



**Amazon**

## **Exam Questions AWS-Certified-Data-Engineer-Associate**

AWS Certified Data Engineer - Associate (DEA-C01)

### NEW QUESTION 1

A data engineer needs to join data from multiple sources to perform a one-time analysis job. The data is stored in Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3.

Which solution will meet this requirement MOST cost-effectively?

- A. Use an Amazon EMR provisioned cluster to read from all source
- B. Use Apache Spark to join the data and perform the analysis.
- C. Copy the data from DynamoDB, Amazon RDS, and Amazon Redshift into Amazon S3. Run Amazon Athena queries directly on the S3 files.
- D. Use Amazon Athena Federated Query to join the data from all data sources.
- E. Use Redshift Spectrum to query data from DynamoDB, Amazon RDS, and Amazon S3 directly from Redshift.

**Answer: C**

#### Explanation:

Amazon Athena Federated Query is a feature that allows you to query data from multiple sources using standard SQL. You can use Athena Federated Query to join data from Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3, as well as other data sources such as MongoDB, Apache HBase, and Apache Kafka<sup>1</sup>. Athena Federated Query is a serverless and interactive service, meaning you do not need to provision or manage any infrastructure, and you only pay for the amount of data scanned by your queries. Athena Federated Query is the most cost-effective solution for performing a one-time analysis job on data from multiple sources, as it eliminates the need to copy or move data, and allows you to query data directly from the source.

The other options are not as cost-effective as Athena Federated Query, as they involve additional steps or costs. Option A requires you to provision and pay for an Amazon EMR cluster, which can be expensive and time-consuming for a one-time job. Option B requires you to copy or move data from DynamoDB, RDS, and Redshift to S3, which can incur additional costs for data transfer and storage, and also introduce latency and complexity. Option D requires you to have an existing Redshift cluster, which can be costly and may not be necessary for a one-time job. Option E also does not support querying data from RDS directly, so you would need to use Redshift Federated Query to access RDS data, which adds another layer of complexity<sup>2</sup>. References:

- ? Amazon Athena Federated Query
- ? Redshift Spectrum vs Federated Query

### NEW QUESTION 2

A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account.

Which solution will meet these requirements?

- A. Create an S3 bucket for each use cas
- B. Create an S3 bucket policy that grants permissions to appropriate individual IAM user
- C. Apply the S3 bucket policy to the S3 bucket.
- D. Create an Athena workgroup for each use cas
- E. Apply tags to the workgrou
- F. Create an IAM policy that uses the tags to apply appropriate permissions to the workgroup.
- G. Create an IAM role for each use cas
- H. Assign appropriate permissions to the role for each use cas
- I. Associate the role with Athena.
- J. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use cas
- K. Apply the resource policy to the specific tables that Athena uses.

**Answer: B**

#### Explanation:

Athena workgroups are a way to isolate query execution and query history among users, teams, and applications that share the same AWS account. By creating a workgroup for each use case, the company can control the access and actions on the workgroup resource using resource-level IAM permissions or identity-based IAM policies. The company can also use tags to organize and identify the workgroups, and use them as conditions in the IAM policies to grant or deny permissions to the workgroup. This solution meets the requirements of separating query processes and access to query history among users, teams, and applications that are in the same AWS account. References:

- ? Athena Workgroups
- ? IAM policies for accessing workgroups
- ? Workgroup example policies

### NEW QUESTION 3

A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisioned capacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use AWS Glue to crawl the data source
- B. Store metadata in the AWS Glue Data Catalog
- C. Use Amazon Athena to query the data
- D. Use SQL for structured data source
- E. Use PartiQL for data that is stored in JSON format.
- F. Use AWS Glue to crawl the data source
- G. Store metadata in the AWS Glue Data Catalog
- H. Use Redshift Spectrum to query the data
- I. Use SQL for structured data source
- J. Use PartiQL for data that is stored in JSON format.
- K. Use AWS Glue to crawl the data source
- L. Store metadata in the AWS Glue Data Catalog
- M. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format
- N. Store the transformed data in an S3 bucket
- O. Use Amazon Athena to query the original and transformed data from the S3 bucket.
- P. Use AWS Lake Formation to create a data lake
- Q. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format

- R. Store the transformed data in an S3 bucket
- S. Use Amazon Athena or Redshift Spectrum to query the data.

**Answer:** A

**Explanation:**

The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format.

AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores<sup>1</sup>. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog<sup>2</sup>. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components<sup>3</sup>. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python<sup>4</sup>. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.

Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs<sup>1</sup>. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.

Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster<sup>5</sup>. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.

Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes<sup>6</sup>. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data.

Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning<sup>7</sup>. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B. References:

- ? What is Amazon Athena?
- ? Data Catalog and crawlers in AWS Glue
- ? AWS Glue Data Catalog
- ? Columnar Storage Formats
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? AWS Glue Schema Registry
- ? What is AWS Glue?
- ? Amazon Redshift Serverless
- ? Amazon Redshift provisioned clusters
- ? [Querying external data using Amazon Redshift Spectrum]
- ? [Using stored procedures in Amazon Redshift]
- ? [What is AWS Lambda?]
- ? [PartiQL for Amazon Athena]
- ? [Federated queries in Amazon Athena]
- ? [Amazon Athena pricing]
- ? [Top 10 performance tuning tips for Amazon Athena]
- ? [AWS Glue ETL jobs]
- ? [AWS Lake Formation jobs]

**NEW QUESTION 4**

A company maintains an Amazon Redshift provisioned cluster that the company uses for extract, transform, and load (ETL) operations to support critical analysis tasks. A sales team within the company maintains a Redshift cluster that the sales team uses for business intelligence (BI) tasks.

The sales team recently requested access to the data that is in the ETL Redshift cluster so the team can perform weekly summary analysis tasks. The sales team needs to join data from the ETL cluster with data that is in the sales team's BI cluster.

The company needs a solution that will share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution must minimize usage of the computing resources of the ETL cluster.

Which solution will meet these requirements?

- A. Set up the sales team BI cluster as a consumer of the ETL cluster by using Redshift data sharing.
- B. Create materialized views based on the sales team's requirement
- C. Grant the sales team direct access to the ETL cluster.
- D. Create database views based on the sales team's requirement
- E. Grant the sales team direct access to the ETL cluster.

- F. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every wee
- G. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

**Answer:** A

**Explanation:**

Redshift data sharing is a feature that enables you to share live data across different Redshift clusters without the need to copy or move data. Data sharing provides secure and governed access to data, while preserving the performance and concurrency benefits of Redshift. By setting up the sales team BI cluster as a consumer of the ETL cluster, the company can share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution also minimizes the usage of the computing resources of the ETL cluster, as the data sharing does not consume any storage space or compute resources from the producer cluster. The other options are either not feasible or not efficient. Creating materialized views or database views would require the sales team to have direct access to the ETL cluster, which could interfere with the critical analysis tasks. Unloading a copy of the data from the ETL cluster to an Amazon S3 bucket every week would introduce additional latency and cost, as well as create data inconsistency issues. References:

? Sharing data across Amazon Redshift clusters

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

**NEW QUESTION 5**

A company uses AWS Step Functions to orchestrate a data pipeline. The pipeline consists of Amazon EMR jobs that ingest data from data sources and store the data in an Amazon S3 bucket. The pipeline also includes EMR jobs that load the data to Amazon Redshift.

The company's cloud infrastructure team manually built a Step Functions state machine. The cloud infrastructure team launched an EMR cluster into a VPC to support the EMR jobs. However, the deployed Step Functions state machine is not able to run the EMR jobs.

Which combination of steps should the company take to identify the reason the Step Functions state machine is not able to run the EMR jobs? (Choose two.)

- A. Use AWS CloudFormation to automate the Step Functions state machine deployment
- B. Create a step to pause the state machine during the EMR jobs that fail
- C. Configure the step to wait for a human user to send approval through an email message
- D. Include details of the EMR task in the email message for further analysis.
- E. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR job
- F. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs use
- G. Use Access Analyzer for S3 to check the S3 access properties.
- H. Check for entries in Amazon CloudWatch for the newly created EMR cluster
- I. Change the AWS Step Functions state machine code to use Amazon EMR on EKS
- J. Change the IAM access policies and the security group configuration for the Step Functions state machine code to reflect inclusion of Amazon Elastic Kubernetes Service (Amazon EKS).
- K. Query the flow logs for the VPC
- L. Determine whether the traffic that originates from the EMR cluster can successfully reach the data provider
- M. Determine whether any security group that might be attached to the Amazon EMR cluster allows connections to the data source servers on the informed ports.
- N. Check the retry scenarios that the company configured for the EMR job
- O. Increase the number of seconds in the interval between each EMR task
- P. Validate that each fallback state has the appropriate catch for each decision state
- Q. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

**Answer:** BD

**Explanation:**

To identify the reason why the Step Functions state machine is not able to run the EMR jobs, the company should take the following steps:

? Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. The state machine code should have an IAM role that allows it to invoke the EMR APIs, such as RunJobFlow, AddJobFlowSteps, and DescribeStep. The state machine code should also have IAM permissions to access the Amazon S3 buckets that the EMR jobs use as input and output locations. The company can use Access Analyzer for S3 to check the access policies and permissions of the S3 buckets<sup>12</sup>. Therefore, option B is correct.

? Query the flow logs for the VPC. The flow logs can provide information about the network traffic to and from the EMR cluster that is launched in the VPC. The company can use the flow logs to determine whether the traffic that originates from the EMR cluster can successfully reach the data providers, such as Amazon RDS, Amazon Redshift, or other external sources. The company can also determine whether any security group that might be attached to the EMR cluster allows connections to the data source servers on the informed ports. The company can use Amazon VPC Flow Logs or Amazon CloudWatch Logs Insights to query the flow logs<sup>3</sup>. Therefore, option D is correct.

Option A is incorrect because it suggests using AWS CloudFormation to automate the Step Functions state machine deployment. While this is a good practice to ensure consistency and repeatability of the deployment, it does not help to identify the reason why the state machine is not able to run the EMR jobs. Moreover, creating a step to pause the state machine during the EMR jobs that fail and wait for a human user to send approval through an email message is not a reliable way to troubleshoot the issue. The company should use the Step Functions console or API to monitor the execution history and status of the state machine, and use Amazon CloudWatch to view the logs and metrics of the EMR jobs. Option C is incorrect because it suggests changing the AWS Step Functions state machine code to use Amazon EMR on EKS. Amazon EMR on EKS is a service that allows you to run EMR jobs on Amazon Elastic Kubernetes Service (Amazon EKS) clusters. While this service has some benefits, such as lower cost and faster execution time, it does not support all the features and integrations that EMR on EC2 does, such as EMR Notebooks, EMR Studio, and EMRFS. Therefore, changing the state machine code to use EMR on EKS may not be compatible with the existing data pipeline and may introduce new issues. Option E is incorrect because it suggests checking the retry scenarios that the company configured for the EMR jobs. While this is a good practice to handle transient failures and errors, it does not help to identify the root cause of why the state machine is not able to run the EMR jobs. Moreover, increasing the number of seconds in the interval between each EMR task may not improve the success rate of the jobs, and may increase the execution time and cost of the state machine. Configuring an Amazon SNS topic to store the error messages may help to notify the company of any failures, but it does not provide enough information to troubleshoot the issue.

References:

? 1: Manage an Amazon EMR Job - AWS Step Functions

? 2: Access Analyzer for S3 - Amazon Simple Storage Service

? 3: Working with Amazon EMR and VPC Flow Logs - Amazon EMR

? [4]: Analyzing VPC Flow Logs with Amazon CloudWatch Logs Insights - Amazon Virtual Private Cloud

? [5]: Monitor AWS Step Functions - AWS Step Functions

? [6]: Monitor Amazon EMR clusters - Amazon EMR

? [7]: Amazon EMR on Amazon EKS - Amazon EMR

**NEW QUESTION 6**

A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models.

The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.

Which change should the engineer make to gain access to SageMaker Studio?

- A. Add the AWSServiceRoleForGlue managed policy to the data engineer's IAM user.
- B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.
- C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.
- D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

**Answer: B**

**Explanation:**

This solution meets the requirement of gaining access to SageMaker Studio to use AWS Glue interactive sessions. AWS Glue interactive sessions are a way to use AWS Glue DataBrew and AWS Glue Data Catalog from within SageMaker Studio. To use AWS Glue interactive sessions, the data engineer's IAM user needs to have permissions to assume the AWS Glue service role and the SageMaker execution role. By adding a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy, the data engineer can grant these permissions and avoid the access denied error. The other options are not sufficient or necessary to resolve the error. References:

- ? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions
- ? Troubleshoot Errors - Amazon SageMaker
- ? AccessDeniedException on sagemaker:CreateDomain in AWS SageMaker Studio, despite having SageMakerFullAccess

**NEW QUESTION 7**

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage. Which solution will meet these requirements with the LEAST operational overhead?

- A. Create snapshots of the gp2 volume
- B. Create new gp3 volumes from the snapshot
- C. Attach the new gp3 volumes to the EC2 instances.
- D. Create new gp3 volume
- E. Gradually transfer the data to the new gp3 volume
- F. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.
- G. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.
- H. Use AWS DataSync to create new gp3 volume
- I. Transfer the data from the original gp2 volumes to the new gp3 volumes.

**Answer: C**

**Explanation:**

Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. References:

- ? Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs (Section: How to migrate from gp2 to gp3)
- ? Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs (Section: How to Switch from GP2 Volumes to GP3 Volumes)
- ? Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud (Section: Modifying the volume type)

**NEW QUESTION 8**

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3. The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata. Which solution will meet these requirements with the LEAST operational overhead?

- A. Use Amazon Aurora as the data catalog
- B. Create AWS Lambda functions that will connect to the data catalog
- C. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog
- D. Schedule the Lambda functions to run periodically.
- E. Use the AWS Glue Data Catalog as the central metadata repository
- F. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata change
- G. Schedule the crawlers to run periodically to update the metadata catalog.
- H. Use Amazon DynamoDB as the data catalog
- I. Create AWS Lambda functions that will connect to the data catalog
- J. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog
- K. Schedule the Lambda functions to run periodically.
- L. Use the AWS Glue Data Catalog as the central metadata repository
- M. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog
- N. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

**Answer: B**

**Explanation:**

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions<sup>12</sup>. The other options are not optimal for the following reasons:

- ? A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain

AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

? D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog.

References:

? 1: AWS Glue Data Catalog

? 2: AWS Glue Crawlers

? : Amazon Aurora

? : AWS Lambda

? : Amazon DynamoDB

## NEW QUESTION 9

A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT data. The company's analytics department will use the data catalog to index the data. Which solution will meet these requirements MOST cost-effectively?

- A. Create an AWS Glue Data Catalog
- B. Configure an AWS Glue Schema Registry
- C. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.
- D. Create an Amazon Redshift provisioned cluster
- E. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.
- F. Create an Amazon Athena workgroup
- G. Explore the data that is in Amazon S3 by using Apache Spark through Athena
- H. Provide the Athena workgroup schema and tables to the analytics department.
- I. Create an AWS Glue Data Catalog
- J. Configure an AWS Glue Schema Registry
- K. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API
- L. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

**Answer: C**

### Explanation:

The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost-effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python<sup>1</sup>. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers<sup>2</sup>. You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore<sup>3</sup>. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications<sup>4</sup>. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis<sup>5</sup>.

Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle<sup>1</sup>. You also save on the operational overhead and complexity of managing data warehouse infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently.

Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components<sup>6</sup>. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog<sup>7</sup>. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores<sup>8</sup>. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure<sup>9</sup>. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made<sup>6,7</sup>. AWS Glue charges you based on the compute time and the data processed by your ETL jobs<sup>8</sup>. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads<sup>9</sup>. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark.

Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity<sup>10</sup>. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL<sup>11</sup>. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command<sup>12</sup>. While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster<sup>10</sup>. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries<sup>11</sup>. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark. Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless

compute service that lets you run code without provisioning or managing servers<sup>13</sup>. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made<sup>67</sup>. AWS Lambda charges you based on the number of requests and the duration of your functions<sup>13</sup>. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads<sup>9</sup>. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark. References:

- ? What is Amazon Athena?
- ? Apache Spark on Amazon Athena
- ? Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs
- ? Managing Athena workgroups
- ? Using Amazon QuickSight to visualize data in Amazon Athena
- ? AWS Glue Data Catalog
- ? AWS Glue Schema Registry
- ? What is AWS Glue?
- ? Amazon Redshift Serverless
- ? Amazon Redshift provisioned clusters
- ? Querying external data using Amazon Redshift Spectrum
- ? Using stored procedures in Amazon Redshift
- ? What is AWS Lambda?
- ? [Creating and using AWS Lambda UDFs]
- ? [Using the Amazon Redshift Data API]
- ? [What is AWS Step Functions?]
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

#### NEW QUESTION 10

A company is migrating its database servers from Amazon EC2 instances that run Microsoft SQL Server to Amazon RDS for Microsoft SQL Server DB instances. The company's analytics team must export large data elements every day until the migration is complete. The data elements are the result of SQL joins across multiple tables. The data must be in Apache Parquet format. The analytics team must store the data in Amazon S3. Which solution will meet these requirements in the MOST operationally efficient way?

- A. Create a view in the EC2 instance-based SQL Server databases that contains the required data element
- B. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucket
- C. Schedule the AWS Glue job to run every day.
- D. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server database
- E. Configure the query to direct the output .csv objects to an S3 bucket
- F. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.
- G. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data element
- H. Create and run an AWS Glue crawler to read the view
- I. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucket
- J. Schedule the AWS Glue job to run every day.
- K. Create an AWS Lambda function that queries the EC2 instance-based databases by using Java Database Connectivity (JDBC). Configure the Lambda function to retrieve the required data, transform the data into Parquet format, and transfer the data into an S3 bucket
- L. Use Amazon EventBridge to schedule the Lambda function to run every day.

**Answer:** A

#### Explanation:

Option A is the most operationally efficient way to meet the requirements because it minimizes the number of steps and services involved in the data export process. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including Amazon S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. By creating a view in the SQL Server databases that contains the required data elements, the AWS Glue job can select the data directly from the view without having to perform any joins or transformations on the source data. The AWS Glue job can then transfer the data in Parquet format to an S3 bucket and run on a daily schedule.

Option B is not operationally efficient because it involves multiple steps and services to export the data. SQL Server Agent is a tool that can run scheduled tasks on SQL Server databases, such as executing SQL queries. However, SQL Server Agent cannot directly export data to S3, so the query output must be saved as .csv objects on the EC2 instance. Then, an S3 event must be configured to trigger an AWS Lambda function that can transform the .csv objects to Parquet format and upload them to S3. This option adds complexity and latency to the data export process and requires additional resources and configuration.

Option C is not operationally efficient because it introduces an unnecessary step of running an AWS Glue crawler to read the view. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. However, in this scenario, the schema and format of the data elements are already known and fixed, so there is no need to run a crawler to discover them. The AWS Glue job can directly select the data from the view without using the Data Catalog. Running a crawler adds extra time and cost to the data export process.

Option D is not operationally efficient because it requires custom code and configuration to query the databases and transform the data. An AWS Lambda function is a service that can run code in response to events or triggers, such as Amazon EventBridge. Amazon EventBridge is a service that can connect applications and services with event sources, such as schedules, and route them to targets, such as Lambda functions. However, in this scenario, using a Lambda function to query the databases and transform the data is not the best option because it requires writing and maintaining code that uses JDBC to connect to the SQL Server databases, retrieve the required data, convert the data to Parquet format, and transfer the data to S3. This option also has limitations on the execution time, memory, and concurrency of the Lambda function, which may affect the performance and reliability of the data export process.

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? AWS Glue Documentation
- ? Working with Views in AWS Glue
- ? Converting to Columnar Formats

#### NEW QUESTION 10

A data engineer must use AWS services to ingest a dataset into an Amazon S3 data lake. The data engineer profiles the dataset and discovers that the dataset contains personally identifiable information (PII). The data engineer must implement a solution to profile the dataset and obfuscate the PII.

Which solution will meet this requirement with the LEAST operational effort?

- A. Use an Amazon Kinesis Data Firehose delivery stream to process the dataset
- B. Create an AWS Lambda transform function to identify the PII
- C. Use an AWS SDK to obfuscate the PII
- D. Set the S3 data lake as the target for the delivery stream.
- E. Use the Detect PII transform in AWS Glue Studio to identify the PII
- F. Obfuscate the PII
- G. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
- H. Use the Detect PII transform in AWS Glue Studio to identify the PII
- I. Create a rule in AWS Glue Data Quality to obfuscate the PII
- J. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
- K. Ingest the dataset into Amazon DynamoDB
- L. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data
- M. Use the same Lambda function to ingest the data into the S3 data lake.

**Answer: C**

**Explanation:**

AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue Studio is a graphical interface that allows you to easily author, run, and monitor AWS Glue ETL jobs. AWS Glue Data Quality is a feature that enables you to validate, cleanse, and enrich your data using predefined or custom rules. AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows.

Using the Detect PII transform in AWS Glue Studio, you can automatically identify and label the PII in your dataset, such as names, addresses, phone numbers, email addresses, etc. You can then create a rule in AWS Glue Data Quality to obfuscate the PII, such as masking, hashing, or replacing the values with dummy data. You can also use other rules to validate and cleanse your data, such as checking for null values, duplicates, outliers, etc. You can then use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake. You can use AWS Glue DataBrew to visually explore and transform the data, AWS Glue crawlers to discover and catalog the data, and AWS Glue jobs to load the data into the S3 data lake.

This solution will meet the requirement with the least operational effort, as it leverages the serverless and managed capabilities of AWS Glue, AWS Glue Studio, AWS Glue Data Quality, and AWS Step Functions. You do not need to write any code to identify or obfuscate the PII, as you can use the built-in transforms and rules in AWS Glue Studio and AWS Glue Data Quality. You also do not need to provision or manage any servers or clusters, as AWS Glue and AWS Step Functions scale automatically based on the demand. The other options are not as efficient as using the Detect PII transform in AWS Glue Studio, creating a rule in AWS Glue Data Quality, and using an AWS Step Functions state machine. Using an Amazon Kinesis Data Firehose delivery stream to process the dataset, creating an AWS Lambda transform function to identify the PII, using an AWS SDK to obfuscate the PII, and setting the S3 data lake as the target for the delivery stream will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. Using the Detect PII transform in AWS Glue Studio to identify the PII, obfuscating the PII, and using an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake will not be as effective as creating a rule in AWS Glue Data Quality to obfuscate the PII, as you will need to manually obfuscate the PII after identifying it, which can be error-prone and time-consuming. Ingesting the dataset into Amazon DynamoDB, creating an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data, and using the same Lambda function to ingest the data into the S3 data lake will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. You will also incur additional costs and complexity by using DynamoDB as an intermediate data store, which may not be necessary for your use case. References:

? AWS Glue

? AWS Glue Studio

? AWS Glue Data Quality

? [AWS Step Functions]

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

**NEW QUESTION 14**

A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations.

Which combination of AWS services will implement a data mesh? (Choose two.)

- A. Use Amazon Aurora for data storage
- B. Use an Amazon Redshift provisioned cluster for data analysis.
- C. Use Amazon S3 for data storage
- D. Use Amazon Athena for data analysis.
- E. Use AWS Glue DataBrew for centralized data governance and access control.
- F. Use Amazon RDS for data storage
- G. Use Amazon EMR for data analysis.
- H. Use AWS Lake Formation for centralized data governance and access control.

**Answer: BE**

**Explanation:**

A data mesh is an architectural framework that organizes data into domains and treats data as products that are owned and offered for consumption by different teams<sup>1</sup>. A data mesh requires a centralized layer for data governance and access control, as well as a distributed layer for data storage and analysis. AWS Glue can provide data catalogs and ETL operations for the data mesh, but it cannot provide data governance and access control by itself<sup>2</sup>. Therefore, the company needs to use another AWS service for this purpose. AWS Lake Formation is a service that allows you to create, secure, and manage data lakes on AWS<sup>3</sup>. It integrates with AWS Glue and other AWS services to provide centralized data governance and access control for the data mesh. Therefore, option E is correct. For data storage and analysis, the company can choose from different AWS services depending on their needs and preferences. However, one of the benefits of a data mesh is that it enables data to be stored and processed in a decoupled and scalable way<sup>1</sup>. Therefore, using serverless or managed services that can handle large volumes and varieties of data is preferable. Amazon S3 is a highly scalable, durable, and secure object storage service that can store any type of data. Amazon Athena is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Therefore, option B is a good choice for data storage and analysis in a data mesh. Option A, C, and D are not optimal because they either use relational databases that are not suitable for storing diverse and unstructured data, or they require more management and provisioning than serverless services. References:

? 1: What is a Data Mesh? - Data Mesh Architecture Explained - AWS

? 2: AWS Glue - Developer Guide

? 3: AWS Lake Formation - Features

? [4]: Design a data mesh architecture using AWS Lake Formation and AWS Glue

? [5]: Amazon S3 - Features

? [6]: Amazon Athena - Features

#### NEW QUESTION 19

A data engineer needs to create an AWS Lambda function that converts the format of data from .csv to Apache Parquet. The Lambda function must run only if a user uploads a .csv file to an Amazon S3 bucket.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Create an S3 event notification that has an event type of s3:ObjectCreated:\*. Use a filter rule to generate notifications only when the suffix includes .cs
- B. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- C. Create an S3 event notification that has an event type of s3:ObjectTagging:\* for objects that have a tag set to .cs
- D. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- E. Create an S3 event notification that has an event type of s3:\*. Use a filter rule to generate notifications only when the suffix includes .cs
- F. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
- G. Create an S3 event notification that has an event type of s3:ObjectCreated:\*. Use a filter rule to generate notifications only when the suffix includes .cs
- H. Set an Amazon Simple Notification Service (Amazon SNS) topic as the destination for the event notification.
- I. Subscribe the Lambda function to the SNS topic.

**Answer:** A

#### Explanation:

Option A is the correct answer because it meets the requirements with the least operational overhead. Creating an S3 event notification that has an event type of s3:ObjectCreated:\* will trigger the Lambda function whenever a new object is created in the S3 bucket. Using a filter rule to generate notifications only when the suffix includes .csv will ensure that the Lambda function only runs for .csv files. Setting the ARN of the Lambda function as the destination for the event notification will directly invoke the Lambda function without any additional steps.

Option B is incorrect because it requires the user to tag the objects with .csv, which adds an extra step and increases the operational overhead.

Option C is incorrect because it uses an event type of s3:\*, which will trigger the Lambda function for any S3 event, not just object creation. This could result in unnecessary invocations and increased costs.

Option D is incorrect because it involves creating and subscribing to an SNS topic, which adds an extra layer of complexity and operational overhead.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.2: S3 Event Notifications and Lambda Functions, Pages 67-69

? Building Batch Data Analytics Solutions on AWS, Module 4: Data Transformation, Lesson 4.2: AWS Lambda, Pages 4-8

? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon S3, Pages 1-5

#### NEW QUESTION 20

A company is planning to migrate on-premises Apache Hadoop clusters to Amazon EMR. The company also needs to migrate a data catalog into a persistent storage solution.

The company currently stores the data catalog in an on-premises Apache Hive metastore on the Hadoop clusters. The company requires a serverless solution to migrate the data catalog.

Which solution will meet these requirements MOST cost-effectively?

- A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.
- B. Configure a Hive metastore in Amazon EMR
- C. Migrate the existing on-premises Hive metastore into Amazon EMR
- D. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.
- E. Configure an external Hive metastore in Amazon EMR
- F. Migrate the existing on-premises Hive metastore into Amazon EMR
- G. Use Amazon Aurora MySQL to store the company's data catalog.
- H. Configure a new Hive metastore in Amazon EMR
- I. Migrate the existing on-premises Hive metastore into Amazon EMR
- J. Use the new metastore as the company's data catalog.

**Answer:** A

#### Explanation:

AWS Database Migration Service (AWS DMS) is a service that helps you migrate databases to AWS quickly and securely. You can use AWS DMS to migrate the Hive metastore from the on-premises Hadoop clusters into Amazon S3, which is a highly scalable, durable, and cost-effective object storage service. AWS Glue Data Catalog is a serverless, managed service that acts as a central metadata repository for your data assets. You can use AWS Glue Data Catalog to scan the Amazon S3 bucket that contains the migrated Hive metastore and create a data catalog that is compatible with Apache Hive and other AWS services. This solution meets the requirements of migrating the data catalog into a persistent storage solution and using a serverless solution. This solution is also the most cost-effective, as it does not incur any additional charges for running Amazon EMR or Amazon Aurora MySQL clusters. The other options are either not feasible or not optimal. Configuring a Hive metastore in Amazon EMR (option B) or an external Hive metastore in Amazon EMR (option C) would require running and maintaining Amazon EMR clusters, which would incur additional costs and complexity. Using Amazon Aurora MySQL to store the company's data catalog (option G) would also incur additional costs and complexity, as well as introduce compatibility issues with Apache Hive. Configuring a new Hive metastore in Amazon EMR (option D) would not migrate the existing data catalog, but create a new one, which would result in data loss and inconsistency. References:

? Using AWS Database Migration Service

? Populating the AWS Glue Data Catalog

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

#### NEW QUESTION 24

A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size. A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.

Which solution will meet this requirement with the LEAST operational effort?

- A. Create and run an Apache Spark job in an AWS Glue notebook
- B. Configure the job to read the S3 file and calculate the number of distinct customers.
- C. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file
- D. Run SQL queries from Amazon Athena to calculate the number of distinct customers.
- E. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.

F. Use AWS Glue DataBrew to create a recipe that uses the COUNT\_DISTINCT aggregate function to calculate the number of distinct customers.

**Answer:** D

**Explanation:**

AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets<sup>1</sup>.

The solution that meets the requirement with the least operational effort is to use AWS Glue DataBrew to create a recipe that uses the COUNT\_DISTINCT aggregate function to calculate the number of distinct customers. This solution has the following advantages:

- ? It does not require you to write any code, as DataBrew provides a graphical user interface that lets you explore, transform, and visualize your data. You can use DataBrew to concatenate the columns that contain customer first names and last names, and then use the COUNT\_DISTINCT aggregate function to count the number of unique values in the resulting column<sup>2</sup>.

- ? It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes<sup>1</sup>.

- ? It does not require you to create or update any AWS Glue Data Catalog entries, as DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources<sup>3</sup>.

Option A is incorrect because it suggests creating and running an Apache Spark job in an AWS Glue notebook. This solution has the following disadvantages:

- ? It requires you to write code, as AWS Glue notebooks are interactive development environments that allow you to write, test, and debug Apache Spark code using Python or Scala. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

- ? It requires you to provision and manage a development endpoint, which is a serverless Apache Spark environment that you can connect to your notebook. You need to specify the type and number of workers for your development endpoint, and monitor its status and metrics.

- ? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

Option B is incorrect because it suggests creating an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file, and running SQL queries from Amazon Athena to calculate the number of distinct customers. This solution has the following disadvantages:

- ? It requires you to create and run a crawler, which is a program that connects to your data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the Data Catalog. You need to specify the data store, the IAM role, the schedule, and the output database for your crawler.

- ? It requires you to write SQL queries, as Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. You need to use Athena to concatenate the columns that contain customer first names and last names, and then use the COUNT(DISTINCT) aggregate function to count the number of unique values in the resulting column.

Option C is incorrect because it suggests creating and running an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers. This solution has the following disadvantages:

- ? It requires you to write code, as Amazon EMR Serverless is a service that allows you to run Apache Spark jobs on AWS without provisioning or managing any infrastructure. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

- ? It requires you to create and manage an Amazon EMR Serverless cluster, which is a fully managed and scalable Spark environment that runs on AWS Fargate. You need to specify the cluster name, the IAM role, the VPC, and the subnet for your cluster, and monitor its status and metrics.

- ? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

References:

- ? 1: AWS Glue DataBrew - Features
- ? 2: Working with recipes - AWS Glue DataBrew
- ? 3: Working with data sources and data targets - AWS Glue DataBrew
- ? [4]: AWS Glue notebooks - AWS Glue
- ? [5]: Development endpoints - AWS Glue
- ? [6]: Populating the AWS Glue Data Catalog - AWS Glue
- ? [7]: Crawlers - AWS Glue
- ? [8]: Amazon Athena - Features
- ? [9]: Amazon EMR Serverless - Features
- ? [10]: Creating an Amazon EMR Serverless cluster - Amazon EMR
- ? [11]: Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

**NEW QUESTION 26**

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.

The company wants to minimize the effort and time required to incorporate third-party datasets.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.
- B. Use API calls to access and integrate third-party datasets from AWS
- C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.
- D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

**Answer:** A

**Explanation:**

AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed<sup>12</sup>.

The other options are not optimal for the following reasons:

- ? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead<sup>3</sup>.

- ? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams.

Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data.

? D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data.

References:

- ? 1: AWS Data Exchange User Guide
- ? 2: AWS Data Exchange FAQs
- ? 3: AWS Glue Developer Guide
- ? : AWS CodeCommit User Guide
- ? : Amazon Kinesis Data Streams Developer Guide
- ? : Amazon Elastic Container Registry User Guide
- ? : Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

### NEW QUESTION 30

A company is migrating on-premises workloads to AWS. The company wants to reduce overall operational overhead. The company also wants to explore serverless options.

The company's current workloads use Apache Pig, Apache Oozie, Apache Spark, Apache Hbase, and Apache Flink. The on-premises workloads process petabytes of data in seconds. The company must maintain similar or better performance after the migration to AWS.

Which extract, transform, and load (ETL) service will meet these requirements?

- A. AWS Glue
- B. Amazon EMR
- C. AWS Lambda
- D. Amazon Redshift

**Answer:** A

#### Explanation:

AWS Glue is a fully managed serverless ETL service that can handle petabytes of data in seconds. AWS Glue can run Apache Spark and Apache Flink jobs without requiring any infrastructure provisioning or management. AWS Glue can also integrate with Apache Pig, Apache Oozie, and Apache Hbase using AWS Glue Data Catalog and AWS Glue workflows. AWS Glue can reduce the overall operational overhead by automating the data discovery, data preparation, and data loading processes. AWS Glue can also optimize the cost and performance of ETL jobs by using AWS Glue Job Bookmarking, AWS Glue Crawlers, and AWS Glue Schema Registry. References:

- ? AWS Glue
- ? AWS Glue Data Catalog
- ? AWS Glue Workflows
- ? [AWS Glue Job Bookmarking]
- ? [AWS Glue Crawlers]
- ? [AWS Glue Schema Registry]
- ? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

### NEW QUESTION 32

A data engineer uses Amazon Redshift to run resource-intensive analytics processes once every month. Every month, the data engineer creates a new Redshift provisioned cluster. The data engineer deletes the Redshift provisioned cluster after the analytics processes are complete every month. Before the data engineer deletes the cluster each month, the data engineer unloads backup data from the cluster to an Amazon S3 bucket.

The data engineer needs a solution to run the monthly analytics processes that does not require the data engineer to manage the infrastructure manually.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.
- B. Use Amazon Redshift Serverless to automatically process the analytics workload.
- C. Use the AWS CLI to automatically process the analytics workload.
- D. Use AWS CloudFormation templates to automatically process the analytics workload.

**Answer:** B

#### Explanation:

Amazon Redshift Serverless is a new feature of Amazon Redshift that enables you to run SQL queries on data in Amazon S3 without provisioning or managing any clusters. You can use Amazon Redshift Serverless to automatically process the analytics workload, as it scales up and down the compute resources based on the query demand, and charges you only for the resources consumed. This solution will meet the requirements with the least operational overhead, as it does not require the data engineer to create, delete, pause, or resume any Redshift clusters, or to manage any infrastructure manually. You can use the Amazon Redshift Data API to run queries from the AWS CLI, AWS SDK, or AWS Lambda functions<sup>12</sup>.

The other options are not optimal for the following reasons:

- ? A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month. This option is not recommended, as it would still require the data engineer to create and delete a new Redshift provisioned cluster every month, which can incur additional costs and time. Moreover, this option would require the data engineer to use Amazon Step Functions to orchestrate the workflow of pausing and resuming the cluster, which can add complexity and overhead.
- ? C. Use the AWS CLI to automatically process the analytics workload. This option is vague and does not specify how the AWS CLI is used to process the analytics workload. The AWS CLI can be used to run queries on data in Amazon S3 using Amazon Redshift Serverless, Amazon Athena, or Amazon EMR, but each of these services has different features and benefits. Moreover, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to provision and configure some resources, such as Amazon EMR clusters or Amazon Athena workgroups.
- ? D. Use AWS CloudFormation templates to automatically process the analytics workload. This option is also vague and does not specify how AWS CloudFormation templates are used to process the analytics workload. AWS CloudFormation is a service that lets you model and provision AWS resources using templates. You can use AWS CloudFormation templates to create and delete a Redshift provisioned cluster every month, or to create and configure other AWS resources, such as Amazon EMR, Amazon Athena, or Amazon Redshift Serverless. However, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to write and maintain the AWS CloudFormation templates, and to monitor the status and performance of the resources.

References:

- ? 1: Amazon Redshift Serverless
- ? 2: Amazon Redshift Data API
- ? : Amazon Step Functions
- ? : AWS CLI
- ? : AWS CloudFormation

### NEW QUESTION 33

A company is migrating a legacy application to an Amazon S3 based data lake. A data engineer reviewed data that is associated with the legacy application. The data engineer found that the legacy data contained some duplicate information. The data engineer must identify and remove duplicate information from the legacy application data. Which solution will meet these requirements with the LEAST operational overhead?

- A. Write a custom extract, transform, and load (ETL) job in Python
- B. Use the `DataFrame.drop_duplicates()` function by importing the Pandas library to perform data deduplication.
- C. Write an AWS Glue extract, transform, and load (ETL) job
- D. Use the FindMatches machine learning (ML) transform to transform the data to perform data deduplication.
- E. Write a custom extract, transform, and load (ETL) job in Python
- F. Import the Python dedupe library
- G. Use the dedupe library to perform data deduplication.
- H. Write an AWS Glue extract, transform, and load (ETL) job
- I. Import the Python dedupe library
- J. Use the dedupe library to perform data deduplication.

**Answer: B**

#### Explanation:

AWS Glue is a fully managed serverless ETL service that can handle data deduplication with minimal operational overhead. AWS Glue provides a built-in ML transform called FindMatches, which can automatically identify and group similar records in a dataset. FindMatches can also generate a primary key for each group of records and remove duplicates. FindMatches does not require any coding or prior ML experience, as it can learn from a sample of labeled data provided by the user. FindMatches can also scale to handle large datasets and optimize the cost and performance of the ETL job. References:

- ? AWS Glue
- ? FindMatches ML Transform
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

### NEW QUESTION 36

A company has five offices in different AWS Regions. Each office has its own human resources (HR) department that uses a unique IAM role. The company stores employee records in a data lake that is based on Amazon S3 storage. A data engineering team needs to limit access to the records. Each HR department should be able to access records for only employees who are within the HR department's Region. Which combination of steps should the data engineering team take to meet this requirement with the LEAST operational overhead? (Choose two.)

- A. Use data filters for each Region to register the S3 paths as data locations.
- B. Register the S3 path as an AWS Lake Formation location.
- C. Modify the IAM roles of the HR departments to add a data filter for each department's Region.
- D. Enable fine-grained access control in AWS Lake Formation
- E. Add a data filter for each Region.
- F. Create a separate S3 bucket for each Region
- G. Configure an IAM policy to allow S3 access
- H. Restrict access based on Region.

**Answer: BD**

#### Explanation:

AWS Lake Formation is a service that helps you build, secure, and manage data lakes on Amazon S3. You can use AWS Lake Formation to register the S3 path as a data lake location, and enable fine-grained access control to limit access to the records based on the HR department's Region. You can use data filters to specify which S3 prefixes or partitions each HR department can access, and grant permissions to the IAM roles of the HR departments accordingly. This solution will meet the requirement with the least operational overhead, as it simplifies the data lake management and security, and leverages the existing IAM roles of the HR departments.

The other options are not optimal for the following reasons:

- ? A. Use data filters for each Region to register the S3 paths as data locations. This option is not possible, as data filters are not used to register S3 paths as data locations, but to grant permissions to access specific S3 prefixes or partitions within a data location. Moreover, this option does not specify how to limit access to the records based on the HR department's Region.
- ? C. Modify the IAM roles of the HR departments to add a data filter for each department's Region. This option is not possible, as data filters are not added to IAM roles, but to permissions granted by AWS Lake Formation. Moreover, this option does not specify how to register the S3 path as a data lake location, or how to enable fine-grained access control in AWS Lake Formation.
- ? E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region. This option is not recommended, as it would require more operational overhead to create and manage multiple S3 buckets, and to configure and maintain IAM policies for each HR department. Moreover, this option does not leverage the benefits of AWS Lake Formation, such as data cataloging, data transformation, and data governance.

References:

- ? 1: AWS Lake Formation
- ? 2: AWS Lake Formation Permissions
- ? : AWS Identity and Access Management
- ? : Amazon S3

### NEW QUESTION 39

A data engineer runs Amazon Athena queries on data that is in an Amazon S3 bucket. The Athena queries use AWS Glue Data Catalog as a metadata table. The data engineer notices that the Athena query plans are experiencing a performance bottleneck. The data engineer determines that the cause of the performance bottleneck is the large number of partitions that are in the S3 bucket. The data engineer must resolve the performance bottleneck and reduce Athena query planning time. Which solutions will meet these requirements? (Choose two.)

- A. Create an AWS Glue partition index
- B. Enable partition filtering.
- C. Bucket the data based on a column that the data have in common in a WHERE clause of the user query
- D. Use Athena partition projection based on the S3 bucket prefix.
- E. Transform the data that is in the S3 bucket to Apache Parquet format.
- F. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

**Answer:** AC

**Explanation:**

The best solutions to resolve the performance bottleneck and reduce Athena query planning time are to create an AWS Glue partition index and enable partition filtering, and to use Athena partition projection based on the S3 bucket prefix.

AWS Glue partition indexes are a feature that allows you to speed up query processing of highly partitioned tables cataloged in AWS Glue Data Catalog. Partition indexes are available for queries in Amazon EMR, Amazon Redshift Spectrum, and AWS Glue ETL jobs. Partition indexes are sublists of partition keys defined in the table. When you create a partition index, you specify a list of partition keys that already exist on a given table. AWS Glue then creates an index for the specified keys and stores it in the Data Catalog. When you run a query that filters on the partition keys, AWS Glue uses the partition index to quickly identify the relevant partitions without scanning the entire table metadata. This reduces the query planning time and improves the query performance<sup>1</sup>.

Athena partition projection is a feature that allows you to speed up query processing of highly partitioned tables and automate partition management. In partition projection, Athena calculates partition values and locations using the table properties that you configure directly on your table in AWS Glue. The table properties allow Athena to 'project', or determine, the necessary partition information instead of having to do a more time-consuming metadata lookup in the AWS Glue Data Catalog. Because in-memory operations are often faster than remote operations, partition projection can reduce the runtime of queries against highly partitioned tables. Partition projection also automates partition management because it removes the need to manually create partitions in Athena, AWS Glue, or your external Hive metastore<sup>2</sup>.

Option B is not the best solution, as bucketing the data based on a column that the data have in common in a WHERE clause of the user query would not reduce the query planning time. Bucketing is a technique that divides data into buckets based on a hash function applied to a column. Bucketing can improve the performance of join queries by reducing the amount of data that needs to be shuffled between nodes. However, bucketing does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario<sup>3</sup>.

Option D is not the best solution, as transforming the data that is in the S3 bucket to Apache Parquet format would not reduce the query planning time. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, Parquet does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario<sup>4</sup>.

Option E is not the best solution, as using the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects would not reduce the query planning time. S3DistCP is a tool that can copy large amounts of data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCP can also aggregate smaller files into larger files to improve the performance of sequential access. However, S3DistCP does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario<sup>5</sup>. References:

- ? Improve query performance using AWS Glue partition indexes
- ? Partition projection with Amazon Athena
- ? Bucketing vs Partitioning
- ? Columnar Storage Formats
- ? S3DistCp
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

**NEW QUESTION 44**

A data engineer has a one-time task to read data from objects that are in Apache Parquet format in an Amazon S3 bucket. The data engineer needs to query only one column of the data.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Configure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe- Write a SQL SELECT statement on the dataframe to query the required column.
- B. Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.
- C. Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.
- D. Run an AWS Glue crawler on the S3 object
- E. Use a SQL SELECT statement in Amazon Athena to query the required column.

**Answer:** B

**Explanation:**

Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources.

Option A is not a good solution because it involves writing custom code and configuring an AWS Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration. Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the performance and reliability of the data retrieval process.

Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.

Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them. Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.

- References:
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
  - ? S3 Select and Glacier Select - Amazon Simple Storage Service
  - ? AWS Lambda - FAQs
  - ? What Is AWS Glue DataBrew? - AWS Glue DataBrew

- ? Populating the AWS Glue Data Catalog - AWS Glue
- ? What is Amazon Athena? - Amazon Athena

#### NEW QUESTION 48

A company stores details about transactions in an Amazon S3 bucket. The company wants to log all writes to the S3 bucket into another S3 bucket that is in the same AWS Region.

Which solution will meet this requirement with the LEAST operational effort?

- A. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function
- B. Program the Lambda function to write the event to Amazon Kinesis Data Firehose
- C. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.
- D. Create a trail of management events in AWS CloudTrail
- E. Configure the trail to receive data from the transactions S3 bucket
- F. Specify an empty prefix and write-only event
- G. Specify the logs S3 bucket as the destination bucket.
- H. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function
- I. Program the Lambda function to write the events to the logs S3 bucket.
- J. Create a trail of data events in AWS CloudTrail
- K. Configure the trail to receive data from the transactions S3 bucket
- L. Specify an empty prefix and write-only event
- M. Specify the logs S3 bucket as the destination bucket.

**Answer:** D

#### Explanation:

This solution meets the requirement of logging all writes to the S3 bucket into another S3 bucket with the least operational effort. AWS CloudTrail is a service that records the API calls made to AWS services, including Amazon S3. By creating a trail of data events, you can capture the details of the requests that are made to the transactions S3 bucket, such as the requester, the time, the IP address, and the response elements. By specifying an empty prefix and write-only events, you can filter the data events to only include the ones that write to the bucket. By specifying the logs S3 bucket as the destination bucket, you can store the CloudTrail logs in another S3 bucket that is in the same AWS Region. This solution does not require any additional coding or configuration, and it is more scalable and reliable than using S3 Event Notifications and Lambda functions. References:

- ? Logging Amazon S3 API calls using AWS CloudTrail
- ? Creating a trail for data events
- ? Enabling Amazon S3 server access logging

#### NEW QUESTION 52

A data engineer needs to securely transfer 5 TB of data from an on-premises data center to an Amazon S3 bucket. Approximately 5% of the data changes every day. Updates to the data need to be regularly proliferated to the S3 bucket. The data includes files that are in multiple formats. The data engineer needs to automate the transfer process and must schedule the process to run periodically.

Which AWS service should the data engineer use to transfer the data in the MOST operationally efficient way?

- A. AWS DataSync
- B. AWS Glue
- C. AWS Direct Connect
- D. Amazon S3 Transfer Acceleration

**Answer:** A

#### Explanation:

AWS DataSync is an online data movement and discovery service that simplifies and accelerates data migrations to AWS as well as moving data to and from on-premises storage, edge locations, other cloud providers, and AWS Storage services<sup>1</sup>. AWS DataSync can copy data to and from various sources and targets, including Amazon S3, and handle files in multiple formats. AWS DataSync also supports incremental transfers, meaning it can detect and copy only the changes to the data, reducing the amount of data transferred and improving the performance. AWS DataSync can automate and schedule the transfer process using triggers, and monitor the progress and status of the transfers using CloudWatch metrics and events<sup>1</sup>.

AWS DataSync is the most operationally efficient way to transfer the data in this scenario, as it meets all the requirements and offers a serverless and scalable solution. AWS Glue, AWS Direct Connect, and Amazon S3 Transfer Acceleration are not the best options for this scenario, as they have some limitations or drawbacks compared to AWS DataSync. AWS Glue is a serverless ETL service that can extract, transform, and load data from various sources to various targets, including Amazon S3<sup>2</sup>. However, AWS Glue is not designed for large-scale data transfers, as it has some quotas and limits on the number and size of files it can process<sup>3</sup>. AWS Glue also does not support incremental transfers, meaning it would have to copy the entire data set every time, which would be inefficient and costly.

AWS Direct Connect is a service that establishes a dedicated network connection between your on-premises data center and AWS, bypassing the public internet and improving the bandwidth and performance of the data transfer. However, AWS Direct Connect is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS DataSync, AWS Storage Gateway, or AWS CLI. AWS Direct Connect also has some hardware and location requirements, and charges you for the port hours and data transfer out of AWS.

Amazon S3 Transfer Acceleration is a feature that enables faster data transfers to Amazon S3 over long distances, using the AWS edge locations and optimized network paths. However, Amazon S3 Transfer Acceleration is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS CLI, AWS SDK, or third-party software. Amazon S3 Transfer Acceleration also charges you for the data transferred over the accelerated endpoints, and does not guarantee a performance improvement for every transfer, as it depends on various factors such as the network conditions, the distance, and the object size. References:

- ? AWS DataSync
- ? AWS Glue
- ? AWS Glue quotas and limits
- ? [AWS Direct Connect]
- ? [Data transfer options for AWS Direct Connect]
- ? [Amazon S3 Transfer Acceleration]
- ? [Using Amazon S3 Transfer Acceleration]

#### NEW QUESTION 56

A data engineer is configuring an AWS Glue job to read data from an Amazon S3 bucket. The data engineer has set up the necessary AWS Glue connection details and an associated IAM role. However, when the data engineer attempts to run the AWS Glue job, the data engineer receives an error message that

indicates that there are problems with the Amazon S3 VPC gateway endpoint.

The data engineer must resolve the error and connect the AWS Glue job to the S3 bucket. Which solution will meet this requirement?

- A. Update the AWS Glue security group to allow inbound traffic from the Amazon S3 VPC gateway endpoint.
- B. Configure an S3 bucket policy to explicitly grant the AWS Glue job permissions to access the S3 bucket.
- C. Review the AWS Glue job code to ensure that the AWS Glue connection details include a fully qualified domain name.
- D. Verify that the VPC's route table includes inbound and outbound routes for the Amazon S3 VPC gateway endpoint.

**Answer:** D

**Explanation:**

The error message indicates that the AWS Glue job cannot access the Amazon S3 bucket through the VPC endpoint. This could be because the VPC's route table does not have the necessary routes to direct the traffic to the endpoint. To fix this, the data engineer must verify that the route table has an entry for the Amazon S3 service prefix (com.amazonaws.region.s3) with the target as the VPC endpoint ID. This will allow the AWS Glue job to use the VPC endpoint to access the S3 bucket without going through the internet or a NAT gateway. For more information, see [Gateway endpoints](#). References:

? [Troubleshoot the AWS Glue error "VPC S3 endpoint validation failed"](#)

? [Amazon VPC endpoints for Amazon S3](#)

? [\[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide\]](#)

**NEW QUESTION 61**

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

- A. Use Amazon EMR and Apache Ranger.
- B. Use a Hive metastore on an EMR cluster.
- C. Use the AWS Glue Data Catalog.
- D. Use a metastore on an Amazon RDS for MySQL DB instance.

**Answer:** C

**Explanation:**

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:

? [Using the AWS Glue Data Catalog as the metastore for Hive \(Section: Specifying](#)

[AWS Glue Data Catalog as the metastore\)](#)

? [Metadata Management: Hive Metastore vs AWS Glue \(Section: AWS Glue Data Catalog\)](#)

? [AWS Glue Data Catalog support for Spark SQL jobs \(Section: Importing metadata from an existing Hive metastore\)](#)

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide \(Chapter 5, page 131\)](#)

**NEW QUESTION 62**

.....

## Thank You for Trying Our Product

### We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

### AWS-Certified-Data-Engineer-Associate Practice Exam Features:

- \* AWS-Certified-Data-Engineer-Associate Questions and Answers Updated Frequently
- \* AWS-Certified-Data-Engineer-Associate Practice Questions Verified by Expert Senior Certified Staff
- \* AWS-Certified-Data-Engineer-Associate Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* AWS-Certified-Data-Engineer-Associate Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

**100% Actual & Verified — Instant Download, Please Click**  
[Order The AWS-Certified-Data-Engineer-Associate Practice Test Here](#)