Grow Data Skills

AWS DynamoDB

Amazon DynamoDB is a managed NoSQL database service provided by AWS. It offers fast and predictable performance with seamless scalability. DynamoDB allows users to create database tables that can store and retrieve any amount of data and serve any level of request traffic.

- **Table**: It's the primary data storage container in DynamoDB. Like other databases, a table in DynamoDB holds data in rows and columns. However, because it's a NoSQL database, there's no requirement for each row to have the same columns.

- **Items**: These are the individual rows or records in a DynamoDB table. An item consists of one or more attributes.

- **Attributes**: These are the data elements or columns in an item. An attribute has a name and a value. The value can be of different types such as String, Number, Binary, Boolean, List, Map, etc.

- **Primary Key**: It uniquely identifies each item in a table. The primary key consists of one attribute (Partition Key) or two attributes (Partition Key and Sort Key).

    - **Partition Key (PK):** It determines the partition in which the item will be stored. DynamoDB uses the partition key's value to distribute data across partitions for scalability and performance.

    - **Sort Key (SK)**: It's the second part of the primary key and used to sort items in a partition.

AWS DynamoDB

Grow Data Skills

- **Secondary Indexes:** These are optional and allow you to search for items based on attributes other than the primary key. There are two types of secondary indexes:
    - **Global Secondary Index (GSI)**: Allows querying data on any attribute (or combination of attributes), even if it's not the primary key. It can span multiple partitions and is scoped to the entire table.
    - **Local Secondary Index (LSI):** Allows querying data based on the primary partition key and an alternate sort key, but it's scoped to a single partition.
- **DynamoDB Accelerator (DAX):** It's a fully managed caching service that sits in front of your DynamoDB table and provides up to a 10x performance improvement on read-intensive workloads.
- **Consistency:** DynamoDB offers two types of read consistency:
    - **Eventually Consistent Reads:** Data might not reflect the results of a recently completed write operation, but it offers better read performance.
    - **Strongly Consistent Reads:** It returns a result that reflects all writes that received a successful response before the read.

AWS DynamoDB - Use Cases

Amazon DynamoDB's flexible schema, automatic scaling capabilities, and high availability make it suitable for a variety of use cases. Here are some of the best use cases for DynamoDB:

- **Web Applications**: DynamoDB is commonly used as a backend datastore for web applications that require consistent, single-digit millisecond latency. It can handle both small and massive amounts of user data.

- **Mobile Apps**: For mobile applications that need to store user profiles, preferences, or state, DynamoDB provides a low-latency data access layer.

- **Gaming**: DynamoDB can handle the high-rate leaderboards, game states, and player profiles often needed in the gaming industry, especially for popular games with millions of players.

- **Ad Tech**: Due to its high throughput and low latency, DynamoDB is a great fit for advertising technology platforms, especially for real-time bidding environments.

- **IoT Applications**: DynamoDB can store time-series data from IoT devices. For instance, sensor data can be recorded with timestamps and queried efficiently using a composite primary key.

- **Session Storage**: For websites and online platforms with millions of active users, DynamoDB can be used to manage user session data with fast access times.

- **E-Commerce**: DynamoDB can store product catalogs, manage shopping carts, and handle order histories in e-commerce platforms.

- **Content Management Systems (CMS):** For managing metadata about content, users, permissions, and other entities.

AWS EMR

AWS EMR (Amazon Elastic MapReduce) is a cloud-native big data platform that allows processing of vast amounts of data quickly and cost-effectively across resizable clusters of Amazon EC2 instances. AWS EMR provides a managed environment that simplifies the setup, configuration, and management of big data processing frameworks such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, and others.

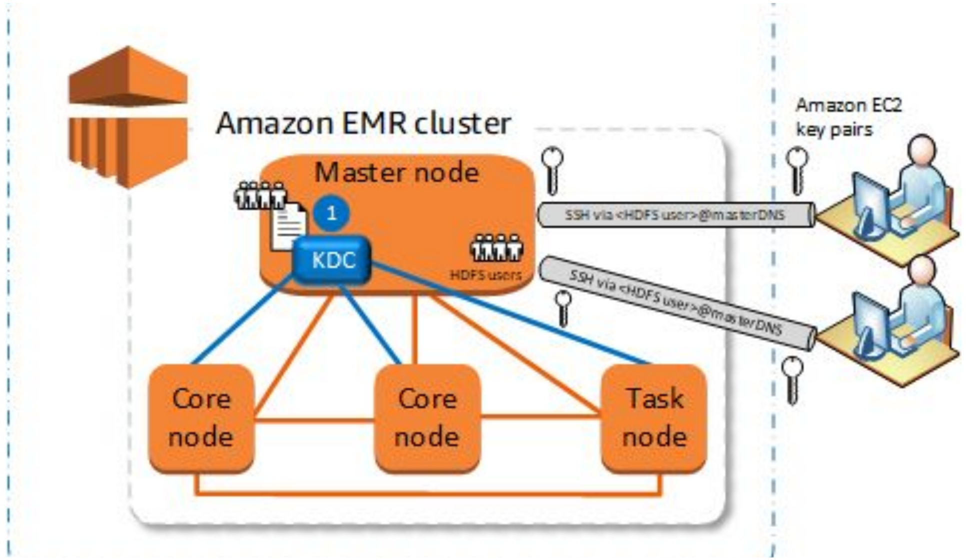
Main architectural components of AWS EMR:

Cluster: This is the collection of EC2 instances. They can be easily resized and consist of one master node, core nodes, and optional task nodes. Each of these has specific roles in the processing of data.

- **Master Node:** Manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing.
- **Core Nodes:** Run software components to process data and store data on the Hadoop Distributed File System (HDFS) on your cluster. Every cluster should have at least one core node.
- **Task Nodes:** Task nodes are optional. They run software components to process data but do not store data in HDFS. You can use them to add processing power to your cluster.

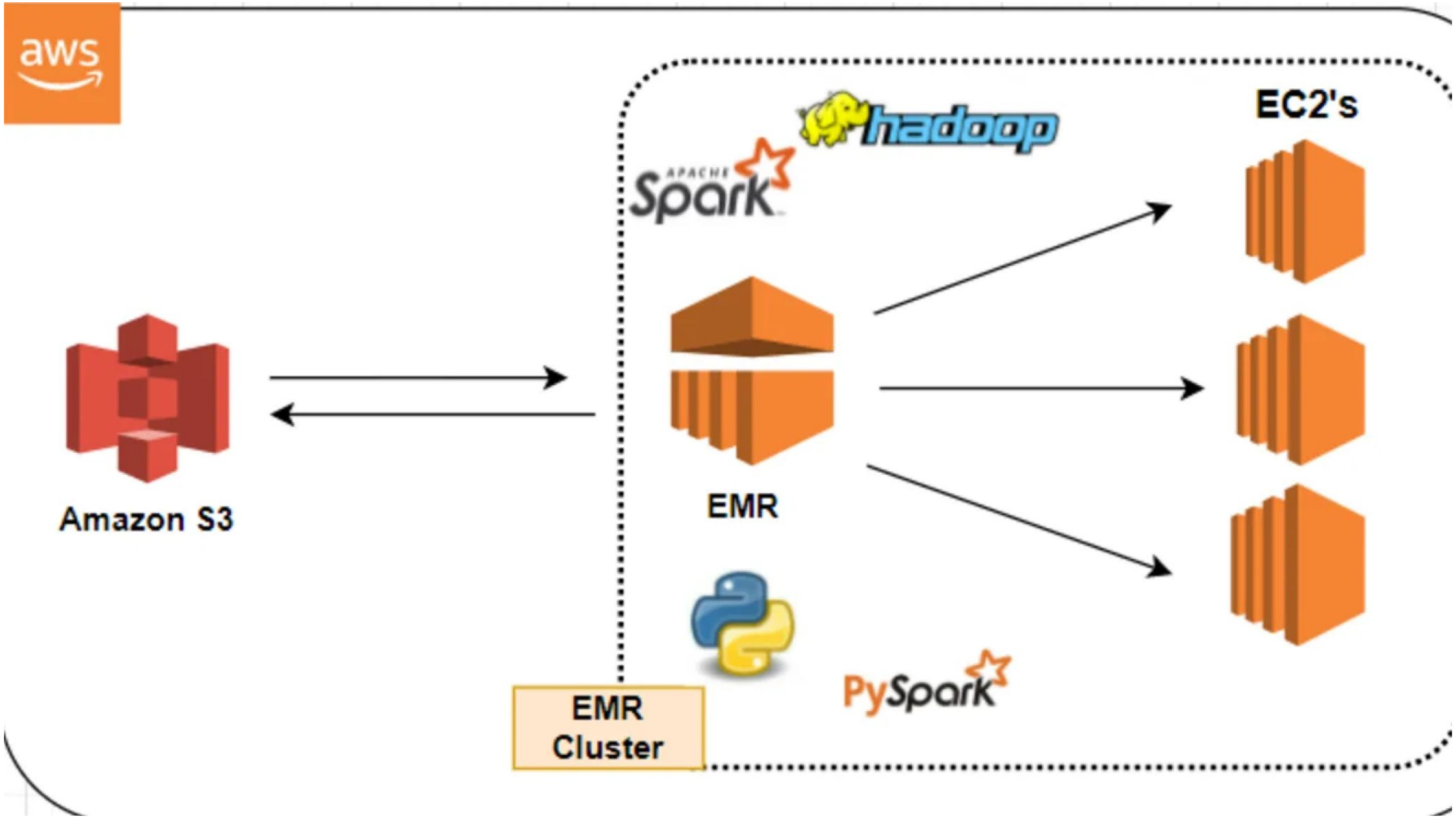
AWS EMR - Architecture

Grow Data Skills



AWS EMR - Data Pipeline

Grow Data Skills


aws
hadoop
APACHE Spark
EC2's
Amazon S3
EMR
PySpark
EMR Cluster