Copyright Notice

These slides are distributed under the Creative Commons License.

<u>DeepLearning.Al</u> makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite <u>DeepLearning.Al</u> as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode



Data Storage and Queries

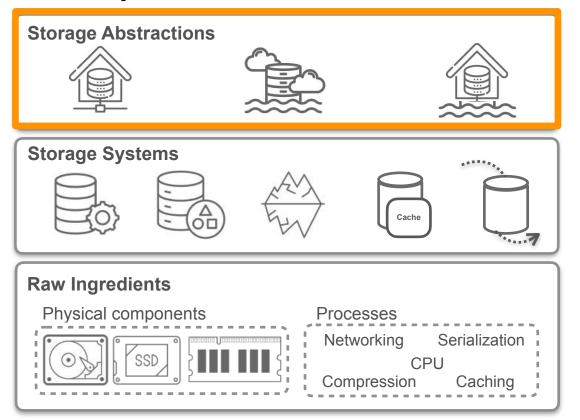
Storage Abstractions



Storage Abstractions

Week 2 Overview

Storage Hierarchy



Storage Hierarchy



Data Warehouse

Cloud data warehouse

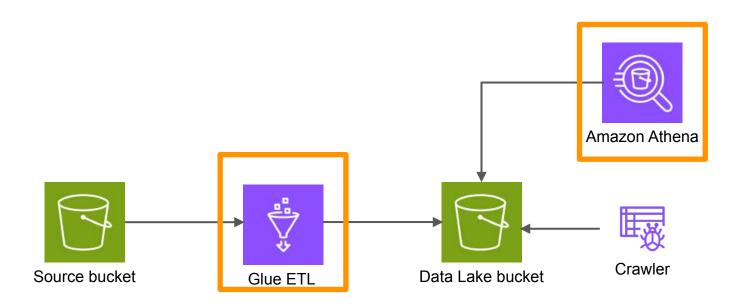
Data Lake

Supports growing storage needs

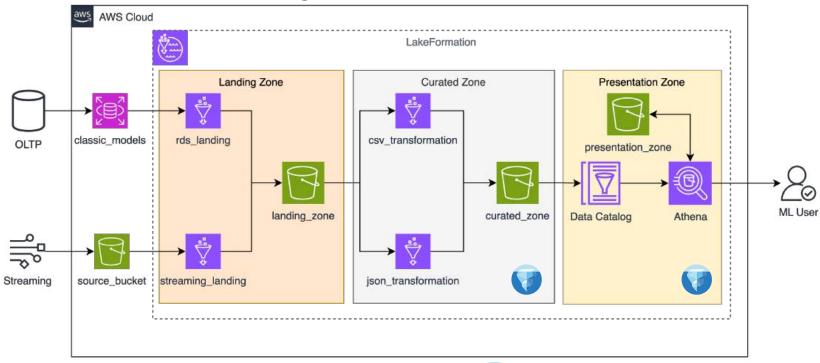
Data Lakehouse

Combines the advantages of data warehouses and data lakes

Week 2 Lab - Simple Data Lake



Week 2 Lab - Building a Data Lakehouse







Storage Abstractions

Data Warehouse -Key Architectural ideas



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.

Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)



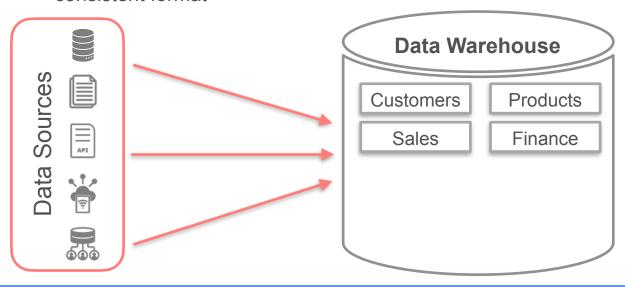
Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format



Subject-Oriented

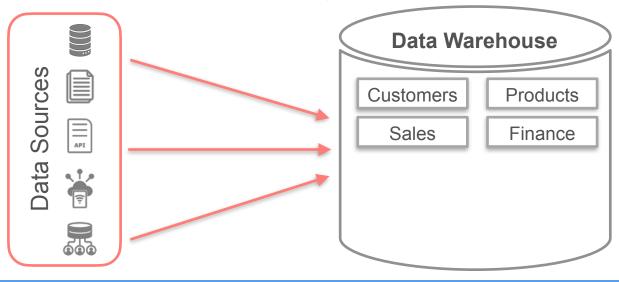
Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile



Subject-Oriented

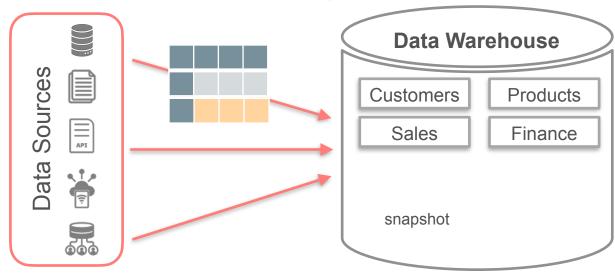
Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile



Subject-Oriented

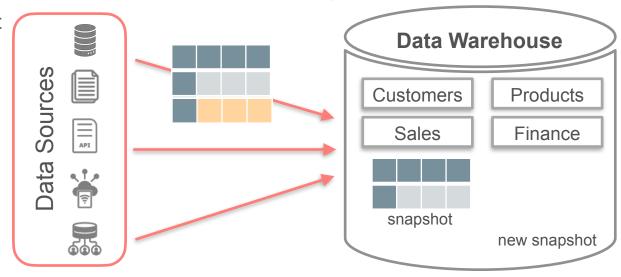
Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile



Subject-Oriented

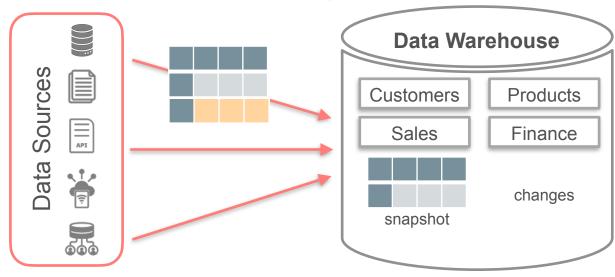
Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

Combines data from different sources into a consistent format

Nonvolatile



Subject-Oriented

Organizes and stores data around key business domains

(Models data to support decision making)

Integrated

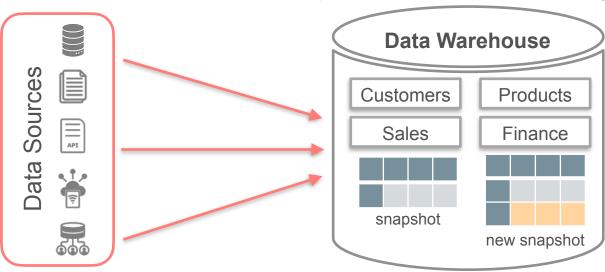
Combines data from different sources into a consistent format

Nonvolatile

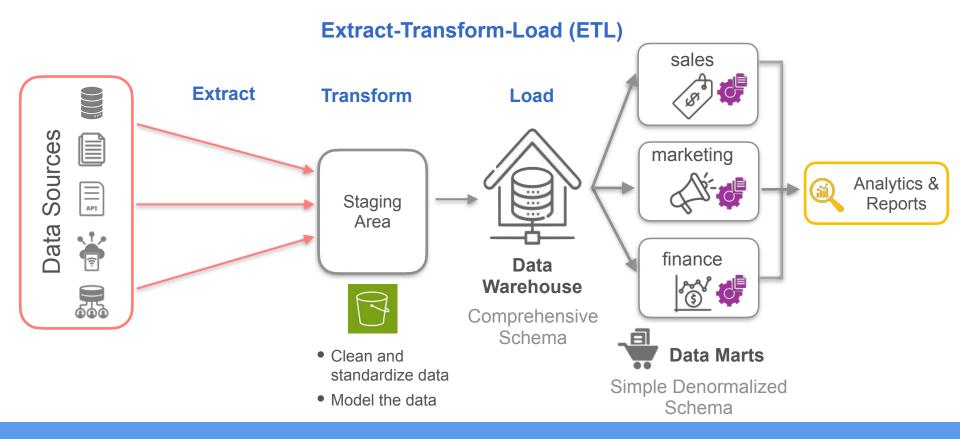
Data is read-only and cannot be deleted or updated

Time-variant

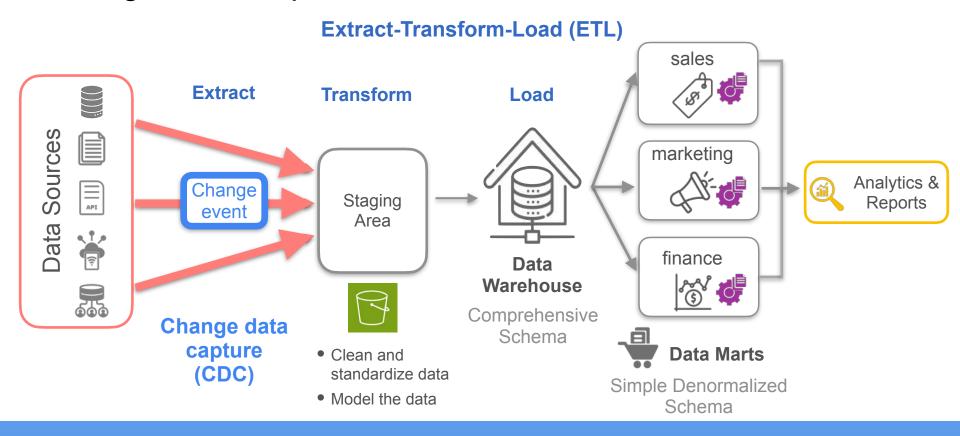
Stores current and historical data (Unlike OLTP systems)



Data Warehouse-Centric Architecture



Change Data Capture



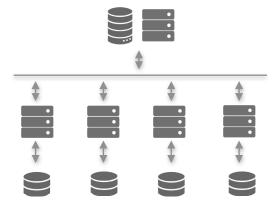
Data Warehouse Implementation

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

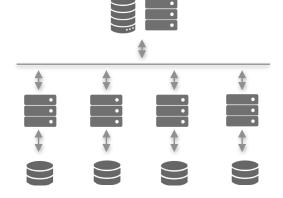
Data Warehouse Implementation

Early Data Warehouses



Big monolithic server

Data Warehouses with Massively Parallel Processing (MPP)

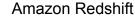


- Scans large amounts of data in parallel
- Complex configurations and requires effort to maintain

Modern Cloud Data Warehouses









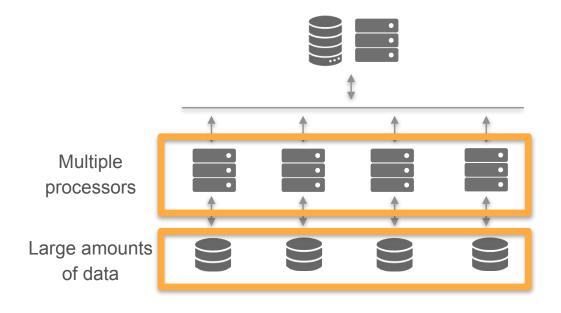
- Separates compute from storage
- Expands the capability of MPP systems



Storage Abstractions

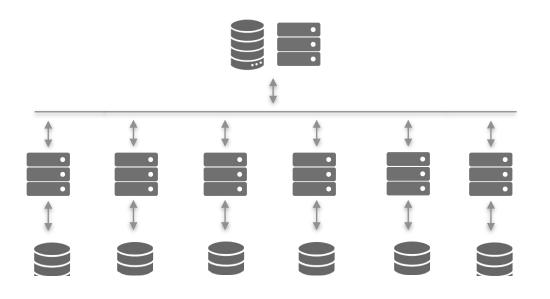
Modern Cloud Data Warehouses

Massively Parallel Processing



Massively Parallel Processing (MPP)

Massively Parallel Processing - Cloud Data Warehouses

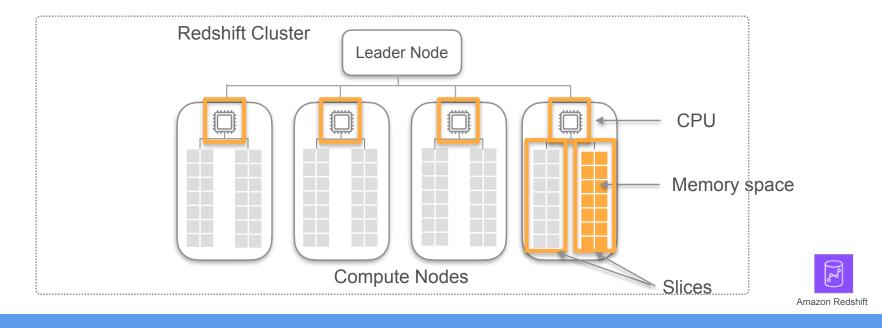


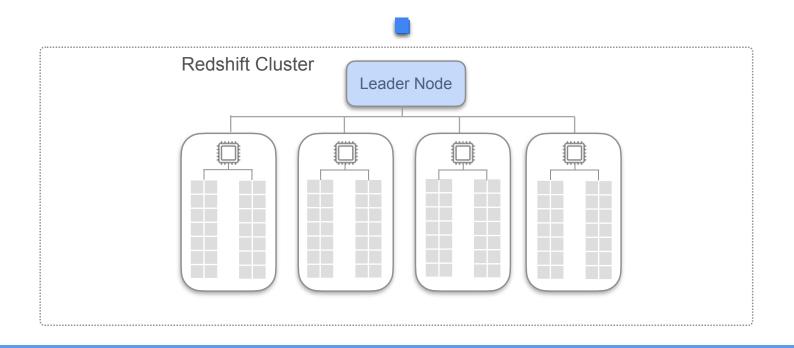




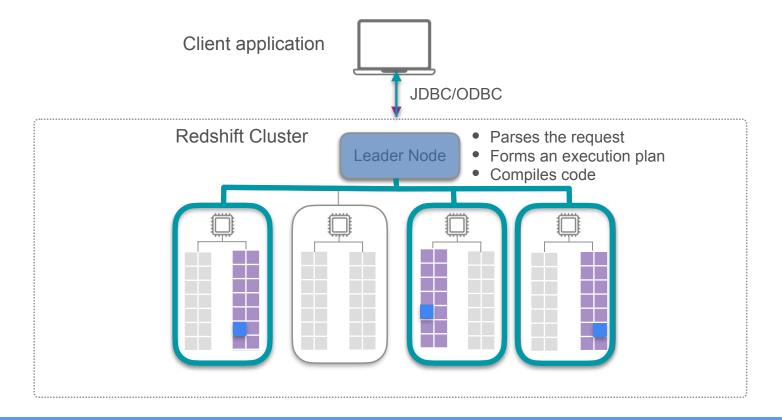


Massively Parallel Processing (MPP)

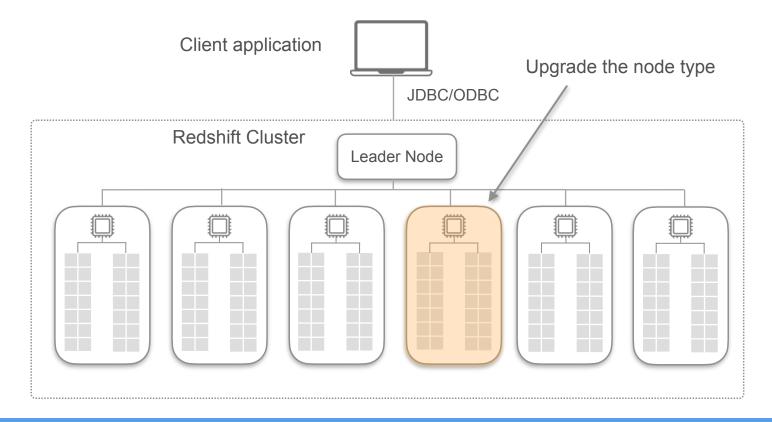








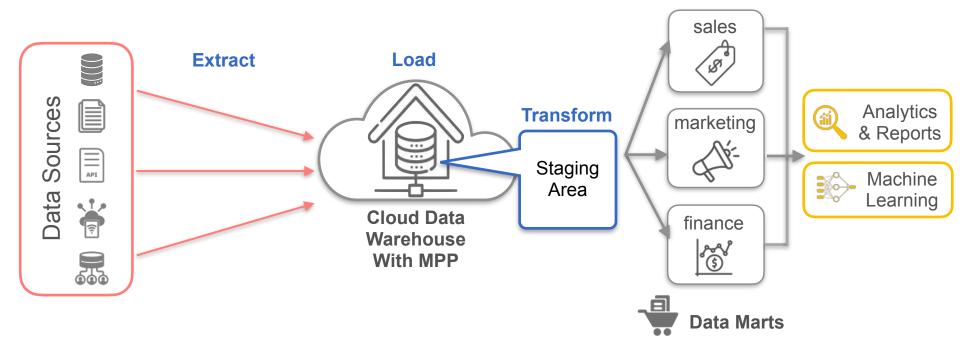






Data Warehouse-Centric Architecture

Extract-Load-Transform (ELT)



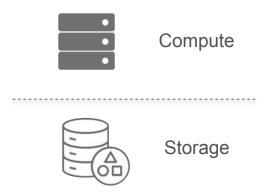
Cloud Data Warehouse

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



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



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



Processing Tools













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

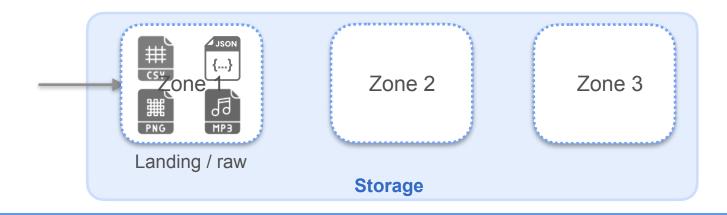


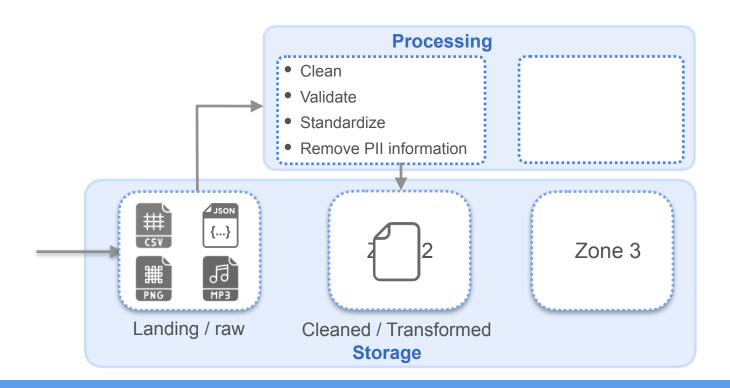


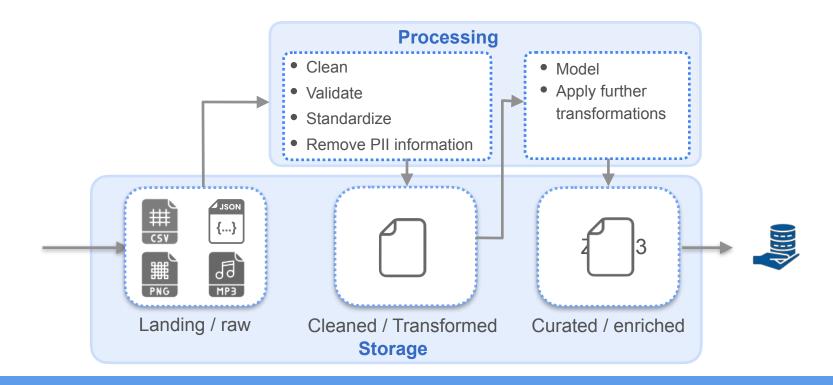
Storage Abstractions

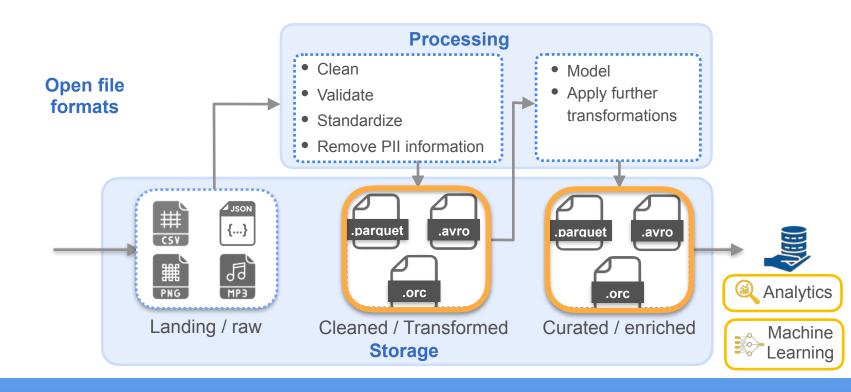
Next-Generation Data Lakes

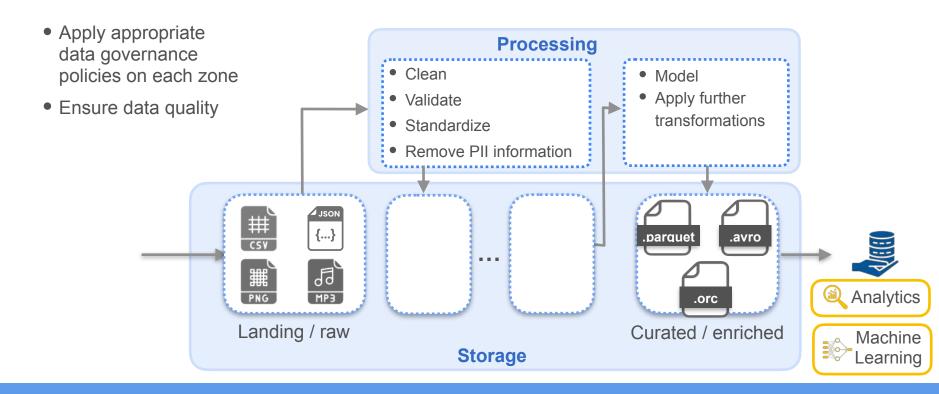






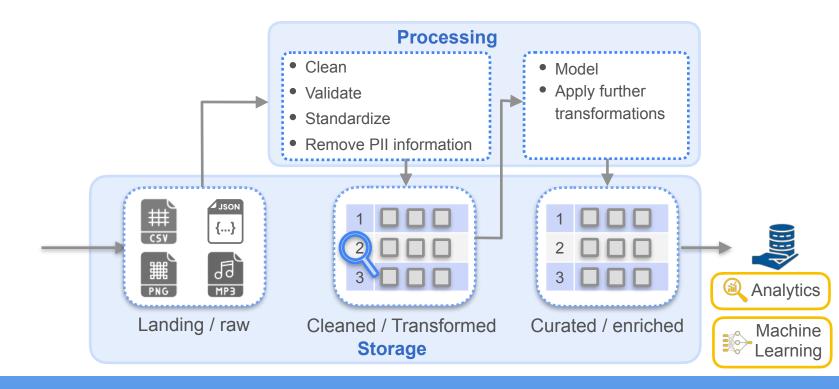






Data Partitioning

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

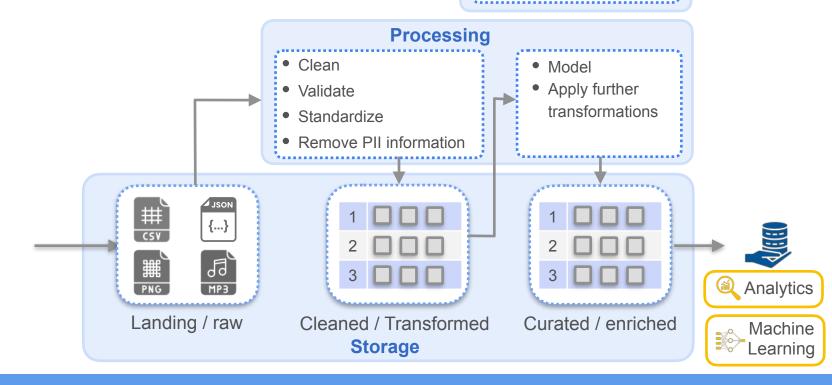


Data Catalog

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

Catalog

- Metadata
- Schema





Data Lake

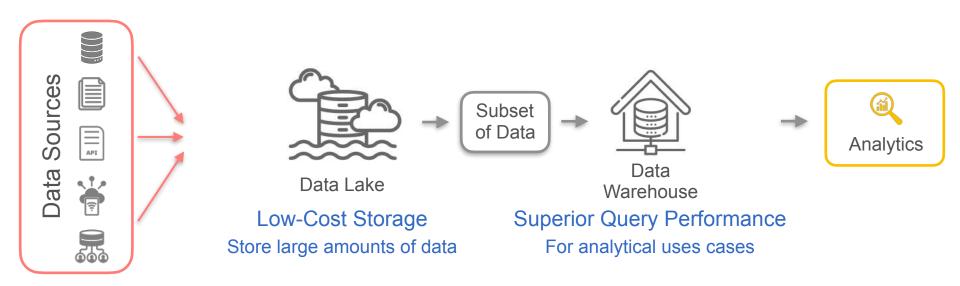
Low-Cost Storage
Store large amounts of data

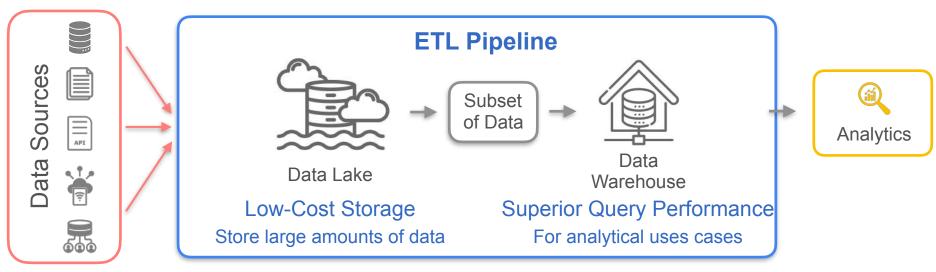
and



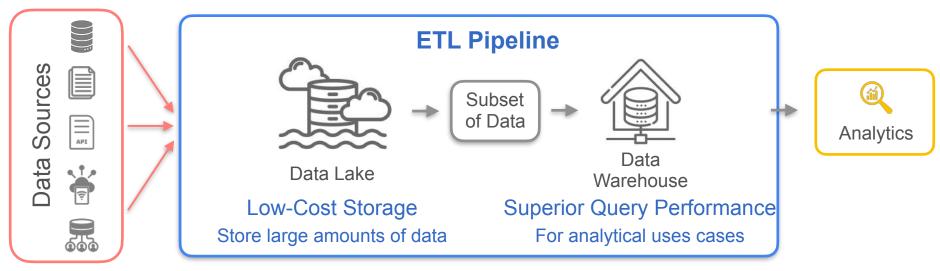
Superior Query Performance

For analytical uses cases





Expensive Solution



Expensive Solution

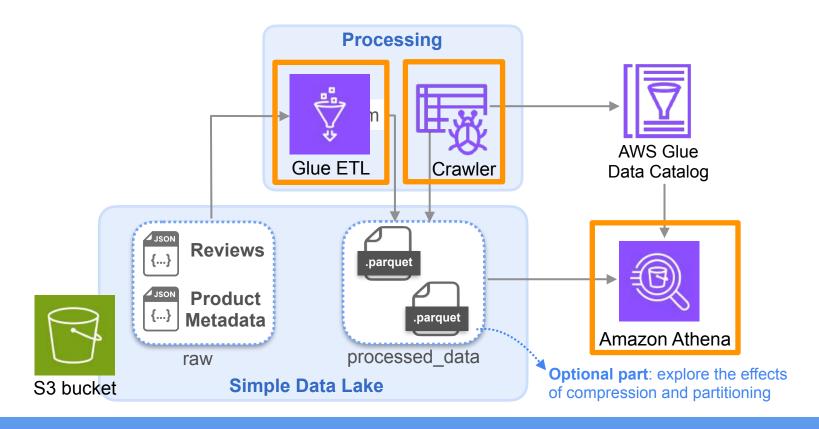
- Can introduce bugs/failures
- Can cause issues with data quality, duplication, consistency



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

```
"reviewerID": "A2SUAM1J3GNN3B"
              asin": "0000013714",
  JSON
              reviewerName": "J. McDonald",
              helpful" [2, 3].
  \{...\}
              reviewText": "Great purchase!',
              overall 5.0,
Reviews
              summary": "Heavenly Highway Hymns"
              unixReviewTime": 1252800000,
              reviewTime": "09 13, 2009"
                    Import the review data into a tabular data frame
                                      reviewTextToverallIsummarv
                                                                unixReviewTime | reviewTime
 reviewerID
            asin
                 reviewerName l
                               helpful
                    Process the data
```

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

Processing the JSON Files

Complete two functions to process the raw JSON files



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

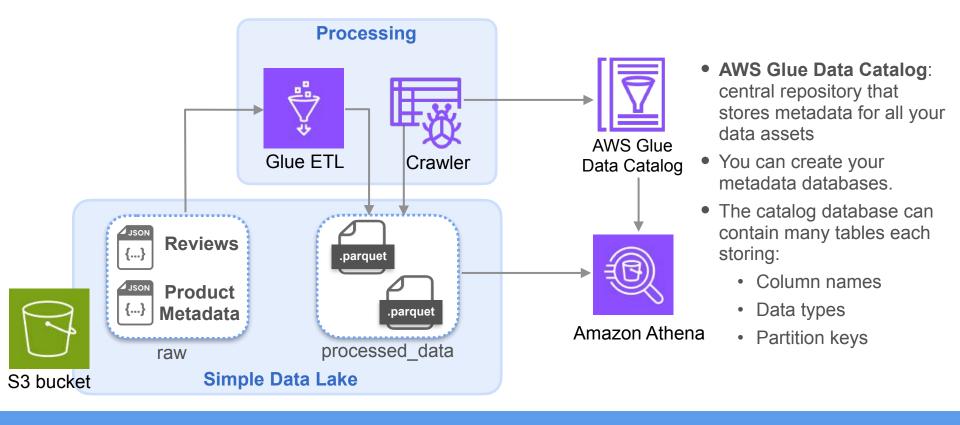




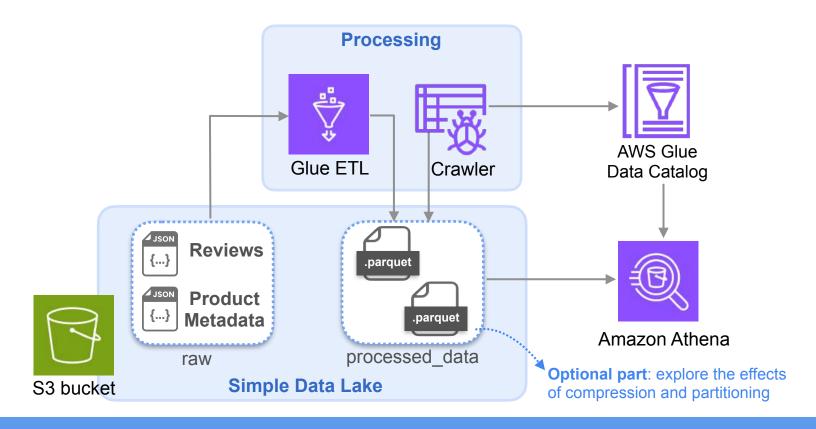
Lab Walkthrough

Simple Data Lake with AWS Glue (Part 2)

Glue Data Catalog



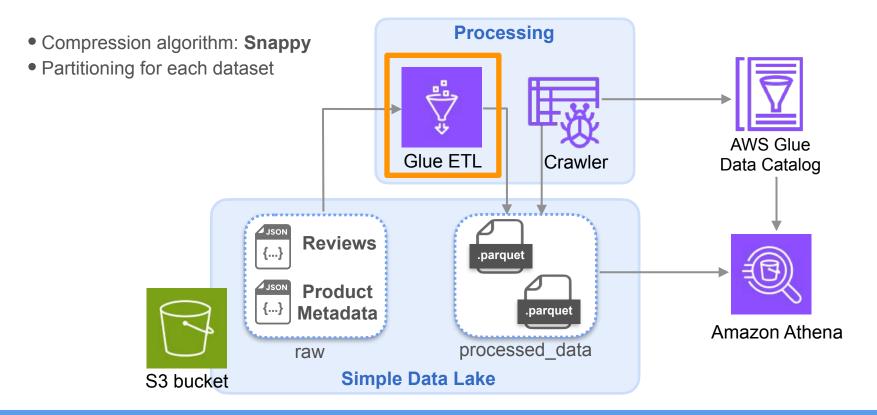
Lab Overview

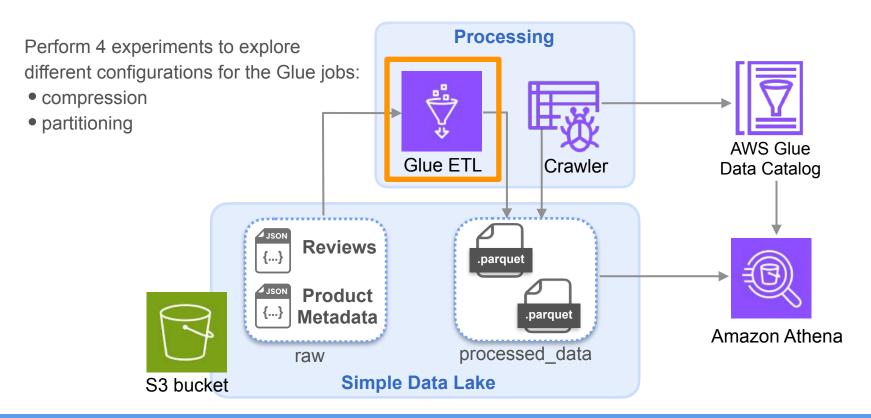




Lab Walkthrough

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





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

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

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

faster

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	Motodoto	No partitioning	Snappy	4	51 MiB
2	2 Metadata	No partitioning	Gzip	4	32.7 MiB
2	Dovious	No partitioning	Snappy	4	556.6 MiB
3	Reviews	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.

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

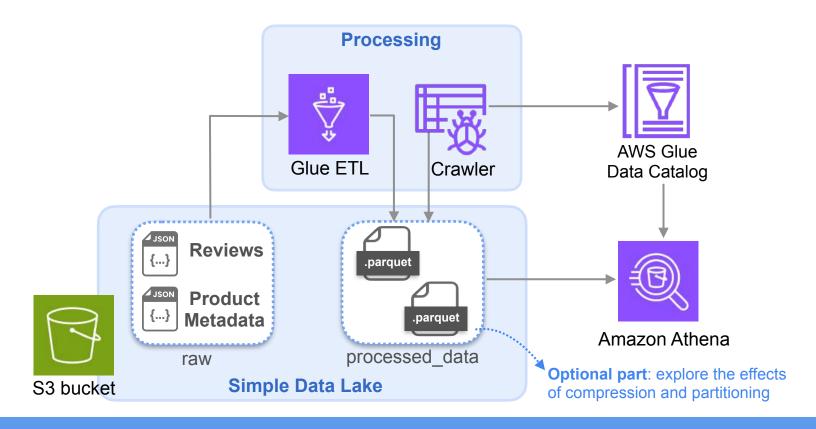
What you'll run?

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

What you'll compare?

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

Lab Overview





Storage Abstractions

Data Lakehouse

Data Lakehouse



Data Lake

- Flexibility
- Low-Cost Storage



Data Warehouse

- Superior query performance
- Robust data management

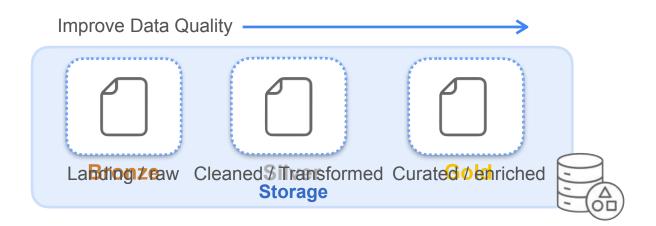


Data Lakehouse

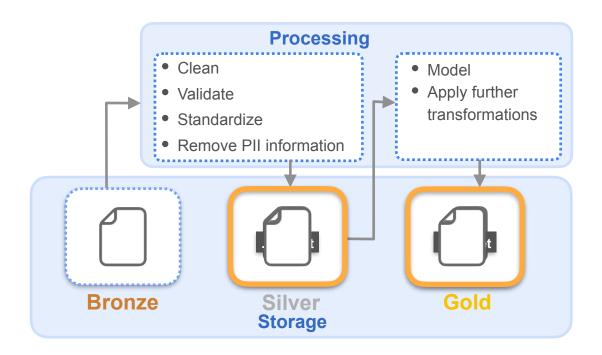
Data Lake Features

a databricks

Medallion Architecture



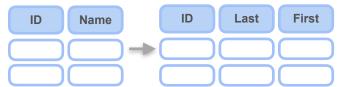
Data Lake Features



Data Lakehouse Catalog Metadata Schema **Processing** Clean Model Apply further Validate transformations Standardize Remove PII information .parquet .parquet **Bronze** Silver Gold **Storage**

Data Management from Data Warehouses

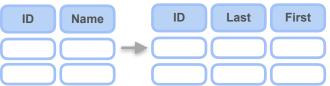
Schema Enforcement



Data Lakehouse Catalog Metadata Schema **Processing** Clean Model Apply further Validate transformations Standardize Remove PII information .parquet .parquet **Bronze** Silver Gold **Storage**

Data Management from Data Warehouses

Schema Enforcement



ACID

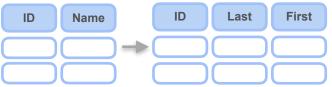
- Atomic, Consistent, Isolated, Durable
- Concurrent read, insert, update, delete

Data Lakehouse Catalog Metadata Schema **Processing** Clean Model Apply further Validate transformations Standardize Remove PII information .parquet .parquet Gold **Bronze** Silver

Storage

Data Management from Data Warehouses

Schema Enforcement



ACID

- Atomic, Consistent, Isolated, Durable
- Concurrent read, insert, update, delete

Data Governance & Security

- Robust access controls, data auditing capabilities, data lineage
- Incremental updates & deletions

Connector

APIs

 Rollback to or access any version of your historical data



Storage Abstractions

Date Lakehouse Implementations

Open Table Formats

Specialized storage formats that add transactional features to your data lakehouse

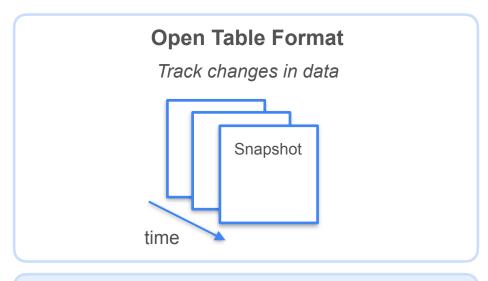
- Allows you to update and delete records
- Supports ACID principles







Hadoop
Update
Delete
Incremental

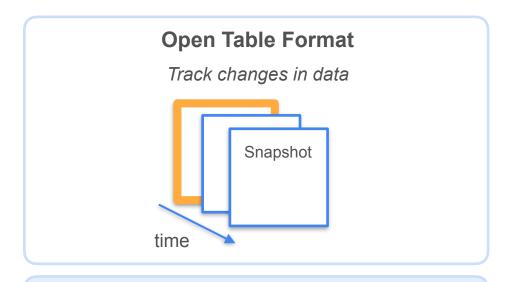


Snapshot:

Reflects the state of the data at a given time

Data

Insert, Update, Delete



Data

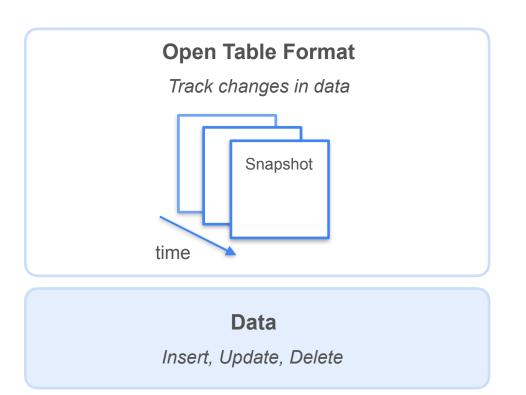
Insert, Update, Delete

Snapshot:

Reflects the state of the data at a given time

Time Travel:

Query any previous version of a table



Snapshot:

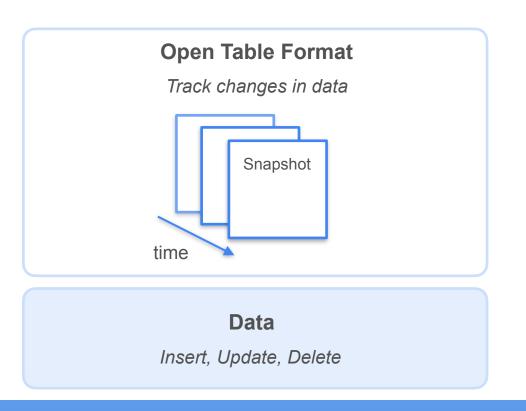
Reflects the state of the data at a given time

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



Snapshot:

Reflects the state of the data at a given time

Time Travel:

Query any previous version of a table

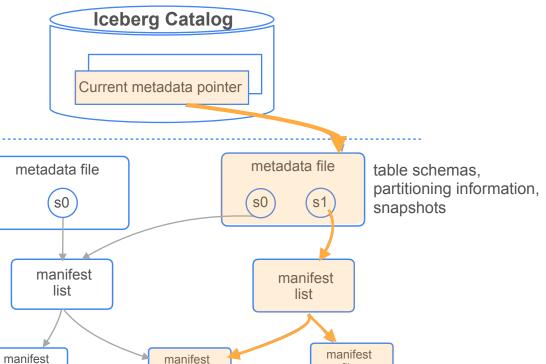
Schema & Partition Evolution:

Ability to query the data even if you make changes to the schema or partitioning

Open Source

Different query engines can access the data





file

data files

file

data files

Storage layer

file

data files

Metadata layer

(Parquet)

Storage Options

Production Database	Data Warehouse
Process small amounts of structured data	 Bring together large volumes of structured / semistructured 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 / semistructured 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 / semistructured data Query current and historical data 	 Process large volumes of structured, semistructured and unstructured data Save on storage cost 	Data management & discoverability featuresLow latency queries
Use case: reporting, analytics	Use case: reporting, analytics	Use case: machine learning	Use case: machine learning, analytics, reporting





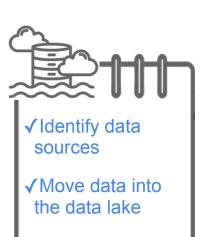
Storage Abstractions

Lakehouse Architecture on AWS

AWS Lake Formation Groups Users Setting up access controls {...} **AWS Lake Formation** Cataloging data Defining storage Managing permissions

AWS Lake Formation

Data Sources



- √ Catalog data
- ✓ Manage permissions

Setting up a Data Lake



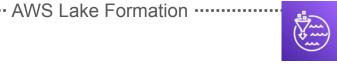










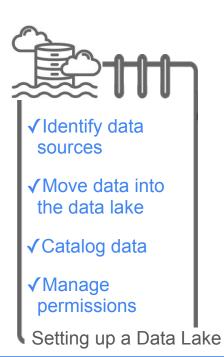


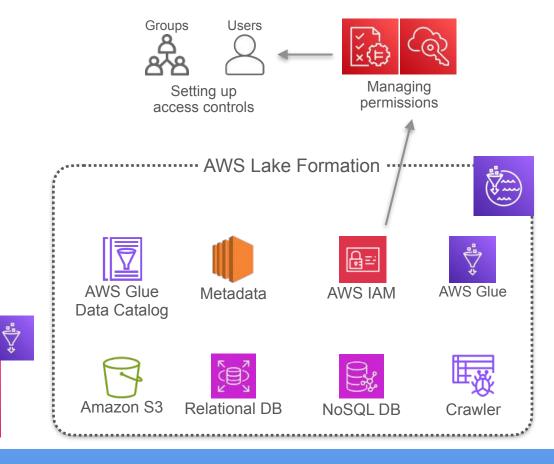


AWS Lake Formation

Glue

Console

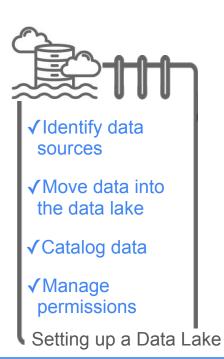


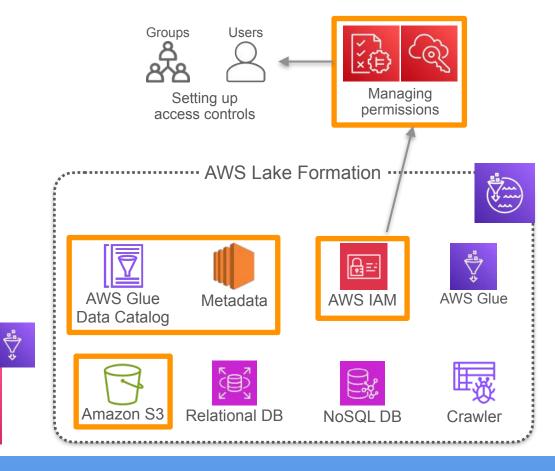


AWS Lake Formation

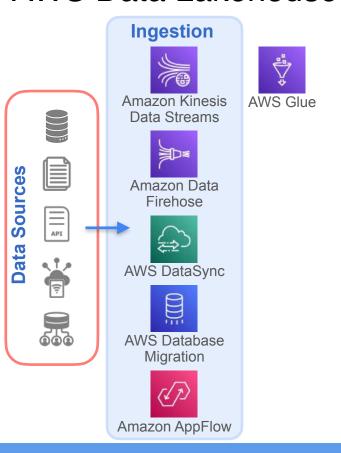
Glue

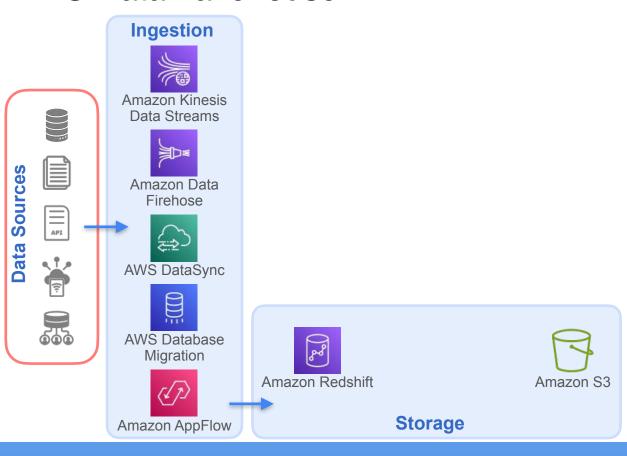
Console

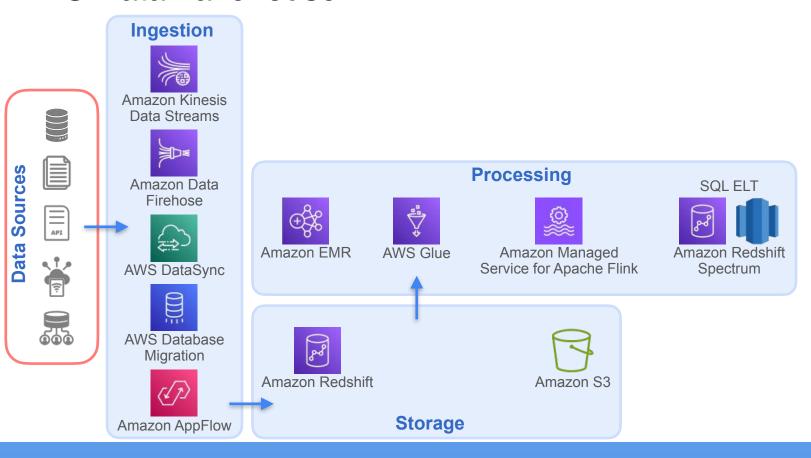


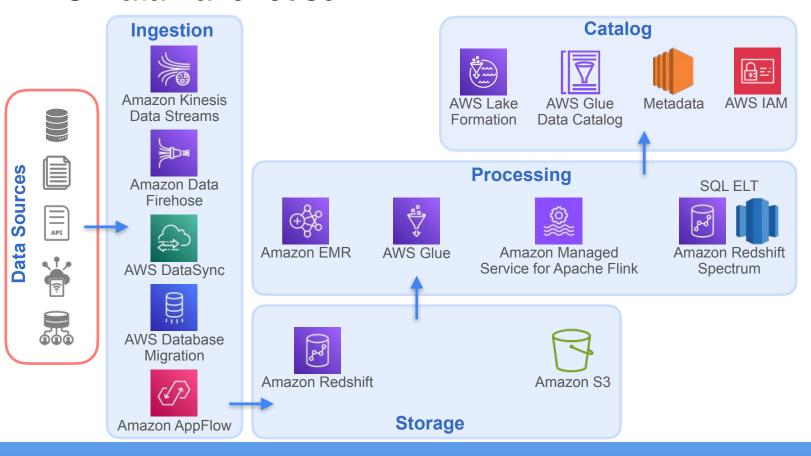


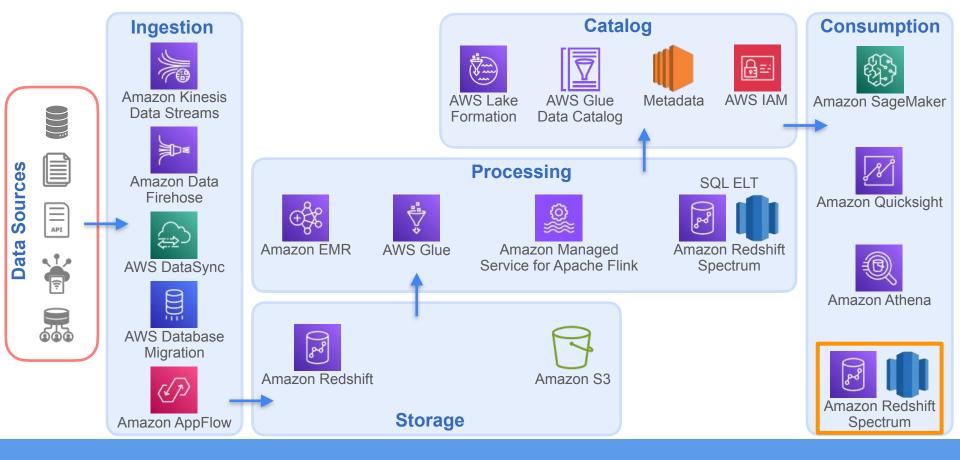








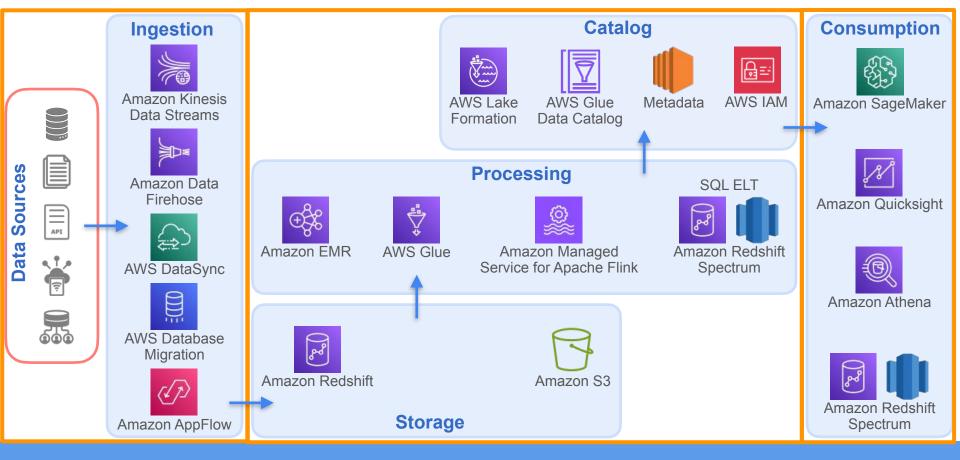


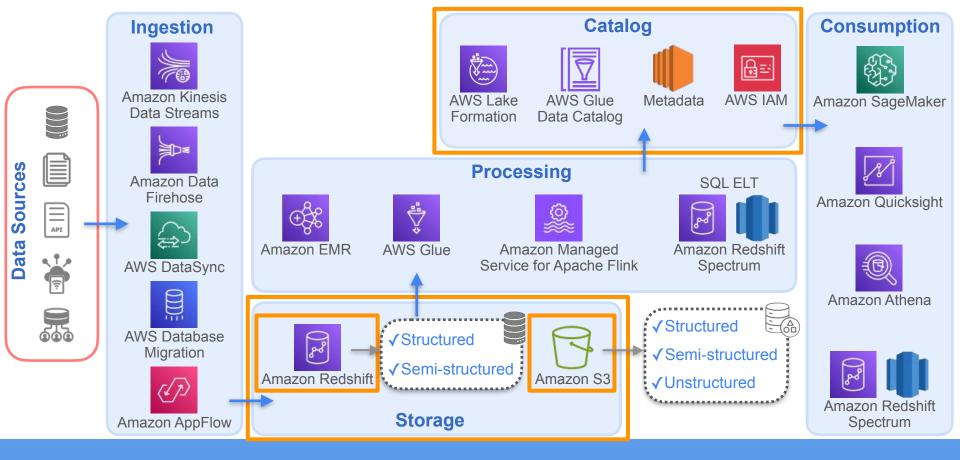


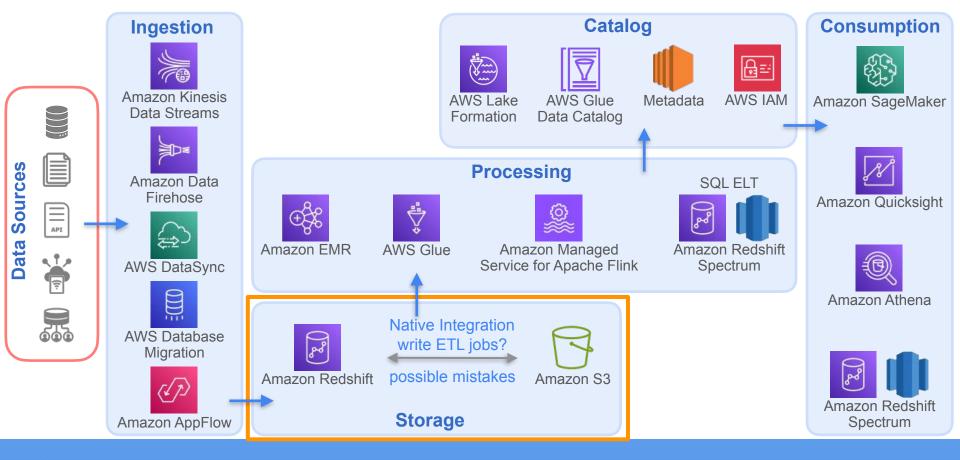


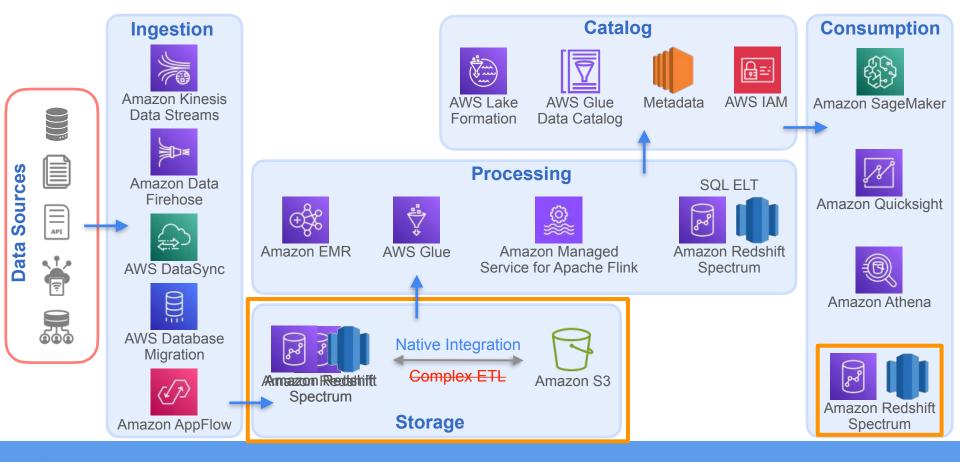
Storage Abstractions

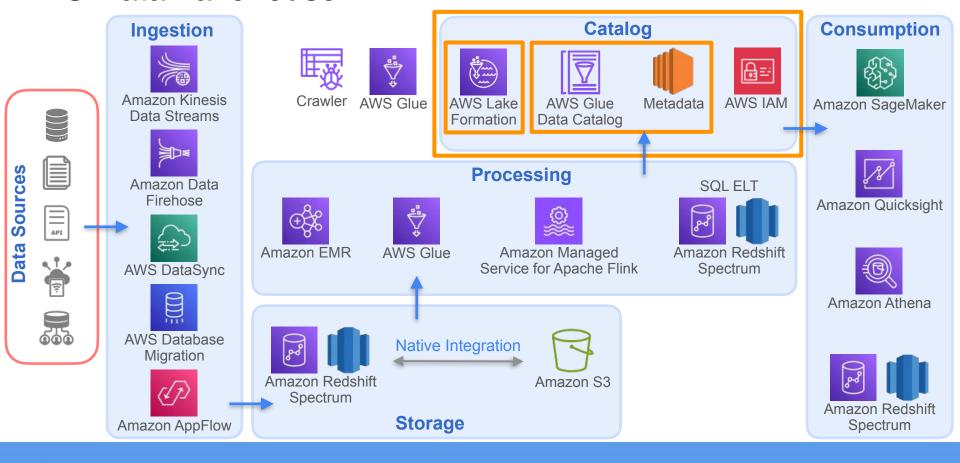
Implementing a Lakehouse on AWS

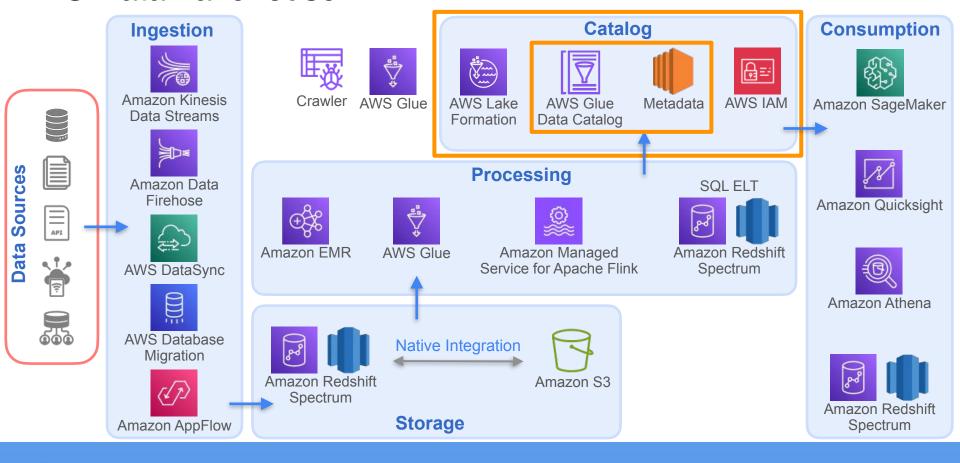


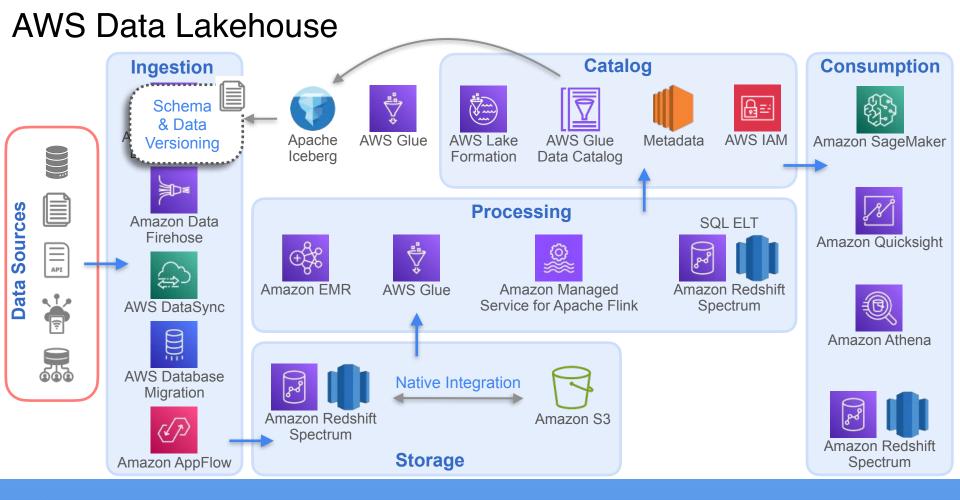


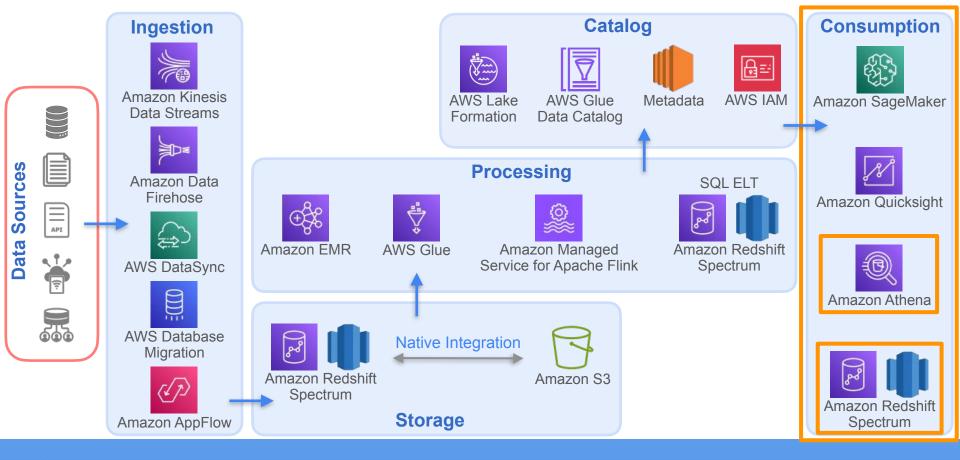


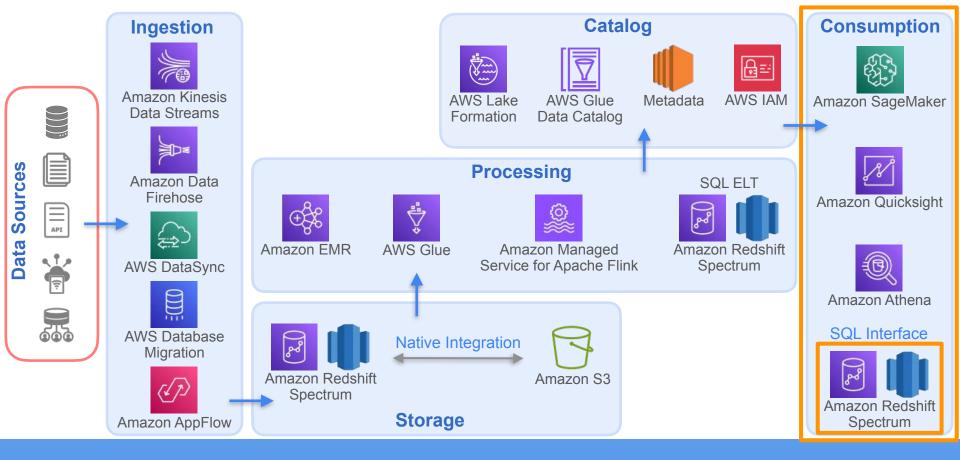


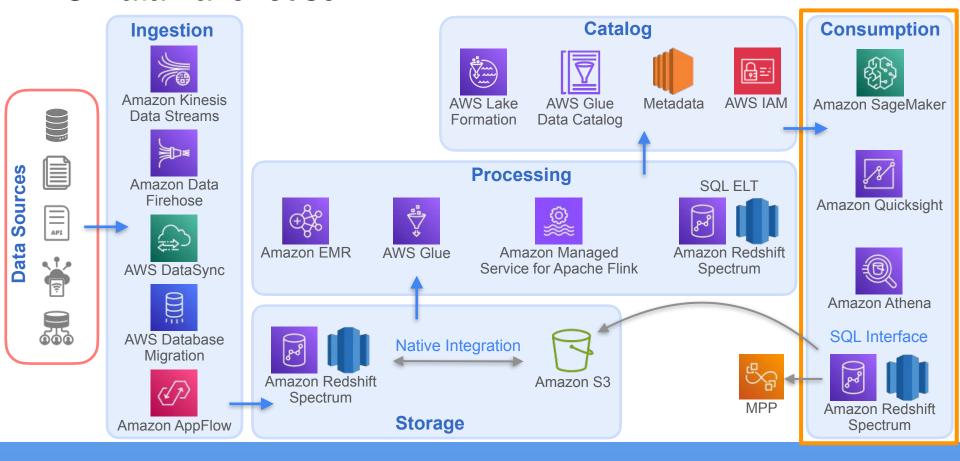


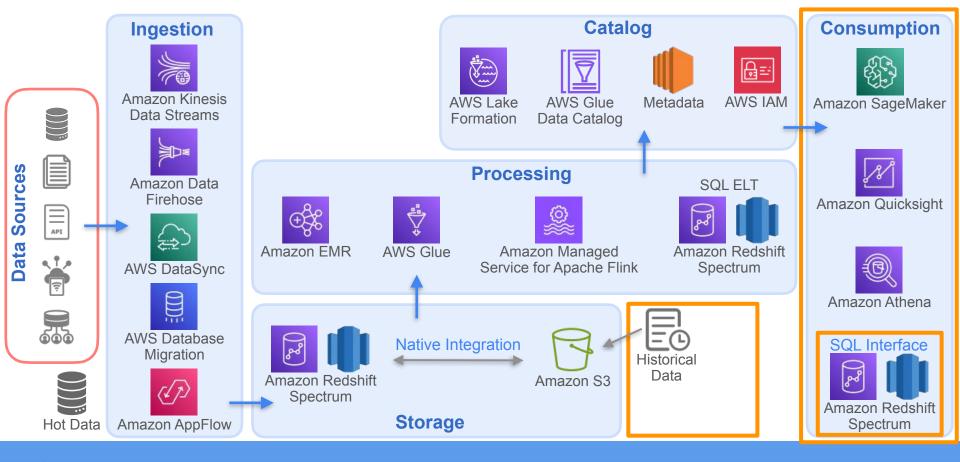


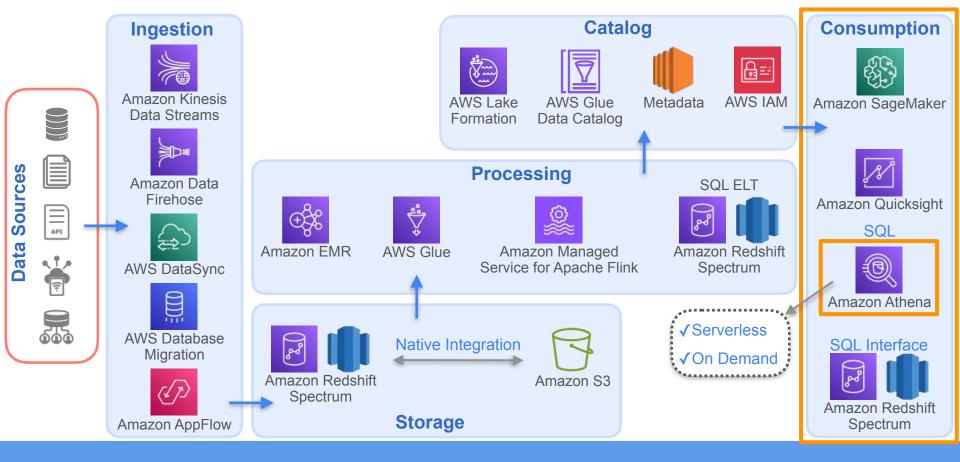


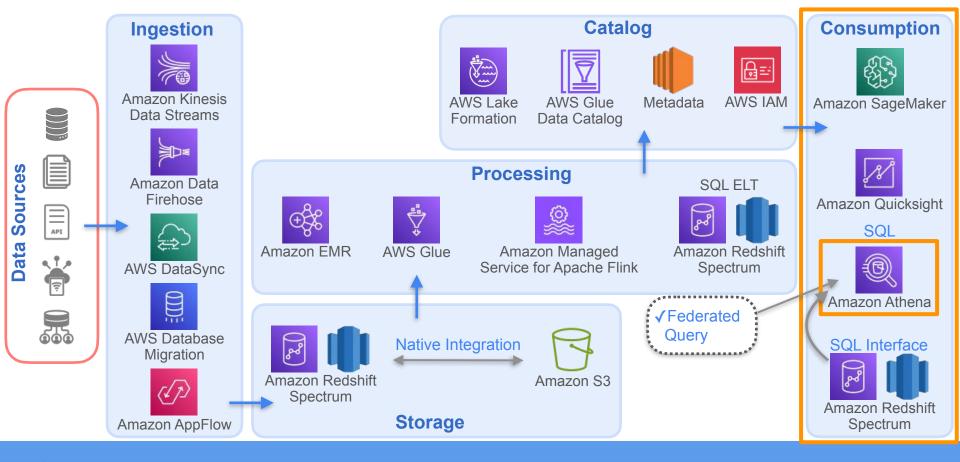


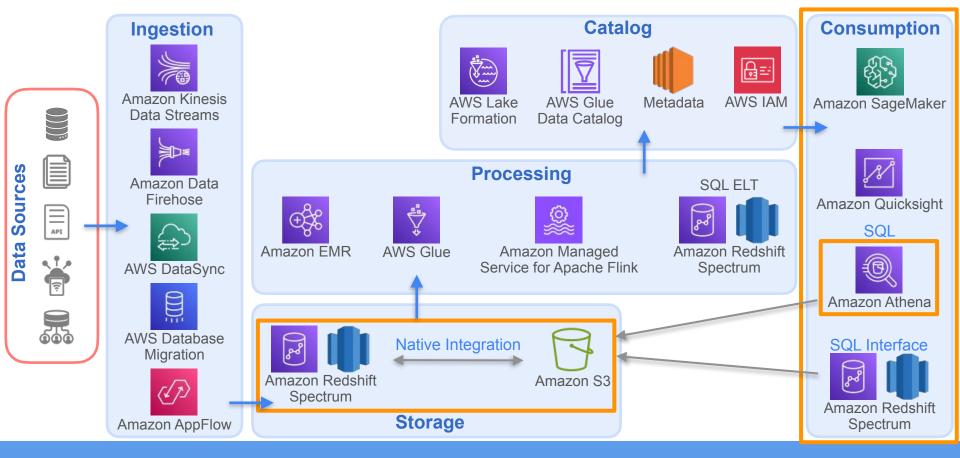








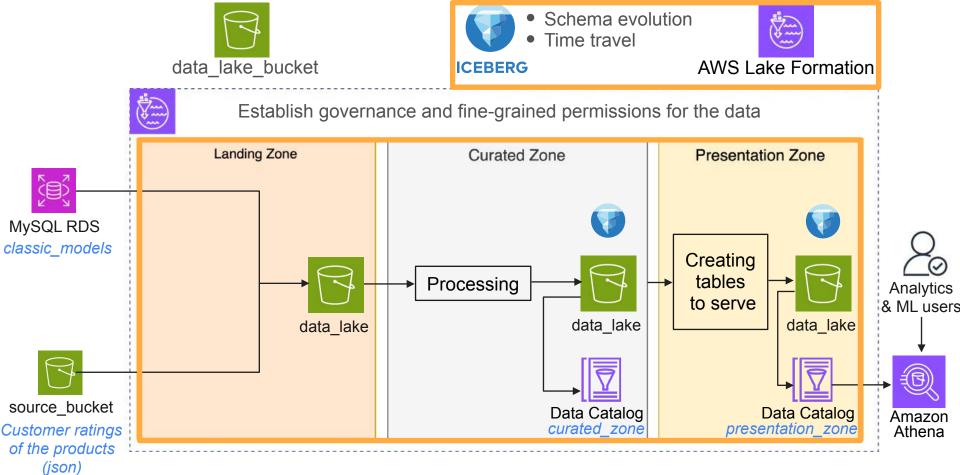






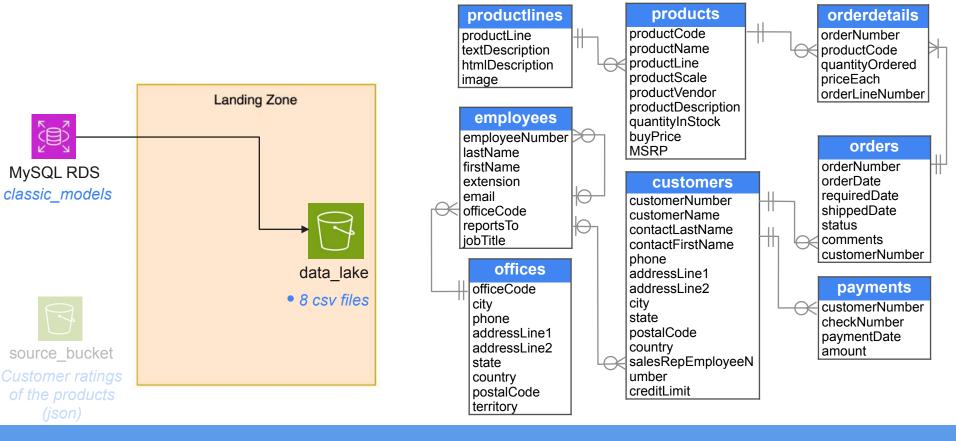
Lab Walkthrough

Building a Data Lakehouse with AWS LakeFormation (Part 1)

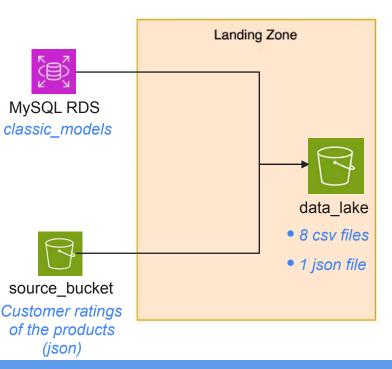


• Each csv file:

s3://{data lake bucket}/landing zone/rds/{table name}



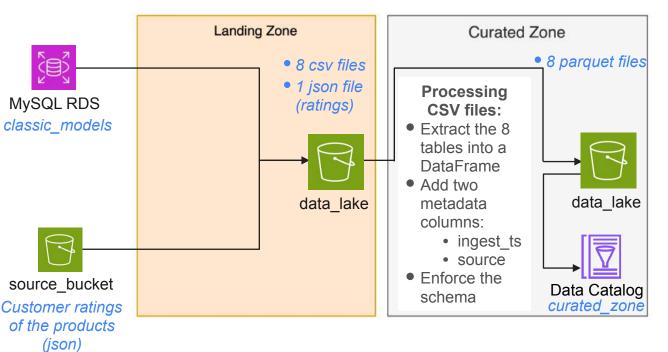
- Each csv file:s3://{data_lake_bucket}/landing_zone/rds/{table_name}
- JSON file: s3://{data_lake_bucket}/landing_zone/json/ratings



```
"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 product product ingest_ts
Number
           Code
                  Rating
```

Each parquet file:

s3://{data_lake_bucket}/curated_zone/{table_name}



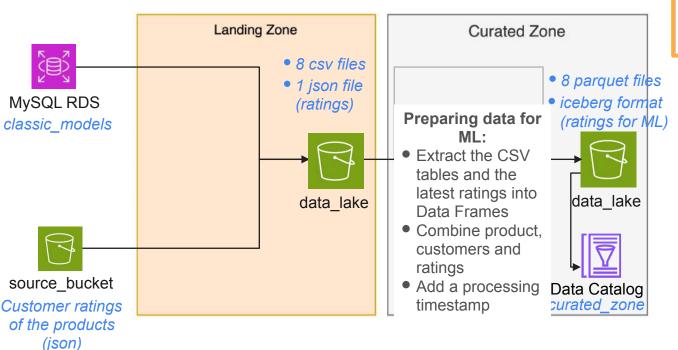
You are provided with a schema for the processed tables.

Each parquet file:

s3://{data_lake_bucket}/curated_zone/{table_name}

• Iceberg data:

s3://{data_lake_bucket}/curated_zone/ratings_for_ML/iceberg



products

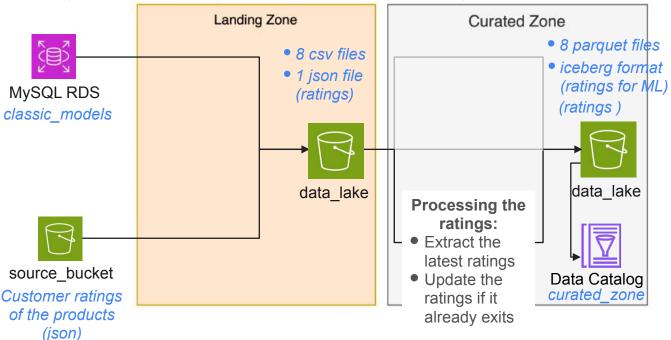
productCode
productName
productLine
productScale
productVendor
productDescription
quantityInStock
buyPrice
MSRP

customers

customerNumber
customerName
contactLastName
contactFirstName
phone
addressLine1
addressLine2
city
state
postalCode
country
salesRepEmployeeN
umber
creditLimit

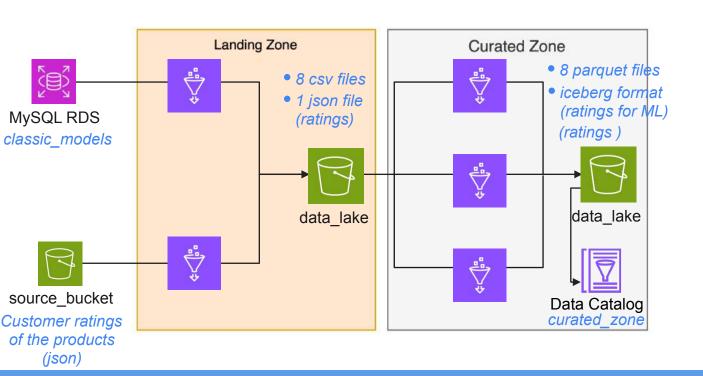
```
"customerNumber": 103,
    "productCode": "S18_1589",
    "productRating": 3
},
{
    "customerNumber": 103,
    "productCode": "S18_2870",
    "productRating": 5
},
{
    "customerNumber": 103,
    "productCode": "S18_3685",
    "productTating": 4
},
{
    "customerNumber": 103,
    "productRating":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

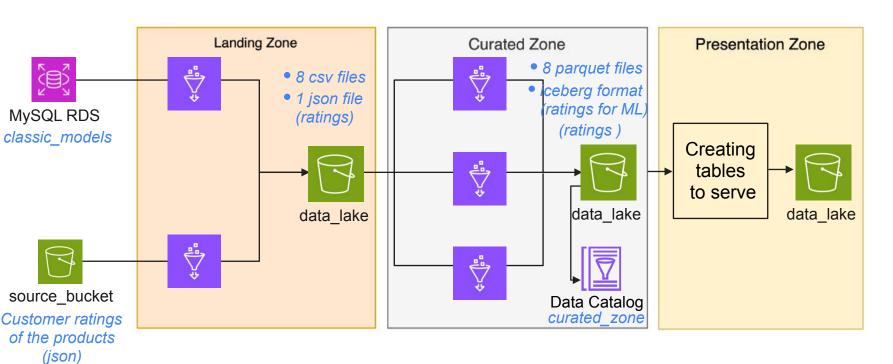


For analytics end users:

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

For ML end users:

Ratings for ML table



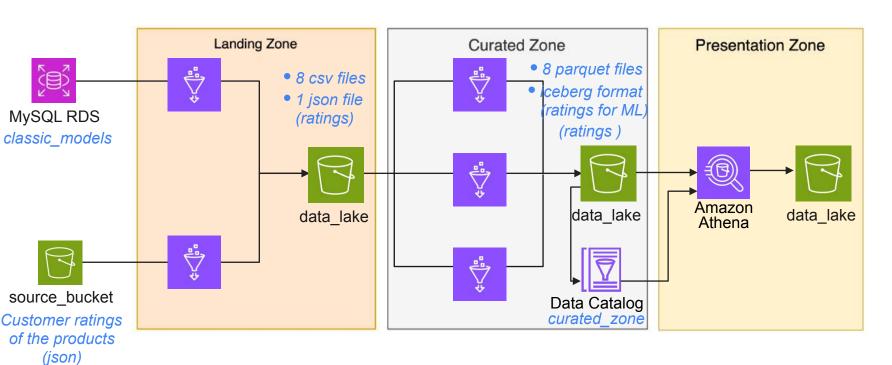


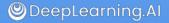
For analytics end users:

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

For ML end users:

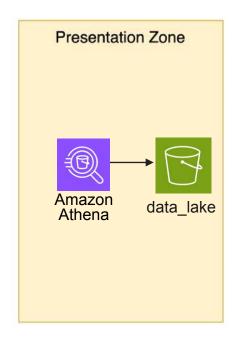
Ratings for ML table





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'])
```

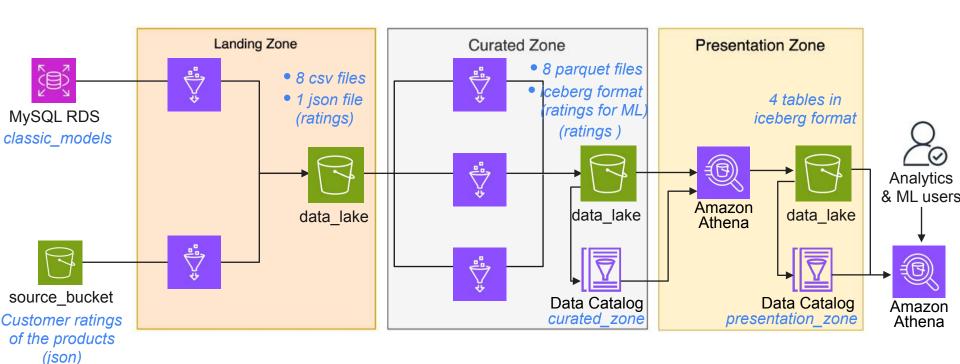


For analytics end users:

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

For ML end users:

Ratings for ML table

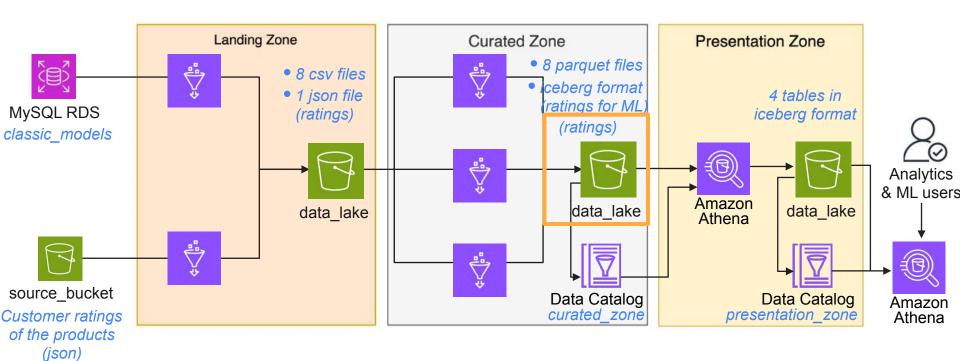




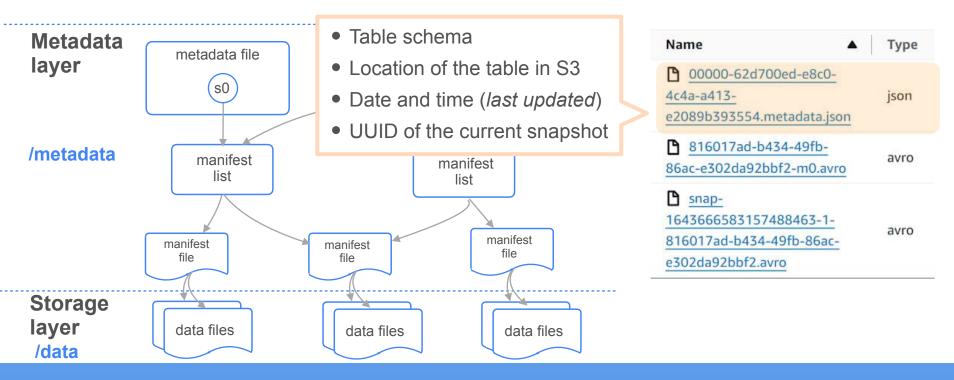
Lab Walkthrough

Building a Data Lakehouse with AWS LakeFormation (Part 2)

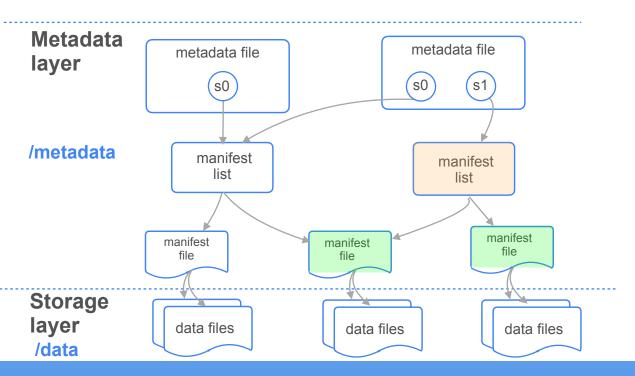


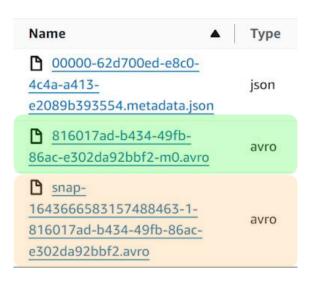




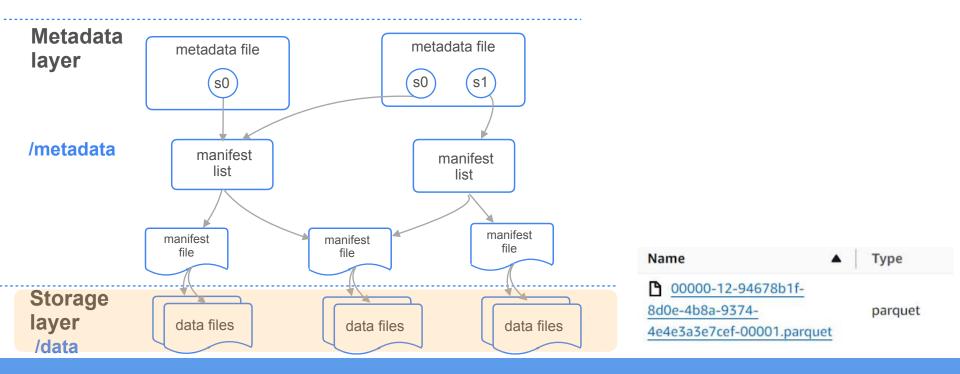


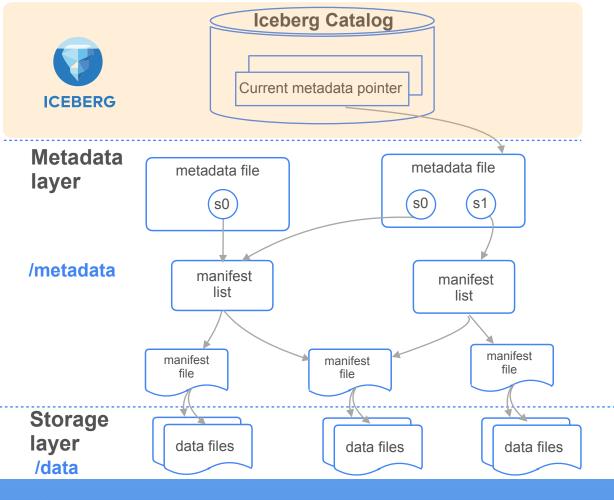






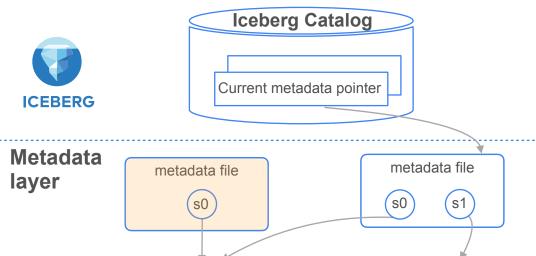








- Contains a catalog table for each iceberg file in the curated and presentation zone
- Catalog tables are organized:
 - curated_zone database
 - presentation_zone database



manifest

file

data files

manifest

list

manifest

file

data files

manifest

list

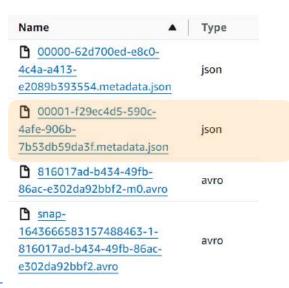
manifest

file

data files

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



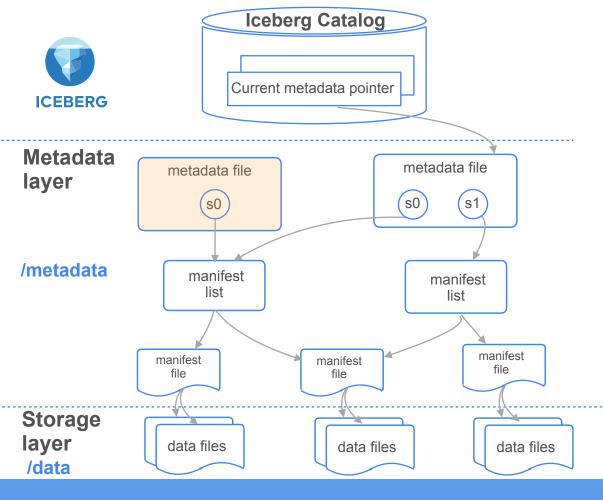
The metadata file will change without the need to completely re-write the data

/metadata

Storage

layer

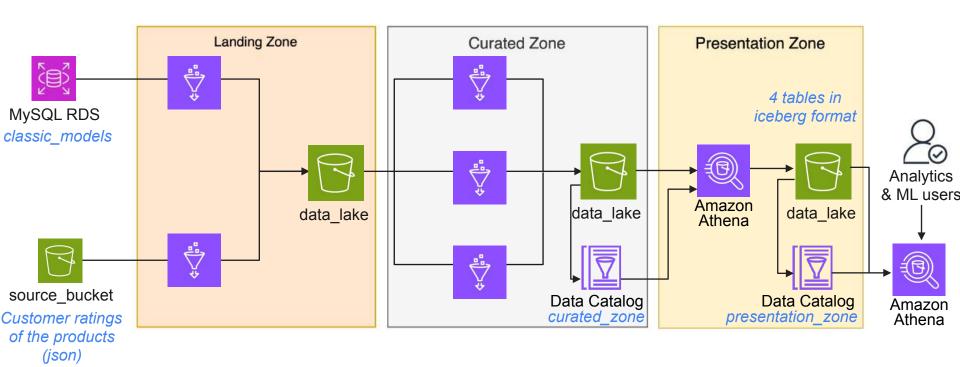
/data



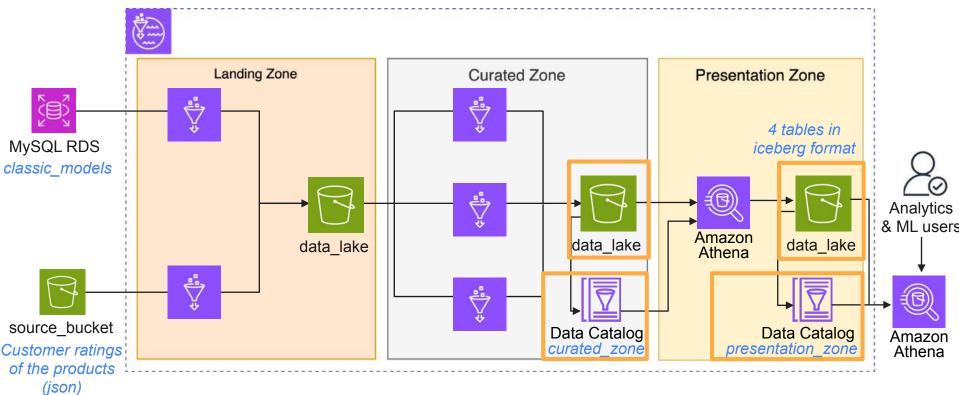
Time Travel

Query the new and old versions of the ratings table.

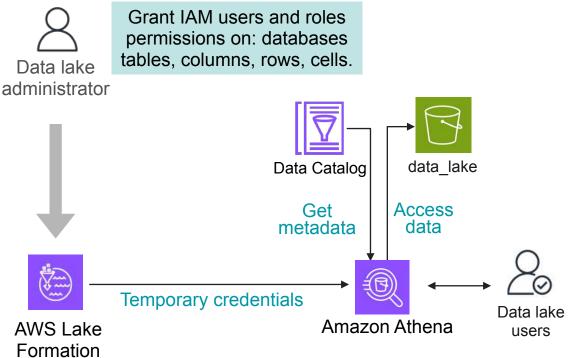




- Metadata-level (catalog resources)
- Storage access

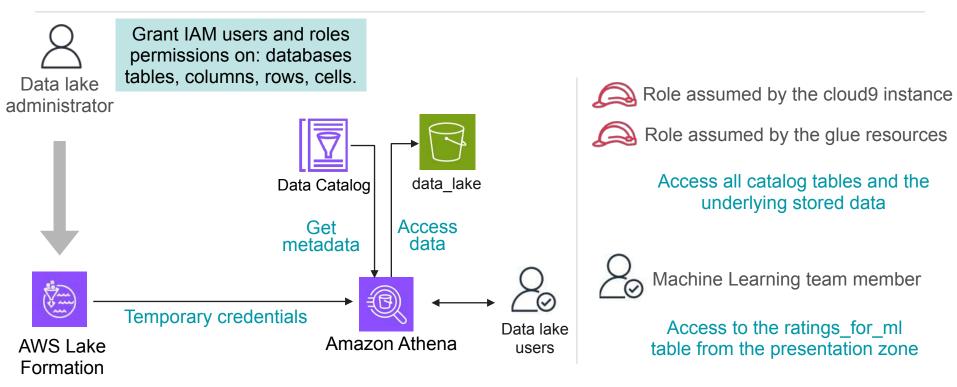


- Metadata-level (catalog resources)
- Storage access



- 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 finegrained permissions to access specific resources

- Metadata-level (catalog resources)
- Storage access





Storage Abstractions

Summary

Data Warehouse



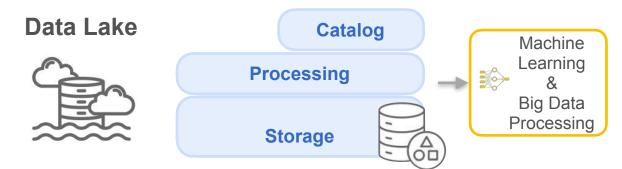


- Low-latency query performance
- High storage cost

Data Warehouse

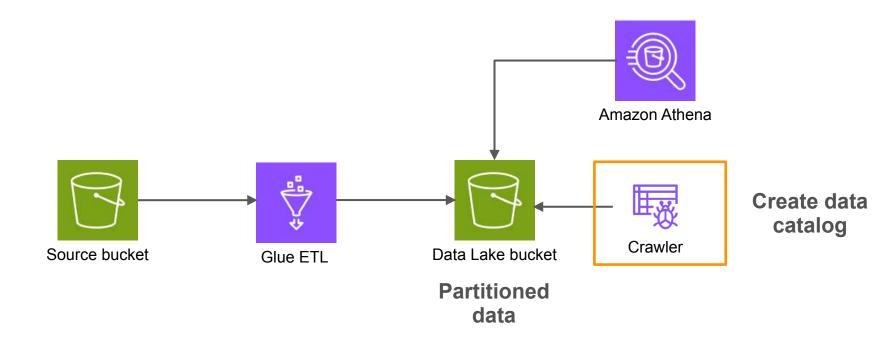


- Low-latency query performance
- High storage cost



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

First lab



Data Warehouse





- Low-latency query performance
- High storage cost

Data Lake





Storage





- 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

