Data & Analytics

Amazon Athena

- Serverless query service to analyze data stored in Amazon S3
- Uses standard SQL language to query the files (built on Presto)
- Supports multiple formats: CSV, JSON, ORC, Avro, and Parquet
- Pricing: $5.00 per TB of data scanned
- Commonly used with Amazon QuickSight for reporting and dashboards

Use Cases

- Business intelligence, analytics, and reporting
- Analyzing VPC Flow Logs, ELB Logs, CloudTrail trails, and more

Exam Tip

- Use Athena to analyze data in S3 using serverless SQL

Diagram Explanation

- Data is stored in an S3 bucket
- Amazon Athena loads and queries the data using SQL
- Results can be visualized in Amazon QuickSight as dashboards or reports

Amazon Athena – Performance Improvement

- Use columnar data formats to reduce scan size and cost:
  - Recommended formats: Apache Parquet or ORC
  - Significant performance improvements
  - AWS Glue can be used to convert data into Parquet or ORC
- Compress data to reduce retrieval size:
  - Supported compression formats: bzip2, gzip, lz4, snappy, zlib, zstd, etc.
- Partition datasets in S3 to enable efficient querying on virtual columns:
  - Partitioning structure:
    - `s3://yourBucket/pathToTable/<PARTITION_COLUMN_NAME>=<VALUE>/...`
  - Example:
    - `s3://athena-examples/flight/parquet/year=1991/month=1/day=1/`
- Use larger files (greater than 128 MB) to reduce query overhead

Amazon Athena – Federated Query

- Enables running SQL queries across various data sources:
  - Relational
  - Non-relational
  - Object storage
  - Custom data sources
  - Sources can be on AWS or on-premises

- Utilizes **Data Source Connectors**:
  - These run as AWS Lambda functions
  - Allow querying services like CloudWatch Logs, DynamoDB, RDS, etc.
- Query results can be stored back in Amazon S3

Example Data Sources

- S3
- DynamoDB
- Redshift
- DocumentDB
- ElastiCache
- Aurora, MySQL, SQL Server
- HBase on EMR
- On-premises databases

Redshift Overview

- Redshift is based on PostgreSQL, but it is not intended for OLTP workloads
- It is designed for OLAP (Online Analytical Processing), suitable for analytics and data warehousing
- Offers up to 10x better performance compared to traditional data warehouses
- Uses **columnar storage** instead of row-based storage
- Employs a **parallel query engine** for high performance
- Supports two deployment modes:
  - Provisioned cluster
  - Serverless cluster
- SQL interface available for querying data
- Integrates with BI tools such as Amazon QuickSight and Tableau

Comparison with Athena

- Redshift offers faster performance for:
  - Complex queries
  - Joins
  - Aggregations
  - Thanks to indexes and optimized execution engine

Redshift Cluster

- **Leader Node**:
  - Handles query planning and aggregates the results
- **Compute Nodes**:
  - Execute the actual queries
  - Return results to the leader node

Provisioned Mode

- Instance types must be selected in advance
- Option to reserve instances for cost savings

Query Interface

- Access via JDBC/ODBC
- SQL queries are sent to the cluster
    - Example: `SELECT COUNT(*), ... FROM MY_TABLE GROUP BY ...`

Redshift – Snapshots & Disaster Recovery

- Some Redshift clusters support **Multi-AZ** mode

Snapshots

- Snapshots are point-in-time backups stored internally in S3
- They are **incremental**, meaning only changes are saved
- You can restore a snapshot to create a new cluster

Snapshot Types

- **Automated Snapshots**:

    - Created every 8 hours, every 5 GB of data changes, or based on a schedule
    - Retention period configurable: from 1 to 35 days

- **Manual Snapshots**:

    - Persist until explicitly deleted

Cross-Region Copy

- Redshift can automatically copy snapshots (automated or manual) to another AWS Region
- Enables disaster recovery and regional backup strategies

Loading Data into Redshift

- **Large inserts** are significantly more efficient than small inserts
- Best practice: write data in **batches**

Recommended Loading Methods

- **Amazon Kinesis Data Firehose**:

    - Can load streaming data directly into Redshift (via S3 using COPY command)

- **Amazon S3 + COPY Command**:

    - Preferred method for bulk data loads
    - Example:

    ```
    copy customer
    from 's3://mybucket/mydata'
    iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
    ```

Networking Options

- **Without Enhanced VPC Routing**:
  - Data transfer goes through the internet
- **With Enhanced VPC Routing**:
  - Data is routed through the VPC for better control and security

Additional Integration

- **EC2 instances** using JDBC drivers can also load data into Redshift

Redshift Spectrum

- Allows querying data directly from **Amazon S3** without the need to load it into Redshift
- Requires a **Redshift cluster** to be available to initiate the query

How It Works

- SQL query is submitted from the Redshift cluster
- The query is delegated to **thousands of Redshift Spectrum nodes** for execution
- Useful for querying external tables stored in S3

Example Query

```
SELECT COUNT(*), …
FROM S3.EXT_TABLE
GROUP BY …
```

Architecture Components

- Redshift cluster (Leader and Compute Nodes)
- Redshift Spectrum nodes (scale out query execution)
- Amazon S3 as data source
- JDBC/ODBC access from clients

Amazon OpenSearch Service

- Amazon OpenSearch is the **successor to Amazon Elasticsearch**
- Unlike DynamoDB (which supports only primary key/index queries), OpenSearch allows:
  - Full-text search
  - Partial match queries
  - Search on any field
- Often used **in combination** with other databases for enhanced search capabilities

Deployment Modes

- Managed cluster
- Serverless cluster

Query Support

- SQL is **not supported natively**, but can be enabled via a plugin

Data Ingestion

- Integrates with:
  - Kinesis Data Firehose
  - AWS IoT
  - CloudWatch Logs

Security Features

- Integration with:
  - AWS Cognito
  - IAM for access control
  - KMS for encryption
  - TLS for data in transit
- Includes **OpenSearch Dashboards** for data visualization

OpenSearch Patterns

Common Integration with DynamoDB

- **DynamoDB Table** stores operational data
- **DynamoDB Stream** captures item-level changes
- **AWS Lambda Function** processes the stream and sends the data to **Amazon OpenSearch**

Access Patterns

- Use DynamoDB API for:
  - **CRUD operations** (Create, Read, Update, Delete)
  - Retrieving items by key or index
- Use OpenSearch API for:
  - **Search operations**
  - Full-text search
  - Filtering and querying non-key fields

Pattern Summary

- This integration provides the best of both:
  - High-performance key-based access with DynamoDB
  - Rich search capabilities with OpenSearch

OpenSearch Patterns – CloudWatch Logs

Integration Options

1. **Real-Time Ingestion**

- Use a **CloudWatch Logs Subscription Filter**
- Connects directly to a **Lambda Function**
- Lambda sends log data to **Amazon OpenSearch**
- Managed by AWS for simplified setup

2. **Near Real-Time Ingestion**

- Use a **CloudWatch Logs Subscription Filter**
- Sends data to **Kinesis Data Firehose**
- Firehose delivers data to **Amazon OpenSearch**

Summary

- Choose **Lambda** for low-latency, real-time streaming
- Use **Kinesis Data Firehose** for a more scalable, near real-time pipeline

OpenSearch Patterns – Kinesis Integration

Near Real-Time Ingestion

- **Kinesis Data Streams → Kinesis Data Firehose → Amazon OpenSearch**
- Kinesis Data Firehose handles:
  - Buffering
  - Optional **data transformation**
  - Automatic delivery to OpenSearch

Real-Time Ingestion

- **Kinesis Data Streams → Lambda Function → Amazon OpenSearch**
- Lambda processes records in real time and pushes them to OpenSearch

Summary

- Use **Firehose** for managed, near real-time delivery with optional transformation
- Use **Lambda** for full control and low-latency, real-time delivery

Amazon EMR

- **EMR** stands for **Elastic MapReduce**
- Used to create and manage **Hadoop clusters** for processing and analyzing large-scale data

Features

- Can scale to **hundreds of EC2 instances**
- Comes pre-installed with big data tools such as:
  - Apache Spark
  - Apache HBase
  - Presto
  - Apache Flink
- Handles cluster provisioning and configuration automatically
- Supports **auto-scaling**
- Integrated with **Spot Instances** to reduce cost

Use Cases

- Large-scale **data processing**
- **Machine learning** workloads
- **Web indexing**
- General **big data** analytics

Amazon EMR – Node Types & Purchasing

Node Types

- **Master Node**:
  - Manages the cluster
  - Coordinates jobs and monitors health
  - Long-running
- **Core Node**:
  - Executes tasks
  - Stores data
  - Long-running
- **Task Node** (optional):
  - Executes tasks only
  - Typically configured as **Spot Instances**

Purchasing Options

- **On-Demand**:
  - Reliable and predictable
  - Won't be interrupted or terminated
- **Reserved Instances**:
  - Minimum 1-year commitment
  - Cost savings
  - EMR automatically uses reserved instances if available
- **Spot Instances**:
  - Low cost
  - Can be terminated
  - Less reliable

Cluster Duration

- Can be either:
  - **Long-running** cluster
  - **Transient** (temporary) cluster for short-term workloads

Amazon QuickSight

- **Serverless**, machine learning-powered **business intelligence (BI)** service
- Used to create **interactive dashboards** and visualizations
- Automatically scalable and embeddable
- Supports **per-session pricing**

Use Cases

- Business analytics
- Creating visualizations
- Ad-hoc data analysis
- Gaining insights from business data

Data Integration

- Integrated with multiple AWS data sources:
  - Amazon RDS
  - Aurora
  - Athena
  - Redshift
  - Amazon S3

Performance Features

- Uses **SPICE** (Super-fast, Parallel, In-memory Calculation Engine) for in-memory computation when data is imported

Security Features

- **Enterprise Edition** supports **Column-Level Security (CLS)**

QuickSight Integrations

Amazon QuickSight supports a wide range of data sources across AWS, SaaS, imports, and on-premises systems.

Data Sources – AWS Services

- Amazon RDS
- Amazon Redshift
- Amazon Athena
- Amazon S3
- Amazon Aurora
- Amazon Timestream
- ELF & CLF (Extended and Common Log Format)
- Amazon OpenSearch

Data Sources – SaaS

- Direct integrations with various Software-as-a-Service platforms (not explicitly listed in the slide)

Data Sources – Imports

- Static or scheduled data imports from external systems

Data Sources – On-Premises

- JDBC-compatible on-premises databases

These integrations allow QuickSight to provide flexible, real-time, or near real-time business intelligence from a broad array of sources.

QuickSight – Dashboard & Analysis

Users and Groups

- QuickSight supports:
  - **Users** (available in standard version)
  - **Groups** (available in enterprise version)
- Note: These users and groups exist **only within QuickSight**, not in IAM

Dashboard

- A **dashboard** is a **read-only** snapshot of an **analysis**
- It retains the configuration of the analysis:
  - Filters
  - Parameters
  - Controls
  - Sorting

Sharing

- You can share both **analyses** and **dashboards** with QuickSight Users or Groups
- To share a dashboard:
  - It must first be **published**
- Shared users can also access the **underlying data**

AWS Glue

- Fully managed **extract, transform, and load (ETL)** service
- Used to **prepare and transform data** for analytics
- Fully **serverless**, no infrastructure to manage

Typical ETL Flow

1. **Extract**:

   - Data is read from sources such as:
     - Amazon S3
     - Amazon RDS

2. **Transform**:

   - Data is cleaned, enriched, or reshaped using **Glue ETL jobs**

3. **Load**:

   - Transformed data is written to destinations such as:
     - Amazon Redshift (Data Warehouse)

AWS Glue is ideal for building scalable and automated data pipelines.

AWS Glue – Convert Data into Parquet Format

Workflow

1. **Input**:

   - CSV files are uploaded into an **S3 Bucket**

2. **Trigger**:

   - **S3 PUT event** triggers an **AWS Lambda function**
   - Alternatively, **EventBridge** can be used to trigger the ETL job

3. **ETL Process**:

   - The Lambda or EventBridge trigger starts a **Glue ETL Job**

    - Glue reads the CSV files and converts them into **Parquet** format

4. **Output**:
    - The converted Parquet files are written back to **another S3 Bucket**

5. **Analysis**:
    - The Parquet data can then be analyzed efficiently using **Amazon Athena**

Summary

- This setup automates the transformation of incoming CSV data into an optimized, columnar format (Parquet), enabling cost-effective querying in Athena.

Glue Data Catalog: Catalog of Datasets

Overview

- The **AWS Glue Data Catalog** is a **central metadata repository** used to store metadata about datasets
- It enables **data discovery**, **querying**, and **management** of datasets across various services

Supported Sources

- Amazon S3
- Amazon RDS
- Amazon DynamoDB
- JDBC data sources

Components

- **Databases**: Logical grouping of tables
- **Tables**: Contain metadata for the datasets

Metadata Generation

- AWS Glue **Crawlers** scan data sources and **automatically populate the catalog** with metadata

Integration with Other Services

- **Amazon Athena**: Uses the Glue Data Catalog to run queries on S3 data
- **Amazon Redshift Spectrum**: Queries S3 data using cataloged metadata
- **Amazon EMR**: Can use the Data Catalog as a Hive-compatible metastore
- **Glue Jobs (ETL)**: Use the catalog for input/output dataset definitions

Summary

- Centralized metadata layer that powers analytics across AWS services
- Supports schema discovery and reusability for ETL and querying

Glue – Things to Know at a High-Level

Glue Job Bookmarks

- Prevent reprocessing of already-processed data
- Useful for incremental ETL workflows

Glue DataBrew

- GUI-based tool for **cleaning and normalizing data**

- Uses **pre-built transformations**
- No-code interface for data analysts and engineers

Glue Studio

- Visual interface to:
  - Create ETL workflows
  - Run jobs
  - Monitor execution

Glue Streaming ETL

- Built on **Apache Spark Structured Streaming**
- Enables real-time data processing
- Compatible with:
  - **Kinesis Data Streams**
  - **Apache Kafka**
  - **Amazon MSK** (Managed Streaming for Kafka)

AWS Lake Formation

- A **data lake** is a centralized repository to store all your data for analytics
- **AWS Lake Formation** is a fully managed service that simplifies the setup and management of data lakes

Key Features

- Helps you **discover, cleanse, transform, and ingest** data into your lake
- Automates complex manual steps:
  - Data collection
  - Cleansing
  - Moving
  - Cataloging
- Can perform **data deduplication** using **ML transforms**
- Supports both **structured** and **unstructured** data

Integrations & Sources

- Provides built-in blueprints for:
  - Amazon S3
  - Amazon RDS
  - Relational and NoSQL databases

Security

- Offers **fine-grained access control**:
  - Supports **row-level** and **column-level** permissions for applications

Architecture

- Built on top of **AWS Glue**

AWS Lake Formation – Data Sources

Ingestion Sources

- **Amazon S3**

- **Amazon RDS**
- **Aurora**
- **On-Premises Databases** (both SQL and NoSQL)

Lake Formation Workflow

1. **Source Crawlers**:
    - Discover schema and populate the **Data Catalog**
2. **ETL and Data Preparation**:
    - Data is cleaned and transformed as needed
3. **Data Catalog**:
    - Central metadata repository for all ingested data
4. **Security Settings**:
    - Define **access control policies** (row/column-level)
5. **Data Lake**:
    - Final processed data is stored in **Amazon S3**

Consumer Services

- Data in the lake can be accessed by:
    - **Amazon Athena**
    - **Amazon Redshift**
    - **Amazon EMR**
    - Other analytics tools
    - Authorized **users and applications**

AWS Lake Formation provides an integrated platform to build and secure a modern data lake.

AWS Lake Formation – Centralized Permissions Example

Data Sources

- **Amazon S3**
- **Amazon RDS**
- **Amazon Aurora**

Ingestion

- Data from various sources is ingested into the **Data Lake**, stored in **Amazon S3**

Centralized Access Control

- AWS Lake Formation applies centralized **access control policies**
- Supports **column-level security**
- Ensures that users and services only access authorized portions of data

Data Consumers

- **Amazon Athena** users can query the data lake with permissions enforced
- **Amazon QuickSight** can visualize data with access control applied

Summary

Lake Formation provides a secure, centralized way to govern access across your analytics ecosystem while supporting fine-grained permissions.

Amazon Managed Service for Apache Flink

Overview

- Formerly known as **Kinesis Data Analytics for Apache Flink**
- Apache Flink is a **stream processing framework** (Java, Scala, SQL)

Key Features

- Fully **managed service** to run Apache Flink applications on AWS
- Supports:
  - **Provisioned compute resources**
  - **Parallel computation**
  - **Automatic scaling**
  - **Application backups** via **checkpoints** and **snapshots**
- Full compatibility with the **Apache Flink programming model**
  - Enables complex **real-time data transformations**

Data Sources

- Supports streaming input from:
  - **Amazon Kinesis Data Streams**
  - **Amazon MSK (Managed Streaming for Apache Kafka)**
- **Does NOT support Amazon Kinesis Data Firehose** as a data source

This service simplifies building and running stateful stream processing applications on AWS infrastructure.

Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Overview

- Fully managed **Apache Kafka** service on AWS
- Alternative to **Amazon Kinesis** for streaming data

Key Features

- Create, update, and delete **Kafka clusters**
- AWS manages:
  - **Kafka broker nodes**
  - **Zookeeper nodes**
- **Deployed within your VPC**
  - Supports **Multi-AZ** (up to 3 Availability Zones) for high availability
- Automatic recovery from common Kafka failures
- **Data persistence** on **EBS volumes**, retained as long as needed

Amazon MSK Serverless

- Run Kafka **without managing capacity**
- Automatically provisions resources
- Scales **compute and storage** based on workload

MSK is suitable for real-time data pipelines, event-driven applications, and stream processing with full compatibility with the Kafka ecosystem.

Apache Kafka at a High Level

Core Components

- **MSK Cluster**:
  - Consists of multiple **Kafka brokers** (e.g., Broker 1, Broker 2, Broker 3)
  - Brokers handle storage, message distribution, and client connections
  - Brokers replicate data between each other for fault tolerance
- **Producers**:
  - Custom applications that **write data to Kafka topics**
- **Consumers**:
  - Custom applications that **read data from Kafka topics**

Data Flow

- Data is **written to a topic** by producers
- Data is **polled from a topic** by consumers
- Data replication ensures durability across brokers

Integration Targets

- Kafka topics can be consumed and processed by:
  - **Amazon EMR**
  - **Amazon S3**
  - **Amazon SageMaker**
  - **Amazon Kinesis**
  - **Amazon RDS**
  - **IoT devices** and many other sources/sinks

Kafka is ideal for building decoupled, real-time data pipelines and streaming analytics platforms.

Kinesis Data Streams vs. Amazon MSK

Kinesis Data Streams

- **Message size limit**: 1 MB
- Uses **Shards** to manage data throughput
  - Supports **Shard Splitting & Merging** to scale
- **Encryption**:
  - **TLS** for in-flight encryption
  - **KMS** for at-rest encryption

Amazon MSK (Managed Streaming for Apache Kafka)

- **Default message size**: 1 MB, configurable (e.g., up to 10 MB)
- Uses **Kafka Topics with Partitions**
    - Only **adds partitions**, cannot remove or merge
- **Encryption**:
    - Supports **PLAINTEXT** or **TLS** for in-flight encryption
    - **KMS** for at-rest encryption

Summary

Feature	Kinesis Data Streams	Amazon MSK
Message Size Limit	1 MB	1 MB (default), configurable
Scaling Mechanism	Shard splitting/merging	Add partitions only
In-Flight Encryption	TLS	PLAINTEXT or TLS
At-Rest Encryption	KMS	KMS

Amazon MSK Consumers

Amazon MSK (Managed Streaming for Apache Kafka) supports a wide variety of consumer services and applications.

MSK Consumers Include:

- **Kinesis Data Analytics for Apache Flink**
    - Real-time analytics using Flink applications
- **AWS Glue Streaming ETL Jobs**
    - Powered by **Apache Spark Streaming**
- **AWS Lambda**
    - Event-driven processing of Kafka streams
- **Amazon EC2**
    - Custom applications running Kafka clients
- **Amazon ECS / Amazon EKS**
    - Containerized applications that consume Kafka topics

These consumers allow MSK to integrate seamlessly into real-time processing pipelines, batch processing workflows, and container-based architectures.

Big Data Ingestion Pipeline

Goals of the Architecture

- The ingestion pipeline should be **fully serverless**
- Ability to **collect data in real time**
- Perform **data transformation** during ingestion
- Allow querying of the transformed data using **SQL**
- Store the resulting **reports** in **Amazon S3**

- Load the transformed and queried data into a **data warehouse**
- Build **dashboards** on top of the warehouse data

This pipeline supports real-time analytics, cost-effective storage, and scalable visualization, all within a serverless framework.

Big Data Ingestion Pipeline – Architecture

Data Sources

- **IoT Devices** send real-time data into the pipeline

Ingestion Layer

- **Amazon Kinesis Data Streams**: receives and buffers real-time data
- **Amazon Kinesis Data Firehose**: delivers streaming data to downstream services

Processing Layer

- **AWS Lambda**:
  - Triggers on data arrival (e.g., every 1 minute)
  - Transforms data before delivery or storage
- **Amazon SQS** (optional):
  - Used for decoupling and buffering events between services

Storage Layer

- **Amazon S3**:
  - **Ingestion Bucket**: stores raw or pre-processed data
  - **Reporting Bucket**: stores transformed, query-ready data

Query and Analytics

- **Amazon Athena**:
  - SQL-based querying over S3 data
- **Amazon Redshift Serverless**:
  - Loads processed data for further analysis and BI
- **Amazon QuickSight**:
  - Visualizes the data from Redshift and S3 for dashboards

Summary

This pipeline supports end-to-end real-time ingestion, transformation, storage, querying, and dashboarding using a fully **serverless AWS architecture**.

Big Data Ingestion Pipeline – Discussion

Component Roles

- **IoT Core**:

    - Collects and ingests data from **IoT devices**
- **Amazon Kinesis**:
    - Ideal for **real-time data collection**
- **Amazon Kinesis Data Firehose**:
    - Delivers data to **Amazon S3** in near real-time (every ~1 minute)
    - Can be combined with **AWS Lambda** for inline **data transformation**
- **Amazon S3**:
    - Stores both raw and transformed data
    - Can emit **event notifications** to **Amazon SQS**
- **Amazon SQS**:
    - Buffers notifications and decouples downstream consumers
- **AWS Lambda**:
    - Can subscribe to **SQS**
    - Alternatively, S3 events could directly trigger Lambda
- **Amazon Athena**:
    - Serverless SQL engine that queries data in S3
    - Stores results back into S3
- **Reporting Bucket**:
    - Contains analyzed output
    - Used by **AWS QuickSight**, **Amazon Redshift**, or other reporting tools

This architecture enables scalable, modular, and fully serverless big data processing and analytics.