Practice Test #4 (AWS Certified Developer Associate - DVA-C01) - Results

Return to review

Attempt 2

All knowledge areas

All questions

Question 1: **Correct**

You have a popular three-tier web application that is used by users throughout the globe receiving thousands of incoming requests daily. You have AWS Route 53 policies to automatically distribute weighted traffic to the API resources located at URL api.global.com.

What is an alternative way of distributing traffic to a web application?

- CloudFront
- ELB (Correct)
- Auto Scaling
- S3

Explanation
Correct option:

ELB

Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. Route 53 failover policy is similar to an ELB in that when using failover routing, it lets you route traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy.

Incorrect options:

CloudFront - Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds.

S3 - Amazon S3 provides you a way to store and retrieve any amount of data. It cannot be used to distribute traffic.

Auto Scaling - AWS Auto Scaling will automatically scale resources as needed to align to your selected scaling strategy, so you maintain performance and pay only for the resources you need. It cannot be used to distribute traffic.

Reference:

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html

Question 2: **Correct**

The development team at a company wants to insert vendor records into an Amazon DynamoDB table as soon as the vendor uploads a new file into an Amazon S3 bucket.

As a Developer Associate, which set of steps would you recommend to achieve this?

- (●) Create an S3 event to invoke a Lambda function that inserts records into DynamoDB **(Correct)**
- ( ) Set up an event with Amazon CloudWatch Events that will monitor the S3 bucket and then insert the records into DynamoDB
- ( ) Develop a Lambda function that will poll the S3 bucket and then insert the records into DynamoDB
- ( ) Write a cron job that will execute a Lambda function at a scheduled time and insert the records into DynamoDB

Explanation
Correct option:

Create an S3 event to invoke a Lambda function that inserts records into DynamoDB

The Amazon S3 notification feature enables you to receive notifications when certain events happen in your bucket. To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications. You store this configuration in the notification subresource that is associated with a bucket.

Amazon S3 APIs such as PUT, POST, and COPY can create an object. Using these event types, you can enable notification when an object is created using a specific API, or you can use the s3:ObjectCreated:* event type to request notification regardless of the API that was used to create an object.

For the given use-case, you would create an S3 event notification that triggers a Lambda function whenever we have a PUT object operation in the S3 bucket. The Lambda function in turn would execute custom code to inserts records into DynamoDB.

Amazon S3 supports the following destinations where it can publish events:

- **Amazon Simple Notification Service (Amazon SNS) topic**

  Amazon SNS is a flexible, fully managed push messaging service. Using this service, you can push messages to mobile devices or distributed services. With SNS you can publish a message once, and deliver it one or more times. For more information about SNS, see the Amazon SNS product detail page.

- **Amazon Simple Queue Service (Amazon SQS) queue**

  Amazon SQS is a scalable and fully managed message queuing service. You can use SQS to transmit any volume of data without requiring other services to be always available. In your notification configuration, you can request that Amazon S3 publish events to an SQS queue.

  Currently, Standard SQS queue is only allowed as an Amazon S3 event notification destination, whereas FIFO SQS queue is not allowed. For more information about Amazon SQS, see the Amazon SQS product detail page.

- **AWS Lambda**

  AWS Lambda is a compute service that makes it easy for you to build applications that respond quickly to new information. AWS Lambda runs your code in response to events such as image uploads, in-app activity, website clicks, or outputs from connected devices.

  You can use AWS Lambda to extend other AWS services with custom logic, or create your own backend that operates at AWS scale, performance, and security. With AWS Lambda, you can easily create discrete, event-driven applications that execute only when needed and scale automatically from a few requests per day to thousands per second.

  AWS Lambda can run custom code in response to Amazon S3 bucket events. You upload your custom code to AWS Lambda and create what is called a Lambda function. When Amazon S3 detects an event of a specific type (for example, an object created event), it can publish the event to AWS Lambda and invoke your function in Lambda. In response, AWS Lambda executes your function.

Warning

If your notification ends up writing to the bucket that triggers the notification, this could cause an execution loop. For example, if the bucket triggers a Lambda function each time an object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.

For more information and an example of using Amazon S3 notifications with AWS Lambda, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html

Incorrect options:

Write a cron job that will execute a Lambda function at a scheduled time and insert the records into DynamoDB - This is not efficient because there may not be any unprocessed file in the S3 bucket when the cron triggers the Lambda on schedule. So this is not the correct option.

Set up an event with Amazon CloudWatch Events that will monitor the S3 bucket and then insert the records into DynamoDB - The CloudWatch event cannot directly insert records into DynamoDB as it's not a supported target type. The CloudWatch event needs to use something like a Lambda function to insert the records into DynamoDB.

Develop a Lambda function that will poll the S3 bucket and then insert the records into DynamoDB - This is not efficient because there may not be any unprocessed file in the S3 bucket when the Lambda function polls the S3 bucket at a given time interval. So this is not the correct option.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html

Question 3: **Correct**

Your company has been hired to build a resilient mobile voting app for an upcoming music award show that expects to have 5 to 20 million viewers. The mobile voting app will be marketed heavily months in advance so you are expected to handle millions of messages in the system. You are configuring Amazon Simple Queue Service (SQS) queues for your architecture that should receive messages from 20 KB to 200 KB.

Is it possible to send these messages to SQS?

- ○ No, the max message size is 128KB
- ◉ Yes, the max message size is 256KB (Correct)
- ○ No, the max message size is 64KB
- ○ Yes, the max message size is 512KB

Explanation
Correct option:

Yes, the max message size is 256KB

The minimum message size is 1 byte (1 character). The maximum is 262,144 bytes (256 KB).

Q: How do I configure the maximum message size for Amazon SQS?

To configure the maximum message size, use the console or the SetQueueAttributes method to set the MaximumMessageSize attribute. This attribute specifies the number of bytes that an Amazon SQS message can contain. Set this attribute to a value between 1,024 bytes (1 KB), and 262,144 bytes (256 KB). For more information, see Using Amazon SQS Message Attributes in the *Amazon SQS Developer Guide*.

To send messages larger than 256 KB, use the Amazon SQS Extended Client Library for Java. This library lets you send an Amazon SQS message that contains a reference to a message payload in Amazon S3 that can be as large as 2 GB.

Q: What kind of data can I include in a message?

Amazon SQS messages can contain up to 256 KB of text data, including XML, JSON and unformatted text. The following Unicode characters are accepted:

#x9 | #xA | #xD | [#x20 to #xD7FF] | [#xE000 to #xFFFD] | [#x10000 to #x10FFFF]

For more information, see the XML 1.0 Specification.

via - https://aws.amazon.com/sqs/faqs/

Incorrect options:

Yes, the max message size is 512KB - The max size is 256KB

No, the max message size is 128KB - The max size is 256KB

No, the max message size is 64KB - The max size is 256KB

Reference:

https://aws.amazon.com/sqs/faqs/

Question 4: **Correct**

A developer has pushed a Lambda function that pushes data into an RDS MySQL database with the following Python code:

```
def handler(event, context):
    mysql = mysqlclient.connect()
    data = event['data']
    mysql.execute(f"INSERT INTO foo (bar) VALUES (${data});")
    mysql.close()
    return
```

On the first execution, the Lambda function takes 2 seconds to execute. On the second execution and all the subsequent ones, the Lambda function takes 1.9 seconds to execute.

What can be done to improve the execution time of the Lambda function?

- ○ Upgrade the MySQL instance type
- ○ Change the runtime to Node.js
- ○ Increase the Lambda function RAM
- ◉ Move the database connection out of the handler (Correct)

Explanation
Correct option:

Move the database connection out of the handler

Here at every Lambda function execution, the database connection handler will be created, and then closed. These connections steps are expensive in terms of time, and thus should be moved out of the `handler` function so that they are kept in the function execution context, and re-used across function calls. This is what the function should look like in the end:

```
mysql = mysqlclient.connect()

def handler(event, context):
    data = event['data']
    mysql.execute(f"INSERT INTO foo (bar) VALUES (${data});")
    return
```

Incorrect options:

Upgrade the MySQL instance type - The bottleneck here is the MySQL connection object, not the MySQL instance itself.

Change the runtime to Node.js - Re-writing the function in another runtime won't improve the performance.

Increase the Lambda function RAM - While this may help speed-up the Lambda function, as increasing the RAM also increases the CPU allocated to your function, it only makes sense if RAM or CPU is a critical factor in the Lambda function performance. Here, the connection object is at fault.

Reference:

https://docs.aws.amazon.com/lambda/latest/dg/running-lambda-code.html

Question 5: **Correct**

Your company stores confidential data on an Amazon Simple Storage Service (S3) bucket. New security compliance guidelines require that files be stored with server-side encryption. The encryption used must be Advanced Encryption Standard (AES-256) and the company does not want to manage S3 encryption keys.

Which of the following options should you use?

- ◉ SSE-S3 (Correct)
- ○ Client Side Encryption

SSE-C

SSE-KMS

Explanation
Correct option:

SSE-S3

Using Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key employing strong multi-factor encryption. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

Incorrect options:

SSE-C - You manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption when you access your objects.

Client-Side Encryption - You can encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.

SSE-KMS - Similar to SSE-S3 and also provides you with an audit trail of when your key was used and by whom. Additionally, you have the option to create and manage encryption keys yourself.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html

Question 6: **Correct**

After reviewing your monthly AWS bill you notice that the cost of using Amazon SQS has gone up substantially after creating new queues; however, you know that your queue clients do not have a lot of traffic and are receiving empty responses.

Which of the following actions should you take?

Use LongPolling (Correct)

Increase the VisibilityTimeout

Decrease DelaySeconds

Use a FIFO queue

Explanation
Correct option:

Use LongPolling

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

Amazon SQS provides short polling and long polling to receive messages from a queue. By default, queues use short polling. With short polling, Amazon SQS sends the response right away, even if the query found no messages. With long polling, Amazon SQS sends a response after it collects at least one available message, up to the maximum number of messages specified in the request. Amazon SQS sends an empty response only if the polling wait time expires.

Long polling makes it inexpensive to retrieve messages from your Amazon SQS queue as soon as the messages are available. Long polling helps reduce the cost of using Amazon SQS by eliminating the number of empty responses (when there are no messages available for a ReceiveMessage request) and false empty responses (when messages are available but aren't included in a response). When the wait time for the ReceiveMessage API action is greater than 0, long polling is in effect. The maximum long polling wait time is 20 seconds.

Exam Alert:

Please review the differences between Short Polling vs Long Polling:

Amazon SQS short and long polling

PDF | Kindle | RSS

Amazon SQS provides short polling and long polling to receive messages from a queue. By default, queues use short polling.

With *short polling*, the `ReceiveMessage` request queries only a subset of the servers (based on a weighted random distribution) to find messages that are available to include in the response. Amazon SQS sends the response right away, even if the query found no messages.

With *long polling*, the `ReceiveMessage` request queries all of the servers for messages. Amazon SQS sends a response after it collects at least one available message, up to the maximum number of messages specified in the request. Amazon SQS sends an empty response only if the polling wait time expires.

The following sections explain the details of short polling and long polling.

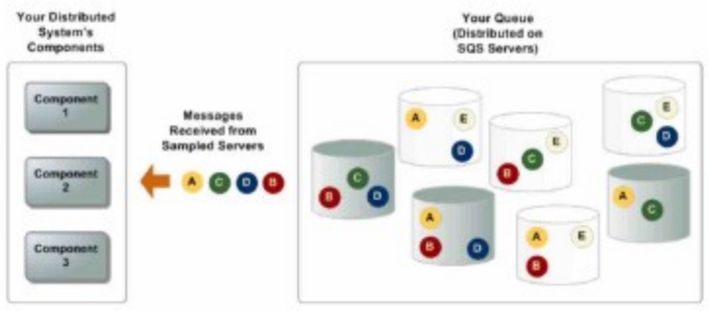
Topics

- Consuming messages using short polling
- Consuming message using long polling
- Differences between long and short polling

Consuming messages using short polling

When you consume messages from a queue using short polling, Amazon SQS samples a subset of its servers (based on a weighted random distribution) and returns messages from only those servers. Thus, a particular `ReceiveMessage` request might not return all of your messages. However, if you have fewer than 1,000 messages in your queue, a subsequent request will return your messages. If you keep consuming from your queues, Amazon SQS samples all of its servers, and you receive all of your messages.

The following diagram shows the short-polling behavior of messages returned from a standard queue after one of your system components makes a receive request. Amazon SQS samples several of its servers (in gray) and returns messages A, C, D, and B from these servers. Message E isn't returned for this request, but is returned for a subsequent request.



via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-short-and-long-polling.html

Consuming message using long polling

When the wait time for the `ReceiveMessage` API action is greater than 0, *long polling* is in effect. The maximum long polling wait time is 20 seconds. Long polling helps reduce the cost of using Amazon SQS by eliminating the number of empty responses (when there are no messages available for a `ReceiveMessage` request) and false empty responses (when messages are available but aren't included in a response). For information about enabling long polling for a new or existing queue using the Amazon SQS console, see the Configuring queue parameters (console). For best practices, see Setting up long polling.

Long polling offers the following benefits:

- Eliminate empty responses by allowing Amazon SQS to wait until a message is available in a queue before sending a response. Unless the connection times out, the response to the `ReceiveMessage` request contains at least one of the available messages, up to the maximum number of messages specified in the `ReceiveMessage` action.
- Eliminate false empty responses by querying all—rather than a subset of—Amazon SQS servers.

> **Note**
>
> You can confirm that a queue is empty when you perform a long poll and the `ApproximateNumberOfMessagesDelayed`, `ApproximateNumberOfMessagesNotVisible`, and `ApproximateNumberOfMessagesVisible` metrics are equal to 0 at least 1 minute after the producers stop sending messages (when the queue metadata reaches eventual consistency). For more information, see Available CloudWatch metrics for Amazon SQS.

- Return messages as soon as they become available.

Differences between long and short polling

Short polling occurs when the `WaitTimeSeconds` parameter of a `ReceiveMessage` request is set to `0` in one of two ways:

- The `ReceiveMessage` call sets `WaitTimeSeconds` to `0`.
- The `ReceiveMessage` call doesn't set `WaitTimeSeconds`, but the queue attribute `ReceiveMessageWaitTimeSeconds` is set to `0`.

via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-short-and-long-polling.html

Incorrect options:

Increase the VisibilityTimeout - Because there is no guarantee that a consumer received a message, the consumer must delete it. To prevent other consumers from processing the message again, Amazon SQS sets a visibility timeout. Visibility timeout will not help with cost reduction.

Use a FIFO queue - FIFO queues are designed to enhance messaging between applications when the order of operations and events has to be enforced. FIFO queues will not help with cost reduction. In fact, they are costlier than standard queues.

Decrease DelaySeconds - This is similar to VisibilityTimeout. The difference is that a message is hidden when it is first added to a queue for DelaySeconds, whereas for visibility timeouts a message is hidden only after it is consumed from the queue. DelaySeconds will not help with cost reduction.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-short-and-long-polling.html

Question 7: **Correct**

A developer is configuring an Application Load Balancer (ALB) to direct traffic to the application's EC2 instances and Lambda functions.

Which of the following characteristics of the ALB can be identified as correct? (Select two)

- [ ] An ALB has three possible target types: Hostname, IP and Lambda
- [x] An ALB has three possible target types: Instance, IP and Lambda **(Correct)**
- [ ] If you specify targets using an instance ID, traffic is routed to instances using any private IP address from one or more network interfaces
- [x] You can not specify publicly routable IP addresses to an ALB **(Correct)**
- [ ] If you specify targets using IP addresses, traffic is routed to instances using the primary private IP address

Explanation
Correct options:

An ALB has three possible target types: Instance, IP and Lambda

When you create a target group, you specify its target type, which determines the type of target you specify when registering targets with this target group. After you create a target group, you cannot change its target type. The following are the possible target types:

1. `Instance` - The targets are specified by instance ID
2. `IP` - The targets are IP addresses
3. `Lambda` - The target is a Lambda function

You can not specify publicly routable IP addresses to an ALB

When the target type is IP, you can specify IP addresses from specific CIDR blocks only. You can't specify publicly routable IP addresses.

Incorrect options:

If you specify targets using an instance ID, traffic is routed to instances using any private IP address from one or more network interfaces - If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address specified in the primary network interface for the instance.

If you specify targets using IP addresses, traffic is routed to instances using the primary private IP address - If you specify targets using IP addresses, you can route traffic to an instance using any private IP address from one or more network interfaces. This enables multiple applications on an instance to use the same port.

An ALB has three possible target types: Hostname, IP and Lambda - This is incorrect, as described in the correct explanation above.

Reference:

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html

Question 8: **Correct**

An IT company uses a blue/green deployment policy to provision new Amazon EC2 instances in an Auto Scaling group behind a new Application Load Balancer for each new application version. The current set up requires the users to log in after every new deployment.

As a Developer Associate, what advice would you give to the company for resolving this issue?

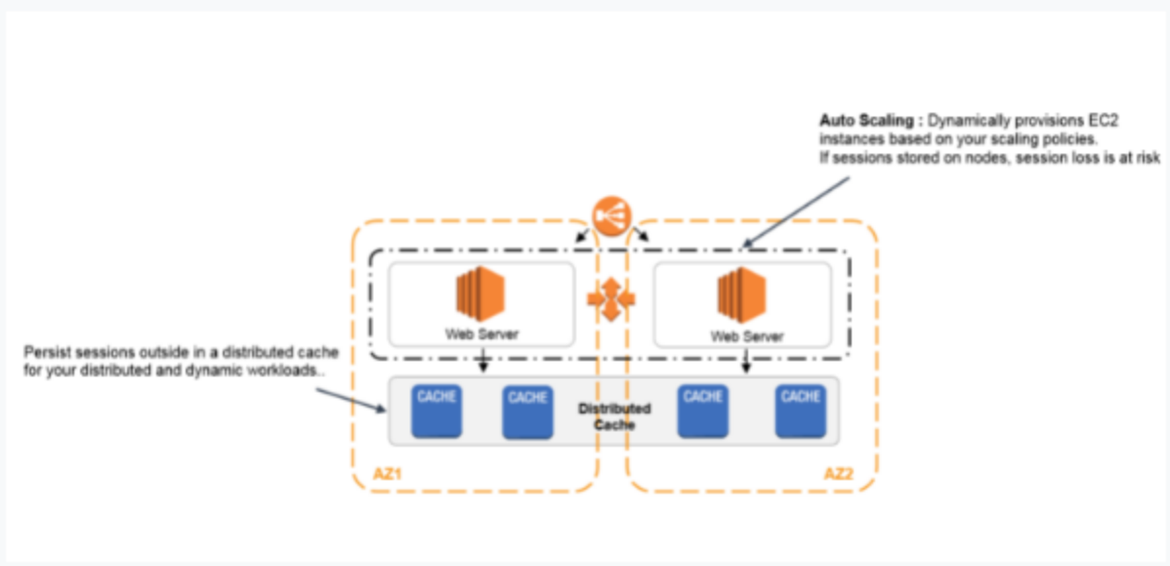
- ○ Enable sticky sessions in the Application Load Balancer
- ○ Use multicast to replicate session information
- ◉ Use ElastiCache to maintain user sessions (Correct)
- ○ Use rolling updates instead of a blue/green deployment

Explanation
Correct option:

Use ElastiCache to maintain user sessions

Amazon ElastiCache allows you to seamlessly set up, run, and scale popular open-Source compatible in-memory data stores in the cloud. Build data-intensive apps or boost the performance of your existing databases by retrieving data from high throughput and low latency in-memory data stores. Amazon ElastiCache is a popular choice for real-time use cases like Caching, Session Stores, Gaming, Geospatial Services, Real-Time Analytics, and Queuing.

To address scalability and to provide a shared data storage for sessions that can be accessed from any individual web server, you can abstract the HTTP sessions from the web servers themselves. A common solution to for this is to leverage an In-Memory Key/Value store such as Redis and Memcached via ElastiCache.



via - https://aws.amazon.com/caching/session-management/

Incorrect options:

Use rolling updates instead of a blue/green deployment - With rolling deployments, Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. It leaves the rest of the instances in the environment running the old version of the application. When processing a batch, Elastic Beanstalk detaches all instances in the batch from the load balancer, deploys the new application version, and then reattaches the instances.

This means that some of the users can experience session disruptions when the instances maintaining the sessions were detached as part of the given batch. So this option is incorrect.

Enable sticky sessions in the Application Load Balancer - As the Application Load Balancer itself is replaced on each new deployment, so maintaining sticky sessions via the Application Load Balancer will not work.

Use multicast to replicate session information - This option has been added as a distractor.

References:

https://aws.amazon.com/caching/session-management/

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html#environments-cfg-rollingdeployments-method

Question 9: **Correct**

You are planning to build a fleet of EBS-optimized EC2 instances to handle the load of your new application. Due to security compliance, your organization wants any secret strings used in the application to be encrypted to prevent exposing values as clear text.

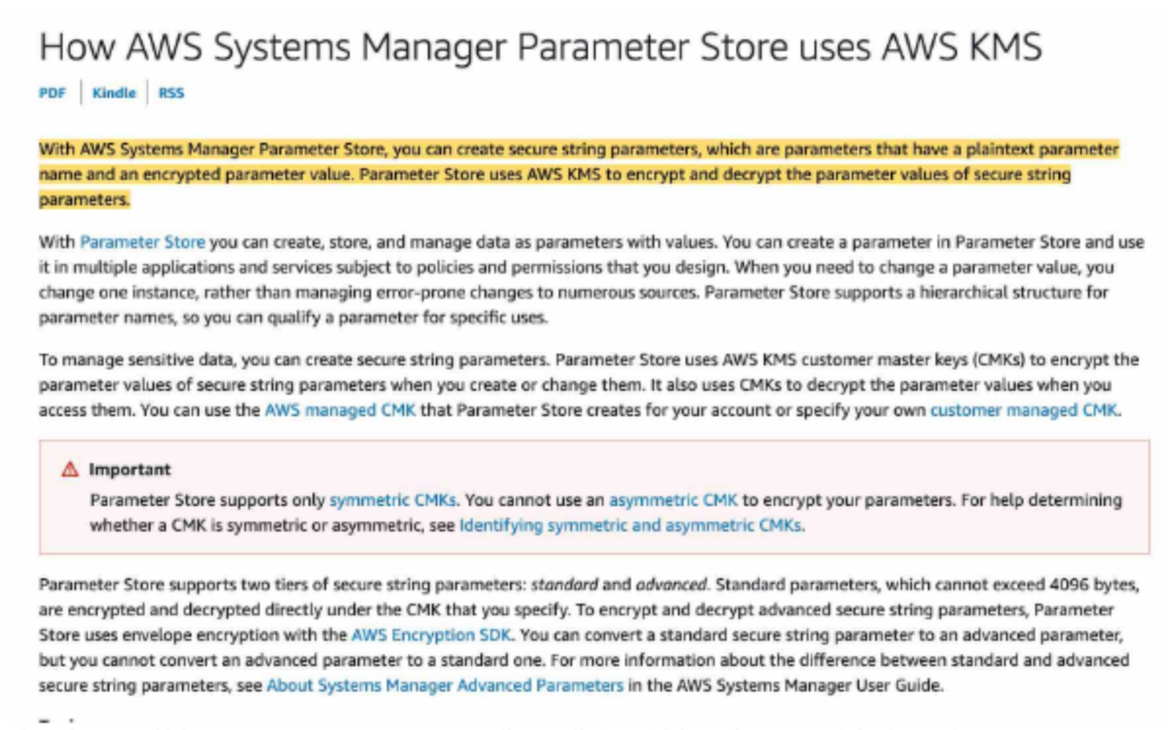
The solution requires that decryption events be audited and API calls to be simple. How can this be achieved? (select two)

- [x] Store the secret as SecureString in SSM Parameter Store (Correct)
- [ ] Audit using SSM Audit Trail
- [ ] Encrypt first with KMS then store in SSM Parameter store
- [x] Audit using CloudTrail (Correct)
- [ ] Store the secret as PlainText in SSM Parameter Store

Explanation
Correct options:

Store the secret as SecureString in SSM Parameter Store

With AWS Systems Manager Parameter Store, you can create SecureString parameters, which are parameters that have a plaintext parameter name and an encrypted parameter value. Parameter Store uses AWS KMS to encrypt and decrypt the parameter values of Secure String parameters. Also, if you are using customer-managed CMKs, you can use IAM policies and key policies to manage to encrypt and decrypt permissions. To retrieve the decrypted value you only need to do one API call.



How AWS Systems Manager Parameter Store uses AWS KMS

PDF | Kindle | RSS

With AWS Systems Manager Parameter Store, you can create secure string parameters, which are parameters that have a plaintext parameter name and an encrypted parameter value. Parameter Store uses AWS KMS to encrypt and decrypt the parameter values of secure string parameters.

With Parameter Store you can create, store, and manage data as parameters with values. You can create a parameter in Parameter Store and use it in multiple applications and services subject to policies and permissions that you design. When you need to change a parameter value, you change one instance, rather than managing error-prone changes to numerous sources. Parameter Store supports a hierarchical structure for parameter names, so you can qualify a parameter for specific uses.

To manage sensitive data, you can create secure string parameters. Parameter Store uses AWS KMS customer master keys (CMKs) to encrypt the parameter values of secure string parameters when you create or change them. It also uses CMKs to decrypt the parameter values when you access them. You can use the AWS managed CMK that Parameter Store creates for your account or specify your own customer managed CMK.

Important
Parameter Store supports only symmetric CMKs. You cannot use an asymmetric CMK to encrypt your parameters. For help determining whether a CMK is symmetric or asymmetric, see Identifying symmetric and asymmetric CMKs.

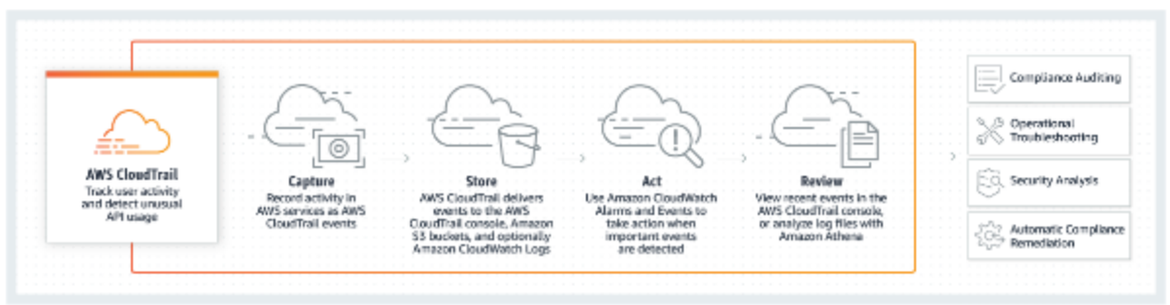
Parameter Store supports two tiers of secure string parameters: *standard* and *advanced*. Standard parameters, which cannot exceed 4096 bytes, are encrypted and decrypted directly under the CMK that you specify. To encrypt and decrypt advanced secure string parameters, Parameter Store uses envelope encryption with the AWS Encryption SDK. You can convert a standard secure string parameter to an advanced parameter, but you cannot convert an advanced parameter to a standard one. For more information about the difference between standard and advanced secure string parameters, see About Systems Manager Advanced Parameters in the AWS Systems Manager User Guide.

via - https://docs.aws.amazon.com/kms/latest/developerguide/services-parameter-store.html

Audit using CloudTrail

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. CloudTrail provides an event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command-line tools, and other AWS services.

CloudTrail will allow you to see all API calls made to SSM and KMS.



Incorrect options:

Encrypt first with KMS then store in SSM Parameter store - This could work but will require two API calls to get the decrypted value instead of one. So this is not the right option.

Store the secret as PlainText in SSM Parameter Store - Plaintext parameters are not secure and shouldn't be used to store secrets.

Audit using SSM Audit Trail - This is a made-up option and has been added as a distractor.

Reference:

https://docs.aws.amazon.com/kms/latest/developerguide/services-parameter-store.html

Question 10: **Incorrect**

You have a Java-based application running on EC2 instances loaded with AWS CodeDeploy agents. You are considering different options for deployment, one is the flexibility that allows for incremental deployment of your new application versions and replaces existing versions in the EC2 instances. The other option is a strategy in which an Auto Scaling group is used to perform a deployment.

Which of the following options will allow you to deploy in this manner? (Select two)

- [ ] In-place Deployment (Correct)
- [ ] Cattle Deployment
- [x] Warm Standby Deployment (Incorrect)
- [ ] Pilot Light Deployment
- [x] Blue/green Deployment (Correct)

Explanation
Correct options:

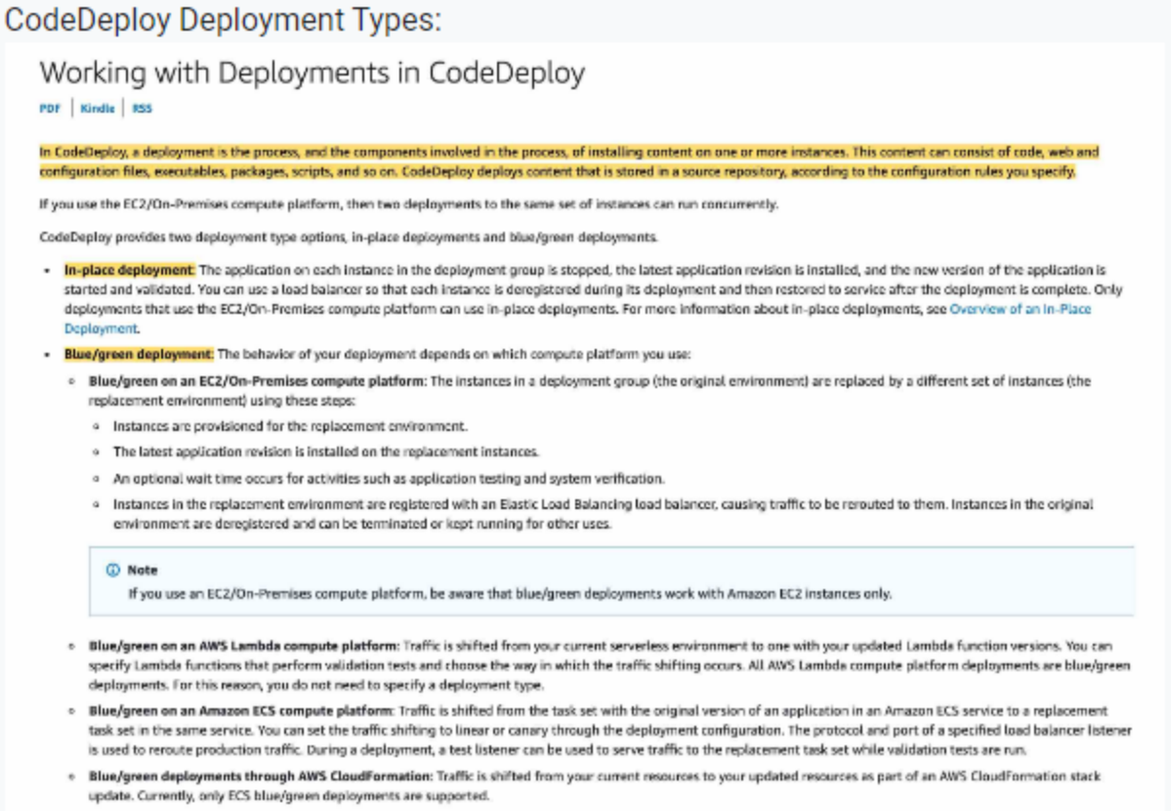
In-place Deployment

The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete.

Blue/green Deployment

With a blue/green deployment, you provision a new set of instances on which CodeDeploy installs the latest version of your application. CodeDeploy then re-routes load balancer traffic from an existing set of instances running the previous version of your application to the new set of instances running the latest version. After traffic is re-routed to the new instances, the existing instances can be terminated.

CodeDeploy Deployment Types:



Working with Deployments in CodeDeploy

PDF | Kindle | RSS

In CodeDeploy, a deployment is the process, and the components involved in the process, of installing content on one or more instances. This content can consist of code, web and configuration files, executables, packages, scripts, and so on. CodeDeploy deploys content that is stored in a source repository, according to the configuration rules you specify.

If you use the EC2/On-Premises compute platform, then two deployments to the same set of instances can run concurrently.

CodeDeploy provides two deployment type options, in-place deployments and blue/green deployments.

- **In-place deployment:** The application on each instance in the deployment group is stopped, the latest application revision is installed, and the new version of the application is started and validated. You can use a load balancer so that each instance is deregistered during its deployment and then restored to service after the deployment is complete. Only deployments that use the EC2/On-Premises compute platform can use in-place deployments. For more information about in-place deployments, see Overview of an In-Place Deployment.
- **Blue/green deployment:** The behavior of your deployment depends on which compute platform you use:
  - **Blue/green on an EC2/On-Premises compute platform:** The instances in a deployment group (the original environment) are replaced by a different set of instances (the replacement environment) using these steps:
    - Instances are provisioned for the replacement environment.
    - The latest application revision is installed on the replacement instances.
    - An optional wait time occurs for activities such as application testing and system verification.
    - Instances in the replacement environment are registered with an Elastic Load Balancing load balancer, causing traffic to be rerouted to them. Instances in the original environment are deregistered and can be terminated or kept running for other uses.

    Note
    If you use an EC2/On-Premises compute platform, be aware that blue/green deployments work with Amazon EC2 instances only.
  - **Blue/green on an AWS Lambda compute platform:** Traffic is shifted from your current serverless environment to one with your updated Lambda function versions. You can specify Lambda functions that perform validation tests and choose the way in which the traffic shifting occurs. All AWS Lambda compute platform deployments are blue/green deployments. For this reason, you do not need to specify a deployment type.
  - **Blue/green on an Amazon ECS compute platform:** Traffic is shifted from the task set with the original version of an application in an Amazon ECS service to a replacement task set in the same service. You can set the traffic shifting to linear or canary through the deployment configuration. The protocol and port of a specified load balancer listener is used to reroute production traffic. During a deployment, a test listener can be used to serve traffic to the replacement task set while validation tests are run.
  - **Blue/green deployments through AWS CloudFormation:** Traffic is shifted from your current resources to your updated resources as part of an AWS CloudFormation stack update. Currently, only ECS blue/green deployments are supported.

via - https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html

Incorrect options:

Cattle Deployment - This is a good option if you have cattle in a farm

Warm Standby Deployment - This is not a valid CodeDeploy deployment option. The term "Warm Standby" is used to describe a Disaster Recovery scenario in which a scaled-down version of a fully functional environment is always running in the cloud.

Pilot Light Deployment - This is not a valid CodeDeploy deployment option. "Pilot Light" is a Disaster Recovery approach where you simply replicate part of your IT structure for a limited set of core services so that the AWS cloud environment seamlessly takes over in the event of a disaster.

References:

https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html

https://aws.amazon.com/blogs/publicsector/rapidly-recover-mission-critical-systems-in-a-disaster/

Question 11: **Correct**

A company wants to add geospatial capabilities to the cache layer, along with query capabilities and an ability to horizontally scale. The company uses Amazon RDS as the database tier.

Which solution is optimal for this use-case?

- ○ Migrate to Amazon DynamoDB to utilize the automatically integrated DynamoDB Accelerator (DAX) along with query capability features
- ◉ Leverage the capabilities offered by ElastiCache for Redis with cluster mode enabled **(Correct)**
- ○ Leverage the capabilities offered by ElastiCache for Redis with cluster mode disabled
- ○ Use CloudFront caching to cater to demands of increasing workloads

Explanation
Correct option:

Leverage the capabilities offered by ElastiCache for Redis with cluster mode enabled

You can use Amazon ElastiCache to accelerate your high volume application workloads by caching your data in-memory providing sub-millisecond data retrieval performance. When used in conjunction with any database including Amazon RDS or Amazon DynamoDB, ElastiCache can alleviate the pressure associated with heavy request loads, increase overall application performance and reduce costs associated with scaling for throughput on other databases.

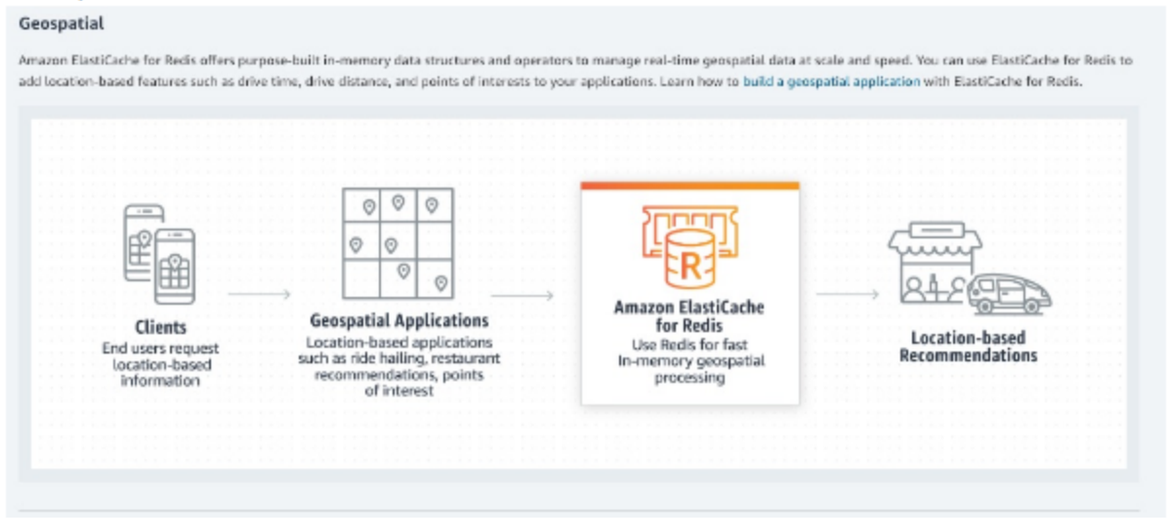
Amazon ElastiCache makes it easy to deploy and manage a highly available and scalable in-memory data store in the cloud. Among the open source in-memory engines available for use with ElastiCache is Redis, which added powerful geospatial capabilities in its newer versions.

You can leverage ElastiCache for Redis with cluster mode enabled to enhance reliability and availability with little change to your existing workload. Cluster Mode comes with the primary benefit of horizontal scaling up and down of your Redis cluster, with almost zero impact on the performance of the cluster.

Enabling Cluster Mode provides a number of additional benefits in scaling your cluster. In short, it allows you to scale in or out the number of shards (horizontal scaling) versus scaling up or down the node type (vertical scaling). This means that Cluster Mode can scale to very large amounts of storage (potentially 100s of terabytes) across up to 90 shards, whereas a single node can only store as much data in memory as the instance type has capacity for.

Cluster Mode also allows for more flexibility when designing new workloads with unknown storage requirements or heavy write activity. In a read-heavy workload, one can scale a single shard by adding read replicas, up to five, but a write-heavy workload can benefit from additional write endpoints when cluster mode is enabled.

Geospatial on Amazon ElastiCache for Redis:



via - https://aws.amazon.com/blogs/database/work-with-cluster-mode-on-amazon-elasticache-for-redis/

Incorrect options:

Leverage the capabilities offered by ElastiCache for Redis with cluster mode disabled - For a production workload, you should consider using a configuration that includes replication to enhance the protection of your data. Also, only vertical scaling is possible when cluster mode is disabled. The use case mentions horizontal scaling as a requirement, hence disabling cluster mode is not an option.

Use CloudFront caching to cater to demands of increasing workloads - One of the purposes of using CloudFront is to reduce the number of requests that your origin server must respond to directly. With CloudFront caching, more objects are served from CloudFront edge locations, which are closer to your users. This reduces the load on your origin server and reduces latency. However, the use case mentions that in-memory caching is needed for enhancing the performance of the application. So, this option is incorrect.

Migrate to Amazon DynamoDB to utilize the automatically integrated DynamoDB Accelerator (DAX) along with query capability features - Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement – from milliseconds to microseconds – even at millions of requests per second. Database migration is a more elaborate effort compared to implementing and optimizing the caching layer.

References:

https://aws.amazon.com/blogs/database/work-with-cluster-mode-on-amazon-elasticache-for-redis/

https://aws.amazon.com/blogs/database/amazon-elasticache-utilizing-redis-geospatial-capabilities/

Question 12: **Correct**

A financial services company with over 10,000 employees has hired you as the new Senior Developer. Initially caching was enabled to reduce the number of calls made to all API endpoints and improve the latency of requests to the company's API Gateway.

For testing purposes, you would like to invalidate caching for the API clients to get the most recent responses. Which of the following should you do?

- (●) Using the Header Cache-Control: max-age=0 (Correct)
- ( ) Using the request parameter ?cache-control-max-age=0
- ( ) Using the Header Bypass-Cache=1
- ( ) Use the Request parameter: ?bypass_cache=1

Explanation
Correct option:

Using the Header Cache-Control: max-age=0

A client of your API can invalidate an existing cache entry and reload it from the integration endpoint for individual requests. The client must send a request that contains the Cache-Control: max-age=0 header. The client receives the response directly from the integration endpoint instead of the cache, provided that the client is authorized to do so. This replaces the existing cache entry with the new response, which is fetched from the integration endpoint.

Invalidate an API Gateway cache entry

A client of your API can invalidate an existing cache entry and reload it from the integration endpoint for individual requests. The client must send a request that contains the `Cache-Control: max-age=0` header. The client receives the response directly from the integration endpoint instead of the cache, provided that the client is authorized to do so. This replaces the existing cache entry with the new response, which is fetched from the integration endpoint.

To grant permission for a client, attach a policy of the following format to an IAM execution role for the user.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"
      ]
    }
  ]
}
```

This policy allows the API Gateway execution service to invalidate the cache for requests on the specified resource (or resources). To specify a group of targeted resources, use a wildcard (*) character for `account-id`, `api-id`, and other entries in the ARN value of `Resource`. For more information on how to set permissions for the API Gateway execution service, see Control access to an API with IAM permissions.

If you don't impose an `InvalidateCache` policy (or choose the **Require authorization** check box in the console), any client can invalidate the API cache. If most or all of the clients invalidate the API cache, this could significantly increase the latency of your API.

When the policy is in place, caching is enabled and authorization is required. You can control how unauthorized requests are handled by choosing an option from **Handle unauthorized requests** in the API Gateway console.

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html#invalidate-method-caching

Incorrect options:

Use the Request parameter: ?bypass_cache=1 - Method parameters take query string but this is not one of them.

Using the Header Bypass-Cache=1 - This is a made-up option.

Using the request parameter ?cache-control-max-age=0 - To invalidate cache it requires a header and not a request parameter.

Reference:

https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html#invalidate-method-caching

Question 13: **Correct**

Your AWS CodeDeploy deployment to T2 instances succeed. The new application revision makes API calls to Amazon S3 however the application is not working as expected due to authorization exceptions and you were assigned to troubleshoot the issue.

Which of the following should you do?

- ○ Fix the IAM permissions for the CodeDeploy service role
- ○ Make the S3 bucket public
- ○ Enable CodeDeploy Proxy
- ◉ Fix the IAM permissions for the EC2 instance role (Correct)

Explanation
Correct option:

Fix the IAM permissions for the EC2 instance role

You should use an IAM role to manage temporary credentials for applications that run on an EC2 instance. When you use a role, you don't have to distribute long-term credentials (such as a user name and password or access keys) to an EC2 instance. Instead, the role supplies temporary permissions that applications can use when they make calls to other AWS resources. In this case, make sure your role has access to the S3 bucket.

Incorrect options:

Fix the IAM permissions for the CodeDeploy service role - The fact that CodeDeploy deployed the application to EC2 instances tells us that there was no issue between those two. The actual issue is between the EC2 instances and S3.

Make the S3 bucket public - This is not a good practice, you should strive to provide least privilege access. You may have files in here that should not be allowed public access and you are opening the door to security breaches.

Enable CodeDeploy Proxy - This is not correct as we don't need to look into CodeDeploy settings but rather between EC2 and S3 permissions.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

Question 14: **Correct**

Your company is in the process of building a DevOps culture and is moving all of its on-premise resources to the cloud using serverless architectures and automated deployments. You have created a CloudFormation template in YAML that uses an AWS Lambda function to pull HTML files from GitHub and place them into an Amazon Simple Storage Service (S3) bucket that you specify.

Which of the following AWS CLI commands can you use to upload AWS Lambda functions and AWS CloudFormation templates to AWS?

- ○ `cloudformation zip` and `cloudformation deploy`
- ◉ `cloudformation package` and `cloudformation deploy` (Correct)
- ○ `cloudformation package` and `cloudformation upload`
- ○ `cloudformation zip` and `cloudformation upload`

Explanation
Correct option:

`cloudformation package` and `cloudformation deploy`

AWS CloudFormation gives developers and businesses an easy way to create a collection of related AWS and third-party resources and provision them in an orderly and predictable fashion.

How CloudFormation Works:

via - https://aws.amazon.com/cloudformation/

The `cloudformation package` command packages the local artifacts (local paths) that your AWS CloudFormation template references. The command will upload local artifacts, such as your source code for your AWS Lambda function.

The `cloudformation deploy` command deploys the specified AWS CloudFormation template by creating and then executing a changeset.

Incorrect options:

`cloudformation package` and `cloudformation upload` - The `cloudformation upload` command does not exist.

`cloudformation zip` and `cloudformation upload` - Both commands do not exist, this is a made-up option.

`cloudformation zip` and `cloudformation deploy` - The `cloudformation zip` command does not exist, this is a made-up option.

Reference:

https://docs.aws.amazon.com/cli/latest/reference/cloudformation/package.html

Question 15: **Incorrect**

A company has several Linux-based EC2 instances that generate various log files which need to be analyzed for security and compliance purposes. The company wants to use Kinesis Data Streams (KDS) to analyze this log data.

Which of the following is the most optimal way of sending log data from the EC2 instances to KDS?

- Run cron job on each of the instances to collect log data and send it to Kinesis Data Streams
- Install and configure Kinesis Agent on each of the instances **(Correct)**
- Use Kinesis Producer Library (KPL) to collect and ingest data from each EC2 instance **(Incorrect)**
- Install AWS SDK on each of the instances and configure it to send the necessary files to Kinesis Data Streams

Explanation
Correct option:

Install and configure Kinesis Agent on each of the instances

Kinesis Agent is a stand-alone Java software application that offers an easy way to collect and send data to Kinesis Data Streams. The agent continuously monitors a set of files and sends new data to your stream. The agent handles file rotation, checkpointing, and retry upon failures. It delivers all of your data in a reliable, timely, and simple manner. It also emits Amazon CloudWatch metrics to help you better monitor and troubleshoot the streaming process.

You can install the agent on Linux-based server environments such as web servers, log servers, and database servers. After installing the agent, configure it by specifying the files to monitor and the stream for the data. After the agent is configured, it durably collects data from the files and reliably sends it to the stream.

The agent can also pre-process the records parsed from monitored files before sending them to your stream. You can enable this feature by adding the dataProcessingOptions configuration setting to your file flow. One or more processing options can be added and they will be performed in the specified order.

Incorrect options:

Run cron job on each of the instances to collect log data and send it to Kinesis Data Streams - This solution is possible, though not an optimal one. This solution requires writing custom code and tracking file/log changes, retry failures and so on. Kinesis Agent is built to handle all these requirements and integrates with Data Streams.

Install AWS SDK on each of the instances and configure it to send the necessary files to Kinesis Data Streams - Kinesis Data Streams APIs that are available in the AWS SDKs, helps you manage many aspects of Kinesis Data Streams, including creating streams, resharding, and putting and getting records. You will need to write custom code to handle new data in the log files and send it over to your stream. Kinesis Agent does it easily, as it is designed to continuously monitor a set of files and send new data to your stream.

Use Kinesis Producer Library (KPL) to collect and ingest data from each EC2 instance - The KPL is an easy-to-use, highly configurable library that helps you write to a Kinesis data stream. It acts as an intermediary between your producer application code and the Kinesis Data Streams API actions. This is not optimal compared to Kinesis Agent which is designed to continuously monitor a set of files and send new data to your stream.

Reference:

https://docs.aws.amazon.com/streams/latest/dev/writing-with-agents.html

Question 16: **Correct**

You have moved your on-premise infrastructure to AWS and are in the process of configuring an AWS Elastic Beanstalk deployment environment for production, development, and testing. You have configured your production environment to use a rolling deployment to prevent your application from becoming unavailable to users. For the development and testing environment, you would like to deploy quickly and are not concerned about downtime.

Which of the following deployment policies meet your needs?

- (●) All at once (Correct)
- ( ) Immutable
- ( ) Rolling with additional batches
- ( ) Rolling

Explanation
Correct option:

All at once

This is the quickest deployment method. Suitable if you can accept a short loss of service, and if quick deployments are important to you. With this method, Elastic Beanstalk deploys the new application version to each instance. Then, the web proxy or application server might need to restart. As a result, your application might be unavailable to users (or have low availability) for a short time.

Overview of Elastic Beanstalk Deployment Policies:

The following list provides summary information about the different deployment policies and adds related considerations.

- **All at once** – The quickest deployment method. Suitable if you can accept a short loss of service, and if quick deployments are important to you. With this method, Elastic Beanstalk deploys the new application version to each instance. Then, the web proxy or application server might need to restart. As a result, your application might be unavailable to users (or have low availability) for a short time.
- **Rolling** – Avoids downtime and minimizes reduced availability, at a cost of a longer deployment time. Suitable if you can't accept any period of completely lost service. With this method, your application is deployed to your environment one batch of instances at a time. Most bandwidth is retained throughout the deployment.
- **Rolling with additional batch** – Avoids any reduced availability, at a cost of an even longer deployment time compared to the *Rolling* method. Suitable if you must maintain the same bandwidth throughout the deployment. With this method, Elastic Beanstalk launches an extra batch of instances, then performs a rolling deployment. Launching the extra batch takes time, and ensures that the same bandwidth is retained throughout the deployment.
- **Immutable** – A slower deployment method, that ensures your new application version is always deployed to new instances, instead of updating existing instances. It also has the additional advantage of a quick and safe rollback in case the deployment fails. With this method, Elastic Beanstalk performs an immutable update to deploy your application. In an immutable update, a second Auto Scaling group is launched in your environment and the new version serves traffic alongside the old version until the new instances pass health checks.
- **Traffic splitting** – A canary testing deployment method. Suitable if you want to test the health of your new application version using a portion of incoming traffic, while keeping the rest of the traffic served by the old application version.

The following table compares deployment method properties.

Deployment methods

Method	Impact of failed deployment	Deploy time	Zero downtime	No DNS change	Rollback process	Code deployed to
All at once	Downtime	◷	⊗ No	⊘ Yes	Manual redeploy	Existing instances
Rolling	Single batch out of service; any successful batches before failure running new application version	◷ ◷ †	⊘ Yes	⊘ Yes	Manual redeploy	Existing instances
Rolling with an additional batch	Minimal if first batch fails; otherwise, similar to **Rolling**	◷ ◷ ◷ †	⊘ Yes	⊘ Yes	Manual redeploy	New and existing instances
Immutable	Minimal	◷ ◷ ◷ ◷	⊘ Yes	⊘ Yes	Terminate new instances	New instances
Traffic splitting	Percentage of client traffic routed to new version temporarily impacted	◷ ◷ ◷ ◷ ††	⊘ Yes	⊘ Yes	Reroute traffic and terminate new instances	New instances
Blue/green	Minimal	◷ ◷ ◷ ◷	⊘ Yes	⊗ No	Swap URL	New instances

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html

Incorrect options:

Rolling - With this method, your application is deployed to your environment one batch of instances at a time. Most bandwidth is retained throughout the deployment. Avoids downtime and minimizes reduced availability, at a cost of a longer deployment time. Suitable if you can't accept any period of completely lost service.

Rolling with additional batches - With this method, Elastic Beanstalk launches an extra batch of instances, then performs a rolling deployment. Launching the extra batch takes time, and ensures that the same bandwidth is retained throughout the deployment. This policy also avoids any reduced availability, although at a cost of an even longer deployment time compared to the Rolling method. Finally, this option is suitable if you must maintain the same bandwidth throughout the deployment.

Immutable - A slower deployment method, that ensures your new application version is always deployed to new instances, instead of updating existing instances. It also has the additional advantage of a quick and safe rollback in case the deployment fails. With this method, Elastic Beanstalk performs an immutable update to deploy your application. In an immutable update, a second Auto Scaling

group is launched in your environment and the new version serves traffic alongside the old version until the new instances pass health checks.

Reference:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html

Question 17: **Correct**

You are storing your video files in a separate S3 bucket than your main static website in an S3 bucket. When accessing the video URLs directly the users can view the videos on the browser, but they can't play the videos while visiting the main website.

What is the root cause of this problem?

- ○ Amend the IAM policy
- ○ Change the bucket policy
- ◉ Enable CORS (Correct)
- ○ Disable Server-Side Encryption

Explanation
Correct option:

Enable CORS

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

To configure your bucket to allow cross-origin requests, you create a CORS configuration, which is an XML document with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that will support for each origin, and other operation-specific information.

For the given use-case, you would create a `<CORSRule>` in `<CORSConfiguration>` for bucket B to allow access from the S3 website origin hosted on bucket A.

Please see this note for more details:

How do I configure CORS on my bucket?

To configure your bucket to allow cross-origin requests, you create a CORS configuration, which is an XML document with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that will support for each origin, and other operation-specific information.

You can add up to 100 rules to the configuration. You add the XML document as the `cors` subresource to the bucket either programmatically or by using the Amazon S3 console. For more information, see Enabling cross-origin resource sharing (CORS).

Instead of accessing a website by using an Amazon S3 website endpoint, you can use your own domain, such as `example1.com` to serve your content. For information about using your own domain, see Configuring a static website using a custom domain registered with Route 53. The following example `cors` configuration has three rules, which are specified as `CORSRule` elements:

- The first rule allows cross-origin PUT, POST, and DELETE requests from the `http://www.example1.com` origin. The rule also allows all headers in a preflight OPTIONS request through the `Access-Control-Request-Headers` header. In response to preflight OPTIONS requests, Amazon S3 returns requested headers.
- The second rule allows the same cross-origin requests as the first rule, but the rule applies to another origin, `http://www.example2.com`.
- The third rule allows cross-origin GET requests from all origins. The `*` wildcard character refers to all origins.

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html

Incorrect options:

Change the bucket policy - A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that grants permissions. With this policy, you can do things such as allow one IP address to access the video file in the S3 bucket. In this scenario, we know that's not the case because it works using the direct URL but it doesn't work when you click on a link to access the video.

Amend the IAM policy - You attach IAM policies to IAM users, groups, or roles, which are then subject to the permissions you've defined. This scenario refers to public users of a website and they need not have an IAM user account.

Disable Server-Side Encryption - Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it, if the video file is encrypted at rest then there is nothing you need to do because AWS handles encrypt and decrypt. Disabling encryption is not an issue because you can access the video directly using an URL but not from the main website.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html

Question 18: **Incorrect**

A voting system hosted on-premise was recently migrated to AWS to lower cost, gain scalability, and to better serve thousands of concurrent users. When one of the AWS resource state changes, it generates an event and will need to trigger AWS Lambda. The AWS resource whose state changes and AWS Lambda does not have direct integration.

Which of the following methods can be used to trigger AWS Lambda?

- ○ Open a support ticket with AWS
- ◉ AWS Lambda Custom Sources **(Incorrect)**
- ○ Cron jobs to trigger AWS Lambda to check the state of your service
- ○ CloudWatch Events Rules with AWS Lambda **(Correct)**

Explanation
Correct option:

CloudWatch Events Rules with AWS Lambda

You can create a Lambda function and direct CloudWatch Events to execute it on a regular schedule. You can specify a fixed rate (for example, execute a Lambda function every hour or 15 minutes), or you can specify a Cron expression.

CloudWatch Events Key Concepts:

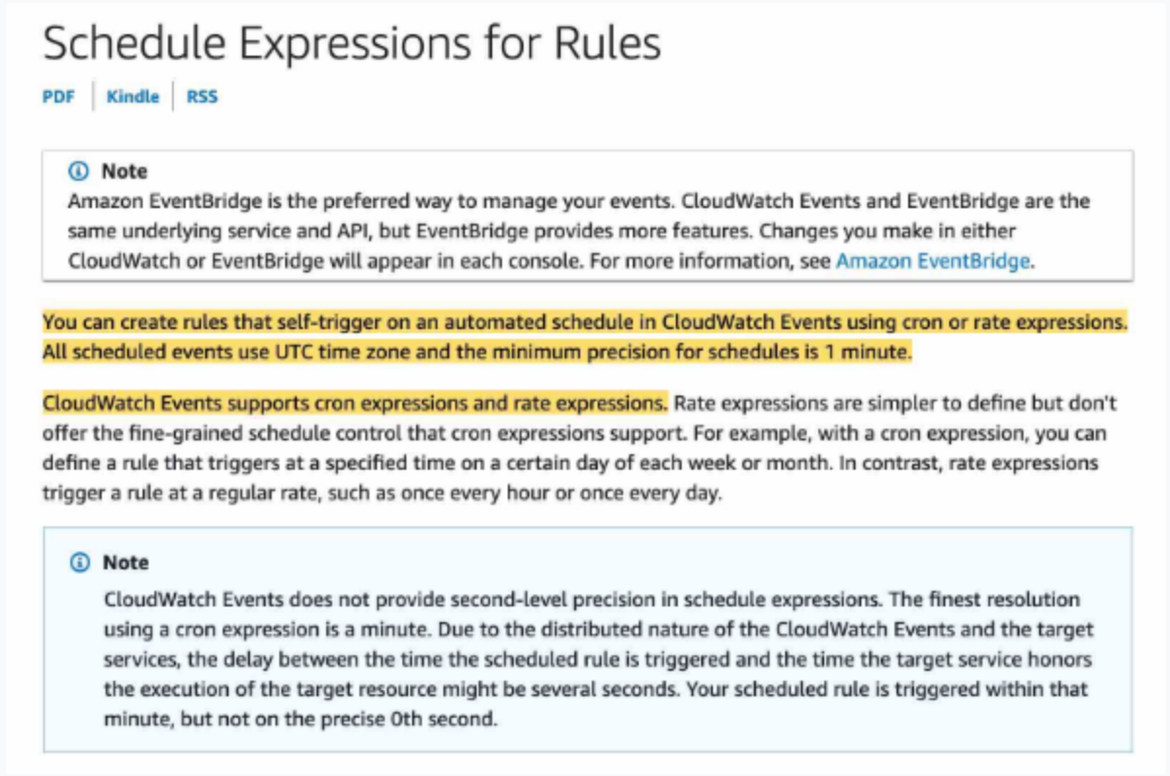
Before you begin using CloudWatch Events, you should understand the following concepts:

- **Events** – An event indicates a change in your AWS environment. AWS resources can generate events when their state changes. For example, Amazon EC2 generates an event when the state of an EC2 instance changes from pending to running, and Amazon EC2 Auto Scaling generates events when it launches or terminates instances. AWS CloudTrail publishes events when you make API calls. You can generate custom application-level events and publish them to CloudWatch Events. You can also set up scheduled events that are generated on a periodic basis. For a list of services that generate events, and sample events from each service, see CloudWatch Events Event Examples From Supported Services.
- **Rules** – A rule matches incoming events and routes them to targets for processing. A single rule can route to multiple targets, all of which are processed in parallel. Rules are not processed in a particular order. This enables different parts of an organization to look for and process the events that are of interest to them. A rule can customize the JSON sent to the target, by passing only certain parts or by overwriting it with a constant.
- **Targets** – A target processes events. Targets can include Amazon EC2 instances, AWS Lambda functions, Kinesis streams, Amazon ECS tasks, Step Functions state machines, Amazon SNS topics, Amazon SQS queues, and built-in targets. A target receives events in JSON format.

A rule's targets must be in the same Region as the rule.

via - https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html

Schedule Expressions for CloudWatch Events Rules:



Schedule Expressions for Rules

PDF | Kindle | RSS

Note
Amazon EventBridge is the preferred way to manage your events. CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes you make in either CloudWatch or EventBridge will appear in each console. For more information, see Amazon EventBridge.

You can create rules that self-trigger on an automated schedule in CloudWatch Events using cron or rate expressions. All scheduled events use UTC time zone and the minimum precision for schedules is 1 minute.

CloudWatch Events supports cron expressions and rate expressions. Rate expressions are simpler to define but don't offer the fine-grained schedule control that cron expressions support. For example, with a cron expression, you can define a rule that triggers at a specified time on a certain day of each week or month. In contrast, rate expressions trigger a rule at a regular rate, such as once every hour or once every day.

Note
CloudWatch Events does not provide second-level precision in schedule expressions. The finest resolution using a cron expression is a minute. Due to the distributed nature of the CloudWatch Events and the target services, the delay between the time the scheduled rule is triggered and the time the target service honors the execution of the target resource might be several seconds. Your scheduled rule is triggered within that minute, but not on the precise 0th second.

via -
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

Incorrect options:

AWS Lambda Custom Sources - This is a made-up option and has been added as a distractor.

Open a support ticket with AWS - You can, although the AWS support team will not add a custom configuration for you, they will step you through creating event rule with Lambda.

Cron jobs to trigger AWS Lambda to check the state of your service - You would need an additional server for your cron job instead you should consider using a cron expression with CloudWatch.

References:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html

https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html

Question 19: **Correct**

You work as a developer doing contract work for the government on AWS gov cloud. Your applications use Amazon Simple Queue Service (SQS) for its message queue service. Due to recent hacking attempts, security measures have become stricter and require you to store data in encrypted queues.

Which of the following steps can you take to meet your requirements without making changes to the existing code?

- ○ Use the SSL endpoint
- ○ Use Secrets Manager
- ○ Use Client side encryption
- ◉ Enable SQS KMS encryption (Correct)

Explanation
Correct option:

Enable SQS KMS encryption

Server-side encryption (SSE) lets you transmit sensitive data in encrypted queues. SSE protects the contents of messages in queues using keys managed in AWS Key Management Service (AWS KMS).

AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. When you use Amazon SQS with AWS KMS, the data keys that encrypt your message data are also encrypted and stored with the data they protect.

You can choose to have SQS encrypt messages stored in both Standard and FIFO queues using an encryption key provided by AWS Key Management Service (KMS).

Incorrect options:

Use the SSL endpoint - The given use-case needs encryption at rest. When using SSL, the data is encrypted during transit, but the data needs to be encrypted at rest as well, so this option is incorrect.

Use Client-side encryption - For additional security, you can build your application to encrypt messages before they are placed in a message queue but will require a code change, so this option is incorrect.

**Use Secrets Manager ** - AWS Secrets Manager enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager APIs, eliminating the need to hardcode sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB. Secrets Manager cannot be used for encrypting data at rest.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-server-side-encryption.html

Question 20: **Correct**

A media company is building an application that needs to store video files in Amazon S3. Management requires that the files be encrypted before they are sent to Amazon S3 for storage. The encryption keys need to be managed by an in-house security team but the key itself is stored on AWS.

Which solution should the company use to meet these requirements?

- ○ Use client-side encryption with Amazon S3 managed key
- ◉ Use client-side encryption with an AWS KMS managed customer master key (CMK) **(Correct)**
- ○ Use server-side encryption with customer-provided encryption key (SSE-C)
- ○ Use server-side encryption with a client-side master key

Explanation
Correct option:

Use client-side encryption with an AWS KMS managed customer master key (CMK)

You have the following options for protecting data at rest in Amazon S3:

Server-Side Encryption – Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.

Client-Side Encryption – Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.

As the company wants to make sure that the files are encrypted before they are sent to Amazon S3, therefore you should use client-side encryption.

To enable client-side encryption, you have the following options:

Use a customer master key (CMK) stored in AWS Key Management Service (AWS KMS).

Use a master key you store within your application.

As the use-case mentions that the encryption keys need to be managed by an in-house security team but the key itself should be stored on AWS, therefore you must use the customer master key (CMK) stored in AWS Key Management Service (AWS KMS).

Incorrect options:

Use client-side encryption with Amazon S3 managed key - As mentioned earlier in the explanation, to meet the requirements of the given use-case, the company must use a customer master key (CMK) stored in AWS Key Management Service (AWS KMS).

Also, there is no such thing as "client-side encryption with Amazon S3 managed key" as explained above.

Use server-side encryption with a customer-provided encryption key (SSE-C)

Use server-side encryption with a client-side master key

Both these options are incorrect as the use-case mandates client-side encryption.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingClientSideEncryption.html

Question 21: **Correct**

Your organization is looking into making a change to all virtual machines in the cloud. They would like to take advantage of running a bootstrap script for their Windows Server 2012 Base AMI virtual machines when they first boot.

Which of the following options will allow the organization to achieve this?

- ○ EC2 Meta Data
- ○ Mount EFS network drives
- ◉ EC2 User Data (Correct)
- ○ Custom AMI

Explanation
Correct option:

EC2 User Data

When you launch an instance in Amazon EC2, you have the option of passing user data to the instance that can be used to perform common automated configuration tasks and even run scripts after the instance starts. A use case for user data would be if you run web servers for various small businesses, they can all use the same AMI and retrieve their content from the Amazon S3 bucket you specify in the user data at launch.

Incorrect options:

Mount EFS network drives - Amazon EFS is a file storage service for use with Amazon EC2 that can provide storage for up to thousands of Amazon EC2 instances. EFS does not provide instructions but instead a storage service.

EC2 MetaData - Meta Data is information about your running instance, for example, you can access information such as the local IP address of your instance.

Custom AMI - With a Custom AMI, you can pre-install software and configure your Operating System to have all it needs before launching. If you were using User Data, you would have software or other tasks that you can specify when the instance launches but may take longer to boot so a Custom AMI can be recommended.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

Question 22: **Correct**

An organization uses Alexa as its intelligent assistant to improve productivity throughout the organization. A group of developers manages custom Alexa Skills written in Node.Js to control conference-room equipment settings and start meetings using voice activation. The manager has requested developers that all functions code should be monitored for error rates with the possibility of creating alarms on top of them.

Which of the following options should be chosen? (select two)

- [ ] CloudTrail
- [ ] SSM
- [ ] X-Ray
- [x] CloudWatch Alarms (Correct)
- [x] CloudWatch Metrics (Correct)

Explanation
Correct options:

CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and visualizes it using automated dashboards so you can get a unified view of your AWS resources, applications, and services that run in AWS and on-premises. You can correlate your metrics and logs to better understand the health and performance of your resources. You can also create alarms based on metric value thresholds you specify, or that can watch for anomalous metric behavior based on machine learning algorithms.

How CloudWatch works:

via - https://aws.amazon.com/cloudwatch/

CloudWatch Metrics

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. Metric data is kept for 15 months, enabling you to view both up-to-the-minute data and historical data.

CloudWatch retains metric data as follows:

Data points with a period of less than 60 seconds are available for 3 hours. These data points are high-resolution custom metrics. Data points with a period of 60 seconds (1 minute) are available for 15 days Data points with a period of 300 seconds (5 minute) are available for 63 days Data points with a period of 3600 seconds (1 hour) are available for 455 days (15 months)

CloudWatch Alarms

You can use an alarm to automatically initiate actions on your behalf. An alarm watches a single metric over a specified time, and performs one or more specified actions, based on the value of the metric relative to a threshold over time. The action is a notification sent to an Amazon SNS topic or an Auto Scaling policy. You can also add alarms to dashboards.

CloudWatch alarms send notifications or automatically make changes to the resources you are monitoring based on rules that you define. Alarms work together with CloudWatch Metrics.

A metric alarm has the following possible states:

OK – The metric or expression is within the defined threshold.

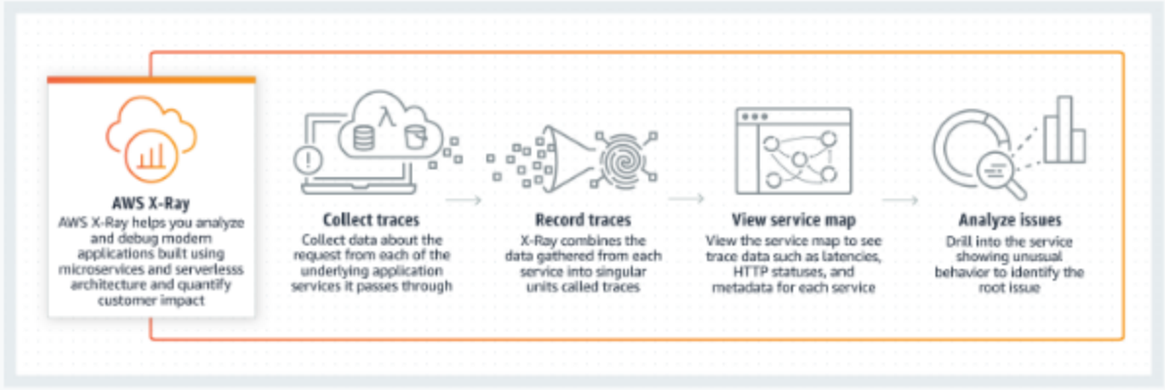
ALARM – The metric or expression is outside of the defined threshold.

INSUFFICIENT_DATA – The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state.

Incorrect options:

X-Ray - AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application's underlying components.

How X-Ray Works:

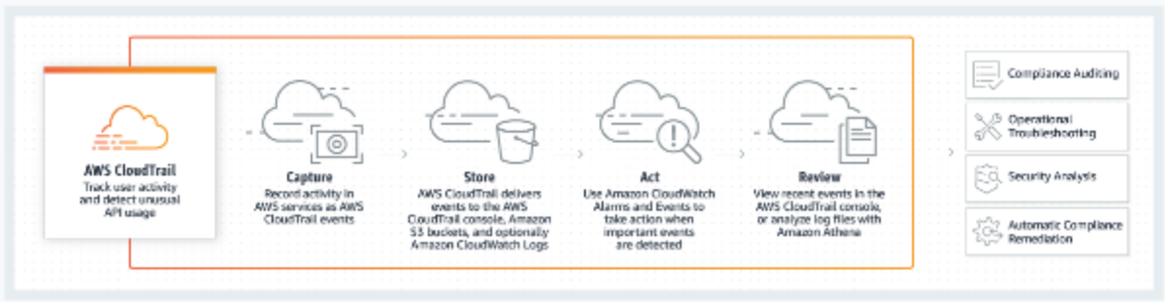


via - https://aws.amazon.com/xray/

X-Ray cannot be used to capture metrics and set up alarms as per the given use-case, so this option is incorrect.

CloudTrail - CloudWatch is a monitoring service whereas CloudTrail is more of an audit service where you can find API calls made on services and by whom.

How CloudTrail Works:



via - https://aws.amazon.com/cloudtrail/

Systems Manager - Using AWS Systems Manager, you can group resources, like Amazon EC2 instances, Amazon S3 buckets, or Amazon RDS instances, by application, view operational data for monitoring and troubleshooting, and take action on your groups of resources. Systems Manager cannot be used to capture metrics and set up alarms as per the given use-case, so this option is incorrect.

References:

https://aws.amazon.com/cloudwatch/

https://aws.amazon.com/cloudtrail/

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html

Question 23: **Correct**

A senior cloud engineer designs and deploys online fraud detection solutions for credit card companies processing millions of transactions daily. The Elastic Beanstalk application sends files to Amazon S3 and then sends a message to an Amazon SQS queue containing the path of the uploaded file in S3. The engineer wants to postpone the delivery of any new messages to the queue for at least 10 seconds.

Which SQS feature should the engineer leverage?

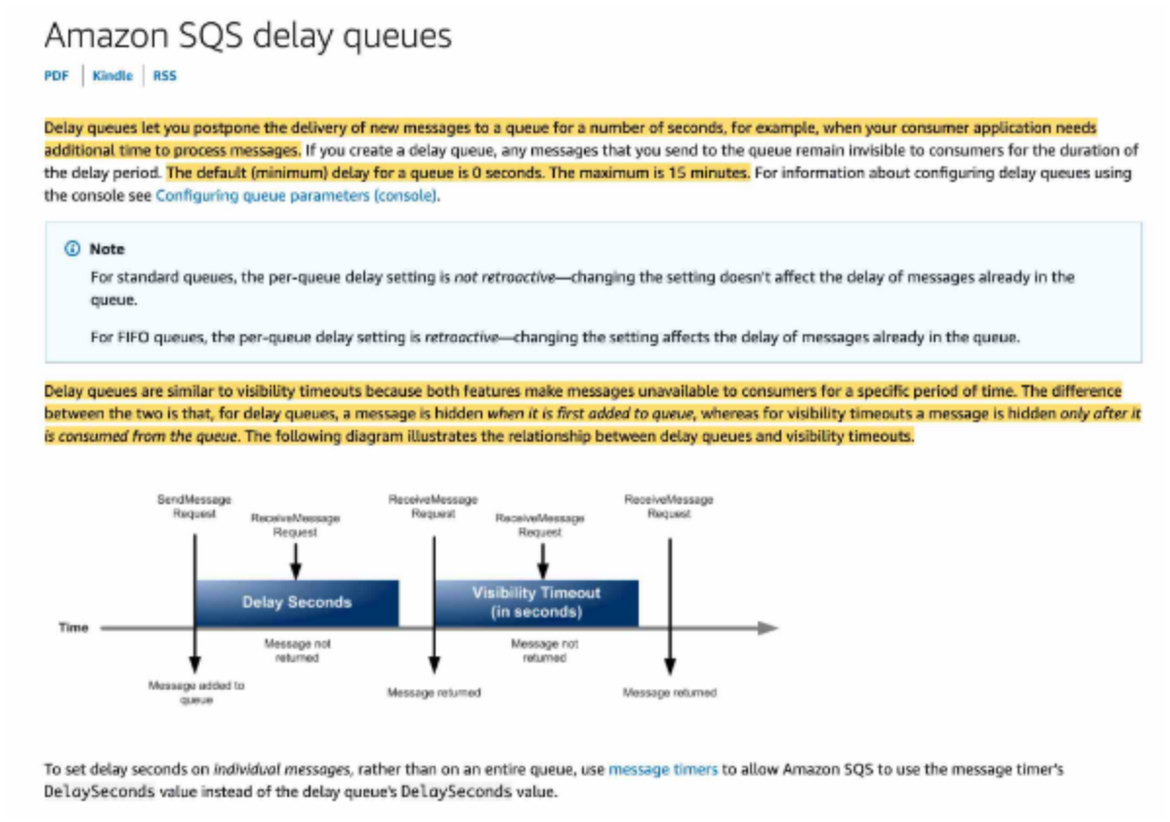
- ○ Enable LongPolling
- ○ Implement application-side delay
- ◉ Use DelaySeconds parameter (Correct)
- ○ Use visibility timeout parameter

Explanation
Correct option:

Use DelaySeconds parameter

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

Delay queues let you postpone the delivery of new messages to a queue for several seconds, for example, when your consumer application needs additional time to process messages. If you create a delay queue, any messages that you send to the queue remain invisible to consumers for the duration of the delay period. The default (minimum) delay for a queue is 0 seconds. The maximum is 15 minutes.



Amazon SQS delay queues

PDF | Kindle | RSS

Delay queues let you postpone the delivery of new messages to a queue for a number of seconds, for example, when your consumer application needs additional time to process messages. If you create a delay queue, any messages that you send to the queue remain invisible to consumers for the duration of the delay period. The default (minimum) delay for a queue is 0 seconds. The maximum is 15 minutes. For information about configuring delay queues using the console see Configuring queue parameters (console).

Note

For standard queues, the per-queue delay setting is *not retroactive*—changing the setting doesn't affect the delay of messages already in the queue.

For FIFO queues, the per-queue delay setting is *retroactive*—changing the setting affects the delay of messages already in the queue.

Delay queues are similar to visibility timeouts because both features make messages unavailable to consumers for a specific period of time. The difference between the two is that, for delay queues, a message is hidden *when it is first added to queue*, whereas for visibility timeouts a message is hidden *only after it is consumed from the queue*. The following diagram illustrates the relationship between delay queues and visibility timeouts.

To set delay seconds on *individual messages*, rather than on an entire queue, use message timers to allow Amazon SQS to use the message timer's `DelaySeconds` value instead of the delay queue's `DelaySeconds` value.

via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html

Incorrect options:

Implement application-side delay - You can customize your application to delay sending messages but it is not a robust solution. You can run into a scenario where your application crashes before sending a message, then that message would be lost.

Use visibility timeout parameter - Visibility timeout is a period during which Amazon SQS prevents other consumers from receiving and processing a given message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours. You cannot use visibility timeout to postpone the delivery of new messages to the queue for a few seconds.

Enable LongPolling - Long polling makes it inexpensive to retrieve messages from your Amazon SQS queue as soon as the messages are available. Long polling helps reduce the cost of using Amazon SQS by eliminating the number of empty responses (when there are no messages available for a ReceiveMessage request) and false empty responses (when messages are available but aren't included in a response). When the wait time for the ReceiveMessage API action is greater than 0, long polling is in effect. The maximum long polling wait time is 20 seconds. You cannot use LongPolling to postpone the delivery of new messages to the queue for a few seconds.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-delay-queues.html

Question 24: **Incorrect**

You have a popular web application that accesses data stored in an Amazon Simple Storage Service (S3) bucket. Developers use the SDK to maintain the application and add new features. Security compliance requests that all new objects uploaded to S3 be encrypted using SSE-S3 at the time of upload. Which of the following headers must the developers add to their request?

- 'x-amz-server-side-encryption': 'SSE-KMS'
- 'x-amz-server-side-encryption': 'aws:kms' **(Incorrect)**
- 'x-amz-server-side-encryption': 'SSE-S3'
- 'x-amz-server-side-encryption': 'AES256' **(Correct)**

Explanation
Correct option:

'x-amz-server-side-encryption': 'AES256'

Server-side encryption protects data at rest. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it rotates regularly. Amazon S3 server-side encryption uses one of the strongest block ciphers available to encrypt your data, 256-bit Advanced Encryption Standard (AES-256).

SSE-S3 Overview:

Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3)

PDF | Kindle | RSS

Server-side encryption protects data at rest. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it rotates regularly. Amazon S3 server-side encryption uses one of the strongest block ciphers available to encrypt your data, 256-bit Advanced Encryption Standard (AES-256).

If you need server-side encryption for all of the objects that are stored in a bucket, use a bucket policy. For example, the following bucket policy denies permissions to upload an object unless the request includes the `x-amz-server-side-encryption` header to request server-side encryption:

```
{
  "Version": "2012-10-17",
  "Id": "PutObjPolicy",
  "Statement": [
    {
      "Sid": "DenyIncorrectEncryptionHeader",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    },
    {
      "Sid": "DenyUnEncryptedObjectUploads",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::YourBucket/*",
      "Condition": {
        "Null": {
          "s3:x-amz-server-side-encryption": "true"
        }
      }
    }
  ]
}
```

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html

'x-amz-server-side-encryption': 'SSE-S3' - SSE-S3 (Amazon S3-Managed Keys) is an option available but it's not a valid header value.

'x-amz-server-side-encryption': 'SSE-KMS' - SSE-KMS (AWS KMS-Managed Keys) is an option available but it's not a valid header value. A valid value would be 'aws:kms'

'x-amz-server-side-encryption': 'aws:kms' - Server-side encryption is the encryption of data at its destination by the application or service that receives it. AWS Key Management Service (AWS KMS) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Amazon S3 uses AWS KMS customer master keys (CMKs) to encrypt your Amazon S3 objects. AWS KMS encrypts only the object data. Any object metadata is not encrypted.

This is a valid header value and you can use if you need more control over your keys like create, rotating, disabling them using AWS KMS. Otherwise, if you wish to let AWS S3 manage your keys just stick with SSE-S3.

References:

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingEncryption.html

Question 25: **Correct**

You are a manager for a tech company that has just hired a team of developers to work on the company's AWS infrastructure. All the developers are reporting to you that when using the AWS CLI to execute commands it fails with the following exception: You are not authorized to perform this operation. Encoded authorization failure message: 6h34GtpmGjJJUm946eDVBfzWQJk6z5GePbbGDs9Z2T8xZj9EZtEduSnTbmrR7pMqpJrVYJCew2m8YBZQf4HRWEtrpncANrZMsnzk.

Which of the following actions will help developers decode the message?

- (●) AWS STS decode-authorization-message (Correct)
- ( ) Use KMS decode-authorization-message
- ( ) AWS IAM decode-authorization-message

AWS Cognito Decoder

Explanation
Correct option:

AWS STS decode-authorization-message

Use decode-authorization-message to decode additional information about the authorization status of a request from an encoded message returned in response to an AWS request. If a user is not authorized to perform an action that was requested, the request returns a Client.UnauthorizedOperation response (an HTTP 403 response). The message is encoded because the details of the authorization status can constitute privileged information that the user who requested the operation should not see. To decode an authorization status message, a user must be granted permissions via an IAM policy to request the DecodeAuthorizationMessage (sts:DecodeAuthorizationMessage) action.

Incorrect options:

AWS IAM decode-authorization-message - The IAM service does not have this command, as it's a made-up option.

Use KMS decode-authorization-message - The KMS service does not have this command, as it's a made-up option.

AWS Cognito Decoder - The Cognito service does not have this command, as it's a made-up option.

Reference:

https://docs.aws.amazon.com/cli/latest/reference/sts/decode-authorization-message.html

Question 26: **Correct**

A firm uses AWS DynamoDB to store information about people's favorite sports teams and allow the information to be searchable from their home page. There is a daily requirement that all 10 million records in the table should be deleted then re-loaded at 2:00 AM each night.

Which option is an efficient way to delete with minimal costs?

Scan and call DeleteItem

Call PurgeTable

Delete then re-create the table (Correct)

Scan and call BatchDeleteItem

Explanation
Correct option:

Delete then re-create the table

The DeleteTable operation deletes a table and all of its items. After a `DeleteTable` request, the specified table is in the `DELETING` state until DynamoDB completes the deletion.

Scan and call DeleteItem - Scan is a very slow operation for 10 million items and this is not the best-fit option for the given use-case.

Scan and call BatchDeleteItem - Scan is a very slow operation for 10 million items and this is not the best-fit option for the given use-case.

Call PurgeTable - This is a made-up option and has been added as a distractor.

Reference:

https://docs.aws.amazon.com/cli/latest/reference/dynamodb/delete-table.html

Question 27: **Correct**

A development team is storing sensitive customer data in S3 that will require encryption at rest. The encryption keys must be rotated at least annually.

What is the easiest way to implement a solution for this requirement?

- Import a custom key into AWS KMS and automate the key rotation on an annual basis by using a Lambda function
- Use AWS KMS with automatic key rotation (Correct)
- Use SSE-C with automatic key rotation on an annual basis
- Encrypt the data before sending it to Amazon S3

Explanation
Correct option:

Use AWS KMS with automatic key rotation - Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it. You have three mutually exclusive options, depending on how you choose to manage the encryption keys: Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS), Server-Side Encryption with Customer-Provided Keys (SSE-C).

When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed CMK, or you can specify a customer managed CMK that you have already created. If you don't specify a customer managed CMK, Amazon S3 automatically creates an AWS managed CMK in your AWS account the first time that you add an object encrypted with SSE-KMS to a bucket. By default, Amazon S3 uses this CMK for SSE-KMS.

You can choose to have AWS KMS automatically rotate CMKs every year, provided that those keys were generated within AWS KMS HSMs.

Incorrect options:

Encrypt the data before sending it to Amazon S3 - The act of encrypting data before sending it to Amazon S3 is called Client-Side encryption. You will have to handle the key generation, maintenance and rotation process. Hence, this is not the right choice here.

Import a custom key into AWS KMS and automate the key rotation on an annual basis by using a Lambda function - When you import a custom key, you are responsible for maintaining a copy of your imported keys in your key management infrastructure so that you can re-import them at any time. Also, automatic key rotation is not supported for imported keys. Using Lambda functions to rotate keys is a possible solution, but not an optimal one for the current use case.

Use SSE-C with automatic key rotation on an annual basis - With Server-Side Encryption with Customer-Provided Keys (SSE-C), you manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects. The keys are not stored anywhere in Amazon S3. There is no automatic key rotation facility for this option.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingKMSEncryption.html

Question 28: **Correct**

You have been hired at a company that needs an experienced developer to help with a continuous integration/continuous delivery (CI/CD) workflow on AWS. You configure the company's workflow to run an AWS CodePipeline pipeline whenever the application's source code changes in a repository hosted in AWS Code Commit and compiles source code with AWS Code Build. You are configuring ProjectArtifacts in your build stage.

Which of the following should you do?

- ○ Configure AWS CodeBuild to store output artifacts on EC2 servers
- ○ Contact AWS Support to allow AWS CodePipeline to manage build outputs
- ○ Give AWS CodeCommit permissions to upload the build output to your Amazon S3 bucket
- ◉ Give AWS CodeBuild permissions to upload the build output to your Amazon S3 bucket (Correct)

Explanation
Correct option:

Give AWS CodeBuild permissions to upload the build output to your Amazon S3 bucket

If you choose ProjectArtifacts and your value type is S3 then the build project stores build output in Amazon Simple Storage Service (Amazon S3). For that, you will need to give AWS CodeBuild permissions to upload.

Incorrect options:

Configure AWS CodeBuild to store output artifacts on EC2 servers - EC2 servers are not a valid output location, so this option is ruled out.

Give AWS CodeCommit permissions to upload the build output to your Amazon S3 bucket - AWS CodeCommit is the repository that holds source code and has no control over compiling the source code, so this option is incorrect.

Contact AWS Support to allow AWS CodePipeline to manage build outputs - You can set AWS CodePipeline to manage its build output locations instead of AWS CodeBuild. There is no need to contact AWS Support.

Reference:

https://docs.aws.amazon.com/codebuild/latest/userguide/create-project.html#create-project-cli

Question 29: **Correct**

An IT company has a web application running on Amazon EC2 instances that needs read-only access to an Amazon DynamoDB table.

As a Developer Associate, what is the best-practice solution you would recommend to accomplish this task?

- ○ Create an IAM user with Administrator access and configure AWS credentials for this user on the given EC2 instance
- ◉ Create an IAM role with an AmazonDynamoDBReadOnlyAccess policy and apply it to the EC2 instance profile (Correct)
- ○ Run application code with AWS account root user credentials to ensure full access to all AWS services
- ○ Create a new IAM user with access keys. Attach an inline policy to the IAM user with read-only access to DynamoDB. Place the keys in the code. For security, redeploy the code whenever the keys rotate

Explanation
Correct option:

Create an IAM role with an AmazonDynamoDBReadOnlyAccess policy and apply it to the EC2 instance profile

As an AWS security best practice, you should not create an IAM user and pass the user's credentials to the application or embed the credentials in the application. Instead, create an IAM role that you attach to the EC2 instance to give temporary security credentials to applications running on the instance. When an application uses these credentials in AWS, it can perform all of the operations that are allowed by the policies attached to the role.

So for the given use-case, you should create an IAM role with an AmazonDynamoDBReadOnlyAccess policy and apply it to the EC2 instance profile.

When to Create an IAM Role (Instead of a User)

Create an IAM role in the following situations:

You're creating an application that runs on an Amazon Elastic Compute Cloud (Amazon EC2) instance and that application makes requests to AWS.

Don't create an IAM user and pass the user's credentials to the application or embed the credentials in the application. Instead, create an IAM role that you attach to the EC2 instance to give temporary security credentials to applications running on the instance. When an application uses these credentials in AWS, it can perform all of the operations that are allowed by the policies attached to the role. For details, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances.

You're creating an app that runs on a mobile phone and that makes requests to AWS.

Don't create an IAM user and distribute the user's access key with the app. Instead, use an identity provider like Login with Amazon, Amazon Cognito, Facebook, or Google to authenticate users and map the users to an IAM role. The app can use the role to get temporary security credentials that have the permissions specified by the policies attached to the role. For more information, see the following:

- Amazon Cognito Overview in the *AWS Mobile SDK for Android Developer Guide*
- Amazon Cognito Overview in the *AWS Mobile SDK for iOS Developer Guide*
- About Web Identity Federation

Users in your company are authenticated in your corporate network and want to be able to use AWS without having to sign in again—that is, you want to allow users to federate into AWS.

Don't create IAM users. Configure a federation relationship between your enterprise identity system and AWS. You can do this in two ways:

- If your company's identity system is compatible with SAML 2.0, you can establish trust between your company's identity system and AWS. For more information, see About SAML 2.0-based Federation.
- Create and use a custom proxy server that translates user identities from the enterprise into IAM roles that provide temporary AWS security credentials. For more information, see Enabling Custom Identity Broker Access to the AWS Console.

via - https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html

Incorrect options:

Create a new IAM user with access keys. Attach an inline policy to the IAM user with read-only access to DynamoDB. Place the keys in the code. For security, redeploy the code whenever the keys rotate

Create an IAM user with Administrator access and configure AWS credentials for this user on the given EC2 instance

Run application code with AWS account root user credentials to ensure full access to all AWS services

As mentioned in the explanation above, it is dangerous to pass an IAM user's credentials to the application or embed the credentials in the application. The security implications are even higher when you use an IAM user with admin privileges or use the AWS account root user. So all three options are incorrect.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html

Question 30: **Incorrect**

A developer is creating access credentials for an Amazon EC2 instance that hosts the web application using AWS SDK for Java.

If the default credentials provider chain is used on the instance, which parameter will be checked first for the required credentials?

- (●) The system environment variables: AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY are checked first **(Incorrect)**
- ( ) Credentials delivered through the Amazon EC2 container service
- ( ) Instance profile credentials, which exist within the instance metadata associated with the IAM role for the EC2 instance
- ( ) Parameters `aws.accessKeyId` and `aws.secretKey` will be checked in the Java system properties **(Correct)**

Explanation
Correct option:

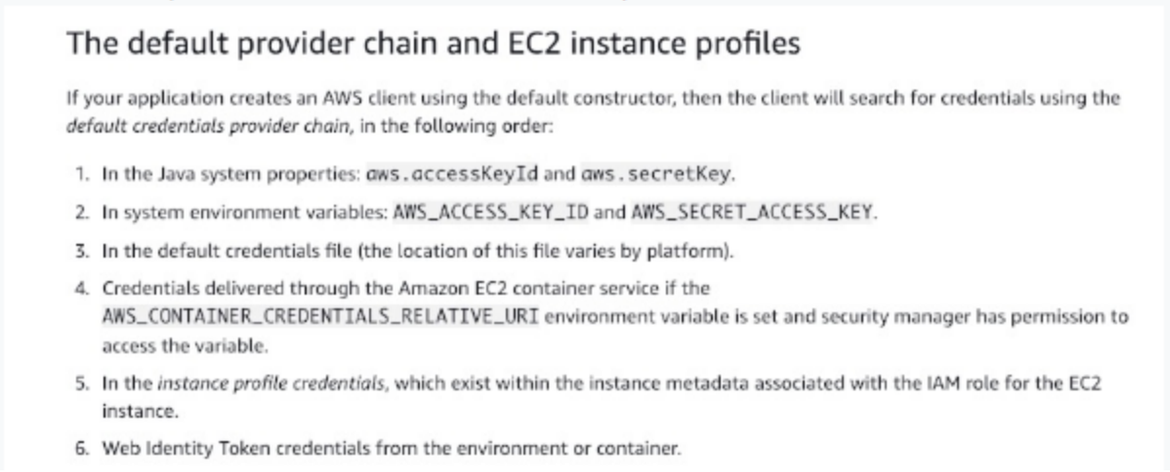
Parameters `aws.accessKeyId` and `aws.secretKey` will be checked in the Java system properties

If your application creates an AWS client using the default constructor, then the client will search for credentials using the default credentials provider chain, in the following order:

1. In the Java system properties: `aws.accessKeyId` and `aws.secretKey`.
2. In system environment variables: AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.
3. In the default credentials file (the location of this file varies by platform).
4. Credentials delivered through the Amazon EC2 container service if the AWS_CONTAINER_CREDENTIALS_RELATIVE_URI environment variable is set and security manager has permission to access the variable.
5. In the instance profile credentials, which exist within the instance metadata associated with the IAM role for the EC2 instance.
6. Web Identity Token credentials from the environment or container.

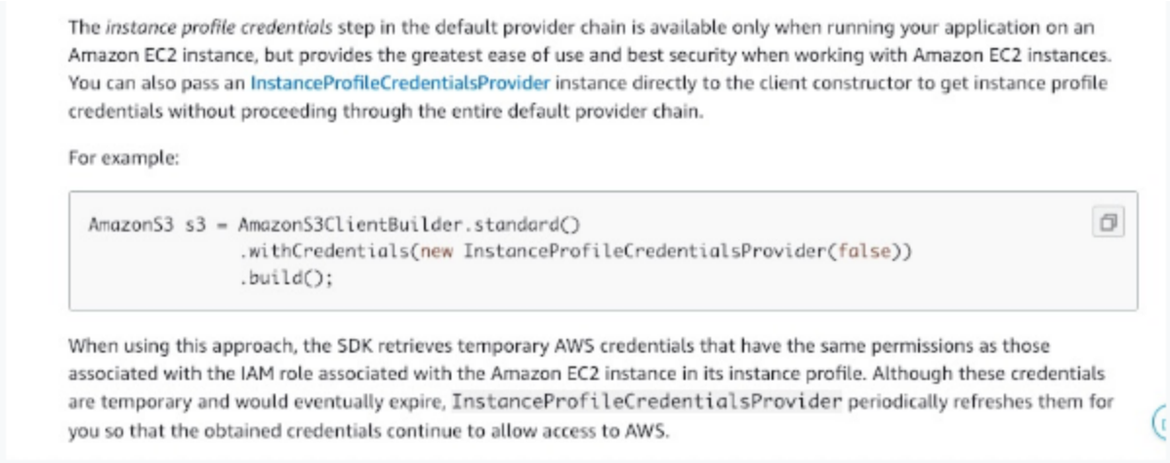
The `instance profile credentials` step in the default provider chain is available only when running your application on an Amazon EC2 instance, but provides the greatest ease of use and best security when working with Amazon EC2 instances. You can also pass an InstanceProfileCredentialsProvider instance directly to the client constructor to get instance profile credentials without proceeding through the entire default provider chain.

The default provider chain and EC2 instance profiles:


The default provider chain and EC2 instance profiles

If your application creates an AWS client using the default constructor, then the client will search for credentials using the *default credentials provider chain*, in the following order:

1. In the Java system properties: `aws.accessKeyId` and `aws.secretKey`.
2. In system environment variables: `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
3. In the default credentials file (the location of this file varies by platform).
4. Credentials delivered through the Amazon EC2 container service if the `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` environment variable is set and security manager has permission to access the variable.
5. In the *instance profile credentials*, which exist within the instance metadata associated with the IAM role for the EC2 instance.
6. Web Identity Token credentials from the environment or container.


The *instance profile credentials* step in the default provider chain is available only when running your application on an Amazon EC2 instance, but provides the greatest ease of use and best security when working with Amazon EC2 instances. You can also pass an InstanceProfileCredentialsProvider instance directly to the client constructor to get instance profile credentials without proceeding through the entire default provider chain.

For example:

```
AmazonS3 s3 = AmazonS3ClientBuilder.standard()
              .withCredentials(new InstanceProfileCredentialsProvider(false))
              .build();
```

When using this approach, the SDK retrieves temporary AWS credentials that have the same permissions as those associated with the IAM role associated with the Amazon EC2 instance in its instance profile. Although these credentials are temporary and would eventually expire, `InstanceProfileCredentialsProvider` periodically refreshes them for you so that the obtained credentials continue to allow access to AWS.

via - https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-roles.html

Incorrect options:

The system environment variables: AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY are checked first

Credentials delivered through the Amazon EC2 container service

Instance profile credentials, which exist within the instance metadata associated with the IAM role for the EC2 instance

These three options contradict the explanation provided above, so these options are incorrect.

Reference:

https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-roles.html

Question 31: **Correct**

You are revising options that would be best for monitoring a few EC2 instances you currently manage. Amazon CloudWatch has metrics available to monitor your EC2 instances for CPU load, I/O, and network I/O. Your budget does not allow for spending on monitoring so using the default monitoring available is your preferred choice.

With default monitoring, at what interval will these metrics be collected?

- ○ 1 minute
- ○ 2 minutes
- ○ 10 minutes
- ◉ 5 minutes (Correct)

Explanation
Correct option:

5 minutes

By default, your instance is enabled for basic monitoring and Amazon EC2 sends metric data to CloudWatch in 5-minute periods. You can optionally enable detailed monitoring. After you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period for the instance.

The following describes the data interval and charge for basic and detailed monitoring for instances.

Basic monitoring Data is available automatically in 5-minute periods at no charge.

Detailed monitoring Data is available in 1-minute periods for an additional charge.

Incorrect options:

10 minutes

2 minutes

1 minute

These three options contradict the details provided in the explanation above, so these options are not correct.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-cloudwatch.html

Question 32: **Correct**

An order management system uses a cron job to poll for any new orders. Every time a new order is created, the cron job sends this order data as a message to the message queues to facilitate downstream order processing in a reliable way. To reduce costs and improve performance, the company wants to move this functionality to AWS cloud.

Which of the following is the most optimal solution to meet this requirement?

- Use Amazon Simple Notification Service (SNS) to push notifications and use AWS Lambda functions to process the information received from SNS
- Use Amazon Simple Notification Service (SNS) to push notifications when an order is created. Configure different Amazon Simple Queue Service (SQS) queues to receive these messages for downstream processing **(Correct)**
- Use Amazon Simple Notification Service (SNS) to push notifications to Kinesis Data Firehose delivery streams for processing the data for downstream applications
- Configure different Amazon Simple Queue Service (SQS) queues to poll for new orders

Explanation
Correct option:

Use Amazon Simple Notification Service (SNS) to push notifications when an order is created. Configure different Amazon Simple Queue Service (SQS) queues to receive these messages for downstream processing

Amazon SNS works closely with Amazon Simple Queue Service (Amazon SQS). These services provide different benefits for developers. Amazon SNS allows applications to send time-critical messages to multiple subscribers through a "push" mechanism, eliminating the need to periodically check or "poll" for updates. Amazon SQS is a message queue service used by distributed applications to exchange messages through a polling model, and can be used to decouple sending and receiving components—without requiring each component to be concurrently available.

Using Amazon SNS and Amazon SQS together, messages can be delivered to applications that require immediate notification of an event, and also stored in an Amazon SQS queue for other applications to process at a later time.

When you subscribe an Amazon SQS queue to an Amazon SNS topic, you can publish a message to the topic and Amazon SNS sends an Amazon SQS message to the subscribed queue. The Amazon SQS message contains the subject and message that were published to the topic along with metadata about the message in a JSON document.

SNS-SQS fanout is the right solution for this use case.

Sample SNS-SQS Fanout message:


When you subscribe an Amazon SQS queue to an Amazon SNS topic, you can publish a message to the topic and Amazon SNS sends an Amazon SQS message to the subscribed queue. The Amazon SQS message contains the subject and message that were published to the topic along with metadata about the message in a JSON document. The Amazon SQS message will look similar to the following JSON document.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:
}
```


via - https://docs.aws.amazon.com/sns/latest/dg/sns-sqs-as-subscriber.html

Incorrect options:

Configure different Amazon Simple Queue Service (SQS) queues to poll for new orders - Amazon SQS cannot be used as a polling service, as messages need to be pushed to the queue, which are then handled by the queue consumers.

Use Amazon Simple Notification Service (SNS) to push notifications and use AWS Lambda functions to process the information received from SNS - Amazon SNS and AWS Lambda are integrated so you can invoke Lambda functions with Amazon SNS notifications. When a message is published to an SNS topic that has a Lambda function subscribed to it, the Lambda function is invoked with the payload of the published message. For the given scenario, we need a service that can store the message data pushed by SNS, for further processing. AWS Lambda does not have capacity to store the message data. In case a Lambda function is unable to process a specific message, it will be left unprocessed. Hence this option is not correct.

Use Amazon Simple Notification Service (SNS) to push notifications to Kinesis Data Firehose delivery streams for processing the data for downstream applications - You can subscribe Amazon Kinesis Data Firehose delivery streams to SNS topics, which allows you to send notifications to additional storage and analytics endpoints. However, Kinesis is built for real-time processing of big data. Whereas, SQS is meant for decoupling dependent systems with easy methods to transmit data/messages. SQS also is a cheaper option when compared to Firehose. Therefore this option is not the right fit for the given use case.

References:

https://docs.aws.amazon.com/sns/latest/dg/sns-sqs-as-subscriber.html

https://docs.aws.amazon.com/sns/latest/dg/sns-lambda-as-subscriber.html

https://docs.aws.amazon.com/sns/latest/dg/sns-firehose-as-subscriber.html

Question 33: **Correct**

A .NET developer team works with many ASP.NET web applications that use EC2 instances to host them on IIS. The deployment process needs to be configured so that multiple versions of the application can run in AWS Elastic Beanstalk. One version would be used for development, testing, and another version for load testing.

Which of the following methods do you recommend?

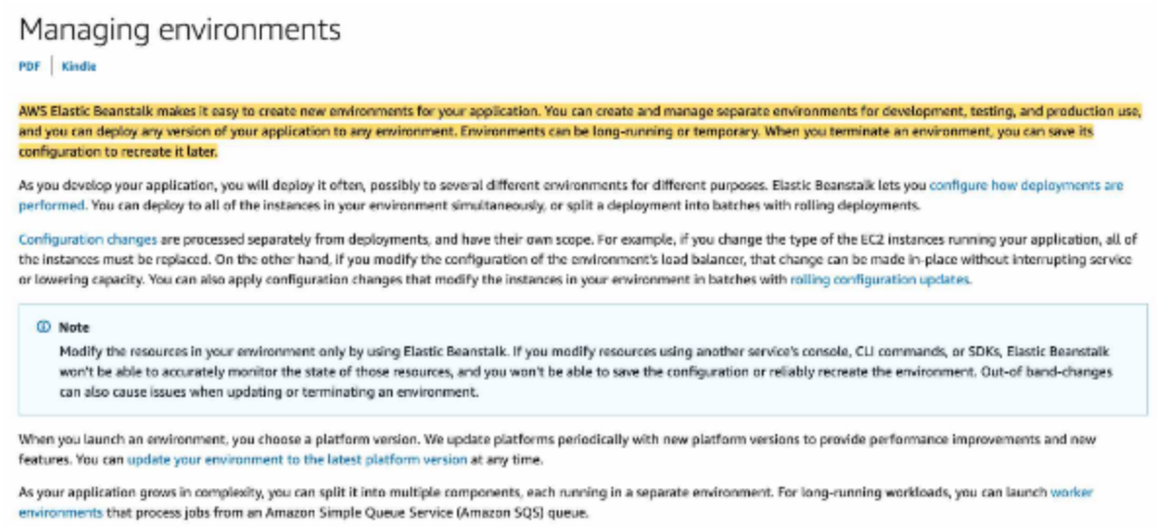
- ○ You cannot have multiple development environments in Elastic Beanstalk, just one development and one production environment
- ◉ Define a dev environment with a single instance and a 'load test' environment that has settings close to production environment **(Correct)**
- ○ Use only one Beanstalk environment and perform configuration changes using an Ansible script
- ○ Create an Application Load Balancer to route based on hostname so you can pass on parameters to the development Elastic Beanstalk environment. Create a file in .ebextensions/ to know how to handle the traffic coming from the ALB

Explanation
Correct option:

Define a dev environment with a single instance and a 'load test' environment that has settings close to production environment

AWS Elastic Beanstalk makes it easy to create new environments for your application. You can create and manage separate environments for development, testing, and production use, and you can deploy any version of your application to any environment. Environments can be long-running or temporary. When you terminate an environment, you can save its configuration to recreate it later.

It is common practice to have many environments for the same application. You can deploy multiple environments when you need to run multiple versions of an application. So for the given use-case, you can set up 'dev' and 'load test' environment.



Managing environments

PDF | Kindle

AWS Elastic Beanstalk makes it easy to create new environments for your application. You can create and manage separate environments for development, testing, and production use, and you can deploy any version of your application to any environment. Environments can be long-running or temporary. When you terminate an environment, you can save its configuration to recreate it later.

As you develop your application, you will deploy it often, possibly to several different environments for different purposes. Elastic Beanstalk lets you configure how deployments are performed. You can deploy to all of the instances in your environment simultaneously, or split a deployment into batches with rolling deployments.

Configuration changes are processed separately from deployments, and have their own scope. For example, if you change the type of the EC2 instances running your application, all of the instances must be replaced. On the other hand, if you modify the configuration of the environment's load balancer, that change can be made in-place without interrupting service or lowering capacity. You can also apply configuration changes that modify the instances in your environment in batches with rolling configuration updates.

ⓘ **Note**

Modify the resources in your environment only by using Elastic Beanstalk. If you modify resources using another service's console, CLI commands, or SDKs, Elastic Beanstalk won't be able to accurately monitor the state of those resources, and you won't be able to save the configuration or reliably recreate the environment. Out-of band-changes can also cause issues when updating or terminating an environment.

When you launch an environment, you choose a platform version. We update platforms periodically with new platform versions to provide performance improvements and new features. You can update your environment to the latest platform version at any time.

As your application grows in complexity, you can split it into multiple components, each running in a separate environment. For long-running workloads, you can launch worker environments that process jobs from an Amazon Simple Queue Service (Amazon SQS) queue.

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.html

You cannot have multiple development environments in Elastic Beanstalk, just one development, and one production environment - Incorrect, use the Create New Environment wizard in the AWS Management Console for BeanStalk to guide you on this.

Use only one Beanstalk environment and perform configuration changes using an Ansible script - Ansible is an open-source deployment tool that integrates with AWS. It allows us to deploy the infrastructure. Elastic Beanstalk provisions the servers that you need for hosting the application and it also handles multiple environments, so Beanstalk is a better option.

Create an Application Load Balancer to route based on hostname so you can pass on parameters to the development Elastic Beanstalk environment. Create a file in .ebextensions/ to know how to handle the traffic coming from the ALB - This is not a good design if you need to load test because you will have two versions on the same instances and may not be able to access resources in the system due to the load testing.

Reference:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.environments.html

Question 34: **Correct**

Your company manages hundreds of EC2 instances running on Linux OS. The instances are configured in several Availability Zones in the eu-west-3 region. Your manager has requested to collect system memory metrics on all EC2 instances using a script.

Which of the following solutions will help you collect this data?

- (●) Use a cron job on the instances that pushes the EC2 RAM statistics as a Custom metric into CloudWatch (Correct)
- ( ) Extract RAM statistics using the instance metadata
- ( ) Extract RAM statistics using X-Ray
- ( ) Extract RAM statistics from the standard CloudWatch metrics for EC2 instances

Explanation

Correct option:

"Use a cron job on the instances that pushes the EC2 RAM statistics as a Custom metric into CloudWatch"

The Amazon CloudWatch Monitoring Scripts for Amazon Elastic Compute Cloud (Amazon EC2) Linux-based instances demonstrate how to produce and consume Amazon CloudWatch custom metrics. These Perl scripts comprise a fully functional example that reports memory, swap, and disk space utilization metrics for a Linux instance. You can set a cron schedule for metrics reported to CloudWatch and report memory utilization to CloudWatch every x minutes.

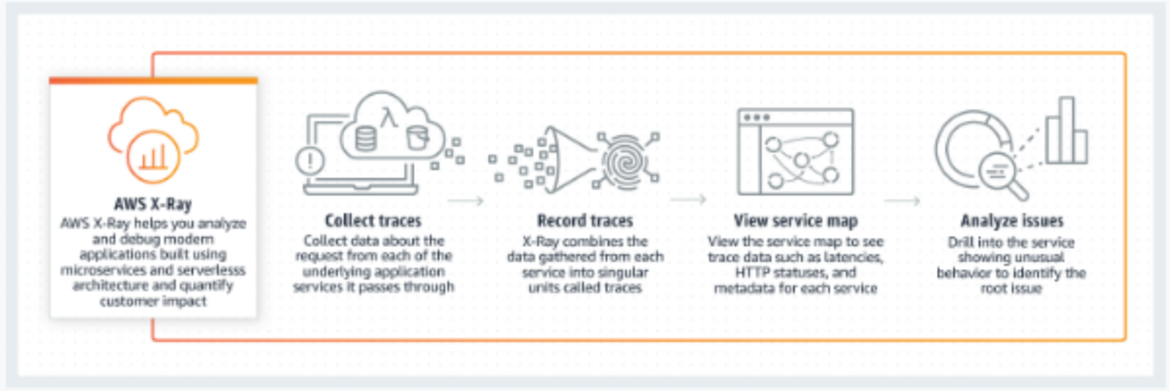
Incorrect options:

"Extract RAM statistics using the instance metadata" - Instance metadata is data about your instance that you can use to configure or manage the running instance. Instance metadata is divided into categories, for example, hostname, events, and security groups. The instance metadata can only provide the ID of the RAM disk specified at launch time. So this option is incorrect.

"Extract RAM statistics from the standard CloudWatch metrics for EC2 instances" - Amazon EC2 sends metrics to Amazon CloudWatch. By default, each data point covers the 5 minutes that follow the start time of activity for the instance. If you've enabled detailed monitoring, each data point covers the next minute of activity from the start time. The standard CloudWatch metrics don't have any metrics for memory utilization details.

"Extract RAM statistics using X-Ray" - AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application's underlying components.

How X-Ray Works:



via - https://aws.amazon.com/xray/

X-Ray cannot be used to extract RAM statistics for EC2 instances.

For more information visit
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/mon-scripts.html

Question 35: **Correct**

Your company leverages Amazon CloudFront to provide content via the internet to customers with low latency. Aside from latency, security is another concern and you are looking for help in enforcing end-to-end connections using HTTPS so that content is protected.

Which of the following options is available for HTTPS in AWS CloudFront?

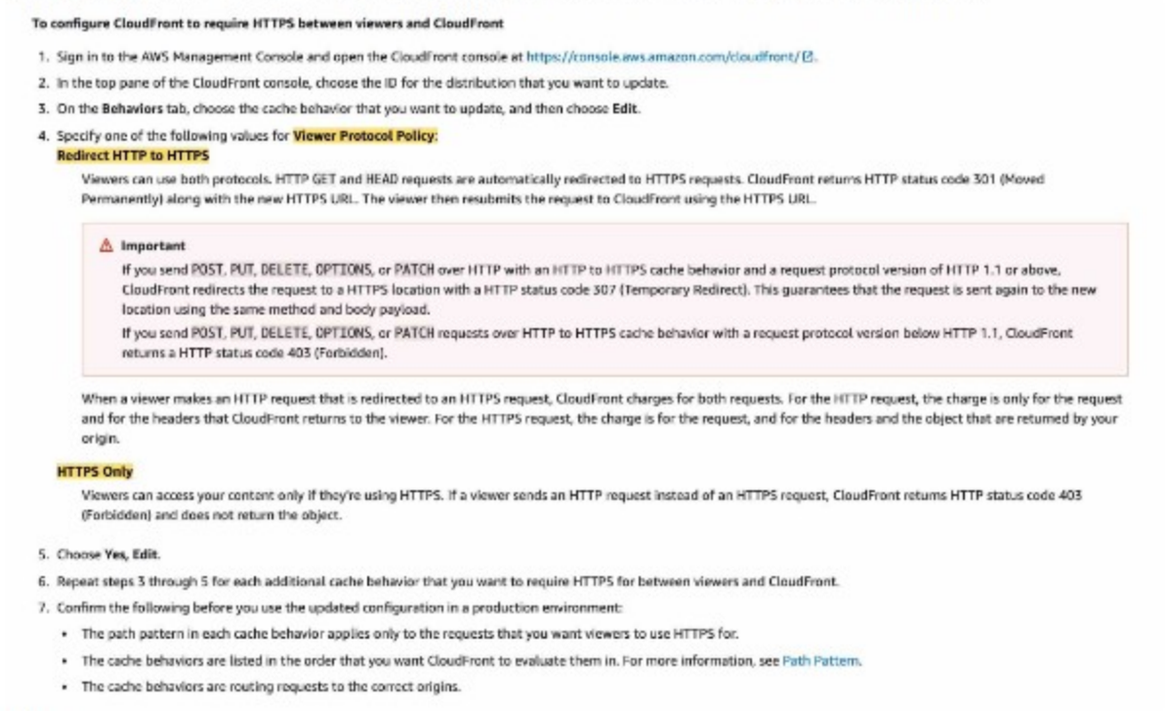
- (●) Between clients and CloudFront as well as between CloudFront and backend (Correct)
- ( ) Neither between clients and CloudFront nor between CloudFront and backend
- ( ) Between CloudFront and backend only
- ( ) Between clients and CloudFront only

Explanation
Correct option:

Between clients and CloudFront as well as between CloudFront and backend

For web distributions, you can configure CloudFront to require that viewers use HTTPS to request your objects, so connections are encrypted when CloudFront communicates with viewers.

Requiring HTTPS for Communication Between Viewers and CloudFront:



To configure CloudFront to require HTTPS between viewers and CloudFront

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. In the top pane of the CloudFront console, choose the ID for the distribution that you want to update.
3. On the **Behaviors** tab, choose the cache behavior that you want to update, and then choose **Edit**.
4. Specify one of the following values for **Viewer Protocol Policy**:
   Redirect HTTP to HTTPS
   Viewers can use both protocols. HTTP GET and HEAD requests are automatically redirected to HTTPS requests. CloudFront returns HTTP status code 301 (Moved Permanently) along with the new HTTPS URL. The viewer then resubmits the request to CloudFront using the HTTPS URL.

   ⚠ **Important**
   If you send POST, PUT, DELETE, OPTIONS, or PATCH over HTTP with an HTTP to HTTPS cache behavior and a request protocol version of HTTP 1.1 or above, CloudFront redirects the request to a HTTPS location with a HTTP status code 307 (Temporary Redirect). This guarantees that the request is sent again to the new location using the same method and body payload.
   If you send POST, PUT, DELETE, OPTIONS, or PATCH requests over HTTP to HTTPS cache behavior with a request protocol version below HTTP 1.1, CloudFront returns a HTTP status code 403 (Forbidden).

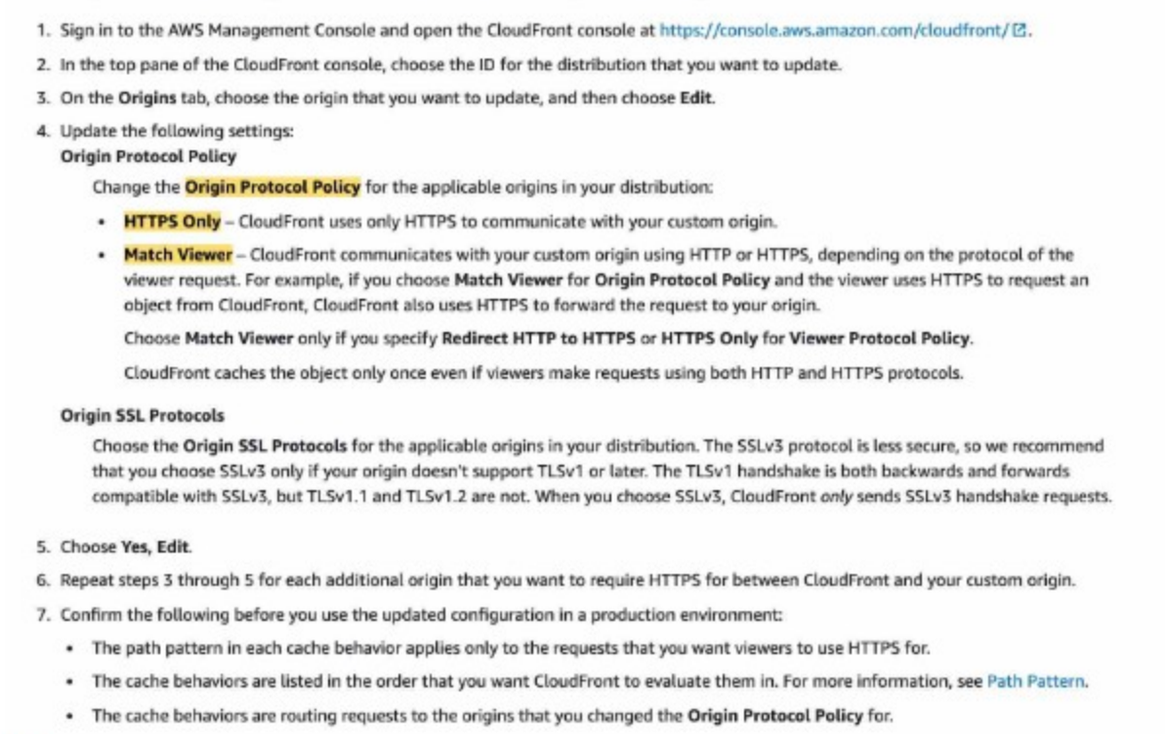
   When a viewer makes an HTTP request that is redirected to an HTTPS request, CloudFront charges for both requests. For the HTTP request, the charge is only for the request and for the headers that CloudFront returns to the viewer. For the HTTPS request, the charge is for the request, and for the headers and the object that are returned by your origin.

   HTTPS Only
   Viewers can access your content only if they're using HTTPS. If a viewer sends an HTTP request instead of an HTTPS request, CloudFront returns HTTP status code 403 (Forbidden) and does not return the object.
5. Choose **Yes, Edit**.
6. Repeat steps 3 through 5 for each additional cache behavior that you want to require HTTPS for between viewers and CloudFront.
7. Confirm the following before you use the updated configuration in a production environment:
   - The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
   - The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path Pattern.
   - The cache behaviors are routing requests to the correct origins.

via - https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-viewers-to-cloudfront.html

You also can configure CloudFront to use HTTPS to get objects from your origin, so connections are encrypted when CloudFront communicates with your origin.

Requiring HTTPS for Communication Between CloudFront and Your Custom Origin:



To configure CloudFront to require HTTPS between CloudFront and your custom origin

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/.
2. In the top pane of the CloudFront console, choose the ID for the distribution that you want to update.
3. On the **Origins** tab, choose the origin that you want to update, and then choose **Edit**.
4. Update the following settings:
   Origin Protocol Policy
   Change the **Origin Protocol Policy** for the applicable origins in your distribution:
   - **HTTPS Only** – CloudFront uses only HTTPS to communicate with your custom origin.
   - **Match Viewer** – CloudFront communicates with your custom origin using HTTP or HTTPS, depending on the protocol of the viewer request. For example, if you choose **Match Viewer** for **Origin Protocol Policy** and the viewer uses HTTPS to request an object from CloudFront, CloudFront also uses HTTPS to forward the request to your origin.
     Choose **Match Viewer** only if you specify **Redirect HTTP to HTTPS** or **HTTPS Only** for **Viewer Protocol Policy**.
     CloudFront caches the object only once even if viewers make requests using both HTTP and HTTPS protocols.

   Origin SSL Protocols
   Choose the **Origin SSL Protocols** for the applicable origins in your distribution. The SSLv3 protocol is less secure, so we recommend that you choose SSLv3 only if your origin doesn't support TLSv1 or later. The TLSv1 handshake is both backwards and forwards compatible with SSLv3, but TLSv1.1 and TLSv1.2 are not. When you choose SSLv3, CloudFront *only* sends SSLv3 handshake requests.
5. Choose **Yes, Edit**.
6. Repeat steps 3 through 5 for each additional origin that you want to require HTTPS for between CloudFront and your custom origin.
7. Confirm the following before you use the updated configuration in a production environment:
   - The path pattern in each cache behavior applies only to the requests that you want viewers to use HTTPS for.
   - The cache behaviors are listed in the order that you want CloudFront to evaluate them in. For more information, see Path Pattern.
   - The cache behaviors are routing requests to the origins that you changed the **Origin Protocol Policy** for.

via - https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-cloudfront-to-custom-origin.html

Incorrect options:

Between clients and CloudFront only - This is incorrect as you can choose to require HTTPS between CloudFront and your origin.

Between CloudFront and backend only - This is incorrect as you can choose to require HTTPS between viewers and CloudFront.

Neither between clients and CloudFront nor between CloudFront and backend - This is incorrect as you can choose HTTPS settings both for communication between viewers and CloudFront as well as between CloudFront and your origin.

References:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/secure-connections-supported-viewer-protocols-ciphers.html#secure-connections-supported-ciphers-cloudfront-to-origin

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-viewers-to-cloudfront.html

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-cloudfront-to-custom-origin.html

Question 36: **Incorrect**

A developer is migrating an on-premises application to AWS Cloud. The application currently processes user uploads and uploads them to a local directory on the server. All such file uploads must be saved and then made available to all instances in an Auto Scaling group.

As a Developer Associate, which of the following options would you recommend for this use-case?

- ○ Use Instance Store type of EC2 instances and share the files via file synchronization software
- ◉ Use Amazon EBS as the storage volume and share the files via file synchronization software (Incorrect)
- ○ Use Amazon EBS and configure the application AMI to use a snapshot of the same EBS instance while launching new instances
- ○ Use Amazon S3 and make code changes in the application so all uploads are put on S3 (Correct)

Explanation
Correct option:

Use Amazon S3 and make code changes in the application so all uploads are put on S3

Amazon S3 is an object storage built to store and retrieve any amount of data from anywhere on the Internet. It's a simple storage service that offers an extremely durable, highly available, and infinitely scalable data storage infrastructure at very low costs.

Amazon S3 provides a simple web service interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. Using this web service, you can easily build applications that make use of Internet storage.

You can use S3 PutObject API from the application to upload the objects in a single bucket, which is then accessible from all instances.

Incorrect options:

Use Amazon EBS and configure the application AMI to use a snapshot of the same EBS instance while launching new instances - Using EBS to share data between instances is not possible because EBS volume is tied to an instance by definition. Creating a snapshot would only manage to move the stale data into the new instances.

Use Instance Store type of EC2 instances and share the files via file synchronization software

Use Amazon EBS as the storage volume and share the files via file synchronization software

Technically you could use file synchronization software on EC2 instances with EBS or Instance Store type, but that involves a lot of development effort and still would not be as production-ready as just using S3. So both these options are incorrect.

Reference:

https://aws.amazon.com/s3/faqs/

Question 37: **Incorrect**

An IT company is using AWS CloudFormation to manage its IT infrastructure. It has created a template to provision a stack with a VPC and a subnet. The output value of this subnet has to be used in another stack.

As a Developer Associate, which of the following options would you suggest to provide this information to another stack?

- ○ Use Fn::Transform
- ◉ Use Fn::ImportValue (Incorrect)

- ○ Use 'Export' field in the Output section of the stack's template (Correct)
- ○ Use 'Expose' field in the Output section of the stack's template

Explanation
Correct option:

Use 'Export' field in the Output section of the stack's template

To share information between stacks, export a stack's output values. Other stacks that are in the same AWS account and region can import the exported values.

To export a stack's output value, use the Export field in the Output section of the stack's template. To import those values, use the Fn::ImportValue function in the template for the other stacks.

Incorrect options:

Use 'Expose' field in the Output section of the stack's template - 'Expose' is a made-up option, and only given as a distractor.

Use Fn::ImportValue - To import the values exported by another stack, we use the Fn::ImportValue function in the template for the other stacks. This function is not useful for the current scenario.

Use Fn::Transform - The intrinsic function Fn::Transform specifies a macro to perform custom processing on part of a stack template. Macros enable you to perform custom processing on templates, from simple actions like find-and-replace operations to extensive transformations of entire templates. This function is not useful for the current scenario.

Reference:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-stack-exports.html

Question 38: **Correct**

You are assigned as the new project lead for a web application that processes orders for customers. You want to integrate event-driven processing anytime data is modified or deleted and use a serverless approach using AWS Lambda for processing stream events.

Which of the following databases should you choose from?

- ◉ DynamoDB (Correct)
- ○ RDS
- ○ Kinesis
- ○ ElastiCache

Explanation
Correct option:

DynamoDB

A DynamoDB stream is an ordered flow of information about changes to items in a DynamoDB table. When you enable a stream on a table, DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table, and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared before and after they were modified, in near real-time.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attributes of the items that were modified.

DynamoDB Streams Overview:

Capturing Table Activity with DynamoDB Streams

PDF | Kindle | RSS

Many applications can benefit from the ability to capture changes to items stored in a DynamoDB table, at the point in time when such changes occur. The following are some example use cases:

- An application in one AWS Region modifies the data in a DynamoDB table. A second application in another Region reads these data modifications and writes the data to another table, creating a replica that stays in sync with the original table.
- A popular mobile app modifies data in a DynamoDB table, at the rate of thousands of updates per second. Another application captures and stores data about these updates, providing near-real-time usage metrics for the mobile app.
- A global multi-player game has a multi-master topology, storing data in multiple AWS Regions. Each master stays in sync by consuming and replaying the changes that occur in the remote Regions.
- An application automatically sends notifications to the mobile devices of all friends in a group as soon as one friend uploads a new picture.
- A new customer adds data to a DynamoDB table. This event invokes another application that sends a welcome email to the new customer.

DynamoDB Streams enables solutions such as these, and many others. DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours. Applications can access this log and view the data items as they appeared before and after they were modified, in near-real time.

Encryption at rest encrypts the data in DynamoDB streams. For more information, see DynamoDB Encryption at Rest.

A *DynamoDB stream* is an ordered flow of information about changes to items in a DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attributes of the items that were modified. A *stream record* contains information about a data modification to a single item in a DynamoDB table. You can configure the stream so that the stream records capture additional information, such as the "before" and "after" images of modified items.

DynamoDB Streams helps ensure the following:

- Each stream record appears exactly once in the stream.
- For each item that is modified in a DynamoDB table, the stream records appear in the same sequence as the actual modifications to the item.

DynamoDB Streams writes stream records in near-real time so that you can build applications that consume these streams and take action based on the contents.

via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

Incorrect options:

RDS - By itself, RDS cannot be used to stream events like DynamoDB, so this option is ruled out. However, you can use Amazon Kinesis for streaming data from RDS.

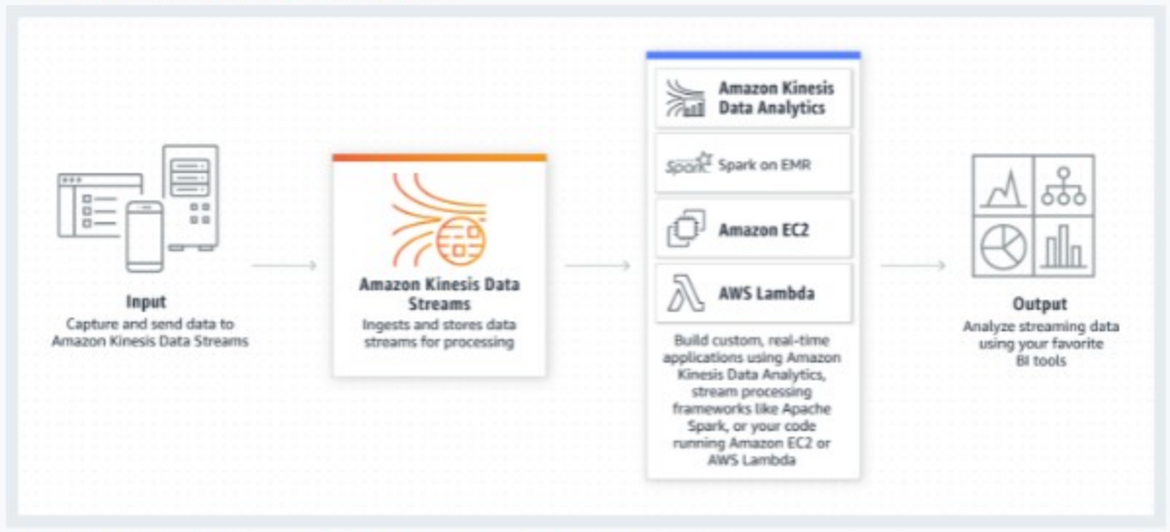
Please refer to this excellent blog for more details on using Kinesis for streaming data from RDS: https://aws.amazon.com/blogs/database/streaming-changes-in-a-database-with-amazon-kinesis/

ElastiCache - ElastiCache works as an in-memory data store and cache, it cannot be used to stream data like DynamoDB.

Kinesis - Kinesis is not a database, so this option is ruled out.

Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs. You can continuously add various types of data such as clickstreams, application logs, and social media to an Amazon Kinesis data stream from hundreds of thousands of sources.

How Kinesis Data Streams Work



via - https://aws.amazon.com/kinesis/data-streams/

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

Question 39: **Correct**

You have uploaded a zip file to AWS Lambda that contains code files written in Node.Js. When your function is executed you receive the following output, 'Error: Memory Size: 10,240 MB Max Memory Used'.

Which of the following explains the problem?

- ○ The uncompressed zip file exceeds AWS Lambda limits
- ◉ Your Lambda function ran out of RAM (Correct)
- ○ You have uploaded a zip file larger than 50 MB to AWS Lambda

○ Your zip file is corrupt

Explanation
Correct option:

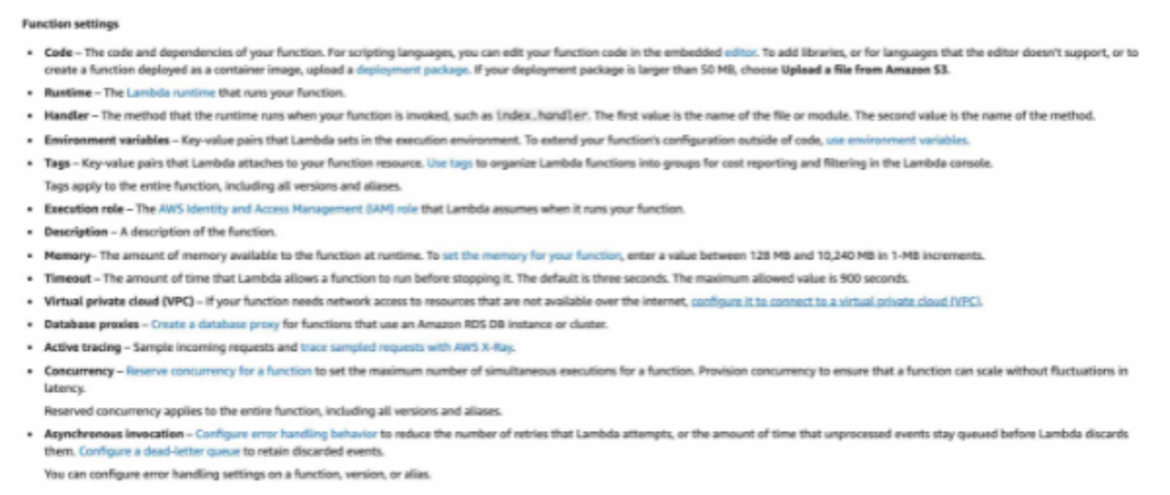
Your Lambda function ran out of RAM

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume.

How Lambda function works:

via - https://aws.amazon.com/lambda/

The maximum amount of memory available to the Lambda function at runtime is 10,240 MB. Your Lambda function was deployed with 10,240 MB of RAM, but it seems your code requested or used more than that, so the Lambda function failed.



Function settings

- **Code** – The code and dependencies of your function. For scripting languages, you can edit your function code in the embedded editor. To add libraries, or for languages that the editor doesn't support, or to create a function deployed as a container image, upload a deployment package. If your deployment package is larger than 50 MB, choose **Upload a file from Amazon S3**.
- **Runtime** – The Lambda runtime that runs your function.
- **Handler** – The method that the runtime runs when your function is invoked, such as `index.handler`. The first value is the name of the file or module. The second value is the name of the method.
- **Environment variables** – Key-value pairs that Lambda sets in the execution environment. To extend your function's configuration outside of code, use environment variables.
- **Tags** – Key-value pairs that Lambda attaches to your function resource. Use tags to organize Lambda functions into groups for cost reporting and filtering in the Lambda console.

  Tags apply to the entire function, including all versions and aliases.
- **Execution role** – The AWS Identity and Access Management (IAM) role that Lambda assumes when it runs your function.
- **Description** – A description of the function.
- **Memory** – The amount of memory available to the function at runtime. To set the memory for your function, enter a value between 128 MB and 10,240 MB in 1-MB increments.
- **Timeout** – The amount of time that Lambda allows a function to run before stopping it. The default is three seconds. The maximum allowed value is 900 seconds.
- **Virtual private cloud (VPC)** – If your function needs network access to resources that are not available over the internet, configure it to connect to a virtual private cloud (VPC).
- **Database proxies** – Create a database proxy for functions that use an Amazon RDS DB instance or cluster.
- **Active tracing** – Sample incoming requests and trace sampled requests with AWS X-Ray.
- **Concurrency** – Reserve concurrency for a function to set the maximum number of simultaneous executions for a function. Provision concurrency to ensure that a function can scale without fluctuations in latency.

  Reserved concurrency applies to the entire function, including all versions and aliases.
- **Asynchronous invocation** – Configure error handling behavior to reduce the number of retries that Lambda attempts, or the amount of time that unprocessed events stay queued before Lambda discards them. Configure a dead-letter queue to retain discarded events.

  You can configure error handling settings on a function, version, or alias.

via - https://docs.aws.amazon.com/lambda/latest/dg/configuration-console.html

Incorrect options:

Your zip file is corrupt - A memory size error states that Lambda was able to extract so the file is not corrupt

The uncompressed zip file exceeds AWS Lambda limits - This is not correct as your function was able to execute.

You have uploaded a zip file larger than 50 MB to AWS Lambda - This is not correct as your lambda function was able to execute

Reference:

https://docs.aws.amazon.com/lambda/latest/dg/configuration-console.html

Question 40: **Correct**

A cybersecurity company is publishing critical log data to a log group in Amazon CloudWatch Logs, which was created 3 months ago. The company must encrypt the log data using an AWS KMS customer master key (CMK), so any future data can be encrypted to meet the company's security guidelines.

How can the company address this use-case?

○ Use the AWS CLI `create-log-group` command and specify the KMS key ARN

○ Use the AWS CLI `describe-log-groups` command and specify the KMS key ARN

◉ Use the AWS CLI `associate-kms-key` command and specify the KMS key ARN (Correct)

○ Enable the encrypt feature on the log group via the CloudWatch Logs console

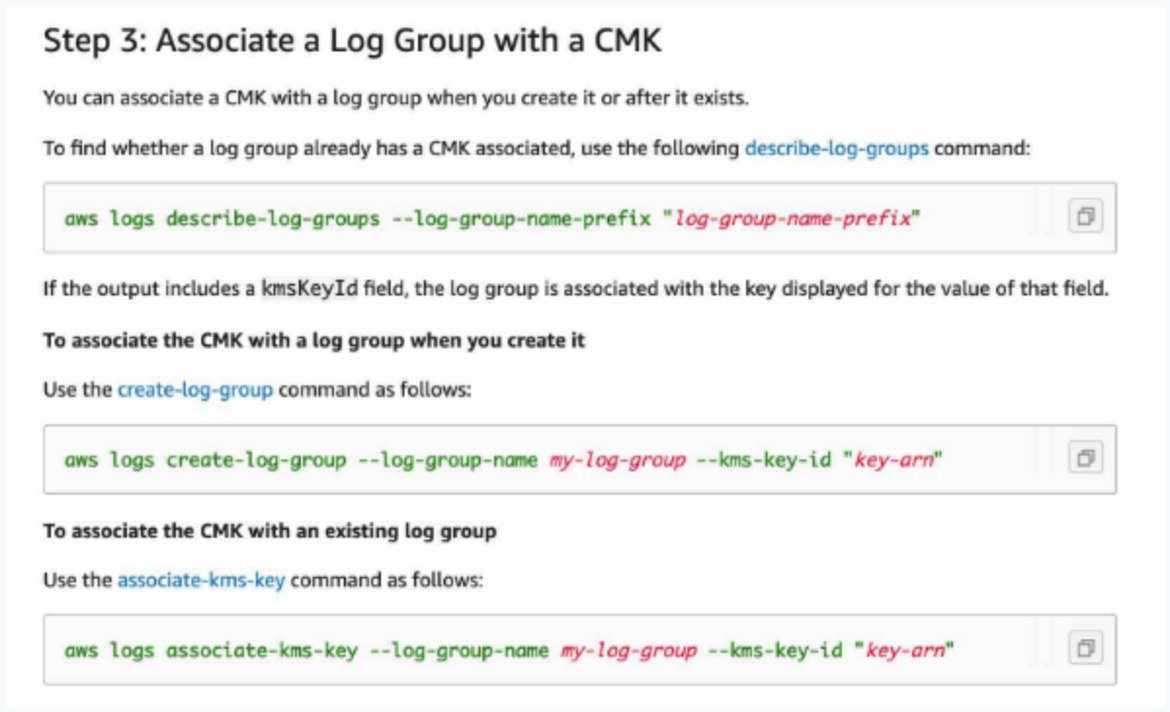
Explanation
Correct option:

Use the AWS CLI `associate-kms-key` command and specify the KMS key ARN

Log group data is always encrypted in CloudWatch Logs. You can optionally use AWS AWS Key Management Service for this encryption. If you do, the encryption is done using an AWS KMS (AWS KMS) customer master key (CMK). Encryption using AWS KMS is enabled at the log group level, by associating a CMK with a log group, either when you create the log group or after it exists.

After you associate a CMK with a log group, all newly ingested data for the log group is encrypted using the CMK. This data is stored in an encrypted format throughout its retention period. CloudWatch Logs decrypts this data whenever it is requested. CloudWatch Logs must have permissions for the CMK whenever encrypted data is requested.

To associate the CMK with an existing log group, you can use the `associate-kms-key` command.



Step 3: Associate a Log Group with a CMK

You can associate a CMK with a log group when you create it or after it exists.

To find whether a log group already has a CMK associated, use the following describe-log-groups command:

```
aws logs describe-log-groups --log-group-name-prefix "log-group-name-prefix"
```

If the output includes a `kmsKeyId` field, the log group is associated with the key displayed for the value of that field.

To associate the CMK with a log group when you create it

Use the create-log-group command as follows:

```
aws logs create-log-group --log-group-name my-log-group --kms-key-id "key-arn"
```

To associate the CMK with an existing log group

Use the associate-kms-key command as follows:

```
aws logs associate-kms-key --log-group-name my-log-group --kms-key-id "key-arn"
```

via - https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/encrypt-log-data-kms.html

Incorrect options:

Enable the encrypt feature on the log group via the CloudWatch Logs console - CloudWatch Logs console does not have an option to enable encryption for a log group.

Use the AWS CLI `describe-log-groups` command and specify the KMS key ARN - You can use the `describe-log-groups` command to find whether a log group already has a CMK associated with it.

Use the AWS CLI `create-log-group` command and specify the KMS key ARN - You can use the `create-log-group` command to associate the CMK with a log group when you create it.

Reference:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/encrypt-log-data-kms.html

Question 41: **Correct**

You are a system administrator whose company recently moved its production application to AWS and migrated data from MySQL to AWS DynamoDB. You are adding new tables to AWS DynamoDB and need to allow your application to query your data by the primary key and an alternate key. This option must be added when first creating tables otherwise changes cannot be made afterward.

Which of the following actions should you take?

- ○ Migrate away from DynamoDB
- ○ Call Scan
- ○ Create a GSI
- ◉ Create a LSI (Correct)

Explanation
Correct option:

Create an LSI

LSI stands for Local Secondary Index. Some applications only need to query data using the base table's primary key; however, there may be situations where an alternate sort key would be helpful. To give your application a choice of sort keys, you can create one or more local secondary indexes on a table and issue Query or Scan requests against these indexes.

Differences between GSI and LSI:

DynamoDB supports two types of secondary indexes:

- **Global secondary index** — An index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions. A global secondary index is stored in its own partition space away from the base table and scales separately from the base table.
- **Local secondary index** — An index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.

You should consider your application's requirements when you determine which type of index to use. The following table shows the main differences between a global secondary index and a local secondary index.

Characteristic	Global Secondary Index	Local Secondary Index
Key Schema	The primary key of a global secondary index can be either simple (partition key) or composite (partition key and sort key).	The primary key of a local secondary index must be composite (partition key and sort key).
Key Attributes	The index partition key and sort key (if present) can be any base table attributes of type string, number, or binary.	The partition key of the index is the same attribute as the partition key of the base table. The sort key can be any base table attribute of type string, number, or binary.
Size Restrictions Per Partition Key Value	There are no size restrictions for global secondary indexes.	For each partition key value, the total size of all indexed items must be 10 GB or less.
Online Index Operations	Global secondary indexes can be created at the same time that you create a table. You can also add a new global secondary index to an existing table, or delete an existing global secondary index. For more information, see Managing Global Secondary Indexes.	Local secondary indexes are created at the same time that you create a table. You cannot add a local secondary index to an existing table, nor can you delete any local secondary indexes that currently exist.
Queries and Partitions	A global secondary index lets you query over the entire table, across all partitions.	A local secondary index lets you query over a single partition, as specified by the partition key value in the query.
Read Consistency	Queries on global secondary indexes support eventual consistency only.	When you query a local secondary index, you can choose either eventual consistency or strong consistency.
Provisioned Throughput Consumption	Every global secondary index has its own provisioned throughput settings for read and write activity. Queries or scans on a global secondary index consume capacity units from the index, not from the base table. The same holds true for global secondary index updates due to table writes.	Queries or scans on a local secondary index consume read capacity units from the base table. When you write to a table, its local secondary indexes are also updated; these updates consume write capacity units from the base table.
Projected Attributes	With global secondary index queries or scans, you can only request the attributes that are projected into the index. DynamoDB does not fetch any attributes from the table.	If you query or scan a local secondary index, you can request attributes that are not projected in to the index. DynamoDB automatically fetches those attributes from the table.

via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html

Incorrect options:

Call Scan - Scan is an operation on the data. Once you create your local secondary indexes on a table you can then issue Scan requests again.

Create a GSI - GSI (Global Secondary Index) is an index with a partition key and a sort key that can be different from those on the base table.

Migrate away from DynamoDB - Migrating to another database that is not NoSQL may cause you to make changes that require substantial code changes.

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html

Question 42: **Incorrect**

Your company manages MySQL databases on EC2 instances to have full control. Applications on other EC2 instances managed by an ASG make requests to these databases to get information that displays data on dashboards viewed on mobile phones, tablets, and web browsers.

Your manager would like to scale your Auto Scaling group based on the number of requests per minute. How can you achieve this?

- ○ Attach additional Elastic File Storage
- ○ You create a CloudWatch custom metric and build an alarm to scale your ASG (Correct)
- ○ Attach an Elastic Load Balancer
- ◉ You enable detailed monitoring and use that to scale your ASG (Incorrect)

Explanation
Correct option:

You create a CloudWatch custom metric and build an alarm to scale your ASG

Here we need to scale on the metric "number of requests per minute", which is a custom metric we need to create, as it's not readily available in CloudWatch.

Metrics produced by AWS services are standard resolution by default. When you publish a custom metric, you can define it as either standard resolution or high resolution. When you publish a high-resolution metric, CloudWatch stores it with a resolution of 1 second, and you can read and retrieve it with a period of 1 second, 5 seconds, 10 seconds, 30 seconds, or any multiple of 60 seconds.

Incorrect options:

Attach an Elastic Load Balancer - This is not what you need for auto-scaling. An Elastic Load Balancer distributes workloads across multiple compute resources and checks your instances' health status to name a few, but it does not automatically increase and decrease the number of instances based on the application requirement.

Attach additional Elastic File Storage - This is a file storage service designed for performance. Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on-demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth. This cannot be used to facilitate auto-scaling.

How EFS Works:

via - https://aws.amazon.com/efs/

You enable detailed monitoring and use that to scale your ASG - The detailed monitoring metrics won't provide information about database /application-level requests per minute, so this option is not correct.

Reference:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/publishingMetrics.html

Question 43: **Correct**

A communication platform serves millions of customers and deploys features in a production environment on AWS via CodeDeploy. You are reviewing scripts for the deployment process located in the AppSec file.

Which of the following options lists the correct order of lifecycle events?

- ValidateService => BeforeInstall =>DownloadBundle => ApplicationStart
- BeforeInstall => ApplicationStart => DownloadBundle => ValidateService
- **DownloadBundle => BeforeInstall => ApplicationStart => ValidateService** (Correct)
- BeforeInstall => ValidateService =>DownloadBundle => ApplicationStart

Explanation
Correct option:

DownloadBundle => BeforeInstall => ApplicationStart => ValidateService

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.

You can specify one or more scripts to run in a hook. Each hook for a lifecycle event is specified with a string on a separate line.

Please review the correct order of lifecycle events:

Lifecycle Event Hook Availability

The following table lists the lifecycle event hooks available for each deployment and rollback scenario.

Lifecycle event name	In-place deployment[1]	Blue/green deployment: Original instances	Blue/green deployment: Replacement instances	Blue/green deployment rollback: Original instances	Blue/green deployment rollback: Replacement instances
ApplicationStop	✓		✓		
DownloadBundle[2]	✓		✓		
BeforeInstall	✓		✓		
Install[2]	✓		✓		
AfterInstall	✓		✓		
ApplicationStart	✓		✓		
ValidateService	✓		✓		
BeforeBlockTraffic	✓	✓			✓
BlockTraffic[2]	✓	✓			✓
AfterBlockTraffic	✓	✓			✓
BeforeAllowTraffic	✓		✓	✓	
AllowTraffic[2]	✓		✓	✓	

via - https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#reference-appspec-file-structure-hooks-run-order

Incorrect options:

BeforeInstall => ApplicationStart => DownloadBundle => ValidateService

ValidateService => BeforeInstall =>DownloadBundle => ApplicationStart

BeforeInstall => ValidateService =>DownloadBundle => ApplicationStart

These three options contradict the details provided in the explanation above, so these options are not correct.

Reference:

https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#reference-appspec-file-structure-hooks-run-order

Question 44: **Correct**

A company developed an app-based service for citizens to book transportation rides in the local community. The platform is running on AWS EC2 instances and uses Amazon Relational Database Service (RDS) for storing transportation data. A new feature has been requested where receipts would be emailed to customers with PDF attachments retrieved from Amazon Simple Storage Service (S3).

Which of the following options will provide EC2 instances with the right permissions to upload files to Amazon S3 and generate S3 Signed URL?

- ○ Run `aws configure` on the EC2 instance
- ○ EC2 User Data
- ◉ Create an IAM Role for EC2 (Correct)
- ○ CloudFormation

Explanation
Correct option:

Create an IAM Role for EC2

IAM roles have been incorporated so that your applications can securely make API requests from your instances, without requiring you to manage the security credentials that the applications use. Instead of creating and distributing your AWS credentials, you can delegate permission to make API requests using IAM roles.

Amazon EC2 uses an instance profile as a container for an IAM role. When you create an IAM role using the IAM console, the console creates an instance profile automatically and gives it the same name as the role to which it corresponds.

Incorrect options:

EC2 User Data - You can specify user data when you launch an instance and you would not want to hard code the AWS credentials in the user data.

Run `aws configure` on the EC2 instance - When you first configure the CLI you have to run this command, afterward you should not need to if you want to obtain credentials to authenticate to other AWS services. An IAM role will receive

temporary credentials for you so you can focus on using the CLI to get access to other AWS services if you have the permissions.

CloudFormation - AWS CloudFormation gives developers and businesses an easy way to create a collection of related AWS and third-party resources and provision them in an orderly and predictable fashion.

How CloudFormation Works:

via - https://aws.amazon.com/cloudformation/

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

Question 45: **Incorrect**

You are designing a high-performance application that requires millions of connections. You have several EC2 instances running Apache2 web servers and the application will require capturing the user's source IP address and source port without the use of X-Forwarded-For.

Which of the following options will meet your needs?

- (●) Classic Load Balancer **(Incorrect)**
- ( ) Elastic Load Balancer
- ( ) Application Load Balancer
- ( ) Network Load Balancer **(Correct)**

Explanation
Correct option:

Network Load Balancer

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration. Incoming connections remain unmodified, so application software need not support X-Forwarded-For.

Network Load Balancer overview

A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second. After the load balancer receives a connection request, it selects a target from the target group for the default rule. It attempts to open a TCP connection to the selected target on the port specified in the listener configuration.

When you enable an Availability Zone for the load balancer, Elastic Load Balancing creates a load balancer node in the Availability Zone. By default, each load balancer node distributes traffic across the registered targets in its Availability Zone only. If you enable cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. For more information, see Availability Zones.

If you enable multiple Availability Zones for your load balancer and ensure that each target group has at least one target in each enabled Availability Zone, this increases the fault tolerance of your applications. For example, if one or more target groups does not have a healthy target in an Availability Zone, we remove the IP address for the corresponding subnet from DNS, but the load balancer nodes in the other Availability Zones are still available to route traffic. If a client doesn't honor the time-to-live (TTL) and sends requests to the IP address after it is removed from DNS, the requests fail.

For TCP traffic, the load balancer selects a target using a flow hash algorithm based on the protocol, source IP address, source port, destination IP address, destination port, and TCP sequence number. The TCP connections from a client have different source ports and sequence numbers, and can be routed to different targets. Each individual TCP connection is routed to a single target for the life of the connection.

For UDP traffic, the load balancer selects a target using a flow hash algorithm based on the protocol, source IP address, source port, destination IP address, and destination port. A UDP flow has the same source and destination, so it is consistently routed to a single target throughout its lifetime. Different UDP flows have different source IP addresses and ports, so they can be routed to different targets.

Elastic Load Balancing creates a network interface for each Availability Zone you enable. Each load balancer node in the Availability Zone uses this network interface to get a static IP address. When you create an Internet-facing load balancer, you can optionally associate one Elastic IP address per subnet.

When you create a target group, you specify its target type, which determines whether you register targets by instance ID or IP address. If you register targets by instance ID, the source IP addresses of the clients are preserved and provided to your applications. If you register targets by IP address, the source IP addresses are the private IP addresses of the load balancer nodes.

You can add and remove targets from your load balancer as your needs change, without disrupting the overall flow of requests to your application. Elastic Load Balancing scales your load balancer as traffic to your application changes over time. Elastic Load Balancing can scale to the vast majority of workloads automatically.

You can configure health checks, which are used to monitor the health of the registered targets so that the load balancer can send requests only to the healthy targets.

via -

https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Incorrect options:

Application Load Balancer - An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it evaluates the listener rules in priority order to determine which rule to apply and then selects a target from the target group for the rule action.

One of many benefits of the Application Load Balancer is its support for path-based routing. You can configure rules for your listener that forward requests based on the URL in the request. This enables you to structure your application as smaller services, and route requests to the correct service based on the content of the URL. For needs relating to network traffic go with Network Load Balancer.

Elastic Load Balancer - Elastic Load Balancing is the service itself that offers different types of load balancers.

Classic Load Balancer - It is a basic load balancer that distributes traffic. If your account was created before 2013-12-04, your account supports EC2-Classic instances and you will benefit in using this type of load balancer. The classic load balancer can be used regardless of when your account was created and whether you use EC2-Classic or whether your instances are in a VPC but just remember its the basic load balancer AWS offers and not advanced as the others.

Reference:

https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Question 46: **Correct**

A website serves static content from an Amazon Simple Storage Service (Amazon S3) bucket and dynamic content from an application load balancer. The user base is spread across the world and latency should be minimized for a better user experience.

Which technology/service can help access the static and dynamic content while keeping the data latency low?

- (●) Configure CloudFront with multiple origins to serve both static and dynamic content at low latency to global users (Correct)
- ( ) Use Global Accelerator to transparently switch between S3 bucket and load balancer for different data needs
- ( ) Use CloudFront's Origin Groups to group both static and dynamic requests into one request for further processing
- ( ) Use CloudFront's Lambda@Edge feature to server data from S3 buckets and load balancer programmatically on-the-fly

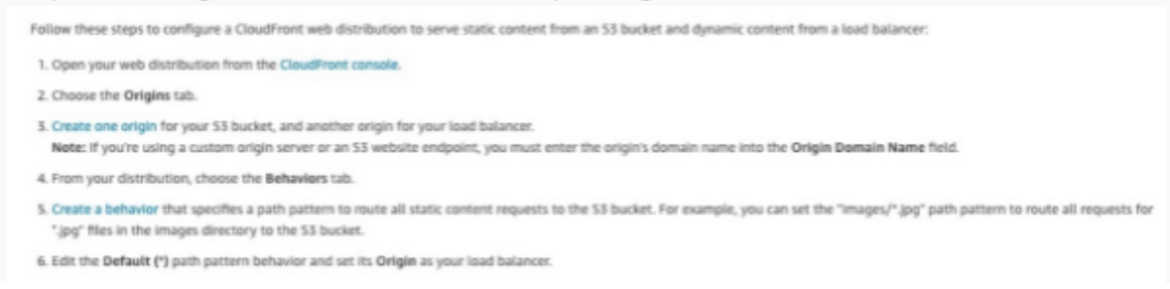
Explanation
Correct option:

Configure CloudFront with multiple origins to serve both static and dynamic content at low latency to global users

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

You can configure a single CloudFront web distribution to serve different types of requests from multiple origins.

Steps to configure CLoudFront for multiple origins:


Follow these steps to configure a CloudFront web distribution to serve static content from an S3 bucket and dynamic content from a load balancer:

1. Open your web distribution from the CloudFront console.
2. Choose the **Origins** tab.
3. Create one origin for your S3 bucket, and another origin for your load balancer.
   Note: If you're using a custom origin server or an S3 website endpoint, you must enter the origin's domain name into the **Origin Domain Name** field.
4. From your distribution, choose the **Behaviors** tab.
5. Create a behavior that specifies a path pattern to route all static content requests to the S3 bucket. For example, you can set the "images/*.jpg" path pattern to route all requests for *.jpg" files in the images directory to the S3 bucket.
6. Edit the **Default (*)** path pattern behavior and set its **Origin** as your load balancer.

via - https://aws.amazon.com/premiumsupport/knowledge-center/cloudfront-distribution-serve-content/

Incorrect options:

Use CloudFront's Lambda@Edge feature to server data from S3 buckets and load balancer programmatically on-the-fly - AWS Lambda@Edge is a general-purpose serverless compute feature that supports a wide range of computing needs and customizations. Lambda@Edge is best suited for computationally intensive operations. This is not relevant for the given use case.

Use Global Accelerator to transparently switch between S3 bucket and load balancer for different data needs - AWS Global Accelerator is a networking service that improves the performance of your users' traffic by up to 60% using Amazon Web Services' global network infrastructure.

With Global Accelerator, you are provided two global static public IPs that act as a fixed entry point to your application, improving availability. On the back end, add or remove your AWS application endpoints, such as Application Load Balancers, Network Load Balancers, EC2 Instances, and Elastic IPs without making user-facing changes. Global Accelerator automatically re-routes your traffic to your nearest healthy available endpoint to mitigate endpoint failure.

CloudFront improves performance for both cacheable content (such as images and videos) and dynamic content (such as API acceleration and dynamic site delivery). Global Accelerator is a good fit for non-HTTP use cases, such as gaming (UDP), IoT (MQTT), or Voice over IP, as well as for HTTP use cases that specifically require static IP addresses or deterministic, fast regional failover.

Global Accelerator is not relevant for the given use-case.

Use CloudFront's Origin Groups to group both static and dynamic requests into one request for further processing - You can set up CloudFront with origin failover for scenarios that require high availability. To get started, you create an Origin Group with two origins: a primary and a secondary. If the primary origin is unavailable or returns specific HTTP response status codes that indicate a failure, CloudFront automatically switches to the secondary origin. Origin Groups are for origin failure scenarios and not for request routing.

References:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-overview.html

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html

Question 47: **Incorrect**

An organization recently began using AWS CodeCommit for its source control service. A compliance security team visiting the organization was auditing the software development process and noticed developers making many git push commands within their development machines. The compliance team requires that encryption be used for this activity.

How can the organization ensure source code is encrypted in transit and at rest?

- ◉ Enable KMS encryption **(Incorrect)**
- ○ Use a git command line hook to encrypt the code client side
- ○ Use AWS Lambda as a hook to encrypt the pushed code
- ○ Repositories are automatically encrypted at rest **(Correct)**

Explanation
Correct option:

Repositories are automatically encrypted at rest

Data in AWS CodeCommit repositories is encrypted in transit and at rest. When data is pushed into an AWS CodeCommit repository (for example, by calling git push), AWS CodeCommit encrypts the received data as it is stored in the repository.

AWS Key Management Service and encryption for AWS CodeCommit repositories

PDF | Kindle | RSS

Data in CodeCommit repositories is encrypted in transit and at rest. When data is pushed into a CodeCommit repository (for example, by calling **git push**), CodeCommit encrypts the received data as it is stored in the repository. When data is pulled from a CodeCommit repository (for example, by calling **git pull**), CodeCommit decrypts the data and then sends it to the caller. This assumes the IAM user associated with the push or pull request has been authenticated by AWS. Data sent or received is transmitted using the HTTPS or SSH encrypted network protocols.

The first time you create a CodeCommit repository in a new AWS Region in your AWS account, CodeCommit creates an AWS-managed key (the `aws/codecommit` key) in that same AWS Region in AWS Key Management Service (AWS KMS). This key is used only by CodeCommit (the `aws/codecommit` key). It is stored in your AWS account. CodeCommit uses this AWS-managed key to encrypt and decrypt the data in this and all other CodeCommit repositories within that region in your AWS account.

Important

CodeCommit performs the following AWS KMS actions against the default `aws/codecommit` key. An IAM user does not need explicit permissions for these actions, but the user must not have any attached policies that deny these actions for the `aws/codecommit` key. When you create your first repository, your AWS account must not have any of the following permissions set to `deny`:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:ReEncrypt"`
- `"kms:GenerateDataKey"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:DescribeKey"`

via - https://docs.aws.amazon.com/codecommit/latest/userguide/encryption.html

Incorrect options:

Enable KMS encryption - You don't have to. The first time you create an AWS CodeCommit repository in a new region in your AWS account, CodeCommit creates an AWS-managed key in that same region in AWS Key Management Service (AWS KMS) that is used only by CodeCommit.

Use AWS Lambda as a hook to encrypt the pushed code - This is not needed as CodeCommit handles it for you.

Use a git command line hook to encrypt the code client-side - This is not needed as CodeCommit handles it for you.

Reference:

https://docs.aws.amazon.com/codecommit/latest/userguide/encryption.html

For more information visit
https://docs.aws.amazon.com/codecommit/latest/userguide/encryption.html

Question 48: **Correct**

As part of internal regulations, you must ensure that all communications to Amazon S3 are encrypted.

For which of the following encryption mechanisms will a request get rejected if the connection is not using HTTPS?

- ○ SSE-S3
- ◉ SSE-C (Correct)
- ○ SSE-KMS
- ○ Client Side Encryption

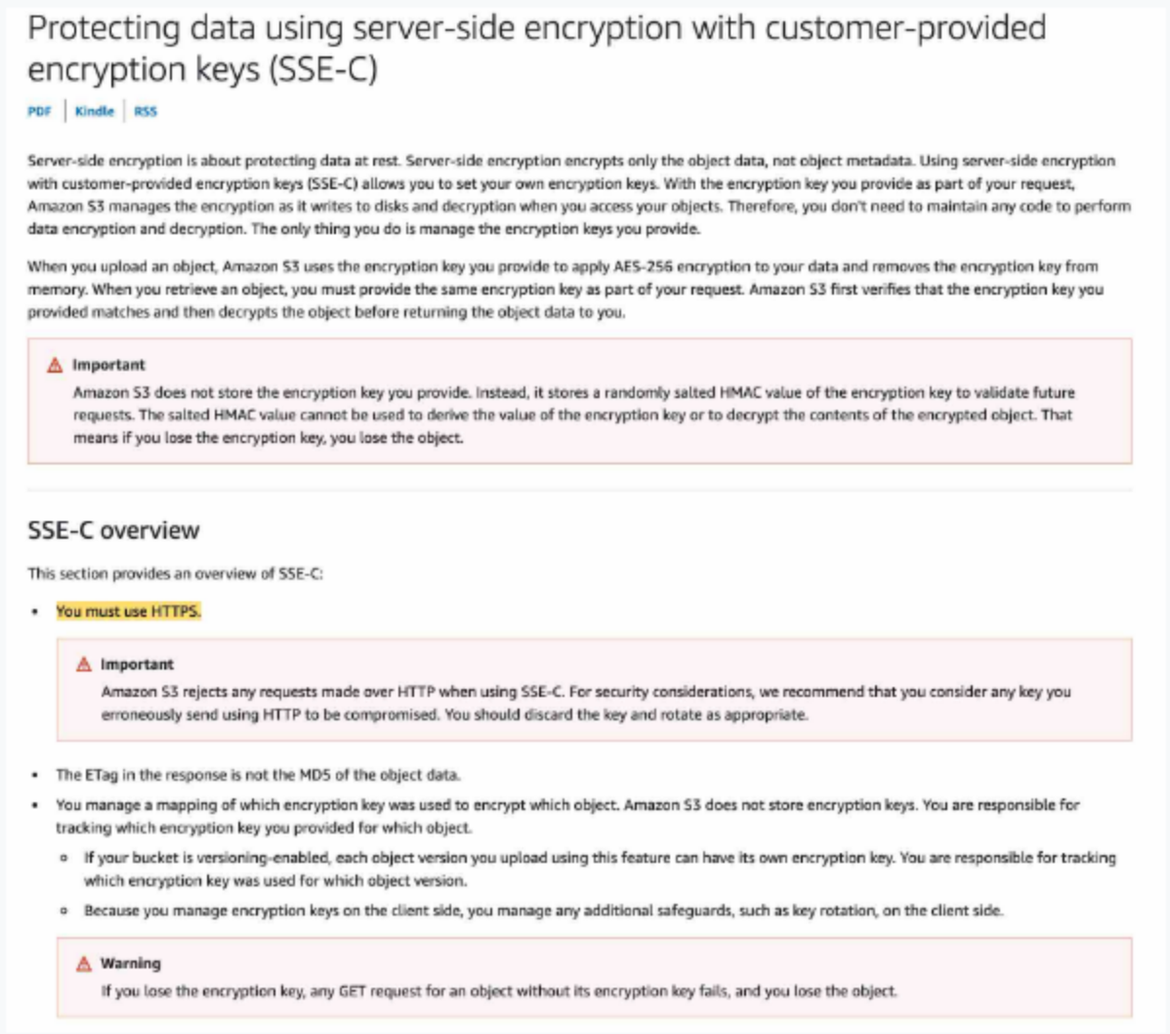
Explanation
Correct option:

SSE-C

Server-side encryption is about protecting data at rest. Server-side encryption encrypts only the object data, not object metadata. Using server-side encryption with customer-provided encryption keys (SSE-C) allows you to set your encryption keys.

When you upload an object, Amazon S3 uses the encryption key you provide to apply AES-256 encryption to your data and removes the encryption key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. Amazon S3 first verifies that the encryption key you provided matches and then decrypts the object before returning the object data to you.

Amazon S3 will reject any requests made over HTTP when using SSE-C. For security considerations, AWS recommends that you consider any key you send erroneously using HTTP to be compromised.



Protecting data using server-side encryption with customer-provided encryption keys (SSE-C)

PDF | Kindle | RSS

Server-side encryption is about protecting data at rest. Server-side encryption encrypts only the object data, not object metadata. Using server-side encryption with customer-provided encryption keys (SSE-C) allows you to set your own encryption keys. With the encryption key you provide as part of your request, Amazon S3 manages the encryption as it writes to disks and decryption when you access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing you do is manage the encryption keys you provide.

When you upload an object, Amazon S3 uses the encryption key you provide to apply AES-256 encryption to your data and removes the encryption key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. Amazon S3 first verifies that the encryption key you provided matches and then decrypts the object before returning the object data to you.

Important

Amazon S3 does not store the encryption key you provide. Instead, it stores a randomly salted HMAC value of the encryption key to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means if you lose the encryption key, you lose the object.

SSE-C overview

This section provides an overview of SSE-C:

- You must use HTTPS.

Important

Amazon S3 rejects any requests made over HTTP when using SSE-C. For security considerations, we recommend that you consider any key you erroneously send using HTTP to be compromised. You should discard the key and rotate as appropriate.

- The ETag in the response is not the MD5 of the object data.
- You manage a mapping of which encryption key was used to encrypt which object. Amazon S3 does not store encryption keys. You are responsible for tracking which encryption key you provided for which object.
  - If your bucket is versioning-enabled, each object version you upload using this feature can have its own encryption key. You are responsible for tracking which encryption key was used for which object version.
  - Because you manage encryption keys on the client side, you manage any additional safeguards, such as key rotation, on the client side.

Warning

If you lose the encryption key, any GET request for an object without its encryption key fails, and you lose the object.

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerSideEncryptionCustomerKeys.html

Incorrect options:

SSE-KMS - It is not mandatory to use HTTPS.

Client-Side Encryption - Client-side encryption is the act of encrypting data before sending it to Amazon S3. It is not mandatory to use HTTPS for this.

SSE-S3 - It is not mandatory to use HTTPS. Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers.

References:

https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerSideEncryptionCustomerKeys.html

Question 49: **Correct**

You have an Amazon Kinesis Data Stream with 10 shards, and from the metrics, you are well below the throughput utilization of 10 MB per second to send data. You send 3 MB per second of data and yet you are receiving ProvisionedThroughputExceededException errors frequently.

What is the likely cause of this?

- Metrics are slow to update
- You have too many shards
- The partition key that you have selected isn't distributed enough (Correct)
- The data retention period is too long

Explanation
Correct option:

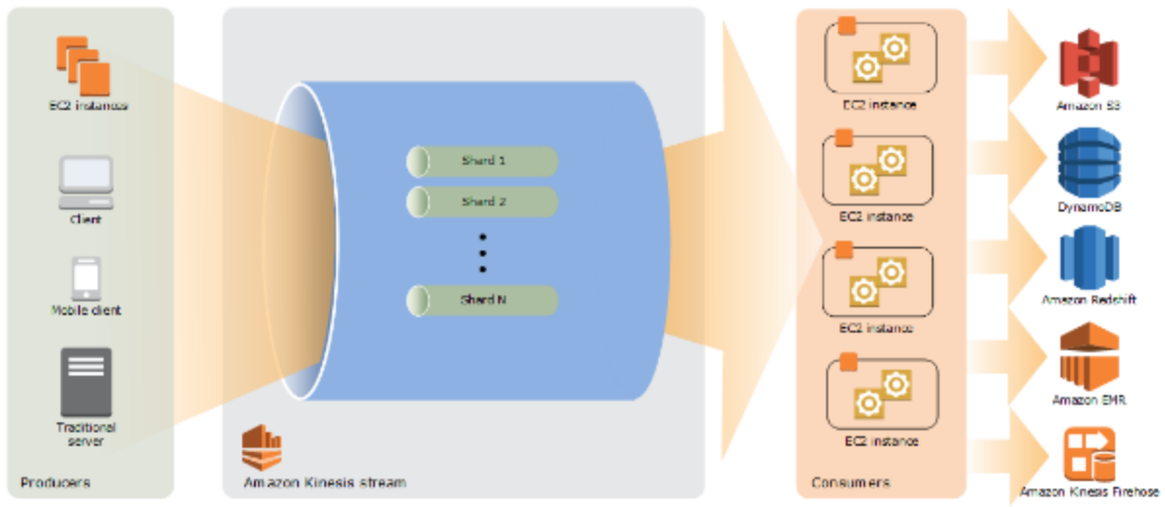
The partition key that you have selected isn't distributed enough

Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs.

A Kinesis data stream is a set of shards. A shard is a uniquely identified sequence of data records in a stream. A stream is composed of one or more shards, each of which provides a fixed unit of capacity.

The partition key is used by Kinesis Data Streams to distribute data across shards. Kinesis Data Streams segregates the data records that belong to a stream into multiple shards, using the partition key associated with each data record to determine the shard to which a given data record belongs.

Kinesis Data Streams Overview:



via - https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html

For the given use-case, as the partition key is not distributed enough, all the data is getting skewed at a few specific shards and not leveraging the entire cluster of shards.

You can also use metrics to determine which are your "hot" or "cold" shards, that is, shards that are receiving much more data, or much less data, than expected. You could then selectively split the hot shards to increase capacity for the hash keys that target those shards. Similarly, you could merge cold shards to make better use of their unused capacity.

Incorrect options:

Metrics are slow to update - Metrics are a CloudWatch concept. This option has been added as a distractor.

You have too many shards - Too many shards is not the issue as you would see a LimitExceededException in that case.

The data retention period is too long - Your streaming data is retained for up to 7 days. The data retention period is not an issue causing this error.

References:

https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html

https://docs.aws.amazon.com/streams/latest/dev/kinesis-using-sdk-java-resharding-strategies.html

Question 50: **Incorrect**

Your development team uses the AWS SDK for Java on a web application that uploads files to several Amazon Simple Storage Service (S3) buckets using the SSE-KMS encryption mechanism. Developers are reporting that they are receiving permission errors when trying to push their objects over HTTP. Which of the following headers should they include in their request?

- 'x-amz-server-side-encryption': 'SSE-S3'
- 'x-amz-server-side-encryption': 'AES256' (Incorrect)
- 'x-amz-server-side-encryption': 'aws:kms' (Correct)
- 'x-amz-server-side-encryption': 'SSE-KMS'

Explanation
Correct option:

'x-amz-server-side-encryption': 'aws:kms'

Server-side encryption is the encryption of data at its destination by the application or service that receives it. AWS Key Management Service (AWS KMS) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Amazon S3 uses AWS KMS customer master keys (CMKs) to encrypt your Amazon S3 objects. AWS KMS encrypts only the object data. Any object metadata is not encrypted.

If the request does not include the x-amz-server-side-encryption header, then the request is denied.


SSE-KMS Highlights

The highlights of SSE-KMS are as follows:

- You can choose a customer managed CMK that you create and manage, or you can choose an AWS managed CMK that Amazon S3 creates in your AWS

account and manages for you. Like a customer managed CMK, your AWS managed CMK is unique to your AWS account and Region. Only Amazon S3 has permission to use this CMK on your behalf. Amazon S3 only supports symmetric CMKs.

- You can create, rotate, and disable auditable customer managed CMKs from the AWS KMS console.
- The ETag in the response is not the MD5 of the object data.
- The data keys used to encrypt your data are also encrypted and stored alongside the data they protect.
- The security controls in AWS KMS can help you meet encryption-related compliance requirements.

Requiring Server-Side Encryption

To require server-side encryption of all objects in a particular Amazon S3 bucket, you can use a policy. For example, the following bucket policy denies upload object (`s3:PutObject`) permission to everyone if the request does not include the `x-amz-server-side-encryption` header requesting server-side encryption with SSE-KMS.

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyUnEncryptedObjectUploads",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::awsexamplebucket1/*",
         "Condition":{
            "StringNotEquals":{
               "s3:x-amz-server-side-encryption":"aws:kms"
            }
         }
      }
   ]
}
```

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html

Incorrect options:

'x-amz-server-side-encryption': 'SSE-S3' - This is an invalid header value. The correct value is 'x-amz-server-side-encryption': 'AES256'. This refers to Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3).

'x-amz-server-side-encryption': 'SSE-KMS' - Invalid header value. SSE-KMS is an encryption option.

'x-amz-server-side-encryption': 'AES256' - This is the correct header value if you are using SSE-S3 server-side encryption.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html

Question 51: **Correct**

A firm maintains a highly available application that receives HTTPS traffic from mobile devices and web browsers. The main Developer would like to set up the Load Balancer routing to route traffic from web servers to smart.com/api and from mobile devices to smart.com/mobile. A developer advises that the previous recommendation is not needed and that requests should be sent to api.smart.com and mobile.smart.com instead.

Which of the following routing options were discussed in the given use-case? (select two)

- [ ] Web browser version
- [x] Host based (Correct)
- [x] Path based (Correct)
- [ ] Client IP
- [ ] Cookie value

Explanation
Correct options:

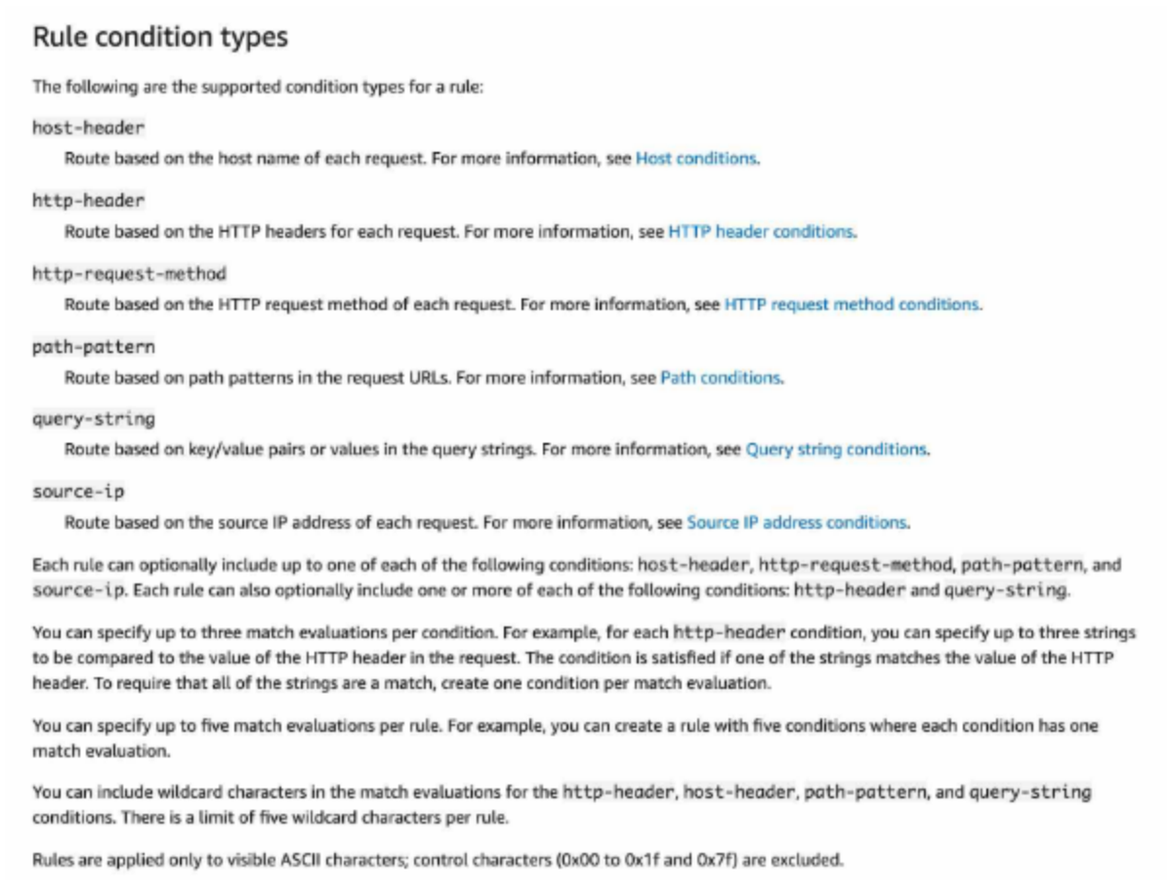
Path based

You can create a listener with rules to forward requests based on the URL path. This is known as path-based routing. If you are running microservices, you can route traffic to multiple back-end services using path-based routing. For example, you can route general requests to one target group and request to render images to another target group.

This path-based routing allows you to route requests to, for example, /api to one set of servers (also known as target groups) and /mobile to another set. Segmenting your traffic in this way gives you the ability to control the processing environment for each category of requests. Perhaps /api requests are best processed on Compute Optimized instances, while /mobile requests are best handled by Memory Optimized instances.

Host based

You can create Application Load Balancer rules that route incoming traffic based on the domain name specified in the Host header. Requests to api.example.com can be sent to one target group, requests to mobile.example.com to another, and all others (by way of a default rule) can be sent to a third. You can also create rules that combine host-based routing and path-based routing. This would allow you to route requests to api.example.com/production and api.example.com/sandbox to distinct target groups.



Rule condition types

The following are the supported condition types for a rule:

`host-header`
Route based on the host name of each request. For more information, see Host conditions.

`http-header`
Route based on the HTTP headers for each request. For more information, see HTTP header conditions.

`http-request-method`
Route based on the HTTP request method of each request. For more information, see HTTP request method conditions.

`path-pattern`
Route based on path patterns in the request URLs. For more information, see Path conditions.

`query-string`
Route based on key/value pairs or values in the query strings. For more information, see Query string conditions.

`source-ip`
Route based on the source IP address of each request. For more information, see Source IP address conditions.

Each rule can optionally include up to one of each of the following conditions: `host-header`, `http-request-method`, `path-pattern`, and `source-ip`. Each rule can also optionally include one or more of each of the following conditions: `http-header` and `query-string`.

You can specify up to three match evaluations per condition. For example, for each `http-header` condition, you can specify up to three strings to be compared to the value of the HTTP header in the request. The condition is satisfied if one of the strings matches the value of the HTTP header. To require that all of the strings are a match, create one condition per match evaluation.

You can specify up to five match evaluations per rule. For example, you can create a rule with five conditions where each condition has one match evaluation.

You can include wildcard characters in the match evaluations for the `http-header`, `host-header`, `path-pattern`, and `query-string` conditions. There is a limit of five wildcard characters per rule.

Rules are applied only to visible ASCII characters; control characters (0x00 to 0x1f and 0x7f) are excluded.

via - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html#rule-condition-types

Incorrect options:

Client IP - This option has been added as a distractor. Routing is not based on the client's IP address.

Web browser version - Routing has nothing to do with the client's web browser, if it was then there is something sneaky going on.

Cookie value - Application Load Balancers support load balancer-generated cookies only and you cannot modify them. When routing sticky sessions to route requests to the same target then cookies are needed to be supported by the client's browser.

Reference:

https://aws.amazon.com/blogs/aws/new-host-based-routing-support-for-aws-application-load-balancers/

Question 52: **Correct**

You are working on a project that has over 100 dependencies. Every time your AWS CodeBuild runs a build step it has to resolve Java dependencies from external Ivy repositories which take a long time. Your manager wants to speed this process up in AWS CodeBuild.

Which of the following will help you do this with minimal effort?

- (●) Cache dependencies on S3 (Correct)
- ( ) Ship all the dependencies as part of the source code
- ( ) Reduce the number of dependencies
- ( ) Use Instance Store type of EC2 instances to facilitate internal dependency cache

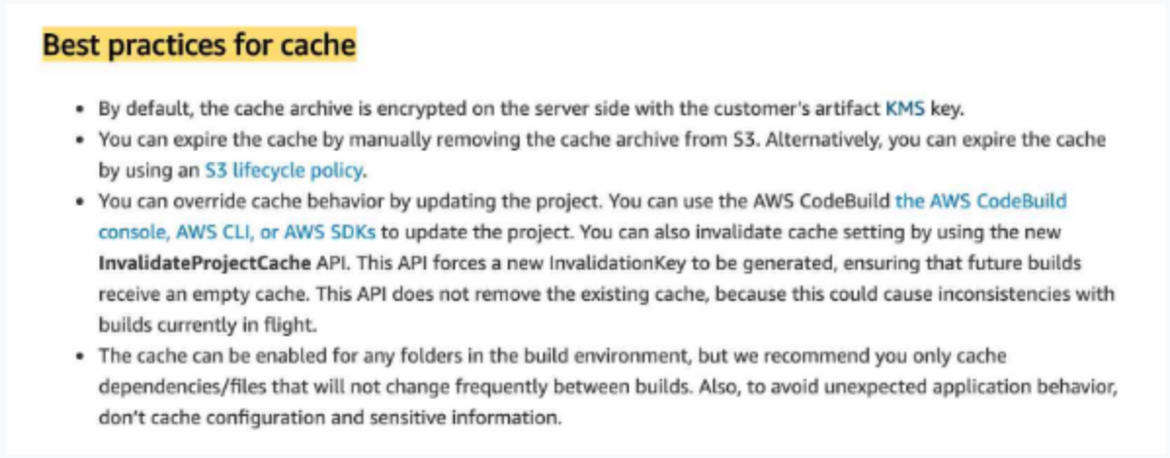
Explanation
Correct option:

Cache dependencies on S3

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your build servers.

Downloading dependencies is a critical phase in the build process. These dependent files can range in size from a few KBs to multiple MBs. Because most of the dependent files do not change frequently between builds, you can noticeably reduce your build time by caching dependencies in S3.

Best Practices for Caching Dependencies:



Best practices for cache

- By default, the cache archive is encrypted on the server side with the customer's artifact KMS key.
- You can expire the cache by manually removing the cache archive from S3. Alternatively, you can expire the cache by using an S3 lifecycle policy.
- You can override cache behavior by updating the project. You can use the AWS CodeBuild the AWS CodeBuild console, AWS CLI, or AWS SDKs to update the project. You can also invalidate cache setting by using the new **InvalidateProjectCache** API. This API forces a new InvalidationKey to be generated, ensuring that future builds receive an empty cache. This API does not remove the existing cache, because this could cause inconsistencies with builds currently in flight.
- The cache can be enabled for any folders in the build environment, but we recommend you only cache dependencies/files that will not change frequently between builds. Also, to avoid unexpected application behavior, don't cache configuration and sensitive information.

via - https://aws.amazon.com/blogs/devops/how-to-enable-caching-for-aws-codebuild/

Incorrect options:

Reduce the number of dependencies - This is ideal but sometimes you may not have control over this as your application needs those dependencies, so this option is ruled out.

Ship all the dependencies as part of the source code - This is not a good practice as doing this will increase your build time. If your dependencies are not changing then its best to cache them.

Use Instance Store type of EC2 instances to facilitate internal dependency cache - An instance store provides temporary block-level storage for your instance. This storage is located on disks that are physically attached to the host computer. Instance store is ideal for the temporary storage of information that changes frequently, such as buffers, caches, scratch data, and other temporary content, or for data that is replicated across a fleet of instances, such as a load-balanced pool of web servers.

Instance Store cannot be used to facilitate the internal dependency cache for the code build process.

Reference:

https://aws.amazon.com/blogs/devops/how-to-enable-caching-for-aws-codebuild/

Question 53: **Incorrect**

An Amazon Simple Queue Service (SQS) has to be configured between two AWS accounts for shared access to the queue. AWS account A has the SQS queue in its account and AWS account B has to be given access to this queue.

Which of the following options need to be combined to allow this cross-account access? (Select three)

- [x] The account B administrator creates an IAM role and attaches a trust policy to the role with account B as the principal **(Incorrect)**
- [ ] The account A administrator delegates the permission to assume the role to any users in account A
- [ ] The account A administrator attaches a trust policy to the role that identifies account B as the AWS service principal who can assume the role
- [ ] The account A administrator creates an IAM role and attaches a permissions policy **(Correct)**
- [x] The account A administrator attaches a trust policy to the role that identifies account B as the principal who can assume the role **(Correct)**
- [ ] The account B administrator delegates the permission to assume the role to any users in account B **(Correct)**

Explanation
Correct options:

The account A administrator creates an IAM role and attaches a permissions policy

The account A administrator attaches a trust policy to the role that identifies account B as the principal who can assume the role

The account B administrator delegates the permission to assume the role to any users in account B

To grant cross-account permissions, you need to attach an identity-based permissions policy to an IAM role. For example, the AWS account A administrator can create a role to grant cross-account permissions to AWS account B as follows:

1. The account A administrator creates an IAM role and attaches a permissions policy—that grants permissions on resources in account A—to the role.
2. The account A administrator attaches a trust policy to the role that identifies account B as the principal who can assume the role.
3. The account B administrator delegates the permission to assume the role to any users in account B. This allows users in account B to create or access queues in account A.

Incorrect options:

The account B administrator creates an IAM role and attaches a trust policy to the role with account B as the principal - As mentioned above, the account A administrator needs to create an IAM role and then attach a permissions policy. So, this option is incorrect.

The account A administrator delegates the permission to assume the role to any users in account A - This is irrelevant, as users in account B need to be given access.

The account A administrator attaches a trust policy to the role that identifies account B as the AWS service principal who can assume the role - AWS service principal is given as principal in the trust policy when you need to grant the permission to assume the role to an AWS service. The given use case talks about giving permission to another account. So, service principal is not an option here.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-overview-of-managing-access.html

Question 54: **Correct**

You are getting ready for an event to show off your Alexa skill written in JavaScript. As you are testing your voice activation commands you find that some intents are not invoking as they should and you are struggling to figure out what is happening. You included the following code `console.log(JSON.stringify(this.event))` in hopes of getting more details about the request to your Alexa skill.

You would like the logs stored in an Amazon Simple Storage Service (S3) bucket named `MyAlexaLog`. How do you achieve this?

- Use CloudWatch integration feature with Glue
- Use CloudWatch integration feature with Kinesis
- Use CloudWatch integration feature with Lambda
- Use CloudWatch integration feature with S3 (Correct)

Explanation
Correct option:

Use CloudWatch integration feature with S3

You can export log data from your CloudWatch log groups to an Amazon S3 bucket and use this data in custom processing and analysis, or to load onto other systems.

Exporting CloudWatch Log Data to Amazon S3:

Exporting Log Data to Amazon S3

PDF | Kindle | RSS

You can export log data from your log groups to an Amazon S3 bucket and use this data in custom processing and analysis, or to load onto other systems.

To begin the export process, you must create an S3 bucket to store the exported log data. You can store the exported files in your Amazon S3 bucket and define Amazon S3 lifecycle rules to archive or delete exported files automatically.

Exporting to S3 buckets that are encrypted with AES-256 is supported. Exporting to S3 buckets encrypted with SSE-KMS is not supported. For more information, see How Do I Enable Default Encryption for an S3 Bucket?.

You can export logs from multiple log groups or multiple time ranges to the same S3 bucket. To separate log data for each export task, you can specify a prefix that will be used as the Amazon S3 key prefix for all exported objects.

Log data can take up to 12 hours to become available for export. For near real-time analysis of log data, see Analyzing Log Data with CloudWatch Logs Insights or Real-time Processing of Log Data with Subscriptions instead.

via - https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/S3Export.html

Incorrect options:

Use CloudWatch integration feature with Kinesis - You can use both to do custom processing or analysis but with S3 you don't have to process anything. Instead, you configure the CloudWatch settings to send logs to S3.

Use CloudWatch integration feature with Lambda - You can use both to do custom processing or analysis but with S3 you don't have to process anything. Instead, you configure the CloudWatch settings to send logs to S3.

Use CloudWatch integration feature with Glue - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics. Glue is not the right fit for the given use-case.

Reference:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/S3Export.html

Question 55: **Correct**

A development team uses the AWS SDK for Java to maintain an application that stores data in AWS DynamoDB. The application makes use of `Scan` operations to return several items from a 25 GB table. There is no possibility of creating indexes to retrieve these items predictably. Developers are trying to get these specific rows from DynamoDB as fast as possible.

Which of the following options can be used to improve the performance of the Scan operation?

- ○ Use a ProjectionExpression
- ○ Use a Query
- ◉ Use parallel scans (Correct)
- ○ Use a FilterExpression

Explanation
Correct option:

Use parallel scans

By default, the Scan operation processes data sequentially. Amazon DynamoDB returns data to the application in 1 MB increments, and an application performs additional Scan operations to retrieve the next 1 MB of data. The larger the table or index being scanned, the more time the Scan takes to complete. To address these issues, the Scan operation can logically divide a table or secondary index into multiple segments, with multiple application workers scanning the segments in parallel.

To make use of a parallel Scan feature, you will need to run multiple worker threads or processes in parallel. Each worker will be able to scan a separate partition of a table concurrently with the other workers.

How DynamoDB parallel Scan works:

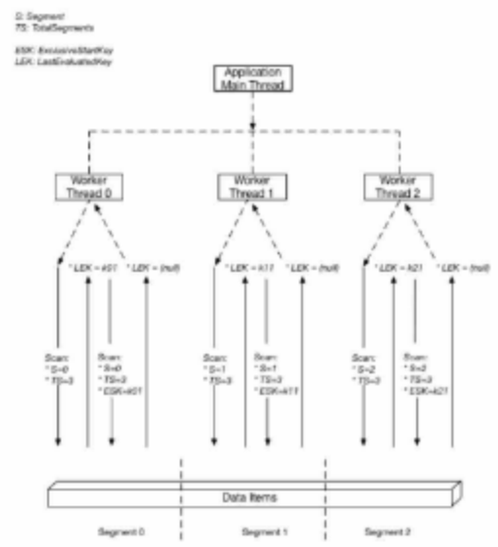
By default, the Scan operation processes data sequentially. Amazon DynamoDB returns data to the application in 1 MB increments, and an application performs additional Scan operations to retrieve the next 1 MB of data.

The larger the table or index being scanned, the more time the Scan takes to complete. In addition, a sequential Scan might not always be able to fully use the provisioned read throughput capacity: Even though DynamoDB distributes a large table's data across multiple physical partitions, a Scan operation can only read one partition at a time. For this reason, the throughput of a Scan is constrained by the maximum throughput of a single partition.

To address these issues, the Scan operation can logically divide a table or secondary index into multiple *segments*, with multiple application workers scanning the segments in parallel. Each worker can be a thread (in programming languages that support multithreading) or an operating system process. To perform a parallel scan, each worker issues its own Scan request with the following parameters:

- `Segment` – A segment to be scanned by a particular worker. Each worker should use a different value for `Segment`.
- `TotalSegments` – The total number of segments for the parallel scan. This value must be the same as the number of workers that your application will use.

The following diagram shows how a multithreaded application performs a parallel Scan with three degrees of parallelism.



In this diagram, the application spawns three threads and assigns each thread a number. (Segments are zero-based, so the first number is always 0.) Each thread issues a Scan request, setting `Segment` to its designated number and setting `TotalSegments` to 3. Each thread scans its designated segment, retrieving data 1 MB at a time, and returns the data to the application's main thread.

via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Scan.html#Scan.ParallelScan

Incorrect options:

Use a ProjectionExpression - A projection expression is a string that identifies the attributes you want. To retrieve a single attribute, specify its name. For multiple attributes, the names must be comma-separated

Use a FilterExpression - If you need to further refine the Scan results, you can optionally provide a filter expression. A filter expression determines which items within the Scan results should be returned to you. All of the other results are discarded.

A filter expression is applied after a Scan finishes, but before the results are returned. Therefore, a Scan consumes the same amount of read capacity, regardless of whether a filter expression is present.

Use a Query - This could work if we were able to create an index, but the question says: "There is no possibility of creating indexes to retrieve these items predictably". As such, we cannot use a Query.

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Scan.html#Scan.ParallelScan

Question 56: **Correct**

You have a web application hosted on EC2 that makes GET and PUT requests for objects stored in Amazon Simple Storage Service (S3) using the SDK for PHP. As the security team completed the final review of your application for vulnerabilities, they noticed that your application uses hardcoded IAM access key and secret access key to gain access to AWS services. They recommend you leverage a more secure setup, which should use temporary credentials if possible.

Which of the following options can be used to address the given use-case?

- ○ Use the SSM parameter store
- ○ Use environment variables
- ○ Hardcode the credentials in the application code
- ◉ Use an IAM Instance Role (Correct)

Explanation
Correct option:

Use an IAM Instance Role

An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts. The AWS SDK will use the EC2 metadata service to obtain temporary credentials thanks to the IAM instance role. This is the most secure and common setup when deploying any kind of applications onto an EC2 instance.

Incorrect options:

Use environment variables - This is another option if you configure AWS CLI on the EC2 instance. When configuring the AWS CLI you will set the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY environment variables. This practice may not be bad for one instance but once you start running more EC2 instances this is not a good practice because you may have to change credentials on each instance whereas an IAM Role gets temporary permissions.

Hardcode the credentials in the application code - It will work for sure, but it's not a good practice from a security point of view.

Use the SSM parameter store - With parameter store you can store data such as passwords. The problem is that you need the SDK to access parameter store and without credentials, you cannot use the SDK. Use parameter store for other uses such as database connection strings or other secret codes when you have already authenticated to AWS.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html

Question 57: **Correct**

A user has an IAM policy as well as an Amazon SQS policy that apply to his account. The IAM policy grants his account permission for the `ReceiveMessage` action on `example_queue`, whereas the Amazon SQS policy gives his account permission for the `SendMessage` action on the same queue.

Considering the permissions above, which of the following options are correct? (Select two)

- [ ] Adding only an IAM policy to deny the user of all actions on the queue is not enough. The SQS policy should also explicitly deny all action
- [ ] If the user sends a `SendMessage` request to `example_queue`, the IAM policy will deny this action
- [x] The user can send a `ReceiveMessage` request to `example_queue`, the IAM policy allows this action (Correct)
- [x] If you add a policy that denies the user access to all actions for the queue, the policy will override the other two policies and the user will not have access to `example_queue` (Correct)
- [ ] Either of IAM policies or Amazon SQS policies should be used to grant permissions. Both cannot be used together

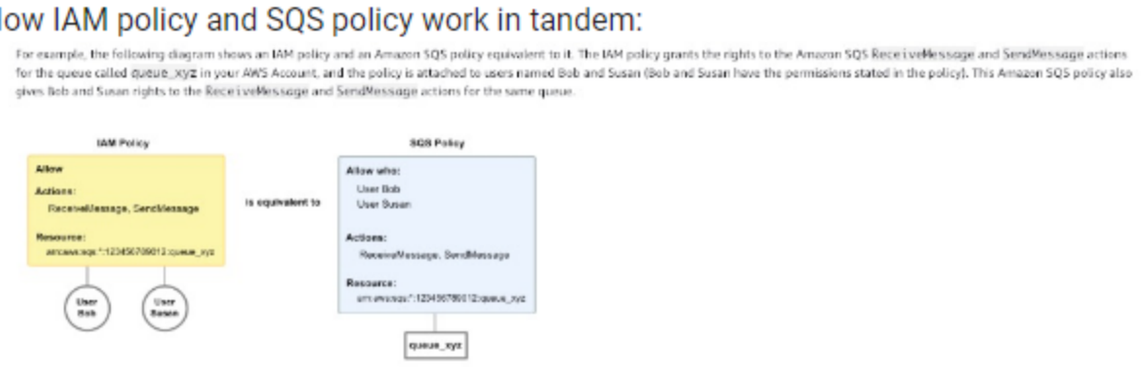
Explanation

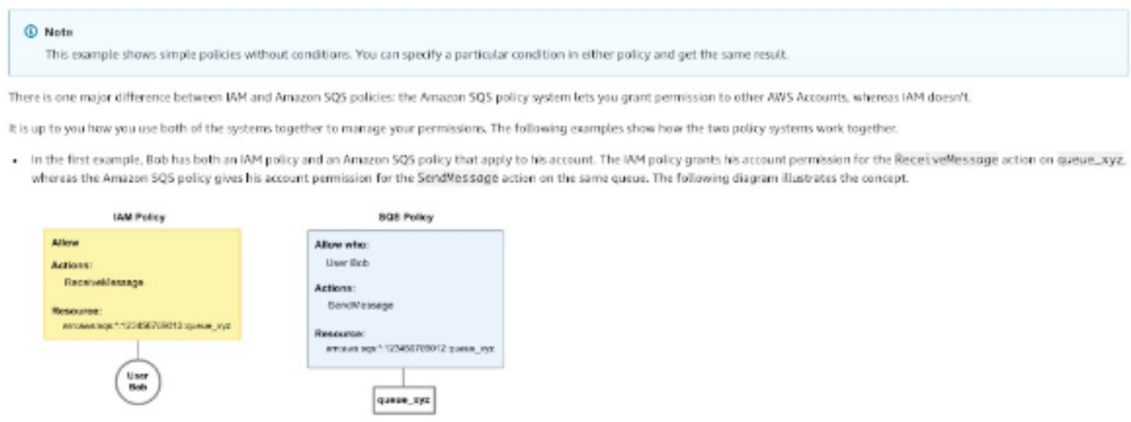
Correct options:

The user can send a `ReceiveMessage` request to `example_queue`, the IAM policy allows this action

The user has both an IAM policy and an Amazon SQS policy that apply to his account. The IAM policy grants his account permission for the `ReceiveMessage` action on `example_queue`, whereas the Amazon SQS policy gives his account permission for the `SendMessage` action on the same queue.

How IAM policy and SQS policy work in tandem:





via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-using-identity-based-policies.html

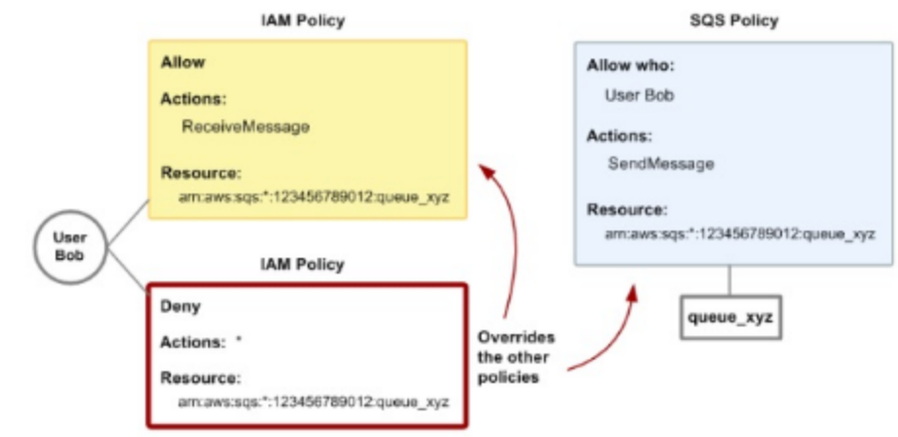
If you add a policy that denies the user access to all actions for the queue, the policy will override the other two policies and the user will not have access to `example_queue`

To remove the user's full access to the queue, the easiest thing to do is to add a policy that denies him access to all actions for the queue. This policy overrides the other two because an explicit deny always overrides an allow.

You can also add an additional statement to the Amazon SQS policy that denies the user any type of access to the queue. It has the same effect as adding an IAM policy that denies the user access to the queue.

How IAM policy and SQS policy work in tandem:

- In the second example, Bob abuses his access to `queue_xyz`, so it becomes necessary to remove his entire access to the queue. The easiest thing to do is to add a policy that denies him access to all actions for the queue. This policy overrides the other two because an explicit `deny` always overrides an `allow`. For more information about policy evaluation logic, see Using custom policies with the Amazon SQS Access Policy Language. The following diagram illustrates the concept.



You can also add an additional statement to the Amazon SQS policy that denies Bob any type of access to the queue. It has the same effect as adding an IAM policy that denies Bob access to the queue. For examples of policies that cover Amazon SQS actions and resources, see Basic examples of Amazon SQS policies. For more information about writing Amazon SQS policies, see Using custom policies with the Amazon SQS Access Policy Language.

via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-using-identity-based-policies.html

Incorrect options:

If the user sends a `SendMessage` request to `example_queue`, the IAM policy will deny this action - If the user sends a `SendMessage` request to `example_queue`, the Amazon SQS policy allows the action. The IAM policy has no explicit deny on this action, so it plays no part.

Either of IAM policies or Amazon SQS policies should be used to grant permissions. Both cannot be used together - There are two ways to give your users permissions to your Amazon SQS resources: using the Amazon SQS policy system and using the IAM policy system. You can use one or the other, or both. For the most part, you can achieve the same result with either one.

Adding only an IAM policy to deny the user of all actions on the queue is not enough. The SQS policy should also explicitly deny all action - The user can be denied access using any one of the policies. Explicit deny in any policy will override all other allow actions defined using either of the policies.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-using-identity-based-policies.html

Question 58: **Correct**

As a Full-stack Web Developer, you are involved with every aspect of a company's platform from development with PHP and JavaScript to the configuration of NoSQL databases with Amazon DynamoDB. You are not concerned about your response receiving stale data from your database and need to perform 16 eventually consistent reads per second of 12 KB in size each.

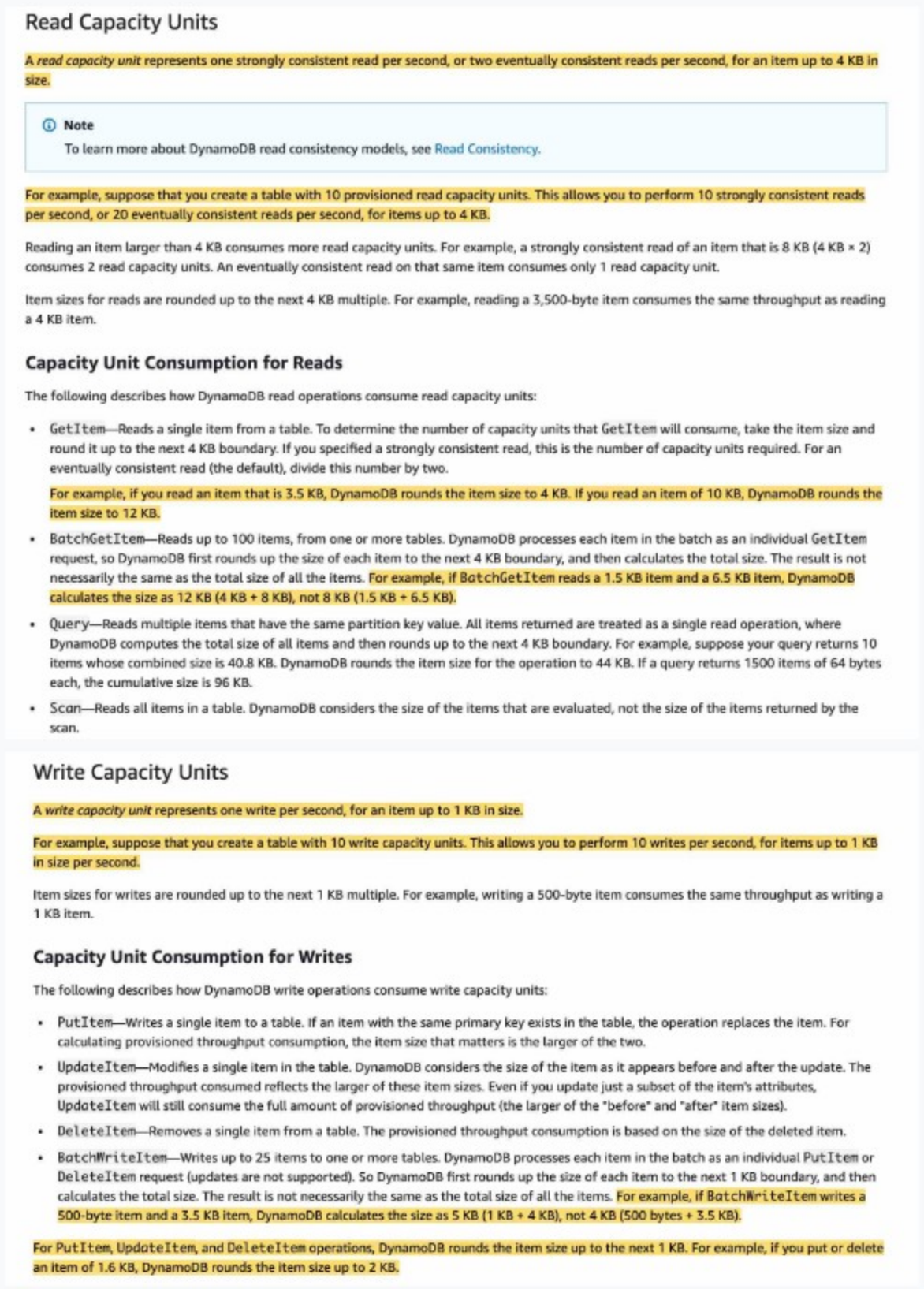
How many read capacity units (RCUs) do you need?

- (●) 24 (Correct)
- ( ) 12
- ( ) 192
- ( ) 48

Explanation

Correct option:

Before proceeding with the calculations, please review the following:



Read Capacity Units

A *read capacity unit* represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size.

Note
To learn more about DynamoDB read consistency models, see Read Consistency.

For example, suppose that you create a table with 10 provisioned read capacity units. This allows you to perform 10 strongly consistent reads per second, or 20 eventually consistent reads per second, for items up to 4 KB.

Reading an item larger than 4 KB consumes more read capacity units. For example, a strongly consistent read of an item that is 8 KB (4 KB × 2) consumes 2 read capacity units. An eventually consistent read on that same item consumes only 1 read capacity unit.

Item sizes for reads are rounded up to the next 4 KB multiple. For example, reading a 3,500-byte item consumes the same throughput as reading a 4 KB item.

Capacity Unit Consumption for Reads

The following describes how DynamoDB read operations consume read capacity units:

- `GetItem`—Reads a single item from a table. To determine the number of capacity units that `GetItem` will consume, take the item size and round it up to the next 4 KB boundary. If you specified a strongly consistent read, this is the number of capacity units required. For an eventually consistent read (the default), divide this number by two.

  For example, if you read an item that is 3.5 KB, DynamoDB rounds the item size to 4 KB. If you read an item of 10 KB, DynamoDB rounds the item size to 12 KB.
- `BatchGetItem`—Reads up to 100 items, from one or more tables. DynamoDB processes each item in the batch as an individual `GetItem` request, so DynamoDB first rounds up the size of each item to the next 4 KB boundary, and then calculates the total size. The result is not necessarily the same as the total size of all the items. For example, if `BatchGetItem` reads a 1.5 KB item and a 6.5 KB item, DynamoDB calculates the size as 12 KB (4 KB + 8 KB), not 8 KB (1.5 KB + 6.5 KB).
- `Query`—Reads multiple items that have the same partition key value. All items returned are treated as a single read operation, where DynamoDB computes the total size of all items and then rounds up to the next 4 KB boundary. For example, suppose your query returns 10 items whose combined size is 40.8 KB. DynamoDB rounds the item size for the operation to 44 KB. If a query returns 1500 items of 64 bytes each, the cumulative size is 96 KB.
- `Scan`—Reads all items in a table. DynamoDB considers the size of the items that are evaluated, not the size of the items returned by the scan.

Write Capacity Units

A *write capacity unit* represents one write per second, for an item up to 1 KB in size.

For example, suppose that you create a table with 10 write capacity units. This allows you to perform 10 writes per second, for items up to 1 KB in size per second.

Item sizes for writes are rounded up to the next 1 KB multiple. For example, writing a 500-byte item consumes the same throughput as writing a 1 KB item.

Capacity Unit Consumption for Writes

The following describes how DynamoDB write operations consume write capacity units:

- `PutItem`—Writes a single item to a table. If an item with the same primary key exists in the table, the operation replaces the item. For calculating provisioned throughput consumption, the item size that matters is the larger of the two.
- `UpdateItem`—Modifies a single item in the table. DynamoDB considers the size of the item as it appears before and after the update. The provisioned throughput consumed reflects the larger of these item sizes. Even if you update just a subset of the item's attributes, `UpdateItem` will still consume the full amount of provisioned throughput (the larger of the "before" and "after" item sizes).
- `DeleteItem`—Removes a single item from a table. The provisioned throughput consumption is based on the size of the deleted item.
- `BatchWriteItem`—Writes up to 25 items to one or more tables. DynamoDB processes each item in the batch as an individual `PutItem` or `DeleteItem` request (updates are not supported). So DynamoDB first rounds up the size of each item to the next 1 KB boundary, and then calculates the total size. The result is not necessarily the same as the total size of all the items. For example, if `BatchWriteItem` writes a 500-byte item and a 3.5 KB item, DynamoDB calculates the size as 5 KB (1 KB + 4 KB), not 4 KB (500 bytes + 3.5 KB).

For `PutItem`, `UpdateItem`, and `DeleteItem` operations, DynamoDB rounds the item size up to the next 1 KB. For example, if you put or delete an item of 1.6 KB, DynamoDB rounds the item size up to 2 KB.

via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ProvisionedThroughput.html

24

One read capacity unit represents **two** eventually consistent reads per second, for an item up to **4 KB** in size. So that means that for an item of 12KB in size, we need 3 RCU (12 KB / 4 KB) for **two** eventually consistent reads per second. As we need 16 eventually consistent reads per second, we need 3 * (16 / 2) = 24 RCU.

Incorrect options:

12

192

48

These three options contradict the details provided in the explanation above, so these are incorrect.

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ProvisionedThroughput.html

Question 59: **Correct**

A data analytics company with its IT infrastructure on the AWS Cloud wants to build and deploy its flagship application as soon as there are any changes to the source code.

As a Developer Associate, which of the following options would you suggest to trigger the deployment? (Select two)

- [x] Keep the source code in an AWS CodeCommit repository and start AWS CodePipeline whenever a change is pushed to the CodeCommit repository **(Correct)**
- [ ] Keep the source code in an Amazon EBS volume and start AWS CodePipeline whenever there are updates to the source code
- [ ] Keep the source code in Amazon EFS and start AWS CodePipeline whenever a file is updated
- [ ] Keep the source code in an Amazon S3 bucket and set up AWS CodePipeline to recur at an interval of every 15 minutes
- [x] Keep the source code in an Amazon S3 bucket and start AWS CodePipeline whenever a file in the S3 bucket is updated **(Correct)**

Explanation
Correct option:

Keep the source code in an AWS CodeCommit repository and start AWS CodePipeline whenever a change is pushed to the CodeCommit repository

Keep the source code in an Amazon S3 bucket and start AWS CodePipeline whenever a file in the S3 bucket is updated

AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define.

How CodePipeline Works:

via - https://aws.amazon.com/codepipeline/

Using change detection methods that you specify, you can make your pipeline start when a change is made to a repository. You can also make your pipeline start on a schedule.

When you use the console to create a pipeline that has a CodeCommit source repository or S3 source bucket, CodePipeline creates an Amazon CloudWatch Events rule that starts your pipeline when the source changes. This is the recommended change detection method.

If you use the AWS CLI to create the pipeline, the change detection method defaults to starting the pipeline by periodically checking the source (CodeCommit, Amazon S3, and GitHub source providers only). AWS recommends that you disable periodic checks and create the rule manually.

Start a pipeline execution in CodePipeline

PDF | Kindle | RSS

When a pipeline execution starts, it runs a revision through every stage and action in the pipeline.

There are two ways to start a pipeline execution in AWS CodePipeline:

- **Automatically**: Using change detection methods that you specify, you can make your pipeline start when a change is made to a repository. You can also make your pipeline start on a schedule. The following are the automatic change detection methods:
  - When you use the console to create a pipeline that has a CodeCommit source repository or S3 source bucket, CodePipeline creates an Amazon CloudWatch Events rule that starts your pipeline when the source changes. This is the recommended change detection method. If you use the AWS CLI to create the pipeline, the change detection method defaults to starting the pipeline by periodically checking the source (CodeCommit, Amazon S3, and GitHub source providers only). We recommend that you disable periodic checks and create the rule manually. For more information, see Use CloudWatch Events to start a pipeline (CodeCommit source).
  - When you use the console to create a pipeline that has a GitHub repository, CodePipeline creates a webhook that starts your pipeline when the source changes. This is the recommended change detection method. If you use the AWS CLI to create the pipeline, the change detection method defaults to starting the pipeline by periodically checking the source. We recommend that you disable periodic checks and create the webhook. For more information, see Use webhooks to start a pipeline (GitHub source).
  - Most source actions in CodePipeline, such as GitHub, require either a configured change detection resource (such as a webhook or CloudWatch Events rule) or use the option to poll the repository for source changes. For pipelines with a Bitbucket Cloud source action, you do not have to set up a webhook or default to polling. The connections action manages your source change detection for you.
- **Manually**: You can use the console or the AWS CLI to start a pipeline manually. For information, see Start a pipeline manually in AWS CodePipeline.

By default, pipelines are configured to start automatically using change detection methods.

> **Note**
> Your pipeline runs only when something changes in the source repository and branch that you have defined.

via - https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-about-starting.html

Incorrect options:

Keep the source code in Amazon EFS and start AWS CodePipeline whenever a file is updated

Keep the source code in an Amazon EBS volume and start AWS CodePipeline whenever there are updates to the source code

Both EFS and EBS are not supported as valid source providers for CodePipeline to check for any changes to the source code, hence these two options are incorrect.

Keep the source code in an Amazon S3 bucket and set up AWS CodePipeline to recur at an interval of every 15 minutes - As mentioned in the explanation above, although you could have the change detection method start the pipeline by periodically checking the S3 bucket, but this method is inefficient.

Reference:

https://docs.aws.amazon.com/codepipeline/latest/userguide/pipelines-about-starting.html

Question 60: **Incorrect**

A development team had enabled and configured CloudTrail for all the Amazon S3 buckets used in a project. The project manager owns all the S3 buckets used in the project. However, the manager noticed that he did not receive any object-level API access logs when the data was read by another AWS account.

What could be the reason for this behavior/error?

- ○ The meta-data of the bucket is in an invalid state and needs to be corrected by the bucket owner from AWS console to fix the issue
- ○ The bucket owner also needs to be object owner to get the object access logs **(Correct)**
- ◉ CloudTrail needs to be configured on both the AWS accounts for receiving the access logs in cross-account access **(Incorrect)**
- ○ CloudTrail always delivers object-level API access logs to the requester and not to object owner

Explanation
Correct option:

The bucket owner also needs to be object owner to get the object access logs

If the bucket owner is also the object owner, the bucket owner gets the object access logs. Otherwise, the bucket owner must get permissions, through the object ACL, for the same object API to get the same object-access API logs.

Incorrect options:

CloudTrail always delivers object-level API access logs to the requester and not to object owner - CloudTrail always delivers object-level API access logs to the requester. In addition, CloudTrail also delivers the same logs to the bucket owner only if the bucket owner has permissions for the same API actions on that object.

CloudTrail needs to be configured on both the AWS accounts for receiving the access logs in cross-account access

The meta-data of the bucket is in an invalid state and needs to be corrected by the bucket owner from AWS console to fix the issue

These two options are incorrect and are given only as distractors.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/cloudtrail-logging-s3-info.html#cloudtrail-object-level-crossaccount

Question 61: **Correct**

DevOps engineers are developing an order processing system where notifications are sent to a department whenever an order is placed for a product. The system also pushes identical notifications of the new order to a processing module that would allow EC2 instances to handle the fulfillment of the order. In the case of processing errors, the messages should be allowed to be re-processed at a later stage. The order processing system should be able to scale transparently without the need for any manual or programmatic provisioning of resources.

Which of the following solutions can be used to address this use-case?

- (●) SNS + SQS (Correct)
- ( ) SQS + SES
- ( ) SNS + Kinesis
- ( ) SNS + Lambda

Explanation
Correct option:

SNS + SQS

Amazon SNS enables message filtering and fanout to a large number of subscribers, including serverless functions, queues, and distributed systems. Additionally, Amazon SNS fans out notifications to end users via mobile push messages, SMS, and email.

How SNS Works:

via - https://aws.amazon.com/sns/

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

Because each buffered request can be processed independently, Amazon SQS can scale transparently to handle the load without any provisioning instructions from you.

SNS and SQS can be used to create a fanout messaging scenario in which messages are "pushed" to multiple subscribers, which eliminates the need to periodically check or poll for updates and enables parallel asynchronous processing of the message by the subscribers. SQS can allow for later re-processing and dead letter queues. This is called the fan-out pattern.

Incorrect options:

SNS + Kinesis - You can use Amazon Kinesis Data Streams to collect and process large streams of data records in real-time. Kinesis Data Streams stores records from 24 hours (by default) to 8760 hours (365 days). However, you need to manually provision shards in case the load increases or you need to use CloudWatch alarms to set up auto scaling for the shards. Since Kinesis does not support transparent scaling so this option is not the right fit for the given use case.

SNS + Lambda - Amazon SNS and AWS Lambda are integrated so you can invoke Lambda functions with Amazon SNS notifications. The Lambda function receives the message payload as an input parameter and can manipulate the information in the message, publish the message to other SNS topics, or send the message to other AWS services. However, your EC2 instances cannot "poll" from Lambda functions and as such, this would not work.

SQS + SES - This will not work as the messages need to be processed twice (once for sending the notification and later for order fulfillment) and SQS only allows for one consuming application.

References:

https://aws.amazon.com/sns/

https://aws.amazon.com/getting-started/tutorials/send-fanout-event-notifications/

Question 62: **Correct**

A company would like to migrate the existing application code from a GitHub repository to AWS CodeCommit.

As an AWS Certified Developer Associate, which of the following would you recommend for migrating the cloned repository to CodeCommit over HTTPS?

- ◉ Use Git credentials generated from IAM (Correct)
- ○ Use authentication offered by GitHub secure tokens
- ○ Use IAM user secret access key and access key ID
- ○ Use IAM Multi-Factor authentication

Explanation
Correct option:

Use Git credentials generated from IAM - CodeCommit repositories are Git-based and support the basic functionalities of Git such as Git credentials. AWS recommends that you use an IAM user when working with CodeCommit. You can access CodeCommit with other identity types, but the other identity types are subject to limitations.

The simplest way to set up connections to AWS CodeCommit repositories is to configure Git credentials for CodeCommit in the IAM console, and then use those credentials for HTTPS connections. You can also use these same credentials with any third-party tool or individual development environment (IDE) that supports HTTPS authentication using a static user name and password.

An IAM user is an identity within your Amazon Web Services account that has specific custom permissions. For example, an IAM user can have permissions to create and manage Git credentials for accessing CodeCommit repositories. This is the recommended user type for working with CodeCommit. You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

Authentication and access control for AWS CodeCommit:

Authentication

Because CodeCommit repositories are Git-based and support the basic functionality of Git, including Git credentials, we recommend that you use an IAM user when working with CodeCommit. You can access CodeCommit with other identity types, but the other identity types are subject to limitations, as described below.

Identity types:

- **IAM user** – An IAM user is an identity within your Amazon Web Services account that has specific custom permissions. For example, an IAM user can have permissions to create and manage Git credentials for accessing CodeCommit repositories. **This is the recommended user type for working with CodeCommit.** You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

  You can generate Git credentials or associate SSH public keys with your IAM user, or you can install and configure **git-remote-codecommit**. These are the easiest ways to set up Git to work with your CodeCommit repositories. With Git credentials, you generate a static user name and password in IAM. You then use these credentials for HTTPS connections with Git and any third-party tool that supports Git user name and password authentication. With SSH connections, you create public and private key files on your local machine that Git and CodeCommit use for SSH authentication. You associate the public key with your IAM user, and you store the private key on your local machine. **git-remote-codecommit** extends Git itself, and does not require setting up Git credentials for the user.

  In addition, you can generate access keys for each user. Use access keys when you access AWS services programmatically, either through one of the AWS SDKs or by using the AWS Command Line Interface (AWS CLI). The SDK and CLI tools use the access keys to cryptographically sign your requests. If you don't use the AWS tools, you must sign the requests yourself. CodeCommit supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the *AWS General Reference*.

via - https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control.html

Incorrect options:

Use IAM Multi-Factor authentication - AWS Multi-Factor Authentication (MFA) is a simple best practice that adds an extra layer of protection on top of your user name and password. With MFA enabled, when a user signs in to an AWS Management Console, they will be prompted for their user name and password (the first factor—what they know), as well as for an authentication code from their AWS MFA device (the second factor—what they have). Taken together, these multiple factors provide increased security for your AWS account settings and resources.

Use IAM user secret access key and access key ID - Access keys are long-term credentials for an IAM user or the AWS account root user. You can use access keys to sign programmatic requests to the AWS CLI or AWS API (directly or using the AWS SDK). As a best practice, AWS suggests using temporary security credentials (IAM roles) instead of access keys.

Use authentication offered by GitHub secure tokens - Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the GitHub API or the command line. This option is specific to GitHub only and hence not useful for the given use case.

References:

https://docs.aws.amazon.com/codecommit/latest/userguide/auth-and-access-control.html

https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html

Question 63: **Correct**

Your mobile application needs to perform API calls to DynamoDB. You do not want to store AWS secret and access keys onto the mobile devices and need all the calls to DynamoDB made with a different identity per mobile device.

Which of the following services allows you to achieve this?

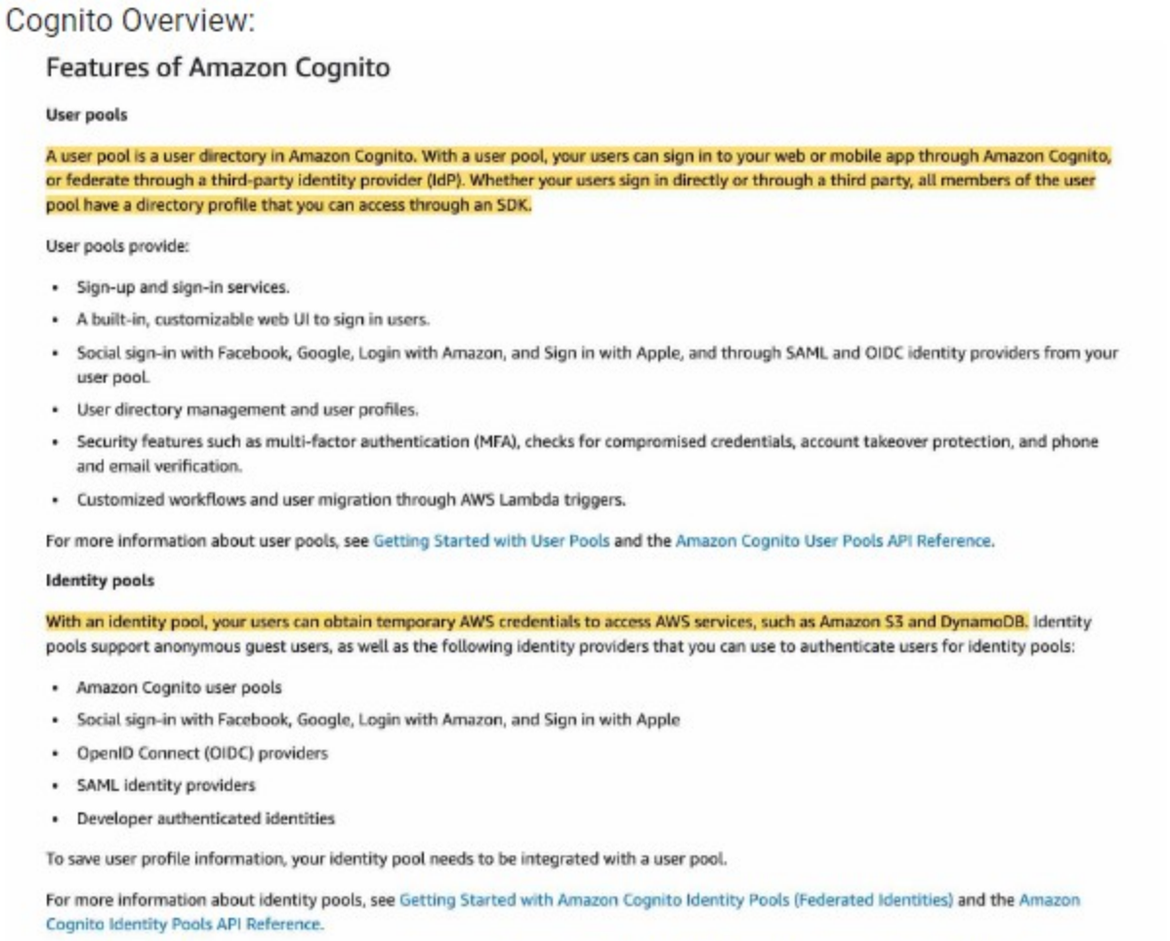
- (●) Cognito Identity Pools (Correct)
- ( ) IAM
- ( ) Cognito User Pools
- ( ) Cognito Sync

Explanation
Correct option:

"Cognito Identity Pools"

Amazon Cognito identity pools provide temporary AWS credentials for users who are guests (unauthenticated) and for users who have been authenticated and received a token. Identity pools provide AWS credentials to grant your users access to other AWS services.

Cognito Overview:



Features of Amazon Cognito

User pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, and through SAML and OIDC identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

For more information about user pools, see Getting Started with User Pools and the Amazon Cognito User Pools API Reference.

Identity pools

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the following identity providers that you can use to authenticate users for identity pools:

- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

To save user profile information, your identity pool needs to be integrated with a user pool.

For more information about identity pools, see Getting Started with Amazon Cognito Identity Pools (Federated Identities) and the Amazon Cognito Identity Pools API Reference.

via - https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Incorrect options:

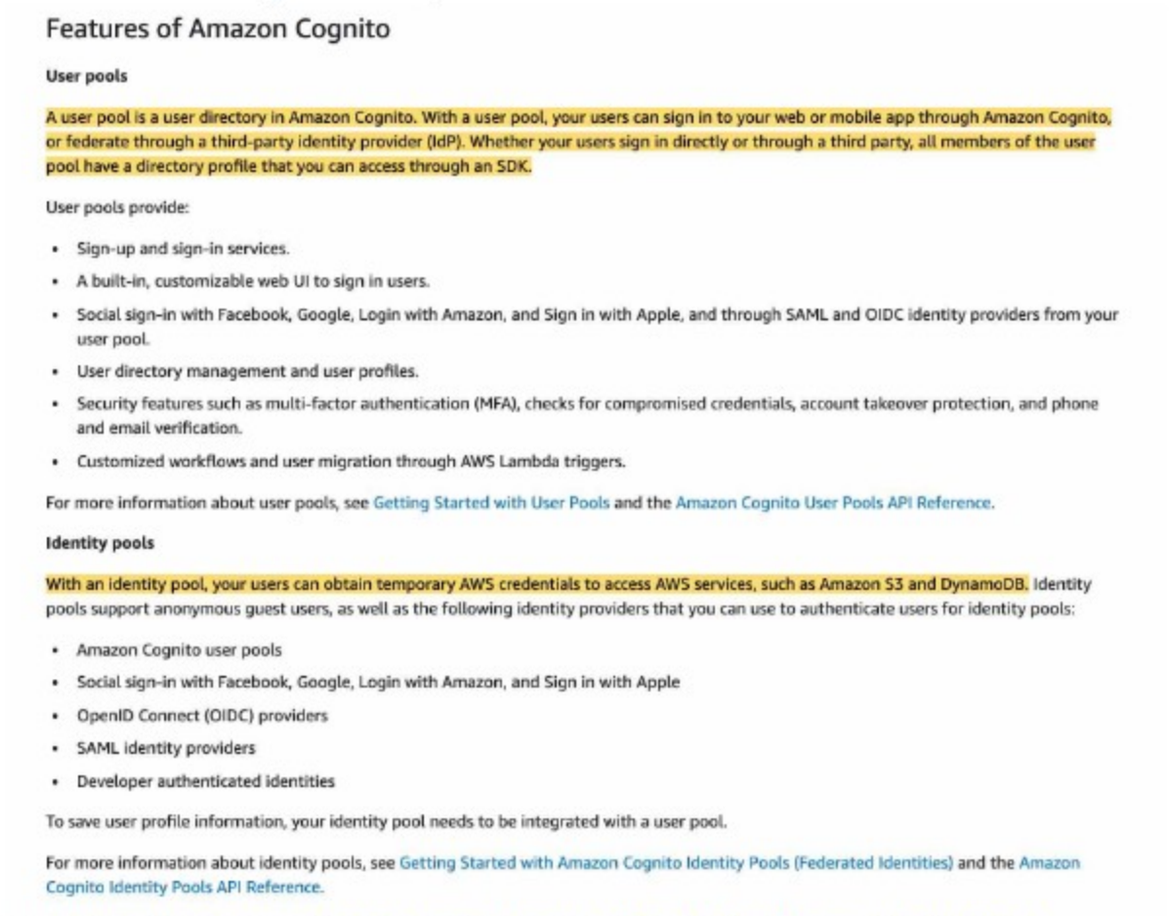
"Cognito User Pools" - AWS Cognito User Pools is there to authenticate users for your applications which looks similar to Cognito Identity Pools. The difference is that Identity Pools allows a way to authorize your users to use the various AWS services and User Pools is not about authorizing to AWS services but to provide add sign-up and sign-in functionality to web and mobile applications.

"Cognito Sync" - You can use it to synchronize user profile data across mobile devices and the web without requiring your own backend. The client libraries cache data locally so your app can read and write data regardless of device connectivity status.

"IAM" - This is not a good solution because it would require you to have an IAM user for each mobile device which is not a good practice or manageable way of handling deployment.

Exam Alert:

Please review the following note to understand the differences between Cognito User Pools and Cognito Identity Pools:



Features of Amazon Cognito

User pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, and through SAML and OIDC identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

For more information about user pools, see Getting Started with User Pools and the Amazon Cognito User Pools API Reference.

Identity pools

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the following identity providers that you can use to authenticate users for identity pools:

- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

To save user profile information, your identity pool needs to be integrated with a user pool.

For more information about identity pools, see Getting Started with Amazon Cognito Identity Pools (Federated Identities) and the Amazon Cognito Identity Pools API Reference.

via - https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Reference:

https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Question 64: **Incorrect**

A development team is considering Amazon ElastiCache for Redis as its in-memory caching solution for its relational database.

Which of the following options are correct while configuring ElastiCache? (Select two)

- [x] While using Redis with cluster mode enabled, you cannot manually promote any of the replica nodes to primary **(Correct)**
- [ ] While using Redis with cluster mode enabled, asynchronous replication mechanisms are used to keep the read replicas synchronized with the primary. If cluster mode is disabled, the replication mechanism is done synchronously
- [x] If you have no replicas and a node fails, you experience no loss of data when using Redis with cluster mode enabled **(Incorrect)**
- [ ] You can scale write capacity for Redis by adding replica nodes
- [ ] All the nodes in a Redis cluster must reside in the same region **(Correct)**

Explanation
Correct options:

All the nodes in a Redis cluster must reside in the same region

All the nodes in a Redis cluster (cluster mode enabled or cluster mode disabled) must reside in the same region.

While using Redis with cluster mode enabled, you cannot manually promote any of the replica nodes to primary

While using Redis with cluster mode enabled, there are some limitations:

1. You cannot manually promote any of the replica nodes to primary.
2. Multi-AZ is required.
3. You can only change the structure of a cluster, the node type, and the number of nodes by restoring from a backup.

Incorrect options:

While using Redis with cluster mode enabled, asynchronous replication mechanisms are used to keep the read replicas synchronized with the primary. If cluster mode is disabled, the replication mechanism is done synchronously - When you add a read replica to a cluster, all of the data from the primary is copied to the new node. From that point on, whenever data is written to the primary, the changes are asynchronously propagated to all the read replicas, for both the Redis offerings (cluster mode enabled or cluster mode disabled).

If you have no replicas and a node fails, you experience no loss of data when using Redis with cluster mode enabled - If you have no replicas and a node fails, you experience loss of all data in that node's shard, when using Redis with cluster mode enabled. If you have no replicas and the node fails, you experience total data loss in Redis with cluster mode disabled.

You can scale write capacity for Redis by adding replica nodes - This increases only the read capacity of the Redis cluster, write capacity is not enhanced by read replicas.

Reference:

https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/Replication.Redis.Groups.html

Question 65: **Correct**

An e-commerce company has implemented AWS CodeDeploy as part of its AWS cloud CI/CD strategy. The company has configured automatic rollbacks while deploying a new version of its flagship application to Amazon EC2.

What occurs if the deployment of the new version fails?

- ○ AWS CodePipeline promotes the most recent working deployment with a SUCCEEDED status to production

- The last known working deployment is automatically restored using the snapshot stored in Amazon S3
- CodeDeploy switches the Route 53 alias records back to the known good green deployment and terminates the failed blue deployment
- A new deployment of the last known working version of the application is deployed with a new deployment ID (Correct)

Explanation
Correct option:

A new deployment of the last known working version of the application is deployed with a new deployment ID

AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during deployment, and handles the complexity of updating your applications.

CodeDeploy rolls back deployments by redeploying a previously deployed revision of an application as a new deployment. These rolled-back deployments are technically new deployments, with new deployment IDs, rather than restored versions of a previous deployment.

To roll back an application to a previous revision, you just need to deploy that revision. AWS CodeDeploy keeps track of the files that were copied for the current revision and removes them before starting a new deployment, so there is no difference between redeploy and rollback. However, you need to make sure that the previous revisions are available for rollback.

Incorrect options:

The last known working deployment is automatically restored using the snapshot stored in Amazon S3 - CodeDeploy deployment does not have a snapshot stored on S3, so this option is incorrect.

AWS CodePipeline promotes the most recent working deployment with a SUCCEEDED status to production - The use-case does not talk about using CodePipeline, so this option just acts as a distractor.

CodeDeploy switches the Route 53 alias records back to the known good green deployment and terminates the failed blue deployment - The use-case does not talk about the blue/green deployment, so this option has just been added as a distractor.

Reference:

https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-rollback-and-redeploy.html