Vendor: Snowflake

Exam Code:DEA-C01

Exam Name: SnowPro Advanced: Data Engineer Certification Exam

Version: Demo

QUESTION 1

A financial services company stores financial data in Amazon Redshift. A data engineer wants to run real-time queries on the financial data to support a web-based trading application. The data engineer wants to run the queries from within the trading application.

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

A. Establish WebSocket connections to Amazon Redshift.

B. Use the Amazon Redshift Data API.

C. Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.

D. Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Correct Answer: B

Explanation: The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options. Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups. Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups. Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results. References: Using the Amazon Redshift Data API Calling the Data API Amazon Redshift Data API Reference AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

---

QUESTION 2

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. Confiqure 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 objects. Use a SQL SELECT statement in Amazon Athena to query the required column.

Correct 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

---

QUESTION 3

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.

Which solution will meet this requirement with the LEAST effort?

A. Use Apache Airflow to refresh the materialized views.

B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.

C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.

D. Use an AWS Glue workflow to refresh the materialized views.

Correct Answer: C

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

Glue jobs, defining Glue workflows to orchestrate the jobs, and scheduling the workflows using triggers. References: Query editor V2 Working with materialized views Scheduling queries [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

QUESTION 4

A data engineer must orchestrate a data pipeline that consists of one AWS Lambda function and one AWS Glue job. The solution must integrate with AWS services.

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

A. Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.

B. Use an Apache Airflow workflow that is deployed on an Amazon EC2 instance. Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

C. Use an AWS Glue workflow to run the Lambda function and then the AWS Glue job.

D. Use an Apache Airflow workflow that is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

Correct Answer: A

Explanation: AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows. You can use Step Functions to create state machines that define the sequence and logic of the tasks in your workflow. Step Functions supports various types of tasks, such as Lambda functions, AWS Glue jobs, Amazon EMR clusters, Amazon ECS tasks, etc. You can use Step Functions to monitor and troubleshoot your workflows, as well as to handle errors and retries. Using an AWS Step Functions workflow that includes a state machine to run the Lambda function and then the AWS Glue job will meet the requirements with the least management overhead, as it leverages the serverless and managed capabilities of Step Functions. You do not need to write any code to orchestrate the tasks in your workflow, as you can use the Step Functions console or the AWS Serverless Application Model (AWS SAM) to define and deploy your state machine. You also do not need to provision or manage any servers or clusters, as Step Functions scales automatically based on the demand. The other options are not as efficient as using an AWS Step Functions workflow. Using an Apache Airflow workflow that is deployed on an Amazon EC2 instance or on Amazon Elastic Kubernetes Service (Amazon EKS) will require more management overhead, as you will need to provision, configure, and maintain the EC2 instance or the EKS cluster, as well as the Airflow components. You will also need to write and maintain the Airflow DAGs to orchestrate the tasks in your workflow. Using an AWS Glue workflow to run the Lambda function and then the AWS Glue job will not work, as AWS Glue workflows only support AWS Glue jobs and crawlers as tasks, not Lambda functions. References: AWS Step Functions AWS Glue AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.3: AWS Step Functions

---

QUESTION 5

A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption. The company must also use encryption keys that only specific employees can access.

Which solution will meet these requirements with the LEAST effort?

A. Use an AWS CloudHSM cluster to store the encryption keys. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the objects. Deploy an IAM policy that restricts access to the CloudHSM cluster.

B. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer information. Restrict access to the keys that encrypt the objects.

C. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.

D. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Correct Answer: C

Explanation: Option C is the best solution to meet the requirements with the least effort because server-side encryption with AWS KMS keys (SSE-KMS) is a feature that allows you to encrypt data at rest in Amazon S3 using keys managed by AWS Key Management Service (AWS KMS). AWS KMS is a fully managed service that enables you to create and manage encryption keys for your AWS services and applications. AWS KMS also allows you to define granular access policies for your keys, such as who can use them to encrypt and decrypt data, and under what conditions. By using SSE-KMS, you canprotect your S3 objects by using encryption keys that only specific employees can access, without having to manage the encryption and decryption process yourself. Option A is not a good solution because it involves using AWS CloudHSM, which is a service that provides hardware security modules (HSMs) in the AWS Cloud. AWS CloudHSM allows you to generate and use your own encryption keys on dedicated hardware that is compliant with various standards and regulations. However, AWS CloudHSM is not a fully managed service and requires more effort to set up and maintain than AWS KMS. Moreover, AWS CloudHSM does not integrate with Amazon S3, so you have to configure the process that writes to S3 to make calls to CloudHSM to encrypt and decrypt the objects, which adds complexity and latency to the data protection process. Option B is not a good solution because it involves using server-side encryption with customer-provided keys (SSE-C), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that you provide and manage yourself. SSE-C requires you to send your encryption key along with each request to upload or retrieve an object. However, SSE-C does not provide any mechanism to restrict access to the keys that encrypt the objects, so you have to implement your own key management and access control system, which adds more effort and risk to the data protection process. Option D is not a good solution because it involves using server-side encryption with Amazon S3 managed keys (SSE-S3), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that are managed by Amazon S3. SSE-S3 automatically encrypts and decrypts your objects as they are uploaded and downloaded from S3. However, SSE-S3 does not allow you to control who can access the encryption keys or under what conditions. SSE-S3 uses a single encryption key for each S3 bucket, which is shared by all users who have access to the bucket. This means that you cannot restrict access to the keys that encrypt the objects by specific employees, which does not meet the requirements. References: AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide Protecting Data Using Server-Side Encryption with AWS KMS?anaged Encryption Keys (SSE-KMS) - Amazon Simple Storage Service What is AWS Key Management Service? - AWS Key Management Service What is AWS CloudHSM? - AWS CloudHSM Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C) - Amazon Simple Storage Service Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) - Amazon Simple Storage Service

---

QUESTION 6

A company is building an analytics solution. The solution uses Amazon S3 for data lake storage and Amazon Redshift for a data warehouse. The company wants to use Amazon Redshift Spectrum to query the data that is in Amazon S3.

Which actions will provide the FASTEST queries? (Choose two.)

A. Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.

B. Use a columnar storage file format.

C. Partition the data based on the most common query predicates.

D. Split the data into files that are less than 10 KB.

E. Use file formats that are not

Correct Answer: BC

Explanation: Amazon Redshift Spectrum is a feature that allows you to run SQL queries directly against data in Amazon S3, without loading or transforming the data. Redshift Spectrum can query various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset ofcolumns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance. On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance. Therefore, using a columnar storage file format, such as Parquet, will provide faster queries, as it allows Redshift Spectrum to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Additionally, partitioning the data based on the most common query predicates, such as date, time, region, etc., will provide faster queries, as it allows Redshift Spectrum to prune the partitions that do not match the query criteria, reducing the amount of data scanned from S3. Partitioning also improves the performance of joins and aggregations, as it reduces data skew and shuffling. The other options are not as effective as using a columnar storage file format and partitioning the data. Using gzip compression to compress individual files to sizes that are between 1 GB and 5 GB will reduce the data size, but it will not improve the query performance significantly, as gzip is not a splittable compression algorithm and requires decompression before reading. Splitting the data into files that are less than 10 KB will increase the number of files and the metadata overhead, which will degrade the query performance. Using file formats that are not supported by Redshift Spectrum, such as XML, will not work, as Redshift Spectrum will not be able to read or parse the data. References: Amazon Redshift Spectrum Choosing the Right Data Format AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.3: Amazon Redshift Spectrum

---

QUESTION 7

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

Which solution will meet these requirements MOST cost-effectively?

A. Write a custom Python application. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.

B. Write a PySpark ETL script. Host the script on an Amazon EMR cluster.

C. Write an AWS Glue PySpark job. Use Apache Spark to transform the data.

D. Write an AWS Glue Python shell job. Use pandas to transform the data.

Correct Answer: D

Explanation: AWS Glue is a fully managed serverless ETL service that can handle various data sources and formats, including .csv files in Amazon S3. AWS Glue provides two types of jobs: PySpark and Python shell. PySpark jobs use Apache Spark to process large-scale data in parallel, while Python shell jobs use Python scripts to process small-scale data in a single execution environment. For this requirement, a Python shell job is more suitable and cost-effective, as the size of each S3 object is less than 100 MB, which does not require distributed processing. A Python shell job can use pandas, a popular Python library fordata analysis, to transform the .csv data as needed. The other solutions are not optimal or relevant for this requirement. Writing a custom Python application and hosting it on an Amazon EKS cluster would require more effort and resources to set up and manage the Kubernetes environment, as well as to handle the data ingestion and transformation logic. Writing a PySpark ETL script and hosting it on an Amazon EMR cluster would also incur more costs and complexity to provision and configure the EMR cluster, as well as to use Apache Spark for processing small data files. Writing an AWS Glue PySpark job would also be less efficient and economical than a Python shell job, as it would involve unnecessary overhead and charges for using Apache Spark for small data files. References: AWS Glue Working with Python Shell Jobs pandas [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

QUESTION 8

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 tables. Specify primary and foreign keys for all tables.

B. Use the ALL distribution style for large tables. Specify primary and foreign keys for all tables.

C. Use the ALL distribution style for rarely updated small tables. Specify primary and foreign keys for all tables.

D. Specify a combination of distribution, sort, and partition keys for all tables.

Correct 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 orextensively. 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

---

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)

Correct 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

---

QUESTION 10

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

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

Which solution will meet these requirements MOST cost-effectively?

A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.

B. Configure a Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use AWS Glue Data Catalog to store the company\\'s data catalog as an external data catalog.

C. Configure an external Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use Amazon Aurora MySQL to store the company\\'s data catalog.

D. Configure a new Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use the new metastore as the company\\'s data catalog.

Correct 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 highlyscalable, 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

---

QUESTION 11

A company has a frontend ReactJS website that uses Amazon API Gateway to invoke REST APIs. The APIs perform the functionality of the website. A data engineer needs to write a Python script that can be occasionally invoked through API Gateway. The code must return results to API Gateway.

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

A. Deploy a custom Python script on an Amazon Elastic Container Service (Amazon ECS) cluster.

B. Create an AWS Lambda Python function with provisioned concurrency.

C. Deploy a custom Python script that can integrate with API Gateway on Amazon Elastic Kubernetes Service (Amazon EKS).

D. Create an AWS Lambda function. Ensure that the function is warm byscheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by usingmock events.

Correct Answer: B

Explanation: AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. You can use Lambda to create functions that perform custom logic and integrate with other AWS services, such as

API Gateway. Lambda automatically scales your application by running code in response to each trigger. You pay only for the compute time you consume1.

Amazon ECS is a fully managed container orchestration service that allows you to run and scale containerized applications on AWS. You can use ECS to deploy, manage, and scale Docker containers using either Amazon EC2 instances or

AWS Fargate, a serverless compute engine for containers2.

Amazon EKS is a fully managed Kubernetes service that allows you to run Kubernetes clusters on AWS without needing to install, operate, or maintain your own Kubernetes control plane. You can use EKS to deploy, manage, and scale

containerized applications using Kubernetes on AWS3.

The solution that meets the requirements with the least operational overhead is to create an AWS Lambda Python function with provisioned concurrency. This solution has the following advantages:

It does not require you to provision, manage, or scale any servers or clusters, as Lambda handles all the infrastructure for you. This reduces the operational complexity and cost of running your code.

It allows you to write your Python script as a Lambda function and integrate it with API Gateway using a simple configuration. API Gateway can invoke your Lambda function synchronously or asynchronously, and return the results to the

frontend website.

It ensures that your Lambda function is ready to respond to API requests without any cold start delays, by using provisioned concurrency. Provisioned concurrency is a feature that keeps your function initialized and hyper-ready to respond in

double-digit milliseconds. You can specify the number of concurrent executions that you want to provision for your function.

Option A is incorrect because it requires you to deploy a custom Python script on an Amazon ECS cluster. This solution has the following disadvantages:

It requires you to provision, manage, and scale your own ECS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

It requires you to configure your ECS cluster to integrate with API Gateway, either using an Application Load Balancer or a Network Load Balancer. This adds another layer of complexity to your architecture. Option C is incorrect because it

requires you to deploy a custom Python script that can integrate with API Gateway on Amazon EKS. This solution has the following disadvantages:

It requires you to provision, manage, and scale your own EKS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process.

It requires you to configure your EKS cluster to integrate with API Gateway, either using an Application Load Balancer, a Network Load Balancer, or a service of type LoadBalancer. This adds another layer of complexity to your architecture.

Option D is incorrect because it requires you to create an AWS Lambda function and ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events. This

solution has the following disadvantages:

It does not guarantee that your Lambda function will always be warm, as Lambda may scale down your function if it does not receive any requests for a long period of time. This may cause cold start delays when your function is invoked by

API Gateway.

It incurs unnecessary costs, as you pay for the compute time of your Lambda function every time it is invoked by the EventBridge rule, even if it does not perform any useful work1.

References:

1: AWS Lambda - Features

2: Amazon Elastic Container Service - Features

3: Amazon Elastic Kubernetes Service - Features [4]: Building API Gateway REST API with Lambda integration - Amazon API Gateway [5]: Improving latency with Provisioned Concurrency - AWS Lambda [6]: Integrating Amazon ECS with Amazon API Gateway - Amazon Elastic Container Service [7]: Integrating Amazon EKS with Amazon API Gateway - Amazon Elastic Kubernetes Service [8]: Managing concurrency for a Lambda function - AWS Lambda

---

QUESTION 12

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

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

A. Use Amazon EMR and Apache Ranger.

B. Use a Hive metastore on an EMR cluster.

C. Use the AWS Glue Data Catalog.

D. Use a metastore on an Amazon RDS for MySQL DB instance.

Correct Answer: C

Explanation: The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive

metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically

discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such

as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:

Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying AWS Glue Data Catalog as the metastore)

Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)

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

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)