Grow Data Skills

AWS Kinesis

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data. It allows you to build applications that can continuously ingest and process large streams of data records in real-time.

Key Features:

- **Real-Time Processing:** Kinesis provides the capability to process and analyze data as soon as it arrives, which is critical for time-sensitive applications that need to make quick decisions based on the latest data.

- **Scalability**: Automatically scales to handle massive data throughput, without the need to manage infrastructure.

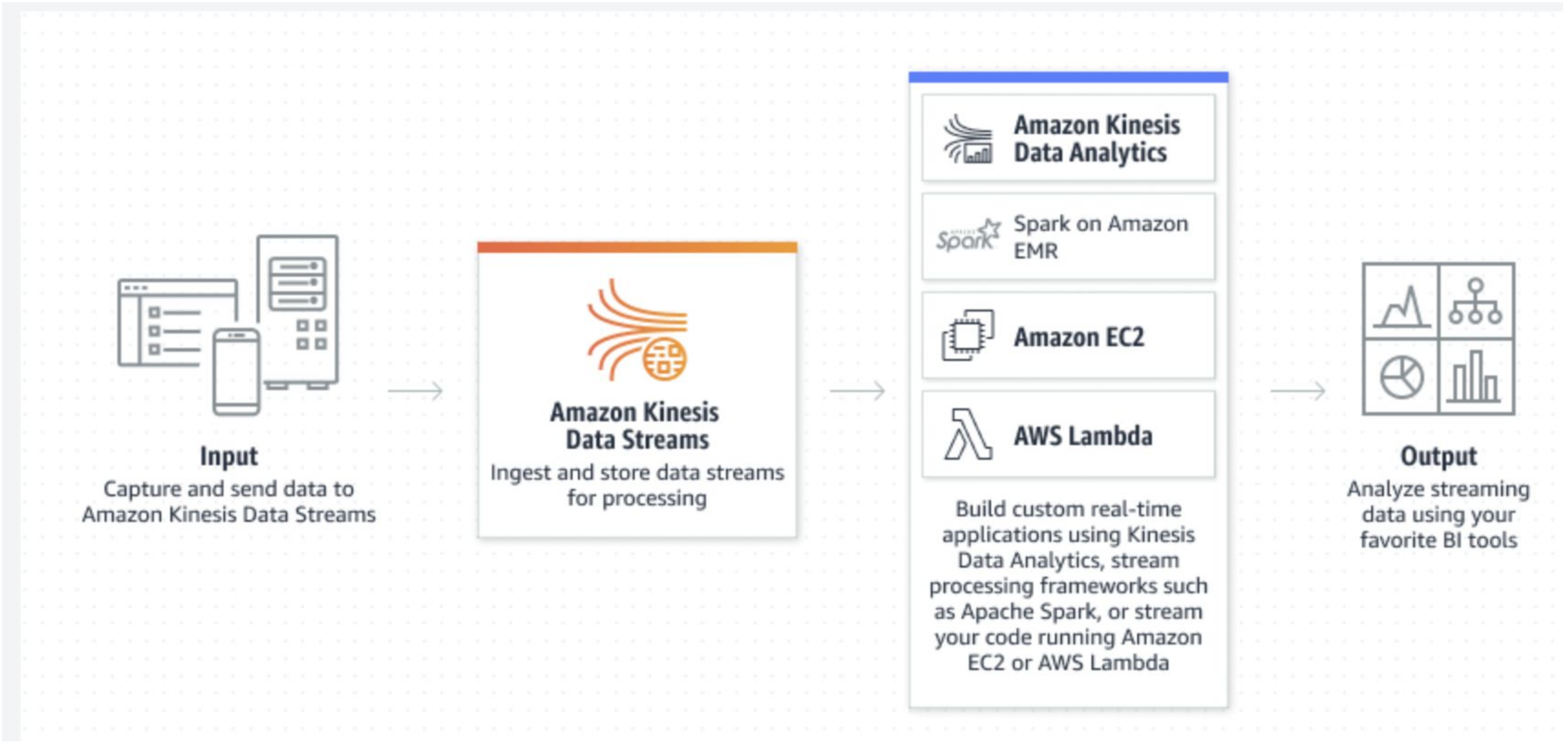
- **Data Collection from Various Sources:** Can collect data from countless sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events.

Components of Amazon Kinesis:

- **Kinesis Data Streams:** For building custom, real-time applications that process or analyze streaming data.
- **Kinesis Data Firehose:** For effortlessly loading streaming data into AWS data stores.
- **Kinesis Data Analytics:** For processing and analyzing streaming data with standard SQL.

AWS Kinesis

Grow Data Skills


Input
Capture and send data to Amazon Kinesis Data Streams
Amazon Kinesis Data Streams
Ingest and store data streams for processing
Amazon Kinesis Data Analytics
Spark on Amazon EMR
Amazon EC2
AWS Lambda
Build custom real-time applications using Kinesis Data Analytics, stream processing frameworks such as Apache Spark, or stream your code running Amazon EC2 or AWS Lambda
Output
Analyze streaming data using your favorite BI tools

AWS Kinesis vs Kafka

- **Origin and Hosting**
  - **Kinesis**: A managed service provided by AWS, offering seamless integration with other AWS services and a serverless option for handling data streams without managing infrastructure.
  - **Kafka**: An open-source platform initially developed by LinkedIn and now a part of the Apache Software Foundation. Kafka can be self-hosted or managed by various cloud providers through services like Confluent.

- **Deployment and Management**
  - **Kinesis**: Being a managed service, it significantly reduces the operational load of managing and scaling infrastructure. It's ideal for teams preferring cloud-native tools with minimal setup.
  - **Kafka**: Requires more effort to set up, manage, and scale, especially in a self-hosted environment. However, it offers more control and flexibility over the system.

- **Performance and Scalability**
  - **Kinesis**: Automatically scales to accommodate data throughput but has limits that can be increased upon request to AWS. Scaling involves increasing the number of shards in a stream.
  - **Kafka**: Highly scalable and can handle high throughput with proper hardware and configuration. Scaling typically involves adding more brokers (servers) and properly partitioning topics.
- **Data Retention**
  - **Kinesis**: Default data retention is 24 hours, extendable up to 365 days. Extended retention comes at an additional cost.
  - **Kafka**: Configurable data retention policies based on time or size. It can retain data for as long as the disk space allows, supporting long-term storage if needed.

AWS Kinesis vs Kafka

- <u>**Pricing**</u>
  - **Kinesis**: Charges are based on the amount of data ingested, stored, and processed, along with the number of shards used.
  - **Kafka**: For self-hosted Kafka, the main cost is infrastructure and operational efforts. Managed Kafka services charge based on usage, similar to Kinesis, but specific costs vary by provider.

- <u>**Ecosystem and Integrations**</u>
  - **Kinesis**: Natively integrates with AWS services like AWS Lambda, Amazon S3, Amazon Redshift, and Amazon DynamoDB, offering a streamlined experience for AWS users.
  - **Kafka**: Boasts a broad ecosystem and a large number of connectors for integrating with various databases, data lakes, and applications. This makes it highly versatile across different environments.

- <u>**Use Cases**</u>
  - **Kinesis**: Ideal for AWS-centric organizations looking for a managed service for real-time analytics, log and event data collection, and application monitoring.
  - **Kafka**: Suited for organizations requiring high throughput and scalability, complex real-time analytics, and a high degree of flexibility and control over their streaming data architecture.

AWS DynamoDB

- DynamoDB is a NoSQL database provided by Amazon Web Service (AWS). It can provide extremely high performance, more than 10 trillion requests per day with peaks greater than 20 million requests per second, and can support virtually any size with horizontal scaling. It is not uncommon for DynamoDB to serve over petabytes of data.

- From a 10,000 feet view, DynamoDB is basically a key-value store. It can be thought as a hash-map backed by some persistent storage. Two most important operations supported by DynamoDB are Get and Put.

- Unsurprisingly, the semantics of Get and Put operations are consistent with our understanding of hash-map or key-value store. For instance, we can use Put operation to persists a key-value pair to the DynamoDB and a subsequent Get operation will return the value we stored previously.
    - **Put("key", "value");**
    - **Get("Key") → Returns "value"**

- **<u>DynamoDB in CAP Theorem:</u>** DynamoDB is designed to provide high **availability** and **partition** tolerance. It achieves this by replicating data across multiple AWS regions and Availability Zones, ensuring that the system remains operational even in the event of server failures or network partitions. This design choice means that DynamoDB can continue to process reads and writes, maintaining the system's availability and partition tolerance.

AWS DynamoDB Architecture

Amazon DynamoDB's architecture is designed to provide fast, predictable performance at any scale for a wide variety of use cases, including mobile, web, gaming, ad tech, IoT, and many more. It's a fully managed, multi-region, durable database with built-in security, backup and restore, and in-memory caching. Here are the key components and architectural aspects of DynamoDB:

- **<u>Core Components</u>**
    - **Tables**: The fundamental unit of data storage in DynamoDB. Tables contain items, and items contain attributes. Unlike traditional relational databases, there's no schema imposed on a table, allowing each item to have a different number of attributes.

    - **Items**: Each item is a collection of attributes that is uniquely identifiable among all of the other items. In DynamoDB, there's no limit to the number of items you can store in a table.

    - **Attributes**: Each item is composed of one or more attributes (name-value pairs), which are the fundamental data elements that DynamoDB stores. Attributes in DynamoDB are similar to fields or columns in other database systems.

AWS DynamoDB Architecture

- **Key Architectural Features**
  - **Primary Key**: The primary key uniquely identifies each item in the table, allowing for quick access to data. DynamoDB supports two types of primary keys:
    - **Simple Primary Key**: Composed of a single attribute (Partition Key).
    - **Composite Primary Key**: Consists of two attributes (Partition Key and Sort Key).

  - **Streams**: DynamoDB Streams capture a time-ordered sequence of item-level modifications in any DynamoDB table and store this information for up to 24 hours. Applications can access this stream and view, update, or delete items in a table.

  - **DynamoDB Accelerator (DAX)**: An in-memory cache for DynamoDB tables that delivers fast read performance for your tables at scale by reducing the response times from milliseconds to microseconds, even at millions of requests per second.

  - **Partitions**: DynamoDB automatically spreads the data across multiple storage partitions for scalability and performance. Each partition is a self-contained unit of storage and computing.

AWS DynamoDB Architecture

- <u>**Key Architectural Features**</u>
  - **Primary Key**: The primary key uniquely identifies each item in the table, allowing for quick access to data. DynamoDB supports two types of primary keys:
    - **Simple Primary Key**: Composed of a single attribute (Partition Key).
    - **Composite Primary Key**: Consists of two attributes (Partition Key and Sort Key).

  - **Streams**: DynamoDB Streams capture a time-ordered sequence of item-level modifications in any DynamoDB table and store this information for up to 24 hours. Applications can access this stream and view, update, or delete items in a table.

  - **DynamoDB Accelerator (DAX)**: An in-memory cache for DynamoDB tables that delivers fast read performance for your tables at scale by reducing the response times from milliseconds to microseconds, even at millions of requests per second.

  - **Partitions**: DynamoDB automatically spreads the data across multiple storage partitions for scalability and performance. Each partition is a self-contained unit of storage and computing.