Amazon

Data-Engineer-Associate

AWS Certified Data Engineer Associate (DEA-C01)

QUESTION & ANSWERS

QUESTION: 1

A Cloud Data Engineering Team is implementing a system for real-time data ingestion through an API. The architecture needs to include data transformation before storage. The system must handle large files and store them efficiently post-transformation. The team is focused on using a serverless architecture on AWS, with an emphasis on Infrastructure as Code (IaC) for standardized and repeatable deployments across various environments.

Which combination of actions should the Cloud Data Engineering Team take to implement IaC for serverless deployments of data ingestion and transformation pipelines? (Select THREE)

Option A :

Set up AWS Data Pipeline in the AWS SAM template for data movement and transformation.

Option B :

Use AWS Elastic MapReduce (EMR) for data transformation in Kinesis Firehose.

Option C :

Define an Amazon API Gateway in the AWS SAM template for data ingestion.

Option D :

Configure DynamoDB in the AWS SAM template for storing the transformed data.

Option E :

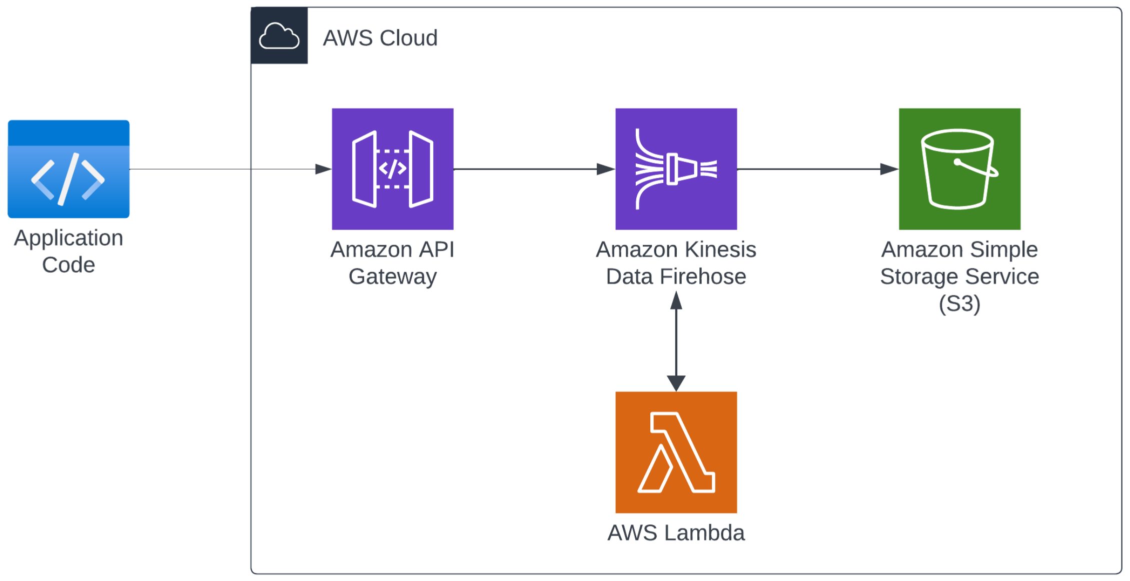
Utilize AWS Serverless Application Model (SAM) to declare AWS Lambda functions integrated with Amazon Kinesis Data Firehose.

Option F :

Configure Amazon S3 bucket creation in the AWS SAM template for storing the transformed data.

Correct Answer: C,E,F

Explanation/Reference:


AWS Cloud
Application Code
Amazon API Gateway
Amazon Kinesis Data Firehose
Amazon Simple Storage Service (S3)
AWS Lambda

Utilize AWS Serverless Application Model (SAM) to declare AWS Lambda functions integrated with Amazon Kinesis Data Firehose.

AWS SAM is a framework for building serverless applications on AWS. It simplifies the process of defining AWS Lambda functions and their associated resources, such as Amazon Kinesis Data Firehose, which is commonly used for data ingestion and streaming.

By using SAM to declare these resources, the team can ensure standardized, repeatable deployments across different environments.

Define an Amazon API Gateway in the AWS SAM template for data ingestion.

Defining an Amazon API Gateway in the AWS SAM template is a common practice for creating serverless data ingestion endpoints.

API Gateway can serve as a front door to manage and route incoming data to other services like AWS Lambda or Kinesis. This approach is aligned with the principles of Infrastructure as Code and ensures a consistent deployment mechanism.

Configure Amazon S3 bucket creation in the AWS SAM template for storing the transformed data.

Amazon S3 is a scalable and secure storage service, often used for storing large volumes of data, including the output of data transformation processes.

By configuring S3 bucket creation in the AWS SAM template, the team can automate the provisioning of storage resources, ensuring that the data storage layer is consistently deployed alongside the processing and ingestion components.

Incorrect Answers:

Configure DynamoDB in the AWS SAM template for storing the transformed data.

While DynamoDB, a NoSQL database service, can be used for storing transformed data, its suitability depends on the nature of the data and access patterns. If the use case involves high throughput and low latency access to data with simple query patterns, DynamoDB can be a good choice.

However, for large-scale analytical workloads, a different data storage solution like Amazon S3 might be more appropriate.

Also, there is no direct integration of Kinesis Firehose and DynamoDB.

Set up AWS Data Pipeline in the AWS SAM template for data movement and transformation.

AWS Data Pipeline is a web service for orchestrating and automating data movement and transformation, but it is not typically associated with serverless architectures and not supported by AWS SAM.

AWS SAM is more focused on deploying serverless applications like Lambda functions, API Gateway, and event sources like Kinesis.

Use AWS Elastic MapReduce (EMR) for data transformation in Kinesis Firehose.

AWS EMR is a managed cluster platform for processing large-scale data using open-source tools such as Apache Hadoop and Spark.

While powerful for certain types of data processing, it is not inherently serverless and does not align well with the serverless deployment model that the team is aiming for as SAM does not support EMR.

Reference:

What is SAM?

QUESTION: 2

As a Data Engineering Consultant, you are implementing a data processing solution using AWS Glue, which leverages Apache Spark under the hood. You need to explain to your team how AWS Glue, using Apache Spark, manages data processing jobs differently than a standalone Apache Spark environment.

Which of the following points would you emphasize as a key difference in the AWS Glue implementation of Spark?

Option A :

Apache Spark in AWS Glue is limited to processing only structured data, such as tables in a database, while standalone Apache Spark can handle both structured and unstructured data formats.

Option B :

In AWS Glue, Spark jobs can only process data stored in Amazon S3, whereas standalone Spark can process data from various sources including HDFS, S3, and local filesystems.

Option C :

AWS Glue requires a deep understanding of Spark internals to optimize and manage jobs, while in a standalone Spark environment, the job optimization and management can be handled without in-depth knowledge of Spark.

Option D :

AWS Glue provides a managed Spark environment where Spark's native UI is replaced with Glue's job monitoring and logging features, providing a more integrated experience within the AWS ecosystem.

Correct Answer: D

Explanation/Reference:

AWS Glue is a fully managed ETL service that simplifies the management of data transformation jobs. AWS Glue provides its own job monitoring and logging features, which are integrated with other AWS services like Amazon CloudWatch. This setup gives a more AWS-centric operational experience compared to using Spark's native UI.

Incorrect Answers:

In AWS Glue, Spark jobs can only process data stored in Amazon S3, whereas standalone Spark can process data from various sources including HDFS, S3, and local filesystems.

While AWS Glue is optimized for Amazon S3 as a data store, it's not limited to S3.

AWS Glue can connect to various data sources, including relational databases and Kafka streams, similar to standalone Apache Spark.

AWS Glue requires a deep understanding of Spark internals to optimize and manage jobs, while in a standalone Spark environment, the job optimization and management can be handled without in-depth knowledge of Spark.

In fact, the opposite is true: AWS Glue abstracts many of the complexities of managing and optimizing Spark jobs, which reduces the need for in-depth knowledge of Spark's internals.

In a standalone Spark environment, administrators and engineers often need to have a deeper understanding of Spark to manually optimize and manage clusters and jobs.

Apache Spark in AWS Glue is limited to processing only structured data, such as tables in a database, while standalone Apache Spark can handle both structured and unstructured data formats.

AWS Glue's Apache Spark implementation can process both structured and unstructured data formats.

This capability is similar to standalone Spark, which can also handle various data formats like JSON, CSV, Parquet, etc. Glue's ETL scripts and transformations are capable of processing a wide range of data formats.

Reference:

How AWS Glue works

QUESTION: 3

A Data Engineering Team at a financial services company is developing a data API to serve real-time, user-specific transactional data from an Amazon RDS for PostgreSQL database to their mobile banking application.

The data is highly dynamic, with frequent reads and writes. The API must offer low latency and high availability, and be capable of scaling automatically to handle peak loads during business hours.

Given these requirements, which architecture should the team implement?

Option A :

Set up an Amazon ECS cluster with Fargate to host a custom-built REST API, connected to the RDS instance. Implement an in-memory caching layer within the API application to cache common queries and reduce direct calls to the database.

Option B :

Use Amazon API Gateway integrated with AWS AppSync, which directly connects to the RDS for PostgreSQL database. Leverage the GraphQL capabilities of AppSync for efficient, tailored queries.

Option C :

Employ Amazon API Gateway with an AWS Step Functions state machine to orchestrate Lambda functions for different query types, reducing the load on RDS by distributing the processing. Use Amazon DynamoDB to cache frequently accessed data.

Option D :

Configure Amazon API Gateway with a serverless AWS Lambda function. The Lambda function should use efficient connection management strategies when accessing the RDS instance. Utilize Amazon ElastiCache to cache frequent queries and reduce database load.

Correct Answer: D