Amazon Kinesis Data Streams FAQs

General

Q: What is Amazon Kinesis Data Streams?

Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs. You can continuously add various types of data such as clickstreams, application logs, and social media to an Amazon Kinesis data stream from hundreds of thousands of sources. Within seconds, the data will be available for your Amazon Kinesis Applications to read and process from the stream.

Q: What does Amazon Kinesis Data Streams manage on my behalf?

Amazon Kinesis Data Streams manages the infrastructure, storage, networking, and configuration needed to stream your data at the level of your data throughput. You do not have to worry about provisioning, deployment, ongoing-maintenance of hardware, software, or other services for your data streams. In addition, Amazon Kinesis Data Streams synchronously replicates data across three availability zones, providing high availability and data durability.

Q: What can I do with Amazon Kinesis Data Streams?

Amazon Kinesis Data Streams is useful for rapidly moving data off data producers and then continuously processing the data, be it to transform the data before emitting to a data store, run real-time metrics and analytics, or derive more complex data streams for further processing. The following are typical scenarios for using Amazon Kinesis Data Streams:

- Accelerated log and data feed intake: Instead of waiting to batch up the data, you can have your data producers push data to an Amazon Kinesis data stream as soon as the data is produced, preventing data loss in case of data producer failures. For example, system and application logs can be continuously added to a data stream and be available for processing within seconds.
- Real-time metrics and reporting: You can extract metrics and generate reports from Amazon Kinesis data stream data in real-time. For example, your Amazon Kinesis Application can work on metrics and reporting for system and application logs as the data is streaming in, rather than wait to receive data batches.
- Real-time data analytics: With Amazon Kinesis Data Streams, you can run real-time streaming data analytics. For example, you can add clickstreams to your Amazon Kinesis data stream and have your Amazon Kinesis Application run analytics in real-time, enabling you to gain insights out of your data at a scale of minutes instead of hours or days.
- Complex stream processing: You can create Directed Acyclic Graphs (DAGs) of Amazon Kinesis Applications and data streams. In this scenario, one or more Amazon Kinesis Applications can add data to another Amazon Kinesis data stream for further processing, enabling successive stages of stream processing.

Q: How do I use Amazon Kinesis Data Streams?

After you sign up for Amazon Web Services, you can start using Amazon Kinesis Data Streams by:

- Creating an Amazon Kinesis data stream through either AWS Management Console or CreateStream operation.
- Configuring your data producers to continuously add data to your data stream.
- Building your Amazon Kinesis Applications to read and process data from your data stream, using either Amazon Kinesis API or Amazon Kinesis Client Library (KCL).

Q: What are the limits of Amazon Kinesis Data Streams?

The throughput of an Amazon Kinesis data stream is designed to scale without limits via increasing the number of shards within a data stream. However, there are certain limits you should keep in mind while using Amazon Kinesis Data Streams:

- By default, Records of a stream are accessible for up to 24 hours from the time they are added to the stream. You can raise this limit to up to 7 days by enabling extended data retention.
- The maximum size of a data blob (the data payload before Base64-encoding) within one record is 1 megabyte (MB).
- Each shard can support up to 1000 PUT records per second.

For more information about other API level limits, see Amazon Kinesis Data Streams Limits.

Q: How does Amazon Kinesis Data Streams differ from Amazon SQS?

Amazon Kinesis Data Streams enables real-time processing of streaming big data. It provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Amazon Kinesis Applications. The Amazon Kinesis Client Library (KCL) delivers all records for a given partition key to the same record processor, making it easier to build multiple applications reading from the same Amazon Kinesis data stream (for example, to perform counting, aggregation, and filtering).

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable hosted queue for storing messages as they travel between computers. Amazon SQS lets you easily move data between distributed application components and helps you build applications in which messages are processed independently (with message-level ack/fail semantics), such as automated workflows.

Q: When should I use Amazon Kinesis Data Streams, and when should I use Amazon SQS?

We recommend Amazon Kinesis Data Streams for use cases with requirements that are similar to the following:

- Routing related records to the same record processor (as in streaming MapReduce). For example, counting and aggregation are simpler when all records for a given key are routed to the same record processor.
- Ordering of records. For example, you want to transfer log data from the application host to the processing/archival host while maintaining the order of log statements.
- Ability for multiple applications to consume the same stream concurrently. For example, you have one application that updates a real-time dashboard and another that archives data to Amazon Redshift. You want both applications to consume data from the same stream concurrently and independently.
- Ability to consume records in the same order a few hours later. For example, you have a billing application and an audit application that runs a few hours behind the billing application. Because Amazon Kinesis Data Streams stores data for up to 7 days, you can run the audit application up to 7 days behind the billing application.

We recommend Amazon SQS for use cases with requirements that are similar to the following:

- Messaging semantics (such as message-level ack/fail) and visibility timeout. For example, you have a queue of work items and want to track the successful completion of each item independently. Amazon SQS tracks the ack/fail, so the application does not have to maintain a persistent checkpoint/cursor. Amazon SQS will delete acked messages and redeliver failed messages after a configured visibility timeout.
- Individual message delay. For example, you have a job queue and need to schedule individual jobs with a delay. With Amazon SQS, you can configure individual messages to have a delay of up to 15 minutes.
- Dynamically increasing concurrency/throughput at read time. For example, you have a work queue and want to add more readers until the backlog is cleared. With Amazon Kinesis Data Streams, you can scale up to a sufficient number of shards (note, however, that you'll need to provision enough shards ahead of time).
- Leveraging Amazon SQS's ability to scale transparently. For example, you buffer requests and the load changes as a result of occasional load spikes or the natural growth of your business. Because each buffered request can be processed independently, Amazon SQS can scale transparently to handle the load without any provisioning instructions from you.

Key concepts

Q: What is a shard?

Shard is the base throughput unit of an Amazon Kinesis data stream. One shard provides a capacity of 1MB/sec data input and 2MB/sec data output. One shard can support up to 1000 PUT records per second. You will specify the number of shards needed when you create a data stream. For example, you can create a data stream with two shards. This data stream has a throughput of 2MB/sec data input and 4MB/sec data output, and allows up to 2000 PUT records per second. You can monitor shard-level metrics in Amazon Kinesis Data Streams and add or remove shards from your data stream dynamically as your data throughput changes by resharding the data stream.

Q: What is a record?

A record is the unit of data stored in an Amazon Kinesis data stream. A record is composed of a sequence number, partition key, and data blob. Data blob is the data of interest your data producer adds to a data stream. The maximum size of a data blob (the data payload before Base64-encoding) is 1 megabyte (MB).

Q: What is a partition key?

Partition key is used to segregate and route records to different shards of a data stream. A partition key is specified by your data producer while adding data to an Amazon Kinesis data stream. For example, assuming you have a data stream with two shards (shard 1 and shard 2). You can configure your data producer to use two partition keys (key A and key B) so that all records with key A are added to shard 1 and all records with key B are added to shard 2.

Q: What is a sequence number?

A sequence number is a unique identifier for each record. Sequence number is assigned by Amazon Kinesis when a data producer calls PutRecord or PutRecords operation to add data to an Amazon Kinesis data stream. Sequence numbers for the same partition key generally increase over time; the longer the time period between PutRecord or PutRecords requests, the larger the sequence numbers become.

Creating data streams

Q: How do I create an Amazon Kinesis data stream?

After you sign up for Amazon Web Services, you can create an Amazon Kinesis data stream through either Amazon Kinesis Management Console or CreateStream operation.

Q: How do I decide the throughput of my Amazon Kinesis data stream?

The throughput of an Amazon Kinesis data stream is determined by the number of shards within the data stream. Follow the steps below to estimate the initial number of shards your data stream needs. Note that you can dynamically adjust the number of shards within your data stream via resharding.

1. Estimate the average size of the record written to the data stream in kilobytes (KB), rounded up to the nearest 1 KB. (average_data_size_in_KB)
2. Estimate the number of records written to the data stream per second. (number_of_records_per_second)
3. Decide the number of Amazon Kinesis Applications consuming data concurrently and independently from the data stream. (number_of_consumers)

4. Calculate the incoming write bandwidth in KB (incoming_write_bandwidth_in_KB), which is equal to the average_data_size_in_KB multiplied by the number_of_records_per_seconds.
5. Calculate the outgoing read bandwidth in KB (outgoing_read_bandwidth_in_KB), which is equal to the incoming_write_bandwidth_in_KB multiplied by the number_of_consumers.

You can then calculate the initial number of shards (number_of_shards) your data stream needs using the following formula:

number_of_shards = max (incoming_write_bandwidth_in_KB/1000, outgoing_read_bandwidth_in_KB/2000)

Q: What is the minimum throughput I can request for my Amazon Kinesis data stream?

The throughput of an Amazon Kinesis data stream scales by unit of shard. One single shard is the smallest throughput of a data stream, which provides 1MB/sec data input and 2MB/sec data output.

Q: What is the maximum throughput I can request for my Amazon Kinesis data stream?

The throughput of an Amazon Kinesis data stream is designed to scale without limits. By default, each account can provision 10 shards per region. You can use the Amazon Kinesis Data Streams Limits form to request more than 10 shards within a single region.

Q: How can record size affect the throughput of my Amazon Kinesis data stream?

A shard provides 1MB/sec data input rate and supports up to 1000 PUT records per sec. Therefore, if the record size is less than 1KB, the actual data input rate of a shard will be less than 1MB/sec, limited by the maximum number of PUT records per second.

Adding data to Kinesis data streams

Q: How do I add data to my Amazon Kinesis data stream?

You can add data to an Amazon Kinesis data stream via PutRecord and PutRecords operations, Amazon Kinesis Producer Library (KPL), or Amazon Kinesis Agent.

Q: What is the difference between PutRecord and PutRecords?

PutRecord operation allows a single data record within an API call and PutRecords operation allows multiple data records within an API call. For more information about PutRecord and PutRecords operations, see PutRecord and PutRecords.

Q: What is Amazon Kinesis Producer Library (KPL)?

Amazon Kinesis Producer Library (KPL) is an easy to use and highly configurable library that helps you put data into an Amazon Kinesis data stream. KPL presents a simple, asynchronous, and reliable interface that enables you to quickly achieve high producer throughput with minimal client resources.

Q: What programming languages or platforms can I use to access Amazon Kinesis API?

Amazon Kinesis API is available in Amazon Web Services SDKs. For a list of programming languages or platforms for Amazon Web Services SDKs, see Tools for Amazon Web Services.

Q: What programming language is Amazon Kinesis Producer Library (KPL) available in?

Amazon Kinesis Producer Library (KPL)'s core is built with C++ module and can be compiled to work on any platform with a recent C++ compiler. The library is currently available in a Java interface. We are looking to add support for other programming languages.

Q: What is Amazon Kinesis Agent?

Amazon Kinesis Agent is a pre-built Java application that offers an easy way to collect and send data to your Amazon Kinesis data stream. You can install the agent on Linux-based server environments such as web servers, log servers, and database servers. The agent monitors certain files and continuously sends data to your data stream. For more information, see Writing with Agents.

Q: What platforms do Amazon Kinesis Agent support?

Amazon Kinesis Agent currently supports Amazon Linux or Red Hat Enterprise Linux.

Q: Where do I get Amazon Kinesis Agent?

You can download and install Amazon Kinesis Agent using the following command and link:

On Amazon Linux: sudo yum install –y aws-kinesis-agent

On Red Hat Enterprise Linux: sudo yum install –y https://s3.amazonaws.com/streaming-data-agent/aws-kinesis-agent-latest.amzn1.noarch.rpm

From GitHub: awlabs/amazon-kinesis-agent

Q: How do I use Amazon Kinesis Agent?

After installing Amazon Kinesis Agent on your servers, you configure it to monitor certain files on the disk and then continuously send new data to your Amazon Kinesis data stream. For more information, see Writing with Agents.

Q: What happens if the capacity limits of an Amazon Kinesis data stream are exceeded while the data producer adds data to the data stream?

The capacity limits of an Amazon Kinesis data stream are defined by the number of shards within the data stream. The limits can be exceeded by either data throughput or the number of

PUT records. While the capacity limits are exceeded, the put data call will be rejected with a ProvisionedThroughputExceeded exception. If this is due to a temporary rise of the data stream's input data rate, retry by the data producer will eventually lead to completion of the requests. If this is due to a sustained rise of the data stream's input data rate, you should increase the number of shards within your data stream to provide enough capacity for the put data calls to consistently succeed. In both cases, Amazon CloudWatch metrics allow you to learn about the change of the data stream's input data rate and the occurrence of ProvisionedThroughputExceeded exceptions.

Q: What data is counted against the data throughput of an Amazon Kinesis data stream during a PutRecord or PutRecords call?

Your data blob, partition key, and data stream name are required parameters of a PutRecord or PutRecords call. The size of your data blob (before Base64 encoding) and partition key will be counted against the data throughput of your Amazon Kinesis data stream, which is determined by the number of shards within the data stream.

Enhanced fan-out

Q: What is enhanced fan-out?

Enhanced fan-out is an optional feature for Kinesis Data Streams consumers that provides logical 2 MB/sec throughput pipes between consumers and shards. This allows customers to scale the number of consumers reading from a data stream in parallel, while maintaining high performance.

Q: How do consumers use enhanced fan-out?

Consumers must first register themselves with the Kinesis Data Streams service. By default, consumer registration activates enhanced fan-out. If you are using the KCL, KCL version 2.x takes care of registering your consumers automatically, and uses the name of the KCL application as the consumer name. Once registered, all registered consumers will have their own logical enhanced fan-out throughput pipes provisioned for them. Then, consumers use the HTTP/2 SubscribeToShard API to retrieve data inside of these throughput pipes. The HTTP/1 GetRecords API does not currently support enhanced fan-out, so you will need to upgrade to KCL 2.x, or alternatively register your consumer and have the consumer call the SubscribeToShard API.

Q: How is enhanced fan-out utilized by a consumer?

Consumers utilize enhanced fan-out by retrieving data with the SubscribeToShard API. The name of the registered consumer is used within the SubscribeToShard API, which leads to utilization of the enhanced fan-out benefit provided to the registered consumer.

Q: When should I use enhanced fan-out?
You should use enhanced fan-out if you have, or expect to have, multiple consumers retrieving data from a stream in parallel, or if you have at least one consumer that requires the use of the SubscribeToShard API to provide sub-200ms data delivery speeds between producers and consumers.

Q: Can I have consumers using enhanced fan-out, and others not?

Yes, you can have multiple consumers using enhanced fan-out and others not using enhanced fan-out at the same time. The use of enhanced fan-out does not impact the limits of shards for traditional GetRecords usage.

Q: Is there a limit on the number of consumers using enhanced fan-out on a given stream?

There is a default limit of 5 consumers using enhanced fan-out per data stream. If you need more than 5, please submit a limit increase request though AWS support. Keep in mind that you can have more than 5 total consumers reading from a stream by having 5 consumers using enhanced fan-out and other consumers not using enhanced fan-out at the same time.

Q: How do consumers register to use enhanced fan-out and the HTTP/2 SubscribeToShard API?

We recommend using KCL 2.x, which will automatically register your consumer and use both enhanced fan-out and the HTTP/2 SubscribeToShard API. Otherwise, you can manually register a consumer using the RegisterStreamConsumer API and then you can use the SubscribeToShard API with the name of the consumer you registered.

Q: Is there a cost associated with the use of enhanced fan-out?

Yes, there is an on-demand hourly cost for every combination of shard in a stream and consumer (a consumer-shard hour) registered to use enhanced fan-out, in addition to a data retrieval cost for every GB retrieved. See the Kinesis Data Streams pricing page for more details.

Q: How is a consumer-shard hour calculated?

A consumer-shard hour is calculated by multiplying the number of registered stream consumers with the number of shards in the stream. For example, if a consumer-shard hour costs $0.015, for a 10 shard data stream, this consumer using enhanced fan-out would be able to read from 10 shards, and thus incur a consumer-shard hour charge of $0.15 per hour (1 consumer x 10 shards x $0.015 per consumers-shard hour). If there were two consumers registered for enhanced fan-out simultaneously, the total consumer-shard hour charge would be $0.30 per hour (2 consumers x 10 shards x $0.015).

Q: Does consumer-shard hour billing for enhanced fan-out automatically prorate if I terminate or start a consumer within the hour?

Yes, you will only pay for the prorated portion of the hour the consumer was registered to use enhanced fan-out.

Q: How does billing for enhanced fan-out data retrievals work?

You pay a low per GB rate that is metered per byte of data retrieved by consumers using enhanced fan-out. There is no payload roundup or delivery minimum.

Q: Do I need to change my producers or my data stream to use enhanced fan-out?

No, enhance fan-out can be activated without impacting data producers or data streams.

Reading and processing data from Kinesis data streams

Q: What is an Amazon Kinesis Application?

An Amazon Kinesis Application is a data consumer that reads and processes data from an Amazon Kinesis data stream. You can build your applications using either Amazon Kinesis Data Analytics, Amazon Kinesis API or Amazon Kinesis Client Library (KCL).

Q: What is Amazon Kinesis Client Library (KCL)?

Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET is a pre-built library that helps you easily build Amazon Kinesis Applications for reading and processing data from an Amazon Kinesis data stream.

KCL handles complex issues such as adapting to changes in data stream volume, load-balancing streaming data, coordinating distributed services, and processing data with fault-tolerance. KCL enables you to focus on business logic while building applications. KCL 2.x supports both the HTTP/1 GetRecords and HTTP/2 SubscribeToShard APIs with enhanced fan-out for retrieving data from a stream. KCL 1.x does not support the SubscribeToShard API or enhanced fan-out.

Q: How do I upgrade from KCL 1.x to 2.x to use SubscribeToShard and enhanced fan-out?

Visit the Kinesis Data Streams user documentation to learn how to upgrade from KCL 1.x to KCL 2.x.

Q: What is the SubscribeToShard API?

The SubscribeToShard API is a high performance streaming API that pushes data from shards to consumers over a persistent connection without a request cycle from the client. The SubscribeToShard API uses the HTTP/2 protocol to deliver data to registered consumers whenever new data arrives on the shard, typically within 70ms, offering ~65% faster delivery compared to the GetRecords API.. The consumers will enjoy fast delivery even when multiple registered consumers are reading from the same shard.

Q: Can I use SubscribeToShard without using enhanced fan-out?

No, SubscribeToShard requires the use of enhanced fan-out, which means you also need to register your consumer with the Kinesis Data Streams service before you can use SubscribeToShard.

Q: How long does the SubscribeToShard persistent connection last?

The persistent connection can last up to 5 minutes.

Q: Does the Kinesis Client Library (KCL) support SubscribeToShard?

Yes, version 2.x of the KCL uses SubscribeToShard and enhanced fan-out to retrieve data with high performance from a Kinesis data stream.

Q: Is there a cost associated with using SubscribeToShard?

No, there is no additional cost associated with SubscribeToShard, but you must use SubscribeToShard with enhanced fan-out which does have an additional hourly cost for each consumer-shard combination and per GB of data delivered by enhanced fan-out.

Q: Do I need to use enhanced fan-out if I want to use SubscribeToShard?

Yes, to use SubscribeToShard you need to register your consumers, and registration activates enhanced fan-out. By default, your consumer will utilize enhanced fan-out automatically when data is retrieved via SubscribeToShard.

Q: What is Amazon Kinesis Connector Library?

Amazon Kinesis Connector Library is a pre-built library that helps you easily integrate Amazon Kinesis Data Streams with other AWS services and third-party tools. Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET is required for using Amazon Kinesis Connector Library. The current version of this library provides connectors to Amazon DynamoDB, Amazon Redshift, Amazon S3, and Elasticsearch. The library also includes sample connectors of each type, plus Apache Ant build files for running the samples.

Q: What is Amazon Kinesis Storm Spout?

Amazon Kinesis Storm Spout is a pre-built library that helps you easily integrate Amazon Kinesis Data Streams with Apache Storm. The current version of Amazon Kinesis Storm Spout fetches data from Amazon Kinesis data stream and emits it as tuples. You will add the spout to your Storm topology to leverage Amazon Kinesis Data Streams as a reliable, scalable, stream capture, storage, and replay service.

Q: What programming language are Amazon Kinesis Client Library (KCL), Amazon Kinesis Connector Library, and Amazon Kinesis Storm Spout available in?

Amazon Kinesis Client Library (KCL) is currently available in Java, Python, Ruby, Node.js, and .NET. Amazon Kinesis Connector Library and Amazon Kinesis Storm Spout are currently available in Java. We are looking to add support for other programming languages.

Q: Do I have to use Amazon Kinesis Client Library (KCL) for my Amazon Kinesis Application?

No, you can also use Amazon Kinesis API to build your Amazon Kinesis Application. However, we recommend using Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET if applicable because it performs heavy-lifting tasks associated with distributed stream processing, making it more productive to develop applications.

Q: How does Amazon Kinesis Client Library (KCL) interact with an Amazon Kinesis Application?

Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET acts as an intermediary between Amazon Kinesis Data Streams and your Amazon Kinesis Application. KCL uses the IRecordProcessor interface to communicate with your application. Your application implements this interface, and KCL calls into your application code using the methods in this interface.

For more information about building application with KCL, see Developing Consumer Applications for Amazon Kinesis Using the Amazon Kinesis Client Library.

Q: What is a worker and a record processor generated by Amazon Kinesis Client Library (KCL)?

An Amazon Kinesis Application can have multiple application instances and a worker is the processing unit that maps to each application instance. A record processor is the processing unit that processes data from a shard of an Amazon Kinesis data stream. One worker maps to one or more record processors. One record processor maps to one shard and processes records from that shard.

At startup, an application calls into Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET to instantiate a worker. This call provides KCL with configuration information for the application, such as the data stream name and AWS credentials. This call also passes a reference to an IRecordProcessorFactory implementation. KCL uses this factory to create new record processors as needed to process data from the data stream. KCL communicates with these record processors using the IRecordProcessor interface.

Q: How does Amazon Kinesis Client Library (KCL) keep tracking data records being processed by an Amazon Kinesis Application?

Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET automatically creates an Amazon DynamoDB table for each Amazon Kinesis Application to track and maintain state information such as resharding events and sequence number checkpoints. The DynamoDB table shares the same name with the application so that you need to make sure your application name doesn’t conflict with any existing DynamoDB tables under the same account within the same region.

All workers associated with the same application name are assumed to be working together on the same Amazon Kinesis data stream. If you run an additional instance of the same application code, but with a different application name, KCL treats the second instance as an entirely separate application also operating on the same data stream.

Please note that your account will be charged for the costs associated with the Amazon DynamoDB table in addition to the costs associated with Amazon Kinesis Data Streams.

For more information about how KCL tracks application state, see Tracking Amazon Kinesis Application state.

Q: How can I automatically scale up the processing capacity of my Amazon Kinesis Application using Amazon Kinesis Client Library (KCL)?

You can create multiple instances of your Amazon Kinesis Application and have these application instances run across a set of Amazon EC2 instances that are part of an Auto Scaling group. While the processing demand increases, an Amazon EC2 instance running your application instance will be automatically instantiated. Amazon Kinesis Client Library (KCL) for Java | Python | Ruby | Node.js | .NET will generate a worker for this new instance and automatically move record processors from overloaded existing instances to this new instance.

Q: Why does GetRecords call return empty result while there is data within my Amazon Kinesis data stream?

One possible reason is that there is no record at the position specified by the current shard iterator. This could happen even if you are using TRIM_HORIZON as shard iterator type. An Amazon Kinesis data stream represents a continuous stream of data. You should call GetRecords operation in a loop and the record will be returned when the shard iterator advances to the position where the record is stored.

Q: What is ApproximateArrivalTimestamp returned in GetRecords operation?

Each record includes a value called ApproximateArrivalTimestamp. It is set when the record is successfully received and stored by Amazon Kinesis. This timestamp has millisecond precision and there are no guarantees about the timestamp accuracy. For example, records in a shard or across a data stream might have timestamps that are out of order.

Q: What happens if the capacity limits of an Amazon Kinesis data stream are exceeded while Amazon Kinesis Application reads data from the data stream?

The capacity limits of an Amazon Kinesis data stream are defined by the number of shards within the data stream. The limits can be exceeded by either data throughput or the number of read data calls. While the capacity limits are exceeded, the read data call will be rejected with a ProvisionedThroughputExceeded exception. If this is due to a temporary rise of the data stream’s output data rate, retry by the Amazon Kinesis Application will eventually lead to completions of the requests. If this is due to a sustained rise of the data stream’s output data rate, you should increase the number of shards within your data stream to provide enough capacity for the read data calls to consistently succeed. In both cases, Amazon CloudWatch metrics allow you to learn about the change of the data stream’s output data rate and the occurrence of ProvisionedThroughputExceeded exceptions.

Managing Kinesis data streams

Q: How do I change the throughput of my Amazon Kinesis data stream?

There are two ways to change the throughput of your data stream. You can use the UpdateShardCount API or the AWS Management Console to scale the number of shards in a data stream, or you can change the throughput of an Amazon Kinesis data stream by adjusting the number of shards within the data stream (resharding).

Q: How long does it take to change the throughput of my Amazon Kinesis data stream using UpdateShardCount or the AWS Management Console?

Typical scaling requests should take a few minutes to complete. Larger scaling requests will take longer than smaller ones.

Q: What are the limitations of UpdateShardCount?

For information about limitations of UpdateShardCount, see the Amazon Kinesis Data Streams Service API Reference.

Q: Does Amazon Kinesis Data Streams remain available when I change the throughput of my Amazon Kinesis data stream using UpdateShardCount or via resharding?

Yes. You can continue adding data to and reading data from your Amazon Kinesis data stream while you use UpdateShardCount or reshard to change the throughput of the data stream.

Q: What is resharding?

Resharding is the process used to scale your data stream using a series of shard splits or merges. In a shard split, a single shard is divided into two shards, which increases the throughput of the data stream. In a shard merge, two shards are merged into a single shard, which decreases the throughput of the data stream. For more information, see Resharding a Data Stream in the Amazon Kinesis Data Streams developer guide.

Q: How often can I and how long does it take to change the throughput of my Amazon Kinesis data stream by resharding it?

A resharding operation such as shard split or shard merge takes a few seconds. You can only perform one resharding operation at a time. Therefore, for an Amazon Kinesis data stream with only one shard, it takes a few seconds to double the throughput by splitting one shard. For a data stream with 1000 shards, it takes 30K seconds (8.3 hours) to double the throughput by splitting 1000 shards. We recommend increasing the throughput of your data stream ahead of the time when extra throughput is needed.

Q: How do I change the data retention period of my Amazon Kinesis data stream?

Amazon Kinesis stores your data for up to 24 hours by default. You can raise data retention period to up to 7 days by enabling extended data retention.

For more information about changing data retention period, see Changing Data Retention Period.

Q: How do I monitor the operations and performance of my Amazon Kinesis data stream?

Amazon Kinesis Data Streams Management Console displays key operational and performance metrics such as throughput of data input and output of your Amazon Kinesis data streams. Amazon Kinesis Data Streams also integrates with Amazon CloudWatch so that you can collect, view, and analyze CloudWatch metrics for your data streams and shards within those data streams. For more information about Amazon Kinesis Data Streams metrics, see Monitoring Amazon Kinesis Data Streams with Amazon CloudWatch.

Please note that all stream-level metrics are free of charge. All enabled shard-level metrics are charged at Amazon CloudWatch Pricing.

Q: How do I manage and control access to my Amazon Kinesis data stream?

Amazon Kinesis Data Streams integrates with AWS Identity and Access Management (IAM), a service that enables you to securely control access to your AWS services and resources for your users. For example, you can create a policy that only allows a specific user or group to add data to your Amazon Kinesis data stream. For more information about access management and control of your data stream, see Controlling Access to Amazon Kinesis Data Streams Resources using IAM.

Q: How do I log API calls made to my Amazon Kinesis data stream for security analysis and operational troubleshooting?

Amazon Kinesis integrates with Amazon CloudTrail, a service that records AWS API calls for your account and delivers log files to you. For more information about API call logging and a list of supported Amazon Kinesis API operations, see Logging Amazon Kinesis API calls Using Amazon CloudTrail.

Q: How do I effectively manage my Amazon Kinesis data streams and the costs associated with these data streams?

Amazon Kinesis Data Streams allows you to tag your Amazon Kinesis data streams for easier resource and cost management. A tag is a user-defined label expressed as a key-value pair that helps organize AWS resources. For example, you can tag your data streams by cost centers so that you can categorize and track your Amazon Kinesis Data Streams costs based on cost centers. For more information about Amazon Kinesis Data Streams tagging, see Tagging Your Amazon Kinesis Data Streams.

Q: How can I describe how I'm utilizing my shard limit?

You can understand how you're utilizing your shard limit for an account using the DescribeLimits API. The DescribeLimits API will return the shard limit and the number of open shards in your account. If you need to raise your shard limit, please request a limit increase.

Security

Q: When I use Kinesis Data Streams, how secure is my data?

Kinesis is secure by default. Only the account and data stream owners have access to the Kinesis resources they create. Kinesis supports user authentication to control access to data. You can use AWS IAM policies to selectively grant permissions to users and groups of users. You can securely put and get your data from Kinesis through SSL endpoints using the HTTPS protocol. If you need extra security you can use server-side encryption with AWS KMS master keys to encrypt data stored in your data stream. AWS KMS allows you to use AWS generated KMS master keys for encryption, or if you prefer you can bring your own master key into AWS KMS. Lastly, you can use your own encryption libraries to encrypt data on the client-side before putting the data into Kinesis.

Q: Can I privately access Kinesis Data Streams APIs from my Amazon Virtual Private Cloud (VPC) without using public IPs?

Yes, you can privately access Kinesis Data Streams APIs from your Amazon Virtual Private Cloud (VPC) by creating VPC Endpoints. With VPC Endpoints, the routing between the VPC and Kinesis Data Streams is handled by the AWS network without the need for an Internet gateway, NAT gateway, or VPN connection. The latest generation of VPC Endpoints used by Kinesis Data Streams are powered by AWS PrivateLink, a technology that enables private connectivity between AWS services using Elastic Network Interfaces (ENI) with private IPs in your VPCs. To learn more about PrivateLink, visit the PrivateLink documentation.

Encryption

Q: Can I encrypt the data I put into a Kinesis data stream?

Yes, and there are two options for encrypting the data you put into a Kinesis data stream. You can use server-side encryption , which is a fully managed feature that automatically encrypts and decrypts data as you put and get it from a data stream. Or you can write encrypted data to a data stream by encrypting and decrypting on the client side.

Q: Why should I use server-side encryption instead of client-side encryption?

Customers often choose server-side encryption over client-side encryption for one of the following reasons:

- It hard to enforce client-side encryption
- They want a second layer of security on top of client-side encryption
- It is hard to implement client-side key management schemes

Q: What is server-side encryption?

Server-side encryption for Kinesis Data Streams automatically encrypts data using a user specified AWS KMS master key (CMK) before it is written to the data stream storage layer, and decrypts the data after it is retrieved from storage. Encryption makes writes impossible and the payload and the partition key unreadable unless the user writing or reading from the data stream has the permission to use the key selected for encryption on the data stream. As a result, server-side encryption can make it easier to meet internal security and compliance requirements governing your data.

With server-side encryption your client-side applications (producers and consumers) do not need to be aware of encryption, they do not need to manage CMKs or cryptographic operations, and your data is encrypted when it is at rest and in motion within the Kinesis Data Streams service.

All CMKs used by the server-side encryption feature are provided by the AWS KMS. AWS KMS makes it easy to use an AWS-managed CMK for Kinesis(a “one-click” encryption method), your own AWS KMS generated CMK, or a CMK that you imported for encryption.

Q: Is there a server-side encryption getting started guide?

Yes, there is a getting started guide in the user documentation.

Q: Does server-side encryption interfere with how my applications interact with Kinesis Data Streams?

Possibly. This depends on the key you use for encryption and the permissions governing access to the key.

- If you use the AWS-managed CMK for Kinesis (key alias = aws/kinesis) your applications will not be impacted by enabling or disabling encryption with this key.
- If you use a different master key, like a custom AWS KMS master key or one you imported into the AWS KMS service, and if your producers and consumers of a data stream do not have permission to use the AWS KMS CMK used for encryption, then your PUT and GET requests will fail. Before you can use server-side encryption you must configure AWS KMS key policies to allow encryption and decryption of messages. For examples and more information about AWS KMS permissions, see AWS KMS API Permissions: Actions and Resources Reference in the AWS Key Management Service Developer Guide or the permissions guidelines in the Kinesis Data Streams server-side encryption user documentation.

Q: Is there an additional cost associated with the use of server-side encryption?

Yes, however if you are using the AWS-managed CMK for Kinesis and are not exceeding the free tier KMS API usage costs, then your use of server-side encryption is free. The following describes the costs by resource:

Keys:

- The AWS-managed CMK for Kinesis (alias = “aws/kinesis”) is free.
- User generated KMS keys are subject to KMS key costs. Learn more.

KMS API Usage:

- API usage costs apply for every CMK, including custom ones. Kinesis Data Streams calls KMS approximately every 5 minutes when it is rotating the data key. In a 30-day month, the total cost of KMS API calls initiated by a Kinesis data stream should be less than a few dollars. Please note that this cost scales with the number of user credentials you use on your data producers and consumers because each user credential requires a unique API call to AWS KMS. When you use IAM role for authentication, each assume-role-call will result in unique user credentials and you might want to cache user credentials returned by the assume-role-call to save KMS costs.

Q: Which AWS regions offer server-side encryption for Kinesis Data Streams?

Kinesis Data Streams server-side encryption is available in the AWS GovCloud region and all public regions except the China (Beijing) region.

Q: How do I start, update, or remove server-side encryption from a data stream?

All of these operations can be completed using the AWS management console or using the AWS SDK. To learn more, see the Kinesis Data Streams server-side encryption getting started guide.

Q: What encryption algorithm is used for server-side encryption?

Kinesis Data Streams uses an AES-GCM 256 algorithm for encryption.

Q: If I encrypt a data stream that already has data written to it, either in plain text or ciphertext, will all of the data in the data stream be encrypted or decrypted if I update encryption?

No, only new data written into the data stream will be encrypted (or left decrypted) by the new application of encryption.

Q: What does server-side encryption for Kinesis Data Streams encrypt?

Server-side encryption encrypts the payload of the message along with the partition key, which is specified by the data stream producer applications.

Q: Is server-side encryption a shard specific feature or a stream specific feature?

Server-side encryption is a stream specific feature.

Q: Can I change the CMK that is used to encrypt a specific data stream?

Yes, using the AWS management console or the AWS SDK you can choose a new master key to apply to a specific data stream.

Q: Can you walk me through the encryption lifecycle of my data from the point in time when I send it to a Kinesis data stream with server-side encryption enabled, and when I retrieve it?

The following walks you through how Kinesis Data Streams uses AWS KMS CMKs to encrypt a message before it is stored in the PUT path, and to decrypt it after it is retrieved in the GET path. Kinesis and AWS KMS perform the following actions (including decryption) when you call putRecord(s) or getRecords on a data stream with server-side encryption enabled.

1. Data is sent from a customer's Kinesis producer application (client) to Kinesis using SSL via HTTPS.
2. Data is received by Kinesis, stored in RAM, and encryption is applied to the payload and partition key of a record.
3. Kinesis requests a plaintext input keying material (IKM) and a copy of the IKM is encrypted by using the customer's selected KMS master key.

4. AWS KMS creates an IKM, encrypts it by using the master key, and sends both the plaintext IKM and the encrypted IKM to Kinesis.
5. Kinesis uses the plaintext IKM to derive data keys that are unique per-record.
6. Kinesis encrypts the payload and partition key using the data keys and removes the plaintext key from memory.
7. Kinesis appends the encrypted IKM to the encrypted data.
8. The plaintext IKM is cached in memory for reuse until it expires after 5 minutes.
9. Kinesis delivers the encrypted message to a backend store where it is stored at rest and fetch-able by a getRecords call.

Kinesis and AWS KMS perform the following actions (including decryption) when you call getRecords.

1. When a getRecords call is made, the frontend of Kinesis retrieves the encrypted record from the backend service.
2. Kinesis makes a request to KMS using a token generated by the customer's request. AWS KMS authorizes it.
3. Kinesis decrypts the encrypted IKM stored with the record.
4. Kinesis recreates the per-record data keys from the decrypted IKM.
5. If authorized, Kinesis decrypts the payload and removes the plaintext data key from memory.
6. Kinesis delivers the payload over SSL and HTTPS to the Kinesis consumer (client) requesting the records.

Pricing and billing

Q: Is Amazon Kinesis Data Streams available in AWS Free Tier?

No. Amazon Kinesis Data Streams is not currently available in AWS Free Tier. AWS Free Tier is a program that offers free trial for a group of AWS services. For more details about AWS Free Tier, see AWS Free Tier.

Q: How much does Amazon Kinesis Data Streams cost?

Amazon Kinesis Data Streams uses simple pay as you go pricing. There is neither upfront cost nor minimum fees, and you only pay for the resources you use. The cost of Amazon Kinesis Data Streams has two core dimensions and three optional dimensions:

- Hourly Shard cost determined by the number of shards within your Amazon Kinesis data stream.
- PUT Payload Unit cost determined by the number of 25KB payload units that your data producers add to your data stream.

Optional:

- Extended data retention is an optional cost determined by the number of shard hours incurred by your data stream. When extended data retention is enabled, you pay the extended retention rate for each shard in your stream.
- Enhanced fan-out is an optional cost with two cost dimensions: consumer-shard hours and data retrievals. Consumer-shard hours reflect the number of shards in a stream multiplied by the number of consumers using enhanced fan-out. Data retrievals are determined by the number of GBs delivered to consumers using enhanced fan-out.

For more information about Amazon Kinesis Data Streams costs, see Amazon Kinesis Data Streams Pricing.

Q: Does my PUT Payload Unit cost change by using PutRecords operation instead of PutRecord operation?

PUT Payload Unit charge is calculated based on the number of 25KB payload units added to your Amazon Kinesis data stream. PUT Payload Unit cost is consistent when using PutRecords operation or PutRecord operation.

Q: Am I charged for shards in "CLOSED" state?

A shard could be in "CLOSED" state after resharding. You will not be charged for shards in "CLOSED" state.

Q: Other than Amazon Kinesis Data Streams costs, are there any other costs that might incur to my Amazon Kinesis Data Streams usage?

If you use Amazon EC2 for running your Amazon Kinesis Applications, you will be charged for Amazon EC2 resources in addition to Amazon Kinesis Data Streams costs.

Amazon Kinesis Client Library (KCL) uses Amazon DynamoDB tables to track state information of record processing. If you use KCL for your Amazon Kinesis Applications, you will be charged for Amazon DynamoDB resources in addition to Amazon Kinesis Data Streams costs.

If you enable Enhanced Shard-Level Metrics, you will be charged for Amazon CloudWatch costs associated with enabled shard-level metrics in addition to Amazon Kinesis Data Streams costs.

Please note that the above are three common, but not exhaustive, cases.