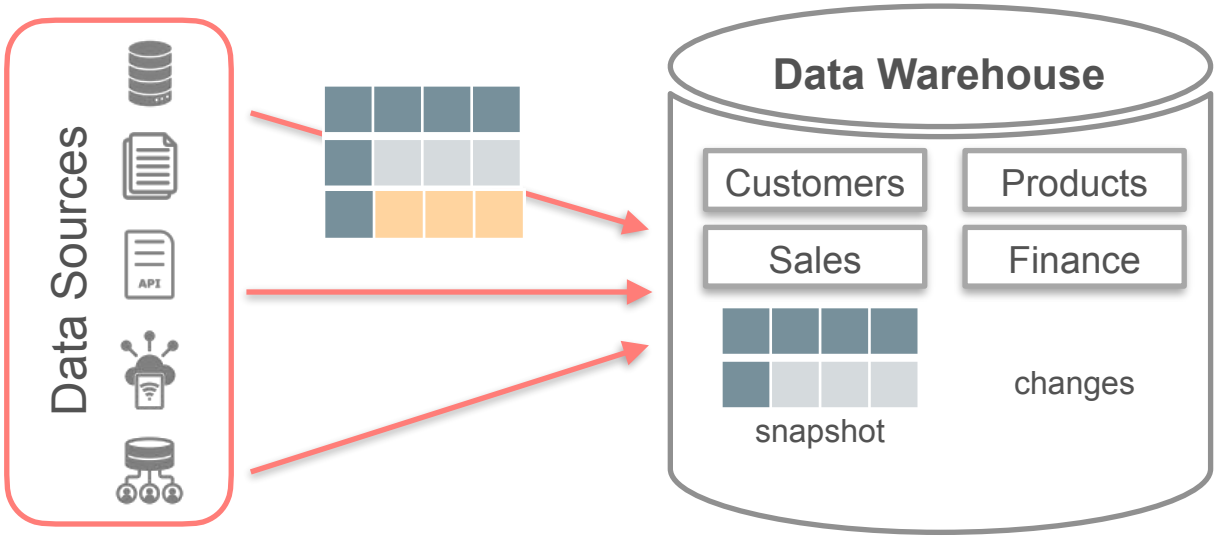
(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile

Data is read-only and cannot be deleted or updated



Data Warehouse

Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

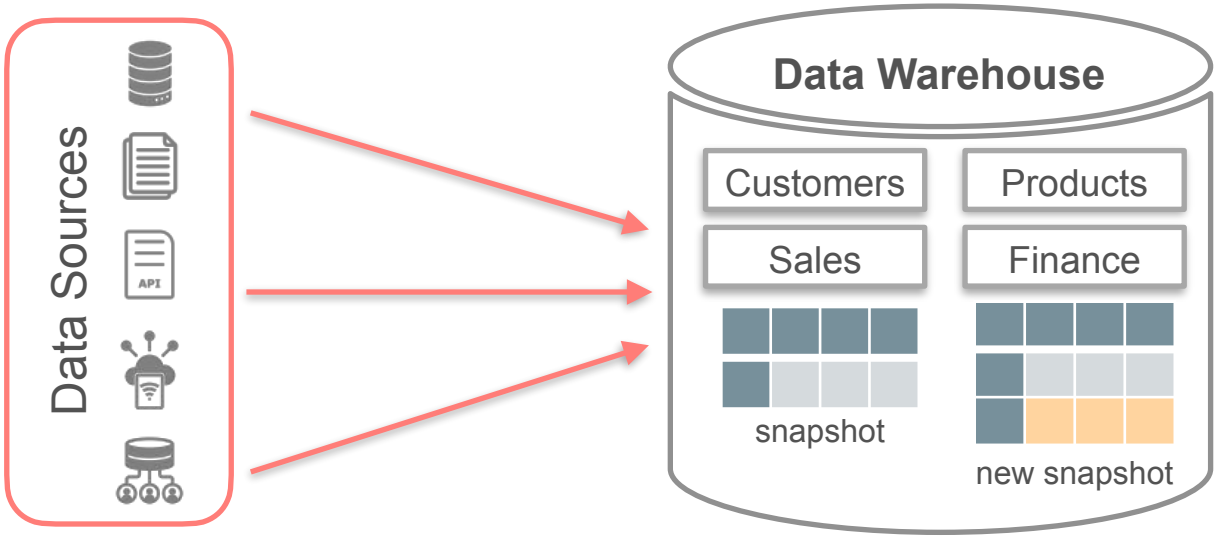
Combines data from different sources into a consistent format

Nonvolatile

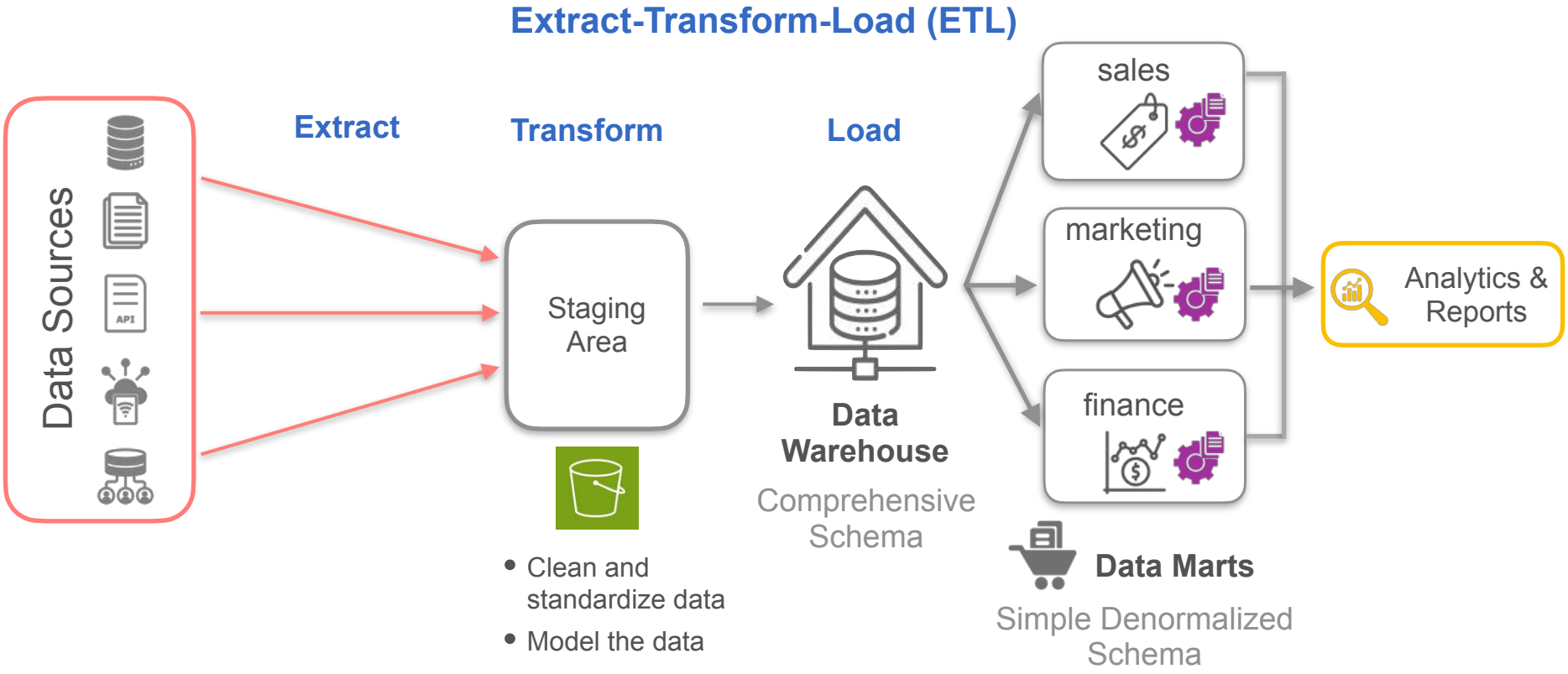
Data is read-only and cannot be deleted or updated

Time-variant

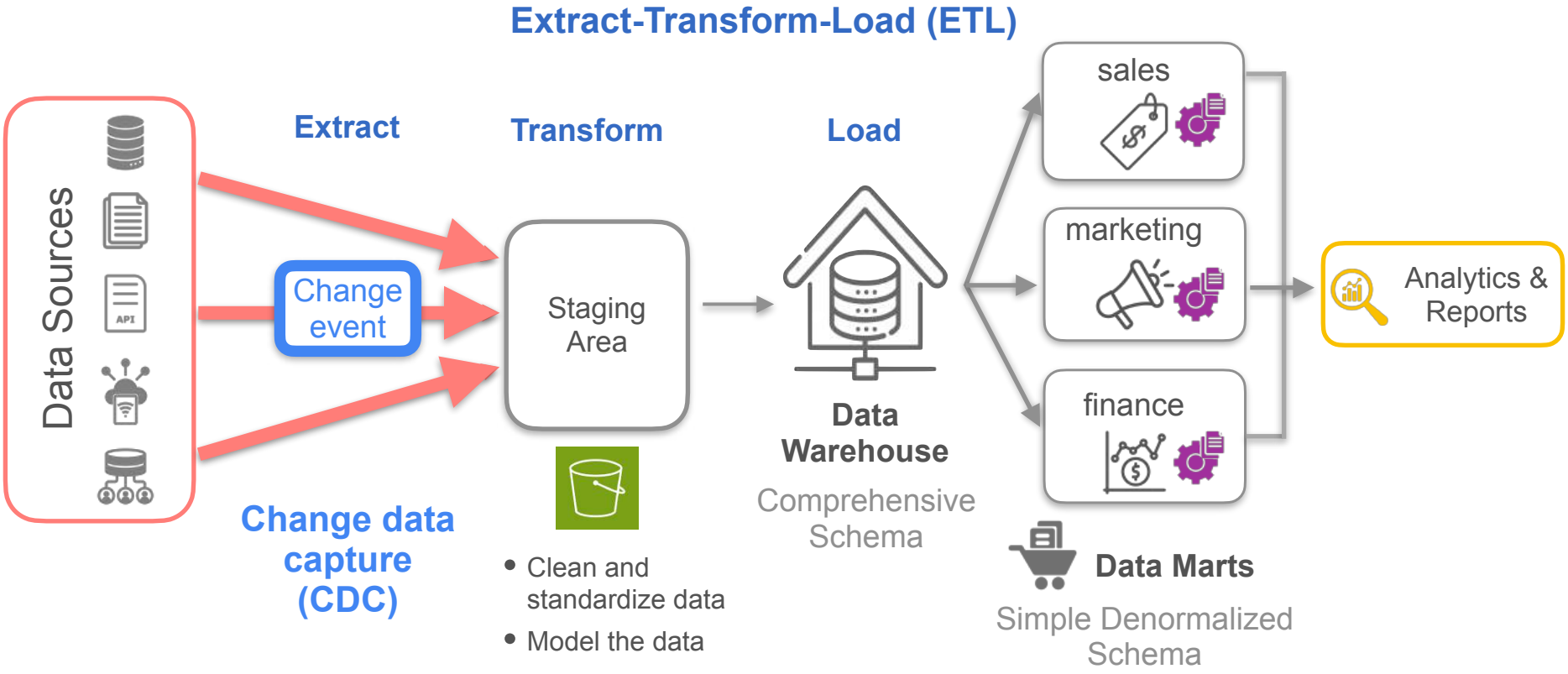
Stores current and historical data (Unlike OLTP systems)



Data Warehouse-Centric Architecture



Change Data Capture

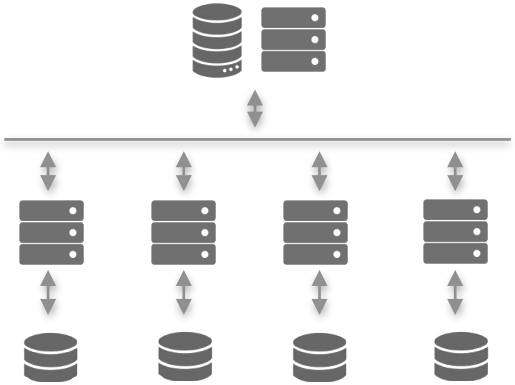


Data Warehouse Implementation

Early Data Warehouses

Big monolithic server

Data Warehouses with Massively Parallel Processing (MPP)



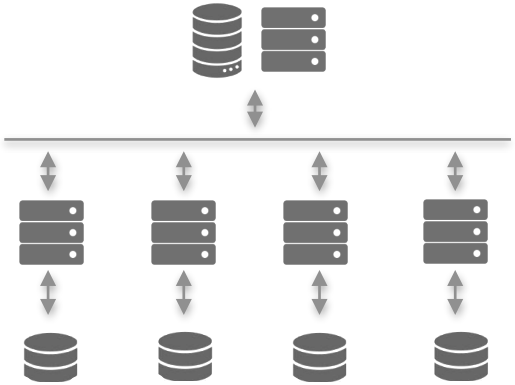
- Scans large amounts of data in parallel
- Complex configurations and requires effort to maintain

Data Warehouse Implementation

Early Data Warehouses

Big monolithic server

Data Warehouses with Massively Parallel Processing (MPP)



- Scans large amounts of data in parallel
- Complex configurations and requires effort to maintain

Modern Cloud Data Warehouses

Amazon Redshift

Google Big Query

snowflake

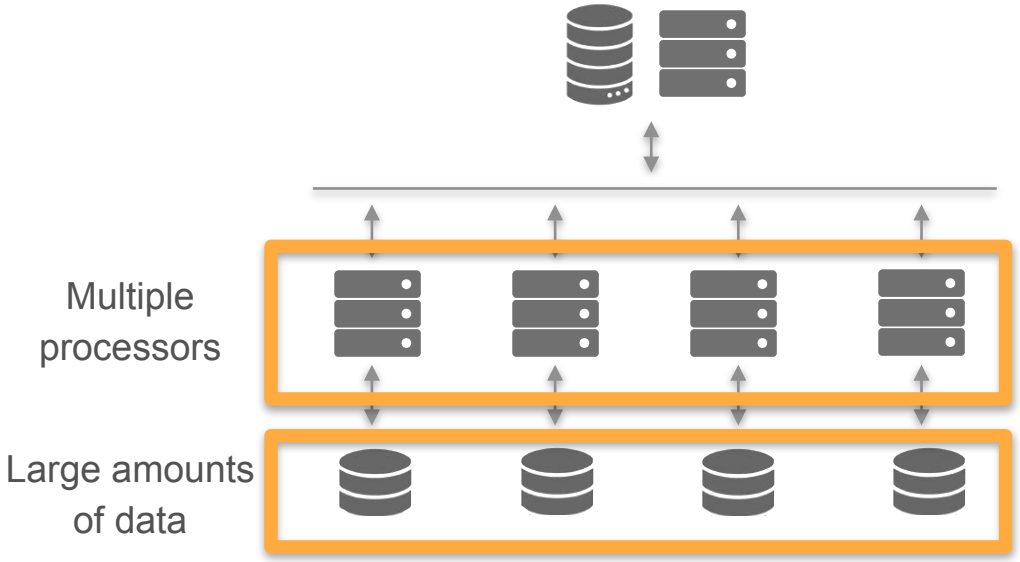
- Separates compute from storage
- Expands the capability of MPP systems

DeepLearning.AI

Storage Abstractions

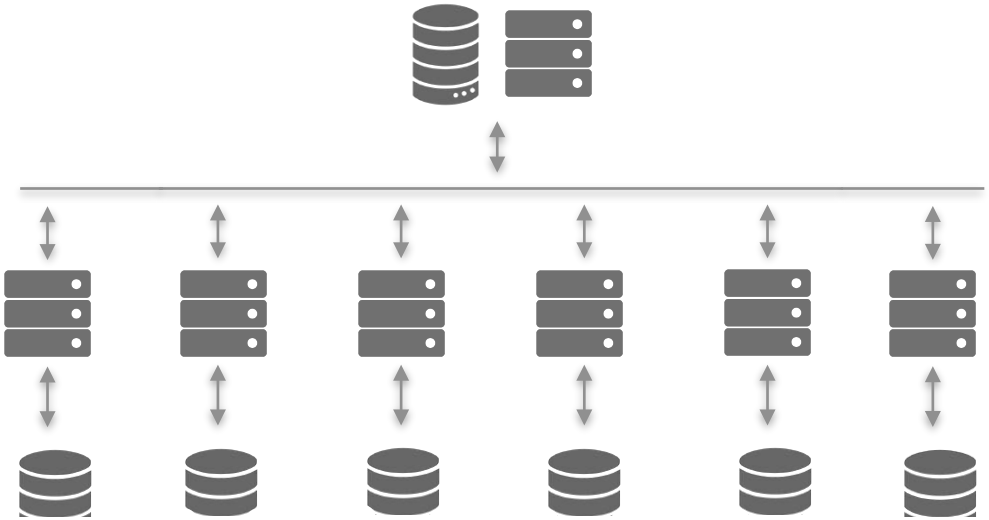
Modern Cloud Data Warehouses

Massively Parallel Processing



Massively Parallel Processing (MPP)

Massively Parallel Processing - Cloud Data Warehouses



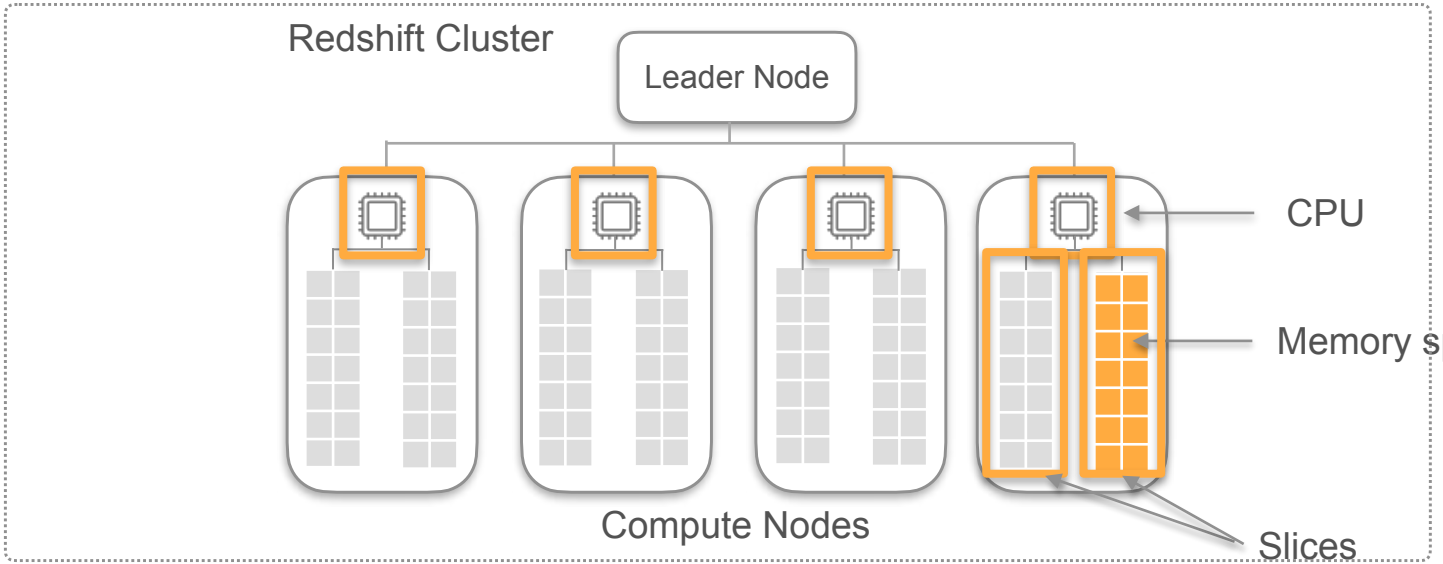
Massively Parallel Processing (MPP)

Amazon Redshift

Google Big Query

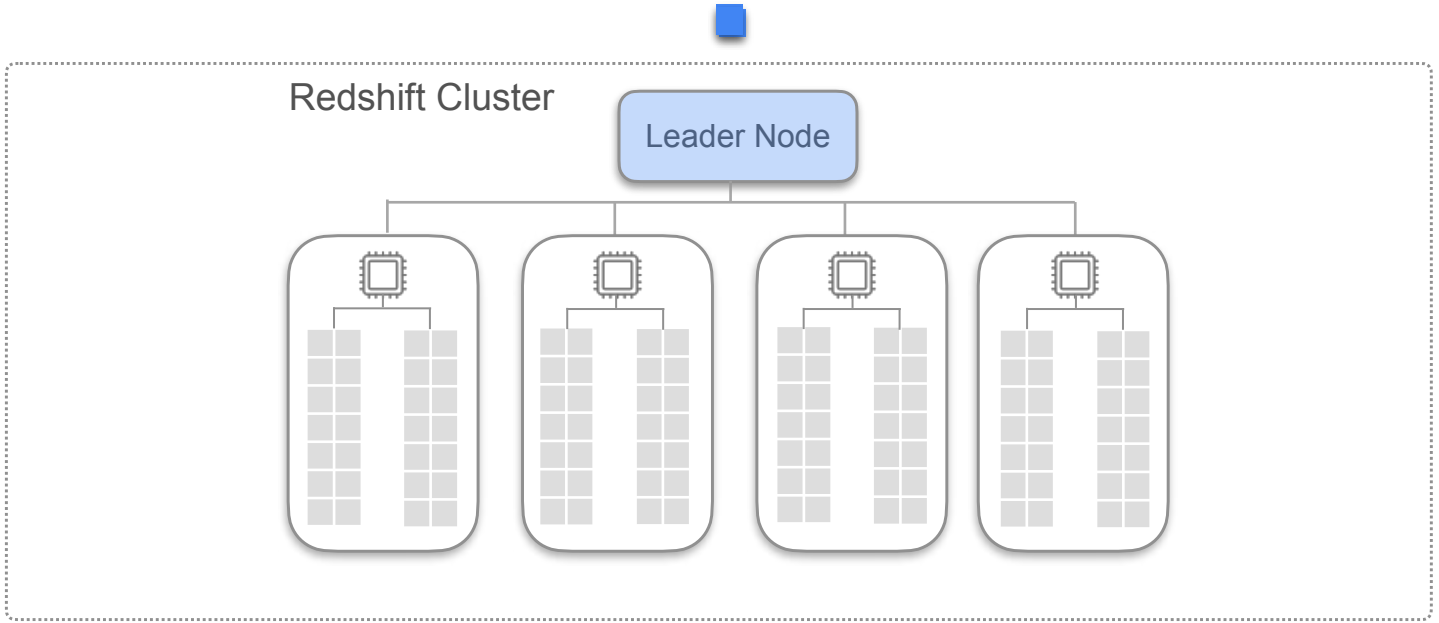
snowflake

MPP Architecture for Amazon Redshift



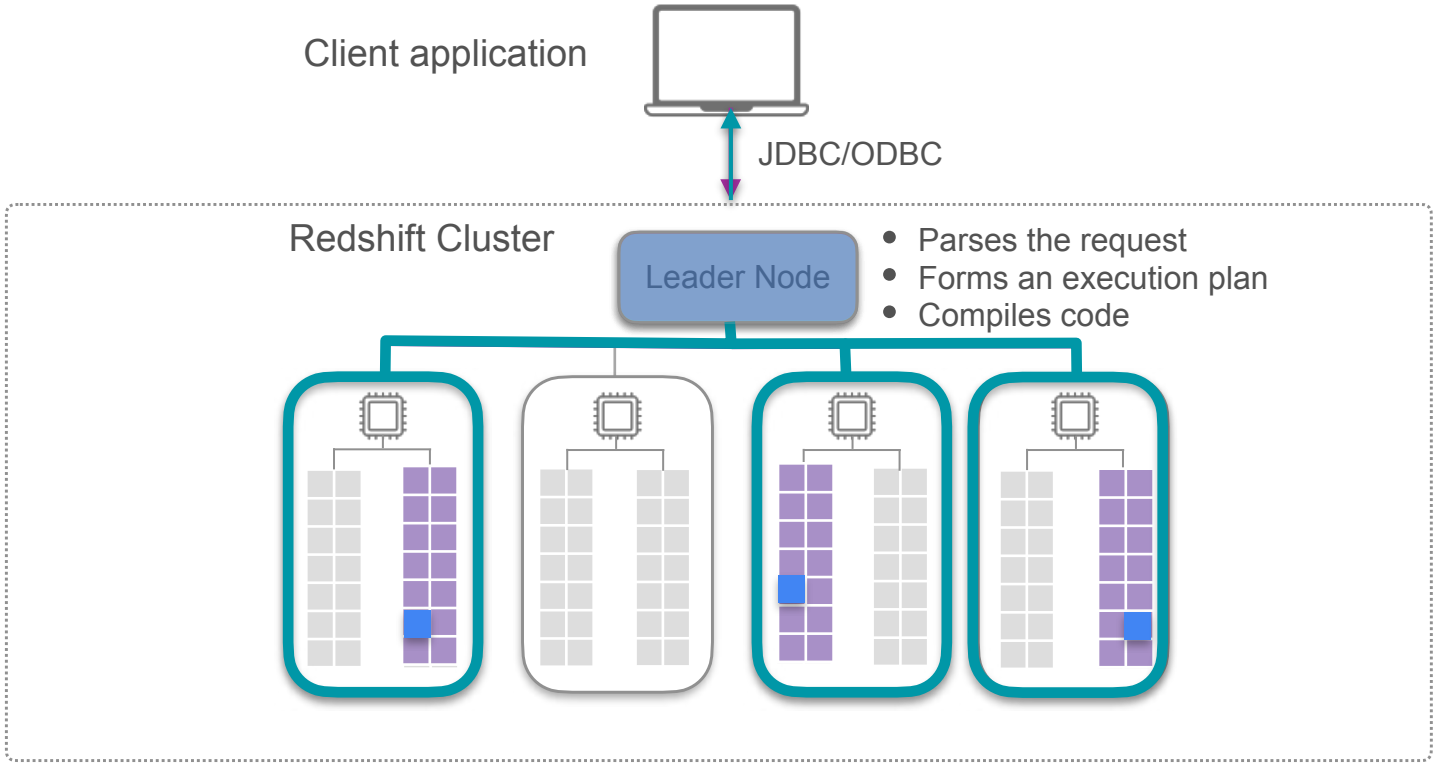
Amazon Redshift

MPP Architecture for Amazon Redshift



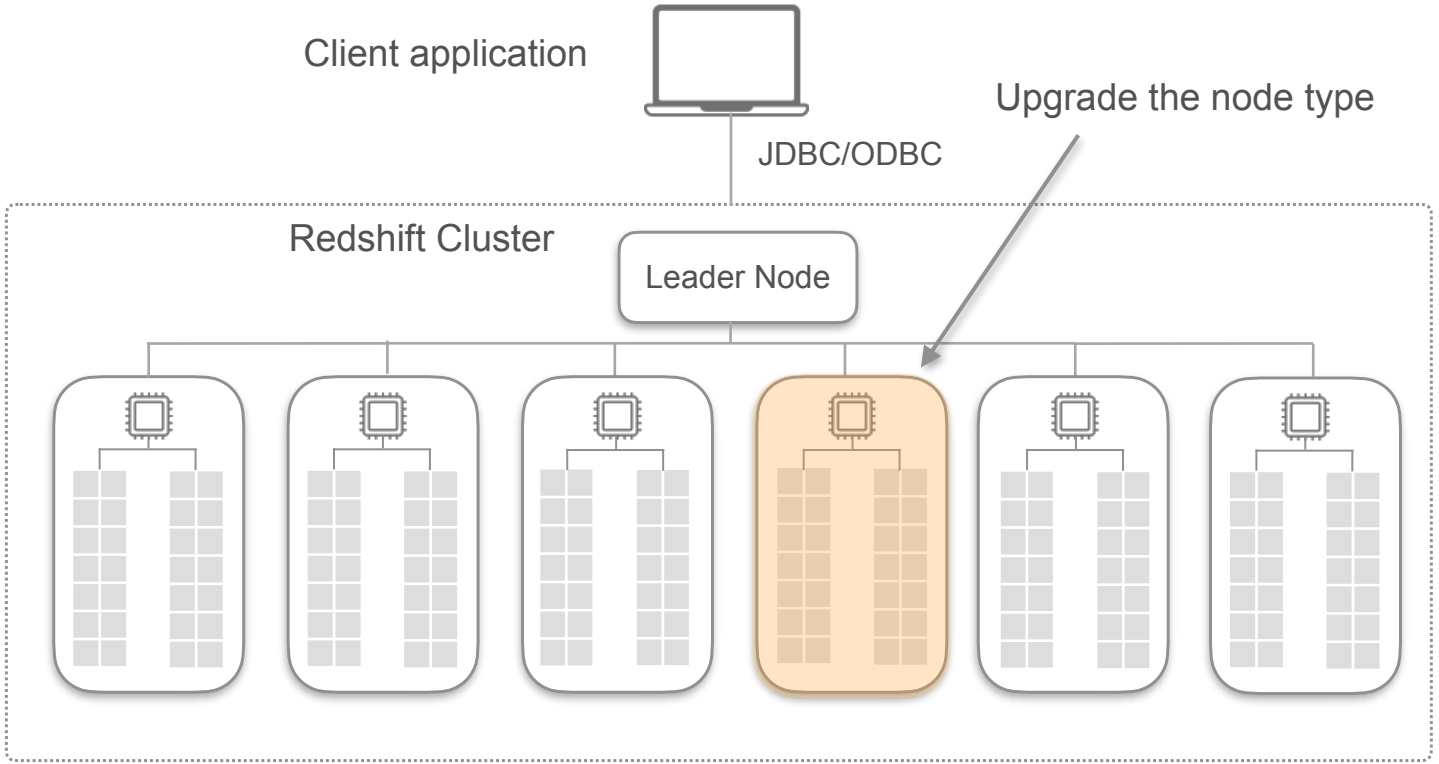
Amazon Redshift

MPP Architecture for Amazon Redshift



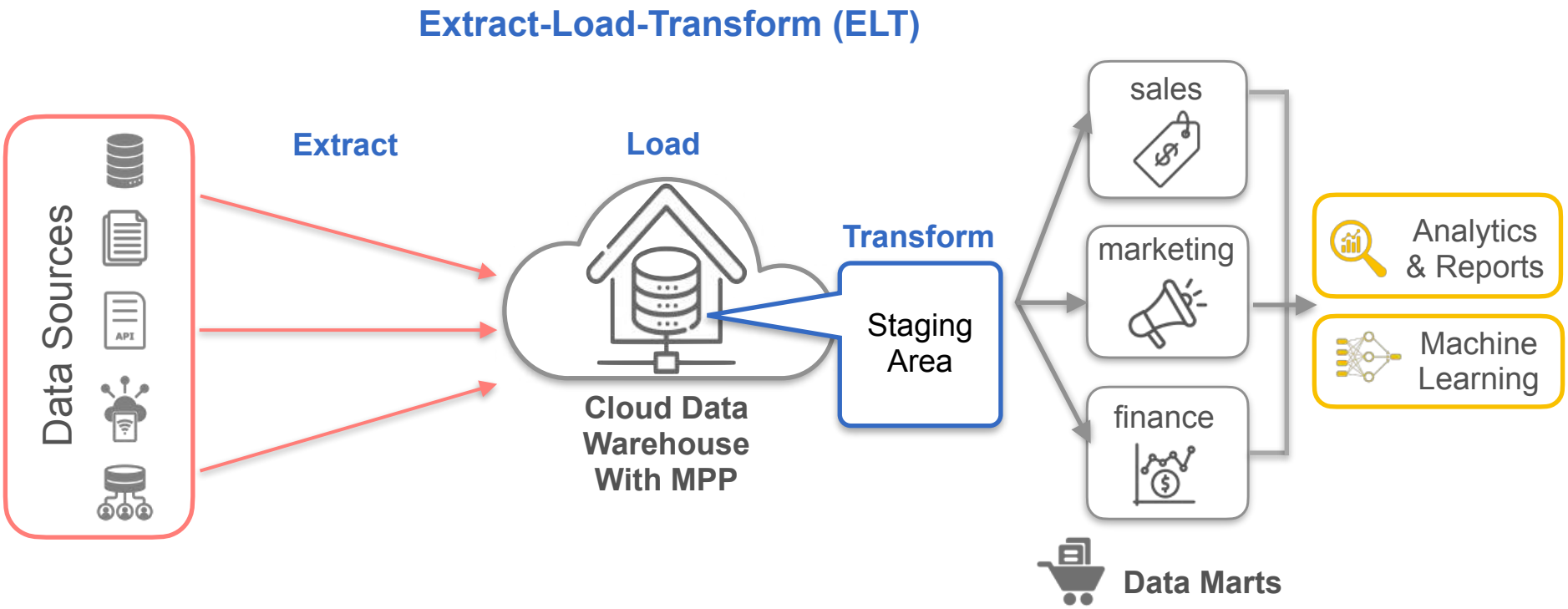
Amazon Redshift

MPP Architecture for Amazon Redshift



Amazon Redshift

Data Warehouse-Centric Architecture



Cloud Data Warehouse

Columnar Architecture

Order ID	Price	Product SKU	Quantity	Customer ID
1	40	458650	10	67t
2	23	902348	14	56t
3	45	1255893	12	87q
4	50	456829	13	98q

Facilitates high performance analytical queries

Separation of Compute and Storage

Compute

Storage

Cloud Data Warehouse

Traditional Data Warehouse	Cloud Data Warehouse
• Stored data is highly structured • Data modeled to enable analytical queries	• Stored data is highly structured • Data modeled to enable analytical queries • High processing from MPP • Columnar Storage • Separation of storage and compute **Efficiently stores and processes data for high-volume analytical workloads**

DeepLearning.AI

Storage Abstractions

Data Lakes - Key Architectural ideas

Data Lake

- Central repository for storing large volumes of data
- No fixed schema or predefined set of transformations
- **Schema-on-read pattern:**
  - Reader determines the schema when reading the data

Data Lake 1.0

Combined different storage and processing technologies

Storage

Amazon S3

Processing Tools

Apache Pig

Shortcomings of Data Lake 1.0

Data Swamp

- No proper data management
- No data cataloging
- No data discovery tools
- No guarantee on the data integrity and quality

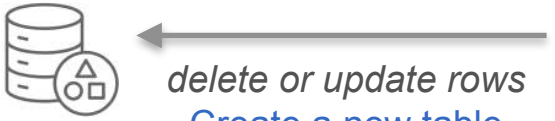
Shortcomings of Data Lake 1.0

Data Swamp

- No proper data management
- No data cataloging
- No data discovery tools
- No guarantee on the data integrity and quality

Write-only storage

- Data Manipulation Language (DML) operations were painful to implement



- Difficult to comply with data regulations

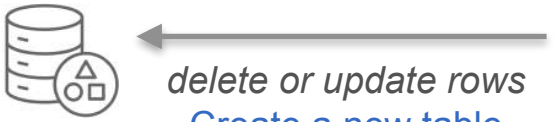
Shortcomings of Data Lake 1.0

Data Swamp

- No proper data management
- No data cataloging
- No data discovery tools
- No guarantee on the data integrity and quality

Write-only storage

- Data Manipulation Language (DML) operations were painful to implement



- Difficult to comply with data regulations

No schema management and data modeling

- Hard to process stored data
- Data not optimized for query operations such as joins

Joins

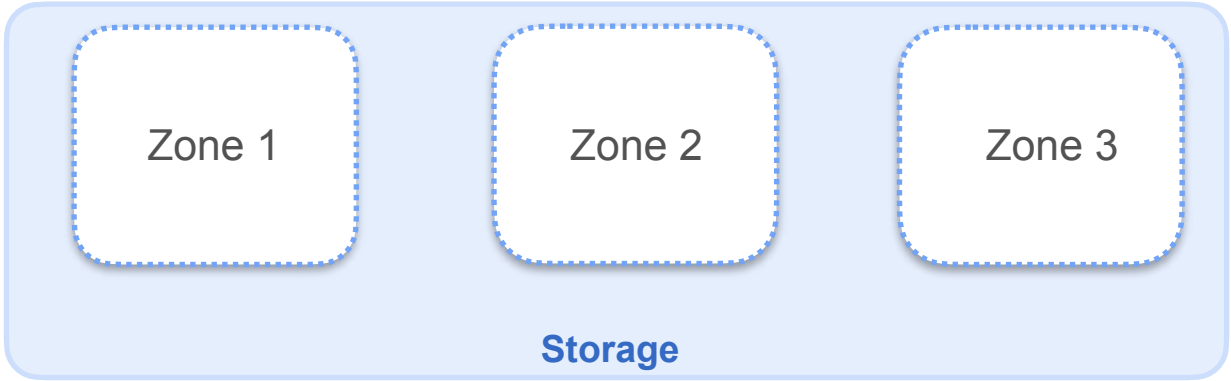
DeepLearning.AI

Storage Abstractions

Next-Generation Data Lakes

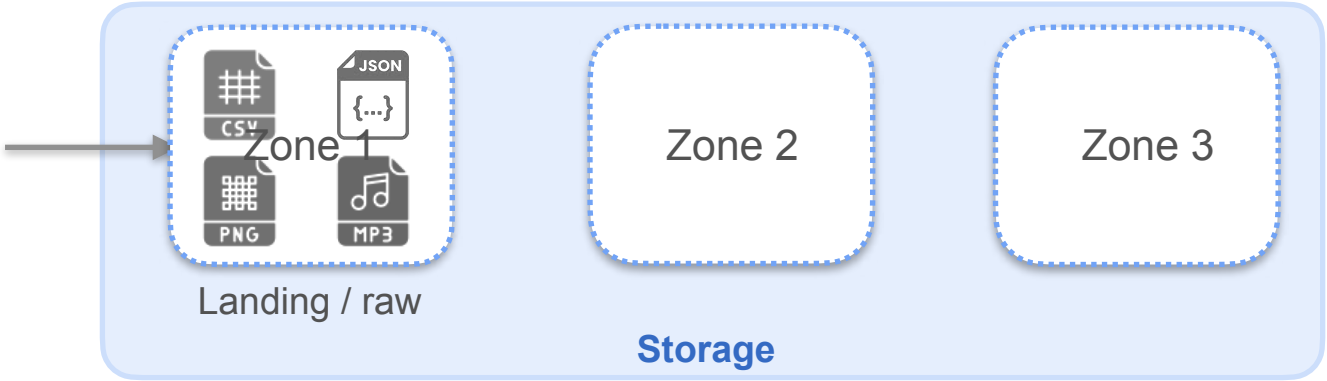
Data Zones

Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees



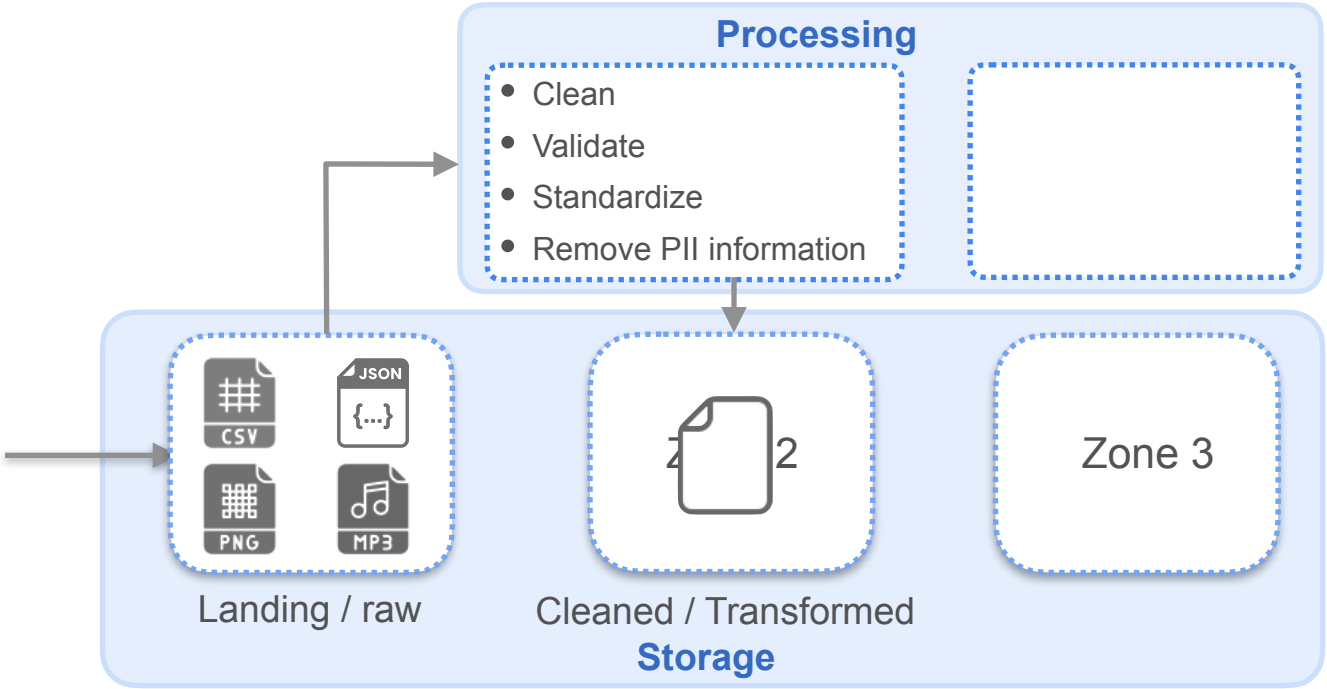
Data Zones

Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees



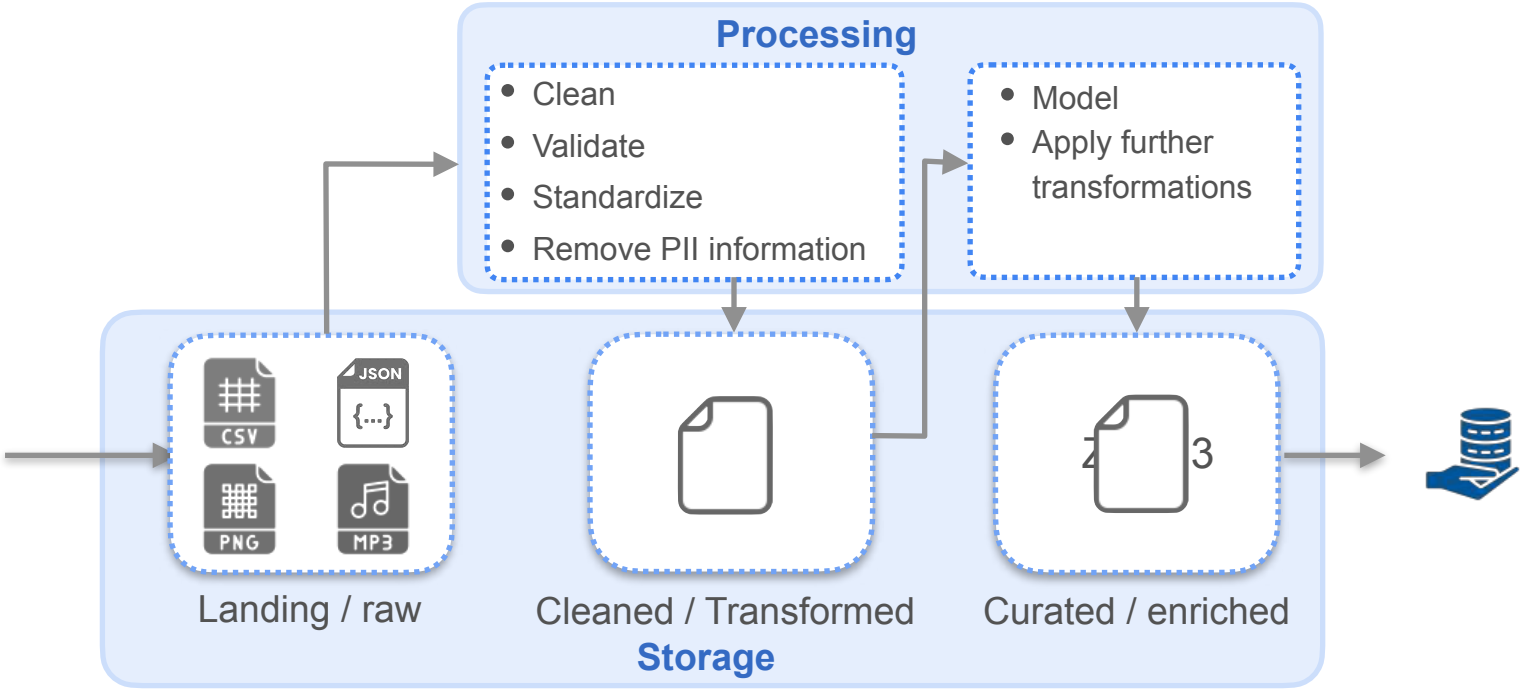
Data Zones

Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees



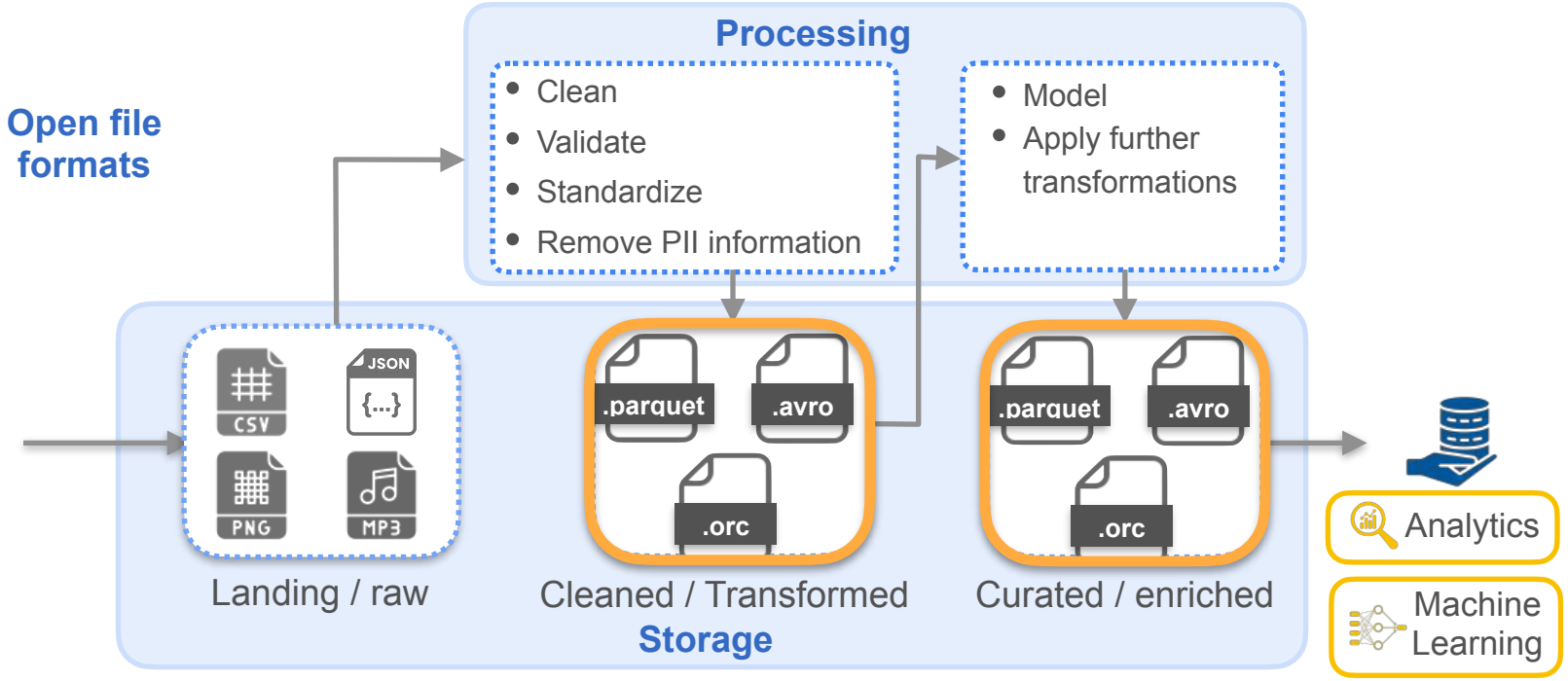
Data Zones

Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees



Data Zones

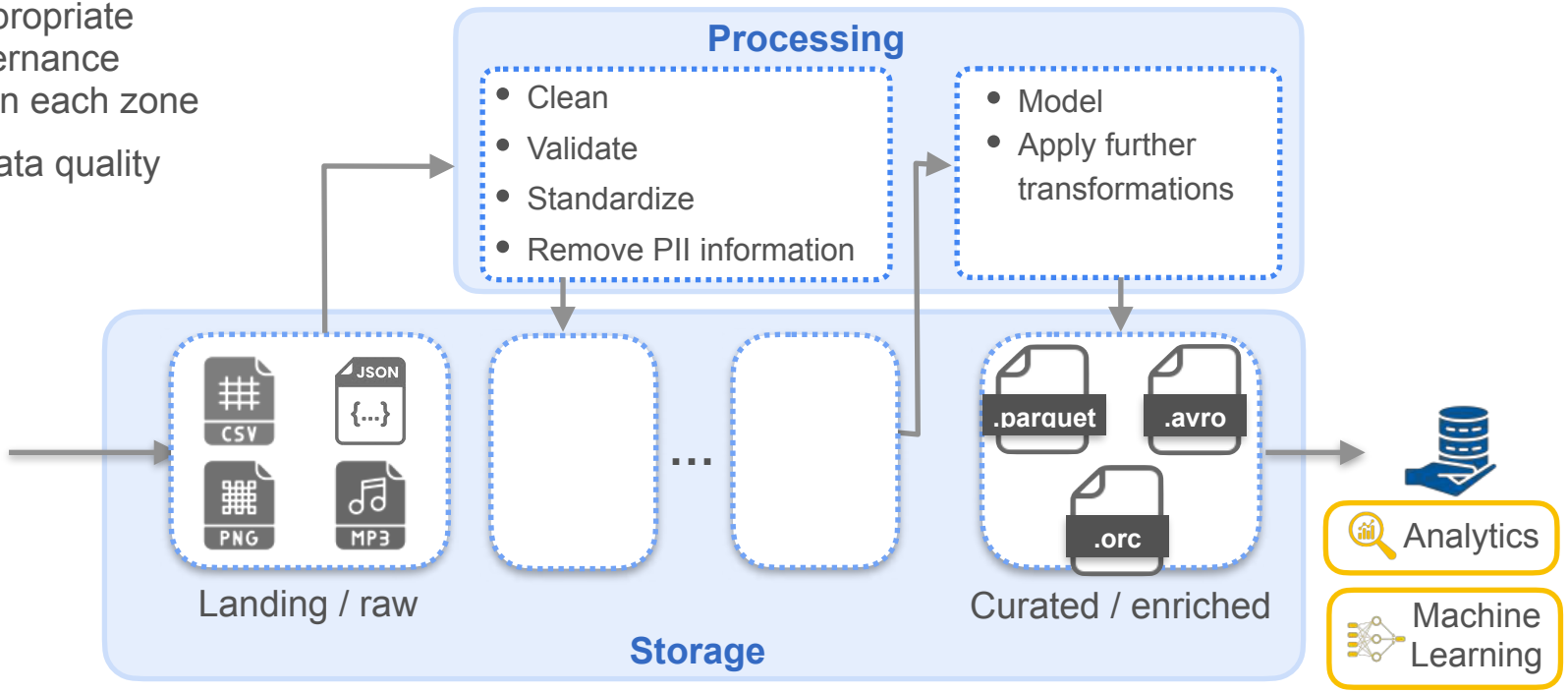
Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees



Data Zones

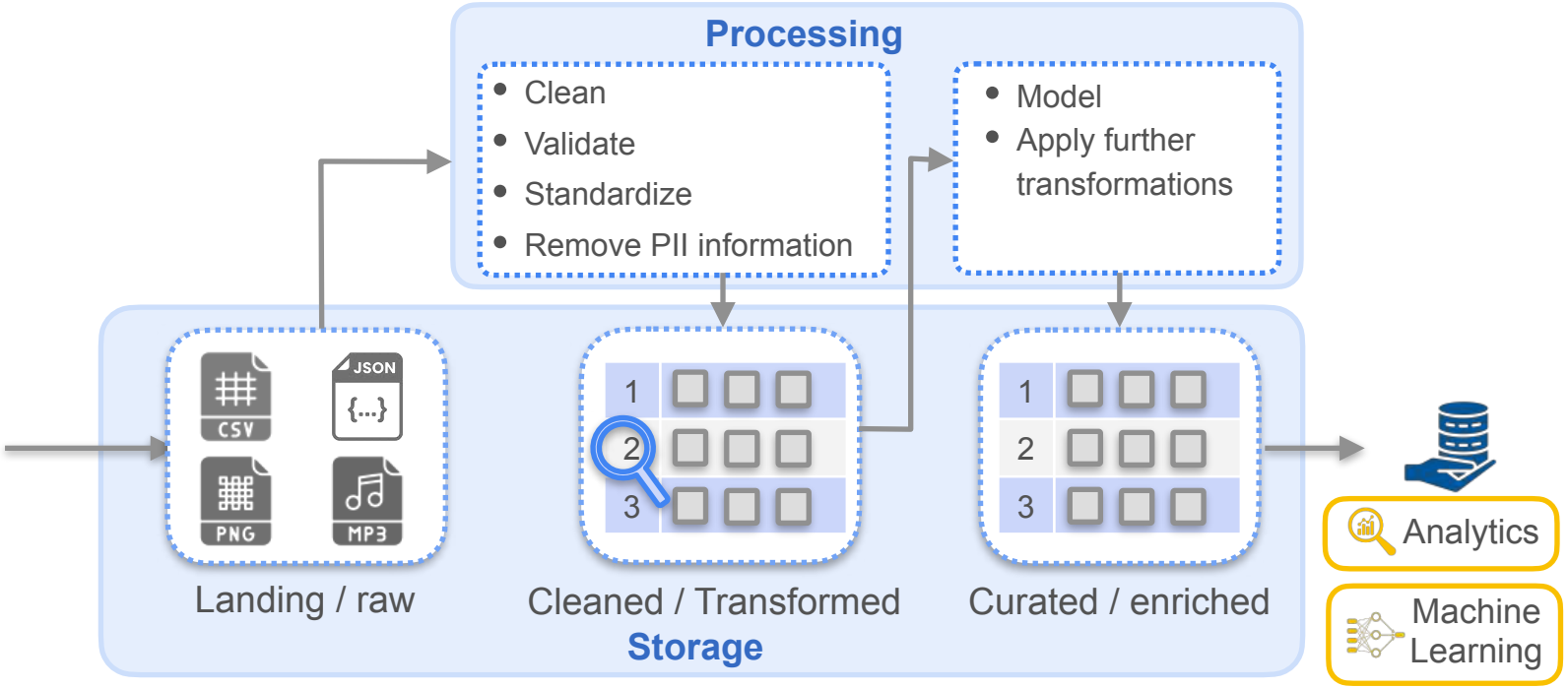
Used to organize data in a data lake, where each zone houses data that has been processed to varying degrees

- Apply appropriate data governance policies on each zone
- Ensure data quality



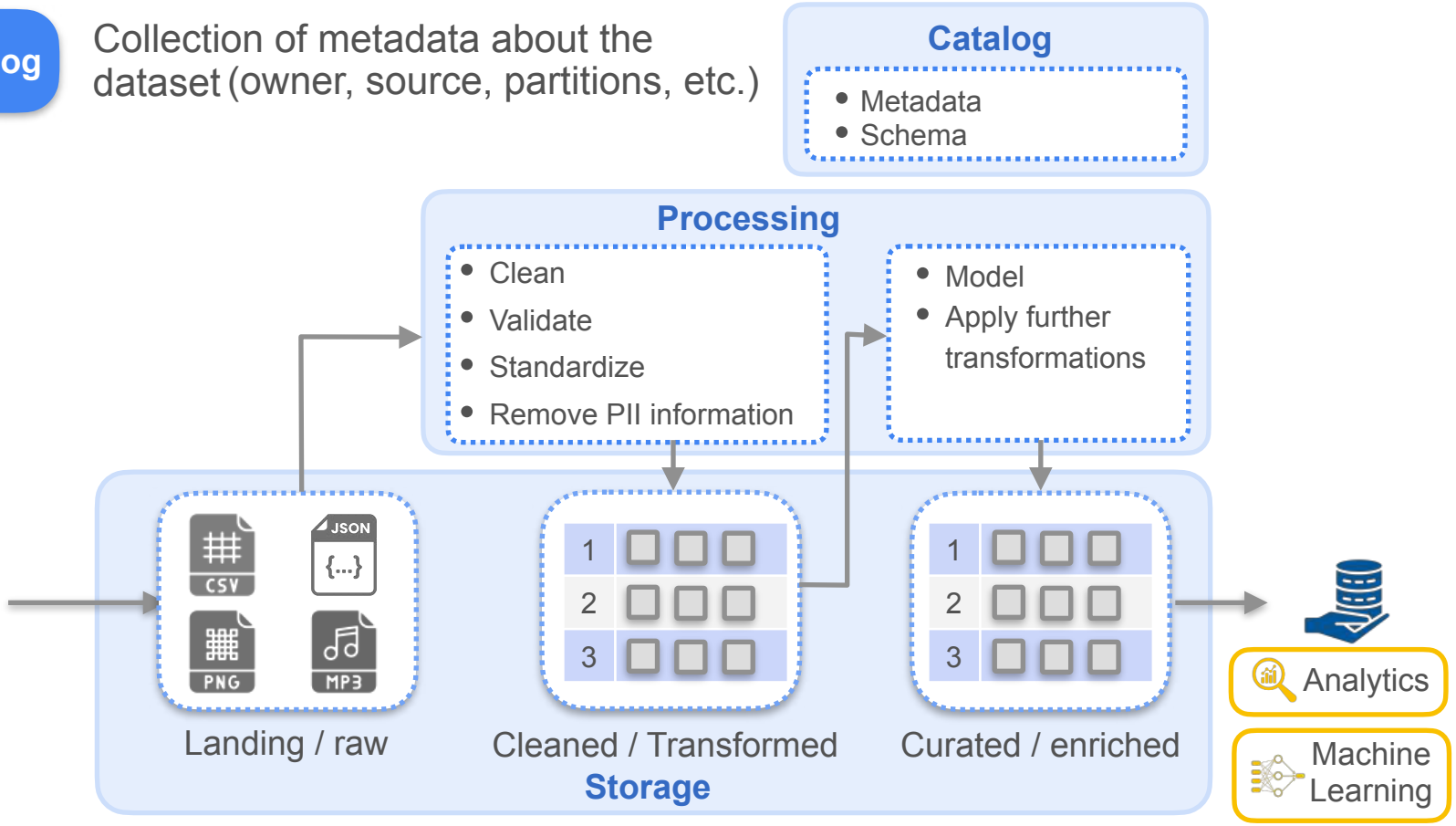
Data Partitioning

Divide a dataset into smaller, more manageable parts based on a set of criteria (e.g. time, date, location recorded in the data)



Data Catalog

Collection of metadata about the dataset (owner, source, partitions, etc.)



Separate Data Lakes and Data Warehouses

Data Lake

Low-Cost Storage

Store large amounts of data

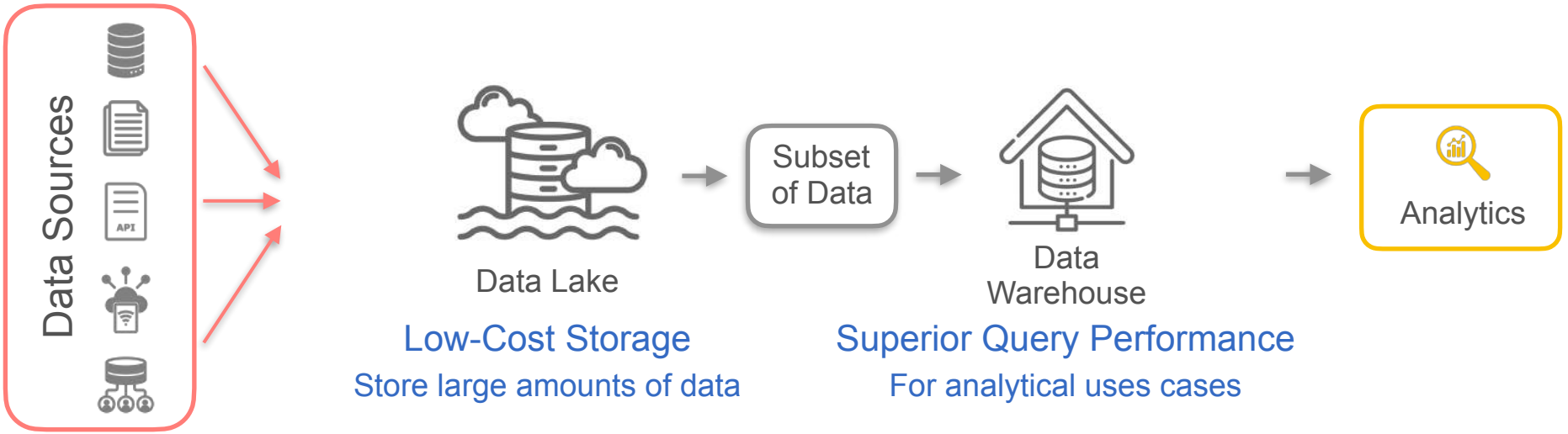
and

Data Warehouse

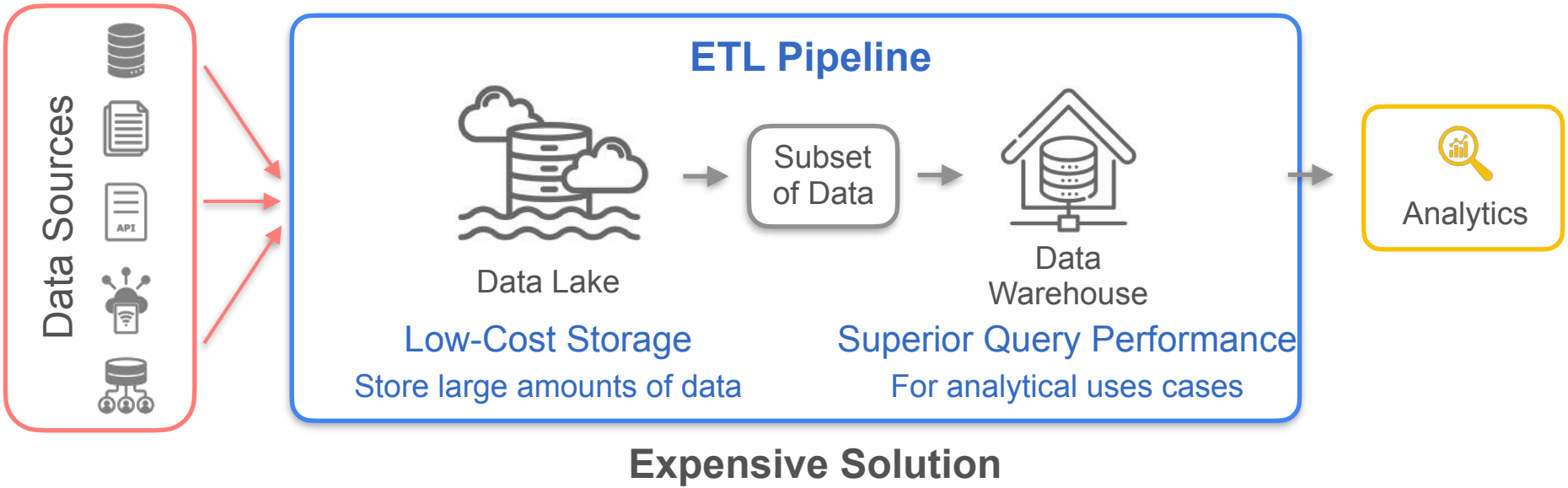
Superior Query Performance

For analytical uses cases

Separate Data Lakes and Data Warehouses

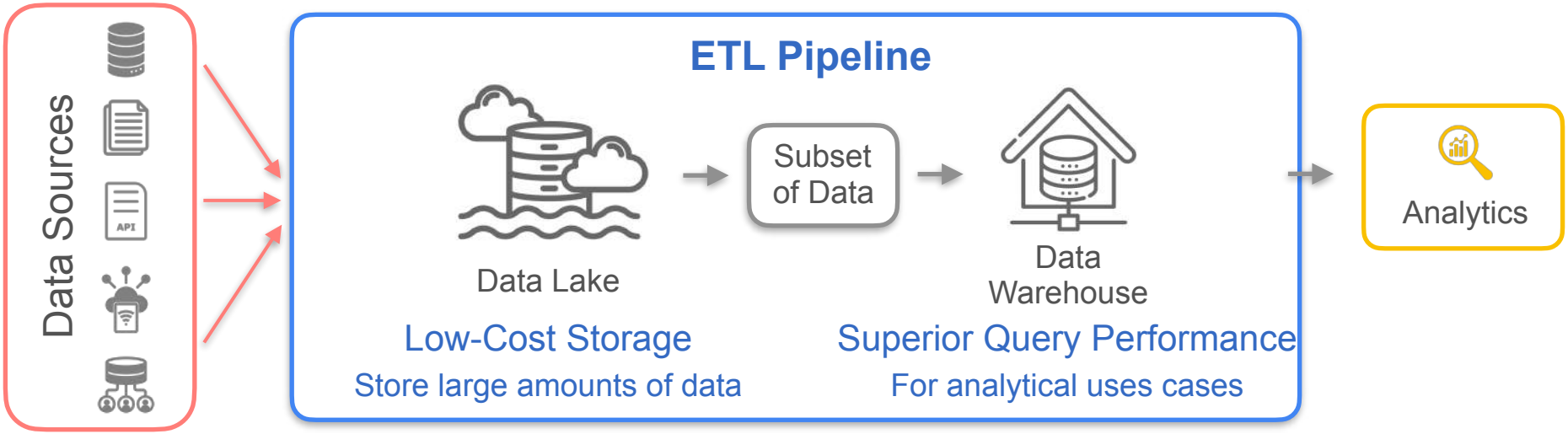


Separate Data Lakes and Data Warehouses



Expensive Solution

Separate Data Lakes and Data Warehouses



Expensive Solution

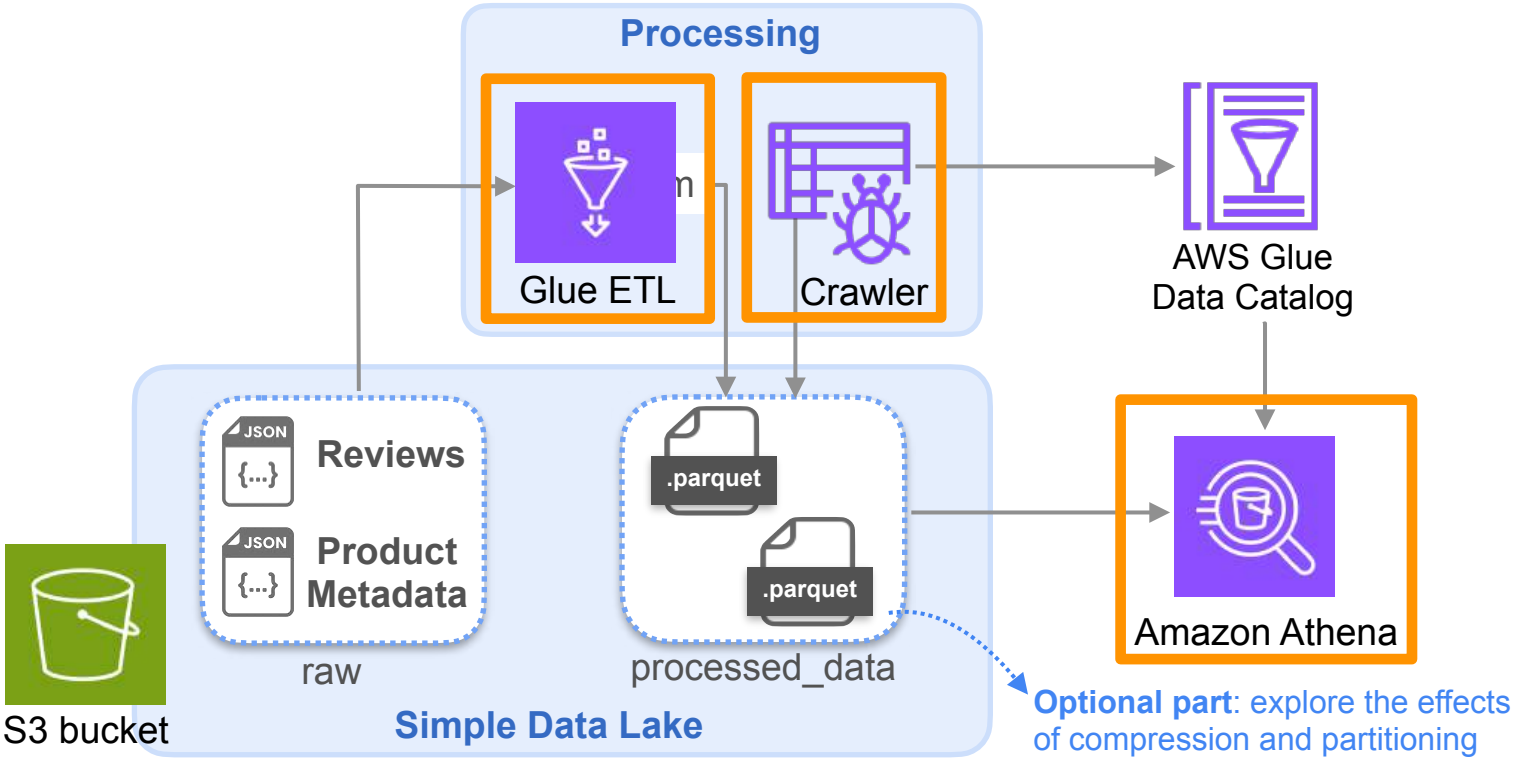
- Can introduce bugs/failures
- Can cause issues with data quality, duplication, consistency

DeepLearning.AI

Lab Walkthrough

Simple Data Lake with AWS Glue (Part 1)

Lab Overview



Processing the JSON Files

Complete two functions to process the raw JSON files

Reviews


```
{
  "reviewerID": "A2SUAM1J3GNN3B",
  "asin": "0000013714",
  "reviewerName": "J. McDonald",
  "helpful": [2, 3],
  "reviewText": "Great purchase!",
  "overall": 5.0,
  "summary": "Heavenly Highway Hymns",
  "unixReviewTime": 1252800000,
  "reviewTime": "09 13, 2009"
}
```


Import the review data into a tabular data frame

reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime

Process the data

reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime	year	helpful	totalHelpful

Processing the JSON Files

Complete two functions to process the raw JSON files

Product Metadata

```
{
"asin": "0000031852",
"description": "Girls Ballet Tutu Zebra Hot Pink",
"title": "Girls Ballet Tutu Zebra Hot Pink",
"price": 3.17,
"related": {
 "also bought": ["B00JHONNS", "B002BZX8Z6","B007R2RM8W"],
 "also viewed": ["B002BZX8Z6", "B00JHONN1S","B00BFXLZ8M"],
 "bought together": ["B002BZX8Z6"]
},
"salesRank":{"Toys & Games": 211836},
 "brand": "Coxlures",
 "categories": [["Sports & Outdoors", "Dance"]]
}
```

Import the metadata data into a tabular data frame

asin	description	title	price	related	salesRank	brand	categories

Process the data

asin	description	title	price	brand	sales_category	sales_rank

- Drop null entries from numerical columns
- Replace null values with empty strings in the other columns

Processing the JSON Files

Complete two functions to process the raw JSON files

Use the code to define the Glue jobs in Terraform

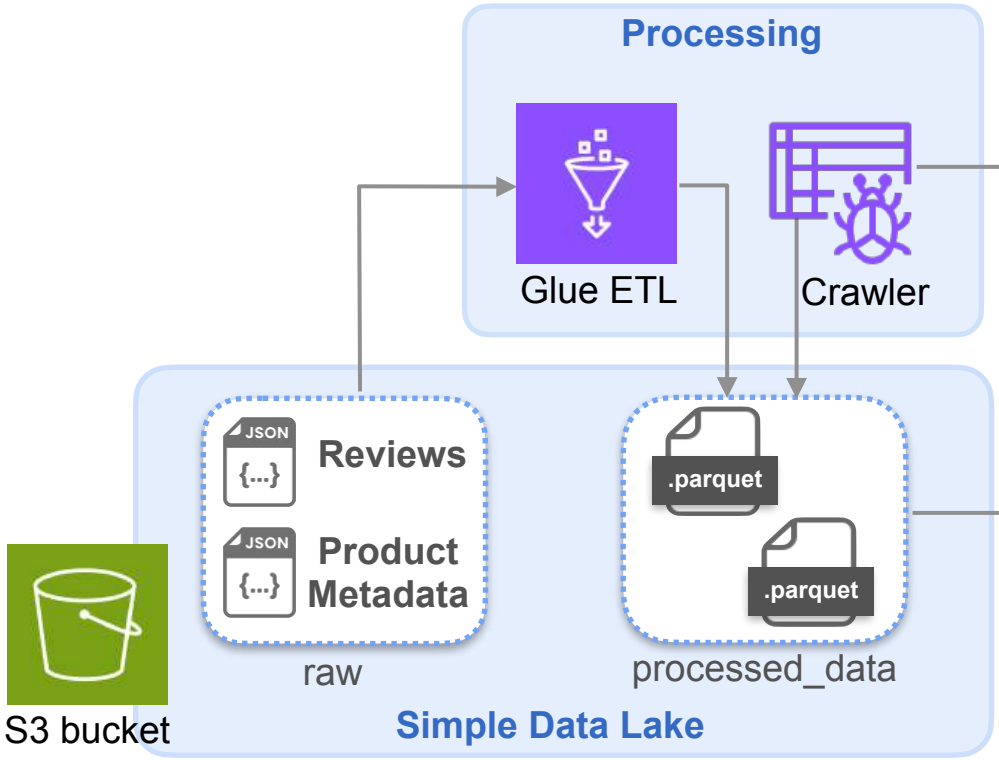
Glue ETL

DeepLearning.AI

Lab Walkthrough

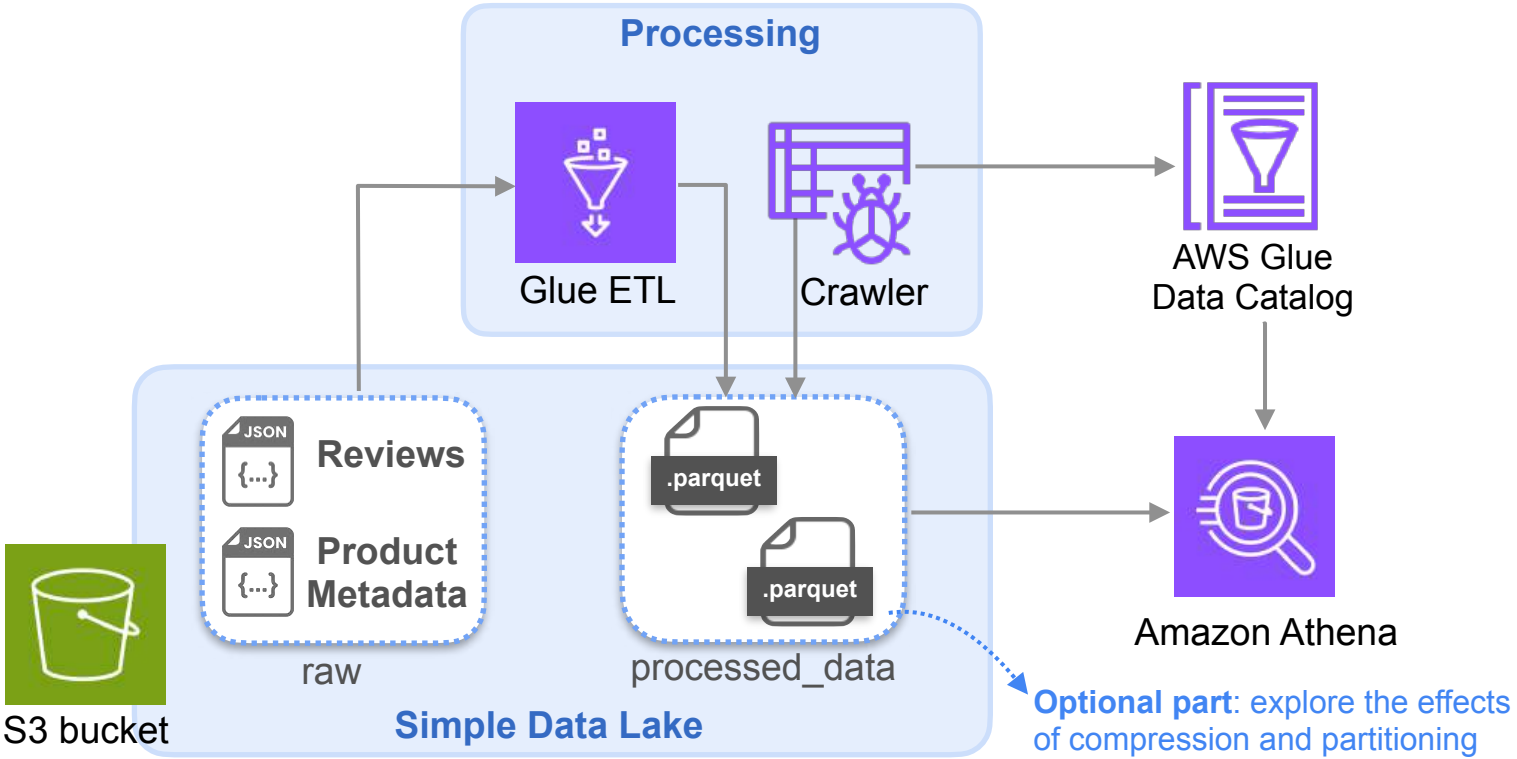
Simple Data Lake with AWS Glue (Part 2)

Glue Data Catalog



- **AWS Glue Data Catalog**: central repository that stores metadata for all your data assets
- You can create your metadata databases.
- The catalog database can contain many tables each storing:
  - Column names
  - Data types
  - Partition keys

Lab Overview



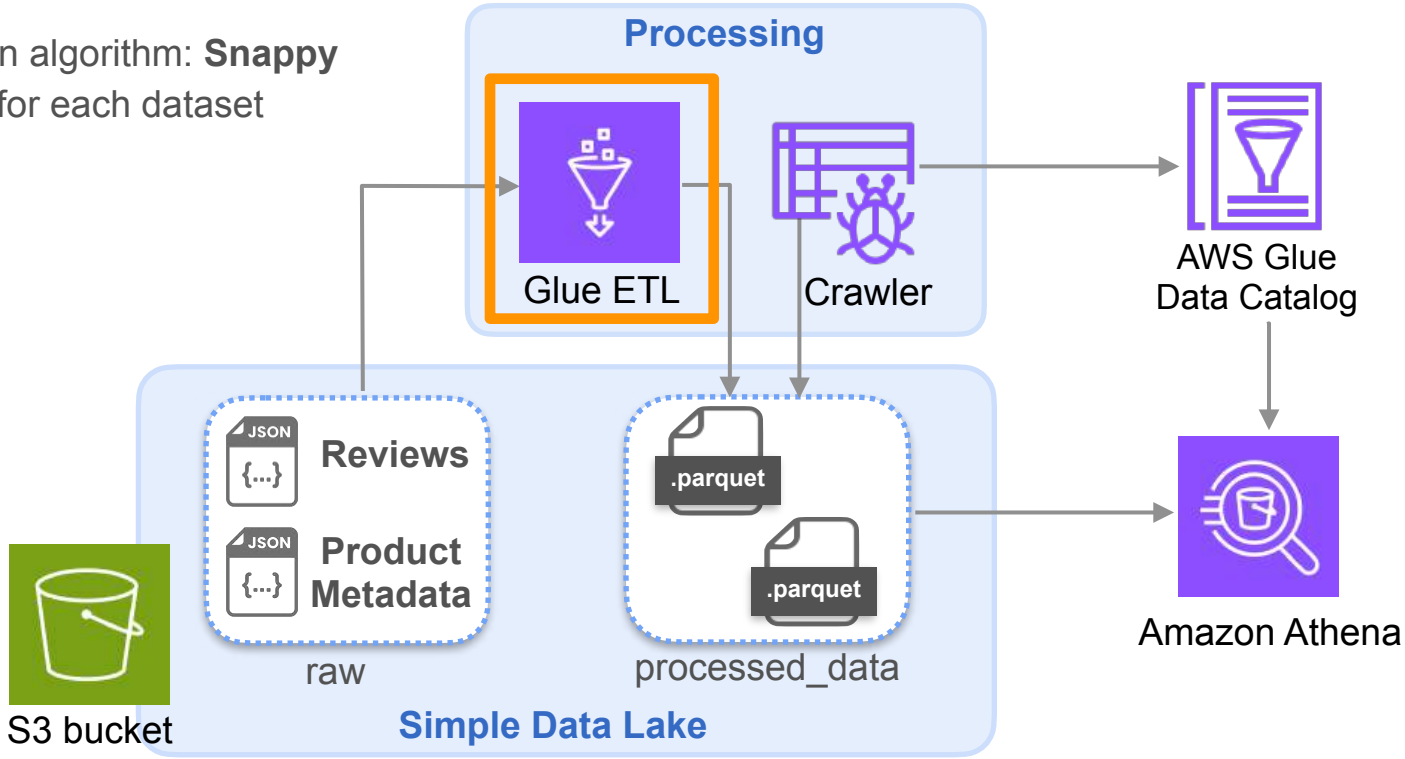
DeepLearning.AI

Lab Walkthrough

Simple Data Lake with AWS Glue (Part 3 - Optional)

Lab Experiments

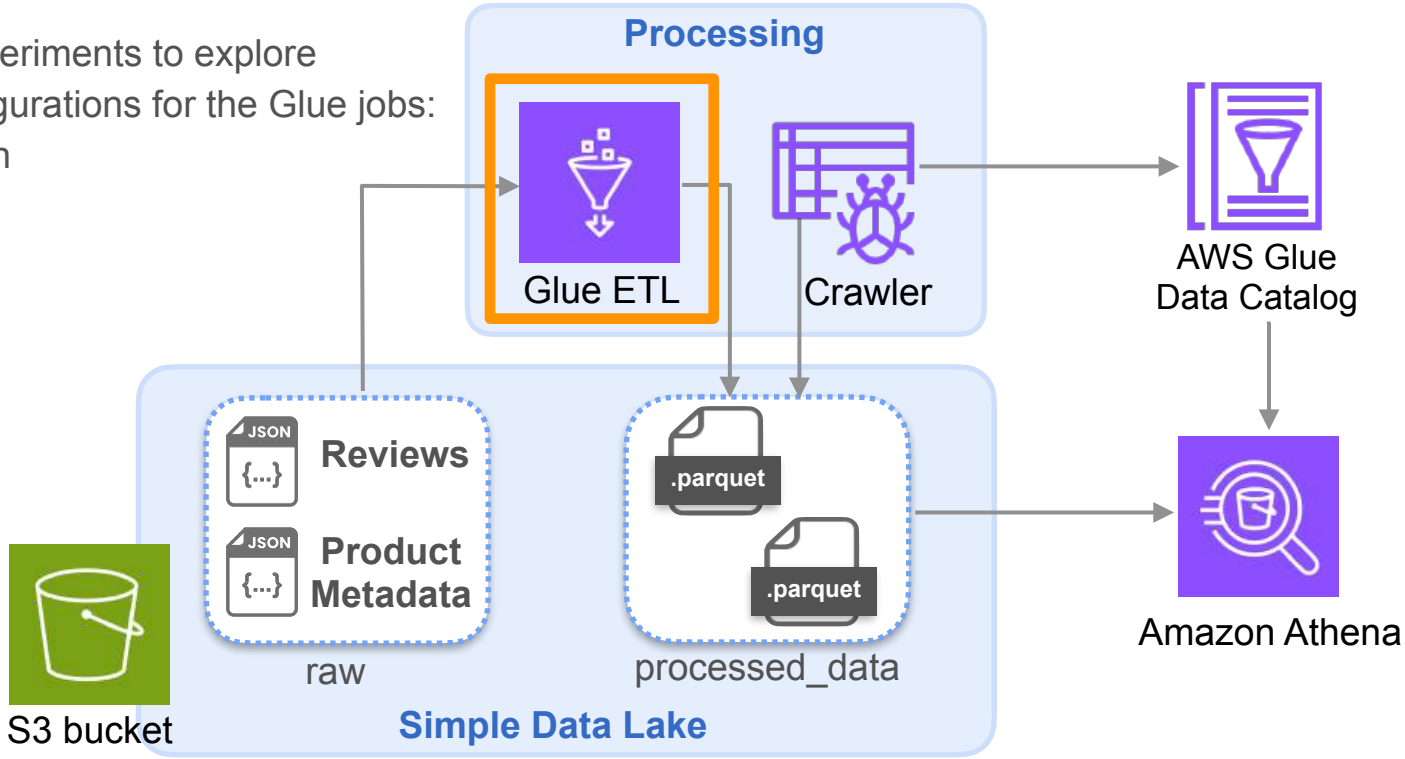
- Compression algorithm: **Snappy**
- Partitioning for each dataset



Lab Experiments

Perform 4 experiments to explore different configurations for the Glue jobs:

- compression
- partitioning



Lab Experiments

Experiment	Dataset	Partitioning	Compression Algorithm	Number of Objects	Total Size of the Processed Data
1	Metadata	No partitioning	Uncompressed	4	93.6 MiB
		No partitioning	Snappy	4	51 MiB

```
!aws s3 ls --summarize --human-readable --recursive
s3://{BUCKET_NAME}/processed_data/uncompressed/no_partition/toys_metadata
```

Lab Experiments

Experiment	Dataset	Partitioning	Compression Algorithm	Number of Objects	Total Size of the Processed Data
1	Metadata	No partitioning	Uncompressed	4	93.6 MiB
		No partitioning	Snappy	4	51 MiB
2	Metadata	No partitioning	Snappy	4	51 MiB
		No partitioning	Gzip	4	32.7 MiB

faster

Lab Experiments

Experiment	Dataset	Partitioning	Compression Algorithm	Number of Objects	Total Size of the Processed Data
1	Metadata	No partitioning	Uncompressed	4	93.6 MiB
		No partitioning	Snappy	4	51 MiB
2	Metadata	No partitioning	Snappy	4	51 MiB
		No partitioning	Gzip	4	32.7 MiB
3	Reviews	No partitioning	Snappy	4	556.6 MiB
		Partitioning year, month	Snappy	556	578.0 MiB

- Choose an appropriate key that organizes the data into meaningful files that are aligned with your query pattern.
- A bad partitioning key divides data into too many small files.

Lab Experiments

Experiment	Dataset	Partitioning	Compression Algorithm	Number of Objects	Total Size of the Processed Data
1	Metadata	No partitioning	Uncompressed	4	93.6 MiB
		No partitioning	Snappy	4	51 MiB
2	Metadata	No partitioning	Snappy	4	51 MiB
		No partitioning	Gzip	4	32.7 MiB
3	Reviews	No partitioning	Snappy	4	556.6 MiB
		Partitioning year, month	Snappy	556	578.0 MiB
4	Reviews	Partitioning year, month	Snappy	556	578.0 MiB
		Partitioning asin	Snappy	Timeout	

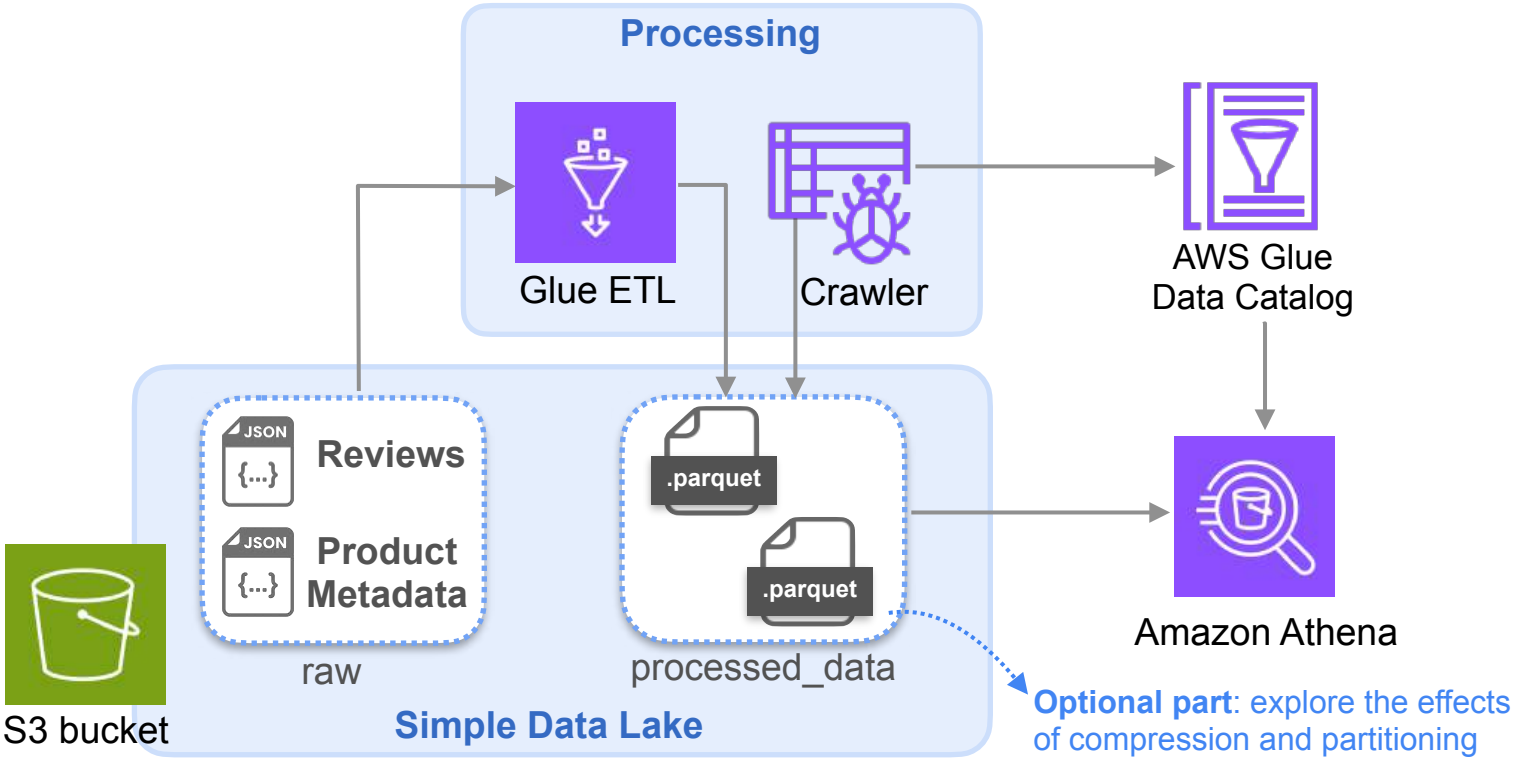
What you'll run?

Dataset	Partitioning	Compression Algorithm
Metadata	No partitioning	Uncompressed
	No partitioning	Snappy
	No partitioning	Gzip
Reviews	No partitioning	Snappy
	Partitioning year, month	Snappy
	Partitioning asin	Snappy

What you'll compare?

Experiment	Dataset	Partitioning	Compression Algorithm
1	Metadata	No partitioning	Uncompressed
		No partitioning	Snappy
2	Metadata	No partitioning	Snappy
		No partitioning	Gzip
3	Reviews	No partitioning	Snappy
		Partitioning year, month	Snappy
4	Reviews	Partitioning year, month	Snappy
		Partitioning asin	Snappy

Lab Overview



DeepLearning.AI

Storage Abstractions

Data Lakehouse

Data Lakehouse

Data Lake

- Flexibility
- Low-Cost Storage

+

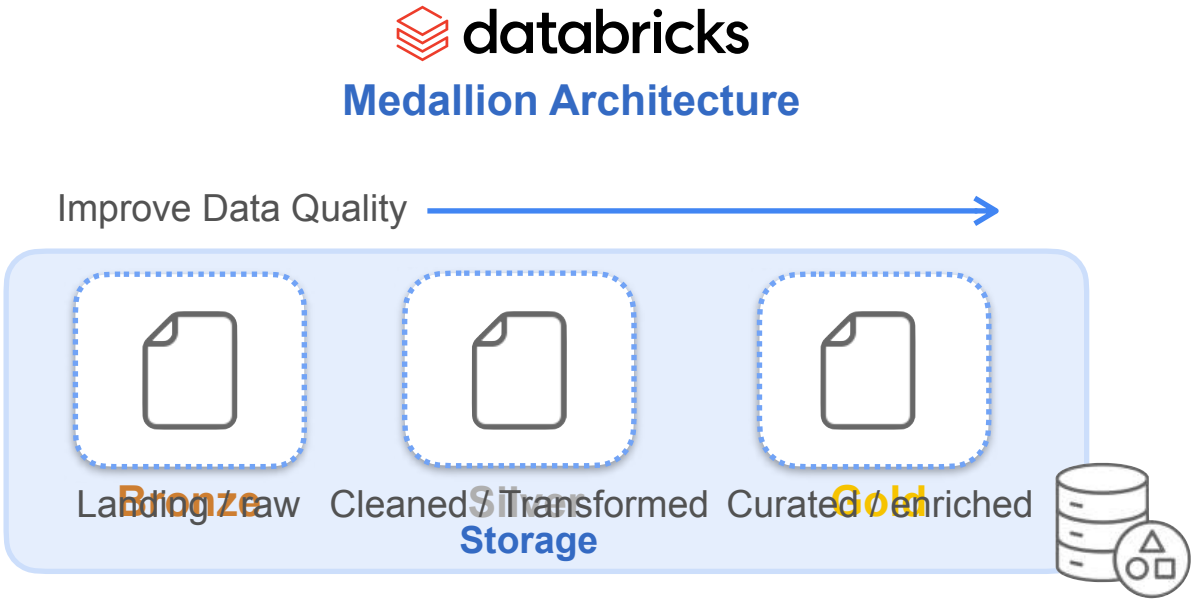
Data Warehouse

- Superior query performance
- Robust data management

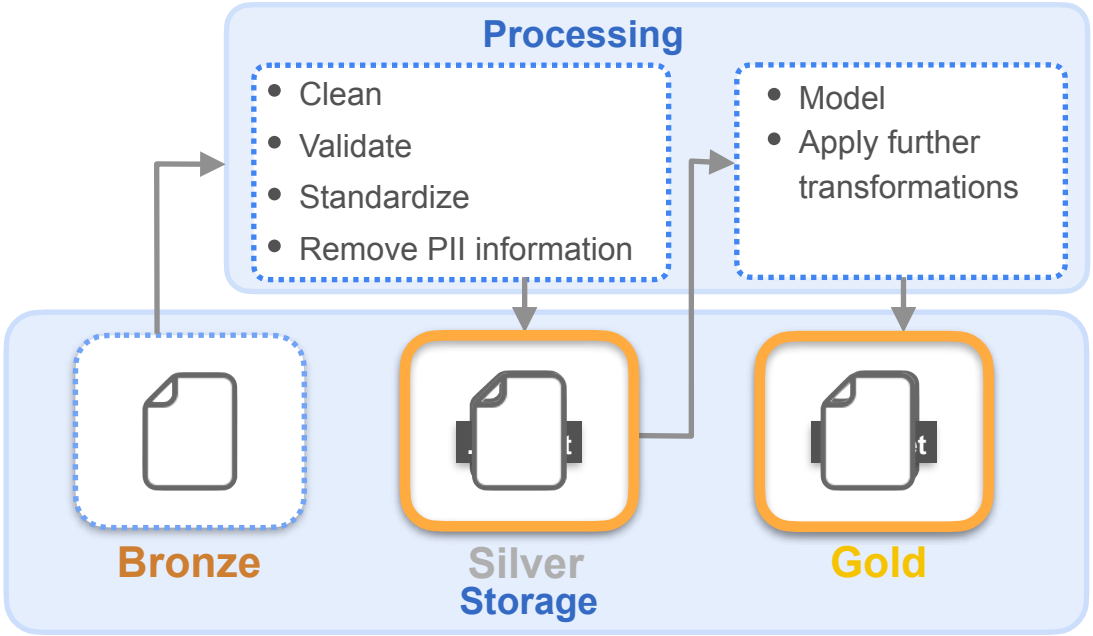
=

Data Lakehouse

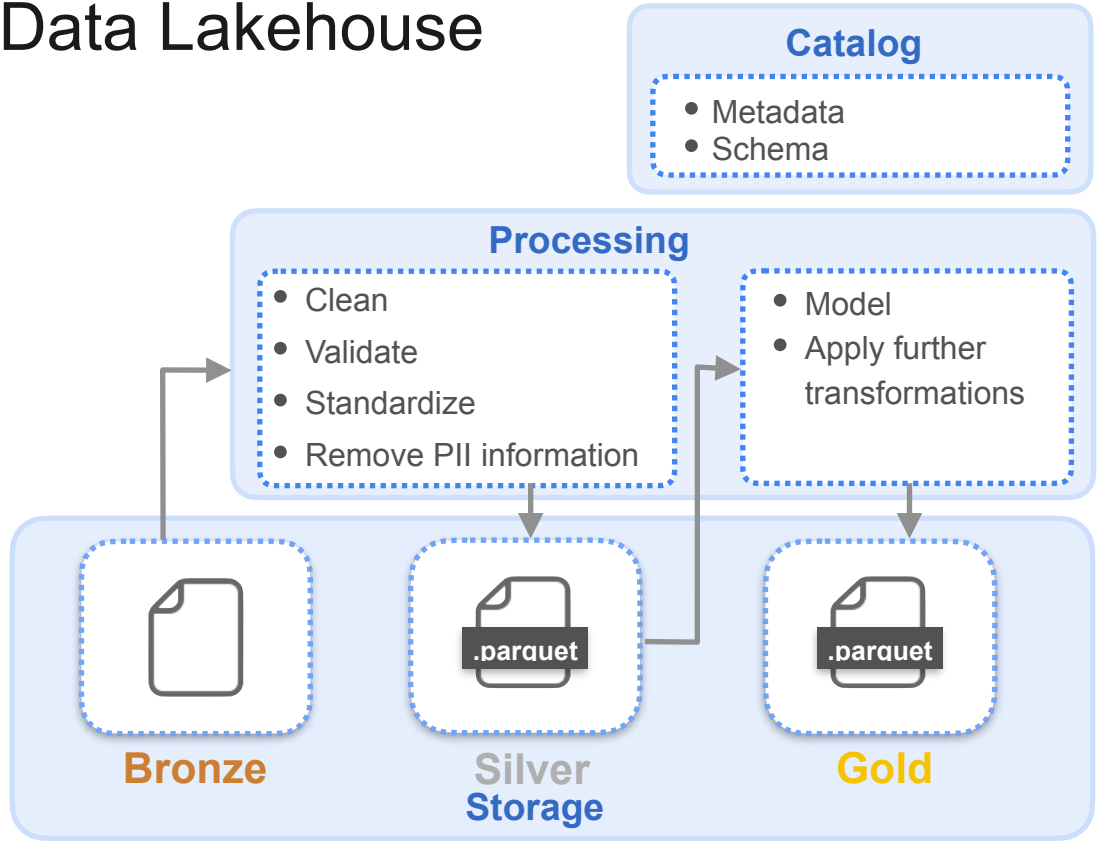
Data Lake Features



Data Lake Features

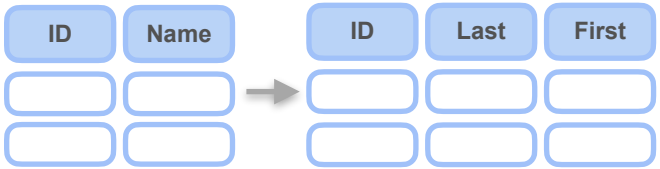


Data Lakehouse

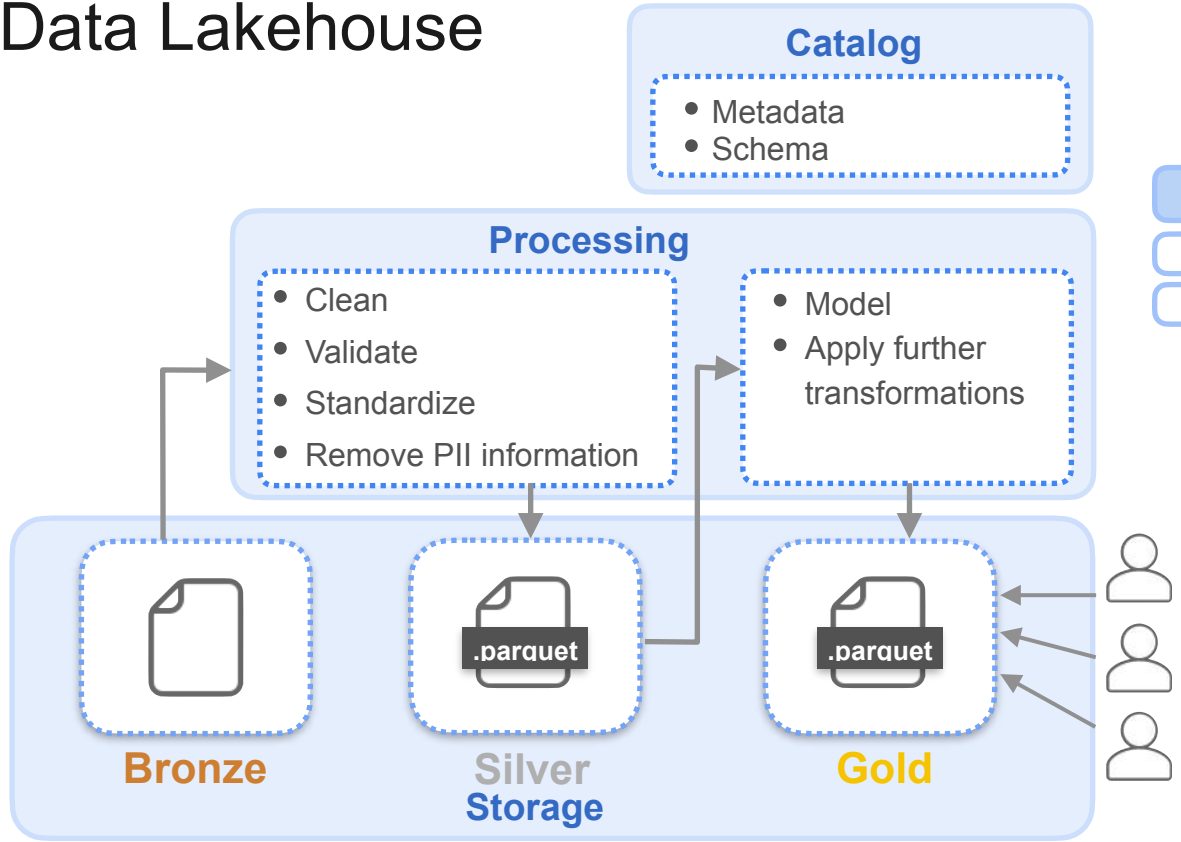


Data Management from Data Warehouses

Schema Enforcement

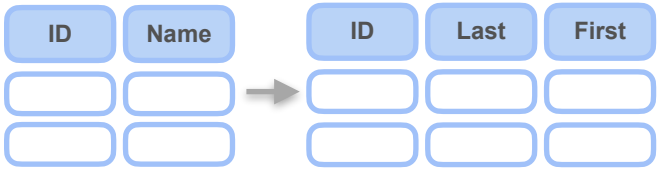


Data Lakehouse



Data Management from Data Warehouses

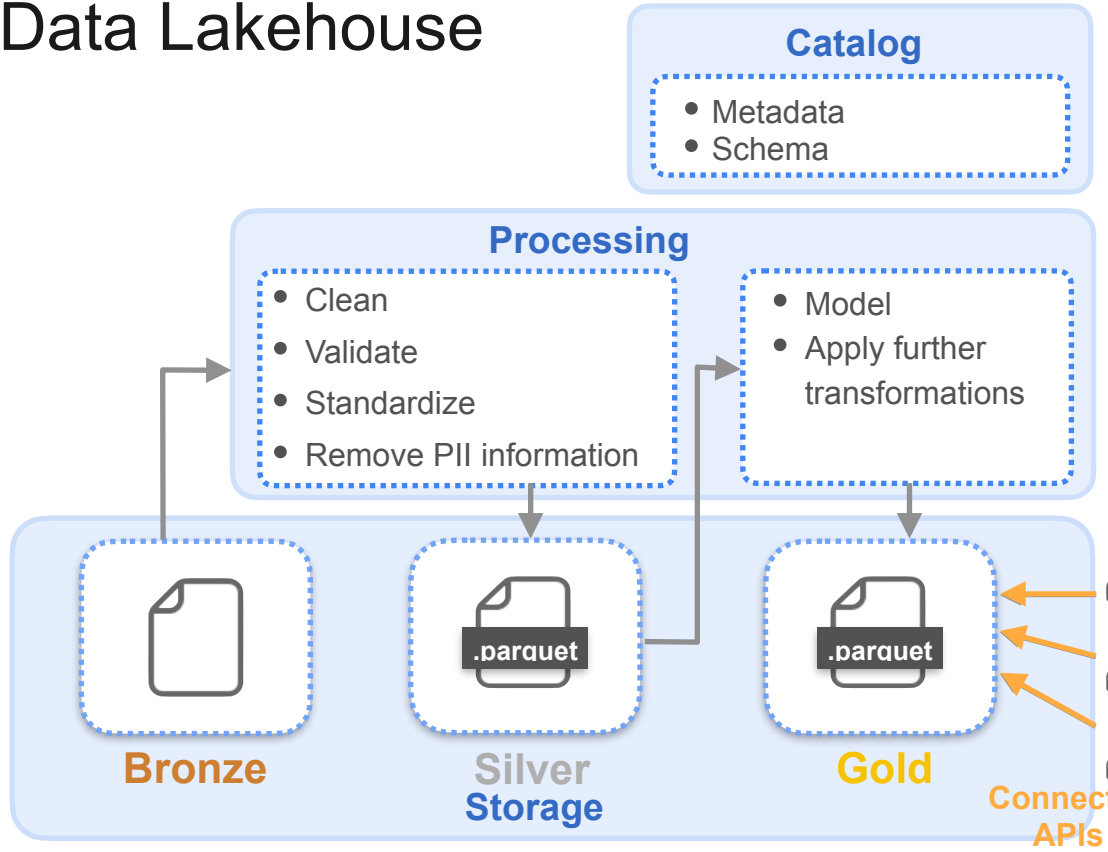
Schema Enforcement



ACID

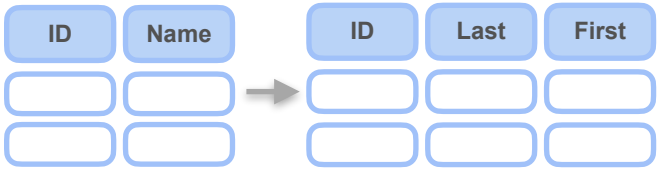
- **A**tomic, **C**onsistent, **I**solated, **D**urable
- Concurrent read, insert, update, delete

Data Lakehouse



Data Management from Data Warehouses

Schema Enforcement



ACID

- **A**tomic, **C**onsistent, **I**solated, **D**urable
- Concurrent read, insert, update, delete

Data Governance & Security

- Robust access controls, data auditing capabilities, data lineage
- Incremental updates & deletions
- Rollback to or access any version of your historical data

DeepLearning.AI

Storage Abstractions

Date Lakehouse Implementations

Open Table Formats

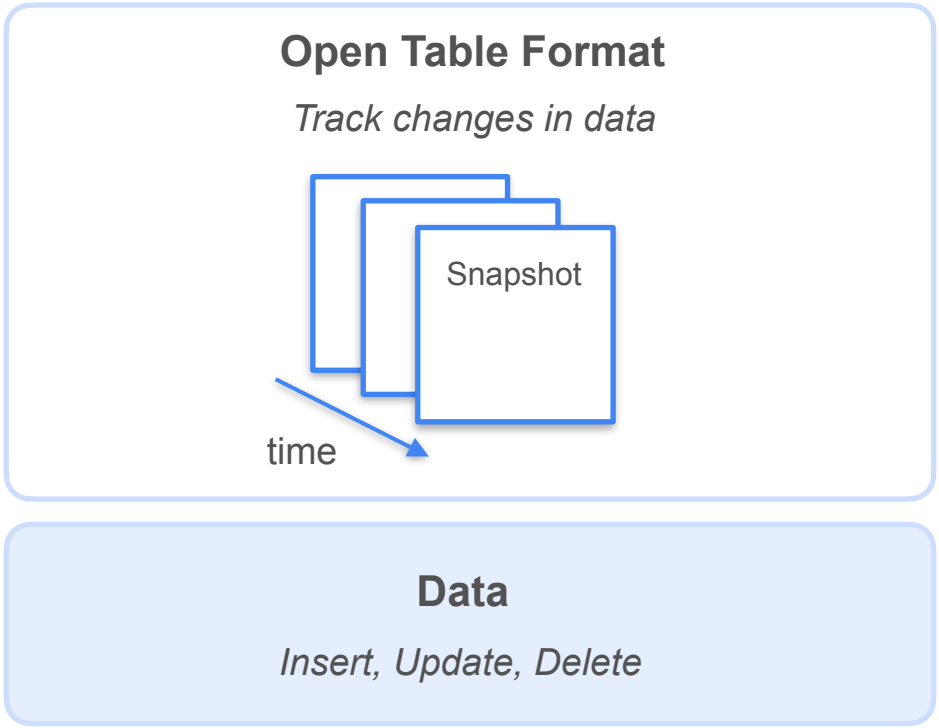
Open Table Formats

Specialized storage formats that add transactional features to your data lakehouse

- Allows you to update and delete records
- Supports ACID principles

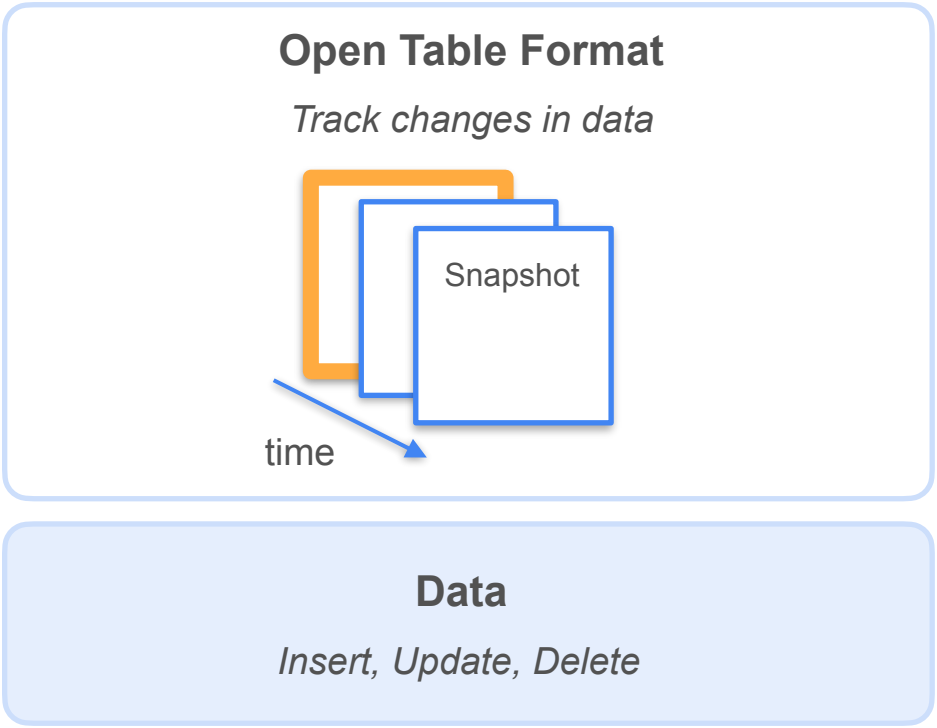
Hadoop
Update
Delete
Incremental

Open Table Formats



Snapshot:
Reflects the state of the data at a given time

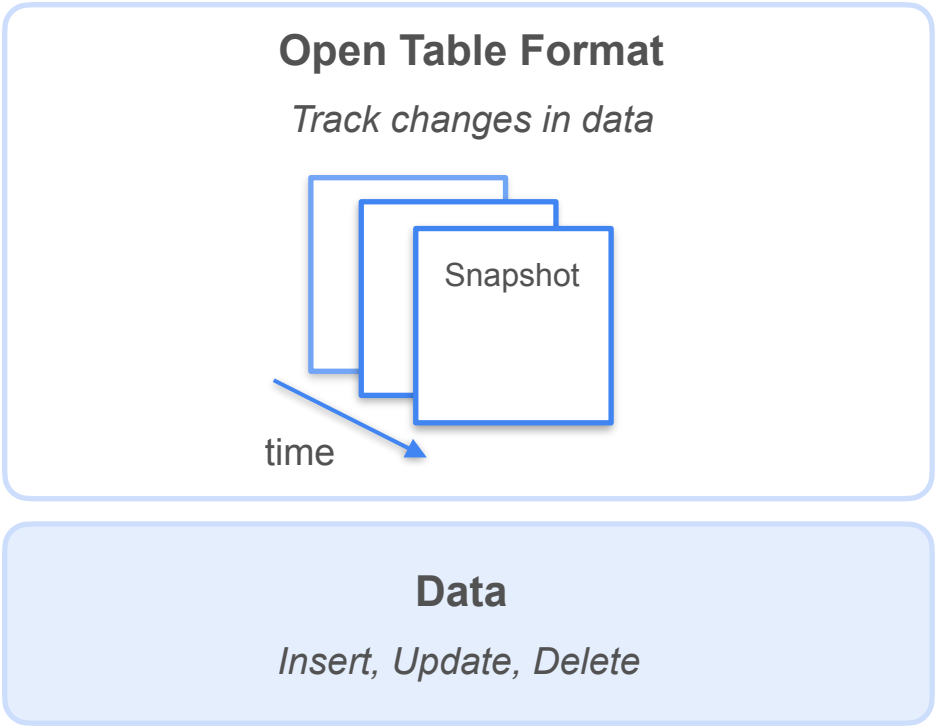
Open Table Formats



Snapshot:
Reflects the state of the data at a given time

Time Travel:
Query any previous version of a table

Open Table Formats

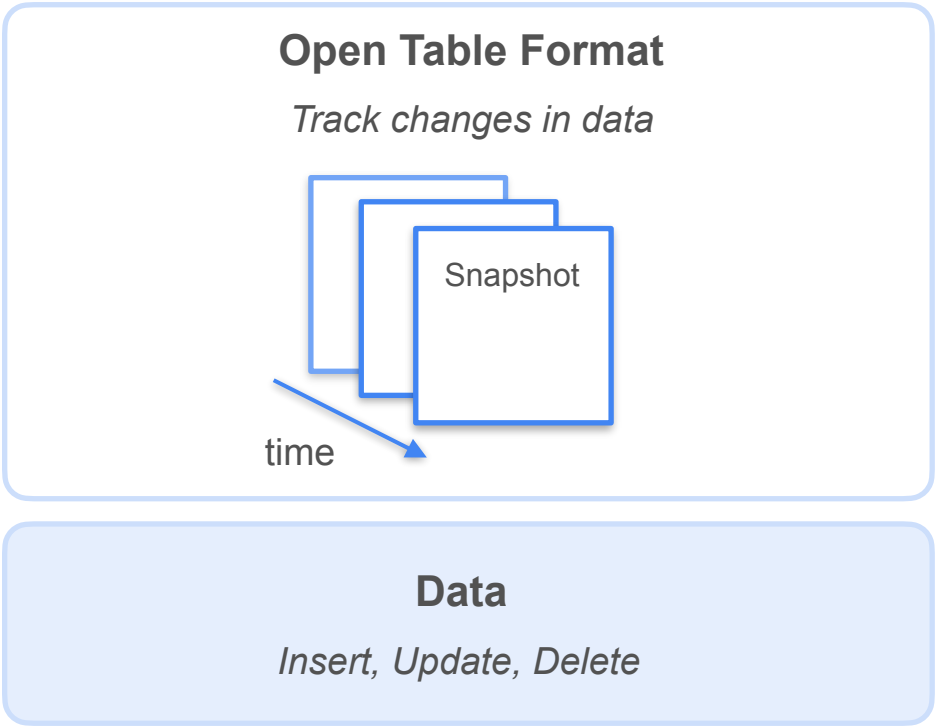


Snapshot:
Reflects the state of the data at a given time

Time Travel:
Query any previous version of a table

Schema & Partition Evolution:
Ability to query the data even if you make changes to the schema or partitioning

Open Table Formats



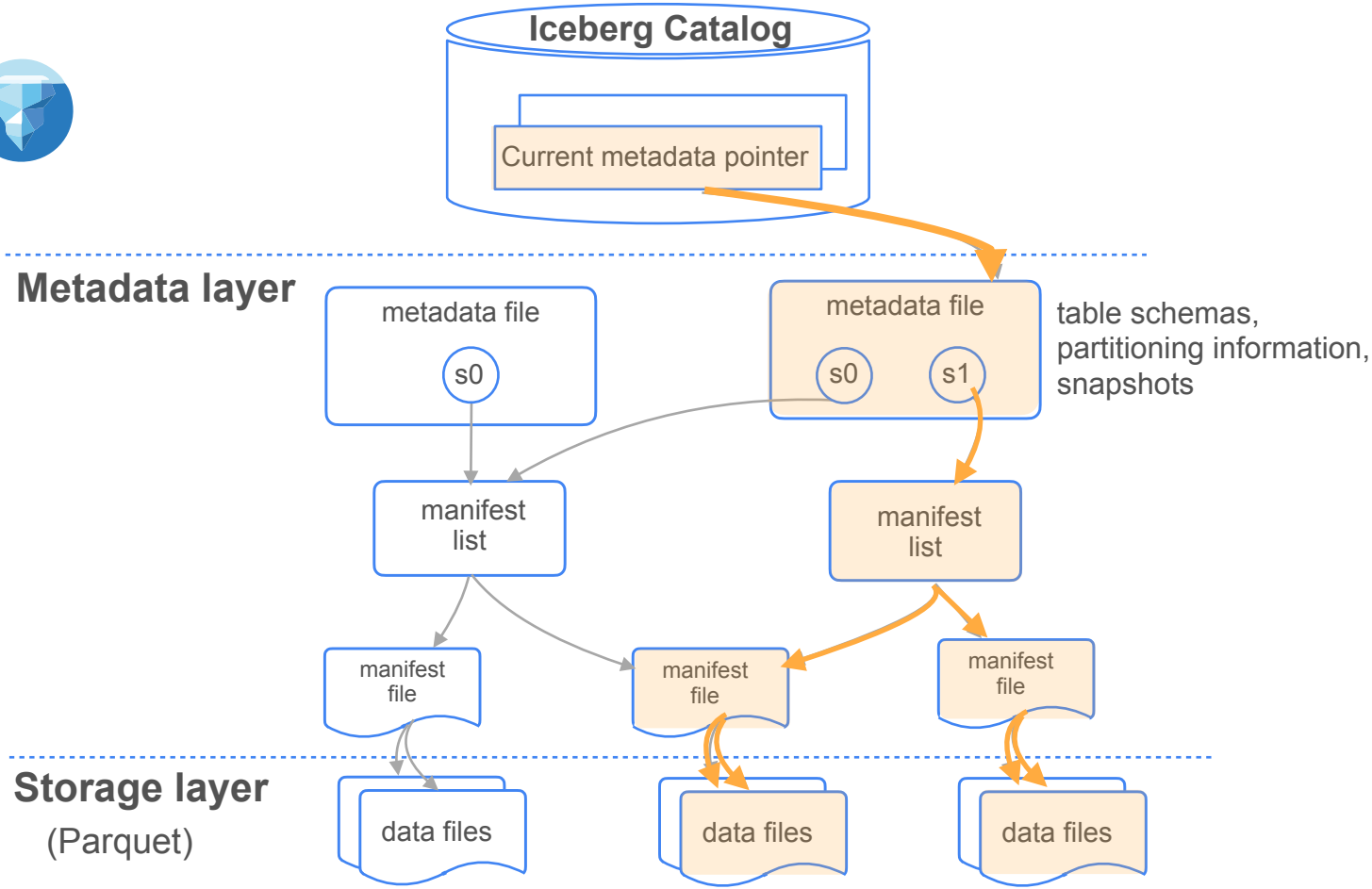
Snapshot:
Reflects the state of the data at a given time

Time Travel:
Query any previous version of a table

Schema & Partition Evolution:
Ability to query the data even if you make changes to the schema or partitioning

Open Source
Different query engines can access the data

ICEBERG


Iceberg Catalog
Current metadata pointer
Metadata layer
metadata file
s0
metadata file
s0
s1
table schemas,
partitioning information,
snapshots
manifest
list
manifest
list
manifest
file
manifest
file
manifest
file
Storage layer
(Parquet)
data files
data files
data files

Storage Options

Production Database	Data Warehouse		
• Process small amounts of structured data	• Bring together large volumes of structured / semi-structured data • Query current and historical data		
Use case: analytics, reporting	**Use case**: analytics, reporting		

Storage Options

Production Database	Data Warehouse	Data Lake	
• Process small amounts of structured data	• Bring together large volumes of structured / semi-structured data • Query current and historical data	• Process large volumes of structured, semi-structured and unstructured data • Save on storage cost	
Use case: reporting, analytics	**Use case**: reporting, analytics	**Use case**: machine learning	

Storage Options

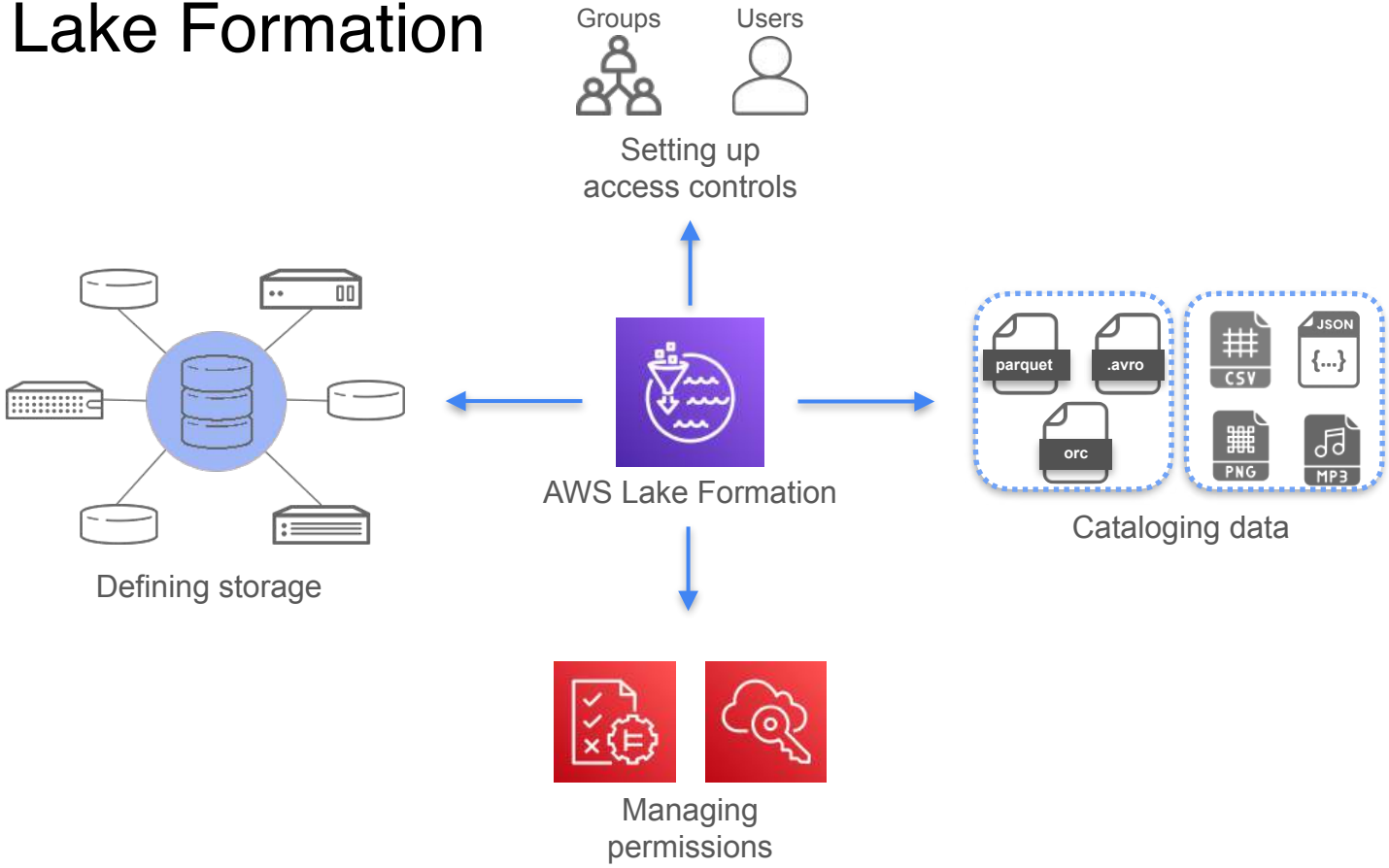
Production Database	Data Warehouse	Data Lake	Data Lakehouse
• Process small amounts of structured data	• Bring together large volumes of structured / semi-structured data • Query current and historical data	• Process large volumes of structured, semi-structured and unstructured data • Save on storage cost	• Data management & discoverability features • Low latency queries
Use case: reporting, analytics	**Use case**: reporting, analytics	**Use case**: machine learning	**Use case**: machine learning, analytics, reporting

DeepLearning.AI

Storage Abstractions

Lakehouse Architecture on AWS

AWS Lake Formation



AWS Lake Formation

- ✓Identify data sources
- ✓Move data into the data lake
- ✓Catalog data
- ✓Manage permissions

Setting up a Data Lake

Data Sources

Amazon S3

Relational DB

NoSQL DB

Crawler

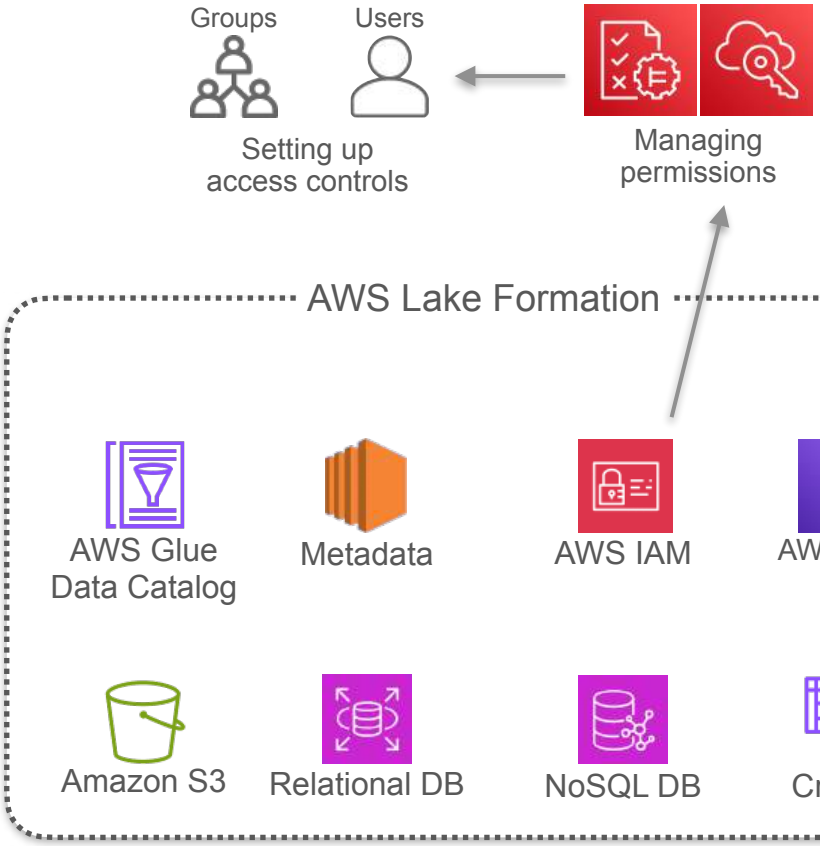
AWS Lake Formation

AWS Glue Data Catalog

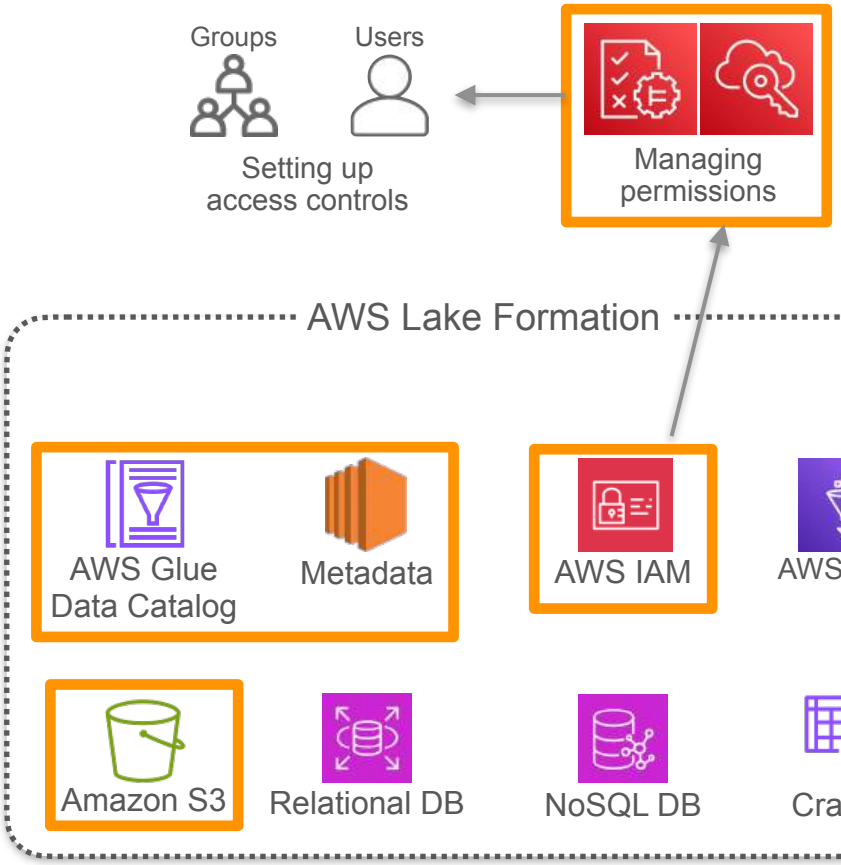
Metadata

AWS IAM

AWS Lake Formation

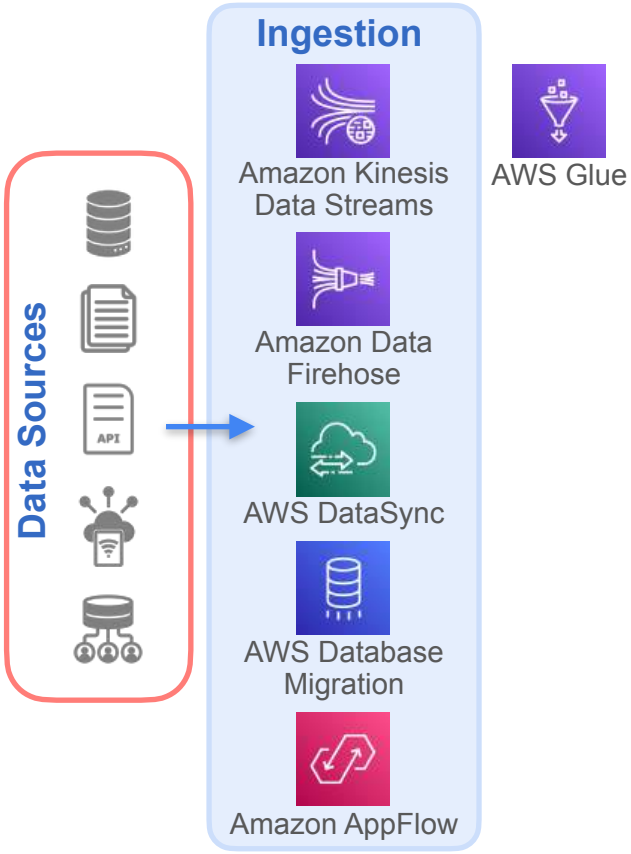


AWS Lake Formation

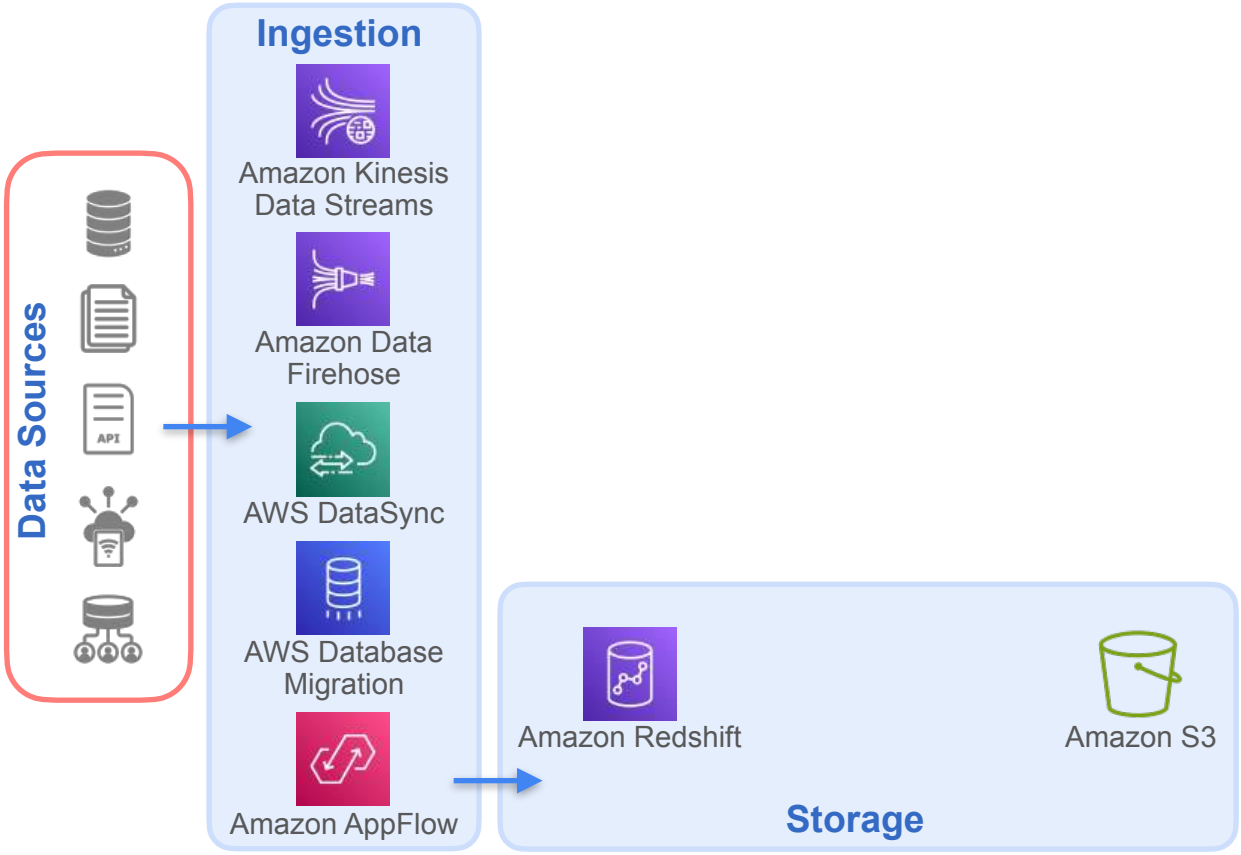


AWS Data Lakehouse

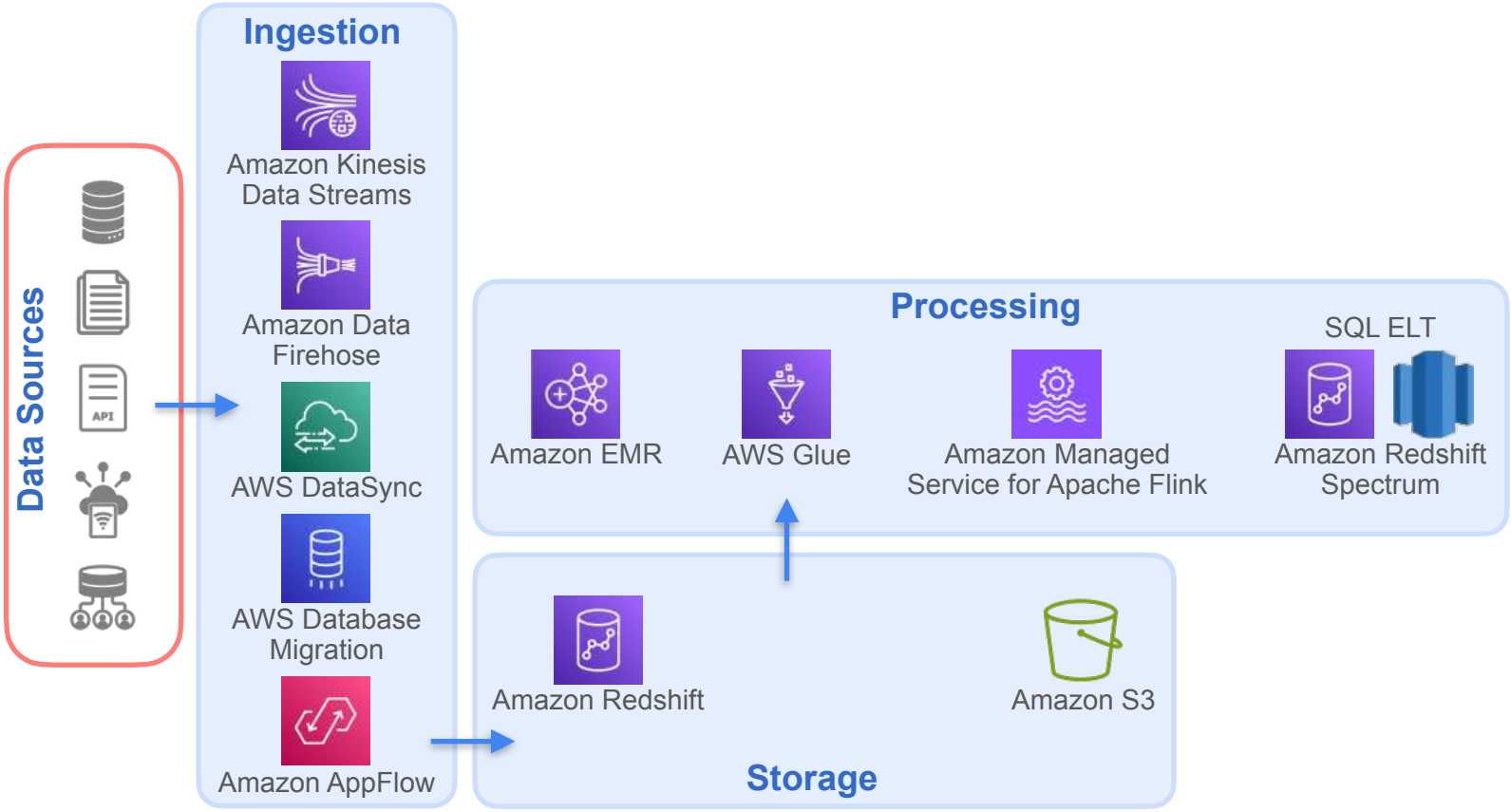
AWS Data Lakehouse



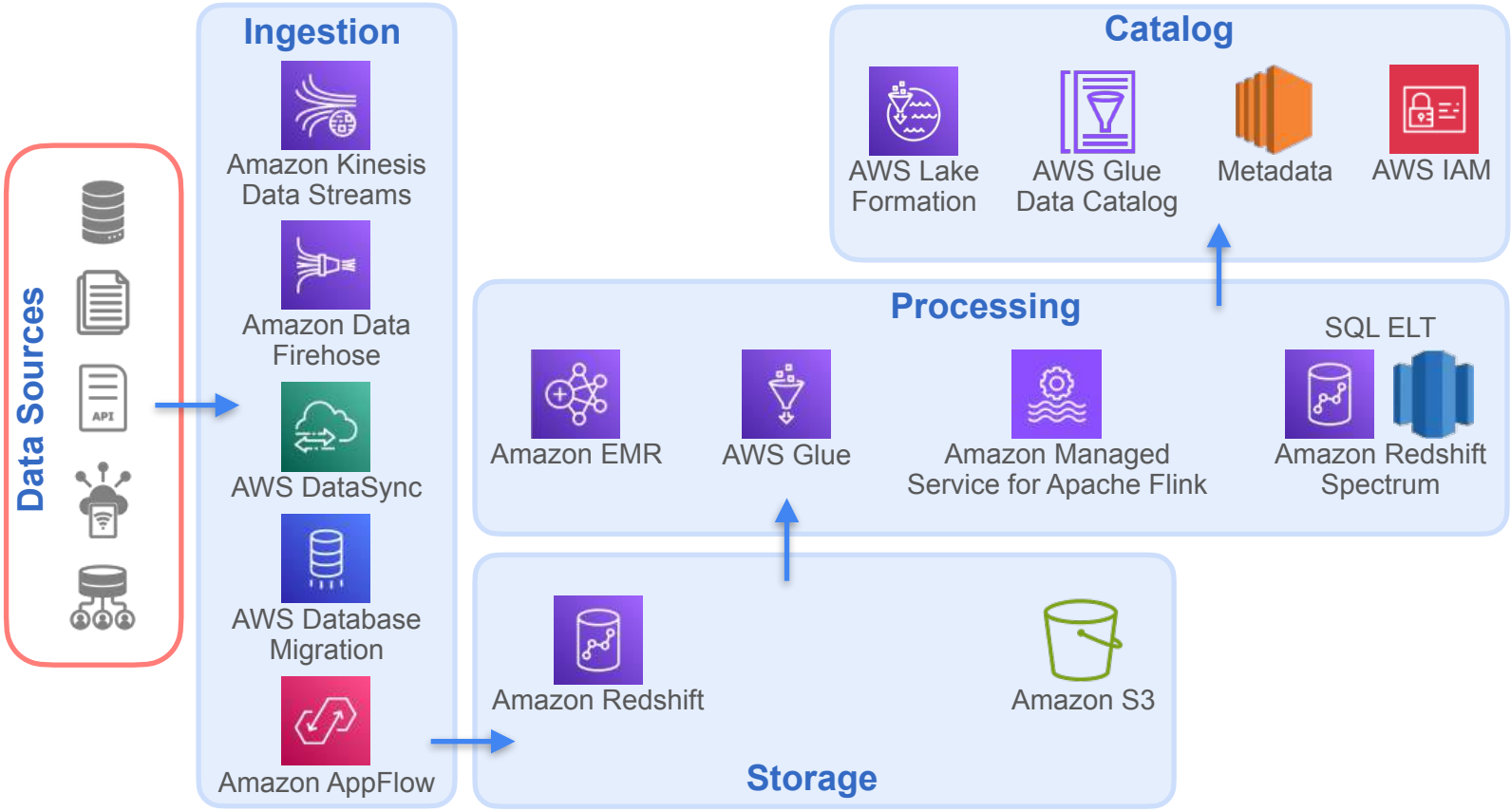
AWS Data Lakehouse



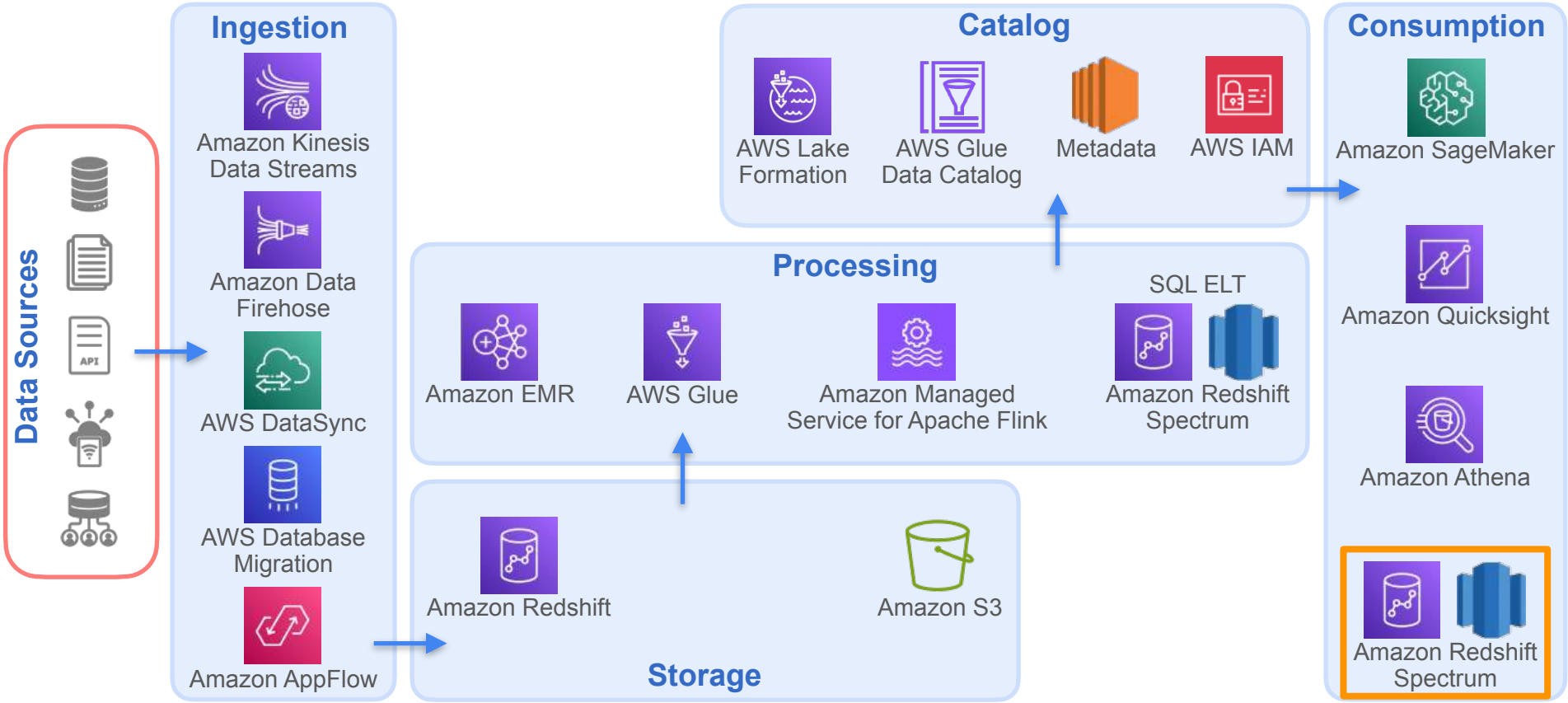
AWS Data Lakehouse



AWS Data Lakehouse



AWS Data Lakehouse

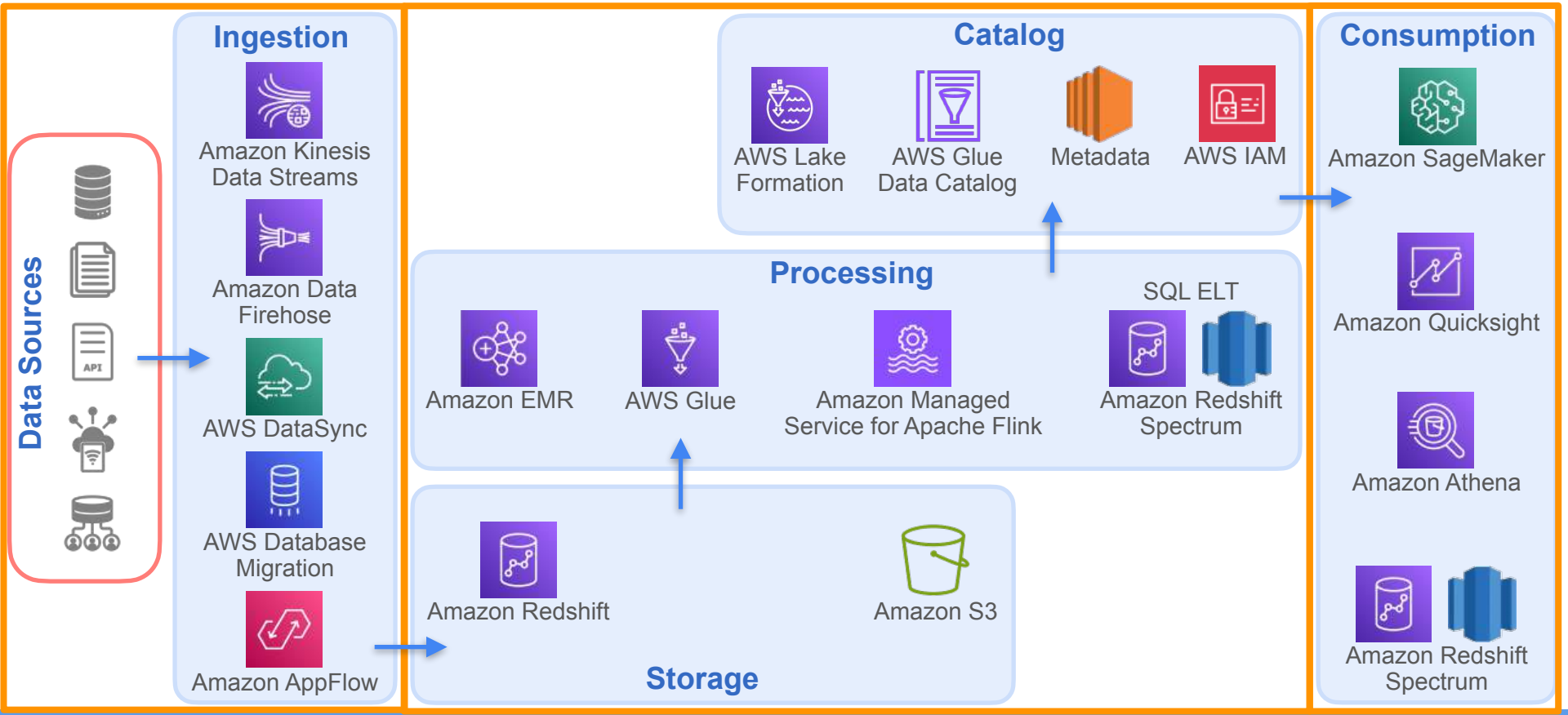


DeepLearning.AI

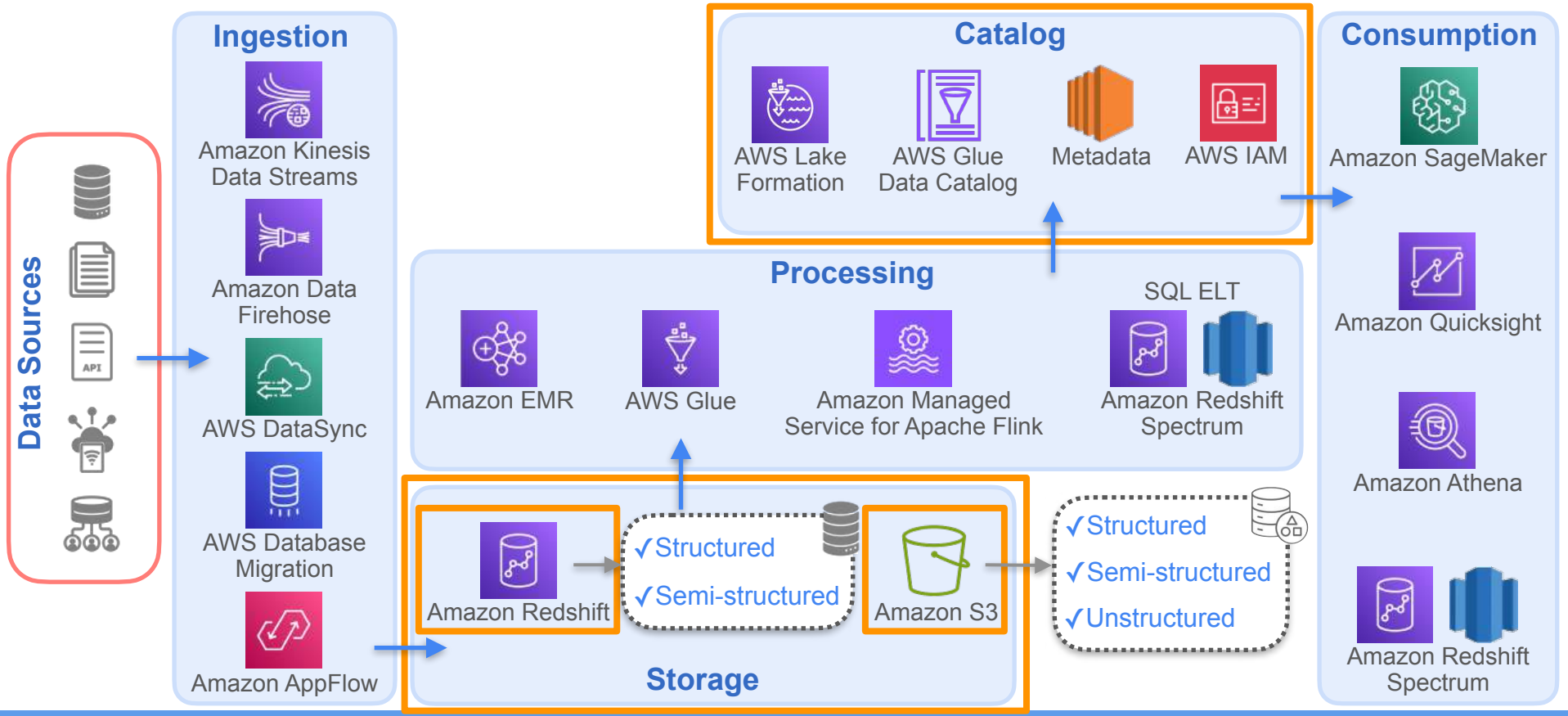
Storage Abstractions

Implementing a Lakehouse on AWS

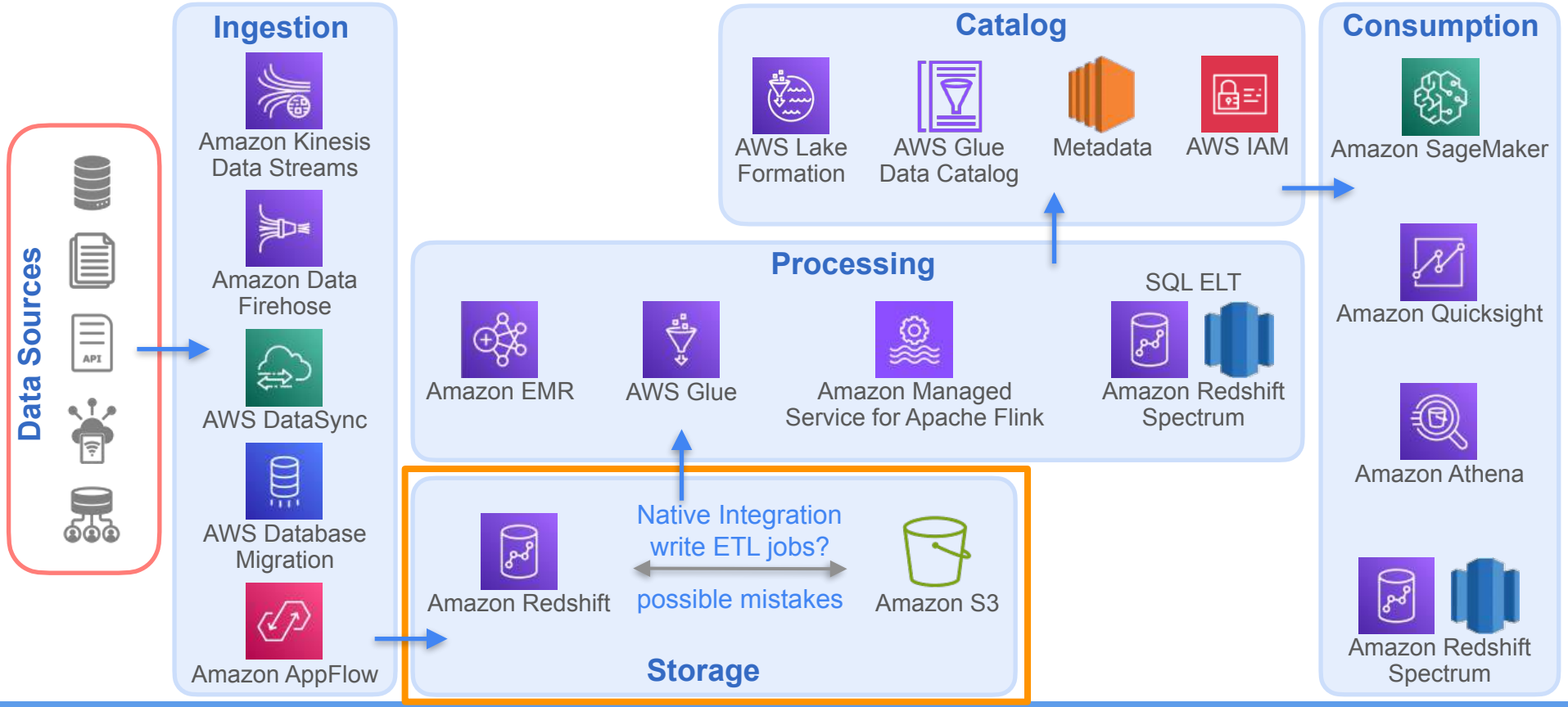
AWS Data Lakehouse



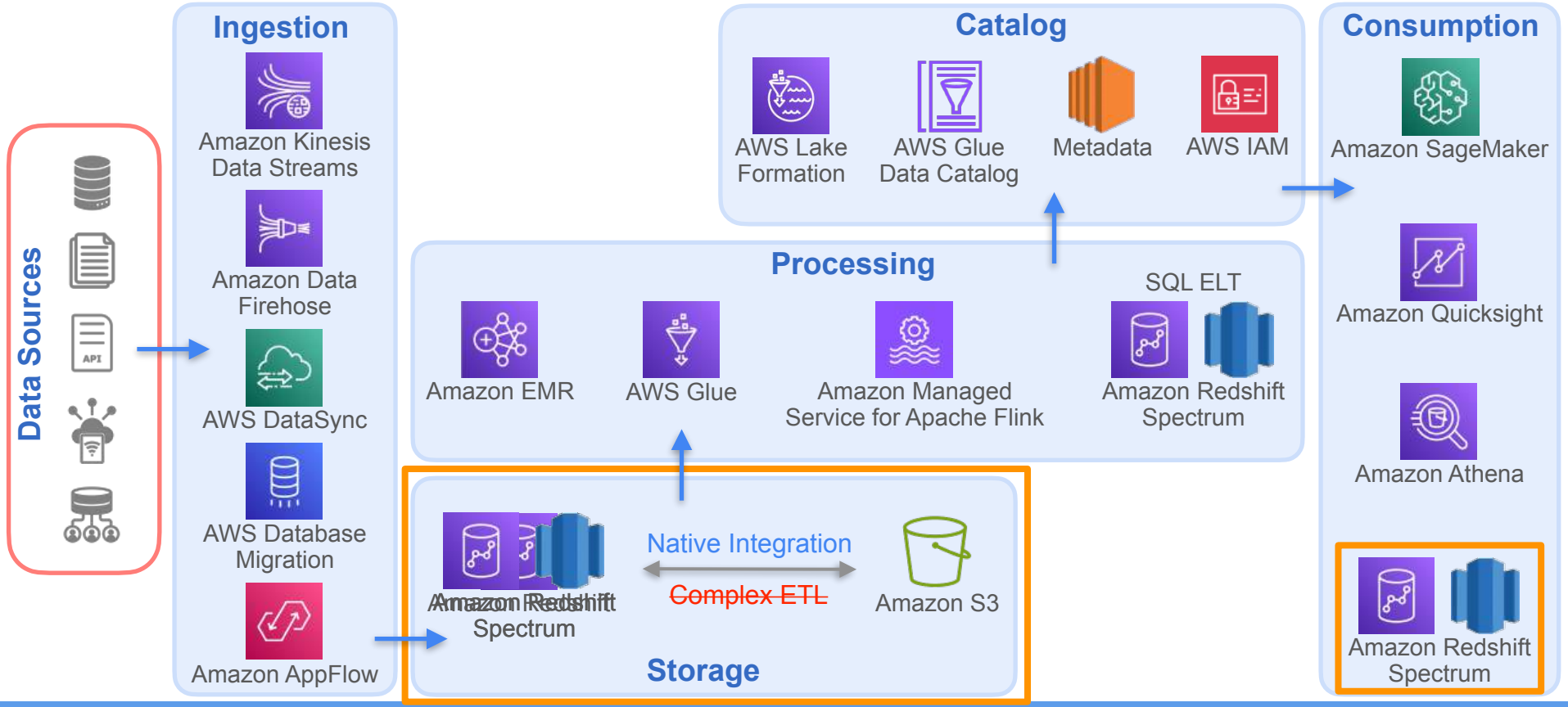
AWS Data Lakehouse



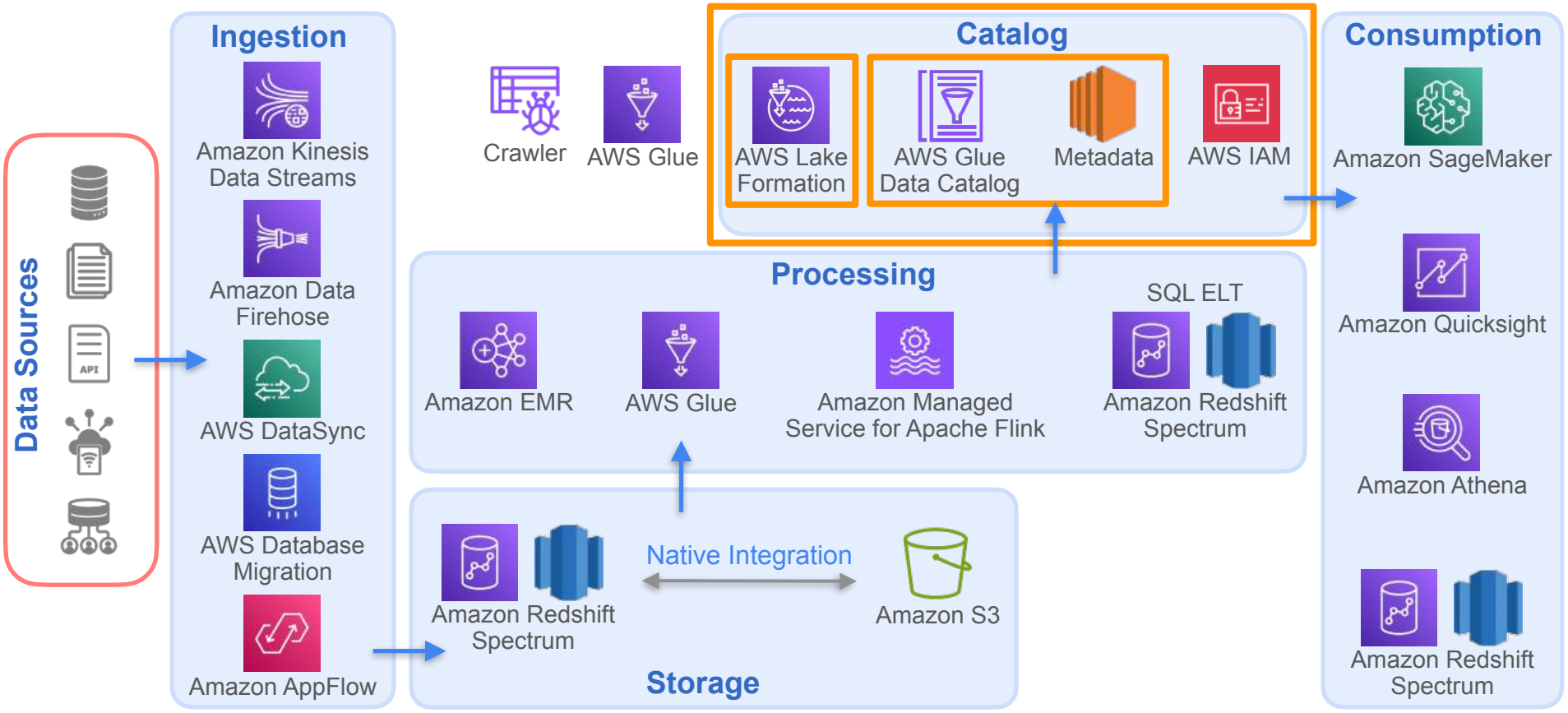
AWS Data Lakehouse



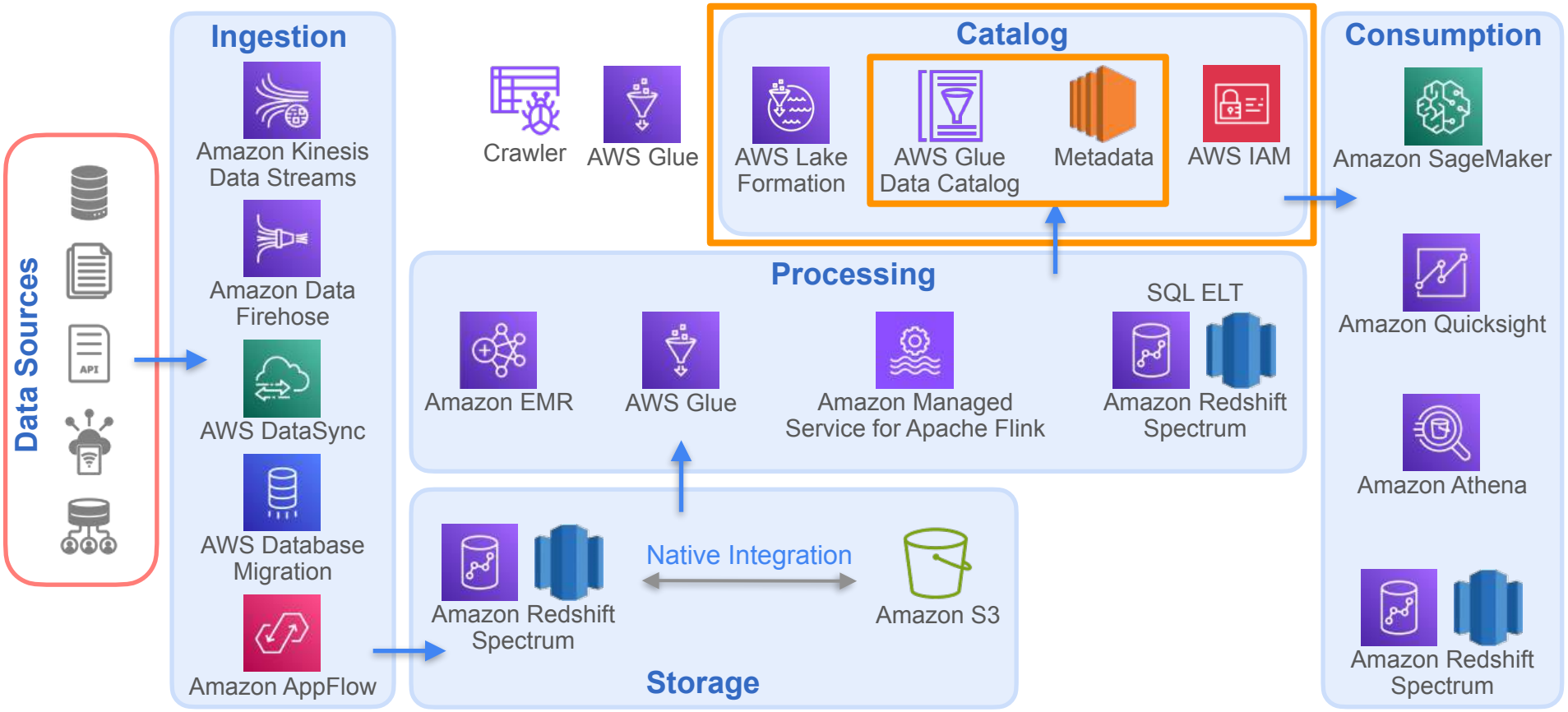
AWS Data Lakehouse



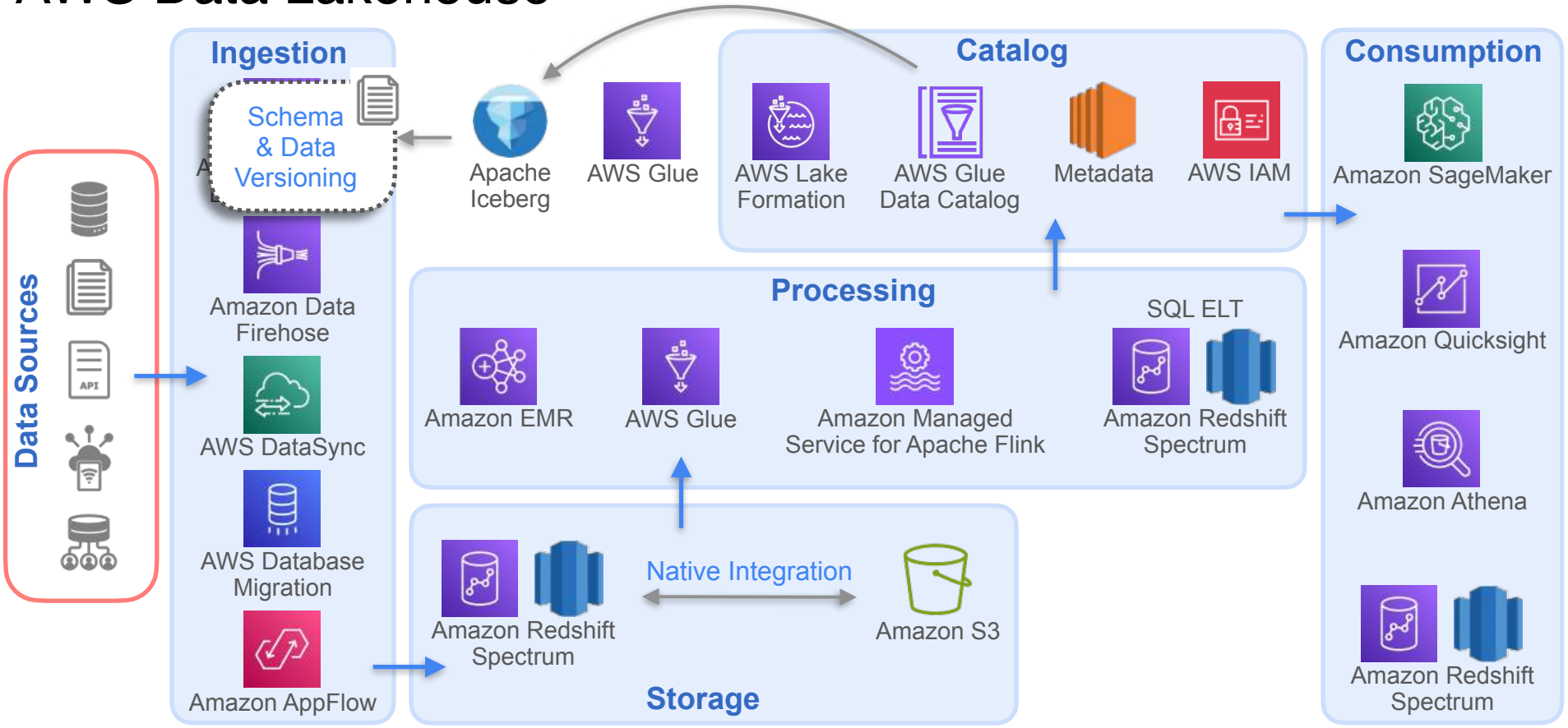
AWS Data Lakehouse



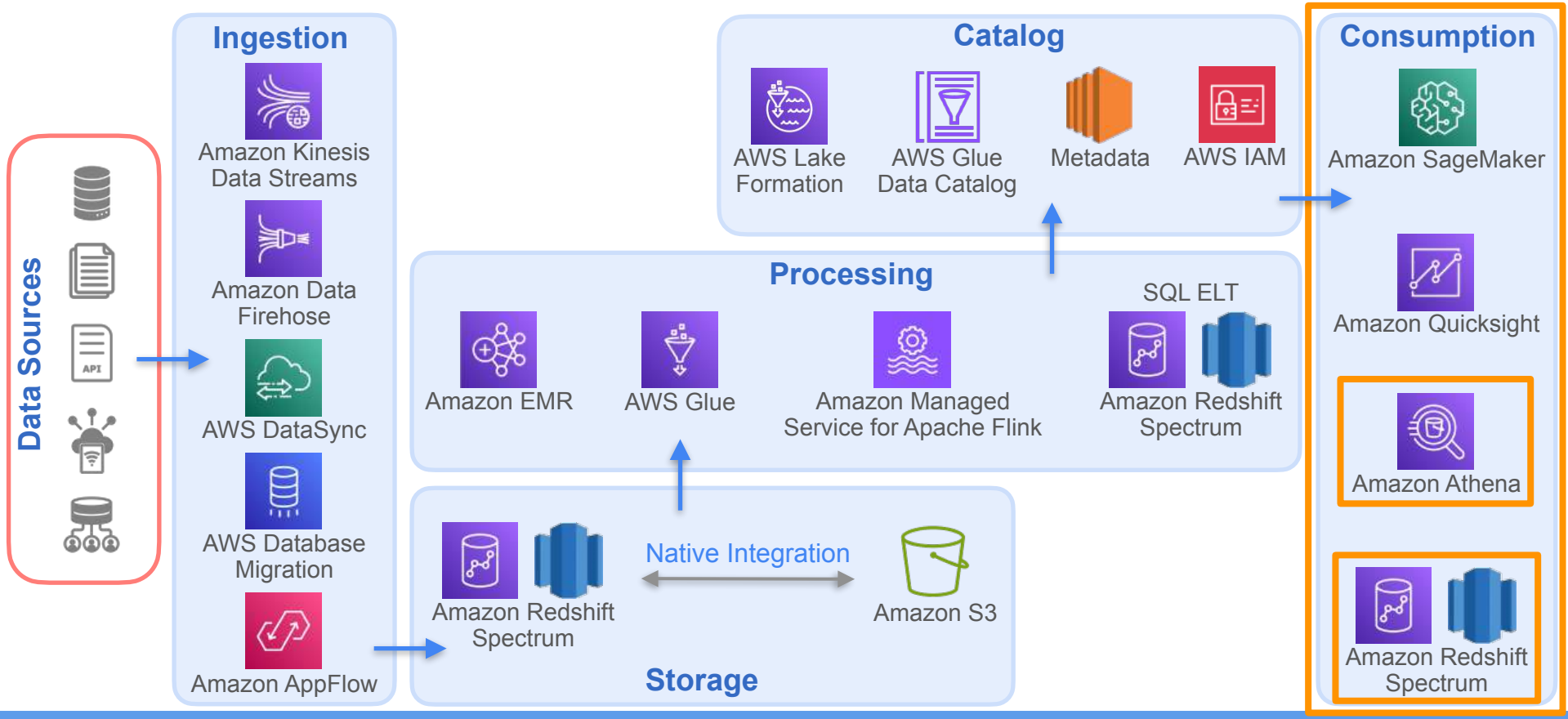
AWS Data Lakehouse



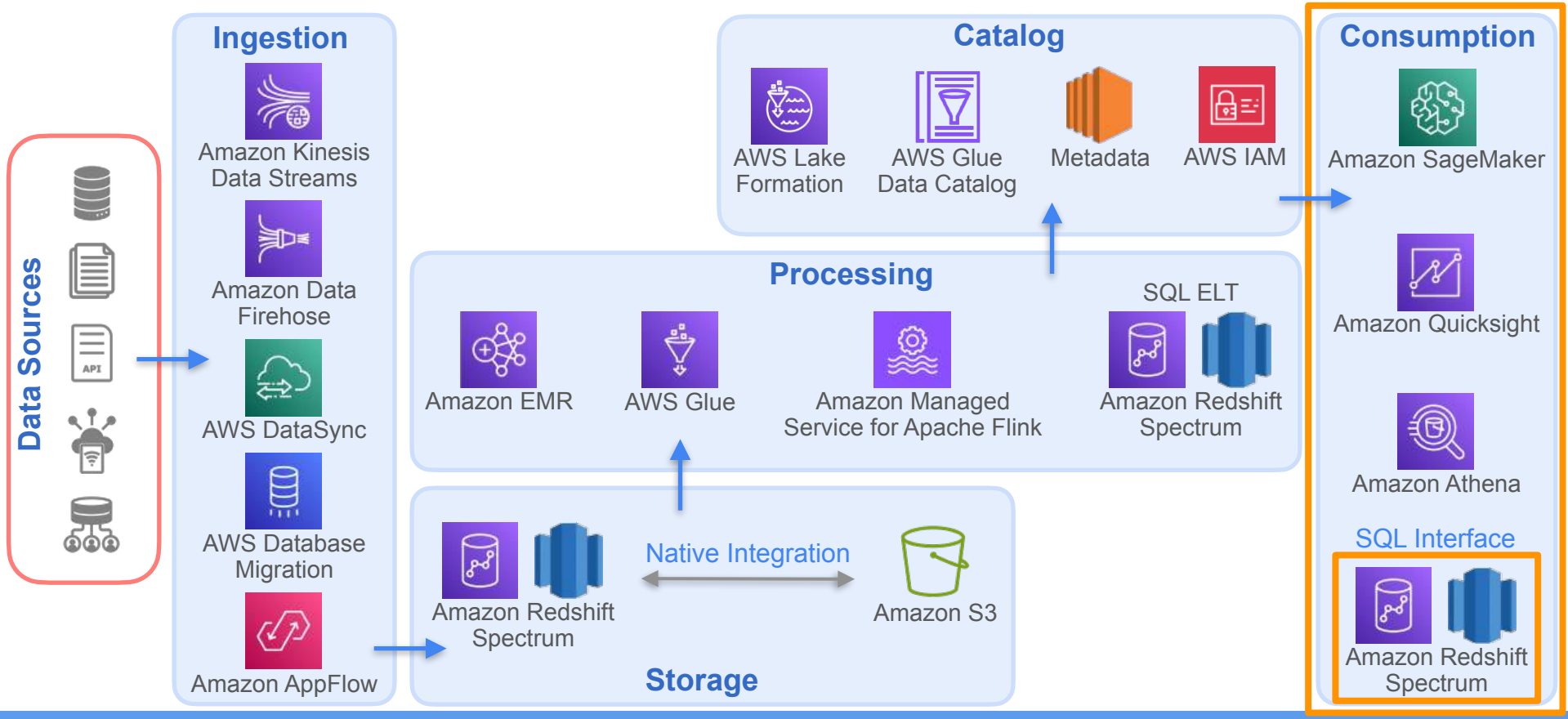
AWS Data Lakehouse



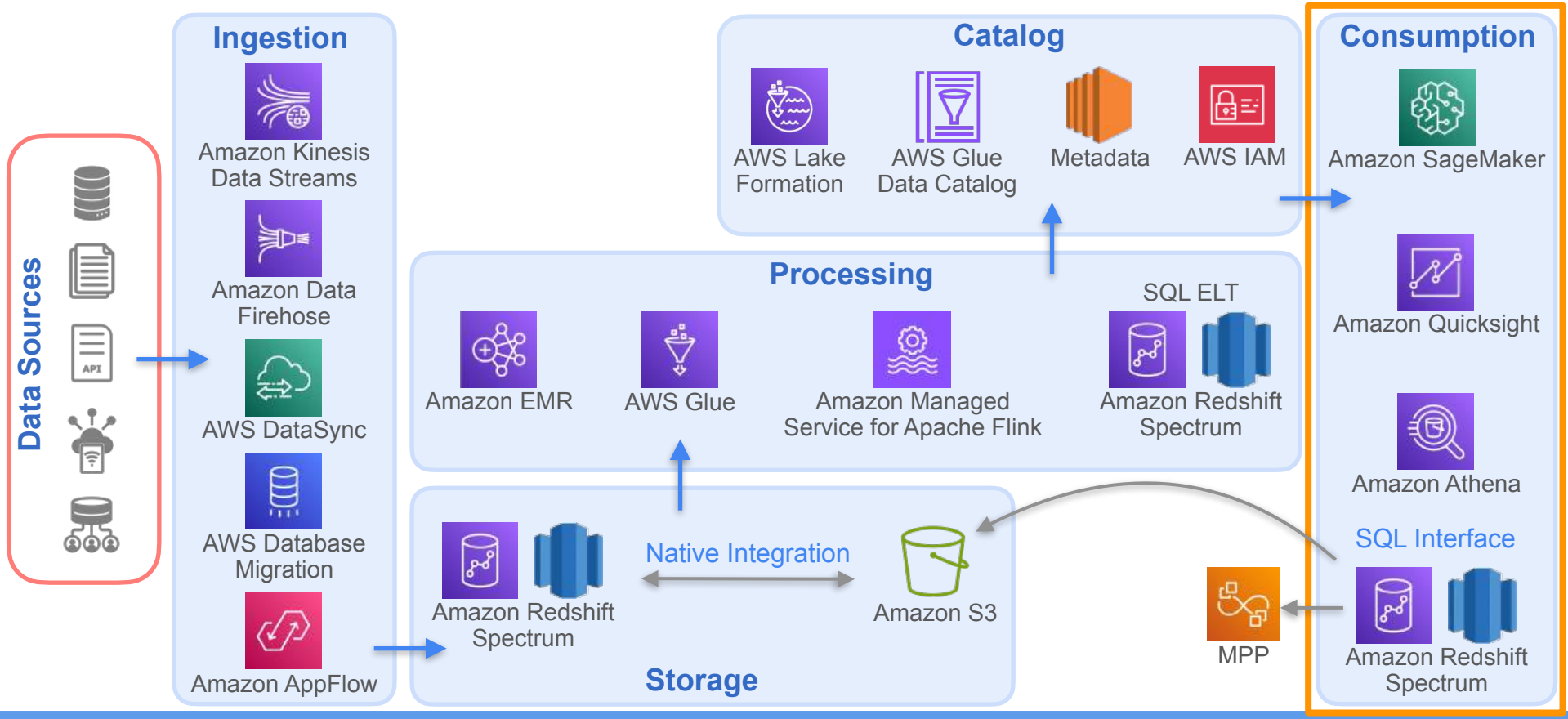
AWS Data Lakehouse



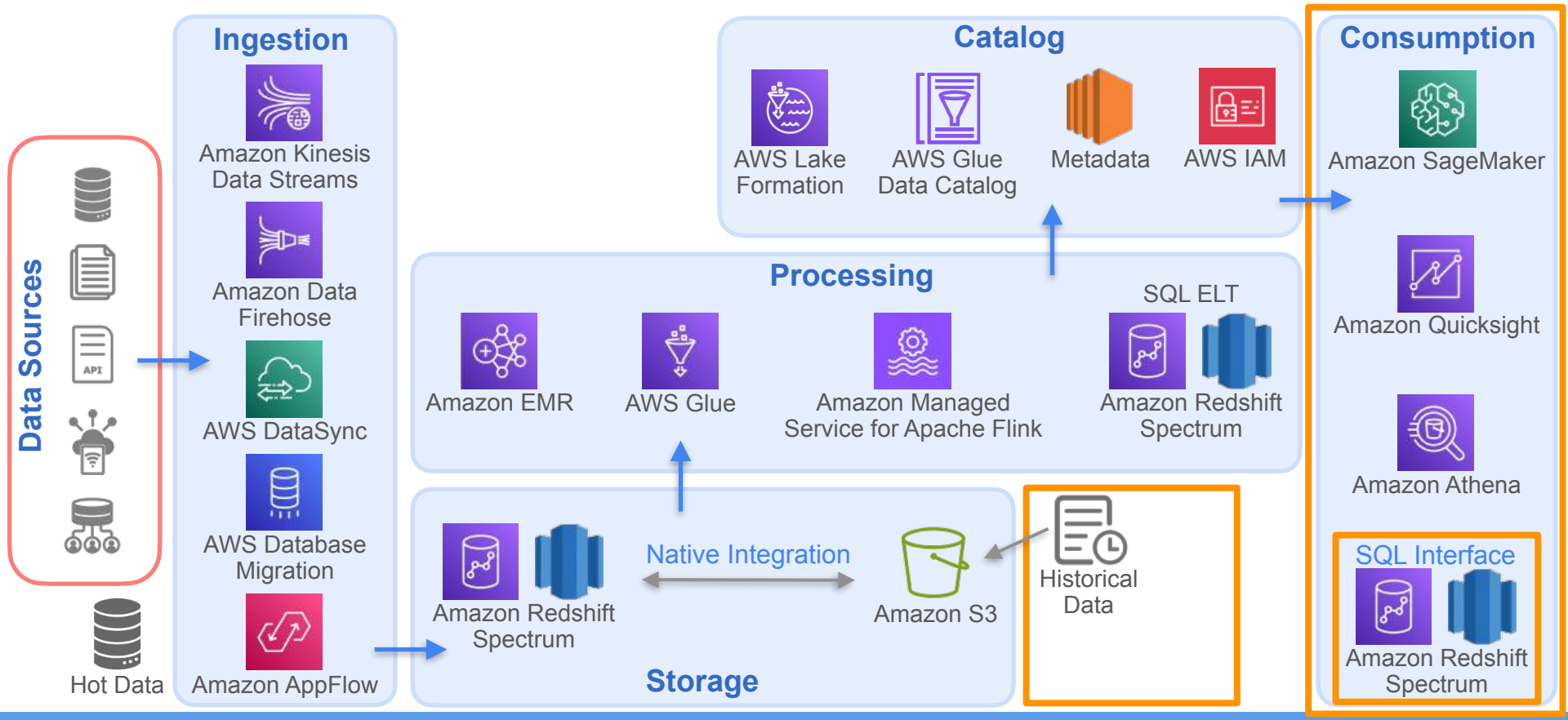
AWS Data Lakehouse



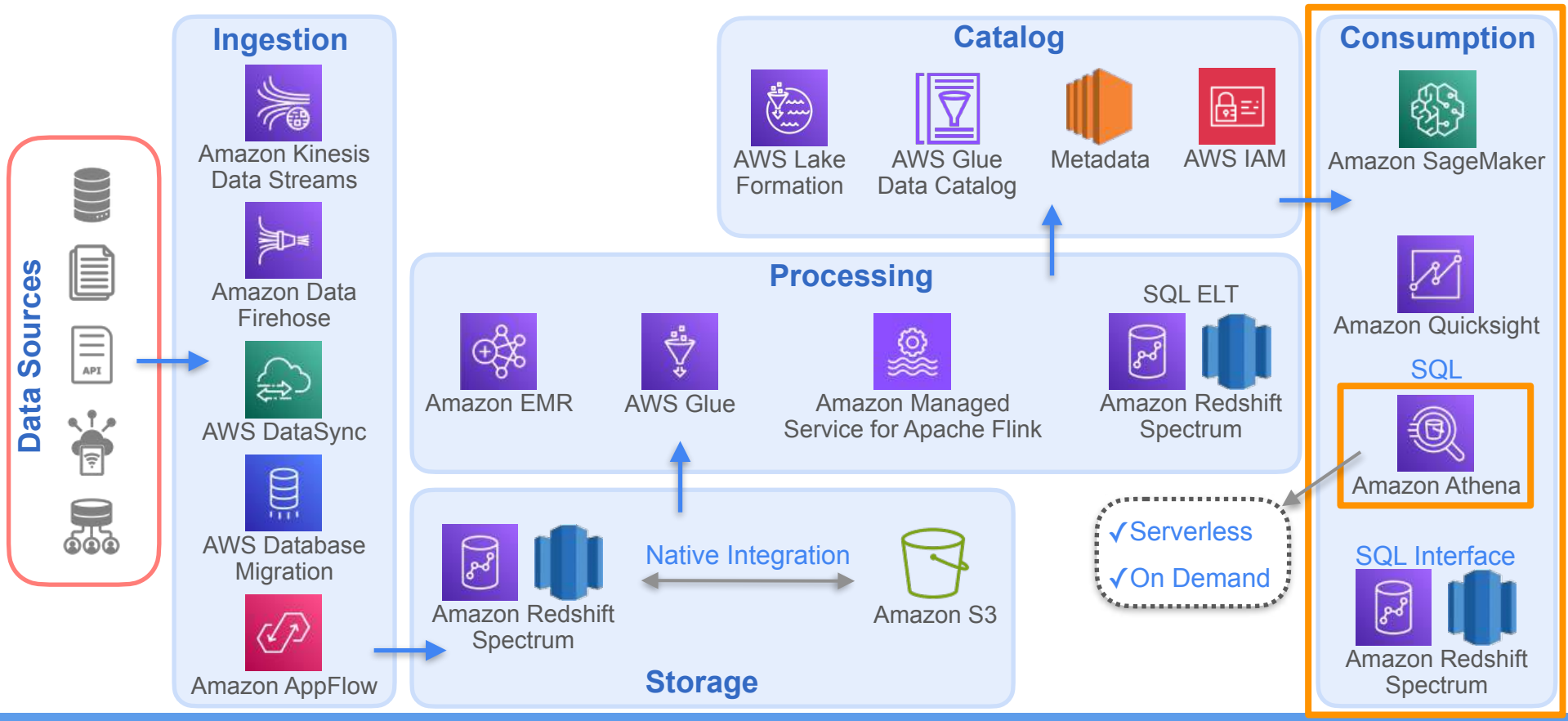
AWS Data Lakehouse



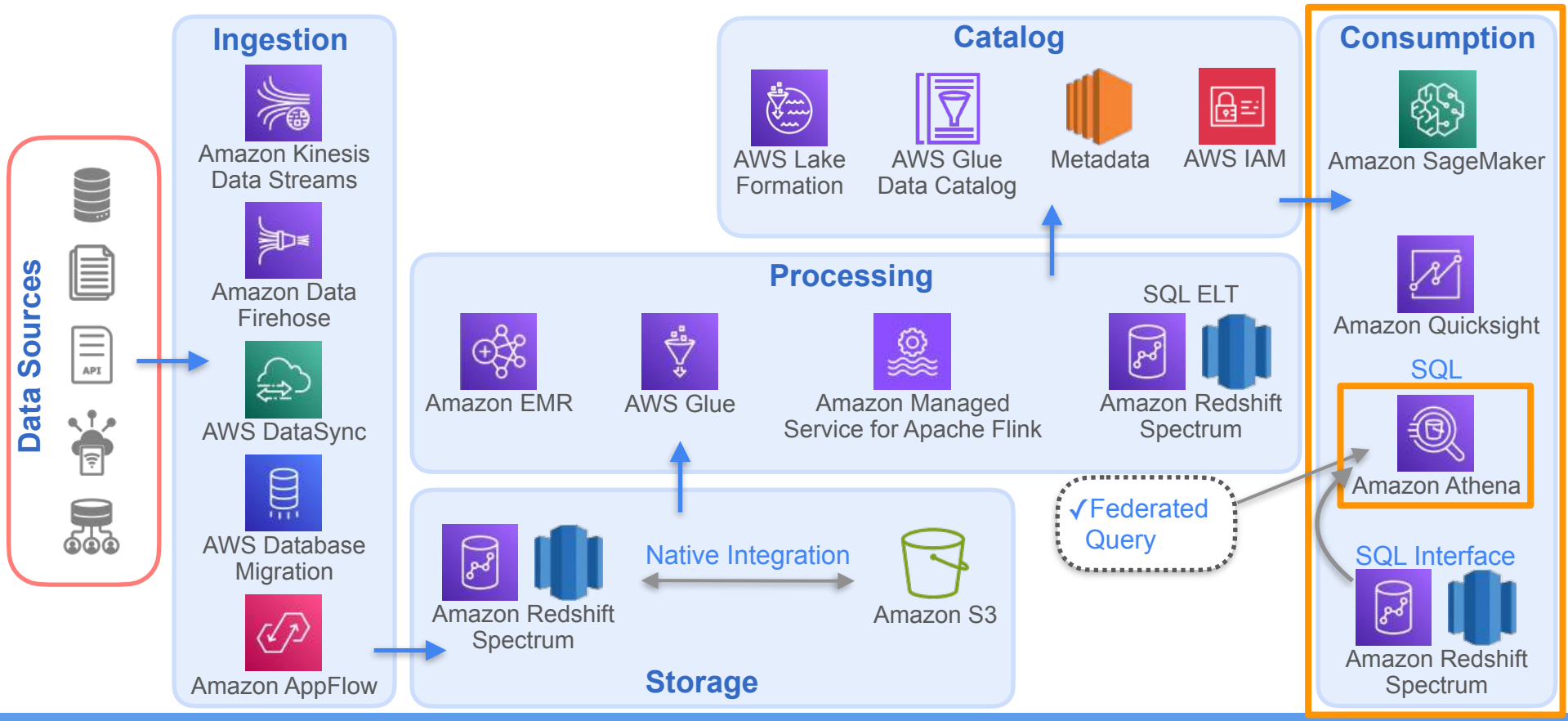
AWS Data Lakehouse



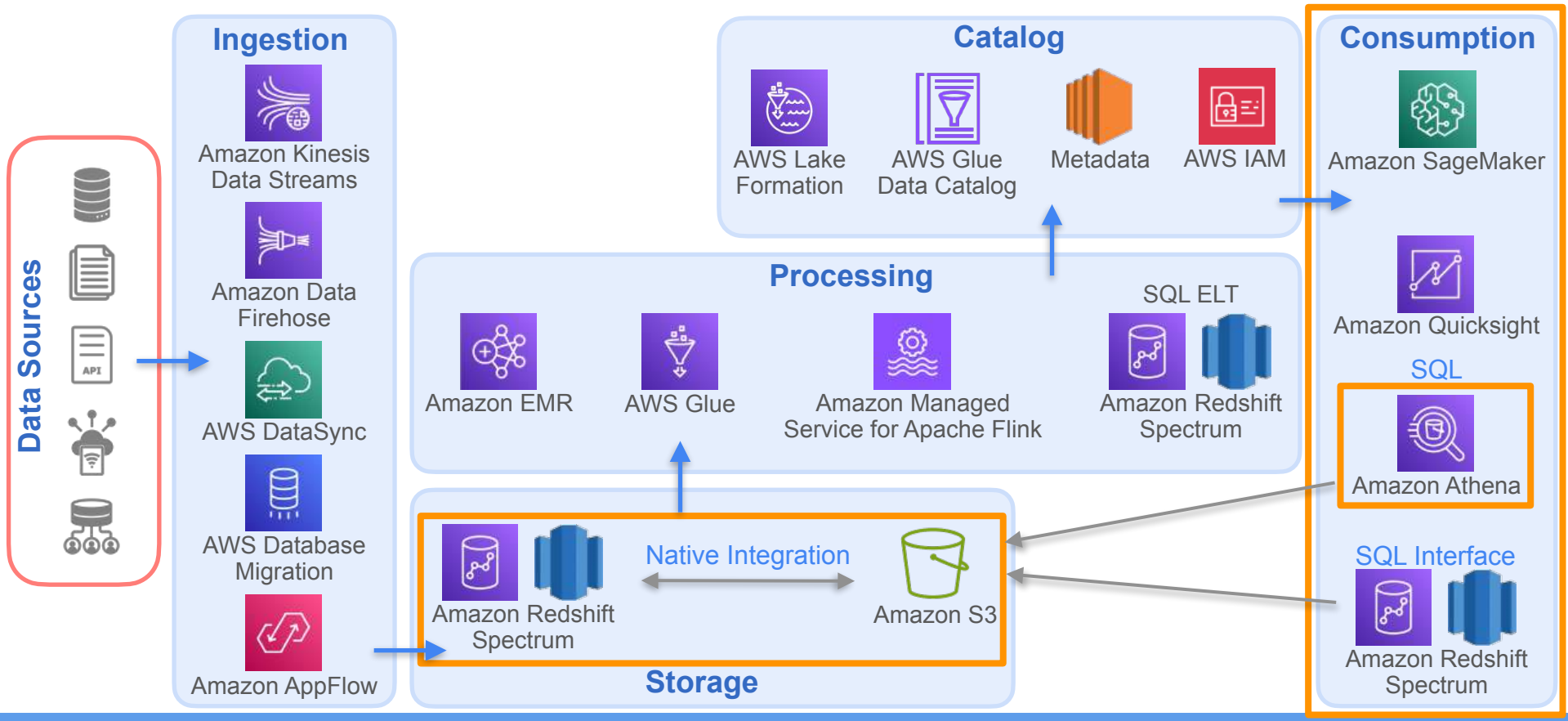
AWS Data Lakehouse



AWS Data Lakehouse



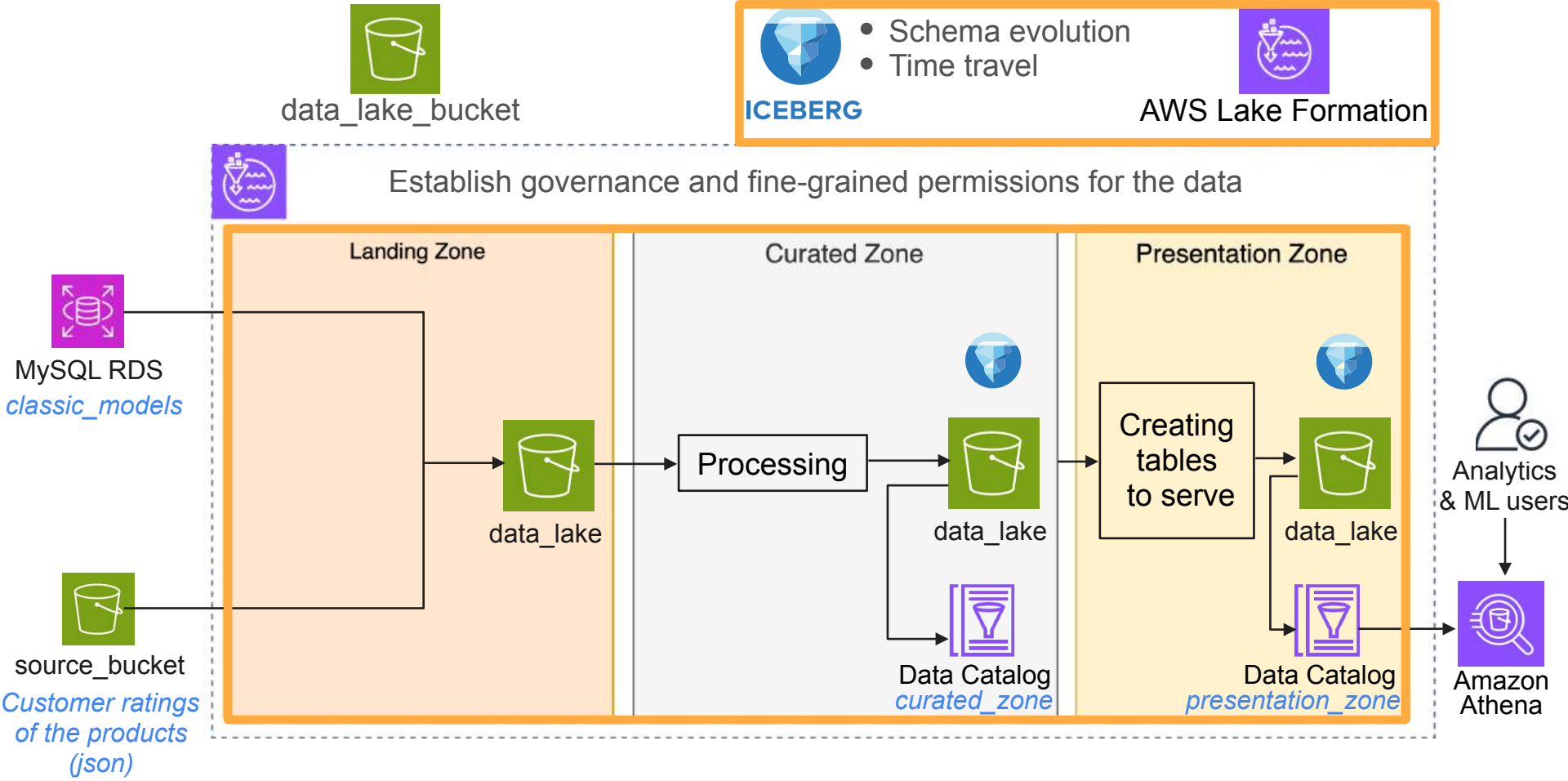
AWS Data Lakehouse



DeepLearning.AI

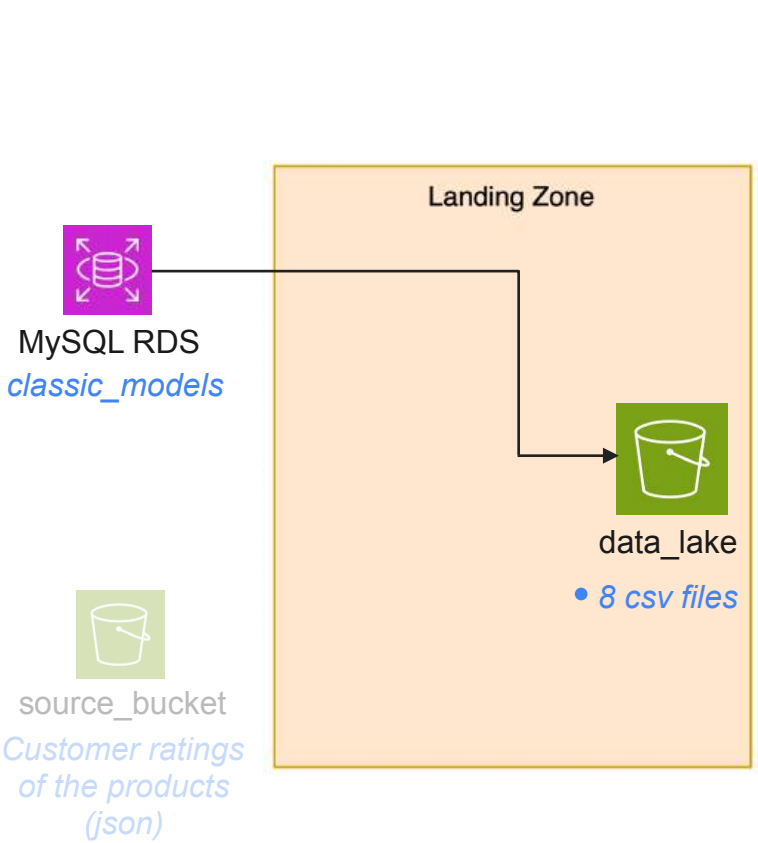
Lab Walkthrough

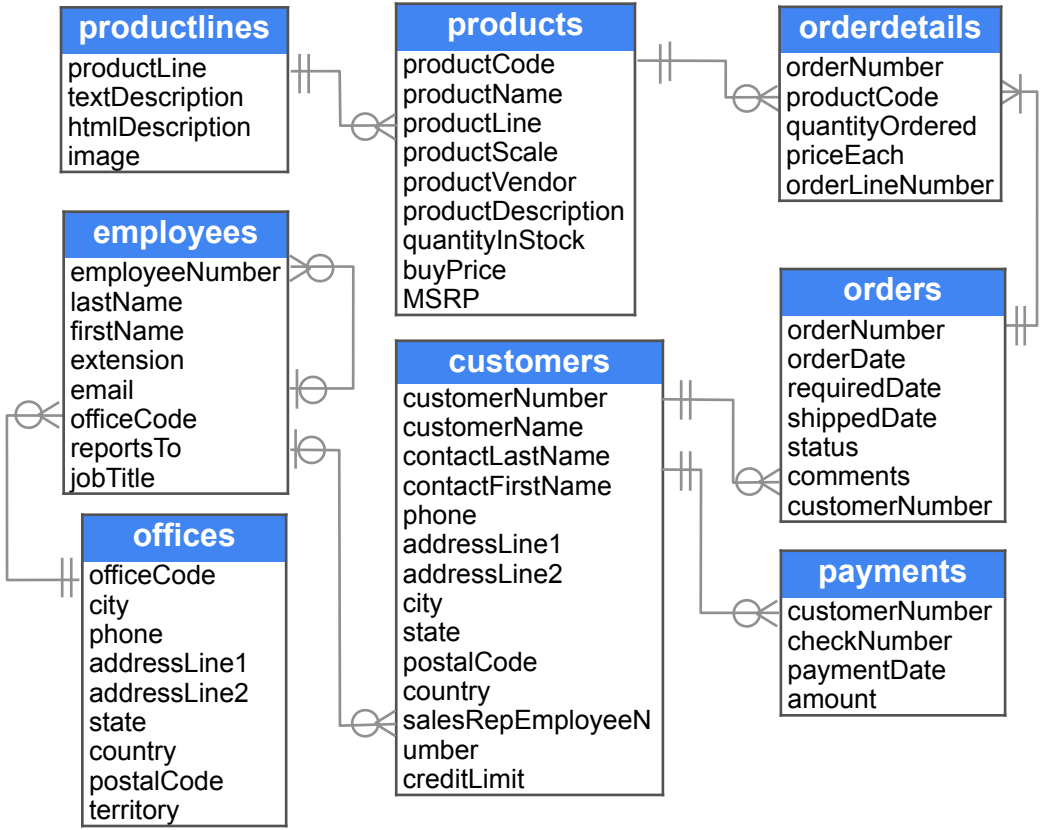
Building a Data Lakehouse with AWS LakeFormation (Part 1)


data_lake_bucket
ICEBERG
Schema evolution
Time travel
AWS Lake Formation
Establish governance and fine-grained permissions for the data
Landing Zone
Curated Zone
Presentation Zone
MySQL RDS
classic_models
source_bucket
Customer ratings of the products (json)
data_lake
Processing
data_lake
Data Catalog
curated_zone
Creating tables to serve
data_lake
Data Catalog
presentation_zone
Analytics & ML users
Amazon Athena

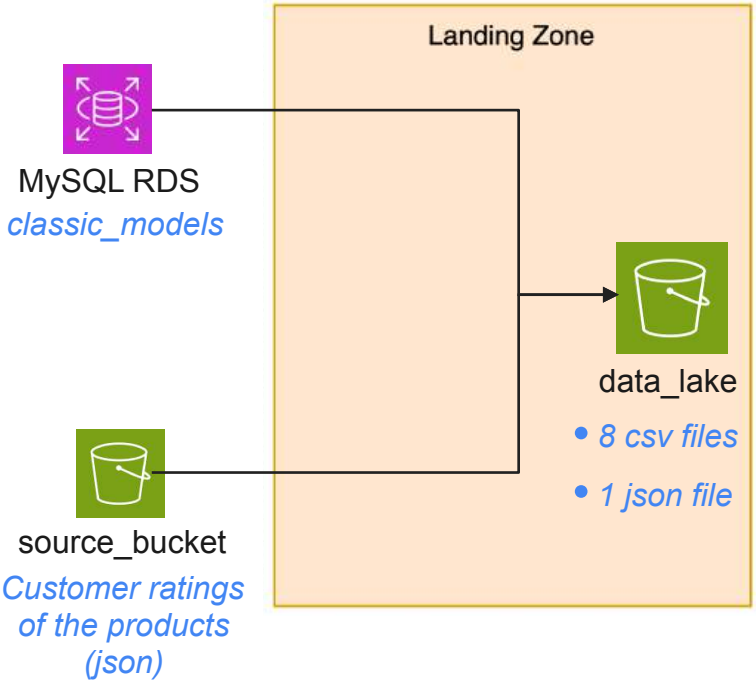
- Each csv file:

  s3://{data_lake_bucket}/landing_zone/rds/{table_name}





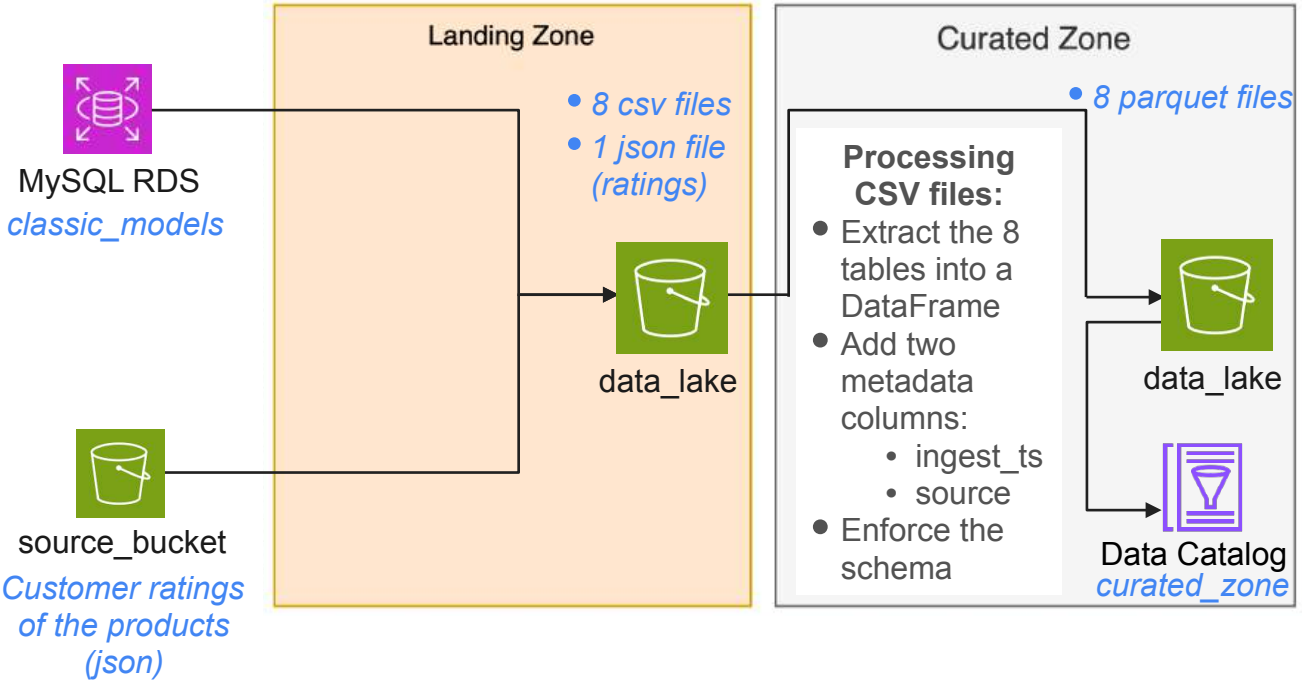
- Each csv file:
  s3://{data_lake_bucket}/landing_zone/rds/{table_name}
- JSON file:
  s3://{data_lake_bucket}/landing_zone/json/ratings




```
[
  {
    "customerNumber": 103,
    "productCode": "S18_1589",
    "productRating": 3
  },
  {
    "customerNumber": 103,
    "productCode": "S18_2870",
    "productRating": 5
  },
  {
    "customerNumber": 103,
    "productCode": "S18_3685",
    "productRating": 4
  },
  {
    "customerNumber": 103,
    "productCode": "S24_1628",
    "productRating": 3
  },
]
```

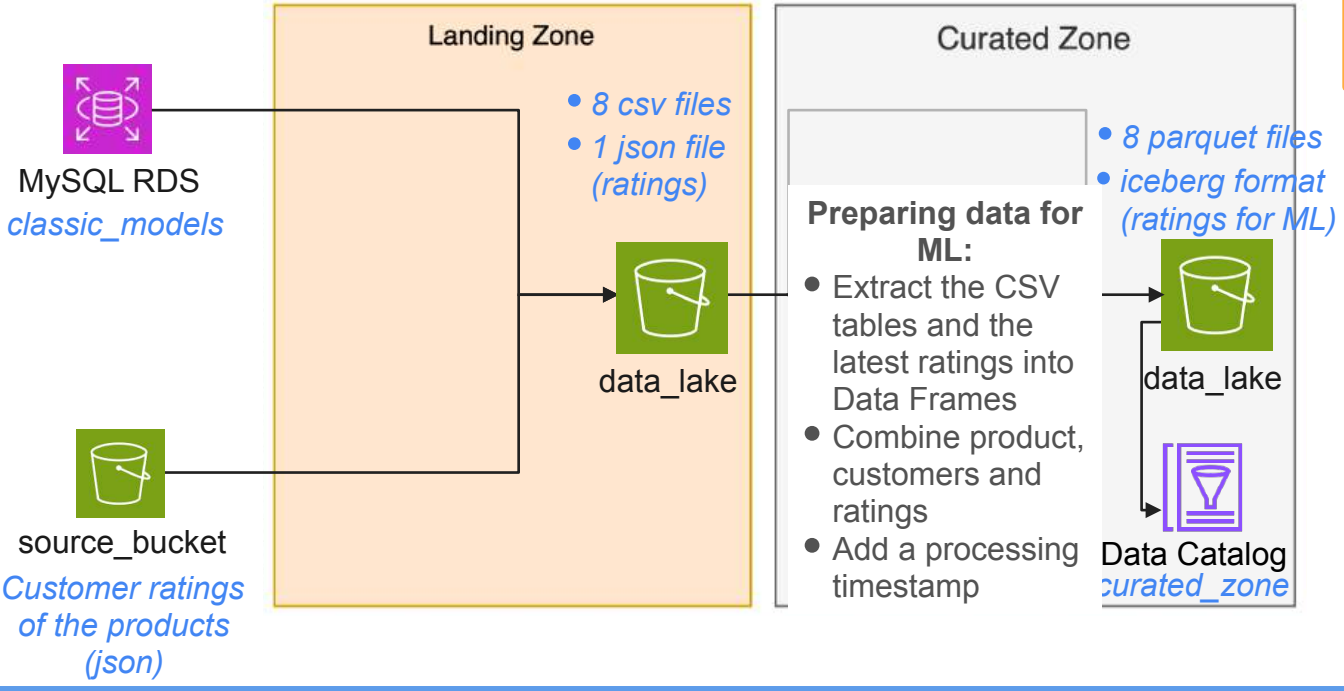

customer Number	product Code	product Rating	ingest_ts

- Each parquet file:
  s3://{data_lake_bucket}/curated_zone/{table_name}



You are provided with a schema for the processed tables.

- Each parquet file:
  s3://{data_lake_bucket}/curated_zone/{table_name}
- Iceberg data:
  s3://{data_lake_bucket}/curated_zone/ratings_for_ML/iceberg

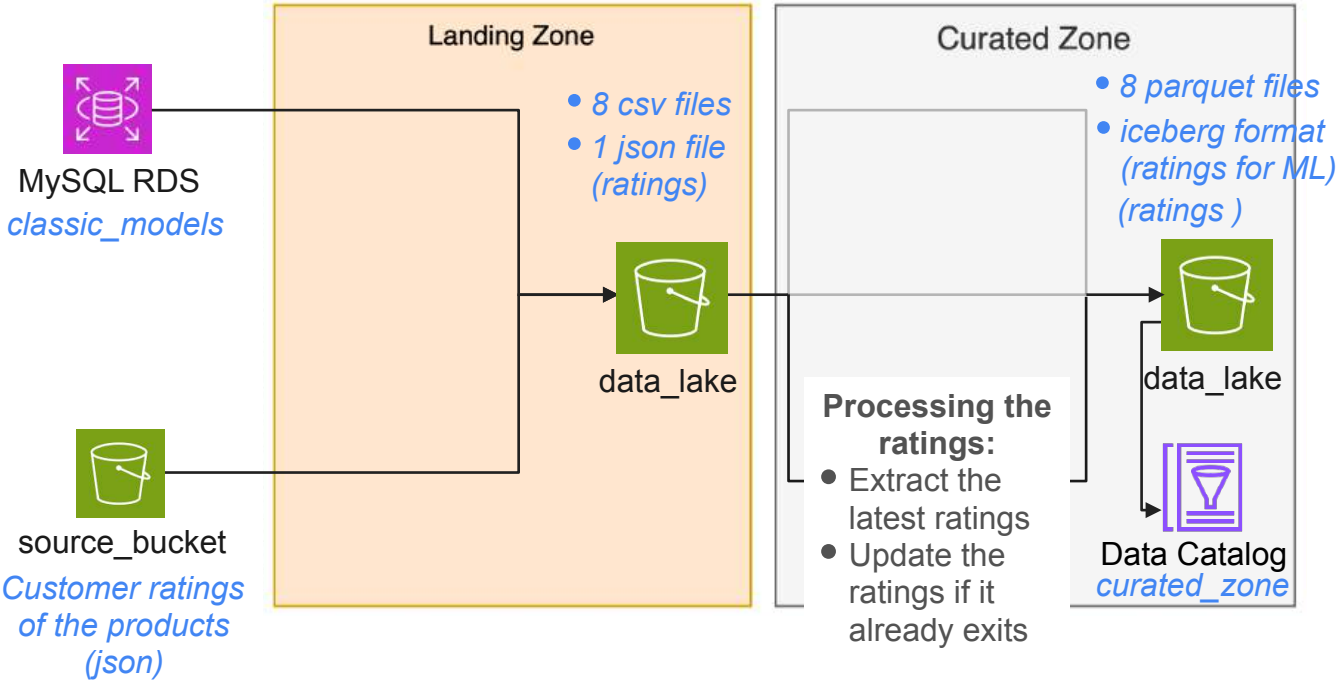


products
productCode
productName
productLine
productScale
productVendor
productDescription
quantityInStock
buyPrice
MSRP

customers
customerNumber
customerName
contactLastName
contactFirstName
phone
addressLine1
addressLine2
city
state
postalCode
country
salesRepEmployeeNumber
creditLimit

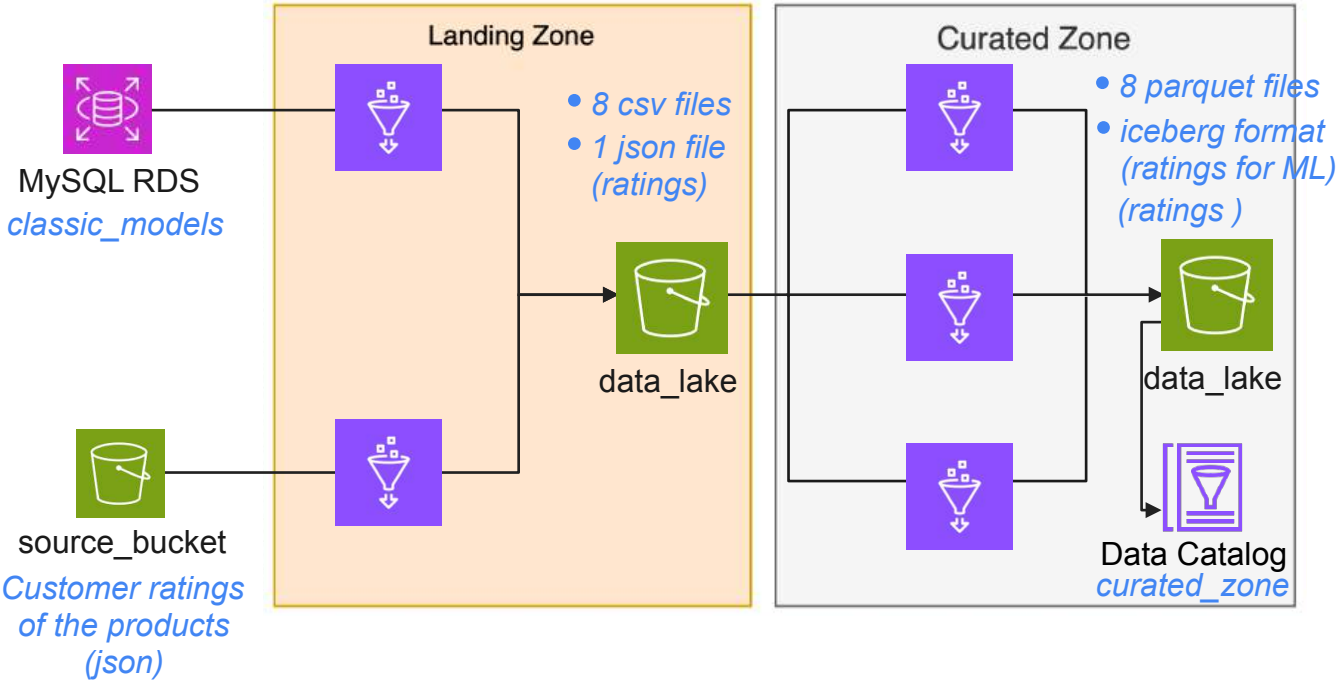

```
[
  {
    "customerNumber": 103,
    "productCode": "S18_1589",
    "productRating": 3
  },
  {
    "customerNumber": 103,
    "productCode": "S18_2870",
    "productRating": 5
  },
  {
    "customerNumber": 103,
    "productCode": "S18_3685",
    "productRating": 4
  },
  {
    "customerNumber": 103,
    "productCode": "S24_1628",
    "productRating": 3
  },
```

- Each parquet file:
  s3://{data_lake_bucket}/curated_zone/{table_name}
- Iceberg data:
  s3://{data_lake_bucket}/curated_zone/ratings_for_ML/iceberg
  s3://{data_lake_bucket}/curated_zone/ratings/iceberg




```
[
  {
    "customerNumber": 103,
    "productCode": "S18_1589",
    "productRating": 3
  },
  {
    "customerNumber": 103,
    "productCode": "S18_2870",
    "productRating": 5
  },
  {
    "customerNumber": 103,
    "productCode": "S18_3685",
    "productRating": 4
  },
  {
    "customerNumber": 103,
    "productCode": "S24_1628",
    "productRating": 3
  },
]
```

Terraform

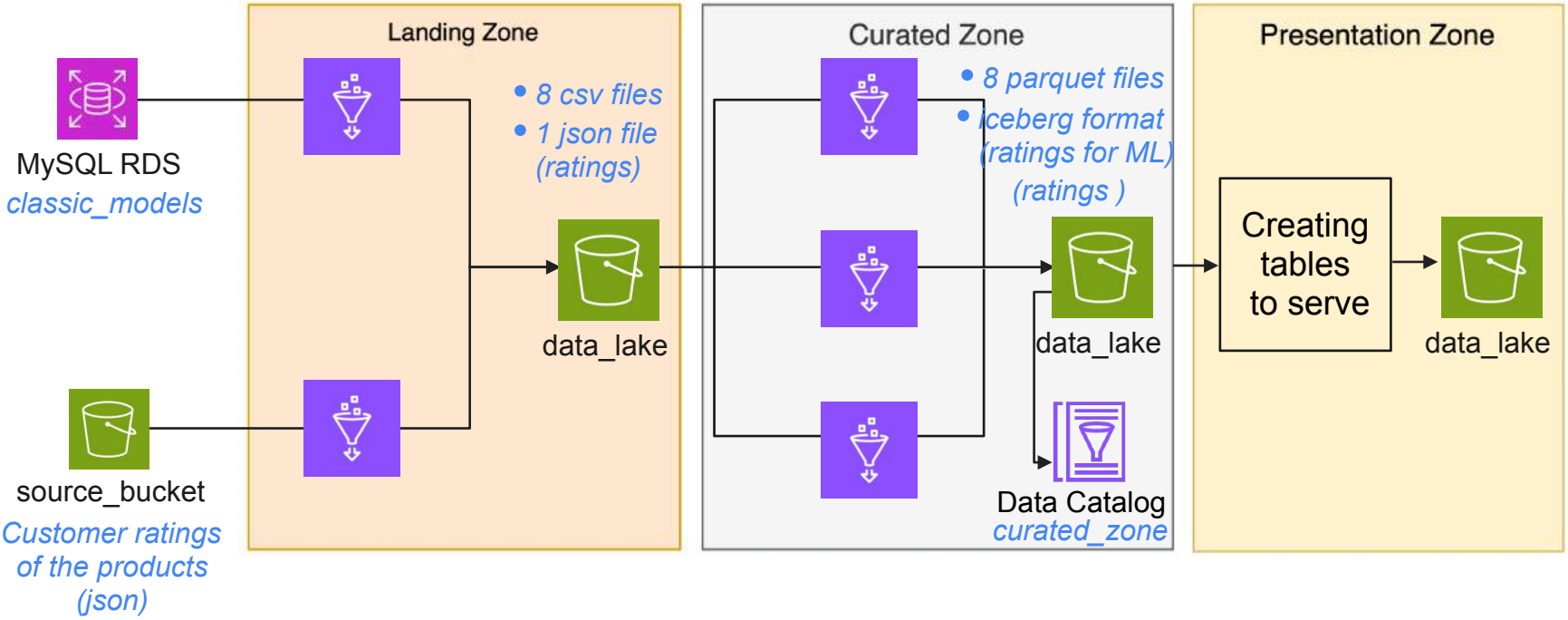

Landing Zone
Curated Zone
MySQL RDS
classic_models
source_bucket
Customer ratings of the products (json)
• 8 csv files
• 1 json file (ratings)
data_lake
• 8 parquet files
• iceberg format (ratings for ML) (ratings)
data_lake
Data Catalog
curated_zone

For analytics end users:

- Average sales grouped by year and month
- Average ratings per product
- Latest ratings

For ML end users:

- Ratings for ML table

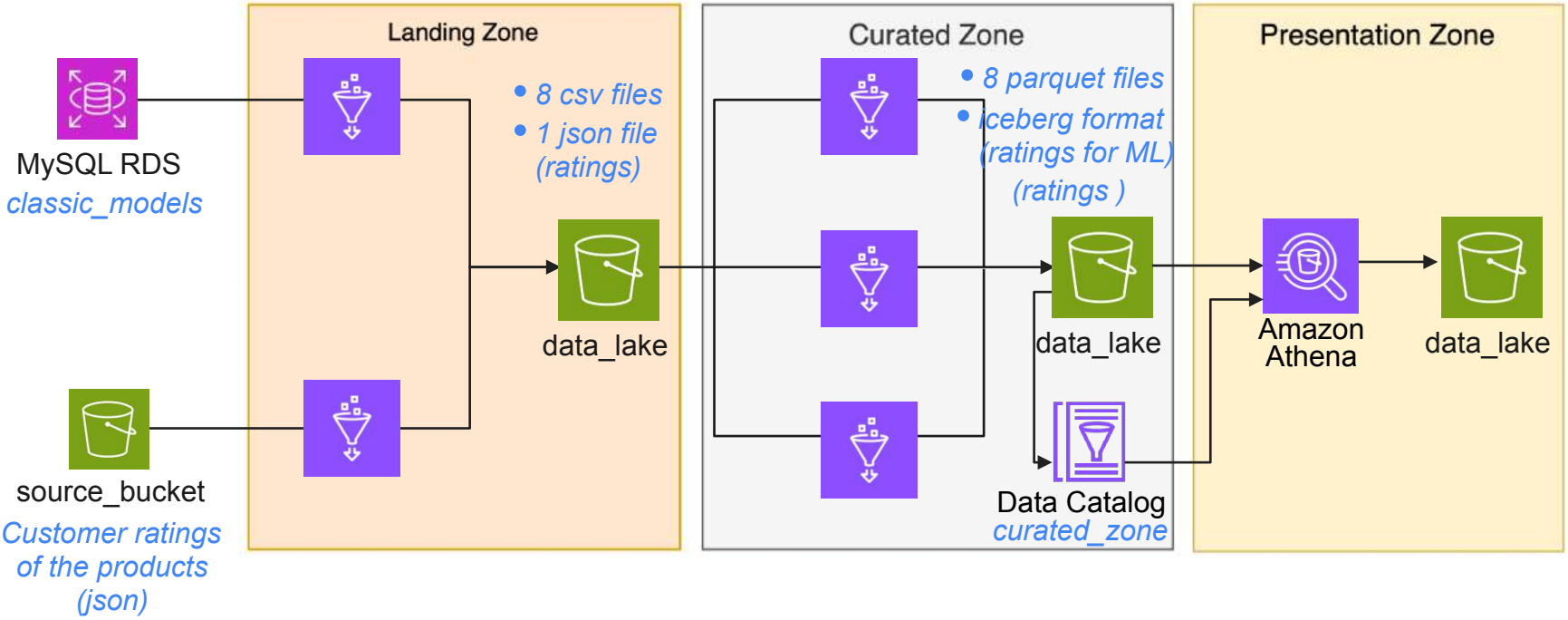


For analytics end users:

- Average sales grouped by year and month
- Average ratings per product
- Latest ratings

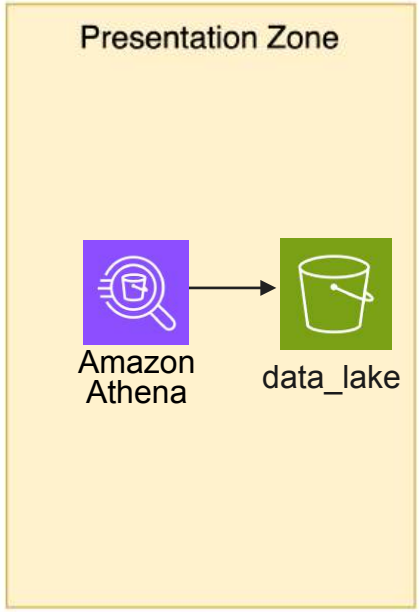
For ML end users:

- Ratings for ML table



Creating the ratings table using Athena

```
ctas_query = f"""CREATE TABLE ratings WITH (
    table_type='ICEBERG',
    location='s3://{BUCKET_NAME}/presentation_zone/ratings',
    is_external=false
) AS SELECT * FROM {CURATED_DATABASE_NAME}.ratings;
"""
response = wr.athena.start_query_execution(
    sql=ctas_query,
    database=PRESENTATION_DATABASE_NAME,
    wait=True,
    s3_output = f's3://{BUCKET_NAME}/athena_output/ratings'
)
print(response['Status']['State'])
```

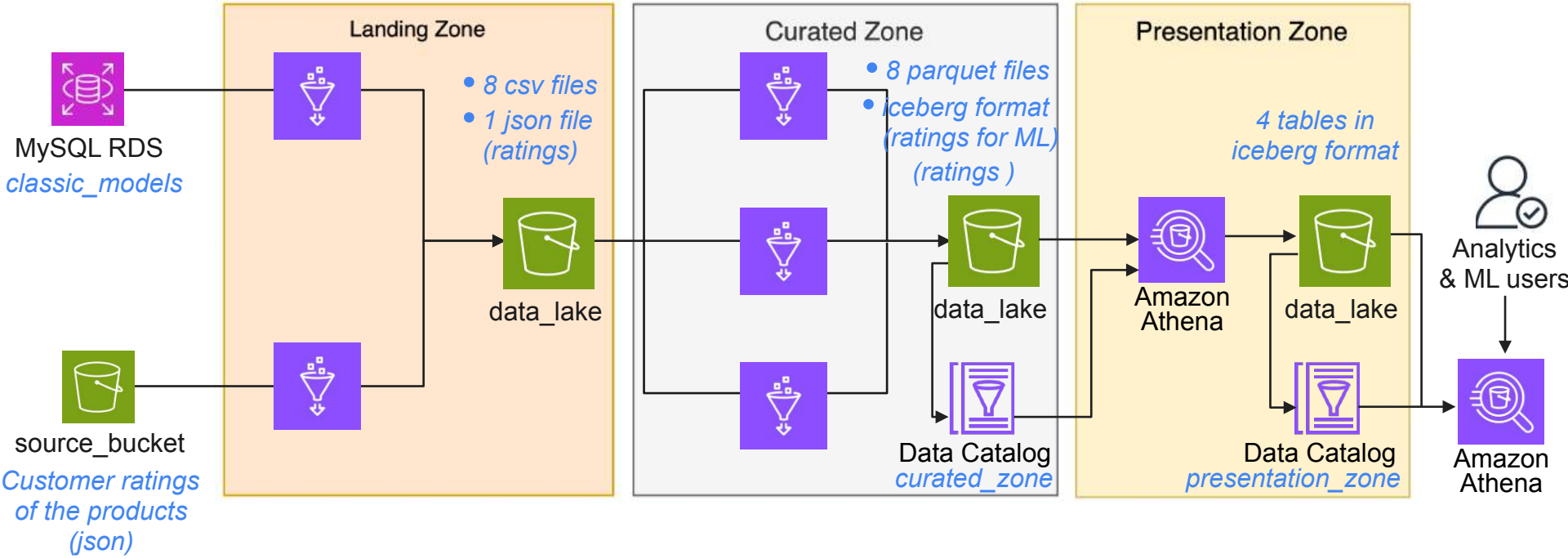


For analytics end users:

- Average sales grouped by year and month
- Average ratings per product
- Latest ratings

For ML end users:

- Ratings for ML table



DeepLearning.AI

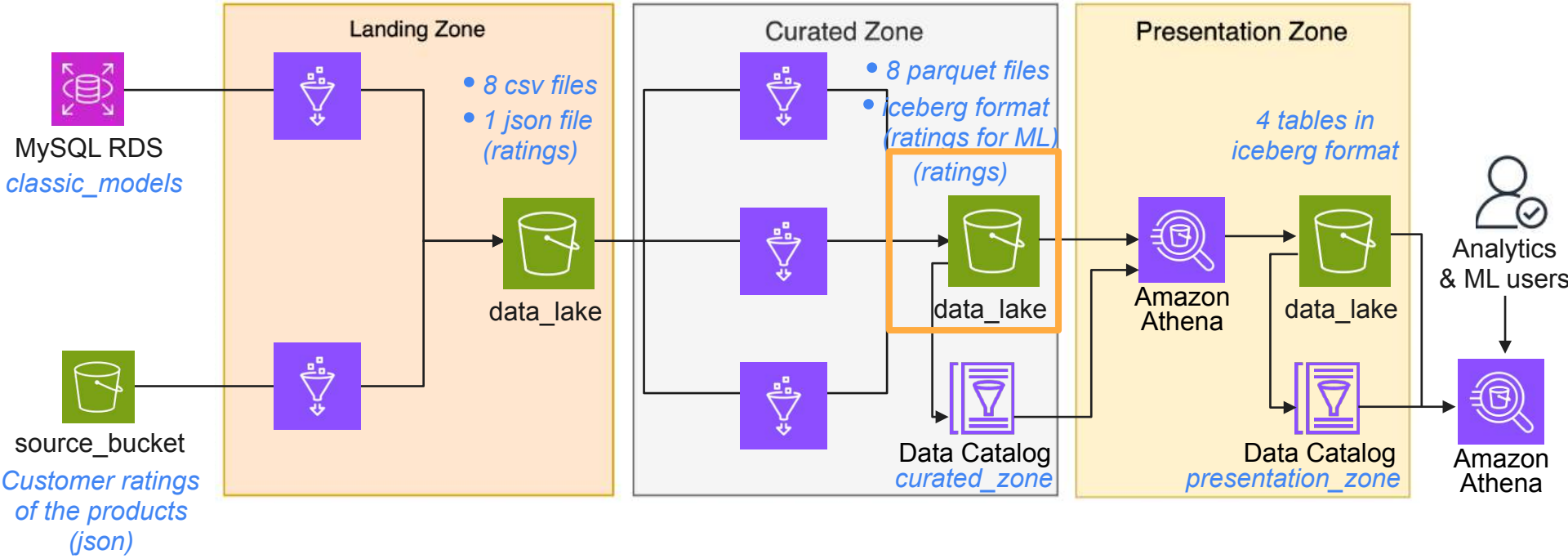
Lab Walkthrough

Building a Data Lakehouse with AWS LakeFormation (Part 2)

ICEBERG

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/**metadata**

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/**data**



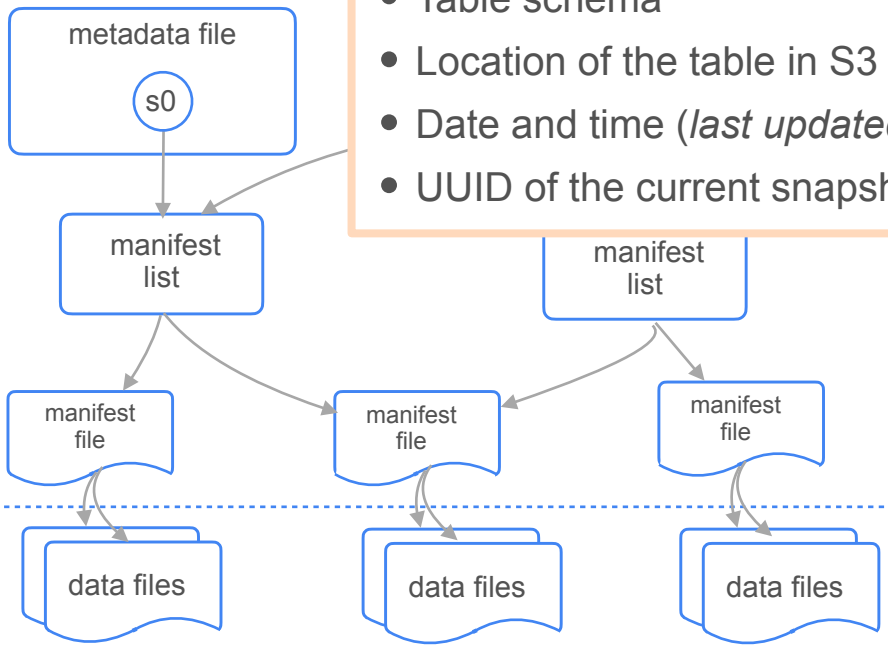
ICEBERG

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/metadata

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/data

Metadata layer

/metadata



- Table schema
- Location of the table in S3
- Date and time (*last updated*)
- UUID of the current snapshot

Name	Type
00000-62d700ed-e8c0-4c4a-a413-e2089b393554.metadata.json	json
816017ad-b434-49fb-86ac-e302da92bbf2-m0.avro	avro
snap-1643666583157488463-1-816017ad-b434-49fb-86ac-e302da92bbf2.avro	avro

Storage layer

/data

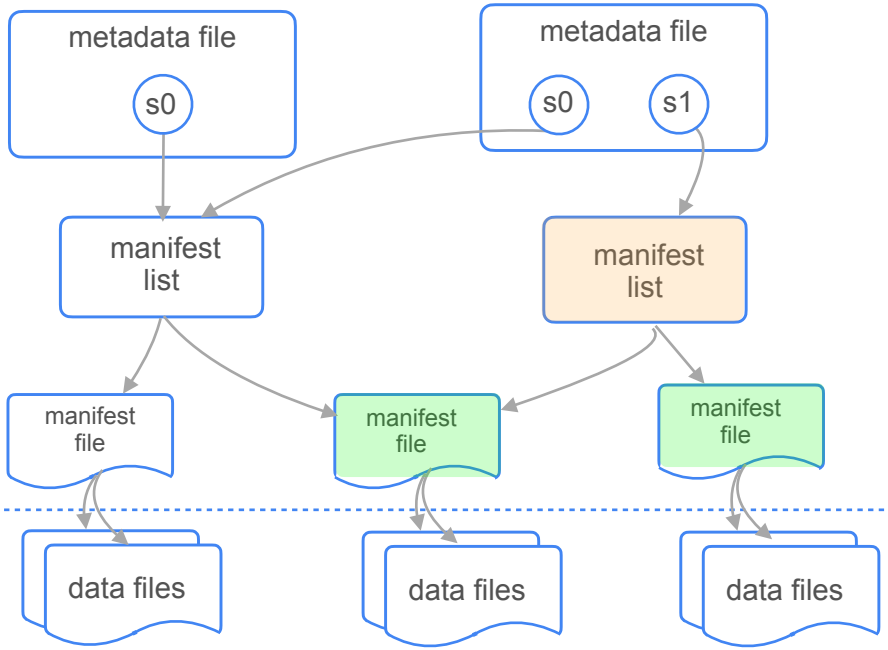
ICEBERG

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/metadata

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/data

Metadata layer

/metadata



Storage layer

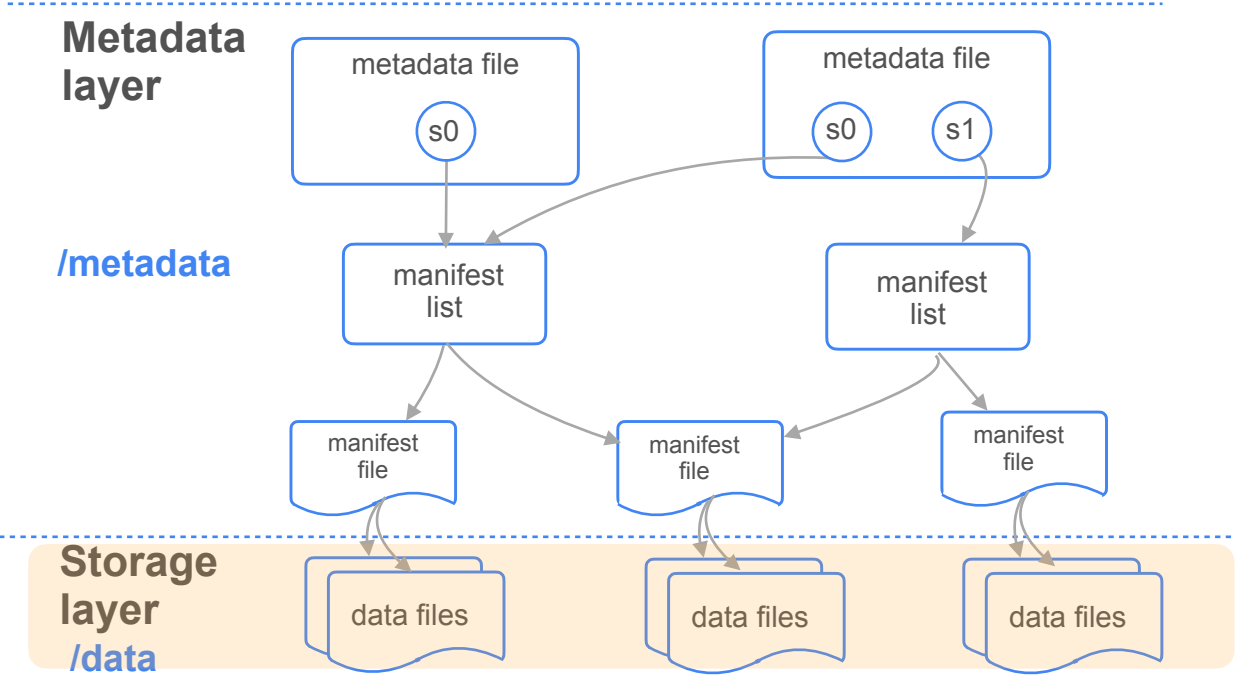
/data

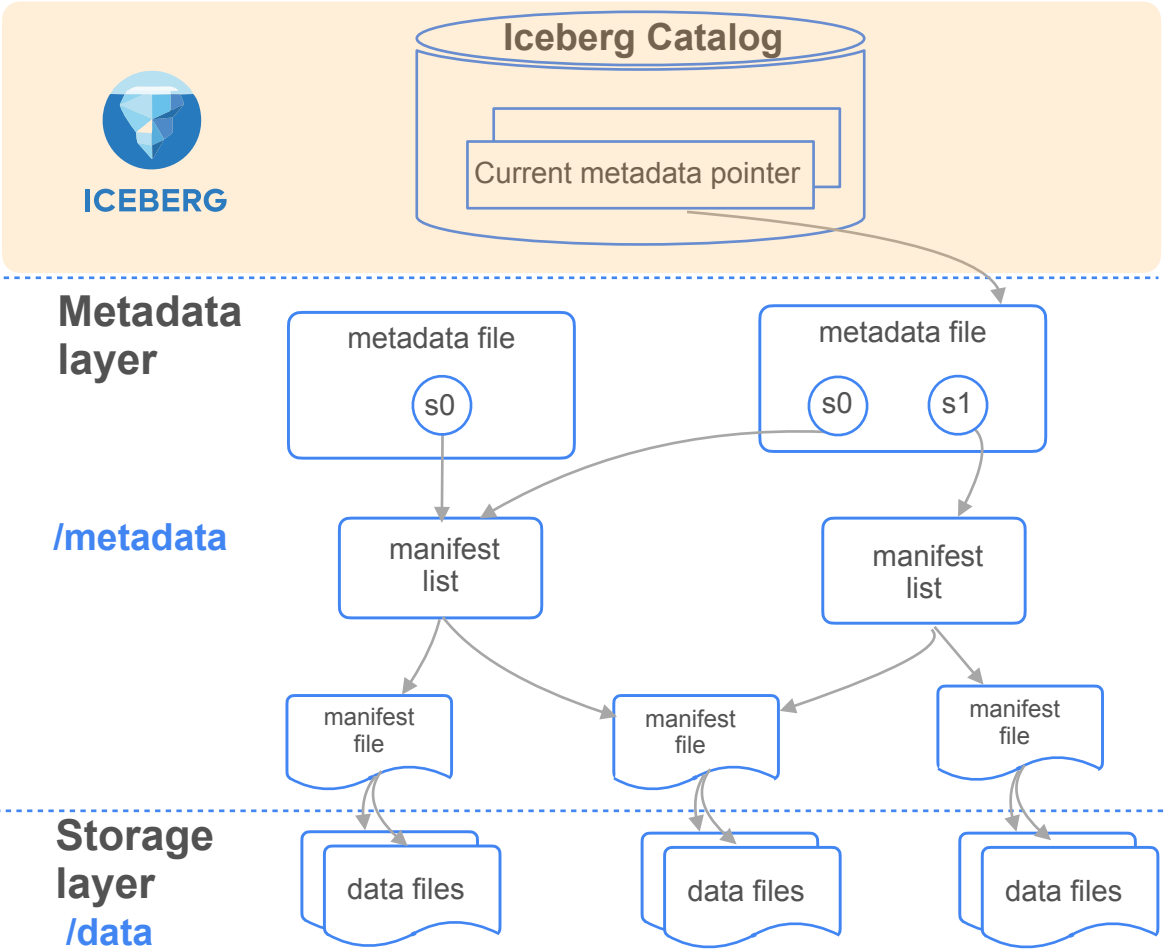
Name	Type
00000-62d700ed-e8c0-4c4a-a413-e2089b393554.metadata.json	json
816017ad-b434-49fb-86ac-e302da92bbf2-m0.avro	avro
snap-1643666583157488463-1-816017ad-b434-49fb-86ac-e302da92bbf2.avro	avro

ICEBERG

s3://{data_lake_bucket}/curated_zone/ratings/iceberg/metadata

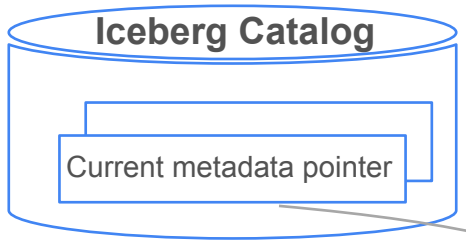
s3://{data_lake_bucket}/curated_zone/ratings/iceberg/data





Glue Data Catalog

- Contains a catalog table for each iceberg file in the curated and presentation zone
- Catalog tables are organized:
  - `curated_zone` database
  - `presentation_zone` database



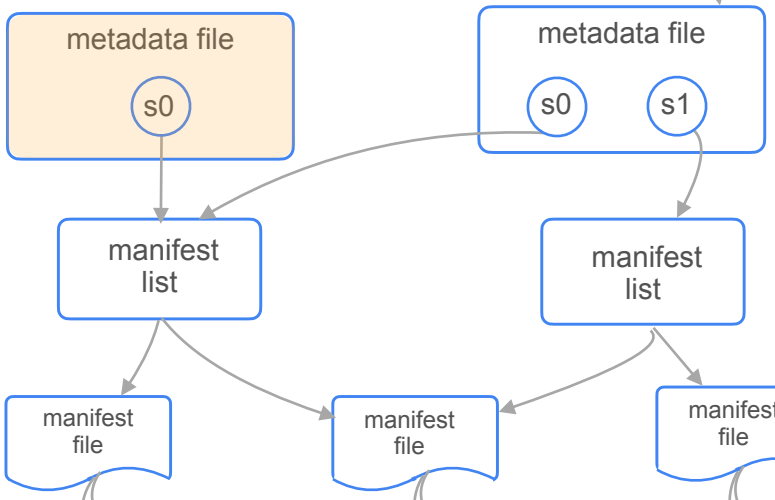
Schema evolution

- Add a new column to the ratings table at:

  s3://{data_lake_bucket}/curated_zone/ratings/iceberg
- Apply the transformation in Terraform using the “alter_table” module

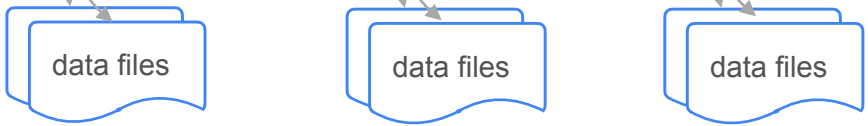
Metadata layer

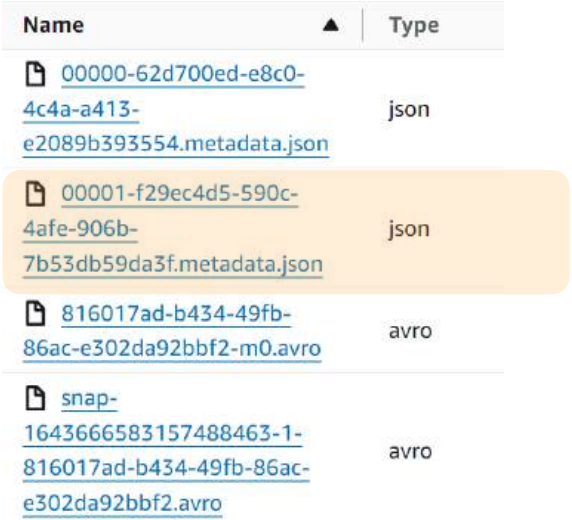
/metadata



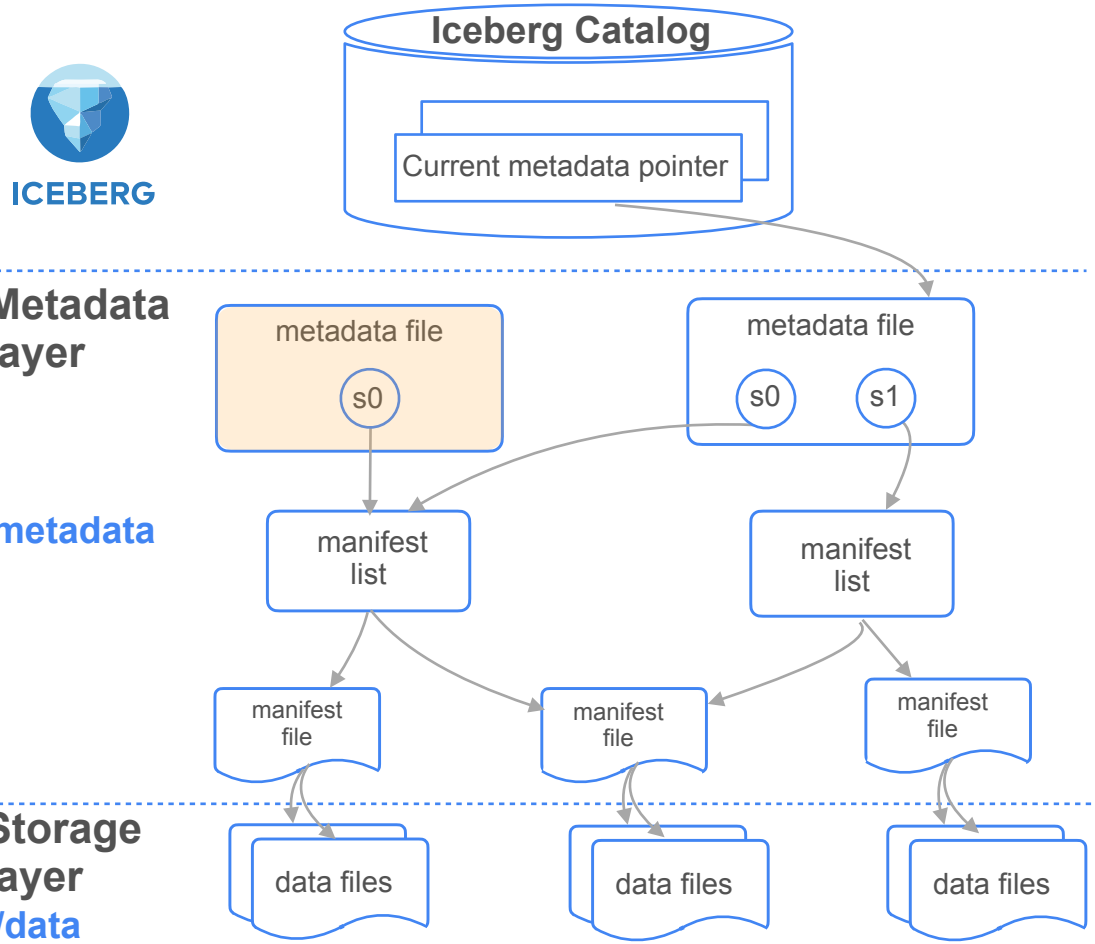
Storage layer

/data





The metadata file will change without the need to completely re-write the data

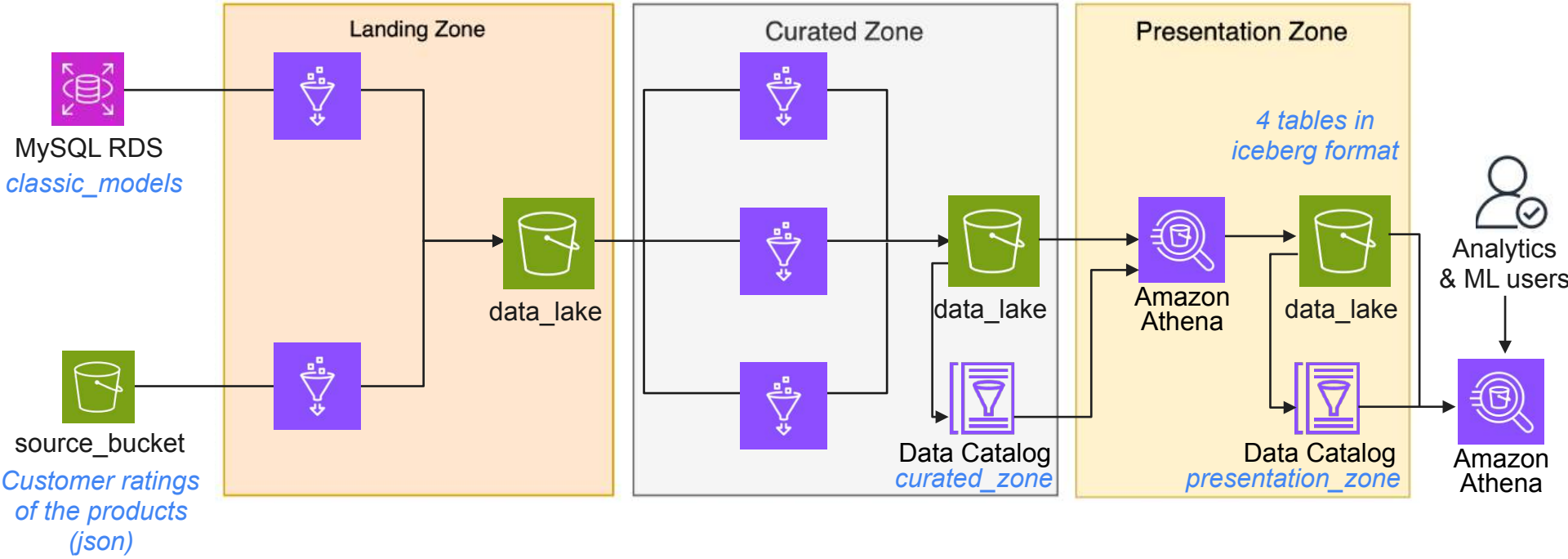


Time Travel

Query the new and old versions of the ratings table.

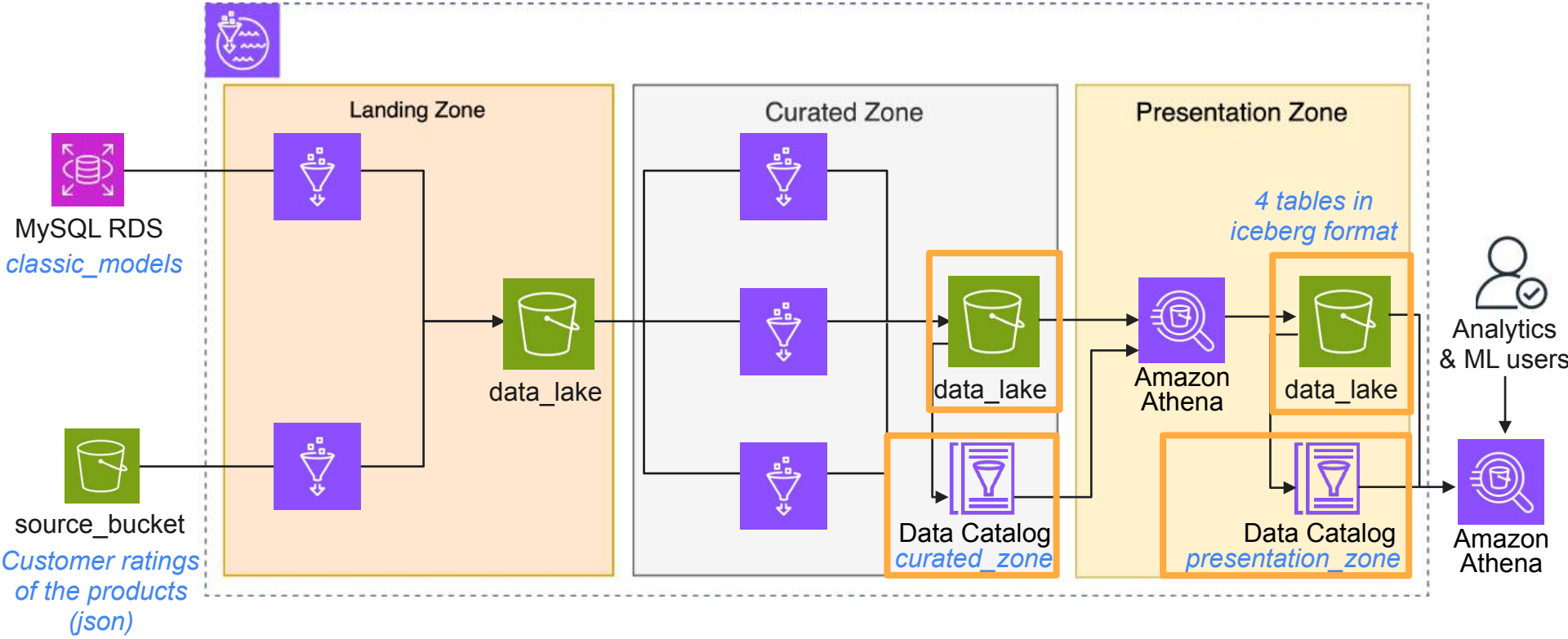
Name	Type
00000-62d700ed-e8c0-4c4a-a413-e2089b393554.metadata.json	json
00001-f29ec4d5-590c-4afe-906b-7b53db59da3f.metadata.json	json
816017ad-b434-49fb-86ac-e302da92bbf2-m0.avro	avro
snap-1643666583157488463-1-816017ad-b434-49fb-86ac-e302da92bbf2.avro	avro

Apply fine-grained permissions to the data:



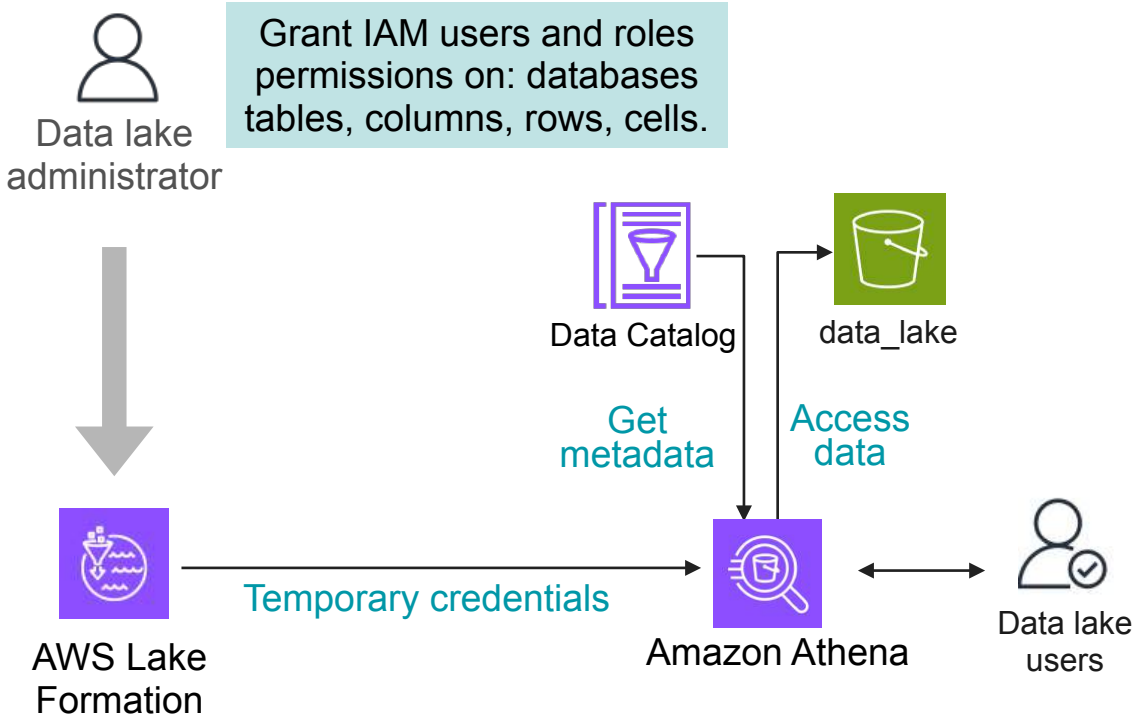
Apply fine-grained permissions to the data:

- Metadata-level (catalog resources)
- Storage access



Apply fine-grained permissions to the data:

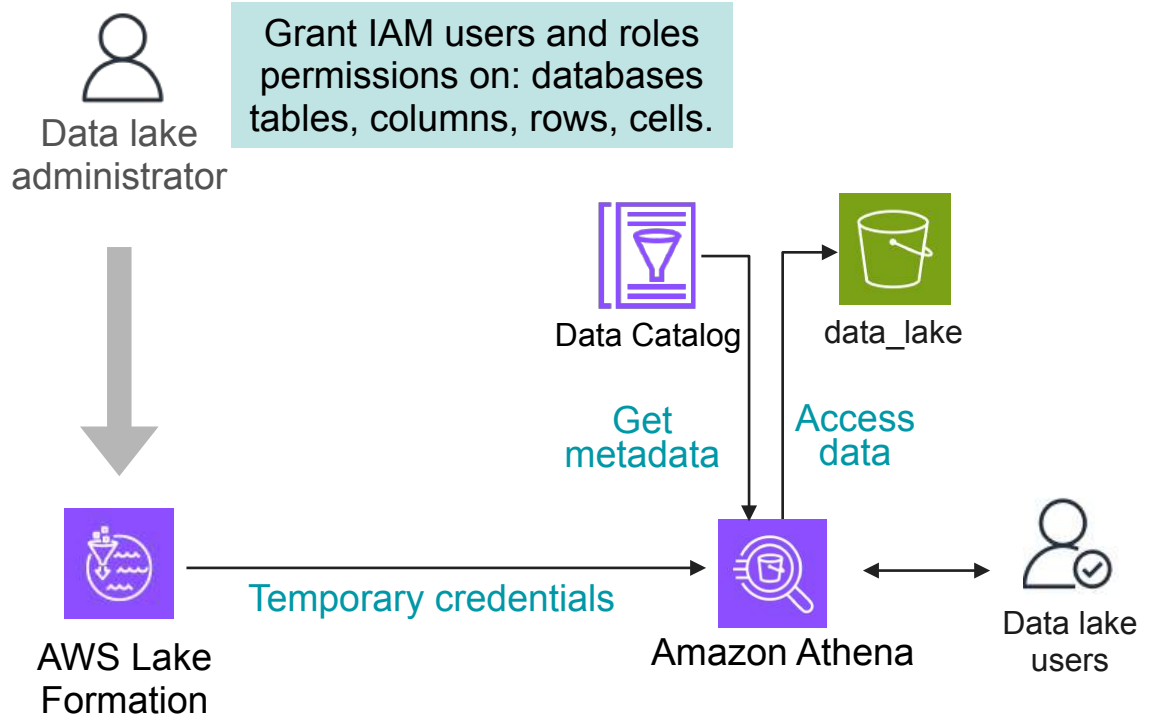
- Metadata-level (catalog resources)
- Storage access



- No need for a detailed IAM policy for data lake users.
- Lake formation permissions are meant to augment regular IAM permissions.
- Data lake users need to be attached to IAM policy to access:
  - AWS Glue service
  - Lake Formation service
  - Amazon Athena
- Use lake formation to grant users fine-grained permissions to access specific resources

Apply fine-grained permissions to the data:

- Metadata-level (catalog resources)
- Storage access



Role assumed by the cloud9 instance

Role assumed by the glue resources

Access all catalog tables and the underlying stored data

Machine Learning team member

Access to the ratings_for_ml table from the presentation zone

DeepLearning.AI

Storage Abstractions

Summary

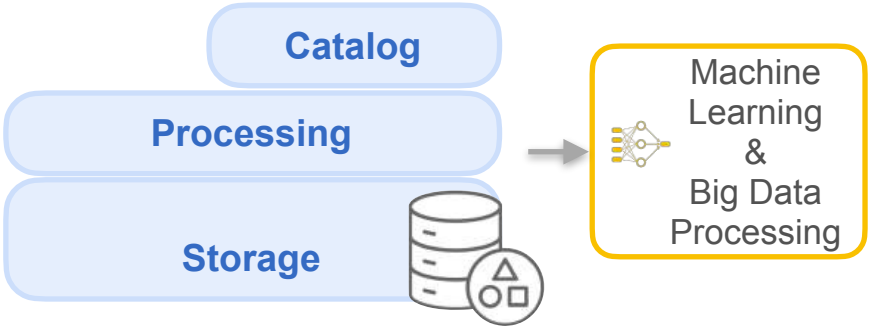
Data Warehouse

- Low-latency query performance
- High storage cost

Data Warehouse

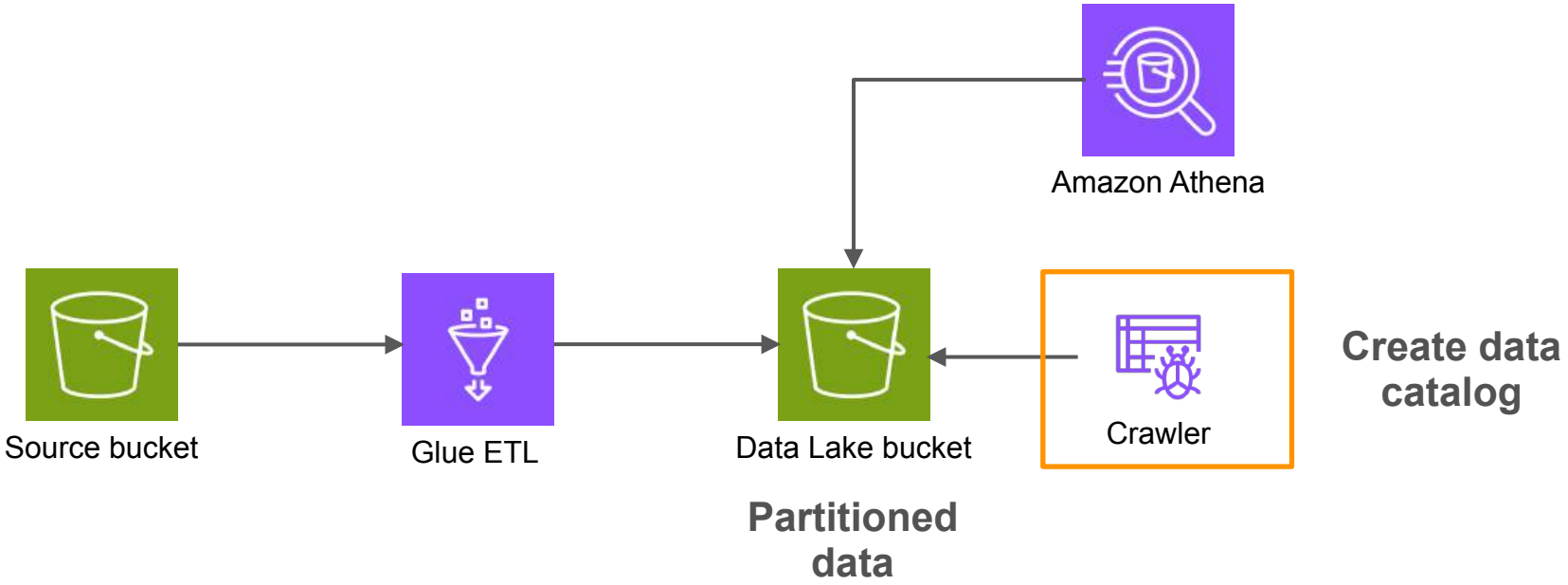
- Low-latency query performance
- High storage cost

Data Lake



- Large amounts of structured and unstructured data
- Supports applications requiring lots of data
- Can turn into data swamp

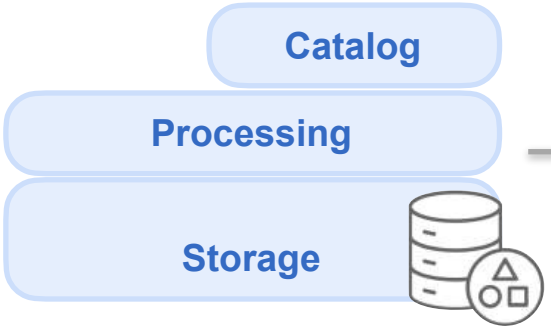
First lab

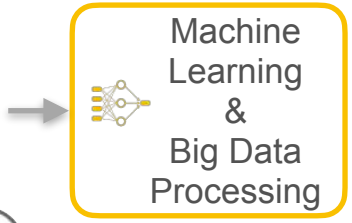

Amazon Athena
Source bucket
Glue ETL
Data Lake bucket
Crawler
Create data catalog
Partitioned data

Data Warehouse

- Low-latency query performance
- High storage cost

Data Lake





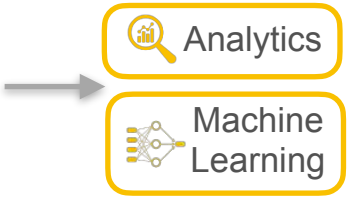
- Large amounts of structured and unstructured data
- Supports applications requiring lots of data
- Can turn into data swamp

Data Lakehouse

Scalable, low-cost, and flexible storage

+

Structured query and data management



Lab Assignment