



Amazon

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)

About ExamBible

Your Partner of IT Exam

Found in 1998

ExamBible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, ExamBible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

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

Which solution will meet these requirements?

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

Answer: B

Explanation:

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

? Athena Workgroups

? IAM policies for accessing workgroups

? Workgroup example policies

NEW QUESTION 3

A media company uses software as a service (SaaS) applications to gather data by using third-party tools. The company needs to store the data in an Amazon S3 bucket. The company will use Amazon Redshift to perform analytics based on the data.

Which AWS service or feature will meet these requirements with the LEAST operational overhead?

- A. Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- B. Amazon AppFlow
- C. AWS Glue Data Catalog
- D. Amazon Kinesis

Answer: B

Explanation:

Amazon AppFlow is a fully managed integration service that enables you to securely transfer data between SaaS applications and AWS services like Amazon S3 and Amazon Redshift. Amazon AppFlow supports many SaaS applications as data sources and targets, and allows you to configure data flows with a few clicks. Amazon AppFlow also provides features such as data transformation, filtering, validation, and encryption to prepare and protect your data. Amazon AppFlow meets the requirements of the media company with the least operational overhead, as it eliminates the need to write code, manage infrastructure, or monitor data pipelines. References:

? Amazon AppFlow

? Amazon AppFlow | SaaS Integrations List

? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

NEW QUESTION 4

A company uses an Amazon QuickSight dashboard to monitor usage of one of the company's applications. The company uses AWS Glue jobs to process data for the dashboard. The company stores the data in a single Amazon S3 bucket. The company adds new data every day.

A data engineer discovers that dashboard queries are becoming slower over time. The data engineer determines that the root cause of the slowing queries is long-running AWS Glue jobs.

Which actions should the data engineer take to improve the performance of the AWS Glue jobs? (Choose two.)

- A. Partition the data that is in the S3 bucket
- B. Organize the data by year, month, and day.
- C. Increase the AWS Glue instance size by scaling up the worker type.
- D. Convert the AWS Glue schema to the DynamicFrame schema class.
- E. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.
- F. Modify the 1AM role that grants access to AWS glue to grant access to all S3 features.

Answer: AB

Explanation:

Partitioning the data in the S3 bucket can improve the performance of AWS Glue jobs by reducing the amount of data that needs to be scanned and processed. By organizing the data by year, month, and day, the AWS Glue job can use partition pruning to filter out irrelevant data and only read the data that matches the query criteria. This can speed up the data processing and reduce the cost of running the AWS Glue job. Increasing the AWS Glue instance size by scaling up the worker type can also improve the performance of AWS Glue jobs by providing more memory and CPU resources for the Spark execution engine. This can help the AWS Glue job handle larger data sets and complex transformations more efficiently. The other options are either incorrect or irrelevant, as they do not affect the performance of the AWS Glue jobs. Converting the AWS Glue schema to the DynamicFrame schema class does not improve the performance, but rather provides additional functionality and flexibility for data manipulation. Adjusting the AWS Glue job scheduling frequency does not improve the performance, but rather reduces the frequency of data updates. Modifying the IAM role that grants access to AWS Glue does not improve the performance, but rather affects the security and permissions of the AWS Glue service. References:

? Optimising Glue Scripts for Efficient Data Processing: Part 1 (Section: Partitioning Data in S3)

? Best practices to optimize cost and performance for AWS Glue streaming ETL jobs (Section: Development tools)

? Monitoring with AWS Glue job run insights (Section: Requirements)

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 133)

NEW QUESTION 5

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

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

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

Answer: A

Explanation:

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

AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores¹. 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. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components³. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python⁴. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.

Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs¹. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.

Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an

Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster⁵. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.

Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes⁶. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data.

Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning⁷. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option B. References:

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

NEW QUESTION 6

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 classes
- B. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes
- 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 infrequently
- 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-Tiering
- G. Activate the Deep Archive Access tier.
- H. Use S3 Intelligent-Tiering
- 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 7

A company's data engineer needs to optimize the performance of table SQL queries. The company stores data in an Amazon Redshift cluster. The data engineer cannot increase the size of the cluster because of budget constraints.

The company stores the data in multiple tables and loads the data by using the EVEN distribution style. Some tables are hundreds of gigabytes in size. Other tables are less than 10 MB in size.

Which solution will meet these requirements?

- A. Keep using the EVEN distribution style for all table
- B. Specify primary and foreign keys for all tables.
- C. Use the ALL distribution style for large table
- D. Specify primary and foreign keys for all tables.
- E. Use the ALL distribution style for rarely updated small table
- F. Specify primary and foreign keys for all tables.
- G. Specify a combination of distribution, sort, and partition keys for all tables.

Answer: C

Explanation:

This solution meets the requirements of optimizing the performance of table SQL queries without increasing the size of the cluster. By using the ALL distribution style for rarely updated small tables, you can ensure that the entire table is copied to every node in the cluster, which eliminates the need for data redistribution during joins. This can improve query performance significantly, especially for frequently joined dimension tables. However, using the ALL distribution style also increases the storage space and the load time, so it is only suitable for small tables that are not updated frequently or extensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. References:

- ? Choose the best distribution style
- ? Distribution styles
- ? Working with data distribution styles

NEW QUESTION 8

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 data
- 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 Serverless
- 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 9

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 10

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.

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

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

Answer: C

Explanation:

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

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

NEW QUESTION 10

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 15

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.

The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata.

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

- A. Use Amazon Aurora as the data catalog
- B. Create AWS Lambda functions that will connect to the data catalog
- C. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog
- D. Schedule the Lambda functions to run periodically.
- E. Use the AWS Glue Data Catalog as the central metadata repository
- F. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata change
- G. Schedule the crawlers to run periodically to update the metadata catalog.

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

Answer: B

Explanation:

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions¹².

The other options are not optimal for the following reasons:

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

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

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

References:

? 1: AWS Glue Data Catalog

? 2: AWS Glue Crawlers

? : Amazon Aurora

? : AWS Lambda

? : Amazon DynamoDB

NEW QUESTION 16

A company uses Amazon Athena to run SQL queries for extract, transform, and load (ETL) tasks by using Create Table As Select (CTAS). The company must use Apache Spark instead of SQL to generate analytics.

Which solution will give the company the ability to use Spark to access Athena?

- A. Athena query settings
- B. Athena workgroup
- C. Athena data source
- D. Athena query editor

Answer: C

Explanation:

Athena data source is a solution that allows you to use Spark to access Athena by using the Athena JDBC driver and the Spark SQL interface. You can use the Athena data source to create Spark DataFrames from Athena tables, run SQL queries on the DataFrames, and write the results back to Athena. The Athena data source supports various data formats, such as CSV, JSON, ORC, and Parquet, and also supports partitioned and bucketed tables. The Athena data source is a cost-effective and scalable way to use Spark to access Athena, as it does not require any additional infrastructure or services, and you only pay for the data scanned by Athena.

The other options are not solutions that give the company the ability to use Spark to access Athena. Option A, Athena query settings, is a feature that allows you to configure various parameters for your Athena queries, such as the output location, the encryption settings, the query timeout, and the workgroup. Option B, Athena workgroup, is a feature that allows you to isolate and manage your Athena queries and resources, such as the query history, the query notifications, the query concurrency, and the query cost. Option D, Athena query editor, is a feature that allows you to write and run SQL queries on Athena using the web console or the API. None of these options enable you to use Spark instead of SQL to generate analytics on Athena. References:

? Using Apache Spark in Amazon Athena

? Athena JDBC Driver

? Spark SQL

? Athena query settings

? [Athena workgroups]

? [Athena query editor]

NEW QUESTION 19

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 made^{6,7}. 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 24

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 26

A company uses Amazon RDS for MySQL as the database for a critical application. The database workload is mostly writes, with a small number of reads.

A data engineer notices that the CPU utilization of the DB instance is very high. The high CPU utilization is slowing down the application. The data engineer must reduce the CPU utilization of the DB Instance.

Which actions should the data engineer take to meet this requirement? (Choose two.)

- A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization
- B. Optimize the problematic queries.
- C. Modify the database schema to include additional tables and indexes.
- D. Reboot the RDS DB instance once each week.
- E. Upgrade to a larger instance size.
- F. Implement caching to reduce the database query load.

Answer: AE

Explanation:

Amazon RDS is a fully managed service that provides relational databases in the cloud. Amazon RDS for MySQL is one of the supported database engines that you can use to run your applications. Amazon RDS provides various features and tools to monitor and optimize the performance of your DB instances, such as Performance Insights, Enhanced Monitoring, CloudWatch metrics and alarms, etc.

Using the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization and optimizing the problematic queries will help reduce the CPU utilization of the DB instance. Performance Insights is a feature that allows you to analyze the load on your DB instance and determine what is causing performance issues. Performance Insights collects, analyzes, and displays database performance data using an interactive dashboard. You can use Performance Insights to identify the top SQL statements, hosts, users, or processes that are consuming the most CPU resources. You can also drill down into the details of each query and see the execution plan, wait events, locks, etc. By using Performance Insights, you can pinpoint the root cause of the high CPU utilization and optimize the queries accordingly. For example, you can rewrite the queries to make them more efficient, add or remove indexes, use prepared statements, etc. Implementing caching to reduce the database query load will also help reduce the CPU utilization of the DB instance. Caching is a technique that allows you to store frequently accessed data in a fast and scalable storage layer, such as Amazon ElastiCache. By using caching, you can reduce the number of requests that hit your database, which in turn reduces the CPU load on your DB instance. Caching also improves the performance and availability of your application, as it reduces the latency and increases the throughput of your data access. You can use caching for various scenarios, such as storing session data, user preferences, application configuration, etc. You can also use caching for read-heavy workloads, such as displaying product details, recommendations, reviews, etc.

The other options are not as effective as using Performance Insights and caching. Modifying the database schema to include additional tables and indexes may or may not improve the CPU utilization, depending on the nature of the workload and the queries. Adding more tables and indexes may increase the complexity and overhead of the database, which may negatively affect the performance. Rebooting the RDS DB instance once each week will not reduce the CPU utilization, as it will not address the underlying cause of the high CPU load. Rebooting may also cause downtime and disruption to your application. Upgrading to a larger instance size may reduce the CPU utilization, but it will also increase the cost and complexity of your solution. Upgrading may also not be necessary if you can optimize the queries and reduce the database load by using caching. References:

? Amazon RDS

? Performance Insights

? Amazon ElastiCache

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 3: Data Storage and Management, Section 3.1: Amazon RDS

NEW QUESTION 31

A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The

company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations. Which combination of AWS services will implement a data mesh? (Choose two.)

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

Answer: BE

Explanation:

A data mesh is an architectural framework that organizes data into domains and treats data as products that are owned and offered for consumption by different teams¹. A data mesh requires a centralized layer for data governance and access control, as well as a distributed layer for data storage and analysis. AWS Glue can provide data catalogs and ETL operations for the data mesh, but it cannot provide data governance and access control by itself². Therefore, the company needs to use another AWS service for this purpose. AWS Lake Formation is a service that allows you to create, secure, and manage data lakes on AWS³. It integrates with AWS Glue and other AWS services to provide centralized data governance and access control for the data mesh. Therefore, option E is correct. For data storage and analysis, the company can choose from different AWS services depending on their needs and preferences. However, one of the benefits of a data mesh is that it enables data to be stored and processed in a decoupled and scalable way¹. Therefore, using serverless or managed services that can handle large volumes and varieties of data is preferable. Amazon S3 is a highly scalable, durable, and secure object storage service that can store any type of data. Amazon Athena is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Therefore, option B is a good choice for data storage and analysis in a data mesh. Option A, C, and D are not optimal because they either use relational databases that are not suitable for storing diverse and unstructured data, or they require more management and provisioning than serverless services. References:

- ? 1: What is a Data Mesh? - Data Mesh Architecture Explained - AWS
- ? 2: AWS Glue - Developer Guide
- ? 3: AWS Lake Formation - Features
- ? [4]: Design a data mesh architecture using AWS Lake Formation and AWS Glue
- ? [5]: Amazon S3 - Features
- ? [6]: Amazon Athena - Features

NEW QUESTION 33

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

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

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

Answer: A

Explanation:

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

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

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

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

References:

- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.2: S3 Event Notifications and Lambda Functions, Pages 67-69
- ? Building Batch Data Analytics Solutions on AWS, Module 4: Data Transformation, Lesson 4.2: AWS Lambda, Pages 4-8
- ? AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon S3, Pages 1-5

NEW QUESTION 38

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 EM
- C. Migrate the existing on-premises Hive metastore into Amazon EM
- 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 EM
- F. Migrate the existing on-premises Hive metastore into Amazon EM
- G. Use Amazon Aurora MySQL to store the company's data catalog.

- H. Configure a new Hive metastore in Amazon EM
- I. Migrate the existing on-premises Hive metastore into Amazon EM
- 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 41

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

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

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

Answer: D

Explanation:

AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets¹.

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

? It does not require you to write any code, as DataBrew provides a graphical user

interface that lets you explore, transform, and visualize your data. You can use DataBrew to concatenate the columns that contain customer first names and last names, and then use the COUNT_DISTINCT aggregate function to count the number of unique values in the resulting column².

? It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes¹.

? It does not require you to create or update any AWS Glue Data Catalog entries, as

DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources³.

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

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

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

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

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

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

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

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

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

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

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

References:

? 1: AWS Glue DataBrew - Features

? 2: Working with recipes - AWS Glue DataBrew

? 3: Working with data sources and data targets - AWS Glue DataBrew

? [4]: AWS Glue notebooks - AWS Glue

? [5]: Development endpoints - AWS Glue

? [6]: Populating the AWS Glue Data Catalog - AWS Glue

? [7]: Crawlers - AWS Glue

- ? [8]: Amazon Athena - Features
- ? [9]: Amazon EMR Serverless - Features
- ? [10]: Creating an Amazon EMR Serverless cluster - Amazon EMR
- ? [11]: Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

NEW QUESTION 43

A company has used an Amazon Redshift table that is named Orders for 6 months. The company performs weekly updates and deletes on the table. The table has an interleaved sort key on a column that contains AWS Regions. The company wants to reclaim disk space so that the company will not run out of storage space. The company also wants to analyze the sort key column. Which Amazon Redshift command will meet these requirements?

- A. VACUUM FULL Orders
- B. VACUUM DELETE ONLY Orders
- C. VACUUM REINDEX Orders
- D. VACUUM SORT ONLY Orders

Answer: C

Explanation:

Amazon Redshift is a fully managed, petabyte-scale data warehouse service that enables fast and cost-effective analysis of large volumes of data. Amazon Redshift uses columnar storage, compression, and zone maps to optimize the storage and performance of data. However, over time, as data is inserted, updated, or deleted, the physical storage of data can become fragmented, resulting in wasted disk space and degraded query performance. To address this issue, Amazon Redshift provides the VACUUM command, which reclaims disk space and resorts rows in either a specified table or all tables in the current schema¹.

The VACUUM command has four options: FULL, DELETE ONLY, SORT ONLY, and REINDEX. The option that best meets the requirements of the question is VACUUM REINDEX, which re-sorts the rows in a table that has an interleaved sort key and rewrites the table to a new location on disk. An interleaved sort key is a type of sort key that gives equal weight to each column in the sort key, and stores the rows in a way that optimizes the performance of queries that filter by multiple columns in the sort key. However, as data is added or changed, the interleaved sort order can become skewed, resulting in suboptimal query performance. The VACUUM REINDEX option restores the optimal interleaved sort order and reclaims disk space by removing deleted rows. This option also analyzes the sort key column and updates the table statistics, which are used by the query optimizer to generate the most efficient query execution plan^{2,3}.

The other options are not optimal for the following reasons:

? A. VACUUM FULL Orders. This option reclaims disk space by removing deleted rows and resorts the entire table. However, this option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option is the most resource-intensive and time-consuming, as it rewrites the entire table to a new location on disk.

? B. VACUUM DELETE ONLY Orders. This option reclaims disk space by removing deleted rows, but does not resort the table. This option is not suitable for tables that have any sort key, as it does not improve the query performance by restoring the sort order. Moreover, this option does not analyze the sort key column and update the table statistics.

? D. VACUUM SORT ONLY Orders. This option resorts the entire table, but does not reclaim disk space by removing deleted rows. This option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option does not analyze the sort key column and update the table statistics.

References:

- ? 1: Amazon Redshift VACUUM
- ? 2: Amazon Redshift Interleaved Sorting
- ? 3: Amazon Redshift ANALYZE

NEW QUESTION 47

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform. The company wants to minimize the effort and time required to incorporate third-party datasets. Which solution will meet these requirements with the LEAST operational overhead?

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

Answer: A

Explanation:

AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed^{1,2}.

The other options are not optimal for the following reasons:

? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead³.

? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data.

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

References:

- ? 1: AWS Data Exchange User Guide
- ? 2: AWS Data Exchange FAQs

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

NEW QUESTION 51

A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long-running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.

Which table views should the data engineer use to meet this requirement?

- A. STL USAGE CONTROL
- B. STL ALERT EVENT LOG
- C. STL QUERY METRICS
- D. STL PLAN INFO

Answer: B

Explanation:

The STL ALERT EVENT LOG table view records anomalies when the query optimizer identifies conditions that might indicate performance issues. These conditions include skewed data distribution, missing statistics, nested loop joins, and broadcasted data. The STL ALERT EVENT LOG table view can help the data engineer to identify and troubleshoot the root causes of performance issues and optimize the query execution plan. The other table views are not relevant for this requirement. STL USAGE CONTROL records the usage limits and quotas for Amazon Redshift resources. STL QUERY METRICS records the execution time and resource consumption of queries. STL PLAN INFO records the query execution plan and the steps involved in each query. References:

- ? STL ALERT EVENT LOG
- ? System Tables and Views
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 52

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 56

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 60

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 62

A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.

A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

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

- A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day
- B. Use the query result reuse feature of Amazon Athena for the SQL queries.
- C. Add an Amazon ElastiCache cluster between the BI application and Athena.
- D. Change the format of the files that are in the dataset to Apache Parquet.

Answer: B

Explanation:

The best solution to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs is to use the query result reuse feature of Amazon Athena for the SQL queries. This feature allows you to run the same query multiple times without incurring additional charges, as long as the underlying data has not changed and the query results are still in the query result location in Amazon S3. This feature is useful for scenarios where you have a petabyte-scale dataset that is updated infrequently, such as once a day, and you have a BI application that runs the same queries repeatedly, such as every hour. By using

the query result reuse feature, you can reduce the amount of data scanned by your queries and save on the cost of running Athena. You can enable or disable this feature at the workgroup level or at the individual query level¹.

Option A is not the best solution, as configuring an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon S3 Lifecycle policies are rules that you can define to automatically transition objects between different storage classes based on specified criteria, such as the age of the object². S3 Glacier Deep Archive is the lowest-cost storage class in Amazon S3, designed

for long-term data archiving that is accessed once or twice in a year³. While moving data to S3 Glacier Deep Archive can reduce the storage cost, it would also increase the retrieval cost and latency, as it takes up to 12 hours to restore the data from S3 Glacier Deep Archive³. Moreover, Athena does not support querying data that is in S3 Glacier or S3 Glacier Deep Archive storage classes⁴. Therefore, using this option would not meet the requirements of running on-demand SQL queries on the dataset.

Option C is not the best solution, as adding an Amazon ElastiCache cluster between the BI application and Athena would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon ElastiCache is a service that offers fully managed in-memory data stores, such as Redis and Memcached, that can improve the performance and scalability of web applications by caching frequently accessed data. While using ElastiCache can reduce the latency and load on the BI application, it would not reduce the amount of data scanned by Athena, which is the main factor that determines the cost of running Athena. Moreover, using ElastiCache would introduce additional infrastructure costs and operational overhead, as you would have to provision, manage, and scale the ElastiCache cluster, and integrate it with the BI application and Athena. Option D is not the best solution, as changing the format of the files that are in the dataset to Apache Parquet would not cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs, but rather increase the complexity. 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, changing the format of the files that are in the dataset to Apache Parquet would require additional processing and transformation steps, such as using AWS Glue or Amazon EMR to convert the files from their original format to Parquet, and storing the converted files in a separate location in Amazon S3. This would increase the complexity and the operational overhead of the data pipeline, and also incur additional costs for using AWS Glue or Amazon EMR. References:

- ? Query result reuse
- ? Amazon S3 Lifecycle
- ? S3 Glacier Deep Archive
- ? Storage classes supported by Athena
- ? [What is Amazon ElastiCache?]
- ? [Amazon Athena pricing]
- ? [Columnar Storage Formats]
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 67

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 71

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 dat

- 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 73

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 74

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 77

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 `DataFramedrop_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 81

A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.

A data engineer needs to find a solution to process the streaming data. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse.

The solution must support near real-time analytics of the streaming data and the previous day's data. Which solution will meet these requirements with the LEAST operational overhead?

- A. Load data into Amazon Kinesis Data Firehose
- B. Load the data into Amazon Redshift.
- C. Use the streaming ingestion feature of Amazon Redshift.
- D. Load the data into Amazon S3. Use the `COPY` command to load the data into Amazon Redshift.
- E. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Answer: B

Explanation:

The streaming ingestion feature of Amazon Redshift enables you to ingest data from streaming sources, such as Amazon Kinesis Data Streams, into Amazon Redshift tables in near real-time. You can use the streaming ingestion feature to process the streaming data from the wearable devices, hospital equipment, and patient records. The streaming ingestion feature also supports incremental updates, which means you can append new data or update existing data in the Amazon Redshift tables. This way, you can store the data in an Amazon Redshift Serverless warehouse and support near real-time analytics of the streaming data and the previous day's data. This solution meets the requirements with the least operational overhead, as it does not require any additional services or components to ingest and process the streaming data. The other options are either not feasible or not optimal. Loading data into Amazon Kinesis Data Firehose and then into Amazon Redshift (option A) would introduce additional latency and cost, as well as require additional configuration and management. Loading data into Amazon S3 and then using the `COPY` command to load the data into Amazon Redshift (option C) would also introduce additional latency and cost, as well as require additional storage space and ETL logic. Using the Amazon Aurora zero-ETL integration with Amazon Redshift (option D) would not work, as it requires the data to be stored in Amazon Aurora first, which is not the case for the streaming data from the healthcare company. References:

- ? Using streaming ingestion with Amazon Redshift

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.5: Amazon Redshift Streaming Ingestion

NEW QUESTION 84

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 86

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

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

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

Answer: B

Explanation:

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

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

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

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

References:

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

? S3 Select and Glacier Select - Amazon Simple Storage Service

? AWS Lambda - FAQs

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

NEW QUESTION 87

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

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

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

Answer: D

Explanation:

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

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

NEW QUESTION 92

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 94

A company stores data in a data lake that is in Amazon S3. Some data that the company stores in the data lake contains personally identifiable information (PII). Multiple user groups need to access the raw data. The company must ensure that user groups can access only the PII that they require.

Which solution will meet these requirements with the LEAST effort?

- A. Use Amazon Athena to query the data
- B. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM role
- C. Assign each user to the IAM role that matches the user's PII access requirements.
- D. Use Amazon QuickSight to access the data
- E. Use column-level security features in QuickSight to limit the PII that users can retrieve from Amazon S3 by using Amazon Athena
- F. Define QuickSight access levels based on the PII access requirements of the users.
- G. Build a custom query builder UI that will run Athena queries in the background to access the data
- H. Create user groups in Amazon Cognito
- I. Assign access levels to the user groups based on the PII access requirements of the users.
- J. Create IAM roles that have different levels of granular access
- K. Assign the IAM roles to IAM user group
- L. Use an identity-based policy to assign access levels to user groups at the column level.

Answer: A

Explanation:

Amazon Athena is a serverless, interactive query service that enables you to analyze data in Amazon S3 using standard SQL. AWS Lake Formation is a service that helps you build, secure, and manage data lakes on AWS. You can use AWS Lake Formation to create data filters that define the level of access for different IAM roles based on the columns, rows, or tags of the data. By using Amazon Athena to query the data and AWS Lake Formation to create data filters, the company can meet the requirements of ensuring that user groups can access only the PII that they require with the least effort. The solution is to use Amazon Athena to query the data in the data lake that is in Amazon S3. Then, set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. For example, a data filter can allow a user group to access only the columns that contain the PII that they need, such as name and email address, and deny access to the columns that contain the PII that they do not need, such as phone number and social security number. Finally, assign each user to the IAM role that matches the user's PII access requirements. This way, the user groups can access the data in the data lake securely and efficiently. The other options are either not feasible or not optimal. Using Amazon QuickSight to access the data (option B) would require the company to pay for the QuickSight service and to configure the column-level security features for each user. Building a custom query builder UI that will run Athena queries in the background to access the data (option C) would require the company to develop and maintain the UI and to integrate it with Amazon Cognito. Creating IAM roles that have different levels of granular access (option D) would require the company to manage multiple IAM roles and policies and to ensure that they are aligned with the data schema.

References:

? Amazon Athena

? AWS Lake Formation

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.3: Amazon Athena

NEW QUESTION 95

.....

Relate Links

100% Pass Your AWS-Certified-Data-Engineer-Associate Exam with ExamBible Prep Materials

<https://www.exambible.com/AWS-Certified-Data-Engineer-Associate-exam/>

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - <https://www.exambible.com/>