**Kinesis Overview**

* Processes and analyses large amounts of data in real-time, as it's generated.
* Collects data from various sources like application logs, website traffic, IoT devices, and video streams.

**Key components:**

* **Kinesis Data Streams:** Captures, processes, and stores data streams for later analysis.
* **Kinesis Data Firehose:**Delivers data streams to other AWS services like data lakes and analytics platforms.
* **Kinesis Data Analytics:** Analyses data streams using SQL or Apache Flink for real-time insights.
* **Kinesis Video Streams:** Captures, processes, and stores video streams for analytics, machine learning, and playback.

**Benefits:**

* **Scalability:** Handles large data volumes without compromising performance.
* **Real-time insights:** Gain insights from data as it's generated, enabling quicker decision-making.
* **Flexibility:** Supports various data formats and integrates with different AWS services.
* **Cost-effectiveness:** Pay only for the resources you use.

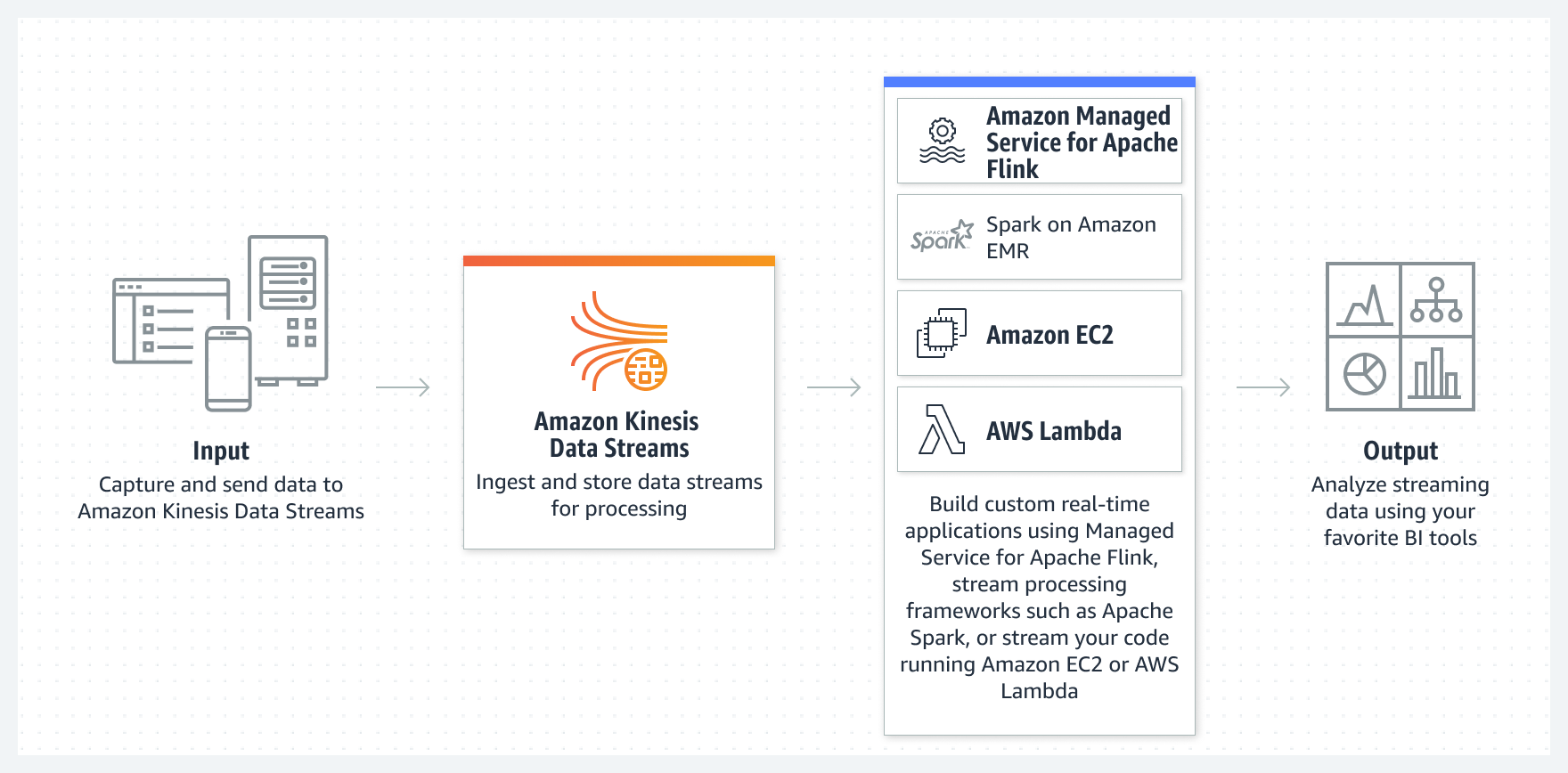
**Note:**

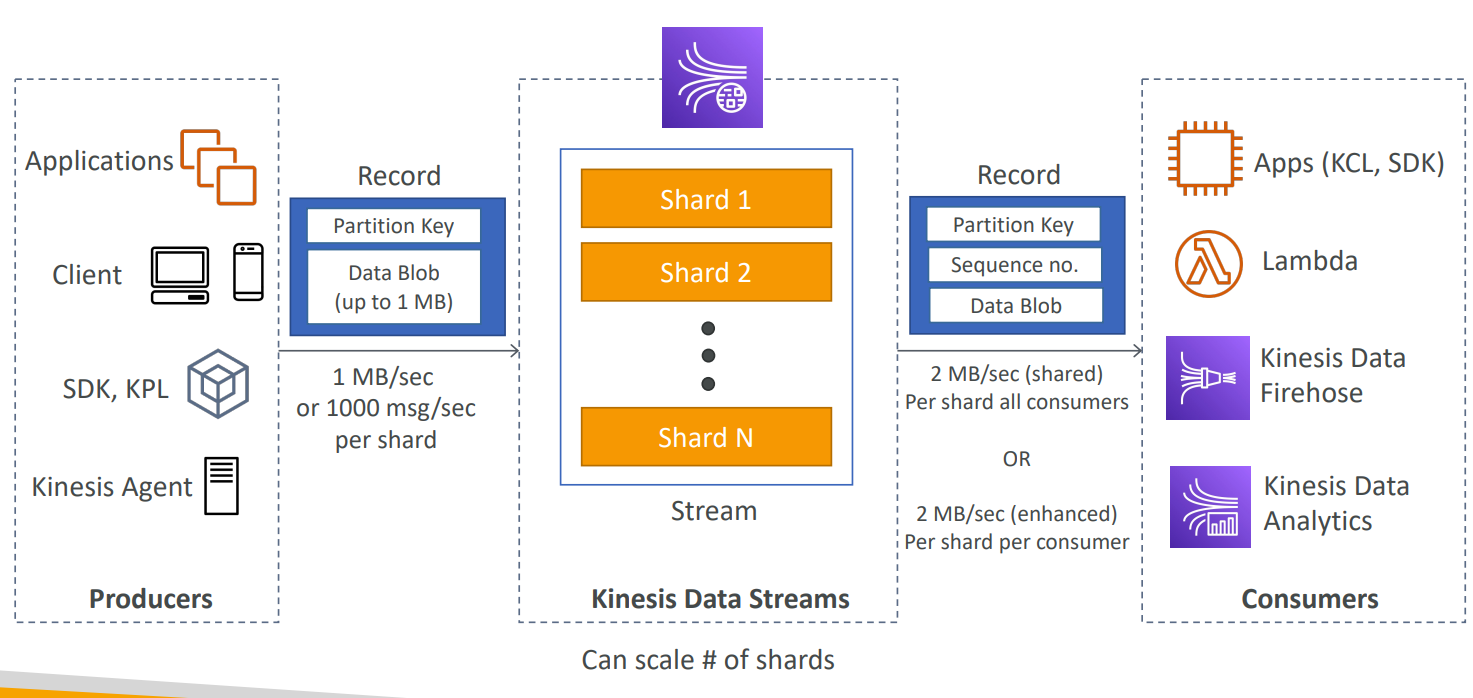
* Kinesis is a managed service, so you don't need to manage the underlying infrastructure.
* It's widely used for various applications like fraud detection, stock market analysis, and IoT analytics.

**Kinesis Data Streams**

**Data Management:**

* **Retention:** Data can be stored between 1 day and 365 days.
* **Reprocessing:** You can replay data for further analysis.
* **Immutability:** Once data is inserted, it cannot be deleted.
* **Partitioning:**Data with the same partition key goes to the same shard (maintains order).





**Producers and Consumers:**

* **Producers:** Use AWS SDK, Kinesis Producer Library (KPL), or Kinesis Agent to send data.
* **Consumers:**
  + **Custom consumers:** Develop your own using Kinesis Client Library (KCL) or AWS SDK.
  + **Managed consumers:** Use AWS Lambda, Kinesis Data Firehose, or Kinesis Data Analytics.

**Capacity Modes:**

* **Provisioned mode:**

Choose the number of shards (fixed capacity), scale manually or automatically.

* + Each shard has 1MB/s write and 2MB/s read capacity.
  + Pay per shard per hour.
* **On-demand mode:**

No need to provision, scales automatically.

* + Default 4MB/s write and 4000 records/s read capacity.
  + Pay per stream per hour and data processed.

**Security:**

* **IAM policies:** Control access and authorization.
* **Encryption:** Secure data in transit with HTTPS and at rest with KMS.
* **Client-side encryption:** Optional but more complex.
* **VPC Endpoints:** Access Kinesis securely within your VPC.
* **CloudTrail:** Monitor API calls for security purposes.

For more details, refer to the official documentation: [Real-Time Streaming Analytics - Amazon Kinesis Data Streams - AWS](https://aws.amazon.com/kinesis/data-streams/?p=pm&c=aa&pd=kinesis&z=4)

**Kinesis Data Streams Producers**

**Purpose:**

* Put data records into data streams for processing and analysis.

**Data Record Structure:**

* **Sequence number:** Unique identifies a record within a partition key & shard.
* **Partition key:** Determines which shard a record belongs to, ensuring related data is grouped together.
* **Data blob:** The actual data, up to 1 MB in size.

**Producer Options:**

* **AWS SDK:** Basic producer for simple use cases.
* **Kinesis Producer Library (KPL):** Offers advanced features like batching, compression, and retries. Available for C++ and Java.
* **Kinesis Agent:** Monitors log files and sends data to Kinesis streams automatically.

**Throughput:**

* **Write throughput per shard:** 1 MB/sec or 1000 records/sec.
* **APIs:**
  + **PutRecord:** Puts a single record into a stream.
  + **PutRecords:** Batches multiple records for efficiency (recommended for cost savings and higher throughput).

**ProvisionedThroughputExceeded Error:**

* Error indicating a producer is exceeding the capacity of a shard.
* Occurs when write throughput exceeds 1MB/sec or 1000 records/sec per shard.
* Solutions:
  + Increase the number of shards (horizontal scaling)
  + Use batching with PutRecords to reduce API calls.
  + Implement exponential backoff and retries in your producer code.

**Kinesis Data Streams Consumers**

* Retrieve and process data records from Kinesis data streams.
* Available as managed services or you can build your own custom consumers.

**Managed Consumers**

* **AWS Lambda:** Serverless function triggered by Kinesis events, good for small, stateless tasks.
* **Kinesis Data Analytics**: Analyzes data using SQL or Apache Flink, suitable for complex transformations.
* **Kinesis Data Firehose:** Delivers data to other AWS services like S3 or Redshift.

**Custom Consumer**

* **AWS SDK:** Build your own consumer with more control, using Classic or Enhanced Fan-out.
* **Kinesis Client Library (KCL):** Simplifies reading data with built-in features.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Shared (Classic) Fan-out** | **Enhanced Fan-out** |
| Read Throughput | 2 MB/sec shared across all consumers | 2 MB/sec per consumer |
| Latency | ~200 ms | ~70 ms |
| Cost | Lower | Higher |
| Scaling | Manual | Automatic |
| Data Access | Pull (GetRecords API) | Push (SubscribeToShard API) |

**Visual representations:**

* **Classic Fan-out**: Imagine multiple consumers sharing a single pipe with limited flow.
* **Enhanced Fan-out:** Imagine each consumer having its own dedicated pipe with full flow.

**Kinesis Consumers – AWS Lambda**

**•** Supports Classic & Enhanced fan-out consumers

**•** Read records in batches

• Can configure batch size and batch window

• If error occurs, Lambda retries until succeeds or data expired

• Can process up to 10 batches per shard simultaneously

**Kinesis Client Library (KCL)**

**Purpose:** Simplifies reading data from Kinesis Data Streams in distributed applications.

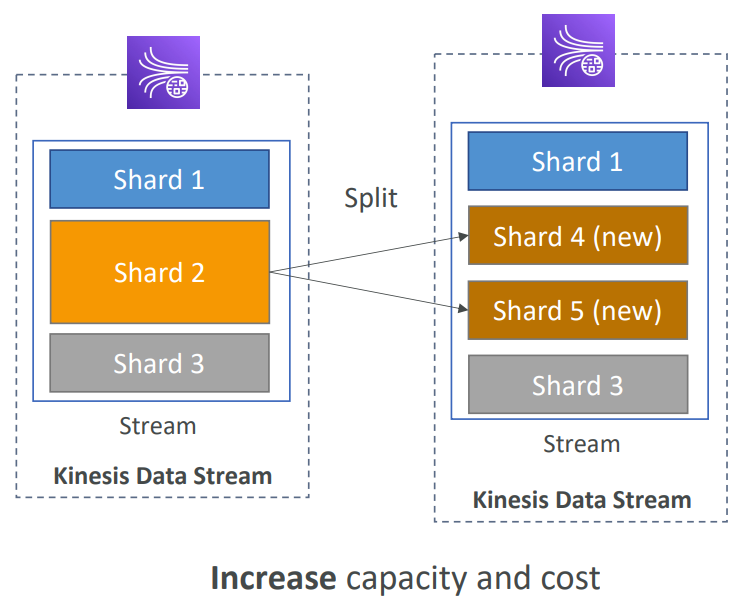
**Key Features:**

* Java library
* **Shard-level parallelism:** Each shard is read by one KCL instance, enabling horizontal scaling.
* **Checkpointing:** Records progress in DynamoDB for fault tolerance and recovery.
* **Work coordination:** Uses DynamoDB to track workers and distribute shards for load balancing.
* **Deployment flexibility:** Runs on EC2, Elastic Beanstalk, or on-premises.
* **Ordered reads:** Ensures records are processed in order within each shard.
* **Versioning:** KCL 1.x supports shared fan-out, while KCL 2.x supports both shared and enhanced fan-out.

**Notes:**

* **IAM access:** KCL requires IAM permissions for DynamoDB access.
* **Shards:** Number of KCL instances should match the number of shards for optimal performance.

**Kinesis Operation – Shard Splitting**

****

**Purpose:**

* Increase the capacity of a Kinesis data stream (each shard has a limit of 1 MB/s data in or 1000 records/s).
* Divide a shard that is receiving too much data (“hot shard”).

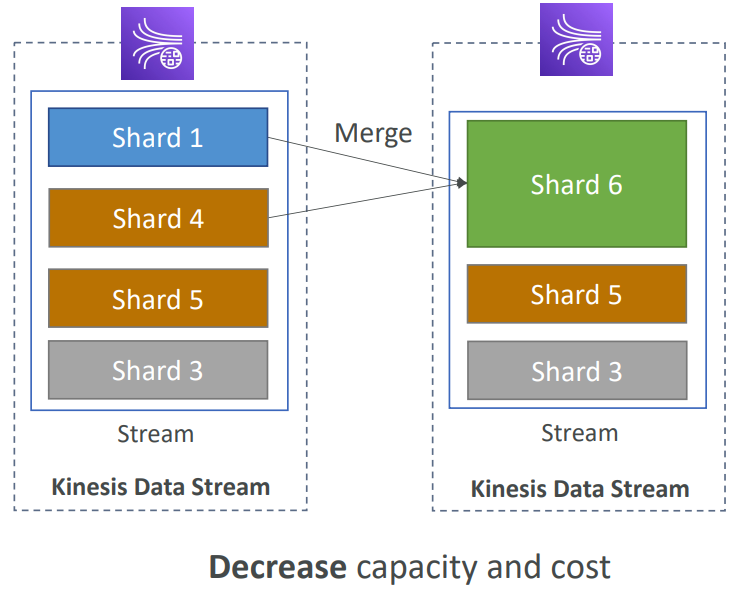
**Process:**

* The old shard is closed and stops accepting new data.
* The old shard’s data is split into two new shards.
* The new shards become active and start accepting new data.
* The old shard is eventually deleted once its data expires (retention period).

**Key Considerations:**

* Shard splitting is a manual process. There is no automatic scaling in Kinesis.
* You can only split a shard into two new shards in a single operation.

**Kinesis Operation - Shard Merging**

****

* **Purpose:**
  + Decrease the capacity of a Kinesis data stream and save costs.
  + Combine two shards with low data traffic (“cold shards”) into one shard.
* **Process:**
  + The two shards you want to merge are closed and stop accepting new data.
  + The data from the two closed shards is combined into a single new shard.
  + The new shard becomes active and starts accepting new data.
  + The old shards are eventually deleted once their data expires (retention period).
* **Limitations:**
  + Shard merging is a manual process. There is no automatic scaling in Kinesis.
  + You can only merge two shards at a time.
  + The shards you want to merge must be adjacent (their hash key ranges must be consecutive).

**Kinesis Data Firehose**

* **Purpose:** Load streaming data into various destinations like S3, Redshift, OpenSearch, etc.
* **Key features:**
  + **Fully managed:** No need to manage infrastructure or scaling.
  + **Automatic scaling:** Automatically adjusts to data volume.
  + **Near real-time:** Delivery within 60 seconds for non-full batches.
  + Supports various data formats and transformations.
  + **Custom Lambda transformations:** Perform complex data manipulation.
  + **Backup to S3**: Send failed or all data to a backup S3 bucket
* **Considerations:**
  + **Near real-time, not real-time:** Data might have a slight delay.
  + **No data storage or replay:** Data is delivered to the destination, not stored within Firehose.
* **Cost:** Pay per data processed and sent through Firehose.

|  |  |  |
| --- | --- | --- |
| **Features** | **Kinesis Data Streams** | **Kinesis Data Firehose** |
| Purpose | Real-time data ingestion, processing, and analysis | Load data into data stores or analytics platforms |
| Management | Self-managed | Fully managed |
| Scaling | Manual (shard splitting/merging) | Automatic |
| Latency | Real-time (~200 ms) | Near real-time (min. 60 seconds) |
| Data Storage | Yes (1-365 days) | None |
| Replay Capability | Yes | No |
| Destinations | Custom applications, AWS services, etc. | S3, Redshift, OpenSearch, 3rd party, custom HTTP |
| Cost | Pay per shard per hour + data in/out | Pay per data processed |

**Kinesis Data Analytics for SQL applications**

* **Real-time analytics:** Processes streaming data from Kinesis Data Streams and Firehose using SQL.
* **Reference data:** Enriches streams with data from Amazon S3.
* **Managed service:** No server provisioning or scaling needed.
* **Cost:** Pay for actual usage.
* **Output:** Sends results to Kinesis Data Streams or Firehose.
* **Use cases:**Time-series analytics, real-time dashboards, real-time metrics.

**Kinesis Data Analytics for Apache Flink**

* **Flexibility:** Uses Apache Flink (Java, Scala, or SQL) for advanced processing and analysis.
* **Managed cluster:** Runs Flink applications on managed AWS cluster.
* **Scalability:**Automatically scales compute resources and supports parallel computation.
* **Backups:** Checkpoints and snapshots for application resilience.
* **Features:** Access to all Apache Flink programming capabilities.
* **Limitation:** Can't read directly from Kinesis Firehose (use Kinesis Analytics for SQL instead)