Amazon Kinesis

Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information.

Collection of services for processing streams of various data.

Data is processed in “shards” – with each shard able to ingest 1000 records per second.

There is a default limit of 500 shards, but you can request an increase to unlimited shards.

A record consists of a partition key, sequence number, and data blob (up to 1 MB).

Transient data store – default retention of 24 hours, but can be configured for up to 7 days.

There are four types of Kinesis service and these are detailed below.

Kinesis Video Streams

Kinesis Video Streams makes it easy to securely stream video from connected devices to AWS for analytics, machine learning (ML), and other processing.

Durably stores, encrypts, and indexes video data streams, and allows access to data through easy-to-use APIs.

Producers provide data streams.

Stores data for 24 hours by default, up to 7 days.

Stores data in shards – 5 transaction per second for reads, up to a max read rate of 2MB per second and 1000 records per second for writes up to a max of 1MB per second.

Consumers receive and process data.

Can have multiple shards in a stream.

Supports encryption at rest with server-side encryption (KMS) with a customer master key.

Kinesis Data Streams

Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs.

Kinesis Data Streams enables real-time processing of streaming big data.

Kinesis Data Streams is useful for rapidly moving data off data producers and then continuously processing the data.

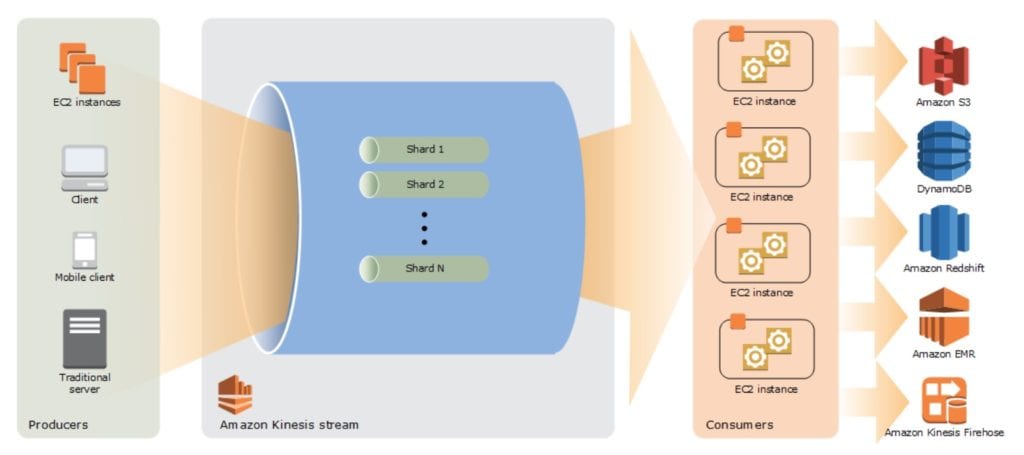
Kinesis Data Streams **stores data** for later processing by applications (key difference with Firehose which delivers data directly to AWS services).

Common use cases include:

* Accelerated log and data feed intake.
* Real-time metrics and reporting.
* Real-time data analytics.
* Complex stream processing.

The following diagram illustrates the high-level architecture of Kinesis Data Streams.

* Producers continually push data to Kinesis Data Streams.
* Consumers process the data in real time.
* Consumers can store their results using an AWS service such as Amazon DynamoDB, Amazon Redshift, or Amazon S3.
* Kinesis Streams applications are consumers that run on EC2 instances.
* Shards are uniquely identified groups or data records in a stream.
* Records are the data units stored in a Kinesis Stream.



A producer creates the data that makes up the stream.

Producers can be used through the following:

* Kinesis Streams API.
* Kinesis Producer Library (KPL).
* Kinesis Agent.

A record is the unit of data stored in a Amazon Kinesis data stream.

A record is composed of a sequence number, partition key, and data blob.

By default, records of a stream are accessible for up to 24 hours from the time they are added to the stream (can be raised to 7 days  by enabling extended data retention).

A 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) within one record is 1 megabyte (MB).

A 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.

Each shard can support up to 1000 PUT records per second.

A stream is composed of one or more shards.

Consumers are the EC2 instances that analyze the data received from a stream.

Consumers are known as Amazon Kinesis Streams Applications.

When the data rate increases, add more shards to increase the size of the stream.

Remove shards when the data rate decreases.

Partition keys are used to group data by shard within a stream.

Kinesis Streams uses KMS master keys for encryption.

To read from or write to an encrypted stream the producer and consumer applications must have permission to access the master key.

Kinesis Data Streams replicates synchronously across three AZs.

Check out this AWS Hands-On Labs video showing how to create an Amazon Kinesis stream and integrate it with an AWS Lambda function to consume events using an event-source mapping:

Kinesis Data Firehose

Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools.

Captures, transforms, and loads streaming data.

Enables near real-time analytics with existing business intelligence tools and dashboards.

Kinesis Data Streams can be used as the source(s) to Kinesis Data Firehose.

You can configure Kinesis Data Firehose to transform your data before delivering it.

With Kinesis Data Firehose you don’t need to write an application or manage resources.

Firehose can batch, compress, and encrypt data before loading it.

Firehose synchronously replicates data across three AZs as it is transported to destinations.

Each delivery stream stores data records for up to 24 hours.

A source is where your streaming data is continuously generated and captured.

A delivery stream is the underlying entity of Amazon Kinesis Data Firehose.

A record is the data of interest your data producer sends to a delivery stream.

The maximum size of a record (before Base64-encoding) is 1000 KB.

A destination is the data store where your data will be delivered.

Firehose Destinations include:

* Amazon S3.
* Amazon Redshift.
* Amazon Elasticsearch Service.
* Splunk.

Producers provide data streams.

No shards, totally automated.

Can encrypt data with an existing AWS Key Management Service (KMS) key.

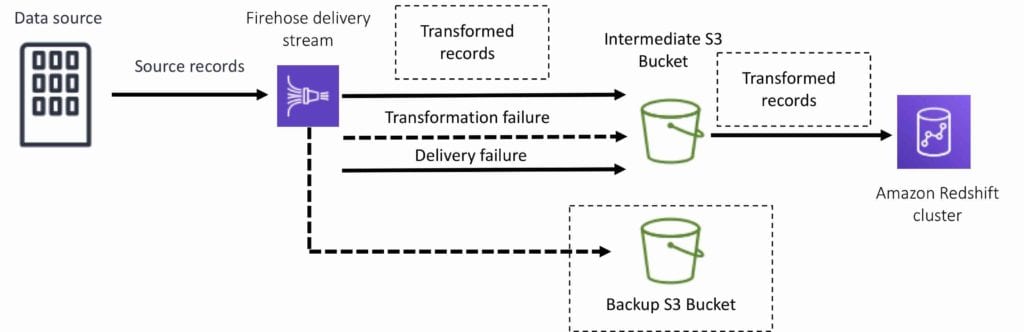
Server-side-encryption can be used if Kinesis Streams is used as the data source.

Firehose can invoke an AWS Lambda function to transform incoming data before delivering it to a destination.

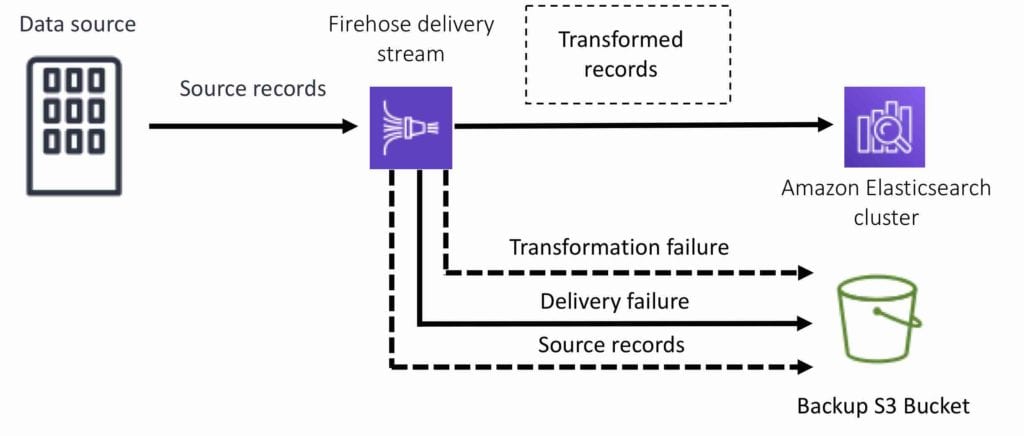
For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket:



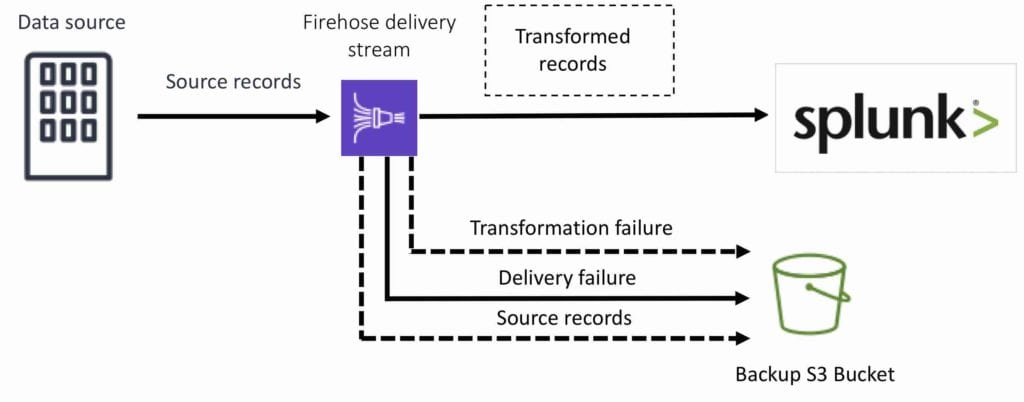
For Amazon Redshift destinations, streaming data is delivered to your S3 bucket first. Kinesis Data Firehose then issues an Amazon Redshift **COPY** command to load data from your S3 bucket to your Amazon Redshift cluster. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket:



For Amazon Elaticsearch destinations, streaming data is delivered to your Amazon ES cluster, and it can optionally be backed up to your S3 bucket concurrently:



For Splunk destinations, streaming data is delivered to Splunk, and it can optionally be backed up to your S3 bucket concurrently:



Kinesis Data Analytics

Amazon Kinesis Data Analytics is the easiest way to process and analyze real-time, streaming data.

Can use standard SQL queries to process Kinesis data streams.

Provides real-time analysis.

Use cases:

* Generate time-series analytics.
* Feed real-time dashboards.
* Create real-time alerts and notifications.

Quickly author and run powerful SQL code against streaming sources.

Can ingest data from Kinesis Streams and Kinesis Firehose.

Output to S3, RedShift, Elasticsearch and Kinesis Data Streams.

Sits over Kinesis Data Streams and Kinesis Data Firehose.

A Kinesis Data Analytics application consists of three components:

* Input – the streaming source for your application.
* Application code – a series of SQL statements that process input and produce output.
* Output – one or more in-application streams to hold intermediate results.

Kinesis Data Analytics supports two types of inputs: streaming data sources and reference data sources:

* A streaming data source is continuously generated data that is read into your application for processing.
* A reference data source is static data that your application uses to enrich data coming in from streaming sources.

Can configure destinations to persist the results.

Supports Kinesis Streams and Kinesis Firehose (S3, RedShift, ElasticSearch) as destinations.

IAM can be used to provide Kinesis Analytics with permissions to read records from sources and write to destinations.

# Amazon Athena

## General

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.

Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

Athena is easy to use – simply point to your data in Amazon S3, define the schema, and start querying using standard SQL.

Amazon Athena uses Presto with full standard SQL support and works with a variety of standard data formats, including CSV, JSON, ORC, Apache Parquet and Avro.

While Amazon Athena is ideal for quick, ad-hoc querying and integrates with Amazon QuickSight for easy visualization, it can also handle complex analysis, including large joins, window functions, and arrays.

Amazon Athena uses a managed Data Catalog to store information and schemas about the databases and tables that you create for your data stored in Amazon S3.

With Amazon Athena, you don’t have to worry about managing or tuning clusters to get fast performance.

Athena is optimized for fast performance with Amazon S3.

Athena automatically executes queries in parallel, so that you get query results in seconds, even on large datasets.

Most results are delivered within seconds.

With Athena, there’s no need for complex ETL jobs to prepare data for analysis.

This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

Athena is out-of-the-box integrated with [**AWS Glue**](https://aws.amazon.com/glue/) Data Catalog, allowing you to create a unified metadata repository across various services, crawl data sources to discover schemas and populate your Catalog with new and modified table and partition definitions, and maintain schema versioning.

You can also use Glue’s fully-managed ETL capabilities to transform data or convert it into columnar formats to optimize cost and improve performance.

## Use Cases

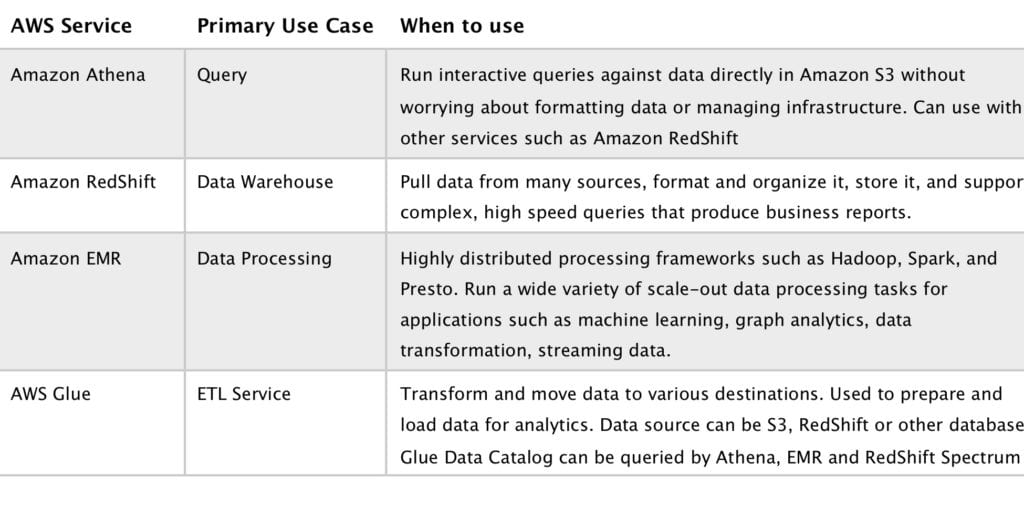
Query services like Amazon Athena, data warehouses like Amazon Redshift, and sophisticated data processing frameworks like Amazon EMR, all address different needs and use cases.

Amazon Redshift provides the fastest query performance for enterprise reporting and business intelligence workloads, particularly those involving extremely complex SQL with multiple joins and sub-queries.

Amazon EMR makes it simple and cost effective to run highly distributed processing frameworks such as Hadoop, Spark, and Presto when compared to on-premises deployments. Amazon EMR is flexible – you can run custom applications and code, and define specific compute, memory, storage, and application parameters to optimize your analytic requirements.

Amazon Athena provides the easiest way to run ad-hoc queries for data in S3 without the need to setup or manage any servers.

The table below shows the primary use case and situations for using a few AWS query and analytics services:



## Pricing

With Amazon Athena, you pay only for the queries that you run.

You are charged based on the amount of data scanned by each query.

You can get significant cost savings and performance gains by compressing, partitioning, or converting your data to a columnar format, because each of those operations reduces the amount of data that Athena needs to scan to execute a query.

# Amazon EMR

Amazon EMR is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data.

EMR utilizes a hosted Hadoop framework running on Amazon EC2 and Amazon S3.

Managed Hadoop framework for processing huge amounts of data.

Also support Apache Spark, HBase, Presto and Flink.

Most commonly used for log analysis, financial analysis, or extract, translate and loading (ETL) activities.

A Step is a programmatic task for performing some process on the data (e.g. count words).

A cluster is a collection of EC2 instances provisioned by EMR to run your Steps.

EMR uses Apache Hadoop as its distributed data processing engine which is an open source, Java software framework that supports data-intensive distributed applications running on large clusters of commodity hardware.

EMR is a good place to deploy Apache Spark, an open-source distributed processing used for big data workloads which utilizes in-memory caching and optimized query execution.

You can also launch Presto clusters. Presto is an open-source distributed SQL query engine designed for fast analytic queries against large datasets.

EMR launches all nodes for a given cluster in the same Amazon EC2 Availability Zone.

You can access Amazon EMR by using the AWS Management Console, Command Line Tools, SDKS, or the EMR API.

With EMR you have access to the underlying operating system (you can SSH in).

# AWS Glue

AWS Glue is a fully-managed, pay-as-you-go, extract, transform, and load (ETL) service that automates the time-consuming steps of data preparation for analytics.

AWS Glue automatically discovers and profiles data via the Glue Data Catalog, recommends and generates ETL code to transform your source data into target schemas.

AWS Glue runs the ETL jobs on a fully managed, scale-out Apache Spark environment to load your data into its destination.

AWS Glue also allows you to setup, orchestrate, and monitor complex data flows.

You can create and run an ETL job with a few clicks in the AWS Management Console.

Simply point AWS Glue to your data stored on AWS, and AWS Glue discovers data and stores the associated metadata (e.g. table definition and schema) in the AWS Glue Data Catalog.

Once cataloged, data is immediately searchable, queryable, and available for ETL.

AWS Glue consists of a Data Catalog which is a central metadata repository, an ETL engine that can automatically generate Scala or Python code, and a flexible scheduler that handles dependency resolution, job monitoring, and retries.

Together, these automate much of the undifferentiated heavy lifting involved with discovering, categorizing, cleaning, enriching, and moving data, so you can spend more time analyzing your data.

AWS Glue crawlers connect to a source or target data store, progress through a prioritized list of classifiers to determine the schema for the data, and then creates metadata in the AWS Glue Data Catalog.

The metadata is stored in tables in a data catalog and used in the authoring process of ETL jobs.

You can run crawlers on a schedule, on-demand, or trigger them based on an event to ensure that your metadata is up-to-date.

AWS Glue automatically generates the code to extract, transform, and load data.

Simply point AWS Glue to a source and target, and AWS Glue creates ETL scripts to transform, flatten, and enrich the data.

The code is generated in Scala or Python and written for Apache Spark.

AWS Glue helps clean and prepare data for analysis by providing a Machine Learning Transform called FindMatches for deduplication and finding matching records.

## Use Cases

Use AWS Glue to discover properties of data, transform it, and prepare it for analytics.

Glue can automatically discover both structured and semi-structured data stored in data lakes on [**Amazon S3**](https://aws.amazon.com/s3/), data warehouses in [**Amazon Redshift**](https://aws.amazon.com/redshift/), and various databases running on AWS.

It provides a unified view of data via the Glue Data Catalog that is available for ETL, querying and reporting using services like [**Amazon Athena**](https://aws.amazon.com/athena/), [**Amazon EMR**](https://aws.amazon.com/emr/), and [**Amazon Redshift Spectrum**](https://aws.amazon.com/redshift/).

Glue automatically generates Scala or Python code for ETL jobs that you can further customize using tools you are already familiar with.

AWS Glue is serverless, so there are no compute resources to configure and manage.