BUSINESS INTELLIGENCE AND
DATA ANALYTICS

Institute of Analytics
Recognized Programme

University of Petra

307401 Big Data

Introduction to Amazon Web Services (AWS)

What is Cloud Computing?

Definition:

Cloud computing is the on-demand delivery of IT resources over the internet with pay-as-you-go pricing.

Key Features:

- Access to computing services like servers, storage, databases, networking, and software.
- Available anytime, from anywhere, via the internet.
- Scalable and flexible according to business needs.

Traditional IT vs. Cloud Computing

Feature	Traditional IT	Cloud Computing
Infrastructure	Physical hardware	Virtual infrastructure
Cost	Capital expenditure (CapEx)	Operational expenditure (OpEx)
Setup time	Weeks to months	Minutes
Scalability	Limited by physical capacity	Instantly scalable
Maintenance	Handled by in-house teams	Handled by cloud provider

Infrastructure as Software

- In cloud computing, infrastructure (servers, storage, networks) is treated as software.
- Users can provision and manage resources using code or simple interfaces.
- This eliminates the need to purchase and manage physical hardware.

Cloud Computing Models

Three Primary Service Models:

1. Infrastructure as a Service (IaaS)

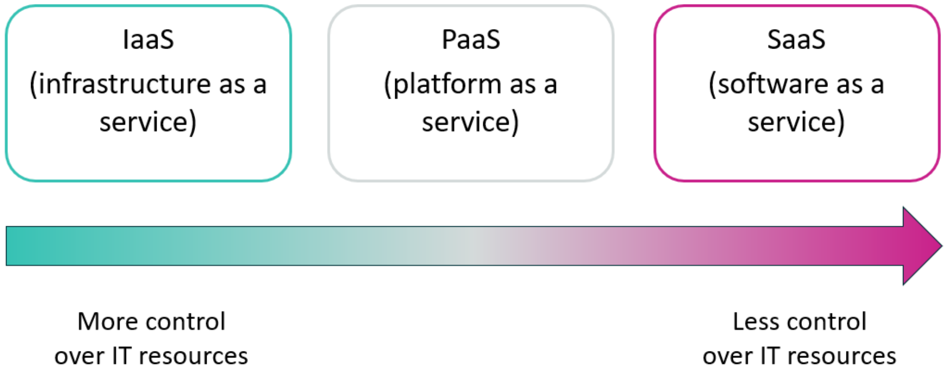
1. Provides virtualized computing resources over the internet.
2. Examples: Amazon EC2, Google Compute Engine

2. Platform as a Service (PaaS)

1. Provides hardware and software tools over the internet (usually for app development).
2. Examples: AWS Elastic Beanstalk, Google App Engine

3. Software as a Service (SaaS)

1. Delivers software applications over the internet.
2. Examples: Google Workspace, Microsoft 365



Benefits of Cloud Computing

- **Cost Efficiency:** Pay only for what you use.
- **Scalability:** Instantly adjust resources to meet demand.
- **Agility:** Deploy applications and services rapidly.
- **Global Reach:** Deliver services worldwide with minimal latency.
- **No Maintenance:** Cloud provider handles updates and maintenance.

Examples of Cloud Services

Category	Example Service	Description
Compute	Amazon EC2	Virtual servers in the cloud
Storage	Amazon S3	Scalable object storage
Database	Amazon RDS	Managed relational database service
Networking	Amazon VPC	Isolated virtual networks

Introduction to Amazon Web Services (AWS)

What is AWS?

Amazon Web Services (AWS) is a **comprehensive cloud computing platform** provided by Amazon.

Key Features:

- Delivers **on-demand** access to compute, storage, databases, networking, analytics, and more.
- Offers a **pay-as-you-go** pricing model.
- Operates as a **secure, global infrastructure** serving millions of customers.

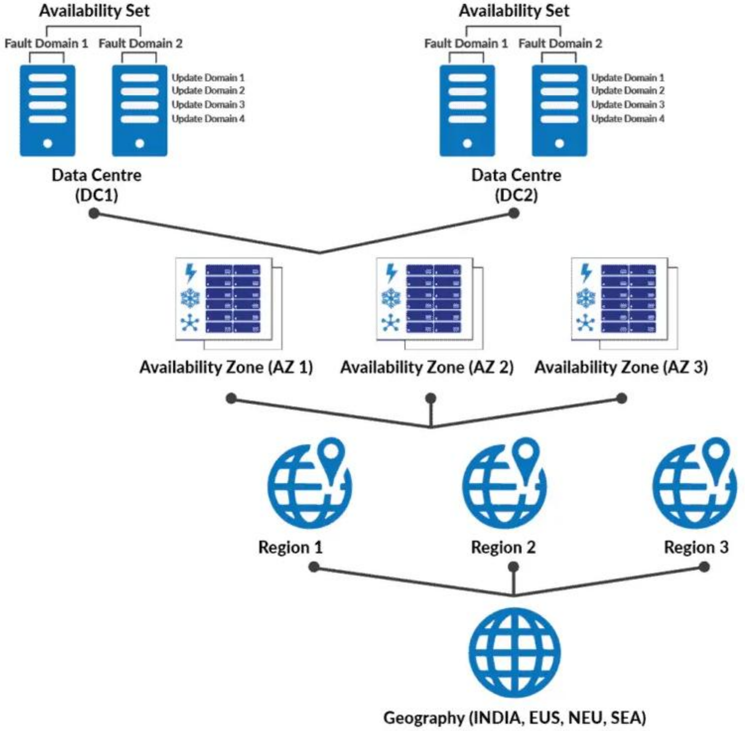
Characteristics of AWS

- **Secure:** Data encryption, identity management, and compliance.
- **Flexible:** Choose the right tools, platforms, and configurations.
- **Scalable:** Instantly adapt to demand changes.
- **Reliable:** Built-in redundancy and failover mechanisms.
- **Global:** Data centers in multiple regions and availability zones.

AWS Global Footprint

AWS operates in:

- **Regions:** Geographic locations (e.g., US East, EU Central)
- **Availability Zones:** Isolated data centers within regions
- **Edge Locations:** For content delivery and low-latency access



Purpose for this Infrastructure:

1. High availability
2. Fault tolerance
3. Low latency

https://aws.amazon.com/about-aws/global-infrastructure/#AWS_Global_Infrastructure_Map

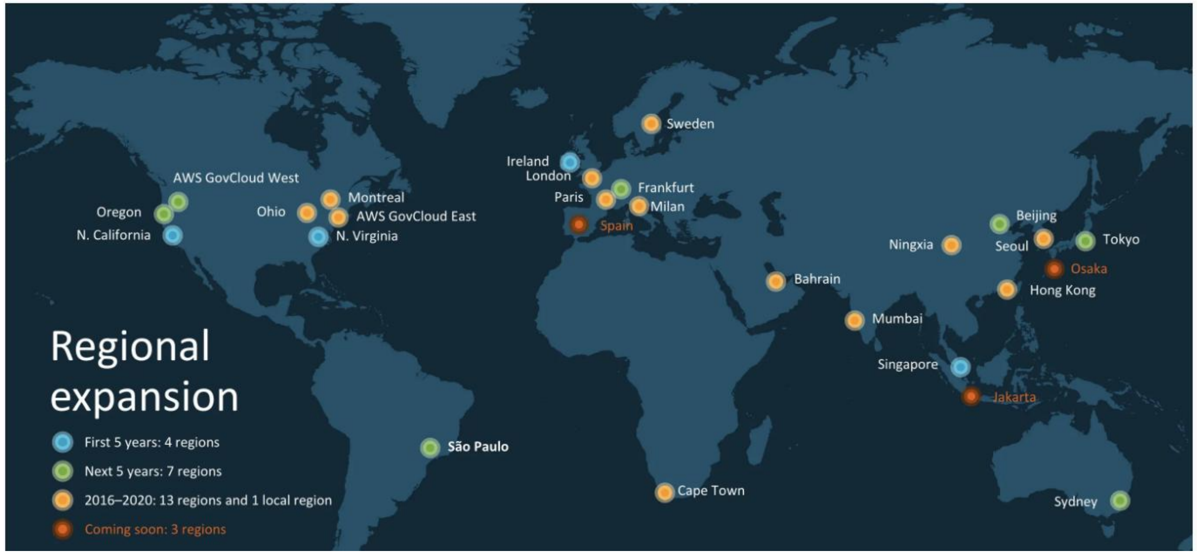
https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

AWS Regions

- Each **Region** is a separate geographic area.
- Contains **2 or more Availability Zones**.
- Customers choose Regions based on:
  - **Latency**
  - **Legal compliance**
  - **Service availability**
  - **Cost differences**

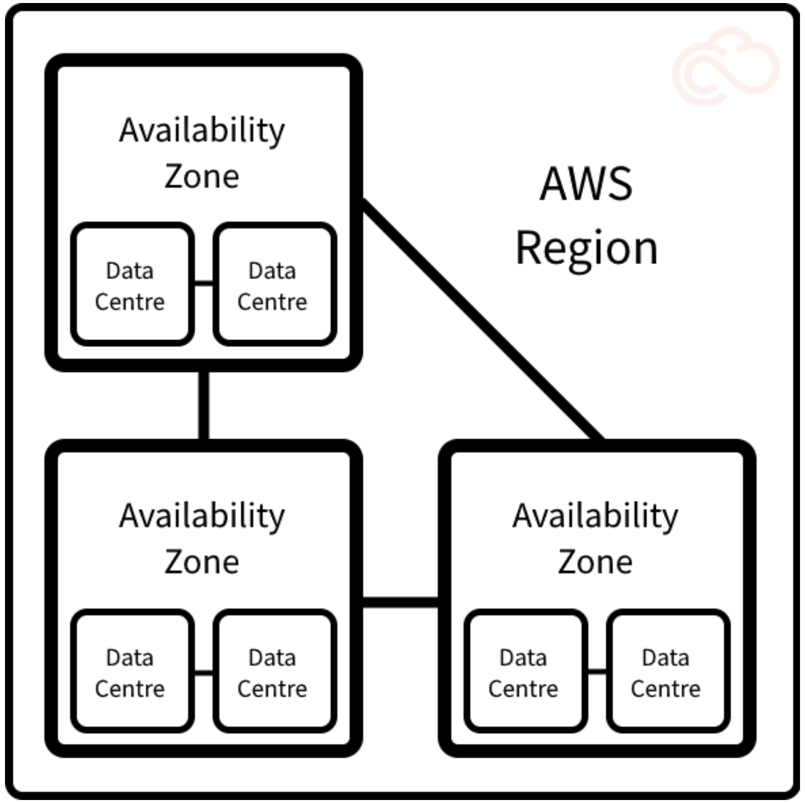
Example:

- Europe (Ireland) → eu-west-1
- US East (N. Virginia) → us-east-1



Availability Zones (AZs)

- Each Region includes multiple **Availability Zones**.
- AZs are **physically isolated** but **interconnected** using high-speed private links.
- Designed for **fault isolation** and **resiliency**.
- **Best Practice:**
  Deploy across multiple AZs to build **highly available** and **disaster-resilient** systems.

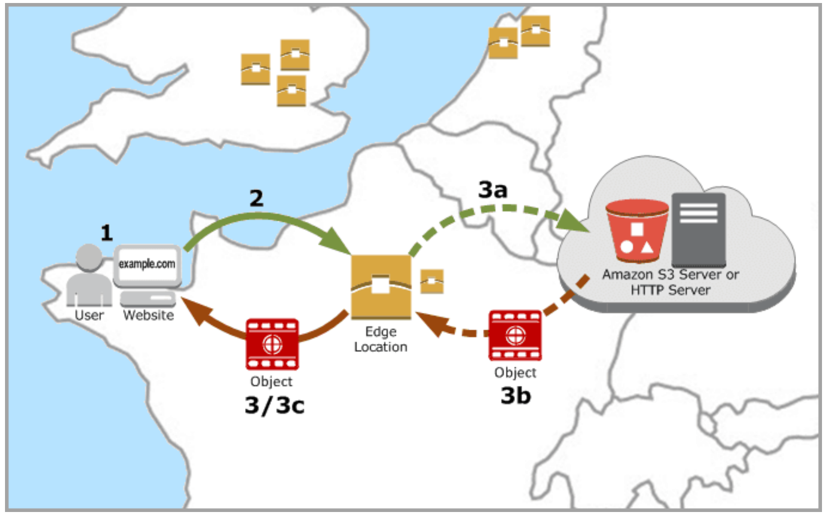


AWS Data Centers

- Secure, fault-tolerant physical facilities.
- Each AZ consists of **one or more data centers**.
- Features include:
  - Redundant power and networking
  - Access controls and physical security
  - Environmental protections (cooling, fire suppression)

Points of Presence (PoPs)

- Includes **Edge Locations** and **Regional Edge Caches**.
- Used primarily for **content delivery** via Amazon CloudFront (CDN).
- Enables **fast delivery** of static and dynamic content with **low latency**.



AWS Service Categories

Why Organize Services into Categories?

- AWS offers over 200 services.
- To make it easier to find, understand, and select the right services, AWS organizes them into **functional categories**.
- These categories reflect how businesses use cloud services to meet specific goals.

Compute services –

- Amazon EC2
- AWS Lambda
- AWS Elastic Beanstalk
- Amazon EC2 Auto Scaling
- Amazon ECS
- Amazon EKS
- Amazon ECR
- AWS Fargate

Security, Identity, and Compliance services –

- AWS IAM
- Amazon Cognito
- AWS Shield
- AWS Artifact
- AWS KMS

Storage services –

- Amazon S3
- Amazon S3 Glacier
- Amazon EFS
- Amazon EBS

Database services –

- Amazon RDS
- Amazon DynamoDB
- Amazon Redshift
- Amazon Aurora

Networking and Content Delivery services –

- Amazon VPC
- Amazon Route 53
- Amazon CloudFront
- Elastic Load Balancing

Management and Governance services –

- AWS Trusted Advisor
- AWS CloudWatch
- AWS CloudTrail
- AWS Well-Architected Tool
- AWS Auto Scaling
- AWS Command Line Interface
- AWS Config
- AWS Management Console
- AWS Organizations

AWS Cost Management services –

- AWS Cost & Usage Report
- AWS Budgets
- AWS Cost Explorer

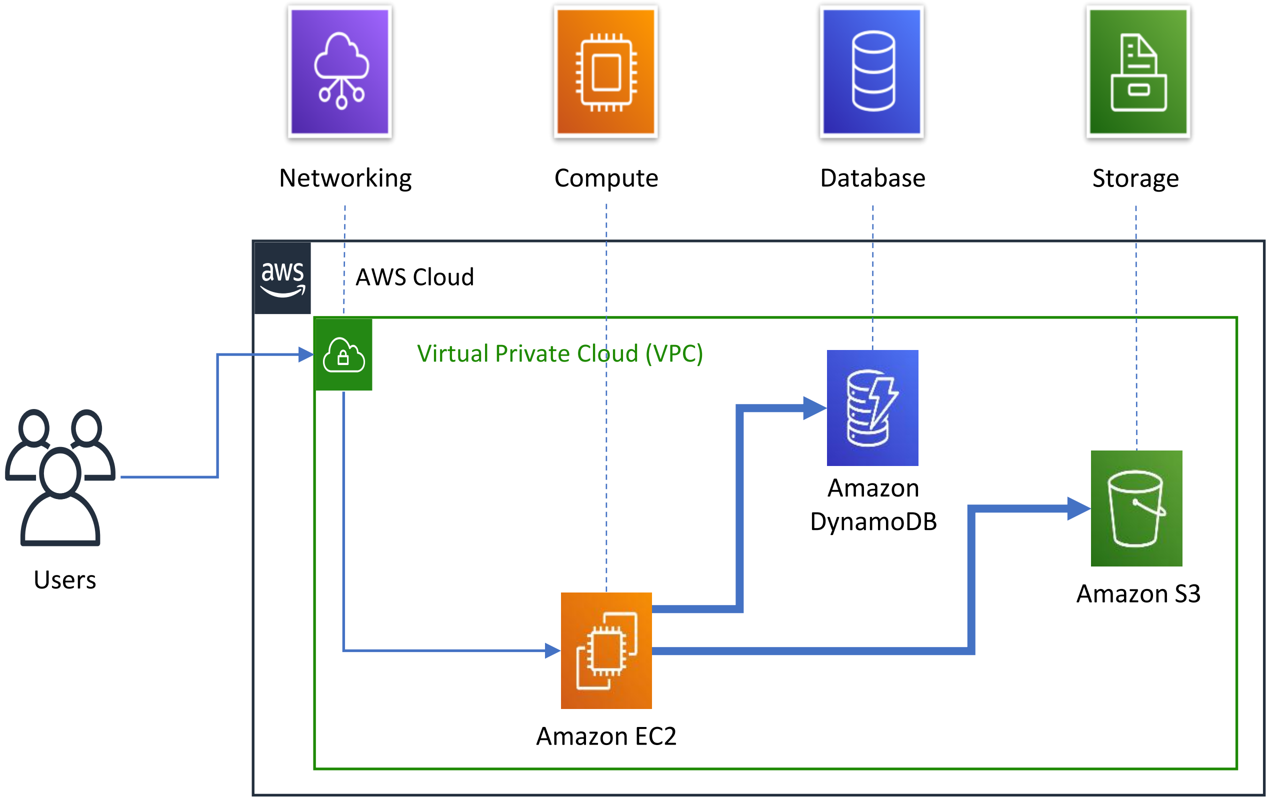
Core AWS Service Categories

Category	Focus Area
Compute	Virtual servers and serverless apps e.g. EC2, Lambda, Elastic Beanstalk
Storage	Data storage (object, block, file) e.g. S3, EBS, Glacier
Database	Managed SQL and NoSQL databases e.g. RDS, DynamoDB, Redshift
Networking & Content Delivery	Secure connectivity and low-latency delivery e.g. VPC, Route 53, CloudFront
Security, Identity & Compliance	Access control, encryption, and governance e.g. IAM, KMS, Shield
Management & Monitoring	Visibility, logging, and optimization e.g. CloudWatch, CloudTrail
Developer Tools	CI/CD, version control, code deployment
Analytics	Big data processing and analysis e.g. Athena, Glue, QuickSight
Machine Learning	AI/ML model training and deployment e.g. SageMaker, Rekognition

Example Use Case

A simple web application hosted on AWS:

- **Networking:** Amazon VPC
- **Compute:** Amazon EC2
- **Storage:** Amazon S3
- **Database:** Amazon DynamoDB
- **Security:** IAM roles and policies
- This modular setup shows how services integrate seamlessly.



Compute Services

Purpose: Deliver computing capacity in various forms

Service	Description
Amazon EC2	Virtual servers in the cloud
AWS Lambda	Serverless functions triggered by events
AWS Elastic Beanstalk	Managed app deployment environment
Amazon ECS/EKS	Containers and Kubernetes management
AWS Fargate	Serverless containers

Storage Services

Purpose: Store, archive, and retrieve data efficiently

Service	Description
Amazon S3	Scalable object storage
Amazon EBS	Block storage for EC2 instances
Amazon EFS	Elastic file storage for shared access
Amazon S3 Glacier	Archival storage with low retrieval cost

Database Services

Purpose: Run databases without managing infrastructure

Service	Description
Amazon RDS	Managed relational databases (e.g., MySQL, PostgreSQL)
Amazon Aurora	High-performance relational DB compatible with MySQL/PostgreSQL
Amazon DynamoDB	Managed NoSQL key-value/document store
Amazon Redshift	Data warehouse for large-scale analytics

Networking & Content Delivery

Purpose: Connect resources securely and deliver content globally

Service	Description
Amazon VPC	Isolated virtual networks
Route 53	Scalable DNS and domain name routing
CloudFront	Global content delivery network (CDN)
Elastic Load Balancing	Distributes traffic to maintain performance

Security, Identity, and Compliance

Purpose: Protect data, manage user access, and meet compliance standards

Service	Description
AWS IAM	Identity and Access Management
AWS KMS	Key Management Service for encryption
AWS Shield	DDoS protection
AWS Artifact	Compliance and audit documentation

Monitoring & Management

Purpose: Improve visibility, control, and efficiency

Service	Description
Amazon CloudWatch	Monitors metrics, logs, alarms
AWS CloudTrail	Tracks user activity and API usage
AWS Config	Audits configuration changes
AWS Trusted Advisor	Optimization and best practices review

AWS Cloud Security and Identity Access Management (IAM)

Shared Responsibility Model

AWS uses a shared responsibility model to divide security duties between AWS and the customer.

Responsibility Area	AWS (Provider)	Customer (User)
Physical security	✓	
Infrastructure (hardware, VMs)	✓	
Network configuration		✓
OS, apps, and data		✓
Access control and permissions		✓

Summary: AWS secures the **cloud itself;** customers secure what they do **in the cloud**.

Responsibilities

AWS is responsible for:

- Securing data centers, servers, networking, and storage infrastructure.
- Providing encrypted services and physical redundancy.
- Maintaining hardware, hypervisors, and virtualization layers.

The customer is responsible for:

- Securing your applications and data.
- Managing access through IAM (users, roles, groups).
- Configuring firewalls (security groups) and encryption.
- Regularly updating your OS, patches, and credentials.

Security at Different Service Levels

Cloud Model	Responsibility Level
IaaS	Customer manages OS, apps, firewalls
PaaS	Customer manages data and app logic
SaaS	Customer manages only the data and access

Example:

- EC2 (IaaS) requires OS patching.
- Lambda (PaaS) abstracts OS, customer only writes code.

What is IAM (Identity and Access Management)?

IAM is the AWS service that allows you to:

- **Authenticate:** Who is trying to access the system.
- **Authorize:** What they are allowed to do.

Key Functions:

- Define **users**, **groups**, **roles**, and **policies**
- Control access to AWS **resources** (e.g., S3, EC2)

IAM Key Concepts

Concept	Description
User	Represents a person or service needing access
Group	A collection of users with shared permissions
Role	Temporary access granted to trusted entities
Policy	A JSON document defining allowed or denied actions on resources

IAM Roles – Use Cases

IAM Roles are used to:

- Grant applications (e.g., EC2 instances) permission to access AWS services.
- Delegate access to users in other AWS accounts.
- Enable temporary access with conditions (e.g., session duration).

IAM Policies

- Written in JSON format
- Two types:
  - Identity-based: Attach to users, groups, or roles
  - Resource-based: Attach directly to resources (e.g., S3 bucket)
- Example Allow Policy:

```
{
  "Effect": "Allow",
  "Action": ["s3:*"],
  "Resource": "arn:aws:s3:::example-bucket/*"
}
```

Introduction to AWS Data Engineering

What is Data Engineering?

- Data engineering is the process of designing, building, and maintaining the **infrastructure and systems** that enable the collection, storage, processing, and analysis of large amounts of data.
- It involves tasks such as data integration, data transformation, data storage and retrieval, data quality assurance, and data pipeline development.
- Data engineering is a crucial component of data-driven organizations and enables them to effectively leverage the power of data to gain insights, make informed decisions, and drive business growth

Data Pipeline

- Data pipeline is the infrastructure that you build to support data-driven decision-making.
- At the most basic level, any data pipeline infrastructure must be able to:
  1. Bring data in
  2. Store it
  3. Provide the means to process and analyze data to derive insights.
- As the data engineer or data scientist, it's your job to build the pipeline to figure out what's appropriate and how you're going to do it.
- Often this will include **exploring and experimenting** to get to know the data and the best way to derive value from it.

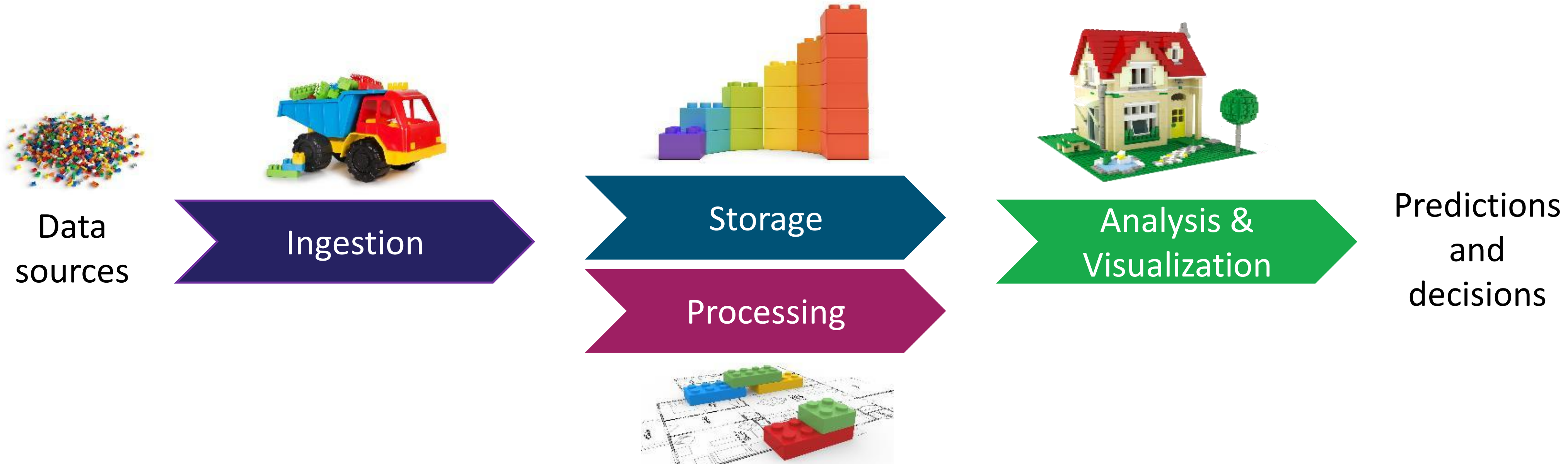
Collect data

Store and process data

Build something useful with data

Layers of the pipeline infrastructure

- The infrastructure layers that you need to build include methods to ingest and store data from the data sources that you have identified.
- You need to make the stored data accessible for decision-making processes.
- You must also create the infrastructure to process, analyze, and visualize data by using tools that are appropriate to the use case.
- To build an appropriate infrastructure, you will need to understand the nature of the data and the intended type of processing and analysis to be performed.


Data sources

Ingestion

Storage

Processing

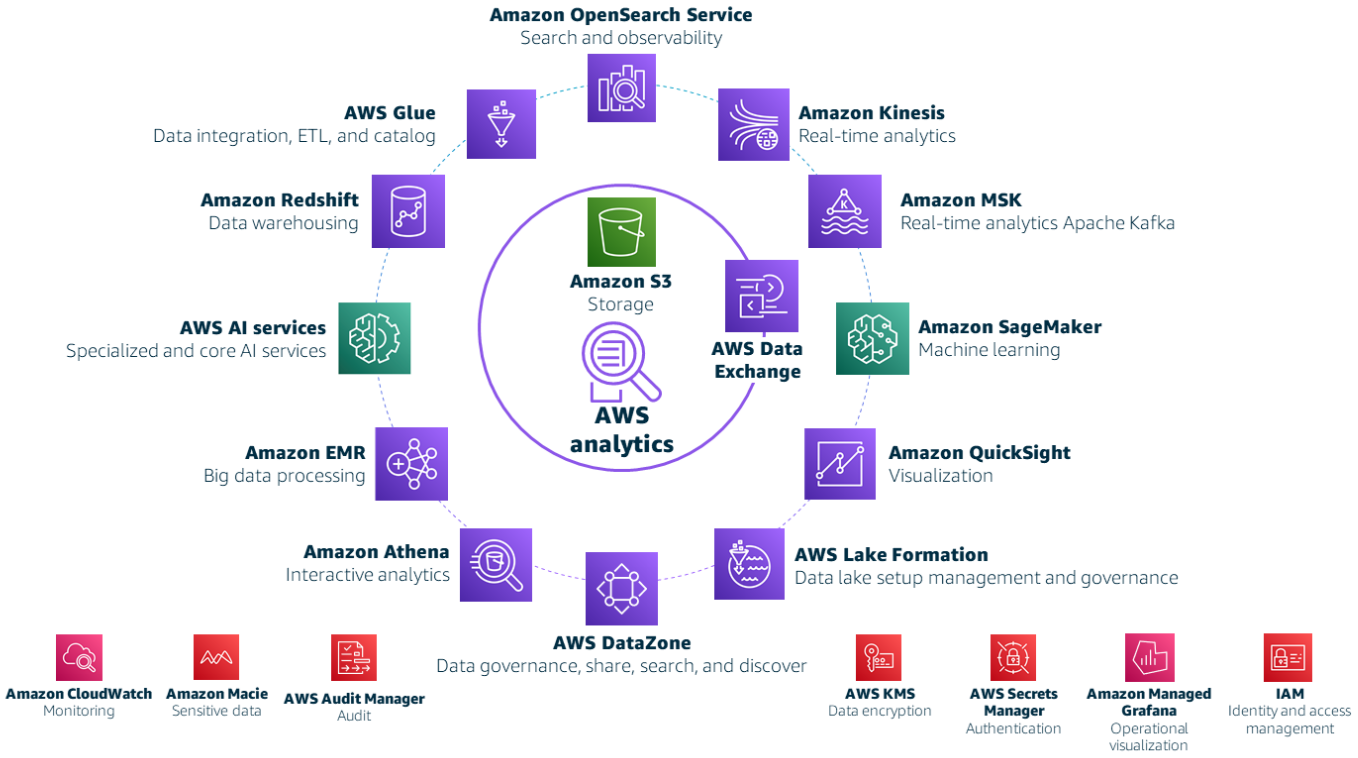
Analysis & Visualization

Predictions and decisions

AWS Data Engineering Related Technologies

- **Core AWS Services**:
  - **S3**: Scalable object storage for raw/staged/curated data.
  - **Athena**: Serverless querying on S3 using SQL.
  - **Glue**: Serverless data integration (ETL), data cataloging.
  - **Redshift**: Data warehousing and analytics.
  - **Kinesis**: Real-time data streaming.
  - **Lambda**: Serverless compute for lightweight ETL.
  - **EMR**: Managed Hadoop/Spark clusters.
  - **Data Pipeline / Step Functions**: Workflow orchestration.

AWS modern data architecture



Amazon OpenSearch Service Search and observability

AWS Glue Data integration, ETL, and catalog

Amazon Kinesis Real-time analytics

Amazon Redshift Data warehousing

Amazon MSK Real-time analytics Apache Kafka

Amazon S3 Storage

AWS analytics

AWS Data Exchange

AWS AI services Specialized and core AI services

Amazon SageMaker Machine learning

Amazon EMR Big data processing

Amazon QuickSight Visualization

Amazon Athena Interactive analytics

AWS Lake Formation Data lake setup management and governance

AWS DataZone Data governance, share, search, and discover

Amazon CloudWatch Monitoring

Amazon Macie Sensitive data

AWS Audit Manager Audit

AWS KMS Data encryption

AWS Secrets Manager Authentication

Amazon Managed Grafana Operational visualization

IAM Identity and access management

AWS Services for Big Data Workloads

AWS offers managed services that cover the full data lifecycle—from ingestion to storage, analysis, and prediction.

Function	Service
Data Storage	Amazon S3, Amazon Glacier
Batch Processing	Amazon EMR (Hadoop/Spark)
Real-Time Streams	Amazon Kinesis, Amazon MSK (Kafka)
Data Integration	AWS Glue, AWS Data Pipeline
Analytics	Amazon Athena, Amazon Redshift, QuickSight

amazon
S3

Amazon S3

- **Amazon S3 (Simple Storage Service):** A highly scalable, durable, and available object storage service provided by Amazon Web Services (AWS).
- **Core Purpose:** S3 is designed to store and retrieve any amount of data from anywhere on the web. It's often referred to as "storage for the internet."
- **Key Concept:** Data is stored as **objects** within **buckets**. An object is a file (e.g., image, video, document) and any associated metadata. A bucket is a container for those objects.

S3 Storage Classes

S3 offers different storage classes optimized for specific use cases, based on data access frequency and cost.

- **S3 Standard:** General-purpose storage for frequently accessed data. It's the default and is ideal for a wide range of use cases like websites, mobile apps, and big data analytics.
- **S3 Intelligent-Tiering:** Automatically moves data between two access tiers based on usage patterns. It's perfect for data with unknown or changing access needs.
- **S3 Standard-IA (Infrequent Access):** For long-lived, infrequently accessed data. It's cheaper than Standard but has a per-gigabyte retrieval fee.
- **S3 One Zone-IA:** Same as Standard-IA, but data is stored in a single Availability Zone, making it less durable but even more cost-effective. Ideal for secondary backup copies or data that can be easily recreated.
- **S3 Glacier:** Low-cost storage for data archiving. Data retrieval takes minutes to hours.
- **S3 Glacier Deep Archive:** The lowest-cost storage option for long-term data archiving (10+ years). Retrieval time is in hours.

S3 is a Premier Data Lake Foundation

- A **data lake** is a centralized repository that allows you to store all your structured and unstructured data at any scale. S3 is the perfect foundation for a data lake due to these key attributes:
- **Massive Scalability:** S3 provides virtually **unlimited storage**, allowing you to store petabytes or even exabytes of data without managing underlying infrastructure.
- **Durability and Reliability:** With **11 nines of durability** (99.999999999%), your data is safe and protected against loss.
- **Cost-Effectiveness:** You only pay for the storage you use, making it very affordable. The different storage classes allow you to optimize costs for data that isn't accessed frequently.
- **Schema-on-Read:** S3 allows you to store data in its original, raw format. You can apply a schema only when you query the data, providing incredible flexibility for diverse data types.
- **Integration with the AWS Ecosystem:** S3 seamlessly integrates with a vast array of other AWS analytics, machine learning, and business intelligence services, which is essential for a functional data lake.

S3's Role in a Data Lake Architecture

- **Data Ingestion:** S3 serves as the primary **landing zone** for all raw data, regardless of format (structured, semi-structured, or unstructured). Data can be ingested from streaming sources, databases, IoT devices, or log files.
- **Raw Zone Storage:** The raw, unprocessed data is stored in its original format in an S3 bucket. This "raw zone" ensures data immutability and provides an auditable history.
- **Data Transformation:** AWS services like **AWS Glue** (ETL) or **Amazon EMR** (big data processing) read the data from S3, transform it, and write the refined, "cleaned" data back to S3.
- **Analytics & Consumption:** Data is now ready for analysis. Services like **Amazon Athena** (interactive SQL queries) and **Amazon Redshift Spectrum** (data warehousing) can query the data directly in S3.
- **Machine Learning: Amazon SageMaker** can access the S3 data lake to build, train, and deploy machine learning models at scale.

amazon
ATHENA

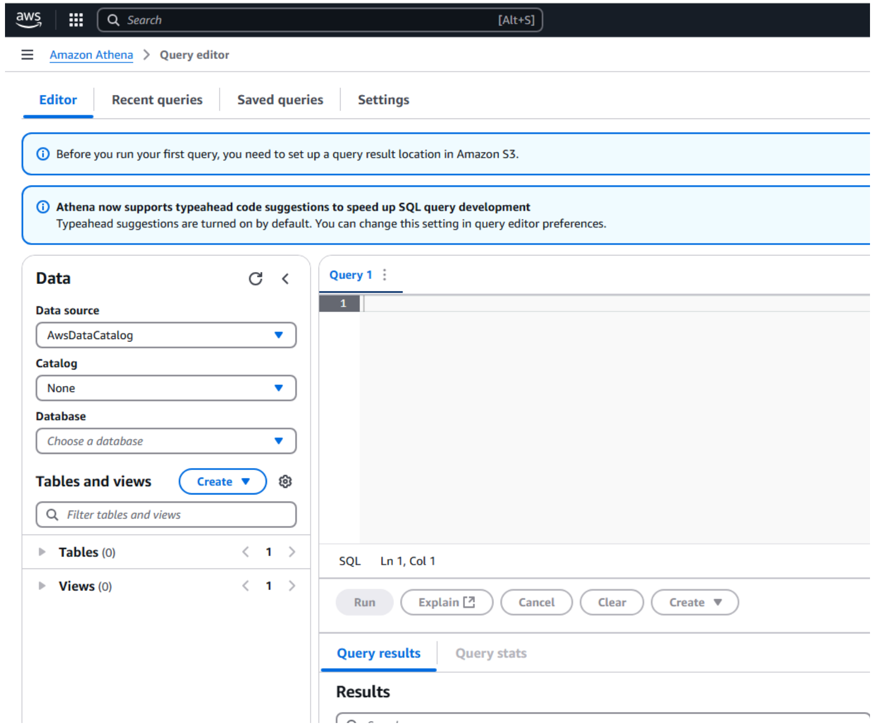
Introduction to Amazon Athena

Definition:

- AWS Athena is a serverless query service for directly analyzing data in Amazon S3 using standard SQL.
- No infrastructure to set up or manage — you pay only for the data scanned per query.
- **Ad hoc querying**: instantly run queries without ETL pipelines.

Trino and Athena

- Athena is built on the open-source Trino query engine
- Trino (formerly PrestoSQL) enables fast, scalable query processing
- Distributed SQL query engine for big data processing



Before you run your first query, you need to set up a query result location in Amazon S3.

Athena now supports typeahead code suggestions to speed up SQL query development
Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.

Data source: AwsDataCatalog

Catalog: None

Database: Choose a database

Athena Core Components

1. Data Sources:

- Data stored in Amazon S3.
- Supports various data formats, including CSV, JSON, ORC, Avro, and Parquet.

2. Query Engine:

- Built on Presto, an open-source distributed SQL query engine optimized for low-latency data analysis.

3. Integration with AWS Glue:

- Utilizes AWS Glue Data Catalog for managing metadata and schema definitions.

Common Data File Formats

Format	Text / Binary	Row vs Column	Schema Handling	Compression	Splitable	Typical Use Cases
CSV	Plain-text	Row	None (external schema)	No (gzip/zip external)	No (gzipped) / Yes (plain)	Spreadsheets, exports
JSON / NDJSON	Plain-text	Row (hierarchical)	Self-describing (names and values)	No (gzip/bz2 common)	No (compressed) / Yes (NDJSON)	APIs, logs, documents
Avro	Binary	Row	Required (schema in header)	Yes (Snappy/Deflate)	Yes	Kafka, service exchange
Parquet	Binary	Column	Required (embedded)	Yes (Snappy/Gzip/Zstd)	Yes	Analytics, data lakes
ORC	Binary	Column	Required (embedded)	Yes (Zlib/Snappy/Zstd)	Yes	Hive, warehousing
XML	Plain-text	Tree-structured	Self-describing (tags)	No (gzip external)	No (compressed) / Yes (plain)	Legacy configs/docs

How Athena Works

1. Amazon S3 Bucket

Stores raw or preprocessed data in formats like CSV, JSON, Parquet, ORC, or Avro.

2. AWS Glue Crawler

Automatically scans the data in S3 to detect schema and partition information.

3. AWS Glue Data Catalog

Stores metadata (tables, columns, types, partitions) that define how Athena interprets the data.

4. Amazon Athena

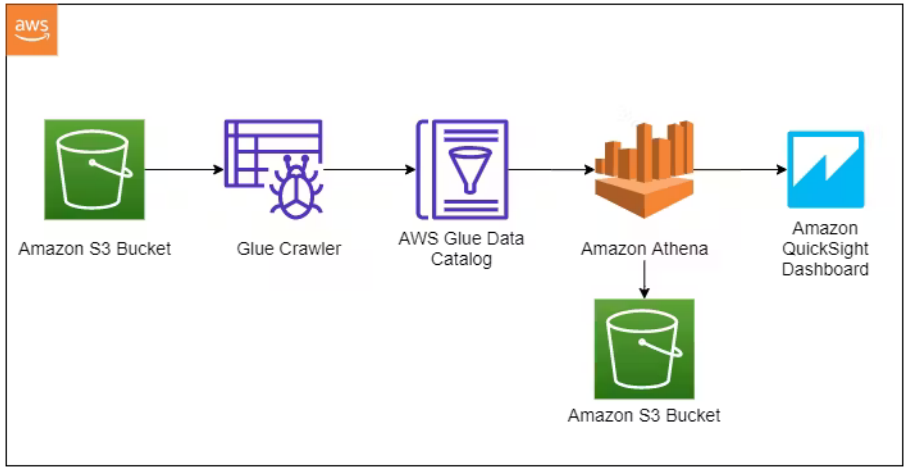
A serverless SQL query engine that uses the Glue Data Catalog to execute SQL queries directly on S3 data.

5. Amazon S3 (Query Results)

Athena writes the query results back to another (or the same) S3 bucket for further use.

6. Amazon QuickSight Dashboard

Connects to Athena to visualize the results, build dashboards, and generate business insights.



Optimizing Queries with Compression, Partitions and Buckets

Athena's cost is based on data scanned — so optimizing file formats and partitions can massively reduce costs in Big Data environments.

Golden Rule: Less data scanned = Faster query + Lower cost

Optimization Methods:

Compression

- **Purpose:** Store data using compressed formats (e.g., GZIP, Parquet) to:
- Save S3 storage space.
- Reduce the amount of data Athena scans (saving cost).

Buckets

- **Definition:** Logical grouping of related records based on **high-cardinality** fields.
- **Purpose:** Speed up query performance by minimizing full dataset scans.
- **Example:** Splitting taxi trips into separate buckets by each day's timestamp.

Partitioning

- **Definition:** Splitting data into logical directories based on **low-cardinality** fields.
- **Example:** Partitioning taxi trips by **payment type** (Credit card, Cash).

Optimizing Athena queries by using buckets

- When you work with large datasets that are spread out across multiple files, two major goals are to optimize query performance and to minimize cost. Cost for Athena is based on usage, which is the amount of data that is scanned, and prices vary based on your Region.
- Three possible strategies that you can use to minimize your costs and improve performance are compressing data, bucketizing data, and partitioning data.
- **Compressing data:** Compress your data by using one of the open standards for file compression (such as Gzip or tar). Compression results in a smaller size for the dataset when it's stored in Amazon S3.
- The cardinality of your data also effects how you should optimize your queries. For more information, see Cardinality (SQL Statements). There are two options to optimize based upon high or low cardinality. These are:
    - **Bucketizing data:** For high cardinality with data, store records in distinct *buckets* (not to be confused with S3 buckets) based upon a shared value in a specific field. Consider bucketizing data as part of the preprocessing phase of your data pipeline. In this lab, data for a single month will be bucketized separately from the original dataset, which contains data for an entire year. This strategy will help to optimize performance.
    - **Partitioning data:** You can also use partitions to improve performance and reduce cost. Partitioning is used with low-cardinality data, which means that fields have few unique or distinct values.

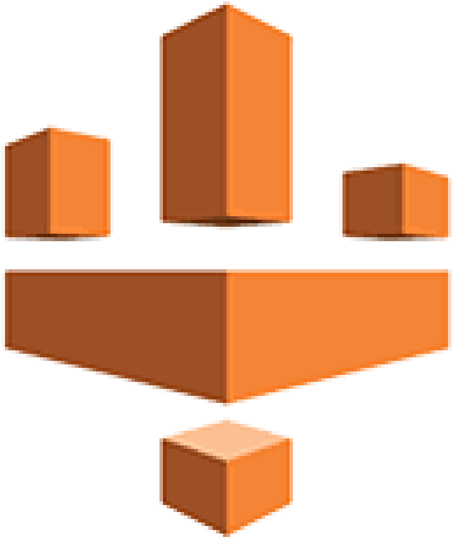
This slide is extracted from AWS Engineering Course: Athena Lab 4

Handling Large Files

1. Use appropriate file formats (Parquet/ORC)
2. Implement effective partitioning strategies
3. Apply bucketing for high-cardinality columns
4. Use columnar compression
5. Control query scope with filters

Create a Table with Partition in Athena

```
CREATE EXTERNAL TABLE sales_records ( product_id INT, sales_amount DECIMAL(10, 2), transaction_date DATE )
PARTITIONED BY (year INT, month INT)
LOCATION 's3://your-bucket/sales-data/';
```



Amazon Glue

Introduction to AWS Glue

What is AWS Glue?

- AWS Glue is a fully managed, serverless data integration service that simplifies the process of discovering, preparing, and combining data for analytics, machine learning, and application development.

Key Features:

- **Serverless Architecture:** No infrastructure to manage; AWS handles provisioning and scaling.
- **Data Cataloging:** Centralized metadata repository for all data assets.
- **Automated ETL:** Automatically generates code to extract, transform, and load data.
- **Support for Multiple Data Sources:** Integrates with various AWS and third-party data sources.

Core Components of AWS Glue

1. **AWS Glue Data Catalog:** A centralized metadata repository that stores table definitions, job metadata, and other control information to manage your ETL environment.

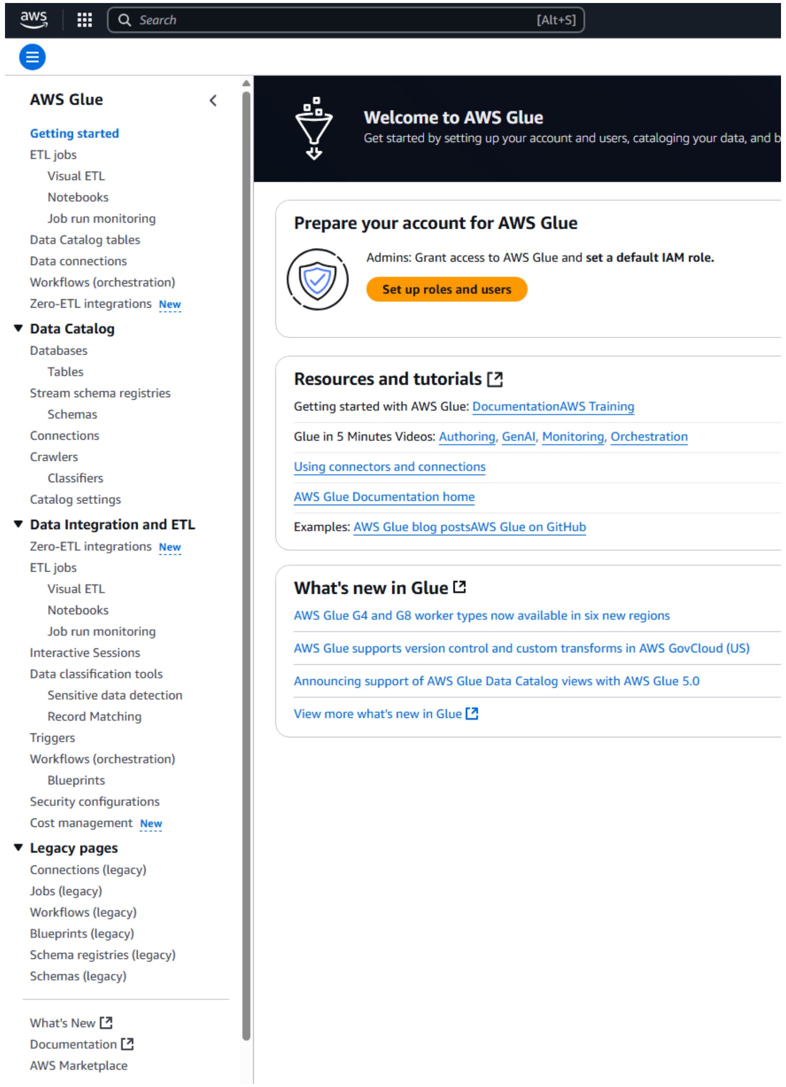
2. **Crawlers:** Automated processes that scan data sources to infer schemas and populate the Data Catalog.

3. **Jobs:** Scripts that define the ETL process, written in Python or Scala, which can be generated automatically or customized.

4. **Triggers:** Mechanisms to initiate jobs based on schedules or events.

5. **Workflows:** Orchestrate multiple jobs and crawlers to build complex ETL pipelines.

6. **Transformers**: Built-in and custom data transformation functions.



Welcome to AWS Glue

Get started by setting up your account and users, cataloging your data, and b

Prepare your account for AWS Glue

Admins: Grant access to AWS Glue and **set a default IAM role.**

Resources and tutorials

Getting started with AWS Glue: DocumentationAWS Training

Glue in 5 Minutes Videos: Authoring, GenAI, Monitoring, Orchestration

Using connectors and connections

AWS Glue Documentation home

Examples: AWS Glue blog postsAWS Glue on GitHub

What's new in Glue

AWS Glue G4 and G8 worker types now available in six new regions

AWS Glue supports version control and custom transforms in AWS GovCloud (US)

Announcing support of AWS Glue Data Catalog views with AWS Glue 5.0

View more what's new in Glue

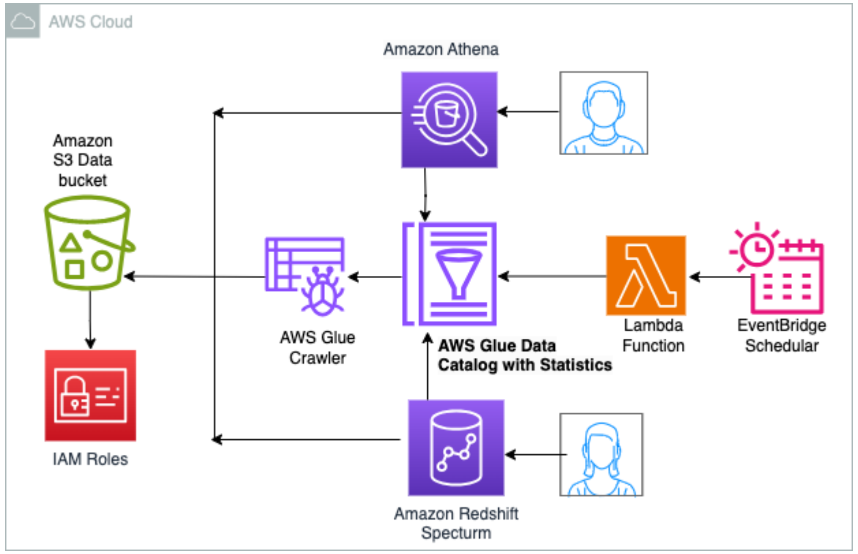
AWS Glue Data Catalog

Purpose:

- Acts as a persistent metadata store for all your data assets, enabling data discovery and schema management.

Features:

- **Schema Versioning:** Tracks changes to data schemas over time.
- **Integration:** Works seamlessly with Amazon Athena, Redshift Spectrum, and Amazon EMR.
- **Security:** Supports fine-grained access control using AWS Identity and Access Management (IAM).



AWS Glue Crawlers

Functionality:

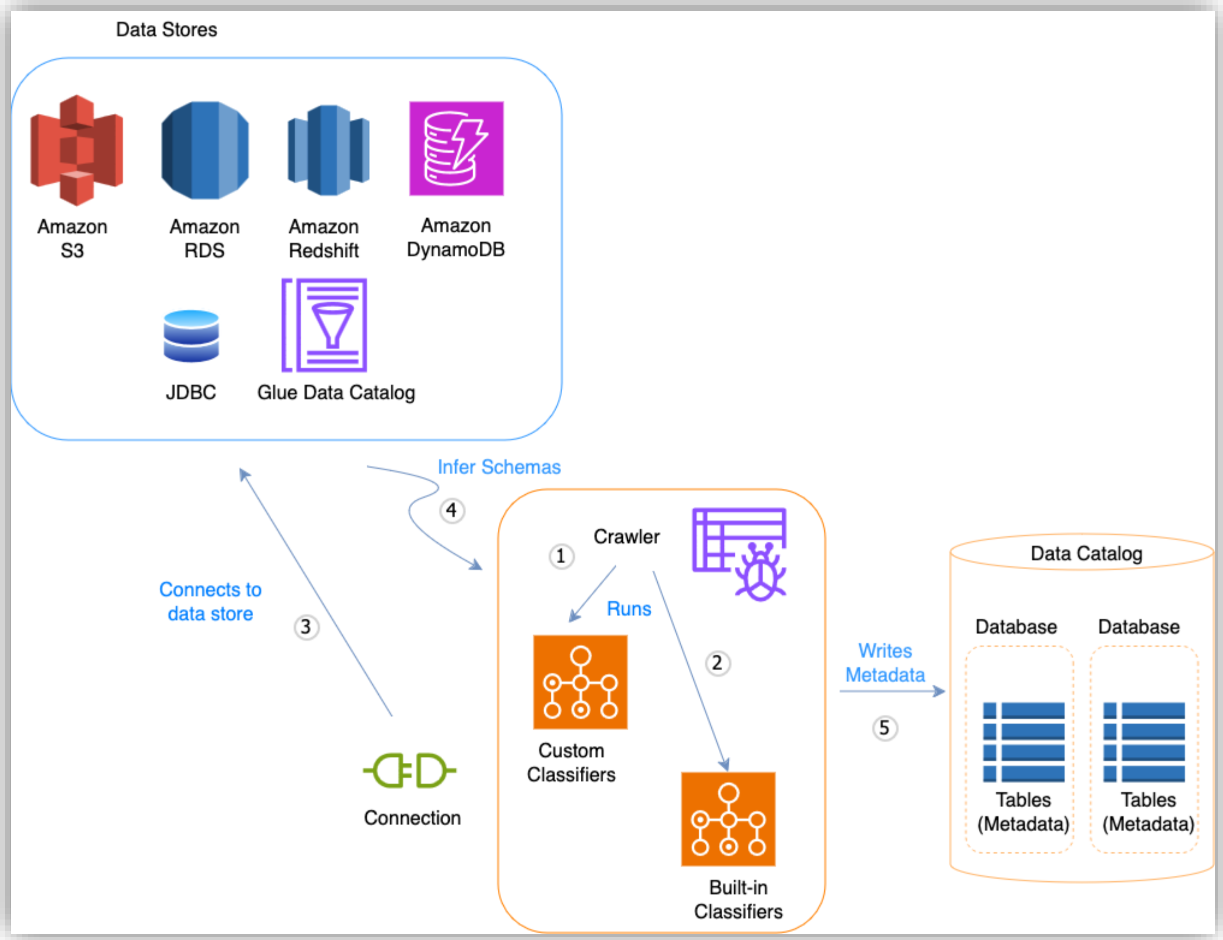
- Connect to various data sources, infer schemas, and populate the Data Catalog with metadata.

Supported Data Stores:

- Amazon S3, Amazon RDS, Amazon Redshift, DynamoDB, and more.

Custom Classifiers:

- Define custom logic to classify data formats not supported by default.



AWS Glue Jobs

Types of Jobs:

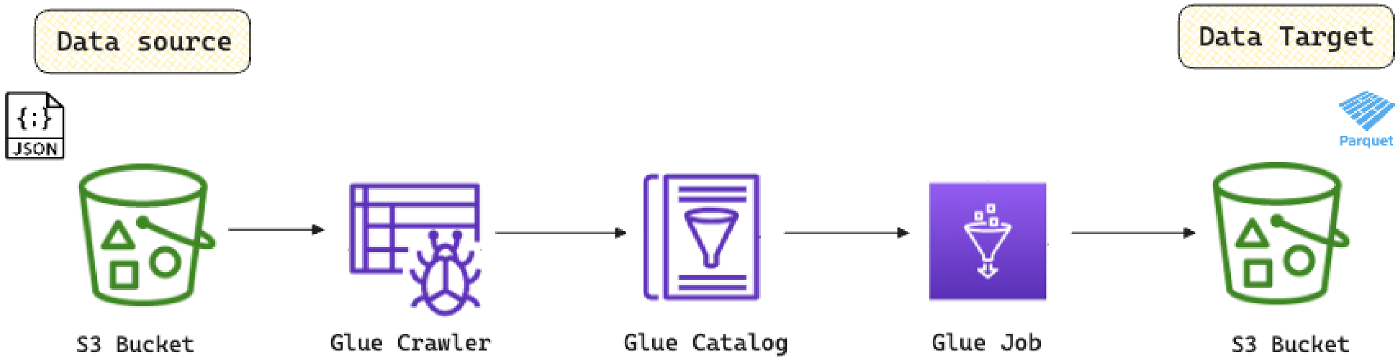
- **ETL Jobs:** Perform extract, transform, and load operations.
- **Streaming Jobs:** Process real-time data streams using Apache Spark Structured Streaming.

Development Options:

- **AWS Glue Studio:** Visual interface to create and manage jobs.
- **Script Editor:** Write custom scripts in Python or Scala.

Job Monitoring:

- Track job runs, monitor logs, and set up alerts for failures or delays.



Triggers and Workflows

Triggers:

- **Scheduled Triggers:** Initiate jobs at specified times.
- **Event-Based Triggers:** Start jobs in response to events like new data arrival.

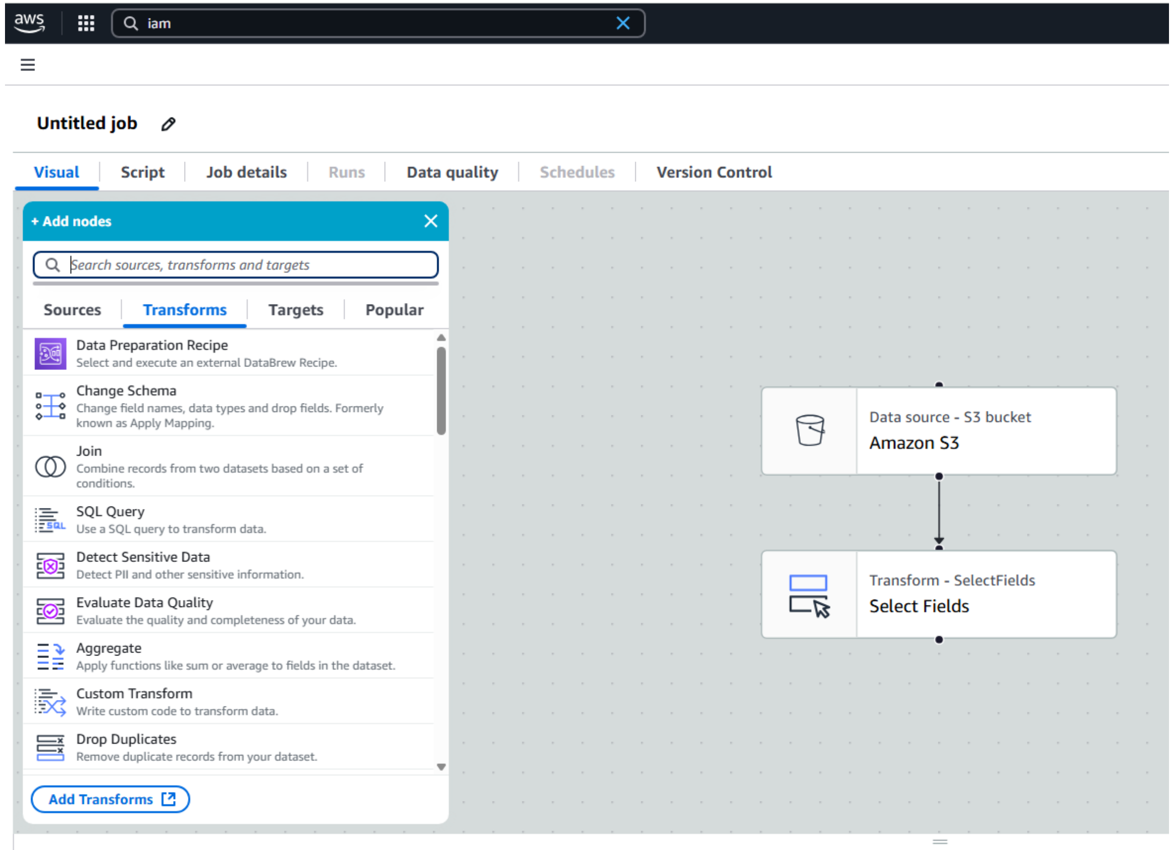
Workflows:

- Define a sequence of jobs and crawlers with dependencies to automate complex ETL processes.

AWS Glue Studio

Features:

- User-friendly interface to create, run, and monitor ETL jobs without writing code.
- Drag-and-drop functionality to design data flows.
- Automatic code generation with the option to customize scripts.



Data Preparation Recipe
Select and execute an external DataBrew Recipe.

Change Schema
Change field names, data types and drop fields. Formerly known as Apply Mapping.

Join
Combine records from two datasets based on a set of conditions.

SQL Query
Use a SQL query to transform data.

Detect Sensitive Data
Detect PII and other sensitive information.

Evaluate Data Quality
Evaluate the quality and completeness of your data.

Aggregate
Apply functions like sum or average to fields in the dataset.

Custom Transform
Write custom code to transform data.

Drop Duplicates
Remove duplicate records from your dataset.

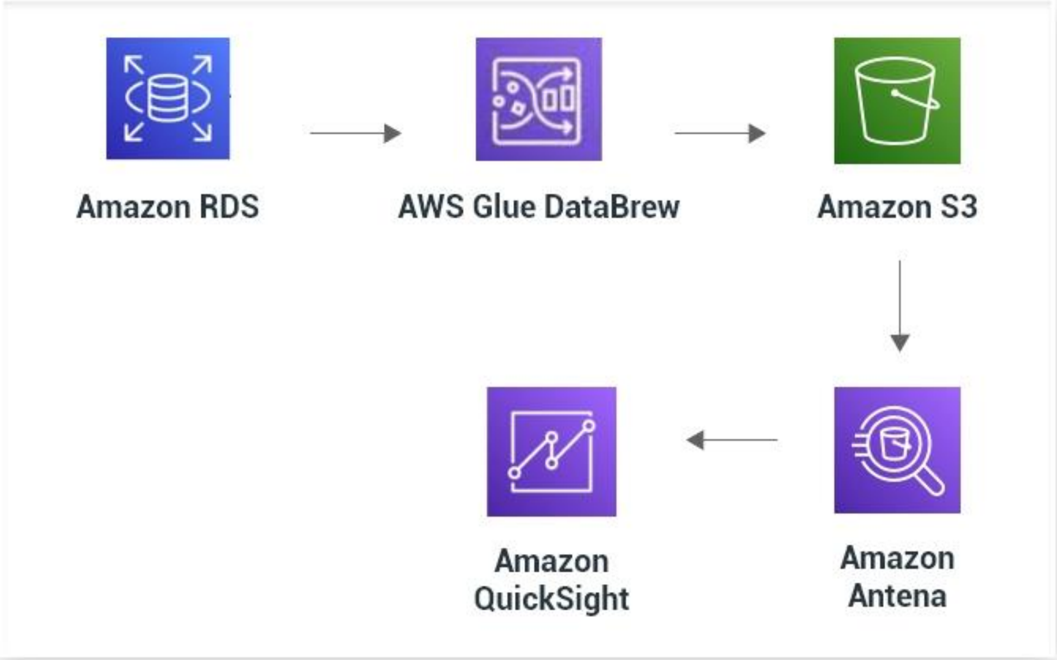
AWS Glue DataBrew

Overview:

- Visual data preparation tool for data analysts and scientists.

Capabilities:

- Clean and normalize data without writing code.
- Apply transformations, handle missing values, and create data profiles.



Module 4 Lab: Querying Data Using Amazon Athena

In this lab, you will learn how to query CSV data stored in Amazon S3 using **Amazon Athena** and **AWS Glue**.

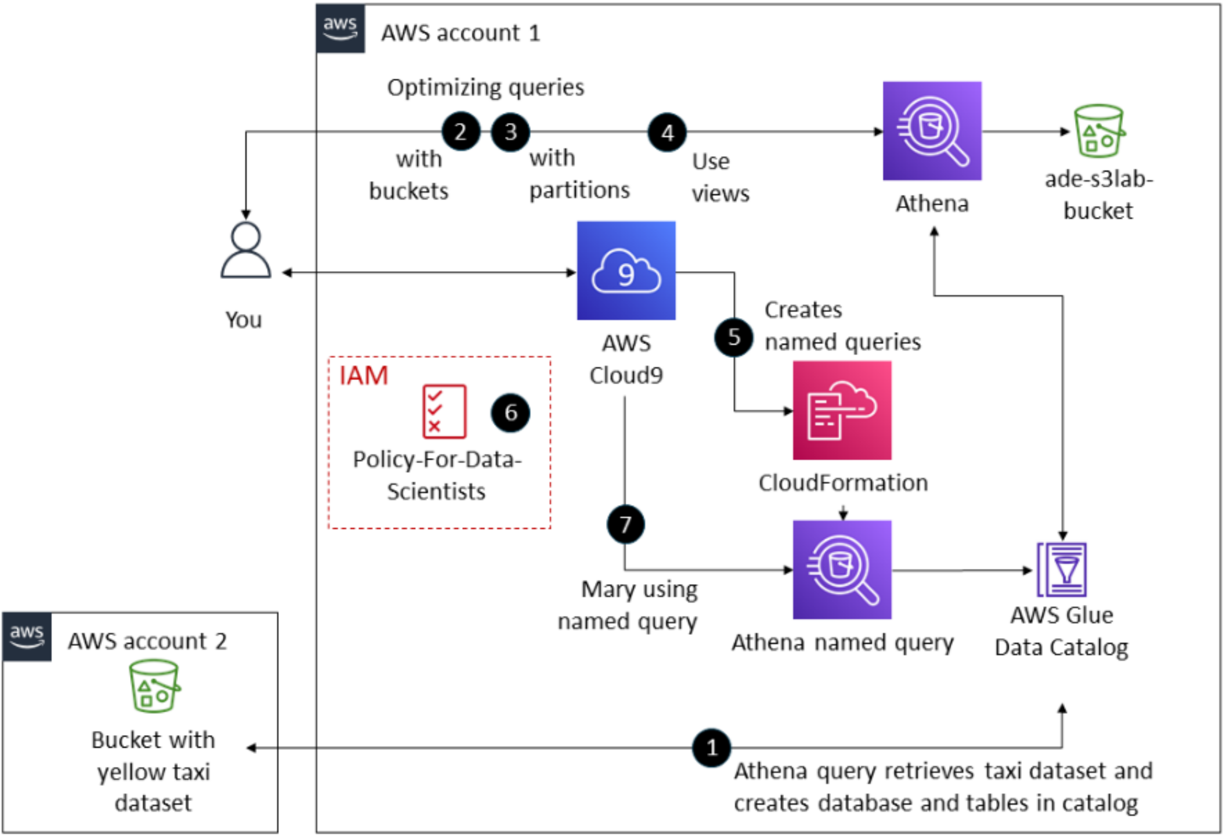
Athena enables SQL queries on S3 data without moving it, while Glue helps define and manage metadata.

You will implement a lab in AWS Academy Data Engineering Course (Module 4) where we simulate the role of a data scientist building a scalable, permission-controlled solution that supports team-wide access following the principle of least privilege.

Objectives

You will:

- Create a Glue database and table via the Athena editor
- Define schema and use bulk column features
- Query and optimize data in S3 with Athena
- Create views and named queries
- Deploy named queries using CloudFormation
- Validate IAM access through AWS CLI
- This training uses a 2017 taxi trip dataset and is restricted to required AWS services only.



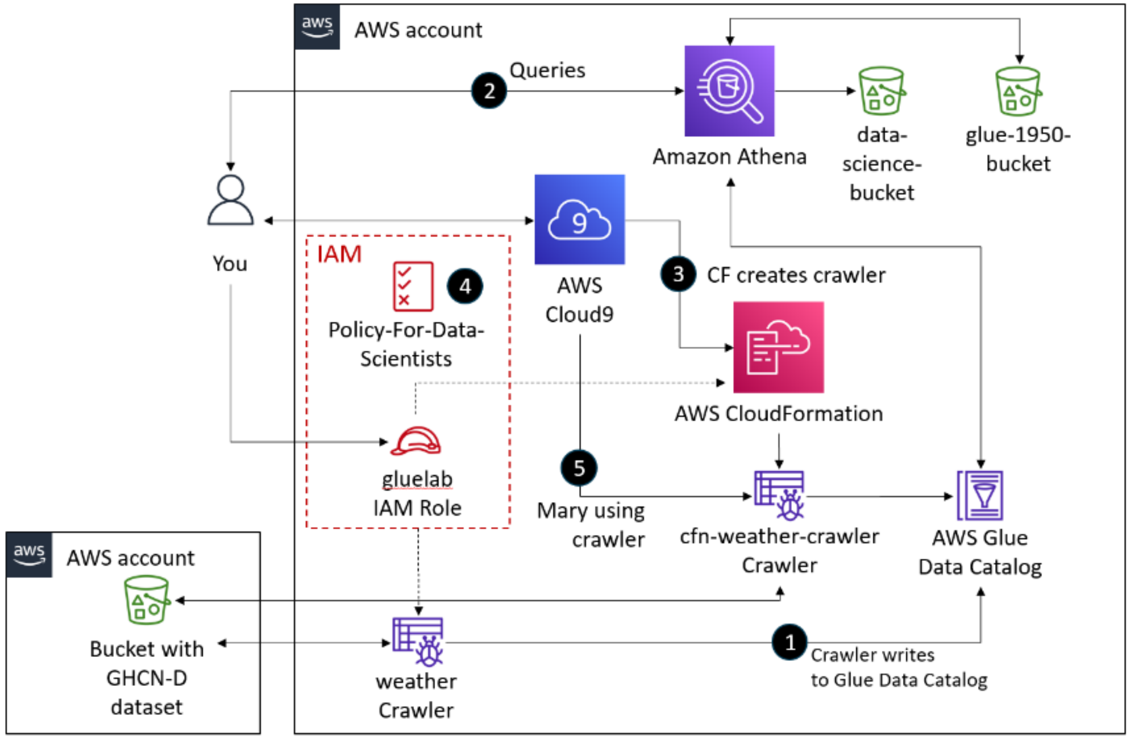
Module 7 Lab: Performing ETL on a Dataset by Using AWS Glue

In this lab, you will learn how to use AWS Glue to import a dataset from Amazon Simple Storage Service (**Amazon S3**).

You will then extract the data, transform its schema, and load the dataset into an **AWS Glue** database for later analysis by using Athena.

After completing this lab, you will be able to do the following:

- Access AWS Glue in the AWS Management Console and create a **crawler**.
- Create an AWS Glue database with tables and a schema by using a crawler.
- Query data in the AWS Glue database by using Athena.
- Create and deploy an AWS Glue crawler by using an AWS CloudFormation template.
- Review an AWS Identity and Access Management (IAM) policy for users to run an AWS Glue crawler and query an AWS Glue database in Athena.
- Confirm that a user with the IAM policy can use the AWS Command Line Interface (AWS CLI) to access the AWS Glue database that the crawler created.
- Confirm that a user can run the AWS Glue crawler when source data changes.



amazon
REDSHIFT

Amazon Redshift

Amazon Redshift is a fully managed data warehouse service offered by Amazon Web Services (AWS).

It is designed for large-scale data storage and fast query performance, making it suitable for data analytics and business intelligence workloads.

Here are some key features and benefits of Amazon Redshift:

Key Features and Benefits:

- **High Performance**:
  - Redshift uses columnar storage and data compression to reduce the amount of data read from disk, improving query performance.
  - Massively parallel processing (MPP) architecture allows Redshift to distribute and parallelize queries across multiple nodes.
- **Scalability**:
  - Redshift allows you to start with a small cluster and scale up to petabytes of data.
  - You can easily resize your Redshift cluster by adding or removing nodes.

Amazon Redshift

- **Cost-Effective**:
  - Pay only for the resources you use with on-demand pricing.
  - Reserved instance pricing and the ability to pause and resume clusters can further reduce costs.
- **Managed Service**:
  - AWS manages the setup, operation, and scaling of Redshift, including backups, patching, and monitoring.
  - Automatic backups and snapshots provide data protection and recovery.
- **Data Integration**:
  - Redshift integrates with various AWS services such as Amazon S3, Amazon RDS, Amazon DynamoDB, and AWS Glue for data ingestion and ETL processes.
  - Supports integration with third-party ETL tools and BI platforms.
- **Advanced Analytics**:
  - Supports SQL-based querying and analytics.
  - Redshift Spectrum allows you to run queries on data stored in Amazon S3 without loading it into Redshift.
  - Machine learning integration with Amazon SageMaker for predictive analytics.

Core Components of Amazon Redshift

1. Clusters:

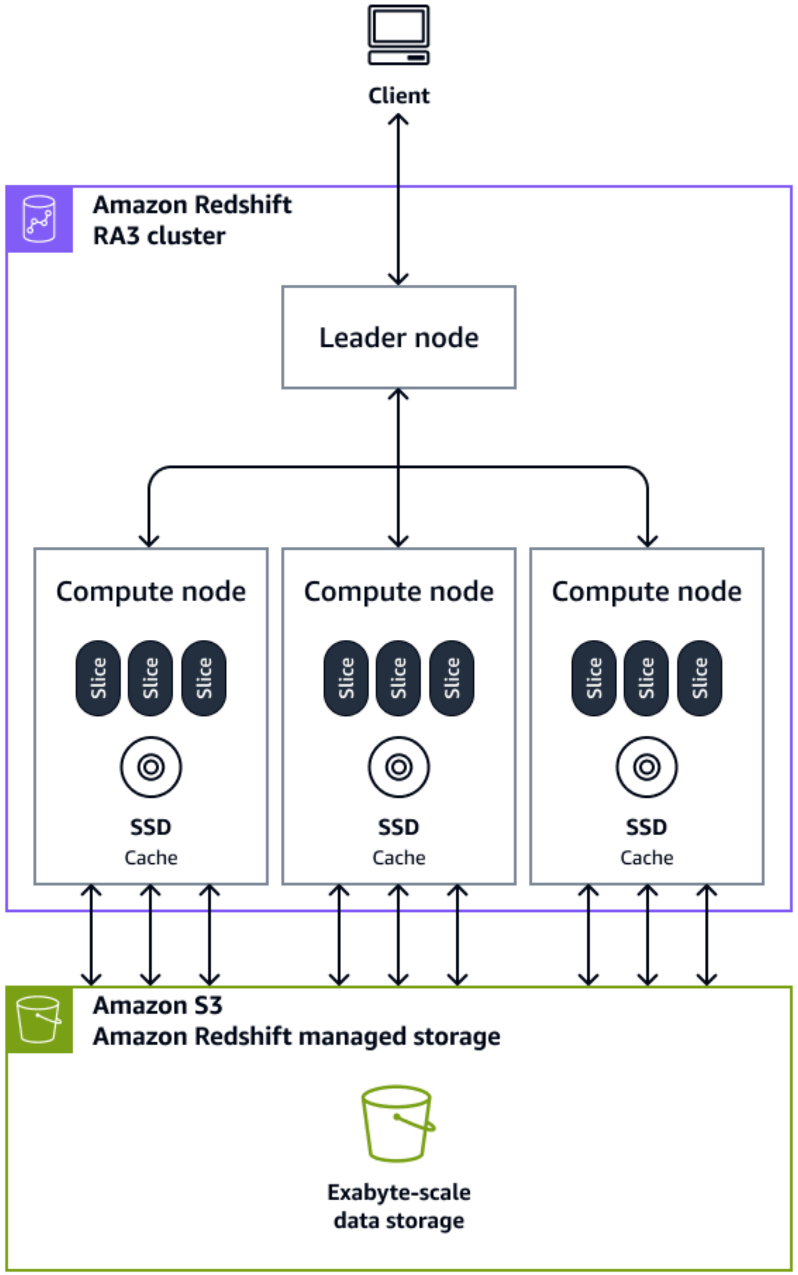
- The core infrastructure component, composed of one or more compute nodes.

2. Leader Node:

- Manages communications with client programs and all communication with compute nodes.

3. Compute Nodes:

- Handle data processing tasks and store data.



Query Processing and Optimization

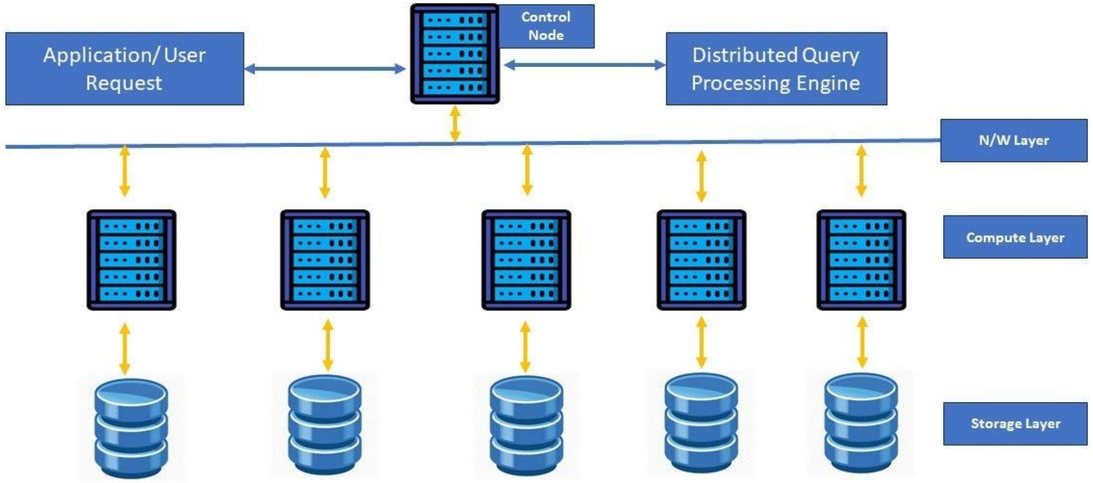
Massively Parallel Processing (MPP):

- Distributes data and query load across multiple nodes for efficient processing.

Query Optimization:

- Redshift's query optimizer evaluates multiple execution strategies and selects the most efficient one.

MPP Architecture


Control Node
Application/ User Request
Distributed Query Processing Engine
N/W Layer
Compute Layer
Storage Layer

Redshift Deployment Options

1. Provisioned Clusters

With provisioned clusters, you manage your Amazon Redshift resources manually.

This option gives you complete control over cluster sizing, scaling, and maintenance windows.

2. Redshift Serverless

Amazon Redshift Serverless lets you access and analyze data without the configurations of a provisioned data warehouse.

Resources are automatically provisioned, and capacity is intelligently scaled to deliver fast performance for demanding workloads.

Key benefits of Redshift Serverless:

- i. **Automatic resource provisioning**: No need to manually configure cluster size
- ii. **Pay-per-use pricing**: You don't incur charges when the data warehouse is idle
- iii. **Auto-scaling**: Capacity adjusts based on workload demands
- iv. **Simplified management**: No need to monitor or tune the cluster

Redshift Serverless organizes resources using namespaces (for database objects) and workgroups (for compute resources).

Workgroups house compute resources like Redshift Processing Units (RPUs), security groups, and usage limits.

Use Cases

Data Warehousing:

- Centralized repository for structured and semi-structured data.

Business Intelligence:

- Supports integration with BI tools like Tableau, Looker, and Amazon QuickSight.

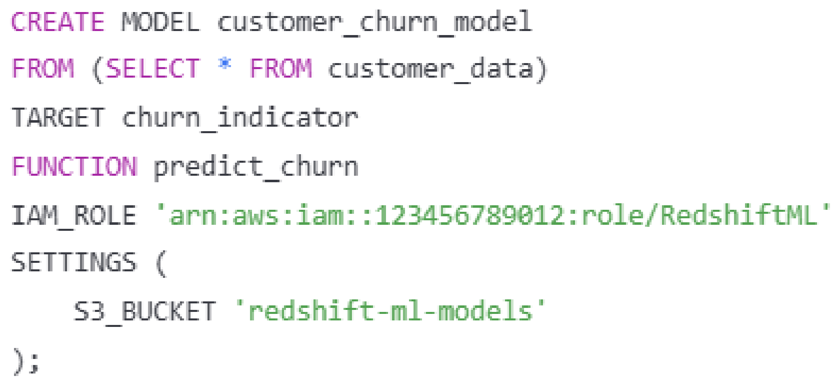
Machine Learning:

- Provides data for training and inference in machine learning models.

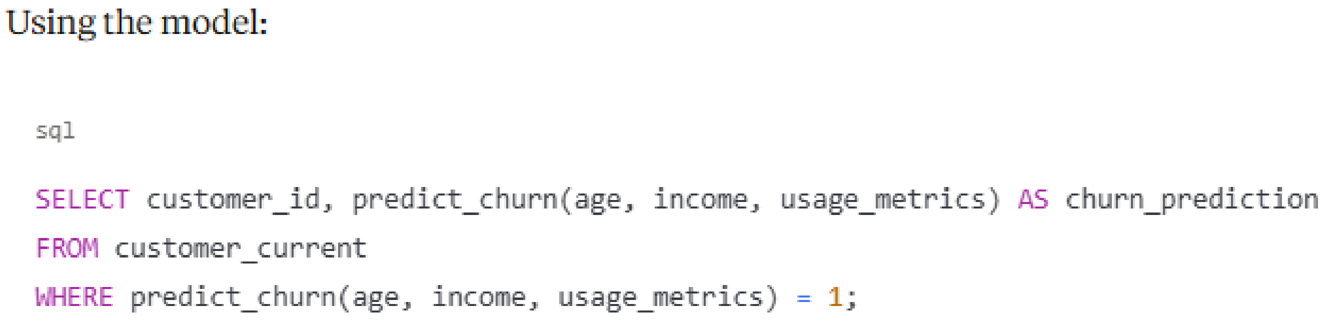
Redshift ML

- Amazon Redshift ML makes it easy for data analysts and database developers to create, train, and apply machine learning models using familiar SQL commands.
- It leverages Amazon SageMaker without requiring users to learn new tools or languages.

Create Customer Churn Model



```
CREATE MODEL customer_churn_model
FROM (SELECT * FROM customer_data)
TARGET churn_indicator
FUNCTION predict_churn
IAM_ROLE 'arn:aws:iam::123456789012:role/RedshiftML'
SETTINGS (
    S3_BUCKET 'redshift-ml-models'
);
```



Using the model:

```sql
SELECT customer_id, predict_churn(age, income, usage_metrics) AS churn_prediction
FROM customer_current
WHERE predict_churn(age, income, usage_metrics) = 1;
```

Setting Up AWS Redshift

Setting Up a Provisioned Cluster

1. **Sign in to AWS Console**: Navigate to the Redshift service
2. **Create Cluster**: Choose "Create cluster" and configure settings:
    1. Cluster identifier
    2. Node type
    3. Number of nodes
    4. Database configurations (name, port, credentials)
    5. Network and security settings
    6. Maintenance and backup options
3. **Launch Cluster**: Review settings and create the cluster

----

Setting Up Redshift Serverless

1. **Sign in to AWS Console**: Navigate to the Redshift service
2. **Create Serverless Workspace**: Configure settings:
    1. Namespace name (for database objects)
    2. Workgroup name (for compute resources)
    3. Network and security settings
    4. RPU capacity settings (optional)
3. **Create Workspace**: Review settings and create the serverless workspace

amazon EMR

Introduction to Amazon EMR

What is Amazon EMR?

- Amazon EMR (Elastic MapReduce) is a managed cloud-native **Big Data Platform** provided by Amazon Web Services (AWS).
- It simplifies running big data frameworks, such as **Apache Hadoop** and **Apache Spark**, on AWS to process and analyze vast amounts of data.

Key Features:

- **Scalability:** Handles large-scale data sets, allowing dynamic scaling of resources based on workload demands.
- **Performance:** Utilizes distributed processing frameworks to decrease command execution time.
- **Simplified Management**: AWS handles provisioning, configuration, and tuning of clusters
- **Flexibility**: Supports multiple deployment options and various open-source big data frameworks
- **Integration:** Seamlessly integrates with various AWS services like Amazon S3, Amazon RDS, and Amazon DynamoDB.
- **Security**: Integrates with AWS security services for comprehensive protection

Use Cases

Use Case	Description	Tools/Frameworks	Example
Big Data Processing (ETL)	Transform raw data into usable formats	Apache Spark, Hadoop MapReduce	Daily ETL jobs to process clickstream data and store it in S3
Data Warehousing & Analytics	Perform complex queries on large datasets	Hive, Presto, Trino	Ad-hoc analysis of petabytes of user logs for BI reports
Machine Learning	Train ML models on distributed data	Spark MLlib, integration with SageMaker	Recommendation engine on product-user interaction data
Real-time Stream Processing	Ingest and analyze real-time data streams	Spark Streaming, Apache Flink	Fraud detection on real-time transaction streams
Log and Event Analysis	Analyze system logs for monitoring and security	Hadoop, Spark, Hive	Parsing application logs for error pattern detection
Genomics & Scientific Research	Process large scientific datasets	Custom Hadoop/Spark applications	Genome sequencing analysis across distributed nodes
Graph Processing	Perform graph-based computations	Apache Giraph, Spark GraphX	Social network analysis for influencer detection
Web Indexing & Crawling	Process and index web content	Hadoop, custom MapReduce	Large-scale web crawling and search index creation
Data Lake Analytics	Query data directly from Amazon S3 without loading into a database	Hive, Presto, Trino	Analyzing semi-structured data in an S3-based data lake
Financial Risk Modeling	Run complex financial simulations	Spark, Scala/Python	Monte Carlo simulations for value-at-risk (VaR) analysis

EMR Architecture Overview

Core Components

AWS EMR architecture consists of several layers, each providing specific functionality to the cluster:

1. **Storage Layer**
   1. Amazon S3 (primary data store)
   2. HDFS (Hadoop Distributed File System)
   3. EMRFS (EMR File System)
   4. Local instance storage
2. **Cluster Layer**
   1. **Primary Node (formerly Master Node)**: Manages the cluster and coordinates distribution of data and tasks
   2. **Core Nodes**: Run tasks and store data in HDFS
   3. **Task Nodes**: Run tasks but don't store data
3. **Processing Layer**
   1. Open-source frameworks like Hadoop, Spark, Hive, and others
   2. EMR-optimized versions of these frameworks

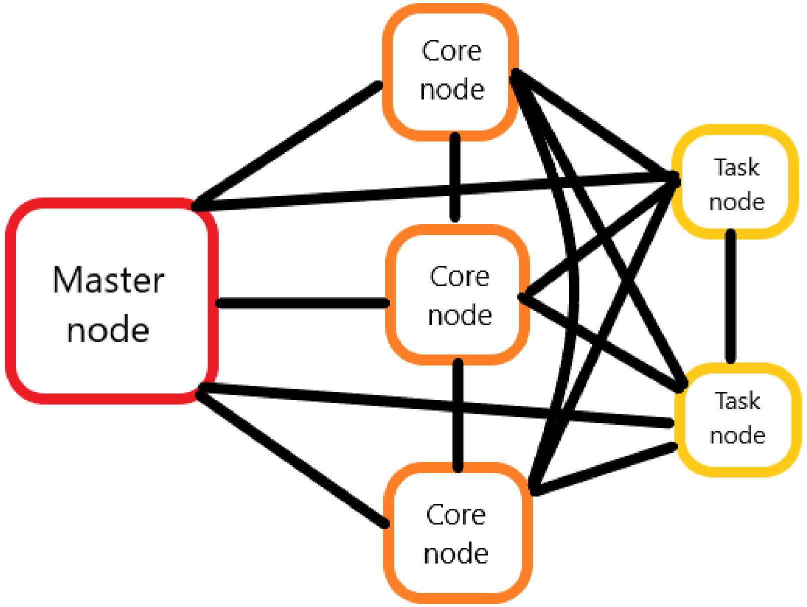
Core Components of Amazon EMR

1. Clusters:

- The central component of Amazon EMR, composed of Amazon EC2 instances.
- Each instance in the cluster is called a **Node**.

2. Node Types:

- **Primary Node:** Manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing.
- **Core Node:** Runs tasks and stores data in the Hadoop Distributed File System (HDFS) on your cluster.
- **Task Node:** Runs tasks but does not store data in HDFS. Task nodes are optional.



Data Storage and Processing

Storage Options:

- **Amazon S3:** EMR File System (EMRFS) allows EMR clusters to directly access data stored in Amazon S3 as if it were a file system like HDFS.
- **HDFS:** A distributed and scalable Hadoop file system enabling data distribution across instances in the cluster.

Processing Frameworks:

- **Apache Hadoop MapReduce:** An open-source programming model for distributed computing.
- **Apache Spark:** A cluster framework and programming model for processing big data workloads, supporting multiple interactive query modules such as SparkSQL

EMR Studio and Notebooks

EMR Studio:

- An integrated development environment (IDE) that makes it easy for data scientists and data engineers to develop, visualize, and debug data engineering and data science applications written in R, Python, Scala, and PySpark.

EMR Notebooks:

- Provides a managed environment based on Jupyter Notebooks for interactive data analysis and visualization.

Deployment Options

AWS EMR offers different deployment options to meet different operational requirements:

1. EMR on EC2

The traditional deployment model where EMR runs on Amazon EC2 instances.

Key features:

- Full control over cluster configuration and instance selection
- Customizable AMIs (Amazon Machine Images)
- Ability to install custom software
- Ideal for workloads requiring specific hardware configurations

Use when:

- You need maximum control and flexibility
- You want to customize underlying infrastructure
- You need specific EC2 instance types for performance optimization

Deployment Options

2. **A serverless deployment** option that eliminates the need to configure, manage, or scale clusters.

Key features:

- Automatic resource provisioning and scaling
- Pay only for resources used by your applications
- Zero infrastructure management
- Currently supports Apache Spark and Apache Hive

Use when:

- You want to simplify operations
- You have variable or unpredictable workloads
- You want to avoid managing clusters
- You prefer focusing on applications rather than infrastructure

Data Processing Frameworks

EMR supports a variety of open-source frameworks for data processing. The latest EMR releases include:

Apache Hadoop

The foundation framework providing:

- HDFS (Hadoop Distributed File System)
- YARN (resource management)
- MapReduce (programming model)

Apache Spark

A unified analytics engine offering:

- In-memory processing
- Support for SQL, streaming, machine learning, and graph processing
- Performance up to 100x faster than traditional MapReduce

Apache Hive

Data warehouse software providing:

- SQL-like interface (HiveQL)
- Structured data querying capabilities
- Integration with various storage system

Data Processing Frameworks

Presto and Trino

Distributed SQL query engines for:

- High-performance interactive analytics
- Querying data across multiple sources

Apache Flink

A stream processing framework offering:

- Real-time data processing
- Event-time processing
- State management

HBase

A NoSQL database providing:

- Random, real-time read/write access
- Storage for large tables with billions of rows

Data Processing Patterns

1. **Batch Processing**:
    1. Processing large volumes of data periodically
    2. ETL jobs for data warehousing
2. **Interactive Analysis**:
    1. SQL queries with Hive, Presto, or Spark SQL
    2. Ad-hoc analysis in notebooks
3. **Streaming Processing**:
    1. Real-time data processing with Spark Streaming or Flink
    2. Processing data from Kinesis or Kafka
4. **Machine Learning**:
    1. Building and deploying models with Spark MLlib
    2. Feature engineering and model scoring

Setting Up AWS EMR

Setting Up EMR on EC2

1. Sign in to AWS Console: Navigate to the EMR service

2. Create Cluster: Choose "Create cluster" and configure:

1. Cluster name
2. Release version (applications and versions)
3. Instance types and count
4. Networking settings
5. Security configurations
6. Bootstrap actions (optional)

3. Launch Cluster: Review settings and create

Setting Up EMR Serverless

1. Sign in to AWS Console: Navigate to the EMR service

2. Create Application:

1. Choose application type (Spark or Hive)
2. Select release version
3. Configure capacity settings (optional)
4. Set up network and security options

3. Submit Jobs: Use the EMR Studio, AWS CLI, or SDK to submit jobs

Hands-On Exercises

Exercise 1: Setting Up an EMR Cluster

1. Launch a basic EMR cluster using the AWS Management Console
2. Connect to the cluster using SSH
3. Explore the cluster web interfaces (YARN, Spark, etc.)
4. Submit a simple job and monitor its execution

Exercise 2: Processing Data with Spark

1. Prepare sample dataset in S3
2. Create a Spark application to process the data
3. Submit the application to EMR
4. Analyze the results and optimize the application

Exercise 3: EMR Serverless Deployment

1. Create an EMR Serverless application
2. Configure pre-initialized capacity
3. Submit Spark or Hive jobs
4. Monitor job execution and resource usage

Exercise 4: Security Implementation

1. Create a security configuration with encryption
2. Set up Kerberos authentication
3. Implement Lake Formation integration
4. Test fine-grained access controls

Exercise 5: Performance Optimization

1. Benchmark a standard EMR configuration
2. Apply performance optimization techniques
3. Measure the impact of optimizations
4. Develop a performance tuning checklist

Lab: Processing Logs by Using Amazon EMR

In this lab, you will use Amazon EMR to process logs that are stored in an S3 bucket.

To process the data in Amazon EMR, you will use Apache Hive.

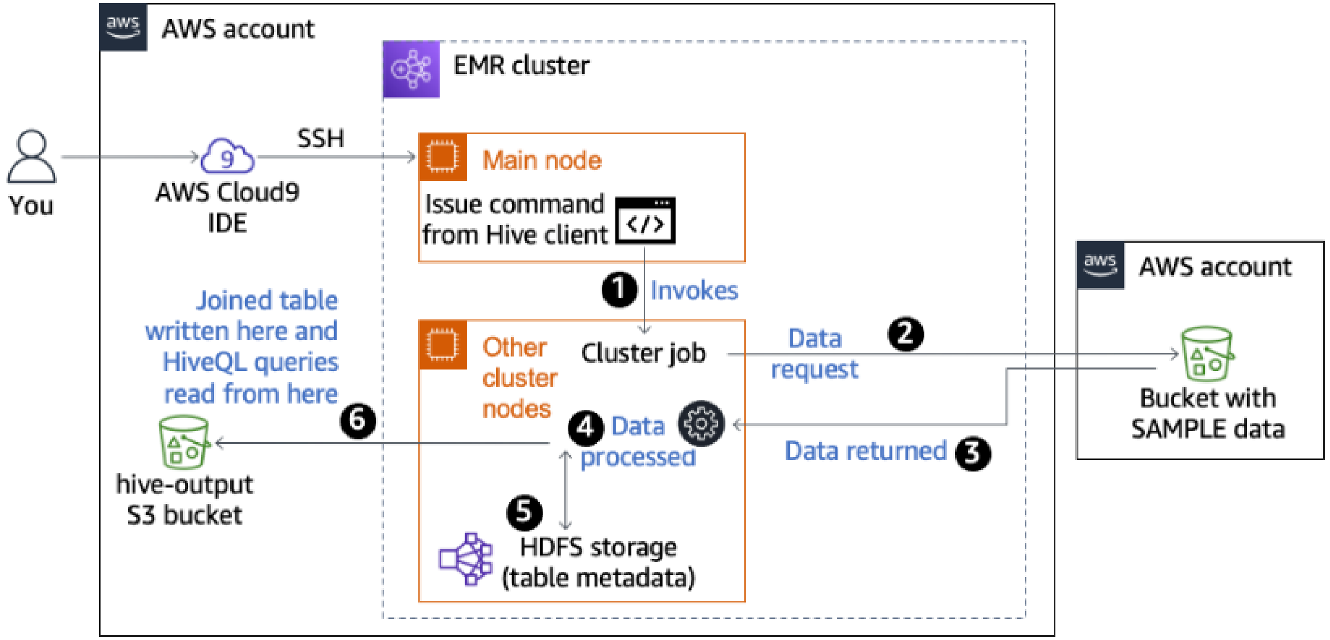
Hive is an open-source data warehouse infrastructure framework that is built on top of Hadoop and is often used to process large datasets.

With Hive, you can query large datasets with SQL-like queries.

When you run a Hive query, the request is translated into a Yet Another Resource Negotiator (YARN) job. YARN is the part of the Hadoop software that coordinates job scheduling.

After completing this lab, you should be able to do the following:

- Launch an EMR cluster through the AWS Management Console.
- Run Hive interactively.
- Use Hive commands to create tables from log data that is stored in Amazon S3.
- Use Hive commands to join tables and store the joined table in Amazon S3.
- Use Hive to query tables that are stored in Amazon S3.



amazon
Kinesis

What is Amazon Kinesis?

AWS Kinesis is a suite of services designed to collect, process, and analyze real-time, streaming data at massive scale.

This means you can get timely insights and react quickly to new information.

Why use Kinesis?

Imagine you need to:

- Analyze clickstreams from your website in real-time to personalize user experience.
- Process IoT sensor data from thousands of devices.
- Ingest application logs for immediate anomaly detection.
- Feed data into real-time dashboards or machine learning models.
- Kinesis provides the building blocks for these scenarios.

Core Kinesis Services

Kinesis is not a single service but a family, the main ones are:

1. **Kinesis Data Streams (KDS):** The foundational service. It's a massively scalable and durable real-time data streaming service. You build custom applications to process or analyze streaming data.
2. **Kinesis Data Firehose:** The easiest way to reliably load streaming data into data lakes, data stores, and analytics services. It's more about *delivery* than custom processing.
3. **Kinesis Data Analytics:** Allows you to process and analyze streaming data using standard SQL or Apache Flink. It can read from Data Streams or Firehose.
4. **Kinesis Video Streams:** Securely streams video from connected devices to AWS for analytics, machine learning (ML), playback, and other processing.

Kinesis Data Streams (KDS)

Concept: Think of a Kinesis Data Stream as a highly available, durable pipe for your data. Data flows into the stream as **records**.

Key Terminology:

- **Data Record:** The unit of data stored in a Kinesis stream. It consists of a sequence number, a partition key, and a data blob (up to 1MB).
- **Producer:** An application that puts data records into a Kinesis Data Stream (e.g., web servers, IoT devices, mobile apps).
- **Consumer:** An application that gets records from a Kinesis Data Stream and processes them (e.g., an EC2 instance, Lambda function, Kinesis Data Analytics application).
- **Shard:** The base throughput unit of a Kinesis Data Stream.
  - Each shard provides a capacity of 1MB/second data input and 2MB/second data output.
  - One shard can support up to 1000 PUT records per second.
  - You specify the number of shards when you create a stream. You can scale the number of shards up or down.
  - **Ordering is guaranteed within a shard.**
- **Partition Key:** Used to group data by shard within a stream. Kinesis uses the partition key to segregate and route records to different shards. Records with the same partition key go to the same shard. This is crucial for ordered processing of related records.
- **Sequence Number:** A unique identifier for each record within its shard. Kinesis assigns it after you write the record.

Kinesis Data Streams (KDS)

How it Works:

1. **Producers** (e.g., your application server, an IoT device) send data records to your Kinesis Data Stream using the AWS SDK, Kinesis Producer Library (KPL), or Kinesis Agent.
2. When sending a record, you specify the stream name, a **partition key**, and the data itself.
3. Kinesis hashes the partition key and maps it to a specific **shard**.
4. Data is stored in shards for a configurable **retention period** (default 24 hours, max 365 days).
5. **Consumers** (e.g., EC2 instances running custom code, AWS Lambda functions, Kinesis Data Analytics) read data from the shards.
   i. **Shared Throughput Consumers (Classic):** Each consumer gets a fraction of the shard's 2MB/s output. Max 5 consumers per shard this way.
   ii. **Enhanced Fan-Out Consumers:** Each consumer gets its own dedicated 2MB/s throughput per shard, allowing many consumers to read from the same stream without impacting each other.

When to Use KDS:

- You need custom real-time data processing.
- You require low latency (sub-second).
- You need ordered processing for specific keys.
- You want to feed data into multiple real-time applications.
- Building real-time dashboards, anomaly detection, real-time metrics.

Kinesis Data Firehose

Concept: Firehose is all about *delivering* streaming data to a destination. It's a fully managed service that requires minimal to no coding for basic delivery.

Key Terminology:

- **Delivery Stream:** The Firehose entity you create.
- **Source:** Where Firehose gets the data (e.g., Kinesis Data Streams, direct PUT from producers, CloudWatch Logs, Events).
- **Destinations:** Where Firehose sends the data:
  - Amazon S3
  - Amazon Redshift (often via S3)
  - Amazon Elasticsearch Service (now Amazon OpenSearch Service)
  - Splunk, a real-time search and analysis engine to quickly and easily search through large volumes of log data
  - Custom HTTP endpoints or supported third-party partner destinations.
- **Data Transformation (Optional):** You can use an AWS Lambda function to transform incoming source data before delivering it (e.g., convert JSON to CSV, filter records).
- **Buffering:** Firehose buffers incoming data (by size or time) before delivering it to the destination to optimize for batch writes.

Kinesis Data Firehose

How it Works:

1. Producers send data to a Firehose Delivery Stream (or Firehose reads from a Kinesis Data Stream).
2. Firehose buffers the data.
3. (Optional) Firehose invokes your Lambda function for data transformation.
4. Firehose delivers the (transformed) data to the configured destination in batches.
5. It handles retries, scaling, and monitoring automatically.

When to Use Firehose:

- You need to load streaming data into S3, Redshift, OpenSearch, Splunk, or HTTP endpoints with minimal effort.
- You want a managed service that handles scaling, retries, and batching.
- Simple transformations (via Lambda) are sufficient.
- You don't need sub-second processing latency for the *delivery* part (buffering introduces some latency).

KDS vs. Firehose:

- **KDS:** For real-time *processing* with custom consumer applications. You manage consumers.
- **Firehose:** For real-time *delivery* to destinations. AWS manages delivery.
- You can use them together: KDS for initial ingestion and fan-out to multiple consumers, one of which could be Firehose for archiving raw data to S3.

Kinesis Data Analytics

Concept: Process and analyze streaming data in real-time using standard SQL or Apache Flink.

Two Flavors:

- **Kinesis Data Analytics for SQL Applications:**
  - Write standard SQL queries on your streaming data.
  - Define a schema for your incoming data.
  - Use concepts like tumbling windows, sliding windows, and anomaly detection functions.
  - Output results to another Kinesis Data Stream, a Firehose delivery stream, or a Lambda function.
- **Kinesis Data Analytics for Apache Flink:**
  - Build sophisticated stream processing applications using Java, Scala, or Python with the Apache Flink framework.
  - Offers more flexibility and power than SQL for complex event processing, stateful computations, etc.
  - Fully managed Flink environment.

Kinesis Data Analytics

How it Works:

1. Define an input: a Kinesis Data Stream or a Kinesis Data Firehose delivery stream.
2. Write your SQL query or Flink application code.
3. (SQL) Kinesis Data Analytics infers a schema or you can define it.
4. (SQL) Your SQL query runs continuously on the incoming data.
5. Define an output (destination): another KDS, Firehose, or Lambda.
6. The service continuously reads from the source, processes data, and sends results to the destination.

When to Use Kinesis Data Analytics:

- You need real-time analytics (e.g., aggregations, filtering, time-series analysis) on streaming data.
- You prefer using SQL for its simplicity for common analytics tasks.
- You need the power of Apache Flink for complex stream processing without managing Flink clusters yourself.
- Real-time dashboards, alerting, generating metrics for leaderboards.

AWS Kinesis Video Streams

Kinesis Video Streams makes it easy to securely stream video from millions of connected devices to AWS for analytics, machine learning, playback, and other processing.

It can ingest data from cameras, smartphones, security cameras, and other video sources.

How it Works:

- **Producers:** Devices (e.g., cameras with the Kinesis Video Streams Producer SDK) or applications that send video data to a Kinesis video stream.
- **Video Stream:** A resource that transports and stores video data. Each stream can contain multiple fragments.
- **Fragment:** A self-contained sequence of media data, typically a few seconds long.
- **Consumers:** Applications that retrieve and process the video data. This can include:
  - Live or on-demand playback using HLS or DASH.
  - Integration with Amazon Rekognition Video for real-time video analysis (e.g., object detection, face recognition).
  - Custom ML models for video analytics.
  - Saving processed video or clips to S3.
- **WebRTC:** Kinesis Video Streams also provides fully managed WebRTC capabilities for real-time, two-way media streaming between web browsers, mobile applications, and connected devices.

Use Cases:

- Live video streaming for events or monitoring.
- Smart home camera solutions.
- Industrial monitoring with video analytics.
- Dashcam video storage and analysis.
- Real-time baby monitors.
- Two-way video conferencing using WebRTC.

Comparison Table

Feature	Data Streams	Firehose	Analytics
Use Case	Custom processing	Easy delivery	Real-time analytics
Scalability	Manual shard management	Auto-scaling	Scales with input
Processing	Application/consumer	No processing	SQL/Flink
Latency	Low (ms)	Higher (seconds)	Low
Destination	Custom	S3, Redshift, etc.	KDS, KDF

Real-World Applications

E-Commerce:

- Real-time inventory tracking
- Customer behavior analysis
- Dynamic pricing strategies
- Fraud detection

Media & Entertainment:

- Viewer engagement metrics
- Content recommendation
- Ad performance tracking
- Live event monitoring

IoT & Manufacturing:

- Sensor data processing
- Predictive maintenance
- Quality control
- Supply chain optimization

Financial Services:

- Transaction monitoring
- Risk assessment
- Compliance reporting
- Market analysis

Some of the Main Big Data Providers on the Cloud

Feature/Service Category	Amazon AWS	Google Cloud Platform	Microsoft Azure	Cloudera Data Platform (CDP)
Object Storage	Amazon S3	Google Cloud Storage	Azure Blob Storage	Cloudera Data Lake Storage (S3, ADLS, GCS)
Data Warehouse	Amazon Redshift	BigQuery	Azure Synapse Analytics (SQL Data Warehouse)	Cloudera Data Warehouse (Impala, Hive LLAP)
Batch Data Processing	Amazon EMR	Google Cloud Dataproc	Azure HDInsight	Cloudera Data Engineering (Apache Spark)
Stream Data Processing	Amazon Kinesis	Google Cloud Dataflow	Azure Stream Analytics	Cloudera Streams Messaging (Apache Kafka)
ETL/Data Integration	AWS Glue	Google Cloud Data Fusion	Azure Data Factory	Cloudera DataFlow (Apache NiFi)
Real-time Messaging	Amazon Kinesis Data Streams	Google Cloud Pub/Sub	Azure Event Hubs	Cloudera Streams Messaging (Apache Kafka)
Serverless Query Service	Amazon Athena	Google BigQuery	Azure Synapse Analytics (Serverless)	Cloudera Data Warehouse (Impala, Hive LLAP)
Data Lake Storage	Amazon S3 (with AWS Lake Formation)	Google Cloud Storage (with Dataproc)	Azure Data Lake Storage	Cloudera Data Lake Storage (S3, ADLS, GCS)
Big Data Analytics	AWS Glue/Athena	Google Cloud Dataflow/BigQuery	Azure Synapse Analytics/HDInsight	Cloudera Data Engineering, Data Science Workbench
Managed Apache Spark	Amazon EMR	Google Cloud Dataproc	Azure Databricks	Cloudera Data Engineering
NoSQL Database	Amazon DynamoDB	Google Cloud Bigtable	Azure Cosmos DB	Cloudera Operational Database (HBase, Kudu)
Machine Learning	Amazon SageMaker	Google AI Platform	Azure Machine Learning	Cloudera Machine Learning (CDSW)
Interactive Analysis	Amazon QuickSight	Looker	Power BI	Cloudera Data Warehouse, Hue
Data Exploration	Amazon Elasticsearch Service (OpenSearch)	Google Cloud Dataflow/Data Studio	Azure Data Explorer	Cloudera Data Science Workbench
Workflow Orchestration	AWS Step Functions	Google Cloud Composer	Azure Data Factory	Cloudera DataFlow (Apache NiFi)
Serverless Computing	AWS Lambda	Google Cloud Functions	Azure Functions	Cloudera Data Engineering (serverless Spark)
Managed Database	Amazon RDS	Google Cloud SQL	Azure SQL Database	Cloudera Operational Database (HBase, Kudu)
Pre-built AI Services	AWS AI Services (Rekognition, Polly, etc.)	Google Cloud AI (Vision, Speech, etc.)	Azure Cognitive Services	Cloudera Machine Learning (CDSW)