**Some Kinesis Data Streams Records are Skipped When Using the Kinesis Client Library**

The most common cause of skipped records is an unhandled exception thrown from processRecords. The Kinesis Client Library (KCL) relies on your processRecords code to handle any exceptions that arise from processing the data records. Any exception thrown from processRecords is absorbed by the KCL. To avoid infinite retries on a recurring failure, the KCL does not resend the batch of records processed at the time of the exception. The KCL then calls processRecords for the next batch of data records without restarting the record processor. This effectively results in consumer applications observing skipped records. To prevent skipped records, handle all exceptions within processRecords appropriately.

**Records Belonging to the Same Shard are Processed by Different Record Processors at the Same Time**

For any running Kinesis Client Library (KCL) application, a shard only has one owner. However, multiple record processors may temporarily process the same shard. In the case of a worker instance that loses network connectivity, the KCL assumes that the unreachable worker is no longer processing records, after the failover time expires, and directs other worker instances to take over. For a brief period, new record processors and record processors from the unreachable worker may process data from the same shard.

You should set a failover time that is appropriate for your application. For low-latency applications, the 10-second default may represent the maximum time you want to wait. However, in cases where you expect connectivity issues such as making calls across geographical areas where connectivity could be lost more frequently, this number may be too low.

Your application should anticipate and handle this scenario, especially because network connectivity is usually restored to the previously unreachable worker. If a record processor has its shards taken by another record processor, it must handle the following two cases to perform graceful shutdown:

1. After the current call to processRecords is completed, the KCL invokes the shutdown method on the record processor with shutdown reason 'ZOMBIE'. Your record processors are expected to clean up any resources as appropriate and then exit.
2. When you attempt to checkpoint from a 'zombie' worker, the KCL throws ShutdownException. After receiving this exception, your code is expected to exit the current method cleanly.

For more information, see [Handling Duplicate Records](https://docs.aws.amazon.com/streams/latest/dev/kinesis-record-processor-duplicates.html).

**Consumer Application is Reading at a Slower Rate Than Expected**

The most common reasons for read throughput being slower than expected are as follows:

1. Multiple consumer applications have total reads exceeding the per-shard limits. For more information, see [Quotas and Limits](https://docs.aws.amazon.com/streams/latest/dev/service-sizes-and-limits.html). In this case, increase the number of shards in the Kinesis data stream.
2. The [limit](https://docs.aws.amazon.com/kinesis/latest/APIReference/API_GetRecords.html#API_GetRecords_RequestSyntax) that specifies the maximum number of **GetRecords** per call may have been configured with a low value. If you are using the KCL, you may have configured the worker with a low value for the maxRecords property. In general, we recommend using the system defaults for this property.
3. The logic inside your processRecords call may be taking longer than expected for a number of possible reasons; the logic may be CPU intensive, I/O blocking, or bottlenecked on synchronization. To test if this is true, test run empty record processors and compare the read throughput. For information about how to keep up with the incoming data, see [Resharding, Scaling, and Parallel Processing](https://docs.aws.amazon.com/streams/latest/dev/kinesis-record-processor-scaling.html).

If you have only one consumer application, it is always possible to read at least two times faster than the put rate. That’s because you can write up to 1,000 records per second for writes, up to a maximum total data write rate of 1 MB per second (including partition keys). Each open shard can support up to 5 transactions per second for reads, up to a maximum total data read rate of 2 MB per second. Note that each read (**GetRecords** call) gets a batch of records. The size of the data returned by **GetRecords** varies depending on the utilization of the shard. The maximum size of data that **GetRecords** can return is 10 MB. If a call returns that limit, subsequent calls made within the next 5 seconds throw ProvisionedThroughputExceededException.

**GetRecords Returns Empty Records Array Even When There is Data in the Stream**

Consuming, or getting records is a pull model. Developers are expected to call [GetRecords](https://docs.aws.amazon.com/kinesis/latest/APIReference/API_GetRecords.html) in a continuous loop with no back-offs. Every call to **GetRecords** also returns a ShardIterator value, which must be used in the next iteration of the loop.

The **GetRecords** operation does not block. Instead, it returns immediately; with either relevant data records or with an empty Records element. An empty Records element is returned under two conditions:

1. There is no more data currently in the shard.
2. There is no data near the part of the shard pointed to by the ShardIterator.

The latter condition is subtle, but is a necessary design tradeoff to avoid unbounded seek time (latency) when retrieving records. Thus, the stream-consuming application should loop and call **GetRecords**, handling empty records as a matter of course.

In a production scenario, the only time the continuous loop should be exited is when the NextShardIterator value is NULL. When NextShardIterator is NULL, it means that the current shard has been closed and the ShardIteratorvalue would otherwise point past the last record. If the consuming application never calls **SplitShard** or **MergeShards**, the shard remains open and the calls to **GetRecords** never return a NextShardIterator value that is NULL.

If you use the Kinesis Client Library (KCL), the above consumption pattern is abstracted for you. This includes automatic handling of a set of shards that dynamically change. With the KCL, the developer only supplies the logic to process incoming records. This is possible because the library makes continuous calls to **GetRecords** for you.

**Shard Iterator Expires Unexpectedly**

A new shard iterator is returned by every **GetRecords** request (as NextShardIterator), which you then use in the next **GetRecords** request (as ShardIterator). Typically, this shard iterator does not expire before you use it. However, you may find that shard iterators expire because you have not called **GetRecords** for more than 5 minutes, or because you've performed a restart of your consumer application.

If the shard iterator expires immediately, before you can use it, this might indicate that the DynamoDB table used by Kinesis does not have enough capacity to store the lease data. This situation is more likely to happen if you have a large number of shards. To solve this problem, increase the write capacity assigned to the shard table. For more information, see [Using a Lease Table to Track the Shards Processed by the KCL Consumer Application](https://docs.aws.amazon.com/streams/latest/dev/shared-throughput-kcl-consumers.html#shared-throughput-kcl-consumers-leasetable).

**Consumer Record Processing Falling Behind**

For most use cases, consumer applications are reading the latest data from the stream. In certain circumstances, consumer reads may fall behind, which may not be desired. After you identify how far behind your consumers are reading, look at the most common reasons why consumers fall behind.

Start with the GetRecords.IteratorAgeMilliseconds metric, which tracks the read position across all shards and consumers in the stream. Note that if an iterator's age passes 50% of the retention period (by default, 24 hours, configurable up to 365 days), there is risk for data loss due to record expiration. A quick stopgap solution is to increase the retention period. This stops the loss of important data while you troubleshoot the issue further. For more information, see [Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch](https://docs.aws.amazon.com/streams/latest/dev/monitoring-with-cloudwatch.html). Next, identify how far behind your consumer application is reading from each shard using a custom CloudWatch metric emitted by the Kinesis Client Library (KCL), MillisBehindLatest. For more information, see [Monitoring the Kinesis Client Library with Amazon CloudWatch](https://docs.aws.amazon.com/streams/latest/dev/monitoring-with-kcl.html).

Here are the most common reasons consumers can fall behind:

* Sudden large increases to GetRecords.IteratorAgeMilliseconds or MillisBehindLatest usually indicate a transient problem, such as API operation failures to a downstream application. You should investigate these sudden increases if either of the metrics consistently display this behavior.
* A gradual increase to these metrics indicates that a consumer is not keeping up with the stream because it is not processing records fast enough. The most common root causes for this behavior are insufficient physical resources or record processing logic that has not scaled with an increase in stream throughput. You can verify this behavior by looking at the other custom CloudWatch metrics that the KCL emits associated with the processTask operation, including RecordProcessor.processRecords.Time, Success, and RecordsProcessed.
  + If you see an increase in the processRecords.Time metric that correlates with increased throughput, you should analyze your record processing logic to identify why it is not scaling with the increased throughput.
  + If you see an increase to the processRecords.Time values that are not correlated with increased throughput, check to see if you are making any blocking calls in the critical path, which are often the cause of slowdowns in record processing. An alternative approach is to increase your parallelism by increasing the number of shards. Finally, confirm you have an adequate amount of physical resources (memory, CPU utilization, etc.) on the underlying processing nodes during peak demand.

**Unauthorized KMS master key permission error**

This error occurs when a consumer application reads from an encrypted stream without permissions on the KMS master key. To assign permissions to an application to access a KMS key, see [Using Key Policies in AWS KMS](https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html) and [Using IAM Policies with AWS KMS](https://docs.aws.amazon.com/kms/latest/developerguide/iam-policies.html).

# **Low-Latency Processing**

Propagation delay is defined as the end-to-end latency from the moment a record is written to the stream until it is read by a consumer application. This delay varies depending upon a number of factors, but it is primarily affected by the polling interval of consumer applications.

For most applications, we recommend polling each shard one time per second per application. This enables you to have multiple consumer applications processing a stream concurrently without hitting Amazon Kinesis Data Streams limits of 5 GetRecords calls per second. Additionally, processing larger batches of data tends to be more efficient at reducing network and other downstream latencies in your application.

The KCL defaults are set to follow the best practice of polling every 1 second. This default results in average propagation delays that are typically below 1 second.

Kinesis Data Streams records are available to be read immediately after they are written. There are some use cases that need to take advantage of this and require consuming data from the stream as soon as it is available. You can significantly reduce the propagation delay by overriding the KCL default settings to poll more frequently, as shown in the following examples.

Java KCL configuration code:

kinesisClientLibConfiguration = new

KinesisClientLibConfiguration(applicationName,

streamName,

credentialsProvider,

workerId).withInitialPositionInStream(initialPositionInStream).withIdleTimeBetweenReadsInMillis(250);

[Amazon EMR](https://aws.amazon.com/emr/) is a managed Hadoop framework that you use to process vast amounts of data. One of the reasons that customers choose Amazon EMR is its security features. For example, customers like [FINRA](https://aws.amazon.com/solutions/case-studies/finra-data-validation/) in regulated industries such as financial services, and in healthcare, choose Amazon EMR as part of their data strategy. They do so to adhere to strict regulatory requirements from entities such as the Payment Card Industry Data Security Standard (PCI) and the Health Insurance Portability and Accountability Act (HIPAA).

This post walks you through some of the principles of Amazon EMR security. It also describes features that you can use in Amazon EMR to help you meet the security and compliance objectives for your business. We cover some common security best practices that we see used. We also show some sample configurations to get you started. For more in-depth information, see [Security](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-security.html) in the EMR Management Guide.

## Encryption, encryption, encryption

Our CTO, Werner Vogels, is fond of saying “Dance like nobody’s watching, encrypt like everybody is.” Properly protecting your data at rest and in transit using encryption is a core component of our [well-architected pillar of security](https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf). Amazon EMR [security configurations](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-create-security-configuration.html) (described in the EMR documentation) make it easy for you to encrypt data. A security configuration is like a template for encryption and other security configurations that you can apply to any cluster when you launch it.

### Encryption at rest

You have multiple options to [encrypt data at rest](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-data-encryption-options.html#emr-encryption-s3) in your EMR clusters. EMR by default uses the EMR file system (EMRFS) to read from and write data to [Amazon S3](https://aws.amazon.com/s3/). To encrypt data in Amazon S3, you can specify one of the following options:

* **SSE-S3:**Amazon S3 manages the encryption keys for you
* **SSE-KMS:** You use an [AWS Key Management Service](https://aws.amazon.com/kms/) (AWS KMS) customer master key (CMK) to encrypt your data server-side on Amazon S3. Be sure to use policies that allow access by Amazon EMR.
* **CSE-KMS/CSE-C:** Amazon S3 encryption and decryption takes place client-side on your Amazon EMR cluster. You can use keys provided by AWS KMS (CSE-KMS) or use a custom Java class that provides the master key (CSE-C).

***#Glacier Select allows you to perform filtering directly against Glacier objects using standard SQL statements. It is the simplest, quickest, and most cost-effective option for querying cold data stored in Glacier.***

***#Kibana runs on port 5601 by default, so opening up port 443 won’t help. IAM doesn’t know about your browser, and KMS does not integrate with Kibana*.**

***#KPI, or Key Performance Indicator, is a newer visualization in Quicksight that lets you visualize a comparison between a key value and its target value.***

***#Dblink allows you to offload queries in Redshift to another database entirely. A materialized view could work for this, but it will always copy from the beginning of the table, and not just handle what has changed since the last incremental update. Read through https://amzn.to/2fraaHv if this question confused you.***

***#RANDOM\_CUT\_FOREST is a function in Kinesis Data Analytics intended for anomaly detection. By using serverless services such as Kinesis, Lambda, and SNS we ensure the scalability of this system, and the choice of Kinesis Streams instead of Firehose ensures real-time delivery of the data.***

## What happens if a record fails?

**The KPL PutRecords** operation often sends multiple records to the stream per request. If a single record fails, it is automatically added back to the KPL buffer and retried. The failure of one record does not impact the processing of other records in the request. For more information, check out these resources:

**Kinesis Data Streams** synchronously replicates data across three Availability Zones in an AWS Region. Kinesis Data Streams is not suited for data persistence or for long-term data storage. However, data will be retained for 24 hours and you can extend the retention period by up to 7 days.

## What happens if the consumer fails?

The KCL consumes and processes the data from a Kinesis data stream. The KCL also offers check pointing, which means it stores a cursor in DynamoDB to durably track the records that have been read from a shard in a Kinesis stream. In the event that a KCL worker fails in the middle of reading data from the stream, KCL will be able to use that cursor and pick up from the exact spot where the failed application left off.

Amazon QuickSight is a cloud-powered business analytics service that allows users to build visualizations, perform ad-hoc analysis, and quickly get business insights from their data.

Amazon QuickSight is built with "SPICE" – a Super-fast, Parallel, In-memory Calculation Engine. Built from the ground up for the cloud, SPICE uses a combination of columnar storage, in-memory technologies enabled through the latest hardware innovations and machine code generation to run interactive queries on large datasets and get rapid responses.

Amazon Redshift is a fully managed, scalable data warehouse that enables secure analytics at scale.

For Amazon QuickSight to connect to an Amazon Redshift instance, you must create a new security group for that instance. This security group contains an inbound rule authorizing access from the appropriate CIDR address block for the Amazon QuickSight servers in that AWS Region. You can configure such a security group irrespective of the Redshift cluster having been created in a VPC or not.

By default, an S3 object is owned by the AWS account that uploaded it. This is true even when the bucket is owned by another account. Because the Amazon Redshift data files from the UNLOAD command were put into your bucket by another account, you (the bucket owner) don't have default permission to access those files.

To get access to the data files, an AWS Identity and Access Management (IAM) role with cross-account permissions must run the UNLOAD command again. Follow these steps to set up the Amazon Redshift cluster with cross-account permissions to the bucket:

1. From the account of the S3 bucket, create an IAM role (Bucket Role) with permissions to the bucket.
2. From the account of the Amazon Redshift cluster, create another IAM role (Cluster Role) with permissions to assume the Bucket Role.
3. Update the Bucket Role to grant bucket access and create a trust relationship with the Cluster Role.
4. From the Amazon Redshift cluster, run the UNLOAD command using the Cluster Role and Bucket Role.

This resolution doesn't apply to Amazon Redshift clusters or S3 buckets that use server-side encryption with AWS Key Management Service (AWS KMS).

***Amazon QuickSight*** is a cloud-powered business analytics service that allows users to build visualizations, perform ad-hoc analysis, and quickly get business insights from their data.

Amazon QuickSight is built with "SPICE" – a Super-fast, Parallel, In-memory Calculation Engine. Built from the ground up for the cloud, SPICE uses a combination of columnar storage, in-memory technologies enabled through the latest hardware innovations and machine code generation to run interactive queries on large datasets and get rapid responses.

Amazon QuickSight offers Standard and Enterprise editions. Both editions offer a full set of features for creating and sharing data visualizations. Enterprise edition additionally offers encryption at rest and Microsoft Active Directory integration. In the Enterprise edition, you select a Microsoft Active Directory directory in AWS Directory Service. You use that active directory to identify and manage your Amazon QuickSight users and administrators.

A QuickSight Author is a user who can connect to data sources (within AWS or outside), create visuals and analyze data. Authors can create interactive dashboards using advanced QuickSight capabilities such as parameters and calculated fields, and publish dashboards with other users in the account.

A QuickSight Reader is a user who consumes interactive dashboards. Readers can log in via their organization’s preferred authentication mechanism (QuickSight username/password, SAML portal or AD auth), view shared dashboards, filter data, drill down to details or export data as a CSV file, using a web browser or mobile app. Readers do not have any allocated SPICE capacity.

For the given use-case, the dashboard will query from a total of 200 GB of uncompressed data, so the source dataset must support at least 200 GB of data, which is only possible in the Enterprise edition of QuickSight. You should also note that the data stored in SPICE can be reused multiple times without incurring additional costs, hence this option is the best fit for the given scenario.

via - <https://docs.aws.amazon.com/quicksight/latest/user/data-source-limits.html>

***Amazon Kinesis Data Firehose*** is an extract, transform, and load (ETL) service that reliably captures, transforms, and delivers streaming data to data lakes, data stores, and analytics services. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon OpenSearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards.

***Amazon Kinesis Data Streams*** is a massively scalable, highly durable data ingestion and processing service optimized for streaming data. Amazon Kinesis Data Streams is integrated with a number of AWS services, including Amazon Kinesis Data Firehose for near real-time transformation

***Amazon Kinesis Data Streams*** is a fully managed, serverless streaming data service that makes it easy to elastically ingest and store logs, events, clickstreams, and other forms of streaming data in real-time. Kinesis Data Streams has two capacity modes: on-demand and provisioned, and both come with specific billing options.

In the on-demand mode, pricing is based on the volume of data ingested and retrieved along with a per-hour charge for each data stream in your account.

With provisioned capacity mode, you specify the number of shards necessary for your application based on its write and read request rate. A shard is a unit of capacity that provides 1 MB/second of write and 2 MB/second of read throughout. A record is the data that your data producer adds to your Amazon Kinesis data stream. A PUT Payload Unit is counted in 25 KB payload “chunks” that comprise a record. For example, a 5 KB record contains one PUT Payload Unit, a 45 KB record contains two PUT Payload Units, and a 1 MB record contains 40 PUT Payload Units. PUT Payload Unit is charged a per-million PUT Payload Units rate. In the provisioned mode (applicable for the given scenario), you pay by the shard hour (1MB/second ingress, 2MB/second egress) and PUT Payload Units, per 1,000,000 units.

For the existing use-case, the Kinesis Data Stream has been partitioned into two shards, one for each of the two vehicles. Therefore, data from the 20 sensors of Vehicle A is going to the first shard and the data from the ten sensors of Vehicle B is going to the other shard, causing an imbalance and thereby resulting in a throughput bottleneck for Vehicle A. The correct solution is to partition the shards on the basis of the sensor ID which will result in an even distribution across the two shards. The use-case already states that the overall stream throughput is less than the assigned Kinesis Data Streams throughput, so there is no need to provision extra shards.

***AWS Lambda*** lets you run code without provisioning or managing servers. You pay only for the compute time you consume. Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

You can use an AWS Lambda function to process messages in an Amazon Simple Queue Service (Amazon SQS) queue. Lambda event source mappings support standard queues and first-in, first-out (FIFO) queues. With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously.

***AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume. Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.***

***You can use an AWS Lambda function to process messages in an Amazon Simple Queue Service (Amazon SQS) queue. Lambda event source mappings support standard queues and first-in, first-out (FIFO) queues. With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously***.

***The Redshift vacuum operation*** re-sorts rows and reclaims space in either a specified table or all tables in the current database. Amazon Redshift does not reclaim free space automatically. Such available space is created whenever you delete or update rows on a table. This process is a design choice inherited from PostgreSQL and a routine maintenance process that you need to follow for your tables to maximize the performance of your Redshift cluster. By running a vacuum command on one of your tables, you can reclaim any free space that is the result of delete and update operations. At the same time, the data of the table gets sorted. Thus, you end up with a compact and sorted table, which is useful for the performance of your cluster

**GetRecords.IteratorAgeMilliseconds** - GetRecords.IteratorAgeMilliseconds measures the age in milliseconds of the last record in the stream for all GetRecords requests. A value of zero for this metric indicates that the records are current within the stream. A lower value is preferred. To monitor any performance issues, increase the number of consumers for your stream so that the data is processed more quickly. To optimize your application code, increase the number of consumers to reduce the delay in processing records.

**Question**

An e-commerce company runs its workloads on Amazon EMR clusters. The data analytics team at the company manually installs third-party libraries on the newly launched clusters by logging onto the master nodes. The team wants to develop an automated solution that will replace this human intervention.

Which of the following options would you recommend for the given requirement? (Select two)

**Answer**

**Upload the required installation scripts in Amazon S3 and execute them using custom bootstrap actions**

You can use a bootstrap action to install additional software or customize the configuration of the EMR cluster instances. Bootstrap actions are scripts that run on the cluster after Amazon EMR launches the instance using the Amazon Linux Amazon Machine Image (AMI). Bootstrap actions run before Amazon EMR installs the applications that you specify when you create the cluster and before cluster nodes begin processing data.

**Provision an Amazon EC2 instance with Amazon Linux and install the required third-party libraries on the instance. Create an AMI using this EC2 instance and then use this AMI to launch the EMR cluster**

You can create Amazon EMR clusters that have custom Amazon Machine Images (AMI) running Amazon Linux. You can create the AMI from an EC2 instance running Amazon Linux. Make sure that you have installed all the required third-party libraries on this EC2 instance. This allows you to preload additional software on your AMI and use these AMIs to launch your EMR clusters.

**Question**

A logistics company has multiple AWS accounts hosting its portfolio of IT applications that serve the company's retail and enterprise customers. A CloudWatch Logs agent is installed on each of the EC2 instances running these IT applications. The company wants to aggregate all security events in a centralized AWS account dedicated to log storage. The centralized operations team at the company needs to perform near-real-time gathering and collating events across multiple AWS accounts.

Which of the following solutions would you recommend to address these requirements?

**Answer**

**Set up Kinesis Data Firehose in the logging account and then subscribe the delivery stream to CloudWatch Logs streams in each application AWS account via subscription filters. Persist the log data in an Amazon S3 bucket inside the logging AWS account**

You can configure Amazon Kinesis Data Firehose to aggregate and collate CloudWatch Logs from different AWS accounts and receive their log events in a centralized logging AWS Account (this is known as cross-account data sharing) by using a CloudWatch Logs destination and then creating a Subscription Filter. This log event data can be read from a centralized Amazon Kinesis Firehose delivery stream to perform downstream processing and analysis.

You can collaborate with an owner of a different AWS account and receive their log events on your AWS resources, such as an Amazon Kinesis or Amazon Kinesis Data Firehose stream (this is known as cross-account data sharing). You can use a subscription filter with Kinesis Streams, Lambda, or Kinesis Data Firehose. Logs that are sent to a receiving service through a subscription filter are Base64 encoded and compressed with the gzip format.

***By partitioning your data***, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost. You can partition your data by any key. A common practice is to partition the data based on time, often leading to a multi-level partitioning scheme. For example, a customer who has data coming in every hour might decide to partition by year, month, date, and hour. Another customer, who has data coming from many different sources but that is loaded only once per day, might partition by a data source identifier and date.

While analyzing the new product launches for the different pet types, most of the queries will be related to the pet type, so partitioning data by pet type will reduce the amount of data scanned by Athena and hence reduce the associated costs.

***AWS Database Migration*** Service helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. With AWS Database Migration Service, you can continuously replicate your data with high availability and consolidate databases into a petabyte-scale data warehouse by streaming data to Amazon Redshift and Amazon S3.

You can migrate data to Amazon Redshift databases using AWS Database Migration Service. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. With an Amazon Redshift database as a target, you can migrate data from all of the other supported source databases.

The Amazon Redshift cluster must be in the same AWS account and the same AWS Region as the replication instance. During a database migration to Amazon Redshift, AWS DMS first moves data to an Amazon S3 bucket. When the files reside in an Amazon S3 bucket, AWS DMS then transfers them to the proper tables in the Amazon Redshift data warehouse. AWS DMS creates the S3 bucket in the same AWS Region as the Amazon Redshift database. The AWS DMS replication instance must be located in that same region

**Glue** is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics. AWS Glue job is meant to be used for batch ETL data processing.

Using AWS Glue involves significant development efforts to write custom migration scripts to copy the database data into Redshift

**Amazon EMR** is the industry-leading cloud big data platform for processing vast amounts of data using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink and Presto. With EMR you can run Petabyte-scale analysis at less than half of the cost of traditional on-premises solutions and over 3x faster than standard Apache Spark. For short-running jobs, you can spin up and spin down clusters and pay per second for the instances used. For long-running workloads, you can create highly available clusters that automatically scale to meet demand. Amazon EMR uses Hadoop, an open-source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances.

Using EMR involves significant infrastructure management efforts to set up and maintain the EMR cluster. Additionally, this option involves a major development effort to write custom migration jobs to copy the database data into Redshift.

**Amazon Kinesis Data Streams (KDS)** is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events.

However, the user is expected to manually provision an appropriate number of shards to process the expected volume of the incoming data stream. The throughput of an Amazon Kinesis data stream is designed to scale without limits via increasing the number of shards within a data stream. Additionally, KDS cannot directly write into Redshift since you will need to use Kinesis Data Firehose to deliver the stream into Redshift.

**Configure a daily AWS Glue ETL job to convert the data files to Apache Parquet format and partition these converted files. Create a periodic AWS Glue crawler to automatically crawl the partitioned data on a daily basis**

AWS states that you can improve the performance of your query by compressing, partitioning, or converting your data into columnar formats. Amazon Athena supports open-source columnar data formats such as Apache Parquet and Apache ORC. Converting your data into a compressed, columnar format lowers your cost and improves query performance by enabling Athena to scan fewer data from S3 when executing your query.

Therefore, converting the CSV files to Parquet format and partitioning will help improve the query performance in Amazon Athena.

Also, you can use AWS Glue crawlers to automatically infer database and table schema from your data in Amazon S3 and store the associated metadata in the AWS Glue Data Catalog. Athena uses the AWS Glue Data Catalog to store and retrieve table metadata for the Amazon S3 data in your Amazon Web Services account. The table metadata lets the Athena query engine know how to find, read, and process the data that you want to query.

Partitioning divides your table into parts and keeps the related data together based on column values such as date, country, region, etc. Partitions act as virtual columns. You define them at table creation, and they can help reduce the amount of data scanned per query, thereby improving performance. You can restrict the amount of data scanned by a query by specifying filters based on the partition.

***Question***

An e-commerce company wants to develop a user click analytics dashboard to see near-real-time user click patterns. The user clicks are currently ingested from various devices through Amazon Kinesis Data Streams. The dashboard must be refreshed automatically every ten seconds to display the most updated data. The company is looking for an easy-to-implement solution that can be put into production as soon as possible.

Which solution would you recommend for the given requirements?

**Use Amazon Kinesis Data Firehose to push the data into an Amazon OpenSearch Service. Visualize the data by using OpenSearch (Kibana) dashboards**

You can load streaming data into your Amazon OpenSearch Service domain from many different sources. Some sources, like Amazon Kinesis Data Firehose and Amazon CloudWatch Logs, have built-in support for OpenSearch Service. Others, like Amazon S3, Amazon Kinesis Data Streams, and Amazon DynamoDB, use AWS Lambda functions as event handlers.

Amazon Kinesis Data Firehose manages all underlying infrastructure, storage, networking, and configuration needed to capture and load your data into Amazon S3, Amazon Redshift, or Amazon OpenSearch Service. OpenSearch (Kibana) dashboards support auto-refresh of data.

***Question***

A legacy application is migrating its large dataset from the on-premises JDBC database to Amazon S3 for easy analysis and cost-effective storage. While migrating, the team noticed that the AWS Glue job runs for a long time and eventually fails with lost nodes.

As a Data Analyst, which of the following options would you suggest as a corrective measure to ensure the successful completion of the Glue job?

**Read the JDBC table in parallel**

AWS Glue uses a single connection to read the entire dataset. If you're migrating a large JDBC table, the ETL job might run for a long time without signs of progress on the AWS Glue side. Then, the job might eventually fail because of disk space issues (lost nodes). To resolve this issue, read the JDBC table in parallel. If the job still fails with lost nodes, use an SQL expression as a pushdown predicate.

If the table doesn't have numeric columns (INT or BIGINT), then use the hashfield option to partition the data. Set hashfield to the name of a column in the JDBC table. For best results, choose a column with an even distribution of values.

If the table has numeric columns, set the hashpartitions and hashexpression options in the table or while creating the DynamicFrame:

1. Hashpartitions: defines the number of tasks that AWS Glue creates to read the data
2. Hashexpression: divides rows evenly among tasks

If you're not partitioning the data appropriately, the data isn't distributed to the new nodes, and then you will receive the lost node errors.

**Question**

A company recently decided to go all-in on AWS and use the platform to host its website, order and stock management systems and fulfillment applications. The company wants to migrate its on-premises Oracle database to Aurora MySQL. The company has hired you to carry out the migration with minimal downtime using AWS DMS. The company has mandated that the migration must have minimal impact on the performance of the source database and you must validate that the data was migrated accurately from the source to the target before the cutover.

Which of the following solutions will MOST effectively address this use-case?

**Configure DMS data validation on the migration task so it can compare the source and target data for the DMS task and report any mismatches**

You can use AWS DMS data validation to ensure that your data has migrated accurately from the source to the target. DMS compares the source and target records and then reports any mismatches. In addition, for a CDC-enabled task, AWS DMS compares the incremental changes and reports any mismatches. As part of data validation, DMS compares each row in the source with its corresponding row at the target and verifies that those rows contain the same data. For this comparison, DMS issues appropriate queries to retrieve the data. These queries consume additional resources at the source and the target as well as additional network resources.

**Amazon S3** offers a range of storage classes that you can choose from based on the data access, resiliency, and cost requirements of your workloads. S3 storage classes are purpose-built to provide the lowest cost storage for different access patterns. S3 storage classes are ideal for virtually any use case, including those with demanding performance needs, data residency requirements, unknown or changing access patterns, or archival storage.

The S3 storage classes include **S3 Intelligent-Tiering** for automatic cost savings for data with unknown or changing access patterns, **S3 Standard** for frequently accessed data, **S3 Standard-Infrequent Access (S3 Standard-IA)** and **S3 One Zone-Infrequent Access (S3 One Zone-IA)** for less frequently accessed data, **S3 Glacier Instant Retrieval** for archive data that needs immediate access, **S3 Glacier Flexible Retrieval (formerly S3 Glacier)** for rarely accessed long-term data that does not require immediate access, and **Amazon S3 Glacier Deep Archive (S3 Glacier Deep Archive)** for long-term archive and digital preservation with retrieval in hours at the lowest cost storage in the cloud. If you have data residency requirements that can’t be met by an existing AWS Region, you can use the **S3 Outposts** storage class to store your S3 data on premises. Amazon S3 also offers capabilities to manage your data throughout its lifecycle. Once an S3 Lifecycle policy is set, your data will automatically transfer to a different storage class without any changes to your application.

## SUPPORTED TRANSITIONS AND RELATED CONSTRAINTS

In an S3 Lifecycle configuration, you can define rules to transition objects from one storage class to another to save on storage costs. When you don't know the access patterns of your objects, or if your access patterns are changing over time, you can transition the objects to the S3 Intelligent-Tiering storage class for automatic cost savings. For information about storage classes, see [Using Amazon S3 storage classes](https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-class-intro.html).

### **SUPPORTED LIFECYCLE TRANSITIONS**

Amazon S3 supports the following lifecycle transitions between storage classes using an S3 Lifecycle configuration. You can transition from the following:

* The S3 Standard storage class to any other storage class.
* Any storage class to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.
* The S3 Standard-IA storage class to the S3 Intelligent-Tiering, S3 One Zone-IA, or S3 Glacier Instant Retrieval storage classes.
* The S3 Intelligent-Tiering storage class to the S3 One Zone-IA storage class.
* The S3 Glacier Flexible Retrieval storage class to the S3 Glacier Deep Archive storage class.

**Tree Maps**

To visualize one or two measures for a dimension, you can use tree maps.

Each rectangle on the tree map represents one item in the dimension. Rectangle size represents the proportion of the value for the selected measure that the item represents compared to the whole for the dimension. You can optionally use rectangle color to represent another measure for the item. Rectangle color represents where the value for the item falls in the range for the measure, with darker colors indicating higher values and lighter colors indicating lower ones.

Color gradient customization on heat and tree maps allows you to select colors for lower, intermediate, and upper limits so that the gradient is applied within these colors. You can configure this under Color in the visual settings.

Tree maps can be used to represent hierarchical data in the form of nested rectangles. The nodes in this type of chart are categorized with two dimensions of size and color, giving us a very compact and data-rich visualization.

**Waterfall Charts** - Use a waterfall chart to visualize a sequential summation as values are added or subtracted. In a waterfall chart, the initial value goes through a (positive or negative) change, with each change represented as a bar. The final total is represented by the last bar. Waterfall charts are also known as bridges because the connectors between the bars bridge the bars together, showing that they visually belong to the same story.

**Funnel Charts** - Use a funnel chart to visualize data that moves across multiple stages in a linear process. In a funnel chart, each stage of a process is represented in blocks of different shapes and colors. The first stage, known as the head, is the largest block and is followed by the smaller stages, known as the neck, in a funnel shape. The size of the block representing each stage in a funnel chart is a percentage of the total and is proportionate to its value. The bigger the size of the block, the bigger its value.

Funnel charts are often useful in business contexts because you can view trends or potential problem areas in each stage, such as bottlenecks. For example, they can help you visualize the amount of the potential revenue in each stage of a sale, from first contact to final sale and on through maintenance.

**Heat Maps** - Use heat maps to show a measure for the intersection of two dimensions, with color-coding to easily differentiate where values fall in the range. Heat maps can also be used to show the count of values for the intersection of the two dimensions.

Each rectangle on a heat map represents the value for the specified measure for the intersection of the selected dimensions. Rectangle color represents where the value falls in the range for the measure, with darker colors indicating higher values and lighter colors indicating lower ones.

An **AWS Glue crawler** is a program that connects to a data store (source or target) such as Amazon S3, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the AWS Glue Data Catalog. You can use a Lambda function that is triggered by an S3:ObjectCreated:\* event notification on the S3 bucket to invoke the AWS Glue crawler on-demand. This obviates the need to periodically run the crawler on a schedule to update the new data into the existing data catalog tables.

You can use a crawler to populate the AWS Glue Data Catalog with tables. A crawler can crawl multiple data stores in a single run. Upon completion, the crawler creates or updates one or more tables in your Data Catalog. Extract, transform, and load (ETL) jobs that you define in AWS Glue use these Data Catalog tables as sources and targets. The ETL job reads from and writes to the data stores that are specified in the source and target Data Catalog tables.

**Store the data in Apache ORC, partitioned by date and sorted by device type of the device** - Apache Parquet and ORC are columnar storage formats that are optimized for fast retrieval of data and used in AWS analytical applications.

By partitioning your data, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost. You can partition your data by any key. A common practice is to partition the data based on time, often leading to a multi-level partitioning scheme. For example, a customer who has data coming in every hour might decide to partition by year, month, date, and hour. Another customer, who has data coming from many different sources but that is loaded only once per day, might partition by a data source identifier and date.

For the given use case, as the company does daily analysis, so it only needs to look at the data generated for a given date. Hence partitioning by date offers significant performance and cost advantages. Since the company also wants to do analysis for product improvements for each device type, it is better to keep the data sorted by device type, so it allows faster query execution.

**Use an exclude pattern for the Glue crawler to filter out the unwanted files**

An exclude pattern tells the crawler to skip certain files or paths. Exclude patterns reduce the number of files that the crawler must list, making the crawler run faster. For example, use an exclude pattern to exclude metafiles and files that have already been crawled.

***Quastion-*** *A Data Analyst is configuring Amazon Athena to create a table for each file stored under the same prefix in Amazon S3. After running CREATE TABLE statement in Athena with expected columns and their data types, the analyst has issued a SELECT query. However, the query has returned zero records.*

*Which of the following is the right way to configure the Amazon S3 location path?*

**When loading multiple files into a single table, use a single COPY command**

Amazon Redshift is a fully managed, scalable data warehouse that enables secure analytics at scale. Amazon Redshift is an MPP (massively parallel processing) database, where all the compute nodes divide and parallelize the work of ingesting data. Each node is further subdivided into slices, with each slice having one or more dedicated cores, equally dividing the processing capacity. When you load data into Amazon Redshift, you should aim to have each slice do an equal amount of work. When splitting your data files, ensure that they are of approximately equal size – between 1 MB and 1 GB after compression. The number of files should be a multiple of the number of slices in your cluster.

When loading multiple files into a single table, use a single COPY command for the table, rather than multiple COPY commands. Amazon Redshift automatically parallelizes the data ingestion. Using a single COPY command to bulk load data into a table ensures optimal use of cluster resources and the quickest possible throughput.

***Question-****A data analytics company needs to set up a data lake on Amazon S3 for a healthcare client. The data lake is split into raw and curated zones. For compliance reasons, the source data needs to be kept for a minimum of 5 years. The source data arrives in the raw zone and is then processed via an AWS Glue-based ETL job into the curated zone. The data analysts run ad-hoc queries only on the data in the curated zone using Athena. The team is concerned about the cost of data storage in both the raw and curated zones as the data is increasing at a rate of 2 TB daily in each zone.*

*Which of the following options would you implement together as the MOST cost-optimal solution? (Select two)*

**Setup a lifecycle policy to transition the raw zone data into Glacier Deep Archive after 1 day of object creation**-You can manage your objects so that they are stored cost-effectively throughout their lifecycle by configuring their Amazon S3 Lifecycle. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. For example, you might choose to transition objects to the S3 Standard-IA storage class 30 days after you created them, or archive objects to the S3 Glacier storage class one year after creating them.

For the given use-case, the raw zone consists of the source data, so it cannot be deleted due to compliance reasons. Therefore, you should use a lifecycle policy to transition the raw zone data into Glacier Deep Archive after 1 day of object creation.

**Use Glue ETL job to write the transformed data in the curated zone using a compressed file format**-AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics.

You cannot transition the curated zone data into Glacier Deep Archive because it is used by business analysts for ad-hoc querying. Therefore, the best optimization is to have the curated zone data stored in a compressed format via the Glue job. The compressed data would reduce the storage cost incurred on the data in the curated zone.

***Question-*** *A Data Analyst is troubleshooting a failed AWS Glue ETL job that transforms data to be used for downstream query processing via Athena. While troubleshooting, the analyst noticed that some of the tables that are visible from the AWS Glue console aren't visible on the console of Amazon Athena.*

*Which of the following would you identify as the most likely reasons for this behavior? (Select two)*

**AWS Glue lists tables that are created from Amazon DynamoDB tables**

**AWS Glue Data Catalog has a list of tables created in XML format**

You might see more tables in the AWS Glue console than in the Athena console for the following reasons:

1. You've created tables that point to different data sources. The Athena console displays tables that point to Amazon Simple Storage Service (Amazon S3) paths only. AWS Glue lists tables that point to different data sources, such as Amazon Relational Database Service (Amazon RDS) DB instances and Amazon DynamoDB tables.
2. You've created tables in formats that aren't supported by Athena, such as XML. These tables appear in the AWS Glue Data Catalog, but not in the Athena console.

***Question-*** *A popular healthcare brand sells drugs through an online portal as well as their physical stores. The data from all these transactions is fed into Amazon Kinesis Data Streams using the Kinesis Producer Library (KPL). The Data Streams are managed via Auto Scaling configuration. On the other hand, Kinesis Client Library (KCL) ingests the incoming data into the company's warehousing system to be used for downstream analytics. Lately, the support team has come across issues arising out of duplicate records.*

*Which of the following would you identify as the most likely reason for this behavior?*

**The producer is experiencing network-related timeouts, forcing duplicate entries into the Kinesis Data Stream** - There are two primary reasons why records may be delivered more than once to your Amazon Kinesis Data Streams application: producer retries and consumer retries.

Consider a producer that experiences a network-related timeout after it makes a call to PutRecord, but before it can receive an acknowledgment from Amazon Kinesis Data Streams. The producer cannot be sure if the record was delivered to Kinesis Data Streams. Assuming that every record is important to the application, the producer would have been written to retry the call with the same data. If both PutRecord calls on that same data were successfully committed to Kinesis Data Streams, then there will be two Kinesis Data Streams records.

Applications that need strict guarantees should embed a primary key within the record to remove duplicates later when processing. Note that the number of duplicates due to producer retries is usually low compared to the number of duplicates due to consumer retries.

*Question- A Company has developed an end-to-end AWS cloud-based Internet-of-Things (IoT) solution that provides customers with integrated IoT functionality in devices including baby monitors, security cameras and entertainment systems. The company is using Kinesis Data Streams (KDS) to process IoT data from these devices. Multiple consumer applications are using the incoming data streams and the engineers have noticed a performance lag for the data delivery speed between producers and consumers of the data streams.*

*Which of the following would you suggest to improve the performance for the given use case?*

**Use Enhanced Fanout feature of Kinesis Data Streams to support the desired read throughput for the downstream applications**

Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. By default, the 2MB/second/shard output is shared between all of the applications consuming data from the stream.

You should use enhanced fan-out if you have multiple consumers retrieving data from a stream in parallel. With enhanced fan-out developers can register stream consumers to use enhanced fan-out and receive their own 2MB/second pipe of read throughput per shard, and this throughput automatically scales with the number of shards in a stream.

***Question-*** *A subscription streaming service company uses AWS Cloud for analytics, recommendation engines and video transcoding. To monitor and optimize this network, the analytics team at the company has developed a solution for ingesting, augmenting, and analyzing the multiple terabytes of data its network generates daily in the form of virtual private cloud (VPC) flow logs. This would enable the company to identify performance-improvement opportunities such as identifying apps that are communicating across regions and collocating them. The VPC flow logs data is funneled into Kinesis Data Streams which further acts as the source of a delivery stream for Kinesis Firehose. The engineering team has now configured a Kinesis Agent to send the VPC flow logs data from another set of network devices to the same Firehose delivery stream. They noticed that data is not reaching Firehose as expected.*

*Which of the following options would you identify as the MOST plausible root cause behind this issue?*

**Kinesis Agent cannot write to a Kinesis Firehose for which the delivery stream source is already set as Kinesis Data Streams**

Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics tools. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, transform, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security.

When a Kinesis data stream is configured as the source of a Firehose delivery stream, Firehose’s PutRecord and PutRecordBatch operations are disabled and Kinesis Agent cannot write to Firehose delivery stream directly. Data needs to be added to the Kinesis data stream through the Kinesis Data Streams PutRecord and PutRecords operations instead. Therefore, this option is correct.

***Question****- The web development team at an IT company has about 200 TB of web-log data that is stored in an Amazon S3 bucket as raw text. Each log file is identified by a key of the type year-month-day\_log\_HHmmss.txt where HHmmss denotes the time the log file was created. The data analytics team has created an Amazon Athena table that links to the given S3 bucket. The data analytics team executes several queries every hour against a subset of the table's columns. The company wants a Hive-metastore compatible solution that costs less and requires less maintenance to support the ongoing analytics on this log data.*

*As an AWS Certified Data Analytics Specialist, which of the following solutions would you combine to address these requirements? (Select three)*

**Change the log files to Apache Parquet format**

**Partition the data by using a key prefix of the form date=year-month-day/ to the S3 objects**

**Drop and recreate the table with the PARTITIONED BY clause. Load the partitions by executing the MSCK REPAIR TABLE statement**

Amazon Athena is an interactive query service that makes it easy to analyze data stored 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.

By partitioning your data, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost. You can partition your data by any key. A common practice is to partition the data based on time, often leading to a multi-level partitioning scheme.

Athena can use Apache Hive style partitions, whose data paths contain key value pairs connected by equal signs (for example, country=us/... or year=2021/month=01/day=26/...). Thus, the paths include both the names of the partition keys and the values that each path represents.

Athena can also use non-Hive style partitioning schemes. For example, CloudTrail logs and Kinesis Data Firehose delivery streams use separate path components for date parts such as data/2021/01/26/us/6fc7845e.json. For such non-Hive compatible data, you use ALTER TABLE ADD PARTITION to add the partitions manually.

Since the given use case needs a hive-metastore compatible solution, you can use a key prefix of the form date=year-month-day/ for partitioning data and use MSCK REPAIR TABLE statement to load the partitions.

***Question****-A government healthcare agency receives multiple compressed (gzip) CSV files containing data about contagious diseases for the past month aggregated from all government-managed hospitals. The files are about ~300GB and are stored in Amazon S3 Glacier. As per the government guidelines, the agency needs to query a portion of this data to prepare a report every month.*

*Which of the following is the most cost-effective way to query this data?*

**Load the data into Amazon S3 from S3 Glacier and query the required data with Amazon S3 Select** - S3 Select is an Amazon S3 feature that makes it easy to retrieve specific data from the contents of an object using simple SQL expressions without having to retrieve the entire object. S3 Select simplifies and improves the performance of scanning and filtering the contents of objects into a smaller, targeted dataset by up to 400%. With S3 Select, you can also perform operational investigations on log files in Amazon S3 without the need to operate or manage a compute cluster.

You can use S3 Select to retrieve a subset of data using SQL clauses, like SELECT and WHERE, from objects stored in CSV, JSON, or Apache Parquet format. It also works with objects that are compressed with GZIP or BZIP2 (for CSV and JSON objects only) and server-side encrypted objects.

You can use S3 Select with AWS Lambda to build serverless applications that use S3 Select to efficiently and easily retrieve data from Amazon S3 instead of retrieving and processing entire objects. You can also use S3 Select with Big Data frameworks, such as Presto, Apache Hive, and Apache Spark to scan and filter the data in Amazon S3.

***Question*** *- A company runs an analytics workload with heavy reads and writes through the workload lifecycle. The data analytics team at the company is interested in using Amazon S3 as the data lake to support this workload. The team has hired you to advise them on the S3 data consistency model.*

*Which of the following statements would you identify as correct?*

**Amazon S3 is strongly consistent for all GET, PUT and LIST operations** - After a successful write of a new object, or an overwrite or delete of an existing object, any subsequent read request immediately receives the latest version of the object. S3 also provides strong consistency for list operations, so after a write, you can immediately perform a listing of the objects in a bucket with any changes reflected.

For all existing and new objects, and in all regions, all S3 GET, PUT, and LIST operations, as well as operations that change object tags, ACLs, or metadata, are now strongly consistent. What you write is what you will read, and the results of a LIST will be an accurate reflection of what’s in the bucket.

This improvement is great for data lakes, but other types of applications will also benefit. Because S3 now has strong consistency, migration of on-premises workloads and storage to AWS should now be easier than ever before.

***Question -*** *An e-commerce application runs on a single EC2 instance and processes one Kinesis data stream that has four shards. The instance has one KCL worker configured on it. As part of application scaling, another EC2 instance has been added to this configuration.*

*What is the outcome of this change?*

**When the KCL worker starts up on the second instance, it load-balances with the first instance, and each instance will now process two shards** - Resharding enables you to increase or decrease the number of shards in a stream to adapt to changes in the rate of data flowing through the stream. Resharding is typically performed by an administrative application that monitors shard data-handling metrics. Although the KCL itself doesn't initiate resharding operations, it is designed to adapt to changes in the number of shards that result from resharding.

KCL tracks the shards in the stream using an Amazon DynamoDB table. When new shards are created as a result of resharding, the KCL discovers the new shards and populates new rows in the table. The workers automatically discover the new shards and create processors to handle the data from them. The KCL also distributes the shards in the stream across all the available workers and record processors.

The following example illustrates how the KCL helps you handle scaling and resharding:

1. For example, if your application is running on one EC2 instance, and is processing one Kinesis data stream that has four shards. This one instance has one KCL worker and four record processors (one record processor for every shard). These four record processors run in parallel within the same process.
2. Next, if you scale the application to use another instance, you have two instances processing one stream that has four shards. When the KCL worker starts up on the second instance, it load-balances with the first instance, so that each instance now processes two shards.
3. If you then decide to split the four shards into five shards. The KCL again coordinates the processing across instances: one instance processes three shards, and the other processes two shards. Similar coordination occurs when you merge shards.

***Question*** *- A data analytics company wants to store data from Athena CTAS (CREATE TABLE AS SELECT) query results in Amazon S3. A junior analyst wants to understand the distinction between partitioning scheme vs bucketing scheme for storing data via such CTAS queries.*

*As an AWS Certified Data Analytics Specialist, which of the following options would you identify as the right fit for choosing bucketing vs partitioning? (Select two)*

**Partitioning CTAS query results works well when the number of partitions you plan to have is limited and the partitioned columns have low cardinality**

**Bucketing CTAS query results works well when you bucket data by the column that has high cardinality and evenly distributed values**

By partitioning your data, you can restrict the amount of data scanned by each query, thus improving performance and reducing cost. You can partition your data by any key. A common practice is to partition the data based on time, often leading to a multi-level partitioning scheme. For example, a customer who has data coming in every hour might decide to partition by year, month, date, and hour. Another customer, who has data coming from many different sources but that is loaded only once per day, might partition by a data source identifier and date.

When you run a CTAS query, Athena writes the results to a specified location in Amazon S3. If you specify partitions, it creates them and stores each partition in a separate partition folder in the same location. Having partitions in Amazon S3 helps with Athena query performance because this helps you run targeted queries for only specific partitions. As a best practice, you should partition data by those columns that have similar characteristics, such as records from the same department, and that can have a limited number of possible values (low cardinality), such as a limited number of distinct departments in an organization.

Bucketing CTAS query results works well when you bucket data by the column that has high cardinality and evenly distributed values. For example, columns storing timestamp data could potentially have a very large number of distinct values, and their data is evenly distributed across the data set. To choose the column by which to bucket the CTAS query results, use the column that has a high number of values (high cardinality) and whose data can be split for storage into many buckets that will have roughly the same amount of data. Columns that are sparsely populated with values are not good candidates for bucketing.

***Question***

*An AWS Glue job is scheduled to be run on the Sunday of every week. The Glue job copies data from certain folders in an S3 bucket to Redshift. To prevent reprocessing of old data, job bookmarks have been enabled on the AWS Glue job. However, the ETL job is reprocessing data that was already processed in an earlier run.*

*What could be the underlying issue and how should it be fixed to stop reprocessing of data? (Select two)*

**You have multiple concurrent jobs with job bookmarks, and the max concurrency isn't set to 1** - Ensure that the maximum number of concurrent runs for the job is 1. When you have multiple concurrent jobs with job bookmarks and the maximum concurrency is set to 1, the job bookmark doesn't work correctly.

**The job.commit() object is missing** - Ensure that your job run script ends with the following commit: job.commit(). When you include this object, AWS Glue records the timestamp and path of the job run. If you run the job again with the same path, AWS Glue processes only the new files. If you don't include this object and job bookmarks are enabled, the job reprocesses the already processed files along with the new files and creates redundancy in the job's target data store.

***Question***

*A company runs a real-time data processing application that uses Kinesis Client Library (KCL) to help consume and process data from the real-time data streams. The development team has raised a query on the viability of using the same DynamoDB table for different KCL applications.*

*Which of the following are correct statements for KCL while consuming Kinesis Data Streams? (Select two)*

**Each KCL application must use its own DynamoDB table**

Users can't use different KCL applications with the same DynamoDB table for the following reasons:

1. Scan operations are used to obtain leases from a DynamoDB table. Therefore, if a table contains leases of different KCL applications, each application could receive a lease that isn't related to the application itself.
2. Shard IDs in streams are used as primary keys in DynamoDB tables during checkpointing. When different KCL applications use the same DynamoDB table and the same shard IDs are used in the streams, inconsistencies in checkpoints can occur.

**You can only use DynamoDB for checkpointing KCL** - Users can only use DynamoDB as a checkpointing table for the KCL. A DynamoDB table is required as a checkpointing table for the KCL because the KCL behavior and implementation are interconnected with DynamoDB in the following ways:

1. The KCL includes ShardSyncTask.java, which guarantees that shard leases in a stream are included in the DynamoDB table. This check is conducted periodically in the KCL.
2. The KCL includes DynamoDBLeaseTaker.java and DynamoDBLeaseRenewer.java, which are components that manage and update leases in the KCL. DynamoDBLeaseTaker.java and DynamoDBLeaseRenewer.java work with DynamoDBLeaseRefresher.java to make frequent API requests to DynamoDB.
3. When the KCL makes checkpoints, requests from DynamoDBCheckpointer.java and DynamoDBLeaseCoordinator.java are made to DynamoDB.

***Question***

*A data analytics company leverages AWS Cloud to process Internet of Things (IoT) sensor data from the field devices of a logistics company. The analytics company stores the sensor data in Amazon DynamoDB tables. To detect anomalous behaviors and respond quickly, all changes to the items stored in the DynamoDB tables must be logged in near real-time.*

*As an AWS Certified Data Analytics Specialist, which of the following solutions would you suggest to meet the requirements of the given use-case so that it requires minimal custom development and infrastructure maintenance?*

**Set up DynamoDB Streams to capture and send updates to a Lambda function that outputs records to Kinesis Data Analytics (KDA) via Kinesis Data Streams (KDS). Detect and analyze anomalies in KDA and send notifications via SNS**

A DynamoDB stream is an ordered flow of information about changes to items in a DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table for up to 24 hours.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attributes of the items that were modified. A stream record contains information about a data modification to a single item in a DynamoDB table.

DynamoDB Streams supports the following stream record views:

KEYS\_ONLY — Only the key attributes of the modified item NEW\_IMAGE — The entire item, as it appears after it was modified OLD\_IMAGE — The entire item, as it appears before it was modified NEW\_AND\_OLD\_IMAGES — Both the new and the old images of the item

You can process DynamoDB streams in multiple ways. The most common approaches use AWS Lambda or a standalone application that uses the Kinesis Client Library (KCL) with the DynamoDB Streams Kinesis Adapter. The KCL is a client-side library that provides an interface to process DynamoDB stream changes. If you enable DynamoDB Streams on a table, you can associate the stream Amazon Resource Name (ARN) with an AWS Lambda function that you write. Immediately after an item in the table is modified, a new record appears in the table's stream. AWS Lambda polls the stream and invokes your Lambda function synchronously when it detects new stream records.

***Question***

*A news media company uses an ad-hoc Kinesis Forehose based solution to ingest raw data in JSON format and then deliver it to an Amazon S3 bucket. The data analytics team at the company uses Apache Spark SQL to analyze this data via Amazon EMR, which is configured to use AWS Glue Data Catalog as the metastore. An AWS Glue crawler runs every four hours to update the schema of the data catalog. The team has noticed that it sometimes obtains outdated data. You have been hired by the company as an AWS Certified Data Analytics Specialist to build a solution for ensuring that the team always has access to the current data.*

*Which of the following represents the best solution to meet this requirement?*

An AWS Glue crawler is a program that connects to a data store (source or target) such as Amazon S3, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the AWS Glue Data Catalog. You can use a Lambda function that is triggered by an S3:ObjectCreated:\* event notification on the S3 bucket to invoke the AWS Glue crawler on-demand. This obviates the need to periodically run the crawler on a schedule to update the new data into the existing data catalog tables.

***Question***

*A business is moving their data to Amazon Redshift. A core table with billions of rows needs to be moved to Redshift. This table contains certain columns that have sensitive data that can only be accessed by the finance team. Once the data is moved to Redshift, queries will be run on this table by multiple teams.*

*How will you configure the requirement such that the columns holding sensitive data are only accessible to members of the finance team?*

**Grant the finance team (defined as a group) permissions to read from the table. Use the GRANT SQL command to allow read-only access to a subset of columns having non-sensitive data to the other users**

Since March 2020, Amazon Redshift supports column-level access control for data in Redshift. Customers can use column-level GRANT and REVOKE statements to help meet their security and compliance needs.

Redshift's table-level access controls for the data in Redshift are already in use by many customers, but they also want the ability to control access in more detail. You can now control access to columns without having to implement view-based access control or use another system. Column-level access control is available in all Amazon Redshift regions.

GRANT command defines access privileges for a user or user group. Privileges include access options such as being able to read data in tables and views, write data, create tables, and drop tables. Use this command to give specific privileges for a table, database, schema, function, procedure, language, or column.

The syntax for column-level privileges on Amazon Redshift tables and views looks like the below:

GRANT { { SELECT | UPDATE } ( column\_name [, ...] ) [, ...] | ALL [ PRIVILEGES ] ( column\_name [,...] ) } ON { [ TABLE ] table\_name [, ...] } TO { username | GROUP group\_name | PUBLIC } [, ...]

***Question***

*A stock trading company uses Amazon Redshift to power the Business Intelligence (BI) specific queries which are run on Redshift. The data analytics team at the company needs to provide the sales team access to a historical trades table whose data is stored in Apache Parquet format in an S3 bucket of the company's data lake. The data analytics team should provide access to only a few specific columns in the historical trades table so that the access does not violate the compliance regulations.*

*Which of the following options should be combined together to build a solution for the given use case? (Select three)*

**Create an IAM role for Amazon Redshift which has a policy to allow Redshift to access AWS Lake Formation**

**Create an external schema in Amazon Redshift by using the Amazon Redshift IAM role**

**Grant permissions in Lake Formation to allow the Amazon Redshift IAM role to access the specific columns of the historical trades table**

Amazon Redshift is a fully managed, scalable data warehouse that enables secure analytics at scale. Using Redshift Spectrum, Amazon Redshift customers can easily query their data in Amazon S3. Redshift Spectrum is a built-in feature of Amazon Redshift, and your existing queries and BI tools will continue to work seamlessly. Under the hood, AWS manages a fleet of thousands of Redshift Spectrum nodes spread across multiple Availability Zones. These are transparently scaled and allocated to your queries based on the data that you need to process, with no provisioning or commitments. Redshift Spectrum is also highly concurrent—you can access your Amazon S3 data from any number of Amazon Redshift clusters. To access this data on S3 via Redshift Spectrum, you need to create an external schema in Amazon Redshift.

Amazon Redshift Spectrum supports column level access control for data stored in Amazon S3 and managed by AWS Lake Formation. Column level access control can be used to limit access to only the specific columns of a table rather than allowing access to all columns of a table. To use this feature, an administrator needs to create an IAM role for Amazon Redshift and create the policy to allow Redshift to access AWS Lake Formation. The administrator can then use the Lake Formation console to specify the tables and columns that the role is allowed access to. The column level access control policies can also be created and managed by the SQL grant statements.

***Question***

*An AWS Database Migration Service (AWS DMS) task is migrating data to Amazon Redshift as the target. The task has changed to one-by-one mode even though the configuration was set to bulk operation mode.*

*What is the reason for this change and how can it be fixed?*

**The bulk operation task of AWS DMS failed, forcing the task to change to one-by-one mode. When all the transactions from this failed batch are applied, the AWS DMS will switch back to Batch Apply mode automatically**

AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS Database Migration Service supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora.

When a migration task that is replicating data to Amazon Redshift has an issue applying a batch, AWS DMS doesn't fail the whole batch. AWS DMS breaks the batch down and switches to a one-by-one mode to apply transactions. When AWS DMS encounters the transaction that caused the batch to fail, AWS DMS logs the transaction to the awsdms\_apply\_exceptions table on the Amazon Redshift target. Then, AWS DMS applies the other transactions in the batch one by one until all transactions from that batch are applied onto the target. Finally, AWS DMS switches back to Batch Apply mode for a new batch and continues to use Batch Apply unless another batch fails.

***Question***

*A subscription streaming service delivers billions of hours of content from Amazon S3 to customers around the world. Amazon S3 also serves as the data lake for its big data analytics solution. The data lake has a staging zone where intermediary query results are kept only for 24 hours. These results are also heavily referenced by other parts of the analytics pipeline.*

*Which of the following is the MOST cost-effective option to store this intermediary query data?*

**Store the intermediary query results in S3 Standard storage class**

S3 Standard offers high durability, availability, and performance object storage for frequently accessed data. Because it delivers low latency and high throughput, S3 Standard is appropriate for a wide variety of use cases, including cloud applications, dynamic websites, content distribution, mobile and gaming applications, and big data analytics. As there is no minimum storage duration charge and no retrieval fee (remember that intermediary query results are heavily referenced by other parts of the analytics pipeline), this is the MOST cost-effective storage class amongst the given options.

***Question***

*An IT company stores a huge volume of data on Amazon S3 buckets. For a new business requirement, the company needs to copy this data from Amazon S3 into HDFS hosted on the Amazon EMR cluster.*

*Which is the most optimal way of copying this data from S3 to EMR cluster?*

**Use S3DistCp to copy data between Amazon S3 and Amazon EMR clusters** - Use S3DistCp to copy data between Amazon S3 and Amazon EMR clusters. S3DistCp is installed on Amazon EMR clusters by default.

S3DistCp is similar to DistCp, but optimized to work with AWS, particularly Amazon S3. The command for S3DistCp in Amazon EMR version 4.0 and later is s3-dist-cp, which you add as a step in a cluster or at the command line. Using S3DistCp, you can efficiently copy large amounts of data from Amazon S3 into HDFS where it can be processed by subsequent steps in your Amazon EMR cluster. You can also use S3DistCp to copy data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCp is more scalable and efficient for parallel copying large numbers of objects across buckets and across AWS accounts.

Like DistCp, S3DistCp uses MapReduce to copy in a distributed manner. It shares the copy, error handling, recovery, and reporting tasks across several servers.

***Question***

*A company wants to store all of its consumer data on Amazon S3. Before storing the data, the company must clean it by standardizing the formats of a few of the data columns. A single data record might range in size from 500 KB to 10 MB.*

*Which of these options represents the right solution?*

**Use Amazon Managed Streaming for Apache Kafka. Create a topic for the initial raw data. Use a Kafka producer to write data on this topic. Use the Apache Kafka consumer API to create a consumer application (that can be hosted on Amazon EC2 instance) that reads data from this topic, transforms the data as needed, and writes it to Amazon S3 for final storage**

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that makes it easy for you to build and run applications that use Apache Kafka to process streaming data. Apache Kafka is an open-source platform for building real-time streaming data pipelines and applications. With Amazon MSK, you can use native Apache Kafka APIs to populate data lakes, stream changes to and from databases, and power machine learning and analytics applications.

Apache Kafka stores records in topics. Data producers write records to topics and consumers read records from topics. Each record in Apache Kafka consists of a key, a value, and a timestamp.

Amazon MSK is preferred over Amazon Kinesis Data Streams and Kinesis Firehose because of the laters limitations on record size. The maximum record size of an Amazon MSK is 100 MB.

**Question**

*As part of application functionality, data is pushed from Amazon Kinesis Data Firehose to Amazon Simple Storage Service (Amazon S3). However, the development team noticed that Kinesis Data Firehose is creating several small files in the Amazon S3 bucket, as opposed to a much lower expected number of files.*

*Which of the following would you attribute as the most likely cause behind this issue?*

**Kinesis Data Firehose delivery stream has scaled**

If a limit increase was requested or Kinesis Data Firehose has automatically scaled, then the Data Firehose delivery stream can scale. By default, Kinesis Data Firehose automatically scales delivery streams up to a certain limit. Amazon Kinesis' automatic scaling behavior reduces the likelihood of throttling without requiring a limit increase.

When Kinesis Data Firehose's delivery stream scales, it can cause an effect on the buffering hints of Data Firehose. The overall buffer size (SizeInMBs) of the delivery stream scales proportionally but inversely. For example, if the capacity of Kinesis Data Firehose increases by two times the original buffer size limit, the buffer size is halved. If Kinesis Data Firehose scales up to four times, the buffer size reduces to one-quarter of the overall buffer size.

There is also a proportional number of parallel buffering within the Kinesis Data Firehose delivery stream, where data is delivered simultaneously from all these buffers. For example, Kinesis Data Firehose can buffer the data and create a single file based on the buffer size limit. If Kinesis Data Firehose scales to double the buffer limit, then two separate channels will create the files within the same time interval. If Kinesis Data Firehose scales up to four times, there will be four different channels creating four files in S3 during the same time interval.

***Question***

*A university is carrying out research on multiple economic parameters such as student loan, student credit card usage, monthly spends, etc for its students from different countries. The university wants to identify the trends and outliers present in this data. The entire data is stored in Parquet format on Amazon S3.*

*Which of the following would you recommend as the best option to visualize this data with the least effort?*

**Use heat map visualizations in Amazon QuickSight with Amazon Athena as the data source**

Amazon QuickSight allows everyone in your organization to understand your data by asking questions in natural language, exploring through interactive dashboards, or automatically looking for patterns and outliers powered by machine learning.

Use heat maps to show a measure for the intersection of two dimensions, with color-coding to easily differentiate where values fall in the range. Heat maps can also be used to show the count of values for the intersection of the two dimensions.

Each rectangle on a heat map represents the value for the specified measure for the intersection of the selected dimensions. Rectangle color represents where the value falls in the range for the measure, with darker colors indicating higher values and lighter colors indicating lower ones.

Heat maps and pivot tables display data in a similar tabular fashion. Use a heat map if you want to identify trends and outliers because the use of color makes these easier to spot. Use a pivot table if you want to further analyze data on the visual, for example by changing column sort order or applying aggregate functions across rows or columns.

***Question***

*A company has created a data warehouse using Redshift that is used to analyze data from Amazon S3. From the usage patterns, the data analytics team has detected that after 30 days, the data is rarely queried in Redshift and it's not "hot data" anymore. The team would like to preserve the SQL querying capability on the data and get the queries started immediately. Also, the team wants to adopt a pricing model that allows the company to save the maximum amount of cost on Redshift.*

*As an AWS Certified Data Analytics Specialist, which of the following options would you recommend? (Select two)*

**Move the data to S3 Standard IA after 30 days** - S3 Standard-IA is for data that is accessed less frequently but requires rapid access when needed. S3 Standard-IA offers high durability, high throughput, and low latency of S3 Standard, with a low per GB storage price and per GB retrieval fee. This combination of low cost and high performance makes S3 Standard-IA ideal for long-term storage, backups, and as a data store for disaster recovery files. The minimum storage duration charge is 30 days.

**Analyze the cold data with Athena** - Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to set up or manage, and customers pay only for the queries they run. You can use Athena to process logs, perform ad-hoc analysis, and run interactive queries.

Moving the data to S3 Glacier will prevent us from being able to query it. Therefore, we should migrate the data to S3 Standard IA and use Athena to analyze the cold data.

***Question***

*A trading firm wants to migrate its on-premises Apache Hadoop cluster to an Amazon Elastic Map Reduce (EMR) cluster. The cluster is only operational during normal business hours. The EMR cluster must be highly available to prevent intraday cluster failures. The data must survive when the cluster is terminated at the end of each business day.*

*Which of the following options would you recommend to address these requirements? (Select three)*

**EMR File System (EMRFS) for storage**

The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like data encryption. This ensures that the data persists even when the cluster is terminated at the end of each business day.

**Multiple master nodes in a single Availability Zone**

An EMR cluster with multiple master nodes ensures that the master node is no longer a single point of failure. If one of the master nodes fails, the cluster uses the other two master nodes and runs without interruption. This would ensure high availability to prevent intraday cluster failures.

**AWS Glue Data Catalog as the metastore for Apache Hive** - The AWS Glue Data Catalog is an index to the location, schema, and runtime metrics of your data. You use the information in the Data Catalog to create and monitor your ETL jobs. The catalog would persist even when the cluster is terminated at the end of each business day.

***Question***

*A research agency stores and manages the global seismological data for the last 100 years. The data has a velocity of 1GB per minute. You would like to store the data with only the most relevant attributes to build a predictive model for earthquakes.*

*Which of the following solutions would you use to build the most cost-effective solution with the LEAST amount of infrastructure maintenance?*

**Ingest the data in Kinesis Data Firehose and use an intermediary Lambda function to filter and transform the incoming stream before the output is dumped on S3**

Amazon Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you’re already using today. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security.

***Question***

A consulting firm uses Amazon Athena to analyze data in Amazon S3 using SQL. A rapid expansion plan has resulted in the company doubling its data analysts in a year resulting in high usage costs for Athena. Preliminary investigation suggests that most of the day-to-day queries run for only a few seconds fetching limited data. The firm wants a cap on the amount of data each query can fetch while also defining different thresholds on hourly or daily aggregates on data scanned by the queries.

As an AWS Certified Data Analytics Specialist, how will you configure this requirement?

**Configure multiple per-workgroup limits by utilizing the workgroup-wide data usage control limit on Athena** - Athena allows you to set two types of cost controls: per-query limit and per-workgroup limit. For each workgroup, you can set only one per-query limit and multiple per-workgroup limits.

The workgroup-wide data usage control limit specifies the total amount of data scanned for all queries that run in this workgroup during the specified time period. You can create multiple limits per workgroup. The workgroup-wide query limit allows you to set multiple thresholds on hourly or daily aggregates on data scanned by queries running in the workgroup.

If the aggregate amount of data scanned exceeds the threshold, you can push a notification to an Amazon SNS topic. You can also create an alarm and an action on any metric that Athena publishes from the CloudWatch console.

***Question***

*A healthcare company uses an Amazon Redshift database cluster to store sensitive user data. The regulatory guidelines mandate logging so that any database authentication attempts as well as the connections/disconnections are recorded. Also, the logs must include a record of each query executed against the database along with the database user who executed that query.*

*Which of the following options represent the best solution for these requirements?*

**Enable audit logging for Amazon Redshift**

Amazon Redshift logs information about connections and user activities in your database. These logs help you to monitor the database for security and troubleshooting purposes, a process called database auditing. The logs are stored in Amazon S3 buckets.

Audit logging is not turned on by default in Amazon Redshift. When you turn on logging on your cluster, Amazon Redshift creates and uploads logs to Amazon S3 that capture data from the time audit logging is enabled to the present time. Each logging update is a continuation of the information that was already logged.

***Question***

*Consider the following scenario on Amazon S3: A folder INPUT-FOLDER1 has 10 files, 8 files with schema SCH\_A and 2 files with SCH\_B, and another folder INPUT-FOLDER2 has 10 files, 7 files with the schema SCH\_A and 3 files with the schema SCH\_B. The schemas are defined as follows:*

*SCH\_A:*

*{ "id": 1, "first\_name": "John", "last\_name": "Doe"}*

*{ "id": 2, "first\_name": "Li", "last\_name": "Juan"}*

*SCH\_B:*

*{"city":"Dublin","country":"Ireland"}*

*{"city":"Paris","country":"France"}*

*What is the outcome, when the crawler crawls the Amazon Simple Storage Service (Amazon S3) path s3://INPUT-FOLDER1 and s3://INPUT-FOLDER2 separately?*

**For S3 path s3://INPUT-FOLDER1, the crawler creates one table with columns of both the schemas. And for S3 path s3://INPUT-FOLDER2, the crawler creates two tables, each table having columns of one schema respectively**

For schemas to be considered similar, the following conditions must be true: 1. The partition threshold is higher than 0.7 (70%). 2. The maximum number of different schemas (also referred to as "clusters" in this context) doesn't exceed 5.

The crawler infers the schema at the folder level and compares the schemas across all folders. If the schemas that are compared match, that is, if the partition threshold is higher than 70%, then the schemas are denoted as partitions of a table. If they don’t match, then the crawler creates a table for each folder, resulting in a higher number of tables.

Suppose that the folder DOC-EXAMPLE-FOLDER1 has 10 files, 8 files with schema SCH\_A and 2 files with SCH\_B.

Suppose that the files with the schema SHC\_A are similar to the following:

{ "id": 1, "first\_name": "John", "last\_name": "Doe"}

{ "id": 2, "first\_name": "Li", "last\_name": "Juan"}

Suppose that the files with the schema SCH\_B are similar to the following:

{"city":"Dublin","country":"Ireland"}

{"city":"Paris","country":"France"}

When the crawler crawls the Amazon Simple Storage Service (Amazon S3) path s3://DOC-EXAMPLE-FOLDER1, the crawler creates one table. The table comprises columns of both schema SCH\_A and SCH\_B. This is because 80% of the files in the path belong to the SCH\_A schema, and 20% of the files belong to the SCH\_B schema. Therefore, the partition threshold value is met. Also, the number of different schemas hasn't exceeded the number of clusters, and the cluster size limit isn't exceeded.

Suppose that the folder DOC-EXAMPLE-FOLDER2 has 10 files, 7 files with the schema SCH\_A and 3 files with the schema SCH\_B.

When the crawler crawls the Amazon S3 path s3://DOC-EXAMPLE-FOLDER2, the crawler creates one table for each file. This is because 70% of the files belong to the schema SCH\_A and 30% of the files belong to the schema SCH\_B. This means that the partition threshold isn't met. You can check the crawler logs in Amazon CloudWatch to get information on the created tables.

***Question***

*A data-storage service uses Amazon S3 under the hood to power its storage offerings which allow the customers to upload and view the files immediately. Currently, all the customer files are uploaded directly under a single S3 bucket. The data analytics team has started seeing scalability issues where customer file uploads are failing during the peak access hours with more than 5000 requests per second.*

*Which of the following represents the MOST resource-efficient and cost-optimal way of resolving this issue?*

**Change the application architecture to create customer-specific custom prefixes within the single bucket and then upload the daily files into those prefixed locations**

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Your applications can easily achieve thousands of transactions per second in request performance when uploading and retrieving storage from Amazon S3. Amazon S3 automatically scales to high request rates. For example, your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in a bucket.

There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by parallelizing reads. For example, if you create 10 prefixes in an Amazon S3 bucket to parallelize reads, you could scale your read performance to 55,000 read requests per second. Please see this example for more clarity on prefixes: if you have a file f1 stored in an S3 object path like so s3://your\_bucket\_name/folder1/sub\_folder\_1/f1, then /folder1/sub\_folder\_1/ becomes the prefix for file f1.

Some data lake applications on Amazon S3 scan millions or billions of objects for queries that run over petabytes of data. These data lake applications achieve single-instance transfer rates that maximize the network interface used for their Amazon EC2 instance, which can be up to 100 Gb/s on a single instance. These applications then aggregate throughput across multiple instances to get multiple terabits per second. Therefore creating customer-specific custom prefixes within the single bucket and then uploading the daily files into those prefixed locations is the BEST solution for the given constraints.

***Question\***

*A financial services firm is modernizing its message queuing system by migrating from self-managed message-oriented middleware systems to Amazon SQS. The firm is using SQS to migrate several applications to the cloud to ensure high availability and cost efficiency while simplifying administrative complexity and overhead. The data analytics team at the firm expects a peak rate of about 2,400 transactions per second to be processed via SQS. The messages must be processed in the order they are received.*

*Which of the following options can be used to implement this system most cost-effectively?*

**Use Amazon SQS FIFO queue in batch mode of 8 transactions per operation to process the transactions at the peak rate**

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues - Standard queues vs FIFO queues.

For FIFO queues, the order in which messages are sent and received is strictly preserved (i.e. First-In-First-Out). On the other hand, the standard SQS queues offer best-effort ordering. This means that occasionally, messages might be delivered in an order different from which they were sent.

By default, FIFO queues support up to 300 transactions (API calls) per second (300 send, receive, or delete operations per second). When you batch 10 transactions per operation (maximum), FIFO queues can support up to 3,000 (300*10) transactions per second. Therefore, you need to process 8 transactions per operation so that the FIFO queue can support up to 2,400 (300*8) transactions per second, which satisfies the peak rate constraint.

***Question***

*The data analytics team at a retail company has set up a workflow to ingest the clickstream data into the raw zone of the S3 data lake. The team wants to run some SQL-based data sanity checks on the raw zone of the data lake.*

*What AWS services would you recommend for this use-case such that the solution is cost-effective and easy to maintain?*

**Use Athena to run SQL based analytics against S3 data**

Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to set up or manage, and customers pay only for the queries they run. You can use Athena to process logs, perform ad-hoc analysis, and run interactive queries.

***Question***

A company uses Amazon Simple Storage Service (Amazon S3) as a storage service for storing various media files, log files, audit files, etc. The company has hired you as an AWS Certified Data Analytics Specialist to also configure Amazon EMR to use Amazon S3 as the Hadoop storage layer instead of the Hadoop Distributed File System (HDFS).

How will you configure this requirement?

**You can't configure Amazon EMR to use Amazon S3 instead of HDFS for the Hadoop storage layer**

You can't configure Amazon EMR to use Amazon S3 instead of HDFS for the Hadoop storage layer. HDFS and the EMR File System (EMRFS), which uses Amazon S3, are both compatible with Amazon EMR, but they're not interchangeable. HDFS is an implementation of the Hadoop FileSystem API, which models POSIX file system behavior. EMRFS is an object store, not a file system.

The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like data encryption.

You can now use [**Amazon Athena Workgroups**](https://docs.aws.amazon.com/athena/latest/ug/manage-queries-control-costs-with-workgroups.html) - A new resource type that can be used to separate query execution and query history between Users, Teams, or Applications running under the same AWS account. Because Workgroups act as resources, you can use [resource-based policies](https://docs.aws.amazon.com/athena/latest/ug/workgroups-iam-policy.html) to control access to a Workgroup. For example, if you have two teams or different applications using Athena, you can limit access by assigning them to different Workgroups. You can also separate ad hoc usage from scheduled reports by assigning them to different Workgroups. Queries running in one Workgroup are not visible to users or applications running in a different Workgroup. You can also temporarily disable Workgroups, preventing users from running queries, or permanently delete them. [Fine-Grained Access Control](https://docs.aws.amazon.com/athena/latest/ug/fine-grained-access-to-glue-resources.html) for Tables and Databases defined in the Glue Data Catalog can further restrict access to specific databases and tables.