AWS Services Used – Detailed Explanation (Student-Friendly)

This document explains **each AWS service used in the mini project**, why it was required, how it works internally, and how it was used specifically in our implementation. This is written for **clear understanding, viva preparation, and theory exams**.

---

1. Amazon S3 (Simple Storage Service)

What is Amazon S3?

Amazon S3 is an **object-based cloud storage service** provided by AWS. It allows users to store and retrieve any amount of data from anywhere over the internet.

S3 stores data in the form of **objects**, and these objects are stored inside **buckets**.

Why did we use Amazon S3 in this project?

In big data systems, storage must be: - Scalable - Cost-effective - Durable - Easy to integrate with processing tools

Amazon S3 satisfies all these requirements and is also available in the **AWS Free Tier**, making it ideal for student projects.

How S3 was used in our project

We used Amazon S3 as a **Data Lake**:

1. **Raw Data Bucket**
2. Stores raw sales data in JSON format
3. Data is uploaded from a local Python script
4. Acts as the input source for AWS Glue
5. **Processed Data Bucket**
6. Stores transformed data in Parquet format
7. Used by Amazon Athena for analytics

Key Concepts Used

- Buckets
- Objects
- Folder-like prefixes
- Schema-on-read

---

2. AWS IAM (Identity and Access Management)

What is AWS IAM?

AWS IAM is a service that manages **authentication and authorization** in AWS. It controls **who can access what** and **what actions are allowed**.

Why did we use IAM?

AWS services do not have permission to access other services by default. IAM is required to: - Allow AWS Glue to read data from S3 - Allow AWS Glue to write processed data back to S3

How IAM was used in our project

- We created an **IAM Role** for AWS Glue
- Attached required policies:
- AmazonS3FullAccess
- AWSGlueServiceRole
- This role was assigned to the Glue Crawler and Glue ETL Job

---

Key Concept

- Least privilege access (only required permissions are given)

---

3. AWS Glue

AWS Glue is a **serverless ETL (Extract, Transform, Load) service** that is commonly used in big data architectures.

In this project, AWS Glue was used in two different ways:

3.1 AWS Glue Crawler

What is a Glue Crawler?

A Glue Crawler is an automated tool that scans data stored in S3 and **detects its structure (schema)**.

Why did we use a Glue Crawler?

- Our data was in raw JSON format
- We did not want to manually define schema
- Glue Crawler automatically creates tables

How the Glue Crawler works internally

1. Scans files in S3
2. Identifies file format (JSON)
3. Infers column names and data types
4. Creates a table in Glue Data Catalog

How it was used in our project

- Ran once on the raw data bucket
- Created a table inside Glue Data Catalog
- That table was later used by the ETL job

3.2 AWS Glue ETL Job

What is a Glue ETL Job?

A Glue ETL job is a **Spark-based data processing job** that runs in a fully managed environment.

Why did we use Glue ETL?

- To process large volumes of data
- To convert inefficient JSON format into optimized Parquet format
- To simulate real-world big data transformation

How Glue ETL works internally

1. Reads data from Glue Data Catalog
2. Loads data into Spark DynamicFrames

3. Applies transformations
4. Writes output to S3

How Glue ETL was used in our project

- Read raw JSON data
- No complex transformations (simple conversion)
- Wrote data in Parquet format to processed S3 bucket

---

4. AWS Glue Data Catalog

What is Glue Data Catalog?

Glue Data Catalog is a **central metadata repository** that stores information about datasets such as: - Table name - Columns - Data types - Data location in S3

Why is it important?

- Acts like a database schema
- Used by both Glue and Athena
- Avoids schema duplication

How it was used in our project

- Glue Crawler created tables automatically
- Glue ETL job read data using catalog tables
- Athena queried data using catalog metadata

---

5. Amazon Athena

What is Amazon Athena?

Amazon Athena is a **serverless interactive query service** that allows users to run SQL queries directly on data stored in S3.

Why did we use Athena?

- No server setup required
- Supports SQL queries
- Integrates with Glue Data Catalog

- Free Tier friendly for limited usage

How Athena works internally

1. Reads metadata from Glue Data Catalog
2. Scans data stored in S3
3. Executes SQL queries
4. Stores query results in S3

How Athena was used in our project

- Queried Parquet data
- Performed aggregation queries
- Generated analytical insights

---

6. Python (Local Environment)

Why was Python used?

Python was used to: - Generate sample data - Simulate real-world sales transactions - Avoid paid AWS streaming services

---

Role of Python in the architecture

- Acts as data producer
- Creates input dataset for the pipeline
- Keeps project free-tier compliant

---

7. Why Paid Services Were Not Used

Services like Kinesis, EMR, and Redshift were intentionally avoided because: - They are not fully covered under Free Tier - They may cause unexpected billing - Batch processing was sufficient for academic objectives

---

8. Summary of Service Roles

Service	Role in Project
Amazon S3	Data storage (raw + processed)

Service	Role in Project
IAM	Security & permissions
Glue Crawler	Schema detection
Glue ETL	Data transformation
Glue Data Catalog	Metadata storage
Athena	SQL analytics
Python	Data generation

---

9. Project Flowchart and Ordered Processing

This section explains **how data flows through the project step by step**, in the exact order of execution. This is extremely important for **viva, theory exams, and interviews**.

9.1 Logical Flow of the Project

Below is the **flowchart-style representation** of the project:

```
START
  |
  v
[Local Python Script]
  |
  |  (Generate sales data in JSON format)
  v
[Amazon S3 - Raw Data Bucket]
  |
  |  (Stores raw JSON data)
  v
[AWS Glue Crawler]
  |
  |  (Scans S3 and detects schema)
  v
[AWS Glue Data Catalog]
  |
  |  (Stores metadata: tables, columns, data types)
  v
[AWS Glue ETL Job]
  |
  |  (Reads JSON data, converts to Parquet)
```

```
  v
[Amazon S3 - Processed Data Bucket]
  |
  |  (Stores optimized Parquet files)
  v
[Amazon Athena]
  |
  |  (Runs SQL queries on processed data)
  v
END (Analytical Output)
```

9.2 Ordered Step-by-Step Processing Explanation

Step 1: Data Generation

- A local Python script generates sales transaction data
- Data is created in JSON format
- This simulates real-world business data

Step 2: Raw Data Storage (Amazon S3)

- Generated JSON file is uploaded to the S3 raw data bucket
- S3 acts as a scalable data lake

Step 3: Schema Detection (Glue Crawler)

- Glue Crawler scans the raw S3 data
- Automatically identifies columns and data types
- Creates a table in Glue Data Catalog

Step 4: Metadata Management (Glue Data Catalog)

- Stores schema information
- Acts as a bridge between S3, Glue ETL, and Athena

Step 5: Data Transformation (Glue ETL Job)

- Reads raw JSON data using catalog metadata
- Converts JSON into Parquet format
- Writes processed data to another S3 bucket

Step 6: Processed Data Storage (Amazon S3)

- Stores columnar Parquet files
- Optimized for analytical queries

Step 7: Data Analysis (Amazon Athena)

- Athena reads metadata from Glue Data Catalog
- Executes SQL queries on Parquet data in S3
- Produces analytical results

9.3 Why This Ordered Flow Is Important

- Ensures separation of raw and processed data
- Improves query performance
- Reduces storage and query cost
- Follows industry-standard data lake architecture

---

10. One-Line Explanation (Viva Ready)

> "The project follows an ordered data pipeline where data is generated, stored in S3, cataloged using Glue, transformed into Parquet format, and finally analyzed using Amazon Athena, all within AWS Free Tier limits."