

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 Kafka¹. 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 complexity². References:

? Amazon Athena Federated Query

? Redshift Spectrum vs Federated Query

NEW QUESTION 2

A data engineer maintains custom Python scripts that perform a data formatting process that many AWS Lambda functions use. When the data engineer needs to modify the Python scripts, the data engineer must manually update all the Lambda functions.

The data engineer requires a less manual way to update the Lambda functions. Which solution will meet this requirement?

- A. Store a pointer to the custom Python scripts in the execution context object in a shared Amazon S3 bucket.
- B. Package the custom Python scripts into Lambda layer
- C. Apply the Lambda layers to the Lambda functions.
- D. Store a pointer to the custom Python scripts in environment variables in a shared Amazon S3 bucket.
- E. Assign the same alias to each Lambda function
- F. Call each Lambda function by specifying the function's alias.

Answer: B

Explanation:

Lambda layers are a way to share code and dependencies across multiple Lambda functions. By packaging the custom Python scripts into Lambda layers, the data engineer can update the scripts in one place and have them automatically applied to all the Lambda functions that use the layer. This reduces the manual effort and ensures consistency across the Lambda functions. The other options are either not feasible or not efficient. Storing a pointer to the custom Python scripts in the execution context object or in environment variables would require the Lambda functions to download the scripts from Amazon S3 every time they are invoked, which would increase latency and cost. Assigning the same alias to each Lambda function would not help with updating the Python scripts, as the alias only points to a specific version of the Lambda function code. References:

? AWS Lambda layers

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.4: AWS Lambda

NEW QUESTION 3

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 week
- 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 4

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

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

- A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost- optimized storage classe
- B. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classe
- C. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.
- D. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequentl
- E. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.
- F. Use S3 Intelligent-Tierin
- G. Activate the Deep Archive Access tier.
- H. Use S3 Intelligent-Tierin
- I. Use the default access tier.

Answer: D

Explanation:

S3 Intelligent-Tiering is a storage class that automatically moves objects between four access tiers based on the changing access patterns. The default access tier consists of two tiers: Frequent Access and Infrequent Access. Objects in the Frequent Access tier have the same performance and availability as S3 Standard, while objects in the Infrequent Access tier have the same performance and availability as S3 Standard-IA. S3 Intelligent-Tiering monitors the access patterns of each object and moves them between the tiers accordingly, without any operational overhead or retrieval fees. This solution can optimize S3 storage costs for data with unpredictable and variable access patterns, while ensuring millisecond latency for data retrieval. The other solutions are not optimal or relevant for this requirement. Using S3 Storage Lens standard metrics and activity metrics can provide insights into the storage usage and access patterns, but they do not automate the data movement between storage classes. Creating S3 Lifecycle policies for the S3 buckets can move objects to more cost-optimized storage classes, but they require manual configuration and maintenance, and they may incur retrieval fees for data that is accessed unexpectedly. Activating the Deep Archive Access tier for S3 Intelligent-Tiering can further reduce the storage costs for data that is rarely accessed, but it also increases the retrieval time to 12 hours, which does not meet the requirement of millisecond latency. References:

? S3 Intelligent-Tiering

? S3 Storage Lens

? S3 Lifecycle policies

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 5

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 AWSGlueServiceRole 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 6

A company uses Amazon S3 to store semi-structured data in a transactional data lake. Some of the data files are small, but other data files are tens of terabytes.

A data engineer must perform a change data capture (CDC) operation to identify changed data from the data source. The data source sends a full snapshot as a JSON file every day and ingests the changed data into the data lake.

Which solution will capture the changed data MOST cost-effectively?

- A. Create an AWS Lambda function to identify the changes between the previous data and the current dat
- B. Configure the Lambda function to ingest the changes into the data lake.
- C. Ingest the data into Amazon RDS for MySQL
- D. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.
- E. Use an open source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data.
- F. Ingest the data into an Amazon Aurora MySQL DB instance that runs Aurora Serverles
- G. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

Answer: C

Explanation:

An open source data lake format, such as Apache Parquet, Apache ORC, or Delta Lake, is a cost-effective way to perform a change data capture (CDC) operation on semi-structured data stored in Amazon S3. An open source data lake format allows you to query data directly from S3 using standard SQL, without the need to move or copy data to another service. An open source data lake format also supports schema evolution, meaning it can handle changes in the data structure over

time. An open source data lake format also supports upserts, meaning it can insert new data and update existing data in the same operation, using a merge command. This way, you can efficiently capture the changes from the data source and apply them to the S3 data lake, without duplicating or losing any data. The other options are not as cost-effective as using an open source data lake format, as they involve additional steps or costs. Option A requires you to create and maintain an AWS Lambda function, which can be complex and error-prone. AWS Lambda also has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the CDC operation. Option B and D require you to ingest the data into a relational database service, such as Amazon RDS or Amazon Aurora, which can be expensive and unnecessary for semi-structured data. AWS Database Migration Service (AWS DMS) can write the changed data to the data lake, but it also charges you for the data replication and transfer. Additionally, AWS DMS does not support JSON as a source data type, so you would need to convert the data to a supported format before using AWS DMS. References:

- ? What is a data lake?
- ? Choosing a data format for your data lake
- ? Using the MERGE INTO command in Delta Lake
- ? [AWS Lambda quotas]
- ? [AWS Database Migration Service quotas]

NEW QUESTION 7

A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline. Which AWS service or feature will meet these requirements MOST cost-effectively?

- A. AWS Step Functions
- B. AWS Glue workflows
- C. AWS Glue Studio
- D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Answer: B

Explanation:

AWS Glue workflows are a cost-effective way to orchestrate complex ETL jobs that involve multiple crawlers, jobs, and triggers. AWS Glue workflows allow you to visually monitor the progress and dependencies of your ETL tasks, and automatically handle errors and retries. AWS Glue workflows also integrate with other AWS services, such as Amazon S3, Amazon Redshift, and AWS Lambda, among others, enabling you to leverage these services for your data processing workflows. AWS Glue workflows are serverless, meaning you only pay for the resources you use, and you don't have to manage any infrastructure.

AWS Step Functions, AWS Glue Studio, and Amazon MWAA are also possible options for orchestrating ETL pipelines, but they have some drawbacks compared to AWS Glue workflows. AWS Step Functions is a serverless function orchestrator that can handle different types of data processing, such as real-time, batch, and stream processing. However, AWS Step Functions requires you to write code to define your state machines, which can be complex and error-prone. AWS Step Functions also charges you for every state transition, which can add up quickly for large-scale ETL pipelines.

AWS Glue Studio is a graphical interface that allows you to create and run AWS Glue ETL jobs without writing code. AWS Glue Studio simplifies the process of building, debugging, and monitoring your ETL jobs, and provides a range of pre-built transformations and connectors. However, AWS Glue Studio does not support workflows, meaning you cannot orchestrate multiple ETL jobs or crawlers with dependencies and triggers. AWS Glue Studio also does not support streaming data sources or targets, which limits its use cases for real-time data processing.

Amazon MWAA is a fully managed service that makes it easy to run open-source versions of Apache Airflow on AWS and build workflows to run your ETL jobs and data pipelines. Amazon MWAA provides a familiar and flexible environment for data engineers who are familiar with Apache Airflow, and integrates with a range of AWS services such as Amazon EMR, AWS Glue, and AWS Step Functions. However, Amazon MWAA is not serverless, meaning you have to provision and pay for the resources you need, regardless of your usage. Amazon MWAA also requires you to write code to define your DAGs, which can be challenging and time-consuming for complex ETL pipelines. References:

- ? AWS Glue Workflows
- ? AWS Step Functions
- ? AWS Glue Studio
- ? Amazon MWAA
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 8

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour. Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

- A. Configure AWS Glue triggers to run the ETL jobs even/ hour.
- B. Use AWS Glue DataBrew to clean and prepare the data for analytics.
- C. Use AWS Lambda functions to schedule and run the ETL jobs even/ hour.
- D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
- E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Answer: AD

Explanation:

The correct answer is to configure AWS Glue triggers to run the ETL jobs every hour and use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift. AWS Glue triggers are a way to schedule and orchestrate ETL jobs with the least operational overhead. AWS Glue connections are a way to securely connect to data sources and targets using JDBC or MongoDB drivers. AWS Glue DataBrew is a visual data preparation tool that does not support MongoDB as a data source. AWS Lambda functions are a serverless option to schedule and run ETL jobs, but they have a limit of 15 minutes for execution time, which may not be enough for complex transformations. The Redshift Data API is a way to run SQL commands on Amazon Redshift clusters without needing a persistent connection, but it does not support loading data from AWS Glue ETL jobs. References:

- ? AWS Glue triggers
- ? AWS Glue connections
- ? AWS Glue DataBrew
- ? [AWS Lambda functions]
- ? [Redshift Data API]

NEW QUESTION 9

A data engineer needs to use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket. When the data engineer connects to the QuickSight dashboard, the data engineer receives an error message that indicates insufficient permissions. Which factors could cause the permissions-related errors? (Choose two.)

- A. There is no connection between QuickSight and Athena.
- B. The Athena tables are not cataloged.
- C. QuickSight does not have access to the S3 bucket.
- D. QuickSight does not have access to decrypt S3 data.
- E. There is no IAM role assigned to QuickSight.

Answer: CD

Explanation:

QuickSight does not have access to the S3 bucket and QuickSight does not have access to decrypt S3 data are two possible factors that could cause the permissions-related errors. Amazon QuickSight is a business intelligence service that allows you to create and share interactive dashboards based on various data sources, including Amazon Athena. Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. To use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket, you need to grant QuickSight access to both Athena and S3, as well as any encryption keys that are used to encrypt the S3 data. If QuickSight does not have access to the S3 bucket or the encryption keys, it will not be able to read the data from Athena and display it on the dashboard, resulting in an error message that indicates insufficient permissions.

The other options are not factors that could cause the permissions-related errors. Option A, there is no connection between QuickSight and Athena, is not a factor, as QuickSight supports Athena as a native data source, and you can easily create a connection between them using the QuickSight console or the API. Option B, the Athena tables are not cataloged, is not a factor, as QuickSight can automatically discover the Athena tables that are cataloged in the AWS Glue Data Catalog, and you can also manually specify the Athena tables that are not cataloged. Option E, there is no IAM role assigned to QuickSight, is not a factor, as QuickSight requires an IAM role to access any AWS data sources, including Athena and S3, and you can create and assign an IAM role to QuickSight using the QuickSight console or the API. References:

? Using Amazon Athena as a Data Source

? Granting Amazon QuickSight Access to AWS Resources

? Encrypting Data at Rest in Amazon S3

NEW QUESTION 10

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 Python¹. Amazon Athena also supports Apache Spark, an open-source distributed processing framework that can run large-scale data analytics applications across clusters of servers². 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 metastore³. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications⁴. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis⁵.

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 idle¹. 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 components⁶. 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 Catalog⁷. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores⁸. 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 infrastructure⁹. 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^{6,7}. AWS Glue charges you based on the compute time and the data processed by your ETL jobs⁸. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads⁹. 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 capacity¹⁰. 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 SQL¹¹. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command¹². 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 cluster¹⁰. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries¹¹. 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 servers¹³. 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 made⁶⁷. AWS Lambda charges you based on the number of requests and the duration of your functions¹³. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads⁹. 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 12

A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.

A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.

Which solution will meet these requirements with the LEAST latency?

- A. Schedule an AWS Glue crawler to run every morning.
- B. Manually run the AWS Glue CreatePartition API twice each day.
- C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call.
- D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Answer: C

Explanation:

The best solution to ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket with the least latency is to use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call. This way, the Data Catalog is updated as soon as new data is written to S3, and the partition information is immediately available for querying by other services. The Boto3 AWS Glue create partition API call allows you to create a new partition in the Data Catalog by specifying the table name, the database name, and the partition values¹. You can use this API call in your code that writes data to S3, such as a Python script or an AWS Glue ETL job, to create a partition for each new S3 object key that matches the partitioning scheme.

Option A is not the best solution, as scheduling an AWS Glue crawler to run every morning would introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. 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 Catalog². Crawlers can be scheduled to run periodically, such as daily or hourly, but they cannot run continuously or in real-time. Therefore, using a crawler to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency.

Option B is not the best solution, as manually running the AWS Glue CreatePartition API twice each day would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. Moreover, manually running the API would require more operational overhead and human intervention than using code that writes data to S3 to invoke the API automatically.

Option D is not the best solution, as running the MSCK REPAIR TABLE command from the AWS Glue console would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. The MSCK REPAIR TABLE command is a SQL command that you can run in the AWS Glue console to add partitions to the Data Catalog based on the S3 object keys that match the partitioning scheme³. However, this command is not meant to be run frequently or in real-time, as it can take a long time to scan the entire S3 bucket and add the partitions. Therefore, using this command to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. References:

? AWS Glue CreatePartition API

? Populating the AWS Glue Data Catalog

? MSCK REPAIR TABLE Command

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 13

During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script.

A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.

Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

- A. Store the credentials in the AWS Glue job parameters.
- B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
- C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
- D. Store the credentials in AWS Secrets Manager.
- E. Grant the AWS Glue job 1AM role access to the stored credentials.

Answer: DE

Explanation:

AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt the credentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the credentials.

The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled. References:

? AWS Secrets Manager

? AWS Glue

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

NEW QUESTION 18

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 19

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views. Which solution will meet this requirement with the LEAST effort?

- A. Use Apache Airflow to refresh the materialized views.
- B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.
- C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.
- D. Use an AWS Glue workflow to refresh the materialized views.

Answer: C

Explanation:

The query editor v2 in Amazon Redshift is a web-based tool that allows users to run SQL queries and scripts on Amazon Redshift clusters. The query editor v2 supports creating and managing materialized views, which are precomputed results of a query that can improve the performance of subsequent queries. The query editor v2 also supports scheduling queries to run at specified intervals, which can be used to refresh materialized views automatically. This solution requires the least effort, as it does not involve any additional services, coding, or configuration. The other solutions are more complex and require more operational overhead. Apache Airflow is an open-source platform for orchestrating workflows, which can be used to refresh materialized views, but it requires setting up and managing an Airflow environment, creating DAGs (directed acyclic graphs) to define the workflows, and integrating with Amazon Redshift. AWS Lambda is a serverless compute service that can run code in response to events, which can be used to refresh materialized views, but it requires creating and deploying Lambda functions, defining UDFs within Amazon Redshift, and triggering the functions using events or schedules. AWS Glue is a fully managed ETL service that can run jobs to transform and load data, which can be used to refresh materialized views, but it requires creating and configuring Glue jobs, defining Glue workflows to orchestrate the jobs, and scheduling the workflows using triggers. References:

? Query editor V2

? Working with materialized views

? Scheduling queries

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 24

A data engineer needs to build an extract, transform, and load (ETL) job. The ETL job will process daily incoming .csv files that users upload to an Amazon S3 bucket. The size of each S3 object is less than 100 MB.

Which solution will meet these requirements MOST cost-effectively?

- A. Write a custom Python application
- B. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.
- C. Write a PySpark ETL script

- D. Host the script on an Amazon EMR cluster.
- E. Write an AWS Glue PySpark job
- F. Use Apache Spark to transform the data.
- G. Write an AWS Glue Python shell job
- H. Use pandas to transform the data.

Answer: D

Explanation:

AWS Glue is a fully managed serverless ETL service that can handle various data sources and formats, including .csv files in Amazon S3. AWS Glue provides two types of jobs: PySpark and Python shell. PySpark jobs use Apache Spark to process large-scale data in parallel, while Python shell jobs use Python scripts to process small-scale data in a single execution environment. For this requirement, a Python shell job is more suitable and cost-effective, as the size of each S3 object is less than 100 MB, which does not require distributed processing. A Python shell job can use pandas, a popular Python library for data analysis, to transform the .csv data as needed. The other solutions are not optimal or relevant for this requirement. Writing a custom Python application and hosting it on an Amazon EKS cluster would require more effort and resources to set up and manage the Kubernetes environment, as well as to handle the data ingestion and transformation logic. Writing a PySpark ETL script and hosting it on an Amazon EMR cluster would also incur more costs and complexity to provision and configure the EMR cluster, as well as to use Apache Spark for processing small data files. Writing an AWS Glue PySpark job would also be less efficient and economical than a Python shell job, as it would involve unnecessary overhead and charges for using Apache Spark for small data files. References:

- ? AWS Glue
- ? Working with Python Shell Jobs
- ? pandas
- ? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 29

A company loads transaction data for each day into Amazon Redshift tables at the end of each day. The company wants to have the ability to track which tables have been loaded and which tables still need to be loaded.

A data engineer wants to store the load statuses of Redshift tables in an Amazon DynamoDB table. The data engineer creates an AWS Lambda function to publish the details of the load statuses to DynamoDB.

How should the data engineer invoke the Lambda function to write load statuses to the DynamoDB table?

- A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.
- B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridge
- C. Configure an EventBridge rule to invoke the Lambda function.
- D. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue
- E. Configure the SQS queue to invoke the Lambda function.
- F. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

Answer: B

Explanation:

The Amazon Redshift Data API enables you to interact with your Amazon Redshift data warehouse in an easy and secure way. You can use the Data API to run SQL commands, such as loading data into tables, without requiring a persistent connection to the cluster. The Data API also integrates with Amazon EventBridge, which allows you to monitor the execution status of your SQL commands and trigger actions based on events. By using the Data API to publish an event to EventBridge, the data engineer can invoke the Lambda function that writes the load statuses to the DynamoDB table. This solution is scalable, reliable, and cost-effective. The other options are either not possible or not optimal. You cannot use a second Lambda function to invoke the first Lambda function based on CloudWatch or CloudTrail events, as these services do not capture the load status of Redshift tables. You can use the Data API to publish a message to an SQS queue, but this would require additional configuration and polling logic to invoke the Lambda function from the queue. This would also introduce additional latency and cost. References:

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

NEW QUESTION 31

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 C) 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 35

A company stores data from an application in an Amazon DynamoDB table that operates in provisioned capacity mode. The workloads of the application have predictable throughput load on a regular schedule. Every Monday, there is an immediate increase in activity early in the morning. The application has very low usage during weekends.

The company must ensure that the application performs consistently during peak usage times

Which solution will meet these requirements in the MOST cost-effective way?

A. Increase the provisioned capacity to the maximum capacity that is currently present during peak load times.

B. Divide the table into two table

C. Provision each table with half of the provisioned capacity of the original tabl

D. Spread queries evenly across both tables.

E. Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage time

F. Schedule lower capacity during off-peak times.

G. Change the capacity mode from provisioned to on-deman

H. Configure the table to scale up and scale down based on the load on the table.

Answer: C

Explanation:

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB offers two capacity modes for throughput capacity: provisioned and on-demand. In provisioned capacity mode, you specify the number of read and write capacity units per second that you expect your application to require. DynamoDB reserves the resources to meet your throughput needs with consistent performance. In on-demand capacity mode, you pay per request and DynamoDB scales the resources up and down automatically based on the actual workload. On-demand capacity mode is suitable for unpredictable workloads that can vary significantly over time¹.

The solution that meets the requirements in the most cost-effective way is to use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times and lower capacity during off-peak times. This solution has the following advantages:

? It allows you to optimize the cost and performance of your DynamoDB table by adjusting the provisioned capacity according to your predictable workload patterns. You can use scheduled scaling to specify the date and time for the scaling actions, and the new minimum and maximum capacity limits. For example, you can schedule higher capacity for every Monday morning and lower capacity for weekends².

? It enables you to take advantage of the lower cost per unit of provisioned capacity mode compared to on-demand capacity mode. Provisioned capacity mode charges a flat hourly rate for the capacity you reserve, regardless of how much you use. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on-demand capacity mode¹.

? It ensures that your application performs consistently during peak usage times by having enough capacity to handle the increased load. You can also use auto scaling to automatically adjust the provisioned capacity based on the actual utilization of your table, and set a target utilization percentage for your table or global secondary index. This way, you can avoid under-provisioning or over-provisioning your table².

Option A is incorrect because it suggests increasing the provisioned capacity to the maximum capacity that is currently present during peak load times. This solution has the following disadvantages:

? It wastes money by paying for unused capacity during off-peak times. If you provision the same high capacity for all times, regardless of the actual workload, you are over-provisioning your table and paying for resources that you don't need¹.

? It does not account for possible changes in the workload patterns over time. If your peak load times increase or decrease in the future, you may need to manually adjust the provisioned capacity to match the new demand. This adds operational overhead and complexity to your application².

Option B is incorrect because it suggests dividing the table into two tables and provisioning each table with half of the provisioned capacity of the original table. This solution has the following disadvantages:

? It complicates the data model and the application logic by splitting the data into two separate tables. You need to ensure that the queries are evenly distributed across both tables, and that the data is consistent and synchronized between them. This adds extra development and maintenance effort to your application³.

? It does not solve the problem of adjusting the provisioned capacity according to the workload patterns. You still need to manually or automatically scale the capacity of each table based on the actual utilization and demand. This may result in under-provisioning or over-provisioning your tables².

Option D is incorrect because it suggests changing the capacity mode from provisioned to on-demand. This solution has the following disadvantages:

? It may incur higher costs than provisioned capacity mode for predictable workloads. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on-demand capacity mode, as you can reserve the capacity you need at a lower rate¹.

? It may not provide consistent performance during peak usage times, as on-demand capacity mode may take some time to scale up the resources to meet the sudden increase in demand. On-demand capacity mode uses adaptive capacity to handle bursts of traffic, but it may not be able to handle very large spikes or sustained high throughput. In such cases, you may experience throttling or increased latency.

References:

? 1: Choosing the right DynamoDB capacity mode - Amazon DynamoDB

? 2: Managing throughput capacity automatically with DynamoDB auto scaling - Amazon DynamoDB

? 3: Best practices for designing and using partition keys effectively - Amazon DynamoDB

? [4]: On-demand mode guidelines - Amazon DynamoDB

? [5]: How to optimize Amazon DynamoDB costs - AWS Database Blog

? [6]: DynamoDB adaptive capacity: How it works and how it helps - AWS Database Blog

? [7]: Amazon DynamoDB pricing - Amazon Web Services (AWS)

NEW QUESTION 36

A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.

A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation.

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

A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucke

B. Create a pipeline in Apache Spark.

C. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucke

- D. Create a pipeline in Apache Spark.
- E. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.
- F. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table.
- G. Access the table from Amazon S3.

Answer: B

Explanation:

AWS Glue is a fully managed service that provides a serverless data integration platform. It can automatically discover and categorize data from various sources, including SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. It can also infer the schema of the data and store it in the AWS Glue Data Catalog, which is a central metadata repository. AWS Glue can then use the schema information to generate and run Apache Spark code to extract, transform, and load the data into an Amazon S3 bucket. AWS Glue can also monitor and optimize the performance and cost of the data pipeline, and handle any schema changes that may occur in the source data. AWS Glue can meet the SLA of loading the data into the S3 bucket within 15 minutes of data creation, as it can trigger the data pipeline based on events, schedules, or on-demand. AWS Glue has the least operational overhead among the options, as it does not require provisioning, configuring, or managing any servers or clusters. It also handles scaling, patching, and security automatically. References:

- ? AWS Glue
- ? [AWS Glue Data Catalog]
- ? [AWS Glue Developer Guide]
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 41

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 45

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the data. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

- A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.
- B. Use an S3 bucket that is in the same account that uses Athena to query the data.
- C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.
- D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.
- E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Answer: CE

Explanation:

Using an S3 bucket that is in the same AWS Region where the company runs Athena queries can improve query performance by reducing data transfer latency and costs. Preprocessing the .csv data to Apache Parquet format can also improve query performance by enabling columnar storage, compression, and partitioning, which can reduce the amount of data scanned and fetched by the query. These solutions can optimize the storage solution for the POC test without requiring much effort or changes to the existing data pipeline. The other solutions are not optimal or relevant for this requirement. Adding a randomized string to the beginning of the keys in Amazon S3 can improve the throughput across partitions, but it can also make the data harder to query and manage. Using an S3 bucket that is in the same account that uses Athena to query the data does not have any significant impact on query performance, as long as the proper permissions are granted. Preprocessing the .csv data to JSON format does not offer any benefits over the .csv format, as both are row-based and verbose formats that require more data scanning and fetching than columnar formats like Parquet. References:

- ? Best Practices When Using Athena with AWS Glue
- ? Optimizing Amazon S3 Performance
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 48

A manufacturing company collects sensor data from its factory floor to monitor and enhance operational efficiency. The company uses Amazon Kinesis Data Streams to publish the data that the sensors collect to a data stream. Then Amazon Kinesis Data Firehose writes the data to an Amazon S3 bucket.

The company needs to display a real-time view of operational efficiency on a large screen in the manufacturing facility. Which solution will meet these requirements with the LOWEST latency?

- A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data
- B. Use a connector for Apache Flink to write data to an Amazon Timestream database
- C. Use the Timestream database as a source to create a Grafana dashboard.
- D. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created
- E. Use the Lambda function to publish the data to Amazon Aurora
- F. Use Aurora as a source to create an Amazon QuickSight dashboard.
- G. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data
- H. Create a new Data Firehose delivery stream to publish data directly to an Amazon Timestream database
- I. Use the Timestream database as a source to create an Amazon QuickSight dashboard.
- J. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time
- K. Publish the data to an Amazon Timestream database
- L. Use the Timestream database as a source to create a Grafana dashboard.

Answer: C

Explanation:

This solution will meet the requirements with the lowest latency because it uses Amazon Managed Service for Apache Flink to process the sensor data in real time and write it to Amazon Timestream, a fast, scalable, and serverless time series database. Amazon Timestream is optimized for storing and analyzing time series data, such as sensor data, and can handle trillions of events per day with millisecond latency. By using Amazon Timestream as a source, you can create an Amazon QuickSight dashboard that displays a real-time view of operational efficiency on a large screen in the manufacturing facility. Amazon QuickSight is a fully managed business intelligence service that can connect to various data sources, including Amazon Timestream, and provide interactive visualizations and insights.

The other options are not optimal for the following reasons:

? A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is similar to option C, but it uses Grafana instead of Amazon QuickSight to create the dashboard. Grafana is an open source visualization tool that can also connect to Amazon Timestream, but it requires additional steps to set up and configure, such as deploying a Grafana server on Amazon EC2, installing the Amazon Timestream plugin, and creating an IAM role for Grafana to access Timestream. These steps can increase the latency and complexity of the solution.

? B. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard. This option is not suitable for displaying a real-time view of operational efficiency, as it introduces unnecessary delays and costs in the data pipeline. First, the sensor data is written to an S3 bucket by Amazon Kinesis Data Firehose, which can have a buffering interval of up to 900 seconds. Then, the S3 bucket sends a notification to a Lambda function, which can incur additional invocation and execution time. Finally, the Lambda function publishes the data to Amazon Aurora, a relational database that is not optimized for time series data and can have higher storage and performance costs than Amazon Timestream.

? D. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time.

Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is also not suitable for displaying a real-time view of operational efficiency, as it uses AWS Glue bookmarks to read sensor data from the S3 bucket. AWS Glue bookmarks are a feature that helps AWS Glue jobs and crawlers keep track of the data that has already been processed, so that they can resume from where they left off. However, AWS Glue jobs and crawlers are not designed for real-time data processing, as they can have a minimum frequency of 5 minutes and a variable start-up time. Moreover, this option also uses Grafana instead of Amazon QuickSight to create the dashboard, which can increase the latency and complexity of the solution.

References:

? 1: Amazon Managed Streaming for Apache Flink

? 2: Amazon Timestream

? 3: Amazon QuickSight

? : Analyze data in Amazon Timestream using Grafana

? : Amazon Kinesis Data Firehose

? : Amazon Aurora

? : AWS Glue Bookmarks

? : AWS Glue Job and Crawler Scheduling

NEW QUESTION 52

A company stores daily records of the financial performance of investment portfolios in .csv format in an Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog. Which solution will meet these requirements?

- A. Create an IAM role that includes the AmazonS3FullAccess policy
- B. Associate the role with the crawler
- C. Specify the S3 bucket path of the source data as the crawler's data store
- D. Create a daily schedule to run the crawler
- E. Configure the output destination to a new path in the existing S3 bucket.
- F. Create an IAM role that includes the AWSGlueServiceRole policy
- G. Associate the role with the crawler
- H. Specify the S3 bucket path of the source data as the crawler's data store
- I. Create a daily schedule to run the crawler
- J. Specify a database name for the output.
- K. Create an IAM role that includes the AmazonS3FullAccess policy
- L. Associate the role with the crawler
- M. Specify the S3 bucket path of the source data as the crawler's data store
- N. Allocate data processing units (DPUs) to run the crawler every day
- O. Specify a database name for the output.
- P. Create an IAM role that includes the AWSGlueServiceRole policy
- Q. Associate the role with the crawler
- R. Specify the S3 bucket path of the source data as the crawler's data store
- S. Allocate data processing units (DPUs) to run the crawler every day
- T. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:

To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

- ? Create an IAM role that has the necessary permissions to access the S3 data and the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions¹.
 - ? Associate the role with the crawler.
 - ? Specify the S3 bucket path of the source data as the crawler's data store. The crawler will scan the data and infer the schema and format².
 - ? Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data³.
 - ? Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.
- Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.
- Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.
- Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.
- Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

References:

- ? 1: AWS managed (predefined) policies for AWS Glue - AWS Glue
- ? 2: Data Catalog and crawlers in AWS Glue - AWS Glue
- ? 3: Scheduling an AWS Glue crawler - AWS Glue
- ? [4]: Parameters set on Data Catalog tables by crawler - AWS Glue
- ? [5]: AWS Glue pricing - Amazon Web Services (AWS)

NEW QUESTION 55

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 functions¹².

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 56

A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.

A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data. Which solution will meet this requirement?

- A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data.
- B. Apply the default settings to the EC2 instances.
- C. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data.
- D. Apply the default settings to the EC2 instances.
- E. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume.
- F. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data.
- G. Apply the default settings to the EC2 instances.
- H. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store (Amazon EBS) volume.

- I. Attach an additional EC2 instance store volume to contain the application data.
- J. Apply the default settings to the EC2 instances.

Answer: C

Explanation:

Amazon EC2 instances can use two types of storage volumes: instance store volumes and Amazon EBS volumes. Instance store volumes are ephemeral, meaning they are only attached to the instance for the duration of its life cycle. If the instance is stopped, terminated, or fails, the data on the instance store volume is lost. Amazon EBS volumes are persistent, meaning they can be detached from the instance and attached to another instance, and the data on the volume is preserved. To meet the requirement of persisting the data even if the EC2 instances are terminated, the data engineer must use Amazon EBS volumes to store the application data. The solution is to launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume, which is the default option for most AMIs. Then, the data engineer must attach an Amazon EBS volume to each instance and configure the application to write the data to the EBS volume. This way, the data will be saved on the EBS volume and can be accessed by another instance if needed. The data engineer can apply the default settings to the EC2 instances, as there is no need to modify the instance type, security group, or IAM role for this solution. The other options are either not feasible or not optimal. Launching new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data (option A) or by using an AMI that is backed by a root Amazon EBS volume that contains the application data (option B) would not work, as the data on the AMI would be outdated and overwritten by the new instances. Attaching an additional EC2 instance store volume to contain the application data (option D) would not work, as the data on the instance store volume would be lost if the instance is terminated. References:

? Amazon EC2 Instance Store

? Amazon EBS Volumes

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.1: Amazon EC2

NEW QUESTION 57

A company uses an Amazon Redshift provisioned cluster as its database. The Redshift cluster has five reserved ra3.4xlarge nodes and uses key distribution. A data engineer notices that one of the nodes frequently has a CPU load over 90%. SQL Queries that run on the node are queued. The other four nodes usually have a CPU load under 15% during daily operations.

The data engineer wants to maintain the current number of compute nodes. The data engineer also wants to balance the load more evenly across all five compute nodes.

Which solution will meet these requirements?

- A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.
- B. Change the distribution key to the table column that has the largest dimension.
- C. Upgrade the reserved node from ra3.4xlarge to ra3.16xlarge.
- D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Answer: B

Explanation:

Changing the distribution key to the table column that has the largest dimension will help to balance the load more evenly across all five compute nodes. The distribution key determines how the rows of a table are distributed among the slices of the cluster. If the distribution key is not chosen wisely, it can cause data skew, meaning some slices will have more data than others, resulting in uneven CPU load and query performance. By choosing the table column that has the largest dimension, meaning the column that has the most distinct values, as the distribution key, the data engineer can ensure that the rows are distributed more uniformly across the slices, reducing data skew and improving query performance.

The other options are not solutions that will meet the requirements. Option A, changing the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load. The sort key determines the order in which the rows of a table are stored on disk, which can improve the performance of range-restricted queries, but not the load balancing. Option C, upgrading the reserved node from ra3.4xlarge to ra3.16xlarge, will not maintain the current number of compute nodes, as it will increase the cost and the capacity of the cluster. Option D, changing the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load either.

The primary key is a constraint that enforces the uniqueness of the rows in a table, but it does not influence the data layout or the query optimization. References:

? Choosing a data distribution style

? Choosing a data sort key

? Working with primary keys

NEW QUESTION 59

A company needs to build a data lake in AWS. The company must provide row-level data access and column-level data access to specific teams. The teams will access the data by using Amazon Athena, Amazon Redshift Spectrum, and Apache Hive from Amazon EMR.

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

- A. Use Amazon S3 for data lake storage.
- B. Use S3 access policies to restrict data access by rows and column.
- C. Provide data access through Amazon S3.
- D. Use Amazon S3 for data lake storage.
- E. Use Apache Ranger through Amazon EMR to restrict data access by rows and column.
- F. Provide data access by using Apache Pig.
- G. Use Amazon Redshift for data lake storage.
- H. Use Redshift security policies to restrict data access by rows and column.
- I. Provide data access by using Apache Spark and Amazon Athena federated queries.
- J. Use Amazon S3 for data lake storage.
- K. Use AWS Lake Formation to restrict data access by rows and column.
- L. Provide data access through AWS Lake Formation.

Answer: D

Explanation:

Option D is the best solution to meet the requirements with the least operational overhead because AWS Lake Formation is a fully managed service that simplifies the process of building, securing, and managing data lakes. AWS Lake Formation allows you to define granular data access policies at the row and column level for different users and groups. AWS Lake Formation also integrates with Amazon Athena, Amazon Redshift Spectrum, and Apache Hive on Amazon EMR, enabling these services to access the data in the data lake through AWS Lake Formation.

Option A is not a good solution because S3 access policies cannot restrict data access by rows and columns. S3 access policies are based on the identity and permissions of the requester, the bucket and object ownership, and the object prefix and tags. S3 access policies cannot enforce fine-grained data access control at the row and column level. Option B is not a good solution because it involves using Apache Ranger and Apache Pig, which are not fully managed services and

require additional configuration and maintenance. Apache Ranger is a framework that provides centralized security administration for data stored in Hadoop clusters, such as Amazon EMR. Apache Ranger can enforce row-level and column-level access policies for Apache Hive tables. However, Apache Ranger is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters. Apache Pig is a platform that allows you to analyze large data sets using a high-level scripting language called Pig Latin. Apache Pig can access data stored in Amazon S3 and process it using Apache Hive. However, Apache Pig is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters.

Option C is not a good solution because Amazon Redshift is not a suitable service for data lake storage. Amazon Redshift is a fully managed data warehouse service that allows you to run complex analytical queries using standard SQL. Amazon Redshift can enforce row-level and column-level access policies for different users and groups. However, Amazon Redshift is not designed to store and process large volumes of unstructured or semi-structured data, which are typical characteristics of data lakes. Amazon Redshift is also more expensive and less scalable than Amazon S3 for data lake storage.

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? What Is AWS Lake Formation? - AWS Lake Formation
- ? Using AWS Lake Formation with Amazon Athena - AWS Lake Formation
- ? Using AWS Lake Formation with Amazon Redshift Spectrum - AWS Lake Formation
- ? Using AWS Lake Formation with Apache Hive on Amazon EMR - AWS Lake Formation
- ? Using Bucket Policies and User Policies - Amazon Simple Storage Service
- ? Apache Ranger
- ? Apache Pig
- ? What Is Amazon Redshift? - Amazon Redshift

NEW QUESTION 62

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 63

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 67

A company uses Amazon RDS to store transactional data. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.

The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.

Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

- A. Turn on the public access setting for the DB instance.
- B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.
- C. Configure the Lambda function to run in the same subnet that the DB instance uses.
- D. Attach the same security group to the Lambda function and the DB instance.
- E. Include a self-referencing rule that allows access through the database port.
- F. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Answer: CD

Explanation:

To enable the Lambda function to connect to the RDS DB instance privately without using the public internet, the best combination of steps is to configure the Lambda function to run in the same subnet that the DB instance uses, and attach the same security group to the Lambda function and the DB instance. This way, the Lambda function and the DB instance can communicate within the same private network, and the security group can allow traffic between them on the database port. This solution has the least operational overhead, as it does not require any changes to the public access setting, the network ACL, or the security group of the DB instance.

The other options are not optimal for the following reasons:

? A. Turn on the public access setting for the DB instance. This option is not recommended, as it would expose the DB instance to the public internet, which can compromise the security and privacy of the data. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? B. Update the security group of the DB instance to allow only Lambda function invocations on the database port. This option is not sufficient, as it would only modify the inbound rules of the security group of the DB instance, but not the outbound rules of the security group of the Lambda function. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

? E. Update the network ACL of the private subnet to include a self-referencing rule

that allows access through the database port. This option is not necessary, as the network ACL of the private subnet already allows all traffic within the subnet by default. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

References:

- ? 1: Connecting to an Amazon RDS DB instance
- ? 2: Configuring a Lambda function to access resources in a VPC
- ? 3: Working with security groups
- ? : Network ACLs

NEW QUESTION 72

A company currently stores all of its data in Amazon S3 by using the S3 Standard storage class.

A data engineer examined data access patterns to identify trends. During the first 6 months, most data files are accessed several times each day. Between 6 months and 2 years, most data files are accessed once or twice each month. After 2 years, data files are accessed only once or twice each year.

The data engineer needs to use an S3 Lifecycle policy to develop new data storage rules. The new storage solution must continue to provide high availability.

Which solution will meet these requirements in the MOST cost-effective way?

- A. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 month
- B. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
- C. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 month
- D. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.
- E. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 month
- F. Transfer objects to S3 Glacier Deep Archive after 2 years.
- G. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 month
- H. Transfer objects to S3 Glacier Deep Archive after 2 years.

Answer: C

Explanation:

To achieve the most cost-effective storage solution, the data engineer needs to use an S3 Lifecycle policy that transitions objects to lower-cost storage classes based on their access patterns, and deletes them when they are no longer needed. The storage classes should also provide high availability, which means they should be resilient to the loss of data in a single Availability Zone¹. Therefore, the solution must include the following steps:

? Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. S3 Standard-IA is designed for data that is accessed less frequently, but requires rapid access when needed. It offers the same high durability, throughput, and low latency as S3 Standard, but with a lower storage cost and a retrieval fee².

Therefore, it is suitable for data files that are accessed once or twice each month. S3 Standard-IA also provides high availability, as it stores data redundantly across multiple Availability Zones¹.

? Transfer objects to S3 Glacier Deep Archive after 2 years. S3 Glacier Deep Archive is the lowest-cost storage class that offers secure and durable storage for data that is rarely accessed and can tolerate a 12-hour retrieval time. It is ideal for long-term archiving and digital preservation³. Therefore, it is suitable for data files that are accessed only once or twice each year. S3 Glacier Deep Archive also provides high availability, as it stores data across at least three geographically dispersed Availability Zones¹.

? Delete objects when they are no longer needed. The data engineer can specify an expiration action in the S3 Lifecycle policy to delete objects after a certain period of time. This will reduce the storage cost and comply with any data retention policies.

Option C is the only solution that includes all these steps. Therefore, option C is the correct answer.

Option A is incorrect because it transitions objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. S3 One Zone-IA is similar to S3 Standard-

IA, but it stores data in a single Availability Zone. This means it has a lower availability and durability than S3 Standard-IA, and it is not resilient to the loss of data in a single Availability Zone¹. Therefore, it does not provide high availability as required.

Option B is incorrect because it transfers objects to S3 Glacier Flexible Retrieval after 2 years. S3 Glacier Flexible Retrieval is a storage class that offers secure and durable storage for data that is accessed infrequently and can tolerate a retrieval time of minutes to hours. It is more expensive than S3 Glacier Deep Archive, and it is not suitable for data that is accessed only once or twice each year³. Therefore, it is not the most cost-effective option.

Option D is incorrect because it combines the errors of option A and B. It transitions objects to S3 One Zone-IA after 6 months, which does not provide high availability, and it transfers objects to S3 Glacier Flexible Retrieval after 2 years, which is not the most cost-effective option.

References:

- ? 1: Amazon S3 storage classes - Amazon Simple Storage Service
- ? 2: Amazon S3 Standard-Infrequent Access (S3 Standard-IA) - Amazon Simple Storage Service
- ? 3: Amazon S3 Glacier and S3 Glacier Deep Archive - Amazon Simple Storage Service
- ? [4]: Expiring objects - Amazon Simple Storage Service
- ? [5]: Managing your storage lifecycle - Amazon Simple Storage Service
- ? [6]: Examples of S3 Lifecycle configuration - Amazon Simple Storage Service
- ? [7]: Amazon S3 Lifecycle further optimizes storage cost savings with new features
- What's New with AWS

NEW QUESTION 75

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 performance¹.

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 metastore².

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 scenario³.

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 scenario⁴.

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 scenario⁵. 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 79

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 services¹. 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 events¹.

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 S3². 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 process³. 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 81

A company is planning to use a provisioned Amazon EMR cluster that runs Apache Spark jobs to perform big data analysis. The company requires high reliability. A big data team must follow best practices for running cost-optimized and long-running workloads on Amazon EMR. The team must find a solution that will maintain the company's current level of performance.

Which combination of resources will meet these requirements MOST cost-effectively? (Choose two.)

A. Use Hadoop Distributed File System (HDFS) as a persistent data store.

B. Use Amazon S3 as a persistent data store.

C. Use x86-based instances for core nodes and task nodes.

D. Use Graviton instances for core nodes and task nodes.

E. Use Spot Instances for all primary nodes.

Answer: BD

Explanation:

The best combination of resources to meet the requirements of high reliability, cost-optimization, and performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.

Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics¹. Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication¹. Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets². EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS².

Graviton instances are powered by Arm-based AWS Graviton² processors that deliver up to 40% better price performance over comparable current generation x86-based instances³. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open-source databases³. Graviton instances are compatible with Amazon EMR, and can be used for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances.

Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. References:

? Amazon S3 - Cloud Object Storage

? EMR File System (EMRFS)

? AWS Graviton² Processor-Powered Amazon EC2 Instances

? [Plan and Configure EC2 Instances]

? [Amazon EC2 Spot Instances]

? [Best Practices for Amazon EMR]

NEW QUESTION 83

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 84

A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption. The company must also use encryption keys that only specific employees can access. Which solution will meet these requirements with the LEAST effort?

- A. Use an AWS CloudHSM cluster to store the encryption key
- B. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the object
- C. Deploy an IAM policy that restricts access to the CloudHSM cluster.
- D. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer informatio
- E. Restrict access to the keys that encrypt the objects.
- F. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer informatio
- G. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.
- H. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer informatio
- I. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Answer: C

Explanation:

Option C is the best solution to meet the requirements with the least effort because server-side encryption with AWS KMS keys (SSE-KMS) is a feature that allows you to encrypt data at rest in Amazon S3 using keys managed by AWS Key Management Service (AWS KMS). AWS KMS is a fully managed service that enables you to create and manage encryption keys for your AWS services and applications. AWS KMS also allows you to define granular access policies for your keys, such as who can use them to encrypt and decrypt data, and under what conditions. By using SSE-KMS, you can protect your S3 objects by using encryption keys that only specific employees can access, without having to manage the encryption and decryption process yourself.

Option A is not a good solution because it involves using AWS CloudHSM, which is a service that provides hardware security modules (HSMs) in the AWS Cloud. AWS CloudHSM allows you to generate and use your own encryption keys on dedicated hardware that is compliant with various standards and regulations. However, AWS CloudHSM is not a fully managed service and requires more effort to set up and maintain than AWS KMS. Moreover, AWS CloudHSM does not integrate with Amazon S3, so you have to configure the process that writes to S3 to make calls to CloudHSM to encrypt and decrypt the objects, which adds complexity and latency to the data protection process. Option B is not a good solution because it involves using server-side encryption with customer-provided keys (SSE-C), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that you provide and manage yourself. SSE-C requires you to send your encryption key along with each request to upload or retrieve an object. However, SSE-C does not provide any mechanism to restrict access to the keys that encrypt the objects, so you have to implement your own key management and access control system, which adds more effort and risk to the data protection process.

Option D is not a good solution because it involves using server-side encryption with Amazon S3 managed keys (SSE-S3), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that are managed by Amazon S3. SSE-S3 automatically encrypts and decrypts your objects as they are uploaded and downloaded from S3. However, SSE-S3 does not allow you to control who can access the encryption keys or under what conditions. SSE-S3 uses a single encryption key for each S3 bucket, which is shared by all users who have access to the bucket. This means that you cannot restrict access to the keys that encrypt the objects by specific employees, which does not meet the requirements.

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide
- ? Protecting Data Using Server-Side Encryption with AWS KMS–Managed Encryption Keys (SSE-KMS) - Amazon Simple Storage Service
- ? What is AWS Key Management Service? - AWS Key Management Service
- ? What is AWS CloudHSM? - AWS CloudHSM
- ? Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C) - Amazon Simple Storage Service
- ? Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) - Amazon Simple Storage Service

NEW QUESTION 85

.....

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](#)