
PLURALSIGHT

Data Engineering

THE HARTFORD

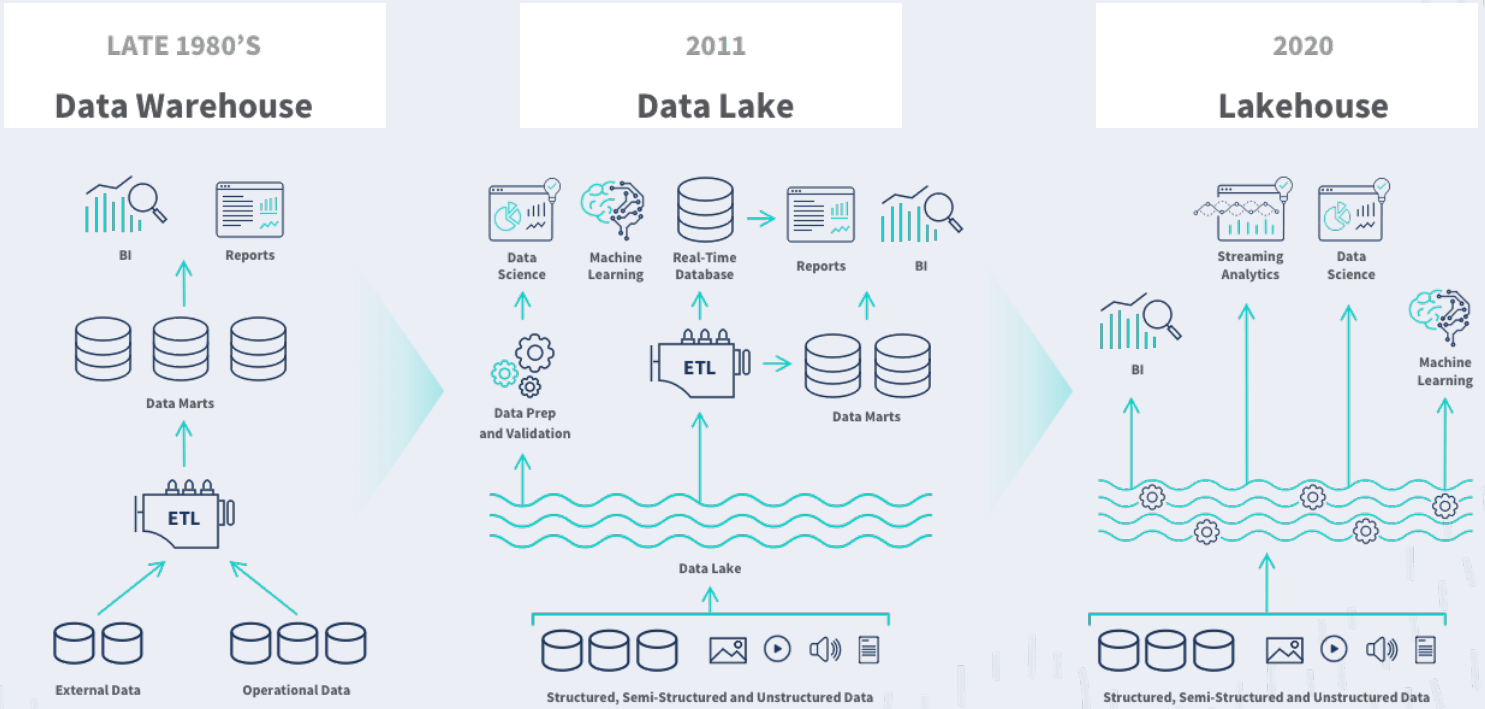
Tech Catalyst Bootcamp

Tarek Atwan
Instructor, Pluralsight

PLURALSIGHT

Quick Review

Modern Data Architecture



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.

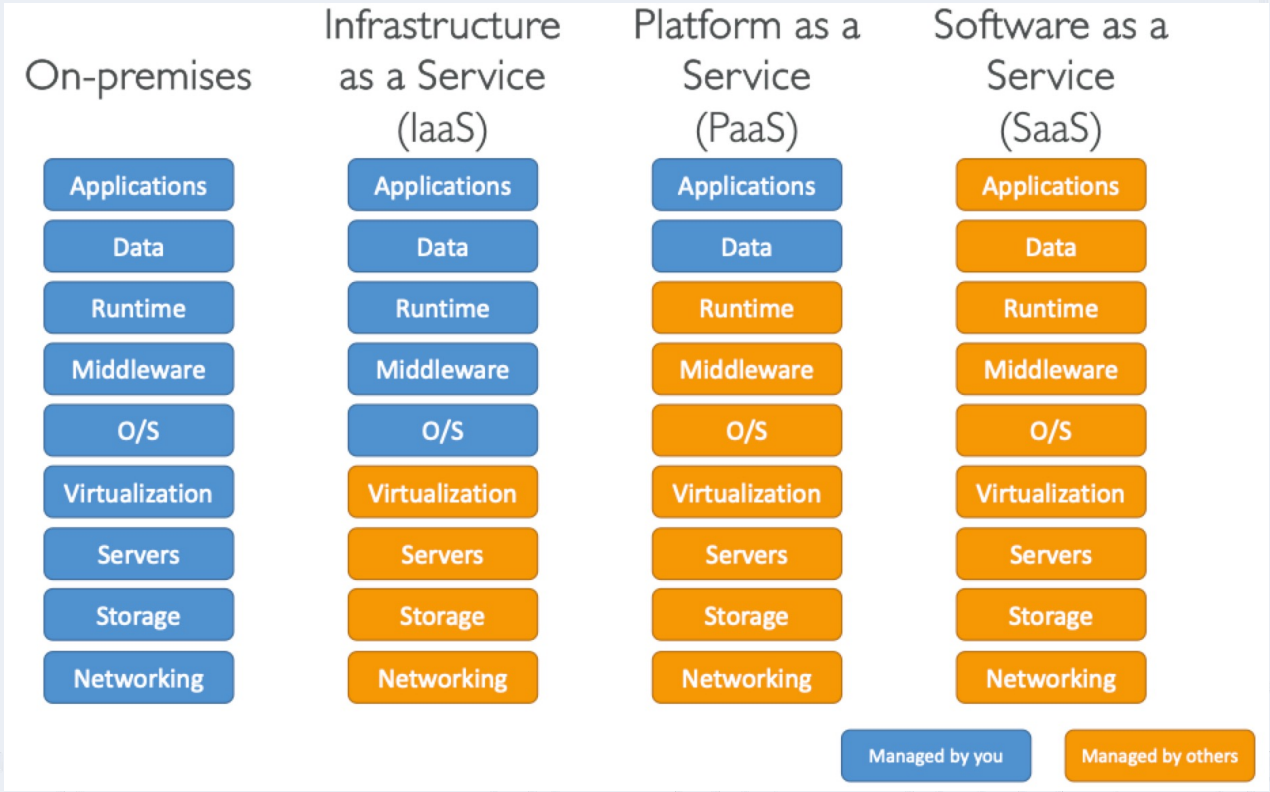
ASSOCIATE

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.**

Types of Cloud Computing



Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) contains the basic building blocks for cloud IT and typically provides access to networking features, computers (virtual or on dedicated hardware), and data storage space.

IaaS provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

Platform as a Service (PaaS)

Platform as a Service (PaaS) removes the need for your organization to manage the underlying infrastructure (usually hardware and operating systems) and allows you to focus on the deployment and management of your applications.

This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

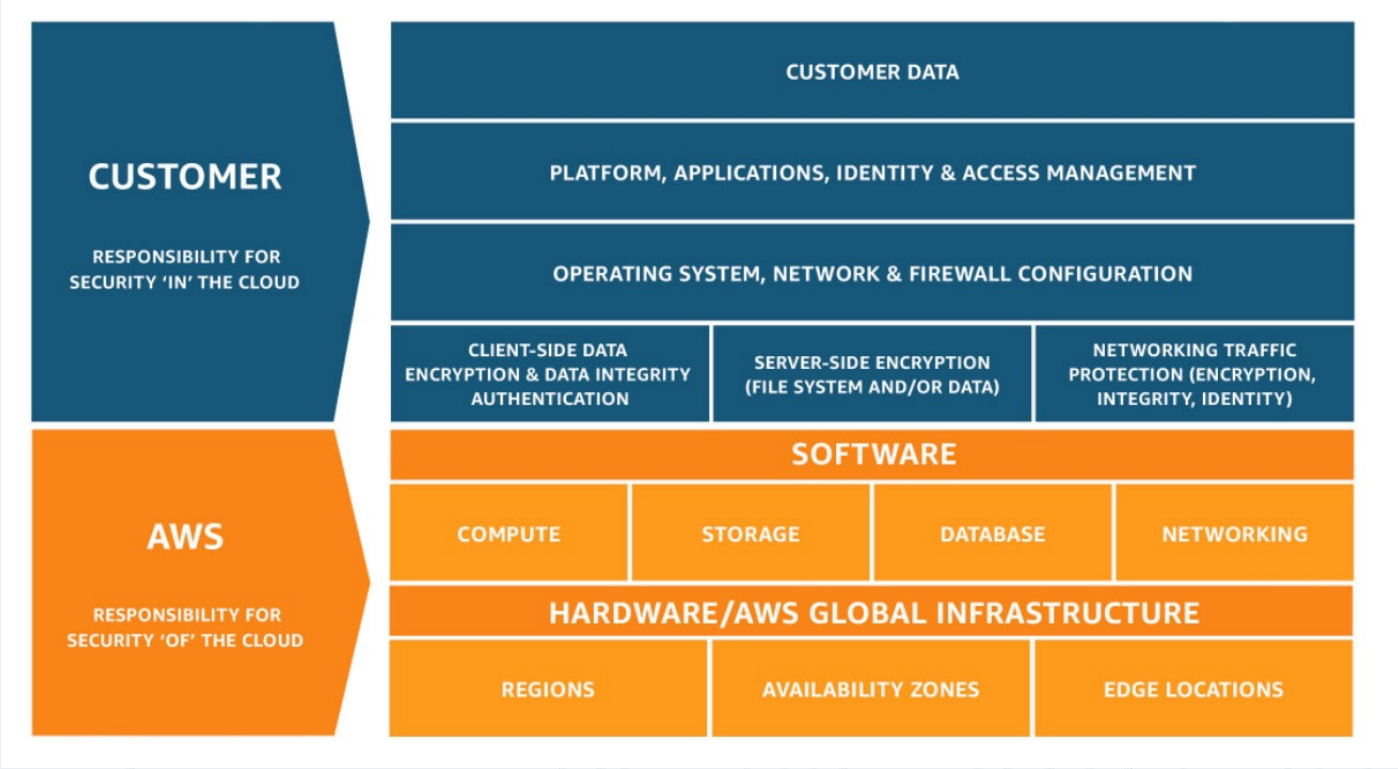
Software as a Service (SaaS)

Software as a Service (SaaS) provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications.

With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software.

A common example of a SaaS application is web-based email which you can use to send and receive email without having to manage feature additions to the email product or maintain the servers and operating systems that the email program is running on.

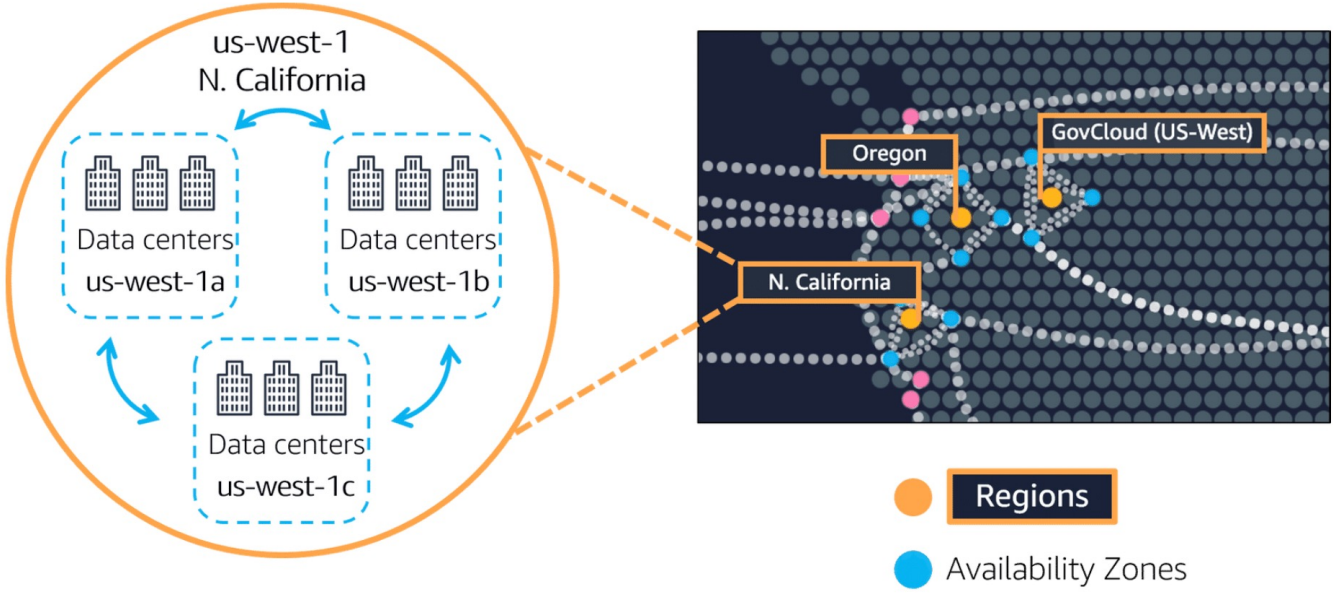
AWS Shared Responsibility Model



AWS Regions and Availability Zones

https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

Availability Zones



Availability Zones

- An Availability Zone is a single data center or a group of data centers within a Region.
- Availability Zones are located tens of miles apart from each other.
- This is close enough to have low latency (the time between when content requested and received) between Availability Zones. However, if a disaster occurs in one part of the Region, they are distant enough to reduce the chance that multiple Availability Zones are affected.

Provisioning AWS Resources

- AWS Management Console
- Command Land Interface (CLI)
- AWS Software Development Kits (SDKs)
- Other tools like AWS CloudFormation

CloudFormation

CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).

- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 instances using this security group
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

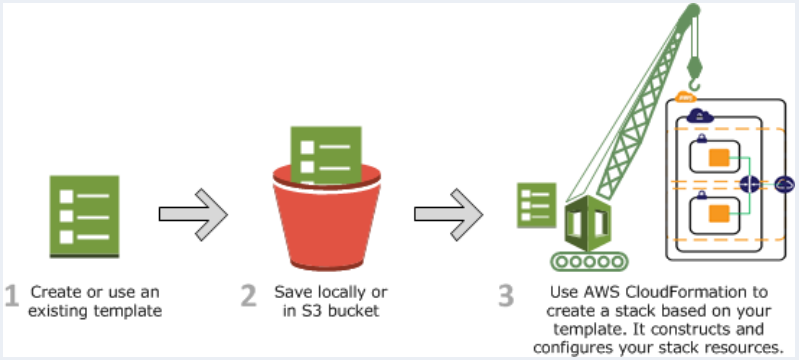
- **Infrastructure as code**
  - No resources are manually created, which is excellent for control
  - Changes to the infrastructure are reviewed through code

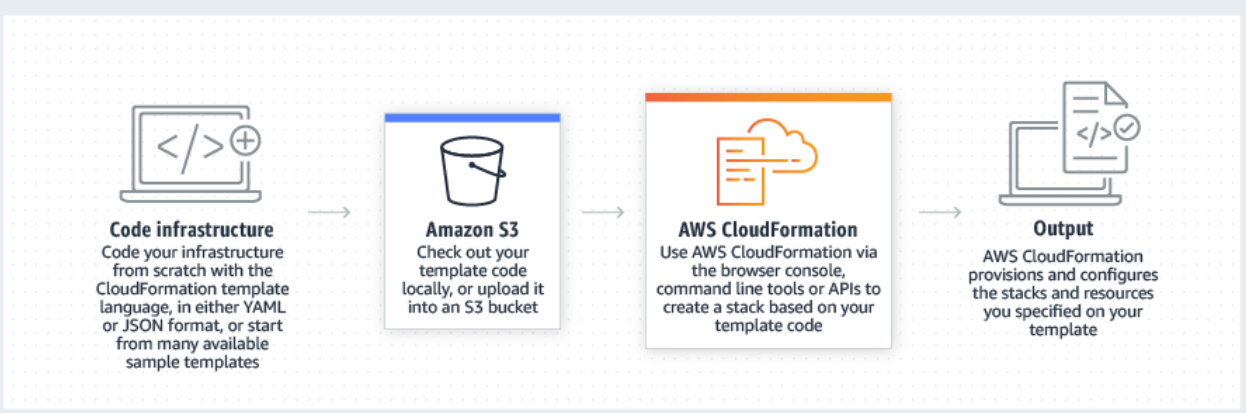
CloudFormation

Cost

- Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
- You can estimate the costs of your resources using the CloudFormation template
- Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

AWS CloudFormation





AWS CloudFormation

With **AWS CloudFormation**, you can treat your **infrastructure as code**. This means that you can build an environment by writing lines of code instead of using the AWS Management Console to individually provision resources.

AWS CloudFormation provisions your resources in a safe, repeatable manner, enabling you to frequently build your infrastructure and applications without having to perform manual actions. It determines the right operations to perform when managing your stack and rolls back changes automatically if it detects errors.

CloudFormation Template

Creating an S3 Bucket Example

JSON

```
{
    "Resources": {
        "HelloBucket": {
            "Type": "AWS::S3::Bucket"
        }
    }
}
```

YAML

```
Resources:
  HelloBucket:
    Type: 'AWS::S3::Bucket'
```

CloudFormation Template

Creating an S3 Bucket Example

JSON

```
{
    "Resources": {
        "HelloBucket": {
            "Type": "AWS::S3::Bucket",
            "Properties": {
                "AccessControl": "PublicRead"
            }
        }
    }
}
```

YAML

```
Resources:
  HelloBucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      AccessControl: PublicRead
```

CloudFormation Template

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "A sample template",
    "Resources": {
        "MyEC2Instance": {
            "Type": "AWS::EC2::Instance",
            "Properties": {
                "ImageId": "ami-0ff8a91507f77f867",
                "InstanceType": "t2.micro",
                "KeyName": "testkey",
                "BlockDeviceMappings": [
                    {
                        "DeviceName": "/dev/sdm",
                        "Ebs": {
                            "VolumeType": "io1",
                            "Iops": 200,
                            "DeleteOnTermination": false,
                            "VolumeSize": 20
                        }
                    }
                ]
            }
        }
    }
}
```

CloudFormation Template

YAML

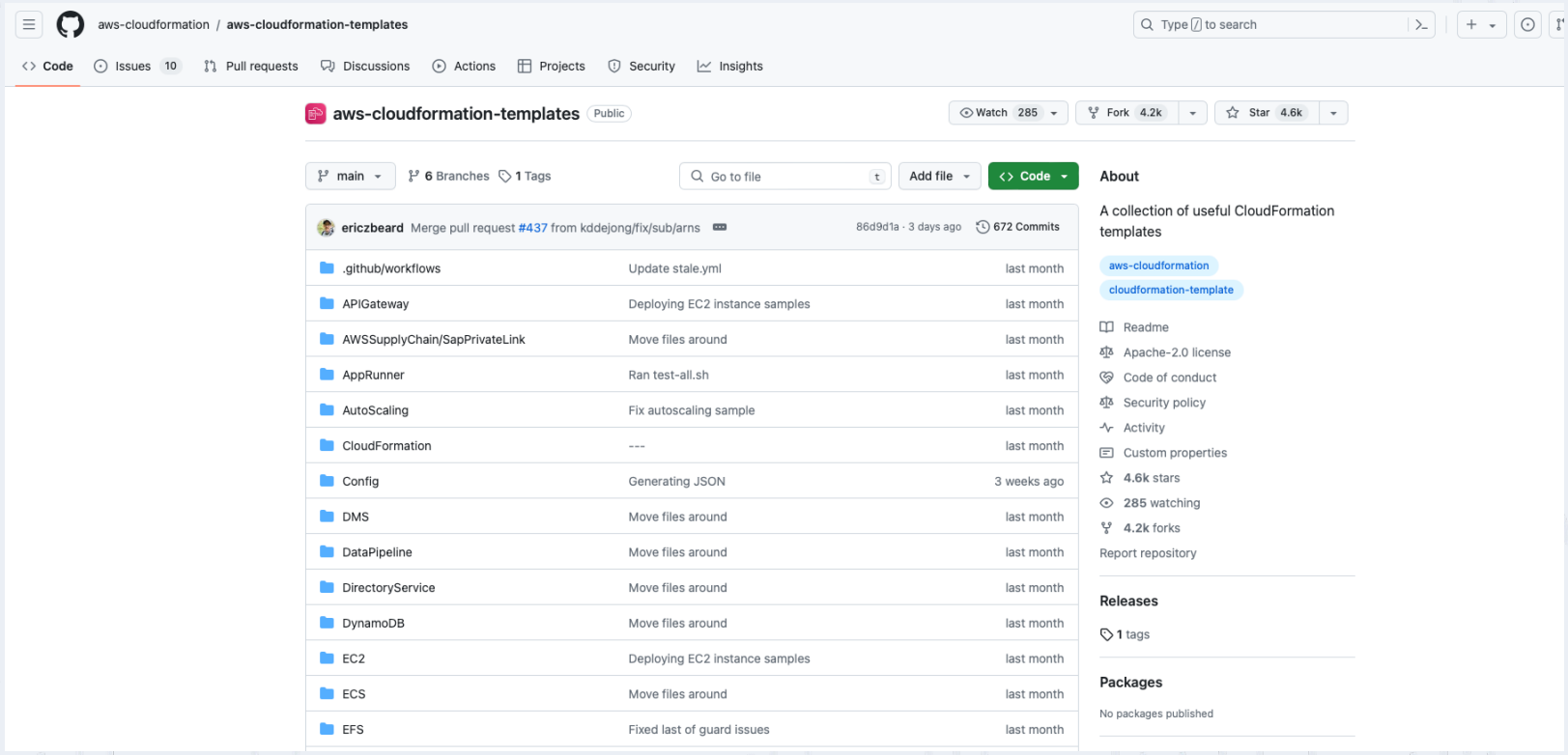
```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template
Resources:
  MyEC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-0ff8a91507f77f867
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        - DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

CloudFormation Stack

When you use CloudFormation, you manage related resources as a single unit called a **stack**. You create, update, and delete a collection of resources by creating, updating, and deleting stacks.

All the resources in a stack are defined by the stack's CloudFormation template.

CloudFormation Repo



Storage

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
- Statis 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

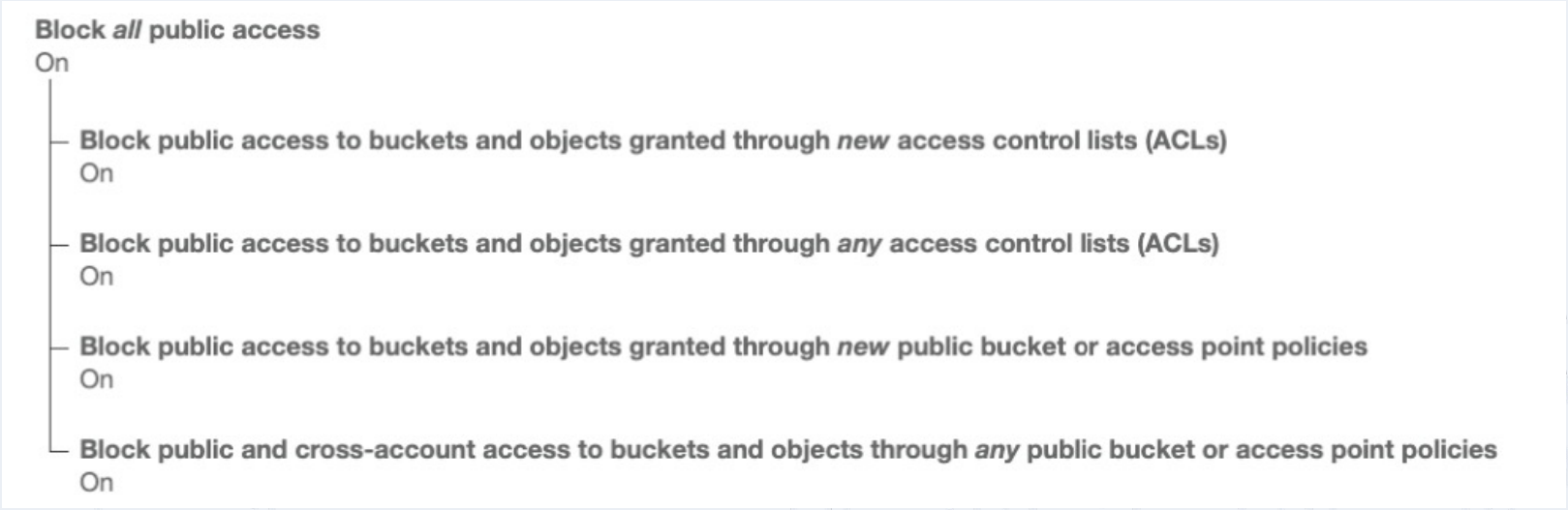
You use S3 Bucket policies to:

- 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/*"
            ]
        }
    ]
}
```

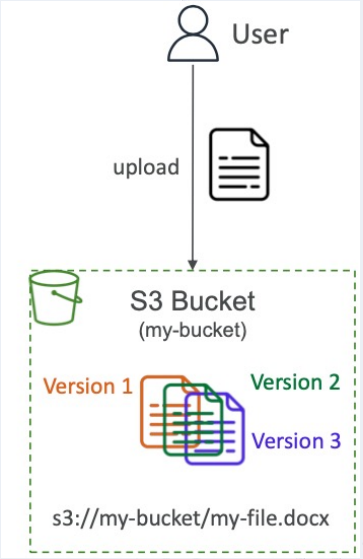
Block Public Access

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



S3 Versioning

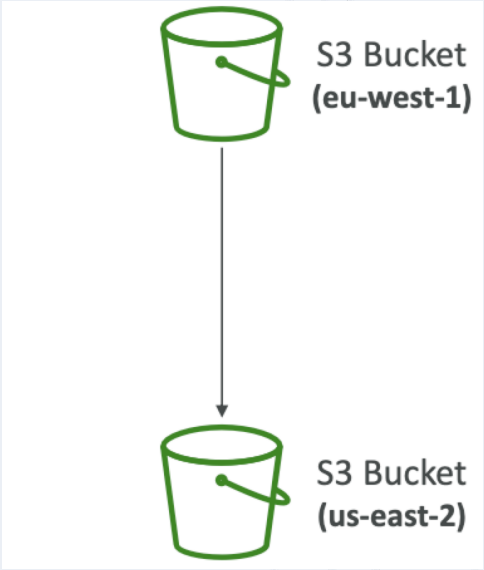
- You can version your files in Amazon S3
- It is enabled at the Bucket Level
- Same key overwrite will change the ”version”
- It is best practices to version you buckets
- A file without versioning will have a version “null”
- Suspending versioning does not delete the previous versions



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

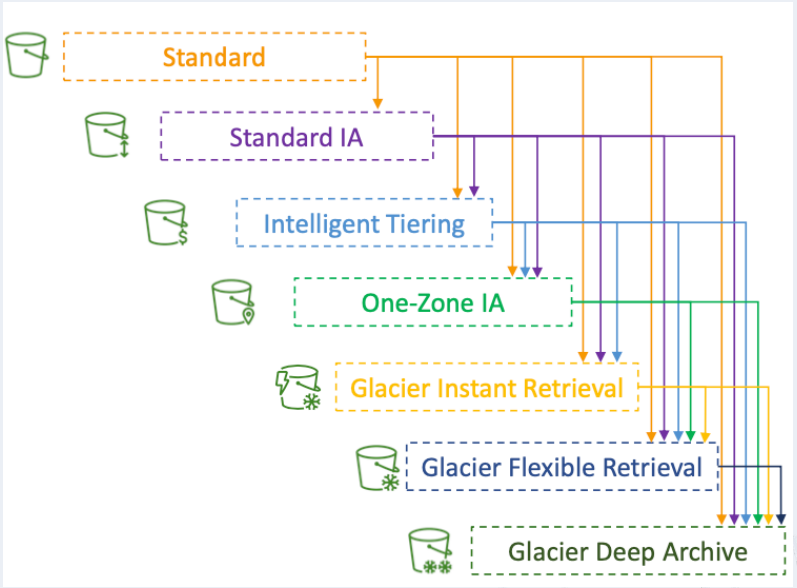
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 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 (pet 1000 objects)		$0.0025					

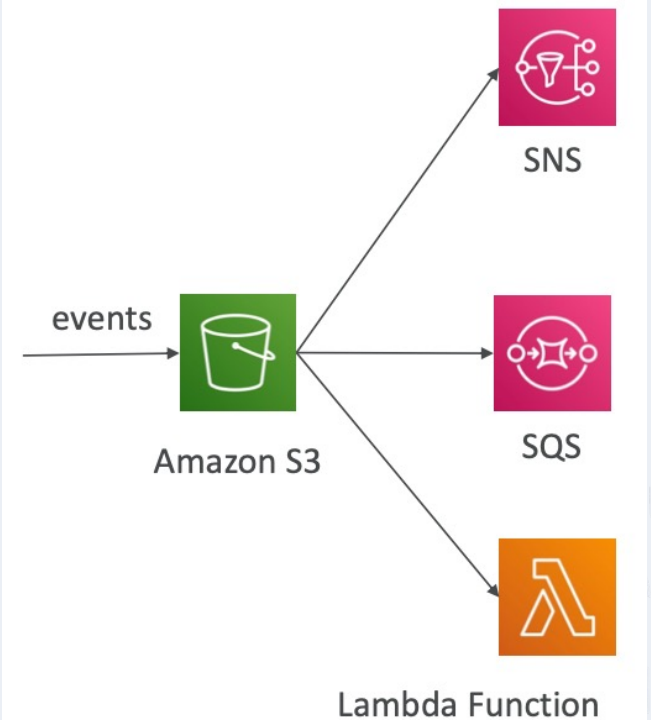
S3 Storage Classes

- 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

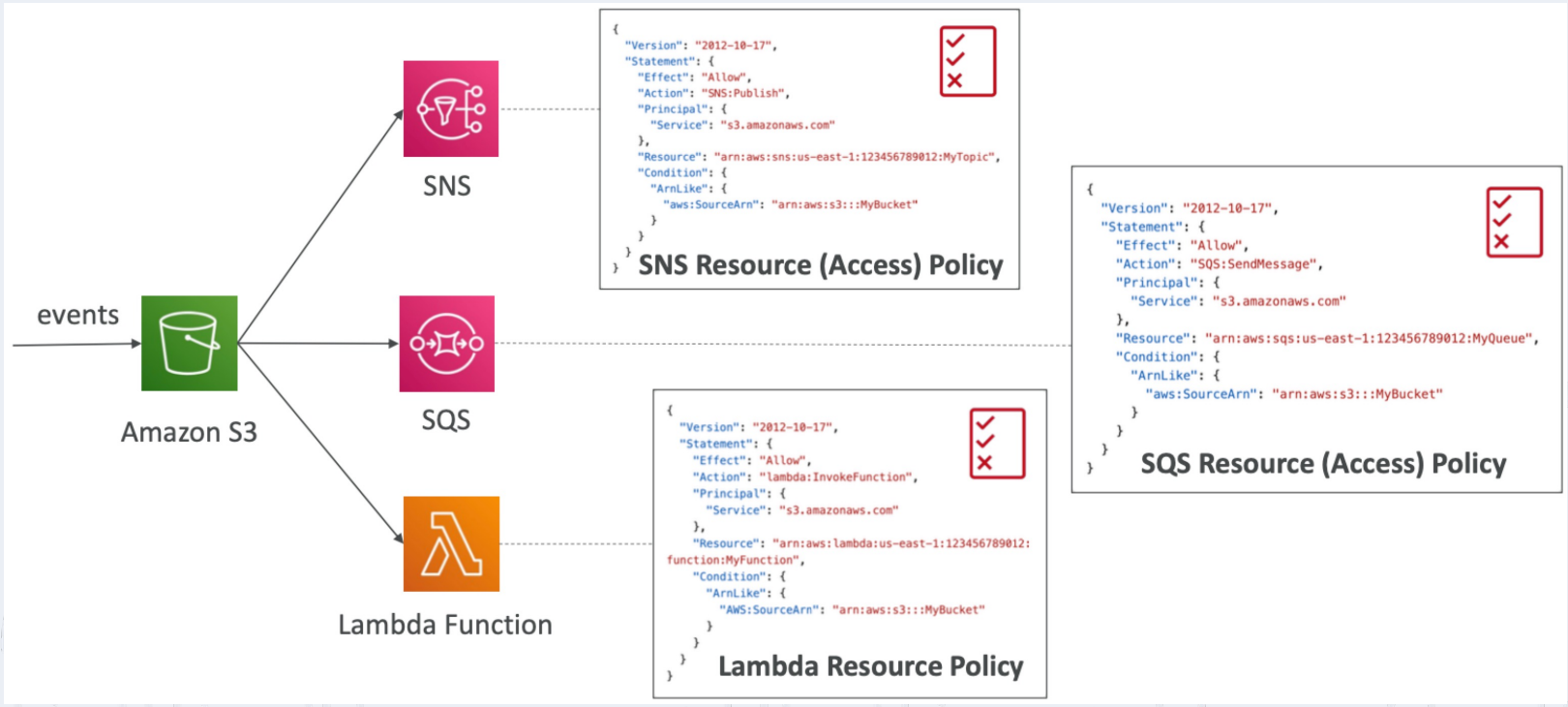


S3 Event Notification

- Event notification on **ObjectCreated**, **ObjecteRemoved**, **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

SQL Databases

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

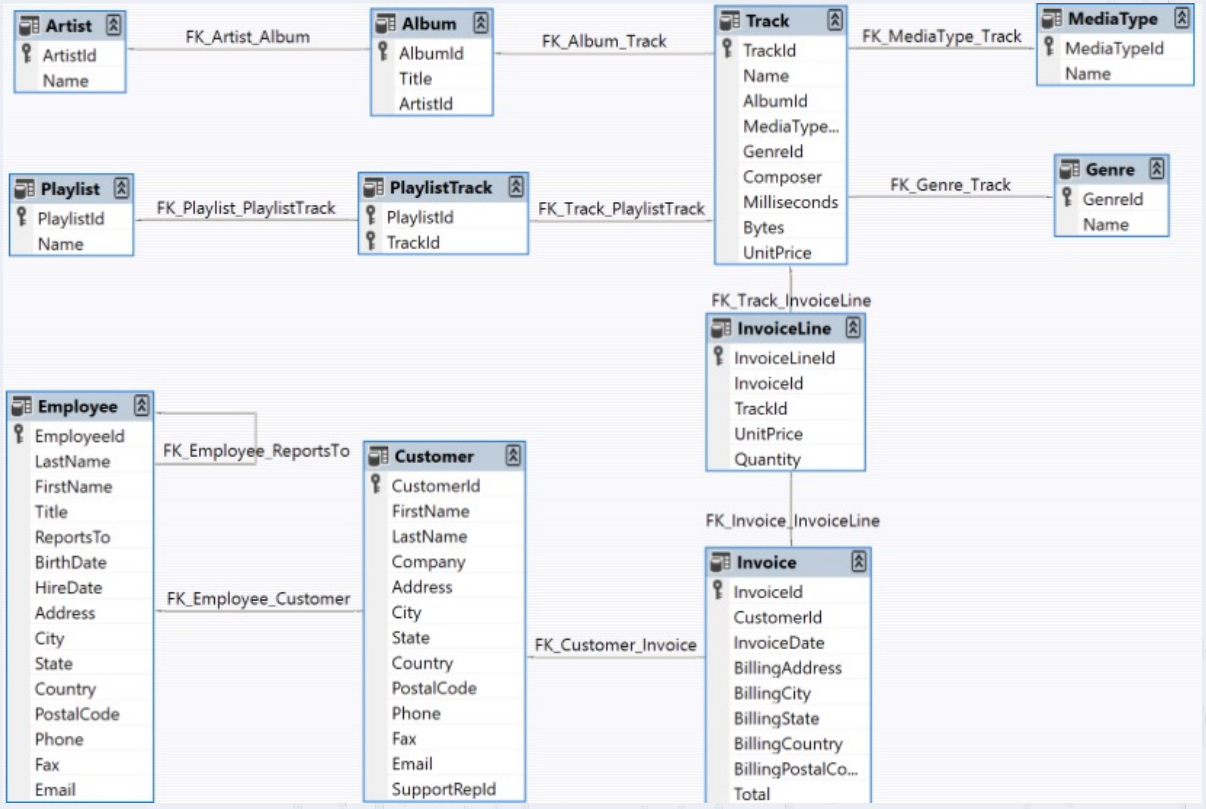
Rank			DBMS	Database Model	Score		
Jun 2024	May 2024	Jun 2023			Jun 2024	May 2024	Jun 2023
1.	1.	1.	Oracle	Relational, Multi-model	1244.08	+7.79	+12.61
2.	2.	2.	MySQL	Relational, Multi-model	1061.34	-22.39	-102.59
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	821.56	-2.73	-108.50
4.	4.	4.	PostgreSQL	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

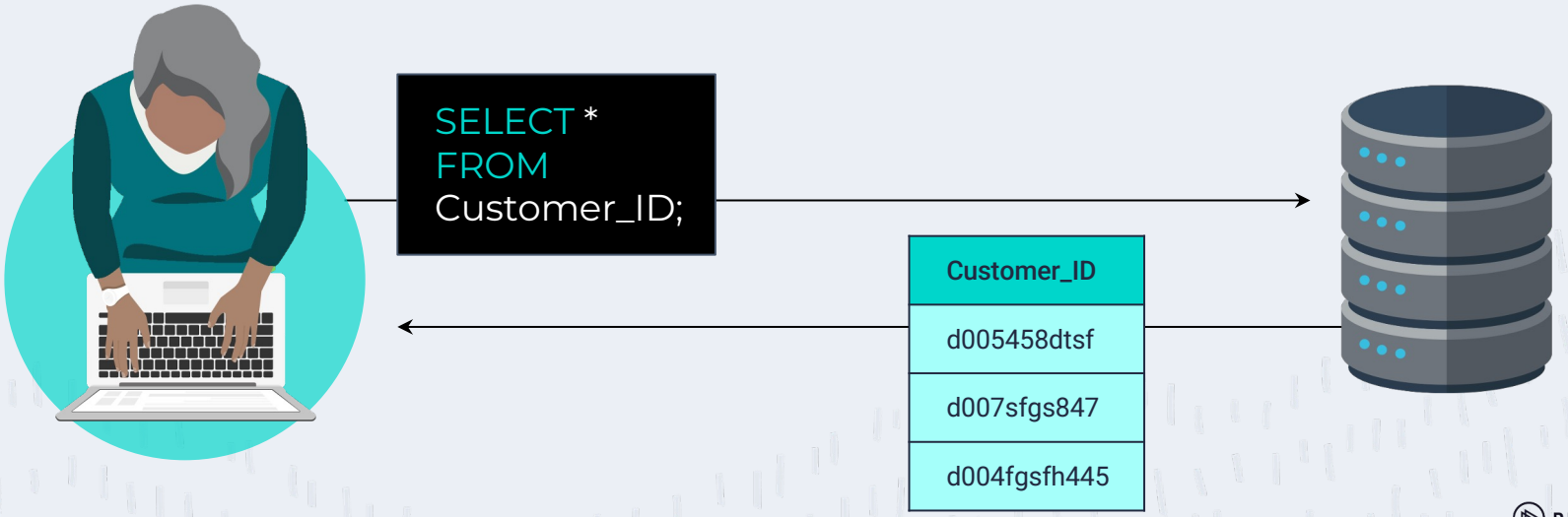
Transactions

- **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 **R**ead **U**pdate **D**elete is a set of operations used with persistent storage.

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

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

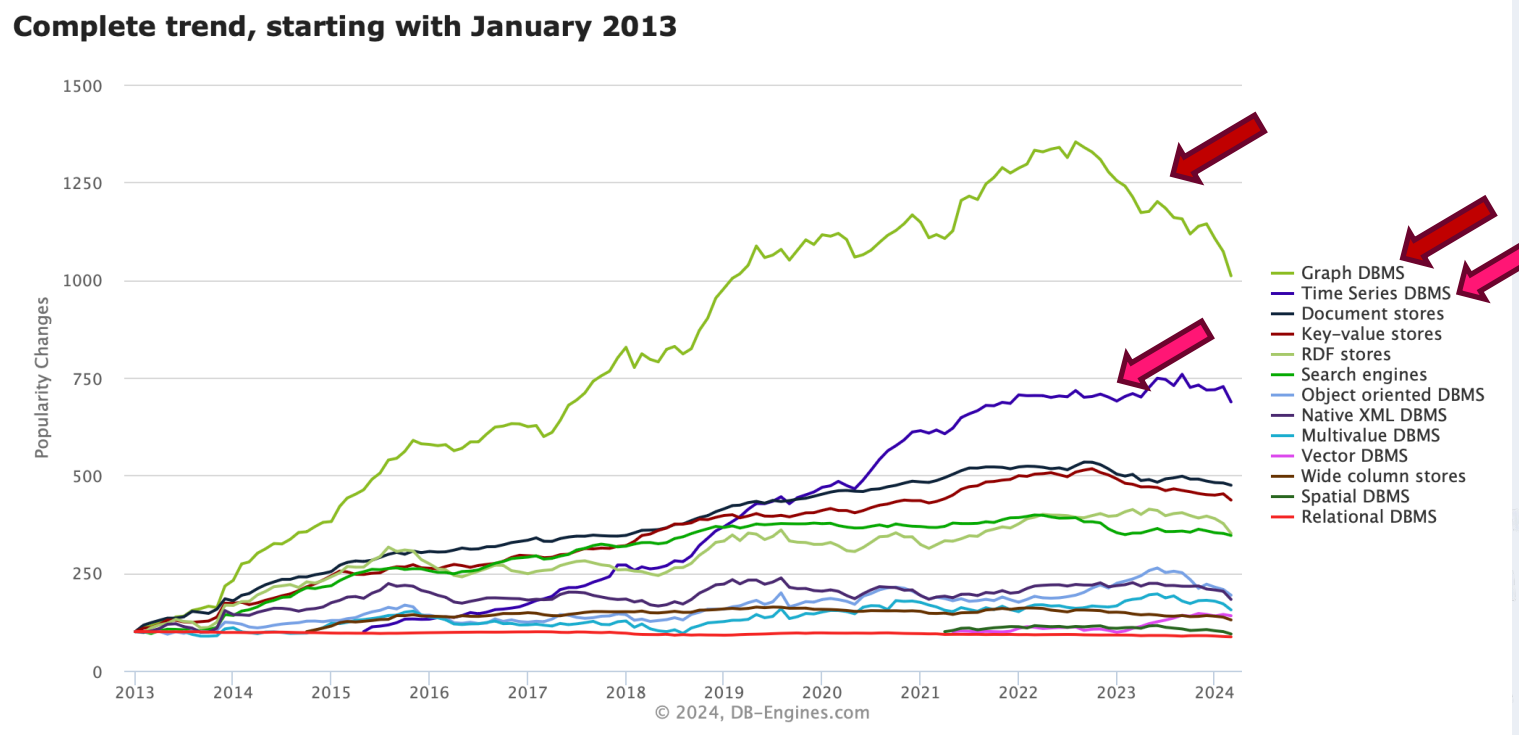
SQL JOINS

`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

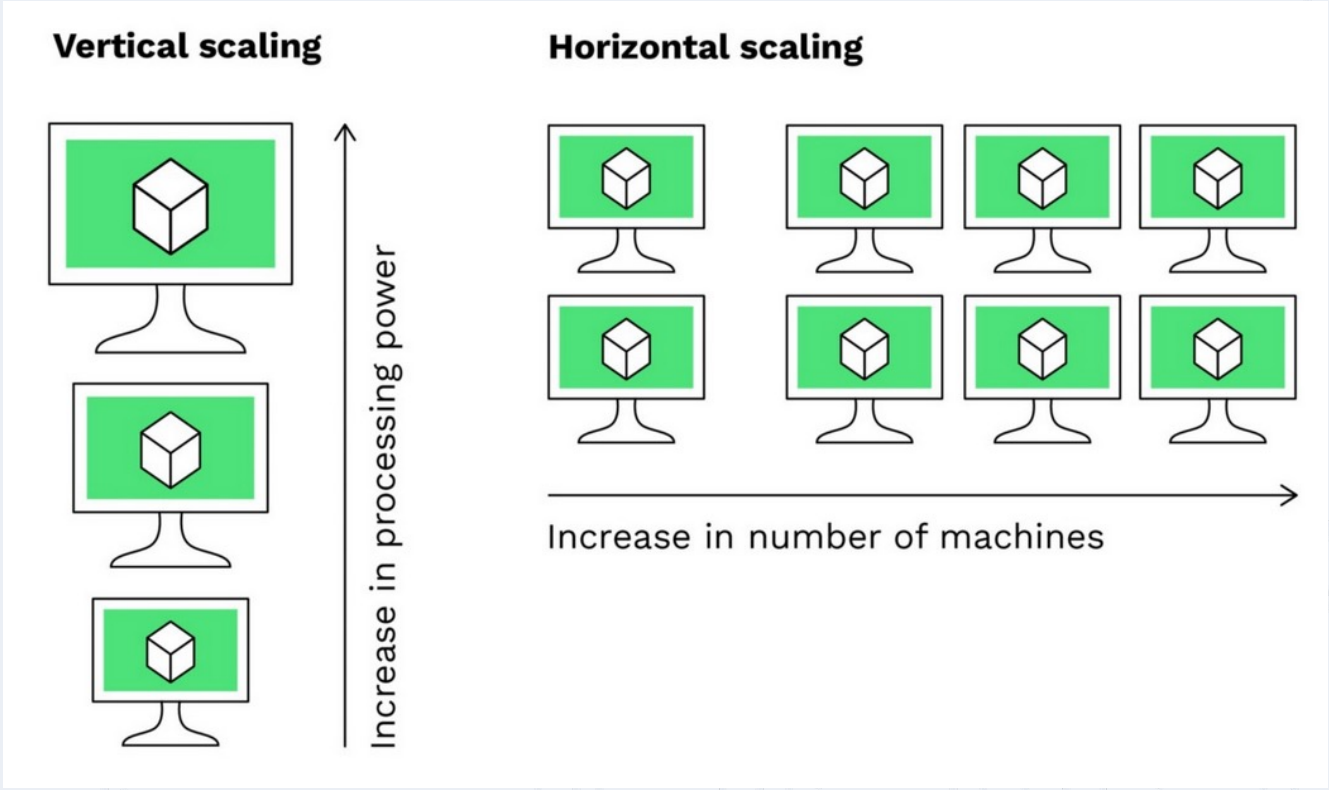


NoSQL Databases

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



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

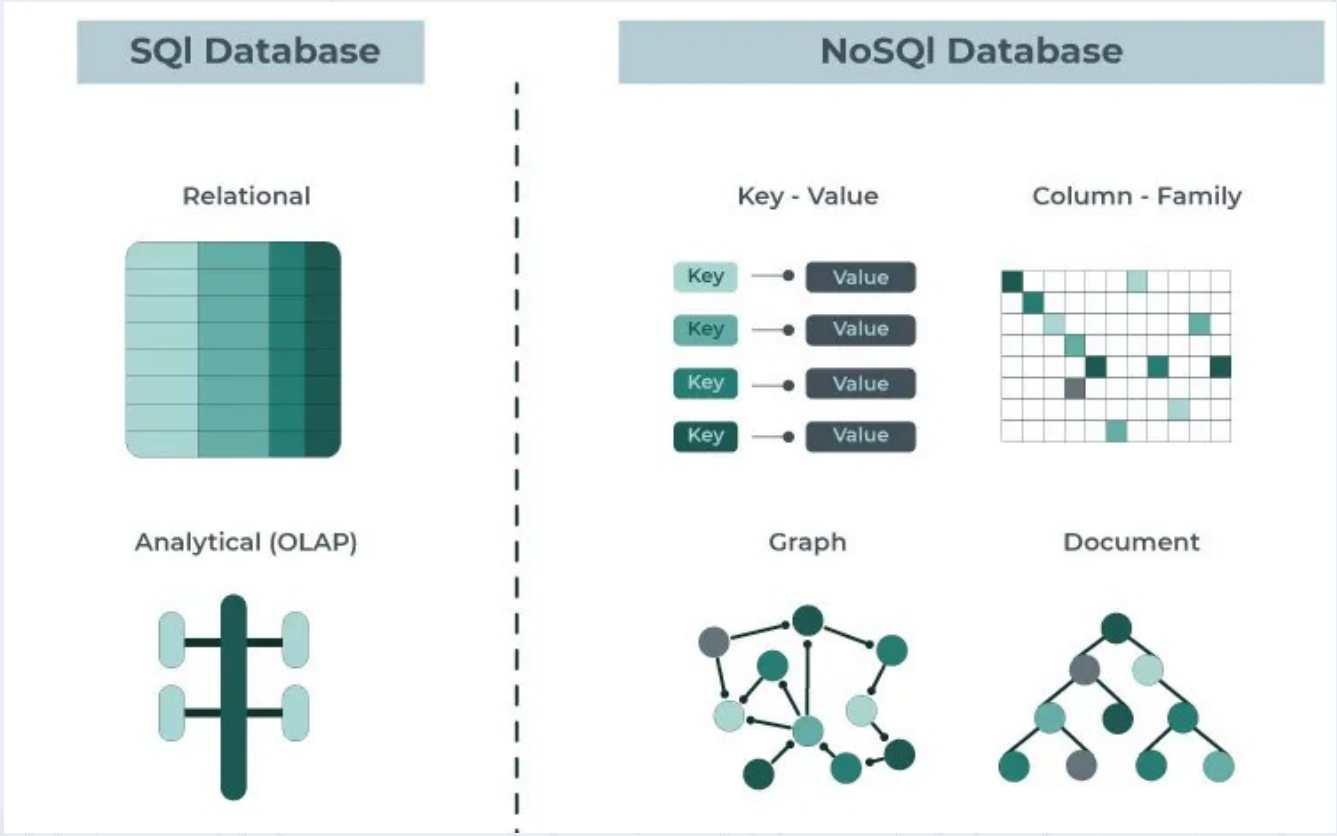
- Store data in columns instead of rows, which allows for efficient querying and analysis of large datasets.
- Organize data into column families, which are sets of columns treated as a single entity.
- Examples include Cassandra and HBase.

Graph Databases

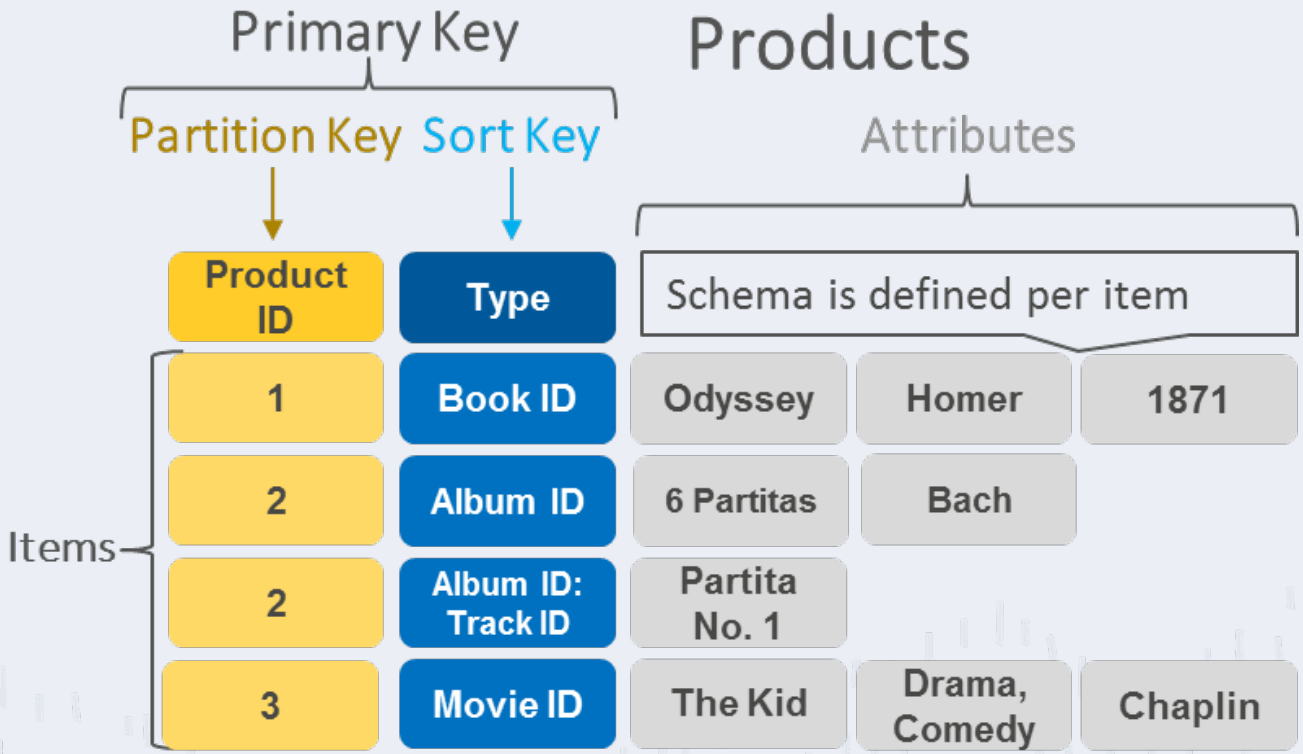
- Focus on the relationships between data, storing it in the form of nodes and edges.
- Designed to handle complex, interconnected data and are well-suited for applications that require traversing and analyzing relationships.
- Examples include Neo4j and Amazon Neptune.

abases:

SQL vs NoSQL



Key Value



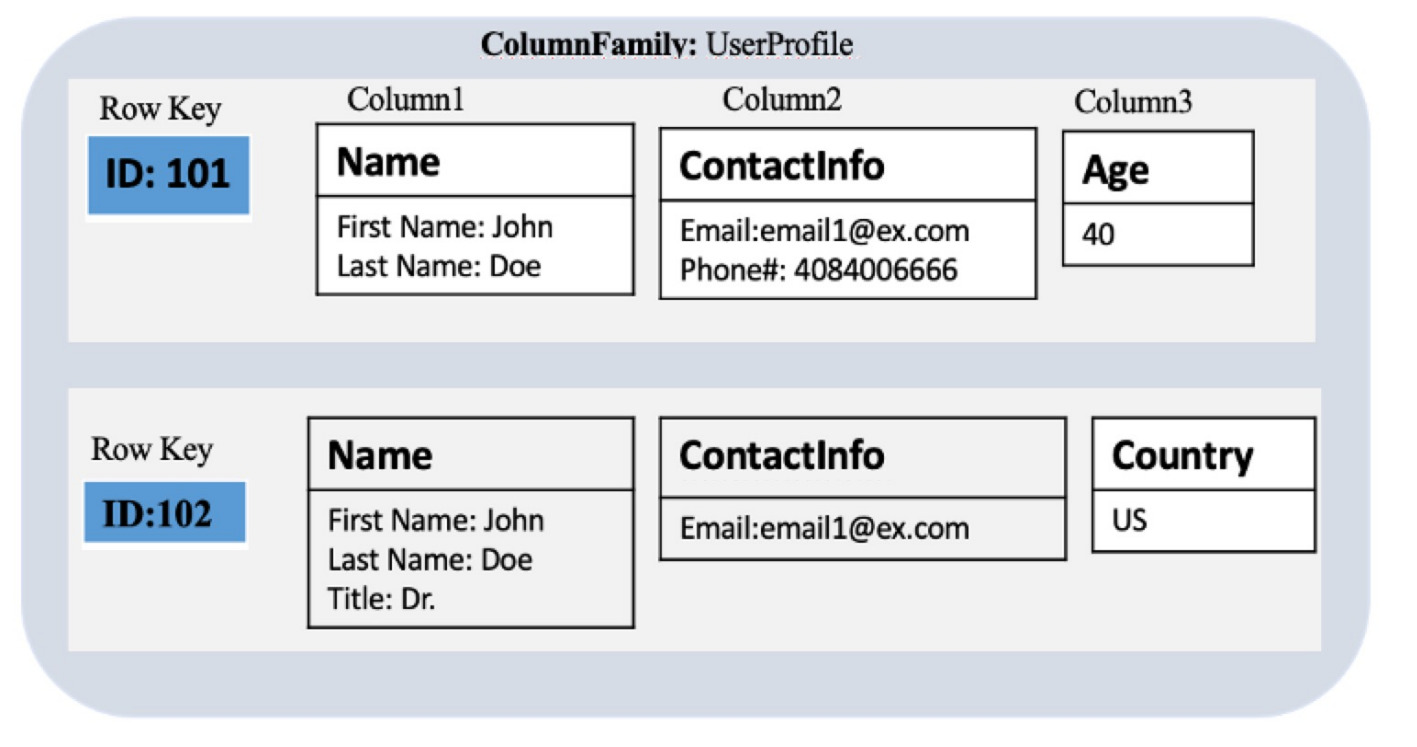
Document

Key	Document

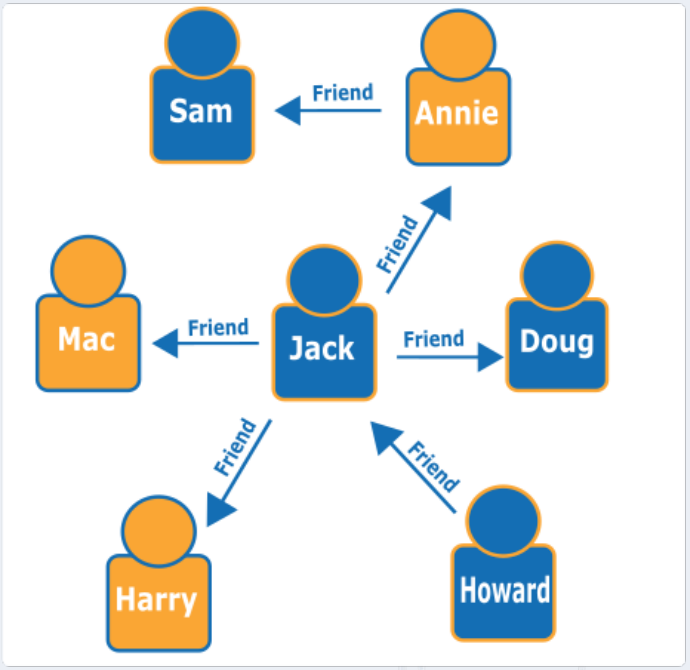
101	{
	"ID": "1001",
	"ItemsOrdered":[
	{
	"ItemID": "1",
	"Quantity": "2",
	"cost": "1000",
	},
	{
	"ItemID": "1001",
	"Quantity": "2",
	"cost": "1000",
	}] ,
	"OrderDate": "05/11/2019"
	}

102	"ID": "1002",
	"ItemsOrdered":[
	{
	"ItemID": "2890",
	"Quantity": "11",
	"cost": "10000",
	}]
	"OrderDate": "05/11/2019"
	}

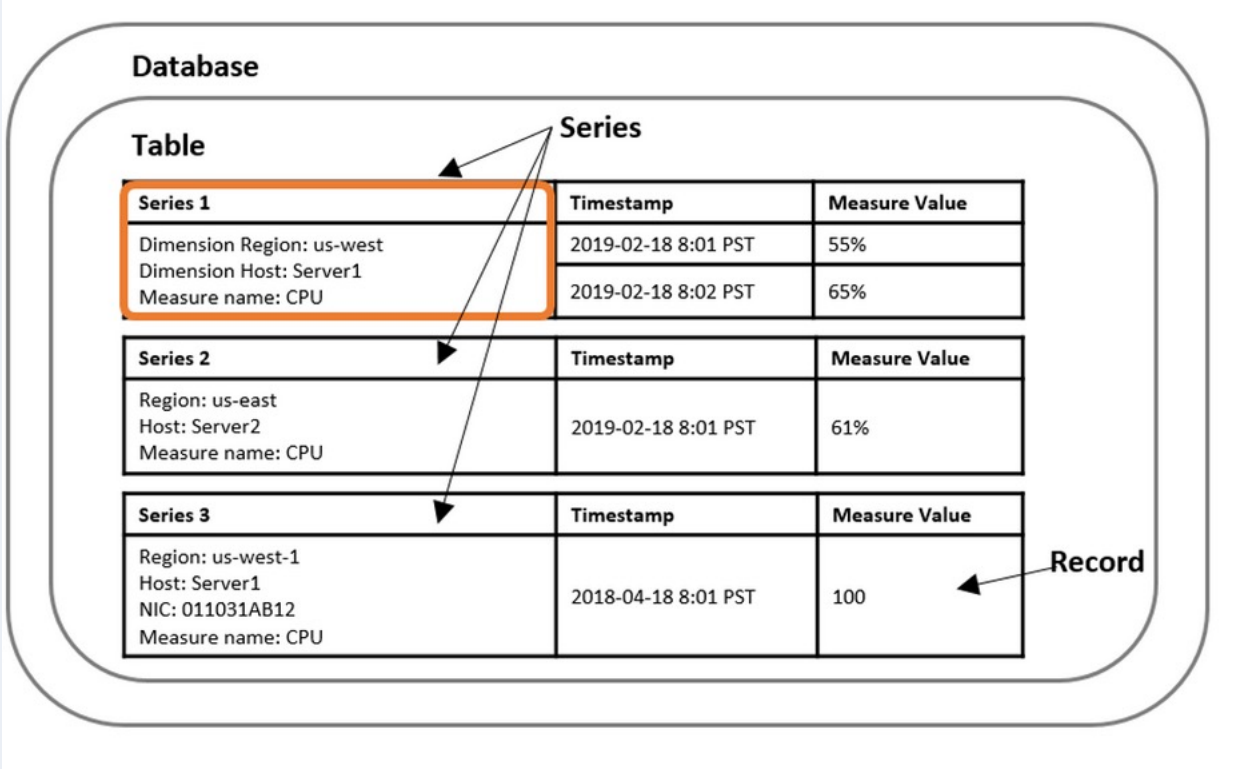
Wide Columns



Graph



Time Series



AWS Databases

AWS Database Services

Database	Use cases	AWS services
Relational	Traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), e-commerce	•Amazon Aurora — Designed for unparalleled high performance and availability at global scale with full MySQL and PostgreSQL compatibility •Amazon RDS — Set up, operate, and scale a relational database in the cloud with just a few clicks •Amazon Redshift — Accelerate your time to insights with fast, easy, and secure cloud data warehousing at scale
Key-value	High-traffic web applications, e-commerce systems, gaming applications	•Amazon DynamoDB — Fast, flexible NoSQL database service for single-digit millisecond performance at any scale
In-memory	Caching, session management, gaming leaderboards, geospatial applications	•Amazon ElastiCache — Unlock microsecond latency and scale with in-memory caching •Amazon MemoryDB for Redis — Redis-compatible, durable, in-memory database service for ultra-fast performance
Document	Content management, catalogs, user profiles	•Amazon DocumentDB (with MongoDB compatibility) — Scale JSON workloads with ease using a fully managed document database service
Wide column	High-scale industrial apps for equipment maintenance, fleet management, and route optimization	•Amazon Keyspaces — A scalable, highly available, and managed Apache Cassandra–compatible database service
Graph	Fraud detection, social networking, recommendation engines	•Amazon Neptune — Build and run graph applications with highly connected datasets
Time series	Internet of Things (IoT) applications, DevOps, industrial telemetry	•Amazon Timestream — Fast, scalable, serverless time series database
Ledger	Systems of record, supply chain, registrations, banking transactions	•Amazon Ledger Database Service (QLDB) — Maintain an immutable, cryptographically verifiable log of data changes

Amazon Aurora

Compatibility and Performance:

- MySQL and PostgreSQL compatible relational database engine.
- Combines high-end commercial database speed and availability with the simplicity and cost-effectiveness of open source databases.
- Up to 5 times faster than standard MySQL and 3 times faster than standard PostgreSQL.

Scalability and Availability:

- Distributed, fault-tolerant, self-healing storage system that auto-scales up to 128TB per database instance.
- High performance and availability with up to 15 low-latency read replicas.
- Point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones (AZs).

Amazon Aurora

Zero-ETL Integration:

- Aurora MySQL zero-ETL integration with Amazon Redshift available in public preview.
- Enables near real-time analytics and machine learning on data stored in Aurora MySQL-Compatible Edition.
- Transactional data available in Amazon Redshift within seconds without complex data pipelines.

Amazon Aurora

Amazon DynamoDB

Performance and Scalability:

- Key-value and document database with single-digit millisecond performance at any scale.
- Can handle more than 10 trillion requests per day and support peaks of more than 20 million requests per second.

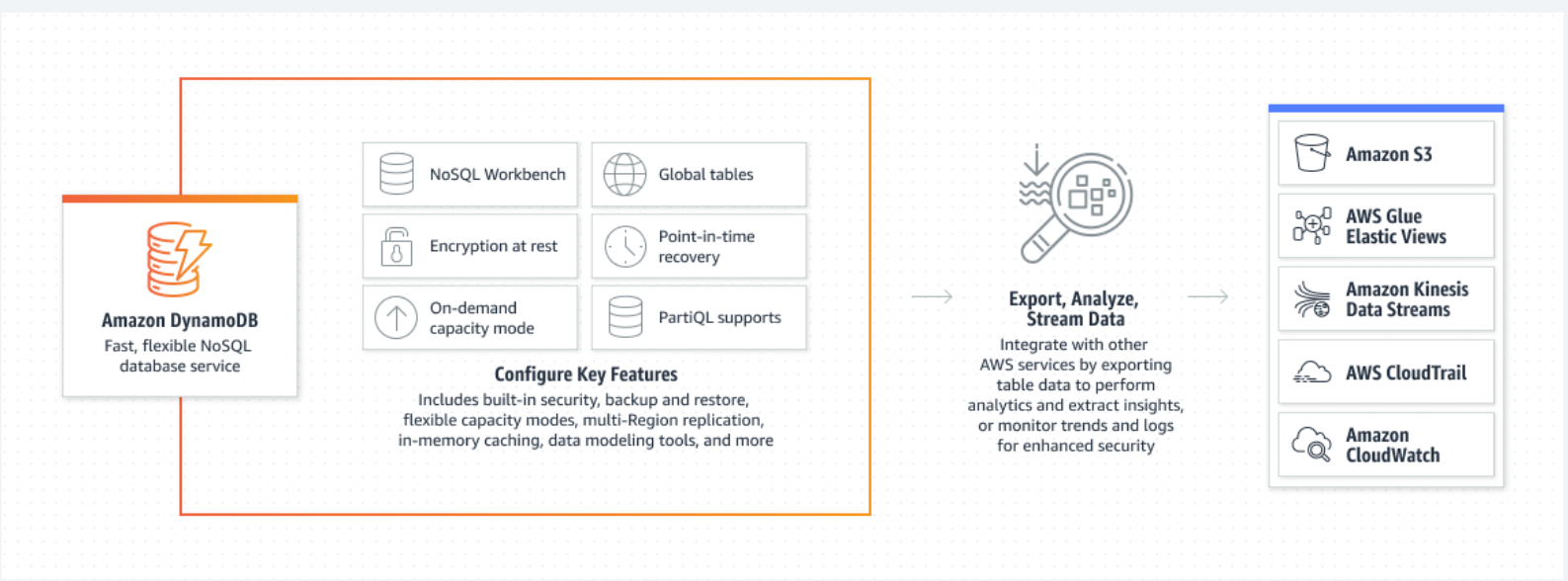
Fully Managed Service:

- Multi-Region database with built-in security, backup and restore, and in-memory caching.
- Suitable for internet-scale applications.

Use Cases:

- Trusted by fast-growing businesses like Lyft, Airbnb, and Redfin.
- Used by enterprises such as Samsung, Toyota, and Capital One for mission-critical workloads.
- Ideal for mobile, web, gaming, ad tech, Internet of Things (IoT), and other applications needing low-latency data access.

Amazon DynamoDB



Amazon ElastiCache

Service Overview:

- Simplifies deployment, operation, and scaling of in-memory caches in the cloud.
- Improves web application performance by retrieving data from fast, managed in-memory caches.

ElastiCache Serverless:

- Simplifies cache management and scales automatically.
- Creates a highly available and scalable cache in under a minute.
- Eliminates the need for planning, provisioning, and managing cache cluster capacity.
- Automatically stores data across multiple Availability Zones (AZs) with a 99.99% SLA.
- Pay only for data stored and compute consumed, with no upfront commitments or additional costs.

Amazon ElastiCache

In-Memory Caching Engines Supported:

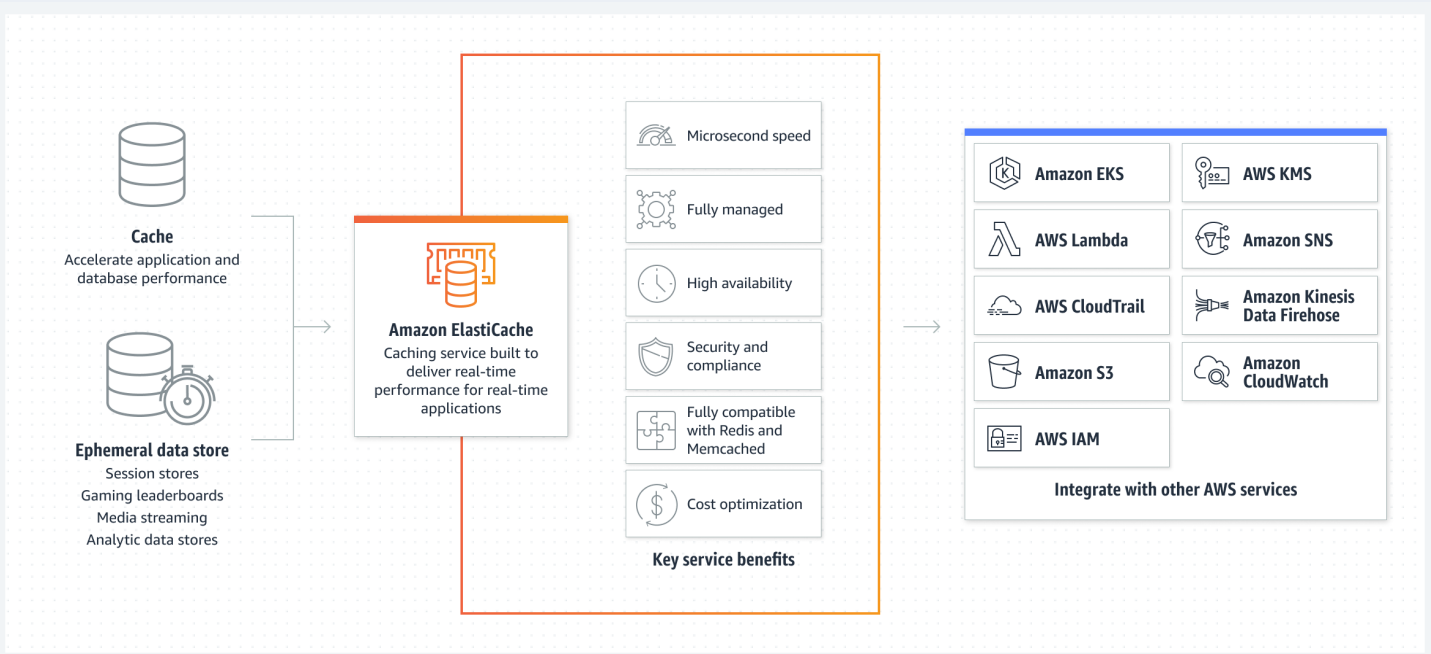
Redis:

- Fast, open-source in-memory key-value data store.
- Functions as a database, cache, message broker, and queue.
- Amazon ElastiCache for Redis is fully managed, scalable, and secure, supporting single-node and up to 15-shard clusters, scaling to 3.55 TiB of in-memory data.
- Ideal for high-performance use cases like web apps, mobile apps, gaming, ad-tech, and IoT.

Memcached:

- Widely adopted memory object caching system.
- Amazon ElastiCache for Memcached is protocol-compliant, supporting popular Memcached tools seamlessly.

Amazon ElastiCache



Amazon Keyspaces (for Apache Cassandra)

Service Overview:

- Scalable, highly available, and managed Apache Cassandra–compatible database service.
- Allows running Cassandra workloads on AWS using existing Cassandra application code and developer tools.

Management and Maintenance:

- No need to provision, patch, or manage servers.
- No installation, maintenance, or operation of software required.

Serverless Architecture:

- Pay only for the resources used.
- Automatically scales tables up and down in response to application traffic.

Amazon Keyspaces (for Apache Cassandra)

Performance and Scalability:

- Supports thousands of requests per second with virtually unlimited throughput and storage.

Security and Backup:

- Data is encrypted by default.
- Enables continuous backup of table data using point-in-time recovery.

Enterprise Features:

- Provides performance, elasticity, and features necessary for business-critical Cassandra workloads at scale.

Amazon Relational Database Services (RDS)

Service Overview:

- Simplifies the setup, operation, and scaling of relational databases in the cloud.
- Provides cost-efficient and resizable capacity.
- Automates administration tasks like hardware provisioning, database setup, patching, and backups

Performance and Focus:

- Allows focusing on applications by providing fast performance, high availability, security, and compatibility.

Database Instance Types:

- Available on several instance types optimized for memory, performance, or I/O.

Amazon Keyspaces (for Apache Cassandra)

Database Engines:

- Supports six familiar database engines: MySQL, MariaDB, PostgreSQL, Oracle Database, Microsoft SQL Server, and Amazon RDS on AWS Outposts.

Migration and Replication:

- Facilitates easy migration or replication of existing databases to Amazon RDS using AWS Database Migration Service.

Amazon RDS

Hosted Relational Database

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

RDS databases offer full **ACID** compliance

- Atomicity
- Consistency
- Isolation
- Durability

Amazon RDS

Amazon QLDB

Service Overview:

- Fully managed ledger database with a transparent, immutable, and cryptographically verifiable transaction log.
- Owned by a central trusted authority, tracking each application data change and maintaining a complete verifiable history.

Use Cases:

- Ideal for recording economic and financial activities in organizations.
- Examples include tracking banking transactions, verifying insurance claim data lineage, and tracing supply chain item movements.

Amazon QLDB

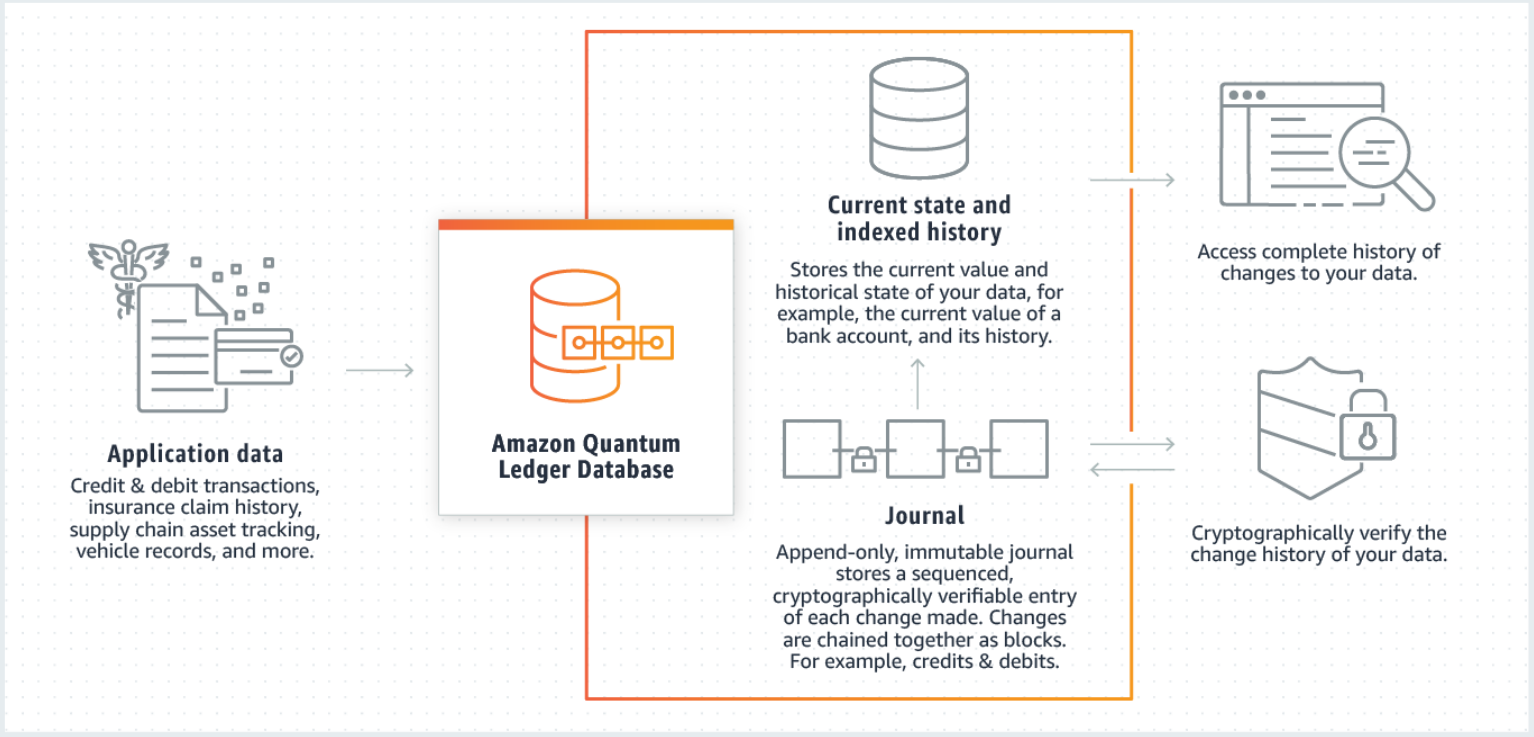
Comparison with Blockchain:

- Simplifies implementation compared to blockchain frameworks like Hyperledger Fabric and Ethereum.
- No need for setting up and managing a blockchain network with multiple nodes.

Technical Features:

- Immutable transactional log (journal) tracks each data change.
- Familiar SQL-like API, flexible document data model, and full transaction support.
- Serverless architecture that automatically scales with application demands.
- No servers to manage and no read/write limits to configure.

Amazon QLDB



Amazon Timestream

Service Overview:

- Fast, scalable, fully managed time series database service.
- Designed for IoT and operational applications.
- Efficiently stores and analyzes trillions of events per day at 1/10th the cost of relational databases.

Data Characteristics:

- Time series data measures changes over time.
- Data typically arrives in time order, is append-only, and queries are over time intervals.

Purpose-Built for Time Series Data:

- Optimized for storing and processing data by time intervals, unlike relational databases.
- Suitable for log data for DevOps, sensor data for IoT applications, and industrial telemetry data.

Amazon Timestream

Adaptive Query Processing:

- Understands data location and format as it grows over time, simplifying and speeding up analysis.

Automated Data Management:

- Automates rollups, retention, tiering, and compression of data.
- Manages data at the lowest possible cost.

Serverless Architecture:

- No servers to manage.
- Handles server provisioning, software patching, setup, configuration, and data retention/tiering.
- Frees users to focus on building applications.

Amazon Timestream

Amazon Neptune

Service Overview:

- Fast, reliable, fully-managed graph database service.
- Optimized for building and running applications with highly connected datasets.

Graph Database Engine:

- Purpose-built for high performance, storing billions of relationships.
- Queries the graph with millisecond latency.

Supported Graph Models and Query Languages:

- Supports Property Graph and W3C's RDF graph models.
- Compatible with Apache TinkerPop Gremlin and SPARQL query languages.
- Efficiently navigates highly connected datasets.

Amazon Neptune

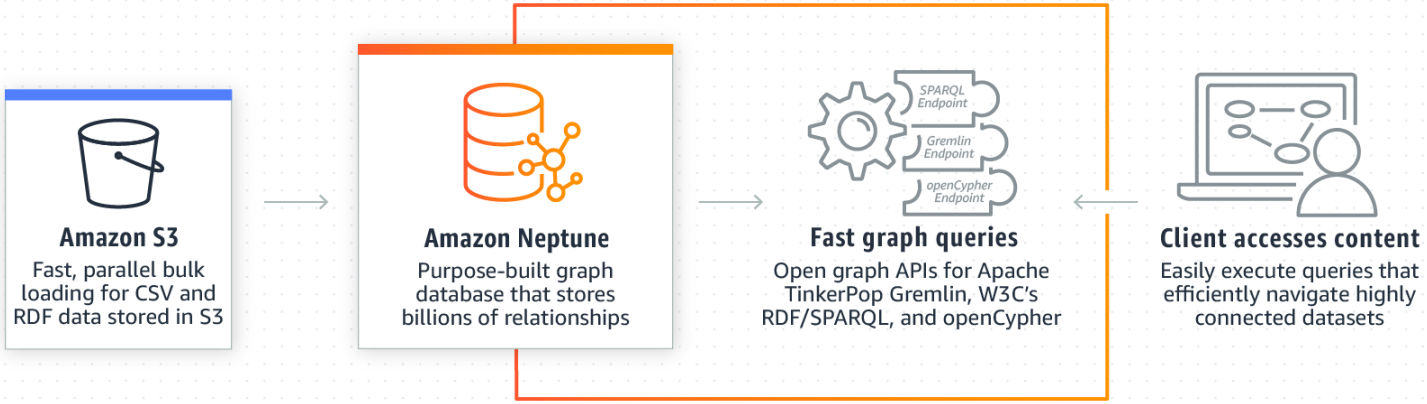
Use Cases:

- Suitable for recommendation engines, fraud detection, knowledge graphs, drug discovery, and network security.

Neptune Analytics:

- Analytics engine for quickly analyzing large volumes of graph data.
- Retrieves insights and trends from data stored in Amazon S3 or Neptune database.
- Uses built-in algorithms, vector search, and in-memory computing to run queries on tens of billions of relationships in seconds.

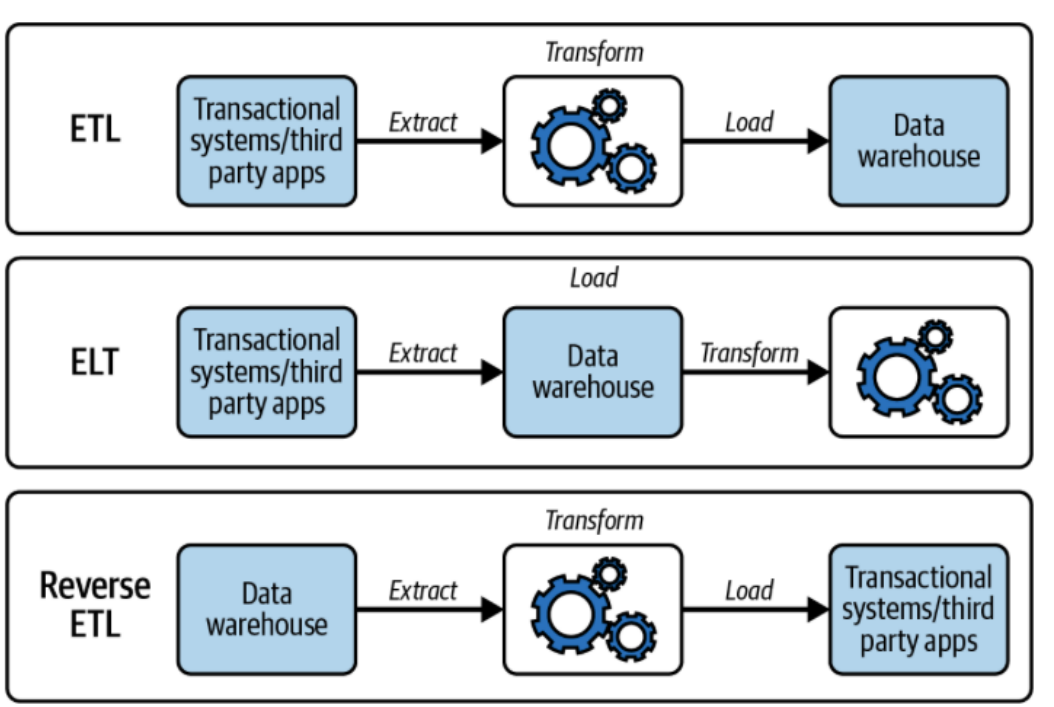
Amazon Neptune



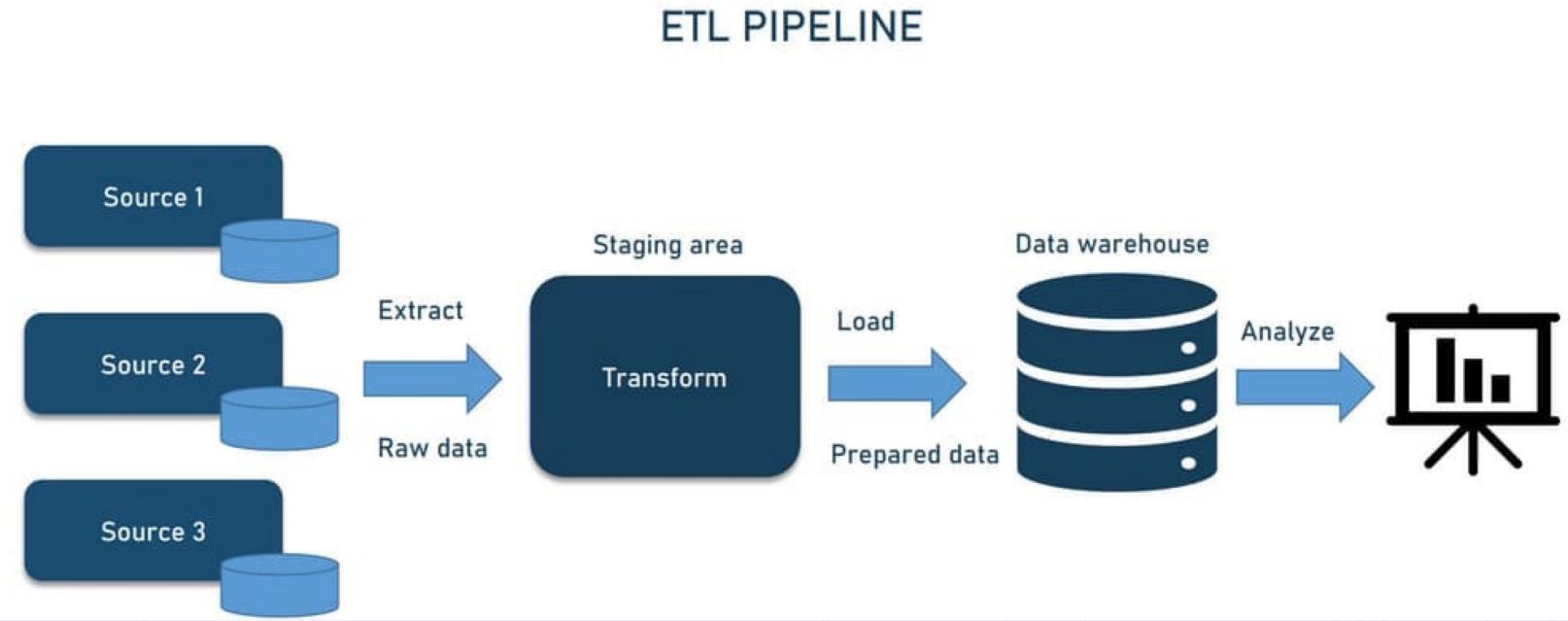
AWS Analytics Services

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

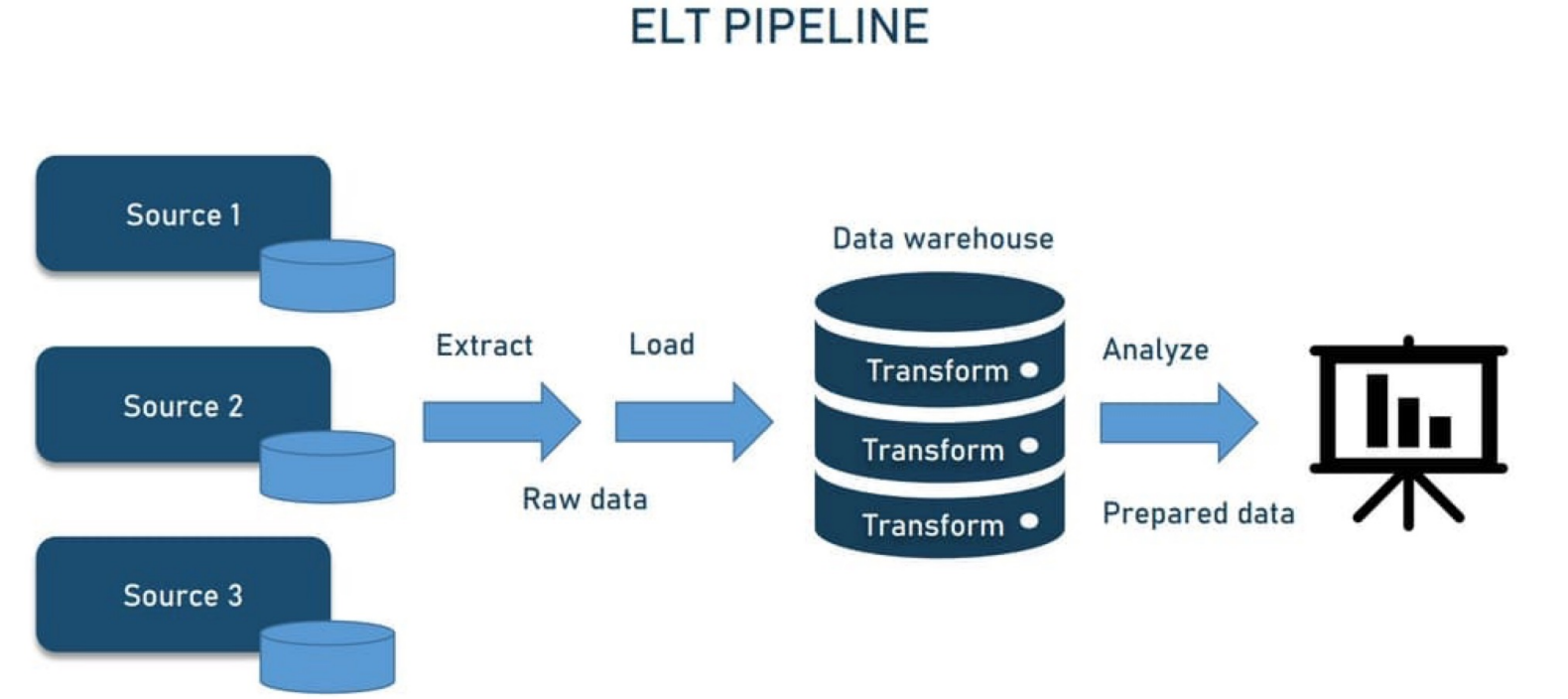
ETL, ELT, and Reverse ETL



Data Engineering



Data Engineering



AWS Glue

AWS Glue

Serverless discovery and definition of table definitions and schema

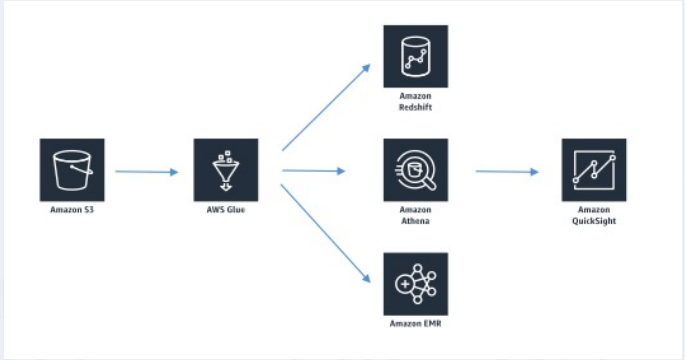
- S3 “data lakes”
- RDS
- Redshift
- DynamoDB
- Most other SQL databases

Custom ETL jobs

- Trigger-driven, on a schedule, or on demand
- Fully managed

Glue Crawler / Data Catalog

- Glue crawler scans data in S3, creates schema
- Can run periodically
- Populates the Glue Data Catalog
  - Stores only table definition
  - Original data stays in S3
- Once cataloged, you can treat your unstructured data like it's structured
  - Redshift Spectrum
  - Athena
  - EMR
  - Quicksight



Glue ETL

- Automatic code generation (example Python)
- Encryption
  - Server-side (at rest)
  - SSL (in transit)
- Can be event-driven
- Can provision additional “DPU’s” (data processing units) to increase performance of underlying Spark jobs
  - Enabling job metrics can help you understand the maximum capacity in DPU’s you need
- Errors reported to CloudWatch
  - Could tie into SNS for notification

Glue ETL

- Transform data, Clean Data, Enrich Data (before doing
- analysis)
    - Generate ETL code in Python or Scala, you can modify the code
    - Can provide your own Spark or PySpark scripts
    - Target can be S3, JDBC (RDS, Redshift), or in Glue Data Catalog
- Fully managed, cost effective, pay only for the resources consumed
- Jobs are run on a serverless Spark platform
- Glue Scheduler to schedule the jobs
- Glue Triggers to automate job runs based on “events”

Glue ETL - Transformations

- Bundled Transformations:

  DropFields, DropNullFields – remove (null) fields
  Filter – specify a function to filter records
  Join – to enrich data
  Map - add fields, delete fields, perform external lookups

- Machine Learning Transformations:

  FindMatches ML: identify duplicate or matching records in your dataset, even when the records do not have a common unique identifier and no fields match exactly.

- Format conversions: CSV, JSON, Avro, Parquet, ORC, XML
- Apache Spark transformations (example: K-Means)

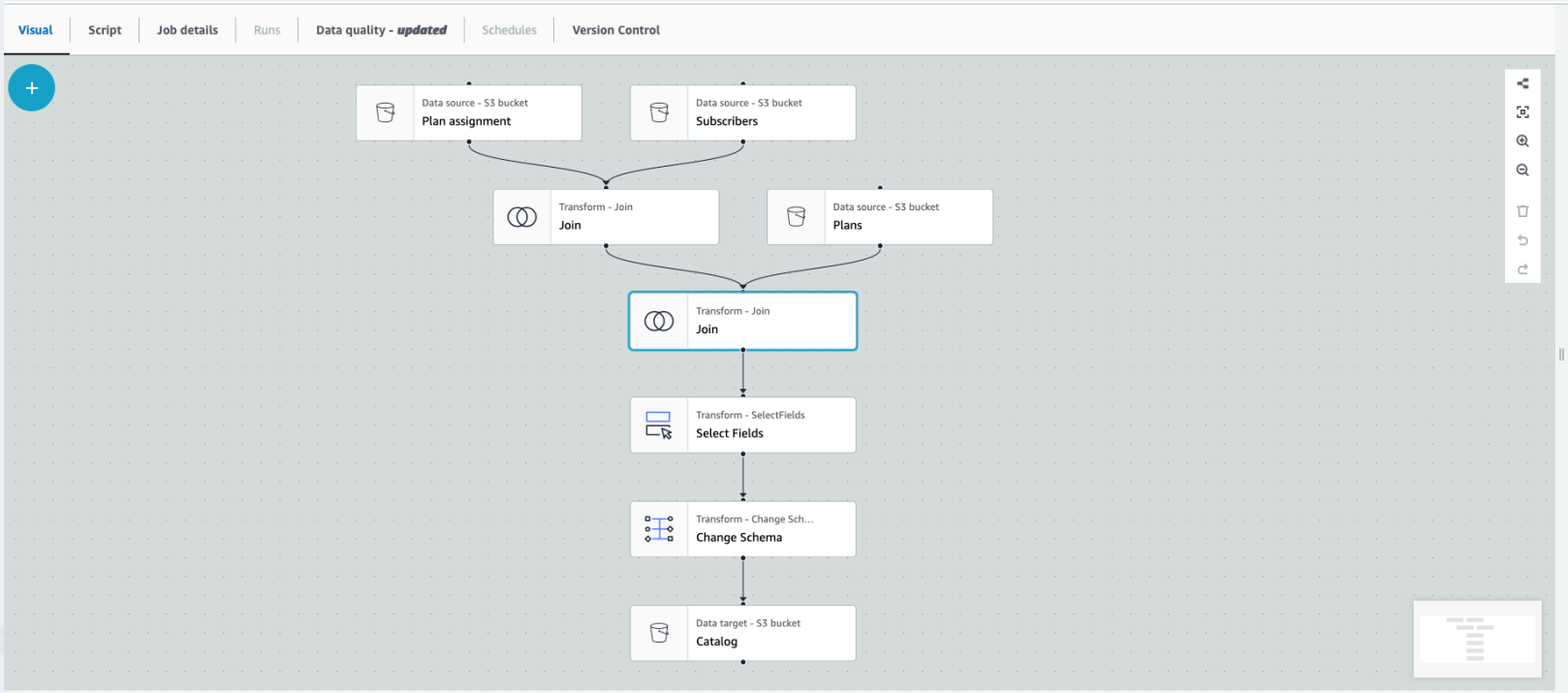
  Can convert between Spark DataFrame and Glue DynamicFrame

AWS Glue Studio

AWS Glue Script
Use when: You need to write custom code for complex data transformations or require more control over the ETL process.

Key features:
- Supports custom code in Java, Scala, or Python.
- Allows for more complex data transformations.
- Provides more control over the ETL process.

AWS Glue Studio (Visual ETL)



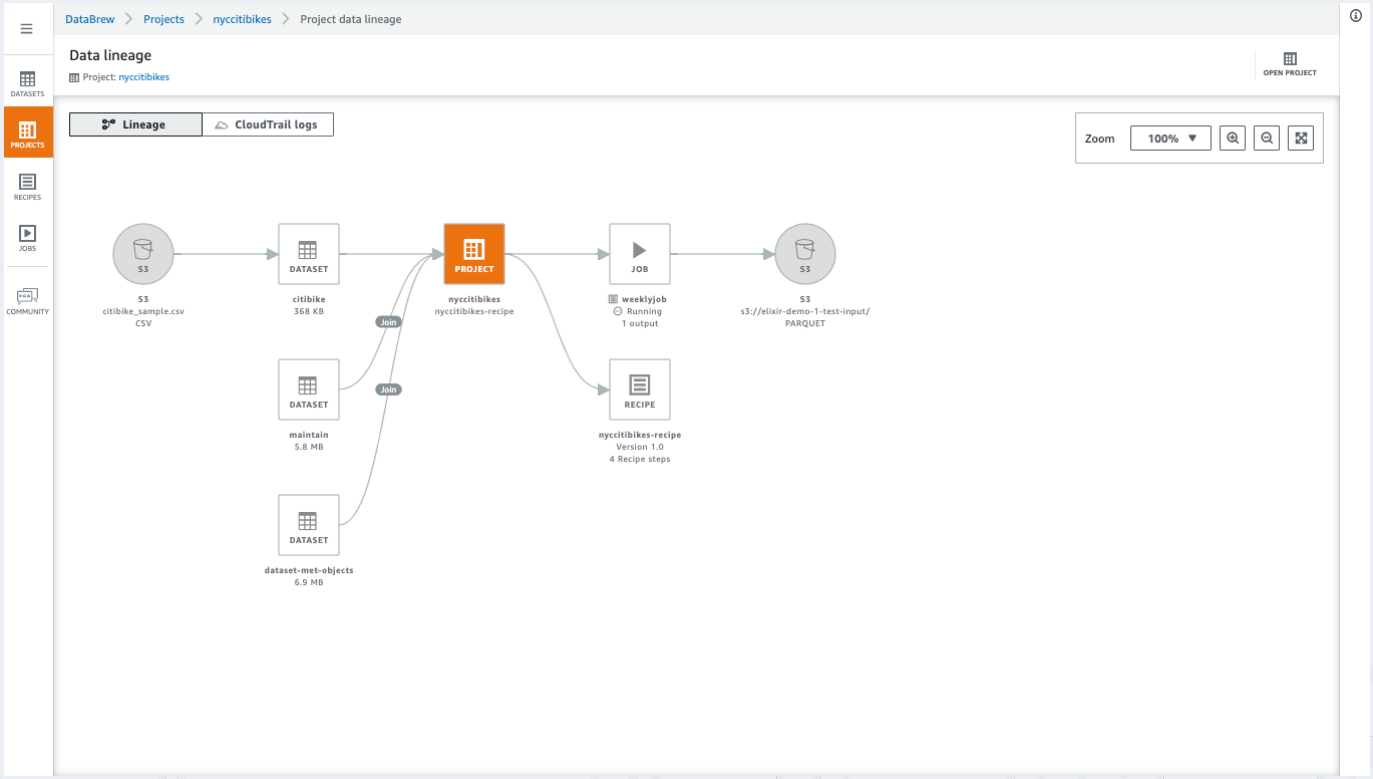
AWS Glue Studio

AWS Glue Studio
Use when: You need to create, run, and monitor ETL jobs visually and want a centralized view of the overall ETL workflow.

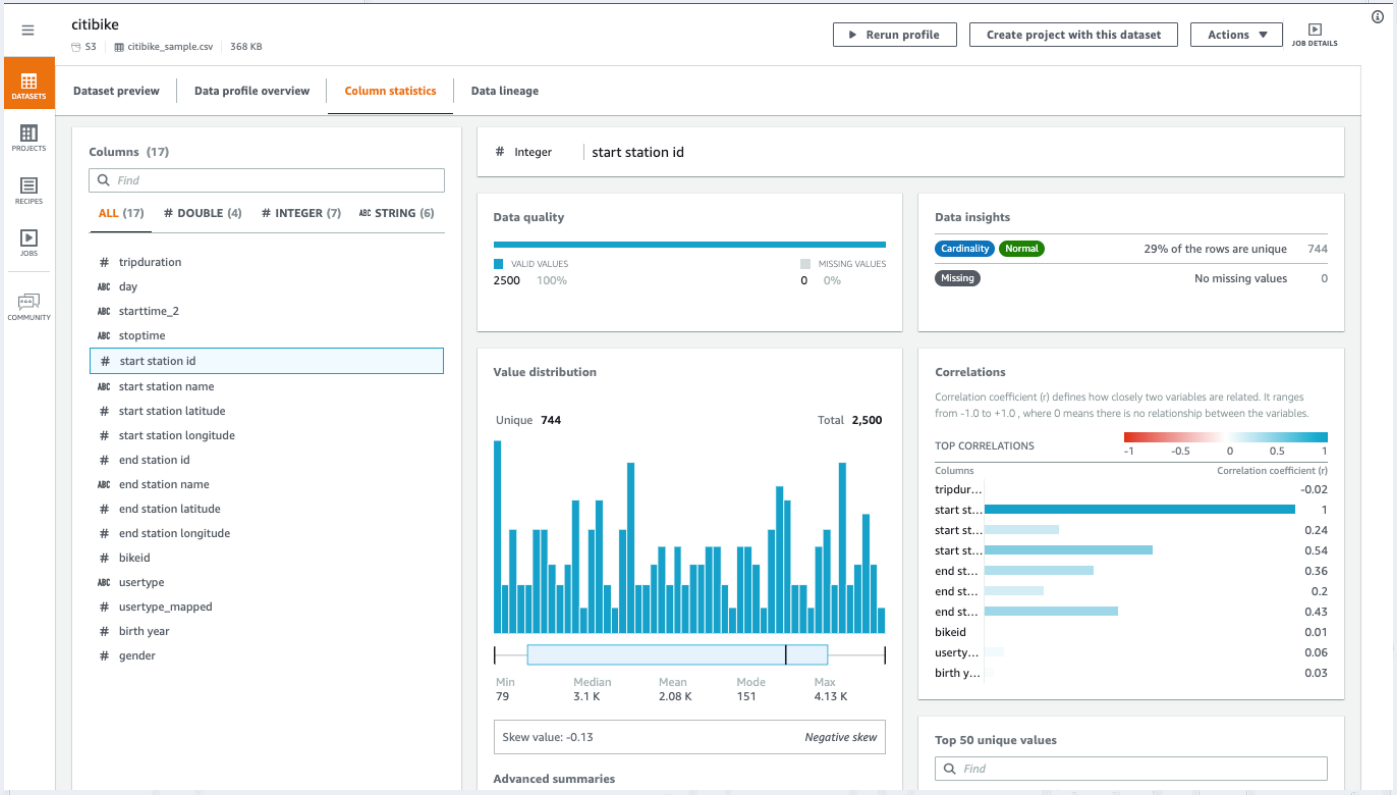
Key features:

- Visual interface for creating and managing ETL jobs.
- Provides a single pane view for monitoring and managing jobs.
- Supports data transformation and data profiling.
- Integrated with AWS Glue DataBrew for advanced data preparation.

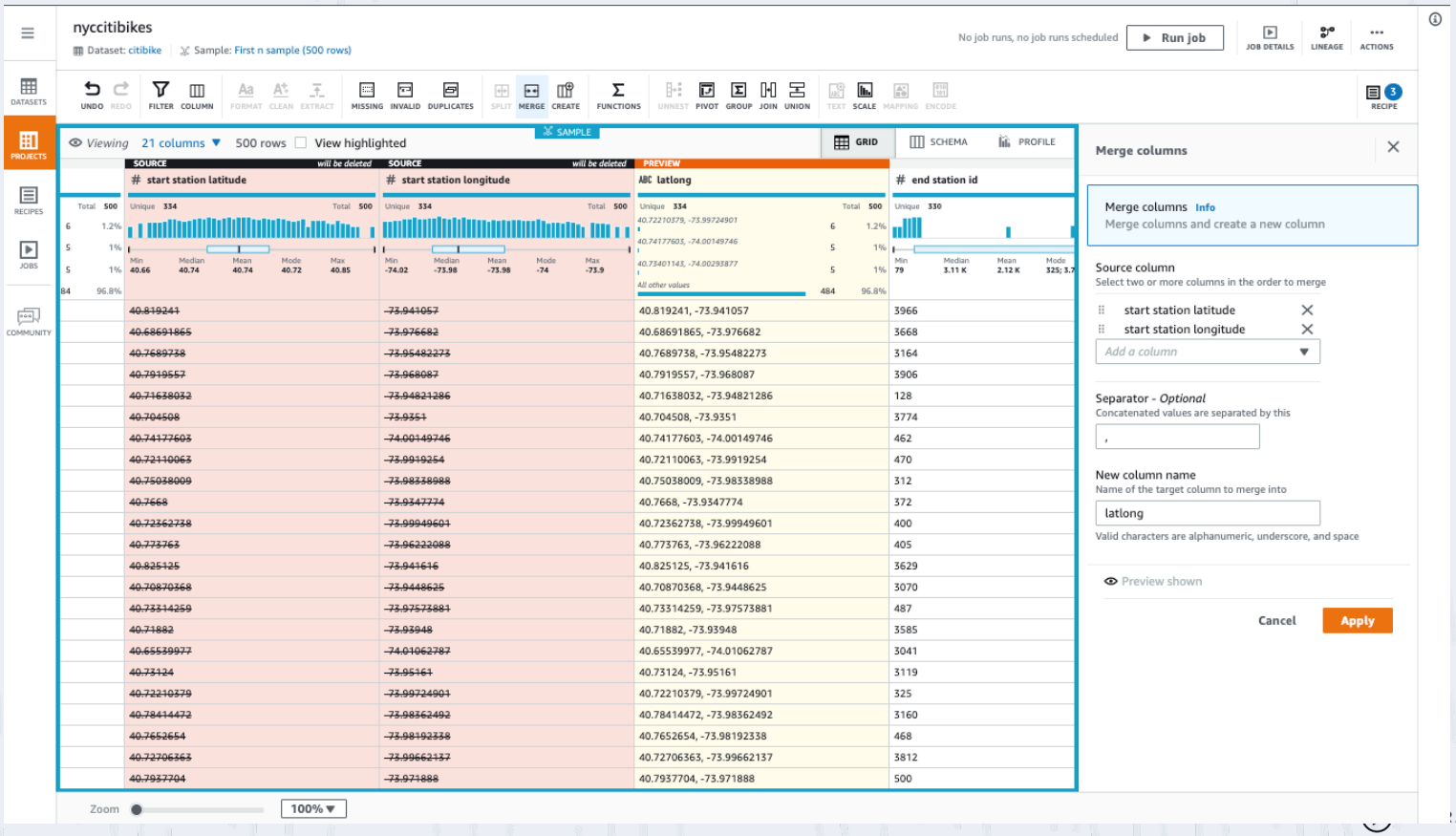
AWS Glue DataBrew



AWS Glue DataBrew



AWS Glue DataBrew



AWS Glue Studio

AWS Glue DataBrew
Use when: You need to prepare and transform data for analysis or machine learning purposes without writing code.

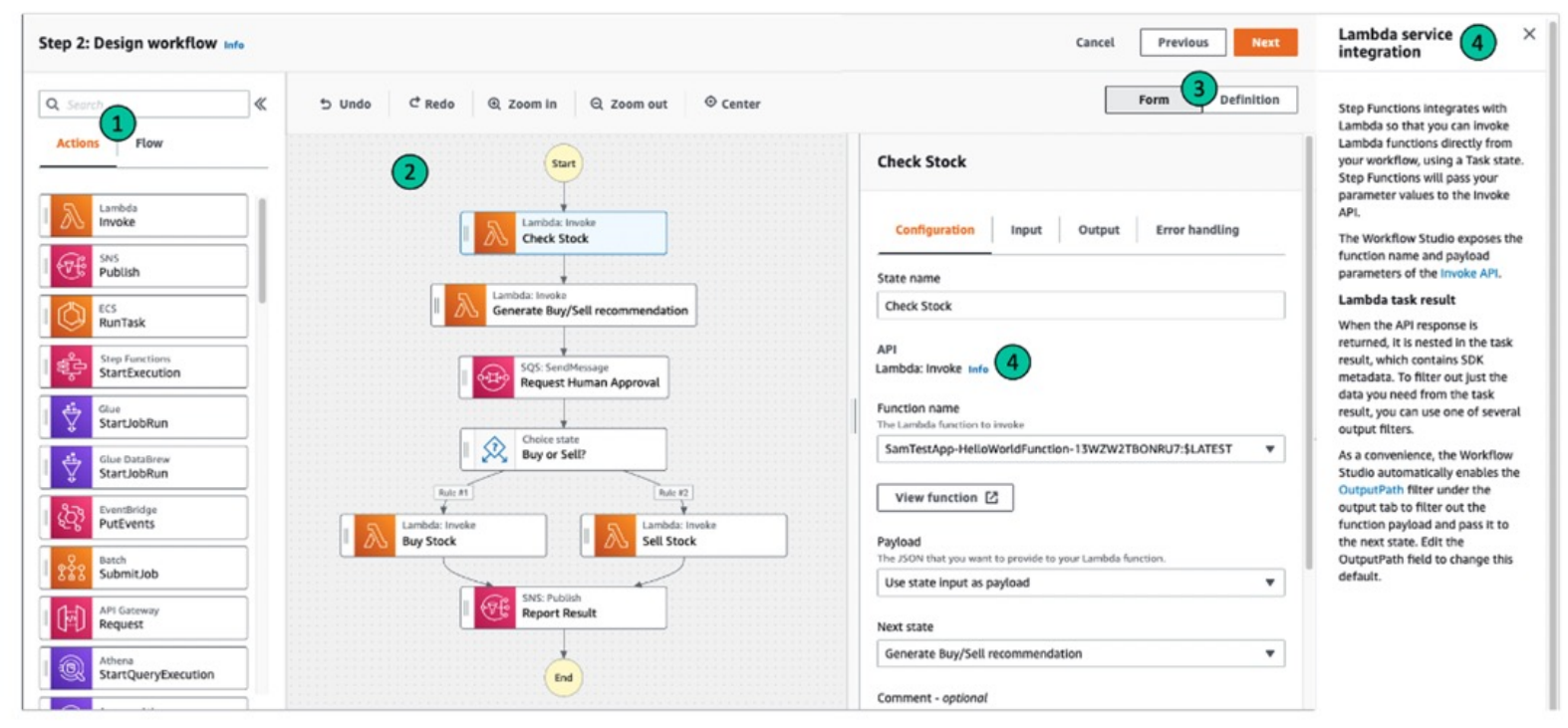
Key features:
- Visual data preparation tool for cleaning and normalizing data.
- Provides over 200 transformations without requiring code.
- Supports data profiling and data quality checks.
- Integrated with AWS Glue Studio for advanced ETL workflows.

What is Orchestration

Orchestration

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

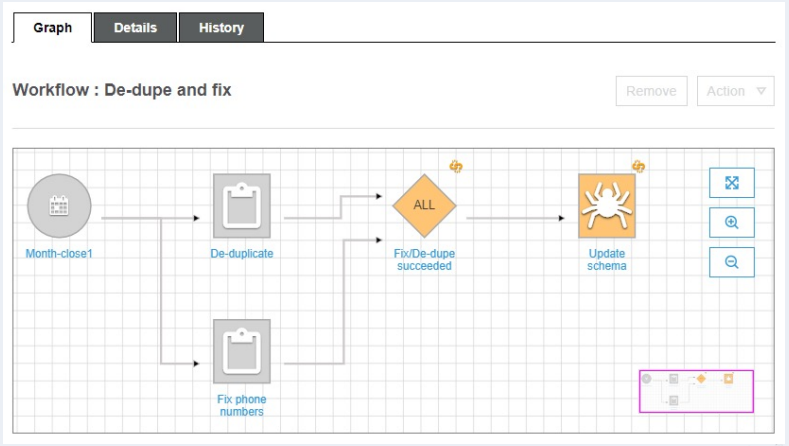
Amazon Step Function



Apache Airflow

Glue Workflows

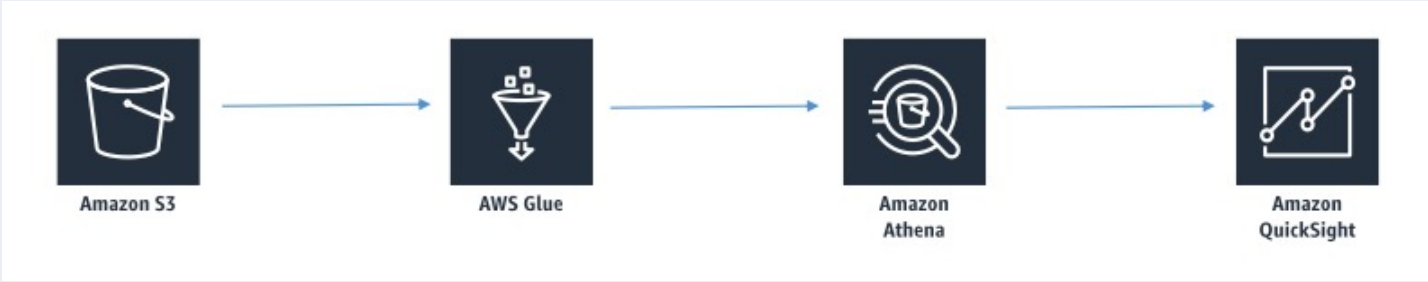
- Design multi-job, multi-crawler ETL processes run together
- Create from AWS Glue blueprint, from the console, or API
- This is only for orchestrating complex ETL operations using
- Glue



Amazon Athena

- Interactive query service for S3 (SQL)
  - No need to load data, it stays in S3
- Presto under the hood
- **Serverless**!
- Supports many data formats
  - CSV, TSV (human readable)
  - JSON (human readable)
  - ORC (columnar, splittable)
  - Parquet (columnar, splittable)
  - Avro (splittable)
  - Snappy, Zlib, LZO, Gzip compression
- Unstructured, semi-structured, or structured

Amazon Athena – Typical Pattern



Thank you!

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