

1) A company is migrating a legacy application to Amazon EC2 instances. The application uses a user name and password that are stored in the source code to connect to a MySQL database. The company will migrate the database to an Amazon RDS for MySQL DB instance. As part of the migration, the company needs to implement a secure way to store and automatically rotate the database credentials.

Which solution will meet these requirements?

- A) Store the database credentials in environment variables in an Amazon Machine Image (AMI). Rotate the credentials by replacing the AMI.
- B) Store the database credentials in AWS Systems Manager Parameter Store. Configure Parameter Store to automatically rotate the credentials.
- C) Store the database credentials in environment variables on the EC2 instances. Rotate the credentials by relaunching the EC2 instances.
- D) Store the database credentials in AWS Secrets Manager. Configure Secrets Manager to automatically rotate the credentials.

2) A developer is creating a web application that must give users the ability to post comments and receive feedback in near real time.

Which solutions will meet these requirements? (Select TWO.)

- A) Create an AWS AppSync schema and corresponding APIs. Use an Amazon DynamoDB table as the data store.
- B) Create a WebSocket API in Amazon API Gateway. Use an AWS Lambda function as the backend. Use an Amazon DynamoDB table as the data store.
- C) Create an AWS Elastic Beanstalk application that is backed by an Amazon RDS database. Configure the application to allow long-lived TCP/IP sockets.
- D) Create a GraphQL endpoint in Amazon API Gateway. Use an Amazon DynamoDB table as the data store.
- E) Establish WebSocket connections to Amazon CloudFront. Use an AWS Lambda function as the CloudFront distribution's origin. Use an Amazon Aurora DB cluster as the data store.

3) A developer is adding sign-up and sign-in functionality to an application. The application must make an API call to a custom analytics solution to log user sign-in events.

Which combination of actions should the developer perform to meet these requirements? (Select TWO.)

- A) Use Amazon Cognito to provide the sign-up and sign-in functionality.
- B) Use AWS Identity and Access Management (IAM) to provide the sign-up and sign-in functionality.
- C) Configure an AWS Config rule to make the API call when a user is authenticated.
- D) Invoke an Amazon API Gateway method to make the API call when a user is authenticated.
- E) Invoke an AWS Lambda function to make the API call when a user is authenticated.

4) A company is using Amazon API Gateway for its REST APIs in an AWS account. A developer wants to allow only IAM users from another AWS account to access the APIs.

Which combination of steps should the developer take to meet these requirements? (Select TWO.)

- A) Create an IAM permission policy. Attach the policy to each IAM user. Set the method authorization type for the APIs to AWS_IAM. Use Signature Version 4 to sign the API requests.
- B) Create an Amazon Cognito user pool. Add each IAM user to the user pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate by using the IAM credentials in Amazon Cognito. Add the ID token to the request headers.
- C) Create an Amazon Cognito identity pool. Add each IAM user to the identity pool. Set the method authorization type for the APIs to COGNITO_USER_POOLS. Authenticate by using the IAM credentials in Amazon Cognito. Add the access token to the request headers.
- D) Create a resource policy for the APIs to allow access for each IAM user only.
- E) Create an Amazon Cognito authorizer for the APIs to allow access for each IAM user only. Set the method authorization type for the APIs to COGNITO_USER_POOLS.

5) A developer is building a new application that transforms text files to .pdf files. A separate application writes the text files to a source Amazon S3 bucket. The new application must read the files as they arrive in Amazon S3 and must convert the files to .pdf files by using an AWS Lambda function. The developer has written an IAM policy to allow access to Amazon S3 and Amazon CloudWatch Logs.

What should the developer do to ensure that the Lambda function has the correct permissions?

- A) Create a Lambda execution role by using AWS Identity and Access Management (IAM). Attach the IAM policy to the role. Assign the Lambda execution role to the Lambda function.
- B) Create a Lambda execution user by using AWS Identity and Access Management (IAM). Attach the IAM policy to the user. Assign the Lambda execution user to the Lambda function.
- C) Create a Lambda execution role by using AWS Identity and Access Management (IAM). Attach the IAM policy to the role. Store the IAM role as an environment variable in the Lambda function.
- D) Create a Lambda execution user by using AWS Identity and Access Management (IAM). Attach the IAM policy to the user. Store the IAM user credentials as environment variables in the Lambda function.

6) A developer is working on an application that stores highly confidential data in a database. The developer must use AWS Key Management Service (AWS KMS) with envelope encryption to protect the data.

How should the developer configure the data encryption to meet these requirements?

- A) Encrypt the data by using a KMS key. Store the encrypted data in the database.
- B) Encrypt the data by using a generated data key. Store the encrypted data in the database.
- C) Encrypt the data by using a generated data key. Store the encrypted data and the data key ID in the database.
- D) Encrypt the data by using a generated data key. Store the encrypted data and the encrypted data key in the database.

7) A developer is adding Amazon ElastiCache for Memcached to a company's existing record storage application. The developer has decided to use lazy loading based on an analysis of common record handling patterns.

Which pseudocode example will correctly implement lazy loading?

- A)

```
record_value = db.query("UPDATE Records SET Details = {1} WHERE ID == {0}",
                        record_key, record_value)
cache.set (record_key, record_value)
```
- B)

```
record_value = cache.get(record_key)
if (record_value == NULL)
    record_value = db.query("SELECT Details FROM Records WHERE ID == {0}",
                        record_key)
cache.set (record_key, record_value)
```
- C)

```
record_value = cache.get (record_key)
db.query("UPDATE Records SET Details = {1} WHERE ID == {0}", record_key,
        record_value)
```
- D)

```
record_value = db.query("SELECT Details FROM Records WHERE ID == {0}",
                        record_key)
if (record_value != NULL)
    cache.set (record_key, record_value)
```

8) A developer is building a web application that uses Amazon API Gateway. The developer wants to maintain different environments for development (dev) and production (prod) workloads. The API will be backed by an AWS Lambda function with two aliases: one for dev and one for prod.

How can the developer maintain these environments with the LEAST amount of configuration?

- A) Create a REST API for each environment. Integrate the APIs with the corresponding dev and prod aliases of the Lambda function. Deploy the APIs to their respective stages. Access the APIs by using the stage URLs.
- B) Create one REST API. Integrate the API with the Lambda function by using a stage variable in place of an alias. Deploy the API to two different stages: dev and prod. Create a stage variable in each stage with different aliases as the values. Access the API by using the different stage URLs.
- C) Create one REST API. Integrate the API with the dev alias of the Lambda function. Deploy the API to the dev environment. Configure a canary release deployment for the prod environment where the canary will integrate with the Lambda prod alias.
- D) Create one REST API. Integrate the API with the prod alias of the Lambda function. Deploy the API to the prod environment. Configure a canary release deployment for the dev environment where the canary will integrate with the Lambda dev alias.

9) A developer wants to track the performance of an application that runs on a fleet of Amazon EC2 instances. The developer wants to view and track statistics, such as the average request latency and the maximum request latency, across the fleet. The developer wants to receive immediate notification if the average response time exceeds a threshold.

Which solution will meet these requirements?

- A) Configure a cron job on each EC2 instance to measure the response time and update a log file stored in an Amazon S3 bucket every minute. Use an Amazon S3 event notification to invoke an AWS Lambda function that reads the log file and writes new entries to an Amazon OpenSearch Service cluster. Visualize the results in OpenSearch Dashboards. Configure OpenSearch Service to send an alert to an Amazon Simple Notification Service (Amazon SNS) topic when the response time exceeds the threshold.
- B) Configure the application to write the response times to the system log. Install and configure the Amazon Inspector agent on the EC2 instances to continually read the logs and send the response times to Amazon EventBridge (Amazon CloudWatch Events). View the metrics graphs in the EventBridge (CloudWatch Events) console. Configure an EventBridge (CloudWatch Events) custom rule to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold.
- C) Configure the application to write the response times to a log file. Install and configure the Amazon CloudWatch agent on the EC2 instances to stream the application log to CloudWatch Logs. Create a metric filter of the response time from the log. View the metrics graphs in the CloudWatch console. Create a CloudWatch alarm to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold.
- D) Install and configure AWS Systems Manager Agent (SSM Agent) on the EC2 instances to monitor the response time and send the response time to Amazon CloudWatch as a custom metric. View the metrics graphs in Amazon QuickSight. Create a CloudWatch alarm to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold.

10) A developer is testing an application locally and has deployed the application to an AWS Lambda function. To avoid exceeding the deployment package size quota, the developer did not include the dependencies in the deployment file. When the developer tests the application remotely, the Lambda function does not run because of missing dependencies.

Which solution will resolve this issue?

- A) Use the Lambda console editor to update the code and include the missing dependencies.
- B) Create an additional .zip file that contains the missing dependencies. Include the .zip file in the original Lambda deployment package.
- C) Add references to the missing dependencies in the Lambda function's environment variables.
- D) Create a layer that contains the missing dependencies. Attach the layer to the Lambda function.

Answers

- 1) D – [AWS Secrets Manager](#) helps protect the credentials that are needed to access databases, applications, services, and other IT resources. With Secrets Manager, you can rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets by making a Secrets Manager API call, eliminating the need to hardcode sensitive information in plaintext. Secrets Manager offers [secret rotation](#) with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB (with MongoDB compatibility).
- 2) A, B – [AWS AppSync](#) simplifies application development by giving you the ability to create a flexible API to securely access, manipulate, and combine data from one or more data sources. AWS AppSync is a managed service that uses GraphQL to help applications get the exact data that they need. You can use AWS AppSync to build scalable applications that require [real-time updates](#) on a range of data sources, including Amazon DynamoDB.
- In [Amazon API Gateway](#), you can [create a WebSocket API](#) as a stateful frontend for an AWS service (such as AWS Lambda or DynamoDB) or for an HTTP endpoint. The WebSocket API invokes the backend based on the content of the messages that the API receives from client applications. Unlike a REST API, which receives and responds to requests, a WebSocket API supports two-way communication between client applications and the backend.
- 3) A, E – [Amazon Cognito](#) adds user sign-up, sign-in, and access control to web and mobile applications. You can also create an AWS Lambda function to make an API call to a custom analytics solution and then invoke that function by using an [Amazon Cognito post authentication trigger](#).
- 4) A, D – A [resource policy](#) can grant API access in one AWS account to users in a different AWS account by using [Signature Version 4](#) (SigV4) protocols.
- 5) A – An AWS Lambda function's [execution role](#) grants the Lambda function permission to access AWS services and resources. You provide this role when you create a function, and Lambda assumes the role when a function is invoked.
- 6) D – [Envelope encryption](#) is the practice of encrypting plaintext data with a data key and then encrypting the data key under another key. You must store the encrypted form of the data key so that you can use the data key to decrypt the encrypted data in the database.
- 7) B – [Lazy loading](#) is a caching strategy in which a record does not load until the record is needed. When you implement lazy loading, the application first checks the cache for a record. If the record is not present, the application retrieves the record from the database and stores the record in the cache.
- 8) B – With deployment stages in Amazon API Gateway, you can manage multiple release stages for each API. You can configure [stage variables](#) so that an API deployment stage can interact with different backend endpoints. You can use API Gateway stage variables to [reference a single AWS Lambda function](#) with multiple versions and aliases.
- 9) C – You can configure the [Amazon CloudWatch agent](#) to stream logs and metrics to CloudWatch. You also can create [metric filters](#) from logs that are stored in CloudWatch Logs.
- 10) D – You can configure an AWS Lambda function to pull in additional code and content in the form of [layers](#). A layer is a .zip file archive that contains libraries, a custom runtime, or other dependencies. With layers, you can use libraries in a Lambda function without needing to include the libraries in a deployment package.