



PLURALSIGHT

# Data Engineering



Tech Catalyst Bootcamp



**Tarek Atwan**  
Instructor, Pluralsight

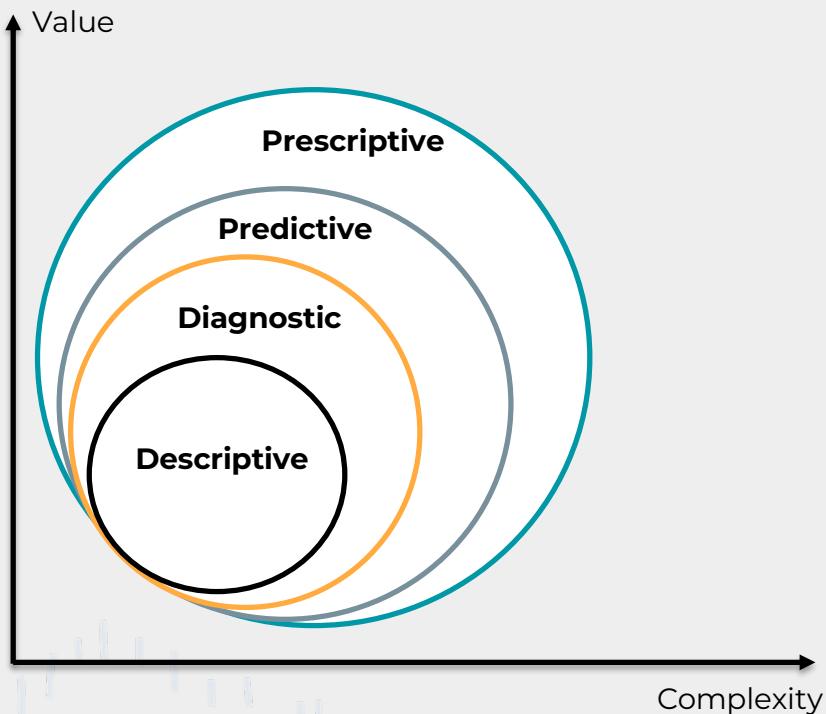
Proprietary and confidential

 PLURALSIGHT

# What is the end goal?

---

# Types of Data Analytics



## Descriptive

What is happening in my business?

## Diagnostic

Why is it happening?

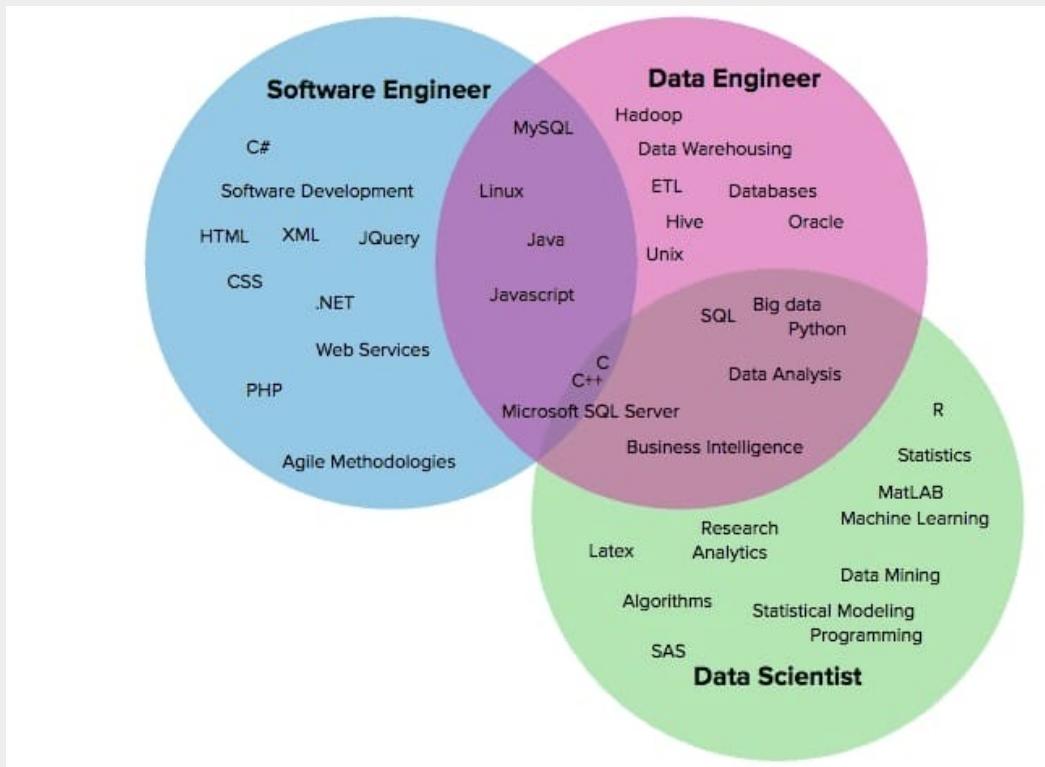
## Predictive

What is likely to happen?

## Prescriptive

What do I need to do?

# Data Engineer Skills

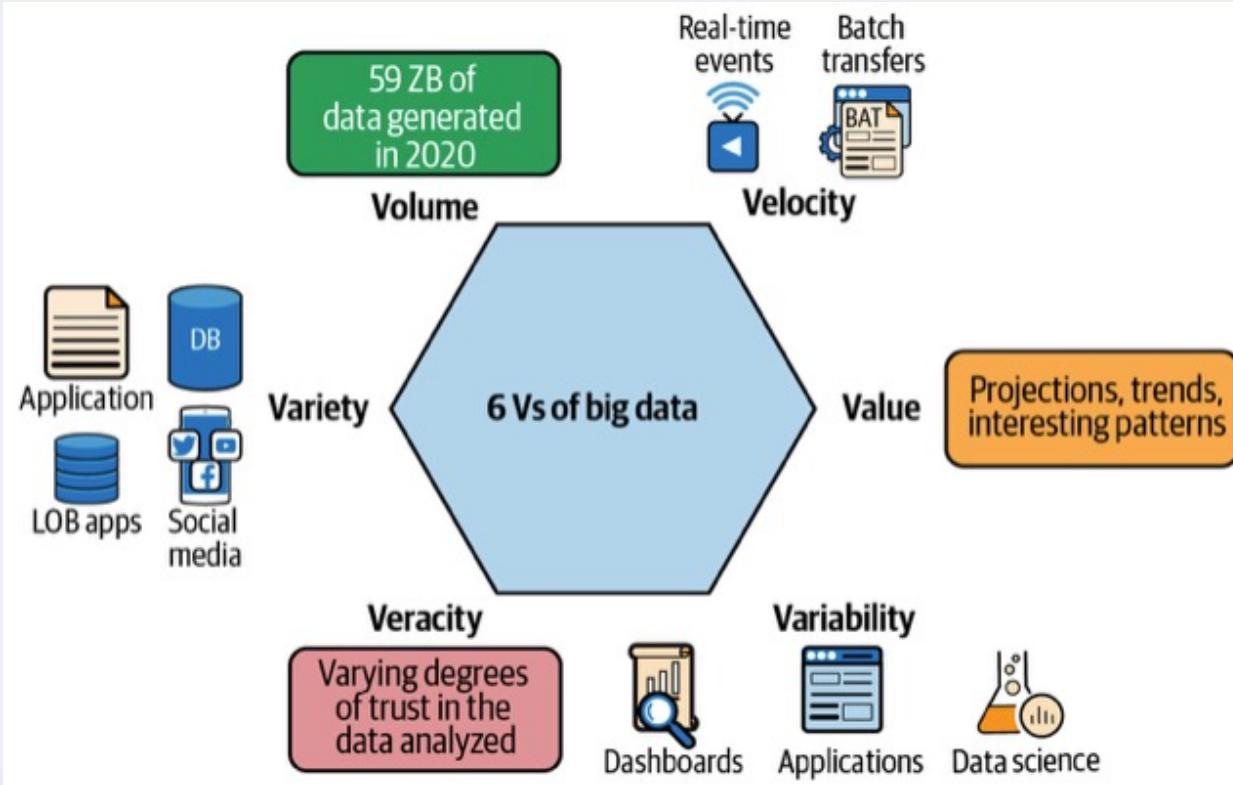


Proprietary and confidential

# Data Engineering Concepts Review

---

# What is Big Data



# Types of Data

## Unstructured data

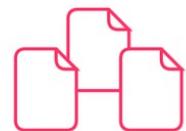
The university has 5600 students.  
John's ID is number 1, he is 18 years old and already holds a B.Sc. degree.  
David's ID is number 2, he is 31 years old and holds a Ph.D. degree. Robert's ID is number 3, he is 51 years old and also holds the same degree as David, a Ph.D. degree.

## Semi-structured data

```
<University>
  <Student ID="1">
    <Name>John</Name>
    <Age>18</Age>
    <Degree>B.Sc.</Degree>
  </Student>
  <Student ID="2">
    <Name>David</Name>
    <Age>31</Age>
    <Degree>Ph.D. </Degree>
  </Student>
  ...
</University>
```

## Structured data

ID	Name	Age	Degree
1	John	18	B.Sc.
2	David	31	Ph.D.
3	Robert	51	Ph.D.
4	Rick	26	M.Sc.
5	Michael	19	B.Sc.



### Unstructured

PDFs, JPEGs, MP3, Movies, ...

### Semi-structured

CSV, JSON, XML, MongoDB, ...

### Structured

Oracle, MSSQL, MySQL, DB2, ...

# Types of Data

## Structured Data

Data that is **organized** in a define manner or **schema**, typically found in **relational databases**.

- Easily queryable
- Organized in rows and columns
- Has a consistent structure

## Unstructured Data

Data that doesn't have a predefined structure or schema.

- Not easily queryable without preprocessing
- May come in various formats

## Semi-Structured Data

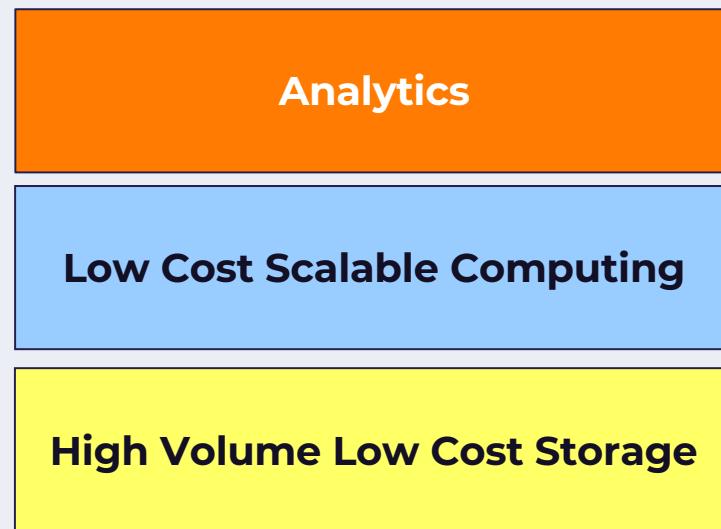
Data that is not as organized as structured data but has some level of structure in the forms of tags, hierarchies, or other patterns.

- Elements might be tagged or categorized in some way
- More flexible than structured data but not as chaotic as unstructured data.

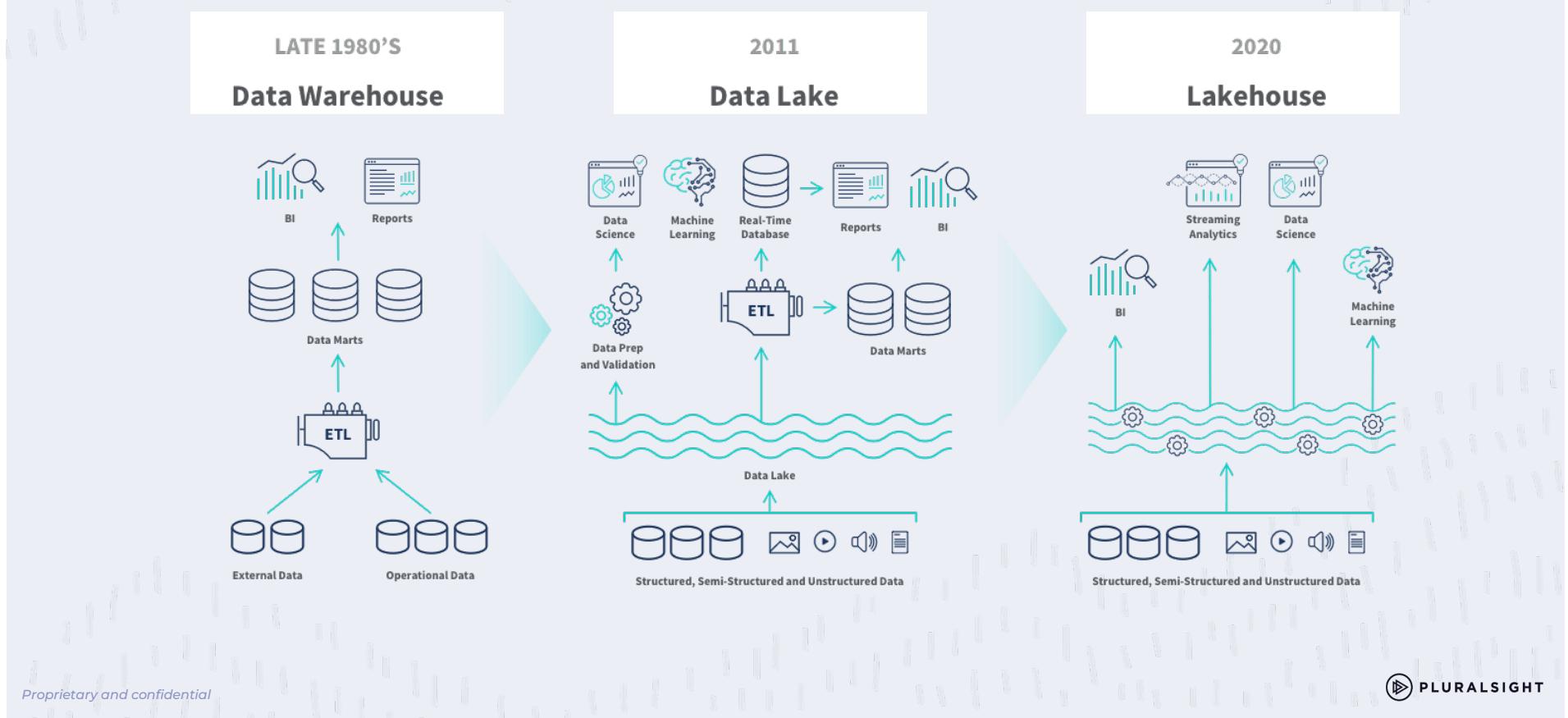
# Big Data Stack

A Big Data stack typically has three layers

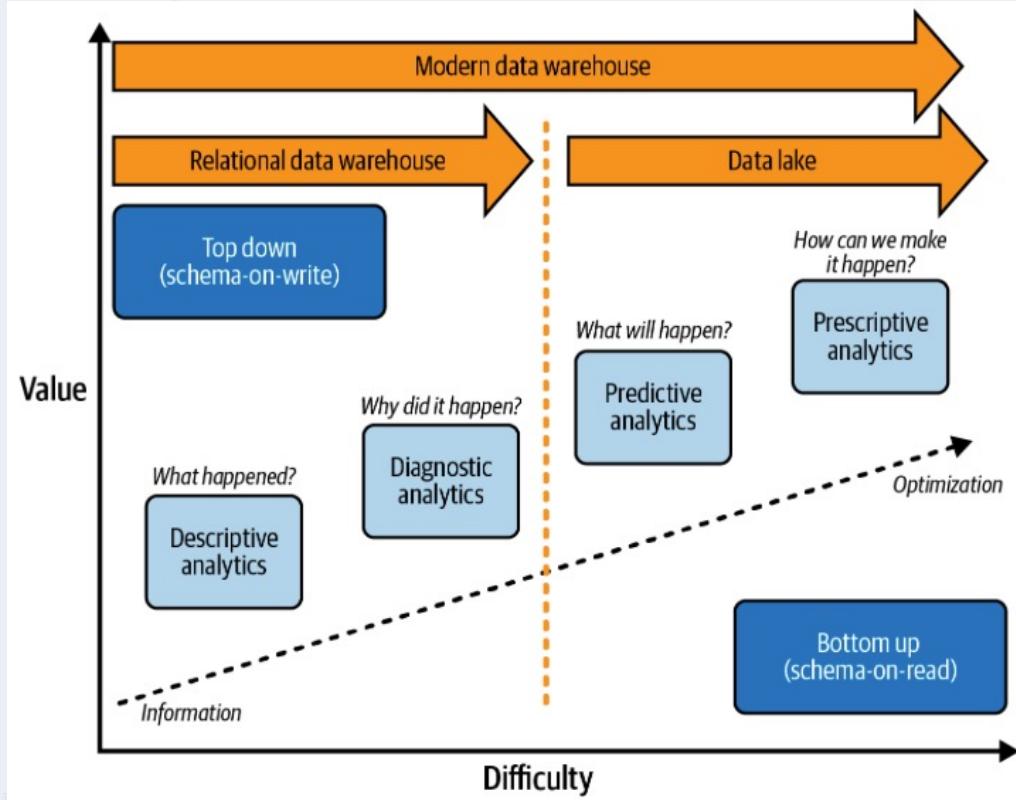
The technology and tools at each layer vary widely dependent on business needs



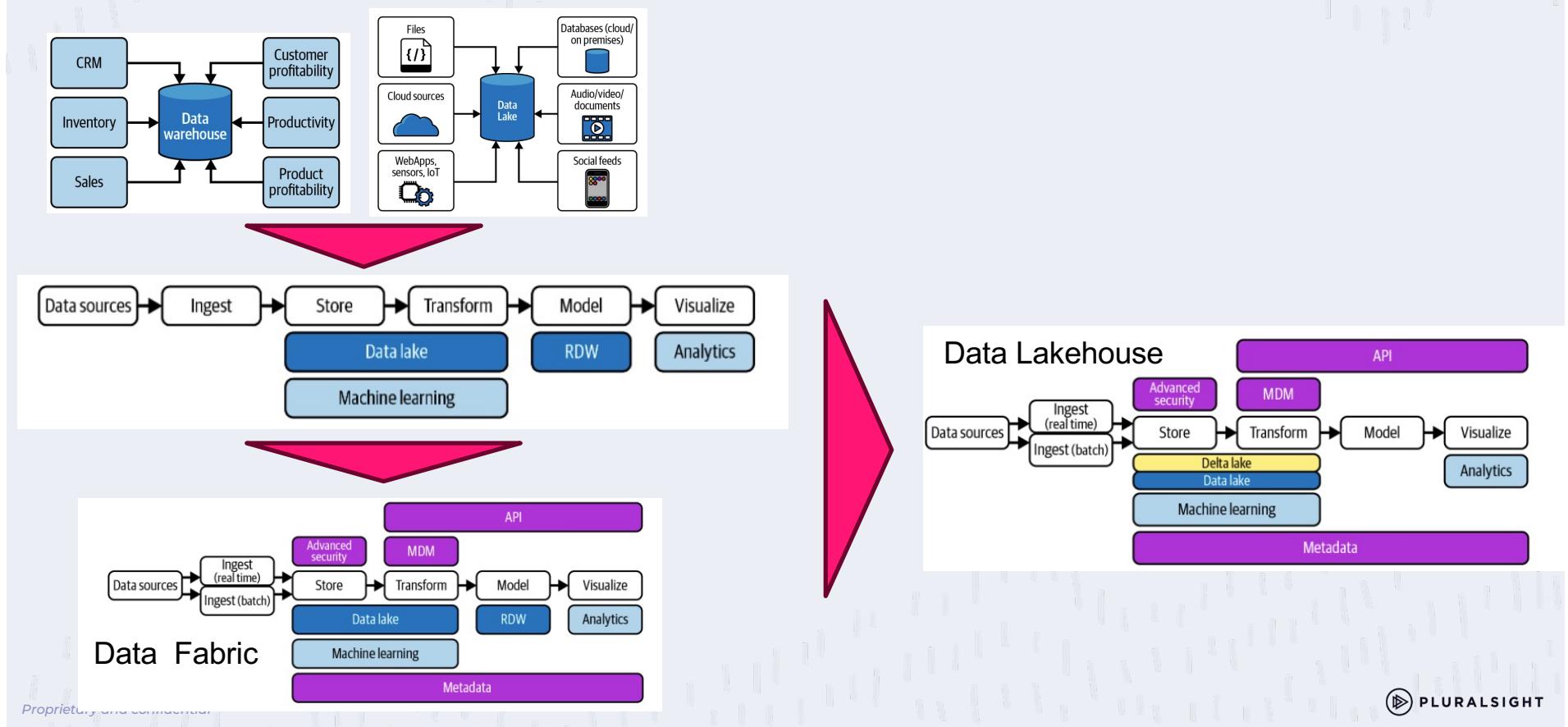
# Modern Data Architecture



# Big Data Analytics



# Evolution Summary



Proprietary and Confidential

# Data Sources

- **JDBC**
  - Java Database Connectivity
  - Platform-independent
  - Language-dependent
- **ODBC**
  - Open Database Connectivity
  - Platform-dependent (using drivers)
  - Language-independent
- **Raw Logs**
- **API's**
- **Streams**
- **CSV**
- **XML**
- **JSON**
- **Avro**
- **Parquet**
- **ORC**

# Avro, ORC, and Parquet

Feature	Avro	Parquet	ORC
Data Format	Row-oriented	Columnar	Columnar
Schema Evolution	Excellent	Good	Good
Compression	Snappy, Deflate, Bzip2	Snappy, Gzip, LZO	Zlib, Snappy
Read Performance	Good for all rows	Excellent for reading specific columns	Excellent for reading specific columns
Write Performance	Excellent	Good	Good
Support for Complex Data Types	Yes	Yes	Yes
Use Cases	Data serialization, streaming data	Analytical queries, BI tools	Analytical queries, BI tools
Schema Storage	Included in file	Separate schema file	Separate schema file
Interoperability	Good (used in Hadoop ecosystem, Kafka)	Good (used in Hadoop ecosystem, Presto, Spark)	Good (used in Hadoop ecosystem, Hive, Spark)
Tooling and Libraries	Avro libraries available for most languages	Supported by major big data tools	Supported by major big data tools
File Size	Larger due to row-oriented storage	Smaller due to columnar compression	Smaller due to columnar compression
Suitability for Small Files	Good	Not optimal	Not optimal
Complex Queries	Moderate	Excellent	Excellent
Splitable	Yes	Yes	Yes

# Data Engineering Process

---

# What is ETL

---

# ETL Pipelines

ETL stands for **Extract, Transform, and Load**.

It is a process used to move data from source systems into a data warehouse.

# ETL Pipelines - Extract

## Extract

- Retrieve raw data from source systems, which can be databases, CRMs, flat files, APIs, or other data repositories
- Ensure data integrity during the extraction phase
- Can be done in real-time or in batches, depending on requirements

# ETL Pipelines - Transform

## Transform

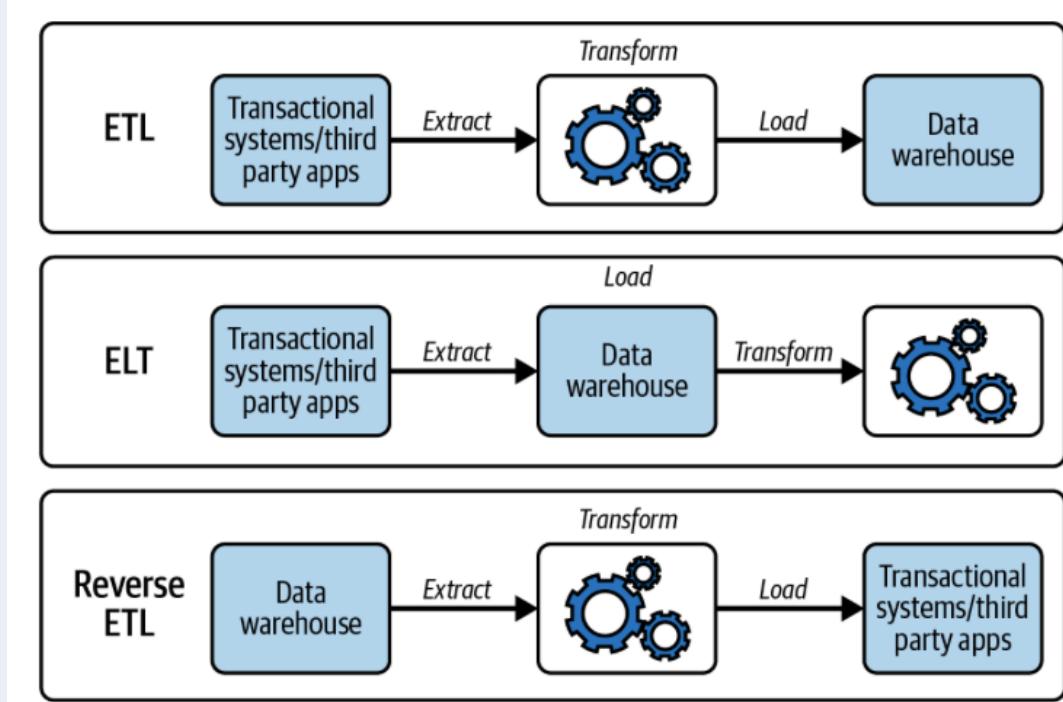
- Convert extracted data into a format suitable for the target data warehouse
- Can involve various operations such as:
  - Data cleansing (e.g. removing duplicates, fixing errors)
  - Data enrichment (e.g. adding additional data from other sources)
  - Format changes (e.g. data formatting, string manipulation)
  - Aggregates or computations (e.g. calculating totals or averages)
  - Encoding or decoding data
  - Handling missing values

# ETL Pipelines - Load

## Load

- Move the transformed data into the target data warehouse or another data repository
- Can be done in batches (all at once) or in a streaming manner (as data becomes available)
- Ensure that data maintains its integrity during the loading phase

# ETL, ELT, and Reverse ETL



# Data Engineering

## DATA ENGINEERING PROCESS

Data ingestion

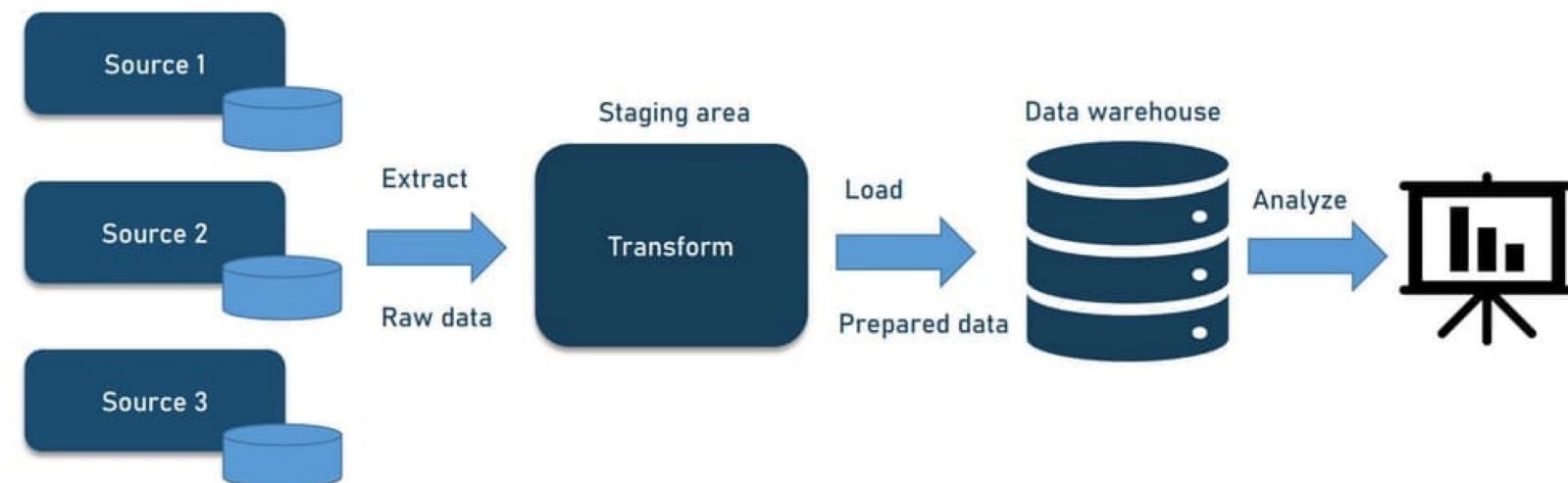
Data transformation

Data serving

Data flow orchestration

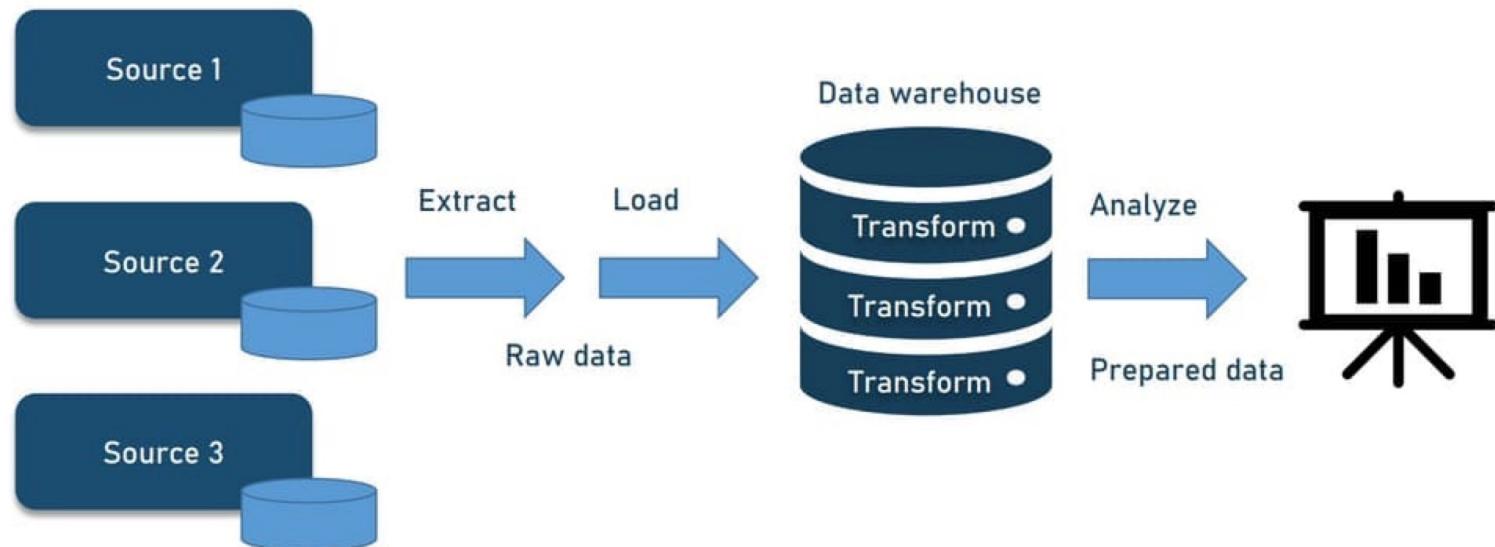
# Data Engineering

## ETL PIPELINE

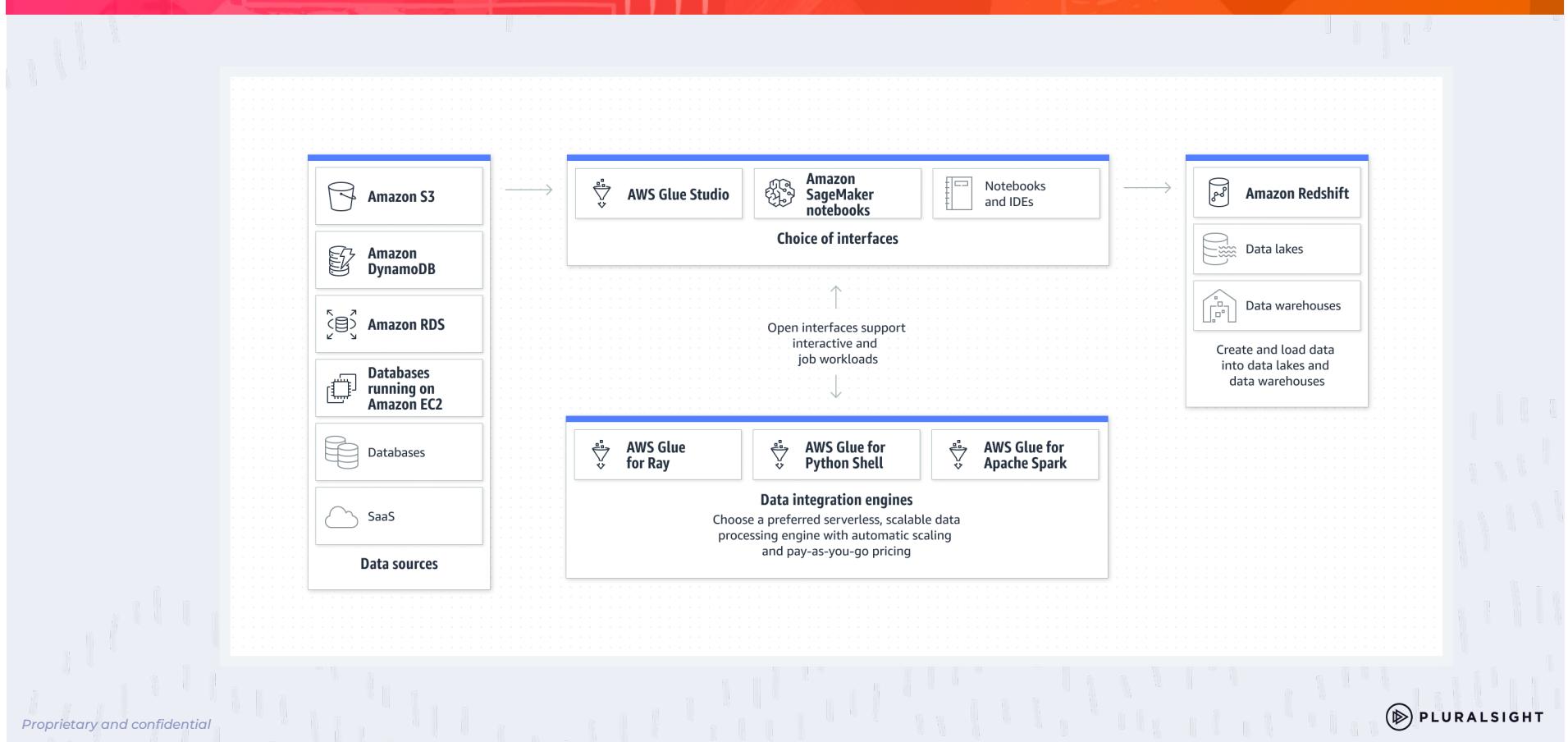


# Data Engineering

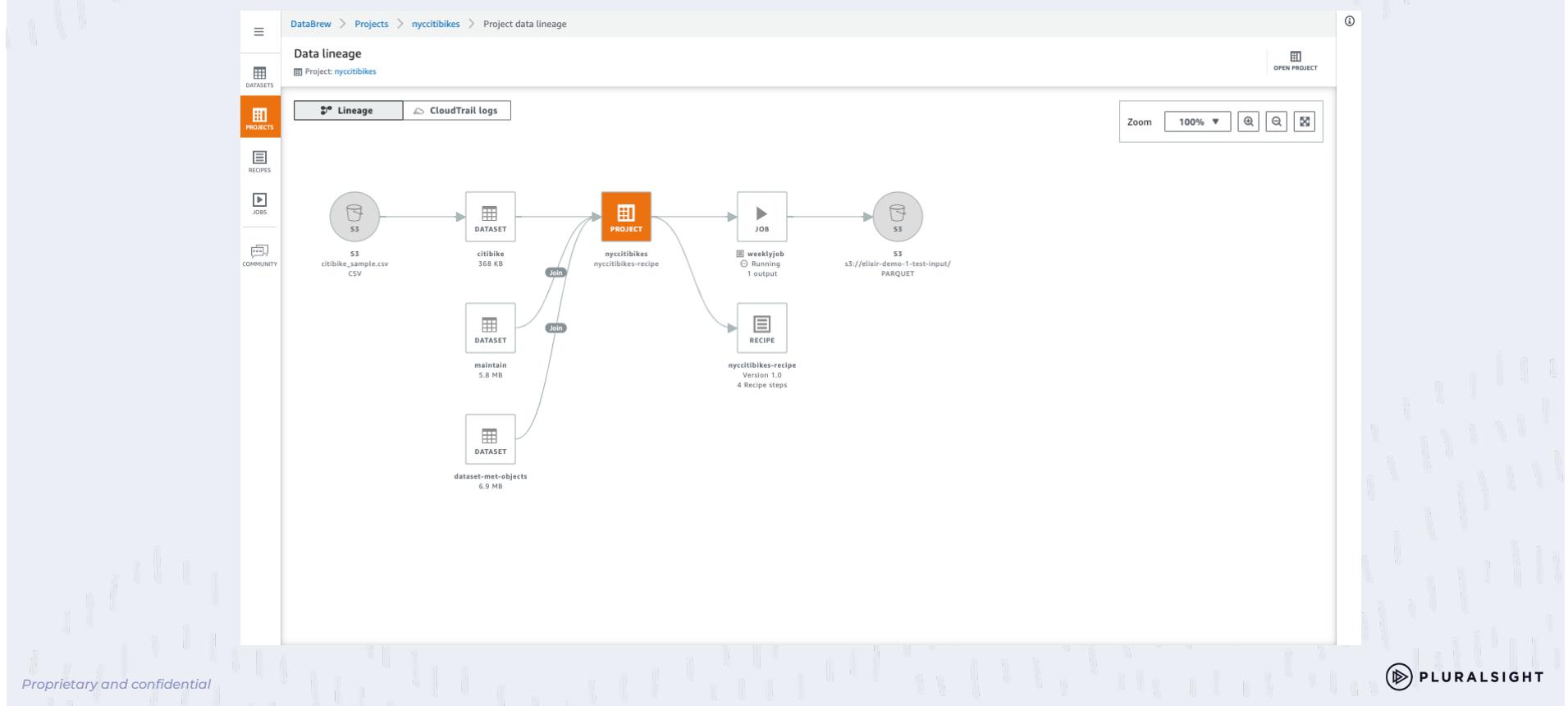
## ELT PIPELINE



# AWS Glue



# AWS Glue DataBrew



# AWS Glue DataBrew

The screenshot shows the AWS Glue DataBrew interface for the "citibike" dataset. The dataset is based on "citibike\_sample.csv" (368 KB) from S3. The left sidebar includes links for Datasets, Projects, Recipes, Jobs, and Community.

The main content area has tabs for Dataset preview, Data profile overview, Column statistics (selected), and Data lineage. The "Column statistics" tab displays the following information:

- Columns (17):** tripduration, day, starttime\_2, stoptime, start station id, start station name, start station latitude, start station longitude, end station id, end station name, end station latitude, end station longitude, bikeid, usertype, usertype\_mapped, birth year, gender.
- # Integer:** start station id
- Data quality:** 2500 valid values, 0 missing values (100%).
- Data insights:** 29% of the rows are unique (744). No missing values.
- Value distribution:** Unique 744, Total 2,500. Histogram showing the distribution of start station id values. Statistics: Min 79, Median 3.1 K, Mean 2.08 K, Mode 151, Max 4.13 K. Skew value: -0.13 (Negative skew).
- Correlations:** Correlation coefficient ( $r$ ) ranges from -1.0 to +1.0. Top correlations:

Column	Correlation coefficient ( $r$ )
tripdur...	-0.02
start st...	1
start st...	0.24
start st...	0.54
end st...	0.36
end st...	0.2
end st...	0.43
bikeid	0.01
usertype	0.06
birth y...	0.03
- Top 50 unique values:** A search bar for finding specific values.

At the bottom right is the Pluralsight logo.

# AWS Glue DataBrew

nyccitibikes

Dataset: citibike | Sample: First n sample (500 rows)

No job runs, no job runs scheduled

Run job

JOBS DETAILS LINEAGE ACTIONS

RECIPE

Viewing 21 columns ▾ 500 rows  View highlighted

SOURCE will be deleted SOURCE will be deleted PREVIEW

# start station latitude # start station longitude ABC latlong # end station id

Total 500 Unique 334 Total 500 Unique 334 Total 500 Unique 334

Min 40.66 Median 40.74 Mean 40.74 Mode 40.72 Max 40.85 Min -74.02 Median -73.98 Mean -73.98 Mode -74 Max -73.9

All other values 484 96.8% 3966 1.2% 3668 1.2% 3164 1.2% 3906 1.2% 128 1.2% 3774 1.2% 462 1.2% 470 1.2% 312 1.2% 372 1.2% 400 1.2% 405 1.2% 3629 1.2% 3070 1.2% 487 1.2% 3585 1.2% 3041 1.2% 3119 1.2% 325 1.2% 3160 1.2% 468 1.2% 3812 1.2% 500 1.2%

Merge columns

Merge columns Info Merge columns and create a new column

Source column Select two or more columns in the order to merge

start station latitude X  
start station longitude X

Add a column

Separator - Optional Concatenated values are separated by this

New column name Name of the target column to merge into

latlong

Valid characters are alphanumeric, underscore, and space

Preview shown

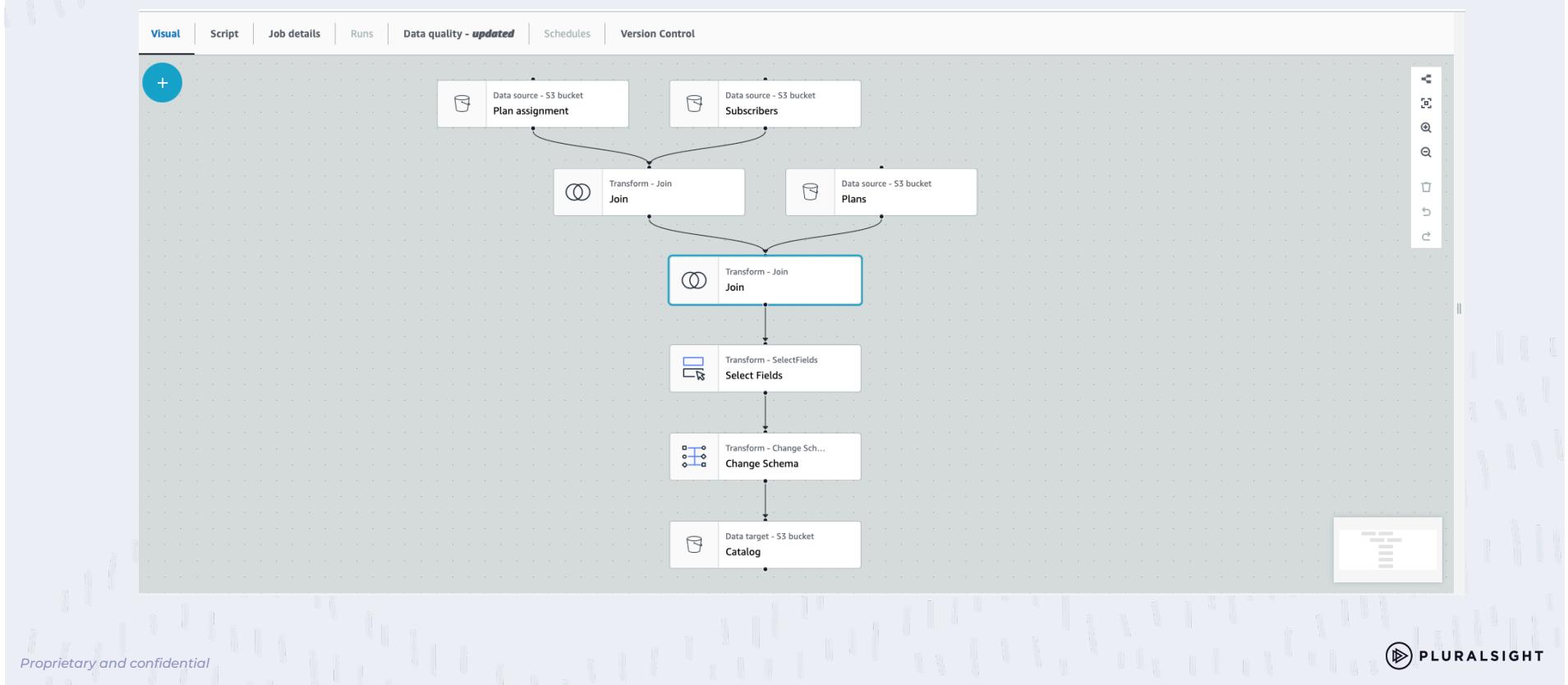
Cancel Apply

Zoom 100% ▾

Proprietary and confidential

ALSIGHT

# AWS Glue Studio (Visual ETL)



# What is Orchestration

---

# Orchestration

- A large part of data engineering is extracting, transforming, and loading data between sources.
- Orchestration is a process of dependency management, facilitated through automation.
- The data orchestrator manages scheduling, triggering, monitoring, and even resource allocation.

# Orchestration

- Workflow management
- Automation
- Error handling
- Recovery
- Monitoring, alerting
- Resource optimization
- Observability
- Debugging
- Compliance/Auditing

# Amazon Step Function

Step 2: Design workflow [Info](#)

Actions **1** Flow

Undo Redo Zoom in Zoom out Center

Cancel Previous Next

**3** Definition

**4** Lambda service integration

The screenshot shows the Step Functions Workflow Studio interface. On the left, a sidebar lists various actions: Lambda Invoke, SNS Publish, ECS RunTask, Step Functions StartExecution, Glue StartJobRun, Glue DataBrew StartJobRun, EventBridge PutEvents, Batch SubmitJob, API Gateway Request, and Athena StartQueryExecution. The main area displays a workflow diagram with a start node, a 'Lambda: Invoke Check Stock' step, a 'Lambda: Invoke Generate Buy/Sell recommendation' step, an 'SQS: SendMessage Request Human Approval' step, a 'Choice state Buy or Sell?' step with two branches labeled 'Rule #1' and 'Rule #2', a 'Lambda: Invoke Buy Stock' step, a 'Lambda: Invoke Sell Stock' step, an 'SNS: Publish Report Result' step, and an end node. A tooltip 'Check Stock' is shown over the first Lambda step. On the right, a detailed configuration pane for the 'Check Stock' step is open, divided into Configuration, Input, Output, and Error handling tabs. The Configuration tab shows the state name 'Check Stock', API as 'Lambda: Invoke' (with a tooltip 'Info'), function name 'SamTestApp-HelloWorldFunction-13WZW2TBONRU7:\$LATEST', and payload 'Use state input as payload'. The Input tab shows the 'Next state' as 'Generate Buy/Sell recommendation'. The Output tab and Error handling tabs are collapsed.

**Check Stock**

Configuration Input Output Error handling

State name

API **4** Lambda: Invoke [Info](#)

Function name The Lambda function to invoke

View function

Payload The JSON that you want to provide to your Lambda function.

Next state

Comment - optional

Step Functions integrates with Lambda so that you can invoke Lambda functions directly from your workflow, using a Task state. Step Functions will pass your parameter values to the Invoke API.

The Workflow Studio exposes the function name and payload parameters of the [Invoke API](#).

**Lambda task result**

When the API response is returned, it is nested in the task result, which contains SDK metadata. To filter out just the data you need from the task result, you can use one of several output filters.

As a convenience, the Workflow Studio automatically enables the **OutputPath** filter under the output tab to filter out the function payload and pass it to the next state. Edit the **OutputPath** field to change this default.

# Apache Airflow

The screenshot displays the Apache Airflow web interface against a red-to-orange gradient background. At the top left is the Apache Airflow logo. To its right is the main title "Apache Airflow". Below the title are two sections: "DAGs" and "XComs".

**DAGs:** This section shows a list of three DAGs: "example\_nested\_branch\_dag", "space\_life\_sciences\_pipeline", and "tutorial\_taskflow\_api". Each entry includes a status indicator (green circle), owner ("airflow"), runs (number of runs), schedule, last run, next run, and a "Details" button.

**XComs:** This section shows a table of XCom records from the DAG "example\_nested\_branch\_dag". The table has columns: Key, Value, Timestamp, Dag ID, and Task ID. The data is as follows:

Key	Value	Timestamp	Dag ID	Task ID
skipper_key	(followed: true_1)	2023-04-01, 11:58:32	example_nested_branch_dag	branch_1
return_value	true_1	2023-04-01, 11:58:32	example_nested_branch_dag	branch_1
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	branch_2
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	branch_2
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	false_1
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	false_1
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	false_2
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	false_2
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	false_3
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	false_3
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	join_1
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	join_1
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	join_2
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	join_2
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	true_1
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	true_1
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	true_2
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	true_2
skipper_key	(followed: true_1)	2023-04-01, 11:58:33	example_nested_branch_dag	branch_1
return_value	true_1	2023-04-01, 11:58:33	example_nested_branch_dag	branch_1

Proprietary and confidential

PLURALSIGHT

# SQL Databases

---

*Proprietary and confidential*



# Database Ranking

<https://db-engines.com/en/ranking>

- Complete ranking
- Relational DBMS
- Key-value stores
- Document stores
- Time Series DBMS
- Graph DBMS
- Search engines
- RDF stores
- Object oriented DBMS
- Vector DBMS
- Wide column stores
- Multivalue DBMS
- Spatial DBMS
- Native XML DBMS
- Event Stores
- Content stores
- Navigational DBMS

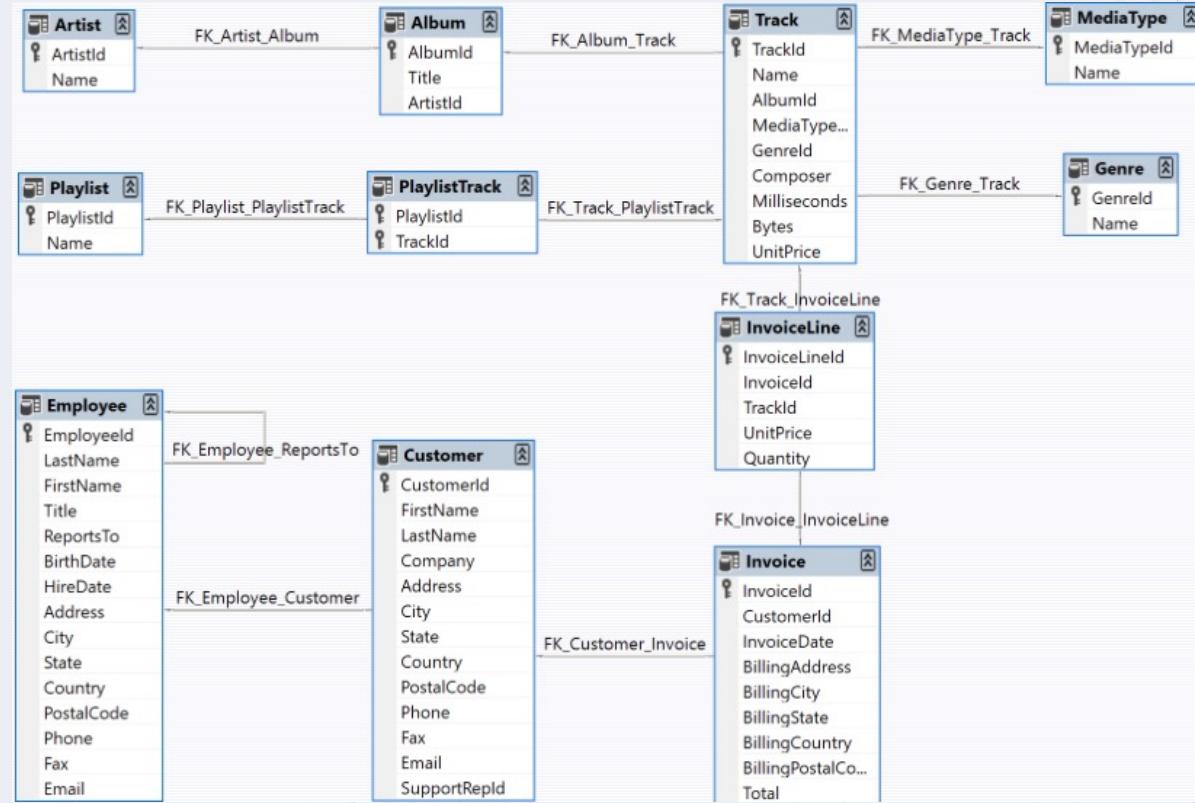
421 systems in ranking, June 2024										
Rank			DBMS			Database Model		Score		
Jun 2024	May 2024	Jun 2023						Jun 2024	May 2024	Jun 2023
1.	1.	1.	Oracle	+		Relational, Multi-model	ⓘ	1244.08	+7.79	+12.61
2.	2.	2.	MySQL	+		Relational, Multi-model	ⓘ	1061.34	-22.39	-102.59
3.	3.	3.	Microsoft SQL Server	+		Relational, Multi-model	ⓘ	821.56	-2.73	-108.50
4.	4.	4.	PostgreSQL	+		Relational, Multi-model	ⓘ	636.25	-9.30	+23.43
5.	5.	5.	MongoDB	+		Document, Multi-model	ⓘ	421.08	-0.58	-4.29
6.	6.	6.	Redis	+		Key-value, Multi-model	ⓘ	155.94	-1.86	-11.41
7.	7.	↑ 8.	Elasticsearch			Search engine, Multi-model	ⓘ	132.83	-2.52	-10.92
8.	↑ 9.	↑ 11.	Snowflake	+		Relational		130.36	+9.03	+16.23
9.	↓ 8.	↓ 7.	IBM Db2			Relational, Multi-model	ⓘ	125.90	-2.56	-18.99
10.	10.	10.	SQLite	+		Relational		111.41	-2.91	-19.81

# Relational Databases

- A relational database management system (RDBMS) is a software layer of tools and services that manages relational tables. In practice, the terms RDBMS and relational database are considered to be synonyms. A relational database provides a consistent interface between applications, users, and relational database. Organizations use RDBMS for managing large amounts of business critical information from various departments.
- Multiple users can work with the same database in different ways. For example, they can perform database operations and aggregate key data points without creating data redundancy. Relational database management systems also give your database administrator greater access control over your data.

# Relational Databases - ERDs

Entities, their data types, and relationships are all illustrated in the diagram.



# How Relational Databases Work

## Data Model

- **Tables:** Represent real-world objects or concepts (entities).
- **Attributes:** Columns in a table holding specific kinds of data.
- **Fields:** Store the actual values of attributes.
- **Primary Key:** Unique identifier for each row in a table.
- **Foreign Key:** References the primary key of another table, creating logical connections between tables.
- **Example:** An orders table can have a foreign key (customer ID) linking to the customer table.

# How Relational Databases Work

## SQL (Structured Query Language)

- **Primary Interface:** Used to communicate with relational databases.
- **ANSI Standard:** Became a standard in 1986, supported by all popular relational database engines.
- **Functions:** Used to update, delete, store, retrieve data, and manage the database.
- **Ease of Use:** Uses common English keywords and integrates well with programming languages like Java.

# How Relational Databases Work

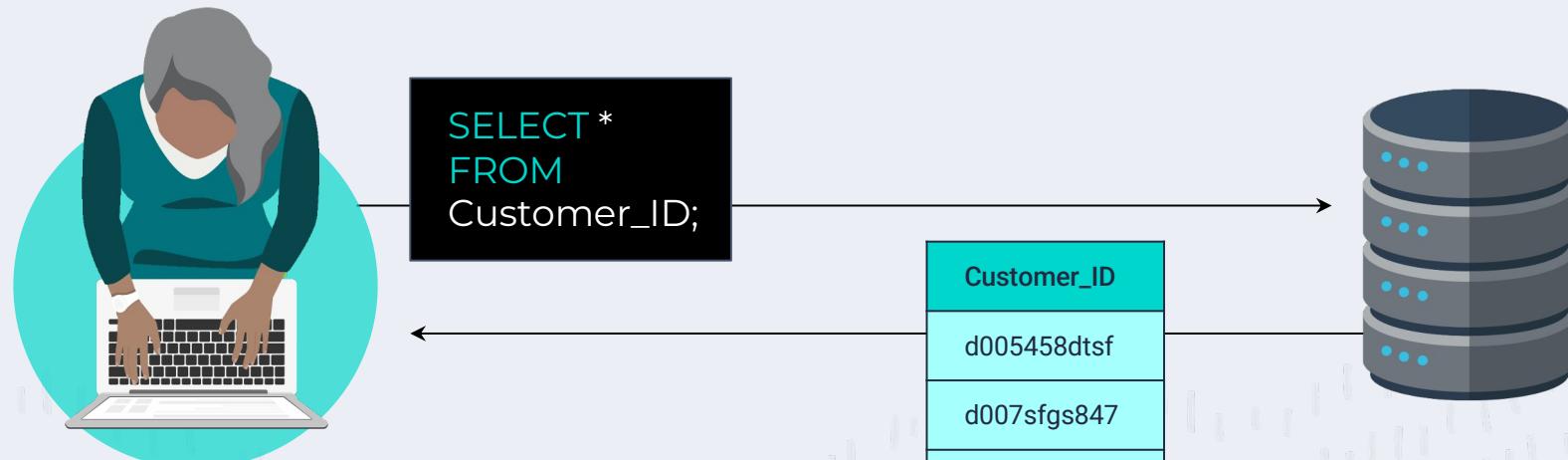
## Transactions

- **Definition:** One or more SQL statements forming a single logical unit of work.
- **All-or-Nothing:** Entire transaction must complete, or none of the components go through.
- **Outcome:** Results in a COMMIT (successful) or a ROLLBACK (unsuccessful).
- **Properties:** Treated coherently, reliably, independently, and isolated from other transactions.

# SQL

SQL (often pronounced "sequel") stands for Structured Query Language.

It is a powerful tool that enables programmers to create, populate, manipulate, and access databases. It also provides an easy method for dealing with server-side storage.



# SQL

Data using SQL is stored in tables on the server, much like spreadsheets you would create in Microsoft Excel.

This makes the data easy to visualize and search.

Customer_ID	Date_ID
d005458dtsf	6/26/2019
d007sfgs847	8/3/2018
d004fgsfh445	12/3/2018

Order_ID	Customer_ID	Date_ID
10001	d005458dtsf	6/26/2019
10002	d007sfgs847	8/3/2018
10003	d004fgsfh445	12/3/2018

# CRUD Operations

**Create Read Update Delete** is a set of operations used with persistent storage.

<b>Create</b>	INSERT INTO table (column1, column2, column3)
<b>Read</b>	SELECT * FROM table
<b>Update</b>	UPDATE table SET column1 = VALUE WHERE id = 1
<b>Delete</b>	DELETE FROM table WHERE id = 5

These tools are fundamental to all programming languages, not just SQL.

# SQL JOINS

## INNER JOIN

Returns records that have matching values in both tables.

## LEFT JOIN

Returns all records from the left table and the matched records from the right table.

## RIGHT JOIN

Returns all records from the right table and the matched records from the left table.

## CROSS JOIN

Returns records that match every row of the left table with every row of the right table. This type of join has the potential to make very large tables.

## FULL OUTER JOIN

Places null values within the columns that do not match between the two tables, after an inner join is performed.

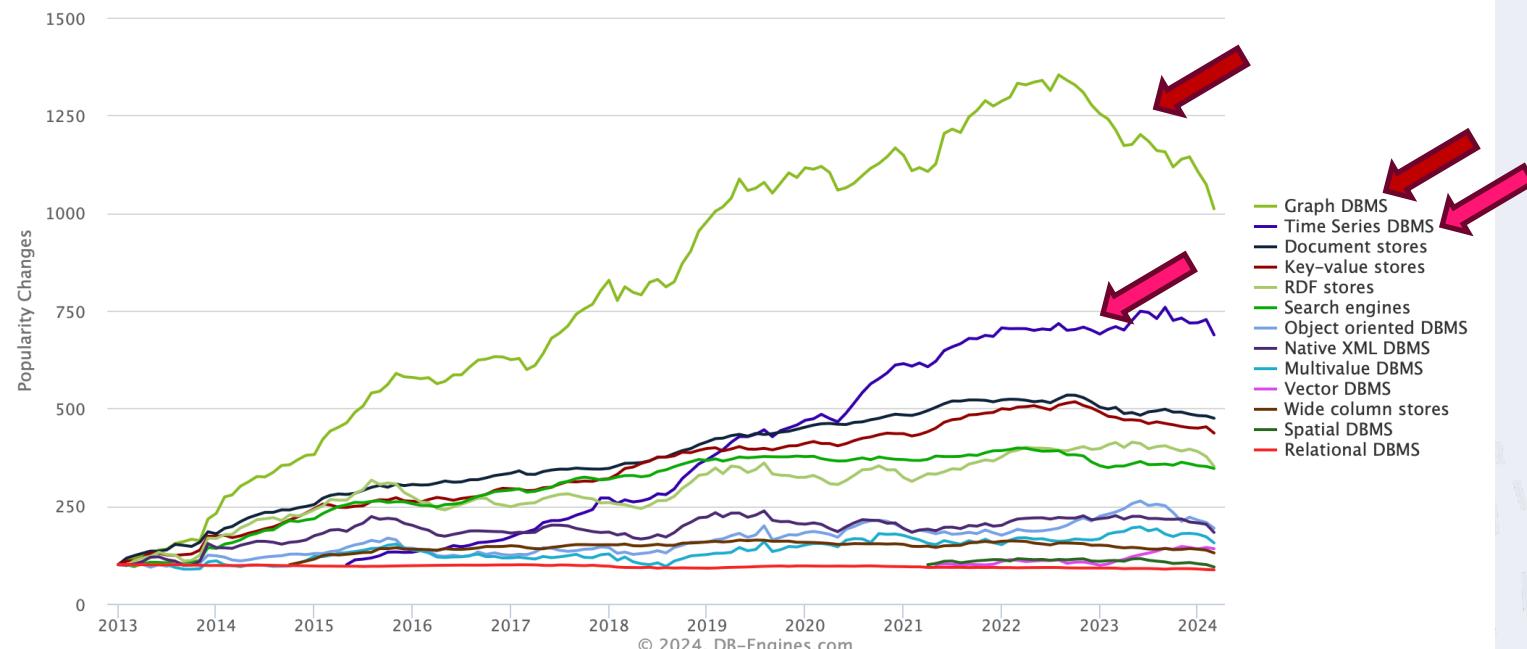
# NoSQL Databases

---

# Database Trends

[https://db-engines.com/en/ranking\\_categories](https://db-engines.com/en/ranking_categories)

Complete trend, starting with January 2013



Proprietary and confidential

PLURALSIGHT

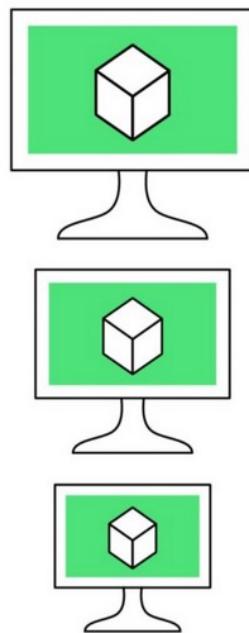
# NoSQL Databases

## Key Features of NoSQL Databases

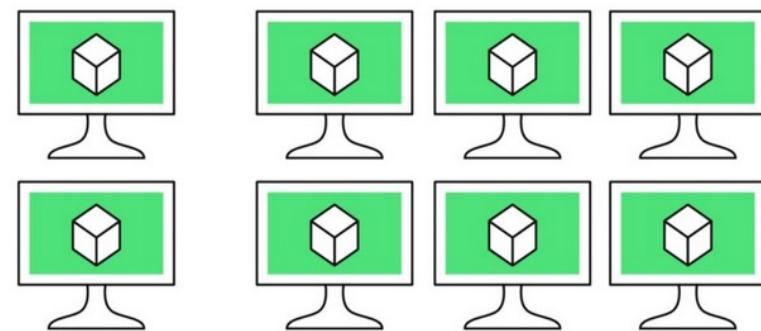
- 1. Flexible Data Models:** NoSQL databases store data in various formats such as JSON, XML, and key-value pairs, which allows for more flexible schema design and easier adaptation to changing data structures.
- 2. Scalability:** NoSQL databases are designed for **horizontal scaling**, which enables them to handle large volumes of data and high traffic without significant performance degradation.
- 3. Simple Design:** NoSQL databases often have simpler designs compared to relational databases, making them easier to implement and manage.
- 4. High Availability:** NoSQL databases are designed to provide high availability and reliability by replicating data across multiple nodes and ensuring that data is accessible even if some nodes go offline.
- 5. Support for SQL-like Query Languages:** Many NoSQL databases support SQL-like query languages, making it easier for developers familiar with SQL to work with these databases

# Scalability

## Vertical scaling



## Horizontal scaling



# SQL vs NoSQL Terminology

SQL	MongoDB	DynamoDB	Cassandra	Couchbase
Table	Collection	Table	Table	Data bucket
Row	Document	Item	Row	Document
Column	Field	Attribute	Column	Field
Primary key	ObjectId	Primary key	Primary key	Document ID
Index	Index	Secondary index	Index	Index
View	View	Global secondary index	Materialized view	View
Nested table or object	Embedded document	Map	Map	Map
Array	Array	List	List	List

# Types of NoSQL Databases

## Document Databases

- Store data in flexible, semi-structured documents, often in JSON or XML format.
- Allow for nested data structures and flexible schemas, making them well-suited for handling unstructured and semi-structured data.
- Examples include MongoDB and Couchbase.

## Key-Value Stores

- Store data as simple key-value pairs, where the value can be anything from a simple string to a complex object.
- Provide fast read/write operations and are optimized for simple data access.
- Examples include Redis and Memcached.

# Types of NoSQL Databases

## Column-Oriented Stores

- Store data in columns instead of rows, which allows for efficient querying and analysis of large datasets.
- Organize data into column families, which are sets of columns treated as a single entity.
- Examples include Cassandra and HBase.

## Graph Databases

- Focus on the relationships between data, storing it in the form of nodes and edges.
- Designed to handle complex, interconnected data and are well-suited for applications that require traversing and analyzing relationships.
- Examples include Neo4j and Amazon Neptune.

# SQL vs NoSQL

## SQL Database

Relational

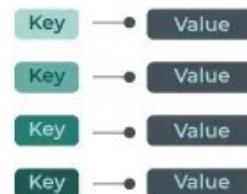


Analytical (OLAP)

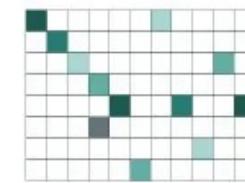


## NoSQL Database

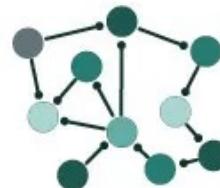
Key - Value



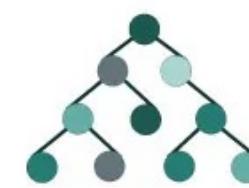
Column - Family



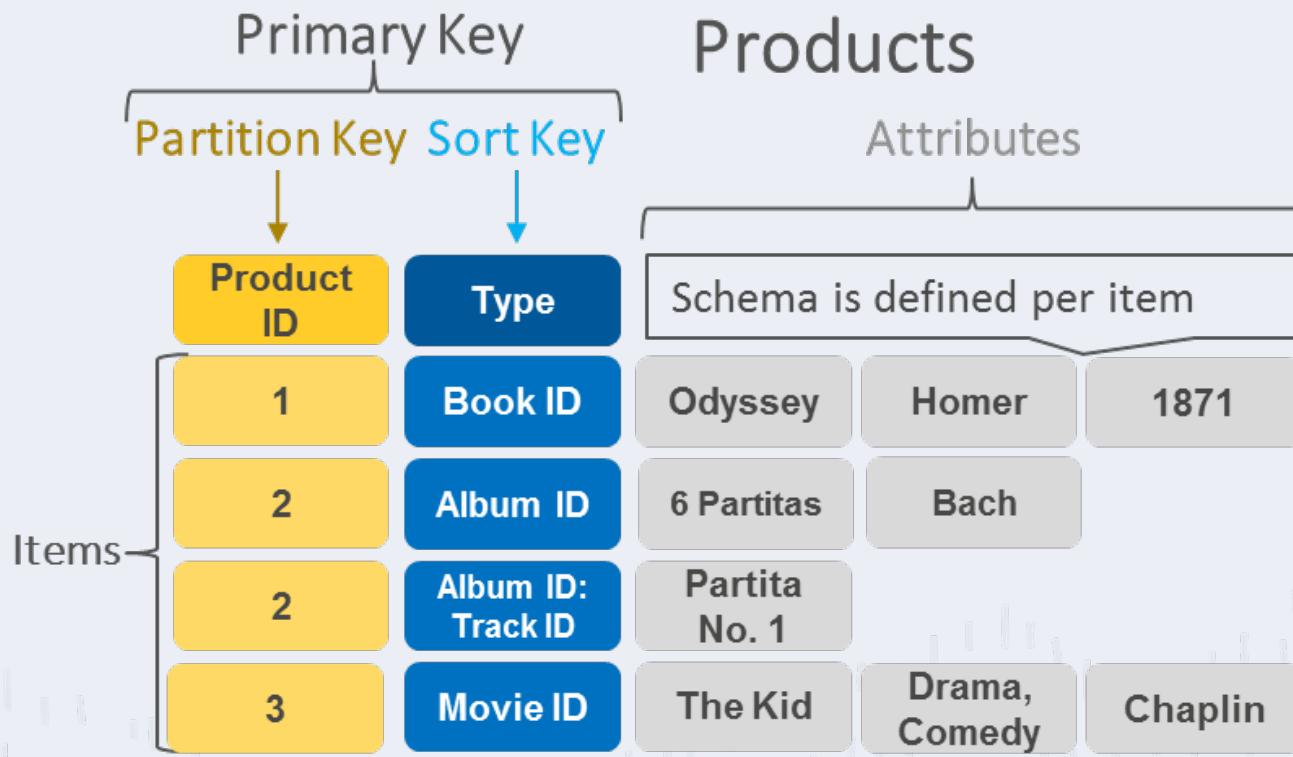
Graph



Document



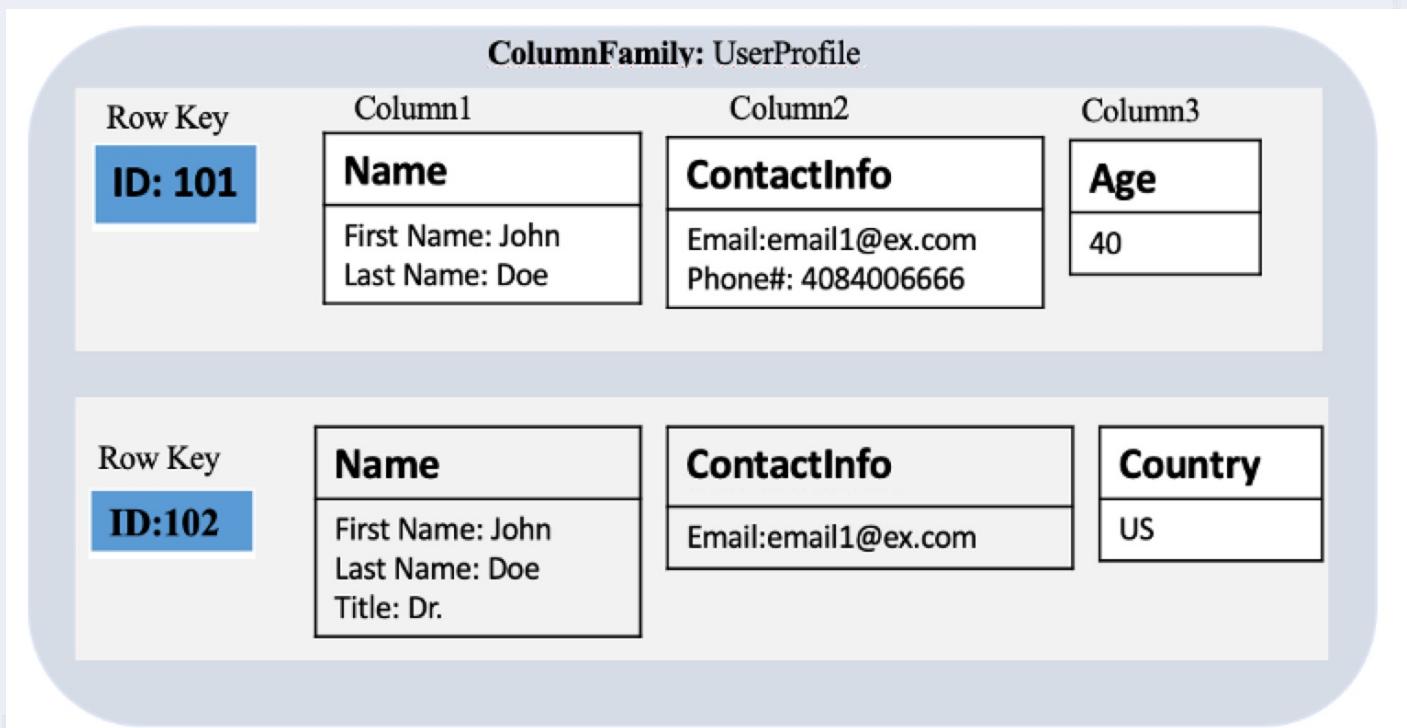
# Key Value



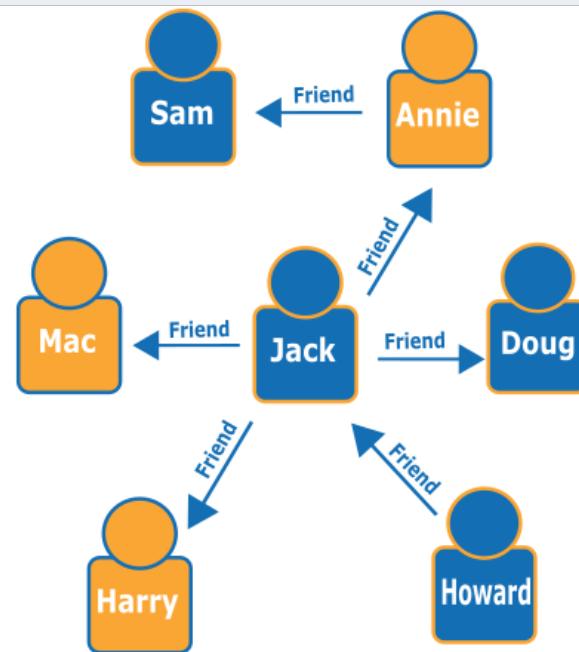
# Document

Key	Document
101	<pre>{   "ID": "1001",   "ItemsOrdered": [     {       "ItemID": "1",       "Quantity": "2",       "cost": "1000",     },     {       "ItemID": "1001",       "Quantity": "2",       "cost": "1000",     }   ],   "OrderDate": "05/11/2019" }</pre>
102	<pre>{   "ID": "1002",   "ItemsOrdered": [     {       "ItemID": "2890",       "Quantity": "11",       "cost": "10000",     }   ],   "OrderDate": "05/11/2019" }</pre>

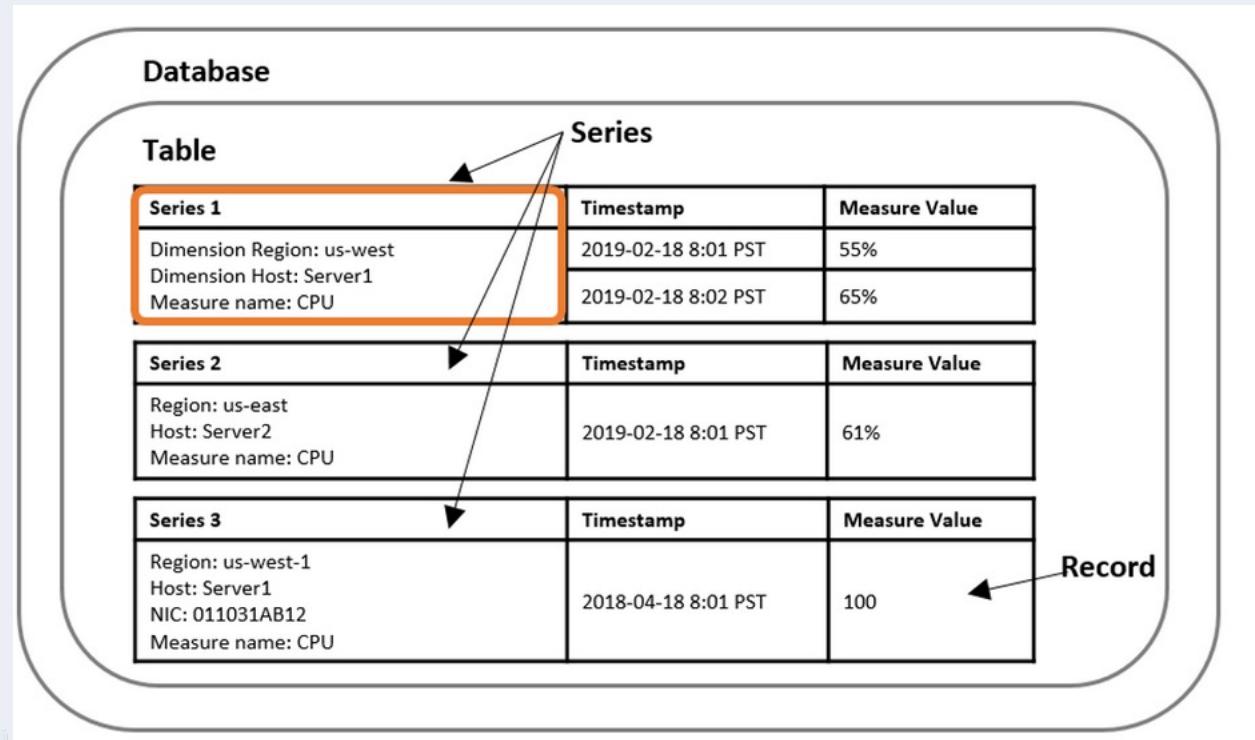
# Wide Columns



# Graph



# Time Series



# Intro to Amazon Web Services

---

# AWS Certifications

## FOUNDATIONAL

Knowledge-based certification for foundational understanding of AWS Cloud.

**No prior experience needed.**



## PROFESSIONAL

Role-based certifications that validate advanced skills and knowledge required to design secure, optimized, and modernized applications and to automate processes on AWS.

**2 years of prior AWS Cloud experience recommended.**



## ASSOCIATE

Role-based certifications that showcase your knowledge and skills on AWS and build your credibility as an AWS Cloud professional. **Prior cloud and/or strong on-premises IT experience recommended.**



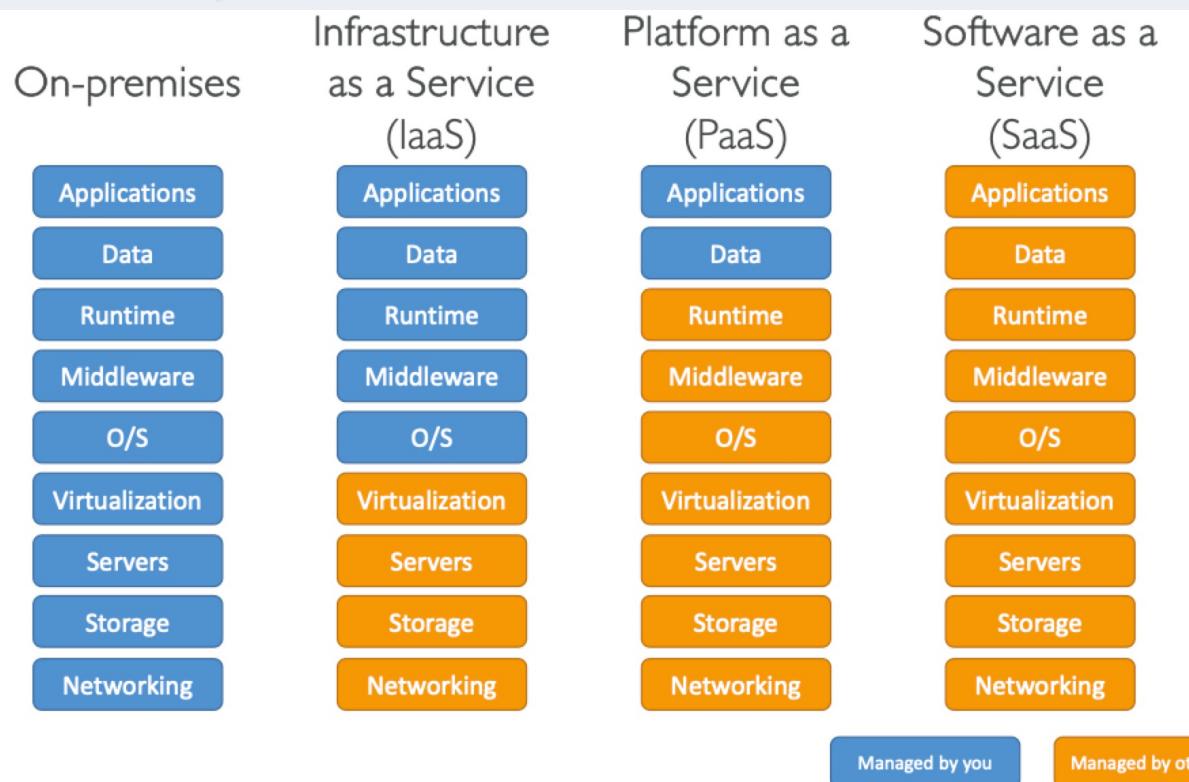
## SPECIALTY

Dive deeper and position yourself as a trusted advisor to your stakeholders and/or customers in these strategic areas. **Refer to the exam guides on the exam pages for recommended experience.**

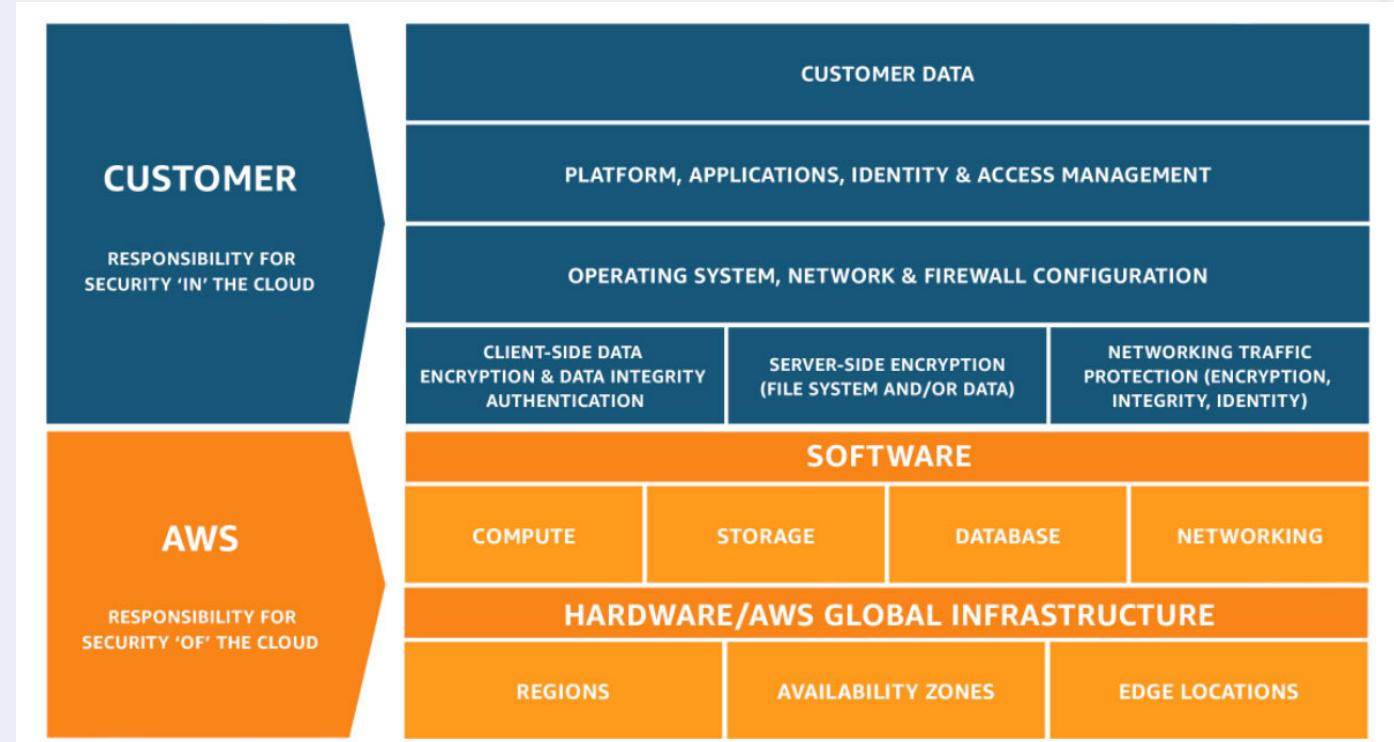


Proprietary and confidential

# Types of Cloud Computing

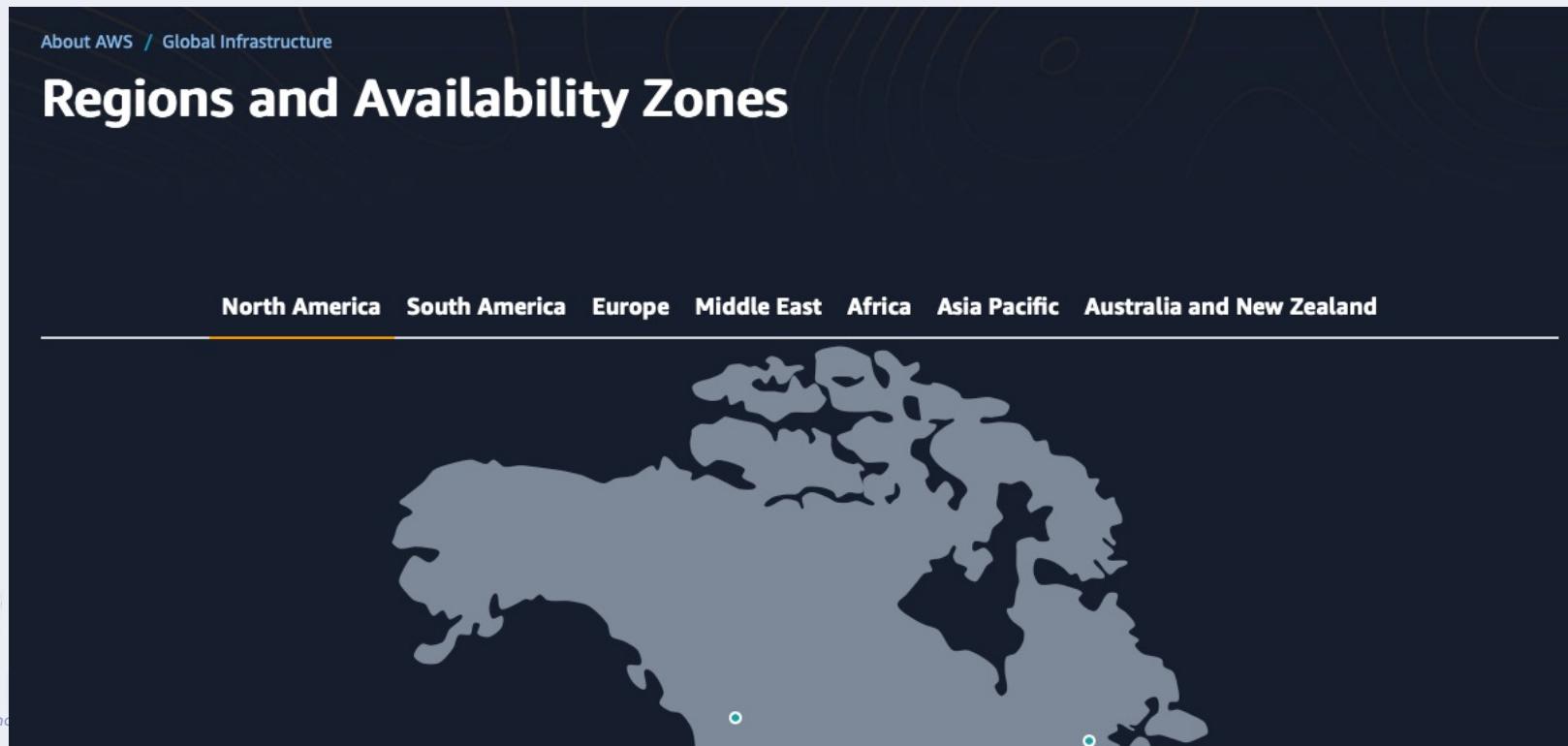


# AWS Shared Responsibility Model



# AWS Regions and Availability Zones

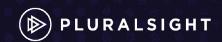
[https://aws.amazon.com/about-aws/global-infrastructure/regions\\_az/](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/)



# Storage

---

*Proprietary and confidential*



# Amazon S3

- ❑ Amazon S3 is one of the main building blocks of AWS
- ❑ Many websites for example use Amazon S3 as a backbone



## Amazon S3 Use Cases

- Backup and Storage
- Disaster Recovery
- Archiving
- Hybrid Cloud Storage
- Application Hosting
- Media Hosting
- Data Lakes and Big Data Analytics
- Software Delivery
- Static Websites

# Amazon S3 - Buckets

- Store Objects “files” in buckets. Think of them as “directories”
- Buckets must have a globally unique name (across all regions and all accounts)
- Buckets are defined at the region level

# Amazon S3 - Objects

- Objects (files) have a key
- The key is the full FULL path:  
`s3://mybucket/myfile.txt`  
`s3://mybucket/folder1/subfolder1/myfile.txt`
- The key is composed of a **prefix** + **object name**  
`s3://mybucket/folder1/subfolder1/myfile.txt`
- There is no concept of “directories” within buckets. Though in the UI you make think otherwise.
- They are just keys with very long names that contain "/" slashes
- Object values are the content of the body: Max object size is 5TB, if large it must be “multi-part” upload.
- Version ID if versioning is enabled
- Metadata

# S3 Bucket Policies

JSON based policies

- Resources: buckets and objects
- Effect: Allow or Deny
- Actions: Set of API to allow or deny
- Principal: The account or user to apply the policy to

You use S3 Bucket policies to:

- Grant public access to the bucket
- Force objects to be encrypted at upload
- Grant access to another account (Cross Account)

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "PublicRead",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Resource": [  
        "arn:aws:s3:::examplebucket/*"  
      ]  
    }  
  ]  
}
```

# Block Public Access

- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on

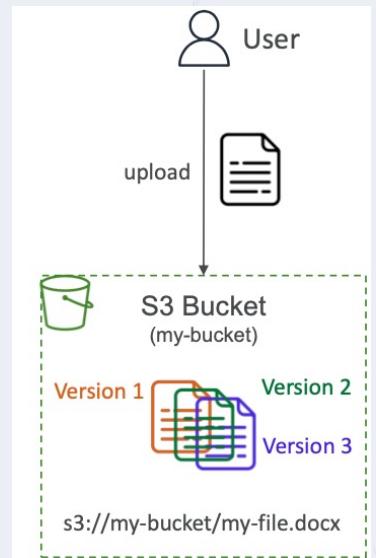
**Block all public access**

On

- Block public access to buckets and objects granted through *new* access control lists (ACLs)  
On
- Block public access to buckets and objects granted through *any* access control lists (ACLs)  
On
- Block public access to buckets and objects granted through *new* public bucket or access point policies  
On
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies  
On

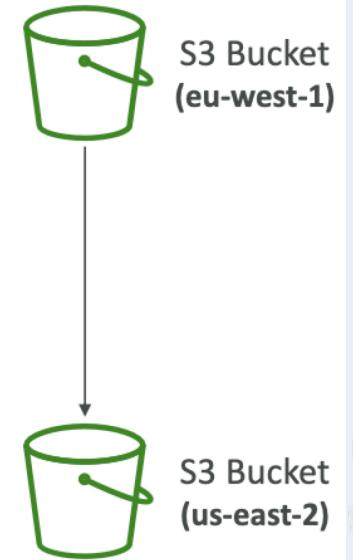
# S3 Versioning

- You can version your files in Amazon S3
- It is enabled at the Bucket Level
- Same key overwrite will change the "version"
- It is best practices to version your buckets
- A file without versioning will have a version "null"
- Suspending versioning does not delete the previous versions



# S3 Replication

- Must enable Versioning in source and destination buckets
- Cross-Region Replication (CRR)
- Same-Region Replication (SRR)
- Bucket can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3
- Once you enable Replication, only new objects are replicated
- You can replicate existing objects using S3 Batch Replication



# S3 Storage Classes

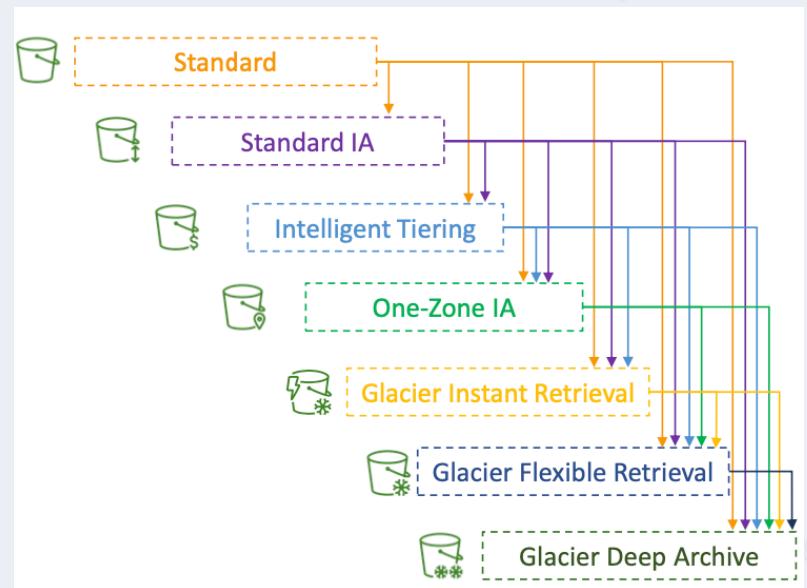
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

# S3 Storage Classes

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	\$0.023	\$0.0025 - \$0.023	\$0.0125	\$0.01	\$0.004	\$0.0036	\$0.00099
Retrieval Cost (per 1000 request)	GET: \$0.0004 POST: \$0.005	GET: \$0.0004 POST: \$0.005	GET: \$0.001 POST: \$0.01	GET: \$0.001 POST: \$0.01	GET: \$0.01 POST: \$0.02	GET: \$0.0004 POST: \$0.03  Expedited: \$10 Standard: \$0.05 Bulk: free	GET: \$0.0004 POST: \$0.05  Standard: \$0.10 Bulk: \$0.025
Retrieval Time	Instantaneous					Expedited (1 – 5 mins) Standard (3 – 5 hours) Bulk (5 – 12 hours)	Standard (12 hours) Bulk (48 hours)
Monitoring Cost (per 1000 objects)		\$0.0025					

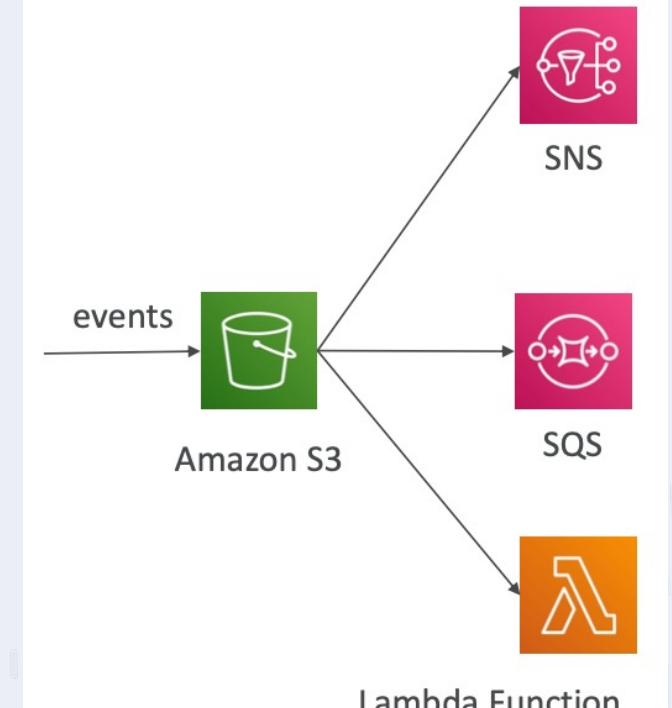
# S3 Storage Classes

- You can transition objects between storage classes
- For infrequently access objects, move them to Standard IA
- For archive objects that you do not need fast access to, move them to Glacier or Glacier Deep Archive
- Moving objects can be automated using Lifecycle Rules

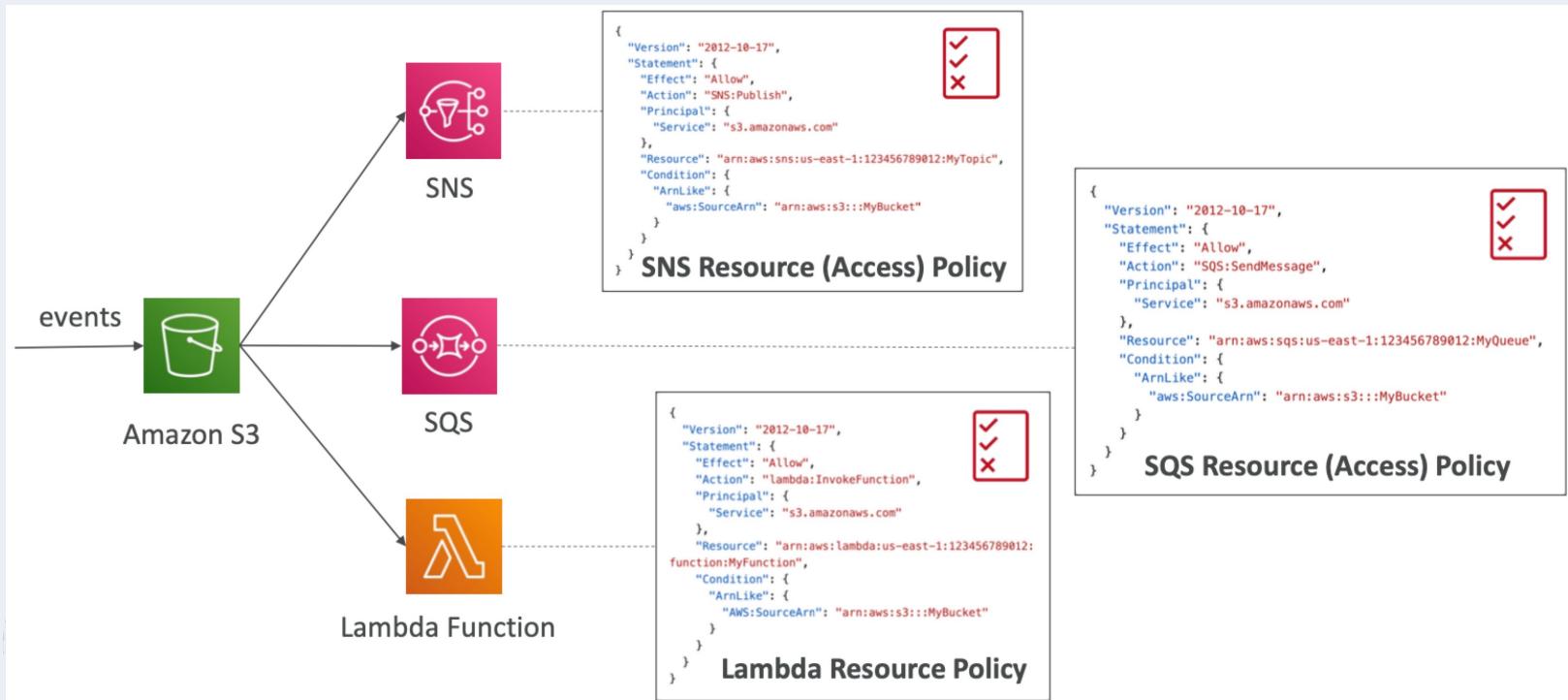


# S3 Event Notification

- Event notification on ObjectCreated, ObjectRemoved, ObjectRestore, ObjectReplication ..etc.
- Object name filtering possible example “\*.jpg”
- S3 events notifications typically delivered in seconds but can sometimes take a minute or longer



# S3 Event Notification – IAM Permissions



# Databases

---



<https://docs.aws.amazon.com/whitepapers/latest/aws-overview/database.html>

# AWS Database Services

Database	Use cases	AWS services
Relational	Traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), e-commerce	<ul style="list-style-type: none"><li>• <a href="#">Amazon Aurora</a> — Designed for unparalleled high performance and availability at global scale with full MySQL and PostgreSQL compatibility</li><li>• <a href="#">Amazon RDS</a> — Set up, operate, and scale a relational database in the cloud with just a few clicks</li><li>• <a href="#">Amazon Redshift</a> — Accelerate your time to insights with fast, easy, and secure cloud data warehousing at scale</li></ul>
Key-value	High-traffic web applications, e-commerce systems, gaming applications	<ul style="list-style-type: none"><li>• <a href="#">Amazon DynamoDB</a> — Fast, flexible NoSQL database service for single-digit millisecond performance at any scale</li></ul>
In-memory	Caching, session management, gaming leaderboards, geospatial applications	<ul style="list-style-type: none"><li>• <a href="#">Amazon ElastiCache</a> — Unlock microsecond latency and scale with in-memory caching</li><li>• <a href="#">Amazon MemoryDB for Redis</a> — Redis-compatible, durable, in-memory database service for ultra-fast performance</li></ul>
Document	Content management, catalogs, user profiles	<ul style="list-style-type: none"><li>• <a href="#">Amazon DocumentDB (with MongoDB compatibility)</a> — Scale JSON workloads with ease using a fully managed document database service</li></ul>
Wide column	High-scale industrial apps for equipment maintenance, fleet management, and route optimization	<ul style="list-style-type: none"><li>• <a href="#">Amazon Keyspaces</a> — A scalable, highly available, and managed Apache Cassandra-compatible database service</li></ul>
Graph	Fraud detection, social networking, recommendation engines	<ul style="list-style-type: none"><li>• <a href="#">Amazon Neptune</a> — Build and run graph applications with highly connected datasets</li></ul>
Time series	Internet of Things (IoT) applications, DevOps, industrial telemetry	<ul style="list-style-type: none"><li>• <a href="#">Amazon Timestream</a> — Fast, scalable, serverless time series database</li></ul>
Ledger	Systems of record, supply chain, registrations, banking transactions	<ul style="list-style-type: none"><li>• <a href="#">Amazon Ledger Database Service (QLDB)</a> — Maintain an immutable, cryptographically verifiable log of data changes</li></ul>

# Amazon RDS

Hosted Relational Database

- Amazon Aurora
- MySQL
- PostgreSQL
- MariaDB
- Oracle
- SQL Server



# Analytics

---



<https://docs.aws.amazon.com/whitepapers/latest/aws-overview/analytics.html>

# AWS Glue



Proprietary and confidential

# Thank you!

**If you have any additional questions, please  
reach out to me at: (email address).**



PLURALSIGHT